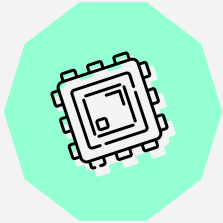
더 나은 PE Parser 제작 프로젝트

시스템 보안



KU Seiong 2022

Table of contents

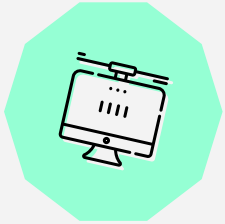
01 **Problem**
프로젝트의 필요성
타겟과 목표 설정

02 **Design**
기존 프로그램 분석
기획 및 설계

03 **Solution**
구현환경 구축
UI 적용

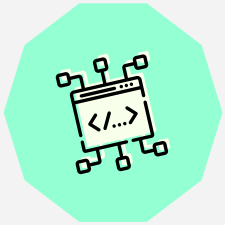
04 **Testing**
실제 사용성 테스트
향후 보완 계획

팀 구성안



프로젝트 문서 및 버전 공유는 Git, 개발 언어는 Python으로 함

박신영 ___ 2018271333	팀장, 분석 정보 선별 및 시각화 기획 화면 설계 및 프로토타입 디자인 발표자료 제작
김선우 ___ 2019270102	분석 정보 선별 및 시각화 기획 시스템 개발(Python) 버전 관리, GUI 적용
서혁 ___ 2020270121	분석 정보 선별 및 시각화 기획 테스팅 리서치, QA 발표자



01

Problem

: 리버싱을 어렵게 하는 요소

파일 분석을 위해 왜 **PE 구조**를 알아야 할까?

PE 분석은 왜 어려울까?

1 프로젝트의 필요성

✓ **파일 분석을 위해 PE 구조를 알아야 하는 이유**

Reversing

프로그램의 해부

실행 파일을 가지고 원래 **코드**를 추적하는 과정

프로그램의 실행 순서

1 EXE 실행

2 **PE 정보** 읽어오기

3 바이너리를 메모리에 올리기 위한 각종 데이터 설정

4 개발자의 코드 실행

PE 분석으로 알 수 있는 것

파일이 실행되기 위한 **모든 정보**

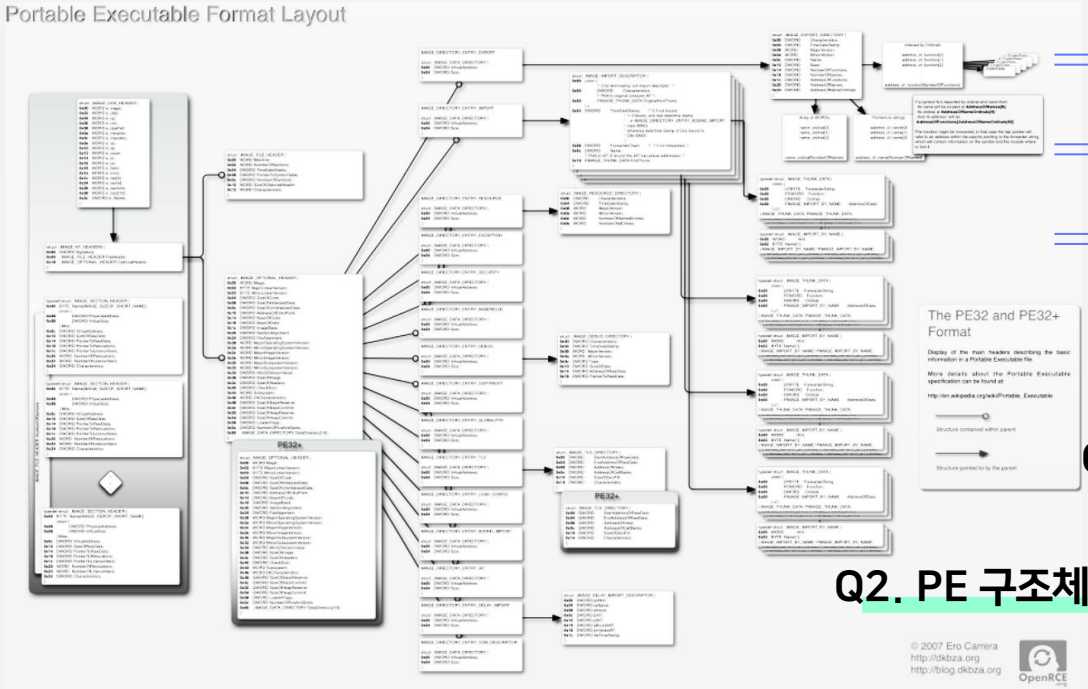
프로그램이 사용하는 **API 확인**

파일이 로딩되는 **메모리 주소**

기초 분석 지점과 메모리 해석을 위해 PE 구조 파악은 반드시 필요하다 !

1 프로젝트의 필요성

✓ **PE 분석의 어려움**



⟹ 한 눈에 파악하기 어려운 구조

⟹ 상대주소와 절대주소의 개념

⟹ 방대한 정보의 양

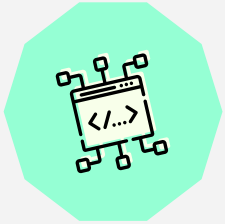
Q1. 정말 이걸 다 봐야 할까?

Q2. PE 구조체의 모든 요소를 알아야 할까?

원하는 **핵심 정보**를 간편하게 파악 가능하고

처음 보는 사람도 **쉽게 사용**할 수 있는 **분석 툴**을

만들 수 있지 않을까?



02

Design

: 새로운 **프로그램**의 설계

기존 프로그램을 사용하며 아쉬웠던 부분은?

정보의 **중요도**를 고려할 수 있을까?

1 PE 구조 알기

✓ **이론 및 기술적 배경 조사**

- 구체적인 타겟 설정과 시나리오 설계를 위해 PE 구조 학습을 우선으로 진행함

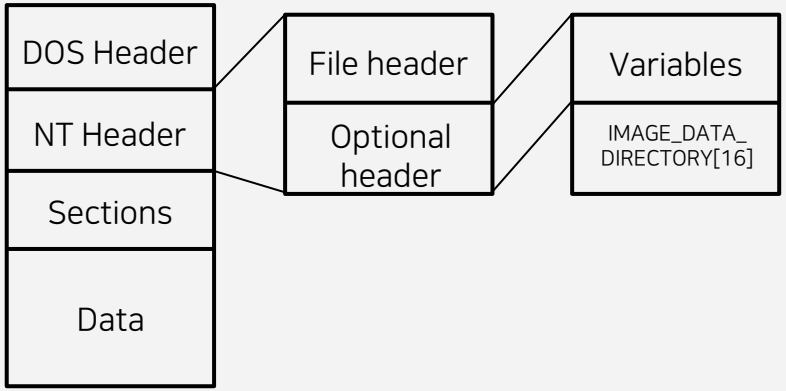
PE 구조란?

우리가 만든 이 파일이 File

실행할 수 있는, 이식 가능한 다른 곳에 옮겨져도 Portable

실행이 가능하도록 Executable

만들어 놓은 포맷 Format



실행계열 : EXE, SCR

드라이버계열 : SYS, VXD

라이브러리 계열 : DLL, OCX, CPL, DRV

오브젝트 계열 : OBJ

1 PE 구조 알기

✓ **PE 구조체의 넘버**

IMAGE_DOS_HEADER

e_magic : 시그니처 (MZ)

e_lfanew : NT 헤더를 가리킴

IMAGE_NT_HEADER

signature + 구조체 2개

IMAGE_FILE_HEADER

NumberofSections : 섹션 (.text, .rdata, .data, ...) 개수

TimeDateStamp : 빌드 완료 시각

SizeOfOptionalHeader : 뒤따라오는 Optional 헤더의 크기, 가변적

IMAGE_OPTIONAL_HEADER

변수들 + DataDirectory

SizeOfCode : 기계어가 담긴 코드 영역의 크기

AddressOfEntryPoint : 프로그램 로드 후 레지스터가 가리켜야 할 메모리 주소

BaseOfCode : 기계어가 탑재될 메모리 주소

ImageBase : 파일이 메모리에 탑재되는 지점 (ex. 0x401000)

