# **LAB 1**

## FTK IMAGER -WINDOWS

* Used for disk imaging
* Needed to maintain the originality of the original file so we create a bit-by-bit exact copy of the original device. It will include all information like Master Boot Record, files, Partition table entries, configuration files, settings, folders, deleted information etc.
* Image formats:
  + DD(Raw)
  + SMART
  + E01
  + AFF
* Export file hash list as CSV to see the hash signatures and analyse them online

# **LAB 2**

## DC3DD

### dcccdd (dc3dd)

* + department of cyber crime data duplicater
  + built over the normal dd command
  + hash can be easily generated
  + it gives a detailed view about what is happening

### dcfldd

* + dd: data duplicater (in built command and creates a bit by bit copy)
  + can be used to create the raw image file
  + it does not show detailed view of what is happening

### sudo fdisk -l : shows all the attached devices

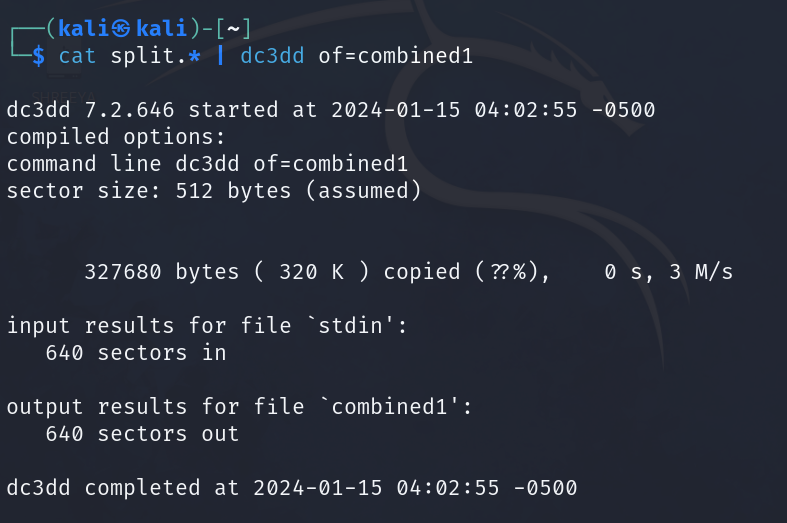
* + by default sector size is 512 bytes
  + Disklabel type: dos -> this means that dos partition table is being used
  + newer type of partition table is gpt which allows a lot more nuber of partition
  + every os has different Ids which are assigned to them.. The Id for Kali Linux is 83

### dc3dd USAGE

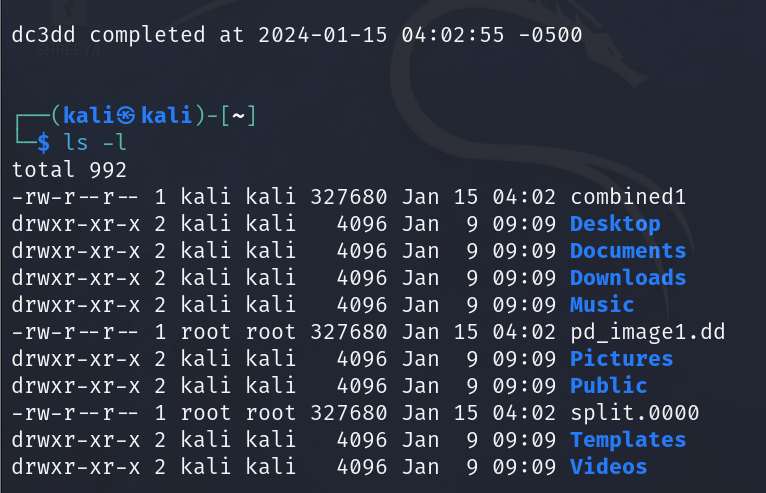
* sudo apt-get install dc3dd
* sudo dc3dd if=/dev/sdb1 of=pd\_image.dd
  + -if: input file
  + -of: output file
  + To abort simply do ctrl+C
  + After abort you will get a report like this:



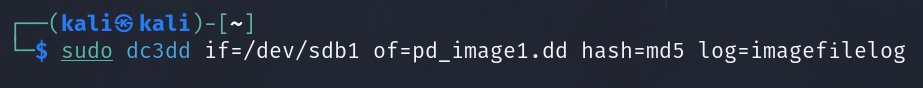
* with dc3dd we can split the files into chunks
  + sudo dc3dd if=/dev/sdb of=pd\_image1.dd ofsz=10M ofs=split.0000



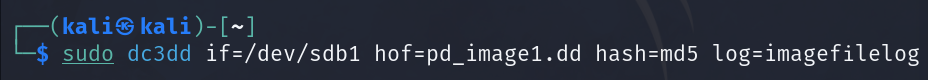
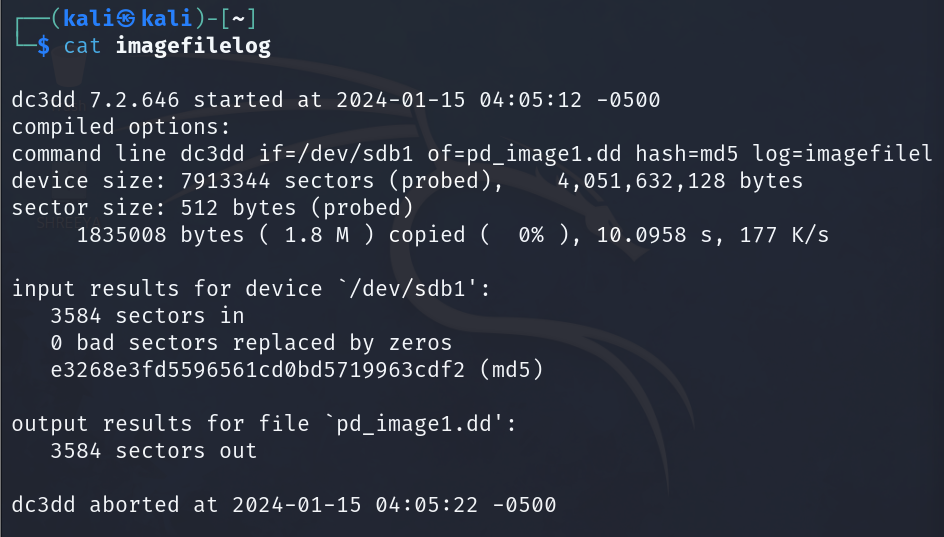
* this combines all files with the name starting with ‘split’ into one file named ‘combined1’



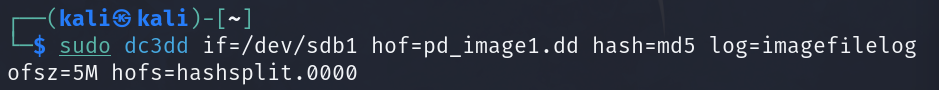
we can see the combined output.



