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**

```
sansforensics@siftworkstation: ~/host
$ mmls fatDisk.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   0000000031   0000000032   Unallocated
002:  000:000   0000000032   0000251903   0000251872   DOS FAT16 (0x06)
sansforensics@siftworkstation: ~/host
$
```

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.

```
sansforensics@siftworkstation: ~/host
$ dd if=fatDisk.dd of=MBR.dd count=1
1+0 records in
1+0 records out
512 bytes copied, 0.0028937 s, 177 kB/s
sansforensics@siftworkstation: ~/host
$ xxd MBR.dd
00000000: fabe 007c bf00 7ab9 0001 fc0e 1f0e 07f3  ...|..z.........
00000010: a5ea 167a 0000 bbbe 7b33 c980 3f80 7506  ...z....{3..?.u.
00000020: fec5 8bf3 eb07 803f 0075 02fe c183 c310  .......?.u......
00000030: 81fb fe7b 72e5 83f9 0474 0b81 f903 0174  ...{r....t.....t
00000040: 0abb a57a eb2c bb87 7aeb 278b 4c02 8b14  ...z.,..z.'.L...
00000050: b801 02bb 007c cd13 7305 bbbc 7aeb 132e  .....|..s...z...
00000060: a1fe 7d3d 55aa 7405 bbbc 7aeb 05ea 007c  ..}=U.t...z....|
00000070: 0000 2e8a 073c 0074 0c53 bb07 00b4 0ecd  .....<.t.S......
00000080: 105b 43eb edeb fe4e 6f20 626f 6f74 6162  .[C....No bootab
00000090: 6c65 2070 6172 7469 746f 6e20 696e 2074  le partiton in t
000000a0: 6162 6c65 0049 6e76 616c 6964 2050 6172  able.Invalid Par
000000b0: 7469 746f 6e20 7461 626c 6500 496e 7661  titon table.Inva
000000c0: 6c69 6420 6f72 2064 616d 6167 6564 2042  lid or damaged B
000000d0: 6f6f 7461 626c 6520 7061 7274 6974 696f  ootable partitio
000000e0: 6e00 0000 0000 0000 0000 0000 0000 0000  n...............
000000f0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000100: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000110: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000120: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000130: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000140: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000150: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000160: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000170: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000180: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000190: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000001a0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
000001b0: 0000 0000 0000 0000 0127 735b 0000 8001  .........'s[....
000001c0: 0100 060f 60eb 2000 0000 e0d7 0300 0000  ....`. .........
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.
sansforensics@siftworkstation: ~/host
```

- 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).**

```
sansforensics@siftworkstation: ~/host
$ dd if=fatDisk.dd skip=32 count=1 of=VBR.dd
1+0 records in
1+0 records out
512 bytes copied, 0.00361121 s, 142 kB/s
sansforensics@siftworkstation: ~/host
$ xxd VBR.dd
00000000: eb3c 904d 5344 4f53 352e 3000 0204 0400  .<.MSDOS5.0.....
00000010: 0200 0200 00f8 f600 3f00 ff00 2000 0000  ........?... ...
00000020: e0d7 0300 8000 292e 3b32 784e 4f20 4e41  ......).;2xNO NA
00000030: 4d45 2020 2020 4641 5431 3620 2020 33c9  ME    FAT16   3.
00000040: 8ed1 bcf0 7b8e d9b8 0020 8ec0 fcbd 007c  ....{.... .....|
00000050: 384e 247d 248b c199 e83c 0172 1c83 eb3a  8N$}$....<.r...:
00000060: 66a1 1c7c 2666 3b07 268a 57fc 7506 80ca  f..|&f;.&.W.u...
00000070: 0288 5602 80c3 1073 eb33 c98a 4610 98f7  ..V....s.3..F...
00000080: 6616 0346 1c13 561e 0346 0e13 d18b 7611  f..F..V..F....v.
00000090: 6089 46fc 8956 feb8 2000 f7e6 8b5e 0b03  `.F..V.. ....^..
000000a0: c348 f7f3 0146 fc11 4efe 61bf 0000 e8e6  .H...F..N.a.....
000000b0: 0072 3926 382d 7417 60b1 0bbe a17d f3a6  .r9&8-t.`....}..
000000c0: 6174 324e 7409 83c7 203b fb72 e6eb dca0  at2Nt... ;.r....
000000d0: fb7d b47d 8bf0 ac98 4074 0c48 7413 b40e  .}.}....@t.Ht...
000000e0: bb07 00cd 10eb efa0 fd7d ebe6 a0fc 7deb  .........}....}.
000000f0: e1cd 16cd 1926 8b55 1a52 b001 bb00 00e8  .....&.U.R......
00000100: 3b00 72e8 5b8a 5624 be0b 7c8b fcc7 46f0  ;.r.[.V$..|...F.
00000110: 3d7d c746 f429 7d8c d989 4ef2 894e f6c6  =}.F.)}...N..N..
00000120: 0696 7dcb ea03 0000 200f b6c8 668b 46f8  ..}..... ...f.F.
00000130: 6603 461c 668b d066 c1ea 10eb 5e0f b6c8  f.F.f..f....^...
00000140: 4a4a 8a46 0d32 e4f7 e203 46fc 1356 feeb  JJ.F.2....F..V..
00000150: 4a52 5006 536a 016a 1091 8b46 1896 9233  JRP.Sj.j...F...3
00000160: d2f7 f691 f7f6 4287 caf7 761a 8af2 8ae8  ......B...v.....
00000170: c0cc 020a ccb8 0102 807e 020e 7504 b442  .........~..u..B
00000180: 8bf4 8a56 24cd 1361 6172 0b40 7501 4203  ...V$..aar.@u.B.
00000190: 5e0b 4975 06f8 c341 bb00 0060 666a 00eb  ^.Iu...A...`fj..
000001a0: b042 4f4f 544d 4752 2020 2020 0d0a 5265  .BOOTMGR    ..Re
000001b0: 6d6f 7665 2064 6973 6b73 206f 7220 6f74  move disks or ot
000001c0: 6865 7220 6d65 6469 612e ff0d 0a44 6973  her media....Dis
000001d0: 6b20 6572 726f 72ff 0d0a 5072 6573 7320  k error...Press
000001e0: 616e 7920 6b65 7920 746f 2072 6573 7461  any key to resta
000001f0: 7274 0d0a 0000 0000 0000 00ac cbd8 55aa  rt............U.
sansforensics@siftworkstation: ~/host
```

