Lab 1 part 2: Basic Static Techniques (5%)

What you need:

- The Malware Analysis Virtual Machine you prepared in a previous project

Malware Samples

This project uses two files in this folder:

C:\Users\Administrator\Desktop\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.

Flag PMA 101.1 PEview (5 pts)

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

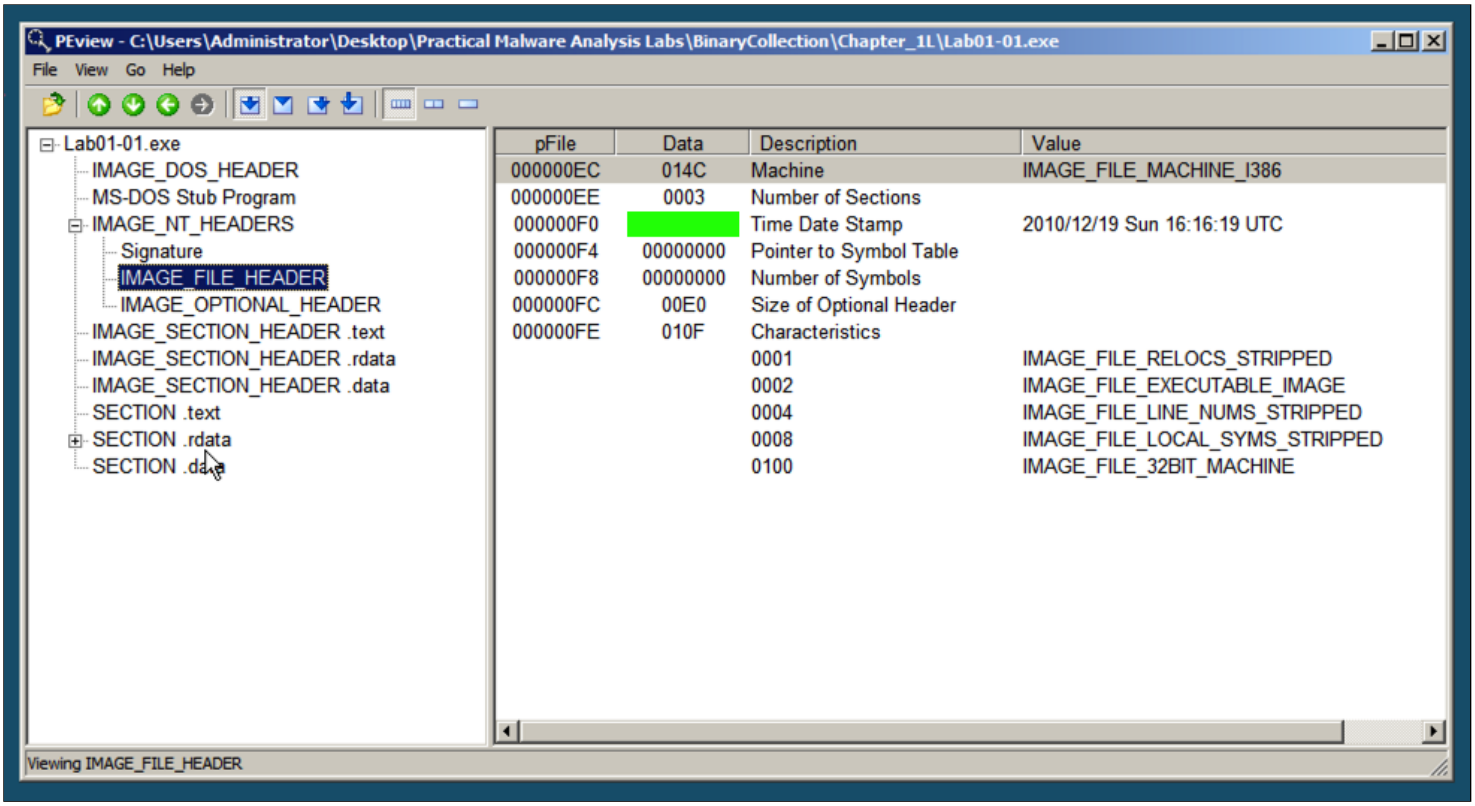
Click **Start** and type **PEVIEW**

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

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.



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.

