Universidad Autónoma de México
Facultad de Ingeniería
Dirección de Ingeniería Eléctrica

Análisis forense de la bitácora en Ext3

Tesina para la obtención del Título de Ingeniero en Computación presenta: Gregorio Narváez Rosado

Asesor de Tesis: M.C. Alejandro Velásquez Mena

TABLA DE CONTENIDO

Introducción

La recuperación de datos borrados ya sean por causa accidental o intencional es parte fundamental del cómputo forense, este escrito que a continuación se presenta es resultado del desarrollo de una técnica de recuperación de información borrada en sistemas de archivo EXT3 que por lo general se encuentran en sistemas Linux. Las técnicas desarrolladas en 2007 por el autor, fueron publicadas en el documento de investigación *Taking Advatage of Ext3 journaling File System in a forensic investigation*. Para obtener la certificación como analista forense otorgada (GCFA) por el Instituto SANS de los Estados Unidos de América.

La primera aplicación práctica de de estas técnicas se dieron en 2009 para un cliente que por cuestiones de confidencialidad no puede ser mencionado, solicitó que se realizara un análisis forense para recuperar su información contenida archivos que fueron borrados de forma intencional. La información se encontraba almacenada en un servidor con sistema operativo Linux Fedora 8, y el sistema de archivos instalado era Ext3. Esta unidad de disco no es la principal donde se almacenaba el SO, y recientemente se había cambiado debido a un daño físico justo después del borrado de los archivos por lo que toda la información que incluía entre otras cosas las bitácoras de acceso de las cuentas de usuario se habían perdido (El cliente no deseaba entrar en problemas de litigio y la recuperación de datos del medio físico se considero fuera de su presupuesto).

Sin embargo la información era de suma importancia y se buscaba tener algún tipo de indicio sobre la actividad previa al borrado de los archivos por lo que se llevó a cabo la recuperación de los archivos borrados utilizando técnicas de recuperación por medio de la bitácora.

El sistema de archivos Ext3 tiene un uso extendido en los sistemas Linux. Aun con la aparición de Ext4, Ext3 sigue en uso por lo que es importante entender como maneja la información para poder llevar a cabo un análisis adecuado, en especial para poder recuperar archivos borrados y actividad del sistema de archivos.

Existía la creencia de que no es posible recuperar un archivo que ha sido borrado de un sistema de archivos Ext3/4.

Hoy en día se sabe que es posible recuperar archivos que han sido borrados utilizando los metadatos que se encuentran en la estructura de la bitácora del sistema de archivos, también es posible obtener la actividad temporal del sistema de archivos de forma extendida utilizando la información de la bitácora, como si fuera una “Máquina del tiempo”. Gracias a la bitácora es posible realizar las siguientes actividades forenses:

- Recuperación de archivos
- Rastreo de actividad en el sistema de archivos

El disco donde se almacenaba los archivos había sido desconectado y se habían sido almacenado sin volverse a montar, por lo que existía una muy buena probabilidad de que el la bitácora contenga bastante información para llevar a cabo la recuperación de los archivos y generar una línea temporal que proporcione una idea de cuando fueron borrados.

El presente reporte incluye una introducción a los conceptos generales de cómputo forense, metodología y mejores prácticas así como el desarrollo práctico-teórico para poder llevar a cabo el análisis forense en este tipo de escenarios

Capítulo 1. Experiencia Profesional

De 1999 a 2002 laboré para Radiomóvil Dipsa S.A de C.V. mejor conocida como Telcel, desempeñando el cargo de Jefe de área para la Dirección de Ingeniería del Corporativo encargado del área de investigación de sistemas de prepago (Plataforma Amigo). La plataforma que se introdujo en esa época estaba basada en servidores Sun Starfire 10000, pertenecientes a la extinta Sun Microsystems (adquirida por Oracle), en la que se ejecutaba una aplicación propietaria para el control del cobro y la llamada en tiempo real, el cual hacía uso de una base de datos Oracle y protocolo de señalización SS7 para la comunicación del sistema con las centrales celulares.

Se llevaron a cabo trabajos de integración de plataformas de terceros que complementaran las funciones o solventaran las deficiencias de la plataforma de prepago. Resultado de estos trabajos fueron el sistema de monitoreo y alerta temprana basada en TNG Unicenter de Computer Asociated para los sistemas de soporte y control del *333, y el Sistema de Activación Integral el cual permitía llevar a cabo la activación de una línea amigo y los servicios relacionados como el buzón de voz y el servicio de mensajes cortos en cuestión de minutos, lo que permitió una tasa de activación de 20,000 números al día para la zona metropolitana con una tasa de error del 0.01%

De 2002 a 2005 establecí un pequeño negocio en la ciudad de Mérida, Yucatán conocido como Centro de Negocios Netcafe Millennium, el cual contaba con 2 plantas y ofrecía servicios de centro de cómputo, principalmente para profesionistas que por su giro requerían de un ambiente de trabajo de oficina pero no podían contar con este ya sea por su costo o por su actividad, pero también se daba servicio a estudiantes y turistas.

Además de la administración del negocio, también estaba a cargo de la planeación técnica, por lo que para cumplir con los objetivos del negocio se diseño una red que contaba con 15 estaciones de trabajo, un servidor de recuperación de datos Windows XP, un servidor de respaldos, dos firewalls dedicado, antivirus centralizado (Norton AV), sistema de respaldo centralizado (Norton Ghost) y un programa de mantenimiento y prevención, lo cual permitía ofrecer a los usuarios confiabilidad, seguridad y rapidez en el servicio. También se contaba con una sala de capacitación con 6 estaciones de trabajo en el cual se impartían cursos de capacitación orientados a personas mayores en el uso de Windows XP, navegación por internet, correo electrónico y paquetería Office.

De 2005 a la fecha me he desempeñado como consultor Independiente con base de operaciones en Mérida, Yucatán y el Distrito Federal, he impartido cursos de arquitectura en redes Inalámbricas, seguridad en redes y hackeo ético del EC-Council, en los últimos dos

años he buscado extender mis servicios vía Internet a través de las plataformas Elance y oDesk.

Entre las actividades llevadas a cabo para clientes en diversas partes del mundo se pueden mencionar endurecimiento de servidores virtuales privados (VPS por sus siglas en inglés), migración de sitios web y administración del servidor para un cliente en Brasil.

Para clientes de Reino Unido, Canadá y los Estados Unidos como consultor de seguridad y desarrollo en áreas como recuperación de archivos, análisis forense, capacitación en mejores prácticas, adecuación de scripts para diversas tareas en Linux shell.

También a lo largo de mi experiencia profesional he tenido la oportunidad de impartir conferencias en la Universidad Anáhuac Mayab y el Instituto tecnológico de Mérida sobre inseguridad en redes Inalámbricas en 2008

En la actualidad me encuentro certificado en manejo de Incidentes y análisis forense por el Instituto SANS de los Estados Unidos desde 2005 y 2006 respectivamente, en pruebas de penetración por eLearningSecurity en 2012 y en arquitectura de redes Inalámbricas y seguridad en redes por el EC-Council en 2006.

Capítulo 2. Análisis forense de la bitácora en Ext3

Con el avance de la tecnología conlleva su lado oscuro y es que a través de la historia el ser humano siempre encuentra como aplicar las nuevas tecnologías de formas criminales y nocivas. Desafortunadamente las tecnologías de la información no están exentas de este tipo de aplicación y es que hoy en día con el uso de la tecnología en todos los aspectos de la vida cotidiana, su uso para fines criminales es cada vez más común.

Aunado al avance de las tecnologías de la información ha surgido otra disciplina que se ha desarrollado es el cómputo forense el cual es una disciplina de la seguridad en TI la cual se enfoca en adquirir, analizar y reportar información que permita la resolución de un crimen u acción ilegal en el cual se ve involucrado algún componente tecnológico como lo son redes de datos, computadoras e incluso equipo celular.

2.1 Cómputo Forense y Evidencia Digital

El cómputo forense es la disciplina que realiza la preservación, identificación, extracción, interpretación y documentación de evidencia almacenada en una computadora/dispositivo digital, que incluyan las reglas de evidencia, proceso legal, integridad de la evidencia, reporte de hechos de la información encontrada, y proveer de una opinión experta en una corte legal u otros procesos legales y/o administrativos sobre las hallazgos y conclusiones alcanzadas.

Por ser una actividad que generalmente tiene repercusiones legales el especialista en cómputo forense debe de ser una persona muy disciplinada, metódica, con gran capacidad de análisis y observación, sin embargo aunque es una disciplina altamente técnica se requiere que también se involucre en los aspectos legales, para poder trabajar de una forma efectiva con el personal del área legal

Como se aprecia en la definición de cómputo forense, esté gira en torno al trabajo realizado sobre evidencia digital, por lo que es necesario explicar que es evidencia digital en el contexto del cómputo forense

Evidencia Digital o Evidencia Electrónica es cualquier información con valor probativo almacenado o transmitido en forma digital. La evidencia puede ser cualquier cosa con que establece o no un hecho y puede ser desde una dirección IP o un grupo de bytes que representan la firma de un programa en el sistema que se está analizando

Esta definición general está en función del contexto de la metodología y no como una definición legal de evidencia ya que esta varia en gran medida de la legislación bajo lo que se encuentre desarrollando el caso. La legislación de nuestro país desafortunadamente se

encuentra muy retrasada con respecto a la de otros países en materia de crímenes donde se encuentran involucrados elementos tecnológicos. La evidencia digital es muy frágil ya que el simple hecho de navegar un sistema de archivos de un dispositivo se cambian los tiempos de último acceso, modificación, etc. de los archivos.

2.2 Volatilidad.

Mucha de la información que existe en un dispositivo electrónico tiene un tiempo de vida corto, por lo que es necesario conocer qué elementos de evidencia se pierden primero para poder tener un plan de prioridades sobre la adquisición de evidencia. El siguiente diagrama muestra la volatilidad de elementos de evidencia en un sistema.

Fig. 2.1: Volatilidad de diferentes fuentes de evidencia

En análisis forense para dispositivos móviles por la diferencia de arquitectura y sistema operativo no siempre es posible aplicar el principio de volatilidad, sin embargo se debe tener presente para no perder de vista la fragilidad de la evidencia digital.

2.3 Metodología

El campo del cómputo forense obtiene sus bases del método científico. Aunque cada caso es diferente, ya sea por objetivos, alcance, ambiente o tecnología involucrada una investigación se puede dividir en 4 actividades fundamentales, las cuales a su vez tienen varias actividades subordinadas:

2.3.1 Preparación.

Mucho antes de que un investigador forense sea llamado a un incidente, existe toda una serie de preparativos que se deben de realizar como son la capacitación y actualización de los recursos humanos y materiales, insumos, adaptadores, etc. Está actividad en general es pasada por alto, lo que conlleva a grandes problemas e incluso a perder un caso ante un juzgado. Algunos ejemplos específicos de preparación se pueden mencionar, verificación del equipo de bloqueo de escritura, revisar que se cuenta con los adaptadores para tarjetas de memoria y SIM, etc.

2.3.2 Aseguramiento y adquisición de evidencia.

El especialista durante la respuesta a un incidente verifica el suceso, asegura el lugar/dispositivo y levanta la evidencia contenida en el dispositivo/sistema para su análisis posterior. Es en esta fase en la que se obtiene la información inicial del caso y se hacen las primeras observaciones, a esta etapa también se le conoce como examen forense.

Es importante que durante esta fase se preste atención en la integridad de la evidencia y que se cuente con los mecanismos necesarios para que durante el resto de la investigación se mantenga.

2.3.3 Investigación y análisis.

Como primer paso se desarrolla una hipótesis que explique las observaciones realizadas, es muy importante mantener un punto de vista objetivo y no caer en el error común de solo buscar elementos que comprueben nuestra hipótesis. Una de las mejores prácticas es buscar elementos que refuten nuestra hipótesis.

Con el objetivo de encontrar artefactos que sustenten o refuten las hipótesis del investigador se debe llevar a cabo un análisis minucioso. Entre los artefactos que generalmente se buscan durante una investigación se encuentran las bitácoras, historial de navegadores de internet, archivos ejecutables, archivos borrados, historial de ejecución de comandos, etc.

Con los artefactos que se descubren en esta fase se evalúa nuestra hipótesis para poder demostrar su veracidad. La evaluación de la hipótesis es un proceso de Investigación y experimentación, ya que se tiene que recurrir a experimentos en ambientes controlados para poder determinar los efectos de ciertas acciones. En computo forense tradicional se puede utilizar una maquina virtual para montar la imagen del equipo bajo investigación y interactuar con el par observar los resultados en dichas acciones. En cómputo forense para dispositivos móviles basados en Windows Mobile se puede utilizar un equipo del mismo

modelo bajo investigación o utilizar el simulador para Windows Mobile que se utiliza para desarrollo de aplicaciones.

2.3.4 Conclusiones y Reporte de resultados.

Para finalizar se genera un reporte de los resultados obtenidos, el cuál debe incluir un resumen ejecutivo y una descripción detallada de los sucedido en el incidente apoyándose en los hallazgos y elementos de evidencia que respaldan las conclusiones del reporte. Es importante mantener presente la audiencia a la que se dirige nuestro reporte ya que por lo general se compone de gente que no es especialista en el área de sistemas, por lo que el analista debe poder expresar sus hallazgos en un vocabulario lo menos técnico posible. También existe la posibilidad del que el analista tenga que presentarse ante una corte para respaldar lo presentado en el reporte.

Las actividades de la metodología general se pueden apreciar en el siguiente diagrama:

Fig. 2.2: Metodología general de investigación forense

Durante la investigación es posible que la evidencia nos lleve a tener que asegurar y llevar a cabo la adquisición y posterior análisis de otros equipos que están involucrados y que la evidencia encontrada nos ayuda a ubicar, por lo que esto se representa con la flecha de doble sentido en la fase de Investigación y Análisis en el diagrama.

Existen diversas metodologías que se han desarrollado a lo largo de los años, sin embargo las diferentes actividades que presentan pueden ser agrupadas en las cuatro actividades principales que se muestran en el diagrama. Es importante aclarar que la metodología que

aquí se presenta es independiente de las herramientas, y se puede llevar acabo utilizando herramientas de código abierto o comerciales, pero el uso de herramientas de código abierto le dan la ventaja de mantener la inversión requerida al mínimo, aunque su principal desventaja es que carecen del nivel de integración que las herramientas comerciales ofrecen.

2.4 Principios y Mejores Prácticas

De forma independiente a la metodología es importante destacar los principios y mejores prácticas que ayudan en gran medida a que una investigación sea exitosa.

2.4.1 Minimizar la perdida de datos.

Buscar siempre que nuestra acciones no modifiquen/borren/corrompan la evidencia al adquirirla o analizarla. No siempre es posible garantizar que no se va afectar la evidencia por lo que en entonces es necesario primero comprender el impacto y alcance de la acción que se va llevar acabo y documentarla.