- 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**

```
sansforensics@siftworkstation: ~/host
$ fsstat -o 32 fatDisk.dd
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: FAT16

OEM Name: MSDOS5.0
Volume ID: 0x78323b2e
Volume Label (Boot Sector): NO NAME
Volume Label (Root Directory): TRAVELDRIVE
File System Type Label: FAT16

Sectors before file system: 32

File System Layout (in sectors)
Total Range: 0 - 251871
* Reserved: 0 - 3
** Boot Sector: 0
* FAT 0: 4 - 249
* FAT 1: 250 - 495
* Data Area: 496 - 251871
** Root Directory: 496 - 527
** Cluster Area: 528 - 251871

METADATA INFORMATION
--------------------------------------------
Range: 2 - 4022022
Root Directory: 2

CONTENT INFORMATION
--------------------------------------------
Sector Size: 512
Cluster Size: 2048
Total Cluster Range: 2 - 62837

FAT CONTENTS (in sectors)
--------------------------------------------
sansforensics@siftworkstation: ~/host
```

- 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**

```
sansforensics@siftworkstation: ~/host
$ fls -o 32 -f fat -rd fatDisk.dd
r/r * 5:        Penguins.jpg
r/r * 7:        demoDocx.docx
r/r * 10:       demoPdfFromDocx.pdf
r/r * 13:       foremost_demo.zip
sansforensics@siftworkstation: ~/host
$
```

- -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

```
sansforensics@siftworkstation: ~/host
$ istat -o 32 -f fat fatDisk.dd 7
Directory Entry: 7
Not Allocated
File Attributes: File, Archive
Size: 9861
Name: _EMODO~1.DOC

Directory Entry Times:
Written:        2012-11-25 22:04:30 (UTC)
Accessed:       2012-11-25 00:00:00 (UTC)
Created:        2012-11-25 22:15:36 (UTC)

Sectors:
2048 2049 2050 2051 2052 2053 2054 2055
2056 2057 2058 2059 2060 2061 2062 2063
2064 2065 2066 2067
sansforensics@siftworkstation: ~/host
```

    - 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.

