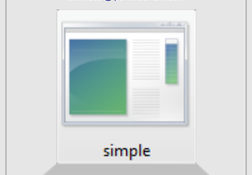
PE101 — a windows executable walkthrough

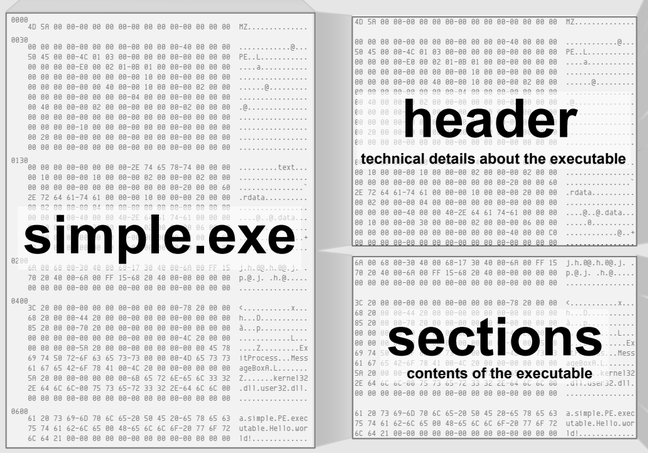
CC BY

Ange Albertini
corkami.com

Dissected PE

SHA-1 b7af4cb51ce38e43e030656eb2698fab408cf9cb
download @ pe101.corkami.com





Hexadecimal dump	ASCII dump	Fields	Values	Explanation
4D 5A 00 00-00 00 00 00-00 00 00 00-00 00 00 00	MZ..............	e_magic	'MZ'	constant signature
Offset:0x30 00 00 00 00-00 00 00 00-00 00 00 00-40 00 00 00	............@...	e_lfanew	0x40	offset of the PE Header ❶
Offset:0x40 50 45 00 00-4C 01 03 00-00 00 00 00-00 00 00 00	PE..L...........	Signature	'PE', 0, 0	constant signature
		Machine	0x14c [intel 386]	processor: ARM/MIPS/Intel/...
		NumberOfSections	3	number of sections ❷
		SizeOfOptionalHeader	0xe0	relative offset of the section table ❷
00 00 00 00-E0 00 02 01...	....a...	Characteristics	0x102 [32b EXE]	EXE/DLL/...
Offset:0x58 ...0B 01 00 00-00 00 00 00	....	Magic	0x10b [32b]	32 bits/64 bits
00 00 00 00-00 00 00 00-00 10 00 00-00 00 00 00	....	AddressOfEntryPoint	0x1000	where execution starts ❺
00 00 00 00-00 00 40 00-00 10 00 00-00 02 00 00	......@.........	ImageBase	0x400000	address where the file should be mapped in memory
00 00 00 00-00 00 00 00-04 00 00 00-00 00 00 00	....	SectionAlignment	0x1000	where sections should start in memory ❷
00 40 00 00-00 02 00 00-00 00 00 00-02 00 00 00	.@..............	FileAlignment	0x200	where sections should start on file ❷
00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00	....	MajorSubsystemVersion	4 [NT 4 or later]	required version of Windows
00 00 00 00-10 00 00 00...	....	SizeOfImage	0x4000	total memory space required
		SizeOfHeaders	0x200	total size of the headers ❸
		Subsystem	2 [GUI]	driver/graphical/command line/...
		NumberOfRvaAndSizes	16	number of data directories ❹
...00 00 00 00-00 00 00 00	....			
00 20 00 00-00 00 00 00-00 00 00 00-00 00 00 00	....	ImportsVA	0x2000	RVA* of the imports ❹
00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00	....			

Offset:0x138

2E 74 65 78-74 00 00 00	.text...	.text	0x1000	0x1000	0x200	0x200	CODE EXECUTE READ
00 10 00 00-00 10 00 00-00 02 00 00-00 02 00 00	....	.rdata	0x1000	0x2000	0x200	0x400	INITIALIZED READ
00 00 00 00-00 00 00 00-00 00 00 00-20 00 00 60	....	.data	0x1000	0x3000	0x200	0x600	DATA READ WRITE
2E 72 64 61-74 61 00 00-00 10 00 00-00 02 00 00	.rdata...........						
00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00	....						
00 00 00 00-40 00 00 40-2E 64 61 74-61 00 00 00	....@..@.data...						
00 10 00 00-00 30 00 00-00 02 00 00-00 06 00 00	.....0..........						
00 00 00 00-00 00 00 00-00 00 00 00-40 00 00 C0	............@..+						

Sections table

RVA* RVA* physical size physical offset

For each section, a SizeofRawData sized block is read from the file at PointerToRawData offset.
It will be loaded in memory at address ImageBase + VirtualAddress in a VirtualSize sized block, with specific characteristics.

