Lab 4:

ext file systems

Forensic Analysis 2023-2024

© Daan Pareit

## Introduction

# Lab Concept

This lab is about understanding ext file systems with its directory entries, inodes and data blocks. It has a very simple goal: capture the flags (CTF). You are handed over an external disk, which you will examine. Files might be hidden, files might be deleted … on purpose. Recover data and find the flags.

# Practicalities

We will use Kali Linux as user environment for analyzing the disk (although any Linux distro could do the job). Use the Kali machine from the first lab.

We will work with an additional Virtual Disk (vmdk) which contains (deleted/hidden) data (“the suspect’s disk”). You can find the vmdk at:

* OneDrive: <https://studenthowest-my.sharepoint.com/:f:/g/personal/daan_pareit_howest_be/ErrJ8sYav4tPjpJW8H3kYBoB_b0u1Yqdo96O1Wv8ajehAg>

On kali, we’ll use tools from the Sleuthkit package. You should already have this installed. If not, you can find the sleuthkit software here: <http://sleuthkit.org/sleuthkit/download.php>

# Learning goals

Knowledge

* File system structure of ext2, ext3, ext4
* inodes
* Directory entries

Skills

* Perform file recovery on ext2 file systems using inodes
* Find file names of deleted files on ext3/ext4 file systems using inodes
* Use these tools properly: fls, istat, icat, blkcat, file, dumpe2fs, fsstat, xxd, dd

## Lab assignments

### Handle with care – acquisition phase

### The usual good practice

As in the previous lab with Windows file systems, you should treat the virtual disk (vmdk) you downloaded as a hard disk from a suspect and should thus handle it forensically correctly. You might want to have a look at *the section “Handle with care – Acquisition phase of external disks” of the first lab*, about how you did this exactly. “Repetition is the mother of learning”, says the proverb.

A reminder of the steps you’ll have to take:

1. Stop/mask the udisks2 service (to prevent automounting).

Open 2 terminals – udevadm monitor + udisksctl monitor

systemctl stop udisks2.service

systemctl mask udisks2.service

1. Physically add the disk (the extra vmdk) to your Kali (adding SCSI disks is hot swappable).
2. Rescan the SCSI bus to discover the new disk.

echo "- - -" > /sys/class/scsi\_host/host2/scan (as root)

1. (optional: read disk metadata with udevadm/hdparm)
2. Create an MD5 hash for the suspect’s disk with md5sum.

