# 악성코드 탐지, 암호화 및 패키징 프로젝트

PE파일에 대한 보고서



**목**

**차**

1. **PE 파일 소개**
2. **PE 파일의 구조**
   1. **DOS 헤더 (MZ헤더)**
   2. **PE헤더 (NT 헤더)**
   3. **이미지 파일 및 선택 헤더**
   4. **섹션 헤더 (또는 섹션 테이블)**
3. **PE 파일의 주요 구성 요소**
4. **참고 문헌**

## 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,ARM등에서 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.

## PE 파일의 구조

## PE의 구조

## 

## PE 구조

## 

## [PE파일(메모장.exe)이 메모리에 로딩되는 모습]

## PE header: DOS header ~ Section header

## PE Body: 그밑 Section

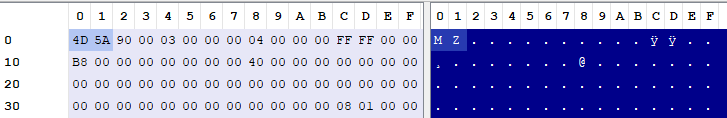
## 

## [VA(Virtual Address, 절대주소)]

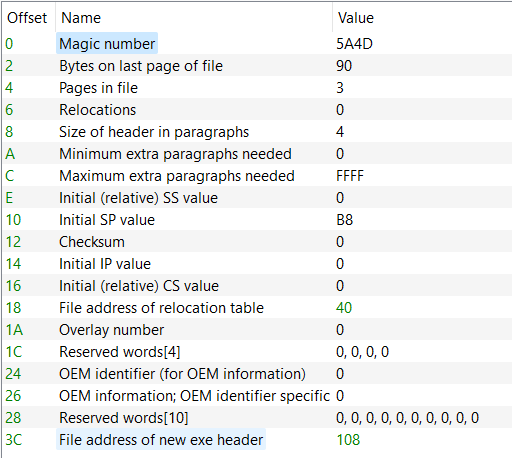
## 2.1. DOS 헤더 (MZ 헤더)

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.

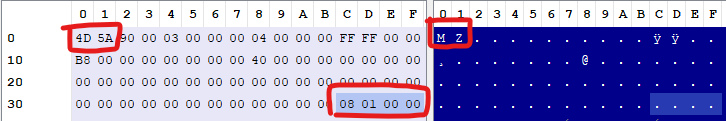
실행 파일(.exe)을 16진수 편집기로 열어보면 가장 먼저 MZ 헤더가 보입니다. 이는 운영 체제에서 유효한 실행 파일인지 확인하는 서명(또는 식별자)으로 사용됩니다. PE 헤더의 "PE" 문자열과 함께 파일의 앞부분에 위치합니다.



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



Hex Editor로 자세히 살펴보면 파일 시작 부분에 Hex 형식으로 정보가 표시되어 있습니다. 특히 마지막 값인 "새 exe 헤더의 파일 주소" - 108(HEX)에 주목해 보세요.



표에서 "매직 넘버" 0x5A4D는 "MZ" 문자열의 16진수 값과 정확히 일치합니다(16진수 편집기에서는 바이트 순서가 역순으로 표시됨, 예: 0x5A -> 0x4D 0x5A).

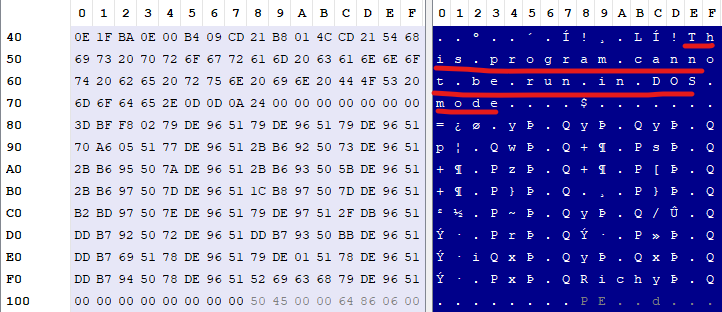
또한 0x108은 int값(4바이트, 32비트)으로 역순 설정됩니다: 0x08 0x01 0x00 0x00.