* + With this we create a log file which contains the hash of the output file
  + The output looks like this: these are the contents of the log

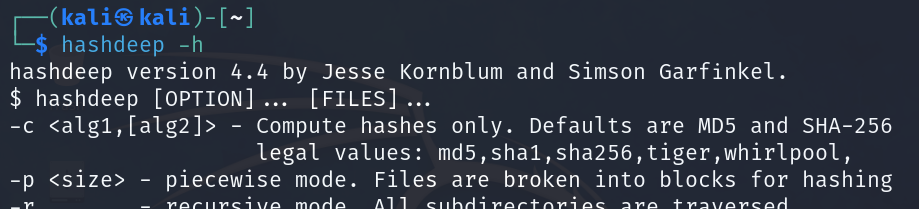


* + This tells dc3dd to compute hash only after the output is generated and not happen “on the fly” ie as the program is running



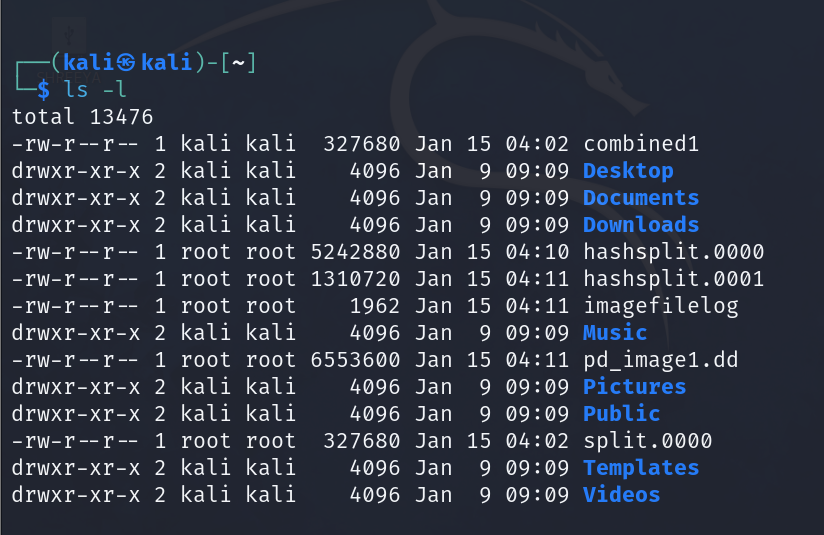
#### HASHING TECHNIQUES : md5sum & sha1sum

* **sudo md5sum /dev/sdb** or **sudo sha1sum /dev/sdb**
  + to see the hash
* you can also test for quicker results with a text file like
  + sudo md5sum hello
  + sudo sha1sum hello
* sha1sum and md5sum are using different algorithms you can check them using
  + **hashdeep -h**
  + to view all the algorithms



* the hash generated needs to be copied and saved then you should compare the hash of the image and the actual disk and make sure that the hashes are matching

Combining the concepts

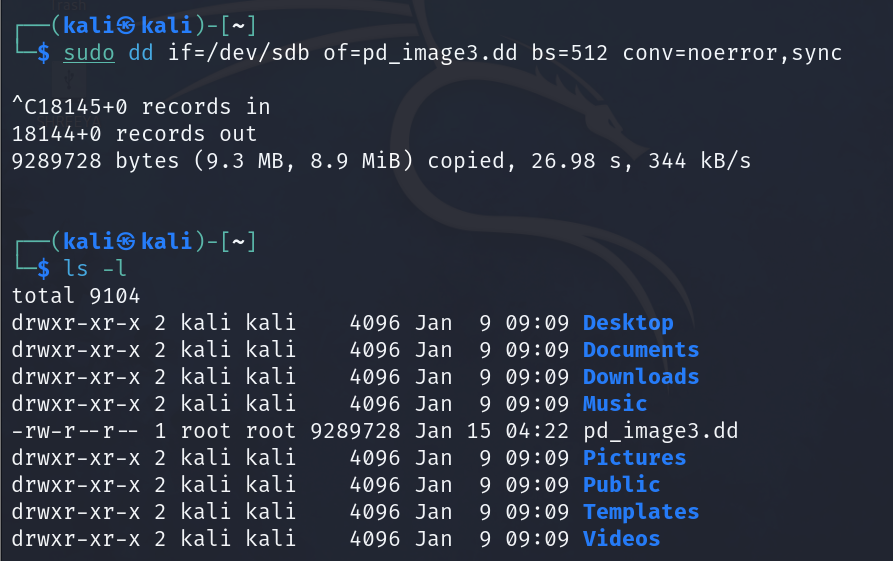
* + Output:
  + 
  + We can see how many hashsplit files are created
* To overwrite the contents of the pendrive:
* Sudo dc3dd if=imagefile of=pendrive (/dev/sdb)

### dcfldd USAGE

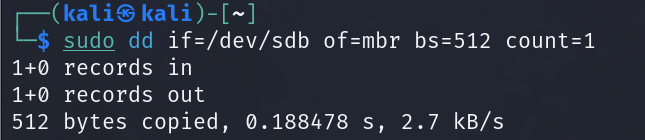
Similar to dc3dd we also have dcfldd(department of defence computer forensics lab) it does almost everthing that dc3dd does and is a little bit more interactive



* + Bs: block size (says how much to be read at once not same as split as we did earlier)
  + Conv=noerror,sync
    - This means that create an image such that you are assuming that there is no problem with the disk
    - Sync: as soon as a bad sector is encountered the bad sector is replaced with 0s. If this is not added then on encountering a bad sector the entire process will abort.
    - BAD SECTOR: a sector that is not readable or there is some issue in reading it
  + This does not show any status info as to how much data has been read or not
  + Output:



* + - Our pd\_image 3 has been created successfully



* + Count: to count the number of sectors
  + Count=1 is basically how many blocks that we are going to copy where each sector size is 512 bytes

### USING XXD TO ANALYSE THE MASTER BOOT RECORD



* 55aa: is the signature of mbr which can be seen at the end of the hexedit
* Xxd is for the hexeditor

# **LAB 3**

### How to check the number of drives currently connected to your device



### dmesg USAGE

* a buffer to store all messages generated by my device
* dmesg -T | less : with time stamps

### gddrescue USAGE

* Sudo apt-get install gddrescue
* Sudo ddrescue -d -r3 /dev/sdb myimage.raw myimage.log
  + This command is most helpful in the case of faulty drives and we can set a number of re-tries in case of failure which in this case is 3

### FILE SIGNATURES & DATA RECOVERY – Finding files by their file signatures

* recovery of deleted data without knowing the metadata of content...called file carving
* using the magic bytes (signatures specific to that file, header and footer)
* signature based tools

#### TOOL 1: FOREMOST

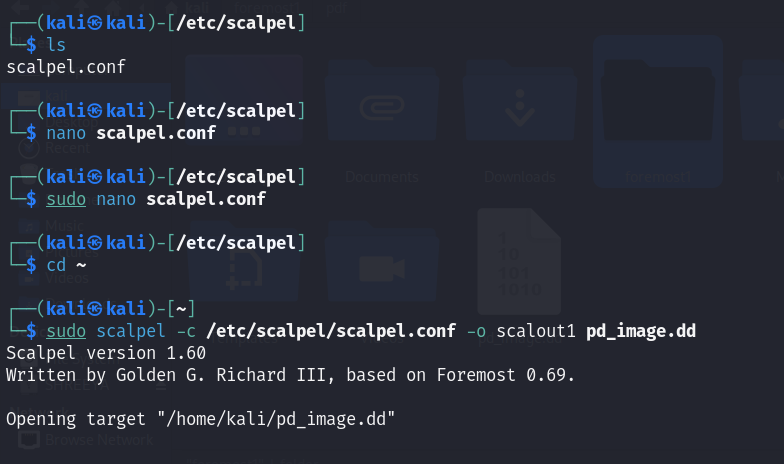
* + **Foremost:** in **/etc/foremost.conf**  you will find the header/footer and signatures of some file types. You can include your own file formats in this file. You can change this
  + Mkdir foremost
  + **Sudo foremost -i pd\_image.dd -o ~/foremost/**- to carve out the files by file type



