



A Arte da Engenharia Reversa



Arquivos Executáveis



Arquivos Executáveis

- Muitos arquivos possuem um formato definido (por exemplo: PNG, MP3, ZIP, etc).
- O formato PE é o formato para arquivos executáveis (.exe), bibliotecas (.dll) e drivers (.sys) do Windows.
- Já no Linux, o formato para executáveis, bibliotecas (*shared objects* / .so) e drivers/módulos de *kernel* (*kernel objects* / .ko) é o ELF.



O que é um formato?

Table of example GIF image values

Byte # (hex)	Hexadecimal	Text or value	Meaning
0	47 49 46 38 39 61	GIF89a	Header
Logical Screen Descriptor			
6	03 00	3	Logical screen width
8	05 00	5	Logical screen height
A	F7		GCT follows for 256 colors with resolution 3 × 8 bits/primary, the lowest 3 bits represent the bit depth minus 1, the highest true bit means that the GCT is present
B	00	0	Background color: index #0; #000000 black
C	00	0	Default pixel aspect ratio, 0:0



HxD - [C:\Users\admin\Downloads\cat.gif]

File Edit Search View Analysis Tools Window Help

cat.gif

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	47	49	46	38	39	61	F2	01	F2	01	F7	00	00	0C	0C	06	GIF89a
00000010	84	7E	64	4C	46	34	7D	49	23	51	1D	1B	C4	BA	A4	7E	„~dLF4)I#Q..Ä°¤~
00000020	62	3C	A4	9E	8B	4D	30	1C	94	63	35	5C	55	4C	A4	80	b<ž<M0.”c5\^L¤€
00000030	5A	34	2A	1C	69	48	2A	6C	30	2B	83	56	2C	7C	72	5B	Z4+IH*10+FV, r[
00000040	2D	11	08	94	8E	74	67	4E	44	BC	A5	8B	4C	3C	2A	99	...”ŽtgND4s<L<*m
00000050	74	4B	E4	D8	D1	A8	8F	6C	94	59	28	84	7E	70	7E	64	tKoÑ..1”Y(.,~p~d
00000060	58	6C	1E	24	2C	1F	13	64	53	39	7C	59	47	3C	3A	2C	X1.\$..dS9 YG<:,
00000070	6C	3D	24	D4	C6	B5	5C	3C	25	7C	48	40	B4	AE	9C	54	1=§Ô€u<% H@’Ø¤T
00000080	52	44	6C	6A	5B	A5	91	84	24	1A	0C	7C	73	67	96	74	RDIj[Y”,\$.. sg-t
00000090	67	94	66	54	BC	AF	93	4C	24	1C	AB	A1	94	3E	2C	1B	g”fT4”L\$.<i”,.
000000A0	97	80	71	5E	32	19	64	55	4D	3C	21	14	6C	5E	43	74	-Eq^2.dUM<!..1^Ct
000000B0	48	29	4C	46	3D	7C	42	1C	7F	6C	4C	F3	E8	E3	1C	0E	H)LF= B..1Löé..

Special editors

Data inspector

Binary (8 bit) Int8 UInt8 Int16 UInt16 Int24 UInt24 Int32 UInt32

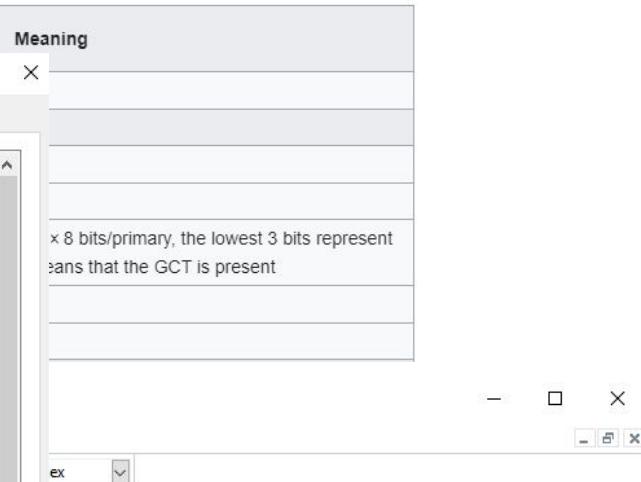
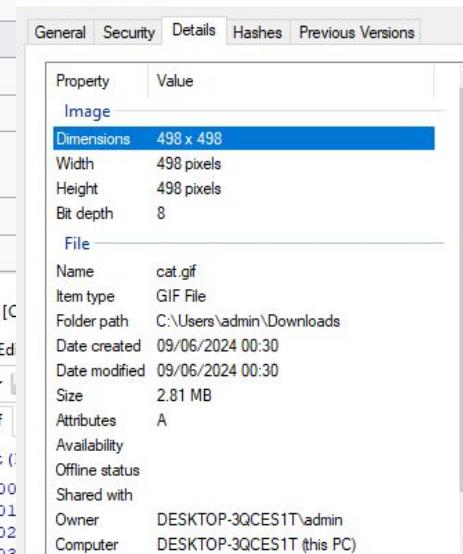
go to: go to:

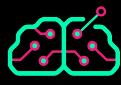
-14 242 498 498 -917006 15860210 32637426 32637426



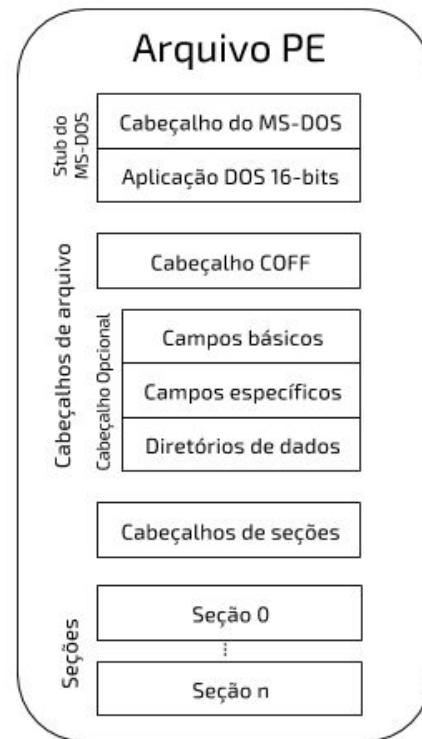
O que é um formato?

Byte # (hex)	Hexadecimal	Text
0	47 49 46 38 39 61	GIF89a
6	03 00	3
8	05 00	5
A	F7	
B	00	0
C	00	0





O Formato PE





Tamanhos dos dados

Microsoft	C (MSVSC)	<stdint.h>	Tamanho em bytes
BYTE	unsigned char	uint8_t	1
WORD	unsigned short	uint16_t	2
DWORD	unsigned long	uint32_t	4
QWORD	__int64	uint64_t	8



Cabeçalhos e Campos

```

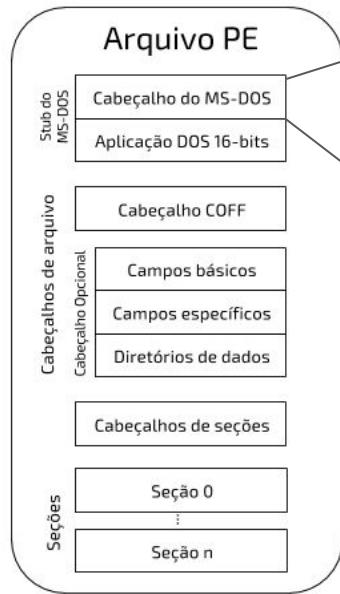
1 typedef struct {
2     uint16_t e_magic;
3     uint16_t e_cblp;
4     uint16_t e_cp;
5     uint16_t e_crlc;
6     uint16_t e_cparhdr;
7     uint16_t e_minalloc;
8     uint16_t e_maxalloc;

```

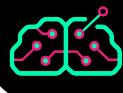
Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	4D	5A	78	00	01	00	00	00	04	00	00	00	00	00	00	00	MZx.....
00000010	00	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00@.....
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000030	00	00	00	00	00	00	00	00	00	00	00	78	00	00	00	00,x..



Cabeçalho do DOS



```
1 typedef struct {
2     uint16_t e_magic;
3     uint16_t e_cblp;
4     uint16_t e_cp;
5     uint16_t e_crlc;
6     uint16_t e_cparhdr;
7     uint16_t e_minalloc;
8     uint16_t e_maxalloc;
9     uint16_t e_ss;
10    uint16_t e_sp;
11    uint16_t e_csum;
12    uint16_t e_ip;
13    uint16_t e_cs;
14    uint16_t e_lfarlc;
15    uint16_t e_ovno;
16    uint16_t e_res[4];
17    uint16_t e_oemid;
18    uint16_t e_oeminfo;
19    uint16_t e_res2[10];
20    uint32_t e_lfanew;
21 } IMAGE_DOS_HEADER;
```



Campos - visualização no HxD

HxD - [C:\Users\admin\Desktop\binarios\strings2.exe]

File Edit Search View Analysis Tools Window Help

strings2.exe e_magic e_lfanew

Offset(h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	Decoded text
00000000	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZ.....VV...
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 80 00 00 00€...
00000040	0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68	..°...!..Lí!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	50 45 00 00 4C 01 03 00 84 89 EC C4 00 00 00 00	PE..L...ñiA....
00000090	00 00 00 00 E0 00 22 00 0B 01 30 00 00 14 00 00à."...0.....
000000A0	00 08 00 00 00 00 00 62 33 00 00 00 20 00 00b3... ..
000000B0	00 40 00 00 00 00 40 00 00 20 00 00 00 02 00 00	.@....@.....
000000C0	04 00 00 00 00 00 00 06 00 00 00 00 00 00 00 00
000000D0	00 80 00 00 00 02 00 00 00 00 00 00 02 00 60 85	.€.....`...
000000E0	00 00 10 00 00 10 00 00 00 00 10 00 00 10 00 00
000000F0	00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00
00000100	0F 33 00 00 4F 00 00 00 00 40 00 00 AC 05 00 00	.3..O....@....
00000110	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000120	00 60 00 00 0C 00 00 00 78 32 00 00 38 00 00 00	.`.....x2..8...
00000130	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000140	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000150	00 00 00 00 00 00 00 00 00 20 00 00 08 00 00 00H...
00000160	00 00 00 00 00 00 00 00 08 20 00 00 48 00 00 00
00000170	00 00 00 00 00 00 00 00 00 2F 74 65 78 74 00 00

Offset(h): 0

Special editors x

Data inspector

Binary (8 bit) 01001101

Int8 go to : 77
UInt8 go to : 77
Int16 go to : 23117
UInt16 go to : 23117
Int24 go to : -7316915
UInt24 go to : 9460301
Int32 go to : 9460301
UInt32 go to : 9460301
Int64 go to : 12894362189
UInt64 go to : 12894362189
LEB128 go to : -51
ULEB128 go to : 77
AnsiChar / char8_ M

Byte order

Little endian Big endian

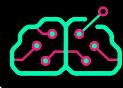
Hexadecimal basis (for integral numbers)

Overwrite

```

1 typedef struct {
2     uint16_t e_magic;
3     uint16_t e_cblp;
4     uint16_t e_cp;
5     uint16_t e_crlc;
6     uint16_t e_cparhdr;
7     uint16_t e_minalloc;
8     uint16_t e_maxalloc;
9     uint16_t e_ss;
10    uint16_t e_sp;
11    uint16_t e_csum;
12    uint16_t e_ip;
13    uint16_t e_cs;
14    uint16_t e_lfarlc;
15    uint16_t e_ovno;
16    uint16_t e_res[4];
17    uint16_t e_oemid;
18    uint16_t e_oeminfo;
19    uint16_t e_res2[10];
20    uint32_t e_lfanew;
21 } IMAGE_DOS_HEADER;

```



Campos importantes do cabeçalho do DOS

- `uint16_t e_magic`
 - 'M' (0x4d) e 'Z' (0x5a), iniciais do Mark Zbikowski, designer do formato de executável do MS-DOS, dentre outras coisas na Microsoft.
- `uint32_t e_lfanew`
 - Offset (no arquivo) da assinatura PE.
 - Assinatura PE: 50 45 00 00 ("PE\0\0")



Campos - visualização no HxD

HxD - [C:\Users\admin\Desktop\binarios\strings2.exe]

File Edit Search View Analysis Tools Window Help

strings2.exe e_magic e_lfanew

Offset(h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	Decoded text
00000000	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZ.....VV..
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 80 00 00 00€...
00000040	0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68	..°...!..Lí!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	50 45 00 00 4C 01 03 00 84 89 EC C4 00 00 00 00	PE..L...%íÄ....
00000090	00 00 00 E0 00 22 00 0B 01 30 00 00 14 00 00à..."...0.....
000000A0	00 08 00 00 00 00 00 62 33 00 00 00 20 00 00b3... ..
000000B0	00 40 00 00 00 00 40 00 00 20 00 00 00 02 00 00	.@....@.....
000000C0	04 00 00 00 00 00 00 06 00 00 00 00 00 00 00 00
000000D0	00 80 00 00 00 02 00 00 00 00 00 00 02 00 60 85	.€.....`..
000000E0	00 00 10 00 00 10 00 00 00 10 00 00 10 00 00 00
000000F0	00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00
00000100	0F 33 00 00 4F 00 00 00 00 40 00 00 AC 05 00 00	.3..0....@..-..
00000110	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000120	00 60 00 00 0C 00 00 00 78 32 00 00 38 00 00 00	.`.....x2..8..
00000130	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000140	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000150	00 00 00 00 00 00 00 00 20 00 00 08 00 00 00 00
00000160	00 00 00 00 00 00 00 00 08 20 00 00 48 00 00 00H...
00000170	00 00 00 00 00 00 00 00 2F 74 F5 78 74 00 00 00text

Offset(h): 0

Overwrite

Special editors x

Data inspector

Binary (8 bit) 01001101

Int8 [go to:](#) 77

Uint8 [go to:](#) 77

Int16 [go to:](#) 23117

Uint16 [go to:](#) 23117

Int24 [go to:](#) -7316915

Uint24 [go to:](#) 9460301

Int32 [go to:](#) 9460301

Uint32 [go to:](#) 9460301

Int64 [go to:](#) 12894362189

Uint64 [go to:](#) 12894362189

LEB128 [go to:](#) -51

ULEB128 [go to:](#) 77

AnsiChar / char8 M

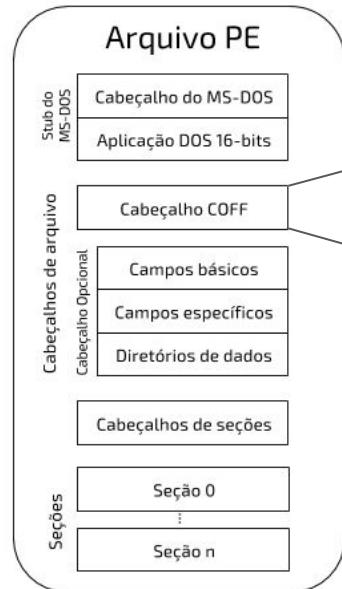
Byte order

Little endian Big endian

Hexadecimal basis (for integral numbers)



Cabeçalho COFF / (PE) File Header



Estrutura de um arquivo PE

```
1 typedef struct {
2     uint16_t Machine;
3     uint16_t NumberOfSections;
4     uint32_t TimeDateStamp;
5     uint32_t PointerToSymbolTable;
6     uint32_t NumberOfSymbols;
7     uint16_t SizeOfOptionalHeader;
8     uint16_t Characteristics;
9 } IMAGE_FILE_HEADER, IMAGE_COFF_HEADER;
```



Campos - visualização no HxD

HxD - [C:\Users\admin\Desktop\binarios\strings2.exe]

File Edit Search View Analysis Tools Window Help

strings2.exe

Offset(h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	Decoded text
00000000	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZ.....ÿÿ..
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 80 00 00 00€...
00000040	0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68	..°..í!..LÍ!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	50 45 00 00 4C 01 03 00 84 89 EC C4 00 00 00 00	PE...ñiÄ....
00000090	00 00 00 00 E0 00 22 00 0B 01 30 00 00 14 00 00à."...0.....
000000A0	00 08 00 00 00 00 00 62 33 00 00 00 20 00 00b3.... ..
000000B0	00 40 00 00 00 00 40 00 00 20 00 00 00 02 00 00	.@....@.. ..
000000C0	04 00 00 00 00 00 00 06 00 00 00 00 00 00 00 00
000000D0	00 80 00 00 00 02 00 00 00 00 00 00 02 00 60 85	.€.....`...
000000E0	00 00 10 00 00 10 00 00 00 10 00 00 10 00 00
000000F0	00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00
00000100	0F 33 00 00 4F 00 00 00 00 40 00 00 AC 05 00 00	.3..O....@..¬...
00000110	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000120	00 60 00 00 0C 00 00 00 78 32 00 00 38 00 00 00	~`.....x2..8...
00000130	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000140	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000150	00 00 00 00 00 00 00 00 20 00 00 08 00 00 00 00
00000160	00 00 00 00 00 00 00 00 08 20 00 00 48 00 00 00H...
00000170	00 00 00 00 00 00 00 00 2F 74 65 70 74 00 00 00�....

