

FORENSICS V2 LAB SERIES

Lab 05: File System

Document Version: 2021-01-14

Copyright © 2021 Network Development Group, Inc. www.netdevgroup.com

NETLAB+ is a registered trademark of Network Development Group, Inc.

Microsoft[®] and Windows[®] are registered trademarks of Microsoft Corporation in the United States and other countries. Google is a registered trademark of Google, LLC. Amazon is a registered trademark of Amazon in the United States and other countries.



Contents

Intr	oduction	3
Obj	ectives	3
Lab	Topology	4
Lab	Settings	5
1	Getting to Know HxD Hex Editor	6
2	Identifying File System Data in a FAT Formatted Evidence File	11
3	Identifying File System Data in an NTFS Formatted Evidence File	25
4	Identifying File System Data in an exFAT Formatted Evidence File	34



Introduction

This module will help the student understand what file systems are. They will also learn the limitations and advantages of each one and how to identify the differences.

Objectives

- J Understand what the most popular file systems are
- J Learn how to identify each file system using hex editors
- \hat{J} Learn what volume serial numbers are and how to decode them
- J Learn to decode volume creation dates and times

Lab 05: File System



Lab Topology





Lab Settings

The information in the table below will be needed to complete the lab. The task sections below provide details on the use of this information.

Virtual Machine	IP Address / Subnet Mask	Account (if needed)	Password (if needed)
Caine	172.16.16.30	caine	Train1ng\$
CSI-Linux	172.16.16.40	csi	csi
DEFT	172.16.16.20	deft	Train1ng\$
WinOS	172.16.16.10	Administrator	Train1ng\$



1 Getting to Know HxD Hex Editor

The ability to understand partitions and file systems is extremely important in performing digital examinations. Partitions determine how much data you can access, and file systems determine how that data is handled. The two are closely related but very different. In this lab, we will use a very useful FreeWare tool called HxD Hex Editor and Disk Editor¹ to access and parse data that will help us identify and understand partitions and file systems better.

- 1. To begin, launch the WinOS virtual machine to access the graphical login screen.
 - a. Select Send CTRL+ALT+DEL from the dropdown menu to be prompted with the login screen.



b. Log in as Administrator using the password: Train1ng\$.



 Once you are logged into the VM, launch the HxD Hex Editor program from the Windows Start menu by navigating to Start Menu > HxD Hex Editor. Alternatively, you can open HxD Hex Editor from the Desktop by double-clicking the icon called HxD:



- 3. Once you have HxD opened, you will see the main interface, as seen in the screenshot below. Let us look at the GUI first. The main GUI window has 2 toolbars by default.
 - a. They are highlighted as items 1 and 2. The toolbar highlighted as item 2 contains quick use icons that can also be found in the Menu bar highlighted as item 1.
 - b. The pane highlighted as item 3 is the area where the files' contents are displayed in hexadecimal and text view.



c. There is a feature called the Data Inspector tab found in the Special editors pane that we will need for this exercise. Let us check if it is open by navigating to View > Toolbars as seen in items 1 and 2. Once there, review the submenu that appears. If the Data Inspector option seen in item 3 does not have a checkmark beside it, then click it. If it does have a checkmark, then exit the menu by clicking on an empty area outside of the menu.



d. Now the Special editors pane will appear, and you will see the Data inspector tab highlighted below, which allows you to view and interpret data in different formats.





4. This powerful hex editor has many features and capabilities. We will cover the options that we will be using in the table below the following screenshots. The options we will be using are found in the Search, View, and Tools dropdown menus.

Tools Menu

HxD HxD		
File Edit Search View Analysis	Tools Window Help	
🗋 ờ - 💭 🎟 🥮 🖻 - 😢	Open main memory Shift+Ctrl+M Open disk Shift+Ctrl+D Open disk image Shift+Ctrl+1 File tools	■ Sector 0
	Options	

Search Menu

HXD HxD								
File Edit	Search View Analysis	Tools Window	v Help					
- 🙆 🗋	Find Replace Find again Find again (reversed)	Ctrl+F Ctrl+R F3 Shift+F3	Vindows (ANSI)	∨ hex		▶ ▶	Sector	0
	Go to	Ctrl+G]					
		s aro gra	vod out unt	l a filo is a	ddad ta th	o ann	lication	that

The options are grayed out until a file is added to the application that will enable the search function.

View Menu

HMD HxD				
File Edit Search View Analysis Tools Window	/ Help			
Adapt to window width Bytes per row Text encoding Offset base Visible columns Byte group size Toolbars Refresh F5	Vindows (ANSI) Hexadecimal Decimal Octal	hex	Sector	0



- 5. As always, please note that all the options are not listed here. There are many other features that will not be used in this course. If you would like to learn more about the tool, feel free to visit the HxD website <u>https://mh-nexus.de/en/hxd/</u>.
- 6. Now that you are familiar with some basic features of HxD, let us use it to look at some FEFs.

Open disk	The <i>Open disk</i> option allows you to add a local disk and view it in hexadecimal and raw text view.
Open disk image	The Open disk image option allows you to open a disk image file and view it in hexadecimal and raw text view.
Find	The <i>Find</i> option allows you to search the data that is shown in the main view pane.
Go to	The <i>Go to</i> option allows the user to go to a specific Offset in the data that is shown in the main view pane.
Offset base	The <i>Offset base</i> option allows you to switch between Hex, Decimal, and Octal characters in the view pane.



2 Identifying File System Data in a FAT Formatted Evidence File

All data stored on a storage device (hard drive or Solid-State Drive) is in the form of a magnetic field or electric charge and can be referred to as a binary digit or bit. To allow access to storage devices, there needs to some form of structure. This is done by the computer and is referred to as a Logical Disk Structure (Partitions and File Systems). To write data to the hard drive, the Logical Disk Structure allocates areas of the drive into individual blocks. This process must first be completed before the drive is usable.

In this lab, we will be reviewing how each Partition and File System is formatted.

1. Let us use HxD to review the FEF and learn how to read the data contained in a partition table. You should still have HxD open. If not, reopen it and click the Open disk image option from the Tools dropdown menu, as seen in items 1 and 2 in the screenshot below.

- M III 🤹 🖻	Open disk	Shift+Ctrl+D	₩ ₩ ₩ Sector 0	Special editors		
	Cle tools	Shilt+Cui+I		Data inspector		
	File tools	,				
	Options			Rinany (9 bit)	Invalid	
				Int8 act	Invalid	
				Ulint8 go to	n Invalid	
				Int16 go to	n Invalid	
				Ulnt16 go to	n: Invalid	
				Int24 go to	n: Invalid	
				UInt24 go to	p: Invalid	
				Int32 go to	p: Invalid	
				UInt32 go to	p: Invalid	
				Int64 go to	o: Invalid	
				UInt64 go to	o: Invalid	
				AnsiChar / char8_t	Invalid	
				WideChar / char16_t	Invalid	
				UTF-8 code point	Invalid	
				Single (float32)	Invalid	
				Double (float64)	Invalid	
				OLETIME	Invalid	
				FILETIME	Invalid	
				DOS date	Invalid	
				DOS time	Invalid	
				DOS time & date	Invalid	
				time t (32 bit)	Invalid	
				Byte order		
				Little endian	O Big endian	
				Show integers in hexadecim	nal base	
Checksum Search						
June					Ref	resh
Algorithm	Chashrum	Ileane			, nen	can
Algonium	Checksum	Usage				



 The Open disk image window will appear. Use this window to browse to This PC > Desktop and double-click the folder Toolbox > Datasets > Lab5. This will open the folder revealing 3 FEFs. Select the file called NDG FAT Lab5.001 and click the Open button as highlighted below.

👂 Open disk image			×	
🗧 🕁 👻 🕇 📙 « Toolbax »	Datasets → Lab5 🛛 🗸 Č	Search Lab5	<u>م</u>	0
Organize		80 - 🔳	0	Special editors
Pictures * ^	Name	Date modified	Typ	Data inspector
Toolbox			1.91	14 4 14 14
	NDG exFAT Lab5.001	6/30/2020 9:49 PM	Wu	Binary (8 bit) Invalid
This PC	NDG exFAT Lab5.001.bt	6/30/2020 9:49 PM	Tex	Int8 go to: Invalid
Desktop	NDG FAT Lab5.001	6/30/2020 9:46 PM	Wi	UInt8 go to: Invalid
r Toolhox	NDG FAT Lab5.001.6t	6/30/2020 9:46 PM	Tex	Int16 go to: Invalid
- IOODOX	MDG NTFS Lab5.001	6/30/2020 9:51 PM	Wit	UInt16 go to: Invalid
Autoruns for Windows	NDG NTFS Lab5.001.txt	6/30/2020 9:52 PM	Tex	Int24 go to: Invalid
V Datasets 4				UInt24 go to: Invalid
CPi				Int32 go to: Invalid
1.562				UInt32 go to: Invalid
Labz				Int64 go to: Invalid
Lab4				UInt64 go to: Invalid
Lab5 5				AnsiChar / cha Invalid
> deft-8.2-002				WideChar / ch Invalid
Magnet Process Captu				IITE-8 code no Invalid
Magnet Process Captu	<		>	Byte order
		1	_	Little endian O Big endian
File name:	~	All files (".")	~	Show integers in heradecimal
		Open Cancel		
		open		
				 Refresh
		[
Algorithm Check	sum	Usage		
-				

3. The Sector size window will appear. This window allows you to select the sector size of the image. In this lab, we will leave the option as 512 (hard disks/floppy disks) and click OK as highlighted below.



Do NOT change the specified sector size from 512 Bytes.



4. You will see the window below appear. As you can see in the screenshot below, the view pane now contains the hexadecimal representation on the left of the pane. This is highlighted as item 1 below. Immediately beside the hexadecimal values is the Decoded text view, highlighted as item 2.

| | | - | + + | 16 | ~ | - V | Vindo

 | ows
 | (ANS
 | 31)
 | | \sim
 | hex
 | | ~
 | | |
 | |
|--------|--|--|---|--|--|--
--
--
--

--
--
--
---|--|---
--
--
--|---
--|---|--|
| b5.001 | | | | | | |

 |
 |
 |
 | |
 |
 | | 2
 | Special editors | |
 | × |
| 00 0 | 1 02 | 03 | 04 | 05 | 06 | 07 | 08

 | 09
 | 0A
 | ов
 | oc | OD
 | OE
 | OF | Decoded text
 | Data inspector | |
 | |
| bo o | 0 00 | 00 | 00 | 00 | 00 | 00 | 00

 | 00
 | 00
 | 00
 | 00 | 00
 | 00
 | 00 |
 | 14 4 5 5 | |
 | |
| 00 0 | 0 00 | 00 | 00 | 00 | 00 | 00 | 00

 | 00
 | 00
 | 00
 | 00 | 00
 | 00
 | 00 |
 | | 1 |
 | |
| 00 0 | 0 00 | 00 | 00 | 00 | 00 | 00 | 00

 | 00
 | 00
 | 00
 | 00 | 00
 | 00
 | 00 | •••••
 | Binary (8 bit) | | 11111010
 | ^ |
| 00 0 | 0 00 | 00 | 00 | 00 | 00 | 00 | 00

 | 00
 | 00
 | 00
 | 00 | 00
 | 00
 | 00 | •••••
 | Int8 | go to: | -6
 | |
| 00 0 | 0 00 | 00 | 00 | 00 | 00 | 00 | 00

 | 00
 | 00
 | 00
 | 00 | 00
 | 00
 | 00 |
 | UInt8 | go to: | 250
 | |
| 00 0 | 0 00 | 00 | 00 | 00 | 00 | 00 | 00

 | 00
 | 00
 | 00
 | 00 | 00
 | 00
 | 00 |
 | Int16 | go to: | 13306
 | |
| 00 0 | 0 00 | 00 | 00 | 00 | 00 | 00 | 00

 | 00
 | 00
 | 00
 | 00 | 00
 | 00
 | 00 |
 | UInt16 | go to: | 13306
 | |
| 00 0 | 0 00 | 00 | 00 | 00 | 00 | 00 | 00

 | 00
 | 00
 | 00
 | 00 | 00
 | 00
 | 00 |
 | Int24 | go to: | -4180998
 | |
| 00 0 | 0 00 | 00 | 00 | 00 | 00 | 00 | 00

 | 00
 | 00
 | 00
 | 00 | 00
 | 00
 | 00 |
 | UInt24 | go to: | 12596218
 | |
| 00 0 | 0 00 | 00 | 00 | 00 | 00 | 00 | 00

 | 00
 | 00
 | 00
 | 00 | 00
 | 00
 | 00 |
 | Int32 | an to: | -1900006406
 | |
| 00 0 | 0 00 | 00 | 00 | 00 | 00 | 00 | 00

 | 00
 | 00
 | 00
 | 00 | 00
 | 00
 | 00 |
 | Illint32 | go to: | 2394960890
 | |
| 00 0 | 0 00 | 00 | 00 | 00 | 00 | 00 | 00

