

Reversing Fundamental

강사. 유효곤

(ugonfor@gmail.com)

Contents

- 수업
- 리버싱이란,
 - 리버싱이 뭐하는 건가요
 - 리버서의 수준
- 바이너리 분석
 - 바이너리 분석 flow
 - PE header
 - DLL injection
 - API hooking : 다음시간
- 실습
 - SimpleShell 과제 리뷰
 - Flare-on7_Garbage
 - Insomnihack kaboom

리버싱이란,

리버싱이 뭐하는 건가요

리버서의 수준

리버싱이란,

■ 리버싱이 뭐하는 건가요

- Reverse Engineering
- 역공학

리버싱이란,

리버싱이 뭐하는 건가요

- 과거 저를 가르쳤던 교수님의 말을 빌리자면...
 - 레시피를 보고, 음식을 만들어내는 것이 코딩
 - 음식을 보고 레시피를 알아내는 것이 리버싱
- 완성된 바이너리를 보고, 내부 로직을 알아내는 기술

리버싱이란,

리버서의 수준 (지극히 개인적인 생각)

- 초급 리버서
 - 리버싱 툴들을 다룰 줄 알면서 최소한의 분석을 할 수 있는 수준
- 중급 리버서(본인도 이 수준에 머물러 있다고 생각함.)
 - 다양한 형태의 바이너리를 분석할 수 있는 수준
 - 감각이 뛰어나서 빠른 시간에 핵심 알고리즘을 알아낼 수 있는 수준
- 고급 리버서
 - 분석 과정을 자동화할 수 있는 수준

바이너리 분석

바이너리 분석 flow

PE header(파일 시그니처, 헤더정보, IAT/EAT)

바이너리 분석

바이너리 분석 flow

- 바이너리 분석 과정
 - 정적분석
 - 로직분석(내부 핵심 알고리즘)
 - 정적분석의 일부지만 워낙 중요하기에 따로 분류함.
 - 동적분석

일반적으로 오른쪽과
같은 순서로
분석을 진행함

분류	과정
정적분석	파일 시그니처
	파일 헤더 정보
	IAT, EAT
	Strings
	Certificate
	Resource
	packing/unpacking/VMProtect 난독화 해제
로직분석	디스어셈블 / 디컴파일
동적분석	Anti-Reversing 기법들 우회
	레지스트리
	패킷 캡처
	Dll 목록확인
	프로세스 목록확인
	파일 I/O 확인
	API tracing

대표적인 것들을 나열하였으며, 이외에도 다양한 방법들 존재

바이너리 분석

바이너리 분석 flow

- 바이너리 분석 과정
 - 정적분석
 - 로직분석(내부 핵심 알고리즘)
 - 정적분석의 일부지만 워낙 중요하기에 따로 분류함.
 - 동적분석

일반적으로 오른쪽과
같은 순서로
분석을 진행함

분류	과정
정적분석	파일 시그니처
	파일 헤더 정보
	IAT, EAT
	Strings
	Certificate
로직분석	Resource
	packing/unpacking/VMProtect 난독화 해제
	디스어셈블 / 디컴파일
	Anti-Reversing 기법들 우회
	레지스트리
동적분석	패킷 캡처
	Dll 목록확인
	프로세스 목록확인
	파일 I/O 확인
	API tracing

대표적인 것들을 나열하였으며, 이외에도 다양한 방법들 존재

바이너리 분석

■ PE header

- 리버싱을 해야하는 바이너리 대상은 대부분 윈도우 바이너리인 경우가 많기 때문에, PE File을 기준으로 설명
 - Client PC가 대부분 윈도우이기 때문
- 대부분의 내용을 elf에 대해서도 적용 가능함.

바이너리 분석

PE header

- 분석 툴

- Exeinfo
- Pe studio
- Cff explorer
- pe explorer
- ...

CFF Explorer VIII - [main - 복사본 - 복사본.exe]

File Settings ?

main - 복사본 - 복사본.exe

File: main - 복사본 - 복사본.exe

- Dos Header
- Nt Headers
 - File Header
 - Optional Header
 - Data Directories [x]
- Section Headers [x]
- Import Directory
- Resource Directory
- Exception Directory
- Relocation Directory
- Debug Directory
- Address Converter
- Dependency Walker
- Hex Editor
- Identifier
- Import Adder
- Quick Disassembler
- Rebuilder
- Resource Editor

Property	Value
File Name	C:\Users\Wryuhygon\... [REDACTED]
File Type	Portable Executable 64
File Info	Microsoft Visual C++ 8.0 (DLL)
File Size	50.31 MB (52749193 bytes)
PE Size	288.00 KB (294912 bytes)
Created	Wednesday 27 October 2021, 22.07.45
Modified	Wednesday 27 October 2021, 12.50.33
Accessed	Wednesday 27 October 2021, 22.07.45
MD5	F5FFAFB450C6EA3CEE93360618125C2C
SHA-1	CAB0251DCDFF9E896D6F6DE97288168D61DD3006

