

Table of Contents

Examination of Disk.....	2
Master Boot Record (MBR).....	2
How to find the beginning of a partition.....	3
NTFS.....	8
Volume Boot Record.....	8
Master File Table (\$MFT file).....	13
STANDARD_INFORMATION block (0x10).....	16
FILE_NAME (0x30).....	17
Resident DATA (0x80).....	18
An example to resident file.....	19
Non resident DATA (0x80).....	26
Example to non resident DATA.....	26
Metasploit timestomp.....	36
Slack space (Metasploit slacker).....	41
Microsoft Dynamic disk (Simple volume).....	53
Structure of the VBLK blocks.....	62
VBLK partition descriptor (0x33).....	63
Microsoft dynamic disks (RAID0 stripe).....	65
VBLK Disk descriptor (0x34).....	73
VBLK Component descriptor (0x32).....	77
Reassemble of a file.....	79
Microsoft dynamic disk RAID5.....	89

Examination of Disk

The hard disk is organized to sectors in hardware level. The sector size is usually 512 (0x200) bytes, some newer disks has the sector size of 4096 (0x1000).

Master Boot Record (MBR)

When we open the disk with a hex editor in the first (to be more exact in the zeroth) sector we are going to find the **Master Boot Record**. It is always 512 (0x200) bytes long. The First 446 (from 0 until 0x1BD) bytes of it contains the bootstarp code. For us it is not really important now. After it comes four 16 (0x10) bytes long partition entries. After these partition entries there is a 2 (0x02) bytes long constant magic value 0x55AA, always this marks the end of the **Master Boot Record**. On the next picture you can see the structure of the MBR and the structure of the partition entries:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x0000	BOOT Starp Code															
0x0010																
0x0020																
0x0030																
0x0040																
0x0050																
0x0060																
0x0070																
0x0080																
0x0090																
0x00A0																
0x00B0																
0x00C0																
0x00D0																
0x00E0																
0x00F0																
0x0100																
0x0110																
0x0120																
0x0130																
0x0140																
0x0150																
0x0160																
0x0170																
0x0180																
0x0190																
0x01A0													First			
0x01B0																
0x01C0	Partition Entry												second			
0x01D0	Partition Entry												third			
0x01E0	Partition Entry												fourth			
0x01F0	Partition Entry												55	AA		

0123456789ABCEDF

Bootable	start of the partition in CHS			Partition type	End of the partition in CHS			Start of partition in sectors (LBA notation)				Length of partition in sectors (LBA notation)			
----------	-------------------------------	--	--	----------------	-----------------------------	--	--	--	--	--	--	---	--	--	--

Some important partition types (0x04): 0x07 NTFS, 0x04 FAT16, 0x0B FAT32 0x0C FAT32 0x82 Linux swap 0x83 linux nativ
Possible values of bootable byte (0x00): 0x80 yes, 0x00 not

From the partition entries today the LBA notation used to be used, because of the disk sizes, so we will deal mainly with those values. The bytes from 0x08 to 0x0B shows to us the start of the partition (it is marked with light blue color on the previous picture). It is a little endian number so you should read the bytes in opposite order.

How to find the beginning of a partition

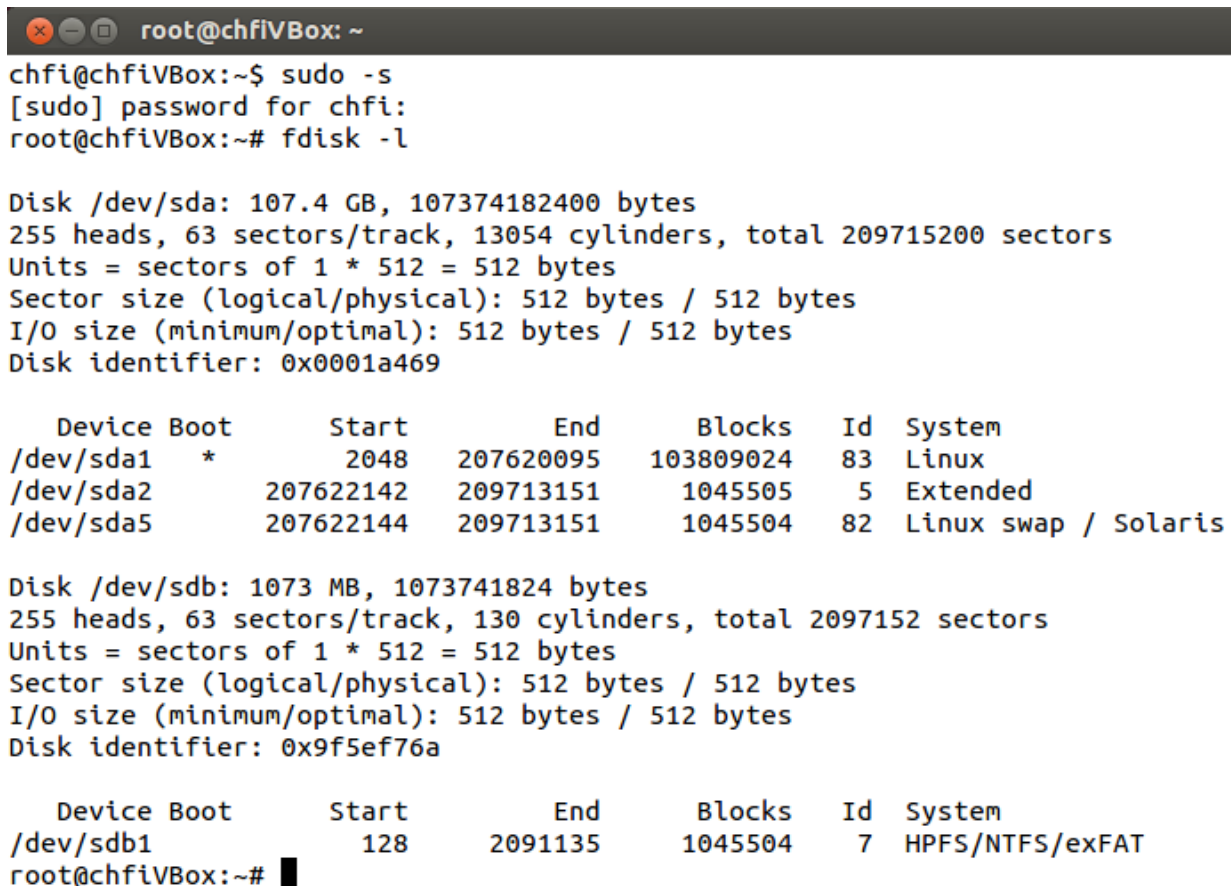
Let us see in a real example, how we can find the start of a partition. Attach the `evidenceNTFSresident.vhd` to your examination virtual machine, then start it.

Log in, then open a terminal. Most of our tool requires root privileges because it want to reach the disk in low level, so first we should become root:

```
sudo -s
```

then we list the disks attached to our environment:

```
fdisk -l
```



```
root@chfiVBox: ~
chfi@chfiVBox:~$ sudo -s
[sudo] password for chfi:
root@chfiVBox:~# fdisk -l

Disk /dev/sda: 107.4 GB, 107374182400 bytes
255 heads, 63 sectors/track, 13054 cylinders, total 209715200 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x0001a469

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1  *        2048     20762095    103809024   83   Linux
/dev/sda2                207622142    209713151     1045505    5   Extended
/dev/sda5                207622144    209713151     1045504   82   Linux swap / Solaris

Disk /dev/sdb: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders, total 2097152 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x9f5ef76a

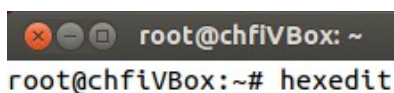
   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1                128      2091135     1045504    7   HPFS/NTFS/exFAT
root@chfiVBox:~#
```

In this case the sdb disk is our attached disk, and it has sector size of 512 bytes as usual. Let us open it with a hex editor. If it is not installed yet you can install one by the following command:

```
apt-get install hexedit
```

then we start the application with the command

```
hexedit
```



```
root@chfiVBox:~# hexedit
```

The application asks you, which file to open. We should open the /dev/sdb file:

A dark gray terminal window header with three window control icons (red close, yellow maximize, green minimize) on the left and the text "root@chfIVBox: ~" on the right.

root@chfIVBox: ~

File name: /dev/sdb█

Then we get the file opened. Here we should search for the first partition entry, it will start at the 0x01BE position:

```

root@chfivBox: ~
00000080  9F 83 C4 10 9E EB 14 B8 01 02 BB 00 7C 8A 56 00 .....|.V.
00000090  8A 76 01 8A 4E 02 8A 6E 03 CD 13 66 61 73 1E FE .v..N..n...fas..
000000A0  4E 11 0F 85 0C 00 80 7E 00 80 0F 84 8A 00 B2 80 N.....~.....
000000B0  EB 82 55 32 E4 8A 56 00 CD 13 5D EB 9C 81 3E FE ..U2..V...]>.
000000C0  7D 55 AA 75 6E FF 76 00 E8 8A 00 0F 85 15 00 B0 }U.un.v.....
000000D0  D1 E6 64 E8 7F 00 B0 DF E6 60 E8 78 00 B0 FF E6 ..d.....`x....
000000E0  64 E8 71 00 B8 00 BB CD 1A 66 23 C0 75 3B 66 81 d.q.....f#.u;f.
000000F0  FB 54 43 50 41 75 32 81 F9 02 01 72 2C 66 68 07 .TCPAu2....r,fh.
00000100  BB 00 00 66 68 00 02 00 00 66 68 08 00 00 00 66 ...fh....fh....f
00000110  53 66 53 66 55 66 68 00 00 00 00 66 68 00 7C 00 SfSfUfh....fh|.
00000120  00 66 61 68 00 00 07 CD 1A 5A 32 F6 EA 00 7C 00 .fah.....Z2...|.
00000130  00 CD 18 A0 B7 07 EB 08 A0 B6 07 EB 03 A0 B5 07 .....
00000140  32 E4 05 00 07 8B F0 AC 3C 00 74 FC BB 07 00 B4 2.....<.t....
00000150  0E CD 10 EB F2 2B C9 E4 64 EB 00 24 02 E0 F8 24 .....+.d..$...$
00000160  02 C3 49 6E 76 61 6C 69 64 20 70 61 72 74 69 74 ..Invalid partit
00000170  69 6F 6E 20 74 61 62 6C 65 00 45 72 72 6F 72 20 ion table.Error
00000180  6C 6F 61 64 69 6E 67 20 6F 70 65 72 61 74 69 6E loading operatin
00000190  67 20 73 79 73 74 65 6D 00 4D 69 73 73 69 6E 67 g system.Missing
000001A0  20 6F 70 65 72 61 74 69 6E 67 20 73 79 73 74 65 operating syste
000001B0  6D 00 00 00 00 62 7A 99 6A F7 5E 9F 00 00 00 02 m....bz.j.^.....
000001C0  03 00 07 FE 3F 81 80 00 00 00 00 E8 1F 00 00 00 ....?.....
000001D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000001E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000001F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA .....U.
00000200  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
--- sdb --0x1BE/0x40000000-----

```

The partition entry now is the following:

```
00 02 03 00 07 FE 3F 81 80 00 00 00 00 E8 1F 00
```

We need the start of the partition what is 80 00 00 00. It is in little endian notation so the value will be: 0x00000080.

This value is given is sectors. We know the sector size from the fdisk -l command, it was 512 bytes. So the starting position of the partition is $0x80 * 512 = 128 * 512 = 65536 = 0x1000$

We can jump to this position by pressing ctrl-g, then type this value:

```

root@chfivBox: ~
00000080  9F 83 C4 10 9E EB 14 B8 01 02 BB 00 7C 8A 56 00 .....|.V.
00000090  8A 76 01 8A 4E 02 8A 6E 03 CD 13 66 61 73 1E FE .v..N..n...fas..
000000A0  4E 11 0F 85 0C 00 80 7E 00 80 0F 84 8A 00 B2 80 N.....~.....
000000B0  EB 82 55 32 E4 8A 56 00 CD 13 5D EB 9C 81 3E FE ..U2..V...]>.
000000C0  7D 55 AA 75 6E FF 76 00 E8 8A 00 0F 85 15 00 B0 }U.un.v.....
000000D0  D1 E6 64 E8 7F 00 B0 DF E6 60 E8 78 00 B0 FF E6 ..d.....`.x....
000000E0  64 E8 71 00 B8 00 BB CD 1A 66 23 C0 75 3B 66 81 d.q.....f#.u;f.
000000F0  FB 54 43 50 41 75 32 81 F9 02 01 72 2C 66 68 07 .TCPAu2....r,fh.
00000100  BB 00 00 66 68 00 02 00 00 66 68 08 00 00 00 66 ...fh....fh....f
00000110  53 66 53 66 55 66 68 00 00 00 00 66 68 00 7C 00 SfSfUfh....fh.|.
00000120  00 66 61 68 00 00 07 CD 1A 5A 32 F6 EA 00 7C 00 .fah.....Z2...|.

                                New position ? 0x10000█

00000160  02 C3 49 6E 76 61 6C 69 64 20 70 61 72 74 69 74 ..Invalid partit
00000170  69 6F 6E 20 74 61 62 6C 65 00 45 72 72 6F 72 20 ion table.Error
00000180  6C 6F 61 64 69 6E 67 20 6F 70 65 72 61 74 69 6E loading operatin
00000190  67 20 73 79 73 74 65 6D 00 4D 69 73 73 69 6E 67 g system.Missing
000001A0  20 6F 70 65 72 61 74 69 6E 67 20 73 79 73 74 65 operating syste
000001B0  6D 00 00 00 00 62 7A 99 6A F7 5E 9F 00 00 00 02 m....bz.j.^.....
000001C0  03 00 07 FE 3F 81 80 00 00 00 00 E8 1F 00 00 00 ....?.....
000001D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000001E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000001F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA .....U.
00000200  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
--- sdb --0x1BE/0x40000000-----

```

There we will find the beginning of an NTFS partition

```

root@chflVBox: ~
00010000 EB 52 90 4E 54 46 53 20 20 20 20 00 02 08 00 00 .R.NTFS .....
00010010 00 00 00 00 00 F8 00 00 3F 00 FF 00 80 00 00 00 .....?.....
00010020 00 00 00 00 80 00 80 00 FF E7 1F 00 00 00 00 00 .....
00010030 55 54 01 00 00 00 00 00 7F FE 01 00 00 00 00 00 UT.....
00010040 F6 00 00 00 01 00 00 00 C6 D0 C4 F4 F3 C4 F4 DC .....
00010050 00 00 00 00 FA 33 C0 8E D0 BC 00 7C FB 68 C0 07 .....3.....|.h..
00010060 1F 1E 68 66 00 CB 88 16 0E 00 66 81 3E 03 00 4E ..hf.....f.>..N
00010070 54 46 53 75 15 B4 41 BB AA 55 CD 13 72 0C 81 FB TFSu..A..U..r...
00010080 55 AA 75 06 F7 C1 01 00 75 03 E9 D2 00 1E 83 EC U.u.....u.....
00010090 18 68 1A 00 B4 48 8A 16 0E 00 8B F4 16 1F CD 13 .h...H.....
000100A0 9F 83 C4 18 9E 58 1F 72 E1 3B 06 0B 00 75 DB A3 .....X.r.;...u..
000100B0 0F 00 C1 2E 0F 00 04 1E 5A 33 DB B9 00 20 2B C8 .....Z3...+.
000100C0 66 FF 06 11 00 03 16 0F 00 8E C2 FF 06 16 00 E8 f.....
000100D0 40 00 2B C8 77 EF B8 00 BB CD 1A 66 23 C0 75 2D @.+..w.....f#.u-
000100E0 66 81 FB 54 43 50 41 75 24 81 F9 02 01 72 1E 16 f..TCPAu$....r..
000100F0 68 07 BB 16 68 70 0E 16 68 09 00 66 53 66 53 66 h...hp..h..fSfSf
00010100 55 16 16 16 68 B8 01 66 61 0E 07 CD 1A E9 6A 01 U...h..fa....j.
00010110 90 90 66 60 1E 06 66 A1 11 00 66 03 06 1C 00 1E ..f`..f...f.....
00010120 66 68 00 00 00 00 66 50 06 53 68 01 00 68 10 00 fh....fP.Sh..h..
00010130 B4 42 8A 16 0E 00 16 1F 8B F4 CD 13 66 59 5B 5A .B.....fY[Z
00010140 66 59 66 59 1F 0F 82 16 00 66 FF 06 11 00 03 16 fYfY.....f.....
00010150 0F 00 8E C2 FF 0E 16 00 75 BC 07 1F 66 61 C3 A0 .....u...fa..
00010160 F8 01 E8 08 00 A0 FB 01 E8 02 00 EB FE B4 01 8B .....
00010170 F0 AC 3C 00 74 09 B4 0E BB 07 00 CD 10 EB F2 C3 ..<.t.....
00010180 0D 0A 41 20 64 69 73 6B 20 72 65 61 64 20 65 72 ..A disk read er
--- sdb --0x10180/0x40000000-----

```

NTFS

Volume Boot Record

- Now let us see how the NTFS volume builds up. The beginning of the volume contains a short relative jump (marked with green 0x00..0x01), to jump over the configuration information (EB XX). The length of the configuration information usually 0x54 bytes long, but never forgot, the jump instruction jumps from the end of the jump instruction always. So if it is 0x54 bytes long you will see an EB 52 instruction there.
- After it at the 0x02 position comes a NOP (0x90) instruction.
- The next important piece of information for us is the length of sectors given in bytes (marked with yellow at position 0x0B..0x0C) as we were talked about it earlier, it is practically always 512 (0x0200) bytes, and given in little endian notation so you will see it as 0x0002.
- Then comes the Cluster length (marked with light blue at position 0x0D). It is measured in sectors. The default cluster length of the cluster is 4096 (0x1000) bytes.
- The next very important information we need is the start position of the \$MFT file (marked with red at the position 0x30..0x37). Practically the \$MFT file (Master File Table) describes the whole filesystem, the directory structure, every files, timestamps, clusters used by it and so on. It means if we want to read the filesystem we should parse this file. This number is again in little endian format, and **it is measured in cluster, not in blocks now**, so if you see there for example 0x5554010000000000 it means the \$MFT file will start at cluster 0x015455. This offset is measured from the beginning of the volume.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0x0000	Short Jump (EB XX) the jump length may vary		NOP	OEM ID (NTFS)									Bytes per sector		Sectors per Cluster	Reserved Sectors	
0x0010	Always 0			Not used ?		Media Descriptor	Always 0			Sectors per track		Number of Heads		Hidden Sectors			
0x0020	Not used by NTFS ?				Not used by NTFS ?				Total number of sectors								
0x0030	Start Cluster of \$MFT fájl									Start Cluster of \$MFTMirr fájl							
0x0040	Clusters Per File Record Segment				Clusters Per Index Block					Volume Serial Number							
0x0050	Checksum			BOOT Starp Code													
0x0060																	
0x0070																	
0x0080																	
0x0090																	
0x00A0																	
0x00B0																	
0x00C0																	
0x00D0																	
0x00E0																	
0x00F0																	
0x0100																	
0x0110																	
0x0120																	
0x0130																	
0x0140																	
0x0150																	
0x0160																	
0x0170																	
0x0180																	
0x0190																	
0x01A0																	
0x01B0																	
0x01C0																	
0x01D0																	
0x01E0																	
0x01F0														55	AA		

Let us examine it on the previous example:

```

root@chfIVBox: ~
00010000 EB 52 90 4E 54 46 53 20 20 20 20 00 02 08 00 00 .R.NTFS .....
00010010 00 00 00 00 00 00 F8 00 00 3F 00 FF 00 80 00 00 00 .....?.....
00010020 00 00 00 00 80 00 80 00 FF E7 1F 00 00 00 00 00 .....
00010030 55 54 01 00 00 00 00 00 7F FE 01 00 00 00 00 00 UT.....
00010040 F6 00 00 00 01 00 00 00 C6 D0 C4 F4 F3 C4 F4 DC .....
00010050 00 00 00 00 FA 33 C0 8E D0 BC 00 7C FB 68 C0 07 .....3.....|.h..
00010060 1F 1E 68 66 00 CB 88 16 0E 00 66 81 3E 03 00 4E ..hf.....f.>..N
00010070 54 46 53 75 15 B4 41 BB AA 55 CD 13 72 0C 81 FB TFSu..A..U..r...
00010080 55 AA 75 06 F7 C1 01 00 75 03 E9 D2 00 1E 83 EC U.u.....u.....
00010090 18 68 1A 00 B4 48 8A 16 0E 00 8B F4 16 1F CD 13 .h...H.....
000100A0 9F 83 C4 18 9E 58 1F 72 E1 3B 06 0B 00 75 DB A3 .....X.r.;...u..
000100B0 0F 00 C1 2E 0F 00 04 1E 5A 33 DB B9 00 20 2B C8 .....Z3...+.
000100C0 66 FF 06 11 00 03 16 0F 00 8E C2 FF 06 16 00 E8 f.....
000100D0 40 00 2B C8 77 EF B8 00 BB CD 1A 66 23 C0 75 2D @.+..w.....f#.u-
000100E0 66 81 FB 54 43 50 41 75 24 81 F9 02 01 72 1E 16 f..TCPAu$....r..
000100F0 68 07 BB 16 68 70 0E 16 68 09 00 66 53 66 53 66 h...hp..h..fSfSf
00010100 55 16 16 16 68 B8 01 66 61 0E 07 CD 1A E9 6A 01 U...h..fa....j.
00010110 90 90 66 60 1E 06 66 A1 11 00 66 03 06 1C 00 1E ..f`...f...f....
00010120 66 68 00 00 00 00 66 50 06 53 68 01 00 68 10 00 fh....fP.Sh..h..
00010130 B4 42 8A 16 0E 00 16 1F 8B F4 CD 13 66 59 5B 5A .B.....fY[Z
00010140 66 59 66 59 1F 0F 82 16 00 66 FF 06 11 00 03 16 fYfY.....f.....
00010150 0F 00 8E C2 FF 0E 16 00 75 BC 07 1F 66 61 C3 A0 .....u...fa..
00010160 F8 01 E8 08 00 A0 FB 01 E8 02 00 EB FE B4 01 8B .....
00010170 F0 AC 3C 00 74 09 B4 0E BB 07 00 CD 10 EB F2 C3 ..<.t.....
00010180 0D 0A 41 20 64 69 73 6B 20 72 65 61 64 20 65 72 ..A disk read er
--- sdb --0x10180/0x40000000-----

```

The block size marked with 0002. As we remember it should be treated as little endian number $0x0200 = 512$ bytes.

Then let's check the cluster size marked with yellow 08. It means the cluster size is the default $0x08 * 0x0200 = 8 * 512 = 4096$ bytes.

The next piece of information we need the start position of the \$MFT file. It is marked with red 5554010000000000. Because it is a little endian value the position is $0x015455$, and it is measured in clusters so we should multiply it by 4096 ($0x1000$). We get: $0x15455000$. But it is measured from the beginning of the partition not from the beginning of the disk, so we should add to it $0x10000$ (the NTFS volume starts at that position). So finally we get $0x15465000$. If we jump to this position by pressing ctrl-g then typing $0x15465000$ we arrive to the beginning of the \$MFT file.

```

root@chflVBox: ~
15465000  46 49 4C 45 30 00 03 00 8D 10 20 00 00 00 00 00 FILE0.....
15465010  01 00 01 00 38 00 01 00 98 01 00 00 00 04 00 00 ....8.....
15465020  00 00 00 00 00 00 00 00 06 00 00 00 00 00 00 00 .....
15465030  02 00 00 00 00 00 00 00 10 00 00 00 60 00 00 00 .....
15465040  00 00 18 00 00 00 00 00 48 00 00 00 18 00 00 00 .....H.....
15465050  5A B9 47 C4 9E 18 CC 01 5A B9 47 C4 9E 18 CC 01 Z.G....Z.G....
15465060  5A B9 47 C4 9E 18 CC 01 5A B9 47 C4 9E 18 CC 01 Z.G....Z.G....
15465070  06 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
15465080  00 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 .....
15465090  00 00 00 00 00 00 00 00 30 00 00 00 68 00 00 00 .....0...h...
154650A0  00 00 18 00 00 00 03 00 4A 00 00 00 18 00 01 00 .....J.....
154650B0  05 00 00 00 00 00 05 00 5A B9 47 C4 9E 18 CC 01 .....Z.G....
154650C0  5A B9 47 C4 9E 18 CC 01 5A B9 47 C4 9E 18 CC 01 Z.G....Z.G....
154650D0  5A B9 47 C4 9E 18 CC 01 00 40 00 00 00 00 00 00 Z.G....@.....
154650E0  00 40 00 00 00 00 00 00 06 00 00 00 00 00 00 00 .@.....
154650F0  04 03 24 00 4D 00 46 00 54 00 00 00 00 00 00 00 ..$.M.F.T.....
15465100  80 00 00 00 48 00 00 00 01 00 40 00 00 00 01 00 ....H....@.....
15465110  00 00 00 00 00 00 00 00 0F 00 00 00 00 00 00 00 .....
15465120  40 00 00 00 00 00 00 00 00 00 01 00 00 00 00 00 @.....
15465130  00 00 01 00 00 00 00 00 00 00 01 00 00 00 00 00 .....
15465140  31 10 55 54 01 00 4C 91 B0 00 00 00 48 00 00 00 1.UT..L....H...
15465150  01 00 40 00 00 00 05 00 00 00 00 00 00 00 00 00 ..@.....
15465160  00 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 .....@.....
15465170  00 10 00 00 00 00 00 00 08 00 00 00 00 00 00 00 .....
15465180  08 00 00 00 00 00 00 00 31 01 54 54 01 00 00 00 .....1.TT....
15465190  FF FF FF FF 00 00 00 00 00 00 01 00 00 00 00 00 .....
154651A0  00 00 01 00 00 00 00 00 31 10 55 54 01 00 4C 91 .....1.UT..L.
154651B0  B0 00 00 00 48 00 00 00 01 00 40 00 00 00 05 00 ....H....@.....
154651C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
154651D0  40 00 00 00 00 00 00 00 00 10 00 00 00 00 00 00 @.....
--- sdb --0x154651D0/0x40000000-----

```

Master File Table (\$MFT file)

The entries in the \$MFT builds up on the following way:

One entry is always 1024 bytes (0x400) bytes long.

- The entry starts with a header what is always 42 bytes long (0x2A)
 - The first four bytes of the entry (0x00..0x03) in this header is a **magic value: FILE** (46494C45)
 - Then first important value for us now is the **offset to the first attribute block** (marked with dark blue at the position 0x14..0x15) it is a little endian number again so if you see here 3800 it means 0x38.
 - Immediately after it comes the **flags** (marked with light blue at position 0x16..0x17) it is also a little endian number. The two bits what we should know now are the **0.th bit** if it is **1** the **file is allocated**, if **0** the file is **deleted**. The **1.th bit** if **1** then the entry is a **directory**, if **0** then the entry is a **file**.
 - The next four bytes (marked with dark blue at position 0x18..0x1B) means the **real size of this entry**. Again it is a little endian number so if you find here 98010000 then it means 0x0198

- The next four bytes (marked with light blue again at position **0x0C..0x0F**) is the **allocated size of this entry**. It is practically always 1024 bytes (0x1000).
- From offset given at the position 0x14..0x15 we will find the first attribute block, and there will be many other attribute blocks. The structure of them depends on their types, but there are some common things, they always start with a 16 (0x10) bytes long attribute header:
 - The first four bytes of these headers gives us the type of the block, it is in little endian number.
 - The second four numbers gives us the length of the block, it is also in little endian notation.

By the help of these two values we can travel through all the blocks in the entry.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x0000	MAGIC value (FILE)				Offset to update sequence		Size of update sequence		\$Logfile Sequence Number (LSN)							
0x0010	sequence number (incremented when the entry is allocated or unallocated)		link count : number of directories have entries for this record (hard links increment)		Offset to first Block (Attribute)		Flags 0. bit : 1 allocated 0 deleted; 1. bit : 1 directory 0 fájl		Real size of the FILE record				Allocated size of the FILE record			
0x0020	File reference to the base FILE record								Next Attribute Id							
0x0030					BLOCK 1											
0x0040																
0x0050																
0x0060																
0x0070					BLOCK 2											
0x0080																
0x0090																
0x00A0																
0x00B0					BLOCK 3											
0x00C0																
0x00D0																
0x00E0																
0x00F0					BLOCK n											
:																
:																
:																
0x0350					BLOCK n+1											
0x0360																
0x0370																
0x0380																
0x0390	BLOCK n+1															
0x03A0																
0x03B0																
0x03C0																
0x03D0	BLOCK n+1															
0x03E0																
0x03F0																

STANDARD_INFORMATION block (0x10)

This block is always resident, and contains the DOS flags, Modify Access Create and Entry times of the file. These time informations can not be hundred percently trusted, because some evidence eliminator applications for example the metasploit timestomp modify these entries.

The values which are important for us now are:

- **0x00..0x03** : Attribute type, in case of Standard information it is 10000000
- **0x04..0x07** : the length of this attribute entry.
- **0x08** : resident flag (0 means resident data, 1 means non resident). The standard information entry is always resident
- **0x09** : Length of stream name, most of the time it is zero, because it is not stored, if standard attribute
- **0x0A..0x0B** : offset to stream name
- **0x18..0x1F** : creation time it is given in windows 64 bit time format what is defined as: 10^7 secundum intervals from 0h 1-Jan 1601. You can make it readable by the help of `w32tm /ntte` command. For example if it is 8A A7 81 D8 9E 18 CC 01 one should use the command `w32tm /ntte 0x01CC189ED881A78A`. It is easy to recognise the windows 64 bit time format, an 0 byte number, always ends with 01. most of the time four three times used to be seen.
- **0x20..0x27** : Last modification time of the file given in windows 64 bit time format.
- **0x28..0x2F** : Last modification time of the MFT entry given in windows 64 bit time format.
- **0x30..0x37** : Last access time of the file. Again it is given in windows 64 bit time format.
- **0x38..0x3B** : DOS file permissions.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x0000	Attribute type (Standard Information10000000)				Length of this attribute entry including the header				Resident flag	Length of stream name	Offset to stream name		Flags		Attribute identifier	
0x0010	Length of data from the end of header				Offset to attribute content		Padding 00, if length stream name not 0 after it the stream name		Creation Time in windows 64 bit time format							
0x0020	Last Modification Time in windows 64 bit time format								Last Modification Time of the File Record in windows 64 bit time format							
0x0030	Last Access Time in windows 64 bit time format								DOS File Permissions (flags)				Maximum number of versions			
0x0040	Version number								Class ID				Owner ID			
0x0050	Security ID				Quota charged				Update Sequence Number (UCN)							

He of course you can find the file name, and again we will find here four time stamps. It is more difficult to

- **0x20..0x27** : creation time it is given in windows 64 bit time format what is defined as: 10^7 secundum intervals from 0h 1-Jan 1601. You can make it readable by the help of `w32tm /ntte` command. For example if it is 8A A7 81 D8 9E 18 CC 01 one should use the command `w32tm /ntte 0x01CC189ED881A78A`. It is easy to recognise the windows 64 bit time format, an 0 byte number, always ends with 01. most of the time four three times used to be seen.
- **0x28..0x2F** : Last modification time of the file given in windows 64 bit time format.
- **0x30..0x37** : Last modification time of the MFT entry given in windows 64 bit time format.
- **0x38..0x3F** : Last access time of the file. Again it is given in windows 64 bit time format.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0x0000	Attribute type (Filename 30000000)				Length of this attribute entry including the header itself				Resident flag	Length of stream name	Offset to stream name		Flags		Attribute identifier		
0x0010	Length of data from the end of header				Offset to attribute content		Padding 00, if length stream name not 0 after it the stream name		MFT record number of the parent directory								
0x0020	Creation Time in windows 64 bit time format								Last Modification Time in windows 64 bit time format								
0x0030	Last Modification Time of the File Record in windows 64 bit time format								Last Access Time in windows 64 bit time format								
0x0040	Allocated size of index								Real size of index								
0x0050	Flags				Reparse value				Filename length	Filename name space	Filename in unicode						
0x0060	so the length of this part is the double of the stored filename length																
0x0070																	

Resident DATA (0x80)

This value can be resident or non resident. We start with the structure of the resident one, in this case we will find here the content of the file.

- **0x00..0x03** : Attribute type, in case of Standard information it is 10000000
- **0x04..0x07** : the length of this attribute entry.
- **0x08** : resident flag (0 means resident data, 1 means non resident). The DATA attribute can be resident or non resident, first we deal with the resident files.
- **0x10..0x13** : Real size of the file.
- **0x14..0x17** : offset to the file content

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0x0000	Attribute type (Data 80000000)				Length of this attribute entry including the header				Resident flag	Length of stream name	Offset to stream name		Flags		Attribute identifier		
0x0010	Real size of data				Offset to file content												
0x0020	File Content																
0x0030																	
0x0040																	
0x0050																	

An example to resident file

The first some entry in the \$MFT file are system reserved. Our files start after those. Let us find one resident file.

\$Mft	0
\$MftMirr	1
\$LogFile	2
\$Volume	3
\$AttrDef	4
\$	5
\$Bitmap	6
\$Boot	7
\$BadClus	8
\$Secure	9
\$Upcase	10
\$Extend	11
reserved	12-
\$Quota	24
\$ObjId	25
\$Reparse	26
\$Rmetadata	27
\$Repair	28
\$txflog	29
\$txf	30
\$tops	31
\$txflogblf	32
\$txflogcontainer	33
\$txflogcontainer 2	34

Just go down, until you find the first non system entry (we will find it at position $0x15465000 + 35 * 0x400 = 0x1546DC00$):

- After we found the MFT entry of the file we can start to parse it. First we should find the offset of the first attribute entry. It is at the position 0x14..0x15 marked with blue color. It has a value of 3800 and it is a little endian number so it is 0x38. It means the first attribute will start at the position 0x38.
- Really from the 0x38 position we find a standard:information block we know it from the first four what bytes are 10000000 it is marked with light blue on the picture.
- The next four bytes (0x3C..3F marked with dark blue) is the length of this attribute now the value of it is 60000000 what is little endian number so 0x60 bytes. It means the second attribute will start at the position $0x38 + 0x60 = 0x98$
- The next byte (0x40 marked with green) is the resident flag it is now 0, what means resident so this attribute is stored in the NTFS entry. It is not suprising, because the standard information is always resident.
- We find here the four timestamps (0x50..0x6F marked with grey). First is the modification time. Now it is: 8AA781D89E18CC01 we can read the value by typing `w32tm /ntte 0x01CC189ED881A78A` The second is the last modification time, now it is 7EED04EA9E18CC01 again one can make it readable by typing `w32tm /ntte 0x01CC189EEA04ED7E`. The next timestamp is the last modification time of the MFT

entry, the value of it now 7EED04EA9E18CC01 again to see the value run the `w32tm /ntte 0x01CC189EEA04ED7E` command. The last one is the last access time of the file, now it is 8AA781D89E18CC01 again if we want to read it run the `w32tm /ntte 0x01CC189ED881A78A` command. Again I want to mention some evidence eliminator (like the actual version of measploit timestamp) deletes or modifies only this value.

