

SAT1: 012: Static Malware Analysis

Overview

Anti-virus can be very helpful in analyzing malware. Typically, most companies are only utilizing one anti-virus technology. In our case, we want to be able to look at multiple signatures from multiple vendors at one time. Virustotal is an excellent resource for this. As you remember, in the traffic malware analysis we used virustotal to look at the Zahlung17.doc file to find file relationships. Now, we will use Virustotal to look at anti-virus signatures.

Time: 30 Minutes

Learning Objectives

Upon completion of this lab, you should be able to:

Log in to the Lab Machine

Select the **Windows 10** machine on the Machines Tab.



Select the → on the **Windows 10** machine and

click on the on the **Machines Tab**, and press **Enter** to log in to the machine.

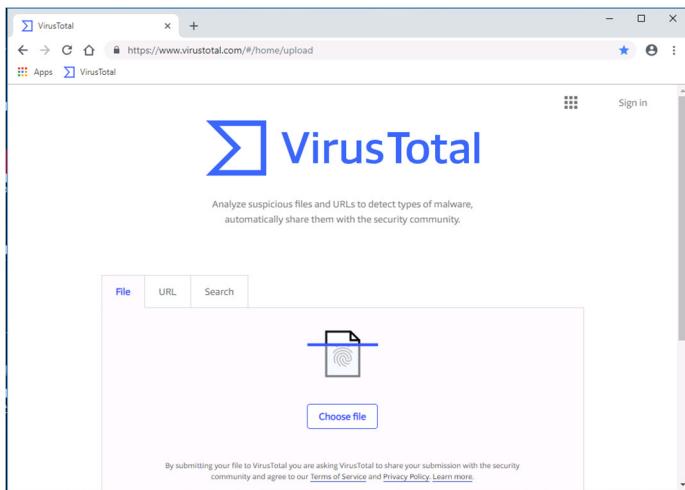




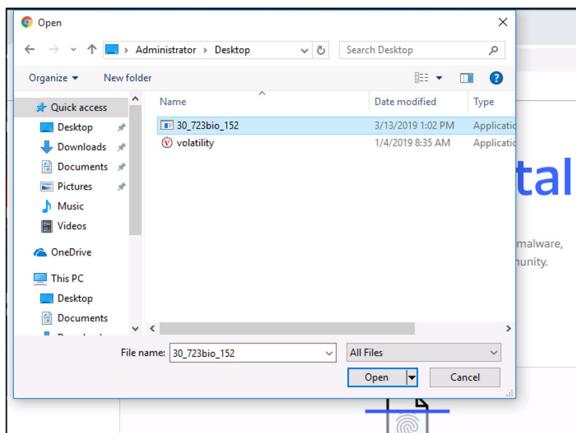
1.0 Identifying Malware

1.1 Open a web browser or new tab in your current browser.

1.2 Navigate to <https://www.virustotal.com>



1.3 Select **Choose File** then navigate to the malware from the previous exercise (it is on the Desktop).



Note: Since this is known malware, we get a large number of anti-virus hits from VirusTotal. But minimal effort on the part of the attacker could change the signature and make it undetected. In this case, we could look at the hash.



The screenshot shows the VirusTotal analysis interface. At the top, it says "52 engines detected this file". Below that, there's a summary table with the following data:

SHA-256	a56876fd456d0737eecc4a8bbe3154b35314ab28accc29abf0df7c518c8...
File name	a56876fd456d0737eecc4a8bbe3154b35314ab28accc29abf0df7c518c8...
File size	60 KB
Last analysis	2019-02-25 08:25:01 UTC
Community score	-128

Below the summary is a table of detection results:

Detection	Details	Relations	Behavior	Community
Acronis	suspicious			
Ad-Aware	Gen:Variant.Feedel.1			
AegisLab	Trojan.Win32.Zonidel.4!c			
AhnLab-V3	Trojan/Win32.Zonidel.C2224358			
ALYac	Trojan.Ransom.LockyCrypt			
Arcabit	Trojan.Feedel.1			

What is Hashing?

Among the most critical tasks that we need to do when acquiring an image is to ensure its integrity. In essence, we want to be able to prove in a court of law or other venue that the image we used for analysis was not tampered with or in any way changed since we acquired it.

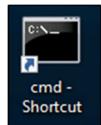
You can only imagine a defense attorney or other representative who will argue that any evidence that you have found on the computer was placed there by law enforcement or the forensic investigator.

Hashing is one-way encryption that creates a unique output (digest) for any input. Hashing is used to assure that nothing changes in the original input. If even a single bit changes in the original input, the hash will change.

