Digital Incident Scene Investigation and Analysis

Coursework 1 - Disk Acquisition & File System Analyst

Name: Muhaned Ali Nouman
Student Number: M00872751

Table of Contents

5-Directory Entries

6-File Content & Extraction

Reference

1-Drive Imaging

A) Drive imaging steps

-I did the following steps to ensure that the e image was acquired correctly and follows best practice:

- I powered up the write blocker & connected it into my machine.
- I connected the evidence diver into the write blocker.

Figure 1.1: write blocker setup.

- In the FTK Imager software I selected the source type to be physical Drive as it shows in Figure 1 below

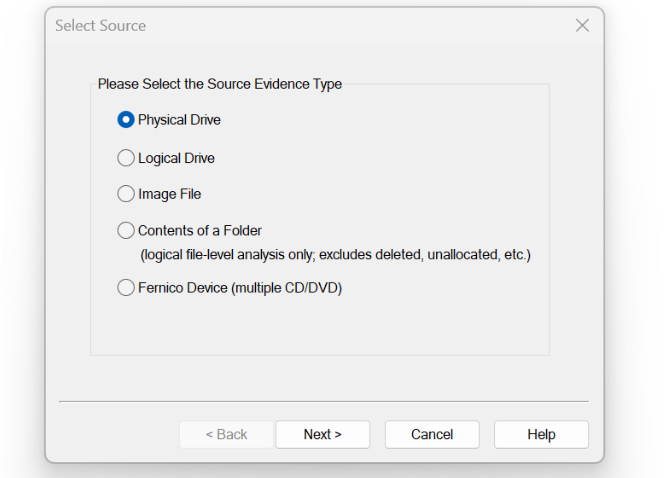


Figure 1.1: Select the evidence source as physical drive.

- In the FTK Imager software I make sure that I selected the correct source USB drive inserted in my write blocker

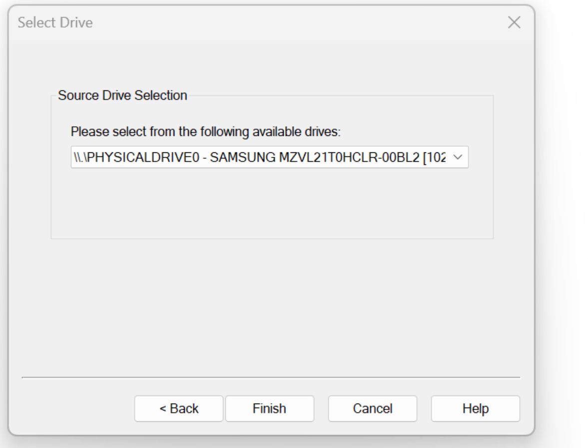


Figure 1.1: Selecting the correct USB drive.

- I make sure that I selected the correct Image type which is set as "Raw".

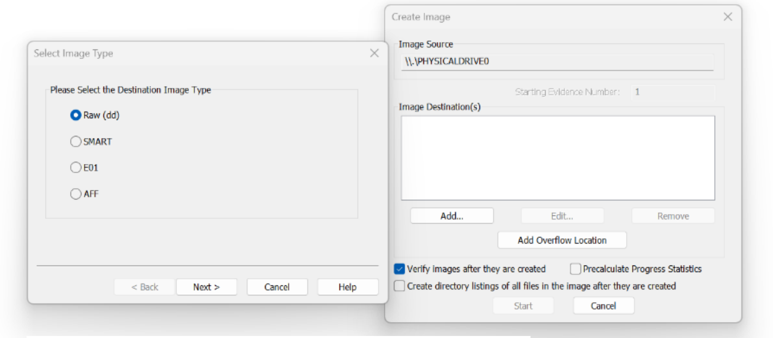


Figure 1.1: Selecting the Destination Image type.

- important step to do was filling the Evidence item information filed.

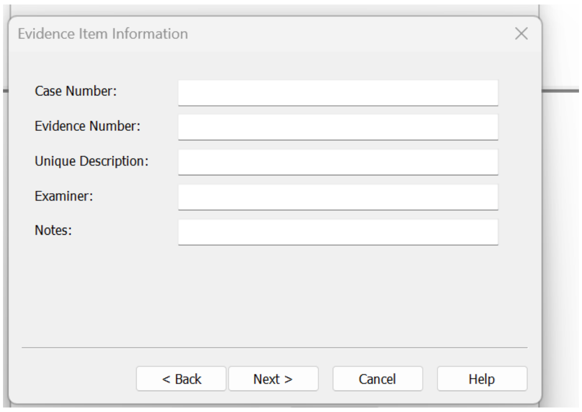


Figure 1.1: Evidence Item information.

- The last step in creating the image was selecting the destination folder & set the image fragment to 0 because I don't want to separate the image file into multiple images.

