

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:
 - o DD(Raw)
 - o SMART
 - o E01
 - o AFF
- Export file hash list as CSV to see the hash signatures and analyse them online

LAB 2

DC3DD

dccdd (dc3dd)

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

dcfldd

- dd: data duplicator (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 number of partition
- every os has different lds which are assigned to them.. The ld 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:

```

^C
3178496 bytes ( 3 M ) copied ( 0% ), 17 s, 181 K/s

input results for device `/dev/sdb1':
6208 sectors in
0 bad sectors replaced by zeros

output results for file `pd_image1.dd':
6208 sectors out

dc3dd aborted at 2024-01-15 03:49:26 -0500

```

- with dc3dd we can split the files into chunks
 - `sudo dc3dd if=/dev/sdb of=pd_image1.dd ofsz=10M ofsplit.0000`

```

(kali@kali)-[~]
$ cat split.* | dc3dd of=combined1

dc3dd 7.2.646 started at 2024-01-15 04:02:55 -0500
compiled options:
command line dc3dd of=combined1
sector size: 512 bytes (assumed)

327680 bytes ( 320 K ) copied (??%), 0 s, 3 M/s

input results for file `stdin':
640 sectors in

output results for file `combined1':
640 sectors out

dc3dd completed at 2024-01-15 04:02:55 -0500

```

- this combines all files with the name starting with 'split' into one file named 'combined1'

```

dc3dd completed at 2024-01-15 04:02:55 -0500

(kali@kali)-[~]
$ ls -l
total 992
-rw-r--r-- 1 kali kali 327680 Jan 15 04:02 combined1
drwxr-xr-x 2 kali kali 4096 Jan 9 09:09 Desktop
drwxr-xr-x 2 kali kali 4096 Jan 9 09:09 Documents
drwxr-xr-x 2 kali kali 4096 Jan 9 09:09 Downloads
drwxr-xr-x 2 kali kali 4096 Jan 9 09:09 Music
-rw-r--r-- 1 root root 327680 Jan 15 04:02 pd_image1.dd
drwxr-xr-x 2 kali kali 4096 Jan 9 09:09 Pictures
drwxr-xr-x 2 kali kali 4096 Jan 9 09:09 Public
-rw-r--r-- 1 root root 327680 Jan 15 04:02 split.0000
drwxr-xr-x 2 kali kali 4096 Jan 9 09:09 Templates
drwxr-xr-x 2 kali kali 4096 Jan 9 09:09 Videos

```

we can see the combined output.

```
(kali㉿kali)-[~]
$ sudo dc3dd if=/dev/sdb1 of=pd_image1.dd hash=md5 log=imagefilelog
```

- 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

```
(kali㉿kali)-[~]
$ cat imagefilelog

dc3dd 7.2.646 started at 2024-01-15 04:05:12 -0500
compiled options:
command line dc3dd if=/dev/sdb1 of=pd_image1.dd hash=md5 log=imagefilel
device size: 7913344 sectors (probed), 4,051,632,128 bytes
sector size: 512 bytes (probed)
1835008 bytes ( 1.8 M ) copied ( 0% ), 10.0958 s, 177 K/s

input results for device `/dev/sdb1':
3584 sectors in
0 bad sectors replaced by zeros
e3268e3fd5596561cd0bd5719963cdf2 (md5)

output results for file `pd_image1.dd':
3584 sectors out

dc3dd aborted at 2024-01-15 04:05:22 -0500
```

```
(kali㉿kali)-[~]
$ sudo dc3dd if=/dev/sdb1 of=pd_image1.dd hash=md5 log=imagefilelog
```

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

```
(kali㉿kali)-[~]
$ sudo dc3dd if=/dev/sdb1 of=pd_image1.dd hash=md5 log=imagefilelog
ofsz=5M hofs=hashsplit.0000
```

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

```
(kali㉿kali)-[~]
$ hashdeep -h
hashdeep version 4.4 by Jesse Kornblum and Simson Garfinkel.
$ hashdeep [OPTION]... [FILES]...
-c <alg1,[alg2]> - Compute hashes only. Defaults are MD5 and SHA-256
                    legal values: md5,sha1,sha256,tiger,whirlpool,
-p <size> - piecewise mode. Files are broken into blocks for hashing
-r - recursive mode. All subdirectories are traversed
```

- 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:

```
(kali㉿kali)-[~]
└─$ ls -l
total 13476
-rw-r--r-- 1 kali kali 327680 Jan 15 04:02 combined1
drwxr-xr-x 2 kali kali 4096 Jan 9 09:09 Desktop
drwxr-xr-x 2 kali kali 4096 Jan 9 09:09 Documents
drwxr-xr-x 2 kali kali 4096 Jan 9 09:09 Downloads
-rw-r--r-- 1 root root 5242880 Jan 15 04:10 hashsplit.0000
-rw-r--r-- 1 root root 1310720 Jan 15 04:11 hashsplit.0001
-rw-r--r-- 1 root root 1962 Jan 15 04:11 imagefilelog
drwxr-xr-x 2 kali kali 4096 Jan 9 09:09 Music
-rw-r--r-- 1 root root 6553600 Jan 15 04:11 pd_image1.dd
drwxr-xr-x 2 kali kali 4096 Jan 9 09:09 Pictures
drwxr-xr-x 2 kali kali 4096 Jan 9 09:09 Public
-rw-r--r-- 1 root root 327680 Jan 15 04:02 split.0000
drwxr-xr-x 2 kali kali 4096 Jan 9 09:09 Templates
drwxr-xr-x 2 kali kali 4096 Jan 9 09:09 Videos
```

- 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 everything that dc3dd does and is a little bit more interactive

```
(kali㉿kali)-[~]
└─$ sudo dd if=/dev/sdb of=pd_image3.dd bs=512 conv=noerror,sync
```

- 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:

```
(kali㉿kali)-[~]
└─$ sudo dd if=/dev/sdb of=pd_image3.dd bs=512 conv=noerror,sync
^C18145+0 records in
18144+0 records out
9289728 bytes (9.3 MB, 8.9 MiB) copied, 26.98 s, 344 kB/s

(kali㉿kali)-[~]
└─$ ls -l
total 9104
drwxr-xr-x 2 kali kali 4096 Jan 9 09:09 Desktop
drwxr-xr-x 2 kali kali 4096 Jan 9 09:09 Documents
drwxr-xr-x 2 kali kali 4096 Jan 9 09:09 Downloads
drwxr-xr-x 2 kali kali 4096 Jan 9 09:09 Music
-rw-r--r-- 1 root root 9289728 Jan 15 04:22 pd_image3.dd
```

