

Get unlimited access to the best of Medium for less than \$1/week. [Become a member](#)



TRYHACKME SERIES

TryHackMe: MBR and GPT Analysis

Learn how MBR and GPT forensics are carried out to identify attacks during the boot process.



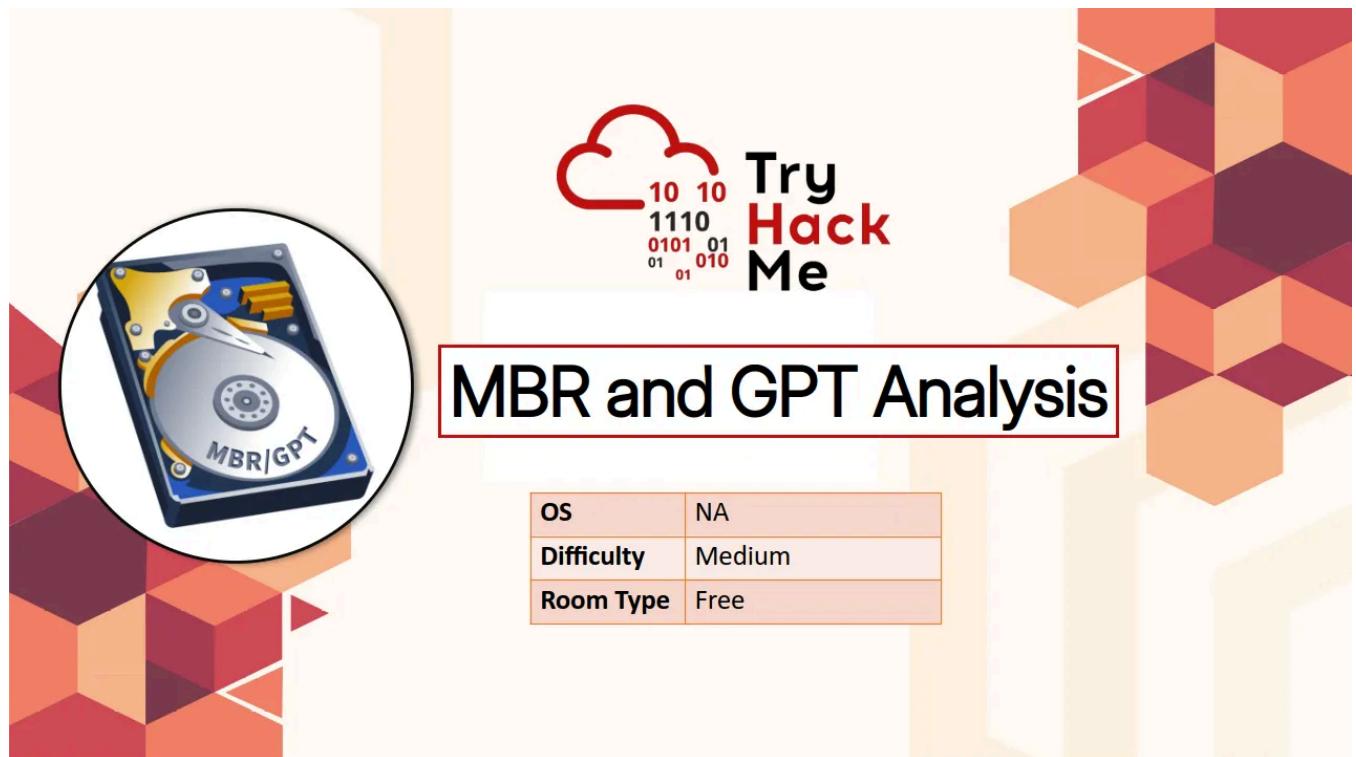
David Varghese · [Follow](#)

8 min read · 11 hours ago

Listen

Share

More



Banner background by [BiZkettE1](#) on Freepik

MBR and GPT Analysis

Learn how MBR and GPT forensics are carried out to identify attacks during the boot process.

tryhackme.com

⚠️ Important

Read the material that accompanies each task before taking a look at this guide. This guide will only cover content that is necessary to answer the questions.

Task 1: Introduction

1. What are the separate sections on a disk known as?

Imagine your disk as a large building storing all the data. This data is stored in binary format, as 1s and 0s, for the computers to understand. The challenge here is that without properly organizing this data, it would be a mess in the whole disk. To solve this problem, the disk is divided into multiple partitions just like rooms in the building, and each partition contains specific data. For example, operating system files can be stored in one partition, personal files can be stored in another, etc. In the Windows OS, these partitions are represented by drive letters such as C, D, E, etc. Other operating systems may use different ways to refer to these partitions.

Partition

2. Which type of malware infects the boot process?

This solves the problem of organizing the data on the disk. However, the computer still needs a map that helps guide it through these partitions, where they start and end, and what they contain. **MBR** (Master Boot Record) and **GPT** (GUID Partition Table) are different partitioning schemes that act as a map for all of the partitions used in the disk. They are like a blueprint of the building (disk) containing all the rooms (partitions). Both partitioning schemes differ in structure and properties, and choosing between them depends on multiple factors, including the disk size, hardware compatibility, and much more. The MBR/GPT is located in the very first sector of the disk and contains information about the structure and partitions of the disk. It also plays a key role during the boot process of a system. Due to this, the MBR/GPT has become an attractive target for attackers to manipulate the boot process by embedding their malware, often known as **Bootkits**, or tampering with them to make the system un-bootable.

Bootkit

Task 2: Boot Process

1. What is the name of the hardware diagnostic check performed during the boot process?

Power-On-Self-Test (POST)

The system is now powered on, and the CPU has executed instructions from the firmware (BIOS/UEFI) installed. The BIOS/UEFI then starts a Power-On-Self-Test to ensure all the system's hardware components are working fine. You may hear a single or multiple beeps during this process in your system; this is how the BIOS/UEFI communicates any errors in the hardware components and displays the error message on the screen, e.g., keyboard not found.



Power-On-Self-Test

2. Which firmware supports a GPT partitioning scheme?

BIOS (Basic Input/Output System) and **UEFI** (Unified Extensible Firmware Interface) are responsible for verifying whether all the hardware components work properly. A system can either use BIOS or UEFI firmware. The difference between them lies in their capabilities and features.

- **BIOS** has been used for decades and is still used in some hardware. It runs in the basic 16-bit mode and supports only up to 2 terabytes of disks. The most important thing to note is that **BIOS** supports the MBR partitioning scheme, which we will discuss later in the task.
- **UEFI** came as a replacement for **BIOS**, offering 32-bit and 64-bit modes with up to 9 zettabytes of disks. **UEFI** offers a secure boot feature to ensure integrity during the system boot process. It also offers redundancy, allowing us to recover from the backup even if the boot code is corrupted. **UEFI uses a GPT partitioning scheme**, unlike the MBR partitioning scheme used by **BIOS**.

UEFI

3. Which device has the operating system to boot the system?

Locate the Bootable Device

After the BIOS/UEFI has performed the POST check, it is time for the BIOS/UEFI to locate bootable devices, such as SSDs, HDDs, or USBs, with the operating system installed. Once the bootable device is located, it starts reading this device. Now, here comes the role of the **MBR/GPT**. The very first sector of the device would either contain the MBR (Master Boot Record) or the GPT (GUID Partition Table). The MBR/GPT would be taking control of the boot process from here. In the upcoming tasks, we will see how the boot process propagates from here if the MBR partitioning scheme is used and what happens if it is GPT instead.

Bootable Device

Task 3: What if MBR?

1. Which component of the MBR contains the details of all the partitions present on the disk?

Partitions Table (Bytes 446-509)

The second component of the MBR is the partition table, which comprises 64 bytes (Bytes 446-509). This table contains the details of all the partitions present on the disk. One of the partitions in the disk contains all the operating system files necessary for booting, known as a bootable partition. The initial bootloader that was started from the bootloader code of the MBR finds the bootable partition from this partition table, and it loads the second bootloader from it. The second bootloader then loads the **Operating System's Kernel**. A partition table is used during this boot process. However, this partition table can also give us valuable information during forensics. Let's dive into the details of this partition table.

An MBR disk has a total of 4 partitions, and each partition is represented by 16 bytes in the partition table. In the screenshot below, four partitions (each with 16 bytes) are highlighted with different colors.

Partition Table

2. What is the standard sector size of a disk in bytes?

The bootable device, which was located in the 3rd step of the boot process, would be in the form of a disk. A disk is divided into multiple sectors, each with a standard size of 512 bytes. The first sector of this disk would contain the MBR. A disk's MBR can be viewed by taking the system's disk image and opening it in a hexadecimal editor.

We will be using the HxD tool, a hexadecimal editor available in the taskbar of the attached machine. Open the HxD tool, click the File button in the options tab, and select Open from the options. Now, you have to input the file's location to be opened. We have extracted the MBR portion from a disk image and saved it at `C:\Analysis\MBR`. You can select this file to examine the MBR for this task.

