

Table of Contents:

Section 1: Disk imaging w/ FTK Imager

- ☐ Using FTK Imager to create a bitstream image of a USB drive
- ☐ Steps for USB drive in Raw (dd)
- ☐ Image of files created by FTK imager
- ☐ Image verification results
- ☐ MD5 & SHA1 computed & report hash
- ☐ Load USB image to FTK imager, and examine the content
- ☐ Expand partition/root to uncover and export deleted files
- ☐ Mount Image
- ☐ Obtain system files
- ☐ Export file hash list

Section 2: Imaging with dd & Netcat

- ☐ Steps for imaging over the network
- ☐ Commands & screenshots used
- ☐ receive data on port 8888 and save the received data as usd.dd
- ☐ use dd to make a full image of USB and use netcat (nc) to send the USB image
- ☐ generate both MD5 and SHA1 hashes of USB device
- ☐ generate both MD5 and SHA1 hashes of your USB image
- ☐ FTK Image verification results

Section 3: Using LiME to dump SIFT Workstation memory

- ☐ Extracting information from memory dump with Foremost
- ☐ Breaking down the Foremost Commands
- ☐ Commands used
- ☐ Directories / subdirectories created & files extracted after running Foremost
- ☐ Image_analyze directory created
- ☐ Results: Lists total number of each file type extracted
- ☐ File_analyze directory created
- ☐ Results: Lists total number of each file type extracted

PART 1 → Disk imaging w/ FTK Imager

A) Using FTK Imager to create a bitstream image of a USB drive

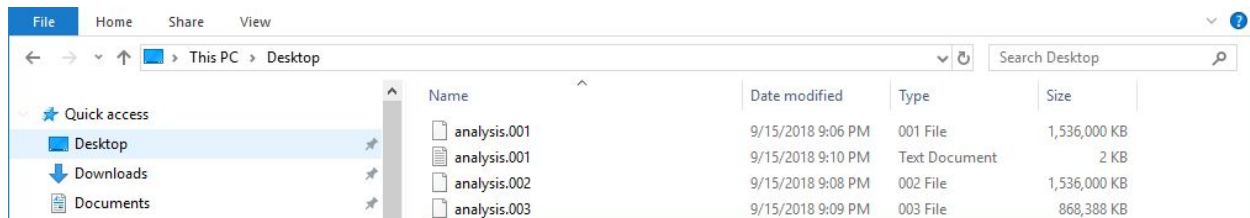
→ Steps for USB drive in Raw (dd)

Select File → Create Disk Image...
Choose Physical Drive
Choose USB Device
Press Finish
Add the image destination
Select Raw (dd) as format
Provide destination folder and image filename information
Press Start

1. After the imaging process was complete, FTK Imager created the following files..

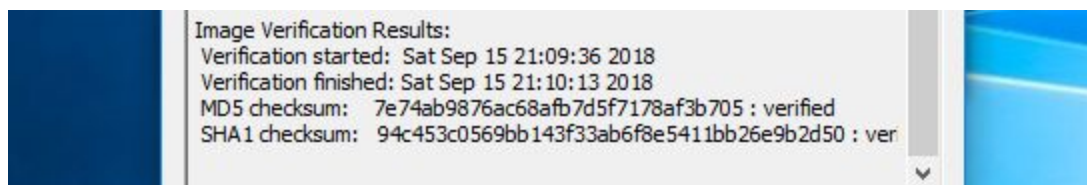
- The image file with extension of .001
- A text file for image summary

→ (Image of files created by FTK imager)



2. During the imaging process, you should have noticed that “Verify images after they are created”; is checked by default. The result is..

- FTK imager will compute the MD5 and SHA1 hashes of the USB drive and the MD5 and SHA1 hashes of the image, and verify the hashes match.



