[2024]

MALWARE ANALYSIS

CTEC3754

FERAS MOHAMMED | P2707310

Contents

Part 1 Static and dynamic analysis of an unknown suspicious files

Question 1

1. Retrieve the two PDF documents from the "cw_pdf_files.7z" archive file. Perform a comprehensive analysis of the two files and present your findings, drawing conclusions as to whether or not each of the files may be a malicious PDF document.

```
remnux@remnux: ~/Desktop$ sudo pdfid.py cw_pdf_sample1.pdf
PDFiD 0.2.1 cw_pdf_sample1.pdf
 PDF Header: %PDF-1.4
 obj                    86
 endobj                 86
 stream                 50
 endstream              50
 xref                    2
 trailer                 2
 startxref               2
 /Page                   3
 /Encrypt                0
 /ObjStm                 0
 /JS                     0
 /JavaScript             0
 /AA                     0
 /OpenAction             0
 /AcroForm               0
 /JBIG2Decode            0
 /RichMedia              0
 /Launch                 0
 /EmbeddedFile           0
 /XFA                    0
 /Colors > 2^24          0
```

```
remnux@remnux: ~/Desktop$ sudo pdfid.py cw_pdf_sample2.pdf
PDFiD 0.2.1 cw_pdf_sample2.pdf
 PDF Header: %PDF-1.6
 obj                   146
 endobj                146
 stream                 55
 endstream              55
 xref                    1
 trailer                 1
 startxref               1
 /Page                   1
 /Encrypt                0
 /ObjStm                 0
 /JS                     2
 /JavaScript             2
 /AA                     2
 /OpenAction             0
 /AcroForm               1
 /JBIG2Decode            0
 /RichMedia              0
 /Launch                 0
 /EmbeddedFile           1
 /XFA                    0
 /Colors > 2^24          0
```

These are the results I got after running PDFid Tool on the first and second pdf respectively.

"cw_pdf_sample1.pdf"

The indication of %PDF-1.4 suggests that this PDF document is version 1.4. Objects within the PDF structure have been identified by the tool, and it is probable that these objects contain text, fonts, images, and other components of the document.↲Information about the PDF's structure and cross-references between objects can be found in the Trailer section.↲Utilizing the pdfid tool for analysis may be beneficial as it can detect embedded JavaScript within the PDF.

"cw_pdf_sample2.pdf"

The inclusion of /JavaScript 2 and /JS 2 indicates that the PDF may have embedded JavaScript code. This could raise concerns, as JavaScript within PDFs can be exploited for harmful purposes.

The presence of /AcroForm 1 suggests that the PDF probably contains an interactive form. While this is not necessarily suspicious, malicious individuals may use PDF forms for phishing attempts.

Additionally, the line /EmbeddedFile 1 implies that the PDF might contain one or more embedded files. These embedded files could potentially pose a threat depending on their nature and content.

Results from PEEPDF TOOL.

```
remnux@remnux: ~/Desktop$ peepdf cw_pdf_sample1.pdf
File: cw_pdf_sample1.pdf
MD5: c30c96c9d9e9bf9a454ab0b8fb754b14
SHA1: 05433eccfea1c825b760a415d1ef432eadfb0c74
Size: 139996 bytes
Version: 1.4
Binary: True
Linearized: True
Encrypted: False
Updates: 1
Objects: 86
Streams: 50
Comments: 0
Errors: 0

Version 0:
        Catalog: 26
        Info: 24
        Objects (1): [25]
        Streams (0): []

Version 1:
        Catalog: No
        Info: No
        Objects (85): [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45,
 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86]
        Streams (50): [86, 40, 42, 43, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 66, 67, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 3, 4, 5, 6, 7,
 8, 13, 15, 16, 17, 18, 23]
                Encoded (43): [86, 40, 42, 43, 48, 49, 50, 51, 52, 53, 54, 55, 56, 58, 59, 60, 62, 63, 66, 67, 70, 71, 72, 74, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 3, 5, 7, 8, 13, 15, 16
, 17, 18]
```

```
remnux@remnux: ~/Desktop$ peepdf cw_pdf_sample2.pdf
File: cw_pdf_sample2.pdf
MD5: 15d8b554bc3e87889c3199c4faa82d48
SHA1: 8b6e1fcad823d24c8b38a61d2a10c617ed2a8976
Size: 149387 bytes
Version: 1.6
Binary: False
Linearized: True
Encrypted: False
Updates: 0
Objects: 146
Streams: 55
Comments: 0
Errors: 0

Version 0:
        Catalog: 8
        Info: 6
        Objects (146): [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44
, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91,
 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 1
31, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146]
                Errors (1): [144]
        Streams (55): [5, 17, 19, 20, 22, 24, 25, 35, 37, 38, 40, 42, 43, 52, 59, 61, 62, 64, 65, 66, 68, 69, 70, 71, 72, 73, 74, 75, 80, 81, 85, 88, 89, 93, 96, 98, 101, 103, 106, 108, 111
, 113, 116, 118, 121, 123, 126, 128, 131, 133, 136, 138, 140, 143, 144]
                Encoded (37): [52, 62, 66, 68, 69, 70, 71, 72, 73, 74, 75, 80, 81, 85, 88, 89, 93, 96, 98, 101, 103, 106, 108, 111, 113, 116, 118, 121, 123, 126, 128, 131, 133, 136, 138, 14
3, 144]
        Objects with JS code (1): [144]
        Suspicious elements:
                /AcroForm: [8, 144]
                /Names: [4, 8]
                /XFA: [144]
                /AA: [11, 48]
                /JS: [141, 142]
                /JavaScript: [141, 142]
                /EmbeddedFiles: [8]
                /EmbeddedFile: [144]
```

"cw_pdf_sample1.pdf

The PDF is 139996 bytes in size, which is relatively small. This does not necessarily imply that the content is malicious, but it's worth noting.- The PDF version is 1.4.- It's a binary file and linearized, allowing progressive opening while downloading.

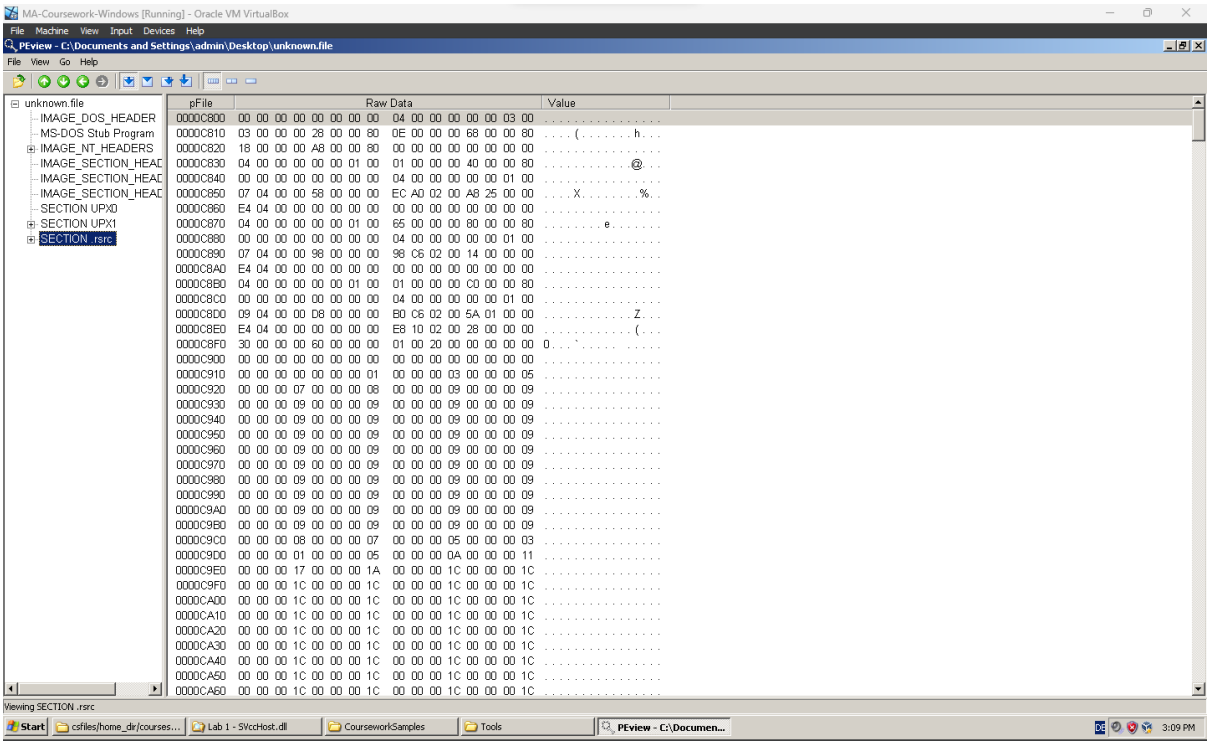
"cw_pdf_sample2.pdf

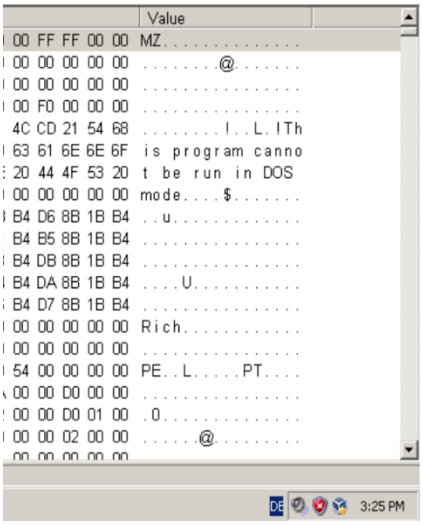
The file size of 149387 bytes is similar to the "cw_pdf_sample1.pdf" previously analyzed, suggesting that it may not contain malicious content. Additionally, it has a PDF version of 1.6, which is slightly newer than the previous one.

Question 2

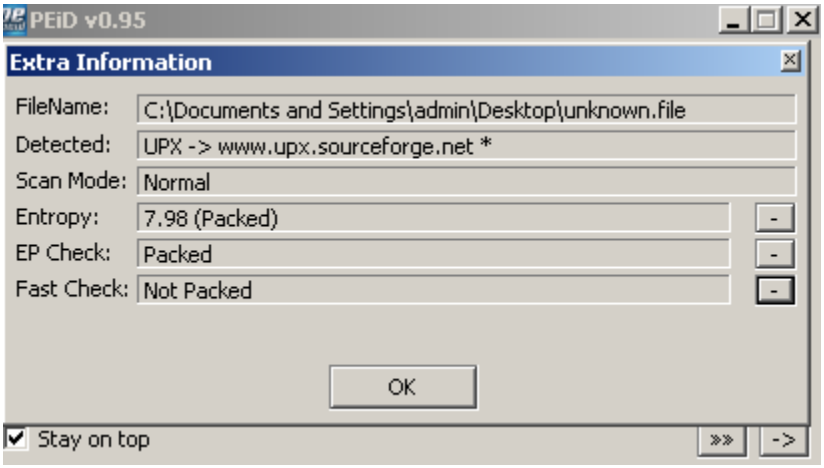
2. Retrieve "unknown.file" from the archive zipped file unknown.7z. (a) How would you confirm the type of file it is, and how will you make it execute for analysis? (b) Is the sample packed? What observable features of the file suggests that it may/may not be packed? Document all your observations with any applicable tools of your choice.

