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

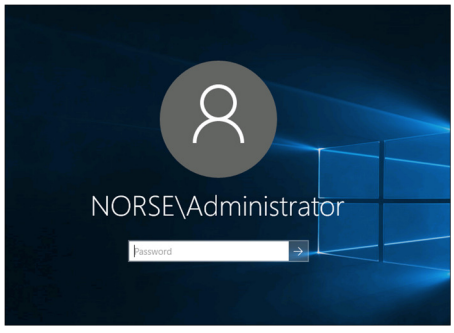
Upon completion of this lab, you should be able to:

Log in to the Lab Machine

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

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

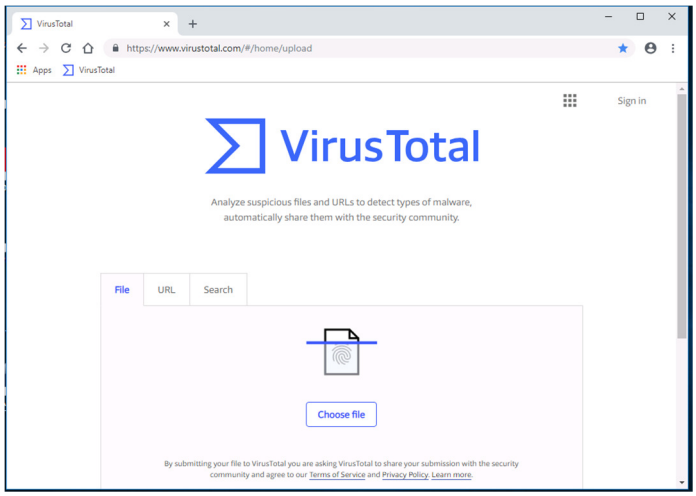
click on the  on the **Machines Tab**, and press **Enter** to log in to the machine.



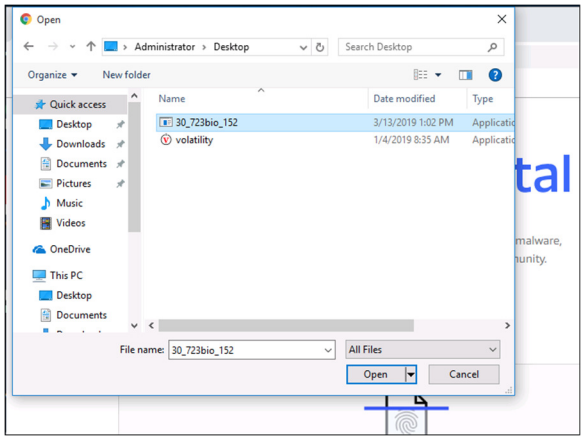
1.0 Identifying Malware

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

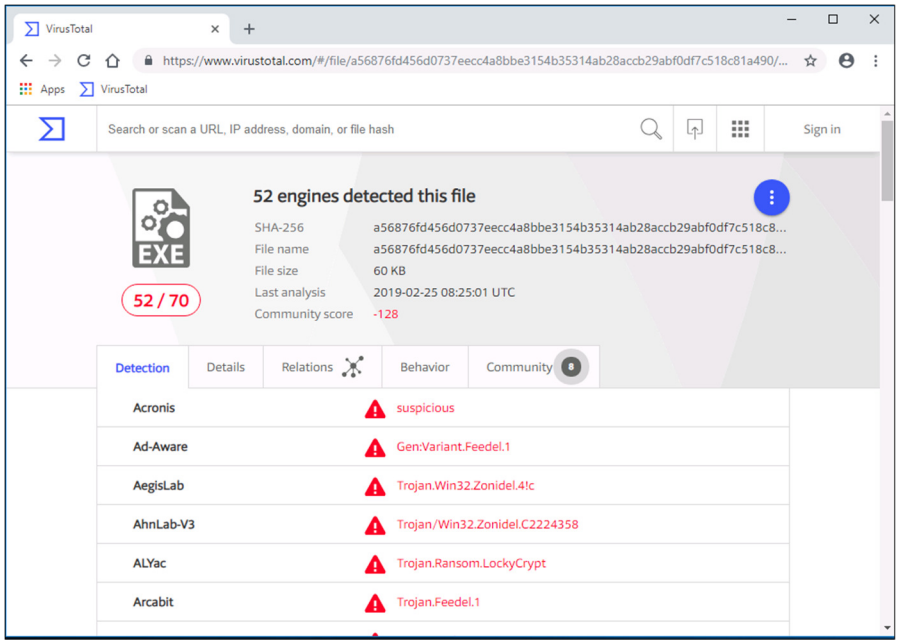
1.2 Navigate to https://www.virustotal.com



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.



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.

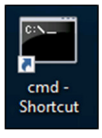
You can only imagine a defense attorney or other representative who will argue that any evidence that you have found on the computer was placed there by law enforcement or the forensic investigator.

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.