- At the position 0x98 we find a filename attribute because the four first bytes are 30000000 it is marked with light blue on the picture.
- The next four bytes (0x9C..0x9F marked with dark blue) shows us the length now the value of it is 68000000. It is a little endian number so the value 0x68. It means the next attribute will start at $0x98 + 0x68 = 0x100$
- The next byte (0xA0 marked with green) is the resident flag it is 0 what means resident so this attribute is stored in the NTFS entry. It is not suprising, because the standard information is always resident.
- We find here the four timestamps (0xB8..0xD8 marked with grey). First is the modification time. Now it is: 8AA781D89E18CC01 we can read the value by typing `w32tm /ntte 0x01CC189ED881A78A` The second is the last modification time, now it is 8AA781D89E18CC01 again one can make it readable by typing `w32tm /ntte 0x01CC189ED881A78A`. The next timestamp is the last modification time of the MFT entry, the value of it now 8AA781D89E18CC01 again to see the value run the `w32tm /ntte 0x01CC189ED881A78A` command. The last one is the last access time of the file, now it is 8AA781D89E18CC01 again if we want to read it run the `w32tm /ntte 0x01CC189ED881A78A` command. Again I want to mention some evidence eliminator (like the actual version of measploit timestamp) deletes or modifies only this value.
- At the position 0x100 we find a Data attribute because the first four bytes of it (0x100..0x103) are 80000000 it is marked with light blue on the picture
- The next four bytes (0x104..0x107) shows us the length of this attribute, what is now 30000000, it means 0x30 bytes.
- The next byte (0x109 marked with green) is the resident flag it is now 0, what means resident so this attribute is stored in the NTFS entry.
- At the position 0x110..0x113 marked with yellow on the picture we find the length of the data.
- After it the next four bytes (0x114..0x117 marked with black on the picture) shows the offset of the file content, it is 18000000 now. It is a little endian number so 0x18. The content of file will start at $0x118. 0x1546DC00 + 0x118 = 0x1546DD18$.
- From the position 0x1546DD18 we should read 0x16 bytes that is the content of the file marked with red.

```

root@chfIVBox: ~
1546DC00 46 49 4C 45 30 00 03 00 B8 26 20 00 00 00 00 00 FILE0....& .....
1546DC10 01 00 01 00 38 00 01 00 38 01 00 00 00 04 00 00 ....8...8.....
1546DC20 00 00 00 00 00 00 00 00 05 00 00 00 23 00 00 00 .....#...
1546DC30 04 00 00 00 00 00 00 00 10 00 00 00 60 00 00 00 .....`...
1546DC40 00 00 00 00 00 00 00 00 48 00 00 00 18 00 00 00 .....H.....
1546DC50 8A A7 81 D8 9E 18 CC 01 7F ED 04 EA 9E 18 CC 01 .....
1546DC60 7F ED 04 EA 9E 18 CC 01 8A A7 81 D8 9E 18 CC 01 .....
1546DC70 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546DC80 00 00 00 00 05 01 00 00 00 00 00 00 00 00 00 00 .....
1546DC90 00 00 00 00 00 00 00 00 30 00 00 00 68 00 00 00 .....0...h...
1546DCA0 00 00 00 00 00 00 04 00 4C 00 00 00 18 00 01 00 .....L.....
1546DCB0 05 00 00 00 00 00 05 00 8A A7 81 D8 9E 18 CC 01 .....
1546DCC0 8A A7 81 D8 9E 18 CC 01 8A A7 81 D8 9E 18 CC 01 .....
1546DCD0 8A A7 81 D8 9E 18 CC 01 00 00 00 00 00 00 00 00 .....
1546DCE0 00 00 00 00 00 00 00 00 20 00 00 00 00 00 00 00 .....
1546DCF0 05 03 61 00 2E 00 74 00 78 00 74 00 58 00 7E 00 ..a...t.x.t.X.~.
1546DD00 80 00 00 00 30 00 00 00 00 00 18 00 00 00 01 00 ....0.....
1546DD10 16 00 00 00 18 00 00 00 61 61 61 61 61 61 61 61 .....aaaaaaa
1546DD20 61 61 0D 0A 61 61 61 61 61 61 61 61 61 61 47 11 aa..aaaaaaaaaG.
1546DD30 FF FF FF FF 82 79 47 11 8A A7 81 D8 9E 18 CC 01 .....yG.....
1546DD40 8A A7 81 D8 9E 18 CC 01 8A A7 81 D8 9E 18 CC 01 .....
1546DD50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546DD60 20 00 00 00 00 00 00 00 15 01 4E 00 65 00 77 00 .....N.e.w.
1546DD70 20 00 54 00 65 00 78 00 74 00 20 00 44 00 6F 00 .T.e.x.t. .D.o.
1546DD80 63 00 75 00 6D 00 65 00 6E 00 74 00 2E 00 74 00 c.u.m.e.n.t...t.
1546DD90 78 00 74 00 00 00 00 00 80 00 00 00 18 00 00 00 x.t.....
1546DDA0 00 00 18 00 00 00 01 00 00 00 00 00 18 00 00 00 .....
1546ddb0 FF FF FF FF 82 79 47 11 00 00 00 00 00 00 00 00 .....yG.....
1546DDC0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546DDD0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
--- sdb --0x1546DC00/0x40000000-----

```

Non resident DATA (0x80)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x0000	Attribute type (Data 80000000)				Length of this attribute entry including the header				Resident flag	Length of stream name	Offset to stream name		Flags		Attribute identifier	
0x0010	Starting VNC of the runlist								Last VCN of the runlist							
0x0020	Offset to the data runs		Compression Unit Size		Padding				Allocated size of the attribute (file)							
0x0030	Real size of the attribute (file)								Initialized data size							
0x0040	Compressed size								Cluster Chains							
0x0050																
0x0060																
0x0070																

Example to non resident DATA

Let us see in a real example, how we can find the start of a partition. Attach the evidenceNTFSnonresident.vhd to your examination virtual machine, then start it. This vhd contains only one file called as a.txt with a lot of letter “a” as content.

It is a new disk, so we start the process from the beginning, it will be good, to check if we memorized the partition table well.

First we need root right so run the `sudo -s` command, and give your password, to become root.

Then we run the `fdisk -l` command, to see the disks, and the block size of it.

```

root@chfiVBox: ~
chfi@chfiVBox:~$ sudo -s
[sudo] password for chfi:
root@chfiVBox:~# fdisk -l

```

For me this disk is the sdc the relevant part of the `fdisk -l` is on the next picture, as we can see the unit size is 512 again:


```

Disk /dev/sdc: 1073 MB, 1073741824 bytes
41 heads, 40 sectors/track, 1278 cylinders, total 2097152 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x8978eeed

   Device Boot      Start         End      Blocks   Id  System
/dev/sdc1           128       2091135     1045504    7   HPFS/NTFS/exFAT


```

then we start the hexedit application by typing hexedit (if someone like the color output then can start it as hexedit --color then the different kind of characters for example printable characters, non printables are marked with different color):

A terminal window with a dark background. The title bar shows three window control buttons (close, minimize, maximize) and the text 'root@chfivBox: ~'. The command prompt is 'root@chfivBox:~#' and the command 'hexedit' has been entered.

```
root@chfivBox:~# hexedit
```

We type the name of our disk for me /dev/sdc

A terminal window with a dark background. The title bar shows three window control buttons (close, minimize, maximize) and the text 'root@chfivBox: ~'. The command prompt is 'root@chfivBox:~#'. The text 'File name: /dev/sdc' is displayed on the screen.

```
root@chfivBox:~#  
File name: /dev/sdc
```

File name: /dev/sdc

At the beginning of the disk there will be the Master Boot Record (MBR):

```

root@chfIVBox: ~
00000000  33 C0 8E D0 BC 00 7C 8E C0 8E D8 BE 00 7C BF 00 3.....|.....|..
00000010  06 B9 00 02 FC F3 A4 50 68 1C 06 CB FB B9 04 00 .....Ph.....
00000020  BD BE 07 80 7E 00 00 7C 0B 0F 85 10 01 83 C5 10 ....~..|.....
00000030  E2 F1 CD 18 88 56 00 55 C6 46 11 05 C6 46 10 00 ....V.U.F...F..
00000040  B4 41 BB AA 55 CD 13 5D 72 0F 81 FB 55 AA 75 09 .A..U..]r...U.u.
00000050  F7 C1 01 00 74 03 FE 46 10 66 60 80 7E 10 00 74 ....t..F.f`.~.t
00000060  26 66 68 00 00 00 00 66 FF 76 08 68 00 00 68 00 &fh....f.v.h..h.
00000070  7C 68 01 00 68 10 00 B4 42 8A 56 00 8B F4 CD 13 |h..h...B.V.....
00000080  9F 83 C4 10 9E EB 14 B8 01 02 BB 00 7C 8A 56 00 .....|.V.
00000090  8A 76 01 8A 4E 02 8A 6E 03 CD 13 66 61 73 1E FE .v..N..n...fas..
000000A0  4E 11 0F 85 0C 00 80 7E 00 80 0F 84 8A 00 B2 80 N.....~.....
000000B0  EB 82 55 32 E4 8A 56 00 CD 13 5D EB 9C 81 3E FE ..U2..V...]>..
000000C0  7D 55 AA 75 6E FF 76 00 E8 8A 00 0F 85 15 00 B0 }U.un.v.....
000000D0  D1 E6 64 E8 7F 00 B0 DF E6 60 E8 78 00 B0 FF E6 ..d.....`.x....
000000E0  64 E8 71 00 B8 00 BB CD 1A 66 23 C0 75 3B 66 81 d.q.....f#.u;f.
000000F0  FB 54 43 50 41 75 32 81 F9 02 01 72 2C 66 68 07 .TCPAu2....r,fh.
00000100  BB 00 00 66 68 00 02 00 00 66 68 08 00 00 00 66 ...fh....fh...f
00000110  53 66 53 66 55 66 68 00 00 00 00 66 68 00 7C 00 SfsfUfh....fh.|.
00000120  00 66 61 68 00 00 07 CD 1A 5A 32 F6 EA 00 7C 00 .fah.....Z2...|.
00000130  00 CD 18 A0 B7 07 EB 08 A0 B6 07 EB 03 A0 B5 07 .....
00000140  32 E4 05 00 07 8B F0 AC 3C 00 74 FC BB 07 00 B4 2.....<.t....
00000150  0E CD 10 EB F2 2B C9 E4 64 EB 00 24 02 E0 F8 24 .....+.d..$...$
00000160  02 C3 49 6E 76 61 6C 69 64 20 70 61 72 74 69 74 ..Invalid partit
00000170  69 6F 6E 20 74 61 62 6C 65 00 45 72 72 6F 72 20 ion table.Error
00000180  6C 6F 61 64 69 6E 67 20 6F 70 65 72 61 74 69 6E loading operatin
00000190  67 20 73 79 73 74 65 6D 00 4D 69 73 73 69 6E 67 g system.Missing
000001A0  20 6F 70 65 72 61 74 69 6E 67 20 73 79 73 74 65 operating syste
000001B0  6D 00 00 00 00 62 7A 99 ED EE 78 89 00 00 00 02 m....bz...x....
000001C0  03 00 07 28 A8 06 80 00 00 00 00 E8 1F 00 00 00 ...(.....
000001D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000001E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000001F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA .....U.
--- sdc --0x0/0x40000000-----

```

We can find the first partition entry. It starts at position 0x1BE marked with blue.

The start position of the first partition is 80000000. Again it is little endian so 0x80. It is given in blocks so we multiply it by 512 (0x200) and we get 0x10000. To see this partition we should jump there by pressing ctrl-g then type 0x10000:


```

root@chfIVBox: ~
00000000  33 C0 8E D0 BC 00 7C 8E C0 8E D8 BE 00 7C BF 00 3.....|.....|..
00000010  06 B9 00 02 FC F3 A4 50 68 1C 06 CB FB B9 04 00 .....Ph.....
00000020  BD BE 07 80 7E 00 00 7C 0B 0F 85 10 01 83 C5 10 ....~..|.....
00000030  E2 F1 CD 18 88 56 00 55 C6 46 11 05 C6 46 10 00 ....V.U.F...F..
00000040  B4 41 BB AA 55 CD 13 5D 72 0F 81 FB 55 AA 75 09 .A..U..]r...U.u.
00000050  F7 C1 01 00 74 03 FE 46 10 66 60 80 7E 10 00 74 ....t..F.f`.~..t
00000060  26 66 68 00 00 00 00 66 FF 76 08 68 00 00 68 00 &fh....f.v.h..h.
00000070  7C 68 01 00 68 10 00 B4 42 8A 56 00 8B F4 CD 13 |h..h...B.V.....
00000080  9F 83 C4 10 9E EB 14 B8 01 02 BB 00 7C 8A 56 00 .....|.V.
00000090  8A 76 01 8A 4E 02 8A 6E 03 CD 13 66 61 73 1E FE .v..N..n...fas..
000000A0  4E 11 0F 85 0C 00 80 7E 00 80 0F 84 8A 00 B2 80 N.....~.....
000000B0  EB 82 55 32 E4 8A 56 00 CD 13 5D EB 9C 81 3E FE ..U2..V...]>..
000000C0  7D 55 AA 75 6E FF 76 00 E8 8A 00 0F 85 15 00 B0 }U.un.v.....
000000D0  D1 E6 64 E8 7F 00 B0 DF E6 60 E8 78 00 B0 FF E6 ..d.....`.x....
000000E0  64 E8 71 00 B8 00 BB CD 1A 66 23 C0 75 3B 66 81 d.q.....f#.u;f.

                                New position ? 0x10000█

00000120  00 66 61 68 00 00 07 CD 1A 5A 32 F6 EA 00 7C 00 .fah.....Z2...|.
00000130  00 CD 18 A0 B7 07 EB 08 A0 B6 07 EB 03 A0 B5 07 .....
00000140  32 E4 05 00 07 8B F0 AC 3C 00 74 FC BB 07 00 B4 2.....<.t....
00000150  0E CD 10 EB F2 2B C9 E4 64 EB 00 24 02 E0 F8 24 ....+..d..$...$
00000160  02 C3 49 6E 76 61 6C 69 64 20 70 61 72 74 69 74 ..Invalid partit
00000170  69 6F 6E 20 74 61 62 6C 65 00 45 72 72 6F 72 20 ion table.Error
00000180  6C 6F 61 64 69 6E 67 20 6F 70 65 72 61 74 69 6E loading operatin
00000190  67 20 73 79 73 74 65 6D 00 4D 69 73 73 69 6E 67 g system.Missing
000001A0  20 6F 70 65 72 61 74 69 6E 67 20 73 79 73 74 65 operating syste
000001B0  6D 00 00 00 00 62 7A 99 ED EE 78 89 00 00 00 02 m....bz...x....
000001C0  03 00 07 28 A8 06 80 00 00 00 00 E8 1F 00 00 00 ...(.
000001D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000001E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000001F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA .....U.
--- sdc          --0x0/0x40000000-----

```

There we find the Volume Boot Record again. We will need the following three informations:

```

root@chflVBox: ~
00010000 EB 52 90 4E 54 46 53 20 20 20 20 00 02 08 00 00 .R.NTFS .....
00010010 00 00 00 00 00 F8 00 00 3F 00 40 00 80 00 00 00 .....?.@.....
00010020 00 00 00 00 80 00 80 00 FF E7 1F 00 00 00 00 00 .....
00010030 55 54 01 00 00 00 00 00 7F FE 01 00 00 00 00 00 UT.....
00010040 F6 00 00 00 01 00 00 00 FE 9A CF 5C C0 CF 5C 94 .....\. \.
00010050 00 00 00 00 FA 33 C0 8E D0 BC 00 7C FB 68 C0 07 .....3....|.h..
00010060 1F 1E 68 66 00 CB 88 16 0E 00 66 81 3E 03 00 4E ..hf.....f.>..N
00010070 54 46 53 75 15 B4 41 BB AA 55 CD 13 72 0C 81 FB TFSu..A..U..r...
00010080 55 AA 75 06 F7 C1 01 00 75 03 E9 D2 00 1E 83 EC U.u.....u.....
00010090 18 68 1A 00 B4 48 8A 16 0E 00 8B F4 16 1F CD 13 .h...H.....
000100A0 9F 83 C4 18 9E 58 1F 72 E1 3B 06 0B 00 75 DB A3 .....X.r.;...u..
000100B0 0F 00 C1 2E 0F 00 04 1E 5A 33 DB B9 00 20 2B C8 .....Z3...+.
000100C0 66 FF 06 11 00 03 16 0F 00 8E C2 FF 06 16 00 E8 f.....
000100D0 40 00 2B C8 77 EF B8 00 BB CD 1A 66 23 C0 75 2D @.+..w.....f#.u-
000100E0 66 81 FB 54 43 50 41 75 24 81 F9 02 01 72 1E 16 f..TCPAu$.r..
000100F0 68 07 BB 16 68 70 0E 16 68 09 00 66 53 66 53 66 h...hp..h..fsfsf
00010100 55 16 16 16 68 B8 01 66 61 0E 07 CD 1A E9 6A 01 U...h..fa....j.
00010110 90 90 66 60 1E 06 66 A1 11 00 66 03 06 1C 00 1E ..f`..f...f.....
00010120 66 68 00 00 00 00 66 50 06 53 68 01 00 68 10 00 fh....fP.Sh..h..
00010130 B4 42 8A 16 0E 00 16 1F 8B F4 CD 13 66 59 5B 5A .B.....fy[Z
00010140 66 59 66 59 1F 0F 82 16 00 66 FF 06 11 00 03 16 fyfy....f.....
00010150 0F 00 8E C2 FF 0E 16 00 75 BC 07 1F 66 61 C3 A0 .....u...fa..
00010160 F8 01 E8 08 00 A0 FB 01 E8 02 00 EB FE B4 01 8B .....
00010170 F0 AC 3C 00 74 09 B4 0E BB 07 00 CD 10 EB F2 C3 ..<.t.....
00010180 0D 0A 41 20 64 69 73 6B 20 72 65 61 64 20 65 72 ..A disk read er
00010190 72 6F 72 20 6F 63 63 75 72 72 65 64 00 0D 0A 42 ror occurred...B
000101A0 4F 4F 54 4D 47 52 20 69 73 20 6D 69 73 73 69 6E 00TMGR is missin
000101B0 67 00 0D 0A 42 4F 4F 54 4D 47 52 20 69 73 20 63 g...BOOTMGR is c
000101C0 6F 6D 70 72 65 73 73 65 64 00 0D 0A 50 72 65 73 ompressed...Pres
000101D0 73 20 43 74 72 6C 2B 41 6C 74 2B 44 65 6C 20 74 s Ctrl+Alt+Del t
000101E0 6F 20 72 65 73 74 61 72 74 0D 0A 00 00 00 00 00 restart.....
000101F0 00 00 00 00 00 00 00 00 80 9D B2 CA 00 00 55 AA .....U.
--- sdc --- -0x10200/0x40000000-----

```

- **0x0B..0x0C** (marked with yellow) The sector size given in bytes (we already know it should be 512 = 0x200 bytes, but check it for the safety)
- **0x0D** (marked with blue): The size of cluster given in sectors. Now it is 0x08 so the size of the cluster is 0x200 * 0x08 = 0x1000 = 4096 bytes.
- **0x30..0x37** (marked with red): the starting position of the \$MFT given in clusters and measured from the start of the volume. The value here is 5554010000000000 So the position will be 0x015455 * 0x1000 + 0x10000 = 0x15465000. We can jump there by pressing ctrl-g then type 0x15465000

Here starts the \$MFT file, we recognize it from the “FILE” magic value:


```

root@chfiVBox: ~
15465000  46 49 4C 45 30 00 03 00 8D 10 20 00 00 00 00 00 FILE0.....
15465010  01 00 01 00 38 00 01 00 98 01 00 00 00 04 00 00 ....8.....
15465020  00 00 00 00 00 00 00 00 06 00 00 00 00 00 00 00 .....
15465030  02 00 00 00 00 00 00 00 10 00 00 00 60 00 00 00 .....`...
15465040  00 00 18 00 00 00 00 00 48 00 00 00 18 00 00 00 .....H.....
15465050  0E 05 16 CF 9F 18 CC 01 0E 05 16 CF 9F 18 CC 01 .....
15465060  0E 05 16 CF 9F 18 CC 01 0E 05 16 CF 9F 18 CC 01 .....
15465070  06 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
15465080  00 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 .....
15465090  00 00 00 00 00 00 00 00 30 00 00 00 68 00 00 00 .....0...h...
154650A0  00 00 18 00 00 00 03 00 4A 00 00 00 18 00 01 00 .....J.....
154650B0  05 00 00 00 00 00 05 00 0E 05 16 CF 9F 18 CC 01 .....
154650C0  0E 05 16 CF 9F 18 CC 01 0E 05 16 CF 9F 18 CC 01 .....
154650D0  0E 05 16 CF 9F 18 CC 01 00 40 00 00 00 00 00 00 .....@.....
154650E0  00 40 00 00 00 00 00 00 06 00 00 00 00 00 00 00 ..@.....
154650F0  04 03 24 00 4D 00 46 00 54 00 00 00 00 00 00 00 ..$.M.F.T.....
15465100  80 00 00 00 48 00 00 00 01 00 40 00 00 00 01 00 ....H.....@.....
15465110  00 00 00 00 00 00 00 00 0F 00 00 00 00 00 00 00 .....
15465120  40 00 00 00 00 00 00 00 00 00 01 00 00 00 00 00 @.....
15465130  00 00 01 00 00 00 00 00 00 00 01 00 00 00 00 00 .....
15465140  31 10 55 54 01 00 34 93 B0 00 00 00 48 00 00 00 1.UT..4....H...
15465150  01 00 40 00 00 00 05 00 00 00 00 00 00 00 00 00 ..@.....
15465160  00 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 .....@.....
15465170  00 10 00 00 00 00 00 00 08 00 00 00 00 00 00 00 .....
15465180  08 00 00 00 00 00 00 00 31 01 54 54 01 00 00 00 .....1.TT....
15465190  FF FF FF FF 00 00 00 00 00 00 01 00 00 00 00 00 .....
154651A0  00 00 01 00 00 00 00 00 31 10 55 54 01 00 34 93 .....1.UT..4.
154651B0  B0 00 00 00 48 00 00 00 01 00 40 00 00 00 05 00 ....H.....@.....
154651C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
154651D0  40 00 00 00 00 00 00 00 00 10 00 00 00 00 00 00 @.....
154651E0  08 00 00 00 00 00 00 00 08 00 00 00 00 00 00 00 .....
154651F0  31 01 54 54 01 00 00 00 FF FF FF FF 00 00 02 00 1.TT.....
--- sdc          --0x154651F0/0x40000000-----

```

As we remember the first values are system values so we jump to the 35th entry. One entry is always 1024 (0x400) bytes long. So we should go to $0x15465000 + 35 * 0x400 = 0x1546DC00$.

We can jump there by pressing ctrl-g and type 0x1546DC00

```

root@chfIVBox: ~
1546DC00 46 49 4C 45 30 00 03 00 13 28 20 00 00 00 00 00 FILE0....( .....
1546DC10 01 00 01 00 38 00 01 00 50 01 00 00 00 04 00 00 ....8...P.....
1546DC20 00 00 00 00 00 00 00 00 06 00 00 00 23 00 00 00 .....#...
1546DC30 03 00 00 00 00 00 00 00 10 00 00 00 60 00 00 00 .....`...
1546DC40 00 00 00 00 00 00 00 00 48 00 00 00 18 00 00 00 .....H.....
1546DC50 0A 80 55 2A A0 18 CC 01 1C 5A FB 7A A0 18 CC 01 ..U*....Z.z...
1546DC60 1C 5A FB 7A A0 18 CC 01 0A 80 55 2A A0 18 CC 01 .Z.z....U*...
1546DC70 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546DC80 00 00 00 00 05 01 00 00 00 00 00 00 00 00 00 00 .....
1546DC90 00 00 00 00 00 00 00 00 30 00 00 00 68 00 00 00 .....o...h...
1546DCA0 00 00 00 00 00 00 04 00 4C 00 00 00 18 00 01 00 .....L.....
1546DCB0 05 00 00 00 00 00 05 00 0A 80 55 2A A0 18 CC 01 .....U*...
1546DCC0 0A 80 55 2A A0 18 CC 01 0A 80 55 2A A0 18 CC 01 ..U*....U*...
1546DCD0 0A 80 55 2A A0 18 CC 01 00 00 00 00 00 00 00 00 ..U*.....
1546DCE0 00 00 00 00 00 00 00 00 20 00 00 00 00 00 00 00 .....
1546DCF0 05 03 61 00 2E 00 74 00 78 00 74 00 58 00 7E 00 ..a...t.x.t.X.~.
1546DD00 80 00 00 00 48 00 00 00 01 00 00 00 00 00 05 00 ....H.....
1546DD10 00 00 00 00 00 00 00 00 07 00 00 00 00 00 00 00 .....
1546DD20 40 00 00 00 00 00 00 00 00 80 00 00 00 00 00 00 @.....
1546DD30 80 7F 00 00 00 00 00 00 80 7F 00 00 00 00 00 00 .....
1546DD40 31 08 80 FE 01 00 34 93 FF FF FF FF 82 79 47 11 1....4....yG.
1546DD50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546DD60 20 00 00 00 00 00 00 00 15 01 4E 00 65 00 77 00 .....N.e.w.
1546DD70 20 00 54 00 65 00 78 00 74 00 20 00 44 00 6F 00 .T.e.x.t. .D.o.
1546DD80 63 00 75 00 6D 00 65 00 6E 00 74 00 2E 00 74 00 c.u.m.e.n.t...t.
1546DD90 78 00 74 00 00 00 00 00 80 00 00 00 18 00 00 00 x.t.....
1546DDA0 00 00 18 00 00 00 01 00 00 00 00 00 18 00 00 00 .....
1546ddb0 FF FF FF FF 82 79 47 11 00 00 00 00 00 00 00 00 .....yG.....
1546DDC0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546DDD0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546DDE0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546DDF0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 .....
--- sdc --0x1546DC00/0x40000000-----

```

- **0x14..0x15** (marked with dark blue) means the offset to the first attribute. Now it is 3800 what means 0x38 because it is in little endian notation
- **0x38..0x3B** (marked with light blue) is the type of the attribute 10000000 means 0x10 what is the standard information block
- **0x3C..0x3F** (marked with dark blue) is the length of this attribute. Now it is 60000000 what means 0x60 bytes, so the next attribute going to start at the position 0x38 0x60 = 0x98
- **0x40** (marked with green) is the resident flag. It is 0 obviously, because the standard information block is always resident.
- **0x50..0x6F** (marked with grey) contains the time stamps.
 - First is the modification time. Now it is: 0A80552AA018CC01 we can read the value by typing `w32tm /ntte 0x01CC18A02A55800A`
 - The second is the last modification time, now it is 1C5AFB7AA018CC01 again one can make it readable by typing `w32tm /ntte 0x01CC18A07AFB5A1C`.
 - The next timestamp is the last modification time of the MFT entry, the value of it now 1C5AFB7AA018CC01 again to see the value run the `w32tm /ntte 0x01CC18A07AFB5A1C` command.
 - The last one is the last access time of the file, now it is 0A80552AA018CC01 again if we want to read it run the `w32tm /ntte 0x01CC18A02A55800A` command.

- *Again I want to mention some evidence eliminator (like the actual version of measploit timestomp) deletes or modifies only these values.*
- **0x98..0x9B** (marked with light blue) : contains the second attribute. The value of it is 30000000 means 0x30 file_name attribute.
- **0x9C..0x9F** (marked with dark blue) is the length of this attribute. The value of it now is 68000000, but it is given in little endian format, what means 0x68. So the third attribute starts at 0x98 0x68 = 0x0100
- **0xA0** (marked with green) is the resident flag. Again this attribute is always resident so not surprising we get a value 0 here.
- **0xB8..0xD8** : (marked with grey) contains the time stamps.
 - First is the modification time. Now it is: 0A80552AA018CC01 we can read the value by typing `w32tm /ntte 0x01CC18A02A55800A`
 - The second is the last modification time, now it is 0A80552AA018CC01 again one can make it readable by typing `w32tm /ntte 0x01CC18A02A55800A`.
 - The next timestamp is the last modification time of the MFT entry, the value of it now 0A80552AA018CC01 again to see the value run the `w32tm /ntte 0x01CC18A02A55800A` command.
 - The last one is the last access time of the file, now it is 0A80552AA018CC01 again if we want to read it run the `w32tm /ntte 0x01CC18A02A55800A` command.
- **0x0100..0x0103** (marked with light blue) : here starts the third attribute. Now it has a value 80000000 means 0x80 DATA.
- **0x0104..0x0107** (marked with dark blue) : the length of this attribute now contains 48000000 what means 0x48 bytes. So the next attribute starts at 0x0100 0x48 = 0x0148
- **0x0108** (marked with green) : is the resident flag, what is now 1, because we have a non resident file.
- **0x0120..0x0123** (marked with dark blue) : is the offset where the cluster chains definition going to start. Now it has a value 40000000 what means 0x40. So the description of the cluster chain starts at **0x0140** (marked with black) : here starts the cluster chain. Now it has a value 0x31 what should be interpreted on the following way: the first number 3 (0x03) means the start cluster is stored on 3 bytes. The second number 1 (0x01) means the number of the clusters continuously occupied by the file. First the number of clusters is stored so we should read the next one byte
- **0x0141** (marked with yellow) : now it is 0x08 so the first cluster chain occupies 0x08 continuous clusters. If we calculate $0x08 * 0x1000 = 0x8000 = 32768$ bytes. The start cluster is stored on three bytes as we remember so we should read the next three bytes:
- **0x0142..0x0144** (marked with red) : is the start cluster of the file measured from the beginning of the volume. Now it has the value 80FE01 means 0x01FE80. So the file will begin at the position $0x01FE80 * 0x1000 + 0x10000 = 0x1FE90000$.
- **0x130..0x138** (marked with purple) : is the real size of the file. It has a value 807F000000000000 it is in little endian so the value is 0x7F80 so the filesize is 32640 bytes. We can see that it is smaller than we calculated from the number of clusters so the last cluster will not be fully utilized.

Now let us jump to the position 0x1FE90000 by pressing ctrl-g then type 0x1FE90000, and really there we will find the content of the file a lot of letter "a" and sometimes a carriage return line feed:

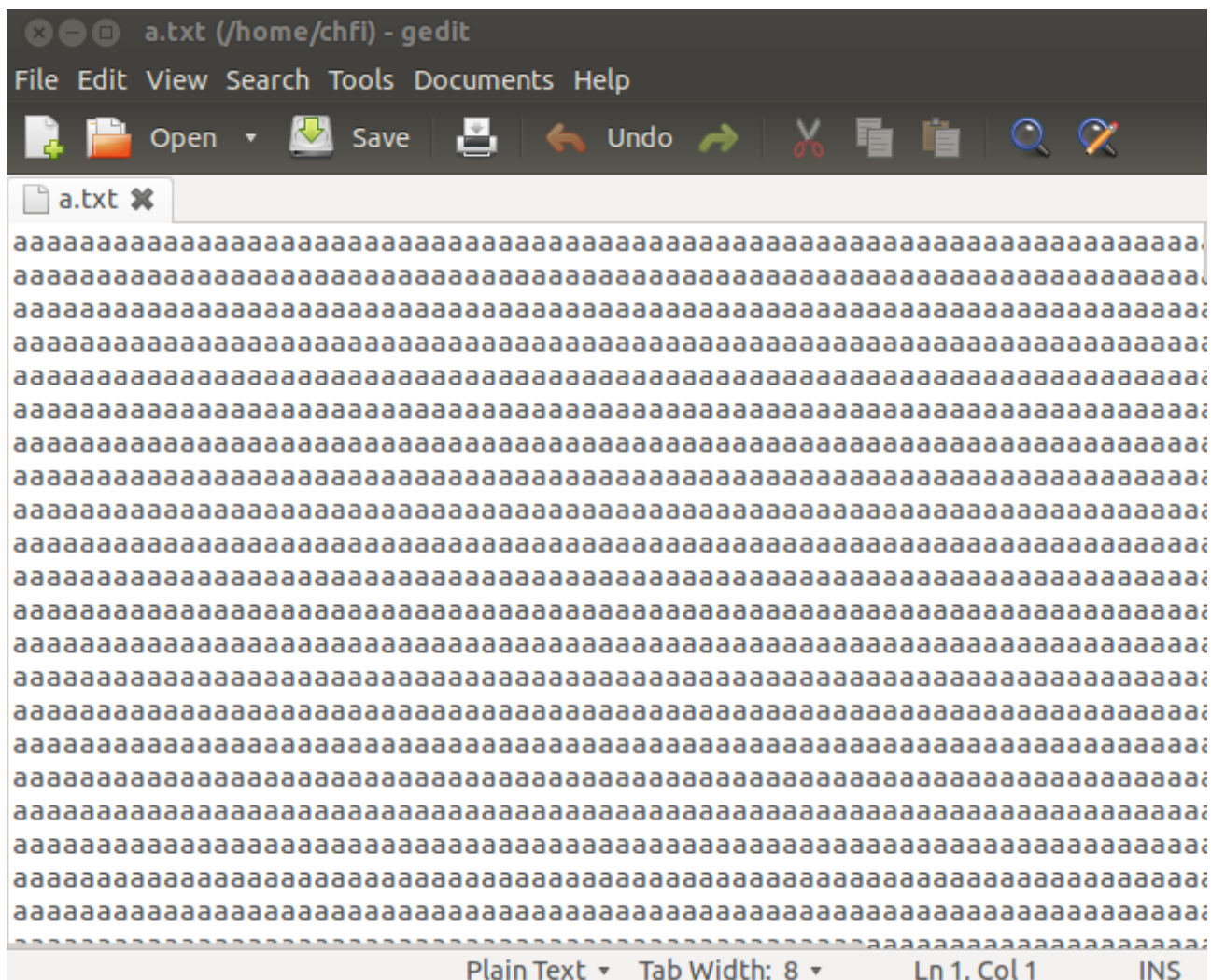
```
root@chfiVBox: ~
1FE90000  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
1FE90010  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
1FE90020  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
1FE90030  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
1FE90040  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
1FE90050  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
1FE90060  61 61 61 61 0D 0A 61 61 61 61 61 61 61 61 61 61 aaaa. .aaaaaaaaaa
1FE90070  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
1FE90080  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
1FE90090  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
1FE900A0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
1FE900B0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
1FE900C0  61 61 61 61 61 61 61 61 61 61 61 0D 0A 61 61 61 61 aaaaaaaaa. .aaaa
1FE900D0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
1FE900E0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
1FE900F0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
1FE90100  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
1FE90110  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
1FE90120  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
1FE90130  0D 0A 61 61 61 61 61 61 61 61 61 61 61 61 61 61 . .aaaaaaaaaaaaaaaa
1FE90140  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
1FE90150  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
1FE90160  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
1FE90170  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
1FE90180  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
1FE90190  61 61 61 61 61 61 0D 0A 61 61 61 61 61 61 61 61 aaaaaa. .aaaaaaaa
1FE901A0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
1FE901B0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
1FE901C0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
1FE901D0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
1FE901E0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
1FE901F0  61 61 61 61 61 61 61 61 61 61 61 61 0D 0A 61 61 aaaaaaaaaaaaa. .aa
--- sdc          --0x1FE90000/0x40000000-----
```

to extract the content of the file we can use for example a simpel dd command as:

```
dd if=/dev/sdc of=./a.txt bs=1 skip=535363584 count=32640
```

```
root@chfiVBox: ~
root@chfiVBox:~#
root@chfiVBox:~# dd if=/dev/sdc of=./a.txt bs=1 skip=535363584 count=32640
32640+0 records in
32640+0 records out
32640 bytes (33 kB) copied, 0.298404 s, 109 kB/s
root@chfiVBox:~#
root@chfiVBox:~#
```

after it we can open the newly restore a.txt by any text editor like nano, gedit, vi...



