



MALWARE ANALYSIS

Basic Static Analysis

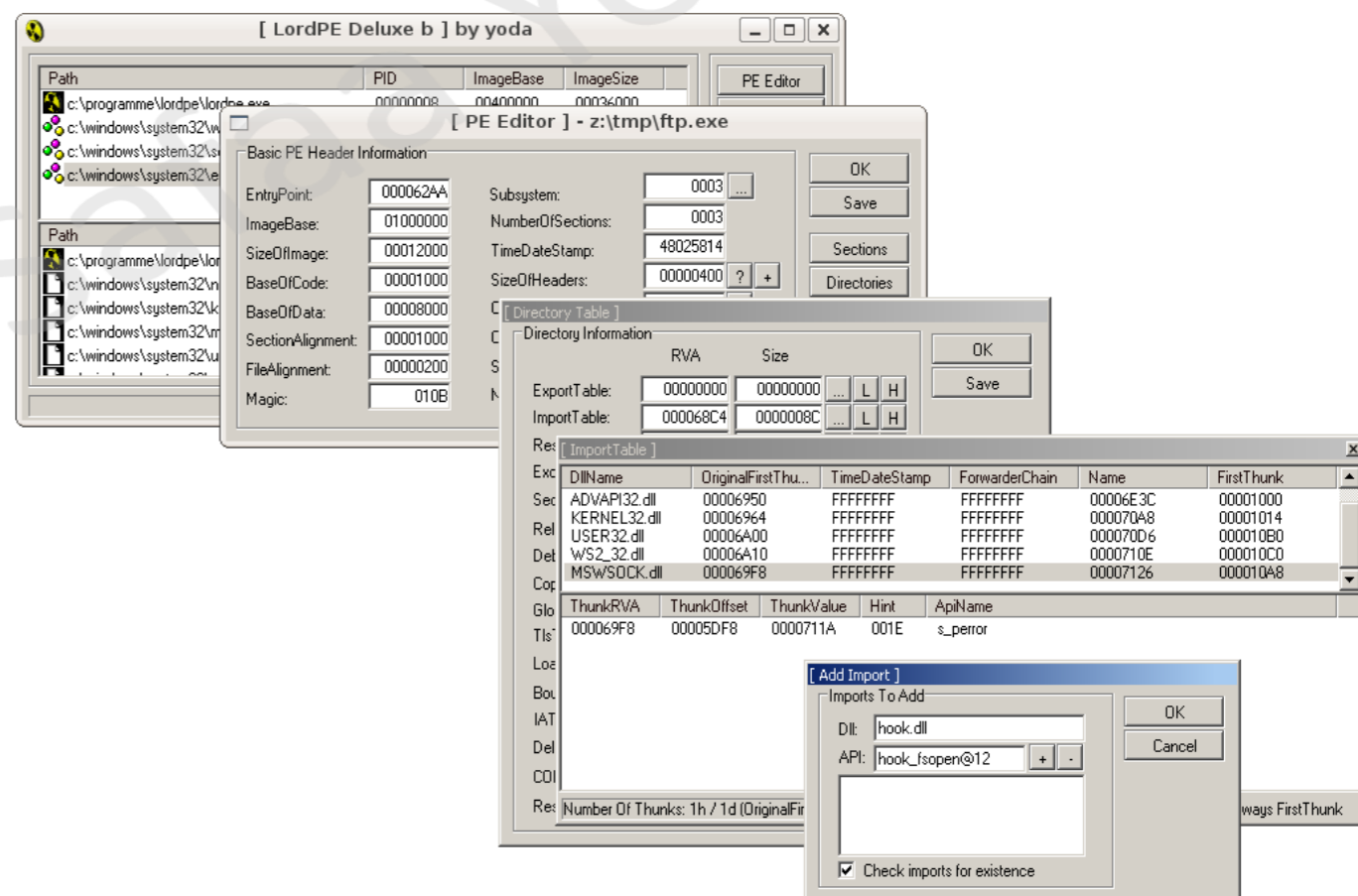
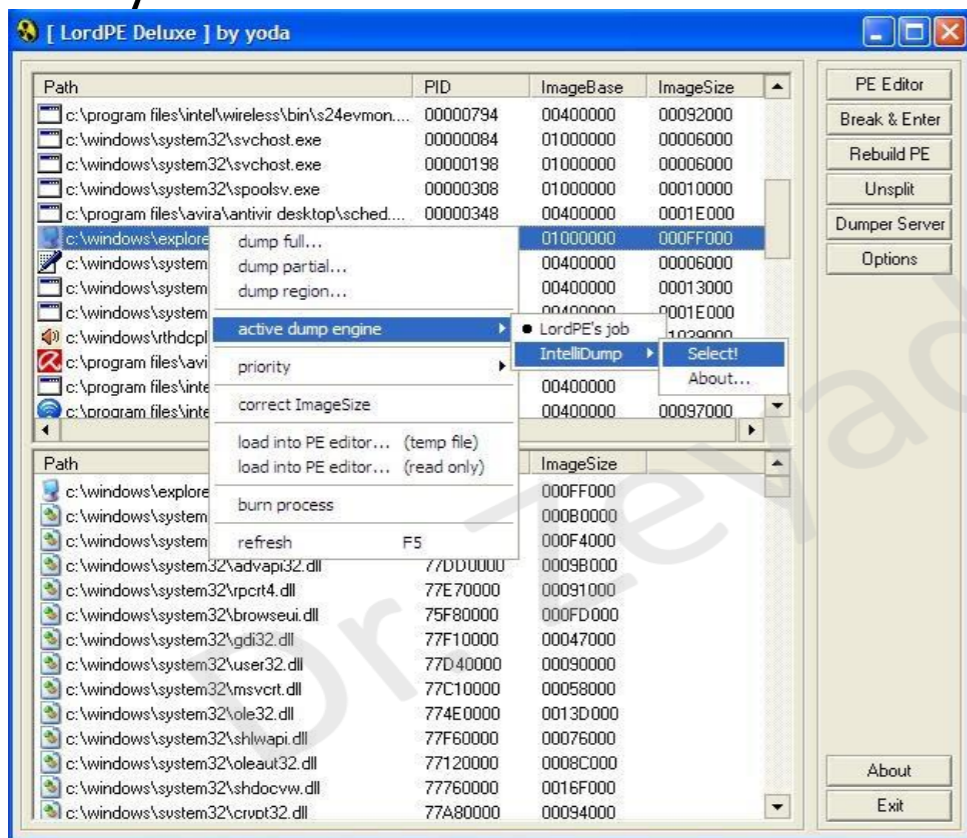
Dr. Zeyad Safaa Younus Saffawi

