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

Preguntas Frecuentes sobre Seguridad en Aplicaciones Web (OWASP FAQ)

Enero 25, 2005

Autor: Sangita Pakala
Traducción: Alberto Pena Leiras
Edición: Juan Carlos Calderón Rojas

Introducción

1. ¿Sobre qué tratan estas preguntas frecuentes?

Esta lista de preguntas frecuentes da respuesta a algunas de las cuestiones que los desarrolladores suelen realizarse acerca de la seguridad en aplicaciones Web. Esta lista no es especifica para una plataforma o lenguaje en particular, discute los problemas y soluciones que son aplicables a cualquier plataforma.

2. ¿Cuáles son las amenazas más usuales en las aplicaciones Web?

A la hora de desarrollar una aplicación, generalmente nos centramos más en la funcionalidad que en la seguridad. Los atacantes se aprovechan de ello explotando la aplicación de diferentes maneras. Las amenazas más comunes en las aplicaciones Web son la inyección de código SQL, el ejecución inter-sitio (Cross Site Scripting), la manipulación de variables y la explotación de funcionalidad importante como el recordatorio de contraseñas, etcétera. Se han creado secciones separadas en esta lista para cada una de estas amenazas.

3. ¿Quién Desarrolló esta lista de preguntas frecuentes?

Esta lista es una documento que evoluciona constantemente con contribuciones de la comunidad de seguridad. Sangita Pakala y su equipo de Paladio Networks desarrollaron la primera versión de esta lista y mantienen esta pagina.

4. ¿Cómo puedo contribuir a esta lista?

Necesitamos sus opiniones y contribuciones para mejorar la lista. Nos encanta escuchar sobre:

- Nuevas preguntas para agregar a la lista
- Mejores respuestas a las preguntas actuales
- Nuevas ligas a documentos y herramientas
- Sugerencias para mejorar esta lista.

Puede enviar sus contribuciones a appsecfaq@owasp.org o mailto:owasp-spanish@lists.sourceforge.net

Sobre Inicio De Sesión (Login)

5. ¿Qué aspectos debo recordar a la hora de diseñar las páginas de inicio de sesión (login)?

- Desde la página de Inicio de sesión, el usuario deberá ser dirigido a una página de autenticación. Una vez autenticado, deberá enviarse al usuario a la siguiente página. Este tema se trata en la siguiente pregunta.
- La contraseña nunca debe enviarse en claro (sin cifrar) ya que podría ser robada con un rastreador(sniffer). Guardar la contraseña en claro en la base de datos también es peligroso. El mejor método para cifrar y enviar contraseñas es la técnica de cifrado MD5 con sal (Salted MD5).
- La mejor forma de gestionar sesiones es manejar una ficha (token) de sesión con dos valores, uno para antes de la autenticación y otro para después.

6. ¿Es realmente necesario redirigir al usuario a una nueva página después del inicio de sesión?

Consideremos que la aplicación cuenta con una página de entrada que envía al servidor el nombre de usuario y la contraseña mediante una petición POST. Si un usuario actualiza o refresca la segunda página (la que sigue a la de entrada), se enviará de nuevo la misma petición (incluyendo el nombre de usuario y contraseña). Ahora supongamos que un usuario válido entra en la aplicación navega a través de ella y después cierra la sesión pero no cierra la ventana. Un atacante podrá pulsar el botón "Atrás" en el navegador hasta llegar a la segunda página. Bastaría con que actualizase la página para que el nombre de usuario y la contraseña se reenviasen y validasen, completándose el proceso de inicio de sesión con las credenciales del usuario anterior.

Ahora supongamos que página de entrada de la aplicación envía el nombre de usuario y contraseña a una página intermedia de autenticación. Una vez autenticado, el usuario es redirigido a la siguiente página (la que antes era la segunda) con una ficha de sesión. En este caso, aunque el atacante consiga llegar ésta página y pulse en actualizar, el nombre de usuario y contraseña no serán reenviados. Esto ocurre porque la petición que se

reenviará en este caso será la correspondiente a la segunda página, que no contiene el nombre de usuario y contraseña. Por tanto, siempre es mejor redirigir al usuario.

7. ¿Cómo funciona la técnica MD5 con sal (salted-MD5)?

Esta técnica funciona de la siguiente forma: la base de datos almacena el hash MD5 de la contraseña (MD5 es una técnica de encripción de una vía que permite que el valor original no pueda recuperarse). Cuando el cliente hace una petición de la página de entrada, el servidor genera un número aleatorio, la sal, y lo envía al cliente junto con la página. Un código JavaScript de cliente se encarga de aplicar la función hash MD5 a la contraseña proporcionada por el usuario. El mismo código concatena la sal al resultado anterior y vuelve a aplicar MD5 al conjunto. El resultado se envía al servidor. El servidor toma el hash de la contraseña de la base de datos, le concatena la sal y le aplica de nuevo la función hash MD5. Si el usuario ha introducido correctamente la contraseña, los dos resultados deberían coincidir y, en tal caso, el usuario será autenticado.

8. ¿Como puede ser explotada mi función de recordatorio de contraseña?

La función para recordar contraseñas olvidadas puede ser implementada de muchas formas. Una forma muy común es hacer al usuario una pregunta "pista" a la cual el usuario ha enviado la respuesta en el proceso de registro. Las preguntas suelen ser del tipo: ¿Cuál es tu color favorito?, ¿Cuál es tu pasatiempo favorito?, etc. Una vez contestada la pregunta, el sistema bien muestra la contraseña correcta o bien ofrece otra temporal . En este método un atacante trata de robar la contraseña de un usuario puede ser capaz de adivinar la respuesta correcta a la pregunta e incluso reestablecer la contraseña.

9. En la función de recordar contraseña, ¿Es mejor mostrar la contraseña o permitir al usuario reestablecerla?

Si se muestra la contraseña antigua en la pantalla, esta podría ser vista por otros usuarios. Por tanto, es buena idea no mostrar en la pantalla la contraseña y permitir cambiar a una nueva. mas aún, para poder mostrar la contraseña, ésta deberá almacenarse de forma que se pueda recuperar, lo que no es aconsejable. Si la contraseña se guarda mediante un función hash

de una vía, la única forma de implementar una función de este tipo es permitiendo cambiar la contraseña antigua.
(un hash es el resultado obtenido cuando pasamos una cadena a una funcion hash de una vía. Es resultado es tal que es imposible regresar al valor original desde él. Las contraseñas son almacenadas mejor en la base de datos como hashes no recuperables)

10. ¿Existe riesgo en enviar la nueva contraseña al correo electrónico autorizado por el usuario?

