$ Sudo fdisk -l: to view the disk informn.

$ Sudo dc3dd if=/dev/sdb1 of=image.dd : to create a image of the particular disk.

$ Mkdir foremost: directory to store carved contents

$ cd /etc/foremost.conf: conf. file is inside the etc folder.

$ Sudo foremost -i image.dd -o foremost : carve out the contents from image file into new directory foremost.

Use -t to list diff data types, jpg, zip, etc or all

$mkdir recoverjpeg: empty folder

$ sudo recoverjpeg image.dd -o recoverjpeg: carve out into recoverjpeg folder.

$ Mkdir scalpel: to store the carved contents.

$ Sudo scalpel -o scalpel image.dd: to carve out the contents.

⇨ You have to change the configuration file accordingly to extract the type of files from the disk as in foremost tool.

Sudo apt-get install bulk-extractor

Sudo bulk_extractor -o bulk image.dd: to use bulk tool to carve

⇨ Bulk tool carves out various types of file like zip and others that are generally not done by scalpel and foremost.

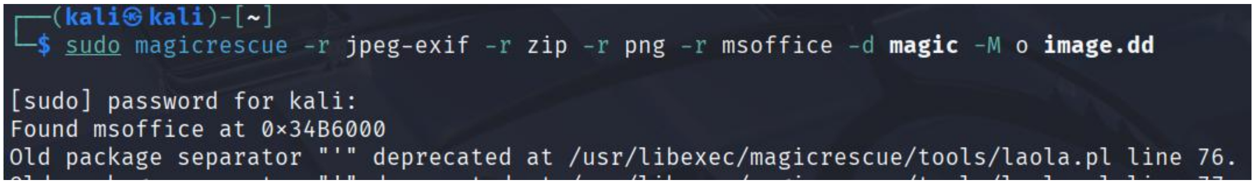
Sudo apt-get install magicrescue

The conf. files are known as recipes.

Present in : cd /usr/share/magicrescue

Cd recipes: this contains the file signature of files that this tool can extract.

To extract:



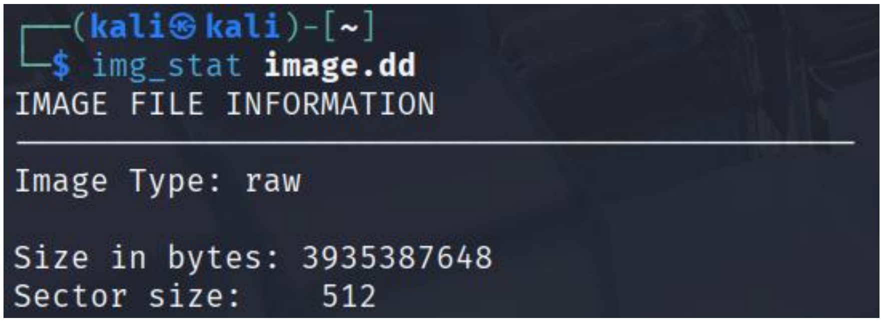
```
┌──(kali㉿kali)-[~]
└─$ sudo magicrescue -r jpeg-exif -r zip -r png -r msoffice -d magic -M o image.dd

[sudo] password for kali:
Found msoffice at 0×34B6000
Old package separator "'" deprecated at /usr/libexec/magicrescue/tools/laola.pl line 76.
```

-d -M o : means more with output from the file specified to the directory magic which will hold the carved data.

- ➔ The above listed tools are based on file signatures, magic number or whatever you wanna call it.

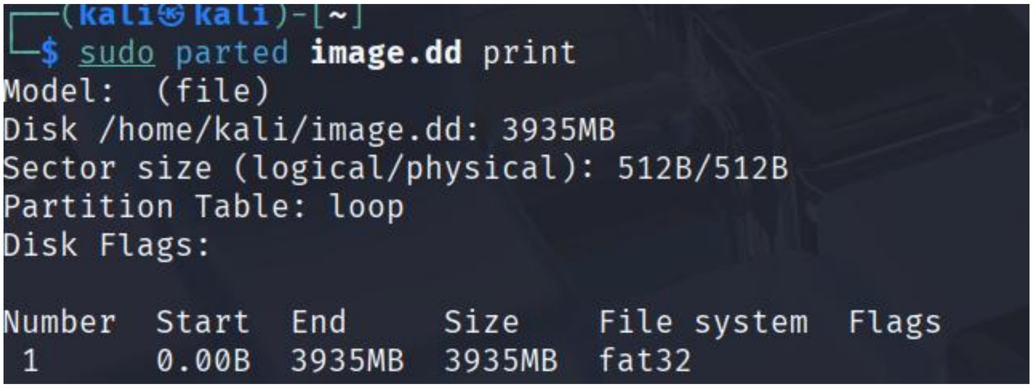
Now based on file system we use tools like sleuthkit.



```
┌──(kali㉿kali)-[~]
└─$ img_stat image.dd
IMAGE FILE INFORMATION
--------------------------------------------
Image Type: raw

Size in bytes: 3935387648
Sector size:    512
```