512

3. Which component of the MBR is responsible for finding the bootable partition?

This Bootloader code contains the **Initial Bootloader**. The initial bootloader is the first thing that executes in the MBR. This initial bootloader code has a primary purpose of finding the bootable partition from the **partition table** present on the MBR.

Note: To better understand the initial bootloader code, it can be disassembled into assembly language, which is beyond the scope of this task.

Bootloader Code

4. What is the magic number inside the MBR?

These two bytes `55 AA` are also known as a **Magic Number** and they indicate that the MBR has been ended now. If these two bytes are changed to some value other than `55 AA`, the system cannot boot. These bytes often get corrupted due to bad sectors on the disk and sometimes can be intentionally changed by malware to interrupt the normal boot process.

55 AA

5. What is the maximum number of partitions MBR can support?

An MBR disk has a total of 4 partitions, and each partition is represented by 16 bytes in the partition table. In the screenshot below, four partitions (each with 16 bytes) are highlighted with different colors.

000001B0	80 20	em...c(s*x....e
000001C0	21 00 07 FE FF FF 00 08 00 00 B0 23 03 00 FE	!..bÿÿ....#..þ
000001D0	FF FF 07 FE FF FF 00 B8 23 03 00 80 38 01 00 FE	ÿÿ.þÿÿ.#!..€8..þ
000001E0	FF FF 07 FE FF FF 00 38 5C 04 00 C0 D4 01 00 00	ÿÿ.þÿÿ.8..AØ...
000001F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00U*

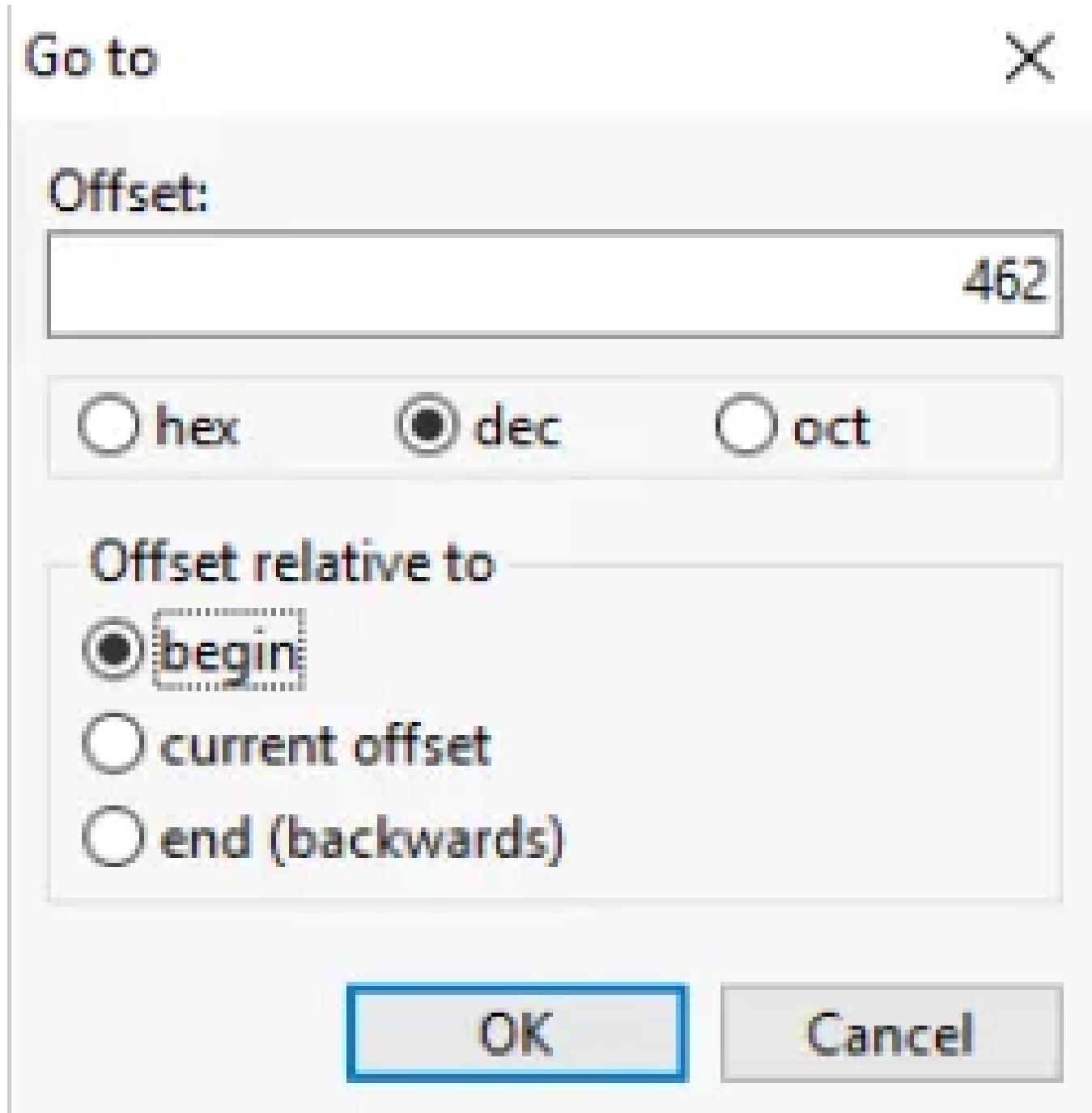
4

6. What is the size of the second partition in the MBR attached to the machine? (rounded to the nearest GB)

Launch the VM and load the MBR file in HxD Editor. To calculate the size of the partition we first need to locate the partition table.

From the write-up we know that the partition table starts at byte 446. We also know that each entry in the partition table is 16 bytes. Using this information we can deduce that the entry for the 2nd partition will start from byte 462 ($446 + 16$).

We can directly jump to this location using “Search → Go to”.



The below image shows the partition table entry for the 2nd partition.

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	33	C0	8E	D0	BC	00	7C	8E	C0	8E	D8	BE	00	7C	BF	00	3ÀŽÐ4. ŽÀŽØ%. ð.
00000010	06	B9	00	02	FC	F3	A4	50	68	1C	06	CB	FB	B9	04	00	.¹..üó¤Ph..ÉÙ¹..
00000020	BD	BE	07	80	7E	00	00	7C	0B	0F	85	0E	01	83	C5	10	¾.€~... fÅ.
00000030	E2	F1	CD	18	88	56	00	55	C6	46	11	05	C6	46	10	00	âñí.^V.UÆF..ÆF..
00000040	B4	41	BB	AA	55	CD	13	5D	72	0F	81	FB	55	AA	75	09	'A»*UÍ.]r..GU*u.
00000050	F7	C1	01	00	74	03	FE	46	10	66	60	80	7E	10	00	74	÷Á..t.bF.f`€~..t
00000060	26	66	68	00	00	00	00	66	FF	76	08	68	00	00	68	00	&fh....fÿv.h..h.
00000070	7C	68	01	00	68	10	00	B4	42	8A	56	00	8B	F4	CD	13	h..h..`BŠV.<ðí.
00000080	9F	83	C4	10	9E	EB	14	B8	01	02	BB	00	7C	8A	56	00	ÝfÅ.žë.,...». ŠV.
00000090	8A	76	01	8A	4E	02	8A	6E	03	CD	13	66	61	73	1C	FE	Šv.ŠN.Šn.í.fas.p
000000A0	4E	11	75	0C	80	7E	00	80	0F	84	8A	00	B2	80	EB	84	N.u.€~.€..š..€ë..
000000B0	55	32	E4	8A	56	00	CD	13	5D	EB	9E	81	3E	FE	7D	55	U2äŠV.í.]ěž.>p}U
000000C0	AA	75	6E	FF	76	00	E8	8D	00	75	17	FA	B0	D1	E6	64	*unýv.è..u.ú°Ñæd
000000D0	E8	83	00	B0	DF	E6	60	E8	7C	00	B0	FF	E6	64	E8	75	èf.ºßæ`è .ºÿædèu
000000E0	00	FB	B8	00	BB	CD	1A	66	23	C0	75	3B	66	81	FB	54	.ù.,»í.f#Àu;f.ùT
000000F0	43	50	41	75	32	81	F9	02	01	72	2C	66	68	07	BB	00	CPAu2.ù..r,fh..».
00000100	00	66	68	00	02	00	00	66	68	08	00	00	00	66	53	66	.fh....fh....fSf
00000110	53	66	55	66	68	00	00	00	00	66	68	00	7C	00	00	66	SfUfh....fh. ..f
00000120	61	68	00	00	07	CD	1A	5A	32	F6	EA	00	7C	00	00	CD	ah...í.Z2öê. ..í
00000130	18	A0	B7	07	EB	08	A0	B6	07	EB	03	A0	B5	07	32	E4	. ..ë. ¶.ë. µ.2ä
00000140	05	00	07	8B	F0	AC	3C	00	74	09	BB	07	00	B4	0E	CD	...<ð~<.t.»...`í
00000150	10	EB	F2	F4	EB	FD	2B	C9	E4	64	EB	00	24	02	E0	F8	.ëòðéý+Éädë.\$.àø
00000160	24	02	C3	49	6E	76	61	6C	69	64	20	70	61	72	74	69	\$.ÃInvalid parti
00000170	74	69	6F	6E	20	74	61	62	6C	65	00	45	72	72	6F	72	tion table.Error
00000180	20	6C	6F	61	64	69	6E	67	20	6F	70	65	72	61	74	69	loading operati
00000190	6E	67	20	73	79	73	74	65	6D	00	4D	69	73	73	69	6E	ng system.Missin
000001A0	67	20	6F	70	65	72	61	74	69	6E	67	20	73	79	73	74	g operating syst
000001B0	65	6D	00	00	00	63	7B	9A	95	72	12	19	00	00	80	20	em...c{š*ř....€
000001C0	21	00	07	FE	FF	FF	00	08	00	00	00	F0	BF	03	00	FE	!..þÿ....ð.þ
000001D0	FF	FF	07	FE	FF	FF	00	F8	BF	03	00	38	EB	01	00	FE	ÿ.þÿ....ð.þ
000001E0	FF	FF	07	FE	FF	FF	00	30	AB	05	00	C0	D4	01	00	00	ÿ.þÿ.0«..ÀÖ...
000001F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	55	AAU*