 | 00
 | 00
 | 00
 | 00 | 00
 | 00
 | 00 | •••••
 | Int64 | go to: | 8035340264637244410
 | |
| 00 0 | 0 00 | 00 | 00 | 00 | 00 | 00 | 00

 | 00
 | 00
 | 00
 | 00 | 00
 | 00
 | 00 | •••••
 | III.ukeA | <u>yo to:</u> | 0005040264607244410
 | |
| 00 0 | 0 00 | 00 | 00 | 00 | 00 | 00 | 00

 | 00
 | 00
 | 00
 | 00 | 00
 | 00
 | 00 |
 | AuriChau (ala | <u>yo to:</u> | 6953349204037244410
 | |
| 00 0 | 0 00 | 00 | 00 | 00 | 00 | 00 | 00

 | 00
 | 00
 | 00
 | 00 | 00
 | 00
 | 00 |
 | AnsiChar / cha | r8_t | u
ar
 | |
| 00 0 | 0 00 | 00 | 00 | 00 | 00 | 00 | 00

 | 00
 | 00
 | 00
 | 00 | 00
 | 00
 | 00 |
 | WideChar / ch | arlb_t | 2/8
 | |
| 00 0 | 0 00 | 00 | 00 | 00 | 00 | 00 | 00

 | 00
 | 00
 | 00
 | 00 | 00
 | 00
 | 00 |
 | LITE-8 code no | int | Invalid code unit
 | • |
| 00 0 | 0 00 | 00 | 00 | 00 | 00 | 00 | 00

 | 00
 | 00
 | 00
 | 00 | 00
 | 00
 | 00 |
 | Byte order | |
 | |
| 00 0 | 0 00 | 00 | 00 | 00 | 00 | 00 | 00

 | 00
 | 00
 | 00
 | 00 | 00
 | 00
 | 00 |
 | Little endia | n | ○ Big endian
 | |
| 00 0 | 0 00 | 00 | 00 | 00 | 00 | 00 | 04

 | 41
 | 11
 | 00
 | 00 | 00
 | 80
 | 01 | ····· ··· ··· ·· ·· ·· ·· ·· ·· ·· ·· ·
 | Show integer | s in hexadeo | cimal base
 | |
| Searc | h (0 hi | ts) | | | | |

 |
 |
 |
 | |
 |
 | |
 | | |
 | |
| | | | | | | |

 |
 |
 |
 | |
 |
 | |
 | | | ✓ Refres
 | h |
| ithm | _ | 6 | beck | | | - | -

 | _
 | _
 | -
 | - | -
 | -
 | Ucad |
 | | |
 | _ |
| iun n | | | anecs | Jum | | |

 |
 |
 |
 | |
 |
 | usay |
 | | |
 | |
| | | | | | | |

 |
 |
 |
 | |
 |
 | |
 | | |
 | |
| | | | | | | |

 |
 |
 |
 | |
 |
 | |
 | | |
 | |
| | | | | | | |

 |
 |
 |
 | |
 |
 | |
 | | |
 | |
| sult: | | | | | | |

 |
 |
 |
 | |
 |
 | |
 | | |
 | |
| | S.UU
00
00
00
00
00
00
00
00
00 | 00 01 02
00 01 02
00 00 00
00 00
000 | 00 01 02 03
00 00 00 00 00
00 00 00
00 00 00
00 00 00 00
00 00
00 00 00
00 00
00 00 00
00 00
00 00 00
00 00 00
00 00 00
00 00
00 00 00
00 00
00 00 00
00 00 00
00 00
00 00 00
00 00
00 00 00 | 00 01 02 03 04
00 00 00 00 00 00
00 00 00
00 00 00 00
00 00 00
00 00 00 00
00 00
00 00 00
00 00 00 00
00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 | 00 01 02 03 04 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | 00 01 02 03 04 05 06 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | 00 01 02 03 04 05 06 07 00 <t< td=""><td>00 01 02 03 04 05 06 07 08 00 <t< td=""><td>00 01 02 03 04 05 06 07 08 09 00 <t< td=""><td>00 01 02 03 04 05 06 07 08 09 0. 00 <t< td=""><td>000 01 02 03 04 05 06 07 06 09 0.4 08 00</td><td>000 01 02 03 04 05 06 07 08 09 0Å 08 00 <t< td=""><td>00 01 02 03 04 05 06 07 08 09 0A 0B 0C 00 <td< td=""><td>OP O1 O2 O3 O4 O5 O6 O7 O8 O9 OA OB OC OD <th< td=""><th>Souri Cont O2 O3 O4 O5 O6 O7 O8 O9 OA OB
OC OD OE OF 00 <td< th=""><td>00 01 02 03 04 05 06 07 08 09 0Å 08 0C 00 <td< td=""><td>Special editors 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Decoded text Decoded text Data inspector 00</td><td>Dot 01 02 03 04 05 04 05 04 05 06 07 08 09 0A 08 0C 00</td><td>Special editors OC 01 02 03 04 05 06 07 06 09 0A 08 0C 0D 02 0F Decoded text Data inspector Data inspector</td></td<></td></td<></th></th<></td></td<></td></t<></td></t<></td></t<></td></t<></td></t<> | 00 01 02 03 04 05 06 07 08 00 <t< td=""><td>00 01 02 03 04 05 06 07 08 09 00 <t< td=""><td>00 01 02 03 04 05 06 07 08 09 0. 00 <t< td=""><td>000 01 02 03 04 05 06 07 06 09 0.4 08 00</td><td>000 01 02 03 04 05 06 07 08 09 0Å 08 00 <t< td=""><td>00 01 02 03 04 05 06 07 08 09 0A 0B 0C 00
00 <td< td=""><td>OP O1 O2 O3 O4 O5 O6 O7 O8 O9 OA OB OC OD <th< td=""><th>Souri Cont O2 O3 O4 O5 O6 O7 O8 O9 OA OB OC OD OE OF 00 <td< th=""><td>00 01 02 03 04 05 06 07 08 09 0Å 08 0C 00 <td< td=""><td>Special editors 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Decoded text Decoded text Data inspector 00</td><td>Dot 01 02 03 04 05 04 05 04 05 06 07 08 09 0A 08 0C 00</td><td>Special editors OC 01 02 03 04 05 06 07 06 09 0A 08 0C 0D 02 0F Decoded text Data inspector Data inspector</td></td<></td></td<></th></th<></td></td<></td></t<></td></t<></td></t<></td></t<> | 00 01 02 03 04 05 06 07 08 09 00 <t< td=""><td>00 01 02 03 04 05 06 07 08 09 0. 00 <t< td=""><td>000 01 02 03 04 05 06 07 06 09 0.4 08 00</td><td>000 01 02 03 04 05 06 07 08 09 0Å 08 00 <t< td=""><td>00 01 02 03 04 05 06 07 08 09 0A 0B 0C 00 <td< td=""><td>OP O1
 O2 O3 O4 O5 O6 O7 O8 O9 OA OB OC OD <th< td=""><th>Souri Cont O2 O3 O4 O5 O6 O7 O8 O9 OA OB OC OD OE OF 00 <td< th=""><td>00 01 02 03 04 05 06 07 08 09 0Å 08 0C 00 <td< td=""><td>Special editors 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Decoded text Decoded text Data inspector 00</td><td>Dot 01 02 03 04 05 04 05 04 05 06 07 08 09 0A 08 0C 00</td><td>Special editors OC 01 02 03 04 05 06 07 06 09 0A 08 0C 0D 02 0F Decoded text Data inspector Data inspector</td></td<></td></td<></th></th<></td></td<></td></t<></td></t<></td></t<> | 00 01 02 03 04 05 06 07 08 09 0. 00 <t< td=""><td>000 01 02 03 04 05 06 07 06 09 0.4 08 00</td><td>000 01 02 03 04 05 06 07 08 09 0Å 08 00 <t< td=""><td>00 01 02 03 04 05 06 07 08 09 0A 0B 0C 00 <td< td=""><td>OP O1 O2 O3 O4 O5 O6 O7 O8 O9 OA OB OC OD <th< td=""><th>Souri Cont O2 O3 O4 O5 O6 O7 O8 O9 OA OB OC OD OE OF 00 <td< th=""><td>00 01 02 03 04 05 06 07 08 09 0Å 08 0C 00 00 00 00 00
00 <td< td=""><td>Special editors 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Decoded text Decoded text Data inspector 00</td><td>Dot 01 02 03 04 05 04 05 04 05 06 07 08 09 0A 08 0C 00</td><td>Special editors OC 01 02 03 04 05 06 07 06 09 0A 08 0C 0D 02 0F Decoded text Data inspector Data inspector</td></td<></td></td<></th></th<></td></td<></td></t<></td></t<> | 000 01 02 03 04 05 06 07 06 09 0.4 08 00 | 000 01 02 03 04 05 06 07 08 09 0Å 08 00 <t< td=""><td>00 01 02 03 04 05 06 07 08 09 0A 0B 0C 00 <td< td=""><td>OP O1 O2 O3 O4 O5 O6 O7 O8 O9 OA OB OC OD <th< td=""><th>Souri Cont O2 O3 O4 O5 O6 O7 O8 O9 OA OB OC OD OE OF 00 <td< th=""><td>00 01 02 03 04 05 06 07 08 09 0Å 08 0C 00 <td< td=""><td>Special editors 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Decoded text Decoded text Data inspector 00</td><td>Dot 01 02 03 04 05 04 05 04 05 06 07 08 09 0A 08 0C 00
 00 00</td><td>Special editors OC 01 02 03 04 05 06 07 06 09 0A 08 0C 0D 02 0F Decoded text Data inspector Data inspector</td></td<></td></td<></th></th<></td></td<></td></t<> | 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 00 <td< td=""><td>OP O1 O2 O3 O4 O5 O6 O7 O8 O9 OA OB OC OD <th< td=""><th>Souri Cont O2 O3 O4 O5 O6 O7 O8 O9 OA OB OC OD OE OF 00 <td< th=""><td>00 01 02 03 04 05 06 07 08 09 0Å 08 0C 00 <td< td=""><td>Special editors 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Decoded text Decoded text Data inspector 00</td><td>Dot 01 02 03 04 05 04 05 04 05 06 07 08 09 0A 08 0C 00</td><td>Special editors OC 01 02 03 04 05 06 07 06 09 0A 08 0C 0D 02 0F Decoded text Data inspector Data inspector</td></td<></td></td<></th></th<></td></td<> | OP O1 O2 O3 O4 O5 O6 O7 O8 O9 OA OB OC OD OD <th< td=""><th>Souri Cont O2 O3 O4 O5 O6 O7 O8 O9 OA OB OC OD OE OF 00 <td< th=""><td>00 01 02 03 04 05 06 07 08 09 0Å 08 0C 00 <td< td=""><td>Special editors 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Decoded text Decoded text Data inspector 00</td><td>Dot 01 02 03 04 05 04 05 04 05 06 07 08 09 0A 08 0C 00
00 00</td><td>Special editors OC 01 02 03 04 05 06 07 06 09 0A 08 0C 0D 02 0F Decoded text Data inspector Data inspector</td></td<></td></td<></th></th<> | Souri Cont O2 O3 O4 O5 O6 O7 O8 O9 OA OB OC OD OE OF 00 <td< th=""><td>00 01 02 03 04 05 06 07 08 09 0Å 08 0C 00 <td< td=""><td>Special editors 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Decoded text Decoded text Data inspector 00</td><td>Dot 01 02 03 04 05 04 05 04 05 06 07 08 09 0A 08 0C 00</td><td>Special editors OC 01 02 03 04 05 06 07 06 09 0A 08 0C 0D 02 0F Decoded text Data inspector Data inspector</td></td<></td></td<> | 00 01 02 03 04 05 06 07 08 09 0Å 08 0C 00 <td< td=""><td>Special editors 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Decoded text Decoded text Data inspector 00</td><td>Dot 01 02 03 04 05 04 05 04 05 06 07 08 09 0A 08 0C 00</td><td>Special editors OC 01 02 03 04 05 06 07 06 09 0A 08 0C 0D 02 0F Decoded text Data inspector Data inspector</td></td<> | Special editors 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Decoded text Decoded text Data inspector 00 | Dot 01 02 03 04 05 04 05 04 05 06 07 08 09 0A 08 0C 00
00 00 | Special editors OC 01 02 03 04 05 06 07 06 09 0A 08 0C 0D 02 0F Decoded text Data inspector Data inspector |

5. Now let us begin reviewing the Master Boot Record (MBR – Partition Table). The MBR - partition table can be found in the first sector (sector 0) of storage disks. It can be broken into 3 sections: the Bootstrap code Area, the partition table, and the Boot Record Signature. Each of these sections is stored at specific offsets. The table below provides a description of their locations and sizes.

