# 악성코드 탐지, 암호화 및 패키징 프로젝트

PE파일에 대한 보고서



**Ⅱ. PE 파일의 구조**

**2.1. DOS 헤더 (MZ헤더)**

**2.2. PE 헤더 (NT 헤더)**

**2.3. 이미지 파일 및 선택 헤더**

**2.4. 섹션 헤더(또는 섹션 테이블)**

**Ⅰ. PE 파일 소개**

**목**

**차**

**Ⅵ. 참고 문헌**

**Ⅲ. PE 파일의 주요 구성 요소**

## Ⅰ. PE 파일 소개

Portable Executable(PE) 형식은 Windows 운영 체제에서 실행 파일(.exe), 객체 코드, 동적 링크 라이브러리(DLL)을 지원하는 핵심 파일 형식입니다. PE 파일은 컴퓨터 시스템이 프로그램을 메모리에 로드하고 실행하는 데 필요한 정보를 포함하고 있으며, 각종 아키텍처(예: x86, x64, ARM)에서 작동할 수 있도록 설계되었습니다. 이 형식은 Windows의 초기 버전인 MS-DOS 실행 파일 형식을 확장하여 만들어졌으며, 현대적인 Windows 시스템과 DOS 간의 호환성을 제공하는 중요한 역할을 합니다.

PE 형식은 유연성이 뛰어난 것이 특징입니다. 코드, 데이터, 리소스 파일 등이 개별 섹션에 저장되며, 이를 통해 운영 체제는 필요한 정보만을 메모리에 효율적으로 로드할 수 있습니다. 이러한 구조는 프로그램의 보안과 성능을 최적화할 수 있도록 지원합니다. 예를 들어, .text 섹션은 실행할 코드만을 포함하고, .data 섹션은 초기화된 변수를 저장합니다. 이로 인해 Windows는 불필요한 메모리 사용을 방지하고, 프로그램의 실행 속도와 보안을 유지합니다.

또한, PE 형식은 동적 링크 라이브러리(DLL) 파일을 관리하는 데도 중요한 역할을 합니다. DLL은 여러 프로그램에서 공용으로 사용될 수 있는 코드를 포함하고 있어, 시스템 자원의 낭비를 줄이고 유지 보수성을 높입니다. PE 파일은 이러한 DLL의 참조, 로드, 연결을 처리하는 메커니즘을 갖추고 있어 Windows 운영 체제의 효율적인 동작을 가능하게 합니다.

PE 파일 구조에는 **DOS 헤더**, **PE 헤더**, **섹션 헤더**, **데이터 디렉토리** 등이 포함되며, 이러한 정보는 실행 파일이 어떻게 로드되고 실행되는지를 결정합니다. PE 형식의 이러한 유연성 덕분에 다양한 보안 도구와 소프트웨어 보호 기법도 PE 파일을 기반으로 적용됩니다.

## Ⅱ. PE 파일의 구조

## PE의 구조

## 

## PE 구조

## 

## [PE파일(메모장.exe)이 메모리에 로딩되는 모습]

## PE header: DOS header ~ Section header

## PE Body: 그밑 Section

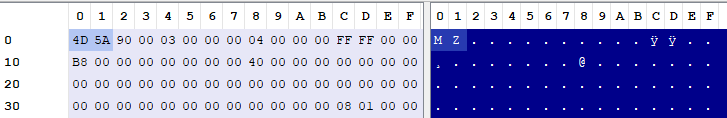
## 

## [VA(Virtual Address, 절대주소)]

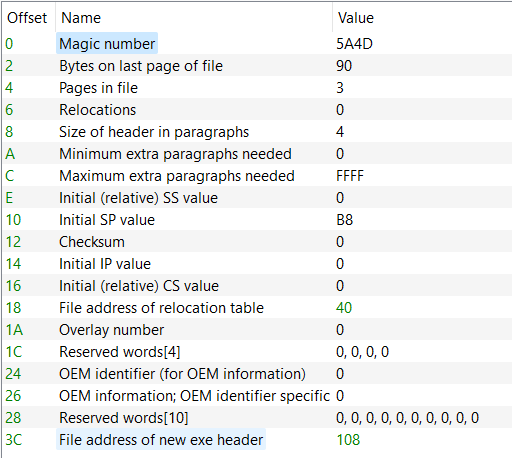
## 2.1. DOS 헤더 (MZ 헤더)

파일의 가장 첫 번째 부분인 **MZ 헤더**는 오래된 MS-DOS 형식의 흔적입니다. 이 부분에는 소규모 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 유지됩니다.

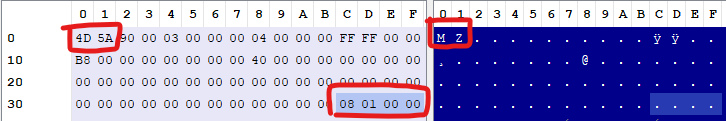
실행 파일(.exe)을 16진수 편집기로 열면 가장 먼저 MZ 헤더가 보입니다. 이것은 운영 체제에서 서명(또는 식별자)으로 사용되어 유효한 실행 파일인지 확인합니다(파일에서 약간 앞쪽에 있는 PE 헤더 주소의 “PE” 문자열과 함께).