3. FTK imager created the following hash algorithms to verify the image had not been altered..

- **Four hash algorithms → A computed Hash & Report hash for MD5 & SHA1**

Name	analysis.001
Sector count	7880776
MD5 Hash	
Computed hash	7e74ab9876ac68afb7d5f7178af3b705
Report Hash	7e74ab9876ac68afb7d5f7178af3b705
Verify result	Match
SHA1 Hash	
Computed hash	94c453c0569bb143f33ab6f8e5411bb26e9b2d50
Report Hash	94c453c0569bb143f33ab6f8e5411bb26e9b2d50
Verify result	Match
Bad Sector List	
Bad sector(s)	No bad sectors found

B) Load USB image to FTK imager, and examine the content

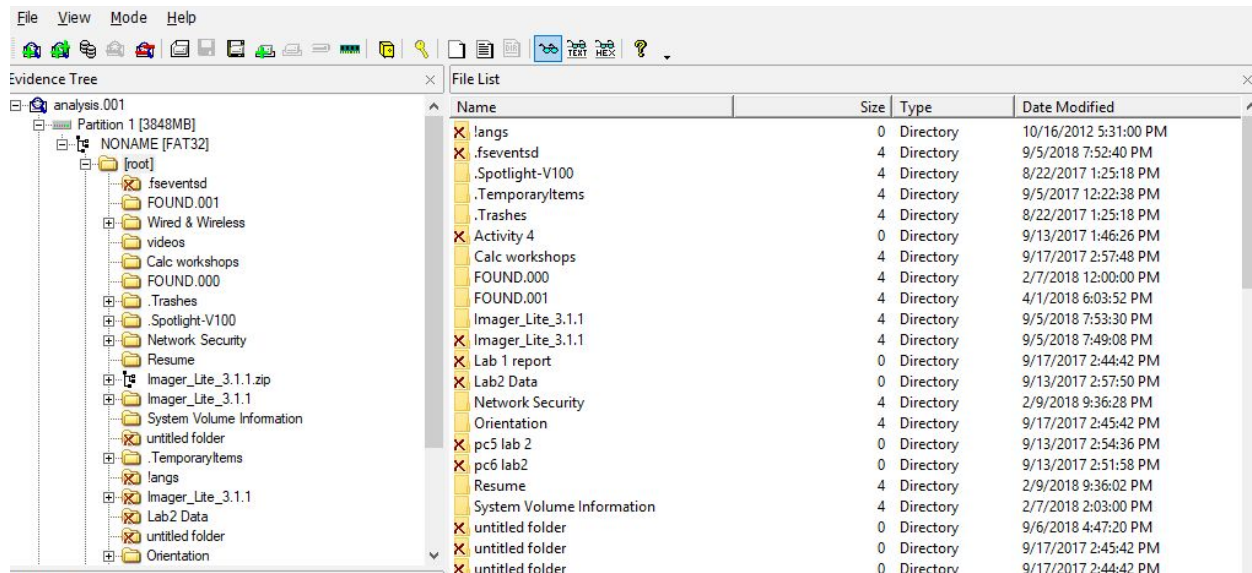
1. screenshot of the FTK imager after you have loaded your USB image

The screenshot displays the AccessData FTK Imager 3.1.1.8 interface. The 'Evidence Tree' on the left shows the loaded image 'analysis.001'. The main window displays a hex dump of the image content, with columns for Address, Hex, and ASCII. The ASCII column shows the beginning of a file system structure, including 'id partition tab', 'le>Error loading', and 'operating syste'. A 'Custom Content Sources' dialog box is open in the bottom left corner, showing 'Evidence:File System|Path|File' and 'Options'.