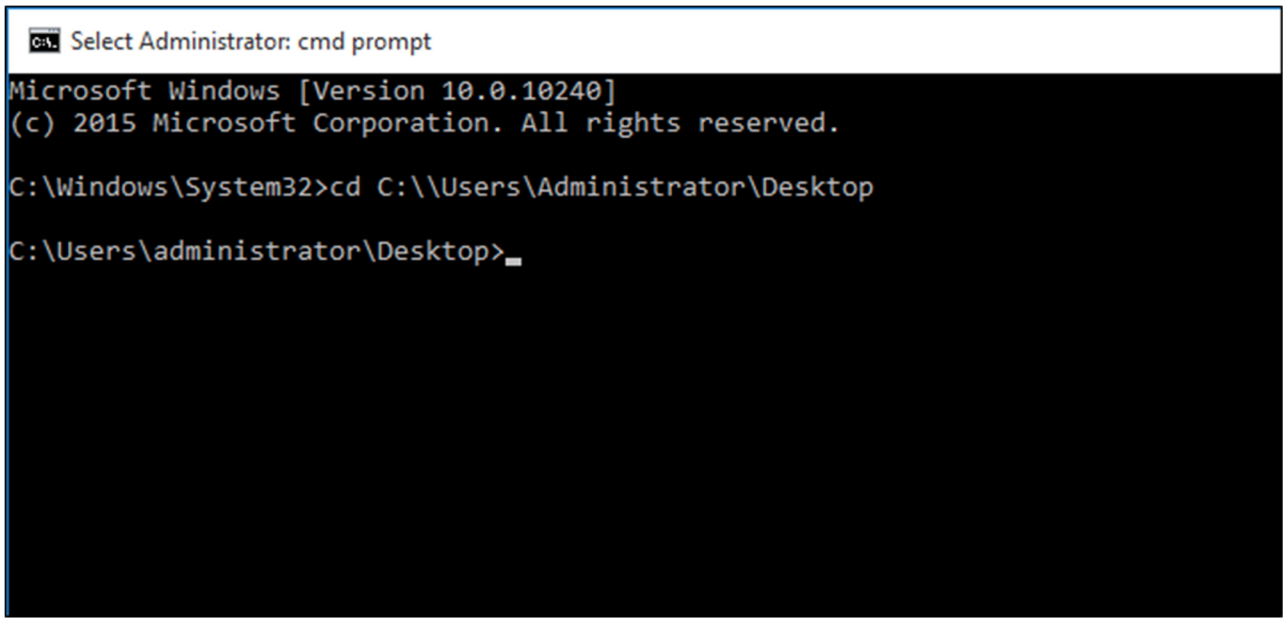
You have probably seen or used hashes when you downloaded software. In fact, when you downloaded Kali, Offensive Security provides you the MD5 hash of it so that you can check that the Kali you downloaded has not been corrupted or otherwise altered in any way before it gets to you.

2.0 Hashing a Malware File

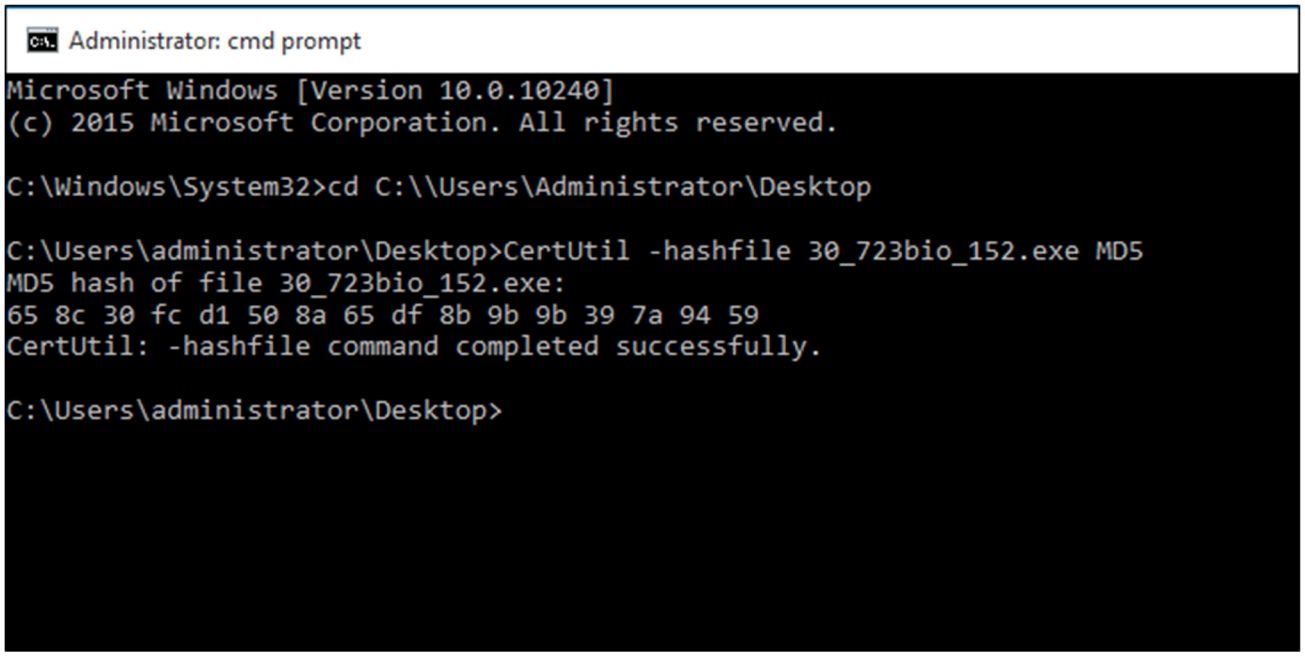
2.1 Open a **Command** prompt.



