

PMA 101c: Basic Static Techniques (20 pts + 30 pts extra)

What you need:

- The Malware Analysis Cloud Machine you prepared in the previous project

Malware Samples

This project uses two files in this folder:

C:\Users\yourname\Desktop\Malware\Practical Malware Analysis Labs\BinaryCollection\Chapter_1L

The two files are **Lab01-01.exe** and **Lab01-01.dll**.

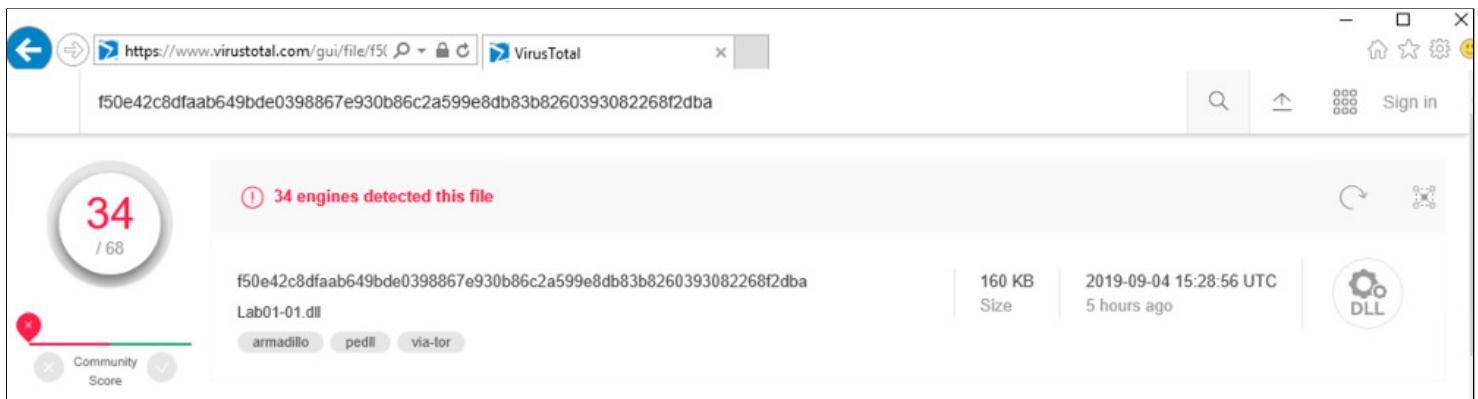
VirusTotal

In a Web browser, go to

<https://www.virustotal.com>

Upload **Lab01-01.dll**. As shown below, some of the engines detect it as malware.

VirusTotal compares a file to a database of antivirus engines. You can upload files, but that may alert attackers that you have detected an intrusion. Using it to search for a hash value of a sample is safer.



34 / 68

34 engines detected this file

f50e42c8dfaab649bde0398867e930b86c2a599e8db83b8260393082268f2dba

Lab01-01.dll

Community Score

160 KB | 2019-09-04 15:28:56 UTC | 5 hours ago

armadillo | pedll | via-lor

DLL

Flag PMA 101.1 PEview (5 pts)

PEview shows the sections that make up a PE (Portable Executable) file.

Goto <http://wjradburn.com/software/>.