sudo md5sum /dev/sdb

a28124d298fe380d6dc2f8b04935bc69 /dev/sdb

1. Make a backup image of *the full disk* using dd (e.g. sdb-backup.img).

sudo dd if=/dev/sdb of=sdb-backup.img status=progress

1. Create an MD5 hash for your backup. Verify that both suspect’s disk and the backup are the same hash.

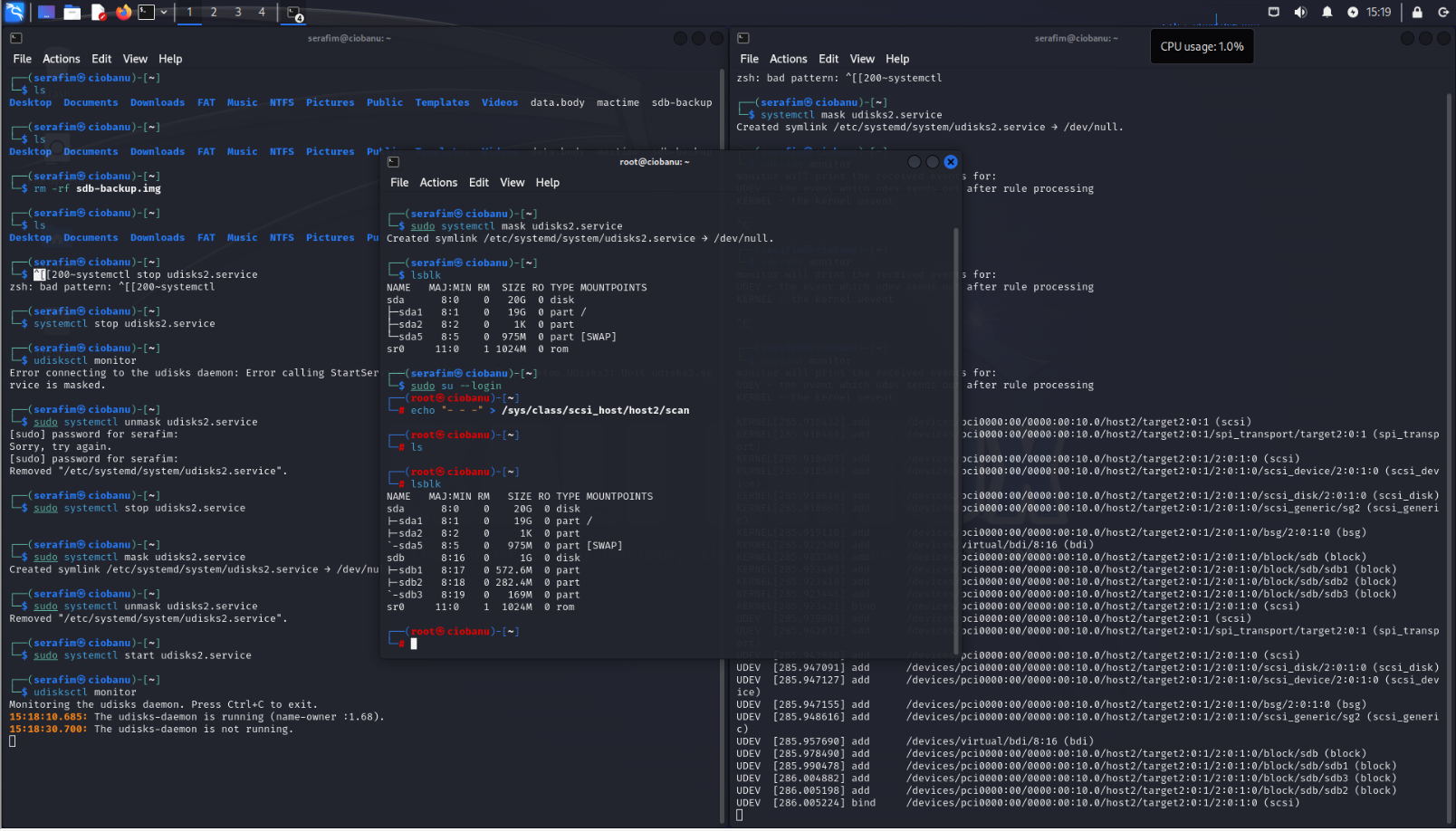


sudo md5sum /dev/sdb sdb-backup.img

1. Disconnect the disk in linux from your device list.

echo 1 > /sys/block/sdb/device/delete (as root)

1. Physically detach the disk (the extra vmdk) from your Kali.
2. (optional: restart udisks2 service)



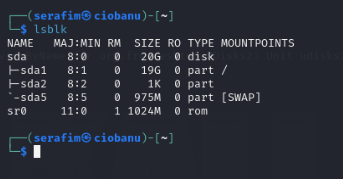
You should not mount the disk itself, because you could alter its data. For the next sections you will always need to perform your investigation on your forensic backup. ***You thus shouldn’t use /dev/sdb in any of the following steps (as it should be disconnected with the steps above) but use your image file instead!***

And again, you can work on the backup image directly (with offsets to access the partitions therein) or create a loopback device to work with (losetup -fPr *imagefile* --show).   
Note: with losetup -l, you can then see which files are made available as loopback block devices.

# Analyze your own ‘kali’ disk

Before inspecting the suspect’s disk, let’s play around with different tools and have a look at the file system of the partition your kali is installed on:

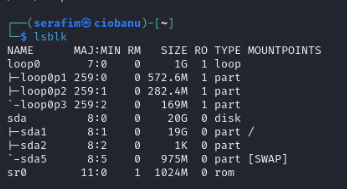
* Use the lsblk tool to find the known block devices by kali, as you have done in previous labs.



* Based on the theory slides you should know also understand the “MAJ:MIN” column in the output of lsblk.
  + What is the major and minor of the partition where kali is installed on?

MAJ = 8; MIN = 1 (meaning partition 1)

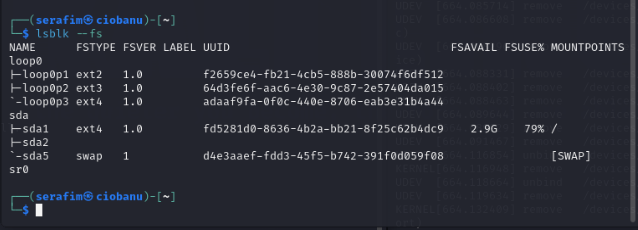
* + What is the major of the loopback device on which you made the image of the suspect’s disk available? And the major of the partitions within the loopback device? Have a look at <https://www.kernel.org/doc/Documentation/admin-guide/devices.txt> to find out the purpose of this major number.