Estos son algunos ejemplos de escenarios en el que es necesario manipular el dispositivo/evidencia y necesariamente existen cambios/perdida de datos que son inevitables por la naturaleza del dispositivo:

- Un celular para el que no se tiene ni el hardware, ni el software para llevar a cabo la adquisición de su contenido, requiere que se navegue de forma manual para visualizarlo. Este tipo de situación era muy común a los inicios del cómputo forense en dispositivos móviles. En este caso siempre y cuando se busque la manera de documentarlo (por ejemplo video grabación) y se mantenga control de la cadena de custodia del elemento en cuestión, es aceptable

- Los programas de adquisición y análisis modernos como Oxygen Forensic Suite o Paraben Device Seizure, requieren de la instalación de agentes en los dispositivos que serán analizados, esta acción repercute en que se está instalando software en la memoria del equipo y con esta acción se puede estar sobrescribiendo información borrada pero que podría ser recuperada. La justificación de este tipo de acción radica en que la huella del controlador (tamaño del archivo) es muy pequeña por lo que el riesgo de sobrescribir es bajo, además que esa pérdida de información si existiera se ve compensada por toda la información a la que se tendrá acceso.

2.4.2 Mantener la integridad de la evidencia recolectada (Cadena de custodia).

Es fundamental contar con el mecanismo/proceso que permita controlar el acceso y auditar todo el proceso desde el análisis hasta la presentación de los resultados, mas adelante en este capítulo cuando se presente la generación de una imagen forense se hablará con más detalle de este punto.

2.4.3 Investigación detallada y metódica.

Se debe realizar un análisis minucioso de toda la evidencia que se tiene en una forma metódica que permita reproducir nuestros resultados. Para poder lograr este objetivo se debe documentar de forma detallada cada paso que se realice durante toda la investigación.

2.4.4 Considere que todos los casos terminaran por presentarse ante una corte.

Cada caso es único y depende del ambiente en que se desarrolla, por ejemplo si se trata de un caso interno para un corporativo existe la posibilidad que no se tenga que presentar ante una corte, sin embargo se debe tratar con la misma disciplina y metodología como si el caso fuera a presentarse de forma legal.

2.5 Dispositivos de Almacenamiento

Los dispositivos de almacenamiento digital poseen una estructura que permite el acceso eficiente de la información. Durante el análisis forense de un medio de almacenamiento digital como lo es un disco duro o una tarjeta de almacenamiento SD o SDHC en el caso de los smartphones, generalmente la adquisición es de la unidad de almacenamiento completa.

2.5.1 Estructura

Para poder tener un entendimiento adecuado sobre los dispositivos de almacenamiento es necesario conocer su estructura interna. El siguiente diagrama nos muestra la estructura de unidad de almacenamiento a nivel general de organización interna:

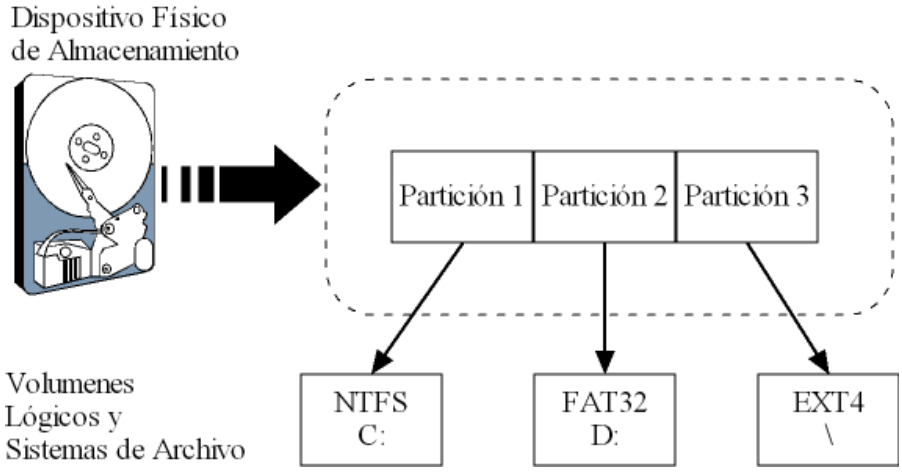

Fig. 2.3: Estructura general de un dispositivo físico de almacenamiento

Como se puede observar en términos generales un dispositivo de almacenamiento contiene una o más particiones y estas a su vez pueden contener uno o más volúmenes. El tratamiento que se le da a cada volumen es dependiente del sistema operativo y el sistema de archivo asociado, por lo que una partición es una colección de sectores consecutivos en un volumen (Carrier 2005 [82-88]).

El tipo de partición más común es el tipo DOS tanto en los equipos de escritorio y portátiles, como en algunos dispositivos móviles como los basado en Windows mobile. La estructura básica de un dispositivo de almacenamiento que utiliza particiones DOS es el siguiente:

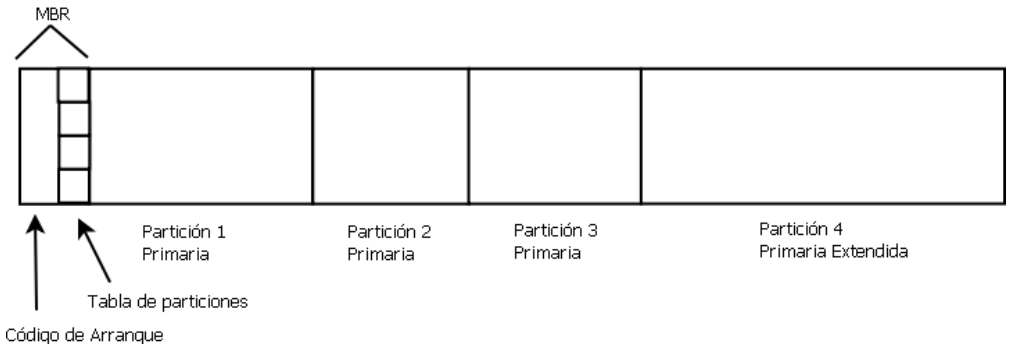

Fig. 2.4: Estructura de una partición PC-DOS

Los primeros 512 bytes, que representan el 1er sector físico de un dispositivo con particiones DOS contiene el Registro Maestro de Arranque (MBR: Master Boot Record) el cual contiene el código ensamblador para arranque y la tabla de particiones del dispositivo. La siguiente tabla muestra la estructura del MBR.

Posición (bytes)	Descripción	Mandatorio
0-445	Código de Arranque	No (Solo es necesario si el dispositivo es de arranque del Sistema)
446-461	1ª Entrada de la Tabla de partición	Si
462-477	2ª Entrada de la Tabla de partición	Si
478-493	3ª Entrada de la Tabla de partición	Si
494-509	4ª Entrada de la Tabla de partición	Si
510-511	Firma (0xAA55)	No

Tabla 2.1: Estructura de la tabla de particiones PC-DOS

Cada una de las 4 posiciones de la tabla de particiones tiene la siguiente estructura:

Posición (Bytes)	Descripción	Mandatorio
0	Bandera de arranque	No
1-3	Dirección CHS Inicial	Si
4	Tipo de Partición	Si
5-7	Dirección Final CHS	Si
8-11	Dirección LBA Inicial	Si
12-15	Tamaño en sectores	Si

Tabla 2.2: Estructura de una entrada de la tabla de particiones

El direccionamiento CHS solo funciona para dispositivos de almacenamientos menores a 8GB, mientras que el direccionamiento LBA funciona para dispositivos de almacenamiento en el rango de Terabytes.

La tabla nos muestra la estructura que tiene una entrada de la tabla de particiones, las cuales contienen el tipo de partición, las direcciones de inicio y final usando direccionamiento CHS (Cylinder, Head, Sector), la dirección inicial LBA (Logical Block Address) y su tamaño en sectores. El direccionamiento CHS era forma en que se podía direccionar un sector específico de un disco duro, pero con forme la capacidad de los discos duros aumento se busco un nuevo modelo de direccionamiento el cual es el LBA (Logic Block Address) que asigna una dirección consecutiva a cada sector.

El byte en la posición 4 de una entrada de la tabla de partición indica el tipo de partición del que se trata, la siguiente tabla da algunos de los valores más comunes.

Valor (hex)	Descripción
0x00	Vacio
0x01	FAT12
0x04	FAT16, 16 a 32 MB
0x06	FAT16, 32MB a 2GB
0x07	NTFS
0x0c	FAT32, LBA
0x82	Solaris X86
0x83	Linux
0xa8	Mac OSX

Tabla 2.3: Tipos de partición más comunes

2.5.2 Identificación del dispositivo físico

A continuación se da un breve ejemplo de cómo conocer la estructura interna de un dispositivo de almacenamiento y las consideraciones que se tienen que tomar en cuenta antes de la adquisición. En este ejemplo se tiene una memoria SD de 2GB conectada a través de un lector de tarjetas a un puerto USB, el sistema operativo de la estación de de trabajo es Windows XP SP3.

Primero se debe conocer cuál es la designación como dispositivo físico de la tarjeta de memoria. Para este propósito se utilizara una herramienta de código abierto llamada *dskwipe*, esta herramienta es para el borrado seguro de dispositivos de almacenamiento pero la opción -l nos despliega los dispositivos físicos y lógicos del sistema

```
C:\TSK-3.2.3\bin>dskwipe -l
Device Name                          Size Type      Partition Type
------------------------------ --------- --------- --------------------
\\.\PhysicalDrive0                  80.0 GB Fixed
\\.\PhysicalDrive1                   2.0 GB Removable
\Device\Harddisk0\Partition0        80.0 GB Fixed
\Device\Harddisk0\Partition1        80.0 GB Fixed     NTFS
\Device\Harddisk1\Partition0         2.0 GB Removable
\Device\Harddisk1\Partition1         2.0 GB Removable
\\.\C:                              80.0 GB Fixed     NTFS
\\.\D:                               2.1 GB Fixed     FAT32
\\.\E:                               2.0 GB Removable
```

La unidad que nos interesa es \\.*PhysicalDrive1* que representa la unidad física de la tarjeta SD de 2 GB, para obtener su estructura de particiones se utiliza la utilería *mmls* del Sleuth Kit. La opción *–t dos* es para indicarle a *mmls* que se trata de una tabla de particiones tipo DOS y se le pasa el dispositivo físico \\.*PhysicalDrive1* como parámetro

```
C:\TSK-3.2.3\bin>mmls -t dos \\.\PhysicalDrive1
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

     Slot    Start        End          Length       Description
```

```
00:  Meta    0000000000   0000000000   0000000001   Primary Table (#0)
01:  -----   0000000000   0000000136   0000000137   Unallocated
02:  00:00   0000000137   0003842047   0003841911   DOS FAT16 (0x06)
```

El resultado nos muestra la ubicación de la tabla de particiones en el sector 0 y con una tamaño de 1 sector (512 Bytes) y una sola partición FAT16 que comienza en el sector 137 con un tamaño de 3841911 sectores, que nos da un tamaño de 1875.9 MB. También se puede apreciar que existe una sección que no tiene ninguna asignación entre la tabla de partición y la partición, con un tamaño de 137 sectores con un tamaño de 68KB, estas área aun cuando no se encuentran asignadas a una partición y por ende no son accesibles por el SO directamente, pueden contener información de una partición anterior o información oculta.

2.6 Ordenamiento a nivel de bytes (Endianess)

Los sistemas de cómputo dependiendo de su arquitectura (CPU y sistema operativo) almacenan la información con distintas formas de ordenamiento, a este concepto se le conoce con el término *endianess* y su importancia radica en que si quiere o un analista se ve en la necesidad de interpretar la información almacenada en un dispositivo físico o en su imagen forense es necesario saber que formato utiliza el equipo bajo análisis para tener éxito.

Los formatos de ordenamiento más comunes son el *big endian* y el *little endian*, la diferencia entre ambos mecanismos de ordenamiento radica en que byte se ordena primero. Una arquitectura de hardware que utiliza big endian almacena primero el byte más significativo de una estructura de datos en la dirección más baja, mientas que el little endian primero almacena el byte menos significativo en la dirección más baja

2.6.1 Big Endian.

Este tipo de ordenamiento por lo general se encuentra en arquitecturas basadas en POWER, PowerPC, PA-RISC y ESPARC ejecutando sistemas operativos como MAC OS, Amiga, Linux, MVS, y Solaris .

2.6.2 Little Endian

El ordenamiento little endian se encuentra por lo general en plataformas 6x86, e Itanium ejecutando la familia de sistema operativo Windows a 32 y 64 bits, OpenVMS, Linux, Solaris y MAC OS. El siguiente ejemplo muestra cómo funcionan ambos tipos de ordenamiento Little.

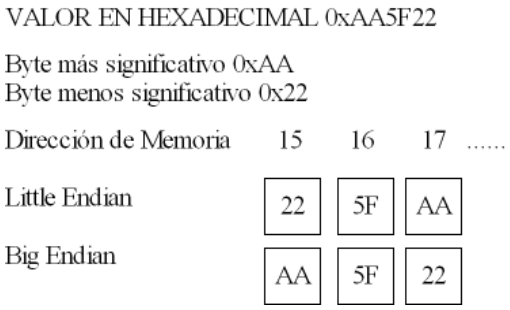

Fig. 2.5: Ordenamiento de bytes little endian y big endian

2.7 Creación de una imagen forense

A continuación se presentará una breve introducción a la creación de imágenes forenses de dispositivos de almacenamiento. Como se explicó con anterioridad, un dispositivo de almacenamiento (disco duro, tarjeta de memoria, CD) contiene estructuras internas que permiten la organización de la información para su consulta y modificación. Estas estructuras son de suma importancia ya que ofrecen evidencia de la actividad en un sistema de archivos, esta información conocida de forma general como metadatos es muy frágil por lo que su adquisición y preservación son la fase más importante del proceso de análisis forense

2.7.1 Tipos de Adquisición.

En el campo del computo forense existen dos tipos de adquisición, la física que es una copia bit a bit del dispositivo de almacenamiento completo, lo que se considera mejor evidencia ya que se puede recuperar información borrada u oculta dentro del dispositivo, encontrar trazas o fragmentos de otros sistemas de archivo.

El segundo tipo de adquisición es la adquisición lógica, la cual se lleva a cabo una copia de la información activa en las particiones de un dispositivo de almacenamiento. Por información activa se entiende aquella información que es visible para el sistema operativo, lo cual tiene como desventaja que no se puede recuperar información oculta/borrada del mismo, tampoco se tienen acceso a las áreas no asignadas del dispositivo de almacenamiento.

