

PE & Windows Basics

Malware Deconstructed:
Episode 1

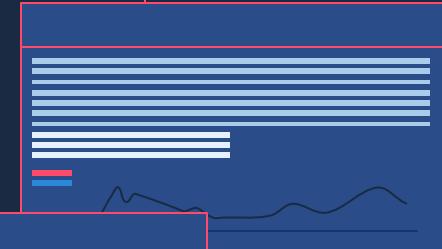


Table of Content

01 Preface

04 Common Windows Security Concepts

02 Preparing our environment

05 What now?

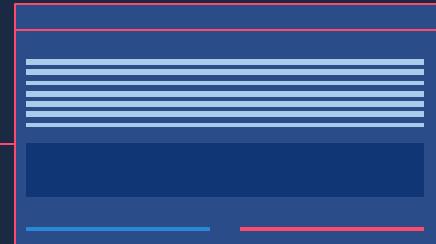
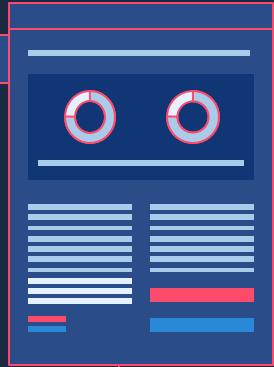
03 PE structure





01

Preface

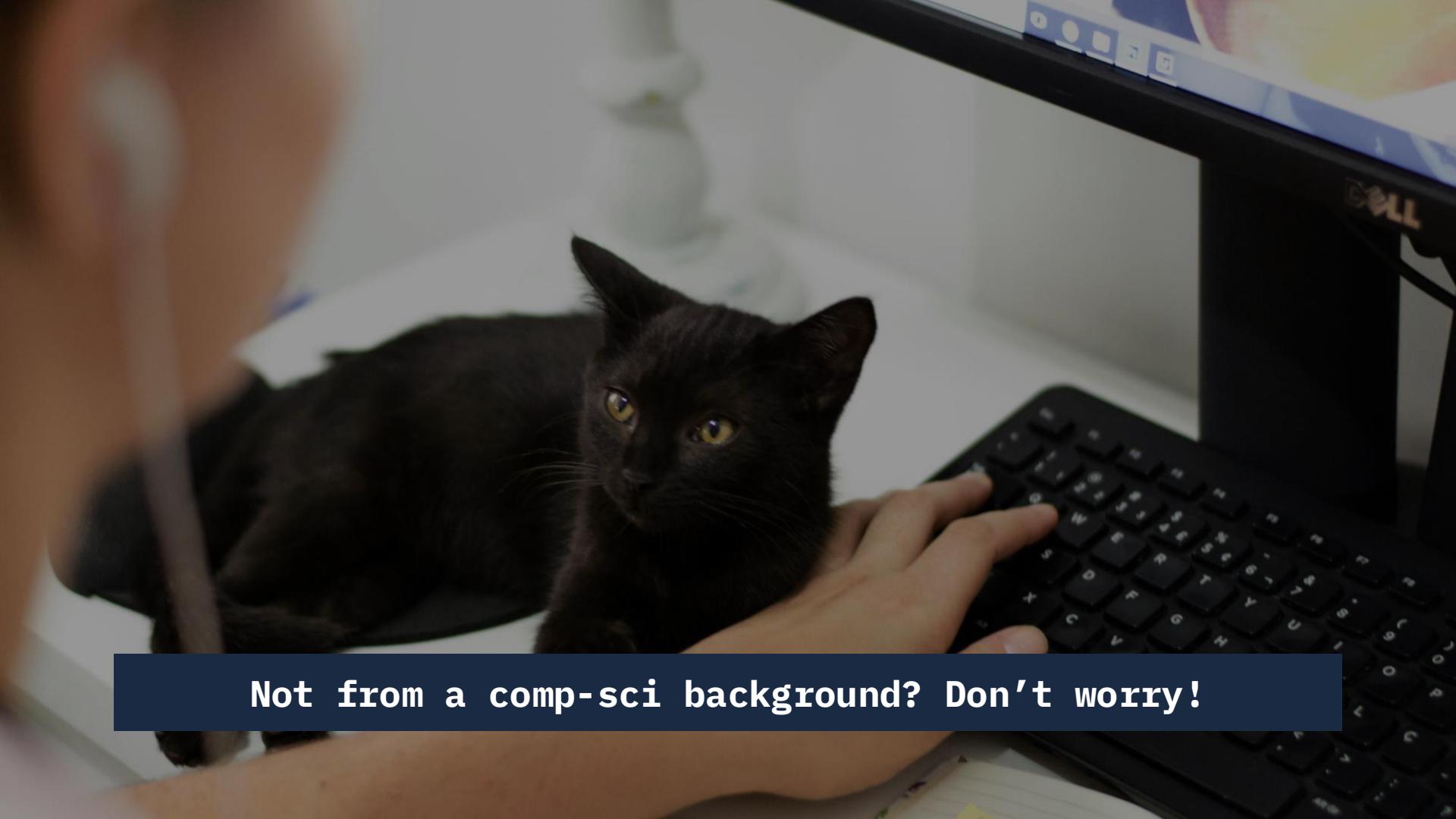


whoami

Azaka / Still

- they/them
- VTuber
- Taiwanese
- Full-time threat intelligence researcher
for 3+ years



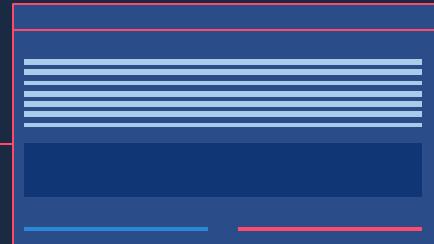
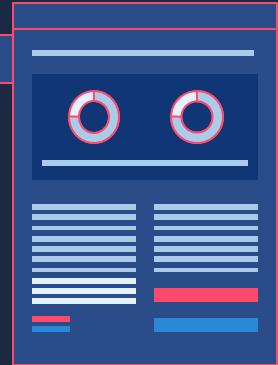
A black cat with yellow eyes is sitting on a person's lap. The person's hand is resting on a black laptop keyboard. The laptop screen shows a blue interface with icons. The Dell logo is visible on the top right corner of the screen.

Not from a comp-sci background? Don't worry!



02

Preparing the environment



Prepare a Safe Environment

Virtual Machine / Sandbox

- Free
 - Hyper-V
 - Only recommended if child partition is Windows 10 or above due to Generation 2 compatibility
 - Qemu
 - Relatively difficult to get into but highly flexible
 - VirtualBox
- Paid
 - VMware Workstation/ESXi
 - Perpetual licenses are being transitioned into subscription-based models



Prepare a Safe Environment

OS for Guests/Child Partitions

- Something not too old or too new
 - Preferably something at least 2 years old
 - Older Windows 10 builds
 - e.g. 1809, 1903, etc.
 - Windows 7
 - Older builds of Debian/CentOS
- Software
 - Office 2016 or below without updates
 - Analysis tools



Prepare a Safe Environment

No network if you're not confident!

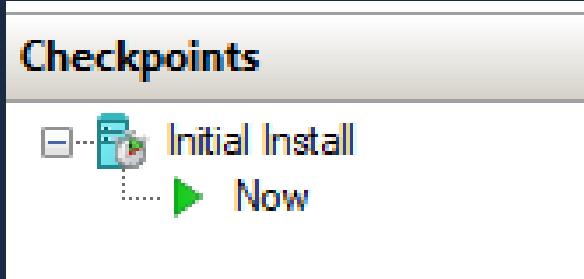
- DO NOT connect the child OS to network of any kind unless you are prepared.
 - The networking should have sufficient quarantining.
 - The child should NOT be able to...
 - Get your real WAN IP address
 - Access any other devices on the intranet
- Unless you are 100% confident that you are sufficiently tunneled and quarantined, DO NOT let it go online.



Prepare a Safe Environment

Make a Checkpoint

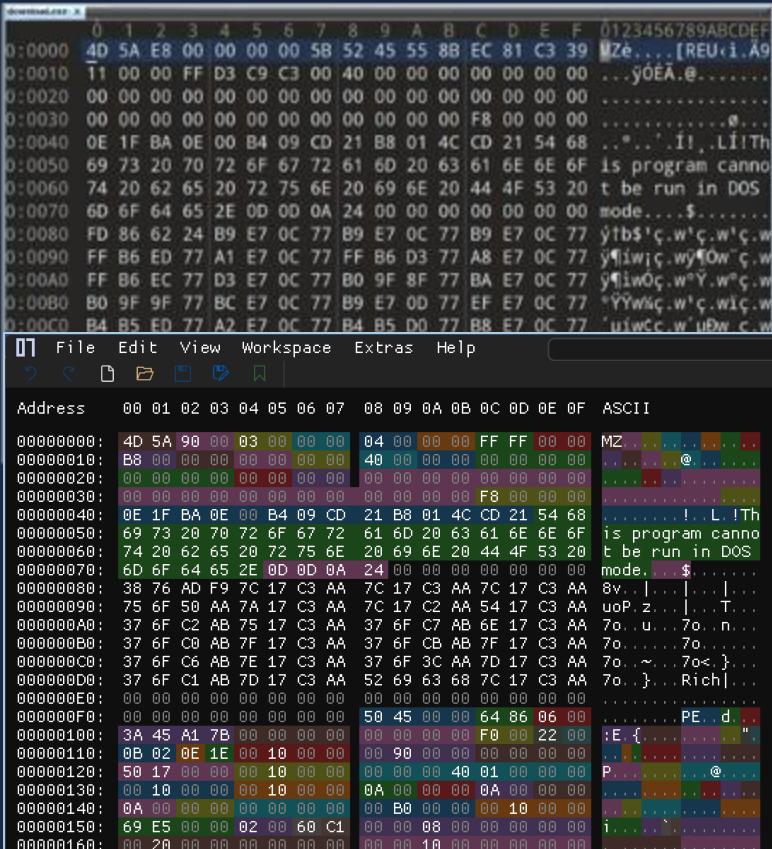
- AFTER you've installed all the necessary software, shutdown and make a checkpoint!
- A checkpoint or restore point allows you to clear your VM to a known clean state.



Other Software

Hex Editor

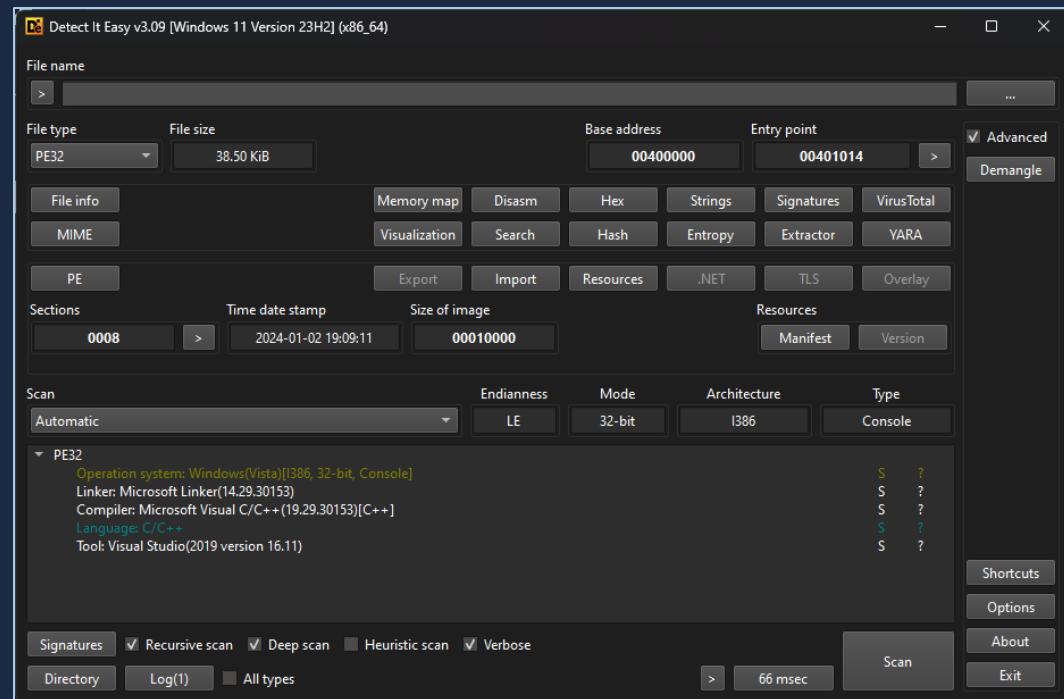
- Personal recommendations
 - 010 Editor (Proprietary)
 - ImHex (FOSS)



Other Software

Detect it Easy

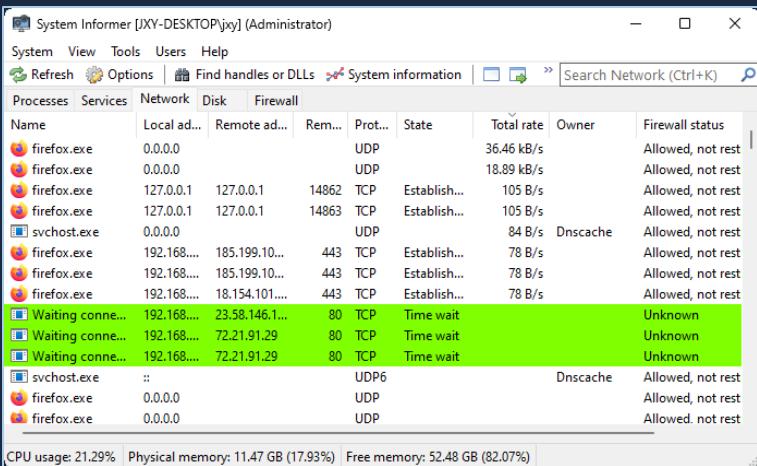
- Detection swiss-knife



Other Software

System Informer

- Previously known as Process Hacker
- Task Manager on steroids



The screenshot shows the System Informer application window titled "System Informer [JXY-DESKTOP\jxy] (Administrator)". The main interface includes a menu bar with System, View, Tools, Users, and Help, and a toolbar with Refresh, Options, Find handles or DLLs, System information, and search fields.

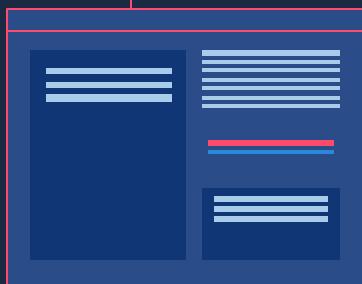
The central part of the window is a table displaying network activity. The columns are: Name, Local ad..., Remote ad..., Rem..., Prot..., State, Total rate, Owner, and Firewall status. The table lists several entries, primarily for Firefox processes (firefox.exe) and svchost.exe. Some entries show established connections to external IP addresses (e.g., 192.168.1.100, 192.168.1.29) and others show waiting connections. The firewall status for most entries is "Allowed, not rest".

Name	Local ad...	Remote ad...	Rem...	Prot...	State	Total rate	Owner	Firewall status
firefox.exe	0.0.0			UDP		36.46 kB/s		Allowed, not rest
firefox.exe	0.0.0			UDP		18.89 kB/s		Allowed, not rest
firefox.exe	127.0.0.1	127.0.0.1	14862	TCP	Establish...	105 B/s		Allowed, not rest
firefox.exe	127.0.0.1	127.0.0.1	14863	TCP	Establish...	105 B/s		Allowed, not rest
svchost.exe	0.0.0			UDP		84 B/s	Dnscache	Allowed, not rest
firefox.exe	192.168...	185.199.10...	443	TCP	Establish...	78 B/s		Allowed, not rest
firefox.exe	192.168...	185.199.10...	443	TCP	Establish...	78 B/s		Allowed, not rest
firefox.exe	192.168...	18.154.101...	443	TCP	Establish...	78 B/s		Allowed, not rest
[Waiting conn...	192.168...	23.58.146.1...	80	TCP	Time wait			Unknown
[Waiting conn...	192.168...	72.21.91.29	80	TCP	Time wait			Unknown
[Waiting conn...	192.168...	72.21.91.29	80	TCP	Time wait			Unknown
svchost.exe	:			UDP6			Dnscache	Allowed, not rest
firefox.exe	0.0.0			UDP				Allowed, not rest
firefox.exe	0.0.0			UDP				Allowed, not rest

CPU usage: 21.29% Physical memory: 11.47 GB (17.93%) Free memory: 52.48 GB (82.07%)



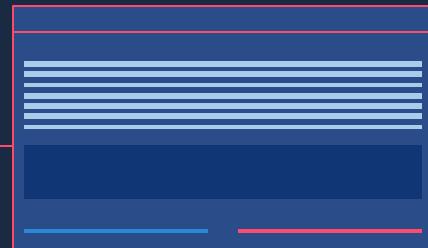
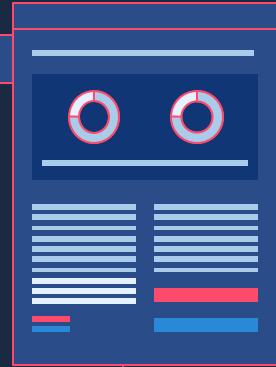
Materials to today's course, including this slide
will be on the GitHub repository
(Still34/malware-deconstructed-lab)





03

Understanding PE Structure



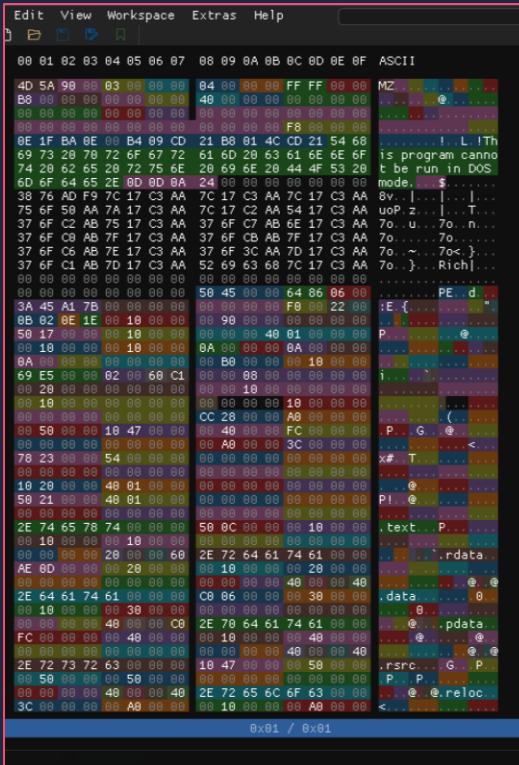
What's This PE Stuff Anyways?

- Enables standardized software loading and execution
- Standard executable format for Windows
 - executable files (EXE)
 - dynamic link libraries (DLL)
 - drivers (SYS)
- Tells Windows
 - what the application is
 - how it should behave
 - where the code is
 - what resources are used, etc.

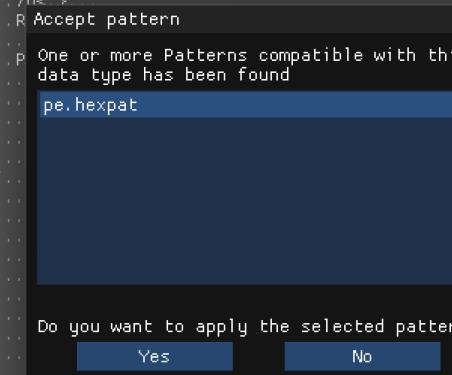


Workshop #1

Open C:\Windows\System32\calc.exe
(or any .EXE files under the directory)
on your machine with a hex editor of your choice



Address	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ASCII
00000000:	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 MZ..
00000010:	B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 ..@..
00000020:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..
00000030:	00 00 00 00 00 00 00 00 00 00 00 F8 00 00 00 ..
00000040:	0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 ..!..L.!Th
00000050:	69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F is program cannot
00000060:	74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 t be run in DOS
00000070:	6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 mode. \$..
00000080:	38 76 AD F9 7C 17 C3 AA 7C 17 C3 AA 7C 17 C3 AA 8v..
00000090:	75 6F 50 AA 7A 17 C3 AA 7C 17 C2 AA 54 17 C3 AA uoP.z.. ..T..
000000A0:	37 6F C2 AB 75 17 C3 AA 37 6F C7 AB 6E 17 C3 AA 7o..u..7o..n..
000000B0:	37 6F C0 AB 7F 17 C3 AA 37 6F CB AB 7F 17 C3 AA 7o..~..7o..`..
000000C0:	37 6F C6 AB 7E 17 C3 AA 37 6F 3C AA 7D 17 C3 AA 7o..~..7o..`..
000000D0:	37 6F C1 AB 7D 17 C3 AA 52 69 63 68 7C 17 C3 AA 7o..}..R Accept pattern
000000E0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..
000000F0:	00 00 00 00 00 00 00 00 50 45 00 00 64 86 06 00 ..P One or more Patterns compatible with this
00000100:	data type has been found
00000110:	3A 45 A1 7B 00 00 00 00 00 00 00 00 F0 00 22 00 :E. {..
00000120:	0B 02 0E 1E 00 10 00 00 00 90 00 00 00 00 00 00 00 ..
00000130:	50 17 00 00 10 00 00 00 00 00 40 01 00 00 00 00 P..
00000140:	00 10 00 00 00 10 00 00 0A 00 00 00 0A 00 00 00 ..
00000150:	0A 00 00 00 00 00 00 00 00 B0 00 00 00 10 00 00 00 ..
00000160:	69 E5 00 00 02 00 60 C1 00 00 08 00 00 00 00 00 00 i..
00000170:	00 20 00 00 00 00 00 00 00 00 10 00 00 00 00 00 ..
00000180:	00 10 00 00 00 00 00 00 00 00 00 00 00 00 10 00 00 ..
00000190:	00 CC 28 00 00 A0 00 00 00 00 00 00 00 00 00 00 ..
000001A0:	00 50 00 00 10 47 00 00 00 40 00 00 FC 00 00 00 P..G..
000001B0:	00 00 00 00 00 00 00 00 00 A0 00 00 3C 00 00 00 ..
000001C0:	78 23 00 00 54 00 00 00 00 00 00 00 00 00 00 00 x#..T..
000001D0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..
000001E0:	10 20 00 00 40 01 00 00 00 00 00 00 00 00 00 00 ..@..
000001F0:	50 21 00 00 40 01 00 00 00 00 00 00 00 00 00 00 P!..@..
00000200:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..
00000210:	2E 74 65 78 74 00 00 00 50 0C 00 00 00 10 00 00 .text..P..
00000220:	00 10 00 00 10 00 00 00 00 00 00 00 00 00 00 00 ..
00000230:	00 00 00 00 20 00 00 60 2E 72 64 61 74 61 00 00 ..rdata..
00000240:	AE 0D 00 00 20 00 00 00 10 00 00 00 20 00 00 ..@..@..
00000250:	00 00 00 00 00 00 00 00 00 00 40 00 00 40 00 00 ..
00000260:	2E 64 61 74 61 00 00 00 C0 06 00 00 00 30 00 00 .data..0..
00000270:	00 00 00 00 30 00 00 00 00 00 00 00 00 00 00 00 ..
00000280:	FC ..
00000290:	00 If you're using ImHex or 010, PE will be automatically identified.
000002A0:	2E ..
000002B0:	00 ..
000002C0:	00 00 00 00 40 00 00 40 2E 72 65 6C 6F 63 00 00 ..@..@..reloc..
000002D0:	3C 00 00 00 00 A0 00 00 00 10 00 00 00 A0 00 00 <..



