Practical Malware Analysis

Chapter 1: B A S I C S T A T I C T E C H N I Q U E S

Akbar Namin
Texas Tech University
Fall 2021

Reference:
Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software 1st Edition
by Michael Sikorski (Author), Andrew Honig (Author)

Techniques

- Antivirus scanning
- Hashes
- A file's strings, functions, and headers

Antivirus Scanning

- Only a First Step
- They rely mainly on a database of identifiable pieces of known suspicious code (file signatures), as well as behavioral and pattern-matching analysis (heuristics) to identify suspect files.
- Malware can easily change its signature and fool the antivirus
- VirusTotal (*http://www.virustotal.com)* is convenient, but using it may alert attackers that they've been caught

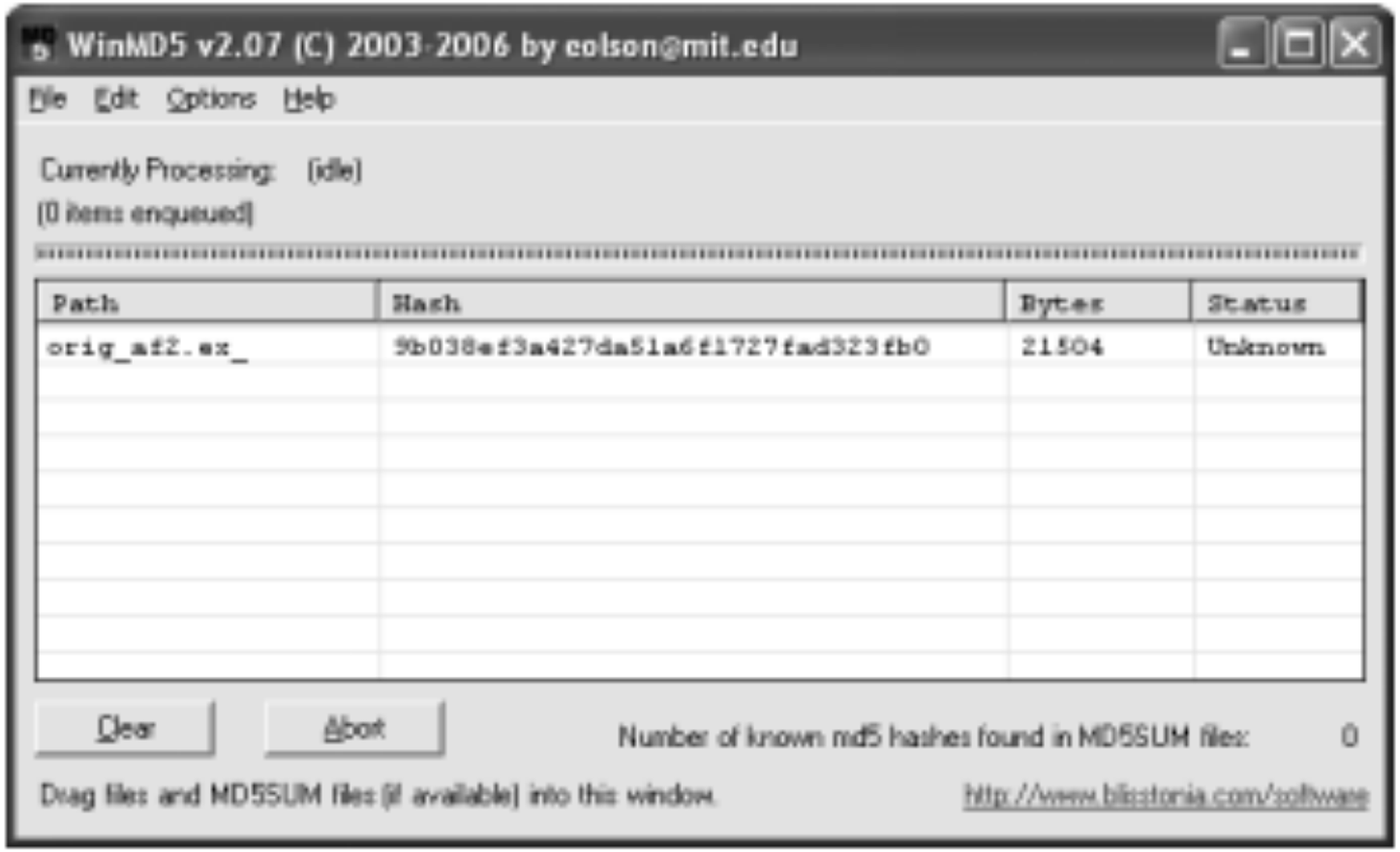