You have probably seen or used hashes when you downloaded software. In fact, when you downloaded Kali, Offensive Security provides you the MD5 hash of it so that you can check that the Kali you downloaded has not been corrupted or otherwise altered in any way before it gets to you.

2.0 Hashing a Malware File

2.1 Open a **Command** prompt.





2.2 Type `cd C:\\\\Users\\\\Administrator\\\\Desktop`

```
Administrator: cmd prompt
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\\Windows\\System32>cd C:\\\\Users\\\\Administrator\\\\Desktop

C:\\Users\\\\administrator\\\\Desktop>
```

2.3 Type `CertUtil -hashfile 30_723bio_152.exe MD5`

```
Administrator: cmd prompt
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\\Windows\\System32>cd C:\\\\Users\\\\Administrator\\\\Desktop

C:\\Users\\\\administrator\\\\Desktop>CertUtil -hashfile 30_723bio_152.exe MD5
MD5 hash of file 30_723bio_152.exe:
65 8c 30 fc d1 50 8a 65 df 8b 9b 9b 39 7a 94 59
CertUtil: -hashfile command completed successfully.

C:\\Users\\\\administrator\\\\Desktop>
```

We could then put this hash in VirusTotal or another web based program to see if the hash is a known bad hash. Another way to obtain data from malware is to look at the strings. A program contains strings if it prints a message, connects to a URL, or copies a file to a specific location.



Note: The most popular hashes are MD5, SHA1, SHA256, and SHA512. As we will see, we can use any of these to ensure the integrity of our forensic image when we use dcfldd or other image acquisition tools.

3.0 Finding Strings in a File

3.1 Go back to the **Command** prompt, or open a new one.

3.2 Type `cd C:\\\\Users\\\\Administrator\\\\Desktop`, if you opened up a new **Command** prompt.

3.3 Type `strings 30_723bio_152.exe | more`

Note: A program contains strings if it prints a message, connects to a URL, or copies a file to a specific location.

One great indication that this program is an executable is `!This` program cannot be run `in DOS mode`, which you can see here.

```
Administrator: cmd prompt

Strings v2.53 - Search for ANSI and Unicode strings in binary images.
Copyright (C) 1999-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

!This program cannot be run in DOS mode.
bJ8
Richk2
.text
.data
<"&!78&
9%9+
)366%
"56(
\J|SSFVHEb_FPKO
r^n^\\AGLAUG1FOCDL
QT%
]@3
xF^Y_JDyTHP@AWeY
rBVSAQgDV[^INK
0.*-;8.
pSAgJAEURzTN^YMWU_r
[ DseY}trnr
mqtUvhedsp
cD_qZYGCM]xH^
bASvSO@cIHKZ[
Template
Resource
Language
-- More --
```

Note: Observe the `--More--` at the bottom left hand corner, by hitting the space bar, you can scroll through each line of text, without this command the entire file would fly by on the screen. Use the up arrows to bring up your last command and remove the `| More` to see.

3.4 Scroll through the rest of the file using **Strings** and see what you can find.

It doesn't appear as though there are any call outs that indicate an external web address, copied files, specific locations, or messages.



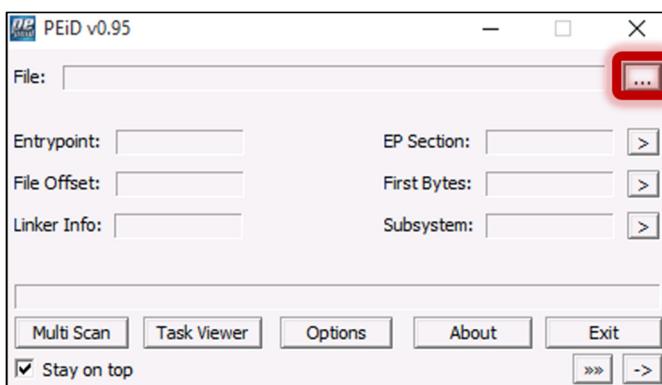
4.0 Detecting Packed Malware

Sometimes the malware is packed or obfuscated. This can be detected with PEiD.

