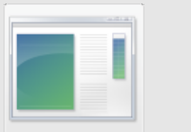
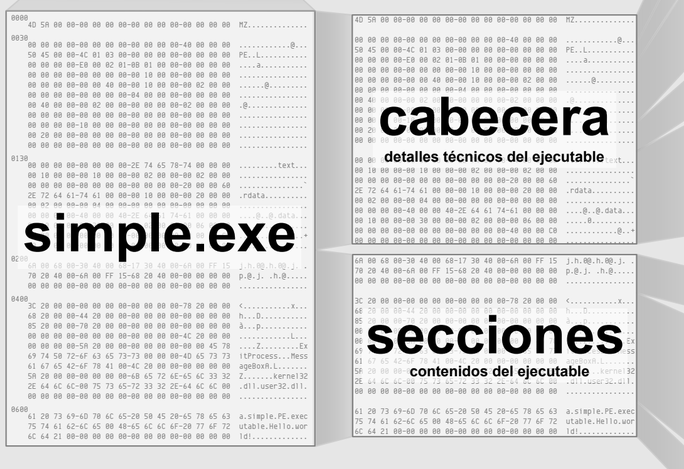
PE 101 — un recorrido por los ejecutables de windows

Ange Albertini
corkami.com

Disección del PE

SHA-1 b7af4cb51ce38e43e030656eb2698fab408cf9cb
descárgalo en pe101.corkami.com





simple.exe

cabecera
detalles técnicos del ejecutable

secciones
contenidos del ejecutable

cabecera de DOS
muestra que es un binario

cabecera PE
muestra que es un binario 'moderno'

cabecera opcional
información del ejecutable

directorios de datos
punteros a estructuras adicionales (exports, imports,...)

tabla de secciones
define como es cargado el archivo en memoria

código
que es ejecutado

imports
enlaces entre el ejecutable y las librerías (Windows)

datos
información utilizada por el código

Volcado hexadecimal	Volcado ASCII	Campos	Valores	Explicación
4D 5A 00 00-00 00 00 00-00 00 00 00-00 00 00 00	MZ..............			
Posición:0x30 00 00 00 00-00 00 00 00-00 00 00 00-40 00 00 00	............@...	e_magic e_lfanew	'MZ' 0x40	firma (constante) posición de la cabecera PE ❶
Posición:0x40 50 45 00 00-4C 01 03 00-00 00 00 00-00 00 00 00	PE..L...........	Signature Machine NumberOfSections SizeOfOptionalHeader Characteristics	'PE', 0, 0 0x14c [intel 386] 3 0xe0 0x102 [32b EXE]	firma (constante) procesador: ARM/MIPS/Intel/... número de secciones ❷ dirección relativa de la tabla de secciones ❷ EXE/DLL/...
00 00 00 00-E0 00 02 01...	....a...			
Posición:0x58 ...0B 01 00 00-00 00 00 00	....	Magic AddressOfEntryPoint ImageBase SectionAlignment FileAlignment MajorSubsystemVersion SizeOfImage SizeOfHeaders Subsystem NumberOfRvaAndSizes	0x10b [32b] 0x1000 0x400000 0x1000 0x200 4 [NT 4 o luego] 0x4000 0x200 2 [GUI] 16	32 bits/64 bits donde comienza la ejecución ❺ dirección de memoria donde debe de ser mapeado el archivo donde deben de empezar las secciones en la memoria ❷ donde deben de empezar en el archivo ❷ versión requerida de Windows espacio total de memoria requerido tamaño total de las cabeceras ❸ controlador/GUI/línea de comandos/... número de directorios de datos ❹
00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00	....			
00 00 00 00-10 00 00 00...	....			
...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* de las imports ❹
00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00	....			

Posición:0x138

2E 74 65 78-74 00 00 00	.text...
00 10 00 00-00 10 00 00-00 02 00 00-00 00 00 00	....
00 00 00 00-00 00 00 00-20 00 00 60	....`
2E 72 64 61-74 61 00 00-00 10 00 00-00 20 00 00	.rdata..... ....
00 02 00 00-00 04 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 00-40 00 00 C0	........@..@..+

Tabla de secciones

Name	VirtualSize	RVA* VirtualAddress	tamaño físico SizeOfRawData	posición física PointerToRawData	Characteristics
.text	0x1000	0x1000	0x200	0x200	CODE EXECUTE READ
.rdata	0x1000	0x2000	0x200	0x400	INITIALIZED READ
.data	0x1000	0x3000	0x200	0x600	DATA READ WRITE

Para cada sección, un bloque del tamaño de SizeofRawData es leído del archivo en la posición PointerToRawData.
El cual será cargado en memoria en la dirección ImageBase + VirtualAddress en un bloque de tamaño VirtualSize, con las características especificadas.