MAJ = 7 for the loopback device itself

MAJ = 259 for the loopback partitions

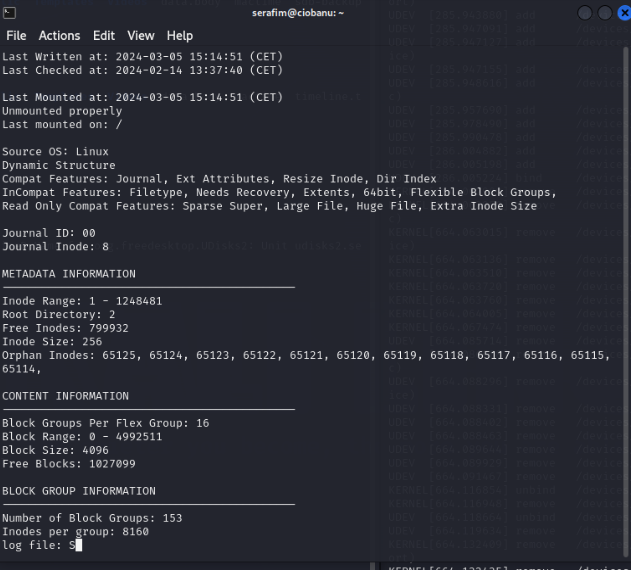
* Now run lsblk again, but with the --fs option to get information of the file systems within the block devices. What file system is the partition of your kali formatted with?



It is **ext4** (if we are talking about boot partition)

* Use dumpe2fs and fsstat to see detailed file system information. What is the block size that is used for you’re the partition kali is installed on?  
  Hint: you might need to pipe to less for easy output scrolling.

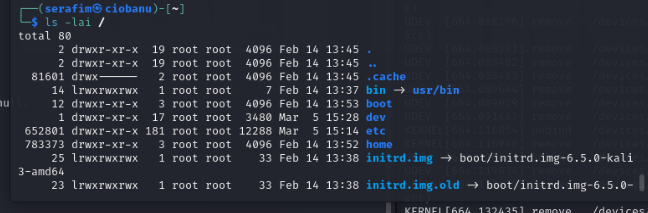
sudo fsstat /dev/sda1 | less



Block size = 4096

* What inode number does the / directory have? Use ifind or ls to figure out.

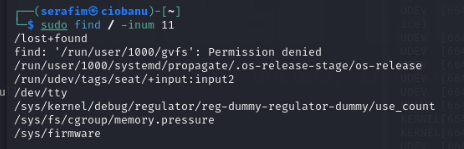
ls -lai /



The inode number is **2**

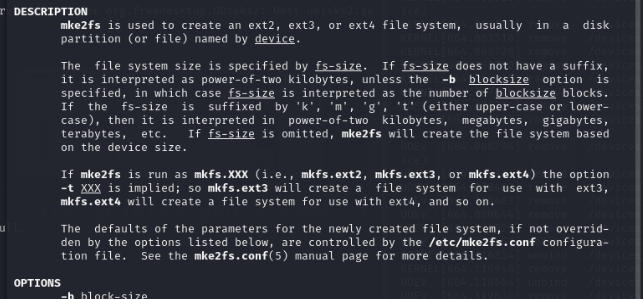
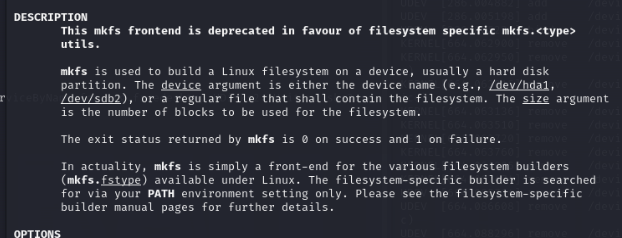
* What file or directory has inode number 11? Use ffind or find -inum to figure out.

