CMSC 449
Malware Analysis

Lecture 2

Basic Static Analysis

Edited for CMSC 691

Types of Malware Analysis

Basic Static Analysis

- Examining the malware while it is “at rest”

- Plain-text strings within the code
- Functions imported
- File metadata
- File similarity metrics (to identify related malware)

- Goal is to find unusual features that guide next analysis steps

Basic Dynamic Analysis

- Observing the output and/or changes when the malware is run
  - But not interfering or interacting with the malware
- Changes to file system
- Created processes / threads
- Network traffic
- Changes to the registry / system configuration
- Can use a sandbox or run malware in a VM
  - The variety of options may surprise you!

Advanced Static Analysis

- Examining the malware’s code in detail

- Disassemblers convert machine code to assembly
  - Organize the code into subroutines, and allow the analyst to more easily trace their way through the code
  - Much, much easier than reading raw assembly code

- Can also decompile machine code into an approximation of C

Advanced Dynamic Analysis

- Using a debugger to control any and all aspects of the malware as it is being executed
    - Registers, stack, memory, and code

- Can “trick” malware to execute in ways it normally wouldn’t
    - May be necessary if it hides behaviors during a sandbox run

	Static	Dynamic
Basic	Looking at details of the malware when it is “at rest”	Running the malware and observing changes/output
Advanced	Closely examining the malware’s code in detail	Running the malware and using a debugger to control details of its execution

Objectives of Malware Analysis

Detection, Classification, and Attribution

- Detection: Is a file benign or malicious?
- Classification: What family of malware is this?
- Attribution: Which person/group used this malware?

Other Analysis Objectives

- Determining what malicious behaviors it performed
- Deeply understanding a function(s) in the file
- Identifying related malware samples
- Creating a signature for the malware

Malware Triage

- Hundreds of thousands of unique, previously unseen malicious files are created every day

- Many of these are minor alterations of existing malware
    - Malware authors continually update their malware to add new capabilities and evade detection

- Not enough time for human analysts to look at everything!
    - **Triage:** Give most attention to new/unusual/important samples!

Malware Triage

- Large malware analysis shops probably perform different levels of analysis depending upon priority

- All samples receive automated basic static analysis

- Many samples receive sandbox runs

- A handful of samples are flagged for manual analysis

Levels of Analysis

- Analysis time by a human can also vary
    - Again, depends on objectives and importance of file

- Sometimes, just need to take a quick look

- But may also spend days (or longer!) figuring out exactly what a file does

Basic Static Analysis

Static Analysis

- Learning properties of a file without running it

- For now, just doing basic static analysis
  - Analyzing file properties / metadata

- Advanced static analysis involves disassembling / decompiling an executable file to inspect code

Strings

- Sequences of printable characters in a file
- Running strings on a file is usually first step of analysis
- Gives hints about functionality of program
- Example: `strings -n 8 [file path] | less`
  - Gets all strings of length >= 8 from a file and pipes output to more

FLOSS

- Like strings but more powerful

- Extracts:
  - ASCII strings
  - UTF-16 strings
  - Stack strings
  - Some encoded strings

- `floss -n 8 --no-decoded-strings [file path] | less`

Strings and FLOSS Demo

Lab01-01.exe (strings)

Lab09-02.exe (floss)

File Formats

Common Malware File Formats

- Malware comes in many shapes and sizes
  - Windows executables (PE files)
  - Linux executables (ELF files)
  - Mac executables (Mach-O files)
  - Android and IOS apps (APK and IPA files)
  - Documents (DOCX, PPTX, XLSX, PDF, etc.)
  - And more (Java JARs, Javascript, Python, Powershell, Bash, etc.)

