# Malware Analysis

Introduction to Malware and Basic Static Analysis

Compiled by: Dharmesh D. Dave | Asst. Prof. | National Forensic Sciences University

# Introduction

- “Malware is a peace of malicious code which is made in order to harm or damage the computer system, network or any such digital devices.”
- ”A computer program that is covertly placed onto a computer with the intent to compromise the privacy, accuracy, or reliability of the computer’s data, applications, or operating system.” – NIST
- Types of Malwares:
- Virus, Worm, Trojan, Botnet, APT, Ransomware, Spyware, Adware, Rootkits, Logic Bomb, Backdoor, Launcher, Downloader, etc.

# First Virus

- There were some malware for other platforms before 1986, but in 1986 first malware for PC was appeared.
- That was written by Basit and Amjad who were only 17 and 24 years old at that time.
- The name of the virus was Brain, that is considered as the first virus written for windows based PCs.
- The brothers owns the company called Brain Telecommunication Ltd., the largest ISP of Pakistan.
- Brain motivated the technicians at IBM to create the first antivirus software for the general consumer in 1987.

# First Worm

- At around 8:30 p.m. on November 2, 1988, a maliciously clever program was unleashed on the Internet from a computer at the Massachusetts Institute of Technology.
- The worm only targeted computers running a specific version of the Unix operating system, but it spread widely because it featured multiple vectors of attack. For example, it exploited a backdoor in the Internet's electronic mail system and a bug in the "finger" program that identified network users. It was also designed to stay hidden.
- The culprit was a 23-year-old Cornell University graduate student named Robert Tappan Morris, hence the worm is known as Morris Worm.
- The country's first computer emergency response team was created in Pittsburgh at the direction of the Department of Defense.

# First Trojan

- First trojan horse was possibly embedded in a game intended for the UNIVAC 1108 computer system in 1975, known as the Pervading Animal developed by computer programmer John Walker.
- At the time, "animal programs," which try to guess which animal the user is thinking of with a game of 20 questions, were extremely popular.
- Walker created PERVADE, which installed itself along with ANIMAL. While playing the game, PREVADE examined all computer directories available to the user and then made a copy of ANIMAL in any directories where it wasn't already present. There was no malicious intent here, but ANIMAL and PREVADE fit the definition of a Trojan.

# First Botnet

- The first bots used on IRC were Jyrki Alakuijala's Puppe, which was a chatbot around 1988.
- The first bot used to index web pages was WebCrawler, created in 1994. It was first used by AOL in 1995.
- In 1999 Sub7, a Trojan, and Pretty Park, a worm. These malware introduced the concept of connecting to an Internet Relay Chat (IRC) channel to listen for malicious commands.
- In 2000 the Global Threat Bot aka GTbot which was a fake mIRC client program that can run custom scripts in response to IRC events and, more importantly, that it has access to raw TCP and UDP sockets, capable of some of the first Denial of Service attacks.

# First Rootkits

- 1994 First SunOS rootkits found.
- 1996 First Linux rootkits appear in the wild.
- 1997 Loadable kernel module–based rootkits are mentioned in Phrack.
- 1999 NT Rootkit, the first Windows rootkit, is released by Greg Hoglund.
- In 2005 Sony BMG published CDs with copy protection and digital rights management by a software that included a music player but silently installed a rootkit which limited the user's ability to access the CD.
- Software engineer Mark Russinovich, who created the rootkit detection tool RootkitRevealer, discovered the rootkit on one of his computers.[

# First Spyware

- The word 'spyware' was used for the first time publicly in October 1995.
- In early 2001, Steve Gibson of Gibson Research realized that advertising software had been installed on his system, and suspected it was stealing his personal information. After analysis, he determined that it was adware from the companies Aureate (later Radiate) and Conducent. Gibson developed and released the first anti-spyware program, OptOut. Many more have appeared since then.
- Later in 2000, a parent using ZoneAlarm was alerted to the fact that "Reader Rabbit," educational software marketed to children by the Mattel toy company, was surreptitiously sending data back to Mattel. Since then, "spyware" has taken on its present sense.

# First Adware

- In the beginning, meaning from roughly 1995 on, industry experts considered the first ad-supported software to be part of the larger category of spyware.
- Adware, also known as advertisement-supported software, generates revenue for its developers by automatically generating adverts on your screen, usually within a web browser.
- Adware can take a variety of forms from display and banner ads to full-screen ads, videos, and pop-ups.

# First Adware

- Computer adware infection signs
  - An unexpected change in your web browser home page
  - Web pages that you visit not displaying correctly
  - Being overwhelmed with pop-up ads — sometimes even if not browsing the internet
  - Slow device performance
  - Device crashing
  - Reduced internet speeds
  - Redirected internet searches
  - Random appearance of a new toolbar or browser add-ons

# First Ransomware

- In the late 1980s, criminals were already holding encrypted files hostage in exchange for cash sent via the postal service. One of the first ransomware attacks ever documented was the AIDS trojan (PC Cyborg Virus) that was released via floppy disk in 1989.

# First Logic Bomb

- It is thought to be the first logic bomb attack between the US and the Soviet Union during the Cold War, which occurred in 1982.
- The CIA received evidence that a KGB spy had stolen plans for a sophisticated control system, as well as its software, from a Canadian corporation, which had to be utilized on a Siberian pipeline. Apparently, in the system, a logic bomb had been coded by the CIA to sabotage the enemy.
- Since the introduction of the computer virus, the logic bomb attacks have started in television, movies as well as real life.

# First Time Bomb

- It is thought to be the first logic bomb attack between the US and the Soviet Union during the Cold War, which occurred in 1982.
- The CIA received evidence that a KGB spy had stolen plans for a sophisticated control system, as well as its software, from a Canadian corporation, which had to be utilized on a Siberian pipeline. Apparently, in the system, a logic bomb had been coded by the CIA to sabotage the enemy.
- Since the introduction of the computer virus, the logic bomb attacks have started in television, movies as well as real life.

# The Goals of Malware Analysis

- The purpose of malware analysis is usually to provide the information you need to respond to a network intrusion.
- Once you identify which files require full analysis, malware analysis can be used to develop host-based and network signatures.
- *Host-based signatures*, or indicators, are used to detect malicious code on victim computers. These indicators often identify files created or modified by the malware or specific changes that it makes to the registry.
- *Network signatures* are used to detect malicious code by monitoring network traffic.