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



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



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

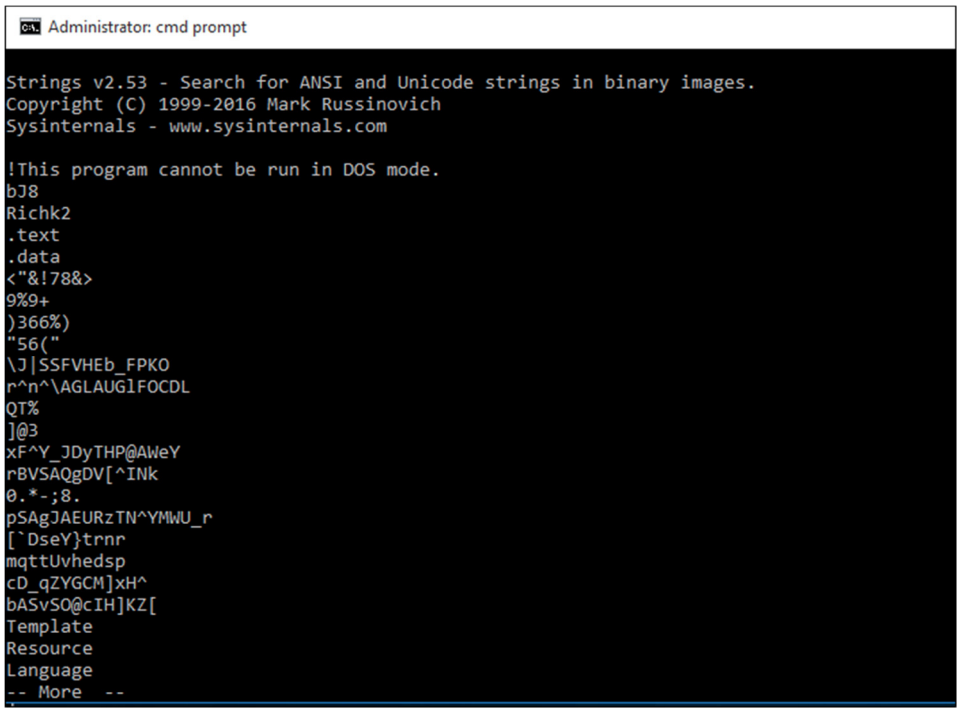
3.1 Go back to the **Command** prompt, or open a new one.

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.



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.

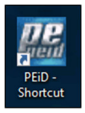
3.4 Scroll through the rest of the file using **Strings** and see what you can find.

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

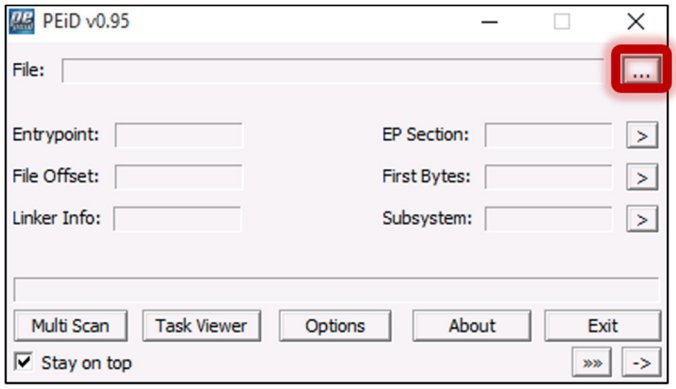
4.0 Detecting Packed Malware

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

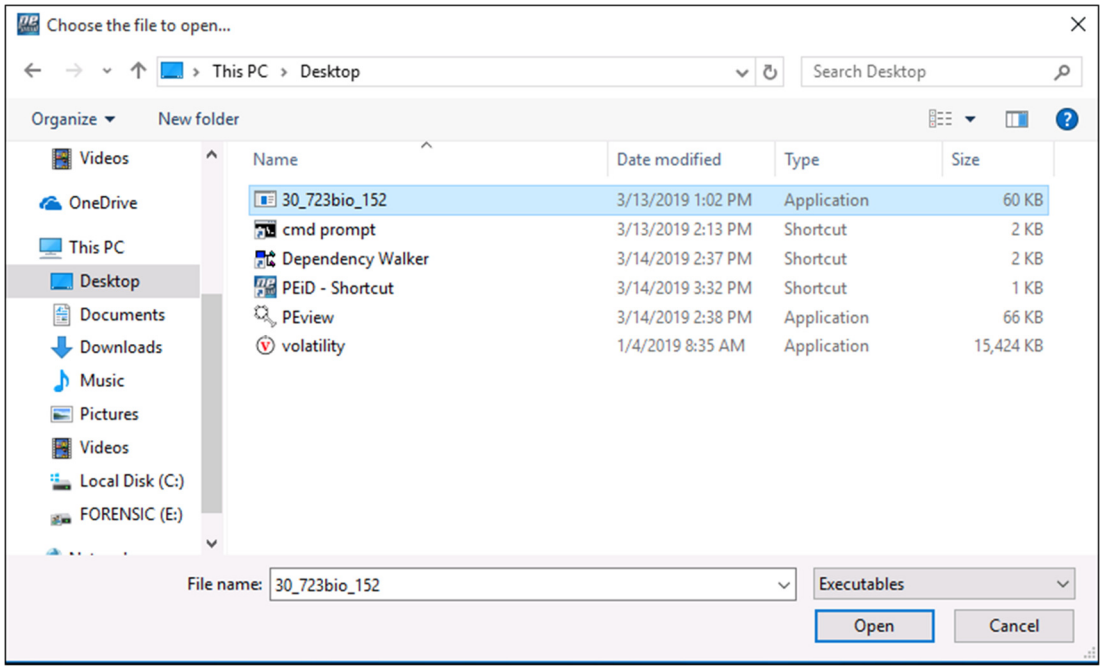
4.1 Open **PEiD**.