Offset(h): 84 Block(h): 84-85 Length(h): 2 Overwrite

Special editors

Data inspector

Binary (8 bit) 01001100

Int8 go to: 76

UInt8 go to: 76

Int16 go to: 332

UInt16 go to: 332

Int24 go to: Invalid

UInt24 go to: Invalid

Int32 go to: Invalid

UInt32 go to: Invalid

Int64 go to: Invalid

UInt64 go to: Invalid

LEB128 go to: -52

ULEB128 go to: 76

AnsiChar / char8_L

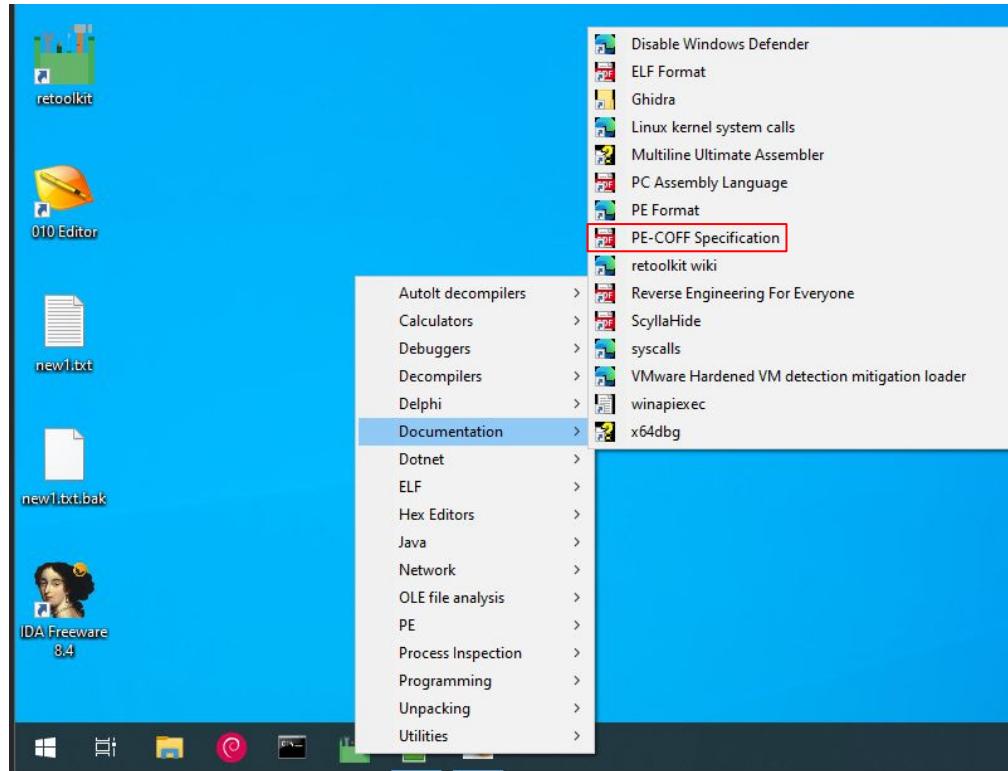
Byte order

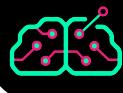
Little endian Big endian

Hexadecimal basis (for integral numbers)



Visualização da documentação do PE-COFF





Visualização da documentação do PE-COFF



Microsoft Portable Executable and Common Object File Format Specification

Revision 11 – June 20, 2017

Abstract

This specification describes the structure of executable (image) files and object files under the Windows® family of operating systems. These files are referred to as Portable Executable (PE) and Common Object File Format (COFF) files, respectively.

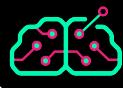
Note

This document is provided to aid in the development of tools and applications for Windows but is not guaranteed to be a complete specification in all respects. Microsoft reserves the right to alter this document without notice.

This revision of the Microsoft Portable Executable and Common Object File Format Specification replaces all previous revisions of this specification.

For the latest information, see:

<http://www.microsoft.com/whdc/system/platform/firmware/PECOFF.mspx>

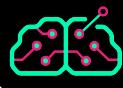


Documentação do PE-COFF - Machine types

3.3.1. Machine Types

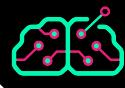
The Machine field has one of the following values that specifies its CPU type. An image file can be run only on the specified machine or on a system that emulates the specified machine.

Constant	Value	Description
IMAGE_FILE_MACHINE_UNKNOWN	0x0	The contents of this field are assumed to be applicable to any machine type
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_EBC	0xebc	EFI byte code
IMAGE_FILE_MACHINE_I386	0x14c	Intel 386 or later processors and compatible processors
IMAGE_FILE_MACHINE_IA64	0x200	Intel Itanium processor family
IMAGE_FILE_MACHINE_M32R	0x9041	Mitsubishi M32R little endian
IMAGE_FILE_MACHINE_MIPS16	0x266	MIPS16
IMAGE_FILE_MACHINE_MIPSFPU	0x366	MIPS with FPU
IMAGE_FILE_MACHINE_MIPSFPU16	0x466	MIPS16 with FPU
IMAGE_FILE_MACHINE_POWERPC	0x1f0	Power PC little endian
IMAGE_FILE_MACHINE_POWERPCFP	0x1f1	Power PC with floating point support
IMAGE_FILE_MACHINE_R4000	0x166	MIPS little endian
IMAGE_FILE_MACHINE_RISCV32	0x5032	RISC-V 32-bit address space



Campos interessantes do cabeçalho COFF

- `uint16_t Machine`
 - Define a arquitetura do binário. Valores comuns incluem 0x14c (32-bits) e 0x8664 (64-bits).
- `uint16_t NumberOfSections`
- `uint32_t TimeDateStamp`
 - Segundos desde 1 de Janeiro de 1970 às 00h (Epoch time), quando o binário foi compilado.



Campos importantes do cabeçalho COFF

- `uint16_t Characteristics`

- Máscara de bits com características do binário.

Valores comuns:

Bit	Nome	Comentários
2	IMAGE_FILE_EXECUTABLE_IMAGE	Obrigatório para arquivos executáveis
9	IMAGE_FILE_32BIT_MACHINE	Arquivo de 32-bits
14	IMAGE_FILE_DLL	O arquivo é uma DLL



Lab 06

Identificando os cabeçalhos do PE



Lab 06 - Identificando os cabeçalhos do PE

1. Abra o programa *strings1.exe* no HxD (ou 010 Editor*) e responda:

a) Qual o offset no arquivo da assinatura PE?

DICA: No offset 0x3c há o campo ***uint32_t e_lfanew***, que contém o offset da assinatura PE.

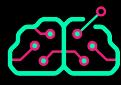
b) Este é um executável de 32 ou 64-bits?

c) Quantas seções este executável possui?

d) Quando ele foi compilado?

* O 010 Editor não está instalado porque a versão de avaliação funciona por 30 dias. Para instalá-lo, basta abrir um Prompt de Comando ou PowerShell e comandar:

```
scoop install 010editor
```



Lab 06 - Respostas

1. a) Qual o offset no arquivo da assinatura PE?

HxD - [C:\Users\admin\Desktop\binarios\strings1.exe]

File Edit Search View Analysis Tools Window Help

strings1.exe

Offset(h)	Decoded text
00000000	MZ.....VV..
00000010@.....
00000020
00000030B...
00000040	...'.Í!,.L!Th
00000050	is program canno
00000060	t be run in DOS
00000070	mode....\$.....
00000080	t°j.ÁÑ.EÁÑ.EÁÑ.E
00000090	É©-EÉÑ.E«.DÁÑ.E
000000A0	«.DÑÑ.E«.DÍÑ.E
000000B0	«.DÁÑ.E.£.DÁÑ.E
000000C0	ÁÑ.EÍÑ.E «.DÁÑ.E
000000D0	;«úÁÑ.E «.DÁÑ.E
000000E0	RichÁÑ.E.....
000000F0	PE..L..æ×Úb....
00000100à.....
00000110-
00000120@.....
00000130
00000140	`.....€.
00000150
00000160
00000170

Special editors

Data inspector

Binary (8 bit) 11110000

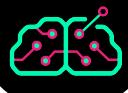
Int8	go to:	-16
UInt8	go to:	240
Int16	go to:	240
UInt16	go to:	240
Int24	go to:	240
UInt24	go to:	240
Int32	go to:	240
UInt32	go to:	240
Int64	go to:	Invalid
UInt64	go to:	Invalid
LEB128	go to:	112
ULEB128	go to:	112
AnsiChar / char8	go to:	Ø

Byte order

Little endian Big endian

Hexadecimal basis (for integral numbers)

Offset(h): 3C Block(h): 3C-3F Length(h): 4 Overwrite



Lab 06 - Respostas

1. b) Este é um executável de 32 ou 64-bits?

HxD - [C:\Users\admin\Desktop\binarios\strings1.exe]

File Edit Search View Analysis Tools Window Help

strings1.exe

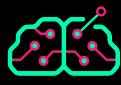
Offset(h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	Decoded text
00000000	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZ.....ÿÿ..
00000010	B8 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00@.....
00000020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000030	00 00 00 00 00 00 00 00 00 00 00 00 F0 00 00 00ñ...
00000040	0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68	..°..í!.LÍ!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 F5 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	86 B0 6A 16 C2 D1 04 45 C2 D1 04 45 C2 D1 04 45	†°j.ÄÅN.EÄN.E
00000090	CB A9 97 45 C8 D1 04 45 A2 AB 05 44 C1 D1 04 45	É@-EÅN.E«.DÄN.E
000000A0	A2 AB 01 44 C2 D1 04 45 A2 AB 00 44 CE D1 04 45	«.DÑN.E«.DÑN.E
000000B0	A2 AB 07 44 C3 D1 04 45 11 A3 05 44 C0 D1 04 45	«.DÄN.E.£.DÄN.E
000000C0	C2 D1 05 45 EC D1 04 45 A6 AB 0D 44 C3 D1 04 45	ÄN.EÄN.E «.DÄN.E
000000D0	A6 AB FB 45 C3 D1 04 45 A6 AB 06 44 C3 D1 04 45	!«.ÜÄN.E «.DÄN.E
000000E0	52 69 63 68 C2 D1 04 45 00 00 00 00 00 00 00 00	RichÄN.E.....
000000F0	S0 45 00 00 1C 01 05 00 9C D7 DA 62 00 00 00 00	PE...æ×Üb.....
00000100	00 00 00 00 E0 00 02 01 0B 01 0E 20 00 0E 00 00à.....
00000110	00 14 00 00 00 00 00 AC 12 00 00 00 10 00 00ñ.....
00000120	00 20 00 00 00 00 40 00 00 10 00 00 00 02 00@.....
00000130	06 00 00 00 00 00 00 06 00 00 00 00 00 00 00 00
00000140	00 60 00 00 00 04 00 00 00 00 00 03 00 40 81	`.....€.
00000150	00 00 10 00 00 10 00 00 00 00 10 00 00 10 00 00
00000160	00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00
00000170	3C 26 00 00 70 00 00 00 40 00 00 F0 01 00 00

Offset(h): F4 Block(h): F4-F5 Length(h): 2

3.3.1. Machine Types

The Machine field has one of the following values that specifies its 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 contents of this field are assumed to be applicable to any machine type
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	0xaax64	ARM64 little endian
IMAGE_FILE_MACHINE_ARMNT	0x1c4	ARM Thumb-2 little endian
IMAGE_FILE_MACHINE_EBC	0xebc	EFI byte code
IMAGE_FILE_MACHINE_I386	0x14c	Intel 386 or later processors and compatible processors
IMAGE_FILE_MACHINE_IA64	0x200	Intel Itanium processor family
IMAGE_FILE_MACHINE_M32R	0x9041	Mitsubishi M32R little endian
IMAGE_FILE_MACHINE_MIPS16	0x266	MIPS16
IMAGE_FILE_MACHINE_MIPSFPU	0x366	MIPS with FPU
IMAGE_FILE_MACHINE_MIPSFPU16	0x466	MIPS16 with FPU
IMAGE_FILE_MACHINE_POWERPC	0x1f0	Power PC little endian
IMAGE_FILE_MACHINE_POWERPCFP	0x1f1	Power PC with floating point support
IMAGE_FILE_MACHINE_R4000	0x166	MIPS little endian
IMAGE_FILE_MACHINE_RISCV32	0x5032	RISC-V 32-bit address space



Lab 06 - Respostas

1. c) Quantas seções este executável possui?

HxD - [C:\Users\admin\Desktop\binarios\strings1.exe]

File Edit Search View Analysis Tools Window Help

strings1.exe

Offset(h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	Decoded text
00000000	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZ.....VV..
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 F0 00 00 00@...
00000040	0E 1F BA EE 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 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	86 B0 6A 16 C2 D1 04 45 C2 D1 04 45 C2 D1 04 45	†@.J.EAN.EAN.E
00000090	CB A9 97 45 C8 D1 04 45 A2 AB 05 44 C1 D1 04 45	É@-EEÑ.E@.DÁÑ.E
000000A0	A2 AB 01 44 D1 04 45 A2 AB 00 44 CE D1 04 45	o@.DNÑ.E@.DÍÑ.E
000000B0	A2 AB 07 44 C3 D1 04 45 11 A3 05 44 C0 D1 04 45	c@.DÁÑ.E.£.DÁÑ.E
000000C0	C2 D1 05 45 EC D1 04 45 A6 AB 0D 44 C3 D1 04 45	ÁÑ.EIÑ.E@.DÁÑ.E
000000D0	A6 AB FB 45 C3 D1 04 45 A6 AB 06 44 C3 D1 04 45	;«ÙEAN.E;«.DÁÑ.E
000000E0	52 69 63 68 C2 D1 04 45 00 00 00 00 00 00 00 00	RichÁÑ.E.....
000000F0	50 45 00 00 4C 01 05 00 9C D7 DA 62 00 00 00 00	PE..L.œ*Úb....
00000100	00 00 00 E0 00 02 01 0B 01 0E 20 00 0E 00 00à.....
00000110	00 14 00 00 00 00 00 AC 12 00 00 00 10 00 00T.....
00000120	00 20 00 00 00 00 40 00 00 10 00 00 00 02 00 00@.....
00000130	06 00 00 00 00 00 00 06 00 00 00 00 00 00 00 00
00000140	00 60 00 00 00 04 00 00 00 00 00 00 03 00 40 81	..`.....@.
00000150	00 00 10 00 00 10 00 00 00 00 10 00 00 10 00 00
00000160	00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00
00000170	3C 26 00 00 70 00 00 00 40 00 F0 01 00 00 00 00

Offset(h): F6 Block(h): F6-F6 Length(h): 1 Overwrite

Special editors

Data inspector

Binary (8 bit) 00000101

Int8 go to: 5

UInt8 go to: 5

Int16 go to: Invalid

UInt16 go to: Invalid

Int24 go to: Invalid

UInt24 go to: Invalid

Int32 go to: Invalid

UInt32 go to: Invalid

Int64 go to: Invalid

UInt64 go to: Invalid

LEB128 go to: 5

ULEB128 go to: 5

AnsiChar / char8_ □

Byte order

Little endian Big endian

Hexadecimal basis (for integral numbers)



Lab 06 - Respostas

1. d) Quando ele foi compilado?

