

Practical Malware Analysis

Ch 1: Malware Analysis Primer

Updated 1-15-16

The Goals of Malware Analysis

Incident Response

- Case history
 - A medical clinic with 10 offices found malware on one of their workstations
 - Hired a consultant to clean & re-image that machine
- All done—case closed?

Incident Response


- After malware is found, you need to know
 - Did an attacker implant a rootkit or trojan on your systems?
 - Is the attacker really gone?
 - What did the attacker steal or add?
 - How did the attack get in
 - Root-cause analysis

Breach clean-up cost LinkedIn nearly \$1 million, another \$2-3 million in upgrades

Summary: LinkedIn executives reveal on quarterly earnings call just what the June theft of 6.5 million passwords cost the company in forensic work and on-going security updates.



By [John Fontana](#) for [Identity Matters](#) | August 3, 2012 -- 17:10 GMT (10:10 PDT)

 [Follow @johnfontana](#)

Comments

0



Vote

1



Like

4



Tweet

51



Share

[more +](#)

LinkedIn spent nearly \$1 million investigating and unraveling the [theft of 6.5 million passwords](#) in June and plans to spend up to \$3 million more updating security on its social networking site.

- Link Ch 1a

Malware Analysis

- Dissecting malware to understand
 - How it works
 - How to identify it
 - How to defeat or eliminate it
- A critical part of incident response

The Goals of Malware Analysis

- Information required to respond to a network intrusion
 - Exactly what happened
 - Ensure you've located all infected machines and files
 - How to measure and contain the damage
 - Find signatures for intrusion detection systems

Signatures

- Host-based signatures
 - Identify files or registry keys on a victim computer that indicate an infection
 - Focus on what the malware did to the system, not the malware itself
 - Different from antivirus signature
- Network signatures
 - Detect malware by analyzing network traffic
 - More effective when made using malware analysis

False Positives

CBS San Francisco Your Home Buy Tickets More FOLLOW US   LOGIN

City College Of San Francisco Computer Lab Security Breached

January 13, 2012 1:58 PM

Share this  Like 1  Tweet 3  +1 0  Share 2  View Comments

 Share CBS Local with your friends. Add us to your Timeline. [What's this?](#)



SAN FRANCISCO (KCBS) – The personal banking data from thousands of City College of San Francisco students, faculty and staff may be at risk because of a virus that infiltrated one computer lab – perhaps years ago.

Incredibly, the breach was only discovered recently – over the Thanksgiving holiday weekend.

KCBS' Holly Quan Reports:

City College of San Francisco (CCSF)

 **Click here to play audio**

What's most disturbing isn't that the IP addresses identified as receiving transmissions belong to the Russian Mafia –

Sponsored Links

 **\$28/Hr Data Entry Jobs At Home**
\$28/hr Part-Time Job Openi...
[StunningLifeStyle.com/Finance](#)

Malware Analysis Techniques

Static v. Dynamic Analysis

- Static Analysis
 - Examines malware without running it
 - Tools: VirusTotal, strings, a disassembler like IDA Pro
- Dynamic Analysis
 - Run the malware and monitor its effect
 - Use a virtual machine and take snapshots
 - Tools: RegShot, Process Monitor, Process Hacker, CaptureBAT
 - RAM Analysis: Mandant Redline and Volatility

Basic Analysis

- Basic static analysis
 - View malware without looking at instructions
 - Tools: VirusTotal, strings
 - Quick and easy but fails for advanced malware and can miss important behavior
- Basic dynamic analysis
 - Easy but requires a safe test environment
 - Not effective on all malware

Advanced Analysis

- Advanced static analysis
 - Reverse-engineering with a disassembler
 - Complex, requires understanding of assembly code
- Advanced Dynamic Analysis
 - Run code in a debugger
 - Examines internal state of a running malicious executable

Types of Malware

Types of Malware

- Backdoor
 - Allows attacker to control the system
- Botnet
 - All infected computers receive instructions from the same Command-and-Control (C&C) server
- Downloader
 - Malicious code that exists only to download other malicious code
 - Used when attacker first gains access

Types of Malware

- Information-stealing malware
 - Sniffers, keyloggers, password hash grabbers
- Launcher
 - Malicious program used to launch other malicious programs
 - Often uses nontraditional techniques to ensure stealth or greater access to a system
- Rootkit
 - Malware that conceals the existence of other code
 - Usually paired with a backdoor

Types of Malware

- Scareware
 - Frightens user into buying something
 - Link Ch 1b

Fake FBI warning tricks man into surrendering himself for possession of child porn

29 Jul, 2013 | by Nishtha Kanal



f Like

3



+1

0



Tweet

3



Share

Secure Your Application Today!

 CHECKMARX

Learn more 

Here's a weird one. We've heard of viruses and malware bringing harm to computers but in a rare instance, a "ransomware" has brought a positive outcome. A man in the US turned himself in to the police after a pop-up caused by a ransomware informed him that child porn had been identified on his machine.

Jay Matthew Riley, a 21-year-old from Virginia was browsing the Internet, when a pop-up containing an "FBI warning" informed him that it had detected child pornography on his machine. The message went on to tell Riley to pay up a fine online or face the consequences.

Types of Malware

- Spam-sending malware
 - Attacker rents machine to spammers
- Worms or viruses
 - Malicious code that can copy itself and infect additional computers

Mass v. Targeted Malware

- Mass malware
 - Intended to infect as many machines as possible
 - Most common type
- Targeted malware
 - Tailored to a specific target
 - Very difficult to detect, prevent, and remove
 - Requires advanced analysis
 - Ex: Stuxnet

General Rules for Malware Analysis

General Rules for Malware Analysis

- Don't Get Caught in Details
 - You don't need to understand 100% of the code
 - Focus on key features
- Try Several Tools
 - If one tool fails, try another
 - Don't get stuck on a hard issue, move along
- Malware authors are constantly raising the bar