Portable Executable (PE) File Format

- The **Portable Executable (PE) format** is the standard file format for Windows executables (.exe), dynamic-link libraries (.dll), and object files (.obj).
- The PE format defines a data structure that contains all the information required by the Windows loader to manage and execute the wrapped code.
- The **PE header** provides critical metadata about the executable, including details about the code, application type, required libraries, and memory requirements.
- For malware analysts, the PE header is particularly valuable, as it offers insights into the internal structure of the program, its dependencies, and potential behavior. Examining the PE header can therefore assist in identifying anomalies, detecting malicious characteristics, and guiding further static and dynamic analysis.

LordPE Deluxe tool

- **LordPE Deluxe** is a **portable executable (PE) file editor** and a **system programmer's tool designed to view, edit, and manipulate various components of Windows executable files**. The application allows analysts to examine PE files in detail, modify headers, and explore the structure of executables and their associated dynamic-link libraries (DLLs). Additionally, LordPE Deluxe can be used to dump PE files from memory, optimize them, and analyze running programs, making it a valuable tool for both software development and malware analysis.



Linked Libraries and Functions

- **Imports** are functions used by a program that reside in a different program or library. They enable programmers to reuse existing code rather than rewriting it from scratch. These imported functions are connected to the main executable through a process known as **linking**. Libraries can be linked in three primary ways: **statically, at runtime, or dynamically**.
- For malware analysts, understanding how libraries are linked is crucial, as it provides insight into the dependencies and behavior of a program. By examining imported functions and their linking method, analysts can infer which external routines the malware relies on and predict how it interacts with the operating system or other software components, facilitating more effective analysis and detection.

Static Linking

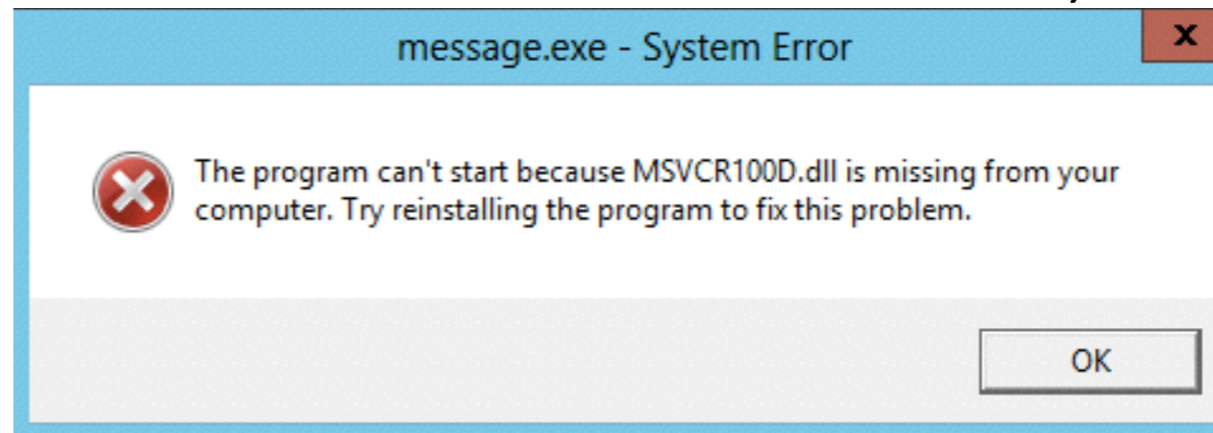
- **Static linking** is rarely used in Windows executables but is more commonly employed in Unix and Linux systems. In this linking method, all required code from the library is copied directly into the executable file.
- As a result, the executable becomes self-contained but significantly larger in size.
- Understanding static linking is important in malware analysis, as it affects file size, portability, and the visibility of external dependencies within the executable.