이는 16진수 편집기가 낮은 주소에서 높은 주소로 16진수 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 DOS에서 실행될 경우 "이 프로그램은 DOS 모드에서 실행할 수 없습니다"라는 텍스트를 출력하고 종료하는 것입니다.

DOS Stub은 MZ 헤더 바로 뒤에 위치합니다.



## 2.2. PE 헤더(또는 NT헤더)

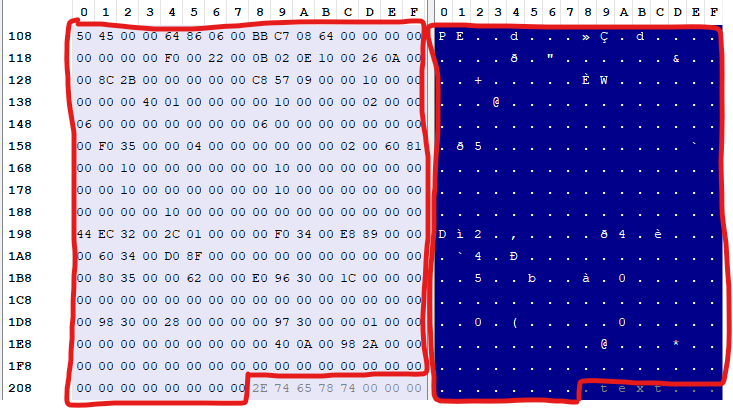
PE 헤더는 MZ 헤더 다음에 위치하며, 현대 Windows 실행 파일의 시작점입니다. 주요 구성 요소는 다음과 같습니다:

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.

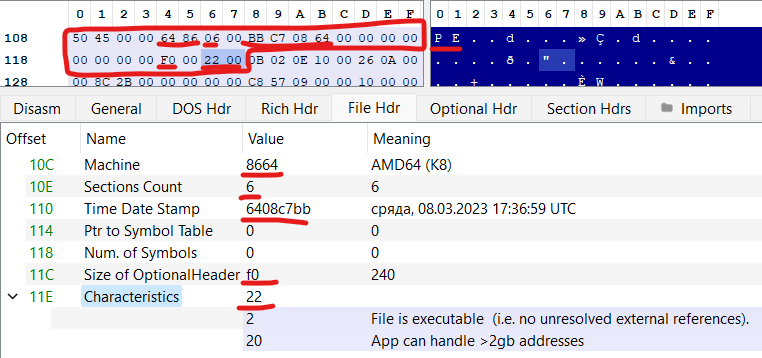


## 자세히 살펴보면 이미지 파일 헤더와 이미지 선택 헤더라는

## 하위 섹션으로 나뉩니다.

## 2.3. 이미지 파일 및 선택 헤더

이미지 파일 헤더의 개요는 다음과 같습니다:

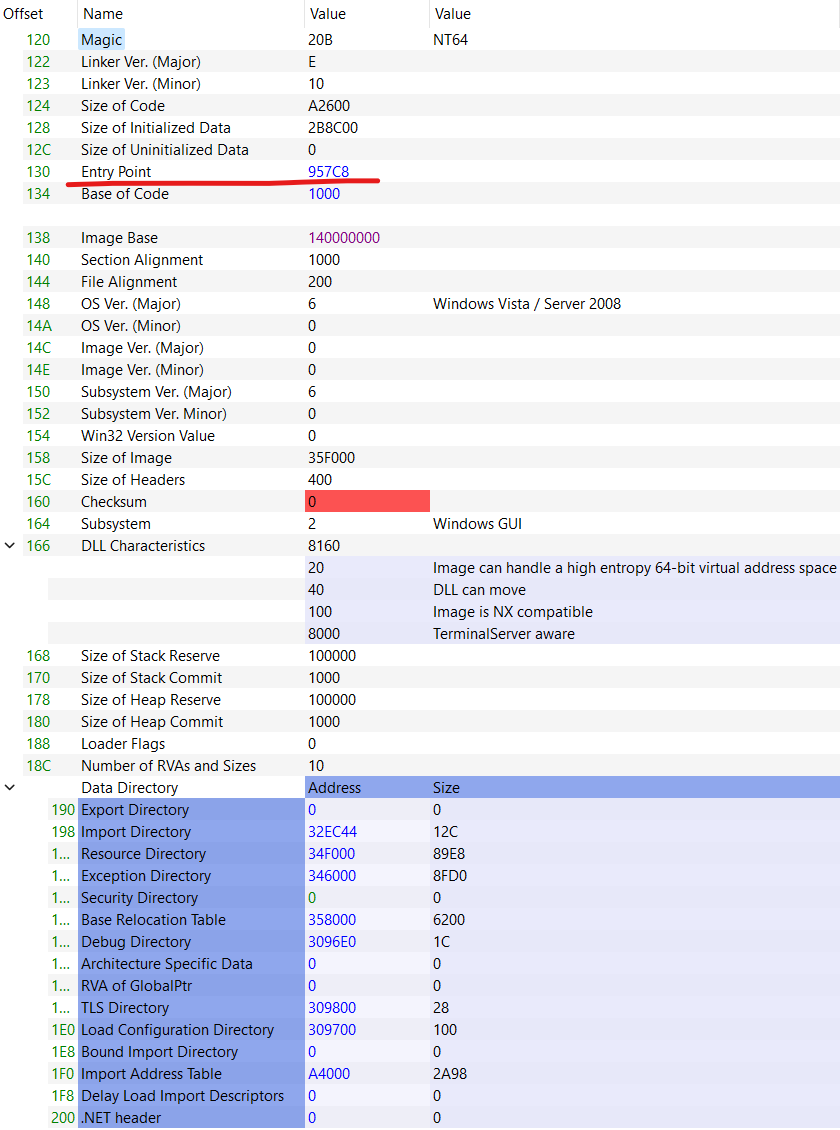


여기에는 실행 파일이 컴파일된 대상 머신 정보, 섹션 수, 타임스탬프 등이 포함되어 있습니다.

그 다음은 이미지 선택 헤더입니다. 이 헤더에는 이미지 파일 헤더보다 훨씬 더 많은 정보가 들어 있습니다.

특히 "Entry Point"라고 표시된 줄에 주목해 보세요. 이는 프로그램 실행 코드의 진입점으로, 사실상 애플리케이션의 "main()" 함수입니다. Windows 로더가 실행을 시작하는 지점입니다.

이 값 0x957C8은 RVA(상대 가상 주소)로, 실제 파일 오프셋이 아닙니다. Windows는 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 Windows가 각 프로세스에 대해 물리적 메모리(RAM)와 독립적인 고유한 VA 공간을 생성하기 때문입니다.

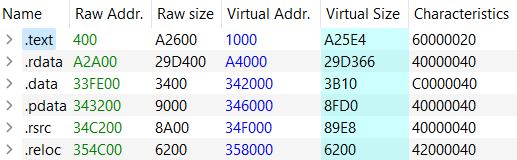
즉, 애플리케이션을 로드할 때 Windows는 메모리에 프로세스 공간을 할당한 다음 파일을 로드하고 확장합니다. 그런 다음 이 Entry Point 주소를라 메모리에서 코드 실행을 시작합니다.

## 2.4. 섹션 헤더(또는 섹션 테이블)

PE 헤더 이후 **섹션 테이블**은 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 API에 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.



여기에는 주로 파일의 각 섹션에 대한 주소, 크기, 특성이 포함됩니다.

바이러니 코드 주입을 수행하려면 새로운 섹션을 추가하거나 기존 섹션을

변경하여 크기를 변경해야 하므로 여기 값을 변경해야 합니다.

## PE 파일의 주요 구성 요소

## 1. 엔트리 포인트: 실행이 시작되는 주소로, 옵션 헤더에서 중요한 부분입니다.

## 2. 가져오기 주소 테이블(IAT): 프로그램이 외부 DLL에서 호출하는 함수 목록입니다. IAT는 실행 시점에 채워지며, 이를 통해 실행 파일이 동적으로 라이브러리와 연결됩니다.

## 3. 내보내기 테이블: PE 파일이 DLL인 경우, 다른 실행 파일에서 사용할 수 있는 함수나 데이터를 나열합니다.

## 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(23권 1호)

## 