Metasploit timestomp

The timestomp command in metasploit capable to delete the time values from the standard information block in the \$MFT file, to make more difficult to analyze an evidence disk. The current version of the tool deletes only the values from the standard information block, so we can get some time value back by the help of the times in the filename block as it is shown in the next example

to try it attach the evidenceTimeStomp.vhd to your examination machine (for me it is attached as /dev/sde) then open it with a hexeditor. As usually the first sector contains the MBR we can start to search for the NTFS partition on the already well known way:

```
root@chfivBox: ~
00000000  33 C0 8E D0 BC 00 7C 8E C0 8E D8 BE 00 7C BF 00 3.....|.....|..
00000010  06 B9 00 02 FC F3 A4 50 68 1C 06 CB FB B9 04 00 .....Ph.....
00000020  BD BE 07 80 7E 00 00 7C 0B 0F 85 10 01 83 C5 10 ....~..|.....
00000030  E2 F1 CD 18 88 56 00 55 C6 46 11 05 C6 46 10 00 ....V.U.F...F..
00000040  B4 41 BB AA 55 CD 13 5D 72 0F 81 FB 55 AA 75 09 .A..U..]r...U.u.
00000050  F7 C1 01 00 74 03 FE 46 10 66 60 80 7E 10 00 74 ....t..F.f`.~..t
00000060  26 66 68 00 00 00 00 66 FF 76 08 68 00 00 68 00 &fh....f.v.h..h.
00000070  7C 68 01 00 68 10 00 B4 42 8A 56 00 8B F4 CD 13 |h..h...B.V.....
00000080  9F 83 C4 10 9E EB 14 B8 01 02 BB 00 7C 8A 56 00 .....|..V.
00000090  8A 76 01 8A 4E 02 8A 6E 03 CD 13 66 61 73 1E FE .v..N..n...fas..
000000A0  4E 11 0F 85 0C 00 80 7E 00 80 0F 84 8A 00 B2 80 N.....~.....
000000B0  EB 82 55 32 E4 8A 56 00 CD 13 5D EB 9C 81 3E FE ..U2..V...]>...
000000C0  7D 55 AA 75 6E FF 76 00 E8 8A 00 0F 85 15 00 B0 }U.un.v.....
000000D0  D1 E6 64 E8 7F 00 B0 DF E6 60 E8 78 00 B0 FF E6 ..d.....`.x....
000000E0  64 E8 71 00 B8 00 BB CD 1A 66 23 C0 75 3B 66 81 d.q.....f#.u;f.
000000F0  FB 54 43 50 41 75 32 81 F9 02 01 72 2C 66 68 07 .TCPAu2....r,fh.
00000100  BB 00 00 66 68 00 02 00 00 66 68 08 00 00 00 66 ...fh....fh....f
00000110  53 66 53 66 55 66 68 00 00 00 00 66 68 00 7C 00 Sfsfufh....fh.|.
00000120  00 66 61 68 00 00 07 CD 1A 5A 32 F6 EA 00 7C 00 .fah.....Z2...|.
00000130  00 CD 18 A0 B7 07 EB 08 A0 B6 07 EB 03 A0 B5 07 .....
00000140  32 E4 05 00 07 8B F0 AC 3C 00 74 FC BB 07 00 B4 2.....<.t....
00000150  0E CD 10 EB F2 2B C9 E4 64 EB 00 24 02 E0 F8 24 .....+..d..$...$
00000160  02 C3 49 6E 76 61 6C 69 64 20 70 61 72 74 69 74 ..Invalid partit
00000170  69 6F 6E 20 74 61 62 6C 65 00 45 72 72 6F 72 20 ion table.Error
00000180  6C 6F 61 64 69 6E 67 20 6F 70 65 72 61 74 69 6E loading operatin
00000190  67 20 73 79 73 74 65 6D 00 4D 69 73 73 69 6E 67 g system.Missing
000001A0  20 6F 70 65 72 61 74 69 6E 67 20 73 79 73 74 65 operating syste
000001B0  6D 00 00 00 00 62 7A 99 76 AA 34 56 00 00 00 02 m....bz.v.4V....
000001C0  03 00 07 FE 3F 81 80 00 00 00 00 E8 1F 00 00 00 ....?.....
000001D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000001E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000001F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA .....U.
--- sde --0x0/0x40000000---
```

We can find the first partition entry. It starts at position 0x01BE marked with blue.

The start position of the first partition is 80000000. Again it is little endian so 0x80. It is given in blocks so we multiply it by 512 (0x200) and we get 0x10000. To see this partition we should jump there by pressing ctrl-g then type 0x10000:

```

root@chfiVBox: ~
00010000 EB 52 90 4E 54 46 53 20 20 20 20 00 02 08 00 00 .R.NTFS .....
00010010 00 00 00 00 00 F8 00 00 3F 00 FF 00 80 00 00 00 .....?.....
00010020 00 00 00 00 80 00 80 00 FF E7 1F 00 00 00 00 00 .....
00010030 55 54 01 00 00 00 00 00 7F FE 01 00 00 00 00 00 UT.....
00010040 F6 00 00 00 01 00 00 00 A0 C5 E1 6C F9 E1 6C 24 .....l..l$
00010050 00 00 00 00 FA 33 C0 8E D0 BC 00 7C FB 68 C0 07 .....3.....|.h..
00010060 1F 1E 68 66 00 CB 88 16 0E 00 66 81 3E 03 00 4E ..hf.....f.>..N
00010070 54 46 53 75 15 B4 41 BB AA 55 CD 13 72 0C 81 FB TFSu..A..U..r...
00010080 55 AA 75 06 F7 C1 01 00 75 03 E9 D2 00 1E 83 EC U..u....u.....
00010090 18 68 1A 00 B4 48 8A 16 0E 00 8B F4 16 1F CD 13 .h...H.....
000100A0 9F 83 C4 18 9E 58 1F 72 E1 3B 06 0B 00 75 DB A3 .....X.r.;...u..
000100B0 0F 00 C1 2E 0F 00 04 1E 5A 33 DB B9 00 20 2B C8 .....Z3...+.
000100C0 66 FF 06 11 00 03 16 0F 00 8E C2 FF 06 16 00 E8 f.....
000100D0 40 00 2B C8 77 EF B8 00 BB CD 1A 66 23 C0 75 2D @.+..w.....f#.u-
000100E0 66 81 FB 54 43 50 41 75 24 81 F9 02 01 72 1E 16 f..TCPAu$.r..
000100F0 68 07 BB 16 68 70 0E 16 68 09 00 66 53 66 53 66 h...hp..h..fSfSf
00010100 55 16 16 16 68 B8 01 66 61 0E 07 CD 1A E9 6A 01 U...h..fa....j.
00010110 90 90 66 60 1E 06 66 A1 11 00 66 03 06 1C 00 1E ..f`..f...f.....
00010120 66 68 00 00 00 00 66 50 06 53 68 01 00 68 10 00 fh....fP.Sh..h..
00010130 B4 42 8A 16 0E 00 16 1F 8B F4 CD 13 66 59 5B 5A .B.....fY[Z
00010140 66 59 66 59 1F 0F 82 16 00 66 FF 06 11 00 03 16 fYfY....f.....
00010150 0F 00 8E C2 FF 0E 16 00 75 BC 07 1F 66 61 C3 A0 .....u...fa..
00010160 F8 01 E8 08 00 A0 FB 01 E8 02 00 EB FE B4 01 8B .....
00010170 F0 AC 3C 00 74 09 B4 0E BB 07 00 CD 10 EB F2 C3 ..<.t.....
00010180 0D 0A 41 20 64 69 73 6B 20 72 65 61 64 20 65 72 ..A disk read er
00010190 72 6F 72 20 6F 63 63 75 72 72 65 64 00 0D 0A 42 ror occurred...B
000101A0 4F 4F 54 4D 47 52 20 69 73 20 6D 69 73 73 69 6E OOTMGR is missin
000101B0 67 00 0D 0A 42 4F 4F 54 4D 47 52 20 69 73 20 63 g...BOOTMGR is c
000101C0 6F 6D 70 72 65 73 73 65 64 00 0D 0A 50 72 65 73 ompressed...Pres
000101D0 73 20 43 74 72 6C 2B 41 6C 74 2B 44 65 6C 20 74 s Ctrl+Alt+Del t
000101E0 6F 20 72 65 73 74 61 72 74 0D 0A 00 00 00 00 00 o restart.....
000101F0 00 00 00 00 00 00 00 00 80 9D B2 CA 00 00 55 AA .....U.
--- sde --0x101F0/0x40000000-----

```

- **0x0B..0x0C** (marked with yellow) The sector size given in bytes (we already know it should be 512 = 0x200 bytes, but check it for the safety)
- **0x0D** (marked with blue): The size of cluster given in sectors. Now it is 0x08 so the size of the cluster is $0x200 * 0x08 = 0x1000 = 4096$ bytes.
- **0x30..0x37** (marked with red): the starting position of the \$MFT given in clusters and measured from the start of the volume. The value here is 5554010000000000 So the position will be $0x015455 * 0x1000 + 0x10000 = 0x15465000$. We can jump there by pressing ctrl-g then type 0x15465000

Here starts the \$MFT file, we recognize it from the “FILE” magic value:

```
root@chfIVBox: ~
15465000 46 49 4C 45 30 00 03 00 8D 10 20 00 00 00 00 00 FILE0.....
15465010 01 00 01 00 38 00 01 00 98 01 00 00 00 04 00 00 ....8.....
15465020 00 00 00 00 00 00 00 00 06 00 00 00 00 00 00 00 .....
15465030 02 00 00 00 00 00 00 00 10 00 00 00 60 00 00 00 .....
15465040 00 00 18 00 00 00 00 00 48 00 00 00 18 00 00 00 .....H.....
15465050 47 E6 4A E1 6E 19 CC 01 47 E6 4A E1 6E 19 CC 01 G.J.n...G.J.n...
15465060 47 E6 4A E1 6E 19 CC 01 47 E6 4A E1 6E 19 CC 01 G.J.n...G.J.n...
15465070 06 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
15465080 00 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 .....
15465090 00 00 00 00 00 00 00 00 30 00 00 00 68 00 00 00 .....0...h...
154650A0 00 00 18 00 00 00 03 00 4A 00 00 00 18 00 01 00 .....J.....
154650B0 05 00 00 00 00 00 05 00 47 E6 4A E1 6E 19 CC 01 .....G.J.n...
154650C0 47 E6 4A E1 6E 19 CC 01 47 E6 4A E1 6E 19 CC 01 G.J.n...G.J.n...
154650D0 47 E6 4A E1 6E 19 CC 01 00 40 00 00 00 00 00 00 G.J.n...@.....
154650E0 00 40 00 00 00 00 00 00 06 00 00 00 00 00 00 00 .@.....
154650F0 04 03 24 00 4D 00 46 00 54 00 00 00 00 00 00 00 ..$.M.F.T.....
15465100 80 00 00 00 48 00 00 00 01 00 40 00 00 00 01 00 ....H....@.....
15465110 00 00 00 00 00 00 00 00 0F 00 00 00 00 00 00 00 .....
15465120 40 00 00 00 00 00 00 00 00 00 01 00 00 00 00 00 @.....
15465130 00 00 01 00 00 00 00 00 00 00 01 00 00 00 00 00 .....
15465140 31 10 55 54 01 00 A4 90 B0 00 00 00 48 00 00 00 1.UT.....H...
15465150 01 00 40 00 00 00 05 00 00 00 00 00 00 00 00 00 ..@.....
15465160 00 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 .....@.....
15465170 00 10 00 00 00 00 00 00 08 00 00 00 00 00 00 00 .....
15465180 08 00 00 00 00 00 00 00 31 01 54 54 01 00 00 00 .....1.TT....
15465190 FF FF FF FF 00 00 00 00 00 00 01 00 00 00 00 00 .....
154651A0 00 00 01 00 00 00 00 00 31 10 55 54 01 00 A4 90 .....1.UT....
154651B0 B0 00 00 00 48 00 00 00 01 00 40 00 00 00 05 00 ....H....@.....
154651C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
154651D0 40 00 00 00 00 00 00 00 00 10 00 00 00 00 00 00 @.....
154651E0 08 00 00 00 00 00 00 00 08 00 00 00 00 00 00 00 .....
154651F0 31 01 54 54 01 00 00 00 FF FF FF FF 00 00 02 00 1.TT.....
--- sde --0x15465000/0x40000000-----
```

As we remember the first values are system values so we jump to the 35th entry. One entry is always 1024 (0x400) bytes long. So we should go to $0x15465000 + 35 * 0x400 = 0x1546DC00$.

We can jump there by pressing ctrl-g and type 0x1546DC00


```

root@chfIVBox: ~
1546DC00 46 49 4C 45 30 00 03 00 3D 45 20 00 00 00 00 00 FILE0...=E .....
1546DC10 01 00 01 00 38 00 01 00 68 01 00 00 00 04 00 00 ....8...h.....
1546DC20 00 00 00 00 00 00 00 00 05 00 00 00 23 00 00 00 .....#...
1546DC30 05 00 00 00 00 00 00 00 10 00 00 00 60 00 00 00 .....`...
1546DC40 00 00 00 00 00 00 00 00 48 00 00 00 18 00 00 00 .....H.....
1546DC50 00 D8 5E AC 3A 00 00 00 00 D8 5E AC 3A 00 00 00 ..^:.....^:....
1546DC60 00 D8 5E AC 3A 00 00 00 00 D8 5E AC 3A 00 00 00 ..^:.....^:....
1546DC70 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546DC80 00 00 00 00 05 01 00 00 00 00 00 00 00 00 00 00 .....
1546DC90 00 00 00 00 00 00 00 00 30 00 00 00 68 00 00 00 .....0...h...
1546DCA0 00 00 00 00 00 00 04 00 4C 00 00 00 18 00 01 00 .....L.....
1546DCB0 05 00 00 00 00 00 05 00 69 B9 6F BA 6F 19 CC 01 .....i.o.o...
1546DCC0 69 B9 6F BA 6F 19 CC 01 69 B9 6F BA 6F 19 CC 01 i.o.o...i.o.o...
1546DCD0 69 B9 6F BA 6F 19 CC 01 00 00 00 00 00 00 00 00 00 i.o.o.....
1546DCE0 00 00 00 00 00 00 00 00 20 00 00 00 00 00 00 00 .....
1546DCF0 05 03 63 00 2E 00 74 00 78 00 74 00 58 00 7E 00 ..c...t.x.t.X.~.
1546DD00 80 00 00 00 60 00 00 00 00 00 18 00 00 00 01 00 ....`.....
1546DD10 42 00 00 00 18 00 00 00 63 63 63 63 63 63 63 63 B.....cccccccc
1546DD20 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 ccccccccccccccccc
1546DD30 63 63 63 63 63 63 63 0D 0A 63 63 63 63 63 63 63 63 cccccccc..cccccccc
1546DD40 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 63 ccccccccccccccccc
1546DD50 63 63 63 63 63 63 63 63 0D 0A 00 00 00 00 00 00 cccccccc.....
1546DD60 FF FF FF FF 82 79 47 11 15 01 4E 00 65 00 77 00 ....yG...N.e.w.
1546DD70 20 00 54 00 65 00 78 00 74 00 20 00 44 00 6F 00 ..T.e.x.t..D.o.
1546DD80 63 00 75 00 6D 00 65 00 6E 00 74 00 2E 00 74 00 c.u.m.e.n.t...t.
1546DD90 78 00 74 00 00 00 00 00 80 00 00 00 18 00 00 00 x.t.....
1546DDA0 00 00 18 00 00 00 01 00 00 00 00 00 18 00 00 00 .....
1546ddb0 FF FF FF FF 82 79 47 11 00 00 00 00 00 00 00 00 ....yG.....
1546DDC0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546DDD0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546DDE0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546DDF0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 05 00 .....
--- sde --0x1546DC00/0x40000000-----

```

- **0x14..0x15** (marked with dark blue) means the offset to the first attribute. Now it is 3800 what means 0x38 because it is in little endian notation
- **0x38..0x3B** (marked with light blue) is the type of the attribute 10000000 means 0x10 what is the standard information block
- **0x3C..0x3F** (marked with dark blue) is the length of this attribute. Now it is 60000000 what means 0x60 bytes, so the next attribute going to start at the position 0x38 0x60 = 0x98
- **0x40** (marked with green) is the resident flag. It is 0 obviously, because the standard information block is always resident.
- **0x50..0x6F** (marked with grey) contains the time stamps.
 - First is the modification time. Now it is: 00D85EAC3A000000 we can read the value by typing `w32tm /ntte 0x00000003AAC5ED800`
 - The second is the last modification time, now it is 00D85EAC3A000000 again one can make it readable by typing `w32tm /ntte 0x00000003AAC5ED800`.
 - The next timestamp is the last modification time of the MFT entry, the value of it now 00D85EAC3A000000 again to see the value run the `w32tm /ntte 0x00000003AAC5ED800` command.
 - The last one is the last access time of the file, now it is 00D85EAC3A000000 again if we want to read it run the `w32tm /ntte 0x00000003AAC5ED800` command.

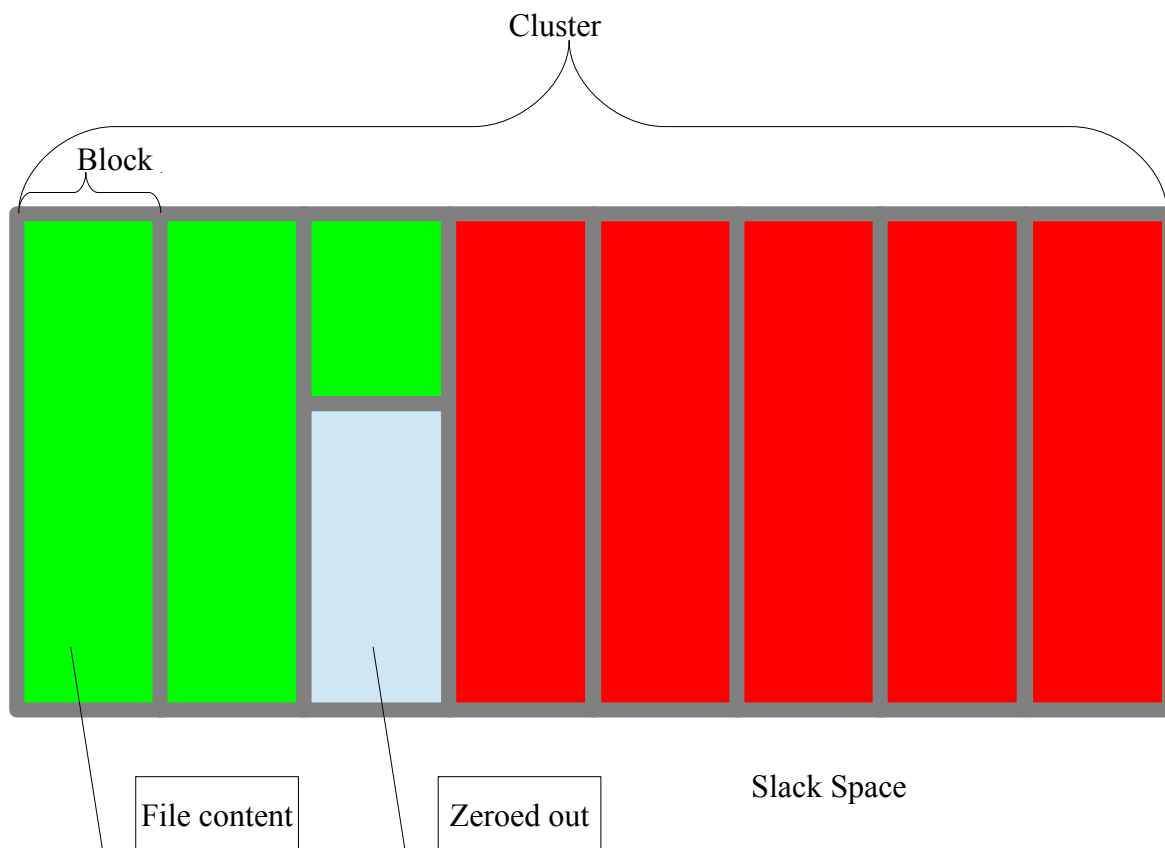
- *As we can see this values now are not valid.*
- **0x98..0x9B** (marked with light blue) : contains the second attribute. The value of it is 30000000 means 0x30 file_name attribute.
- **0x9C..0x9F** (marked with dark blue) is the length of this attribute. The value of it now is 68000000, but it is given in little endian format, what means 0x68. So the third attribute starts at $0x98 \ 0x68 = 0x0100$
- **0xA0** (marked with green) is the resident flag. Again this attribute is always resident so not surprising we get a value 0 here.
- **0xB8..0xD8** : (marked with grey) contains the time stamps.
 - First is the modification time. Now it is: 69B96FBA6F19CC01 we can read the value by typing `w32tm /ntte 0x01CC196FBA6FB969`
 - The second is the last modification time, now it is 69B96FBA6F19CC01 again one can make it readable by typing `w32tm /ntte 0x01CC196FBA6FB969`.
 - The next timestamp is the last modification time of the MFT entry, the value of it now 69B96FBA6F19CC01 again to see the value run the `w32tm /ntte 0x01CC196FBA6FB969` command.
 - The last one is the last access time of the file, now it is 69B96FBA6F19CC01 again if we want to read it run the `w32tm /ntte 0x01CC196FBA6FB969` command.
 - *It contains real values so we should continue any investigation with these ones, or based on this value try to correct the previous time stamp, because only the first three bytes are zeroed out by the timestamp.*
- **0x0100..0x0103** (marked with light blue) : here starts the third attribute. Now it has a value 80000000 means 0x80 DATA.
- **0x0104..0x0107** (marked with dark blue) : the length of this attribute now contains 60000000 what means 0x60 bytes. So the next attribute starts at $0x0100 + 0x60 = 0x0160$
- **0x0108** (marked with green) : is the resident flag, what is now 0, so we have a resident file.
- **0x0110..0x0113** (marked with black) : shows the length of the data (file content) it has a value of 42000000 what means 0x42
- **0x0114..0x0117** (marked with white) : shows the offset to the start of the file content. It has a value 18000000 what means 0x18 so the content will start at $0x0100 + 0x18 = 0x0118$. The end of it will be at position $0x0118 + 0x42 = 0x015A$
- **0x0118.. 0x15A** (marked with red) is the content of the file.

Slack space (Metasploit slacker)

As we have already seen the operating system writes the data in the unit of clusters while the disk works in unit of block. In most of the time in windows environment the cluster size is 4096 bytes while the block size is 512 bytes. If you write some data to a disk most probably it will not be the integer multiplication of 4096 so the last cluster will not be utilized fully.

What does it mean for us. The last cluster of any file mean contain unused space, what we can use to hide data (this is done by the metasploit slacker application). Or here you may find data from some already overwritten file. Some very old windows edition filled it with random data from memory.

Last cluster of a file:



To test it attach the evidenceSlacker.vhd to your examination machine (for me it is attached as /dev/sdf) then open it with a hexeditor. As usually the first sector contains the MBR we can start to search for the NTFS partition on the already well known way:

```
root@chflVBox: ~
00000000  33 C0 8E D0 BC 00 7C 8E C0 8E D8 BE 00 7C BF 00 3.....|.....|..
00000010  06 B9 00 02 FC F3 A4 50 68 1C 06 CB FB B9 04 00 .....Ph.....
00000020  BD BE 07 80 7E 00 00 7C 0B 0F 85 10 01 83 C5 10 ....~..|.....
00000030  E2 F1 CD 18 88 56 00 55 C6 46 11 05 C6 46 10 00 .....V.U.F...F..
00000040  B4 41 BB AA 55 CD 13 5D 72 0F 81 FB 55 AA 75 09 .A..U..]r...U.u.
00000050  F7 C1 01 00 74 03 FE 46 10 66 60 80 7E 10 00 74 ....t..F.f`.~..t
00000060  26 66 68 00 00 00 00 66 FF 76 08 68 00 00 68 00 &fh....f.v.h..h.
00000070  7C 68 01 00 68 10 00 B4 42 8A 56 00 8B F4 CD 13 |h..h...B.V.....
00000080  9F 83 C4 10 9E EB 14 B8 01 02 BB 00 7C 8A 56 00 .....|.V.
00000090  8A 76 01 8A 4E 02 8A 6E 03 CD 13 66 61 73 1E FE .v..N..n...fas..
000000A0  4E 11 0F 85 0C 00 80 7E 00 80 0F 84 8A 00 B2 80 N.....~.....
000000B0  EB 82 55 32 E4 8A 56 00 CD 13 5D EB 9C 81 3E FE ..U2..V...]>..
000000C0  7D 55 AA 75 6E FF 76 00 E8 8A 00 0F 85 15 00 B0 }U.un.v.....
000000D0  D1 E6 64 E8 7F 00 B0 DF E6 60 E8 78 00 B0 FF E6 ..d.....`.x....
000000E0  64 E8 71 00 B8 00 BB CD 1A 66 23 C0 75 3B 66 81 d.q.....f#.u;f.
000000F0  FB 54 43 50 41 75 32 81 F9 02 01 72 2C 66 68 07 .TCPAu2....r,fh.
00000100  BB 00 00 66 68 00 02 00 00 66 68 08 00 00 00 66 ...fh....fh....f
00000110  53 66 53 66 55 66 68 00 00 00 00 66 68 00 7C 00 SfsfUfh....fh.|.
00000120  00 66 61 68 00 00 07 CD 1A 5A 32 F6 EA 00 7C 00 .fah.....Z2...|.
00000130  00 CD 18 A0 B7 07 EB 08 A0 B6 07 EB 03 A0 B5 07 .....
00000140  32 E4 05 00 07 8B F0 AC 3C 00 74 FC BB 07 00 B4 2.....<.t.....
00000150  0E CD 10 EB F2 2B C9 E4 64 EB 00 24 02 E0 F8 24 .....+.d..$...$
00000160  02 C3 49 6E 76 61 6C 69 64 20 70 61 72 74 69 74 ..Invalid partit
00000170  69 6F 6E 20 74 61 62 6C 65 00 45 72 72 6F 72 20 ion table.Error
00000180  6C 6F 61 64 69 6E 67 20 6F 70 65 72 61 74 69 6E loading operatin
00000190  67 20 73 79 73 74 65 6D 00 4D 69 73 73 69 6E 67 g system.Missing
000001A0  20 6F 70 65 72 61 74 69 6E 67 20 73 79 73 74 65 operating syste
000001B0  6D 00 00 00 00 62 7A 99 E2 68 2E F9 00 00 00 02 m....bz..h.....
000001C0  03 00 07 28 A8 06 80 00 00 00 00 E8 1F 00 00 00 ...{(.....
000001D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000001E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000001F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA .....U.
--- sdf --0x0/0x40000000-----
```

We can find the first partition entry. It starts at position 0x01BE marked with blue.

The start position of the first partition is 80000000. Again it is little endian so 0x80. It is given in blocks so we multiply it by 512 (0x200) and we get 0x10000. To see this partition we should jump there by pressing ctrl-g then type 0x10000:

```

root@chflVBox: ~
00010000 EB 52 90 4E 54 46 53 20 20 20 20 00 02 08 00 00 .R.NTFS .....
00010010 00 00 00 00 00 F8 00 00 3F 00 40 00 80 00 00 00 .....?.@.....
00010020 00 00 00 00 80 00 80 00 FF E7 1F 00 00 00 00 00 .....
00010030 55 54 01 00 00 00 00 00 7F FE 01 00 00 00 00 00 UT.....
00010040 F6 00 00 00 01 00 00 00 B7 37 68 64 43 68 64 46 .....7hdChdF
00010050 00 00 00 00 FA 33 C0 8E D0 BC 00 7C FB 68 C0 07 .....3.....|.h..
00010060 1F 1E 68 66 00 CB 88 16 0E 00 66 81 3E 03 00 4E ..hf.....f.>..N
00010070 54 46 53 75 15 B4 41 BB AA 55 CD 13 72 0C 81 FB TFSu..A..U..r...
00010080 55 AA 75 06 F7 C1 01 00 75 03 E9 D2 00 1E 83 EC U.u.....u.....
00010090 18 68 1A 00 B4 48 8A 16 0E 00 8B F4 16 1F CD 13 .h...H.....
000100A0 9F 83 C4 18 9E 58 1F 72 E1 3B 06 0B 00 75 DB A3 .....X.r.;...u..
000100B0 0F 00 C1 2E 0F 00 04 1E 5A 33 DB B9 00 20 2B C8 .....Z3...+.
000100C0 66 FF 06 11 00 03 16 0F 00 8E C2 FF 06 16 00 E8 f.....
000100D0 40 00 2B C8 77 EF B8 00 BB CD 1A 66 23 C0 75 2D @.+..w.....f#.u-
000100E0 66 81 FB 54 43 50 41 75 24 81 F9 02 01 72 1E 16 f..TCPAu$.r..
000100F0 68 07 BB 16 68 70 0E 16 68 09 00 66 53 66 53 66 h...hp..h..fsfsf
00010100 55 16 16 16 68 B8 01 66 61 0E 07 CD 1A E9 6A 01 U...h..fa....j.
00010110 90 90 66 60 1E 06 66 A1 11 00 66 03 06 1C 00 1E ..f`..f...f.....
00010120 66 68 00 00 00 00 66 50 06 53 68 01 00 68 10 00 fh....fP.Sh..h..
00010130 B4 42 8A 16 0E 00 16 1F 8B F4 CD 13 66 59 5B 5A .B.....fy[Z
00010140 66 59 66 59 1F 0F 82 16 00 66 FF 06 11 00 03 16 fyfy....f.....
00010150 0F 00 8E C2 FF 0E 16 00 75 BC 07 1F 66 61 C3 A0 .....u...fa..
00010160 F8 01 E8 08 00 A0 FB 01 E8 02 00 EB FE B4 01 8B .....
00010170 F0 AC 3C 00 74 09 B4 0E BB 07 00 CD 10 EB F2 C3 ..<.t.....
00010180 0D 0A 41 20 64 69 73 6B 20 72 65 61 64 20 65 72 ..A disk read er
00010190 72 6F 72 20 6F 63 63 75 72 72 65 64 00 0D 0A 42 ror occurred...B
000101A0 4F 4F 54 4D 47 52 20 69 73 20 6D 69 73 73 69 6E 00TMGR is missin
000101B0 67 00 0D 0A 42 4F 4F 54 4D 47 52 20 69 73 20 63 g...BOOTMGR is c
000101C0 6F 6D 70 72 65 73 73 65 64 00 0D 0A 50 72 65 73 ompressed...Pres
000101D0 73 20 43 74 72 6C 2B 41 6C 74 2B 44 65 6C 20 74 s Ctrl+Alt+Del t
000101E0 6F 20 72 65 73 74 61 72 74 0D 0A 00 00 00 00 00 restart.....
000101F0 00 00 00 00 00 00 00 00 80 9D B2 CA 00 00 55 AA .....U.
--- sdf --0x10000/0x40000000-----

```

- **0x0B..0x0C** (marked with yellow) The sector size given in bytes (we already know it should be 512 = 0x200 bytes, but check it for the safety)
- **0x0D** (marked with blue): The size of cluster given in sectors. Now it is 0x08 so the size of the cluster is 0x200 * 0x08 = 0x1000 = 4096 bytes.
- **0x30..0x37** (marked with red): the starting position of the \$MFT given in clusters and measured from the start of the volume. The value here is 5554010000000000 So the position will be 0x015455 * 0x1000 + 0x10000 = 0x15465000. We can jump there by pressing ctrl-g then type 0x15465000

Here starts the \$MFT file, we recognize it from the “FILE” magic value:


```
root@chflVBox: ~
15465000  46 49 4C 45 30 00 03 00 8D 10 20 00 00 00 00 00 FILE0.....
15465010  01 00 01 00 38 00 01 00 98 01 00 00 00 04 00 00 ....8.....
15465020  00 00 00 00 00 00 00 00 06 00 00 00 00 00 00 00 .....
15465030  02 00 00 00 00 00 00 00 10 00 00 00 60 00 00 00 .....`...
15465040  00 00 18 00 00 00 00 00 48 00 00 00 18 00 00 00 .....H.....
15465050  76 25 C4 66 39 1A CC 01 76 25 C4 66 39 1A CC 01 v%.f9...v%.f9...
15465060  76 25 C4 66 39 1A CC 01 76 25 C4 66 39 1A CC 01 v%.f9...v%.f9...
15465070  06 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
15465080  00 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 .....
15465090  00 00 00 00 00 00 00 00 30 00 00 00 68 00 00 00 .....0...h...
154650A0  00 00 18 00 00 00 03 00 4A 00 00 00 18 00 01 00 .....J.....
154650B0  05 00 00 00 00 00 05 00 76 25 C4 66 39 1A CC 01 .....v%.f9...
154650C0  76 25 C4 66 39 1A CC 01 76 25 C4 66 39 1A CC 01 v%.f9...v%.f9...
154650D0  76 25 C4 66 39 1A CC 01 00 40 00 00 00 00 00 00 v%.f9....@.....
154650E0  00 40 00 00 00 00 00 00 06 00 00 00 00 00 00 00 .@.....
154650F0  04 03 24 00 4D 00 46 00 54 00 00 00 00 00 00 00 ..$.M.F.T.....
15465100  80 00 00 00 48 00 00 00 01 00 40 00 00 00 01 00 ....H....@.....
15465110  00 00 00 00 00 00 00 00 0F 00 00 00 00 00 00 00 .....
15465120  40 00 00 00 00 00 00 00 00 00 01 00 00 00 00 00 @.....
15465130  00 00 01 00 00 00 00 00 00 00 01 00 00 00 00 00 .....
15465140  31 10 55 54 01 00 DC 92 B0 00 00 00 48 00 00 00 1.UT.....H...
15465150  01 00 40 00 00 00 05 00 00 00 00 00 00 00 00 00 ..@.....
15465160  00 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 .....@.....
15465170  00 10 00 00 00 00 00 00 08 00 00 00 00 00 00 00 .....
15465180  08 00 00 00 00 00 00 00 31 01 54 54 01 00 00 00 .....1.TT....
15465190  FF FF FF FF 00 00 00 00 00 00 01 00 00 00 00 00 .....
154651A0  00 00 01 00 00 00 00 00 31 10 55 54 01 00 DC 92 .....1.UT....
154651B0  B0 00 00 00 48 00 00 00 01 00 40 00 00 00 05 00 ....H....@.....
154651C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
154651D0  40 00 00 00 00 00 00 00 00 10 00 00 00 00 00 00 @.....
154651E0  08 00 00 00 00 00 00 00 08 00 00 00 00 00 00 00 .....
154651F0  31 01 54 54 01 00 00 00 FF FF FF FF 00 00 02 00 1.TT.....
--- sdf --0x154651F0/0x40000000-----
```

As we remember the first values are system values so we jump to the 35th entry. One entry is always 1024 (0x400) bytes long. So we should go to $0x15465000 + 35 * 0x400 = 0x1546DC00$.