Img_stat : gives infomn about the image file.



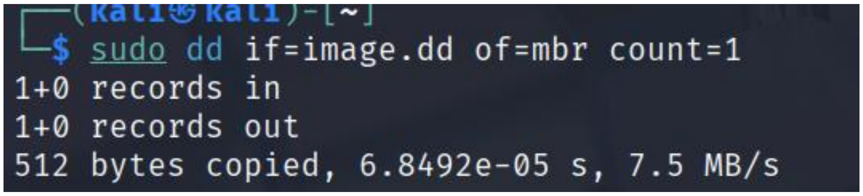
```
┌──(kali㉿kali)-[~]
└─$ sudo parted image.dd print
Model:  (file)
Disk /home/kali/image.dd: 3935MB
Sector size (logical/physical): 512B/512B
Partition Table: loop
Disk Flags:

Number  Start  End     Size    File system  Flags
 1      0.00B  3935MB  3935MB  fat32
```

Mmstat, parted image.dd print: gives the infomn about the type of partition table being used.

Mmstat -i list: supported image formats.

Mmstat -t list: supported partition types



```
┌──(kali㉿kali)-[~]
└─$ sudo dd if=image.dd of=mbr count=1
1+0 records in
1+0 records out
512 bytes copied, 6.8492e-05 s, 7.5 MB/s
```

Extracting the MBR(1st sector) of the disk from the image file. By default it captures the initials 512 bytes(1st sector).

$ xxd mbr : to view the mbr in hex format , 55aa is the MBR signature.

$ mmls image.dd: shows the starting offset of the data (LBA)

$ fsstat -o 8064(starting offset of the data) image.dd: shows the file information.

$fls -d 8064 image.dd ; shows the deleted files which are marked by *

To recover we can use tsk tool.

$ tsk_recover -o 8064 -e image.dd ~/output: e for marked as well as unmarked, ~ for home directory.

➔ You can use sleuthkit to retrieve data by exploring file system and then the sectors which are there.

TestDisk is a free and open-source data recovery utility that helps users recover lost partitions or repair corrupted filesystems.

$ sudo apt-get install testdisk

$ sudo testdisk : to create a new log file > choose the connected drive to the forensic station > select the partition table type (intel by defult for dos)

Deeper search to extract the deleted partitions.

Go to adavanced > undeleted : to view the deleted partitions in the cylinder (CHS offset) of the drive.

Deleted files are marked in red, to recover follow the instruction given.

Volatility : A tool that is used to analyze the memory dumps, RAM memory, volatile memory

$ cd volatililty

$ python2 vol.py **--info**: lists all the plugins.

$ python2 vol.py -f /path/to/dd/file **imageinfo** (plugin): copy the image into the same directory so as to avoid path problems.

$ python2 vol.py -f /path/to/dd/file –profile= WINXPSP2x86 **pslist** : scan through the memory and lists all of its the Eprocess

$ python2 vol.py -f /path/to/dd/file –profile= WINXPSP2x86 **psscan** : looks for hidden ones,

$ python2 vol.py -f /path/to/dd/file –profile= WINXPSP2x86 **pstree** : to view the parent-child reln.

$ python2 vol.py -f /path/to/dd/file –profile= WINXPSP2x86 **psxview** : enumeration to see if all the process started alongside.

$ python2 vol.py -f /path/to/dd/file –profile= WINXPSP2x86 **connscan**: to check if connected to any foreign address.

$ python2 vol.py -f /path/to/dd/file –profile= WINXPSP2x86 **sockets:** to detect listening sockets.

$ python2 vol.py -f /path/to/dd/file –profile= WINXPSP2x86 **cmdline**: gives path where the .exe file runs

$ python2 vol.py -f /path/to/dd/file –profile= WINXPSP2x86 **procdump -p 1484 -D Dump**1(created beforehand): dumps the process 1484 to dump1 directory.

