######################################################################

# #

# 본 문서는 퍼블릭 도메인 문서로, 저작권이 없습니다. #

# 따라서, 누구든지 이 문서로 원하는 것을 할 수 있습니다 #

# 또한 본 문서는 지식의 공유를 위한 문서이며 #

# 이 문서의 내용을 통해 행동한 것의 책임은 모두 독자 본인 에게 있습니다. #

# #

######################################################################

**System Hacking**

**Basic – 2**

****

@@@@@@@@@@@@@@@\*\*^^""~~~"^@@^\*@\*@@\*\*@@@@@@@@@@@@@@@@@@

작성일자 : 2016 년 2월 17일

작성자: GhostHat

목차

1. 윈도우 실행 파일 구조

1-1) PE 파일

1-2) Image Dos Header와 DOS Stub

1-3) Image NT Headers

1-4) Section Header

윈도우 실행 파일 구조

1) PE 파일

= > PE 파일 ? 윈도 실행 파일에 붙어있는 정보를 PE 파일이라 부르며

PE는 커다란 구조체로 구성되어 있고, 내부에서는 수많은 테이블과 멤버가 있다.

대략적으로 PE파일 종류는 아래와 같다.

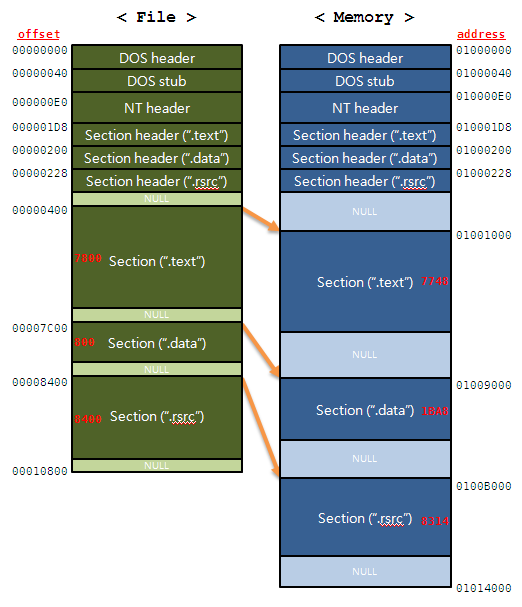


[1] PE 파일의 종류

PE 파일은 파일에 존재할 때 구조와 메모리에 로딩된 후의 모습이 달라지는게 특징이다.

파일에서는 첫 바이트로부터의 거리를 뜻하는 offset을 사용하고,

메모리에서는 VA(Virtual Address)와 RVA(Realative Virtual Address)를 사용한다.



[2] 파일 상태와 메모리에 로드된 상태의 PE 파일

출처: <http://www.reversecore.com/18> = > Reverse Core의 PE 헤더 부분

고정주소 대신 상대주소를 사용하는 이유는 PE 파일이 메모리에 로드될 때

한 주소에만 고정적으로 로딩되는 것이 아니기 때문이다.

Win7 부터는 Rebase 기능이 설정되어 있는 DLL이 대부분이므로 DLL들이

임의의 주소에 로드되는 경우가 많다.

또한 메모리에 로드된 후에 조금 메모리의 사이즈가 커진 것을 볼 수 있는데,

File의 Alignment 값보다 Section Alignment 값이 더 크기 때문이다.

(Alignment:여러 가지 내부 연산 등 처리상의 효율성을 위해 특정 단위로 간격을 맞춰주는

일을 의미, PE에서 남는 공간은 NULL BYTE로 채워주게 된다.

때로는 섹션이 파일 상태에서는 사이즈가 0이지만 메모리에 로드된 후에는

용량이 커지는 경우도 존재한다.)

간단한 설명을 했으니 이제는 PE 파일을 보도록 하자.

1-2) Image Dos Header와 DOS Stub

PE 포맷의 시작 부분에 위치한 40바이트의 구조체인 DOS Header와 Stub는

Windows가 아닌 DOS OS를 위한 것이다.

DOS에서 PE 파일이 실행되는 경우를 위해 만들어진 헤더다.



[3] winnt.h File 내부

보기에도 너무 많다 하지만 저기선 실질적으로 사용되는 멤버만 보도록 하자.

e\_magic과 e\_lfanew만 보도록 하자.

