Chapter 4
Basic Static Analysis

Mohd Zaki Mas'ud

5 Nov 2020

Topic

- Introduction
- Antivirus Scanning
- Hashing Fingerprint
- Finding String
- Pack and Obfuscated
- PE File Format
- Link Libraries and Function

Introduction

Basic Static Analysis

- *Static analysis* describes the process of analysing the code or structure of a program to determine its function.
- The program itself is not run at this time.
- Among The basic techniques are:
  - Using antivirus tools to confirm maliciousness
  - Using hashes to identify malware
  - Gleaning information from a file's strings, functions, and headers
- Each technique can provide different information, and the ones you use depend on your goals.

MALWARE SAMPLE

Malware sources

- Hybrid Analysis (https://www. Hybrid-analysis.com/)
- kernelMode.info(https://www. kernelMode.info/forum/viewforum.php?f=16)
- virusBay(https://beta.virusbay.io)
- Contagio malware dump (https://contagiodump.blogspot.com)
- AVCaesar(https://avcaesar.malware.1u/)
- Malwr(https://malwr.com)
- VirusSHare(https://virusshare.com)
- theZoo (http://thezoo.morirt.com/)

Antivirus Scanning

- First step to analyze a malware is by running it through a multiple antivirus program.
- It safe time especially if the malware has been identified.
- Still if it is already known, malware author will tries to modified it or making a new malware variant.
- Thus can evade the virus signature for previously known malware.

Only a First Step

- Malware can easily change its signature and fool the antivirus
- VirusTotal is convenient, but using it may alert attackers that they've been caught
  - http://www.virustotal.com

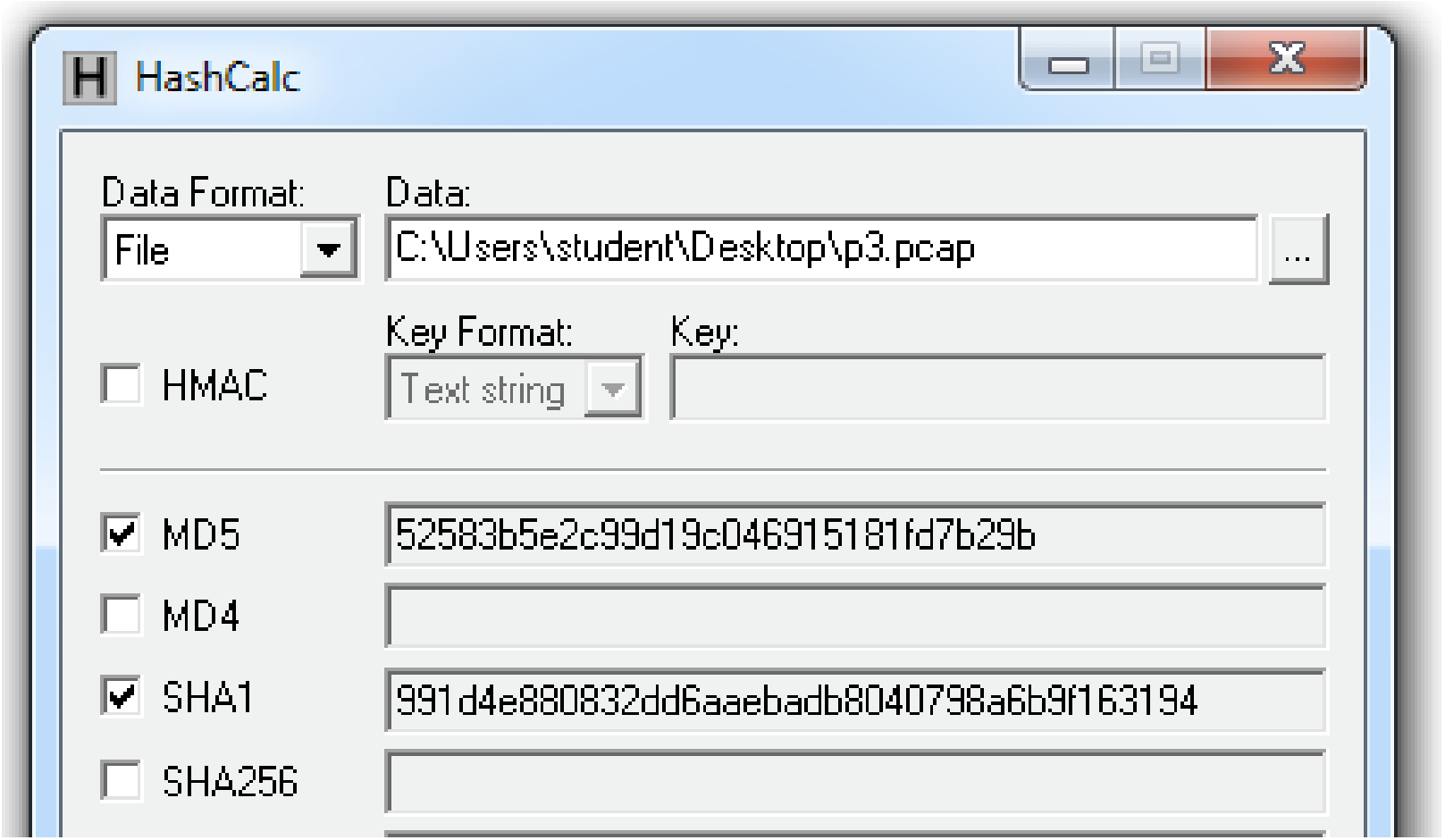
Hashing

A fingerprint for malware

Hashes

- MD5 or SHA-1
- Condenses a file of any size down to a fixed-length fingerprint
- Uniquely identifies a file well in practice
  - There are MD5 collisions but they are not common
  - Collision: two different files with the same hash

HashCalc


HashCalc
Data Format:
File
Data:
C:\Users\student\Desktop\p3.pcap
...
HMAC
Key Format:
Text string
Key:
MD5
52583b5e2c99d19c046915181fd7b29b
MD4
SHA1
991d4e880832dd6aaebadb8040798a6b9f163194
SHA256

Hash Uses

- Label a malware file
- Share the hash with other analysts to identify malware
- Search the hash online to see if someone else has already identified the file

Finding Strings

Strings

- Any sequence of printable characters is a **string**
- Strings are terminated by a **null** (0x00)
- ASCII characters are 8 bits long
  - Now called ANSI
- Unicode characters are 16 bits long
  - Microsoft calls them "wide characters"

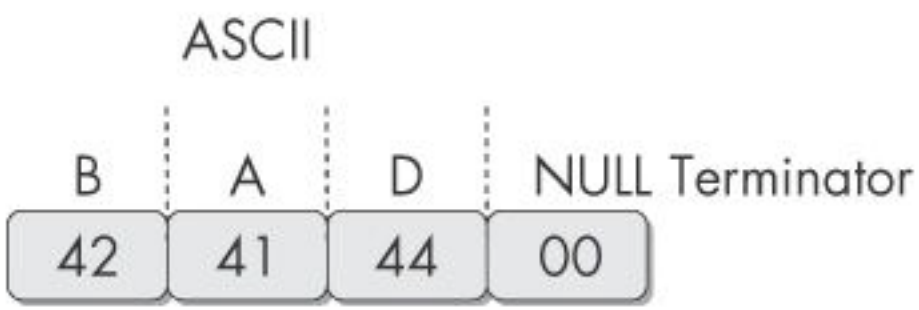


Figure 2-2. ASCII representation of the string BAD

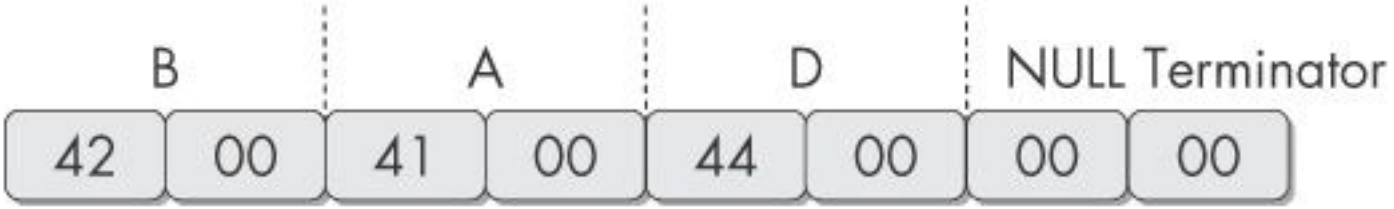


Figure 2-3. Unicode representation of the string BAD

The strings Command

- Native in Linux, also available for Windows
- Finds all strings in a file 3 or more characters long
- Not all is string sometimes it represent memory address, CPU instruction or data used by program
- Invalid string can clearly shown.