Download **PEview version 0.9.9** and unzip it. Double click **PEview.exe**

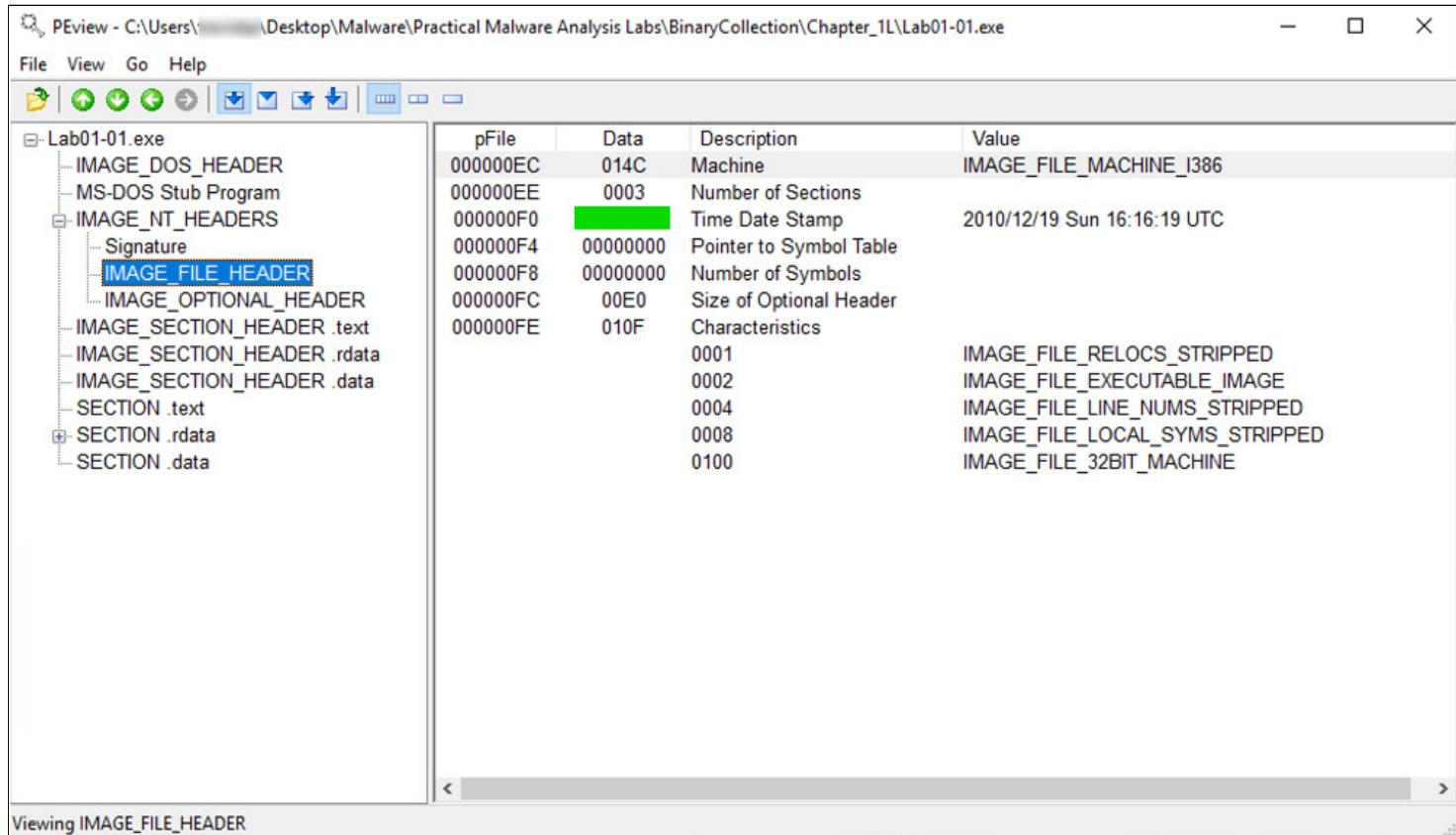
Click **Run** at the security warning.

Navigate to **C:\Users\yourname\Desktop\Malware\Practical Malware Analysis Labs\Binary Collection\Chapter_1L** and open **Lab01-01.exe**.

On the left side, expand the **IMAGE_NT_HEADERS** container and click **IMAGE_FILE_HEADER**.

The "Time Date Stamp" shows when the files were compiled. This is often used as an indication of the time zone the attackers live in. Files that were compiled at the same time are also often regarded as part of the same package.

Find the **Data** that is covered by a green box in the image below. That's the flag.



pFile	Data	Description	Value
000000EC	014C	Machine	IMAGE_FILE_MACHINE_I386
000000EE	0003	Number of Sections	
000000F0		Time Date Stamp	2010/12/19 Sun 16:16:19 UTC
000000F4	00000000	Pointer to Symbol Table	
000000F8	00000000	Number of Symbols	
000000FC	00E0	Size of Optional Header	
000000FE	010F	Characteristics	
	0001		IMAGE_FILE_RELOCS_STRIPPED
	0002		IMAGE_FILE_EXECUTABLE_IMAGE
	0004		IMAGE_FILE_LINE_NUMS_STRIPPED
	0008		IMAGE_FILE_LOCAL_SYMS_STRIPPED
	0100		IMAGE_FILE_32BIT_MACHINE

Flag PMA 101.2: PEiD (5 pts)

PEiD shows what language the sample was written in, or what packer was used if it's packed.

Go to <https://www.softpedia.com/get/Programming/Packers-Crypters-Protectors/PEiD-updated.shtml> and click **Free Download**.