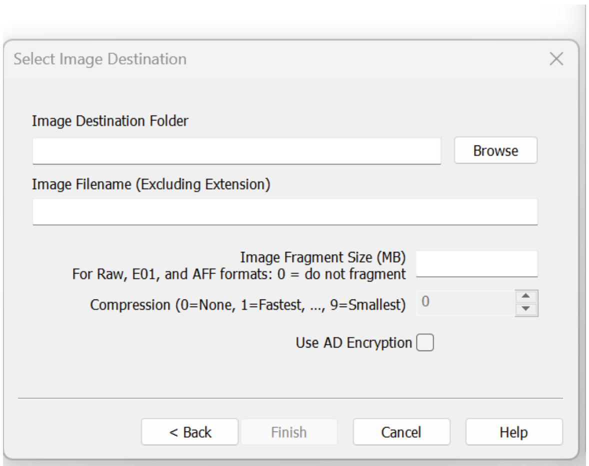
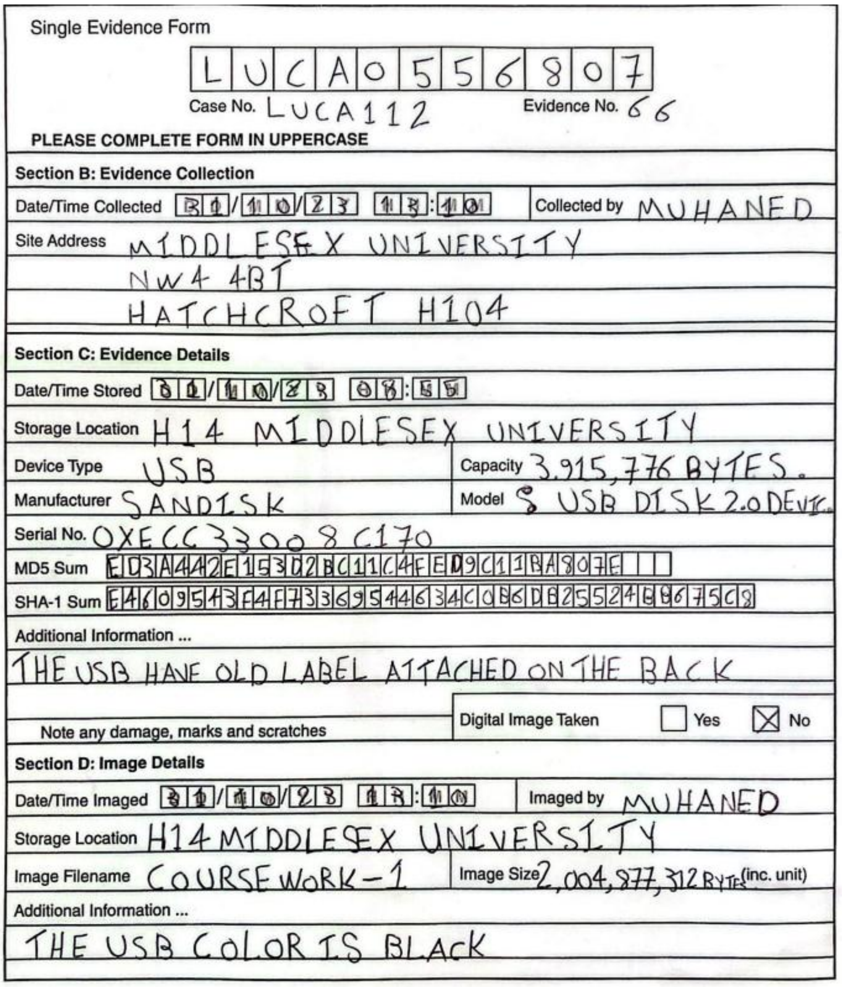


Figure 1.1: Selecting the image destination.

B) Acquisition Form



Single Evidence Form

LUCA0556807

Case No. LUCA112 Evidence No. 66

PLEASE COMPLETE FORM IN UPPERCASE

Section B: Evidence Collection	
Date/Time Collected 31/10/23 13:10	Collected by MUHANED
Site Address MIDDLESEX UNIVERSITY NW4 4BT HATCHCROFT H104	
Section C: Evidence Details	
Date/Time Stored 31/10/23 08:55	
Storage Location H14 MIDDLESEX UNIVERSITY	
Device Type USB	Capacity 3,915,776 BYTES.
Manufacturer SANDISK	Model S USB DISK 2.0 DEVIC.
Serial No. 0XECC33008C170	
MD5 Sum ED3A442E153D2BC11C4FED9C11BA807E	
SHA-1 Sum E4609543F4F73369544634C0B6DB25524BB675C8	
Additional Information ... THE USB HAVE OLD LABEL ATTACHED ON THE BACK	
Note any damage, marks and scratches	Digital Image Taken ☐ Yes ☒ No
Section D: Image Details	
Date/Time Imaged 31/10/23 13:10	Imaged by MUHANED
Storage Location H14 MIDDLESEX UNIVERSITY	
Image Filename COURSEWORK-1	Image Size 2,004,877,312 BYTES (inc. unit)
Additional Information ... THE USB COLOR IS BLACK	

Figure 1.2: Acquisition Form.

C) Image verification

```
[Computed Hashes]
 MD5 checksum:    ed3a442e153d2bc11c4fed9c11ba807e
 SHA1 checksum:   e4609543f4f73369544634c0b6db25524bb675c8

Image Information:
 Acquisition started:   Tue Oct 31 13:08:55 2023
 Acquisition finished:  Tue Oct 31 13:10:20 2023
 Segment list:
  C:\Users\mn982\Desktop\Coursework-1.001

Image Verification Results:
 Verification started:  Tue Oct 31 13:10:20 2023
 Verification finished: Tue Oct 31 13:10:26 2023
 MD5 checksum:    ed3a442e153d2bc11c4fed9c11ba807e : verified
 SHA1 checksum:   e4609543f4f73369544634c0b6db25524bb675c8 : verified
```

Figure 1.3: drive/Imaging Verify result.

- MD5 (Message Digest) checksum:

ed3a442e153d2bc11c4fed9c11ba807e (verified)

- SHA1 (Secure Hash Algorithm) checksum: e4609543f4f73369544634c0b6db25524bb675c8 (verified)
- The verification process started on October 31, 2023, at 13:10:20 and finished at 13:10:26, indicating a relatively quick and successful verification. The image was acquired from the path "C:\Users\mn982\Desktop\Coursework-1.001" starting on October 31, 2023, at 13:08:55 and finishing at 13:10:20.

D) Why Imaging Is Important:

Imaging plays a pivotal role in digital incident site investigations for several compelling reasons. First and foremost, it facilitates meticulous documentation of the digital environment, capturing configurations, files, and systems at the time of the incident. This comprehensive record serves critical legal and analytical purposes, ensuring that investigators have a detailed account of the scene [1],[2]. Moreover, imaging is instrumental in preserving digital evidence by taking snapshots of the digital scene. This not only keeps the evidence intact but also provides investigators with a consistent reference point for analysis, bolstering its legitimacy in court [3].

In the realm of digital forensics, imaging proves indispensable for analysing and reconstructing events. By offering investigators a rich dataset, imaging aids in tracking actions, comprehending the incident's timeframe, and reconstructing the sequence of events. This analytical depth is crucial for a thorough investigation [1]. Additionally, proper imaging establishes a clear chain of custody for digital evidence. This chain of custody is vital for preserving the credibility and admissibility of evidence in court, demonstrating that the evidence remained unaltered throughout the inquiry [3].

The significance of imaging extends to remote investigations, where the incident scene may be geographically distant. Investigators can remotely gather and examine digital evidence through imaging, enhancing the efficiency and speed of incident response [1],[2]. Lastly, imaging techniques are invaluable for data recovery, allowing investigators to retrieve deleted or concealed data essential to the inquiry. This capability is crucial for detecting malicious activity and gaining a comprehensive understanding of the entire event [1].

2-Master Boot Record

A) Location of MBR & number of partitions stored on the device.

