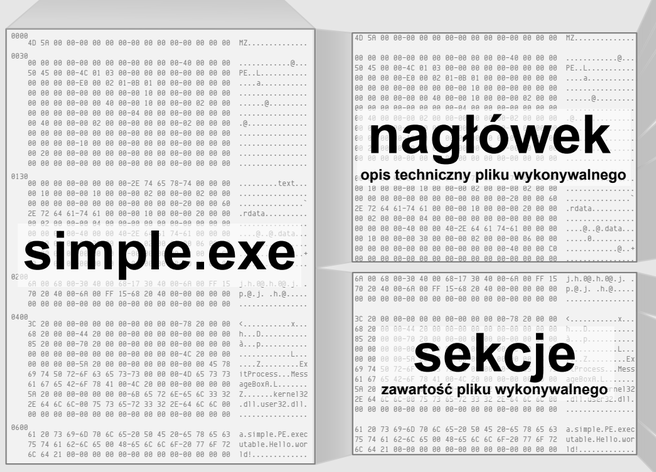
PE101 Plik PE krok po kroku

Ange Albertini
corkami.com

Struktura pliku PE

SHA-1 b7af4cb51ce38e43e030656eb2698fab408cf9cb
pobierz z pe101.corkami.com



Tłumaczenie: Adam Błaszczyk/Gynvael Coldwind

Dane heksadecymalne	ASCII	Pola	Wartości	Opis
4D 5A 00 00-00 00 00 00-00 00 00 00-00 00 00 00	MZ..............	e_magic	'MZ'	stała sygnatura 'MZ'
Offset:0x30 00 00 00 00-00 00 00 00-00 00 00 00-40 00 00 00	............@...	e_lfanew	0x40	ofset nagłówka PE 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	stała sygnatura 'PE'
		Machine	0x14c [intel 386]	rodzaj procesora: ARM/MIPS/Intel/...
		NumberOfSections	3	liczba sekcji ❷
00 00 00 00-E0 00 02 01...	....a...	SizeOfOptionalHeader	0xe0	ofset tablicy sekcji ❷
		Characteristics	0x102 [32b EXE]	EXE/DLL/...
Offset:0x58 ...0B 01 00 00-00 00 00 00	....	Magic	0x10b [32b]	32 bity/64 bity
00 00 00 00-00 00 00 00-00 10 00 00-00 00 00 00	....	AddressOfEntryPoint	0x1000	adres początkowy programu ❺
00 00 00 00-00 00 40 00-00 10 00 00-00 02 00 00	......@.........	ImageBase	0x400000	adres (bazowy) w pamięci
00 00 00 00-00 00 00 00-04 00 00 00-00 00 00 00	....	SectionAlignment	0x1000	wyrównanie sekcji w pamięci ❷
00 40 00 00-00 02 00 00-00 00 00 00-02 00 00 00	.@..............	FileAlignment	0x200	wyrównanie sekcji w pliku ❷
00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00	....	MajorSubsystemVersion	4 [NT 4 lub nowsza]	wymagana wersja systemu Windows
00 00 00 00-10 00 00 00...	....	SizeOfImage	0x4000	całkowity rozmiar w pamięci
		SizeOfHeaders	0x200	całkowita długość wszystkich nagłówków ❸
		Subsystem	2 [GUI]	rodzaj podsystemu: sterownik/GUI/CLI/...
		NumberOfRvaAndSizes	16	liczba wpisów w katalogu danych 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	adres (RVA) tablicy importów* 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
00 10 00 00-00 10 00 00-00 02 00 00-00 02 00 00	....	.rdata	0x1000	0x2000	0x200
00 00 00 00-00 00 00 00-00 00 00 00-20 00 00 60	....	.data	0x1000	0x3000	0x200
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 00-00 00 00 00-40 00 00 C0	............@..+				

Name	VirtualSize	RVA* VirtualAddress	rozmiar fizyczny SizeOfRawData	adres fizyczny 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

Dla każdej sekcji blok, o rozmiarze SizeOfRawData jest odczytywany z pliku z pozycji PointerToRawData i zostaje załadowany do pamięci pod adresem wskazywanym przez wartość VirtualAddress, o rozmiarze VirtualSize i z atrybutami Characteristics.

Offset: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.@....

Kod (asembler x86)	Równoważny kod w C
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);

Offset: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.