- This class will focus on analyzing Windows PE files

Types of PE Files – EXEs

- EXE files are the most common type of Windows PE file
- They have an “entry point” where code begins executing from
  - This is usually boilerplate code that parses command-line arguments and then calls main()
- When executed, the OS creates a process for the EXE file with its own address space

Types of PE Files – DLLs

- Dynamically Linked Library

- Cannot be run directly. Must be loaded into the address space of an existing process

- Allows other PE files to import functions from inside the DLL
  - These functions are called “DLL exports”

Types of PE Files - Others

- SYS files – System device drivers
- MUI files – Multilingual user interface files
- SCR files – Screensaver files
- BAT files – Scripts meant to look like commands?

- These are much less common than EXE and DLL files, but malware does use these file types

- There are other kinds of PE files, but they are more rare

Types of PE files – EXE files

- EXE files are typical Windows executable files.

- Has an “entry point” where code begins executing from.
  - This is usually boilerplate code that parses command-line arguments and then calls main()

The Portable Executable (PE) File Format

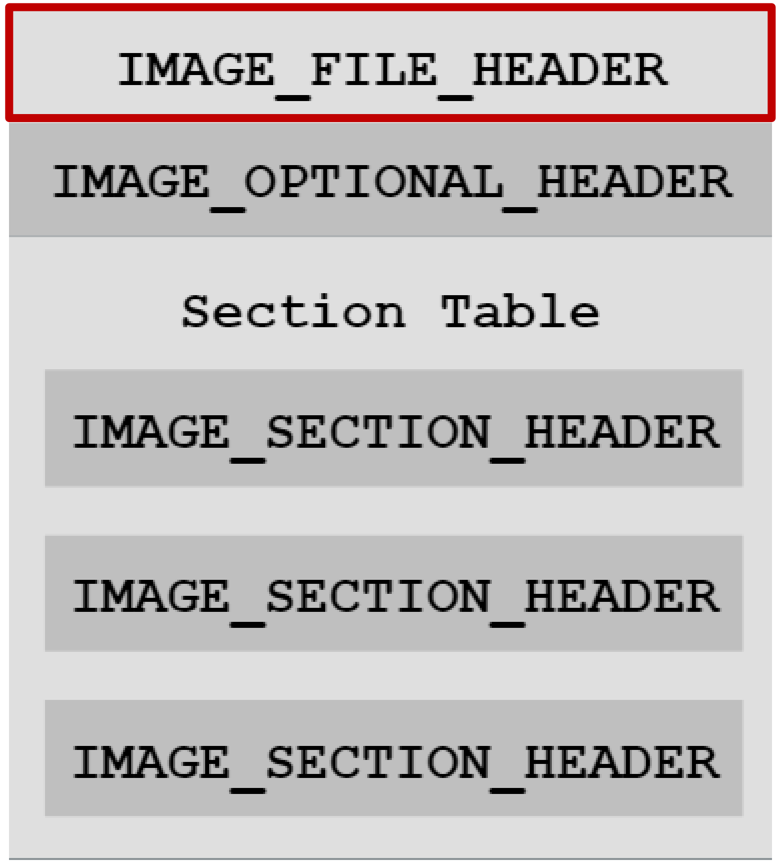
- File format for all Windows executables

- Describes how an executable file is loaded into memory and becomes a process

- Contains lots of metadata that is useful to malware analysts
  - Compilation timestamp
  - File version information
  - Offsets and sizes of various file sections

The IMAGE_FILE_HEADER

- Structure in the PE file format that contains basic file information
  - NumberOfSections
  - TimeDateStamp
  - Characteristics