4.2 Click the three dots to browse for a file.

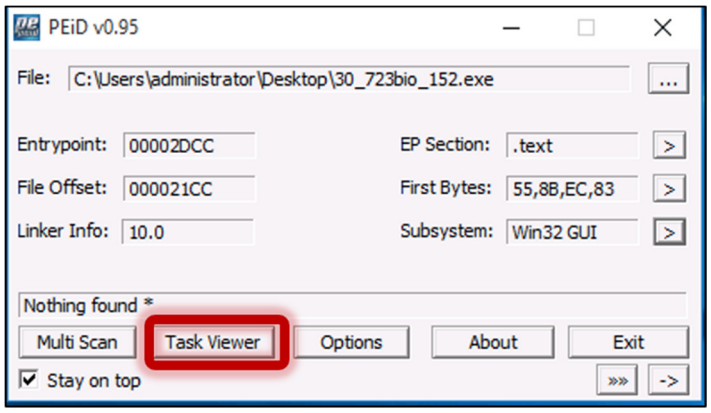


4.3 Select the malware file on the desktop.

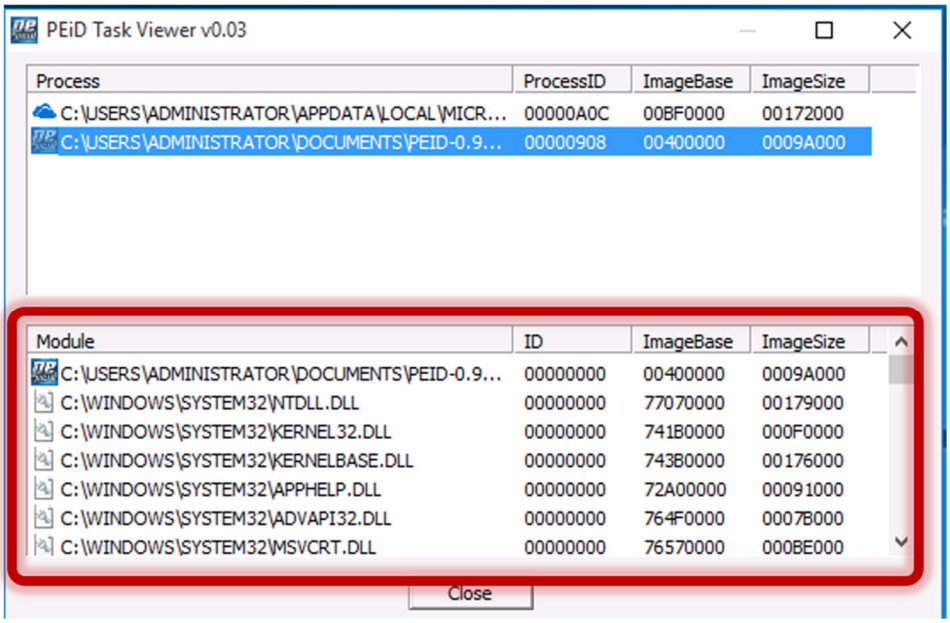


WARNING: Many PEiD plug-ins will run the malware executable without warning! Please take care and use this program in a virtual environment.

4.4 Click **Task Viewer**.



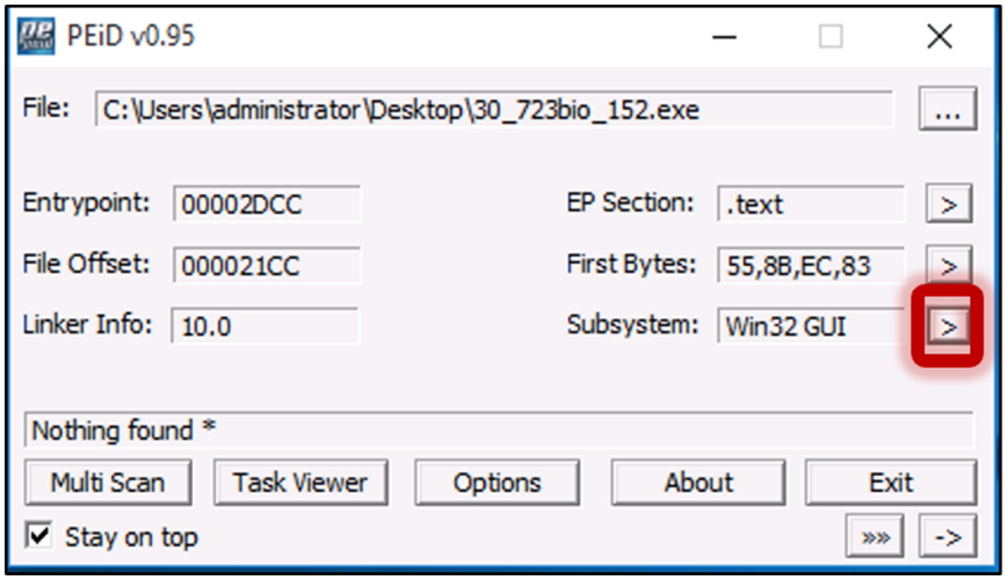
4.5 Click on the process you would like to analyze.