We can jump there by pressing ctrl-g and type 0x1546DC00

```

root@chflVBox: ~
1546DC00  46 49 4C 45 30 00 03 00 B6 2F 20 00 00 00 00 00 FILE0.... / .....
1546DC10  01 00 01 00 38 00 01 00 50 01 00 00 00 04 00 00 ....8...P.....
1546DC20  00 00 00 00 00 00 00 00 03 00 00 00 23 00 00 00 .....#...
1546DC30  03 00 00 00 00 00 00 00 10 00 00 00 60 00 00 00 .....`...
1546DC40  00 00 00 00 00 00 00 00 48 00 00 00 18 00 00 00 .....H.....
1546DC50  A1 F1 8E E7 39 1A CC 01 81 86 6B 27 8A 19 CC 01 ....9....k'....
1546DC60  27 45 97 E7 39 1A CC 01 A1 F1 8E E7 39 1A CC 01 'E..9.....9...
1546DC70  20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546DC80  00 00 00 00 05 01 00 00 00 00 00 00 00 00 00 00 .....
1546DC90  00 00 00 00 00 00 00 00 30 00 00 00 68 00 00 00 .....0...h...
1546DCA0  00 00 00 00 00 00 02 00 4C 00 00 00 18 00 01 00 .....L.....
1546DCB0  05 00 00 00 00 00 05 00 A1 F1 8E E7 39 1A CC 01 .....9...
1546DCC0  A1 F1 8E E7 39 1A CC 01 A1 F1 8E E7 39 1A CC 01 ....9.....9...
1546DCD0  A1 F1 8E E7 39 1A CC 01 00 10 00 00 00 00 00 00 ....9.....
1546DCE0  00 00 00 00 00 00 00 00 20 00 00 00 00 00 00 00 .....
1546DCF0  05 03 71 00 2E 00 74 00 78 00 74 00 00 00 00 00 ..q...t.x.t....
1546DD00  80 00 00 00 48 00 00 00 01 00 00 00 00 00 01 00 ....H.....
1546DD10  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546DD20  40 00 00 00 00 00 00 00 00 10 00 00 00 00 00 00 @.....
1546DD30  C8 04 00 00 00 00 00 00 C8 04 00 00 00 00 00 00 .....
1546DD40  31 01 54 E9 01 00 00 00 FF FF FF FF 82 79 47 11 1.T.....yG.
1546DD50  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546DD60  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546DD70  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546DD80  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546DD90  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546DDA0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546ddb0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546DDC0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546DDD0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546DDE0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546DDF0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 .....
--- sdf                --0x1546DC00/0x40000000-----

```

Here you find the first file called a q.txt, and if you start to scroll down you will find many other files:

- q.txt 1546DC00
- d.txt 1546E000
- e.txt 1546E400
- f.txt 1546E800
- g.txt 1546EC00
- h.txt 1546F000
- i.txt 1546F400
- j.txt 1546F800
- k.txt 1546FC00
- l.txt 15470000
- m.txt 15470400
- n.txt 15470800
- o.txt 15470C00
- p.txt 15471000
- r.txt 15471400

First I show the content the r.txt. We search for its cluster chain on the already well known way:

```

root@chfIVBox: ~
15471400 46 49 4C 45 30 00 03 00 EE 42 20 00 00 00 00 00 FILE0....B .....
15471410 01 00 01 00 38 00 01 00 50 01 00 00 00 04 00 00 ....8...P.....
15471420 00 00 00 00 00 00 00 00 03 00 00 00 31 00 00 00 .....1...
15471430 03 00 00 00 00 00 00 00 10 00 00 00 60 00 00 00 .....`...
15471440 00 00 00 00 00 00 00 00 48 00 00 00 18 00 00 00 .....H.....
15471450 85 B9 7B F9 39 1A CC 01 AE D7 02 B4 3A 1A CC 01 ..{.9.....:...
15471460 AE D7 02 B4 3A 1A CC 01 85 B9 7B F9 39 1A CC 01 ....:.....{.9...
15471470 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
15471480 00 00 00 00 05 01 00 00 00 00 00 00 00 00 00 00 .....
15471490 00 00 00 00 00 00 00 00 30 00 00 00 68 00 00 00 .....0...h...
154714A0 00 00 00 00 00 00 02 00 4C 00 00 00 18 00 01 00 .....L.....
154714B0 05 00 00 00 00 00 05 00 85 B9 7B F9 39 1A CC 01 .....{.9...
154714C0 85 B9 7B F9 39 1A CC 01 85 B9 7B F9 39 1A CC 01 ..{.9.....{.9...
154714D0 85 B9 7B F9 39 1A CC 01 00 10 00 00 00 00 00 00 ..{.9.....
154714E0 00 00 00 00 00 00 00 00 20 00 00 00 00 00 00 00 .....
154714F0 05 03 72 00 2E 00 74 00 78 00 74 00 00 00 00 00 ..r...t.x.t....
15471500 80 00 00 00 48 00 00 00 01 00 00 00 00 00 01 00 ....H.....
15471510 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
15471520 40 00 00 00 00 00 00 00 00 10 00 00 00 00 00 00 @.....
15471530 C8 04 00 00 00 00 00 00 C8 04 00 00 00 00 00 00 .....
15471540 31 01 80 FE 01 00 00 00 FF FF FF FF 82 79 47 11 1.....yG.
15471550 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
15471560 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
15471570 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
15471580 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
15471590 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
154715A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
154715B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
154715C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
154715D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
154715E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
154715F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 .....
--- sdf                --0x15471400/0x40000000-----

```

- **0x14..0x15** (marked with dark blue) means the offset to the first attribute. Now it is 3800 what means 0x38 because it is in little endian notation
- **0x38..0x3B** (marked with light blue) is the type of the attribute 10000000 means 0x10 what is the standard information block
- **0x3C..0x3F** (marked with dark blue) is the length of this attribute. Now it is 60000000 what means 0x60 bytes, so the next attribute is going to start at the position 0x38 0x60 = 0x98
- **0x40** (marked with green) is the resident flag. It is 0 obviously, because the standard information block is always resident.
- **0x50..0x6F** (marked with grey) contains the time stamps.
 - First is the modification time. Now it is: 85B97BF9391ACC01 we can read the value by typing `w32tm /ntte 0x01CC1A39F97BB985`
 - The second is the last modification time, now it is AED702B43A1ACC01 again one can make it readable by typing `w32tm /ntte 0x01CC1A3AB402D7AE`.
 - The next timestamp is the last modification time of the MFT entry, the value of it now AED702B43A1ACC01 again to see the value run the `w32tm /ntte 0x01CC1A3AB402D7AE` command.

- The last one is the last access time of the file, now it is 85B97BF9391ACC01 again if we want to read it run the `w32tm /ntte 0x01CC1A39F97BB985` command.
- *Again I want to mention some evidence eliminator (like the actual version of measploit timestamp) deletes or modifies only these values.*
- **0x98..0x9B** (marked with light blue) : contains the second attribute. The value of it is 30000000 means 0x30 file_name attribute.
- **0x9C..0x9F** (marked with dark blue) is the length of this attribute. The value of it now is 68000000, but it is given in little endian format, what means 0x68. So the third attribute starts at 0x98 0x68 = 0x0100
- **0xA0** (marked with green) is the resident flag. Again this attribute is always resident so not surprising we get a value 0 here.
- **0xB8..0xD8** : (marked with grey) contains the time stamps.
 - First is the modification time. Now it is: 85B97BF9391ACC01 we can read the value by typing `w32tm /ntte 0x01CC1A39F97BB985`
 - The second is the last modification time, now it is 85B97BF9391ACC01 again one can make it readable by typing `w32tm /ntte 0x01CC1A39F97BB985`.
 - The next timestamp is the last modification time of the MFT entry, the value of it now 85B97BF9391ACC01 again to see the value run the `w32tm /ntte 0x01CC1A39F97BB985` command.
 - The last one is the last access time of the file, now it is 85B97BF9391ACC01 again if we want to read it run the `w32tm /ntte 0x01CC1A39F97BB985` command.
- **0x0100..0x0103** (marked with light blue) : here starts the third attribute. Now it has a value 80000000 means 0x80 DATA.
- **0x0104..0x0107** (marked with dark blue) : the length of this attribute now contains 48000000 what means 0x48 bytes. So the next attribute starts at 0x0100 0x48 = 0x0148
- **0x0108** (marked with green) : is the resident flag, what is now 1, because we have a non resident file.
- **0x0120..0x0123** (marked with dark blue) : is the offset where the cluster chains definition going to start. Now it has a value 40000000 what means 0x40. So the description of the cluster chain starts at 0x0140.
- **0x0140** (marked with black) : here starts the description of cluster chain. Now it has a value 0x31 what should be interpreted on the following way: the first number 3 (0x03) means the start cluster is stored on 3 bytes. The second number 1 (0x01) means the number of the clusters continuously occupied by the file. First the number of clusters is stored so we should read the next one byte
- **0x0141** (marked with yellow) : now it is 0x01 so the first cluster chain occupies 0x01 continuous clusters. If we calculate $0x01 * 0x1000 = 0x1000 = 4096$ bytes. The start cluster is stored on three bytes as we remember so we should read the next three bytes:
- **0x0142..0x0144** (marked with red) : is the start cluster of the file measured from the beginning of the volume. Now it has the value 80FE01 means 0x01FE80. So the file will begin at the position $0x01FE80 * 0x1000 + 0x10000 = 0x1FE90000$.
- **0x130..0x138** (marked with purple) : is the real size of the file. It has a value C804000000000000 it is in little endian so the value is 0x04C8 so the filesize is 1224 bytes. We can see that it is smaller than we calculated from the number of clusters so the last cluster will not be fully utilized.

Now let us jump to the position 0x1FE90000 by pressing ctrl-g then type 0x1FE90000, and really there we will find the content of the file a lot of letter "r" and sometimes a carriage return line feed:

```

root@chfIVBox: ~
1FE90000 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 rrrrrrrrrrrrrrrrrr
1FE90010 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 rrrrrrrrrrrrrrrrrr
1FE90020 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 rrrrrrrrrrrrrrrrrr
1FE90030 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 rrrrrrrrrrrrrrrrrr
1FE90040 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 rrrrrrrrrrrrrrrrrr
1FE90050 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 rrrrrrrrrrrrrrrrrr
1FE90060 72 72 72 72 0D 0A 72 72 72 72 72 72 72 72 72 rrrrr..rrrrrrrrrrrr
1FE90070 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 rrrrrrrrrrrrrrrrrr
1FE90080 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 rrrrrrrrrrrrrrrrrr
1FE90090 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 rrrrrrrrrrrrrrrrrr
1FE900A0 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 rrrrrrrrrrrrrrrrrr
1FE900B0 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 rrrrrrrrrrrrrrrrrr
1FE900C0 72 72 72 72 72 72 72 72 72 72 72 0D 0A 72 72 72 rrrrrrrrrrr..rrrr
1FE900D0 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 rrrrrrrrrrrrrrrrrr
1FE900E0 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 rrrrrrrrrrrrrrrrrr
1FE900F0 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 rrrrrrrrrrrrrrrrrr
1FE90100 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 rrrrrrrrrrrrrrrrrr
1FE90110 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 rrrrrrrrrrrrrrrrrr
1FE90120 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 rrrrrrrrrrrrrrrrrr
1FE90130 0D 0A 72 72 72 72 72 72 72 72 72 72 72 72 72 ..rrrrrrrrrrrrrrrr
1FE90140 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 rrrrrrrrrrrrrrrrrr
1FE90150 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 rrrrrrrrrrrrrrrrrr
1FE90160 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 rrrrrrrrrrrrrrrrrr
1FE90170 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 rrrrrrrrrrrrrrrrrr
1FE90180 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 rrrrrrrrrrrrrrrrrr
1FE90190 72 72 72 72 72 72 0D 0A 72 72 72 72 72 72 72 rrrrrrr..rrrrrrrrrr
1FE901A0 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 rrrrrrrrrrrrrrrrrr
1FE901B0 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 rrrrrrrrrrrrrrrrrr
1FE901C0 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 rrrrrrrrrrrrrrrrrr
1FE901D0 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 rrrrrrrrrrrrrrrrrr
1FE901E0 72 72 72 72 72 72 72 72 72 72 72 72 72 72 72 rrrrrrrrrrrrrrrrrr
1FE901F0 72 72 72 72 72 72 72 72 72 72 72 72 0D 0A 72 72 rrrrrrrrrrrrr..rr
--- sdf --0x1FE90000/0x40000000-----

```

```

root@chfIVBox: ~
1FE90600  11 63 3A 5C 66 69 6C 65 6F 6B 5C 6E 63 2E 7A 69 .c:\fileok\nc.zi
1FE90610  70 00 55 77 00 00 00 0C 00 00 00 0A 65 3A 5C 5C p.Uw.....e:\\
1FE90620  64 2E 74 78 74 00 C8 04 00 00 05 0A 65 3A 5C 5C d.txt.....e:\\
1FE90630  65 2E 74 78 74 00 C8 04 00 00 05 0A 65 3A 5C 5C e.txt.....e:\\
1FE90640  66 2E 74 78 74 00 C8 04 00 00 05 0A 65 3A 5C 5C f.txt.....e:\\
1FE90650  67 2E 74 78 74 00 C8 04 00 00 05 0A 65 3A 5C 5C g.txt.....e:\\
1FE90660  68 2E 74 78 74 00 C8 04 00 00 05 0A 65 3A 5C 5C h.txt.....e:\\
1FE90670  69 2E 74 78 74 00 C8 04 00 00 05 0A 65 3A 5C 5C i.txt.....e:\\
1FE90680  6A 2E 74 78 74 00 C8 04 00 00 05 0A 65 3A 5C 5C j.txt.....e:\\
1FE90690  6B 2E 74 78 74 00 C8 04 00 00 05 0A 65 3A 5C 5C k.txt.....e:\\
1FE906A0  6C 2E 74 78 74 00 C8 04 00 00 05 0A 65 3A 5C 5C l.txt.....e:\\
1FE906B0  6D 2E 74 78 74 00 C8 04 00 00 05 0A 65 3A 5C 5C m.txt.....e:\\
1FE906C0  6E 2E 74 78 74 00 C8 04 00 00 05 0A 65 3A 5C 5C n.txt.....e:\\
1FE906D0  6F 2E 74 78 74 00 C8 04 00 00 05 00 00 00 00 00 o.txt.....
1FE906E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FE906F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FE90700  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FE90710  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FE90720  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FE90730  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FE90740  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FE90750  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FE90760  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FE90770  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FE90780  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FE90790  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FE907A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FE907B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FE907C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FE907D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FE907E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FE907F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
--- sdf                --0x1FE90790/0x40000000-----

```

here we can read that, the nc.zip will be stored in the slack space of d.txt, e.txt... o.txt

Let us try to reassemble the hidden file and read the content of it:

The first file is the d.txt, the MFT entry of it starts at the position 0x1546E000. Lets jump there by pressing ctrl-g and type this value

Now I do not describe the entry in detail, because that has been done many time, just mark the important values, and the end result:

```

root@chfivBox: ~
1546E000  46 49 4C 45 30 00 03 00 BA 39 20 00 00 00 00 00 FILE0....9 .....
1546E010  01 00 01 00 38 00 01 00 50 01 00 00 00 04 00 00 ....8...P.....
1546E020  00 00 00 00 00 00 00 00 03 00 00 00 24 00 00 00 .....$.
1546E030  03 00 00 00 00 00 00 00 10 00 00 00 60 00 00 00 .....`...
1546E040  00 00 00 00 00 00 00 00 48 00 00 00 18 00 00 00 .....H.....
1546E050  EA 6E 9B E7 39 1A CC 01 C0 F9 F5 B3 3A 1A CC 01 .n..9.....:
1546E060  C0 F9 F5 B3 3A 1A CC 01 EA 6E 9B E7 39 1A CC 01 ....:....n..9...
1546E070  20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546E080  00 00 00 00 05 01 00 00 00 00 00 00 00 00 00 00 .....
1546E090  00 00 00 00 00 00 00 00 30 00 00 00 68 00 00 00 .....0...h...
1546E0A0  00 00 00 00 00 00 02 00 4C 00 00 00 18 00 01 00 .....L.....
1546E0B0  05 00 00 00 00 00 05 00 EA 6E 9B E7 39 1A CC 01 .....n..9...
1546E0C0  EA 6E 9B E7 39 1A CC 01 EA 6E 9B E7 39 1A CC 01 .n..9....n..9...
1546E0D0  EA 6E 9B E7 39 1A CC 01 00 10 00 00 00 00 00 00 .n..9.....
1546E0E0  00 00 00 00 00 00 00 00 20 00 00 00 00 00 00 00 .....
1546E0F0  05 03 64 00 2E 00 74 00 78 00 74 00 00 00 00 00 ..d...t.x.t....
1546E100  80 00 00 00 48 00 00 00 01 00 00 00 00 00 01 00 ....H.....
1546E110  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546E120  40 00 00 00 00 00 00 00 00 10 00 00 00 00 00 00 @.....
1546E130  C8 04 00 00 00 00 00 00 C8 04 00 00 00 00 00 00 .....
1546E140  31 01 55 E9 01 00 00 00 FF FF FF FF 82 79 47 11 1.U.....yG.
1546E150  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546E160  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546E170  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546E180  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546E190  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546E1A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546E1B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546E1C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546E1D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546E1E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1546E1F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 .....
--- sdf --0x1546E000/0x40000000-----

```

The start cluster will be at the position 55E901 what means $0x01E955 * 0x1000 + 0x10000 = 0x1E965000$

The file has length $0x04C8 = 1224$ bytes again, what means 3 blocks used by the file so the slack space will start from $0x1E965000 + 3 * 0x200 = 0x1E965600$

The length of data in slack space is $5 * 512 = 2560$ bytes

$0x1E965600 = 513168896$

we can extract this data by a simple dd command:

```
dd if=/dev/sdf bs=1 count=2560 skip=513168896 > /out.xxx
```

Similarly the remaining parts of the file can be extracted by the following commands:

```

dd if=/dev/sdf bs=1 count=2560 skip=513172992 >> /out.xxx
dd if=/dev/sdf bs=1 count=2560 skip=513177088 >> /out.xxx
dd if=/dev/sdf bs=1 count=2560 skip=513181184 >> /out.xxx
dd if=/dev/sdf bs=1 count=2560 skip=513185280 >> /out.xxx
dd if=/dev/sdf bs=1 count=2560 skip=513189376 >> /out.xxx

```



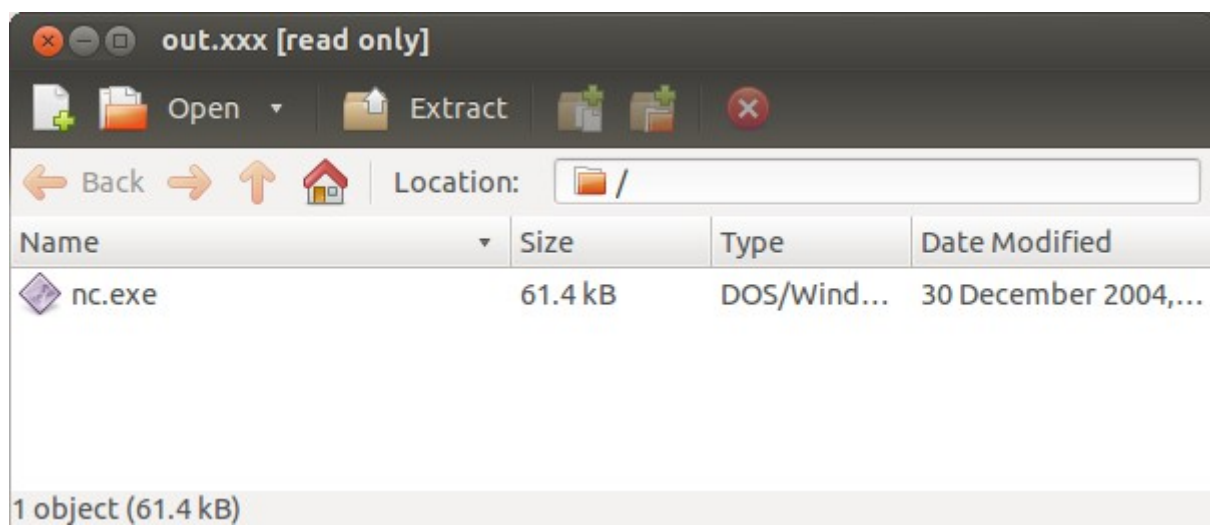
```
dd if=/dev/sdf bs=1 count=2560 skip=513193472 >> /out.xxx
dd if=/dev/sdf bs=1 count=2560 skip=513197568 >> /out.xxx
dd if=/dev/sdf bs=1 count=2560 skip=513201664 >> /out.xxx
dd if=/dev/sdf bs=1 count=2560 skip=513205760 >> /out.xxx
dd if=/dev/sdf bs=1 count=2560 skip=513209856 >> /out.xxx
dd if=/dev/sdf bs=1 count=2560 skip=513213952 >> /out.xxx
```

```
root@chfiVBox: ~
root@chfiVBox:~# dd if=/dev/sdf bs=1 count=2560 skip=513185280 >> /out.xxx
2560+0 records in
2560+0 records out
2560 bytes (2.6 kB) copied, 0.0454549 s, 56.3 kB/s
root@chfiVBox:~# dd if=/dev/sdf bs=1 count=2560 skip=513189376 >> /out.xxx
2560+0 records in
2560+0 records out
2560 bytes (2.6 kB) copied, 0.0269378 s, 95.0 kB/s
root@chfiVBox:~# dd if=/dev/sdf bs=1 count=2560 skip=513193472 >> /out.xxx
2560+0 records in
2560+0 records out
2560 bytes (2.6 kB) copied, 0.0179637 s, 143 kB/s
root@chfiVBox:~# dd if=/dev/sdf bs=1 count=2560 skip=513197568 >> /out.xxx
2560+0 records in
2560+0 records out
2560 bytes (2.6 kB) copied, 0.0266771 s, 96.0 kB/s
root@chfiVBox:~# dd if=/dev/sdf bs=1 count=2560 skip=513201664 >> /out.xxx
2560+0 records in
2560+0 records out
2560 bytes (2.6 kB) copied, 0.0214631 s, 119 kB/s
root@chfiVBox:~# dd if=/dev/sdf bs=1 count=2560 skip=513205760 >> /out.xxx
2560+0 records in
2560+0 records out
2560 bytes (2.6 kB) copied, 0.00506321 s, 506 kB/s
root@chfiVBox:~# dd if=/dev/sdf bs=1 count=2560 skip=513209856 >> /out.xxx
2560+0 records in
2560+0 records out
2560 bytes (2.6 kB) copied, 0.0122342 s, 209 kB/s
root@chfiVBox:~# dd if=/dev/sdf bs=1 count=2560 skip=513213952 >> /out.xxx
2560+0 records in
2560+0 records out
2560 bytes (2.6 kB) copied, 0.00817506 s, 313 kB/s
root@chfiVBox:~#
```

then we can run the file command on linux, to figure aout what type of file it is (it should be a zip file, we already know it):

```
root@chfiVBox: ~
root@chfiVBox:~#
root@chfiVBox:~# file /out.xxx
/out.xxx: Zip archive data, at least v2.0 to extract
root@chfiVBox:~#
root@chfiVBox:~#
```

then we can extract it:



Microsoft Dynamic disk (Simple volume)

If we use a Microsoft dynamic disk it will have some effects:

- You will still find a Master Boot Record in the first sector of the disk, but it is only a dummy MBR, so it most probably points to an invalid position instead of the real start of the volume, but sometimes it points to the correct place, so just do not trust in it (mainly it will be incorrect if RAID, or other special features are used).
- You will find the dynamic disk header (PRIVHEAD) after the MBR (according to the descriptions immediately after the MBR, but I used to find it to start at the position 0x0C00). **I would like to emphasize, the numbers here are stored in big endian!**
- This header contains a pointer to a configuration data. Usually it is 1 megabyte before the end of the disk (recall, you need one megabyte unpartitioned space on the disk to create a dynamic disk).

```
root@chflVBox: ~
00000000  33 C0 8E D0 BC 00 7C FB 50 07 50 1F FC BE 1B 7C 3.....|.P.P...|
00000010  BF 1B 06 50 57 B9 E5 01 F3 A4 CB BD BE 07 B1 04 ...PW.....
00000020  38 6E 00 7C 09 75 13 83 C5 10 E2 F4 CD 18 8B F5 8n.|.u.....
00000030  83 C6 10 49 74 19 38 2C 74 F6 A0 B5 07 B4 07 8B ...It.8,t.....
00000040  F0 AC 3C 00 74 FC BB 07 00 B4 0E CD 10 EB F2 88 ..<.t.....
00000050  4E 10 E8 46 00 73 2A FE 46 10 80 7E 04 0B 74 0B N..F.s*.F..~.t.
00000060  80 7E 04 0C 74 05 A0 B6 07 75 D2 80 46 02 06 83 ..~.t....u..F...
00000070  46 08 06 83 56 0A 00 E8 21 00 73 05 A0 B6 07 EB F...V...!.s....
00000080  BC 81 3E FE 7D 55 AA 74 0B 80 7E 10 00 74 C8 A0 ..>}.U.t..~.t..
00000090  B7 07 EB A9 8B FC 1E 57 8B F5 CB BF 05 00 8A 56 .....W.....V
000000A0  00 B4 08 CD 13 72 23 8A C1 24 3F 98 8A DE 8A FC .....r#..$?....
000000B0  43 F7 E3 8B D1 86 D6 B1 06 D2 EE 42 F7 E2 39 56 C.....B..9V
000000C0  0A 77 23 72 05 39 46 08 73 1C B8 01 02 BB 00 7C .w#r.9F.s.....|
000000D0  8B 4E 02 8B 56 00 CD 13 73 51 4F 74 4E 32 E4 8A .N..V...sQ0tN2..
000000E0  56 00 CD 13 EB E4 8A 56 00 60 BB AA 55 B4 41 CD V.....V..U.A.
000000F0  13 72 36 81 FB 55 AA 75 30 F6 C1 01 74 2B 61 60 .r6..U.u0...t+a`
00000100  6A 00 6A 00 FF 76 0A FF 76 08 6A 00 68 00 7C 6A j.j..v..v.j.h.|j
00000110  01 6A 10 B4 42 8B F4 CD 13 61 61 73 0E 4F 74 0B .j..B....aas.Ot.
00000120  32 E4 8A 56 00 CD 13 EB D6 61 F9 C3 49 6E 76 61 2..V.....a..Inva
00000130  6C 69 64 20 70 61 72 74 69 74 69 6F 6E 20 74 61 lid partition ta
00000140  62 6C 65 00 45 72 72 6F 72 20 6C 6F 61 64 69 6E ble.Error loadin
00000150  67 20 6F 70 65 72 61 74 69 6E 67 20 73 79 73 74 g operating syst
00000160  65 6D 00 4D 69 73 73 69 6E 67 20 6F 70 65 72 61 em.Missing opera
00000170  74 69 6E 67 20 73 79 73 74 65 6D 00 00 00 00 00 ting system.....
00000180  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000190  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000001A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000001B0  00 00 00 00 00 2C 44 63 08 90 18 ED 00 00 00 01 .....,Dc.....
000001C0  01 00 42 FE 3F 81 3F 00 00 00 C3 DD 1F 00 00 00 ..B.??.?.....
000001D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000001E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000001F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA .....U.
--- sdg          --0x0/0x40000000-----
```

As we see the first partition record points to 3F000000 what means 0x3F and given in sectors. So

the start of the volume according to 0x7E00. Jump there by pressing ctrl-g then type 0x7E00

```

root@chfIVBox: ~
00007E00 EB 52 90 4E 54 46 53 20 20 20 20 00 02 02 00 00 .R.NTFS .....
00007E10 00 00 00 00 00 F8 00 00 3F 00 FF 00 3F 00 00 00 .....?...?...
00007E20 00 00 00 00 80 00 80 00 C2 DD 1F 00 00 00 00 00 .....
00007E30 A0 4F 05 00 00 00 00 00 70 F7 07 00 00 00 00 00 .0.....p.....
00007E40 01 00 00 00 04 00 00 00 E7 DD 9E A8 24 9F A8 BA .....$...
00007E50 00 00 00 00 FA 33 C0 8E D0 BC 00 7C FB B8 C0 07 .....3.....|....
00007E60 8E D8 E8 16 00 B8 00 0D 8E C0 33 DB C6 06 0E 00 .....3.....
00007E70 10 E8 53 00 68 00 0D 68 6A 02 CB 8A 16 24 00 B4 ..S.h..hj....$..
00007E80 08 CD 13 73 05 B9 FF FF 8A F1 66 0F B6 C6 40 66 ...s.....f...@f
00007E90 0F B6 D1 80 E2 3F F7 E2 86 CD C0 ED 06 41 66 0F .....?.....Af.
00007EA0 B7 C9 66 F7 E1 66 A3 20 00 C3 B4 41 BB AA 55 8A ..f..f. ...A..U.
00007EB0 16 24 00 CD 13 72 0F 81 FB 55 AA 75 09 F6 C1 01 .$.r...U.u....
00007EC0 74 04 FE 06 14 00 C3 66 60 1E 06 66 A1 10 00 66 t.....f`..f...f
00007ED0 03 06 1C 00 66 3B 06 20 00 0F 82 3A 00 1E 66 6A ....f;. ....fj
00007EE0 00 66 50 06 53 66 68 10 00 01 00 80 3E 14 00 00 .fP.Sfh.....>...
00007EF0 0F 85 0C 00 E8 B3 FF 80 3E 14 00 00 0F 84 61 00 .....>.....a.
00007F00 B4 42 8A 16 24 00 16 1F 8B F4 CD 13 66 58 5B 07 .B..$.....fX[.
00007F10 66 58 66 58 1F EB 2D 66 33 D2 66 0F B7 0E 18 00 fXfX..-f3.f.....
00007F20 66 F7 F1 FE C2 8A CA 66 8B D0 66 C1 EA 10 F7 36 f.....f..f....6
00007F30 1A 00 86 D6 8A 16 24 00 8A E8 C0 E4 06 0A CC B8 .....$.....
00007F40 01 02 CD 13 0F 82 19 00 8C C0 05 20 00 8E C0 66 ..... ..f
00007F50 FF 06 10 00 FF 0E 0E 00 0F 85 6F FF 07 1F 66 61 .....o...fa
00007F60 C3 A0 F8 01 E8 09 00 A0 FB 01 E8 03 00 FB EB FE .....
00007F70 B4 01 8B F0 AC 3C 00 74 09 B4 0E BB 07 00 CD 10 .....<.t.....
00007F80 EB F2 C3 0D 0A 41 20 64 69 73 6B 20 72 65 61 64 .....A disk read
00007F90 20 65 72 72 6F 72 20 6F 63 63 75 72 72 65 64 00 error occurred.
00007FA0 0D 0A 4E 54 4C 44 52 20 69 73 20 6D 69 73 73 69 ..NTLDR is missi
00007FB0 6E 67 00 0D 0A 4E 54 4C 44 52 20 69 73 20 63 6F ng...NTLDR is co
00007FC0 6D 70 72 65 73 73 65 64 00 0D 0A 50 72 65 73 73 mpressed...Press
00007FD0 20 43 74 72 6C 2B 41 6C 74 2B 44 65 6C 20 74 6F Ctrl+Alt+Del to
00007FE0 20 72 65 73 74 61 72 74 0D 0A 00 00 00 00 00 00 restart.....
00007FF0 00 00 00 00 00 00 00 00 83 A0 B3 C9 00 00 55 AA .....U.
--- sdg --0x7FF0/0x40000000-----

```

As we can see now it points to the correct place. But because we can not trust in it let us try to search it on the way it should be done in case of dynamic disks. First of all jump back to the beginning of the disk by pressing ctrl-g then type 0. Start to scroll down until you find the PRIVHEAD magic value:

```

root@chflVBox: ~
000000C00  50 52 49 56 48 45 41 44 00 00 31 BE 00 02 00 0B PRIVHEAD..1.....
000000C10  01 CD F2 8D E4 EB F3 50 00 00 00 00 00 00 00 06 .....P.....
000000C20  00 00 00 00 00 00 07 FF 00 00 00 00 00 00 07 40 .....@
000000C30  63 34 62 64 33 32 34 31 2D 66 39 63 36 2D 34 61 c4bd3241-f9c6-4a
000000C40  63 34 2D 39 63 34 64 2D 64 39 34 37 34 31 31 36 c4-9c4d-d9474116
000000C50  32 61 39 38 00 00 00 00 00 00 00 00 00 00 00 00 2a98.....
000000C60  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000C70  31 62 37 37 64 61 32 30 2D 63 37 31 37 2D 31 31 1b77da20-c717-11
000000C80  64 30 2D 61 35 62 65 2D 30 30 61 30 63 39 31 64 d0-a5be-00a0c91d
000000C90  62 37 33 63 00 00 00 00 00 00 00 00 00 00 00 00 b73c.....
000000CA0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000CB0  32 34 30 36 38 30 30 63 2D 38 38 33 31 2D 34 37 2406800c-8831-47
000000CC0  32 65 2D 62 30 37 33 2D 35 37 64 64 64 36 37 63 2e-b073-57ddd67c
000000CD0  39 33 30 35 00 00 00 00 00 00 00 00 00 00 00 00 9305.....
000000CE0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000CF0  54 61 6E 74 65 72 65 6D 78 70 44 67 31 00 00 00 TanteremxpDg1...
000000D00  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000D10  00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000D20  00 00 3F 00 00 00 00 00 1F DD C3 00 00 00 00 00 ..?.....
000000D30  1F F6 00 00 00 00 00 00 00 08 00 00 00 00 00 00 .....
000000D40  00 00 02 00 00 00 00 00 00 07 FD 00 00 00 01 00 .....
000000D50  00 00 01 00 00 00 00 00 00 05 C9 00 00 00 00 00 .....
000000D60  00 00 E0 DC 65 DC 65 00 00 00 00 00 00 00 00 00 ....e.e.....
000000D70  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000D80  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000D90  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000DA0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000DB0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000DC0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000DD0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000DE0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000DF0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
--- sdg          --0xC00/0x40000000-----

```

The PRIVHEAD block builds up as follows

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0x0000	PRIVHEAD (Magic value)								unknown				major version always 0x02		Minor version always 0x0B		
0x0010	Timestamp								Number								
0x0020	Size								Size								
0x0030	Disk ID GUID string, null padded																
0x0040																	
0x0050																	
0x0060																	
0x0070																	
0x0080	Host ID GUID string, null padded																
0x0090																	
0x00A0																	
0x00B0																	
0x00C0	Disk group ID GUID string, null padded																
0x00D0																	
0x00E0																	
0x00F0																	
0x0100	Disk Group Name string, null padded																
0x0110																	
0x0120	Unknown		Always 0										Logical Disk				
0x0130	start			Logical Disk Size										Configuration			
0x0140	Start			Configuration Size										Number of			
0x0150	TOCs			TOC size										Number of configs			Num-
0x0160	ber of logs			Size of config										Size of			
0x0170	Log			Disk Signature (or zero)					Disk Set								
0x0180	GUID								Disk Set								
0x0190	GUID (?)																