Ch 2: Basic Static Analysis

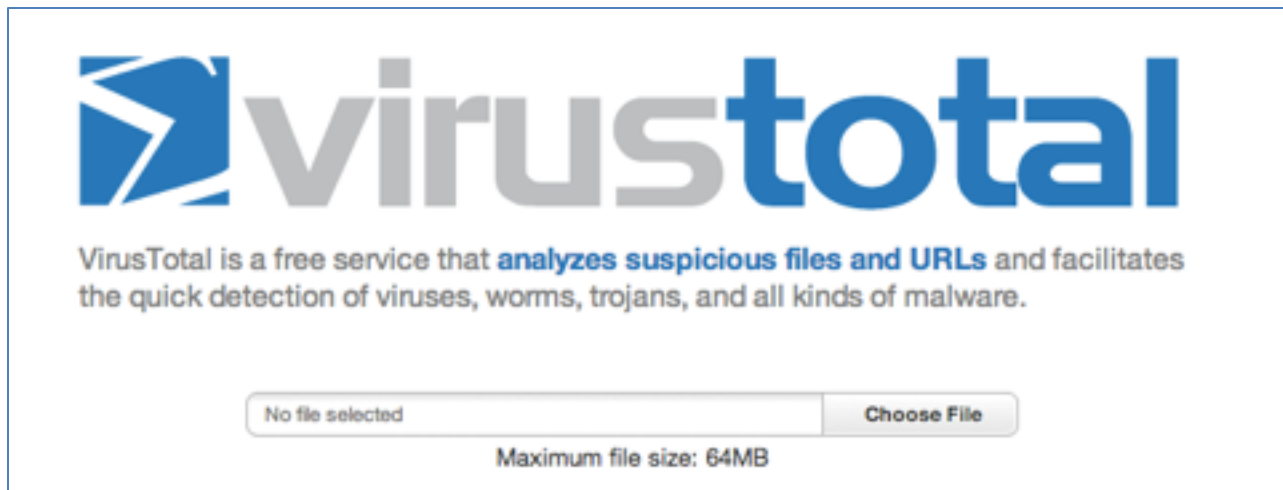
Techniques

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

Antivirus Scanning

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
 - Link Ch 2a



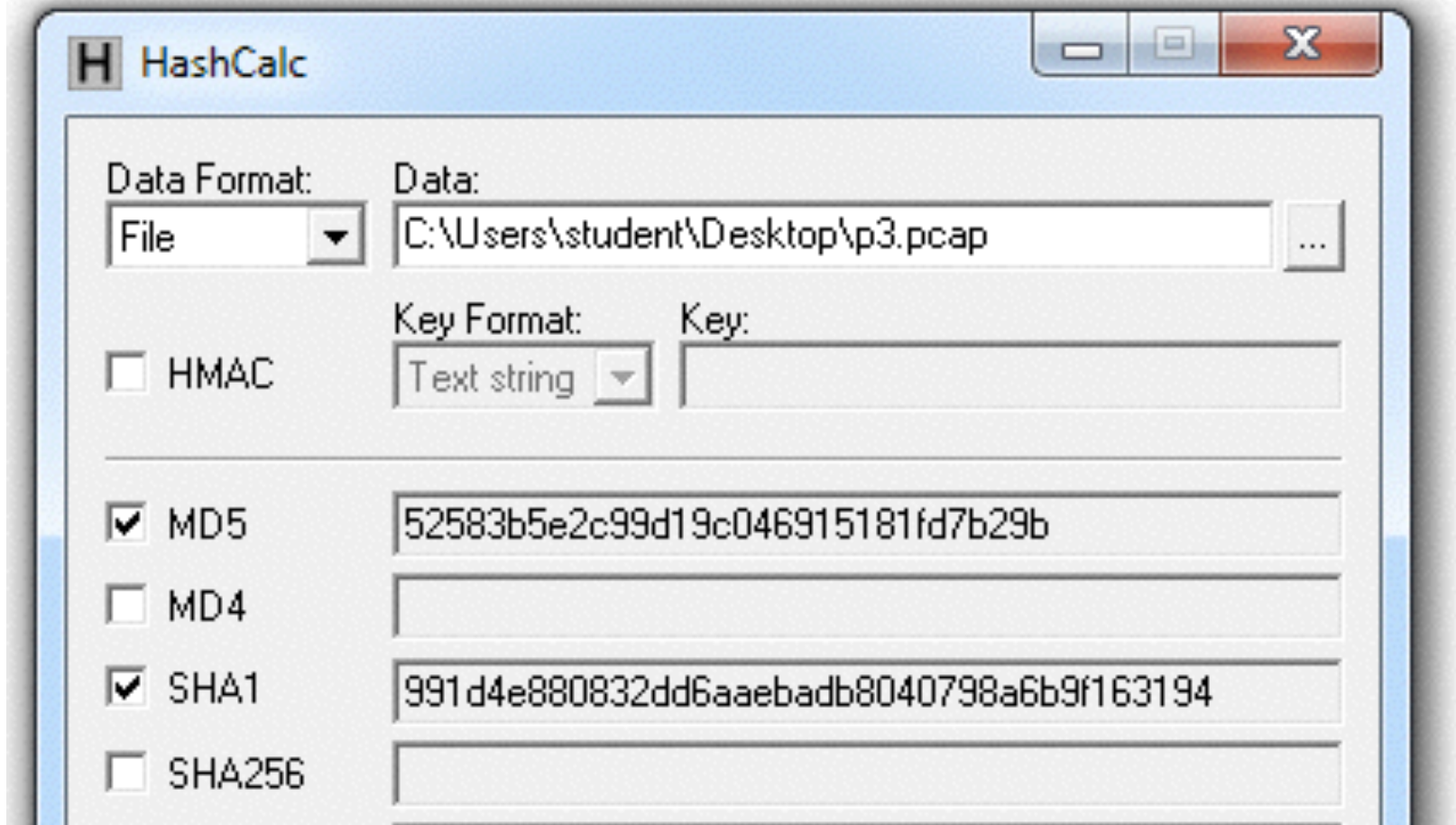
Hashing

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



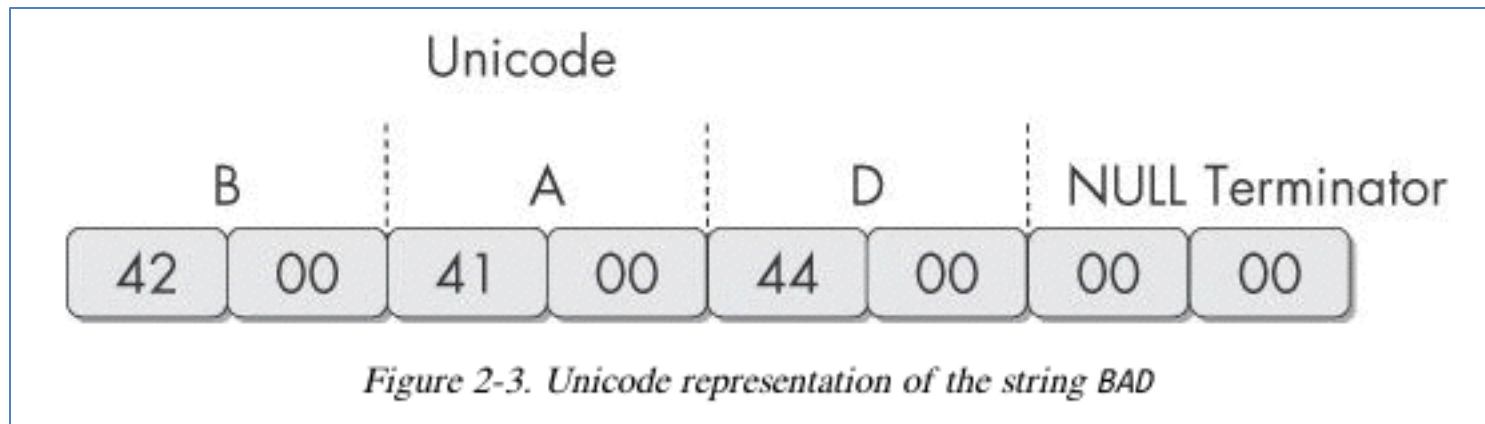
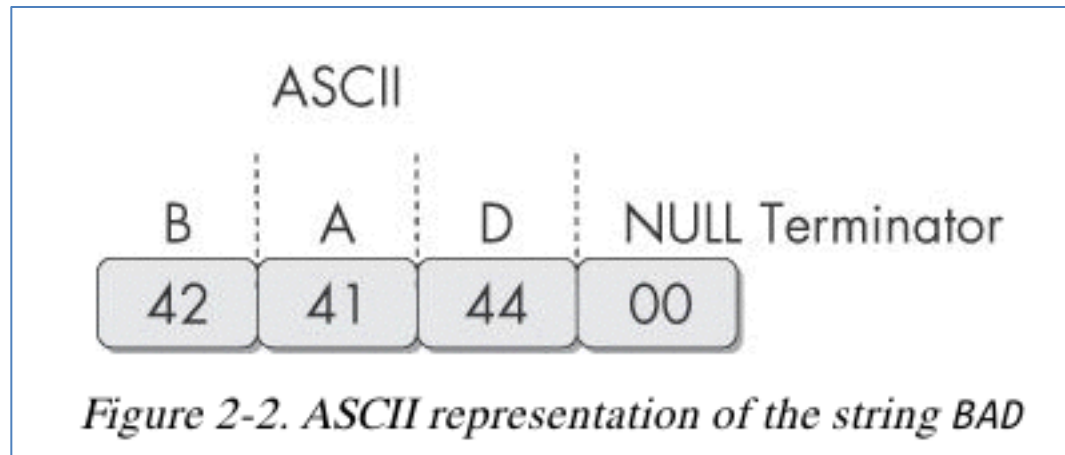
Hash Uses

- 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"