# Analysis Techniques

- The purpose of malware analysis is usually to determine exactly what happened, and to ensure that all infected machines and files are located.
- Goal of analyzing suspected malware is typically to determine exactly what a particular suspect binary can do, how to detect it, and how to measure its damage.
- There are two fundamental approaches to malware analysis:
- **static** and **dynamic**.
  - – Static analysis involves examining the malware without running it.
  - – Dynamic analysis involves running the malware.
  - – Both techniques are further categorized as **basic** or **advanced**.

# Basic Static Analysis

- Basic static analysis consists of examining the executable file without viewing the actual instructions.
- Basic static analysis can confirm whether a file is malicious, provide information about its functionality, and sometimes provide information that will allow you to produce simple network signatures.
- Basic static analysis is straightforward and can be quick, but it's largely ineffective against sophisticated malware, and it can miss important behaviours.

# Basic Dynamic Analysis

- Basic dynamic analysis techniques involve running the malware and observing its behavior on the system in order to remove the infection, produce effective signatures, or both.
- However, before you can run malware safely, you must set up an environment that will allow you to study the running malware without risk of damage to your system or network.
- Like basic static analysis techniques, basic dynamic analysis techniques can be used by most people without deep programming knowledge, but they won't be effective with all malware and can miss important functionality.

# Advanced Static Analysis

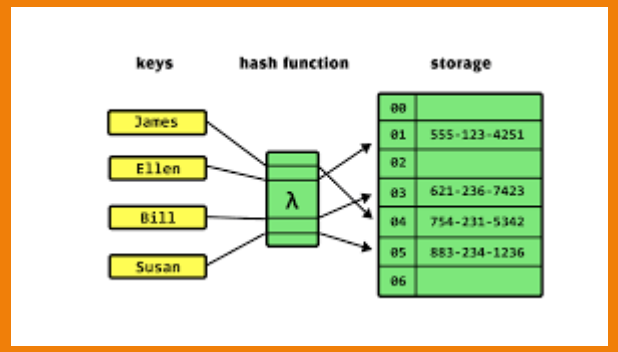
- Advanced static analysis consists of reverse-engineering the malware's internals by loading the executable into a disassembler and looking at the program instructions in order to discover what the program does.
- The instructions are executed by the CPU, so advanced static analysis tells you exactly what the program does.
- However, advanced static analysis has a steeper learning curve than basic static analysis and requires specialized knowledge of disassembly, code constructs, and Windows operating system concepts.

# Advanced Dynamic Analysis

- Advanced dynamic analysis uses a debugger to examine the internal state of a running malicious executable.
- Advanced dynamic analysis techniques provide another way to extract detailed information from an executable.
- These techniques are most useful when you're trying to obtain information that is difficult to gather with the other techniques.

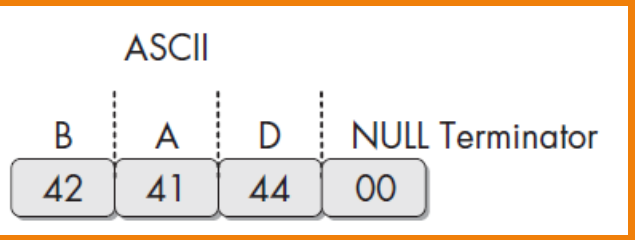
# Basic Static (VirusTotal)

- VirusTotal allow you to upload a file for scanning by multiple antivirus engines. VirusTotal generates a report that provides the total number of engines that marked the file as malicious, the malware name, and, if available, additional information about the malware.



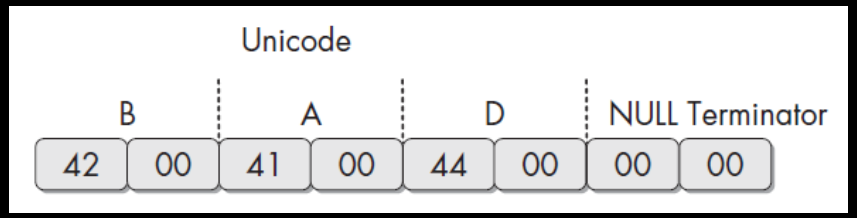

# Basic Static (Hashing)

- Hashing is a common method used to uniquely identify malware. The malicious software is run through a hashing program that produces a unique hash that identifies that malware (a sort of fingerprint).
- The Message-Digest Algorithm 5 (MD5) hash function is the one most commonly used for malware analysis, though the Secure Hash Algorithm 1 (SHA-1) is also popular.
- Once you have a unique hash for a piece of malware, you can use it as follows:
  - Use the hash as a label.
  - Share that hash with other analysts to help them to identify malware.
  - Search for that hash online to see if the file has already been identified.




# Basic Static (Strings)

- A string in a program is a **sequence of characters** such as "the." A program contains strings if it prints a message, connects to a URL, or copies a file to a specific location.
- Searching through the strings can be a simple way to get hints about the functionality of a program. Typically strings are stored in either ASCII or Unicode format.
- Both ASCII and Unicode formats store characters in sequences that end with a NULL terminator to indicate that the string is complete. ASCII strings use 1 byte per character, and Unicode uses 2 bytes per character.



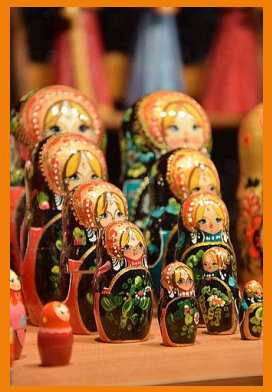
# Basic Static (Strings)

- Sometimes the strings detected by the Strings program are **not actual strings**. For example, if Strings finds the sequence of bytes 0x56, 0x50, 0x33, 0x00, it will interpret that as the string VP3.
- But those bytes may not actually represent that string; they could be a memory address, CPU instructions, or data used by the program. Strings leaves it up to the user to filter out the invalid strings.
- Typically, if a string is short and doesn't correspond to words, it's probably meaningless.