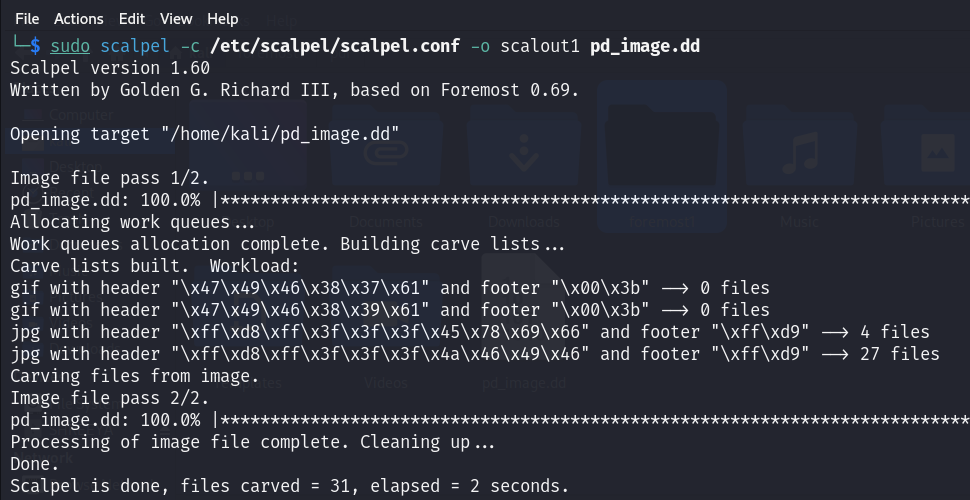
* + You “NEED” to have a folder to extract your ‘foremost’ files

#### TOOL 2: SCALPEL

* You can see the configurations in **/etc/scalpel/scalpel.conf**
* REVERSE keyword can be used to search backwards from where the footer is starting
* After uncommenting the files that we want to extract from scalpel.conf then we run the command **sudo scalpel -c /etc/scalpel/scalpel.conf -o scalout1 pd\_image.dd**

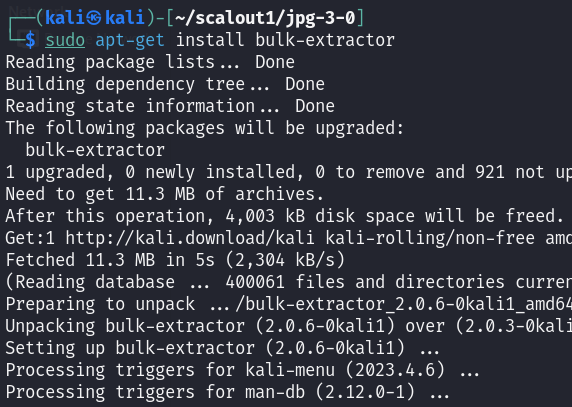


* + We get an output like this:



* + To view the images: “ feh \* ”

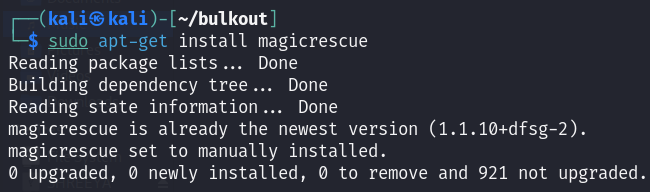
#### Tool 3: bulk extractor

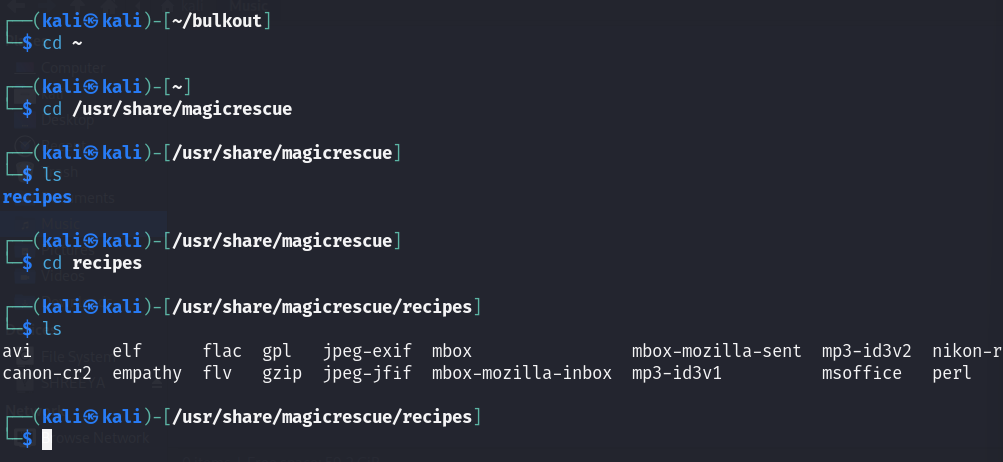


* To get the output:



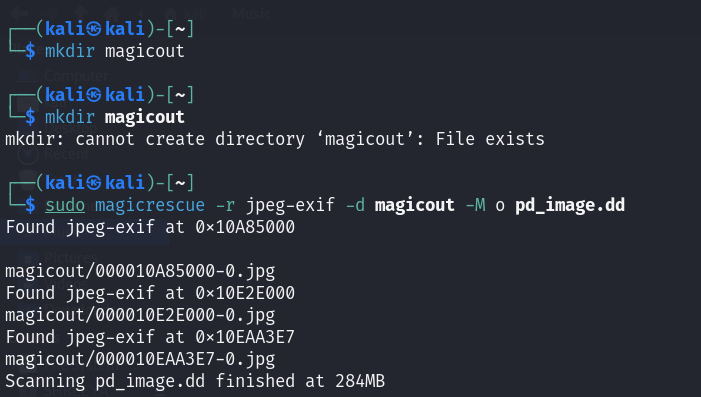
#### Tool 4: magicrescue



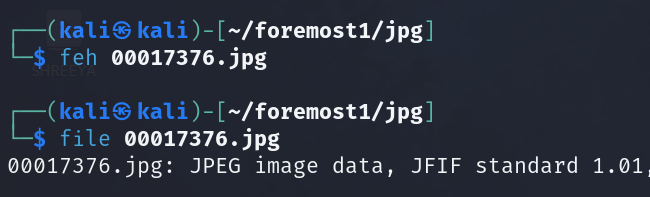




* Sudo magicrescue -r jpeg-exif -d magicout -M o pd\_image.dd
  + -r : the type of recepie we want to read
  + -d: the file in which it will be carved out
  + -M: to specify the file from which we want to extra



* To extract information:



#### Tool 5: binwalk- to extract binaries

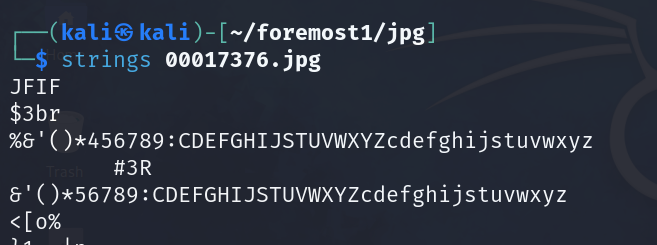
* **Binwalk -e 00017376.jpg**

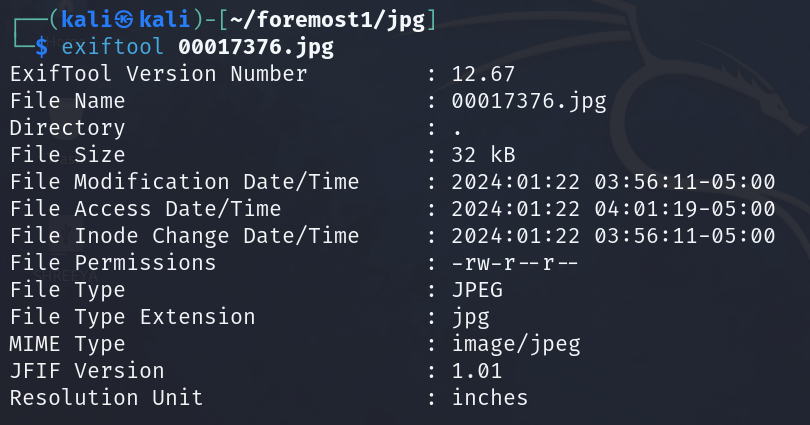


* binwalk -e 3.jpg for extracting from binaries
* binwalk -dd='.\*' 5.jpg for extracting from binaries

#### Tool 6: strings- to extract strings from files

* **Strings image.jpg**





* use '>' tool to embed text/string in binaries or vice versa

#### Tool 7: grep

# **LAB 4**

## FILE SYSTEMS

Looking at file systems is another way to find the data

File systems know where the data is contained

Step 1: Looking at file systems

Step 2: Looking at the data

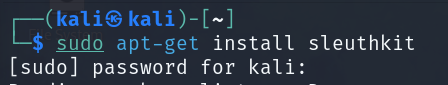
We are going to look at the file system from the image file

Image file copy: bit by bit copy: contains all the info about the sectors

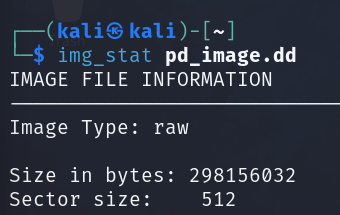
### STEP 1: Tools TO ANALYSE THE FILE SYSTEM

#### SLEUTHKIT

* Sleuthkit: contains a large number of binaries which helps us decipher the file system

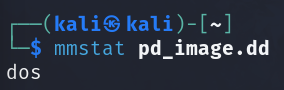


#### Img\_stat: info about the image



#### MMSTAT

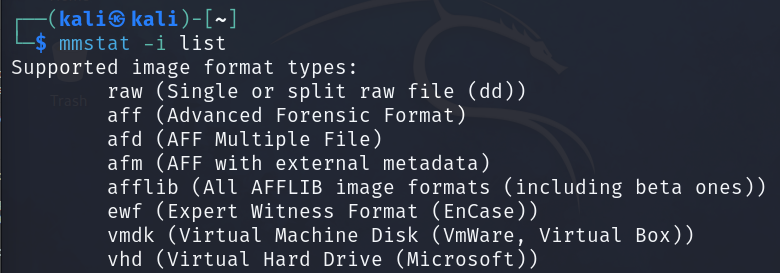
* Mmstat: tell about the layout of the disk from which the image has been taken



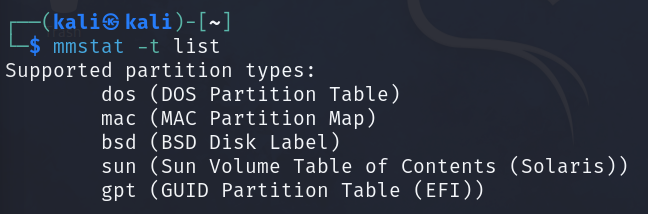
Dos is the type of sector

TYPES OF FORMATS SUPPORTED BY MMSTAT?

* Types of formats supported by mmstat

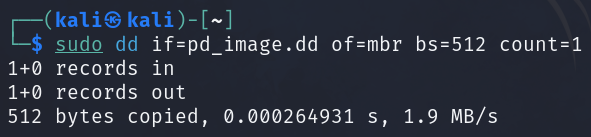
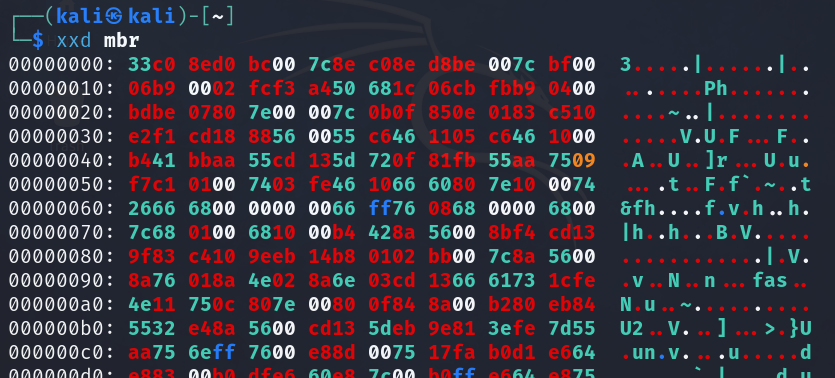
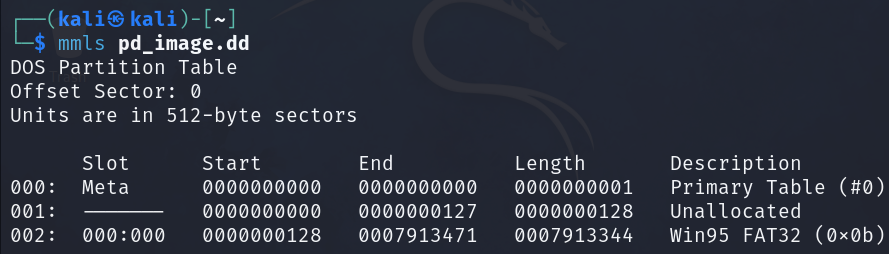
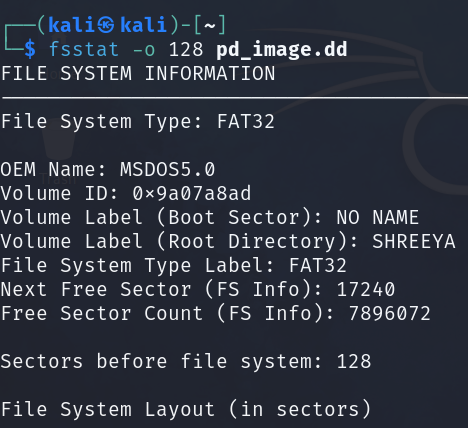
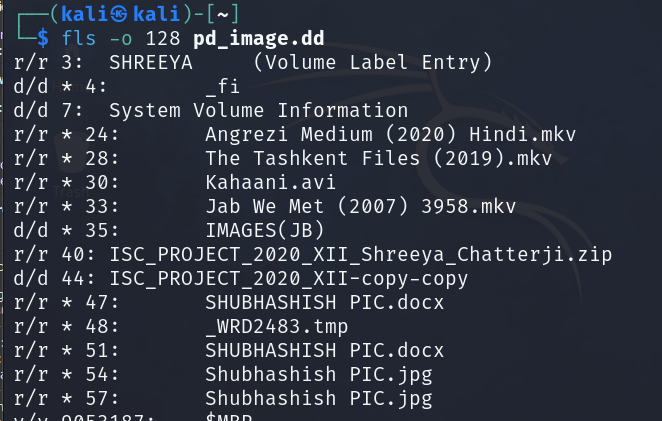
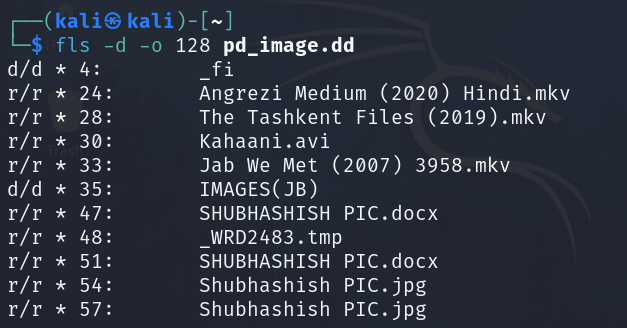