If that doesn't work, you can download the file [here](#)

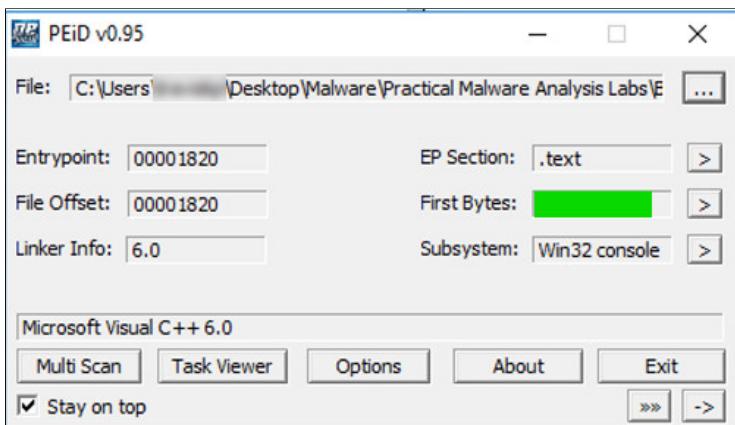
Extract all the files and run the **PEiD** application.

In PEid, in the "File" line, at the right side, click the ... button.

Goto the **Chapter_1L** folder and open the **Lab01-01.exe** file.

On the bottom left, you can see that this file was written in "Microsoft Visual C++", as shown below.

On the right side, note the "First Bytes", covered by a green box in the image below. That's the flag.



Flag PMA 101.3: BinText (5 pts)

BinText is a handy tool to view strings, a very easy and powerful way to analyze a file.

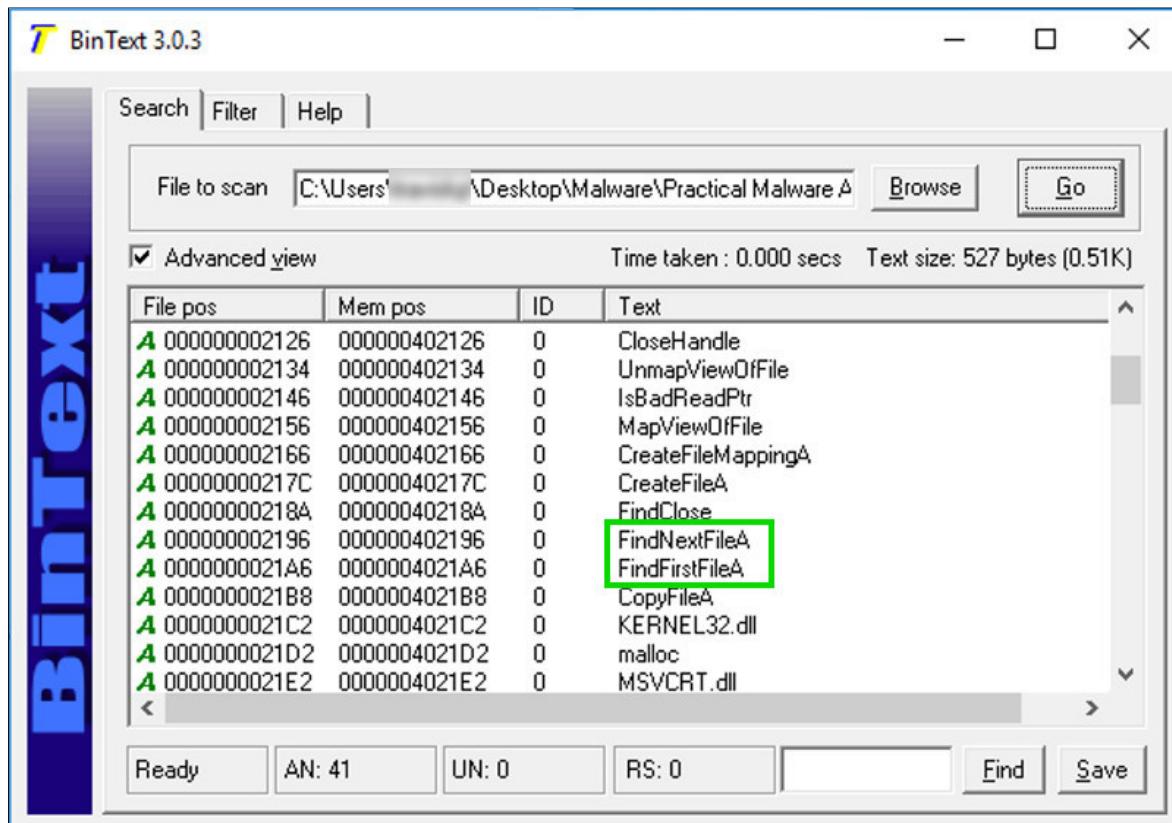
Download BinText from <http://b2b-download.mcafee.com/products/tools/foundstone/bintext303.zip>

Extract and open BinText and click Run at the Security prompt.