HxD - [C:\Users\admin\Desktop\binarios\strings1.exe]

File Edit Search View Analysis Tools Window Help

strings1.exe

Offset(h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	Decoded text
00000000	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZ.....ÿÿ..
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 F0 00 00 00ñ...
00000040	0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68	..°..`í!.Lí!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	86 B0 6A 16 C2 D1 04 45 C2 D1 04 45 C2 D1 04 45	+°j.ÀÑ.EÀÑ.EÀÑ.E
00000090	CB A9 97 45 C8 D1 04 45 A2 AB 05 44 C1 D1 04 45	È@-EEÑ.E«.DÀÑ.E
000000A0	A2 AB 01 44 D1 D1 04 45 A2 AB 00 44 CE D1 04 45	«.DÑÑ.E«.DÌÑ.E
000000B0	A2 AB 07 44 C3 D1 04 45 11 A3 05 44 CO D1 04 45	«.DÀÑ.E.£.DÀÑ.E
000000C0	C2 D1 05 45 EC D1 04 45 A6 AB 0D 44 C3 D1 04 45	ÀÑ.EiÑ.E;«.DÀÑ.E
000000D0	A6 AB FB 45 C3 D1 04 45 A6 AB 06 44 C3 D1 04 45	;«ùEÀÑ.E;«.DÀÑ.E
000000E0	52 69 63 68 C2 D1 04 45 00 00 00 00 00 00 00 00	RichÀÑ.E.....
000000F0	50 45 00 00 4C 01 05 00 9C D7 DA 62 00 00 00 00	PE.L...þxÜþ.....
00000100	00 00 00 00 E0 00 02 01 0B 01 0E 20 00 0E 00 00à.....
00000110	00 14 00 00 00 00 00 00 AC 12 00 00 00 10 00 00ñ.....
00000120	00 20 00 00 00 00 40 00 00 10 00 00 00 02 00 00@.....
00000130	06 00 00 00 00 00 00 06 00 00 00 00 00 00 00 00
00000140	00 60 00 00 00 04 00 00 00 00 00 00 03 00 40 81	`.....@.....
00000150	00 00 10 00 00 10 00 00 00 00 10 00 00 10 00 00
00000160	00 00 00 10 00 00 00 00 00 00 00 00 00 00 00 00
00000170	3C 26 00 00 30 00 00 00 40 00 00 F0 01 00 00

Offset(h): F8 Block(h): F8-FB Length(h): 4

Overwrite

Special editors

Data inspector

Binary (8 bit) 10011100

Int8 go to: -100

UInt8 go to: 156

Int16 go to: -10340

UInt16 go to: 55196

Int24 go to: -2435172

UInt24 go to: 14342044

Int32 go to: 1658509212

UInt32 go to: 1658509212

Int64 go to: Invalid

UInt64 go to: Invalid

LEB128 go to: -61428836

ULEB128 go to: 207006620

AnsiChar / char go to:

Byte order

Little endian Big endian

Hexadecimal basis (for integral numbers)



Lab 06 - Respostas

1. d) Quando ele foi compilado?

HxD - [C:\Users\admin\Desktop\binarios\strings1.exe]

File Edit Search View Analysis Tools Window Help

16 Windows (ANSI) hex

strings1.exe

Offset(h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	Decoded text
00000000	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZ.....ÿÿ..
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 F0 00 00 008...
00000040	0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68	..°...í!,Lí!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	86 B0 6A 16 C2 D1 04 45 C2 D1 04 45 C2 D1 04 45	+ºj.Áñ.EÁñ.EÁñ.E
00000090	CB A9 97 45 C8 D1 04 45 A2 AB 05 44 C1 D1 04 45	É@-EEÑ.E«.DÁñ.E
000000A0	A2 AB 01 44 D1 D1 04 45 A2 AB 00 44 CE D1 04 45	«.DÑñ.E«.DÍñ.E
000000B0	A2 AB 07 44 C3 D1 04 45 11 A3 05 44 C0 D1 04 45	«.DÁñ.E.£.DÁñ.E
000000C0	C2 D1 05 45 EC D1 04 45 A6 AB 0D 44 C3 D1 04 45	Áñ.EiÑ.E;«.DÁñ.E
000000D0	A6 AB FB 45 C3 D1 04 45 A6 AB 06 44 C3 D1 04 45	;«.DÁñ.E;«.DÁñ.E
000000E0	52 69 63 68 C2 D1 04 45 00 00 00 00 00 00 00 00 00	RichÁñ.E.....
000000F0	50 45 00 00 4C 01 05 00 9C D7 DA 62 00 00 00 00	PE..L...»xÜB.....
00000100	00 00 00 00 E0 00 02 01 0B 01 0E 20 00 0E 00 00à.....
00000110	00 14 00 00 00 00 00 AC 12 00 00 00 10 00 00-.....
00000120	00 20 00 00 00 00 40 00 00 10 00 00 00 02 00 00@.....
00000130	06 00 00 00 00 00 00 06 00 00 00 00 00 00 00 00
00000140	00 60 00 00 00 04 00 00 00 00 00 03 00 40 81	.`.....@.
00000150	00 00 10 00 00 10 00 00 00 00 10 00 00 10 00 00
00000160	00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00
00000170	3C 26 00 00 70 00 00 00 00 40 00 00 F0 01 00 00

Offset(h): F8 Block(h): F8-FB Length(h): 4 Overwrite

Special editors

Data inspector

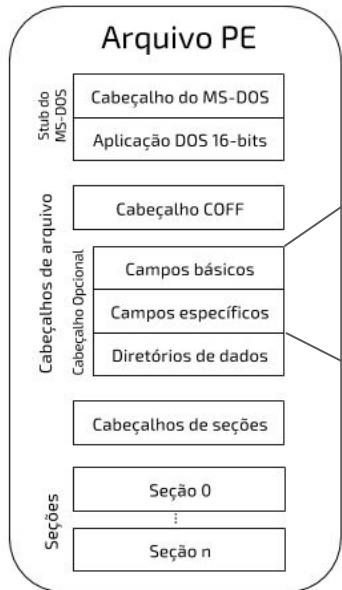
WideChar / char16_t	é
UTF-8 code point	Unexpected continuation
Single (float32)	2.0184632504416E21
Double (float64)	Invalid
OLETIME	Invalid
FILETIME	Invalid
DOS date	28/12/2087
DOS time	Invalid
DOS time & date	Invalid
time_t (32 bit)	22/07/2022 17:00:12
time_t (64 bit)	Invalid
GUID	Invalid
Disassembly (x86-16)	pushfw
Disassembly (x86-32)	pushfd

Byte order
 Little endian Big endian

Hexadecimal basis (for integral numbers)

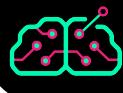


Cabeçalho Opcional



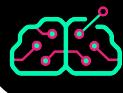
Estrutura de um arquivo PE

```
1 typedef struct {
2     uint16_t Magic;
3     uint8_t MajorLinkerVersion;
4     uint8_t MinorLinkerVersion;
5     uint32_t SizeOfCode;
6     uint32_t SizeOfInitializedData;
7     uint32_t SizeOfUninitializedData;
8     uint32_t AddressOfEntryPoint;
9     uint32_t BaseOfCode;
10    uint32_t BaseOfData;
11    uint32_t ImageBase;
12    uint32_t SectionAlignment;
13    uint32_t FileAlignment;
14    uint16_t MajorOperatingSystemVersion;
15    uint16_t MinorOperatingSystemVersion;
16    uint16_t MajorImageVersion;
17    uint16_t MinorImageVersion;
18    uint16_t MajorSubsystemVersion;
19    uint16_t MinorSubsystemVersion;
20    uint32_t Reserved1;
21    uint32_t SizeOfImage;
22    uint32_t SizeOfHeaders;
23    uint32_t CheckSum;
24    uint16_t Subsystem;
25    uint16_t DllCharacteristics;
26    uint32_t SizeOfStackReserve;
27    uint32_t SizeOfStackCommit;
28    uint32_t SizeOfHeapReserve;
29    uint32_t SizeOfHeapCommit;
30    uint32_t LoaderFlags;
31    uint32_t NumberOfRvaAndSizes;
32    IMAGE_DATA_DIRECTORY DataDirectory[MAX_DIRECTORIES];
33 } IMAGE_OPTIONAL_HEADER_32;
```



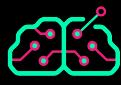
Campos importantes do cabeçalho Opcional

- `uint16_t Magic`
 - Define a arquitetura do binário. Valores comuns incluem 0x10b (32-bits) e 0x20b (64-bits).
- `uint32_t AddressOfEntryPoint`
 - Endereço do *entrypoint* (EP), onde a execução começa.* É opcional para DLLs.



Campos importantes do cabeçalho Opcional

- `uint32_t / uint64_t ImageBase`
 - Endereço virtual do primeiro byte da imagem (cópia do arquivo em memória). Para executáveis em geral, o padrão é 0x400000, com exceção do Windows CE, que usa 0x10000. Já para DLLs, o padrão é 0x10000000.



Prática com 010 Editor

 Repository - Install X

The following template is available in the 010 Editor Repository and can be used to understand binary data from the file 'strings1.exe'. Would you like to install and run this file?

EXE.bt

Parse Windows executable exe, dll, and sys files. Supports 32 and 64 bit.

- Authors: xSpy, Peter Kankowski, SweetScape Software, mirar
- Version: 0.9.4
- Category: Executable
- File Mask: *.exe, *.dll, *.sys
- ID Bytes: 4D 5A //MZ
- Updated: 2023-07-14

[Install](#) [Ignore](#) [Ask me Later](#) [Help](#)



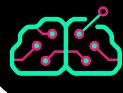
Visualizando PE com 010 Editor

The screenshot shows the 010 Editor application window. The title bar reads "010 Editor - C:\Users\admin\Desktop\binarios\strings1.exe". The menu bar includes File, Edit, Search, View, Format, Scripts, Templates, Debug, Project, Tools, Window, and Help. The toolbar contains various icons for file operations like Open, Save, Find, Replace, and selection tools. The main pane displays the hex dump of the executable file "strings1.exe". The first few bytes are 4D 5A 90 00, followed by the string "MZ.....ÿÿ..". Below the hex dump is the ASCII representation. The right side of the interface features a "Workspace" panel with sections for File, Path, Open Files, Project, Favorite Files, Recent Files, Recent Projects, and Bookmarked Files. At the bottom, there are tabs for Template Results - EXE.bt, Inspector, and Variables. The Inspector tab is currently active, showing a list of data types and their values. The status bar at the bottom indicates "Template executed successfully.", "Pos: 0 [0h]", "Val: 77 4Dh", "Size: 9,216", and buttons for Hex, ANSI, LIT, OVR, and other options.

Name	Type	Value	Start	Size	Color	Comments
> DosHeader	struct IMAGE_DOS_HEADER	0h	40h			
> DosStub	struct IMAGE_DOS_STUB	40h	A8h			
> NtHeader	struct IMAGE_NT_HEADERS	F0h	F8h			
> SectionHeaders[5]	struct IMAGE_SECTION_HEADER	1E8h	C8h			
> Section[0]	struct IMAGE_SECTION_HEADER	400h	E00h			.text
> Section[1]	struct IMAGE_SECTION_HEADER	1200h	C00h			.rdata
> Section[2]	struct IMAGE_SECTION_HEADER	1E00h	200h			.data
> Section[3]	struct IMAGE_SECTION_HEADER	2000h	200h			.rsrc
> Section[4]	struct IMAGE_SECTION_HEADER	2200h	200h			.reloc

Type	Value
Binary	01001101
Signed Byte	77
Unsigned Byte	77
Signed Short	23117
Unsigned Short	23117
Signed Int	9460301
Unsigned Int	9460301
Signed Int64	12894362189
Unsigned Int64	13004262100

Template executed successfully.
Pos: 0 [0h] Val: 77 4Dh Size: 9,216 Hex ANSI LIT OVR ...



Visualizando PE com 010 Editor

010 Editor - C:\Users\admin\Desktop\binarios\strings1.exe

File Edit Search View Format Scripts Templates Debug Project Tools Window Help

Startup strings1.exe X

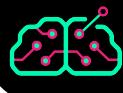
0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF

0000	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZ.....ÿÿ..
0010	B8 00 00 00 00 00 40 00 00 00 00 00 00 00 00 00@.
0020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00ð..
0040	0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68	..º...!í!.Lí!Th
0050	69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6F	is program canno
0060	74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20	t be run in DOS
0070	6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00	mode....\$.....
0080	86 B0 6A 16 C2 D1 04 45 C2 D1 04 45 C2 D1 04 45	ºj.ÁÑ.EÁÑ.EÁÑ.E
0090	CB A9 97 45 C8 D1 04 45 A2 AB 05 44 C1 D1 04 45	É©-ÉÉÑ.EC«.DÁÑ.E
00A0	A2 AB 01 44 D1 D1 04 45 A2 AB 00 44 CE D1 04 45	c«.DÑÑ.EC«.DÍÑ.E
00B0	A2 AB 07 44 C3 D1 04 45 11 A3 05 44 C0 D1 04 45	C«.DÁÑ.E.E.DÁÑ.E
00C0	C2 D1 05 45 FC D1 04 45 A6 AB 0D 44 C3 D1 04 45	ÁÑ.FiÑ.F!«.DÁÑ.E

Template Results - EXE.bt

Name	Value	Start	Size	Type	Color	Com
DosHeader		0h	40h	struct IMAGE_D...		
MZSignature	5A4Dh	0h	2h	WORD		IMAGE_DOS_SIGNAL
UsedBytesInTheLastPage	144	2h	2h	WORD		Bytes on last page
FileSizelnPages	3	4h	2h	WORD		Pages in file
NumberOfRelocationItems	0	6h	2h	WORD		Relocations
HeaderSizelnParagraphs	4	8h	2h	WORD		Size of header in pa
MinimumExtraParagraphs	0	Ah	2h	WORD		Minimum extra pa
MaximumExtraParagraphs	65535	Ch	2h	WORD		Maximum extra pa
InitialRelativeSS	0	Eh	2h	WORD		Initial (relative) SS

Selected: 64 [40h] bytes (Range: 0 [0h] to 63 [3Fh]) Start: 0 [0h] Sel: 64 [40h] Size: 9,216 Hex ANSI LIT OVR



Visualizando PE com 010 Editor

010 Editor - C:\Users\admin\Desktop\binarios\strings1.exe

File Edit Search View Format Scripts Templates Debug Project Tools Window Help

Startup strings1.exe x

0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF

0000 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 MZ.....ÿÿ..

0010 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00@..

0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00ð...

0040 0E 1F BA OE 00 B4 09 CD 21 B8 01 4C CD 21 54 68 ..º...Í!.L!Th

0050 69 73 20 70 72 6F 72 61 6D 20 63 61 6E 6E 6F is program canno

0060 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 t be run in DOS

0070 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00 mode....\$.....

0080 86 B0 6A 16 C2 D1 04 45 C2 D1 04 45 C2 D1 04 45 f°j.ÀÑ.EÀÑ.EÀÑ.E

0090 CB A9 97 45 C8 D1 04 45 A2 AB 05 44 C1 D1 04 45 È@-EÈÑ.EC«.DÀÑ.E

00A0 A2 AB 01 44 D1 D1 04 45 A2 AB 00 44 CE D1 04 45 C«.DÀÑ.EC«.DÌÑ.E

00B0 A2 AB 07 44 C3 D1 04 45 11 A3 05 44 C0 D1 04 45 C«.DÀÑ.E.È.DÀÑ.E

00C0 C2 D1 05 45 EC D1 04 45 46 AB 0D 44 C3 D1 04 45 ÈÑ.FiÑ.FiÑ.DÀÑ.E

Template Results - EXE.bt

Name	Value	Start	Size	Type	Color	Com
AddressOfRelocationTable	64	18h	2h	WORD		File address of relo
OverlayNumber	0	1Ah	2h	WORD		Overlay number
> Reserved[4]		1Ch	8h	WORD		Reserved words
OEMid	0	24h	2h	WORD		OEM identifier (for
OEMinfo	0	26h	2h	WORD		OEM information)
> Reserved2[10]		28h	14h	WORD		Reserved words
AddressOfNewExeHeader	F0h	3Ch	4h	LONG		NtHeader Offset
> DosStub		40h	A8h	struct IMAGE_D...		
> NtHeader		F0h	F8h	struct IMAGE_N...		