sudo find / -inum 11

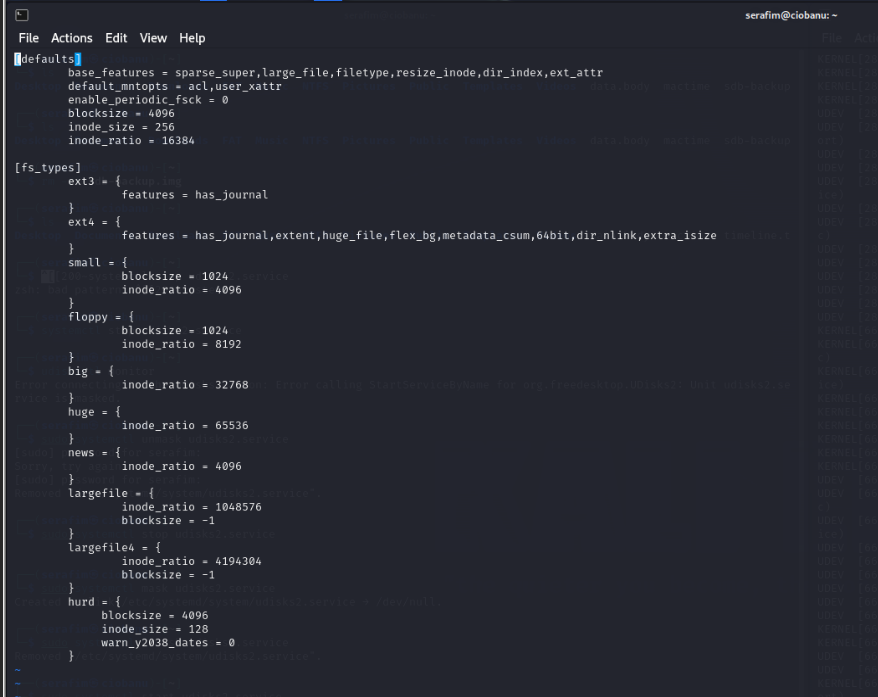


**Most probably lost+found from what I remember**

* Have a look at the man page of mkfs.ext4 , the command to format a partition with the ext4 file system. What tool is actually being called if you would run mkfs.ext4 ?



* This tool has a configuration file in /etc directory. What is this file? Have a look at it. What is mentioned as default block size? What block size will be used when formatting ‘small’ (less than 512 MB) partitions?



I suppose it will use the block size of 1024, as it is specified in the small sector

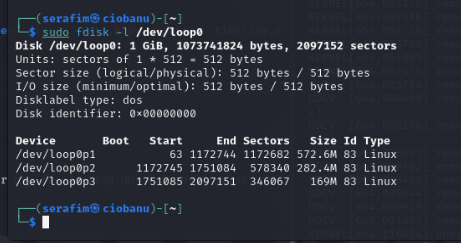
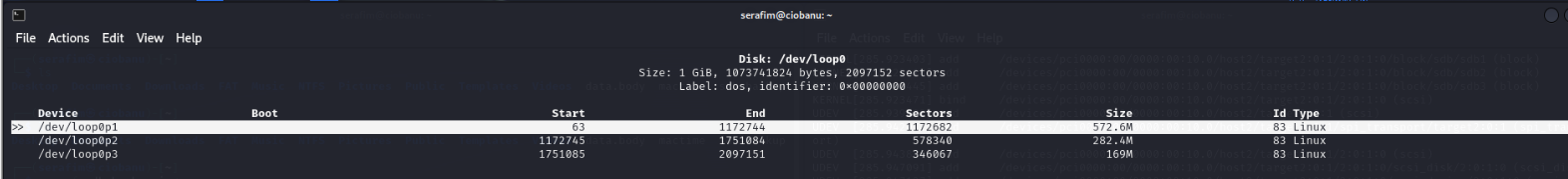
# Analysis of the suspect’s disk: capture the flags

Ok, move on to our suspect disk and get the thrill of finding the flags. We made it easy as the disk wasn’t used intensively and only few files were put on the disk. This small sample is for you to focus on the details of how the file system works.

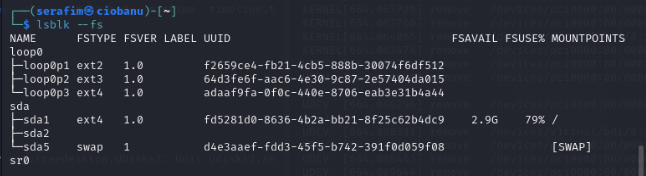
### Disk layout analysis

First, have a look at the disk layout to identify the partitions (cfr previous labs) and the file systems they are formatted with.