Click the **Browse** button.

Goto the **Chapter_1L** folder, select the **Lab01-01.exe** file and click **Go**.

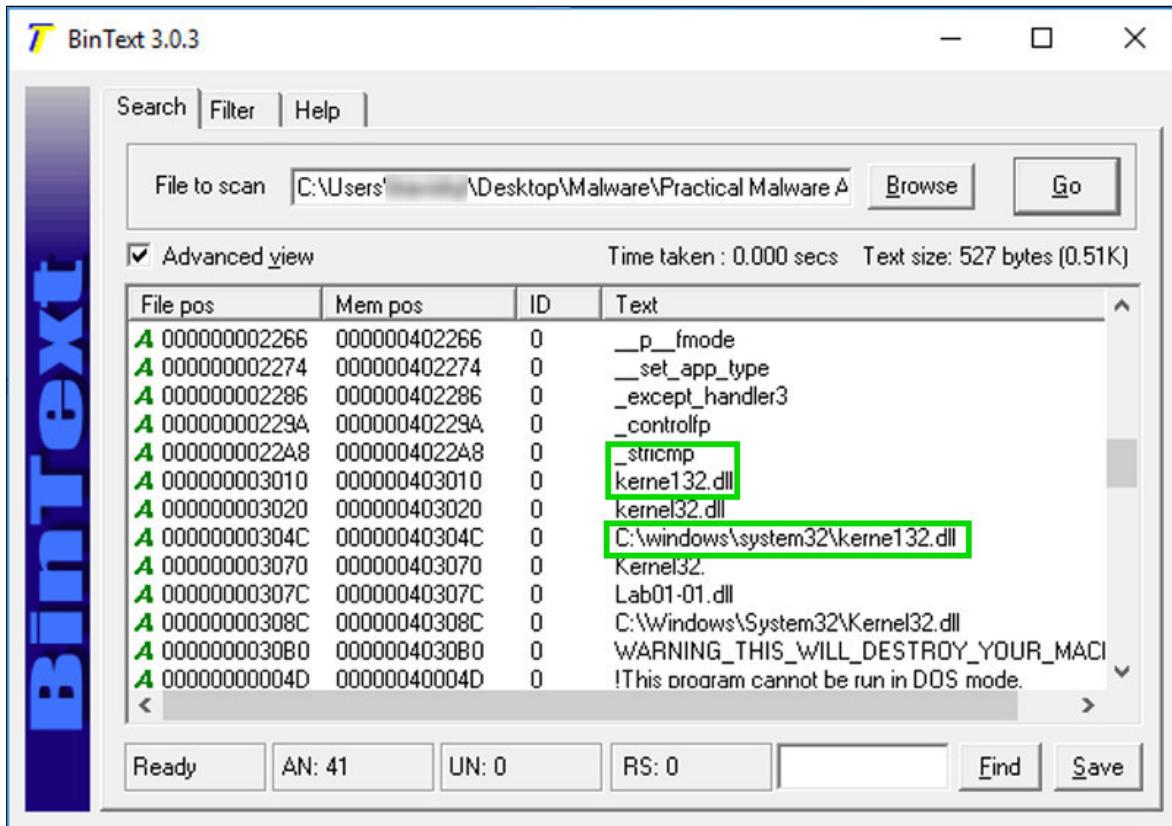
Notice **FindNextFileA** and **FindFirstFileA**, as shown below. These are Windows API functions used to search through a directory.



Scroll down and find these items, as shown below.

- **_strcmp** -- Compares a string to a desired value

- **kerne132.dll** -- A deceptive filename to make the malware look like a Windows system file
- **C:\windows\system32\kerne132.dll** -- The full path to a malicious file, very likely a useful Indicator of Compromise



In BinText, open the **Lab01-01.dll** file and click **Go**.

