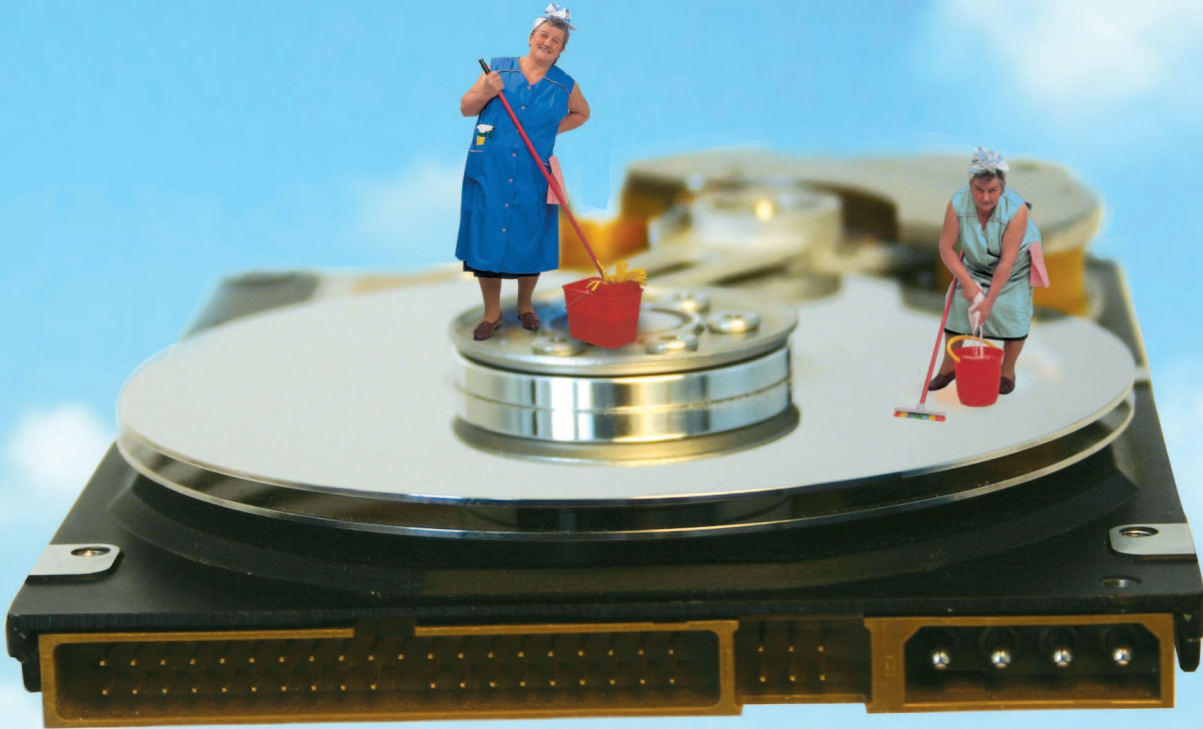


Borrando Datos con Seguridad

LA LIMPIEZA



El tema de las copias de seguridad es algo común, pero rara vez se oye a alguien hablando del borrado de datos seguro. **POR MARCUS NASAREK**

Tiene su viejo disco duro lleno de datos hasta los topes y está pensando en comprar un disco duro nuevo. Para reducir el impacto en el presupuesto familiar decide vender el disco duro antiguo en una de las múltiples subastas que se realizan en la red.

Con el paso del tiempo, su familia entera habrá dejado rastros personales en el disco y, desde luego, deseará eliminar los documentos personales, las cartas, las notas de despidos, las aplicaciones, los álbumes fotográficos y las credenciales de accesos del disco. Afortunadamente, tiene Linux, y una de sus máximas es que la línea de comandos tiene todo lo que se necesita por si fuera necesario.

Probablemente ya conozca cómo borrar un fichero o un directorio. Todas las GUIs adoptan soluciones similares: se pincha en el fichero o el directorio para seleccionarlo y pulsando el botón [Supr] se elimina. O eso es lo que se piensa. Incluso si la aplicación no manda el fichero a la papelera, el contenido aún permanece

almacenado en alguna parte del disco en segmentos de diversos tamaños. Los datos antiguos no desaparecen realmente hasta que se escriben datos nuevos y se empieza a rellenar el disco otra vez.

Si se teclea *rm* en la línea de comandos tampoco ayuda: el comando *rm* simplemente elimina las entradas en los diarios del sistema de ficheros. Sin estas entradas, el sistema de ficheros no tiene forma de localizar los datos, pero esto no significa que los datos hayan desaparecido realmente.