* Mmstat -t list: gives us the supported partition types:



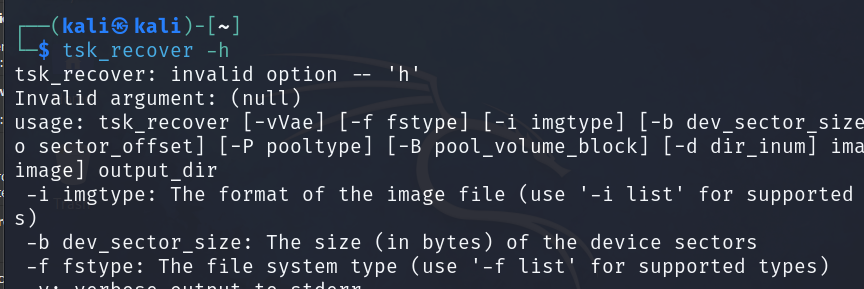
Gpt is a new tis a new type of format

### STEP 2: EXTRACTING SECTORS

* Now we will exract the sectors
* Sudo dd if=pd\_image1.dd of=mbr bs=512 count=1
* 
* Now reading the partition: xxd mbr
* 
* Understanding the dos partition table is key: the id for Linux is 83 and for pendrive is : 0b (FAT32)
* DOS PARTITION TABLE
  + 0: bootable flag
  + Starting chs address
  + Id: 0b, 83
  + Ending chs
  + LBA- logical block address
* **mmls imagefile:** interprets the partition tables
* 
  + Shows the sectors: eg: we can see that the pendrive data is starting from 128 and we can also see the description as FAT32
* **fsstat -o 128 pd\_image.dd : file system stat** the offset of 128 indicates the staring of the pendrive in pd\_image.dd
* 
* File system exists inside the partition so we are looking at the file system of the particular partition which in the above case is our pendrive image part
* 
* From the contents we can see that the data has been stored in a “sparce format” that is not all data is stored in one plac.e for a given address the address to the next data position is written along with it.
* **Fls -o 128 pd\_image.dd :** to list all the files in a given partition
  + 
  + Eg: **r/r 3: shreeya-** 
    - r/r – regular file
    - d/d directory
    - 3: block number
* To recover deleted files:
  + 
* To recover the files:



* + Icat -r(recover) blockstart imge block number
* **Tsk\_recover**





* WinHex

# **LAB 5- TESTDISK**

Testdisk

- sudo apt-get install testdisk

- sudo testdisk ->create -> pendrive->intel->analyze

- sudo testdisk pd\_image1.dd for woring with the image file instad of the pendrive

- p means primary type partition

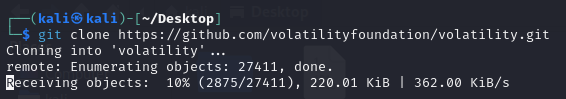
- file system : fat32

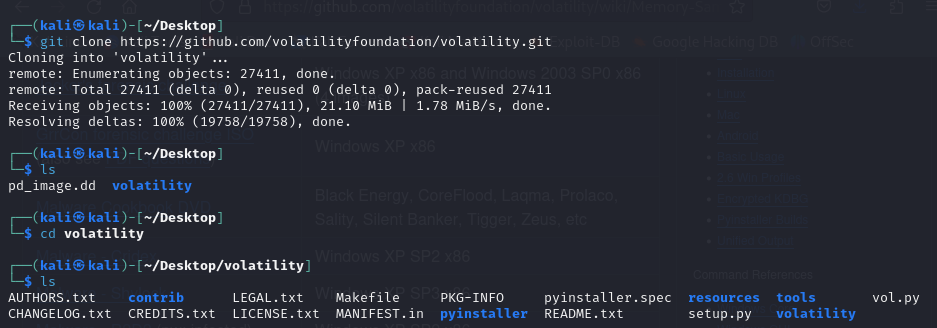
- number of heads/cylinders

- go back->advanced->undelete->go to file you want to recover-> shift colon-> shift C-> shift

# **LAB 6- MEMORY FORENSICS**

MEMORY FORENSICS

* + Approach volatile data first then non-volatile
  + Volatile: memory- memory contains a lot of info about the running processes- all processes rest in the memory and they are taken out from the memory
  + If you have the dump of the machine you can analyse it as it contains a lot of info like registry data, configurations, processes
  + All this info is first loaded into the ram and then it is noted out from the ram
  + From this ram info we need to check if there is any illegitimate process is running
  + TOOL: **VOLATILITY:**
  + 
  + You will need a file to inspect- like one that contains a trojan. Memory images are automatically installed by volatility
  + <https://github.com/volatilityfoundation/volatility/wiki/Memory-Samples>
  + Cridex one we install on the vm- we have to save the page link in the linux box.

