악성코드 분석, 암호화 및 패키징 프로젝트

PE 파일에 대한 보고서

문서 번호 : SRP-002

목 차

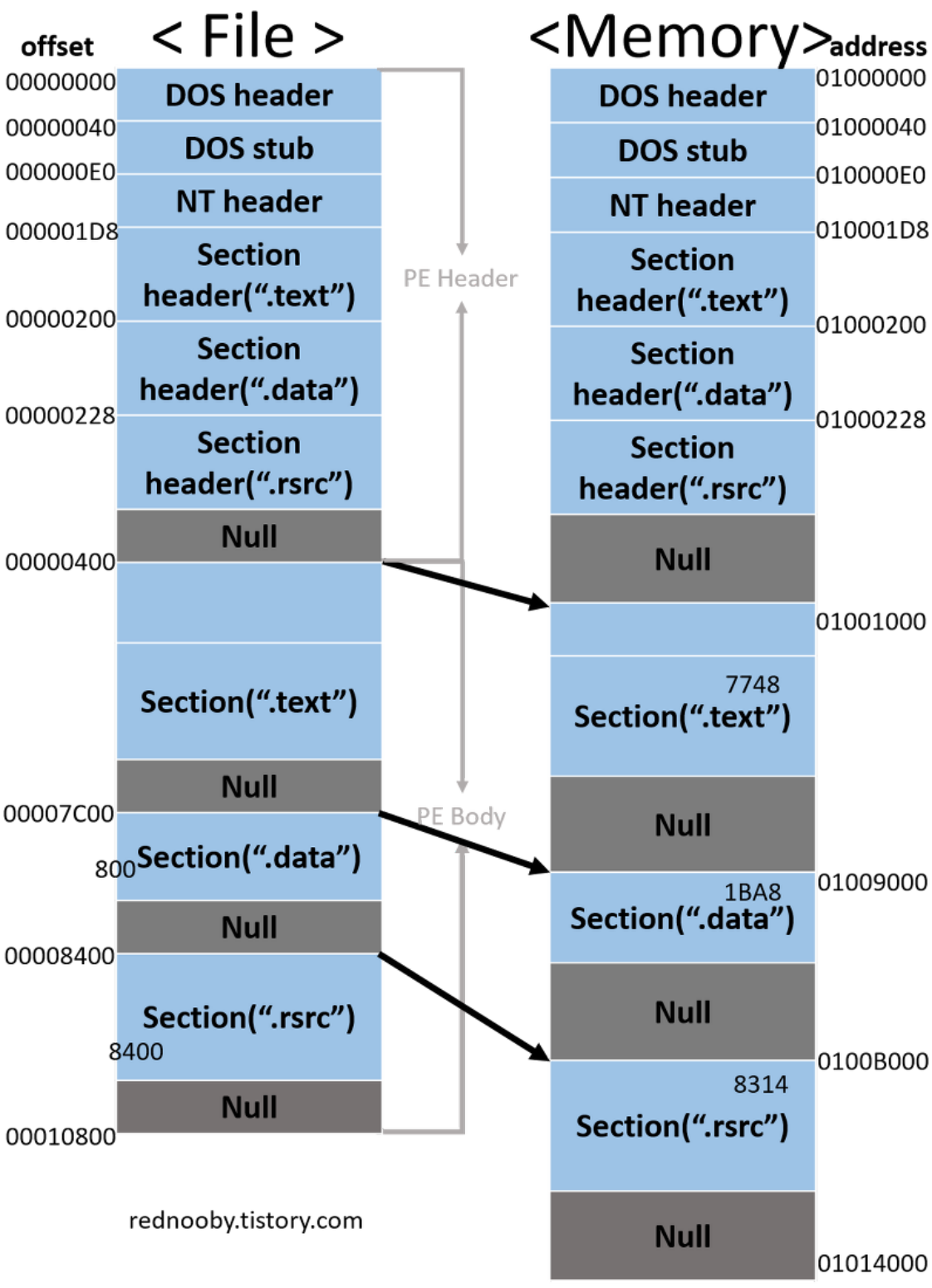
Pre-Vision

I. PE 파일 소개

PE(Portable Executable) 파일 형식은 Windows 운영 체제에서 실행 파일(.exe)이나 DLL 파일을 지원하는 기본 형식입니다. 쉽게 말해, PE 파일은 프로그램이 메모리에 로드되어 실행될 수 있도록 도와주는 중요한 정보를 담고 있습니다. 이 파일 형식은 다양한 컴퓨터 아키텍처, 예를 들어 x86, x64,ARM 등에서 사용할 수 있도록 설계되었습니다.