Sobre el Borrado de Datos

El siguiente comando borra el directorio *mis_datos/*, sus directorios y cualquier fichero que contengan sin pedir confirmación:

```
rm -rv mis_datos/
```

Hablando estrictamente, lo que realmente se ha hecho es eliminar la etiqueta del nombre para todos estos ficheros y

directorios. El comando simplemente ha borrado la entrada en la Tabla de Localización de Ficheros. Cada partición necesita un resumen que le indica dónde están almacenados físicamente en el disco los ficheros y los directorios. Esta solución

Listado 1: Búsqueda de Sectores

```
01 #!/bin/bash
02 # showSector.sh
03 MAX = 1000
04 for ((i=0;i<=$MAX;i=$i+1));
05 do
06 val=`dd if=$1 skip=$i
07   bs=512 count=1 2>/dev/null |
08   hexdump -v -e "%_p" | fgrep
09   $2`
10 if [ "$val" ];
11 then
12 echo -e "Sector $i:\n
13   n$val"
14 fi
15 done
```

Listado 2: ¿Creíste que lo habías borrado?

```

01 john@jack:~$ sudo ./showSector.sh /dev/sda1 Sesamo Sector 101:
02 Esto es un secreto. La contraseña para la caverna es "Abrete Sésamo!".
03 Pero no se lo digas a nadie!
04.....
05.....
06.....
07.....
08.....
09.....

```

es compartida por todos los sistemas operativos modernos, aunque existen algunas diferencias en la forma en la que almacenan y leen los datos.

Pero lo importante es que, cuando se “borra” un fichero o directorio, el sistema simplemente elimina la etiqueta de la tabla, liberando de este modo el espacio. La eliminación física de los datos del espacio donde la etiqueta está apuntando significaría un esfuerzo computacional mucho mayor.

Dicho de otro modo, se tiene una buena oportunidad de recuperar los ficheros que se han borrado por error si se evitan los accesos de escritura a la partición y se empieza a buscar los ficheros perdidos inmediatamente. Se puede encontrar una descripción de los pasos a seguir para recuperar los datos borrados en una partición Ext2 usando herramientas simples en [1]. Y el

COMO en [2] proporciona más detalles.

Incluso formateando o particionando de nuevo una unidad no le ayudará a librarse de los datos no deseados. Cuando se formatea un disco, un sistema operativo moderno simplemente reescribirá la Tabla de Localización de Ficheros y un particionamiento tan sólo cambiará las entradas en la tabla de particiones del disco. Los datos simplemente podrían estar perdidos en algún lugar del nirvana digital, pero aún están físicamente en el disco.

Buscando Pistas

No se necesitan grandes conocimientos para recuperar los datos borrados de un disco. El cuadro titulado “Particiones” explica cómo el disco duro y el sistema operativo manejan los ficheros y los directorios.

Listado 3: Una Solución más Segura

```

01 #!/bin/bash
02 # Parameter $1 is the device
   to be deleted
03 BLOCKSIZE=8192
04 echo -e "Device to delete:
   $1\nBlock size for
   write: $BLOCKSIZE"
05 echo "[`date +%a %T`]" Round
   1"
06 dd if=/dev/zero of=$1
   bs=$BLOCKSIZE
07 echo "[`date +%a %T`]" Round
   2"
08 dd if=/dev/urandom of=$1
   bs=$BLOCKSIZE
09 echo "[`date +%a %T`]" Round
   3"
10 dd if=/dev/zero of=$1
   bs=$BLOCKSIZE
11 echo "[`date +%a %T`]" $1
   deleted"

```

Si sólo se están buscando claves de búsquedas específicas o cadenas, como credenciales de acceso o datos personales, probablemente lo único que se necesita son comandos como *dd*, *hexdump* y *fgrep*. Será preciso disponer de los privilegios de *root* para poder realizar la mayoría de estas operaciones.

¿Pura Coincidencia?

Los ficheros de dispositivos especiales, */dev/random* y */dev/urandom*, utilizan un controlador del kernel para generar números pseudoaleatorios. El término “pseudo” es indicativo de un bien conocido problema de los ordenadores. Los ordenadores no pueden generar números aleatorios, aunque pueden generar más o menos valores aleatorios. Es decir, */dev/random* y */dev/urandom* generan números con un suficiente grado de aleatoriedad para la mayoría de las aplicaciones criptográficas de un PC.

Cuando se generan secuencias de números aleatorios, el generador de números aleatorios basado en el kernel obtiene diversos valores internos de los dispositivos que utiliza para obtener suficiente entropía. La entropía expresa el grado de aleatoriedad de una secuencia de números generada en un periodo de tiempo específico.