Location of MBR:

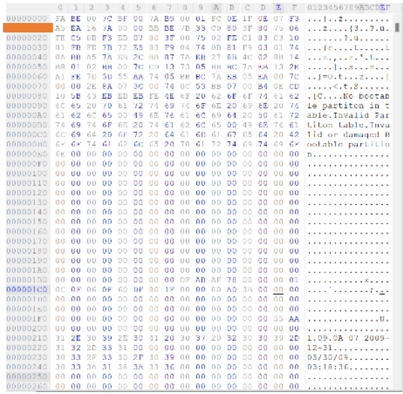


Figure 2.1: Location of MBR.

- MPR will have Ascii conversion, Start and end of Partition table as the above.
- Byte 446 / Byte 0x 1BE is the start of partition table.

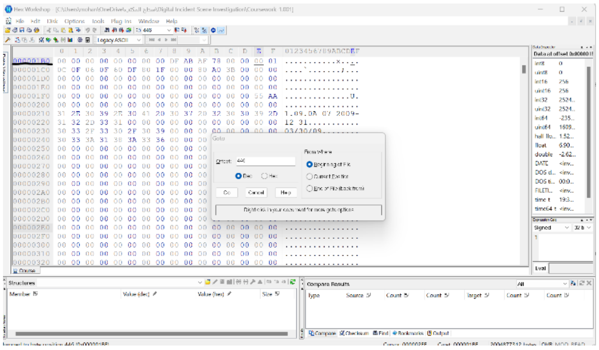


Figure 2.1: Location of MBR

Number of partitions stored on the device:

- There is 1 partition stored in the device which is the 1st partition.

- MBR – 1st Partition:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
000001B0	00	00	00	00	00	00	00	00	DF	AB	AF	78	00	00	00	01	...........x....
000001C0	0C	0F	06	0F	60	DF	80	1F	00	00	80	A0	3B	00	00	00	....`.......;...

Figure 2.1: 1st Partition

B) The size in bytes of the largest partition

Partition Size:

- 1st Partition:

```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
000001B0  00 00 00 00 00 00 00 00 DF AB AF 78 00 00 00 01 ...........x....
000001C0  0C 0F 06 0F 60 DF 80 1F 00 00 80 A0 3B 00 00 00 ....`.......;...
```

Figure 2.1: 1st Partition

Calculation

- Bytes 12-15(Size in Sectors) = 0x 80 A0 3B 00
- Reverse endian = 0x 3B A0 80
- Convert to decimal = 3,907,712 bytes

```
PS C:\sleuthkit-4.12.1-win32\bin> .\mmls.exe .\Coursework-1.001
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   0000008063   0000008064   Unallocated
002:  000:000   0000008064   0003915775   0003907712   DOS FAT16 (0x06)
```

Figure 2.2: 1st Partition Size.

C) The offset address for the start of largest partition.

```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
000001B0  00 00 00 00 00 00 00 00 DF AB AF 78 00 00 00 01  ...........x....
000001C0  0C 0F 06 0F 60 DF 80 1F 00 00 80 A0 3B 00 00 00  ....`.......;...
```

Figure 2.3:1st Partition.

The starting physical sector of the partition.

Calculation:

- Bytes 8-11(Starting of the LBA) = 0x 80 1F 00 00
- Reverse endian = 0x 1F 80
- Convert to decimal = 8,064 sectors start of file system

```
C:\sleuthkit-4.12.1-win32\bin>mmls Coursework-1.001
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   0000008063   0000008064   Unallocated
002:  000:000   0000008064   0003915775   0003907712   DOS FAT16 (0x06)
```

Figure 2.3: byte offset address.

The calculation of the byte offset address for start of file System:

- Starting LBA * Bytes per sector
- 8,064 * 512 = 4,128,768
- Convert to hex = 3F 0000 is where file system starts

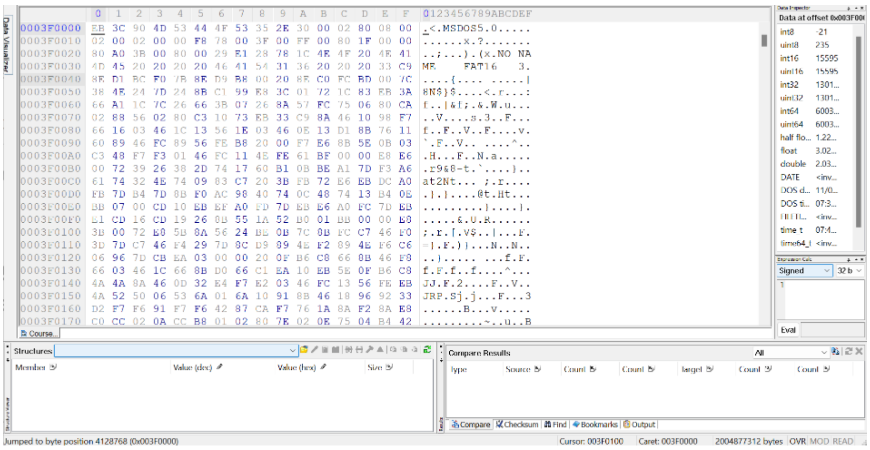


Figure 2.3: Starting Address of Partition (file system).

3-Volume Boot Record

A) Determine the number of bytes per sector.

Calculation:

- Bytes 11-12(bytes per sector) = 0x 00 02

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
003F0000	EB	3C	90	4D	53	44	4F	53	35	2E	30	00	02	80	08	00	.<.MSDOS5.0.....
003F0010	02	00	02	00	00	F8	78	00	3F	00	FF	00	80	1F	00	00	......x.?.......

Figure 3.1: Bytes 11-12 in hex workshop.

- Reverse endian – 0x 02 00
- Convert to decimal = 512 bytes per sector

```
PS C:\sleuthkit-4.12.1-win32\bin> .\mmls.exe .\Coursework-1.001
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors
```

Figure 3.1: bytes per sector

B) Determine the number of sectors per cluster.

Calculation:

- Byte 13(Sector per cluster) = 0x 80

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
003F0000	EB	3C	90	4D	53	44	4F	53	35	2E	30	00	02	80	08	00	.<.MSDOS5.0.....