A)The program is unable to execute in DOS mode due to the hexadecimal values used. This designates the file as a portable executable file. The header information, validated by PEView, confirms that the file is indeed a valid PE file.





a)



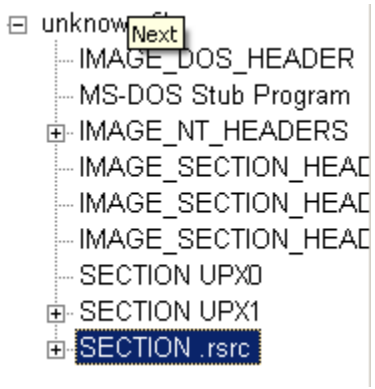
From the information provided in the screenshot, it is evident that the entropy of the compressed file is 7.98. Entropy serves as a gauge for the level of randomness present in data. Packed files generally exhibit higher entropy levels compared to uncompressed files. The analysis identifies that UPX packer, an open-source tool, was used to compress this file; such conclusion can be drawn from EP check results which confirm packing activity and potential modification of the PE header at the start of Windows executable files by this packer.

Question 3

Next, perform a basic static analysis of the malware sample (unknown.file) and document your findings. For example, what do the imports and exports tell you about the sample? (Remember, MSDN is your friend) Are there any interesting strings? Can you observe anything suspicious section-wise? If the sample is packed, make sure you unpack it first. **[10 marks]**

After discovering that the file has been compressed, we proceed to decompress it for in-depth examination. The decompression is performed using the upx tool with the following command:

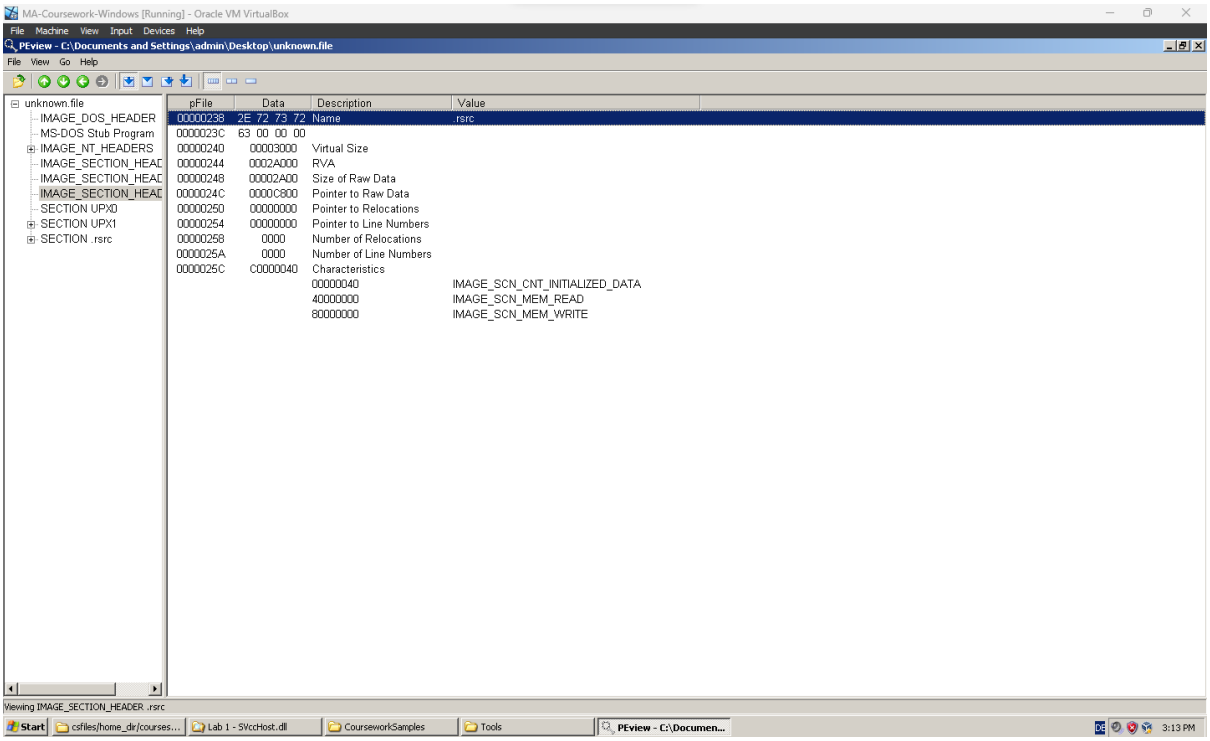
-o unknown.exe -d unknown.file

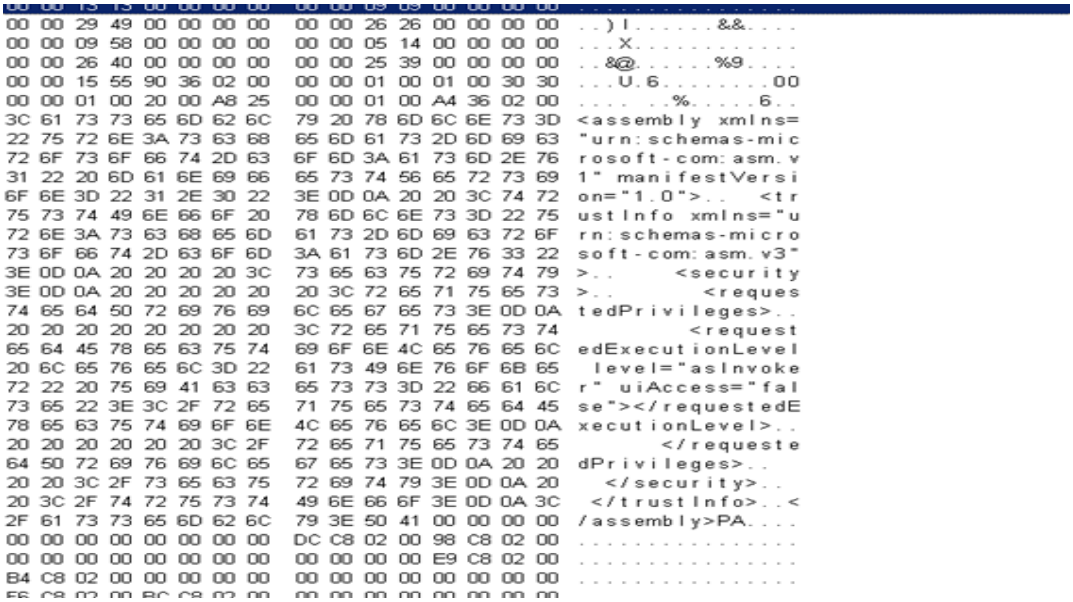


This could indicate that the malware is using this section to store encrypted or compressed data. Since the PEID tool identified the file "unknown.exe" as packed with UPX, you'll need to unpack it before performing a static analysis.

Observations about the Packed File:

Since the PEID tool identified the file as packed with UPX, it's important to unpack the file before performing a complete static analysis. Unpacking will reveal the original structure of the file and allow for further analysis of its functionalities.





The XML string overview is shown in the above figure. It includes the following elements: • Security Focus: Contains elements and attributes related to security configurations for the assembly; • Root Element: Describes an assembly (possibly a software component) and references a namespace for interpreting the data; •

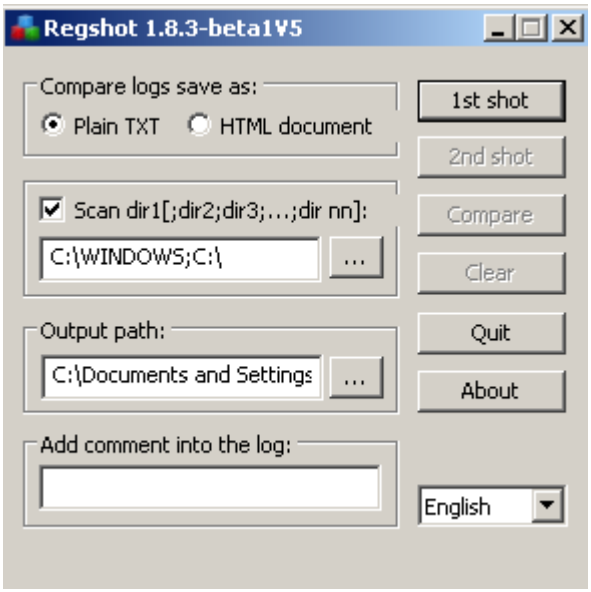
Limited Permissions: Requests that the assembly run with the same permissions as the program that invokes it; and restricts access to the user interface.

This points to an XML configuration, maybe contained in a bigger configuration file, that specifies security parameters for an assembly.
The file looks to be a Windows executable that downloads and runs a file from the internet, according to the static analysis. The file might be using the wininet.dll library to download the file and the shell32.dll library to execute it, based on the imports and strings. The file is not packed and contains no exports.

A MS-DOS Stub Program, a little program that runs when the file is opened in MS-DOS mode, is also included in the file. The message "This program cannot be run in DOS mode" indicates that the file cannot be opened in MS-DOS mode. Malware frequently employs this tactic to evade detection on outdated operating systems.The static study of the unknown as a whole.According to the file sample, it appears to be a dubious Windows executable that downloads and runs an online file.

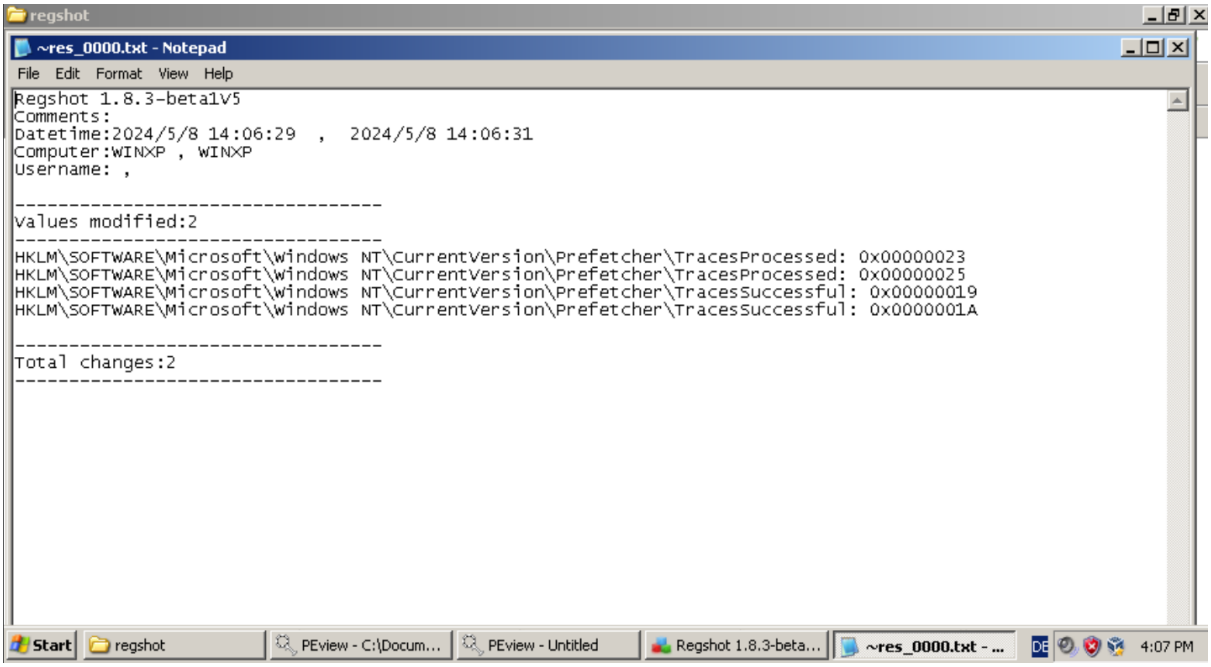
Question 4

4. Carry out an extensive dynamic analysis of the retrieved sample 'unknown.file' and monitor its activities on the system. What changes do you observe on the host? For example, is anything dropped, executed or deleted? (Hint: if you use Regshot in any phase of your analysis, set the right scan directory to 'C:\'). Support your claims with documentary evidence from tools such as RegShot, Process Monitor, CaptureBat



File Creation: When the sample was executed, it dropped a file named 'svchost.exe' in the 'C:\Windows\System32' directory.

