

# Digital Forensics

Federico Conti

2024/25

# Contents

<b>corrupted.dd</b>	<b>3</b>
Initial Setup . . . . .	3
Analysis Process . . . . .	4
Fix FAT Table . . . . .	5
zxgio . . . . .	5
Unlocated Space . . . . .	6

## corrupted.dd

This report details the forensic analysis conducted on the disk image `corrupted.dd`. The objective was to investigate the file system structure, identify signs of corruption, recover inaccessible data, and locate specific string patterns within the image.

### Initial Setup

```
file corrupted.dd
```

```
# Output:
```

```
DOS/MBR boot sector, code offset 0x3c+2, OEM-ID "mkfs.fat", Bytes/sector 2048, FATs 3, root entries
```

```
fsstat corrupted.dd
```

```
# Output:
```

```
...
File System Type: FAT12
OEM Name: mkfs.fat
Volume ID: 0xc8269037
Volume Label (Boot Sector): BILL
Volume Label (Root Directory): BILL
File System Type Label: FAT12
...
File System Layout (in sectors)
Total Range: 0 - 719
* Reserved: 0 - 0
** Boot Sector: 0
* FAT 0: 1 - 1
* FAT 1: 2 - 2
* FAT 2: 3 - 3
* Data Area: 4 - 719
** Root Directory: 4 - 11
** Cluster Area: 12 - 719
```

#### METADATA INFORMATION

```
-----
Range: 2 - 45831
Root Directory: 2
```

#### CONTENT INFORMATION

```
-----
Sector Size: 2048
Cluster Size: 2048
```

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
00000000:	EB	3C	90	6D	6B	66	73	2E	66	61	74	00	08	01	01	00	<.mkfs.fat....
00000010:	03	00	02	D0	02	F8	01	00	10	00	02	00	00	00	00	00	.....
00000020:	00	00	00	00	80	00	29	37	90	26	C8	42	49	4C	4C	20	.....)7.&BILL
00000030:	20	20	20	20	20	20	46	41	54	31	32	20	20	20	0E	1F	.....FAT12...
00000040:	BE	58	7C	AC	22	C0	74	08	56	84	0E	BB	07	00	CD	10	[["t.V.....
00000050:	5E	EB	F0	32	E4	CD	16	CD	19	EB	FE	54	68	69	73	20	^..2.....This
00000060:	69	73	20	6E	6F	74	20	61	20	62	6F	6F	74	61	62	6C	is not a bootabl
00000070:	65	20	64	69	73	6B	2E	20	20	50	6C	65	61	73	65	20	e disk. Please
00000080:	69	6E	73	65	72	74	20	61	20	62	6F	6F	74	61	62	6C	insert a bootabl
00000090:	65	20	66	6C	6F	70	70	79	20	61	6E	64	00	0A	70	72	e floppy and .pr
000000A0:	65	73	73	20	61	6E	79	20	68	65	79	20	74	6F	20	74	ess any key to t
000000B0:	72	79	20	61	67	61	69	6E	20	2E	2E	2E	20	00	0A	00	ry again ... ..
000000C0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000D0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000E0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000F0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000100:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000110:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000120:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000130:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000140:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000150:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000160:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....

```

1 struct bootSector {
2     u8  jumpInstruction[3];
3     char oemID[8];
4     u16 bytesPerSector;
5     u8  sectorsPerCluster;
6     u16 reservedSectors;
7     u8  numberOffFATs;
8     u16 rootEntries;
9     u16 totalSectors16;
10    u8  mediaDescriptor;
11    u16 sectorsPerFAT16;
12    u16 sectorsPerTrack;
13    u16 numberOffHeads;
14    u32 hiddenSectors;
15    u32 totalSectors32;
16    u8  driveNumber;
17    u8  reserved1;
18    u8  bootSignature;
19    u32 volumeSerialNumber;
20    char volumeLabel[11];
21    char fileSystemType[8];
22    u8  bootCode[0x10E];
23    u16 signature;
24 };

```

A sector 0 directly contains a FAT12 Boot Sector, so there is no MBR/GPT table listing partitions, and it is not a bootable image.

- **Volume Label:** BILL
- **Sector Size:** 2048
- **Cluster Size:** 2048
- **Num FATs:** 3