Property	Value
Empty	No additional info available

바이너리 분석

PE header

- **Dos Header** : 16bit Dosbox와의 호환을 위해 남아있는 헤더
- **Nt headers** : 실제 PE File에 대한 정보를 가지고 있는 헤더
 - **File Header** : 파일 자체에 대한 정보
 - **Optional Header** : 파일 내부 코드 실행에 필수적인 정보
 - **Data Directories** : 디렉토리 구조로 관리하는 데이터에 대한 디렉토리 정보
- **Section Header** : .text, .rdata 등 section(segment) 정보를 가지고 있는 헤더

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
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	F8	00	00	00@...
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	27	46	05	37	63	27	6B	64	63	27	6B	64	63	27	6B	64	'F.7c'kdc'kdc'kd
00000090	77	4C	6F	65	68	27	6B	64	77	4C	68	65	64	27	6B	64	wLoeh'kdwLhed'kd
000000A0	77	4C	6E	65	DF	27	6B	64	05	48	96	64	67	27	6B	64	wLneB'kd.H-dg'kd
000000B0	BA	53	6E	65	45	27	6B	64	BA	53	6F	65	72	27	6B	64	°SneE'kd°Soer'kd
000000C0	BA	53	68	65	6A	27	6B	64	77	4C	6A	65	68	27	6B	64	°Shej'kdwLjeh'kd
000000D0	63	27	6A	64	EB	27	6B	64	B9	53	6F	65	70	27	6B	64	c'jdë'kd¹Soep'kd
000000E0	B9	53	69	65	62	27	6B	64	52	69	63	68	63	27	6B	64	¹Sieb'kdRichc'kd
000000F0	00	00	00	00	00	00	00	00	50	45	00	00	64	86	07	00PE .dt..
00000100	92	25	06	61	00	00	00	00	00	00	00	00	F0	00	22	00	'%.a.....8.".
00000110	0B	02	0E	1C	00	36	02	00	00	46	02	00	00	00	00	006....F.....

바이너리 분석

PE header

- Dos Header
 - 알 필요 없음.
 - 이제는 사용되지 않는 부분
 - 과거 16bit와 32bit가 혼용되던 시기에 호환성을 위하여 만든 헤더
- 그럼에도 알아야 하는 것
 1. MZ로 시작함.
 2. e_lfarlc : PE의 시작 지점을 나타냄

main.exe			
Member	Offset	Size	Value
e_magic	00000000	Word	5A4D
e_cblp	00000002	Word	0090
e_cp	00000004	Word	0003
e_crlc	00000006	Word	0000
e_cparhdr	00000008	Word	0004
e_minalloc	0000000A	Word	0000
e_maxalloc	0000000C	Word	FFFF
e_ss	0000000E	Word	0000
e_sp	00000010	Word	00B8
e_csum	00000012	Word	0000
e_ip	00000014	Word	0000
e_cs	00000016	Word	0000
e_lfarlc	00000018	Word	0040
e_ovno	0000001A	Word	0000
e_res	0000001C	Word	0000
	0000001E	Word	0000
	00000020	Word	0000
	00000022	Word	0000
e_oemid	00000024	Word	0000
e_oeminfo	00000026	Word	0000
e_res2	00000028	Word	0000
	0000002A	Word	0000
	0000002C	Word	0000
	0000002E	Word	0000

바이너리 분석

PE header

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
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	F8	00	00	00
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	27	46	05	37	63	27	6B	64	63	27	6B	64	63	27	6B	64	'F.7c'kdc'kdc'kd
00000090	77	4C	6F	65	68	27	6B	64	77	4C	68	65	64	27	6B	64	wLoeh'kdwLhed'kd
000000A0	77	4C	6E	65	DF	27	6B	64	05	48	96	64	67	27	6B	64	wLneß'kd.H-dg'kd
000000B0	BA	53	6E	65	45	27	6B	64	BA	53	6F	65	72	27	6B	64	°SneE'kd°Soer'kd
000000C0	BA	53	68	65	6A	27	6B	64	77	4C	6A	65	68	27	6B	64	°Shej'kdwLjeh'kd
000000D0	63	27	6A	64	EB	27	6B	64	B9	53	6F	65	70	27	6B	64	c'jdë'kd³Soep'kd
000000E0	B9	53	69	65	62	27	6B	64	52	69	63	68	63	27	6B	64	³Sieb'kdwLjeh'kd
000000F0	00	00	00	00	00	00	00	00	50	45	00	00	64	86	07	00PE ..dt..
00000100	92	25	06	61	00	00	00	00	00	00	00	00	F0	00	22	00	'%.a.....ð.%.
00000110	0B	02	0E	1C	00	36	02	00	00	46	02	00	00	00	00	006...F.....