*** Alignment = 섹션을 구분하기 위한 구역 정렬 단위, 일반적으로 0x1000

1 PE 구조 알기

✓ **각 섹션에 담긴 정보**

Padding

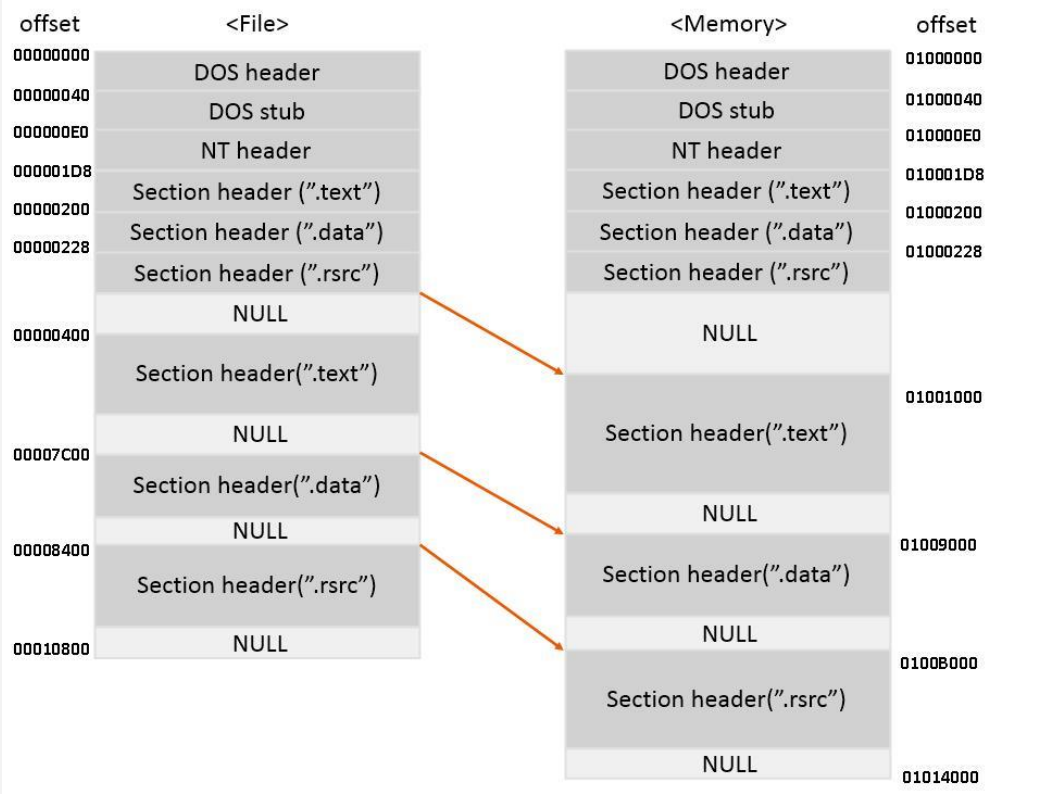
모든 메모리가 값으로 차 있지 않고 null인 부분 존재

구역을 두고 데이터를 다루어 오류 방지

코드가 있는 **.text** 섹션

전역 변수, 정적 변수를
포함하는 **.data** 섹션

문자열, 아이콘 등 리소스
데이터를 포함하는 **.rsrc** 섹션



1 PE 구조 알기

✓ **각 섹션에 담긴 정보**

Packing

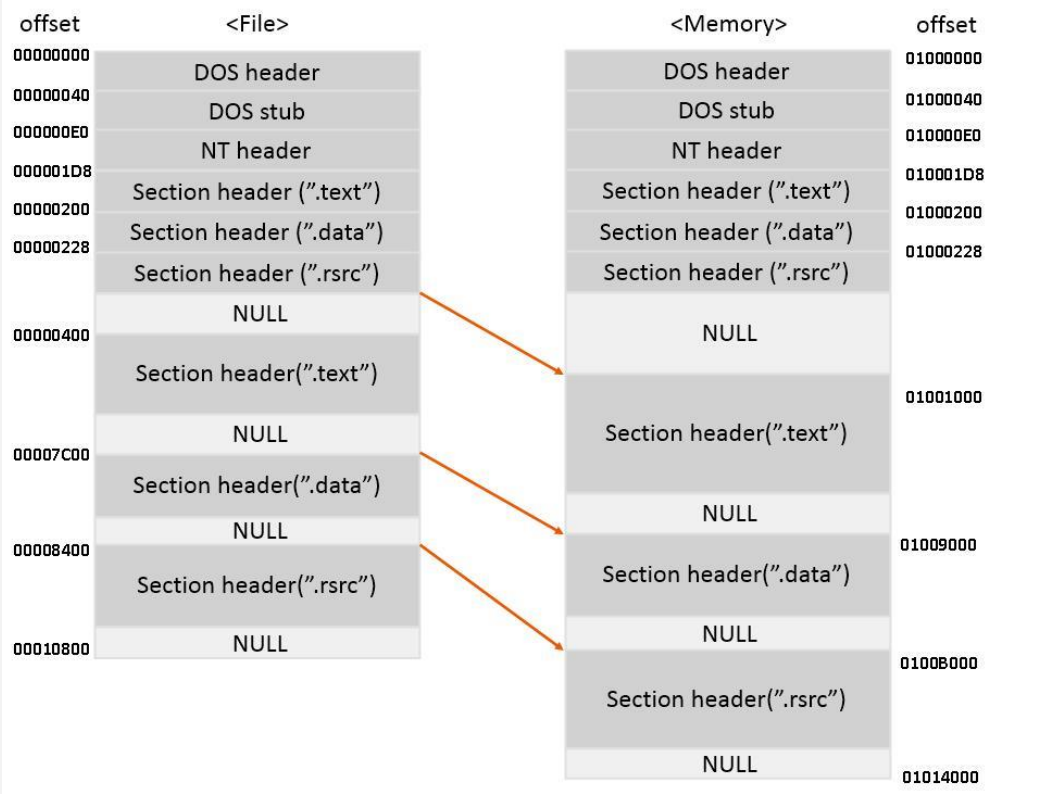
바이너리의 암호화/난독화

Unpack 과정을 거쳐야만 분석 가능

코드가 있는 ~~.text~~ 섹션
UPX0

전역 변수, 정적 변수를 포함하는 ~~.data~~ 섹션
UPX1

문자열, 아이콘 등 리소스 데이터를 포함하는 ~~.rsrc~~ 섹션
UPX2



1 PE 구조 알기

✓ **PE 파일과 메모리 이미지 간 주소 계산**

RAW

RVA = Address

offset	<File>
00000000	DOS header
00000040	DOS stub
000000E0	NT header
000001D8	Section header (".text")
00000200	Section header (".data")
00000228	Section header (".rsrc")

<Memory>	offset
DOS header	01000000
DOS stub	01000040
NT header	010000E0
Section header (".text")	010001D8
Section header (".data")	01000200
Section header (".rsrc")	01000228

- **파일→메모리로 갈 때** RVA = RAW - pointerToRawData + VirtualAddress
- **메모리→파일로 갈 때** RAW = RVA - VirtualAddress + PointerToRawData

파일에 있는 데이터가 메모리의 어디로
들어가는지를 결정하는 중요한 요소 4가지

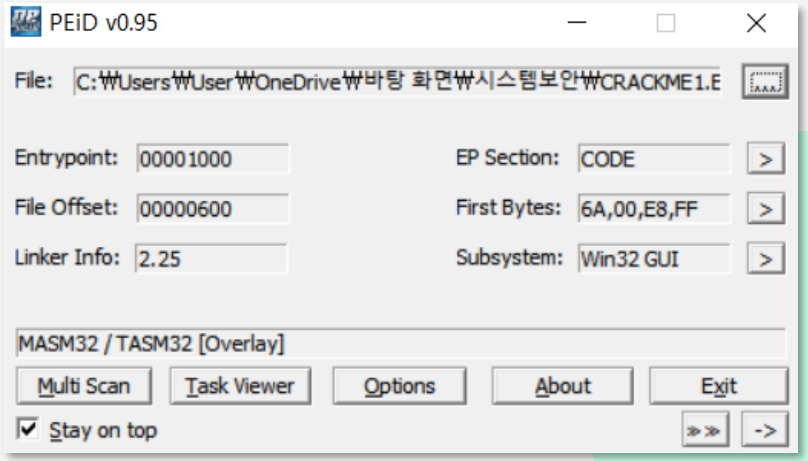
VirtualSize	메모리에서 섹션이 차지하는 크기
VirtualAddress	메모리에서 섹션의 시작 주소
SizeOfRawData	파일에서 섹션이 차지하는 크기
PointerToRawData	파일에서 섹션의 시작 오프셋

