WhoIs 리버싱 스터디

Mungsul

오늘 할 것

- PE 파일 포맷
- DLL (Dynamic Linking Library)

실습 대상

- 대상 바이너리

https://drive.google.com/file/d/1s7tq7tOmnsBRVelSCZWGMQxTi1sb1aS-/view?usp=sharing

- CFF Explorer

https://ntcore.com/?page_id=388

PE

- Portable Executable

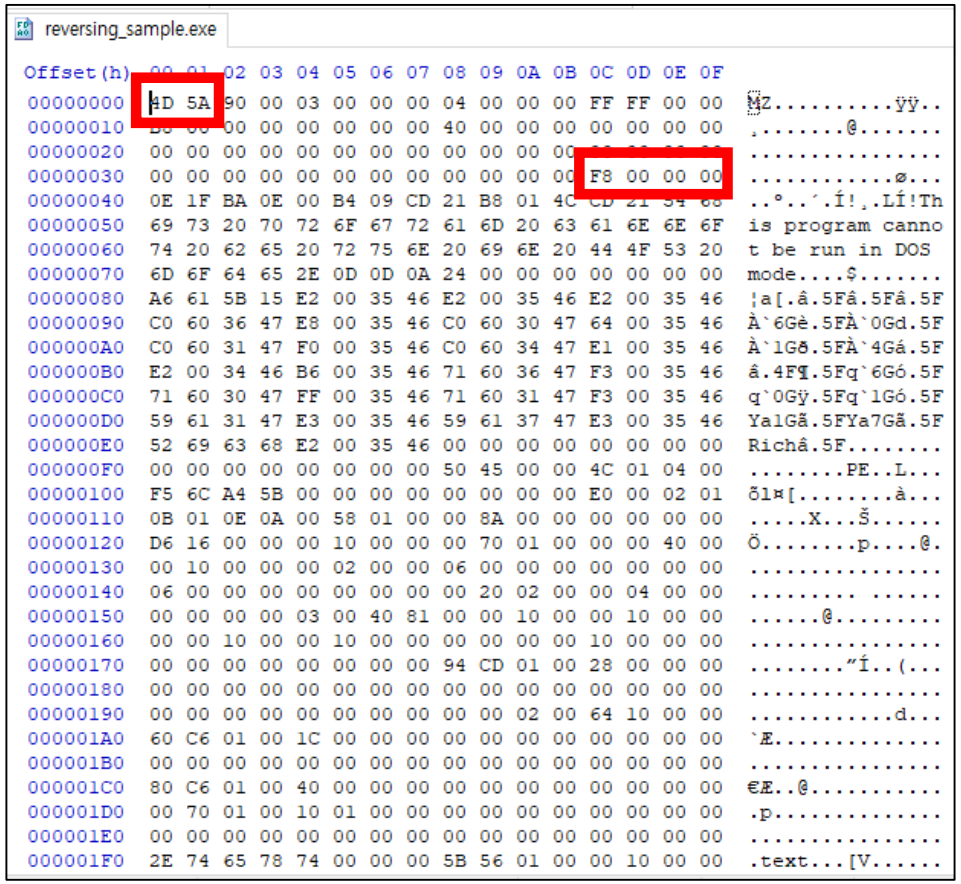
- Windows 에서 사용되는 실행파일 구조 (EXE, DLL, SYS 등)

- https://ko.wikipedia.org/wiki/PE_%ED%8F%AC%EB%A7%B7

PE Header

- Image DOS Header
- DOS Stub
- Image NT Header

Image DOS Header

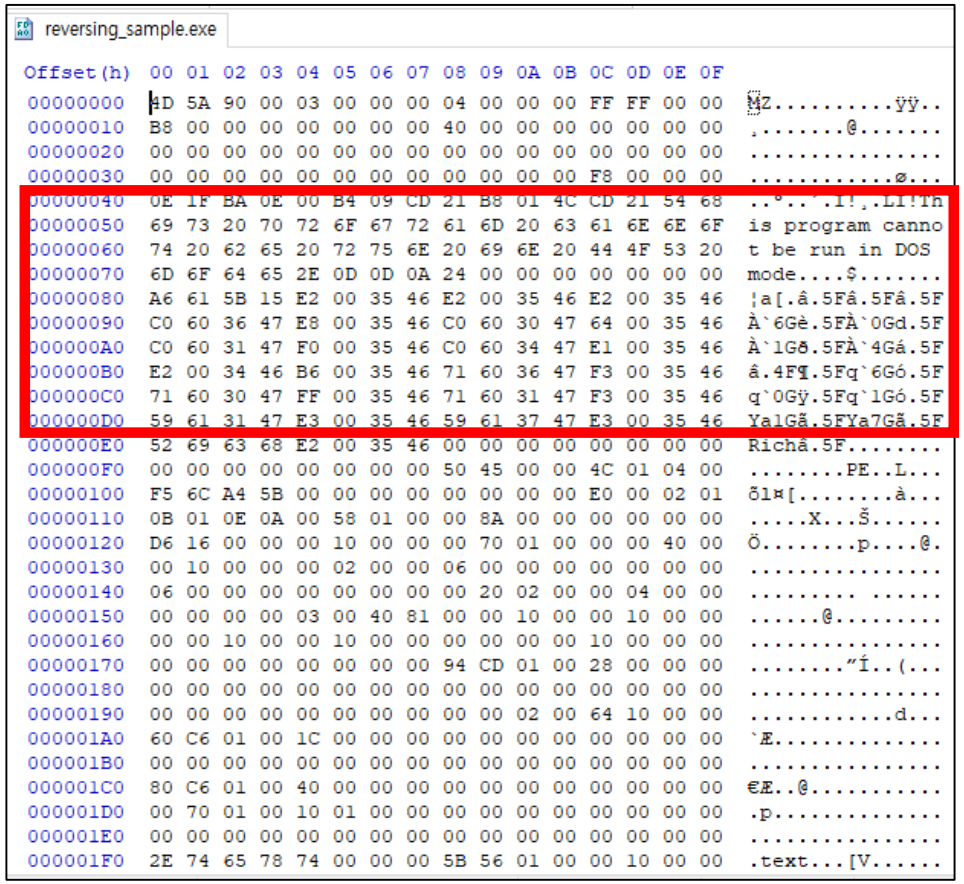

reversing_sample.exe

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000   4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
00000010   B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ¸.......@.......
00000020   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000030   00 00 00 00 00 00 00 00 00 00 00 00 F8 00 00 00  ............ø...
00000040   0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
00000050   69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
00000060   74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS
00000070   6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
00000080   A6 61 5B 15 E2 00 35 46 E2 00 35 46 E2 00 35 46  ¦a[.â.5Fâ.5Fâ.5F
00000090   C0 60 36 47 E8 00 35 46 C0 60 30 47 64 00 35 46  À`6Gè.5FÀ`0Gd.5F
000000A0   C0 60 31 47 F0 00 35 46 C0 60 34 47 E1 00 35 46  À`1Gð.5FÀ`4Gá.5F
000000B0   E2 00 34 46 B6 00 35 46 71 60 36 47 F3 00 35 46  â.4F¶.5Fq`6Gó.5F
000000C0   71 60 30 47 FF 00 35 46 71 60 31 47 F3 00 35 46  q`0Gÿ.5Fq`1Gó.5F
000000D0   59 61 31 47 E3 00 35 46 59 61 37 47 E3 00 35 46  Ya1Gã.5FYa7Gã.5F
000000E0   52 69 63 68 E2 00 35 46 00 00 00 00 00 00 00 00  Richâ.5F........
000000F0   00 00 00 00 00 00 00 00 50 45 00 00 4C 01 04 00  ........PE..L...
00000100   F5 6C A4 5B 00 00 00 00 00 00 00 00 E0 00 02 01  õl¤[........à...
00000110   0B 01 0E 0A 00 58 01 00 00 8A 00 00 00 00 00 00  .....X...Š......
00000120   D6 16 00 00 00 10 00 00 00 70 01 00 00 00 40 00  Ö........p....@.
00000130   00 10 00 00 00 02 00 00 06 00 00 00 00 00 00 00  ................
00000140   06 00 00 00 00 00 00 00 00 20 02 00 00 04 00 00  ......... ......
00000150   00 00 00 00 03 00 40 81 00 00 10 00 00 10 00 00  ......@.........
00000160   00 00 10 00 00 10 00 00 00 00 00 00 10 00 00 00  ................
00000170   00 00 00 00 00 00 00 00 94 CD 01 00 28 00 00 00  ........”Í..(...
00000180   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000190   00 00 00 00 00 00 00 00 00 00 02 00 64 10 00 00  ............d...
000001A0   60 C6 01 00 1C 00 00 00 00 00 00 00 00 00 00 00  `Æ..............
000001B0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001C0   80 C6 01 00 40 00 00 00 00 00 00 00 00 00 00 00  €Æ..@...........
000001D0   00 70 01 00 10 01 00 00 00 00 00 00 00 00 00 00  .p..............
000001E0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001F0   2E 74 65 78 74 00 00 00 5B 56 01 00 00 10 00 00  .text...[V......
```

```
typedef struct _IMAGE_DOS_HEADER {  // DOS .EXE header
    USHORT e_magic;         // Magic number
    USHORT e_cblp;          // Bytes on last page of file
    USHORT e_cp;            // Pages in file
    USHORT e_crlc;          // Relocations
    USHORT e_cparhdr;       // Size of header in paragraphs
    USHORT e_minalloc;      // Minimum extra paragraphs needed
    USHORT e_maxalloc;      // Maximum extra paragraphs needed
    USHORT e_ss;            // Initial (relative) SS value
    USHORT e_sp;            // Initial SP value
    USHORT e_csum;          // Checksum
    USHORT e_ip;            // Initial IP value
    USHORT e_cs;            // Initial (relative) CS value
    USHORT e_lfarlc;        // File address of relocation table
    USHORT e_ovno;          // Overlay number
    USHORT e_res[4];        // Reserved words
    USHORT e_oemid;         // OEM identifier (for e_oeminfo)
    USHORT e_oeminfo;       // OEM information; e_oemid specific
    USHORT e_res2[10];      // Reserved words
    LONG   e_lfanew;        // File address of new exe header
  } IMAGE_DOS_HEADER, *PIMAGE_DOS_HEADER;
```

e_magic => 0x4D5A
e_lfanew => NT header의 시작 오프셋.
왼쪽 이미지 기준에서는 0xF8

DOS Stub

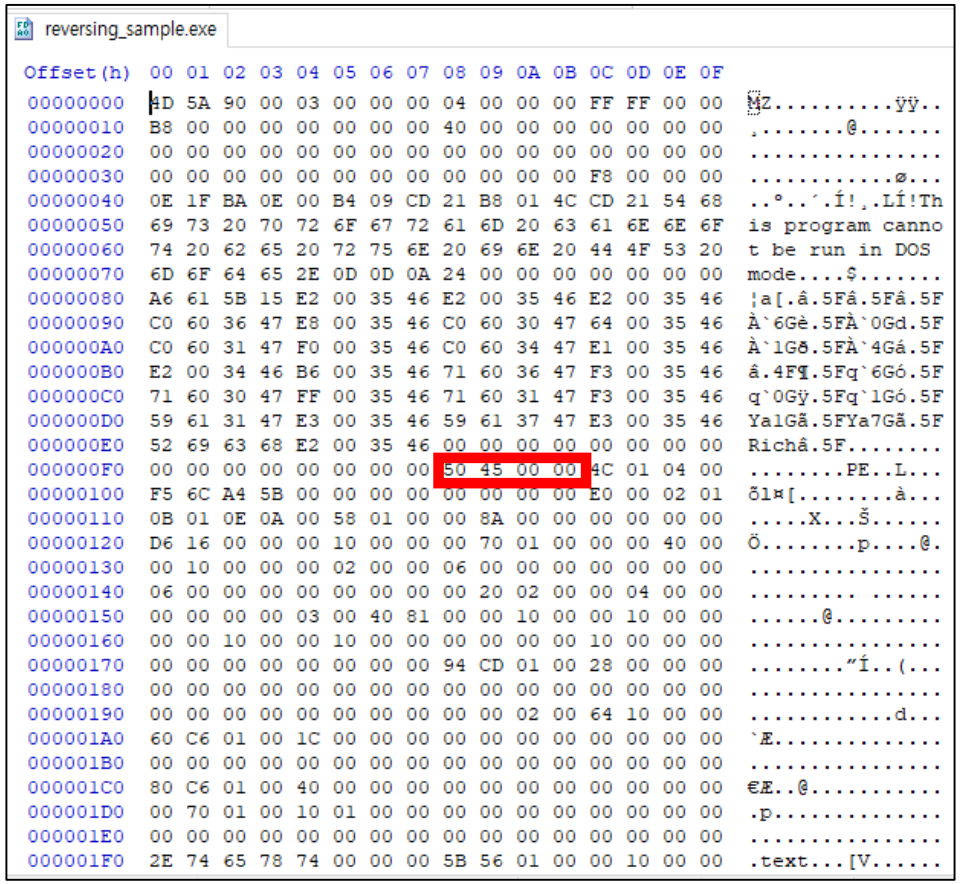

reversing_sample.exe

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000   4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
00000010   B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ¸.......@.......
00000020   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000030   00 00 00 00 00 00 00 00 00 00 00 00 F8 00 00 00  ............ø...
00000040   0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
00000050   69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
00000060   74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS
00000070   6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
00000080   A6 61 5B 15 E2 00 35 46 E2 00 35 46 E2 00 35 46  ¦a[.â.5Fâ.5Fâ.5F
00000090   C0 60 36 47 E8 00 35 46 C0 60 30 47 64 00 35 46  À`6Gè.5FÀ`0Gd.5F
000000A0   C0 60 31 47 F0 00 35 46 C0 60 34 47 E1 00 35 46  À`1Gð.5FÀ`4Gá.5F
000000B0   E2 00 34 46 B6 00 35 46 71 60 36 47 F3 00 35 46  â.4F¶.5Fq`6Gó.5F
000000C0   71 60 30 47 FF 00 35 46 71 60 31 47 F3 00 35 46  q`0Gÿ.5Fq`1Gó.5F
000000D0   59 61 31 47 E3 00 35 46 59 61 37 47 E3 00 35 46  Ya1Gã.5FYa7Gã.5F
000000E0   52 69 63 68 E2 00 35 46 00 00 00 00 00 00 00 00  Richâ.5F........
000000F0   00 00 00 00 00 00 00 00 50 45 00 00 4C 01 04 00  ........PE..L...
00000100   F5 6C A4 5B 00 00 00 00 00 00 00 00 E0 00 02 01  õl¤[........à...
00000110   0B 01 0E 0A 00 58 01 00 00 8A 00 00 00 00 00 00  .....X...Š......
00000120   D6 16 00 00 00 10 00 00 00 70 01 00 00 00 40 00  Ö........p....@.
00000130   00 10 00 00 00 02 00 00 06 00 00 00 00 00 00 00  ................
00000140   06 00 00 00 00 00 00 00 00 20 02 00 00 04 00 00  ......... ......
00000150   00 00 00 00 03 00 40 81 00 00 10 00 00 10 00 00  ......@.........
00000160   00 00 10 00 00 10 00 00 00 00 00 00 10 00 00 00  ................
00000170   00 00 00 00 00 00 00 00 94 CD 01 00 28 00 00 00  ........”Í..(...
00000180   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000190   00 00 00 00 00 00 00 00 00 00 02 00 64 10 00 00  ............d...
000001A0   60 C6 01 00 1C 00 00 00 00 00 00 00 00 00 00 00  `Æ..............
000001B0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001C0   80 C6 01 00 40 00 00 00 00 00 00 00 00 00 00 00  €Æ..@...........
000001D0   00 70 01 00 10 01 00 00 00 00 00 00 00 00 00 00  .p..............
000001E0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001F0   2E 74 65 78 74 00 00 00 5B 56 01 00 00 10 00 00  .text...[V......
```

DOS 시절 실행 파일 포맷과 호환성을 맞추기 위해 들어가 있는 데이터.

