ELF와 PE 실행파일 구조

IT지능정보공학과 김현아

주제 선택 동기

- **운영체제와의 연관성**

  → ELF는 리눅스 및 유닉스 계열 시스템에서, PE는 윈도우 시스템에서 실행 파일 및 라이브러리의 표준 포맷

- **실행 파일은 운영체제 커널에서 직접 로드 및 실행**

  → 시스템 동작의 핵심적인 부분을 차지

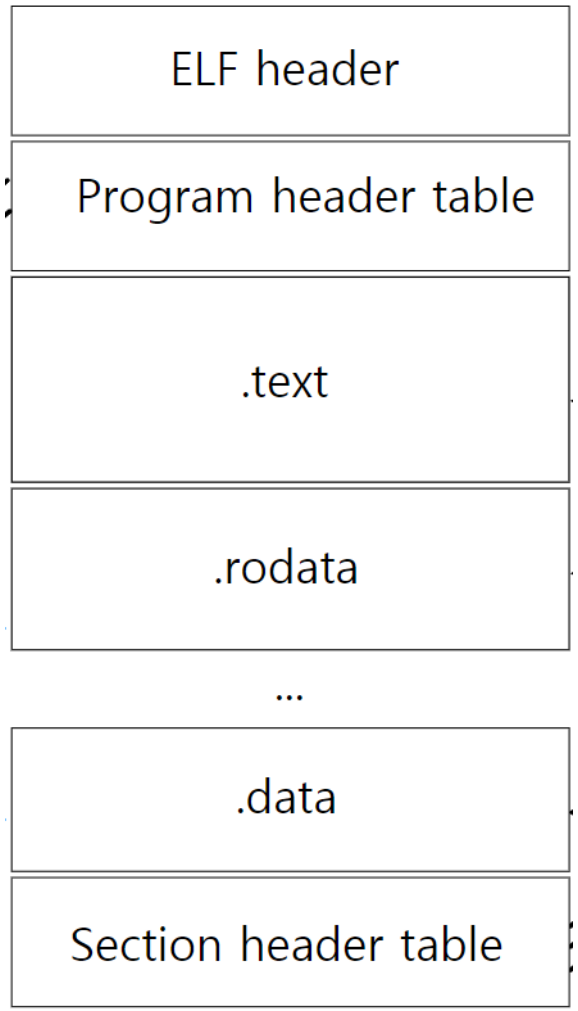
- **대부분의 악성코드는 실행 파일 형태로 시스템에 침투**

  → 실행 파일의 헤더, 섹션, 또는 심볼 테이블을 변조하여 익스플로잇 코드 삽입이나 바이너리 패치를 수행

ELF (Executable and Linkable File)

- Executable (실행 가능한 파일)과 Linkable (연결 가능한 파일)의 약자로, 주로 리눅스 환경에서 사용되는 바이너리 파일 포맷
- 리눅스에서 프로그램 실행과 관련된 모든 핵심 정보를 담고 있는 중요한 파일
- UNIX계열의 실행 파일, 오브젝트 파일, 공유 라이브러리, 코어덤프를 위한 표준 파일 형식
- 소스 코드 작성 → 컴파일 → **오브젝트 파일**→ 링킹 → **실행 파일**

ELF의 구조



파일의 구성

ELF내의 세그먼트들에 대한 정보와 세그먼트들을 어떻게 로드해야하는지에 대한 정보

.text	프로그램의 실행 가능한 명령어
.rodata	읽기 전용 데이터를 가지는 섹션(문자열, 상수)
.bss	초기화 되지 않은 데이터(0으로 자동 초기화)
.data	초기화 된 데이터(전역변수, 정적변수)