바이너리 분석

■ PE header

- NT Header
 - File Header
 - Optional Header
 - Data Directories
- 본격적으로 Pe File에 대한 정보를 나타내는 부분

바이너리 분석

PE header

- File Header
 - Machine
 - NumberOfSections
 - TimeDateStamp
 - PointerToSymbolTable
 - NumberOfSymbols
 - **SizeOfOptionalHeader**
 - Characteristics

main.exe				
Member	Offset	Size	Value	Meaning
Machine	000000FC	Word	8664	AMD64 (K8)
NumberOfSections	000000FE	Word	0007	
TimeDateStamp	00000100	Dword	61062592	
PointerToSymbolT...	00000104	Dword	00000000	
NumberOfSymbols	00000108	Dword	00000000	
SizeOfOptionalHea...	0000010C	Word	00F0	
Characteristics	0000010E	Word	0022	Click here

바이너리 분석

PE header

- Optional Header
 - Magic
 - MajorLinkerVersion
 - MinorLinkerVersion
 - SizeOfCode
 - SizeOfInitializedData
 - SizeOfUninitializedData
 - AddressOfEntryPoint
 - BaseOfCode
 - ImageBase
 - SectionAlignment
 - FileAlignment
 - MajorOperatingSystemVersion
 - MinorOperatingSystemVersion
 - MajorImageVersion
 - MinorImageVersion
- MajorSubsystemVersion
- MinorSubsystemVersion
- Win32VersionValue
- SizeOfImage
- SizeOfHeaders
- Checksum
- Subsystem
- DllCharacteristics
- SizeOfStackReserve
- SizeOfStackCommit
- SizeOfHeapReserve
- SizeOfHeapCommit
- LoaderFlags
- NumberOfRvaAndSizes

main.exe				
Member	Offset	Size	Value	Meaning
Magic	00000110	Word	020B	PE64
MajorLinkerVersion	00000112	Byte	0E	
MinorLinkerVersion	00000113	Byte	1C	
SizeOfCode	00000114	Dword	00023600	
SizeOfInitializedData	00000118	Dword	00024600	
SizeOfUninitializedData	0000011C	Dword	00000000	
AddressOfEntryPoint	00000120	Dword	0000A87C	.text
BaseOfCode	00000124	Dword	00001000	
ImageBase	00000128	Qword	0000000140000000	
SectionAlignment	00000130	Dword	00001000	
FileAlignment	00000134	Dword	00000200	
MajorOperatingSystemVer...	00000138	Word	0005	
MinorOperatingSystemVer...	0000013A	Word	0002	
MajorImageVersion	0000013C	Word	0000	
MinorImageVersion	0000013E	Word	0000	
MajorSubsystemVersion	00000140	Word	0005	
MinorSubsystemVersion	00000142	Word	0002	
Win32VersionValue	00000144	Dword	00000000	
SizeOfImage	00000148	Dword	0005C000	
SizeOfHeaders	0000014C	Dword	00000400	
Checksum	00000150	Dword	032583F0	
Subsystem	00000154	Word	0002	Windows GUI
DllCharacteristics	00000156	Word	8160	Click here
SizeOfStackReserve	00000158	Qword	0000000000100000	
SizeOfStackCommit	00000160	Qword	0000000000001000	
SizeOfHeapReserve	00000168	Qword	0000000000100000	
SizeOfHeapCommit	00000170	Qword	0000000000001000	
LoaderFlags	00000178	Dword	00000000	
NumberOfRvaAndSizes	0000017C	Dword	00000010	

바이너리 분석

PE header

- Data Directory

- Export Directory
- Import Directory
- Resource Directory
- Exception Directory
- Security Directory
- Relocation Directory
- Debug Directory
- Architecture Directory
- Reserved Directory

- TLS Directory

- Configuration Directory
- Bound Import Directory
- Import Address Table Directory
- Delay Import Directory
- .Net MetaData Directory

- 각각에 대해 RVA와 Size 존재

main.exe				
Member	Offset	Size	Value	Section
Export Directory RVA	00000180	Dword	00000000	
Export Directory Size	00000184	Dword	00000000	
Import Directory RVA	00000188	Dword	00035A80	.rdata
Import Directory Size	0000018C	Dword	00000078	
Resource Directory RVA	00000190	Dword	00048000	.rsrc
Resource Directory Size	00000194	Dword	0000F4E0	
Exception Directory RVA	00000198	Dword	00048000	.pdata
Exception Directory Size	0000019C	Dword	00001DB8	
Security Directory RVA	000001A0	Dword	00000000	
Security Directory Size	000001A4	Dword	00000000	
Relocation Directory RVA	000001A8	Dword	00058000	.reloc
Relocation Directory Size	000001AC	Dword	00000748	
Debug Directory RVA	000001B0	Dword	000338E0	.rdata
Debug Directory Size	000001B4	Dword	0000001C	
Architecture Directory RVA	000001B8	Dword	00000000	
Architecture Directory Size	000001BC	Dword	00000000	
Reserved	000001C0	Dword	00000000	
Reserved	000001C4	Dword	00000000	
TLS Directory RVA	000001C8	Dword	00000000	
TLS Directory Size	000001CC	Dword	00000000	
Configuration Directory RVA	000001D0	Dword	00033900	.rdata
Configuration Directory Size	000001D4	Dword	00000138	
Bound Import Directory RVA	000001D8	Dword	00000000	
Bound Import Directory Size	000001DC	Dword	00000000	
Import Address Table Directory...	000001E0	Dword	00025000	.rdata
Import Address Table Directory...	000001E4	Dword	000003F0	
Delay Import Directory RVA	000001E8	Dword	00000000	
Delay Import Directory Size	000001EC	Dword	00000000	
.NET MetaData Directory RVA	000001F0	Dword	00000000	
.NET MetaData Directory Size	000001F4	Dword	00000000	