The strings Command

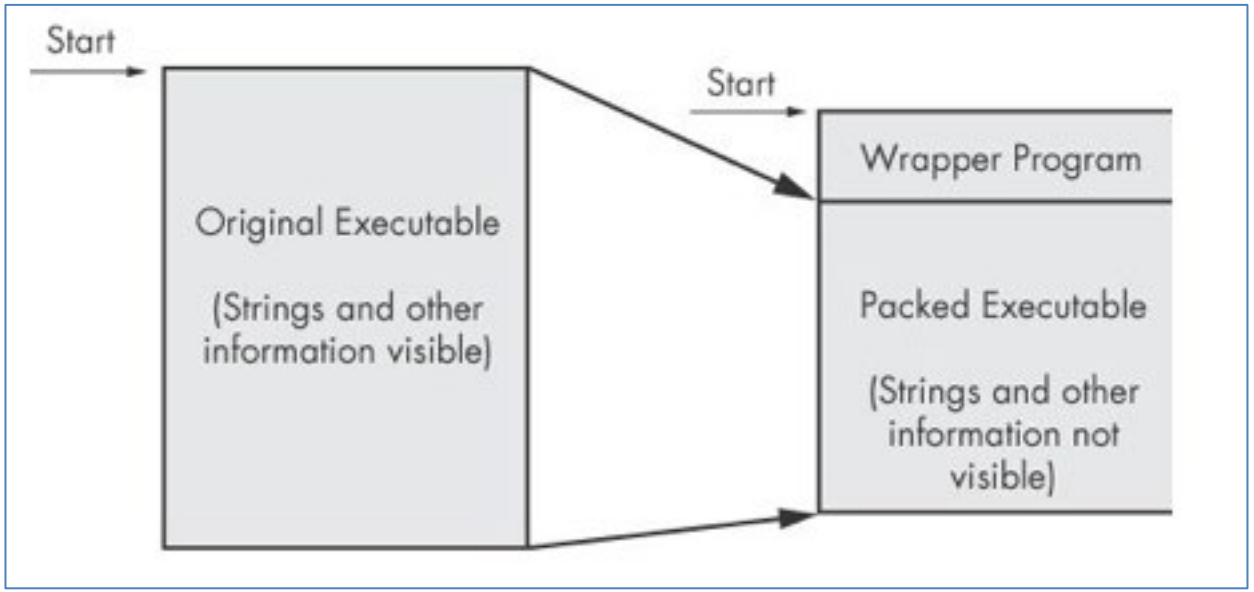
- Bold items can be ignored
- GetLayout and SetLayout are Windows functions
- GDI32.DLL is a Dynamic Link Library

```
C:>strings bp6.ex_
VP3
VW3
t$@
D$4
99.124.22.1 4
e-@
GetLayout 1
GDI32.DLL 3
SetLayout 2
M}C
Mail system DLL is invalid.!Send Mail failed to
send message. 5
```

Packed and Obfuscated Malware

Packing Files

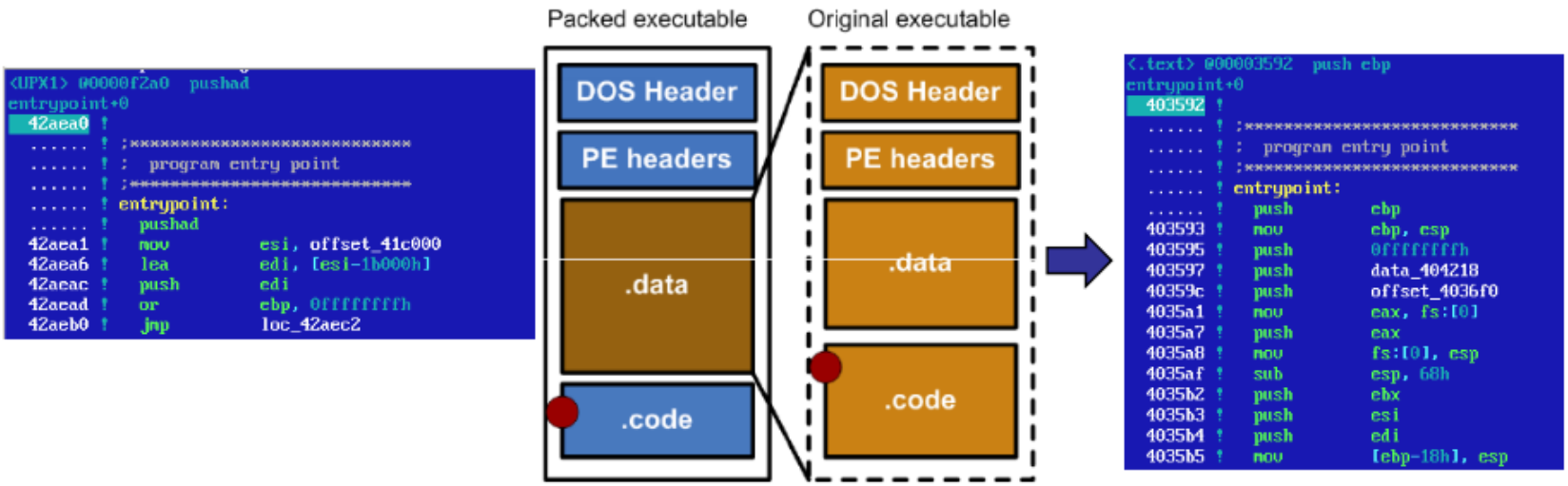
- The code is compressed, like a Zip file
- This makes the strings and instructions unreadable
- All you'll see is the **wrapper** – small code that unpacks the file when it is run



- Legitimate program almost always include many Strings compare to packed or obfuscated malware.
- At least function LoadLibrary and GetProcAddress are visible

How typical packer runtime works

1. Original data is located somewhere in the packer code data section
2. Original data is uncompressed to the originally linked location
3. Control is transferred to original code entry point (OEP)



