DEFINING AND VALIDATING FORENSIC STANDARDS FOR EXT4 FILE SYSTEM

By

Moeed Hussain

A thesis submitted to the faculty of Information Security Department, Military College of Signals, National University of Sciences and Technology, Rawalpindi in partial fulfillment of the requirements for the degree of MS in Information Security

August 2018

ABSTRACT

Linux operating system is one of the most widely used operating systems in the world .The extensive use of this Operating system necessitates the need of performing the forensic analysis of the file system that is most widely used with this operating system .At the file system level when files are created and deleted the data is still persistent on the file system in some scenarios and some part of the data is still recoverable even after secure deletion the data is persistent and some instances of data can still be recovered from the forensics of the file system.NIST has not provided any standards to delete the file permanently for different file systems. Due to which the need arises to define the standards for Anti forensics of EXT4 file system. So that even the smallest amount of data cannot be recovered. In this Context a detailed research has been conducted on the structure of the file system and multiple scenarios have been created to look for data persistence on the file system.

DECLARATION

I hereby declare that no portion of work presented in this thesis has been submitted in support of another award or qualification either at this institution or elsewhere.

DEDICATION

“In the name of Allah, the most Beneficent, the most Merciful"

I dedicate this thesis to my father, mother, bother, and teachers who supported me in every step

ACKNOWLEDGMENTS

All praises to Allah for the strengths and His blessing in completing this thesis.

I would like to convey my gratitude to my supervisor, Asst Prof Mian Muhammad Waseem Iqbal, for his supervision and constant support. His invaluable help of constructive comments and suggestions throughout the experimental and thesis works are major contributions to the success of this research. Also, I would thank my committee members; Asst. Prof Waleed Bin Shahid and Lecturer Narmeen Shafqat for their support and knowledge regarding this topic.

Last, but not the least, I am highly thankful to my Father and Mother (Mr and Mrs Abid Hussain). They have always stood by my dreams and aspirations and have been a great source of inspiration for me. I would like to thank them for all their care, love and support through my times of stress and excitement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter1

Introduction

1.1 Overview

Use of technology in this world is increasing day by day, technology is everywhere e.g. starting from our personal life in terms of smartphones to workstations, laptops, servers used in the office. We can say that technology is in use in almost all the aspects of our life. The technology that is mostly used worldwide is for the purpose of providing information. Due to extensive use of technology has emerged a field by the name of hacking. Hacking is the process in which an unauthorized person affects the Confidentiality Integrity and availability of the information in different forms (in transit and at rest).

Due to the increase in cases of cyber-attacks, needs have risen to investigate the incidents. From here the field of Digital forensics has risen [1] .Almost every technology extensively used in this world involves operating systems. There are different operating systems used worldwide for different purposes. Following are some of the mainly used operating systems throughout the world

- Linux
- Windows
- Mac OS
- Unix

Market share of the above operating systems used on servers is shown as follows

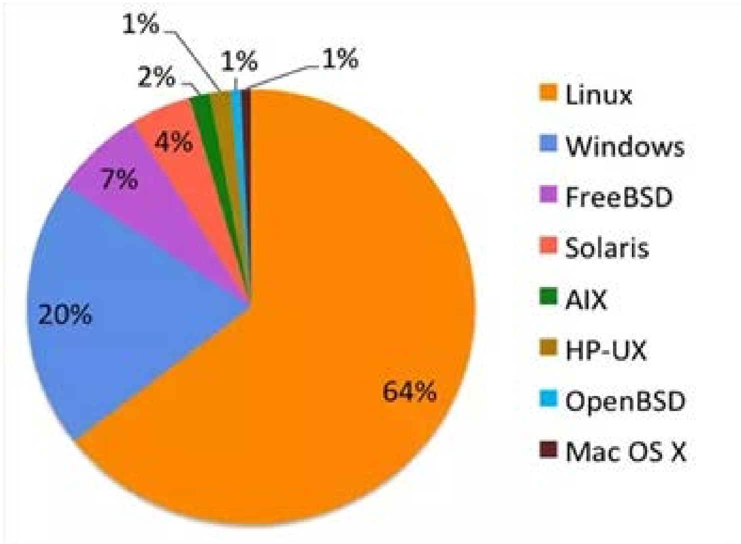


Figure 1: OS Market Share

With the extensive use of Linux operating system throughout the world and more and more cyber-attacks on the information repositories managed by Linux operating system calls for the need of research in the field of digital forensics for Linux operating system .Linux is branched out from Unix Operating system and is open source. Linux has a large market share when it comes to its usage on servers. This operating system is used with different type of devices like servers, Laptops, PC's, netbooks, tablet and mobile devices and many other devices as well. Linux operating system has many flavors like.

- Ubuntu
- Debian
- Fedora
- Open Suse
- Suse Linux Enterprise
- Slackware Linux
- Linux Mint
- CentOS
- Arch Linux
- Redhat Enterprise Linux

An operating system uses file systems to store information on disk storage. Many of the file systems used with Linux operating system are shown as follows.

- Btrfs
- Vfat
- Exfat
- F2FS
- ZFS
- XFS
- JFS
- NILFS2
- NTFS
- Reiser 4
- Reiser FS
- UDF
- EXT2
- EXT3
- EXT4

Out of all these file systems EXT4 has importance because it is used as default operating system with many of the Linux distributions discussed previously. Forensics of Linux operating systems can be divided into File system forensics and Operating system Forensics. When a file is deleted in EXT4 file system some part of it can still be recovered, currently there are no standards to securely delete data at the file system level by NIST which is a well-known authority for defining standards, this thesis deals with the research on data persistence on EXT4 file system and defining Anti-forensic standards for EXT4 file system. [2]

1.2 Problem Statement

When a file is deleted in a file system, it is not completely deleted unless all the data structures associated with the file are over written by another file. National institute of standards and technology has introduced standards for media sanitization but currently there are no specific standards Introduced by NIST to completely wipe all the data in a file system associated with a file. As EXT4 file system is one of the most widely used file system with Linux distributions, research needs to be conducted to define standards for completely wiping data associated with a file in case the media is not to be sanitized.

1.3 Objectives

Three Different file types will be created and deleted in EXT4 file system to look for data persistence in 5 different categories (File System, File Name, Meta Data, Content, and Application). After Observing data persistence we will conclude in which categories data needs to be permanently deleted and define standards for secure file deletion.

1.4 Advantages:

- No part of the meaningful data will be recoverable after the file is deleted.
- Forensic tools are not be able to recover data after the file is deleted.
- People who cannot physically destroy the storage media have the confidence that their data is not recoverable
- Files can be deleted specifically in totality without affecting other files on the file system

1.4 Areas of Application

- Home user Linux distributions using EXT 4 as a file system
- Android smartphones using EXT4 file system
- Servers having EXT4 as a file system.

1.5 Division of Thesis

Chapter 2 discusses detailed literature review of the topic with a focus on already conducted research.

Chapter 3 discusses experimental test bed used for this research is discussed along with different scenarios and implementation results is discussed.

Chapter 4 comprises of results of the research along with screen shots.

In chapter 5, Anti forensic standards which are defined as a result of this research is discussed.

Chapter 6 discusses conclusion and future work related to this topic

Chapter 2

LITERATURE REVIEW

2.1 Structure of Disk Drives

In order to understand the working of EXT 4 file system it is important to understand the working of disk drives. Necessary concepts of disk drives related to this thesis are discussed in this chapter. [3]

2.1.1 Heads and Platters:

There are one or more platters in a disk drive which are coated with magnetic material. A platter has two surfaces i.e. Top and Bottom. Data on platters is stored in a certain way. In a disk drive these platters are arranged on top of one another. Data is read and written on these platters using a dedicated head for each platter. Structure of a disk drive using headers and platters is shown in the following figure.

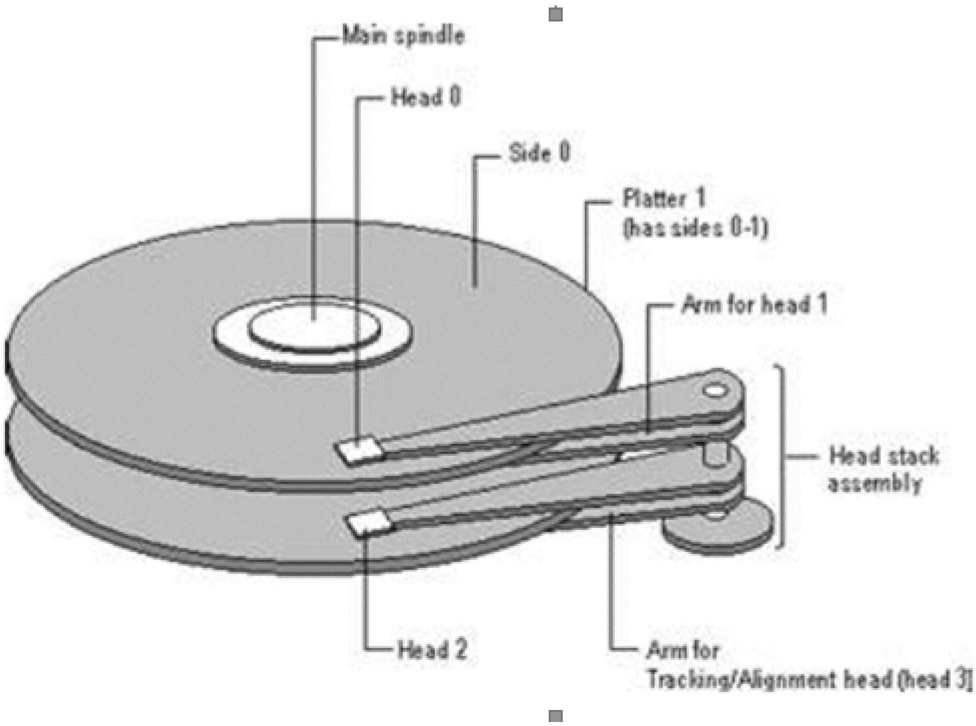


Figure 2: Disk Structure

2.1.2 Tracks and Cylinders:

There are numerous concentric circles on each side of the platter. These circles are called tracks. A column of tracks on these platters is called a cylinder. [3]

2.1.3 Sectors and Blocks:

Tracks on these platters are further divided into parts called sectors. Each sector contains 512 Bytes of data. From this thesis point of view 4 sectors combine to make one block. [3]

2.1.4 Disk Drive Partitioning:

A disk drive can be formatted into different logical partitions or the whole disk drive can be formatted with only one partition depending on the requirement or the features of file systems and disk drives.[3]

2.2 Fourth Extended File System (EXT4):

File systems provide a structure to manage, store, and organize data on a hard disk. EXT4 file system was primarily developed for Linux operating systems. The purpose of launching this file system was to overcome the deficiencies in the predecessors of this file system .Kernel support for EXT4 file system was with the introduction of 2.6.28 [5]

2.2.1 Blocks:

Basic data unit on a disk drive is a sector which is 512 bytes in size. Multiple sectors on a disk drive add up to make a Block. In ext4 block sizes can range from 4KB to 64KB.
By default the size of a block in ext4 is 4KB. [5]

2.2.2 Block Groups:

A block group is a combination of multiple blocks .For a block size of 4KB maximum number of block is a block group can be 32768. File system is divided into multiple block groups as shown in the following figure [6]

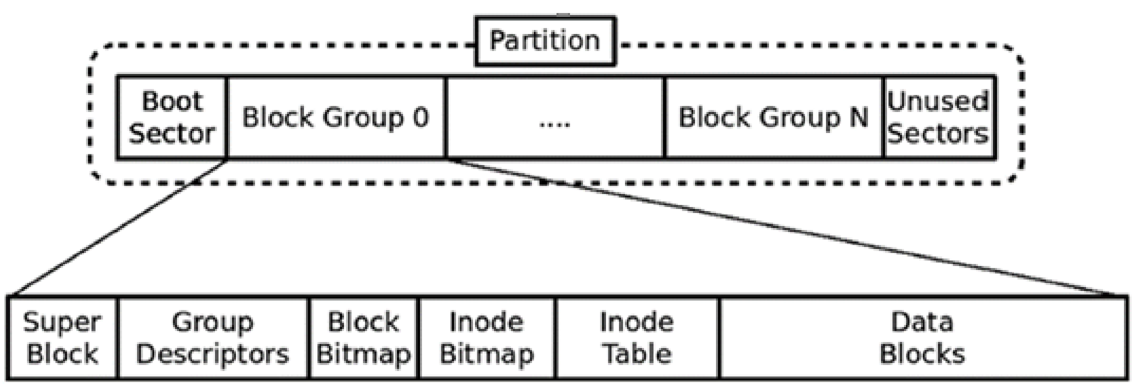


Figure 3 : EXT4 Structure

A brief introduction of the structure is as follows

2.2.2.1 Super Block:

Super Block contains all the administrative data about the file system. The super block exists in the first block of a block group. Every block group in the file system will have the super block until and unless a feature is enabled in the file system which is called sparse super block.[5] If the sparse super block feature is enabled only a limited number of block groups have the super block in them. Information contained in superblock but not limited to includes

- Total Number of Blocks
- Block Size used for the file system
- Number of reserved Blocks before the first Block group
- Total Number of Inodes
- Number of Inodes per Block group
- Volume Name
- Path where file system was last mounted.
- Count of free Inodes and Blocks
- Number of Blocks per Block group
- Last Write time
- Last Mount Time

2.2.2.2 Group Descriptors Table

The block after the super block in a block group is the group descriptor table. Group descriptor block is also present in every block group of the file systems until and unless sparse super feature is enabled in the file system. Whichever block group contains super block also contains group descriptor block as well. Group descriptor data structure of every block group is stored in the group descriptor table [5]. Data structure of group descriptor contains the address of:

- Block Bitmap
- Inode Bitmap

- Inode table

2.2.2.3 Block Bitmap

The block bitmap gives the allocation status of blocks in a group. The starting address of a block bitmap is contained in the group descriptor data structure associated with that block group. One full block is allocated to a block bitmap [5].

2.2.2.4 Inode Bitmap

Allocation status of inodes in a block group is provided by Inode. The starting address of a inode bitmap is contained in the group descriptor data structure associated with that block group. One full block is allocated to an inode bitmap [6].