Address 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ASCII

00000000:	4D 5A 90 00 03
00000010:	B8 00 00 00 00
00000020:	00 00 00 00 00
00000030:	00 00 00 00 00
00000040:	0E 1F BA 0E 00
00000050:	69 73 20 70 72
00000060:	74 20 62 65 20
00000070:	6D 6F 64 65 2E
00000080:	38 76 AD F9 7C
00000090:	75 6F 50 AA 7A
000000A0:	37 6F C2 AB 75
000000B0:	37 6F C0 AB 7F
000000C0:	37 6F C6 AB 7E
000000D0:	37 6F C1 AB 7D
000000E0:	00 00 00 00 00
000000F0:	00 00 00 00 00
00000100:	3A 45 A1 7B 00
00000110:	0B 02 0E 1E 00
00000120:	50 17 00 00 00
00000130:	00 10 00 00 00
00000140:	0A 00 00 00 00
00000150:	69 E5 00 00 02
00000160:	00 20 00 00 00
00000170:	00 10 00 00 00
00000180:	00 00 00 00 00
00000190:	00 50 00 00 10
000001A0:	00 00 00 00 00
000001B0:	78 23 00 00 54
000001C0:	00 00 00 00 00
000001D0:	10 20 00 00 40
000001E0:	50 21 00 00 40
000001F0:	00 00 00 00 00
00000200:	2E 74 65 78 74
00000210:	00 10 00 00 00
00000220:	00 00 00 00 20
00000230:	AE 0D 00 00 00
00000240:	00 00 00 00 00
00000250:	2E 64 61 74 61 00 00 00 C0 06 00 00 00 30 00 00 .data.....0..
00000260:	00 10 00 00 30 00 00 00 00 00 00 00 00 00 00 00,0.....
00000270:	00
00000280:	FC
00000290:	00
000002A0:	2E
000002B0:	00
000002C0:	00 00 00 00 40 00 00 40 2E 72 65 6C 6F 63 00 00@. @. reloc..
000002D0:	3C 00 00 00 00 A0 00 00 00 10 00 00 00 A0 00 00 <.....

[Download PDF](#) [Expand table](#)

Let's try to break this down by hand instead.

File Edit View Workspace Extras Help calc.exe (Read Only) MS-DOS Stub

Address 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ASCII

00000000:	4D 5A	30 00 03 00 00 00	04 00 00 00 FF FF 00 00	MZ
00000010:	00 00 00 00 00 00 00 00	40 00 00 00 00 00 00 00	@	
00000020:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00		
00000030:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	F8 00 00	
00000040:	0E 1F BA 0E 00 B4 09 CD	21 B8 01 4C CD 21 54 68	! L !Th	
00000050:	69 73 20 70 72 6F 67 72	61 6D 20 63 61 6E 6E 6F	is program canno	
00000060:	74 20 62 65 20 72 75 6E	20 69 6E 20 44 4F 53 20	t be run in DOS	
00000070:	6D 6F 64 65 2E 0D 0D 0A	24 00 00 00 00 00 00 00	mode. \$	
00000080:	38 76 AD F9 7C 17 C3 AA	7C 17 C3 AA 7C 17 C3 AA	8v.	
00000090:	75 6F 50 AA 7A 17 C3 AA	7C 17 C2 AA 54 17 C3 AA	uoP. z. . T.	
000000A0:	37 6F C2 AB 75 17 C3 AA	37 6F C7 AB 6E 17 C3 AA	7o. u. 7o. n.	
000000B0:	37 6F C0 AB 7F 17 C3 AA	37 6F CB AB 7F 17 C3 AA	7o. ~. 7o.	
000000C0:	37 6F C6 AB 7E 17 C3 AA	37 6F 3C AA 7D 17 C3 AA	7o. ~. 7o<.)	
000000D0:	37 6F C1 AB 7D 17 C3 AA	52 69 63 68 7C 17 C3 AA	7o. }. Rich	
000000E0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00		
000000F0:	00 00 00 00 00 00 00 00	50 45 00 00 64 86 06 00	PE d.	
00000100:	3A 45 A1 7B 00 00 00 00	00 00 00 00 F0 00 22 00	:E { "	
00000110:	0B 02 0E 1E 00 10 00 00	00 90 00 00 00 00 00 00		
00000120:	50 17 00 00 00 10 00 00	00 00 00 40 01 00 00 00	P. @	
00000130:	00 10 00 00 00 10 00 00	0A 00 00 00 0A 00 00 00		
00000140:	0A 00 00 00 00 00 00 00	00 B0 00 00 10 00 00 00	i.	
00000150:	69 E5 00 00 02 00 60 C1	00 00 08 00 00 00 00 00		
00000160:	00 20 00 00 00 00 00 00	00 00 10 00 00 00 00 00		
00000170:	00 10 00 00 00 00 00 00	00 00 00 10 00 00 00 00		
00000180:	00 00 00 00 00 00 00 00	CC 28 00 00 A0 00 00 00	(
00000190:	00 50 00 00 10 47 00 00	00 40 00 00 FC 00 00 00	P. G. @	
000001A0:	00 00 00 00 00 00 00 00	00 A0 00 00 3C 00 00 00	<	
000001B0:	78 23 00 00 54 00 00 00	00 00 00 00 00 00 00 00	x# T	
000001C0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00		
000001D0:	10 20 00 00 40 01 00 00	00 00 00 00 00 00 00 00	.@	
000001E0:	50 21 00 00 40 01 00 00	00 00 00 00 00 00 00 00	P! @	
000001F0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00		
00000200:	2E 74 65 78 74 00 00 00	50 0C 00 00 00 10 00 00	.text P	
00000210:	00 10 00 00 10 00 00 00	00 00 00 00 00 00 00 00		
00000220:	00 00 00 00 20 00 00 60	2E 72 64 61 74 61 00 00	.rdata	
00000230:	AE 0D 00 00 00 20 00 00	00 10 00 00 00 20 00 00		
00000240:	00 00 00 00 00 00 00 00	00 00 00 00 40 00 00 40	@ @	
00000250:	2E 64 61 74 61 00 00 00	C0 06 00 00 00 30 00 00	.data 0	
00000260:	00 10 00 00 00 30 00 00	00 00 00 00 00 00 00 00		
00000270:	00 00 00 00 40 00 00 C0	2E 70 64 61 74 61 00 00	@ .pdata	
00000280:	FC 00 00 00 40 00 00 00	00 10 00 00 00 40 00 00	@ @	
00000290:	00 00 00 00 00 00 00 00	00 00 00 00 40 00 00 40	@ @	
000002A0:	2E 72 73 72 63 00 00 00	10 47 00 00 50 00 00 00	.rsrc G P	
000002B0:	00 50 00 00 50 00 00 00	00 00 00 00 00 00 00 00	P. P	
000002C0:	00 00 00 00 40 00 00 40	2E 72 65 6C 6F 63 00 00	@ @ .reloc	
000002D0:	3C 00 00 00 A0 00 00 00	00 10 00 00 A0 00 00 00	<	

Page: 0x01 / 0x01 Region: 0x00000000 - 0x0000AFFF (0 - 45055)

File Edit View Workspace Extras Help calc.exe (Read Only) MS-DOS Stub

Address	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ASCII
00000000:	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 MZ, @
00000010:	B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 @
00000020:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 @
00000030:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 @
00000040:	0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C C0 21 54 80 ! L !Th
00000050:	69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6F is program canno
00000060:	74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 t be run in DOS
00000070:	6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 mode, \$
00000080:	38 76 AD F9 7C 17 C3 AA 7C 17 C3 AA 7C 17 C3 AA 8v..
00000090:	75 6F 50 AA 7A 17 C3 AA 7C 17 C2 AA 54 17 C3 AA uoP.z.. ..T.
000000A0:	37 6F C2 AB 75 17 C3 AA 37 6F C7 AB 6E 17 C3 AA 7o..u..7o..n..
000000B0:	37 6F C0 AB 7F 17 C3 AA 37 6F CB AB 7F 17 C3 AA 7o..~..7o..
000000C0:	37 6F C6 AB 7E 17 C3 AA 37 6F 3C AA 7D 17 C3 AA 7o..~..7o..<..}
000000D0:	37 6F C1 AB 7D 17 C3 AA 52 69 63 68 7C 17 C3 AA 7o..}..Rich..
000000E0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000F0:	00 00 00 00 00 00 00 00 50 45 00 00 64 86 06 00 PE d,
00000100:	3A 45 A1 7B 00 00 00 00 00 00 00 00 F0 00 22 00 :E {
00000110:	0B 02 0E 1E 00 10 00 00 00 90 00 00 00 00 00 00 00
00000120:	50 17 00 00 00 10 00 00 00 00 40 01 00 P ..@
00000130:	00 10 00 00 00 10 00 00 0A 00 00 00 0A 00 00 00
00000140:	0A 00 00 00 00 00 00 00 B0 00 00 10 00 00
00000150:	69 E5 00 00 02 00 60 C1 00 00 08 00 00 00 00 00 i ..`
00000160:	00 20 00 00 00 00 00 00 00 00 10 00 00 00 00 00
00000170:	00 10 00 00 00 00 00 00 00 00 00 10 00 00 00 00
00000180:	CC 28 00 00 A0 00 00 00 00 00 00 00 00 00 00 00 ..(
00000190:	00 50 00 00 10 47 00 00 00 40 00 00 FC 00 00 00 P..G..@
000001A0:	00 00 00 00 00 00 00 00 00 A0 00 00 3C 00 00 00 ..<
000001B0:	78 23 00 00 54 00 00 00 00 00 00 00 00 00 00 00 x#..T
000001C0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001D0:	10 20 00 00 40 01 00 00 00 00 00 00 00 00 00 00 ..@..
000001E0:	50 21 00 00 40 01 00 00 00 00 00 00 00 00 00 00 P!..@
000001F0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000200:	2E 74 65 78 74 00 00 00 50 0C 00 00 00 10 00 00 .text P
00000210:	00 10 00 00 10 00 00 00 00 00 00 00 00 00 00 00
00000220:	00 00 00 00 20 00 00 60 2E 72 64 61 74 61 00 00 ..rdata
00000230:	AE 0D 00 00 00 20 00 00 10 00 00 00 20 00 00 00
00000240:	00 00 00 00 00 00 00 00 00 00 40 00 00 40 00 00 ..@..@
00000250:	2E 64 61 74 61 00 00 00 C0 06 00 00 30 00 00 00 .data ..0
00000260:	00 10 00 00 00 30 00 00 00 00 00 00 00 00 00 00
00000270:	00 00 00 00 40 00 00 C0 2E 70 64 61 74 61 00 00 ..@..pdata
00000280:	FC 00 00 00 40 00 00 00 00 10 00 00 40 00 00 00 ..@..@
00000290:	00 00 00 00 00 00 00 00 00 00 40 00 00 40 00 00 ..@..@
000002A0:	2E 72 73 72 63 00 00 00 10 47 00 00 50 00 00 00 .rsrc G P
000002B0:	00 50 00 00 50 00 00 00 00 00 00 00 00 00 00 00 P..P
000002C0:	00 00 00 00 40 00 00 40 2E 72 65 6C 6F 63 00 00 ..@..@.reloc
000002D0:	3C 00 00 00 A0 00 00 00 00 10 00 00 A0 00 00 <..

At location 0x3c, the stub has the file offset to the PE signature. This information enables Windows to properly execute the image file, even though it has an MS-DOS stub. This file offset is placed at location 0x3c during linking.

Region: 0x00000000 - 0x0000AFFF (0 - 45055)

Address	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ASCII
00000000:	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 MZ
00000010:	B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 ..@.
00000020:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..@.
00000030:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..@.
00000040:	0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 ..!L!Th
00000050:	69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F is program canno
00000060:	74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 t be run in DOS
00000070:	6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 mode. \$.
00000080:	38 76 AD F9 7C 17 C3 AA 7C 17 C3 AA 7C 17 C3 AA 8v.. . .
00000090:	75 6F 50 AA 7A 17 C3 AA 7C 17 C2 AA 54 17 C3 AA uoP.z. .T.
000000A0:	37 6F C2 AB 75 17 C3 AA 37 6F C7 AB 6E 17 C3 AA 7o..u..7o..n.
000000B0:	37 6F C0 AB 7F 17 C3 AA 37 6F CB AB 7F 17 C3 AA 7o..~..7o..
000000C0:	37 6F C6 AB 7E 17 C3 AA 37 6F 3C AA 7D 17 C3 AA 7o..~..7o<.)
000000D0:	37 6F C1 AB 7D 17 C3 AA 52 69 63 68 7C 17 C3 AA 7o..}..Rich
000000E0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..
000000F0:	50 45 00 00 64 86 06 00 PE
00000100:	3A 45 A1 7B 00 00 00 00 00 00 F0 00 22 00 :E {".
00000110:	0B 02 0E 1E 00 10 00 00 00 90 00 00 00 00 00 00 ..
00000120:	50 17 00 00 00 10 00 00 00 00 40 01 00 00 00 P..@.
00000130:	00 10 00 00 00 10 00 00 0A 00 00 00 00 00 00 ..
00000140:	0A 00 00 00 00 00 00 00 B0 00 00 10 00 00 ..
00000150:	69 E5 00 00 02 00 60 C1 00 00 08 00 00 00 00 i..`.
00000160:	00 20 00 00 00 00 00 00 00 00 10 00 00 00 00 ..
00000170:	00 10 00 00 00 00 00 00 00 00 00 10 00 00 00 ..
00000180:	00 00 00 00 00 00 00 00 CC 28 00 00 A0 00 00 00 ..(.
00000190:	00 50 00 00 10 47 00 00 00 40 00 00 FC 00 00 00 P..G..@.
000001A0:	00 00 00 00 00 00 00 00 00 A0 00 00 3C 00 00 00 ..<.
000001B0:	78 23 00 00 54 00 00 00 00 00 00 00 00 00 00 x#..T.
000001C0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..
000001D0:	10 20 00 00 40 01 00 00 00 00 00 00 00 00 00 00 ..@.
000001E0:	50 21 00 00 40 01 00 00 00 00 00 00 00 00 00 00 P!..@.
000001F0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..
00000200:	2E 74 65 78 74 00 00 00 50 0C 00 00 00 10 00 00 .text..P.
00000210:	00 10 00 00 10 00 00 00 00 00 00 00 00 00 00 00 ..
00000220:	00 00 00 00 20 00 00 60 2E 72 64 61 74 61 00 00 ..rdata.
00000230:	AE 0D 00 00 00 20 00 00 10 00 00 00 20 00 00 00 ..
00000240:	00 00 00 00 00 00 00 00 00 00 40 00 00 40 00 00 ..@..@.
00000250:	2E 64 61 74 61 00 00 00 C0 06 00 00 30 00 00 00 .data..0.
00000260:	00 10 00 00 00 30 00 00 00 00 00 00 00 00 00 00 ..0.
00000270:	00 00 00 00 40 00 00 C0 2E 70 64 61 74 61 00 00 ..@..pdata.
00000280:	FC 00 00 00 40 00 00 00 00 10 00 00 40 00 00 00 ..@..@.
00000290:	00 00 00 00 00 00 00 00 00 00 40 00 00 40 00 00 ..@..@.
000002A0:	2E 72 73 72 63 00 00 00 10 47 00 00 50 00 00 00 .rsrc..G..P.
000002B0:	00 50 00 00 50 00 00 00 00 00 00 00 00 00 00 00 P..P..
000002C0:	00 00 00 00 40 00 00 40 2E 72 65 6C 6F 63 00 00 ..@..@..reloc.
000002D0:	3C 00 00 00 A0 00 00 00 00 10 00 00 A0 00 00 <.

...in this case, it points to 0xF8.

Signature (Image Only)

After the MS-DOS stub, at the file offset specified at offset 0x3c, is a 4-byte signature that identifies the file as a PE format image file. This signature is "PE\0\0" (the letters "P" and "E" followed by two null bytes).

File Edit View Workspace Extras Help calc.exe (Read Only) ☺

Hex editor

Address 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ASCII

000000A0: 37 6F C2 AB 75 17 C3 AA 37 6F C7 AB E6 17 C3 AA 70..u..70..n..
000000B0: 37 6F C0 AB 7F 17 C3 AA 37 6F CB AB 7F 17 C3 AA 70.....70..
000000C0: 37 6F C6 AB 7E 17 C3 AA 37 6F 3C AA 7D 17 C3 AA 70..~..70<}.
000000D0: 37 6F C1 AB 7D 17 C3 AA 52 69 63 68 7C 17 C3 AA 70..}.Rich|.
000000E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..
000000F0: 00 00 00 00 00 00 00 00 00 00 50 45 00 00 64 86 06 00 PE d.
00000100: 3A 45 A1 7B 00 00 00 00 00 00 F0 00 22 00 VE.
00000110: 0B 02 0E 1E 00 10 00 00 00 90 00 00 00 00 00 00 00 00
00000120: 50 17 00 00 00 10 00 00 00 00 00 40 01 00 00 00 00 P @
00000130: 00 10 00 00 00 10 00 00 0A 00 00 00 0A 00 00 00 00
00000140: 0A 00 00 00 00 00 00 00 00 B0 00 00 00 10 00 00 00
00000150: 69 E5 00 00 02 00 60 C1 00 00 08 00 00 00 00 00 00 i.
00000160: 00 20 00 00 00 00 00 00 00 00 10 00 00 00 00 00 00 00
00000170: 00 10 00 00 00 00 00 00 00 00 00 00 00 00 10 00 00 00
00000180: 00 00 00 00 00 00 00 00 00 CC 28 00 00 A0 00 00 00 00 (.
00000190: 00 50 00 00 10 47 00 00 00 40 00 00 FC 00 00 00 00 P G @
000001A0: 00 00 00 00 00 00 00 00 00 A0 00 00 3C 00 00 00 00 <
000001B0: 78 23 00 00 54 00 00 00 00 00 00 00 00 00 00 00 00 00 x# T
000001C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001D0: 10 20 00 00 40 01 00 00 00 00 00 00 00 00 00 00 00 00 @.
000001E0: 50 21 00 00 40 01 00 00 00 00 00 00 00 00 00 00 00 00 P!
000001F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000200: 2E 74 65 78 74 00 00 00 50 0C 00 00 10 00 00 .text P
00000210: 00 10 00 00 10 00 00 00 00 00 00 00 00 00 00 00 00 00
00000220: 00 00 00 20 00 00 60 2E 72 64 61 74 61 00 00 00 00 00 .rdata
00000230: AE 0D 00 00 20 00 00 00 10 00 00 00 20 00 00 00 00 00
00000240: 00 00 00 00 00 00 00 00 00 00 00 40 00 00 40 00 00 00 .data @ @
00000250: 2E 64 61 74 61 00 00 00 C0 06 00 00 30 00 00 00 00 00
00000260: 00 10 00 00 20 00 00 00 00 00 00 00 00 00 00 00 00 00
Page: 0x01 / 0x01

Pattern Data

Y

Name	Color	Start	End
coffHeader		0x000000F8	0x000000FC
signature		0x000000F8	0x000000FF
architecture		0x000000FC	0x000000FE
numberOfSections		0x000000FE	0x000000FF
timeDateStamp		0x00000100	0x00000103
pointerToSymbolTable		0x00000104	0x00000107
numberOfSymbols		0x00000108	0x0000010B

COFF File Header

COFF File Header

COFF File Header (Object and Image)

At the beginning of an object file, or immediately after the signature of an image file, is a standard COFF file header in the following format. Note that the Windows loader limits the number of sections to 96.

Expand table

Offset	Size	Field	Description
0	2	Machine	The number that identifies the type of target machine. For more information, see Machine Types.
2	2	NumberOfSections	The number of sections. This indicates the size of the section table, which immediately follows the headers.
4	4	TimeDateStamp	The low 32 bits of the number of seconds since 00:00 January 1, 1970 (a C run-time time_t value), which indicates when the file was created.
8	4	PointerToSymbolTable	The file offset of the COFF symbol table, or zero if no COFF symbol table is present. This value should be zero for an image because COFF debugging information is deprecated.
12	4	NumberOfSymbols	The number of entries in the symbol table. This data can be used to locate the string table, which immediately follows the symbol table. This value should be zero for an image because COFF debugging information is deprecated.
16	2	SizeOfOptionalHeader	The size of the optional header, which is required for executable files but not for object files. This value should be zero for an object file. For a description of the header format, see Optional Header (Image Only).
18	2	Characteristics	The flags that indicate the attributes of the file. For specific flag values, see Characteristics.