From the table provided in the write-up we know that the last 4 bytes of the partition table entry contains the count of sectors in a partition.

Offset(h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	Decoded text
00000000	33 C0 8E D0 BC 00 7C 8E C0 8E D8 BE 00 7C BF 00	3ÀŽÐ¾. ŽÀŽØ¾. ž.
00000010	06 B9 00 02 FC F3 A4 50 68 1C 06 CB FB B9 04 00	.¹..üó¤Ph..ÉÙ¹..
00000020	BD BE 07 80 7E 00 00 7C 0B OF 85 0E 01 83 C5 10	¾.€~...fÅ.
00000030	E2 F1 CD 18 88 56 00 55 C6 46 11 05 C6 46 10 00	åñí.^V.UÆF..ÆF..
00000040	B4 41 BB AA 55 CD 13 5D 72 OF 81 FB 55 AA 75 09	'A»*UÍ.]r..úU*u.
00000050	F7 C1 01 00 74 03 FE 46 10 66 60 80 7E 10 00 74	-Á..t.pF.f`€~..t
00000060	26 66 68 00 00 00 00 66 FF 76 08 68 00 00 68 00	&fh....fÿv.h..h.
00000070	7C 68 01 00 68 10 00 B4 42 8A 56 00 8B F4 CD 13	h..h..`BŠV.<ðí.
00000080	9F 83 C4 10 9E EB 14 B8 01 02 BB 00 7C 8A 56 00	ÝfÅ.žë....». ŠV.
00000090	8A 76 01 8A 4E 02 8A 6E 03 CD 13 66 61 73 1C FE	Šv.ŠN.Šn.í.fas.p
000000A0	4E 11 75 0C 80 7E 00 80 OF 84 8A 00 B2 80 EB 84	N.u.€~.€..„Š.‘€ë..
000000B0	55 32 E4 8A 56 00 CD 13 5D EB 9E 81 3E FE 7D 55	U2äŠV.í.]ěž.>p}U
000000C0	AA 75 6E FF 76 00 E8 8D 00 75 17 FA B0 D1 E6 64	*unýv.è..u.ú°Ñæd
000000D0	E8 83 00 B0 DF E6 60 E8 7C 00 B0 FF E6 64 E8 75	èf.ºBæ`è .ºÿædèu
000000E0	00 FB B8 00 BB CD 1A 66 23 C0 75 3B 66 81 FB 54	.ù..»í.f#Àu;f.ùT
000000F0	43 50 41 75 32 81 F9 02 01 72 2C 66 68 07 BB 00	CpAu2.ù..r,fh..».
00000100	00 66 68 00 02 00 00 66 68 08 00 00 00 66 53 66	.fh....fh....fSf
00000110	53 66 55 66 68 00 00 00 00 66 68 00 7C 00 00 66	SfUfh....fh. ..f
00000120	61 68 00 00 07 CD 1A 5A 32 F6 EA 00 7C 00 00 CD	ah....í.Z2öè. ..í
00000130	18 A0 B7 07 EB 08 A0 B6 07 EB 03 A0 B5 07 32 E4	. . .é. ¶.é. µ.2ä
00000140	05 00 07 8B F0 AC 3C 00 74 09 BB 07 00 B4 0E CD	...<ð~<.t..»..‘.í
00000150	10 EB F2 F4 EB FD 2B C9 E4 64 EB 00 24 02 E0 F8	.ëðöëý+Éädë.\$.àø
00000160	24 02 C3 49 6E 76 61 6C 69 64 20 70 61 72 74 69	\$.ÃInvalid parti
00000170	74 69 6F 6E 20 74 61 62 6C 65 00 45 72 72 6F 72	tion table.Error
00000180	20 6C 6F 61 64 69 6E 67 20 6F 70 65 72 61 74 69	loading operati
00000190	6E 67 20 73 79 73 74 65 6D 00 4D 69 73 73 69 6E	ng system.Missin
000001A0	67 20 6F 70 65 72 61 74 69 6E 67 20 73 79 73 74	g operating syst
000001B0	65 6D 00 00 00 63 7B 9A 95 72 12 19 00 00 80 20	em...c{š•r....€
000001C0	21 00 07 FE FF FF 00 08 00 00 00 F0 BF 03 00 FE	!...þÿ....ði...þ
000001D0	FF FF 07 FE FF FF 00 F8 BF 03 00 38 EB 01 00 FE	ÿ.þÿ.øi.þë.þ
000001E0	FF FF 07 FE FF FF 00 30 AB 05 00 C0 D4 01 00 00	ÿ.þÿ.0«..Àô...
000001F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AAU^

The value store in the partition table uses little-endian representation, we need to reverse the bytes to get the big-endian representation so that we can convert it to decimal.

00 38 EB 01 becomes 01 EB 38 00 in big-endian representation. When this hex value is converted to decimal we get 3219512. We know that each sector is 512 bytes. To get the size of the partition we need to multiply the sector count with the sector size. This gives us 16482566144 (32192512 * 512). This is the size of the partition in bytes.

01EB3800

32192512₁₀ 172634000₈ 11110101100111000000000000₂**i Note**

Disk size can be represented as a multiple of 2 (1 KB = 1024 bytes) or multiple of 10 (1 KB = 1000 bytes). The notation that is used depends on the OS. Windows and Linux use the binary notation (1024 bytes) while MacOS uses the decimal notation (1000 bytes). Most disk manufacturers also use the decimal notation as it makes their disk appear larger.

This room expects the sizes in multiples of 10. So to convert bytes to gigabytes we have to divide the 16482566144 twice by 1000 (B → KB → MB → GB).

$$(((32192512 * 512) / 1000) / 1000) / 1000$$
16.482566144

16

Task 4: Threats Targeting MBR

1. Complete this task.

No answer required**Task 5: MBR Tampering Case**

In this task we have been instructed to fix `MBR_Corrupted_Disk`. On loading the image in HxD we can see that the Magic Number of the MBR partition is corrupted (correct

value is 55 AA). We are also told in the write-up that the LBA field of the bootable partition has been corrupted.