	Structure of a generic MBR													
Offsets wi	thin sector	Length												
Decimal	Hexadecimal	(in Bytes)	Description											
000 - 445	000 – 1BD	446	Bootstrap Code Area											
446 - 509	1BE – 1FD	64	Partition Table											
510 - 512	1FE – 1FF	2	Boot Record Signature											



http://blog.hakzone.info/posts-and-articles/bios/analysing-the-masterboot-record-mbr-with-a-hex-editor-hex-workshop/



6. As you saw in the table above, the partition table is located at offset 4446 – 509. We will be focusing on the data within the partition table in this exercise. Each partition entry in the partition table is 16 bytes long. Typically, each drive can have 4 primary partitions or 3 primary partitions and 1 extended partition. This makes sense since the partition table is only 64 bytes in size (446 - 509 bytes); it can only store 4 entries, and each entry is 16 bytes long.



The additional extended partition allows for more logical partitions to be created; however, extended partitions will not be covered in this lab.

7. Now, let us look at the data. Since we will be using decimal values to go to offset 446, we will need to change the Offset base to decimal. To do this, click the View dropdown menu option from the menu bar and hover over the Offset base option, then select Decimal as highlighted in items 1, 2, and 3 below.





8. Now that the offsets are in decimal, let us go to offset 446. Once again, go to the menu bar. This time click the Search dropdown menu option and then click Go to from the dropdown menu or press Ctrl+G as highlighted in items 1 and 2 below.

🗋 🚵 🗝 🔚 🗸	P Find				Ctrl	+F	dov	nvs (/	ANSI))	1	/ d	ec	~					
NDG FAT L	Replace				Ctrl	+R										Special editors			
Offset (d	Find agai	ר 			1.:0.	F3	18	09	10	11	12	13	14	15	Decoded text ^	Data inspector			
00000000	Find agai	i (reve	rsea)	3	niit+	r5	2	00	00	00	00	00		00	Peoplace Serve				
000000011	Go to				Ctrl	+G	50	00	00	00	00	00	00	00					
000000128	00 00 0	0 00	00	00	00	00	00	00	00	00	00	00	00	00		Binary (8 bit)		11111010	1
000000144	00 00 0	0 00	00	00	00	00	00	00	00	00	00	00	00	00		Int8	go to:	-6	-1
000000160	00 00 0	0 00	00	00	00	00	00	00	00	00	00	00	00	00		UInt8	ao to:	250	
000000176	00 00 0	0 00	00	00	00	00	00	00	00	00	00	00	00	00	•••••	Int16	ao to:	13306	
000000192		0 00	00	00	00	00	00	00	00	00	00	00	00	00		UInt16	ao to:	13306	
000000224	00 00 0	0 00	00	00	00	00	00	00	00	00	00	00	00	00		Int24	an to:	-4180998	
000000240	00 00 0	0 00	00	00	00	00	00	00	00	00	00	00	00	00		Llint24	an to:	12596218	-1
000000256	00 00 0	0 00	00	00	00	00	00	00	00	00	00	00	00	00		Int32	go to:	-1900006406	
000000272	00 00 0	0 00	00	00	00	00	00	00	00	00	00	00	00	00		Illint32	go to:	2304060800	
000000288	00 00 0	0 00	00	00	00	00	00	00	00	00	00	00	00	00		Int64	go to:	8035340264637244410	
000000304		0 00	00	00	00	00	00	00	00	00	00	00	00	00		Illint64	go to:	0005040264607044410	
000000320		0 00	00	00	00	00	00	00	00	00	00	00	00	00		AnaiChan (abas0 4	<u>yo to:</u>	4	
00000035	00 00 0	0 00	00	00	00	00	00	00	00	00	00	00	00	00		Ansichar / charo_t		130	
00000036	00 00 0	0 00	00	00	00	00	00	00	00	00	00	00	00	00		WideChar/ charit	<u>j</u> t	200	
00000038	00 00 0	0 00	00	00	00	00	00	00	00	00	00	00	00	00		Dite order		Invalid code unit	_
00000040	00 00 0	0 00	00	00	00	00	00	00	00	00	00	00	00	00		Byte order		O. D	
00000041	00 00 0	0 00	00	00	00	00	00	00	00	00	00	00	00	00		Little endian		O Big endian	
000000432	00 00 0	0 00	00	00	00	00	04	41	11	00	00	00	80	01	····· ··· ·· · · · · · · · · · · · · ·	Show integers in	hexadeo	imal base	
Checksum	Search (0 h	ts)																	
																		✓ Refrest	n
Alaani	4 la												1.0						_
Algon	unn	C	IECKS	um									0	saye					

9. The Go to window will appear. This window allows you to enter an offset, and it will place the cursor at the beginning of the offset. Let us type 446 in the text box highlighted as item 1 below. Next, click the radio button beside dec, as seen in item 2 below. This tells Go to that you are searching for a decimal value. Once you have verified that everything is correct, leave the other option as default and click OK as seen in item 3 below.

Go to	×
Offset:	0
	44¢
🔾 hex 🝳 🖲 dec	⊖ oct
Offset relative to	
egin	
O current offset	
O end (backwards)	
3 01	Canaal
UK	Cancel



10. Your cursor will be taken to offset 446. Let us highlight the 16 bytes after the cursor, as seen in item 1 below. You can use the status bar at the bottom of the main window to count the length of your selection, as highlighted in item 2 below.

FO AO	File Edit	Search	Vie	w A	Analy	/sis	Tool 5	s W	/indo	w ł	Help	NSD			l a	lec		1				-	БX
:	NDG FAT La	b5.001			: -					indo									Special editors				×
_			-	-	-	04		0.0			~~	10		10	10			D	Data inspector	1			
	/rrset(a)	00	01	02	03	04	US	06	07	Uð	09	10	11	12	13	14	15	Decoded text	Data inspector				
0	00000288	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		14 4 10 101				
	00000304	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		Discours (0.1-14)		1000000		
0	00000320	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		Binary (8 bit)		1000000		^
0	00000352	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		Int8	go to:	-128		- 11
0	00000368	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		UInt8	go to:	128		- 11
0	00000384	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		Int16	go to:	384		
0	00000400	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		UInt16	go to:	384		
0	00000416	00	00	00	00	00	00	00	00	00	00	00	00	00	29	00	00		Int24	go to:	65920		
0	00000432	00	00	00	00	00	00	00	00	04	41	11	00	00		80	01		UInt24	go to:	65920		
0	00000448	01	00	OE	FE	ЗF	04	ЗF	00	00	00	86	39	01	00	00	00	þ?.?†9	Int32	ao to:	65920		
0	00000464	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		Illnt32	an to:	65920		- 1
0	00000480	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		Int64	go to:	3062426357675	46240	
	00000496	00	00	00	00	00	00	00	00	00	00	00	00	00	00	55	AA OO		111	<u>go to:</u>	2062420337073	40240	
	00000512	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		Uinto4	<u>yo to:</u>	5002420557075	40240	
	00000520	00	00	00	00	00	nn	00	00	00	00	00	00	00	00	00	00		AnsiChar / char	8_t	ŧ		
0	00000560	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		WideChar / cha	r16_t	Ъ		
0	00000576	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		IITE-8 code noi	nt	Unevnected co	ntinuation	14*
0	00000592	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		Byte order				
0	00000608	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		 Little endian 	i.	🔾 Big endian		
0	00000624	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	······································	Show integers	in hexade	cimal base		
Resu	Checksum	Sear	ch (C	hits))														_				
đ																					~	Refresh	1
	Algo	rithm			Ch	ecks	um									U	sage						
	Expected re	sult:																					
×																		2					
Offs	et(d): 446			B	lock(d): 4	46-4	61							l	Leng	th(d):	16	0.	enwrite			



11. The data you just highlighted is the partition entry for the first partition on the disk. This highlighted data can be broken up into 6 sections. The table below shows the different sections.

0x80 | 0x01 01 00 | 0x0E | 0xFE 3F 04 | 0x3F 00 00 00 | 0x86 39 01 00

Offset	Length (in Bytes)	Description
0x80	1	This is the first character and denotes whether the partition is active or not. 0x80h indicates that the partition is active. Alternately, 0x00 would indicate that the partition is inactive. ($0 = Non$ -Bootable / $80 = Bootable$).
0x01 01 00	3	The next 3 bytes represent the starting sector of the partition. It is stored as a Cylinder Head Sector (CHS) value and is also in little-endian. This means the starting sector is 0x00 01 01.
0×0E	1	The fifth (5 th) value represents the type of filesystem that is on this partition. <i>0E</i> represents a <i>FAT</i> file system.
0xFE 3F 04	3	The next 3 bytes represent the ending sector of the partition. It is also stored as a Cylinder Head Sector (CHS) value and is in little-endian. This means the partition's ending sector is located at 0x04 3F FE.
0x3F 00 00 00	4	The next 4 values indicate the starting sector of the file system in hexadecimal. It is stored in little-endian and so the value is 0x00 00 00 3F or just 0x3F. You can highlight the characters and view the converted data in the Data inspector pane. 0x3F converted to decimal is 63, which indicates that the starting sector for this file system is sector 63.
0x86 39 01 00	4	The last 4 values represent the number of sectors in the partition. This too is stored in little-endian so should be viewed as 00 01 39 86. You can highlight the characters and view the converted data in the <i>Data inspector</i> pane. When you convert this value to decimal, you will get 80262 sectors. To get the partition size, multiply the number of sectors (80262) by the size of each sector (512), in this case, 40,094,144 bytes (approx. 40MB.).



12. The partition entries precede the Boot Record Signature; this can be found at the end of the MBR as hex 55 AA (0x55AA) (bytes 510 and 511).

Offset(d)	00	01	02	03	04	05	06	07	08	69	10	11	12	13	14	15
000000288	00	00	00	0.0	00	00	00	00	00	0.0	00	0.0	00	0.0	00	00
000000304	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000320	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000336	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0.0
000000352	00	00	00	00	00	00	00	00	00	0.0	00	00	00	00	00	0.0
000000368	00	00	00	0.0	00	00	00	00	00	0.0	00	00	00	0.0	00	0.0
000000384	00	00	00	0.0	00	0.0	00	0.0	00	00	00	00	00	0.0	00	00
000000400	00	00	00	0.0	00	0.0	00	00	00	0.0	00	00	00	0.0	00	00
000000416	00	0.0	00	0.0	00	0.0	00	0.0	00	0.0	00	0.0	00	0.0	00	0.0
000000432	00	00	00	0.0	00	00	00	00	04	41	11	0.0	00	0.0	80	01
000000448	01	00	0E	ΕE	ЗF	04	ЗF	0.0	00	00	86	39	01	0.0	00	00
000000464	00	00	00	00	00	0.0	00	00	00	00	00	00	00	0.0	00	0.0
000000480	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000496	00	0.0	00	00	00	00	00	0.0	00	00	00	0.0	00	00	55	AA

13. Now that we learned how to read the partition table, let us move on to the file system. We learned from the partition table that the starting sector for the file system is sector 63. Since each sector is 512 bytes, let us multiply 512 by 63 to get the offset. The result should be 32256.

Calculator				
≡ Pro	gramme	r	_	
				53 × 512 =
			32 2	256
HEX 7E0	0			
DEC 32,	256			
OCT 77	000			
BIN 011	1 1110 0000 0	000		
ų.	88	QWORD		MS
:D+ Bitwise	~ 💥 Bit	Shift \sim		
A	~~	**	CE	(3)
В	()	%	÷
с	7	8	9	×
D	4	5	6	
E	1	2	3	+
F	+/_	0		=



14. As we did earlier, open the Go to window by clicking the Search dropdown menu option and then clicking Go to from the dropdown menu or press Ctrl+G. Once the window appears, type 32256 in the text box as highlighted as item 1 below. Verify that the radio button beside dec is still selected, and then click OK as seen in item 2 below.



15. Now that you are at sector 63, you will be looking at the text and hexadecimal representation of the volume boot record (VBR). The offset 32256 is the location of the first byte in sector 63 (VBR) on this volume. Since we know that each sector is 512 bytes, we can determine that the sector ends at offset 32767 by adding the number of bytes after the first byte (511) and the offset of the first byte (32256). Certain artifacts are located at specific byte offsets within the VBR.