Detecting Packers with PEiD

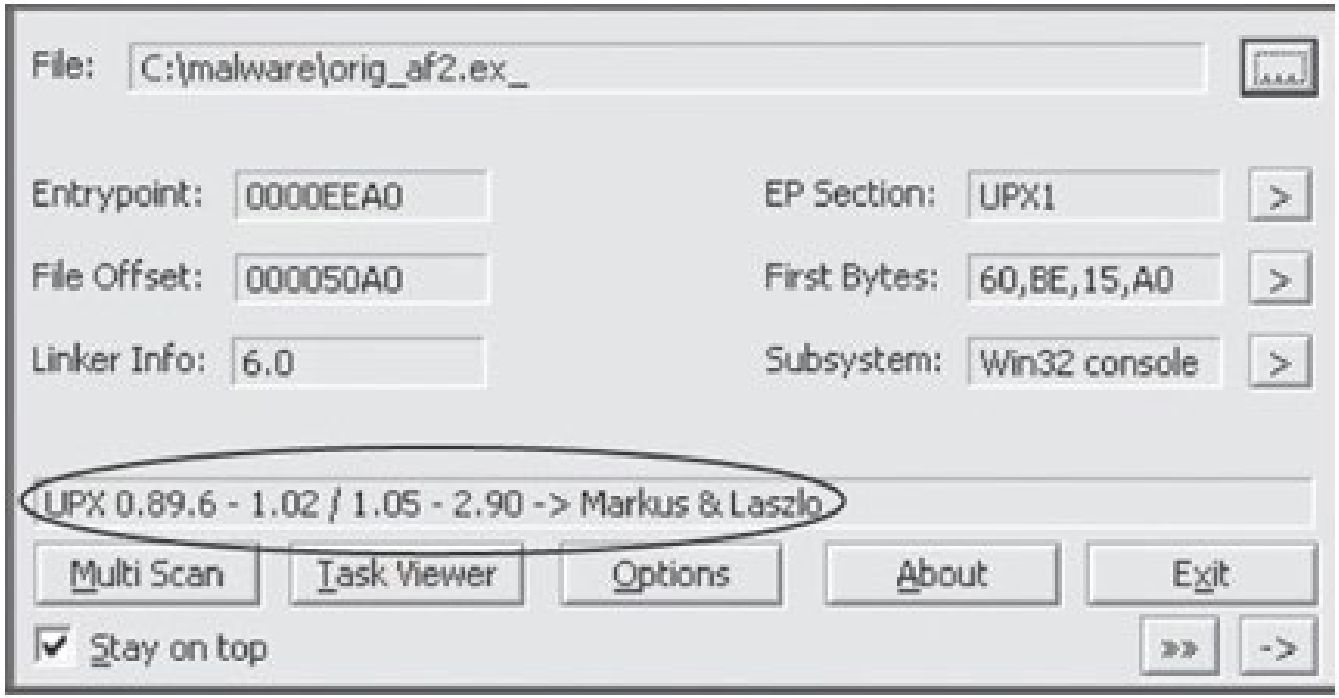
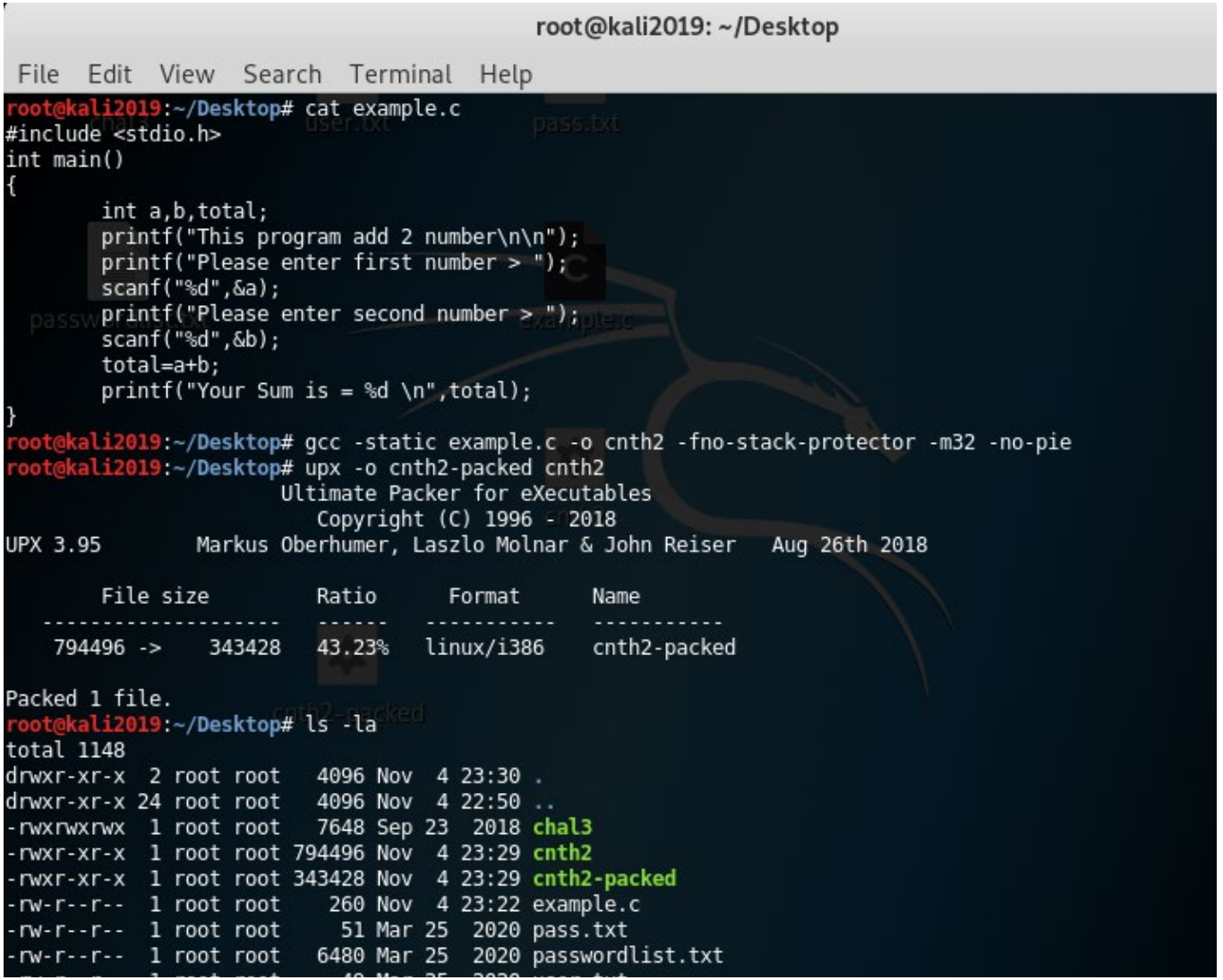
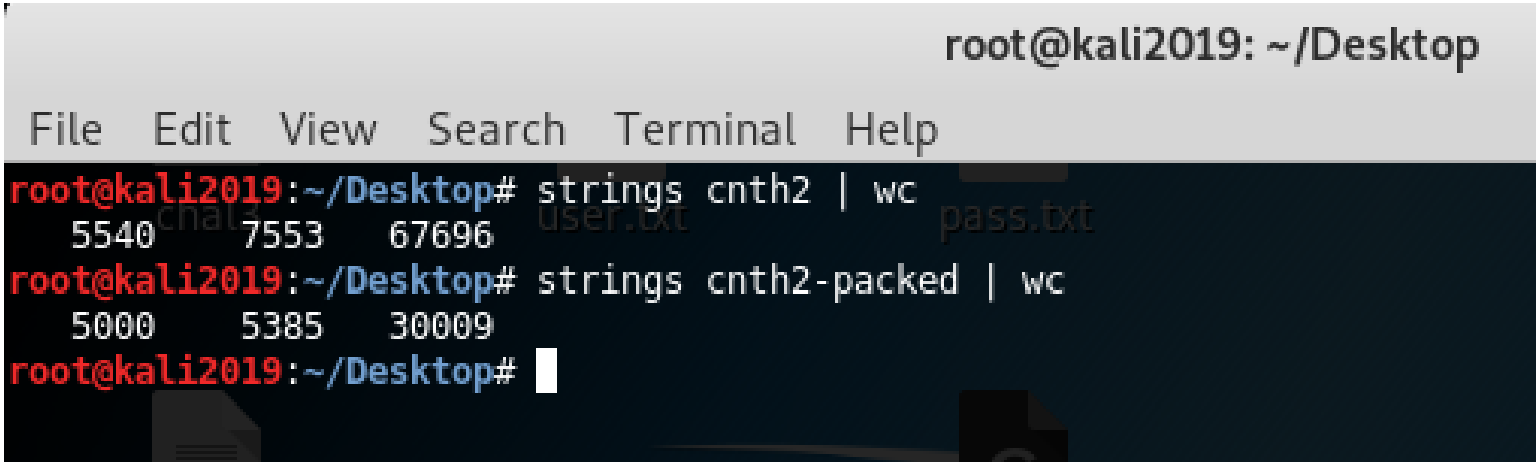


Figure 2-5. The PEiD program

Demo: UPX


root@kali2019: ~/Desktop
File Edit View Search Terminal Help
root@kali2019:~/Desktop# cat example.c
#include <stdio.h>
int main()
{
int a,b,total;
printf("This program add 2 number\n\n");
printf("Please enter first number > ");
scanf("%d",&a);
printf("Please enter second number > ");
scanf("%d",&b);
total=a+b;
printf("Your Sum is = %d \n",total);
}
root@kali2019:~/Desktop# gcc -static example.c -o cnth2 -fno-stack-protector -m32 -no-pie
root@kali2019:~/Desktop# upx -o cnth2-packed cnth2
Ultimate Packer for eXecutables
Copyright (C) 1996 - 2018
UPX 3.95 Markus Oberhumer, Laszlo Molnar & John Reiser Aug 26th 2018
File size Ratio Format Name
794496 -> 343428 43.23% linux/i386 cnth2-packed
Packed 1 file.
root@kali2019:~/Desktop# ls -la
total 1148
drwxr-xr-x 2 root root 4096 Nov 4 23:30 .
drwxr-xr-x 24 root root 4096 Nov 4 22:50 ..
-rwxrwxrwx 1 root root 7648 Sep 23 2018 chal3
-rwxr-xr-x 1 root root 794496 Nov 4 23:29 cnth2
-rwxr-xr-x 1 root root 343428 Nov 4 23:29 cnth2-packed
-rw-r--r-- 1 root root 260 Nov 4 23:22 example.c
-rw-r--r-- 1 root root 51 Mar 25 2020 pass.txt
-rw-r--r-- 1 root root 6480 Mar 25 2020 passwordlist.txt

Packing Obfuscates Strings



NOTE

Many PEiD plug-ins will run the malware executable without warning! (See Chapter 3 to learn how to set up a safe environment for running malware.) Also, like all programs, especially those used for malware analysis, PEiD can be subject to vulnerabilities. For example, PEiD version 0.92 contained a buffer overflow that allowed an attacker to execute arbitrary code. This would have allowed a clever malware writer to write a program to exploit the malware analyst's machine. Be sure to use the latest version of PEiD.