Selected: 4 bytes (Range: 60 [3Ch] to 63 [3Fh]) Start: 60 [3Ch] Sel: 4 [4h] Size: 9,216 Hex ANSI LIT OVR

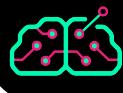
Workspace File Path

- Open Files strings1.exe C:\Users...\binarios\
- Project
- Favorite Files
- Recent Files
- Recent Projects
- Bookmarked Files

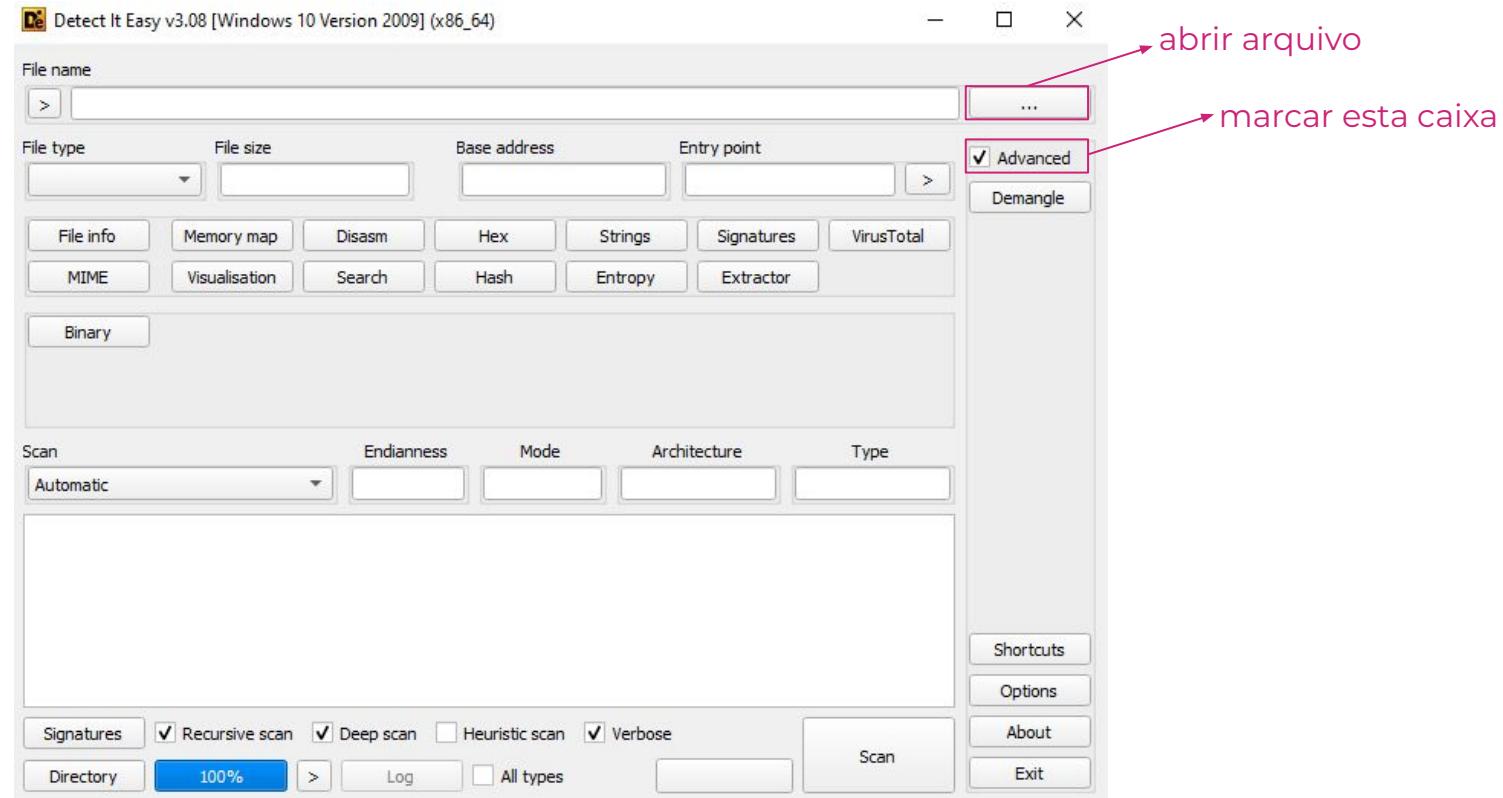
Inspector Type Value

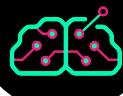
Binary	11110000
Signed Byte	-16
Unsigned Byte	240
Signed Short	240
Unsigned Short	240
Signed Int	240
Unsigned Int	240
Signed Int64	10611948071...

Inspector Variables



Visualizando PE com Detect It Easy (DIE)





Visualizando PE com Detect It Easy (DIE)

Detect It Easy v3.08 [Windows 10 Version 2009] (x86_64)

File name: C:\Users\admin\Desktop\binarios\strings1.exe

File type: PE32 | File size: 9.00 kB | Base address: 00400000 | Entry point: 004012ac

Advanced checkbox is checked.

Buttons: File info, Memory map, Disasm, Hex, Strings, Signatures, VirusTotal, MIME, Visualisation, Search, Hash, Entropy, Extractor.

Tab: PE (highlighted), Export, Import, Resources, .NET, TLS, Overlay.

Sections: 0005 | Time date stamp: 2022-07-22 14:00:12 | Size of image: 00006000 | Resources: Manifest, Version.

Scan: Automatic | Endianness: LE | Mode: 32-bit | Architecture: I386 | Type: Console.

PE32 details:

- Operation system: Windows(Vista)[I386, 32-bit, Console]
- Compiler: EP:Microsoft Visual C/C++ (2017 v.15.5-6)[EXE32]
- Compiler: Microsoft Visual C/C++(19.32.31332)[LTCG/C++]
- Linker: Microsoft Linker(14.32.31332)
- Tool: Visual Studio(2022 version 17.2)

Scan options: Signatures, Recursive scan (checked), Deep scan (checked), Heuristic scan (unchecked), Verbose (checked).

Scan results: 73 msec.

Buttons: Scan, Shortcuts, Options, About, Exit.



Visualizando PE com Detect It Easy (DIE)

PE

Reload < > Hex Disasm Strings Memory map Entropy Heuristic scan Readonly Save

Info

- Nauz File Detector(NFD)
- Detect It Easy(DiE)
- VirusTotal
- Visualization
- Hex
- Disasm
- Hash
- Strings
- Signatures
- Memory map
- Entropy
- Extractor
- Search
- Tools
- IMAGE_DOS_HEADER
- Dos stub
- IMAGE_NT_HEADERS
- IMAGE_FILE_HEADER
- IMAGE_OPTIONAL_HEADER
- IMAGE_DIRECTORY_ENTRIES
- Rich Signature
- Sections
- Info
- Import
- Resources
- Manifest
- Relocs

Name	Offset	Type	Value	
Magic	0000	WORD	010b	NT_HDR32_MAGIC
MajorLinkerVersion	0002	BYTE	0e	
MinorLinkerVersion	0003	BYTE	20	
SizeOfCode	0004	DWORD	00000e00	3.50 KiB
SizeOfInitializedData	0008	DWORD	00001400	5.00 KiB
SizeOfUninitializedData	000c	DWORD	00000000	0 Bytes
AddressOfEntryPoint	0010	DWORD	000012ac	Disasm Section(0)['.text']
BaseOfCode	0014	DWORD	00001000	Hex Section(0)['.text']
BaseOfData	0018	DWORD	00002000	Hex Section(1)['.rdata']
ImageBase	001c	DWORD	00400000	
SectionAlignment	0020	DWORD	00001000	4.00 KiB
FileAlignment	0024	DWORD	00000200	512 Bytes
MajorOperatingSystemVersion	0028	WORD	0006	Windows Vista
MinorOperatingSystemVersion	002a	WORD	0000	
MajorImageVersion	002c	WORD	0000	



Visualizando PE com Detect It Easy (DIE)

Detect It Easy v3.08 [Windows 10 Version 2009] (x86_64)

File name: C:\Users\admin\Desktop\binarios\AnalyseMe-03.exe

File type: PE32 | File size: 4.00 kB | Base address: 00400000 | Entry point: 00401011

Advanced checkbox is checked.

Buttons: File info, Memory map, Disasm, Hex, Strings, Signatures, VirusTotal, MIME, Visualisation, Search, Hash, Entropy, Extractor.

Tab: PE (highlighted), Export, Import, Resources, .NET, TLS, Overlay.

Sections: 0003 | Time date stamp: 2018-12-15 14:45:21 | Size of image: 00004000 | Resources: Manifest, Version.

Scan: Automatic | Endianness: LE | Mode: 32-bit | Architecture: I386 | Type: GUI.

PE32 details:

- Operation system: Windows(Vista)[I386, 32-bit, GUI]
- Compiler: Microsoft Visual C/C++(19.12.25830)[C/C++]
- Linker: Microsoft Linker(5.12.8078)
- Tool: Visual Studio

Scan options: Signatures, Recursive scan (checked), Deep scan (checked), Heuristic scan (unchecked), Verbose (checked).

Scan status: Directory, 100%, Log, All types, 39 msec, Scan button.

Side menu: Shortcuts, Options, About, Exit.



Visualizando PE com Detect It Easy (DIE)

PE

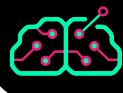
Reload < >

Hex Disasm Strings Memory map Entropy Heuristic scan Readonly Save

Info

- Nauz File Detector(NFD)
- Detect It Easy(DIE)
- VirusTotal
- Visualization
- Hex
- Disasm
- Hash
- Strings
- Signatures
- Memory map
- Entropy
- Extractor
- Search
- Tools
- IMAGE_DOS_HEADER
- Dos stub
- IMAGE_NT_HEADERS
- IMAGE_FILE_HEADER
- IMAGE_OPTIONAL_HEADER
- IMAGE_DIRECTORY_ENTRIES
- Rich Signature
- Sections
- Info
- Import
- Debug

Name	Offset	Type	Value	
Magic	0000	WORD	010b	NT_HDR32_MAGIC
MajorLinkerVersion	0002	BYTE	0e	
MinorLinkerVersion	0003	BYTE	0c	
SizeOfCode	0004	DWORD	00000400	1.00 KiB
SizeOfInitializedData	0008	DWORD	00000800	2.00 KiB
SizeOfUninitializedData	000c	DWORD	00000000	0 Bytes
AddressOfEntryPoint	0010	DWORD	00001011	Disasm Section(0)['.text']
BaseOfCode	0014	DWORD	00001000	Hex Section(0)['.text']
BaseOfData	0018	DWORD	00002000	Hex Section(1)['.rdata']
ImageBase	001c	DWORD	00400000	
SectionAlignment	0020	DWORD	00001000	4.00 KiB
FileAlignment	0024	DWORD	00000200	512 Bytes
MajorOperatingSystemVersion	0028	WORD	0006	Windows Vista
MinorOperatingSystemVersion	002a	WORD	0000	
MajorImageVersion	002c	WORD	0000	



Campos importantes do cabeçalho Opcional

- `uint16_t Subsystem`
 - Subsistema para o qual o binário foi compilado. Os valores mais comuns são 2 (GUI), 3 (CLI) e 1 (Native).
- `uint16_t DllCharacteristics`
 - Máscara de bits com características do binário. Bits importantes incluem o 6 (ASLR) e 8 (DEP/NX).



Strings1.exe no DIE

PE

Reload < >

Hex Disasm Strings Memory map Entropy Heuristic scan Readonly Save

Info

- Nauz File Detector(NFD)
- Detect It Easy(DIE)
- VirusTotal
- Visualization
- Hex
- Disasm
- Hash
- Strings
- Signatures
- Memory map
- Entropy
- Extractor
- Search
- Tools
- IMAGE_DOS_HEADER
- Dos stub
- IMAGE_NT_HEADERS
- IMAGE_FILE_HEADER
- IMAGE_OPTIONAL_HEADER
- IMAGE_DIRECTORY_ENTRIES
- Rich Signature
- Sections
- Info
- Import
- Resources
- Manifest
- Relocs

Name	Offset	Type	Value		
MinorImageVersion	002e	WORD	0000		
MajorSubsystemVersion	0030	WORD	0006		
MinorSubsystemVersion	0032	WORD	0000		
Win32VersionValue	0034	DWORD	00000000		
SizeOfImage	0038	DWORD	00006000		24.00 KiB
SizeOfHeaders	003c	DWORD	00000400		1.00 KiB
CheckSum	0040	DWORD	00000000	Calculate	
Subsystem	0044	WORD	0003	WINDOWS_CUI	
DllCharacteristics	0046	WORD	8140	Flags	
SizeOfStackReserve	0048	DWORD	00100000		1.00 MiB
SizeOfStackCommit	004c	DWORD	00001000		4.00 KiB
SizeOfHeapReserve	0050	DWORD	00100000		1.00 MiB
SizeOfHeapCommit	0054	DWORD	00001000		4.00 KiB
LoaderFlags	0058	DWORD	00000000		
NumberOfRvaAndSizes	005c	DWORD	00000010		



Strings2.exe no DIE

PE

Reload < >

Hex Disasm Strings Memory map Entropy Heuristic scan Readonly Save

Info

- Nauz File Detector(NFD)
- Detect It Easy(DIE)
- VirusTotal
- Visualization
- Hex
- Disasm
- Hash
- Strings
- Signatures
- Memory map
- Entropy
- Extractor
- Search
- Tools
- IMAGE_DOS_HEADER
- Dos stub
- IMAGE_NT_HEADERS
- IMAGE_FILE_HEADER
- IMAGE_OPTIONAL_HEADER
- IMAGE_DIRECTORY_ENTRIES
- Sections
- Info
- Import
- Resources
- Version
- Manifest
- Relocs

Name	Offset	Type	Value		
MinorImageVersion	002e	WORD	0000		
MajorSubsystemVersion	0030	WORD	0006		
MinorSubsystemVersion	0032	WORD	0000		
Win32VersionValue	0034	DWORD	00000000		
SizeOfImage	0038	DWORD	00008000	32.00 KiB	
SizeOfHeaders	003c	DWORD	00000200		512 Bytes
CheckSum	0040	DWORD	00000000	Calculate	
Subsystem	0044	WORD	0002	WINDOWS_GUI	
DllCharacteristics	0046	WORD	8560	Flags	
SizeOfStackReserve	0048	DWORD	00100000	1.00 MiB	
SizeOfStackCommit	004c	DWORD	00001000	4.00 KiB	
SizeOfHeapReserve	0050	DWORD	00100000	1.00 MiB	
SizeOfHeapCommit	0054	DWORD	00001000	4.00 KiB	
LoaderFlags	0058	DWORD	00000000		
NumberOfRvaAndSizes	005c	DWORD	00000010		

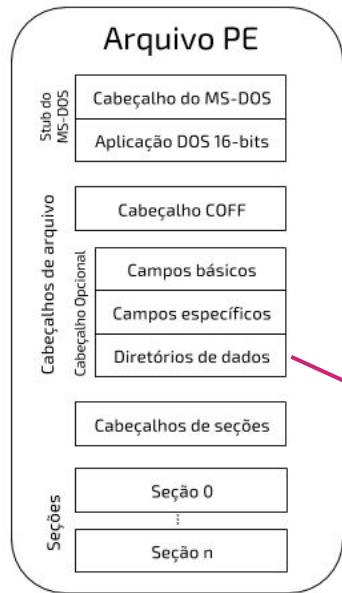


DllCharacteristics

Constante	Valor
IMAGE_DLLCHARACTERISTICS_DYNAMIC_BASE	0x40
IMAGE_DLLCHARACTERISTICS_NX_COMPAT	0x100



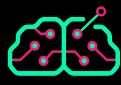
Cabeçalho Opcional Diretório de Dados



```

1 typedef struct {
2     uint16_t Magic;
3     uint8_t MajorLinkerVersion;
4     uint8_t MinorLinkerVersion;
5     uint32_t SizeOfCode;
6     uint32_t SizeOfInitializedData;
7     uint32_t SizeOfUninitializedData;
8     uint32_t AddressOfEntryPoint;
9     uint32_t BaseOfCode;
10    uint32_t BaseOfData;
11    uint32_t ImageBase;
12    uint32_t SectionAlignment;
13    uint32_t FileAlignment;
14    uint16_t MajorOperatingSystemVersion;
15    uint16_t MinorOperatingSystemVersion;
16    uint16_t MajorImageVersion;
17    uint16_t MinorImageVersion;
18    uint16_t MajorSubsystemVersion;
19    uint16_t MinorSubsystemVersion;
20    uint32_t Reserved1;
21    uint32_t SizeOfImage;
22    uint32_t SizeOfHeaders;
23    uint32_t CheckSum;
24    uint16_t Subsystem;
25    uint16_t DllCharacteristics;
26    uint32_t SizeOfStackReserve;
27    uint32_t SizeOfStackCommit;
28    uint32_t SizeOfHeapReserve;
29    uint32_t SizeOfHeapCommit;
30    uint32_t LoaderFlags;
31    uint32_t NumberOfRvaAndSizes;
32    IMAGE_DATA_DIRECTORY DataDirectory[MAX_DIRECTORIES];
33 } IMAGE_OPTIONAL_HEADER_32;

```



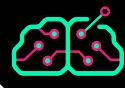
Diretórios de dados

- 16 diretórios ao todo, mas nem todos precisam estar preenchidos.
- Export Table
 - Aponta para a tabela de *exports*, funções exportadas pelo binário.
- Import Table
 - Aponta para a tabela de *imports*, funções importadas (de DLLs) pelo binário.