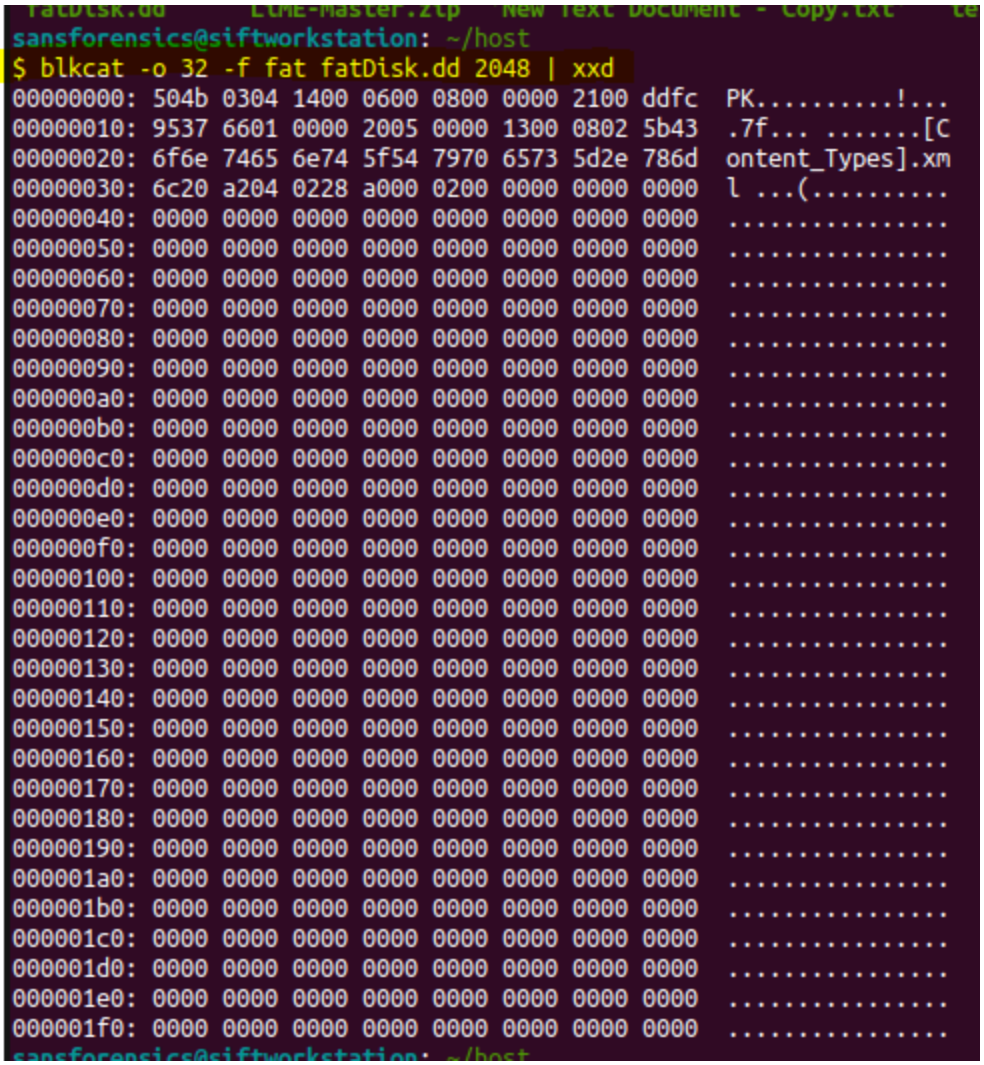
Example: **icat -o 32 -f fat fatDisk.dd 7 > demoDocx.docx**

```
sansforensics@siftworkstation: ~/host
$ icat -o 32 -f fat fatDisk.dd 7 > demoDocx.docx
sansforensics@siftworkstation: ~/host
$ ls
 copy.dd         lab4hints.txt     MBR.dd                        'New Text Document.txt'   test.txt    zero.dd
 demoDocx.docx   LiME-master       ncData.dd                      test_input_file.txt       usb.dd      zeus.vmem
 fatDisk.dd      LiME-master.zip  'New Text Document - Copy.txt'  test_output_file.txt      VBR.dd
sansforensics@siftworkstation: ~/host
```

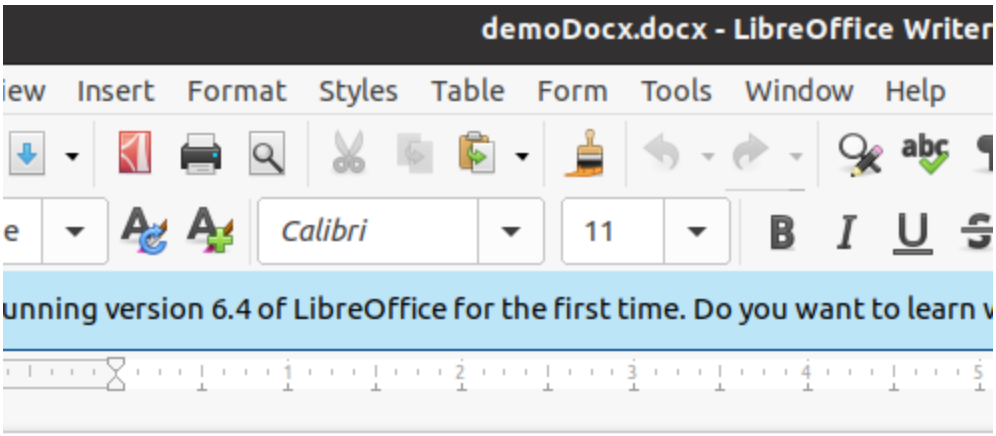
8. Dump out just one datablock of "demoDocx.docx" file.