2.2.2.5 Inode Table:

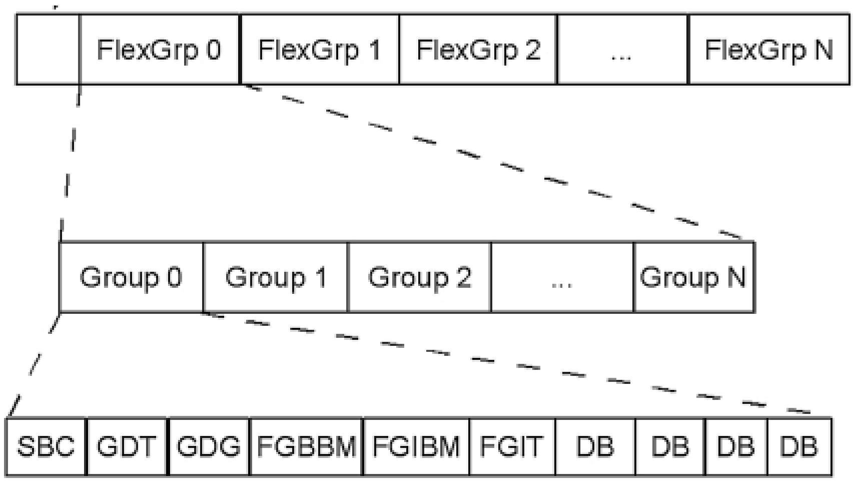
Inode structures associated with a file or directory is stored in the Inode table. Inode contains attributes and disk block locations (Extents) of objects data. Inodes in the inode table contain information such as creation, modification and access times, permissions, size, block addresses where the data is stored [6].

2.2.2.6 Data Blocks:

After all the aforementioned structures exist the data blocks where actual data associated with the file exists.

2.2.3 Flexible Block Groups

Flexible block groups in ext4 is a feature which ties multiple block groups in a file system into one single group. A high level diagram of the layout of file system when flexible block groups is enabled is shown as follows.



Key

Abbreviation	**Meaning**
SBC	Super Block Copy
GDT	Group Descriptor Table
GDG	Group Descriptor Growth Blocks
FGBBM	Flex Group Block Bitmap
FGIBM	Flex Group Inode Bitmap
FGIT	Flex Group Inode Table
DB	Datablock

Figure 4: Flexible Block Group Description

The flexible block group feature in ext4 expands the data bitmap, inode bitmap and the inode table in the first block group of the flexible block group. This allows the file system to account for all the other block groups in the flexible block group and their Meta data can be stored only in first block group. The restriction to keep the administrative data of a block group to that specific block group is removed due to this feature avoiding fragmentation and faster file system reads and writes.

Chapter 3

EXPERIMENTAL TEST BED AND SCENARIOS

3.1 File System Creation

EXT4 file system is created on a 4GB USB disk using mkfs.ext4 command when mounted on Debian GNU/Linux Kali Linux 1.0. The kernel version of this operating system is 3.7-trunk-686-pae. Defaults have been used for the creation of file system on the USB disk.

Experiments were performed using the Sleuthkit (4.5.0), dd command and XXD in Kali Linux.

Some of the default features for ext4 file system creation are that block size is 4096 bytes and journaling is enabled in the file system.

3.2 Scenarios

To define the Anti-forensic standards for ext4 file system we perform file creation and deletion of three different types on the file systems and look at the changes at different categories mentioned by Brian carrier for forensic analysis.

3.2.1 File system Category

The file system category of data is where the general data about a file system are located. The file system category contains the general data that identify how this file system is unique and where other important data are located. In many cases, most of these data are located in a standard data structure in the first sectors of the file system, similar to having a map in the lobby of a building. With this information, the locations of other data, which may vary depending on the size of the file system, can be found.

Analysis of data in the file system category is required for all types of file system analysis because it is during this phase that you will find the location of the data structures in the other categories. Therefore, if any of this data becomes corrupt or is lost, additional analysis is more difficult because you have to find backup copies or guess what the values were. In addition to general layout information, analysis of this category might also show the version of file system, the application that created the file system, the creation date, and the file system label. There is little data in this category that a typical user would be able to set or view without the help of a hex editor. In many cases, the non-layout data in this category is considered non-essential data and may not be accurate.

3.2.4 File name Category

The file name category of data includes the data that assigns human readable names to in-odes and other metadata. The file name category includes the names of files, and it allows the user to refer to a file by its name instead of its metadata address. At its core, this category of data includes only a file's name and its metadata address. Some file systems may also include file type information or temporal information, but that is not standard. In this section, we look at the general concepts of the file name category. The file name category is relatively simple and we need to look only at name-based file recovery.

3.2.3 Meta Data Category

The metadata category includes the data that describe a file or directory. The metadata category is where the descriptive data reside. Here we can find, for example, the last accessed time and the addresses of the data units that a file has allocated. Few tools explicitly identify metadata analysis; instead, it is typically merged with file name category analysis. Many metadata structures are stored in a fixed or dynamic-length table, and each entry has an address. When a file is deleted, the metadata entry is set to the unallocated state, and the OS may wipe some of the values in the entry. Analysis is conducted in the metadata category to determine more details about a specific file or to search for a file that meets certain requirements. This category tends to have more nonessential data than other categories. For example, the last accessed or written times may not be accurate or the OS may not have enforced the access control settings on a file; therefore, an investigator cannot conclude that a user did or did not have read access to a file.

3.2.2 Content Category:

The content category includes the file and directory content data. The content category includes the storage locations that are allocated to files and directories so that they can save data. The data in this category are typically organized into equal sized groups, which I am calling data units even though each file system has a unique name for them, such as cluster or block. A data unit is either in an allocated or unallocated state. There is typically some type of data structure that keeps track of the allocation status of each data unit. When a new file is created or an existing file is made larger, the OS searches for an unallocated data unit and allocates it to a file. The different search strategies will be discussed in the following "Allocation Strategies" section. When a file is deleted, the data units that were allocated to the file are set to the unallocated state and can be allocated to new files.

3.2.5 Application Category

The application category contains data that provide special features like the journal. Three different types of files have been created and deleted on the ext4 file system and their impact has been observed in all different categories. Some file systems contain data that belongs in the application category. These data are not essential to the file system, and they typically exist as special file system data instead of inside a normal file because it is more efficient. This section covers one of the most common application category features, which is called *journaling*. Technically, any file that an OS or an application creates could be designed as a feature in a file system. For example, Acme Software could decide that its OS would be faster if an area of the file system were reserved for an address book. Instead of saving names and addresses in a file, they would be saved to a special section of the volume. This might cause a performance improvement, but it is not essential for the file system.

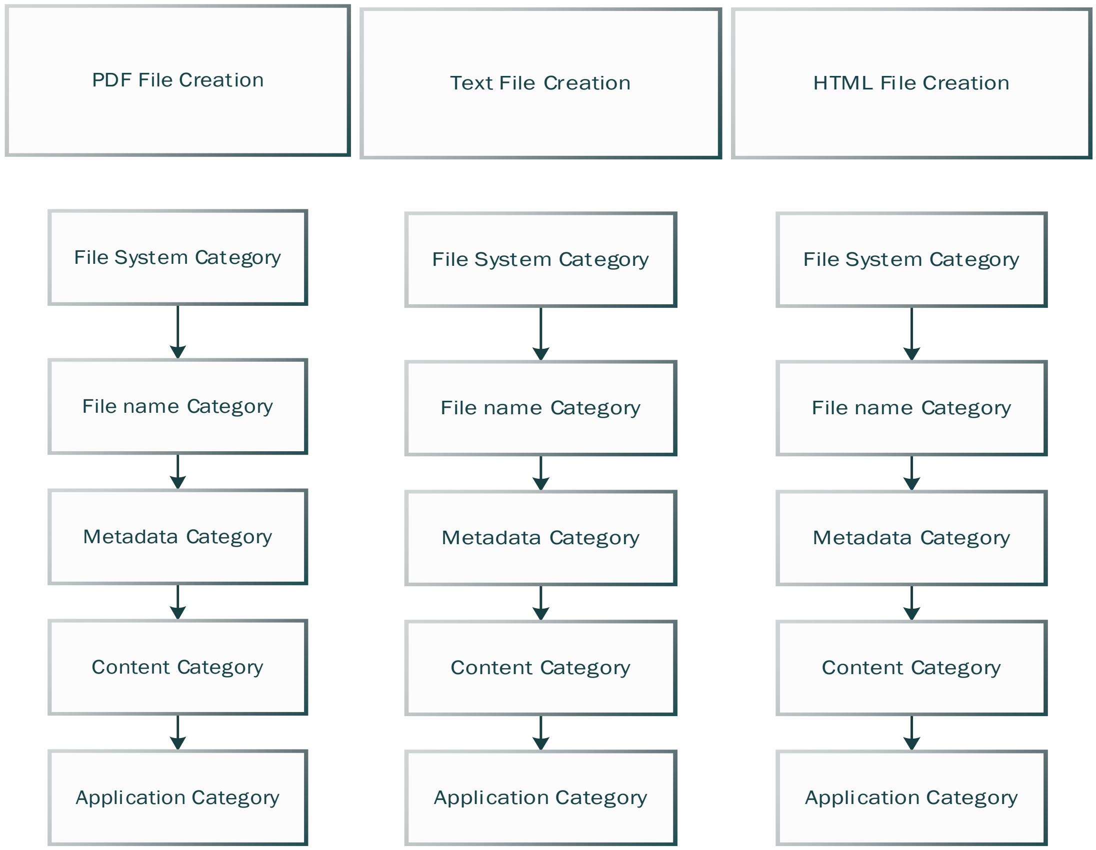


Figure 5: File Types Creation

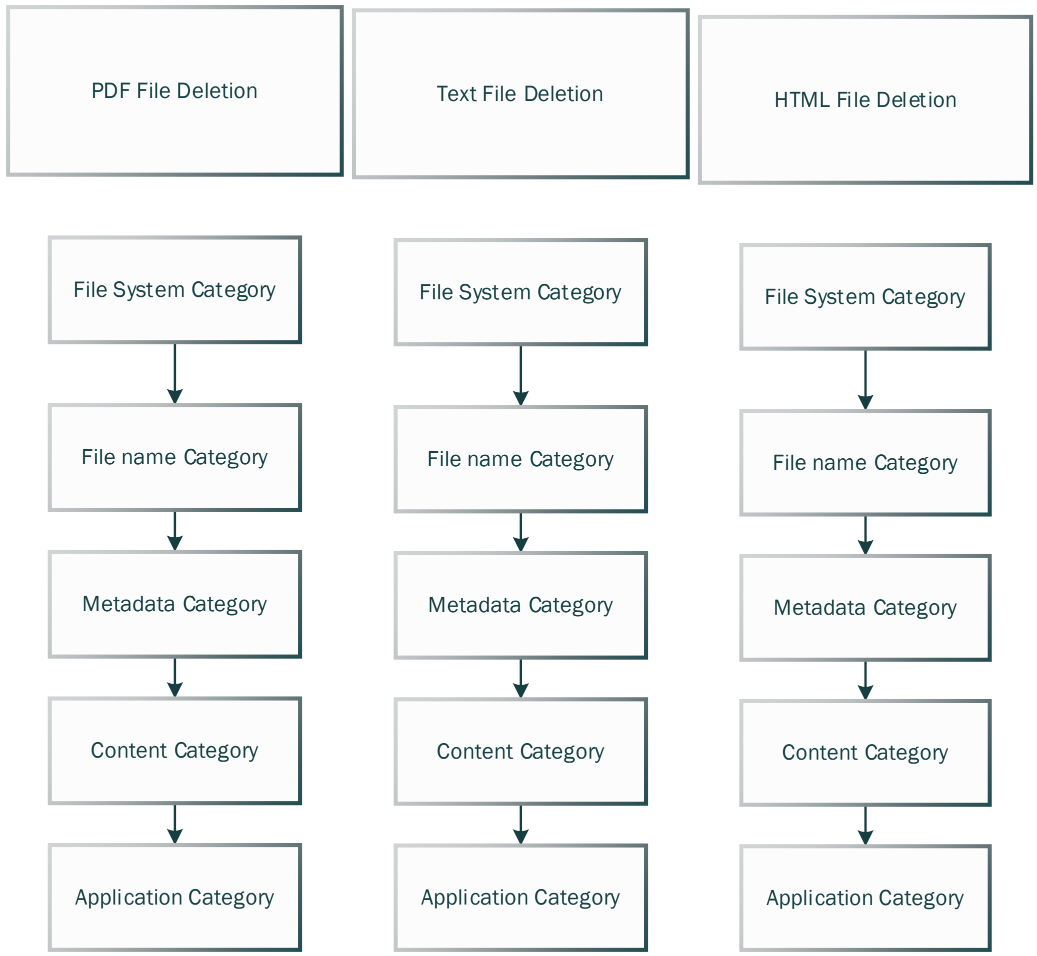


Figure 6: File Types Deletion

Chapter 4

FORENSIC EXAMINATION OF EXT4 FOR DIFFERENT TYPES OF FILES

4.1 Scenario (PDF file Creation)

A simple file pdf is copied into the file system and changes have been seen in all the different categories.