Enviar la contraseña en claro puede ser arriesgado, ya que un atacante podría obtenerla con un rastreador (sniffer). Además, el correo electrónico que contiene la contraseña puede pasar mucho tiempo en el buzón del usuario y ser visto por un atacante.

11. ¿Cuál es la mejor forma de diseñar la función de recordatorio de contraseñas?

En primer lugar deberemos pedir al usuario que proporcione detalles personales o que responda a un pregunta ”pista”. A continuación deberá enviarse un correo electrónico al usuario incluyendo un enlace a una página en la que podrá reestablecer su contraseña. Este enlace deberá estar activo durante un corto periodo de tiempo, y deberá tener SSL implementado. De esta forma la contraseña real no podrá verse. Las ventajas son: la contraseña no se envía por correo electrónico; dado que el enlace está activo durante poco tiempo, no importa que el correo electrónico se almacene durante un tiempo prolongado en el buzón del usuario.

12. ¿Cómo me protejo ante ataques automatizados para adivinar contraseñas? ¿Debo simplemente bloquear el usuario después de 5 intentos fallidos?

La obtención automatizada de contraseñas resulta un importante problema ya que existe un gran número de herramientas disponibles para este propósito. estas herramientas, básicamente, intentan diferentes contraseñas hasta que una concuerda. Bloquear la cuenta de usuario después de 5 intentos fallidos es una buena defensa en contra de estas herramientas. Sin embargo, lo crucial será determinar cuánto tiempo se mantendrá bloqueada dicha cuenta. Si se bloquea durante demasiado tiempo, un atacante podrá denegar el servicio a usuarios válidos mediante el continuo bloqueo de su

cuenta. Si el tiempo es demasiado corto (por ejemplo 1 o 2 minutos), la herramienta de ataque podrá comenzar de nuevo tras ese periodo. El mejor método es insistir en la intervención humana después de un número de intentos fallidos. Un método muy utilizado consiste en mostrar al usuario una palabra aleatoria dentro de una imagen que deberá interpretar e introducir. Dado que esta interpretación no puede ser realizada por una herramienta, podemos frustrar ataques automatizados para adivinar la contraseña.

A continuación se incluyen dos enlaces a herramientas para adivinar contraseñas:

Brutus - http://www.hoobie.net/brutus/

WebCracker - http://www.securityfocus.com/tools/706

13. ¿Y si el atacante a plantado un registrador de tecleos (Keystroke logger) en la computadora cliente? ¿Puedo contrarestar esto?

Los registrador de tecleos infiltrados en las máquinas cliente pueden arruinar todos los esfuerzos por transmitir y almacenar contraseñas de forma segura. Los usuarios por sí mismos pueden no enterarse de que tienen instalado un programa que registra todas las teclas que se pulsan. Dado que el mayor riesgo concierne a la contraseña, si somos capaces de autenticar al usuario sin que tengan que usar el teclado, o sin que tengan que introducir la contraseña completa, el problema estará solucionado. Las formas de hacerlo son:

- Mostrando un teclado gráfico en el que los usuarios tengan que introducir los caracteres seleccionándolos con el ratón. Este resulta especialmente útil para PINs numéricos.
- Pidiendo al usuario que introduzca sólo una parte de la contraseña. Por ejemplo se le podría mostrar el mensaje "Por favor, introduce el primero, tercero y sexto caracteres de tu contraseña", haciendo esta regla aleatoria cada vez.

14. Mi sitio podrá usarse desde máquinas compartidas públicamente como en bibliotecas. ¿Qué precauciones debo tomar?

Si la aplicación va a ser usada desde lugares públicos como bibliotecas, se deberán tomar las siguiente precauciones:

- Deberemos asegurarnos de que las páginas no quedan en la memoria rápida (cache) de los navegadotes mediante las directivas de control de adecuadas.
- Deberá evitarse incluir información relevante o sensible en la URL, ya que ésta quedará guardada en el historial del navegador.
- Tener un teclado gráfico para escribir la contraseña o pedir al usuario que introduzca diferentes partes de la contraseña cada vez. Esto la protegerá de los registradores de tecleos (Keystroke loggers).
- Para evitar ataques con rastreadores (sniffers), debería usarse SSL o MD5 "con sal" para las contraseñas. Deberá borrarse la contraseña en claro de la memoria una vez aplicada la función MD5.

Inyección De SQL

15.¿Qué es la inyección de SQL?

Se trata de una técnica con la cual un atacante puede ejecutar sentencias SQL en la base de datos mediante la manipulación de la entrada proporcionada a la aplicación. Entendamos esto con un ejemplo de página de entrada de una aplicación Web en la que la base de datos se encuentra en un servidor SQL Server. El usuario necesita introducir su nombre de usuario y contraseña en la página Login.asp. Supongamos que el usuario introduce lo siguiente: nombre de usuario 'Obelix' y contraseña 'Dogmatix'.

Estos datos de entrada son usados para construir dinámicamente la sentencia SQL que tendrá el siguiente aspecto:

SELECT * FROM Users WHERE username= 'Obelix' and password='Dogmatix'

Esta consulta devolvería el registro de la tabla "User" que cumple las condiciones dadas. El usuario se considerará autenticado en el caso de que la consulta devuelva uno o más registros. Supongamos que una atacante teclea lo siguiente en la página de entrada:

Nombre de Usuario: ` **or 1=1--**

La sentencia que se construirá será la siguiente:

SELECT * FROM Users WHERE username='' or 1=1-- and password=''

-- indica a SQL Server que lo que viene a continuación son comentarios. De forma que lo que ejecutará SQL Server será:

SELECT * FROM Users WHERE username='' or 1=1

Esta sentencia lo que hará es buscar en la tabla aquellos registros cuyo nombre de usuario es nulo o cuando se cumpla que 1 es igual a 1 (lo que se cumplirá siempre). Por tanto la consulta devolverá todos los registros de la tabla y el usuario se habrá autenticado sin usuario ni contraseña.