Runtime Linking

- **Dynamic linking at runtime** is uncommon in legitimate software but frequently observed in malware, particularly in packed or obfuscated programs.
- Unlike static linking, dynamically linked functions are connected to the necessary libraries only when they are required during execution, rather than at program startup. This technique is typically implemented using Windows API functions such as **LoadLibrary** and **GetProcAddress**.
- For malware analysts, understanding runtime dynamic linking is essential, as it helps reveal hidden dependencies and behaviors that may not be apparent through static analysis alone.

Dynamic Linking

- **Dynamic linking** is the most common method used in Windows executables. In this approach, the host operating system locates and loads the required libraries when the program is executed.
- The **PE header** explicitly lists all libraries and functions that the executable depends on, providing valuable information for analysis.
- For malware analysts, the names of imported libraries and functions can offer important clues about a program's intended behavior. For example, the presence of a function such as **URLDownloadToFile** strongly suggests that the program is designed to download external content, which may indicate malicious activity.



Dependency Walker tool

- **Dependency Walker** is a free tool for Microsoft Windows that allows analysts to examine the imported and exported functions of a portable executable (PE) file. The tool is particularly useful for identifying dynamically linked functions and understanding a program's dependencies.
- In general, legitimate software typically relies on a substantial number of **dynamic-link libraries (DLLs) to provide functionality**. In contrast, malware often imports very few DLLs, especially if it is packed or obfuscated, as the malicious code is designed to minimize visible dependencies and evade detection.
- By analyzing the DLL usage with Dependency Walker, security professionals can gain insight into the program's structure and potentially identify suspicious or malicious behavior.

Dependency Walker tool

Dependency Walker - [sqlservr.exe]

File Edit View Window Help

Tree View:

- SQLSERVER.EXE
 - KERNEL32.DLL
 - NTDLL.DLL
 - ADVAPI32.DLL
 - NTDLL.DLL
 - KERNEL32.DLL
 - NTDLL.DLL
 - RPCRT4.DLL
 - USER32.DLL
 - NTDLL.DLL
 - KERNEL32.DLL
 - NTDLL.DLL
 - GDI32.DLL
 - NTDLL.DLL
 - KERNEL32.DLL
 - NTDLL.DLL
 - RPCRT4.DLL
 - NTDLL.DLL
 - KERNEL32.DLL

Module ^	Time Stamp	Size	Attributes	Machine	Subsystem	Debug	Base	File Ver
ADVAPI32.DLL	08/26/02 9:45a	358,160	A	Intel x86	Win32 console	No	0x77DB0000	5.0.2195.5992
GDI32.DLL	08/26/02 9:45a	222,992	A	Intel x86	Win32 console	No	0x77F40000	5.0.2195.5907
KERNEL32.DLL	11/01/02 5:33p	708,880	A	Intel x86	Win32 console	No	0x77E80000	5.0.2195.6079
MSVCIRT.DLL	12/07/99 8:00a	77,878	A	Intel x86	Win32 GUI	Yes	0x780A0000	6.1.8637.0
MSVCRT.DLL	07/22/02 3:05p	290,869	A	Intel x86	Win32 GUI	Yes	0x78000000	6.1.9359.0

Imports & Exports in Dependency Walker

Dependency Walker - [Lab01-01.exe]

File Edit View Options Profile Window Help

Tree View:

- LAB01-01.EXE
 - KERNEL32.DLL
 - MSVCRT.DLL

Imports (PI) Table:

PI^	Ordinal	Hint	Function
	N/A	657 (0x0291)	malloc
	N/A	585 (0x0249)	exit
	N/A	211 (0x00D3)	_exit
	N/A	72 (0x0048)	_XcptFilter
	N/A	100 (0x0064)	__p__initenv
	N/A	88 (0x0058)	__getmainarg
	N/A	271 (0x010F)	_initterm

Exports (E) Table:

E^	Ordinal	Hint	Function
	108 (0x006C)	106 (0x006A)	_XcptFilter
	147 (0x0093)	145 (0x0091)	__getmainarg
	181 (0x00B5)	179 (0x00B3)	__p__initenv
	187 (0x00BB)	185 (0x00B9)	__p__commo
	192 (0x00C0)	190 (0x00BE)	__p__fmode
	212 (0x00D4)	210 (0x00D2)	__set_app_t
	214 (0x00D6)	212 (0x00D4)	__setuserm

Module List:

Module	File Time Stamp	Li
API-MS-WIN-CORE-CONSOLE-L1-1-0.DLL	01/03/2013 9:43p	01
API-MS-WIN-CORE-DATETIME-L1-1-0.DLL	01/03/2013 9:43p	01
API-MS-WIN-CORE-DEBUG-L1-1-0.DLL	01/03/2013 9:43p	01
API-MS-WIN-CORE-ERRORHANDLING-L1-1-0.DLL	01/03/2013 9:43p	01
API-MS-WIN-CORE-FIBERS-L1-1-0.DLL	01/03/2013 9:43p	01

For Help, press F1

Services.ex_ (malware)

Services.ex_ is a **malware sample** whose **imported DLLs** provide valuable insight into its **functionality**. By analyzing the libraries and functions that the program imports, analysts can infer its potential behavior, such as network communication, file manipulation, or interaction with system resources. Understanding these imports is therefore a critical step in uncovering the malicious actions of **services.ex_** and guiding further analysis.