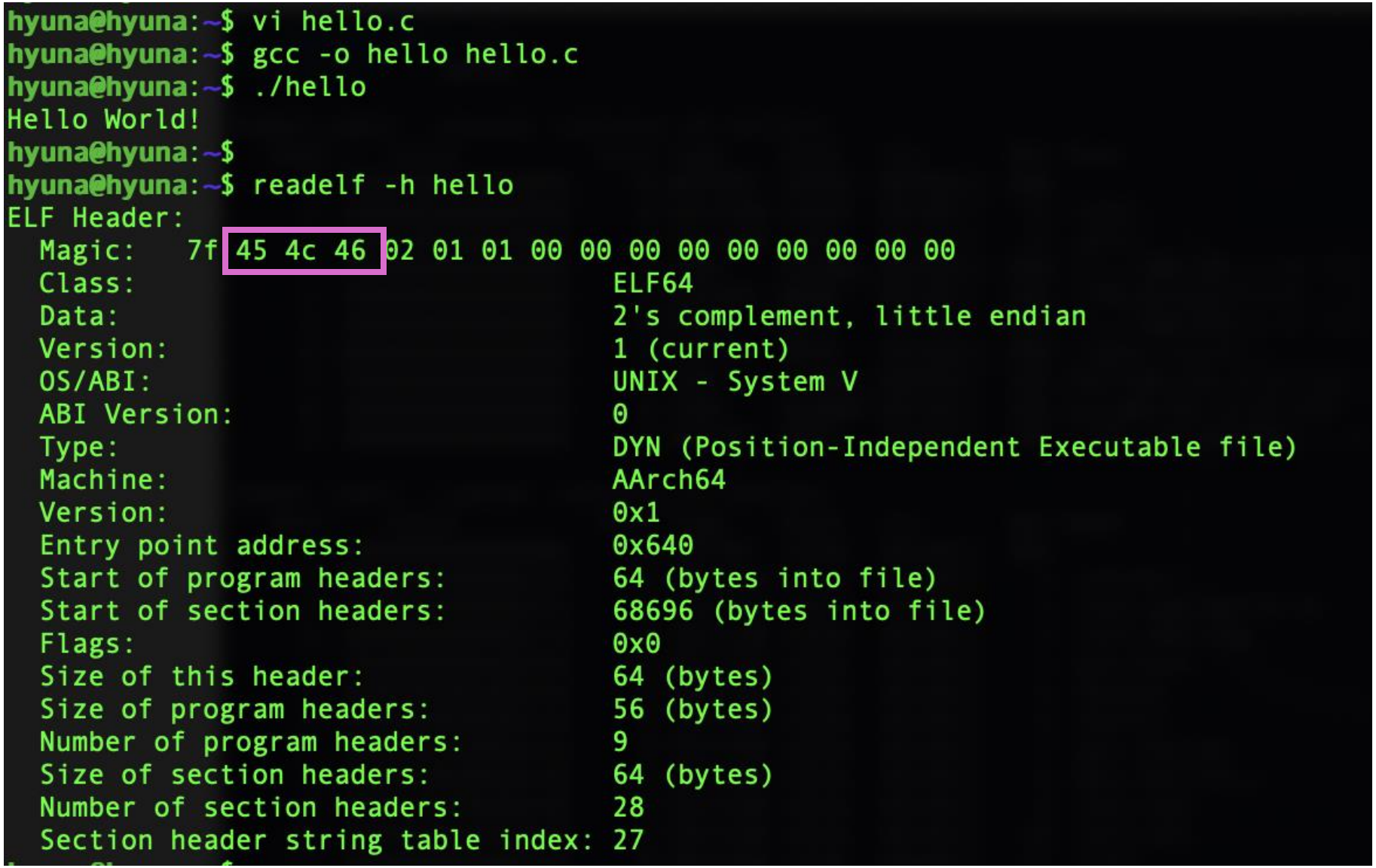
섹션의 위치, 크기, 속성 등의 정보 저장

ELF Header

오프셋		크기 (Bytes)		필드	목적
32-bit	64-bit	32-bit	64-bit		
0x00		4		e_ident[EI_MAG0] 부터 e_ident[EI_MAG3]	0x7F 와 ASCII 코드 ELF ; 이 네 바이트가 매직 넘버를 형성한다.
0x04		1		e_ident[EI_CLASS]	이 바이트는 1 또는 2 로 설정되며 32비트 또는 64비트 형식을 나타낸다.
0x05		1		e_ident[EI_DATA]	이 바이트는 1 또는 2 로 설정되며 리틀 또는 빅엔디언을 나타낸다.
0x06		1		e_ident[EI_VERSION]	오리지널 버전의 ELF인 경우 1 로 설정된다.
0x07		1		e_ident[EI_OSABI]	대상 운영 체제 ABI를 구별한다.. 값 / ABI 0x00 System V 0x01 HP-UX 0x02 NetBSD 0x03 리눅스 0x06 솔라리스 0x07 AIX 0x08 IRIX 0x09 FreeBSD 0x0C OpenBSD 0x0D OpenVMS 이것은 대상 플랫폼과 관련 없이 종종 0 으로 설정된다.

ELF Header

0x12		2		e_machine	0x02 SPARC 0x03 x86 0x08 MIPS 0x14 파워PC 0x28 ARM 0x2A 슈퍼H 0x32 IA-64 0x3E x86-64 0xB7 AArch64
0x14		4		e_version	오리지날 버전의 ELF인 경우 1로 설정된다.
0x18		4	8	e_entry	이것은 엔트리 포인트의 메모리 주소이다. 즉 프로세스가 어디서 실행을 시작하는지를 말해준다. 이 필드는 위에서 정의한 32비트 또는 64비트에 따라 길이가 다르다.
0x1C	0x20	4	8	e_phoff	프로그램 헤더 테이블의 시작을 가리킨다.
0x20	0x28	4	8	e_shoff	섹션 헤더 테이블의 시작을 가리킨다.
0x24	0x30	4		e_flags	대상 아키텍처에 따라 이 필드의 해석이 달라진다.
0x28	0x34	2		e_ehsize	이 헤더의 크기를 가지며 일반적으로 64비트의 경우 64바이트, 32비트의 경우 52바이트이다.
0x2A	0x36	2		e_phentsize	프로그램 헤더 테이블 엔트리의 크기를 갖는다.
0x2C	0x38	2		e_phnum	프로그램 헤더 테이블에서 엔트리의 개수.
0x2E	0x3A	2		e_shentsize	섹션 헤더 테이블 엔트리의 크기를 갖는다.
0x30	0x3C	2		e_shnum	섹션 헤더 테이블에서 엔트리의 개수.
0x32	0x3E	2		e_shstrndx	섹션 이름들을 포함하는 섹션 헤더 테이블 엔트리의 인덱스.



readelf –h hello

Section Header Table

```c
#include <stdio.h>

int bss1;
static int bss2;

int data1 = 1000;
static int data2 = 1000;

int main(){
        char* data1 = "Hello World!";
        puts(data1);
}
```

```
[hyuna@hyuna:~$
[hyuna@hyuna:~$ readelf -S hello
There are 28 section headers, starting at offset 0x10c58:

Section Headers:
  [Nr] Name              Type             Address           Offset
       Size              EntSize          Flags  Link  Info  Align
  [ 0]                   NULL             0000000000000000  00000000
       0000000000000000  0000000000000000           0     0     0
  [ 1] .interp           PROGBITS         0000000000000238  00000238
       000000000000001b  0000000000000000   A       0     0     1