* Is there any substantial amount of free space (neglect the first free sectors)? How many partitions are there? What file system are they formatted with?  
  (tools: fdisk, parted, lsblk)



All of them are formatter with 83, which is related to be compatible with something like ext2,3,4 and etc.



ext2, 3, 4

### Data recovery on an ext2 file system (3 flags)

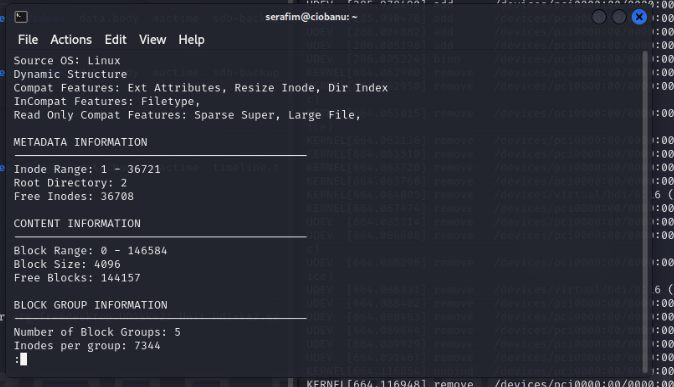
FLAGS FOUND:

FLG-77968854 (recovery of file), FLG-20914599 (another recovery), FLG-36458742 (present on file system)

* Find out which partition is ext2 formatted (lsblk, fsstat). What is its UUID?

UUID=f2659ce4-fb21-4cb5-888b-30074f6df512 (/dev/loop0p1)

* What is the block size used by ext2 for this partition (dumpe2fs, fsstat)?



sudo fsstat /dev/loop0p1 | less

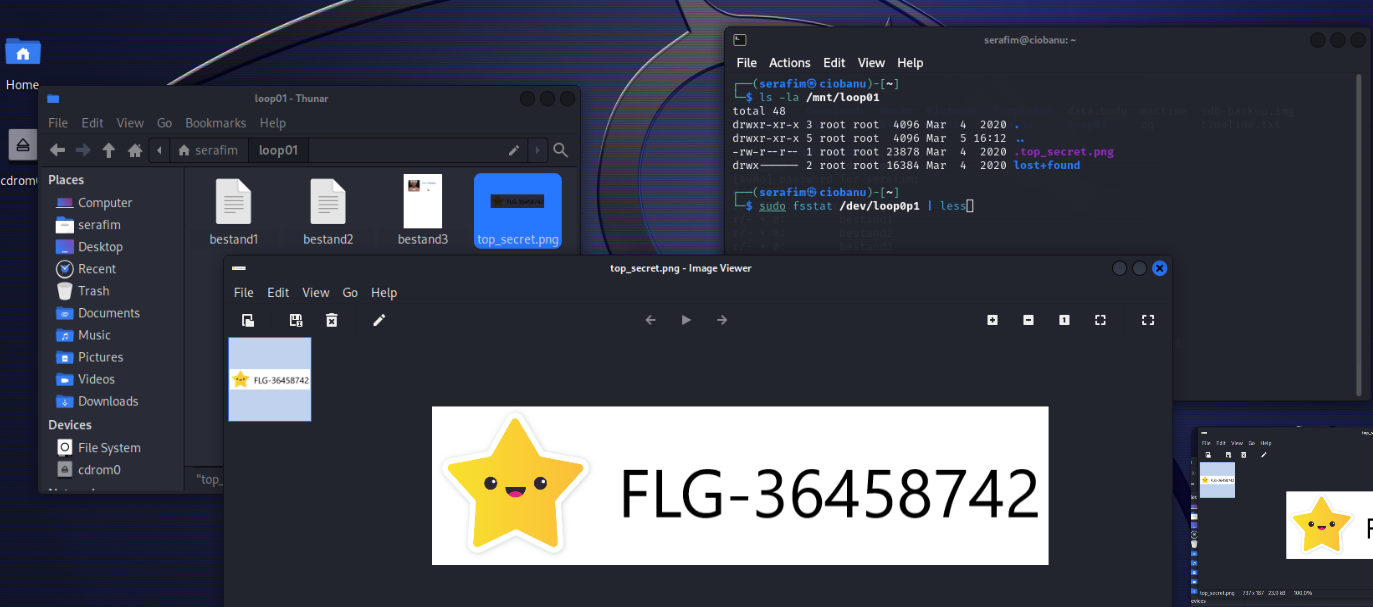
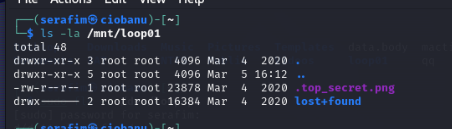
4096

It is bigger than 512MB (the partition)

* What file is not deleted on the disk (and can thus be shown in an operating system if mounted)? It will award you with a flag.