4.1.1 File System Category

Using the Fsstat command of Sleuth kit we get the information like free number of Block and free number of inodes in the file system as shown below

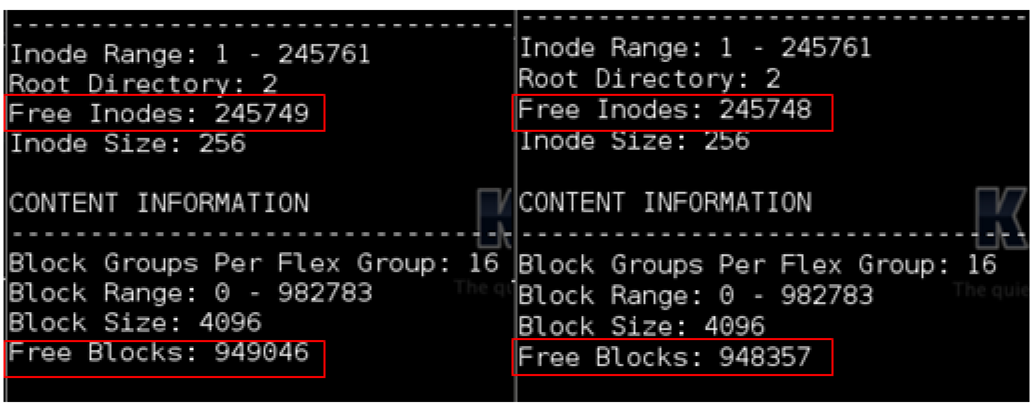


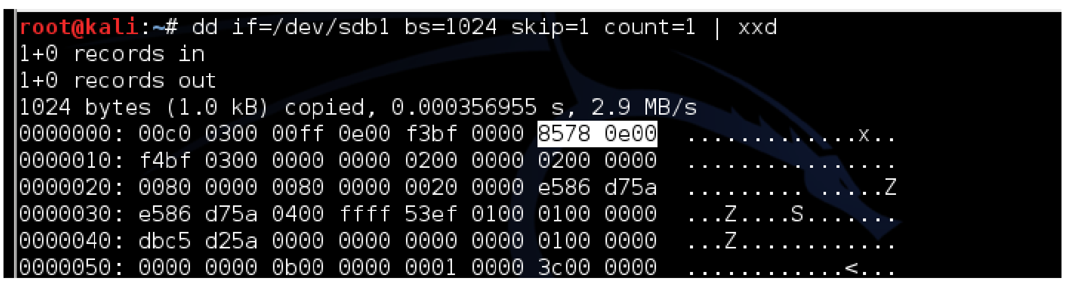
Figure 7: PDF Creation File System Category

When a file is created in the file system these changes occur. When we create a new file pdf file in the file system we can see that there are changes in the free inodes in the superblock group information. We can see that the number of free inodes has reduced from

245749 to 245748.We can also see that the number of free block count has also reduced from 949046 to 948357.Super Block is 1024 bytes in size, we extracted 1024 bytes of the superblock from the file system in hex format to confirm the output shown from the fsstat command of the sleuthkit. There are 4 bytes for number of free blocks starting at 0xc , this is in little endian format .4 bytes after the number of free blocks is the number of free in-odes starting from 0x10

Table 1. PDF File Creation Super Block Data Structure FS Category

Description	Offset	Number of Bytes
Total Inodes Count	0x0	4
Total Block Count	0x4	4
Free Block Count	0xC	4
Free Inode Count	0x10	4
Magic Number	0x38	2



```
root@kali:~# dd if=/dev/sdb1 bs=1024 skip=1 count=1 | xxd
1+0 records in
1+0 records out
1024 bytes (1.0 kB) copied, 0.000356955 s, 2.9 MB/s
0000000: 00c0 0300 00ff 0e00 f3bf 0000 8578 0e00  .............x..
0000010: f4bf 0300 0000 0000 0200 0000 0200 0000  ................
0000020: 0080 0000 0080 0000 0020 0000 e586 d75a  ......... .....Z
0000030: e586 d75a 0400 ffff 53ef 0100 0100 0000  ...Z....S.......
0000040: dbc5 d25a 0000 0000 0000 0000 0100 0000  ...Z............
0000050: 0000 0000 0b00 0000 0001 0000 3c00 0000  ............<...
```

Figure 8: PDF Creation FS Category (HEX)

This is in little endian format, reading the hex values we have 000e7885. Converting it into Decimal values we get 948357. This is the free number of blocks which was also stated by fsstat command. Looking at the number of free Inodes in hex output we can see that

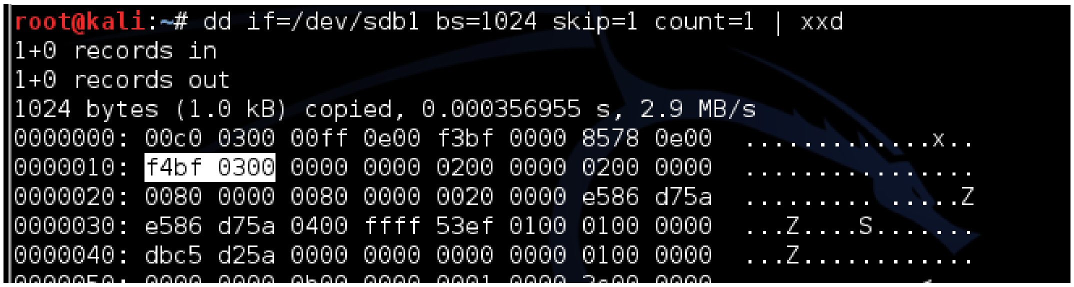


Figure 9: PDF File system Category (HEX)

Reading the data structures we can see that the number of free inodes in the file system is 0003bff4 which is equal to 245748, which is the same as mentioned in the fsstat command of the sleuth kit.

4.1.2 File Name category:

The file name category of data includes the data structures that store the name of each file and directory. This section describes where the data are stored and how to analyze them. Using the sleuth kit command to analyze this category we get

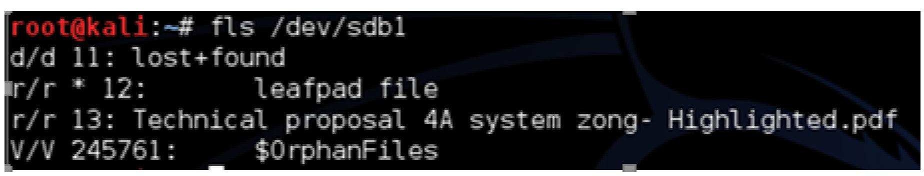


Figure 10: PDF Creation FN Category

We can see the Pdf file r/r shows that it is a file, number 13 shows the inode number.

4.1.3 Meta data category

Using the istat command of the sleuth kit we can see the Meta data associated with this file. We know that inode 13 is associated with this file so we will use inode 13 with this command.

```
root@kali:~# istat /dev/sdb1 13
inode: 13
Allocated
Group: 0
Generation Id: 635268831
uid / gid: 0 / 0
mode: rrwxrw-rw-
Flags: Extents,
size: 2820069
num of links: 1

Inode Times:
Accessed:       2018-04-15 08:56:12.000000000 (PKT)
File Modified:  2018-02-02 10:36:54.000000000 (PKT)
Inode Modified: 2018-04-15 08:56:12.396513566 (PKT)
File Created:   2018-04-15 08:56:12.384513566 (PKT)

Direct Blocks:
33010 33011 33012 33013 33014 33015 33016 33017
33018 33019 33020 33021 33022 33023 33024 33025
33026 33027 33028 33029 33030 33031 33032 33033
33034 33035 33036 33037 33038 33039 33040 33041
33042 33043 33044 33045 33046 33047 33048 33049
```

Figure 11 : PDF Creation MD Category

Inode 13 belongs to block group 0 as mentioned in the above screenshot and it can also be confirmed from the fsstat command

```
Group: 0:
  Inode Range: 1 - 8192
  Block Range: 0 - 32767
  Layout:
    Super Block: 0 - 0
    Group Descriptor Table: 1 - 1
    Group Descriptor Growth Blocks: 2 - 240
    Data bitmap: 241 - 241
    Inode bitmap: 257 - 257
    Inode Table: 273 - 784
    Data Blocks: 785 - 0
  Free Inodes: 8180 (0%)
  Free Blocks: 24297 (0%)
  Total Directories: 2
```

Figure 12 : PDF Creation Meta Data Category

As the inode size is 256 bytes, and the block size is 4096 bytes we can see that, total number of inodes in one block will be 16. Inode table range is from blocks 273 to 784. Inode 13 will be in the first block which is 273. Processing blk 273 in hex we get

```
0000c00: f681 0000 e507 2b00 5ccd d25a 5ccd d25a  ......+.\..Z\..Z
0000c10: f6f8 735a 0000 0000 0000 0100 8815 0000  ..sZ............
0000c20: 0000 0800 0100 0000 0af3 0100 0400 0000  ................
0000c30: 0000 0000 0000 0000 b102 0000 f280 0000  ................
0000c40: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000c50: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000c60: 0000 0000 df6e dd25 0000 0000 0000 0000  .....n.%........
0000c70: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000c80: 1c00 0000 7844 895e 0000 0000 0000 0000  ....xD.^........
0000c90: 5ccd d25a 78d8 ac5b 0000 0000 0000 0000  \..Zx..[........
0000ca0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000cb0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000cc0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000cd0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000ce0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000cf0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
```

Figure 13 : PDF Creation MD Hex

Results after manual processing

Table 2.PDF File Inode Data Structure MD Category

Description	Offset	Number of Bytes	Processed Hexadecimal Bytes in Big endian	Decimal Output
Size in bytes	0x4	4	002b07e5	002b07e5
Access Time	0x08	4	5ad2cd5c	8:56:12 AM April 15 2018
Inode Change Time	0xC	4	5ad2cd5c	8:56:12 AM April 15 2018
File Modification Time	0x10	4	5a73f8f6	10:36:54 AM Feb 2 2018

Number of Extents	0x2A	2	0001	1
Number of Blocks In extents	0x38	2	02b1	689
Upper 16 Bits of Block Address	0x3A	2	0000	0
Lower 32 Bits of Block address	0x3C	4	80f2	33010

4.1.4 Content Category:

First we check the allocation status of a block

Figure 14 : PDF Creation Content Category

To check which inode is connected with this block

Figure 15: PDF Creation Content Category 2

To view what actually resides in one of the blocks associated with the file we get the following

Figure 16 : PDF Creation Content Category HEX

4.1.5 Application category

In this category the journaling is performed by the file system. When copy and paste the same file in the file system and have a look at the journal

```
JBlk    Description
0:      Superblock (seq: 0)
sb version: 4
sb version: 4
sb feature_compat flags 0x00000000
sb feature_incompat flags 0x00000000
sb feature_ro_incompat flags 0x00000000
1:      Allocated Descriptor Block (seq: 44)
2:      Allocated FS Block 273
3:      Allocated Commit Block (seq: 44, sec: 1524087369.54
4:      Allocated Descriptor Block (seq: 45)
5:      Allocated FS Block 257
6:      Allocated FS Block 1
7:      Allocated FS Block 273
8:      Allocated FS Block 8465
9:      Allocated FS Block 0
10:     Allocated FS Block 242
11:     Allocated Commit Block (seq: 45, sec: 1524087397.10
```

Figure 17 : PDF Creation Content Category 1

When we have a look at the journal block we can see that all the changes are in the block group 0 and these are all Meta data changes. There is no change in the data blocks.

```
Group: 0:
  Inode Range: 1 - 8192
  Block Range: 0 - 32767
  Layout:
    Super Block: 0 - 0
    Group Descriptor Table: 1 - 1
    Group Descriptor Growth Blocks: 2 - 240
    Data bitmap: 241 - 241
    Inode bitmap: 257 - 257
    Inode Table: 273 - 784
    Data Blocks: 785 - 0
  Free Inodes: 8180 (0%)
  Free Blocks: 24297 (0%)
  Total Directories: 2
```

Figure 18 : PDF Creation Content Category 2

Block 257 belongs to block group 0. Block 257 is the inode bitmap in which the change has occurred.An inode has changed due to which a change in block 273 has occurred.Similarly there is a change in the Group descriptor table which is block 1 in Block group 0.

4.2 Scenario (PDF File Deletion)

Similarly when we delete a pdf file we can see that there are changes in all of these categories

4.2.1 File System Category

First of all we will see changes in the file system category we can see that the number of free blocks and number of free inodes has changed after we have deleted the pdf file.

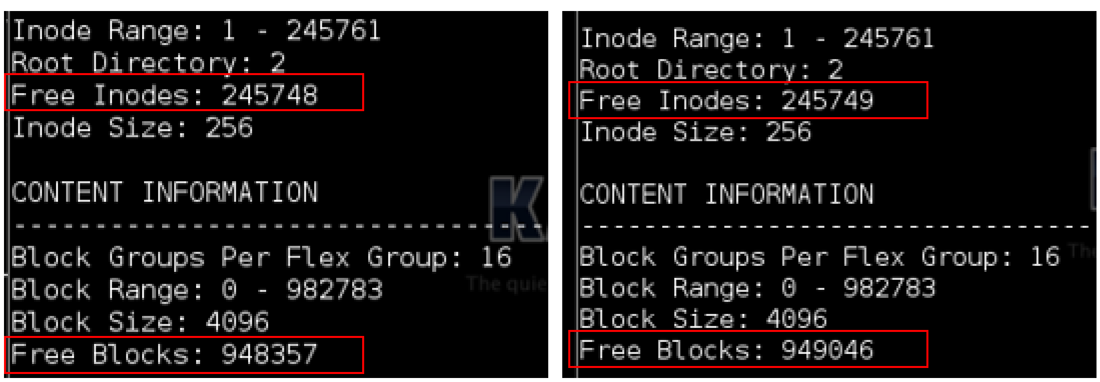


Figure 19 : PDF Deletion FS Category

Verifying this information through the Hex Editor. Relevant Data Structures for Super-block is as follows

Table 3. PDF File Inode Data Structure MD Category

Description	Offset	Number of Bytes
Total Inodes Count	0x0	4
Total Block Count	0x4	4
Free Block Count	0xC	4
Free Inode Count	0x10	4
Magic Number	0x38	2

```
0000000: 00c0 0300 00ff 0e00 f3bf 0000 367b 0e00
0000010: f5bf 0300 0000 0000 0200 0000 0200 0000
0000020: 0080 0000 0080 0000 0020 0000 9c00 d85a
0000030: 9c00 d85a 0900 ffff 53ef 0100 0100 0000
0000040: dbc5 d25a 0000 0000 0000 0000 0100 0000
0000050: 0000 0000 0b00 0000 0001 0000 3c00 0000
0000060: 4602 0000 7b00 0000 56cf c6f7 440a 4d74
0000070: 9ce0 cf5f 7a22 74e0 0000 0000 0000 0000
0000080: 0000 0000 0000 0000 2f6d 6564 6961 2f35
```

Figure 20 : PDF Deletion FS Hex

Looking at number of free Block count we can see that it starts at 0x0C and it is 4 bytes long, converting it into big endian the value is 0e7b36 which in decimal is 949046 which is the same as shown in the screen shot after deletion. Similarly number of free block count can also be shown in the Hex output of the super block. Looking at the data structure we can see that the number of free Inodes count start at offset 0x10 which is also 4 bytes long. Converting the value in big endian we get 0003bff5, converting this value into decimal we get 245749. This can also be confirmed from the screenshot after deletion.