No siempre es posible obtener una imagen física del dispositivo de almacenamiento, en especial cuando trata de dispositivos móviles como son los teléfonos smartphones, ya que por la diversidad de hardware e implementación de software, por lo que habrá ocasiones en que se tendrá que trabajar con la imagen lógica. La conexión a nuestra estación de trabajo para la adquisición de nuestra imagen forense se muestra a continuación

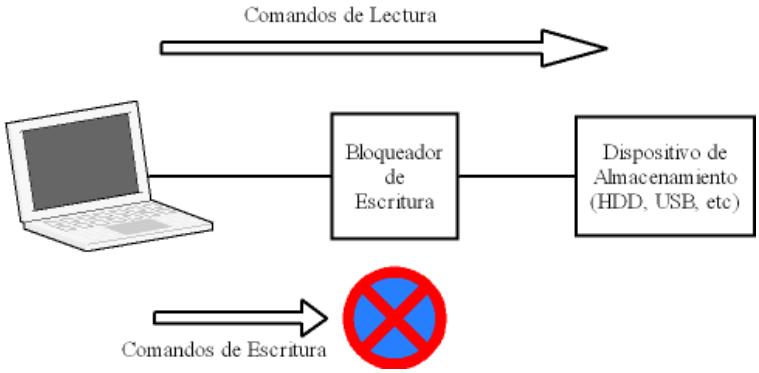

Fig. 2.6: Conexión de bloqueador de escritura para la adquisición de un dispositivo de almacenamiento (tarjeta SD)

```
En el siguiente ejemplo se mostrara como realizar una imagen física de un dispositivo de
almacenamiento, en este caso es la misma tarjeta de memoria SD a la que se hace referencia
en el ejemplo de identificación de unidad física en la sección "Unidad de almacenamiento"
de este capítulo. Una vez conectado nuestro dispositivo de lectura y bloqueador de
escritura, es necesario identificar el dispositivo  físico que se desea copiar, lo cual se
realiza por medio de herramientas como dskwipe y mmls.
C:\Forense\dcfldd>dcfldd if=\\.\PhysicalDrive1 of=c:\Evidencia\SDimage1.dd hash=md5,sha1
md5log=c:\Evidencia\SDimage1.md5 sha1log=c:\Evidencia\SDimage1.sha1 conv=sync
59904 blocks (1872Mb) written.dcfldd:\\.\PhysicalDrive1: Input/Output error

60032+0 records in
60032+0 records out
```

La sintaxis del comando *dcfldd* es la siguiente:

- **if** es el archivo de entrada, en este caso el dispositivo físico 1 el cual ha sido identificado como la tarjeta SD que se desea copiar
- **of** es el archivo de destino donde se quiere almacenar la copia
- **bs** es el tamaño de los bloques de que se desea manejar y acepta diferentes indicadores de unidad como M(Megabye), K(Kilobyte) si no se utiliza indicador de unidad la herramienta asume por defecto que son bytes
- **hash** indica loa algoritmos a utilizar para la generación de resumen (digest), si no se indica utiliza por defecto MD5, en este ejemplo se utiliza MD5 y SHA1

md5log y *sha1log* guardan cada uno el resultado del algoritmo correspondiente en el archivo indicado como parte del parámetro, en este caso el resultado de de MD5 se guarda en el archivo *C:\Evidencia\SDimage.md5* y el de SHA1 en el archivo *C:\Evidencia\SDimage.sha1*

El parámetro *conv* permite pasar opciones de formato de conversión para la salida, en este caso la opción *sync*, rellena los bloques al tamaño indicado en el parámetro *bs*

Lo que el comando va a realizar es una copia bit a bit de la tarjeta SD completa y será almacenado en el archivo *C:\Evidencia\SDmemory.dd* junto con sus cadenas MD5 y SHA1. La extensión del archivo es arbitraria, aunque por lo general se le da la extensión dd o raw para indicar la herramienta con la que se llevo a cabo la imagen o indicar el formato

El formato de la copia que se ha obtenido por medio de la herramienta *dcfldd* se le conoce como RAW o crudo y aunque es el formato con mayor compatibilidad presenta la desventaja de que ocupa al menos tanto espacio como el dispositivo original. Existen otros formatos para imágenes forenses lo cuales se describen más adelante.

2.7.2 Integridad de la evidencia (Cadena de custodia)

La cadena de custodia es el proceso por el cual se busca garantizar la integridad de la evidencia que se asegura y poder auditar el proceso de análisis forense, este concepto proviene del campo forense tradicional. E objetivo principal es poder demostrar que la evidencia no ha sido manipulada, modificada o contaminada desde que se toma control de ella. Los siguientes son algunas de las mejores prácticas para el manejo de evidencia/cadena de custodia

- Llevar un registro detallado de cada pieza de evidencia física (ej: dispositivos, tarjetas de memoria, etc.) o lógica (ej: imagen forense de un dispositivo de almacenamiento)
- Cada persona que entra en contacto o que tiene acceso a la evidencia debe quedar registrada en una bitácora
- Nunca llevar a cabo el análisis sobre la evidencia original (cuando sea posible).
- Siempre sacar al menos una copia de la imagen forense y llevar a cabo los análisis de pruebas sobre la segunda copia.
- Tener un área dedicada que cuente con un control de acceso físico donde la evidencia se almacena.

La integridad de las imágenes forenses se logra a través de la firma digital criptográfica, también conocidas como hash. Una firma digital se obtiene al crear una cadena de identificación única conocida como boletín (digest). Los algoritmos matemáticos que permiten genera la firma digital se definen como procedimientos deterministas que toman un bloque arbitrario de datos y regresan como resultado una cadena de bits de tamaño fijo, la cual de haber un cambio accidental o intencional en los datos de origen resultará en una nueva cadena al aplicarse el algoritmo, lo cual es indicador que la información ha sido alterada. Los datos a los que se le aplica el algoritmo se le conoce por lo general como mensaje y la cadena de longitud fija se le da el nombre de resumen de mensaje (message

digest) o resumen aunque su uso dentro de computo forense nos referimos a su firma digital.

Existen varios algoritmos para la generación de firmas digitales criptográficas, el siguiente cuadro da un resumen de las más utilizadas para la verificación de integridad de evidencia digital en cómputo forense.

Algoritmo	Tamaño en bits del Hash	Longitud de la cadena de salida (caracteres)
MD5	128	32
SHA1	160	40
SHA256	256	64

Tabla 2.4: Algoritmos para generación de hash más comunes en análisis forense

Hay que señalar que los algoritmos SD5 y SHA1 son los más utilizados por las herramientas de código abierto y comerciales, sin embargo a ambas se les han descubierto vulnerabilidades que ponen en duda su uso como herramientas para verificar la integridad. El ataque más conocido y que se ha demostrado que se puede llevar acabo de forma práctica es el ataque de colisión en el cual dos bloques de datos diferentes (mensajes) generan el mismo hash (resumen), la paradoja del cumpleaños demuestra que en un conjunto de *n* personas seleccionadas al azar se encuentran 2 con el mismo día de cumpleaños, conforme *n* se incrementa, la probabilidad se incrementa muy rápido.

La comunidad forense en general continúa usando MD5 y SHA1 como sus principales mecanismos para la comprobación de integridad en evidencia digital, debido a que aunque son vulnerables a los ataques de colisión y de cumpleaños, la infraestructura necesaria no era de fácil acceso. Esto está cambiando con la introducción de unidades de procesamiento gráfico o GPUs comerciales como la línea GeForce de NVDIA, los cuales permiten calcular millones de hashes por segundo. Ante este nuevo panorama se debe comenzar a considerar utilizar algoritmos más seguros como el SHA256 de la familia SHA2 el cual genera una cadena de resumen de 64 caracteres de longitud y no se han encontrado ataques prácticos como en el caso de MD5 y SHA1. Otra opción es el utilizar sobre la misma pieza de evidencia los dos algoritmos más comunes para hash que son MD5 y SHA1 lo que minimiza la posibilidad de se lleve a cabo un ataque exitoso sobre ambos algoritmos de forma simultánea.

Se puede utilizar herramientas de código abierto para la verificación de integridad de nuestra evidencia digital, algunos funcionan en línea de comando como el md5sum

sha1sum en Linux y otras ofrecen una interface gráfica como HashCalc de Lavasoft para plataforma Windows

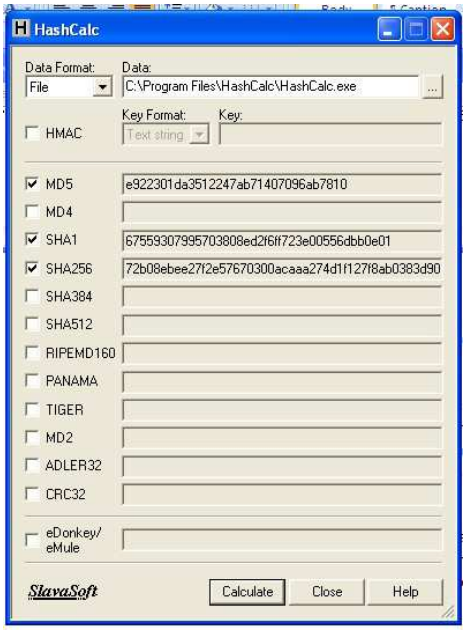

Fig. 2.7: Herramienta HashCalc para cálculo de hash para verificar la integridad de un archivo

2.7.3 Formatos para imágenes forenses

Existen diferentes formatos para la generación de un archivo que contiene la imagen de un dispositivo de almacenamiento

- **Crudo (RAW).** Este formato es el más antiguo y sus orígenes se encuentran con las herramientas de respaldo en UNIX como dd, tiene la ventaja de tener la mayor compatibilidad entre herramientas forenses, pero no ofrece otras características avanzadas como son el compresión, encriptación o mecanismos de integridad integrados al formato. Cuando se utiliza este formato es recomendable estimar un 30% de espacio extra en el dispositivo donde se va almacenar la imagen forense, ya que por discrepancias en las geometrías de los dispositivos de almacenamiento puede haber un incremento en espacio ocupado por la imagen

- **AFF (Advanced Forensic Format).** Este formato de código abierto surge de la necesidad de ofrecer funcionalidad avanzada como lo empezaron a ofrecer los formatos comerciales como EWF. El formato AFF ofrece compresión desde un 50% hasta un 90% de reducción, además permite integrar metadatos como son la firma digital del archivo generado e información creada por el analista durante la adquisición.

- **EWF (Expert Witness Format).** Es un formato propietario creado por Guidence Software, también se le conoce como E01, cual ofrece varias características avanzadas como son compresión, encriptación y mecanismo interno de integridad. El mecanismo de integridad que ofrece este formato se basa en el almacenamiento de de metadatos como son el nombre del analista, fecha y firma digital, pero además la imagen es guardad en bloques predeterminados y a cada bloque se le anexa un valor CRC (Cyclic Redundancy Check) el cual permite ubicar que bloque ha sido alterado/corrompido.

2.7.4 Extracción de las particiones de una imagen forense

Aunque la mayoría de las herramientas forenses realizan de forma automática la extracción de las particiones a partir de una imagen de un dispositivo físico, es necesario conocer cómo llevarlo a cabo de forma manual si la situación así lo requiere. Aquí se presentara dos técnicas para identificar las particiones que se encuentran en una imagen forense para su posterior extracción utilizando *dd* o *dclfdd*.

Una vez que se tiene la imagen forense se abre en un editor hexadecimal y utilizando las tablas 1.1, 1.2 y 1.3 obteniendo los valores de inicio y tipo tamaño de cada partición. En este ejemplo se utilizo la versión demo de Winhex, pero se puede llevar a cabo con cualquier editor de texto.

Como el dispositivo (tarjeta SD) proviene de una laptop, el ordenamiento de bytes para la decodificación es little endian, el concepto de ordenamiento es muy importante cuando se realiza la decodificación de evidencia digital a mano.

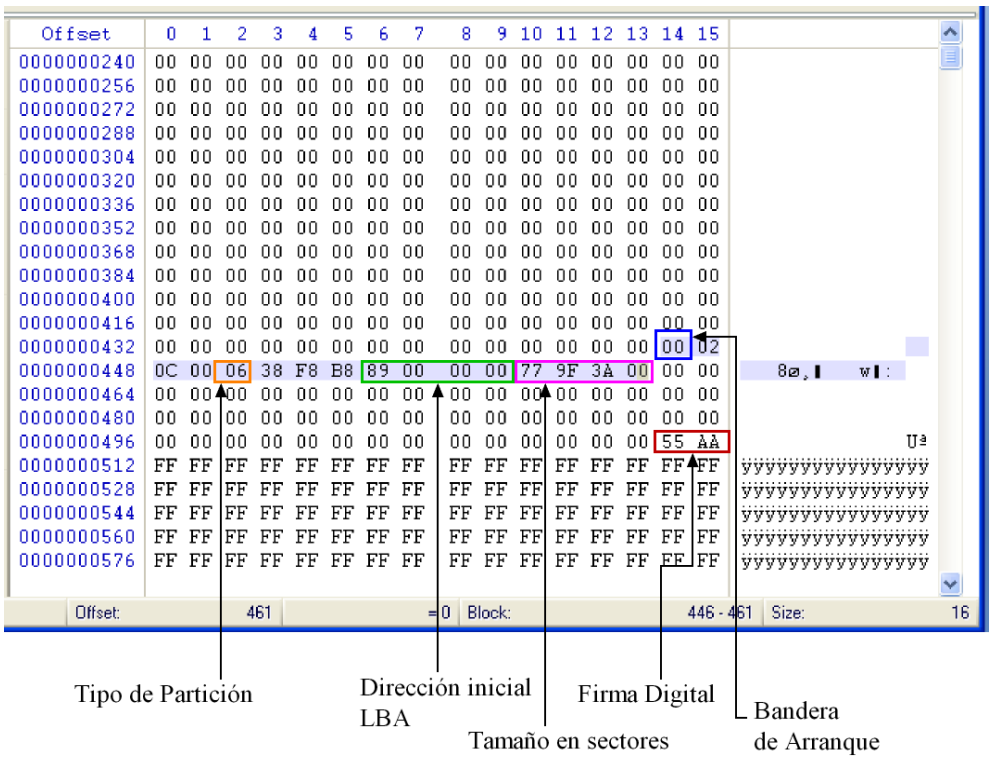

Fig 2.8: Análisis del primer sector de la imagen forense un dispositivo de almacenamiento y su tabla de particiones

En la imagen se ve el primer sector (512 bytes) de la imagen creada, de acuerdo a la información de las tablas se puede localizar la primera partición (byte 446 a 461) la cual nos indica que es una partición que no funciona para arranque de sistema (byte 446), también nos indica que esta es una partición FAT16 (byte 450, valor 0x06) y la dirección inicial LBA es el sector 137 (bytes 454 al 457, valor 0x89) y que el tamaño de la partición es 3841911 sectores (bytes 458 al 461, valor 0x3A9F77). Por último se observa ver la firma (0xAA55) en los bytes 510-511.