- Our pd_image 3 has been created successfully


```
(kali㉿kali)-[~]
$ sudo dd if=/dev/sdb of=mbr bs=512 count=1
1+0 records in
1+0 records out
512 bytes copied, 0.188478 s, 2.7 kB/s
```

- 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

```
(kali㉿kali)-[~]
$ xxd mbr
00000000: 33c0 8ed0 bc00 7c8e c08e d8be 007c bf00 3.....|.....|..
00000010: 06b9 0002 fcf3 a450 681c 06cb fbb9 0400 .. ....Ph.....
00000020: bdb0 0780 7e00 007c 0b0f 850e 0183 c510 .....~..|.....
00000030: e2f1 cd18 8856 0055 c646 1105 c646 1000 .....V.U.F...F..
00000040: b441 bbaa 55cd 135d 720f 81fb 55aa 7509 .A..U..]r...U.u.
00000050: f7c1 0100 7403 fe46 1066 6080 7e10 0074 ... .t..F.f`.~..t
00000060: 2666 6800 0000 0066 ff76 0868 0000 6800 8fh....f.v.h..h.
00000070: 7c68 0100 6810 00b4 428a 5600 8bf4 cd13 |h..h...B.V.....
00000080: 9f83 c410 9eeb 14b8 0102 bb00 7c8a 5600 .....|..V.
00000090: 8a76 018a 4e02 8a6e 03cd 1366 6173 1cfe .v..N..n...fas..
000000a0: 4e11 750c 807e 0080 0f84 8a00 b280 eb84 N.u..~.....
000000b0: 5532 e48a 5600 cd13 5deb 9e81 3efe 7d55 U2..V..]...>}.U
000000c0: aa75 6eff 7600 e88d 0075 17fa b0d1 e664 .un.v...u.....d
000000d0: e883 00b0 dfe6 60e8 7c00 b0ff e664 e875 .. ...`|. ...d.u
000000e0: 00fb b800 bbcd 1a66 23c0 753b 6681 fb54 . ... f#.u;f..T
000000f0: 4350 4175 3281 f902 0172 2c66 6807 bb00 CPAu2....r,fh..
00000100: 0066 6800 0200 0066 6808 0000 0066 5366 .fh...fh...fSf
00000110: 5366 5566 6800 0000 0066 6800 7c00 0066 SfUfh....fh.|..f
00000120: 6168 0000 07cd 1a5a 32f6 ea00 7c00 00cd ah.. ...Z2..|.
00000130: 18a0 b707 eb08 a0b6 07eb 03a0 b507 32e4 .....2.
00000140: 0500 078b f0ac 3c00 7409 bb07 00b4 0ecd .....<.t...
00000150: 10eb f2f4 ebfd 2bc9 e464 eb00 2402 e0f8 .....+..d..$...
00000160: 2402 c349 6e76 616c 6964 2070 6172 7469 $..Invalid parti
00000170: 7469 6f6e 2074 6162 6c65 0045 7272 6f72 tion table.Error
00000180: 206c 6f61 6469 6e67 206f 7065 7261 7469 loading operati
00000190: 6e67 2073 7973 7465 6d00 4d69 7373 696e ng system.Missin
000001a0: 6720 6f70 6572 6174 696e 6720 7379 7374 g operating syst
000001b0: 656d 0000 0063 7b9a 0000 0000 0000 8002 em...c{.....
000001c0: 0300 0bfe 7feb 8000 0000 80bf 7800 0000 ..... ..x...
000001d0: 0000 0000 0000 0000 0000 0000 0000 0000 .....
000001e0: 0000 0000 0000 0000 0000 0000 0000 0000 .....
000001f0: 0000 0000 0000 0000 0000 0000 0000 55aa .....U.
```

- 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

```
(kali㉿kali)-[~]
$ sudo fdisk
fdisk: bad usage
Try 'fdisk --help' for more information.

(kali㉿kali)-[~]
$ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda           8:0    0 80.1G  0 disk
└─sda1        8:1    0 80.1G  0 part /
sdb           8:16   1  3.8G  0 disk
└─sdb1        8:17   1  3.8G  0 part
sr0          11:0    1 1024M  0 rom

(kali㉿kali)-[~]
$ lsusb
Bus 001 Device 003: ID 8564:1000 Transcend Information, Inc. JetFlash
Bus 001 Device 002: ID 80ee:0021 VirtualBox USB Tablet
Bus 001 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub

(kali㉿kali)-[~]
$ fusblock
fusblock: command not found

(kali㉿kali)-[~]
$ fusbk
Command 'fusbk' not found, did you mean:
  command 'lsblk' from deb util-linux
Try: sudo apt install <deb name>

(kali㉿kali)-[~]
$
```

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 ~/foremost1/** to carve out the files by file type

```
(kali㉿kali)-[~]
└─$ sudo nano foremost -i pd_image.dd -o ~/foremost1/

(kali㉿kali)-[~]
└─$ sudo foremost -i pd_image.dd -o ~/foremost1/
Processing: pd_image.dd
|foundat=_rels/.rels ♦(♦
***|

(kali㉿kali)-[~]
└─$ ls
Desktop  Documents  Downloads  foremost  foremost1  Music  pd_image.dd  Pictures  Public  Templates  Videos

(kali㉿kali)-[~]
└─$ cd foremost1

(kali㉿kali)-[~/foremost1]
└─$ ls
audit.txt  docx  exe  jar  jpg  pdf
```

- 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**

```
(kali㉿kali)-[/etc/scalpel]
└─$ ls
scalpel.conf

(kali㉿kali)-[/etc/scalpel]
└─$ nano scalpel.conf

(kali㉿kali)-[/etc/scalpel]
└─$ sudo nano scalpel.conf

(kali㉿kali)-[/etc/scalpel]
└─$ cd ~

(kali㉿kali)-[~]
└─$ sudo scalpel -c /etc/scalpel/scalpel.conf -o scalout1 pd_image.dd
Scalpel version 1.60
Written by Golden G. Richard III, based on Foremost 0.69.

Opening target "/home/kali/pd_image.dd"
```

- We get an output like this:


```

File Actions Edit View Help
└─$ sudo scalpel -c /etc/scalpel/scalpel.conf -o scalout1 pd_image.dd
Scalpel version 1.60
Written by Golden G. Richard III, based on Foremost 0.69.

Opening target "/home/kali/pd_image.dd"

Image file pass 1/2.
pd_image.dd: 100.0% |*****
Allocating work queues...
Work queues allocation complete. Building carve lists...
Carve lists built. Workload:
gif with header "\x47\x49\x46\x38\x37\x61" and footer "\x00\x3b" → 0 files
gif with header "\x47\x49\x46\x38\x39\x61" and footer "\x00\x3b" → 0 files
jpg with header "\xff\xd8\xff\x3f\x3f\x45\x78\x69\x66" and footer "\xff\xd9" → 4 files
jpg with header "\xff\xd8\xff\x3f\x3f\x4a\x46\x49\x46" and footer "\xff\xd9" → 27 files
Carving files from image.
Image file pass 2/2.
pd_image.dd: 100.0% |*****
Processing of image file complete. Cleaning up...
Done.
Scalpel is done, files carved = 31, elapsed = 2 seconds.