# Basic Static (Strings)

```
C:>strings bp6.ex_
VP3
VW3
t$@
D$4
99.124.22.1 ❹
e-@
GetLayout ❶
GDI32.DLL ❸
SetLayout ❷
M}C
Mail system DLL is invalid.!Send Mail failed to send message. ❺
```

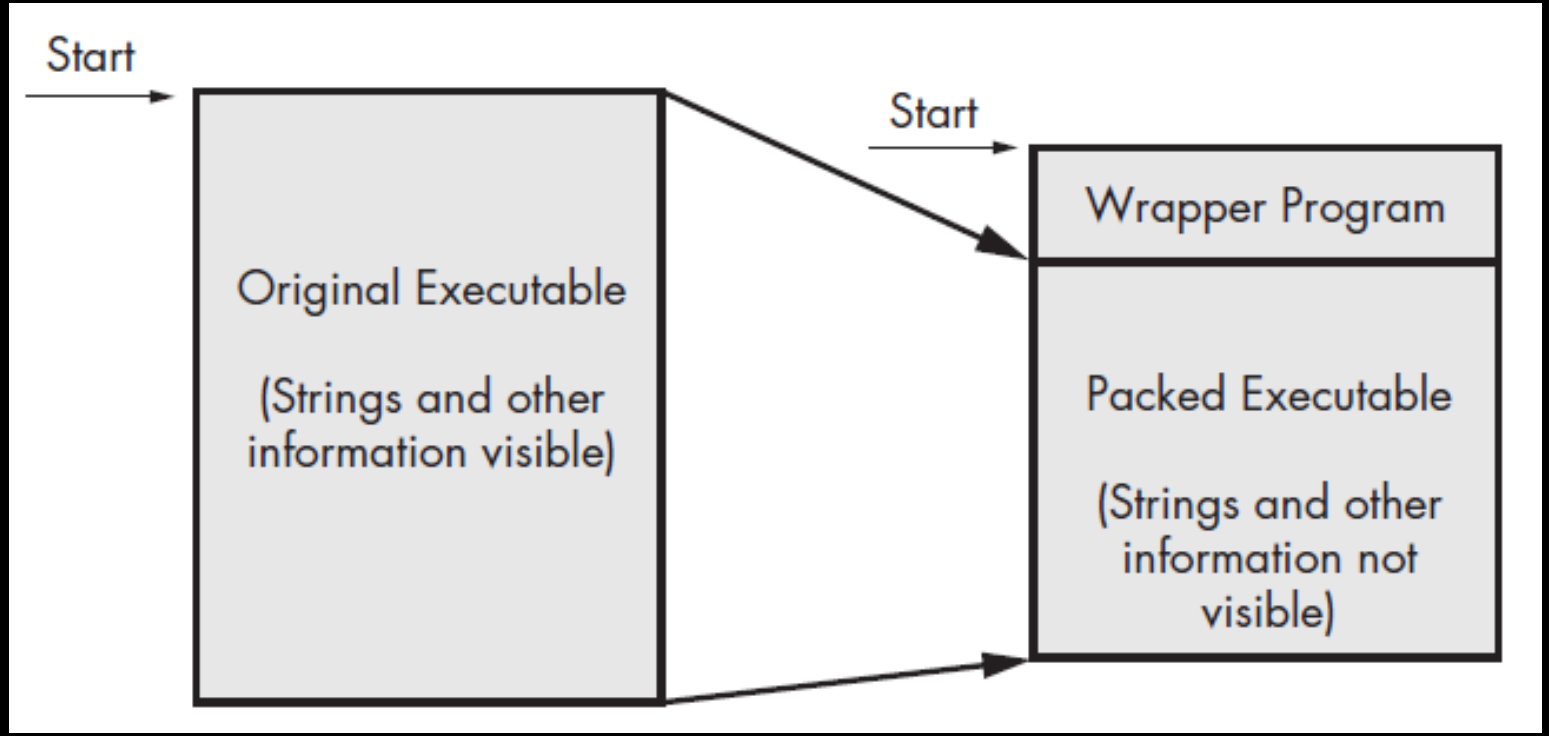
- Use the following tools to see strings:
- 1) **Strings.exe** from sysinternals
- 2) **Bintext** to avoid garbage values



# Packed & Obfuscated Malware

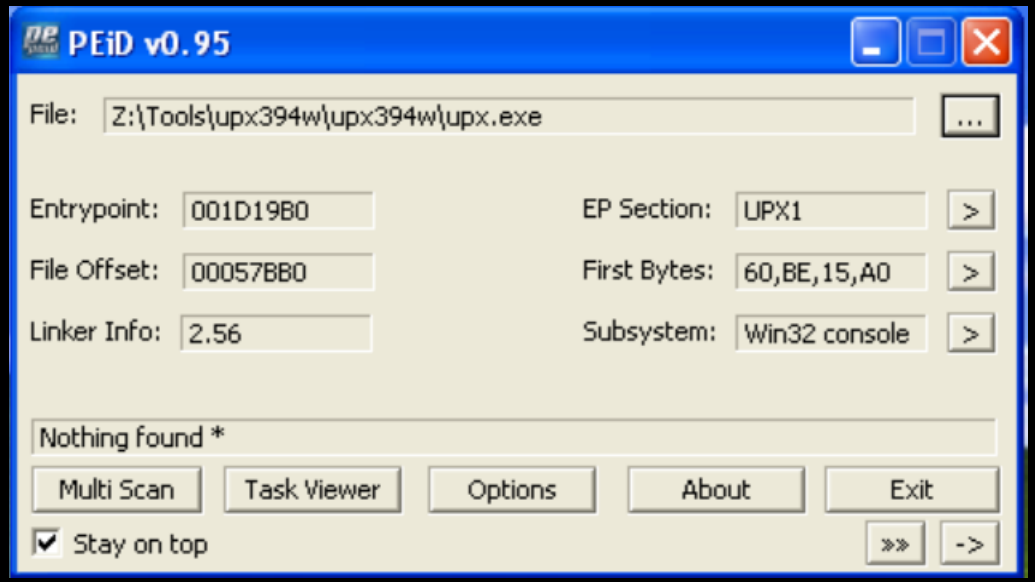
- Obfuscated programs are ones whose execution the malware author has attempted to hide.
- Packed programs are a subset of obfuscated programs in which the malicious program is compressed and cannot be analyzed.
- Both techniques will severely limit your attempts to statically analyze the malware.
- From Strings if we found that it has only a few strings, it is probably either obfuscated or packed, suggesting that it may be malicious.
- Packed and obfuscated code will often include at least the functions **LoadLibrary** and **GetProcAddress**, which are used to load and gain access to additional functions.