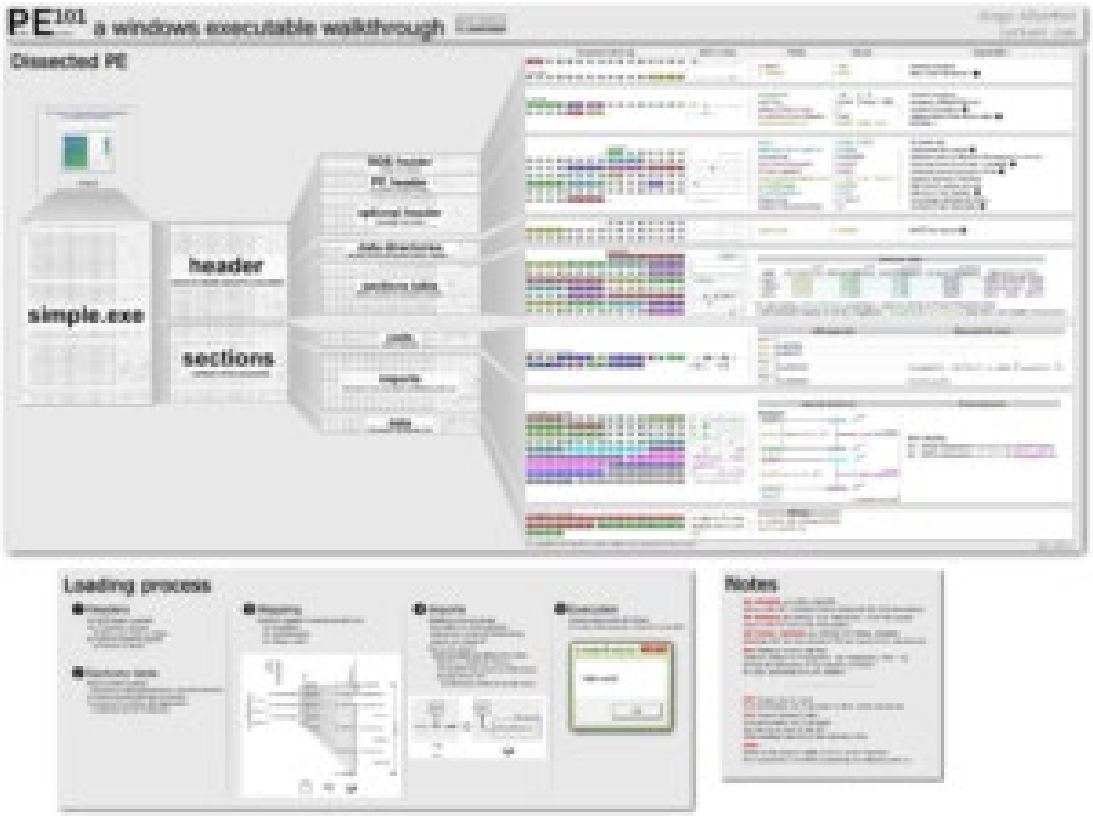
처음에는 MS-DOS 실행 파일을 확장한 형태로 시작되었지만, 지금은 현대적인 Windows 시스템과 DOS 사이의 호환성을 유지하는 역할을 하고 있습니다. . PE 파일은 효율성과 유연성이 뛰어나, 필요에 따라 메모리에 적재될 데이터를 최적화하여 프로그램의 성능과 보안을 높일 수 있습니다. 예를 들어, 실행할 코드와 데이터는 따로따로 저장되어 프로그램이 더 빠르고 안전하게 작동하도록 도와줍니다.