Type::AMD64

Hex editor

Address	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ASCII
000000A0:	37 6F C2 AB 75 17 C3 AA 37 6F C7 AB E6 17 C3 AA 70 .. u .. 70 .. n ..
000000B0:	37 6F C0 AB 7F 17 C3 AA 37 6F CB AB 7F 17 C3 AA 70 70 ..
000000C0:	37 6F C6 AB 7E 17 C3 AA 37 6F 3C AA 7D 17 C3 AA 70 .. ~ .. 70 < }
000000D0:	37 6F C1 AB 7D 17 C3 AA 52 69 63 68 7C 17 C3 AA 70 .. } .. Rich ..
000000E0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000F0:	00 00 00 00 00 00 00 00 00 00 50 45 00 00 64 86 06 00
00000100:	3A 45 A1 7B 00 00 00 00 00 00 F0 00 22 00
00000110:	0B 02 0E 1E 00 10 00 00 00 90 00 00 00 00 00 00
00000120:	50 17 00 00 00 10 00 00 00 00 00 40 01 00 00 00
00000130:	00 10 00 00 00 10 00 00 0A 00 00 00 0A 00 00 00
00000140:	0A 00 00 00 00 00 00 00 00 B0 00 00 00 10 00 00
00000150:	69 E5 00 00 02 00 60 C1 00 00 08 00 00 00 00 00 00
00000160:	00 20 00 00 00 00 00 00 00 00 10 00 00 00 00 00
00000170:	00 10 00 00 00 00 00 00 00 00 00 00 10 00 00 00
00000180:	00 00 00 00 00 00 00 00 00 CC 28 00 00 A0 00 00 00
00000190:	00 50 00 00 10 47 00 00 00 40 00 00 FC 00 00 00
000001A0:	00 00 00 00 00 00 00 00 00 A0 00 00 3C 00 00 00
000001B0:	78 23 00 00 54 00 00 00 00 00 00 00 00 00 00 00
000001C0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001D0:	10 20 00 00 40 01 00 00 00 00 00 00 00 00 00 00
000001E0:	50 21 00 00 40 01 00 00 00 00 00 00 00 00 00 00
000001F0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000200:	2E 74 65 78 74 00 00 00 50 0C 00 00 10 00 00
00000210:	00 10 00 00 10 00 00 00 00 00 00 00 00 00 00 00
00000220:	00 00 00 20 00 00 60 2E 72 64 61 74 61 00 00
00000230:	AE 0D 00 00 20 00 00 00 10 00 00 00 20 00 00 00
00000240:	00 00 00 00 00 00 00 00 00 00 00 40 00 00 40 00
00000250:	2E 64 61 74 61 00 00 00 C0 06 00 00 30 00 00
00000260:	00 10 00 00 30 00 00 00 00 00 00 00 00 00 00 00

Page:

0x01 / 0x01

Pattern Data

Y

Name	Color	Start	End
coffHeader		0x000000F8	0x0000
signature		0x000000F8	0x0000
architecture		0x000000FC	0x0000
numberOfSections		0x000000FE	0x000000FF
timeDateStamp		0x00000100	0x00000103
pointerToSymbolTable		0x00000104	0x00000107
numberOfSymbols		0x00000108	0x0000010B
		0x000001AC	0x000001AD

COFF File Header

We'll pick a few important ones to talk about...

COFF File Header (Object and Image)

At the beginning of an object file, or immediately after the signature of an image file, is a standard COFF file header in the following format. Note that the Windows loader limits the number of sections to 96.

Expand table

Offset	Size	Field	Description
0	2	Machine	The number that identifies the type of target machine. For more information, see Machine Types .
2	2	NumberOfSections	The number of sections. This indicates the size of the section table, which immediately follows the headers.
4	4	TimeDateStamp	The low 32 bits of the number of seconds since 00:00 January 1, 1970 (a C run-time time_t value), which indicates when the file was created.
8	4	PointerToSymbolTable	The file offset of the COFF symbol table, or zero if no COFF symbol table is present. This value should be zero for an image because COFF debugging information is deprecated.
12	4	NumberOfSymbols	The number of entries in the symbol table. This data can be used to locate the string table, which immediately follows the symbol table. This value should be zero for an image because COFF debugging information is deprecated.
16	2	SizeOfOptionalHeader	The size of the optional header, which is required for executable files but not for object files. This value should be zero for an object file. For a description of the header format, see Optional Header (Image Only) .
18	2	Characteristics	The flags that indicate the attributes of the file. For specific flag values, see Characteristics .

Type	AMD64

Hex editor

Address	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ASCII
000000A0:	37 6F C2 AB 75 17 C3 AA 37 6F C7 AB E6 17 C3 AA 70 .. u .. 70 .. n ..
000000B0:	37 6F C0 AB 7F 17 C3 AA 37 6F CB AB 7F 17 C3 AA 70 70 ..
000000C0:	37 6F C6 AB 7E 17 C3 AA 37 6F 3C AA 7D 17 C3 AA 70 .. ~ .. 70< }
000000D0:	37 6F C1 AB 7D 17 C3 AA 52 69 63 68 7C 17 C3 AA 70 .. } .. Rich ..
000000E0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000F0:	00 00 00 00 00 00 00 00 50 45 00 64 86 06 00 .. PE d ..
00000100:	3A 45 A1 7B 00 00 00 00 00 00 00 00 00 00 00 00
00000110:	0B 02 0E 1E 00 10 00 00 00 90 00 00 00 00 00 00 00
00000120:	50 17 00 00 00 10 00 00 00 00 00 40 01 00 00 00 00
00000130:	00 10 00 00 00 10 00 00 0A 00 00 00 0A 00 00 00 00
00000140:	0A 00 00 00 00 00 00 00 00 B0 00 00 10 00 00 00 00
00000150:	69 E5 00 02 00 60 C1 00 00 08 00 00 00 00 00 00 00
00000160:	00 20 00 00 00 00 00 00 00 00 10 00 00 00 00 00 00
00000170:	00 10 00 00 00 00 00 00 00 00 00 10 00 00 00 00 00
00000180:	00 00 00 00 00 00 00 00 00 CC 28 00 00 A0 00 00 00 00
00000190:	00 50 00 00 10 47 00 00 00 40 00 00 FC 00 00 00 00
000001A0:	00 00 00 00 00 00 00 00 00 A0 00 00 3C 00 00 00 00
000001B0:	78 23 00 00 54 00 00 00 00 00 00 00 00 00 00 00 00
000001C0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001D0:	10 20 00 00 40 01 00 00 00 00 00 00 00 00 00 00 00
000001E0:	50 21 00 00 40 01 00 00 00 00 00 00 00 00 00 00 00
000001F0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000200:	2E 74 65 78 74 00 00 00 50 0C 00 00 00 10 00 00
00000210:	00 10 00 00 10 00 00 00 00 00 00 00 00 00 00 00 00
00000220:	00 00 00 20 00 00 60 2E 72 64 61 74 61 00 00 00
00000230:	AE 0D 00 00 00 20 00 00 10 00 00 00 20 00 00 00 00
00000240:	00 00 00 00 00 00 00 00 00 00 40 00 00 40 00 00 00
00000250:	2E 64 61 74 61 00 00 00 C0 06 00 00 30 00 00 00
00000260:	00 10 00 00 20 00 00 00 00 00 00 00 00 00 00 00

Page: 0x01 / 0x01

Pattern Data

Y

Name	Color	Start	End
coffHeader		0x000000F8	0x00000100
signature		0x000000F8	0x00000000
architecture		0x000000FC	0x00000000
numberOfSections		0x000000FE	0x00000000
timeDateStamp		0x00000100	0x00000001
pointerToSymbolTable		0x00000104	0x00000001
numberOfSymbols		0x00000108	0x00000001
		0x0000010C	0x00000001

COFF File Header

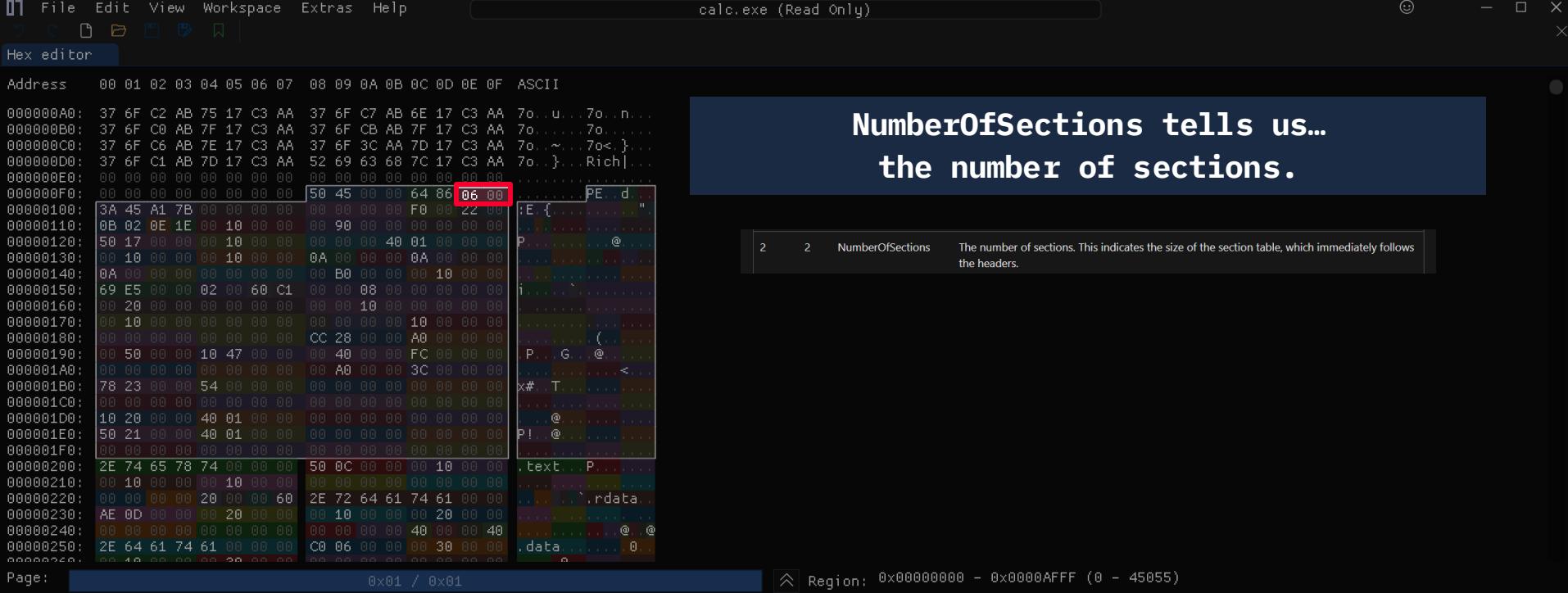
Machine field tells us about the target architecture.

Offset	Size	Field	Description
0	2	Machine	The number that identifies the type of target machine. For more information, see Machine Types.

Machine Types

The Machine field has one of the following values, which specify the CPU type. An image file can be run only on the specified machine or on a system that emulates the specified machine.

Constant	Value	Description
IMAGE_FILE_MACHINE_UNKNOWN	0x0	The content of this field is assumed to be applicable to any machine type
IMAGE_FILE_MACHINE_ALPHA	0x184	Alpha AXP, 32-bit address space
IMAGE_FILE_MACHINE_ALPHA64	0x284	Alpha 64, 64-bit address space
IMAGE_FILE_MACHINE_AM33	0x1d3	Matsushita AM33
IMAGE_FILE_MACHINE_AMD64	0x8664	x64
IMAGE_FILE_MACHINE_ARM	0x1c0	ARM little endian
IMAGE_FILE_MACHINE_ARM64	0xaa64	ARM64 little endian
IMAGE_FILE_MACHINE_ARMNT	0x1c4	ARM Thumb-2 little endian
IMAGE_FILE_MACHINE_AXP64	0x284	AXP 64 (Same as Alpha 64)
IMAGE_FILE_MACHINE_EBC	0xebc	EBC byte code
IMAGE_FILE_MACHINE_I386	0x14c	Intel 386 or later processors and compatible processors
	0x0004	u32
	0x0002	u16
	0 (0x00000000)	
	240 (0x00000000)	



Pattern Data

Y

Name	Color	Start	End	Size	Type	Value
coffHeader		0x000000F8	0x000001FF	0x0108	struct COFFHeader	{ ... }
signature		0x000000F8	0x000000FB	0x0004	String	"PE\x00\x00"
architecture		0x000000FC	0x000000FD	0x0002	enum ArchitectureType	ArchitectureType::AMD64
numberOfSections		0x000000FE	0x000000FF	0x0002	u16	6 (0x0006)
timeDateStamp		0x00000100	0x00000103	0x0004	type::time32_t	Sun Sep 23 13:23:06 2035
pointerToSymbolTable		0x00000104	0x00000107	0x0004	u32	0 (0x00000000)
numberOfSymbols		0x00000108	0x0000010B	0x0004	u32	0 (0x00000000)
		0x0000010C	0x0000010D	0x0002	u16	240 (0x00FA)

COFF File Header

Hex editor

Address	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ASCII
000000A0:	37 6F C2 AB 75 17 C3 AA 37 6F C7 AB E6 17 C3 AA 70 .. u .. 70 .. n ..
000000B0:	37 6F C0 AB 7F 17 C3 AA 37 6F CB AB 7F 17 C3 AA 70 .. . 70 .. .
000000C0:	37 6F C6 AB 7E 17 C3 AA 37 6F 3C AA 7D 17 C3 AA 70 .. ~ .. 70< }
000000D0:	37 6F C1 AB 7D 17 C3 AA 52 69 63 68 7C 17 C3 AA 70 .. } .. Rich ..
000000E0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000F0:	00 00 00 00 00 00 00 00 50 45 00 00 64 86 06 00
00000100:	3A 45 A1 7B 00 00 00 00 00 00 00 F0 00 22 00
00000110:	00 02 0E 1C 00 10 00 00 00 90 00 00 00 00 00 00
00000120:	50 17 00 00 10 00 00 00 00 00 40 01 00 00 00 00
00000130:	00 10 00 00 10 00 00 0A 00 00 00 0A 00 00 00 00
00000140:	0A 00 00 00 00 00 00 B0 00 00 10 00 00 00 00 00
00000150:	69 E5 00 00 02 00 60 C1 00 00 08 00 00 00 00 00 00
00000160:	00 20 00 00 00 00 00 00 00 00 10 00 00 00 00 00
00000170:	00 10 00 00 00 00 00 00 00 00 00 10 00 00 00 00
00000180:	00 00 00 00 00 00 00 00 CC 28 00 00 A0 00 00 00 00
00000190:	00 50 00 00 10 47 00 00 00 40 00 00 FC 00 00 00 00
000001A0:	00 00 00 00 00 00 00 00 00 A0 00 00 3C 00 00 00 00
000001B0:	78 23 00 00 54 00 00 00 00 00 00 00 00 00 00 00
000001C0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001D0:	10 20 00 00 40 01 00 00 00 00 00 00 00 00 00 00
000001E0:	50 21 00 00 40 01 00 00 00 00 00 00 00 00 00 00
000001F0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000200:	2E 74 65 78 74 00 00 00 50 0C 00 00 00 10 00 00
00000210:	00 10 00 00 10 00 00 00 00 00 00 00 00 00 00 00
00000220:	00 00 00 00 20 00 00 60 2E 72 64 61 74 61 00 00
00000230:	AE 0D 00 00 00 20 00 00 10 00 00 00 20 00 00 00
00000240:	00 00 00 00 00 00 00 00 00 00 00 40 00 00 40 00
00000250:	2E 64 61 74 61 00 00 00 C0 06 00 00 00 30 00 00
00000260:	00 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Page: 0x01 / 0x01

TimeDateStamp tells us
when the PE may be compiled*

4 4 TimeDateStamp The low 32 bits of the number of seconds since 00:00 January 1, 1970 (a C run-time time_t value), which indicates when the file was created.

Pattern Data

Name	Color	Start	End	Size	Type	Value
coffHeader		0x000000F8	0x000001FF	0x0108	struct COFFHeader	{ ... }
signature		0x000000F8	0x000000FB	0x0004	String	"PE\x00\x00"
architecture		0x000000FC	0x000000FD	0x0002	enum ArchitectureType	ArchitectureType::AMD64
numberOfSections		0x000000FE	0x000000FF	0x0002	u16	6 (0x0006)
timeDateStamp		0x00000100	0x00000103	0x0004	type::time32_t	Sun Sep 23 13:23:06 2035
pointerToSymbolTable		0x00000104	0x00000107	0x0004	u32	0 (0x00000000)
numberOfSymbols		0x00000108	0x0000010B	0x0004	u32	0 (0x00000000)
		0x0000010C	0x0000010D	0x0002	u16	240 (0xA0AF01)

COFF File Header

Hex editor

Address 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ASCII
000000A0: 37 6F C2 AB 75 17 C3 AA 37 6F C7 AB E6 17 C3 AA To..u...To..n.
000000B0: 37 6F C0 AB 7F 17 C3 AA 37 6F CB AB 7F 17 C3 AA To.....To..
000000C0: 37 6F C6 AB 7E 17 C3 AA 37 6F 3C AA 7D 17 C3 AA To..~..To<}.
000000D0: 37 6F C1 AB 7D 17 C3 AA 52 69 63 68 7C 17 C3 AA To..}.Rich|.
000000E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..
000000F0: 00 00 00 00 00 00 00 00 50 45 00 00 64 86 06 00 PE d
00000100: 3A 45 A1 7B 00 00 00 00 00 00 00 F0 00 22 00 ;E .
00000110: 0B 02 0E 1E 00 10 00 00 00 00 00 00 00 00 00 00 00 9
00000120: 50 17 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00 00
00000130: 00 10 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00 0A
00000140: 0A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0B
00000150: 69 E5 00 00 02 00 60 C1 00 00 00 00 00 00 00 00 00 00
00000160: 00 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000170: 00 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000180: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 CC 2
00000190: 00 50 00 00 10 47 00 00 00 00 00 00 00 00 00 00 00 00 4
000001A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 A
000001B0: 78 23 00 00 54 00 00 00 00 00 00 00 00 00 00 00 00 00
000001C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001D0: 10 20 00 00 40 01 00 00 00 00 00 00 00 00 00 00 00 00
000001E0: 50 21 00 00 40 01 00 00 00 00 00 00 00 00 00 00 00 00
000001F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000200: 2E 74 65 78 74 00 00 00 50 00 00 00 00 00 00 00 00 00
00000210: 00 10 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00 00
00000220: 00 00 00 00 20 00 00 00 60 2E 70 00 00 00 00 00 00 00
00000230: AE 0D 00 00 00 20 00 00 00 00 00 00 00 00 00 00 00 01
00000240: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000250: 2E 64 61 74 61 00 00 00 C0 00 00 00 00 00 00 00 00 00
00000260: 00 10 00 00 00 20 00 00 00 00 00 00 00 00 00 00 00 00
Page:

Pattern Data

Name	Color
coffHeader	
signature	
architecture	
numberOfSections	
timeTimeStamp	
pointerToSymbolTable	
numberOfSymbols	

COFF File Header

*** = could be tampered**

File name: E:\repos\HelloWorldCpp\x64\Release\HelloWorldCpp.exe

File type	File size	Base address	Entry point
PE64	13.50 KiB	0000000140000000	00000001400015c0

File info Memory map Disasm Hex Strings Signatures VirusTotal

MIME Visualization Search Hash Entropy Extractor YARA

PE Export Import Resources .NET TLS Overlay

Sections Time date stamp Size of image Resources

0006 > 2009-06-28 22:33:41 00009000 Manifest Version

Scan Endianness Mode Architecture Type

Automatic LE 64-bit AMD64 Console

PE64

- Operation system: Windows(Vista)|AMD64, 64-bit, Console
- Linker: Microsoft Linker(14.36.32532)
- Compiler: Microsoft Visual C/C++(19.36.32532)[LTCG/C++]
- Language: C/C++
- Tool: Visual Studio(2022 version 17.6)

Signatures Recursive scan Deep scan Heuristic scan Verbose

Directory Log(1) All types Scan 65 msec

Shortcuts Options About Exit

Value

Header { ... } "PE\x00\x00"
SignatureType ArchitectureType::AMD64
6 (0x0006)
Time Sun Sep 23 13:23:06 2035
0 (0x00000000)
0 (0x00000000)
240 (0x00FA)

Hex editor

Address 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ASCII
000000A0: 37 6F C2 AB 75 17 C3 AA 37 6F C7 AB 6E 17 C3 AA 70..u...70..n...
000000B0: 37 6F C0 AB 7F 17 C3 AA 37 6F CB AB 7F 17 C3 AA 70.....70.....
000000C0: 37 6F C6 AB 7E 17 C3 AA 37 6F 3C AA 7D 17 C3 AA 70..~..70< }...
000000D0: 37 6F C1 AB 7D 17 C3 AA 52 69 63 68 7C 17 C3 AA 70..}...Rich|...
000000E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000F0: 00 00 00 00 00 00 00 50 45 00 00 64 86 06 00PE d...
00000100: [3A 45 A1 7B 00 00 00 00 00 00 00 F0 00 22 00]E {.....

* = could be /Brepro artifact

An introduction to deterministic builds with C/C++

Sep 2, 2019 • 22 minutes read

Share on:

What are deterministic builds?

A deterministic build is a process of building the same source code with the same build environment and build instructions producing the same binary in two builds, even if they are made on different machines, build directories and with different names. They are also sometimes called reproducible or hermetic builds if it is guaranteed to produce the same binaries even compiling from different folders.

Deterministic builds are not something that happens naturally. Normal projects do not produce deterministic builds and the reasons that they are not produced can be different for each operating system and compiler.

Possible solutions for Microsoft Visual Studio

Microsoft Visual Studio has a linker flag `/Brepro` that is undocumented by Microsoft. That flag sets the timestamps from the `Portable Executable` format to a `-1` value as can be seen in the image below.