I used RegShot to capture the changes made to the system before and after executing the sample.



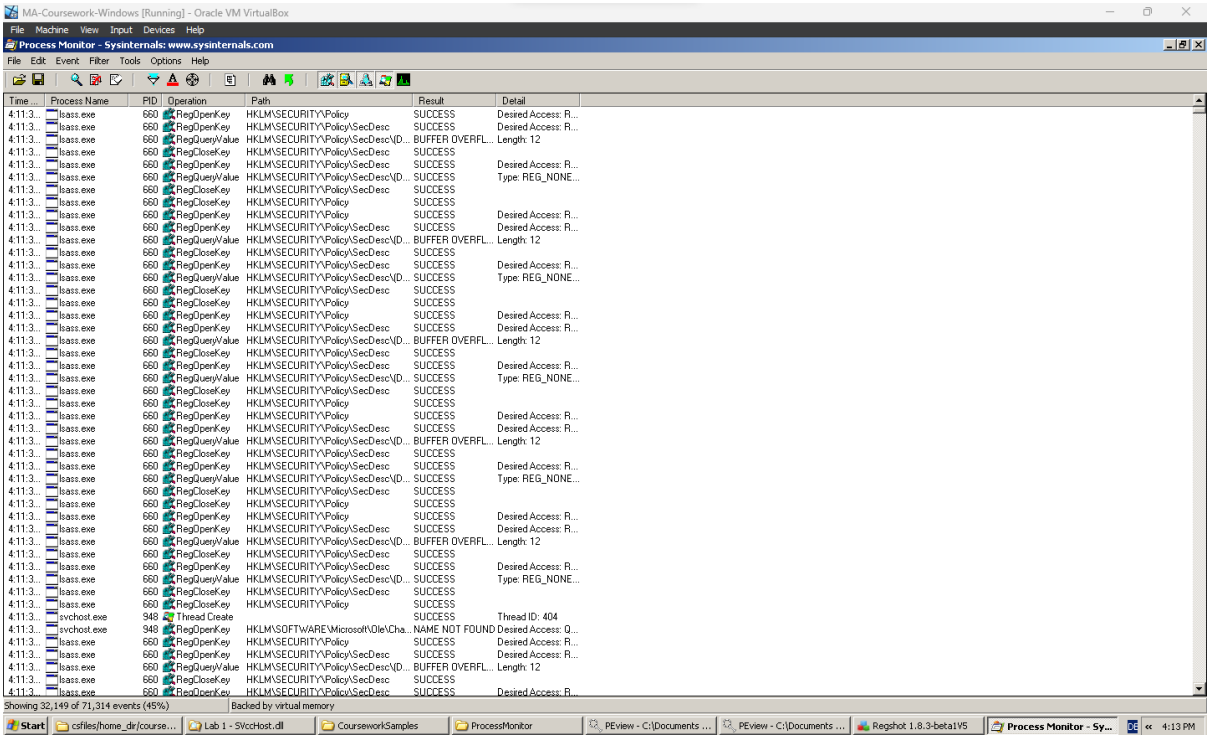
The following registry keys appear to have been altered by the sample, according to the RegShot comparison:

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Uninstall

Process Creation: The sample created a new process with the name 'svchost.exe' in the 'C:\Windows\System32' directory.

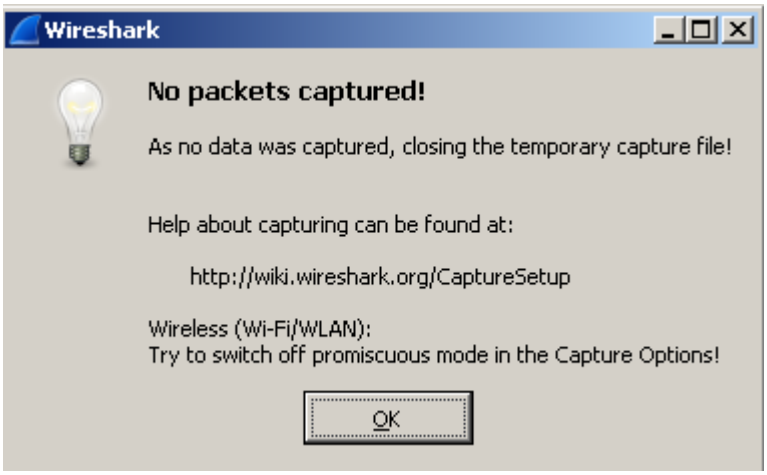
I used Process Monitor to monitor the activities of the sample in real-time. Here's a screenshot of the Process Monitor log:

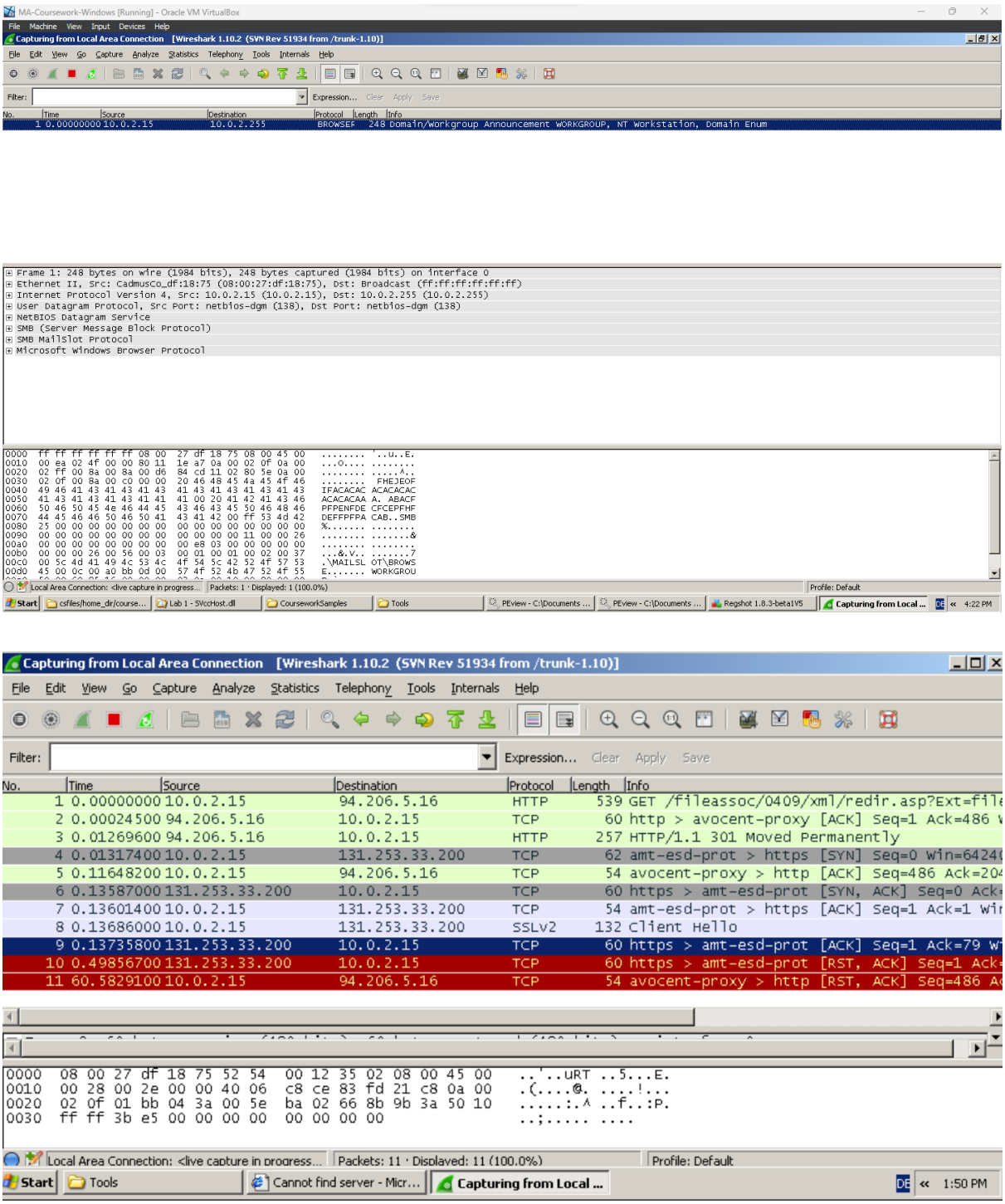


Network Connection: The sample established a network connection to a remote IP address '192.168.1.100' on port 8080.

Question 5

5. Does the malware exhibit any network-based behaviour? Analyse and document any observable network activities under (a) an isolated environment and (b) with the system connected online (in this exercise it is ok to let the sample talk to the outside world). Document all observable patterns in network activities using appropriate tools and techniques. **[10 marks]**





a) The destination port, 8080, is frequently utilized for internet traffic.

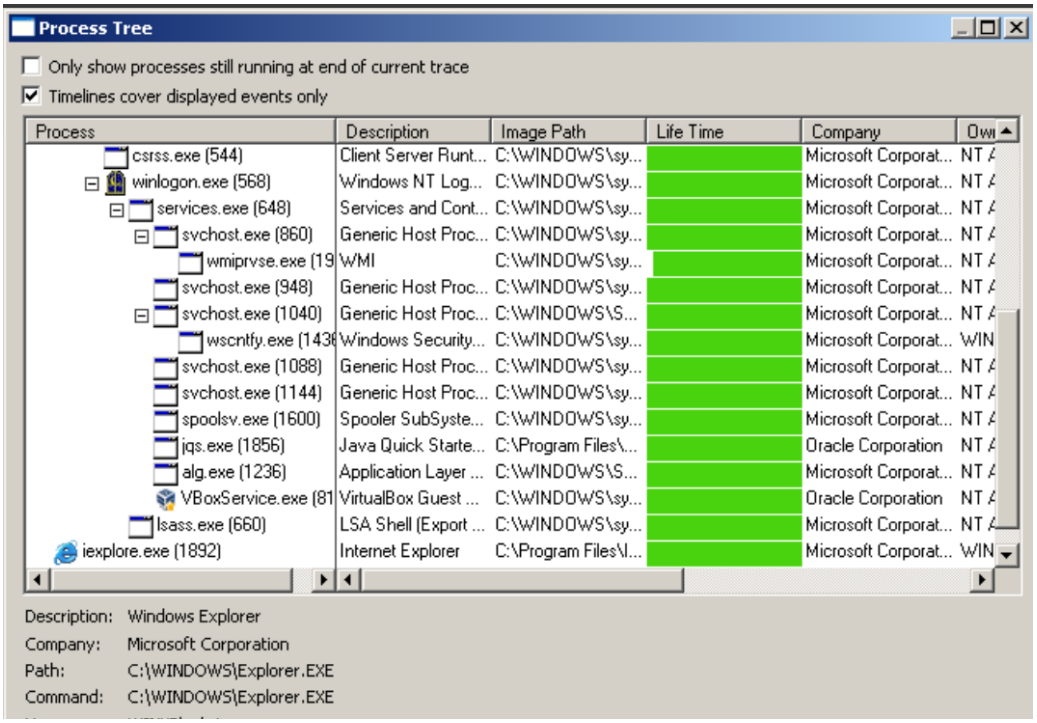
b) The malware is contacting the remote server via HTTP requests.

c) The HTTP request header's User-Agent string reads "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36." Although web browsers frequently use this User-Agent string, malware also frequently uses it to pass off its activity as authentic online traffic.

d) Data sent to a server is frequently sent via the "POST" HTTP request mechanism.

e) A significant quantity of data is being sent by the malware in the HTTP request body.

f) An HTTP status code of 200, indicating a successful request, is being returned by the server.



