Windows Forensics

Understand Analysis Techniques for Your Windows

—

Dr. Chuck Easttom
Dr. William Butler
Jessica Phelan
Ramya Sai Bhagavatula
Sean Steuber
Karely Rodriguez
Victoria Indy Balkissoon
Zehra Naseer

Apress®

Windows Forensics

Understand Analysis Techniques for Your Windows

Dr. Chuck Easttom
Dr. William Butler
Jessica Phelan
Ramya Sai Bhagavatula
Sean Steuber
Karely Rodriguez
Victoria Indy Balkissoon
Zehra Naseer

Apress®

Windows Forensics: Understand Analysis Techniques for Your Windows

Chuck Easttom
Plano, TX, USA

William Butler
Maryland, MD, USA

Jessica Phelan
Austin, TX, USA

Ramya Sai Bhagavatula
Houston, TX, USA

Sean Steuber
Kansas City, MO, USA

Karely Rodriguez
Bonney Lake, WA, USA

Victoria Indy Balkissoon
Lake Mary, FL, USA

Zehra Naseer
Eastvale, CA, USA

ISBN-13 (pbk): 979-8-8688-0192-1
ISBN-13 (electronic): 979-8-8688-0193-8
https://doi.org/10.1007/979-8-8688-0193-8

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Jessica Vakili
Development Editor: Laura Berendson
Editorial Assistant: Smriti Srivastava

Cover designed by eStudioCalamar

Cover image designed by Freepik (www.freepik.com)

Distributed to the book trade worldwide by Springer Science+Business Media New York, 1 New York Plaza, Suite 4600, New York, NY 10004-1562, USA. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

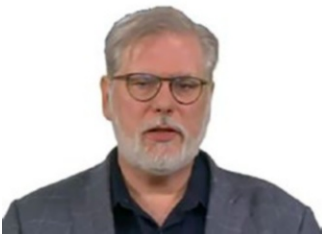
Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub. For more detailed information, please visit https://www.apress.com/gp/services/source-code.

If disposing of this product, please recycle the paper

This book is dedicated to forensic analysts and students of forensics.

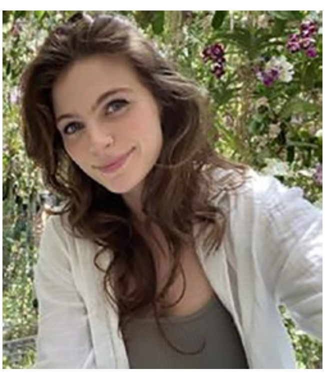
Table of Contents

About the Authors

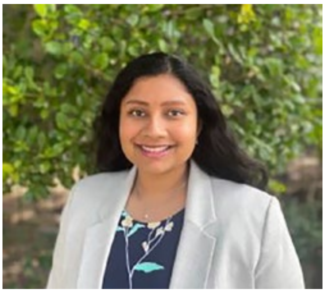


Dr. Chuck Easttom is the author of 42 books, including several on computer security, forensics, and cryptography. He is also an inventor with 26 patents and the author of over 70 research papers. He holds a doctor of science in cybersecurity, a PhD in nanotechnology, a PhD in computer science, and four master's degrees.

Dr. William Butler is Vice President of Academic Affairs and Executive Director of the Center for Cybersecurity Research and Analysis (CCRA) at Capitol Technology University (located in Laurel, Maryland). Before this appointment, Bill served as the Chair of Cybersecurity programs for eight years.



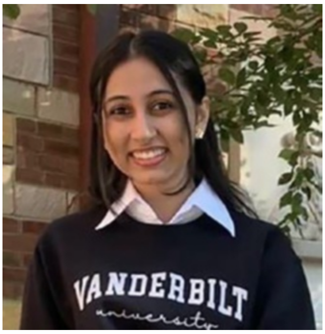
Jessica Phelan is a computer science graduate student at Vanderbilt University. She is currently doing research in natural language processing at the University of Texas at Austin.