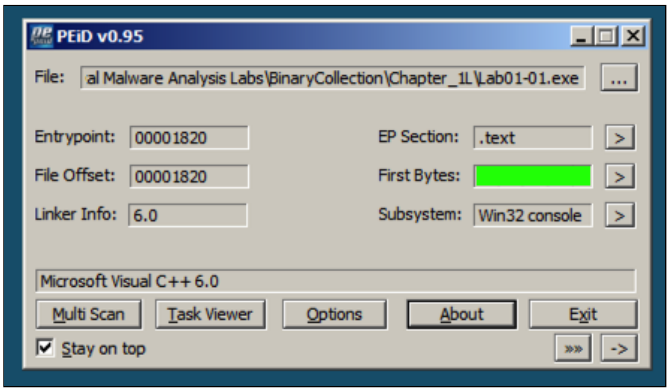
Click **Start** and type **PEID**

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

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)

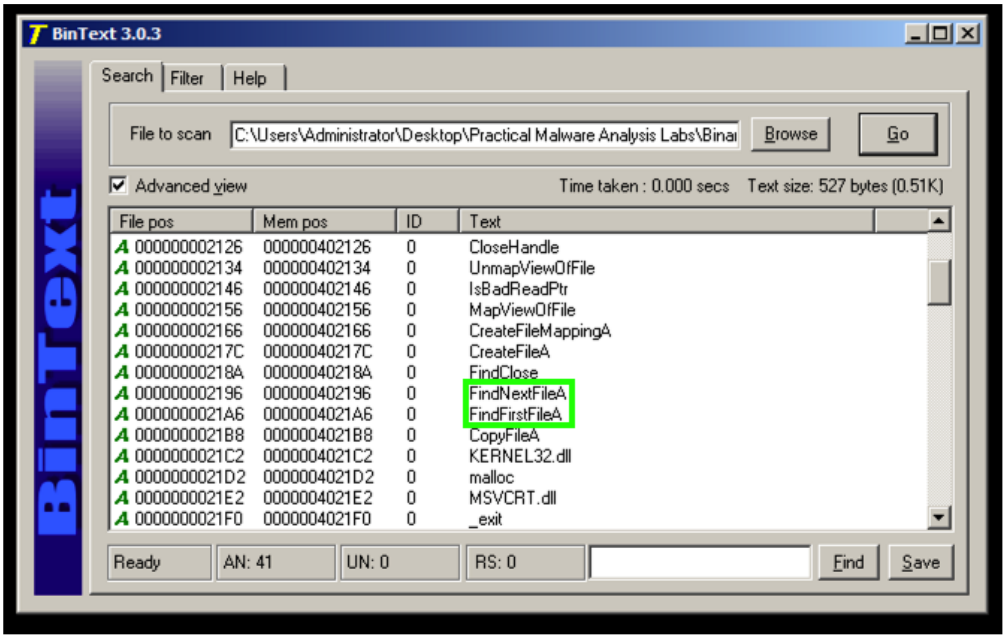
BinText is a handy tool to view strings, a very easy and powerful way to analyze a file.

Click **Start** and type **BinText**

Launch BinText. In BinText, click the **Browse...** button.

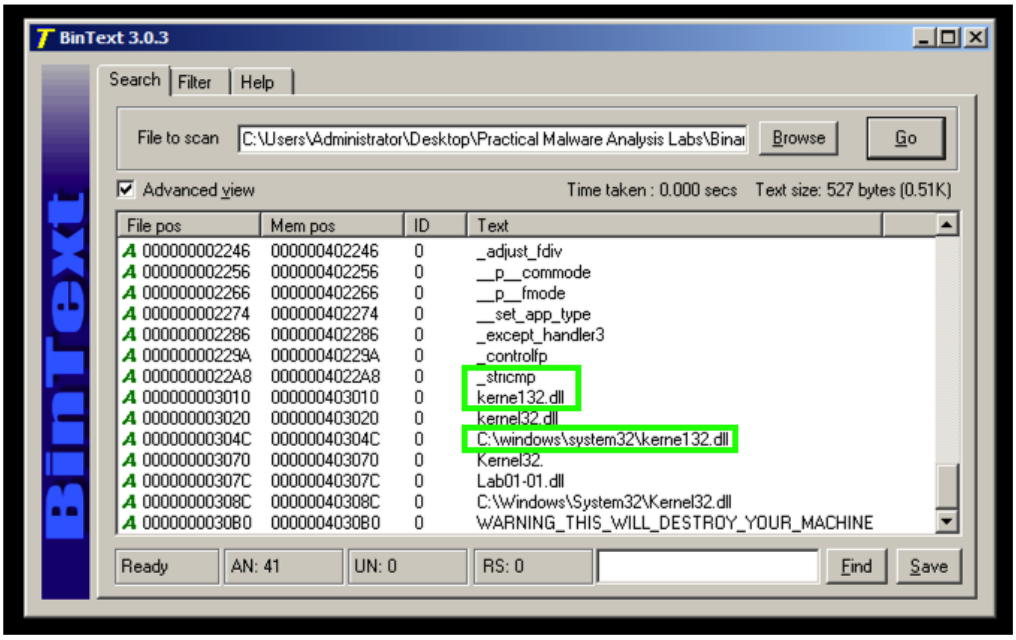
Open the **Lab01-01.exe** file and click **Go**.