16bit assembly로
This program cannot be run in DOS mode 를 출력하는 코드가 짜여져있음.

Image NT Header


reversing_sample.exe

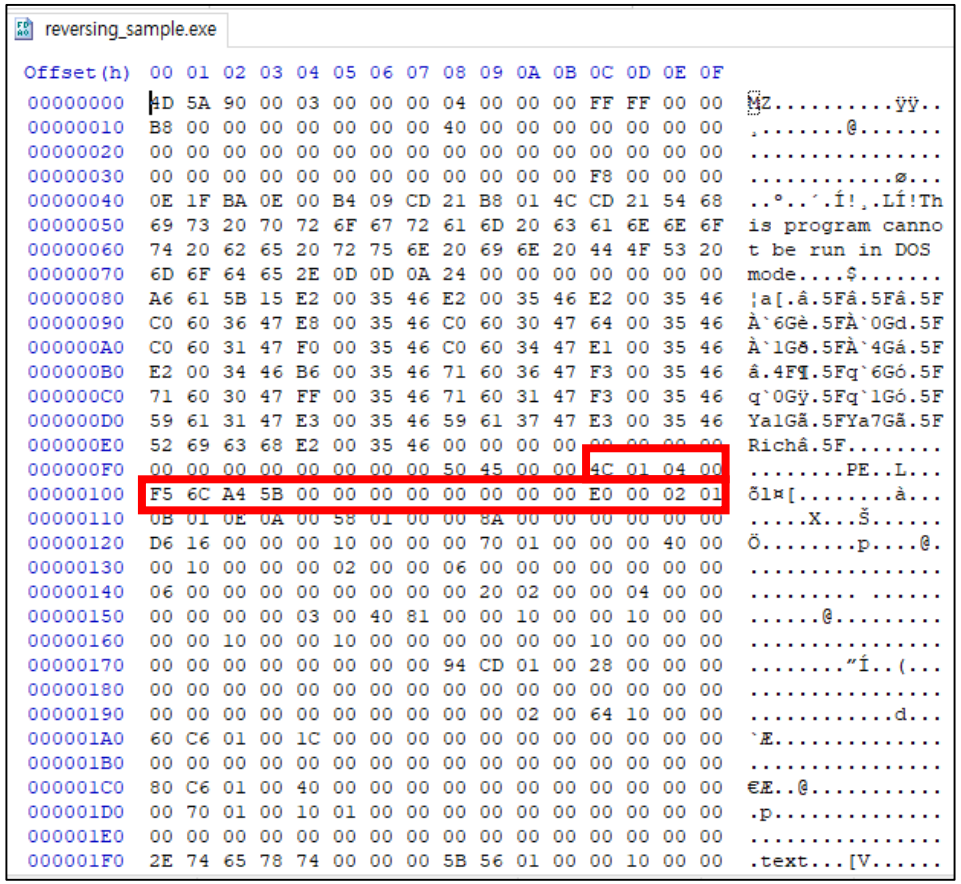
```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
00000010  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ¸.......@.......
00000020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000030  00 00 00 00 00 00 00 00 00 00 00 00 F8 00 00 00  ............ø...
00000040  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
00000050  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
00000060  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS
00000070  6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
00000080  A6 61 5B 15 E2 00 35 46 E2 00 35 46 E2 00 35 46  ¦a[.â.5Fâ.5Fâ.5F
00000090  C0 60 36 47 E8 00 35 46 C0 60 30 47 64 00 35 46  À`6Gè.5FÀ`0Gd.5F
000000A0  C0 60 31 47 F0 00 35 46 C0 60 34 47 E1 00 35 46  À`1Gð.5FÀ`4Gá.5F
000000B0  E2 00 34 46 B6 00 35 46 71 60 36 47 F3 00 35 46  â.4F¶.5Fq`6Gó.5F
000000C0  71 60 30 47 FF 00 35 46 71 60 31 47 F3 00 35 46  q`0Gÿ.5Fq`1Gó.5F
000000D0  59 61 31 47 E3 00 35 46 59 61 37 47 E3 00 35 46  Ya1Gã.5FYa7Gã.5F
000000E0  52 69 63 68 E2 00 35 46 00 00 00 00 00 00 00 00  Richâ.5F........
000000F0  00 00 00 00 00 00 00 00 50 45 00 00 4C 01 04 00  ........PE..L...
00000100  F5 6C A4 5B 00 00 00 00 00 00 00 00 E0 00 02 01  õl¤[........à...
00000110  0B 01 0E 0A 00 58 01 00 00 8A 00 00 00 00 00 00  .....X...Š......
00000120  D6 16 00 00 00 10 00 00 00 70 01 00 00 00 40 00  Ö........p....@.
00000130  00 10 00 00 00 02 00 00 06 00 00 00 00 00 00 00  ................
00000140  06 00 00 00 00 00 00 00 00 20 02 00 00 04 00 00  ......... ......
00000150  00 00 00 00 03 00 40 81 00 00 10 00 00 10 00 00  ......@.........
00000160  00 00 10 00 00 10 00 00 00 00 00 00 10 00 00 00  ................
00000170  00 00 00 00 00 00 00 00 94 CD 01 00 28 00 00 00  ........"Í..(...
00000180  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000190  00 00 00 00 00 00 00 00 00 00 02 00 64 10 00 00  ............d...
000001A0  60 C6 01 00 1C 00 00 00 00 00 00 00 00 00 00 00  `Æ..............
000001B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001C0  80 C6 01 00 40 00 00 00 00 00 00 00 00 00 00 00  €Æ..@...........
000001D0  00 70 01 00 10 01 00 00 00 00 00 00 00 00 00 00  .p..............
000001E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001F0  2E 74 65 78 74 00 00 00 5B 56 01 00 00 10 00 00  .text...[V......
```

```
typedef struct _IMAGE_NT_HEADERS {
    DWORD Signature; // 50 45 00 00
    IMAGE_FILE_HEADER FileHeader;
    IMAGE_OPTIONAL_HEADER32 OptionalHeader;
} IMAGE_NT_HEADERS32, *PIMAGE_NT_HEADERS32;
```

NT Header의 시작은 0x5045 => PE

Signature 뒤에 IMAGE_FILE_HEADER와
IMAGE_OPTIONAL_HEADER32가 옴

IMAGE_FILE_HEADER

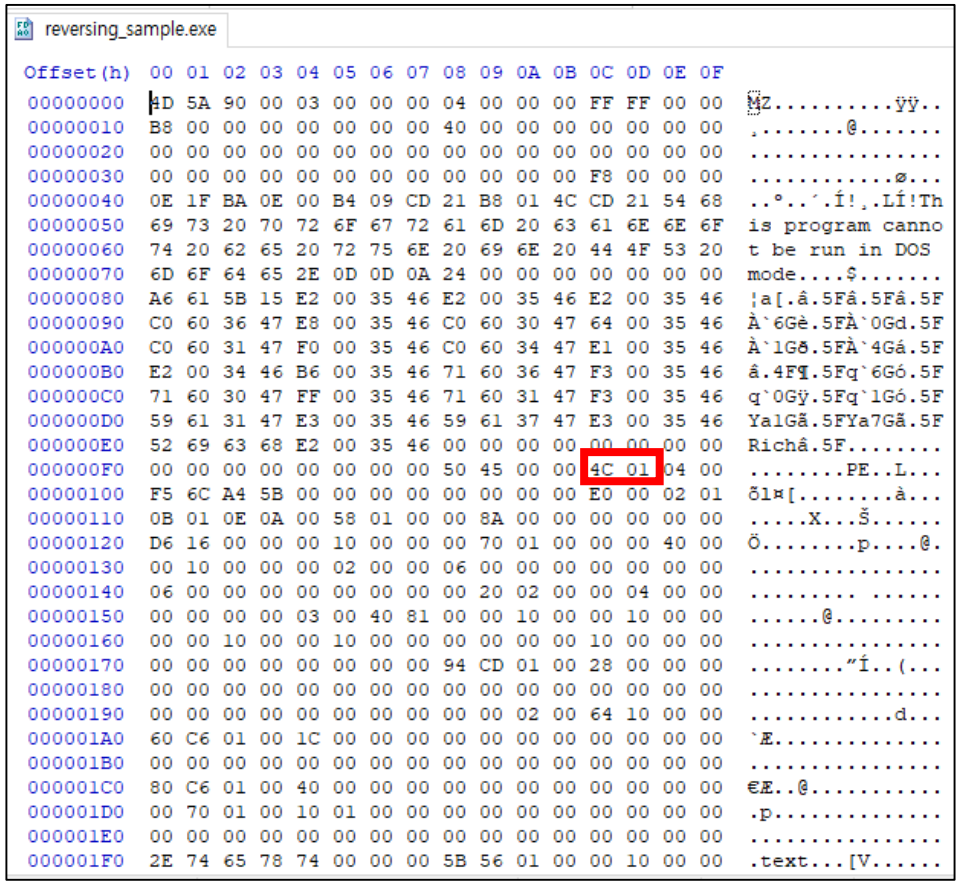

reversing_sample.exe

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000   4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
00000010   B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ¸.......@.......
00000020   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000030   00 00 00 00 00 00 00 00 00 00 00 00 F8 00 00 00  ............ø...
00000040   0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
00000050   69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
00000060   74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS
00000070   6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
00000080   A6 61 5B 15 E2 00 35 46 E2 00 35 46 E2 00 35 46  ¦a[.â.5Fâ.5Fâ.5F
00000090   C0 60 36 47 E8 00 35 46 C0 60 30 47 64 00 35 46  À`6Gè.5FÀ`0Gd.5F
000000A0   C0 60 31 47 F0 00 35 46 C0 60 34 47 E1 00 35 46  À`1Gð.5FÀ`4Gá.5F
000000B0   E2 00 34 46 B6 00 35 46 71 60 36 47 F3 00 35 46  â.4F¶.5Fq`6Gó.5F
000000C0   71 60 30 47 FF 00 35 46 71 60 31 47 F3 00 35 46  q`0Gÿ.5Fq`1Gó.5F
000000D0   59 61 31 47 E3 00 35 46 59 61 37 47 E3 00 35 46  Ya1Gã.5FYa7Gã.5F
000000E0   52 69 63 68 E2 00 35 46 00 00 00 00 00 00 00 00  Richâ.5F........
000000F0   00 00 00 00 00 00 00 00 50 45 00 00 4C 01 04 00  ........PE..L...
00000100   F5 6C A4 5B 00 00 00 00 00 00 00 00 E0 00 02 01  õl¤[........à...
00000110   0B 01 0E 0A 00 58 01 00 00 8A 00 00 00 00 00 00  .....X...Š......
00000120   D6 16 00 00 00 10 00 00 00 70 01 00 00 00 40 00  Ö........p....@.
00000130   00 10 00 00 00 02 00 00 06 00 00 00 00 00 00 00  ................
00000140   06 00 00 00 00 00 00 00 00 20 02 00 00 04 00 00  ......... ......
00000150   00 00 00 00 03 00 40 81 00 00 10 00 00 10 00 00  ......@.........
00000160   00 00 10 00 00 10 00 00 00 00 00 00 10 00 00 00  ................
00000170   00 00 00 00 00 00 00 00 94 CD 01 00 28 00 00 00  ........”Í..(...
00000180   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000190   00 00 00 00 00 00 00 00 00 00 02 00 64 10 00 00  ............d...
000001A0   60 C6 01 00 1C 00 00 00 00 00 00 00 00 00 00 00  `Æ..............
000001B0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001C0   80 C6 01 00 40 00 00 00 00 00 00 00 00 00 00 00  €Æ..@...........
000001D0   00 70 01 00 10 01 00 00 00 00 00 00 00 00 00 00  .p..............
000001E0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001F0   2E 74 65 78 74 00 00 00 5B 56 01 00 00 10 00 00  .text...[V......
```