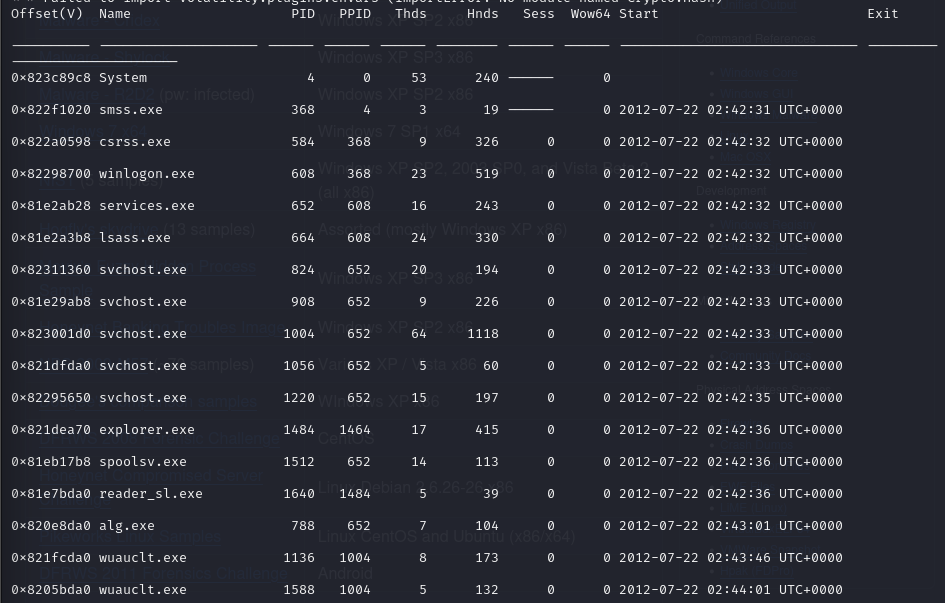


* + The cridex installation was not working so sir sent the file on LMS now we have to install from LMS
  + LMS pe upload nahi hua to pendrive mein file distribute kr rhe hai



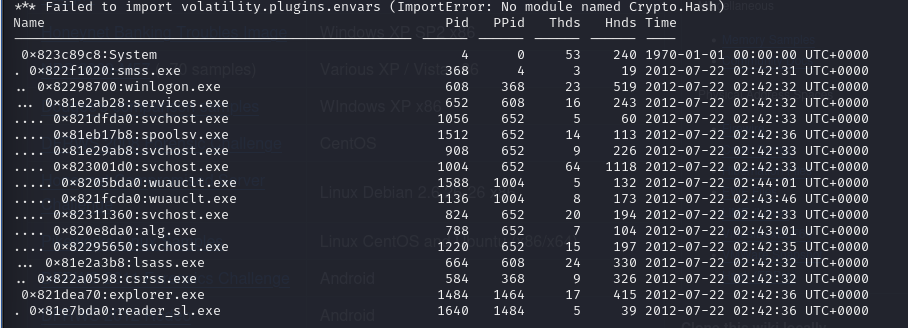
* + **Imginfo:** gives info about the profile of the os of the
  + Cridex: it is the malware or trojan that we are analysisng
  + **STEP 1: Look at the process list : pslist**
    - Python2 vol.py -f cridex.vmem imageinfo pslist
    - Smss: Service management sub system
    - CSRSS: Client Server runtime sub system
    - Winlogon: associated with maintaining the login accounts to start session for authentication
    - Svchost: a system process that allows the running of files by running and utilising dlls
    - Explorer.exe: it is used for all the GUI of our system.
    - Each process is started by the previous processes
    - The legitimate process must be known so that we can differentiate between legit and illegit process
    - Spoolsv.exe- related with printers- to put all printer related data in the main memory
    - Alg.exe- application layer gateway: allows known protocols to pass data block others
    - Wuaclt.exe
    - In our list the only one which is not known as a system process is **reader\_sl.exe**
    - To identify the parent child relationship examine we need to study the numbers and find the process with the process id given as parent





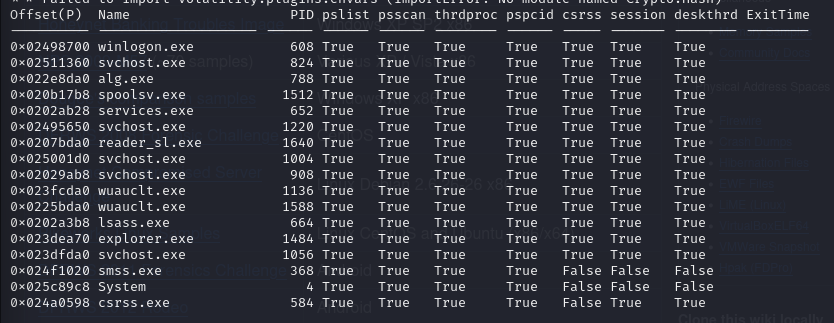
* + Pstree: shows how the parent child relationship
    - From the pstree we could see that the bad file ran after every other process was run so it raises suspicion



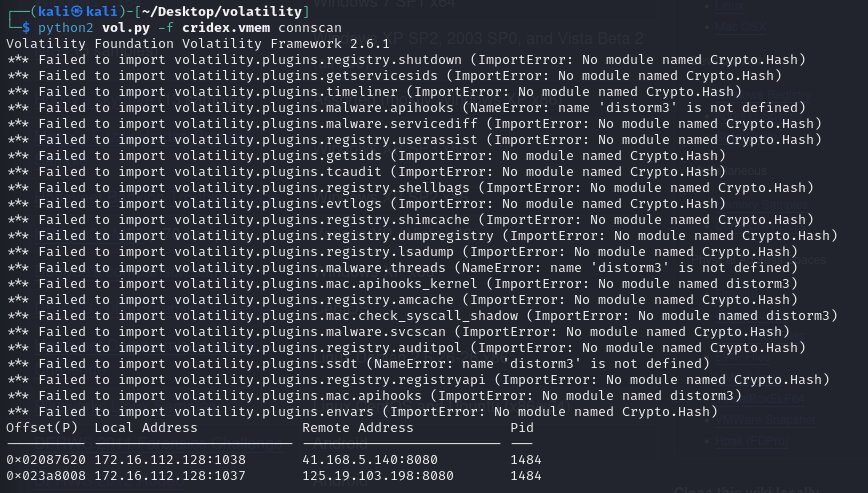


* + - The dots are showing when which process has started under which process
    - We can see that reader\_sl.exe is being started by explorer.exe as its PPID is same as the PID of explorer
  + Psxview: shows all the current processes running under different sessions



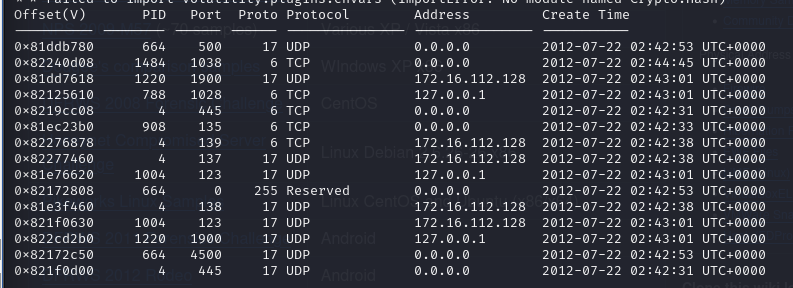


* + Connscan: to see what remote processes are being started by the trojan. When the system has just started immediately to connect to the outer world even when we have not started anything
    - We can see that the connection is being started by PID 1484 that is the PID of Explorer.exe and PPID of reader\_sl.exe