Struktury importów	Opis
deskryptory	
0x203c → 0x204c, 0 INT*	
0x2078 → kernel32.dll → Hint,Name 0,ExitProcess	
0x2068 → 0x204c, 0 IAT*	po załadowaniu,
0x2044 → 0x205a, 0 INT*	0x402068 będzie wskazywać na ExitProcess z kernel32.dll
0x2085 → user32.dll → Hint,Name 0,MessageBoxA	0x402070 będzie wskazywać na MessageBoxA z user32.dll
0x2070 → 0x205a, 0 IAT*	
00000	

Wszystkie adresy są relatywne.*

Offset: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!.

Łańcuchy znakowe

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

Większość plików PE posiada więcej elementów składowych (opis jest tu uproszczony, dla celów prezentacji).

wersja 1.1PL, 2 Lipiec 2012

Ładowany proces

❶ Nagłówki

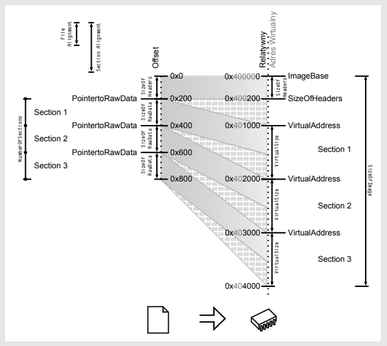
Nagłówek DOS jest parsowany
Nagłówek PE jest parsowany
(jego adres jest odczytywany z *Nagłówka DOS* pod adresem *e_lfanew*)
Opcjonalny nagłówek jest parsowany
(jest on umieszczony za nagłówkiem PE)

❷ Tablica sekcji

Tablica sekcji jest parsowana
(Jest ulokowany pod adresem: ofset (*OptionalHeader*) + *SizeOfOptionalHeader*)
Zawiera *NumberOfSections* elementów
Jest sprawdzane pod kątem
poprawności z wyrównaniami pamięci (alignment)

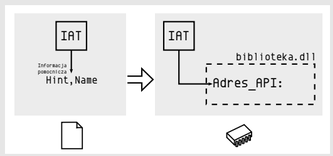
❸ Mapowanie

plik jest mapowany do pamięci według:
- adresu bazowego (ImageBase)
- rozmiarów nagłówków (SizeOfHeaders)
- tablicy sekcji



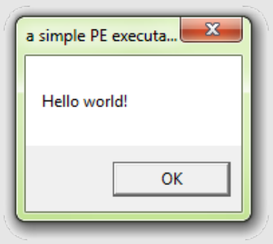
❹ Tablica importów

Katalogi danych są parsowane
są one umieszczone za *Nagłówkiem opcjonalnym*
ich liczba to *NumOfRVAAndSizes*
Tablica importów jest zawsze na pozycji #2
Tablica importów jest parsowana
każdy rekord opisuje jedną bibliotekę DLL
biblioteka ta jest ładowana do pamięci
Tablice *IAT* i *INT* są parsowane równocześnie
dla każdej funkcji API 'name' w bibliotece DLL
jej adres jest umieszczany w tablicy IAT



❺ Uruchomienie

Kod jest wykonywany od adresu *EntryPoint*
odwołania do funkcji API dokonywane są za pośrednictwem tablicy IAT



Informacje dodatkowe

MZ HEADER *czyli* DOS_HEADER
Rozpoczyna się dwoma znakami 'MZ' (Inicjały *Marka Żbikowskiego*, programisty MS-DOS)

PE HEADER *czyli* IMAGE_FILE_HEADERS / COFF
Rozpoczyna się dwoma znakami 'PE' (Portable Executable, czyli przenośny plik wykonywalny)

OPTIONAL HEADER *czyli* IMAGE_OPTIONAL_HEADER
Opcjonalny tylko dla niestandardowych plików PE i jest niezbędny dla plików wykonywalnych

RVA Relatywny Adres Wirtualny
Address relatywny do *ImageBase* (ImageBase ma adres RVA = 0)
Prawie wszystkie adresy w nagłówku to RVA
Adresy w kodzie *nie* są relatywne.

INT Import Name Table
Tablica wskaźników na strukturę składającą się z pól Hint i Name

IAT Import Address Table
Tablica wskaźników zakończona zerem
W tym pliku jest to kopia tablicy INT
Po załadowaniu wskazuje na zaimportowane funkcje API

HINT
Indeks w tablicy eksportów biblioteki DLL, która ma zostać zaimportowana
Nie jest wymagany, ale pozwala przyspieszyć proces ładowania programu