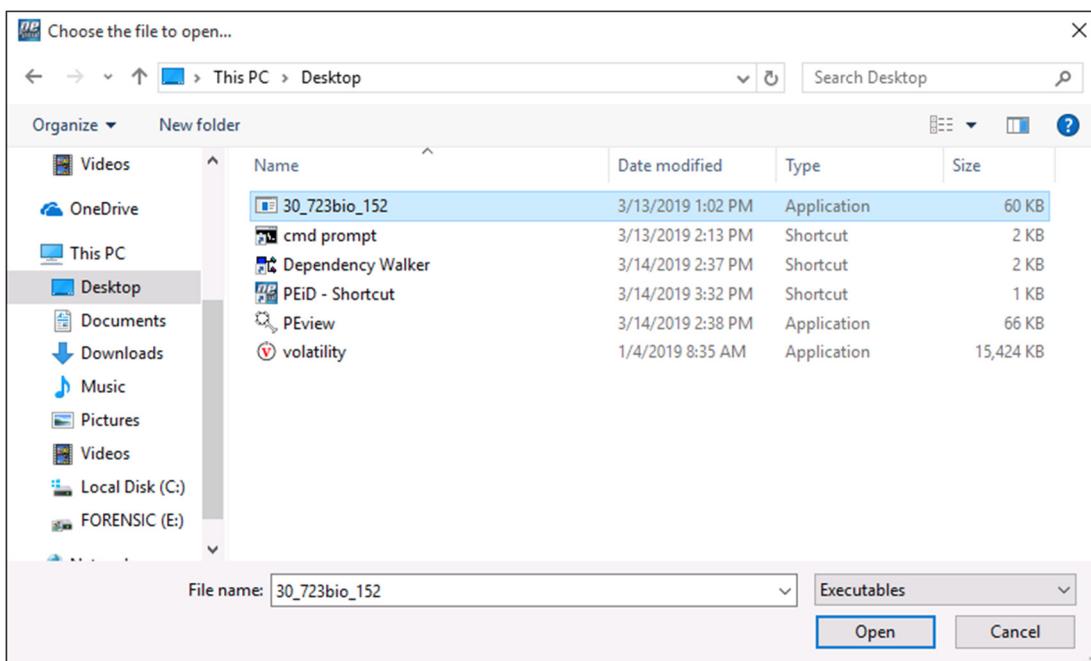
4.1 Open PEiD.



4.2 Click the three dots to browse for a file.



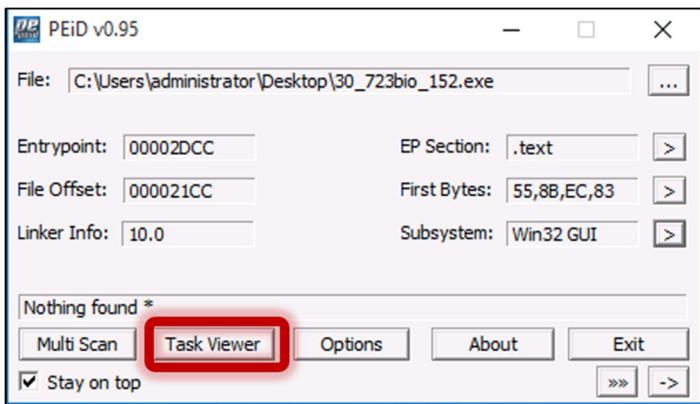
4.3 Select the malware file on the desktop.



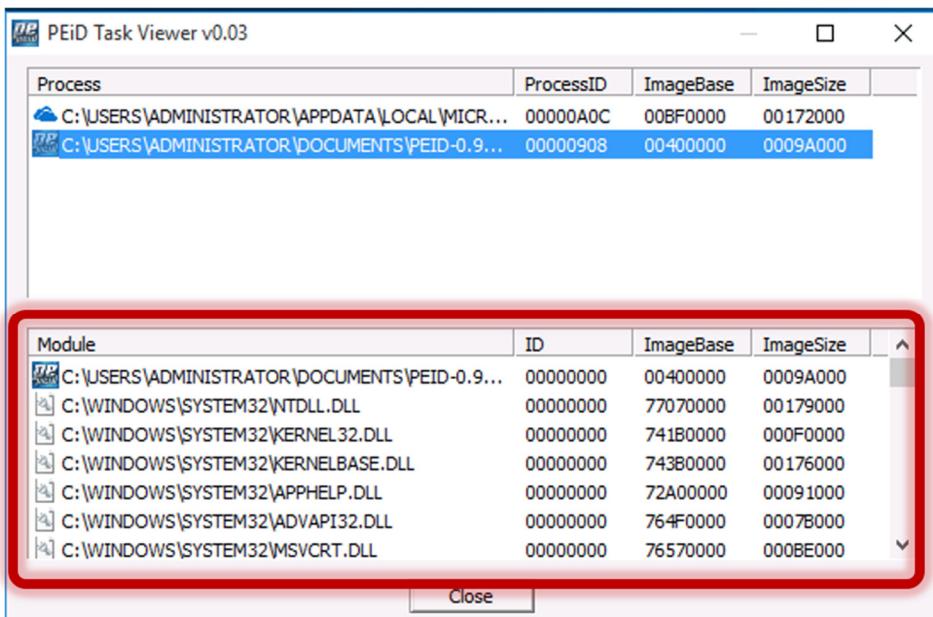
WARNING: Many PEiD plug-ins will run the malware executable without warning! Please take care and use this program in a virtual environment.