; 0x00000000 - 0x0000AFFF (0 - 45055)

Size	Type	Value
0x0108	struct COFFHeader	{ ... }
0x0004	String	"PE\x00\x00"
0x0002	enum ArchitectureType	ArchitectureType::AMD64
0x0002	u16	6 (0x0006)
0x0004	type::time32_t	Sun Sep 23 13:23:06 2035
0x0004	u32	0 (0x00000000)
0x0004	u32	0 (0x00000000)
0x0002	u16	240 (0xA0FA01)

File Edit View Workspace Extras Help calc.exe (Read Only) ☺

Hex editor

Address 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ASCII

000000A0: 37 6F C2 AB 75 17 C3 AA 37 6F C7 AB E6 17 C3 AA 70..u...70..n...

000000B0: 37 6F C0 AB 7F 17 C3 AA 37 6F CB AB 7F 17 C3 AA 70.....70.....

000000C0: 37 6F C6 AB 7E 17 C3 AA 37 6F 3C AA 7D 17 C3 AA 70..~..70<{...}

000000D0: 37 6F C1 AB 7D 17 C3 AA 52 69 63 68 7C 17 C3 AA 70..}...Rich|...

000000E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

000000F0: 00 00 00 00 00 00 00 50 45 00 00 64 86 06 00PE.d...

00000100: [3A 45 A1 7B 00 00 00 00 00 00 00 F0 00 22 00]IE {....}

* = could be /Brepro artifact

An introdu...

Sep 2, 2019 · 22 m...

Share on:

What are deterministi...

A deterministic build is one where the output produced by a compiler and linker is the same given the same source code, build configuration, and dependencies. This is achieved by using a fixed set of tools and libraries, and by ensuring that the build environment and build steps are consistent across all machines. Deterministic builds are often used in environments where reproducibility is important, such as continuous integration systems or in the production of binaries even comp...

Deterministic build...

produce deterministic builds and the reasons that they are not produced can be different for each operating system and compiler.

Use the value of the field as a reference, not a fact!

Name	Size	Type	Value
signature	0x0108	struct COFFHeader	{ ... }
architecture	0x0004	String	"PE\x00\x00"
numberOfSections	0x0002	enum ArchitectureType	ArchitectureType::AMD64
timeDateStamp	0x0002	u16	6 (0x0006)
pointerToSymbolTable	0x0004	type::time32_t	Sun Sep 23 13:23:06 2035
numberOfSymbols	0x0004	u32	0 (0x00000000)
	0x0004	u32	0 (0x00000000)
	0x0002	u16	240 (0x00FA)

COFF File Header

Hex editor

Address	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ASCII
000000A0:	37 6F C2 AB 75 17 C3 AA 37 6F C7 AB E6 17 C3 AA 70 .. u .. 70 .. n ..
000000B0:	37 6F C0 AB 7F 17 C3 AA 37 6F CB AB 7F 17 C3 AA 70 .. . 70 .. .
000000C0:	37 6F C6 AB 7E 17 C3 AA 37 6F 3C AA 7D 17 C3 AA 70 .. ~ .. 70 < }
000000D0:	37 6F C1 AB 7D 17 C3 AA 52 69 63 68 7C 17 C3 AA 70 .. } .. Rich
000000E0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000F0:	00 00 00 00 00 00 00 00 50 45 00 00 64 86 06 00
00000100:	3A 45 A1 7B 00 00 00 00 00 00 00 F0 00 22 00
00000110:	00 02 0E 1E 00 10 00 00 00 90 00 00 00 00 00 00
00000120:	50 17 00 00 00 10 00 00 00 00 00 40 01 00 00 00
00000130:	00 10 00 00 00 10 00 00 0A 00 00 00 0A 00 00 00
00000140:	0A 00 00 00 00 00 00 00 B0 00 00 00 10 00 00 00
00000150:	69 E5 00 02 00 60 C1 00 00 08 00 00 00 00 00 00
00000160:	00 20 00 00 00 00 00 00 00 00 10 00 00 00 00 00
00000170:	00 10 00 00 00 00 00 00 00 00 00 10 00 00 00 00
00000180:	00 00 00 00 00 00 00 00 CC 28 00 00 A0 00 00 00 00
00000190:	00 50 00 00 10 47 00 00 00 40 00 00 FC 00 00 00 00
000001A0:	00 00 00 00 00 00 00 00 00 A0 00 00 3C 00 00 00 00
000001B0:	78 23 00 00 54 00 00 00 00 00 00 00 00 00 00 00
000001C0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001D0:	10 20 00 00 40 01 00 00 00 00 00 00 00 00 00 00
000001E0:	50 21 00 00 40 01 00 00 00 00 00 00 00 00 00 00
000001F0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000200:	2E 74 65 78 74 00 00 00 50 0C 00 00 00 10 00 00
00000210:	00 10 00 00 10 00 00 00 00 00 00 00 00 00 00 00
00000220:	00 00 00 20 00 00 60 2E 72 64 61 74 61 00 00 00
00000230:	AE 0D 00 00 20 00 00 00 10 00 00 00 20 00 00 00
00000240:	00 00 00 00 00 00 00 00 00 00 40 00 00 40 00 00
00000250:	2E 64 61 74 61 00 00 00 C0 06 00 00 00 30 00 00
00000260:	00 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Page:

0x01 / 0x01

Regi

Pattern Data



Name	Color	Start	End	Value
coffHeader		0x000000F8	0x000001FF	{ ... }
signature		0x000000F8	0x000000FB	"PE\x00\x00"
architecture		0x000000FC	0x000000FD	ArchitectureType::AMD64
numberOfSections		0x000000FE	0x000000FF	6 (0x0006)
timeDateStamp		0x00000100	0x00000103	Sun Sep 23 13:23:06 2035
pointerToSymbolTable		0x00000104	0x00000107	0 (0x00000000)
numberOfSymbols		0x00000108	0x0000010B	0 (0x00000000)
		Ax0000001AC	Ax0000001AD	240 (AyAAFA1)

COFF File Header

Characteristics

18

2

Characteristics

The flags that indicate the attributes of the file. For specific flag values, see Characteristics.

Characteristics

The Characteristics field contains flags that indicate attributes of the object or image file. The following flags are currently defined:

Expand table

Flag	Value	Description
IMAGE_FILE_RELocs_STRIPped	0x0001	Image only, Windows CE, and Microsoft Windows NT and later. This indicates that the file does not contain base relocations and must therefore be loaded at its preferred base address. If the base address is not available, the loader reports an error. The default behavior of the linker is to strip base relocations from executable (EXE) files.
IMAGE_FILE_EXECutable_IMAGE	0x0002	Image only. This indicates that the file is valid and can be run. If this flag is not set, it indicates a linker error.
IMAGE_FILE_LINE_NUMS_STRIPped	0x0004	COFF line numbers have been removed. This flag is deprecated and should be zero.
IMAGE_FILE_LOCAL_SYMs_STRIPped	0x0008	COFF symbol table entries for local symbols have been removed. This flag is deprecated and should be zero.
IMAGE_FILE.Aggressive_WS_TRIM	0x0010	Obsolete: Aggressively trim working set. This flag is deprecated for Windows 2000 and later and must be zero.
IMAGE_FILE_LARGE_ADDRESS_AWARE	0x0020	Application can handle > 2-GB addresses.
	0x0040	This flag is reserved for future use.
IMAGE_FILE_BYTES_REVERSED_LO	0x0080	Little endian: the least significant bit (LSB) precedes the most significant bit (MSB) in memory. This flag is deprecated and should be zero.
IMAGE_FILE_32BIT_MACHINE	0x0100	Machine is based on a 32-bit-word architecture.
IMAGE_FILE_DEBUG_STRIPped	0x2000	Debugging information is removed from the image file.
IMAGE_FILE_Removable_RUN_FROM_SWAP	0x4000	If the image is on removable media, fully load it and copy it to the swap file.
IMAGE_FILE_NET_RUN_FROM_SWAP	0x8000	If the image is on network media, fully load it and copy it to the swap file.
IMAGE_FILE_SYSTEM	0x1000	The image file is a system file, not a user program.
IMAGE_FILE_DLL	0x2000	The image file is a dynamic-link library (DLL). Such files are considered executable files for almost all purposes, although they cannot be directly run.
IMAGE_FILE_UP_SYSTEM_ONLY	0x4000	The file should be run only on a uniprocessor machine.
IMAGE_FILE_BYTES_REVERSED_HI	0x8000	Big endian: the MSB precedes the LSB in memory. This flag is deprecated and should be zero.

Hex editor

Address	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ASCII
000000C0:	37 6F C6 AB 7E 17 C3 AA 37 6F 3C AA 7D 17 C3 AA 7o..~...7o<.)...
000000D0:	37 6F C1 AB 7D 17 C3 AA 52 69 63 68 7C 17 C3 AA 7o..}...Rich ...
000000E0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000F0:	00 00 00 00 00 00 00 00 50 45 00 00 64 86 06 00PE.d...
00000100:	3A 45 A1 7B 00 00 00 00 00 00 00 F0 22 22 00 :E { "
00000110:	0B 02 0E 1E 00 10 00 00 00 90 00 00 00 00 00 00 00
00000120:	50 17 00 00 00 10 00 00 00 00 40 01 00 00 00 00 P...@...
00000130:	00 10 00 00 00 10 00 00 0A 00 00 00 0A 00 00 00
00000140:	0A 00 00 00 00 00 00 00 0B 00 00 00 10 00 00
00000150:	69 E5 00 00 02 00 60 C1 00 00 08 00 00 00 00 00 i...`...
00000160:	00 20 00 00 00 00 00 00 00 10 00 00 00 00 00 00
00000170:	00 10 00 00 00 00 00 00 00 00 10 00 00 00 00 00
00000180:	CC 28 00 00 A0 00 00 00 00 00 00 00 00 00 00 00
00000190:	00 50 00 00 10 47 00 00 00 40 00 00 FC 00 00 00 P...G...@...
000001A0:	00 00 00 00 00 00 00 00 00 A0 00 00 3C 00 00 00
000001B0:	78 23 00 00 54 00 00 00 00 00 00 00 00 00 00 00 x#..T...
000001C0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001D0:	10 20 00 00 40 01 00 00 00 00 00 00 00 00 00 00
000001E0:	50 21 00 00 40 01 00 00 00 00 00 00 00 00 00 00 P!...@...
000001F0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000200:	2E 74 65 78 74 74 00 00 50 OC 00 00 10 00 00 .text.P...
00000210:	00 10 00 00 00 10 00 00 00 00 00 00 00 00 00 00
00000220:	00 00 00 00 20 00 00 60 2E 72 64 61 74 61 00 00,rdata...
00000230:	AE BD 00 00 00 20 00 00 00 10 00 00 20 00 00
00000240:	00 00 00 00 00 00 00 00 00 00 40 00 00 40 00 00,@...@...
00000250:	2E 64 61 74 61 00 00 00 C0 06 00 00 30 00 00 .data...0...
00000260:	00 10 00 00 00 30 00 00 00 00 00 00 00 00 00 00
00000270:	00 00 00 00 40 00 00 C0 2E 70 64 61 74 61 00 00,@..,pdata...
00000280:	EC 00 00 00 40 00 00 00 00 00 00 00 00 00 00 00

Page:

0x01 / 0x01

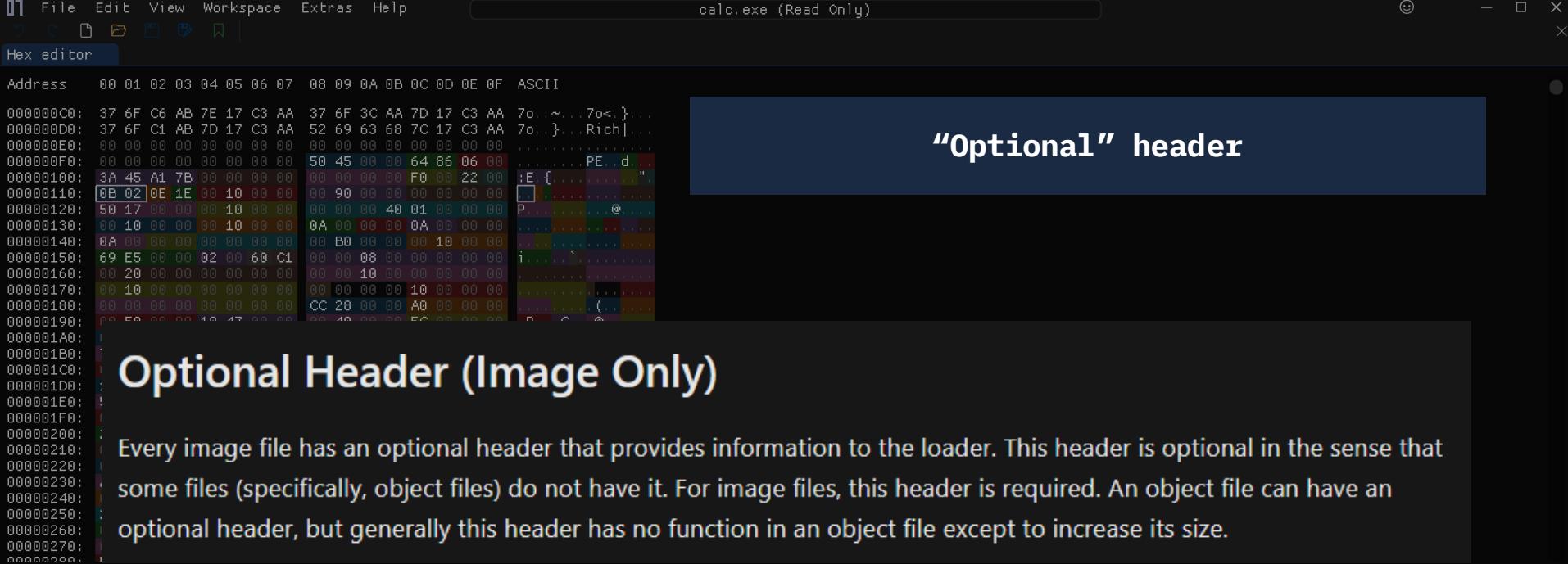
Region: 0x00000000 - 0x0000AFFF (0 - 45055)

Pattern Data

Name	Color	Start	End	Size	Type	Value
sizeOfOptionalHeader	Yellow	0x00000010C	0x00000010D	0x0002	u16	240 (0x00F0)
► characteristics		0x00000010E	0x00000010F	0x0002	bitfield Characteristics { executableImage large...	
▼ optionalHeader		0x000000110	0x0000001FF	0x00F0	struct OptionalHeader { ... }	
magic	Blue	0x000000110	0x000000111	0x0002	enum PEFormat	PEFormat::PE32Plus (0x02)
majorLinkerVersion	Orange	0x000000112	0x000000112	0x0001	u8	14 (0x0E)
minorLinkerVersion	Green	0x000000113	0x000000113	0x0001	u8	30 (0x1E)
sizeOfCode	Red	0x000000114	0x000000117	0x0004	u32	4096 (0x0001000)

Optional Header

"Optional" header



Optional Header (Image Only)

Every image file has an optional header that provides information to the loader. This header is optional in the sense that some files (specifically, object files) do not have it. For image files, this header is required. An object file can have an optional header, but generally this header has no function in an object file except to increase its size.

Name	Color	Start	End	Size	Type	Value
sizeOfOptionalHeader	Yellow	0x0000010C	0x0000010D	0x0002	u16	240 (0x00F0)
► characteristics		0x0000010E	0x0000010F	0x0002	bitfield Characteristics { executableImage largeCode }	
▼ optionalHeader		0x00000110	0x000001FF	0x00F0	struct OptionalHeader { ... }	
magic	Blue	0x00000110	0x00000111	0x0002	enum PEFormat	PEFormat::PE32Plus (0x02)
majorLinkerVersion	Orange	0x00000112	0x00000112	0x0001	u8	14 (0x0E)
minorLinkerVersion	Green	0x00000113	0x00000113	0x0001	u8	30 (0x1E)
sizeOfCode	Red	0x00000114	0x00000117	0x0004	u32	4096 (0x0001000)

Optional Header

File Edit View Workspace Extras Help

Hex editor

Address 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F /

000000C0: 37 6F C6 AB 7E 17 C3 AA 37 6F 3C AA 7D 17 C3 AA 7
000000D0: 37 6F C1 AB 7D 17 C3 AA 52 69 63 68 7C 17 C3 AA 7
000000E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000F0: 00 00 00 00 00 00 00 00 50 45 00 00 64 86 06 00
00000100: 3A 45 A1 7B 00 00 00 00 00 00 F0 00 22 00 00 00
00000110: 0B 02 0E 1E 00 10 00 00 00 90 00 00 00 00 00 00 00
00000120: 50 17 00 00 00 10 00 00 00 00 00 40 01 00 00 00
00000130: 0A 10 00 00 00 10 00 00 0A 00 00 00 0A 00 00 00
00000140: 0A 00 00 00 00 00 00 00 00 B0 00 00 00 10 00 00 00
00000150: 69 E5 00 00 02 00 60 C1 00 00 08 00 00 00 00 00 00
00000160: 00 20 00 00 00 00 00 00 00 00 10 00 00 00 00 00
00000170: 00 10 00 00 00 00 00 00 00 00 00 10 00 00 00 00
00000180: 00 00 00 00 00 00 00 00 CC 28 00 00 A0 00 00 00 00
00000190: 00 50 00 00 40 47 00 00 00 00 00 00 00 00 00 00
000001A0:
000001B0:
000001C0:
000001D0:
000001E0:
000001F0:
00000200:
00000210:
00000220:
00000230:
00000240:
00000250:
00000260:
00000270:
00000280:
Page: 0x01 / 0x01

Pattern Data

Name	Color	Start
sizeOfOptionalHeader	Yellow	0x0000
characteristics	Blue	0x0000
optionalHeader	Green	0x0000
magic	Orange	0x0000
majorLinkerVersion	Red	0x0000
minorLinkerVersion	Green	0x0000
sizeOfCode	Red	0x0000

Optional Header

Optional Header Windows-Specific Fields (Image Only)

The next 21 fields are an extension to the COFF optional header format. They contain additional information that is required by the linker and loader in Windows.

Offset Size Field Description

28/24 4/8 ImageBase The preferred address of the first byte of image when loaded into memory must be a multiple of 64 K. The default for DLLs is 0x1000000. The default for Windows CE EXEs is 0x00010000. The default for Windows NT, Windows 2000, Windows XP, Windows 95, Windows 98, and Windows Me is 0x0400000.

32/32 4 SectionAlignment The alignment (in bytes) of sections when they are loaded into memory. It must be greater than or equal to FileAlignment. The default is the page size for the architecture.

36/36 4 FileAlignment The alignment factor (in bytes) that is used to align the raw data of sections in the image file. The value should be a power of 2 between 512 and 64 K, inclusive. The default is 512. If the SectionAlignment is less than the architecture's page size, then FileAlignment must match SectionAlignment.

40/40 2 MajorOperatingSystemVersion The major version number of the required operating system.

42/42 2 MinorOperatingSystemVersion The minor version number of the required operating system.

44/44 2 MajorImageVersion The major version number of the image.

46/46 2 MinorImageVersion The minor version number of the image.

48/48 2 MajorSubsystemVersion The major version number of the subsystem.

50/50 2 MinorSubsystemVersion The minor version number of the subsystem.

52/52 4 Win32VersionValue Reserved, must be zero.

56/56 4 SizeOfImage The size (in bytes) of the image, including all headers, as the image is loaded in memory. It must be a multiple of SectionAlignment.

60/60 4 SizeOfHeaders The combined size of an MS-DOS stub, PE header, and section headers rounded up to a multiple of FileAlignment.

64/64 4 CheckSum The image file checksum. The algorithm for computing the checksum is incorporated into IMAGHELP.DLL. The following are checked for validation at load time: all drivers; any DLL loaded at boot time, and any DLL that is loaded into a critical Windows process.

68/68 2 Subsystem The subsystem that is required to run this image. For more information, see Windows Subsystem.

70/70 2 DLLCharacteristics For more information, see DLL Characteristics later in this specification.

72/72 4/8 SizeOfStackReserve The size of the stack to reserve. Only SizeOfStackCommit is committed; the rest is made available one page at a time until the reserve size is reached.

76/80 4/8 SizeOfStackCommit The size of the stack to commit.

80/88 4/8 SizeOfHeapReserve The size of the local heap space to reserve. Only SizeOfHeapCommit is committed; the rest is made available one page at a time until the reserve size is reached.

84/96 4/8 SizeOfHeapCommit The size of the local heap space to commit.

88/104 4 LoaderFlags Reserved, must be zero.

92/108 4 NumberOfRvaAndSizes The number of data-directory entries in the remainder of the optional header. Each describes a location and size.

“al” header

header is optional in the sense that it is not required. An object file can have an optional header, but generally this header is not present. This header is used to increase its size.