Hashing

- A fingerprint for malware
- The Message-Digest Algorithm 5 (MD5) or Secure Hash Algorithm 1 (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

```
C:\>md5deep c:\WINDOWS\system32\sol.exe
373e7a863a1a345c60edb9e20ec3231  c:\WINDOWS\system32\sol.exe
```

Figure1:md5deep program to calculate the hash of the Solitaire program

- GUI-based WinMD5 calculator, shown in Figure 2, can calculate and display hashes for several files at a time



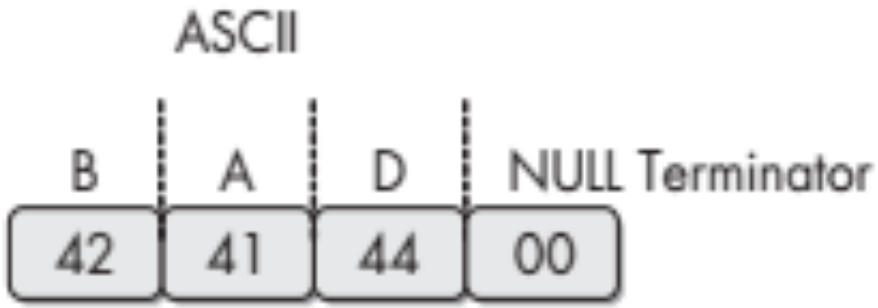
GUI-based WinMD5 calculator

Hash Uses

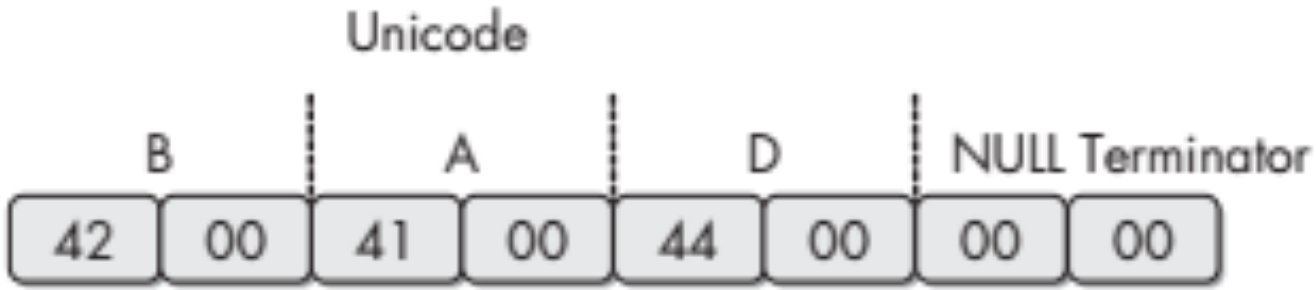
- Label a malware file
- Share the hash with other analysts to identify malware
- Search the hash online to see if someone else has already identified the file

Finding Strings

- 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"