Los otros valores que no se muestran son las direcciones CHS de inicio y fin, que se pueden calcular utilizando la información de las tablas de este capítulo, además que para sobreponerse a la limitante de no poder direccionar particiones mayores a 8.1GB el direccionamiento LBA se convertido en el estándar para direccionamiento de dispositivos de almacenamiento. Otra forma de obtener la tabla de partición es la utilización de *mmls* como se muestra a continuación:

```
C:\Forense\bin>mmls -i raw c:\Evidencia\SDimage.dd
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

     Slot    Start        End          Length       Description
00:  Meta    0000000000   0000000000   0000000001   Primary Table (#0)
01:  -----   0000000000   0000000136   0000000137   Unallocated
02:  00:00   0000000137   0003842047   0003841911   DOS FAT16 (0x06)
```

Como se puede observar los cálculos hechos a mano nos arrojan los mismos resultados que *mmls*, aunque la mayor parte del tiempo se utilizan herramientas que automaticen este tipo de operaciones, un analista forense debe de ser capaz de obtener los mismos resultados en caso de que no se tengan las herramientas necesarias, además que este nivel de conocimiento permite validar nuevas herramientas o poder dar una explicación adecuada de nuestras conclusiones.

Ahora que se tiene ubicada la partición nos hace falta extraerla, en esta ocasión se utiliza de nuevo *dcfldd* para poder obtener la imagen forense de la partición dentro de la imagen del dispositivo físico.

```
dcfldd if=C:\Evidencia\SDimage.dd of=C:\Evidencia\SDFAT16.dd  bs=512 skip=137 count=3841911
hash=md5sha1 md5log=C:\Evidencia\SDFAT16.md5 sha1log=C:\Evidencia\SDFAT16.sha1
```

La sintaxis del comando es muy parecida a la que utilizó para la creación de la imagen, los siguientes son los parámetros que requieren de explicación:

- *if=C:\Evidencia\SDimage.dd* en lugar de pasar el dispositivo físico como entrada, (el archivo que contiene la imagen forense)
- *skip=137* de los datos que se obtiene de la partición sabemos que comienza en el sector 137 (direccionamiento LBA) por lo que es necesario indicarle cuantos sectores saltarse antes de comenzar a leer de la entrada
- *bs=512* indica el tamaño de bloque que dcfldd obtiene de la entrada, en este caso es de 512bytes con lo que para asegurar que está leyendo sectores del dispositivo
- *count=3841911* es el tamaño en sectores de la partición, con esto se le indica a dcfldd el número de bloques que necesita leer y mandar al archivo definido en *of*

El comando extrae la partición y la almacena en el archivo C:\Evidencia\SDFAT16.dd y genera hashes MD5 y SHA1 en los archivs SDFAT16.md5 y SDFAT16.sha1 para el manejo de la integridad del archivo generado y la cadena de custodia.

Muchas herramientas hoy en día no requieren este paso y llevan a cabo la reconstrucción de las particiones de forma automática, sin embargo es importante saber el origen de los resultados de nuestras herramientas.

Capítulo 3. Análisis forense de la bitácora Ext3 para una empresa del sector de la construcción

3.1 El caso

En consultoría los trabajos se dan en mayor parte por recomendación de un cliente anterior, este fue el caso de una empresa de construcción cuyos archivos habían borrados de forma intencional. El nombre de la empresa por acuerdos de confidencialidad no puede ser nombrado por lo que la se llamara LucGale. LucGale mantenía información muy importante de sus proyectos en un servidor Linux Fedora 8, y el sistema de archivos instalado era Ext3. El servidor contaba con 2 unidades de disco y como es muy común los respaldos de la información eran prácticamente nulos. Esta segunda unidad de disco había sido remplazada debido a un daño físico justo después del borrado de los archivos por lo que toda la información que incluía entre otras cosas las bitácoras de acceso de las cuentas de usuario se habían perdido. El cliente no deseaba entrar en problemas de demanda legal y la persona la que se creía responsable ya había sido despedida, y la recuperación de datos del medio físico se considero fuera de su presupuesto. La única opción viable era intentar recuperar los archivos a partir de la bitácora del sistema de archivos Ext3.

Cabe hacer notar que la recuperación de información a partir de la bitácora aunque es una valiosa herramienta en los casos de investigación forense, su éxito depende en gran medida del tiempo entre el evento que llevo a la perdida de información hasta la creación de la imagen de la unidad en cuestión y el volumen de información leída/escrita en el dispositivo en este periodo de tiempo transcurrido.

La unidad de disco era de 200GB tipo PATA (Parrallel ATA) algo común en el mercado mexicano, aun cuando para ese año Western Digital acaba de presentar las primeras unidades con 2TB de capacidad. La imagen del disco se llevo a cabo en las instalaciones del cliente hacia un disco duro de 400GB ya que las mejores prácticas recomiendan que el disco donde se almacena la imagen sea al menos un 15% más grande que el disco original para evitar perdida de información por las diferencias de geometría aun entre discos del mismo tamaño.

El tiempo de la generación del disco se llevo a cabo los procedimientos descritos en el capítulo 2 y tomó unas 4 Hrs debido a que las unidades de disco se montaron en carcasa externas con interface USB 2.0 y el equipo con el que se realizo el trabajo de creación de imagen era un laptop Sony VAIO VGN-SR450G ejecutando Helix 2008 R3 desde un CD.

Una vez obtenida la imagen y tomando su firma digital MD5, se procedió a la recuperación de información utilizando la metodología que a continuación se describe y permitió la recuperación de 8 de los 10 archivos que el cliente requería en un día

3.2 El sistema de archivo Ext3 y la bitácora

El sistema de archivos Ext3 ofrece opciones únicas al investigador forense, para poder entender estas opciones primero se debe tener cierto conocimiento en como Ext3 opera y la composición de sus estructuras internas. Este es un resumen de las estructuras y funcionamiento del sistema de archivo, para mayor información o si el lector desea profundizar los conceptos de los que se hace referencia, puede consultar Brian File System Forensic Analysis de Brian Carrier y Forensic Discovery de Dan Farmer y Wietse Venema.

El sistema de archivos Ext3 fue creado por el Dr. Stephen C. Tweedie en 1999, es un sistema de archivos que hace uso de una bitácora o diario (journal en ingles) y es una evolución del sistema de archivos Ext2. Un sistema de archivos con bitácora, permite al sistema operativo mantener un registro de todos los cambios llevados a cabo en el sistema de archivos antes de que la información sea grabada en el dispositivo de almacenamiento. La bitácora es una estructura de datos circular que se almacena en un área especial del sistema de archivos. La idea de mantener una bitácora es minimizar el riesgo de una corrupción de los datos en el dispositivo de almacenamiento en caso de una perdida repentina de energía o una falla que conduzca a un paro total del sistema. El concepto de bitácora también se puede encontrar en otros sistemas de archivo como son NTFS, JFS, JFS2 y ReiserFS, los cuales ofrecen capacidades similares.

Para explicar el funcionamiento de Ext3, se utiliza un modelo conceptual de 5 capas el cual fue introducido por Brian Carrier [Carrier 2005]. Este modelo es utilizado de forma extensa en la comunidad forense. El modelo nos ofrece un marco de trabajo que permite describir y entender la mayoría de los sistemas de archivo en la actualidad y reduce su complejidad, lo que permite el desarrollo de técnicas y procedimientos de análisis forense, los cuales pueden ser utilizados en diferentes tipos de archivos

El modelo divide la información de un sistema de archivo (estructural y de usuario) en 5 categorías o capas, lo cual también ayuda a comprender como opera el sistema de archivo que se está analizando y poder entender donde se puede encontrar localizada la evidencia que se busca.

Las categorías definidas por el modelo son: Sistema de Archivo, Contenido, Metadatos, Nombre de Archivo y Aplicación. Toda la información que contiene un sistema de archivo pertenece a una de estas 5 categorías. En algunos sistemas de archivo, principalmente FAT

no es posible que se le aplique este modelo fácilmente como con otros sistemas de archivo, debido a su diseño. Sin embargo la validez no se ve afectada y es por esto que herramientas como *"The Sleuth Kit" (TSK)* se basan en este modelo.

A continuación se da una breve descripción de cada una de las categorías para al final mostrar de forma gráfica la relación que existe entre ellas

- **Sistema de Archivo.** En esta categoría se encuentra la información que describe la estructura del sistema de archivo, incluyendo el tamaño de las unidades de datos, posición relativa de las estructura, información de montaje, sector de arranque, etc.

- **Nombre de Archivo.** En esta categoría se incluyen las entradas de directorio donde el sistema de archivo almacena el nombre y el apuntador inicial de cada archivo. Es el equivalente a la tabla de contenido de un libro

- **Metadatos.** Contiene información que describe las características y estructuras de cada archivo, permisos, fechas de creación acceso y modificación, tamaño del archivo, apuntadores a los bloques de contenido/datos, etc.

- **Datos/Contenido.** En esta categoría encuentran las estructuras donde se almacena el contenido real de un archivo.

- **Aplicación.** Algunos sistemas de archivos contienen información que no es esencial para su operación pero que ofrece beneficios como recuperación y auditoria, en algunos casos este tipo de datos pueden ser habilitados por el usuario. Este tipo de información se puede encontrar como un archivo de sistema especial. Ejemplos de este tipo de datos son las bitácoras que se encuentran en EXT3, EXT4 y NTFS el manejo de cuotas de espacio en disco.

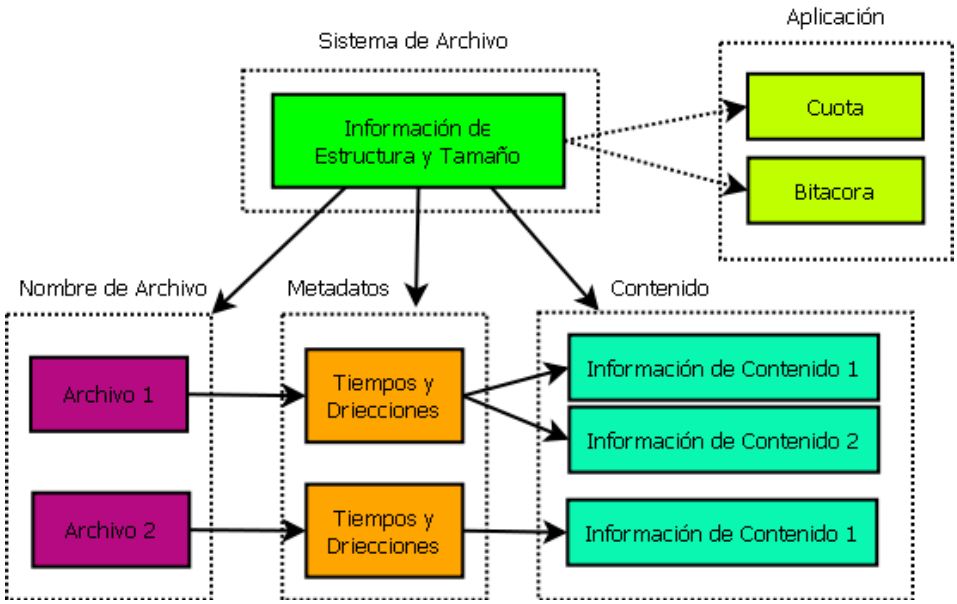

Fig 3.1: Modelo de sistema de archivo conceptual de 5 capas

Por lo general las primeras 4 capas son consideradas estructuras de datos esenciales, mientras que las estructuras de la quinta capa se consideran no esenciales u optativas (Carrier 2005 [174-175]). La mayoría de las técnicas y procedimientos se enfocan a las estructuras esenciales, sin embargo para la resolución de este caso fue necesario enfocarse a las estructuras de la quinta capa, lo cual proporciono información clave.

Aplicando las primeras 4 capas del modelo la relación de las estructuras de EXT3 es la siguiente:

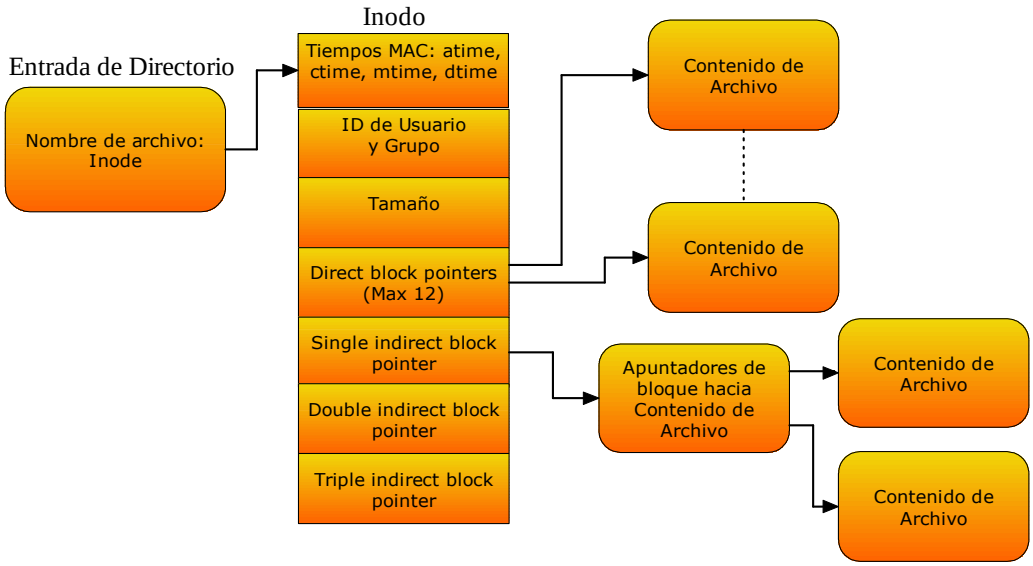

Fig 3.2: Relación entre el nombre de archivo inodo y bloques de contenido

La estructura que corresponde a la quinta capa se le conoce como diario o bitácora (journal en inglés). La estructura almacena los cambios del sistema de archivos. Ext3 ofrece 3 modos de operación de la bitácora, los cuales difieren en el tipo de información y como es almacenada, lo cual tiene impacto en el desempeño del sistema operativo. A continuación se da un resumen de los tres modos de operación:

- **Diario (Journal) -** Registra todos los cambios en los datos y metadatos del sistema de archivos. Este modo de operación minimiza la posibilidad de perder los cambios que se han hecho en cualquier archivo de un sistema de archivos Ext3. La desventaja de este modo de operación es que se tiene una disminución en el desempeño del sistema de archivos ya que los datos son escritos dos veces (una a la bitácora, el segundo al sistema de archivos).