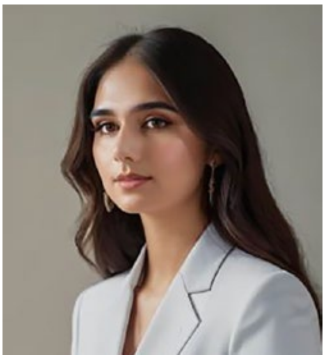
Ramya Sai Bhagavatula is a cybersecurity enthusiast and holds a Security+ Certification from CompTIA. She is currently working as an AI engineer for a medical organization, Baylor Genetics, where she is using her AI expertise to work with genomic data to bring out valuable insights and predictions. She has previously worked for NASA as a Deep Learning Research Intern, where she developed deep learning models to effectively predict severe climate patterns. She was also a lead Data Analyst Intern at an arts organization, Houston Arts Alliance, where she was involved in analyzing in-depth patterns and providing recommendations for their future art grants. Through her dedication to continuous learning and professional development, she pursued her master's in Data Analytics at the University of Houston and is currently pursuing her second master's in Computer Science at Vanderbilt University. She is also 3x Microsoft Certified in AI and Data Engineering. She aspires for her future career path to involve cybersecurity, quantum computing, and AI. In her free time, she loves to volunteer at local organizations to raise awareness about computer science among underprivileged school students. She has also received the Presidential Volunteer Service Award for her volunteer services.

Sean Steuber holds a BS in Engineering from the University of Alabama and an MS in Computer Science from Vanderbilt University and has eight years of professional computer science experience.

Karely Rodriguez is a first-generation DACA recipient and a woman pursuing STEM. She earned a Bachelor of Science in Computer Science and minored in Mathematics at the University of Washington and has continued her education in achieving a Master of Computer Science from Vanderbilt University.

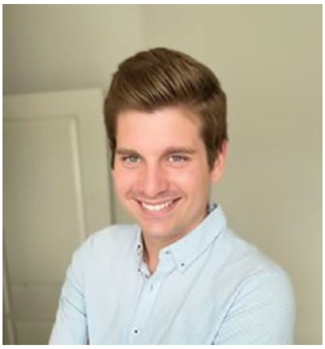


Victoria Indy Balkissoon is working in the Naval Enterprise Research Data Science (N.E.R.D.S.) team at NAWCTSD Orlando where she currently works on developing software applications and data science solutions for the US Navy. She is also currently pursuing a master's degree in Computer Science at Vanderbilt University.



Zehra Naseer holds an MS in Computer Science from Vanderbilt University.

About the Technical Reviewer



Dylan Waggy is a Senior Advisor and Incident Response Analyst with over eight years of experience at a leading American multinational technology company. Armed with a bachelor's in Digital Forensics, he holds certifications in Certified Forensic Computer Examiner (CFCE), Cloud Forensics Responder (GCFR), and Reverse Engineering Malware (GREM). Dylan has successfully contributed to over 300 federal crime cases, assisting law enforcement entities. He has also played a crucial role in establishing digital forensics and incident response teams for Fortune 500 companies. Passionate about proactive and retrospective protection, Dylan takes pride in fortifying organizations against both internal and external threats. His work reflects a commitment to elevating the standards of digital forensics and incident response in today's dynamic cybersecurity landscape.

Acknowledgments

The authors of this book would like to thank the wonderful team at Apress publishing, including the technical reviewer. Without their help, this book would not be possible.

Introduction

Windows is a ubiquitous operating system. As a forensic examiner, you will likely encounter Windows machines quite frequently. Certainly, many forensics tools can extract data from a Windows computer, even if the user of such tools is not well versed in Windows. However, it is important that you fully understand the Windows operating system. This is necessary first so that you can properly understand and interpret the information that such tools provide. Secondly, a thorough knowledge of Windows is important because no tool is perfect. Any tool may miss something. Only by having a solid understanding of the Windows operating system can you identify such gaps and seek the evidence through alternative means.

This book begins with an overview of the Windows operating system. This will provide you a foundational understanding to base the rest of the book on. Then in Chapter 2, you will learn forensic concepts. This includes legal standards such as the Daubert vs. Dow Chemicals case and Federal Rule 702, as well as the scientific method. Subsequent chapters will then go through different portions of Windows including the Windows Registry, Shadow Copy, and related topics. You will also learn to use Microsoft PowerShell to accomplish forensics tasks.

This book is designed for two audiences. The first is the student that is learning forensics. This could be in a university setting or less formal setting. As the book assumes no prior knowledge of either forensics or Microsoft Windows, it can be used by a beginner. The second audience is the professional forensic examiner that requires a more in-depth understanding of Microsoft Windows forensics. This book will provide a depth that will give you a thorough understanding of how to do Windows forensics.

Throughout the book, you will be introduced to forensic tools. These will include commercial tools such as OSForensics as well as open source tools such as Autopsy. The coverage of tools will allow you to actually conduct a detailed forensic examination of a Microsoft Windows computer.