2 기존 프로그램 분석
: PEiD

시작화면의 정보들을 간단히 확인 가능

한 눈에 보는 Entrypoint

파일에 사용된 패커의 종류를 쉽게 파악

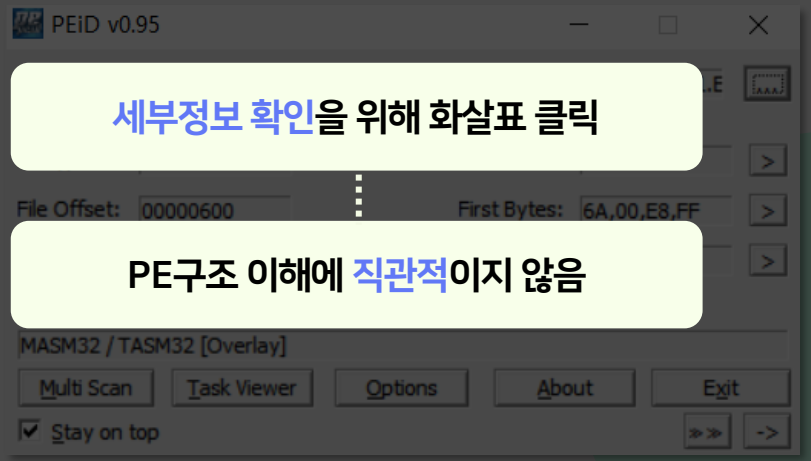


2 기존 프로그램 분석
: PEiD

시작화면의 정보들을 간단히 확인 가능

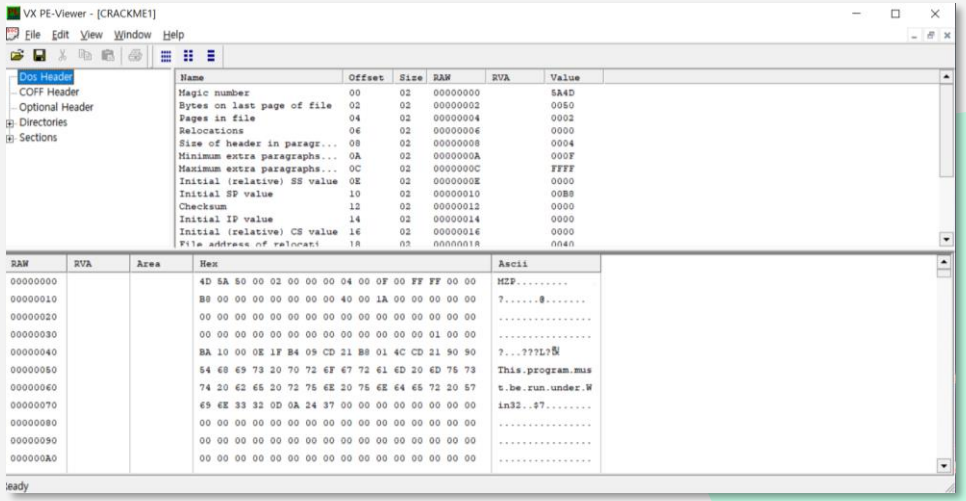
한 눈에 보는 Entrypoint

파일에 사용된 패커의 종류를 쉽게 파악



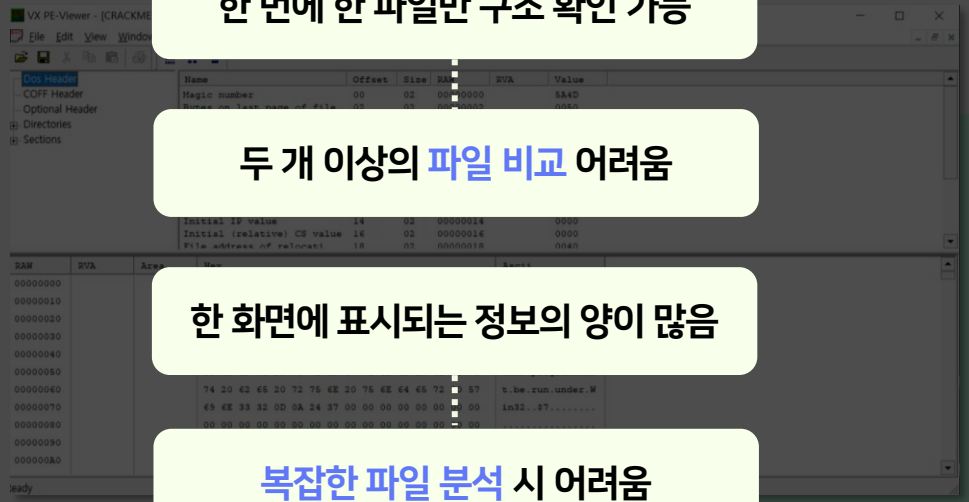
2 기존 프로그램 분석
: VX PE viewer

- 트리 구조로 헤더 파악 용이
- RVA 계산 결과 제공
- Hex/Ascii 값 직관적 표현
- 특정 부분 클릭 시 Hex 값 자동 스크롤



2 기존 프로그램 분석
: VX PE viewer

- 트리 구조로 헤더 파악 용이
- RVA 계산 결과 제공
- Hex/Ascii 값 직관적 표현
- 특정 부분 클릭 시 Hex 값 자동 스크롤



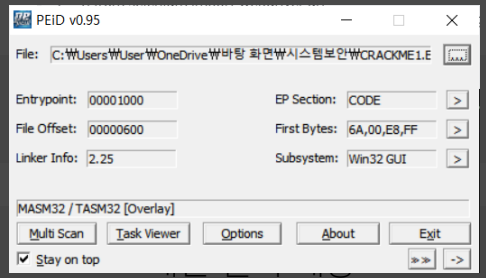
한 번에 한 파일만 구조 확인 가능

두 개 이상의 파일 비교 어려움

한 화면에 표시되는 정보의 양이 많음

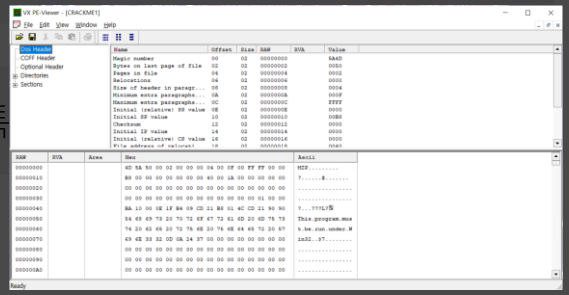
복잡한 파일 분석 시 어려움

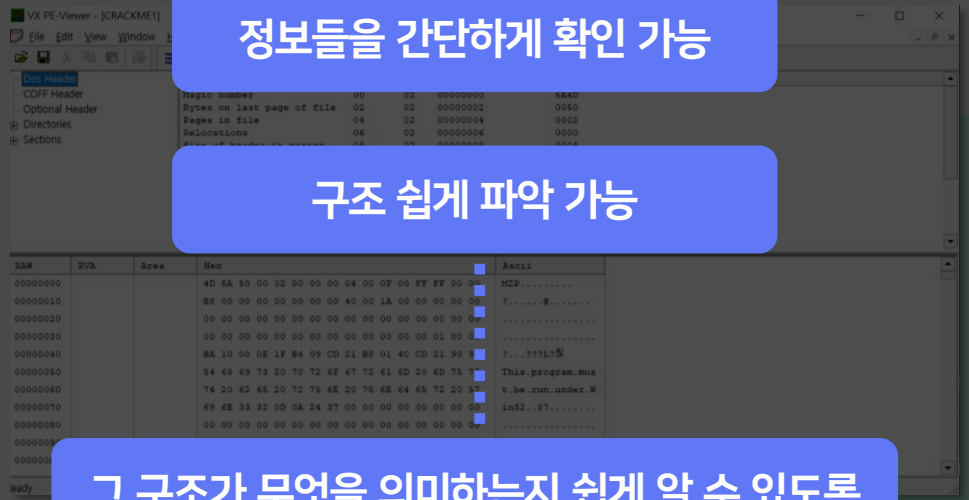