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

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



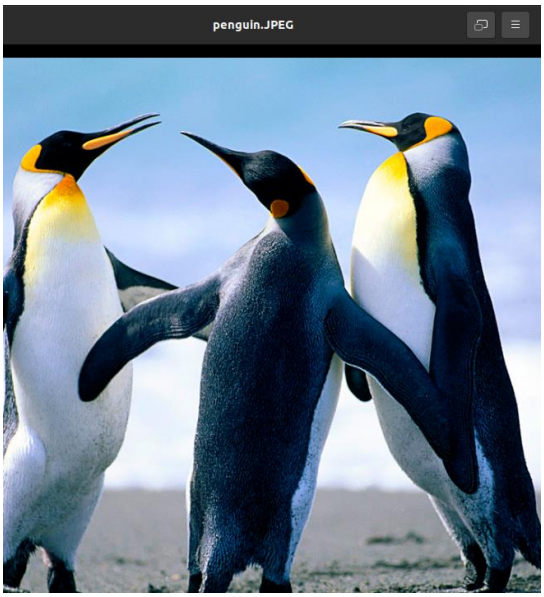
Here is the file output via the application that uses .docx file format:



This is a simple file for demo purposes.

Dumping Penguin JPEG file (just for fun)

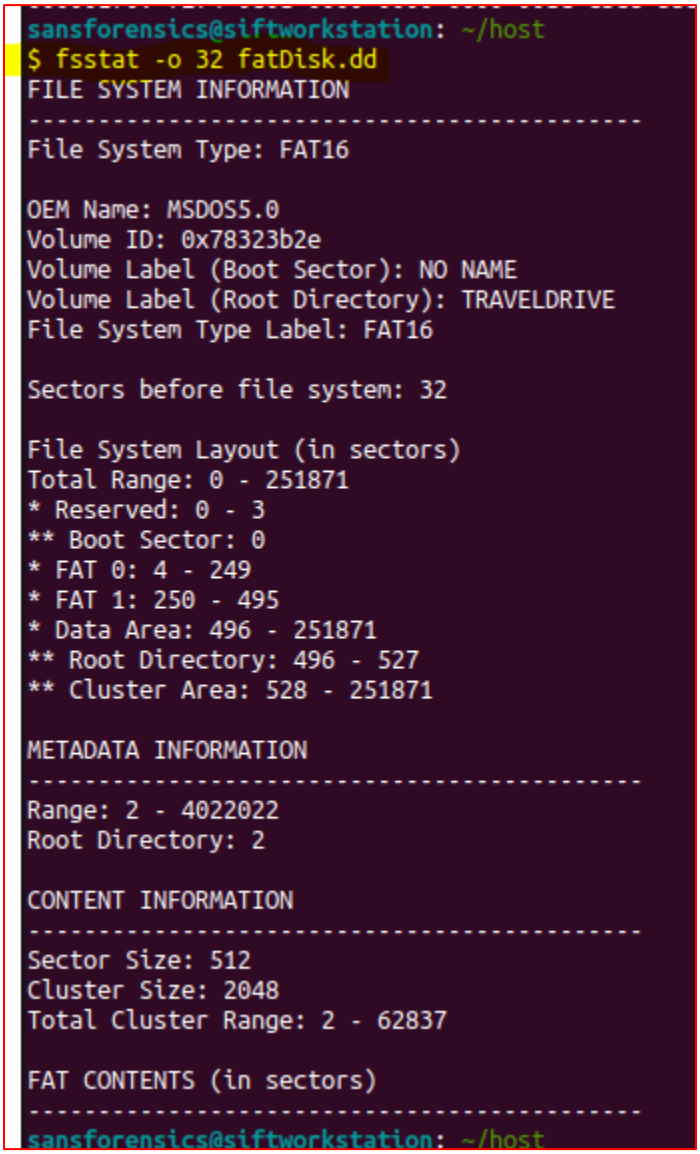
```
sansforensics@siftworkstation: ~/host
$ icat -o 32 -f fat fatDisk.dd 5 > penguin.JPEG
sansforensics@siftworkstation: ~/host
$ ls
 copy.dd          lab4hints.txt     MBR.dd                        'New Text Document.txt'   test_output_file.txt   VBR.dd
 demoDocx.docx    LiME-master       ncData.dd                     penguin.JPEG              test.txt               zero.dd
 fatDisk.dd       LiME-master.zip  'New Text Document - Copy.txt'  test_input_file.txt       usb.dd                 zeus.vmem
sansforensics@siftworkstation: ~/host
$
```



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:



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.