ALBERT-LUDWIGS-UNIVERSITÄT FREIBURG

Amir Alsbih and Christian Schindelhauer

Distributed Storage and

# Computer Forensic

# Amir Alsbih

- □ Chief Information Security Officer at the Haufe Group
- □ E-Mail: alsbiha@informatik.uni-freiburg.de
- □ Web 1.0:
  https://www.xing.com/profile/Amir_Alsbih
  http://www.linkedin.com/profile/view?id=66146035
  https://plus.google.com/111842188330803310837
- □ Web 2.0:
  https://twitter.com/#!/checkm4te

# Introduction

# What we will cover

- □ Fundamentials
- □ Linux Forensics
- □ Windows Forensics
- □ What the time let us perform

# Methodology

- Verification that an incident has taken place
- Environment description
  - For what are the systems used?
  - What type of OS, Hardware,… is use
- Evidence acquisition: 1:1 Copies
- Timeline Analysis: What has happened when
- Media Analysis
- String / Byte Search
- Data Recovery
- Reporting Results

# Forensic principles

- Minimize data loss: Computer evidence is subject to the "observer effect"
- Take notes about everything
- Analyze all data collected (start with the most volatile data)
- Whenever possible, rely on more than one tool
- Understand what you do and why

# Two scenarios

- Dead System
  - Power unplugged
  - System off
  - Hard Drive
  - Floppy Disks
  - CD-Roms

- Live System
  - Power On
  - Processes Running
  - Disks being accessed
  - Removable Media changing

**Memory, Network status and connections and processes runing are destroyed if power is cut**

ALBERT-LUDWIGS-
UNIVERSITÄT FREIBURG

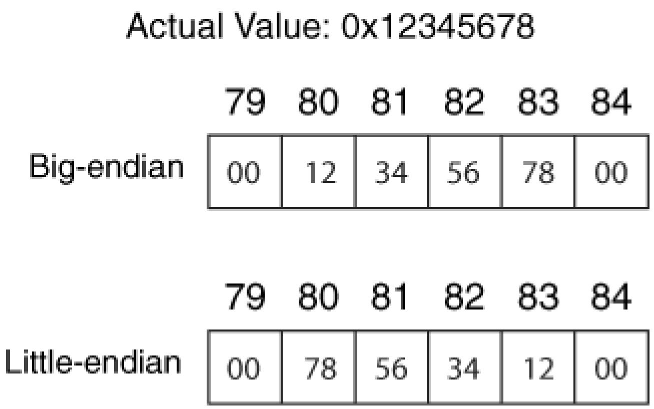
# Incident Response

1. Gather network connections
2. Unplug network
3. Gather volatile data
    1. Processes
    2. System Memory
4. Verify Inicdent
    - Logs, IDS, interviews
5. Gather evidence (make Images)

# File System Essentials

# Numbers

- Decimal
  - Base 10 (0-9)
- Hexadecimal
  - Base 16 (0-9, a-f)
- Binary
  - Base 2 (0,1)



Actual Value: 0x12345678

|  | 79 | 80 | 81 | 82 | 83 | 84 |
|---|---|---|---|---|---|---|
| Big-endian | 00 | 12 | 34 | 56 | 78 | 00 |

|  | 79 | 80 | 81 | 82 | 83 | 84 |
|---|---|---|---|---|---|---|
| Little-endian | 00 | 78 | 56 | 34 | 12 | 00 |

# Example

- Twenty Nine
- Decimal:
  - 29 (9+10+10)
- Hexadecimal:
  - 1D (13+16)
- Binary:
  - 11101 (2^0+2^2+2^3+2^4)

# Data Organization

- File System typically use 512-byte sectors
- For efficiency this sectors are organized in Blocks / Clusters that contains 1-N consecutive sectors
- Physical Layer
  - The drive
- File System Layer
  - Partition Information
- Data Layer / Content
  - The data (Clusters/Blocks)
- Metadata Layer
  - Strcuture Information (FAT, NTFS, RaiserFS, EXT2/3,…)
- File Name Layer / Application
  - Name of the File

# File System Layer

- Describes the file system structural details:
  - Data unit sizes,
  - Structural offsets
  - Mouting informations
  - Stored in the „boot sector / superblock“