```
typedef struct _IMAGE_FILE_HEADER {
    WORD Machine;
    WORD NumberOfSections;
    DWORD TimeDateStamp;
    DWORD PointerToSymbolTable;
    DWORD NumberOfSymbols;
    WORD SizeOfOptionalHeader;
    WORD Characteristics;
} IMAGE_FILE_HEADER, *PIMAGE_FILE_HEADER;
```

Machine => 동작 가능한 머신의 종류
Characteristics => PE 파일 속성

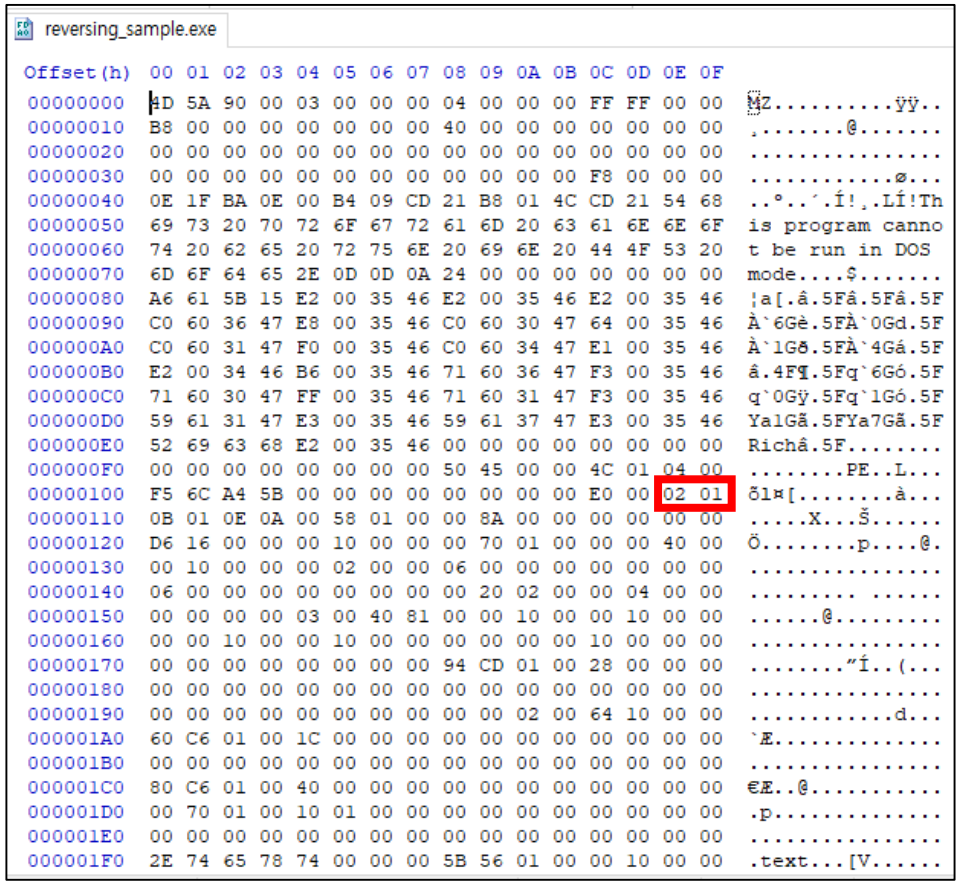
IMAGE_FILE_HEADER


reversing_sample.exe

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000   4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
00000010   B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ........@.......
00000020   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000030   00 00 00 00 00 00 00 00 00 00 00 00 F8 00 00 00  ............ø...
00000040   0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
00000050   69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
00000060   74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS
00000070   6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
00000080   A6 61 5B 15 E2 00 35 46 E2 00 35 46 E2 00 35 46  ¦a[.â.5Fâ.5Fâ.5F
00000090   C0 60 36 47 E8 00 35 46 C0 60 30 47 64 00 35 46  À`6Gè.5FÀ`0Gd.5F
000000A0   C0 60 31 47 F0 00 35 46 C0 60 34 47 E1 00 35 46  À`1Gð.5FÀ`4Gá.5F
000000B0   E2 00 34 46 B6 00 35 46 71 60 36 47 F3 00 35 46  â.4F¶.5Fq`6Gó.5F
000000C0   71 60 30 47 FF 00 35 46 71 60 31 47 F3 00 35 46  q`0Gÿ.5Fq`1Gó.5F
000000D0   59 61 31 47 E3 00 35 46 59 61 37 47 E3 00 35 46  Ya1Gã.5FYa7Gã.5F
000000E0   52 69 63 68 E2 00 35 46 00 00 00 00 00 00 00 00  Richâ.5F........
000000F0   00 00 00 00 00 00 00 00 50 45 00 00 4C 01 04 00  ........PE..L...
00000100   F5 6C A4 5B 00 00 00 00 00 00 00 00 E0 00 02 01  õl¤[........à...
00000110   0B 01 0E 0A 00 58 01 00 00 8A 00 00 00 00 00 00  .....X...Š......
00000120   D6 16 00 00 00 10 00 00 00 70 01 00 00 00 40 00  Ö........p....@.
00000130   00 10 00 00 00 02 00 00 06 00 00 00 00 00 00 00  ................
00000140   06 00 00 00 00 00 00 00 00 20 02 00 00 04 00 00  ......... ......
00000150   00 00 00 00 03 00 40 81 00 00 10 00 00 10 00 00  ......@.........
00000160   00 00 10 00 00 10 00 00 00 00 00 00 10 00 00 00  ................
00000170   00 00 00 00 00 00 00 00 94 CD 01 00 28 00 00 00  ........"Í..(...
00000180   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000190   00 00 00 00 00 00 00 00 00 00 02 00 64 10 00 00  ............d...
000001A0   60 C6 01 00 1C 00 00 00 00 00 00 00 00 00 00 00  `Æ..............
000001B0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001C0   80 C6 01 00 40 00 00 00 00 00 00 00 00 00 00 00  €Æ..@...........
000001D0   00 70 01 00 10 01 00 00 00 00 00 00 00 00 00 00  .p..............
000001E0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001F0   2E 74 65 78 74 00 00 00 5B 56 01 00 00 10 00 00  .text...[V......
```

```
#define IMAGE_FILE_MACHINE_UNKNOWN           0
#define IMAGE_FILE_MACHINE_I386              0x014c // Intel 386.
#define IMAGE_FILE_MACHINE_R3000             0x0162 // MIPS little-endian, 0x160 big-endian
#define IMAGE_FILE_MACHINE_R4000             0x0166 // MIPS little-endian
#define IMAGE_FILE_MACHINE_R10000            0x0168 // MIPS little-endian
#define IMAGE_FILE_MACHINE_WCEMIPSV2         0x0169 // MIPS little-endian WCE v2
#define IMAGE_FILE_MACHINE_ALPHA             0x0184 // Alpha_AXP
#define IMAGE_FILE_MACHINE_SH3               0x01a2 // SH3 little-endian
#define IMAGE_FILE_MACHINE_SH3DSP            0x01a3
#define IMAGE_FILE_MACHINE_SH3E              0x01a4 // SH3E little-endian
#define IMAGE_FILE_MACHINE_SH4               0x01a6 // SH4 little-endian
#define IMAGE_FILE_MACHINE_SH5               0x01a8 // SH5
#define IMAGE_FILE_MACHINE_ARM               0x01c0 // ARM Little-Endian
#define IMAGE_FILE_MACHINE_THUMB             0x01c2
#define IMAGE_FILE_MACHINE_AM33              0x01d3
#define IMAGE_FILE_MACHINE_POWERPC           0x01F0 // IBM PowerPC Little-Endian
#define IMAGE_FILE_MACHINE_POWERPCFP         0x01f1
#define IMAGE_FILE_MACHINE_IA64              0x0200 // Intel 64
#define IMAGE_FILE_MACHINE_MIPS16            0x0266 // MIPS
#define IMAGE_FILE_MACHINE_ALPHA64           0x0284 // ALPHA64
#define IMAGE_FILE_MACHINE_MIPSFPU           0x0366 // MIPS
#define IMAGE_FILE_MACHINE_MIPSFPU16         0x0466 // MIPS
#define IMAGE_FILE_MACHINE_AXP64             IMAGE_FILE_MACHINE_ALPHA64
#define IMAGE_FILE_MACHINE_TRICORE           0x0520 // Infineon
#define IMAGE_FILE_MACHINE_CEF               0x0CEF
#define IMAGE_FILE_MACHINE_EBC               0x0EBC // EFI Byte Code
#define IMAGE_FILE_MACHINE_AMD64             0x8664 // AMD64 (K8)
#define IMAGE_FILE_MACHINE_M32R              0x9041 // M32R little-endian
#define IMAGE_FILE_MACHINE_CEE               0xC0EE
```

IMAGE_FILE_HEADER

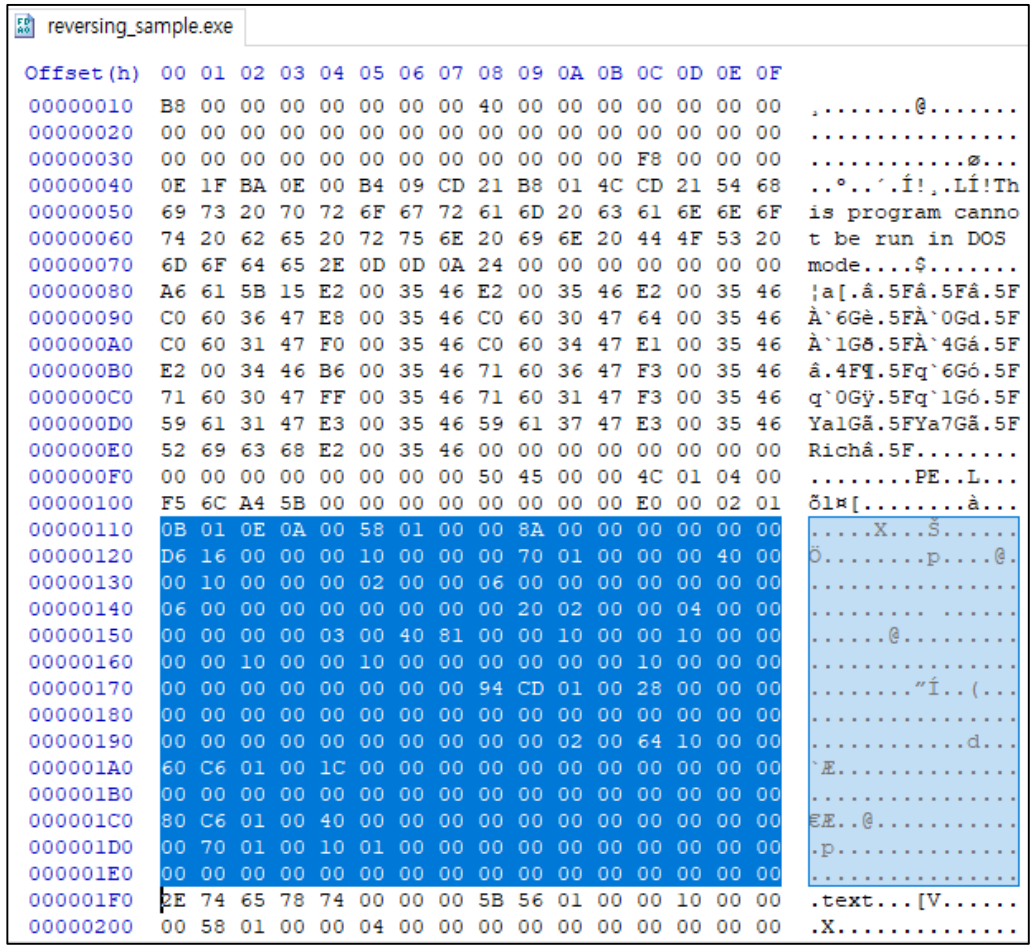

reversing_sample.exe

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
00000010  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ¸.......@.......
00000020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000030  00 00 00 00 00 00 00 00 00 00 00 00 F8 00 00 00  ............ø...
00000040  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
00000050  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
00000060  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS
00000070  6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
00000080  A6 61 5B 15 E2 00 35 46 E2 00 35 46 E2 00 35 46  ¦a[.â.5Fâ.5Fâ.5F
00000090  C0 60 36 47 E8 00 35 46 C0 60 30 47 64 00 35 46  À`6Gè.5FÀ`0Gd.5F
000000A0  C0 60 31 47 F0 00 35 46 C0 60 34 47 E1 00 35 46  À`1Gð.5FÀ`4Gá.5F
000000B0  E2 00 34 46 B6 00 35 46 71 60 36 47 F3 00 35 46  â.4F¶.5Fq`6Gó.5F
000000C0  71 60 30 47 FF 00 35 46 71 60 31 47 F3 00 35 46  q`0Gÿ.5Fq`1Gó.5F
000000D0  59 61 31 47 E3 00 35 46 59 61 37 47 E3 00 35 46  Ya1Gã.5FYa7Gã.5F
000000E0  52 69 63 68 E2 00 35 46 00 00 00 00 00 00 00 00  Richâ.5F........
000000F0  00 00 00 00 00 00 00 00 50 45 00 00 4C 01 04 00  ........PE..L...
00000100  F5 6C A4 5B 00 00 00 00 00 00 00 00 E0 00 02 01  õl¤[........à...
00000110  0B 01 0E 0A 00 58 01 00 00 8A 00 00 00 00 00 00  .....X...Š......
00000120  D6 16 00 00 00 10 00 00 00 70 01 00 00 00 40 00  Ö........p....@.
00000130  00 10 00 00 00 02 00 00 06 00 00 00 00 00 00 00  ................
00000140  06 00 00 00 00 00 00 00 00 20 02 00 00 04 00 00  ......... ......
00000150  00 00 00 00 03 00 40 81 00 00 10 00 00 10 00 00  ......@.........
00000160  00 00 10 00 00 10 00 00 00 00 00 00 10 00 00 00  ................
00000170  00 00 00 00 00 00 00 00 94 CD 01 00 28 00 00 00  ........"Í..(...
00000180  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000190  00 00 00 00 00 00 00 00 00 00 02 00 64 10 00 00  ............d...
000001A0  60 C6 01 00 1C 00 00 00 00 00 00 00 00 00 00 00  `Æ..............
000001B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001C0  80 C6 01 00 40 00 00 00 00 00 00 00 00 00 00 00  €Æ..@...........
000001D0  00 70 01 00 10 01 00 00 00 00 00 00 00 00 00 00  .p..............
000001E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001F0  2E 74 65 78 74 00 00 00 5B 56 01 00 00 10 00 00  .text...[V......
```