E\_magic 멤버는 해당 파일이 PE 파일임을 나타내는 첫 2개의 바이트 MZ(54AD)로 고정된

값을 가지고 있다.

즉 파일의 첫 2바이트가 MZ가 아니면 PE 파일로 인식하지 않는다.

마지막의 e\_lfanew를 보자.

이 멤버는 NT Header가 시작되는 위치의 OffSet을 나타낸다.

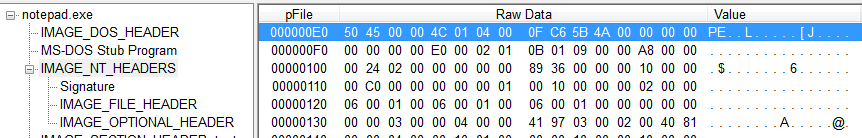
Notepad.exe에서는 사진과 같이 0xE0 값을 가지고 있다.

실제 0xE0 위치에는 NT Header가 있음을 알 수 있다.

1-3) Image NT Headers

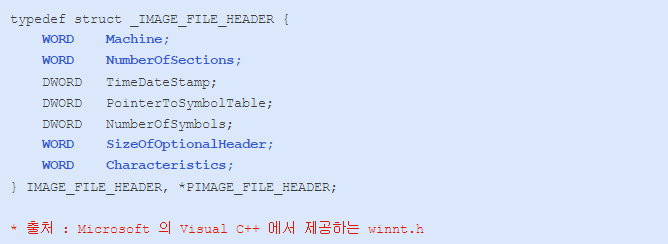
Image NT Headers 헤더임을 나타내는 시그니처인 “P E 0 0” 4 바이트를 시작으로

FileHeader와 OptionalHeader를 멤버로 가지는 구조체다.



[4] IMAGE NT Headers 모습

FileHeader를 살펴보면 구조체는 아래와 같이 정의되어 있다.



[5] FileHeader의 모습

중요한건 파란색이다. Machinke는 이 파일의 실행 대상 플랫폼을 나타낸다.

Machinke을 나타내는 상수값 역시 WinNT.h에 정의되어 있다.

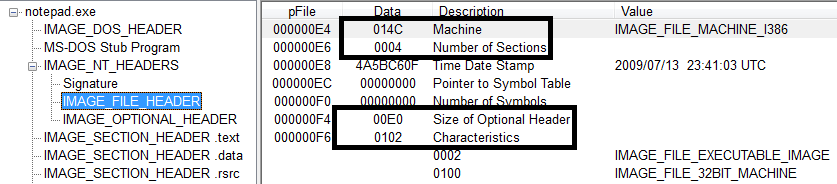
다음으로 NumberOfSections는 파일에 존재하는 섹션의 수를 말한다.

섹션은 파일에 따라 개수가 달라질 수 있기 때문에 이를 알려주기 위함이다.

SizeOfOptionalHeader는 바로 이어지는 Optional Header의 크기를 알려준다.

마지막으로 Characteristics는 이 PE 파일의 특성을 알려주며 이 부분을 통해 DLL인지,

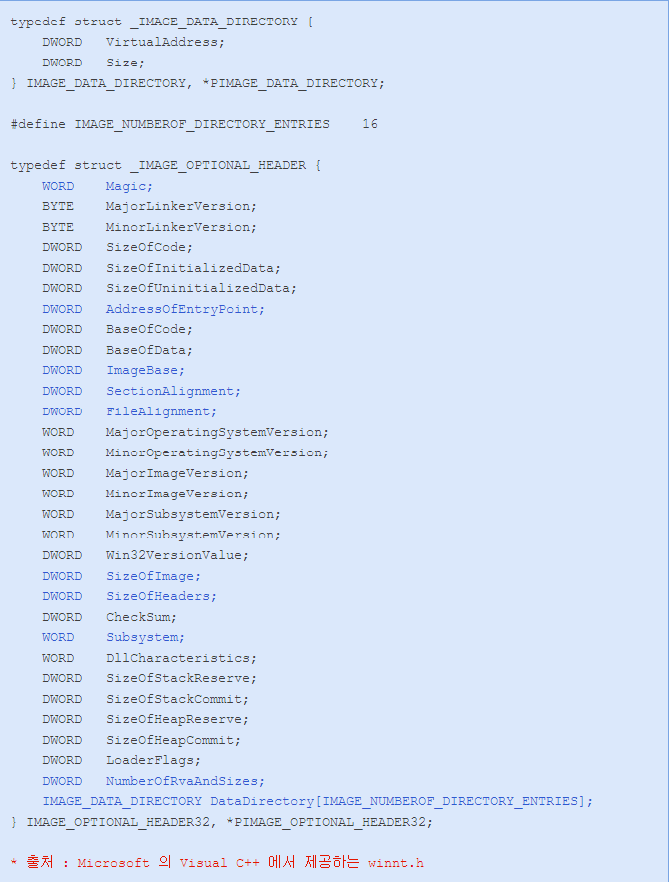
실행파일인지 알 수 있다.