Offset(d)	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	Decoded text	<u>^</u>
000032256	EB	30	90	4D	53	44	4F	53	35	2E	30	00	02	02	06	00	e<.MSD055.0	Sector 63
000032272	02	00	02	00	00	F8	9D	00	ЗF	0.0	FF	0.0	ЗF	00	00	0.0	ø?.ÿ.?	
000032288	86	39	01	0.0	80	0.0	29	6E	7D	54	20	4E	4F	20	4E	41	†9€.)n}T NO NA	
000032304	4D	45	20	20	20	20	46	41	54	31	36	20	20	20	33	09	ME FAT16 <mark>3</mark> É	
000032320	8E	D1	BC	F0	7B	8E	D9	88	00	20	8E	00	FC	8D	00	70	ŹÑ48{ŹŮ,. ŹÀü%.	
000032336	38	4E	24	7D	24	88	C1	99	ЕS	30	01	72	1C	83	EΒ	ЗA	8N\$}\$∢Å™è<.r.fë:	
000032352	66	A1	1C	70	26	66	ЗB	07	26	8A	57	$_{\rm FC}$	75	06	80	CA	f;. &f.&ŠWüu.€È	
000032368	02	88	56	02	80	C3	10	73	EΒ	33	C9	8A	46	10	98	F7	.^V.€Ā.sē3ÉŠF."÷	
000032384	66	± 6	03	46	1C	13	56	$\pm E$	03	46	0E	13	D1	88	76	11	£FVFÑ<∀.	
000032400	60	89	46	\mathbb{PC}	89	56	FE	88	20	0.0	F7	E6	SВ	SE	0B	03	"%Fü%Vp, .÷æ<^	
000032416	C3	48	F7	FЗ	01	46	FC	11	4E	FE	61	BF	00	0.0	E8	E6	ĀH÷ó.Fü.Npa¿èæ	
000032432	00	72	39	26	38	2D	74	± 7	60	81	0B	BE	Al	7D	FЗ	A6	.r9&8-t.'±.%;}ó;	
000032448	61	7.4	32	4E	74	09	83	C7	20	38	FB	72	E6	E8	DC	AO	at2Nt.fÇ ;ûræëÜ	
000032464	FB	7D	Β4	7D	SВ	FO	AC	98	40	7.4	0C	48	74	13	Β4	OE	û}'}<ð⊣"@t.Ht.'.	
000032480	BB	07	00	$^{\rm CD}$	10	E8	ΕF	AO	FD	7D	EΒ	E6	A0	$_{\rm FC}$	7D	EB	»ĺ.ēĭ ý}ēæ ü}ē	
000032496	E1	CD	16	$^{\rm CD}$	19	26	8B	55	1A	52	B0	01	ΒВ	00	00	E8	áÍ.Í.&‹U.R°.»è	
000032512	ЗB	00	72	E8	5B	8A	56	24	ΒE	08	7C	88	FC	67.	46	FO	;.rè[ŠV\$¾. <üÇFð	
000032528	ЗD	7D	C7	46	F4	29	7D	80	D9	89	4E	F2	89	4E	F6	C6	=}CFô)}CÚ%Nô%NÖÆ	
000032544	06	96	7D	08	ΕA	03	00	0.0	20	OF	Β6	08	66	88	46	F8	}Éê¶Éf <fø< th=""><th></th></fø<>	
000032560	66	03	46	10	66	88	DO	66	C1	EΑ	10	E8	5E	OF	B6	C8	f.F.f<ðfÅê.ë^.¶É	
000032576	4A	4A	SA	46	0D	32	Ε4	E7	E2	03	46	$_{\rm EC}$	13	56	FE	E8	JJŠF.2ä÷â.Fü.Vþē	
000032592	4A	52	50	06	53	6A	01	6A	10	91	SВ	46	18	96	92	33	JRP.Sj.j.' <f'3< th=""><th></th></f'3<>	
000032608	D2	E7	F6	91	F7	F6	42	87	CA	F7	76	1A	SA	F2	SA	E8	Ó÷ö`÷öB‡É÷⊽.SòSè	
000032624	CO	CC	02	ΟA	CC	88	01	02	80	7E	02	ΟE	75	04	Β4	42	ÀÍÍ,€~u.′B	
000032640	SВ	F4	SA	56	24	CD	13	61	61	72	0B	40	75	01	42	03	<ôŠV\$1.aar.@u.B.	
000032656	5E	08	49	75	06	F8	C3	41	BB	00	00	60	66	6A	00	EΒ	^.Iu.øÄA≫'fj.ë	
000032672	B0	42	4F	4F	54	4D	47	52	20	20	20	20	OD	ΟA	52	65	°BOOTMGRRe	
000032688	6D	6F	76	65	20	64	69	73	6B	73	20	6F	72	20	6F	74	move disks or ot	
000032704	68	65	72	20	6D	65	64	69	61	2E	FF	OD	0A	44	69	73	her media.ÿDis	
000032720	6B	20	65	72	72	6E	72	FF	0Đ	ΟA	50	72	65	7.3	73	20	k errorÿPress	
000032736	61	6E	79	20	6B	65	79	20	74	6F	20	72	65	7.3	74	61	any key to resta	
000032752	72	74	0D	AO	00	00	00	00	00	00	00	AC	CB	D8	55	AA	rtËØU=	



16. The table below provides the location of the artifacts based on their offset values for the entire volume as well as for the sector. During this exercise, we will use data from the Dec and Universal Offsets columns to locate the relevant entries in the VBR located at sector 63.

0	offset	Univ	ersal	Len	Name	Description
Hex	Dec	Offs	sets	gth (By tes)		
0x7E00	32256	0x00	0	3	Jump Instruction	Jump instructions to skip to boot code field
0x7E03	32259	0x03	3	8	OEM ID	ASCII - MSDOS5.0
0x7E0B	32267	0x0B	11	2	Bytes per Sector	Combined will provide Cluster size
0x7E0D	32269	0x0D	13	1	Sectors per Cluster	
0x7E20	32288	0x20	20	8	Total Sectors on Volume	Volume size
0x7E15	32277	0x15	21	1	Media Descriptor	Common values are 0xF8 and 0xF0 which represents fixed media and removable media, respectively
0x7E27	32295	0x27	39	4	Volume Serial Number	Serial number of the Volume
0x7E36	32310	0x36	54	8	FAT16	File System Type
0x7FFE	32766	0x1FE	51 0	2	Boot Signature	0x55 AA



Normally, the universal offsets would be used to find each section within the Sector 63, however, HxD will display the entire physical disk which will contain sector 63.



17. Let us highlight the first 3 bytes starting at offset 32256. These bytes are known as the Jump instruction, and it instructs the computer to skip over the next few bytes as they are not executable.

Offset(d)	00 01	02 0	3 04	05	06 0	07 08	09	10	11	12	13	14	15	Decoded text	^
000032256	BB 3C	90 4	D 53	4.4	4F 3	3 35	2E	30	00	02	02	06	0.0	MSDOS5.0	Sector 63
000032272	02 00	02 0	0 00	F8	9D (00 3F	00	FF	00	ЗF	00	00	00	ø?.ÿ.?	
000032288	86 39	01 0	0 80	00	29 (E 71	54	20	4E	4F	20	4E	41	†9€.)n}T NO NA	
000032304	4D 45	20 2	0 20	20	46	1 54	31	36	20	20	20	33	09	ME FAT16 <mark>3</mark> Ê	

18. Highlight the next 8 bytes immediately after the Jump instruction. These bytes are located at offset 32259 in our FEF or the 3rd byte from the beginning of the sector. These 8 bytes are the OEM ID and will tell you the name of the file system when converted to text. As seen in the screenshot below, the highlighted text is MSDOS5.0, which indicates that it is a FAT file system.

Offset(d)	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	Decoded text	^
000032256	EB	30	90	4D	53	44	4E	53	35	20	30	00	02	02	06	00	e< MSDOS5.0 Sector 63	3
000032272	02	00	02	00	00	F8	9D	00	ЗF	00	FF	00	ЗF	00	00	00	ø?.ÿ.?	
000032288	86	39	01	00	80	00	29	6E	7D	54	20	4E	4F	20	4E	41	†9€.)n}T NO NA	
000032304	4D	45	20	20	20	20	46	41	54	31	36	20	20	20	33	09	ME FAT16 <mark>3</mark> Ê	

19. Let us highlight the next 2 bytes after the OEM name as indicated below. These bytes are located at offset 32267 in our FEF or the 11th byte from the beginning of the sector. The highlighted bytes are 0x00 02, and these bytes indicate the number of bytes per sector. When converted to decimal, the number of bytes is 512.



20. Now highlight the byte immediately beside the previous one as indicated below. This byte is located at offset 32269 in our FEF or the 13th byte from the beginning of the sector. This value is the number of sectors per cluster and, as seen below, is 0x02 or 2 in decimal. Combining this value with the number of bytes per sector (512) can provide the cluster size for the volume.

Offset(d)	00	01	02	03	04	05	06	07	80	09	10	11	12	13	14	15	Decoded text	^
000032256	EB	30	90	4D	53	44	4F	53	35	2E	30	00	02	02	06	00	ë<.MSDO55.0 Sector 63	3
000032272	02	00	02	00	00	F8	9D	00	ЗF	00	FF	00	ЗF	00	00	00	ø?.ÿ.?	
000032288	86	39	01	00	80	00	29	6E	7D	54	20	4E	4F	20	4E	41	†9€.)n}T NO NA	
000032304	4D	45	20	20	20	20	46	41	54	31	36	20	20	20	33	09	ME FAT16 <mark>3</mark> É	





21. Now that we know the size per cluster, let us look at the media type. This can tell whether the volume is a removable/fixed disk, a floppy disk, or other type of medium. The media type is located at offset 32277 in our FEF or the 21st byte from the beginning of the sector. Let us use Go to again; it should be open already; if not, reopen it and type 32277 and click OK. This will take you to the beginning of the media type entry. Highlight the next byte, as seen below. The hex value F8 represents a hard disk or removable drive (USB drive etc.).

000032256	EB 3C 90 4D 53 44 4	Go to X	ë<.MSDOS5.0 Sector 63
000032288	86 39 01 00 80 00 2 4D 45 20 20 20 20 4	Offset:	t9€.)n}T NO NA ME FATI6 3É ΣNLAJZH 23HL
000032336	38 4E 24 7D 24 8B 0 66 Al 1C 7C 26 66 3	⊖ hex	8N\$)\$<Á™é<.r.fë: f;. &f.&ŠWüu.€Ê
000032368 000032384	02 88 56 02 80 C3 1 66 16 03 46 1C 13 5	Offset relative to	. *V.€Å.së3ËŠF."÷ fFVFŇ<∀.
000032400	C3 48 F7 F3 01 46 F 00 72 39 26 38 20 7	○ current offset ○ end (backwards)	*#Fu%Vp, +æ<^ ÄH+ö.Fü.Npa¿èæ
000032448	61 74 32 4E 74 09 8 FB 7D B4 7D 8B F0 2		at2Nt.fÇ ;ûræëÜ û}`}<ð-"@t.Ht.'.
000032480 000032496	BB 07 00 CD 10 EB E E1 CD 16 CD 19 26 5	3B 55 1A 52 B0 01 BB 00 00 E8	»İ.ëï ý}ë≇ ü}ë áİ.İ.& <u.r°.≫è< td=""></u.r°.≫è<>

22. Let us jump to offset 32288 or the 32nd byte from the beginning of the sector. You will find the total number of sectors on the volume here. Let us use Go to locate the value. Reopen it by clicking the Go to option from the Search dropdown menu option on the menu bar or using Ctrl+G. In the Go to window, type 32288 and click OK. You will be taken to the offset 32288.

Offset:		
		5228
Ohex	() dec	Ooct
Offset rel	ative to	
begin		
() curren	it offset	



23. Let us highlight the next 4 bytes, as seen below. This value is represented as 0x86 39 01, which when converted to decimal is 80262. This indicates that there are 80262 sectors. With this information, we can determine the size of the volume. The size can be found by multiplying the number of sectors (80262) by the sector size (512), which is equal to 41,094,144 bytes (41MB).

24. Next, let us look at the volume serial number. This can be found at the 39th, byte from the beginning of the sector, or offset 32295 in our FEF. Use Go to by browsing to the Search dropdown menu on the menu bar and clicking Go to or using Ctrl+G. Type 32295 and click OK. Highlight the next 4 bytes, as seen below.

000032256	EB 3C 90 4D 53 44 4F 53 35 2E 30 00 g Goto X	Sector 63
000032272	52 00 52 00 55 SD 00 3F 00 FF 00 3	
000032288	86 39 01 00 80 00 29 52 70 54 20 4E Offset:	
000032304	4D 45 20 20 20 20 46 41 54 31 36 20 2	
000032320	SE D1 BC F0 7B SE D9 BS 00 20 SE C0 E	
000032336	38 4E 24 7D 24 8B C1 99 E8 3C 01 72 1 Oher Oder Oort	
000032352	66 A1 1C 7C 26 66 3B 07 26 8A 57 FC 7 Onex Could	
000032368	02 88 56 02 80 C3 10 73 EB 33 C9 8A 4 Offset relative to	
000032384	66 16 03 46 1C 13 56 1E 03 46 0E 13 1	
000032400	60 89 46 FC 89 56 FE 88 20 00 F7 E6 5	
000032416	C3 48 F7 F3 01 46 FC 11 4E FE 61 BF 0 Current offset	
000032432	00 72 39 26 38 2D 74 17 60 B1 0B BE A Oend (backwards)	
000032448	61 74 32 4E 74 09 83 C7 20 38 FB 72 E	
000032464	FB 7D 84 7D 88 F0 AC 98 40 74 0C 48 7	
000032480	BB 07 00 CD 10 EB EF A0 FD 7D EB E6 A	
000032496	E1 CD 16 CD 19 26 8B 55 1A 52 B0 01 Bp 00 00 10 AL.I.4(U.Ke	
000032512	3B 00 72 E8 5B 8A 56 24 BE 0B 7C 8B FC C7 46 F0 ;.re(ŠV\$%.)«üÇF8	

The volume serial number is created when the drive is formatted and can be used to determine if files were ever stored on the drive. The Volume Serial Number byte order is in little-endian, which means it is read from right to left 2054-7D6E.