IMAGE_FILE_HEADER

IMAGE_OPTIONAL_HEADER

Section Table

IMAGE_SECTION_HEADER

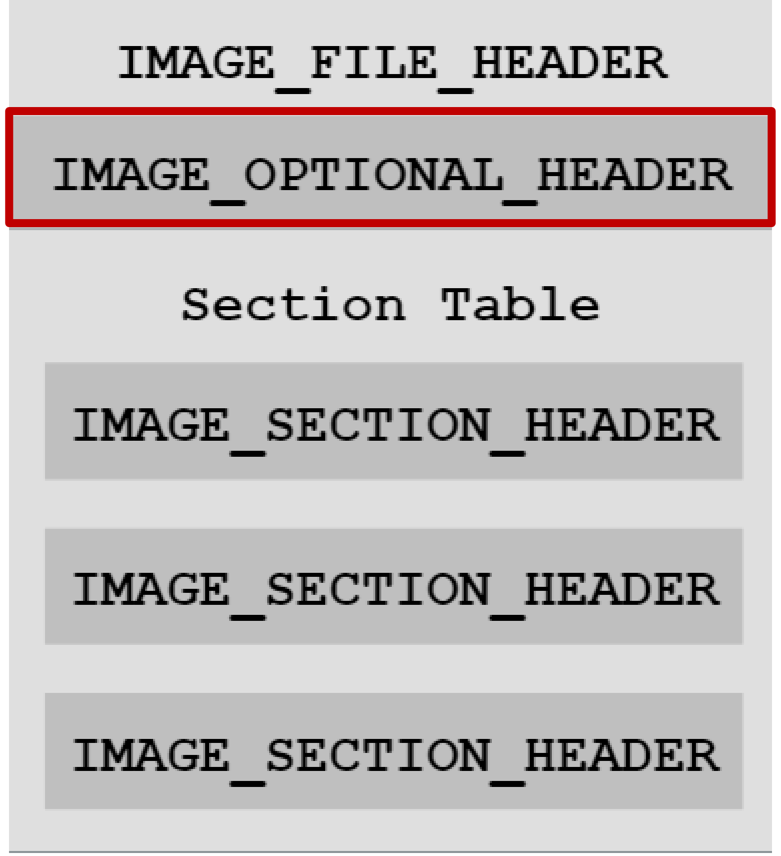
IMAGE_SECTION_HEADER

IMAGE_SECTION_HEADER

The IMAGE_OPTIONAL_HEADER

- Not optional, unlike what the name suggests

- Contains lots of important metadata:
  - AddressOfEntryPoint
  - Sizes of various parts of the file that get loaded into memory
  - Minimum versions of operating system, linker, image, subsystem