Portable Executable File Format

EXE Files

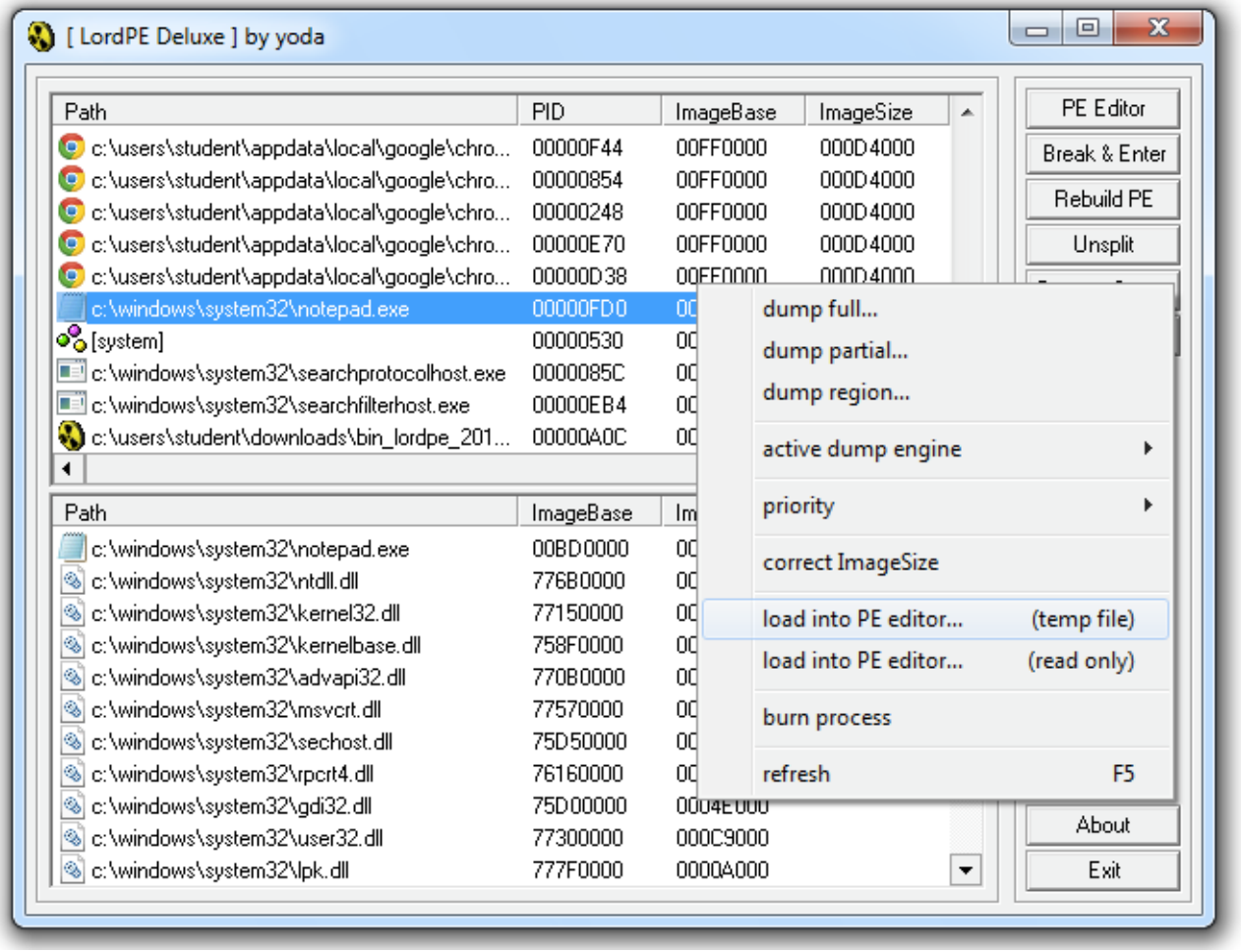
PE Files

- Used by Windows executable files, object code, and DLLs
- A data structure that contains the information necessary for Windows to load the file
- Almost every file executed on Windows is in PE format

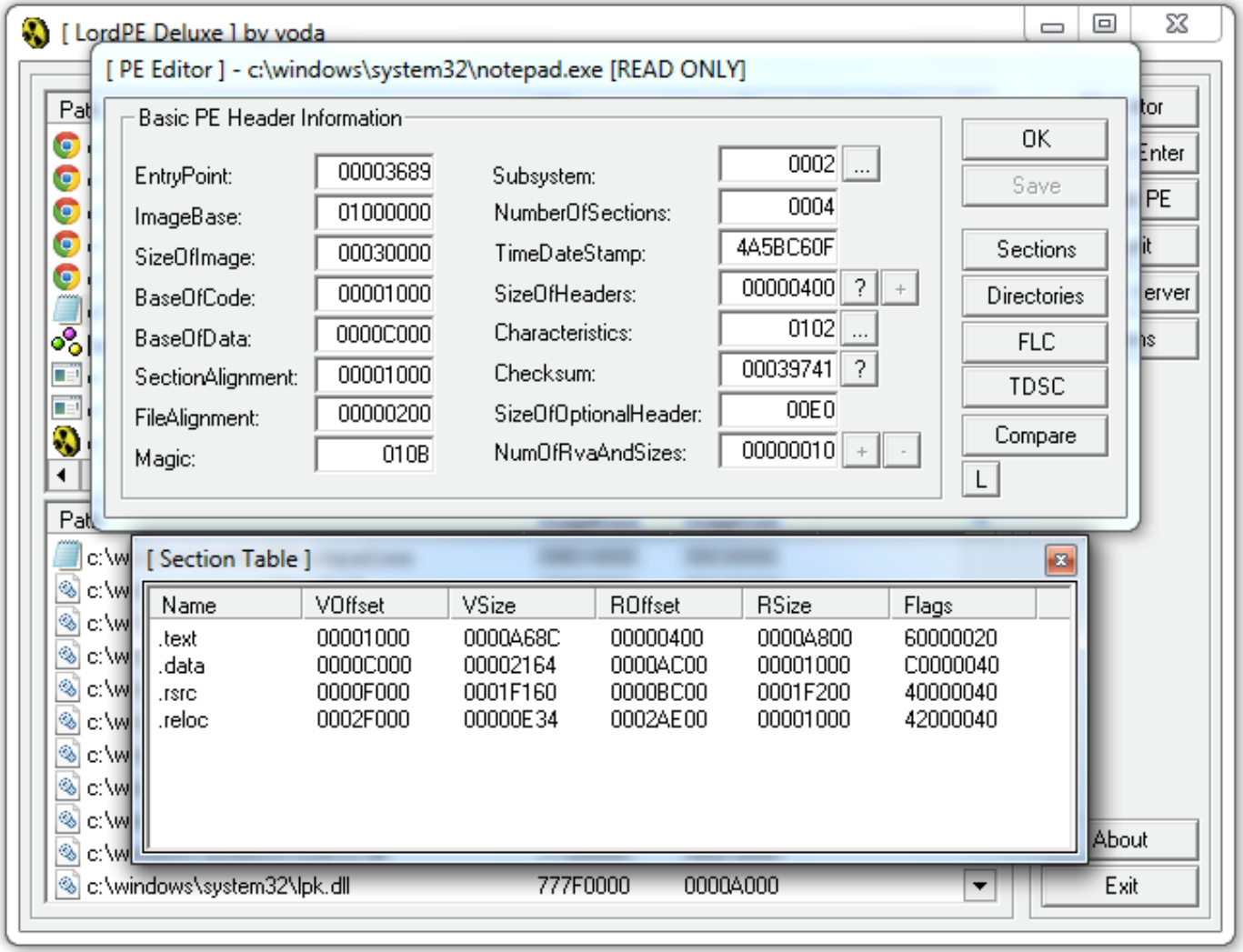
PE Header

- Information about the code
- Type of application
- Required library functions
- Space requirements

LordPE Demo


[LordPE Deluxe] by yoda
Path
PID
ImageBase
ImageSize
c:\users\student\appdata\local\google\chro...
00000F44
00FF0000
000D4000
c:\users\student\appdata\local\google\chro...
00000854
00FF0000
000D4000
c:\users\student\appdata\local\google\chro...
00000248
00FF0000
000D4000
c:\users\student\appdata\local\google\chro...
00000E70
00FF0000
000D4000
c:\users\student\appdata\local\google\chro...
00000D38
c:\windows\system32\notepad.exe
00000FD0
[system]
00000530
c:\windows\system32\searchprotocolhost.exe
0000085C
c:\windows\system32\searchfilterhost.exe
00000EB4
c:\users\student\downloads\bin_lordpe_201...
00000A0C
Path
ImageBase
c:\windows\system32\notepad.exe
00BD0000
c:\windows\system32\ntdll.dll
776B0000
c:\windows\system32\kernel32.dll
77150000
c:\windows\system32\kernelbase.dll
758F0000
c:\windows\system32\advapi32.dll
770B0000
c:\windows\system32\msvcrt.dll
77570000
c:\windows\system32\sechost.dll
75D50000
c:\windows\system32\rpcrt4.dll
76160000
c:\windows\system32\gdi32.dll
75D00000
c:\windows\system32\user32.dll
77300000
000C9000
c:\windows\system32\lpk.dll
777F0000
0000A000
dump full...
dump partial...
dump region...
active dump engine
priority
correct ImageSize
load into PE editor... (temp file)
load into PE editor... (read only)
burn process
refresh F5
PE Editor
Break & Enter
Rebuild PE
Unsplit
About
Exit