ASCII representation of the string BAD



Unicode representation of the string BAD

The strings Command

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

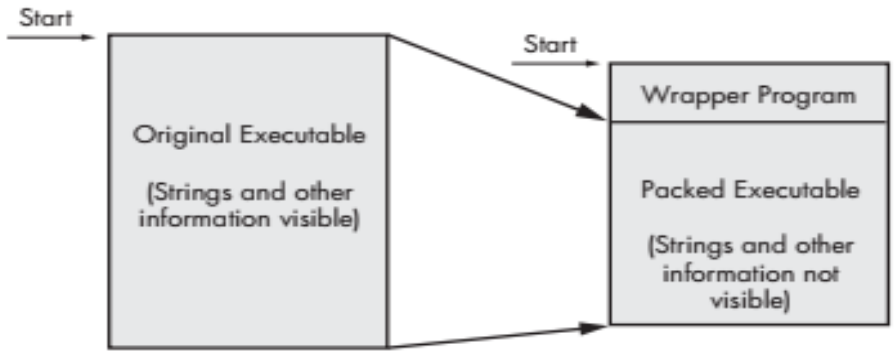
```
C:>strings bp6.ex_
VP3
VW3
t$@
D$4
99.124.22.1 ❹
e-@
GetLayout ❶
GDI32.DLL ❸
SetLayout ❷
M}C
Mail system DLL is invalid.!Send Mail failed to send message. ❺
```

Packed and 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.

Packing Files

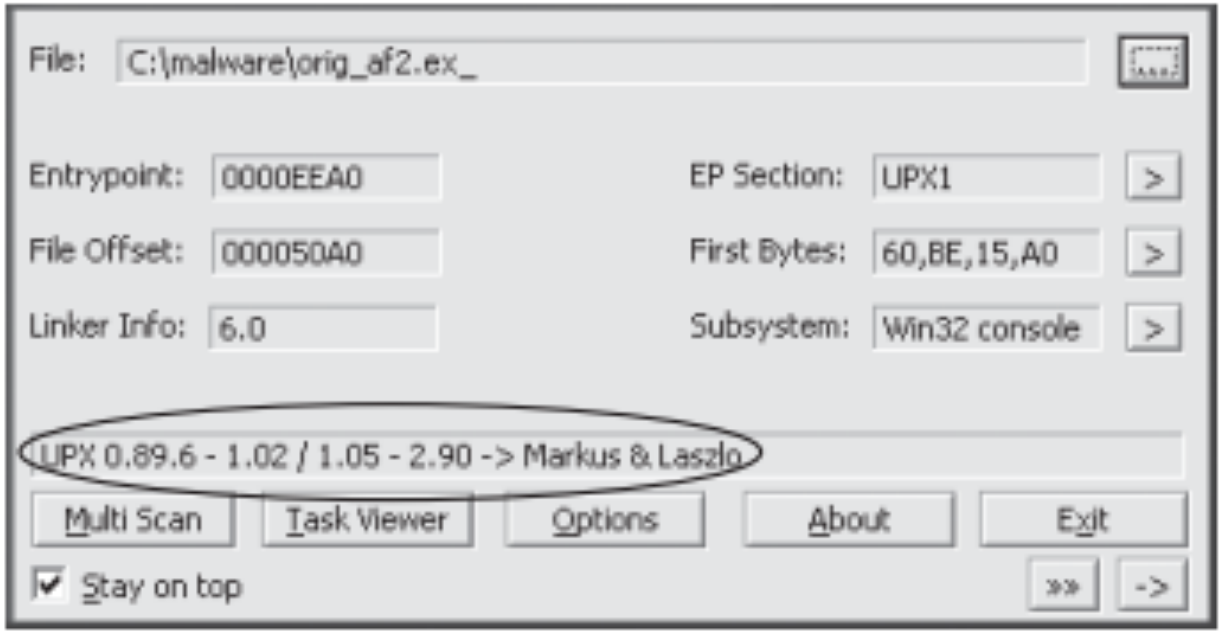
- The code is compressed, like Zip file
- This makes the strings and instructions unreadable
- All you'll see is the wrapper – small code that unpacks the file when it is run



The file on the left is the original executable, with all strings, imports, and other information visible. On the right is a packed executable. All of the packed file's strings, imports, and other information are compressed and invisible to most static analysis tools.