- For example:
- **WS2_32.DLL** → network communication
- **KERNEL32.DLL** → system-level operations (processes, files, memory)



Common DLLs

DLL	Description
<i>Kernel32.dll</i>	This is a very common DLL that contains core functionality, such as access and manipulation of memory, files, and hardware.
<i>Advapi32.dll</i>	This DLL provides access to advanced core Windows components such as the Service Manager and Registry.
<i>User32.dll</i>	This DLL contains all the user-interface components, such as buttons, scroll bars, and components for controlling and responding to user actions.
<i>Gdi32.dll</i>	This DLL contains functions for displaying and manipulating graphics.
<i>Ntdll.dll</i>	This DLL is the interface to the Windows kernel. Executables generally do not import this file directly, although it is always imported indirectly by <i>Kernel32.dll</i> . If an executable imports this file, it means that the author intended to use functionality not normally available to Windows programs.

Common DLLs

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.

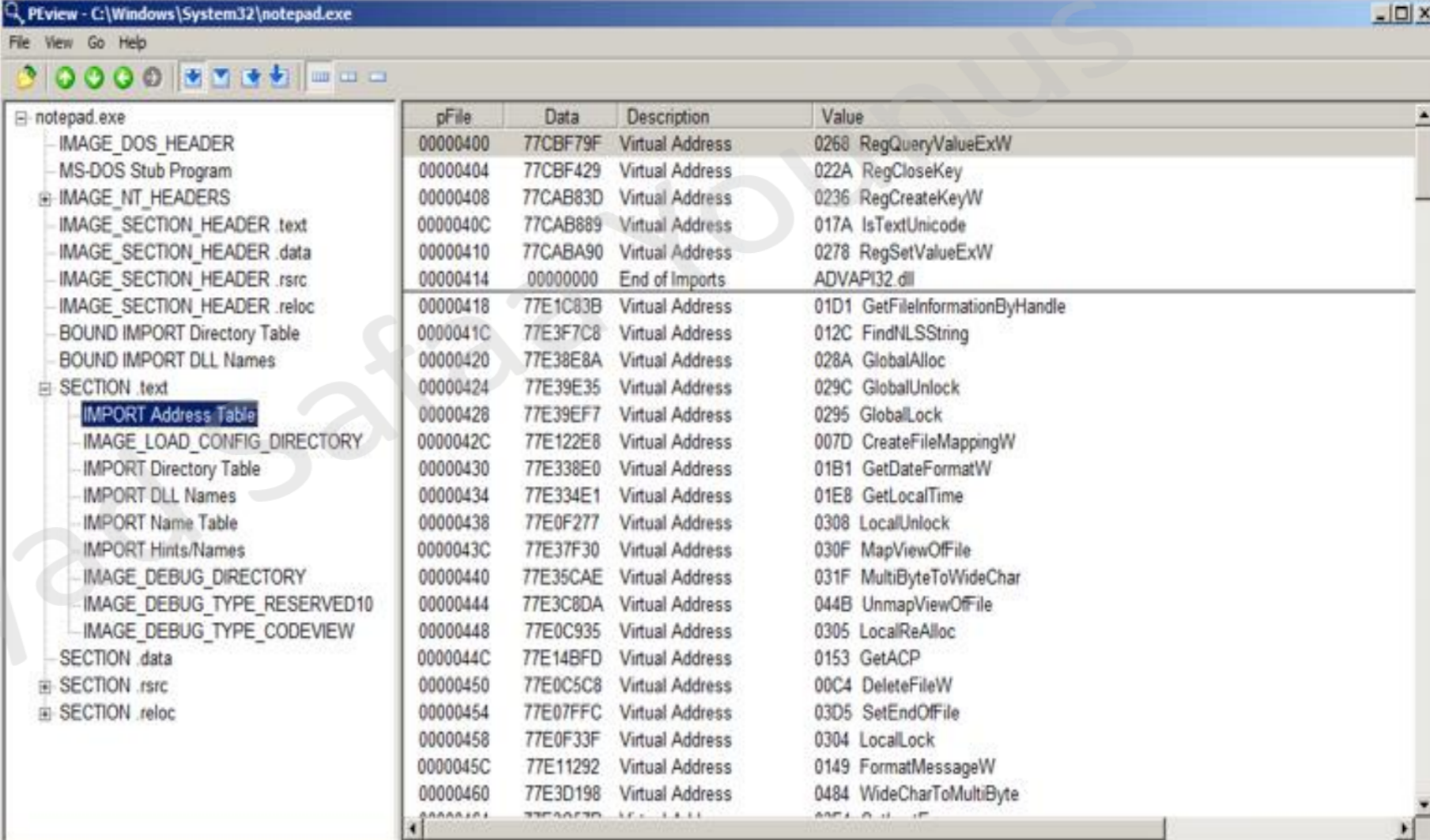
Wininet.dll This DLL contains higher-level networking functions that implement protocols such as FTP, HTTP, and NTP.

Exports

- **DLL and EXE files** can export functions that are used by other programs. While exported functions are commonly found in DLLs, they are relatively rare in executable (EXE) files. Both imported and exported functions are documented in the Portable Executable (PE) header.
- When an **EXE contains exported functions**, these exports may provide significant clues about the program's intended behavior and functionality.
- For example, an exported function named **ServiceMain** typically indicates that the program or malware is designed to run as a Windows service.
- However, **malware authors often attempt to evade analysis** by **renaming, obfuscating, or hiding exported functions**, thereby misleading analysts and complicating the reverse engineering process.