4.4 Click **Task Viewer**.



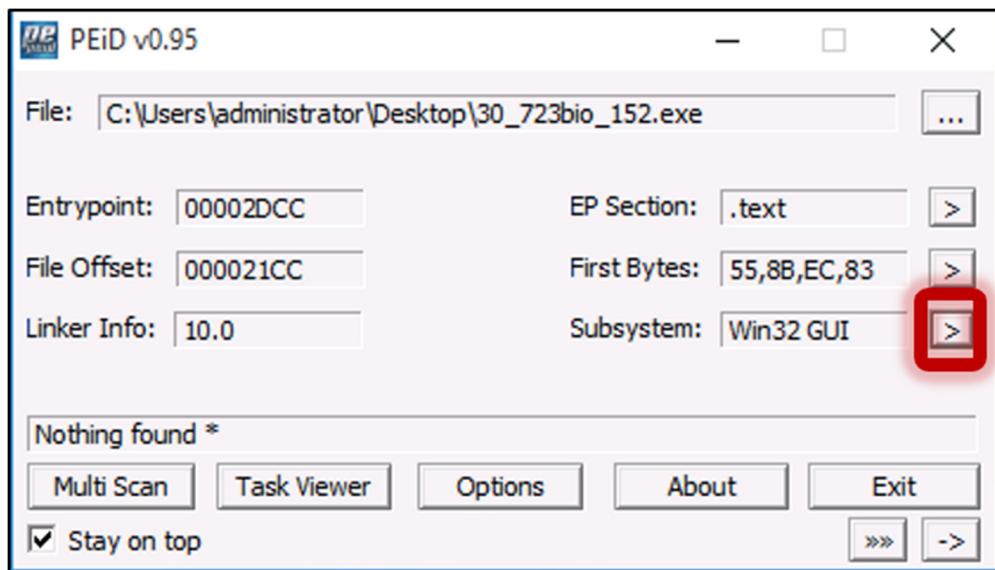
4.5 Click on the process you would like to analyze.



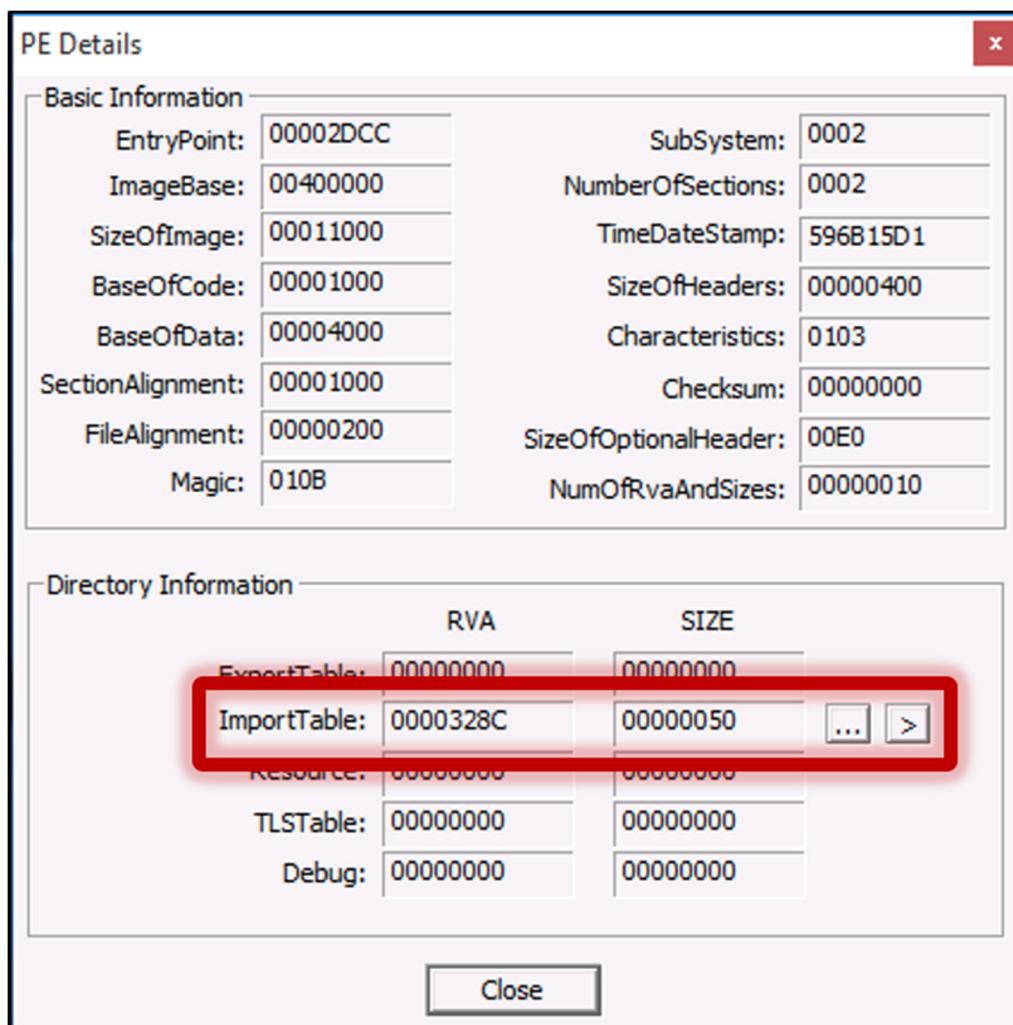
4.6 In the module section, we can see many different .dll files that are accessed.