Main Sections



There are a lot more sections

- But the main ones are enough for now

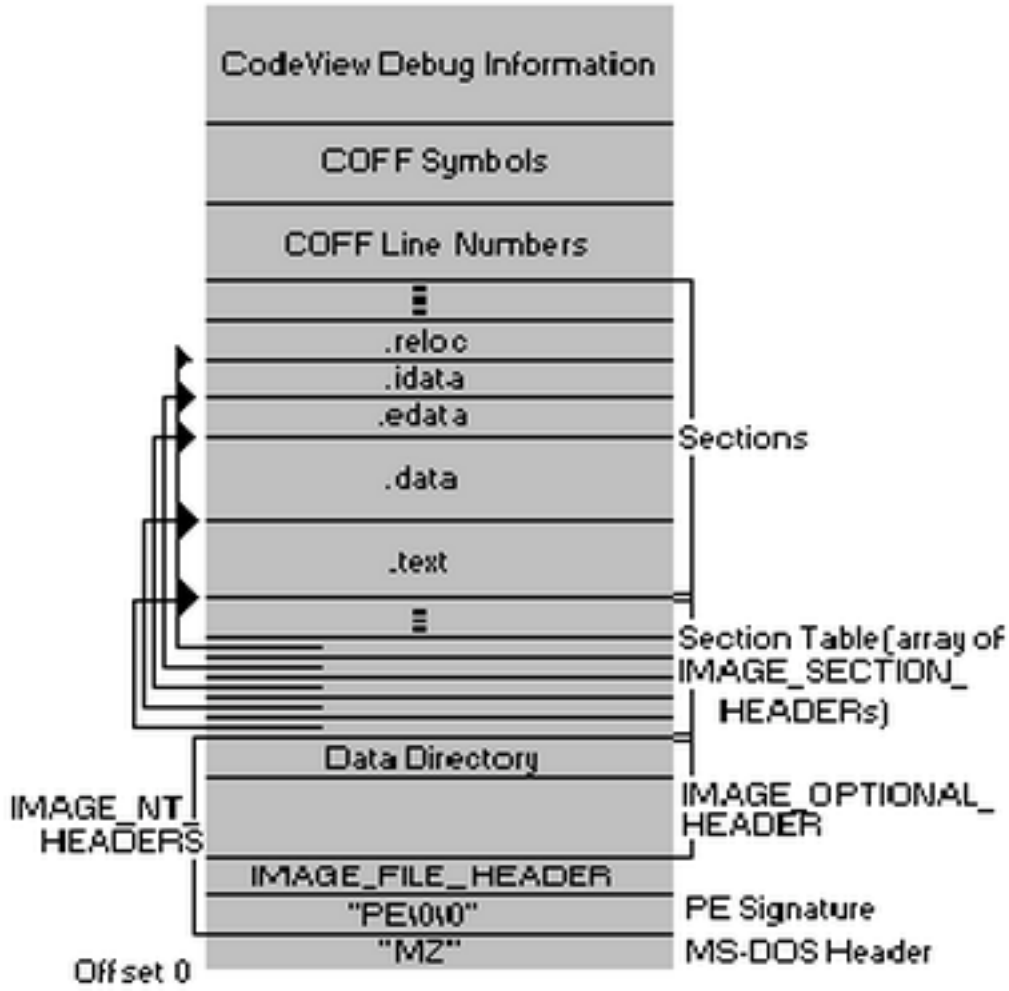


Figure 1. The PE file format

Linked Libraries and Functions

Imports

- Functions used by a program that are stored in a different program, such as library
- Connected to the main EXE by **Linking**
- Can be linked three ways
  - **Statically**
  - At **Runtime**
  - **Dynamically**

Static Linking

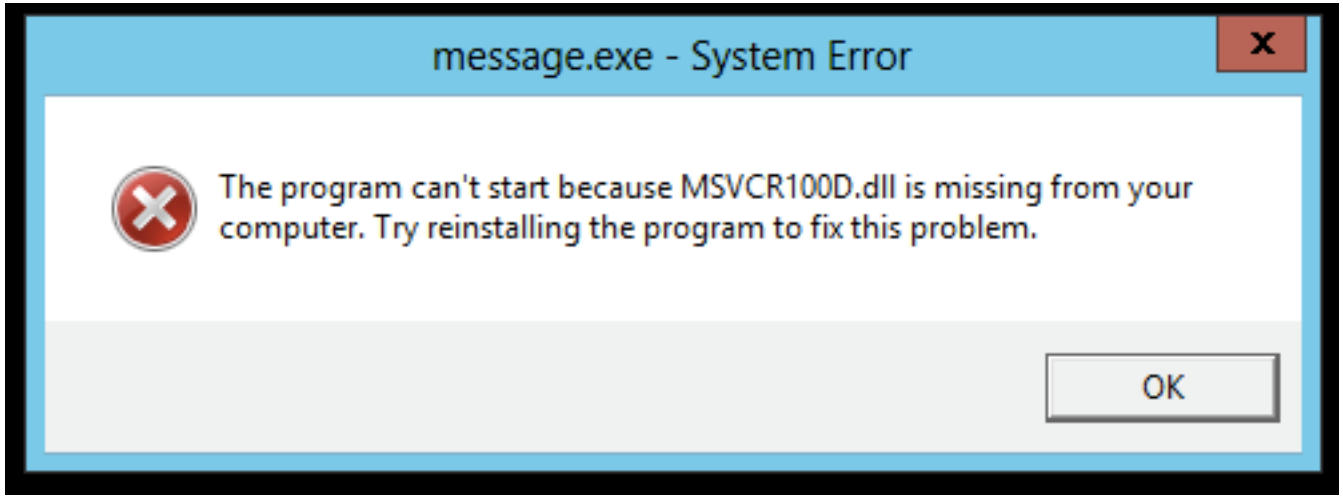
- Rarely used for Windows executables
- Common in Unix and Linux
- All code from the library is copied into the executable
- Makes executable large in size

Runtime Linking

- Unpopular in friendly programs
- Common in malware, especially packed or obfuscated malware
- Connect to libraries only when needed, not when the program starts
- Most commonly done with the **LoadLibrary** and **GetProcAddress** functions

Dynamic Linking

- Most common method
- Host OS searches for necessary libraries when the program is loaded



Clues in Libraries

- The PE header lists every library and function that will be loaded
- Their names can reveal what the program does
- **URLDownloadToFile** indicates that the program downloads something

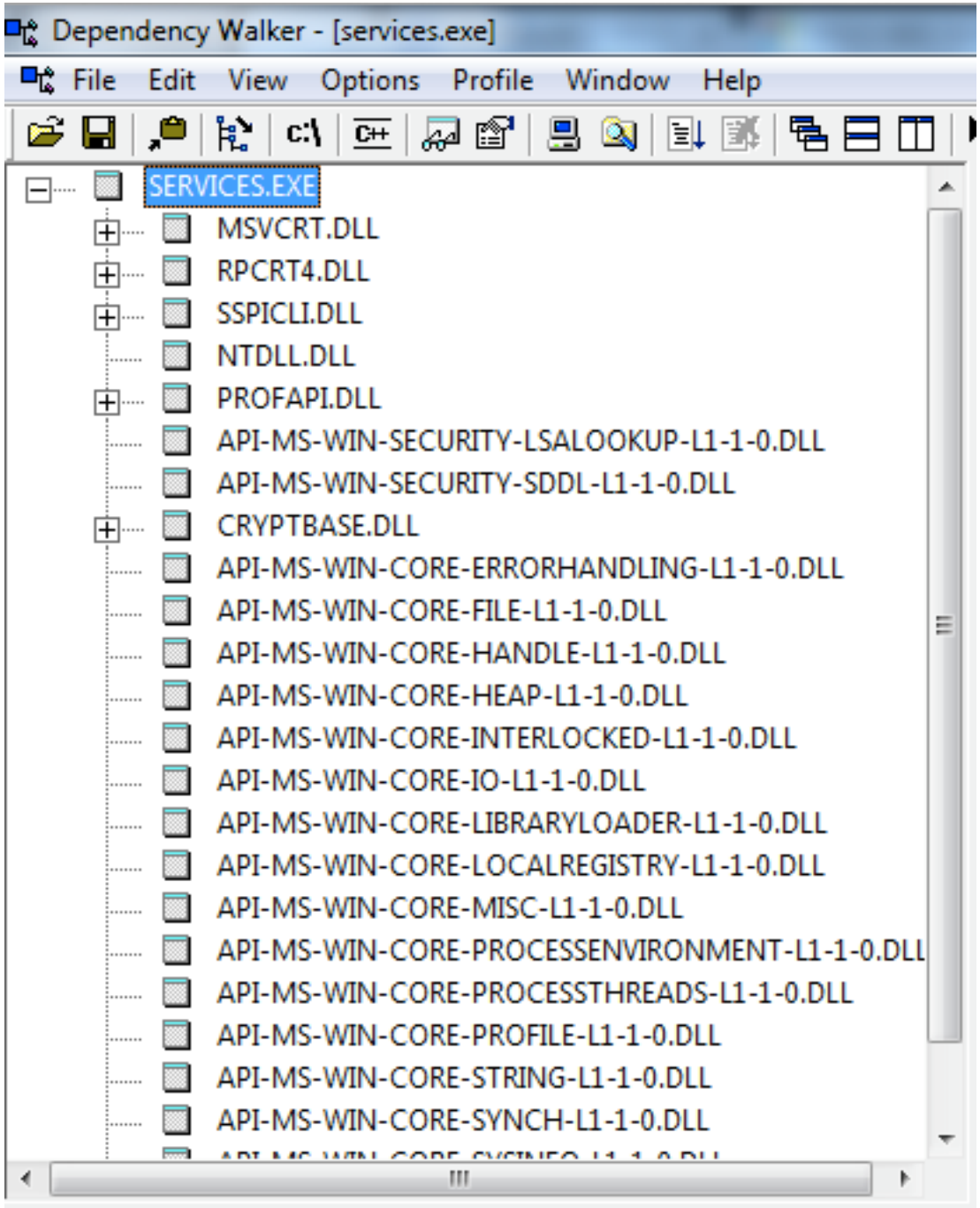
DLL	Description
Kernel32.dll	This is a very common DLL that contains core functionality, such as access and manipulation of memory, files, and hardware.
Advapi32.dll	This DLL provides access to advanced core Windows components such as the Service Manager and Registry.
User32.dll	This DLL contains all 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.