```
#define IMAGE_FILE_RELOCS_STRIPPED          0x0001 // Relocation info stripped from file.
#define IMAGE_FILE_EXECUTABLE_IMAGE         0x0002 // File is executable (i.e. no unresolved externel
references).
#define IMAGE_FILE_LINE_NUMS_STRIPPED       0x0004 // Line nunbers stripped from file.
#define IMAGE_FILE_LOCAL_SYMS_STRIPPED      0x0008 // Local symbols stripped from file.
#define IMAGE_FILE_AGGRESIVE_WS_TRIM        0x0010 // Agressively trim working set
#define IMAGE_FILE_LARGE_ADDRESS_AWARE      0x0020 // App can handle >2gb addresses
#define IMAGE_FILE_BYTES_REVERSED_LO        0x0080 // Bytes of machine word are reversed.
#define IMAGE_FILE_32BIT_MACHINE            0x0100 // 32 bit word machine.
#define IMAGE_FILE_DEBUG_STRIPPED           0x0200 // Debugging info stripped from file in .DBG file
#define IMAGE_FILE_REMOVABLE_RUN_FROM_SWAP  0x0400 // If Image is on removable media, copy and run
from the swap file.
#define IMAGE_FILE_NET_RUN_FROM_SWAP        0x0800 // If Image is on Net, copy and run from the swap file.
#define IMAGE_FILE_SYSTEM                   0x1000 // System File.
#define IMAGE_FILE_DLL                      0x2000 // File is a DLL.
#define IMAGE_FILE_UP_SYSTEM_ONLY           0x4000 // File should only be run on a UP machine
#define IMAGE_FILE_BYTES_REVERSED_HI        0x8000 // Bytes of machine word are reversed.
```

0x0102 => 0x02와 0x0100의 OR 연산으로 나타내짐

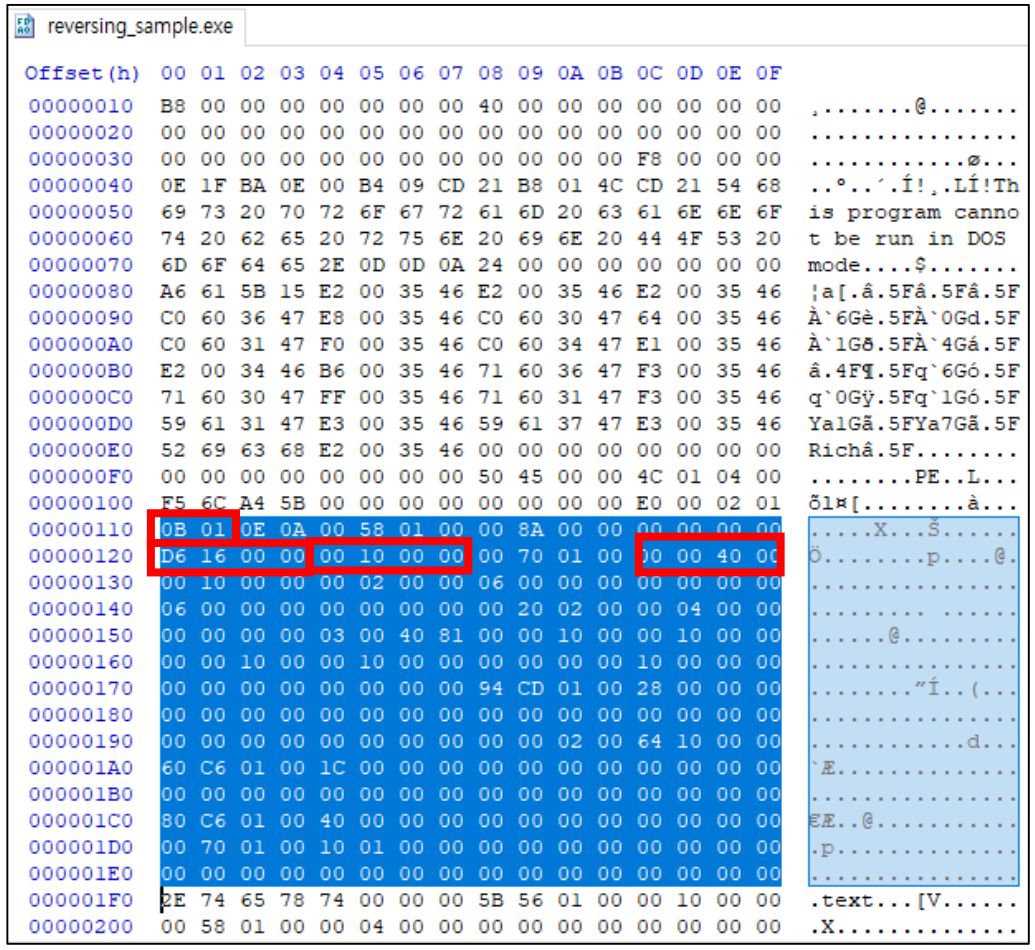
IMAGE_OPTIONAL_HEADER


reversing_sample.exe

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000010  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ¸.......@.......
00000020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000030  00 00 00 00 00 00 00 00 00 00 00 00 F8 00 00 00  ............ø...
00000040  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
00000050  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
00000060  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS
00000070  6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
00000080  A6 61 5B 15 E2 00 35 46 E2 00 35 46 E2 00 35 46  ¦a[.â.5Fâ.5Fâ.5F
00000090  C0 60 36 47 E8 00 35 46 C0 60 30 47 64 00 35 46  À`6Gè.5FÀ`0Gd.5F
000000A0  C0 60 31 47 F0 00 35 46 C0 60 34 47 E1 00 35 46  À`1Gð.5FÀ`4Gá.5F
000000B0  E2 00 34 46 B6 00 35 46 71 60 36 47 F3 00 35 46  â.4F¶.5Fq`6Gó.5F
000000C0  71 60 30 47 FF 00 35 46 71 60 31 47 F3 00 35 46  q`0Gÿ.5Fq`1Gó.5F
000000D0  59 61 31 47 E3 00 35 46 59 61 37 47 E3 00 35 46  Ya1Gã.5FYa7Gã.5F
000000E0  52 69 63 68 E2 00 35 46 00 00 00 00 00 00 00 00  Richâ.5F........
000000F0  00 00 00 00 00 00 00 00 50 45 00 00 4C 01 04 00  ........PE..L...
00000100  F5 6C A4 5B 00 00 00 00 00 00 00 00 E0 00 02 01  õl¤[........à...
00000110  0B 01 0E 0A 00 58 01 00 00 8A 00 00 00 00 00 00  .....X...Š......
00000120  D6 16 00 00 00 10 00 00 00 70 01 00 00 00 40 00  Ö........p....@.
00000130  00 10 00 00 00 02 00 00 06 00 00 00 00 00 00 00  ................
00000140  06 00 00 00 00 00 00 00 00 20 02 00 00 04 00 00  ......... ......
00000150  00 00 00 00 03 00 40 81 00 00 10 00 00 10 00 00  ......@.........
00000160  00 00 10 00 00 10 00 00 00 00 00 00 10 00 00 00  ................
00000170  00 00 00 00 00 00 00 00 94 CD 01 00 28 00 00 00  ........”Í..(...
00000180  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000190  00 00 00 00 00 00 00 00 00 00 02 00 64 10 00 00  ............d...
000001A0  60 C6 01 00 1C 00 00 00 00 00 00 00 00 00 00 00  `Æ..............
000001B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001C0  80 C6 01 00 40 00 00 00 00 00 00 00 00 00 00 00  €Æ..@...........
000001D0  00 70 01 00 10 01 00 00 00 00 00 00 00 00 00 00  .p..............
000001E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001F0  2E 74 65 78 74 00 00 00 5B 56 01 00 00 10 00 00  .text...[V......
00000200  00 58 01 00 00 04 00 00 00 00 00 00 00 00 00 00  .X..............
```

```
typedef struct _IMAGE_OPTIONAL_HEADER {
    //
    // Standard fields.
    //
    USHORT  Magic;
    UCHAR   MajorLinkerVersion;
    UCHAR   MinorLinkerVersion;
    ULONG   SizeOfCode;
    ULONG   SizeOfInitializedData;
    ULONG   SizeOfUninitializedData;
    ULONG   AddressOfEntryPoint;
    ULONG   BaseOfCode;
    ULONG   BaseOfData;
    //
    // NT additional fields.
    //
    ULONG   ImageBase;
    ULONG   SectionAlignment;
    ULONG   FileAlignment;
    USHORT  MajorOperatingSystemVersion;
    USHORT  MinorOperatingSystemVersion;
    USHORT  MajorImageVersion;
    USHORT  MinorImageVersion;
    USHORT  MajorSubsystemVersion;
    USHORT  MinorSubsystemVersion;
    ULONG   Reserved1;
    ULONG   SizeOfImage;
    ULONG   SizeOfHeaders;
    ULONG   CheckSum;
    USHORT  Subsystem;
    USHORT  DllCharacteristics;
    ULONG   SizeOfStackReserve;
    ULONG   SizeOfStackCommit;
    ULONG   SizeOfHeapReserve;
    ULONG   SizeOfHeapCommit;
    ULONG   LoaderFlags;
    ULONG   NumberOfRvaAndSizes;
    IMAGE_DATA_DIRECTORY DataDirectory[IMAGE_NUMBEROF_DIRECTORY_ENTRIES];
} IMAGE_OPTIONAL_HEADER, *PIMAGE_OPTIONAL_HEADER;
```

IMAGE_OPTIONAL_HEADER


reversing_sample.exe

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000010  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ¸.......@.......
00000020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000030  00 00 00 00 00 00 00 00 00 00 00 00 F8 00 00 00  ............ø...
00000040  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
00000050  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
00000060  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS
00000070  6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
00000080  A6 61 5B 15 E2 00 35 46 E2 00 35 46 E2 00 35 46  ¦a[.â.5Fâ.5Fâ.5F
00000090  C0 60 36 47 E8 00 35 46 C0 60 30 47 64 00 35 46  À`6Gè.5FÀ`0Gd.5F
000000A0  C0 60 31 47 F0 00 35 46 C0 60 34 47 E1 00 35 46  À`1Gð.5FÀ`4Gá.5F
000000B0  E2 00 34 46 B6 00 35 46 71 60 36 47 F3 00 35 46  â.4F¶.5Fq`6Gó.5F
000000C0  71 60 30 47 FF 00 35 46 71 60 31 47 F3 00 35 46  q`0Gÿ.5Fq`1Gó.5F
000000D0  59 61 31 47 E3 00 35 46 59 61 37 47 E3 00 35 46  Ya1Gã.5FYa7Gã.5F
000000E0  52 69 63 68 E2 00 35 46 00 00 00 00 00 00 00 00  Richâ.5F........
000000F0  00 00 00 00 00 00 00 00 50 45 00 00 4C 01 04 00  ........PE..L...
00000100  F5 6C A4 5B 00 00 00 00 00 00 00 00 E0 00 02 01  õl¤[........à...
00000110  0B 01 0E 0A 00 58 01 00 00 8A 00 00 00 00 00 00  .....X...Š......
00000120  D6 16 00 00 00 10 00 00 00 70 01 00 00 00 40 00  Ö........p....@.
00000130  00 10 00 00 00 02 00 00 06 00 00 00 00 00 00 00  ................
00000140  06 00 00 00 00 00 00 00 00 20 02 00 00 04 00 00  ......... ......
00000150  00 00 00 00 03 00 40 81 00 00 10 00 00 10 00 00  ......@.........
00000160  00 00 10 00 00 10 00 00 00 00 00 00 10 00 00 00  ................
00000170  00 00 00 00 00 00 00 00 94 CD 01 00 28 00 00 00  ........”Í..(...
00000180  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000190  00 00 00 00 00 00 00 00 00 00 02 00 64 10 00 00  ............d...
000001A0  60 C6 01 00 1C 00 00 00 00 00 00 00 00 00 00 00  `Æ..............
000001B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001C0  80 C6 01 00 40 00 00 00 00 00 00 00 00 00 00 00  €Æ..@...........
000001D0  00 70 01 00 10 01 00 00 00 00 00 00 00 00 00 00  .p..............
000001E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001F0  2E 74 65 78 74 00 00 00 5B 56 01 00 00 10 00 00  .text...[V......
00000200  00 58 01 00 00 04 00 00 00 00 00 00 00 00 00 00  .X..............
```

MAGIC => 0x010B 일 경우 IMAGE_OPTIONAL_HEADER32
0x020B일 경우 IMAGE_OPTIONAL_HEADER64

AddressOfEntryPoint => PE가 로딩되면 시작될 주소. (RVA)

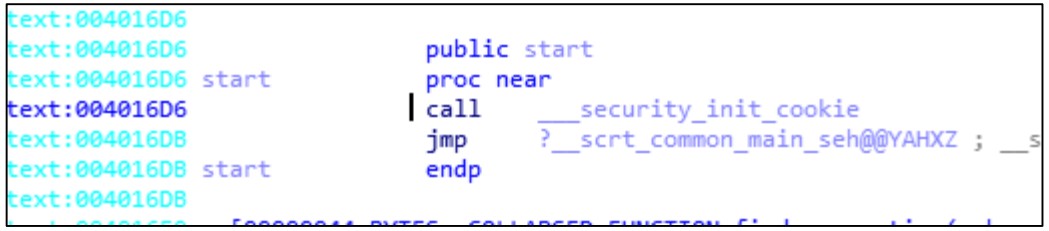
BaseOfCode => 코드 영역의 Base 주소
ImageBase => 0x00400000, 메모리에 올라갈 때 Base 주소

즉, Code 영역은 ImageBase + BaseOfCode 값임.

EntryPoint = ImageBase + AddressOfEntryPoint
0x4016d6 = 0x400000 + 0x16d6

보통 BaseOfCode 값이랑 .text 섹션 VA와 같음.

EntryPoint / VA / RVA



Entrypoint : 0x004016D6

VA : Virtual Address, 메모리에 올라갈 때의 절대 주소

RVA : Relative Virtual Address, 이미지 베이스로부터의 상대 주소

VA로부터 파일 오프셋 구하기

File Offset = VA – (ImageBase + SectionVA) + SectionRaw

SectionRaw(.text) = 0x400
SectionVA(.text) = 0x1000

Section은 뒷장부터 내용이 나옴.

Name	Virtual Size	Virtual Address	Raw Size	Raw Address
Byte[8]	Dword	Dword	Dword	Dword
.text	0001565B	00001000	00015800	00000400
.rdata	000063AA	00017000	00006400	00015C00
.data	000012F0	0001E000	00000A00	0001C000
.reloc	00001064	00020000	00001200	0001CA00

```
:00401060 ; int __cdecl main(int argc, cons
:00401060 _main           proc near
:00401060
:00401060 var_28          = dword ptr -28h
:00401060 var_24          = dword ptr -24h
:00401060 var_1F          = dword ptr -1Fh
:00401060 var_1B          = dword ptr -1Bh
:00401060 var_17          = word ptr -17h
:00401060 var_14          = byte ptr -14h
:00401060 var_4           = dword ptr -4
:00401060 argc            = dword ptr  8
:00401060 argv            = dword ptr  0Ch
:00401060 envp            = dword ptr  10h
:00401060
:00401060                 push    ebp
:00401061                 mov     ebp, esp
:00401063                 sub     esp, 28h
```

ImageBase : 0x400000
SectionVA(.text): 0x1000

VA : 0x401060

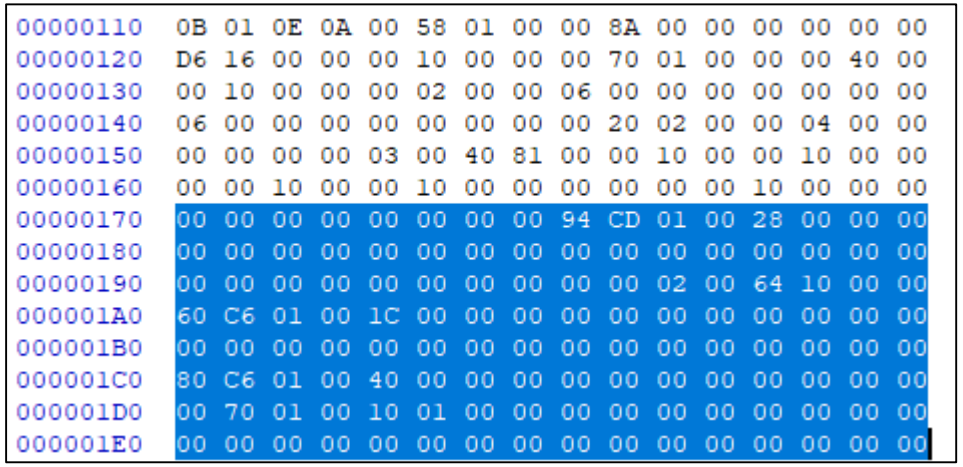
File Offset = ?

섹션을 배우고 다시 봅시다.

IMAGE_DATA_DIRECTORY

- 각 섹션의 VA와 Size 를 지정

```
typedef struct _IMAGE_DATA_DIRECTORY {
DWORD VirtualAddress;
DWORD Size;
} IMAGE_DATA_DIRECTORY, *PIMAGE_DATA_DIRECTORY;
```



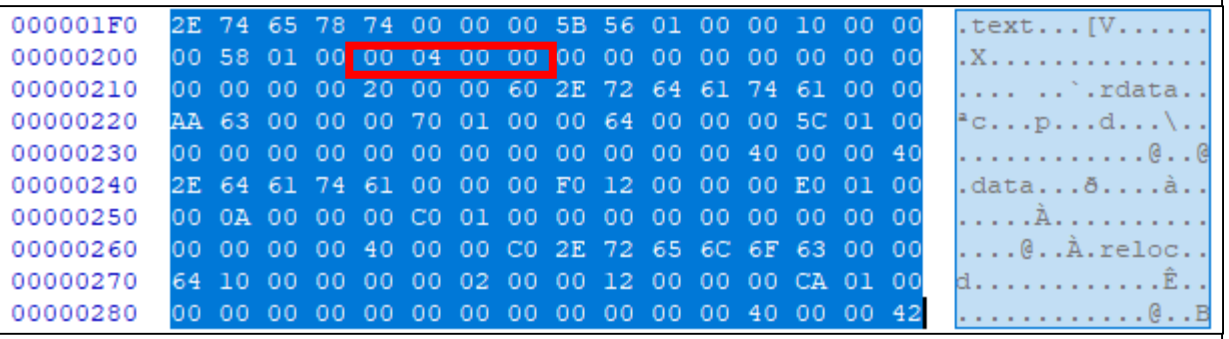
```
00000110  0B 01 0E 0A 00 58 01 00 00 8A 00 00 00 00 00 00
00000120  D6 16 00 00 00 10 00 00 00 70 01 00 00 00 40 00
00000130  00 10 00 00 00 02 00 00 06 00 00 00 00 00 00 00
00000140  06 00 00 00 00 00 00 00 00 20 02 00 00 04 00 00
00000150  00 00 00 00 03 00 40 81 00 00 10 00 00 10 00 00
00000160  00 00 10 00 00 10 00 00 00 00 00 00 10 00 00 00
00000170  00 00 00 00 00 00 00 00 94 CD 01 00 28 00 00 00
00000180  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000190  00 00 00 00 00 00 00 00 00 00 02 00 64 10 00 00
000001A0  60 C6 01 00 1C 00 00 00 00 00 00 00 00 00 00 00
000001B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001C0  80 C6 01 00 40 00 00 00 00 00 00 00 00 00 00 00
000001D0  00 70 01 00 10 01 00 00 00 00 00 00 00 00 00 00
000001E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Value
IMAGE_DIRECTORY_ENTRY_EXPORT
IMAGE_DIRECTORY_ENTRY_IMPORT
IMAGE_DIRECTORY_ENTRY_RESOURCE
IMAGE_DIRECTORY_ENTRY_EXCEPTION
IMAGE_DIRECTORY_ENTRY_SECURITY
IMAGE_DIRECTORY_ENTRY_BASERELOC
IMAGE_DIRECTORY_ENTRY_DEBUG
IMAGE_DIRECTORY_ENTRY_ARCHITECTURE
IMAGE_DIRECTORY_ENTRY_GLOBALPTR
IMAGE_DIRECTORY_ENTRY_TLS
IMAGE_DIRECTORY_ENTRY_LOAD_CONFIG
IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT
IMAGE_DIRECTORY_ENTRY_IAT
IMAGE_DIRECTORY_ENTRY_DELAY_IMPORT
IMAGE_DIRECTORY_ENTRY_COM_DESCRIPTOR

PE Body

- Section Header
- Section body

IMAGE_SECTION_HEADER



```
000001F0  2E 74 65 78 74 00 00 00 5B 56 01 00 00 10 00 00  .text...[V......
00000200  00 58 01 00 00 04 00 00 00 00 00 00 00 00 00 00  .X..............
00000210  00 00 00 00 20 00 00 60 2E 72 64 61 74 61 00 00  .... ..`.rdata..
00000220  AA 63 00 00 00 70 01 00 00 64 00 00 00 5C 01 00  ªc...p...d...\..
00000230  00 00 00 00 00 00 00 00 00 00 00 00 40 00 00 40  ............@..@
00000240  2E 64 61 74 61 00 00 00 F0 12 00 00 00 E0 01 00  .data...ð....à..
00000250  00 0A 00 00 00 C0 01 00 00 00 00 00 00 00 00 00  .....À..........
00000260  00 00 00 00 40 00 00 C0 2E 72 65 6C 6F 63 00 00  ....@..À.reloc..
00000270  64 10 00 00 00 00 02 00 00 12 00 00 00 CA 01 00  d............Ê..
00000280  00 00 00 00 00 00 00 00 00 00 00 00 40 00 00 42  ............@..B
```

섹션 이름은 8글자 까지..