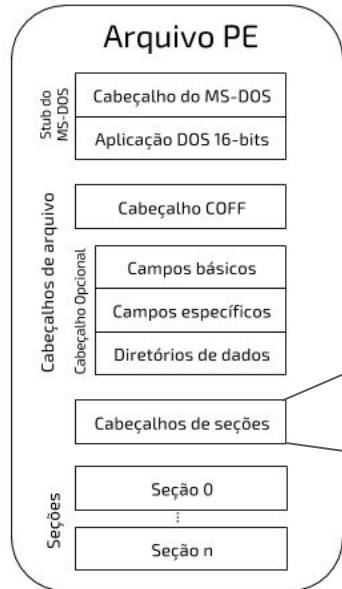


Diretórios de dados

- Resource Table
 - Aponta para uma estrutura de árvore que armazena os *resources* (ícones, janelas, etc) de um binário.

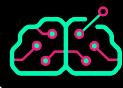


Cabeçalhos de Seções



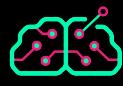
Estrutura de um arquivo PE

```
1 typedef struct {
2     uint8_t Name[SECTION_NAME_SIZE];
3     uint32_t VirtualSize;
4     uint32_t VirtualAddress;
5     uint32_t SizeOfRawData;
6     uint32_t PointerToRawData;
7     uint32_t PointerToRelocations;
8     uint32_t PointerToLinenumbers; // descontinuado
9     uint16_t NumberOfRelocations;
10    uint16_t NumberOfLinenumbers; // descontinuado
11    uint32_t Characteristics;
12 } IMAGE_SECTION_HEADER;
```



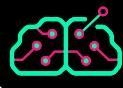
Seções

- São áreas do binário PE que são mapeadas em memória com permissões específicas
- Contém código ou dados
- Uma seção de código, por exemplo, é mapeada em páginas de memória marcadas com permissão de execução



Campos importantes dos cabeçalhos das Seções

- `uint8_t Name[8]`
 - Nome da seção (ex.: .text, .data, etc). Não termina com null.
- `uint32_t VirtualSize`
 - Tamanho em bytes da seção mapeada em memória.
- `uint32_t VirtualAddress`
 - Endereço relativo à base da imagem (ImageBase do cabeçalho opcional) de onde a seção começa em memória.



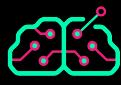
Campos importantes dos cabeçalhos das Seções

- `uint32_t SizeOfRawData`
 - Tamanho em bytes da seção no arquivo.
- `uint32_t PointerToRawData`
 - Offset no arquivo do início da seção.
- `uint32_t Characteristics`
 - Máscara de bits com características da seção.



uint32_t Characteristics

Bit	Nome da flag	Descrição
5	IMAGE_SCN_CNT_CODE	A seção contém código executável
6	IMAGE_SCN_CNT_INITIALIZED_DATA	A seção contém dados inicializados
7	IMAGE_SCN_CNT_UNINITIALIZED_DATA	A seção contém dados não inicializados
29	IMAGE_SCN_MEM_EXECUTE	Terá permissão de execução
30	IMAGE_SCN_MEM_READ	Terá permissão de leitura
31	IMAGE_SCN_MEM_WRITE	Terá permissão de escrita



Principais seções no PE

- .text ou CODE
 - Contém o código do programa. Leitura e execução.
- .data ou DATA
 - Dados inicializados. Leitura e escrita.
- .rdata
 - Dados inicializados. Somente leitura.



Seções no PE - Exemplos

- Caso 1

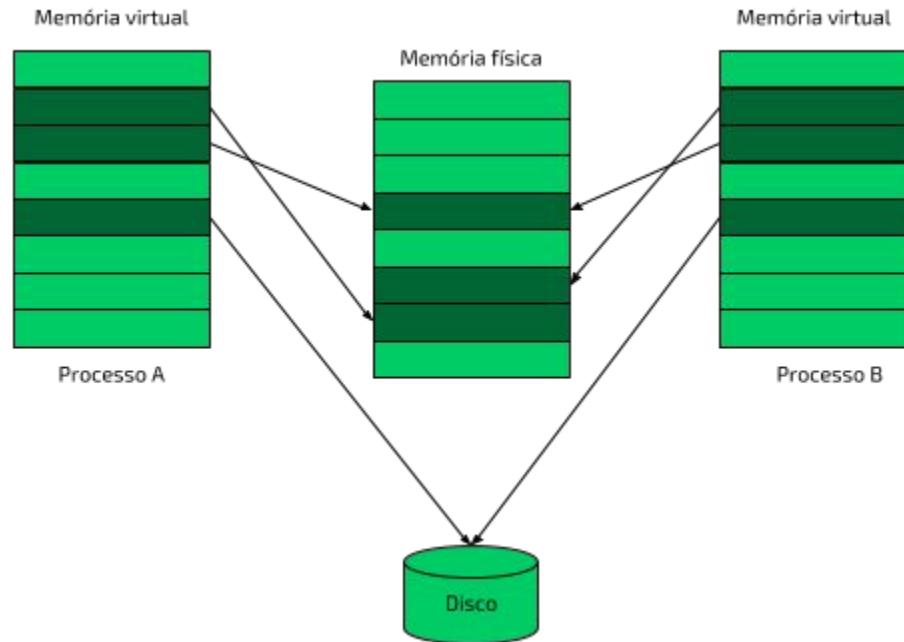
```
char nome[] = "menteb";
// É possível fazer?
nome[0] = 'n';
```

- Caso 2

```
const char *nome = "mente";
// É possível fazer?
nome[0] = 'n';
```



Endereçamento

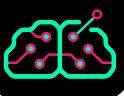




VA x RVA x File Offset

- **Virtual Address (VA)**, é o endereço **virtual** em memória de determinada instrução ou dado
- **Relative Virtual Address (RVA)**, é um valor que, se somado ao VA base, resulta no VA de determinada instrução ou dado
- **File offset** é a posição de determinada instrução ou dado a partir do início do arquivo

Exemplo:
VA: 0x4023A4



RVA para offset em arquivo

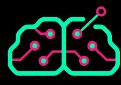
$$S_{RVA} = S \mid S_{VirtualAddress} \leq RVA \leq S_{VirtualAddress} + S_{SizeOfRawData}$$

$$Offset = RVA - S_{RVA_{VirtualAddress}} + S_{RVA_{PointerToRawData}}$$



Lab 07

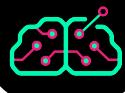
Seções do PE



Lab 07 - Seções do PE

1. Abra o programa *strings7.exe* no DIE e responda:

- a) Quais as características da seção **.text** em memória? Elas fazem sentido?
 - b) Em que seção a string **Papo Binario** está? Quais as características dela em memória? Elas fazem sentido?
-
- retoolkit → PE → Detect It Easy (DIE)
 - Habilitar a caixa "Advanced" pra poder clicar no botão "PE".
 - Sections
 - Strings



Lab 07 - Seções do PE - Respostas

a) Quais as características da seção **.text** em memória? Elas fazem sentido?

PE

Reload < >

Hex Disasm Strings Memory map Entropy Heuristic scan Random

Dump all Save

Info

- Nauz File Detector(NFD)
- Detect It Easy(DiE)
- VirusTotal
- Visualization
- Hex
- Disasm
- Hash
- Strings
- Signatures
- Memory map
- Entropy
- Extractor
- Search
- Tools
- IMAGE_DOS_HEADER
- Dos stub

Sections

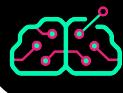
- IMAGE_NT_HEADERS
- IMAGE_FILE_HEADER
- IMAGE_OPTIONAL_HEADER
- IMAGE_DIRECTORY_ENTRIES
- Rich Signature
- Sections
- Info
- Import

Resources

- Manifest
- Relocs

#	Name	VirtualSize	VirtualAddress	SizeOfRawData	PointerToRawData	PointerToRelocations	PointerToLinenumbers	NumberOfRelocations	NumberOfLineNumbers
0	.text	00000ce4	00001000	00000e00	00000400	00000000	00000000	0000	0000
1	.rdata	00000b9e	00002000	00000c00	00001200	00000000	00000000	0000	0000
2	.data	00000384	00003000	00000200	00001e00	00000000	00000000	0000	0000
3	.rsrc	000001e0	00004000	00000200	00002000	00000000	00000000	0000	0000
4	.reloc	00000164	00005000	00000200	00002200	00000000	00000000	0000	0000

Address	00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f	Symbols
0000:1000	55 b8 ec 83 ec 18 56 57 b8 3d b0 20 40 00 33 f6VW.=. @.3.
0000:1010	c7 45 e8 00 21 40 00 c7 45 ec 10 21 40 00 c7 45	.E..!@..E..!@..E
0000:1020	f0 20 21 40 00 c7 45 f4 30 21 40 00 c7 45 f8 5c	. !@..E.0!@..E.\
0000:1030	21 40 00 c7 45 fc 6c 21 40 00 6f 1f 44 00 00	!@..E.1!@..f..D..
0000:1040	ff 74 b5 e8 ff d7 46 83 c4 04 83 fe 06 72 f1 5f	.t....F.....r._
0000:1050	33 c0 5e 8b e5 5d c3 3b 0d 04 30 40 00 75 01 c3	3.^..].;..0@.u..
0000:1060	e9 79 02 00 00 56 6a 01 e8 75 0b 00 00 e8 56 06	.y...Vj..u....V.
0000:1070	00 00 50 e8 a0 0b 00 00 e8 44 06 00 00 8b f0 e8	..P.....D.....
0000:1080	c4 0b 00 00 6a 01 89 30 e8 fa 03 00 00 83 c4 0cj..0.....
0000:1090	5e 84 c0 74 73 db e2 e8 72 08 00 00 68 3a 19 40	^..ts....r...h:@
0000:10a0	00 e8 6e 05 00 00 e8 19 06 00 00 50 e8 3d 0b 00	..n.....P.=..
0000:10b0	00 59 59 85 c0 75 51 e8 12 06 00 00 e8 69 06 00	.YY..uQ.....i..



Lab 07 - Seções do PE - Respostas

a) Quais as características da seção **.text** em memória? Elas fazem sentido?

Section Header

Readonly

Name	Offset	Type	Value
Name	0000	BYTE[8]	.text
VirtualSize	0008	DWORD	00000cce4
VirtualAddress	000c	DWORD	00001000
SizeOfRawData	0010	DWORD	00000e00
PointerToRawData	0014	DWORD	00000400
PointerToRelocations	0018	DWORD	00000000
PointerToLinenumbers	001c	DWORD	00000000
NumberOfRelocations	0020	WORD	0000
NumberOfLinenumbers	0022	WORD	0000
Characteristics	0024	DWORD	60000020

Flags

- TYPE_NO_PAD
- CNT_CODE
- CNT_INITIALIZED_DATA
- CNT_UNINITIALIZED_DATA
- LNK_OTHER
- LNK_INFO
- LNK_REMOVE
- LNK_COMDAT
- NO_DEFER_SPEC_EXC
- GPREL
- MEM_16BIT
- MEM_LOCKED
- MEM_PRELOAD
- LNK_NRELOC_OVFL
- MEM_DISCARDABLE
- MEM_NOT_CACHED
- MEM_NOT_PAGED
- MEM_SHARED
- MEM_EXECUTE
- MEM_READ
- MEM_WRITE

Hex Strings

Address	Hex	Strings
000001e8	00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f	2e 74 65 78 74 00 00 00 e4 00 0e 00 00 00 d4 00 00 00
000001f8	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 20 00 00 60
00000208	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 20 00 00 60



Lab 07 - Seções do PE

b) Em que seção a string **Papo Binario** está? Quais as características dela em memória? Elas fazem sentido?

PE

Reload < >

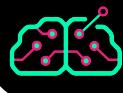
Info Nauz File Detector(NFD)
Detect It Easy(DiE)
VirusTotal
Visualization
Hex
Disasm
Hash
Strings
Signatures
Memory map
Entropy
Extractor
Search
Tools
IMAGE_DOS_HEADER
Dos stub
IMAGE_NT_HEADERS
IMAGE_FILE_HEADER
IMAGE_OPTIONAL_HEADER
IMAGE_DIRECTORY_ENTRIES
Rich Signature
Sections
Info
Import
Resources
Manifest
Relocs

ANSI UTF8 Unicode C Strings 5 Links

Hex Disasm Strings Memory map Entropy Heuristic scan Readonly

Search Save

Offset	Size	Type	String
7	0287	07 A	@.reloc
8	05cb	05 A	9>t@V
9	08af	06 A	u"h@3@
10	0c8a	05 A	f9A@u
11	0ccf	05 A	!t#!=!
12	0cd6	05 A	!t_=!"
13	0e3c	05 A	Sinel
14	0e47	05 A	Sintel
15	1300	0d A	Mente Binaria
16	1310	0c A	Papo Binario
17	1320	0d A	Do Zero Ao Um
18	1330	2a A	Engenharia Reversa - Fundamentos e Pratica
19	135c	0c A	ELFParser-NG
20	1520	05 A	RSDS4
21	1538	33 A	C:\Users\admin\source\repos\teste\Release\teste.pdb



Lab 07 - Seções do PE

b) Em que seção a string **Papo Binario** está? Quais as características dela em memória? Elas fazem sentido?

HxD - [C:\Users\admin\Desktop\binarios\strings1.exe]

File Edit Search View Analysis Tools Window Help

16 Windows (ANSI) hex

strings1.exe

Special editors

Data inspector

Offset(h)	Decoded text
00001270	;)...)Ã)...(.
00001280	(..è(..)ö(..
00001290	Ä(..^(..®(..¢(..
000012A0	€(..^(..D(.......
000012B0	(..^)..Í(..
000012C0	p.Ø.....Ø.
000012D0e.Ø@.Ø.
000012E0
000012F0@.hØ.
00001300	Mente Binaria...
00001310	Papo Binario
00001320	Do Zero Ao Um...
00001330	Engenharía Rever
00001340	sa - Fundamentos
00001350	e Prática..ELFP
00001360	arser-NG....JACK
00001370Ã.....
00001380
00001390
000013A0
000013B0@.Ã™@.....
000013C0	À @.....
000013D0
000013E0

Binary (8 bit) 01010000

Int8	go to: 80
UInt8	go to: 80
Int16	go to: 24912
UInt16	go to: 24912
Int24	go to: 7364944
UInt24	go to: 7364944
Int32	go to: 1869635920
UInt32	go to: 1869635920
Int64	go to: 7955962923802714448
UInt64	go to: 7955962923802714448
LEB128	go to: -48
ULEB128	go to: 80
AnsiChar / char8_t	P

Byte order

Little endian Big endian

Hexadecimal basis (for integral numbers)

Offset(h): 1310 Block(h): 1310-131B Length(h): C Overwrite



Lab 07 - Seções do PE

b) Em que seção a string **Papo Binario** está? Quais as características dela em memória? Elas fazem sentido?