Las aplicaciones leen un flujo de bytes desde los ficheros */dev/random* y */dev/urandom*. Al contrario que */dev/urandom*, */dev/random* sólo proporciona un byte si se ha alcanzado el suficiente grado de entropía. Si este no es el caso, el dispositivo bloquea la salida hasta que se hayan acumulado sufi-

cientos datos para asegurar un buen nivel de entropía para los valores.

Como puede tardar un rato, el modo de no bloqueo del sistema de E/S permite eliminar el bloque, pero esto no mejora la velocidad de respuesta. Los bytes proporcionados por */dev/random* proporcionan números aleatorios perfectamente válidos para poder utilizarse en sistemas criptográficos y son lo suficientemente seguros para ser usados en cifrados largos y como material de alta calidad para generar claves.

/dev/urandom no garantiza el nivel de entropía que se obtiene con el generador del kernel. Al tener una entropía más baja, no se interrumpe el flujo de bytes. Este hecho hace que los números sean más “pseudos” que nunca, pero hay que recordar que no es siempre necesario apuntar tan alto en criptografía.

De hecho, */dev/urandom* está bien para obtener claves temporales, como las claves de sesión en las sesiones web, para rellenar el espacio en disco con ruido o para la autenticación a corto plazo en escenarios de reto-respuesta.

Una unidad de memoria USB es una buena candidata para realizar experimentos. Se puede formatear el “disco” con varios sistemas operativos y llenarlo con datos de pruebas. Supóngase que el dispositivo que se desea investigar es `/dev/sda`; la primera partición del dispositivo sería `/dev/sda1`.

Puede utilizarse el editor hexadecimal `hexdump` para inspeccionar los sectores de la partición y ayudarse de él para encontrar los datos existentes. El siguiente comando crea un volcado de la unidad y muestra cualquier carácter imprimible como código ASCII. Se puede ver el contenido de cualquier fichero borrado, además de los caracteres no estándar, inmediatamente.

```
hexdump -v -e "%07.7_ad: " 2
60/1 "%_p" "\n" 2
/dev/sda1 | less
```

Si este modo de vista no es de su agrado, probablemente prefiera ejecutar Midnight Commander en una ventana de consola y experimentar con esta nueva vista. Midnight Commander es un administrador de archivos que puede también mostrar los datos en formato hexadecimal.

Desde luego, los ficheros no tienen necesariamente que estar almacenados en

sectores contiguos del disco; de hecho, probablemente estén dispersos por todo el disco; los expertos lo denominan fragmentación. Todo depende de cómo el sistema operativo organice el espacio de almacenamiento para conseguir los mejores tiempos posibles de lectura y escritura. Pero como al menos un fichero siempre ocupa un cluster, se tiene una buena oportunidad de descubrir ficheros pequeños de texto de menos de 4096 bytes.

Esta técnica no se utiliza mucho como solución sistemática para buscar un fichero específico. Si se sabe que una determinada cadena aparece en el fichero que se quiere recuperar (por ejemplo, los ficheros LaTeX siempre empiezan por `/document`); un simple script puede ayudar a restringir la búsqueda. Tan sólo hay que pasarle la unidad, la cadena y el número de sectores a buscar al script del Listado 1. Si la búsqueda devuelve resultados, tan sólo hay que investigar más detenidamente los sectores en cuestión.

La ejecución del script del Listado 1 produce la salida que se muestra en el Listado 2 cuando se busca la cadena *Sesame* en el dispositivo `/dev/sda1`.

Hace falta el comando `dd`, con los parámetros `skip` y `count` para investigar la localización.

`skip` permite que el comando se salte un número determinado de bloques. El script proporciona un valor para el parámetro `skip`.

El parámetro `count` devuelve la longitud de la zona bajo investigación. Ambos valores pueden modificarse ligeramente para investigar la zona sospechosa. Una combinación inteligente de estos dos parámetros permitirá realizar una copia de las zonas de interés. Por ejemplo, si el sector 32 contiene una cadena interesante, se puede utilizar el siguiente comando para investigar los 5 sectores de alrededor de esta localización:

```
dd if=/dev/sda1 skip=30 count=52
bs=512 2>/dev/null | hexdump 2
-v -e "%_p" | less
```

Este ejemplo muestra la técnica. Los editores de disco probablemente sean la herramienta elegida para analizar o recuperar ficheros mayores desde los datos del disco. Un editor de disco ayuda a navegar a través de las grandes cantidades de datos y realizar un volcado de los bits interesantes.