.text 섹션의 PointerToRawData : 0x400

```
typedef struct
_IMAGE_SECTION_HEADER {
  BYTE Name[IMAGE_SIZEOF_SHORT_NAME];
union {
  DWORD PhysicalAddress;
  DWORD VirtualSize;
} Misc;
  DWORD VirtualAddress;
  DWORD SizeOfRawData;
  DWORD PointerToRawData;
  DWORD PointerToRelocations;
  DWORD PointerToLinenumbers;
  WORD NumberOfRelocations;
  WORD NumberOfLinenumbers;
  DWORD Characteristics;
} IMAGE_SECTION_HEADER, *PIMAGE_SECTION_HEADER;
```

VA로부터 파일 오프셋 구하기

```
:00401060 ; int __cdecl main(int argc, cons
:00401060 _main           proc near
:00401060
:00401060 var_28          = dword ptr -28h
:00401060 var_24          = dword ptr -24h
:00401060 var_1F          = dword ptr -1Fh
:00401060 var_1B          = dword ptr -1Bh
:00401060 var_17          = word ptr -17h
:00401060 var_14          = byte ptr -14h
:00401060 var_4           = dword ptr -4
:00401060 argc            = dword ptr  8
:00401060 argv            = dword ptr  0Ch
:00401060 envp            = dword ptr  10h
:00401060
:00401060                 push    ebp
:00401061                 mov     ebp, esp
:00401063                 sub     esp, 28h
```

```
00000400  55 8B EC 51 C7 45 FC 00 00 00 00 C7 45 FC 00 00  U‹ìQÇEü....ÇEü..
00000410  00 00 EB 09 8B 45 FC 83 C0 01 89 45 FC 8B 4D 0C  ..ë.‹EüƒÀ.‰E ü‹M.
00000420  51 E8 BA 77 00 00 83 C4 04 39 45 FC 7D 1C 8B 55  Qè°w..ƒÄ.9Eü}.‹U
00000430  0C 03 55 FC 0F BE 02 8B 4D 08 03 4D FC 0F BE 11  ..Uü.¾.‹M..Mü.¾.
00000440  3B C2 74 04 33 C0 EB 07 EB CA B8 01 00 00 00 8B  ;Ât.3Àë.ëÊ¸....‹
00000450  E5 5D C3 CC CC CC CC CC CC CC CC CC CC CC CC CC  å]ÃÌÌÌÌÌÌÌÌÌÌÌÌ
00000460  55 8B EC 83 EC 28 A1 18 E0 41 00 33 C5 89 45 FC  U‹ìƒì(¡.àA.3Å‰Eü
00000470  C6 45 DC 00 33 C0 89 45 DD 89 45 E1 89 45 E5 66  ÆEÜ.3À‰EÝ‰Eá‰Eåf
```

.text 섹션.

ImageBase : 0x400000
SectionVA(.text): 0x1000

VA : 0x401060

PointerToRawData(.text) : 0x400

File Offset = ?

Section Body

- Section Header에 적힌 크기 만큼 차지하는 데이터임
- Section Header에 있는 PointerToRawData 값에 따른 Offset부터 SizeOfRawData 만큼 존재하는 값.
- Section크기는 IMAGE_OPTIONAL_HEADER에 있는 Section alignment 값의 배수만큼 되어야함.

주요 IMAGE_DATA_DIRECTORY

Value
IMAGE_DIRECTORY_ENTRY_EXPORT
IMAGE_DIRECTORY_ENTRY_IMPORT
IMAGE_DIRECTORY_ENTRY_RESOURCE
IMAGE_DIRECTORY_ENTRY_EXCEPTION
IMAGE_DIRECTORY_ENTRY_SECURITY
IMAGE_DIRECTORY_ENTRY_BASERELOC
IMAGE_DIRECTORY_ENTRY_DEBUG
IMAGE_DIRECTORY_ENTRY_ARCHITECTURE
IMAGE_DIRECTORY_ENTRY_GLOBALPTR
IMAGE_DIRECTORY_ENTRY_TLS
IMAGE_DIRECTORY_ENTRY_LOAD_CONFIG
IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT
IMAGE_DIRECTORY_ENTRY_IAT
IMAGE_DIRECTORY_ENTRY_DELAY_IMPORT
IMAGE_DIRECTORY_ENTRY_COM_DESCRIPTOR

EXPORT : 다른 PE에서 사용 가능하도록 함수를 Export함

IMPORT : 다른 PE로 부터 하기위해 DLL을 Import함

RESOURCE : 이 PE가 사용하는 외부 리소스

IAT : DLL에서 어떠한 함수를 사용할지 정해놓음.

IAT(Import Address Table)

```
text:0040129A 55                                  push    ebp
text:0040129B 8B EC                               mov     ebp, esp
text:0040129D 6A 00                               push    0               ; lpTopLevelExceptionFilter
text:0040129F FF 15 04 70 41 00                   call    ds:SetUnhandledExceptionFilter
text:004012A5 FF 75 08                            push    [ebp+ExceptionInfo] ; ExceptionInfo
text:004012A8 FF 15 00 70 41 00                   call    ds:UnhandledExceptionFilter
text:004012AE 68 09 04 00 C0                      push    0C0000409h      ; uExitCode
text:004012B3 FF 15 08 70 41 00                   call    ds:GetCurrentProcess
text:004012B9 50                                  push    eax             ; hProcess
text:004012BA FF 15 0C 70 41 00                   call    ds:TerminateProcess
text:004012C0 5D                                  pop     ebp
text:004012C1 C3                                  retn
```

```
idata:00417000                    ; LONG __stdcall UnhandledExceptionFilter(struct _EXCEPTION_POINTERS *ExceptionInfo)
idata:00417000 ?? ?? ?? ??        extrn UnhandledExceptionFilter:dword
idata:00417000                                    ; CODE XREF: ___raise_securityfailure+E↑p
idata:00417000                                    ; ___scrt_fastfail+FE↑p ...
```

외부 함수를 호출 할 때, 실제 주소를 call 하는 것이 아니라
어떠한 테이블에다 배치하고 간접 호출하는 것을 볼 수 있음.

```
Disassembly:

0:   ff 15 00 00 00 00       call   DWORD PTR ds:0x0
```

IAT(Import Address Table)

- DLL 내에서 어떠한 함수를 사용할지 지정해놓은 테이블

- DLL당 하나당 하나의 IMAGE_IMPORT_DESCRIPTOR를 가짐.

- IMAGE_IMPORT_DESCRIPTOR의 시작주소는 Data Directory 중 Import 부분이 가리키고 있음.

IAT(Import Address Table)

- IMAGE_IMPORT_DESCRIPTOR 에서는 이 구조체에 해당하는 IAT를 가리키고 있음.

- Data Directory에서의 IAT 항목은 IAT들의 시작 주소를 가리킴

IMAGE_IMPORT_DESCRIPTOR

```
typedef struct _IMAGE_IMPORT_DESCRIPTOR {

  union {
    DWORD Characteristics;                // 0 for terminating null import descriptor
    DWORD OriginalFirstThunk;             // RVA to original unbound IAT (PIMAGE_THUNK_DATA)
  } DUMMYUNIONNAME;
  DWORD TimeDateStamp;                    // 0 if not bound,
        // -1 if bound, and real date₩time stamp
        // in IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT (new BIND)
        // O.W. date/time stamp of DLL bound to (Old BIND)
  DWORD ForwarderChain;          // -1 if no forwarders
  DWORD Name;
  DWORD FirstThunk;          // RVA to IAT (if bound this IAT has actual addresses)
} IMAGE_IMPORT_DESCRIPTOR;
typedef IMAGE_IMPORT_DESCRIPTOR UNALIGNED *PIMAGE_IMPORT_DESCRIPTOR;
```

NAME에 Import 할 DLL의 이름의 RVA가 적힌다.

OrigianlFirstThunk는 IMAGE_IMPORT_BY_NAME 구조체들을 가리킨다. (INT의 주소)

FirstThunk는 IAT의 RVA 주소를 가리킨다.

IMAGE_IMPORT_BY_NAME

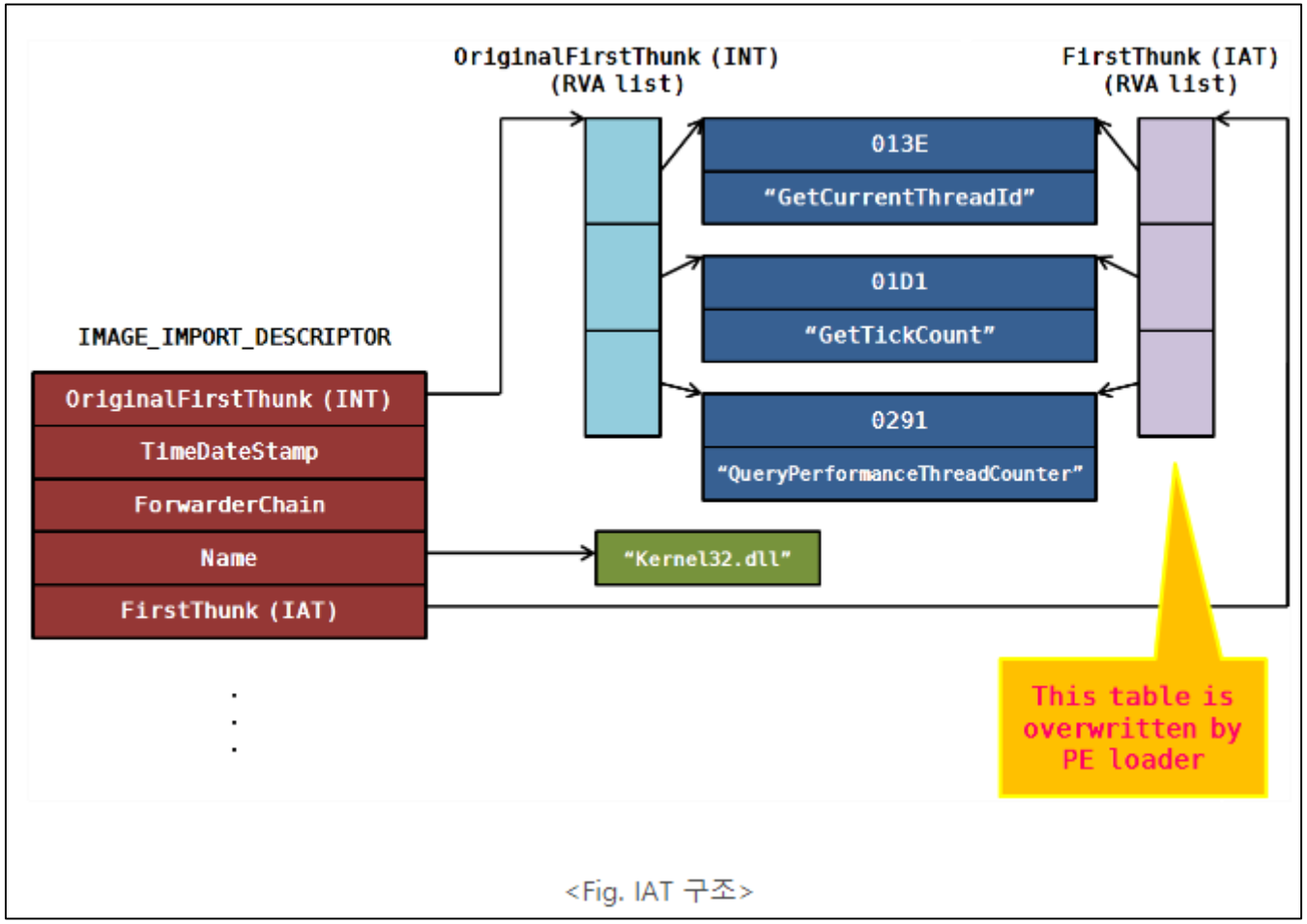
```
typedef
struct _IMAGE_IMPORT_BY_NAME {
WORD Hint; // ordinal
BYTE Name[1];
} IMAGE_IMPORT_BY_NAME,
*PIMAGE_IMPORT_BY_NAME;
```

INT(Import Name Table)는 위 구조체를 가리키고 있는 주소의 연속된 형태임.

IAT도 초기 상태에는 위 구조체를 가리키고 있는 주소의 연속된 형태로 존재함.

구해올 함수의 실제 주소를 얻어오면 IAT들은 overwrite 됨.

IAT(Import Address Table)



<Fig. IAT 구조>

참고 : http://reversecore.com/23?category=216978

IAT(Import Address Table)

- IAT가 채워지는 과정

1. IMAGE_IMPORT_DESCRIPTOR에서 어떤 DLL을 로딩하는지 확인
2. DLL 로딩 후 IMAGE_IMPORT_DESCRIPTOR의 OriginalFirstThunk(INT)를 참조
3. IMAGE_IMPORT_BY_NAME 구조체 주소들을 얻음.
4. IMAGE_IMPORT_BY_NAME 구조체들을 참조하면서 ordinal이나 name으로 로딩된 DLL 로부터 함수 시작 주소를 얻음.
5. IMAGE_IMPORT_DESCRIPTOR의 FirstThunk(IAT)에 채워넣음
6. NULL을 만날 때 까지 3~5 반복

따라가보기

- Data Directory

Import directory RVA : 0x1CD94

Import directory Size : 0x28

(IMAGE_IMPORT_DESCRIPTOR 구조체의 크기는 0x14임, 뒤 0x14만큼이 NULL이면 원소는 하나 밖에 없는 것!)