Detecting Packers with PEiD

- One way to detect packed files is with the PEiD program



Portable Executable File Format

- 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

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

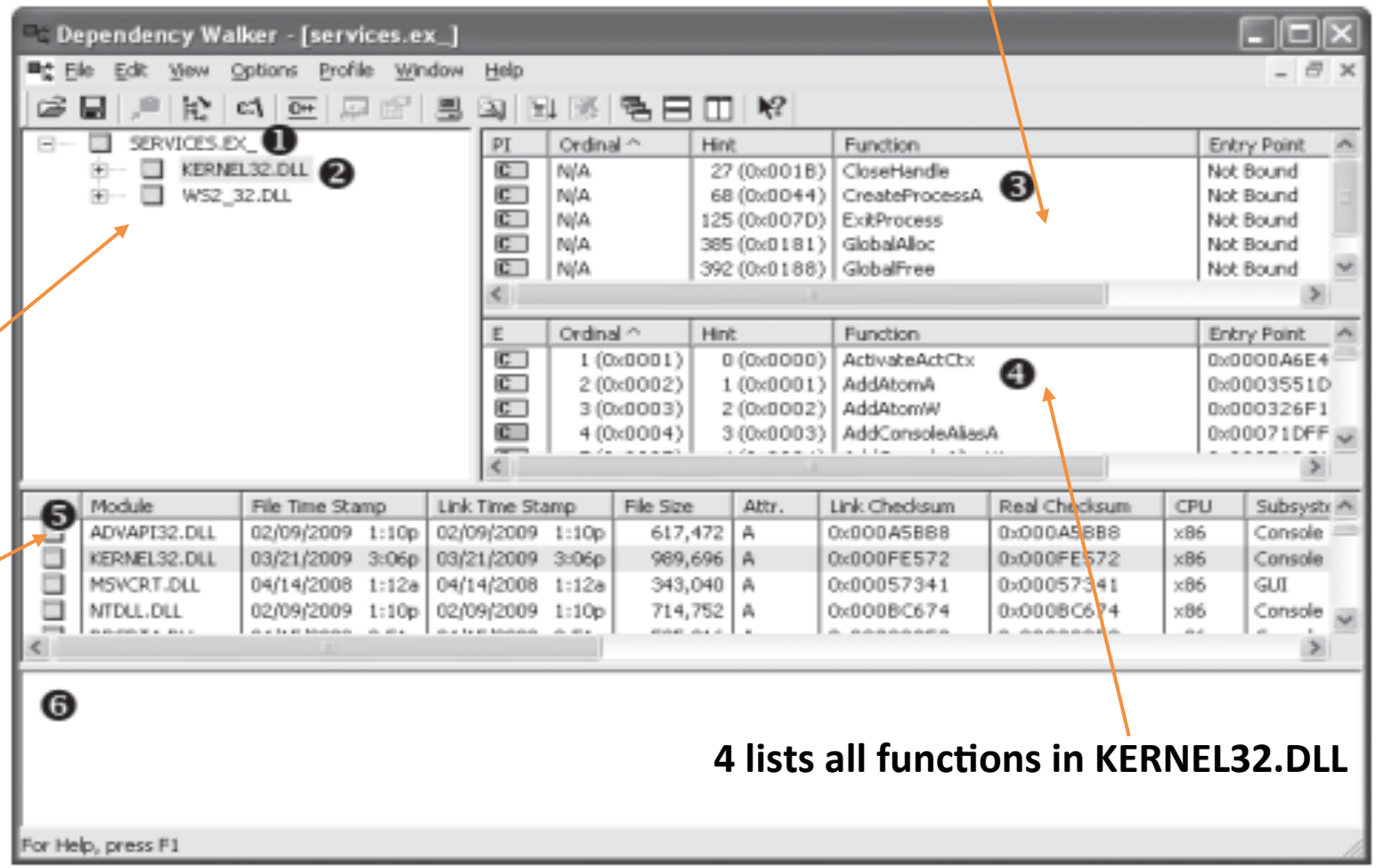
- Most common method
- Host OS searches for necessary libraries when the program is loaded
- The PE header lists every library and function that will be loaded
- Their names can reveal what the program does
- ***URLDownloadToFile*** indicates that the program downloads something