4.2.2 File Name Category:

Examining the file name category after the deletion of pdf file we can see the following output.

Figure 21 : PDF Deletion FN Category

We can see that the file “Technical proposal 4A system zong-Highlighted.pdf “is associated with inode 12 and is still shown after the file is deleted. But there is the indication that the file is deleted as it is marked with *. Previously when the file was created the file was shown in a different way without a “*”.

4.2.3 Meta data category

We know that the inode associated with the file was 12. Using sleuthkit tool we will look into the statistics of this inode.

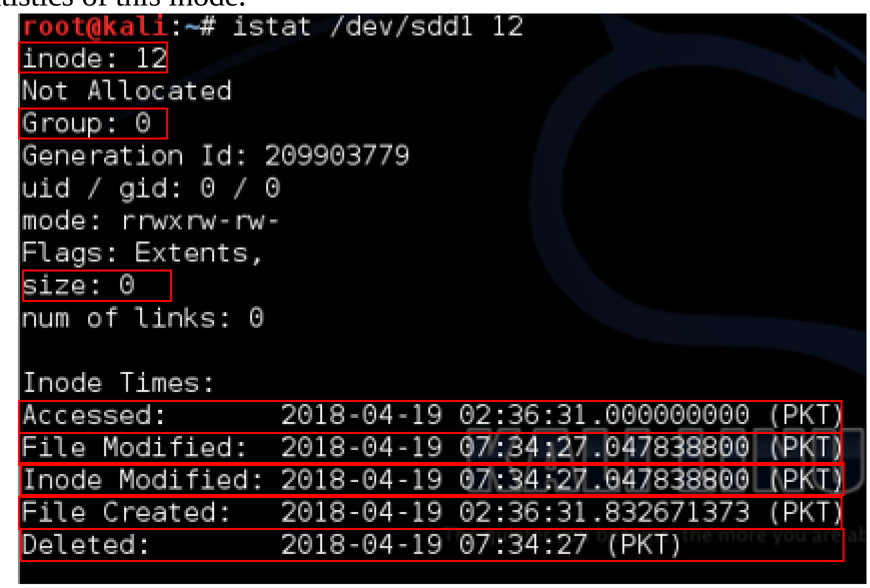


Figure 22 : PDF File Deletion MD Category

From here we can see that the blocks associated with this file are removed and the size is reduced to zero as well. The inode still exists and contains some of the information associated with the previous file until this inode is again allocated to another file.

Now we will parse the inode structure at Hex level to confirm what information is left in the inode associated with the file.As the inode number is 12, it belongs to block group 0 as shown in the following screenshot.

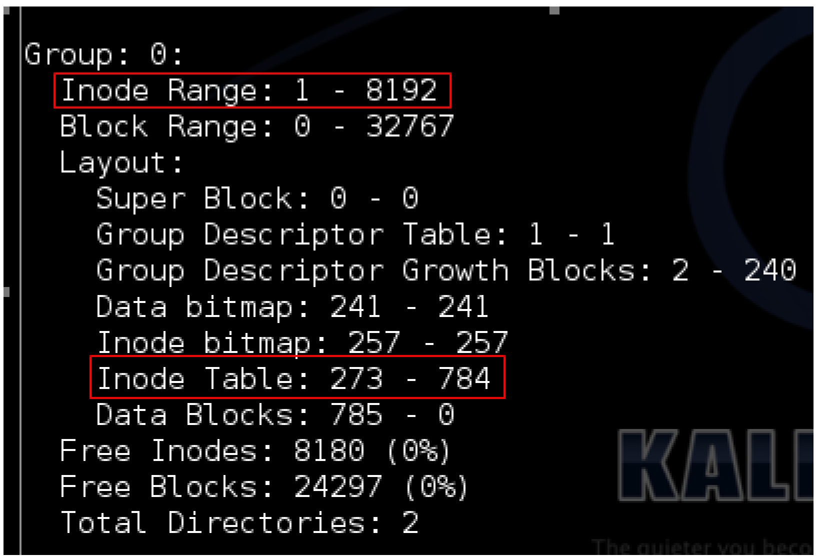


Figure 23 : PDF Deletion MD Verify

Inode range in this group is from 1-8192. As the inode size is 256 bytes and the block size is 4096 bytes, there are 256 bytes per block in this group. The inode table in the group starts from block 273.As there are 16 inodes per block, inode 12 can be found in block 273.Getting the hex output of block 273 we get

```
0000b00: f681 0000 0000 0000 5fba d75a 3300 d85a
0000b10: 3300 d85a 3300 d85a 0000 0000 0000 0000
0000b20: 0000 0800 0100 0000 0af3 0000 0400 0000
0000b30: 0000 0000 0000 0000 0000 0000 0000 0000
0000b40: 0000 0000 0000 0000 0000 0000 0000 0000
0000b50: 0000 0000 0000 0000 0000 0000 0000 0000
0000b60: 0000 0000 a3e0 820c 0000 0000 0000 0000
0000b70: 0000 0000 0000 0000 0000 0000 0000 0000
0000b80: 1c00 0000 40d9 670b 40d9 670b 0000 0000
0000b90: 5fba d75a 343a 86c6 0000 0000 0000 0000
0000ba0: 0000 0000 0000 0000 0000 0000 0000 0000
0000bb0: 0000 0000 0000 0000 0000 0000 0000 0000
0000bc0: 0000 0000 0000 0000 0000 0000 0000 0000
0000bd0: 0000 0000 0000 0000 0000 0000 0000 0000
0000be0: 0000 0000 0000 0000 0000 0000 0000 0000
0000bf0: 0000 0000 0000 0000 0000 0000 0000 0000
```

Figure 24 : PDF Deletion MD HEX

Processing the inode structure using the above information we can see that file of the size has been reduced to 0 as bytes 4-7 show 00000000 which proves the sleuthkit output as well. Access time of the file starts from byte 0x08 which is 4 bytes long, processing the inode for access time we get the Hex value of access time 5ad7ba5f. Now we will verify the Deletion time of the PDF file, processing the inode structure in Hex we can see that the deletion time in hex is 5ad80033. Processing the Extent Structure within the inode structure we can see that the number of extents within the inode, number of blocks associated with the inode and starting address of the block address has changed to zero.

Results After manual Processing

Table 4.PDF FILE DELETION INODE DATA STRUCTURE MD CATEGORY

Description	Offset	Number of Bytes	Processed Hexadecimal Bytes in Big endian	Decimal Output
Size in bytes	0x4	4	00000000	0
Access Time	0x08	4	5ad7ba5f	2:36:31 AM April 19 2018
Inode Change Time	0xC	4	5ad80033	7:34:27 AM April 19 2018
File Modification Time	0x10	4	5ad80033	7:34:27 AM April 19 2018
Deletion Time	0x14	4	5ad80033	7:34:27 AM April 19 2018
Number of Extents	0x2A	2	0000	0
Number of Blocks In extents	0x38	2	0000	0
Upper 16 Bits of Block Address	0x3A	2	0000	0
Lower 32 Bits of Block address	0x3C	4	00000000	0

4.2.4 Content Category:

When the PDF file was created tracked the record of Blocks associated with the file which were starting from 33010 and ending on 33698. Now we will again check the allocation status of the block which is shown as follows.

Figure 25 : PDF Deletion Content Category

We can see that the allocation status of the block has changed and is not allocated. The content actually resides on the block even after deletion

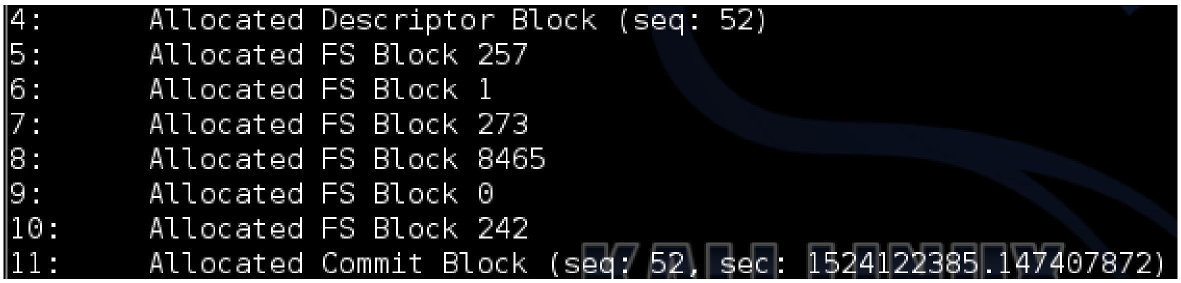
Figure 26 : PDF Deletion Content Category Hex

As this is a pdf, it is not simply understandable, but the content resides on the block and is not zeroed out .To find out to which inode this block is associated with we have the following output.

Figure 27 : PDF Deletion Content Category Inode

We can see that the inode connection with the block is zeroed out and connection of a block with the inode cannot be found.

4.2.5 Application Category:



```
4:      Allocated Descriptor Block (seq: 52)
5:      Allocated FS Block 257
6:      Allocated FS Block 1
7:      Allocated FS Block 273
8:      Allocated FS Block 8465
9:      Allocated FS Block 0
10:     Allocated FS Block 242
11:     Allocated Commit Block (seq: 52, sec: 1524122385.147407872)
```

Figure 28 : PDF Deletion App Category

Analyzing the journal block after file deletion it can be seen that the there are changes in the blocks associated with the file. Block 257 belongs to block group 0. Block 257 is the inode bitmap in which the change has occurred. An inode has changed due to which a change in block 273 has occurred. Similarly there is a change in the Group descriptor table which is block 1 in Block group 0.

4.2.6 PDF File Deletion Summary

Table 5. PDF FILE DELETION SUMMARY

Categories	Data Persistence
File System	✗
File Name	✓

Meta Data	✓
Content	✓
Application	✓

4.3 Scenario (Text File Creation)

Now we will analyze creation of a text file in the ex4 file system.

4.3.1 File System Category:

Status of free Inode and Block count before and after creation of the text file is shown in the following diagram.

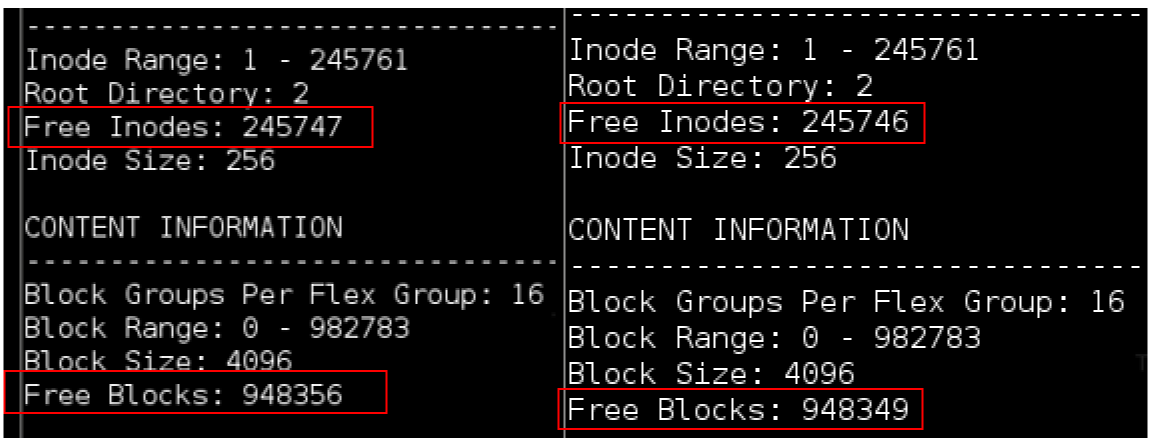
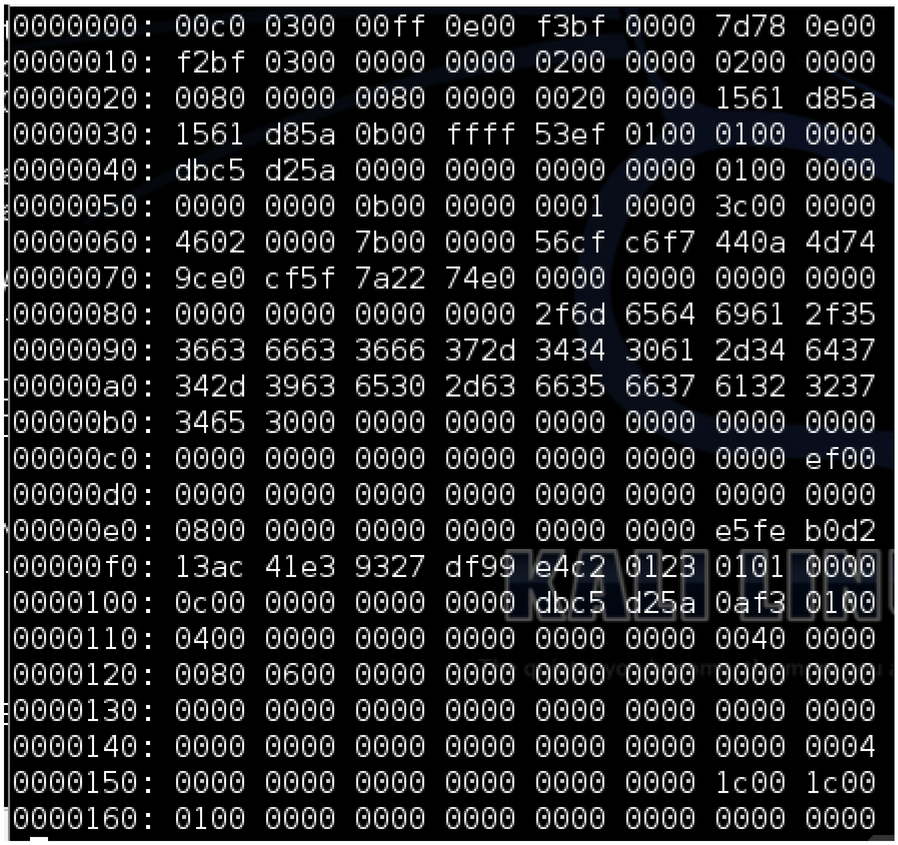


Figure 29 : Text File Creation FS Category

We can see the free inode count and block count has changed in the above screenshot. Verifying this information through the Hex Editor. Relevant Data Structures for Superblock is as follows.