II. PE 파일의 구조


offset
< File >
<Memory>
address
00000000
DOS header
01000000
00000040
DOS stub
01000040
000000E0
NT header
010000E0
000001D8
Section header(".text")
010001D8
00000200
Section header(".data")
01000200
00000228
Section header(".rsrc")
01000228
Null
00000400
Null
01001000
Section(".text")
7748
Section(".text")
Null
00007C00
PE Header
PE Body
Null
800
Section(".data")
1BA8
Section(".data")
01009000
Null
00008400
Null
Section(".rsrc")
8400
0100B000
8314
Section(".rsrc")
Null
00010800
Null
01014000
rednooby.tistory.com

PE 구조

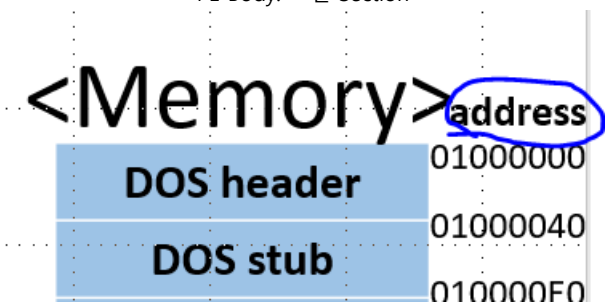
Pre-Vision



[PE 파일(메모장.exe)이 메모리에 로딩되는 모습]

PE header: DOS header ~ Section header

PE Body: 그밑 Section

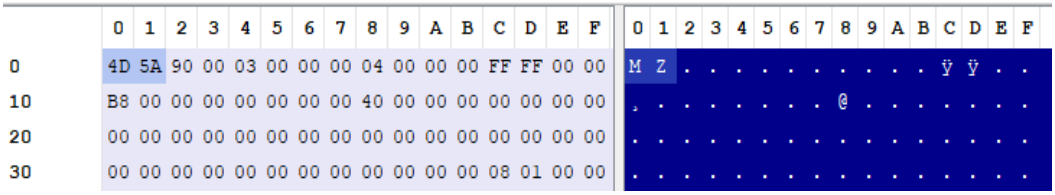


[VA(Virtual Address, 절대주소)]

1. DOS 헤더 (MZ 헤더)

파일의 맨 앞부분에 위치한 MZ 헤더는 오래된 MS-DOS 형식의 흔적입니다. 이 부분에는 작은 DOS 프로그램이 포함되어 있어, DOS 환경에서 실행될 경우 "이 프로그램은 DOS 모드에서 실행될 수 없습니다"라는 메시지를 출력합니다. 현대 시스템에서는 필수적이지 않지만, 이전 시스템과의 호환성을 위해 여전히 유지되고 있습니다.

실행 파일(.exe)을 16 진수 편집기로 열어보면 가장 먼저 MZ 헤더가 보입니다. 이는 운영 체제에서 유효한 실행 파일인지 확인하는 서명(또는 식별자)으로 사용됩니다. PE 헤더의 "PE" 문자열과 함께 파일의 앞부분에 위치합니다.

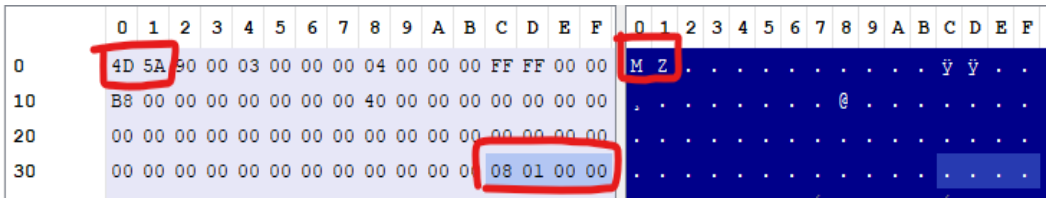


	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00
10	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00
20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	00	00	00	00	00	00	00	00	00	00	00	00	08	01	00	00