2 기존 프로그램 분석
: VX PE viewer



+

Hex/Ascii 값 직관적 표현





정보들을 간단하게 확인 가능

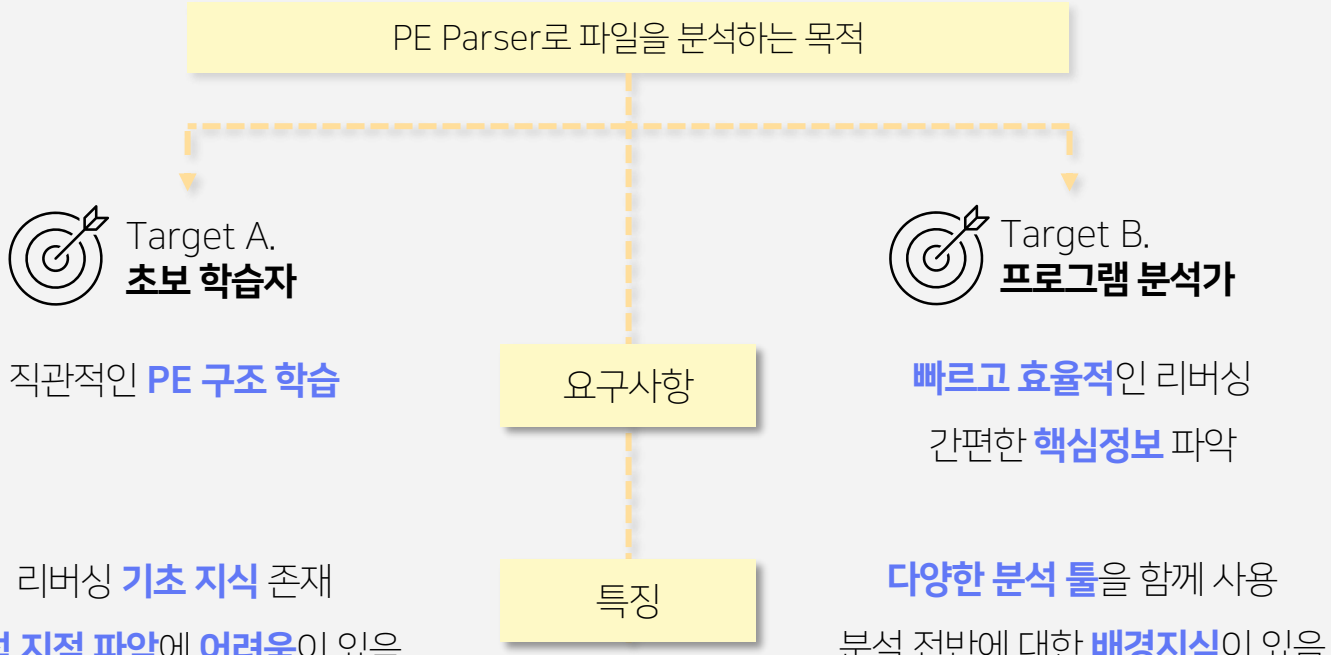
구조 쉽게 파악 가능

그 구조가 무엇을 의미하는지 쉽게 알 수 있도록

3 기획 및 설계

✓ **타겟 페르소나 설정**

- PE 스터디와 기존 툴 사용을 통해 기획 방향을 수립



3 기획 및 설계

✓ **유저 시나리오** : 시나리오를 기반으로 디자인 요구사항 도출

시나리오1. 파일 **패킹 여부**를 확인

분석 프로그램 경로 입력

가장 먼저 패킹 여부 확인

시나리오2. **분석 지점** 찾기

ImaseBase로 메모리 어디에 올라가는지 확인

ImageBase+Base OfCode

찾아낸 섹션을 디버거로 오픈

스트링 / API 검색

✓ **구체적 목표 수립**

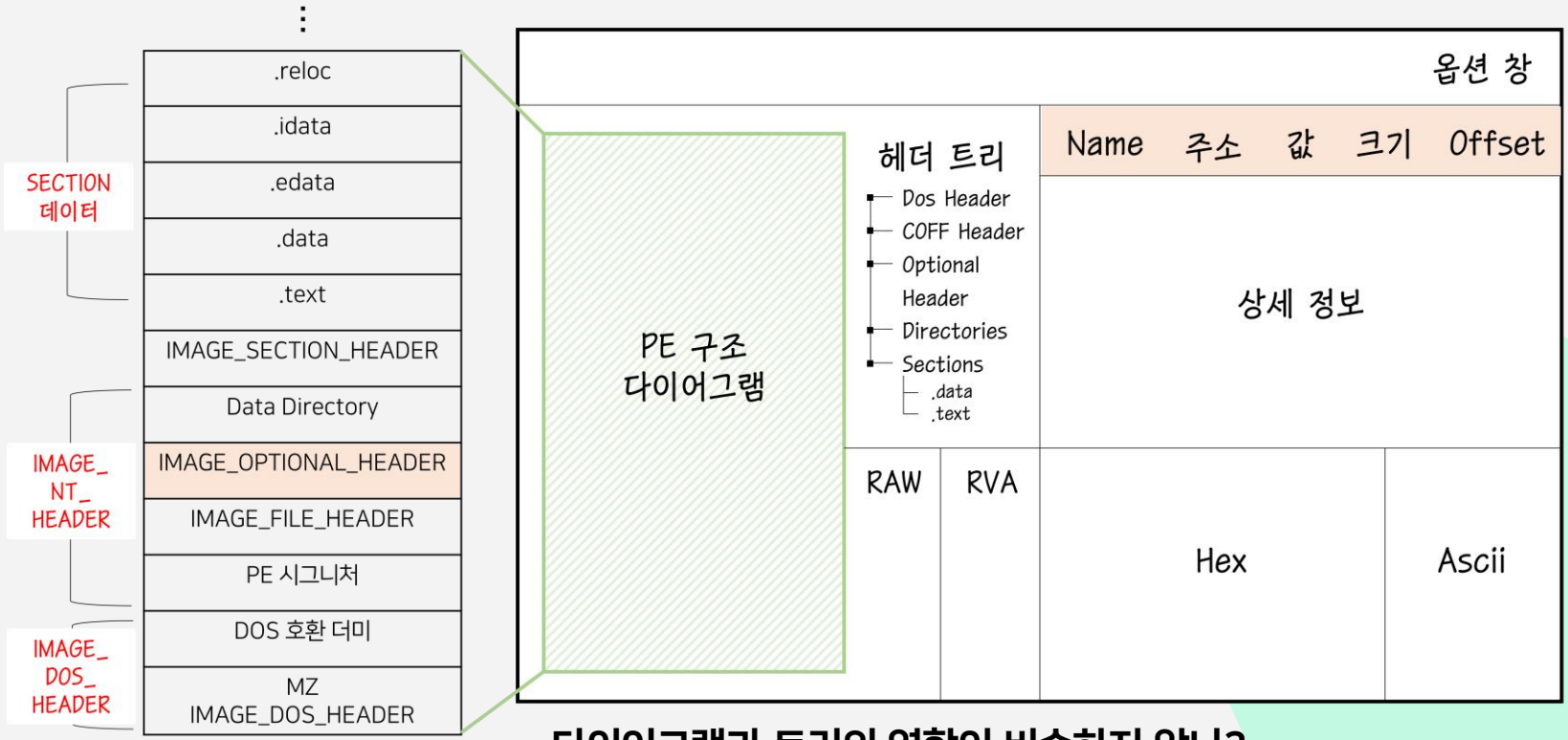
리버싱에 필요한 실행 파일의 정보 (주로 주소) 를 파싱

PE 관련 코드를 작성해보며 **구조체 학습**을 겸함

스트링 형태의 보안 데이터 시각화를 최종 목표로 함

3 기획 및 설계

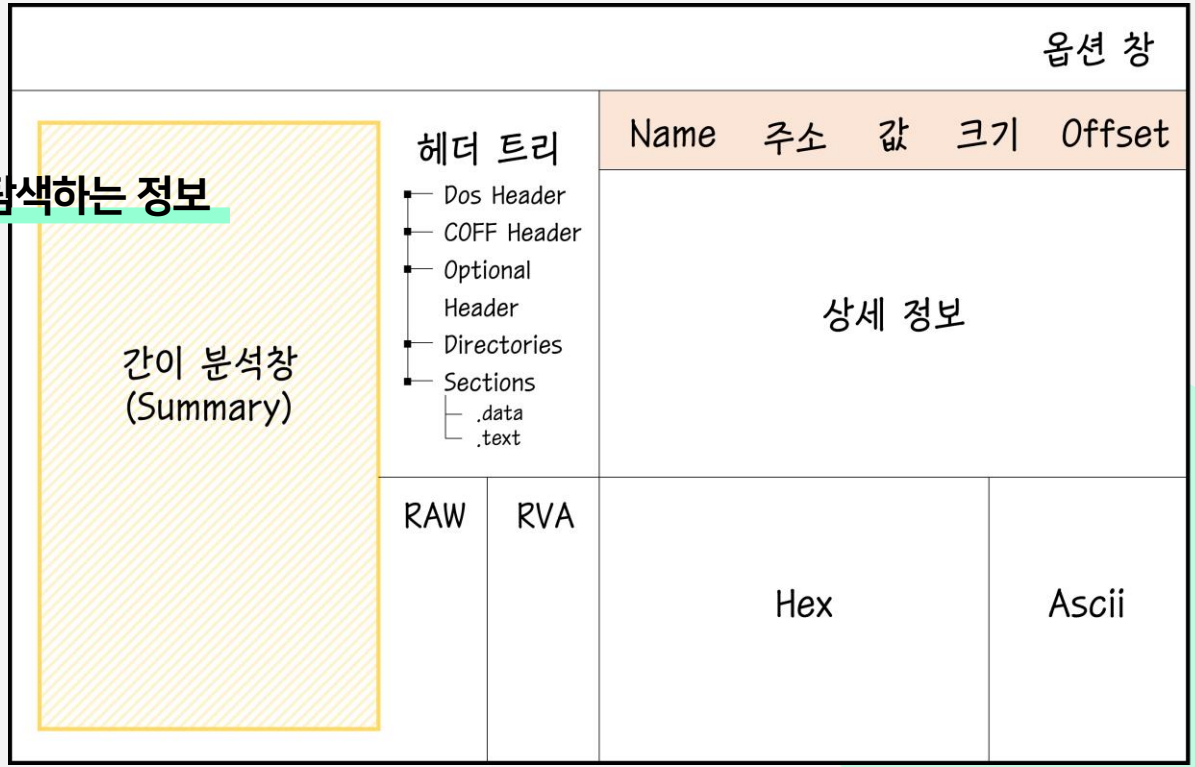
현재 분석 위치를 직관적으로 표현하는 다이어그램



다이어그램과 트리의 역할이 비슷하지 않나?