When we apply it to the actual data we get the following


```

root@chfIVBox: ~
00000C00  50 52 49 56 48 45 41 44 00 00 31 BE 00 02 00 0B PRIVHEAD..1.....
00000C10  01 CD F2 8D E4 EB F3 50 00 00 00 00 00 00 00 06 .....P.....
00000C20  00 00 00 00 00 00 07 FF 00 00 00 00 00 00 07 40 .....@
00000C30  63 34 62 64 33 32 34 31 2D 66 39 63 36 2D 34 61 c4bd3241-f9c6-4a
00000C40  63 34 2D 39 63 34 64 2D 64 39 34 37 34 31 31 36 c4-9c4d-d9474116
00000C50  32 61 39 38 00 00 00 00 00 00 00 00 00 00 00 00 2a98.....
00000C60  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000C70  31 62 37 37 64 61 32 30 2D 63 37 31 37 2D 31 31 1b77da20-c717-11
00000C80  64 30 2D 61 35 62 65 2D 30 30 61 30 63 39 31 64 d0-a5be-00a0c91d
00000C90  62 37 33 63 00 00 00 00 00 00 00 00 00 00 00 00 b73c.....
00000CA0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000CB0  32 34 30 36 38 30 30 63 2D 38 38 33 31 2D 34 37 2406800c-8831-47
00000CC0  32 65 2D 62 30 37 33 2D 35 37 64 64 64 36 37 63 2e-b073-57ddd67c
00000CD0  39 33 30 35 00 00 00 00 00 00 00 00 00 00 00 00 9305.....
00000CE0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000CF0  54 61 6E 74 65 72 65 6D 78 70 44 67 31 00 00 00 TanteremxpDg1...
00000D00  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000D10  00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000D20  00 00 3F 00 00 00 00 00 1F DD C3 00 00 00 00 00 00 ..?.....
00000D30  1F F6 00 00 00 00 00 00 00 08 00 00 00 00 00 00 .....
00000D40  00 00 02 00 00 00 00 00 00 07 FD 00 00 00 01 00 .....
00000D50  00 00 01 00 00 00 00 00 00 05 C9 00 00 00 00 00 .....
00000D60  00 00 E0 DC 65 DC 65 00 00 00 00 00 00 00 00 00 ....e.e.....
00000D70  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000D80  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000D90  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000DA0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000DB0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000DC0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000DD0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000DE0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000DF0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
--- sdg --0xC00/0x40000000-----

```

we should find the logical disk start value at the position (0x011B..0x0123) we will find here 000000000000003F until this point practically all the numbers were stored in little endian format so we would interpret it as 3F00000000000000 but in case of dynamic disk configuration block the numbers are usually stored in **big endian** format, so this value means simply 0x3F

So the data on the disk starts at the offset 0x3F (it is not surely the start of the volume yet, but we should add the offset of the volume start to this value).

Now let us try to find where the volume starts on this logical disk. To do it we need the configuration start value, what is 0000000001FF600. It means the configuration data starts at 0x1FF600 position. It is given in clusters so we must multiply by 512, $0x1FF600 * 0x200 = 3FEC0000$. Let us jump there by pressing ctrl-g then type 0x3FEC0000

```
root@chfIVBox: ~
3FEC0000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC00A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC00B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC00C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC00D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC00E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC00F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0100  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0110  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0120  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0130  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0140  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0150  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0160  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0170  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0180  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0190  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC01A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC01B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC01C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC01D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC01E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC01F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
--- sdg          --0x3FEC0000/0x40000000-----
```

here we found empty block, but if we start to scroll down a bit we find the configuration data. First the TOC block:


```

root@chfivBox: ~
3FEC0200  54 4F 43 42 4C 4F 43 4B 00 00 08 BA 00 00 00 00 TOCBLOCK.....
3FEC0210  00 00 00 05 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0220  00 00 00 00 63 6F 6E 66 69 67 00 00 00 00 00 ....config.....
3FEC0230  00 00 00 00 00 11 00 00 00 00 00 00 05 C9 00 06 .....
3FEC0240  00 01 00 00 00 00 6C 6F 67 00 00 00 00 00 00 .....log.....
3FEC0250  00 00 00 00 00 00 05 DA 00 00 00 00 00 00 00 E0 .....
3FEC0260  00 06 00 01 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0270  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0280  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0290  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC02A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC02B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC02C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC02D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC02E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC02F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0300  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0310  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0320  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0330  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0340  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0350  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0360  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0370  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0380  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC0390  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC03A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC03B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC03C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC03D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC03E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC03F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
--- sdq --0x3FEC0200/0x40000000-----

```

if we continue to scroll down we find after it there will be the VMDB block

```

root@chfIVBox: ~
3FEC2200 56 4D 44 42 00 00 17 24 00 00 00 80 00 00 02 00 VMDB...$.
3FEC2210 00 01 00 04 00 0A 54 61 6E 74 65 72 65 6D 78 70 .....Tanteremxp
3FEC2220 44 67 31 00 00 00 00 00 00 00 00 00 00 00 00 Dg1.....
3FEC2230 00 00 00 00 00 32 34 30 36 38 30 30 63 2D 38 38 .....2406800c-88
3FEC2240 33 31 2D 34 37 32 65 2D 62 30 37 33 2D 35 37 64 31-472e-b073-57d
3FEC2250 64 64 36 37 63 39 33 30 35 00 00 00 00 00 00 00 dd67c9305.....
3FEC2260 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2270 00 00 00 00 00 00 00 00 00 00 00 04 10 00 00 00 .....
3FEC2280 00 00 00 04 10 00 00 00 01 00 00 00 01 00 00 00 .....
3FEC2290 01 00 00 00 01 00 00 00 00 00 00 00 00 00 00 .....
3FEC22A0 00 00 00 00 01 00 00 00 01 00 00 00 01 00 00 00 .....
3FEC22B0 01 00 00 00 00 00 00 00 00 00 00 00 00 01 CD F2 .....
3FEC22C0 8D E4 FC C2 60 00 00 00 00 00 00 00 00 00 00 ....`.....
3FEC22D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC22E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC22F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2300 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2310 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2320 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2330 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2340 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2350 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2360 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2370 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2380 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2390 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC23A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC23B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC23C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC23D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC23E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC23F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
--- sdg --0x3FEC2200/0x40000000-----

```

And when continue the scroll we find the part what is the most important for us the VBLK blocks.

```

root@chfIVBox: ~
3FEC2400 56 42 4C 4B 00 00 00 04 00 00 00 00 00 00 00 00 VBLK.....
3FEC2410 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2420 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2430 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2440 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2450 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2460 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2470 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2480 56 42 4C 4B 00 00 00 05 00 00 00 02 00 00 00 01 VBLK.....
3FEC2490 00 00 08 35 00 00 00 4C 02 04 01 0D 54 61 6E 74 ...5...L....Tant
3FEC24A0 65 72 65 6D 78 70 44 67 31 24 32 34 30 36 38 30 eremxpDg1$240680
3FEC24B0 30 63 2D 38 38 33 31 2D 34 37 32 65 2D 62 30 37 0c-8831-472e-b07
3FEC24C0 33 2D 35 37 64 64 64 36 37 63 39 33 30 35 00 00 3-57ddd67c9305..
3FEC24D0 00 00 00 00 00 00 00 00 04 04 04 FF FF FF FF 04 .....
3FEC24E0 FF FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC24F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2500 56 42 4C 4B 00 00 00 06 00 00 00 00 00 00 00 00 VBLK.....
3FEC2510 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2520 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2530 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2540 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2550 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2560 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2570 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2580 56 42 4C 4B 00 00 00 07 00 00 00 01 00 00 00 01 VBLK.....
3FEC2590 00 00 00 34 00 00 00 3B 02 04 03 05 44 69 73 6B ...4...;....Disk
3FEC25A0 31 24 63 34 62 64 33 32 34 31 2D 66 39 63 36 2D 1$c4bd3241-f9c6-
3FEC25B0 34 61 63 34 2D 39 63 34 64 2D 64 39 34 37 34 31 4ac4-9c4d-d94741
3FEC25C0 31 36 32 61 39 38 00 00 00 00 00 00 00 00 00 00 162a98.....
3FEC25D0 00 04 10 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC25E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC25F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
--- sdg --0x3FEC2400/0x40000000-----

```

Structure of the VBLK blocks

Every VBLK block is 128 (0x80) bytes long, and starts with a 16 (0x10) bytes long header the structure of it is always the same.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x0000	Magic value (VBLK)				Sequence number (starts from 4)				Group number				Record number (x of y)		Number of records	
0x0010	Content of the VBLK block															
0x0020																
0x0030																
0x0040																
0x0050																
0x0060																
0x0070																

The content of the VBLK block is varying, the following ones can be seen on the disks:

Partition 0x33

Volume 0x51

Component 0x32

Disk 0x34 or 0x44 I have never seen this later one.

Disk group 0x35 or 0x45 I have never seen the late one

VLBK partition descriptor (0x33)

Now we will need practically only this information.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x0000	Magic value (VBLK)				Sequence number (starts from 4)				Group number				Record number (x of y)		Number of records	
0x0010	Update Status		Record type and flags		Data Length				Object ID length	Object ID		name length	Partition name			
0x0020	Volume name continuation				Always 0				Log commit ID							
0x0030	Start of partition								Volume Offset							
0x0040	length of partition size	partition size			Parent object ID length	Parent (a component) object ID		Disk object ID length	Disk object ID		Component index length	Component index (optional)				
0x0050																
0x0060																
0x0070																

Let us check it on our example:


```

root@chfIVBox: ~
3FEC2780 56 42 4C 4B 00 00 00 0B 00 00 00 07 00 00 00 01 VBLK.....
3FEC2790 00 00 00 33 00 00 00 32 02 04 0A 08 44 69 73 6B ...3...2....Disk
3FEC27A0 31 2D 30 31 00 00 00 00 00 00 00 00 00 00 04 0B 1-01.....
3FEC27B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC27C0 03 1F DD C3 02 04 08 02 04 03 00 00 00 00 00 00 .....
3FEC27D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC27E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC27F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2800 56 42 4C 4B 00 00 00 0C 00 00 00 05 00 00 00 01 VBLK.....
3FEC2810 00 00 00 32 00 00 00 30 02 04 08 0A 56 6F 6C 75 ...2...0....Volu
3FEC2820 6D 65 31 2D 30 31 06 41 43 54 49 56 45 02 00 00 me1-01.ACTIVE...
3FEC2830 00 00 01 01 00 00 00 00 00 00 04 10 00 00 00 00 .....
3FEC2840 00 00 00 00 02 04 06 00 00 00 00 00 00 00 00 00 .....
3FEC2850 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2860 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2870 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2880 56 42 4C 4B 00 00 00 0D 00 00 00 00 00 00 00 00 VBLK.....
3FEC2890 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC28A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC28B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC28C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC28D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC28E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC28F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2900 56 42 4C 4B 00 00 00 0E 00 00 00 00 00 00 00 00 VBLK.....
3FEC2910 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2920 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2930 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2940 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2950 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2960 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
3FEC2970 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
--- sdg          --0x3FEC2780/0x40000000-----

```

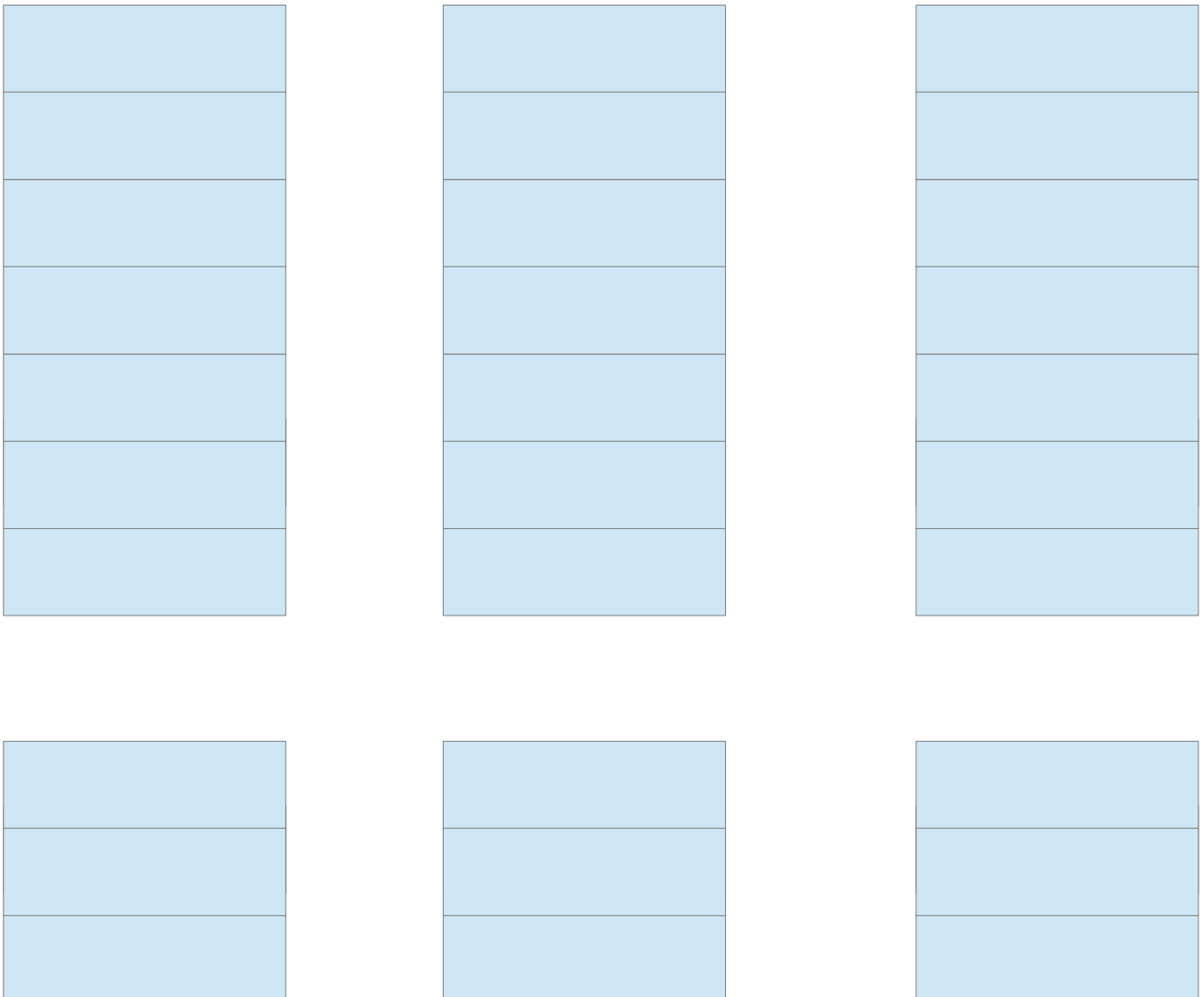
0x12..0x13 : the type of this VBLK, now it is 0x33 means partition.

0x30..0x37 : the start of the partition within the logical disk now it has a value of 0 so the partition will start at $0x3F + 0x00 = 0x3F$.

Microsoft dynamic disks (RAID0 stripe)

Now we have three disks, evidenceRAID0-disk0b.vhd (/dev/sdh), evidenceRAID0-disk1b.vhd (/dev/sdi), and evidenceRAID0-disk2b.vhd (/dev/sdj) and we created a stripe volume across these three disks. Let us try to reassemble this disk.

In this case the disks are divided to stripes. First we should know the stripe size as a basic information. When we want to write the data to the RAID array we divide it to stripe sizes, and write it to the disks at the same time.



Let us again try to find where does the NTFS volume starts. If we try again the MBR let see what we will get

```

root@chfivBox: ~
00000000  33 C0 8E D0 BC 00 7C 8E C0 8E D8 BE 00 7C BF 00 3.....|.....|..
00000010  06 B9 00 02 FC F3 A4 50 68 1C 06 CB FB B9 04 00 .....Ph.....
00000020  BD BE 07 80 7E 00 00 7C 0B 0F 85 10 01 83 C5 10 ....~..|.....
00000030  E2 F1 CD 18 88 56 00 55 C6 46 11 05 C6 46 10 00 .....V.U.F...F..
00000040  B4 41 BB AA 55 CD 13 5D 72 0F 81 FB 55 AA 75 09 .A..U..]r...U.u.
00000050  F7 C1 01 00 74 03 FE 46 10 66 60 80 7E 10 00 74 ....t..F.f`.~..t
00000060  26 66 68 00 00 00 00 66 FF 76 08 68 00 00 68 00 &fh....f.v.h..h.
00000070  7C 68 01 00 68 10 00 B4 42 8A 56 00 8B F4 CD 13 |h..h...B.V.....
00000080  9F 83 C4 10 9E EB 14 B8 01 02 BB 00 7C 8A 56 00 .....|.V.
00000090  8A 76 01 8A 4E 02 8A 6E 03 CD 13 66 61 73 1E FE .v..N..n...fas..
000000A0  4E 11 0F 85 0C 00 80 7E 00 80 0F 84 8A 00 B2 80 N.....~.....
000000B0  EB 82 55 32 E4 8A 56 00 CD 13 5D EB 9C 81 3E FE ..U2..V...].>.
000000C0  7D 55 AA 75 6E FF 76 00 E8 8A 00 0F 85 15 00 B0 }U.un.v.....
000000D0  D1 E6 64 E8 7F 00 B0 DF E6 60 E8 78 00 B0 FF E6 ..d.....`.x....
000000E0  64 E8 71 00 B8 00 BB CD 1A 66 23 C0 75 3B 66 81 d.q.....f#.u;f.
000000F0  FB 54 43 50 41 75 32 81 F9 02 01 72 2C 66 68 07 .TCPAu2....r,fh.
00000100  BB 00 00 66 68 00 02 00 00 66 68 08 00 00 00 66 ...fh....fh....f
00000110  53 66 53 66 55 66 68 00 00 00 00 66 68 00 7C 00 SfSfUfh....fh.|.
00000120  00 66 61 68 00 00 07 CD 1A 5A 32 F6 EA 00 7C 00 .fah.....Z2...|.
00000130  00 CD 18 A0 B7 07 EB 08 A0 B6 07 EB 03 A0 B5 07 .....
00000140  32 E4 05 00 07 8B F0 AC 3C 00 74 FC BB 07 00 B4 2.....<.t....
00000150  0E CD 10 EB F2 2B C9 E4 64 EB 00 24 02 E0 F8 24 .....+..d..$...$
00000160  02 C3 49 6E 76 61 6C 69 64 20 70 61 72 74 69 74 ..Invalid partit
00000170  69 6F 6E 20 74 61 62 6C 65 00 45 72 72 6F 72 20 ion table.Error
00000180  6C 6F 61 64 69 6E 67 20 6F 70 65 72 61 74 69 6E loading operatin
00000190  67 20 73 79 73 74 65 6D 00 4D 69 73 73 69 6E 67 g system.Missing
000001A0  20 6F 70 65 72 61 74 69 6E 67 20 73 79 73 74 65 operating syste
000001B0  6D 00 00 00 00 62 7A 99 4D 9E 79 4F 00 00 00 01 m....bz.M.y0....
000001C0  01 00 42 FE 3F 40 3F 00 00 00 C1 F7 0F 00 00 00 ..B.?@?.....
000001D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000001E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000001F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA .....U.
--- sdh --0x0/0x20000000-----

```

So according to the MBR the partition starts at $0x3F * 0x200 = 0x7E00$

If we jump there by pressing ctrl-g then type 0x7E00. There we find the following data

```
root@chfIVBox: ~
00007E00  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00007E10  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00007E20  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00007E30  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00007E40  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00007E50  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00007E60  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00007E70  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00007E80  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00007E90  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00007EA0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00007EB0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00007EC0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00007ED0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00007EE0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00007EF0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00007F00  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00007F10  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00007F20  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00007F30  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00007F40  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00007F50  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00007F60  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00007F70  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00007F80  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00007F90  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00007FA0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00007FB0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00007FC0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00007FD0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00007FE0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00007FF0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
--- sdh          --0x7E00/0x20000000-----
```

we will get the same result not only on /dev/sdh, but on /dev/sdi, and /dev/sdj as well. So now the Master Boot Record does not contain valid data on neither of the disks so now we must follow the technique used last time, to find the valid partition entry from the dynamic disk data. Now we will need some additional information as well, for example we must know the stripe size, or we should know the order of the disks as two very important ones.

Start to scroll down, to find the PRIVHEAD block again, it will be at the position 0x0C00

```

root@chfivBox: ~
000000C00  50 52 49 56 48 45 41 44 00 00 2F BE 00 02 00 0C PRIVHEAD../.....
000000C10  01 CC 27 8F 4E 44 A3 5C 00 00 00 00 00 00 00 01 ..'.ND.\.....
000000C20  00 00 00 00 00 00 07 FF 00 00 00 00 00 00 07 40 .....@
000000C30  31 32 64 64 62 61 37 64 2D 39 33 63 64 2D 31 31 12ddba7d-93cd-11
000000C40  65 30 2D 62 63 66 33 2D 30 30 30 38 34 34 34 34 e0-bcf3-00084444
000000C50  34 34 34 34 00 00 00 00 00 00 00 00 00 00 00 00 4444.....
000000C60  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000C70  31 62 37 37 64 61 32 30 2D 63 37 31 37 2D 31 31 1b77da20-c717-11
000000C80  64 30 2D 61 35 62 65 2D 30 30 61 30 63 39 31 64 d0-a5be-00a0c91d
000000C90  62 37 33 63 00 00 00 00 00 00 00 00 00 00 00 00 b73c.....
000000CA0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000CB0  31 32 64 64 62 61 37 63 2D 39 33 63 64 2D 31 31 12ddba7c-93cd-11
000000CC0  65 30 2D 62 63 66 33 2D 30 30 30 38 34 34 34 34 e0-bcf3-00084444
000000CD0  34 34 34 34 00 00 00 00 00 00 00 00 00 00 00 00 4444.....
000000CE0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000CF0  48 41 43 4B 2D 41 54 54 41 43 4B 45 52 2D 44 67 HACK-ATTACKER-Dg
000000D00  30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0.....
000000D10  00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000D20  00 00 3F 00 00 00 00 00 0F F7 C1 00 00 00 00 00 ..?.....
000000D30  0F F8 00 00 00 00 00 00 00 08 00 00 00 00 00 00 .....
000000D40  00 00 02 00 00 00 00 00 00 07 FD 00 00 00 01 00 .....
000000D50  00 00 01 00 00 00 00 00 00 05 C9 00 00 00 00 00 .....
000000D60  00 00 E0 FE E7 87 4C 00 00 00 00 00 00 00 00 00 .....L.....
000000D70  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000D80  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000D90  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000DA0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000DB0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000DC0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000DD0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000DE0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000DF0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
--- sdh          --0xC00/0x20000000-----

```

0x011B..0x0122 : The start of the logical disk is at 0x3F again this is not the start of the volume or partition, but the start of the logical disk, we will have to add this value to the start of the partition.
0x012B..0x0132 : The start of the configuration now it is 0xFF800 what means 0xFF800 * 0x200 = 0x1FF00000

press ctrl-g then type 0x1FF00000 to jump to this position. If start to scroll down we will find first the TOCBLOCK, then the VMDB, after it the VBLK entries.

Among them we can find again the partition entry 0x33

Now we have three disks so we will find three partitions one on every disk. As we see the partitions will start at the 0x41 position on every disk. And we can see that, the volume offset is 0 so on every partition the volume will start immediately on the beginning of the partition.

So let us start to calculate the beginning of the NTFS volume, it will be at the position 0x3F + 0x41 = 0x80, what means 0x80 * 0x200 = 0x10000


```
root@chfiVBox: ~
1FF02700 56 42 4C 4B 00 00 00 0A 00 00 00 0A 00 00 00 01 VBLK.....
1FF02710 00 00 40 33 00 00 00 30 01 07 08 44 69 73 6B 32 ..@3...0...Disk2
1FF02720 2D 30 31 00 00 00 00 00 00 00 00 00 00 00 07 00 -01.....
1FF02730 00 00 00 00 00 00 41 00 00 00 00 00 00 00 00 03 .....A.....
1FF02740 0F E8 00 01 06 01 03 00 00 00 00 00 00 00 00 00 .....
1FF02750 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF02760 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF02770 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF02780 56 42 4C 4B 00 00 00 0B 00 00 00 0B 00 00 00 01 VBLK.....
1FF02790 00 00 48 33 00 00 00 32 01 08 08 44 69 73 6B 31 ..H3...2...Disk1
1FF027A0 2D 30 31 00 00 00 00 00 00 00 00 00 00 00 07 00 -01.....
1FF027B0 00 00 00 00 00 00 41 00 00 00 00 00 00 00 00 03 .....A.....
1FF027C0 0F E8 00 01 06 01 02 01 01 00 00 00 00 00 00 00 .....
1FF027D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF027E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF027F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF02800 56 42 4C 4B 00 00 00 0C 00 00 00 0C 00 00 00 01 VBLK.....
1FF02810 00 00 48 33 00 00 00 32 01 09 08 44 69 73 6B 33 ..H3...2...Disk3
1FF02820 2D 30 31 00 00 00 00 00 00 00 00 00 00 00 07 00 -01.....
1FF02830 00 00 00 00 00 00 41 00 00 00 00 00 00 00 00 03 .....A.....
1FF02840 0F E8 00 01 06 01 04 01 02 00 00 00 00 00 00 00 .....
1FF02850 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF02860 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF02870 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF02880 56 42 4C 4B 00 00 00 0D 00 00 00 0F 00 00 00 01 VBLK.....
1FF02890 00 00 02 51 00 00 00 52 01 05 07 56 6F 6C 75 6D ...Q...R...Volum
1FF028A0 65 31 03 67 65 6E 00 41 43 54 49 56 45 00 00 00 e1.gen.ACTIVE...
1FF028B0 00 00 00 00 00 03 01 01 00 00 00 11 01 01 00 00 .....
1FF028C0 00 00 00 00 00 0A 00 00 00 00 00 00 00 00 03 2F ...../
1FF028D0 B8 00 00 00 00 00 07 12 DD BA 85 93 CD 11 E0 BC .....
1FF028E0 F3 00 08 44 44 44 44 02 44 3A 00 00 00 00 00 00 ...DDDD.D:.....
1FF028F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
--- sdh --0x1FF028F1/0x20000000-----
```

Let us jump to this position by pressing ctrl-g then type 0x10000


```
root@chfivBox: ~
00010000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000100A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000100B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000100C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000100D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000100E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000100F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000101A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000101B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000101C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000101D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000101E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000101F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
--- sdh --0x10000/0x20000000-----
```

As we can see there is nothing. What happened? Again we have three disks, so check the other two as well, and really if we check the other disks the /dev/sdj is also empty, but on the /dev/sdi we will find the NTFS volume start:

```

root@chfIVBox: ~
00010000 EB 52 90 4E 54 46 53 20 20 20 20 00 02 08 00 00 .R.NTFS .....
00010010 00 00 00 00 00 F8 00 00 3F 00 FF 00 3F 00 00 00 .....?...?...
00010020 00 00 00 00 80 00 80 00 FF B7 2F 00 00 00 00 00 ...../.....
00010030 00 FD 01 00 00 00 00 00 7F FB 02 00 00 00 00 00 .....
00010040 F6 00 00 00 01 00 00 00 7F CB DC 00 E3 DC 00 96 .....
00010050 00 00 00 00 FA 33 C0 8E D0 BC 00 7C FB 68 C0 07 .....3.....|.h..
00010060 1F 1E 68 66 00 CB 88 16 0E 00 66 81 3E 03 00 4E ..hf.....f.>..N
00010070 54 46 53 75 15 B4 41 BB AA 55 CD 13 72 0C 81 FB TFSu..A..U..r...
00010080 55 AA 75 06 F7 C1 01 00 75 03 E9 D2 00 1E 83 EC U.u.....u.....
00010090 18 68 1A 00 B4 48 8A 16 0E 00 8B F4 16 1F CD 13 .h...H.....
000100A0 9F 83 C4 18 9E 58 1F 72 E1 3B 06 0B 00 75 DB A3 .....X.r.;...u..
000100B0 0F 00 C1 2E 0F 00 04 1E 5A 33 DB B9 00 20 2B C8 .....Z3...+.
000100C0 66 FF 06 11 00 03 16 0F 00 8E C2 FF 06 16 00 E8 f.....
000100D0 40 00 2B C8 77 EF B8 00 BB CD 1A 66 23 C0 75 2D @.+..w.....f#.u-
000100E0 66 81 FB 54 43 50 41 75 24 81 F9 02 01 72 1E 16 f..TCPAu$.r..
000100F0 68 07 BB 16 68 70 0E 16 68 09 00 66 53 66 53 66 h...hp..h..fsfsf
00010100 55 16 16 16 68 B8 01 66 61 0E 07 CD 1A E9 6A 01 U...h..fa.....j.
00010110 90 90 66 60 1E 06 66 A1 11 00 66 03 06 1C 00 1E ..f`.f...f.....
00010120 66 68 00 00 00 00 66 50 06 53 68 01 00 68 10 00 fh....fP.Sh..h..
00010130 B4 42 8A 16 0E 00 16 1F 8B F4 CD 13 66 59 5B 5A .B.....fY[Z
00010140 66 59 66 59 1F 0F 82 16 00 66 FF 06 11 00 03 16 fYfY.....f.....
00010150 0F 00 8E C2 FF 0E 16 00 75 BC 07 1F 66 61 C3 A0 .....u...fa..
00010160 F8 01 E8 08 00 A0 FB 01 E8 02 00 EB FE B4 01 8B .....
00010170 F0 AC 3C 00 74 09 B4 0E BB 07 00 CD 10 EB F2 C3 ..<.t.....
00010180 0D 0A 41 20 64 69 73 6B 20 72 65 61 64 20 65 72 ..A disk read er
00010190 72 6F 72 20 6F 63 63 75 72 72 65 64 00 0D 0A 42 ror occurred...B
000101A0 4F 4F 54 4D 47 52 20 69 73 20 6D 69 73 73 69 6E 00TMGR is missin
000101B0 67 00 0D 0A 42 4F 4F 54 4D 47 52 20 69 73 20 63 g...BOOTMGR is c
000101C0 6F 6D 70 72 65 73 73 65 64 00 0D 0A 50 72 65 73 ompressed...Pres
000101D0 73 20 43 74 72 6C 2B 41 6C 74 2B 44 65 6C 20 74 s Ctrl+Alt+Del t
000101E0 6F 20 72 65 73 74 61 72 74 0D 0A 00 00 00 00 0 o restart.....
000101F0 00 00 00 00 00 00 00 00 80 9D B2 CA 00 00 55 AA .....U.
--- sdi --0x101F0/0x20000000-----

```

So we can see that, it is important to figure out the order of the disks otherwise we can not really find the information, to do it let us write down the GUID of every disk what we find in the PRIVHEAD part. In my example it can be found at the position 0x0C00.

```

root@chfIVBox: ~
00000C00 50 52 49 56 48 45 41 44 00 00 2F BE 00 02 00 0C PRIVHEAD../.....
00000C10 01 CC 27 8F 4E 44 A3 5C 00 00 00 00 00 00 00 01 ..'.ND.\.....
00000C20 00 00 00 00 00 00 07 FF 00 00 00 00 00 00 07 40 .....@
00000C30 31 32 64 64 62 61 37 64 2D 39 33 63 64 2D 31 31 12ddba7d-93cd-11
00000C40 65 30 2D 62 63 66 33 2D 30 30 30 38 34 34 34 34 e0-bcf3-00084444
00000C50 34 34 34 34 00 00 00 00 00 00 00 00 00 00 00 00 4444.....
00000C60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
--- sdh --0xC00/0x20000000-----

```

```
root@chfIVBox: ~
00000C00  50 52 49 56 48 45 41 44 00 00 30 2B 00 02 00 0C PRIVHEAD..0+....
00000C10  01 CC 27 8F 50 A7 DE 5C 00 00 00 00 00 00 00 01 ..'.P..\.....
00000C20  00 00 00 00 00 00 07 FF 00 00 00 00 00 00 07 40 .....@
00000C30  31 32 64 64 62 61 38 30 2D 39 33 63 64 2D 31 31 12ddba80-93cd-11
00000C40  65 30 2D 62 63 66 33 2D 30 30 30 38 34 34 34 34 e0-bcf3-00084444
00000C50  34 34 34 34 00 00 00 00 00 00 00 00 00 00 00 00 4444.....
00000C60  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
--- sdi --0xC00/0x20000000-----

root@chfIVBox: ~
00000C00  50 52 49 56 48 45 41 44 00 00 2F 84 00 02 00 0C PRIVHEAD../.....
00000C10  01 CC 27 8F 51 29 C1 4C 00 00 00 00 00 00 00 01 ..'.Q).L.....
00000C20  00 00 00 00 00 00 07 FF 00 00 00 00 00 00 07 40 .....@
00000C30  31 32 64 64 62 61 38 33 2D 39 33 63 64 2D 31 31 12ddba83-93cd-11
00000C40  65 30 2D 62 63 66 33 2D 30 30 30 38 34 34 34 34 e0-bcf3-00084444
00000C50  34 34 34 34 00 00 00 00 00 00 00 00 00 00 00 00 4444.....
00000C60  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
--- sdj --0xC00/0x20000000-----
```

Now we know the GUID of every disk:

```
/dev/sdh : 12ddba7d-93cd-11e0-bcf3-000844444444
/dev/sdi : 12ddba80-93cd-11e0-bcf3-000844444444
/dev/sdj : 12ddba83-93cd-11e0-bcf3-000844444444
```

Let us which disk does it mean. We can find this information in another VBLK entry what we have not used yet

VBLK Disk descriptor (0x34)

The disk descriptor has the following structure:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x0000	Magic value (VBLK)				Sequence number (starts from 4)				Group number				Record number (x of y)		Number of records	
0x0010	Update Status		Record type and flags		Data Length				Object ID length	Object ID		name length	Diskname			
0x0020	Length of disk ID	DISK ID (GUID string)														
0x0030																
0x0040					Alternate name length		Alternate name		Always 0				Log commit ID			
0x0050	Log commit ID cont.															
0x0060																
0x0070																

if we check this structure there are two important information the Disk Name, and the Disk GUID. We already know the Disk GUID from the PRIVHEAD field of every disks so we can pair them.


```

root@chfIVBox: ~
1FF02500 56 42 4C 4B 00 00 00 06 00 00 00 03 00 00 00 01 VBLK.....
1FF02510 00 00 00 34 00 00 00 3A 01 02 05 44 69 73 6B 31 ...4.....Disk1
1FF02520 24 31 32 64 64 62 61 37 64 2D 39 33 63 64 2D 31 $12ddba7d-93cd-1
1FF02530 31 65 30 2D 62 63 66 33 2D 30 30 30 38 34 34 34 1e0-bcf3-0008444
1FF02540 34 34 34 34 34 00 00 00 00 00 00 00 00 00 00 00 44444
1FF02550 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF02560 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF02570 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF02580 56 42 4C 4B 00 00 00 07 00 00 00 05 00 00 00 01 VBLK.....
1FF02590 00 00 00 34 00 00 00 3A 01 03 05 44 69 73 6B 32 ...4.....Disk2
1FF025A0 24 31 32 64 64 62 61 38 30 2D 39 33 63 64 2D 31 $12ddba80-93cd-1
1FF025B0 31 65 30 2D 62 63 66 33 2D 30 30 30 38 34 34 34 1e0-bcf3-0008444
1FF025C0 34 34 34 34 34 00 00 00 00 00 00 00 00 00 00 00 44444
1FF025D0 00 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF025E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF025F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF02600 56 42 4C 4B 00 00 00 08 00 00 00 07 00 00 00 01 VBLK.....
1FF02610 00 00 00 34 00 00 00 3A 01 04 05 44 69 73 6B 33 ...4.....Disk3
1FF02620 24 31 32 64 64 62 61 38 33 2D 39 33 63 64 2D 31 $12ddba83-93cd-1
1FF02630 31 65 30 2D 62 63 66 33 2D 30 30 30 38 34 34 34 1e0-bcf3-0008444
1FF02640 34 34 34 34 34 00 00 00 00 00 00 00 00 00 00 00 44444
1FF02650 00 06 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF02660 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF02670 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF02680 56 42 4C 4B 00 00 00 09 00 00 00 09 00 00 00 01 VBLK.....
1FF02690 00 00 10 32 00 00 00 32 01 06 0A 56 6F 6C 75 6D ...2...2...Volum
1FF026A0 65 31 2D 30 31 06 41 43 54 49 56 45 01 00 00 00 e1-01.ACTIVE...
1FF026B0 00 01 03 00 00 00 00 00 00 00 07 00 00 00 00 00 .....
1FF026C0 00 00 00 01 05 00 01 80 01 03 00 00 00 00 00 00 .....
1FF026D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF026E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF026F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
--- sdi --0x1FF02500/0x20000000-----