하지만 그것은 단순히 “MZ” 서명이 아닙니다. 바로 뒤에 다음과 같은 추가 정보가 있습니다.



Hex Editor에서 자세히 살펴보면 파일의 시작 부분에 정보가 Hex형식으로 표시되어 있는 것을 볼 수 있습니다. 마지막 값인 “새 exe 헤더의 파일 주소” – 108(HEX)을 살펴보세요.



위 표의 “매직 넘버” 0x5A4D는 문자열 “MZ”의 16진수 값과 정확히 같습니다(16진수 편집기에서 역순으로 바이트 순서로, 예: 0x5A -> 0x4D 0x5A).

또한 0x108은 다시 역순으로 int값(4바이트,32비트)으로 설정됩니다

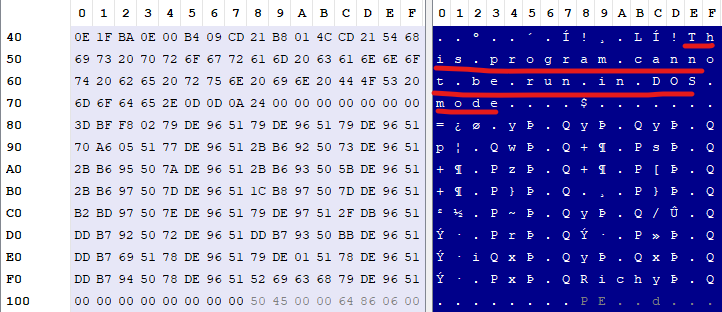
: 0x08 0x01 0x00 0x00.

이는 16진수 편집기가 낮은 주소에서 높은 주소로 16진수 바이트를 출력하고, 리틀 엔디어(예: x86/ia32)머신은 다중 바이트 엔터티의 낮은 자릿수를 낮은 주소에 저장하기 때문입니다.

**DOS Stub**

이것은 단 하나의 목적을 가진 작은 DOS프로그램입니다. 즉, 프로그램이 DOS에서 실행되는 경우 “이 프로그램은 DOS모드에서 실행할 수 없습니다”라는 텍스트를 인쇄하고 종료하는 것입니다.

DOS Stub은 MZ 헤더 바로 뒤에 있습니다.



## 2.2. PE 헤더(또는 NT헤더)

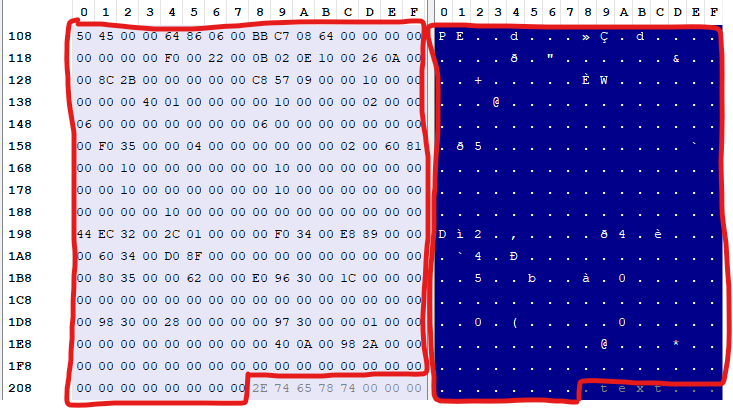
PE 헤더는 MZ 헤더 다음에 위치하며, 현대 Windows 실행 파일의 시작점입니다. 세부 사항은 다음과 같습니다:

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, 더 구체적으로 Windows로더가 프로그램을 로드하고 실행하는데 사용할 정보가 들어 있습니다.

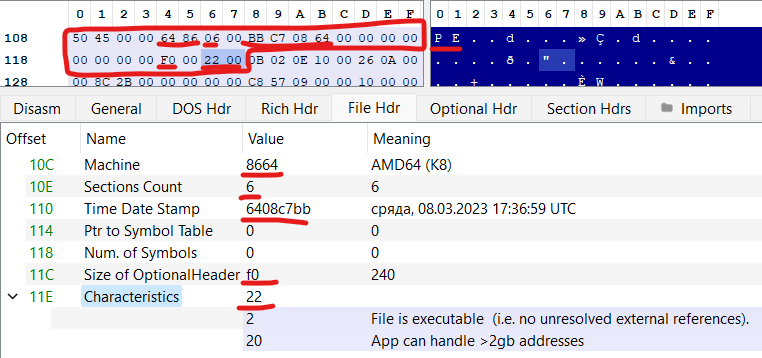
문자열 ”PE”로 시작하는데, 이는 또한 Portable Exectable파일이라는 서명이기도 합니다.



이를 자세히 살펴보면 이미지 파일 헤더와 이미지 선택 헤더라는 하위 섹션에 도달하게 됩니다.