하지만 단순히 "MZ" 서명만 있는 것이 아닙니다. 그 뒤에 추가 정보가 이어집니다:

Offset	Name	Value
0	Magic number	5A4D
2	Bytes on last page of file	90
4	Pages in file	3
6	Relocations	0
8	Size of header in paragraphs	4
A	Minimum extra paragraphs needed	0
C	Maximum extra paragraphs needed	FFFF
E	Initial (relative) SS value	0
10	Initial SP value	B8
12	Checksum	0
14	Initial IP value	0
16	Initial (relative) CS value	0
18	File address of relocation table	40
1A	Overlay number	0
1C	Reserved words[4]	0, 0, 0, 0
24	OEM identifier (for OEM information)	0
26	OEM information; OEM identifier specific	0
28	Reserved words[10]	0, 0, 0, 0, 0, 0, 0, 0, 0, 0
3C	File address of new exe header	108

Hex Editor 로 자세히 살펴보면 파일 시작 부분에 Hex 형식으로 정보가 표시되어 있습니다.

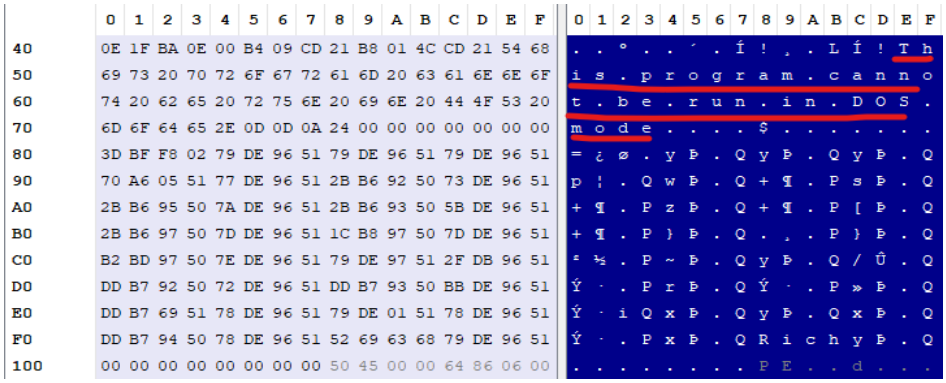


	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00
10	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00
20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	00	00	00	00	00	00	00	00	00	00	00	00	08	01	00	00

표에서 "매직 넘버" 0x5A4D 는 "MZ" 문자열의 16 진수 값과 정확히 일치합니다. 또한 0x108 은 int 값(4 바이트, 32 비트)으로 역순 설정됩니다: 0x08 0x01 0x00 0x00. 이는 16 진수 편집기가 낮은 주소에서 높은 주소로 16 진수 바이트를 출력하고, 리틀 엔디언(예: x86/ia32) 기계에서는 다중 바이트 엔터티의 낮은 자릿수를 낮은 주소에 저장하기 때문입니다.

DOS Stub

이는 단 하나의 목적을 가진 작은 DOS 프로그램입니다. 프로그램이 DOS 에서 실행될 경우 "이 프로그램은 DOS 모드에서 실행할 수 없습니다"라는 텍스트를 출력하고 종료하는 것입니다. DOS Stub 은 MZ 헤더 바로 뒤에 위치합니다.