The strings Command

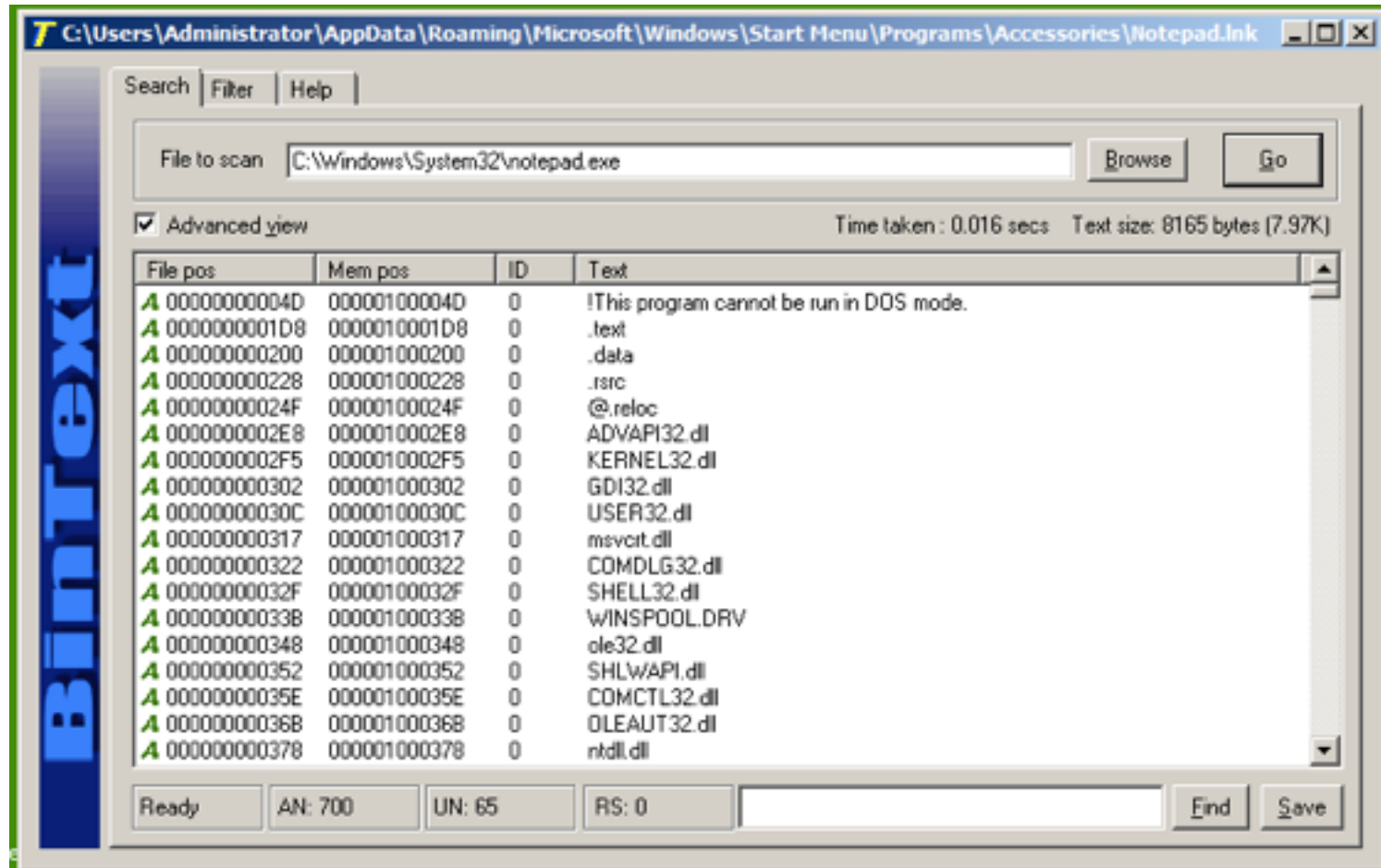
- Native in Linux, also available for Windows
- Finds all strings in a file 3 or more characters long

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
```

BinText

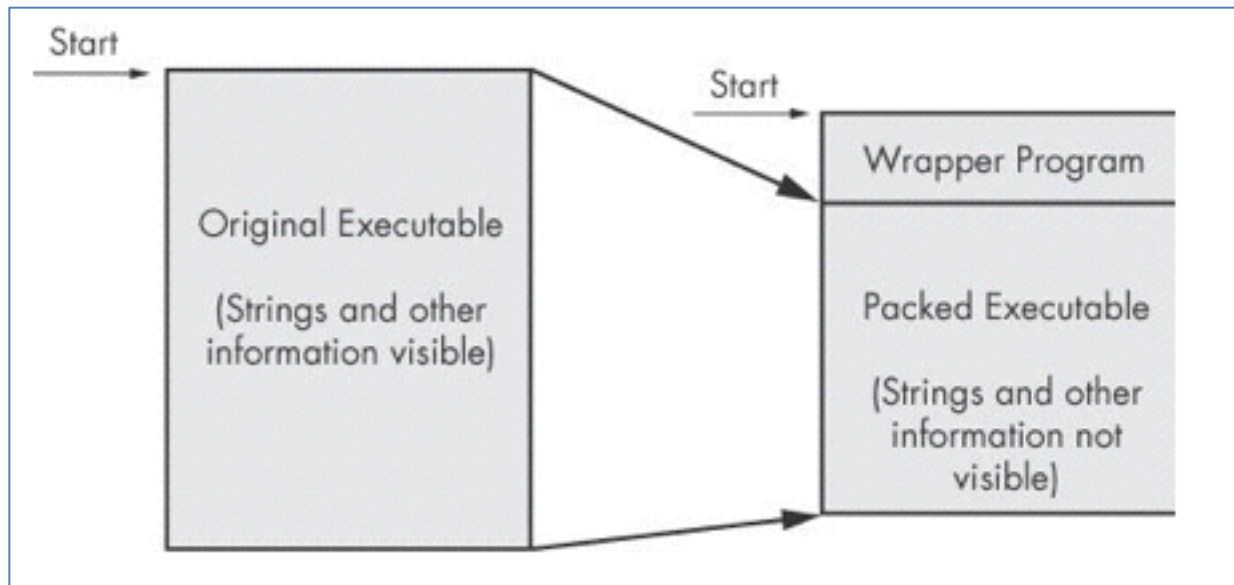


- Link Ch 2i

Packed and Obfuscated 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



Detecting Packers with PEiD

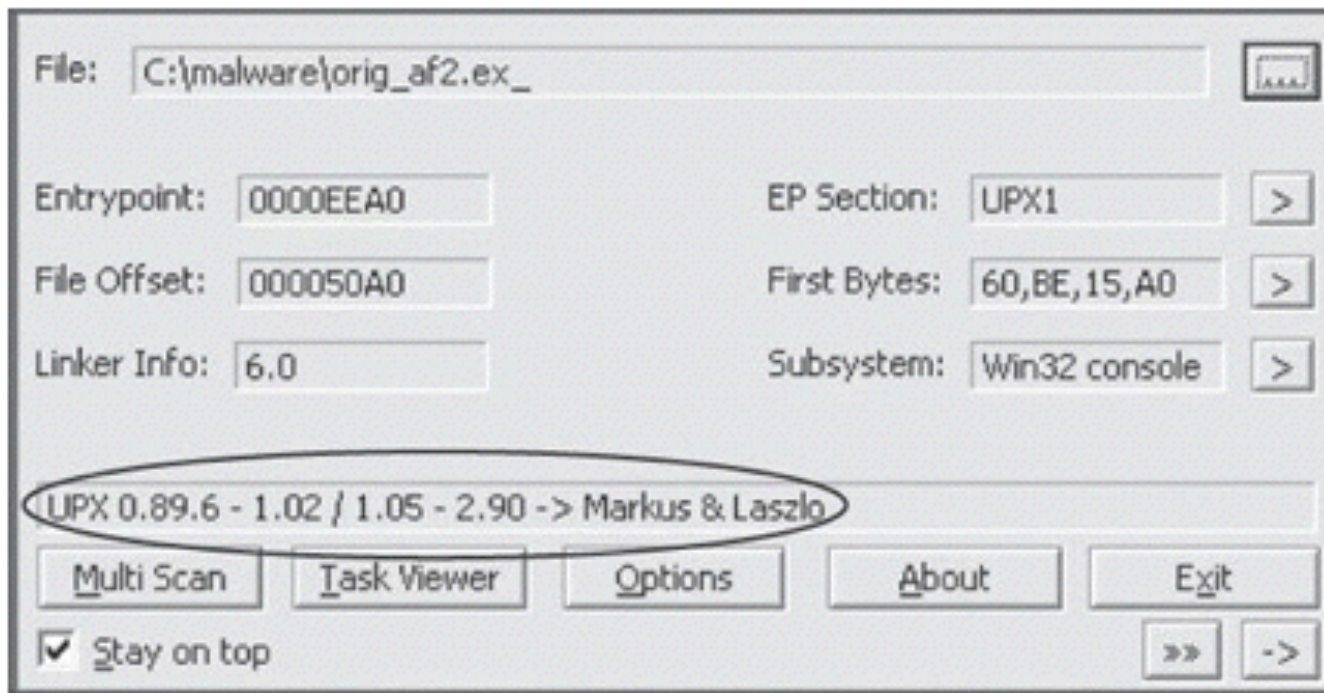


Figure 2-5. The PEiD program

Demo: UPX