# Data Layer

- Allocated
  - Data block is actively being used by a file
  - Data exists in a file on the system
  - Not deleted
- Unallocated
  - Data block is not being used by a file
  - Data may or may not exists in the block
  - May contain deleted or unused data
  - Pieces of files are called „file fragments“

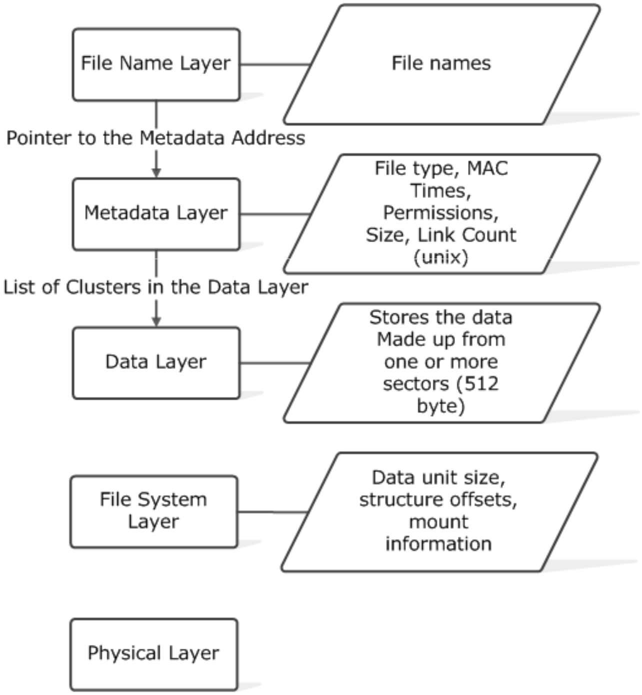
# Metadata Layer

- □ The Metadata layer contains the structures and values that decribe a file
- □ Like a Card Catalog in a libery
- □ Contains pointers to the data layer and informations such as MACtimes, permissions and size
- □ Each metatada strcture is given an addres

# File Name Layer

- File Names are stored in:
  - File Metadata in Windows
  - Directory File in Unix
- Filenames points to the Metadata Address

# File System Layers


File Name Layer
File names
Pointer to the Metadata Address
Metadata Layer
File type, MAC Times, Permissions, Size, Link Count (unix)
List of Clusters in the Data Layer
Data Layer
Stores the data Made up from one or more sectors (512 byte)
File System Layer
Data unit size, structure offsets, mount information
Physical Layer

# File System Layer

# X86-Based Systems

- □ (0x00) 0 - 445: Boot Code
- □ (0x1BE) 446 – 461: Partition Table #1
- □ (0x1CE) 462 – 477: Partition Table #2
- □ (0x1DE) 478 – 493: Partition Table #3
- □ (0x1EE) 494 – 509: Partition Table #4
- □ (0x1FE) 510 – 511: 0x55AA (End of MBR)

# Partition Table Entry

| Offset | Lenth (bytes) | Content |
|---|---|---|
| 0 | 1 | State of  partition:<br>- 00H inactive<br>- 80H active |
| 1 | 3 | Partition start head (CHS): 8-bit head value, a 6-bit sector value, and a 10-bit cylinder value |
| 4 | 1 | Partition Type |
| 5 | 3 | Partion end head (CHS): ): 8-bit head value, a 6-bit sector value, and a 10-bit cylinder value |
| 8 | 4 | Starting LBA Address |
| 12 | 4 | Size in Sectors |

# Common Types of Partition

| Hex Value | Type |
|---|---|
| 0x01 | FAT 12 |
| 0x0E | FAT 16 |
| 0x0C | FAT 32 |
| 0x83 | Linux Native |
| 0x82 | Linux Swap |
| 0x05 | Extended |
| 0x07 | NTFS |
| 0xa5 | FreeBSD |
| 0xa6 | OpenBSD |
| 0xa8 | Mac OSX |
| 0xfb | Vmware File System |