Dependency Walker

- Shows Dynamically Linked Functior
  - Normal programs have a lot of DLLs
  - Malware often has very few DLLs

1 & 2 shows the program as well as the DLLs being

3 shows imported functions



5 & 6 list additional information about the versions of DLLs that would be loaded if you ran the program and any reported errors, respectively

4 lists all functions in KERNEL32.DLL

Dependency Walker's analysis of SERVICES.EX_

- A program's DLLs can tell you a lot about its functionality

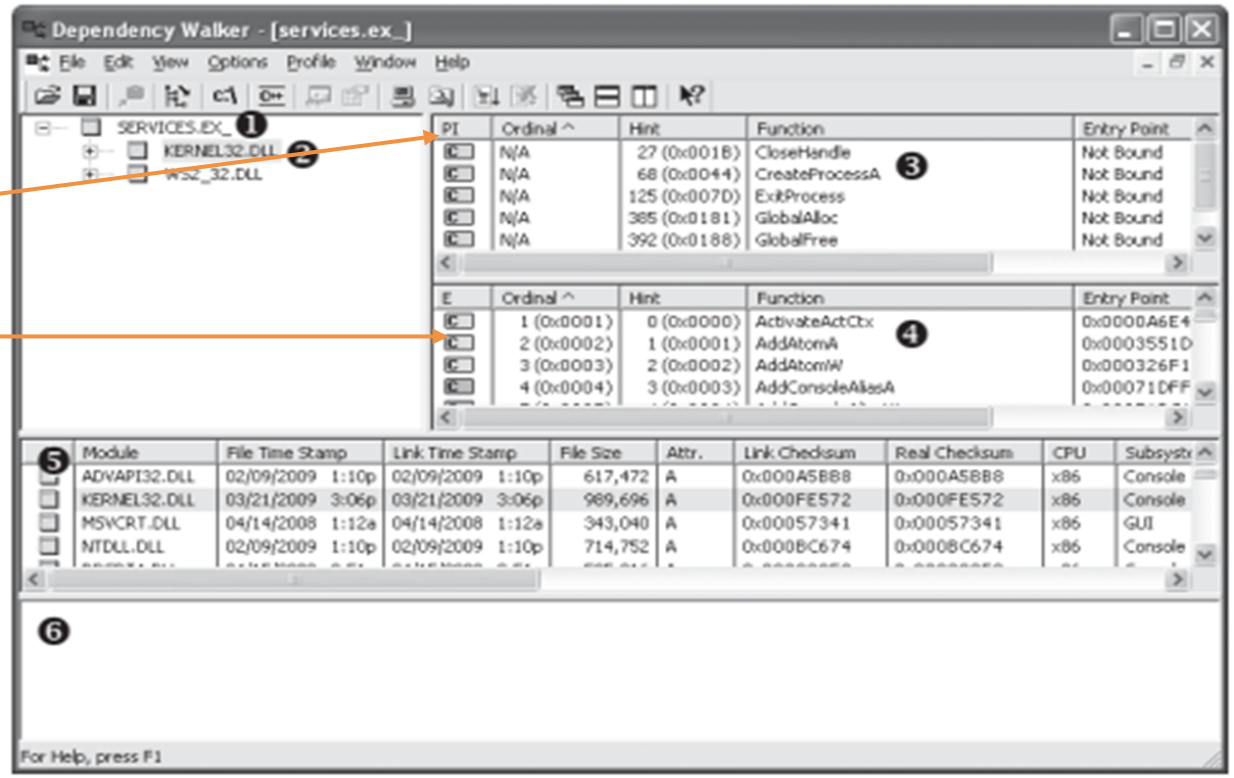
DLL	Description
Kernel32.dll	This is a very common DLL that contains core functionality, such as access and manipulation of memory, files, and hardware.
Advapi32.dll	This DLL provides access to advanced core Windows components such as the Service Manager and Registry.
User32.dll	This DLL contains all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