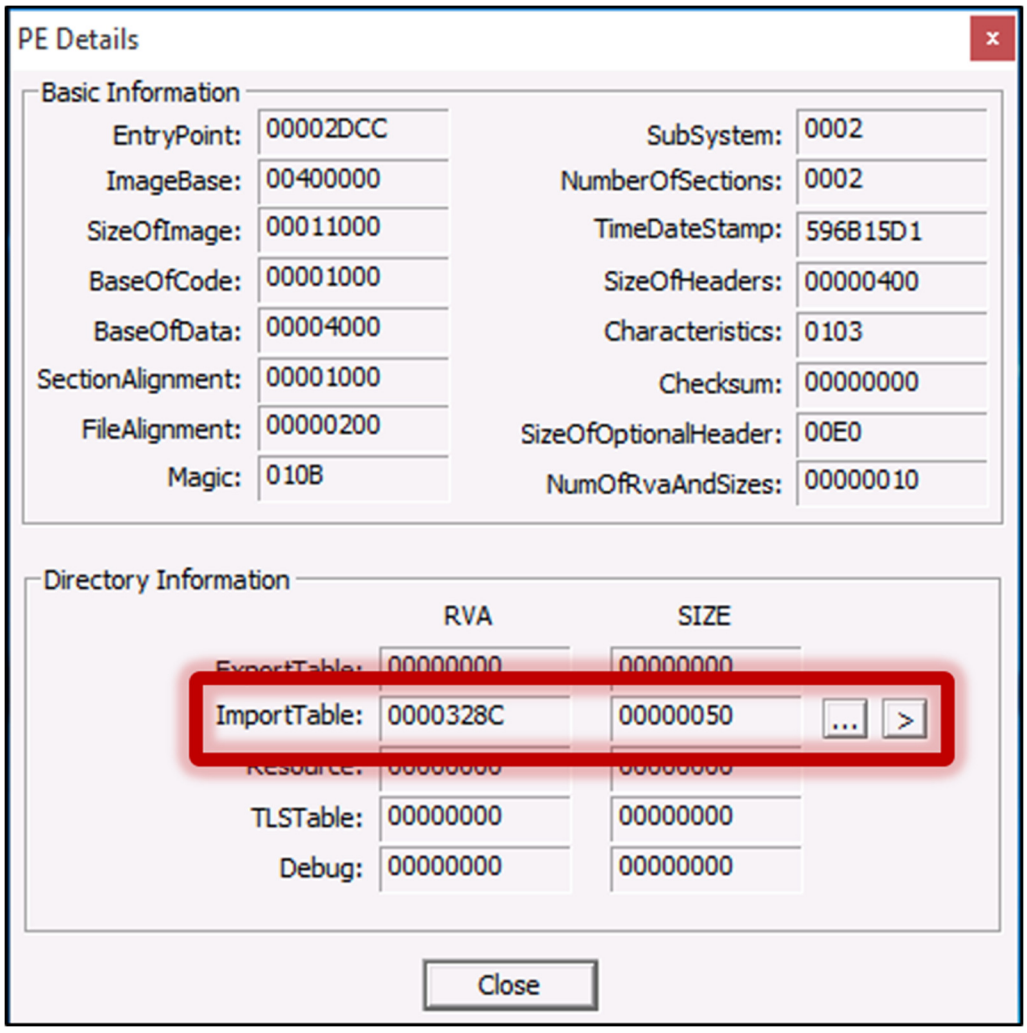
4.6 In the module section, we can see many different .dll files that are accessed.

4.7 Click **Close**.

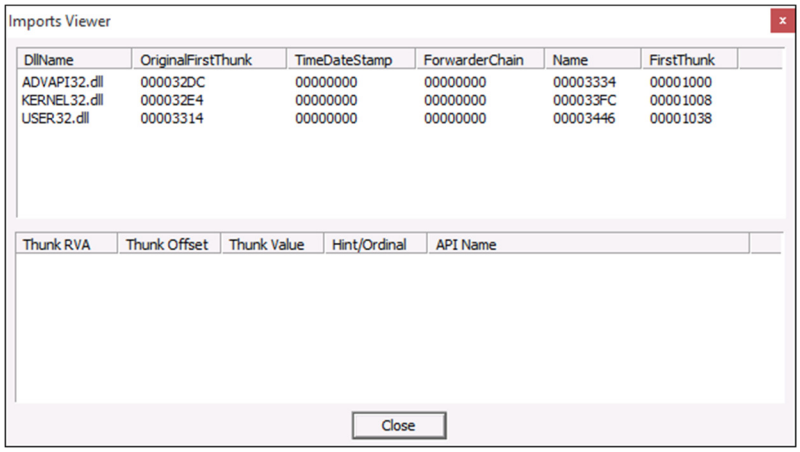
4.8 Click the arrow next to Subsystem.



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.



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.

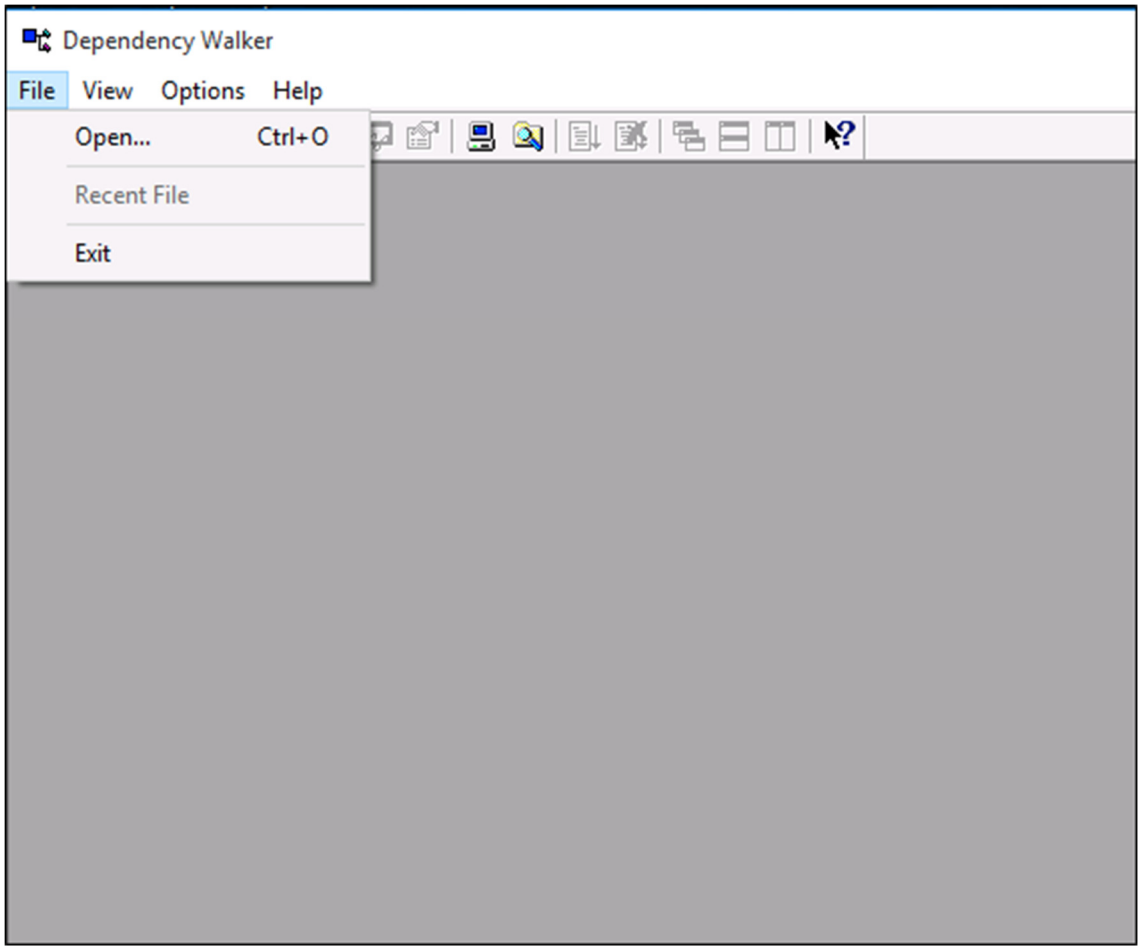
WARNING: Many PEiD plug-ins will run the malware executable without warning! Please take care and use this program in a virtual environment.