Notice these items, as shown below:

- **Sleep** -- Windows API function used to sleep
- **CreateProcessA** -- Windows API function used to launch a program
- **sleep** and **hello** -- Commands that can be sent over the network to tell the malware to sleep, and some function called "hello"

File pos	Mem pos	ID	Text
A 000000000210A	00001000210A	0	CloseHandle
A 0000000002118	000010002118	0	Sleep
A 0000000002120	000010002120	0	CreateProcessA
A 0000000002132	000010002132	0	CreateMutexA
A 0000000002142	000010002142	0	OpenMutexA
A 000000000214E	00001000214E	0	KERNEL32.dll
A 000000000215C	00001000215C	0	WS2_32.dll
A 000000000216A	00001000216A	0	strcmp
A 0000000002172	000010002172	0	MSVCRT.dll
A 0000000002188	000010002188	0	_initterm
A 0000000002194	000010002194	0	malloc
A 000000000219E	00001000219E	0	_adjust_fdiv
A 0000000026018	000010026018	0	sleep
A 0000000026020	000010026020	0	hello

The command to launch a program is missing. To see it, click the **Filter** tab and adjust the "Min. text length" to **4** as shown below.

STAGE 1: Characters included in the definition of a string

Character filters (checkboxes): CR, LF, Space, Tab, !, " (double quote), #, \$, %, &, ' (apostrophe), (,), *, +, . (comma), - (minus), . (period), /, 0-9, :, ;, <, >, =, ? (question mark), _ (underscore), ^, <, >, =, ?, @, {, }, |, ~ (tilde), ^ (circumflex), ` (backtick), a-z, a-e, ö, ü, ö, ü, ä, ö, ü.

STAGE 2: String size

Min text length: **4**, Max text length: 1024, Discard strings with 3 or more repeated characters.

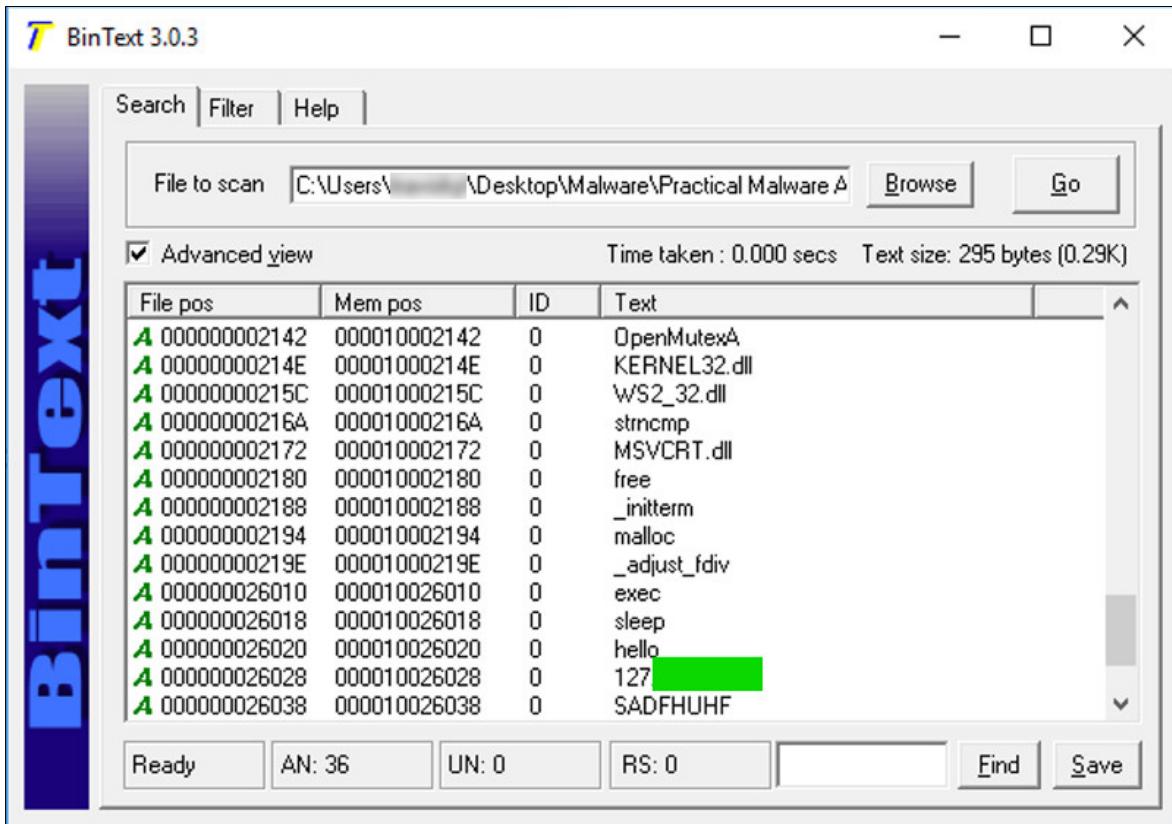
STAGE 3: Essentials

MUST contain these: (empty text field).