Imports & Exports in Dependency Walker

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

Import (PI)

Export(E)



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

Ex: A Packed Program

- Very few functions
- All you see is the unpacker

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

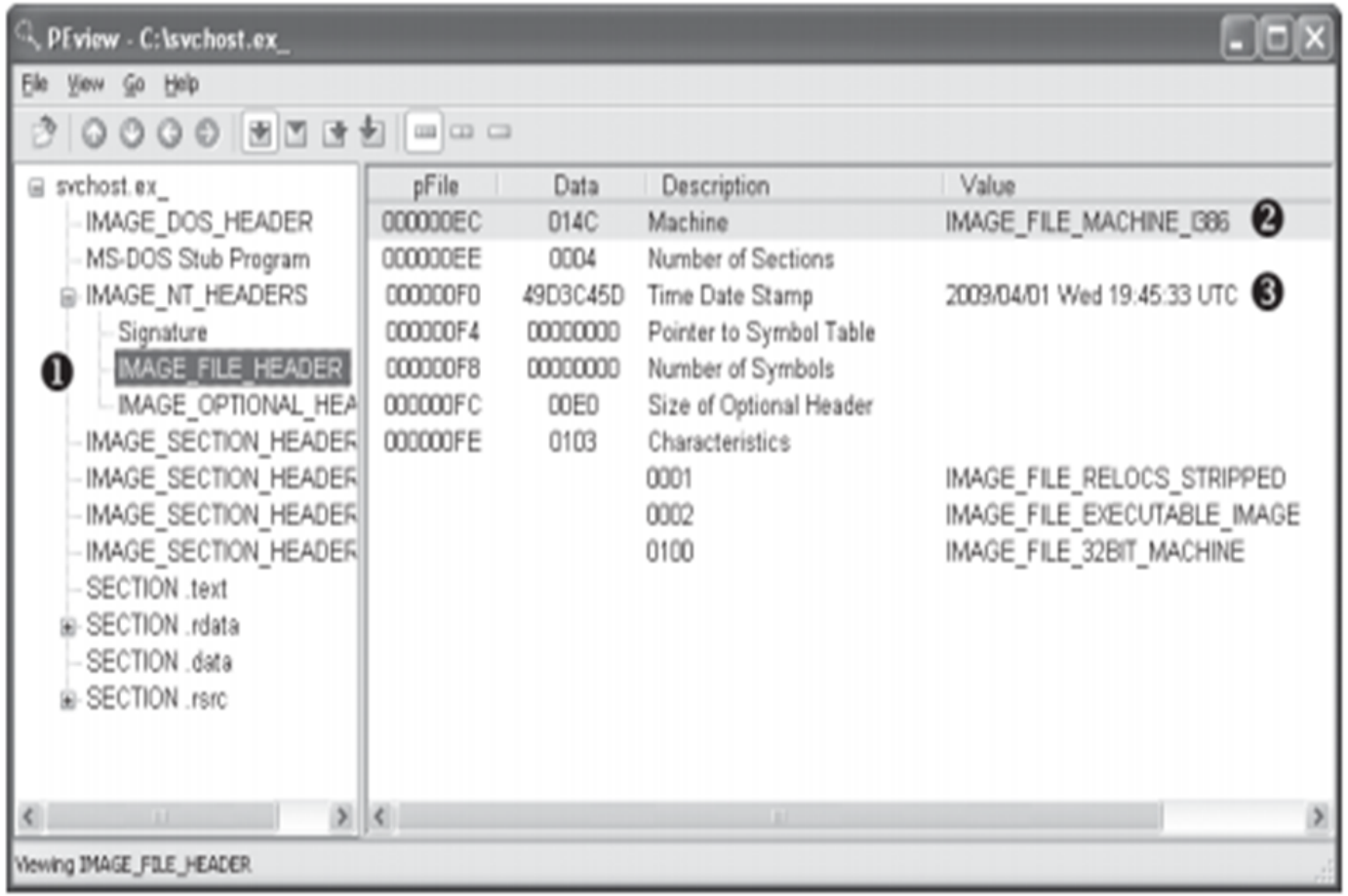
DLLs and Functions Imported from PackedProgram.exe