- .rdata Section Header

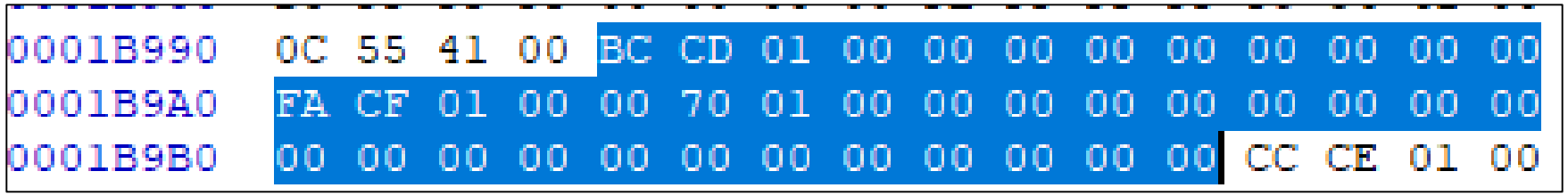
rdata VA : 0x17000

rdata PointerToRawData : 0x15C00

따라가보기

- 0x1CD94 => ".rtext" 섹션에 속해있음.

File Offset = 0x1B994 = 0x1CD94 – 0x17000 + 0x15C00



IMAGE_IMPORT_DESCRIPTOR 구조체

OriginalFirstThunk : 0x01CDBC (File Offset : 0x1B9BC)

NAME : 0x01CFFA (File Offset : 0x1BBFA)

FirstThunk : 0x17000 (File Offset : 0x15C00)

따라가보기

```
0001BBF0   65 48 61 6E 64 6C 65 57 00 00 4B 45 52 4E 45 4C   eHandleW..KERNEL
0001BC00   33 32 2E 64 6C 6C 00 00 C4 04 52 74 6C 55 6E 77   32.dll..Ä.RtlUnw
```

NAME : 0x01CFFA (File Offset : 0x1BBFA)

```
0001B9B0   00 00 00 00 00 00 00 00 00 00 00 00 CC CE 01 00
0001B9C0   E8 CE 01 00 06 CF 01 00 1A CF 01 00 2E CF 01 00
0001B9D0   4A CF 01 00 64 CF 01 00 7A CF 01 00 90 CF 01 00
0001B9E0   AA CF 01 00 C0 CF 01 00 D4 CF 01 00 E6 CF 01 00
0001B9F0   08 D0 01 00 14 D0 01 00 24 D0 01 00 34 D0 01 00
0001BA00   4C D0 01 00 64 D0 01 00 7C D0 01 00 A4 D0 01 00
0001BA10   B0 D0 01 00 BE D0 01 00 CC D0 01 00 D6 D0 01 00
0001BA20   E4 D0 01 00 F6 D0 01 00 08 D1 01 00 18 D1 01 00
0001BA30   24 D1 01 00 3A D1 01 00 50 D1 01 00 66 D1 01 00
0001BA40   74 D1 01 00 8A D1 01 00 9C D1 01 00 AE D1 01 00
0001BA50   B8 D1 01 00 C4 D1 01 00 D0 D1 01 00 E2 D1 01 00
0001BA60   F2 D1 01 00 00 D2 01 00 12 D2 01 00 1E D2 01 00
0001BA70   32 D2 01 00 42 D2 01 00 54 D2 01 00 60 D2 01 00
0001BA80   6C D2 01 00 86 D2 01 00 A0 D2 01 00 BA D2 01 00
0001BA90   CA D2 01 00 DC D2 01 00 F0 D2 01 00 00 D3 01 00
0001BAA0   12 D3 01 00 1E D3 01 00 2C D3 01 00 3A D3 01 00
0001BAB0   4E D3 01 00 5A D3 01 00 6A D3 01 00 78 D3 01 00
0001BAC0   88 D3 01 00 98 D3 01 00 00 00 00 00 9D 05 55 6E
```

OriginalFirstThunk : 0x01CDBC (File Offset : 0x1B9BC)

```
00015C00   CC CE 01 00 E8 CE 01 00 06 CF 01 00 1A CF 01 00
00015C10   2E CF 01 00 4A CF 01 00 64 CF 01 00 7A CF 01 00
00015C20   90 CF 01 00 AA CF 01 00 C0 CF 01 00 D4 CF 01 00
00015C30   E6 CF 01 00 08 D0 01 00 14 D0 01 00 24 D0 01 00
00015C40   34 D0 01 00 4C D0 01 00 64 D0 01 00 7C D0 01 00
00015C50   A4 D0 01 00 B0 D0 01 00 BE D0 01 00 CC D0 01 00
00015C60   D6 D0 01 00 E4 D0 01 00 F6 D0 01 00 08 D1 01 00
00015C70   18 D1 01 00 24 D1 01 00 3A D1 01 00 50 D1 01 00
00015C80   66 D1 01 00 74 D1 01 00 8A D1 01 00 9C D1 01 00
00015C90   AE D1 01 00 B8 D1 01 00 C4 D1 01 00 D0 D1 01 00
00015CA0   E2 D1 01 00 F2 D1 01 00 00 D2 01 00 12 D2 01 00
00015CB0   1E D2 01 00 32 D2 01 00 42 D2 01 00 54 D2 01 00
00015CC0   60 D2 01 00 6C D2 01 00 86 D2 01 00 A0 D2 01 00
00015CD0   BA D2 01 00 CA D2 01 00 DC D2 01 00 F0 D2 01 00
00015CE0   00 D3 01 00 12 D3 01 00 1E D3 01 00 2C D3 01 00
00015CF0   3A D3 01 00 4E D3 01 00 5A D3 01 00 6A D3 01 00
00015D00   78 D3 01 00 88 D3 01 00 98 D3 01 00 00 00 00 00
```

FirstThunk : 0x17000 (File Offset : 0x15C00)

초기에 FirstTunk와 OriginalFirstThunk는 데이터가 동일하다..!

따라가보기

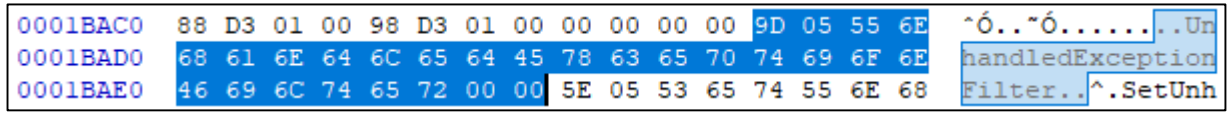
- OrigianlFirstThink(INT)들의 원소들

첫번째 원소 : 0x01CECC (File Offset : 0x1BACC)

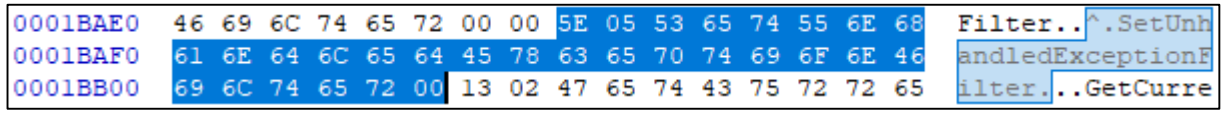
두번째 원소 : 0x01CEE8 (File Offset : 0x1BAE8)

세번째 원소 : 0x01CF06 (File Offset : 0x1BB06)

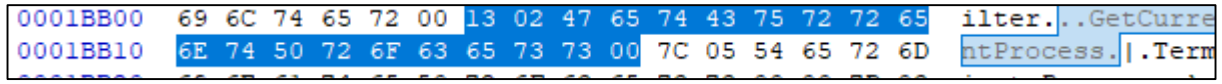
…



ordinal : 0x059D, Name : UnhandledExceptionFilter



ordinal : 0x055E, Name : SetUnhandledExceptionFilter



ordinal : 0x0213, Name : GetCurrentProcess

따라가보기

- IAT가 로딩되는 지점은 ?

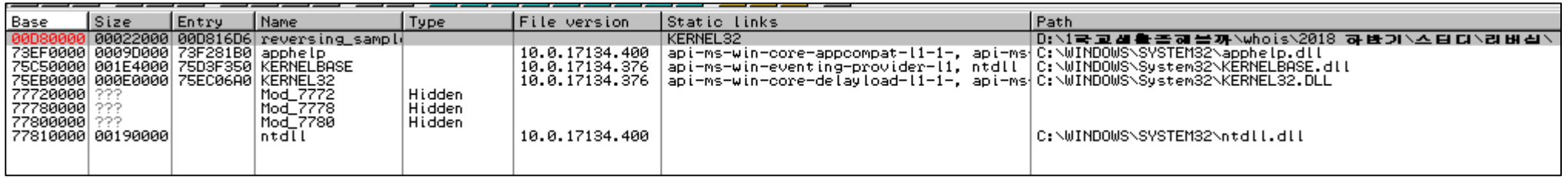
KERNEL32.DLL IAT RVA : 0x17000

ImageBase : 0x400000

즉, 이 프로그램이 시작 될 때 IAT는 0x417000에 로딩됨.

정말로 따라가보기

- Windows Vista 이상에서는 Imagebase가 랜덤으로 매핑됨.



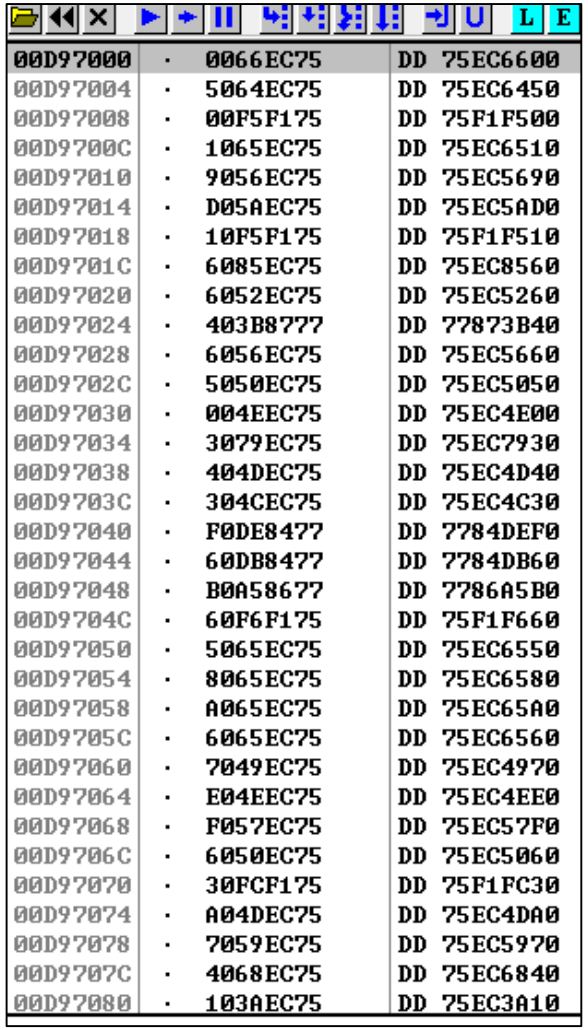
Base	Size	Entry	Name	Type	File version	Static links	Path
00D80000	00022000	00D816D6	reversing_sampl			KERNEL32	D:\1국교생활종례보고\whois\2018 하반기\스터디\리버싱\
73EF0000	0009D000	73F281B0	apphelp		10.0.17134.400	api-ms-win-core-appcompat-l1-1-, api-ms-	C:\WINDOWS\SYSTEM32\apphelp.dll
75C50000	001E4000	75D3F350	KERNELBASE		10.0.17134.376	api-ms-win-eventing-provider-l1, ntdll	C:\WINDOWS\System32\KERNELBASE.dll
75EB0000	000E0000	75EC06A0	KERNEL32		10.0.17134.376	api-ms-win-core-delayload-l1-1-, api-ms-	C:\WINDOWS\System32\KERNEL32.DLL
77720000	???		Mod_7772	Hidden			
77780000	???		Mod_7778	Hidden			
77800000	???		Mod_7780	Hidden			
77810000	00190000		ntdll		10.0.17134.400		C:\WINDOWS\SYSTEM32\ntdll.dll

로딩된 Base : 0xD80000

- 계산 식은 똑같음.

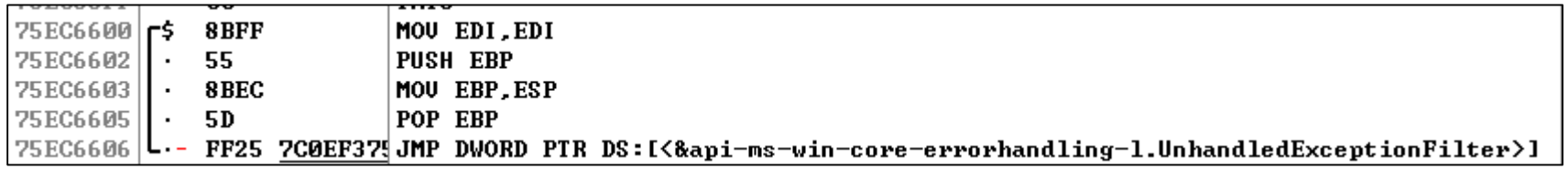
0xD80000 + 0x17000 => IAT 주소

정말로 따라가보기



00D97000	0066EC75	DD 75EC6600
00D97004	5064EC75	DD 75EC6450
00D97008	00F5F175	DD 75F1F500
00D9700C	1065EC75	DD 75EC6510
00D97010	9056EC75	DD 75EC5690
00D97014	D05AEC75	DD 75EC5AD0
00D97018	10F5F175	DD 75F1F510
00D9701C	6085EC75	DD 75EC8560
00D97020	6052EC75	DD 75EC5260
00D97024	403B8777	DD 77873B40
00D97028	6056EC75	DD 75EC5660
00D9702C	5050EC75	DD 75EC5050
00D97030	004EEC75	DD 75EC4E00
00D97034	3079EC75	DD 75EC7930
00D97038	404DEC75	DD 75EC4D40
00D9703C	304CEC75	DD 75EC4C30
00D97040	F0DE8477	DD 7784DEF0
00D97044	60DB8477	DD 7784DB60
00D97048	B0A58677	DD 7786A5B0
00D9704C	60F6F175	DD 75F1F660
00D97050	5065EC75	DD 75EC6550
00D97054	8065EC75	DD 75EC6580
00D97058	A065EC75	DD 75EC65A0
00D9705C	6065EC75	DD 75EC6560
00D97060	7049EC75	DD 75EC4970
00D97064	E04EEC75	DD 75EC4EE0
00D97068	F057EC75	DD 75EC57F0
00D9706C	6050EC75	DD 75EC5060
00D97070	30FCF175	DD 75F1FC30
00D97074	A04DEC75	DD 75EC4DA0
00D97078	7059EC75	DD 75EC5970
00D9707C	4068EC75	DD 75EC6840
00D97080	103AEC75	DD 75EC3A10

0xD80000 + 0x17000 = 0xD97000



75EC6600	8BFF	MOV EDI,EDI
75EC6602	55	PUSH EBP
75EC6603	8BEC	MOV EBP,ESP
75EC6605	5D	POP EBP
75EC6606	FF25 7C0EF375	JMP DWORD PTR DS:[<&api-ms-win-core-errorhandling-l1.UnhandledExceptionFilter>]

첫번째 원소인 0x75EC6600을 따라가보면
UnhandledExceptionFilter로 jmp하는 구문임!