PE

Reload < >

Hex Disasm Strings Memory map Entropy Heuristic scan Readonly Dump all Save

Info

- Nauz File Detector(NFD)
- Detect It Easy(DiE)
- VirusTotal
- Visualization
- Hex
- Disasm
- Hash
- Strings
- Signatures
- Memory map
- Entropy
- Extractor
- Search
- Tools
- IMAGE_DOS_HEADER
- Dos stub
- IMAGE_NT_HEADERS
- IMAGE_FILE_HEADER
- IMAGE_OPTIONAL_HEADER
- IMAGE_DIRECTORY_ENTRIES
- Rich Signature
- Sections
- Info
- Import
- Resources
- Manifest
- Relocs

#	Name	VirtualSize	VirtualAddress	SizeOfRawData	PointerToRawData	PointerToRelocations	PointerToLinenumbers	NumberOfRelocations	NumberOfLineNumbers
0	.text	00000ce4	00001000	00000e00	00000400	00000000	00000000	0000	000
1	.rdata	00000b9e	00002000	00000c00	00001200	00000000	00000000	0000	000
2	.data	00000384	00003000	00000200	00001e00	00000000	00000000	0000	000
3	.rsrc	000001e0	00004000	00000200	00002000	00000000	00000000	0000	000
4	.reloc	00000164	00005000	00000200	00002200	00000000	00000000	0000	000

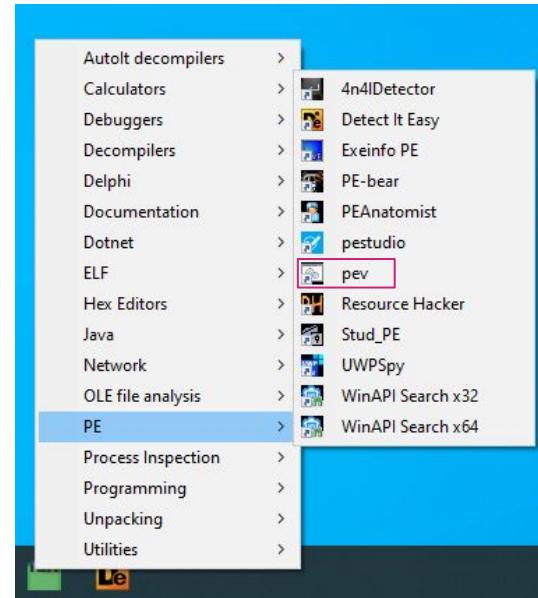
Hex **Strings**

Address	00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f	Symbols
0000:1000	55 8b ec 83 ec 18 56 57 8b 3d b0 20 40 00 33 f6VW.=. @.3.
0000:1010	c7 45 e8 00 21 40 00 c7 45 ec 10 21 40 00 c7 45	.E..!@..E..!@..E
0000:1020	f0 20 21 40 00 c7 45 f4 30 21 40 00 c7 45 f8 5c	. !@..E.0!@..E.\
0000:1030	21 40 00 c7 45 fc 6c 21 40 00 66 0f 1f 44 00 00	!@..E.1!@..f..D..
0000:1040	ff 74 b5 e8 ff d7 46 83 c4 04 83 fe 06 72 f1 5f	.t....F.....r._
0000:1050	33 c0 5e 8b e5 5d c3 3b 0d 04 30 40 00 75 01 c3	3.^..]..;..0@.u..
0000:1060	e9 79 02 00 00 56 6a 01 e8 75 0b 00 00 e8 56 06	.y....Vj...u....V.
0000:1070	00 00 50 e8 a0 0b 00 00 e8 44 06 00 08 b0 e8	..P.....D.....
0000:1080	c4 0b 00 00 6a 01 89 30 e8 fa 03 00 00 83 c4 0cj..0.....
0000:1090	5e 84 c0 74 73 db e2 e8 72 08 00 00 68 3a 19 40	^..ts....r....h:@
0000:10a0	00 e8 6e 05 00 00 e8 19 06 00 00 50 e8 3d 0b 00	..n.....P.=..
0000:10b0	00 59 59 85 c0 75 51 e8 12 06 00 00 e8 69 06 00	.YY..uQ.....i..



Lab 07 - Seções do PE

b) Em que seção a string **Papo Binario** está? Quais as características dela em memória? Elas fazem sentido?



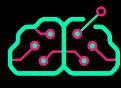


Lab 07 - Seções do PE

b) Em que seção a string **Papo Binario** está? Quais as características dela em memória? Elas fazem sentido?

```
pev
Welcome to pev - the PE file analysis toolkit!
Available tools:
cupload.exe
ofs2rva.exe
pedis.exe
pehash.exe
peldd.exe
pepack.exe
peres.exe
pescan.exe
pesec.exe
pestr.exe
readpe.exe
rva2ofs.exe

C:\Users\admin\AppData\Local\Programs\retoolkit\pe\pev>pestr -s C:\Users\admin\Desktop\binarios\strings1.exe | findstr Papo
.rdata Papo Binario
C:\Users\admin\AppData\Local\Programs\retoolkit\pe\pev>-
```



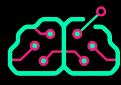
Lab 07 - Seções do PE

Também daria para verificar as características da seção **.text** com readpe

```
pev
C:\Users\admin\AppData\Local\Programs\retoolkit\pe\pev>readpe -S C:\Users\admin\Desktop\binarios\strings1.exe
Sections
  Section
    Name: .text
    Virtual Size: 0xce4 (3300 bytes)
    Virtual Address: 0x1000
    Size Of Raw Data: 0xe00 (3584 bytes)
    Pointer To Raw Data: 0x400
    Number Of Relocations: 0
    Characteristics: 0x60000020
    Characteristic Names
      IMAGE_SCN_CNT_CODE
      IMAGE_SCN_MEM_EXECUTE
      IMAGE_SCN_MEM_READ
  Section
    Name: .rdata
    Virtual Size: 0xb9e (2974 bytes)
    Virtual Address: 0x2000
    Size Of Raw Data: 0xc00 (3072 bytes)
    Pointer To Raw Data: 0x1200
    Number Of Relocations: 0
    Characteristics: 0x40000040
    Characteristic Names
      IMAGE_SCN_CNT_INITIALIZED_DATA
      IMAGE_SCN_MEM_READ
  Section
    Name: .data
    Virtual Size: 0x384 (900 bytes)
    Virtual Address: 0x3000
```



Imports

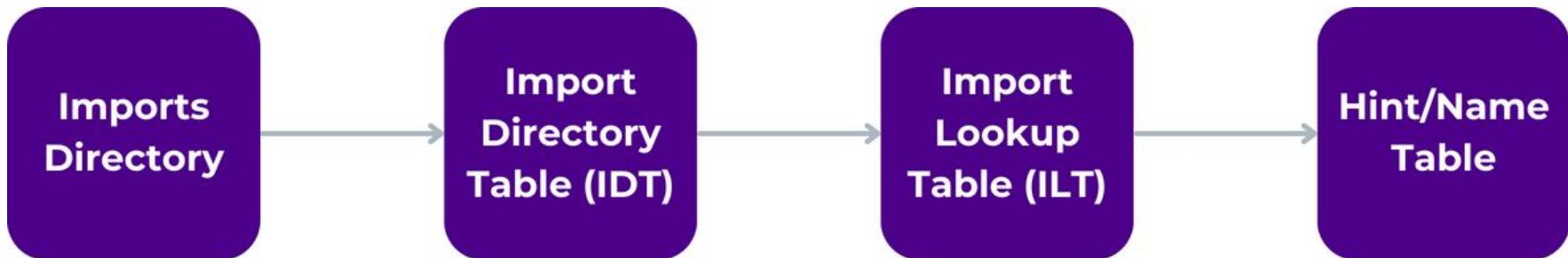


Imports

- São as definições de funções importadas de outros módulos. Por exemplo:
 - kernel32.ExitProcess
 - user32.MessageBox
 - advapi32.RegOpenKeyA



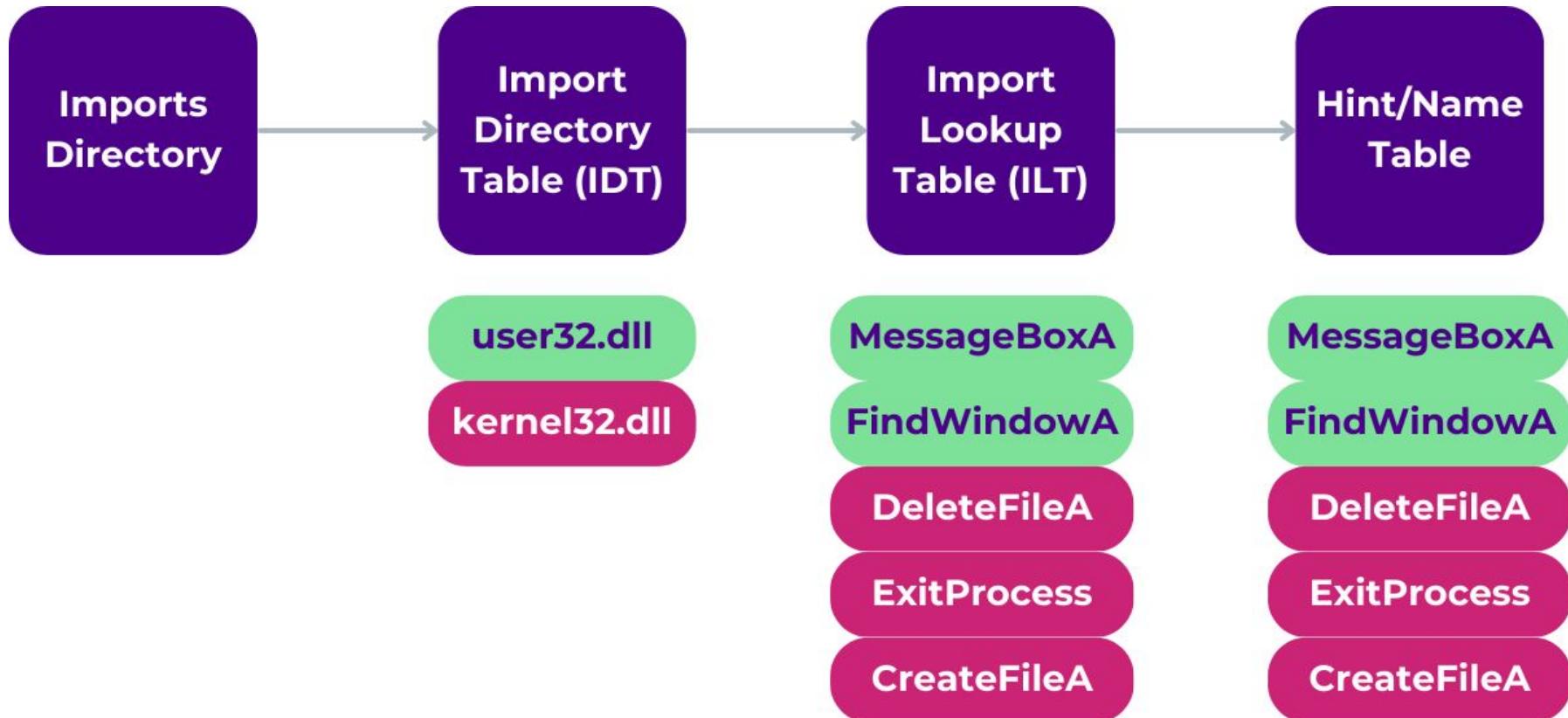
Imports



- Para cada elemento na IDT, o *loader*:
 - Carrega a DLL correspondente.
 - Lê as entradas na ILT (uma por função importada).
 - Lê o nome da função na Hint/Name Table



Imports - Exemplo





Import Directory

- Os 16 elementos do diretório de dados possuem a seguinte estrutura:

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

- O elemento de índice 1 dele é o IMAGE_DIRECTORY_ENTRY_IMPORT, que é o **Import Directory**. O VA dele contém o endereço da **Import Directory Table (IDT)**.



Import Directory Table

- Esta tabela é um array de elementos do seguinte tipo:

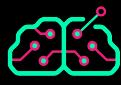
```
1 typedef struct {
2     union {
3         uint32_t Characteristics;
4         uint32_t OriginalFirstThunk; // Endereço da ILT
5     } u1;
6     uint32_t TimeDateStamp;
7     uint32_t ForwarderChain;
8     uint32_t Name; // Endereço do nome da DLL
9     uint32_t FirstThunk; // Endereço da IAT
10 } IMAGE_IMPORT_DESCRIPTOR;
```

RVAs
(mesmo
cálculo)



Import Directory Table (Atualização da MS)

Offset	Size	Field	Description
0	4	Import Lookup Table RVA (Characteristics)	The RVA of the import lookup table. This table contains a name or ordinal for each import. (The name "Characteristics" is used in Winnt.h, but no longer describes this field.)
4	4	Time/Date Stamp	The stamp that is set to zero until the image is bound. After the image is bound, this field is set to the time/data stamp of the DLL.
8	4	Forwarder Chain	The index of the first forwarder reference.
12	4	Name RVA	The address of an ASCII string that contains the name of the DLL. This address is relative to the image base.
16	4	Import Address Table RVA (Thunk Table)	The RVA of the import address table. The contents of this table are identical to the contents of the import lookup table until the image is bound.



Import Lookup Table

- A ILT é um array de números de 32-bits (ou 64-bits para PE32+). O último bit diz se a importação é ordinal (1) ou nome (0). Os outros bits armazenam o endereço da estrutura do nome, chamada de Hint/Name Table:

```
typedef struct _IMAGE_IMPORT_BY_NAME {  
    WORD    Hint;  
    CHAR    Name[1];  
} IMAGE_IMPORT_BY_NAME, *PIMAGE_IMPORT_BY_NAME;
```



Import Address Table

- Finalmente, a IAT é igual a ILT, até que o binário seja carregado pelo *loader*.
- Após o carregamento, a IAT contém os endereços, em 32-bits, das funções importadas.



Vendo imports no DIE

Detect It Easy v3.08 [Windows 10 Version 2009] (x86_64)

File name: C:\Users\admin\Desktop\binarios\strings1.exe

File type: PE32 | File size: 9.00 KIB | Base address: 00400000 | Entry point: 004012ac

Advanced checkbox is checked.

Buttons: File info, Memory map, Disasm, Hex, Strings, Signatures, VirusTotal, MIME, Visualisation, Search, Hash, Entropy, Extractor.

Tab: Import is selected.

Sections: 0005 | Time date stamp: 2022-07-22 14:00:12 | Size of image: 00006000 | Resources: Manifest, Version.

Scan: Automatic | Endianness: LE | Mode: 32-bit | Architecture: I386 | Type: Console.

PE32 section details:

- Operation system: Windows(Vista)[I386, 32-bit, Console]
- Compiler: EP:Microsoft Visual C/C++ (2017 v.15.5-6)[EXE32]
- Compiler: Microsoft Visual C/C++ (19.32.31332)[LTCG/C++]
- Linker: Microsoft Linker(14.32.31332)
- Tool: Visual Studio(2022 version 17.2)

Scan options: Signatures, Recursive scan (checked), Deep scan (checked), Heuristic scan (unchecked), Verbose (checked).

Scan button: Scan (70 msec).

Buttons: Shortcuts, Options, About, Exit.



Vendo imports no DIE

PE

Reload < >

Info

- Nauz File Detector(NFD)
- Detect It Easy(DIE)
- VirusTotal
- Visualization
- Hex
- Disasm
- Hash
- Strings
- Signatures
- Memory map
- Entropy
- Extractor
- Search
- Tools
- IMAGE_DOS_HEADER
- Dos stub

IMAGE_NT_HEADERS

- IMAGE_FILE_HEADER
- IMAGE_OPTIONAL_HEADER
- IMAGE_DIRECTORY_ENTRIES
- Rich Signature

Sections

- Info

Import

Resources

- Manifest
- Relocs

Hash 64 Hash 32

#	OriginalFirstThunk	TimeDateStamp	ForwarderChain	Name	FirstThunk	Hash	Name
0	00002710	00000000	00000000	000027f4	00002034	23ee5677	VCRUNTIME140.dll
1	0000278c	00000000	00000000	000029d0	000020b0	3da4841e	api-ms-win-crt-studio-l1-1-0.dll
2	0000273c	00000000	00000000	000029f0	00002060	62e42675	api-ms-win-crt-runtime-l1-1-0.dll
3	00002734	00000000	00000000	00002a12	00002058	e5294772	api-ms-win-crt-math-l1-1-0.dll
4	0000272c	00000000	00000000	00002a32	00002050	f5d53e1b	api-ms-win-crt-locale-l1-1-0.dll
5	00002724	00000000	00000000	00002a54	00002048	0a54f811	api-ms-win-crt-heap-l1-1-0.dll
6	000026dc	00000000	00000000	00002b90	00002000	46273d4d	KERNEL32.dll

Save

#	Thunk	Ordinal	Hint	Name
0	00002b38		02fa	GetSystemTimeAsFileTime
1	00002a90		0587	SetUnhandledExceptionFilter
2	00002aae		0224	GetCurrentProcess
3	00002ac2		05a6	TerminateProcess
4	00002ad6		039b	IsProcessorFeaturePresent
5	00002b7c		0286	GetModuleHandleW
6	00002b68		0394	IsDebuggerPresent
7	00002b52		0270	InitializeCriticalSection

Save



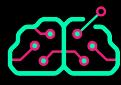
Lab 08

Observando a IAT



Lab 08 - Observando a IAT

1. No DIE, abra o arquivo **AnalyzeMe-04.exe**, marque a caixa **Advanced** e clique em **PE**.
2. Vá até **IMAGE_NT_HEADERS** → **IMAGE_OPTIONAL_HEADER** → **IMAGE_DIRECTORY_ENTRIES** e navegue até o diretório **12 - IAT** e memorize o valor em **Address** dele, que na verdade é um RVA.
3. Agora clique em **Sections** no menu da esquerda e observe a qual seção este RVA pertence.
4. Calcule o VA a partir do RVA e volte para a **IAT** em **IMAGE_DIRECTORY_ENTRIES**. Você vai precisar deste VA no passo seguinte.
5. Feche o DIE. Abra o binário no x32dbg. Clique em qualquer byte na aba **Dump 1** para selecioná-lo e aperte **Ctrl+G**. Digite o VA calculado no passo 4 e pressione **ENTER**.