5.0 Analyzing Malware

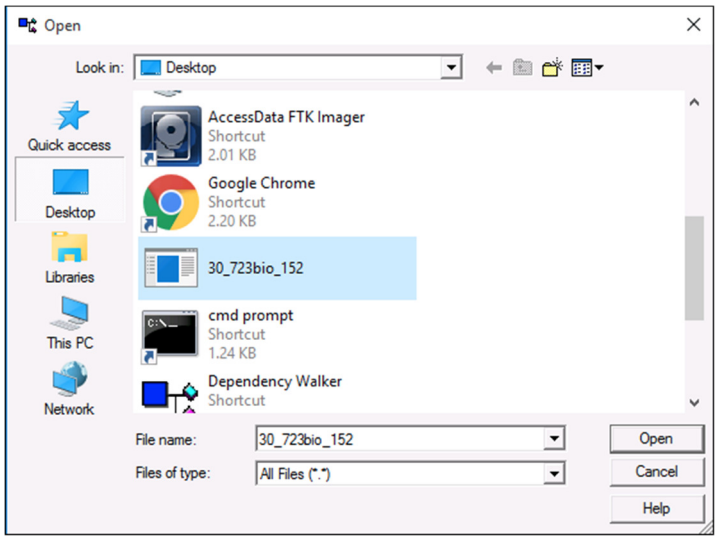
5.1 Open **Dependency Walker**.

Note: Dependency Walker will create an error message that is fine, the software has been around for a while, and hasnt been updated to run well with Windows 10.

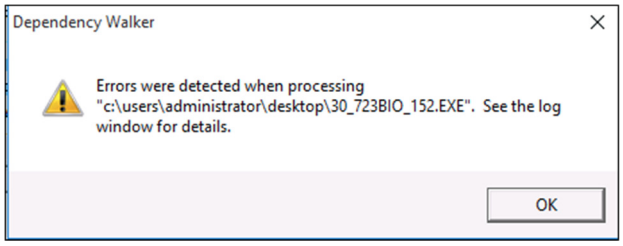
5.2 Open the malware file on the Desktop by clicking on **File** and clicking on **Open**.



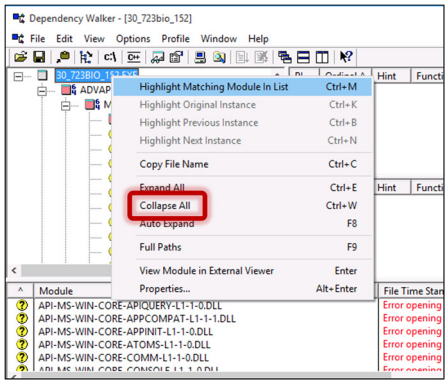
5.3 Navigate to the file using the popup window.



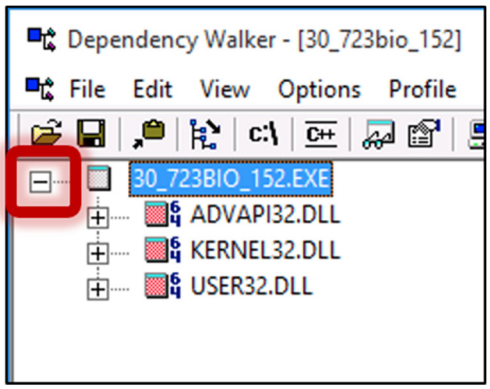
Note: If you get an error message, just disregard the error message. This malware spans across many versions of Windows and the version it was tested on utilizes this dll but it doesn't exist in later versions of Windows.