Offset (h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	Decoded text
0000000000	B3 C0 8E D0 BC 00 7C 8E C0 8E D8 BE 00 7C BF 00	3ÀŽĐ4. ŽÀŽØ%. č.
0000000010	06 B9 00 02 FC F3 A4 50 68 1C 06 CB FB B9 04 00	.¹..üó¤Ph..Ëü¹..
0000000020	BD BE 07 80 7E 00 00 7C 0B 0F 05 0E 01 83 C5 10	¾.€~...fÅ.
0000000030	E2 F1 CD 18 88 56 00 55 C6 46 11 05 C6 46 10 00	áñí.^V.UÆF..ÆF..
0000000040	B4 41 BB AA 55 CD 13 5D 72 0F 81 FB 55 AA 75 09	’A»“Ui.]r..ÛUºu.
0000000050	F7 C1 01 00 74 03 FE 46 10 66 60 80 7E 10 00 74	±Á..t.pF.f`€~..t
0000000060	26 66 68 00 00 00 66 FF 76 08 68 00 00 68 00	&fh....fýv.h..h.
0000000070	7C 68 01 00 68 10 00 B4 42 8A 56 00 8B F4 CD 13	h..h.. ‘BŠV. <x>8í.</x>
0000000080	9F 83 C4 10 9E EB 14 B8 01 02 BB 00 7C 8A 56 00	ÝfÅ.žë.,...». ŠV.
0000000090	8A 76 01 8A 4E 02 8A 6E 03 CD 13 66 61 73 1C FE	Šv.ŠN.Šn.Í.fas.þ
0000000A0	4E 11 75 0C 80 7E 00 80 0F 84 8A 00 B2 80 EB 84	N.u.€~.€..„š.‘€č,
0000000B0	55 32 E4 8A 56 00 CD 13 5D EB 9E 81 3E FE 7D 55	U2äŠV.Í.]ěž.>p)U
0000000C0	AA 75 6E FF 76 00 E8 8D 00 75 17 FA B0 D1 E6 64	“unýv.è..u.ú°Ñæd
0000000D0	E8 83 00 B0 DF E6 60 E8 7C 00 B0 FF E6 64 E8 75	èf.‘Bæ`è .‘yædèu
0000000E0	00 FB B8 00 BB CD 1A 66 23 C0 75 3B 66 81 FB 54	.û..»í.f#Au;f.ûT
0000000F0	43 50 41 75 32 81 F9 02 01 72 2C 66 68 07 BB 00	CPAu2.û..r,fh.».
000000100	00 66 68 00 02 00 00 66 68 08 00 00 00 66 53 66	.fh....fh....fSf
000000110	53 66 55 66 68 00 00 00 00 66 68 00 7C 00 00 66	SfUfh....fh. ..f
000000120	61 68 00 00 07 CD 1A 5A 32 F6 EA 00 7C 00 00 CD	ah...í.z2öé. ..í
000000130	18 A0 B7 07 EB 08 A0 B6 07 EB 03 A0 B5 07 32 E4	. .-ë. ¶.ë. µ.2ä
000000140	05 00 07 8B F0 AC 3C 00 74 09 BB 07 00 B4 0E CD	...<ð-<.t.»..‘í
000000150	10 EB F2 F4 EB FD 2B C9 E4 64 EB 00 24 02 E0 F8	.ëòðëý+Éädë.Ş.àø
000000160	24 02 C3 49 6E 76 61 6C 69 64 20 70 61 72 74 69	Ş.ÀInvalid parti
000000170	74 69 6F 6E 20 74 61 62 6C 65 00 45 72 72 6F 72	tion table.Error
000000180	20 6C 6F 61 64 69 6E 67 20 6F 70 65 72 61 74 69	loading operati
000000190	6E 67 20 73 79 73 74 65 6D 00 4D 69 73 73 69 6E	ng system.Missin
0000001A0	67 20 6F 70 65 72 61 74 69 6E 67 20 73 79 73 74	g operating syst
0000001B0	65 6D 00 00 00 63 7B 9A CF AD 76 E5 00 00 80 20	em...c{šI.vå..€
0000001C0	21 00 07 FE FF FF FF FF FF 00 F0 BF 03 00 00	!..þÿÿÿÿÿÿ.ðç...
0000001D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000001E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000001F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF FFyy
000000200	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

The values can be modified by double clicking in the hex editor and then typing the new value. Once done save the file to persist the changes.

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
0000000000	B3	C0	8E	D0	BC	00	7C	8E	C0	8E	D8	BE	00	7C	BF	00	3AŽĐ4. ŽÀŽØ%. č.
0000000010	06	B9	00	02	FC	F3	A4	50	68	1C	06	CB	FB	B9	04	00	. „.üó¤Ph..Éú¹..
0000000020	BD	BE	07	80	7E	00	00	7C	0B	0F	85	0E	01	83	C5	10	¾.€~...fÅ.
0000000030	E2	F1	CD	18	88	56	00	55	C6	46	11	05	C6	46	10	00	áñí.~V.UÆF..ÆF..
0000000040	B4	41	BB	AA	55	CD	13	5D	72	0F	81	FB	55	AA	75	09	'A»*UÍ.]r..ÛU*u.
0000000050	F7	C1	01	00	74	03	FE	46	10	66	60	80	7E	10	00	74	÷Á..t.pF.f`€~..t
0000000060	26	66	68	00	00	00	66	FF	76	08	68	00	00	68	00	00	&fh....fýv.h..h.
0000000070	7C	68	01	00	68	10	00	B4	42	8A	56	00	8B	F4	CD	13	h..h.. BŠV.<ðí.
0000000080	9F	83	C4	10	9E	EB	14	B8	01	02	BB	00	7C	8A	56	00	ÝfÄ.žě.,...». ŠV.
0000000090	8A	76	01	8A	4E	02	8A	6E	03	CD	13	66	61	73	1C	FE	Šv.ŠN.Šn.í.fas.p
00000000A0	4E	11	75	0C	80	7E	00	80	0F	84	8A	00	B2	80	EB	84	N.u.€~.€..„š..“€„..
00000000B0	55	32	E4	8A	56	00	CD	13	5D	EB	9E	81	3E	FE	7D	55	U2äŠV.í.]jěz.>p}U
00000000C0	AA	75	6E	FF	76	00	E8	8D	00	75	17	FA	B0	D1	E6	64	*unýv.è..u.ú°Ñæd
00000000D0	E8	83	00	B0	DF	E6	60	E8	7C	00	B0	FF	E6	64	E8	75	èf.ºBæ`è .ºýædèu
00000000E0	00	FB	B8	00	BB	CD	1A	66	23	C0	75	3B	66	81	FB	54	.ù..»í.f#Au;f.ùT
00000000F0	43	50	41	75	32	81	F9	02	01	72	2C	66	68	07	BB	00	CPAu2.ù..r,fh.».
000000100	00	66	68	00	02	00	00	66	68	08	00	00	00	66	53	66	.fh....fh....fSf
000000110	53	66	55	66	68	00	00	00	00	66	68	00	7C	00	00	66	SfUfh....fh. ..f
000000120	61	68	00	00	07	CD	1A	5A	32	F6	EA	00	7C	00	00	CD	ah...í.ZZöë. ..í
000000130	18	A0	B7	07	EB	08	A0	B6	07	EB	03	A0	B5	07	32	E4	. ..ě. ¶.ě. µ.2ä
000000140	05	00	07	8B	F0	AC	3C	00	74	09	BB	07	00	B4	0E	CD	...<ð-<.t.»..í
000000150	10	EB	F2	F4	EB	FD	2B	C9	E4	64	EB	00	24	02	E0	F8	.ëòöëý+Éädë.Ş.àø
000000160	24	02	C3	49	6E	76	61	6C	69	64	20	70	61	72	74	69	\$.ÃInvalid parti
000000170	74	69	6F	6E	20	74	61	62	6C	65	00	45	72	72	6F	72	tion table.Error
000000180	20	6C	6F	61	64	69	6E	67	20	6F	70	65	72	61	74	69	loading operati
000000190	6E	67	20	73	79	73	74	65	6D	00	4D	69	73	73	69	6E	ng system.Missin
0000001A0	67	20	6F	70	65	72	61	74	69	6E	67	20	73	79	73	74	g operating syst
0000001B0	65	6D	00	00	00	63	7B	9A	CF	AD	76	E5	00	00	80	20	em...c(šI.vå..€
0000001C0	21	00	07	FE	FF	FF	00	08	00	00	F0	BF	03	00	00	00	!..þý....ð¿...
0000001D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000001F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00U*
000000200	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

1. How many partitions are on the disk?

If we look at the above images we can see that only the first 16 bytes in the partition table has any value. Every byte after that is set to 0. From this we can conclude that this disk only has a single partition.

1

2. What is the first byte at the starting LBA of the partition? (represented by two hexadecimal digits)

From the write-up we know that the LBA address field contains the sector from which the data for this partition starts. Since this value is in little-endian we need to reverse it to get the big-endian notation. Once its reversed we can calculate the decimal value. The data for the partition starts at sector 2048 .