Lab 08 - Observando a IAT

Agora observando no dump, responda às seguintes perguntas:

1. Por que os valores mudaram na imagem?
2. Quantas funções este binário importa?
3. EXTRA: Clique com o botão direito no dump e escolha a opção **Address** para visualizar endereços*. Há alguma diferença entre as funções na IAT e na ILT (veja **Imports** no DIE)? Pesquise o motivo.

* Para voltar a visualizar os bytes em hexa com o dump em ASCII, clique novamente com o botão direito em qualquer byte no dump e escolha a opção **Hex → ASCII**.



Lab 08 - Observando a IAT - Respostas

The screenshot shows the Immunity Debugger's memory dump view. A pink arrow points from the label "RVA" to the "Address" column header of the table below. The table lists the following sections:

	Section	Address	Size	Description
10	COM_DESCRIPTOR	00000000	00000000	0 Bytes
11	BOUND_IMPORT	00000000	00000000	0 Bytes
12	IAT	00000200	00000050	80 Bytes Section(1)['.rdata']
13	DELAY_IMPORT	00000000	00000000	0 Bytes
14	COM_DESCRIPTOR	00000000	00000000	0 Bytes
15	RESERVED	00000000	00000000	0 Bytes



Lab 08 - Observando a IAT - Respostas

PE

Reload < >

Hex Disasm Strings Memory map Entropy Heuristic scan Readonly

Dump all Save

Name VirtualSize VirtualAddress SizeOfRawData PointerToRawData PointerToRelocations PointerToLinenumbers NumberOfRelocations NumberOfLineNumbers

0 .text	00000466	00001000	00000600	00000400	00000000	00000000	0000	0000
1 .rdata	000004b0	00002000	00000600	00000a00	00000000	00000000	0000	0000
2 .data	00000018	00003000	00000200	00001000	00000000	00000000	0000	0000

Hex Strings

Address	00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f	Symbols
0000:2000	1c 24 00 00 90 24 00 00 84 24 00 00 72 24 00 00	\$...\$...\$...r\$...
0000:2010	e6 23 00 00 f2 23 00 00 fe 23 00 00 0e 24 00 00	.#...#...#...\$...
0000:2020	4c 24 00 00 2a 24 00 00 38 24 00 00 60 24 00 00	L\$...*\$...8\$...`\$...
0000:2030	00 00 00 00 c6 23 00 00 b0 23 00 00 a0 23 00 00#...#...#...
0000:2040	8c 23 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.#.....
0000:2050	01 02 03 04 00 00 00 00 42 38 5f 54 72 6d 64 76B8_Trmvd
0000:2060	61 6f 23 42 69 6e 66 77 5c 4d 71 65 63 6e 66 58	ao#Binfw\MqeencfX
0000:2070	56 6b 71 70 75 63 6f 46 6f 7a 23 43 75 67 70 70	VkqpucoFoz#Cugpp
0000:2080	20 43 67 60 69 76 6a 6b 6e 71 5f 40 49 44 7b 45	Cg`ivjknq_@ID{E
0000:2090	50 4b 2d 60 6c 6e 00 00 42 38 5f 73 69 6c 67 6b	PK-\ln..B8_silgk
0000:20a0	77 71 5f 57 79 71 77 61 6d 31 31 58 44 70 6a 72	wq_Wyqwam11XDpj
0000:20b0	65 70 70 58 56 6f 6e 6b 75 71 66 2a 73 7b 70 00	eppXVonkuqf*s{p.



Lab 08 - Observando a IAT - Respostas

PE

Reload < >

Hex Disasm Strings Memory map Entropy Heuristic scan Readonly Save

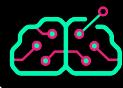
Name	Offset	Type	Value		
Magic	0000	WORD	010b	NT_HDR32_MAGIC	
MajorLinkerVersion	0002	BYTE	0e		
MinorLinkerVersion	0003	BYTE	10		
SizeOfCode	0004	DWORD	00000600		1.50 KiB
SizeOfInitializedData	0008	DWORD	00000800		2.00 KiB
SizeOfUninitializedData	000c	DWORD	00000000		0 Bytes
AddressOfEntryPoint	0010	DWORD	00001001	Disasm	Section(0)['.text']
BaseOfCode	0014	DWORD	00001000	Hex	Section(0)['.text']
BaseOfData	0018	DWORD	00002000	Hex	Section(1)['.rdata']
ImageBase	001c	DWORD	00400000		
SectionAlignment	0020	DWORD	00001000		4.00 KiB
FileAlignment	0024	DWORD	00000200		512 Bytes
MajorOperatingSystemVersion	0028	WORD	0006	Windows Vista	
MinorOperatingSystemVersion	002a	WORD	0000		
MajorImageVersion	002c	WORD	0000		



Lab 08 - Observando a IAT - Respostas

The screenshot shows the Immunity Debugger interface with the following details:

- Registers:** CPU pane showing registers EIP, ECX, EDX, ESI, EDI, EBX, ECX, EDX, EBP, ESP, ESI, EDI.
- Stack:** Stack pane showing the stack dump starting at address 402000, with the expression `analyseme-04.CloseHandle` highlighted.
- Memory:** Memory pane showing memory dump starting at address 777A1000, with the expression `analyseme-04.CloseHandle` highlighted.
- Registers pane:** Shows assembly code for the `OptionalHeader.AddressOfEntryPoint` function, including instructions like `push ebp`, `mov esp, ebp`, and `call dword ptr ds:[<GetCommandLine>]`.
- Registers pane (continued):** Shows the continuation of the assembly code, including `push esi`, `mov al, [esi]`, and `jmp analyseme-04.401038`.
- Registers pane (final part):** Shows the final part of the assembly code, including `inc esi`, `jmp analyseme-04.401038`, and `jmp analyseme-04.401037`.
- Registers pane (bottom):** Shows the assembly code for the `analyseme-04.40103E` function, including `inc esi`, `jmp analyseme-04.40103E`, and `jmp analyseme-04.401037`.
- Registers pane (bottom-most):** Shows the assembly code for the `analyseme-04.40106E` function, including `pop ecx`, `pop pop ecx`, and `je analyseme-04.40106E`.
- Registers pane (right side):** Shows the assembly code for the `analyseme-04.401001` function, including `mov eax, ebx` and `push eax`.
- Registers pane (far right):** Shows assembly code for the `kernel32.BaseThreadInitThunk` function, including `return to kernel32.BaseThreadInitThunk+19` from `???`.

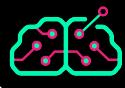


Lab 08 - Observando a IAT - Respostas

Endereços de funções



Address	Hex	ASCII
00402000	80 35 9F 75	80 F9 9E 75
00402010	50 3C 9F 75	60 8A 9E 75
00402020	00 11 9F 75	40 59 9F 75
00402030	00 00 00 00	E0 B1 50 74
00402040	20 9C 5B 74	00 00 00 00
00402050	01 02 03 04	00 00 00 00
00402060	61 6F 23 42	69 6E 66 77
00402070	56 6B 71 70	75 63 6F 46
00402080	20 43 67 60	69 76 6A 6B
00402090	50 4B 2D 60	6C 6E 00 00
004020A0	77 71 5F 57	79 71 77 61
004020B0	65 70 70 58	56 6F 6E 6B
004020C0	4C 6D 79 6D	6C 6E 62 2B
004020D0	70 63 77 6D	62 6E 66 3F
004020E0	30 39 23 53	69 6C 67 6B
004020F0	31 39 23 57	77 71 23 4A
		54 22 36 2A
		6C 6D 62 60
		19#WV3*.]rb}lmb`



Lab 08 - Observando a IAT - Respostas

Endereços de funções

Address	Hex	ASCII			
00402000	80 35 9F 75	80 F9 9E 75	C0 5D 7E 77	80 25 9F 75	.5.u°.uA]~w.%.
00402010	50 3C 9F 75	60 8A 9E 75	B0 3A 9F 75	D0 37 9F 75	P<.u'..u°:.uD7.u
00402020	00 11 9F 75	40 59 9F 75	70 27 9F 75	20 13 9F 75	...u@Y.up'.u ..u
00402030	00 00 00 00	E0 B1 50 74	E0 16 55 74	50 64 4E 74±Pta.UtPdNt
00402040	20 9C 5B 74	00 00 00 00	00 00 00 00	00 00 00 00	.[t.....
00402050	01 02 03 04	00 00 00 00	42 38 5F 54	72 6D 64 76 B8_Trmdv
00402060	61 6F 23 42	69 6E 66 77	5C 4D 71 65	63 6E 66 58	ao#Binfw\MqecnfX
00402070	56 6B 71 70	75 63 6F 46	6F 7A 23 43	75 67 70 70	VkqpucoFoz#Cugpp
00402080	20 43 67 60	69 76 6A 6B	6E 71 5F 40	49 44 78 45	Cg_ivjknq_@ID{E
00402090	50 4B 2D 60	6C 6E 00 00			
004020A0	77 71 5F 57	79 71 77 61	+ Enter expression to follow...		
004020B0	65 70 70 58	56 6F 6E 6B			
004020C0	4C 6D 79 6D	6C 6E 62 2B			
004020D0	70 63 77 6D	62 6E 66 3F			
004020E0	30 39 23 53	69 6C 67 6B			
004020F0	31 39 23 57	56 33 2A 00			
00402100	55 55 55 55	55 55 55 55			

759F3580

Correct expression! -> kernel32.CloseHandle

OK Cancel



Lab 08 - Observando a IAT - Respostas

AnalyseMe-04.exe - PID: 3540 - Module: kernel32.dll - Thread: Main Thread 6656 - x32dbg

File View Debug Tracing Plugins Favourites Options Help Oct 28 2023 (TitanEngine)

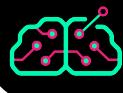
CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols

●	759F3580	- FF25 3411A575	jmp dword ptr ds:[<CloseHandle>]	CloseHandle
●	759F3586	CC	int3	
●	759F3587	CC	int3	
●	759F3588	CC	int3	
●	759F3589	CC	int3	
●	759F358A	CC	int3	



Lab 08 - Observando a IAT - Respostas

The screenshot shows the Immunity Debugger interface. The assembly pane on the right displays assembly code for the `kernel32.CloseHandle` function, specifically the `jmp dword ptr ds:[<GetHandle>]` instruction at address 0x401000. The memory dump pane at the bottom shows the memory state at address 0x401000, where the value is 0x50 3C 9F 75. The registers pane shows CPU registers like EIP, ECX, and ESP. The stack pane shows the current stack contents.



Lab 08 - Observando a IAT - Respostas

Address	Value	ASCII	Comments
00402000	759F3580	.5.u	kernel32.CloseHandle
00402004	759EF980	°u.u	kernel32.GetProcessHeap
00402008	777E5DC0	À]~W	ntdll.RtlAllocateHeap
0040200C	759F2580	.%.u	kernel32.GetCommandLineA
00402010	759F3C50	P<.u	kernel32.WriteFile
00402014	759E8A60	`..u	kernel32.lstrcat
00402018	759F3AB0	°:..u	kernel32.GetTempPathA
0040201C	759F37D0	D7.u	kernel32.CreateFileA
00402020	759F1100	...u	kernel32.GetModuleHandleA
00402024	759F5940	@Y.u	kernel32.ExitProcess
00402028	759F2770	p'.u	kernel32.IsDebuggerPresent
0040202C	759F1320	...u	kernel32.GetStartupInfoA
00402030	00000000	
00402034	7450B1E0	à±Pt	wininet.InternetReadFile
00402038	745516E0	à.Ut	wininet.InternetCloseHandle
0040203C	744E6450	PdNt	wininet.InternetOpenA
00402040	74550000	à+	wininet.InternetOpenUrlA

Command: Commands are comma separated (like assembly instructions): mov eax, ebx

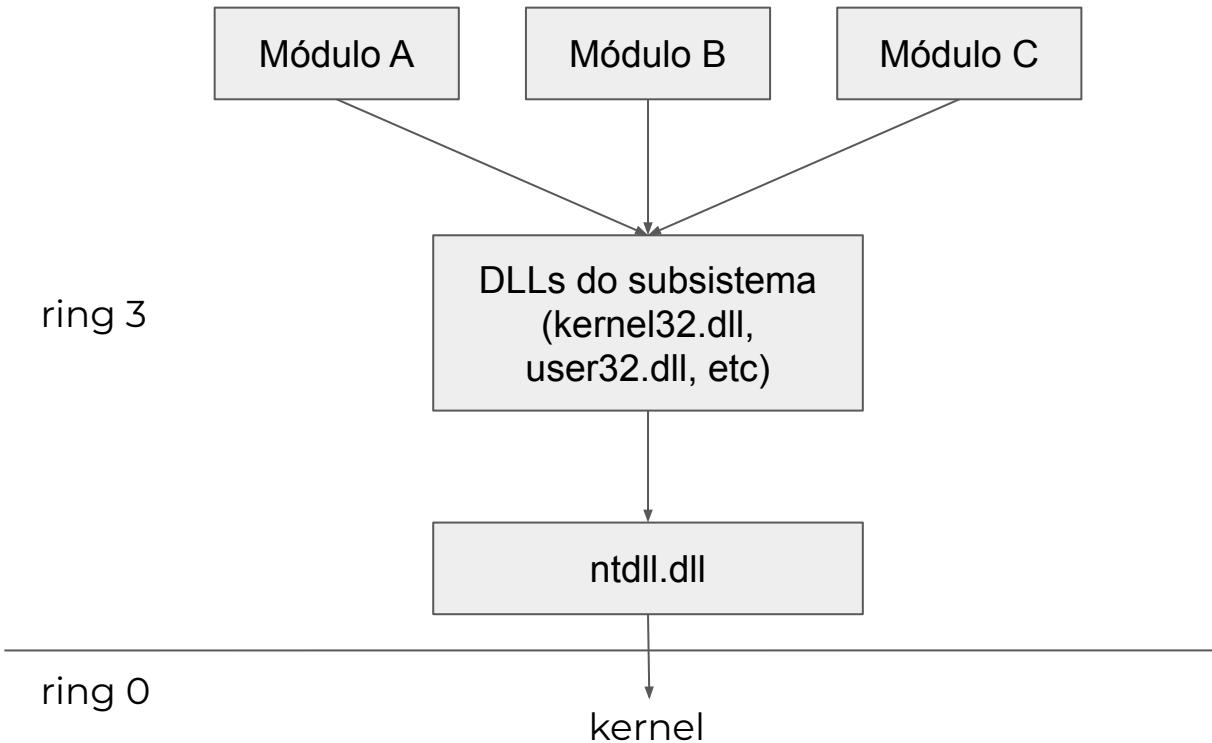
Paused analyseme-04.exe: 00402000 -> 00402003 (0x00000004 bytes)



Execução de Programas

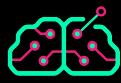


Execução de Programas





Windows API



MessageBox (USER32.DLL)

```
int MessageBox(  
    [in, optional] HWND      hWnd,  
    [in, optional] LPCTSTR   lpText,  
    [in, optional] LPCTSTR   lpCaption,  
    [in]          UINT      uType  
);
```

```
#include <windows.h>  
  
int main(void) {  
    //MessageBox(0, ...)  
    //MessageBox(nullptr, ...)  
    MessageBox(NULL, "Cash", "Johnny", MB_OK);  
}
```



Linux C Library (libc)

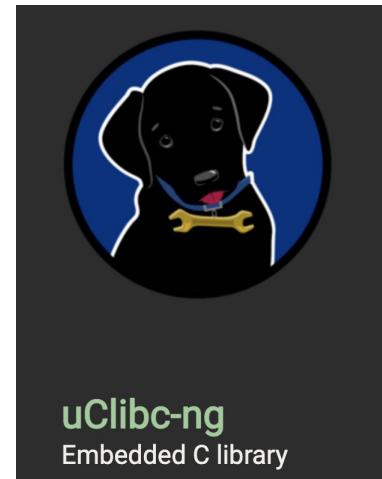


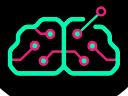
Funções da libc

- printf, malloc, strstr, strlen, etc.

