

PE File Format

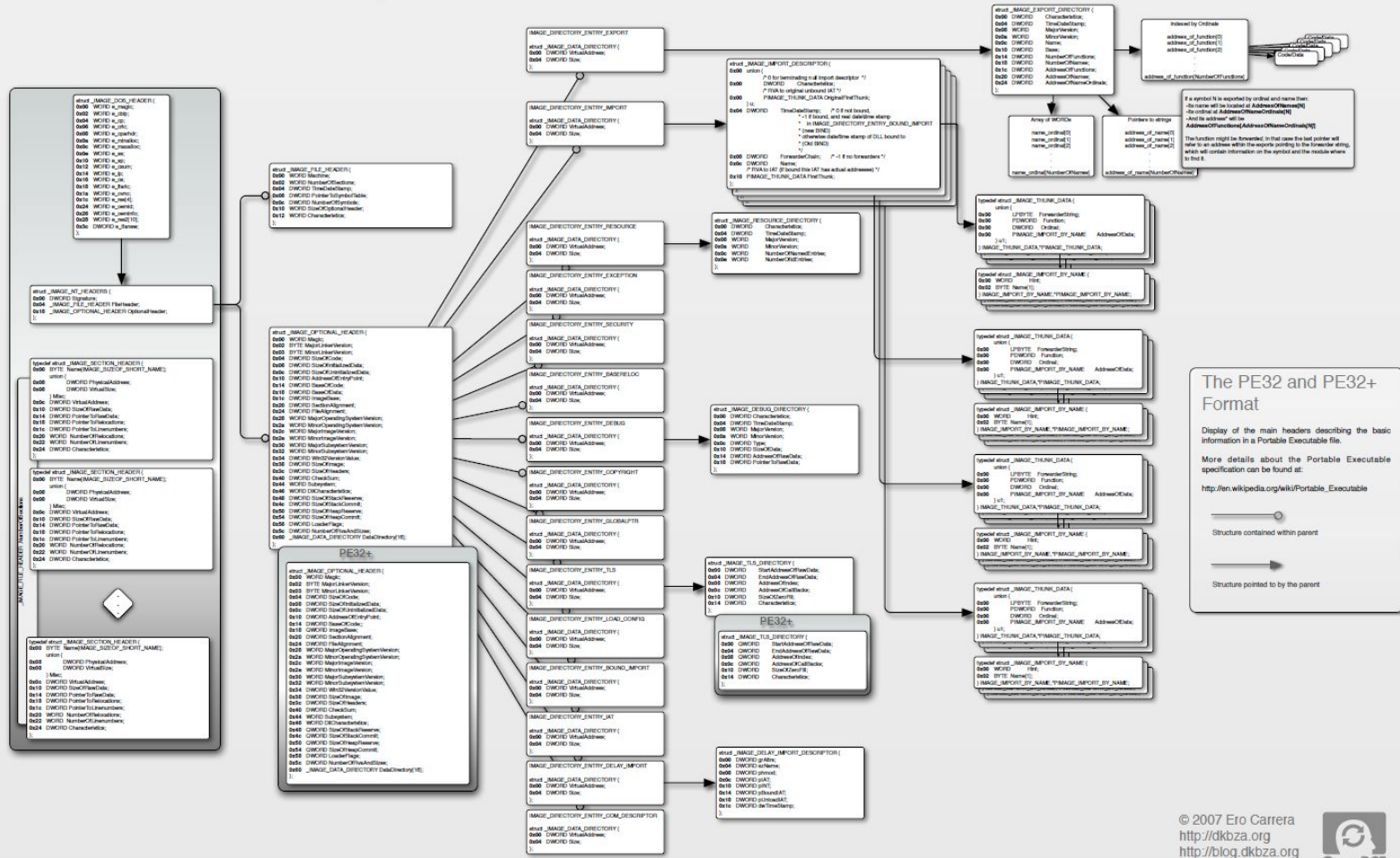
Azlan Mukhtar
CYSECA Solutions Sdn Bhd

Introduction to PE/COFF

- PE stands for Portable Executable
- Microsoft introduced PE in Windows NT 3.1
- It originates from Unix COFF
- Features dynamic linking, symbol exporting/importing
- Can contain Intel, Alpha, MIPS and even .NET MSIL binary code
- 64-bit version is called PE32+