```
root@kali: ~/126
File Edit View Search Terminal Help
root@kali:~/126# cat chatty.c
#include <stdio.h>
main()
{
char name[10];
printf("This program contains readable strings\n");
printf("Enter your name: ");
scanf("%s", name);
printf("Hello %s\n", name);
}

root@kali:~/126# gcc -static chatty.c -o chatty
root@kali:~/126# upx -o chatty-packed chatty
                Ultimate Packer for eXecutables
                Copyright (C) 1996 - 2011
UPX 3.08      Markus Oberhumer, Laszlo Molnar & John Reiser   Dec 12th 2011

  File size      Ratio      Format      Name
  -----
592800 -> 272588 45.98% linux/elf386 chatty-packed

Packed 1 file.
root@kali:~/126# ls -l
total 852
-rwxr-xr-x 1 root root 592800 Aug 16 20:34 chatty
-rw-r--r-- 1 root root 174 Aug 16 20:27 chatty.c
-rwxr-xr-x 1 root root 272588 Aug 16 20:34 chatty-packed
root@kali:~/126#
```

Packing Obfuscates Strings

```
root@kali:~/126# strings chatty | wc
1962    4498    33817
root@kali:~/126# strings chatty-packed | wc
3950    4290    23623
root@kali:~/126#
```


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

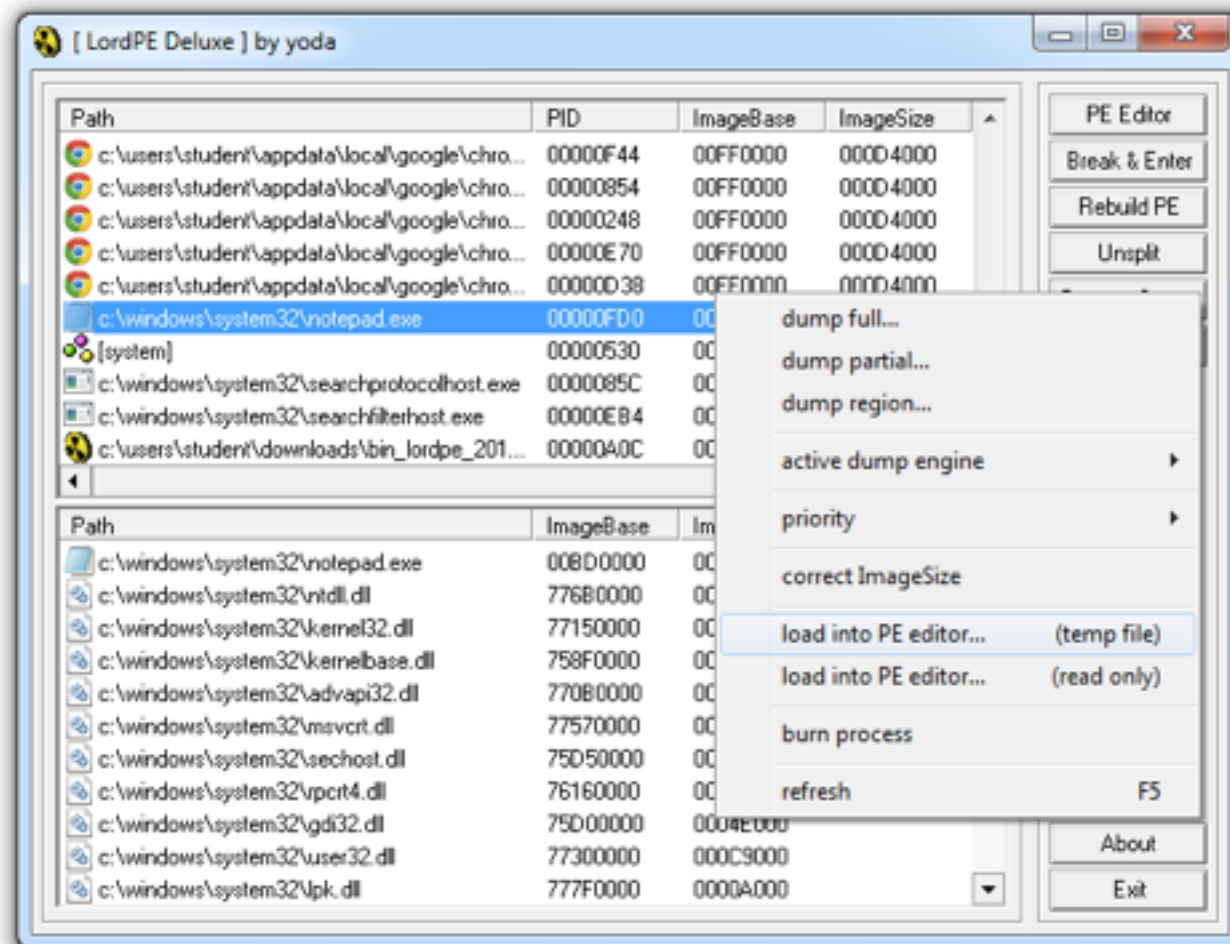
PE Files

- 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

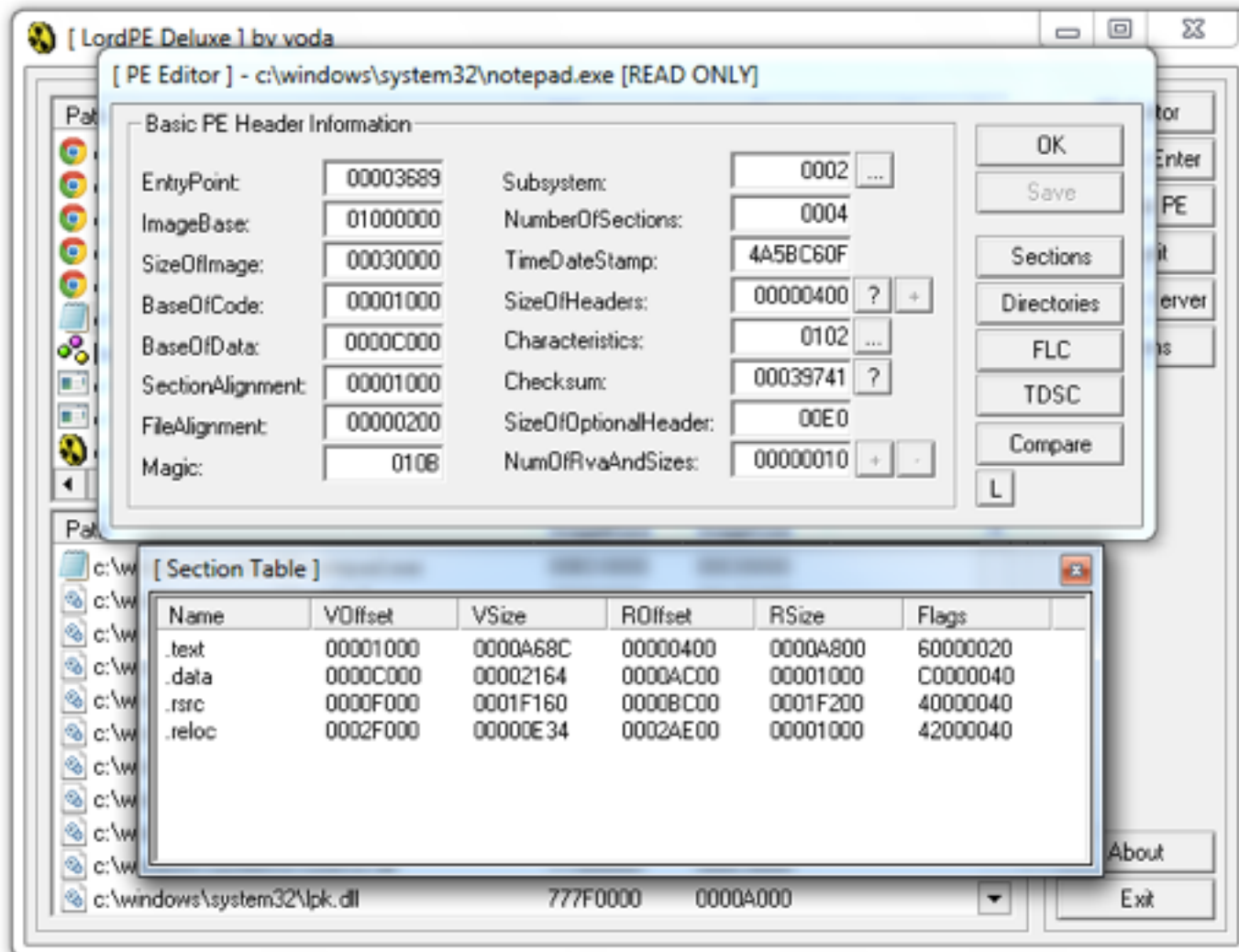
PE Header

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

LordPE Demo



Main Sections



There are a lot more sections

- But the main ones are enough for now
- Link Ch 2c

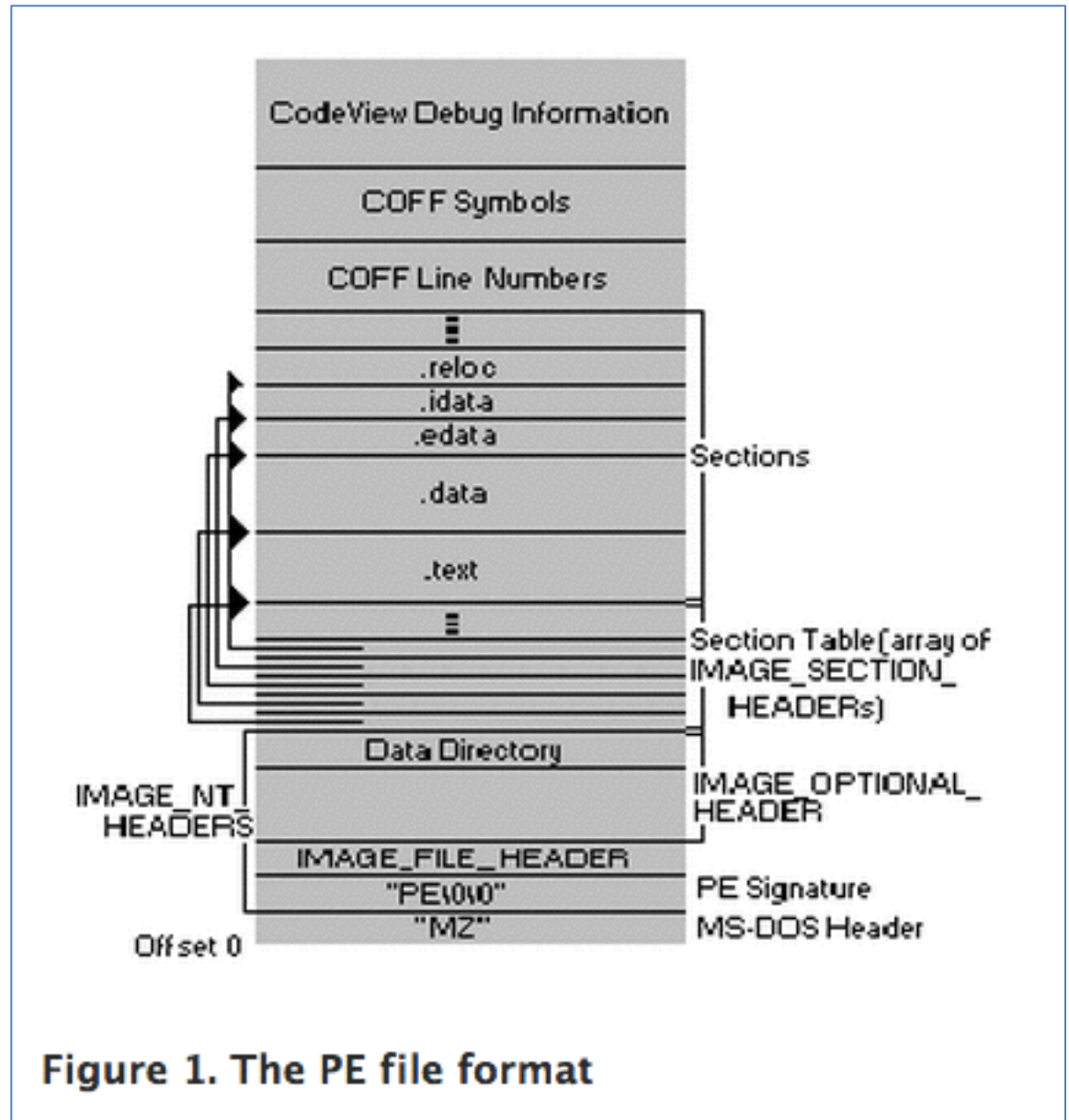