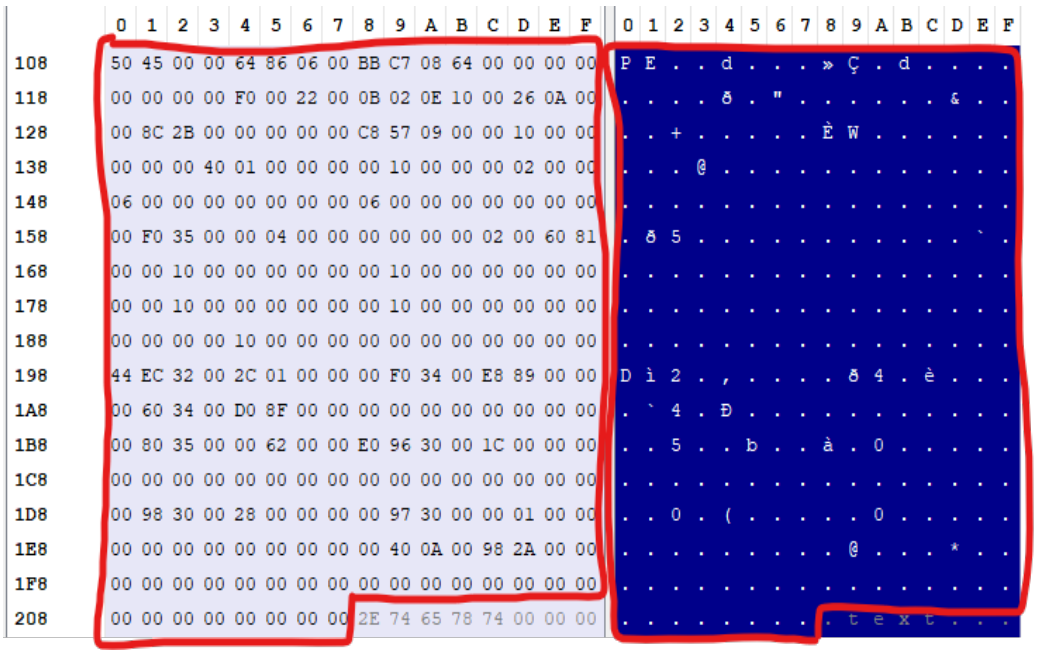
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
40	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	.	.	º	.	.	´	.	Í	!	¸	.	L	Í	!	T	h
50	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	i	s	.	p	r	o	g	r	a	m	.	c	a	n	n	o
60	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	t	.	b	e	.	r	u	n	.	i	n	.	D	O	S	.
70	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00	m	o	d	e	.	.	.	.	$	.	.	.	.	.	.	.
80	3D	BF	F8	02	79	DE	96	51	79	DE	96	51	79	DE	96	51	=	¿	ø	.	y	Þ	.	Q	y	Þ	.	Q	y	Þ	.	Q
90	70	A6	05	51	77	DE	96	51	2B	B6	92	50	73	DE	96	51	p	¦	.	Q	w	Þ	.	Q	+	¶	.	P	s	Þ	.	Q
A0	2B	B6	95	50	7A	DE	96	51	2B	B6	93	50	5B	DE	96	51	+	¶	.	P	z	Þ	.	Q	+	¶	.	P	[	Þ	.	Q
B0	2B	B6	97	50	7D	DE	96	51	1C	B8	97	50	7D	DE	96	51	+	¶	.	P	}	Þ	.	Q	.	¸	.	P	}	Þ	.	Q
C0	B2	BD	97	50	7E	DE	96	51	79	DE	97	51	2F	DB	96	51	²	½	.	P	~	Þ	.	Q	y	Þ	.	Q	/	Û	.	Q
D0	DD	B7	92	50	72	DE	96	51	DD	B7	93	50	BB	DE	96	51	Ý	·	.	P	r	Þ	.	Q	Ý	·	.	P	»	Þ	.	Q
E0	DD	B7	69	51	78	DE	96	51	79	DE	01	51	78	DE	96	51	Ý	·	i	Q	x	Þ	.	Q	y	Þ	.	Q	x	Þ	.	Q
F0	DD	B7	94	50	78	DE	96	51	52	69	63	68	79	DE	96	51	Ý	·	.	P	x	Þ	.	Q	R	i	c	h	y	Þ	.	Q
100	00	00	00	00	00	00	00	00	50	45	00	00	64	86	06	00	.	.	.	.	.	.	.	.	P	E	.	.	d	.	.	.

2. PE 헤더(또는 NT 헤더)

PE 헤더는 MZ 헤더 다음에 위치하며, 현대 Windows 실행 파일의 시작점입니다. 주요 구성 요소는 다음과 같습니다:

1. COFF 파일 헤더: 대상 플랫폼(예: 머신 타입), 섹션 수, 타임스탬프 등의 메타데이터를 포함합니다.
2. 옵션 헤더: 이름과 달리 실행 파일에서 필수적인 헤더로, 프로그램의 로드 및 실행에 중요한 정보를 담고 있습니다. 여기에는 프로그램의 시작 지점(엔트리 포인트), 코드 크기, 데이터 위치 등의 정보가 포함됩니다.