Notepad.exe

- Notepad .exe is a safe and clean Windows program
- useful to learn how PE files work.
- It shows how legitimate applications use DLLs and imported functions, and it can also be used as a practical example to study a real PE file for instance, by opening it in tools like **PEview** to observe how it calls **DLLs** such as **kernel32.dll** or **user32.dll**.



PEview - C:\Windows\System32\notepad.exe

pFile	Data	Description	Value
00000400	77CBF79F	Virtual Address	0268 RegQueryValueExW
00000404	77CBF429	Virtual Address	022A RegCloseKey
00000408	77CAB83D	Virtual Address	0236 RegCreateKeyW
0000040C	77CAB889	Virtual Address	017A IsTextUnicode
00000410	77CABA90	Virtual Address	0278 RegSetValueExW
00000414	00000000	End of Imports	ADVAPI32.dll
00000418	77E1C83B	Virtual Address	01D1 GetFileInformationByHandle
0000041C	77E37FC8	Virtual Address	012C FindNLSString
00000420	77E38E8A	Virtual Address	028A GlobalAlloc
00000424	77E39E35	Virtual Address	029C GlobalUnlock
00000428	77E39EF7	Virtual Address	0295 GlobalLock
0000042C	77E122E8	Virtual Address	007D CreateFileMappingW
00000430	77E338E0	Virtual Address	01B1 GetDateFormatW
00000434	77E334E1	Virtual Address	01E8 GetLocalTime
00000438	77E0F277	Virtual Address	0308 LocalUnlock
0000043C	77E37F30	Virtual Address	030F MapViewOfFile
00000440	77E35CAE	Virtual Address	031F MultiByteToWideChar
00000444	77E3C8DA	Virtual Address	044B UnmapViewOfFile
00000448	77E0C935	Virtual Address	0305 LocalReAlloc
0000044C	77E14BFD	Virtual Address	0153 GetACP
00000450	77E0C5C8	Virtual Address	00C4 DeleteFileW
00000454	77E07FFC	Virtual Address	03D5 SetEndOfFile
00000458	77E0F33F	Virtual Address	0304 LocalLock
0000045C	77E11292	Virtual Address	0149 FormatMessageW
00000460	77E3D198	Virtual Address	0484 WideCharToMultiByte