  [ 2] .note.gnu.bu[...] NOTE             0000000000000254  00000254
       0000000000000024  0000000000000000   A       0     0     4
  [ 3] .note.ABI-tag     NOTE             0000000000000278  00000278
       0000000000000020  0000000000000000   A       0     0     4
  [ 4] .gnu.hash         GNU_HASH         0000000000000298  00000298
       000000000000001c  0000000000000000   A       5     0     8
  [ 5] .dynsym           DYNSYM           00000000000002b8  000002b8
       00000000000000f0  0000000000000018   A       6     3     8
  [ 6] .dynstr           STRTAB           00000000000003a8  000003a8
       0000000000000092  0000000000000000   A       0     0     1
  [ 7] .gnu.version      VERSYM           000000000000043a  0000043a
       0000000000000014  0000000000000002   A       5     0     2
  [ 8] .gnu.version_r    VERNEED          0000000000000450  00000450
       0000000000000030  0000000000000000   A       6     1     8
  [ 9] .rela.dyn         RELA             0000000000000480  00000480
       00000000000000c0  0000000000000018   A       5     0     8
  [10] .rela.plt         RELA             0000000000000540  00000540
       0000000000000078  0000000000000018  AI       5    21     8
  [11] .init             PROGBITS         00000000000005b8  000005b8
       0000000000000018  0000000000000000  AX       0     0     4
  [12] .plt              PROGBITS         00000000000005d0  000005d0
       0000000000000070  0000000000000000  AX       0     0     16
  [13] .text             PROGBITS         0000000000000640  00000640
       0000000000000140  0000000000000000  AX       0     0     64
  [14] .fini             PROGBITS         0000000000000780  00000780
       0000000000000014  0000000000000000  AX       0     0     4
  [15] .rodata           PROGBITS         0000000000000798  00000798
       0000000000000015  0000000000000000   A       0     0     8
  [16] .eh_frame_hdr     PROGBITS         00000000000007b0  000007b0
       000000000000003c  0000000000000000   A       0     0     4
  [17] .eh_frame         PROGBITS         00000000000007f0  000007f0
       00000000000000b4  0000000000000000   A       0     0     8
```