00000800|

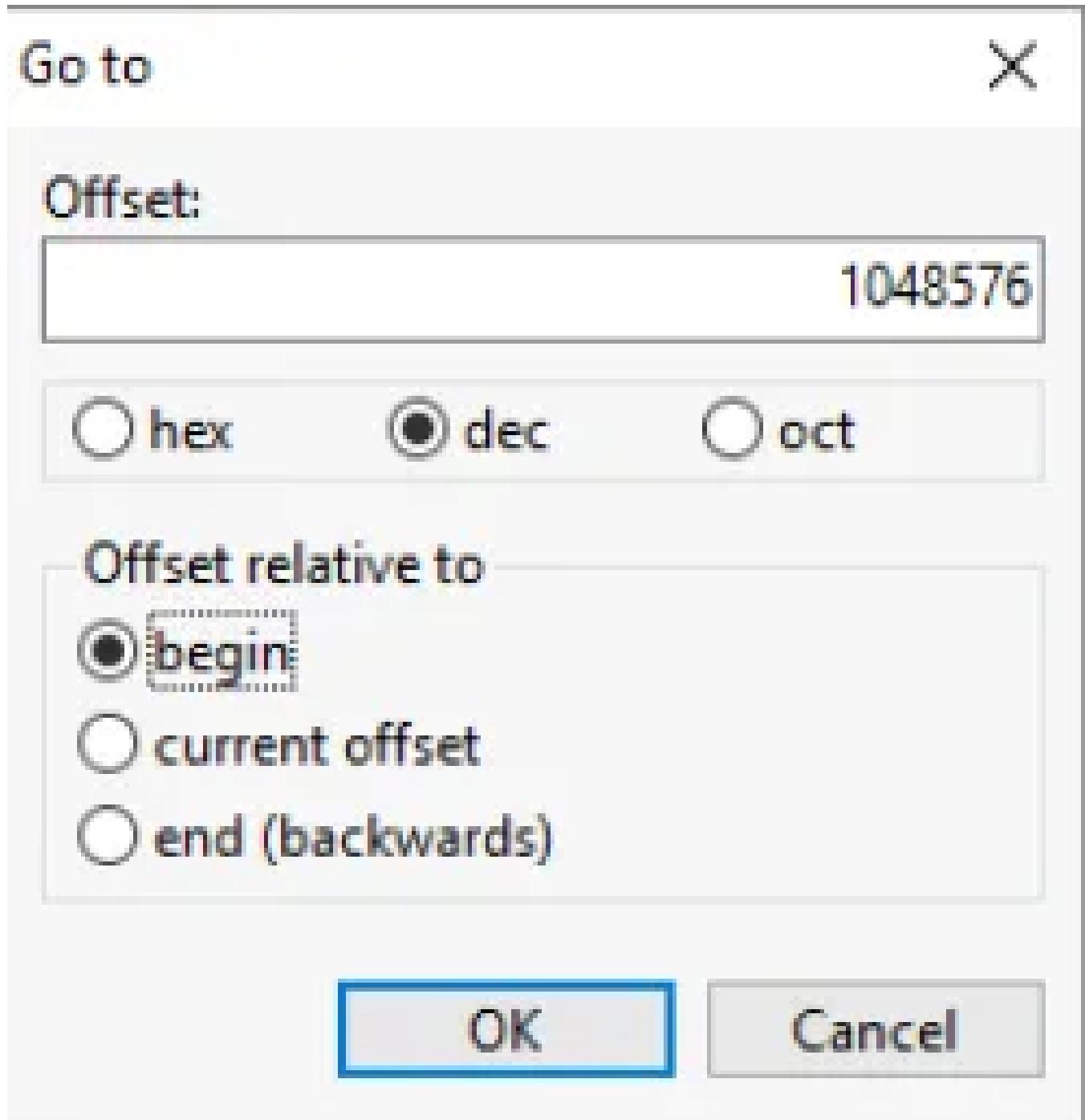
2048₁₀ 4000₈ 100000000000₂

We need the address, for that we need to multiply the value by 512 (sector size) which gives us 1048576 .

2048*512

1,048,576

We can use “Search → Go to” to jump to the address.



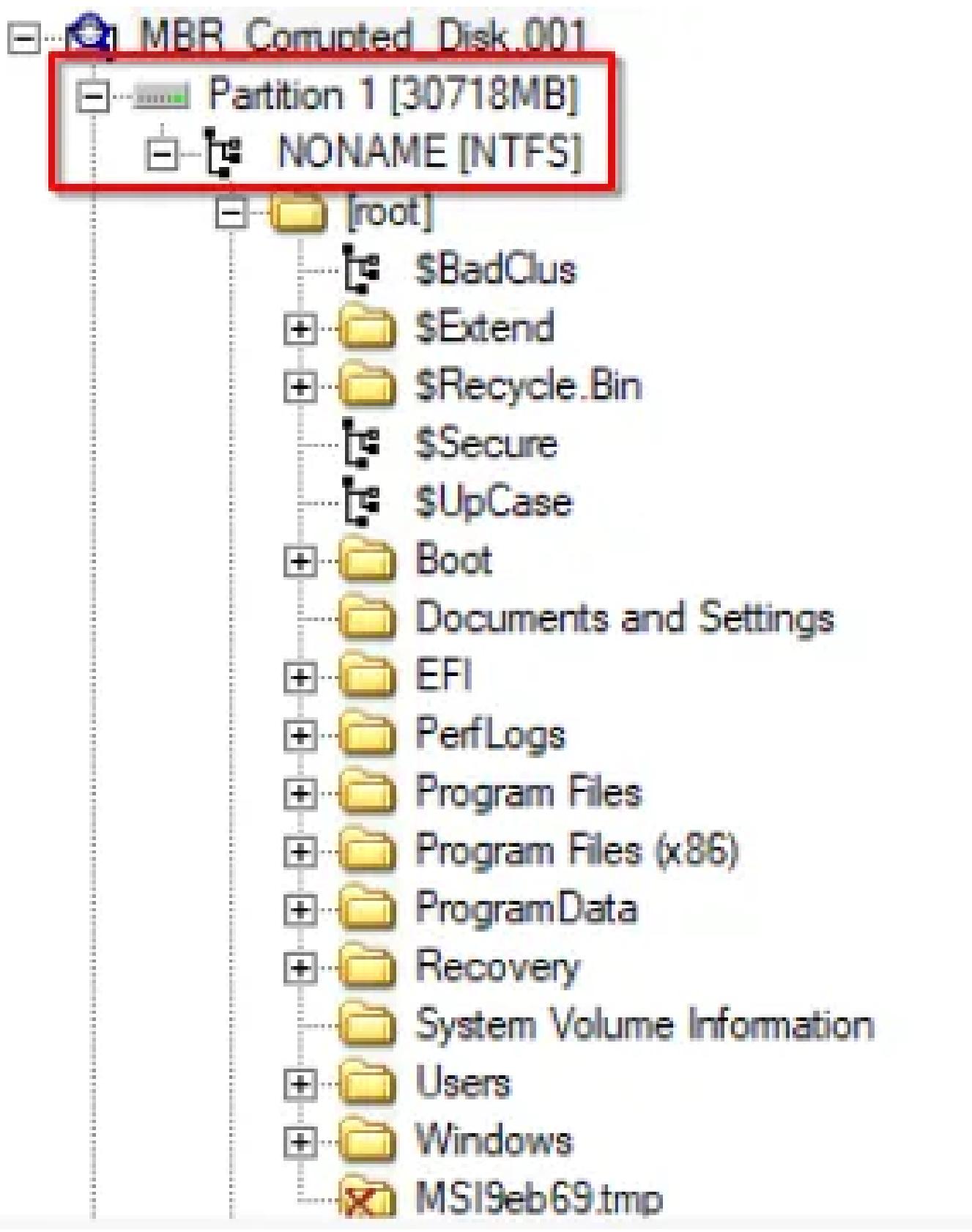
We can see that the 1st byte at this address has the value `EB`.