PE 헤더에는 애플리케이션에 대한 중요한 정보가 들어 있으며, 구체적으로 Windows 로더가 프로그램을 로드하고 실행하는 데 사용할 정보가 포함되어 있습니다.
"PE" 문자열로 시작하는데, 이는 Portable Executable 파일이라는 서명이기도 합니다.

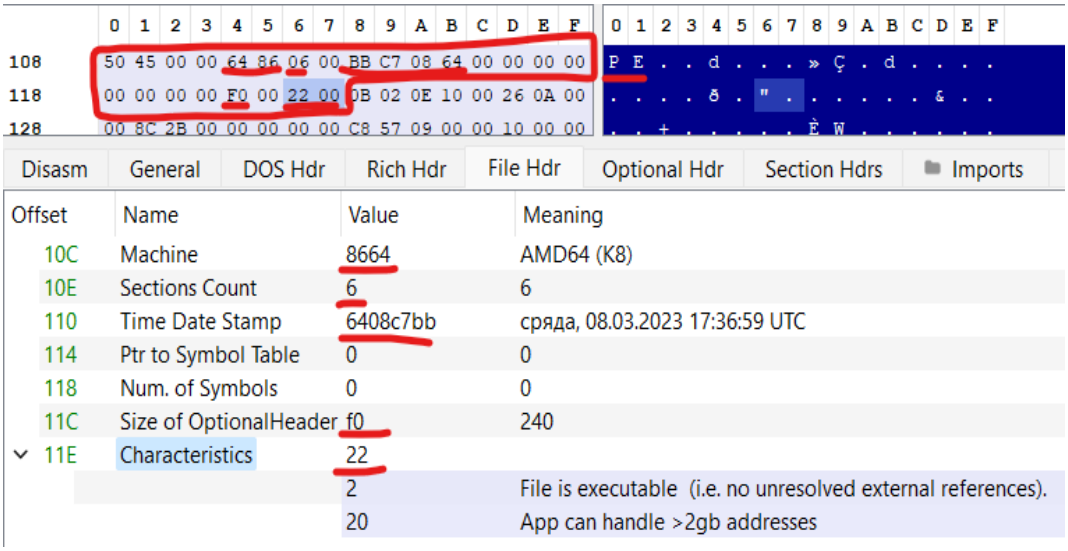


	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
108	50	45	00	00	64	86	06	00	BB	C7	08	64	00	00	00	00	P	E	.	.	d	.	.	.	»	Ç	.	d	.	.	.	.
118	00	00	00	00	F0	00	22	00	0B	02	0E	10	00	26	0A	00	.	.	.	.	ð	.	"	.	.	.	.	.	.	&	.	.
128	00	8C	2B	00	00	00	00	00	C8	57	09	00	00	10	00	00	.	.	+	.	.	.	.	.	È	W	.	.	.	.	.	.
138	00	00	00	40	01	00	00	00	00	10	00	00	00	02	00	00	.	.	.	@	.	.	.	.	.	.	.	.	.	.	.	.
148	06	00	00	00	00	00	00	00	06	00	00	00	00	00	00	00	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
158	00	F0	35	00	00	04	00	00	00	00	00	00	00	02	00	60	81	.	ð	5	.	.	.	.	.	.	.	.	.	.	`	.
168	00	00	10	00	00	00	00	00	00	10	00	00	00	00	00	00	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
178	00	00	10	00	00	00	00	00	00	10	00	00	00	00	00	00	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
188	00	00	00	00	10	00	00	00	00	00	00	00	00	00	00	00	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
198	44	EC	32	00	2C	01	00	00	00	F0	34	00	E8	89	00	00	D	ì	2	.	,	.	.	.	.	ð	4	.	è	.	.	.
1A8	00	60	34	00	D0	8F	00	00	00	00	00	00	00	00	00	00	.	`	4	.	Ð	.	.	.	.	.	.	.	.	.	.	.
1B8	00	80	35	00	00	62	00	00	E0	96	30	00	1C	00	00	00	.	.	5	.	.	b	.	.	à	.	0	.	.	.	.	.
1C8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
1D8	00	98	30	00	28	00	00	00	00	97	30	00	00	01	00	00	.	.	0	.	(	.	.	.	.	.	0	.	.	.	.	.
1E8	00	00	00	00	00	00	00	00	00	40	0A	00	98	2A	00	00	.	.	.	.	.	.	.	.	.	@	.	.	.	*	.	.
1F8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
208	00	00	00	00	00	00	00	00	2E	74	65	78	74	00	00	00	.	.	.	.	.	.	.	.	.	t	e	x	t	.	.	.