FFF (0 - 45055)

Type	Value
u16	240 (0x00F0)
bitfield Characteristics { executableImage largeCode }	{ ... }
struct OptionalHeader { ... }	
enum PEFormat	PEFormat::PE32Plus (0x02)
u8	14 (0x0E)
u8	30 (0x1E)
u32	4096 (0x00001000)

Hex editor

Address 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ASCII

000000C0:	37 6F C6 AB 7E 17 C3 AA	37 6F 3C AA 7D 17 C3 AA	70 .. ~... 70< }.
000000D0:	37 6F C1 AB 7D 17 C3 AA	52 69 63 68 7C 17 C3 AA	70 .. }.. Rich!
000000E0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000000F0:	00 00 00 00 00 00 00 00	50 45 00 00 64 86 06 00 PE d.
00000100:	38 45 A1 7B 00 00 00 00	00 00 00 F0 22 00	:E { "
00000110:	0B 02 0E 1E 00 10 00 00	00 90 00 00 00 00 00 00 P. @
00000120:	50 17 00 00 00 10 00 00	00 00 00 40 01 00 00 00
00000130:	0A 00 00 00 00 10 00 00	0A 00 00 00 0A 00 00 00
00000140:	0A 00 00 00 00 00 00 00	00 B0 00 00 10 00 00 00 i.
00000150:	69 E5 00 00 02 00 50 C1	00 00 08 00 00 00 00 00
00000160:	00 20 00 00 00 00 00 00	00 00 10 00 00 00 00 00
00000170:	00 10 00 00 00 00 00 00	00 00 00 00 10 00 00 00
00000180:	00 00 00 00 00 00 00 00	CC 28 00 00 A0 00 00 00 (
00000190:	00 50 00 00 10 47 00 00	00 40 00 FC 00 00 00 00 P. G. @
000001A0:	00 00 00 00 00 00 00 00	00 A0 00 3C 00 00 00 00 <
000001B0:	78 23 00 00 54 00 00 00	00 00 00 00 00 00 00 00	x# T
000001C0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000001D0:	10 20 00 00 40 01 00 00	00 00 00 00 00 00 00 00 @
000001E0:	50 21 00 00 40 01 00 00	00 00 00 00 00 00 00 00 P! @
000001F0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000200:	2E 74 65 78 74 00 00 00	50 0C 00 00 10 00 00 00	.text P.
00000210:	00 10 00 00 00 10 00 00	00 00 00 00 00 00 00 00
00000220:	00 00 00 00 20 00 00 60	2E 72 64 61 74 61 00 00rdata
00000230:	AE 0D 00 00 00 20 00 00	00 10 00 00 00 20 00 00
00000240:	00 00 00 00 00 00 00 00	00 00 00 00 40 00 00 40 @. @
00000250:	2E 64 61 74 61 00 00 00	C0 06 00 00 30 00 00 00	.data 0
00000260:	00 10 00 00 00 30 00 00	00 00 00 00 00 00 00 00
00000270:	00 00 00 00 40 00 00 C0	2E 70 64 61 74 61 00 00 @. pdata.
00000280:	EC 00 00 00 40 00 00 00	00 00 00 00 00 00 00 00

Page:

0x01 / 0x01

Pattern Data

Y

Name	Color	Start	End
sizeOfOptionalHeader	Yellow	0x0000010C	0x0000010E
► characteristics		0x0000010E	0x0000010F
▼ optionalHeader		0x00000110	0x00000110
magic	Blue	0x00000110	0x00000110
majorLinkerVersion	Orange	0x00000112	0x00000112
minorLinkerVersion	Green	0x00000113	0x00000113
sizeOfCode	Red	0x00000114	0x00000114

Optional Header

Target subsystem

68/68	2	Subsystem	The subsystem that is required to run this image. For more information, see Windows Subsystem .
-------	---	-----------	---

Windows Subsystem

The following values defined for the Subsystem field of the optional header determine which Windows subsystem (if any) is required to run the image.

[Expand table](#)

Constant	Value	Description
IMAGE_SUBSYSTEM_UNKNOWN	0	An unknown subsystem
IMAGE_SUBSYSTEM_NATIVE	1	Device drivers and native Windows processes
IMAGE_SUBSYSTEM_WINDOWS_GUI	2	The Windows graphical user interface (GUI) subsystem
IMAGE_SUBSYSTEM_WINDOWS_CUI	3	The Windows character subsystem
IMAGE_SUBSYSTEM_OS2_CUI	5	The OS/2 character subsystem
IMAGE_SUBSYSTEM_POSIX_CUI	7	The Posix character subsystem
IMAGE_SUBSYSTEM_NATIVE_WINDOWS	8	Native Win9x driver
IMAGE_SUBSYSTEM_WINDOWS_CE_GUI	9	Windows CE
IMAGE_SUBSYSTEM_EFI_APPLICATION	10	An Extensible Firmware Interface (EFI) application
IMAGE_SUBSYSTEM_EFI_BOOT_SERVICE_DRIVER	11	An EFI driver with boot services
IMAGE_SUBSYSTEM_EFI_RUNTIME_DRIVER	12	An EFI driver with run-time services
IMAGE_SUBSYSTEM_EFI_ROM	13	An EFI ROM image
IMAGE_SUBSYSTEM_XBOX	14	XBOX
IMAGE_SUBSYSTEM_WINDOWS_BOOT_APPLICATION	16	Windows boot application.

...)	eImage lang	
E32Plus (0x021			
01000)			

Hex editor

Address 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ASCII

000000C0:	37 6F C6 AB 7E 17 C3 AA	37 6F 3C AA 7D 17 C3 AA	70 .. ~ .. 70 < ..
000000D0:	37 6F C1 AB 7D 17 C3 AA	52 69 63 68 7C 17 C3 AA	70 .. } .. Rich ..
000000E0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
000000F0:	00 00 00 00 00 00 00 00	50 45 00 00 64 86 06 00 PE d ..
00000100:	38 45 A1 7B 00 00 00 00	00 00 00 F0 00 22 00 { E .. "
00000110:	0B 02 0E 1E 00 10 00 00	00 90 00 00 00 00 00 00 P .. @ ..
00000120:	50 17 00 00 00 10 00 00	00 00 00 40 01 00 00 00 P .. @ ..
00000130:	00 10 00 00 00 10 00 00	0A 00 00 00 OA 00 00 00 B0 .. 00 10 00 00
00000140:	0A 00 00 00 00 00 00 00	00 B0 00 00 00 00 00 00 i .. , ..
00000150:	69 E5 00 00 02 00	00 00 08 00 00 00 00 00 i .. , ..
00000160:	60 C1	00 00 10 00 00 00 00 00 i .. , ..
00000170:	00 20 00 00 00 00 00 00	00 00 10 00 00 00 00 00 i .. , ..
00000180:	00 00 00 00 00 00 00 00	CC 28 00 00 A0 00 00 00 (.. P .. G .. @ ..
00000190:	00 50 00 00 10 47 00 00	00 40 00 FC 00 00 00 00 P .. G .. @ ..
000001A0:	00 00 00 00 00 00 00 00	00 A0 00 3C 00 00 00 00 < ..
000001B0:	78 23 00 00 54 00 00 00	00 00 00 00 00 00 00 00 x# .. T ..
000001C0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 @ ..
000001D0:	10 20 00 00 40 01 00 00	00 00 00 00 00 00 00 00 @ ..
000001E0:	50 21 00 00 40 01 00 00	00 00 00 00 00 00 00 00 P! .. @ ..
000001F0:	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00text .. P ..
00000200:	2E 74 65 78 74 00 00 00	50 0C 00 00 00 10 00 00text .. P ..
00000210:	00 10 00 00 00 10 00 00	00 00 00 00 00 00 00 00rdata ..
00000220:	00 00 00 00 20 00 00 60	2E 72 64 61 74 61 00 00rdata ..
00000230:	AE 0D 00 00 00 20 00 00	00 10 00 00 00 20 00 00rdata ..
00000240:	00 00 00 00 00 00 00 00	00 00 00 00 40 00 00 40 @ .. @ ..
00000250:	2E 64 61 74 61 00 00 00	C0 06 00 00 30 00 00 00data .. 0 ..
00000260:	00 10 00 00 00 30 00 00	00 00 00 00 00 00 00 00 0 ..
00000270:	00 00 00 00 40 00 00 00	2E 70 64 61 74 61 00 00pdata ..
00000280:	EC 00 00 00 40 00 00 00	00 00 00 00 00 00 00 00pdata ..

Page:

0x01 / 0x01

Pattern Data

Y

Name	Color	Start	End
sizeOfOptionalHeader	Yellow	0x0000010C	0x0000010D
► characteristics	Yellow	0x0000010E	0x0000010F
▼ optionalHeader	Yellow	0x00000110	0x000001FF
magic	Blue	0x00000110	0x00000111
majorLinkerVersion	Orange	0x00000112	0x00000112
minorLinkerVersion	Green	0x00000113	0x00000113
sizeOfCode	Red	0x00000114	0x00000117

Optional Header

DLL characteristics

70/70

2

DLLCharacteristics

For more information, see DLL Characteristics later in this specification.

DLL Characteristics

The following values are defined for the DLLCharacteristics field of the optional header.

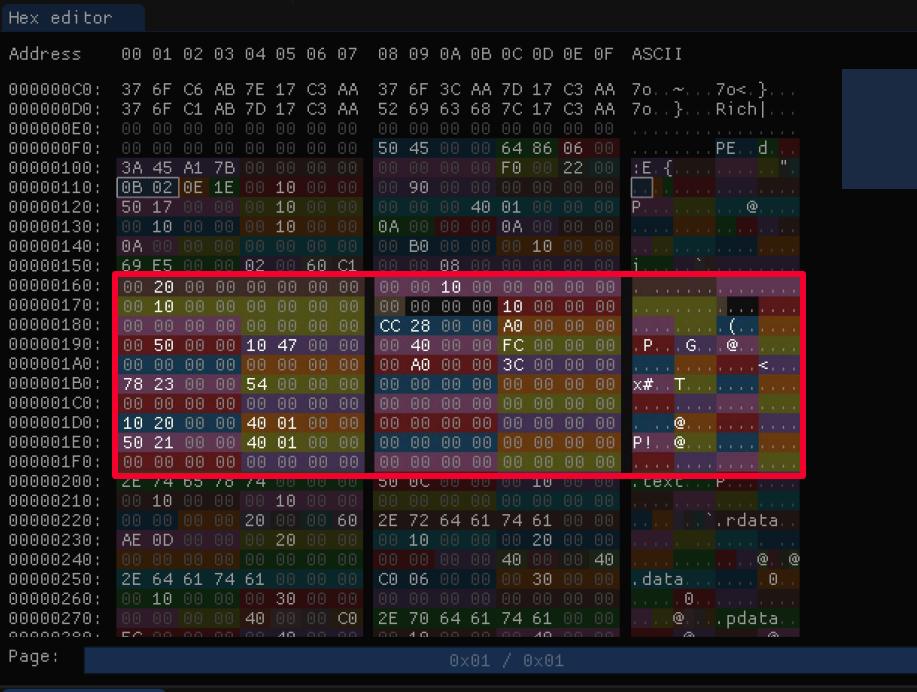
Expand table

Constant	Value	Description
IMAGE_DLLCHARACTERISTICS_HIGH_ENTROPY_VA	0x0020	Image can handle a high entropy 64-bit virtual address space.
IMAGE_DLLCHARACTERISTICS_DYNAMIC_BASE	0x0040	DLL can be relocated at load time.
IMAGE_DLLCHARACTERISTICS_FORCE_INTEGRITY	0x0080	Code Integrity checks are enforced.
IMAGE_DLLCHARACTERISTICS_NX_COMPAT	0x0100	Image is NX compatible.
IMAGE_DLLCHARACTERISTICS_NO_ISOLATION	0x0200	Isolation aware, but do not isolate the image.
IMAGE_DLLCHARACTERISTICS_NO_SEH	0x0400	Does not use structured exception (SE) handling. No SE handler may be called in this image.
IMAGE_DLLCHARACTERISTICS_NO_BIND	0x0800	Do not bind the image.
IMAGE_DLLCHARACTERISTICS_APPCONTAINER	0x1000	Image must execute in an AppContainer.
IMAGE_DLLCHARACTERISTICS_WDM_DRIVER	0x2000	A WDM driver.
IMAGE_DLLCHARACTERISTICS_GUD_CF	0x4000	Image supports Control Flow Guard.
IMAGE_DLLCHARACTERISTICS_TERMINAL_SERVER_AWARE	0x8000	Terminal Server aware.

```

}
(0x00F0)
executableImage | lang
}
mat::PE32Plus (0x021
0x0E)
0x1E)
(0x00001000)

```



Pattern Data			
Name	Color	Start	End
sizeOfOptionalHeader	Yellow	0x0000010C	0x00000111
► characteristics	Yellow	0x0000010E	0x00000111
▼ optionalHeader	Blue	0x00000110	0x00000111
magic	Blue	0x00000110	0x00000111
majorLinkerVersion	Orange	0x00000112	0x00000111
minorLinkerVersion	Green	0x00000113	0x00000111
sizeOfCode	Red	0x00000114	0x00000111

Optional Header

Optional Header Data Directories

Each data directory gives the address and size of a table or string that Windows uses. These data directory entries are all loaded into memory so that the system can use them at run time. A data directory is an 8-byte field that has the following declaration:

```
C++ Copy
typedef struct _IMAGE_DATA_DIRECTORY {
    DWORD VirtualAddress;
    DWORD Size;
} IMAGE_DATA_DIRECTORY, *PIMAGE_DATA_DIRECTORY;
```

The first field, VirtualAddress, is actually the RVA of the table. The RVA is the address of the table relative to the base address of the image when the table is loaded. The second field gives the size in bytes. The data directories, which form the last part of the optional header, are listed in the following table.

Note that the number of directories is not fixed. Before looking for a specific directory, check the NumberOfRvaAndSizes field in the optional header.

Also, do not assume that the RVAs in this table point to the beginning of a section or that the sections that contain specific tables have specific names.

Offset (PE/PE32+)	Size	Field	Description
96/112	8	Export Table	The export table address and size. For more information see .edata Section (Image Only).
104/120	8	Import Table	The import table address and size. For more information, see The .idata Section.
112/128	8	Resource Table	The resource table address and size. For more information, see The .rsrc Section.
120/136	8	Exception Table	The exception table address and size. For more information, see The .pdata Section.
128/144	8	Certificate Table	The attribute certificate table address and size. For more information, see The Attribute Certificate Table (Image Only).

...
FO)
oleImage | lang
:PE32Plus (0x021
0001000)

Address	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ASCII
000000C0:	37 6F C6 AB 7E 17 C3 AA 37 6F 3C AA 7D 17 C3 AA 7o. ~. 7o< }.
000000D0:	37 6F C1 AB 7D 17 C3 AA 52 69 63 68 7C 17 C3 AA 7o. }.. Rich!
000000E0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000F0:	00 00 00 00 00 00 00 00 50 45 00 00 64 86 06 00
00000100:	3A 45 A1 7B 00 00 00 00 00 00 F0 00 22 00 :E { "
00000110:	0B 02 0E 1E 00 10 00 00 00 90 00 00 00 00 00 00
00000120:	50 17 00 00 00 10 00 00 00 00 40 01 00 00 00 P. @
00000130:	0A 10 00 00 00 10 00 00 0A 00 00 0A 00 00 00
00000140:	0A 00 00 00 00 00 00 00 0B 00 00 00 10 00 00
00000150:	69 E5 00 00 02 00 60 C1 00 00 08 00 00 00 00 00 i. `
00000160:	00 20 00 00 00 00 00 00 00 00 10 00 00 00 00 00
00000170:	00 10 00 00 00 00 00 00 00 00 10 00 00 00 00 00
00000180:	CC 28 00 00 A0 00 00 00 00 00 00 00 00 00 00 00 (.
00000190:	00 50 00 00 10 47 00 00 00 40 00 00 FC 00 00 P. G. @
000001A0:	00 00 00 00 00 00 00 00 00 00 00 A0 00 00 3C 00 00 <.
000001B0:	78 23 00 00 54 00 00 00 00 00 00 00 00 00 00 00 x# T.
000001C0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001D0:	10 20 00 00 40 01 00 00 00 00 00 00 00 00 00 00
000001E0:	50 21 00 00 40 01 00 00 00 00 00 00 00 00 00 00 P! @
000001F0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000200:	2E 74 65 78 74 00 00 00 50 0C 00 00 00 10 00 00 .text. P.
00000210:	00 10 00 00 00 10 00 00 00 00 00 00 00 00 00 00
00000220:	00 00 00 00 20 00 00 60 2E 72 64 61 74 61 00 00
00000230:	AE 0D 00 00 00 20 00 00 00 10 00 00 00 20 00 00
00000240:	00 00 00 00 00 00 00 00 00 00 40 00 00 40 00 00 @. @.
00000250:	2E 64 61 74 61 00 00 00 C0 06 00 00 30 00 00 .data. 0.
00000260:	00 10 00 00 00 30 00 00 00 00 00 00 00 00 00 00
00000270:	00 00 00 00 40 00 00 00 2E 70 64 61 74 61 00 00
00000280:	00 00 00 00 40 00 00 00 00 00 00 00 00 00 00 00

Page:	0x01 / 0x01
Pattern Data	
Name	Color
sizeOfOptionalHeader	Yellow
characteristics	White
optionalHeader	Blue
magic	Dark Blue
majorLinkerVersion	Orange
minorLinkerVersion	Green
sizeOfCode	Red

Optional Header

Section table

Section Table (Section Headers)

- Section Flags
- Grouped Sections (Object Only)

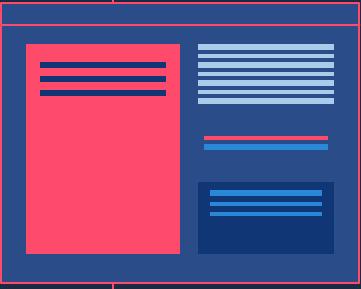
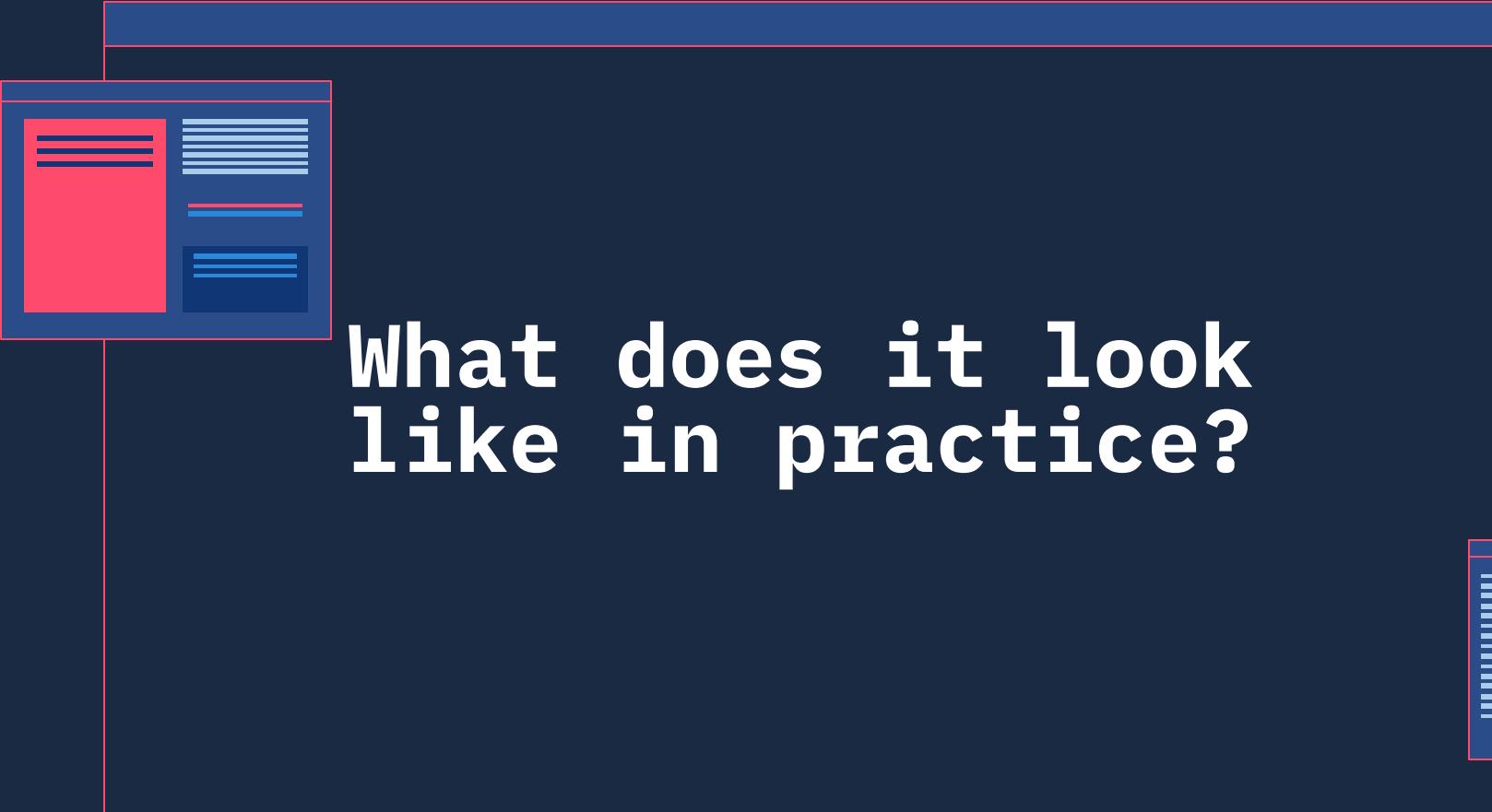
Each row of the section table is, in effect, a section header. This table immediately follows the optional header, if any. This positioning is required because the file header does not contain a direct pointer to the section table. Instead, the location of the section table is determined by calculating the location of the first byte after the headers. Make sure to use the size of the optional header as specified in the file header.

The number of entries in the section table is given by the `NumberOfSections` field in the file header. Entries in the section table are numbered starting from one (1). The code and data memory section entries are in the order chosen by the linker.

In an image file, the VAs for sections must be assigned by the linker so that they are in ascending order and adjacent, and they must be a multiple of the `SectionAlignment` value in the optional header.

Each section header (section table entry) has the following format, for a total of 40 bytes per entry.