4.7 Click **Close**.

4.8 Click the arrow next to Subsystem.



4.9 Click the arrow next to Import Table.





4.10 The Imports Viewer shows us the three DLLs being imported. We will take a deeper look at these later.

Imports Viewer						
DllName	OriginalFirstThunk	TimeDateStamp	ForwarderChain	Name	FirstThunk	
ADVAPI32.dll	000032DC	00000000	00000000	00003334	00001000	
KERNEL32.dll	000032E4	00000000	00000000	000033FC	00001008	
USER32.dll	00003314	00000000	00000000	00003446	00001038	

Thunk RVA	Thunk Offset	Thunk Value	Hint/Ordinal	API Name	

4.11 Close the open Windows and exit **PEiD**.

In this case, we are lucky and the malware does not appear to have a packer or be obfuscated. However, if it were packed we would have to unpack the program before we can perform analysis.

WARNING: Many PEiD plug-ins will run the malware executable without warning! Please take care and use this program in a virtual environment.



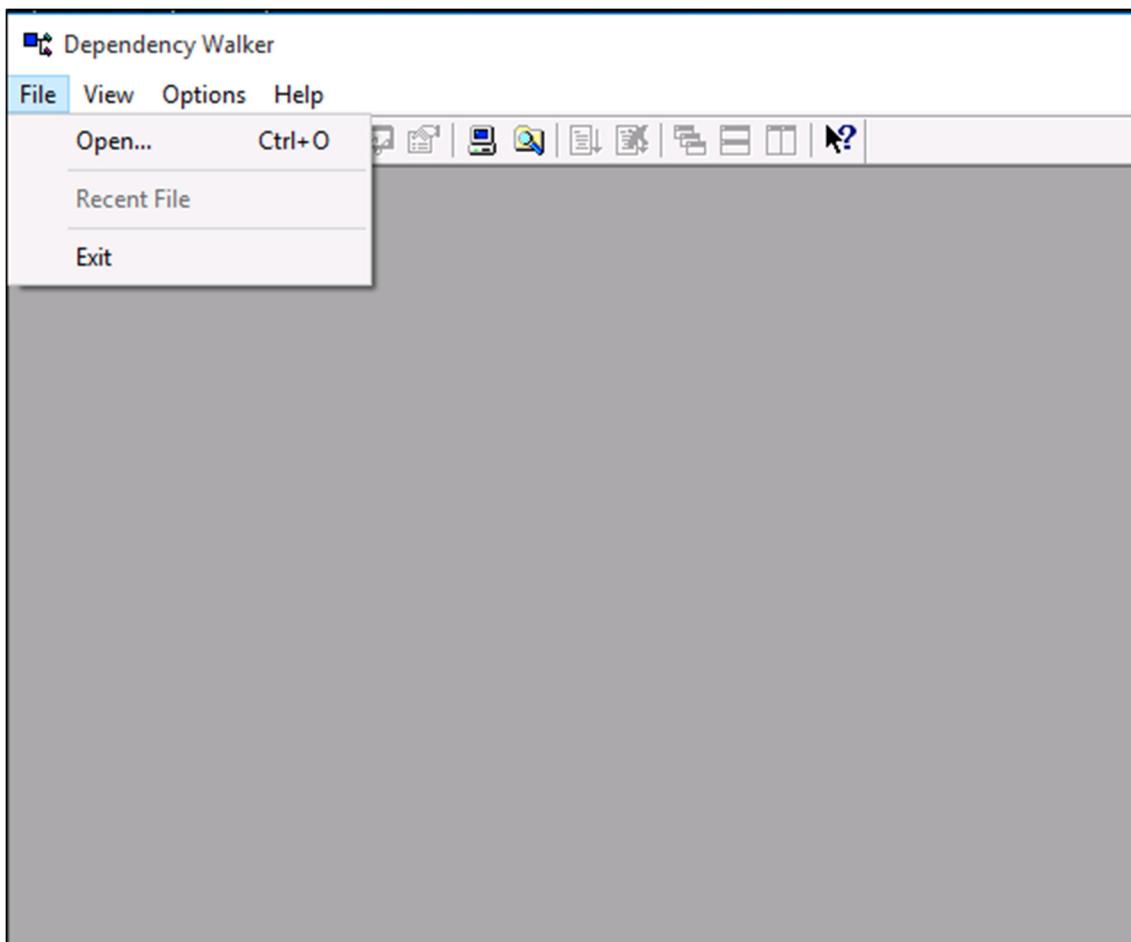
5.0 Analyzing Malware

5.1 Open **Dependency Walker**.

Note: Dependency Walker will create an error message that is fine, the software has been around for a while, and hasn't been updated to run well with Windows 10.

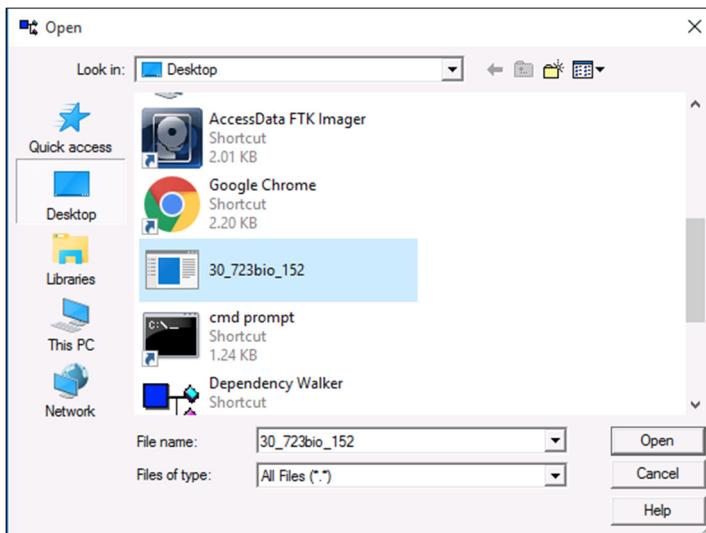


5.2 Open the malware file on the Desktop by clicking on **File** and clicking on **Open**.

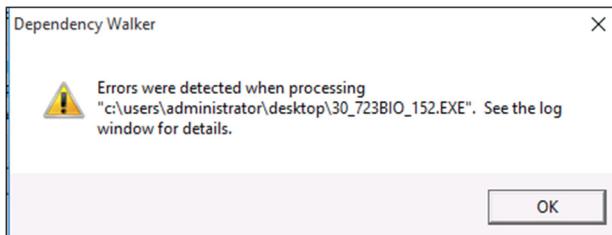




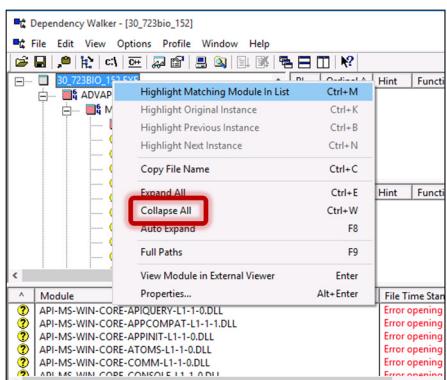
5.3 Navigate to the file using the popup window.



Note: If you get an error message, just disregard the error message. This malware spans across many versions of Windows and the version it was tested on utilizes this dll but it doesn't exist in later versions of Windows.

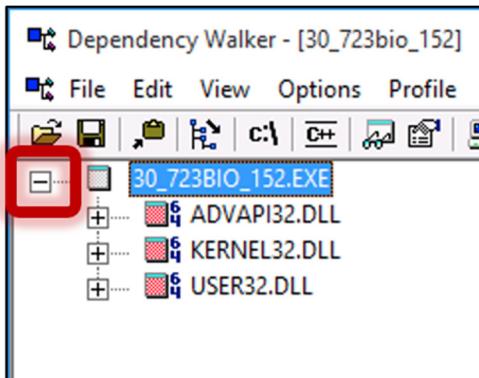


5.4 Right-click the file and select **Collapse All**. This will collapse the view, which will allow us to expand as we analyze.





