**CIS-387: Digital Forensics (4 credits)**

**With Dr. Jinhua Guo**

**Lab 4**

**Demetrius Johnson**

**October 07, 2022**

# ACTIVITY: FAT File System Analysis with Sleuthkit

**Use the Sleuthkit commandline tools to analyze the image file, fatDisk.dd:**

## 1. Open the SANS Investigative Forensic Toolkit (SIFT) Workstation.

## 2. Find the offset of the starting sector for the FAT partition.

Command: **mmls fatDisk.dd**

Text

Description automatically generated

The first partition is FAT table (1 sector in length as shown), the next 32 sectors are unallocated (reserved for FAT table, which is calculated based on **Number of FAT structures \* size of each FAT**), and partition 002 (the first volume) starts at sector 32 and ends at sector 251903.

## 3. Carve out the MBR (the first sector of the disk) and VBR (boot sector, the first sector of the FAT partition) records with dd command and check it with xxd command.

Example**: dd if=fatDisk.dd of=MBR.dd count=1 🡪 this will get MBR (master boot record) sector**

Example**: xxd MBR.dd** 🡪 print out file in hex format.

A picture containing graphical user interface

Description automatically generated

* Notice, 512 bytes copied because dd command copied the 1st sector (sector 0: and each sector is 512 bytes).
* Also, notice xxd command outputs a table, where the first column is hexadecimal counter for the starting byte, where each line outputs 16 bytes. For example, 0x00000000 outputs the first 16 bytes of the file, then line two 0x00000010 outputs the next 16 bytes of the file on that line, etc.
* Notice the last two bytes are 55aa = MBR signature; the above output 1st sector (sector 0) is the MBR = 512 bytes.

Now I use: **dd if=fatDisk.dd skip=32 count=1 of=VBR.dd to copy VBR (volume boot record) sector 🡪 first sector in volume 1 (partition 002).**

Text

Description automatically generated with low confidence

* Notice 55aa is last 2 bytes = MBR signature; thus a copy of MBR is in the VBR (boot sector for partition/volume).

## 4. Find the image’s file system information (use the offset you got from mmls in step 2). Report the details of the file system, including reserved area, fat 0, fat 1, data area, root directory, cluster area, sector size, and cluster size.

Command: **fsstat –o <*offset in # of sectors>* fatDisk.dd**

Text

Description automatically generated

* Notice for offset (in sectors) I used 32 🡪 meaning I am reading the boot sector (VBR= volume boot record) of the first volume of FAT image (since volume 1 🡪 partition 2 🡪 starts at sector 32).
* Notice, there is FAT 0 and FAT 1 🡪 they are just copies of each other so that we have a backup FAT table.

## 5. Use fls to list all deleted files and directories.

Command: **fls -o <offset in sectors> -f fat -rd fatDisk.dd**

Text

Description automatically generated

* -fls -o 32 -rd fatDisk.dd --> -r means search recursively -d means search for deleted (marked as deleted) files on fatDisk.dd
  + remember meta data = 32 bytes and includes file name; the first few bytes is used for the short file name and first byte=first letter of file name is over written with 0xe5=marked for deletion 🡪 so you can often lose first letter of file name, but you can find it in another field in the meta data header under the long filename section.
  + Time stamps of FAT file system can often be not very accurate.
* In FAT table also the location of the file will be NULLED
* But using istat we can still recover consecutive (not fragmented!) files because we know the first sector of the first block/sector which was marked with 0xe5=marked as deleted.

## 6. Use istat to view the details of metadata information of each deleted file.

Example: **istat -o <offset> -f fat fatDisk.dd 7**

* now we can recover the files using: istat -o 32 fatDisk.dd 7 --> will tell you if the file location is overwritten yet after having been marked for deletion

Text

Description automatically generated with medium confidence

* + note above the 7 = file ID = the actual numerical value the file system uses because it needs to associate names (human readable) with actual numbers (for computer processing).

## 7. Use icat to dump out data of each deleted file.

Example: **icat -o 32 -f fat fatDisk.dd 7 > demoDocx.docx**

Text

Description automatically generated with medium confidence

## 8. Dump out just one datablock of “demoDocx.docx” file.

Choose a datablock number from your istat result, for example, 2048.

Example: **blkcat -o <offset> -f fat fatDisk.dd 2048 | xxd**

A picture containing table

Description automatically generated

Here is the file output via the application that uses .docx file format:

Graphical user interface, text, application

Description automatically generated

# Dumping Penguin JPEG file (just for fun)

Text

Description automatically generated with low confidence

A group of penguins

Description automatically generated with medium confidence

# Summary/Reflection

**Report the details of the file system, including reserved area, fat 0, fat 1, data**

**area, root directory, cluster area, sector size, and cluster size.**

**File System details:**

Text

Description automatically generated

Notice above, FAT 0 starts at sector 4, FAT 1 starts at sector 250 (and is a backup copy of the FAT table). Root directory starts at sector 496….etc. (as seen above, all data fields for the FAT image file are output).

**Include a brief reflection on what you learned (one or two paragraphs).**

I learned how to use the powerful Sleuth kit tools on the SIFT workstation. Now, I will always have this tool to analyze my own drives and recover files whenever necessary. I also understand how FAT drive formatting works (and drives in general) and how one might develop a file system. For example, the MBR stores valuable information about the drive formatting, and so does the VBR of each volume/partition, and the file system includes ways for the system to be recovered in the event of damages to sectors of the disk. I also enjoyed using the icat and istat programs to locate deleted files and dump their data for recovery. This lab has helped me to understands computers and their architecture in a much greater general sense.