Offset	Size	Field	Description
0	8	Name	An 8-byte, null-padded UTF-8 encoded string. If the string is exactly 8 characters long, there is no terminating null. For longer names, this field contains a slash (/) that is followed by an ASCII representation of a decimal number that is an offset into the string table. Executable images do not use a string table and do not support section names longer than 8 characters. Long names in object files are truncated if they are emitted to an executable file.
8	4	VirtualSize	The total size of the section when loaded into memory. If this value is greater than <code>SizeOfRawData</code> , the section is zero-padded. This field is valid only for executable images and should be set to zero for object files.
12	4	VirtualAddress	For executable images, the address of the first byte of the section relative to the image base



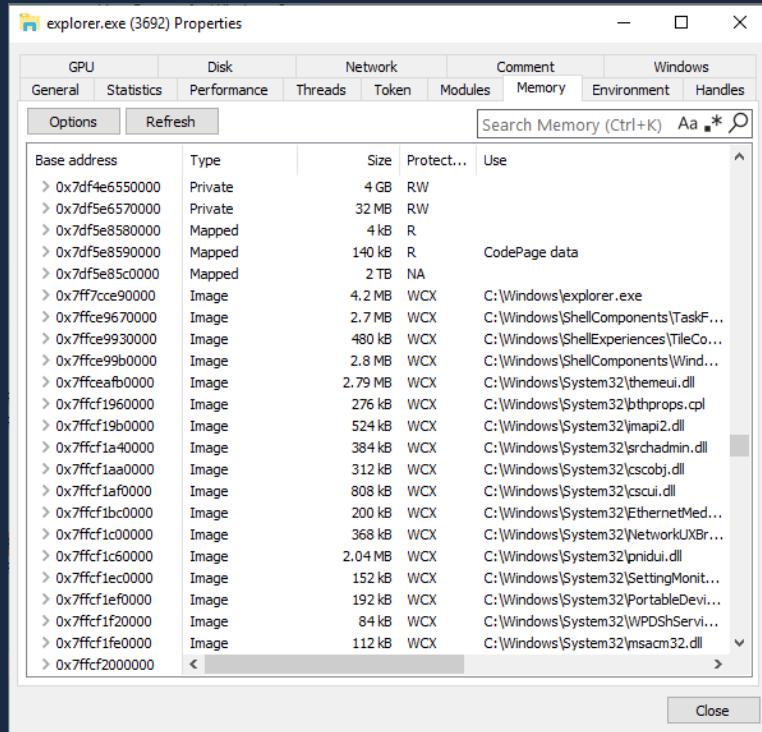
What does it look
like in practice?



Workshop #2

Examine a process' loaded modules
via System Informer

explorer.exe (3692) Properties					
GPU		Disk		Network	
General		Statistics		Performance	
Options Refresh				Search Memory (Ctrl+K) Aa * <input type="text"/>	
Base address	Type	Size	Protect...	Use	
0x0ef0000	Mapped	64 kB	RW	CSRSS Port Heap (ID 2)	
0x0f00000	Mapped	12 kB	R	Activation context data	
0x0f10000	Mapped	104 kB	R	ApiSetMap	
0x0f30000	Private	512 kB	RW	Stack (thread 2012)	
0x0fb0000	Mapped	16 kB	R	System activation context data	
0xfc00000	Mapped	12 kB	R	Process activation context data	
0xfd00000	Private	8 kB	RW	Shim data	
0xfe00000	Mapped	32 kB	R		
0xffff0000	Mapped	16 kB	R	C:\Windows\en-US\explorer.exe.mui	
0x10000000	Private	2 MB	RW		
0x100000000	Mapped	788 kB	R	C:\Windows\System32\locale.nls	
0x12000000	Mapped	4 kB	R	C:\Windows\System32\en-US\Appli...	
0x12d00000	Mapped	4 kB	R	Activation context data	
0x12e00000	Mapped	4 kB	R	C:\Windows\System32\en-US\Input...	
0x12f00000	Mapped	8 kB	R	C:\Windows\System32\en-US\stobje...	
0x13000000	Mapped	8 kB	R		
0x13100000	Mapped	12 kB	R	Activation context data	
0x13200000	Mapped	12 kB	R	Activation context data	
0x13300000	Mapped	12 kB	R	Activation context data	
0x13400000	Mapped	8 kB	R	Activation context data	
0x13500000	Mapped	4 kB	RW	Activation context data	
0x13600000	Private	4 kB	RW		
0x13700000	Mapped	4 kB	R		
0x13800000					



Select the “Memory” tab

explorer.exe (3692) Properties

GPU Disk Network Comment Windows

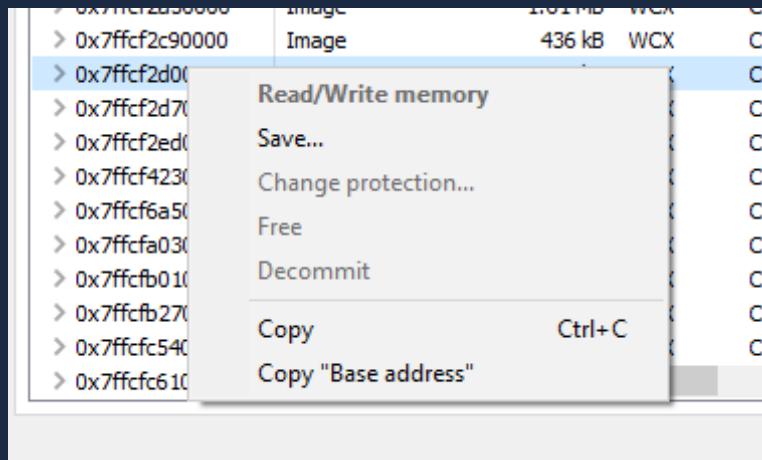
General Statistics Performance Threads Token Modules Memory Environment Handles

Options Refresh Search Memory (Ctrl+K) Aa *

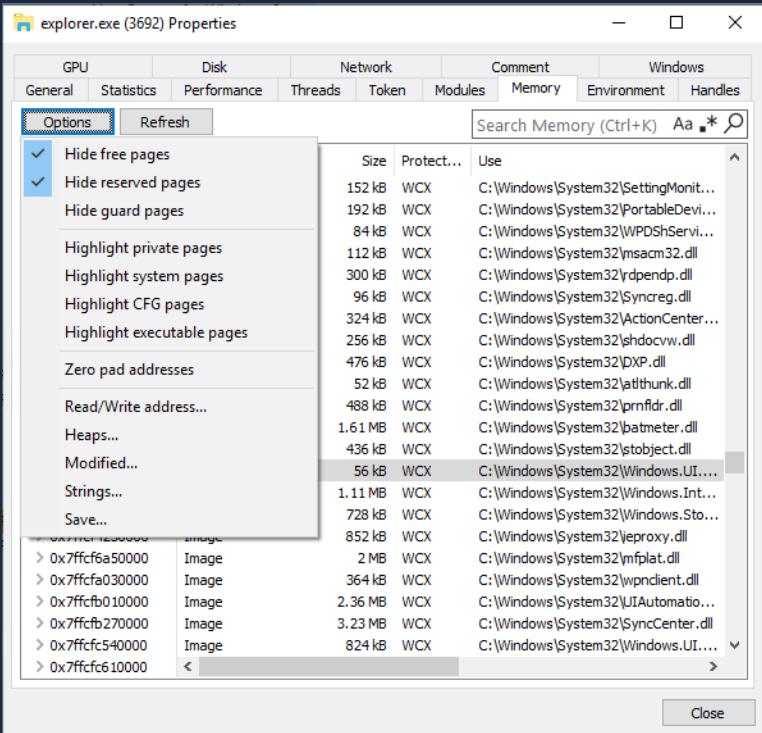
Base address	Type	Size	Protect...	Use
> 0x7df4e6550000	Private	4 GB	RW	
> 0x7df5e6570000	Private	32 MB	RW	
> 0x7df5e8580000	Mapped	4 kB	R	
> 0x7df5e8590000	Mapped	140 kB	R	CodePage data
> 0x7df5e85c0000	Mapped	2 TB	NA	
> 0x7ff7cce90000	Image	4.2 MB	WCX	C:\Windows\explorer.exe
> 0x7ffcce9670000	Image	2.7 MB	WCX	C:\Windows\System32\TaskF...
> 0x7fcce9930000	Image	480 kB	WCX	C:\Windows\System32\ShellExperiences\TileCo...
> 0x7fcce99b0000	Image	2.8 MB	WCX	C:\Windows\System32\ShellComponents\Wind...
> 0x7fceaf80000	Image	2.79 MB	WCX	C:\Windows\System32\themeui.dll
> 0x7fcf1960000	Image	276 kB	WCX	C:\Windows\System32\bthprops.cpl
> 0x7fcf19b0000	Image	524 kB	WCX	C:\Windows\System32\imap2.dll
> 0x7fcf1a40000	Image	384 kB	WCX	C:\Windows\System32\rchadmin.dll
> 0x7fcf1aa0000	Image	312 kB	WCX	C:\Windows\System32\cscoobj.dll
> 0x7fcf1af0000	Image	808 kB	WCX	C:\Windows\System32\cscui.dll
> 0x7fcf1bc0000	Image	200 kB	WCX	C:\Windows\System32\EthernetMed...
> 0x7fcf1c00000	Image	368 kB	WCX	C:\Windows\System32\NetworkUXBr...
> 0x7fcf1c60000	Image	2.04 MB	WCX	C:\Windows\System32\pnidui.dll
> 0x7fcf1ec0000	Image	152 kB	WCX	C:\Windows\System32\SettingMonit...
> 0x7fcf1ef0000	Image	192 kB	WCX	C:\Windows\System32\PortableDevic...
> 0x7fcf1f20000	Image	84 kB	WCX	C:\Windows\System32\WPDShServi...
> 0x7fcf1fe0000	Image	112 kB	WCX	C:\Windows\System32\msacm32.dll
> 0x7fcf2000000				

Close

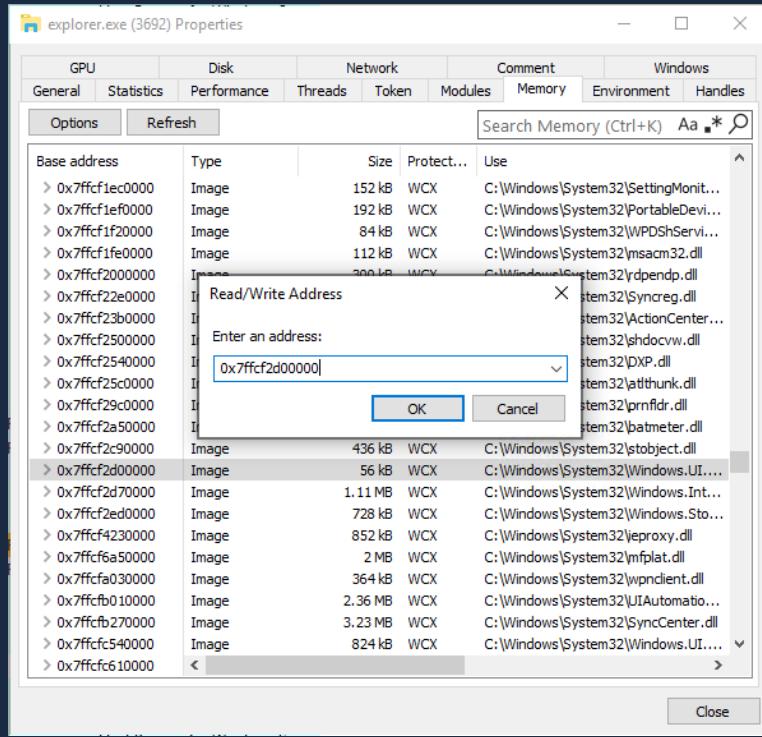
Select the “Memory” tab, scroll down to rows labeled “Image”



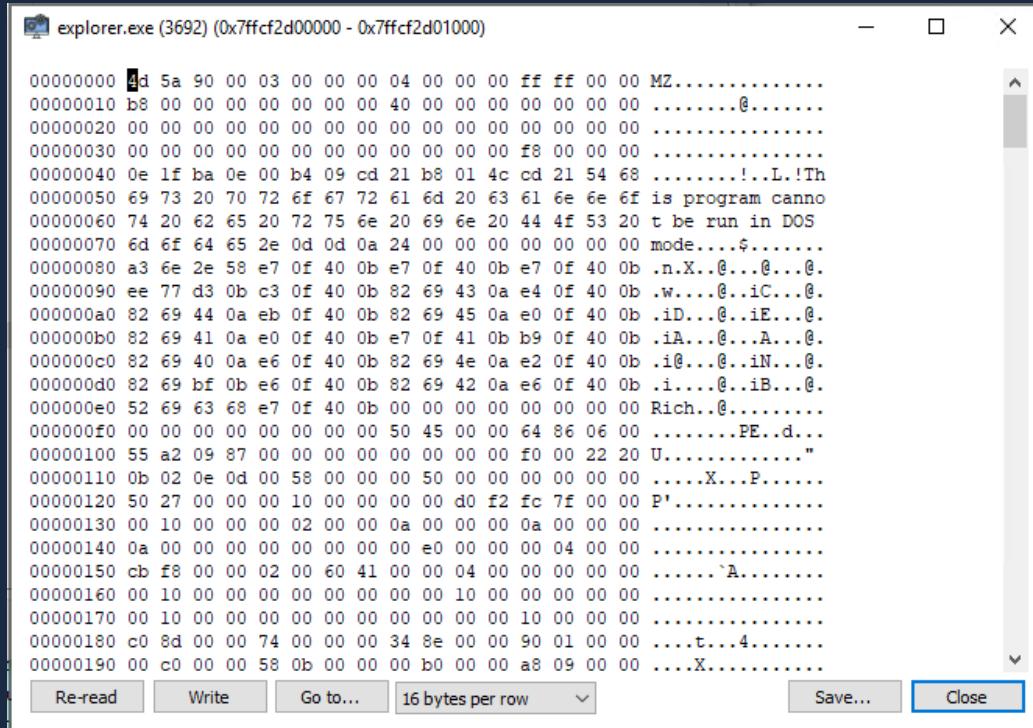
Select any of them, right click, Copy "Base address"



Click “Options” > “Read/Write address...”



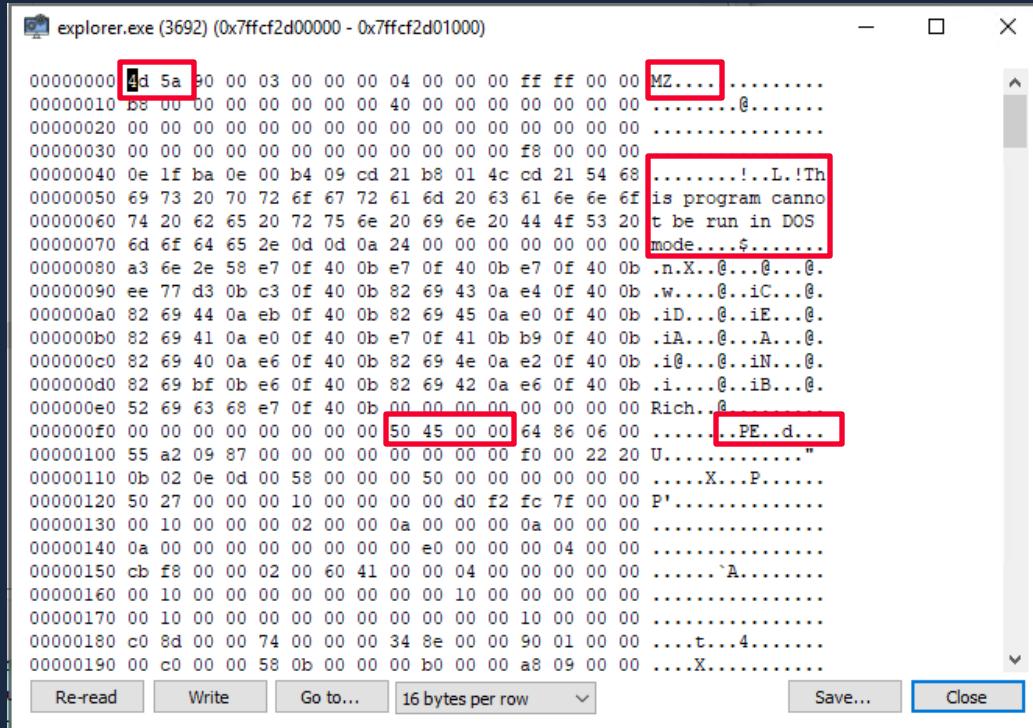
Paste the address you copied



The screenshot shows a Windows Task Manager window titled "explorer.exe (3692) (0x7ffcf2d00000 - 0x7ffcf2d01000)". The main pane displays the raw memory dump of the process. The memory starts with the MZ header, followed by various program sections and resources. The dump is presented in hex format, with each row containing 16 bytes per row. At the bottom of the window, there are several buttons: "Re-read", "Write", "Go to...", "16 bytes per row" (with a dropdown arrow), "Save...", and "Close".

```
00000000 4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00 MZ.....
00000010 b8 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00 .....@....
00000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000040 0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 54 68 .....!..L!Th
00000050 69 73 20 70 72 6f 67 72 61 6d 20 63 61 6e 6e 6f is program canno
00000060 74 20 62 65 20 72 75 6e 20 69 6e 20 44 4f 53 20 t be run in DOS
00000070 6d 6f 64 65 2e 0d 0d 0a 24 00 00 00 00 00 00 mode.....$....
00000080 a3 6e 2e 58 e7 0f 40 0b e7 0f 40 0b e7 0f 40 0b .n.X...@...@...@.
00000090 ee 77 d3 0b c3 0f 40 0b 82 69 43 0a e4 0f 40 0b .w....@..iC...@.
000000a0 82 69 44 0a eb 0f 40 0b 82 69 45 0a e0 0f 40 0b .iD...@..iE...@.
000000b0 82 69 41 0a e0 0f 40 0b e7 0f 41 0b b9 0f 40 0b .iA...@...A...@.
000000c0 82 69 40 0a e6 0f 40 0b 82 69 4e 0a e2 0f 40 0b .i@...@..iN...@.
000000d0 82 69 bf 0b e6 0f 40 0b 82 69 42 0a e6 0f 40 0b .i....@..iB...@.
000000e0 52 69 63 68 e7 0f 40 0b 00 00 00 00 00 00 00 00 Rich...@.....
000000f0 00 00 00 00 00 00 00 50 45 00 00 64 86 06 00 .....PE.d...
00000100 55 a2 09 87 00 00 00 00 00 00 00 f0 00 22 20 U....."
00000110 0b 02 0e 0d 00 58 00 00 00 50 00 00 00 00 00 00 .....X...P.....
00000120 50 27 00 00 00 10 00 00 00 00 d0 f2 fc 7f 00 00 P'.....
00000130 00 10 00 00 00 02 00 00 0a 00 00 00 0a 00 00 00 .....
00000140 0a 00 00 00 00 00 00 e0 00 00 00 04 00 00 00 00 .....
00000150 cb f8 00 00 02 00 60 41 00 00 04 00 00 00 00 00 .....`A.....
00000160 00 10 00 00 00 00 00 00 00 00 10 00 00 00 00 00 .....
00000170 00 10 00 00 00 00 00 00 00 00 00 00 10 00 00 00 .....
00000180 c0 8d 00 00 74 00 00 00 34 8e 00 00 90 01 00 00 .....t...4.....
00000190 00 c0 00 00 58 0b 00 00 b0 00 00 a8 09 00 00 .....X.....
```

This is what the loaded PE module in memory looks like!



MZ + PE headers

explorer.exe (3692) (0x7ffcf2d00000 - 0x7ffcf2d01000)

```
00000180 c0 8d 00 00 74 00 00 00 34 8e 00 00 90 01 00 00 .....t....4.....
00000190 00 c0 00 00 58 0b 00 00 00 b0 00 00 a8 09 00 00 .....X.....
000001a0 00 00 00 00 00 00 00 d0 00 00 58 01 00 00 .....X.....
000001b0 e0 7a 00 00 54 00 00 00 00 00 00 00 00 00 00 00 .z..T.....
000001c0 00 00 00 00 00 00 00 48 73 00 00 28 00 00 00 .....Hs..(...
000001d0 40 72 00 00 08 01 00 00 00 00 00 00 00 00 00 00 @r.....
000001e0 b8 73 00 00 d0 02 00 00 00 00 00 00 00 00 00 00 .s.....
000001f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
00000200 2e 74 65 78 74 00 00 00 c6 57 00 00 10 00 00 .text....W....
00000210 00 58 00 00 00 04 00 00 00 00 00 00 00 00 00 00 .X.....
00000220 00 00 00 00 20 00 00 60 2e 72 64 61 74 61 00 00 .....`rdata...
00000230 9a 2a 00 00 00 70 00 00 00 2c 00 00 00 5c 00 00 .*..p...,..\.
00000240 00 00 00 00 00 00 00 00 00 00 00 40 00 00 40 .....@..@...
00000250 2e 64 61 74 61 00 00 00 28 0b 00 00 00 a0 00 00 .data...(....
00000260 00 04 00 00 00 88 00 00 00 00 00 00 00 00 00 00 ...
00000270 00 00 00 00 40 00 00 c0 2e 70 64 61 74 61 00 00 .....@...pdata...
00000280 a8 09 00 00 00 b0 00 00 00 0a 00 00 00 8c 00 00 ...
00000290 00 00 00 00 00 00 00 00 00 00 00 40 00 00 40 .....@..@...
000002a0 2e 72 73 72 63 00 00 00 58 0b 00 00 00 c0 00 00 .rsrc..X....
000002b0 00 0c 00 00 00 96 00 00 00 00 00 00 00 00 00 00 ...
000002c0 00 00 00 00 40 00 00 40 2e 72 65 6c 6f 63 00 00 .....@..@.reloc...
000002d0 58 01 00 00 00 q0 00 00 00 02 00 00 00 a2 00 00 X...
000002e0 00 00 00 00 00 00 00 00 00 00 00 40 00 00 42 .....@..B...
000002f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
00000300 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
00000310 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
```

Re-read Write Go to... 16 bytes per row Save... Close

Common section names



What if I take those headers away?