2. Cool FTK Imager features and screenshots.

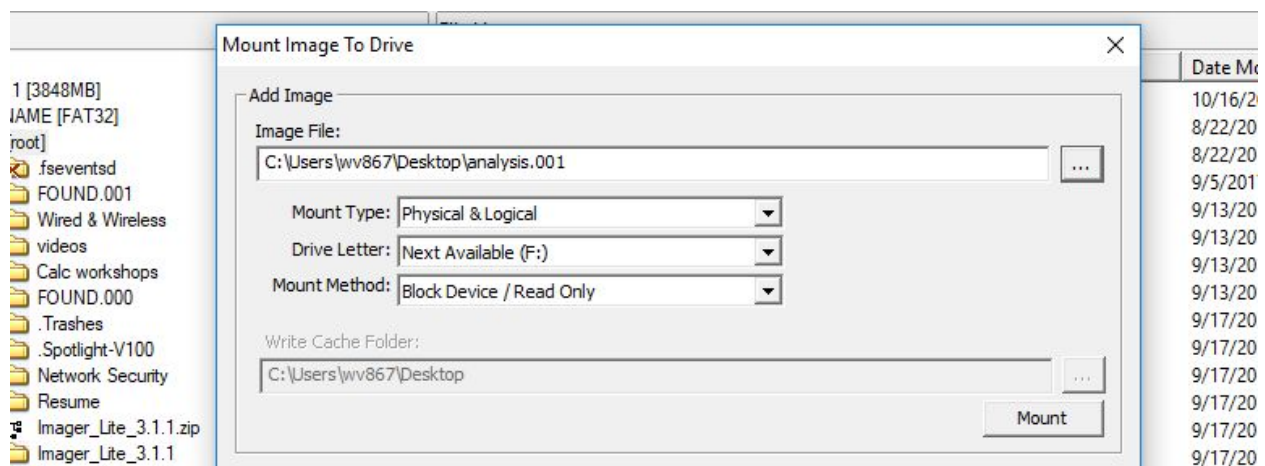
Expand partition/root to uncover and export deleted files →

- Expanding the evidence tree [analysis.001 / partition / root / ...]
- Directories & files with red x's have been deleted in the past.
- These files can be recovered by running the export command



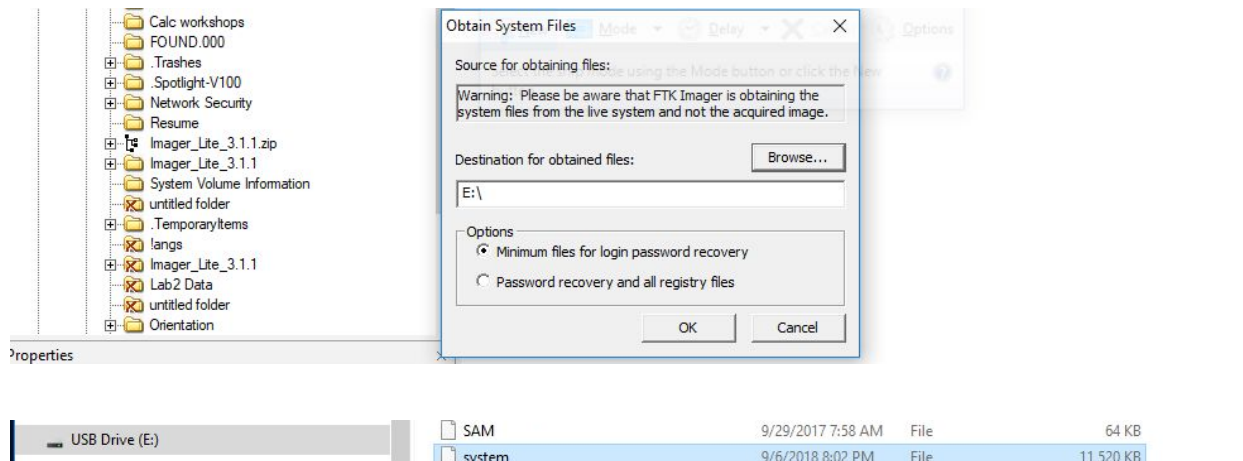
Mount Image →

- FTK Image becomes a drive letter on your machine
- Local disc f now has the same thing
- Retrieved forensically, can see things that would not have shown up originally
- Can now run files against a malware scanner (before working in files)