3 기획 및 설계

✓ **화면 설계**



가장 먼저 / 빈번하게 탐색하는 정보

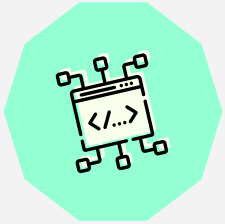
섹션 이름

NumberOfSections

TimeDateStamp

BaseOfCode

ImageBase



03

Solution

: 새롭게 제안하는 PE Parser

구현의 시작부터 UI 적용까지

03

Solution

: 새롭게 제안하는 PE Parser

구현의 시작부터 UI 적용까지

04

Testing

: **시행착오**와 **수정**의 과정

목표한 **기능**이 잘 구현되었을까?

유지보수는 어떻게 해야 할까?

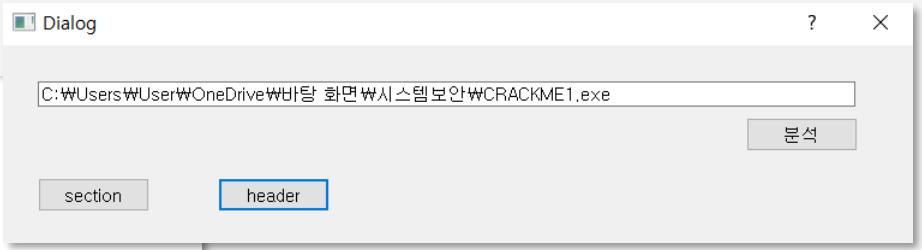
Ver 0.0

```
분석할 파일 경로를 입력해주세요 >> C:₩Users₩tlsdu₩2022₩CRACKME1.exe

[ IMAGE_DOS_HEADER ]
Name                          | Offset    | RAW       | Value
------------------------------| ----------| ----------| ----------
e_magic                       | 0x0       | 0x0       | 0x5a4d
e_cblp                        | 0x2       | 0x2       | 0x50
e_cp                          | 0x4       | 0x4       | 0x2
e_crlc                        | 0x6       | 0x6       | 0x0
e_cparhdr                     | 0x8       | 0x8       | 0x4
e_minalloc                    | 0xa       | 0xa       | 0xf
e_maxalloc                    | 0xc       | 0xc       | 0xffff
e_ss                          | 0xe       | 0xe       | 0x0
e_sp                          | 0x10      | 0x10      | 0xb8
e_csum                        | 0x12      | 0x12      | 0x0
e_ip                          | 0x14      | 0x14      | 0x0
e_cs                          | 0x16      | 0x16      | 0x0
e_lfarlc                      | 0x18      | 0x18      | 0x40
e ovno                        | 0x1a      | 0x1a      | 0x1a
[ IMAGE_NT_HEADERS ]
Name                          | Offset    | RAW       | Value
------------------------------| ----------| ----------| ----------
Signature                     | 0x0       | 0x100     | 0x4550

[ IMAGE_FILE_HEADER ]
Name                          | Offset    | RAW       | Value
------------------------------| ----------| ----------| ----------
Machine                       | 0x0       | 0x104     | 0x14c
NumberOfSections              | 0x2       | 0x106     | 0x6
TimeDateStamp                 | 0x4       | 0x108     | 0xAD92429  [Wed Oct  8 12:18:49
PointerToSymbolTable          | 0x8       | 0x10c     | 0x0
NumberOfSymbols               | 0xc       | 0x110     | 0x0
SizeOfOptionalHeader          | 0x10      | 0x114     | 0xe0
Characteristics               | 0x12      | 0x116     | 0x818e
```

- 기본적으로 출력을 확인하는 **prototype**
- **UI 미구현**
- 내부 **header**와 **section**을 모두 확인

  가능하나 사람이 **직접** 일일이 **탐색**해야 함

Ver 0.1




```
분석할 파일 경로를 입력해주세요 >> C:₩Users₩tlsdu₩2022₩CRACKME1.exe

[ IMAGE_DOS_HEADER ]
Name                          | Offset    | RAW       | Value
------------------------------| ----------| ----------| ----------
e_magic                       | 0x0       | 0x0       | 0x5a4d
e_cblp                        | 0x2       | 0x2       | 0x50
e_cp                          | 0x4       | 0x4       | 0x2
e_crlc                        | 0x6       | 0x6       | 0x0
e_cparhdr                     | 0x8       | 0x8       | 0x4
e_minalloc                    | 0xa       | 0xa       | 0xf
e_maxalloc                    | 0xc       | 0xc       | 0xffff
e_ss                          | 0xe       | 0xe       | 0x0
e_sp                          | 0x10      | 0x10      | 0xb8
e_csum                        | 0x12      | 0x12      | 0x0
e_ip                          | 0x14      | 0x14      | 0x0
e_cs                          | 0x16      | 0x16      | 0x0
e_lfarlc                      | 0x18      | 0x18      | 0x40
e_ovno                        | 0x1a      | 0x1a      | 0x1a
[ IMAGE_NT_HEADERS ]
Name                          | Offset    | RAW       | Value
------------------------------| ----------| ----------| ----------
Signature                     | 0x0       | 0x100     | 0x4550

[ IMAGE_FILE_HEADER ]
Name                          | Offset    | RAW       | Value
------------------------------| ----------| ----------| ----------
Machine                       | 0x0       | 0x104     | 0x14c
NumberOfSections              | 0x2       | 0x106     | 0x6
TimeDateStamp                 | 0x4       | 0x108     | 0xAD92429  [Wed Oct  8 12:18:49
PointerToSymbolTable          | 0x8       | 0x10c     | 0x0
NumberOfSymbols               | 0xc       | 0x110     | 0x0
SizeOfOptionalHeader          | 0x10      | 0x114     | 0xe0
Characteristics               | 0x12      | 0x116     | 0x818e
```

- **UI**를 **최초 적용**한 버전
- **Section, header 버튼 추가**
- 아직 정보가 **UI** 에는 출력 X
- Ver 0.1부터 **테스트 진행** 후 업데이트

테스트를 위한 **기능 명세서**

✓ 기획의도에 맞추어 작성한 기능 명세서를 기반으로 **테스트 케이스**를 나눔

구분	주요기능(기능 명칭)	기능정의(주요 역할)	기능설명
화면구성	PE구조 다이어그램	PE구조를 볼 수 있게 함	PE구조 다이어그램을 통해 현재 지금 분석하고 있는 위치를 다이어그램에 표시하고, 해당 부분에 하이라이트를 나타냄으로써 분석의 효율을 높이는 역할을 함
	헤더 트리	헤더를 볼 수 있게 함	이전에 사용했던 vxpeviewer의 장점을 살려서 헤더트리를 보여줌으로써 분석에 용이하게 함
	상세 정보	분석 부분을 상세히 볼 수 있게 함	Name, 주소, 값, 크기, Offset등을 화면에 표시하는 부분으로 원하는 부분의 상세정보를 알고 싶을 때 용이하고, 이를 통해 다양한 부분의 상세 정보를 볼 수 있음
	RAW	분석하는 부분의 RAW값	분석하는 부분의 절대주소인 RAW의 값이 들어가는 부분으로 분석하고자 하는 부분을 클릭 시 절대주소를 바로 출력함
	RVA	분석하는 부분의 RVA값	분석하는 부분의 상대주소인 RVA의 값이 들어가는 부분으로 분석하고자 하는 부분을 클릭 시 상대주소를 바로 출력함
	HEX	분석하는 부분의 HEX값	분석하고자 하는 부분을 클릭 시 vxpeviewer에서 표시되는 것처럼 해당 부분의 HEX부분을 표시해서 어디 부분을 분석하는지 한 번에 보기 용이하게 함
	Ascii	분석하는 부분의 Ascii값	HEX값에 따른 Ascii값을 화면에서 바로 보이게하여 분석하는 부분을 착각하지 않게 만들고, Ascii값을 바로바로 보이게하면 분석에 더욱 용이한 역할을 할 것임