Dump w/ PE header in tact = OK

OllyDumpEx v1.80 - msieexec.exe

Module: Module C:\Windows\System32\msieexec.exe
 Memory 0000000140000000 (00001000) / Imag / R / msieexec / PE
 Address 7FFCEF0A0000

Dump Cancel

List Section: Base Only All Memory Address Range 7FFCEF0A0000 - 7FFCF20A0000
Dump Mode: Rebuild Binary (Raw) Binary (Virtual)
Header Source: Memory File Dummy /PE32+ EXE ReScan Image

Search Area: Select All Memory (exclude listed module) Search Image

Search Mode: Strict Fuzzy (slow)

Format: PE ELF

Image: Image Base: 7FFCEF0A0000 Fix Virtual Offset
Image Size: 0004CC00
Entry Point: 00002D40 Get RIP as DEP

Option: Prefer Original Characteristics
 Fix Corrupted Image Header Structure
 Disable Relocation
 Auto Adjust Image Base Address
 Rebuild Data Directory (Follow ImageBase Change)
 Search All Occurrences

Section: Select All Select BaseModule Select Private/All Select Private/Exec DeSelect All

Address	Size	Owner	Section	Type	Access	VirtualOffset	VirtualSize	Character...
7FFCEF0A040000	00027000	aclayers	.text	Imag	R	00001000	00026EOF	60000020
7FFCEF0C740000	0001CA00	aclayers	.rdata	Imag	R E	00028000	0001C8EE	40000040
7FFCEF0E30000	00062000	aclayers	.data	Imag	R	00045000	00447E04	C0000040
7FFCEF0E40000	0001C000	aclayers	.pdata	Imag	R W	00480000	00001B48	40000040
7FFCEF0E80000	00000800	aclayers	.rsrc	Imag	R W	0048F000	00000408	40000040
7FFCEF0E90000	00000400	aclayers	.reloc	Imag	R W	00490000	00000880	42000040

PE32+ x86_64 DLL from Memory, BaseAddress=7FFCEF0A0000, BinaryDump, Relocation



Dump w/o PE header = ???

OllyDumpEx v1.80 - msieexec.exe

Module: Module: C:\Windows\System32\msieexec.exe
 Memory: 00007FFCEFOA0000 (00001000) / Imag / R / aclayers / PE
 Address: 7FFCEFOA0000

Dump: Dump
 Cancel

List Section: Base Only All Memory Address Range 7FFCEFOA0000 - 7FFCF20A0000
Dump Mode: Rebuild Binary (Raw) Binary (Virtual)
Header Source: Memory File Dummy | PE32+ EXE | Rescan Image
Search: Select All Memory (exclude listed module) | Search Image
Search Mode: Strict Fuzzy (slow)
Search Result: PE
 ELF

Image:
Image Base: 00000000 Fix Virtual Offset
Image Size: 00000000
Entry Point: 00000000 Get RIP as DEP
Option:
 Prefer Original Characteristics
 Fix Corrupted Image Header Structure
 Disable Relocation
 Auto Adjust Image Base Address
 Rebuild DataDirectory (Follow ImageBase Change)
 Search All Occurrences

Section:
 Select All Select BaseModule Select Private/All Select Private/Exec DeSelect All

Address	Size	Owner	Section	Type	Access	VirtualOffset	VirtualSize	Character...
00007FFCEFOA0000	00000000							

Search failed, BaseAddress invalid, Address=7FFCEFOA0000 Size=-

Dump w/o PE header = ???



OllyDumpEx v1.80 - msieexec.exe

Module
Base: Module C:\Windows\System32\msieexec.exe
 Memory 00007FFCEFOA0000 (00001000) / Imag / R / aclayers / PE
 Address 7FFCEFOA0000

List Section: Base Only All Memory Address Range 7FFCEFOA0000 - 7FFCF20A0000
Dump Mode: Rebuild Binary (Raw) Binary (Virtual)

Dump Cancel

We'll come back to this in the future
:)

Address Size Owner Section Type Access VirtualOffset VirtualSize Character...

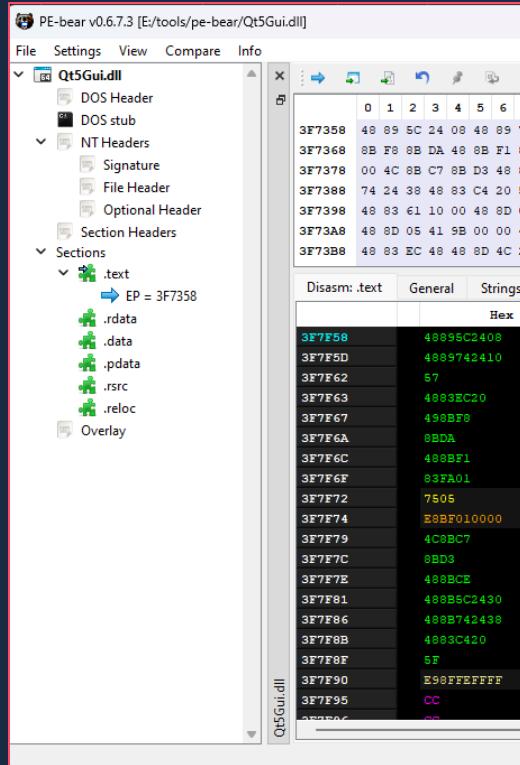
Search failed, BaseAddress invalid, Address=7FFCEFOA0000 Size=-



"ok but I ain't bout to bust out a hex editor every time"

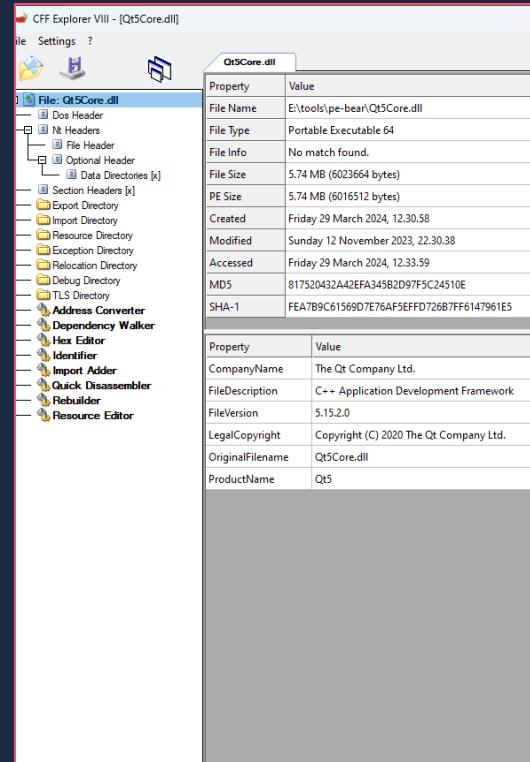
PE Examining Tools

- PE-Bear
 - Developed by hasherezade
 - “PE-bear is a multiplatform reversing tool for PE files. Its objective is to deliver fast and flexible “first view” for malware analysts, stable and capable to handle malformed PE files.”



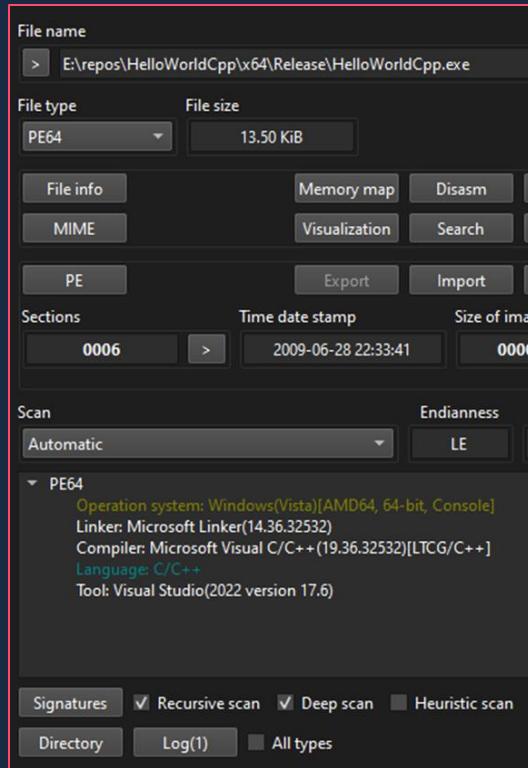
PE Examining Tools

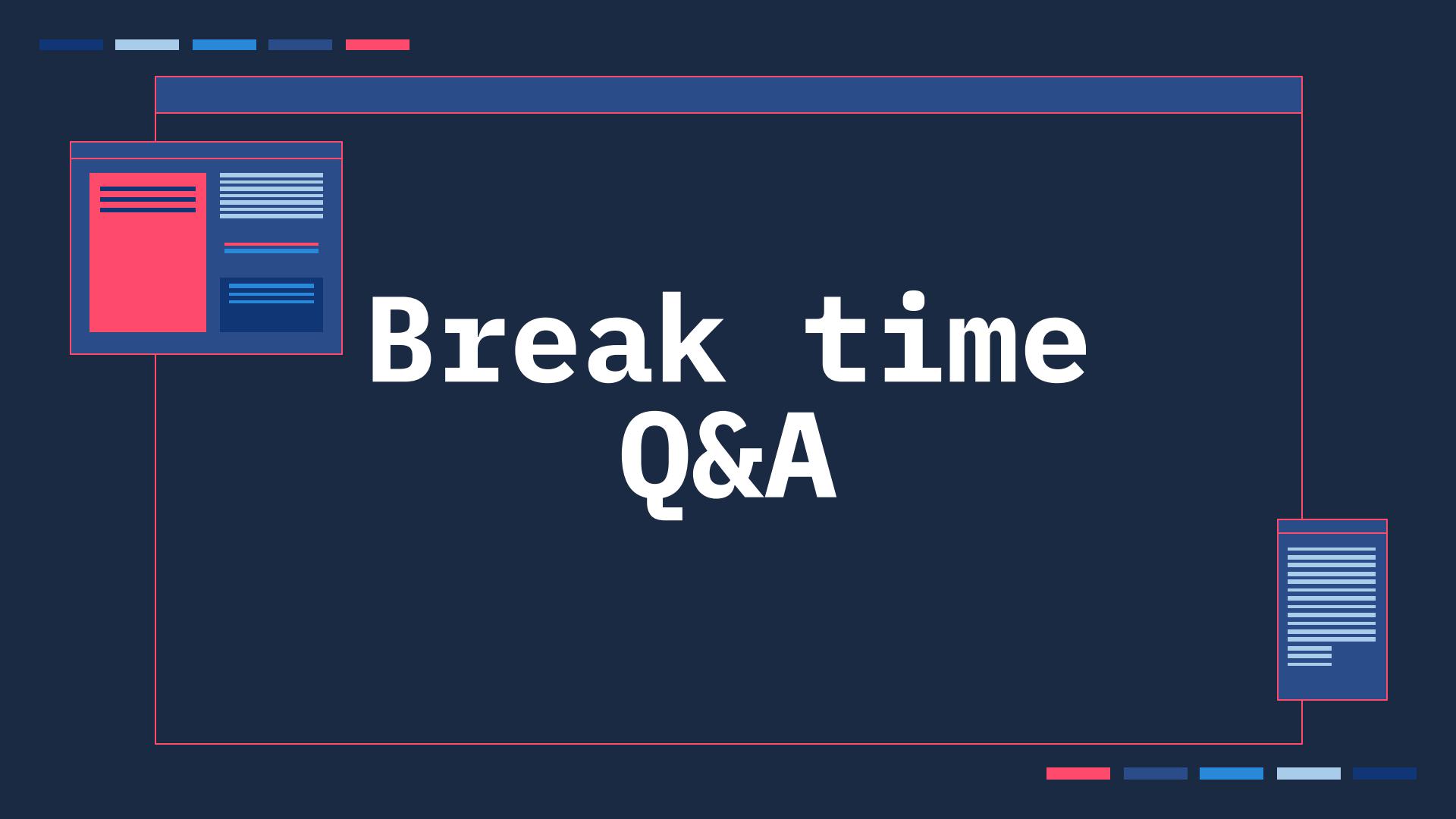
- CFF Explorer
 - Developed by NTCore / Erik Pistelli (@erikpistelli)
 - “CFF Explorer was designed to make PE editing as easy as possible, but without losing sight on the portable executable’s internal structure.”



PE Examining Tools

- Detect it Easy
 - Developed by Hors (@horsicq)
 - "DIE" is a cross-platform application, apart from Windows version there are also available versions for Linux and Mac OS.



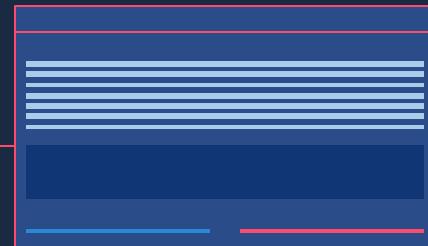
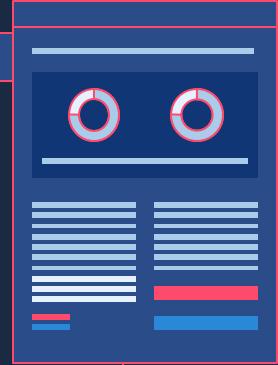


Break time Q&A



04

Core Windows Security Concepts



Memory Permissions

- Windows & most modern OS have the concept of permissions when it comes with memory, most commonly:
 - Read (R)
 - Write (W)
 - Execute (X)



Memory Permissions

- Here's a region of memory that is set with Read (R) + Execute (X)...

```
mov qword ptr [rsp + 8], rbx  
mov qword ptr [rsp + 0x10], rsi  
push rdi  
sub rsp, 0x20  
...
```

Memory Permissions

- Here's a region of memory that is set with Read (R) + Execute (X)...

```
mov qword ptr [rsp + 8], rbx
mov qword ptr [rsp + 0x10], rsi
push rdi
sub rsp, 0x20
...
```

Memory Permissions

- Here's a region of memory that is set with Read (R) + Execute (X)...

```
mov qword ptr [rsp + 8], rbx  
mov qword ptr [rsp + 0x10], rsi  
push rdi  
sub rsp, 0x20  
...
```

Memory Permissions

- Here's a region of memory that is set with Read (R) + Execute (X)...

```
mov qword ptr [rsp + 8], rbx  
mov qword ptr [rsp + 0x10], rsi  
push rdi  
sub rsp, 0x20  
...
```



Memory Permissions

- Here's the same page but only Read/Write (R/W)...

```
mov qword ptr [rsp + 8], rbx  
mov qword ptr [rsp + 0x10], rsi  
push rdi  
sub rsp, 0x20  
...
```

Memory Permissions

- Here's the same page but only Read/Write (R/W)...

```
mov qword ptr [rsp + 8], rbx
mov qword ptr [rsp + 0x10], rsi
push rdi
sub rsp, 0x20
...
```

STATUS_ACCESS_VIOLATION



Data Execution Prevention
(DEP)

Memory Permissions

- How allocating a memory works in Windows
 - Developer calls VirtualAlloc
 - VirtualAlloc -> NtAllocateVirtualMemory



Memory Permissions

- How allocating a memory works in Windows
 - Developer calls VirtualAlloc
 - VirtualAlloc -> NtAllocateVirtualMemory

```
...  
LPVOID VirtualAlloc(  
    [in, optional] LPVOID lpAddress,  
    [in]          SIZE_T dwSize,  
    [in]          DWORD  flAllocationType,  
    [in]          DWORD  flProtect  
);
```



Memory Permissions

```
...  
LPVOID VirtualAlloc(  
    [in, optional] LPVOID lpAddress,  
    [in]          SIZE_T dwSize,  
    [in]          DWORD  flAllocationType,  
    [in]          DWORD  flProtect  
);
```

Constants

[Expand table](#)

Constant/value	Description
PAGE_EXECUTE 0x10	Enables execute access to the committed region of pages. An attempt to write to the committed region results in an access violation. This flag is not supported by the CreateFileMapping function.
PAGE_EXECUTE_READ 0x20	Enables execute or read-only access to the committed region of pages. An attempt to write to the committed region results in an access violation. Windows Server 2003 and Windows XP: This attribute is not supported by the CreateFileMapping function until Windows XP with SP2 and Windows Server 2003 with SP1.
PAGE_EXECUTE_READWRITE 0x40	Enables execute, read-only, or read/write access to the committed region of pages. Windows Server 2003 and Windows XP: This attribute is not supported by the CreateFileMapping function until Windows XP with SP2 and Windows Server 2003 with SP1.
PAGE_EXECUTE_WRITECOPY 0x80	Enables execute, read-only, or copy-on-write access to a mapped view of a file mapping object. An attempt to write to a committed copy-on-write page results in a private copy of the page being made for the process. The private page is marked as PAGE_EXECUTE_READWRITE, and the change is written to the new page. This flag is not supported by the VirtualAlloc or VirtualAllocEx functions. Windows Vista, Windows Server 2003 and Windows XP: This attribute is not supported by the CreateFileMapping function until Windows Vista with SP1 and Windows Server 2008.
PAGE_NOACCESS 0x01	Disables all access to the committed region of pages. An attempt to read from, write to, or execute the committed region results in an access violation. This flag is not supported by the CreateFileMapping function.
PAGE_READONLY 0x02	Enables read-only access to the committed region of pages. An attempt to write to the committed region results in an access violation. If Data Execution Prevention is enabled, an attempt to execute code in the committed region results in an access violation.
PAGE_READWRITE 0x04	Enables read-only or read/write access to the committed region of pages. If Data Execution Prevention is enabled, attempting to execute code in the committed region results in an access violation.
PAGE_WRITECOPY 0x08	Enables read-only or copy-on-write access to a mapped view of a file mapping object. An attempt to write to a committed copy-on-write page results in a private copy of the page being made for the process. The private page is marked as PAGE_READWRITE, and the change is written to the new page. If Data Execution Prevention is enabled, attempting to execute code in the committed region results in an access violation. This flag is not supported by the VirtualAlloc or VirtualAllocEx functions.

Memory Permissions

- You can also change the memory permission at runtime with `VirtualProtect`

```
•••  
BOOL VirtualProtect(  
    [in]  LPVOID lpAddress,  
    [in]  SIZE_T dwSize,  
    [in]  DWORD  flNewProtect,  
    [out] PDWORD lpflOldProtect  
);
```

Workshop #3

Memory permissions workshop

- VirtualAlloc / VirtualProtect usage
- Typical shellcode scenario

```
int main(void) {
    // Based on https://github.com/vyrus001/shellGo/blob/master/main.go
    // Spawns calc via WinExec
    unsigned char shellcode[] = {
        0x50, 0x51, 0x52, 0x53, 0x56, 0x57, 0x55, 0x6A, 0x60, 0x5A, 0x68, 0x
        0x59, 0x48, 0x83, 0xEC, 0x28, 0x65, 0x48, 0x8B, 0x32, 0x48, 0x8B, 0x
        0x10, 0x48, 0xAD, 0x48, 0x8B, 0x30, 0x48, 0x8B, 0x7E, 0x30, 0x03, 0x
        0x28, 0x8B, 0x74, 0x1F, 0x20, 0x48, 0x01, 0xFE, 0x8B, 0x54, 0x1F, 0x
        0x8D, 0x52, 0x02, 0xAD, 0x81, 0x3C, 0x07, 0x57, 0x69, 0x6E, 0x45, 0x
        0x1C, 0x48, 0x01, 0xFE, 0x8B, 0x34, 0xAE, 0x48, 0x01, 0xF7, 0x99, 0x
        0x30, 0x5D, 0x5F, 0x5E, 0x5B, 0x5A, 0x59, 0x58, 0xC3,
    };

    // Allocate a memory region with RWX permissions
    void* exec_mem = VirtualAlloc(0, sizeof(shellcode), MEM_COMMIT | MEM_RES
    if (!exec_mem) {
        printf("Memory allocation failed\n");
        return 1;
    }

    // Copy the shellcode to the allocated memory
    RtlMoveMemory(exec_mem, shellcode, sizeof(shellcode));

    // Cast the allocated memory as a function and execute it
    exec_shellcode(exec_mem);
}
```

```
int main(void) {
    // Based on https://github.com/vyrus001/shellGo/blob/master/main.go
    // Spawns calc via WinExec
    unsigned char shellcode[] = {
        0x50, 0x51, 0x52, 0x53, 0x56, 0x57, 0x55, 0x6A, 0x60, 0x5A, 0x68, 0x63, 0x61, 0x6C,
        0x63, 0x54,
        0x59, 0x48, 0x83, 0xEC, 0x28, 0x65, 0x48, 0x8B, 0x32, 0x48, 0x8B, 0x76, 0x18, 0x48,
        0x8B, 0x76,
        0x10, 0x48, 0xAD, 0x48, 0x8B, 0x30, 0x48, 0x8B, 0x7E, 0x30, 0x03, 0x57, 0x3C, 0x8B,
        0x5C, 0x17,
        0x28, 0x8B, 0x74, 0x1F, 0x20, 0x48, 0x01, 0xFE, 0x8B, 0x54, 0x1F, 0x24, 0x0F, 0xB7,
        0x2C, 0x17,
        0x8D, 0x52, 0x02, 0xAD, 0x81, 0x3C, 0x07, 0x57, 0x69, 0x6E, 0x45, 0x75, 0xEF, 0x8B,
        0x74, 0x1F,
        0x1C, 0x48, 0x01, 0xFE, 0x8B, 0x34, 0xAE, 0x48, 0x01, 0xF7, 0x99, 0xFF, 0xD7, 0x48,
        0x83, 0xC4,
        0x30, 0x5D, 0x5F, 0x5E, 0x5B, 0x5A, 0x59, 0x58, 0xC3,
    };
    // Allocate a memory region with RWX permissions
    void* exec_mem = VirtualAlloc(0, sizeof(shellcode), MEM_COMMIT | MEM_RESERVE,
        PAGE_EXECUTE_READWRITE);
    if (!exec_mem) {
        // Error handling
    }
}
```

Shellcode for calling
WinExec("calc")

```
,  
  
    // Allocate a memory region with RWX permissions  
    void* exec_mem = VirtualAlloc(0, sizeof(shellcode), MEM_COMMIT | MEM_RESERVE,  
        PAGE_EXECUTE_READWRITE);  
    if (!exec_mem) {  
        printf("Memory allocation failed\n");  
        return 1;  
    }  
  
    // Copy the shellcode to the allocated memory  
    RtlMoveMemory(exec_mem, shellcode, sizeof(shellcode));  
  
    // Cast the allocated memory as a function and execute it  
    exec_shellcode(exec_mem);  
  
    // Change memory to RW and try to execute again  
    DWORD old{};  
    VirtualProtect(exec_mem, sizeof(shellcode), PAGE_READWRITE, &old);  
    exec_shellcode(exec_mem);  
  
    return 0;  
}
```