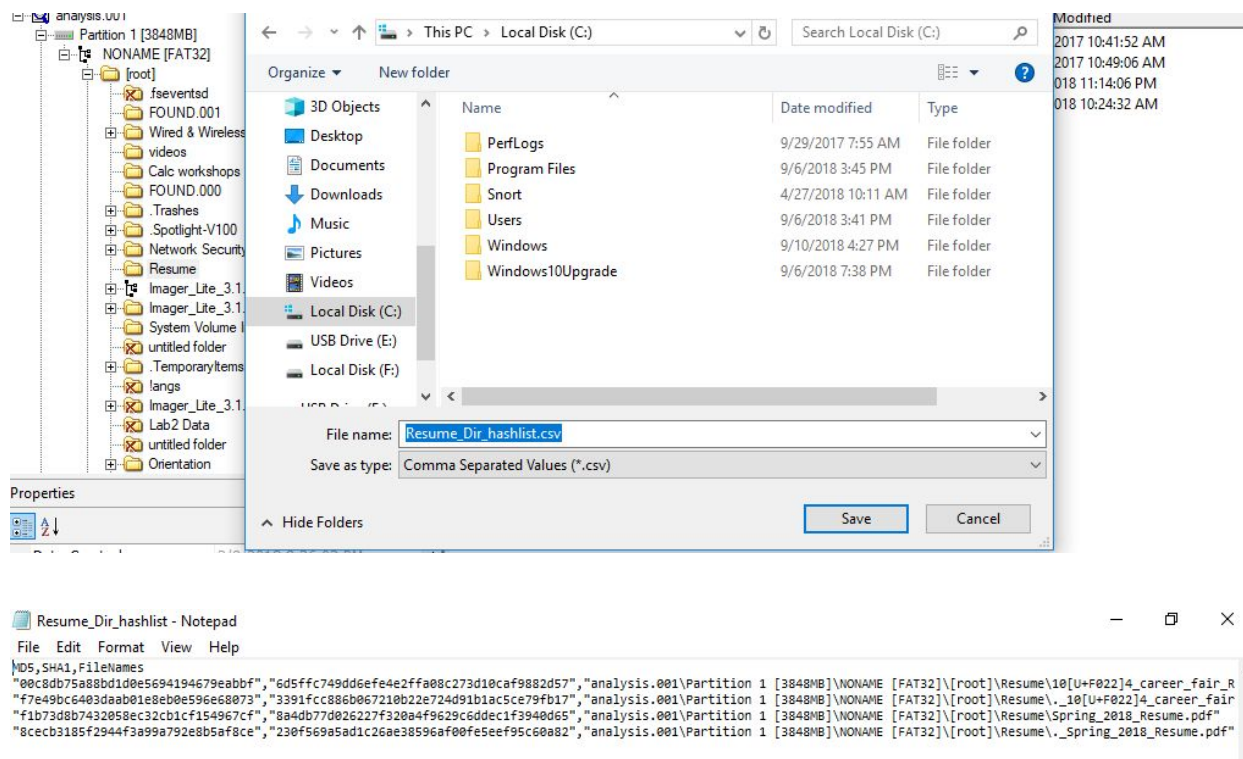
Obtain system files →

- For form/browser passwords
- In this case, SAM & System files were created
- Forensically, data stored in these files can be very helpful



Export file hash list →

- Will create a MD5 hash for everything inside of the selected directory.
- \Create a CSV file with all hashes saved in (C:)
- Hashes can be compared against a white-list for further analysis



PART 2 → Imaging with dd & Netcat

Description: **dd** and netcat (**nc**) can be used for imaging over a network. This process can be mimicked by sending a full image of a USB from one terminal(ex: **suspect machine**) to another terminal(ex: **forensic machine**).

→ Steps for imaging over the network

Create the md5 and sha1 hashes for the USB

```
md5sum /dev/sdb1  
shasum /dev/sdb1
```

Use dd to make a full image of your USB and pipe to netcat

```
nc -l 8888 > usb.dd (ex: on forensic machine)  
dd if=/dev/sdb1 | nc 127.0.0.1 8888 -w 3 (ex: on suspect machine)
```

Create the md5 and sha1 hashes for usb.dd

```
Md5sum usb.dd  
Shasum usb.dd
```

Ensure the md5 hash of the USB device matches the md5 hash of USB image

Ensure the sha1 hash of the USB device matches the sha1 hash of USB image

4. Commands & screenshots used

- Command used on the forensic machine to receive data on port 8888 and save the received data as usb.dd