# Packed & Obfuscated Malware



# Basic Static (PEiD)

- One way to detect packed files is with the PEiD program. You can use PEiD to detect the type of packer or compiler employed to build an application, which makes analyzing the packed file much easier.
- *PEiD can be subject to vulnerabilities. For example, PEiD version 0.92 contained a buffer overflow that allowed an attacker to execute arbitrary code.*



# PE File Format

- The Portable Executable (PE) file format is used by Windows executables, object code, and DLLs.
- The PE file format is a data structure that contains the information necessary for the Windows OS loader to manage the wrapped executable code.
- PE files begin with a header that includes information about the code, the type of application, required library functions, and space requirements.
- The information in the PE header is of great value to the malware analyst.

# Linked Libraries and Functions

- One of the most useful pieces of information that we can gather about an executable is the list of functions that it imports
- Imports are functions used by one program that are actually stored in a different program, such as code libraries that contain functionality common to many programs. Code libraries can be connected to the main executable by linking.
- Code libraries can be linked statically, at runtime, or dynamically.

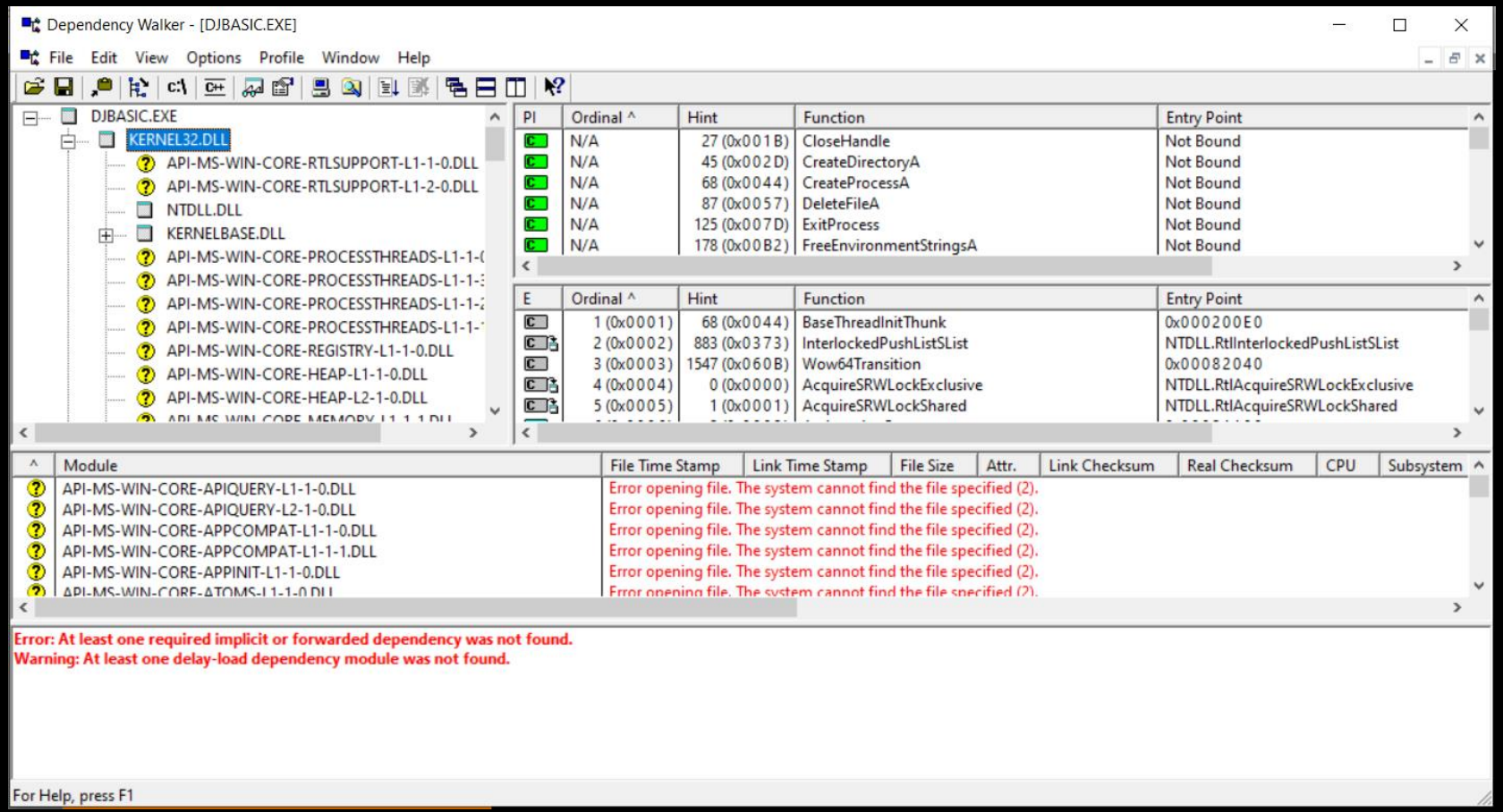
# Linked Libraries and Functions

- Three types of Linking of Libraries:
- 1) Static

- 2) Dynamic

- 3) Runtime

# Basic Static (Dependency Walker)



# Common DLLs

| DLL | Description |
|---|---|
| ***Kernel32.dll*** | This is a very common DLL that contains **core functionality**, such as access and manipulation of memory, files, and hardware. |
| ***Advapi32.dll*** | This DLL provides access to **advanced core Windows components** such as the Service Manager and Registry. |
| ***User32.dll*** | This DLL contains all the **user-interface components**, such as buttons, scroll-bars, and components for controlling and responding to user actions. |
| ***Gdi32.dll*** | This DLL contains functions for displaying and manipulating **graphics**. |
| ***Ntdll.dll*** | This DLL is the **interface to the Windows kernel**. Executables generally do not import this file directly, although it is always imported indirectly by *Kernel32.dll*. If an executable imports this file, it means that the author intended to use functionality not normally available to Windows programs. Some tasks, such as hiding functionality or manipulating processes, will use this interface. |
| ***WSock32.dll and Ws2_32.dll*** | These are **networking DLLs**. A program that accesses either of these most likely connects to a network or performs network-related tasks |
| ***Wininet.dll*** | *This DLL contains **higher-level networking functions** that implement protocols such as FTP, HTTP, and NTP.* |