25. Let us look at the Filesystem type. As the name suggests, this artifact will tell what type of filesystem is on the volume. The Filesystem type is located at offset 32310 in our FEF or the 54th byte from the beginning of the sector. Let us use Go to again; it should be open already; if not, reopen it and type 32310 and click OK. This will take you to the beginning of the Filesystem type entry. Highlight the next 8 bytes, as seen below. As you can see in the textual version of the highlighted text, the Filesystem type is FAT16.

Offset(d) 00 01 02 03 04 05	6 06	07	80	09	10	11	12	13	14	15	Decoded text
Go to X	4F	53	35	2E	30	00	02	02	06	00	ë<.MSDOS5.0 Sector 63
200460 ** *	9D	0.0	ЗF	00	FF	0.0	ЗF	00	00	00	ø?.ÿ.?
Offset:	29	6E	7D	54	20	4E	4F	20	4E	41	t9€.)n)T NO NA
32316	46	41	54	31	36	20	20	20	33	69	ME FAT16 SÉ
	D9	88	00	20	θE	C0.	FC	BD	00	73	ŽŇ486{ZU ZAü≒.
⊖hex ⊚dec ⊖oct	C1	99	ΕS	30	01	72	1C	83	EΒ	ЗA	8N\$}\$<Å™è<.r.fë:
	ЗB	0.7	26	8A	57	\mathbf{FC}	75	06	80	CA	f;. &f.&ŠWüu.€È
Offset relative to	10	73	EΒ	33	C9	8A	46	10	98	E7	.^V.€Ã.së3ÉŠF."÷
begin	56	1E	03	46	0E	13	D1	88	76	11	£FVFÑ<∀.
O current offset	FE	88	20	00	F7	E6	8B	SE	0B	03	"‰Fü‰Vp÷æ<^
	FC	11	4E	ΕE	61	BF	00	0.0	ЕS	E6	ĀH÷ó.Fü.Nþa¿èæ
O end (backwards)	74	17	60	81	0B	BE	Al	7D	FЗ	A6	.r9&8-t.'±.%;}ó;
	83	$\odot 7$	20	38	FB	72	E6	E8	DC	AO	at2Nt.fÇ ;ûræëÜ
OK Cancel	AC	98	40	7.4	0C	48	74	13	Β4	OE	û}'}<ð"@t.Ht.'.
	EF	AO	FD	7D	EΒ	E6	A0	$_{\rm FC}$	7D	ΕB	»Í.ëĭ ý}ēæ ü}ē
000032496 E1 CD 16 CD 19 20	SB	55	1A	52	B0	01	BB	00	00	E8	áÍ.Í.& <u.r°.»è< td=""></u.r°.»è<>

- 26. The artifacts we covered were only some of the things that are stored in the FAT file system. Each version of the file system stores data at a slightly different location. When manually reviewing the MBR and VBR as we just did, be sure to use a chart or guide to identify the necessary artifacts associated with the file system you are examining.
- 27. Let us move on to another type of file system.



3 Identifying File System Data in an NTFS Formatted Evidence File

NTFS stands for New Technology File System and is a file system created by Microsoft; it is the file system used with most versions of Microsoft Windows. In this exercise, we will review the artifacts you can find in the NTFS VBR.

1. Let us use HxD to review the FEF and learn how to read the data in the partition table. You should still have HxD open; if not, reopen it and click the Open disk image option from the Tools dropdown menu, as seen in items 1 and 2 in the screenshot below.

) 🤷 🕶 🔛 📗 🕮 🎯 🕶 🕫 🔳 Open ma	in memory Shift+Ctrl+M 🛛 🖉 🛛 🐗	▶ ▶ Sector 0 👙		
Open dis	c Shift+Ctrl+D	Special editors		
Upen dis	timage Shift+Ctrl+1	Data inspector		
File tools	• • •			
Options		14 4 9 91		
		Binary (8 bit)	Invalid	^
		Int8	go to: Invalid	
		UInt8	go to: Invalid	
		int16	go to: Invalid	
		Uint16	go to: Invalid	
		Int24	go to: Invalid	
		Uint24	go to: Invalid	
		Int32	go to: Invalid	
		UInt32	go to: Invalid	
		Int64	go to: Invalid	
		UInt64	go to: Invalid	
		AnsiChar / char8_t	Invalid	
		WideChar / char16_t	Invalid	
		UTF-8 code point	Invalid	
		Single (float32)	Invalid	
		Double (float64)	Invalid	
		OLETIME	Invalid	
		FILETIME	Invalid	
		DOS date	Invalid	
		DOS time	Invalid	
		DOS time & date	Invalid	
		time t (32 bit)	Invalid	
		Byte order		
		 Little endian 	🔿 Big end	dian
		Show integers in hexa	adecimal base	
Checksum Search				
				 ✓ Refresh
Algorithm Checksum	Usage			
The state of the second	1.			
<u> </u>				



2. The Open disk image window will appear. Use this window to browse to This PC > Desktop and double-click the folder Toolbox > Datasets > Lab5. This will open the folder revealing 3 FEFs. Select the file called NDG NTFS Lab5.001 and click the Open button as highlighted below. The NTFS image file will now be loaded in HxD.

Open disk image		×	
🕈 🧾 🛛 Toolbex: > Datasets > Lab5 🛛 🛩 🔇) Search LabS 🖉		
Diganize - New folder	10 · 01 (Special editors
Pictures d Name	Date modified	TVF	Data inspector
Toolbor -		111	14 4 5 51
Si NDG eshAr Labb.Q0	6V30/2020-3649 PM	SHALL	Binary (Bhiti Invalid A
This PC	\$730/2010-9649 PM	1D	Int8 go to: Invalid
Desktop 2 NDG FAT Lab5.001	\$/30/2020 SH46 PM	100-	Ulat8 go to; Invalid
Toolbox	6/30/2020 3h46 PM	To	Int16 go.tb; Invalid.
NDG NTFSLab5.001	6/30/2020-9:51 PM	300-	Unt16 goite: Invalid
Autoruns for Windows NDG NTFS Lab5.001.txt	£/30/2020 952 PM	Tes	Int24 go.to: Invalid
👻 Datasets [4			Unt24 go to: Invalid
CP i			Int32 go to; Invelid
Lab?			Uint32, go to: Invalid
			Int64 go.to: Invalid
Lab4			Uint64 goto: Invalid
Lab5			AnsiChar / cha Irivalid
> deft-8.2-002			WideChar / ch Invalid
Magnet Process Cantu			ITTE-R code no lecalid
integration of the second second second second second second second second second second second second second s		>	Byte order
	T in a second	# ¹¹	Little endian () Big endian
File name	Open Cancel		Sbow integers in hexadecimal
			✓ Refresh
Algorithm Checksum	Usage		
Einacted ranith			



3. As we did before, let us begin by reviewing the Master Boot Record (MBR – Partition Table) in the first sector to locate the VBR. As we learned in the previous exercise, the first partition entry is located at offset 446. Since we will be using decimal to go to offset 446, we will need to change the Offset base to decimal. To do this, click the View dropdown menu option from the menu bar and hover over the Offset base option, then select Decimal as highlighted in items 1, 2, and 3 below.

	abs (V		Ad Byt	apt ti æs pr	o wir er rov	ndow w	wid	th	do	ws (/	ANS))	100	d	ec	1	9 14	4	Þ	P-1	Sector	<u>9</u>
- He Children			Tex	t end	odin	g		2					_									
oriset(a)	00		Off	aet b				4.		He	xade	cima	Ē.	13	14	15	Deco	aed	L CE	exc		
0000000000	FA	-	Vis	ible r	olun	nns	_	- 5		10e	cima	-		3	FB	FC	U3 Å2	D4.	114	P.P	. ûŭ	Sector
000000016	BF		8.4			ite			-	Oc	Fal			-	18	00	6		¥é.		İ.,	
000000032	00	-	- Oyu	a gru	odb 3	99 1		0.5	1	1.945	ene.			100	00	00	00.01		200		2222	
000000048	00		Too	olbar	\$				10	00	00	00	00	00	00	00		• • •	• • •			
00000064	00	3	0.2	es no				rc.	10	00	00	00	00	00	00	00						
080000000	00	1	her	resn	1			10	10	00	00	00	00	00	00	00			100			
000000096	00	00	00	00	00	00	80	60	00	00	00	00	00	60	00	00	0.00	• • •	•••	• • •		
000000112	00	00	00	00	00	DG	00	00	00	00	00	80	00	60	00	88		•••		•••		
00000128	00	00	00	00	00	00	00	00	00	00	00	00	00	80	00	00	52.53				1.2.2	
000000144	00	00	00	00	00	80	00	60	00	00	00	00	00	00	00	86				•••		
000000160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				•••		
000000176	00	00	00	00	00	00	00	00	00	0.0	00	00	00	00	00	00		•••		• • •		
00000192	00	00	00	00	00	00	op	00	00	00	00	00	00	00	00	00		•••		• • •		
000000208	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0.000		•••			
000000224	00	80	60	00	00	00	00	00	00	00	00	00	00	00	00	00	5265	0.05	833	100	10.00	
000000240	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	10103				1.7.5	
00000256	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		• • •	•••	••••	· · ·	
00000272	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00						
000000288	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	23.23					
000000304	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	1.1.1.1	***	•••			
00000320	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00			•••	• • •		
000000336	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	12.52		333	antes		
000000352	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00						

4. Now that the offsets are in decimal, let us use Go to Ctlg jump to offset 446. This can be opened by going to the Search dropdown menu option Ctrl+G in the menu bar and then clicking Go to from the dropdown menu or pressing Ctrl+G. When the Go to window appears, type 446 in the text box, verify that dec is selected, and then click OK as seen below.

🗋 🖻 🕶 🗐 🖉	Find					Ctrl	+F	do	ws (A	ANSI)	~	d	ec	-	14	4 1	▶ Secto	r	1
NDC EATI	Replac	e				Ctrl	+R	E												
NDGRATE	Find ag	gain					F3	Ŀ.												
Offset(d	Find ag	gain (rever	sed)	S	hift+	F3	2	09	10	11	12	13	14	15	Decod	ed te	xt		1
00000014	Go to.					Ctrl	+G	4	00	G	io to						×			
00000016		- 11						0	00	0										
000000176	00 00	00	00	00	00	00	00	00	00	6	Offse	t;	_	_				3		
000000192	00 00	00	00	00	00	00	00	00	00	C							446			
000000208	00 00	00	00	00	00	00	00	00	00	04			-			1993		-		
000000224	00 00	00	00	00	00	00	00	00	00	Q	Oh	eх	2	🖲 de	ec	Ooct				
000000240	00 00	00	00	00	-00	00	00	00	00	0	1.21003									
000000256	00 00	00	00	00	00	00	00	00	00	C	Offs	et re	lativ	e to						
000000272	00 00	00	00	00	00	00	00	00	00	0	٥b	egin								
000000288	00 00	00	00	00	00	00	00	00	00	0	Oc	urrer	it of	fset						
000000304	00 00	00	00	00	00	00	00	00	00	0	0.	nd (l		ward	1					
000000320	00 00	00	00	00	00	00	00	00	00	C	U e	nun	JOCK.	00010	5/					
000000336	00 00	00	00	00	00	00	00	00	00	C			-			16				
000000352	00 00	00	00	00	00	00	00	00	00	C				0	К	Ca	ncel			
000000368	00 00	00	00	00	00	00	00	00	00	C			-			1	12272			
000000384	00 00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
000000400	00 00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
000000416	00 00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
000000432	00 00	00	00	00	00	00	00	04	41	11	00	00	00	B0	01		· · · · I	<i>⊾</i> ق.		
000000448	01 00	07	FE	ЗF	-04	ЗF	00	00	00	86	39	01	00	00	00	þ?	. ?	t9		
000000464	00 00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
000000480	00 00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
000000496	00 00	00	00	00	00	00	00	00	00	00	00	00	00	55	AA			U*		
000000512	00 00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				Sector	1
000000528	00 00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
000000544	00 00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
000000560	00 00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
000000576	00 00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
000000592	00 00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
000000608	00 00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
000000624	00 00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
000000640	00 00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
000000656	00 00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					



5. Your cursor will be taken to offset 446. Let us highlight the 16 bytes after the cursor, as seen in item 1 below. You can use the status bar at the bottom of the main window to count the length of your selection, as highlighted in item 2 below.

