Portable Executable (PE) Files Explained

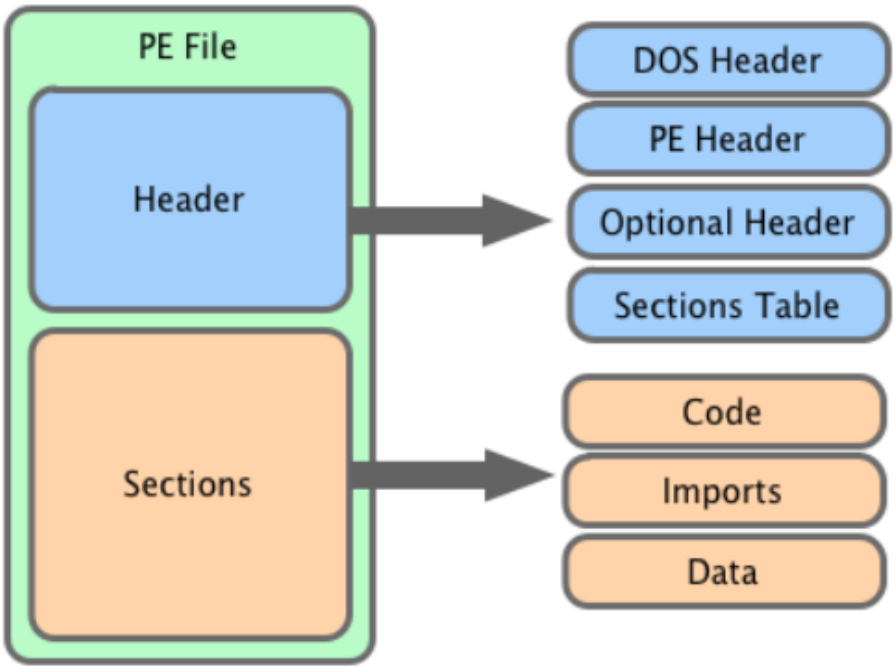
Background/Introduction

The Track 2 Data Science Challenge revolves around the construction of a classifier that can identify a given portable executable (PE) file as benign or malware. PE files are structured very differently than the ordinary csv, text, or xlsx files that data scientists more frequently encounter and train their models on. The information below provides an overview on the structure of PE files and their components.

PE File Structure

Whether you know it or not, it's a near certainty that you've encountered PE files before. If you've used Microsoft Windows, you've used PE files. It's a file format developed by Microsoft™ that is used on a large number of file types. The most common ones that you're likely to recognize are .exe and .dll files (executables and dynamic-linked libraries respectively). PE files are made of two parts—The header and sections. The header contains details about the PE file itself, while the sections contain the contents of the executable.

The header is broken up into a number of different parts—the DOS header, the PE header, optional header, and the sections table. There can be any number of sections within a PE file, and they can be named anything the author would like. However, there must almost always be sections containing the actual code of the file, the import table, and the data used by the code. It ends up looking something like this:[1]



[2]

[1] Grunzweig, Josh. "Basic Packers: Easy as Pie". April 24, 2013. Trustwave. https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/basic-packers-easy-as-pie/.

[2] Grunzweig, Josh. "Basic Packers: Easy as Pie".

PE Headers

AS previously discussed, the PE File header is composed of four primary subsections with descriptions provided below.

DOS Header

The purpose of the DOS header is fairly simple. The DOS header starts with the first 64 bytes of every PE file and identifies the file as a valid executable compatible on a Windows device.[3]

PE Header

Like other executable files, a PE file has a collection of fields that define what the rest of file looks like. The header contains info such as the location and size of code. The first few hundred bytes of the typical PE file are taken up by the MS-DOS stub. There are some basic sub-sections defined in the header section itself; they are listed below:[4]

- Signature
  -Contains the signature so that it can be easily understandable by windows loader. The letters P.E. followed by two 0's tell everything.
- Machines
  - This is a number that identifies the type of machine on the target system, such as Intel, AMD, etc.
- Number of Sections
  - This defines the size of the section table, which immediately follow the header.
- Size of Optional Header
  - This lies between top of the optional header and the start of the section table. This is the size of the optional header that is required for an executable file. This value should be zero for an object file.

Optional Header

This header can contain a wealth of information, such as the size of code on the disk, the checksum of the file, the size of the stack and heap, etc. It also contains the address of the entry point, or where code execution will begin.

Data directories are contained within the optional header. Data directories contain tables that contain information about resources, imports, exports, debug information, and local thread storage to name a few. The import information is of particular interest.

Consider the following example as it relates to imports. We have five executables that all share some piece of code. Now, it doesn't really make a ton of sense for all of these executables to store this code by themselves. Instead, it makes a lot more sense to break out this code into a separate library and simply have each executable load this library at runtime. This is essentially what the import table is—A list of libraries and their associated functions that an executable wishes to load at runtime. This table of functions and libraries is replaced in a packed file, and is generated when executed. This means that if we wish to unpack a binary that has been packed, we'll have to reconstruct this information in order for the

[3] Revers3r. "Malware researcher's handbook (demystifying PE file)". INFOSEC. November 26, 2015. https://resources.infosecinstitute.com/topic/2-malware-researchers-handbook-demystifying-pe-file/.

[4] Revers3r. "Malware researcher's handbook (demystifying PE file

unpacked PE file to be valid and to work as expected. Once this header is parsed, execution flow moves onto the Sections Table.[5]

Sections Table

The Sections Table outlines all sections that are present within the PE file. This includes the name of the section, the location of the section, size of the section both in the file and in virtual memory, as well as any flags associated with that section. Sections make up the bulk of the PE file itself, so it is important to have this table of information on hand.[6]

PE Sections

As mentioned earlier, PE sections typically at the very least contain a section for code, a section for imports, and a section for data.

Code

This is normally the first section and contains the executable code for the application. Inside this section is also an entry point of the application: the address of the first application instruction that will be executed. An application can have more than one section with the executable code.[7]

Import Section

The import section contains the actual addresses for all functions needed by the PE file. These addresses are populated at runtime, as every Windows system may be different. As such, it's possible that a function may not be located at the same memory location between Windows versions. By populating the import table with these addresses at runtime, it allows us to use this PE file on multiple Microsoft Windows machines.[8]

Data

The data section represents uninitialized data for the application, including all variables declared as static within a function or source module. It also contains the read-only data, such as literal strings, constants, and debug directory information. All other variables (except automatic variables, which appear on the stack) are stored in the data section. Basically, these are application or module global variables.[9]

---

[5] Grunzweig, Josh. "Basic Packers: Easy as Pie".
[6] Grunzweig, Josh. "Basic Packers: Easy as Pie".
[7] Daulagupu, Satyajit. "Understanding PE Structure, The Layman's Way – Malware Analysis Part 2. Tech Zealots. May 10, 2018. https://tech-zealots.com/malware-analysis/pe-portable-executable-structure-malware-analysis-part-2/.
[8] Grunzweig, Josh. "Basic Packers: Easy as Pie".
[9] Plachy, Johannes. July 27, 2018. https://blog.kowalczyk.info/articles/pefileformat.html.