바이너리 분석

PE header

- Section Header

main.exe									
Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	Characteristics
00000200	00000208	0000020C	00000210	00000214	00000218	0000021C	00000220	00000222	00000224
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.text	00023570	00001000	00023600	00000400	00000000	00000000	0000	0000	60000020
.rdata	00011818	00025000	00011A00	00023A00	00000000	00000000	0000	0000	40000040
.data	000103B8	00037000	00000E00	00035400	00000000	00000000	0000	0000	C0000040
.pdata	00001DB8	00048000	00001E00	00036200	00000000	00000000	0000	0000	40000040
._RDATA	000000F4	0004A000	00000200	00038000	00000000	00000000	0000	0000	40000040
.rsrc	0000F4E0	0004B000	0000F600	00038200	00000000	00000000	0000	0000	40000040
.reloc	00000748	0005B000	00000800	00047800	00000000	00000000	0000	0000	42000040

바이너리 분석

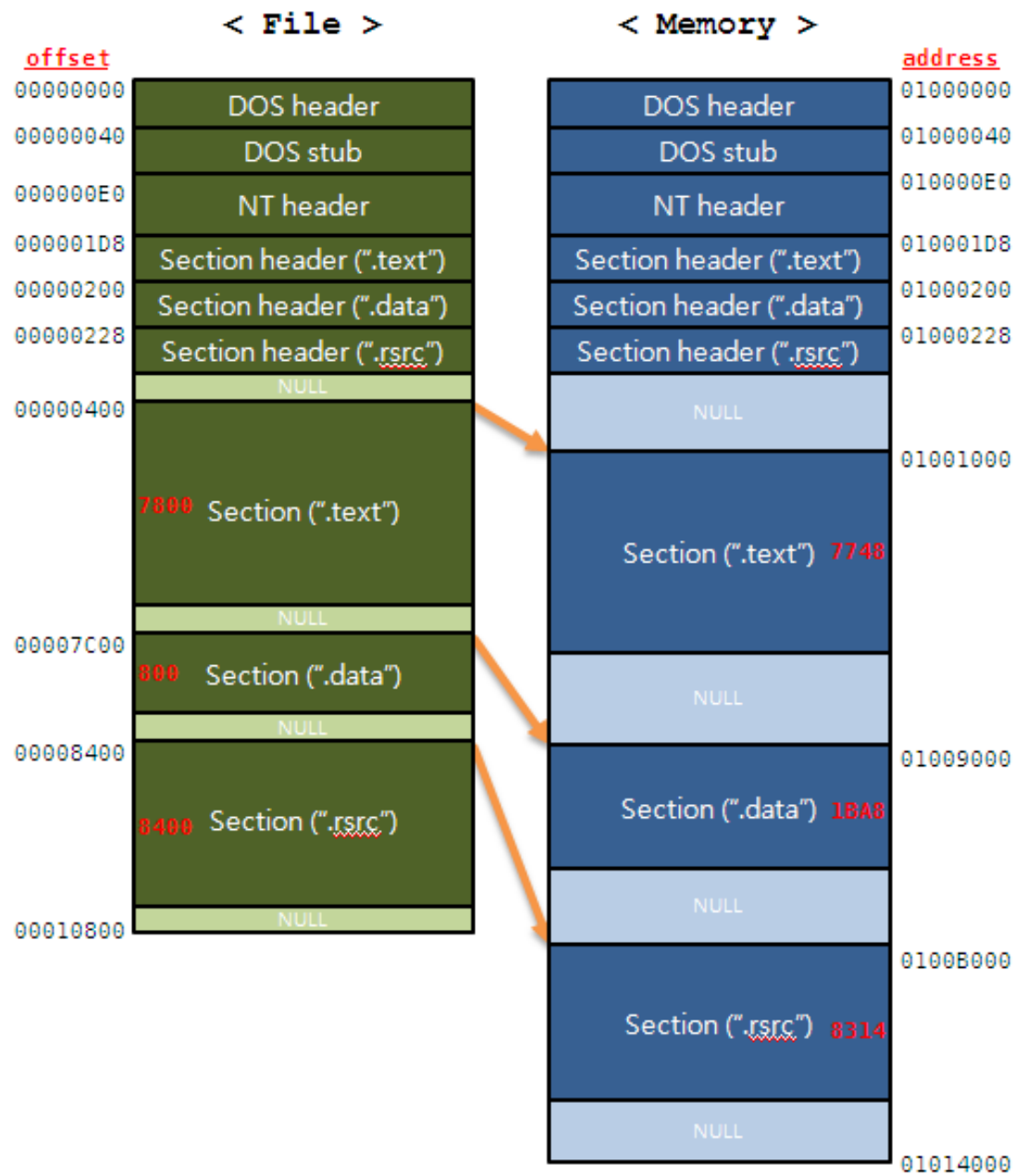
■ PE header

- Section Header
 - VirtualSize
 - VirtualAddress
 - SizeOfRawData
 - PointerToRawData
 - Characteristics
- 나머지는 사용되지 않음.