# MBR Example

```
0000000: eb48 9010 8ed0 bc00 b0b8 0000 8ed8 8ec0
0000384: 0048 6172 6420 4469 736b 0052 6561 6400
0000400: 2045 7272 6f72 00bb 0100 b40e cd10 ac3c
0000416: 0075 f4c3 0000 0000 0000 0000 0000 0000
0000432: 0000 0000 0000 0000 0000 0000 0000 0001
0000448: 0100 07fe 3f7f 3f00 0000 4160 1f00 8000
0000464: 0180 83fe 3f8c 8060 1f00 cd2f 0300 0000
0000480: 018d 83fe 3fcc 4d90 2200 40b0 0f00 0000
0000496: 01cd 05fe ffff 8d40 3200 79eb 9604 55aa.
```

# Solution

| # | Flag | Type | Starting Sector | Size |
|---|---|---|---|---|
| 1 | 0x00 | 0x07 | 0x0000003f (63) | 0x001f6041 (2,056,257) |
| 2 | 0x80 | 0x83 | 0x001f6080 (2,056,320) | 0x00032fcd (208,845) |
| 3 | 0x00 | 0x83 | 0x0022904d (2,265,165) | 0x000fb040 (1,028,160) |
| 4 | 0x00 | 0x05 | 0x0032408d (3,293,325) | 0x0496eb79 (76,999,545) |

# Using Tools

fdisk -lu:

- □ Device
- □ Boot flag
- □ Start
- □ End
- □ #Blocks (Clusters*BS)
- □ Id
- □ File System type
- □ Not all partitions are shown like linux swap!

mls

- □ Slot
- □ Start
- □ End
- □ Length (in Clusters)
- □ Description

# Using Tools (2)

- mmls from the Sleuth Kit :

  Slot Start End Length Description Device

  - Boot flag
  - Start
  - End
  - #Blocks
  - Id
  - File System type
  - Not all partitions are shown like linux swap!

# Data Layer

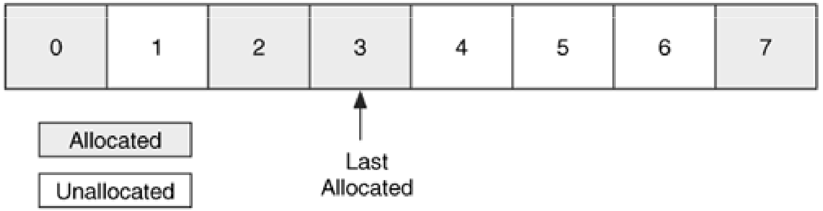
# Background

□ When a file is splitted over the file system in no consecutive blocks than it is “fragmented”.

□ Allocated blocks that do not have a corresponding metadata structure are called orphan blocks

# Allocation Strategies

- □ first available
- □ next available
- □ best fit



*Source:File System Forensic Analysis*

# Allocation Strategies

- □ first available: searches for an available block starting with the first block in the file system (for every data block).
- □ next available: searches for an available block starting with the block that was allocated last.
- □ best fit: searches for consecutive blocks that fit the needed amount of data. The whole file will moved –if possible- when data size increases, and there is not enough free blocks at the end of the file.

# Data-Layer Tools

- dcat: displays the raw or hex content of an allocated block
- dls: extract unallocated data to a file
- dcalc: calculates the orginal offset of an unallocated block in the allocation bitmap (1 = allocated, 0 = unallocated)