3. 이미지 파일 및 선택 헤더

이미지 파일 헤더의 개요는 다음과 같습니다:

Pre-Vision



여기에는 실행 파일이 컴파일된 대상 머신 정보, 섹션 수, 타임스탬프 등이 포함되어 있습니다.

그 다음은 이미지 선택 헤더입니다. 이 헤더에는 이미지 파일 헤더보다 훨씬 더 많은 정보가 들어 있습니다. Entry Point 는 실행 코드의 진입점으로, 사실상 애플리케이션의 "main()" 함수입니다. Windows 로더가 실행을 시작하는 지점입니다.

이 값 0x957C8 은 RVA(상대 가상 주소)로, 실제 파일 오프셋이 아닙니다. Windows 는 Image Base 값을 더해 VA(가상 주소)로 변환하고, Windows 로더가 로드하고 매핑한 후 이 코드의 가상 메모리 주소를 얻습니다. 이 주소를 "VA"라고 부르는 이유는 Windows 가 각 프로세스에 대해 물리적 메모리(RAM)와 독립적인 고유한 VA 공간을 생성하기 때문입니다.
즉, 애플리케이션을 로드할 때 Windows 는 메모리에 프로세스 공간을 할당한 다음 파일을 로드하고 확장합니다. 그런 다음 이 Entry Point 주소를라 메모리에서 코드 실행을 시작합니다.

Pre-Vision

Offset	Name	Value	Value
120	Magic	20B	NT64
122	Linker Ver. (Major)	E	
123	Linker Ver. (Minor)	10	
124	Size of Code	A2600	
128	Size of Initialized Data	2B8C00	
12C	Size of Uninitialized Data	0	
130	Entry Point	957C8	
134	Base of Code	1000	
138	Image Base	140000000	
140	Section Alignment	1000	
144	File Alignment	200	
148	OS Ver. (Major)	6	Windows Vista / Server 2008
14A	OS Ver. (Minor)	0	
14C	Image Ver. (Major)	0	
14E	Image Ver. (Minor)	0	
150	Subsystem Ver. (Major)	6	
152	Subsystem Ver. Minor)	0	
154	Win32 Version Value	0	
158	Size of Image	35F000	
15C	Size of Headers	400	
160	Checksum	0	
164	Subsystem	2	Windows GUI
166	DLL Characteristics	8160	
		20	Image can handle a high entropy 64-bit virtual address space
		40	DLL can move
		100	Image is NX compatible
		8000	TerminalServer aware
168	Size of Stack Reserve	100000	
170	Size of Stack Commit	1000	
178	Size of Heap Reserve	100000	
180	Size of Heap Commit	1000	
188	Loader Flags	0	
18C	Number of RVAs and Sizes	10	
	Data Directory	Address	Size
190	Export Directory	0	0
198	Import Directory	32EC44	12C
1...	Resource Directory	34F000	89E8
1...	Exception Directory	346000	8FD0
1...	Security Directory	0	0
1...	Base Relocation Table	358000	6200
1...	Debug Directory	3096E0	1C
1...	Architecture Specific Data	0	0
1...	RVA of GlobalPtr	0	0
1...	TLS Directory	309800	28
1E0	Load Configuration Directory	309700	100
1E8	Bound Import Directory	0	0
1F0	Import Address Table	A4000	2A98
1F8	Delay Load Import Descriptors	0	0
200	.NET header	0	0