🗋 🙆 🕶 🔚 🛛	8.5	3	1	E	16	5	~	W	indo	ws (4	NSI)		1	d	lec		1	4 4	Þ	₽1	Sec
🔛 NDG FAT Lab	5.001	PD ad	NE)G N	TFS (ab5.	001														
Offset(d)	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	Dec	ode	i te	at	
000000144	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
000000160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
000000176	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
000000192	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
000000208	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
000000224	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
000000240	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
000000256	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
000000272	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
000000288	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
000000304	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
000000320	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
000000336	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
000000352	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
000000368	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
000000384	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
000000400	00	00	00	00	00	00	00	00	00	00	00	00	00	00	10	00					
000000416	00	00	00	00	00	00	00	00	00	00	00	00	00	00	- 10	00					
000000432	00	00	00	00	00	00	00	00	04	41	п	00	00		80	01			1		
00000044	01	00	97	EE	3F	04	3F	00	80	00	86	3.9	01	00	00	90	200	þ?.	tion of	191	
000000464	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
000000480	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
000000496	00	00	00	00	00	00	00	00	00	00	00	00	00	00	55	AA					.Uª
)ffset(d): 446			В	ock	d): 4	46-4	51							6	ena	th(d):	16				

6. As we did with the partition entry above, let us break it down to learn what type of file system the entry is referring to and what sector the VBR is located in.



The highlighted value is 0x80 | 0x01 01 00 | 0x07 | 0xFE 3F 04 | 0x3F 00 00 00 | 0x86 39 01 00

- 7. The only difference between this entry and the previous FAT entry is that the 5th byte that represents file system type is 07, which is used to represent NTFS/exFAT file systems. The starting sector of this file system (where the VBR is located) is the same as before. When converted, it is 63. Let us jump to sector 63.
- 8. Since we know that each sector is 512 bytes, we can determine that the sector ends at offset 32,767 by adding the number of bytes after the first byte (511) and the offset of the first byte (32256). Certain artifacts are located at specific byte offsets within the VBR. Like before, we will provide the location of the artifacts based on their offset values for the entire volume as well as for the sector.
- 9. The table below provides the location of the artifacts based on their offset values for the entire volume as well as for the sector. During this exercise, we will use data from the Dec and Universal Offsets columns to locate the relevant entries in the VBR located at sector 63.



10. Since the calculations for this file system are the same as the FAT file system we reviewed earlier, we already know that the starting sector is located at offset 32256 in our FEF. Let us reopen Go to by clicking the Search dropdown menu option, and clicking Go to from the menu, or typing Ctrl+G. In the Go to window, type 32256 and click OK as highlighted below.

		B2256
) hex	● dec	Ooct
Offset rel	ative to	
egin		
() curren	t offset	
🔿 end (b	ackwards)	
6		_

O t	ffset	Univ	ersal sets	Lengt	Name	Description
Hex	Dec			(Byte s)		
0x7E00	32256	0x00	0	3	Jump Instruction	Jump instructions to skip to boot code field
0x7E03	32259	0x03	3	8	OEM ID	ASCII – NTFS
0x7E0B	32267	0x0B	11	2	Bytes per Sector	Combined will provide Cluster size
0x7E0D	32269	0x0D	13	1	Sectors per Cluster	
0x7E20	32288	0x28	40	8	Total Sectors on Volume	Volume size
0x7E15	32277	0x15	21	1	Media Descriptor	Common values are 0xF8 and 0xF0 which represents fixed media and removable media, respectively
0x7E48	32328	0x48	72	4	Volume Serial Number	Serial number of the Volume
0x7FFE	32766	0x1FE	51 0	2	Boot Signature	0x55 AA



11. At offset 32256 (Sector 63), begin by highlighting the OEM ID. This is located at offset 32259 in our FEF or the 3rd byte from the beginning of the sector. Highlight the following 8 bytes after 32259 to reveal the OEM ID as highlighted below. As seen in the textual version of the highlighted data, the OEM ID is NTFS, which indicates that it is an NTFS file system.

Offset(d)	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	Decoded text	^
000032256	EB	52	90	łΕ	54	46	5.3	20	20	20	20	00	02	08	00	00	ëR.NTFS	53
000032272	00	00	00	00	00	F8	00	00	ЗF	00	FF	00	ЗF	00	00	0.0	ø?.ÿ.?	
000032288	00	00	00	00	80	00	00	00	85	39	01	00	00	00	00	00	€9	
000032304	10	OD	00	00	00	00	00	00	02	00	00	00	00	00	00	00		
000032320	F6	00	00	00	01	00	00	00	CO	DD	2A	AO	1D	28	A0	\mathbf{FC}	öÂÝ* .+ ü	

12. Highlight the next 2 bytes that follow the OEM ID as highlighted below. These are located at offset 32267 in our FEF or the 11th byte from the beginning of the sector. As seen below, the value is 0x00 02 or 512 bytes when converted to decimal and indicates the number of bytes per sector.

Offset(d)	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	Decoded text			٥
000032256	EB	52	90	4E	54	46	53	20	20	20	20	00	02	08	00	00	ëR.NTFS	Sector	63	
000032272	00	0.0	00	0.0	00	FS	00	0.0	ЗF	0.0	FF	00	ЗF	00	00	00	ø?.ÿ.?			
000032288	00	0.0	00	0.0	80	0.0	00	0.0	85	39	01	00	00	00	00	00	€9			
000032304	10	0D	00	00	00	00	00	00	02	00	00	00	00	00	00	00				

13. Now highlight the byte immediately beside the previous one, as seen below. This byte is located at offset 32269 in our FEF or the 13th byte from the beginning of the sector. This value is the number of sectors per cluster and, as seen below, is 0x08 or 8 in decimal. Combining this value with the number of bytes per sector can provide the cluster size for the volume.

Offset(d)	00	01	02	03	04	05	06	07	80	09	10	11	12	13	14	15	Decoded text	1	5
000032256	EB	52	90	4E	54	46	53	20	20	20	20	00	02	08	00	00	ER.NTFS	63	
000032272	00	00	00	00	00	F8	00	00	ЗE	00	ΕE	00	ЗE	00	00	00	ø?.ÿ.? <mark>.</mark>		
000032288	00	00	00	00	80	00	00	00	85	39	01	00	00	00	00	00	€9		
000032304	10	0D	00	00	00	00	00	00	02	00	00	00	00	00	00	00			



In this case, 512 bytes per sector * 8 sectors = size of each cluster 4096 bytes.



14. Let us jump to offset 32296 or the 40th byte from the beginning of the sector. In an NTFS VBR, you will find the total number of sectors on the volume here. Let us use Go to, in order to locate the value. Reopen it by clicking the Go to option from the Search dropdown menu option on the menu bar. In the Go to window, type 32296 and click OK. You will be taken to the offset 32296.

11966		3229
hex) dec	Ooct
Offset rel	ative to	
🖲 begin		
() curren	t offset	
	- [abarrada]	

15. Let us highlight the next 8 bytes as highlighted below. This value is represented as 0x85 39 01 00 00 00 00, which when converted to decimal is 80261. This indicates that there are 80261 sectors in the volume. The size of the volume in bytes can be found by multiplying the number of sectors (80261) by the sector size (512), which is equal to 41,093,632 bytes (41MB).

NDG NTES Labs	001										Special editors			2
1400 1411 5 2005	4001									14	Special cultors			
ffset(d) 0	0 01 0	2 03 04	05 06	07 08	09 10	11 1:	2 13 14	4 15	Decoded text		Data inspector			
Calculator		-	Ω ×	00 00	00 00	00 00	0 00 00	0 00			14 4 5 51			
■ Standard	93		3	00 00	00 00	00 00	0 00 00	0 00			Bay Charles and	_		
	1		261 - 512 -	00 00	00 00	00 01	0 00 01	0 00			Binary (8 bit)		10000101	^
				00.00	00 00	00 04	0 00 01	0 00			Int8	go to:	-123	_
	41	.093.	632	00 00	00 00	00 01	0 00 00	0 00			UInt8	go to:	133	
				0 00	00 00	00 01		0 00			Int16	go to:	14725	
AC 40	M+	M- MS	AC.	0 00	00 00	00 0	0000	0 00			Uint16	go to:	14725	
1/4	CE	c	G	00 00	00 00	00 00	0 00 00	0 00			Int24	go to:	80261	
				0 00	00 00	00 00	0 00 0	0 00			UInt24	ao to:	80261	
½x	A ²	44	÷	00 00	00 00	00 00	0 00 0	000			Int32	no to:	80261	
		1421		10 20	20 20	00 03	2 08 00	0 00	ëR.NTFS	Sector 63	Ulint32	no to:	80261	
	8	а	÷.	C 3F	OG FF	00 31	100 00	0 00	Ø?.9.?		int64	no to:	80261	
4	5	6	-	0 07	00.00	00.0	00.00	0.00			Lilothi	an to:	80261	
				A E9	FE 01	90 91	66 60	0 1E	*ö.üó*éþf`.		Bute order	90.000	1.00001	
1	2	3	+	16 1C	00 1E	66 61	0 00 8	0 00	.f;ffh		C Lible andian		OPie realize	
			-	0 68	10 00	B4 43	2 SA 10	6 0E	.fP.Shh BŠ		Cittle endian			
*/-	0	್ರಾ	=	56 59	58 5A	66 5	66 55	9 1F	«ôİ.fY[ZfYfY.		Show integers in he	vadacima	I hace	

The byte order of the values represented as 0x85 39 01 00 00 00 00 00 is little-endian, which mean the values are reversed 0x00 00 00 00 01 39 85. The preceding zeroes are dropped leaving 0x01 39 85 which converts to 80261.



16. Next, let us identify the starting extent for the Master File Table (MFT). This is located at offset 32304 in our FEF or the 48th byte from the start of the sector. Let us use Go to by browsing to the Search dropdown menu on the menu bar and clicking Go to or using Ctrl+G. Type 32304 and click OK. Highlight the following 8 bytes as seen below. These bytes are 0x10 0D 00 00 00 00 00 00 and represent 3344 when converted to decimal. This number is the starting cluster of the MFT. An examiner would be able to browse to the specific cluster and extract or view the contents of the MFT with this information.

🔛 File Edit S	earch 🎒 Analysis 1	ools Window Help					_ 8' ×
1 - H	Find	Ctrl+F dows (A	ANSI)	dec	🗑 🛛 🕊 🕨 🕨 Sector 6.	i3 🤤 of 251,904	
INDG NTFS	🔓 Replace	Ctrl+R				Special editors	×
Offset(d	Find again Find again (reversed)	F3 Shift+F3 18 09	10 11 12 1	3 14 15	Decoded text	Data inspector	
00003208	- Go to	Ctrl+G	00 00 00 0	0 00 00		14 4 6 6	
000032090	00 00 00 00 00	00 00 00 00 00	00 00 00 00 0	0 00 00		Rieson (R. Nith)	00010000
000032128	00 00 00 00 00	00 00 00 00 00	00 00 00 0	0 00 00		Land Land	16
000032144	00 00 00 00 00	00 00 00 00 00	00 00 00 0	0 00 00		ints go to:	10
000032160	00 00 00 00 00	00 00 00 00 00	00 00 00 0	0 00 00		Uint8 go to:	16
000032176	00 00 00 00 00	00 00 00 00 00	00 00 00 0	0 00 00		int16 go to:	3344
000032192	00 00 00 00 00	00 00 00 00 00	00 00 00 0	0 00 00		Uint16 go to:	3344
000032208	00 00 00 00 00	00 00 00 00 00	00 00 00 0	0 00 00		Int24 go to:	3344
000032224	00 00 00 00 00	00 00 00 00 00	00 00 00 0	0 00 00		UInt24 go to:	3344
000032240	00 00 00 00 00	00 00 00 00 00	00 00 00 0	0 00 00		Int32 go to:	3344
000032256	EB 52 90 4E 54	46 53 20 20 20	20 00 02 0	8 00 00	ëR.NIFS Sector 63	Ulat22 ao to:	2244
000032272	00 00 00 00 00	F8 00 00 3F 00	FF DO SF D	0 00 00	ø?.ÿ.?	Guilde go to.	2244
000032288	00 00 00 00 80	00 00 00 85 39	01 00 00 0	0 00 00	····€···9·····	into4 go to:	3344
000032304	10.0D.00.00.00	00 00 00 02 00	00 00 00 0	0 00 00		Uint64 go to:	3344
000032320	F6 00 00 00 01	ou oo ou co bb	2A A0 1D 2	B AO FC	0AY* .+ u	AnsiChar / char8_t	0
000032336	00 00 00 00 FA	33 CO SE DO BC	00 7C FB 6	8 CO 07	u3AZĐ4. JuhA.	WideChar / char16_t	ഐ
000032352	IF 1E 68 66 00	CH BS 16 UE UU	66 31 3E U	3 00 m	ata Y	UTF-8 code point	□ (U+0010)
000032366	54 46 53 /5 15 EE 37 7E 06 F7	C1 01 00 75 02	E0 13 72 0	E 83 E	010	Single (float32)	4.68594206470219E-42
000032304	19 69 17 00 R4	49 88 16 0F 00	98 F4 16 1	E CD 1	Offset:	Double (float64)	1 65215551969313E-320
000032416	9F 83 C4 18 9E	58 1F 72 E1 38	06 DB 00 7	5 DB A	2200	OLETIME	1900-12-20
000032432	OF 00 C1 2E OF	00 04 1E 5A 33	DB B9 00 2	0 2B C	- Advected and a	EU ETIME	1601-01-01
000032448	66 FF 06 11 00	03 16 OF 00 SE	C2 FF 06 1	6 00 E	⊖hex	FILETIME	1001-01-01
000032464	4B 00 2B C8 77	EF B8 00 BB CD	1A 66 23 C	0 75 2		DOS date	1986-08-16
000032480	66 31 FB 54 43	50 41 75 24 81	F9 02 01 7	2 1E 1	Offset relative to	DOStime	1:40:32 AM
000032496	68 07 BB 16 68	52 11 16 68 09	00 66 53 6	€ 53 €	begin	DOS time & date	Invalid
000032512	55 16 16 16 68	B8 01 66 61 OE	07 CD 1A 3	3 CO B	🔘 current offset	time t (32 bit)	1970-01-01 12:55:44 AM
000032528	0A 13 B9 F6 0C	FC F3 AA E9 FE	01 90 90 6	6 60 1	O end (backwards)	Byte order	
000032544	06 66 A1 11 00	66 03 06 1C 00	1E 66 68 0	0 00 0		little endian	
000032560	00 66 50 06 53	68 01 00 68 10	00 B4 42 S	A 16 0	0	C Entre endidit	Objection
000032576	00 16 1F 8B F4	CD 13 66 59 5B	5A 66 59 6	6 59 1	OK Cancel	Show integers in hexadecima	al base
000032592	OF 92 16 00 66	FF 06 11 00 03	16 OF 00 8	E C2 F			
Offset(d): 32304	Block(d): 32	304-32311		Length(d)	: 8 Overwrite	e	



