Chapter 10. FAT Data Structures

In the previous chapter, we examined the basic concepts of a FAT file system and how to analyze it. Now we are going to get more detailed and examine the data structures that make up FAT. This chapter will ignore the five-category model and instead focus on each individual data structure. This makes it easier to understand FAT because many of the data structures are in more than one category. I assume that you have already read Chapter 9, "FAT Concepts and Analysis," or that you are reading it in parallel. All the hexdumps and data shown in this chapter correspond to the data that were analyzed in Chapter 9 using tools from *The Sleuth Kit* (TSK).

Boot Sector

The boot sector is located in the first sector of FAT file system and contains the bulk of the file system category of data. FAT12/16 and FAT32 have different versions of the boot sector, but they both have the same initial 36 bytes. The data structure for the first 36 bytes is given in Table 10.1, and the data structures for the remaining bytes are given in Tables 10.2 and 10.3.

Table 10.1. Data structure for the first 36 bytes of the FAT boot sector.

Byte Range	Description	Essential
0–2	Assembly instruction to jump to boot code.	No (unless it is a bootable file system)
3–10	OEM Name in ASCII.	No
11–12	Bytes per sector. Allowed values include 512, 1024, 2048, and 4096.	Yes
13–13	Sectors per cluster (data unit). Allowed values are powers of 2, but the cluster size must be 32KB or smaller.	Yes
14–15	Size in sectors of the reserved area.	Yes
16–16	Number of FATs. Typically two for redundancy, but according to Microsoft it can be one for some small storage devices.	Yes
17–18	Maximum number of files in the root directory for FAT12 and FAT16. This is 0 for FAT32 and typically 512 for FAT16.	Yes
19–20	16-bit value of number of sectors in file system. If the number of sectors is larger than can be represented in this 2-byte value, a 4-byte value exists later in the data structure and this should be 0.	Yes
21–21	Media type. According to the Microsoft documentation, 0xf8 should be used for fixed disks and 0xf0 for removable.	No
22–23	16-bit size in sectors of each FAT for FAT12 and FAT16. For FAT32, this field is 0.	Yes
24–25	Sectors per track of storage device.	No
26–27	Number of heads in storage device.	No
28–31	Number of sectors before the start of partition.[1]	No

[1] My testing has shown that for file systems in an extended partition, Windows sets this value based on the beginning of the extended partition, not the beginning of the disk.

32–35	32-bit value of number of sectors in file system. Either this value or the 16-bit value above must be 0.	Yes

Table 10.2. Data structure for the remainder of the FAT12/16 boot sector.

Byte Range	Description	Essential
0–35	See Table 10.1.	Yes
36–36	BIOS INT13h drive number.	No
37–37	Not used.	No
38–38	Extended boot signature to identify if the next three values are valid. The signature is 0x29.	No
39–42	Volume serial number, which some versions of Windows will calculate based on the creation date and time.	No
43–53	Volume label in ASCII. The user chooses this value when creating the file system.	No
54–61	File system type label in ASCII. Standard values include "FAT," "FAT12," and "FAT16," but nothing is required.	No
62–509	Not used.	No
510–511	Signature value (0xAA55).	No

Table 10.3. Data structure for the remainder of the FAT32 boot sector.

Byte Range	Description	Essential
0–35	See Table 10.1.	Yes
36–39	32-bit size in sectors of one FAT.	Yes
40–41	Defines how multiple FAT structures are written to. If bit 7 is 1, only one of the FAT structures is active and its index is described in bits 0–3. Otherwise, all FAT structures are mirrors of each other.	Yes
42–43	The major and minor version number.	Yes
44–47	Cluster where root directory can be found.	Yes
48–49	Sector where FSINFO structure can be found.	No
50–51	Sector where backup copy of boot sector is located (default is 6).	No
52–63	Reserved.	No
64–64	BIOS INT13h drive number.	No
65–65	Not used.	No
66–66	Extended boot signature to identify if the next three values are valid. The signature is 0x29.	No
67–70	Volume serial number, which some versions of Windows will calculate based on the creation date and time.	No
71–81	Volume label in ASCII. The user chooses this value when creating the file system.	No
82–89	File system type label in ASCII. Standard values include "FAT32," but nothing is required.	No
90–509	Not used.	No

510–511	Signature value (0xAA55).	No

The first value in the boot sector, bytes 0 to 2, is a boot code instruction tells the computer where to find the code needed to boot the operating system. If the file system is not used to boot the computer, the value is not needed. You could use this value to identify what boot code is used. Note that DOS and Windows require that the value be set on non-bootable file systems, but other OSes, such a Linux, do not.