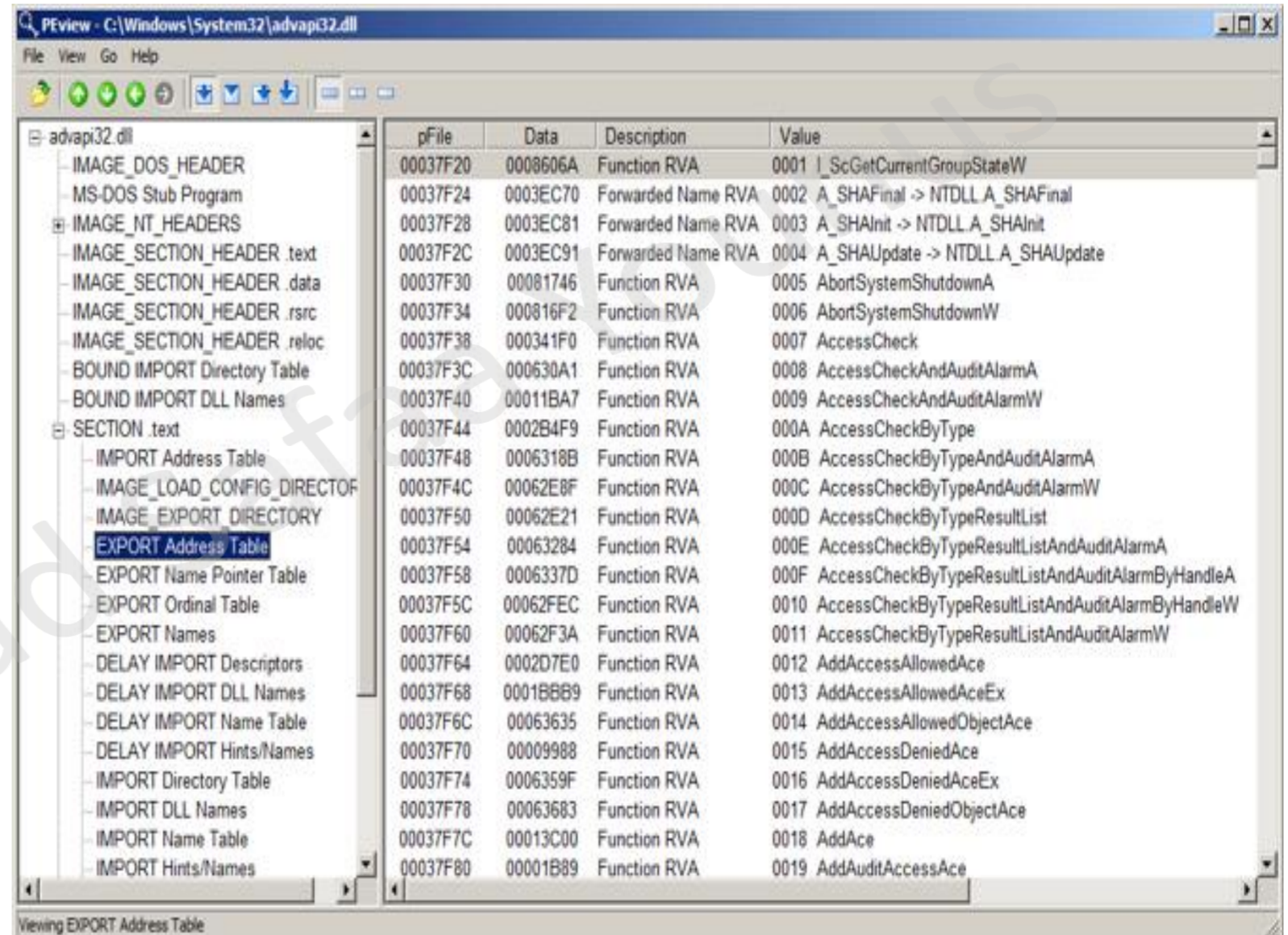
Viewing IMPORT Address Table

Advapi32.dll

Advapi32.dll is a **Windows system library** that provides functions for:

1. Security and user permissions
2. Service control
3. Registry access
4. Authentication

Malware often uses it to **elevate privileges** or **change system settings**.



pFile	Data	Description	Value
00037F20	0008606A	Function RVA	0001 ScGetCurrentGroupStateW
00037F24	0003EC70	Forwarded Name RVA	0002 A_SHAFinal -> NTDLL.A_SHAFinal
00037F28	0003EC81	Forwarded Name RVA	0003 A_SHAInit -> NTDLL.A_SHAInit
00037F2C	0003EC91	Forwarded Name RVA	0004 A_SHAUpdate -> NTDLL.A_SHAUpdate
00037F30	00081746	Function RVA	0005 AbortSystemShutdownA
00037F34	000816F2	Function RVA	0006 AbortSystemShutdownW
00037F38	000341F0	Function RVA	0007 AccessCheck
00037F3C	000630A1	Function RVA	0008 AccessCheckAndAuditAlarmA
00037F40	00011BA7	Function RVA	0009 AccessCheckAndAuditAlarmW
00037F44	0002B4F9	Function RVA	000A AccessCheckByType
00037F48	0006318B	Function RVA	000B AccessCheckByTypeAndAuditAlarmA
00037F4C	00062E8F	Function RVA	000C AccessCheckByTypeAndAuditAlarmW
00037F50	00062E21	Function RVA	000D AccessCheckByTypeResultList
00037F54	00063284	Function RVA	000E AccessCheckByTypeResultListAndAuditAlarmA
00037F58	0006337D	Function RVA	000F AccessCheckByTypeResultListAndAuditAlarmByHandleA
00037F5C	00062FEC	Function RVA	0010 AccessCheckByTypeResultListAndAuditAlarmByHandleW
00037F60	00062F3A	Function RVA	0011 AccessCheckByTypeResultListAndAuditAlarmW
00037F64	0002D7E0	Function RVA	0012 AddAccessAllowedAce
00037F68	0001B8B9	Function RVA	0013 AddAccessAllowedAceEx
00037F6C	00063635	Function RVA	0014 AddAccessAllowedObjectAce
00037F70	00009988	Function RVA	0015 AddAccessDeniedAce
00037F74	0006359F	Function RVA	0016 AddAccessDeniedAceEx
00037F78	00063683	Function RVA	0017 AddAccessDeniedObjectAce
00037F7C	00013C00	Function RVA	0018 AddAce
00037F80	00001B89	Function RVA	0019 AddAuditAccessAce

Example of Malware

- A **keylogger** is a type of malicious software (malware) designed to secretly record the keys pressed on a keyboard by the user. Its main goal is to steal sensitive information such as usernames, passwords, credit card numbers, and private messages without the user's knowledge.
- Keyloggers often run in the background and use Windows system libraries to monitor keyboard and mouse activities, then store or send the collected data to the attacker.
- **Keylogger Analysis**
 - A keylogger typically imports **User32.dll** and makes use of several Windows API functions related to input handling.
 - The function **SetWindowsHookEx** is commonly used in spyware and keyloggers, as it allows the program to install a hook procedure that intercepts keyboard and mouse events. This is one of the most popular techniques for capturing keystrokes.
 - The malware may export functions such as **LowLevelKeyboardProc** and **LowLevelMouseProc**, which are used as callback procedures to capture keyboard and mouse input and forward the collected data to another component or remote destination.
 - Additionally, the function **RegisterHotKey** can be used to define a specific key combination (for example, **Ctrl + Shift + P**) that triggers a particular action, such as harvesting, saving, or transmitting the logged keystroke data.

Example of Malware

- A **Packed Program** : **A Dead End**
 - A very short import list and no readable strings usually indicate the file is packed/obfuscated.
 - Very few functions, Normal programs (even “Hello World”) import more functions.
 - Because the sample is packed, basic static analysis won’t reveal much.