[6] Characteristics Field

PE View로 보면 위와 같은 Characteristics Field를 볼 수 있다.

이제 가장 멤버가 많고 중요한 구조체인 Image Optional Headers를 보자.



[7] Image Optional Headers

중요한 것은 마찬가지로 파란색으로 표시했다.

Magic은 Image\_Optional\_Header32인지 64인지를 구분하게 해 준다.



[8] Magic Field

Address of EntryPoint는 파일이 메모리에 메핑된 후 코드 시작 주소를 나타낸다.

PE 로뎌는 ImageBase 값에 이 값을 더해서 코드 시작 지점을 설정해 준다.



[9] 코드 시작 지점을 나타내는 엔트리 포인트

SectionAlignment와 FileAlignment는 각각 메모리, 파일 상태에서의 정렬값으로 ,

섹션에서 크기가 남더라도 0으로 채워 Alignment 값을 맞춘다.

각 섹션은 반드시 Alignment 값의 배수여야 한다.



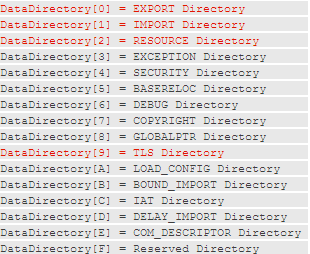
[10] Alignment 정렬

SubSystem은 동작환경을 정의한 부분이다.

Sys 파일은 0x1, GUI 프로그램인 경우 0x2, CLI 프로그램 경우 0x3을 가진다.

NumberOfRvaAndSize를 통해 디렉토리 수를 정해줄 수 있으나 일반적으로

아래와 같은 정형화된 16개 디렉토리를 가진다.



[11] Data Directory 구조체

출처: <http://www.reversecore.com/21> = > ReverseCore

먼저 Export Directory는 DLL 등 파일에서 외부 함수를 공개하기 위한

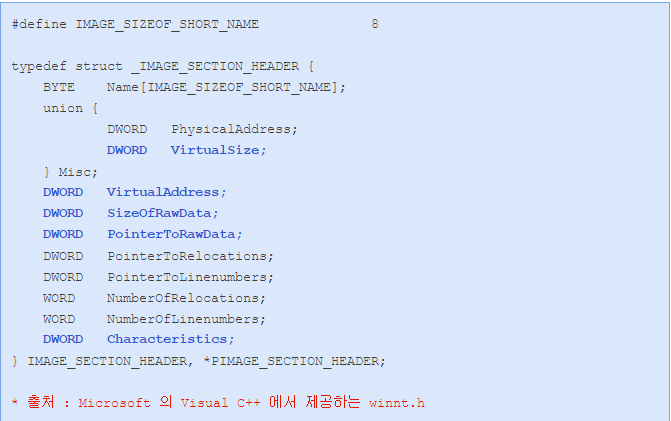
정보를 가지고 있다.

Import Directory는 프로그램 실행을 위해 Import하는 DLL 이름과

사용할 함수 정보가 담겨진 INT,IAT 주소 등의 정보가 있다.

1-4) Section Header

섹션은 실제 파일의 내용들이 존재하는 부분이다.



[12] Section Header의 모습

VirtualSize와 VirtualAddress는 메모리상에서의 크기와 주소,

SizeOfRawData와 PointerToRawData는 파일 상에서의 크기와 OffSet이다.

환경: Windows7 32 bit

테스트 대상: C:\Windows\System32\notepad.exe

테스트 도구: PE View – <http://wjradburm.com/software>

Immunity Debugger