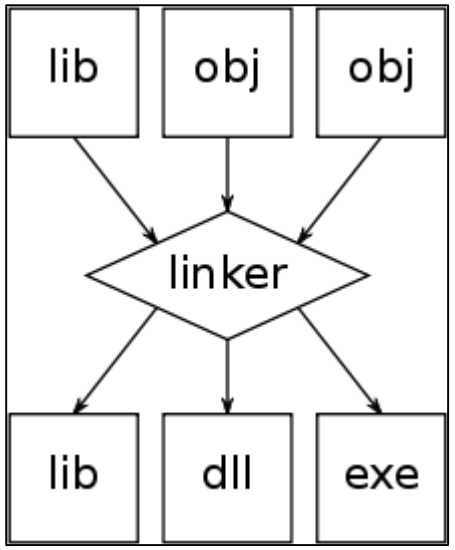
0x75EC6600은 KERNEL32.DLL의 영역임

즉, 프로그램이 로딩되면서 IAT 데이터가 바뀐 것을 확인할 수 있다.

Linking

- Object Code 를 하나로 묶는 작업!

- Object Code
  - Main Program
  - Library



Linking

- Static Linking
  - 사용할 Library 코드들을 한 실행파일에 모두 포함시키는 것.
  - 실행 파일의 크기가 커짐!
  - Library 로딩을 하지 않기 때문에 속도가 빠름
  - 다시 컴파일 할 때 속도가 느림.
- Dynamic Linking
  - 사용할 Library 코드들을 나중에 로딩해서 사용하는 것
  - 실행 파일의 크기가 Static Linking 보다는 대체로 작음
  - Library 로딩을 해야 되기 때문에 속도가 Static 보다는 빠름
  - 다시 컴파일 할 때 속도가 Static보다는 빠름

Dynamic Linking

- Windows에서는 .dll (dynamic linking library)
  - kernel32.dll
  - user32.dll
  - etc..

- Linux에서는 .so (shared object)
  - libc.so.6
  - etc..

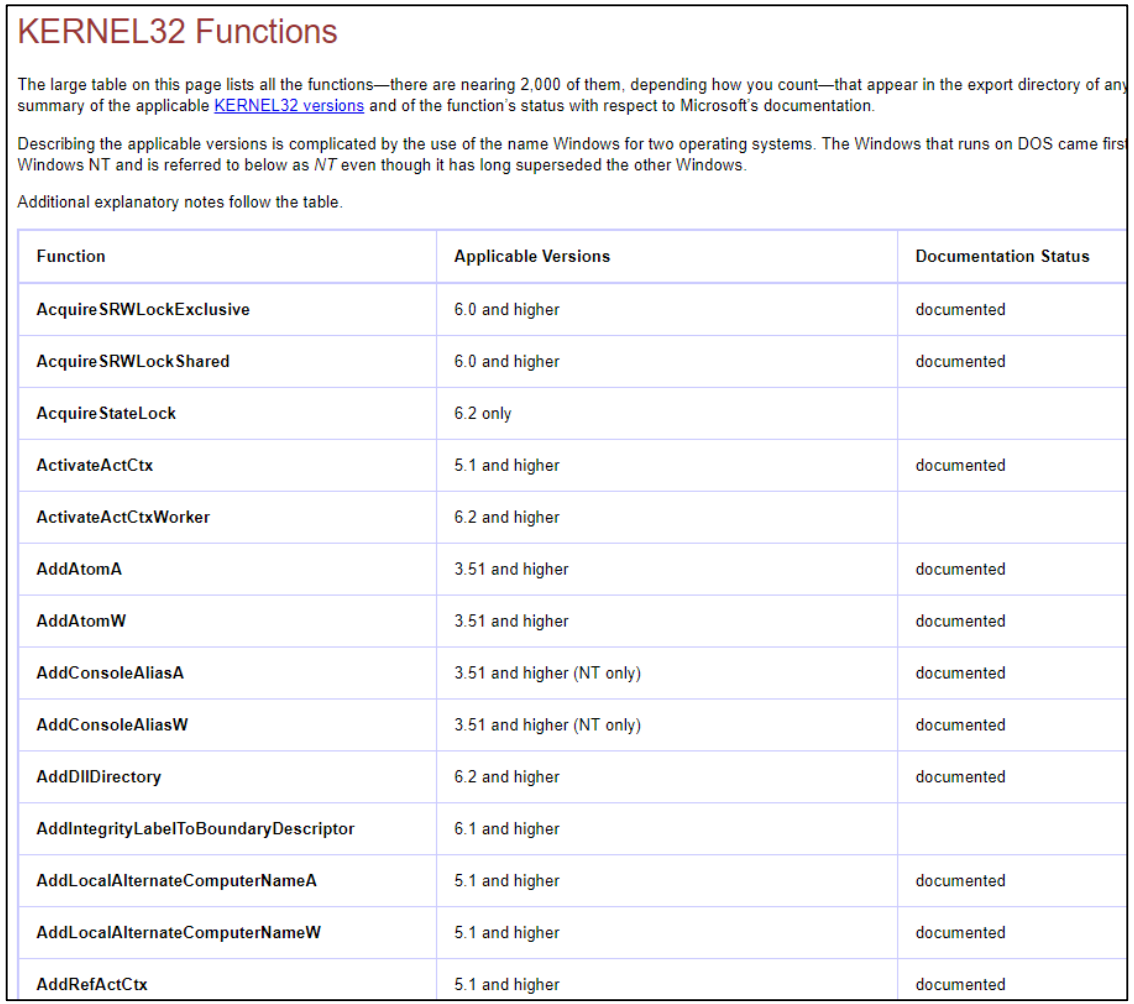
DLL (Dynamic Linking Library)

- Windows에서 구현된 동적 라이브러리.

- Microsoft 개발자들이 만들어 놓은 Windows API들의 실제 코드들이 들어가 있음.

- 우리도 만들 수 있다

kernel32.dll



KERNEL32 Functions

The large table on this page lists all the functions—there are nearing 2,000 of them, depending how you count—that appear in the export directory of any summary of the applicable KERNEL32 versions and of the function's status with respect to Microsoft's documentation.

Describing the applicable versions is complicated by the use of the name Windows for two operating systems. The Windows that runs on DOS came first Windows NT and is referred to below as *NT* even though it has long superseded the other Windows.

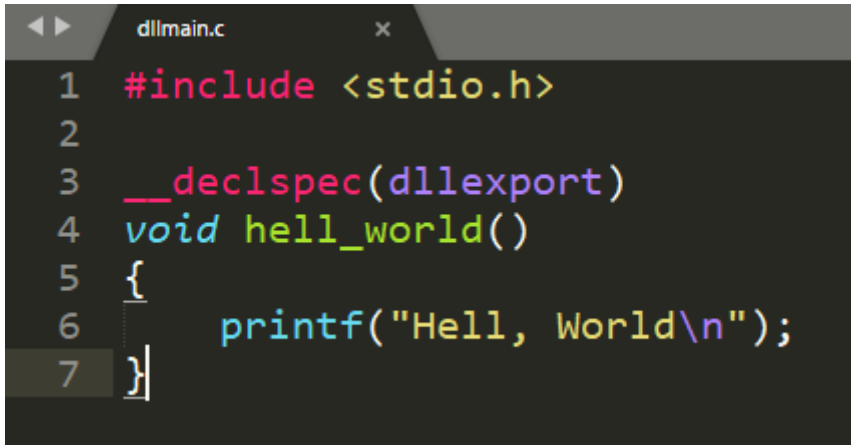
Additional explanatory notes follow the table.

Function	Applicable Versions	Documentation Status
AcquireSRWLockExclusive	6.0 and higher	documented
AcquireSRWLockShared	6.0 and higher	documented
AcquireStateLock	6.2 only	
ActivateActCtx	5.1 and higher	documented
ActivateActCtxWorker	6.2 and higher	
AddAtomA	3.51 and higher	documented
AddAtomW	3.51 and higher	documented
AddConsoleAliasA	3.51 and higher (NT only)	documented
AddConsoleAliasW	3.51 and higher (NT only)	documented
AddDllDirectory	6.2 and higher	documented
AddIntegrityLabelToBoundaryDescriptor	6.1 and higher	
AddLocalAlternateComputerNameA	5.1 and higher	documented
AddLocalAlternateComputerNameW	5.1 and higher	documented
AddRefActCtx	5.1 and higher	documented

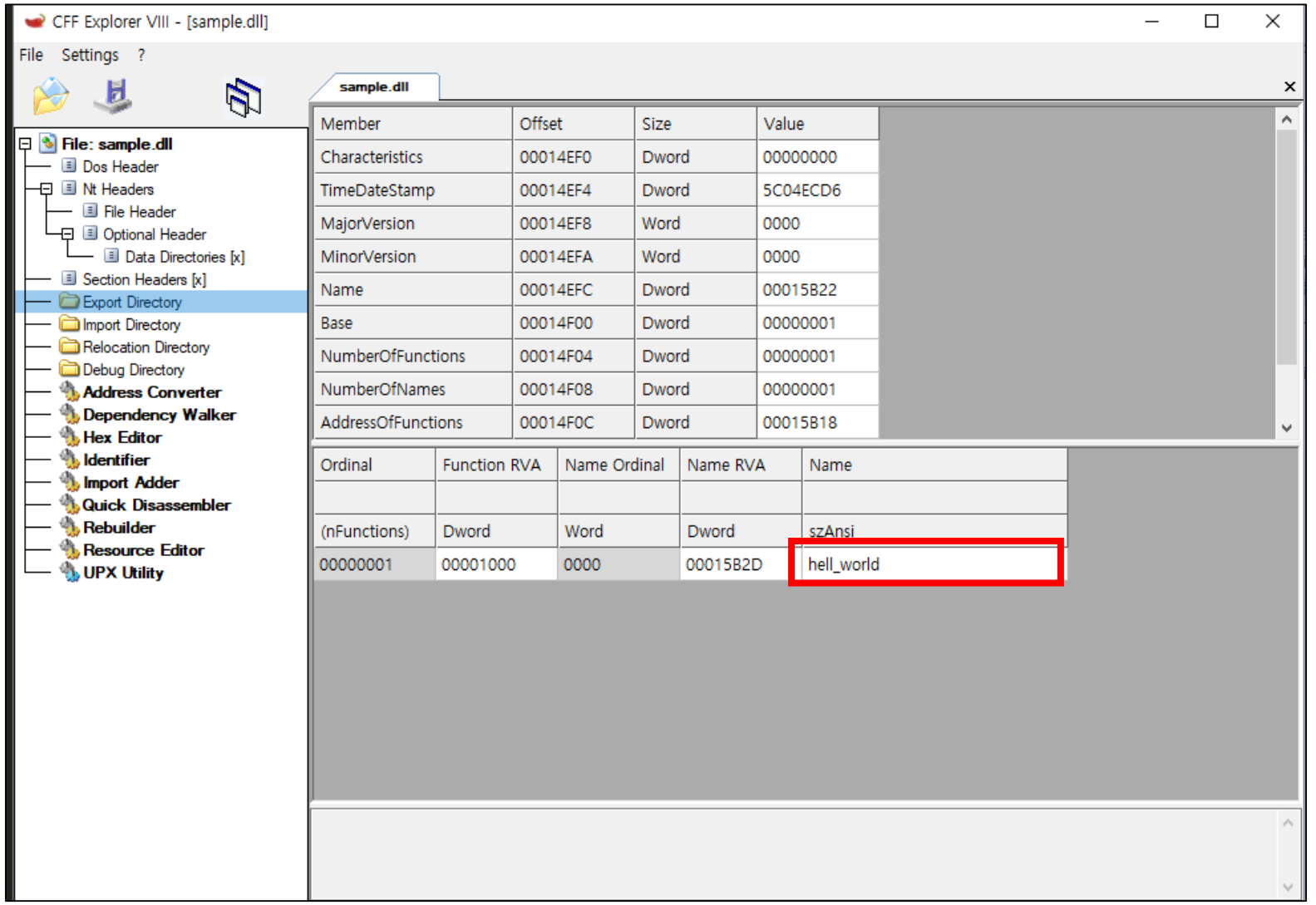
http://www.geoffchappell.com/studies/windows/win32/kernel32/api/

Sample DLL

- https://drive.google.com/file/d/1avmecfAFq0Tm1HrRNB19CKQNr52W7jjR/view?usp=sharing



Sample DLL


CFF Explorer VIII - [sample.dll]
File Settings ?
sample.dll
File: sample.dll
Dos Header
Nt Headers
File Header
Optional Header
Data Directories [x]
Section Headers [x]
Export Directory
Import Directory
Relocation Directory
Debug Directory
Address Converter
Dependency Walker
Hex Editor
Identifier
Import Adder
Quick Disassembler
Rebuilder
Resource Editor
UPX Utility
Member | Offset | Size | Value
Characteristics | 00014EF0 | Dword | 00000000
TimeDateStamp | 00014EF4 | Dword | 5C04ECD6
MajorVersion | 00014EF8 | Word | 0000
MinorVersion | 00014EFA | Word | 0000
Name | 00014EFC | Dword | 00015B22
Base | 00014F00 | Dword | 00000001
NumberOfFunctions | 00014F04 | Dword | 00000001
NumberOfNames | 00014F08 | Dword | 00000001
AddressOfFunctions | 00014F0C | Dword | 00015B18
Ordinal | Function RVA | Name Ordinal | Name RVA | Name
(nFunctions) | Dword | Word | Dword | szAnsi
00000001 | 00001000 | 0000 | 00015B2D | hell_world

DLL 함수 사용하기

- 묵시적 링킹(implicit linking)
  - 함수가 어느 DLL에 있는지 밝히지 않고 사용.

- 명시적 링킹 (explicit linking)
  - DLL 파일을 나중에 불러서 원하는 함수를 동적으로 호출

implicit Linking

- dll과 lib 파일이 필요함.

- dll을 만들때 lib 파일도 같이 만들어지긴 함!

```
#include <stdio.h>
#pragma comment(lib, "sample.lib")
__declspec(dllimport) void hell_world(void);

int main()
{
    hell_world();
}
```

```
D:₩1학교생활좀해볼까₩whois₩2018 하반기₩스터디₩리버싱₩making_dll>a
Hell, World

D:₩1학교생활좀해볼까₩whois₩2018 하반기₩스터디₩리버싱₩making_dll>
```

explicit Linking

- 직접 DLL 을 불러서 사용함.

- LoadLibrary, GetProcAddress 함수 사용

```
#include <stdio.h>
#include <windows.h>
int main()
{
    HMODULE hDll = LoadLibraryA("sample.dll");
    FARPROC hell_world;
    hell_world = GetProcAddress(hDll, "hell_world");
    hell_world();
}
```

```
D:₩1학교생활좀해볼까₩whois₩2018 하반기₩스터디₩리버싱₩making_dll>b
Hell, World

D:₩1학교생활좀해볼까₩whois₩2018 하반기₩스터디₩리버싱₩making_dll>
```

Export Address Table

- IMAGE_EXPORT_DIRECTORY

```
typedef struct _IMAGE_EXPORT_DIRECTORY {
    DWORD   Characteristics;
    DWORD   TimeDateStamp;          // creation time date stamp
    WORD    MajorVersion;
    WORD    MinorVersion;
    DWORD   Name;                   // address of library file name
    DWORD   Base;                   // ordinal base
    DWORD   NumberOfFunctions;      // number of functions
    DWORD   NumberOfNames;          // number of names
    DWORD   AddressOfFunctions;     // address of function start address
array
    DWORD   AddressOfNames;         // address of functino name string array
    DWORD   AddressOfNameOrdinals;  // address of ordinal array
} IMAGE_EXPORT_DIRECTORY, *PIMAGE_EXPORT_DIRECTORY;
```

IMAGE_IMPORT_DIRECTORY와 구조가 비슷함.