Kernel32.dll	User32.dll
--------------	------------

GetModuleHandleA	MessageBoxA
------------------	-------------

LoadLibraryA	
--------------	--

GetProcAddress	
----------------	--

ExitProcess	
-------------	--

VirtualAlloc	
--------------	--

VirtualFree	
-------------	--

The PE File Headers and Sections

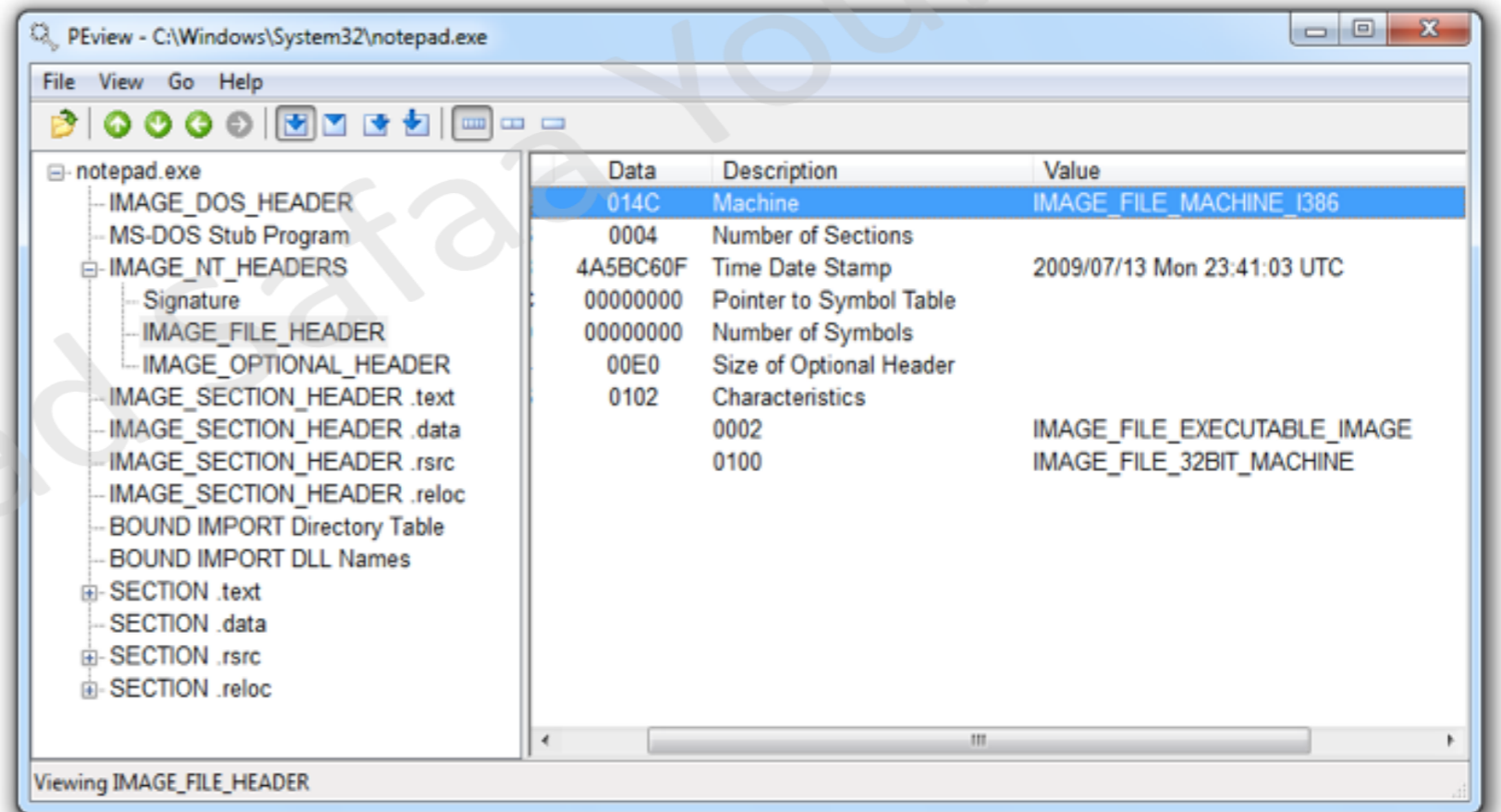
Common PE Sections

- **.text:** Contains the executable code of the program. This is where the main instructions that the CPU executes are stored.
- **.rdata:** Contains read-only data such as **imports and exports information**. Holds data that is globally accessible but cannot be modified during execution.
- **.data:** Stores global and static variables that are accessed and modified throughout the program while it is running.
- **.rsrc:** Stores resources needed by the executable, such as **strings, icons, images, dialogs, and menus**.

Examining PE Files with PView

- The **PE header** stores key metadata about the executable within its header. **PView** tool can be used to inspect these headers.

- **Key part of Header:**
- **IMAGE_DOS_HEADER:**
Historical, not useful.
- **IMAGE_NT_HEADERS:**
Signature (always the same).
- **IMAGE_FILE_HEADER:**
Includes compile time (can be forged).
- **IMAGE_OPTIONAL_HEADER**
: Defines subsystem
(Console or GUI).