sudo icat -f ext2 /dev/loop0p1 12 > ~/loop01/top\_secret.png

You can also just `sudo mount /dev/loop0p1 /mnt/loop01`

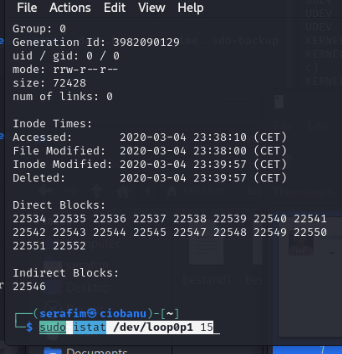


* Find the 2 other flags in the ext2 partition by recovering the data of deleted files (fls, istat, icat, blkcat, file)? Be able to say in which inode and in which data blocks the flags are found.

**FIRST ONE**

sudo icat -f ext2 /dev/loop0p1 15 > ~/loop01/bestand3

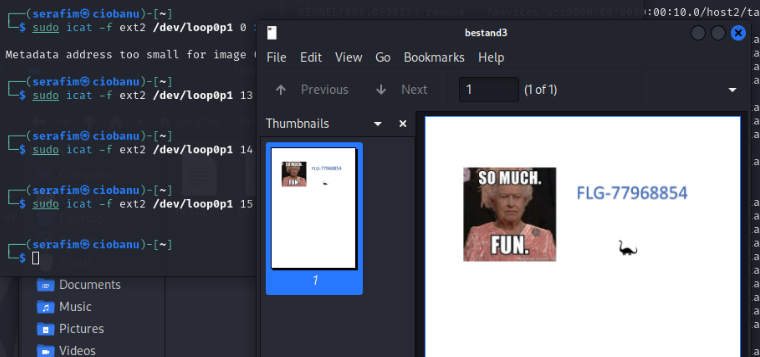
sudo istat /dev/loop0p1 15



sudo blkcat /dev/loop0p1 22546



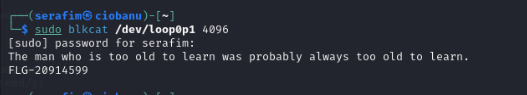
Inode 15, block 22546



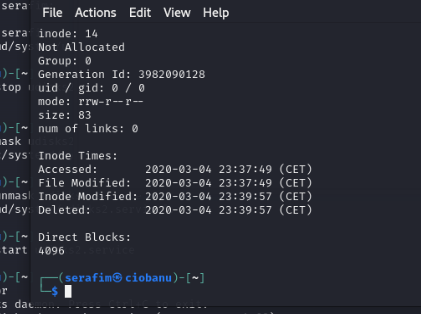
**SECOND ONE:**

sudo icat -f ext2 /dev/loop0p1 14 > ~/loop01/bestand2

sudo blkcat /dev/loop0p1 4096

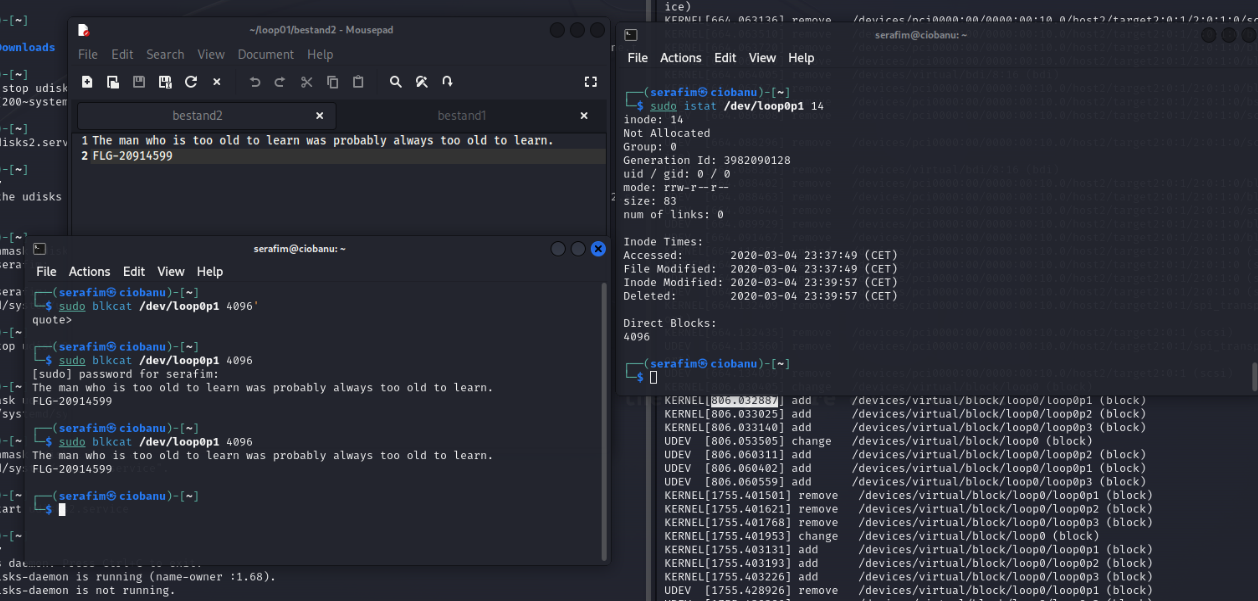


sudo istat /dev/loop0p1 14



Inode 14, block 4096

For the flag starting with “FLG-2”, upload a screenshot of the tool and its output which reveals the data block number of that flag.



* Can you link file names of deleted files to the recovered file data?

No, you can not.

### Data recovery on an ext3 file system (1 flag)

**FLAGS FOUND:**

**FLG-99658658 (from the file name)**

* Find out which partition is ext3 formatted (lsblk, fsstat).