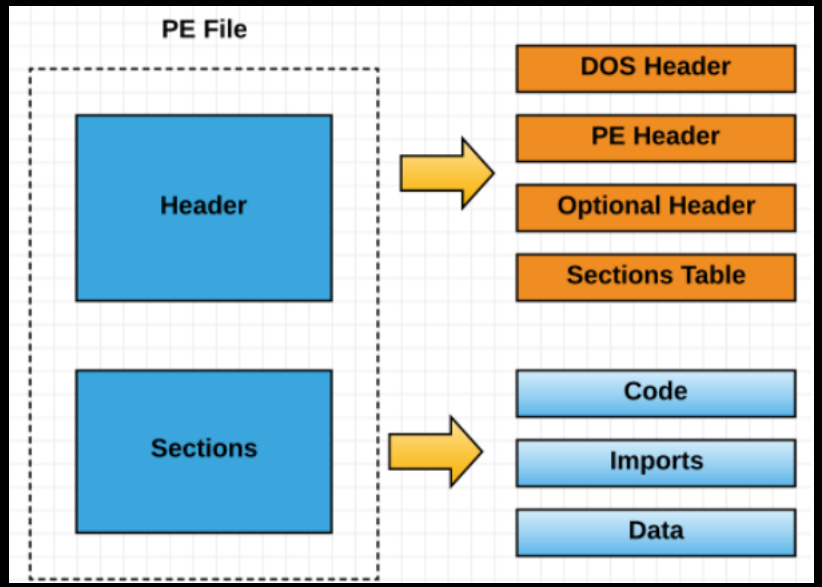
# Functions

- **IMPORT:**
- The PE file header also includes information about specific functions used by an executable.
- The names of these Windows functions can give you a good idea about what the executable does.

- **EXPORT:**
- Like imports, PE file export functions to interact with other programs and code.
- DLL implements one or more functions and exports them for use by an executable that can then import and use them.
- EXEs are not designed to provide functionality for other EXEs, and exported functions are rare.

# PE File Header and Sections

- The PE file format contains a header followed by a series of sections. The header contains metadata about the file itself.
- Sections contains useful information and come common sections are as below:
- i) .text
- ii) .rdata
- iii) .data
- iv) .rsrc
- v) .reloc
- vi) .pdata

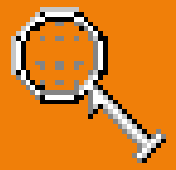



More Details: https://learn.microsoft.com/en-us/windows/win32/debug/pe-format

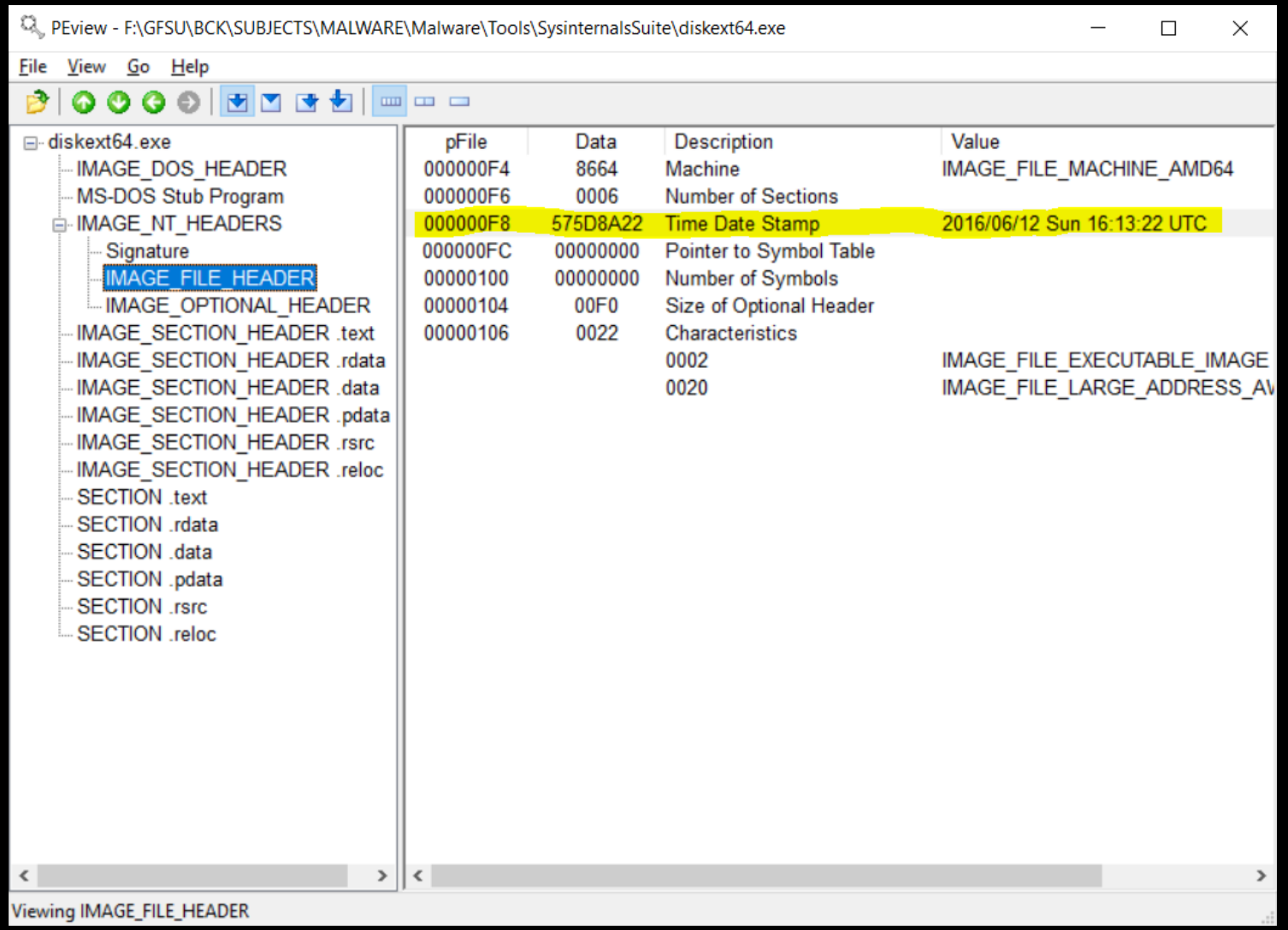
# PE File Header and Sections

| Sections | Description |
|---|---|
| ***.text*** | Contains instructions (executable code) that the CPU executes. Generally, this is the only section that can execute. |
| ***.rdata*** | Holds read-only data that is globally accessible within the program. Typically contains the import and export information. |
| ***.data*** | Stores global data accessed throughout the program. Local data is not stored in this section, or anywhere else in the PE file. |
| ***.idata*** | Sometimes present and stores the import function information; if this section is not present, the import function information is stored in the .rdata section |
| ***.edata*** | Sometimes present and stores the export function information; if this section is not present, the export function information is stored in the .rdata section |
| ***.pdata*** | Present only in 64-bit executables and stores exception-handling information |
| ***.rsrc*** | Stores the resources used by the executable that are not considered part of the executable, such as icons, images, menus, and strings. Strings can be stored either in the .rsrc section or in the main program. |
| ***.reloc*** | Contains information for relocation of library files |