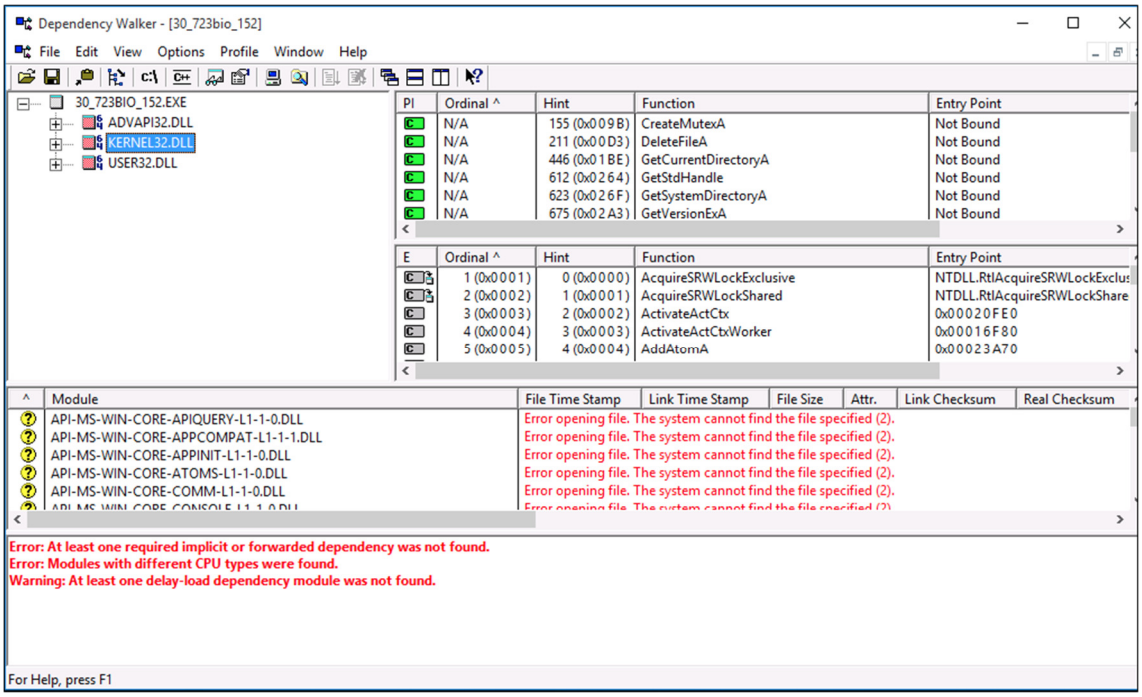
5.4 Right-click the file and select **Collapse All**. This will collapse the view, which will allow us to expand as we analyze.



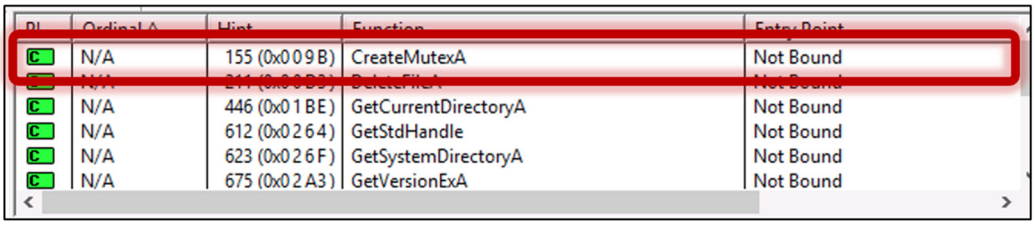
5.5 Click the '+' next to the malware name to expand once.



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



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.