# Metadata Layer

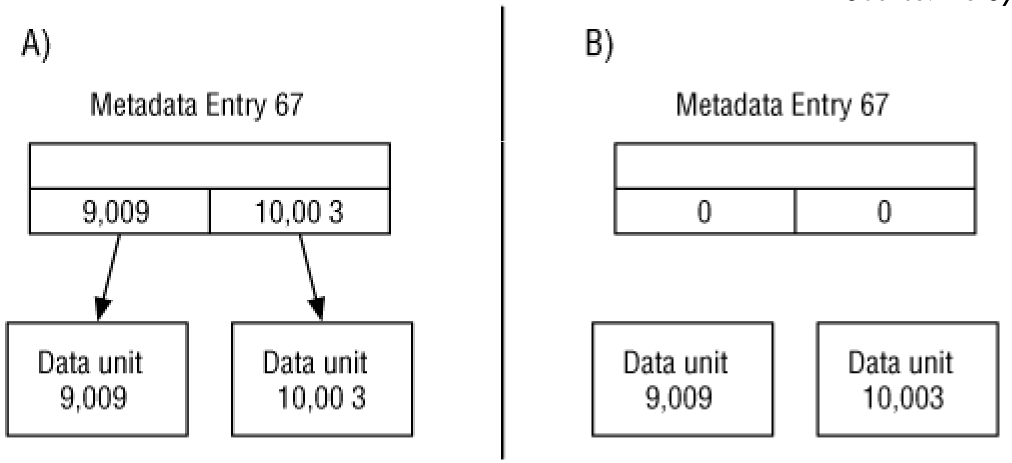
- □ Logical File Address: A data unit that is allocated to a file also has a logical file address.
- □ A data unit's logical file address is relative to the start of the file to which it is allocated.

# Slack Space

- Slack Space: When the size of a file is not a multiple of the block size the file will have “slack space”.
- There are two places for slack space:
  - between the end of the file and the end of the sector in which the file ends. (typically this days filled with 0x00)
  - The sectors after the end of the file to the end of the block. (can contain older data)

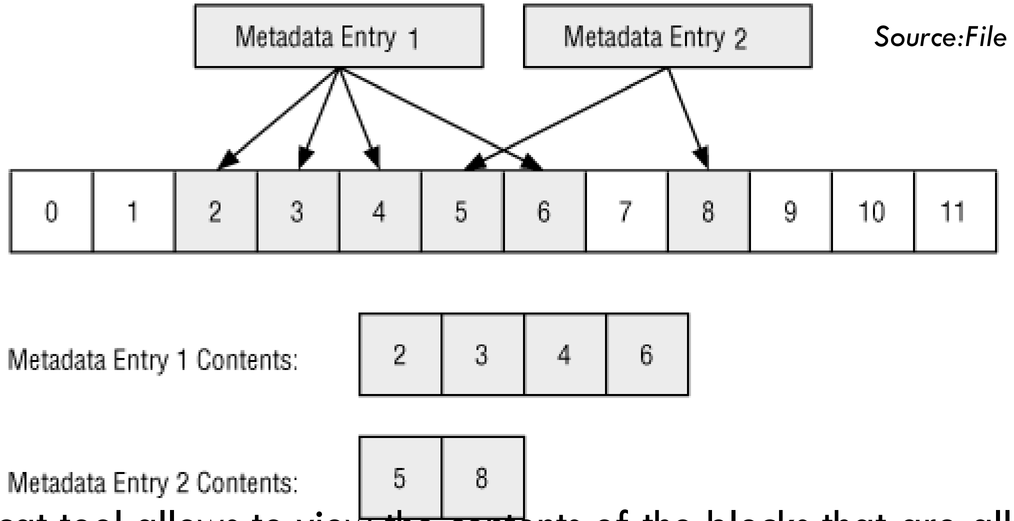
# Two scenarios of unallocated data

*Source:File System Forensic Analysis*



(A) the block pointers are not wiped when the entry is unallocated and in (B) they are wiped