Figure 3.2: Byte 13

- Reverse endian – not possible
- Convert to decimal = 128 Sectors per cluster.

```
PS C:\sleuthkit-4.12.1-win32\bin> .\fsstat.exe -o 8064 .\Coursework-1.001
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: FAT16

OEM Name: MSDOS5.0
Volume ID: 0x1c7828e1
Volume Label (Boot Sector): NO NAME
Volume Label (Root Directory): FAT_CW1
File System Type Label: FAT16

Sectors before file system: 8064

File System Layout (in sectors)
Total Range: 0 - 3907711
* Reserved: 0 - 7
** Boot Sector: 0
* FAT 0: 8 - 127
* FAT 1: 128 - 247
* Data Area: 248 - 3907711
** Root Directory: 248 - 279
** Cluster Area: 280 - 3907607
** Non-clustered: 3907608 - 3907711
```

Figure 3.2: 128 Sectors per cluster.

C) Determine the number of reserved sectors.

Calculation:

```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
003F0000  EB 3C 90 4D 53 44 4F 53 35 2E 30 00 02 80 08 00  .<.MSDOS5.0.....
```

Figure 3.3: Bytes 14-15.

- Bytes 14-15(size in sectors of reserved area) = 0x 08 00
- Reverse endian – 0x 00 08
- Convert to decimal = 8 reserved sectors

```
File System Type: FAT16

OEM Name: MSDOS5.0
Volume ID: 0x1c7828e1
Volume Label (Boot Sector): NO NAME
Volume Label (Root Directory): FAT_CW1
File System Type Label: FAT16

Sectors before file system: 8064

File System Layout (in sectors)
Total Range: 0 - 3907711
* Reserved: 0 - 7
** Boot Sector: 0
* FAT 0: 8 - 127
* FAT 1: 128 - 247
* Data Area: 248 - 3907711
** Root Directory: 248 - 279
** Cluster Area: 280 - 3907607
** Non-clustered: 3907608 - 3907711
```

Figure 3.3: Size of Reserved Area in fsstat.

D) Determine the number of FAT's.

Calculation:

```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
003F0000  EB 3C 90 4D 53 44 4F 53 35 2E 30 00 02 80 08 00  .<.MSDOS5.0.....
003F0010  02 00 02 00 00 F8 78 00 3F 00 FF 00 80 1F 00 00  ......x.?.......
```

Figure 3.4: Bytes 16.

- Bytes 16(Number of FAT) = 0x 02
- Reverse endian – not necessary
- Convert to decimal = 2 FAT.

```
File System Layout (in sectors)
Total Range: 0 - 3907711
* Reserved: 0 - 7
** Boot Sector: 0
* FAT 0: 8 - 127
* FAT 1: 128 - 247
* Data Area: 248 - 3907711
** Root Directory: 248 - 279
** Cluster Area: 280 - 3907607
** Non-clustered: 3907608 - 3907711
```

Figure 3.4: the amount of FAT's in fsstat.

E) Determine the maximum number of entries allowed in the root directory.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
003F0000	EB	3C	90	4D	53	44	4F	53	35	2E	30	00	02	80	08	00	.<.MSDOS5.0.....
003F0010	02	00	02	00	00	F8	78	00	3F	00	FF	00	80	1F	00	00	......x.?.......

Figure 3.5: byte 17 and 18

Calculation:

- Bytes 17-18(MAX number of file in RD for fat 16) = 0x 00 02
- Reverse endian – 0x 200
- Convert to decimal = 512 Bytes

```
Sectors before file system: 8064

File System Layout (in sectors)
Total Range: 0 - 3907711
* Reserved: 0 - 7
** Boot Sector: 0
* FAT 0: 8 - 127
* FAT 1: 128 - 247
* Data Area: 248 - 3907711
** Root Directory: 248 - 279
** Cluster Area: 280 - 3907607
** Non-clustered: 3907608 - 3907711

METADATA INFORMATION
--------------------------------------------
Range: 2 - 62519430
Root Directory: 2

CONTENT INFORMATION
--------------------------------------------
Sector Size: 512
Cluster Size: 65536
Total Cluster Range: 2 - 30527
```

Figure 3.5: Max number of entries allowed in the RD.

F) Determine the number of sectors per FAT.

Calculation

• Bytes 22-23(size in sector for each fat) = 0x 78 00

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
003F0000	EB	3C	90	4D	53	44	4F	53	35	2E	30	00	02	80	08	00
003F0010	02	00	02	00	00	F8	78	00	3F	00	FF	00	80	1F	00	00

Figure 3.6: Bytes 22-23

• Reverse endian – 0x 78

• Convert to decimal = 120 Sectors

Size of FAT area

• FAT area size = No. of FAT's * Size of each FAT

- 2 * 120 = 240

```
File System Layout (in sectors)
Total Range: 0 - 3907711
* Reserved: 0 - 7
** Boot Sector: 0
* FAT 0: 8 - 127
* FAT 1: 128 - 247
* Data Area: 248 - 3907711
** Root Directory: 248 - 279
** Cluster Area: 280 - 3907607
** Non-clustered: 3907608 - 3907711
```

Figure 3.6: number of sectors per FAT.

A) FAT 1

FAT 1 Sector Address

- Size of Reserved = Start of FAT 1

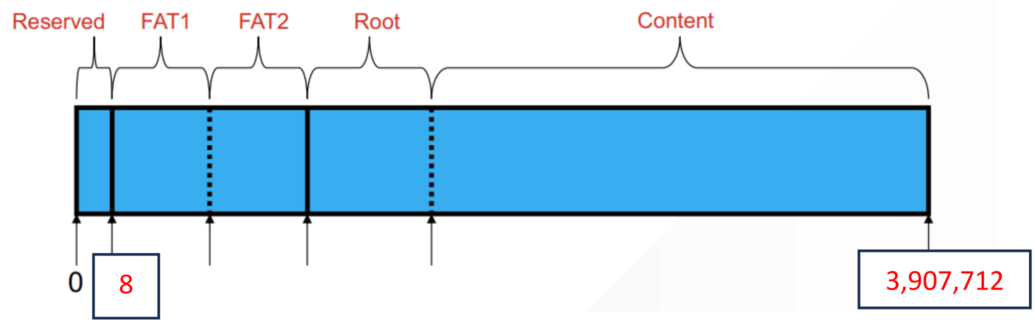


Figure 4.1: FAT 1 Sector Address

Calculation:

- No of reserved sectors * bytes per sector
  8 * 512 = 4096 bytes
- Convert to hex = 0x 1000
- Adding the offset to partition
  - 0x 1000 + 0x 3F0000 = 0x 3F 1000

```
File System Layout (in sectors)
Total Range: 0 - 3907711
* Reserved: 0 - 7
** Boot Sector: 0
* FAT 0: 8 - 127
* FAT 1: 128 - 247
* Data Area: 248 - 3907711
** Root Directory: 248 - 279
** Cluster Area: 280 - 3907607
** Non-clustered: 3907608 - 3907711
```