바이너리 분석

PE header

- RVA, RWA 변환
 - RVA : IDA에서 보여주는 주소값, 실제로 바이너리가 실행 되었을 때 메모리에 매핑되는 주소값
 - RWA : 바이너리를 HxD와 같이 실제로 뜯어보았을 때 주소값
- 헤더 : imagebase에서 같은 offset 에 존재
- 나머지 데이터 : 해당하는 section의 VA를 확인하고 offset을 더해주어야 함.



바이너리 분석

PE header(IAT/EAT)

- Import Address Table
 - 어떤 라이브러리에서 어떤 함수를 사용하는 지 기술하는 테이블
 - DLL을 통해서 함수를 불러와야 하는 데, 이에 대한 정보를 기술하고 있는 테이블
- Import Directory
 - (a.k.a IMAGE_IMPORT_DESCRIPTOR)
 - 자세히 그 라이브러리에서 어떤 함수를 호출하는 지에 대한 정보가 있음.

```
typedef struct IMAGE_IMPORT_DESCRIPTOR {  
    union {  
        DWORD Characteristics;  
        DWORD OriginalFirstThunk; // INT(Import Name Table) address (RVA)  
    };  
    DWORD TimeDateStamp;  
    DWORD ForwarderChain;  
    DWORD Name; // library name string address (RVA)  
    DWORD FirstThunk; // IAT(Import Address Table) address (RVA)  
} IMAGE_IMPORT_DESCRIPTOR;  
  
typedef struct IMAGE_IMPORT_BY_NAME {  
    WORD Hint; // ordinal  
    BYTE Name[1]; // function name string  
} IMAGE_IMPORT_BY_NAME, *PIMAGE_IMPORT_BY_NAME;
```

바이너리 분석

PE header(IAT/EAT)

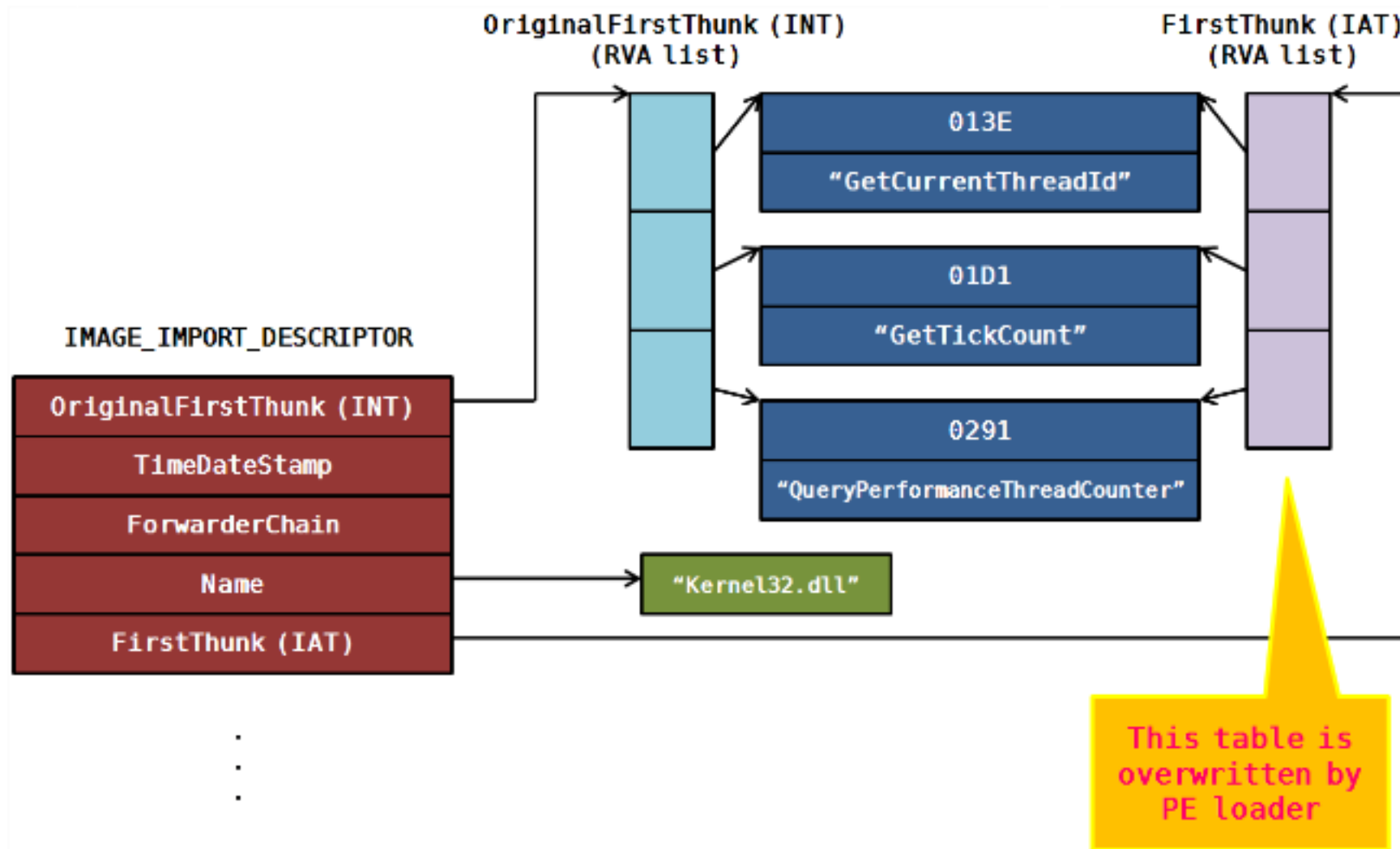
- Import Address Table
 - 어떤 라이브러리에서 어떤 함수를 사용하는 지 기술하는 테이블
 - DLL을 통해서 함수를 불러와야 하는 데, 이에 대한 정보를 기술하고 있는 테이블
- Import Directory
 - (a.k.a IMAGE_IMPORT_DESCRIPTOR)
 - 자세히 그 라이브러리에서 어떤 함수를 호출하는 지에 대한 정보가 있음.