유닛 테스트

- ✓ 소스 코드의 특정 **모듈이 의도된 대로 정확히 작동하는지** 검증하는 절차
- ✓ 모든 함수와 메소드에 대한 **테스트 케이스를 작성**하는 절차
- ✓ 내부 함수인 **print_info** (pe정보 출력함수) 만 진행

Print_info 함수에 오류가 없음을 검증

```
import unittest
import pefile
import ver001

class ver0Test(unittest.TestCase):
        def test_print_info(self):
            path = input("분석할 파일 경로를 입력>>")
            print()
            pe = pefile.PE(path)
            hinfo = [pe.DOS_HEADER.dump_dict(), pe.NT_HEADERS.dump_dict(),

            for h in hinfo:
                ver001.print_info(h)

if __name__ =='__main__':
    unittest.main(argv=['first-arg-is-ignored'],exit=False)
```

```
분석할 파일 경로를 입력>>C:₩Users₩User₩OneDrive₩바탕 화면₩시스템보안₩CRACKME1.exe

.
----------------------------------------------------------------------
Ran 1 test in 1.926s

OK

Name                          | Offset    | RAW       | Value
------------------------------| ----------| ----------| ----------
e_magic                       | 0x0       | 0x0       | 0x5a4d
e_cblp                        | 0x2       | 0x2       | 0x50
e_cp                          | 0x4       | 0x4       | 0x2
e_crlc                        | 0x6       | 0x6       | 0x0
e_cparhdr                     | 0x8       | 0x8       | 0x4
e_minalloc                    | 0xa       | 0xa       | 0xf
```

비기능 테스트

- ✓ 제품의 **성능, 안정성, 확장성** 확인
- ✓ 제품이 얼마나 잘 동작하는지 확인
- ✓ **확장한 테스트 케이스**로 엑셀 파일 작성
- ✓ 직접 제품 사용 후 **개선 사항 및 버전 관리** 피드백

Ver 0.1 테스트 케이스

ID	Category	Preconditions	Steps	Expected result	EXE	SCR	SYS	DLL	OCX	CPL	DRV	OBJ	실행불가파일	Comment
UI001	분석	QT designer를 켜고, 같은 폴더내에 ui파일이 있는 상태이며, 정상적인 파일 경로를 입력해둔 상태	UI에서 분석버튼 클릭	해당 파일의 이름이 파이썬 결과에 출력됨.	PASS	PASS	PASS	PASS	PASS	PASS	PASS	Error	Error	
UI002	section	QT designer를 켜고, 같은 폴더내에 ui파일이 있는 상태이며, 정상적인 파일 경로를 입력해둔 상태	UI에서 section버튼 클릭	해당 파일의 section들이 파이썬 결과에 출력됨	PASS	PASS	PASS	PASS	PASS	PASS	PASS	Error	Error	아직 UI에 아무런 정보도 표시되지 않는 점이 아쉽다
UI003	header	QT designer를 켜고, 같은 폴더내에 ui파일이 있는 상태이며, 정상적인 파일 경로를 입력해둔 상태	UI에서 header 버튼 클릭	해당 파일의 header정보들이 파이썬 결과에 출력됨	PASS	PASS	PASS	PASS	PASS	PASS	PASS	Error	Error	

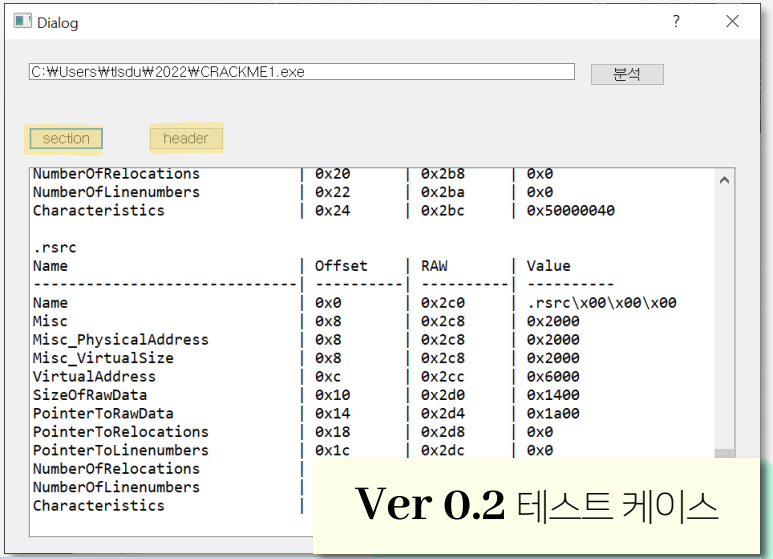
Ver 0.2

개선한 사항

- ✓ **상세정보창** 처음 도입한 버전
- ✓ Header, section, 분석 버튼을 각각 클릭 시 **UI에 정보가 출력**되도록 변경

개선해야할 사항

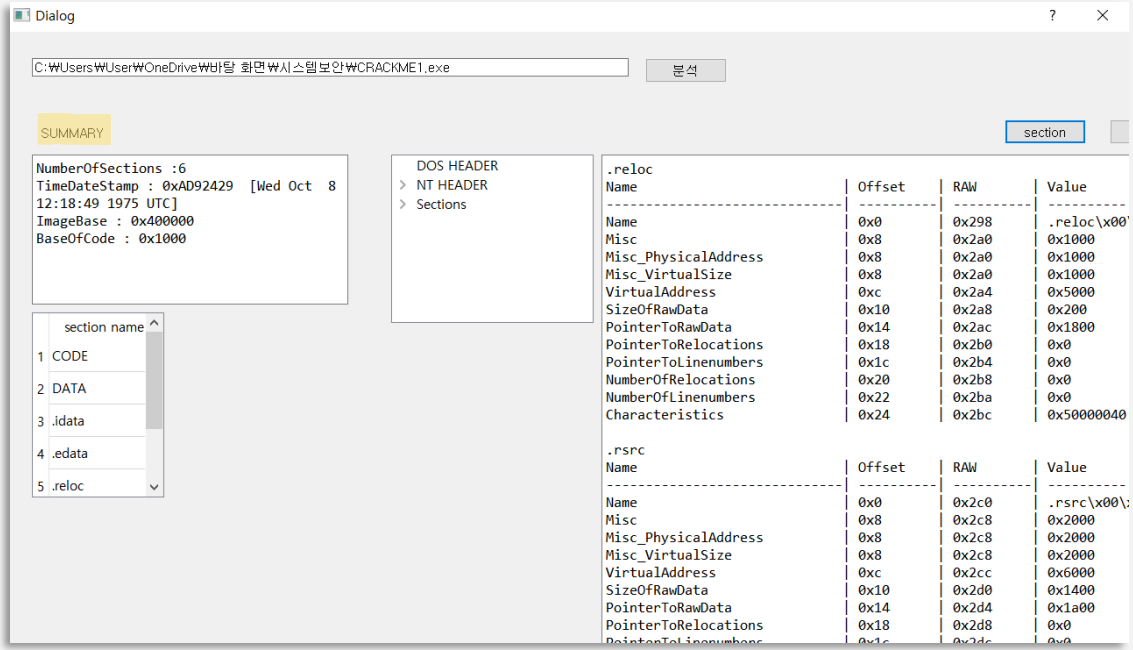
- ✓ 아직 python 코드 출력 부분에 내용 출력
- ✓ Header와 section **분리 X**



Ver 0.2 테스트 케이스

	ID	Category	Preconditions	Steps	Expected result	EXE	SCR	SYS	DLL	OCX	CPL	DRV	OBJ	실행불가파일	Comment
UI	UI001	분석	QT designer에 ui파일을 넣어두고, 정상적인 파일 경로를 입력해둔 상태	UI에서 분석버튼 클릭	해당 파일의 이름이 파이썬 결과 및 UI에 출력됨.	PASS	PASS	PASS	PASS	PASS	PASS	PASS	Error	Error	이전버전의 단점을 보완해 UI에 정보가 출력됨.
	UI002	section	QT designer에 ui파일을 넣어두고, 정상적인 파일 경로를 입력해둔 상태	UI에서 section버튼 클릭	해당 파일의 section들이 파이썬 결과 및 UI에 출력됨	PASS	PASS	PASS	PASS	PASS	PASS	PASS	Error	Error	이전 버전의 단점을 보완해 UI에 정보가 출력되나, 세부사항을 자세히 보려면 일일히 찾아야하는 아쉬움이 있음
	UI003	header	QT designer에 ui파일을 넣어두고, 정상적인 파일 경로를 입력해둔 상태	UI에서 header 버튼 클릭	해당 파일의 header정보들이 파이썬 결과 및 UI에 출력됨	PASS	PASS	PASS	PASS	PASS	PASS	PASS	Error	Error	