readelf –S hello

```
[ 5] .dynsym            DYNSYM          0000000000002b8  000002b8
     00000000000000f0   000000000000018   A       6     3      8
```

```
[25] .symtab            SYMTAB          0000000000000000  00010048
     0000000000008d0   000000000000018          26    69      8
```

⌐ main() 함수 내부의 실행 코드

```
[13] .text              PROGBITS        0000000000000640  00000640
     000000000000140   0000000000000000  AX       0     0      64
```

```
[15] .rodata            PROGBITS        0000000000000798  00000798
     000000000000015   0000000000000000   A       0     0      8
```

ㄴ "Hello World! " ⇢ 문자열

⌐ int data1 = 1000; ⇢ 초기화된 정적 변수

```
[22] .data              PROGBITS        0000000000020000  00010000
     000000000000018   0000000000000000  WA       0     0      8
[23] .bss               NOBITS          0000000000020018  00010018
     000000000000010   0000000000000000  WA       0     0      4
```

ㄴ int bss1; ⇢ 초기화 되지 않은 변수

Symbol Table

.dynsym: 공유 라이브러리 심볼

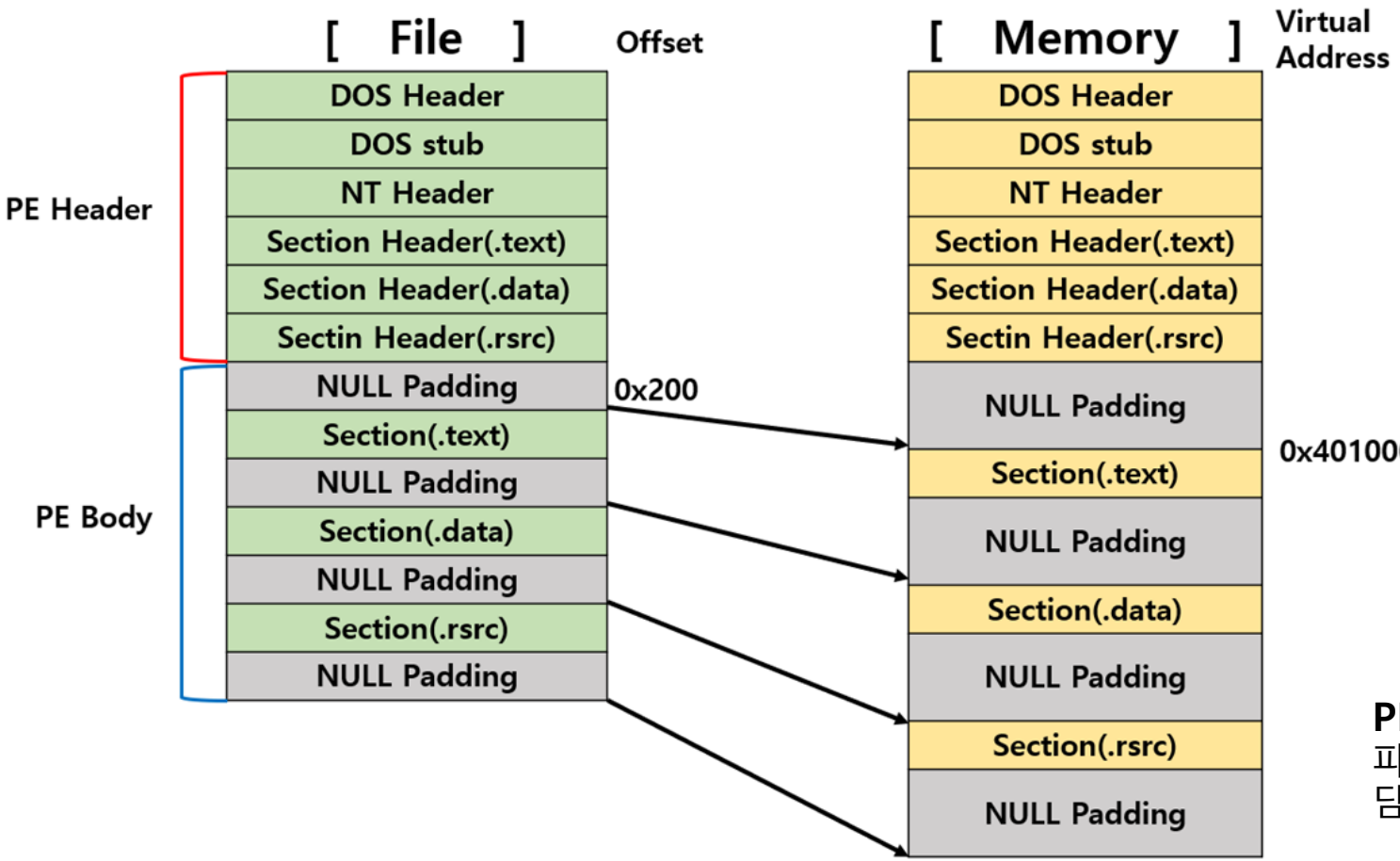
.symtab: 프로그램 속 모든 심볼

```
hyuna@hyuna:~$
hyuna@hyuna:~$ readelf -s hello

Symbol table '.dynsym' contains 10 entries:
   Num:    Value          Size Type    Bind   Vis      Ndx Name
     0: 0000000000000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 00000000000005b8     0 SECTION LOCAL  DEFAULT   11 .init
     2: 0000000000020000     0 SECTION LOCAL  DEFAULT   22 .data
     3: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND _[...]@GLIBC_2.34 (2)
     4: 0000000000000000     0 NOTYPE  WEAK   DEFAULT  UND _ITM_deregisterT[...]
     5: 0000000000000000     0 FUNC    WEAK   DEFAULT  UND _[...]@GLIBC_2.17 (3)
