

SCP 이다영

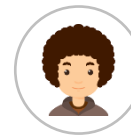
PE File Format

Contents



1. PE 파일이란?

2. RAW & RVA & VA



3. IAT & EAT

4. API 찾기 실습



1. PE(Portable Executable) 파일이란?

: Windows 운영체제에서 사용되는 실행 파일 형식

실행 계열: EXE, SCR

드라이버 계열: SYS, VXD

라이브러리 계열: DLL, OCX, CPL, DRV

오브젝트 파일 계열: OBJ

PE 파일 종류

Q. PE 파일 포맷, 왜 이해해야 해?

“ 실행파일의 어떤 부분이 어떻게 메모리에 로드되고,
어떤 API들이 사용되는지 코드를 보지 않아도 확인할 수 있어!



2-1. RAW(File Offset)

: 파일이 실행되기 전의 주소

Easy_CrackMe.exe

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
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	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	D0	00	00	00Đ...
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	3B	22	9F	06	7F	43	F1	55	7F	43	F1	55	7F	43	F1	55	;"ÿ..CăU.CăU.CăU
00000090	49	65	FB	55	65	43	F1	55	FC	5F	FF	55	73	43	F1	55	IeûUeCăUü_ÿUsCăU
000000A0	7F	43	F0	55	54	43	F1	55	FC	4B	AC	55	7A	43	F1	55	.CăUTCăUüK~UzCăU
000000B0	49	65	FA	55	7D	43	F1	55	B8	45	F7	55	7E	43	F1	55	IeúU}CăU,E÷U~CăU
000000C0	52	69	63	68	7F	43	F1	55	00	00	00	00	00	00	00	00	Rich.CăU.....

2-2. RVA(Relative Virtual Address)

: 특정한 위치(Image Base)로부터 얼마만큼 떨어진 곳에 위치하는지 알려주는 주소

pFile	Data	Description	Value
00000148	00000000	RVA	EXPORT Table
0000014C	00000000	Size	
00000150	00005444	RVA	IMPORT Table
00000154	0000003C	Size	
00000158	00009000	RVA	RESOURCE Table
0000015C	000000CC	Size	
00000160	00000000	RVA	EXCEPTION Table
00000164	00000000	Size	
00000168	00000000	Offset	CERTIFICATE Table
0000016C	00000000	Size	

2-3. VA(Virtual Address)

: 프로세스 가상 메모리의 절대주소

가상 메모리

메인 메모리보다 훨씬 큰 기억공간인 디스크를 가상의 메모리 공간으로 이용하는 기법

2-3. VA(Virtual Address)

파일을 실행하게 되면 프로세스의 어딘가에 존재하는
가상의 메모리 주소공간에 실행한 파일이 올라가게 된다. ^{65%}

→ 매핑(mapping)

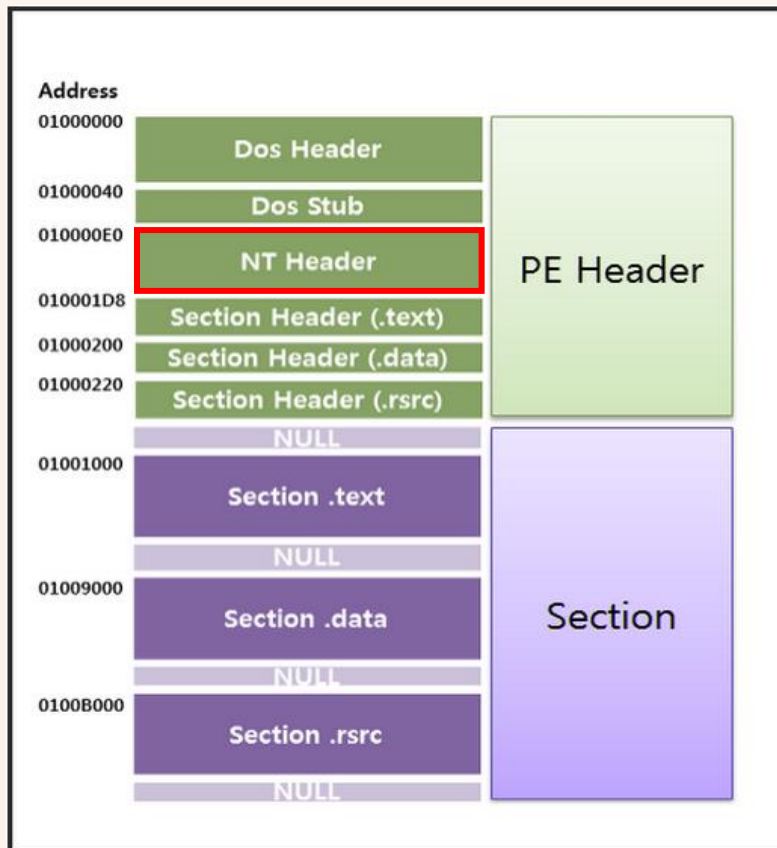


Image Base

: 가상메모리에서의 PE파일이 로딩 되는 주소



VA는 Image Base를 기준으로 만들어지는 주소 !

2-3. VA(Virtual Address)

PE헤더에서 나오는 거의 모든 주소값은 모두 RVA를 의미한다.