Dependency Walker

Shows Dynamically Linked Functions

- Normal programs have a lot of DLLs
- Malware often has very few DLLs

Services.exe



Services.ex_ (malware)

Imports & Exports in Dependency Walker

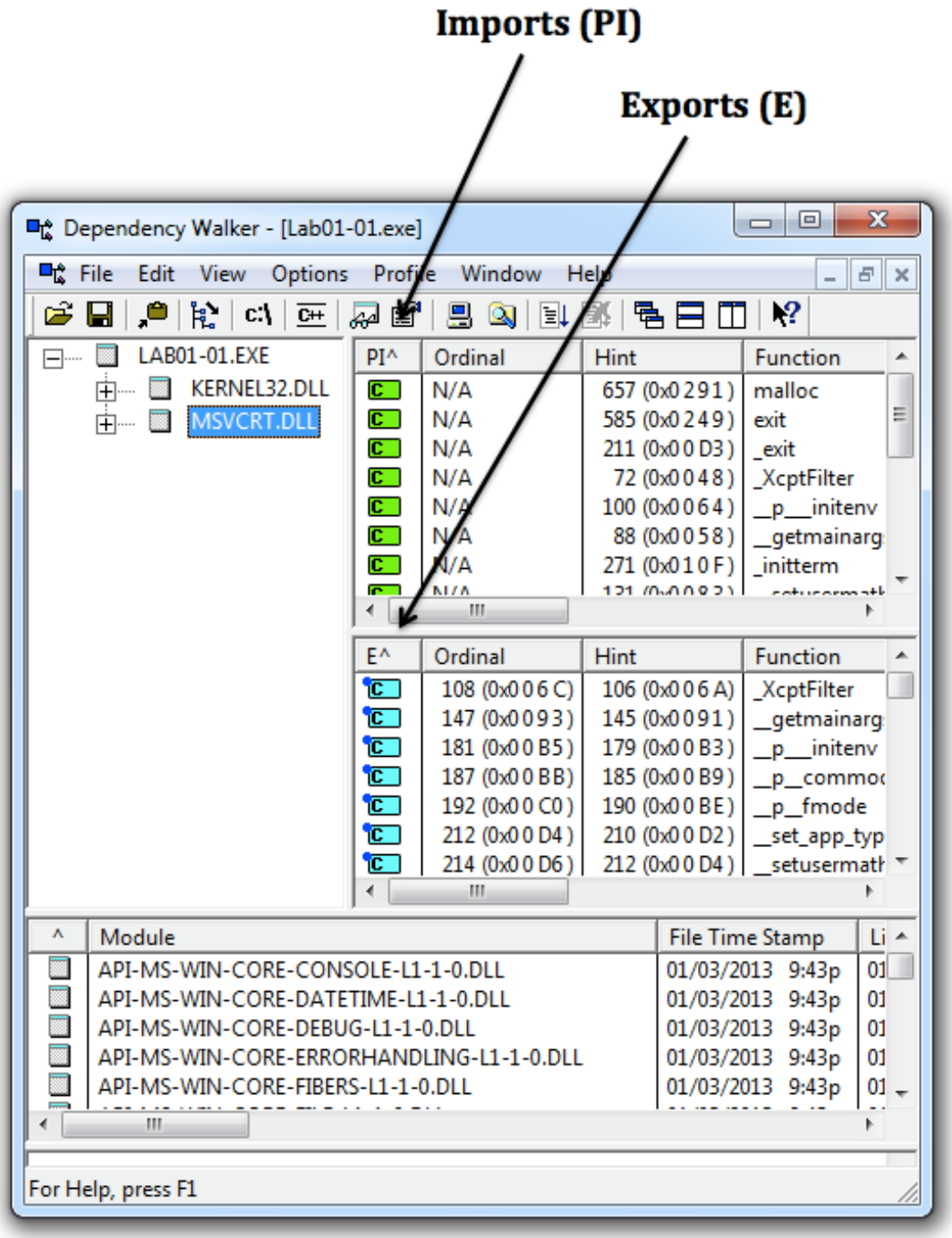


Table 2-1. Common DLLs

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.

Exports

- DLLs **export** functions
- EXEs **import** functions
- Both exports and imports are listed in the PE header

Example: Keylogger

- Imports User32.dll and uses the function **SetWindowsHookEx** which is a popular way keyloggers receive keyboard inputs
- It exports **LowLevelKeyboardProc** and **LowLevelMouseProc** to send the data elsewhere
- It uses **RegisterHotKey** to define a special keystroke like Ctrl+Shift+P to harvest the collected data

Library Associated With a Key Logger

Kernel32.dll	User32.dll	User32.dll (continued)
CreateDirectoryW	BeginDeferWindowPos	**ShowWindow**
CreateFileW	CallNextHookEx	ToUnicodeEx
CreateThread	CreateDialogParamW	TrackPopupMenu
DeleteFileW	CreateWindowExW	TrackPopupMenuEx
ExitProcess	DefWindowProcW	TranslateMessage
FindClose	DialogBoxParamW	UnhookWindowsHookEx
FindFirstFileW	EndDialog	UnregisterClassW
FindNextFileW	GetMessageW	UnregisterHotKey
GetCommandLineW	GetSystemMetrics	
GetCurrentProcess	GetWindowLongW	**GDI32.dll**
GetCurrentThread	GetWindowRect	GetStockObject
GetFileSize	GetWindowTextW	SetBkMode
GetModuleHandleW	InvalidateRect	SetTextColor
GetProcessHeap	IsDlgButtonChecked	
GetShortPathNameW	IsWindowEnabled	**Shell32.dll**
HeapAlloc	LoadCursorW	CommandLineToArgvW
HeapFree	LoadIconW	SHChangeNotify
IsDebuggerPresent	LoadMenuW	SHGetFolderPathW
MapViewOfFile	MapVirtualKeyW	ShellExecuteExW
OpenProcess	MapWindowPoints	ShellExecuteW
ReadFile	MessageBoxW	
SetFilePointer	**RegisterClassExW**	**Advapi32.dll**
WriteFile	**RegisterHotKey**	RegCloseKey
	SendMessageA	RegDeleteValueW
	SetClipboardData	RegOpenCurrentUser
	SetDlgItemTextW	RegOpenKeyExW
	SetWindowTextW	RegQueryValueExW
	SetWindowsHookExW	RegSetValueExW

