



**OLD DOMINION
UNIVERSITY**

**CS 762
MEMORY ANALYSIS AND FORENSICS
PROJECT
STATIC ANALYSIS**

**Performed by Pavan P Galagali
(UIN: 01125293)**

INTRODUCTION

This report is about setting up a virtual lab and to perform basic static analysis on a malware using Windows malware analysis tools. Usually the first step in studying the malware is Static Analysis.

Static Analysis:

Static Analysis describes the process of analysing the code or structure of a program to determine

Its function. It employs different tools and techniques to quickly determine whether a file is Malicious or not, provide information about its functionality.

In this project I will be using the following techniques in static analysis of a malware

1. Using antivirus tools to confirm maliciousness
2. Gleaning information from a file's strings, functions and headers.

SETTING UP A VIRTUAL LAB

A virtual machine (VM) is an emulation of a computer system. Virtual machines are based on computer architectures and provide functionality of a physical computer.

- Firstly, download and install Virtual Box. Virtual Box is a free program provided by Oracle. This is the software that powers the entire virtualization process.
- Download the ISO file of Windows XP OS from the Microsoft developer's website.
- Now that you have Virtual Box installed and Windows XP, it's time to get your OS set up.
- Open Virtual Box, click New and set up an OS by giving name, memory size etc.
- Now select your new VM on the left and click Start at the top.

One of the advantages in analysing malware on virtual machines rather than physical is that a malware might infect a computer when executed and spread to other files in the OS.

However, a malware executed in a VM will spread through the VM but not affect the underlying actual OS.

ANALYSIS OF MALWARE ON WINDOWS XP OS

While analysing prospective malware for the first time, a good first step is to run it through multiple antivirus programs, which may already have identified it.

VIRUS TOTAL

Using antivirus tools to confirm maliciousness

Because the various antivirus programs use different signatures and heuristics, it's useful to run several different antivirus programs against the same piece of suspected malware. Websites such as Virus Total (<http://www.virustotal.com/>) allow you to upload a file for scanning by multiple antivirus engines. Virus Total generates a report that provides the total number of engines that marked the file as malicious, the malware name, and, if available, additional information about the malware.

1. Upload the *Lab01-03.exe* file to <http://www.VirusTotal.com/>. Does it match any existing antivirus definitions?

For the file *Lab01-03.exe*, VirusTotal.com reports a variety of different signatures with vague-sounding names. The most common signature is that of a file packed with the FSG packer.

63/71 definitions match.