17. Next, let us look at the volume serial number. This can be found at the 72nd byte from the beginning of the sector or offset 32328 in our FEF. Use Go to by browsing to the Search dropdown menu on the menu bar, and clicking Go to or using Ctrl+G seen in items 1 and 2. Type 32328 and click OK as seen in items 3 and 4. Before going any further, let us click the checkbox beside Show integers in hexadecimal base as seen in item 5. This will change the values in the Data Inspector tab to hexadecimal. Now that you are at the offset 32328 (752nd byte), highlight the next 4 bytes as seen in item 6 below. These bytes represent the volume serial number and are stored in little-endian. When converted, the volume serial number will read A02A-DDC0, as seen in item 7.

🗋 🚵 👻 💭 Find	Ctrl+F	dov	vs (A	NSI)		~	d	ec	1	-		
NDG NTFS Ass Replace	Ctrl+R F3											Special editors
Offset (d Find again (reversed)	Shift+F3)8	09	10	11	12	13	14	15	Decoded text	^	Data inspector
00003217 🚐 Go to	Ctrl+G	70	00	00	00	00	00	00	00			
00003219		-10	00	00	00	00	00	00	00	• • • • • • • • • • • • • • • • • • •		
000032208 00 00 00 00 00 00	00 00	00	00	00	00	00	00	00	00			Binary (8 bit) 11000000
000032224 00 00 00 00 00 00	00 00	00	00	00	00	00	00	00	00	•••••		Int8 go to: -40
000032240 00 00 00 00 00 00	00 00	00	00	00	00	00	00	00	00	•••••		Uint8 go to: C0
000(Go to X	53 20	20	20	20	00	02	08	00	00	er.NTFS		Int162240
000(00 00	3F	00	FF	00	3F	00	00	00	·····ø··?·ÿ·?···		Illente antes DDCO
32328	00 00	85	39	01	00	00	00	00	00	····£····9·····		Unclu go to: DDC0
	00 00	02	00	00	00	00	00	00	00			Int24 go to: 2ADDC0
UUUI Ohex @dec Ooct	00 00	60	DD.	44	ALL.	10	28	AU	FC	0		UInt24 go to: 2ADDC0
OOO(Officet relative to	CU SE	DU	BC	00	76	F B	DO	00	07	usazuwa.juna.		70Int32 go to: -5FD52240
DODI begin	44 DD	UE	00	CD	10	JE	03	00	TP TP	TEGU (ARTÍ E A		UInt32 go to: A02ADDC0
DOD(Current offset	41 00	DE	00	ED.	10	14	100	01	F D	1750. A»-01.1		Int64 go to: Invalid
000(O end (backwards)	OI DD	07	00	00	TP D	16	1 12	CD	40	b		Illint64 as too Involid
	15 72	0E	20	00	1 T	00	75	DP	10	Yel av rát uffe		Data and a
	D4 1F	53	33	DB	BG	00	20	2B	C8	1)A.5A.1d,u0Σ 1 73Πι ⊥Γ		Dyte order
	16 DF	00	SE	C2	FF	06	16	00	FS	fü Žîü é		Little endian O Big endian
000032464 4B 00 2B CS 77 FF	B8 00	BB	CD	14	66	23	cn	75	20	£ +Ê₩ïŤ f#àu_	G	Extension to a superior of the
00000000 60 01 FD 50 11 10 50	11 75	24	01		00	01	20	15	10	f orching of r	5	E show megers in nexadecimal

18. As with the previous file system, there are many artifacts that can be unearthed but will not be covered in this exercise. We will now review the final file system that this lab will cover.



4 Identifying File System Data in an exFAT Formatted Evidence File

The Extensible File Allocation Table (exFAT) file system is becoming very common. It is like FAT but has a larger file capacity limit and can handle data better. As such, it is important to understand how to identify exFAT volumes and learn where to locate their file system artifacts. In this exercise, we will review the artifacts you can find in the exFAT VBR.

1. Let us use HxD to review the FEF and read the data stored in the partition table. You should still have HxD open. If not, reopen it and click the Open disk image option from the Tools dropdown menu, as seen in items 1 and 2 in the screenshot below.

We Had	~ Д	×
Copen main memory Shift+Ctrl+M Shift+Ctrl+M Open disk Shift+Ctrl+D	0 🔅	×
Open disk image Shift+Ctrl+1	Data inconstar	
File tools >	Data inspector	
Options	14 4 E EI	
	Rinary (8 bit) Invalid	~
	int8 go to: invalid	
	Uint8 go to: Invalid	
	Int16 pp to: Invalid	
	Ulnt16 go to: Invalid	
	Int24 go to: Invalid	
	Uint24 go to: Invalid	
	Int32 go to: Invalid	
	Ulnt32 go to: Invalid	
	Int64 go to: Invalid	
	UInt64 go to: Invalid	
	AnsiChar / char8_t Invalid	
	WideChar / char16_t Invalid	
	UTF-8 code point Invalid	
	Single (float32) Invalid	
	Double (float64) Invalid	
	OLETIME Invalid	10
	FILETIME Invalid	
	DOS date Invalid	
	DOS time Invalid	
	DOS time & date Invalid	
	time t (32 bit) Invalid	¥
	Byte order	
	Little endian O Big endian	
	Show integers in hexadecimal base	



 The Open disk image window will appear. Use this window to browse to This PC > Desktop and double-click the folder Toolbox > Datasets > Lab5. This will open the folder revealing 3 FEFs. Select the file called NDG exFAT Lab5.001 and click the Open button as highlighted below. The NTFS image file will now be loaded in HxD.



3. As we did before, let us begin by reviewing the Master Boot Record (MBR – Partition Table) in the first sector to locate the VBR. This means going to the first partition entry located at offset 446. Since we will be using decimal to go to offset 446, we will need to change the Offset base to decimal. To do this, click the View dropdown menu option from the menu bar and hover over the Offset base option, then select Decimal as highlighted in items 1, 2, and 3 below.

A	in t		0.4		1.8.	÷							16	31		1	124		2.2	. 122	÷ .				
	豊く		Au	spru	u wir	100%	with	tu	20	ws (/	11/151				lec		¥ :	14	4.1	<u> </u>	ļį	Sector			
👪 NDG NTFS L	ab5.00) —	вус	es pe	errov	N																			
Offset(d)	0.0	_	Tex	t end	odin	g		- ?		122	22	33	24	113	14	15	Dec	nde	d t	ext					
	2		Off	set b	185E			•		He	cade	cima		6		- 22	-					222	1001		a
000000000	FA		Visi	ble c	olun	nns		•		Dei	rima			E.	E.B.	FU	u3/	AZĐ+	4. <	02.	۲.	uu	Sec	FOL	Ð,
000000016	BF		Byt	e arc	oun s	i7e			-	00	tal			P	18	90	ĉ • •		0¥e	•••	• 1	84 - C			
000000032	00	-							-		-			po	00	90	•••	••••	• • • •	•••	••	••			
00000048	00		Too	lbar	s				12	90	00	90	00	90	00	90	•••	••••	• • • •	•••	••	••			
00000064	00	2	Ref	rech				55	-19	90	00	90	00	90	00	90	•••	••••	• • • •	•••	••	••			
000000080	00	1	iver	resit				12	_10	90	00	90	00	90	00	90		• • • •	• • • •	•••	••	••			
000000096	00	90	00	90	00	90	00	90	00	90	00	90	00	90	00	90		• • • •	• • • •	•••	••	••			
000000112	00	90	00	90	00	90	00	90	00	90	00	90	00	90	00	90		• • • •	• • • •	•••	••	••			
000000128	00	90	00	90	00	90	00	90	00	90	00	90	00	90	00	90		• • • •	• • • •	•••	••	••			
000000144	00	90	00	90	00	90	00	90	00	90	00	90	00	90	00	90	•••	••••	• • • •	•••	••	••			
000000160	00	90	00	90	00	90	00	90	00	90	00	90	00	90	00	90	•••	• • • •	• • • •	•••	••	••			
000000176	00	90	00	90	00	90	00	90	00	90	00	90	00	90	00	00		• • • •	• • • •	• • •	••	••			
000000192	00	90	00	90	00	90	00	90	00	00	00	00	00	00	00	00		••••	• • • •	•••	••	÷ •			
000000208	00	00	00	90	00	00	00	00	00	00	00	00	00	00	00	00		• • • •		•••	••	÷ -			
000000224	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				•••	••	÷ •			
000000240	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				•••	••	÷ •			
000000256	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					••				
000000272	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					••				
000000288	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					•••	÷.			
000000304	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					•••	÷.			
000000320	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					••				
000000336	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					••				
000000352	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00									
000000368	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00									
000000384	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00									
000000400	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00									
000000416	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00									
000000432	00	00	00	00	00	00	00	00	04	41	11	00	00	00	80	01				A		ē.			
000000448	01	00	07	FE	3F	04	3F	00	00	00	86	39	01	00	00	00		.b?	2	. +9					
000000464	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00									
000000480	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00									
000000496	00	00	00	00	00	00	00	00	00	00	00	00	00	00	55	AA						U.a.			
000000512	00	-00	00	-00	00	-00	00	-00	00	00	00	00	00	00	00	00							Sec	tor	ŝ



4. Now let us use Go to to jump to offset 446. This can be opened by going to the Search dropdown menu option in the menu bar and then clicking Go to from the dropdown menu or pressing Ctrl+G. When the Go to window appears, type 446 in the text box, verify that dec is selected, and then click OK as seen below.

		4
hex	() dec	Ooct
Offset rel	ative to	
begin		
<u> </u>	t offset	
Curren	a survey	

5. Your cursor will be taken to offset 446. As we did before, highlight the 16 bytes after the cursor, as seen in item 1 below. You can use the status bar at the bottom of the main window to count the length of your selection, as highlighted in item 2 below.