IMAGE_FILE_HEADER

IMAGE_OPTIONAL_HEADER

Section Table

IMAGE_SECTION_HEADER

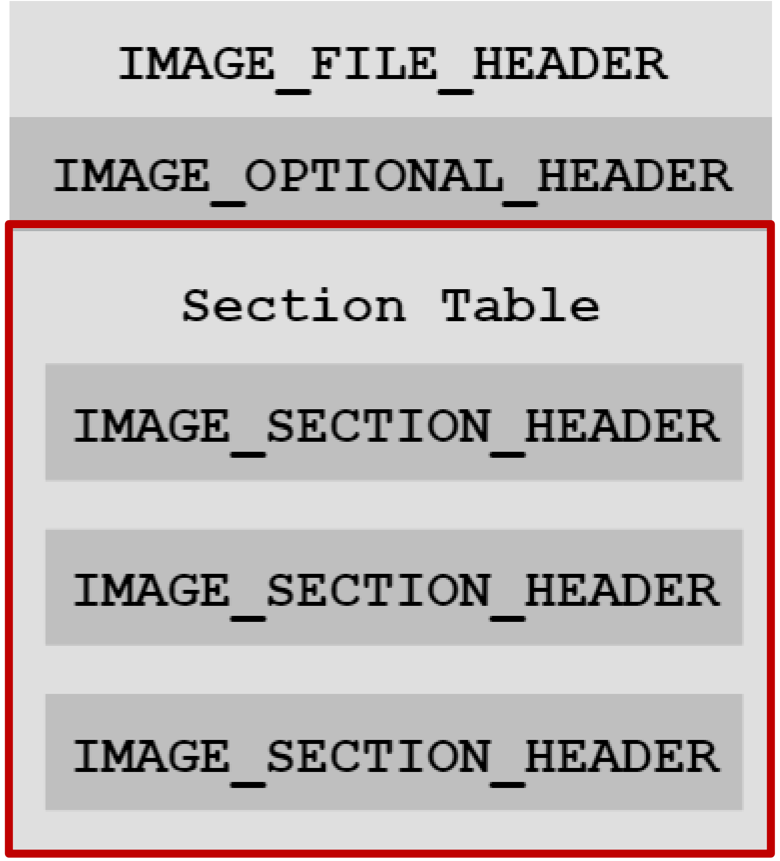
IMAGE_SECTION_HEADER

IMAGE_SECTION_HEADER

The Section Table

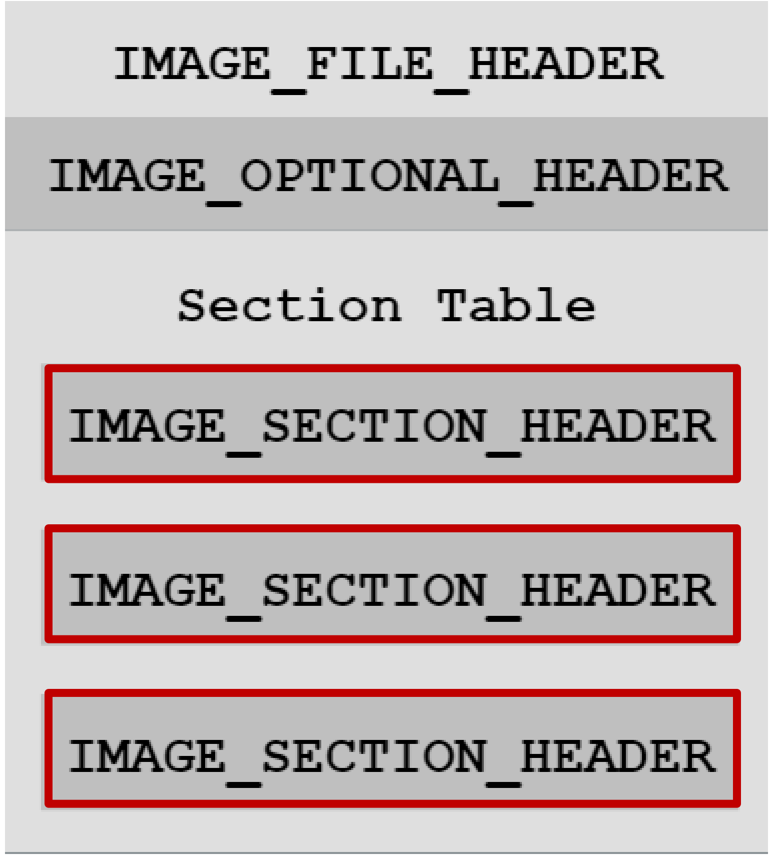
- Each section corresponds to a contiguous area of memory in a process

- Section table contains an array of IMAGE_SECTION_HEADERs



IMAGE_SECTION_HEADERs

- Each contains that section’s:
  - Name
  - VirtualAddress
  - VirtualSize
  - SizeOfRawData
  - Characteristics



Common PE Sections

Section name	Contents
.text	Executable code
.data	Initialized data
.idata	Import Address Table
.rsrc	Resource Directory Table
.rdata	Read-only initialized data

- Many other common section names
- Unusual section names are a malicious indicator

PE File Format Demo
(Detect it Easy and PE-Bear)

Lab03-03.exe

Imports

- The Import Address Table (IAT) lists which functions a file is importing from various DLLs.

- Knowing which functions a file imports tells an analyst what actions that file can potentially do
  - Especially imports that are part of the Windows API

- Commonly the second step in basic static analysis, after investigating strings

Resources

- Additional file(s) or data contained within a PE file

- In legitimate files, resources are often icons, language packs, an application manifest, etc.

- Malware often hides things in the Resources section!
  - Configuration data
  - Another “embedded” executable file

Resources and Imports Demo

Lab03-03.exe