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.

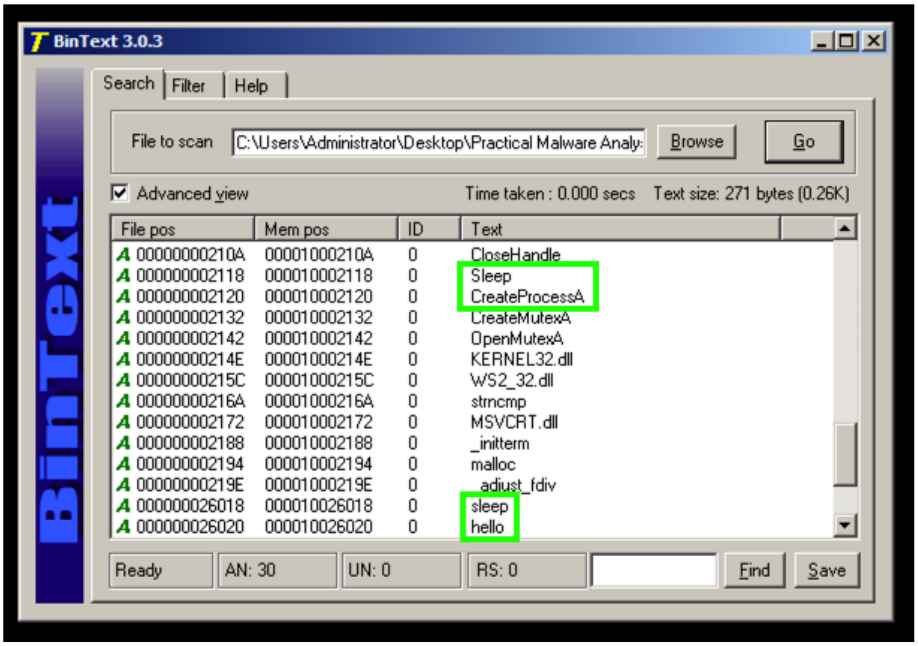
- **_stricmp** -- Compares a string to a desired value
- **kerne132.dll** -- A deceptive filename to make the malware look like a Windows system file
- **C:\windows\system32\kerne132.dll** -- The full path to a malicious file, very likely a useful Indicator of Compromise



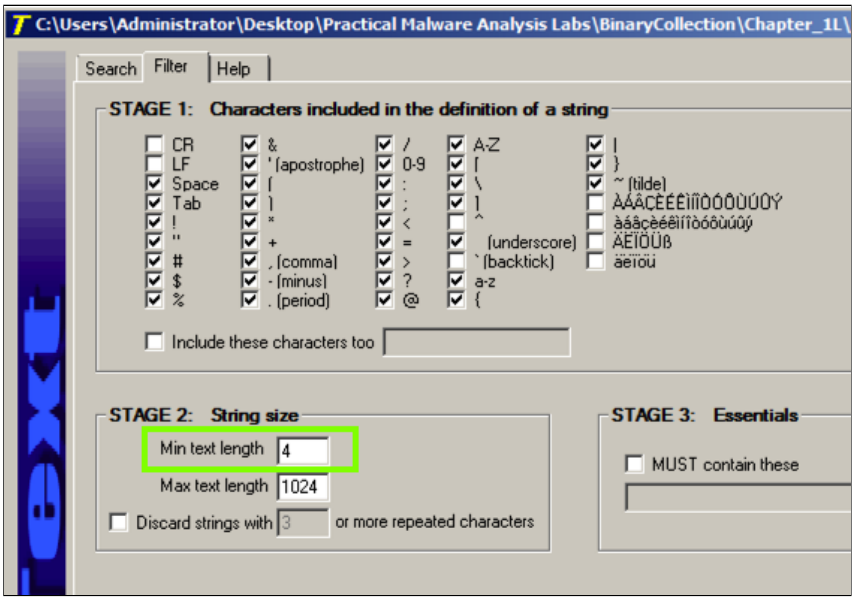
In BinText, open the **Lab01-01.dll** file and click **Go**.