Click the **Search** tab. At the top right, click **Go**.

Now you can see that the command to launch a program is **exec**, as shown below.

Near the bottom, find the IP address beginning with **127**, covered by a green box in the image below. That's the flag.



Flag PMA 101.4: Dependency Walker (5 pts)

Go to <http://www.dependencywalker.com/> and click **Download Version 2.2.6000 for x64 [468k]**

Extract the files and open the **depends** application.

In Dependency Walker, click the opening folder icon and open the **Lab01-01.exe** file.

An error message will popup, but you can ignore it and click ok. Learn more about it at <https://stackoverflow.com/questions/33604738/dependency-walker-with-windows-10>

The top left pane is called the **"Module Dependency Tree View"**.

It shows the EXE file and the two Windows libraries it uses: MSVCRT.DLL and KERNEL32.DLL, as shown below (click the - next to KERNEL32.DLL).

In the top left pane, click **MSVCRT.DLL**. The top right pane shows **"Parent Imports"**. These are the functions the EXE file uses from the library.

As shown below, this executable uses only a small number of library functions, and none of them indicate much about its purpose. One of them is named **_strcmp**, which indicates that this program performs a string comparison, but that's a very common operation.

Dependency Walker - [Lab01-01]

File Edit View Options Profile Window Help

Module: LAB01-01.EXE

PI Ordinal Hint Function

?	N/A	111 (0x006F)	_p_fmode
?	N/A	129 (0x0081)	_set_app_type
?	N/A	131 (0x0083)	_setusermatherr
?	N/A	157 (0x009D)	_adjust_fdiv
?	N/A	183 (0x00B7)	_controlfp
?	N/A	202 (0x00CA)	_except_handler3
?	N/A	211 (0x00D3)	_exit
?	N/A	271 (0x010F)	_initterm
?	N/A	449 (0x01C1)	_stricmp
?	N/A	585 (0x0249)	exit
?	N/A	657 (0x0291)	malloc

File Time Stamp Link Time Stamp File Size Attr. Link Checksum Real Checksum CPU

?	Module	File Time Stamp	Link Time Stamp	File Size	Attr.	Link Checksum	Real Checksum	CPU
?	API-MS-WIN-CORE-APIQUERY-L1-1-0.DLL	Error opening file. The system cannot find the file specified (2).						
?	API-MS-WIN-CORE-APPCOMPAT-L1-1-0.DLL	Error opening file. The system cannot find the file specified (2).						
?	API-MS-WIN-CORE-COMM-L1-1-0.DLL	Error opening file. The system cannot find the file specified (2).						
?	API-MS-WIN-CORE-CONSOLE-L1-1-0.DLL	Error opening file. The system cannot find the file specified (2).						
?	API-MS-WIN-CORE-CONSOLE-L2-1-0.DLL	Error opening file. The system cannot find the file specified (2).						
?	API-MS-WIN-CORE-DATETIME-L1-1-0.DLL	Error opening file. The system cannot find the file specified (2).						

For Help, press F1

In the top left pane, click **KERNEL32.DLL**.

The top right pane shows that this file uses several functions that manipulate files, including **FindNextFileA** and **FindFirstFileA**, as shown below.

This suggests that the malware searches through the file system.

Dependency Walker - [Lab01-01]

File Edit View Options Profile Window Help

Module: LAB01-01.EXE

PI Ordinal Hint Function

?	N/A	27 (0x001B)	CloseHandle
?	N/A	40 (0x0028)	CopyFileA
?	N/A	52 (0x0034)	CreateFileA
?	N/A	53 (0x0035)	CreateFileMappingA
?	N/A	144 (0x0090)	FindClose
?	N/A	148 (0x0094)	FindFirstFileA
?	N/A	157 (0x009D)	FindNextFileA
?	N/A	437 (0x01B5)	IsBadReadPtr
?	N/A	470 (0x01D6)	MapViewOfFile
?	N/A	698 (0x0280)	UnmapViewOfFile

E Ordinal Hint Function

1 (0x0001)	0 (0x0000)	AcquireSRWLockExclusive
2 (0x0002)	1 (0x0001)	AcquireSRWLockShared
3 (0x0003)	2 (0x0002)	ActivateActCtx
4 (0x0004)	3 (0x0003)	ActivateActCtxWorker
5 (0x0005)	4 (0x0004)	AddAtomA