The media type value is used to identify if the file system is on fixed or removable media, but Microsoft Windows does not use it. A second copy of the media type exists in the file allocation table, and it is the one that Windows uses [Microsoft 2001]. The concepts of the other fields were discussed in Chapter 9.

From bytes 36 onward, FAT12 and FAT16 have a different layout than FAT32. The one value that they both have in common is the signature 0x55 in byte 510 and 0xAA in byte 511. Note that this is the same signature at the same location that the DOS partition table uses in its first sector (you'll also see it again in the first NTFS sector). The data structure values for the rest of the FAT12 and FAT16 boot sector are given in Table 10.2.

The data structure for the rest of the FAT32 boot sector is given in Table 10.3.

The difference between the FAT12/16 and FAT32 boot sector is that the FAT32 sector includes data to make the file system more scalable and flexible. There can be different policies for how the FAT structures are written to and a backup copy of the boot sector exists. There is also a version field, but there seems to be only one version used by Microsoft at the time of this writing.

The data between bytes 62 to 509 in a FAT12/16 file system, and bytes 90 to 509 in a FAT32 file system do not have a specified purpose, but are typically used to store boot code and error messages. Here is a hex dump of the first sector of a FAT32 file system from a Windows XP system:

```
# dcat -f fat fat-4.dd 0 | xxd
0000000: eb58 904d 5344 4f53 352e 3000 0202 2600  .X.MSDOS5.0...&.
0000016: 0200 0000 00f8 0000 3f00 4000 c089 0100  ........?.@.....
0000032: 4023 0300 1d03 0000 0000 0000 0200 0000  @#..............
0000048: 0100 0600 0000 0000 0000 0000 0000 0000  ................
0000064: 8000 2903 4619 4c4e 4f20 4e41 4d45 2020  ..).F.LNO NAME
0000080: 2020 4641 5433 3220 2020 33c9 8ed1 bcf4    FAT32 3.....
0000096: 7b8e c18e d9bd 007c 884e 028a 5640 b408  {......|.N..V@..
0000112: cd13 7305 b9ff ff8a f166 0fb6 c640 660f  ..s......f...@f.
0000128: b6d1 80e2 3ff7 e286 cdc0 ed06 4166 0fb7  ....?.......Af..
[REMOVED]
0000416: 0000 0000 0000 0000 0000 0000 0d0a 5265  ..............Re
0000432: 6d6f 7665 2064 6973 6b73 206f 7220 6f74  move disks or ot
0000448: 6865 7220 6d65 6469 612e ff0d 0a44 6973  her media....Dis
0000464: 6b20 6572 726f 72ff 0d0a 5072 6573 7320  k error...Press
0000480: 616e 7920 6b65 7920 746f 2072 6573 7461  any key to resta
0000496: 7274 0d0a 0000 0000 00ac cbd8 0000 55aa  rt............U.
```

The first line shows us that the OEM name is "MSDOS5.0," which may have been generated by a Windows 2000 or XP system. The data is written in little endian ordering, so the data structure fields that are numbers will appear in reverse order and strings will appear in the expected order. Bytes 11 to 12 show us that each sector is 512 bytes, (0x0200) and byte 13 shows us that the size of each cluster in the data area is 2 sectors, which is 1024 bytes. Bytes 14 to 15 show us that there are 38 (0x0026) sectors in the reserved area, so we know that the FAT area will start in sector 38, and byte 16 shows that there are two FAT structures. Bytes 19 to 20 contain the 16-bit file system size value and it is 0, which means that the 32-bit field