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"



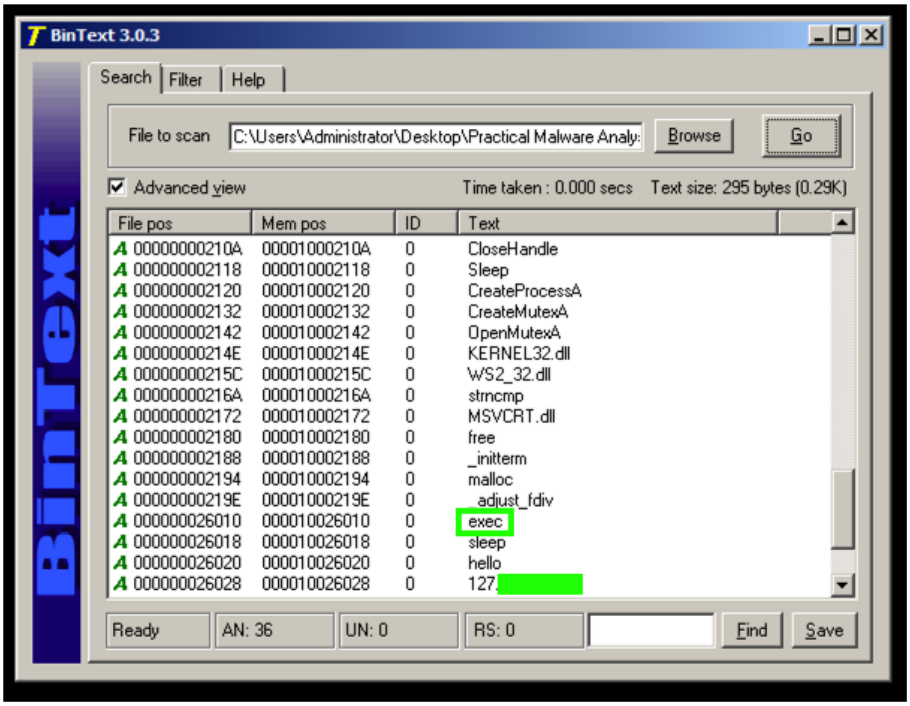
The command to launch a program is missing. To see it, click the **Filter** tab and adjust the "Min. text length" to **4** as shown below.



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)