En la siguiente dirección se puede consultar más información al respecto:

http://www.securiteam.com/securityreviews/5DP0N1P76E.html

¿Sólo son ASP y SQL Server vulnerables a este tipo de ataques?

Todas las plataformas son vulnerables a la inyección de SQL. Una inadecuada validación de la entrada y la creación dinámica de las sentencias SQL son las condiciones que permiten este tipo de ataques. La sintaxis a usar para la inyección de SQL dependerá del gestor de base de datos usado. A lo largo de nuestras auditorias hemos encontrado muchas aplicaciones vulnerables corriendo con otros gestores de bases de datos. El ejemplo anterior podría funcionar en SQL Server, Oracle y MySQL, lo que demuestra que el origen del problema no es el tipo de base de datos sino una inadecuada validación de la entrada de usuario junto con la generación dinámica de sentencias SQL.

16. Además del usuario y contraseña, ¿qué otras variables son candidatas para la inyección de SQL en nuestras aplicaciones?

Cualquier campo de entrada que participa en la creación de la cláusula WHERE de una consulta a una base de datos. Por ejemplo: números de cuentas, números de tarjetas de crédito, etc, en el caso de banca en línea. Además de los campos de un formulario, un atacante puede usar campos ocultos y parámetros del URL para realizar la inyección de comandos

17. ¿Cómo evitamos ataques de inyección de SQL?

Es sencillo proteger una aplicación de la inyección de SQL durante su desarrollo. Deberá chequearse toda la información que provenga del cliente antes de construir una sentencia SQL. El mejor método es eliminar toda entrada no deseada y aceptar tan sólo la esperada.

Aunque la validación de la entrada en el servidor es el método más efectivo, otro método de prevención es evitar el uso de SQL dinámicas. Esto se puede lograr mediante el uso de procedimientos almacenados y parámetros. Para aplicaciones escritas en Java, se pueden usar las sentencias del tipo CallableStatements y PreparedStatements. Para aplicaciones ASP, se puede usar el objeto Command de ADO.

En el siguiente artículo se da más información sobre la inyección SQL en Oracle:

http://www.integrigy.com/info/IntegrigyIntrotoSQLInjectionAttacks.pdf

18. Estoy usando validación de la entrada mediante JavaScripts de Cliente, ¿no es esto suficiente?

No. A pesar de que la validación de la entrada en el cliente impide que un atacante introduzca código malicioso directamente sobre los campos de entrada, solo esto no es suficiente para evitar ataques de Inyección de SQL. Los Scripts de cliente sólo validan la entrada en el navegador. Pero esto no es garantía de que la información será la misma cuando llegue al servidor. Existen herramientas que pueden capturar la información que viaja desde el navegador al servidor y que pueden manipular dicha información antes de ser enviada al servidor. El atacante podría además introducir comandos en las variables de URL, ya que estas no son verificadas por los Scripts de cliente.

19. Estoy usando procedimientos almacenados para la autenticación, ¿soy vulnerable?

No. El uso de procedimientos almacenados evita la inyección de comandos SQL ya que la entrada del usuario ya no es usada para la construcción dinámica de una sentencia SQL. Dado que un procedimiento almacenado es una colección de sentencias SQL precompiladas y que acepta la entrada como parámetros, se evita la generación dinámica de la consulta. Aunque la entrada es puesta en la consulta precompilada tal cual, dado que la consulta en sí está en un formato diferente. No tiene el efecto de cambiar la consulta como se esperaría. Usando procedimientos almacenados estamos permitiendo a la base de datos que manipule la ejecución de la consulta en ves de pedirle que ejecute una consulta ya construida.

20. ¿Los Servlets de Java son vulnerables a la inyección de SQL?

Sí, lo son si la entrada de usuario no es revisada adecuadamente y las sentencias SQL se construyen dinámicamente. Los Servlets de Java también cuentan con características que evitan la inyección de SQL, como CallableStatements y PreparedStatements. Igual que los procedimientos almacenados y las variables enlazadas, evitan la generación dinámica de las sentencias SQL.

Manipulación De Variables

21.¿Por qué no puedo fiarme de lo que proviene del navegador?

Hay ocasiones en las que la información es modificada antes de que llegue al servidor. Los atacantes que naveguen por la aplicación pueden modificar la información de una petición POST o GET. Existe un gran número de herramientas como "Aquiles" que son capaces de interceptar la información y que permiten al atacante manipularla. Además, la información que el usuario envía o ve a través de una página Web, tiene que viajar a través de Internet antes de llegar al servidor. Aunque el cliente y el servidor puedan ser confiables, no podemos asegurar que la información no ha sido modificada tras ser enviada por el navegador. Los atacantes podrían capturar la información en tránsito y modificarla.

22.¿Qué información puede ser manipulada por un atacante?

La manipulación de las variables en las URLs es muy sencilla. Pero los atacantes también pueden modificar la información contenida en campos de formularios y campos ocultos.

23.¿Cómo manipulan los atacantes la información? ¿Qué herramientas usan?

Para manipular la información, incluyendo campos de formularios, campos ocultos y galletas (cookies), los atacantes usan herramientas conocidas como proxys HTTP. Una vez que el navegador está configurado para funcionar a través del proxy HTTP, la herramienta puede ver toda la información que fluye entre el cliente y el servidor, permitiendo incluso modificar cualquier información antes de ser enviada. Algunas de estas herramientas son:

WebScarab, puede bajarse en www.owasp.org

Achilles. Puede bajarse en http://www.mavensecurity.com/achilles

Odysseus, puede encontrarse en:
http://www.wastelands.gen.nz/odysseus/index.php

24.Estoy usando SSL, ¿pueden los atacantes modificar la información?

SSL proporciona seguridad en dos aspectos. Primero cuando el cliente se conecta al servidor, puede estar seguro que esta comunicándose con el

servidor correcto verificando el certificado que el servidor envía. Segundo, SSL asegura la confidencialidad de los datos, dado que el cliente y el servidor intercambian mensajes encriptados que no pueden ser entendidos por nadie mas.