```

- To view the images: "feh *"

TOOL 3: BULK EXTRACTOR

```

(kali㉿kali)-[~/scalout1/jpg-3-0]
└─$ sudo apt-get install bulk-extractor
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages will be upgraded:
  bulk-extractor
1 upgraded, 0 newly installed, 0 to remove and 921 not up
Need to get 11.3 MB of archives.
After this operation, 4,003 kB disk space will be freed.
Get:1 http://kali.download/kali kali-rolling/non-free amd64 bulk-extractor 2.0.6-0kali1 amd64 11.3 MB
Fetched 11.3 MB in 5s (2,304 kB/s)
(Reading database ... 400061 files and directories current
Preparing to unpack .../bulk-extractor_2.0.6-0kali1_amd64.deb
Unpacking bulk-extractor (2.0.6-0kali1) over (2.0.3-0kali1)
Setting up bulk-extractor (2.0.6-0kali1) ...
Processing triggers for kali-menu (2023.4.6) ...
Processing triggers for man-db (2.12.0-1) ...

```

- To get the output:

```

(kali㉿kali)-[~]
└─$ sudo bulk_extractor -o bulkout pd_image.dd
bulk_extractor version: 2.0.6

```

TOOL 4: MAGICRESCUE

```

(kali㉿kali)-[~/bulkout]
└─$ sudo apt-get install magicrescue
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
magicrescue is already the newest version (1.1.10+dfsg-2).
magicrescue set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 921 not upgraded.

```



```

(kali㉿kali)-[~/bulkout]
$ cd ~

(kali㉿kali)-[~]
$ cd /usr/share/magicrescue

(kali㉿kali)-[/usr/share/magicrescue]
$ ls
recipes

(kali㉿kali)-[/usr/share/magicrescue]
$ cd recipes

(kali㉿kali)-[/usr/share/magicrescue/recipes]
$ ls
avi      elf      flac    gpl     jpeg-exif  mbox      mbox-mozilla-sent  mp3-id3v2  nikon-r
canon-cr2  empathy  flv     gzip    jpeg-jfif  mbox-mozilla-inbox  mp3-id3v1      msoffice   perl

```

```

(kali㉿kali)-[/usr/share/magicrescue/recipes]
$ cat jpeg-exif
# Extracts jpeg files with the Exif magic bytes. These usually originate from
# digital camaras or other devices.
# Depends on jpegtran from libjpeg: http://freshmeat.net/projects/libjpeg/
# See also jpeg-jfif
6 string Exif
0 int32 ffd80000 ffff0000
extension jpg
command jpegtran -copy all -outfile "$1"

```

- 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

```

(kali㉿kali)-[~]
$ mkdir magicout

(kali㉿kali)-[~]
$ mkdir magicout
mkdir: cannot create directory 'magicout': File exists

(kali㉿kali)-[~]
$ sudo magicrescue -r jpeg-exif -d magicout -M o pd_image.dd
Found jpeg-exif at 0x10A85000

magicout/000010A85000-0.jpg
Found jpeg-exif at 0x10E2E000
magicout/000010E2E000-0.jpg
Found jpeg-exif at 0x10EAA3E7
magicout/000010EAA3E7-0.jpg
Scanning pd_image.dd finished at 284MB

```

- To extract information:

```
(kali㉿kali)-[~/foremost1/jpg]
$ feh 00017376.jpg

(kali㉿kali)-[~/foremost1/jpg]
$ file 00017376.jpg
00017376.jpg: JPEG image data, JFIF standard 1.01
```

TOOL 5: BINWALK- TO EXTRACT BINARIES

- **Binwalk -e 00017376.jpg**

```
(kali㉿kali)-[~/foremost1/jpg]
$ binwalk -e 00017376.jpg
```

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	JPEG image data, JFIF standard 1.01

- 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**

```
(kali㉿kali)-[~/foremost1/jpg]
$ strings 00017376.jpg
JFIF
$3br
%&'()*456789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz
#3R
&'()*56789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz
<[o%
14

(kali㉿kali)-[~/foremost1/jpg]
$ exiftool 00017376.jpg
ExifTool Version Number      : 12.67
File Name                     : 00017376.jpg
Directory                     : .
File Size                     : 32 kB
File Modification Date/Time   : 2024:01:22 03:56:11-05:00
File Access Date/Time        : 2024:01:22 04:01:19-05:00
File Inode Change Date/Time   : 2024:01:22 03:56:11-05:00
File Permissions              : -rw-r--r--
File Type                     : JPEG
File Type Extension           : jpg
MIME Type                     : image/jpeg
JFIF Version                  : 1.01
Resolution Unit                : inches
```

- 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

```
(kali㉿kali)-[~]  
$ sudo apt-get install sleuthkit  
[sudo] password for kali:
```

IMG_STAT: INFO ABOUT THE IMAGE

```
(kali㉿kali)-[~]  
$ img_stat pd_image.dd  
IMAGE FILE INFORMATION  
-----  
Image Type: raw  
  
Size in bytes: 298156032  
Sector size: 512
```

MMSTAT

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

```
(kali㉿kali)-[~]  
$ mmstat pd_image.dd  
dos
```

Dos is the type of sector

TYPES OF FORMATS SUPPORTED BY MMSTAT?