Table 6. TEXT FILE CREATION SUPER BLOCK DATA STRUCTURE FS CATEGORY

Description	Offset	Number of Bytes
Total Inodes Count	0x0	4
Total Block Count	0x4	4
Free Block Count	0xC	4
Free Inode Count	0x10	4
Magic Number	0x38	2



```
0000000: 00c0 0300 00ff 0e00 f3bf 0000 7d78 0e00
0000010: f2bf 0300 0000 0000 0200 0000 0200 0000
0000020: 0080 0000 0080 0000 0020 0000 1561 d85a
0000030: 1561 d85a 0b00 ffff 53ef 0100 0100 0000
0000040: dbc5 d25a 0000 0000 0000 0000 0100 0000
0000050: 0000 0000 0b00 0000 0001 0000 3c00 0000
0000060: 4602 0000 7b00 0000 56cf c6f7 440a 4d74
0000070: 9ce0 cf5f 7a22 74e0 0000 0000 0000 0000
0000080: 0000 0000 0000 0000 2f6d 6564 6961 2f35
0000090: 3663 6663 3666 372d 3434 3061 2d34 6437
00000a0: 342d 3963 6530 2d63 6635 6637 6132 3237
00000b0: 3465 3000 0000 0000 0000 0000 0000 0000
00000c0: 0000 0000 0000 0000 0000 0000 0000 ef00
00000d0: 0000 0000 0000 0000 0000 0000 0000 0000
00000e0: 0800 0000 0000 0000 0000 0000 e5fe b0d2
00000f0: 13ac 41e3 9327 df99 e4c2 0123 0101 0000
0000100: 0c00 0000 0000 0000 dbc5 d25a 0af3 0100
0000110: 0400 0000 0000 0000 0000 0000 0040 0000
0000120: 0080 0600 0000 0000 0000 0000 0000 0000
0000130: 0000 0000 0000 0000 0000 0000 0000 0000
0000140: 0000 0000 0000 0000 0000 0000 0000 0004
0000150: 0000 0000 0000 0000 0000 0000 1c00 1c00
0000160: 0100 0000 0000 0000 0000 0000 0000 0000
```

Figure 30 : Text File Creation FS (Hex)

As the free block count is at offset 0xC which is 4 bytes in size, we can see that the number of free blocks in hexadecimal is 000e787d which is equal to 948349 which is equivalent to number of free blocks as mentioned in sleuth kit output.

4.3.2 Filename category:

Analyzing the filename category after creation of the text file we get the following output.

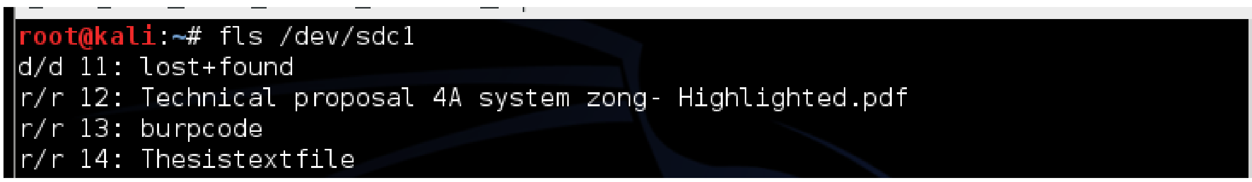


Figure 31 : Text File Creation File Name Category

We can see that the thesis text file is shown here and the inode associated with this file is inode 14.

4.3.3 Metadata Category:

To analyze the file creation in the metadata category we process the inode number in the sleuthkit and see what metadata information of the file we can get.

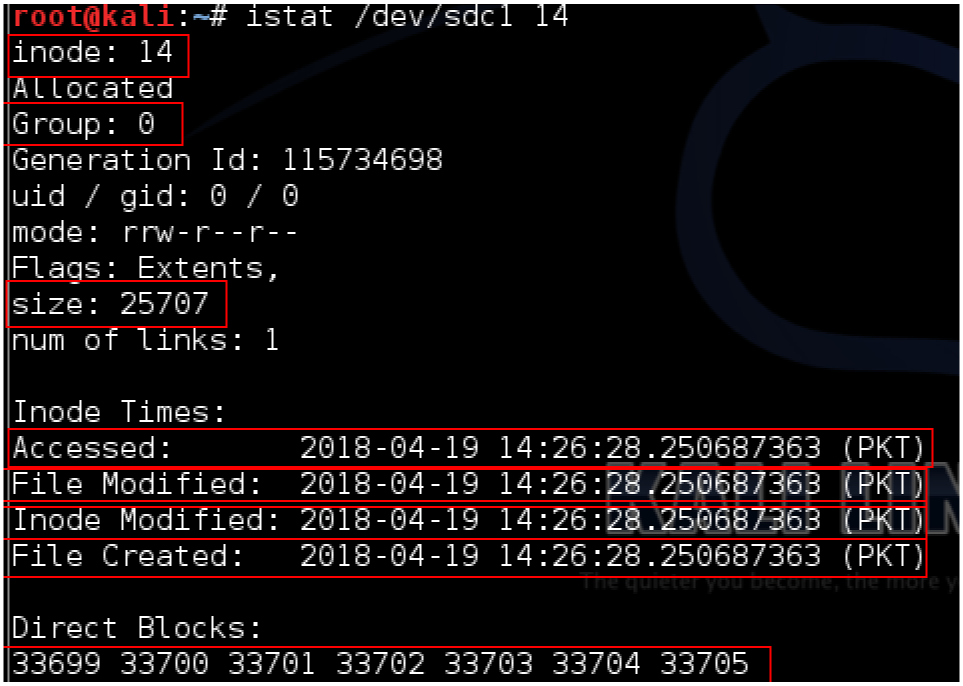


Figure 32: Text File Creation MD Category

Inode for this file is also found in block group 0. The size of the file is 25707 bytes. Times associated with this inode along with blocks are also shown in the above screenshot. Inode range in this group is from 1-8192. As the inode size is 256 bytes and the block size is 4096 bytes, there are 4096/256 bytes inodes per block in this group. The inode table in the group starts from block 273.

As there are 16 inodes per block, inode 14 can be found in block 273. Processing block 273 and moving to 14th 256 byte structure in this block we have the following Hex output.

```
0000d00: a481 0000 6b64 0000 c460 d85a c460 d85a  ....kd...`.Z.`.Z
0000d10: c460 d85a 0000 0000 0000 0100 3800 0000  .`.Z........8...
0000d20: 0000 0800 0100 0000 0af3 0100 0400 0000  ................
0000d30: 0000 0000 0000 0000 0700 0000 a383 0000  ................
0000d40: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000d50: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000d60: 0000 0000 aaf8 e506 0000 0000 0000 0000  ................
0000d70: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000d80: 1c00 0000 0cbe c43b 0cbe c43b 0cbe c43b  .......;...;...;
0000d90: c460 d85a 0cbe c43b 0000 0000 0000 0000  .`.Z...;........
0000da0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000db0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000dc0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000dd0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000de0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
```

Figure 33 : Text File Creation MD Hex

Results After manual Processing.

TABLE 7. TEXT FILE CREATION INODE DATA STRUCTURE MD CATEGORY

Description	Offset	Number of Bytes	Processed Hexadecimal Bytes in Big endian	Decimal Output
Size in bytes	0x4	4	0000646b	25707
Access Time	0x08	4	5ad860c4	2:26:28 PM April 19 2018
Inode Change Time	0xC	4	5ad860c4	2:26:28 PM April 19 2018
File Modification Time	0x10	4	5ad860c4	2:26:28 PM April 19 2018
Number of Extents	0x2A	2	0001	1
Number of Blocks In extents	0x38	2	0007	7

Upper 16 Bits of Block Address	0x3A	2	0000	0
Lower 32 Bits of Block address	0x3C	4	000083a3	33699

This confirms the sleuth kit output.

4.3.4 Content category:

In this category we will check the allocation status of a block.

Figure 34 : Text File Creation Content Category

Now we will verify which inode is allocated to this block.

Figure 35: Text File Creation Content (Inode)

Now we will process the content of this block.

```
root@kali:~# blkcat /dev/sdc1 33699
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
```

Figure 36 : Text File Creation (Hex)

We can see that the content is in the data block which is associated with the file.

4.4 Scenario (Text File Deletion):

Now we will analyze creation of a text file in the ex4 file system

4.4.1 File System Category:

Status of free Inode and Block count before and after creation of the text file is shown in the following diagram.

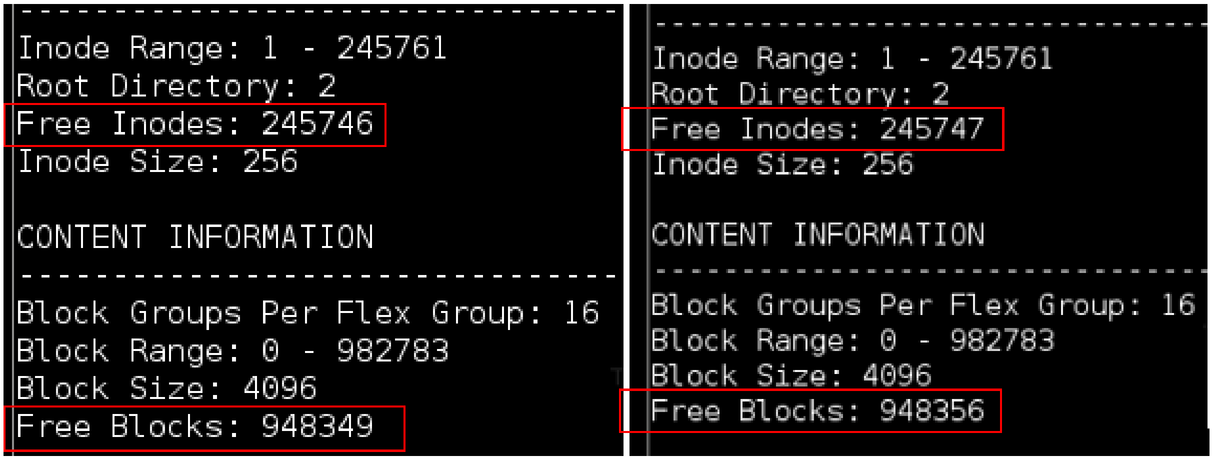


Figure 37 : Text File Deletion FS Category

We can see the free inode count and block count has changed in the above screenshot. Verifying this information through the Hex Editor. Relevant Data Structures for Superblock is as follows

Table 8. TEXT FILE DELETION SUPER BLOCK DATA STRUCTURE FS CATEGORY

Description	Offset	Number of Bytes
Total Inodes Count	0x0	4
Total Block Count	0x4	4
Free Block Count	0xC	4
Free Inode Count	0x10	4
Magic Number	0x38	2

Figure 38 : Text File Deletion FS Hex

As the free block count is at offset 0xC which is 4 bytes in size, we can see that the number of free blocks in hexadecimal is 000e787d which is equal to 948349 which is equivalent to number of free blocks as mentioned in sleuth kit output.

4.4.2 File Name Category:

Now we will analyze the text file deletion in file name category.

Figure 39 : Text File Deletion File Name Category

Although the file is deleted but it is still visible along with its inode as a deleted file.

4.4.4 Metadata Category:

To analyze the file deletion in the metadata category we process the inode number in the sleuth kit and see what metadata information of the file we can get.

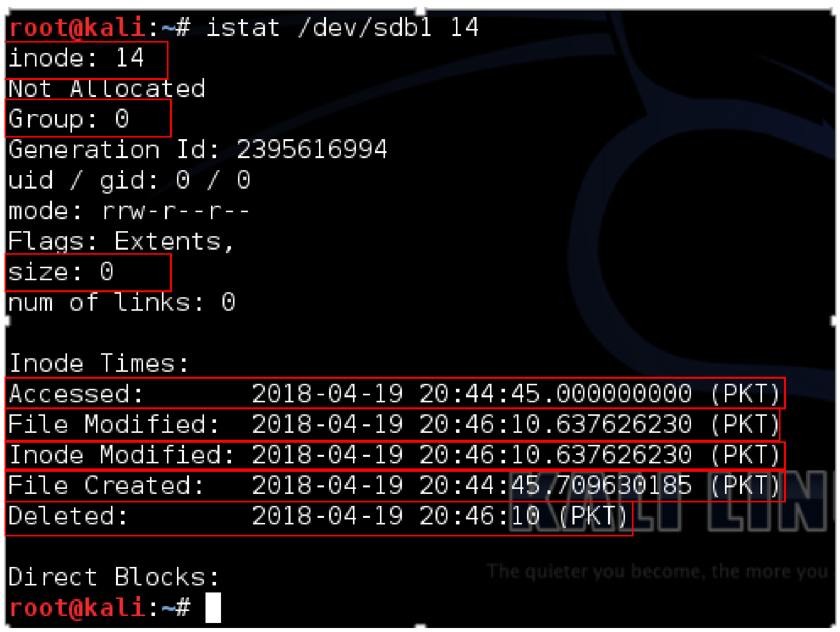


Figure 40 : Text File Deletion MD Category

Inode for this file is also found in block group 0. The size of the file has reduced to 0 bytes. Times associated with this inode along with blocks are also shown in the above screenshot. Inode range in this group is from 1-8192. As the inode size is 256 bytes and the block size is 4096 bytes, there are 4096/256 bytes inodes per block in this group. The inode table in the group starts from block 273. As there are 16 inodes per block, inode 14 can be found in block 273. Processing block 273 and moving to 14th 256 byte structure in this block we have the following Hex output.

```
0000d00: a481 0000 0000 0000 6db9 d85a c2b9 d85a  ........m..Z...Z
0000d10: c2b9 d85a c2b9 d85a 0000 0000 0000 0000  ...Z...Z........
0000d20: 0000 0800 0100 0000 0af3 0000 0400 0000  ................
0000d30: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000d40: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000d50: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000d60: 0000 0000 e236 ca8e 0000 0000 0000 0000  .....6..........
0000d70: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000d80: 1c00 0000 d89d 0598 d89d 0598 0000 0000  ................
0000d90: 6db9 d85a a463 30a9 0000 0000 0000 0000  m..Z.c0.........
0000da0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000db0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000dc0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000dd0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000de0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000df0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
```

Figure 41 : Text File Deletion MD HEX

Results After manual Processing.