     6: 0000000000000000     0 NOTYPE  WEAK   DEFAULT  UND __gmon_start__
     7: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND abort@GLIBC_2.17 (3)
     8: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND puts@GLIBC_2.17 (3)
     9: 0000000000000000     0 NOTYPE  WEAK   DEFAULT  UND _ITM_registerTMC[...]

Symbol table '.symtab' contains 94 entries:
   Num:    Value          Size Type    Bind   Vis      Ndx Name
     0: 0000000000000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 0000000000000318     0 SECTION LOCAL  DEFAULT    1 .interp
     2: 0000000000000338     0 SECTION LOCAL  DEFAULT    2 .note.gnu.build-id
     3: 0000000000000358     0 SECTION LOCAL  DEFAULT    3 .note.ABI-tag
     4: 0000000000000378     0 SECTION LOCAL  DEFAULT    4 .gnu.hash
     5: 0000000000000398     0 SECTION LOCAL  DEFAULT    5 .dynsym
     6: 0000000000000438     0 SECTION LOCAL  DEFAULT    6 .dynstr
     7: 00000000000004d6     0 SECTION LOCAL  DEFAULT    7 .gnu.version
     8: 00000000000004e8     0 SECTION LOCAL  DEFAULT    8 .gnu.version_r
     9: 0000000000000508     0 SECTION LOCAL  DEFAULT    9 .rela.dyn
    10: 0000000000000568     0 SECTION LOCAL  DEFAULT   10 .rela.plt
    11: 00000000000005b8     0 SECTION LOCAL  DEFAULT   11 .init
    12: 00000000000005d0     0 SECTION LOCAL  DEFAULT   12 .plt
    13: 0000000000000640     0 SECTION LOCAL  DEFAULT   13 .text
    14: 0000000000000758     0 SECTION LOCAL  DEFAULT   14 .fini
    15: 0000000000000770     0 SECTION LOCAL  DEFAULT   15 .rodata
    16: 00000000000007a0     0 SECTION LOCAL  DEFAULT   16 .eh_frame_hdr
    17: 0000000000000818     0 SECTION LOCAL  DEFAULT   17 .eh_frame
    18: 0000000000020000     0 SECTION LOCAL  DEFAULT   18 .init_array