- **Ordenado (Ordered) –** Solo se registran los cambios de los metadatos del sistema archivos (inodes), pero los cambios en los datos son llevados a cabo antes de que actualicen los metadatos asociados, de esta forma se mantiene sincronizados los registros de la bitácora con las operaciones de escritura. Este es el modo de operación por omisión de Ext3.

- **Respuesta (Write back) –** Solo registra los cambios al sistema de archivos (metadatos) pero no existe sincronización entre la bitácora y el sistema de archivos, ya que se utilice el mecanismo estándar de escritura del sistema de archivos para salvar los cambios a disco. Este modo es el más rápido pero el menos seguro en Ext3.

Desde el punto de vista forense el primer modo de operación (bitácora) es el que nos puede dar más información sobre las actividades del sistema de archivo y facilita la recuperación de archivos borrados, los otros dos modos solo permiten recuperar información de los metadatos. Desafortunadamente el comportamiento por omisión de Ext3 el modo bitácora. El analista forense debe mantener presente las diferencias, ventajas y desventajas de cada uno de los modos de operación de la bitácora de Ext3 durante una investigación.

En la mayoría de los casos la bitácora existe en el mismo sistema de archivo Ext3. Reside en un área especial dentro del sistema de archivos. La primera estructura que encuentra en una bitácora se le conoce como súper bloque de la bitácora (`journal superblock`) y mantiene la información el tamaño de bloque de la bitácora, número total de bloques para almacenamiento de la bitácora, ubicación de inicio de la bitácora, número de secuencia de la primera transacción, ubicación de la primera transacción e información general de la estructura de la bitácora. El mecanismo de la bitácora que permite rastrear los cambios en el sistema de archivo está basado en secuencias de transacciones, la cual está compuesta de la siguiente forma:

- **Bloque descriptor (Descriptor block):** Toda transacción comienza con un bloque que describe el inicio de la secuencia.
- **Bloque de Metadatos (Metadata block):** Existen uno o más bloques de metadatos por cada transacción, estos bloques son los que contienen los cambios que se realizan al sistema de archivos.
- **Bloque de Asignación (Commit block):** Dependiendo del modo de operación de la bitácora, este bloque indica el final de una transacción exitosa.
- **Bloque de Revocación (Revoke block):** Si ocurre un error durante una de las operaciones la transacción, se crea un bloque de revocación, el cual contiene una lista de los bloques del sistema de archivos que necesitan ser restaurados durante la revisión de consistencia.

La siguiente gráfica ilustra la composición de la bitácora:

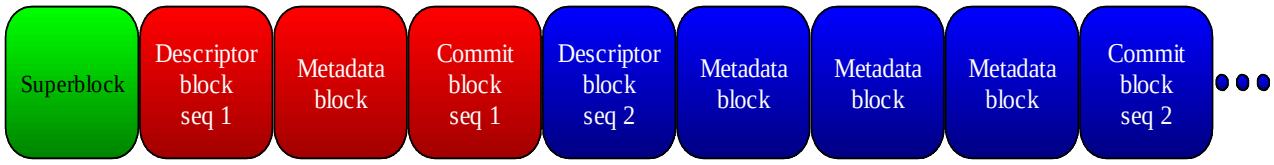

Fig 3.3: Estructura general de la bitácora en un sistema de archivos Ext3

Es interesante ver que el mecanismo de la bitácora actúa a nivel de bloques, y esto es el fundamento por el cual es posible desarrollar la técnicas aplicadas en este caso, ya que los metadatos (inodes) del sistema de archivos son modificado, el bloque completo donde reside los metadatos modificados es guardado en la bitácora. Esto implica que los metadatos vecinos al metadato modificado que residen en el mismo bloque también son almacenados en la bitácora. Este concepto recibe el nombre del "efecto de espectador", el cual es descrito por Dan Farmer y Witse Venema (Farmer y Venema 2007 [25-26]).

3.3 Ciclo de Vida de la Bitácora

Es necesario conocer el ciclo de vida de la bitácora de Ext3, ya que tiene un gran impacto sobre cómo se lleva a cabo la misma. La bitácora se reinicia cada vez que el sistema de archivos se monta de nuevo (por ejemplo al inicio del arranque del sistema) o si el espacio disponible para la bitácora se llena, por lo que la bitácora regresa al principio y sobres escribe el contenido como una lista circular; esto destruye evidencia que contiene la bitácora por lo que se recomienda obtener una imagen de la bitácora tan pronto sea posible, utilizando herramientas de TSK o tomando por una imagen completa del dispositivo para análisis posterior.

Para poder entender mejor, a continuación se muestran una serie de tomas en diferentes momentos de una bitácora Ext3 utilizando la herramienta jls del TSK. Esta herramienta permite el navegar la bitácora. Si se ejecuta el comando *jls sda6img.dd* sobre un archivo que contiene la imagen forense llamado sda6, el cual es un sistema de archivos Ext3, el cual se monta inmediatamente después de haber sido inicializado utilizando la herramienta mke2fs:

```
JBlk    Descriptrion
0:      Superblock (seq: 0)
1:      Unallocated FS Block Unknown
2:      Unallocated FS Block Unknown
3:      Unallocated FS Block Unknown
4:      Unallocated FS Block Unknown
5:      Unallocated FS Block Unknown
6:      Unallocated FS Block Unknown
7:      Unallocated FS Block Unknown
8:      Unallocated FS Block Unknown
9:      Unallocated FS Block Unknown
10:     Unallocated FS Block Unknown
[REMOVED]
```

Aquí se muestra el estado de la bitácora justo después que el sistema de archivos fue creado. Se observa que el número de secuencia en el súper bloque es 0 y todos los bloques son marcados como “no asignados” y que no hay ningún tipo de transacción o información almacenados. Ahora se monta el sistema de archivos:

```
JBlk    Description
0:      Superblock (seq: 0)
1:      Allocated Descriptor Block (seq: 2)
2:      Allocated FS Block 183
3:      Allocated Commit Block (seq: 2)
4:      Unallocated FS Block Unknown
5:      Unallocated FS Block Unknown
6:      Unallocated FS Block Unknown
7:      Unallocated FS Block Unknown
8:      Unallocated FS Block Unknown
9:      Unallocated FS Block Unknown
[REMOVED]
```

Cuando el sistema de archivos está siendo montado, se observa que el bloque 1 de la bitácora el número de secuencia comienza a incrementarse. Después de que una serie de archivos que son copiados se observa lo siguiente:

```
JBlk    Description
0:      Superblock (seq: 0)
1:      Allocated Descriptor Block (seq: 2)
2:      Allocated FS Block 183
3:      Allocated Commit Block (seq: 2)
4:      Allocated Descriptor Block (seq: 3)
5:      Allocated FS Block 295094
6:      Allocated FS Block 1
7:      Allocated FS Block 295095
8:      Allocated FS Block 295093
9:      Allocated FS Block 295595
10:     Allocated FS Block 0
[REMOVED]
```

Se aprecia que después de copiar los archivos que el número de operación se ha incrementado de forma secuencial; también nos muestra los bloques del sistema de archivo que han sido actualizados durante el proceso de copiado. A continuación se desmonta y se vuelve a montar el sistema de archivos:

```
JBlk    Description
0:      Superblock (seq: 0)
1:      Allocated Descriptor Block (seq: 16)
2:      Allocated FS Block 327682
3:      Allocated Commit Block (seq: 16)
4:      Unallocated Descriptor Block (seq: 3)
5:      Unallocated FS Block 295094
6:      Unallocated FS Block 1
7:      Unallocated FS Block 295095
8:      Unallocated FS Block 295093
9:      Unallocated FS Block 295595
10:     Unallocated FS Block 0
[REMOVED]
```

El número de secuencia continúa incrementándose, pero debido a que el sistema de archivo fue desmontado de forma limpia, la bitácora es reiniciada y la siguiente transacción que es registrada se almacena en el bloque 1 de la bitácora, con esto se puede observar que el mecanismo de operación de la bitácora sobrescribe la evidencia (bloques 2 y 3). También se puede notar que las transacciones restantes que contiene nuestra evidencia pertenecientes a los registros durante el tiempo del incidente ahora son marcados como *"no asignados" (unallocated)*.

Mientras otras operaciones ocurren en nuestro sistema de archivo de prueba como puede ser abrir el archivo o modificarlo, estas actividades don registradas en la bitácora:

```
JBlk    Description
0:      Superblock (seq: 0)
1:      Allocated Descriptor Block (seq: 16)
2:      Allocated FS Block 327682
3:      Allocated Commit Block (seq: 16)
4:      Allocated Descriptor Block (seq: 17)
5:      Allocated FS Block 182
6:      Allocated FS Block 1
7:      Allocated FS Block 183
8:      Allocated FS Block 295595
9:      Allocated FS Block 683
10:     Allocated FS Block 181
[REMOVED]
```

Como se observa el mecanismo de lista circular de la bitácora continua sobrescribiendo y eliminando evidencia como se puede observar en los bloques 4 a 10. Se debe tener en cuenta que si el modo de operación de la bitácora es basado en metadatos (ordenado/ respuesta) los bloques mostrados en la bitácora pueden contener algún tipo de metadatos, los cuales pueden ser parte de la tabla de inodos, los mapas de bits de los inodos o las direcciones de los bloques que contienen información. Si el modo de operación es tipo

"diario" (journal) entonces la información grabada en la bitácora también incluirá una copia de los bloques de datos con el contenido de los archivos.

Una última observación con respecto a la preservación de la evidencia en la bitácora: cuando se monta un sistema de archivos Ext3 (recuerde siempre se debe montar en forma de solo escritura), tome en cuenta que se debe tomar precauciones adicionales, de lo contrario la integridad de la imagen se verá comprometida en el momento que la bitácora se reinicie y se ejecute sobre el sistema de archivos en el momento que este sea montado (esto aplica también a las imágenes forenses). Para demostrar la necesidad las precauciones aquí mencionadas, se toma una firma de una imagen forense que contiene un sistema de archivos Ext3, se monta y después se desmonta, tomando una segunda firma de la imagen, para comparar ambas firmas como se muestra en este ejemplo:

```
[root@Akula1 workbench]# md5sum sda7img.dd > sda7img.md5
[root@Akula1 workbench]# mount -t ext3 -o loop sda7img.dd test
[root@Akula1 workbench]# umount test
[root@Akula1 workbench]# md5sum sda7img.dd > sda7img.md5a
[root@Akula1 workbench]# more *md5*
::::::::::::::
sda7img.md5
::::::::::::::
2bd7c92834e8396362892bb945106ac9  sda7img.dd
::::::::::::::
sda7img.md5a
::::::::::::::
831cbea560ec9d48d28bc5219b5eff58  sda7img.dd
```

Como se observa al montar la imagen sin la opción *"ro" (read-only)* la bitácora es reiniciada/ejecutada y las firmas son diferentes, recuerde que si no es posible demostrar que la imagen es la misma de la que se obtuvo en el momento de recolectar la imagen su validez en un proceso judicial queda cuestionado e incluso puede ser descartada por completo.

3.4 Proceso de borrado de un archivo: Ext2 Vs Ext3

En esta sección revisara como un sistema de archivos Ext2 (sin bitácora) lleva acabo el borrado de un archivo y como se compara con el procedimiento de Ext3, el sistema de archivos Ext2 se encuentra en un archivo de imagen forense con el nombre "sda5img.dd" el cual se encuentra en formato crudo y fue generado con la herramienta dd.

En Ext2 el borrado de un archivo se puede resumir como el marcado como "no asignado" de la entrada de directorio, inodo y bloques de datos que conforman un archivo, esta marcación se lleva a cabo en los mapas de bits de bloques y en el de inodos de cada grupo de bloques. En este ejemplo se tiene un archivo conocido como reference.pdf con un tamaño de 562378 bytes, su estructura es la que se muestra a continuación:

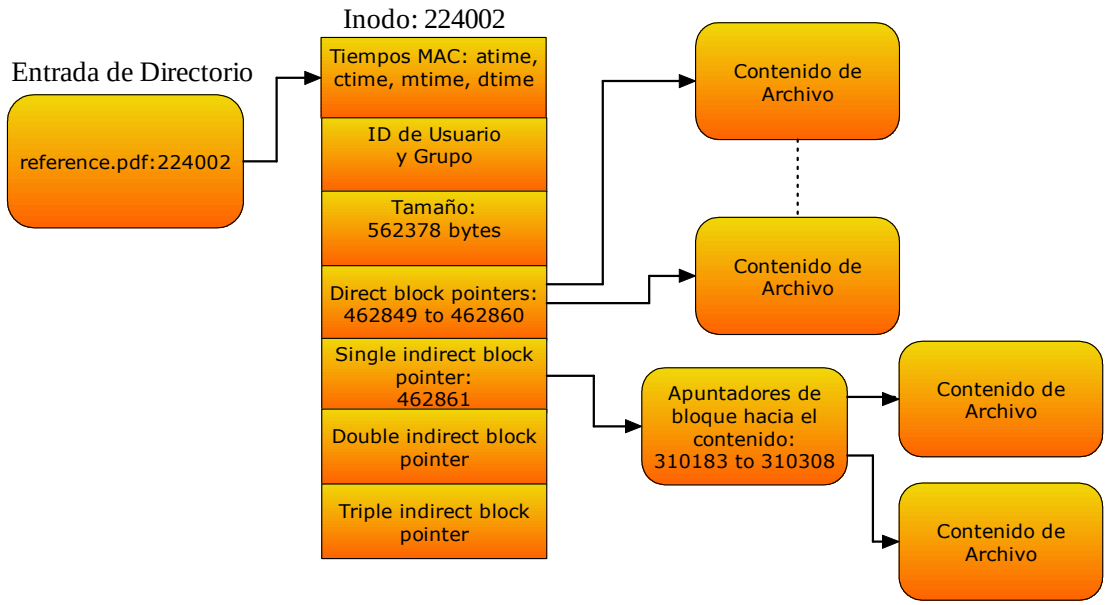

Fig 3.4: Estructura de un archivo antes de ser borrado en Ext2

Cuando el archivo "reference.pdf" es borrado en Ext2 el inodo y los bloques de contenido son marcados como "no asignados" para que el sistema operativo los pueda reutilizar de ser necesario, pero toda la información que forma el archivo todavía existe como lo muestra el siguiente diagrama:

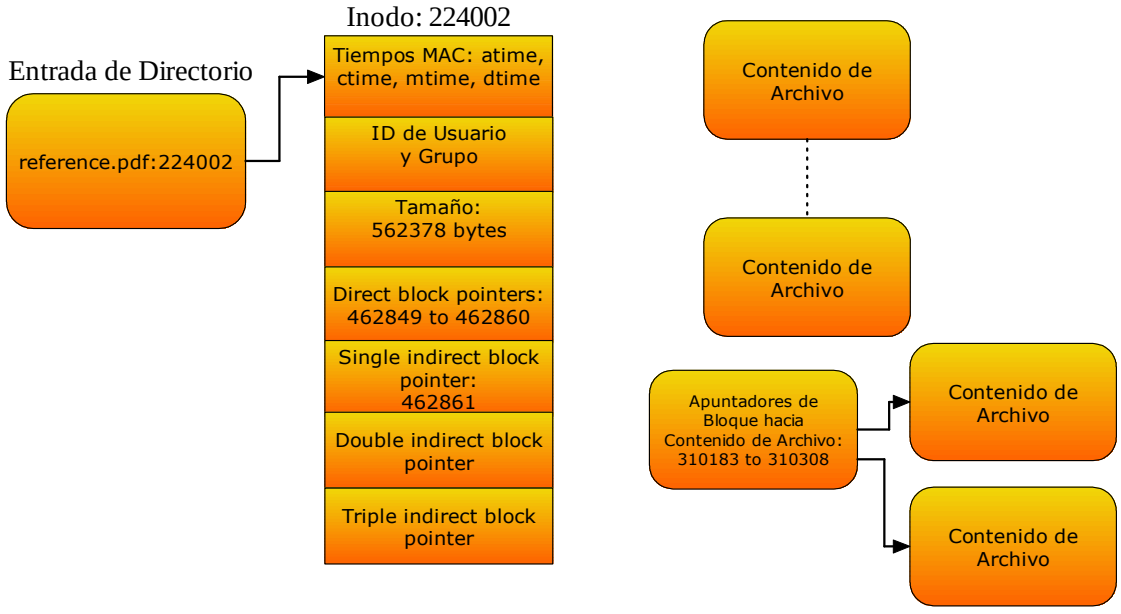

Fig 3.5: Estructura de un archivo después de ser borrado en Ext2

Esto se puede observar en la imagen de un sistema utilizando dos herramientas de TSK, *ils* e *istat*. A continuación se puede apreciar la salida que se obtiene al utilizar la herramienta ils en una imagen que contiene un sistema de archivos Ext2 utilizando la opción -r para obtener el listado completo de todos los inodos borrados.

```
[root@Akula1 workbench]# ils -r sda5img.dd
class|host|device|start_time
ils|Akula1||1194288876
st_ino|st_alloc|st_uid|st_gid|st_mtime|st_atime|st_ctime|st_mode|st_nlink|st_size|st_block0
|st_block1
[REMOVED]
224002|f|0|0|1182963421|1194249392|1194282669|100644|0|562378|462849|462850
224003|f|0|0|1182963371|1194249453|1194282669|100644|0|69330|462988|462989
[REMOVED]
```

Tomando el primer inodo de la lista (inodo 224002), nos muestra un probable archivo con un tamaño de 562378 bytes. A continuación se obtiene la información del inodo 224002 utilizando *istat*.

```
[root@Akula1 workbench]# istat sda5img.dd 224002
inode: 224002
Not Allocated
Group: 14
Generation Id: 231094706
uid / gid: 0 / 0
mode: -rw-r--r--
size: 562378
num of links: 0

Extended Attributes  (Block: 689)
security.selinux=root:object_r:file_t:s0

Inode Times:
Accessed:       Mon Nov  5 01:56:32 2007
File Modified:  Wed Jun 27 11:57:01 2007
Inode Modified: Mon Nov  5 11:11:09 2007
Deleted:        Mon Nov  5 11:11:09 2007

Direct Blocks:
462849 462850 462851 462852 462853 462854 462855 462856
462857 462858 462859 462860 462862 462863 462864 462865
462866 462867 462868 462869 462870 462871 462872 462873
462874 462875 462876 462877 462878 462879 462880 462881
462882 462883 462884 462885 462886 462887 462888 462889
462890 462891 462892 462893 462894 462895 462896 462897
462898 462899 462900 462901 462902 462903 462904 462905
462906 462907 462908 462909 462910 462911 462912 462913
462914 462915 462916 462917 462918 462919 462920 462921
462922 462923 462924 462925 462926 462927 462928 462929
462930 462931 462932 462933 462934 462935 462936 462937
462938 462939 462940 462941 462942 462943 462944 462945
462946 462947 462948 462949 462950 462951 462952 462953
462954 462955 462956 462957 462958 462959 462960 462961
462962 462963 462964 462965 462966 462967 462968 462969
462970 462971 462972 462973 462974 462975 462976 462977
462978 462979 462980 462981 462982 462983 462984 462985
462986 462987

Indirect Blocks:
462861
[root@Akula1 workbench]#
```

Como se observa la salida de *istat* nos proporciona bastante información de gran utilidad, como son los tiempos MAC, permisos del archivo y lo más importante desde el punto de vista de recuperación de datos, el tamaño del archivo. También se ha recuperado una lista de los apuntadores directos e indirectos que nos guían a los bloques que contienen la información original del archivo.

Es importante resaltar que a menos que los bloques de datos hayan sido reescritos por el SO, el proceso de recuperación es bastante sencillo, ya que se puede utilizar la herramienta *icat* del TSK y solo resta verificar que el archivo es recuperado de forma exitosa abriéndolo.

```
[root@Akula1 workbench]# icat -r sda5img.dd 224002 > recovered
[root@Akula1 workbench]# file recovered
recovered: PDF document, version 1.3
```

Para comprender mejor la diferencia entre ambos mecanismos se utiliza el mismo archivo "reference.pdf" el cual ahora es copiado a una partición con formato Ext3 y al cual se le toma una imagen forense en formato crudo. El tamaño del archivo es el mismo pero los inodos y bloques son diferentes por tratarse una partición distinta, el archivo presenta la siguiente estructura:

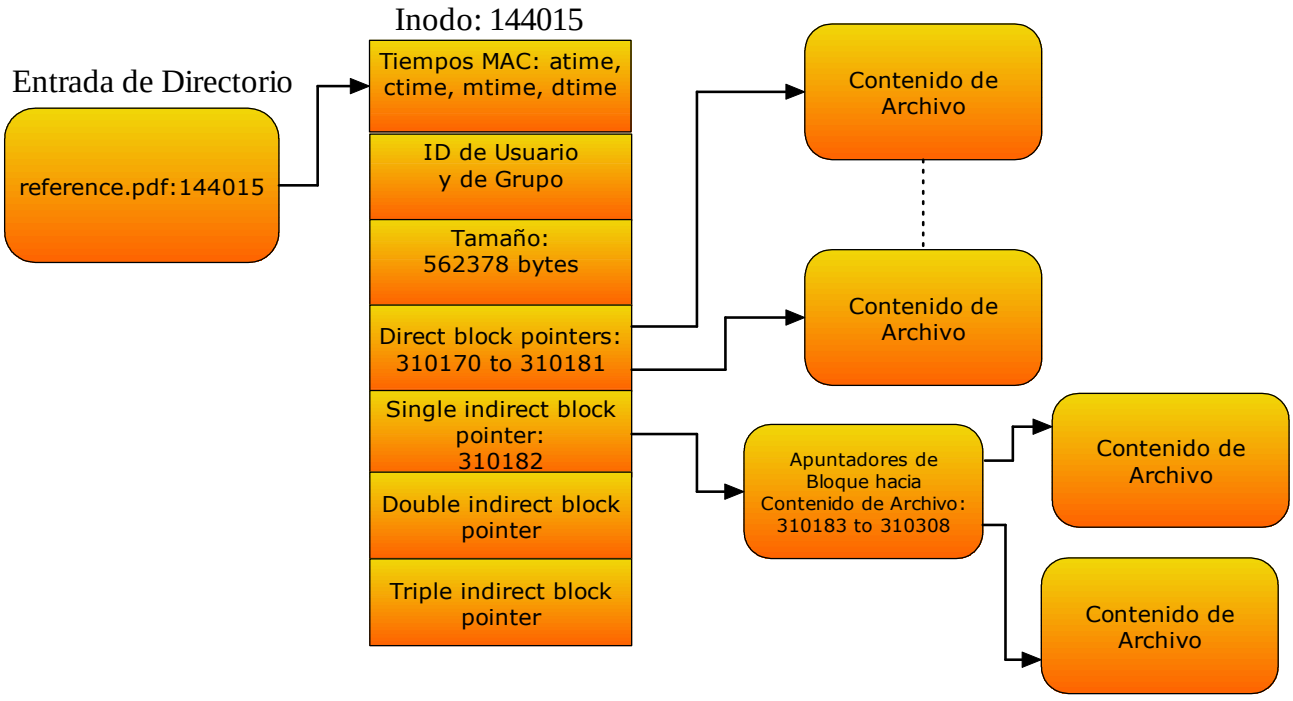

Fig 3.6: Estructura de un archivo antes de ser borrado en Ext3

En Ext3 el SO lleva a cabo medidas adicionales cuando se borra un archivo. Dentro del inodo correspondiente el tamaño del archivo así como los apuntadores de bloque directos e indirectos son sobrescritos con ceros, destruyendo los enlaces que nos permiten rastrear y recuperar los bloques de datos de un archivo como se muestra en el siguiente diagrama:

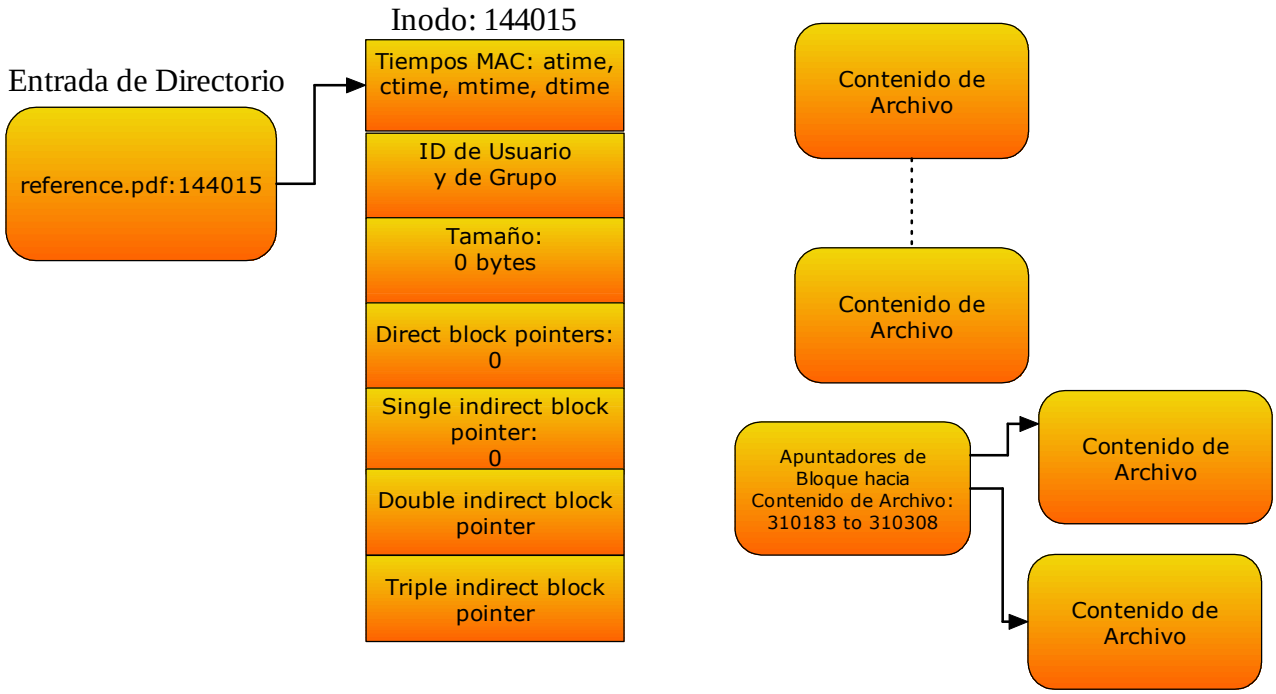

Fig 3.7: Estructura de un archivo después de ser borrado en Ext3

Ahora se puede ver lo que sucede en un sistema de archivos Ext3. En la partición de prueba se borran varios archivos entre ellos “reference.pdf” y se llevan a cabo los mismos pasos para recuperar el archivo como se hizo en el sistema de archivos Ext2:

```
[root@Akula1 workbench]# ils -r sda1img.dd
class|host|device|start_time
ils|Akula1||1194252024
st_ino|st_alloc|st_uid|st_gid|st_mtime|st_atime|st_ctime|st_mode|st_nlink|st_size|st_block0
|st_block1
144014|f|0|0|1194231994|1194231994|1194231994|40755|0|0|0|0
144015|f|0|0|1194231994|1194231982|1194231994|100644|0|0|0|0
[REMOVED]
[root@Akula1 workbench]# istat sda1img.dd 144015
inode: 144015
Not Allocated
Group: 9
Generation Id: 3670456940
uid / gid: 0 / 0
mode: -rw-r--r--
size: 0
num of links: 0

Extended Attributes  (Block: 295595)
security.selinux=root:object_r:file_t:s0

Inode Times:
Accessed:       Sun Nov  4 21:06:22 2007
File Modified:  Sun Nov  4 21:06:34 2007
Inode Modified: Sun Nov  4 21:06:34 2007
Deleted:        Sun Nov  4 21:06:34 2007

Direct Blocks:
[root@Akula1 workbench]#
```

Como se observa la salida del comando *ils* confirma la teoría sobre el mecanismo de borrado de Ext3, los enlaces hacia los bloques de datos (apuntadores) han sido remplazados con ceros. Es por esto que hubo un tiempo en el que se consideró como imposible la recuperación de archivos borrados en Ext3.