```

From here we get:

```

Disk1 : 12ddba7d-93cd-11e0-bcf3-000844444444
Disk2 : 12ddba80-93cd-11e0-bcf3-000844444444
Disk3 : 12ddba83-93cd-11e0-bcf3-000844444444

```

if we pair it with the other information:

```

/dev/sdh : 12ddba7d-93cd-11e0-bcf3-000844444444
/dev/sdi : 12ddba80-93cd-11e0-bcf3-000844444444
/dev/sdj : 12ddba83-93cd-11e0-bcf3-000844444444

```

then we get

```

Disk1 : /dev/sdh
Disk2 : /dev/sdi
Disk3 : /dev/sdj

```

So we identified the disks. Now we need the order. This information can be found in the partition

VBLK entries

```

root@chfiVBox: ~
1FF02700 56 42 4C 4B 00 00 00 0A 00 00 00 0A 00 00 00 01 VBLK.....
1FF02710 00 00 40 33 00 00 00 30 01 07 08 44 69 73 6B 32 ..@3...0...Disk2
1FF02720 2D 30 31 00 00 00 00 00 00 00 00 00 00 00 07 00 -01.....
1FF02730 00 00 00 00 00 00 41 00 00 00 00 00 00 00 00 03 .....A.....
1FF02740 0F E8 00 01 06 01 03 00 00 00 00 00 00 00 00 00 .....
1FF02750 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF02760 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF02770 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF02780 56 42 4C 4B 00 00 00 0B 00 00 00 0B 00 00 00 01 VBLK.....
1FF02790 00 00 48 33 00 00 00 32 01 08 08 44 69 73 6B 31 ..H3...2...Disk1
1FF027A0 2D 30 31 00 00 00 00 00 00 00 00 00 00 00 07 00 -01.....
1FF027B0 00 00 00 00 00 00 41 00 00 00 00 00 00 00 00 03 .....A.....
1FF027C0 0F E8 00 01 06 01 02 01 01 00 00 00 00 00 00 00 .....
1FF027D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF027E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF027F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF02800 56 42 4C 4B 00 00 00 0C 00 00 00 0C 00 00 00 01 VBLK.....
1FF02810 00 00 48 33 00 00 00 32 01 09 08 44 69 73 6B 33 ..H3...2...Disk3
1FF02820 2D 30 31 00 00 00 00 00 00 00 00 00 00 00 07 00 -01.....
1FF02830 00 00 00 00 00 00 41 00 00 00 00 00 00 00 00 03 .....A.....
1FF02840 0F E8 00 01 06 01 04 01 02 00 00 00 00 00 00 00 .....
1FF02850 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF02860 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF02870 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF02880 56 42 4C 4B 00 00 00 0D 00 00 00 0F 00 00 00 01 VBLK.....
1FF02890 00 00 02 51 00 00 00 52 01 05 07 56 6F 6C 75 6D ...Q...R...Volum
1FF028A0 65 31 03 67 65 6E 00 41 43 54 49 56 45 00 00 00 e1.gen.ACTIVE...
1FF028B0 00 00 00 00 00 03 01 01 00 00 00 11 01 01 00 00 .....
1FF028C0 00 00 00 00 00 0A 00 00 00 00 00 00 00 00 03 2F ...../
1FF028D0 B8 00 00 00 00 00 07 12 DD BA 85 93 CD 11 E0 BC .....
1FF028E0 F3 00 08 44 44 44 44 02 44 3A 00 00 00 00 00 00 ...DDDD.D:.....
1FF028F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
--- sdh --0x1FF028F1/0x20000000-----

```

Here the last piece of information is the component index optional information (if you check the picture can recognize, it is not exist in the first block, I just marked the “place of it”). Here you can read the order:

```

Disk2 : 0
Disk1 : 1
Disk3 : 2

```

So the order of the disks what we wanted to know:

```

1. Disk1 : /dev/sdh
0. Disk2 : /dev/sdi
2. Disk3 : /dev/sdj

```

This is why we found the NTFS entry on the /dev/sdi, that is the 0th disk.

The other information we should know is the stripe size, let us try to find it as well. This piece of

information can be found in the following VBLK entry:

VBLK Component descriptor (0x32)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x0000	Magic value (VBLK)				Sequence number (starts from 4)				Group number				Record number (x of y)		Number of records	
0x0010	Update Status		Record type and flags		Data Length				Object ID length	Object ID		name length	volume name			
0x0020	volume name cont.						volume state length	Volume state					component type 1:stripe, 2: basic or span, 3: RAID	Always 0		
0x0030	Always 0		length of number of children	Number of children	Log commit ID								Always 0			
0x0040	Always 0				length of parent id	Parent (a volume) ID		Always 0	Length of stripe size	stripe size		length of number of disks	number of disks			
0x0050																
0x0060																
0x0070																

Here the one before the last value is the stripe size. Let us check it on our example

```

root@chfIVBox: ~
1FF02680 56 42 4C 4B 00 00 00 09 00 00 00 09 00 00 00 01 VBLK.....
1FF02690 00 00 10 32 00 00 00 32 01 06 0A 56 6F 6C 75 6D ...2...2...Volum
1FF026A0 65 31 2D 30 31 06 41 43 54 49 56 45 01 00 00 00 e1-01.ACTIVE....
1FF026B0 00 01 03 00 00 00 00 00 00 00 07 00 00 00 00 00 .....
1FF026C0 00 00 00 01 05 00 01 80 01 03 00 00 00 00 00 00 .....
1FF026D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF026E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF026F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF02700 56 42 4C 4B 00 00 00 0A 00 00 00 0A 00 00 00 01 VBLK.....
1FF02710 00 00 40 33 00 00 00 30 01 07 08 44 69 73 6B 32 ..@3...0...Disk2
1FF02720 2D 30 31 00 00 00 00 00 00 00 00 00 00 00 07 00 -01.....
1FF02730 00 00 00 00 00 00 41 00 00 00 00 00 00 00 00 03 .....A.....
1FF02740 0F E8 00 01 06 01 03 00 00 00 00 00 00 00 00 00 .....
1FF02750 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF02760 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF02770 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF02780 56 42 4C 4B 00 00 00 0B 00 00 00 0B 00 00 00 01 VBLK.....
1FF02790 00 00 48 33 00 00 00 32 01 08 08 44 69 73 6B 31 ..H3...2...Disk1
1FF027A0 2D 30 31 00 00 00 00 00 00 00 00 00 00 00 07 00 -01.....
1FF027B0 00 00 00 00 00 00 41 00 00 00 00 00 00 00 00 03 .....A.....
1FF027C0 0F E8 00 01 06 01 02 01 01 00 00 00 00 00 00 00 .....
1FF027D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF027E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF027F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF02800 56 42 4C 4B 00 00 00 0C 00 00 00 0C 00 00 00 01 VBLK.....
1FF02810 00 00 48 33 00 00 00 32 01 09 08 44 69 73 6B 33 ..H3...2...Disk3
1FF02820 2D 30 31 00 00 00 00 00 00 00 00 00 00 00 07 00 -01.....
1FF02830 00 00 00 00 00 00 41 00 00 00 00 00 00 00 00 03 .....A.....
1FF02840 0F E8 00 01 06 01 04 01 02 00 00 00 00 00 00 00 .....
1FF02850 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF02860 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FF02870 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
--- sdi --0x1FF02870/0x20000000-----

```

Here we see it is a component entry (0x32), and the stripe size is stored on 0x01 byte and the value of it is 0x80. What means, the stripe size now is $0x80 * 0x200 = 0x10000 = 65536$ byte (64 kB)

Now we know everything, to reassemble a file on the disk.

Reassemble of a file

Let us check the NTFS entry, and search the first non system file. As you remember the NTFS volume begins on disk /dev/sdi at the position 0x10000

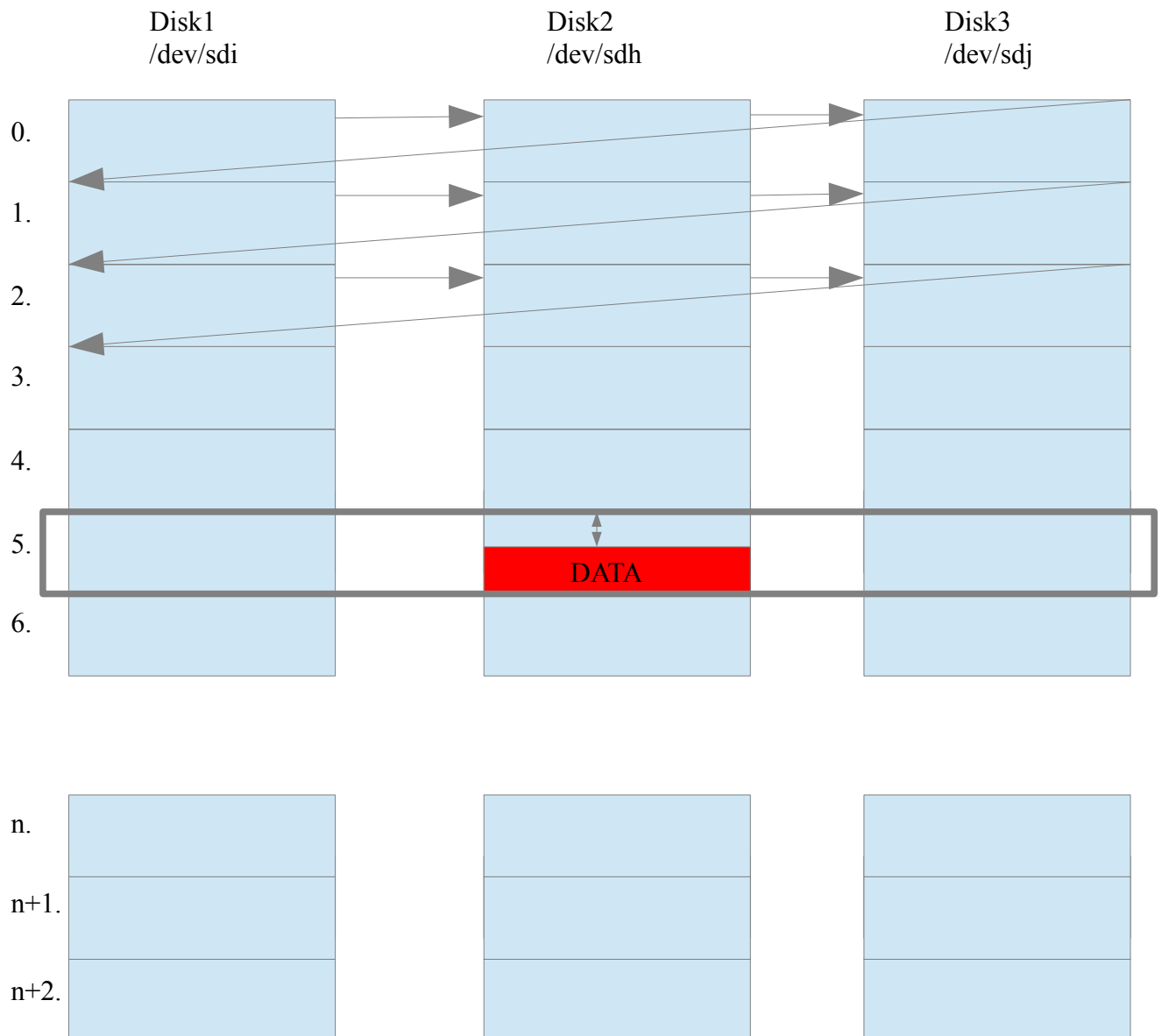
```
root@chfIVBox: ~
00010000 EB 52 90 4E 54 46 53 20 20 20 20 00 02 08 00 00 .R.NTFS .....
00010010 00 00 00 00 00 F8 00 00 3F 00 FF 00 3F 00 00 00 .....?...?...
00010020 00 00 00 00 80 00 80 00 FF B7 2F 00 00 00 00 00 ...../.....
00010030 00 FD 01 00 00 00 00 00 7F FB 02 00 00 00 00 00 .....
00010040 F6 00 00 00 01 00 00 00 7F CB DC 00 E3 DC 00 96 .....
00010050 00 00 00 00 FA 33 C0 8E D0 BC 00 7C FB 68 C0 07 .....3.....|.h..
00010060 1F 1E 68 66 00 CB 88 16 0E 00 66 81 3E 03 00 4E ..hf.....f.>..N
00010070 54 46 53 75 15 B4 41 BB AA 55 CD 13 72 0C 81 FB TFSu..A..U..r....
00010080 55 AA 75 06 F7 C1 01 00 75 03 E9 D2 00 1E 83 EC U.u.....u.....
00010090 18 68 1A 00 B4 48 8A 16 0E 00 8B F4 16 1F CD 13 .h...H.....
000100A0 9F 83 C4 18 9E 58 1F 72 E1 3B 06 0B 00 75 DB A3 .....X.r.;...u..
000100B0 0F 00 C1 2E 0F 00 04 1E 5A 33 DB B9 00 20 2B C8 .....Z3...+..
000100C0 66 FF 06 11 00 03 16 0F 00 8E C2 FF 06 16 00 E8 f.....
000100D0 40 00 2B C8 77 EF B8 00 BB CD 1A 66 23 C0 75 2D @.+..w.....f#.u-
000100E0 66 81 FB 54 43 50 41 75 24 81 F9 02 01 72 1E 16 f..TCPAu$.r..
000100F0 68 07 BB 16 68 70 0E 16 68 09 00 66 53 66 53 66 h...hp..h..fSfSf
00010100 55 16 16 16 68 B8 01 66 61 0E 07 CD 1A E9 6A 01 U...h..fa.....j.
00010110 90 90 66 60 1E 06 66 A1 11 00 66 03 06 1C 00 1E ..f`..f...f.....
00010120 66 68 00 00 00 00 66 50 06 53 68 01 00 68 10 00 fh....fP.Sh..h..
00010130 B4 42 8A 16 0E 00 16 1F 8B F4 CD 13 66 59 5B 5A .B.....fY[Z
00010140 66 59 66 59 1F 0F 82 16 00 66 FF 06 11 00 03 16 fYfY.....f.....
00010150 0F 00 8E C2 FF 0E 16 00 75 BC 07 1F 66 61 C3 A0 .....u...fa..
00010160 F8 01 E8 08 00 A0 FB 01 E8 02 00 EB FE B4 01 8B .....
00010170 F0 AC 3C 00 74 09 B4 0E BB 07 00 CD 10 EB F2 C3 ..<.t.....
00010180 0D 0A 41 20 64 69 73 6B 20 72 65 61 64 20 65 72 ..A disk read er.
00010190 72 6F 72 20 6F 63 63 75 72 72 65 64 00 0D 0A 42 ror occurred...B
000101A0 4F 4F 54 4D 47 52 20 69 73 20 6D 69 73 73 69 6E OOTMGR is missin
000101B0 67 00 0D 0A 42 4F 4F 54 4D 47 52 20 69 73 20 63 g...BOOTMGR is c.
000101C0 6F 6D 70 72 65 73 73 65 64 00 0D 0A 50 72 65 73 ompressed...Pres
000101D0 73 20 43 74 72 6C 2B 41 6C 74 2B 44 65 6C 20 74 s Ctrl+Alt+Del t.
000101E0 6F 20 72 65 73 74 61 72 74 0D 0A 00 00 00 00 00 o restart.....
000101F0 00 00 00 00 00 00 00 00 80 9D B2 CA 00 00 55 AA .....U.
--- sdi --0x10000/0x20000000-----
```

0x0B..0x0D : As we can see here the block size is 0x0200 (as usual) and one cluster contains 0x08 blocks (as usual). So the cluster size is $0x0200 * 0x08 = 0x1000 = 4096$ bytes again

0x30..0x37 : The MFT file starts at 0x01FD00. If you recall the previous calculations, it means the file starts at $0x01FD00 * 0x1000 + 0x10000 = 0x1FD10000$. **But this calculation is totally wrong in this case.** If you jump to this position on any of the disks you find there nothing.

Now the calculation works on different way, because instead of writing continuously to one disk now the data written to three disks. So the calculation should be done on the following way:

- Calculate, in which stripe the data is.
- Calculate within that stripe on which disk the data is
- Calculate on that disk within that stripe where exactly the data starts



Let us calculate the correct position now. The MFT start at 0x01FD00, what is given in clusters so we should multiply it by the cluster size what is 0x1000 now. $0x01FD00 * 0x1000 = 0x1FD00000$.

Now we should calculate first, in which stripe this byte exists. So simply divide this value by the stripe size, what is 0x10000 multiplied by the number of disks what is 3 now, and take the integer part of it. $INT(0x1FD00000 / (3 * 0x10000)) = 0x0A9A$ so the MFT will be somewhere in the 0x0A9Ath stripe.

The second step is to calculate within this stripe which disk will contain the data. To get it we should subtract from the original offset (0x1FD00000) the stripe number multiplied by the stripe size multiplied by the number of disks. So $0x1FD00000 - 0x0A9A * 3 * 0x10000 = 0x1FD00000 - 0x1FCE0000 = 0x20000$. Now we should divide this value by the size of one stripe, and take the integer part of it $INT(0x20000 / 0x10000) = 0x02$ what means, the data will be on disk 3 (we started the numbering of the disks from 1 not from 0 this is why we should add one now).

The final step is to calculate the offset within this stripe. To get it take the previous 0x20000 value and subtract from it the disk number (the 0x02, not the one because of our numbering) multiplied by the stripe size: $0x20000 - 0x02 * 0x10000 = 0$ What means, the data will immediately at the beginning of the stripe without any offset.

So we find the MFT file on the disk 3 (for me now it is /dev/sdj), at the beginning of the 0x0A9A stripe. So the position is $0x0A9A * 0x10000 = 0x0A9A0000$. But do not forget now everything is measured from the beginning of the logical disk not from the beginning of the physical disk so we should add to it 0x10000 as we did earlier. So the final offset on disk 3 (/dev/sdj) is $0x0A9A0000 + 0x10000 = 0x0A9B0000$. If we check it really we see the next:

```

root@chfiVBox: ~
0A9B0000  46 49 4C 45 30 00 03 00 8D 10 40 00 00 00 00 00 FILE0.....@.....
0A9B0010  01 00 01 00 38 00 01 00 98 01 00 00 00 04 00 00 ....8.....
0A9B0020  00 00 00 00 00 00 00 00 06 00 00 00 00 00 00 00 .....
0A9B0030  03 00 00 00 00 00 00 00 10 00 00 00 60 00 00 00 .....`...
0A9B0040  00 00 18 00 00 00 00 00 48 00 00 00 18 00 00 00 .....H.....
0A9B0050  74 7E 5F DB 8F 27 CC 01 74 7E 5F DB 8F 27 CC 01 t~_..'.t~_..'.
0A9B0060  74 7E 5F DB 8F 27 CC 01 74 7E 5F DB 8F 27 CC 01 t~_..'.t~_..'.
0A9B0070  06 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A9B0080  00 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 .....
0A9B0090  00 00 00 00 00 00 00 00 30 00 00 00 68 00 00 00 .....0...h...
0A9B00A0  00 00 18 00 00 00 03 00 4A 00 00 00 18 00 01 00 .....J.....
0A9B00B0  05 00 00 00 00 00 05 00 74 7E 5F DB 8F 27 CC 01 .....t~_..'.
0A9B00C0  74 7E 5F DB 8F 27 CC 01 74 7E 5F DB 8F 27 CC 01 t~_..'.t~_..'.
0A9B00D0  74 7E 5F DB 8F 27 CC 01 00 40 00 00 00 00 00 00 t~_..'....@.....
0A9B00E0  00 40 00 00 00 00 00 00 06 00 00 00 00 00 00 00 .@.....
0A9B00F0  04 03 24 00 4D 00 46 00 54 00 00 00 00 00 00 00 ..$.M.F.T.....
0A9B0100  80 00 00 00 48 00 00 00 01 00 40 00 00 00 01 00 ....H.....@.....
0A9B0110  00 00 00 00 00 00 00 00 0F 00 00 00 00 00 00 00 .....
0A9B0120  40 00 00 00 00 00 00 00 00 00 01 00 00 00 00 00 @.....
0A9B0130  00 00 01 00 00 00 00 00 00 00 01 00 00 00 00 00 .....
0A9B0140  31 10 00 FD 01 00 BC 91 B0 00 00 00 48 00 00 00 1.....H...
0A9B0150  01 00 40 00 00 00 05 00 00 00 00 00 00 00 00 00 ..@.....
0A9B0160  00 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 .....@.....
0A9B0170  00 10 00 00 00 00 00 00 08 00 00 00 00 00 00 00 .....
0A9B0180  08 00 00 00 00 00 00 00 31 01 FF FC 01 00 00 00 .....1.....
0A9B0190  FF FF FF FF 00 00 00 00 00 00 01 00 00 00 00 00 .....
0A9B01A0  00 00 01 00 00 00 00 00 31 10 00 FD 01 00 BC 91 .....1.....
0A9B01B0  B0 00 00 00 48 00 00 00 01 00 40 00 00 00 05 00 ....H.....@.....
0A9B01C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A9B01D0  40 00 00 00 00 00 00 00 00 10 00 00 00 00 00 00 @.....
0A9B01E0  08 00 00 00 00 00 00 00 08 00 00 00 00 00 00 00 .....
0A9B01F0  31 01 FF FC 01 00 00 00 FF FF FF FF 00 00 03 00 1.....
--- sdj          --0xA9B01F0/0x20000000-----

```

Again as we used to do we should find the first non system entry, what used to be the 35th one. Because one entry is 1 kB in size $35 * 1 \text{ kB} = 35 \text{ kB}$, the MFT started at the beginning of the stripe, and the stripe size is 64 kB now, so it will be on this disk, we should only scroll down to find it.

$35 * 0x400 + 0x0A9B0000 = 0x0A9B8C00$ lets go there by pressing ctrl-g then type this value

```

root@chfiVBox: ~
0A9B8C00  46 49 4C 45 30 00 03 00 5C 26 40 00 00 00 00 00 FILE0...\&@.....
0A9B8C10  01 00 01 00 38 00 01 00 60 01 00 00 00 04 00 00 ....8...`.....
0A9B8C20  00 00 00 00 00 00 00 00 04 00 00 00 23 00 00 00 .....#...
0A9B8C30  03 00 00 00 00 00 00 00 10 00 00 00 60 00 00 00 .....`...
0A9B8C40  00 00 00 00 00 00 00 00 48 00 00 00 18 00 00 00 .....H.....
0A9B8C50  7C C7 77 CF 91 27 CC 01 54 7F 09 D7 91 27 CC 01 |.w..'.T....'..
0A9B8C60  54 7F 09 D7 91 27 CC 01 7C C7 77 CF 91 27 CC 01 T....'|.w..'.
0A9B8C70  20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A9B8C80  00 00 00 00 05 01 00 00 00 00 00 00 00 00 00 00 .....
0A9B8C90  00 00 00 00 00 00 00 00 30 00 00 00 70 00 00 00 .....0...p...
0A9B8CA0  00 00 00 00 00 00 02 00 58 00 00 00 18 00 01 00 .....X.....
0A9B8CB0  05 00 00 00 00 00 05 00 7C C7 77 CF 91 27 CC 01 .....|.w..'.
0A9B8CC0  7C C7 77 CF 91 27 CC 01 7C C7 77 CF 91 27 CC 01 |.w..'.|.w..'.
0A9B8CD0  7C C7 77 CF 91 27 CC 01 00 00 00 00 00 00 00 00 |.w..'.
0A9B8CE0  00 00 00 00 00 00 00 00 20 00 00 00 00 00 00 00 .....
0A9B8CF0  0B 03 70 00 61 00 74 00 74 00 65 00 72 00 6E 00 ..p.a.t.t.e.r.n.
0A9B8D00  2E 00 74 00 78 00 74 00 80 00 00 00 50 00 00 00 ..t.x.t....P...
0A9B8D10  01 00 00 00 00 00 03 00 00 00 00 00 00 00 00 00 .....
0A9B8D20  33 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 .....
0A9B8D30  00 40 03 00 00 00 00 00 52 34 03 00 00 00 00 00 .@.....R4.....
0A9B8D40  52 34 03 00 00 00 00 00 31 09 80 FB 02 21 2B B3 R4.....1....!+.
0A9B8D50  D5 00 01 00 00 30 C5 87 FF FF FF FF 82 79 47 11 .....0.....yG.
0A9B8D60  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A9B8D70  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A9B8D80  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A9B8D90  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A9B8DA0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A9B8DB0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A9B8DC0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A9B8DD0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A9B8DE0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A9B8DF0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 .....
--- sdj                --0xA9B8DF0/0x20000000-----

```

you will find here an entry of a file called pattern.txt. This file is stored on two cluster chains described as 310980FB02, and 212BB3D5. Let us start to interpret them. First obviously the first one.

The 0x31 means three bytes gives the first cluster of this chain, and one byte give us the length of this chain.

First the length of the chain is stored, so it is 0x09 now

Then the first cluster of the chain, what is 0x02FB80 now, what means $0x02FB80 * 0x1000 = 0x2FB80000$

Let us try to find this position. Again we should use the previous three steps

- Calculate, in which stripe the data is.

- Calculate within that stripe on which disk the data is
- Calculate on that disk within that stripe where exactly the data starts

We get the stripe number as: $\text{INT}(0x2FB80000 / (3 * 0x10000)) = 0x0FE8$

The disk number is $0x2FB80000 - 0x0FE8 * 3 * 0x10000 = 0$. $\text{INT}(0 / 0x10000) = 0$ it means we will find this data on disk 1 what is /dev/sdi for me.

Offset within the stripe is $0 - 0 * 0x10000 = 0$ so the data will start immediately at the beginning of this stripe.

The length of the cluster chain is 0x09 what means $0x09 * 0x1000 = 0x9000$ bytes what is less than the stripe size, so we do not have to take care to the stripe border.

Now find the position on the disk /dev/sdi. It is the stripe number 0x0FE8 multiplied by the stripe size 0x10000 plus the offset of the logical disk what is also 0x10000. $0x0FE8 * 0x10000 + 0x10000 = 0x0FE90000$ and the end of this chain will be at 0x0FE98FFF. Check it by jumping to these positions:

```

root@chfivBox: ~
0FE90000  41 61 30 41  61 31 41 61  32 41 61 33  41 61 34 41  Aa0Aa1Aa2Aa3Aa4A
0FE90010  61 35 41 61  36 41 61 37  41 61 38 41  61 39 41 62  a5Aa6Aa7Aa8Aa9Ab
0FE90020  30 41 62 31  41 62 32 41  62 33 41 62  34 41 62 35  0Ab1Ab2Ab3Ab4Ab5
0FE90030  41 62 36 41  62 37 41 62  38 41 62 39  41 63 30 41  Ab6Ab7Ab8Ab9Ac0A
0FE90040  63 31 41 63  32 41 63 33  41 63 34 41  63 35 41 63  c1Ac2Ac3Ac4Ac5Ac
0FE90050  36 41 63 37  41 63 38 41  63 39 41 64  30 41 64 31  6Ac7Ac8Ac9Ad0Ad1
0FE90060  41 64 32 41  64 33 41 64  34 41 64 35  41 64 36 41  Ad2Ad3Ad4Ad5Ad6A
0FE90070  64 37 41 64  38 41 64 39  41 65 30 41  65 31 41 65  d7Ad8Ad9Ae0Ae1Ae.
0FE90080  32 41 65 33  41 65 34 41  65 35 41 65  36 41 65 37  2Ae3Ae4Ae5Ae6Ae7
0FE90090  41 65 38 41  65 39 41 66  30 41 66 31  41 66 32 41  Ae8Ae9Af0Af1Af2A
0FE900A0  66 33 41 66  34 41 66 35  41 66 36 41  66 37 41 66  f3Af4Af5Af6Af7Af
0FE900B0  38 41 66 39  41 67 30 41  67 31 41 67  32 41 67 33  8Af9Ag0Ag1Ag2Ag3.
0FE900C0  41 67 34 41  67 35 41 67  36 41 67 37  41 67 38 41  Ag4Ag5Ag6Ag7Ag8A
0FE900D0  67 39 41 68  30 41 68 31  41 68 32 41  68 33 41 68  g9Ah0Ah1Ah2Ah3Ah
0FE900E0  34 41 68 35  41 68 36 41  68 37 41 68  38 41 68 39  4Ah5Ah6Ah7Ah8Ah9
0FE900F0  41 69 30 41  69 31 41 69  32 41 69 33  41 69 34 41  Ai0Ai1Ai2Ai3Ai4A
0FE90100  69 35 41 69  36 41 69 37  41 69 38 41  69 39 41 6A  i5Ai6Ai7Ai8Ai9Aj
0FE90110  30 41 6A 31  41 6A 32 41  6A 33 41 6A  34 41 6A 35  0Aj1Aj2Aj3Aj4Aj5
0FE90120  41 6A 36 41  6A 37 41 6A  38 41 6A 39  41 6B 30 41  Aj6Aj7Aj8Aj9Ak0A
0FE90130  6B 31 41 6B  32 41 6B 33  41 6B 34 41  6B 35 41 6B  k1Ak2Ak3Ak4Ak5Ak
0FE90140  36 41 6B 37  41 6B 38 41  6B 39 41 6C  30 41 6C 31  6Ak7Ak8Ak9Al0Al1.
0FE90150  41 6C 32 41  6C 33 41 6C  34 41 6C 35  41 6C 36 41  Al2Al3Al4Al5Al6A
0FE90160  6C 37 41 6C  38 41 6C 39  41 6D 30 41  6D 31 41 6D  l7Al8Al9Am0Am1Am
0FE90170  32 41 6D 33  41 6D 34 41  6D 35 41 6D  36 41 6D 37  2Am3Am4Am5Am6Am7
0FE90180  41 6D 38 41  6D 39 41 6E  30 41 6E 31  41 6E 32 41  Am8Am9An0An1An2A.
0FE90190  6E 33 41 6E  34 41 6E 35  41 6E 36 41  6E 37 41 6E  n3An4An5An6An7An
0FE901A0  38 41 6E 39  41 6F 30 41  6F 31 41 6F  32 41 6F 33  8An9Ao0Ao1Ao2Ao3
0FE901B0  41 6F 34 41  6F 35 41 6F  36 41 6F 37  41 6F 38 41  Ao4Ao5Ao6Ao7Ao8A.
0FE901C0  6F 39 41 70  30 41 70 31  41 70 32 41  70 33 41 70  o9Ap0Ap1Ap2Ap3Ap
0FE901D0  34 41 70 35  41 70 36 41  70 37 41 70  38 41 70 39  4Ap5Ap6Ap7Ap8Ap9.
0FE901E0  41 71 30 41  71 31 41 71  32 41 71 33  41 71 34 41  Aq0Aq1Aq2Aq3Aq4A
0FE901F0  71 35 41 71  36 41 71 37  41 71 38 41  71 39 41 72  q5Aq6Aq7Aq8Aq9Ar
--- sdi          --0xFE901F0/0x20000000-----

```



```

root@chfIVBox: ~
0FE98EF0  78 37 55 78 38 55 78 39 55 79 30 55 79 31 55 79 x7Ux8Ux9Uy0Uy1Uy
0FE98F00  32 55 79 33 55 79 34 55 79 35 55 79 36 55 79 37 2Uy3Uy4Uy5Uy6Uy7
0FE98F10  55 79 38 55 79 39 55 7A 30 55 7A 31 55 7A 32 55 Uy8Uy9Uz0Uz1Uz2U
0FE98F20  7A 33 55 7A 34 55 7A 35 55 7A 36 55 7A 37 55 7A z3Uz4Uz5Uz6Uz7Uz
0FE98F30  38 55 7A 39 56 61 30 56 61 31 56 61 32 56 61 33 8Uz9Va0Va1Va2Va3
0FE98F40  56 61 34 56 61 35 56 61 36 56 61 37 56 61 38 56 Va4Va5Va6Va7Va8V
0FE98F50  61 39 56 62 30 56 62 31 56 62 32 56 62 33 56 62 a9Vb0Vb1Vb2Vb3Vb
0FE98F60  34 56 62 35 56 62 36 56 62 37 56 62 38 56 62 39 4Vb5Vb6Vb7Vb8Vb9.
0FE98F70  56 63 30 56 63 31 56 63 32 56 63 33 56 63 34 56 Vc0Vc1Vc2Vc3Vc4V
0FE98F80  63 35 56 63 36 56 63 37 56 63 38 56 63 39 56 64 c5Vc6Vc7Vc8Vc9Vd
0FE98F90  30 56 64 31 56 64 32 56 64 33 56 64 34 56 64 35 0Vd1Vd2Vd3Vd4Vd5
0FE98FA0  56 64 36 56 64 37 56 64 38 56 64 39 56 65 30 56 Vd6Vd7Vd8Vd9Ve0V.
0FE98FB0  65 31 56 65 32 56 65 33 56 65 34 56 65 35 56 65 e1Ve2Ve3Ve4Ve5Ve
0FE98FC0  36 56 65 37 56 65 38 56 65 39 56 66 30 56 66 31 6Ve7Ve8Ve9Vf0Vf1
0FE98FD0  56 66 32 56 66 33 56 66 34 56 66 35 56 66 36 56 Vf2Vf3Vf4Vf5Vf6V
0FE98FE0  66 37 56 66 38 56 66 39 56 67 30 56 67 31 56 67 f7Vf8Vf9Vg0Vg1Vg
0FE98FF0  32 56 67 33 56 67 34 56 67 35 56 67 36 56 67 37 2Vg3Vg4Vg5Vg6Vg7
0FE99000  49 4E 44 58 28 00 09 00 CF 37 40 00 00 00 00 00 INDX(....7@.....
0FE99010  00 00 00 00 00 00 00 00 40 00 00 00 80 06 00 00 .....@.....
0FE99020  E8 0F 00 00 00 00 00 00 15 00 05 00 05 00 CC 01 .....
0FE99030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0FE99040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0FE99050  00 00 00 00 00 00 00 00 04 00 00 00 00 00 04 00 .....
0FE99060  68 00 52 00 00 00 00 00 05 00 00 00 00 00 05 00 h.R.....
0FE99070  74 7E 5F DB 8F 27 CC 01 74 7E 5F DB 8F 27 CC 01 t~_..'.t~_..'.
0FE99080  74 7E 5F DB 8F 27 CC 01 74 7E 5F DB 8F 27 CC 01 t~_..'.t~_..'.
0FE99090  00 90 00 00 00 00 00 00 A0 8C 00 00 00 00 00 00 .....
0FE990A0  06 00 00 00 00 00 00 00 08 03 24 00 41 00 74 00 .....$.A.t..
0FE990B0  74 00 72 00 44 00 65 00 66 00 00 00 00 00 00 00 t.r.D.e.f.....
0FE990C0  08 00 00 00 00 00 08 00 68 00 52 00 00 00 00 00 .....h.R.....
0FE990D0  05 00 00 00 00 00 05 00 74 7E 5F DB 8F 27 CC 01 .....t~_..'.
0FE990E0  74 7E 5F DB 8F 27 CC 01 74 7E 5F DB 8F 27 CC 01 t~_..'.t~_..'.
--- sdi          --0xFE98EF0/0x20000000-----

```

to get the data back we can use a simple dd command:

```
dd if=/dev/sdi bs=1 skip=266928128 count=36864 > ./pattern.txt
```

Now reconstruct the second cluster chain, described as 212BB3D5, what should be interpreted as follows:

0x21 means, the start cluster is stored on two bytes, and the chain length is stored on 1 byte. First the chain length is stored, what is 2B now
 The next two bytes 0xD5B3 is the start cluster of the chain. **But it is relative number to the previous one not absolute! And also do not forget, it is a signed number!** So it is -2A4D.