    19: 0000000000020008     0 SECTION LOCAL  DEFAULT   19 .fini_array
    20: 0000000000020010     0 SECTION LOCAL  DEFAULT   20 .dynamic
    21: 0000000000020000     0 SECTION LOCAL  DEFAULT   21 .got
    22: 0000000000020010     0 SECTION LOCAL  DEFAULT   22 .data
    23: 0000000000020020     0 SECTION LOCAL  DEFAULT   23 .bss
    24: 0000000000000000     0 SECTION LOCAL  DEFAULT   24 .comment
    25: 0000000000000000     0 FILE    LOCAL  DEFAULT  ABS Scrt1.o
    26: 0000000000000278     0 NOTYPE  LOCAL  DEFAULT    3 $d
    51: 0000000000000818     0 NOTYPE  LOCAL  DEFAULT   17 $d
    52: 0000000000020018     0 NOTYPE  LOCAL  DEFAULT   23 $d
    53: 0000000000000000     0 FILE    LOCAL  DEFAULT  ABS hello.c
    54: 000000000002001c     0 NOTYPE  LOCAL  DEFAULT   23 $d
    55: 0000000000020020     4 OBJECT  LOCAL  DEFAULT   23 bss2
    56: 0000000000020010     0 NOTYPE  LOCAL  DEFAULT   22 $d
    57: 0000000000020014     4 OBJECT  LOCAL  DEFAULT   22 data2
    58: 00000000000007a0     0 NOTYPE  LOCAL  DEFAULT   15 $d
    59: 0000000000000758     0 NOTYPE  LOCAL  DEFAULT   13 $x
    60: 0000000000000880     0 NOTYPE  LOCAL  DEFAULT   17 $d
    61: 0000000000000000     0 FILE    LOCAL  DEFAULT  ABS crtstuff.c
    62: 00000000000008a0     0 NOTYPE  LOCAL  DEFAULT   17 $d
    63: 00000000000008a0     0 OBJECT  LOCAL  DEFAULT   17 __FRAME_END__
    64: 0000000000000000     0 FILE    LOCAL  DEFAULT  ABS
    65: 000000000001fda0     0 OBJECT  LOCAL  DEFAULT  ABS _DYNAMIC
    66: 00000000000007b0     0 NOTYPE  LOCAL  DEFAULT   16 __GNU_EH_FRAME_HDR
    67: 000000000001ffd0     0 OBJECT  LOCAL  DEFAULT  ABS _GLOBAL_OFFSET_TABLE_
```