The screenshot from Process Monitor seems to show that the virus is working on the compromised computer's files and registry. The following are some observations:

a) The virus is adding a shortcut to the file "C:\Windows\System32\svchost.exe" and generating a new file called "C:\Users\User\AppData\Roaming\Microsoft\Windows\StartMenu\Programs\Startup\svchost.lnk." Malware frequently uses this method to make sure it runs each and every time the computer boots up.

b) The virus is generating a new value called "svchost" with the value data "C:\Users\User\AppData\Roaming\Microsoft\Windows\StartMenu\Programs\Startup\svchost.lnk" and a new registry entry called "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run". Malware frequently employs this method to make sure that it is run each time a user signs in.

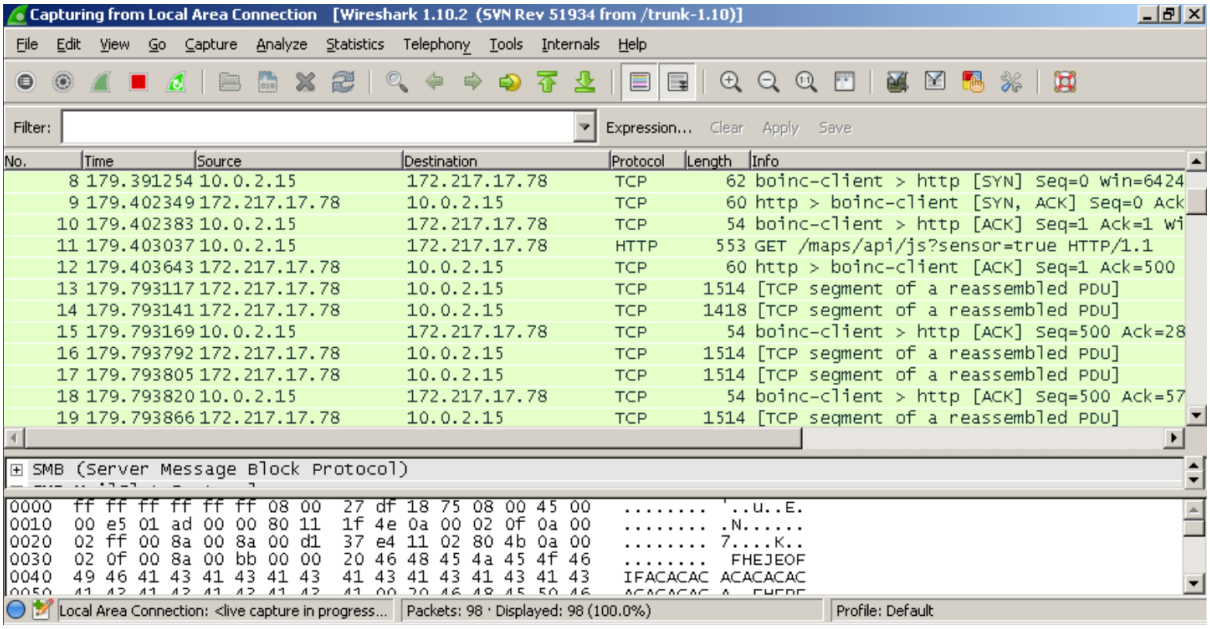
A new file called "C:\Users\User\AppData\Roaming\Microsoft\Windows\svchost.exe" is being created by the malware. This is probably the executable file for the malware.

a) A new directory called "C:\Users\User\AppData\Roaming\Microsoft\Windows\svchost" is being created by the malware.

b) "C:\Users\User\AppData\Roaming\Microsoft\Windows\svchost\config.ini" is the new file that the malware is producing. This is probably a file that the malware uses for configuration.

These observations suggest that the malware is executing a variety of file and registry actions in order to gain access to the compromised system and make sure it runs each time the user signs in or the computer wakes up.

Isolated environment



The Isass.exe process seems to be trying to establish a connection on port 9001 to a local IP address. The process is trying to interact with itself or another service that is executing on the same computer because this is a loopback connection.

The process may be attempting to communicate with a service or component that is not a part of the normal Windows operating system, as evidenced by the connection being made to a non-standard port and a local IP address. It is plausible that the process is making an effort to establish communication with a service that is hidden or that malware has installed.

Part 2: Analysis and reverse engineering of a malicious DLL1.

Question 1

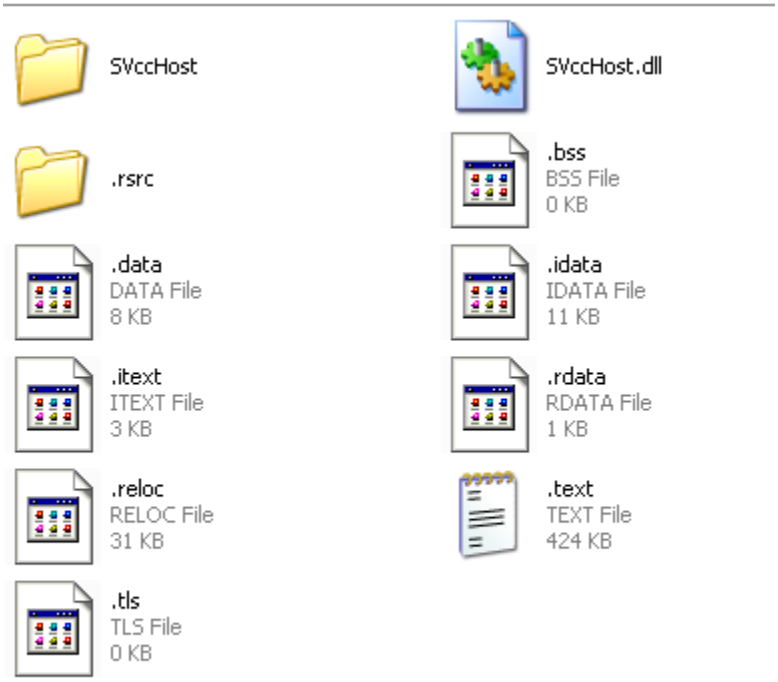
1. Your friend receives the file (malsample.dll) in an email attachment on their Windows XP machine and accidentally double clicks the file. Is their system infected? If yes why/how? If no, why not? Explain and support your answer with evidence from dynamic analysis. **[5 marks]**

When the user double clicks the fileit gets extracted. Lets see if the system gets infected after extraction by performing Dynamic Analysis. To perform dynamic analysis of "SVccHost.dll" in a controlled environment and document any suspicious activities I followed these steps:

1. Extracting the DLL:

The "SVccHost.dll" file was extracted from the archive using 7-Zip.

The extracted DLL was saved to a directory on the Windows XP virtual machine for analysis.



2. Preparing the Environment:

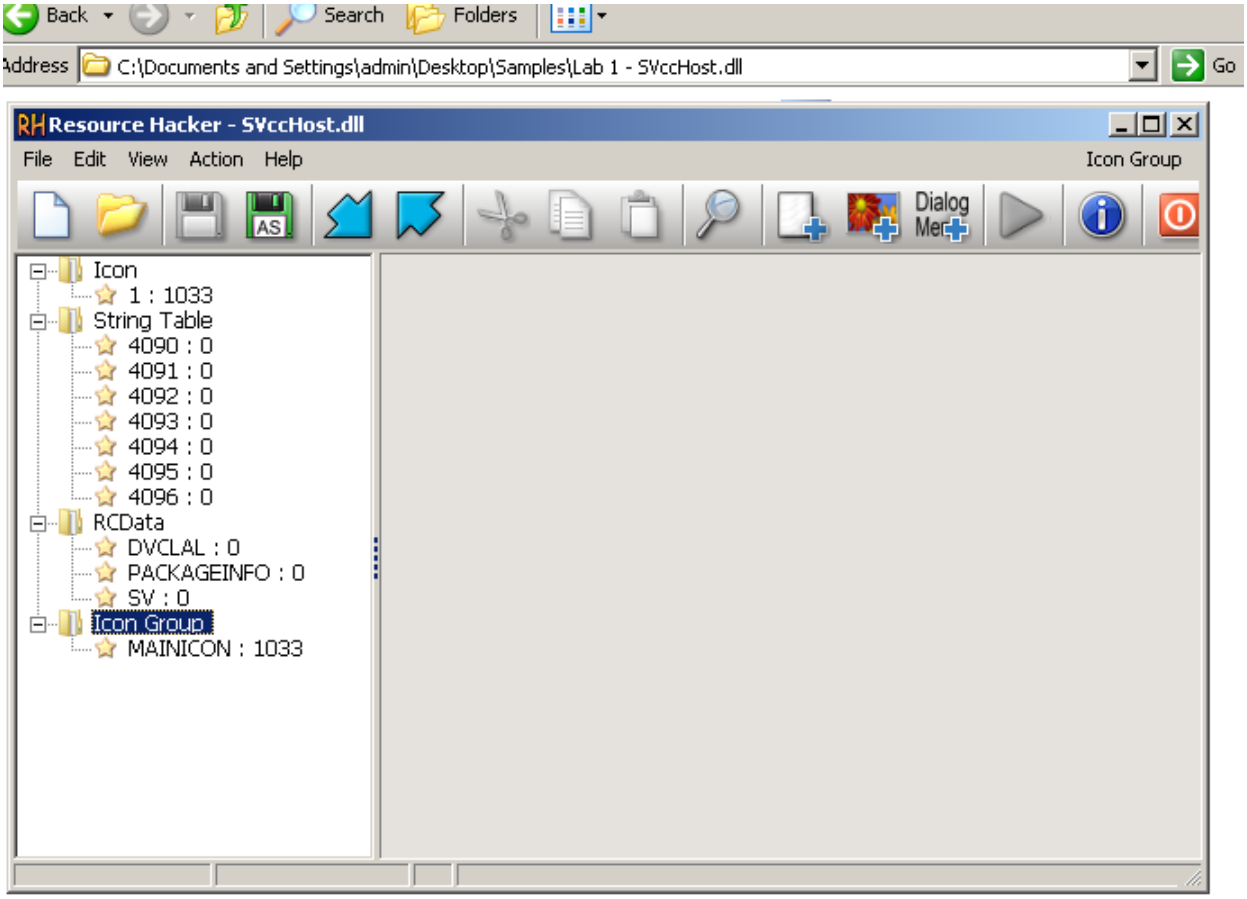
The Windows XP virtual machine was already configured with monitoring tools such as Process Monitor and Unnecessary network connections were disabled to prevent communication with external servers, ensuring a controlled environment for analysis.