Los servicios de recuperación de datos profesionales utilizan técnicas de análisis de superficie para recuperar los datos de

Particiones

El disco duro almacena los datos que maneja en sectores de 512 bytes. Los discos duros modernos usan LBA (Logical Block Addressing, Direccionamiento de Bloques Lógicos) para numerar los sectores en secuencia. Al mismo tiempo, el disco duro controla y monitoriza diversos parámetros y proporciona la interfaz para acceder a los mismos.

Los parámetros incluyen varias medidas de temperatura, pero lo más importante es una tabla de sectores defectuosos. El dispositivo etiqueta los sectores como defectuosos e impide que se puedan acceder a ellos. Si se nota un incremento considerable de sectores defectuosos, es probable que en un futuro cercano se produzca un fallo total del disco.

Los sectores marcados como defectuosos no están disponibles para el acceso normal. Los expertos pueden utilizar herramientas especiales para poder acceder a dichos sectores, sin embargo, es bastante improbable que contengan datos de alto secreto.

Un disco duro debe contener una partición, que agrupa cierta cantidad de sectores. La tabla de partición contiene la forma de las particiones. Esta tabla está localizada en el primer sector, también denominado MBR (Master Boot Record, Registro de Arranque Maestro), empezando en el byte 446 y con una longitud de 64

bytes. El MBR también contiene el cargador del sistema para poder arrancarlo.

El sistema operativo maneja los datos de la partición en una especie de base de datos que contiene la estructura de una tabla. Este es el caso de la FAT (File Allocation Table, Tabla de Localización de Ficheros): DOS y Windows utilizan diversas variantes de la FAT conocidas como FAT12, FAT16 y FAT32.

Linux soporta un amplio rango de sistemas de ficheros. Son comunes Ext2, Ext3 y ReiserFS. Los dos últimos utilizan un diario, algo similar a una base de datos genuina, para proporcionar una gestión extremadamente efectiva de las entradas en la misma, ofreciendo de este modo bastantes ventajas, especialmente con particiones grandes. Como contrapunto, esto hace mucho más difícil la localización de los datos borrados.

El sistema de ficheros agrupa los sectores para formar clusters con la idea de mejorar el rendimiento. Dependiendo de la elección del sistema de ficheros, los clusters pueden ser de diversos tamaños. Si se tiene un tamaño de cluster de cuatro KBytes, los ficheros normalmente ocuparán un tamaño múltiplo de este valor y como mínimo un único cluster. Un fichero que contenga un solo byte ocupará por ello al menos 4096 bytes de espacio en el disco.

zonas del disco que han sido reescritas. Estas técnicas de análisis de superficie se basan en el hecho de que la cabeza escritora no abarca toda la pista antigua, dejando datos residuales a los lados de la misma.

Borrado Seguro

A la luz de todo esto, probablemente se prefiera utilizar un método estándar de borrado para borrar los datos y eliminar cualquier posibilidad de recuperación de los mismos. Las siguientes soluciones son probablemente las más conocidas:

- Guía BSI para salvaguardar los documentos clasificados en los sistemas de procesamientos de datos (VSITR), Informe N° 11, [BMI]
- El estándar 5220.22-M del departamento de defensa de los EE.UU., [DOD]
- El algoritmo de Bruce Schneier, [BSA]
- El algoritmo de Peter Gutmann, [PGA]

Los algoritmos anteriormente mencionados sobrescriben los sectores múltiples veces con patrones específicos de datos. Los patrones comprenden los bytes 0x00, 0xFF y valores aleatorios. Con la idea de generar datos genuinamente aleatorios y hacer que sea imposible substraer estos datos de la señal leída, la mayoría de las soluciones usan la generación de números aleatorios.

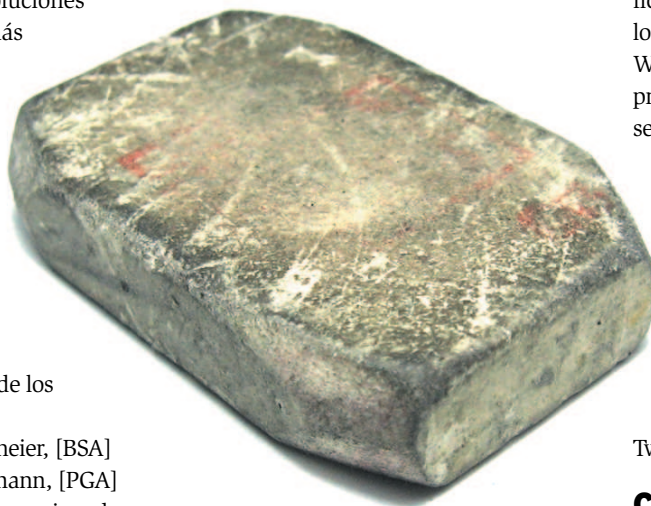
Linux proporciona a los usuarios un alto nivel de seguridad basándose en ficheros de dispositivos especiales como `/dev/urandom`, que crea datos aleatorios simples. `/dev/zero` proporciona cualquier cantidad de bytes con valor cero (0x00). Con una combinación de ambos se obtienen todas las herramientas que se necesitan.