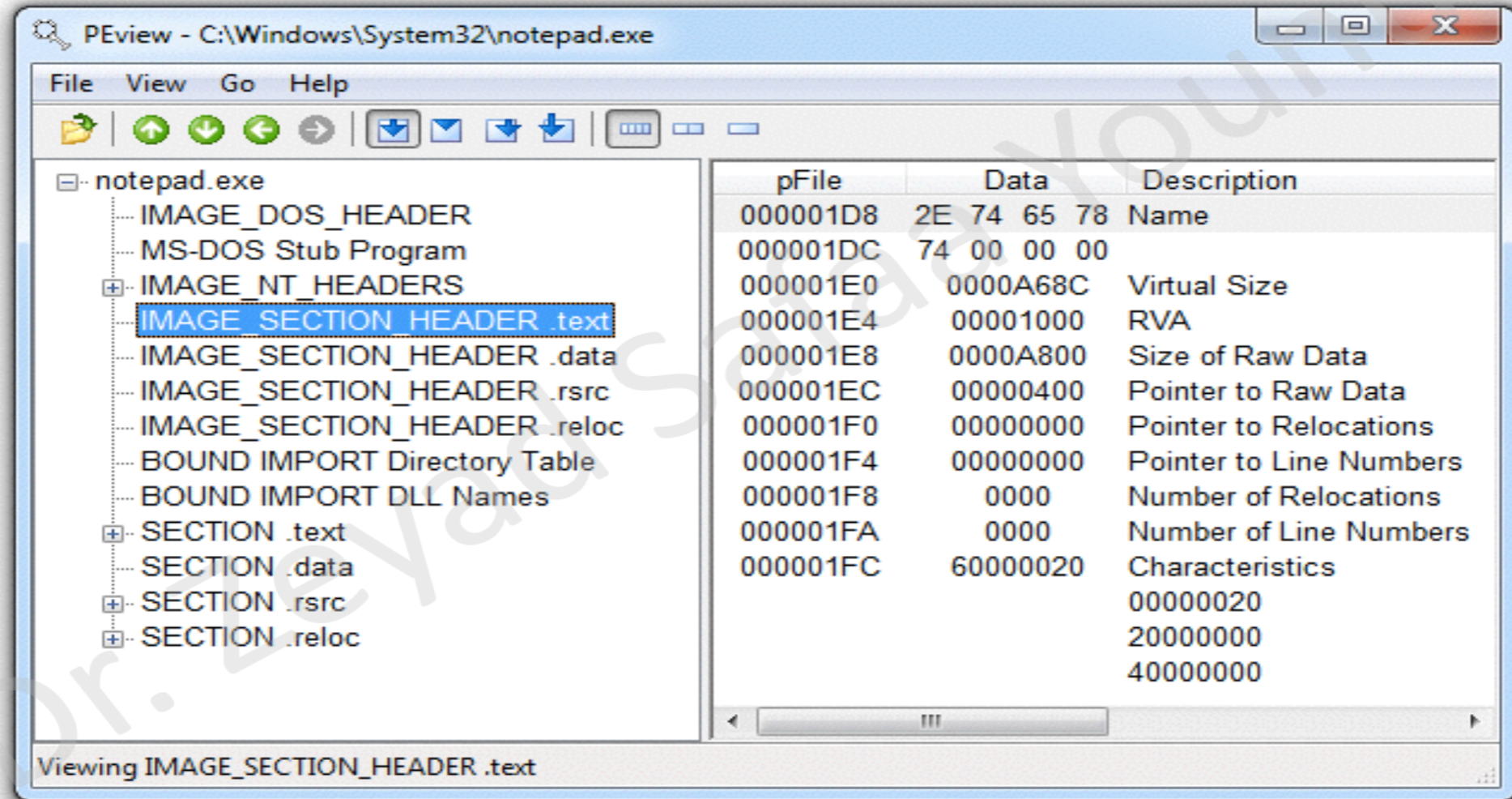
Time Date Stamp

- Shows when this executable was compiled
- Older programs are more likely to be known to antivirus software
- But sometimes the date is wrong
 - All Delphi programs show June 19, 1992
 - Date can also be faked

IMAGE_SECTION_HEADER

- **Virtual Size** – size in memory (**RAM**)
- **Size of Raw Data** – size on **disk**
 - For **.text** section, normally equal, or nearly equal size.
 - Big difference: may indicate packed code.
 - Unusual section names may be suspicious
- Packed executables show Virtual Size much larger than Size of Raw Data for **.text** section
- Notes:
 - *Normal program*: similar sizes, standard section names.
 - *Packed malware*: unusual names, Raw Data = 0, large Virtual Size.

.text file Not Packed



Information for packedprogram.exe

Name	Virtual size	Size of raw data
.text	A000	0000
.data	3000	0000
.rdata	4000	0000
.rsrc	19000	3400
Dijfpds	20000	0000
.sdfuok	34000	3313F
Kijijl	1000	0200

References

- Sikorski, M., & Honig, A. (2012). *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software*. No Starch Press.
- Bowne, S. (n.d.). *CNIT 126 Ch 0: Malware Analysis Primer & 1: Basic Static Techniques* [PowerPoint slides]. SlideShare.
<https://www.slideshare.net/slideshow/cnit-126-ch-0-malware-analysis-primer-1-basic-static-techniques/71040577>
- Bowne, S. (n.d.). *CNIT 126: Ch 0-1 Basic Static Techniques (Part 1)* [Video]. YouTube. <https://www.youtube.com/watch?v=4ZY4LC2XpFM>