Offset:0x200/RVA:0x401000

Hexadecimal dump	ASCII dump
6A 00 68 00-30 40 00 68-17 30 40 00-6A 00 FF 15	j.h.0@.h.0@.j. .
70 20 40 00-6A 00 FF 15-68 20 40 00-00 00 00 00	p.@.j. .h.@.....

x86 assembly

```
push 0
push 0x403000
push 0x403017
push 0
call [0x402070]   ➙MessageBox(0, "Hello World!","a simple PE executable", 0);
push 0
call [0x402068]   ➙ExitProcess(0);
```

Equivalent C code

Offset:0x400/RVA:0x402000

Hexadecimal dump	ASCII dump
3C 20 00 00-00 00 00 00-00 00 00 00-78 20 00 00	<...........x..
68 20 00 00-44 20 00 00-00 00 00 00-00 00 00 00	h...D...........
85 20 00 00-70 20 00 00-00 00 00 00-00 00 00 00	à...p...........
00 00 00 00-00 00 00 00-00 00 00 00-4C 20 00 00	............L...
00 00 00 00-5A 20 00 00-00 00 00 00-00 00 45 78	....Z.........Ex
69 74 50 72-6F 63 65 73-73 00 00 00-4D 65 73 73	itProcess...Mess
61 67 65 42-6F 78 41 00-4C 20 00 00-00 00 00 00	ageBoxA.L.......
5A 20 00 00-00 00 00 00-6B 65 72 6E-65 6C 33 32	Z.......kernel32
2E 64 6C 6C-00 75 73 65-72 33 32 2E-64 6C 6C 00	.dll.user32.dll.

Imports structures

descriptors

0x203c → 0x204c, 0 INT* → Hint,Name 0,ExitProcess
0x2078 → kernel32.dll
0x2068 → 0x204c, 0 IAT*
0x2044 → 0x205a, 0 INT* → Hint,Name 0,MessageBoxA
0x2085 → user32.dll
0x2070 → 0x205a, 0 IAT*
00000

All addresses here are RVAs*

Consequences

after loading,
0x402068 will point to kernel32.dll´s ExitProcess
0x402070 will point to user32.dll´s MessageBoxA

Offset:0x600/RVA:0x403000

Hexadecimal dump	ASCII dump
61 20 73 69-6D 70 6C 65-20 50 45 20-65 78 65 63	a.simple.PE.exec
75 74 61 62-6C 65 00 48-65 6C 6C 6F-20 77 6F 72	utable.Hello.wor
6C 64 21 00	ld!.

Strings

```
a simple PE executable\0
Hello world!\0
```

This is the whole file, however, most PE files contain more elements. Explanations are simplified, for conciseness.

version 1, 3rd May 2012

Loading process

❶ Headers

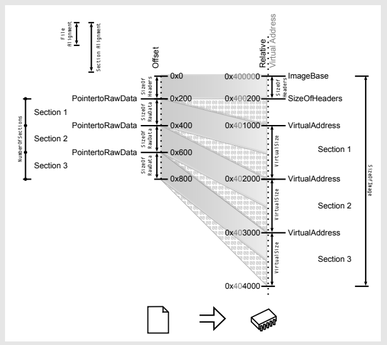
the *DOS Header* is parsed
the *PE Header* is parsed
(its offset is *DOS Header´s* e_lfanew)
the *Optional Header* is parsed
(it follows the *PE Header*)

❷ Sections table

Sections table is parsed
(it is located at: offset (*OptionalHeader*) + *SizeOfOptionalHeader*)
it contains *NumberOfSections* elements
it is checked for validity with alignments:
FileAlignments and *SectionAlignments*

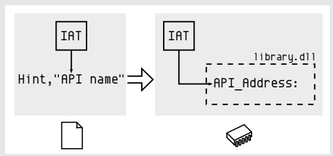
❸ Mapping

the file is mapped in memory according to:
the *ImageBase*
the *SizeOfHeaders*
the Sections table



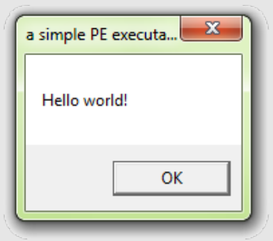
❹ Imports

DataDirectories are parsed
they follow the *OptionalHeader*
their number is *NumOfRVAAndSizes*
imports are always #2
Imports are parsed
each descriptor specifies a *DLLname*
this DLL is loaded in memory
IAT and *INT* are parsed simultaneously
for each API in *INT*
its address is written in the *IAT* entry



❺ Execution

Code is called at the *EntryPoint*
the calls of the code go via the IAT to the APIs



Notes

MZ HEADER *aka* DOS_HEADER
Starts with 'MZ' (initials of *Mark Zbikowski* MS-DOS developer)

PE HEADER *aka* IMAGE_FILE_HEADERS / COFF file header
Starts with 'PE' (Portable Executable)

OPTIONAL HEADER *aka* IMAGE_OPTIONAL_HEADER
Optional only for non-standard PEs but required for executables

RVA Relative Virtual Address
Address relative to ImageBase (at ImageBase, RVA = 0)
Almost all addresses of the headers are RVAs
In code, addresses are *not* relative.

INT Import Name Table
Null-terminated list of pointers to Hint, Name structures

IAT Import Address Table
Null-terminated list of pointers
On file it is a copy of the INT
After loading it points to the imported APIs

HINT
Index in the exports table of a DLL to be imported
Not required but provides a speed-up by reducing look-up