Figure 4.1: Fat 1 Number of reserved

```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
003F1000  F8 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
003F1010  FF FF FF FF 0B 00 0C 00 FF FF 0E 00 0F 00 10 00  ................
003F1020  11 00 12 00 13 00 14 00 15 00 16 00 17 00 18 00  ................
003F1030  19 00 1A 00 1B 00 1C 00 1D 00 1E 00 FF FF 20 00  ..............  .
003F1040  21 00 22 00 23 00 24 00 25 00 26 00 27 00 28 00  !.".#.$.%.&.'.(.
003F1050  29 00 2A 00 2B 00 2C 00 2D 00 2E 00 2F 00 30 00  ).*.+.,.-.../.0.
003F1060  31 00 32 00 33 00 FF FF 35 00 36 00 37 00 38 00  1.2.3...5.6.7.8.
```

Figure 4.1: FAT 1 location.

B) FAT 2

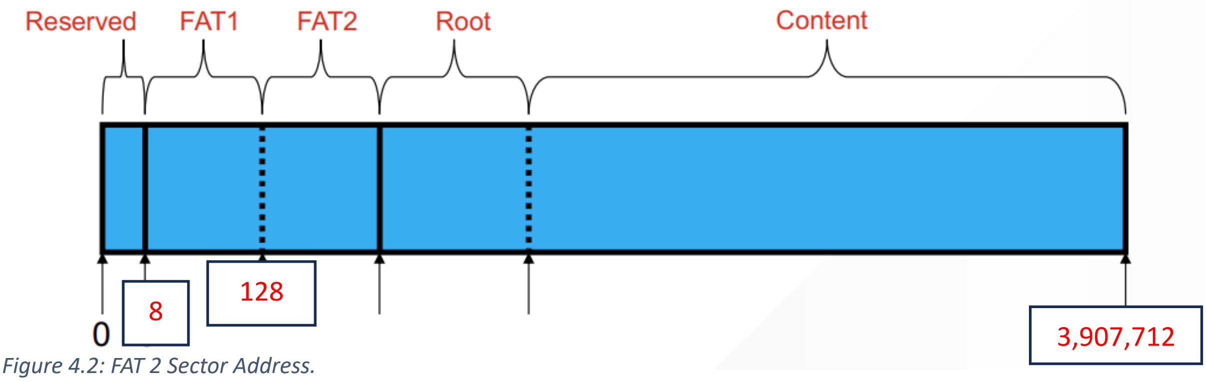


Figure 4.2: FAT 2 Sector Address.

Calculation:

- (No. of reserved sectors + Size of each FAT) * BPS
  (8 + 120) * 512 = 65536
  Convert to hex = 0x 10000
- Adding the offset to partition
  - 0x 10000 + 0x 3F0000 = 0x 40 0000

```
File System Layout (in sectors)
Total Range: 0 - 3907711
* Reserved: 0 - 7
** Boot Sector: 0
* FAT 0: 8 - 127
* FAT 1: 128 - 247
* Data Area: 248 - 3907711
** Root Directory: 248 - 279
** Cluster Area: 280 - 3907607
** Non-clustered: 3907608 - 3907711
```

Figure 4.2: FAT 2 Sector Address.

```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
00400000  F8 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
00400010  FF FF FF FF 0B 00 0C 00 FF FF 0E 00 0F 00 10 00   ................
00400020  11 00 12 00 13 00 14 00 15 00 16 00 17 00 18 00   ................
00400030  19 00 1A 00 1B 00 1C 00 1D 00 1E 00 FF FF 20 00   ..............  .
00400040  21 00 22 00 23 00 24 00 25 00 26 00 27 00 28 00   !.".#.$.%.&.'.(.
00400050  29 00 2A 00 2B 00 2C 00 2D 00 2E 00 2F 00 30 00   ).*.+.,.-.../.0.
00400060  31 00 32 00 33 00 FF FF 35 00 36 00 37 00 38 00   1.2.3...5.6.7.8.
```

Figure 4.2: FAT 2 location.

C) Root Directory

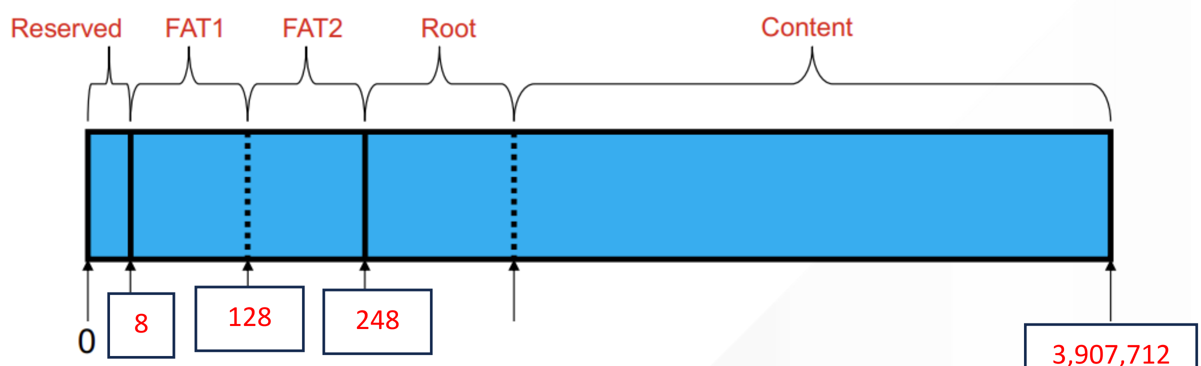


Figure 4.3: Root Directory Sector Address.

Calculation:

- No of reserved sectors + (no. of FAT's * Size of each FAT)
  8 + (2 * 120) = 248 sectors
- For offset need to multiply by Bytes per Sector (BPS)
  - 248 * 512 = 126976 bytes

  Convert to hex = 0x 1 F000
- Adding the offset to partition
  - 0x 1 F000 + 0x 3F0000 = 0x 40 F000

```
File System Layout (in sectors)
Total Range: 0 - 3907711
* Reserved: 0 - 7
** Boot Sector: 0
* FAT 0: 8 - 127
* FAT 1: 128 - 247
* Data Area: 248 - 3907711
** Root Directory: 248 - 279
** Cluster Area: 280 - 3907607
** Non-clustered: 3907608 - 3907711
```

Figure 4.3: Root Directory Sector Address.

```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
0040F000  46 41 54 5F 43 57 31 20 20 20 20 08 00 00 00 00  FAT_CW1    .....
0040F010  00 00 00 00 00 00 90 3E 63 53 00 00 00 00 00 00  .......>cS......
0040F020  E5 48 20 20 20 20 20 20 54 58 54 20 18 C7 EB 6E  .H      TXT ...n
0040F030  62 46 62 46 00 00 35 70 46 42 02 00 0D 00 00 00  bFbF..5pFB......
0040F040  32 53 20 20 20 20 20 20 54 58 54 20 18 04 EC 6E  2S      TXT ...n
```

Figure 4.3: Root Directory location.