5.5 Click the '+' next to the malware name to expand once.



5.6 Click on the name **Kernel32.dll**.

PI	Ordinal ^	Hint	Function	Entry Point
0	N/A	155 (0x009B)	CreateMutexA	Not Bound
0	N/A	211 (0x00D3)	DeleteFileA	Not Bound
0	N/A	446 (0x01BE)	GetCurrentDirectoryA	Not Bound
0	N/A	612 (0x0264)	GetStdHandle	Not Bound
0	N/A	623 (0x026F)	GetSystemDirectoryA	Not Bound
0	N/A	675 (0x02A3)	GetVersionExA	Not Bound

E	Ordinal ^	Hint	Function	Entry Point
0	1 (0x0001)	0 (0x0000)	AcquireSRWLockExclusive	NTDLL.RtlAcquireSRWLockExclusive
0	2 (0x0002)	1 (0x0001)	AcquireSRWLockShared	NTDLL.RtlAcquireSRWLockShare
0	3 (0x0003)	2 (0x0002)	ActivateActCtx	0x00020FEO
0	4 (0x0004)	3 (0x0003)	ActivateActCtxWorker	0x00016F80
0	5 (0x0005)	4 (0x0004)	AddAtomA	0x00023A70

Module	File Time Stamp	Link Time Stamp	File Size	Attr.	Link Checksum	Real Checksum
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-L1-1-1.DLL	Error opening file. The system cannot find the file specified (2).					
API-MS-WIN-CORE-APPINIT-L1-1-0.DLL	Error opening file. The system cannot find the file specified (2).					
API-MS-WIN-CORE-ATOMS-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).					

For Help, press F1

If you look at the first function you will notice it is **CreateMutexA**. If this executable is creating a Mutex A, then it is likely it will create a Mutex B; thus, looking for multiple processes is a good idea.

PI	Ordinal ^	Hint	Function	Entry Point
0	N/A	155 (0x009B)	CreateMutexA	Not Bound
0	N/A	211 (0x00D3)	DeleteFileA	Not Bound
0	N/A	446 (0x01BE)	GetCurrentDirectoryA	Not Bound
0	N/A	612 (0x0264)	GetStdHandle	Not Bound
0	N/A	623 (0x026F)	GetSystemDirectoryA	Not Bound
0	N/A	675 (0x02A3)	GetVersionExA	Not Bound

5.7 Close **Dependency Viewer**.

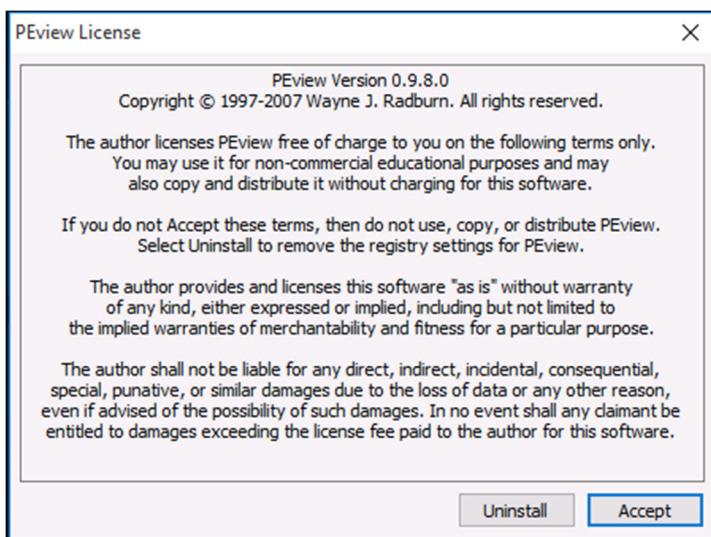


6.0 Analyzing Malware II

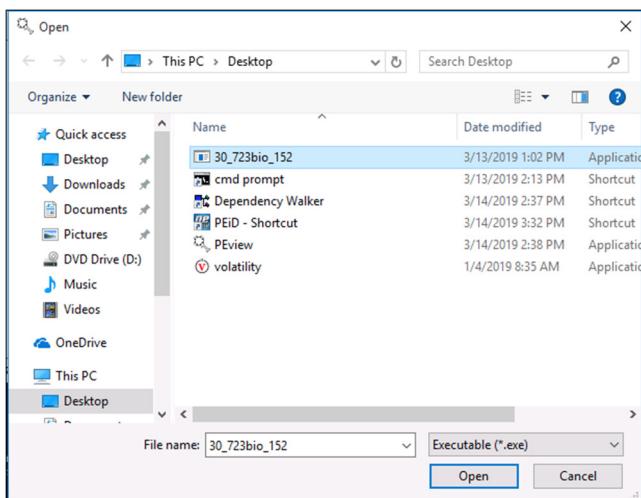
6.1 Open PE View.



Accept the License Agreement.