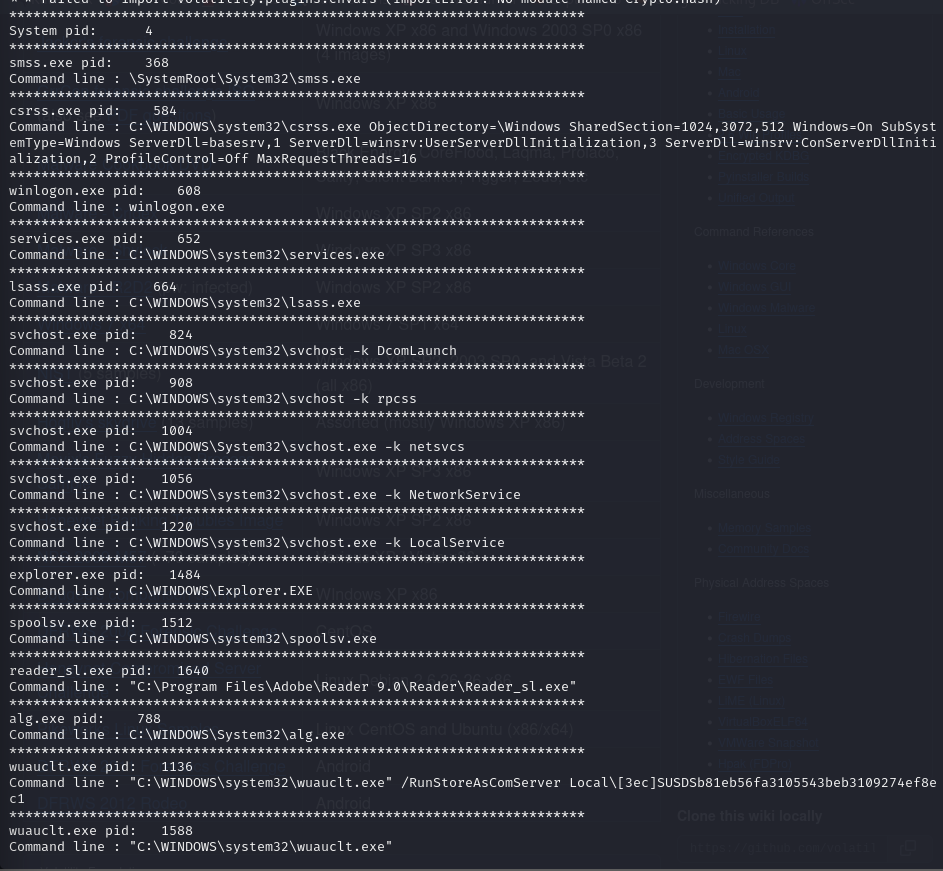
* + Sockets: tells us which ports are being used and how is the connection being established.





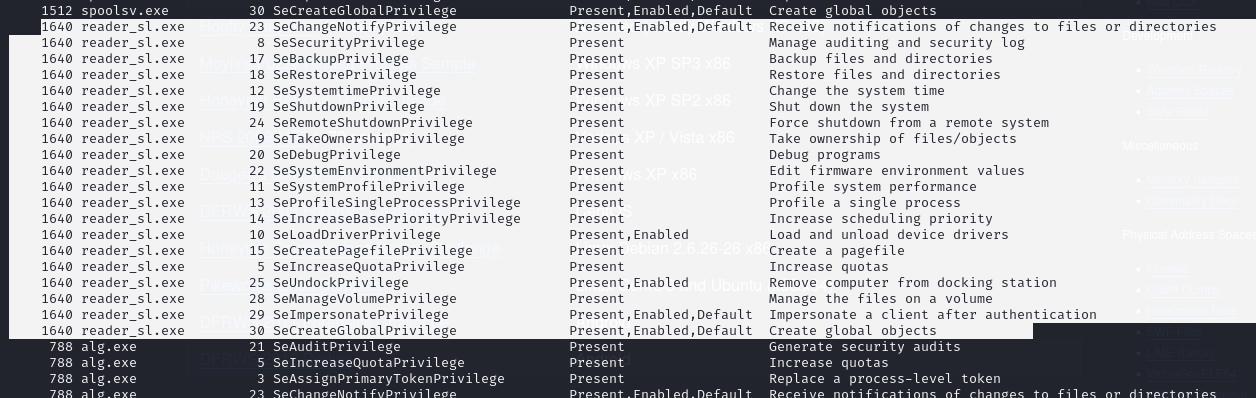
* + Cmdline: gives us a list of the different processes are being run from the different locations.
    - We notice that all processes are running from system32 expect for the PID1640 which is running from the program files so our suspicion is almost fully confirmed

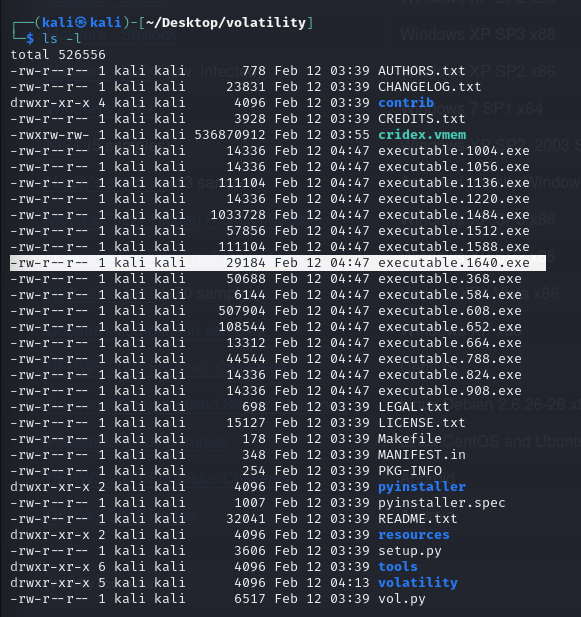




* + **STEP 2:** Now to confirm the nature of the infected data we take a dump of the malware file.
  + Look at the privileges: **privs**
    - Now we can study what the reader\_sl is doing
    - We can see that there are soo many scary things it can do

****

****

* + Taking the dump: **procdump -p 1640 –dump-dir .**
    - **<I made a mistake in this step I used -d instead of -p so all the processes procdump is saved>**
  + Ls -l
    - ****
  + File executable.1640.exe