The GNU C Library (glibc)

musl libc





Linux libc wrappers



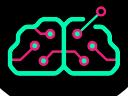
write

```
#include <unistd.h>

ssize_t write(int fd, const void *buf, size_t count);
```

```
#include <unistd.h>

int main(void) {
    char *msg = "oi\n";
    write(1, msg, 4);
}
```



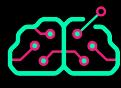
Introdução à Assembly x64



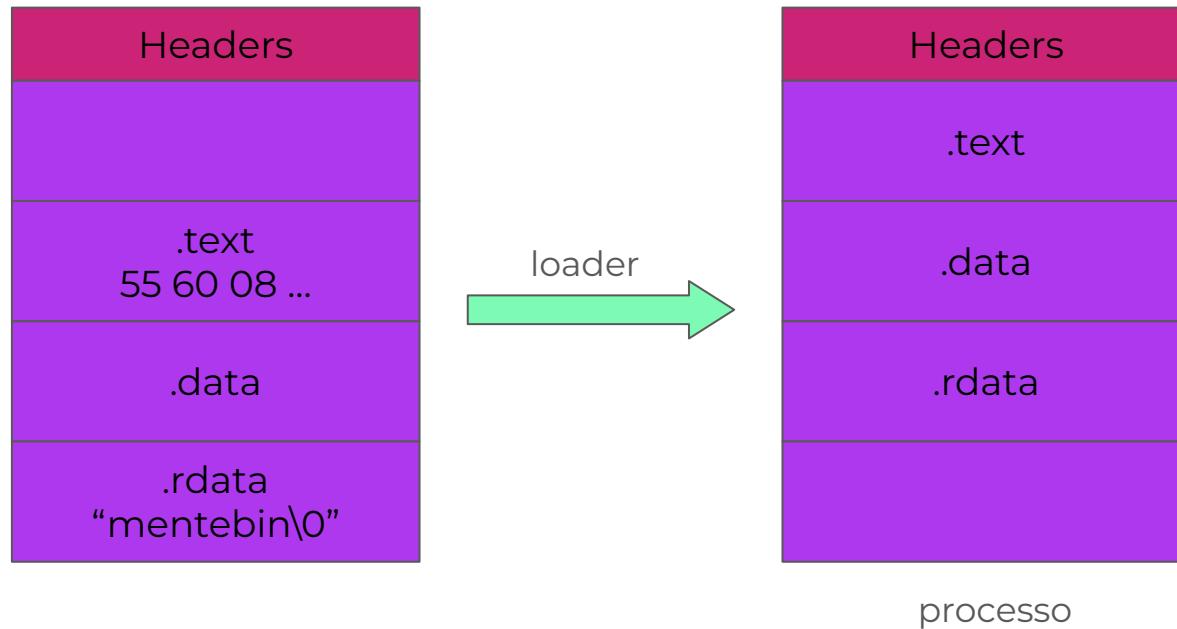
Arquiteturas

- O que é, exatamente, Assembly?



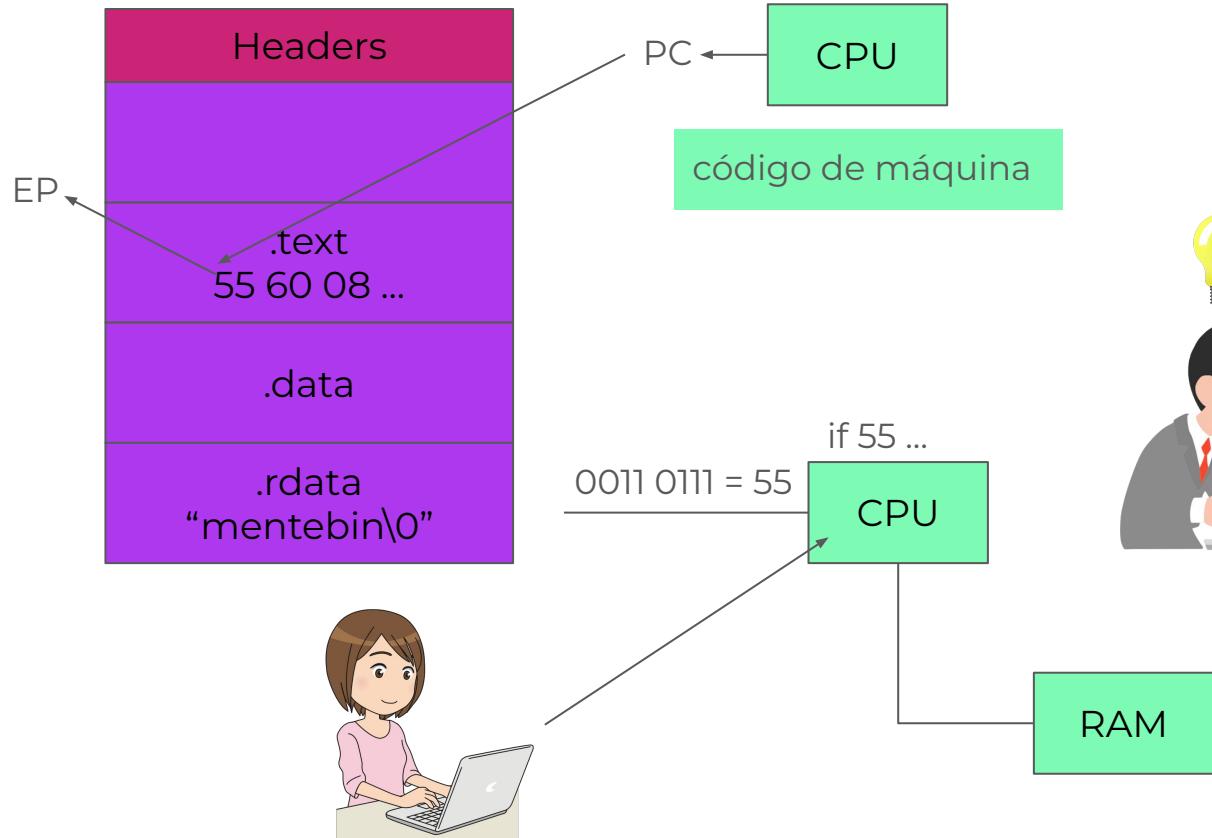


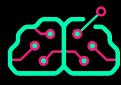
Criação do processo



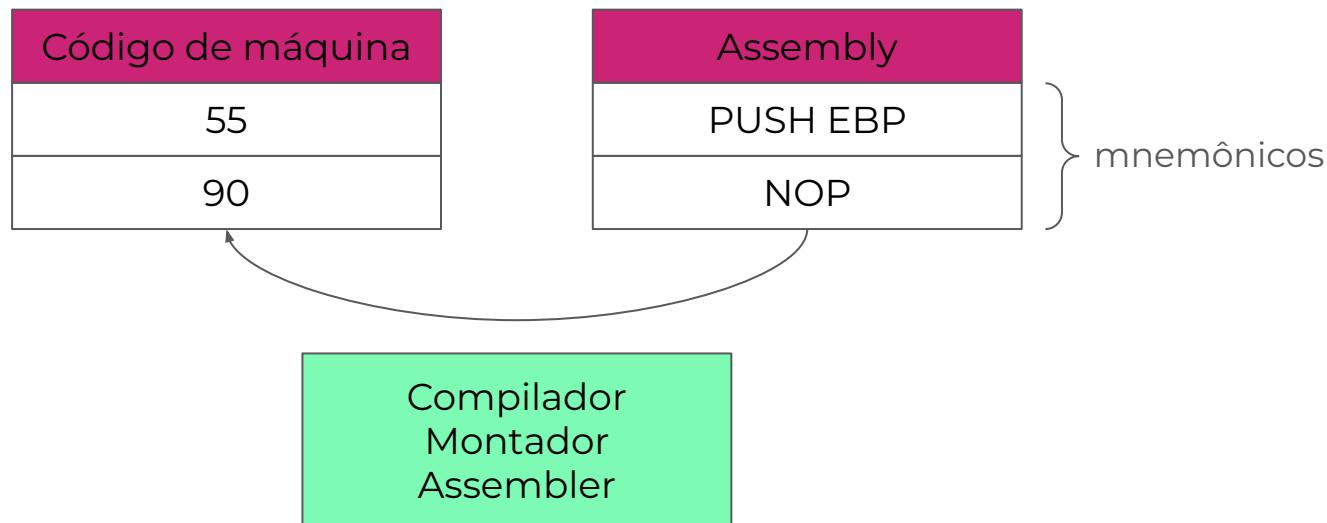


O processo e o processador





O processo e o processador



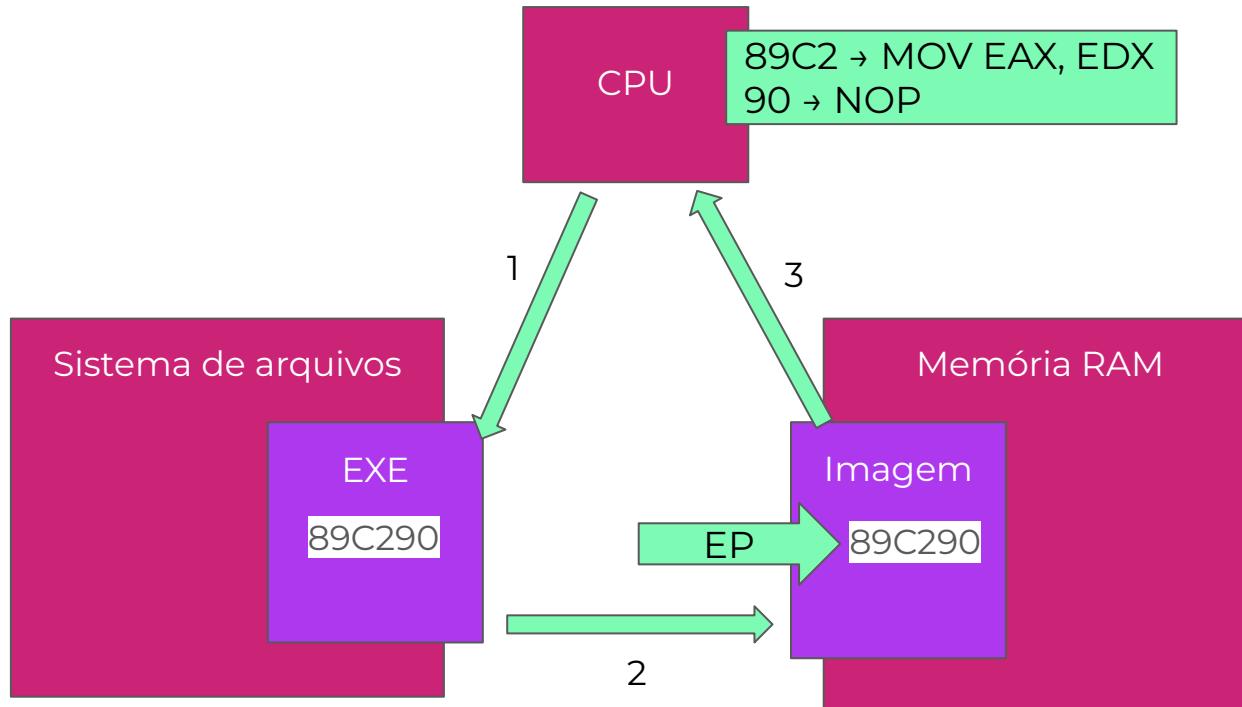


Registradores

- Control Registers (CR)
- **Status/Control Register EFLAGS**
- Segment Registers
- **General Purpose Registers (GPR)**
- **Instruction Pointer (IP)**
- Model Specific Registers (MSR)
- Descriptor Table Registers (DTR)
- SSE
- x87 FPU e MMX
- Debug Registers



Processo de carregamento (simplificado)

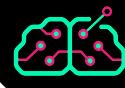




Registradores de Propósito Geral (GPRs)

- Oito GPRs no modo IA-32.
- Todos de 32-bits.

Registrador	Significado	Uso sugerido
EAX	Accumulator	Usado em operações aritméticas
EBX	Base	Ponteiro para dados
ECX	Counter	Contador em repetições
EDX	Data	Usado em operações de E/S
ESI	Source Index	Ponteiro para uma <i>string</i> de origem
EDI	Destination Index	Ponteiro para uma <i>string</i> de destino
ESP	Stack Pointer	Ponteiro para o topo da pilha
EBP	Base Pointer	Ponteiro para a base do <i>stack frame</i>

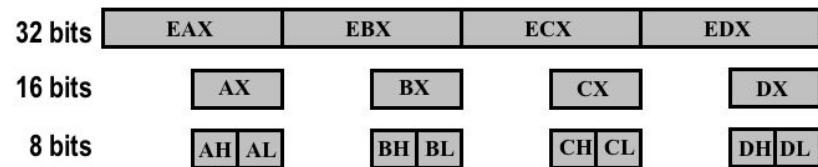


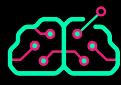
Registradores de Propósito Geral (GPRs)

- Os primeiros quatro podem ser acessados como registradores de 8 e 16-bits.
- Os registradores de 16-bits consistem em dois de 8-bits.
 - Ex.: EAX (32-bits), AX (16-bits), AL (metade baixa, de 8-bits, de AX) e AH (metade alta de AX).

X86-32 register board:

Hybrid puzzle: type-1 and type-2.





Instruction Pointer (IP)

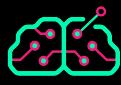
- Contém o endereço da próxima instrução a ser executada.
- É incrementado de acordo com o tamanho da instrução.



EFLAGS

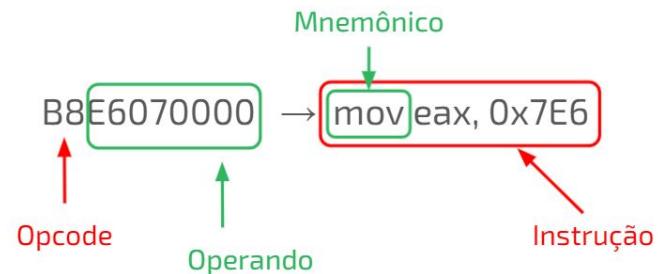
- Dez flags de sistema
- Uma flag de controle
- Seis flags de estado

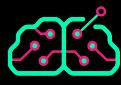
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	I D	V P	V F	A C	V M	R F	0	N T	I O P L	O F	D F	I F	T F	S F	Z F	0	A F	0	P F	1	C F	



Instruções

- O que é uma instrução?
 - Sequência de 1 a 15 bytes de tamanho, podendo compreender *operation code (opcode)*, **operandos** e **prefixo**.
- O processador entende determinadas sequências de bytes como instruções válidas. Exemplo:





MOV

- Copia dados de um lugar para outro. O destino pode ser um registrador ou um endereço de memória. Já a origem pode ser um literal (*immediate*), um registrador ou um endereço de memória.
 - MOV EAX, 1234
 - MOV EAX, EBX
 - MOV EAX, DWORD PTR DS:[402000]
 - MOV DWORD PTR DS:[402000], 1234
 - MOV BX, WORD PTR DS:[402000]
 - MOV AL, BYTE PTR SS:[19FF7C]
 - MOV ECX, [ESP]