Name : 이 라이브러리 이름

NumberOfFunctions : export할 함수 개수

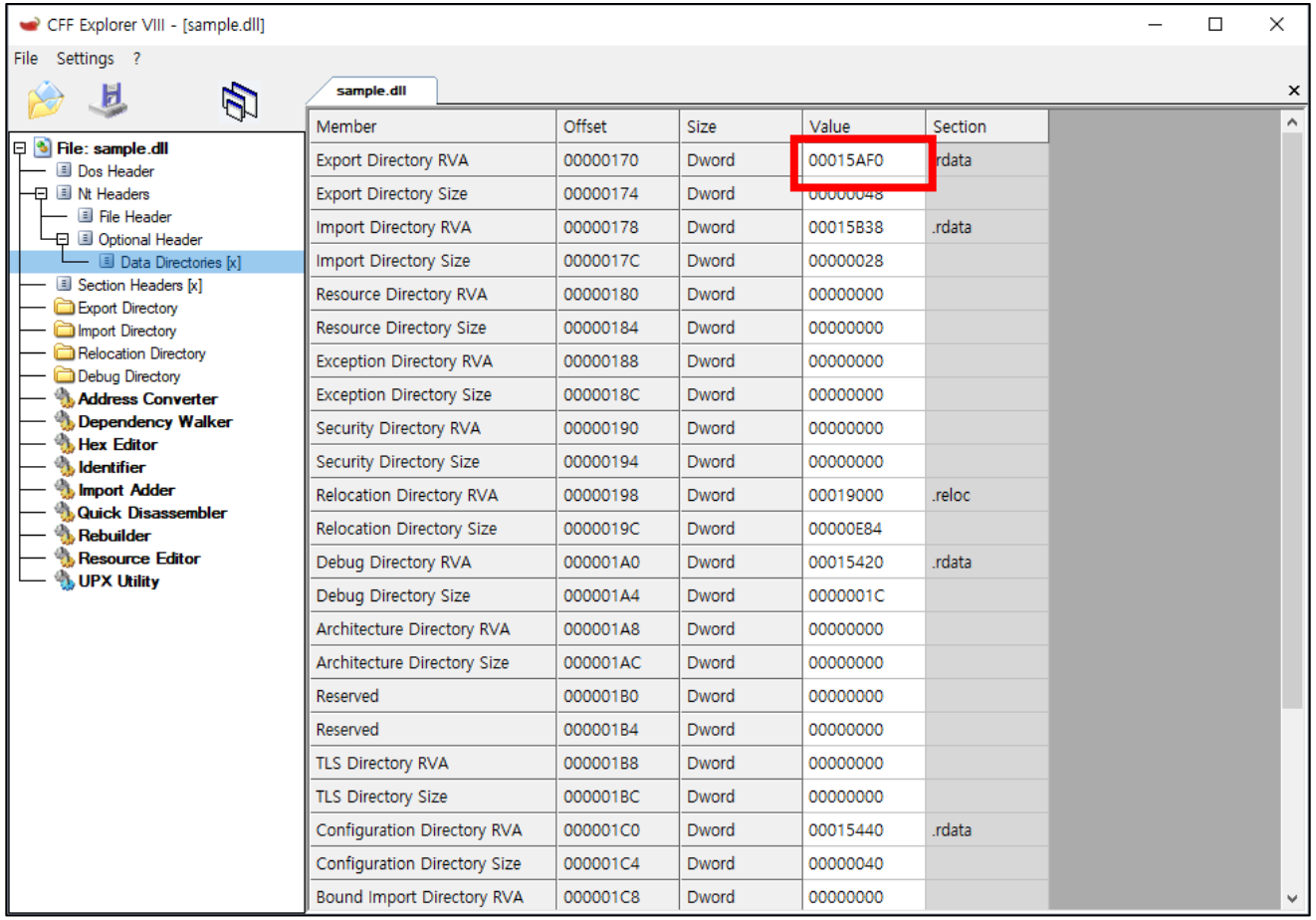
NumberOfNames : export할 함수 중에서 이름 개수

AddressOfFunctions : export 함수들 배열의 주소

AddressOfNames : 함수 이름 배열의 주소

AddressOfOrdinals : ordinal 배열의 주소

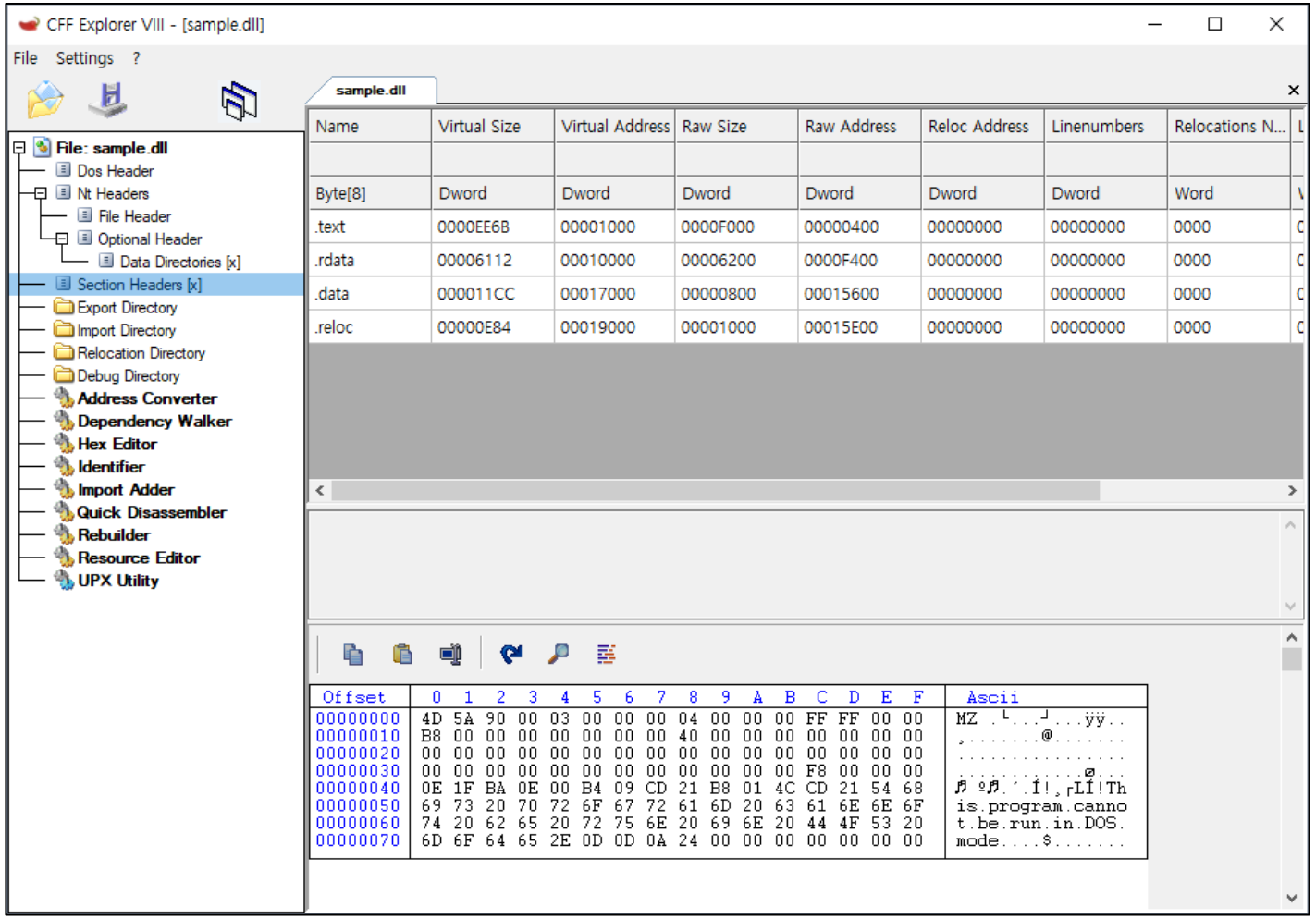
Export Address Table



Member	Offset	Size	Value	Section
Export Directory RVA	00000170	Dword	00015AF0	rdata
Export Directory Size	00000174	Dword	00000048	
Import Directory RVA	00000178	Dword	00015B38	.rdata
Import Directory Size	0000017C	Dword	00000028	
Resource Directory RVA	00000180	Dword	00000000	
Resource Directory Size	00000184	Dword	00000000	
Exception Directory RVA	00000188	Dword	00000000	
Exception Directory Size	0000018C	Dword	00000000	
Security Directory RVA	00000190	Dword	00000000	
Security Directory Size	00000194	Dword	00000000	
Relocation Directory RVA	00000198	Dword	00019000	.reloc
Relocation Directory Size	0000019C	Dword	00000E84	
Debug Directory RVA	000001A0	Dword	00015420	.rdata
Debug Directory Size	000001A4	Dword	0000001C	
Architecture Directory RVA	000001A8	Dword	00000000	
Architecture Directory Size	000001AC	Dword	00000000	
Reserved	000001B0	Dword	00000000	
Reserved	000001B4	Dword	00000000	
TLS Directory RVA	000001B8	Dword	00000000	
TLS Directory Size	000001BC	Dword	00000000	
Configuration Directory RVA	000001C0	Dword	00015440	.rdata
Configuration Directory Size	000001C4	Dword	00000040	
Bound Import Directory RVA	000001C8	Dword	00000000	

Export Directory RVA : 0x15AF0

Export Address Table

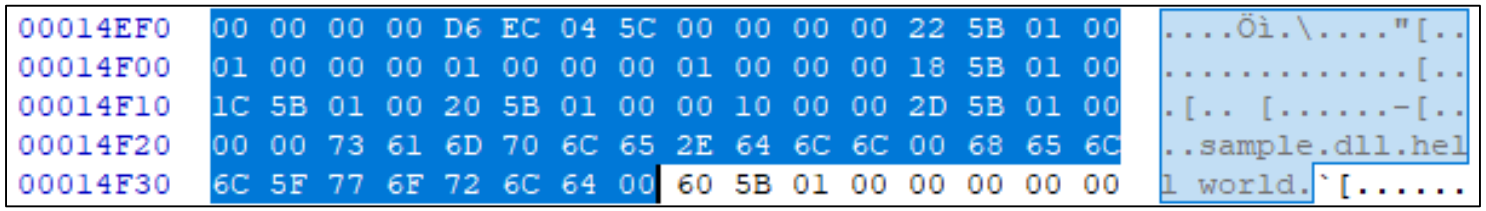


.rdata VA : 0x10000
.rdata Raw : 0xF400

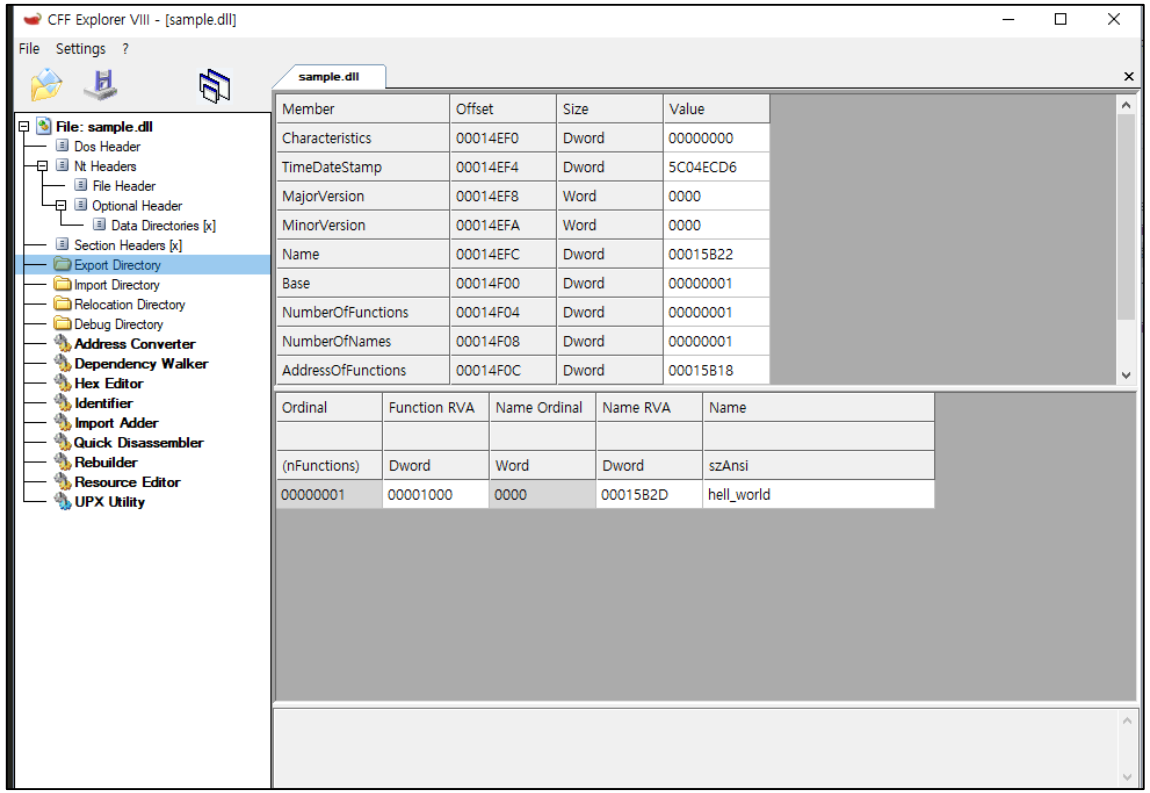
Export Address Table

- Export Directory RVA : 0x15AF0
- .rdata VA : 0x10000
- .rdata RAW : 0xF400

즉, 0x15AF0 – 0x10000 + 0xF400 => Export Directory File Offset



Export Address Table



Name RVA : 0x15822 (FileOffset : 0x14F22)

Ordinal Base : 0x1

NumberOfFunctions : 0x1

NumberOfNames : 0x1

AddressOfFunctions 0x15818 (FileOffset : 0x14F18)

```
00014F20  00 00 73 61 6D 70 6C 65 2E 64 6C 6C 00 68 65 6C  ..sample.dll.hel
00014F30  6C 5F 77 6F 72 6C 64 00 60 5B 01 00 00 00 00 00  l world.`[......
```

Export Address Table

Ordinal	Function RVA	Name Ordinal	Name RVA	Name
(nFunctions)	Dword	Word	Dword	szAnsi
00000001	00001000	0000	00015B2D	hell_world

```
00014F10  1C 5B 01 00 20 5B 01 00 00 10 00 00 2D 5B 01 00  .[.. [.....-[..
00014F20  00 00 73 61 6D 70 6C 65 2E 64 6C 6C 00 68 65 6C  ..sample.dll.hel
00014F30  6C 5F 77 6F 72 6C 64 00 60 5B 01 00 00 00 00 00  l world.`[......
```

Function RVA : 0x1000 => (RawOffset : 0x400)
hell_world 함수의 코드!
Name RVA : 0x015B2D (RawOffset : 0x14F2D)
hell_world 함수의 이름을 볼 수 있음.

```
00014F10  1C 5B 01 00 20 5B 01 00 00 10 00 00 2D 5B 01 00  .[.. [.....-[..
00014F20  00 00 73 61 6D 70 6C 65 2E 64 6C 6C 00 68 65 6C  ..sample.dll.hel
00014F30  6C 5F 77 6F 72 6C 64 00 60 5B 01 00 00 00 00 00  l world.`[......
```

과제

- https://drive.google.com/file/d/0B2mPMLw5nQb4WVZHaWVvM0NnTDQ/view?usp=sharing
- https://drive.google.com/file/d/1vJb0Z2vCPt1tcBKpBbUfbdzTXbAUBKG2/view?usp=sharing

- 이거 둘 복구하기
- 복구가 정상적으로 되면 Hell, World라는 문자열이 콘솔에 출력되어야함!