4. 섹션 헤더(또는 섹션 테이블)

PE 헤더 이후 섹션 테이블은 파일의 다른 영역을 정의합니다. 일반적인 섹션은 다음과 같습니다

.text	프로그램의 실행 가능한 코드가 들어 있는 섹션.
.rdata	문자열 또는 상수와 같은 읽기 전용 데이터가 포함됩니다.
.data	초기화된 변수들이 들어 있습니다.
.rsrc	아이콘, 비트맵, 대화 상자와 같은 리소스.
.reloc	동적 연결을 위한 코드 재배치 정보
.bss	초기화되지 않은 데이터
.idata	애플리케이션에서 호출하는 모든 API 에 대한 데이터 가져오기

.edata	데이터 내보내기- 애플리케이션에서 코딩한 모든 공개 API

섹션 헤더(또는 섹션 테이블)에는 파일의 섹션에 대한 정보가 들어 있습니다. 이는 다양한 요인에 따라 파일마다 다를 수 있습니다.

Name	Raw Addr.	Raw size	Virtual Addr.	Virtual Size	Characteristics
.text	400	A2600	1000	A25E4	60000020
.rdata	A2A00	29D400	A4000	29D366	40000040
.data	33FE00	3400	342000	3B10	C0000040
.pdata	343200	9000	346000	8FD0	40000040
.rsrc	34C200	8A00	34F000	89E8	40000040
.reloc	354C00	6200	358000	6200	42000040

여기에는 주로 파일의 각 섹션에 대한 주소, 크기, 특성이 포함됩니다. 바이너리 코드 주입을 수행하려면 새로운 섹션을 추가하거나 기존 섹션을 변경하여 크기를 변경해야 하므로 여기 값을 변경해야 합니다.

III. 결론

PE(Portable Executable) 파일은 Windows 시스템에서 실행 파일과 라이브러리의 중요한 형식으로, 그 구조와 작동 원리를 이해하는 것은 보안과 성능 최적화에 필수적입니다. PE 파일의 헤더, 섹션, 엔트리 포인트 등 구성 요소는 프로그램이 메모리에 적재되고 실행되는 방식에 큰 영향을 미칩니다.

이 보고서에서는 PE 파일의 구조를 상세히 분석하고, 악성코드 탐지와 보안 취약점 분석에 있어서 PE 파일의 중요성을 강조했습니다. 특히 PE 파일의 특정 섹션에 바이너리 코드를 주입하거나 변조하는 기술은 악성코드가 자주 사용하는 방법이므로, PE 파일 분석은 보안 전문가에게 필수적인 작업입니다.

결론적으로, PE 파일의 구조를 깊이 이해하는 것은 악성코드 탐지 및 보안 강화를 위해 매우 중요합니다. 이러한 지식을 바탕으로 더 강력하고 신뢰할 수 있는 보안 솔루션을 개발하고, 악성 행위로부터 시스템을 보호하는 데 기여할 수 있을 것입니다.

IV. 참고문헌

"System.Reflection.PortableExecutable Namespace"
https://learn.microsoft.com/en-us/dotnet/api/system.reflection.portableexecutable?view=net-8.0
" System.Reflection.PortableExecutable API"
https://apisof.net/catalog/044f0ee831a1a64e9a2cbdfd1884b985?fx=net46
"The Windows Portable Executable (PE) File Format", (2023.12.17) form
https://yuriygeorgiev.com/2023/12/18/windows-portable-executable-pe-file-format/

"PE 구조와 인공신경망을 사용한 패커 식별 방안 연구", (2020.02) form
https://repository.hanyang.ac.kr/handle/20.500.11754/123806
"윈도우 PE 포맷 바이너리 데이터를 활용한 Bidirectional LSTM 기반 경량 악성코드 탐지모델", (2022.02.23)
http://www.jics.or.kr, 한국 인터넷 정보학회(23 권 1 호)