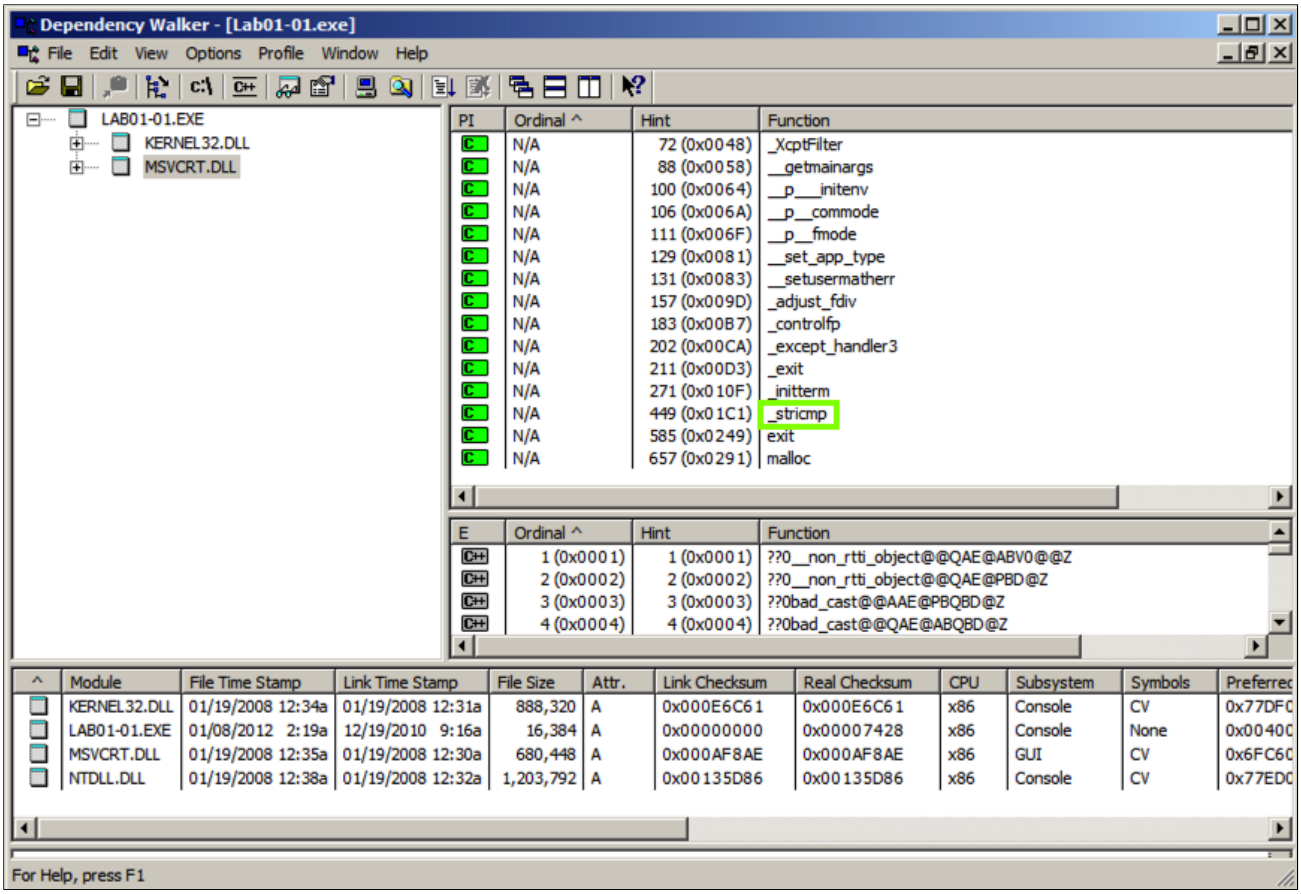
Click **Start** and type **Dependency**

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

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.

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.

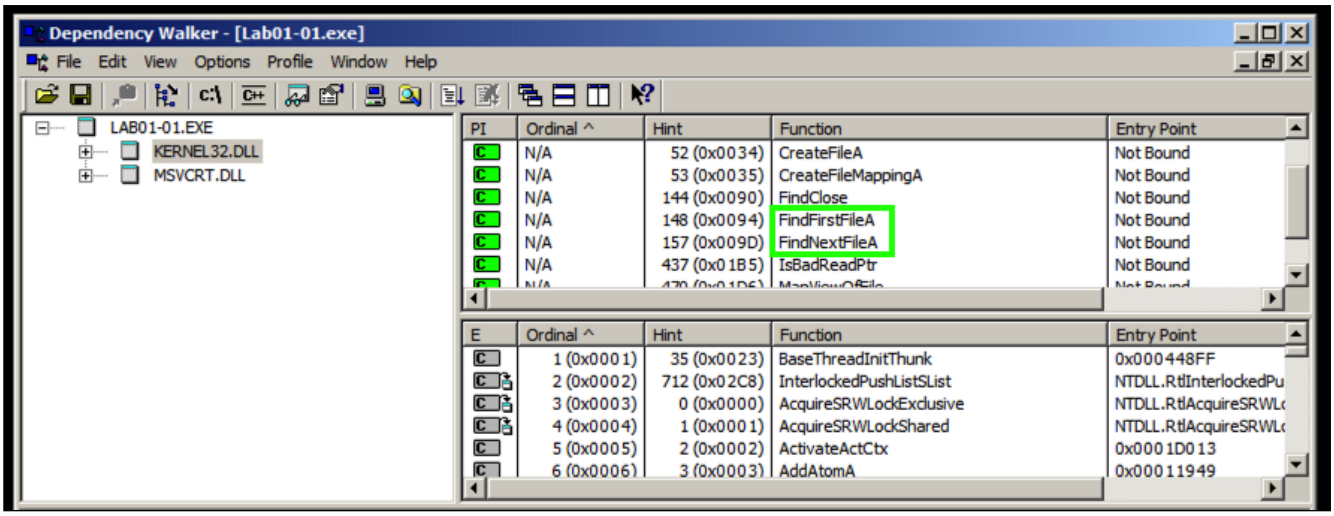
As shown below, this executable uses only a small number of library functions, and none of them indicate much about its purpose. One of them is named **_stricmp**, which indicates that this progam performs a string copmarison, but that's a very common operation.