****

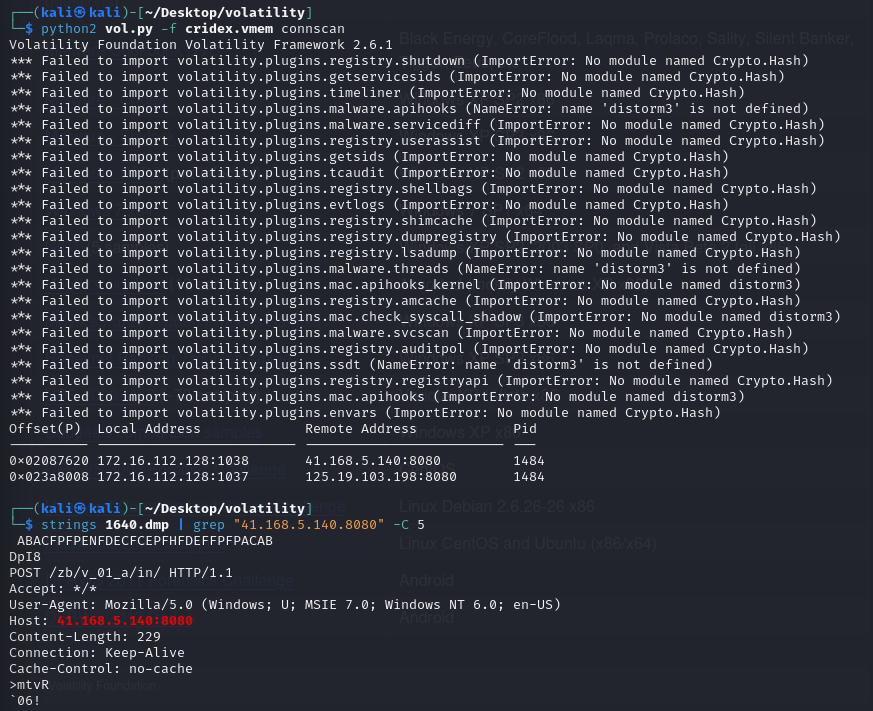
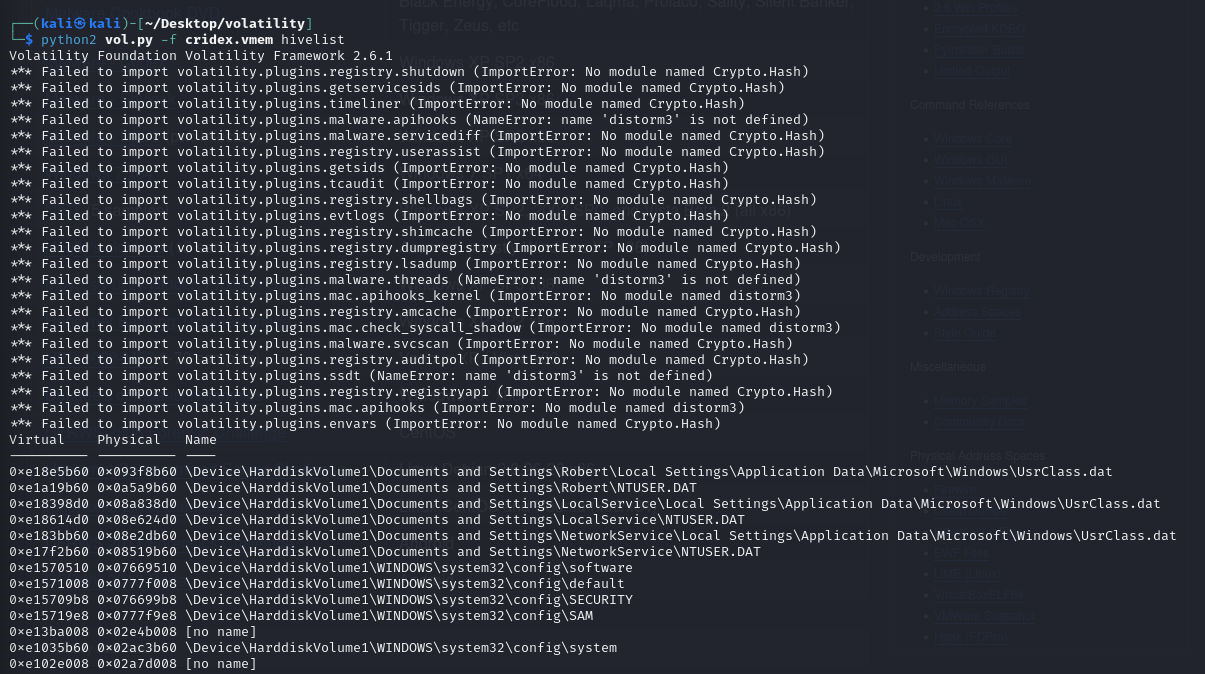
* + Md5sum executable.1640.exe- hash is created
    - Now paste this hash in “virustotal.com” – it is labelled as a bad one

****

* + Memdump -p 1640 –dump-dir
    - A dmp file is created

****

****

* + Strings 1640.dmp | grep ”string you want to search” -c 5 : showing the first 5 instances of the one being searched for
    - Trying to find the relevant info inside the memory
    - ****
    - **We can see that in the memory we find links to the ip address we found earlier in the connscan**
  + There are a lot of plugins associated with the volatility scan we have used some of them
  + **To** see the registry info: hivelist
    - **In windows we have “registry editor”** for the same task
    - The WINDOWS/Retaildemo/Run has info about all the startup processes
    - ****
  + Printkey: prints the registry keys, it’s uplinks and values: from the link we insert we can find the actual process being run- we will get the real name of the folder from runtime: which is taking the name reader\_sl.exe- how it is starting etc. The idea is that RAMS are full of info. Volatility is one tool. This is a command based tool there are other GUI based tools that do the same thing like FTK and other software. So if your system is being infected by malware and find the process and take the necessary steps and study how the malware is being run. Study which functions are being used for the task and find the functions that are assisting the same. Eg: virtual alloc- this is used by malware to get a lot o fvirtual memory space. Study the PE Header and what it will do

# **LAB 7- autopsy**

AUTOPSY

- ingest files to fetch data

- can do imaging as well

- During copying bit by bit, we are copying the hashes, OS, partitions, sectors etc

- Data cooling methods : process of refining the data

- shunting out unnecessary data (setup files of known softwares)

- 2 methods :

- we tell which files are known

- reduce the data size to be searched

INGEST MODULES :

- recent activity : files that were recently accessed acc to timezone

- hash lookup :

- go to global settings

- NSRL : national software reference library : gives list of known hashsets

- NSRL downloads->current RDS HashSet

- Type of hashsets :

- known

- notable

- no change

- file type identifiaction

- looks for file signatures

- global settings

- smtp could not understand anything other than ascii data.

- MIME : for sending data which is not just ascii,binary(block data) : multipurpose internet mail extensions

- text/plain is the default value for textual files

- application/octet-stream is the default value for all other cases

- https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics\_of\_HTTP/MIME\_types/Common\_types

- global settings

- create new mime type(refer to the word doc lab 7)

- to get signatire, use xxd myprogram.c

- offset will be 0

- offset relative to start

- now i can search for c files in my image

- extension mismatch

- flags files

- mismatch between file type and the extension

- embedded file extractor

- searches for doc, ppt, xlsx

- picture analyzer :

- metadata, geolocation of images

- keyword search :

- allows youi to extract certain numbers data (credit card no, phone numbers, emails)

- allows you to retrieve text from image as well (ocr)

- global settings :

- new list

- new keyword : specific strings to search for

- email parser

- parses your psd/osd files (outlook files)

- encryption detection

- check whether a file has something encrypted in it

- algo works by finding highest entropy (more randomness means stronger to deetct, so file is encrypted)

- interesting files identifier :

- adoble creative cloud, dropbox.exe, cloudme.exe, torr, vpn etc

- cryp wallets

- encryption progarams

- privacy programs

- central repository

- items that are flagged are stored here

- photo rec carver

- used for extracting images

- part of sleuthkit

- virtual machine extractor

- any ova, vm files

- data source integrity

- computes and verifies the existing hashes

- android analyser

- uses aleapp tool to analyze images from android phones

- ios

- uses ileapp

- dji

- for drone shot images

- plaso

- looks for artiifical files

- looks for timelines

- parses windows registry files

- parses pe header files

- yara

- static analyzer which gives rulesfor malwares identification and its classification

- one yara rule will have : name, condition, strings, metadata

- wannacry is similar to this

- gpx parser

- images and files ke metadata mein longitude and latitude ka data hota hai

- exiftool image.jpg

- gives data about image