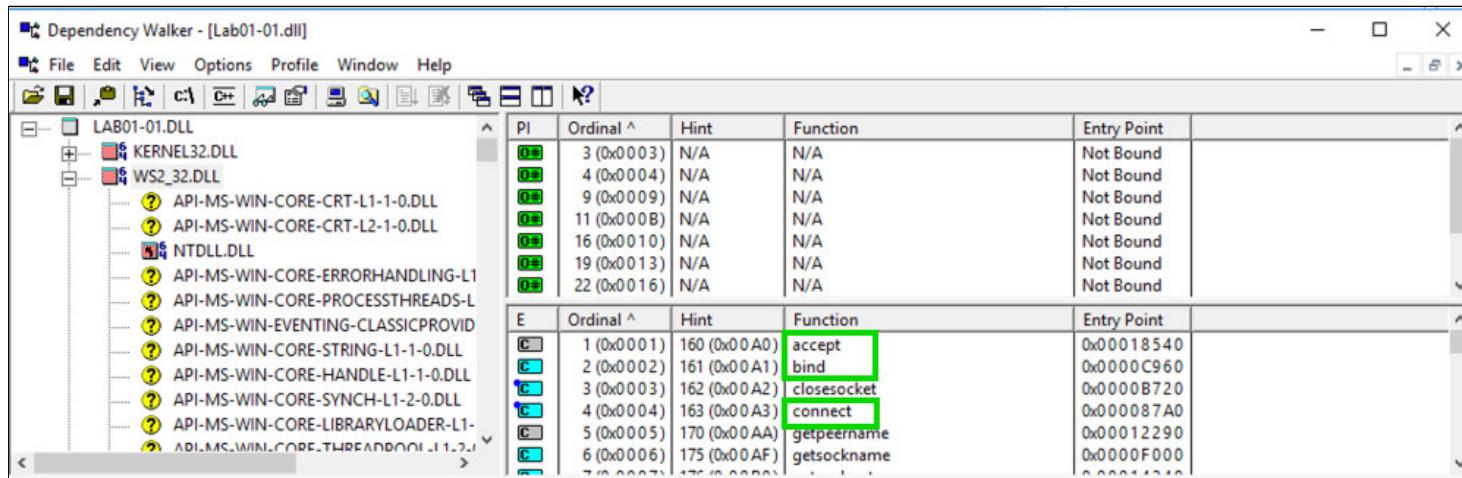
In Dependency Walker, open **Lab01-01.dll**.

In the top left pane, partially collapse the tree to match the image below and click **WS2_32.DLL**.

The top right pane doesn't show function names this time, it only shows "Ordinal" numbers. This is called [Linking by Ordinal](#), and it's an annoyance to us because we can't easily see what functions are in use.

Click **WS3_32.DLL** in the top left pane.

The center-right pane shows the **Exports** of WS2_32.DLL, which include **accept**, **bind**, and **connect**. These are the standard [Berkeley Sockets](#) functions used for networking. This suggest that the malware performs some networking functions, such as connecting to a server and opening a listening port.