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

IMAGE_FILE_HEADER entry

1) displays the main parts of a PE header



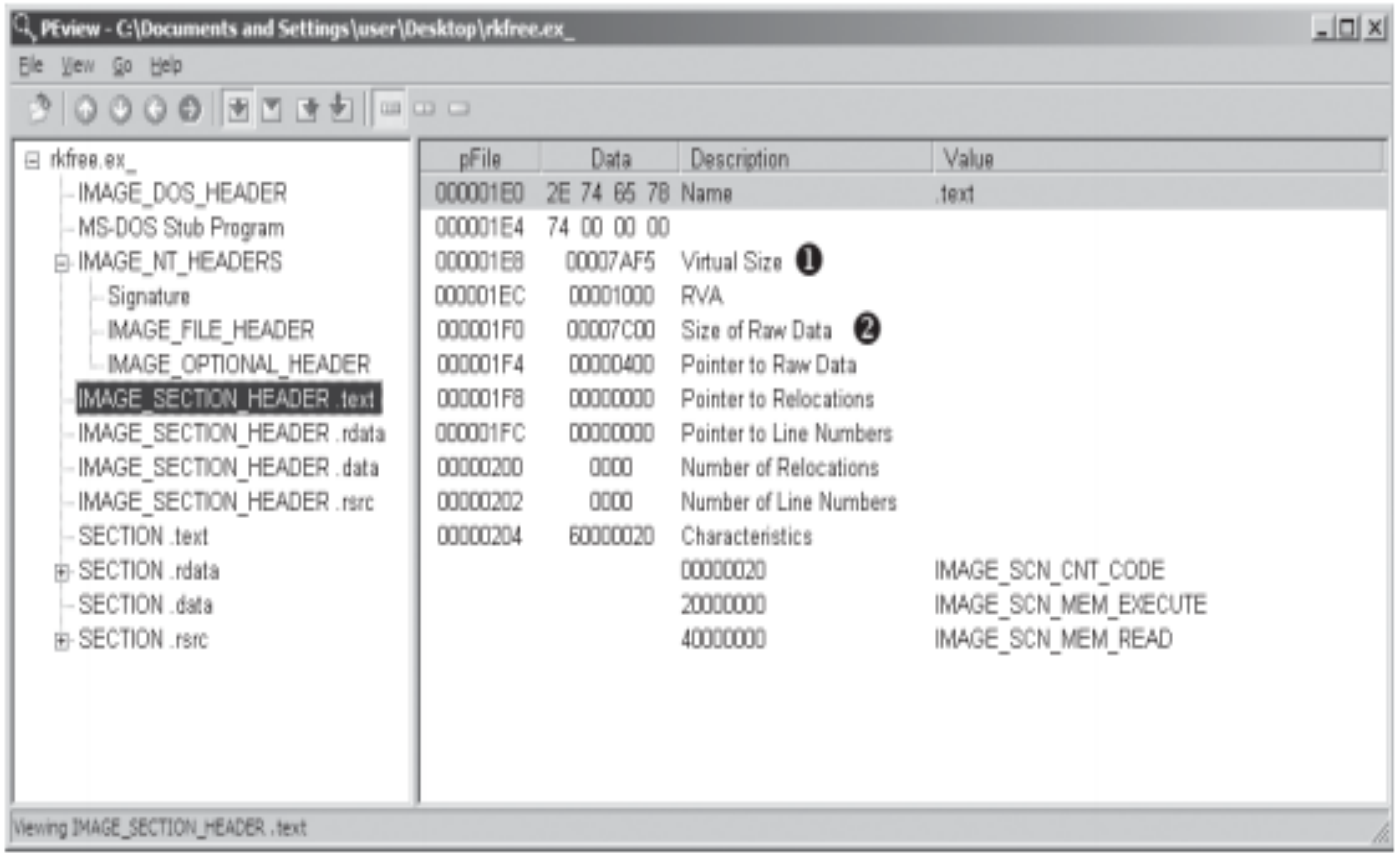
2) contains basic information about the file

3) 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

Viewing the IMAGE_SECTION_HEADER .text section in the PEview program

- 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



Sections from PotentialKeylogger.exe.