## 2.3. 이미지 파일 및 선택 헤더

이미지 파일 헤더의 개요는 다음과 같습니다.



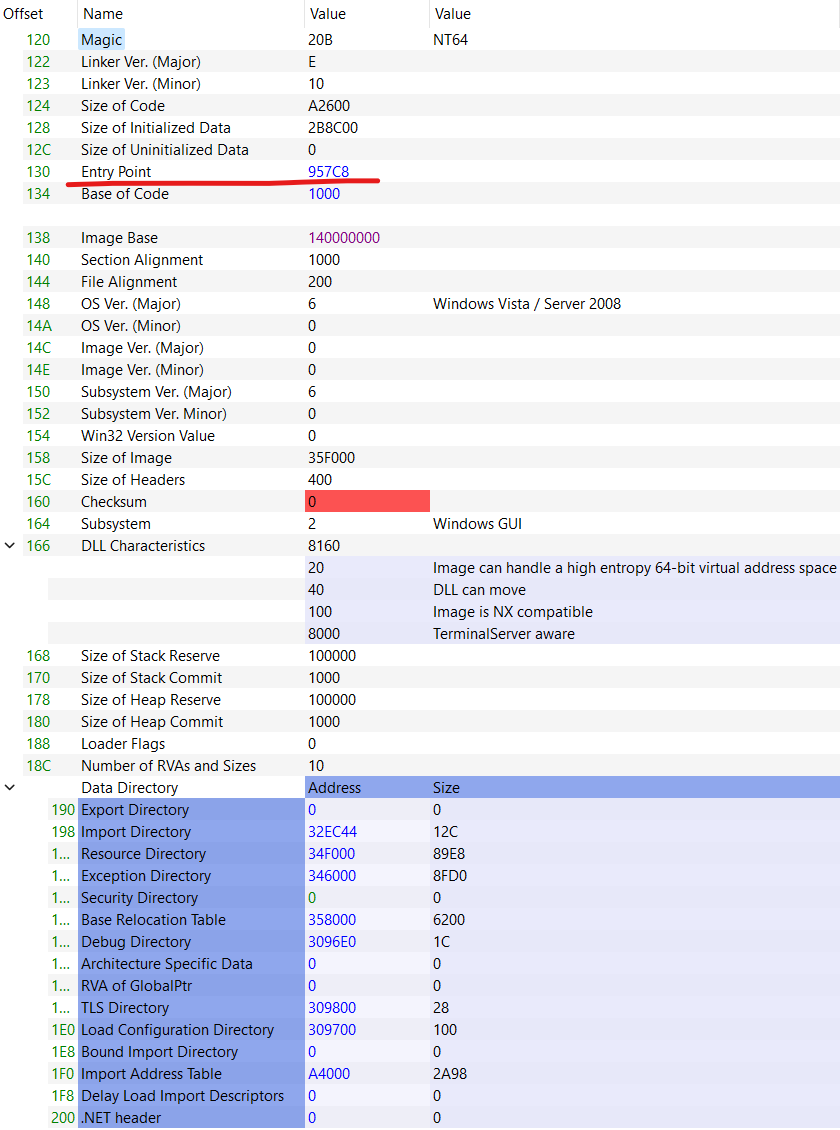
여기에는 실행 파일이 컴파일된 대상 머신에 대한 정보, 섹션 수, 시간 날짜 스탬프 등이 포함되어 있습니다.

그 다음 은 이미지 선택 헤더입니다. 여기에는 이미지 파일 헤더보다 훨씬 더 많은 정보가 들어 있습니다.

하지만 “Entry Point”이라고 하는 밑줄이 그어진 줄을 주목하세요. 이것은 프로그램 실행 코드의 Entry Point입니다. 사실상 애플리케이션의”main()”함수입니다. Windows 로더가 실행을 시작하는 곳입니다.

이 값 0x957C8은 실제 파일 오프셋인 RVA(상대 가상 주소)라는 점에 유의하세요. Windows는 Image Base값을 더하여 VA(가상 주소)로 변환하고 Windows Loader가 로드하고 매핑한 후 이 코드의 가상 메모리에서 주소를 가져옵니다. 이 주소는 Windows가 물리적 메모리(RAM)와 무관하게 각 프로세스에 대해 고유한 VA공간을 생성하기 때문에 “VA”라고 합니다.

즉, 애플리케이션을 로드할 때 Windows는 메모리에 프로세스 공간을 할당한 다음, 파일을 로드하고 확장합니다. 그런 다음 이 Entry Point 주소를 따라 메모리에서 코드를 실행하기 시작합니다.

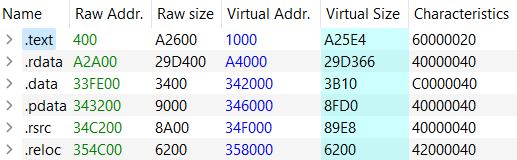


## 2.4. 섹션 헤더(또는 섹션 테이블)

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.

## Ⅲ. 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.

## Ⅳ. 참고 문헌

## “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)

## 