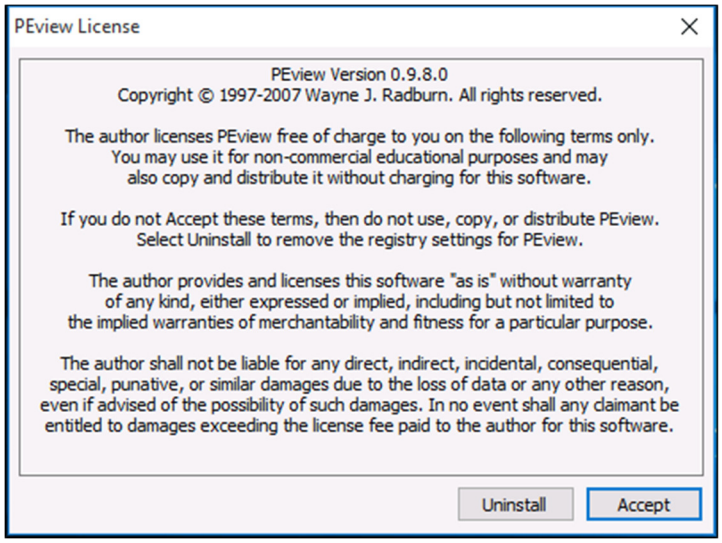


5.7 Close **Dependency Viewer**.

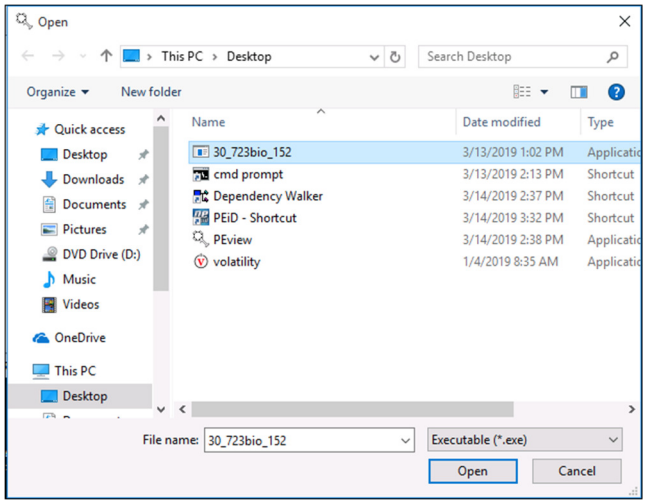
6.0 Analyzing Malware II

6.1 Open PE View.

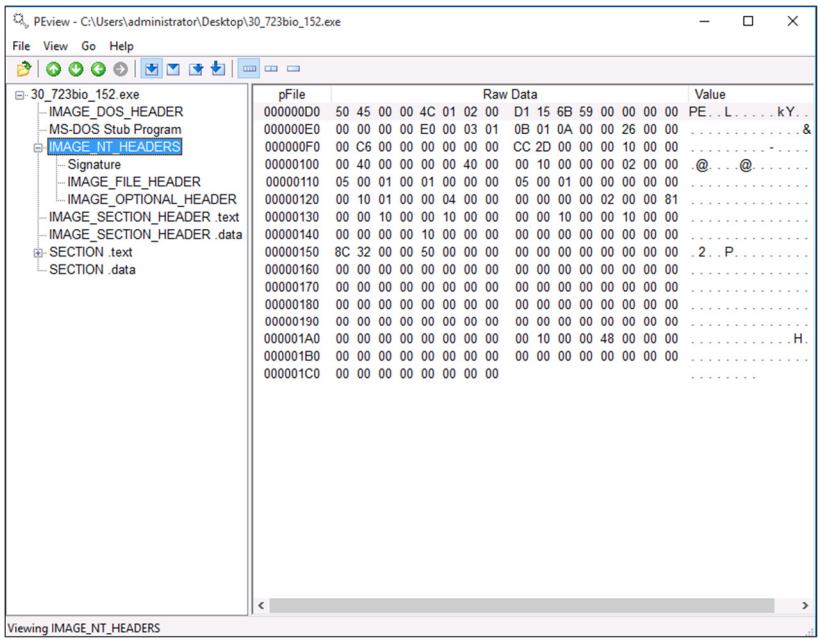
Accept the License Agreement.



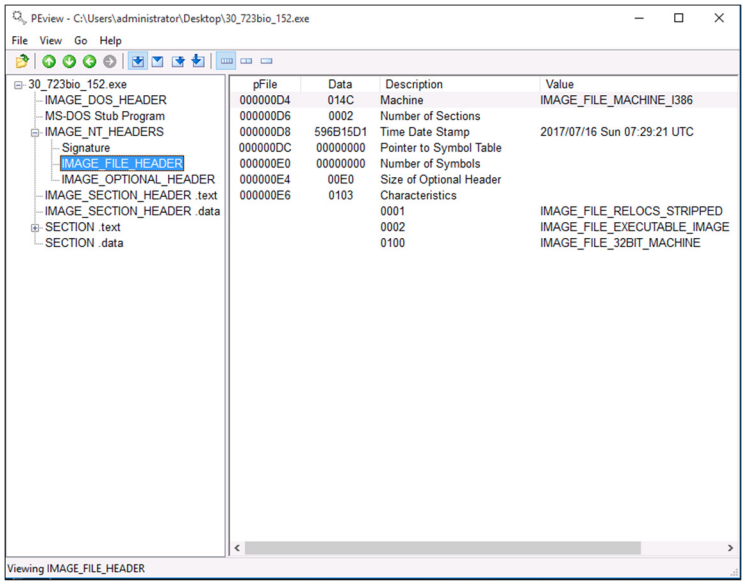
6.2 Open the malware file.



6.3 Expand the **IMAGE_NT_HEADERS** section.

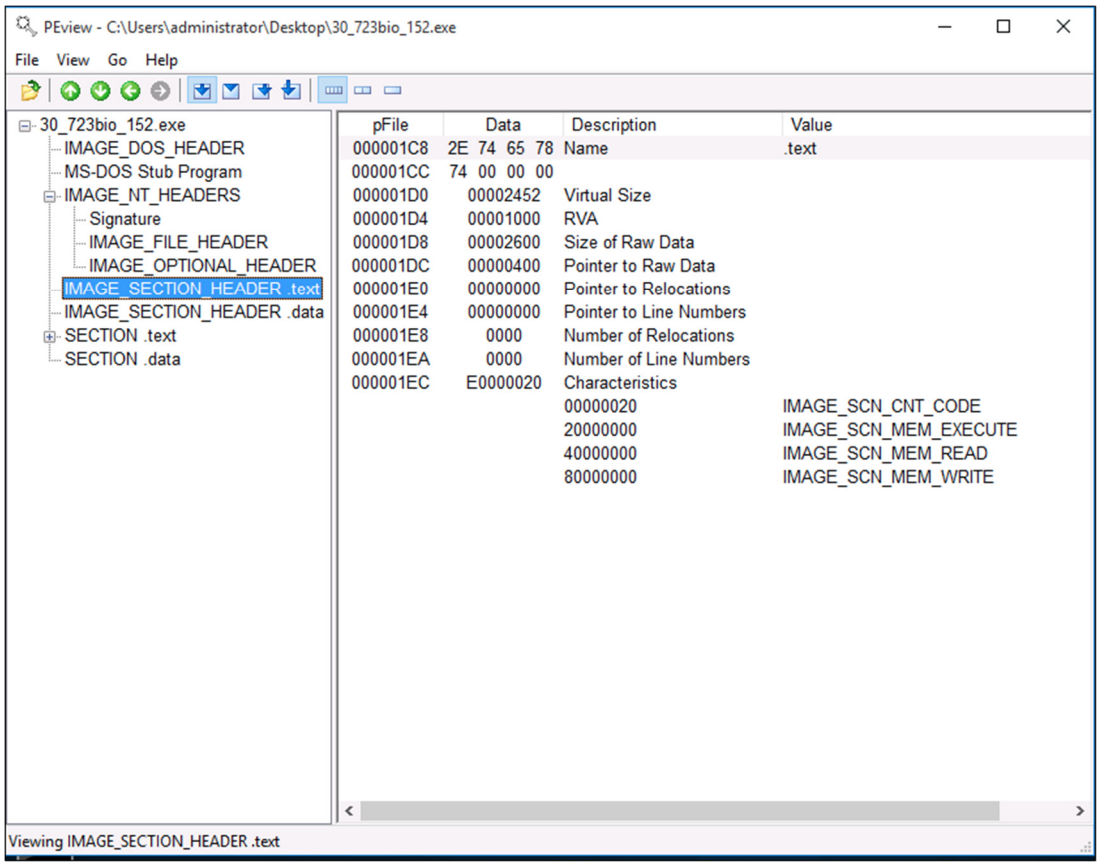


6.4 Click on the **IMAGE_FILE_HEADER** section.



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`



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.