More Details: https://learn.microsoft.com/en-us/windows/win32/debug/pe-format



# Basic Static (Examining PE Files with PEview)

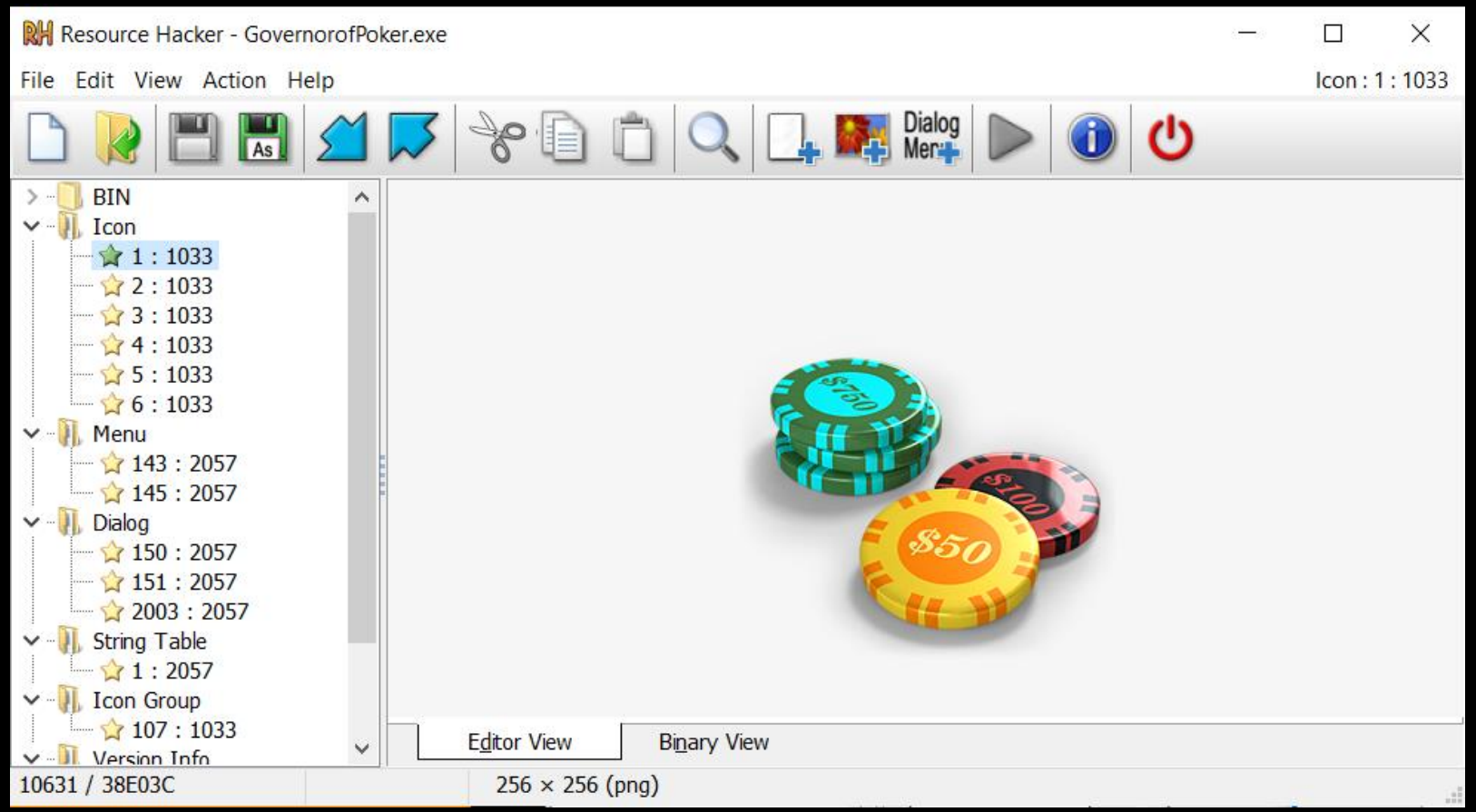


# PE Header Summary

| Field | Information revealed |
|---|---|
| ***Imports*** | Functions from other libraries that are used by the malware. |
| ***Exports*** | Functions in the malware that are meant to be called by other programs or libraries. |
| ***Time Date Stamp*** | Time when the program was compiled. |
| ***Sections*** | Names of sections in the file and their sizes on disk and in memory. |
| ***Subsystem*** | Indicates whether the program is a command-line or GUI application. |
| ***Resources*** | Strings, icons, menus, and other information included in the file. |