Ver 1.0



개선한 사항

- ✓ 기획한 화면설계에 근접한 최초의 형태
- ✓ **Header 트리**와 **section 리스트** 분리

개선해야할 사항

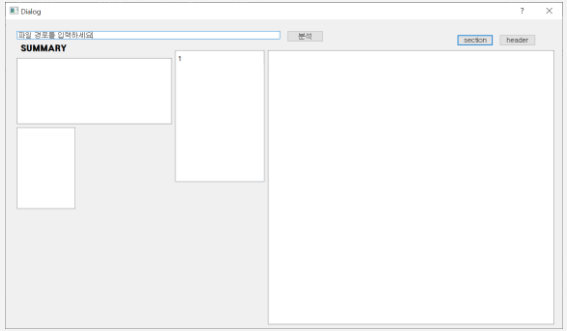
- ✓ 이전의 정보가 잔류하는 오류 존재
- ✓ 기획 때 의도했던 상세 사항 시각화
- 기능 미비

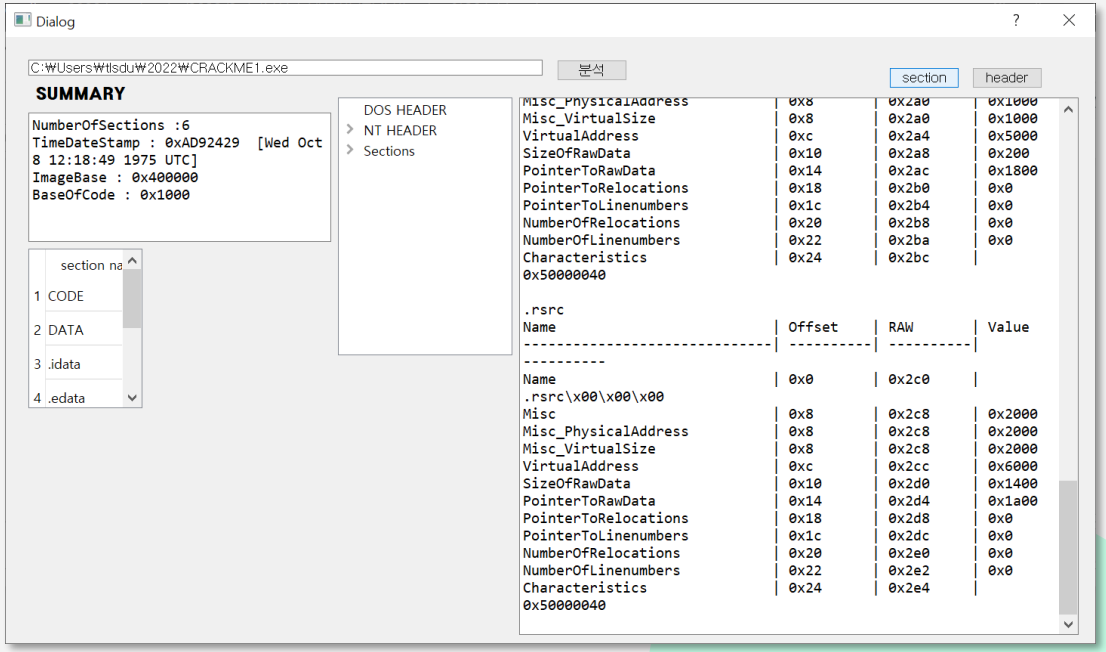
Ver 1.0

Ver 1.0 테스트 케이스

	ID	Category	Preconditions	Steps	Expected result	EXE	SCR	SYS	DLL	OCX	CPL	DRV	OBJ	실행불가파일	Comment
UI	UI001	분석	QT designer에 ui 파일을 넣어두고, 정상적인 파일 경로를 입력해둔 상태	UI에서 분석버튼 클릭	해당 파일의 이름이 UI에 존재하는 상세정보란에 출력됨.	PASS	PASS	PASS	PASS	PASS	PASS	PASS	Error	Error	분석을 누르면 아직도 파이썬 결과 창에 해당 사항이 출력됨.
	UI002	section	UI에서 분석버튼을 클릭한 후의 상태	UI에서 section버튼 클릭	해당 파일의 section의 내용 전부가 UI에 존재하는 상세정보란에 출력됨.	PASS	PASS	PASS	PASS	PASS	PASS	PASS	Error	Error	실행불가 파일을 넣어도 이전의 정보가 남아있어서 이전 정보의 section과 header를 볼 수 있다는 단점이 있다.
	UI003	header		UI에서 header 버튼 클릭	header정보 전부가 UI에 존재하는 상세정보란에 출력됨	PASS	PASS	PASS	PASS	PASS	PASS	PASS	Error	Error	
	UI004	section_name	QT designer에 ui 파일을 넣어두고, 정상적인 파일 경로를 입력해둔 상태	UI에서 분석버튼 클릭	해당 파일의 section name을 볼 수 있는 구조가 UI에 출력됨.	PASS	PASS	PASS	PASS	PASS	PASS	PASS	Error	Error	실행불가 파일을 넣어도 이전의 정보가 남아있다.
	UI005	tree			트리 구조가 출력되고, 해당 트리 부분을 클릭하면 입력받은 파일의 해당부분을 상세정보란에 바로 보여줌.	PASS	PASS	PASS	PASS	PASS	PASS	PASS	Error	Error	지금 넣은 파일의 해당 section name이 없는 경우에도 이전에 넣었던 파일의 section name의 양식이 남아있어 값이 없는 section name이 잔류한다. 분석 버튼을 여러 번 누를 경우 tree가 여러 개가 출력된다.
	UI006	Summary			해당 파일의 요약정보인 ImageBase, BaseOfCode등이 UI에 출력됨.	PASS	PASS	PASS	PASS	PASS	PASS	PASS	Error	Error	실행불가 파일을 넣어도 이전의 정보가 남아있다.

Ver 1.1





개선한 사항

- ✓ 1.0과 형태 유지하되 오류 수정
- ✓ 트리의 잔류 오류 수정
- ✓ 분석 클릭 시 해당 내용이 python 결과창에 나오지 않게 수정

개선해야할 사항

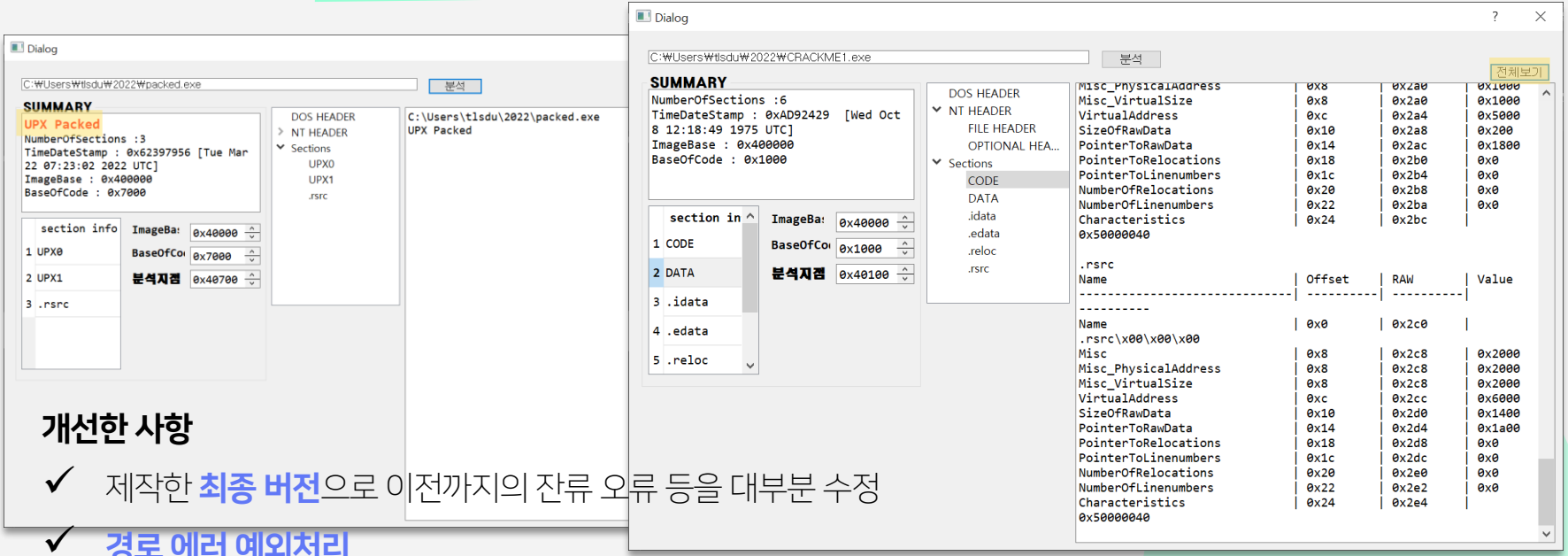
- ✓ Section, header 창의 잔류 오류 수정해야 함
- ✓ **경로가 잘못된 파일 입력 시 에러**
- ✓ 오류 사항이 화면에 출력되지 않음