El cuadro titulado “¿Pura Coincidencia?” proporciona información sobre la calidad de los dos generadores de números aleatorios: `/dev/random` y `/dev/urandom`. Solamente `/dev/urandom` es capaz de proporcionar la cantidad de números aleatorios necesaria para algo tan grande como un disco duro.

Los dos ficheros, `/dev/zero` y `/dev/urandom`, pueden usarse conjuntamente con una de las soluciones mencionadas anteriormente. Para uso doméstico, con la ejecución del script que

aparece en el Listado 3 debería ser suficiente. Pero hay que tener cuidado de elegir el dispositivo adecuado si lo que quiere es estar seguro de la eliminación definitiva de los datos.

El script del Listado 3 primero llena el disco con ceros, luego con valores aleatorios y posteriormente con ceros de nuevo. Con tres rondas, normalmente, es suficiente. Por otro lado, el programa tardará unas horas en borrar un disco de 80Gb.



Las normas de protección de datos normalmente optan por siete rondas, tal como hace el DOD de EE.UU. con el estándar 5220.22-M. El algoritmo de Peter Gutman, que es el más moderno de nuestros candidatos desde el punto de vista tecnológico, realiza 35 rondas y requiere bastante paciencia por parte de los usuarios. Si se comprueba el disco tras su ejecución lo único que se podrá observar son ceros.

Borrando Ficheros Individuales

Algo similar a la solución ofrecida en el Listado 3 puede teóricamente aplicarse al borrado de un único fichero del disco. Pero este programa no es apto para el borrado recursivo de directorios completos y probablemente se prefiera buscar una alternativa o mejorar el programa para que se ajuste a las necesidades.

Son más convnientes aplicaciones como Wipe. La mayoría de las distribuciones incluyen la herramienta, que permite el borrado de árboles de directorios completos:

```
wipe -r directorio
```

Sin embargo, sólo se puede confiar en Wipe si se deshabilita la caché de escritura del disco duro. El programa requiere el acceso exclusivo al disco duro en cada ronda, como explica la página oficial del proyecto [6].

Bajo circunstancias normales debería ser suficiente con un kernel que soporte bloqueos, suponiendo que se acuerde de especificar la opción `mand` cuando se monte. Si el sistema no se ajusta a estos requerimientos o si el sistema mueve los ficheros que se sobrescriben a una localización diferente, programas como Wipe no deben usarse, ya que proporcionan una falsa sensación de seguridad, lo que es peor.

Wipe utiliza el patrón de Peter Gutmann para crear cadenas que utiliza para la sobrescritura. Para ello, accede a los ficheros especiales `/dev/urandom` y `/dev/random`, que proporcionan el nivel de entropía necesario. Para acelerar el proceso,

Wipe también utiliza el generador de números pseudo-aleatorios de Mersenne Twister (PRNG).

Conclusiones

Es interesante notar que el departamento de defensa de EE.UU. estipula la destrucción física de los dispositivos de almacenamiento magnético que contienen datos de alta confidencialidad. Si se manejan datos sensibles, siempre se debería tener en cuenta que no existe el software perfecto. Como punto a destacar, las herramientas Linux deberían proporcionar a la mayoría de los usuarios más seguridad de la que realmente necesitan. ■

RECURSOS

- [1] Recuperando datos borrados en Linux: <http://wiki.yak.net/592>
- [2] Recuperando datos en un sistema de ficheros Ext2: <http://www.faqs.org/docs/Linux-mini/Ext2fs-Undeletion.html>
- [3] Guía para el manejo de datos confidenciales en las oficinas USA: http://www.dss.mil/isec/nispom_0195.htm
- [4] Página web de Bruce Schneier: <http://www.schneier.com>
- [5] Página web de Peter Gutmann: <http://www.cs.auckland.ac.nz/~pgut001>
- [6] Wipe: <http://wipe.sourceforge.net>