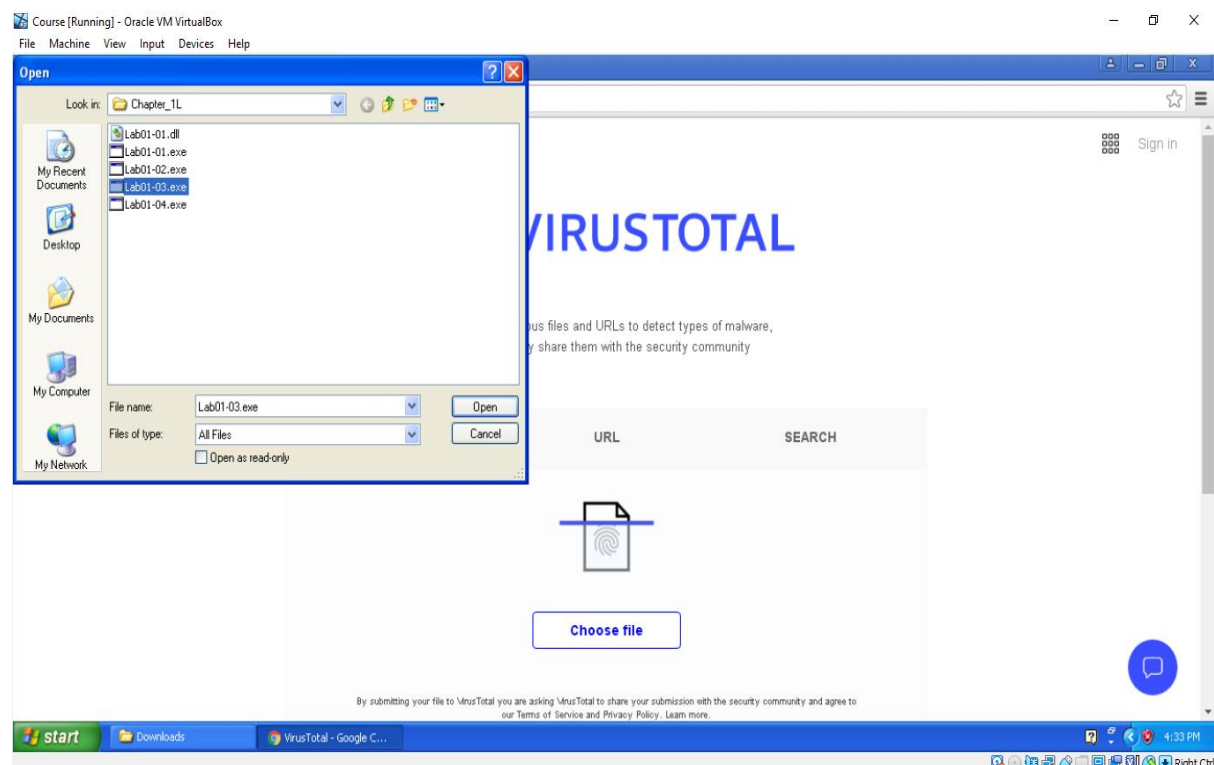


Figure 1: Selecting file for Virus Total tool

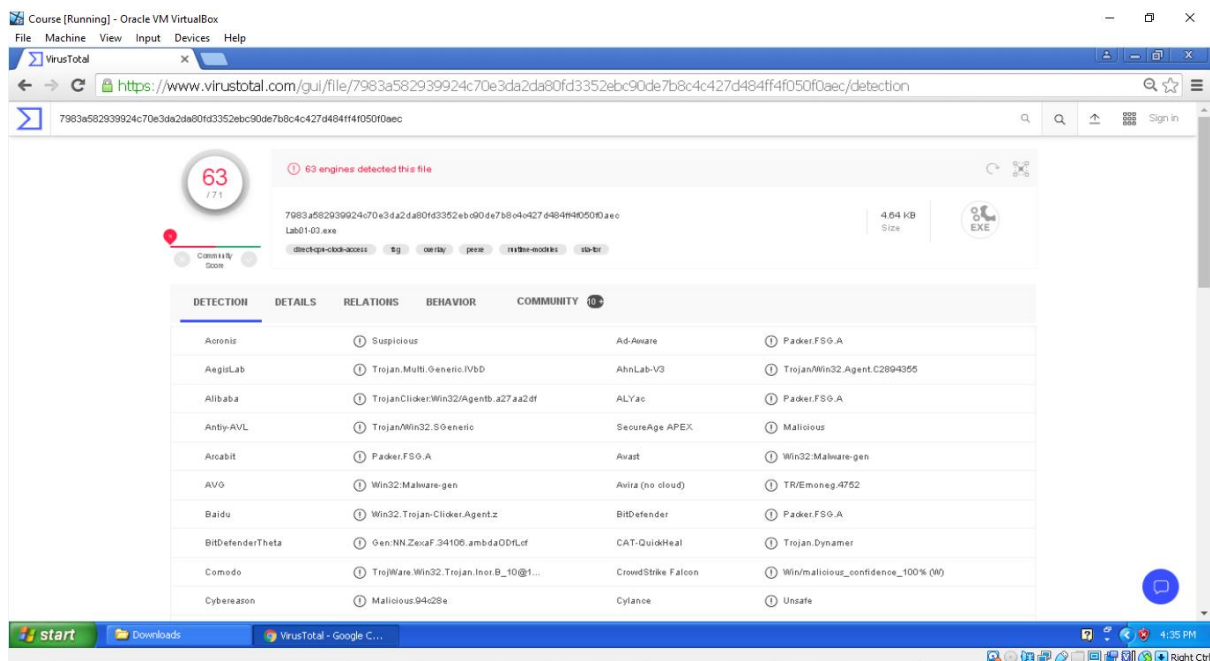


Figure 2: Finding virus signatures in Lab01-03.exe

2. Upload the Lab01-04.exe file to <http://www.VirusTotal.com/>. Does it match any existing antivirus definitions?

For the Lab01-04.exe file, the results from VirusTotal.com suggest a program related to a downloader. 60/70 definitions match.

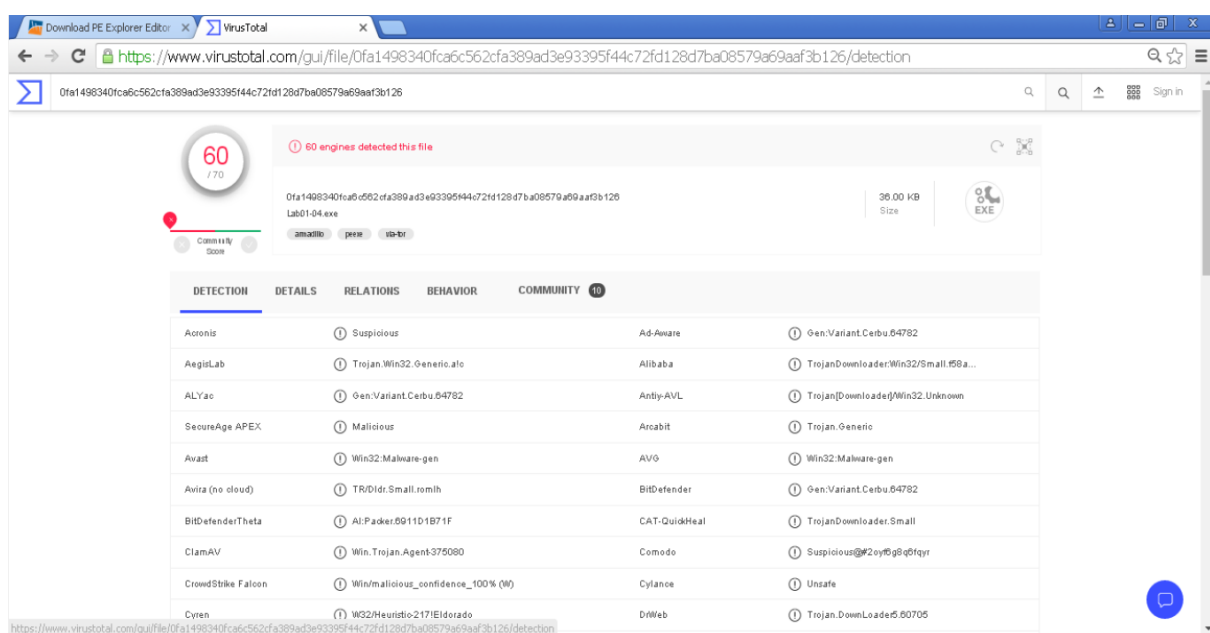


Figure 3: Finding virus signatures in Lab01-04.exe

3. Are there any indications that this file is packed or obfuscated? If so, what are these indicators? If the file is packed, unpack it if possible.