# metadata lookup technique

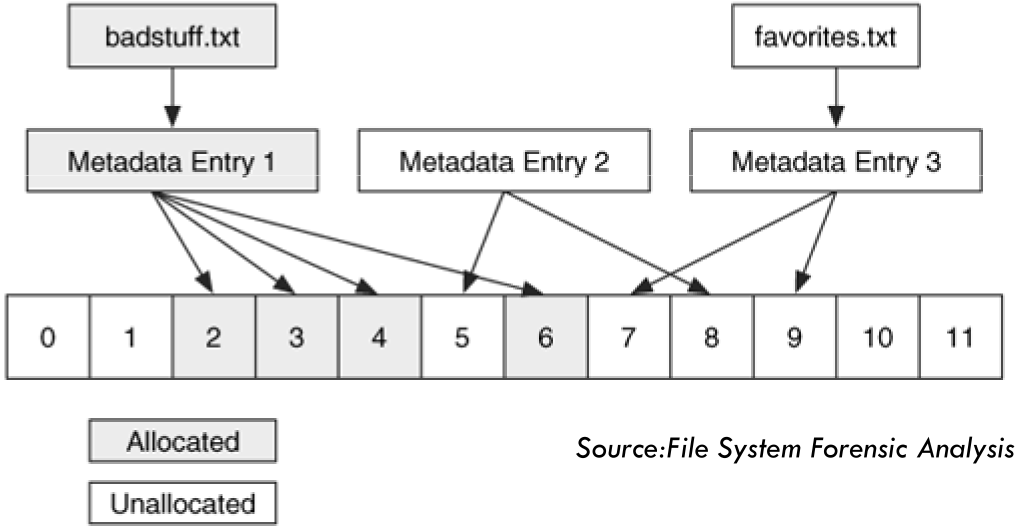


*Source:File System Forensic Analysis*

- icat tool allows to view the contents of the blocks that are allocated to a metadata structure

# Filename Layer

# recover files by their unallocated name



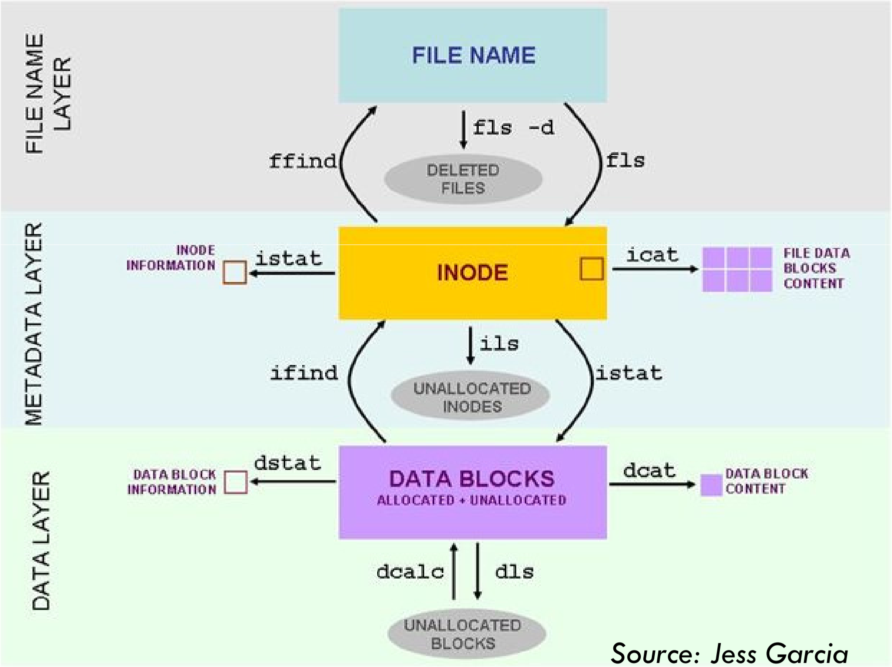
*Source:File System Forensic Analysis*

# Journals

- □ If the OS was writing data to the disk or if it was waiting to write some data to disk when the crash occurred, the file system could be in an inconsistent state. There could be an allocated metadata structure with allocated data units, but no pointers between them and no file name pointing to the metadata structure.
- □ Before any metadata changes are made to the file system, an entry is made in the journal that describes the changes that will occur. After the changes are made, another entry is made in the journal to show that the changes occurred.
- □ If the system crashes, the scanning program reads the journal and locates the entries that were not completed. The program then either completes the changes or rolls them back to the original state.

# Tool Overview

ALBERT-LUDWIGS-
UNIVERSITÄT FREIBURG



*Source: Jess Garcia*

# When nothing helps

Data-Carving

# Data Carving (foremost)

- Data carving is a process where a chunk of data is searched for signatures that correspond to the start and end of known file types.
- An example tool that performs this is foremost (http://foremost.sourceforge.net).
  foremost analyzes a raw file system or disk image based on the contents of a configuration file, which has an entry for each signature.

```
jpg y 200000 \xff\xd8 \xff\xd9
```

# References

- File System Forensic Analysis
- SANS Institute
- Hacking Exposed Computer Forensics
- Jess Garcia's Website