D) Cluster 2

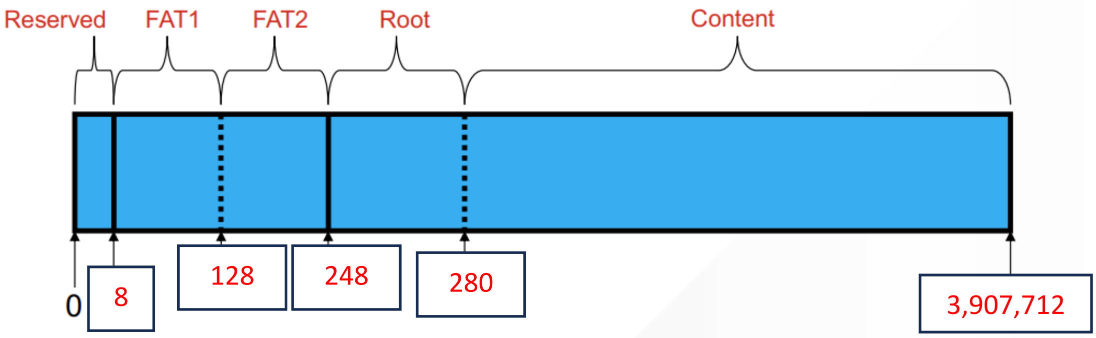


Figure 4.4: Cluster 2 Adress.

- Calculate size of root directory (RD**):**
  Size of RD = No. of RD entries * size of each entry
  512 * 32 = 16,384 bytes

- Converting to a sector value:
  Size of RD / Bytes Per Sector = No. of sectors in RD
  16384 / 512 = 32 sectors

- Adding to the sector address for the start of the RD:
  (Sector address of RD + size in sectors of RD) * BPS
  (248 + 32) * 512 = 143360
- Convert to decimal = 0x 2 3000
- Adding the offset to partition
  0x 2 3000+ 0x 3F0000 = 0x 41 3000

```
File System Layout (in sectors)
Total Range: 0 - 3907711
* Reserved: 0 - 7
** Boot Sector: 0
* FAT 0: 8 - 127
* FAT 1: 128 - 247
* Data Area: 248 - 3907711
** Root Directory: 248 - 279
** Cluster Area: 280 - 3907607
** Non-clustered: 3907608 - 3907711
```

Figure 4.4: Cluster 2 Sector Adress.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
00413000	2E	20	20	20	20	20	20	20	20	20	20	10	00	22	0E	6F	. ..".o
00413010	62	46	62	46	00	00	0F	6F	62	46	02	00	00	00	00	00	bFbF...obF......
00413020	2E	2E	20	20	20	20	20	20	20	20	20	10	00	22	0E	6F	.. ..".o
00413030	62	46	62	46	00	00	0F	6F	62	46	00	00	00	00	00	00	bFbF...obF......
00413040	42	30	00	30	00	32	00	30	00	32	00	0F	00	B3	2E	00	B0.0.2.0.2......
00413050	6A	00	70	00	67	00	00	00	FF	FF	00	00	FF	FF	FF	FF	j.p.g...........

Figure 4.4: Cluster location.

5-Directory Entries

A) Select one directory entry that is an allocated file and larger than one cluster. Provide a screenshot of the hexadecimal output.

Identifying the file in the root directory:

- byte 11 (of each entry)
  Calculation:
  - Byte 11 = 0x 20
  - Entry type = File

Hexadecimal output:

• 20th Entry.

```
           0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
0040F220  49 4D 47 30 30 31 7E 32 4A 50 47 20 00 1E 0F 6F   IMG001~2JPG ...o
0040F230  62 46 63 53 00 00 59 84 65 41 63 00 F2 D5 01 00   bFcS..Y.eAc.....
```

Figure 5.1: 20th Entry.

Calculating the File size in Directory Entries:

- Bytes 28-31 = 0x F2 D5 01 00
- Reverse endian – 0x 1 D5 F2
- Convert to decimal = 120,306 bytes

```
PS C:\sleuthkit-4.12.1-win32\bin> .\istat.exe -o 8064 .\Coursework-1.001 20
Directory Entry: 20
Allocated
File Attributes: File, Archive
Size: 120306
Name: IMG001~2.JPG
```

Figure 5.1:20TH ENTRY shows as an Allocated file in istat verification.

Calculating the size of a cluster:

- sectors per cluster * bytes per sector = size of cluster in bytes
- 128 * 512 = 65536 bytes

Comparing the file size is greater than one cluster:

- 120,306 bytes > 65536 bytes

Calculating number of clusters required for the file:

- Size of file / Bytes per cluster
- 120,306 / 65536 = 1.846
- Rounding up to nearest integer = 2 Clusters

B) Filename of the entry

Calculating the filename of the entry:

- Bytes 0-10 = 0x 49 4D 47 30 30 31 7E 32 4A 50 47

ASCII table:

- 49= I, 4D=M, 47=G, 30=0, 30=0, 31=1, 7F=~, 32=2, 4A=J, 50=P,47=G
- Ascii Conversion = IMG001~2.JPG

```
           0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
0040F220  49 4D 47 30 30 31 7E 32 4A 50 47 20 00 1E 0F 6F   IMG001~2JPG ...o
0040F230  62 46 63 53 00 00 59 84 65 41 63 00 F2 D5 01 00   bFcS..Y.eAc.....
```

Figure 5.2: 20th Entry

```
PS C:\sleuthkit-4.12.1-win32\bin> .\istat.exe -o 8064 .\Coursework-1.001 20
Directory Entry: 20
Allocated
File Attributes: File, Archive
Size: 120306
Name: IMG001~2.JPG
```

Figure 5.2: Verification – istat shows the entry number along with the Allocated filename.