/dev/loop0p2

64d3fe6f-aac6-4e30-9c87-2e57404da015

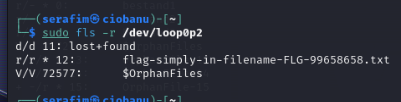
* What is the block size used by ext3 for this partition (dumpe2fs, fsstat)? Why?

sudo fsstat /dev/loop0p2 | less



It might have the **1024** block size for backwards compatibility with older systems maybe, or it is just because it is rather a small system. The partition is smaller that 512MB (partition)

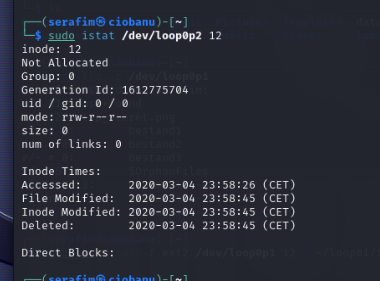
* Find the 1 flag in the ext3 partition, related to a deleted file (fls, istat)?



* Can you know the inode and the data blocks the deleted file was using?

The inode - 12.

But you cannot find the blocks it used, from the istat at least.

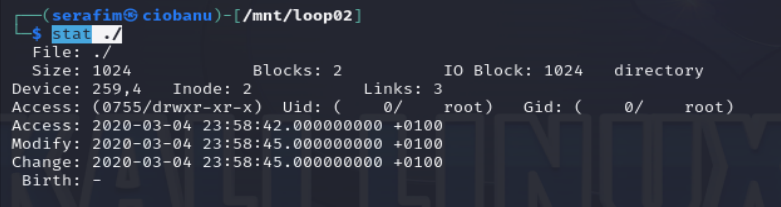


* A directory is actually a file itself, which also uses data block(s). In that/those data block(s) it keeps a list of file names and their inodes. This is what tools like fls are parsing. What data block was used for the directory that had the deleted file? Upload a screenshot of the tool(s) and its output which reveals the data block used for this directory.

sudo mount /dev/loop0p2 /mnt/loop02

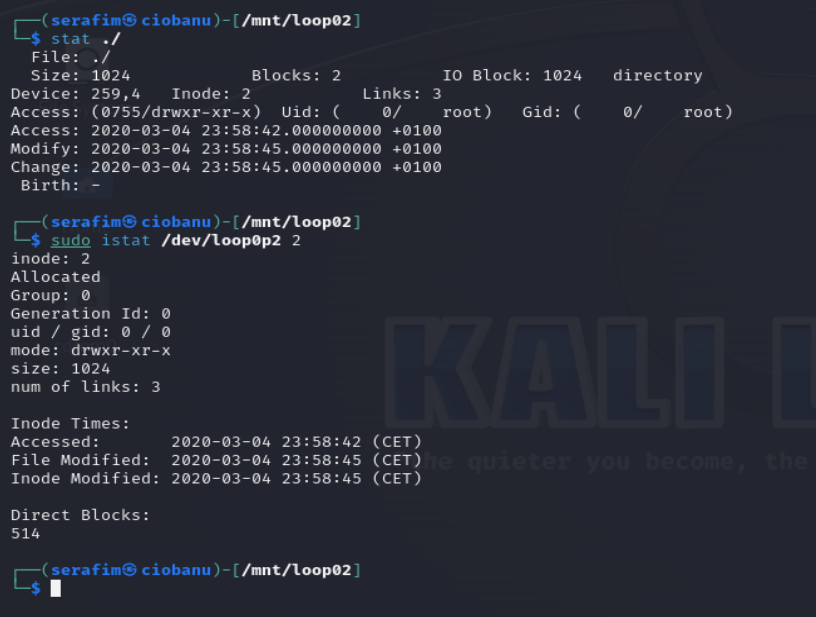
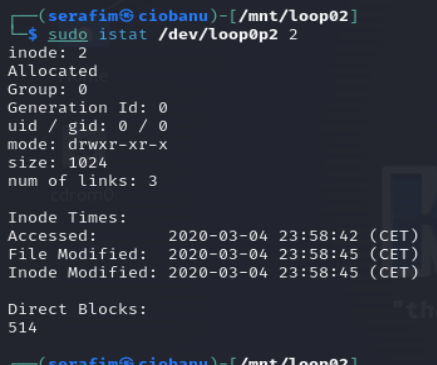
cd /mnt/loop02