Aquí esta como funciona SSL: cuando el cliente pide una pagina SSL, el servidor envía un certificado que es obtenido de una autoridad certificadora confiable. El certificado contienen la llave pública del servidor. Después de asegurarse que el certificado es correcto y que el servidor es genuino, el cliente genera un número aleatorio, la llave de sesión. La llave es encriptada con la llave publica del servidor y enviada. El servidor desencripta el mensaje con su llave privada. Ahora ambos lados tienen una llave de sesión conocida solamente por ellos dos. Toda comunicación desde y hacia ellos es encriptada y desencriptada con la llave de sesión.

A pesar de que SSL proporciona un buen nivel de seguridad, no es suficiente. Todas las herramientas mencionadas anteriormente pueden modificar información incluso cuando el sitio esta corriendo sobre SSL. Veamos cómo trabaja Achiles con SSL. Achiles cuenta con un certificado falso con un par de llaves generadas por él mismo. Cuando el cliente realizar una petición con SSL, Achiles la envía tal cual al servidor. Entonces el servidor envía como respuesta su certificado con su llave pública . Achiles lo intercepta, genera una llave de sesión y la manda al servidor cifrada con la llave pública del servidor. De esta forma consigue establecer una conexión SSL con el servidor. Entonces, envía a la parte cliente su propio certificado y llave pública. El navegador cliente mostrará un mensaje diciendo que el certificado no es de confianza y preguntará al usuario si quiere aceptarlo, pero ya que se trata del navegador del atacante, éste acepta el certificado. Entonces el cliente genera un llave de sesión, la cifra con la llave pública de Achiles y la envía. De esta forma ahora Achiles ha establecido dos conexiones SSL – una con el servidor y otra con el cliente. Descifra la información que proviene del servidor y se la muestra en texto plano al atacante y luego la cifra de nuevo con la llave del cliente y la envía . Para el tráfico en la otra dirección utiliza el mismo método.

25. ¿Hay alguna forma de evitar que este tipo de herramienta proxy modifiquen los datos enviados a la aplicación?

La mayor amenaza que representan este tipo de herramientas es la manipulación de la información que viaja del cliente al servidor. Una forma de protegerse es firmando los mensajes enviados desde el cliente mediante un Applet Java cargado en la máquina cliente. Dado que este applet será el encargado de validar el certificado y no el navegador, una herramienta proxy no será capaz de intervenir con un certificado falso. La llave publica del certificado puede entonces ser usada para firmar digitalmente cada mensaje enviado entre el cliente y el servidor. Un atacante tendría que reemplazar el certificado en el applet con un certificado falso, esto incrementa la dificultad para el atacante.

Memoria Rápida (Cache) Del Navegador

26.¿Cómo puede usarse la memoria rápida (cache) del navegador para realizar ataques?

Los navegadores tienen la capacidad de almacenar temporalmente las páginas visitadas. Estos archivos se almacenan en una carpeta (la carpeta Archivos Temporales de Internet en el caso de Internet Explorer). Cuando solicitamos estas páginas de nuevo, el navegador las obtiene de dicha carpeta, lo que resulta mucho más rápido que volver a descargarlas del servidor. Consideremos un escenario en el que un usuario ha realizado el inicio de sesión en una aplicación Web mediante un nombre de usuario y contraseña; el usuario visita diferentes páginas que contienen información sensible, el usuario cierra la sesión. Supongamos el navegador guarda en memoria rápida un página con información de la tarjeta de crédito del usuario. Supongamos que un atacante tiene acceso a la misma máquina y busca entre los archivo temporales de Internet entonces accedería a el número de tarjeta de crédito. El atacante no necesita saber el nombre de usuario y contraseña para acceder a la información de la tarjeta de crédito del usuario.

27.¿Cómo puede asegurarme de que las páginas con información sensible no se guardan en memoria rápida?

La encabezados en la respuesta enviada por el servidor incluye algunas directivas de control de memoria rápida que pueden ser configuradas mediante código. Esta directivas controlan los contenidos que el navegador del cliente guarda en memoria rápida. Las directivas a configurar son cache-control : no-cache ó cache-control : no-store.

28.¿Qué diferencia hay entre las directivas de control de memoria rápida "no-cache" y "no-store"?

La directiva "no-cache" indica que la respuesta del servidor no puede ser usada para servir a una nueva petición, es decir, la memoria rápida no mostrará una respuesta que incluya esta directiva y deberá ser el servidor quien la proporcione. La directiva "no-cache" puede incluir algunos nombres de campos, en este caso la respuesta podrá ser mostrada desde la memoria rápida exceptuando aquellos campos que deberán ser proporcionados por el servidor.

La directiva "no-store" se aplica al mensaje completo e indica que no se podrá almacenar en memoria rápida ninguna parte de la respuesta o petición que la generó.

29.¿Estoy totalmente seguro con estas directivas?

Estas directivas resuelven el problema hasta cierto punto, ya que no están soportadas por las caches HTTP 1.0.

30.¿Dónde puedo encontrar más información sobre caching?

Algunos enlaces útiles son:

Caching Tutorial for Web Authors and Webmasters por Mark Nottingham en http://www.mnot.net/cache_docs/

RFC de HTTP en:
http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html - sec14.9.1

Ejecución Inter-Sitio (Cross Site Scripting o XSS[1])

31.¿Qué es el XSS?

La ejecución inter-sitio es un tipo de ataque que puede llevarse a cabo para robar información sensible perteneciente a los usuarios de un sitio Web. Esta confía en que el servidor reflejará la entrada del usuario sin verificar si existe JavaScript incrustado. Esto puede ser usado para robar galletas (Cookies) e identificadores de sesión. Veamos cómo funciona, a menudo nos

[1] Se usan las siglas XSS para evitar confusiones con las hojas de estilo CSS

encontramos con la siguiente situación: escribimos una URL en el navegador, supongamos que es www.abcd.com/mypage.asp, y recibimos una página de error que nos dice que la página "www.abcd.com/mypage.asp" no existe. En otras palabras, la página muestra la información tecleada por el usuario en el navegador. Este tipo de páginas pueden ser explotadas mediante XSS.

Pensemos en qué ocurriría si la entrada contiene un script. En este caso, al devolver dicha entrada al navegador, éste la interpretará no como HTML normal sino como lo que es, un script, ejecutando por tanto los comandos que incluya, pudiendo incluir código maligno.

Un atacante podría enviar a un usuario un enlace que contenga un script como parte de la URL. Cuando el usuario haga clic en el enlace, el script se ejecuta en el navegador del usuario. Dicho script puede hacer sido escrito para enviar al atacante información importante del usuario.