PE VIEW

PE view is a free and easy to use application to browse through the information stored in Portable Executable (PE) file headers and the different sections of the file.

When we open the file in PView, we see several indications that the file is packed. The first is that the file sections have no names.

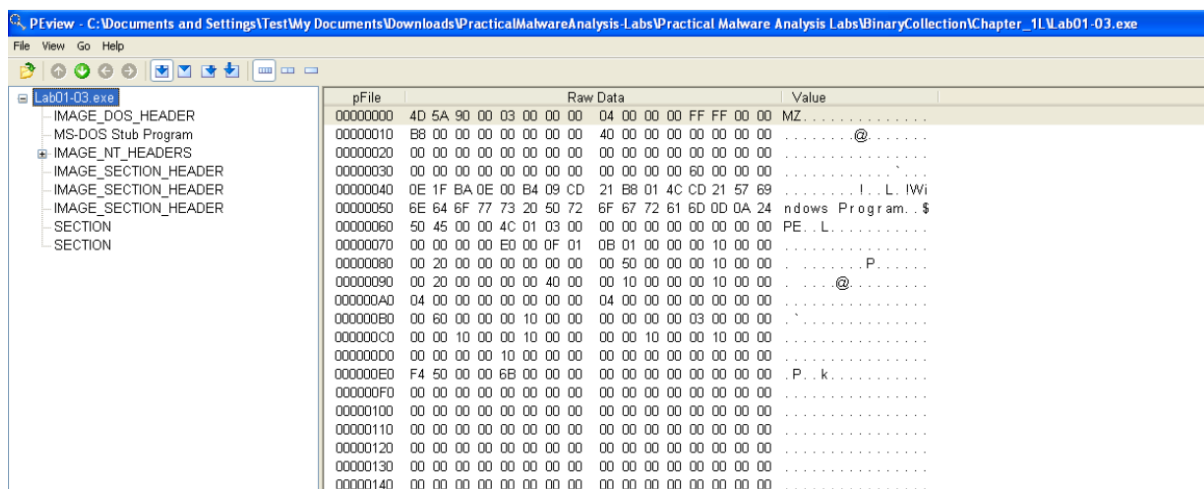


Figure 4: PView for Lab01-03.exe to check packed status

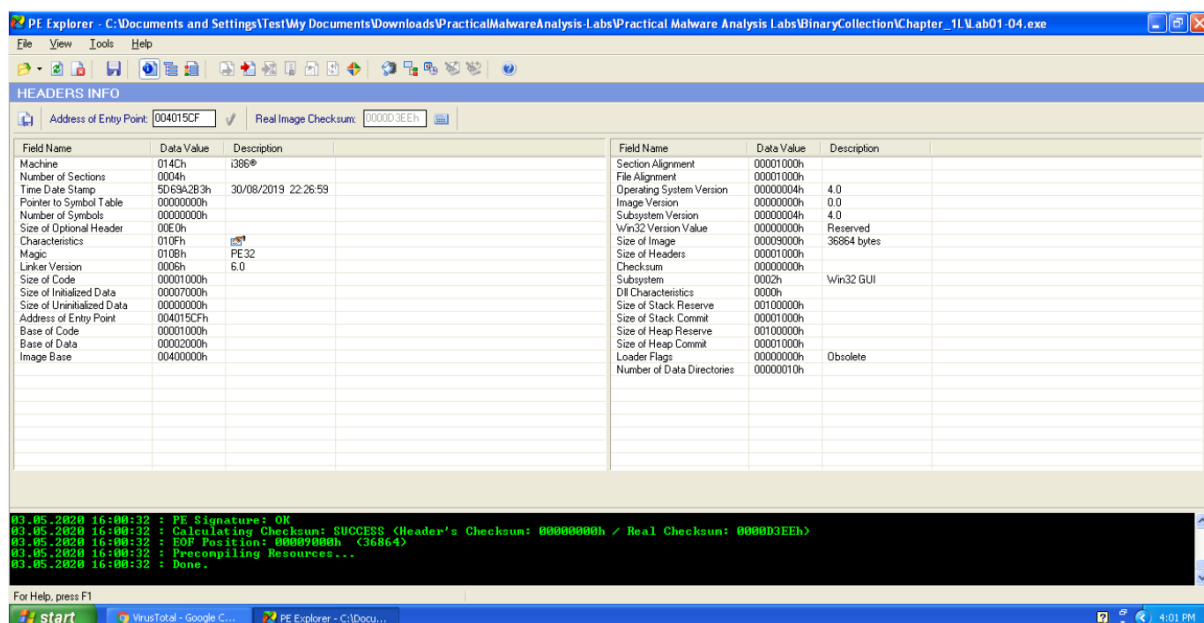


Figure 5: Using PE Explorer to find timestamp of file creation

Using PE Explorer we can observe the timestamp of the file when it was compiled.

The imports from advapi32.dll tell us that program does something with permissions, and we can assume that it tries to access protected files using special permissions.