Use `VirtualAlloc` to
allocate a `RWX` memory

```
void exec_shellcode(void* addr) {
    __try {
        ((void(*)())addr)();
        printf("[+] Executed %p\n", addr);
    }
    __except (1) {
        printf("[-] Exception %08lx\n", GetExceptionCode());
    }
    return;
}

int main(void) {
    // Based on https://github.com/vyrus001/shellGo/blob/master/main.go
    // Spawns calc via WinExec
    unsigned char shellcode[] = {
        0x50, 0x51, 0x52, 0x53, 0x56, 0x57, 0x55, 0x6A, 0x60, 0x5A, 0x68, 0x63, 0x61, 0x6C,
        0x63, 0x54,
        0x59, 0x48, 0x83, 0xEC, 0x28, 0x65, 0x48, 0x8B, 0x32, 0x48, 0x8B, 0x76, 0x18, 0x48,
        0x8B, 0x76,
        0x10, 0x48, 0xAD, 0x48, 0x8B, 0x30, 0x48, 0x8B, 0x7E, 0x30, 0x03, 0x57, 0x3C, 0x8B,
        0x5C, 0x17,
        0x28, 0x8B, 0x74, 0x1F, 0x20, 0x48, 0x01, 0xFE, 0x8B, 0x54, 0x1F, 0x24, 0x0F, 0xB7,
```

Try to execute the shellcode, and if it throws, catch the exception

```
,  
  
    // Allocate a memory region with RWX permissions  
    void* exec_mem = VirtualAlloc(0, sizeof(shellcode), MEM_COMMIT | MEM_RESERVE,  
    PAGE_EXECUTE_READWRITE);  
    if (!exec_mem) {  
        printf("Memory allocation failed\n");  
        return 1;  
    }  
  
    // Copy the shellcode to the allocated memory  
    RtlMoveMemory(exec_mem, shellcode, sizeof(shellcode));  
  
    // Cast the allocated memory as a function and execute it  
    exec_shellcode(exec_mem);  
  
    // Change memory to RW and try to execute again  
    DWORD old{};  
    VirtualProtect(exec_mem, sizeof(shellcode), PAGE_READWRITE, &old);  
    exec_shellcode(exec_mem);  
  
    return 0;  
}
```

Try again by changing
the memory permission
to RW without the X

Calculator

Log Script Symbols Source Threads Handles Trace

Standard

0

MC	MR	M+	M-	MS	M ⁺
%	√	x^2	$\frac{1}{x}$		
CE	C	⊗	÷		
7	8	9	×		
4	5	6	—		
1	2	3	+		
±	0	.	=		

00000000140001244
00000000140001245
00000000140001246

CC CC

CCCC10E1E58400_00000000

```
mov rbx,rax
test rax,rax
jne shellcode-demo.1400011B6
lea rcx,qword ptr ds:[140002290]
call shellcode-demo.140001010
lea eax,qword ptr ds:[rbx+1]
jmp shellcode-demo.140001222
movaps xmm0,xmmword ptr ss:[rbp-29]
mov rcx,rbx
movaps xmm1,xmmword ptr ss:[rbp-19]
movups xmmword ptr ds:[rax],xmm0
movaps xmm0,xmmword ptr ss:[rbp-9]
movups xmmword ptr ds:[rax+10],xmm1
movaps xmm1,xmmword ptr ss:[rbp+7]
movups xmmword ptr ds:[rax+20],xmm0
movaps xmm0,xmmword ptr ss:[rbp+17]
movups xmmword ptr ds:[rax+30],xmm1
movaps xmm1,xmmword ptr ss:[rbp+27]
movups xmmword ptr ds:[rax+40],xmm0
movsd xmm0,qword ptr ss:[rbp+37]
movups xmmword ptr ds:[rax+50],xmm1
movsd qword ptr ds:[rax+60],xmm0
mov byte ptr ds:[rax+68],C3
call shellcode-demo.140001070
mov edx,69
mov dword ptr ss:[rbp-39],0
lea r9,qword ptr ss:[rbp-39]
mov rcx,rbx
lea r8d,qword ptr ds:[rdx-65]
call qword ptr ds:[<&VirtualProtect>]
mov rcx,rbx
call shellcode-demo.140001070
xor eax,eax
mov rcx,qword ptr ss:[rbp+47]
xor rcx,rsp
call shellcode-demo.140001250
mov rbx,qword ptr ss:[rsp+C0]
add rsp,80
pop rbp
ret
int3
int3
int3
int3
int3
int3
int3
int3
nop word ptr ds:[rax+40],0x
```

00000000140002290

...works great the first time!

```
7 movaps xmm1,xmmword ptr ss:[rbp+27]
0 movups xmmword ptr ds:[rax+40],xmm0
5 37 movsd xmm0,qword ptr ss:[rbp+37]
0 movups xmmword ptr ds:[rax+50],xmm1
0 60 movsd qword ptr ds:[rax+60],xmm0
C3 mov byte ptr ds:[rax+68],C3
FFF call shellcode-demo.140001070
0000 mov edx,69
00000000 69:'i'
C7 mov dword ptr ss:[rbp-39],0
9B lea r9,qword ptr ss:[rbp-39]
D0000 mov rcx,rbx
FFF lea r8d,qword ptr ds:[rdx-65]
call qword ptr ds:[<&VirtualProtect>]
0000 mov rcx,rbx
47 call shellcode-demo.140001070
xor eax,eax
mov rcx,qword ptr ss:[rbp+47]
xor rcx,rsp
0000 call shellcode-demo.140001250
4 C0000000 000001604E990000
B0000000 [-] Exception c0000005
F8400 00000000
A92D0000
10
FFFF
```

...not so great once we changed the perms to RW!

C:\Space\shellcode-demo.exe

[+] Executed 000001604E990000
[-] Exception c0000005

Takeaway

- VirtualAlloc / VirtualProtect with RWX are almost always a red flag when dealing with *known* malware; great for placing breakpoint.

```
int main(void) {
    // Based on https://github.com/vyrus001/shellGo/blob/master/main.go
    // Spawns calc via WinExec
    unsigned char shellcode[] = {
        0x50, 0x51, 0x52, 0x53, 0x56, 0x57, 0x55, 0x6A, 0x60, 0x5A, 0x68, 0x
        0x59, 0x48, 0x83, 0xEC, 0x28, 0x65, 0x48, 0x8B, 0x32, 0x48, 0x8B, 0x
        0x10, 0x48, 0xAD, 0x48, 0x8B, 0x30, 0x48, 0x8B, 0x7E, 0x30, 0x03, 0x
        0x28, 0x8B, 0x74, 0x1F, 0x20, 0x48, 0x01, 0xFE, 0x8B, 0x54, 0x1F, 0x
        0x8D, 0x52, 0x02, 0xAD, 0x81, 0x3C, 0x07, 0x57, 0x69, 0x6E, 0x45, 0x
        0x1C, 0x48, 0x01, 0xFE, 0x8B, 0x34, 0xAE, 0x48, 0x01, 0xF7, 0x99, 0x
        0x30, 0x5D, 0x5F, 0x5E, 0x5B, 0x5A, 0x59, 0x58, 0xC3,
    };

    // Allocate a memory region with RWX permissions
    void* exec_mem = VirtualAlloc(0, sizeof(shellcode), MEM_COMMIT | MEM_RES
    if (!exec_mem) {
        printf("Memory allocation failed\n");
        return 1;
    }

    // Copy the shellcode to the allocated memory
    RtlMoveMemory(exec_mem, shellcode, sizeof(shellcode));

    // Cast the allocated memory as a function and execute it
    exec_shellcode(exec_mem);
}
```

Takeaway

- VirtualAlloc / VirtualProtect with RWX are almost always a red flag when dealing with *known* malware; great for placing breakpoint.
- But not always - there are legitimate uses that involve RWX memory!
 - Dynamic code generation/JIT
 - Custom memory allocators

```
int main(void) {
    // Based on https://github.com/vyrus001/shellGo/blob/master/main.go
    // Spawns calc via WinExec
    unsigned char shellcode[] = {
        0x50, 0x51, 0x52, 0x53, 0x56, 0x57, 0x55, 0x6A, 0x60, 0x5A, 0x68, 0x
        0x59, 0x48, 0x83, 0xEC, 0x28, 0x65, 0x48, 0x8B, 0x32, 0x48, 0x8B, 0x
        0x10, 0x48, 0xAD, 0x48, 0x8B, 0x30, 0x48, 0x8B, 0x7E, 0x30, 0x03, 0x
        0x28, 0x8B, 0x74, 0x1F, 0x20, 0x48, 0x01, 0xFE, 0x8B, 0x54, 0x1F, 0x
        0x8D, 0x52, 0x02, 0xAD, 0x81, 0x3C, 0x07, 0x57, 0x69, 0x6E, 0x45, 0x
        0x1C, 0x48, 0x01, 0xFE, 0x8B, 0x34, 0xAE, 0x48, 0x01, 0xF7, 0x99, 0x
        0x30, 0x5D, 0x5F, 0x5E, 0x5B, 0x5A, 0x59, 0x58, 0xC3,
    };

    // Allocate a memory region with RWX permissions
    void* exec_mem = VirtualAlloc(0, sizeof(shellcode), MEM_COMMIT | MEM_RES
    if (!exec_mem) {
        printf("Memory allocation failed\n");
        return 1;
    }

    // Copy the shellcode to the allocated memory
    RtlMoveMemory(exec_mem, shellcode, sizeof(shellcode));

    // Cast the allocated memory as a function and execute it
    exec_shellcode(exec_mem);
}
```

ASLR

- ASLR = Address Space Layout Randomization
- ASLR randomizes memory location for programs.
- Protects against exploits by making memory addresses unpredictable.



```

RDX R3 00000001400014D0 shellcode-c 48:83EC 28 sub rsp,28
00000001400014D4 E8 C7030000 call shellcode-demo.1400018A0
00000001400014D8 48:83C4 28 add rsp,28
00000001400014D0 E9 72FFFFFF jmp shellcode-demo.140001354
00000001400014E2 CC
00000001400014E3 CC
00000001400014E4 40:53 push rbx
00000001400014E8 48:83EC 20 mov rbx,rcx
00000001400014EA 48:8BD9 xor ecx,ecx
00000001400014ED 33C9
FF15 330B0000 call qword ptr ds:[<&SetUnhandledExceptionFilter>]
00000001400014F5 48:8BC8 mov rcx,rbx
FF15 20B080000 call qword ptr ds:[<&UnhandledExceptionFilter>]
00000001400014FE 48:8BC8 mov rcx,rax
FF15 2C0B0000 call qword ptr ds:[<&GetCurrentProcess>]
0000000140001504 48:8BC8 mov edx,C0000409
BA 090400C0 add esp,20
0000000140001507 48:83C4 20 pop rbp
000000014000150A 5B
0000000140001511 40:53 jmp qword ptr ds:[<&TerminateProcess>]
0000000140001518 48:89C2 08 mov qword ptr ss:[rsp+8],rcx
000000014000151D 48:83EC 38 sub rsp,38
0000000140001520 FF15 4C0B0000 call qword ptr ds:[<&IsProcessorFeaturePresent>]
0000000140001526 85C0 test eax,eax
000000014000152E 74 07 je shellcode-demo.140001537
0000000140001530 B9 02000000 mov ecx,2
0000000140001533 CD 20 int3
0000000140001537 48:8D00 E22B0000 mov rcx,qword ptr ds:[140004120]
000000014000153E E8 A9000000 call shellcode-demo.1400015EC
0000000140001543 48:8942 38 mov rax,qword ptr ss:[rsp+38]
0000000140001548 48:8905 C92C0000 mov qword ptr ds:[140004218],rax
000000014000154F 48:83C0 38 int3
0000000140001554 48:89C0 08 add rax,8
0000000140001558 48:8905 592C0000 mov qword ptr ds:[140004188],rax
0000000140001562 48:8B05 B22C0000 mov rax,qword ptr ds:[140004218]
0000000140001566 48:8905 23B00000 mov qword ptr ds:[140004090],rax
0000000140001570 48:83C0 40 int3
0000000140001572 48:8905 272C0000 mov qword ptr ds:[1400041A0],rax
0000000140001579 C705 FD2A0000 090400C0 mov dword ptr ds:[140004084],1
0000000140001583 C705 F72A0000 01000000 mov dword ptr ds:[140004084],1
000000014000158D C705 012B0000 01000000 mov dword ptr ds:[140004098],1
0000000140001597 48:83C0 08 int3
000000014000159C 48:6BC0 00 imul rax,rax,0
00000001400015A0 48:8D00 F92A0000 lea rcx,qword ptr ds:[1400040A0]
00000001400015A7 C4 70401 02000000 mov qword ptr ds:[rcx+rax],2
00000001400015B0 48:8000 00 mov eax,0
00000001400015B4 48:6BC0 00 imul rax,rax,0
00000001400015B8 48:8B00 412A0000 mov rcx,qword ptr ds:[140004000]
00000001400015BF 48:89C4 20 mov qword ptr ss:[rsp+rax+20],rcx
00000001400015C4 B8 08000000 mov eax,8
00000001400015C9 48:6BC0 01 imul rax,rax,1
00000001400015D0 48:8B00 6C2A0000 mov rcx,qword ptr ds:[140004040]
00000001400015D4 48:89C0 20 mov rax,rcx

```

000E910F2FA78

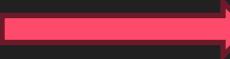
```

RDX R3 00000001400014D0 shellcode-c 48:83EC 28 sub rsp,28
00000001400014D4 E8 C7030000 call shellcode-demo.7FF6BA4D18A0
00000001400014D8 48:83C4 28 add rsp,28
00000001400014D0 E9 72FFFFFF jmp shellcode-demo.7FF6BA4D1354
00000001400014E2 CC
00000001400014E3 CC
00000001400014E4 40:53 push rbx
00000001400014E8 48:83EC 20 mov rbx,rcx
00000001400014EA 48:8BD9 xor ecx,ecx
00000001400014ED 33C9
FF15 330B0000 call qword ptr ds:[<&SetUnhandledExceptionFilter>]
00000001400014F5 48:8BC8 mov rcx,rbx
FF15 20B080000 call qword ptr ds:[<&UnhandledExceptionFilter>]
00000001400014FE 48:8BC8 mov rcx,rax
FF15 2C0B0000 call qword ptr ds:[<&GetCurrentProcess>]
0000000140001504 48:8BC8 mov edx,C0000409
BA 090400C0 add esp,20
0000000140001507 48:83C4 20 pop rbp
000000014000150A 5B
0000000140001511 40:53 jmp qword ptr ds:[<&TerminateProcess>]
0000000140001518 48:89C2 08 mov qword ptr ss:[rsp+8],rcx
000000014000151D 48:83EC 38 sub rsp,38
0000000140001520 FF15 4C0B0000 call qword ptr ds:[<&IsProcessorFeaturePresent>]
0000000140001526 85C0 test eax,eax
000000014000152E 74 07 je shellcode-demo.7FF6BA4D1537
0000000140001530 B9 09000000 mov ecx,2
0000000140001533 CD 20 int3
0000000140001537 48:8D00 E22B0000 mov rcx,qword ptr ds:[7FF6BA4D120]
000000014000153E E8 A9000000 call shellcode-demo.7FF6BA4D15EC
0000000140001543 48:8942 38 mov rax,qword ptr ss:[rsp+38]
0000000140001548 48:8905 C92C0000 mov qword ptr ds:[7FF6BA4D18A0],rax
000000014000154F 48:83C0 38 int3
0000000140001554 48:89C0 08 add rax,8
0000000140001558 48:8905 592C0000 mov qword ptr ds:[7FF6BA4D188],rax
0000000140001562 48:8B05 B22C0000 mov rax,qword ptr ds:[7FF6BA4D18A0]
0000000140001566 48:8905 23B00000 mov qword ptr ds:[7FF6BA4D180],rax
0000000140001570 48:83C0 40 int3
0000000140001572 48:8905 272C0000 mov qword ptr ds:[7FF6BA4D1A0],rax
0000000140001579 C705 FD2A0000 090400C0 mov dword ptr ds:[7FF6BA4D184],1
0000000140001583 C705 F72A0000 01000000 mov dword ptr ds:[7FF6BA4D184],1
000000014000158D C705 012B0000 01000000 mov dword ptr ds:[7FF6BA4D188],1
0000000140001597 48:83C0 08 int3
000000014000159C 48:6BC0 00 imul rax,rax,0
00000001400015A0 48:8D00 F92A0000 lea rcx,qword ptr ds:[7FF6BA4D1A0]
00000001400015A7 C4 70401 02000000 mov qword ptr ds:[rcx+rax],2
00000001400015B0 48:8000 00 mov eax,0
00000001400015B4 48:6BC0 00 imul rax,rax,0
00000001400015B8 48:8B00 412A0000 mov rcx,qword ptr ds:[7FF6BA4D180]
00000001400015BF 48:89C4 20 mov qword ptr ss:[rsp+rax+20],rcx
00000001400015C4 B8 08000000 mov eax,8
00000001400015C9 48:6BC0 01 imul rax,rax,1
00000001400015D0 48:8B00 6C2A0000 mov rcx,qword ptr ds:[7FF6BA4D184]
00000001400015D4 48:89C0 20 mov rax,rcx

```

00061BA4FFAE8

J007FF6BA4D14D0 shellcode-demo.exe:\$14D0 #8D0 <EntryPoint>



Hex	ASCII
31 6A 1000 CC CC CC CC CC CC 48 89 5C 24 10 48 89 74	????????H,\$.H.t
31 6A 1010 24 20 57 48 83 EC 20 48 8B DA 48 89 5D 08	W.H.i.H.U.H.H.P
31 6A 1020 08 40 83 C1 45 48 8B E8 36 08 00 00 40 4C	H.A.I.oE.G..L
31 6A 1030 00 00 48 8B 0D 48 8B 0D 48 8B 0D 48 8B 0D	0.W.H.i.H.U.H.H.P
31 6A 1040 00 00 48 8B 17 33 16 34 26 16 00 48 8B C4 33	...3.G\$..H.\$83
31 6A 1050 07 0F 87 CA 88 54 24 20 28 01 03 C2 48 29 46 38	X..E.T\$oN.AHF8
31 6A 1060 48 88 74 24 48 44 83 C4 20 5F C3 CC CC H.tsHH_A	_A11111
31 6A 1070 CC CC CC CC 48 89 5C 24 08 48 89 74 24 10 48 89	III11H,\$.H.ts.H.
31 6A 1080 FF 24 18 44 0F 87 E1 0F 87 0A 0F 87 0F 87	FF.FF.FF.FF.FF.FF.FF

INT3 breakpoint 'entry breakpoint' at <shellcode-demo.EntryPoint> (00000001400014D0)

Hex	ASCII
31 6A 1000 CC CC CC CC CC CC 48 89 5C 24 10 48 89 74	????????H,\$.H.t
31 6A 1010 24 20 57 48 83 EC 20 48 8B DA 48 89 5D 08	W.H.i.H.U.H.H.P
31 6A 1020 08 40 83 C1 45 48 8B E8 36 08 00 00 40 4C	H.A.I.oE.G..L
31 6A 1030 00 00 48 8B 0D 48 8B 0D 48 8B 0D 48 8B 0D	0.W.H.i.H.U.H.H.P
31 6A 1040 00 00 48 8B 17 33 16 34 26 16 00 48 8B C4 33	...3.G\$..H.\$83
31 6A 1050 D7 0F 87 CA 88 54 24 20 28 01 03 C2 48 29 46 38	X..E.T\$oN.AHF8
31 6A 1060 48 88 74 24 48 44 83 C4 20 5F C3 CC CC H.tsHH_A	_A11111
31 6A 1070 CC CC CC CC 48 89 5C 24 08 48 89 74 24 10 48 89	III11H,\$.H.ts.H.
31 6A 1080 FF 24 18 44 0F 87 E1 0F 87 0A 0F 87 0F 87	FF.FF.FF.FF.FF.FF.FF

INT3 breakpoint 'entry breakpoint' at <shellcode-demo.EntryPoint> (00007FF6BA4D14D0)



ASLR on Windows

- (Usually) Controlled on a per-application basis
 - Based on the DYNAMICBASE flag in the PE header



ASLR on Windows

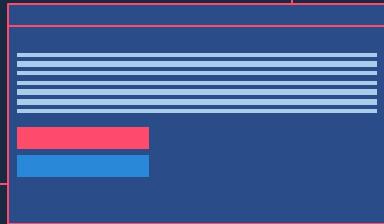
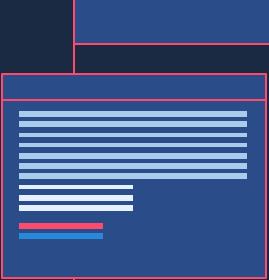
- However, this is not always true:
 - ASLR can be forcibly applied starting from Windows 8 via Mandatory ASLR in Windows Security options (may cause issues for applications that aren't designed to be ASLR aware).
 - ASLR is mandatory for Windows on ARM as the concept of ASLR is always applied on this architecture (allegedly).



Why mention ASLR?

- It's a “good-to-know” knowledge.
- Less confusing when you see address jumping everywhere whenever you debug something after restarting the app/computer.
- You can turn off ASLR in most modern Windows debuggers these days (x64dbg/IDA).





**...and here's where the
abrupt end takes place!**

I've run out of time preparing the materials.

What now?

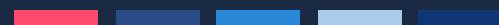
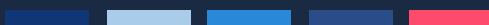
Further Reading

Windows Internals 7th. Ed. / ired.team / vxunderground papers are all great starting points.



Feedback

It's my first time doing this type of stream! Tell me what you like and don't like or mistakes I've made!



Thanks !



azaka.fun



@AzakaSekai_