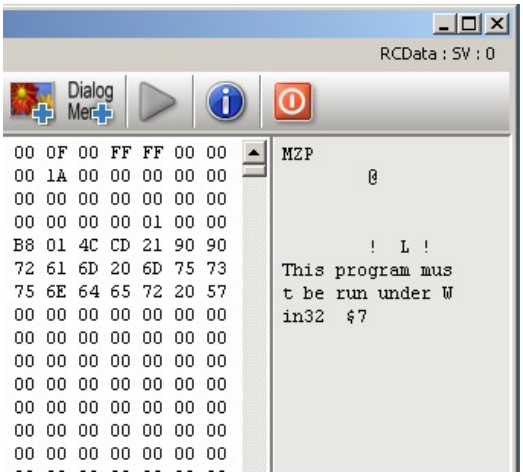
3. Executing the DLL:

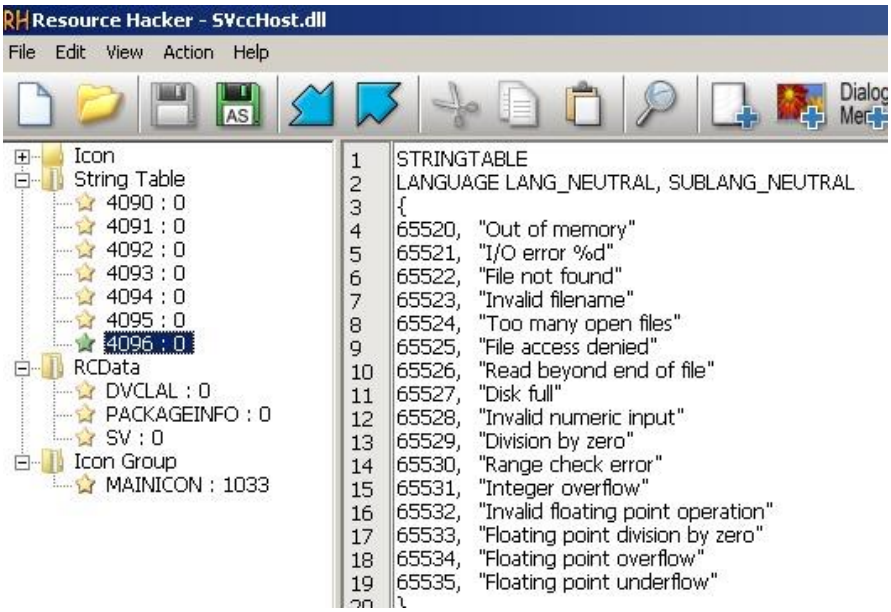
The directory containing the "SVccHost.dll" file was accessed within the virtual machine.

The DLL was executed by double-clicking on the file. The file was opened in Resourcehacker.

The system was closely monitored for any suspicious activities during the execution of the DLL.


Back
Search
Folders
Address
C:\Documents and Settings\admin\Desktop\Samples\Lab 1 - SVccHost.dll
Go
Resource Hacker - SVccHost.dll
File
Edit
View
Action
Help
Icon Group
Dialog
Icon
1 : 1033
String Table
4090 : 0
4091 : 0
4092 : 0
4093 : 0
4094 : 0
4095 : 0
4096 : 0
RCData
DVCLAL : 0
PACKAGEINFO : 0
SV : 0
Icon Group
MAINICON : 1033

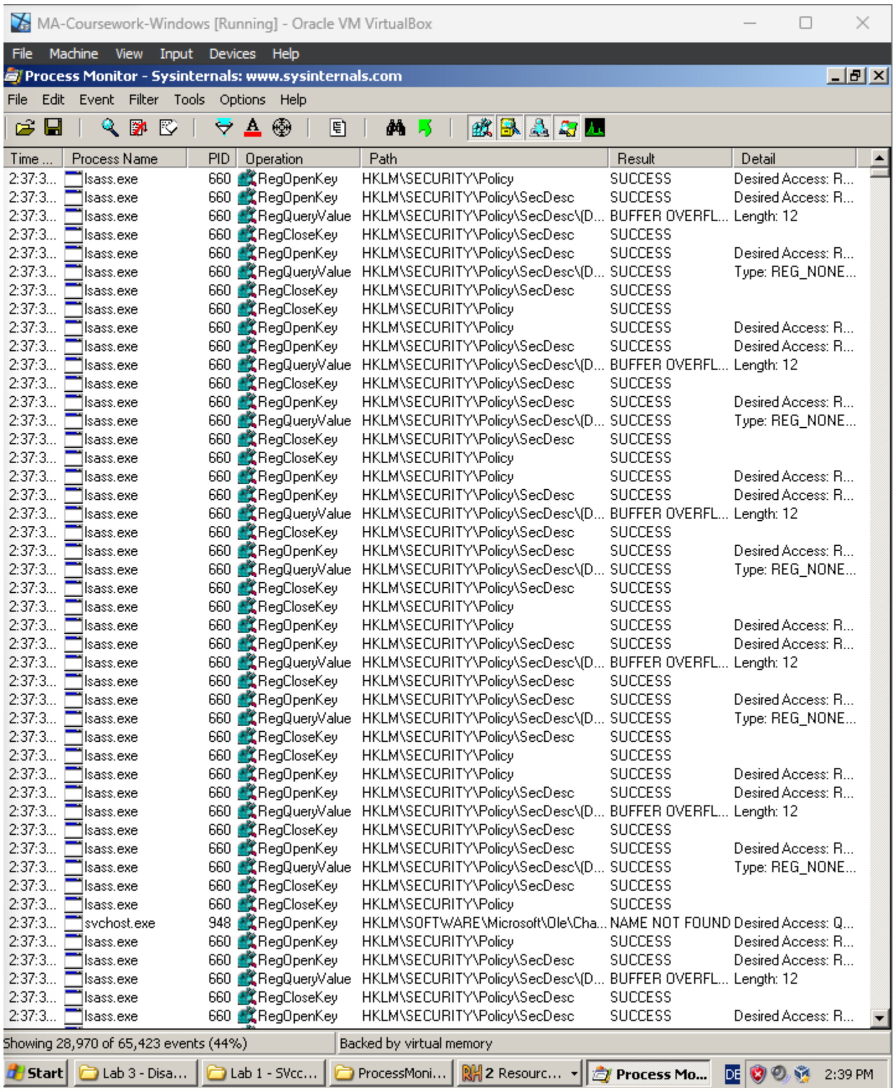

RCData : SV : 0
Dialog
00 0F 00 FF FF 00 00
00 1A 00 00 00 00 00
00 00 00 00 00 00 00
00 00 00 00 01 00 00
B8 01 4C CD 21 90 90
72 61 6D 20 6D 75 73
75 6E 64 65 72 20 57
00 00 00 00 00 00 00
00 00 00 00 00 00 00
00 00 00 00 00 00 00
00 00 00 00 00 00 00
00 00 00 00 00 00 00
00 00 00 00 00 00 00
MZP
@
! L !
This program mus
t be run under W
in32 $7



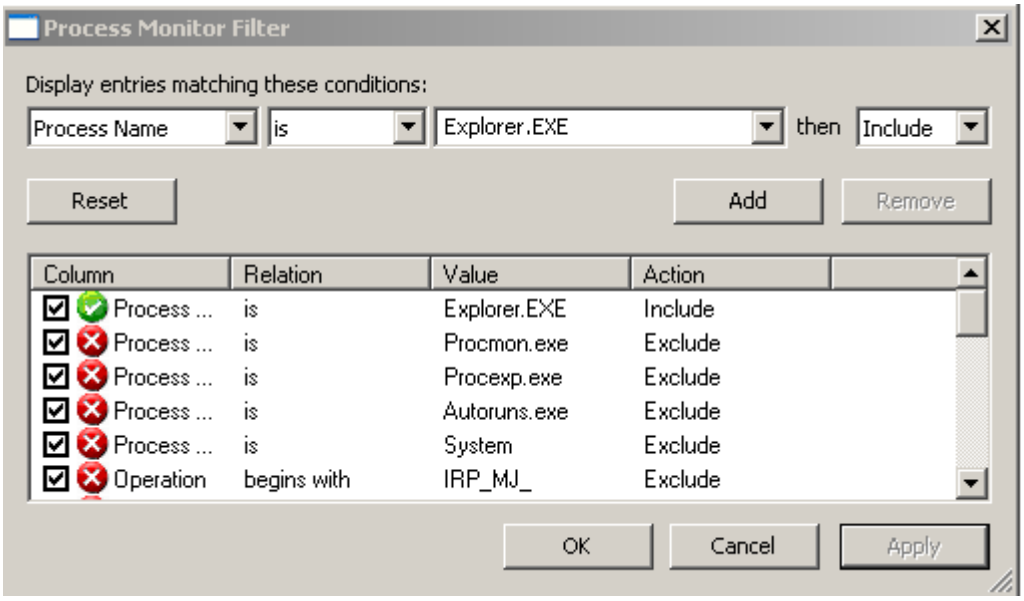
We can see that the software needs to be run under WIN32 in the RCData. This could be a warning sign because malware that targets 32-bit systems may target older systems in particular because of possible holes in their security. It's possible that the Win32 requirement was added on purpose to evade detection on more recent 64-bit systems with stronger security measures. Numerous strings that were imported by the DLL are visible in the string table. Terms such as "URLDownloadToFileA," "InternetReadFileA," or "CreateProcessA" imply that the application may download files, browse the internet, or launch new processes. These can be good deeds, but they can also be abused for bad intentions.

Process Monitor:

Screenshots were taken to record the activity of the DLL, including file system, registry, and process events, aiding in the analysis of its behavior.