4. When was this program compiled?

The time stamp reveals that the file was compiled on 30-08-2019.

| | |
|-----------------|---|
| Imports | Functions from other libraries that are used by the malware |
| Exports | Functions in the malware that are meant to be called by other programs or libraries |
| Time Date Stamp | Time when the program was compiled |
| Sections | Names of sections in the file and their sizes on disk and in memory |
| Subsystem | Indicates whether the program is a command-line or GUI application |
| Resources | Strings, icons, menus, and other information included in the file |

PACKED AND OBFUSCATED MALWARE

Obfuscated programs are ones whose execution the malware author has attempted to hide. Packed programs are a subset of obfuscated programs in which the malicious program is compressed and cannot be analysed. Malware that is packed or obfuscated contains very few strings.

Detecting Packed Files

One way to detect packed files is with the PEiD program. You can use PEiD to detect the type of packer or compiler employed to build an application, which makes analysing the packed file much easier.

Next, we see that the first section has a virtual size of 0x3000, but a raw data size of 0. We run PEiD to confirm, and it identifies the packer as FSG 1.0 -> dulek/xt.

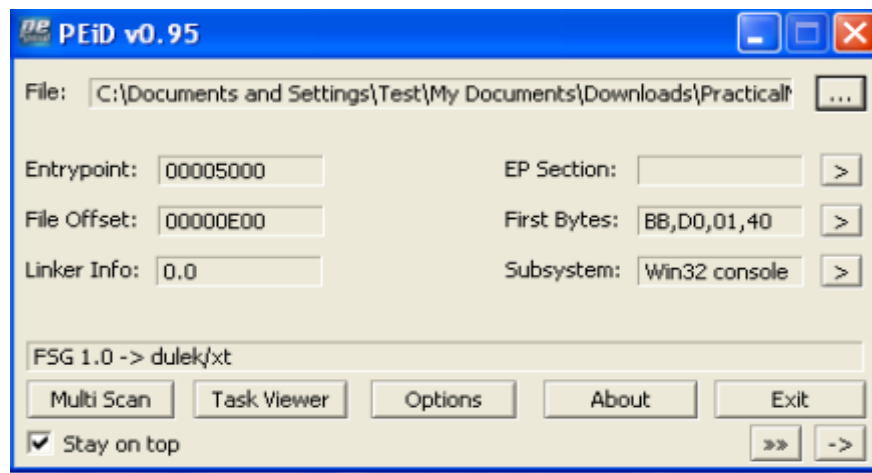


Figure 6: PEiD showing that the file Lab01-03.exe is packed using FSG 1.0

5. Do any imports hint at this program's functionality? If so, which imports are they and what do they tell you?

To confirm that the file is packed, we search for the imports, but there doesn't seem to be an import table. An executable file without an import table is extremely rare, and its absence tells us that we should try another tool, because PView is having trouble processing this file.



Figure 7: PEiD showing signs of packed file Lab01-03.exe

If we run PEiD program on Lab01-04.exe file we can see that the file is unpacked and these files are identified as "Microsoft Visual C++" files, which shows that they are unpacked

PEiD gives no indication that the file is packed or obfuscated.

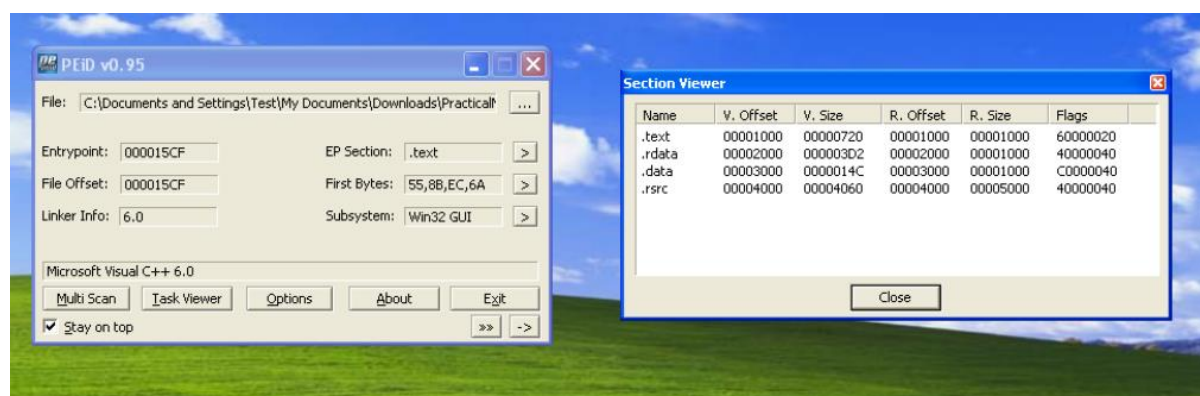


Figure 8: PEiD showing file Lab01-04.exe as an unpacked file

Likewise, the EP section shows the valid file names as well as actual raw sizes for them all, rather than UPX0-3 or blanks, as well as a raw size of 0, typically seen for packed files. As this is not packed, there is no need to unpack it.

DEPENDENCY WALKER

Now we have to check the dynamically linked list in the executable with any of the tools available. The libraries used and functions calls are often the most important parts of a program, and identifying them is particularly important, because it allows us to guess at what the program does.

The Dependency Walker program lists only dynamically linked functions in an executable. Dependency Walker also displays the dependencies of the file which will result in a minimum set of required files. Dependency Walker also displays detailed information about those files including the file path, version number, machine type, debug information etc.