0000FFFF0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000100000	EB 52 90 4E 54 46 53 20 20 20 20 00 02 08 00 00	ER NTFS
000100010	00 00 00 00 00 F8 00 00 3F 00 FF 00 00 08 00 00ø.?ÿ....
000100020	00 00 00 00 80 00 80 00 FF EF BF 03 00 00 00 00€.€.ÿiÿ....
000100030	00 00 0C 00 00 00 00 00 02 00 00 00 00 00 00 00
000100040	F6 00 00 00 01 00 00 00 98 C0 23 A0 D5 23 A0 90	ö.....~À# Ö# .
000100050	00 00 00 00 FA 33 C0 8E D0 BC 00 7C FB 68 C0 07ú3ÀŽĐ4. ÚhÀ.
000100060	1F 1E 68 66 00 CB 88 16 0E 00 66 81 3E 03 00 4E	..hf.È^...f.>..N
000100070	54 46 53 75 15 B4 41 BB AA 55 CD 13 72 0C 81 FB	TFSu.'A»*Úí.r..Ù
000100080	55 AA 75 06 F7 C1 01 00 75 03 E9 DD 00 1E 83 EC	U*u.÷Á..u.éÝ..fi
000100090	18 68 1A 00 B4 48 8A 16 0E 00 8B F4 16 1F CD 13	.h...'HŠ...<ô..í.
0001000A0	9F 83 C4 18 9E 58 1F 72 E1 3B 06 0B 00 75 DB A3	ÝfÄ.žX.rá;...uÛ£
0001000B0	0F 00 C1 2E 0F 00 04 1E 5A 33 DB B9 00 20 2B C8	..Á.....Z3Û¹. +È
0001000C0	66 FF 06 11 00 03 16 0F 00 8E C2 FF 06 16 00 E8	fÿ.....ŽÂý...è
0001000D0	4B 00 2B C8 77 EF B8 00 BB CD 1A 66 23 C0 75 2D	K.+Èwí.,»í.f#Àu-
0001000E0	66 81 FB 54 43 50 41 75 24 81 F9 02 01 72 1E 16	f.ÛTCPAu\$.ù...r..
0001000F0	68 07 BB 16 68 52 11 16 68 09 00 66 53 66 53 66	h.»..hR..h..fSfSf
000100100	55 16 16 16 68 B8 01 66 61 0E 07 CD 1A 33 C0 BF	U...h..fa..í.3Àí
000100110	0A 13 B9 F6 0C FC F3 AA E9 FE 01 90 90 66 60 1E	..÷ö.üó*éþ...f`.
000100120	06 66 A1 11 00 66 03 06 1C 00 1E 66 68 00 00 00	.f...f.....fh...
000100130	00 66 50 06 53 68 01 00 68 10 00 B4 42 8A 16 0E	.fP.Sh..h..'BŠ..
000100140	00 16 1F 8B F4 CD 13 66 59 5B 5A 66 59 66 59 1F	...<öÍ.fY[ZfYfY.
000100150	0F 82 16 00 66 FF 06 11 00 03 16 0F 00 8E C2 FF	,...fÿ.....ŽÂý
000100160	0E 16 00 75 BC 07 1F 66 61 C3 A1 F6 01 E8 09 00	...u4..faÃ;ö.è..
000100170	A1 FA 01 E8 03 00 F4 EB FD 8B F0 AC 3C 00 74 09	;ú.è..öëý<ö~<.t.
000100180	B4 0E BB 07 00 CD 10 EB F2 C3 0D 0A 41 20 64 69	'..»..í.ëòÃ..A di
000100190	73 6B 20 72 65 61 64 20 65 72 72 6F 72 20 6F 63	sk read error oc
0001001A0	63 75 72 72 65 64 00 0D 0A 42 4F 4F 54 4D 47 52	curred...BOOTMGR
0001001B0	20 69 73 20 63 6F 6D 70 72 65 73 73 65 64 00 0D	is compressed..
0001001C0	0A 50 72 65 73 73 20 43 74 72 6C 2B 41 6C 74 2B	.Press Ctrl+Alt+
0001001D0	44 65 6C 20 74 6F 20 72 65 73 74 61 72 74 0D 0A	Del to restart..

EB

3. What is the type of the partition?

In the above image if we look at the ASCII column we can see that the partition uses NTFS. Alternatively, we can load the modified image in FTK Imager to get the partition type.



NTFS

4. What is the size of the partition? (rounded to the nearest GB)

FTK shows us the size of the partition in MB besides the partition name. FTK shows the size in binary notation but in this room we are required to provide the sizes in

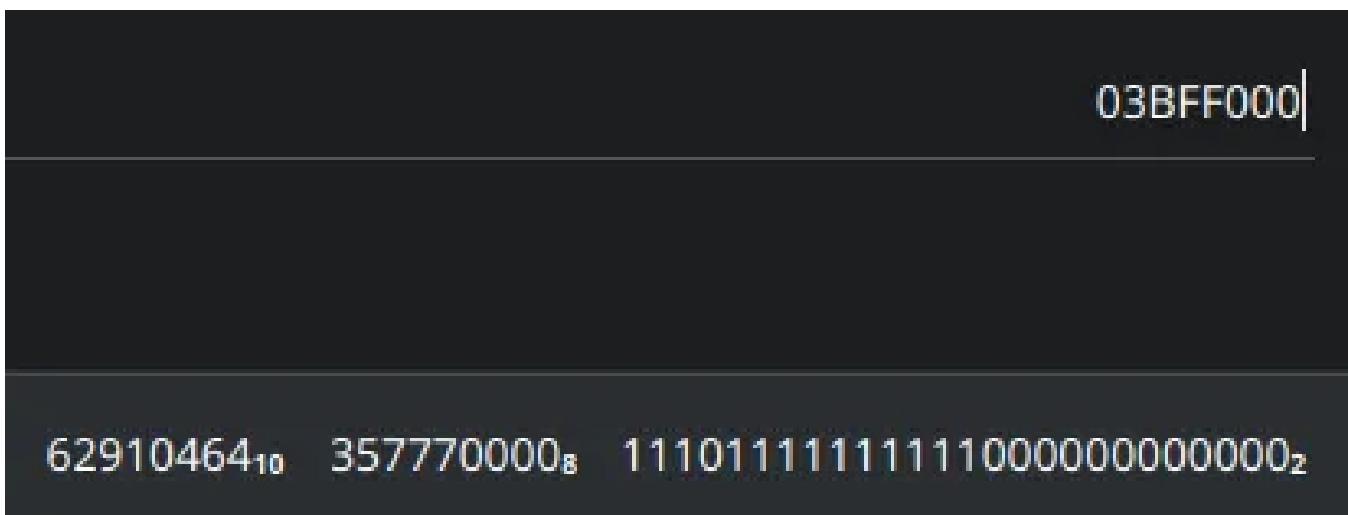
decimal notation. So, we first need to multiply the size twice by 1024 to get the size in bytes then we can divide it thrice by 1000 to get the size in GB.



An alternative approach would be to use the value provided in the no. of sectors field (last field) in the partition entry.

000000170	74	69	6F	6E	20	74	61	62	6C	65	00	45	72	72	6F	72	tion table.Error
000000180	20	6C	6F	61	64	69	6E	67	20	6F	70	65	72	61	74	69	loading operati
000000190	6E	67	20	73	79	73	74	65	6D	00	4D	69	73	73	69	6E	ng system.Missin
0000001A0	67	20	6F	70	65	72	61	74	69	6E	67	20	73	79	73	74	g operating syst
0000001B0	65	6D	00	00	00	63	7B	9A	CF	AD	76	E5	00	00	80	20	em...c{ší.vå...€
0000001C0	21	00	07	FE	FF	FF	00	08	00	00	00	F0	B0	03	00	00	!...þý....ðč... .
0000001D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Flip the bits to get the big-endian representation, then convert the value to decimal. This will give us 62910464 . This is the no. of sectors that make up the partition to get the size we need to multiple this value by 512 which will give us the size in bytes. We divide this value thrice by 1000 to get the size in GB.



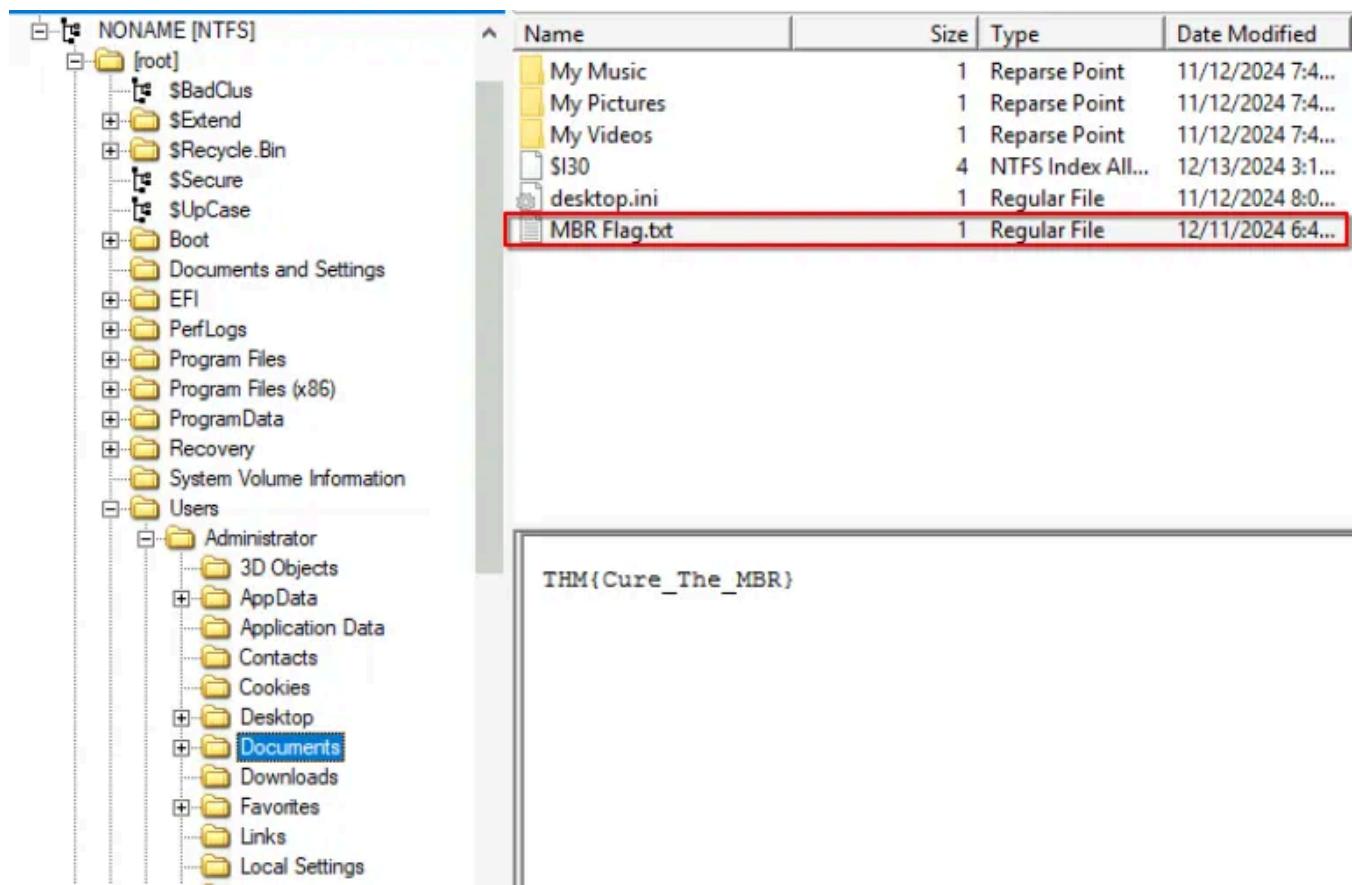
$$(((62910464 * 512) / 1000) / 1000) / 1000$$