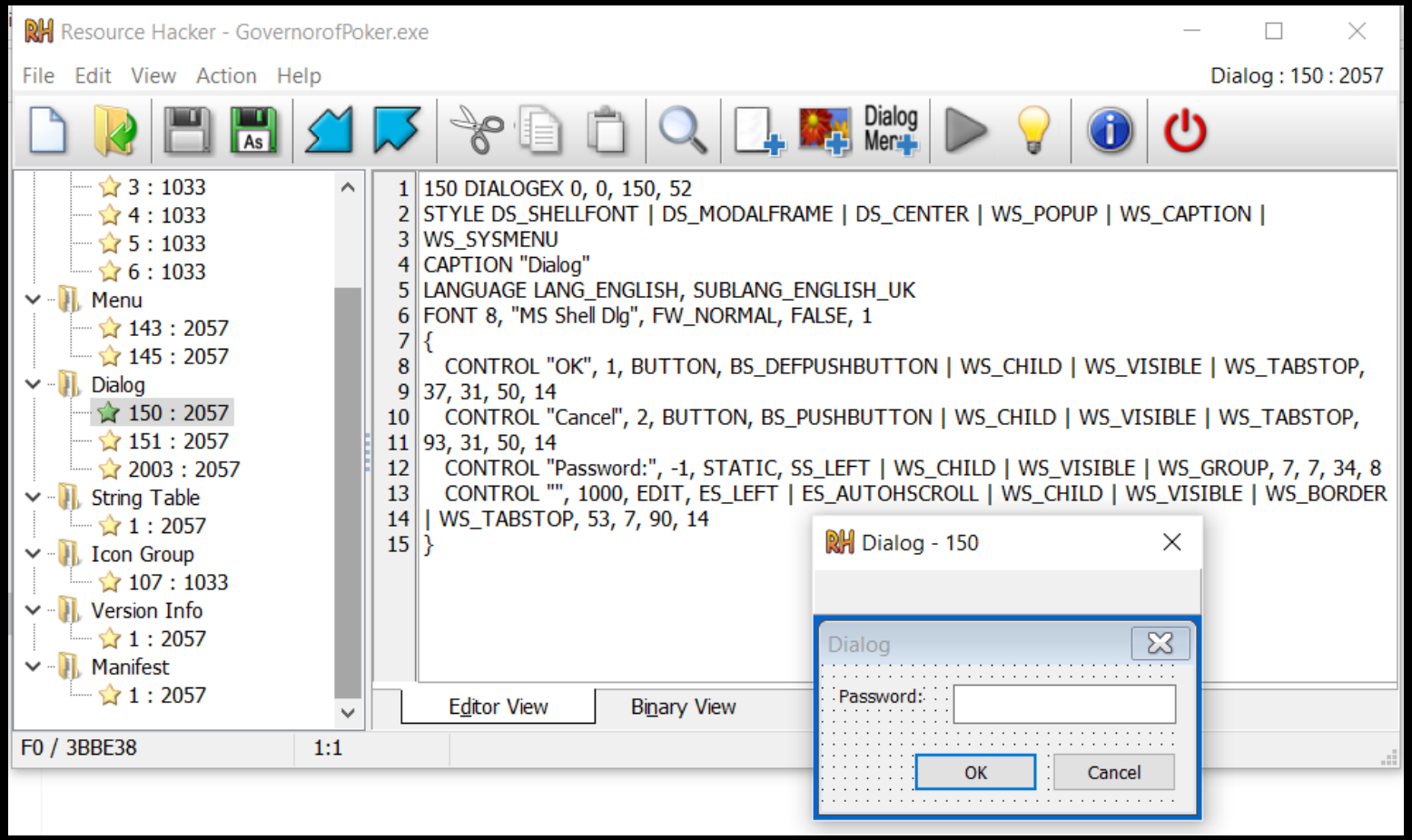
# Resource Hacker

## Basic Static (Resource Hacker)

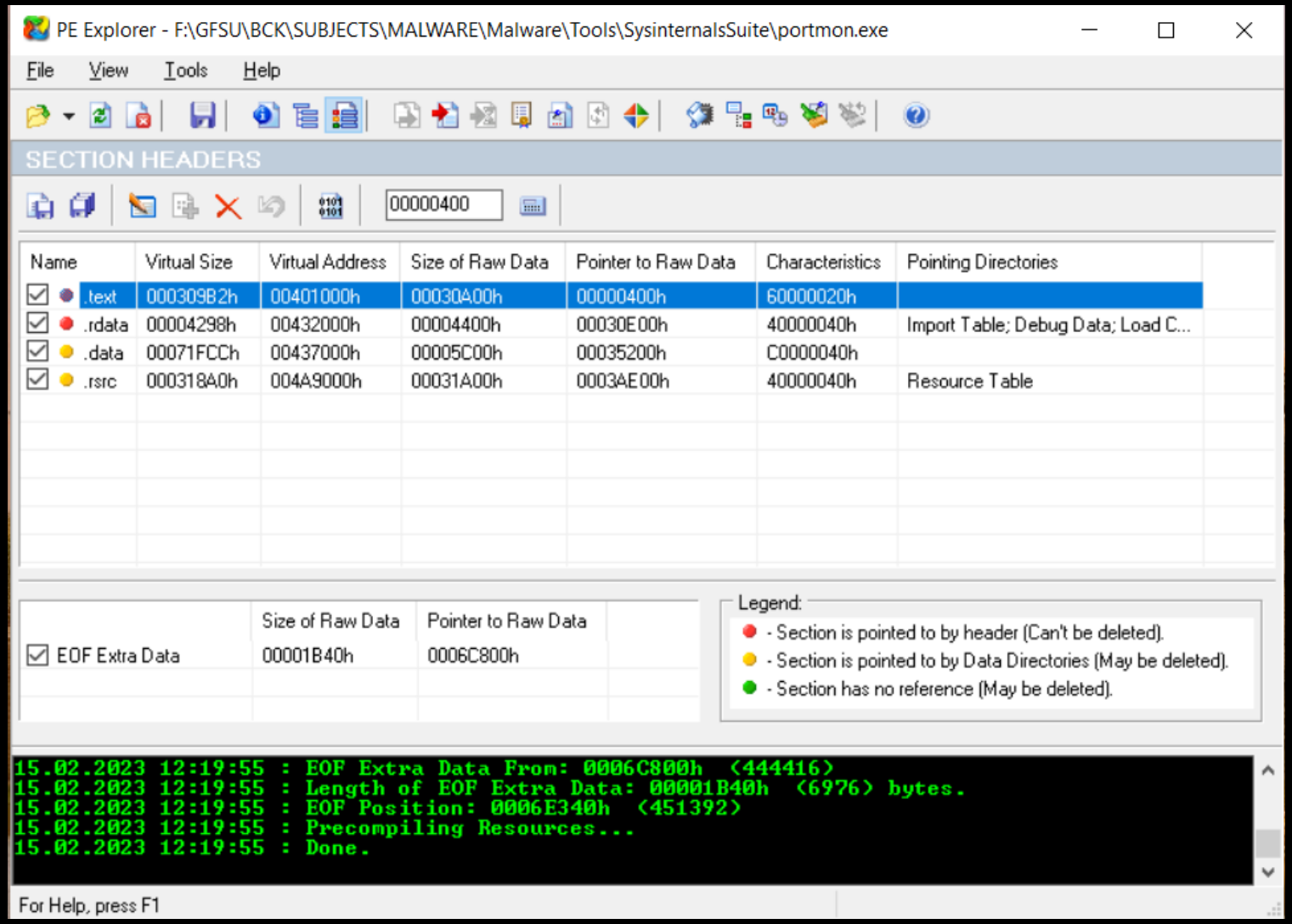


# Resource Hacker

## Basic Static (Resource Hacker)



# Basic Static (PE Explorer)



# Limitations of Basic Static Analysis

- Only few functionality can be observed and can not confirm about malware.
- Obfuscated or Packed Binary can not be analyzed.
- Runtime Linking Libraries can not be analyzed / detected in static analysis.

