CS7038 - Malware Analysis - Wk04.1
Static Analysis Introduction

Coleman Kane
kaneca@mail.uc.edu

February 1, 2017

UNIVERSITY OF Cincinnati

Static Analysis

Static Analysis is the process of documenting your observations about what identifying characteristics a malware sample exhibits. The goal of this process is that, after analysis, you have extracted some identifying characteristics from a malware sample that can be used to help you search further for more samples of that malware (and, hopefully, others that are similar to it).

We distinguish **static analysis** to focus on how a sample "looks", for the purpose of identifying any samples of it that may be dormant and inactive within your attack surface. This is different from **dynamic analysis**, where we are trying to define the actions it takes or may take when executed on a system.

Files Have Structure

Most information formats in computing have "*structure*" to them:

PNG Files are described as "*datastreams*" that begin with a sequence of 8 bytes, followed by one or more "chunks", which themselves have substructure:

`https://www.w3.org/TR/PNG/#5DataRep`

EXE Files are the files interpreted by Windows to contain Program and Dynamically Linkable Library code. These contain two or three header blocks (of a defined byte length), plus a *section directory* (like a table of contents), followed by one or more *sections*, each of which has their own substructure. Each section in the PE32 file must contain a reference entry in the section directory.

`https://en.wikibooks.org/wiki/X86_Disassembly/Windows_Executable_Files#PE_Files`

MS-CFB Files, also known as OLE are a container format common to many Microsoft applications and systems. Many people associate these with the older MSOffice files, DOC, XLS, PPT, etc. These are even more complex structures, mimicking a filesystem within a file, complete with hierarchy and block-based storage allocation.

`https://winprotocoldoc.blob.core.windows.net/productionwindowsarchives/MS-CFB/[MS-CFB].pdf`

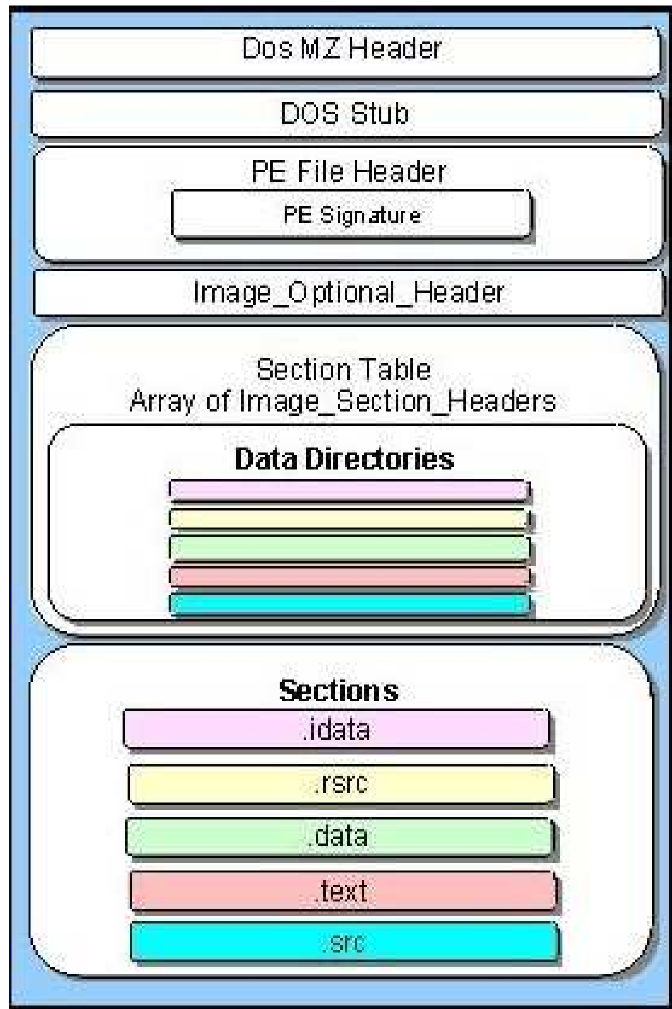
PNG File Visual

PNG Header	89 50 4E 47 0D 0A 1A 0A			
IHDR	length	IHDR	chunk data	CRC
extra tEXt chunk	length	tEXt	<html> <!--	CRC
extra tEXt chunk	length	tEXt	_ random chars ...	
	... random chars ...			
	--> <decoder HTML and script goes here ..>			
	<script type=text/undefined>/*...			CRC
IDAT chunk	length	IDAT	pixel data	CRC
IDAT chunk	length	IDAT	pixel data	CRC
IDAT chunk	length	IDAT	pixel data	CRC
IEND chunk	0	IEND	CRC	

Figure 1: From: http://stegosploit.info/

UNIVERSITY OF Cincinnati

PE32 File Visual



The PE32 file format,
which is the preferred container for Windows executable programs and dynamically linkable libraries, has a number of components inside.

From: http://resources.infosecinstitute.com/2-malware-researchers-handbook-demystifying-pe-file/

Even better diagram: http://images2015.cnblogs.com/blog/268182/201509/268182-20150906154155451-80554465.jpg

UNIVERSITY OF Cincinnati

Backus-Naur Form, C++ Form

Sometimes it is more convenient to describe the structure in terms of a *grammar*. This is particularly helpful if you intend to write a parser for the language. A common format used to describe language and structure is *Backus-Naur Form* (a.k.a. *BNF*).

An example could be:

```
< pe_file >         ::= mz_header dos_stub pe_header [ optional_header ]
                        < section_table > < sections >
< section_table > ::= section_descriptor [ < section_table > ]
< sections >        ::= pe_section [ < sections > ]
```

Another approach might be to define them as C data structures:

```
struct pe_file {
  char mz_header[0x40];
  char dos_stub[0xc0];
  char pe_header[0x18];
  char optional_header[0xe0];
  struct section_table s_tbl; /* You'd define this */
  struct section *sections;   /*    and this, elsewhere. */
};
```

UNIVERSITY OF Cincinnati

Unstructured Data

The previous slides described some well-defined file layouts, or *structures*. Generally speaking, a file is a container of a bunch of data. You may need to analyze that data as either *structured* or *unstructured* data.

Structured Data is content which has meaning or context associated with it based upon its organization. The information, therefore, is a combination of the data bytes, as well as the placement of that data within a particular structure. For instance, the value "`1000`" has a different meaning when in the field for `section size` versus `PE header offset`.

Unstructured Data is content for which you do not have any assigned meaning or context associated with its positioning. Upon initial review, much of the content within malware that you have yet to analyze fits this *unstructured* definition.

The task of **Reverse Engineering** includes attempting to derive what the meaning of *unstructured data* is within a malicious artifact.

Data Extraction

It is very common to extract data from artifacts, simultaneously using structured and unstructured approaches.

Structured data extraction attempts to pull content out of an artifact, and, using context, define its meaning and report both to the user. One of the most common, and broadly-applicable tools toward this end is the open-source Perl project, `exiftool`: `http://www.sno.phy.queensu.ca/~phil/exiftool/`
Example: `exiftool document.pdf`

Unstructured data extraction attempts to pull content out of an artifact, frequently matching a pattern, and report these findings to an analyst in order to catalog them and possibly derive meaning from them. An exceedingly common utility for this is distributed with nearly all Linux systems: `strings`
Example (show all strings length>=6): `strings -n6 file.exe`
Similar: `grep -a -o -P '[\x20-\x7f]{6,}' file.exe`