C) Date and time the file was created.

```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
0040F220  49 4D 47 30 30 31 7E 32 4A 50 47 20 00 1E 0F 6F  IMG001~2JPG█...o
0040F230  62 46 63 53 00 00 59 84 65 41 63 00 F2 D5 01 00  bFcS..Y.eAc.....
```

Figure 5.3: 20th Entry

Calculating creation time for the file:

- Bytes 14-15 = 0x 0F 6F
- Reverse endian = 0x 6F 0F
- Binary = 0b 0110 1111 0000 1111

Splitting up to represent seconds, minutes, hours:

- Bits 0 - 4 = Second = 1111 = 15 * 2 = 30
- Bits 5 - 10 =Minute = 000111 = 56
- Bits 11-15 = Hour =1101 = 13
- The creation time is at: 13:56:30

```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
0040F220  49 4D 47 30 30 31 7E 32 4A 50 47 20 00 1E 0F 6F  IMG001~2JPG█...o
0040F230  62 46 63 53 00 00 59 84 65 41 63 00 F2 D5 01 00  bFcS..Y.eAc.....
```

Figure 5.3: Creation Date Hex.

Calculating creation date for the file:

- Bytes 16-17= 0x 62 46
- Reverse endian = 0x 46 62
- Binary = 0b 0100 0110 0110 0010

Splitting up to represent seconds, minutes, hours:

- Bits 0 - 4 = Date = 00010 = 2
- Bits 5 - 8 = Month =0011= 3
- Bits 9-15 = Year= 100011= 35 + 1980 = 2015
- The creation date is: 02-03-2015.

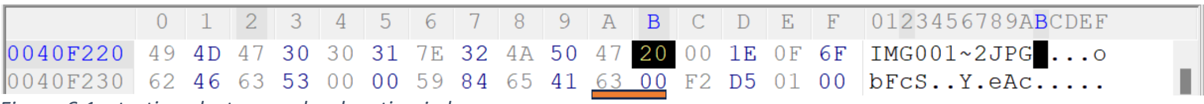
```
PS C:\sleuthkit-4.12.1-win32\bin> .\istat.exe -o 8064 .\Coursework-1.001 20
Directory Entry: 20
Allocated
File Attributes: File, Archive
Size: 120306
Name: IMG001~2.JPG

Directory Entry Times:
Written:        2012-11-05 16:34:50 (GMT Standard Time)
Accessed:       2021-11-03 00:00:00 (GMT Standard Time)
Created:        2015-03-02 13:56:30 (GMT Standard Time)
```

Figure 5.3: Verification – istat for the time and the date.

A) Start of File Content Location

Calculating the Starting Cluster Number:

- Bytes 26-27 = 0x 63 00
- Reverse endian – 0x 63
- Convert to decimal = 99 is Starting Cluster Number



```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
0040F220 49 4D 47 30 30 31 7E 32 4A 50 47 20 00 1E 0F 6F  IMG001~2JPG ...o
0040F230 62 46 63 53 00 00 59 84 65 41 63 00 F2 D5 01 00  bFcS..Y.eAc.....
```

Figure 6.1: starting cluster number location in hex.

Calculation Start of File Content Location:

To calculate the start of file content location we need:

- Bytes per sector (BPS) = 512
- Sectors per cluster (SPC) = 128
- Sector address for cluster Two = 280
- Starting Cluster no. of file = 99
- Offset to start of partition = 0x 3F0000 Bytes

The formula:

(Start cluster no. – 2) * SPC + Sector no. of cluster two.

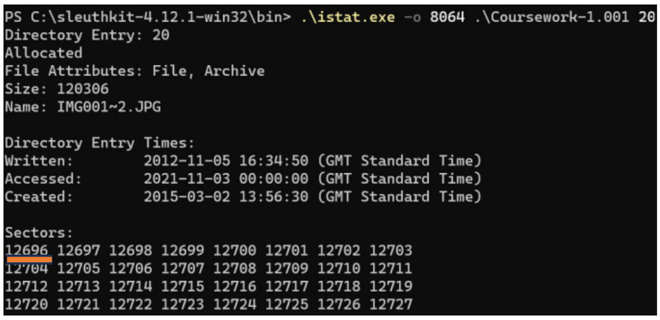
- (99 – 2) * 128 + 280 = 12696 sectors

This gives the sector number for where file starts.

Byte offset:

Sector where file starts * BPS:

- 12696 * 512 = 6500352
- Convert to hex = 0x 63 3000



```
PS C:\sleuthkit-4.12.1-win32\bin> .\istat.exe -o 8064 .\Coursework-1.001 20
Directory Entry: 20
Allocated
File Attributes: File, Archive
Size: 120306
Name: IMG001~2.JPG

Directory Entry Times:
Written:        2012-11-05 16:34:50 (GMT Standard Time)
Accessed:       2021-11-03 00:00:00 (GMT Standard Time)
Created:        2015-03-02 13:56:30 (GMT Standard Time)

Sectors:
12696 12697 12698 12699 12700 12701 12702 12703
12704 12705 12706 12707 12708 12709 12710 12711
12712 12713 12714 12715 12716 12717 12718 12719
12720 12721 12722 12723 12724 12725 12726 12727
```

Figure 6.1: Matches the starting offset I calculated.

Absolute offset:

- Need to add offset address for start of file system
- 0x 63 3000+ 0x 3F0000 = 0x A2 3000

A) End of File Content Location

Calculation:

- File size = 120,306 bytes
- Convert to hex = 0x 1 D5 F2

(Start of file + file size) – 1 = end of file content

(0x A2 3000 + 0x 1 D5 F2) – 0x 1 = 0x A4 05F1

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
00A405A0	C4	27	90	69	09	38	02	9A	48	69	FB	C2	0E	47	3C	E3	.'.i.8..Hi...G<.
00A405B0	D6	95	40	CB	8E	80	01	8A	68	21	BB	0C	0F	2F	91	93	..@.....h!.../..
00A405C0	93	41	63	E5	01	C7	5C	7E	B4	BA	82	D6	E3	80	C3	37	.Ac...\~.......7
00A405D0	5E	09	15	18	63	BF	1D	B1	9A	69	EA	38	3B	88	EC	78	^...c....i.8;..x
00A405E0	23	8F	98	0E	3E	B4	A5	89	52	DC	64	66	8D	C1	A5	63	#...>...R.df...c
00A405F0	FF	D9	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00A40600	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00A40610	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00A40620	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00A40630	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....

Figure 6.2: the mark of the last byte of the file content.