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.

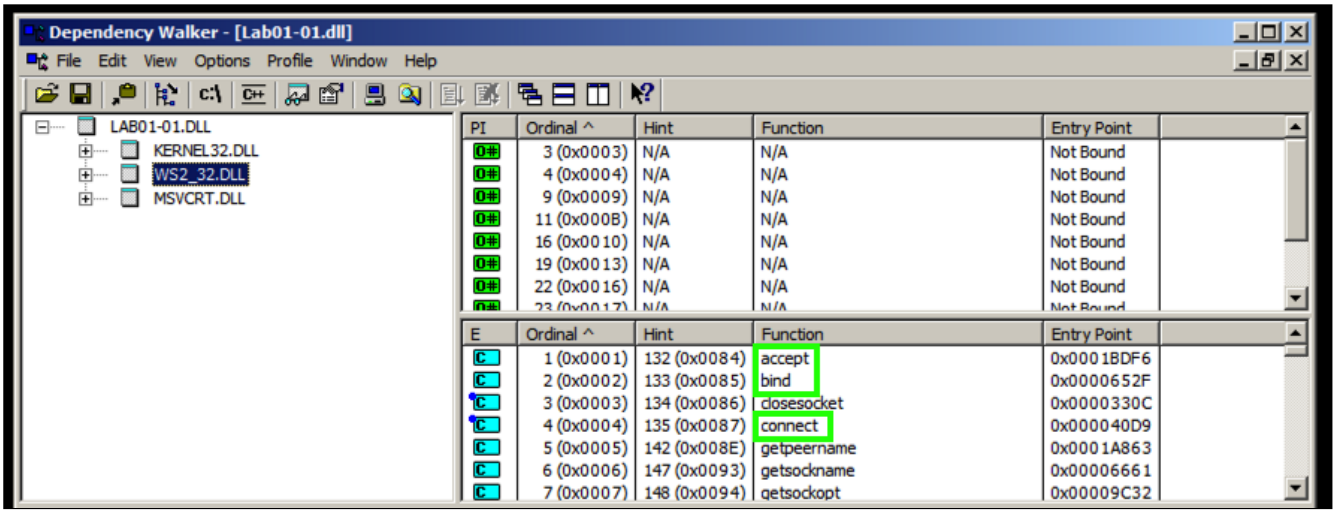


In Dependency Walker, open **Lab01-01.dll**.

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

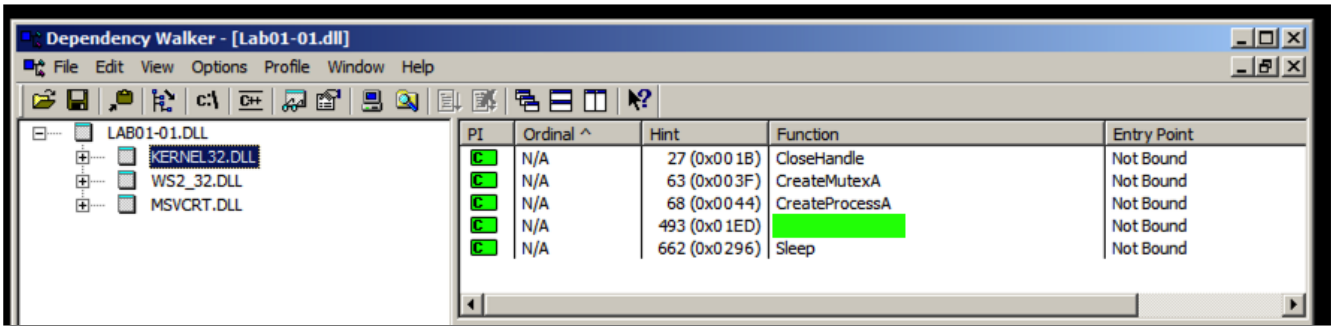
The top right pane doesn't show function names this time, it only shows "Ordinal" numbers. This is called Linking by Ordinal, and it's an annoyance to us because we can't easily see what functions are in use.

However, 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.



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.



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 ths tutorial:

Analyzing dependencies with Dependency Walker

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