Figure 1. The PE file format

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

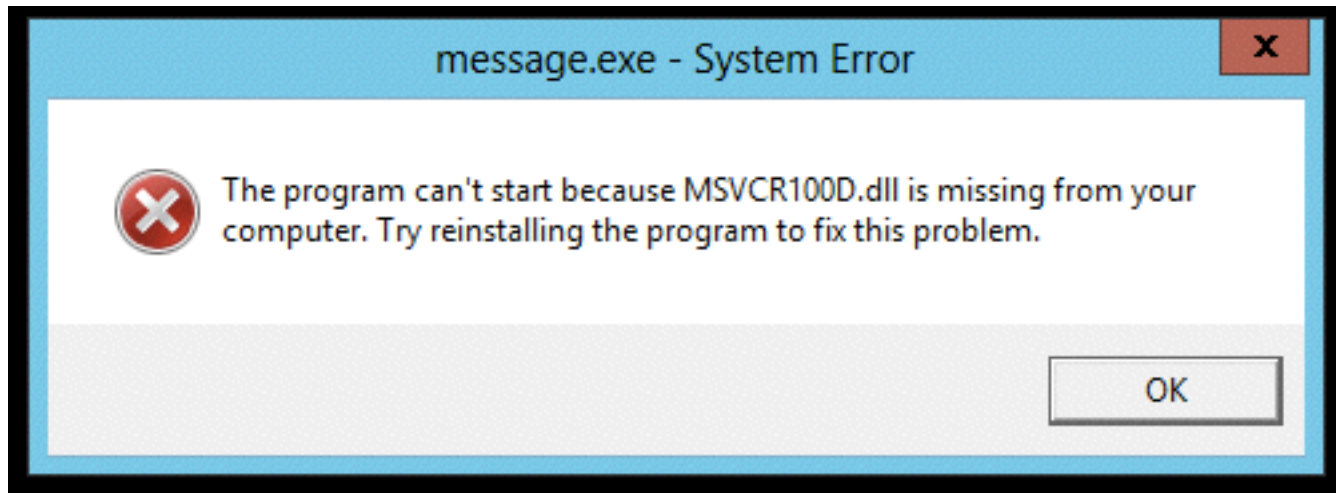
- Rarely used for Windows executables
- Common in Unix and Linux
- All code from the library is copied into the executable
- Makes executable large in size

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

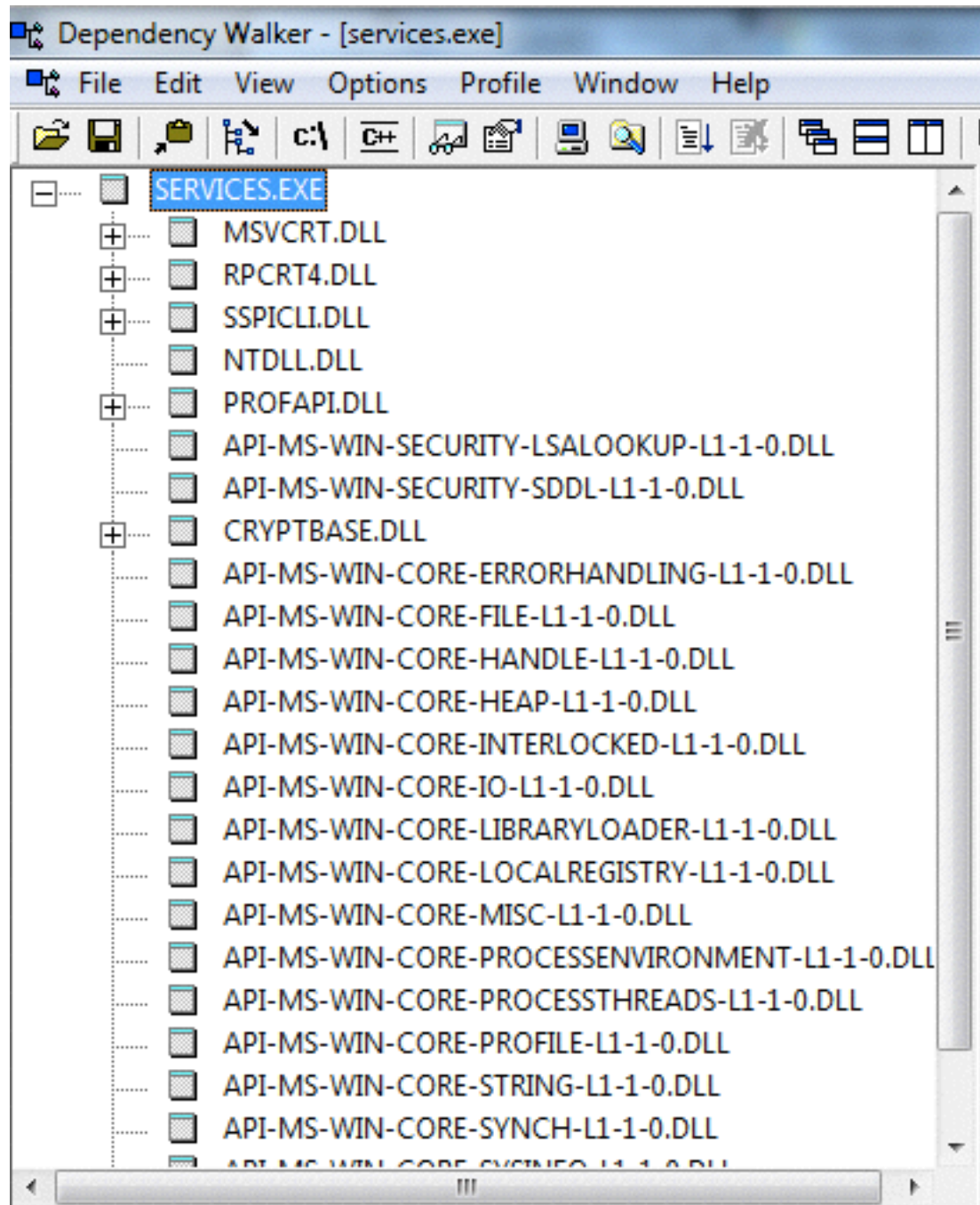
- The PE header lists every library and function that will be loaded
- Their names can reveal what the program does
- **URLDownloadToFile** indicates that the program downloads something

Dependency Walker

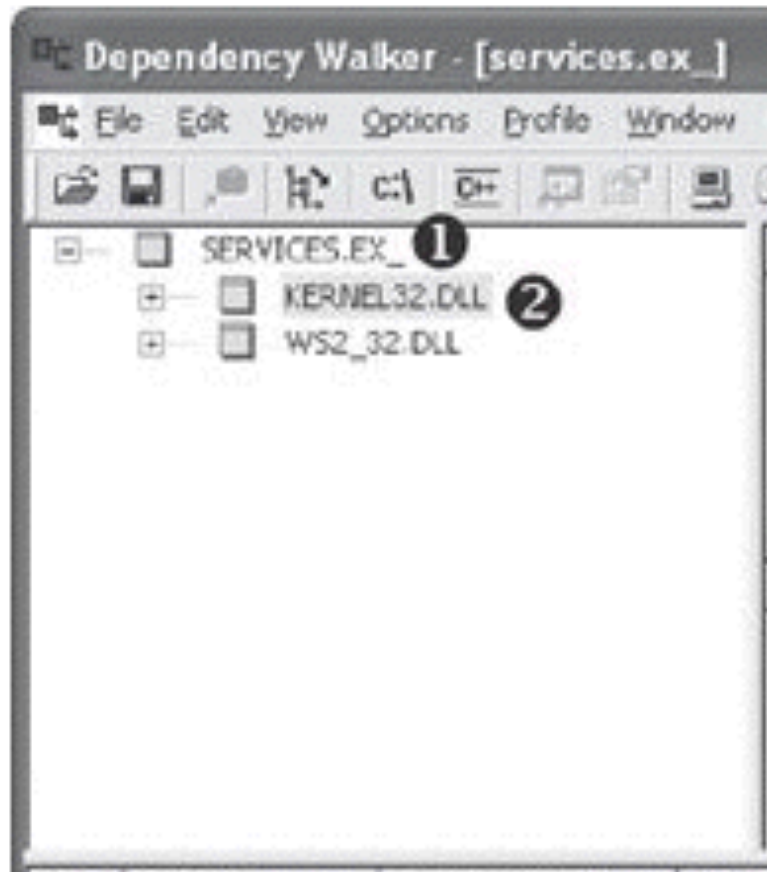
Shows Dynamically Linked Functions

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

Services.exe



Services.ex_ (malware)



Imports & Exports in Dependency Walker

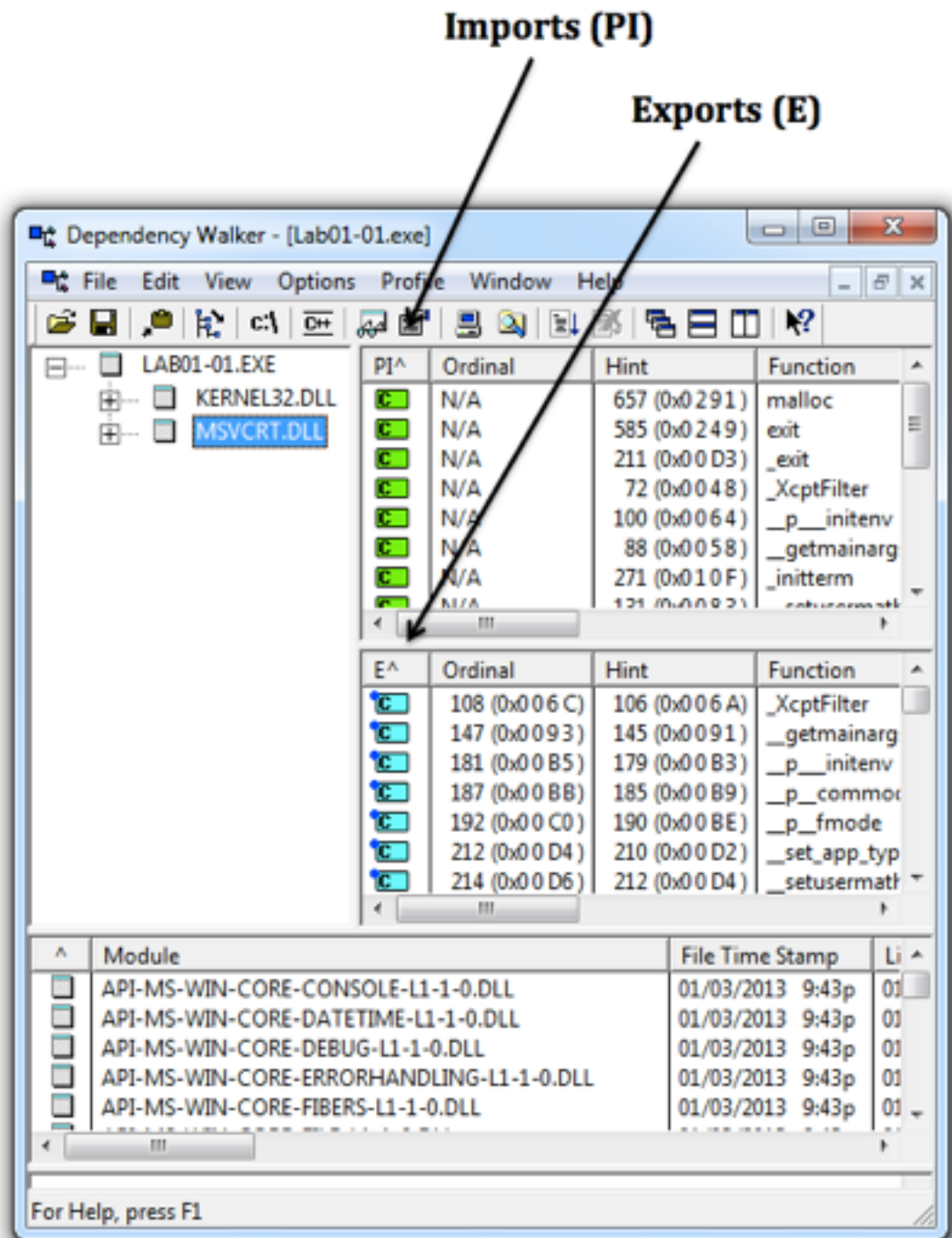


Table 2-1. Common DLLs

DLL	Description
<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.

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.

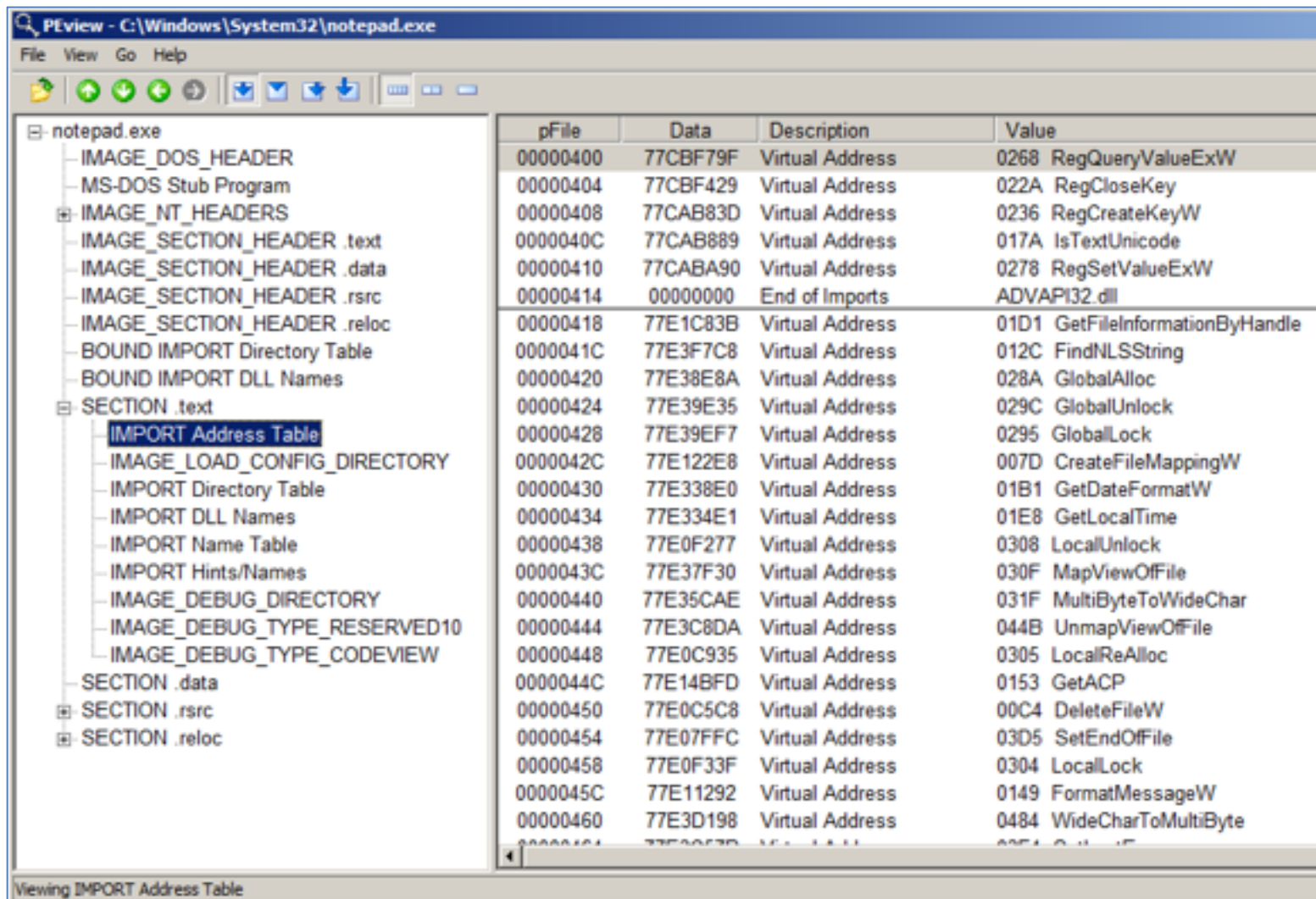
WSock32.dll and *Ws2_32.dll* These are networking DLLs. A program that accesses either of these most likely connects to a network or performs network-related tasks.

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

Notepad.exe



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

File View Go Help

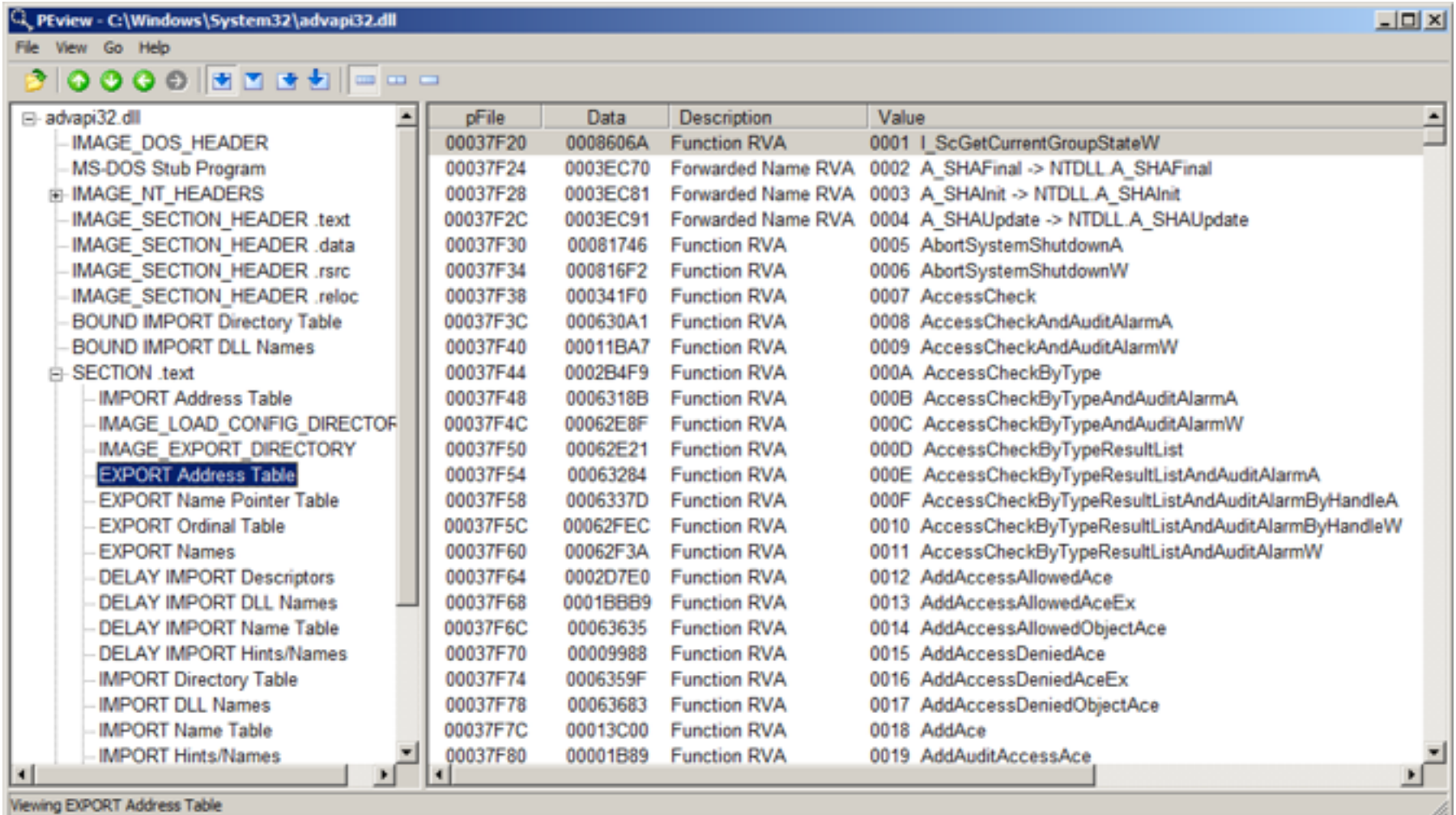
notepad.exe

- 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
 - IMPORT Address Table**