PE Format layout

Portable Executable Format Layout



MZ Header



```
struct _IMAGE_DOS_HEADER {  
0x00 WORD e_magic;  
0x02 WORD e_cblp;  
0x04 WORD e_cp;  
0x06 WORD e_crlc;  
0x08 WORD e_cparhdr;  
0x0a WORD e_minalloc;  
0x0c WORD e_maxalloc;  
0x0e WORD e_ss;  
0x10 WORD e_sp;  
0x12 WORD e_csum;  
0x14 WORD e_ip;  
0x16 WORD e_cs;  
0x18 WORD e_lfarlc;  
0x1a WORD e_ovno;  
0x1c WORD e_res[4];  
0x24 WORD e_oemid;  
0x26 WORD e_oeminfo;  
0x28 WORD e_res2[10];  
0x3c DWORD e_lfanew;  
};
```

PE header – File header

MZ Header

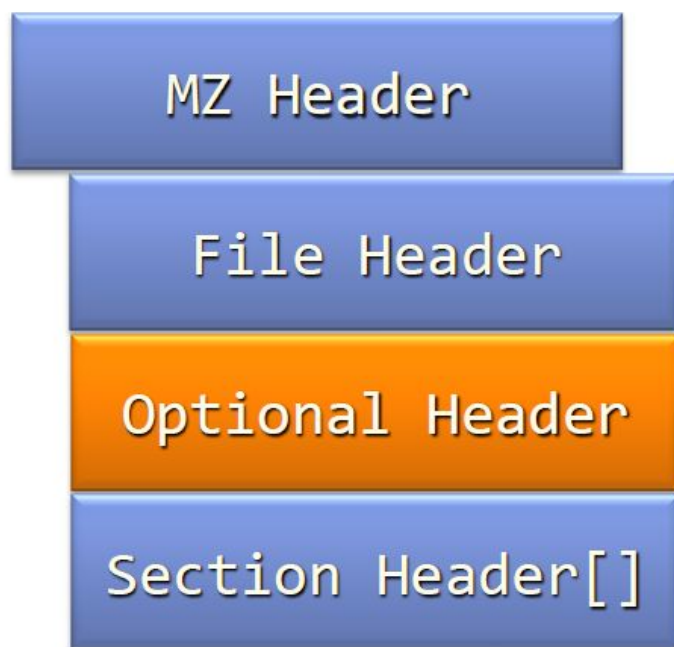
File Header

Optional Header

Section Header[]

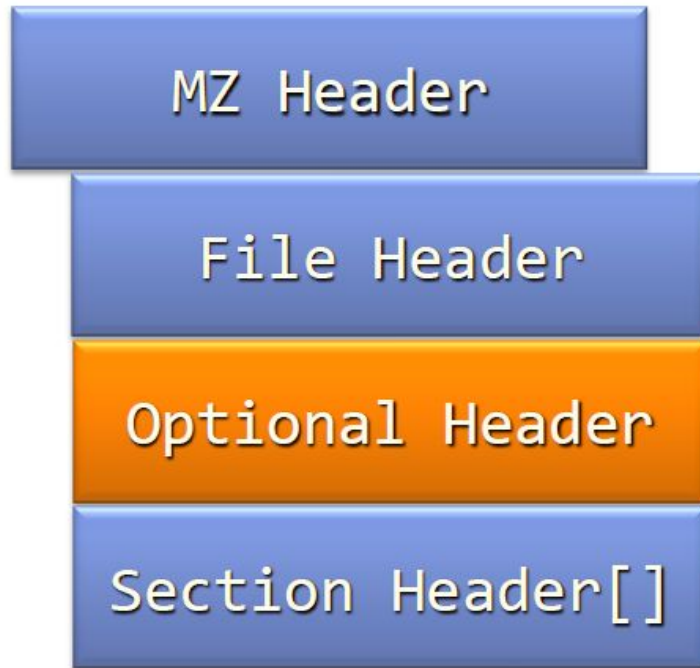
```
struct _IMAGE_FILE_HEADER {  
    0x00 WORD Machine;  
    0x02 WORD NumberOfSections;  
    0x04 DWORD TimeDateStamp;  
    0x08 DWORD PointerToSymbolTable;  
    0x0c DWORD NumberOfSymbols;  
    0x10 WORD SizeOfOptionalHeader;  
    0x12 WORD Characteristics;  
};
```