- Types of formats supported by mmstat

```
(kali㉿kali)-[~]
$ mmstat -i list
Supported image format types:
    raw (Single or split raw file (dd))
    aff (Advanced Forensic Format)
    afd (AFF Multiple File)
    afm (AFF with external metadata)
    afflib (All AFFLIB image formats (including beta ones))
    ewf (Expert Witness Format (EnCase))
    vmdk (Virtual Machine Disk (VmWare, Virtual Box))
    vhd (Virtual Hard Drive (Microsoft))
```

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

```
(kali㉿kali)-[~]
$ mmstat -t list
Supported partition types:
    dos (DOS Partition Table)
    mac (MAC Partition Map)
    bsd (BSD Disk Label)
    sun (Sun Volume Table of Contents (Solaris))
    gpt (GUID Partition Table (EFI))
```

Gpt is a new type of format

STEP 2: EXTRACTING SECTORS

- Now we will extract the sectors
- Sudo dd if=pd_image.dd of=mbr bs=512 count=1

```
(kali㉿kali)-[~]
$ sudo dd if=pd_image.dd of=mbr bs=512 count=1
1+0 records in
1+0 records out
512 bytes copied, 0.000264931 s, 1.9 MB/s
```

- Now reading the partition: xxd mbr


```

(kali㉿kali)-[~]
$ xxd mbr
00000000: 33c0 8ed0 bc00 7c8e c08e d8be 007c bf00 3.....|.....|..
00000010: 06b9 0002 fcf3 a450 681c 06cb fbb9 0400 ..Ph.....
00000020: bdb0 0780 7e00 007c 0b0f 850e 0183 c510 ....~..|.....
00000030: e2f1 cd18 8856 0055 c646 1105 c646 1000 ....V.U.F...F..
00000040: b441 bbaa 55cd 135d 720f 81fb 55aa 7509 .A..U..]r...U.u.
00000050: f7c1 0100 7403 fe46 1066 6080 7e10 0074 ...t..F.f`.~..t
00000060: 2666 6800 0000 0066 ff76 0868 0000 6800 &fh....f.v.h..h.
00000070: 7c68 0100 6810 00b4 428a 5600 8bf4 cd13 |h..h...B.V.....
00000080: 9f83 c410 9eeb 14b8 0102 bb00 7c8a 5600 .....|.V.
00000090: 8a76 018a 4e02 8a6e 03cd 1366 6173 1cfe .v..N..n...fas..
000000a0: 4e11 750c 807e 0080 0f84 8a00 b280 eb84 N.u..~.....
000000b0: 5532 e48a 5600 cd13 5deb 9e81 3efe 7d55 U2..V..]...>}.U
000000c0: aa75 6eff 7600 e88d 0075 17fa b0d1 e664 .un.v...u....d
000000d0: e883 00b0 dfa6 60e8 7c00 b0ff e664 e875 ..l...d..u

```

- 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

```

(kali㉿kali)-[~]
$ mmls pd_image.dd
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

Slot      Start      End      Length    Description
000:  Meta      0000000000 0000000000 0000000001 Primary Table (#0)
001:  _____ 0000000000 00000000127 00000000128 Unallocated
002:  000:000 00000000128 0007913471 0007913344 Win95 FAT32 (0x0b)

```

- 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 starting of the pendrive in pd_image.dd

```

(kali@kali)-[~]
└─$ fsstat -o 128 pd_image.dd
FILE SYSTEM INFORMATION
-----
File System Type: FAT32

OEM Name: MSDOS5.0
Volume ID: 0x9a07a8ad
Volume Label (Boot Sector): NO NAME
Volume Label (Root Directory): SHREEYA
File System Type Label: FAT32
Next Free Sector (FS Info): 17240
Free Sector Count (FS Info): 7896072

Sectors before file system: 128

File System Layout (in sectors)

```

-
- 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

FAT CONTENTS (in sectors)

```

16384-16391 (8) → EOF
16392-16399 (8) → 17040
16400-16407 (8) → EOF
16408-16527 (120) → EOF
16528-16535 (8) → EOF
16536-16543 (8) → EOF
16544-16551 (8) → EOF
16552-16559 (8) → EOF
16560-16567 (8) → EOF
16568-16575 (8) → EOF
16576-16583 (8) → EOF
16584-16591 (8) → EOF
16592-16599 (8) → EOF
16600-16607 (8) → EOF

```

-
- From the contents we can see that the data has been stored in a "sparse format" that is not all data is stored in one place 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

```
(kali㉿kali)-[~]
$ fls -o 128 pd_image.dd
r/r 3: SHREEYA (Volume Label Entry)
d/d * 4: _fi
d/d 7: System Volume Information
r/r * 24: Angrezi Medium (2020) Hindi.mkv
r/r * 28: The Tashkent Files (2019).mkv
r/r * 30: Kahaani.avi
r/r * 33: Jab We Met (2007) 3958.mkv
d/d * 35: IMAGES(JB)
r/r 40: ISC_PROJECT_2020_XII_Shreeya_Chatterji.zip
d/d 44: ISC_PROJECT_2020_XII-copy-copy
r/r * 47: SHUBHASHISH PIC.docx
r/r * 48: _WRD2483.tmp
r/r * 51: SHUBHASHISH PIC.docx
r/r * 54: Shubhashish PIC.jpg
r/r * 57: Shubhashish PIC.jpg
v/v 0052187: $MPP
```

-
- Eg: r/r 3: shreeya-
 - r/r – regular file
 - d/d directory
 - 3: block number

- To recover deleted files:

```
(kali㉿kali)-[~]
$ fls -d -o 128 pd_image.dd
d/d * 4: _fi
r/r * 24: Angrezi Medium (2020) Hindi.mkv
r/r * 28: The Tashkent Files (2019).mkv
r/r * 30: Kahaani.avi
r/r * 33: Jab We Met (2007) 3958.mkv
d/d * 35: IMAGES(JB)
r/r * 47: SHUBHASHISH PIC.docx
r/r * 48: _WRD2483.tmp
r/r * 51: SHUBHASHISH PIC.docx
r/r * 54: Shubhashish PIC.jpg
r/r * 57: Shubhashish PIC.jpg
```

○

- To recover the files:

```
(kali㉿kali)-[~]
$ icat -r 128 pd_image.dd 33
```

- Icat -r(recover) blockstart image block number

- Tsk_recover

```
(kali㉿kali)-[~]
$ tsk_recover -h
tsk_recover: invalid option -- 'h'
Invalid argument: (null)
usage: tsk_recover [-vVae] [-f fstype] [-i imgtype] [-b dev_sector_size]
o sector_offset] [-P pooltype] [-B pool_volume_block] [-d dir_inum] ima
image] output_dir
-i imgtype: The format of the image file (use '-i list' for supported
s)
-b dev_sector_size: The size (in bytes) of the device sectors
-f fstype: The file system type (use '-f list' for supported types)
-v verbose output to stderr
```

```
(kali@kali)-[~]
$ tsk_recover -e -o 128 pd_image.dd ~/recoverout/
Error writing file: /home/kali/recoverout///Angrezi Medium (
```