6.2 Open the malware file.





6.3 Expand the **IMAGE_NT_HEADERS** section.

The screenshot shows the PEview interface with the file '30_723bio_152.exe' open. The left pane displays the file structure with the 'IMAGE_NT_HEADERS' section expanded. The right pane shows the raw data for each field in the IMAGE_NT_HEADERS structure, including fields like pFile, Raw Data, and Value. The 'Value' column contains hex values and some ASCII characters.

pFile	Raw Data	Value
000000D0	50 45 00 00 4C 01 02 00	PE...kY...
000000E0	00 00 00 00 E0 00 03 01&
000000F0	00 C6 00 00 00 00 00 00	CC 2D 00 00 10 00 00
00000100	00 40 00 00 00 40 00 00	00 10 00 00 02 00 00
00000110	05 00 01 00 01 00 00 00	05 00 01 00 00 00 00
00000120	00 10 01 00 00 04 00 00	00 00 00 00 02 00 00
00000130	00 00 10 00 00 10 00 00	00 00 10 00 00 10 00 00
00000140	00 00 00 00 10 00 00 00	00 00 00 00 00 00 00
00000150	8C 32 00 00 50 00 00 00	00 00 00 00 00 00 00
00000160	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00
00000170	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00
00000180	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00
00000190	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00
000001A0	00 00 00 00 00 00 00 00	00 10 00 00 48 00 00 00
000001B0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00
000001C0	00 00 00 00 00 00 00 00	00 00 00 00 00 00

6.4 Click on the **IMAGE_FILE_HEADER** section.

The screenshot shows the PEview interface with the file '30_723bio_152.exe' open. The left pane displays the file structure with the 'IMAGE_FILE_HEADER' section expanded. The right pane shows the data for each field in the IMAGE_FILE_HEADER structure, including fields like Data, Description, and Value. The 'Value' column contains hex values and some ASCII characters.

Data	Description	Value
014C	Machine	IMAGE_FILE_MACHINE_I386
0002	Number of Sections	
596B15D1	Time Date Stamp	2017/07/16 Sun 07:29:21 UTC
000000DC	Pointer to Symbol Table	
000000E0	Number of Symbols	
000000E4	Size of Optional Header	
0103	Characteristics	
0001		IMAGE_FILE_RELOCS_STRIPPED
0002		IMAGE_FILE_EXECUTABLE_IMAGE
0100		IMAGE_FILE_32BIT_MACHINE

The Time Date Stamp is the date and time the malware was compiled. This is very useful information in malware analysis and incident response. The newer the timestamp, the more likely this malware has evaded security mechanisms.



6.5 Expand the IMAGE_SECTION_HEADER.text

The screenshot shows the PEView interface with the file '30_723bio_152.exe' open. The left pane displays the file structure with nodes like 'IMAGE_DOS_HEADER', 'MS-DOS Stub Program', 'IMAGE_NT_HEADERS', 'Signature', 'IMAGE_FILE_HEADER', 'IMAGE_OPTIONAL_HEADER', 'IMAGE_SECTION_HEADER.text', 'SECTION .text', and 'SECTION .data'. The 'IMAGE_SECTION_HEADER.text' node is selected and expanded, showing its details in the right pane. The right pane has four columns: 'pFile', 'Data', 'Description', and 'Value'. The 'Value' column contains memory addresses and their corresponding values and descriptions. For example, the 'Name' entry has the value '.text'. The 'Characteristics' entry at address 000001EC has several flags listed: 00000020 (IMAGE_SCN_CNT_CODE), 20000000 (IMAGE_SCN_MEM_EXECUTE), 40000000 (IMAGE_SCN_MEM_READ), and 80000000 (IMAGE_SCN_MEM_WRITE). The status bar at the bottom of the window says 'Viewing IMAGE_SECTION_HEADER.text'.

pFile	Data	Description	Value
000001C8	2E 74 65 78	Name	.text
000001CC	74 00 00 00		
000001D0	00002452	Virtual Size	
000001D4	00001000	RVA	
000001D8	00002600	Size of Raw Data	
000001DC	00000400	Pointer to Raw Data	
000001E0	00000000	Pointer to Relocations	
000001E4	00000000	Pointer to Line Numbers	
000001E8	0000	Number of Relocations	
000001EA	0000	Number of Line Numbers	
000001EC	E0000020	Characteristics	
	00000020	IMAGE_SCN_CNT_CODE	
	20000000	IMAGE_SCN_MEM_EXECUTE	
	40000000	IMAGE_SCN_MEM_READ	
	80000000	IMAGE_SCN_MEM_WRITE	

This data can be useful in looking at functions to see what the executable is doing. From this, we can see that the executable writes to a file, performs some functions, and deletes a file.

6.6 Click through each of the sections to see what other information you can find!

Great job, you have completed LAB012!

Thank You, you may now close this module.