PE Header – optional header



```
struct _IMAGE_OPTIONAL_HEADER {
0x00 WORD Magic;
0x02 BYTE MajorLinkerVersion;
0x03 BYTE MinorLinkerVersion;
0x04 DWORD SizeOfCode;
0x08 DWORD SizeOfInitializedData;
0x0c DWORD SizeOfUninitializedData;
0x10 DWORD AddressOfEntryPoint;
0x14 DWORD BaseOfCode;
0x18 DWORD BaseOfData;
0x1c DWORD ImageBase;
0x20 DWORD SectionAlignment;
0x24 DWORD FileAlignment;
0x28 WORD MajorOperatingSystemVersion;
0x2a WORD MinorOperatingSystemVersion;
0x2c WORD MajorImageVersion;
0x2e WORD MinorImageVersion;
0x30 WORD MajorSubsystemVersion;
0x32 WORD MinorSubsystemVersion;
0x34 DWORD Win32VersionValue;
0x38 DWORD SizeOfImage;
0x3c DWORD SizeOfHeaders;
0x40 DWORD CheckSum;
0x44 WORD Subsystem;
0x46 WORD DllCharacteristics;
0x48 DWORD SizeOfStackReserve;
0x4c DWORD SizeOfStackCommit;
0x50 DWORD SizeOfHeapReserve;
0x54 DWORD SizeOfHeapCommit;
0x58 DWORD LoaderFlags;
0x5c DWORD NumberOfRvaAndSizes;
0x60 _IMAGE_DATA_DIRECTORY DataDirectory[16];
};
```


PE Header – optional header – data directory



| |
|--------------------------------------|
| 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_COPYRIGHT |
| 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 |

```
struct _IMAGE_DATA_DIRECTORY {  
    0x00  DWORD VirtualAddress;  
    0x04  DWORD Size;  
};
```

PE Header – section header

MZ Header

File Header

Optional Header

Section Header[]

```
typedef struct _IMAGE_SECTION_HEADER {  
0x00 BYTE Name[IMAGE_SIZEOF_SHORT_NAME];  
    union {  
0x08     DWORD PhysicalAddress;  
0x08     DWORD VirtualSize;  
    } Misc;  
0x0c     DWORD VirtualAddress;  
0x10     DWORD SizeOfRawData;  
0x14     DWORD PointerToRawData;  
0x18     DWORD PointerToRelocations;  
0x1c     DWORD PointerToLinenumbers;  
0x20     WORD  NumberOfRelocations;  
0x22     WORD  NumberOfLinenumbers;  
0x24     DWORD Characteristics;  
};
```


PE Loading

File on disk



Image in memory



RVA = Relative Virtual Address = Offset
from image base in memory

Importing Symbols

- Symbols (functions/data) can be imported from external DLLs
- The loader will load external DLLs automatically
- All the dependencies are loaded as well
- DLLs will be loaded only once
- External addresses are written to the Import Address Table (IAT)

Importing Symbols - continued

- Every PE has one IMAGE_IMPORT_DESCRIPTOR
- The descriptor points to two parallel lists of symbols to import
 - Import Address Table (IAT)
 - Import Name Table (INT)
- The primary list (IAT) is overwritten by the loader, the second one is not
- Executables can be pre-bound to DLLs to speed up loading
- Symbols can be imported by ASCII name or ordinal (usually by Name)

Import Descriptors

IMAGE_DIRECTORY_ENTRY_IMPORT

```
struct _IMAGE_DATA_DIRECTORY {  
  0x00 DWORD VirtualAddress;  
  0x04 DWORD Size;  
};
```

```
struct _IMAGE_IMPORT_DESCRIPTOR {  
  0x00 union {  
    /* 0 for terminating null import descriptor */  
    0x00 DWORD Characteristics;  
    /* RVA to original unbound IAT */  
    0x00 PIMAGE_THUNK_DATA OriginalFirstThunk;  
  } u;  
  0x04 DWORD TimeDateStamp; /* 0 if not bound,  
  0x08 DWORD ForwarderChain; /* -1 if no forwarders */  
  0x0c DWORD Name;  
  /* RVA to IAT (if bound this IAT has actual addresses) */  
  0x10 PIMAGE_THUNK_DATA FirstThunk;  
};
```

```
typedef struct _IMAGE_THUNK_DATA {  
  union {  
    0x00 LPBYTE ForwarderString;  
    0x00 PDWORD Function;  
    0x00 DWORD Ordinal;  
    0x00 PIMAGE_IMPORT_BY_NAME AddressOfData;  
  } u1;  
} IMAGE_THUNK_DATA, *PIMAGE_THUNK_DATA;
```

```
typedef struct _IMAGE_IMPORT_BY_NAME {  
  0x00 WORD Hint;  
  0x02 BYTE Name[1];  
} IMAGE_IMPORT_BY_NAME, *PIMAGE_IMPORT_BY_NAME;
```