CHAPTER 1

Introduction to Windows

Introduction

It is certainly possible to perform Windows forensics without a deep understanding of the operating system. That is, however, a serious mistake. The various automated forensics tools, many of which you will see in this textbook, can provide you evidence, but they cannot interpret the evidence for you. Furthermore, the automated tools cannot always catch everything. To be a truly competent Windows forensic examiner, you must have an understanding of the operating system itself. The goal of this chapter is to provide you a working knowledge of the Windows operating system and a strong foundation for learning more. To be able to truly perform forensics on any system, you need a deep understanding of that system.

What Is an Operating System?

Before delving too deeply into the Windows operating system, it is helpful to first explore what an operating system is. An operating system (OS) is the underlying software that provides a computer user with all the basic services of resource management on the machine, including a file system structure for data storage and a means of communicating with all the various computer hardware. The operating system controls input and output (I/O) from disk storage (hard drives, solid-state drives, etc.), and other computer components. It is also the job of the operating system to make sure programs running on the computer do not interfere with each other when competing for system resources. This involves memory and resource management.

C. Easttom et al., *Windows Forensics*, https://doi.org/10.1007/979-8-8688-0193-8_1

The core of any operating system is referred to as the kernel. The kernel is the core of the operating system. A process is an executing instance of a program. The kernel ensures that processes are allocated the necessary resources and are executed without interfering with each other. There are three types of kernel. With a monolithic kernel, all the system services run along with the main kernel thread in a single memory space. This makes them fast but potentially less secure, as a bug in one service can affect the entire system. A microkernel will manage the core system services like networking, file system drivers, etc., as separate processes, usually in user space. This can provide increased system stability and security but might be slower due to the additional overhead of communication between the kernel and the service processes. A hybrid kernel is a mix of monolithic and microkernel designs.

Most modern operating systems support multitasking. Multitasking is the ability of an operating system to simultaneously support two or more running programs. When multitasking, it seems to the user that both programs are running simultaneously even though they are not. The computer simply switches control between the programs, giving the illusion they are running at the same time. For example, imagine you printed a file while browsing the Internet, streaming music, and checking your email. It may appear as though all these programs are running simultaneously, but in reality, the computer runs the software in between sending packets of data to the printer.

One common way to accomplish multitasking is called preemptive multitasking, sometimes referred to as time slicing, which is a process that allows multiple programs to share control of the operating system. For example, two or more programs can share the CPU for processing information, but no single program can totally take charge of a computer system. All programs running in preemptive mode are allowed to run for a set period of time, called the time slice, by an operating system process known as the scheduler. At the end of the time slice, a process is interrupted so the next process in line can run. This way, all the processes on the computer can share the CPU fairly. Since each time slice is quite brief, a few milliseconds, it appears the system is performing tasks simultaneously.

History of Windows

Microsoft Windows was released as just a graphical user interface (GUI) for the MS-DOS (Microsoft Disk Operating System) operating system. Windows itself was not actually an operating system. In fact, versions 1.0 to 3.11 were simply GUIs on top of MS-DOS.

Windows 1.0 was released in 1985 but received very little notice from the public. Windows 2.0 and 2.1 were released in 1987 and 1988, respectively, but were still not widely popular. Windows 3.0 was released in 1990, then 3.1 in 1992. Most of the public began to use Windows with version 3.1. It became quite popular.

Windows NT was released in 1993 and was a separate product from the consumer Windows versions. Windows NT was designed to be used in a work environment, on a local area network. While the interface looked quite similar to the consumer version, the internals were different. There were workstation and server versions of Windows NT.

Windows 95 marked a shift in the consumer version of Windows. While not entirely a stand-alone operating system, it was not simply a GUI either. Furthermore, Windows 95 was 32 bits (at least most of it). The fusion of the GUI with the operating system has continued throughout subsequent versions. The general outline and description of various versions is given here:

Windows 1.0 (1985): The first version of Windows was essentially a graphical shell for MS-DOS, allowing users to run programs in a graphical environment. It introduced basic features like scroll bars, windows, and icons.

Windows 2.0 (1987): Improved on the first version with better graphics support and overlapping windows. It was during this era that Microsoft introduced the Excel and Word programs.

Windows 3.0 and 3.1 (1990–1992): These versions marked the true beginning of Windows' dominance. They supported 16 colors and improved the interface significantly. Windows 3.1, in particular, saw widespread adoption.

Windows 95 (1995): A major milestone, Windows 95 introduced the Start menu, Taskbar, and the concept of "plug and play" hardware. It also integrated MS-DOS with Windows more tightly.