# AV Signature

- AV products rely heavily on **signatures**, be it a simple hash, a collection of strings, a series of bytes representing code, or a complex set of identifying rules, to detect a piece of malware.
- But no matter how the signature is created, it is based mostly on the malware code.
- A matching signature results in a scanned file being tagged as malicious.
- If there are no matching signatures, then the file is assumed to be benign.

# AV Signature

- During scanning of a binary, be it on disk or in memory, the scan engine looks for the **entry point** of the executable.
- The entry point is where the first instruction is located, and it is the key to following the **execution flow** of the binary.
- The entry point leads to the **binary code** itself.
- When the binary code is located, the scan engine **compares** all the detection signatures it has in its database to the binary code.
- If a match is found, then the binary is tagged as malicious.


# CLAM AV

- ClamAV is an open source (GPLv2) anti-virus toolkit, designed especially for e-mail scanning on mail gateways.
- ClamAV is designed to scan files quickly.
- ClamAV's bytecode signature runtime, powered by either LLVM or our custom bytecode interpreter, allows the ClamAV signature writers to create and distribute very complex detection routines and remotely enhance the scanner’s functionality.
- Signed signature databases ensure that ClamAV will only execute trusted signature definitions.

More Details: https://github.com/Cisco-Talos/clamav https://docs.clamav.net/Introduction.html


# CLAM AV

- > sudo apt-get install clamav OR download package and install
- The ClamAV signatures by default exist in compressed, binary files.
- We can see the criteria for an existing rule to create a modified version of an existing signature.
- Luckily, ClamAV comes with a tool that allows you to decompress and inspect the signatures in its database.
- Typically, the ClamAV signatures exist in /var/lib/clamav on Linux systems.
- You should expect to find main.cld and daily.cld or main.cvd file.

More Details: https://github.com/Cisco-Talos/clamav https://docs.clamav.net/Introduction.html


# CLAM AV

- The main.cld file contains the primary base of signatures and daily.cld contains incremental daily updates.
- **Practical:**
  - > mkdir clamAV_Main
  - > cd clamAV_Main/
  - > sigtool -u /var/lib/clamav/main.cvd
  - > ls -al
  - > cat main.hdb

More Details: https://github.com/Cisco-Talos/clamav https://docs.clamav.net/Introduction.html


# CLAM AV

- **ClamAV Extended signature format:**
- **SigName:Target:Offset:HexadecimalSignature**
  - SigName: name of signature
  - Target:  A number specifying the type of the target file
  - Offset: * = any, n = absolute offset, EOF-n = end of file minus n bytes
  - HexSignature: The body-based content matching format.
  - You can **use sigtool --hex-dump to convert any data into a hex-string**
- Use the generated sample hex-string based signature in file--**TestHelloWorld:0:*:48656c6c6f20576f726c640a**
- To use these signatures, you need to place them into a file with a **.ndb** extension.
- When using the custom signature database, you need to specify its location on the command line for clamscan using the -d flag.
- > clamscan -d signature name.ndb *File*

More Details: https://github.com/Cisco-Talos/clamav  https://docs.clamav.net/Introduction.html


# CLAM AV

| Target Type | Description |
|---|---|
| 0 | any file |
| 1 | Portable Executable, both 32- and 64-bit |
| 2 | OLE2 containers, including specific macros. Primarily used by MS Office and MSI installation files |
| 3 | HTML (normalized) |
| 4 | Mail file |
| 5 | Graphics |
| 6 | ELF |
| 7 | ASCII text file (normalized) |
| 8 | Unused |
| 9 | Mach-O files |
| 10 | PDF files |
| 11 | Flash files |
| 12 | Java class files |

More Details: https://github.com/Cisco-Talos/clamav https://docs.clamav.net/Introduction.html


# CLAM AV

```
ddave@ddave-5570: ~/Desktop/clamAV
ddave@ddave-5570:~/Desktop/clamAV$ sigtool -u /var/lib/clamav/main.cld
ddave@ddave-5570:~/Desktop/clamAV$ ls
abc.txt  main.cdb  main.fp   main.hsb   main.ldb  main.msb  main.sfp
COPYING  main.crb  main.hdb  main.info  main.mdb  main.ndb  win.exe
ddave@ddave-5570:~/Desktop/clamAV$
```

```
ddave@ddave-5570: ~/Desktop/clamAV
ddave@ddave-5570:~/Desktop/clamAV$ sigtool --hex-dump
GetProcAddress
47657450726f63416464726573730a
ddave@ddave-5570:~/Desktop/clamAV$ clamscan -d test.ndb win.exe
/home/ddave/Desktop/clamAV/win.exe: OK

----------- SCAN SUMMARY -----------
Known viruses: 1
Engine version: 0.103.6
Scanned directories: 0
Scanned files: 1
Infected files: 0
Data scanned: 1.61 MB
Data read: 0.28 MB (ratio 5.72:1)
Time: 0.052 sec (0 m 0 s)
Start Date: 2023:02:20 18:48:14
End Date:   2023:02:20 18:48:14
```

# References


- Milošević, N. (2013). History of malware. arXiv preprint arXiv:1302.5392.
- https://us.norton.com/internetsecurity-malware-when-were-computer-viruses-first-written-and-what-were-their-original-purposes.html
- https://www.fbi.gov/news/stories/morris-worm-30-years-since-first-major-attack-on-internet-110218
- https://www.sciencedirect.com/topics/computer-science/trojan-horse
- https://www.kaspersky.co.in/resource-center/threats/a-brief-history-of-computer-viruses-and-what-the-future-holds
- https://abusix.com/resources/botnets/a-brief-history-of-bots-and-how-theyve-shaped-the-internet-today/
- https://www.trendmicro.co.uk/media/wp/botnet-chronicles-whitepaper-en.pdf
- https://www.first.org/resources/papers/conference2004/c17.pdf
- https://www.kaspersky.co.in/resource-center/threats/adware
- https://github.com/mandiant/flare-floss
- Hacking Exposed Malware & Rootkits by Michael Davis
- Practical Malware Analysis by Michael Sikorski