El artículo "Ejecución inter-sitio, ¿son vulnerables tus aplicaciones?" es una buena fuente de información:
http://www.spidynamics.com/whitepapers/SPIcross-sitescripting.pdf

32.¿Qué información puede robar un atacante mediante XSS?

Un atacante puede robar el identificador de sesión de un usuario válido mediante *XSS*. El identificador de sesión es muy valioso ya que es un elemento secreto que el usuario presenta a una aplicación tras hacer un *inicio de sesión* y hasta que la cierra. Si el identificador de sesión se guarda en una *galleta*, el atacante puede escribir un *script* que se ejecutará en el navegador del usuario y que consultará el valor de la *galleta* para luego enviárselo. El atacante podrá entonces usar el identificador de sesión válido para entrar en la aplicación sin pasar por el proceso de *inicio de sesión*. El *script* podría además recoger otra información de la página, incluyendo todo su contenido.

33.Aparte de las enlaces a páginas de error enviadas por correo electrónico, ¿hay otros métodos para realizar ataques de XSS?

Sí, hay otros métodos. Pongamos el caso de una aplicación de tablero de anuncios (bulletin board) que contiene páginas en las que lo escrito por un

usuario puede ser consultado por el resto, el atacante introduce un *script* en esta página, cuando un usuario válido intenta ver la página, el *script* se ejecuta en el navegador del usuario y podrá enviar información al atacante.

34.¿Cómo puedo evitar ataques XSS?

Este tipo de ataques se pueden evitar durante el desarrollo (codificación) de la aplicación. Deberá validarse toda la información que entre y salga de y desde la aplicación y "escapar" todos los caracteres especiales que puedan ser usados en un script. Si el código reemplaza los caracteres especiales antes de mostrarlos (según la tabla siguiente), la aplicación estará en gran medida protegida.

<	&lt;
>	&gt;
(	&#40;
)	&#41;
#	&#35;
&	&#38;

35.¿Me puedo proteger de ataques XSS sin modificar el código de la aplicación?

Existe un método que requiere un codificación mínima comparado con implementar la validación completa de la entrada y salida, y que evita los robos de galletas mediante XSS. El navegador Internet Explorer 6 cuenta con un atributo llamado HTTP Only que se puede configurar para las galletas. El uso de este atributo asegurar que las galletas no puedan ser accedidas por scripts.
http://msdn.microsoft.com/library/default.asp?url=/workshop/author/dhtml/httponly_cookies.asp
Mozilla también planea implementar una función similar.

Los investigadores han encontrado un método para saltarse esta función, es el método conocido como rastreo inter-sitio (Cross Site Tracing).

36.¿Qué es el rastreo inter-sitio (Cross Site Tracing ó XST)?¿Cómo puedo evitarlo?

Mediante XST, los atacantes pueden saltarse el atributo *HTTP Only* para robar la información de una *galleta*. *TRACE* es un método de *HTTP* que puede ser enviado al servidor. El servidor envía de vuelta al navegador cualquier información incluida en la petición *TRACE*. En un sitio que usa *galletas*, la información de la *galleta* es enviada al servidor en cada petición. Si enviamos una petición *TRACE* a través en una *URL*, el servidor devolverá toda la información de la *galleta* al navegador. Supongamos ahora una situación similar a la descrita para el caso de *XSS* pero en la que el sitio Web está usando ahora *galletas* de tipo *HTTP Only*. El atacante construye un enlace válido en el que ha incluido un *script* que llama al método *TRACE*. Cuando el usuario hace clic en el enlace, envía al servidor además de la petición *TRACE*, la información de la *galleta*. El servidor entonces devuelve la información de la *galleta* al *script* del navegador. Supongamos que el script contiene además código para enviar la información a los atacantes. Los atacantes han conseguido de esta forma robar la información de galletas HTTP Only. En resumen, HTTP Only evita el acceso directo a las galletas de los JavaScripts, lo que el atacante puede evadir mediante este método de acceso indirecto.

XST puede evitarse deshabilitando el método TRACE en el servidor Web.

El siguiente artículo de Jeremiah Grossman discute en detalle los ataques XST: http://www.cgisecurity.com/whitehat-mirror/WhitePaper_screen.pdf

Identificación Del Servidor Web

37.¿Cómo identifican los atacantes qué servidor Web estoy usando?

Identificar la aplicación que está ejecutándose en el servidor Web se conoce como "*Server Fingerprinting*". La forma más sencilla de hacerlo es enviando una petición al servidor y observando el *banner* (encabezado) enviado como respuesta. Los *banners* generalmente incluyen el nombre del servidor y la versión. Podemos evitar este problema configurando el servidor de forma que no muestra el *banner* o cambiándolo para que muestre otra cosa diferente.

38.¿Cómo puedo falsificar los banners o rescribir las cabeceras desde mi servidor Web?

Existe una serie de herramientas que ayudan a falsificar banners:

URLScan permite cambiar el banner de un servidor Web IIS
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/tools/URLScan.asp

mod_security incluye una función que permite cambiar la identidad de un servidor Apache. Puede ser encontrada en http://www.modsecurity.org/

Servermask permite falsificar banners de IIS. Puede ser encontrada en http://www.servermask.com/

39.Una vez falsificado el banner,¿puede mi servidor Web ser identificado?

Sí, desafortunadamente existen herramientas que identifican al servidor sin depender de los banners. Los diferentes servidor Web pueden implementar de forma diferente funciones no especificadas en las HTTP RFCs. Supongamos que hacemos una base de datos con esas peticiones (o funciones) especiales y las respuestas que da cada tipo de servidor. Entonces podremos enviar peticiones al servidor que queremos identificar y compara sus respuestas con las de la base de datos. Esta es la técnica que utilizan herramientas como Fire & Water, que puede encontrarse en la siguiente dirección: http://www.ntobjectives.com/products/firewater/

El siguiente artículo de Saumil Shah analiza la herramienta httpprint:
http://net-square.com/httprint/httprint_paper.html

La herramienta httpprint puede encontrarse en:
http://net-square.com/httprint/

40.Un amigo me ha contado que es más seguro hacer correr mi servidor bajo un puerto no estándar, como 5001. ¿Es esto verdad?