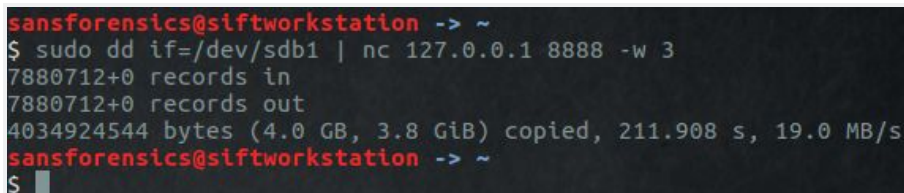
```
nc -l 8888 > usb.dd
```



```
sansforensics@siftworkstation -> ~  
$ nc -l 8888 > usb.dd  
sansforensics@siftworkstation -> ~  
$
```

- Command used on the suspect machine to use dd to make a full image of USB and use netcat (nc) to send the USB image to the forensic machine terminal

```
dd if=/dev/sdb1 | nc 127.0.0.1 8888 -w 3
```

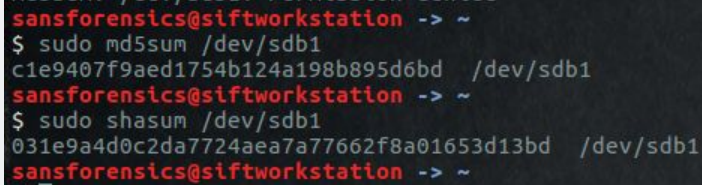


```
sansforensics@siftworkstation -> ~  
$ sudo dd if=/dev/sdb1 | nc 127.0.0.1 8888 -w 3  
7880712+0 records in  
7880712+0 records out  
4034924544 bytes (4.0 GB, 3.8 GiB) copied, 211.908 s, 19.0 MB/s  
sansforensics@siftworkstation -> ~  
$
```


- Commands used to generate both MD5 and SHA1 hashes of **USB device**

```
md5sum /dev/sdb1
```

```
shasum /dev/sdb1
```

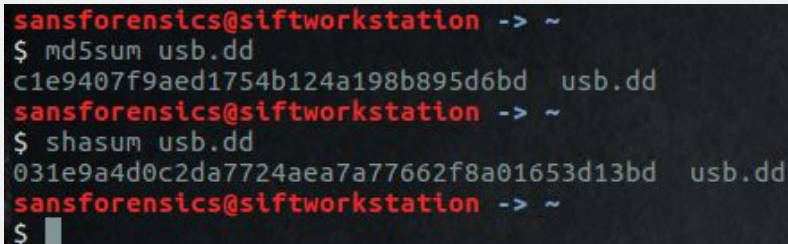


```
sansforensics@siftworkstation -> ~  
$ sudo md5sum /dev/sdb1  
c1e9407f9aed1754b124a198b895d6bd /dev/sdb1  
sansforensics@siftworkstation -> ~  
$ sudo shasum /dev/sdb1  
031e9a4d0c2da7724aea7a77662f8a01653d13bd /dev/sdb1  
sansforensics@siftworkstation -> ~
```

- Commands used to generate both MD5 and SHA1 hashes of your **USB image**

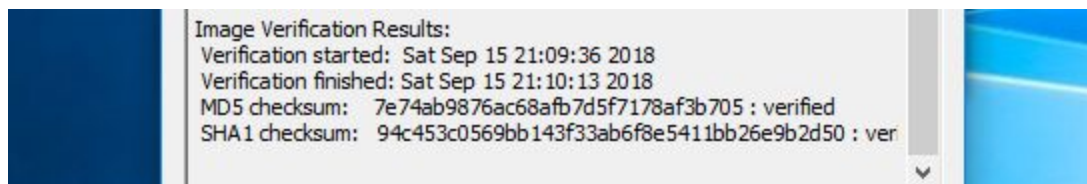
```
Md5sum usb.dd
```

```
Shasum usb.dd
```



```
sansforensics@siftworkstation -> ~  
$ md5sum usb.dd  
c1e9407f9aed1754b124a198b895d6bd usb.dd  
sansforensics@siftworkstation -> ~  
$ shasum usb.dd  
031e9a4d0c2da7724aea7a77662f8a01653d13bd usb.dd  
sansforensics@siftworkstation -> ~  
$
```