32.210157568

32

5. What is the flag hidden in the Administrator's Documents folder?

The Documents folder is located under the users home directory. The home directory for all users are stored in `Users`. In this directory there is a file called `MBR Flag.txt` which contains the flag.



THM{Cure_The_MBR}

Task 6: What if GPT?

1. How many partitions are supported by the GPT?

Previously, we studied how the boot process propagates with the MBR. In this task, we will explore what the boot process looks like when a modern GPT partitioning scheme is used in the disk instead of the MBR.

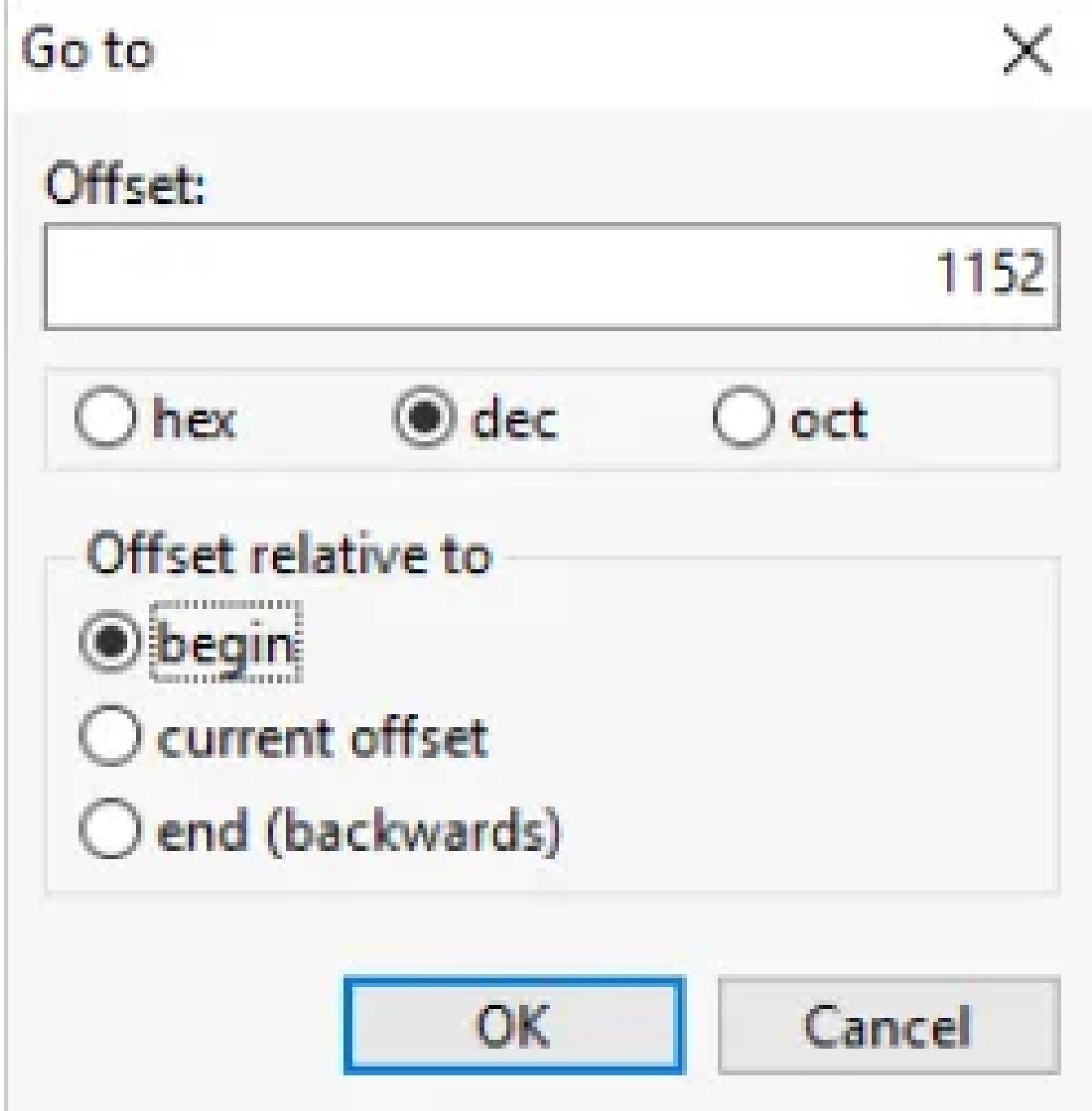
UEFI firmware with the GPT partitioning scheme replaced BIOS firmware with the MBR partitioning scheme. This replacement occurred due to some limitations in the BIOS and the MBR. The GPT supports up to 9 zettabytes of hard disks, unlike the MBR, which supports a maximum of 2 terabytes. The GPT also supports up to 128 partitions, unlike the MBR, which only supports 4. There are some other differences between both of the partitioning schemes.

128

2. What is the partition type GUID of the 2nd partition given in the attached GPT file?

From the write-up we learn that the partition related metadata is stored in the GPT Partition Array. The partition array is located in the 3rd sector on the disk. Each entry in the partition array is 128 bytes. Each sector on the disk is 512 bytes.

We can get the starting address for the 3rd sector by calculating $512 * 2$. We multiply by 2 as the sector count starts from 0 so the 3rd sector becomes the 2nd sector. To this value we need to add 128 (size of entry in partition array) to get the start of the 2nd entry in the partition array. This comes out to 1152 ($(512 * 2) + 128$).



From the table provided in the write-up we know that the first 16 bytes represent the partition type GUID. By following the instructions provided we can get the GUID in the correct format.

00000470	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000480	16 E3 C9 E3 5C 0B B8 4D 81 7D F9 2D F0 02 15 AE EA C7 1A 46 F2 2B 9E 42 89 A7 6C 7A A8 27 6C D9	.ÄEÄ\..M.)Ù-ä..@ êÇ.Fò+žB‰\$lz''1Ù
00000490	00 28 03 00 00 00 00 00 FF A7 03 00 00 00 00 00 00 00 00	.(.....ýS.....
000004A0	00 00 00 00 00 00 00 80 4D 00 69 00 63 00 72 00€M.i.c.r.
000004B0	6F 00 73 00 6F 00 66 00 74 00 20 00 72 00 65 00	o.s.o.f.t. .r.e.
000004C0	73 00 65 00 72 00 76 00 65 00 64 00 20 00 70 00	s.e.r.v.e.d. .p.
000004D0	61 00 72 00 74 00 69 00 74 00 69 00 6F 00 6E 00	a.r.t.i.t.i.o.n.
000004E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000004F0	A2 A0 D0 EB E5 B9 33 44 87 C0 68 B6 B7 26 99 C7	© Đëä'3D‡Àh¶·&MC
00000500	BD 27 24 5C 43 A7 2B 4C 8B 5B 7D EB DB 08 D2 20	÷'‰\CS+L<[]}éÙ.Ø
00000510	00 A8 03 00 00 00 00 00 4E 0C E0 0D 00 00 00 00 00 00N.à.....
00000520		

E3C9E316-0B5C-4DB8-817D-F92DF00215AE

Task 7: Threats Targeting GPT

- ## 1. Complete this task

No answer required

Task 8: UEFI Bootkit Case

- ## 1. Which partition has the bootloader in it?

When we use BIOS firmware, the bootloader responsible for loading the operating system is executed inside the MBR. However, when we use UEFI firmware, which supports a GPT partitioning scheme, the bootloader is not executed from the GPT. Instead, it is located in the **EFI system partition (ESP)** in the form of .efi extension files. These files include the **bootmgr.efi**, **bootx64.efi**, and others, depending on the OS.