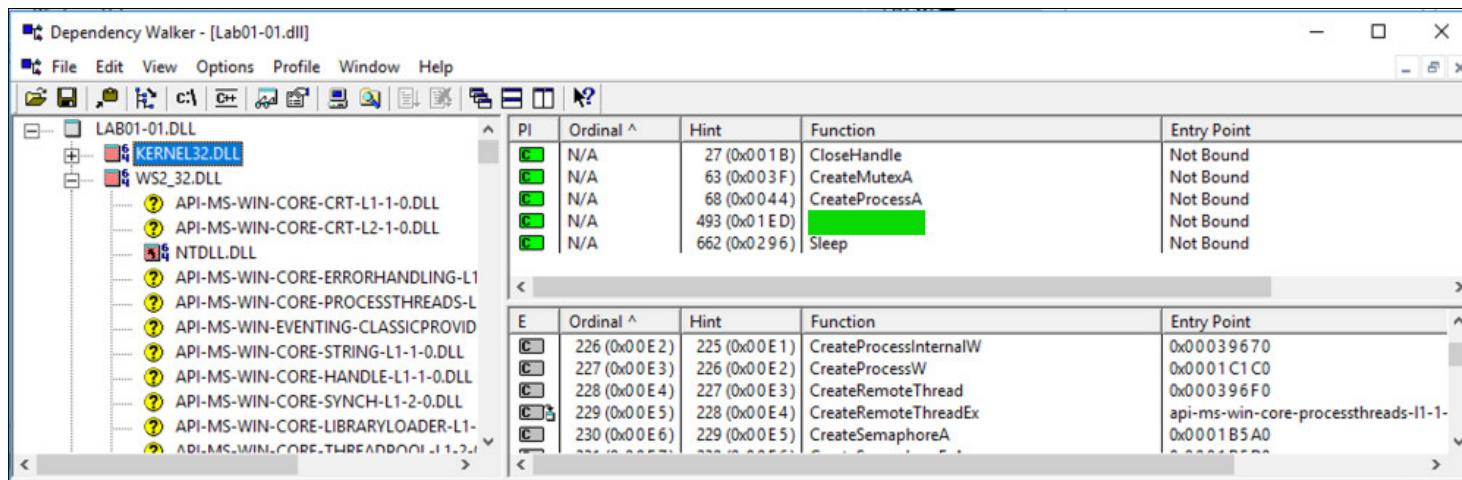


PI	Ordinal ^	Hint	Function	Entry Point
0x0	3 (0x0003)	N/A	N/A	Not Bound
0x0	4 (0x0004)	N/A	N/A	Not Bound
0x0	9 (0x0009)	N/A	N/A	Not Bound
0x0	11 (0x000B)	N/A	N/A	Not Bound
0x0	16 (0x0010)	N/A	N/A	Not Bound
0x0	19 (0x0013)	N/A	N/A	Not Bound
0x0	22 (0x0016)	N/A	N/A	Not Bound

E	Ordinal ^	Hint	Function	Entry Point
1	1 (0x0001)	160 (0x0A0)	accept	0x0018540
2	2 (0x0002)	161 (0x0A1)	bind	0x0000C960
3	3 (0x0003)	162 (0x0A2)	closesocket	0x0000B720
4	4 (0x0004)	163 (0x0A3)	connect	0x000087A0
5	5 (0x0005)	170 (0x0AA)	getpeername	0x0012290
6	6 (0x0006)	175 (0x0AF)	getsockname	0x0000F000
7	7 (0x0007)	176 (0x0B0)		0x0011410

In the top left pane, click **KERNEL32.DLL**. The top right pane shows the five "Parent Imports", which include **CreateProcessA** and **Sleep**, as shown below.

Find the function name that is covered by a green box in the image below. That's the flag.



PI	Ordinal ^	Hint	Function	Entry Point
0x0	N/A	27 (0x01B)	CloseHandle	Not Bound
0x0	N/A	63 (0x03F)	CreateMutexA	Not Bound
0x0	N/A	68 (0x044)	CreateProcessA	Not Bound
0x0	N/A	493 (0x1ED)		Not Bound
0x0	N/A	662 (0x296)	Sleep	Not Bound

E	Ordinal ^	Hint	Function	Entry Point
1	226 (0x0E2)	225 (0x0E1)	CreateProcessInternalW	0x0039670
2	227 (0x0E3)	226 (0x0E2)	CreateProcessW	0x0001C1C0
3	228 (0x0E4)	227 (0x0E3)	CreateRemoteThread	0x000396F0
4	229 (0x0E5)	228 (0x0E4)	CreateRemoteThreadEx	api-ms-win-core-processsthreads-l1-1-0x0001B5A0
5	230 (0x0E6)	229 (0x0E5)	CreateSemaphoreA	0x0001B5A0
6	231 (0x0E7)	229 (0x0E5)		0x0001B5A0

Flag PMA 101.5: Find the Downloaded File (10 pts extra credit)

Analyze the sample **Lab01-04.exe**

It downloads a file from this domain: practicalmalwareanalysis.com

Find that file's name. That's the flag.

Flag PMA 101.6: Find the Imported Function (10 pts extra credit)

Analyze the sample **Lab01-04.exe**

It imports a function from **WINTRUST.DLL**

Find that function's name. That's the flag.

Flag PMA 101.7: Find the Datestamp (10 pts extra)

Find the date when sample **Lab01-04.exe** was compiled, like this: **2000/01/01**. That's the flag.

References

For more information about using Dependency Walker, see this tutorial:

[Analyzing dependencies with Dependency Walker](#)

Ported to Google Cloud 9-4-19

Renumbered and ported to new flag system 8-14-19