65%

“VA(실제 메모리에 로딩되는 주소)를 구하려면?”

➡ **Image Base 주소 + RVA 주소**

2-3. VA(Virtual Address)

Q. 왜 PE Header 대부분의 정보가 RVA일까?

“

‘VA = 0x123’이라고 설정을 해두었는데

해당 위치에 이미 다른 프로그램이 로드되어 있다면?

2-3. VA(Virtual Address)

Q. 왜 PE Header 대부분의 정보가 RVA일까?

“ windows OS에서는 ASLR의 영향을 받아
계속해서 주소값이 바뀐다!
이런 경우에 VA를 이용하게 되면 바뀌는 시작 주소값을
어떻게 예측할 것인가?

: 프로그램을 실행시킬 때마다
각 주소들이 바뀌는 메모리 보호 기법

3-1. IAT(Import Address Table)

: 프로그램이 어떤 라이브러리에서 어떤 함수를 사용하고 있는지를 기술한 테이블

어떤 API들이 사용되는지 확인 -> 어떤 역할을 하는 프로그램인지 추측

3-2. EAT(Export Address Table)

: 라이브러리 파일에서 제공하는 함수를 다른 프로그램에서 가져다 사용할 수 있도록 해주는 매커니즘

00000148	00000000	RVA	EXPORT Table
0000014C	00000000	Size	

4. API 찾기 실습

IMPORT Table의 RVA를 이용해서
실행 프로그램의 API 찾기



65%

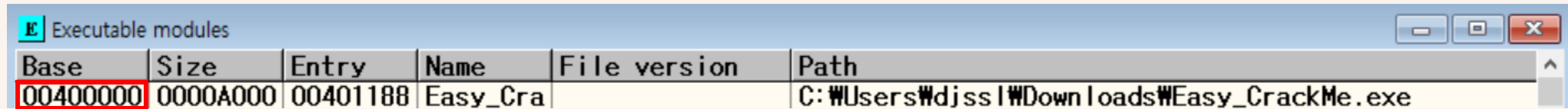
4. API 찾기 실습

(1) IMPORT Table의 RVA

pFile	Data	Description	Value
00000148	00000000	RVA	EXPORT Table
0000014C	00000000	Size	
00000150	00005444	RVA	IMPORT Table
00000154	0000003C	Size	RESOURCE Table
00000158	00009000	RVA	
0000015C	000000CC	Size	EXCEPTION Table
00000160	00000000	RVA	
00000164	00000000	Size	CERTIFICATE Table
00000168	00000000	Offset	
0000016C	00000000	Size	

4. API 찾기 실습

(2) Image Base 확인



E Executable modules					
Base	Size	Entry	Name	File version	Path
00400000	0000A000	00401188	Easy_Cra		C:\Users\Wdjss\Downloads\Easy_CrackMe.exe

$$VA = RVA + \text{Image Base}$$

$$VA = 00005444 + 00400000$$

$$VA = 00405444$$

4. API 찾기 실습

(3) VA 위치로 이동

Address	Hex dump	ASCII
00405444	1C 55 00 00 00 00 00 00 00 00 00 00 00 00 00 00	U.....nU..
00405454	9C 50 00 00 80 54 00 00 00 00 00 00 00 00 00 00	솔..T.....
00405464	14 58 00 00 00 50 00 00 00 00 00 00 00 00 00 00	X...P.....

이 위치가 IMPORT Table이라면 API를 확인할 수 있다!

$$VA = RVA + \text{Image Base}$$

$$VA = 0000551C + 00400000$$

$$VA = 0040551C$$

4. API 찾기 실습

(4) 프로그램의 API 확인

Address	Hex dump																ASCII
0040551C	5C	55	00	00	4E	55	00	00	42	55	00	00	30	55	00	00	WU..NU..BU..OU..
0040552C	00	00	00	00	9E	00	44	69	61	6C	6F	67	42	6F	78	50?DialogBoxP
0040553C	61	72	61	6D	41	00	C6	00	45	6E	64	44	69	61	6C	6F	aramA.?EndDialo
0040554C	67	00	DE	01	4D	65	73	73	61	67	65	42	6F	78	41	00	g.?MessageBoxA.
0040555C	13	01	47	65	74	44	6C	67	49	74	65	6D	54	65	78	74	!!_rGetDlgItemText
0040556C	41	00	55	53	45	52	33	32	2E	64	6C	6C	00	00	77	01	A.USER32.dll..wr
0040557C	47	65	74	4D	6F	64	75	6C	65	48	61	6E	64	6C	65	41	GetModuleHandleA
0040558C	00	00	AF	01	47	65	74	53	74	61	72	74	75	70	49	6E	..?GetStartupIn
0040559C	66	6F	41	00	08	01	47	65	74	43	6F	6D	6D	61	6E	64	foA. _rGetCommand
004055AC	4C	69	6E	65	41	00	DE	01	47	65	74	56	65	72	73	69	LineA.?GetVersi
004055BC	6F	6E	00	00	AF	00	45	78	69	74	50	72	6F	63	65	73	on..?ExitProces

IAT

API



감사합니다 😊