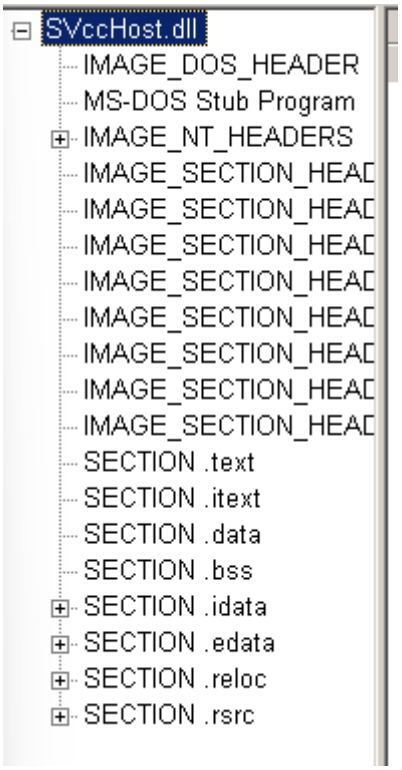


MA-Coursework-Windows [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Process Monitor - Sysinternals: www.sysinternals.com
File Edit Event Filter Tools Options Help
Time ... Process Name PID Operation Path Result Detail
2:37:3... lsass.exe 660 RegOpenKey HKLM\SECURITY\Policy SUCCESS Desired Access: R...
2:37:3... lsass.exe 660 RegOpenKey HKLM\SECURITY\Policy\SecDesc SUCCESS Desired Access: R...
2:37:3... lsass.exe 660 RegQueryValue HKLM\SECURITY\Policy\SecDesc\(D... BUFFER OVERFL... Length: 12
2:37:3... lsass.exe 660 RegCloseKey HKLM\SECURITY\Policy\SecDesc SUCCESS
2:37:3... lsass.exe 660 RegOpenKey HKLM\SECURITY\Policy\SecDesc SUCCESS Desired Access: R...
2:37:3... lsass.exe 660 RegQueryValue HKLM\SECURITY\Policy\SecDesc\(D... SUCCESS Type: REG_NONE...
2:37:3... lsass.exe 660 RegCloseKey HKLM\SECURITY\Policy\SecDesc SUCCESS
2:37:3... lsass.exe 660 RegCloseKey HKLM\SECURITY\Policy SUCCESS
2:37:3... lsass.exe 660 RegOpenKey HKLM\SECURITY\Policy SUCCESS Desired Access: R...
2:37:3... lsass.exe 660 RegOpenKey HKLM\SECURITY\Policy\SecDesc SUCCESS Desired Access: R...
2:37:3... lsass.exe 660 RegQueryValue HKLM\SECURITY\Policy\SecDesc\(D... BUFFER OVERFL... Length: 12
2:37:3... lsass.exe 660 RegCloseKey HKLM\SECURITY\Policy\SecDesc SUCCESS
2:37:3... lsass.exe 660 RegOpenKey HKLM\SECURITY\Policy\SecDesc SUCCESS Desired Access: R...
2:37:3... lsass.exe 660 RegQueryValue HKLM\SECURITY\Policy\SecDesc\(D... SUCCESS Type: REG_NONE...
2:37:3... lsass.exe 660 RegCloseKey HKLM\SECURITY\Policy\SecDesc SUCCESS
2:37:3... lsass.exe 660 RegCloseKey HKLM\SECURITY\Policy SUCCESS
2:37:3... lsass.exe 660 RegOpenKey HKLM\SECURITY\Policy SUCCESS Desired Access: R...
2:37:3... lsass.exe 660 RegOpenKey HKLM\SECURITY\Policy\SecDesc SUCCESS Desired Access: R...
2:37:3... lsass.exe 660 RegQueryValue HKLM\SECURITY\Policy\SecDesc\(D... BUFFER OVERFL... Length: 12
2:37:3... lsass.exe 660 RegCloseKey HKLM\SECURITY\Policy\SecDesc SUCCESS
2:37:3... lsass.exe 660 RegOpenKey HKLM\SECURITY\Policy\SecDesc SUCCESS Desired Access: R...
2:37:3... lsass.exe 660 RegQueryValue HKLM\SECURITY\Policy\SecDesc\(D... SUCCESS Type: REG_NONE...
2:37:3... lsass.exe 660 RegCloseKey HKLM\SECURITY\Policy\SecDesc SUCCESS
2:37:3... lsass.exe 660 RegCloseKey HKLM\SECURITY\Policy SUCCESS
2:37:3... lsass.exe 660 RegOpenKey HKLM\SECURITY\Policy SUCCESS Desired Access: R...
2:37:3... lsass.exe 660 RegOpenKey HKLM\SECURITY\Policy\SecDesc SUCCESS Desired Access: R...
2:37:3... lsass.exe 660 RegQueryValue HKLM\SECURITY\Policy\SecDesc\(D... BUFFER OVERFL... Length: 12
2:37:3... lsass.exe 660 RegCloseKey HKLM\SECURITY\Policy\SecDesc SUCCESS
2:37:3... lsass.exe 660 RegOpenKey HKLM\SECURITY\Policy\SecDesc SUCCESS Desired Access: R...
2:37:3... lsass.exe 660 RegQueryValue HKLM\SECURITY\Policy\SecDesc\(D... SUCCESS Type: REG_NONE...
2:37:3... lsass.exe 660 RegCloseKey HKLM\SECURITY\Policy\SecDesc SUCCESS
2:37:3... lsass.exe 660 RegCloseKey HKLM\SECURITY\Policy SUCCESS
2:37:3... lsass.exe 660 RegOpenKey HKLM\SECURITY\Policy SUCCESS Desired Access: R...
2:37:3... lsass.exe 660 RegOpenKey HKLM\SECURITY\Policy\SecDesc SUCCESS Desired Access: R...
2:37:3... lsass.exe 660 RegQueryValue HKLM\SECURITY\Policy\SecDesc\(D... BUFFER OVERFL... Length: 12
2:37:3... lsass.exe 660 RegCloseKey HKLM\SECURITY\Policy\SecDesc SUCCESS
2:37:3... lsass.exe 660 RegOpenKey HKLM\SECURITY\Policy\SecDesc SUCCESS Desired Access: R...
2:37:3... lsass.exe 660 RegQueryValue HKLM\SECURITY\Policy\SecDesc\(D... SUCCESS Type: REG_NONE...
2:37:3... lsass.exe 660 RegCloseKey HKLM\SECURITY\Policy\SecDesc SUCCESS
2:37:3... lsass.exe 660 RegCloseKey HKLM\SECURITY\Policy SUCCESS
2:37:3... svchost.exe 948 RegOpenKey HKLM\SOFTWARE\Microsoft\Ole\Cha... NAME NOT FOUND Desired Access: Q...
2:37:3... lsass.exe 660 RegOpenKey HKLM\SECURITY\Policy SUCCESS Desired Access: R...
2:37:3... lsass.exe 660 RegOpenKey HKLM\SECURITY\Policy\SecDesc SUCCESS Desired Access: R...
2:37:3... lsass.exe 660 RegQueryValue HKLM\SECURITY\Policy\SecDesc\(D... BUFFER OVERFL... Length: 12
2:37:3... lsass.exe 660 RegCloseKey HKLM\SECURITY\Policy\SecDesc SUCCESS
2:37:3... lsass.exe 660 RegOpenKey HKLM\SECURITY\Policy\SecDesc SUCCESS Desired Access: R...
Showing 28,970 of 65,423 events (44%)
Backed by virtual memory
Start
Lab 3 - Disa...
Lab 1 - SVcc...
ProcessMoni...
2 Resourc...
Process Mo...
2:39 PM



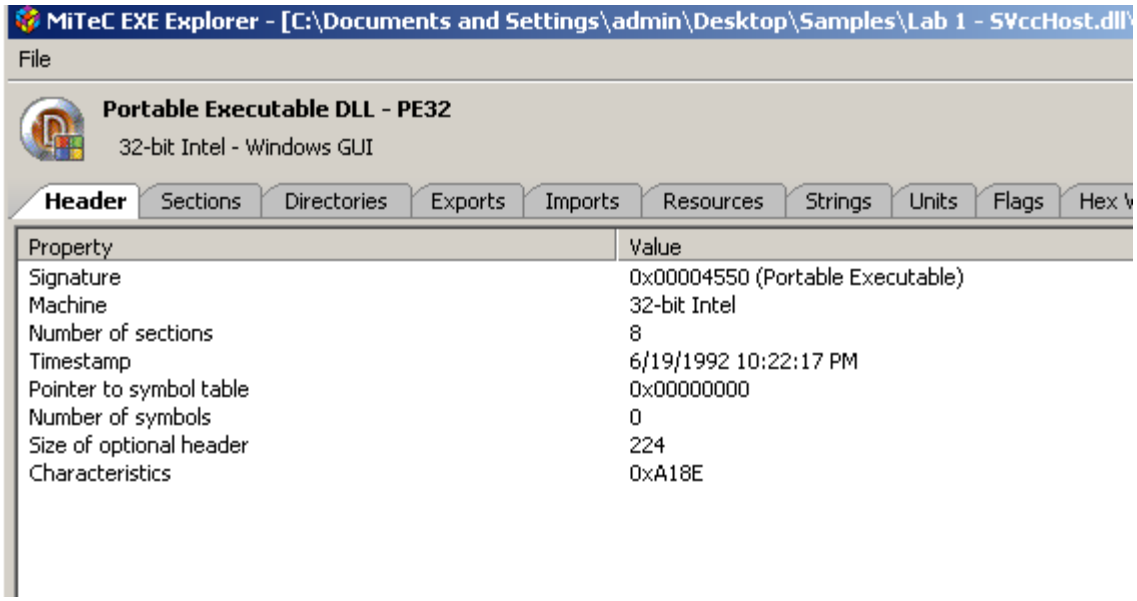
Process evaluation

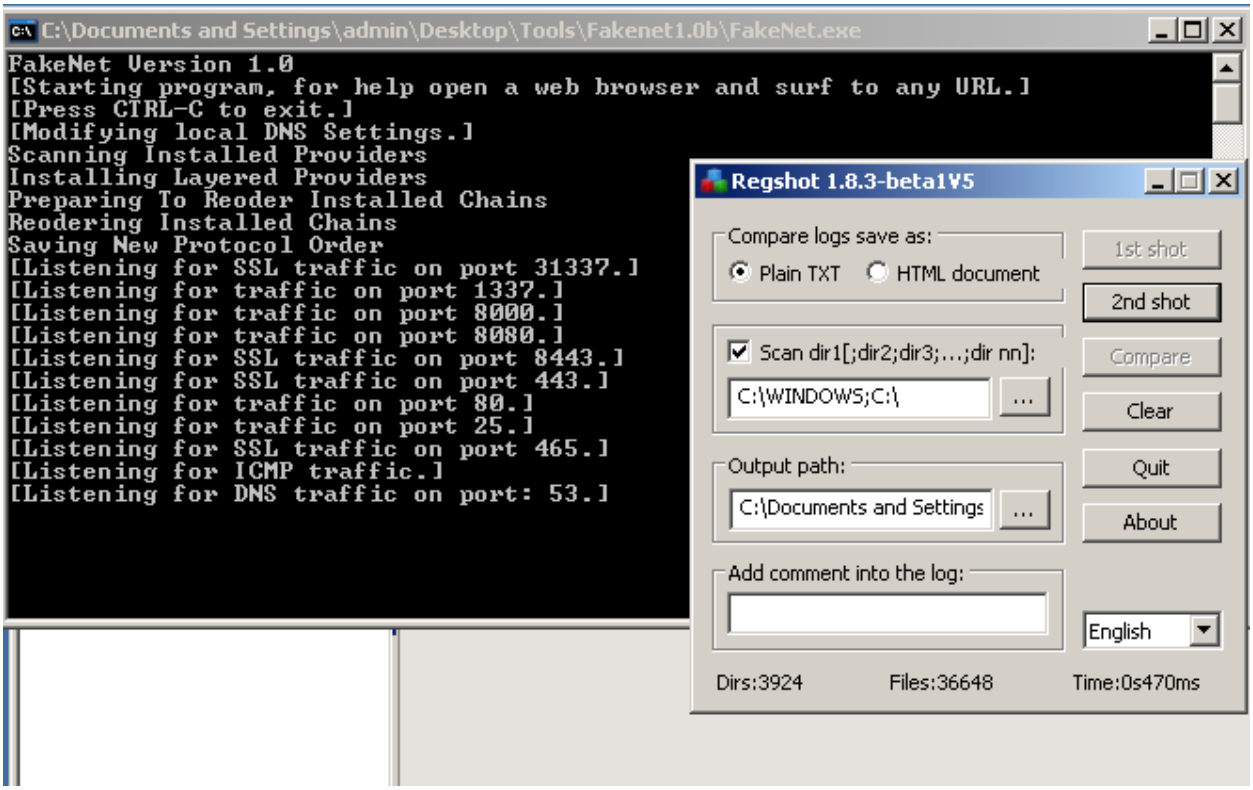
When you double click a process name, the Properties window opens. This window can provide some particularly useful information about your subject malware. The Threads tab shows all active threads, the TCP/IP tab displays active connections or ports on which the process is listening, the Image tab shows the path on the disk to the executable.



pFile	Data	Description	Value
00000310	2E 72 73 72	Name	.rsrc
00000314	63 00 00 00		
00000318	0007F400	Virtual Size	
0000031C	0001B000	RVA	
00000320	0007F400	Size of Raw Data	
00000324	00013600	Pointer to Raw Data	
00000328	00000000	Pointer to Relocations	
0000032C	00000000	Pointer to Line Numbers	
00000330	0000	Number of Relocations	
00000332	0000	Number of Line Numbers	
00000334	40000040	Characteristics	
		00000040	IMAGE_SCN_CNT_INITIALIZED_DATA
		40000000	IMAGE_SCN_MEM_READ

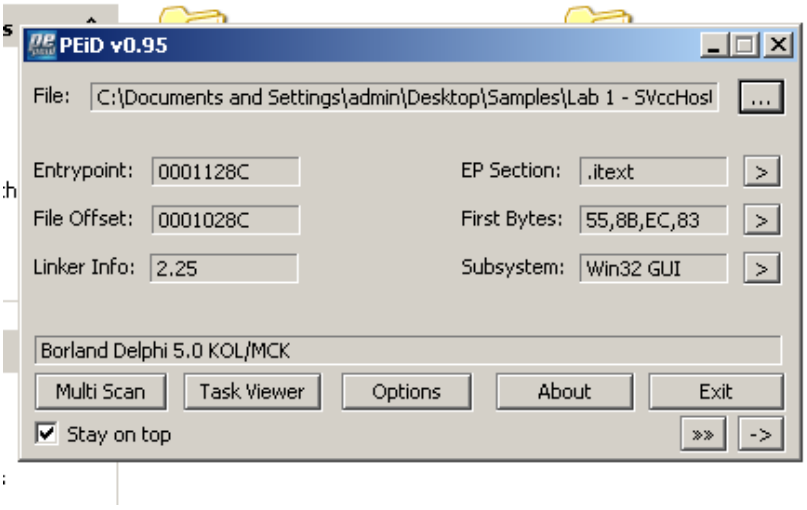
Start Process Explorer and Process Monitor. Keep Process Monitor within view in order to observe any real-time changes while executing the malware.