- WinHex

LAB 5- TESTDISK

Testdisk

- sudo apt-get install testdisk
- sudo testdisk ->create -> pendrive->intel->analyze
- sudo testdisk pd_image1.dd for working with the image file instead 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:**

```
(kali@kali)-[~/Desktop]
$ git clone https://github.com/volatilityfoundation/volatility.git
Cloning into 'volatility' ...
remote: Enumerating objects: 27411, done.
Receiving objects: 10% (2875/27411), 220.01 KiB | 362.00 KiB/s
```

- 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.


```

(kali@kali)-[~/Desktop]
$ git clone https://github.com/volatilityfoundation/volatility.git
Cloning into 'volatility' ...
remote: Enumerating objects: 27411, done.
remote: Total 27411 (delta 0), reused 0 (delta 0), pack-reused 27411
Receiving objects: 100% (27411/27411), 21.10 MiB | 1.78 MiB/s, done.
Resolving deltas: 100% (19758/19758), done.

(kali@kali)-[~/Desktop]
$ ls
pd_image.dd  volatility

(kali@kali)-[~/Desktop]
$ cd volatility

(kali@kali)-[~/Desktop/volatility]
$ ls
AUTHORS.txt  contrib  LEGAL.txt  Makefile  PKG-INFO  pyinstaller.spec  resources  tools  vol.py
CHANGELOG.txt  CREDITS.txt  LICENSE.txt  MANIFEST.in  pyinstaller  README.txt  setup.py  volatility

```

- 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

```

(kali@kali)-[~/Desktop/volatility]
$ python2 vol.py -f cridex.vmem imageinfo
Volatility Foundation Volatility Framework 2.6.1
*** Failed to import volatility.plugins.registry.shutdown (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.getservicesids (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.timeliner (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.malware.apihooks (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.malware.servicediff (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.userassist (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.getsids (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.tcaudit (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.shellbags (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.evtxlogs (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.shimcache (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.lsadump (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.malware.threads (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.mac.apihooks_kernel (ImportError: No module named distorm3)
*** Failed to import volatility.plugins.registry.amcache (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.mac.check_syscall_shadow (ImportError: No module named distorm3)
*** Failed to import volatility.plugins.malware.svcscan (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.auditpol (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.ssd (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.registry.registryapi (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.mac.apihooks (ImportError: No module named distorm3)
*** Failed to import volatility.plugins.envvars (ImportError: No module named Crypto.Hash)
INFO : volatility.debug : Determining profile based on KDBG search...
      Suggested Profile(s): WinXPSP2x86, WinXPSP3x86 (Instantiated with WinXPSP2x86)
      AS Layer1: IA32PagedMemoryPae (Kernel AS)
      AS Layer2: FileAddressSpace (/home/kali/Desktop/volatility/cridex.vmem)
      PAE type: PAE
      DTB: 0x2fe000L
      KDBG: 0x80545ae0L
      Number of Processors: 1
      Image Type (Service Pack): 3
      KPCR for CPU 0: 0xffdff000L
      KUSER_SHARED_DATA: 0xffdf0000L
      Image date and time: 2012-07-22 02:45:08 UTC+0000
      Image local date and time: 2012-07-21 22:45:08 -0400

```

-
- **Imginfo:** gives info about the profile of the os of the
- **Cridex:** it is the malware or trojan that we are analysing
- **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
- Wuauclt.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

```
(kali@kali)-[~/Desktop/volatility]
$ python2 vol.py -f cridex.vmem pslist
```

Offset(V)	Name	PID	PPID	Thds	Hnds	Sess	Wow64	Start
0x823c89c8	System	4	0	53	240	—	0	
0x822f1020	smss.exe	368	4	3	19	—	0	2012-07-22 02:42:31 UTC+0000
0x822a0598	csrss.exe	584	368	9	326	0	0	2012-07-22 02:42:32 UTC+0000
0x82298700	winlogon.exe	608	368	23	519	0	0	2012-07-22 02:42:32 UTC+0000
0x81e2ab28	services.exe	652	608	16	243	0	0	2012-07-22 02:42:32 UTC+0000
0x81e2a3b8	lsass.exe	664	608	24	330	0	0	2012-07-22 02:42:32 UTC+0000
0x82311360	svchost.exe	824	652	20	194	0	0	2012-07-22 02:42:33 UTC+0000
0x81e29ab8	svchost.exe	908	652	9	226	0	0	2012-07-22 02:42:33 UTC+0000
0x823001d0	svchost.exe	1004	652	64	1118	0	0	2012-07-22 02:42:33 UTC+0000
0x821dfda0	svchost.exe	1056	652	5	60	0	0	2012-07-22 02:42:33 UTC+0000
0x82295650	svchost.exe	1220	652	15	197	0	0	2012-07-22 02:42:35 UTC+0000
0x821dea70	explorer.exe	1484	1464	17	415	0	0	2012-07-22 02:42:36 UTC+0000
0x81eb17b8	spoolsv.exe	1512	652	14	113	0	0	2012-07-22 02:42:36 UTC+0000
0x81e7bda0	reader_sl.exe	1640	1484	5	39	0	0	2012-07-22 02:42:36 UTC+0000
0x820e8da0	alg.exe	788	652	7	104	0	0	2012-07-22 02:43:01 UTC+0000
0x821fcda0	wuauclt.exe	1136	1004	8	173	0	0	2012-07-22 02:43:46 UTC+0000
0x8205bda0	wuauclt.exe	1588	1004	5	132	0	0	2012-07-22 02:44:01 UTC+0000

- 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

```
(kali@kali)-[~/Desktop/volatility]
$ python2 vol.py -f cridex.vmem pstree
```

*** Failed to import volatility.plugins.envvars (ImportError: No module named Crypto.Hash)

Name	Pid	PPid	Thds	Hnds	Time
0x823c89c8:System	4	0	53	240	1970-01-01 00:00:00 UTC+0000
0x822ff1020:smss.exe	368	4	3	19	2012-07-22 02:42:31 UTC+0000
0x82298700:winlogon.exe	608	368	23	519	2012-07-22 02:42:32 UTC+0000
0x81e2ab28:services.exe	652	608	16	243	2012-07-22 02:42:32 UTC+0000
0x821dfda0:svchost.exe	1056	652	5	60	2012-07-22 02:42:33 UTC+0000
0x81eb17b8:spoolsv.exe	1512	652	14	113	2012-07-22 02:42:36 UTC+0000
0x81e29ab8:svchost.exe	908	652	9	226	2012-07-22 02:42:33 UTC+0000
0x823001d0:svchost.exe	1004	652	64	1118	2012-07-22 02:42:33 UTC+0000
0x8205bda0:wuauc.lt.exe	1588	1004	5	132	2012-07-22 02:44:01 UTC+0000
0x821fcda0:wuauc.lt.exe	1136	1004	8	173	2012-07-22 02:43:46 UTC+0000
0x82311360:svchost.exe	824	652	20	194	2012-07-22 02:42:33 UTC+0000
0x820e8da0:alg.exe	788	652	7	104	2012-07-22 02:43:01 UTC+0000
0x82295650:svchost.exe	1220	652	15	197	2012-07-22 02:42:35 UTC+0000
0x81e2a3b8:lsass.exe	664	608	24	330	2012-07-22 02:42:32 UTC+0000
0x822a0598:csrss.exe	584	368	9	326	2012-07-22 02:42:32 UTC+0000
0x821dea70:explorer.exe	1484	1464	17	415	2012-07-22 02:42:36 UTC+0000
0x81e7bda0:reader_sl.exe	1640	1484	5	39	2012-07-22 02:42:36 UTC+0000