EFI System Partition

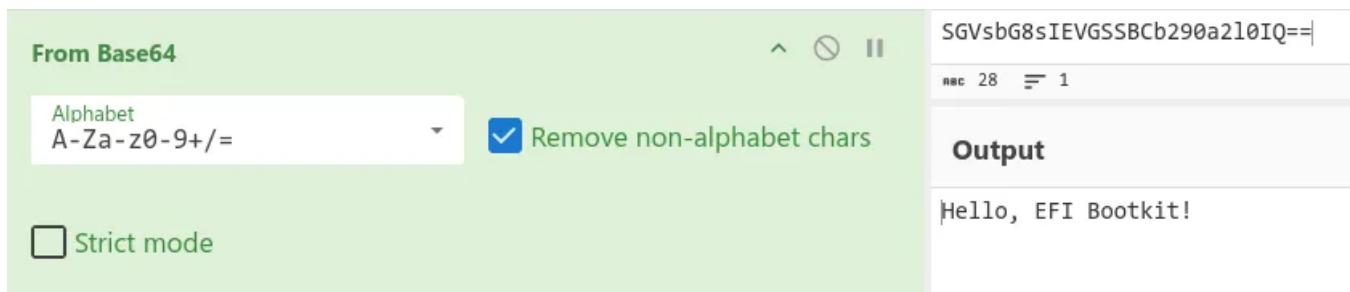
- ## 2. What is the malicious string embedded in the bootloader?

For this task we need to use `bootmgr.efi`. On scrolling a few lines I noticed a string that ended in “==”. Strings that end with equal signs are almost always base64 encoded.

SGVsbG8sIEVGSSBCb290a2l0IQ==

On decoding the string we get the malicious message.

CyberChef — Decode Recipe



The screenshot shows the CyberChef interface. On the left, under 'From Base64', the input is 'SGVsbG8sIEVGSSBCb290a2l0IQ==' with an 'Alphabet' dropdown set to 'A-Za-z0-9+/=' and a checked 'Remove non-alphabet chars' checkbox. Below that is an unchecked 'Strict mode' checkbox. On the right, under 'Output', the result is 'Hello, EFI Bootkit!'. There are also 'rec 28' and 'l 1' status indicators.

Open in app ↗

Medium



Search



If you found this write-up to be useful consider:

- **Liking the Post:** You can like (applaud) the post up to 50 times.
- **Leave a Comment:** Your feedback and comments are invaluable. They not only support me, but also enrich the discussion for other readers.
- **Share the Article:** If you believe others would benefit from this article, please consider sharing it within your network.
- **Follow Me:** Feel free to connect with me on [Medium](#), [LinkedIn](#), [GitHub](#) and [Discord](#).
- **Explore More:** All of my posts can also be accessed on [Source Code](#).

Thank you for your time. Your support motivates me to continue writing.

Security

Tryhackme

Walkthrough

Forensics

Operating Systems

Some rights reserved

[Follow](#)

Written by David Varghese

1.3K Followers · 22 Following

Cybersecurity Student | Digital Forensics | Security Analyst | Software Engineering | davidvarghese.net

No responses yet



What are your thoughts?

[Respond](#)

More from David Varghese



David Varghese

OverTheWire: Bandit Level 16 → Level 17

<https://overthewire.org/wargames/bandit/bandit17.html>

Mar 7, 2021 78 2



David Varghese

OverTheWire: Bandit Level 12 → Level 13

<https://overthewire.org/wargames/bandit/bandit13.html>

Mar 6, 2021 49





David Varghese

OverTheWire: Bandit Level 5 → Level 6

<https://overthewire.org/wargames/bandit/bandit6.html>

Mar 4, 2021 2



...



David Varghese

OverTheWire: Bandit Level 23 → Level 24

<https://overthewire.org/wargames/bandit/bandit24.html>

Mar 17, 2021

6 3

...

[See all from David Varghese](#)

Recommended from Medium



In Offensive Black Hat Hacking & Security by Harshad Shah

Cybersecurity Roadmap 2025

How to start cybersecurity in 2025?



Dec 14, 2024

1121

...



In System Weakness by Sunny Singh Verma [SuNnY]

Silver Platter TryHackMe Motion Graphics Writeup | Beginner Friendly | Detailed Walkthrough |...

A Detailed motion Graphics writeup for TryHackMe room Silver Platter



Jan 13



50



...

Lists



Staff picks

804 stories · 1587 saves



Stories to Help You Level-Up at Work

19 stories · 924 saves



Self-Improvement 101

20 stories · 3239 saves



Productivity 101

20 stories · 2740 saves

 Hammazahmed

Snort: A Step-by-Step Guide to Writing and Testing Simple Rules

What to Expect

Oct 11, 2024  4



...



 In OSINT Team by Karthikeyan Nagaraj

Top 10 Websites Every Cybersecurity Professional Should Bookmark

A Curated List of Resources for Learning, Practicing, and Staying Updated

 6d ago  112  2



...

 Jessica Stillman

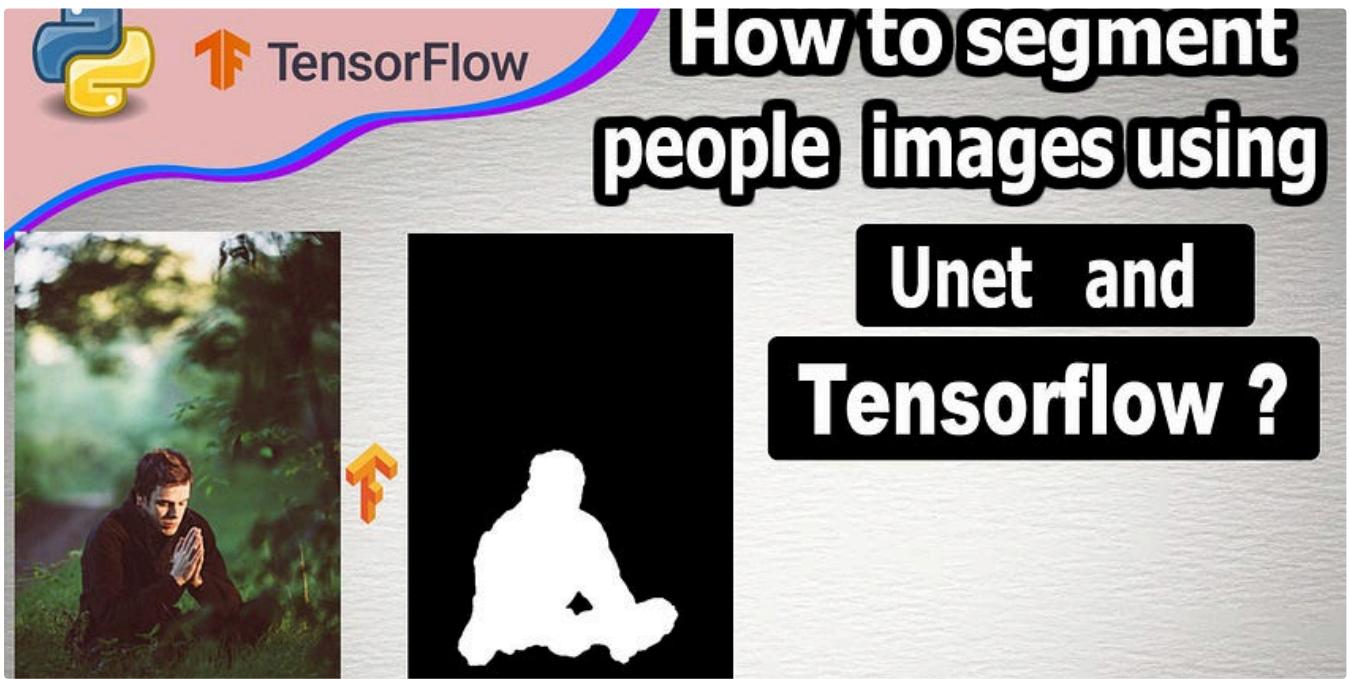
Jeff Bezos Says the 1-Hour Rule Makes Him Smarter. New Neuroscience Says He's Right

Jeff Bezos's morning routine has long included the one-hour rule. New neuroscience says yours probably should too.

♦ Oct 30, 2024 • 20K • 546

[+]

...



The image shows a blog post cover. At the top left are the Python logo and the TensorFlow logo. To the right of the logos is the title "How to segment people images using Unet and Tensorflow ?". Below the title are two images: a color photograph of a person sitting in a forest, and a binary segmentation mask where the person is highlighted in white against a black background. A small TensorFlow logo is positioned between the two images.

 Eran Feit

U-net Image Segmentation - How to segment persons in images

Summary :

★ Jan 3 🙌 10



...

See more recommendations