We open the file with Dependency Walker, and see that it does have an import table, but it imports only two functions: LoadLibrary and GetProcAddress. Packed files often import only these two functions, which further indicate that this file is packed.

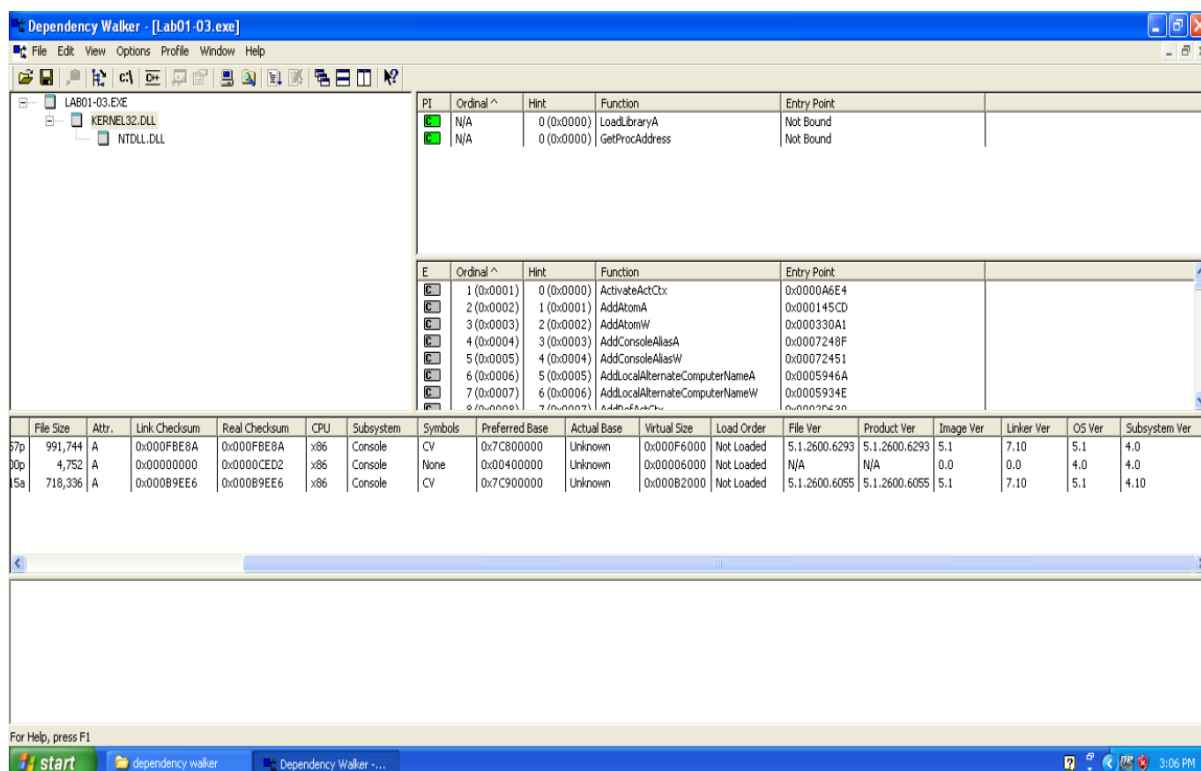


Figure 9: Dependency Walker showing functions imported by Lab01-03.exe kernel32.dll