 - IMAGE_LOAD_CONFIG_DIRECTORY
 - IMPORT Directory Table
 - IMPORT DLL Names
 - IMPORT Name Table
 - IMPORT Hints/Names
 - IMAGE_DEBUG_DIRECTORY
 - IMAGE_DEBUG_TYPE_RESERVED10
 - IMAGE_DEBUG_TYPE_CODEVIEW
 - SECTION .data
 - SECTION .rsrc
 - SECTION .reloc

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	77E3F7C8	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



PEView - C:\Windows\System32\advapi32.dll

File View Go Help

advapi32.dll

- 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
 - IMPORT Address Table
 - IMAGE_LOAD_CONFIG_DIRECTORY
 - IMAGE_EXPORT_DIRECTORY
 - EXPORT Address Table**
 - EXPORT Name Pointer Table
 - EXPORT Ordinal Table
 - EXPORT Names
 - DELAY_IMPORT Descriptors
 - DELAY_IMPORT DLL Names
 - DELAY_IMPORT Name Table
 - DELAY_IMPORT Hints/Names
 - IMPORT Directory Table
 - IMPORT DLL Names
 - IMPORT Name Table
 - IMPORT Hints/Names

pFile	Data	Description	Value
00037F20	0008606A	Function RVA	0001 I_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	0001BBBB9	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

Viewing EXPORT Address Table

iTunesSetup.exe

PEView - C:\Users\Administrator\Downloads\iTunesSetup.exe

File View Go Help

iTunesSetup.exe

- IMAGE_DOS_HEADER
- MS-DOS Stub Program
- + IMAGE_NT_HEADERS
 - IMAGE_SECTION_HEADER .text
 - IMAGE_SECTION_HEADER .rdata
 - IMAGE_SECTION_HEADER .data
 - IMAGE_SECTION_HEADER .rsrc
 - IMAGE_SECTION_HEADER .reloc
 - SECTION .text
 - SECTION .rdata