🗋 👌 • 🔛 🛛	H (3	1	-	+ 1ê	i.	Ŷ	W	indo	ws (A	ANSI			d	lec		🔄 🛛 🖣 🌢 👂 Sect	or
NDG NTFS L	ab5.00	01	1 1	NDG	exFA	TLat	5.00	1										
Offset(d)	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	Decoded text	
000000000	FA	33	CO.	SE	DO	BC	00	70	SB	F4	50	07	50	1F	FB	FC	ú3ÀŽĐ4. <ôP.P.ûü	Sector (
00000016	BF	00	06	B9	00	01	F2	A5	EA	10	06	00	00	CD	18	00	¿ªó¥êÍ	
000000032	00	00	00	00	00	0.0	00	00	00	00	00	0.0	00	00	00	00		
000000045	00	00	00	00	00	0.0	00	00	00	00	00	0.0	00	00	00	00		
000000064	00	00	00	00	00	0.0	00	00	00	00	00.	0.0	00	00	00	0.0		
000000080.	00	00	00	00	00	0.0	00	00	00	00	00	0.0	00	00	00	0.0		
000000096	00	00	00	00	00	00	00	00	00	00	00	0.0	00	00	00	00		
000000112	00	00	00	00	00	00	00	00	00	00	00	0.0	00	00	00	00		
000000128	00	00	00	00	00	00	00	00	00	00	00	0.0	00	00	00	00		
000000144	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000176	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000192	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000208	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000224	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000240	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000256	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000272	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000288	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000304	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000320	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000336	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000352	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000368	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000384	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000000400	00	00	00	00	00		30	-	00	-	100	8.0	-00	00	1	DE		
000000416	00	00	ad	na	90	0.0	00	00	00	0.0	00	-00	00		00	00		
000000432	00	-00	00	00	00	00	00	00	04	41	11	na	00	00	8.0	-617	Δ 🗾	
000000448	In	0.0	-97	THE .	SE	0.4	3F	0.0	-0.0	0.0	86	3.9	-611	0.0	0.0	TIL	55h2 2 55+9 5	
000000464	10	-00	00	00	00	00	00	10.0	0.0	DO	00	DO	00	00	00	00		
000000480	00	00	00	50	00	50	200		00	00	00	00	00	00	00	00		
000000100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	55	22		
000000130	00	- 9.9	00		00		00	- 9.9	00		00	- 44.91	00	- 44	-0.0	and .		



6. Let us break it down to learn what type of file system the entry is referring to and the location of the VBR.



The highlighted value is 0x80 | 0x01 01 00 | 0x07 | 0xFE 3F 04 | 0x3F 00 00 00 | 0x86 39 01 00

- 7. There is no difference between this entry and the NTFS entry because the 5th byte that represents file system type is 07, which is used to represent both NTFS and exFAT file systems. The starting sector of the file system (where the VBR is located) is the same as before. When converted, it is 63. Let us jump to sector 63.
- 8. The table below provides the location of the artifacts based on their offset values for the entire volume as well as for the sector. During this exercise, we will use data from the Dec column to locate the relevant entries in sector 63.

0	ffset	Univ	ersal	Len	Name	Description
Hex	Dec	Off	set	gth (Byt es)		
0x7E00	32256	0x00	0	3	Jump Instructio n	Jump instructions to skip to boot code field
0x7E03	32259	0x03	3	8	OEM ID	ASCII - exFAT
0x7E0B	32267	0x0B	11	53	Must be zero	Replace FAT BIOS parameter block
0x7E40	32320	0x40	64	8	Partition Offset	Sectors from the start of the media
0x7E48	32328	0x48	72	8	Volume Length	Total sector in the volume
0x7E6C	32364	0x6C	10 8	1	Bytes per sector	2 ^N N=Value for Bytes
0x7E6D	32365	0x0D	70 9	1	Sectors per Cluster	2 ^N N=Value for sectors
0x7E48	32328	0x64	10 0	4	Volume Serial Number	Serial number of the Volume
0x7FFE	32766	0x1FE	51 0	2	Boot Signature	0x55 AA



9. Since the calculations for this file system are the same, we already know that the starting sector is located at offset 32256 in our FEF. Let us reopen Go to by clicking the Search dropdown menu option and clicking Go to from the menu or typing Ctrl+G. In the Go to window, type 32256 and click OK as highlighted below.

		β225¢
Ohex	🖲 dec	() oct
Offset rela	ative to	
🖲 begin		
() current	t offset	
O end (h	ackwards)	

10. At offset 32256 (Sector 63), begin by highlighting the OEM ID. This is located at offset 32259 in our FEF or the 3rd byte from the beginning of the sector. Highlight the following 8 bytes after offset 32259 to reveal the OEM ID as highlighted below. As seen in the textual version of the highlighted data, the OEM ID is EXFAT, which indicates that it is an exFAT file system.

Offset(d)	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	Decoded text	^
000032256	EB	7.6	90	45	58	46	41	S 4	20	20	20	00	00	00	00	00	ev.EXFAT Sector	63
000032272	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000032288	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000032304	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		

Unlike the other file systems, the next 53 bytes that follow the OEM ID will always be 0x00 in exFAT.

11. Now let us jump to the Volume length artifact. This is located at offset 32328 in our FEF or the 72nd byte from the beginning of the sector. To do this, let us reopen Go to by clicking the Search dropdown menu option, and clicking Go to from the menu or typing Ctrl+G. In the Go to window, type 32328 and click OK as highlighted below.

iffset:		3232
Ohex) dec	Ooct
Offset rel	ative to	
🖲 begin		
() curren	it offset	

12. Let us highlight the next 8 bytes, as seen below. This value is represented as 0x86 39 01 00 00 00 00 00, which, when converted to decimal, is 80262. This indicates that there are 80262 sectors. The size in bytes can be found by multiplying the number of sectors (80262) by the sector size (512), which is equal to 41,094,144 bytes (41MB).

Offset(d)	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	Decoded text			
000032256	EB	7.6	90	45	58	46	41	54	20	20	20	0.0	00	00	00	00	ëv.EXFAT	1212	Sector	63
000032272	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		16161		
000032288	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		10101		
000032304	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		1(1(1		
000032320	ЗF	0.0	00	0.0	00	0.0	00	0.0	86	89	01	00	00	00	00	00	3ts	0000		
000032336	80	00	00	00	50	00	00	00	00	0T	00	00	10	27	00	00	€Р	0.010		
000032352	05	00	00	00	Ε4	06	C5	70	00	01	00	00	09	03	01	80	ä.Å	€		



13. Next, let us identify the volume serial number for this file system. This can be found at the 100th byte from the beginning of the sector or offset 32356 in our FEF. Let us use Go to by browsing to the Search dropdown menu on the menu bar and clicking Go to or using Ctrl+G. Type 32356 and click OK. Now that you are at the offset 32356 (100th byte), highlight the next 4 bytes as seen below.



The volume serial number is created when the drive is formatted and can be used to determine if files were ever stored on the drive. These bytes represent the volume serial number and are stored in littleendian. When converted, the volume serial number will read 7CC5-06E4.



14. Now, let us use Go to again to go to the bytes per sector artifact. This can be found at the 108th byte from the beginning of the sector or offset 32364 in our FEF. Reopen Go to by browsing to the Search dropdown menu on the menu bar, and clicking Go to or using Ctrl+G. Type 32364 and click OK. You will be taken to offset 32364. Highlight the byte that follows the cursor as highlighted below.

🗋 👌 • 🖬 /	Find	Ctrl+F	NSI)		dec			🛛 🛛 🖣 🌢 🕨 Sector	63 🜻 of 251,904	
NDG FAT I	Replace	Ctrl+R								
	Find again	F3								
Offset(d	Find again (reversed)	Shift+F3	8 09	10 11	1 12	13	14	15	Decoded text	
00003225	Go to	CitleG	2 20	20 00	00	0.0	00	00	ëv.EXFAT	Sector 63
00003227	1 So Marc	curre.	0.00	00.00	00	0.0	00	00		
000032288	00 00 00 00 00	00 00 00 0	0.00	00.00	00	00	00	0.0		
000032304	00 00 00 00 00	00 00 00 00	0.00	00.00	00	0.0	00	00		
000032320	3F 00 00 00 00	00 00 00 5	6 39	01 00	00	00	00	0.0	?	
000032336	80 00 00 00 50	00 00 00 00	0 01	00.00	10	27	00	0.0	€P'	
000032352	05 00 00 00 E4	06 C5 7C 0	0 01	00.00	1 89	03	01	30	ä.Å <mark>.</mark> €	
000032368	00 00 00 00 00	00 00 00 3	3 09	SE DI	SE	C1	SE	D9	3ÉŽŇŽÁŽÚ	
000032384	BC DO 7B BD 00	7C 88 16 6	F 70	B4 41	BB	AA	55	CD	4£D{45. ^.0 'A≫*UÍ	
000032400	13 Go to		×	1 01	74	SE	FE	0.6	.ri.ûU*ucöÁ.t^þ.	
000032416	02			8 66	6 B8	01	00	0.0	. fP°eė¦.fXf,	
000032432	00 Offset:			6 E8	66	BS	01	0.0	.Ś.m fÓàf%Féf,	
000032448	00		32364	3	5 D8	66	AL	40	Ś.1 fÓàfhFØf;@	
000032464	70		plaidudy	p Es	41	0.0	66	58	f@».~1fPéA.fX	
000032480	66 () hex (0)	dec Ood		0.66	5.0	BO	78	ES	f@».€³è4.fP°xè	
000032496	50			B 05	A0	FB	7D	EB.].fXė ü)ë. û)ë	
000032512	00 Offset relative to			8 74	OE	B4	OE	BB	.')<8-"@t.Ht.'.»	
000032528	07 Degin			6 CI	0 16	CD	19	66	1.ëi ý)ëæ1.1.f	
000032544	60 0 03	6		b 00	1 01	00	B4	42	'fi.fP.Sfh'B	
000032560	B2 - Current offset		6 58	66	-58	66	58	f€Š.ol(ôÍ.fXfXfX		
000032576	66 O end (backwai	rds)		0 49	75	D1	C3	66	fXfar±.^Øf@IuÑÄf	
000032592	60			0 66	6 61	C3	42	00	`'.»İ.faÄB.	
000032608	4F (A)	OV C-	000	2 00	1 OD	02	52	65	0.0.T.M.G.RRe	
000032624		UN Ca	ncei	5 65	72	20	6F	74	move disks or ot	
000032640	68 65 72 20 6D	65 64 69 6	1 2E	TO TT	AO C	44	69	73	her media.U. Dis	
000032656	6B 20 65 72 72	6F 72 FF 0	D 02	50.72	65	73	79	20	k errorü Press	
000032672	61 6F 79 20 6B	65 79 20 7	4 6F	20.75	65	73	74	63	any key to resta	
000032688	72 74 0D 02 00	00 00 00 0	0.00	00 00	1 00	00	FF	FF	rt	
000032704	FF FF FF FF FF		मन म	FF FT	7 77	माम	FF	मम	000000000000000000000000000000000000000	
000032720	FF FF FF FF FF	FF FF FF F	FFF	FF FT	FF	FF	FF	FF	111111111111111111111111111111111111111	
000032736	FF FF FF FF FF	FF FF FF F	FFF	FF FI	F FF	R.R.	FF	R.R.	333333333333333333333333333333333333333	
000032752	FF FF FF FF FF	FF FF FF F	FFF	FF 60	BB	9.9	55	AA	11-11-11-11-11-11-11-11-11-11-11-11-11-	
Offect(d): 37364	Block(d): 2	2364-22364				1	enc	h(d)	1	Overwrite

The exFAT file system stores the sector size as 2 to the power of the value in the VBR. The value highlighted below is 0x09 or 9 in decimal; 2 to the power of 9 is 512 as such the number of bytes per sector on this volume is 512 bytes.



15. Next, let's determine the number of sectors per cluster. This is located at offset 32365 in our FEF or the 109th byte from the beginning of the sector. This means, it is right beside the bytes per sector above. Let us highlight the value. As seen below, the value is 0x03 or 3 in decimal. As with above, the calculation is done by raising 2 to the power of the highlighted value. In this case, the equation is 2^{3,} which is equal to 8.

Offset(d)	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	Decoded text	
000032256	EB	7.6	90	45	58	46	41	54	20	20	20	00	00	00	00	00	ëv.EXFAT	Sector 63
000032272	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000032288	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000032304	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000032320	ЗF	00	00	00	00	00	00	00	86	39	01	00	00	00	00	00	?t9	
000032336	80	0.0	00	0.0	50	0.0	00	0.0	00	01	00	0.0	10	27	00	00	€₽	
000032352	05	00	00	00	Ε4	06	C5	70	00	01	00	00	09	0.3	01	80	ä.Å∣ <mark>.</mark> .€	
000032368	00	00	00	00	00	00	00	00	33	09	θE	D1	8E	Cl	SE	D9	3ÉŹŇŹÁŹŮ	
000032384	BC	DO	7B	8D	00	70	88	± 6	6F	70	B4	41	BB	AA	55	CD	4Ð{%. ^.o ′A≫*UÍ	



This means that there are 8 clusters per sector in this volume. Combining this value with the number of bytes per sector can provide the cluster size for the volume. In this case, 512 bytes per sector * 8 sectors = size of each cluster 4096 bytes.

- 16. As with the previous file system, there are many artifacts that can be unearthed but will not be covered in this exercise.
- 17. The artifacts you unearthed in this lab can help you get a better understanding of the way data is stored on a drive and determine its structure. Many technical users manipulate the logical volumes to hide data. Mastering this part of the analysis will give you a head start in detecting these attempts and successfully investigating the dataset.



18. Now that you are done, close the Go to window and the HxD program by clicking the X at the top-right corner as highlighted below.