test.exe						
Module Name	Imports	OFTs	TimeStamp	ForwarderChain	Name RVA	FTs (IAT)
00004F34	N/A	00004A00	00004A04	00004A08	00004A0C	00004A10
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	18	00008050	00000000	00000000	00008534	00008130
msvcrt.dll	2	0000809C	00000000	00000000	0000854C	0000817C
msvcrt.dll	33	000080A8	00000000	00000000	000085DC	00008188

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
00008210	00008210	00D0	DeleteCriticalSection
00008228	00008228	00ED	EnterCriticalSection
00008240	00008240	0118	ExitProcess
0000824E	0000824E	012D	FindClose
0000825A	0000825A	0131	FindFirstFileA
0000826C	0000826C	0142	FindNextFileA
0000827C	0000827C	0161	FreeLibrary
0000828A	0000828A	0185	GetCommandLineA
0000829C	0000829C	01FF	GetLastError
000082AC	000082AC	0212	GetModuleHandleA
000082C0	000082C0	0242	GetProcAddress
000082D2	000082D2	02DF	InitializeCriticalSection

바이너리 분석

PE header(IAT/EAT)



바이너리 분석

PE header(IAT/EAT)

Bound Import Directory Size	00000154	Dword	00000000	
Import Address Table Directory...	00000158	Dword	00008130	.idata
Import Address Table Directory...	0000015C	Dword	000000E0	
Delay Import Directory RVA	00000160	Dword	00000000	