### Analysis Process

Using The Sleuth Kit (TSK):

```
fls -r -p corrupted.dd | grep '\.TXT'
```

# Output:

```

r/r 45: HOMEWORK.TXT
r/r 32: NETWORKS.TXT
r/r * 36: _EADME.TXT

```

- HOMEWORK.TXT** (pseudo-inode 45) > Status: Allocated
  - > Size: 6 bytes
  - > Sector: 619
  - > Readability: Can read it both by using `icat` and by mounting the image.
- NETWORKS.TXT** (pseudo-inode 32) > Status: Allocated
  - > Size: 17,465 bytes
  - > Starting Sector: 345
  - > Readability: Cannot read it by mounting the image, but you can read it using `icat`.  
Explanation: This indicates that the mounted file system has issues following the cluster chain, likely due to corruption in the FAT.
- EADME.TXT** (pseudo-inode 36) > Status: Deleted
  - > Size: 60,646 bytes
  - > Sectors: 457 to 486
  - > Readability: Since this file is deleted, it is expected not to appear in the mounted file system. However, you can recover it using `icat` if the data has not been overwritten.

```

istat corrupted.dd 32
# Output:
Directory Entry: 32 #pseudo inode by TSK
Allocated
File Attributes: File, Archive
Size: 17465
Name: NETWORKS.TXT
...
Sectors: 345

```

## Fix FAT Table

Objective: Rebuild the cluster chain in the FAT for corrupted files, particularly for NETWORKS.TXT.

Analyzing the first FAT it was discovered that FAT0 was overwritten with non-FAT data, likely a fragment of a GIF file.

```
xxd -s $((2048)) -l 2048 corrupted.dd | less
```

```

Address 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ASCII
00000800: 47 49 46 38 39 61 1F 00 1A 00 F7 0F 00 00 00 00 GIF89a.....
00000810: 6B B5 FF 6B CE FF 73 4A 21 73 F7 AD A5 73 C6 C6 k..k..sJ!s...s..
00000820: C6 C6 D6 8C 4A DE D6 D6 E7 E7 E7 F7 CE 84 FF 42 ...J.....B
00000830: 7B FF FF A5 FF FF B5 FF FF FF 00 00 00 00 00 {.....
00000840: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000850: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000860: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000870: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000880: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000890: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

```

28
29 struct FATArea{
30     u8 fat1[reservedArea.bytesPerSector]; //2048 bytes
31     u8 fat2[reservedArea.bytesPerSector];
32     u8 fat3[reservedArea.bytesPerSector];
33 };
34
35
36 FATArea fatArea @ reservedArea.bytesPerSector;
37

```

A known good copy of FAT2 was used to restore FAT0:

```

cp corrupted.dd corrupted_fixed.dd
dd if=corrupted_fixed.dd of=corrupted_fixed.dd bs=2048 skip=3 seek=1 count=1 conv=notrunc

```

After fixing:

- The cluster chain has been rebuilt.
- Both .TXT files can now be mounted correctly, allowing the recovery of the hash for NETWORKS.TXT.

```

sha256sum *.TXT
9b4a458763b06fefc65ba3d36dd0e1f8b5292e137e3db5dea9b1de67dc361311  HOMEWORK.TXT
e9207be4a1dde2c2f3efa3aeb9942858b6aaa65e82a9d69a8e6a71357eb2d03c  NETWORKS.TXT

```

## zxgio

Inside the file corrupted.dd, there are some occurrences of the string zxgio (without quotes). Below is the analysis:

```

strings -t d corrupted.dd | grep -E zxgio
512 zxgio
2832 zxgio
724025 zxgio
1267712 zxgio

```

From fsstat on intact image:

Offset (byte)	Sector	File System Area
512	0	Boot Sector
2832	1	FAT 0 (corrupted)

Offset (byte)	Sector	File System Area
724025	353	Cluster Area (slack space)
1267712	619	Cluster Area (inside HOMEWORK.TXT 619-619 (1) -> EOF)

### 1. Verify sector 353:

- Sector 353 contained the string in slack space (confirmed via dd and xxd).
- Offset 1,267,712 confirmed within HOMEWORK.TXT.
- No occurrence found within actual data of NETWORKS.TXT.

```
istat corrupted_fixed.dd 32
# Output:
Directory Entry: 32
Size: 17465
Name: NETWORKS.TXT
Sectors: 345 346 347 348 349 350 351 352 353
```

```
icat corrupted_fixed.dd 32 | grep zxgio
# Output:
NULL
```

### 2. Slack Space Inspection:

- It can be confirmed that the string is contained in the slack space of cluster 353

```
dd if=corrupted_fixed.dd bs=2048 skip=353 count=1 of=sector353.bin
xxd sector353.bin | less
# Output:
tware....zxgio..
```

## Unlocated Space

The image corrupted.dd has a size of 721 sectors, while the FAT12 file system only uses sectors 0-719. Sector 720, being outside the file system, was extracted and analyzed.

```
dd if=corrupted.dd bs=2048 skip=720 count=1 of=unused_sector720.bin
```

```
file unused_sector720.bin
```

```
# Output:
ASCII text
```

```
echo "ascii text" | base64 -d > hidden_file
```

```
file hidden_file
```

```
# Output:
GIF image data, version 89a, 86 x 33
```

Results:

- The sector contains a long ASCII string without line terminators.
- Analysis revealed it to be Base64 encoding.
- Decoding the string produced a file recognized as a GIF image.