- .text, .rdata, and .rsrc sections each has a Virtual Size and Size of Raw Data value of about the same size.
- The .data section may seem suspicious because it has a much larger virtual size than raw data size
- This is normal for the .data because it is likely not packed and that the PE file header was generated by a compiler.

Section	Virtual size	Size of raw data
.text	7AF5	7C00
.data	17A0	0200
.rdata	1AF5	1C00
.rsrc	72B8	7400

Sections from 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.

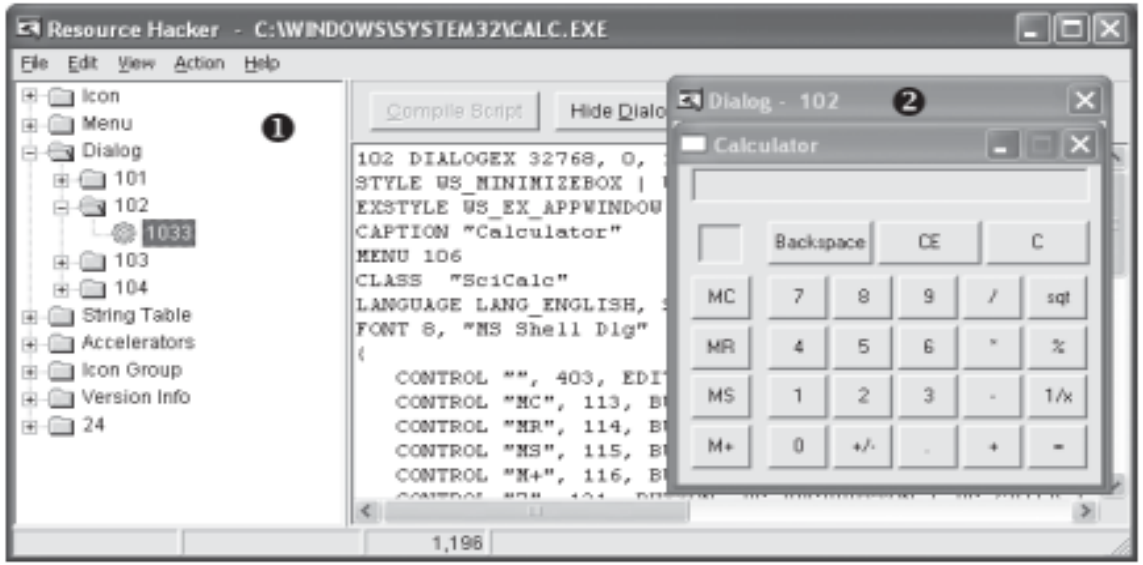
Name	Virtual size	Size of raw data
.text	A000	0000
.data	3000	0000
.rdata	4000	0000
.rsrc	19000	3400
Dijfpds	20000	0000
.sdfuok	34000	3313F
Kijijl	1000	0200

Viewing the Resource Section with Resource Hacker

Resource Hacker tool found at http://www.angusj.com/ to browse the .rsrc

The informative sections for malware analysis include:

- The Icon section lists images shown when the executable is in a file listing.
- The Menu section stores all menus that appear in various windows, such as the File, Edit, and View
- The Dialog section contains the program's dialog menus.
- The String Table section stores strings.
- The Version Info section contains a version number and often the company name and a copyright statement.



Using Other PE File Tools

- **PEBrowse Professional**(similar to Peview) (http://www.smidgeonsoft.prohosting.com/pebrowsepro-file-viewer.html)

- **PE Explorer** (http://www.heaventools.com/) has a rich GUI that allows you to navigate through the various parts of the PE file