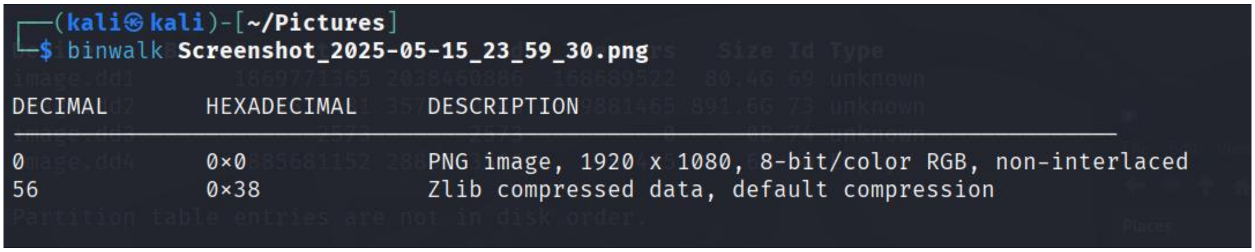
(volatility)

$ **strings** 1484.dmp | grep "41.x.x.x" -c 5 (+- 5 lines): Discover **human-readable text** within the memory (e.g., passwords, URLs, commands).

$ **malfind** -p 1484 -D Output: to check for any code injection.

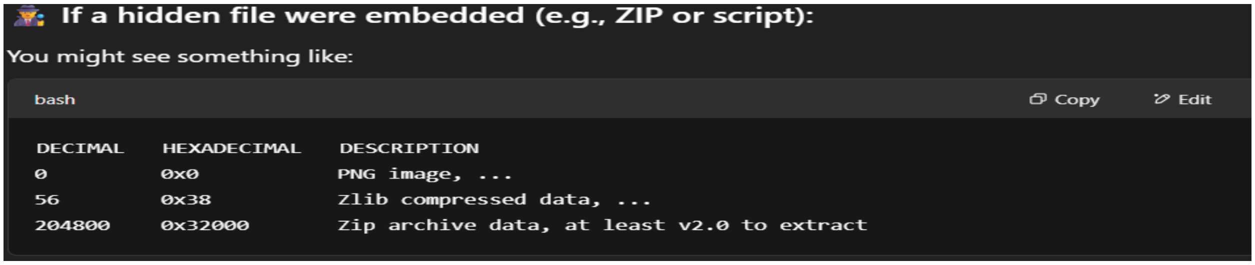
BINWALK: tool to view concatenated data (Appended data, data hiding)

$ binwalk image.jpg



```
┌──(kali㉿kali)-[~/Pictures]
└─$ binwalk Screenshot_2025-05-15_23_59_30.png

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PNG image, 1920 x 1080, 8-bit/color RGB, non-interlaced
56            0x38            Zlib compressed data, default compression
```



🕵️ **If a hidden file were embedded (e.g., ZIP or script):**

You might see something like:

```
DECIMAL    HEXADECIMAL    DESCRIPTION
0          0x0            PNG image, ...
56         0x38           Zlib compressed data, ...
204800     0x32000        Zip archive data, at least v2.0 to extract
```

$ binwalk --dd=".*" image.jpg : used to **carve and extract all identifiable embedded files** from image.jpg, using **smart extraction** based on file signatures (magic bytes).

$chmod +x 131A : **Make the file 131A executable. 131A is the hex of the executable file”**you’re likely referring to the **magic number** or **file signature** at the beginning of the file that identifies its type. Let's clarify what that means.

$ file program.exe : views the informn about the sections of the ELF (striped / not striped).

STEGHIDE: the practice of concealing a message

$ steghide embed -cf Screenshot_2025-05-15_23_59_30.jpg -ef secret.txt: hiding the text file inside the jpg file. cf= cover file, ef= embedded file .

$ steghide extract -sf Screenshot_2025-05-15_23_59_30.png : extracting the embedded file. **Stego File (sf)** – This is the file from which you want to extract the hidden data.

$ **xdg-open** image.jpg : to open the image.

$ sudo dd if=/proc/kcore : representing the **entire physical memory** (RAM) of the system.

$ sudo dmsg : log for the system.

$ sudo ddrescue -d r3 /dev/sdb mydrive1.raw mydrive1.log : starts reading form the faulty drive.