Calculate the first cluster of the chain. First calculate absolute position from the relative one. To do it add to the previous value this one: $0x02FB80 + (-2A4D) = 0x2D133$. It is given in clusters so the value in bytes: $0x2D133 * 0x1000 = 0x2D133000$ Then we can calculate on the already well known way.

We get the stripe number as: $\text{INT}(0x2D133000 / (3 * 0x10000)) = 0x0F06$
 The disk number is $0x2D133000 - 0x0F06 * 3 * 0x10000 = 0x13000$. $\text{INT}(0x13000 / 0x10000) =$

0x1 it means we will find this data on disk 2 what is /dev/sdh for me.

Offset within the stripe is $0x13000 - 1 * 0x10000 = 0x3000$ so the data will start at the offset 0x3000 from the start of the stripe.

The length of the cluster chain is 0x2B what means $0x2B * 0x1000 = 0x02B000$ bytes what is more than the stripe size, so it will continue on the next disk. If we divide it by the stripe size what is 0x10000 we get two stripe will be fully utilized, and a third one will be started, but not fully utilized.

Now find the position on the disk /dev/sdh. It is the stripe number 0x0F06 multiplied by the stripe size 0x10000 plus the offset of the logical disk what is also 0x10000 plus the offset from the beginning of the stripe, what is 0x3000. $0x0F06 * 0x10000 + 0x10000 + 0x3000 = 0x0F073000$.

If we jump there by pressing ctrl-g then type 0x0F073000 we really find the continuation of the file (because it contains a pattern it is easy to check now, the end of the previous part was Vg7 and this one starts with Vg8 what seems to be really a continuation)

```
root@chfIVBox: ~
0F073000 56 67 38 56 67 39 56 68 30 56 68 31 56 68 32 56 Vg8Vg9Vh0Vh1Vh2V
0F073010 68 33 56 68 34 56 68 35 56 68 36 56 68 37 56 68 h3Vh4Vh5Vh6Vh7Vh
0F073020 38 56 68 39 56 69 30 56 69 31 56 69 32 56 69 33 8Vh9Vi0Vi1Vi2Vi3
0F073030 56 69 34 56 69 35 56 69 36 56 69 37 56 69 38 56 Vi4Vi5Vi6Vi7Vi8V
0F073040 69 39 56 6A 30 56 6A 31 56 6A 32 56 6A 33 56 6A i9Vj0Vj1Vj2Vj3Vj
0F073050 34 56 6A 35 56 6A 36 56 6A 37 56 6A 38 56 6A 39 4Vj5Vj6Vj7Vj8Vj9
0F073060 56 6B 30 56 6B 31 56 6B 32 56 6B 33 56 6B 34 56 Vk0Vk1Vk2Vk3Vk4V
0F073070 6B 35 56 6B 36 56 6B 37 56 6B 38 56 6B 39 56 6C k5Vk6Vk7Vk8Vk9Vl.
0F073080 30 56 6C 31 56 6C 32 56 6C 33 56 6C 34 56 6C 35 0Vl1Vl2Vl3Vl4Vl5
0F073090 56 6C 36 56 6C 37 56 6C 38 56 6C 39 56 6D 30 56 Vl6Vl7Vl8Vl9Vm0V.
0F0730A0 6D 31 56 6D 32 56 6D 33 56 6D 34 56 6D 35 56 6D m1Vm2Vm3Vm4Vm5Vm
0F0730B0 36 56 6D 37 56 6D 38 56 6D 39 56 6E 30 56 6E 31 6Vm7Vm8Vm9Vn0Vn1
0F0730C0 56 6E 32 56 6E 33 56 6E 34 56 6E 35 56 6E 36 56 Vn2Vn3Vn4Vn5Vn6V.
0F0730D0 6E 37 56 6E 38 56 6E 39 56 6F 30 56 6F 31 56 6F n7Vn8Vn9Vo0Vo1Vo
0F0730E0 32 56 6F 33 56 6F 34 56 6F 35 56 6F 36 56 6F 37 2Vo3Vo4Vo5Vo6Vo7
0F0730F0 56 6F 38 56 6F 39 56 70 30 56 70 31 56 70 32 56 Vo8Vo9Vp0Vp1Vp2V
0F073100 70 33 56 70 34 56 70 35 56 70 36 56 70 37 56 70 p3Vp4Vp5Vp6Vp7Vp
0F073110 38 56 70 39 56 71 30 56 71 31 56 71 32 56 71 33 8Vp9Vq0Vq1Vq2Vq3
0F073120 56 71 34 56 71 35 56 71 36 56 71 37 56 71 38 56 Vq4Vq5Vq6Vq7Vq8V.
0F073130 71 39 56 72 30 56 72 31 56 72 32 56 72 33 56 72 q9Vr0Vr1Vr2Vr3Vr
0F073140 34 56 72 35 56 72 36 56 72 37 56 72 38 56 72 39 4Vr5Vr6Vr7Vr8Vr9
0F073150 56 73 30 56 73 31 56 73 32 56 73 33 56 73 34 56 Vs0Vs1Vs2Vs3Vs4V.
0F073160 73 35 56 73 36 56 73 37 56 73 38 56 73 39 56 74 s5Vs6Vs7Vs8Vs9Vt
0F073170 30 56 74 31 56 74 32 56 74 33 56 74 34 56 74 35 0Vt1Vt2Vt3Vt4Vt5.
0F073180 56 74 36 56 74 37 56 74 38 56 74 39 56 75 30 56 Vt6Vt7Vt8Vt9Vu0V
0F073190 75 31 56 75 32 56 75 33 56 75 34 56 75 35 56 75 u1Vu2Vu3Vu4Vu5Vu
0F0731A0 36 56 75 37 56 75 38 56 75 39 56 76 30 56 76 31 6Vu7Vu8Vu9Vv0Vv1.
0F0731B0 56 76 32 56 76 33 56 76 34 56 76 35 56 76 36 56 Vv2Vv3Vv4Vv5Vv6V
0F0731C0 76 37 56 76 38 56 76 39 56 77 30 56 77 31 56 77 v7Vv8Vv9Vw0Vw1Vw.
0F0731D0 32 56 77 33 56 77 34 56 77 35 56 77 36 56 77 37 2Vw3Vw4Vw5Vw6Vw7
0F0731E0 56 77 38 56 77 39 56 78 30 56 78 31 56 78 32 56 Vw8Vw9Vx0Vx1Vx2V.
0F0731F0 78 33 56 78 34 56 78 35 56 78 36 56 78 37 56 78 x3Vx4Vx5Vx6Vx7Vx
--- sdh -----0x0F073000/0x20000000-----
```

As we know it occupies this whole stripe so we can extract this data by the following dd command:

```
dd if=/dev/sdh bs=1 skip=252129280 count=53248 >> ./pattern.txt
```

It was the disk 2 so the file will continue on the same stripe in disk 3, and it will start from the beginning of the cluster so the position will be /dev/sdj stripe 0x0F06 what means offset 0x0F06 * 0x10000 + 0x10000 = 0x0F070000. We can jump there by pressing ctrl-g then type this value.

```

root@chfIVBox: ~
0F070000  6E 37 4C 6E 38 4C 6E 39 4C 6F 30 4C 6F 31 4C 6F n7Ln8Ln9Lo0Lo1Lo
0F070010  32 4C 6F 33 4C 6F 34 4C 6F 35 4C 6F 36 4C 6F 37 2Lo3Lo4Lo5Lo6Lo7
0F070020  4C 6F 38 4C 6F 39 4C 70 30 4C 70 31 4C 70 32 4C Lo8Lo9Lp0Lp1Lp2L
0F070030  70 33 4C 70 34 4C 70 35 4C 70 36 4C 70 37 4C 70 p3Lp4Lp5Lp6Lp7Lp
0F070040  38 4C 70 39 4C 71 30 4C 71 31 4C 71 32 4C 71 33 8Lp9Lq0Lq1Lq2Lq3
0F070050  4C 71 34 4C 71 35 4C 71 36 4C 71 37 4C 71 38 4C Lq4Lq5Lq6Lq7Lq8L
0F070060  71 39 4C 72 30 4C 72 31 4C 72 32 4C 72 33 4C 72 q9Lr0Lr1Lr2Lr3Lr
0F070070  34 4C 72 35 4C 72 36 4C 72 37 4C 72 38 4C 72 39 4Lr5Lr6Lr7Lr8Lr9.
0F070080  4C 73 30 4C 73 31 4C 73 32 4C 73 33 4C 73 34 4C Ls0Ls1Ls2Ls3Ls4L
0F070090  73 35 4C 73 36 4C 73 37 4C 73 38 4C 73 39 4C 74 s5Ls6Ls7Ls8Ls9Lt
0F0700A0  30 4C 74 31 4C 74 32 4C 74 33 4C 74 34 4C 74 35 0Lt1Lt2Lt3Lt4Lt5
0F0700B0  4C 74 36 4C 74 37 4C 74 38 4C 74 39 4C 75 30 4C Lt6Lt7Lt8Lt9Lu0L
0F0700C0  75 31 4C 75 32 4C 75 33 4C 75 34 4C 75 35 4C 75 u1Lu2Lu3Lu4Lu5Lu.
0F0700D0  36 4C 75 37 4C 75 38 4C 75 39 4C 76 30 4C 76 31 6Lu7Lu8Lu9Lv0Lv1
0F0700E0  4C 76 32 4C 76 33 4C 76 34 4C 76 35 4C 76 36 4C Lv2Lv3Lv4Lv5Lv6L
0F0700F0  76 37 4C 76 38 4C 76 39 4C 77 30 4C 77 31 4C 77 v7Lv8Lv9Lw0Lw1Lw
0F070100  32 4C 77 33 4C 77 34 4C 77 35 4C 77 36 4C 77 37 2Lw3Lw4Lw5Lw6Lw7
0F070110  4C 77 38 4C 77 39 4C 78 30 4C 78 31 4C 78 32 4C Lw8Lw9Lx0Lx1Lx2L
0F070120  78 33 4C 78 34 4C 78 35 4C 78 36 4C 78 37 4C 78 x3Lx4Lx5Lx6Lx7Lx
0F070130  38 4C 78 39 4C 79 30 4C 79 31 4C 79 32 4C 79 33 8Lx9Ly0Ly1Ly2Ly3
0F070140  4C 79 34 4C 79 35 4C 79 36 4C 79 37 4C 79 38 4C Ly4Ly5Ly6Ly7Ly8L
0F070150  79 39 4C 7A 30 4C 7A 31 4C 7A 32 4C 7A 33 4C 7A y9Lz0Lz1Lz2Lz3Lz
0F070160  34 4C 7A 35 4C 7A 36 4C 7A 37 4C 7A 38 4C 7A 39 4Lz5Lz6Lz7Lz8Lz9
0F070170  4D 61 30 4D 61 31 4D 61 32 4D 61 33 4D 61 34 4D Ma0Ma1Ma2Ma3Ma4M
0F070180  61 35 4D 61 36 4D 61 37 4D 61 38 4D 61 39 4D 62 a5Ma6Ma7Ma8Ma9Mb.
0F070190  30 4D 62 31 4D 62 32 4D 62 33 4D 62 34 4D 62 35 0Mb1Mb2Mb3Mb4Mb5
0F0701A0  4D 62 36 4D 62 37 4D 62 38 4D 62 39 4D 63 30 4D Mb6Mb7Mb8Mb9Mc0M
0F0701B0  63 31 4D 63 32 4D 63 33 4D 63 34 4D 63 35 4D 63 c1Mc2Mc3Mc4Mc5Mc.
0F0701C0  36 4D 63 37 4D 63 38 4D 63 39 4D 64 30 4D 64 31 6Mc7Mc8Mc9Md0Md1
0F0701D0  4D 64 32 4D 64 33 4D 64 34 4D 64 35 4D 64 36 4D Md2Md3Md4Md5Md6M.
0F0701E0  64 37 4D 64 38 4D 64 39 4D 65 30 4D 65 31 4D 65 d7Md8Md9Me0Me1Me
0F0701F0  32 4D 65 33 4D 65 34 4D 65 35 4D 65 36 4D 65 37 2Me3Me4Me5Me6Me7
--- sdj          --0xF070000/0x20000000-----

```

Again it will occupy the whole stripe on this disk so we can extract it by the next dd command

```
dd if=/dev/sdh bs=1 skip=252116992 count=65536 >> ./pattern.txt
```

Because it is the disk 3 the data will continue on disk 1 (/dev/sdi), at the 0x0F06 + 1 stripe. And it will occupy 0x2B - 0x20 + 0x03 = 0x0E clusters, means 0xE000 bytes.

The offset on this disk will be 0x0F07 * 0x10000 + 0x10000 = 0x0F080000 and the end of it will be at 0x0F080000 + 0x0E * 0x1000 = 0x0F08E000. But it is calculated from the clusters, so the real end of the file might be a bit earlier. The real end position: filesize (0x033452) - the data already found. 0x033452 - 0x9000 - 0xD000 - 0x10000 = 0xD452. The real end of the file will be instead of 0x0F08E000 at 0x0F08D452. Let us check these informations by jumping to those positions

```

root@chfivBox: ~
0F080000 32 52 6F 33 52 6F 34 52 6F 35 52 6F 36 52 6F 37 2Ro3Ro4Ro5Ro6Ro7
0F080010 52 6F 38 52 6F 39 52 70 30 52 70 31 52 70 32 52 Ro8Ro9Rp0Rp1Rp2R
0F080020 70 33 52 70 34 52 70 35 52 70 36 52 70 37 52 70 p3Rp4Rp5Rp6Rp7Rp
0F080030 38 52 70 39 52 71 30 52 71 31 52 71 32 52 71 33 8Rp9Rq0Rq1Rq2Rq3
0F080040 52 71 34 52 71 35 52 71 36 52 71 37 52 71 38 52 Rq4Rq5Rq6Rq7Rq8R
0F080050 71 39 52 72 30 52 72 31 52 72 32 52 72 33 52 72 q9Rr0Rr1Rr2Rr3Rr
0F080060 34 52 72 35 52 72 36 52 72 37 52 72 38 52 72 39 4Rr5Rr6Rr7Rr8Rr9
0F080070 52 73 30 52 73 31 52 73 32 52 73 33 52 73 34 52 Rs0Rs1Rs2Rs3Rs4R.
0F080080 73 35 52 73 36 52 73 37 52 73 38 52 73 39 52 74 s5Rs6Rs7Rs8Rs9Rt
0F080090 30 52 74 31 52 74 32 52 74 33 52 74 34 52 74 35 0Rt1Rt2Rt3Rt4Rt5
0F0800A0 52 74 36 52 74 37 52 74 38 52 74 39 52 75 30 52 Rt6Rt7Rt8Rt9Ru0R
0F0800B0 75 31 52 75 32 52 75 33 52 75 34 52 75 35 52 75 u1Ru2Ru3Ru4Ru5Ru.
0F0800C0 36 52 75 37 52 75 38 52 75 39 52 76 30 52 76 31 6Ru7Ru8Ru9Rv0Rv1
0F0800D0 52 76 32 52 76 33 52 76 34 52 76 35 52 76 36 52 Rv2Rv3Rv4Rv5Rv6R
0F0800E0 76 37 52 76 38 52 76 39 52 77 30 52 77 31 52 77 v7Rv8Rv9Rw0Rw1Rw
0F0800F0 32 52 77 33 52 77 34 52 77 35 52 77 36 52 77 37 2Rw3Rw4Rw5Rw6Rw7
0F080100 52 77 38 52 77 39 52 78 30 52 78 31 52 78 32 52 Rw8Rw9Rx0Rx1Rx2R
0F080110 78 33 52 78 34 52 78 35 52 78 36 52 78 37 52 78 x3Rx4Rx5Rx6Rx7Rx
0F080120 38 52 78 39 52 79 30 52 79 31 52 79 32 52 79 33 8Rx9Ry0Ry1Ry2Ry3
0F080130 52 79 34 52 79 35 52 79 36 52 79 37 52 79 38 52 Ry4Ry5Ry6Ry7Ry8R
0F080140 79 39 52 7A 30 52 7A 31 52 7A 32 52 7A 33 52 7A y9Rz0Rz1Rz2Rz3Rz.
0F080150 34 52 7A 35 52 7A 36 52 7A 37 52 7A 38 52 7A 39 4Rz5Rz6Rz7Rz8Rz9
0F080160 53 61 30 53 61 31 53 61 32 53 61 33 53 61 34 53 Sa0Sa1Sa2Sa3Sa4S
0F080170 61 35 53 61 36 53 61 37 53 61 38 53 61 39 53 62 a5Sa6Sa7Sa8Sa9Sb
0F080180 30 53 62 31 53 62 32 53 62 33 53 62 34 53 62 35 0Sb1Sb2Sb3Sb4Sb5.
0F080190 53 62 36 53 62 37 53 62 38 53 62 39 53 63 30 53 Sb6Sb7Sb8Sb9Sc0S
0F0801A0 63 31 53 63 32 53 63 33 53 63 34 53 63 35 53 63 c1Sc2Sc3Sc4Sc5Sc
0F0801B0 36 53 63 37 53 63 38 53 63 39 53 64 30 53 64 31 6Sc7Sc8Sc9Sd0Sd1.
0F0801C0 53 64 32 53 64 33 53 64 34 53 64 35 53 64 36 53 Sd2Sd3Sd4Sd5Sd6S
0F0801D0 64 37 53 64 38 53 64 39 53 65 30 53 65 31 53 65 d7Sd8Sd9Se0Se1Se.
0F0801E0 32 53 65 33 53 65 34 53 65 35 53 65 36 53 65 37 2Se3Se4Se5Se6Se7
0F0801F0 53 65 38 53 65 39 53 66 30 53 66 31 53 66 32 53 Se8Se9Sf0Sf1Sf2S
--- sdi --0xF080000/0x20000000-----

```



```

root@chfIVBox: ~
0F08D380  30 49 7A 31 49 7A 32 49 7A 33 49 7A 34 49 7A 35 0Iz1Iz2Iz3Iz4Iz5
0F08D390  49 7A 36 49 7A 37 49 7A 38 49 7A 39 4A 61 30 4A Iz6Iz7Iz8Iz9Ja0J
0F08D3A0  61 31 4A 61 32 4A 61 33 4A 61 34 4A 61 35 4A 61 a1Ja2Ja3Ja4Ja5Ja
0F08D3B0  36 4A 61 37 4A 61 38 4A 61 39 4A 62 30 4A 62 31 6Ja7Ja8Ja9Jb0Jb1
0F08D3C0  4A 62 32 4A 62 33 4A 62 34 4A 62 35 4A 62 36 4A Jb2Jb3Jb4Jb5Jb6J
0F08D3D0  62 37 4A 62 38 4A 62 39 4A 63 30 4A 63 31 4A 63 b7Jb8Jb9Jc0Jc1Jc
0F08D3E0  32 4A 63 33 4A 63 34 4A 63 35 4A 63 36 4A 63 37 2Jc3Jc4Jc5Jc6Jc7
0F08D3F0  4A 63 38 4A 63 39 4A 64 30 4A 64 31 4A 64 32 4A Jc8Jc9Jd0Jd1Jd2J
0F08D400  64 33 4A 64 34 4A 64 35 4A 64 36 4A 64 37 4A 64 d3Jd4Jd5Jd6Jd7Jd
0F08D410  38 4A 64 39 4A 65 30 4A 65 31 4A 65 32 4A 65 33 8Jd9Je0Je1Je2Je3
0F08D420  4A 65 34 4A 65 35 4A 65 36 4A 65 37 4A 65 38 4A Je4Je5Je6Je7Je8J
0F08D430  65 39 4A 66 30 4A 66 31 4A 66 32 4A 66 33 4A 66 e9Jf0Jf1Jf2Jf3Jf
0F08D440  34 4A 66 35 4A 66 36 4A 66 37 4A 66 38 4A 66 39 4Jf5Jf6Jf7Jf8Jf9
0F08D450  0D 0A 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0F08D460  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0F08D470  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0F08D480  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0F08D490  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0F08D4A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0F08D4B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0F08D4C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0F08D4D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0F08D4E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0F08D4F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0F08D500  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0F08D510  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0F08D520  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0F08D530  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0F08D540  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0F08D550  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0F08D560  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0F08D570  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
--- sdi --0xF08D380/0x20000000-----

```

this last part can be extracted by the next dd command:

```
dd if=/dev/sdi bs=1 skip=252182528 count=54354 >> ./pattern.txt
```

Microsoft dynamic disk RAID5

Now we have three disks, evidenceRAID5-disk0.vhd (/dev/sdk), evidenceRAID5-disk1.vhd (/dev/sdl), and evidenceRAID5-disk2.vhd (/dev/sdm) and we created a RAID5 volume across these three disks. Let us try to reassemble a file on this disk.

In this case the disks are divided again to stripes. First we should know the stripe size, and the order of the disks as a basic information.

Again if we check the MBR

```
root@chflVBox: ~
00000000  33 C0 8E D0 BC 00 7C FB 50 07 50 1F FC BE 1B 7C 3.....|.P.P....|
00000010  BF 1B 06 50 57 B9 E5 01 F3 A4 CB BD BE 07 B1 04 ...PW.....
00000020  38 6E 00 7C 09 75 13 83 C5 10 E2 F4 CD 18 8B F5 8n.|.u.....
00000030  83 C6 10 49 74 19 38 2C 74 F6 A0 B5 07 B4 07 8B ...It.8,t.....
00000040  F0 AC 3C 00 74 FC BB 07 00 B4 0E CD 10 EB F2 88 ..<.t.....
00000050  4E 10 E8 46 00 73 2A FE 46 10 80 7E 04 0B 74 0B N..F.s*.F..~.t.
00000060  80 7E 04 0C 74 05 A0 B6 07 75 D2 80 46 02 06 83 .~.t....u..F...
00000070  46 08 06 83 56 0A 00 E8 21 00 73 05 A0 B6 07 EB F...V...!.s....
00000080  BC 81 3E FE 7D 55 AA 74 0B 80 7E 10 00 74 C8 A0 ..>.]U.t..~.t..
00000090  B7 07 EB A9 8B FC 1E 57 8B F5 CB BF 05 00 8A 56 .....W.....V
000000A0  00 B4 08 CD 13 72 23 8A C1 24 3F 98 8A DE 8A FC .....r#..$?....
000000B0  43 F7 E3 8B D1 86 D6 B1 06 D2 EE 42 F7 E2 39 56 C.....B..9V
000000C0  0A 77 23 72 05 39 46 08 73 1C B8 01 02 BB 00 7C .w#r.9F.s.....|
000000D0  8B 4E 02 8B 56 00 CD 13 73 51 4F 74 4E 32 E4 8A .N..V...sQ0tN2..
000000E0  56 00 CD 13 EB E4 8A 56 00 60 BB AA 55 B4 41 CD V.....V.`..U.A.
000000F0  13 72 36 81 FB 55 AA 75 30 F6 C1 01 74 2B 61 60 .r6..U.u0...t+a`
00000100  6A 00 6A 00 FF 76 0A FF 76 08 6A 00 68 00 7C 6A j.j..v..v.j.h.|j
00000110  01 6A 10 B4 42 8B F4 CD 13 61 61 73 0E 4F 74 0B .j..B....aas.Ot.
00000120  32 E4 8A 56 00 CD 13 EB D6 61 F9 C3 49 6E 76 61 2..V.....a..Inva
00000130  6C 69 64 20 70 61 72 74 69 74 69 6F 6E 20 74 61 lid partition ta
00000140  62 6C 65 00 45 72 72 6F 72 20 6C 6F 61 64 69 6E ble.Error loadin
00000150  67 20 6F 70 65 72 61 74 69 6E 67 20 73 79 73 74 g operating syst
00000160  65 6D 00 4D 69 73 73 69 6E 67 20 6F 70 65 72 61 em.Missing opera
00000170  74 69 6E 67 20 73 79 73 74 65 6D 00 00 00 00 00 ting system.....
00000180  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000190  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000001A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000001B0  00 00 00 00 00 2C 44 63 9D A9 0E ED 00 00 00 01 .....,Dc.....
000001C0  01 00 42 FE 3F 40 3F 00 00 00 C2 EE 0F 00 00 00 ..B.?@?.....
000001D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000001E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000001F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA .....U.
--- sdl                --0x0/0x20000000-----
```

it define the start of first partition at 0x3F what means $0x3F * 0x200 = 0x7E00$, but there you will find nothing. Like in the previous case, we should check the dynamic disk parameters. Find the PRIVHEAD block at the beginning of the disk:


```

root@chfivBox: ~
000000C00  50 52 49 56 48 45 41 44 00 00 2F 52 00 02 00 0C PRIVHEAD../R....
000000C10  01 CC 23 BB 16 5C 8C C1 00 00 00 00 00 00 00 09 ..#..\.....
000000C20  00 00 00 00 00 00 07 FF 00 00 00 00 00 00 07 40 .....@
000000C30  30 39 65 36 34 64 31 33 2D 66 33 39 32 2D 34 34 09e64d13-f392-44
000000C40  33 61 2D 39 33 31 37 2D 37 63 38 34 30 33 61 63 3a-9317-7c8403ac
000000C50  65 34 62 39 00 00 00 00 00 00 00 00 00 00 00 00 e4b9.....
000000C60  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000C70  31 62 37 37 64 61 32 30 2D 63 37 31 37 2D 31 31 1b77da20-c717-11
000000C80  64 30 2D 61 35 62 65 2D 30 30 61 30 63 39 31 64 d0-a5be-00a0c91d
000000C90  62 37 33 63 00 00 00 00 00 00 00 00 00 00 00 00 b73c.....
000000CA0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000CB0  34 35 62 35 62 65 32 65 2D 38 66 66 38 2D 31 31 45b5be2e-8ff8-11
000000CC0  65 30 2D 38 34 30 66 2D 30 30 30 38 34 34 34 34 e0-840f-00084444
000000CD0  34 34 34 34 00 00 00 00 00 00 00 00 00 00 00 00 4444.....
000000CE0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000CF0  48 41 43 4B 2D 41 54 54 41 43 4B 45 52 2D 44 67 HACK-ATTACKER-Dg
000000D00  30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0.....
000000D10  00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000D20  00 00 3F 00 00 00 00 00 0F EE C2 00 00 00 00 00 00 ..?.....
000000D30  0F F7 00 00 00 00 00 00 00 08 00 00 00 00 00 00 .....
000000D40  00 00 02 00 00 00 00 00 00 07 FD 00 00 00 01 00 .....
000000D50  00 00 01 00 00 00 00 00 00 05 C9 00 00 00 00 00 .....
000000D60  00 00 E0 FE E7 87 4C 00 00 00 00 00 00 00 00 00 .....L.....
000000D70  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000D80  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000D90  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000DA0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000DB0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000DC0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000DD0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000DE0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000000DF0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
--- sdk          --0xC00/0x20000000-----

```

0x011B..0x0122 : The start of the logical disk is at 0x3F again this is not the start of the volume or partition, but the start of the logical disk, we will have to add this value to the start of the partition.

0x012B..0x0132 : The start of the configuration now it is 0x0FF700 what means $0x0FF700 * 0x200 = 0x1FEE0000$

Again a bit after this position starts the TOCBLOCK (0x1FEE0200), then the VMDB (0x1FEE2200), and the VBLK blocks from (0x1FEE2400).

We already know that, the stripe size can be found in the VBLK component descriptor 0x32. This block has the following content now

```

root@chfiVBox: ~
1FEE2700 56 42 4C 4B 00 00 00 0A 00 00 00 0F 00 00 00 01 VBLK.....
1FEE2710 00 00 10 32 00 00 00 34 02 04 0B 0A 56 6F 6C 75 ...2...4....Volu
1FEE2720 6D 65 31 2D 30 31 06 41 43 54 49 56 45 03 00 00 me1-01.ACTIVE...
1FEE2730 00 00 01 03 00 00 00 00 00 00 04 0D 00 00 00 00 .....
1FEE2740 00 00 00 00 02 04 0A 00 01 80 01 03 00 00 00 00 .....
1FEE2750 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FEE2760 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FEE2770 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FEE2780 56 42 4C 4B 00 00 00 0B 00 00 00 01 00 00 00 01 VBLK.....
1FEE2790 00 00 08 35 00 00 00 50 02 04 01 11 48 41 43 4B ...5...P....HACK

```

Here we can see that, there are 3 disks, and the stripe size is 0x80 blocks so $0x80 * 0x200 = 0x10000$ bytes.

To get the order of the disks we collect the GUID of every disk from the PRIVHEAD block.

```

root@chfiVBox: ~
00000C00 50 52 49 56 48 45 41 44 00 00 2F 52 00 02 00 0C PRIVHEAD../R....
00000C10 01 CC 23 BB 16 5C 8C C1 00 00 00 00 00 00 00 09 ..#..\.....
00000C20 00 00 00 00 00 00 07 FF 00 00 00 00 00 00 07 40 .....@
00000C30 30 39 65 36 34 64 31 33 2D 66 33 39 32 2D 34 34 09e64d13-f392-44
00000C40 33 61 2D 39 33 31 37 2D 37 63 38 34 30 33 61 63 3a-9317-7c8403ac
00000C50 65 34 62 39 00 00 00 00 00 00 00 00 00 00 00 00 e4b9.....
00000C60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
--- sdk --0xC00/0x20000000-----

```

```

root@chfiVBox: ~
00000C00 50 52 49 56 48 45 41 44 00 00 2F 70 00 02 00 0C PRIVHEAD../p....
00000C10 01 CC 23 BB 16 5F 9B 21 00 00 00 00 00 00 00 09 ..#.._!......
00000C20 00 00 00 00 00 00 07 FF 00 00 00 00 00 00 07 40 .....@
00000C30 63 36 34 37 65 61 31 65 2D 38 39 36 35 2D 34 36 c647ea1e-8965-46
00000C40 65 39 2D 38 65 64 38 2D 38 34 33 34 66 31 37 64 e9-8ed8-8434f17d
00000C50 66 31 63 65 00 00 00 00 00 00 00 00 00 00 00 00 f1ce.....
00000C60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
--- sdl --0xC00/0x20000000-----

```

```

root@chfiVBox: ~
00000C00 50 52 49 56 48 45 41 44 00 00 2F 12 00 02 00 0C PRIVHEAD../.....
00000C10 01 CC 23 BB 16 61 22 51 00 00 00 00 00 00 00 09 ..#..a"Q.....
00000C20 00 00 00 00 00 00 07 FF 00 00 00 00 00 00 07 40 .....@
00000C30 36 38 39 34 65 66 64 39 2D 30 61 64 32 2D 34 63 6894efd9-0ad2-4c
00000C40 36 64 2D 62 39 36 33 2D 33 66 63 63 39 37 30 32 6d-b963-3fcc9702
00000C50 63 30 36 31 00 00 00 00 00 00 00 00 00 00 00 00 c061.....
00000C60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
--- sdm --0xC00/0x20000000-----

```

```

/dev/sdk : 09E64D13-F392-443A-9317-7C8403ACE4B9
/dev/sdl : C647EA1E-8965-46E9-8ED8-8434F17DF1CE
/dev/sdm : 6894EFD9-0AD2-4C6D-B963-3FCC9702C061

```

we can pair them with the values from the VBLK disk descriptor (0x34)

```

root@chfIVBox: ~
1FEE2800 56 42 4C 4B 00 00 00 0C 00 00 00 06 00 00 00 02 VBLK.....
1FEE2810 00 00 00 34 00 00 00 9C 02 04 03 05 44 69 73 6B ...4.....Disk
1FEE2820 31 24 30 39 65 36 34 64 31 33 2D 66 33 39 32 2D 1$09e64d13-f392-
1FEE2830 34 34 33 61 2D 39 33 31 37 2D 37 63 38 34 30 33 443a-9317-7c8403
1FEE2840 61 63 65 34 62 39 61 49 44 45 5C 44 49 53 4B 56 ace4b9aIDE\DISKV
1FEE2850 42 4F 58 5F 48 41 52 44 44 49 53 4B 5F 5F 5F 5F BOX_HARDDISK____
1FEE2860 5F 5F 5F 5F 5F 5F 5F 5F 5F 5F 5F 5F 5F 5F 5F 5F
1FEE2870 5F 5F 5F 5F 5F 5F 5F 31 2E 30 5F 5F 5F 5F 5F 5C 1.0 \.
1FEE2880 56 42 4C 4B 00 00 00 0D 00 00 00 06 00 01 00 02 VBLK.....
1FEE2890 34 32 35 36 36 32 33 39 33 39 36 35 33 31 36 34 4256623939653164.
1FEE28A0 36 32 33 37 33 38 32 44 36 32 36 35 33 35 36 32 6237382D62653562
1FEE28B0 33 34 36 36 32 30 33 35 00 00 00 00 00 00 00 00 34662035.....
1FEE28C0 00 00 04 0B 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FEE28D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FEE28E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FEE28F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FEE2900 56 42 4C 4B 00 00 00 0E 00 00 00 07 00 00 00 02 VBLK.....
1FEE2910 00 00 00 34 00 00 00 9C 02 04 06 05 44 69 73 6B ...4.....Disk
1FEE2920 32 24 63 36 34 37 65 61 31 65 2D 38 39 36 35 2D 2$5c647ea1e-8965-
1FEE2930 34 36 65 39 2D 38 65 64 38 2D 38 34 33 34 66 31 46e9-8ed8-8434f1
1FEE2940 37 64 66 31 63 65 61 49 44 45 5C 44 49 53 4B 56 7df1ceaIDE\DISKV
1FEE2950 42 4F 58 5F 48 41 52 44 44 49 53 4B 5F 5F 5F 5F BOX_HARDDISK____
1FEE2960 5F 5F 5F 5F 5F 5F 5F 5F 5F 5F 5F 5F 5F 5F 5F 5F
1FEE2970 5F 5F 5F 5F 5F 5F 5F 31 2E 30 5F 5F 5F 5F 5F 5C 1.0 \.
1FEE2980 56 42 4C 4B 00 00 00 0F 00 00 00 07 00 01 00 02 VBLK.....
1FEE2990 34 32 35 36 36 32 33 32 33 32 33 34 33 39 33 39 4256623232343939
1FEE29A0 36 32 36 33 33 37 32 44 36 34 33 33 36 34 33 35 6263372D64336435.
1FEE29B0 33 30 33 30 32 30 33 35 00 00 00 00 00 00 00 00 30302035.....
1FEE29C0 00 00 04 0B 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FEE29D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FEE29E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FEE29F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
--- sdk --0x1FEE2800/0x20000000-----

```

```

root@chfIVBox: ~
1FEE2A00 56 42 4C 4B 00 00 00 10 00 00 00 08 00 00 00 02 VBLK.....
1FEE2A10 00 00 00 34 00 00 00 9C 02 04 09 05 44 69 73 6B ...4.....Disk
1FEE2A20 33 24 36 38 39 34 65 66 64 39 2D 30 61 64 32 2D 3$6894efd9-0ad2-
1FEE2A30 34 63 36 64 2D 62 39 36 33 2D 33 66 63 63 39 37 4c6d-b963-3fcc97
1FEE2A40 30 32 63 30 36 31 61 49 44 45 5C 44 49 53 4B 56 02c061aIDE\DISKV
1FEE2A50 42 4F 58 5F 48 41 52 44 44 49 53 4B 5F 5F 5F 5F BOX_HARDDISK____
1FEE2A60 5F 5F 5F 5F 5F 5F 5F 5F 5F 5F 5F 5F 5F 5F 5F 5F
1FEE2A70 5F 5F 5F 5F 5F 5F 5F 31 2E 30 5F 5F 5F 5F 5F 5C 1.0 \.
1FEE2A80 56 42 4C 4B 00 00 00 11 00 00 00 08 00 01 00 02 VBLK.....
1FEE2A90 34 32 35 36 33 38 33 38 36 34 33 35 36 34 36 36 4256383864356466.
1FEE2AA0 33 35 36 36 33 38 32 44 33 32 33 36 36 35 33 37 3566382D32366537
1FEE2AB0 33 39 33 34 32 30 33 34 00 00 00 00 00 00 00 00 39342034.....
1FEE2AC0 00 00 04 0B 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FEE2AD0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FEE2AE0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FEE2AF0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FEE2B00 56 42 4C 4B 00 00 00 12 00 00 00 00 00 00 00 00 VBLK.....