 - IMPORT Address Table
 - IMAGE_DEBUG_DIRECTORY
 - IMAGE_LOAD_CONFIG_DIRECTORY
 - IMAGE_DEBUG_TYPE_CODEVIEW
 - IMPORT Directory Table
 - IMPORT Name Table
 - IMPORT Hints/Names & DLL Names
 - SECTION .data
 - + SECTION .rsrc
 - + SECTION .reloc
 - CERTIFICATE Table

pFile	Raw Data	Value
00000000	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZ.....
00000010	B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00@.....
00000020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000030	00 00 00 00 00 00 00 00 00 00 00 00 00 E8 00 00 00E.....
00000040	0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68!..L!Th
00000050	69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F	is program canno
00000060	74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20	t be run in DOS
00000070	6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00	mode...\$.....
00000080	3C E2 60 D1 78 83 0E 82 78 83 0E 82 78 83 0E 82	<.`x...x...x...
00000090	66 D1 8A 82 59 83 0E 82 66 D1 9B 82 6B 83 0E 82	f...Y...f...k...
000000A0	5F 45 63 82 6D 83 0E 82 5F 45 75 82 71 83 0E 82	_Ec.m..._Eu.q...
000000B0	78 83 0F 82 01 83 0E 82 66 D1 8D 82 17 83 0E 82	x...f...
000000C0	66 D1 9A 82 79 83 0E 82 66 D1 9F 82 79 83 0E 82	f...y...f...y...
000000D0	52 69 63 68 78 83 0E 82 00 00 00 00 00 00 00 00	Richx.....
000000E0	00 00 00 00 00 00 00 00 50 45 00 00 4C 01 05 00PE..L...
000000F0	16 44 F7 55 00 00 00 00 00 00 00 00 00 E0 00 02 01	.D.U.....