test.exe						
Module Name	Imports	OFs	TimeStamp	ForwarderChain	Name RVA	FTs (IAT)
00004F34	N/A	00004A00	00004A04	00004A08	00004A0C	00004A10
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	18	00008050	00000000	00000000	00008534	00008130
msvcrt.dll	2	0000809C	00000000	00000000	0000854C	0000817C
msvcrt.dll	33	000080A8	00000000	00000000	000085DC	00008188

```
.bss:00407FFF
.idata:00408130
.idata:00408130 Imports from KERNEL32.dll
.idata:00408130
.idata:00408130 Section 6. (virtual address 00008000)
.idata:00408130 Virtual size : 000005E8 ( 1512.)
.idata:00408130 Section size in file : 00000600 ( 1536.)
.idata:00408130 Offset to raw data for section: 00004A00
.idata:00408130 Flags C0300040: Data Readable Writable
.idata:00408130 Alignment : 4 bytes
.idata:00408130
.idata:00408130 Segment type: Externs
.idata:00408130 .idata
.idata:00408130 void (__stdcall *DeleteCriticalSection)(LPCRITICAL_SECTION lpCriticalSection)
.idata:00408130 extrn __imp_DeleteCriticalSection@4:dword
.idata:00408130 ; DATA XREF: DeleteCriticalSection(x)fr
.idata:00408134 void (__stdcall *EnterCriticalSection)(LPCRITICAL_SECTION lpCriticalSection)
.idata:00408134 extrn __imp_EnterCriticalSection@4:dword
.idata:00408134 ; DATA XREF: EnterCriticalSection(x)fr
.idata:00408138 void (__stdcall __noreturn *ExitProcess)(UINT uExitCode)
.idata:00408138 extrn __imp_ExitProcess@4:dword
.idata:00408138 ; DATA XREF: ExitProcess(x)fr
.idata:0040813C BOOL (__stdcall *FindClose)(HANDLE hFindFile)
.idata:0040813C extrn __imp_FindClose@4:dword
.idata:0040813C ; DATA XREF: FindClose(x)fr
.idata:00408140 HANDLE (__stdcall *FindFirstFileA)(LPCSTR lpFileName, LPWIN32_FIND_DATAA lpFindFileData)
.idata:00408140 extrn __imp_FindFirstFileA@8:dword
.idata:00408140 ; DATA XREF: FindFirstFileA(x,x)fr
.idata:00408144 BOOL (__stdcall *FindNextFileA)(HANDLE hFindFile, LPWIN32_FIND_DATAA lpFindFileData)
.idata:00408144 extrn __imp_FindNextFileA@8:dword
.idata:00408144 ; DATA XREF: FindNextFileA(x,x)fr
.idata:00408148 BOOL (__stdcall *FreeLibrary)(HMODULE hLibModule)
.idata:00408148 extrn __imp_FreeLibrary@4:dword
.idata:00408148 ; DATA XREF: FreeLibrary(x)fr
.idata:0040814C LPSTR (__stdcall *GetCommandLineA)()
.idata:0040814C extrn __imp_GetCommandLineA@0:dword
.idata:0040814C ; DATA XREF: GetCommandLineA()fr
.idata:00408150 DWORD (__stdcall *GetLastError)()
.idata:00408150 extrn __imp_GetLastError@0:dword
.idata:00408150 ; DATA XREF: GetLastError()fr
.idata:00408154 HMODULE (__stdcall *GetModuleHandleA)(LPCSTR lpModuleName)
.idata:00408154 extrn __imp_GetModuleHandleA@4:dword
.idata:00408154 ; DATA XREF: GetModuleHandleA(x)fr
.idata:00408158 FARPROC (__stdcall *GetProcAddress)(HMODULE hModule, LPCSTR lpProcName)
.idata:00408158 extrn __imp_GetProcAddress@8:dword
.idata:00408158 ; DATA XREF: GetProcAddress(x,x)fr
.idata:0040815C void (__stdcall *InitializeCriticalSection)(LPCRITICAL_SECTION lpCriticalSection)
.idata:0040815C extrn __imp_InitializeCriticalSection@4:dword
.idata:0040815C ; DATA XREF: InitializeCriticalSection(x)fr

00004A00 00408130: .idata:DeleteCriticalSection: (Synchronized with Hex View-1)
```

바이너리 분석

■ PE header(IAT/EAT)

- HxD에서 직접 확인

바이너리 분석

PE header(IAT/EAT)

- Export Address Table
 - 라이브러리 파일에서 제공하는 함수를
다른 프로그램에서 사용할 수 있도록 정
보를 기술하는 테이블

```
typedef struct _IMAGE_EXPORT_DIRECTORY {
    DWORD    Characteristics;
    DWORD    TimeDateStamp;           // creation time date stamp
    WORD     MajorVersion;
    WORD     MinorVersion;
    DWORD    Name;                   // address of library file name
    DWORD    Base;                   // ordinal base
    DWORD    NumberOfFunctions;      // number of functions
    DWORD    NumberOfNames;          // number of names
    DWORD    AddressOfFunctions;     // address of function start address array
    DWORD    AddressOfNames;         // address of function name string array
    DWORD    AddressOfNameOrdinals;  // address of ordinal array
} IMAGE_EXPORT_DIRECTORY, *PIMAGE_EXPORT_DIRECTORY;
```

바이너리 분석

PE header(IAT/EAT)

- Export Address Table
 - 라이브러리 파일에서 제공하는 함수를 다른 프로그램에서 사용할 수 있도록 정보를 기술하는 테이블
- NumberOfFunctions
- NumberOfNames
- AddressOfFunctions
- AddressOfNames
- AddressOfOrdinals

test.dll				
Member	Offset	Size	Value	
Characteristics	00001C00	Dword	00000000	
TimeStamp	00001C04	Dword	617B8FEB	
MajorVersion	00001C08	Word	0000	
MinorVersion	00001C0A	Word	0000	
Name	00001C0C	Dword	00006032	
Base	00001C10	Dword	00000001	
NumberOfFunctions	00001C14	Dword	00000001	
NumberOfNames	00001C18	Dword	00000001	
AddressOfFunctions	00001C1C	Dword	00006028	
AddressOfNames	00001C20	Dword	0000602C	
AddressOfNameOrdinals	00001C24	Dword	00006030	

Ordinal	Function RVA	Name Ordinal	Name RVA	Name
(nFunctions)	Dword	Word	Dword	szAnsi
00000001	00001280	0000	0000603B	func

바이너리 분석

PE header(IAT/EAT)