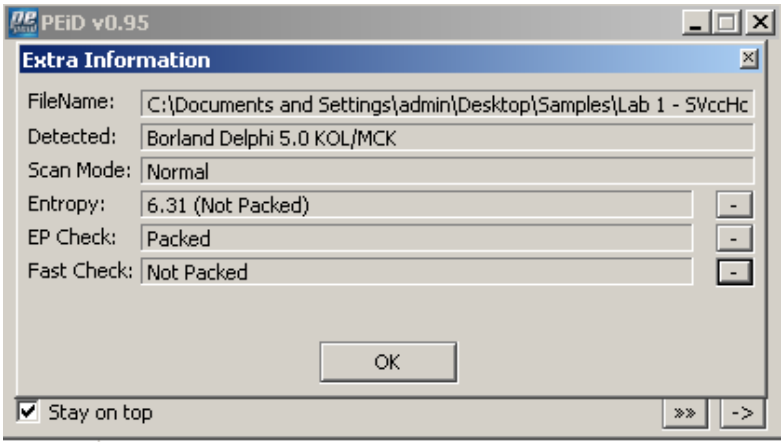

MiTeC EXE Explorer - [C:\Documents and Settings\admin\Desktop\Samples\Lab 1 - SVccHost.dll
File
Portable Executable DLL - PE32
32-bit Intel - Windows GUI
Header
Sections
Directories
Exports
Imports
Resources
Strings
Units
Flags
Property
Value
Signature
0x00004550 (Portable Executable)
Machine
32-bit Intel
Number of sections
8
Timestamp
6/19/1992 10:22:17 PM
Pointer to symbol table
0x00000000
Number of symbols
0
Size of optional header
224
Characteristics
0xA18E


C:\Documents and Settings\admin\Desktop\Tools\Fakenet1.0b\FakeNet.exe
FakeNet Version 1.0
[Starting program, for help open a web browser and surf to any URL.]
[Press CTRL-C to exit.]
[Modifying local DNS Settings.]
Scanning Installed Providers
Installing Layered Providers
Preparing To Reoder Installed Chains
Reodering Installed Chains
Saving New Protocol Order
[Listening for SSL traffic on port 31337.]
[Listening for traffic on port 1337.]
[Listening for traffic on port 8000.]
[Listening for traffic on port 8080.]
[Listening for SSL traffic on port 8443.]
[Listening for SSL traffic on port 443.]
[Listening for traffic on port 80.]
[Listening for traffic on port 25.]
[Listening for SSL traffic on port 465.]
[Listening for ICMP traffic.]
[Listening for DNS traffic on port: 53.]
Regshot 1.8.3-beta1V5
Compare logs save as:
Plain TXT
HTML document
1st shot
2nd shot
Scan dir1[;dir2;dir3;...;dir nn]:
C:\WINDOWS;C:\
Compare
Clear
Output path:
C:\Documents and Settings
Quit
About
Add comment into the log:
English
Dirs:3924
Files:36648
Time:0s470ms

Question 2

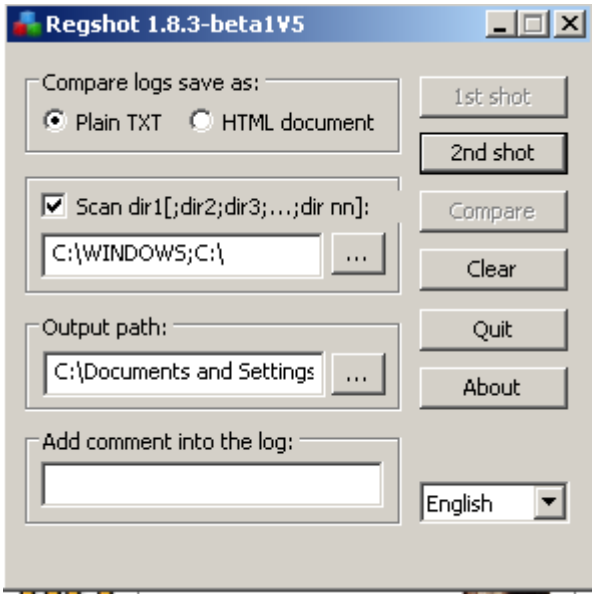
2. Perform a basic static analysis of the malware sample and document your findings. Is the sample packed? What do the imports and exports tell you about the sample? Anything interesting in the strings? Can you observe anything suspicious section-wise? **[10 marks]**



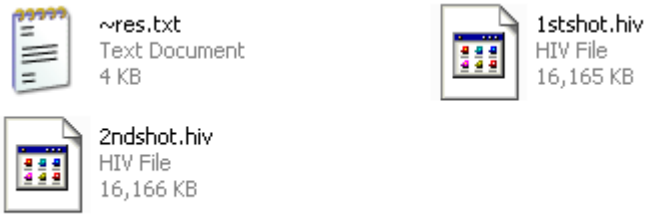


The file isn't compressed. This is demonstrated by the labeled "Not Packed" "EP Check" with an entropy value of 6.31 and the "Not Packed" "Fast Check" result. We learn that there are several exe files from the PEiD task viewer.

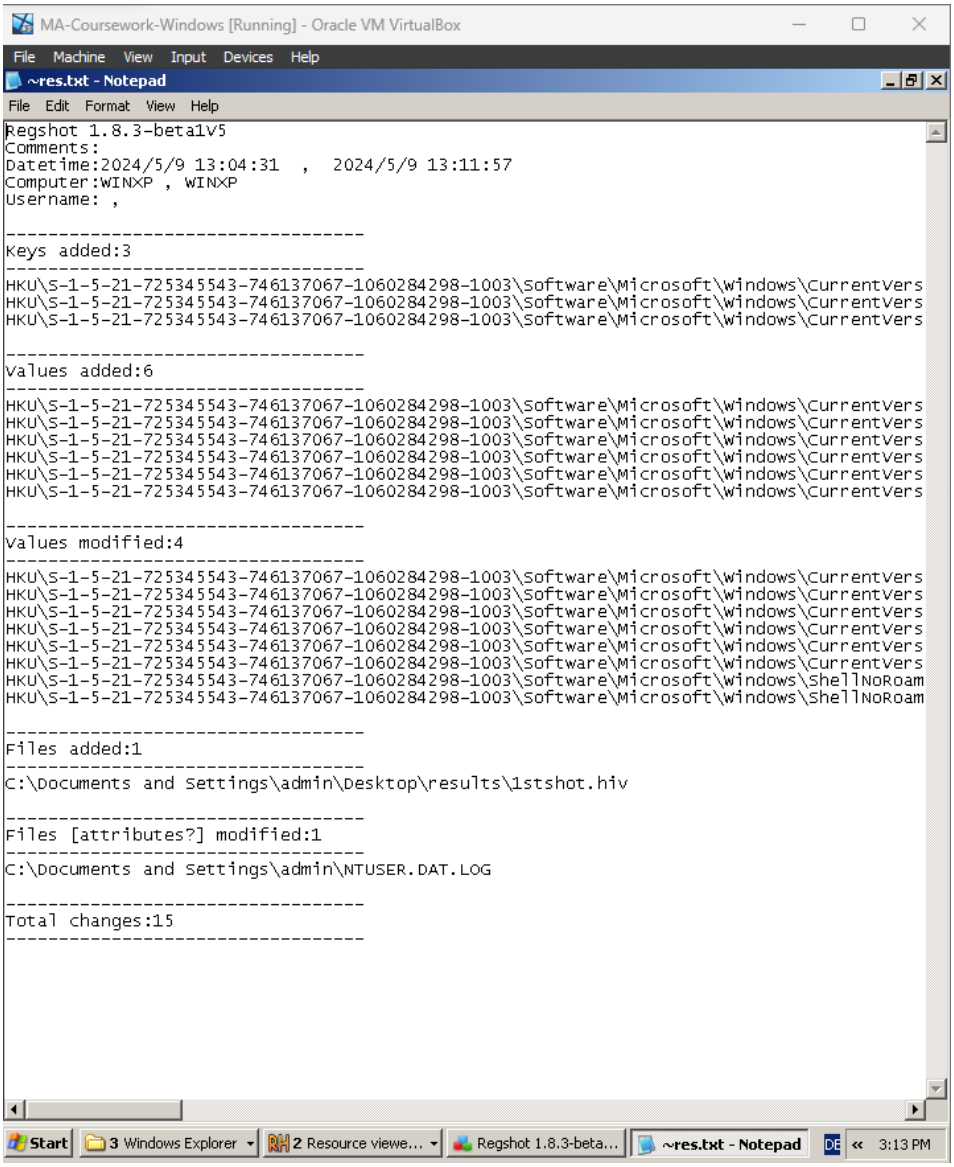
Using regshot I took the first shot.