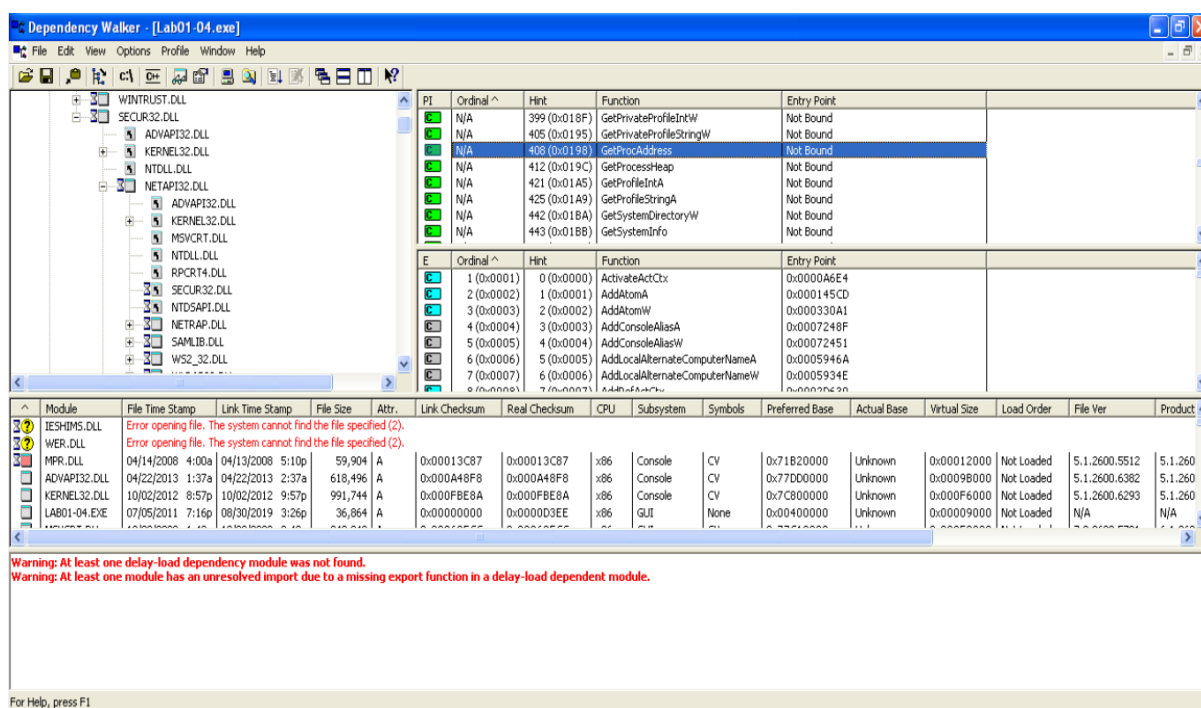


Figure 10: Dependency Walker showing functions imported by Lab01-04.exe

The imports from advapi32.dll tell us that program does something with permissions, and we can assume that it tries to access protected files using special permissions. The imports from

kernel32.dll tell us that the program loads data from the resource section (LoadResource, FindResource, and SizeOfResource), writes a file to disk (CreateFile and WriteFile), and executes a file on the disk (WinExec). We can also guess that the program writes files to the system directory because of the calls to GetWindowsDirectory.

STRINGS

A string in a program is a sequence of characters such as “the.” A program contains strings if it prints a message, connects to a URL, or copies a file to a specific location.

Searching through the strings can be a simple way to get hints about the functionality of a program. For example, if the program accesses a URL, then you will see the URL accessed stored as a string in the program. You can use the Strings program, to search an executable for strings, which are typically stored in either ASCII or Unicode format.

6. What host- or network-based indicators could be used to identify this malware on infected machines?

We are unable to unpack the file the visible imports are uninformative, and we can't see any strings in PE Explorer, it is difficult to suggest what the file might do, or identify any host/network based malware-infection indicators. Using cmd command strings <filename>

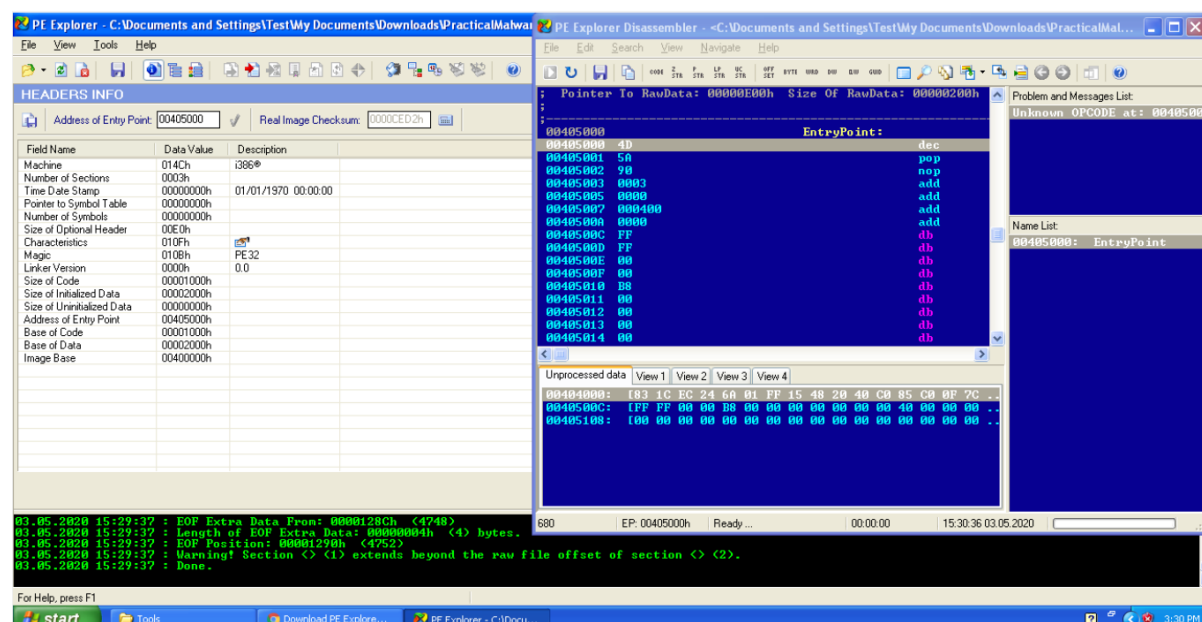


Figure 11: PE Explorer revealing the packed status of Lab01-03.exe by hiding strings