Instruction

Data

Unexplored

External symbol

Lumina function

IDA View-A





Pseudocode-A

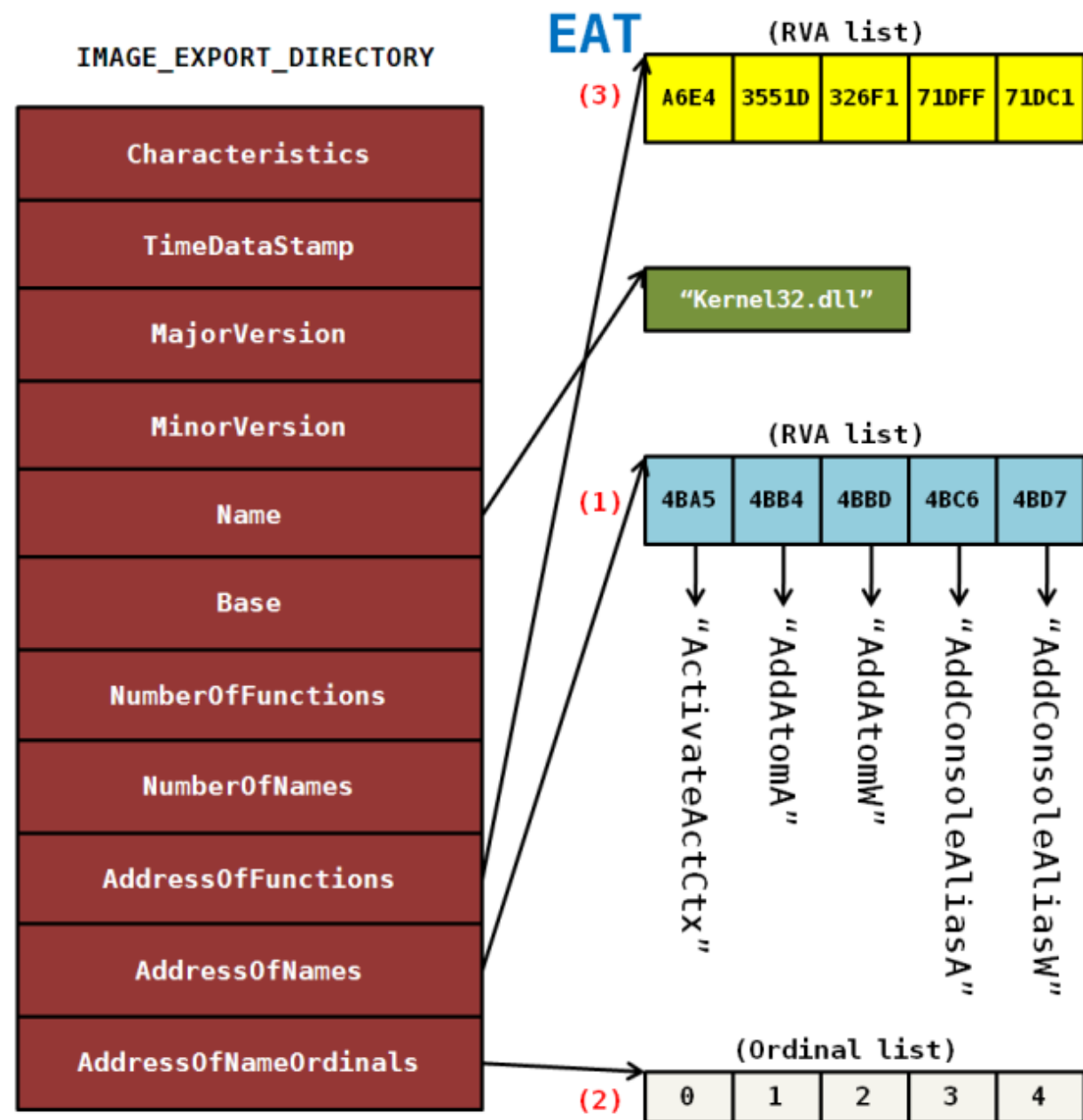
Hex View-1

Structures

Enums

In

Name	Address	Ordinal
 func	6C0C1280	1
 TlsCallback_0	6C0C13B0	
 TlsCallback_1	6C0C1360	
 DllMainCRTStartup(x,x,x)	6C0C1060	[main entry]



바이너리 분석

■ PE header(IAT/EAT)

- HxD에서 직접 확인

실습

Kaboom!

garbage

실습

Kaboom!

실습

■ Flare-on7 garbage

읽어볼만한 책

- <http://www.yes24.com/Product/Goods/7529742?OzSrank=1>
- <http://www.yes24.com/Product/Goods/84767639>
- <http://www.yes24.com/Product/Goods/47245069?OzSrank=1>
- <http://www.yes24.com/Product/Goods/31091391>
- <http://www.yes24.com/Product/Goods/97964411>
- 홍보아닙니다.. :)..

Reference

- 리버싱 핵심원리, <https://reversecore.com>
- <https://docs.microsoft.com/ko-kr/windows/win32/debug/pe-format>
- Flare-on 7 garbage
- <https://github.com/Insomnihack/Teaser-2020/tree/master/kaboom>