Capítulo 4. Resultados

4.1 Recuperación utilizando la bitácora de Ext3

El primer paso en esta técnica de recuperación es tener localizado el inodo del archivo que se desea recuperar. Existen varias formas de llegar al inodo, por ejemplo utilizando la herramienta *debugfs,* que es parte de las herramientas estándar de Linux o utilizar *fls* o *ils* de TSK. Por simplicidad se asume que ya se tiene el inodo del archivo en cuestión y se verifica la información con *ils*:

```
[root@Akula1 workbench]# ils -r sda6img.dd
class|host|device|start_time
ils|Akula1||1194252024
st_ino|st_alloc|st_uid|st_gid|st_mtime|st_atime|st_ctime|st_mode|st_nlink|st_size|st_block0
|st_block1
144014|f|0|0|1194231994|1194231994|1194231994|40755|0|0|0|0
144015|f|0|0|1194231994|1194231982|1194231994|100644|0|0|0|0
144016|f|0|0|1194231994|1182970801|1194231994|100644|0|0|0|0
144017|f|0|0|1194231994|1182970801|1194231994|100644|0|0|0|0
144018|f|0|0|1194231994|1182970801|1194231994|100644|0|0|0|0
144019|f|0|0|1194231994|1182970801|1194231994|100644|0|0|0|0
144020|f|0|0|1194231994|1194231982|1194231994|100644|0|0|0|0
```

Este es el contenido los dos primeros inodos:

```
[root@Akula1 workbench]# istat sda6img.dd 144014
inode: 144014
Not Allocated
Group: 9
Generation Id: 3670456939
uid / gid: 0 / 0
mode: drwxr-xr-x
size: 0
num of links: 0

Extended Attributes  (Block: 295595)
security.selinux=root:object_r:file_t:s0

Inode Times:
Accessed:       Sun Nov  4 21:06:34 2007
File Modified:  Sun Nov  4 21:06:34 2007
Inode Modified: Sun Nov  4 21:06:34 2007
Deleted:        Sun Nov  4 21:06:34 2007

Direct Blocks:
[root@Akula1 workbench]# istat sda6img.dd 144015
inode: 144015
Not Allocated
Group: 9
Generation Id: 3670456940
uid / gid: 0 / 0
mode: -rw-r--r--
size: 0
num of links: 0

Extended Attributes  (Block: 295595)
security.selinux=root:object_r:file_t:s0

Inode Times:
Accessed:       Sun Nov  4 21:06:22 2007
File Modified:  Sun Nov  4 21:06:34 2007
Inode Modified: Sun Nov  4 21:06:34 2007
Deleted:        Sun Nov  4 21:06:34 2007
```

```
Direct Blocks:
[root@Akula1 workbench]#
```

Lo que se puede determinar de la salida anterior es que el inodo 144014 estaba ligado a un directorio y que el inodo 144015 se encontraba signado a un archivo pero sus estadísticas y apuntadores han sido borrados, también sabemos que el inodo 144015 pertenece al grupo de bloques 9 y está marcado como *"no asignado"*. A continuación se extraen los datos del grupo 9:

```
[root@Akula1 workbench]# fsstat sda1img.dd | grep -i "group: 9"
Group: 9:
  Inode Range: 144001 - 160000
  Block Range: 294912 - 327679
  Layout:
    Super Block: 294912 - 294912
    Group Descriptor Table: 294913 - 294913
    Data bitmap: 295093 - 295093
    Inode bitmap: 295094 - 295094
    Inode Table: 295095 - 295594
    Data Blocks: 295595 - 327679
```

Fsstat nos da información importante sobre el grupo de bloques 9, primero que el grupo está compuesto de 16000 inodos (Inode range: 144001-160000), y que la tabla de inodos tiene un tamaño de 500 bloques (Inode Table: 295095-295594).

Cada bloque de la tabla de inodos tiene 32 inodos (16000 dividido entre 500), por lo tanto el inodo 144015 el el 15º inodo en la tabla y su contenido se encuentra almacenado en el primer bloque de la tabla de inodos.

Cabe recordar que la bitácora opera a nivel de bloque, en este caso el bloque que se busca es el 295095 (el primer bloque de la tabla de inodos). Al revisar la salida de *jls* se observa que hay varias referencias al bloque 295095, se utiliza la primera referencia que se encontró, el cual pertenece a la entada 7 de la bitácora. En caso de existir múltiples referencias a un bloque de interés, es probable que se analicen cada una de ellas.

Para determinar el orden cronológico de las referencias, basta con ver el número de secuencia en la bitácora para el bloque de interés, mientras menor sea el número de transacción más antigua será la copia del inodo deseado que reside en el bloque.

```
[root@Akula1 workbench]# jls  sda1img.dd
JBlk    Descriptrion
0:      Superblock (seq: 0)
1:      Allocated Descriptor Block (seq: 2)
2:      Allocated FS Block 183
3:      Allocated Commit Block (seq: 2)
4:      Allocated Descriptor Block (seq: 3)
5:      Allocated FS Block 295094
6:      Allocated FS Block 1
7:      Allocated FS Block 295095
8:      Allocated FS Block 295093
9:      Allocated FS Block 295595
[REMOVED]
```

La salida del comando nos muestra que el bloque 7 (transacción) de la bitácora contiene información sobre una operación sobre la tabla de inodos del grupo 9, ya que la bitácora registra al menos varias copias de los metadatos que han sido modificados, se puede buscar con seguridad una copia del inodo *144015* dentro de la bitácora. Es posible que se tenga que revisar todas las referencias del bloque de interés pero para propósitos prácticos de demostración solo utiliza la primera referencia (más antigua).

Como se ha determinado el inodo de interés es el 15º dentro del grupo de bloques 9, por lo que para poder extraer la copia del inodo 144015 de la bitácora se utilizan una combinación de herramientas; *jcat* para leer el contenido de la bitácora, *dd* para "cortar" el inodo 144015 y *xdd* para visualizar el contenido del inodo en formato hexadecimal, pero antes se necesita conocer el tamaño de los inodos, generalmente el tamaño de un inodo es de 128 bytes en un sistema de archivos Ext2/3, pero para verificar esta información se utiliza el comando del SO Linux *dumpe2fs* en el archivo que contiene la imagen forense que se está analizando (sda6img.dd), como se muestra a continuación:

```
[root@Akula1 workbench]# dumpe2fs sda6img.dd|grep -i "inode size"
dumpe2fs 1.38 (30-Jun-2005)
Inode size:               128
[root@Akula1 workbench]#
```

La herramienta dumpe2fs también se puede utilizar para obtener información de un sistema de archivo. A continuación se ejecuta el siguiente comando:

```
jcat sda6img.dd 8 7 | dd bs=128 skip=14 count=1 | xxd
```

La herramienta *jcat* toma tres parámetros, el primero es el nombre del archivo que contiene la imagen forense que está analizando (sda6img.dd), el segundo es el inodo donde la bitácora comienza y el tercero es la entrada de la bitácora que se desea ver (en este caso la entrada es la 7), la salida es enviada como entrada a la herramienta *dd* que permite extraer solo la información del inodo 144015 que está contenida en el bloque 7. Como se menciono con anterioridad solo nos interesa el 15º inodo del grupo de inodos por lo que *dd* tiene que brincarse los primeros 14 bloques, lo que se define con *skip=14*; el tamaño de los inodos es de 128 bytes por lo que se define *bs=128* para que los bloques que maneje *dd* sean iguales a los inodos. El último comando es la herramienta *xxd* el cual toma como entrada el inodo extraído por *dd* y lo manda a la salida estándar (pantalla) en un formato hexadecimal. Este es el resultado:

```
[root@Akula1 workbench]# jcat sda6img.dd 8 7 | dd bs=128 skip=14 count=1 | xxd
1+0 records in
1+0 records out
128 bytes (128 B) copied, 0.00402034 seconds, 31.8 kB/s
0000000: a481 0000 ca94 0800 b1b3 8246 bc13 2e47  ...........F1..G
0000010: dd96 8246 0000 0000 0000 0100 6004 0000  ...F........`...
0000020: 0000 0000 0000 0000 9abb 0400 9bbb 0400  ................
```

```
0000030: 9cbb 0400 9dbb 0400 9ebb 0400 9fbb 0400  ................
0000040: a0bb 0400 a1bb 0400 a2bb 0400 a3bb 0400  ................
0000050: a4bb 0400 a5bb 0400 a6bb 0400 0000 0000  ................
0000060: 0000 0000 6cba c6da ab82 0400 0000 0000  ....l...........
0000070: 0000 0000 0000 0000 0000 0000 0000 0000  ................
[root@Akula1 workbench]#
```

Lo que se ha obtenido es la copia del inodo 144015 antes de ser modificado a su estado actual, pero es necesario interpretarlo. La estructura de un inodo es muy extensa y fuera del alcance práctico de recuperación, pero existen varias fuentes de información donde se pueden obtener la descripción detallada de la estructura de un inodo (Carrier 2005 [437-441]).

El principal objetivo de la técnica es el localizar los apuntadores de bloques. Es importante hacer notar que los metadatos almacenados por el SO se encuentran en formato little endian (en este caso por ser una plataforma x86) y es así que se debe de interpretar del formato hexadecimal que se obtuvo.

Posición en Bytes	Description
4 a 7	Primeros 32 bits del tamaño del archivo en bytes
40 a 87	Lista de apuntadores de bloque directos (12)
88 a 91	Apuntador de bloque indirecto sencillo
92 a 95	Apuntador de bloque indirecto doble
95 a 99	Apuntador de bloque indirecto triple

Tabla 4.1: Estructura parcial de un inodo

Aplicando la información de la tabla se obtiene lo siguiente:

El tamaño del archivo que se encuentra ligado al inodo 144015 antes de ser borrado es de 562378 bytes (0x0894ca):

```
0000000: a481 0000 ca94 0800 b1b3 8246 bc13 2e47  ...........F1..G
```

Existen 12 apuntadores directos de bloque: 310170 al 310181 (0x4bb9a al 0x4bba5):

```
0000020: 0000 0000 0000 0000 9abb 0400 9bbb 0400  ................
0000030: 9cbb 0400 9dbb 0400 9ebb 0400 9fbb 0400  ................
0000040: a0bb 0400 a1bb 0400 a2bb 0400 a3bb 0400  ................
0000050: a4bb 0400 a5bb 0400 a6bb 0400 0000 0000  ................
```

Existe un apuntador indirecto sencillo de bloque: 310182 (0x4bba6):

```
0000050: a4bb 0400 a5bb 0400 a6bb 0400 0000 0000  ................
```

No hay apuntadores indirectos dobles o triples:

```
0000050: a4bb 0400 a5bb 0400 a6bb 0400 0000 0000  ................
0000060: 0000 0000 6cba c6da ab82 0400 0000 0000  ....l...........
```

Pero no hemos terminado la recuperación, Solo se tienen los primeros 12 bloques de datos, lo que representa 49152 bytes, recordar que el tamaño del archivo es de 562378 bytes (esto se obtiene de la información extraída del inodo). Ahora es necesario seguir el apuntador sencillo indirecto el cual apunta al bloque 310182 (0x4bba6) el cual puede contener apuntadores a los bloques de datos restantes.

Utilizando *dcat* se obtiene el contenido del bloque 31018:

```
[root@Akula1 workbench]# dcat -h sda6img.dd 310182
0       a7bb0400 a8bb0400 a9bb0400 aabb0400     .... .... .... ....
16      abbb0400 acbb0400 adbb0400 aebb0400     .... .... .... ....
32      afbb0400 b0bb0400 b1bb0400 b2bb0400     .... .... .... ....
48      b3bb0400 b4bb0400 b5bb0400 b6bb0400     .... .... .... ....
64      b7bb0400 b8bb0400 b9bb0400 babb0400     .... .... .... ....
80      bbbb0400 bcbb0400 bdbb0400 bebb0400     .... .... .... ....
96      bfbb0400 c0bb0400 c1bb0400 c2bb0400     .... .... .... ....
112     c3bb0400 c4bb0400 c5bb0400 c6bb0400     .... .... .... ....
128     c7bb0400 c8bb0400 c9bb0400 cabb0400     .... .... .... ....
144     cbbb0400 ccbb0400 cdbb0400 cebb0400     .... .... .... ....
160     cfbb0400 d0bb0400 d1bb0400 d2bb0400     .... .... .... ....
176     d3bb0400 d4bb0400 d5bb0400 d6bb0400     .... .... .... ....
192     d7bb0400 d8bb0400 d9bb0400 dabb0400     .... .... .... ....
208     dbbb0400 dcbb0400 ddbb0400 debb0400     .... .... .... ....
224     dfbb0400 e0bb0400 e1bb0400 e2bb0400     .... .... .... ....
240     e3bb0400 e4bb0400 e5bb0400 e6bb0400     .... .... .... ....
256     e7bb0400 e8bb0400 e9bb0400 eabb0400     .... .... .... ....
272     ebbb0400 ecbb0400 edbb0400 eebb0400     .... .... .... ....
288     efbb0400 f0bb0400 f1bb0400 f2bb0400     .... .... .... ....
304     f3bb0400 f4bb0400 f5bb0400 f6bb0400     .... .... .... ....
320     f7bb0400 f8bb0400 f9bb0400 fabb0400     .... .... .... ....
336     fbbb0400 fcbb0400 fdbb0400 febb0400     .... .... .... ....
352     ffbb0400 00bc0400 01bc0400 02bc0400     .... .... .... ....
368     03bc0400 04bc0400 05bc0400 06bc0400     .... .... .... ....
384     07bc0400 08bc0400 09bc0400 0abc0400     .... .... .... ....
400     0bbc0400 0cbc0400 0dbc0400 0ebc0400     .... .... .... ....
416     0fbc0400 10bc0400 11bc0400 12bc0400     .... .... .... ....
432     13bc0400 14bc0400 15bc0400 16bc0400     .... .... .... ....
448     17bc0400 18bc0400 19bc0400 1abc0400     .... .... .... ....
464     1bbc0400 1cbc0400 1dbc0400 1ebc0400     .... .... .... ....
480     1fbc0400 20bc0400 21bc0400 22bc0400     ....  ... !... "...
496     23bc0400 24bc0400 00000000 00000000     #... $... .... ....
512     00000000 00000000 00000000 00000000     .... .... .... ....
528     00000000 00000000 00000000 00000000     .... .... .... ....
544     00000000 00000000 00000000 00000000     .... .... .... ....
[REM0VED]
```

El contenido es una lista de los bloques de datos faltantes, se asume en un principio que son parte del archivo “reference.pdf”, pero hay que tomar en cuenta que pueden haber sido sobrescritos por el SO o por el individuo que los borro. El rango de bloques es de 310183 al 310308 (0x4bbaf to 0x4bc24).

Para finalizar se recupera el archivo “cortando” la información con la ayuda de la herramienta *dd* que nos indica las direcciones que se han extraído de la bitácora y la recuperación del archivo se puede llevar acabo utilizando la herramienta *foremost* o de forma manual.