Ver 1.1

Ver 1.1 테스트 케이스

	ID	Category	Preconditions	Steps	Expected result	EXE	SCR	SYS	DLL	OCX	CPL	DRV	OBJ	실행불가파일	Comment
UI	UI001	분석	QT designer에 ui 파일을 넣어두고, 정상적인 파일 경로를 입력해둔 상태	UI에서 분석버튼 클릭	해당 파일의 이름이 UI에 존재하는 상세정보란에 출력됨.	PASS	PASS	PASS	PASS	PASS	PASS	PASS	Error	Error	파이썬 결과창에 해당 내용이 출력되지 않도록 변경
	UI002	section	UI에서 분석버튼을 클릭한 후의 상태	UI에서 section버튼 클릭	해당 파일의 section의 내용 전부가 UI에 존재하는 상세정보란에 출력됨.	PASS	PASS	PASS	PASS	PASS	PASS	PASS	Error	Error	분석할 수 없는 파일을 올려도 전의 정보가 잔류함
	UI003	header		UI에서 header 버튼 클릭	해당 파일의 header정보 전부가 UI에 존재하는 상세정보란에 출력됨.	PASS	PASS	PASS	PASS	PASS	PASS	PASS	Error	Error	분석할 수 없는 파일을 올려도 전의 정보가 잔류함
	UI004	section_name	QT designer에 ui 파일을 넣어두고, 정상적인 파일 경로를 입력해둔 상태	UI에서 분석버튼 클릭	해당 파일의 section name을 볼 수 있는 구조가 UI에 출력됨.	PASS	PASS	PASS	PASS	PASS	PASS	PASS	Error	Error	분석할 수 없는 파일을 올려도 전의 정보가 잔류함
	UI005	tree			트리 구조가 출력되고, 해당 트리 부분을 클릭하면 입력받은 파일의 해당부분을 상세정보란에 바로 보여줌.	PASS	PASS	PASS	PASS	PASS	PASS	PASS	Error	Error	이전 버전에 있던 오류를 모두 고침
	UI006	Summary			해당 파일의 요약정보인 ImageBase, BaseOfCode등이 UI에 출력됨.	PASS	PASS	PASS	PASS	PASS	PASS	PASS	Error	Error	분석할 수 없는 파일을 올렸을 경우 이전의 정보들이 삭제됨.

Ver 1.2



개선한 사항

- ✓ 제작한 **최종 버전**으로 이전까지의 잔류 오류 등을 대부분 수정
- ✓ **경로 에러 예외처리**
- ✓ Header와 section 전체보기 버튼 추가해 한 번에 볼 수 있는 **전체 창** 제공
- ✓ 기획했던 분석지점과 선별정보들이 UI에 바로 표시되도록 개선
- ✓ **Packing 여부**를 요약창 상단에 강조

Ver 1.2

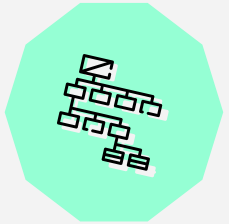
	ID	Category	Preconditions	Steps	Expected result	EXE	SCR	SYS	DLL	OCX	CPL	DRV	OBJ	실행불가파일	Comment
UI	UI001	분석	QT designer에 ui 파일을 넣어두고, 정상적인 파일 경로를 입력해둔 상태	UI에서 분석버튼 클릭	해당 파일의 이름이 UI에 존재하는 상세정보란에 출력됨.	PASS	PASS	PASS	PASS	PASS	PASS	PASS	Error	Error	경로 에러가 발생했을 경우 Summary와 세부사항에 파일이 존재하지 않습니다 출력
	UI002	전체보기	UI에서 분석버튼을 클릭한 후의 상태	UI에서 전체보기버튼 클릭	해당 파일의 section과 header 내용 전부가 UI에 존재하는 상세정보란에 출력됨.	PASS	PASS	PASS	PASS	PASS	PASS	PASS	Error	Error	이전에 분석했던 정보가 잔류하던 오류를 고침
	UI003	section info			해당 파일의 section name을 볼 수 있는 구조가 UI에 출력됨.	PASS	PASS	PASS	PASS	PASS	PASS	PASS	Error	Error	분석할 수 없는 파일을 올려도 전의 정보가 잔류함
	UI004	tree			트리 구조가 출력되고, 해당 트리 부분을 클릭하면 입력받은 파일의 해당부분을 상세정보란에 바로 보여줌.	PASS	PASS	PASS	PASS	PASS	PASS	PASS	Error	Error	이전 버전에 있던 오류를 모두 고침
	UI005	Summary			해당 파일의 요약정보인 ImageBase, BaseOfCode등이 UI에 출력됨.	PASS	PASS	PASS	PASS	PASS	PASS	PASS	Error	Error	분석할 수 없는 파일을 올렸을 경우 이전의 정보들이 삭제됨. ImageBase, BaseOfCode가 따로 출력창이 생겨 해당사항을 표시할 이유가 사라짐
	UI006	ImageBase	QT designer에 ui 파일을 넣어두고, 정상적인 파일 경로를 입력해둔 상태	UI에서 분석버튼 클릭	해당 파일의 ImageBase를 출력	PASS	PASS	PASS	PASS	PASS	PASS	PASS	Error	Error	분석할 수 없는 파일을 올렸을 경우 이전의 정보들이 삭제됨.
	UI007	BaseOfCode			해당 파일의 BaseOfCode를 출력	PASS	PASS	PASS	PASS	PASS	PASS	PASS	Error	Error	분석할 수 없는 파일을 올렸을 경우 이전의 정보들이 삭제됨.
	UI008	분석지점			ImageBase+BaseOfCode 출력	PASS	PASS	PASS	PASS	PASS	PASS	PASS	Error	Error	분석할 수 없는 파일을 올렸을 경우 이전의 정보들이 삭제됨.

Ver 1.2 테스트 케이스

개선해야할 사항

- ✓ Obj 파일을 분석할 수 없는 오류
- ✓ 화면으로 나타나는 부분 클릭 시 **분석지점이 변경되는 로직** 추가

구현 과정의 **어려움**



Challenge 1

: 멤버 변수 크기 출력

- 멤버 변수 **offset의 차**를 이용해 변수 크기를 구하는 방법을 시도
- **이름이 겹치는 변수**의 경우 size = 0 으로 계산

Challenge 2

: GUI 구현의 어려움

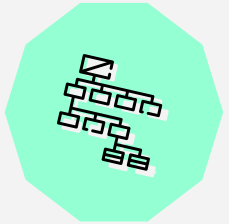
- **Pyqt, Qt designer** 사용의 미숙함 탓에 UI 디자인 과정 자체가 자유롭지 못했음
- 버튼 클릭 시 특정 **시그널 함수 연결 구현 이해**에 시간 소요
- 원하는 내용의 색상 변경 부분도 실제 난이도에 비해 늦은 진행

Challenge 3

: Hex dump 구현

- 예제코드를 응용하려 했으나 시간적 여유 부족
- UI 구현에 좀 더 집중하는 방법을 선택

구현 과정의 **어려움**



Challenge 1
: 멤버 변수 크기 출력

겹치는 변수 제거로 해결

Dump_dict()를 이용,

딕셔너리 형태로 데이터를

가져와 마지막 멤버 크기는

여전히 고민 필요

Challenge 2
: GUI 구현의 어려움

QTreeWidget 시그널로 해결

폰트 색상 변경 성공

다양한 기능 학습 추가 필요

Challenge 3
: Hex dump 구현

HxD 기존 툴과 함께 사용

한 화면에 모든 view X

기존의 프로그램과 함께 잘

활용할 수 있는 툴 제작

Our Next Goals

UI 보완

가독성 향상

PE 구조체 멤버 변수 설명 제공

보다 **친절한 정보 제공 툴**로 보안

데이터 해석의 장벽 낮추기

정보 가공

단순한 origin 정보의 출력뿐만

아니라 주소 계산, API 분석 등

의미 있는 정보로 가공하여 제공

지속적 유지보수

프로젝트 기간 내 해결하지 못한

문제에 대해 추가 고민

배포 후 다양한 사용자의 의견 수렴

Thanks!

Do you have any questions?

https://github.com/kiietls23/PEparser

CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, and infographics & images by Freepik

KU Seiong 2022