Table 9. TEXT FILE DELETION SUPERBLOCK DATA STRUCTURE FS CATEGORY

Description	Offset	Number of Bytes	Processed Hexadecimal Bytes in Big endian	Decimal Output
Size in bytes	0x4	4	0	0
Access Time	0x08	4	5ad8b96d	8:44:45 PM April 19 2018

Inode Change Time	0xC	4	5ad8b9c2	8:46:10 PM April 19 2018
File Modification Time	0x10	4	5ad8b9c2	8:46:10 PM April 19 2018
Deletion Time	0x14	4	5ad8b9c2	8:46:10 PM April 19 2018
Number of Extents	0x2A	2	0000	0
Number of Blocks In extents	0x38	2	0000	0
Upper 16 Bits of Block Address	0x3A	2	0000	0
Lower 32 Bits of Block address	0x3C	4	00000000	0

This also confirms the sleuth kit output.

4.4.5 Content Category:

In this category we will check the allocation status of a block.

Figure 42 : Text File Deletion Content Category

Now we will verify which inode is allocated to this block.

Figure 43 : Text File Deletion Content (Inode)

Now we will process the content of this block.

```
root@kali:~# blkcat /dev/sdb1 33699
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
My name is moeed hussain My name is moeed hussain My name is moeed hussain
```

Figure 44 : Text File Deletion CC Hex

It can be seen that the data still resides in the block even after deletion and is recoverable through data carving process.

4.4.6 Application Category

```
Allocated Commit Block (seq: 87, sec: 1529323351.450980608)
Unallocated Descriptor Block (seq: 81)
Unallocated FS Block 257
Unallocated FS Block 1
Unallocated FS Block 273
Unallocated FS Block 8465
```

Figure 45 : Text File Deletion App Cat

Analyzing the journal block after file deletion it can be seen that the there are changes in the blocks associated with the file. There are changes in the block 257 belongs to block group 0. Block 257 is the inode bitmap in which the change has occurred. An inode has changed due to which a change in block 273 has occurred. Similarly there is a change in the Group descriptor table which is block 1 in Block group 0.

4.4.7 Text File Deletion Summary

TABLE 10. TEXT FILE DELETION SUMMARY

Categories	Data Persistence
File System	×
File Name	✓
Meta Data	✓
Content	✓
Application	✓

4.5 Scenario (HTML File Creation)

Now we will analyze HTML file creation in all 5 different categories

4.5.1 File System Category

First of all we will see changes in the file system category .We can see that the number of free blocks and number of free inodes has changed after we have created the html file.

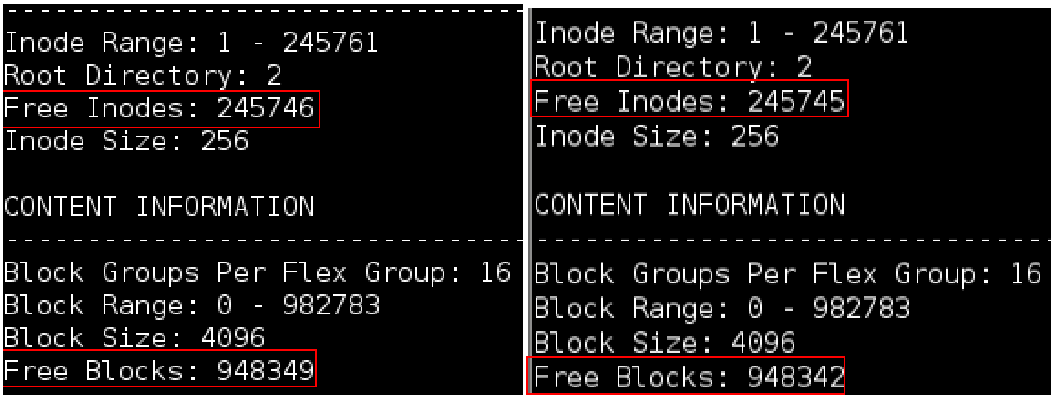


Figure 46 : HTML File Creation FS category

We can see that the number of free inodes has reduced from 245746 to 245745.We can also see that the number of free block count has also reduced from 948349 to 948342This can also be verified from the Hex output of the super block.

Table 11. HTML FILE CREATION SUPER BLOCK DA-
TA STRUCTURE FS CATEGORY

Description	Offset	Number of Bytes
Total Inodes Count	0x0	4
Total Block Count	0x4	4
Free Block Count	0xC	4
Free Inode Count	0x10	4
Magic Number	0x38	2

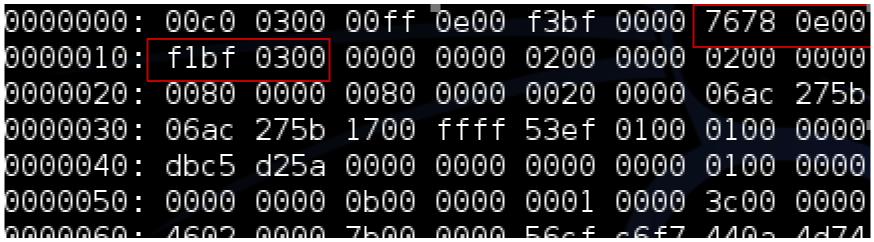


Figure 47 : HTML File Creation (HEX)

As the free block count is at offset 0xC which is 4 bytes in size, we can see that the number of free blocks in hexadecimal is 000e7876 which is equal to 948342 which is equivalent to number of free blocks as mentioned in sleuth kit output. Similarly we can also see that the number of free inodes start at 0x10 which is 4 bytes in size and is 0003bff1 in hexadecimal and is 245745 in decimal, this confirms the sleuth kit output.

4.5.2 Filename Category:

Analyzing the filename category after creation of the Html file we get the following output.

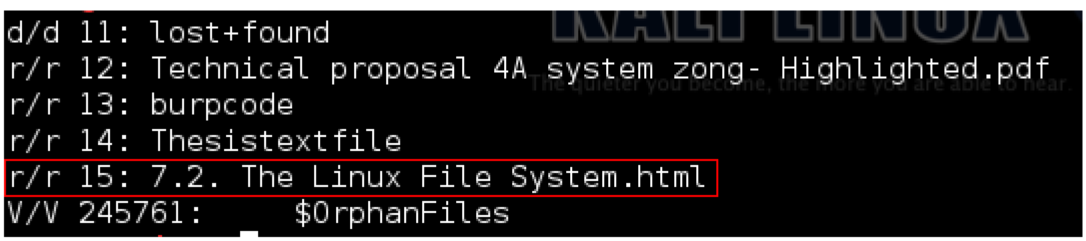


Figure 48 : HTML File Name Cat

We can see that the HTML file is created and inode associated with the file is 15

4.5.3 Metadata Category:

To analyze the file creation in the metadata category we process the inode number in the sleuth kit and see what metadata information of the file we can get.

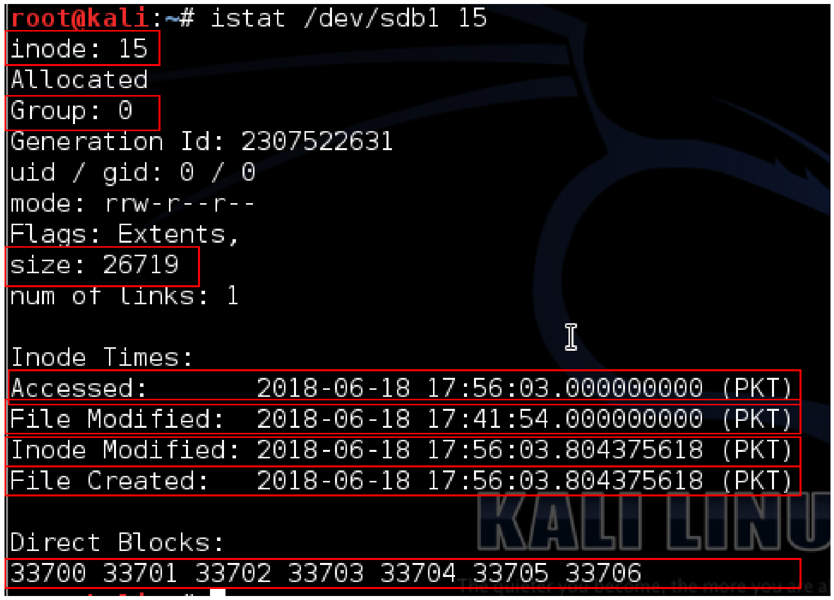


Figure 49 : HTML File Creation MD Category

Inode for this file is also found in block group 0. The size of the file is 26719 bytes. Times associated with this inode along with blocks are also shown in the above screenshot. Inode range in this group is from 1-8192. As the inode size is 256 bytes and the block size is 4096 bytes, there are 4096/256 bytes inodes per block in this group. The inode table in the group starts from block 273.As there are 16 inodes per block, inode 15 can be found in block 273. Processing block 273 and moving to 15th 256 byte structure in this block we have the following Hex output.

```
0000e00: a481 0000 5f68 0000 e3ab 275b e3ab 275b
0000e10: 92a8 275b 0000 0000 0000 0100 3800 0000
0000e20: 0000 0800 0100 0000 0af3 0100 0400 0000
0000e30: 0000 0000 0000 0000 0700 0000 a483 0000
0000e40: 0000 0000 0000 0000 0000 0000 0000 0000
0000e50: 0000 0000 0000 0000 0000 0000 0000 0000
0000e60: 0000 0000 4700 8a89 0000 0000 0000 0000
0000e70: 0000 0000 0000 0000 0000 0000 0000 0000
0000e80: 1c00 0000 0831 c7bf 0000 0000 0000 0000
0000e90: e3ab 275b 0831 c7bf 0000 0000 0000 0000
0000ea0: 0000 0000 0000 0000 0000 0000 0000 0000
0000eb0: 0000 0000 0000 0000 0000 0000 0000 0000
0000ec0: 0000 0000 0000 0000 0000 0000 0000 0000
0000ed0: 0000 0000 0000 0000 0000 0000 0000 0000
0000ee0: 0000 0000 0000 0000 0000 0000 0000 0000
0000ef0: 0000 0000 0000 0000 0000 0000 0000 0000
```

Figure 50 : HTML File Creation MD (HEX)

Results After manual Processing.

Table 12. HTML FILE CREATION INODE DATA STRUCTURE MD CATEGORY

Description	Offset	Number of Bytes	Processed Hexadecimal Bytes in Big endian	Decimal Output
Size in bytes	0x4	4	0000685f	26719 Bytes
File Access Time	0x08	4	5b27abe3	12:56:03 PM June 18, 2018
Inode Change Time	0xC	4	5b27abe3	12:56:03 PM June 18, 2018
File Modification Time	0x10	4	5b27a892	12:41:54 PM June 18, 2018
Number of Extents	0x2A	2	0001	1

Number of Blocks In extents	0x38	2	0007	7
Upper 16 Bits of Block Address	0x3A	2	0000	0
Lower 32 Bits of Block address	0x3C	4	000083a4	33700

This also confirms the sleuth kit output.

4.5.4 Content category:

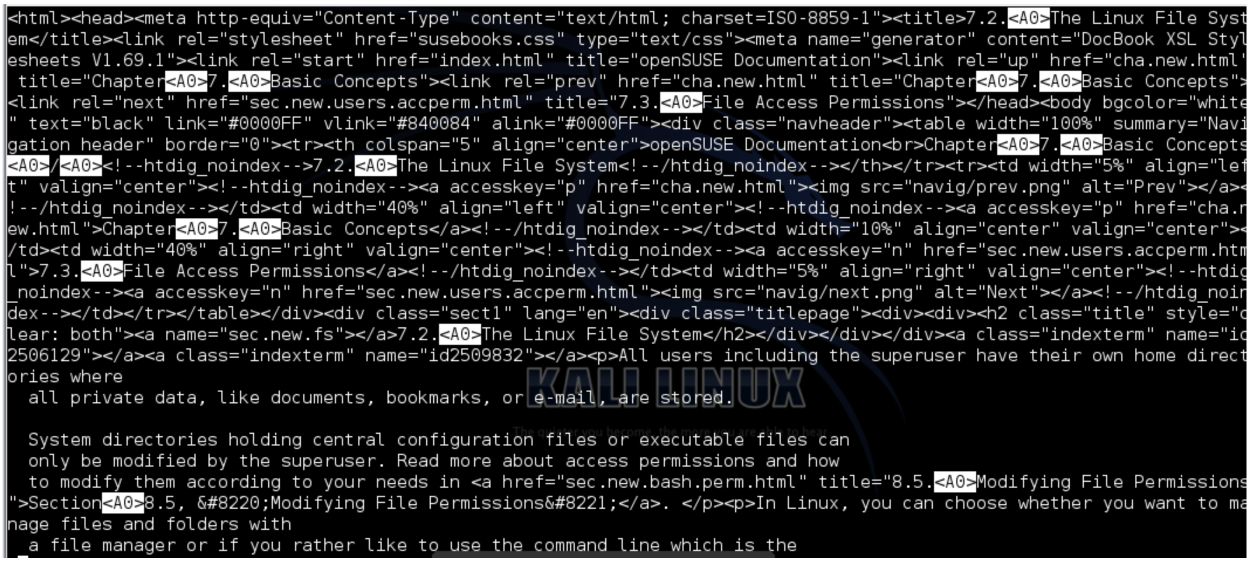
In this category we will check the allocation status of a block.

Figure 51 : HTML File Creation Content Category

Now we will verify which inode is allocated to this block.

Figure 52 : HTML File Creation (Inode)

Now we will process the content of this block.