En caso de que el archivo se encuentre fragmentado, una opción es el extraer los bloques de forma manual con dd y unirlos en un solo archivo para finalmente utilizar foremost y obtener el archivo borrado:

```
[root@Akula1 workbench]# dd bs=4096 skip=310168 count=141 if=sda6img.dd of=recover.dd
141+0 records in
141+0 records out
577536 bytes (578 kB) copied, 0.00310458 seconds, 186 MB/s

[root@Akula1 workbench]# foremost -b 4096 -o recovery -t pdf  recover.dd
Processing: recover.dd
|*|
[root@Akula1 workbench]# cd recovery
[root@Akula1 recovery]# more *.txt
Foremost version 1.5.1 by Jesse Kornblum, Kris Kendall, and Nick Mikus
Audit File

Foremost started at Mon Nov  5 19:59:13 2007
Invocation: foremost -b 4096 -o recovery -t pdf recover.dd
Output directory: /media/workbench/ext3default/recovery
Configuration file: /usr/local/etc/foremost.conf
------------------------------------------------------------------
File: recover.dd
Start: Mon Nov  5 19:59:13 2007
Length: 564 KB (577536 bytes)

Num      Name (bs=4096)        Size      File Offset     Comment

0:      00000002.pdf         553 KB            8192
Finish: Mon Nov  5 19:59:13 2007

1 FILES EXTRACTED

pdf:= 1
------------------------------------------------------------------

Foremost finished at Mon Nov  5 19:59:13 2007
```

En este caso se extrajeron algunos bloques de más antes y después de donde se localiza el contenido del archivo, *foremost* se encarga de determinar el tipo y formato del archivo en forma automática, el siguiente diagrama de flujo muestra los pasos a seguir para aplicación de esta técnica:

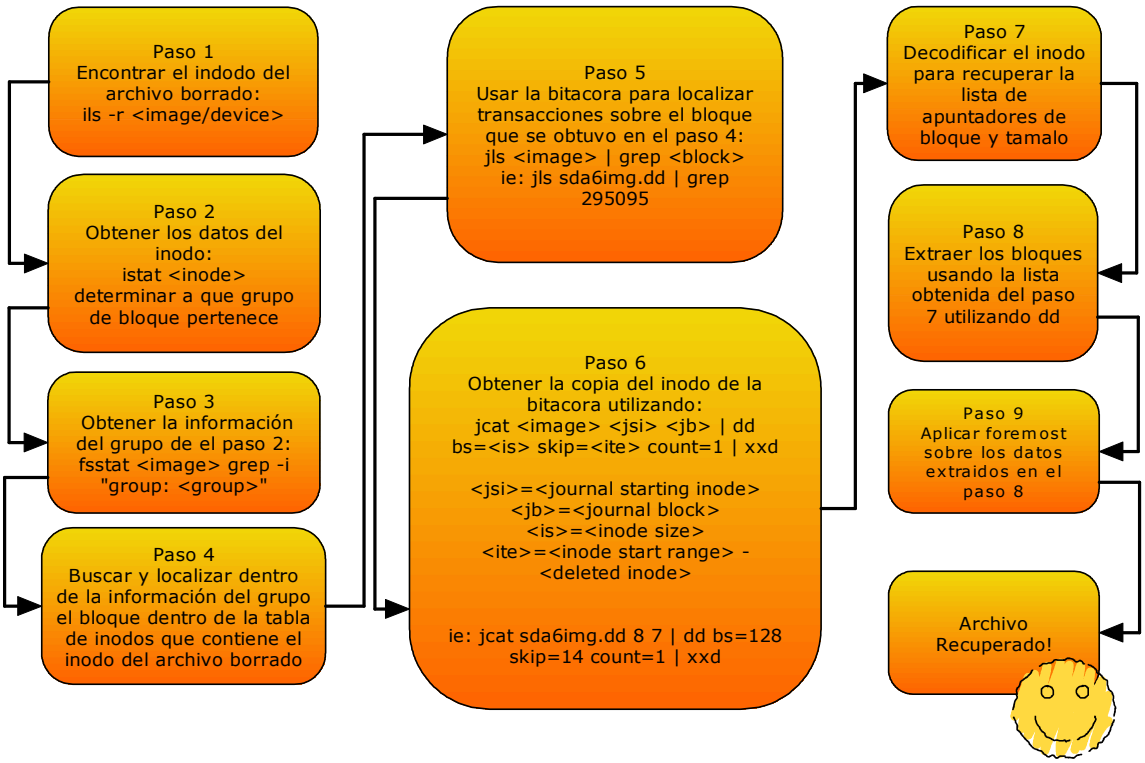

Fig 4.1: Recuperación de un archivo borrado utilizando la bitácora de Ext3

4.2 Línea temporal de un archivo usando la bitácora de Ext3.

Como se ha explicado en un principio la bitácora de Ext3 ofrece múltiples ventajas desde el punto de vista forense, entre estas la recuperación de los tiempos MAC. Los cuales se da una breve introducción:

- **Mtime (Tiempo de Modificación):** Este valor se actualiza cuando el contenido de un archivo/directorio es cambiado (ejemplo: agregar, borrar o cambiar el contenido de un archivo de texto)
- **Atime (Tiempo de Acceso):** Este valor se actualiza cada vez el contenido de un archivo o directorio es leído, copiado o movido a un nuevo volumen. (Por ejemplo abrir un archivo en un programa para su edición o lectura)
- **Ctime (Tiempo de Cambio):** Este valor representa la última fecha de modificación de los metadatos de un archivo (inodo). Los eventos que pueden hacer que se lleve a cabo esta actualización son la creación de un archivo o cuando los permisos o el propietario de un archivo son cambiados
- **Dtime: (Tiempo de Borrado):** Este valor se establece solo cuando el archivo es borrado y se pone a ceros cuando el inodo es “asignado”

Al analizar los valores temporales MAC es posible reconstruir la actividad del archivo en cuestión en un sistema de archivos. Por ejemplo cuando un archivo es creado los valores

mtime, atime y ctime son actualizados con la fecha y hora en el momento de la creación del archivo y el valor dtime es establecido a cero, al mismo tiempo los valores mtime y ctime del directorio que contiene el archivo son actualizados. Otro ejemplo es cuando un archivo es borrado, los valores mtime, atime y ctime son los mismos que el valor de dtime, el cual la fecha y hora del momento de la operación de borrado.

En Ext2 estos valores representan una buena pieza de evidencia que puede permitir al analista forense generar una línea temporal de los eventos que ocurrieron durante un incidente, desafortunadamente en Ext2 los tiempos MAC solo muestran la actividad más reciente (la última) cualquier indicio previo se pierde, pero con la bitácora de Ext3 es posible obtener un historial de actividad más completo.

Como se ha discutido en el capitulo 3 la bitácora almacena en el peor de los casos los cambios a los metadatos, los cuales se encuentran almacenados en los inodos del sistema de archivos. Entre los metadatos de interés se encuentran los tiempos MAC de un archivo, por lo que la oportunidad se presenta de recuperar tantas copias de un mismo inodo como sea posible el cual nos permite generar una línea temporal más detallada, no solo de su última actividad (Farmer & Venema. 2007 [25-26]).

Le herramienta *debugfs* que es estándar en la mayoría de los SO Linux es de gran ayuda, por lo general esta herramienta se utiliza para recuperación de datos de un sistema de archivos corrupto. Para ver su aplicación en la generación de una línea temporal utilizara la imagen forense sda6img.dd la cual contiene el archivo borrado “reference.pdf. Para mantener la integridad de la evidencia se recomienda ejecutar la herramienta *debugfs* con la opción *–c* lo cual hace que el archivo que contiene la imagen sea utilizado en modo de solo lectura.

Si se ejecuta los comandos *debugfs –c –R 'logdump -i <144015>’ sda6img.dd | grep atime* la herramienta debugfs nos muestra los metadatos relacionados con el archivo “reference.pdf” que está asignado al inodo 144015, como se mostro en los ejemplos de recuperación, su salida su vez se pasa al comando grep el cual solo nos muestra los valores correspondientes a atime, de esta forma se puede obtener los demás valores de ctime,mtime y dtime.

```
[root@Akula1 workbench]# debugfs -c -R 'logdump -i <144015>' sda5img.dd|grep atime
debugfs 1.38 (30-Jun-2005)
sda5img.dd: catastrophic mode - not reading inode or group bitmaps
    atime: 0x4682b3b1 -- Wed Jun 27 14:00:01 2007
    atime: 0x4682b3b1 -- Wed Jun 27 14:00:01 2007
    atime: 0x472e1ad0 -- Sun Nov  4 13:17:36 2007
    atime: 0x472e1ad0 -- Sun Nov  4 13:17:36 2007
    atime: 0x472e1ad0 -- Sun Nov  4 13:17:36 2007
    atime: 0x472e1ad0 -- Sun Nov  4 13:17:36 2007
    atime: 0x472e88ae -- Sun Nov  4 21:06:22 2007
    atime: 0x472e88ae -- Sun Nov  4 21:06:22 2007
[root@Akula1 workbench]# debugfs -c -R 'logdump -i <144015>' sda5img.dd|grep mtime
debugfs 1.38 (30-Jun-2005)
sda5img.dd: catastrophic mode - not reading inode or group bitmaps
    mtime: 0x46829dd -- Wed Jun 27 11:57:01 2007
    mtime: 0x46829dd -- Wed Jun 27 11:57:01 2007
    mtime: 0x46829dd -- Wed Jun 27 11:57:01 2007
    mtime: 0x46829dd -- Wed Jun 27 11:57:01 2007
    mtime: 0x46829dd -- Wed Jun 27 11:57:01 2007
    mtime: 0x46829dd -- Wed Jun 27 11:57:01 2007
    mtime: 0x46829dd -- Wed Jun 27 11:57:01 2007
    mtime: 0x472e88ba -- Sun Nov  4 21:06:34 2007
[root@Akula1 workbench]# debugfs -c -R 'logdump -i <144015>' sda5img.dd|grep ctime
debugfs 1.38 (30-Jun-2005)
sda5img.dd: catastrophic mode - not reading inode or group bitmaps
    ctime: 0x472e136c -- Sun Nov  4 12:46:04 2007
    ctime: 0x472e136c -- Sun Nov  4 12:46:04 2007
    ctime: 0x472e136c -- Sun Nov  4 12:46:04 2007
    ctime: 0x472e136c -- Sun Nov  4 12:46:04 2007
    ctime: 0x472e136c -- Sun Nov  4 12:46:04 2007
    ctime: 0x472e136c -- Sun Nov  4 12:46:04 2007
    ctime: 0x472e136c -- Sun Nov  4 12:46:04 2007
    ctime: 0x472e88ba -- Sun Nov  4 21:06:34 2007
[root@Akula1 workbench]# debugfs -c -R 'logdump -i <144015>' sda5img.dd|grep dtime
debugfs 1.38 (30-Jun-2005)
sda5img.dd: catastrophic mode - not reading inode or group bitmaps
    dtime: 0x472e88ba -- Sun Nov  4 21:06:34 2007
```

Ahora se observa la actividad del archivo “reference.pdf”, y se puede determinar que el archivo fue accedido en múltiples ocasiones durante la tarde de Noviembre 4 (atime) y finalmente fue borrado el mismo día a las 21:06 (dtime). También se observa que el valor dtime solo aparece una sola vez y los otros valores MAC tienen el mismo valor que dtime. Toda esta información se recupero de la bitácora. En caso de que el sistema de archivos fuera Ext2 sólo se puede obtener el momento en que fue borrado.

Existen ciertas limitaciones de la herramienta *debugfs* que deben tener presentes:

- La versión estándar de *debugfs* solo permite la revisión de un archivo a la vez
- En algunas ocasiones *debugfs* no reconoce el final de la bitácora y comienza arrojar basura.
- *Debugfs* no reconoce una bitácora que se localiza en un dispositivo externo al sistema de archivos correspondiente.

Existe un parche para el primer problema con debugfs (Farmer & Venema. 2004 [34]) que permite el desplegar todos los tiempos MAC de múltiples archivos, para el tercer problema es posible crear un programa o script para la decodificación de la bitácora como se muestra en la investigación *Taking Advatage of Ext3 journaling File System in a forensic investigation*. (Narváez 2007 [19-30]).

Conclusiones

El desarrollo de nuevas técnicas haciendo uso de herramientas que ya existen es necesario en el cómputo forense, estas nuevas técnicas permiten obtener evidencia en situaciones en las que en un principio se pensaba no era posible obtener información. También es importante señalar que en ocasiones las herramientas con las que cuenta un analista forense tienen sus límites por lo que es necesario desarrollar nuevas o incluso obtener la información de forma manual. Esto solo es posible a través de un entendimiento detallado y minucioso del funcionamiento del componente que se está analizando, como en este caso lo fue la bitácora de Ext3.

También debido al rápido avance de la tecnología, las herramientas necesitan mantenerse actualizadas o incluso crear nuevas. Algunas tecnologías como los discos duros de estado sólido o los teléfonos inteligentes han introducidos nuevos retos y por lo general se encuentran las primeras soluciones a estos problemas en herramientas de código abierto.

Por todo lo anterior es necesario que un analista forense busque comprender la tecnología que hay detrás del dispositivo que se está analizando no solo por la capacidad de extender las herramientas actuales o desarrollar nuevas, si no también, al tener que defender sus hallazgos ante una corte y poder explicar sus hallazgos de forma independiente a la herramienta que se haya utilizado.

Para 2008 se habían liberado herramientas de código abierto que permiten la recuperación de información borrada en Ext3, un ejemplo es la herramienta de código libre *ext3undel* (http://freecode.com/projects/ext3undel) sin embargo el desarrollo de técnicas son de mayor valor para el investigador forense que tiene que conocer la procedencia de los resultados además de recuperar la información, mientras que la recuperación de datos se conforma con solo recuperarla.

El uso de herramientas de código abierto ofrece mayor flexibilidad para poder llevar a cabo nuevas técnicas o conceptos por su arquitectura, sobre las herramientas comerciales, aunque las comerciales ofrecen mayor eficiencia ya que automatizan muchas de las actividades del proceso forense.

Las herramientas de código abierto son accesibles por no implicar un costo monetario excesivo como sucede en las herramientas comerciales que por lo general tiene un costo superior a los $1300 USD, además de que su licencia en la mayoría de los casos requiere de una cuota anual que tiene un costo entre 5 y el 15% del costo de la Licencia del primer año por concepto de mantenimiento.

Por último las herramientas de código abierto ofrecen al analista una segunda opinión que permite comparar los resultados obtenidos con herramientas comerciales y con esto fortalecer o disputar los resultados de un análisis, incluso a determinar si una herramienta no está funcionando como debe con lo que nos permite detectar defectos de software en las herramientas de análisis.

Bibliografía

Capítulo 2

File system forensic analysis. Carrier, B., Pearson Education Inc, 2005, [82-88].

Capítulo 3

File system forensic analysis. Carrier, B., Pearson Education Inc, 2005, [174-175].

http://www.first.org/conference/2007/papers/venema-wietse-slides.pdf, Forensic Discovery, FIRST conference 2007, Marzo 19, 2012

Capítulo 4

File system forensic analysis. Carrier, B., Pearson Education Inc, 2005, [437-441].

http://www.first.org/conference/2007/papers/venema-wietse-slides.pdf, Forensic Discovery, FIRST conference 2007, Marzo 19, 2012 [25-26].

Forensic Discovery, Farmer, D. & Wietse, V., Addison-Wesley, 2004, [34].

http://www.first.org/conference/2007/papers/venema-wietse-slides.pdf, Forensic Discovery, FIRST conference 2007, Marzo 19, 2012 [25-26].

http://computer-forensics.sans.org/community/papers/gcfa/advantage-ext3-journaling-file-system-forensic-investigation_332, *Taking Advatage of Ext3 journaling File System in a forensic investigation*. Gold GCFA Certification. Marzo 20, 2012 [19-30].