00000100	0B 01 09 00 00 1E 01 00 00 BE 7A 06 00 00 00 00 00z.....
00000110	63 BA 00 00 00 10 00 00 00 30 01 00 00 00 40 00	c.....0...@...
00000120	00 10 00 00 00 02 00 00 05 00 00 00 00 00 00 00
00000130	05 00 00 00 00 00 00 00 00 00 7C 06 00 04 00 00
00000140	58 97 7C 06 02 00 40 81 00 00 10 00 00 10 00 00	X. ...@.....
00000150	00 00 10 00 00 10 00 00 00 00 00 00 10 00 00 00
00000160	00 00 00 00 00 00 00 00 74 4D 01 00 64 00 00 00tM..d...
00000170	00 C0 01 00 48 CE 79 06 00 00 00 00 00 00 00 00	...H.y.....
00000180	00 9E 7B 06 18 19 00 00 00 90 7B 06 F8 0B 00 00	..{.....{.....
00000190	F0 31 01 00 1C 00 00 00 00 00 00 00 00 00 00 00	1

Viewing iTunesSetup.exe export Table

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

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

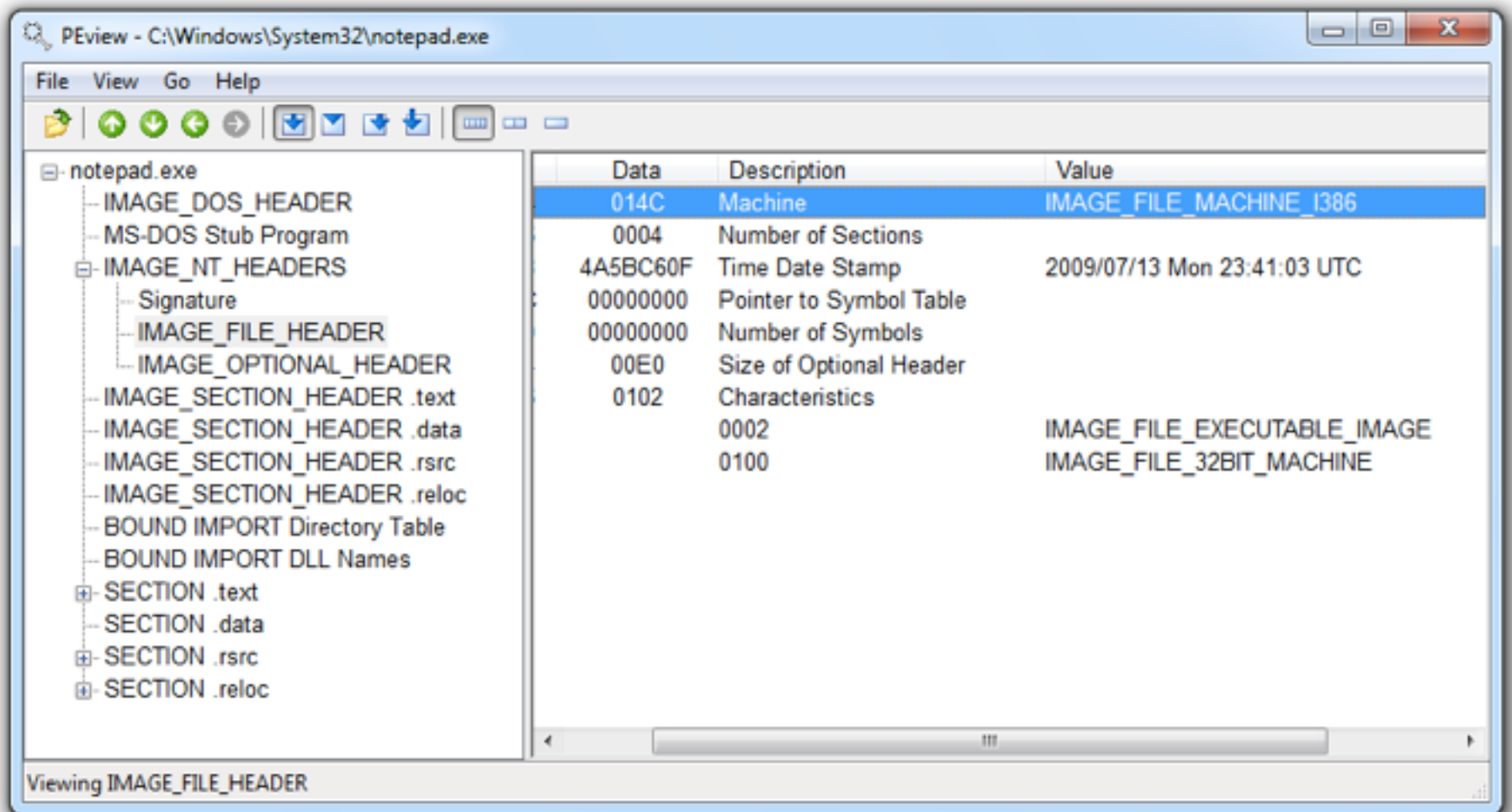
Kernel32.dll	User32.dll
GetModuleHandleA	MessageBoxA
LoadLibraryA	
GetProcAddress	
ExitProcess	
VirtualAlloc	
VirtualFree	

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 (Link Ch 2e)



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

The screenshot shows the PEView application window for notepad.exe. The left pane displays the file's structure, with the `IMAGE_SECTION_HEADER .text` entry selected. The right pane shows a table of the fields for this section header.

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

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
- Strings, icons, and menus
- [Link Ch 2f](#)

Resource Hacker