Ex: A Packed Program

- Very few functions
- All you see is the unpacker

Table 2-3. DLLs and Functions Imported from PackedProgram.exe

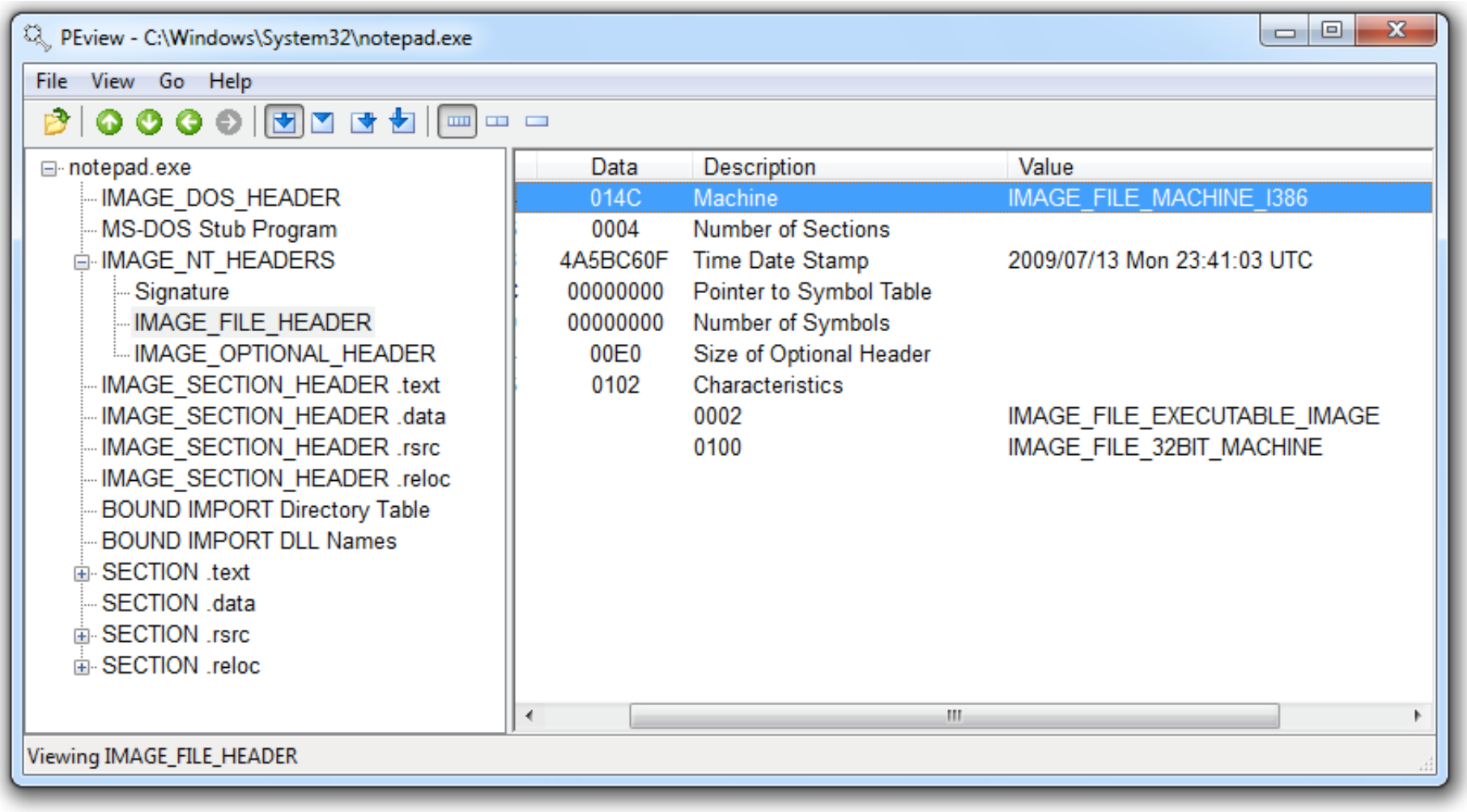
Kernel32.dll	User32.dll
GetModuleHandleA	MessageBoxA
LoadLibraryA	
GetProcAddress	
ExitProcess	
VirtualAlloc	
VirtualFree	

The PE File Headers and Sections

Important PE Sections

- **.text** -- instructions for the CPU to execute
- **.rdata** -- imports & exports
- **.data** – global data
- **.rsrc** – strings, icons, images, menus

PEView


PEview - C:\Windows\System32\notepad.exe
File View Go Help
notepad.exe
IMAGE_DOS_HEADER
MS-DOS Stub Program
IMAGE_NT_HEADERS
Signature
IMAGE_FILE_HEADER
IMAGE_OPTIONAL_HEADER
IMAGE_SECTION_HEADER .text
IMAGE_SECTION_HEADER .data
IMAGE_SECTION_HEADER .rsrc
IMAGE_SECTION_HEADER .reloc
BOUND IMPORT Directory Table
BOUND IMPORT DLL Names
SECTION .text
SECTION .data
SECTION .rsrc
SECTION .reloc
Data Description Value
014C Machine IMAGE_FILE_MACHINE_I386
0004 Number of Sections
4A5BC60F Time Date Stamp 2009/07/13 Mon 23:41:03 UTC
00000000 Pointer to Symbol Table
00000000 Number of Symbols
00E0 Size of Optional Header
0102 Characteristics
0002 IMAGE_FILE_EXECUTABLE_IMAGE
0100 IMAGE_FILE_32BIT_MACHINE
Viewing IMAGE_FILE_HEADER

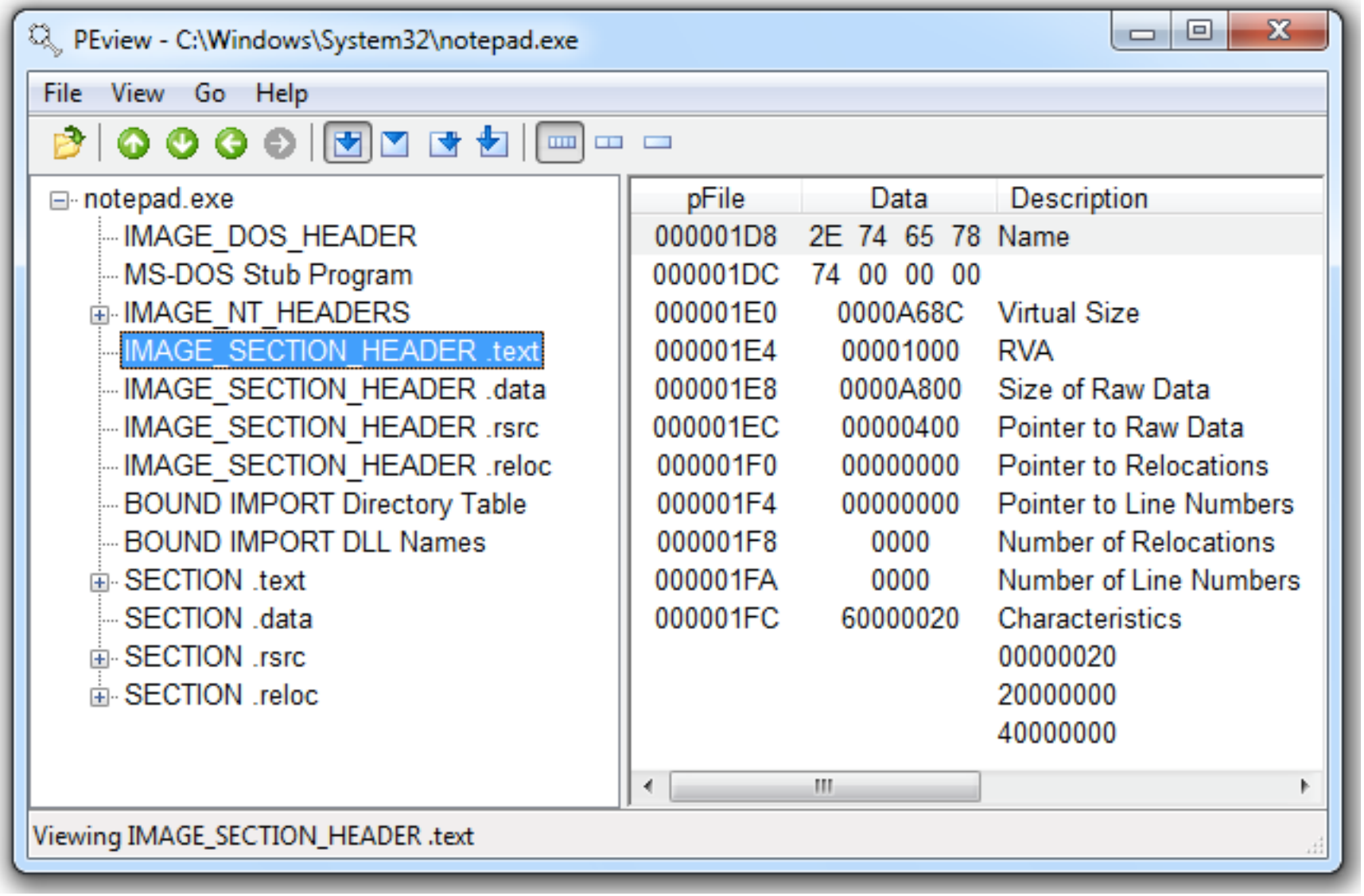
Time Date Stamp

- 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 – RAM
- Size of Raw Data – DISK
- For **.text** section, normally equal, or nearly equal
- Packed executables show Virtual Size much larger than Size of Raw Data for **.text** section

Not Packed


PEview - C:\Windows\System32\notepad.exe
File View Go Help
notepad.exe
IMAGE_DOS_HEADER
MS-DOS Stub Program
IMAGE_NT_HEADERS
IMAGE_SECTION_HEADER .text
IMAGE_SECTION_HEADER .data
IMAGE_SECTION_HEADER .rsrc
IMAGE_SECTION_HEADER .reloc
BOUND IMPORT Directory Table
BOUND IMPORT DLL Names
SECTION .text
SECTION .data
SECTION .rsrc
SECTION .reloc
pFile Data Description
000001D8 2E 74 65 78 Name
000001DC 74 00 00 00
000001E0 0000A68C Virtual Size
000001E4 00001000 RVA
000001E8 0000A800 Size of Raw Data
000001EC 00000400 Pointer to Raw Data
000001F0 00000000 Pointer to Relocations
000001F4 00000000 Pointer to Line Numbers
000001F8 0000 Number of Relocations
000001FA 0000 Number of Line Numbers
000001FC 60000020 Characteristics
00000020
20000000
40000000
Viewing IMAGE_SECTION_HEADER .text

- *PackedProgram.exe.*
- *The sections in this*
- File have a number of anomalies: The sections named Dijfpds, .sdfuok, and Kijijl are unusual, and the .text, .data, and .rdata sections are suspicious.
- The .text section has a Size of Raw Data value of 0, meaning that it takes up no space on disk, and its Virtual Size value is A000, which means that space will be allocated for the .text segment.
- This tells us that a packer will unpack the executable code to the allocated .text section.