Un servidor Web normalmente necesita ser accedido por un número grande de usuarios. Dado que los servidores Web corren normalmente bajo el puerto 80, los navegadotes están configurados para acceder a los servidores Web a través del puerto 80. Si se cambia el puerto en el servidor, se obliga a los usuarios que especifiquen el puerto junto con el dominio. Sin embargo, puede ser una buena idea para una aplicación de tipo Intranet en la que todos los usuario saben dónde conectarse. Esto es más seguro ya que así el Servidor Web no será encontrado por ataques

automatizados (como gusanos) que verifican el puerto 80 y otros puertos estándar.

41. ¿Debe realmente preocuparme que se identifique mi servidor Web?

sí, se debe proteger al servidor contra su identificación ya que éste puede ser el primer paso de un ataque peligroso. Si un atacante averigua que un sitio Web se basa en IIS 5, por ejemplo, entonces buscará las vulnerabilidades conocidas de este servidor Web. Si el servidor Web no está parchado para todas las vulnerabilidades o el atacante encuentra una nueva para la que aún no existe parche, entonces nada podrá impedir su ataque. Una vez comprometido el servidor Web se puede hacer un gran daño. También se puede despistar a algunas herramientas y gusanos manipulando la información de la versión del servidor. algunos atacantes determinados y persistentes pueden recurrir a medidas adicionales para identificar el servidor, mas los problemas que ahora tienen que sobrepasar son mayores cuando es mas difícil identificar el nombre y versión del servidor.

Pruebas (Testing)

42. Quiero encadenar un software de tipo proxy con mi servidor proxy, ¿existen herramientas que permitan hacerlo?

Las herramientas que permiten el encadenamiento de proxys. son:

WebScarab - http://www.owasp.org/development/webscarab
Exodus - http://home.intekom.com/rdawes/exodus.html
Odysseus - http://www.wastelands.gen.nz/odysseus/index.php

43. ¿No pueden ser automatizadas las pruebas? ¿Existen herramientas que pueda correr contra mi aplicación?

Existen herramientas que revisan aplicaciones en busca de fallos de seguridad. Pero estas herramientas sólo pueden buscar un número limitado de vulnerabilidades y pueden no encontrar todos los problemas en la aplicación. Añadido a esto, muchos ataques requieren de la comprensión de las reglas de negocio de la aplicación para decidir, por ejemplo, que variables manipular en una petición determinada, cosa que una herramienta es incapaz de realizar.

En la siguiente presentación Jeremiah Grossman analiza las limitaciones de la revisión automática:
http://www.blackhat.com/presentations/bh-federal-03/bh-fed-03-grossman-up.pdf

Algunas herramientas de este estilo son:
SpikeProxy, herramienta open-source y gratuita disponible en:
http://www.immunitysec.com/spikeproxy.html
AppScan, disponible en:
http://www.sanctuminc.com/trials/index.cfm?type=appscanau
WebInspect, puede encontrarse en:
http://www.spidynamics.com/productline/WE_over.html

44.¿Dónde puedo realizar mis pruebas? ¿Existe una aplicación Web con la que pueda practicar?

OWASP provee una aplicación de ejemplo que puede ser usada para este propósito. WebGoat, como el sitio lo menciona, el objetivo del proyecto en enseñar seguridad en Web en un ambiente interactivo. Hay lecciones sobre la mayoría de las vulnerabilidades comunes .
http://www.owasp.org/development/webgoat

Una aplicación similar para aprender es WebMaven, disponible en http://sourceforge.net/projects/webmaven

Otro sitio interesante es http://www.hackingzone.org/sql/index.php que cuenta con un juego sobre la inyección de SQL.

45.¿Existen herramientas para el revisión de código fuente que permitan predecir vulnerabilidades para lenguajes como .NET, java, php, etc?

Rough Auditing Tool for Security (*RATS*) es una herramienta que revisa código fuente en busca de fallos de seguridad para los lenguajes *C*, *C++*, *Python*, *Perl* y *PHP*. Pude encontrarla en la siguiente dirección:
http://www.securesoftware.com/download_rats.htm

Nos gustaría saber sobre mas herramientas para revisar código fuente. Si sabe de alguna por favor infórmenos y la agregaremos a esta lista.

46.¿Pueden otros protocolos diferentes de HTTP ser interceptados y usados para realizar ataques?

Sí. *Interactive TCP Replay* (*ITR*) es una herramienta que actúa como *proxy* para aplicaciones que no se basen en *HTTP* y que además puede modificar el tráfico. Permite la edición de los mensajes en un editor hexadecimal. *ITR* además guarda en un *registro* todos los mensajes entre el cliente y el servidor. Puede usar diferentes tipos de codificación de caracteres (como *ASCII* o *EBCDIC*) para la edición o el registro. Más información en: http://www.webcohort.com/web_application_security/research/tools.html

Gestión De Galletas(Cookies) Y Sesiones

47.Qué son las galletas seguras?

Una galleta puede ser marcada como "secure" lo cual asegura que la galleta solo se envíe en sesiones SSL. Si "secure" no es especificado, la galleta será enviada sin encriptar en canales no SSL. Galletas sensibles como las fichas de sesión deben ser marcadas como seguras si todas las paginas en el sitio Web que requiere fichas de sesión son SSL. Una cosa a mantener en mente aquí es que, generalmente, las imágenes no son bajadas sobre SSL. Así que si una ficha de sesión será presentada para bajar una imagen esta será enviada en texto claro a menos que el atributo "secure" sea utilizado.

48.¿Puede otro sitio Web robar las galletas que mi sitio Web almacena en la máquina de un usuario?

No, no es posible que un sitio Web acceda a las *galletas* de otro sitio Web. Las *galletas* llevan asociado un atributo de dominio. Sólo una petición que provenga del dominio especificado podrá acceder a la *galleta*. Este atributo tiene un único valor.

49.¿Existe algún riesgo en usar galletas persistentes frente a galletas no persistentes?

Las *galletas* persistentes son datos que el sitio Web guarda en el disco duro (o el equivalente) del usuario para mantener información entre diferentes sesiones del navegador. Estos datos se mantendrán en el sistema del usuario y podrán ser consultados por el sitio Web la próxima vez que el usuario lo visite.

Las *galletas* no persistentes son aquellas que sólo son usadas durante la sesión de navegador en la que fueron creadas. Se almacenan en la memoria y no en el disco duro.