- 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

```
(kali@kali)-[~/Desktop/volatility]
$ python2 vol.py -f cridex.vmem psxview
```

Volatility: Foundation Volatility Framework (ImportError: No module named Crypto.Hash)

Offset(P)	Name	PID	pslist	psscan	thrdproc	pspcid	csrss	session	deskthrd	ExitTime
0x02498700	winlogon.exe	608	True	True	True	True	True	True	True	
0x02511360	svchost.exe	824	True	True	True	True	True	True	True	
0x022e8da0	alg.exe	788	True	True	True	True	True	True	True	
0x020b17b8	spoolsv.exe	1512	True	True	True	True	True	True	True	
0x0202ab28	services.exe	652	True	True	True	True	True	True	True	
0x02495650	svchost.exe	1220	True	True	True	True	True	True	True	
0x0207bda0	reader_sl.exe	1640	True	True	True	True	True	True	True	
0x025001d0	svchost.exe	1004	True	True	True	True	True	True	True	
0x02029ab8	svchost.exe	908	True	True	True	True	True	True	True	
0x023fcda0	wuauc.lt.exe	1136	True	True	True	True	True	True	True	
0x0225bda0	wuauc.lt.exe	1588	True	True	True	True	True	True	True	
0x0202a3b8	lsass.exe	664	True	True	True	True	True	True	True	
0x023dea70	explorer.exe	1484	True	True	True	True	True	True	True	
0x023dfda0	svchost.exe	1056	True	True	True	True	True	True	True	
0x024f1020	smss.exe	368	True	True	True	True	False	False	False	
0x025c89c8	System	4	True	True	True	True	False	False	False	
0x024a0598	csrss.exe	584	True	True	True	True	False	True	True	

- 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

```

(kali@kali)-[~/Desktop/volatility]
$ python2 vol.py -f cridex.vmem connscan
Volatility Foundation Volatility Framework 2.6.1
*** Failed to import volatility.plugins.registry.shutdown (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.getservicesids (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.timeliner (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.malware.apihooks (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.malware.servicediff (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.userassist (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.getsids (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.tcaudit (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.shellbags (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.evtlogs (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.shimcache (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.dumpregistry (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.lsadump (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.malware.threads (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.mac.apihooks_kernel (ImportError: No module named distorm3)
*** Failed to import volatility.plugins.registry.amcache (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.mac.check_syscall_shadow (ImportError: No module named distorm3)
*** Failed to import volatility.plugins.malware.svcscan (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.auditpol (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.ssd (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.registry.registryapi (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.mac.apihooks (ImportError: No module named distorm3)
*** Failed to import volatility.plugins.envvars (ImportError: No module named Crypto.Hash)
Offset(P) Local Address Remote Address Pid
0x02087620 172.16.112.128:1038 41.168.5.140:8080 1484
0x023a8008 172.16.112.128:1037 125.19.103.198:8080 1484

```

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

```

(kali@kali)-[~/Desktop/volatility]
$ python2 vol.py -f cridex.vmem sockets
Volatility Foundation Volatility Framework 2.

```

```

*** Failed to import volatility.plugins.envvars (ImportError: No module named Crypto.Hash)
Offset(V) PID Port Proto Protocol Address Create Time
0x81ddb780 664 500 17 UDP 0.0.0.0 2012-07-22 02:42:53 UTC+0000
0x82240d08 1484 1038 6 TCP 0.0.0.0 2012-07-22 02:44:45 UTC+0000
0x81dd7618 1220 1900 17 UDP 172.16.112.128 2012-07-22 02:43:01 UTC+0000
0x82125610 788 1028 6 TCP 127.0.0.1 2012-07-22 02:43:01 UTC+0000
0x8219cc08 4 445 6 TCP 0.0.0.0 2012-07-22 02:42:31 UTC+0000
0x81ec23b0 908 135 6 TCP 0.0.0.0 2012-07-22 02:42:33 UTC+0000
0x82276878 4 139 6 TCP 172.16.112.128 2012-07-22 02:42:38 UTC+0000
0x82277460 4 137 17 UDP 172.16.112.128 2012-07-22 02:42:38 UTC+0000
0x81e76620 1004 123 17 UDP 127.0.0.1 2012-07-22 02:43:01 UTC+0000
0x82172808 664 0 255 Reserved 0.0.0.0 2012-07-22 02:42:53 UTC+0000
0x81e3f460 4 138 17 UDP 172.16.112.128 2012-07-22 02:42:38 UTC+0000
0x821f0630 1004 123 17 UDP 172.16.112.128 2012-07-22 02:43:01 UTC+0000
0x822cd2b0 1220 1900 17 UDP 127.0.0.1 2012-07-22 02:43:01 UTC+0000
0x82172c50 664 4500 17 UDP 0.0.0.0 2012-07-22 02:42:53 UTC+0000
0x821f0d00 4 445 17 UDP 0.0.0.0 2012-07-22 02:42:31 UTC+0000

```

- 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 PID 1640 which is running from the program files so our suspicion is almost fully confirmed

```

(kali@kali)-[~/Desktop/volatility]
$ python2 vol.py -f cridex.vmem cmdline
Volatility Foundation Volatility Framework 2.