Posición:0x200/RVA:0x401000

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	p.@.j. .h.@....

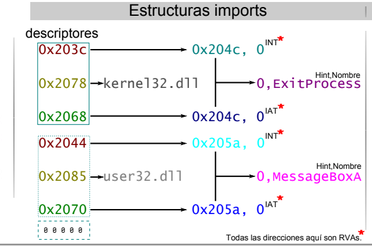
Ensamblador del x86

```
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);
```

Posición:0x400/RVA:0x402000

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.

Estructuras imports



Consecuencias

después de la carga,
0x402068 apuntará a ExitProcess de kernel32.dll
0x402070 apuntará a MessageBoxA de user32.dll

Posición:0x600/RVA:0x403000

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!.

Cadenas

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

Esto es el archivo completo, sin embargo, la mayoría de los PE contienen más elementos. Las explicaciones han sido simplificadas para que resulten más concisas.

versión 1 ES, 6 de Junio del 2012

traducido por Gorka Ramírez

Proceso de carga

❶ Cabeceras

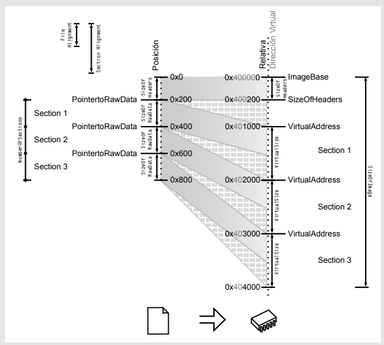
la *cabecera de DOS* es analizada
la *cabecera PE* es analizada
(Su posición de inicio se encuentra en el campo e_lfanew de la *cabecera DOS*)
La *cabecera Opcional* es analizada
(se encuentra a continuación de la *cabecera PE*)

❷ Tabla de Secciones

La tabla de secciones se analiza
(ésta se encuentra en: posición de (*OptionalHeader*) + *SizeOfOptionalHeader*)
contiene *NumberOfSections* elementos
se comprueba que sea válida con las alineaciones:
FileAlignments y *SectionAlignments*

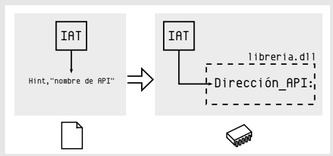
❸ Mapeado

el archivo se mapea en la memoria teniendo en cuenta:
la *ImageBase*
el *SizeOfHeaders*
la *Tabla de Secciones*



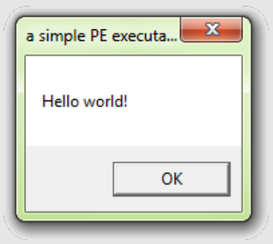
❹ Imports

Se analizan los *DataDirectories*
se encuentran tras el *OptionalHeader*
su número es *NumOfRVAAndSizes*
las *imports* son siempre #2
Se analizan las *imports*
cada descriptor especifica un *DLLname*
dicha DLL es cargada en la memoria
la *IAT* y la *INT* son analizadas simultáneamente
por cada API en la *INT*
su dirección es escrita en la IAT



❺ Ejecución

El código comienza a ejecutarse desde el *EntryPoint*
las llamadas del código se hacen a través de la IAT a las APIs



Notas

MZ HEADER *conocido como* DOS_HEADER
Comienza con 'MZ' (iniciales de *Mark Zbikowski*, desarrollador de MS-DOS)

PE HEADER *conocido como* IMAGE_FILE_HEADERS / COFF file header
Comienza con 'PE' (Portable Executable)

OPTIONAL HEADER *conocido como* IMAGE_OPTIONAL_HEADER
Opcional únicamente para PEs no estándar pero requerida para los ejecutables

RVA (Relative Virtual Address) Dirección Relativa Virtual
Dirección relativa al ImageBase (en el ImageBase, RVA = 0)
Casi todas las direcciones de las cabeceras son RVAs
En el código, las direcciones no son relativas.

INT (Import Name Table) Tabla de Nombres de imports
Lista de punteros acabados en cero que apuntan a estructuras de tipo *Hint, Nombre*

IAT (Import Address Table) Tabla de Direcciones imports
Lista de punteros terminados en cero
En el archivo es una copia del INT
Después de la carga apunta hacia las APIs imports

HINT
Índice en la tabla de exportaciones de la DLL importada
No es necesario pero acelera el proceso reduciendo el tiempo de búsqueda