El riesgo de seguridad de las *galletas* persistentes proviene de que generalmente se almacenan en un archivo de texto en el cliente, de forma que un atacante que accediese a la máquina del usuario podría robar la información.

50. Si uso un identificador de sesión calculado en función de la IP del cliente, ¿estoy protegido contra el robo de la sesión?

El robo de sesión aun es posible, suponga que el atacante está en la misma red LAN que el usuario y usa la misma dirección IP de proxy que usa el usuario para acceder el sitio Web. El atacante puede aun robar la sesión si puede rastrear el identificador de sesión. Tampoco sería posible implementar esto si el dirección IP del cliente cambia durante la sesión, invalidándola. Esto puede pasar si el cliente viene de un banco de servidores proxy.

51. ¿Cuál es el mejor método para transmitir identificadores de sesión: en galletas, en la URL o en variables ocultas?

Trasmitir los identificadores de sesión en la *URL* conlleva ciertos riesgos. Otros usuarios de la misma máquina pueden ver el identificador de sesión, si la *URL* se guarda en la *memoria rápida* del cliente, el ID de sesión también se guardará con ella. Además, los IDs de sesión podrían guardarse también en los *registros(logs)* de otros sitios en el atributo *referrer* (que guarda la *URL* del sitio del que proviene el usuario). Los campos ocultos (o variable ocultas) no son siempre prácticas ya que toda petición no tiene por qué ser de tipo *POST*. Las *galletas* son el método más seguro ya que no se guardan en *memoria rápida*, no son visibles en los registros *W3C* o *referrer* y la mayoría de los usuarios las aceptan.

52. ¿Es útil cifrar las galletas que guardan el identificador de sesión y enviarlas al servidor?

Cifrar solamente el ID de sesión sobre una conexión sin SSL no sirve ya que el ID de sesión se cifrará una vez y siempre será enviado el mismo valor (el

que resulta del cifrado). Un atacante puede usar este valor cifrado para robar la sesión.

53.¿En qué se basa el concepto de usar un identificador (ID) de página además del identificador de sesión?

El ID o *ficha* de sesión tiene la misma duración que la sesión y está unido al usuario que realizó el *inicio de sesión*. Un ID o *ficha* de página tiene la misma duración que la página y representa una página enviada por el servidor. Se trata de una *ficha* única que se proporciona cuando una página se descarga y que es presentado en el momento de acceder a la siguiente página. El servidor esperará un determinado valor del usuario para acceder a la siguiente página. Sólo si la ficha enviada coincide con el esperado se servirá la siguiente página. Este mecanismo puede usarse para asegura que un usuario accede a las páginas siguiendo la secuencia determinada por la aplicación. Un usuario no podrá por tanto pegar una dirección *URL* en el navegador y saltarse páginas solo porque tiene un ID de sesión, dado que la ficha de página no estará autorizada a acceder un URL mas profundo directamente.

Registros (Logs) Y Auditoria

54.¿Qué son los registros W3C?

W3C es un formato de registro usado en los archivos *de registro* de los servidores Web. Estos *registros* guardan los detalles de acceso de cada petición: la fecha y hora, IP de origen, página solicitada, el método usado, versión del protocolo *HTTP*, tipo de navegador, la página de origen (*referrer*), el código de respuesta, etc. Nótese de que se trata de registros de acceso, por lo que existirá un registro para cada petición. Cuando se descarga un página con varios ficheros gif, se guardará una entrada por cada uno de ellos; por tanto, estos *registros* suelen ser voluminosos.

55.¿Necesito mantener un registro en mi aplicación a pesar de tener activados los registros W3C?

Sí, es importante que el sitio Web mantenga un *registro* a nivel de aplicación. Ya que los *registros W3C* guardan una entrada por cada petición *HTTP*, resulta difícil (y en ocasiones imposible) extraer de ellos información útil de "alto nivel". Por ejemplo, resulta muy engorroso identificar en ellos una sesión específica de un usuario y qué acciones ha realizado. Es mejor

que la aplicación lleve un registro de las acciones más importantes, en lugar de tener que extraerlas de los *registros W3C*.

56.¿Qué debo registrar en el registros de mi aplicación?

Deberá guardarse el rastro de toda aquella acción que podamos necesitar revisar para resolver problemas o realizar análisis de lo ocurrido. Es importante resaltar que no es aconsejable guardar información sensible en estos registros, ya que los administradores tienen acceso a los mismos para solucionar problemas. Las acciones que comúnmente se registran son:

- Entrada y salida de sesión para usuarios
- Transacciones críticas (por ejemplo, traspasos de fondos entre cuentas)
- Intentos de inicio de sesión fallidos
- Bloqueos de cuentas
- Violación de políticas

Los datos que normalmente se guardan son:

- Nombre del usuario
- Fecha y hora
- Dirección IP fuente
- Código de error, si procede
- Prioridad

57.¿Debo cifrar mis registros? ¿No afecta esto al rendimiento?

Se requiere el cifrado cuando la información debe ser protegida de ser leída por usuario no autorizados. Sí, el cifrado supone una carga que afecta al rendimiento, de forma que si los *registros* no contienen información sensible pueden dejarse sin cifrar. Aún así, se recomienda el uso de firmas digitales para proteger los *registros* de posibles manipulaciones. Las firmas digitales son procesos menos costosos que el cifrado.

58.¿Puedo fiarme de las direcciones IP registradas en mi registro? ¿Puede un usuario personificas o falsificar su dirección IP?

Un atacante que quiera esconder su dirección IP actual podrá usar un servicio como *anonymizer* o usar "*relays"* de HTTP abiertos (se trata de servidores Web "impropiamente" configurados que están accesibles y que

son usados como *proxys HTTP* para conectarse a otros sitios). En tales casos la dirección IP que estará viendo en los archivos de registro será aquella de los servicios o el "*relay"* abierto que está siendo usado. Por tanto la dirección IP guardada en el *registro* no siempre es fiable.

Criptografía/SSL

59.¿Debería usar SSL de 40 o de 128 bits?