```



```

*** Failed to import volatility.plugins.torvalds (ImportError: No module named Crypto.Hash)
*****
System pid:      4
*****
smss.exe pid:    368
Command line : \SystemRoot\System32\smss.exe
*****
csrss.exe pid:   584
Command line : C:\WINDOWS\system32\csrss.exe ObjectDirectory=\Windows SharedSection=1024,3072,512 Windows=On SubSystemType=Windows ServerDll=basesrv,1 ServerDll=winsrv:UserServerDllInitialization,3 ServerDll=winsrv:ConServerDllInitialization,2 ProfileControl=Off MaxRequestThreads=16
*****
winlogon.exe pid: 608
Command line : winlogon.exe
*****
services.exe pid: 652
Command line : C:\WINDOWS\system32\services.exe
*****
lsass.exe pid:   664
Command line : C:\WINDOWS\system32\lsass.exe
*****
svchost.exe pid: 824
Command line : C:\WINDOWS\system32\svchost -k DcomLaunch
*****
svchost.exe pid: 908
Command line : C:\WINDOWS\system32\svchost -k rpcss
*****
svchost.exe pid: 1004
Command line : C:\WINDOWS\System32\svchost.exe -k netsvcs
*****
svchost.exe pid: 1056
Command line : C:\WINDOWS\system32\svchost.exe -k NetworkService
*****
svchost.exe pid: 1220
Command line : C:\WINDOWS\system32\svchost.exe -k LocalService
*****
explorer.exe pid: 1484
Command line : C:\WINDOWS\Explorer.EXE
*****
spoolsv.exe pid: 1512
Command line : C:\WINDOWS\system32\spoolsv.exe
*****
reader_sl.exe pid: 1640
Command line : "C:\Program Files\Adobe\Reader 9.0\Reader\Reader_sl.exe"
*****
alg.exe pid:     788
Command line : C:\WINDOWS\System32\alg.exe
*****
wuauclt.exe pid: 1136
Command line : "C:\WINDOWS\system32\wuauclt.exe" /RunStoreAsComServer Local\[3ec]SUSDSb81eb56fa3105543beb3109274ef8ec1
*****
wuauclt.exe pid: 1588
Command line : "C:\WINDOWS\system32\wuauclt.exe"

```

- **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

```

(kali@kali)-[~/Desktop/volatility]
$ python2 vol.py -f cridex.vmem privs

```

1512 spoolsv.exe	30 SeCreateGlobalPrivilege	Present,Enabled,Default	Create global objects
1640 reader_sl.exe	23 SeChangeNotifyPrivilege	Present,Enabled,Default	Receive notifications of changes to files or directories
1640 reader_sl.exe	8 SeSecurityPrivilege	Present	Manage auditing and security log
1640 reader_sl.exe	17 SeBackupPrivilege	Present	Backup files and directories
1640 reader_sl.exe	18 SeRestorePrivilege	Present	Restore files and directories
1640 reader_sl.exe	12 SeSystemTimePrivilege	Present	Change the system time
1640 reader_sl.exe	19 SeShutdownPrivilege	Present	Shut down the system
1640 reader_sl.exe	24 SeRemoteShutdownPrivilege	Present	Force shutdown from a remote system
1640 reader_sl.exe	9 SeTakeOwnershipPrivilege	Present	Take ownership of files/objects
1640 reader_sl.exe	20 SeDebugPrivilege	Present	Debug programs
1640 reader_sl.exe	22 SeSystemEnvironmentPrivilege	Present	Edit firmware environment values
1640 reader_sl.exe	11 SeSystemProfilePrivilege	Present	Profile system performance
1640 reader_sl.exe	13 SeProfileSingleProcessPrivilege	Present	Profile a single process
1640 reader_sl.exe	14 SeIncreaseBasePriorityPrivilege	Present	Increase scheduling priority
1640 reader_sl.exe	10 SeLoadDriverPrivilege	Present,Enabled	Load and unload device drivers
1640 reader_sl.exe	15 SeCreatePagefilePrivilege	Present	Create a pagefile
1640 reader_sl.exe	5 SeIncreaseQuotaPrivilege	Present	Increase quotas
1640 reader_sl.exe	25 SeUndockPrivilege	Present,Enabled	Remove computer from docking station
1640 reader_sl.exe	28 SeManageVolumePrivilege	Present	Manage the files on a volume
1640 reader_sl.exe	29 SeImpersonatePrivilege	Present,Enabled,Default	Impersonate a client after authentication
1640 reader_sl.exe	30 SeCreateGlobalPrivilege	Present,Enabled,Default	Create global objects
788 alg.exe	21 SeAuditPrivilege	Present	Generate security audits
788 alg.exe	5 SeIncreaseQuotaPrivilege	Present	Increase quotas
788 alg.exe	3 SeAssignPrimaryTokenPrivilege	Present	Replace a process-level token
788 alg.exe	23 SeChangeNotifyPrivilege	Present,Enabled,Default	Receive notifications of changes to files or directories

- 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

```
(kali㉿kali)-[~/Desktop/volatility]
$ ls -l
total 526556
-rw-r--r-- 1 kali kali 778 Feb 12 03:39 AUTHORS.txt
-rw-r--r-- 1 kali kali 23831 Feb 12 03:39 CHANGELOG.txt
drwxr-xr-x 4 kali kali 4096 Feb 12 03:39 contrib
-rw-r--r-- 1 kali kali 3928 Feb 12 03:39 CREDITS.txt
-rwxrwxrwx 1 kali kali 536870912 Feb 12 03:55 cridex.vmem
-rw-r--r-- 1 kali kali 14336 Feb 12 04:47 executable.1004.exe
-rw-r--r-- 1 kali kali 14336 Feb 12 04:47 executable.1056.exe
-rw-r--r-- 1 kali kali 111104 Feb 12 04:47 executable.1136.exe
-rw-r--r-- 1 kali kali 14336 Feb 12 04:47 executable.1220.exe
-rw-r--r-- 1 kali kali 1033728 Feb 12 04:47 executable.1484.exe
-rw-r--r-- 1 kali kali 57856 Feb 12 04:47 executable.1512.exe
-rw-r--r-- 1 kali kali 111104 Feb 12 04:47 executable.1588.exe
-rw-r--r-- 1 kali kali 29184 Feb 12 04:47 executable.1640.exe
-rw-r--r-- 1 kali kali 50688 Feb 12 04:47 executable.368.exe
-rw-r--r-- 1 kali kali 6144 Feb 12 04:47 executable.584.exe
-rw-r--r-- 1 kali kali 507904 Feb 12 04:47 executable.608.exe
-rw-r--r-- 1 kali kali 108544 Feb 12 04:47 executable.652.exe
-rw-r--r-- 1 kali kali 13312 Feb 12 04:47 executable.664.exe
-rw-r--r-- 1 kali kali 44544 Feb 12 04:47 executable.788.exe
-rw-r--r-- 1 kali kali 14336 Feb 12 04:47 executable.824.exe
-rw-r--r-- 1 kali kali 14336 Feb 12 04:47 executable.908.exe
-rw-r--r-- 1 kali kali 698 Feb 12 03:39 LEGAL.txt
-rw-r--r-- 1 kali kali 15127 Feb 12 03:39 LICENSE.txt
-rw-r--r-- 1 kali kali 178 Feb 12 03:39 Makefile
-rw-r--r-- 1 kali kali 348 Feb 12 03:39 MANIFEST.in
-rw-r--r-- 1 kali kali 254 Feb 12 03:39 PKG-INFO
drwxr-xr-x 2 kali kali 4096 Feb 12 03:39 pyinstaller
-rw-r--r-- 1 kali kali 1007 Feb 12 03:39 pyinstaller.spec
-rw-r--r-- 1 kali kali 32041 Feb 12 03:39 README.txt
drwxr-xr-x 2 kali kali 4096 Feb 12 03:39 resources
-rw-r--r-- 1 kali kali 3606 Feb 12 03:39 setup.py
drwxr-xr-x 6 kali kali 4096 Feb 12 03:39 tools
drwxr-xr-x 5 kali kali 4096 Feb 12 04:13 volatility
-rw-r--r-- 1 kali kali 6517 Feb 12 03:39 vol.py
```