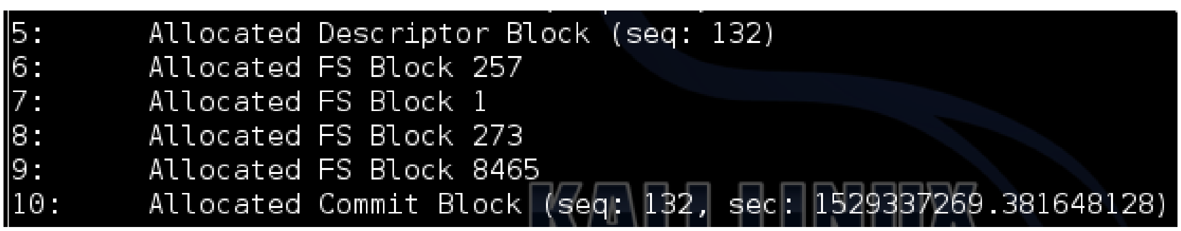
```
<html><head><meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1"><title>7.2.<A0>The Linux File Syst
em</title><link rel="stylesheet" href="susebooks.css" type="text/css"><meta name="generator" content="DocBook XSL Styl
esheets V1.69.1"><link rel="start" href="index.html" title="openSUSE Documentation"><link rel="up" href="cha.new.html"
 title="Chapter<A0>7.<A0>Basic Concepts"><link rel="prev" href="cha.new.html" title="Chapter<A0>7.<A0>Basic Concepts">
<link rel="next" href="sec.new.users.accperm.html" title="7.3.<A0>File Access Permissions"></head><body bgcolor="white
" text="black" link="#0000FF" vlink="#840084" alink="#0000FF"><div class="navheader"><table width="100%" summary="Navi
gation header" border="0"><tr><th colspan="5" align="center">openSUSE Documentation<br>Chapter<A0>7.<A0>Basic Concepts
<A0>/<A0><!--htdig_noindex-->7.2.<A0>The Linux File System<!--/htdig_noindex--></th></tr><tr><td width="5%" align="lef
t" valign="center"><!--htdig_noindex--><a accesskey="p" href="cha.new.html"><img src="navig/prev.png" alt="Prev"></a><
!--/htdig_noindex--></td><td width="40%" align="left" valign="center"><!--htdig_noindex--><a accesskey="p" href="cha.n
ew.html">Chapter<A0>7.<A0>Basic Concepts</a><!--/htdig_noindex--></td><td width="10%" align="center" valign="center"><
/td><td width="40%" align="right" valign="center"><!--htdig_noindex--><a accesskey="n" href="sec.new.users.accperm.htm
l">7.3.<A0>File Access Permissions</a><!--/htdig_noindex--></td><td width="5%" align="right" valign="center"><!--htdig
_noindex--><a accesskey="n" href="sec.new.users.accperm.html"><img src="navig/next.png" alt="Next"></a><!--/htdig_noin
dex--></td></tr></table></div><div class="sect1" lang="en"><div class="titlepage"><div><div><h2 class="title" style="c
lear: both"><a name="sec.new.fs"></a>7.2.<A0>The Linux File System</h2></div></div></div><a class="indexterm" name="id
2506129"></a><a class="indexterm" name="id2509832"></a><p>All users including the superuser have their own home direct
ories where
  all private data, like documents, bookmarks, or e-mail, are stored.

  System directories holding central configuration files or executable files can
  only be modified by the superuser. Read more about access permissions and how
  to modify them according to your needs in <a href="sec.new.bash.perm.html" title="8.5.<A0>Modifying File Permissions
">Section<A0>8.5, &#8220;Modifying File Permissions&#8221;</a>. </p><p>In Linux, you can choose whether you want to ma
nage files and folders with
  a file manager or if you rather like to use the command line which is the
```

Figure 53 : HTML File Creation Content Cat (HEX)

We can see that the content is in the data block which is associated with the file.

4.5.6 Application Category:

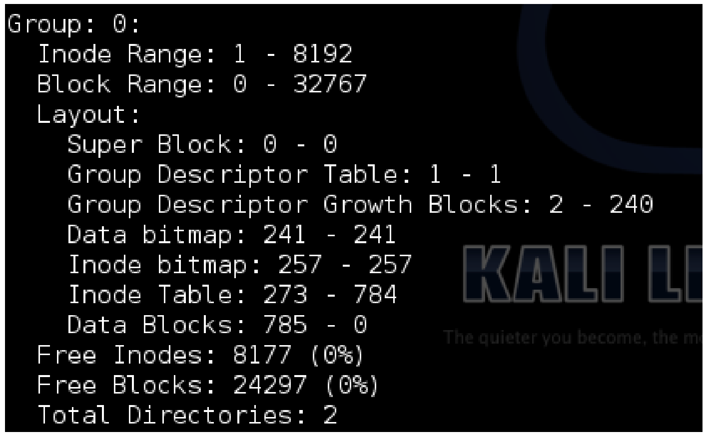
In this category the journaling is performed by the file system.



```
5:      Allocated Descriptor Block (seq: 132)
6:      Allocated FS Block 257
7:      Allocated FS Block 1
8:      Allocated FS Block 273
9:      Allocated FS Block 8465
10:     Allocated Commit Block (seq: 132, sec: 1529337269.381648128)
```

Figure 54 : HTML File Creation App Category

Analyzing the journal block after Html file creation it can be seen that the there are changes in the blocks associated with the file. Block 257 belongs to block group 0. Block 257 is the inode bitmap in which the change has occurred. An inode has changed due to which a change in block 273 has occurred. Similarly there is a change in the Group descriptor table which is block 1 in Block group 0.



```
Group: 0:
  Inode Range: 1 - 8192
  Block Range: 0 - 32767
  Layout:
    Super Block: 0 - 0
    Group Descriptor Table: 1 - 1
    Group Descriptor Growth Blocks: 2 - 240
    Data bitmap: 241 - 241
    Inode bitmap: 257 - 257
    Inode Table: 273 - 784
    Data Blocks: 785 - 0
  Free Inodes: 8177 (0%)
  Free Blocks: 24297 (0%)
  Total Directories: 2
```

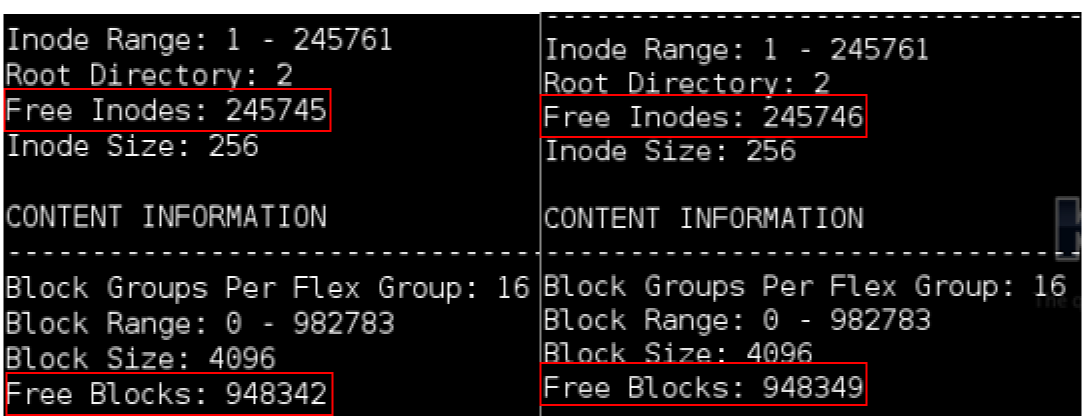
Figure 55 : HTML File Creation App Category

4.6 Scenario (HTML File Deletion):

Now we will analyze HTML file deletion in all 5 different categories

4.6.1 File System Category:

First of all we will see changes in the file system category .We can see that the number of free blocks and number of free inodes has changed after we have deleted the html file



```
Inode Range: 1 - 245761
Root Directory: 2
Free Inodes: 245745
Inode Size: 256

CONTENT INFORMATION
--------------------------------
Block Groups Per Flex Group: 16
Block Range: 0 - 982783
Block Size: 4096
Free Blocks: 948342
```

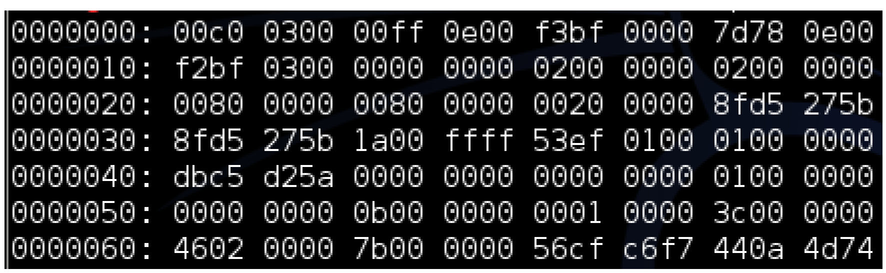
```
--------------------------------
Inode Range: 1 - 245761
Root Directory: 2
Free Inodes: 245746
Inode Size: 256

CONTENT INFORMATION
--------------------------------
Block Groups Per Flex Group: 16
Block Range: 0 - 982783
Block Size: 4096
Free Blocks: 948349
```

Figure 56 : HTML File Detion FS Category

We can see that the number of free inodes has increased from 245745 to 245746.We can also see that the number of free block count has also increased from 948342 to 948346.This can also be verified from the Hex output of the super block.

Table 13. HTML FILE DELETION SUPER BLOCK DATA STRUCTURE FS CATEGORY

Description	Offset	Number of Bytes
Total Inodes Count	0x0	4
Total Block Count	0x4	4
Free Block Count	0xC	4
Free Inode Count	0x10	4
Magic Number	0x38	2



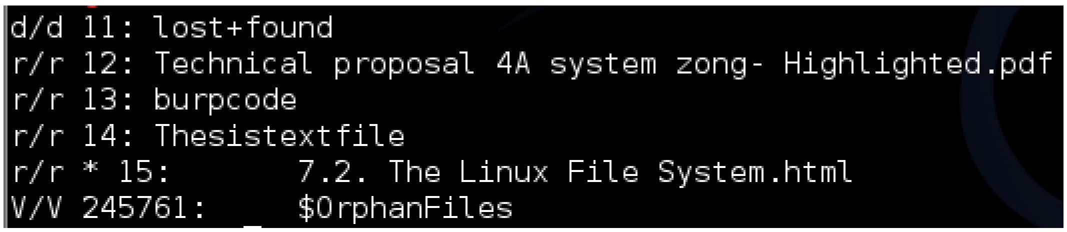
```
0000000: 00c0 0300 00ff 0e00 f3bf 0000 7d78 0e00
0000010: f2bf 0300 0000 0000 0200 0000 0200 0000
0000020: 0080 0000 0080 0000 0020 0000 8fd5 275b
0000030: 8fd5 275b 1a00 ffff 53ef 0100 0100 0000
0000040: dbc5 d25a 0000 0000 0000 0000 0100 0000
0000050: 0000 0000 0b00 0000 0001 0000 3c00 0000
0000060: 4602 0000 7b00 0000 56cf c6f7 440a 4d74
```

Figure 57 : HTML File Deletion FS Category (Hex)

As the free block count is at offset 0xC which is 4 bytes in size, we can see that the number of free blocks in hexadecimal is 000e787d which is equal to 948346 which is equivalent to number of free blocks as mentioned in sleuth kit output. Similarly we can also see that the number of free inodes start at 0x10 which is 4 bytes in size and is 0003bff2 in hexadecimal and is 245746 in decimal , this confirms the sleuth kit output.

4.6.2 Filename Category:

Analyzing the filename category after creation of the Html file we get the following output.



```
d/d 11: lost+found
r/r 12: Technical proposal 4A system zong- Highlighted.pdf
r/r 13: burpcode
r/r 14: Thesistextfile
r/r * 15:       7.2. The Linux File System.html
V/V 245761:     $OrphanFiles
```

Figure 58 : HTML File Deletion FN Category

Although the file is deleted but it is still visible along with its inode as a deleted file.

4.6.3 Metadata Category:

To analyze the file creation in the metadata category we process the inode number in the sleuthkit and see what metadata information of the file we can get.

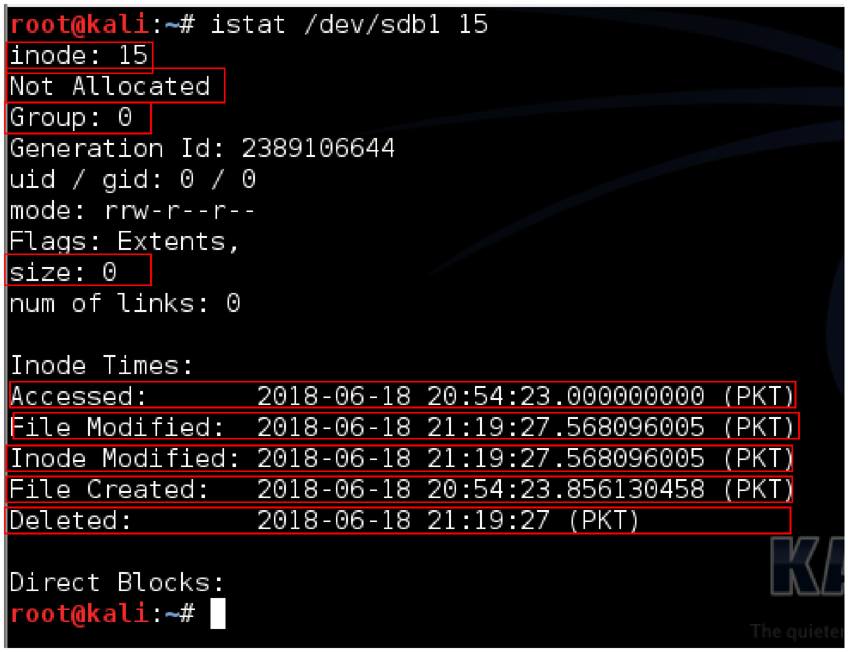


Figure 59 : HTML File Deletion MD Category

Inode for this file is also found in block group 0. The size of the file has reduced to 0 bytes. Times associated with this inode along with blocks are also shown in the above screenshot. Inode range in this group is from 1-8192. As the inode size is 256 bytes and the block size is 4096 bytes, there are 4096/256 bytes inodes per block in this group. The inode table in the group starts from block 273. As there are 16 inodes per block, inode 15 can be found in block 273. Processing block 273 and moving to 15th 256 byte structure in this block we have the following Hex output.

```
0000e00: a481 0000 0000 0000 afd5 275b 8fdb 275b
0000e10: 8fdb 275b 8fdb 275b 0000 0000 0000 0000
0000e20: 0000 0800 0100 0000 0af3 0000 0400 0000
0000e30: 0000 0000 0000 0000 0000 0000 0000 0000
0000e40: 0000 0000 0000 0000 0000 0000 0000 0000
0000e50: 0000 0000 0000 0000 0000 0000 0000 0000
0000e60: 0000 0000 d4df 668e 0000 0000 0000 0000
0000e70: 0000 0000 0000 0000 0000 0000 0000 0000
0000e80: 1c00 0000 14d4 7187 14d4 7187 0000 0000
0000e90: afd5 275b 680e 1ecc 0000 0000 0000 0000
0000ea0: 0000 0000 0000 0000 0000 0000 0000 0000
0000eb0: 0000 0000 0000 0000 0000 0000 0000 0000
0000ec0: 0000 0000 0000 0000 0000 0000 0000 0000
0000ed0: 0000 0000 0000 0000 0000 0000 0000 0000
0000ee0: 0000 0000 0000 0000 0000 0000 0000 0000
0000ef0: 0000 0000 0000 0000 0000 0000 0000 0000
```

Figure 60 : HTML File Deletion MD Category (Hex)

Results After manual Processing.