Existen dos niveles de SSL: del 40 bits y de 128 bits. Este número hace referencia a la longitud de la llave secreta usada para cifrar la sesión. Esta llave es generada para cada sesión SSL y se mantiene durante el resto de la misma. Cuanto más larga sea la llave más difícil será romper la protección de los datos. Por ello, el cifrado de 128 bits es mucho más segura que la de 40 bits. La mayoría de los navegadotes de hoy soportan el cifrado de 128 bits.

60.¿Realmente SSL de 40 bits es inseguro?

Realmente no es inseguro. Ocurre que computacionalmente puede "romperse", mientras que para el caso de una llave de 128 bits no es así. Aunque en el caso de 40 bits, se necesita un número 200 o mas ordenadores. Nadie ha intentado hacerlo para descubrir un número de tarjeta de crédito o algo por el estilo. Por tanto, dependiendo del tipo de datos que maneje la aplicación, se deberá adoptar una longitud u otra. Usar 128 bits es definitivamente mas seguro.

61.Tengo que usar SSL de 40 bits, ¿Como puedo asegurarme que estoy protegido adecuadamente?

Con SSL de 40 bits puede necesitar tomar precauciones adicionales para proteger la información sensible. Un "hash con sal" para transmitir contraseñas es una buena técnica. Esto asegura que la contraseña no puede ser robada incluso si la llave SSL es rota.

62.¿Qué se cifra cuando uso SSL? ¿La petición de páginas también se cifra?

Una vez finalizada la fase de negociación y la conexión HTTPS se ha establecido, se cifra todo, incluidas las peticiones de páginas. Por tanto cualquier dato enviado a través del URL será también cifrado.

63.¿Qué algoritmos de cifrado usa SSL?

SSL soporta varios algoritmos criptográficos. Durante la fase inicial (llamada fase de saludo) el algoritmo de llave pública RSA. Para cifrar los datos con la llave de sesión, usa los siguientes algoritmos: RC2, RC4, IDEA, DES, triple-DES, y MD5.

64. Quiero usar SSL, ¿Por dónde debo empezar? O ¿Cómo compro un certificado SSL?

Existen numerosas entidades de certificación en las que se puede comprar un certificado. Independientemente de la autoridad certificadota (CA) que se use, los pasos a seguir son los siguientes:

1. Crear el par de llaves para el servidor
2. Crear la petición de firma de certificado (CSR). Esto requerirá proporcionar cierto detalles como la localización y el nombre completo válido del servidor.
3. Enviar la petición CSR a la autoridad de certificación (CA).
4. Instalar el certificado enviado por la CA.

Varios

65. ¿Qué son los cortafuegos de aplicación? ¿Qué tan buenos son en realidad?

Los cortafuegos (o firewalls) de aplicación analizan las peticiones a nivel de aplicación. Estos cortafuegos se usan para aplicaciones específicas como un servidor Web o un servidor de bases de datos. Los cortafuegos de aplicaciones Web protegen al servidor Web de ataques basados en HTTP. Monitorizan las peticiones para detectar ataques de inyección de SQL, XSS, codificación de URL, etc. Pero no protegen contra ataques que requieran comprender la lógica de negocio de la aplicación – lo que incluye la mayoría de los ataques basados en la manipulación de variables. Algunos cortafuegos de aplicación son:

Appshield. Una versión está disponible en:
http://www.sanctuminc.com/forms/regform.cfm?type=appshield

Netcontinuum. Obtenga mas información en:
https://www.netcontinuum.com/index_flash.html

66.¿En qué consisten los registros referrer (referrer logs) y las URLs sensibles?

Una cabecera HTTP contiene un campo conocido como Referrer. A la hora de visitar una página Web podemos:

- Escribir su dirección URL completa en la barra de direcciones del navegador
- Hacer clic en un enlace de otra página
- Ser redirigidos desde otra página

En el primer caso, el campo referrer estará vacío, pero en los otros dos casos contendrá la URL de la primera página. Las URL de la primera página se guardará en el registro del servidor Web de la segunda página, cuando el usuario llega a esta segunda desde la primera.

Ahora supongamos que las dos páginas residen en diferentes sitios y que la primera URL contiene información sensible como un ID de sesión. Si la segunda página reside en la máquina a la que pueda acceder un atacante, éste podrá usar esta información consultando los registros.

La información de las URLs será almacenada en los registros de referrer y en el navegador. Por tanto, deberemos tener cuidado de no incluir información sensible en la URL.

67.¿Quiero usar el lenguaje más seguro?¿Cuál es más recomendable?

Puede ser usado cualquier lenguaje, ya que la seguridad de una aplicación Web reside fundamentalmente en las técnicas o prácticas usadas a la hora de programar. Nuestro consejo es usar cualquier lenguaje en el que uno se sienta cómodo. Al margen de esto, hay algunos lenguajes como Java que incluyen características o funciones especiales como las variables enlazadas (bind) que aportan mayor seguridad. Usted puede usar esas características adicionales si decide programar en ese lenguaje.

68.¿Qué libros son aconsejables para aprender técnicas o prácticas de programación segura?

La guía de OWASP para construir Aplicaciones Web y Servicios Web Seguros (The OWASP Guide to Building Secure Web Application and Web

Services). Es una buena guia para los desarrolladores de aplicaciones Web. http://www.owasp.org/documentation/guide

Escribiendo código seguro (Writing Secure Code) por Michael Howard y David LeBlanc. Tiene un capítulo sobre asegurar servicios basados en Web. puede encontrar mas información sobre este libro en: http://www.dwheeler.com/secure-programs

Programación segura para Linux y Unix (Secure Programming for Linux and Unix HOWTO) por David Wheeler por talks habla sobre como escribir aplicaciones segura incluyendo aplicaciones Web; también especifica algunas guías para varios lenguajes. El libro puede ser encontrado en: http://www.dwheeler.com/secure-programs

69. Hay algún programa de entrenamiento sobre programación segura al cual pueda atender?

Microsoft ofrece programas de entrenamiento para desarrollar aplicaciones Web mejoradas en seguridad (security-enhanced), también, desarrollar y desplegar aplicaciones seguras con el marco de Microsoft .net. Puede encontrar mas información en: http://www.microsoft.com/traincert/syllabi/2300AFinal.asp y http://www.microsoft.com/traincert/syllabi/2350BFinal.asp

SPI Dynamics enseña un curso en seguridad en aplicaciones Web. Puede encontrar mas información en: http://www.spidynamics.com/training/