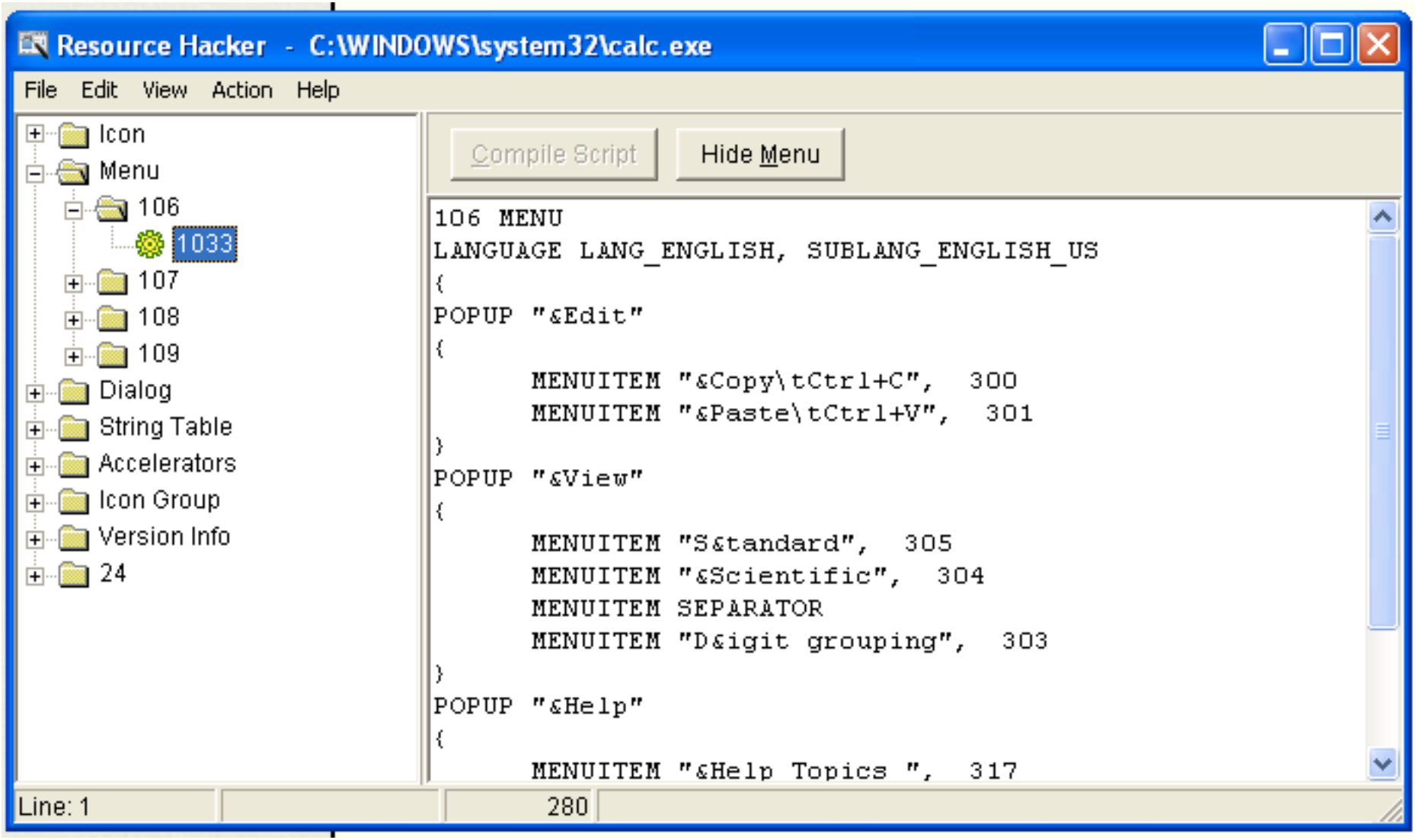
Table 2-6. Section Information for PackedProgram.exe

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

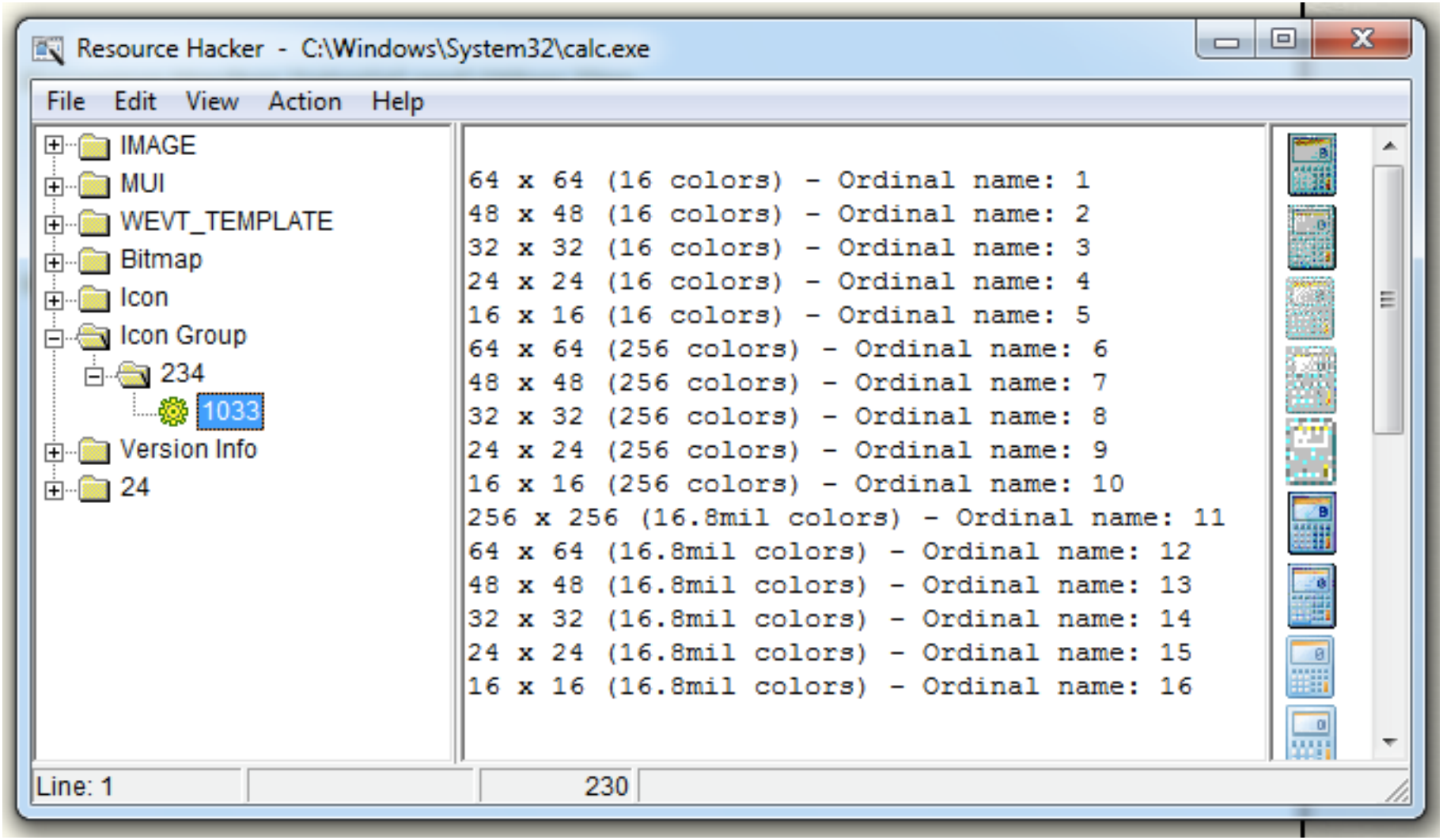
Resource Hacker

- Lets you browse the **.rsrc** section
- Resource Hacker™ has been designed to be the complete resource editing tool: compiling, viewing, decompiling and recompiling resources for both 32bit and 64bit Windows executables.
- Resource Hacker™ can open any type of Windows executable (*.exe; *.dll; *.scr; *.mui etc) so that individual resources can be added modified or deleted within these files.
- Resource Hacker™ can create and compile resource script files (*.rc), and edit resource files (*.res) too

Resource Hacker in Windows XP



Resource Hacker in Windows 7



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

Summary

- Using a suite of relatively simple tools, we can perform static analysis on malware to gain a certain amount of insight into its function
- static analysis is typically only the first step, and further analysis is usually necessary.