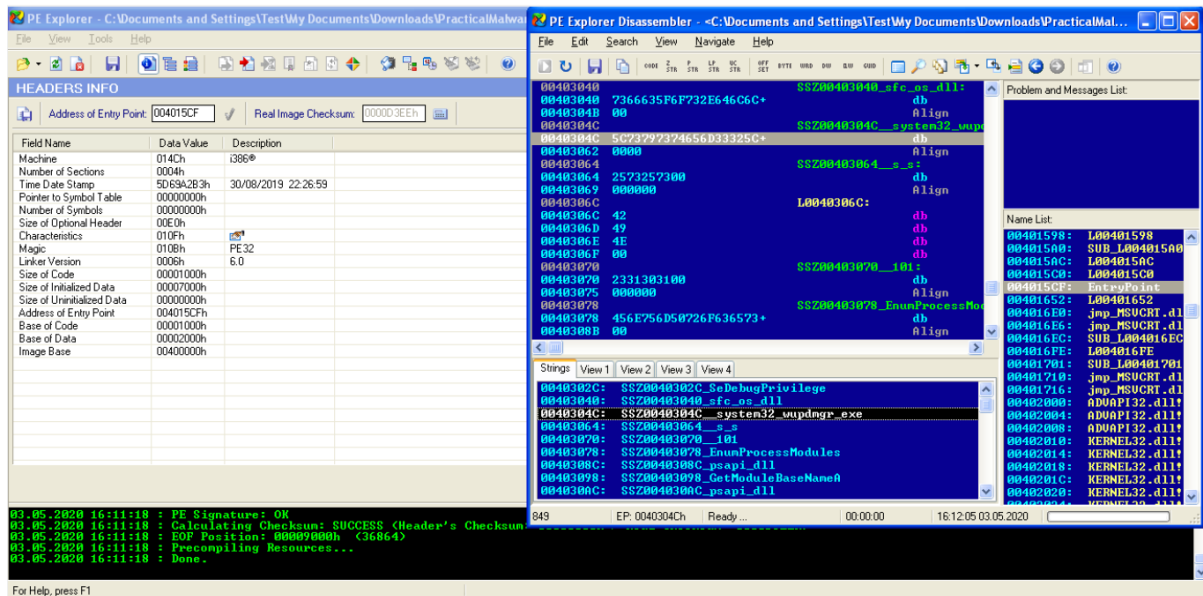
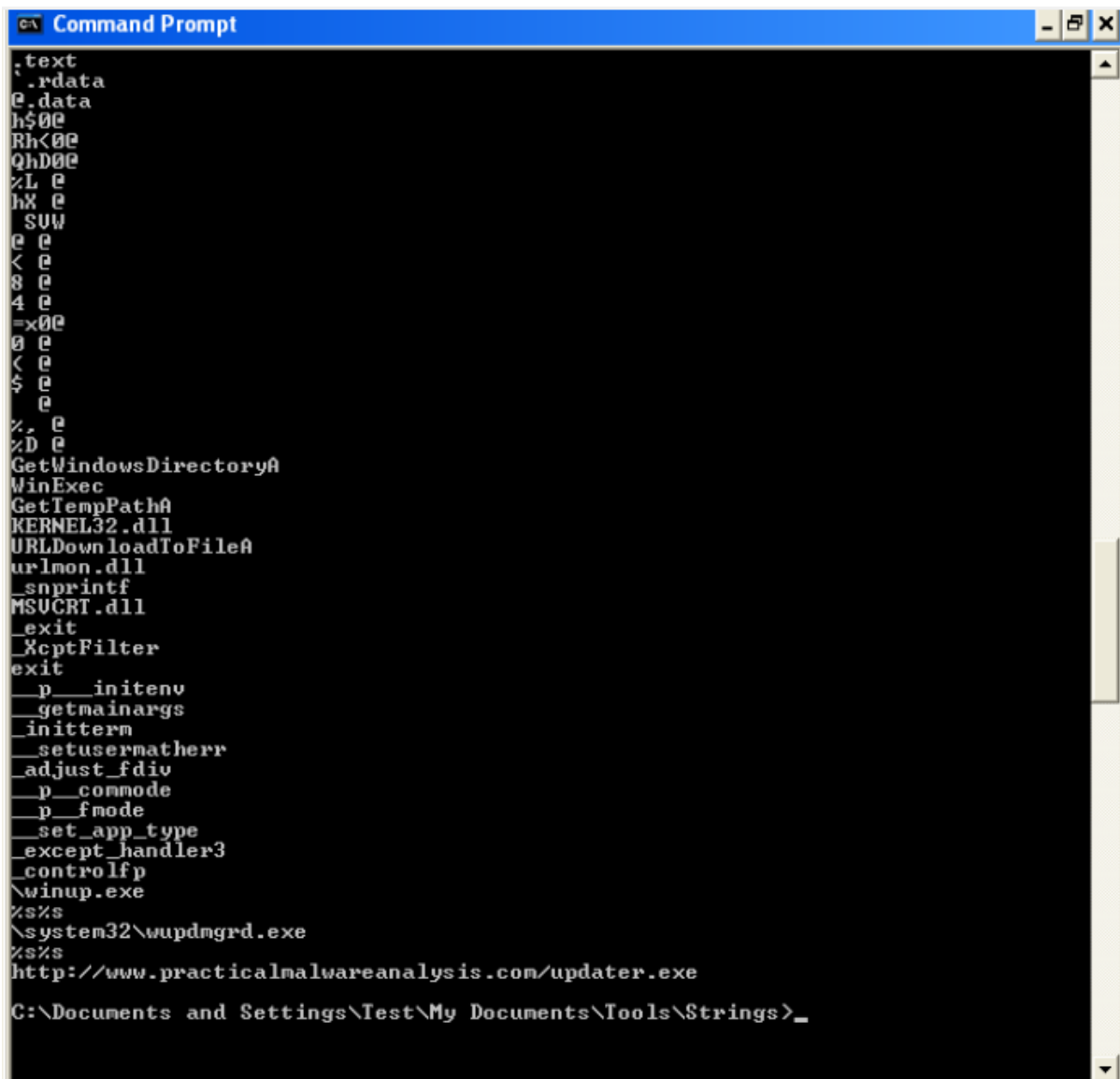


Figure 12: PE Explorer showing strings in Lab01-04.exe

The strings of note here look like ‘\system32\wupdmgr.exe’, ‘psapi.dll’ and ‘winup.exe’ — potentially these are the files which the .dlls identified, create, or execute ‘\system32\wupdmgr.exe’ might correlate with KERNEL32.dll’s GetWindowsDirectory function to write to system directory and the malware might modify the Windows Update Manager.