PE(Portable Executable)

- Windows 운영 체제에서 실행 가능한 파일 형식
- .exe, .dll, .sys …

PE파일의 종류들	
종류	**주요 확장자**
실행 계열	EXE, SCR
라이브러리 계열	DLL, OCX, CPL, DRV
드라이브 계열	SYS, VXD
오브젝트 계열	OBJ

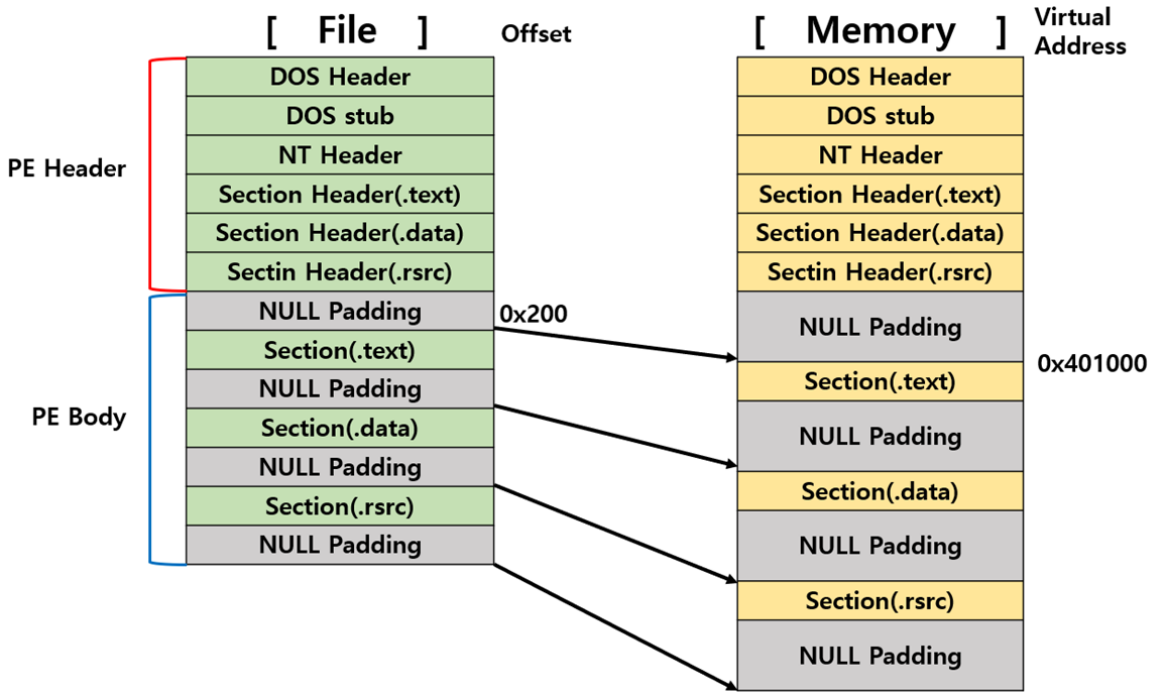


PE Header
파일이 실행되기 전에 필요한 메타데이터나 구조 정보를 담고 있는 부분(구조체 형태)

PE Body
실제 실행코드와 데이터가 포함

NULL Padding

- 최소단위: 컴퓨터가 데이터를 저장하고 처리할 때, 효율적으로 동작하도록 설정한 기준 크기(ex: 0x10)
- 섹션의 시작점: 최소 단위의 배수(ex: 0x10, 0x20)
- 섹션 시작 주소가 배수에 맞지 않으면, **빈 공간**을 만들어 NULL로 채움
  (섹션 시작 주소가 0x25라면, 0x30으로 맞추기 위해 0x26 ~ 0x2F를 NULL로 채움)



VA = RVA + ImageBase

VA
- 가상 메모리의 절대주소
- 프로그램 실행 시, 메모리 상에서 데이터가 실제로 위치한 곳

RVA
- ImageBase에서의 상대주소
- ImageBase에서 얼마나 떨어져 있는지를 나타내므로, 재배치 시에도 주소가 쉽게 계산

ImageBase
- 기준 위치(PE 파일이 메모리에 올라갈 때 0으로 초기화)

감사합니다