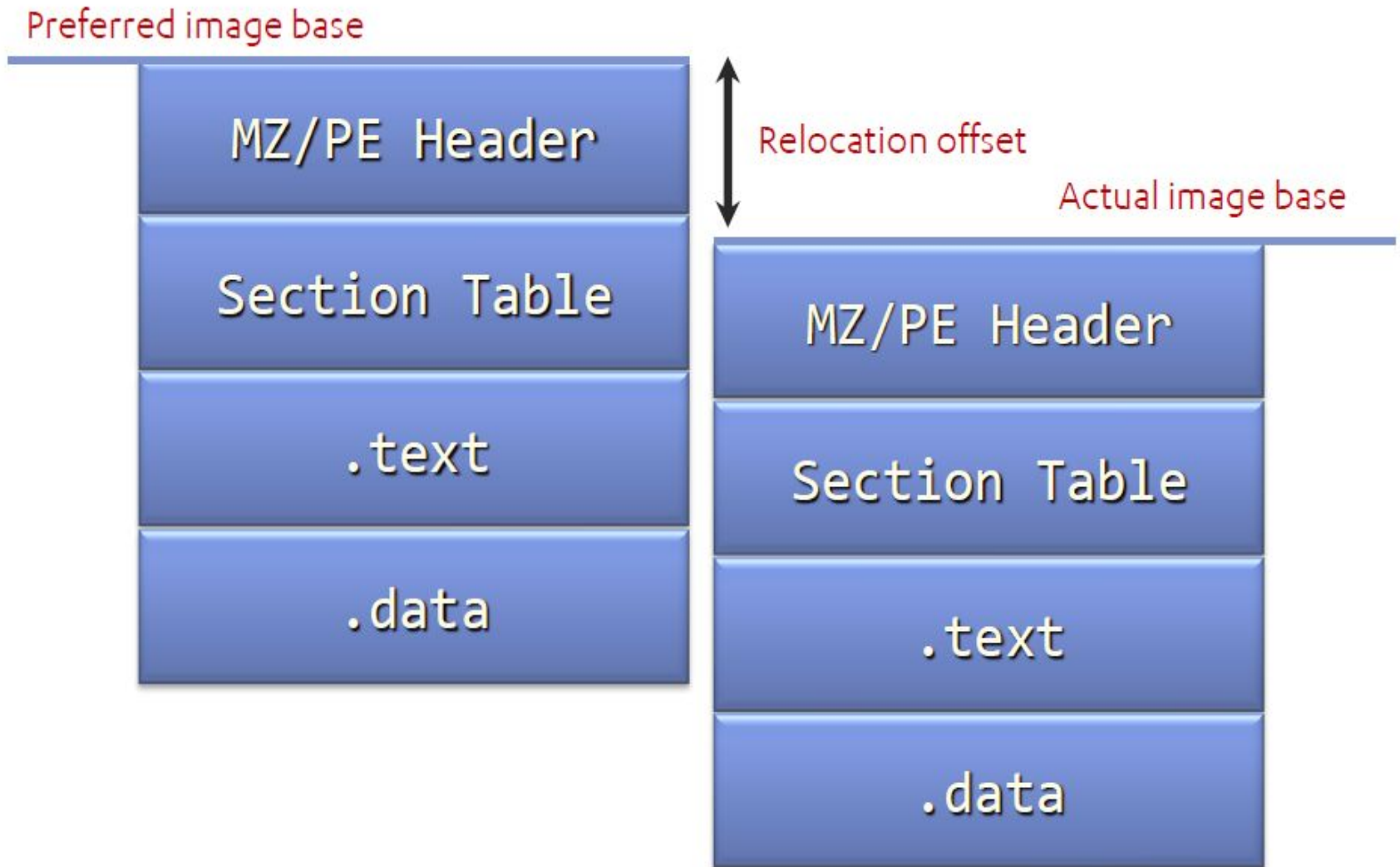
Exports

- Symbols can be exported with ordinals, names or both
- Ordinals are simple index numbers of symbols
- Name is a full ASCII name of the exported symbol
- Exports can be forwarded to another DLL
- Forwarded symbol's address points to a name in the exports section

Resource

- Resources in PE are similar to an archive
- Resource files can be organized into directory trees
- The data structure is quite complex but there are tools to handle it
- Most common resources:
 - Icons
 - Version information
 - GUI resources

Base Relocation



Demo

How to view and edit PE file with CFF Explorer

References

- Microsoft PE and COFF Specification
 - <https://msdn.microsoft.com/en-us/windows/hardware/gg463119.aspx>
- PE Information
 - <https://www.accessroot.com/yates/documents/43.html>
- PE File Format Graphs
 - <http://blog.dkbza.org/2012/08/pe-file-format-graphs.html>
 - <https://github.com/corkami/pics/tree/master/binary/pe101>
 - <https://github.com/corkami/pics/tree/master/binary/pe102>
- Understanding RVAs and Import Tables
 - http://www.sunshine2k.de/reversing/tuts/tut_rvait.htm