- File executable.1640.exe

```
(kali㉿kali)-[~/Desktop/volatility]
$ file executable.1640.exe
executable.1640.exe: PE32 executable (GUI) Intel 80386, for MS Windows, 4 sections
```

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

```
(kali㉿kali)-[~/Desktop/volatility]
$ md5sum executable.1640.exe
12cf6583f5a9171a1d621ae02b4eb626 executable.1640.exe
```

- Memdump -p 1640 -dump-dir
 - A dmp file is created

```
(kali㉿kali)-[~/Desktop/volatility]
$ python2 vol.py -f cridex.vmem memdump -p 1640 --dump-dir .
```

```
(kali㉿kali)-[~/Desktop/volatility]
$ ls -l
total 601952
-rw-r--r-- 1 kali kali 77205504 Feb 12 04:54 1640.dmp
-rw-r--r-- 1 kali kali 778 Feb 12 03:39 AUTHORS.txt
```

- 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

```
(kali@kali)-[~/Desktop/volatility]
$ python2 vol.py -f cridex.vmem connscan
Volatility Foundation Volatility Framework 2.6.1
*** Failed to import volatility.plugins.registry.shutdown (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.getservicesids (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.timeliner (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.malware.apihooks (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.malware.servicediff (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.userassist (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.getsids (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.tcaudit (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.shellbags (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.evlogs (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.shimcache (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.dumpregistry (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.lsdump (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.malware.threads (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.mac.apihooks_kernel (ImportError: No module named distorm3)
*** Failed to import volatility.plugins.registry.amcache (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.mac.check_syscall_shadow (ImportError: No module named distorm3)
*** Failed to import volatility.plugins.malware.svcscan (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.auditpol (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.ssd (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.registry.registryapi (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.mac.apihooks (ImportError: No module named distorm3)
*** Failed to import volatility.plugins.envvars (ImportError: No module named Crypto.Hash)
Offset(P) Local Address Remote Address Pid
0x02087620 172.16.112.128:1038 41.168.5.140:8080 1484
0x023a8008 172.16.112.128:1037 125.19.103.198:8080 1484

(kali@kali)-[~/Desktop/volatility]
$ strings 1640.dmp | grep "41.168.5.140.8080" -C 5
ABACFPFPFNFDECFCEPFHFDEFFFPACAB
DpI8
POST /zb/v_01_a/in/ HTTP/1.1
Accept: */*
User-Agent: Mozilla/5.0 (Windows; U; MSIE 7.0; Windows NT 6.0; en-US)
Host: 41.168.5.140:8080
Content-Length: 229
Connection: Keep-Alive
Cache-Control: no-cache
>mtvR
06!
```

- 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

```
(kali@kali)-[~/Desktop/volatility]
$ python2 vol.py -f cridex.vmem hivelist
Volatility Foundation Volatility Framework 2.6.1
*** Failed to import volatility.plugins.registry.shutdown (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.getservicesids (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.timeliner (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.malware.apihooks (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.malware.servicediff (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.userassist (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.getsids (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.tcaudit (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.shellbags (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.evlogs (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.shimcache (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.dumpregistry (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.lsdump (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.malware.threads (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.mac.apihooks_kernel (ImportError: No module named distorm3)
*** Failed to import volatility.plugins.registry.amcache (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.mac.check_syscall_shadow (ImportError: No module named distorm3)
*** Failed to import volatility.plugins.malware.svcscan (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.auditpol (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.ssd (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.registry.registryapi (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.mac.apihooks (ImportError: No module named distorm3)
*** Failed to import volatility.plugins.envvars (ImportError: No module named Crypto.Hash)
Virtual Physical Name
0x18e5b60 0x093f8b0 \Device\HarddiskVolume1\Documents and Settings\Robert\Local Settings\Application Data\Microsoft\Windows\UserClass.dat
0x1219b60 0x0a2a3b0 \Device\HarddiskVolume1\Documents and Settings\Robert\NTUSER.DAT
0x18398d0 0x08a8380 \Device\HarddiskVolume1\Documents and Settings\LocalService\Local Settings\Application Data\Microsoft\Windows\UserClass.dat
0x18614d0 0x08e6240 \Device\HarddiskVolume1\Documents and Settings\LocalService\NTUSER.DAT
0x183b6b0 0x08e2d80 \Device\HarddiskVolume1\Documents and Settings\NetworkService\Local Settings\Application Data\Microsoft\Windows\UserClass.dat
0x172b60 0x08519b0 \Device\HarddiskVolume1\Documents and Settings\NetworkService\NTUSER.DAT
0x1578510 0x0766950 \Device\HarddiskVolume1\WINDOWS\system32\config\software
0x1571088 0x0777f08 \Device\HarddiskVolume1\WINDOWS\system32\config\default
0x1576998 0x07669b0 \Device\HarddiskVolume1\WINDOWS\system32\config\SECURITY
0x15719e8 0x0777f98 \Device\HarddiskVolume1\WINDOWS\system32\config\SAM
0x13ba088 0x02e4b08 [no name]
0x183b6b0 0x02ac3b0 \Device\HarddiskVolume1\WINDOWS\system32\config\system
0x182e088 0x02a7d08 [no name]
```

-
- 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/Basic_of_HTTP/MIME_types/Common_types
- global settings
 - create new mime type (refer to the word doc lab 7)
 - to get signature, 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 you 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 detect, so file is encrypted)

- interesting files identifier :

- adobe creative cloud, dropbox.exe, cloudme.exe, torr, vpn etc
- crypt wallets
- encryption programs
- 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 artificial files
- looks for timelines
- parses windows registry files
- parses pe header files

- yara

- static analyzer which gives rules for 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