stat ./



The inode is 2, we can find that only if mounted, from what I saw

sudo istat /dev/loop0p2 2



### Data recovery on an ext4 file system (1 flag)

**FLAGS FOUND:**

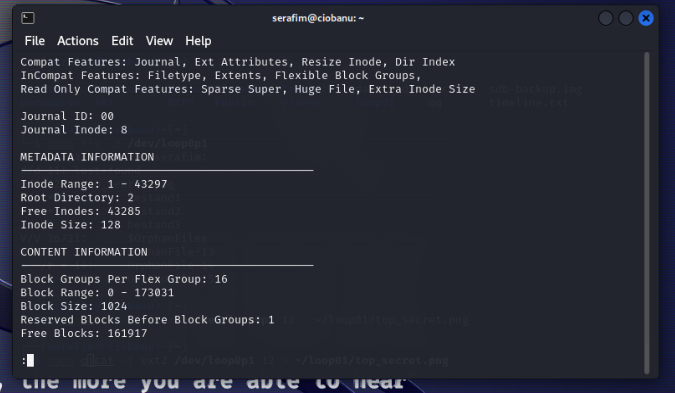
**FLG-61330010 (strings)**

* Find out which partition is ext4 formatted (lsblk, fsstat).

adaaf9fa-0f0c-440e-8706-eab3e31b4a44

/dev/loop0p3

* What is the block size used by ext4 for this partition (dumpe2fs, fsstat)? Why?



sudo fsstat /dev/loop0p3 | less

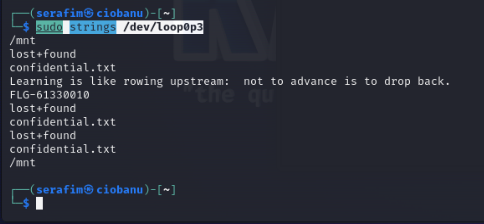
A block size of 1024, again because it looks smaller, and the partition is smaller than 512MB

* What file was deleted? Similar as in ext3, you will not be able to recover the file by looking at the inode. Why?

Because it was probably deleted, or overrwritten, or the contents were deleted.

* To still recover data (including a flag), we could use ‘data carving’ to analyze all data blocks (as these aren’t zeroed) to look for something useful. More on data carving later but try strings as a tool which scans for text.

sudo strings /dev/loop0p3



### Extra bonus flag (1 flag)

We have furthermore hidden deliberately (here again: more on data hiding later) an extra flag within the disk. Can you find it? If you don’t, pay attention during next week’s presentation about the ‘ext file system metadata’. (tools: xxd, dd, file)

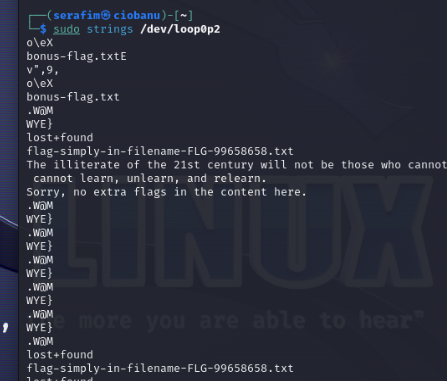
FLAGS FOUND:

FLG-55817745 (xxd reading)

sudo xxd -l 512 /dev/loop0p2



sudo strings /dev/loop0p2



sudo xxd -l 512 /dev/loop0p2 > dump.txt

sudo xxd -r dump.txt dump.zip

unzip dump.zip