```

disk1 : 09E64D13-F392-443A-9317-7C8403ACE4B9


```
disk2 : C647EA1E-8965-46E9-8ED8-8434F17DF1CE
disk3 : 6894EFD9-0AD2-4C6D-B963-3FCC9702C061
```

If we pair it with the previous results:

```
disk1 : /dev/sdk
disk2 : /dev/sdl
disk3 : /dev/sdm
```

We can read the order from the VBLK partition descriptor (0x33)

```
root@chfIVBox: ~
1FEE2500 56 42 4C 4B 00 00 00 06 00 00 00 0B 00 00 00 01 VBLK.....
1FEE2510 00 00 40 33 00 00 00 33 02 04 0C 08 44 69 73 6B ..@3...3...Disk
1FEE2520 33 2D 30 31 00 00 00 00 00 00 00 00 00 00 04 0C 3-01.....
1FEE2530 00 00 00 00 00 00 00 41 00 00 00 00 00 00 00 00 .....A.....
1FEE2540 03 0F E8 00 02 04 0B 02 04 09 00 00 00 00 00 00 .....
1FEE2550 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FEE2560 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FEE2570 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FEE2580 56 42 4C 4B 00 00 00 07 00 00 00 0C 00 00 00 01 VBLK.....
1FEE2590 00 00 48 33 00 00 00 35 02 04 0D 08 44 69 73 6B ..H3...5...Disk
1FEE25A0 32 2D 30 31 00 00 00 00 00 00 00 00 00 00 04 0C 2-01.....
1FEE25B0 00 00 00 00 00 00 00 41 00 00 00 00 00 00 00 00 .....A.....
1FEE25C0 03 0F E8 00 02 04 0B 02 04 06 01 01 00 00 00 00 .....
1FEE25D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FEE25E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FEE25F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FEE2600 56 42 4C 4B 00 00 00 08 00 00 00 0D 00 00 00 01 VBLK.....
1FEE2610 00 00 48 33 00 00 00 35 02 04 0E 08 44 69 73 6B ..H3...5...Disk
1FEE2620 31 2D 30 31 00 00 00 00 00 00 00 00 00 00 04 0C 1-01.....
1FEE2630 00 00 00 00 00 00 00 41 00 00 00 00 00 00 00 00 .....A.....
1FEE2640 03 0F E8 00 02 04 0B 02 04 03 01 02 00 00 00 00 .....
1FEE2650 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FEE2660 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FEE2670 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FEE2680 56 42 4C 4B 00 00 00 09 00 00 00 00 00 00 00 00 VBLK.....
1FEE2690 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FEE26A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FEE26B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FEE26C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FEE26D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FEE26E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1FEE26F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
--- sdk --0x1FEE26F0/0x20000000-----
```

According to it the disk order is:

```
2. disk1 : /dev/sdk
1. disk2 : /dev/sdl
0. disk3 : /dev/sdm
```

As we can see the partition starts at the Offset 0x41. Again it is measured from the beginning of the logical disk so we must add to it the 0x3F value. So the partition will start at 0x80 on the 0th disk

(for me it is the disk 3 /dev/sdm). The 0x80 is measured in blocks so it means $0x80 * 0x200 = 0x10000$.

```

root@chfiVBox: ~
00010000 EB 52 90 4E 54 46 53 20 20 20 20 00 02 08 00 00 .R.NTFS .....
00010010 00 00 00 00 00 F8 00 00 3F 00 20 00 3F 00 00 00 .....?..?...
00010020 00 00 00 00 80 00 80 00 FF CF 1F 00 00 00 00 00 .....
00010030 55 53 01 00 00 00 00 00 FF FC 01 00 00 00 00 00 US.....
00010040 F6 00 00 00 01 00 00 00 12 D0 0A B4 0B 0B B4 74 .....t
00010050 00 00 00 00 FA 33 C0 8E D0 BC 00 7C FB 68 C0 07 .....3.....|.h..
00010060 1F 1E 68 66 00 CB 88 16 0E 00 66 81 3E 03 00 4E ..hf.....f.>..N.
00010070 54 46 53 75 15 B4 41 BB AA 55 CD 13 72 0C 81 FB TFSu..A..U..r....
00010080 55 AA 75 06 F7 C1 01 00 75 03 E9 D2 00 1E 83 EC U.u.....u.....
00010090 18 68 1A 00 B4 48 8A 16 0E 00 8B F4 16 1F CD 13 .h...H.....
000100A0 9F 83 C4 18 9E 58 1F 72 E1 3B 06 0B 00 75 DB A3 .....X.r.;...u..
000100B0 0F 00 C1 2E 0F 00 04 1E 5A 33 DB B9 00 20 2B C8 .....Z3...+.
000100C0 66 FF 06 11 00 03 16 0F 00 8E C2 FF 06 16 00 E8 f.....
000100D0 40 00 2B C8 77 EF B8 00 BB CD 1A 66 23 C0 75 2D @.+..w.....f#.u-
000100E0 66 81 FB 54 43 50 41 75 24 81 F9 02 01 72 1E 16 f..TCPAu$.r...
000100F0 68 07 BB 16 68 70 0E 16 68 09 00 66 53 66 53 66 h...hp..h..fSfSf
00010100 55 16 16 16 68 B8 01 66 61 0E 07 CD 1A E9 6A 01 U...h..fa....j.
00010110 90 90 66 60 1E 06 66 A1 11 00 66 03 06 1C 00 1E ..f`..f...f.....
00010120 66 68 00 00 00 00 66 50 06 53 68 01 00 68 10 00 fh....fP.Sh..h..
00010130 B4 42 8A 16 0E 00 16 1F 8B F4 CD 13 66 59 5B 5A .B.....fY[Z
00010140 66 59 66 59 1F 0F 82 16 00 66 FF 06 11 00 03 16 fYfY.....f.....
00010150 0F 00 8E C2 FF 0E 16 00 75 BC 07 1F 66 61 C3 A0 .....u...fa...
00010160 F8 01 E8 08 00 A0 FB 01 E8 02 00 EB FE B4 01 8B .....
00010170 F0 AC 3C 00 74 09 B4 0E BB 07 00 CD 10 EB F2 C3 ..<.t.....
00010180 0D 0A 41 20 64 69 73 6B 20 72 65 61 64 20 65 72 ..A disk read er
00010190 72 6F 72 20 6F 63 63 75 72 72 65 64 00 0D 0A 42 ror occurred...B
000101A0 4F 4F 54 4D 47 52 20 69 73 20 6D 69 73 73 69 6E OOTMGR is missin.
000101B0 67 00 0D 0A 42 4F 4F 54 4D 47 52 20 69 73 20 63 g...BOOTMGR is c
000101C0 6F 6D 70 72 65 73 73 65 64 00 0D 0A 50 72 65 73 ompressed...Pres.
000101D0 73 20 43 74 72 6C 2B 41 6C 74 2B 44 65 6C 20 74 s Ctrl+Alt+Del t
000101E0 6F 20 72 65 73 74 61 72 74 0D 0A 00 00 00 00 00 o restart.....
000101F0 00 00 00 00 00 00 00 00 80 9D B2 CA 00 00 55 AA .....U.
--- sdm --0x101F0/0x20000000-----

```

As we see the block size is 0x0200 (we already know it), and the size of cluster is 0x08. It is measured in blocks so the size of the cluster is $0x08 * 0x200 = 0x1000 = 4096$ bytes

The MFT file starts at the position 0x015355, it is measured in clusters, so it is $0x015355 * 0x1000 = 0x15355000$

Now let us try to find this position. As you know the RAID 5 stores the xored value of the data. We can check it easily now. This is the first block, so the data will be written to the 0th (disk3 /dev/sdm) and 1th disk (disk2 /dev/sdl). It means the xored value must be stored on the 2nd disk (disk1 /dev/sdk). On the 1th disk we will find no data


```

root@chfIVBox: ~
00010000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000100A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000100B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000100C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000100D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000100E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000100F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010100  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010110  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010120  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010130  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010140  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010150  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010160  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010170  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010180  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010190  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000101A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000101B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000101C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000101D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000101E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000101F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
--- sdl          --0x10000/0x20000000-----

```

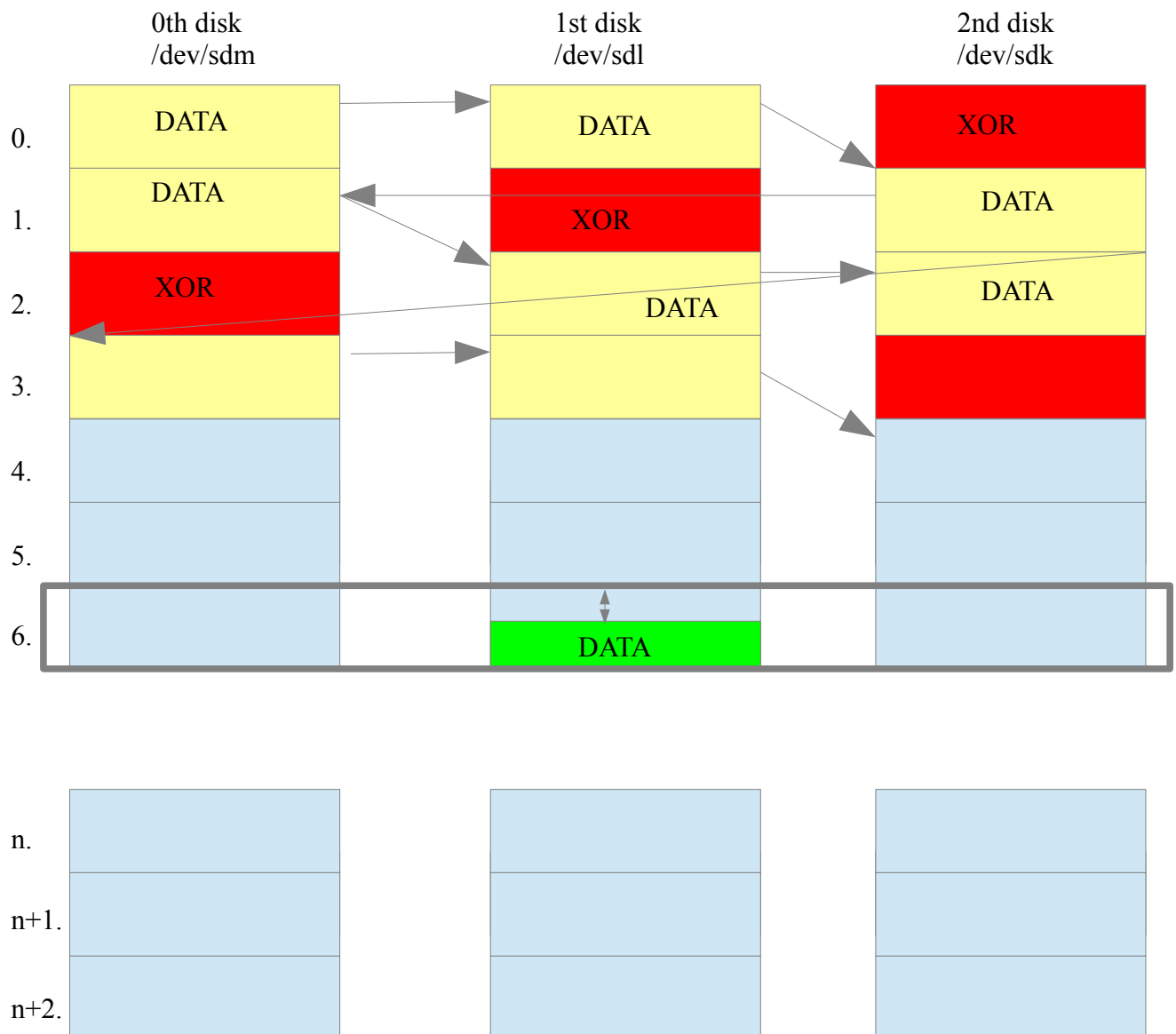
It means on the 2nd disk (disk 1 /dev/sdk) we should find the same NTFS record again, because anything XOR-ed with zero gives back the same value. And really on the disk /dev/sdk we find

```

root@chfiVBox: ~
00010000 EB 52 90 4E 54 46 53 20 20 20 20 00 02 08 00 00 .R.NTFS .....
00010010 00 00 00 00 00 F8 00 00 3F 00 20 00 3F 00 00 00 .....?..?...
00010020 00 00 00 00 80 00 80 00 FF CF 1F 00 00 00 00 00 .....
00010030 55 53 01 00 00 00 00 00 FF FC 01 00 00 00 00 00 US.....
00010040 F6 00 00 00 01 00 00 00 12 D0 0A B4 0B 0B B4 74 .....t
00010050 00 00 00 00 FA 33 C0 8E D0 BC 00 7C FB 68 C0 07 .....3.....|.h..
00010060 1F 1E 68 66 00 CB 88 16 0E 00 66 81 3E 03 00 4E ..hf.....f.>..N.
00010070 54 46 53 75 15 B4 41 BB AA 55 CD 13 72 0C 81 FB TFSu..A..U..r....
00010080 55 AA 75 06 F7 C1 01 00 75 03 E9 D2 00 1E 83 EC U.u.....u.....
00010090 18 68 1A 00 B4 48 8A 16 0E 00 8B F4 16 1F CD 13 .h...H.....
000100A0 9F 83 C4 18 9E 58 1F 72 E1 3B 06 0B 00 75 DB A3 .....X.r.;...u..
000100B0 0F 00 C1 2E 0F 00 04 1E 5A 33 DB B9 00 20 2B C8 .....Z3...+.
000100C0 66 FF 06 11 00 03 16 0F 00 8E C2 FF 06 16 00 E8 f.....
000100D0 40 00 2B C8 77 EF B8 00 BB CD 1A 66 23 C0 75 2D @.+..w.....f#.u-
000100E0 66 81 FB 54 43 50 41 75 24 81 F9 02 01 72 1E 16 f..TCPAu$.r..
000100F0 68 07 BB 16 68 70 0E 16 68 09 00 66 53 66 53 66 h...hp..h..fSfSf
00010100 55 16 16 16 68 B8 01 66 61 0E 07 CD 1A E9 6A 01 U...h..fa.....j.
00010110 90 90 66 60 1E 06 66 A1 11 00 66 03 06 1C 00 1E ..f`...f...f.....
00010120 66 68 00 00 00 00 66 50 06 53 68 01 00 68 10 00 fh....fP.Sh..h..
00010130 B4 42 8A 16 0E 00 16 1F 8B F4 CD 13 66 59 5B 5A .B.....fY[Z
00010140 66 59 66 59 1F 0F 82 16 00 66 FF 06 11 00 03 16 fYfY.....f.....
00010150 0F 00 8E C2 FF 0E 16 00 75 BC 07 1F 66 61 C3 A0 .....u...fa..
00010160 F8 01 E8 08 00 A0 FB 01 E8 02 00 EB FE B4 01 8B .....
00010170 F0 AC 3C 00 74 09 B4 0E BB 07 00 CD 10 EB F2 C3 ..<.t.....
00010180 0D 0A 41 20 64 69 73 6B 20 72 65 61 64 20 65 72 ..A disk read er
00010190 72 6F 72 20 6F 63 63 75 72 72 65 64 00 0D 0A 42 ror occurred...B
000101A0 4F 4F 54 4D 47 52 20 69 73 20 6D 69 73 73 69 6E OOTMGR is missin.
000101B0 67 00 0D 0A 42 4F 4F 54 4D 47 52 20 69 73 20 63 g...BOOTMGR is c
000101C0 6F 6D 70 72 65 73 73 65 64 00 0D 0A 50 72 65 73 ompressed...Pres.
000101D0 73 20 43 74 72 6C 2B 41 6C 74 2B 44 65 6C 20 74 s Ctrl+Alt+Del t
000101E0 6F 20 72 65 73 74 61 72 74 0D 0A 00 00 00 00 00 o restart.....
000101F0 00 00 00 00 00 00 00 00 80 9D B2 CA 00 00 55 AA .....U.
--- sdk --0x10000/0x20000000-----

```

The structure of the disks now looks like as follows:



Again we should follow the three steps used in the previous situation to get the position

- Calculate, in which stripe the data is.
- Calculate within that stripe on which disk the data is
- Calculate on that disk within that stripe where exactly the data starts

Now we should remember that one of the disks stores a xor value, so the useful information is always stored on the number of disks - 1.

So the position of the MFT file can be calculated on the following way

We get the stripe number as: $\text{INT}(0x15355000 / (2 * 0x10000)) = 0x0A9A$

The disk number is $0x15355000 - 0x0A9A * 2 * 0x10000 = 0x15000$. $\text{INT}(0x15000 / 0x10000) = 1$ it means we will find this data on the 2nd disk. But now because the XOR is continuously moving from one disk to another it is not surely the which one is disk1, it can be the disk1 or disk 2 as well,

depending on the stripe number. To figure it out we should calculate the stripe number modulo disk number so $0x0A9A \bmod 3 = 2$. It means the 1st disk and 2nd disk will store data and the 0th disk will store the xor value. So the 2nd data disk now the disk 2 (/dev/sdk)
 Offset within the stripe is $0x15000 - 0 * 0x10000 = 0x5000$ so the data will start at the 0x5000 byte from the beginning of the stripe.

Now find the position on the disk /dev/sdk. The position can be calculated as stripe number 0x0A9A multiplied by the stripe size 0x10000 plus the offset of the logical disk what is also 0x10000 plus the offset from the beginning of the stripe 0x5000. $0x0A9A * 0x10000 + 0x10000 + 0x5000 = 0xA9B5000$

```

root@chfivBox: ~
0A9B5000  46 49 4C 45  30 00 03 00  8D 10 20 00  00 00 00 00  FILE0.....
0A9B5010  01 00 01 00  38 00 01 00  98 01 00 00  00 04 00 00  ....8.....
0A9B5020  00 00 00 00  00 00 00 00  06 00 00 00  00 00 00 00  .....
0A9B5030  02 00 00 00  00 00 00 00  10 00 00 00  60 00 00 00  .....`...
0A9B5040  00 00 18 00  00 00 00 00  48 00 00 00  18 00 00 00  .....H.....
0A9B5050  BA B8 7C 06  BC 23 CC 01  BA B8 7C 06  BC 23 CC 01  ..|.##...|..#..
0A9B5060  BA B8 7C 06  BC 23 CC 01  BA B8 7C 06  BC 23 CC 01  ..|.##...|..#...
0A9B5070  06 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  .....
0A9B5080  00 00 00 00  00 01 00 00  00 00 00 00  00 00 00 00  .....
0A9B5090  00 00 00 00  00 00 00 00  30 00 00 00  68 00 00 00  .....0...h....
0A9B50A0  00 00 18 00  00 00 03 00  4A 00 00 00  18 00 01 00  .....J.....
0A9B50B0  05 00 00 00  00 00 05 00  BA B8 7C 06  BC 23 CC 01  .....|.##..
0A9B50C0  BA B8 7C 06  BC 23 CC 01  BA B8 7C 06  BC 23 CC 01  ..|.##...|..#..
0A9B50D0  BA B8 7C 06  BC 23 CC 01  00 40 00 00  00 00 00 00  ..|.##...@.....
0A9B50E0  00 40 00 00  00 00 00 00  06 00 00 00  00 00 00 00  .@.....
0A9B50F0  04 03 24 00  4D 00 46 00  54 00 00 00  00 00 00 00  ..$.M.F.T.....
0A9B5100  80 00 00 00  48 00 00 00  01 00 40 00  00 00 01 00  ....H.....@.....
0A9B5110  00 00 00 00  00 00 00 00  0F 00 00 00  00 00 00 00  .....
0A9B5120  40 00 00 00  00 00 00 00  00 00 01 00  00 00 00 00  @.....
0A9B5130  00 00 01 00  00 00 00 00  00 00 01 00  00 00 00 00  .....
0A9B5140  31 10 55 53  01 00 8C 94  B0 00 00 00  48 00 00 00  1.US.....H...
0A9B5150  01 00 40 00  00 00 05 00  00 00 00 00  00 00 00 00  ..@.....
0A9B5160  00 00 00 00  00 00 00 00  40 00 00 00  00 00 00 00  .....@.....
0A9B5170  00 10 00 00  00 00 00 00  08 00 00 00  00 00 00 00  .....
0A9B5180  08 00 00 00  00 00 00 00  31 01 54 53  01 00 00 00  .....1.TS....
0A9B5190  FF FF FF FF  00 00 00 00  00 00 01 00  00 00 00 00  .....
0A9B51A0  00 00 01 00  00 00 00 00  31 10 55 53  01 00 8C 94  .....1.US.....
0A9B51B0  B0 00 00 00  48 00 00 00  01 00 40 00  00 00 05 00  ....H.....@.....
0A9B51C0  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  .....
0A9B51D0  40 00 00 00  00 00 00 00  00 10 00 00  00 00 00 00  @.....
0A9B51E0  08 00 00 00  00 00 00 00  08 00 00 00  00 00 00 00  .....
0A9B51F0  31 01 54 53  01 00 00 00  FF FF FF FF  00 00 02 00  1.TS.....
--- sdk          --0xA9B5000/0x20000000-----

```

We need our entry, it will be most probably the 35. so the position of it is $0xA9B5000 + 35 * 0x400 = 0xA9BDC00$. Jump there by pressing ctrl-g then type this value


```

root@chfiVBox: ~
0A9BDC00  46 49 4C 45 30 00 03 00 04 29 20 00 00 00 00 00 FILE0....) .....
0A9BDC10  01 00 01 00 38 00 01 00 50 01 00 00 00 04 00 00 ....8...P.....
0A9BDC20  00 00 00 00 00 00 00 00 06 00 00 00 23 00 00 00 .....#...
0A9BDC30  04 00 00 00 00 00 00 00 10 00 00 00 60 00 00 00 .....`...
0A9BDC40  00 00 00 00 00 00 00 00 48 00 00 00 18 00 00 00 .....H.....
0A9BDC50  F7 83 76 B7 BC 23 CC 01 1E F0 84 F4 BC 23 CC 01 ..v..#.....#..
0A9BDC60  1E F0 84 F4 BC 23 CC 01 F7 83 76 B7 BC 23 CC 01 .....#....v..#...
0A9BDC70  20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A9BDC80  00 00 00 00 05 01 00 00 00 00 00 00 00 00 00 00 .....
0A9BDC90  00 00 00 00 00 00 00 00 30 00 00 00 68 00 00 00 .....0...h....
0A9BDCA0  00 00 00 00 00 00 04 00 4C 00 00 00 18 00 01 00 .....L.....
0A9BDCB0  05 00 00 00 00 00 05 00 F7 83 76 B7 BC 23 CC 01 .....v..#..
0A9BDCC0  F7 83 76 B7 BC 23 CC 01 F7 83 76 B7 BC 23 CC 01 ..v..#....v..#..
0A9BDCD0  F7 83 76 B7 BC 23 CC 01 00 00 00 00 00 00 00 00 ..v..#.....
0A9BDCE0  00 00 00 00 00 00 00 00 20 00 00 00 00 00 00 00 .....
0A9BDCF0  05 03 61 00 2E 00 74 00 78 00 74 00 58 00 7E 00 ..a...t.x.t.X.~.
0A9BDD00  80 00 00 00 48 00 00 00 01 00 00 00 00 00 05 00 ....H.....
0A9BDD10  00 00 00 00 00 00 00 00 1F 00 00 00 00 00 00 00 .....
0A9BDD20  40 00 00 00 00 00 00 00 00 00 02 00 00 00 00 00 @.....
0A9BDD30  00 FE 01 00 00 00 00 00 00 FE 01 00 00 00 00 00 .....
0A9BDD40  31 20 F3 E7 01 00 8C 94 FF FF FF FF 82 79 47 11 1 .....yG.
0A9BDD50  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A9BDD60  20 00 00 00 00 00 00 00 15 01 4E 00 65 00 77 00 .....N.e.w..
0A9BDD70  20 00 54 00 65 00 78 00 74 00 20 00 44 00 6F 00 ..T.e.x.t. .D.o.
0A9BDD80  63 00 75 00 6D 00 65 00 6E 00 74 00 2E 00 74 00 c.u.m.e.n.t...t.
0A9BDD90  78 00 74 00 00 00 00 00 80 00 00 00 18 00 00 00 x.t.....
0A9BDDA0  00 00 18 00 00 00 01 00 00 00 00 00 18 00 00 00 .....
0A9BDDA0  FF FF FF FF 82 79 47 11 00 00 00 00 00 00 00 00 .....yG.....
0A9BDDC0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A9BDDD0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A9BDDE0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0A9BDDF0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 04 00 .....
--- sdk          --0xA9BDC00/0x20000000-----

```

We find here a file called a.txt. Let us try to reassemble it. Now we have only one cluster chain, and it is defined as 3120F3E701. It should be interpreted as

- 0x31 means we store the first cluster on 3 bytes, and we store the length of cluster chain is stored on 1 byte.
- First the chain length is stored so the chain length is 0x20 bytes
- and the starting cluster is 0x01E7F3

Now calculate the position of this cluster. The cluster length is 4096 (0x1000) bytes. So it will start at $0x01E7F3 * 0x1000 = 0x1E7F3000$.

Again we should follow the three steps used in the previous situation to get the position

- Calculate, in which stripe the data is.
- Calculate within that stripe on which disk the data is
- Calculate on that disk within that stripe where exactly the data starts

We get the stripe number as: $\text{INT}(0x1E7F3000 / (2 * 0x10000)) = 0x0F3F$

The disk number is $0x1E7F3000 - 0x0F3F * 2 * 0x10000 = 0x13000$. $\text{INT}(0x13000 / 0x10000) = 1$ it means we will find this data on the 2nd disk. But now because the XOR is continuously moving from one disk to another it is not surely the which one is the 2nd disk, it can be the disk1 or disk 2 as well, depending on the stripe number. To figure it out we should calculate the stripe number modulo disk number so $0x0F3F \bmod 3 = 0$. It means the 0th disk and 1st disk will store data and the 2nd disk will store the xor value. So the 2nd data disk now the disk 1 (/dev/sdl)

Offset within the stripe is $0x13000 - 0 * 0x10000 = 0x3000$ so the data will start at the 0x3000 byte from the beginning of the stripe.

Now find the position on the disk /dev/sdl. The position can be calculated as stripe number 0x0F3F multiplied by the stripe size 0x10000 plus the offset of the logical disk what is also 0x10000 plus the offset from the beginning of the stripe 0x3000. $0x0F3F * 0x10000 + 0x10000 + 0x3000 = 0x0F403000$

```

root@chfIVBox: ~
0F402FF0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 02 00 .....
0F403000  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F403010  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F403020  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F403030  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F403040  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F403050  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa.
0F403060  61 61 61 61 0D 0A 61 61 61 61 61 61 61 61 61 61 aaaa..aaaaaaaaaa
0F403070  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F403080  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F403090  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa.
0F4030A0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F4030B0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F4030C0  61 61 61 61 61 61 61 61 61 61 61 0D 0A 61 61 61 61 aaaaaaaaaa..aaaa.
0F4030D0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F4030E0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F4030F0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F403100  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F403110  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F403120  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F403130  0D 0A 61 61 61 61 61 61 61 61 61 61 61 61 61 61 ..aaaaaaaaaaaaaaaa.
0F403140  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F403150  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa.
0F403160  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F403170  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F403180  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F403190  61 61 61 61 61 61 0D 0A 61 61 61 61 61 61 61 61 aaaaaa..aaaaaaaa.
0F4031A0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F4031B0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F4031C0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa.
0F4031D0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F4031E0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
--- sdl          --0xF403000/0x20000000-----

```

The size of the stripe is 0x10000 bytes and we started at the offset 0x3000 So until it we stored

0x10000 - 0x3000 = 0xD000 bytes. This was the 1st disk of the stripe, so the data will continue on the 0th disk of the next stripe what is the stripe 0x0F40. $0x0F40 \bmod 3 = 1$ it means the 0th data disk is /dev/sdk, and the 1st data disk is /dev/sdm.

Let us calculate the position on 0th disk (/dev/sdk). It is the stripe 0x0F40. The position can be calculated as stripe number 0x0F40 multiplied by the stripe size 0x10000 plus the offset of the logical disk what is also 0x10000 plus the offset from the beginning of the stripe 0x00. $0x0F40 * 0x10000 + 0x10000 + 0x00 = 0x0F410000$, the end of it will be at the end of the stripe 0x0F41FFFF

```

root@chfIVBox: ~
0F410000  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F410010  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F410020  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F410030  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F410040  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F410050  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F410060  0D 0A 61 61 61 61 61 61 61 61 61 61 61 61 61 61 . . aaaaaaaaaaaaaaa.
0F410070  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa.
0F410080  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F410090  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa.
0F4100A0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F4100B0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F4100C0  61 61 61 61 61 61 0D 0A 61 61 61 61 61 61 61 61 aaaaaa. . aaaaaaaa
0F4100D0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F4100E0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F4100F0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F410100  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F410110  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F410120  61 61 61 61 61 61 61 61 61 61 61 61 61 0D 0A 61 61 aaaaaaaaaaaaaa. . aa
0F410130  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F410140  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F410150  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F410160  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa.
0F410170  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F410180  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F410190  61 61 0D 0A 61 61 61 61 61 61 61 61 61 61 61 61 aa. . aaaaaaaaaaaaaa
0F4101A0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa.
0F4101B0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F4101C0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa.
0F4101D0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F4101E0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa.
0F4101F0  61 61 61 61 61 61 61 61 0D 0A 61 61 61 61 61 61 aaaaaaaa. . aaaaaa
--- sdk                --0xF4101F0/0x20000000-----

```

If you go a bit up you will see a lot of letter “a” there as well why? Because for the previous stripe it was the xor disk, and there were only data in one stripe, while no data on the other, and if you xor anything with 0 you will get back the same values now the letter “a”. You can easily check this theory by jumping to 0xF403000, where the “a” letters will start, because on the previous stripe we had a 0x3000 byte offset from the stripe start.

So until now $0xD000 + 0x10000 = 0x1D000$ byte of the file is stored. The length of the file is $0x1FE00$ it means on the last stripe we store $0x1FE00 - 0x1D000 = 0x2E00$ bytes.

It will be on the 1st disk of the stripe 0xF40. What means /dev/sdm, and the position 0x0F40 * 0x10000 + 0x10000 + 0x00 = 0x0F410000, the end of it will be at 0x0F412DFF

```

root@chfivBox: ~
0F40FFF0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0F410000  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F410010  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F410020  61 61 61 61 61 61 61 61 61 61 61 61 0D 0A 61 61 aaaaaaaaaaaaaa..aa
0F410030  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F410040  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F410050  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa.
0F410060  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa.
0F410070  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F410080  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa.
0F410090  61 61 0D 0A 61 61 61 61 61 61 61 61 61 61 61 61 aa..aaaaaaaaaaaaaa
0F4100A0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F4100B0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F4100C0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F4100D0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F4100E0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F4100F0  61 61 61 61 61 61 61 61 61 61 0D 0A 61 61 61 61 61 61 61 61 aaaaaaaa..aaaaaa
0F410100  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F410110  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F410120  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F410130  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F410140  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa.
0F410150  61 61 61 61 61 61 61 61 61 61 61 61 61 61 0D 0A aaaaaaaaaaaaaaaa..
0F410160  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa.
0F410170  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F410180  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F410190  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa.
0F4101A0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F4101B0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa.
0F4101C0  61 61 61 61 0D 0A 61 61 61 61 61 61 61 61 61 61 aaaa..aaaaaaaaaaaa
0F4101D0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa.
0F4101E0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
--- sdm                --0xF4101E0/0x20000000-----

```



```

root@chfIVBox: ~
0F412C70  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F412C80  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F412C90  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F412CA0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F412CB0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F412CC0  61 61 61 61 61 61 61 61 61 61 61 61 61 0D 0A 61 61 aaaaaaaaaaaaaa..aa
0F412CD0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa.
0F412CE0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa.
0F412CF0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F412D00  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa.
0F412D10  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F412D20  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F412D30  61 61 0D 0A 61 61 61 61 61 61 61 61 61 61 61 61 aa..aaaaaaaaaaaaaa
0F412D40  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F412D50  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F412D60  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F412D70  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F412D80  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F412D90  61 61 61 61 61 61 61 61 61 0D 0A 61 61 61 61 61 61 aaaaaaaa..aaaaaaa
0F412DA0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F412DB0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F412DC0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa.
0F412DD0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa
0F412DE0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 aaaaaaaaaaaaaaaaaa.
0F412DF0  61 61 61 61 61 61 61 61 61 61 61 61 61 61 0D 0A aaaaaaaaaaaaaaa..
0F412E00  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0F412E10  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0F412E20  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0F412E30  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0F412E40  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0F412E50  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0F412E60  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
--- sdm          --0xF412E60/0x20000000-----

```