This gives us some host-based indicators, however there is nothing apparent regarding network functions.



```
Command Prompt
.text
.rdata
.e.data
h$00
Rh<00
QhD00
%L %
hX %
SUW
% %
< %
8 %
4 %
-x00
0 %
< %
$ %
% %
% %
GetWindowsDirectoryA
WinExec
GetTempPathA
KERNEL32.dll
URLDownloadToFileA
urlmon.dll
_snprintf
MSUCRT.dll
_exit
_XcptFilter
exit
_p__initenv
_getmainargs
_initterm
_setusermatherr
_adjust_fdiv
_p__commode
_p__fmode
_set_app_type
_except_handler3
_controlfp
\winup.exe
%s%s
\system32\wupdmgr.exe
%s%s
http://www.practicalmalwareanalysis.com/updater.exe
C:\Documents and Settings\Test\My Documents\Tools\Strings>_
```

Figure 13: Displaying strings of Lab01-04.exe using strings.exe in CMD

Examining the strings, we see `www.malwareanalysisbok.com/updater.exe`, which is probably the location that holds the malicious code for download. We also see the string `\system32\wupdmgr.exe`, which, in combination with the call to `GetWindowsDirectory`, suggests that a file in `C:\Windows\System32\wupdmgr.exe` is created or edited by this malware.

RESOURCE HACKER

7. This file has one resource in the resource section. Use Resource Hacker to examine that resource, and then use it to extract the resource. What can you learn from the resource?

Previously overlooked in PEiD's Section Viewer, there is a resources file .rsrc —The Resource Table. This is also seen in PE Explorer's Section Headers.

We are able to open this within Resource Hacker, a tool which can be used to manipulate resources within Windows binaries. Loading Lab01–04.exe into Resource Hacker identifies that resource as binary and lets us search through it.

Most data is illegible, other than a string '*! This program cannot be run in DOS mode*' — which is a stub to print the message if the program is run in MS-DOS. This is included in the DOS header of all PE files — suggesting that actually the binary resource is actually a hidden executable file.

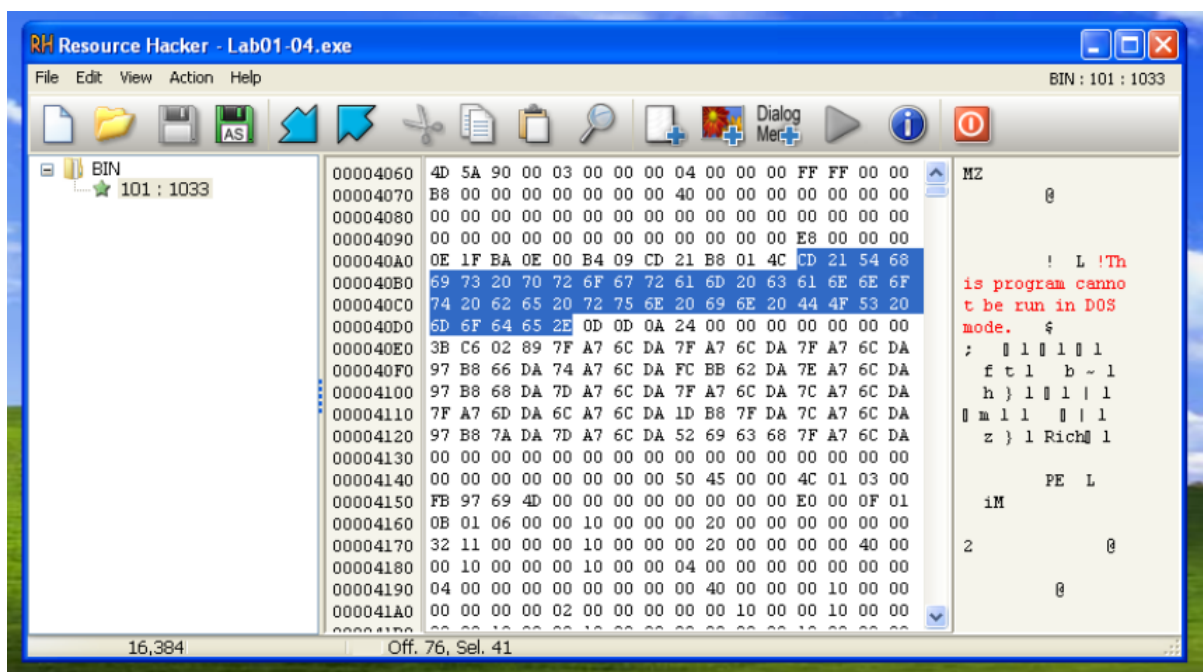


Figure 14: Resource Hacker showing ability to modify windows binaries.

Combining the initial **Lab01–04.exe** and the file hidden in the resource table, it can be assumed that most likely **Lab01–04.exe** might modify permissions to write to directory, potentially to replace with the hidden file, which performs network functionality to download and execute more malware.

This are all the various methods and tools used in Basic Static Analysis of a Malware in Windows.

REFERENCES

- http://venom630.free.fr/pdf/Practical_Malware_Analysis.pdf
- <https://medium.com/practical-malware-analysis-lab-solutions/practical-malware-analysis-lab-solutions-static-analysis-4f892cbae9d>