리버스 엔지니어링 바이블(강병탁 저) 05 PE 헤더(PE Header)에서 발췌
PE에 대한 개념
PE는 Portable Executable File Format의 약자
우리가 만든 이 파일(File)이 실행할 수 있는, 이식 가능한 다른 곳에 옮겨져도(Portable) 실행이 가능하도록(Executable) 만들어놓은 포맷(Format)

PE Header 안에는 실행파일을 실행하기 위한 각종 정보가 기록돼 있다. 실행 파일을 로딩할 때 해당 PE에 들어있는 정보를 토대로 DLL을 로드하고 메모리에 적재될 각종 리소스를 할당한다. 그래서 EXE, DLL을 실행하면 항상 개발자가 만든 코드가 실행되기 전에 PE의 정보부터 읽어와서 바이너리를 메모리에 올리기 위한 각종 데이터를 설정하는 작업을 하게 된다.

-> 이런 내용 참고해서 프로젝트 목적을 'PE 구조를 분석해서 파일이 실행되기 위한 정보를 얻을 수 있음 / 리버싱에 필요한 PE 헤더 정보를 시각화해서 이용하기 위해 PE parser 제작' 정도로 쓰고
추가적으로 PE 관련 코드를 작성해보면서 PE구조에 대해 공부할 수 있으니 파서 만들면서 PE 공부를 하겠다는 걸 프로젝트 목표에 넣어도 괜찮을 것 같아요..?

프로젝트 내용은 일단 간단하게 써도될거같으니까 PE 파일에서 분석(리버싱)에 필요한 부분 선별 -> 파이썬으로개발(GUI까지구현)->테스트 후 보완 정도로??

아래는 구조체랑 필드 중요하다는거 추린건데 일단 리버싱할 때 써본 기억나는거 굵게표시했어요

PE 파일 구조
IMAGE_DOS_HEADER 구조체
e_magic : 현재 파일이 PE 파일인지 체크
e_lfanew : IMAGE_NT_HEADER 구조체 위치 확인

IMAGE_NT_HEADER 구조체
-> Signature, IMAGE_FILE_HEADER, IMAGE_OPTIONAL_HEADER
Signature : PE파일임을 표시. 별로 쓸 일 없음

IMAGE_FILE_HEADER 구조체
파일을 실행하기 위한 가장 기본적인 데이터가 담겨 있는 구조체
Machine : 어떤 cpu에서 이 파일이 실행될 수 있는지 확인(특별히중요하진않음)
NumberOfSections : 섹션 개수. (보통 비주얼 스튜디오에서 별다른 옵션 없이 빌드하면 4개의 섹션존재 .txt .rdata .data .rsrc 하므로 당연히 개수가 4가 됨. 이 값이 달라진다면 패킹이나 프로텍팅 때문이라고 추측 가능)
TimeDateStamp : 파일 빌드 날짜. (변조가능성있어서 신뢰할정보는아님, 델파이의 경우 항상 1992년)

IMAGE_OPTIONAL_HEADER
Magic : 32bit , 64bit 구분. 크게중요치않음
SizeOfCode : 코드 양 전체 크기
MajorLinkerVersion, MinorLinkerVersion : 어떤 버전의 컴파일러로 빌드했는지 확인
ImageBase : PE파일이 실제 메모리번지에매핑되는번지
AddressOfEntryPoint : 실제파일이 메모리에서 실행되는 시작지점
BaseOfCode : 실제코드가 실행되는 번지
SectionAlignment, FileAlignment : 각 섹션을 정렬하기 위한 저장단위(섹션 사이 간격)
Subsystem : 콘솔인지 gui인지 구분

IMAGE_SECTION_HEADER : 각 섹션 이름 시작주소 사이즈 등 정보 관리하는 구조체
보통섹션테이블형태로 파싱하는것같음