B) The allocated clusters for the file in the FAT & Verification

The allocated clusters for the file in the FAT

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
003F1000	F8	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	....
003F1010	FF	FF	FF	FF	0B	00	0C	00	FF	FF	0E	00	0F	00	10	00	....
003F1020	11	00	12	00	13	00	14	00	15	00	16	00	17	00	18	00	....
003F1030	19	00	1A	00	1B	00	1C	00	1D	00	1E	00	FF	FF	20	00	....

Figure 6.2: The allocated clusters for the file in the FAT in hex

- Reserved it indicates start of FAT area.
- Cluster allocated.

Cluster chains related to the file.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
003F1000	F8	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	....
003F1010	FF	FF	FF	FF	0B	00	0C	00	FF	FF	0E	00	0F	00	10	00	....
003F1020	11	00	12	00	13	00	14	00	15	00	16	00	17	00	18	00	....
003F1030	19	00	1A	00	1B	00	1C	00	1D	00	1E	00	FF	FF	20	00	....

Figure 6.2: Cluster chains related to the file in hex.

- There is one chain in the file it shows that the file continues until the EOF mark is present.

C) The two methods to extract the file content.

- There is number of ways for extracting file content from an image such as FTK Imager, Sleuth Kit – icat.

Sleuth Kit – icat.

- using the command “. /Icat -o” in to view the output file.

```
PS C:\sleuthkit-4.12.1-win32\bin> .\icat -o 8064 .\Coursework-1.001 5 > .\outputfile.txt
PS C:\sleuthkit-4.12.1-win32\bin>
```

Figure 6.3: File extraction - icat

- A text file will show in the directory as “Outputfile” when the command pass.

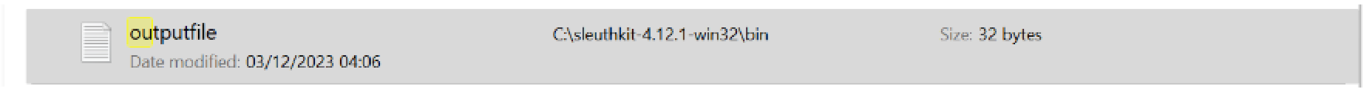


Figure 6.3: The txt file from the image showed in my directory.

- The content of the “Outputfile”.

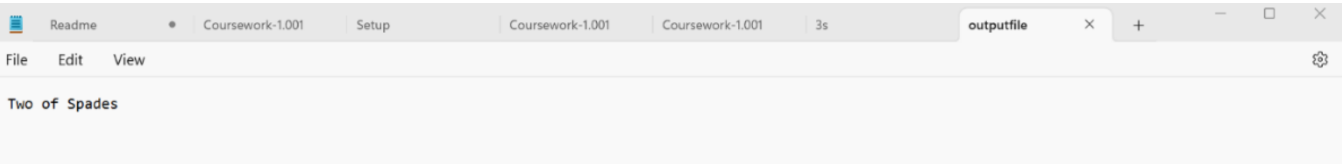


Figure 6.3: The file content from Sleuth Kit – icat match the FTK.

FTK Imager.

- Loading the image in FTK imager software and selecting FAT16 then downloading the root file content.

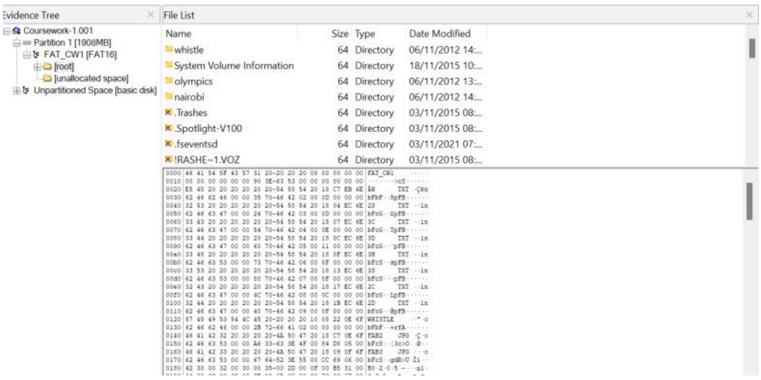


Figure 6.3: image loaded & exported.

- All the root file extracted will show in the download folder.

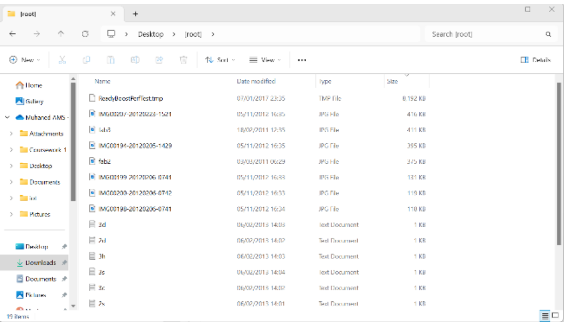


Figure 6.3: The root file extracted from FTK.

- The text file in the root content matches the Sleuth Kit – icat content.

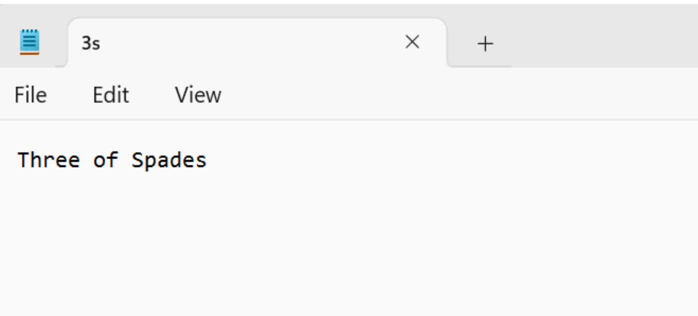


Figure 6.3: The root file content from FTK match the Sleuth Kit – icat.

Reference

1. Ben Filipkowski. (2023). " What are digital forensics and incident response (DFIR)?" Retrieved from: [https://fieldeffect.com/blog/digital-forensics-incident-response#9py65b] (Accessed: [20-11-2023]) Q1.

2. Security institute. (2023). "The Digital Forensics Process".
Retrieved from:
[https://www.esecurityinstitute.com/digital-forensics-process/]
(Accessed: [18-11-2023]) Q1.

3. David Waston, Andrew Jones. (2013). "Digital Forensics Processing and Procedures."
Retrieved from:
[https://shorturl.at/sxF01]
(Accessed: [19-11-2023]) Q1.