Windows 98 (1998): Built on Windows 95 but with additional support for new technologies like USB, DVD, and ACPI.

Windows ME (Millennium Edition) (2000): Aimed at home users, it was not very well received due to its instability and was quickly overshadowed by its NT-based counterparts.

Windows 2000: Part of the NT family, it was geared more toward business users, known for its stability and security.

Windows XP (2001): One of the most successful versions, combining the consumer-friendly interface of the 9x series with the stability of the NT line. XP remained popular for many years, even well beyond its intended life cycle.

Windows Vista (2006): Introduced Aero graphics, improved security, and a new search function. However, it faced criticism for heavy resource requirements and compatibility issues.

Windows 7 (2009): Addressed many of Vista's issues and was praised for its performance, user interface, and enhanced security features.

Windows 8 (2012): Represented a significant overhaul, introducing a touch-centric interface and the Metro design language. However, the removal of the Start menu and focus on touch were controversial.

Windows 8.1 (2013): An update to Windows 8, it brought back the Start button and made several adjustments based on user feedback.

Windows 10 (2015): Aimed to address the criticisms of Windows 8, reintroducing a Start menu and supporting both touch and traditional PC users. It was positioned as a service, with regular updates.

Windows 11 (2021): The latest version as of this writing, Windows 11 introduced a redesigned Start menu, improved window management features like Snap Layouts, and a focus on security and performance.

These are just the client systems. The server operating systems is given in the following brief paragraphs:

Windows NT 3.1 Advanced Server (1993): This was the first version of Microsoft's server operating system, building on the Windows NT architecture, which was designed for robustness and security.

Windows NT 3.5 Server (1994): An update to the original NT system, it included performance improvements and support for new hardware.

Windows NT 3.51 Server (1995): This release focused on interoperability with NetWare networks and included the first version of the web server, Internet Information Services (IIS).

Windows NT 4.0 Server (1996): A major upgrade with a new user interface aligned with Windows 95. It included IIS 2.0 and brought in the concept of domains and user accounts for managing network resources.

Windows 2000 Server (2000): Introduced Active Directory, a directory service for managing domains, users, and resources. It also brought in improved support for web services and scalability.

Windows Server 2003 (2003): This version improved Active Directory and included better default security, IIS 6.0, and support for .NET framework. It was also the first server OS to drop support for older Windows 9x clients.

Windows Server 2003 R2 (2005): An update to the 2003 version, it included enhancements like a common log file system and improved branch office performance.

Windows Server 2008 (2008): Introduced Server Core, a minimal installation option for reduced maintenance and attack surface. It also included Hyper-V for virtualization and improved security and management features.

Windows Server 2008 R2 (2009): This was the first Windows Server OS exclusively for 64-bit processors. It improved upon virtualization with Hyper-V 2.0 and included features like DirectAccess and BranchCache.

Windows Server 2012 (2012): A major release with a focus on cloud computing, it introduced a redesigned user interface based on Windows 8, a new version of Hyper-V, and a new file system (ReFS).

Windows Server 2012 R2 (2013): Included enhancements to Hyper-V, storage, networking, and included the return of the Start button in the UI.

Windows Server 2016 (2016): This version focused on cloud and container support, introducing Docker compatibility, Nano Server for lightweight environments, and enhanced security features like Shielded Virtual Machines.

Windows Server 2019 (2018): Continued the focus on hybrid cloud environments, with improved Kubernetes support, Windows Admin Center for management, and enhanced security features.

Windows Server 2022 (2021): The latest version as of my last update, focusing on advanced multilayer security, hybrid capabilities with Azure, and a flexible application platform.

The File System

Operating systems interact with the file system to access files. A file system refers to the method of organizing files on a storage device. It is an indexing system used by the operating system to keep track of all files on the disk. The file system maintains a file table of all areas on the disk, and it tracks which areas are being used for data and which are free and available at any given time. A file table is a component of a file system used to organize files on a storage device.

Microsoft uses NTFS, New Technology File System. One major improvement of NTFS over FAT was the increased volume sizes NTFS could support. The maximum NTFS volume size is $2^{64}-1$ clusters. NTFS also introduced the Encrypted File System (EFS). This allows the end user to easily encrypt and decrypt individual files and folders. There are several individual files that are key to this file system. Two of the most fundamental are the MFT (Master File Table, some sources call it the Meta File Table) file and the cluster bitmap. The MFT describes all files on the volume, including file names, timestamps, security identifiers, and file attributes such as “read only,” “compressed,” “encrypted,” etc. This file contains one base file record for each file and directory on an NTFS volume. It serves the same purpose as the file allocation table does in FAT and FAT32. The cluster bitmap file is a map of all the clusters on the hard drive. This is an array of bit entries where each bit indicates whether its corresponding cluster is allocated/used or free/unused.