Table 14. HTML FILE DELETION INODE DATA STRUCTURE MD CATEGORY

Description	Offset	Number of Bytes	Processed Hexadecimal Bytes in Big endian	Decimal Output
Size in bytes	0x4	4	0	0
File Access Time	0x08	4	5b27d5af	8:54:23 PM June 18 ,2018
Inode Change Time	0xC	4	5b27db8f	9:19:27 PM June 18, 2018
File Modification Time	0x10	4	5b27db8f	9:19:27 PM June 18, 2018
Deletion Time	0x14	4	5b27db8f	9:19:27 PM June 18, 2018
File creation Time	0x90	4	5b27d5af	8:54:23 PM June 18 ,2018
Number of Extents	0x2A	2	0000	0

Number of Blocks In extents	0x38	2	0000	0
Upper 16 Bits of Block Address	0x3A	2	0000	0
Lower 32 Bits of Block address	0x3C	4	00000000	0

This also confirms the sleuth kit output.

4.6.5 Content Category:

In this category we will check the allocation status of a block.

```
root@kali:~# blkstat /dev/sdb1 33700
Fragment: 33700
Not Allocated
Group: 1
```

Figure 61 : HTML File Deletion Content Cat

Now we will verify which inode is allocated to this block.

```
root@kali:~# ifind /dev/sdb1 -d 33700
Inode not found
```

Figure 62 : HTML File Deletion CC (Inode)

Now processing the content of the block

```
root@kali:~# blkcat /dev/sdb1 33700
<html><head><meta http-equiv="Content-Type" content="text/html; charset=ISO-8859
-1"><title>7.2.�The Linux File System</title><link rel="stylesheet" href="susebo
oks.css" type="text/css"><meta name="generator" content="DocBook XSL Stylesheets
 V1.69.1"><link rel="start" href="index.html" title="openSUSE Documentation"><li
nk rel="up" href="cha.new.html" title="Chapter�7.�Basic Concepts"><link rel="pre
v" href="cha.new.html" title="Chapter�7.�Basic Concepts"><link rel="next" href="
sec.new.users.accperm.html" title="7.3.�File Access Permissions"></head><body bg
color="white" text="black" link="#0000FF" vlink="#840084" alink="#0000FF"><div c
lass="navheader"><table width="100%" summary="Navigation header" border="0"><tr>
<th colspan="5" align="center">openSUSE Documentation<br>Chapter�7.�Basic Concep
ts�/�<!--htdig_noindex-->7.2.�The Linux File System<!--/htdig_noindex--></th></t
r><tr><td width="5%" align="left" valign="center"><!--htdig_noindex--><a accessk
ey="p" href="cha.new.html"><img src="navig/prev.png" alt="Prev"></a><!--/htdig_n
oindex--></td><td width="40%" align="left" valign="center"><!--htdig_noindex--><
a accesskey="p" href="cha.new.html">Chapter�7.�Basic Concepts</a><!--/htdig_noin
dex--></td><td width="10%" align="center" valign="center"></td><td width="40%" a
lign="right" valign="center"><!--htdig_noindex--><a accesskey="n" href="sec.new.
users.accperm.html">7.3.�File Access Permissions</a><!--/htdig_noindex--></td><t
d width="5%" align="right" valign="center"><!--htdig_noindex--><a accesskey="n"
href="sec.new.users.accperm.html"><img src="navig/next.png" alt="Next"></a><!--/
htdig_noindex--></td></tr></table></div><div class="sect1" lang="en"><div class=
```

Figure 63: HTML File Deletion CC (HEX)

Now we can see that the content of the block is still there and is not deleted.

4.6.6 Application Category:

We will analyze the journaling part of the file system.

```
Allocated Descriptor Block (seq: 137)
Allocated FS Block 8465
Allocated FS Block 273
Allocated FS Block 0
Allocated FS Block 242
Allocated FS Block 1
Allocated FS Block 257
Allocated Commit Block (seq: 137, sec: 1529343302.3365939200)
```

Figure 64 : HTML File Deletion App category

Analyzing the journal block after Html file deletion it can be seen that the there are changes in the blocks associated with the file. Block 257 belongs to block group 0. Block 257 is the inode bitmap in which the change has occurred. An inode has changed due to which a change in block 273 has occurred. Similarly there is a change in the Group descriptor table which is block 1 in Block group 0.

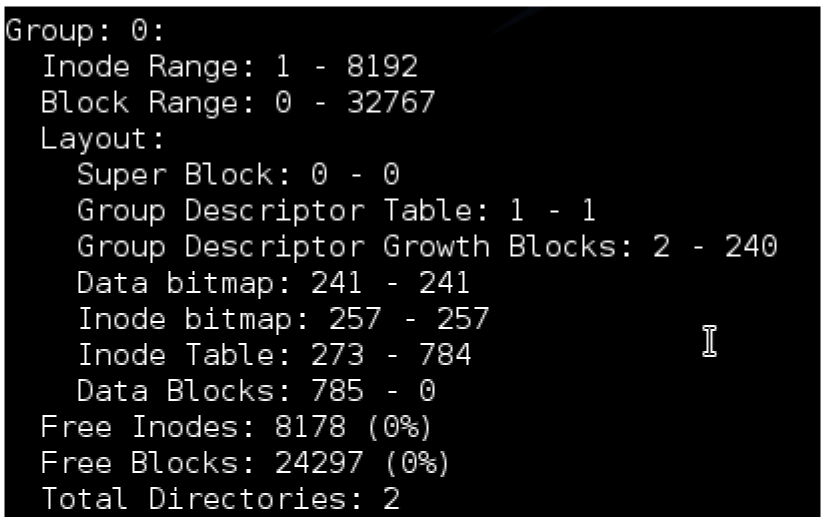


Figure 65 : HTML File Deletion App Category

4.6.7 HTML File Deletion Summary

Table 15. HTML FILE DELETION SUMMARY

Categories	Data Persistence
File System	×

File Name	✓
Meta Data	✓
Content	✓
Application	✓

Chapter 5

DEFINING ANTI FORENSIC STANDARDS FOR EXT4 FILE SYSTEM

5.1 Comparison of EXT4 with EXT3

The main difference between EXT4 and EXT3 is the difference in structure of how they point to blocks .EXT3 uses indirect pointers to point to blocks while ext4 uses Extents instead.

5.1.1 Indirect Pointers EXT3

To avoid making the inode structure too large while supporting large file sizes concept of indirect pointers is used .The most simple is the single indirect pointer and is stored in the inode structure . These pointers point directly to the file data. The double indirect pointer points to the repository for single indirect pointer. The triple indirect pointer points to the file system block which acts as a repository for double indirect pointers. Following is the diagram showing how these are related.

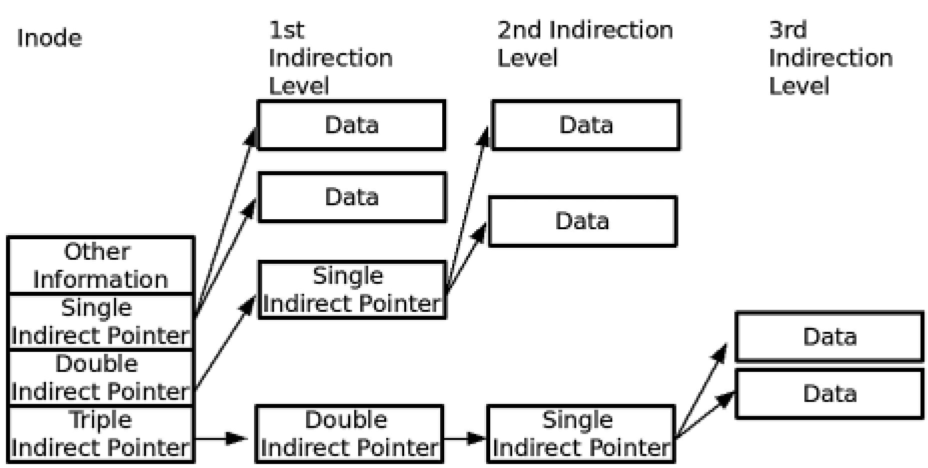


Figure 66: Illustration of Indirect Block Pointers

5.1.2 Extents

Extents were introduced in EXT4 to reduce the amount of metadata needed to keep track of the data blocks for large files. Instead of storing a list of every individual block which makes up the file, the idea is to store just the address of the first and last block of each continuous range of blocks. These continuous ranges of data blocks (and the pairs of numbers which represent them) are called extents.

5.2 Results Summary

In this chapter the results of research conducted is summarized and the Three different types of files have been created and deleted in ext4 file system to observe the changes in all different categories as mentioned by Brian carrier to look at the data persistence after the file is deleted. Results of forensic examination after file deletion are shown in the following table**.**

Table 16. DATA PERSISTENCE

		File Types		
		PDF	TEXT	HTML
Cate-	File System	×	×	×

	File Name	✓	✓	✓
	Meta Data	✓	✓	✓
	Content	✓	✓	✓
	Application	✓	✓	✓

5.3 Defining File Deletion Standards

When a file is deleted the following categories must be deleted to ensure that the file is completely wiped from the file system and no data associated with the file is recoverable.

Categories	Deletion Required
File System	No
File name	Yes
Meta Data	Yes
Content	Yes
Application	Yes

Chapter 6

CONCLUSION AND FUTURE WORK

Lack of standards from a recognized authorities like NIST for securely wiping data at the file system level (EXT4) created the need to work on this research area.

The different types of file were created and deleted from the ext4 file system and changes were observed in all different categories for the data persistence. It was observed that almost all categories associated with the file system has some data even after deletion. This research thesis is very important for people who want to securely wipe data associated with a file in ext4 file systems. This thesis has provided the standards for securely wiping data of a file in EXT4 file system.

Following future work can be taken as the continuation of this Thesis

- Tweaking with the default file system features of EXT4 and observe the changes
- Behavior of EXT4 with different Operating systems

Bibliography

[1] NIST. Computer Forensics Tool Testing (CFTT) Program. [Online]. http://www.cftt.nist.gov/documents.htm

[2] T. Tso and S. Tweedie. "Planned Extensions to the Linux Ext2/3 Filesystem," USENIX Technical Conference, Montery CA. 2002

[3] Amelia Phillips and Christopher Steuart Bill Nelson, Guide to Computer Forensics and Investigations, 4th ed. Boston, MA 02210, USA: Course Technology, 2010.

[4]Carrier, B. The sleuth kit, TSK. http://www.sleuthkit.org/sleuthkit/ (Online).

[5] B. Carrier. File System Forensic Analysis. Upper Saddle River, NJ: Pearson Education, Inc. 2005.

[6]Ext4 disk layout, 2016. (Accessed 26 December 2016) URL https://ext4.wiki.kernel. org/index.php/Ext4_Disk_Layout.

[7]Fairbanks, K.D.,2012. An Analysis of ext4 for digital forensics. Digit. Investig. 9, S118eS130.

[8] Craiger, P., 2005. Recovering digital evidence from linux systems. In: Advances in Digital Forensics. Springer, pp. 233e244.

[9] T. Tso. e2fsprogs. http://e2fsprogs.sourceforge.net/

[10] A. Mathur, M. Cao, S. Bhattacharya, A. Dilger, A. Tomas, and L. Vivier. "The new ext4 filesystem: current status and future plans," Linux Symposium. 2007.

[11] Daniel Phillips. A directory index for ext2. In 5th Annual Linux Showcase and Conference, pages 173–182, 2001.

[12] Baier H, Breitinger F. Security aspects of piecewise hashing in computer forensics. In: IT Security Incident Management & IT Forensics (IMF); 2011. pp. 21–36.

[13] Breitinger F, Stivaktakis G, Roussev V. Evaluating detection error tradeoffs for Bitwise approximate matching algorithms. In: 5th ICST Conference on Digital Forensics & Cyber Crime (ICDF2C); 2013.

[14] Ding Liping, Wang Yongji. Study on Multi-Dimension Computer Forensics Model. Network & Computer Security. 2005. Vol 11.
[15] Li Weiwei. Computer Forensics Analysis based on EnCase System. Jilin Normal University Journal. 2011. Vol 32.

[16] Bovet, Daniel P., and Cesati, Marco. Understanding the Linux Kernel (3rd Edition). O'Reilly Media, 1 November 2005.

[17] B. Grundy, The Law Enforcement and Forensic Examiner Introduction to Linux: A Beginner's Guide, January 2004.

[18] B. Carrier, Defining Digital Forensic Examination and Analysis Tools Using Abstraction Layers, International Journal of Digital Evidence, Vol. 1 Issue 4, Winter 2003.

[19] Baier H, Breitinger F. Security aspects of piecewise hashing in computer forensics. In: IT Security Incident Management & IT Forensics (IMF); 2011. pp. 21–36.