- When comparing the hash values of usb.dd with the hash values of the raw image created by FTK imager in Part 1, **the values are different.**



Unfortunately, generating the different hashes can be caused by plugging in and unplugging the USB when switching from a windows VM (FTK Imager) to a Linux VM(SIFT Workstation), when the USB doesn't automatically mount. Additionally, to be guaranteed that these hashes do match, a write blocker can and should be utilized. Due to the expense of a write blocker, one was not used in this exercise.

Part 3 → Linux memory acquisition using LiME

5. Using LiME to dump SIFT Workstation memory

[screenshot of lsmod result]

→ shows lime grep successful

```
sansforensics@siftworkstation -> ~/Desktop
$ lsmod | grep lime
lime                16384  0
sansforensics@siftworkstation -> ~/Desktop
$
```

6. Extracting information from memory dump with Foremost

[commands and options used to extract information]

→ Breaking down the Foremost Commands

'man foremost' for options on what search parameters can be used

[-t] to define the file types you are interested in collecting.
In this case → (jpg,gif,png) & (zip,exe,doc) were completed.

[-o] to define the output directory / filename

[-i] to define the input or which file to search within.
In this case → (memory_dump.bin file)

→ Commands used:

Foremost -t jpg,png,gif -o image_analyze -i will_memory_dump.bin

```
sansforensics@siftworkstation -> ~/Desktop
$ foremost -t jpg,png,gif -o image_analyze -i will_memory_dump.bin
Processing: will_memory_dump.bin
|*****|
sansforensics@siftworkstation -> ~/Desktop
$
```

Foremost -t exe,zip -o file_analyze -i will_memory_dump.bin

```
cases image_analyze mount_points output pdf will_memory_dump.bin
sansforensics@siftworkstation -> ~/Desktop
$ foremost -t exe,zip -o file_analyze -i will_memory_dump.bin
Processing: will_memory_dump.bin
|*****|
```


7. Directories / subdirectories created & files extracted after running Foremost

[Interesting data in memory]

After defining [-o] output, [image_analyze] & [file_analyze] folders were created
Inside each output directory is [audit.txt] file & folders corresponding to each [-t] file type
[audit.txt] will show a summary of files extracted & "file offset", exactly where file was found
file type folders, ex: [doc, jpg, exe, zip], will contain all extracted files of that type

→ **Image_analyze** directory created

[Audit.txt file & doc, exe, gif, jpg, ost, png subdirectories created]

```
sansforensics@siftworkstation -> ~/Desktop
$ cd image_analyze/
sansforensics@siftworkstation -> ~/D/image_analyze
$ ls
audit.txt  doc  exe  gif  jpg  ost  png
sansforensics@siftworkstation -> ~/D/image_analyze
$
```

Results: Lists total number of each file type extracted

```
352 FILES EXTRACTED
```

```
jpg:= 8
png:= 255
gif:= 84
doc:= 2
ost:= 1
exe:= 2
-----
```

File_analyze directory created

[Audit.txt file & doc, exe, ost, zip subdirectories created]

```
sansforensics@siftworkstation -> ~/Desktop
$ ls
cases  file_analyze  image_analyze  mount_points  output  pdf  will_memory_dump.bin
sansforensics@siftworkstation -> ~/Desktop
$ cd file_analyze/
sansforensics@siftworkstation -> ~/D/file_analyze
$ ls
audit.txt  doc  exe  ost  zip
sansforensics@siftworkstation -> ~/D/file_analyze
```

Results: Lists total number of each file type extracted

```
Finish: Fri Sep 14 22:49:22 2018
```

```
20 FILES EXTRACTED
```

```
zip:= 15
doc:= 2
ost:= 1
exe:= 2
-----
```