Unlike FAT/FAT32, NTFS is a journaling file system, which means it records actions so they can be undone. NTFS uses the NTFS Log ($Logfile) to record information about changes to the volume. With the advent of NTFS, file names can be 1 to 255 characters in length, including the path. You can use uppercase and lowercase (case-aware, but not case-sensitive). You can use spaces and periods. You cannot use these characters:

/ \ : * ? “ < > |

With Windows 2000, Microsoft added reparse points to NTFS. Reparse points provide a mechanism to extend the functionality of the file system and are used to implement several advanced features in Windows. A reparse point is essentially a type of data attribute that can be associated with a file or directory, instructing the file system to treat that file or directory in a special way. There are three types of reparse points:

1. **Junction Points**: Similar to Unix hard links, they allow directories to be aliased at another location in the file system. These are the most common.

2. **Symbolic Links**: Introduced in Windows Vista, they are more flexible than junction points and can point to files or directories and work across local and network paths.

3. **Volume Mount Points**: Allow a volume to be mounted at a directory rather than a drive letter.

Since Windows Vista, NFTS has supported what is called Transactional NTFS (TxF). Developers can use this to write transactions that either succeed completely or fail completely, much like database transactions. TxF allows for grouping a series of file operations into a single transaction. This transaction is atomic, meaning either all operations in the transaction are completed successfully or none of them are applied. This is crucial for maintaining data integrity. Transactions are isolated from each other. Changes made in one transaction are not visible to other transactions until they are committed.

The NTFS boot sector contains values described in Table 1-1.

Table 1-1. *NTFS Boot Sector*

Byte Offset	Field Length	Typical Value	Field Name		Purpose
0x00	3 bytes	0xEB5290	x86 JMP and NOP instructions		This causes execution to continue after the data structures in this boot sector.
0x03	8 bytes	"NTFS" Word "NTFS" followed by four trailing spaces (0x20)	OEM ID		This is the indicator that this is an NTFS file system.
0x0B	2 bytes	0x0200	BPB	Bytes per sector	The number of bytes in a disk sector.
0x0D	1 byte	0x08	BPB	Sectors per cluster	The number of sectors in a cluster.
0x0E	2 bytes	0x0000	BPB	Reserved sectors, unused	
0x10	3 bytes	0x000000	BPB	Unused	This field is always 0.
0x13	2 bytes	0x0000	BPB	Unused by NTFS	This field is always 0.
0x15	1 byte	0xF8	BPB	Media Descriptor	The type of drive. 0xF8 is used to denote a hard drive.
0x16	2 bytes	0x0000	BPB	Unused	This field is always 0.
0x18	2 bytes	0x003F	BPB	Sectors per track	The number of disk sectors in a drive track.
0x1A	2 bytes	0x00FF	BPB	Number of heads	The number of heads on the drive.
0x1C	4 bytes	0x0000003F	BPB	Hidden sectors	The number of sectors preceding the partition.
0x20	4 bytes	0x00000000	BPB	Unused	Not used by NTFS.

(*continued*)

Table 1-1. *(continued)*

Byte Offset	Field Length	Typical Value	Field Name		Purpose
0x24	4 bytes	0x00800080	EBPB	Unused	Not used by NTFS.
0x28	8 bytes	0x00000000007FF54A	EBPB	Total sectors	The partition size in sectors.
0x30	8 bytes	0x0000000000000004	EBPB	$MFT cluster number	The cluster that contains the Master File Table.
0x38	8 bytes	0x000000000007FF54	EBPB	$MFTMirr cluster number	The cluster that contains a backup of the Master File Table.
0x40	1 byte	0xF6	EBPB	Bytes or Clusters per File Record Segment	The number of clusters in a File Record Segment.
0x41	3 bytes	0x000000	EBPB	Unused	This field is not used by NTFS.
0x44	1 byte	0x01	EBPB	Bytes or clusters per index buffer	The number of clusters in an index buffer.
0x45	3 bytes	0x000000	EBPB	Unused	This field is not used by NTFS.
0x48	8 bytes	0x1C741BC9741BA514	EBPB	Volume serial number	A unique random number assigned to this partition.
0x50	4 bytes	0x00000000	EBPB	Checksum, unused	
0x54	426 bytes		Bootstrap code		The code that loads the rest of the operating system.
0x01FE	2 bytes	0xAA55	End-of-sector marker		This flag indicates that this is a valid boot sector.

There is a great deal of information in the boot sector, as you might expect. All of this is used in the booting of the system. Figure 1-1 is a screenshot of the boot sector of an NTFS volume as viewed in OSForensics.

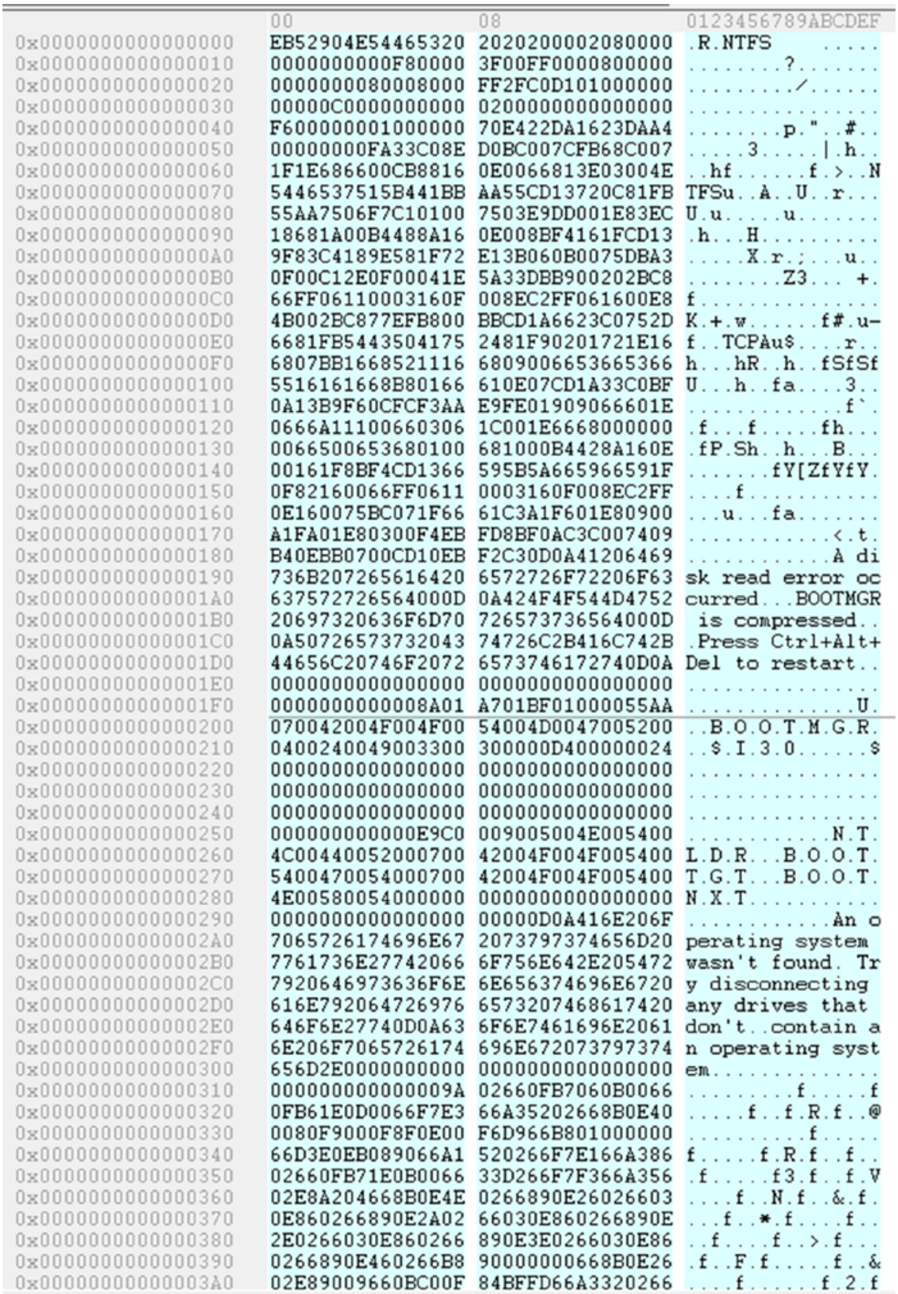


Figure 1-1. *NTFS boot sector*