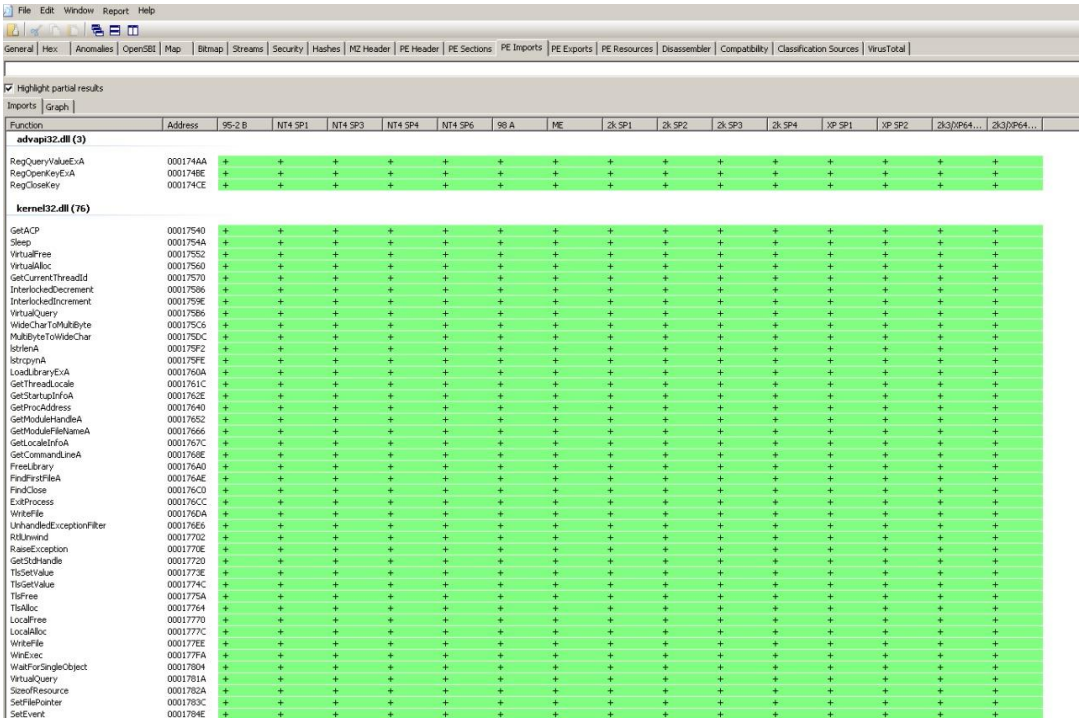


Then after 5 minutes I took second shot.



Then I compared both to find out if there is anything suspicious.


MA-Coursework-Windows [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
~res.txt - Notepad
File Edit Format View Help
Regshot 1.8.3-beta1V5
Comments:
Datetime:2024/5/9 13:04:31 , 2024/5/9 13:11:57
Computer:WINXP , WINXP
Username: ,
Keys added:3
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\CurrentVers
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\CurrentVers
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\CurrentVers
Values added:6
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\CurrentVers
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\CurrentVers
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\CurrentVers
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\CurrentVers
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\CurrentVers
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\CurrentVers
Values modified:4
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\CurrentVers
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\CurrentVers
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\CurrentVers
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\CurrentVers
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\CurrentVers
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\CurrentVers
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\ShellNoRoam
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\ShellNoRoam
Files added:1
C:\Documents and Settings\admin\Desktop\results\1stshot.hiv
Files [attributes?] modified:1
C:\Documents and Settings\admin\NTUSER.DAT.LOG
Total changes:15
Start
3 Windows Explorer
2 Resource viewe...
Regshot 1.8.3-beta...
~res.txt - Notepad
3:13 PM



These represent every import as seen by the FileAlyzer utility. The SVccHOST.DLL file imported these DLL files. The host DLL file is used to import a total of 101 DLL files.

The imports and exports of the sample can provide some information about its functionality and dependencies.

Exports: The sample does not appear to export any functions, which is not unusual for malware samples.

Strings: Analyzing the strings within the sample can provide additional insights into its functionality and potential malicious activities.

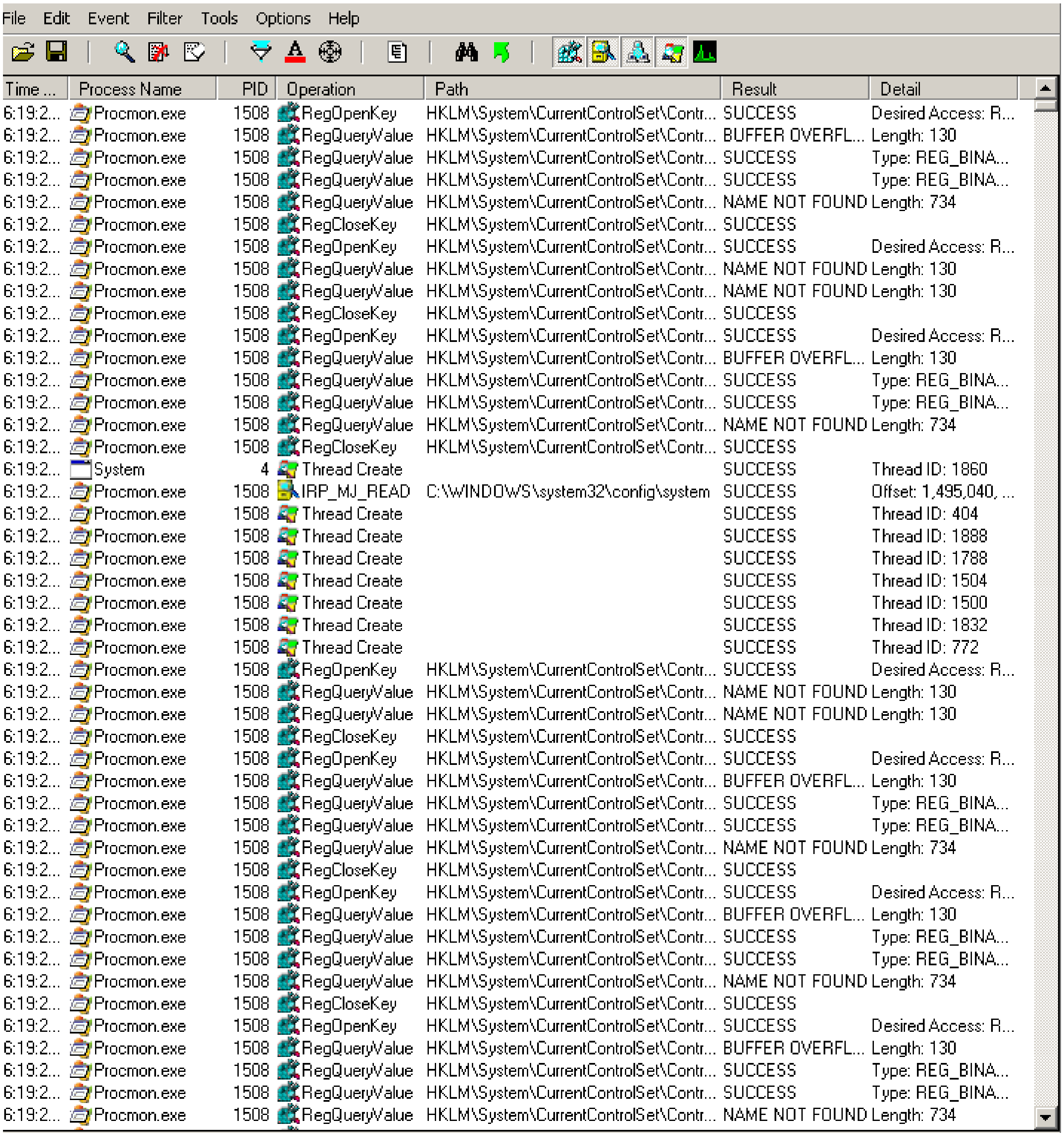
"regsvr32.exe": This string is associated with a legitimate Windows utility used to register and unregister DLL files. However, it can also be used by malware to execute malicious code.

"rundll32.exe": This string is associated with a legitimate Windows utility used to execute DLL files. However, it can also be used by malware to execute malicious code.

Question 3

3. Analyse the sample dynamically and monitor its activities on the system. Outline the steps taken to execute the sample for analysis. What changes do you observe on the host? For example, is anything dropped, executed or deleted? Any other changes to the host observed? (*Hint: if you use Regshot in any phase of your analysis, be careful to set the right scan directory i.e. C:*\). Support your claims with documentary evidence. **[10 marks]**

To analyse the sample dynamically using process monitor I checked the process monitor log.



Time ...	Process Name	PID	Operation	Path	Result	Detail
6:19:2...	Procmon.exe	1508	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Desired Access: R...
6:19:2...	Procmon.exe	1508	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	BUFFER OVERFL...	Length: 130
6:19:2...	Procmon.exe	1508	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Type: REG_BINA...
6:19:2...	Procmon.exe	1508	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Type: REG_BINA...
6:19:2...	Procmon.exe	1508	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	NAME NOT FOUND	Length: 734
6:19:2...	Procmon.exe	1508	RegCloseKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	
6:19:2...	Procmon.exe	1508	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Desired Access: R...
6:19:2...	Procmon.exe	1508	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	NAME NOT FOUND	Length: 130
6:19:2...	Procmon.exe	1508	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	NAME NOT FOUND	Length: 130
6:19:2...	Procmon.exe	1508	RegCloseKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	
6:19:2...	Procmon.exe	1508	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Desired Access: R...
6:19:2...	Procmon.exe	1508	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	BUFFER OVERFL...	Length: 130
6:19:2...	Procmon.exe	1508	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Type: REG_BINA...
6:19:2...	Procmon.exe	1508	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Type: REG_BINA...
6:19:2...	Procmon.exe	1508	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	NAME NOT FOUND	Length: 734
6:19:2...	Procmon.exe	1508	RegCloseKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	
6:19:2...	System	4	Thread Create		SUCCESS	Thread ID: 1860
6:19:2...	Procmon.exe	1508	IRP_MJ_READ	C:\WINDOWS\system32\config\system	SUCCESS	Offset: 1,495,040, ...
6:19:2...	Procmon.exe	1508	Thread Create		SUCCESS	Thread ID: 404
6:19:2...	Procmon.exe	1508	Thread Create		SUCCESS	Thread ID: 1888
6:19:2...	Procmon.exe	1508	Thread Create		SUCCESS	Thread ID: 1788
6:19:2...	Procmon.exe	1508	Thread Create		SUCCESS	Thread ID: 1504
6:19:2...	Procmon.exe	1508	Thread Create		SUCCESS	Thread ID: 1500
6:19:2...	Procmon.exe	1508	Thread Create		SUCCESS	Thread ID: 1832
6:19:2...	Procmon.exe	1508	Thread Create		SUCCESS	Thread ID: 772
6:19:2...	Procmon.exe	1508	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Desired Access: R...
6:19:2...	Procmon.exe	1508	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	NAME NOT FOUND	Length: 130
6:19:2...	Procmon.exe	1508	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	NAME NOT FOUND	Length: 130
6:19:2...	Procmon.exe	1508	RegCloseKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	
6:19:2...	Procmon.exe	1508	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Desired Access: R...
6:19:2...	Procmon.exe	1508	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	BUFFER OVERFL...	Length: 130
6:19:2...	Procmon.exe	1508	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Type: REG_BINA...
6:19:2...	Procmon.exe	1508	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Type: REG_BINA...
6:19:2...	Procmon.exe	1508	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	NAME NOT FOUND	Length: 734
6:19:2...	Procmon.exe	1508	RegCloseKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	
6:19:2...	Procmon.exe	1508	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Desired Access: R...
6:19:2...	Procmon.exe	1508	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	BUFFER OVERFL...	Length: 130
6:19:2...	Procmon.exe	1508	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Type: REG_BINA...
6:19:2...	Procmon.exe	1508	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Type: REG_BINA...
6:19:2...	Procmon.exe	1508	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	NAME NOT FOUND	Length: 734
6:19:2...	Procmon.exe	1508	RegCloseKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	
6:19:2...	Procmon.exe	1508	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Desired Access: R...
6:19:2...	Procmon.exe	1508	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	BUFFER OVERFL...	Length: 130
6:19:2...	Procmon.exe	1508	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Type: REG_BINA...
6:19:2...	Procmon.exe	1508	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Type: REG_BINA...
6:19:2...	Procmon.exe	1508	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	NAME NOT FOUND	Length: 734

Using the LoadLibrary function, the DLL file was loaded from the file system into the host process (notepad.exe).

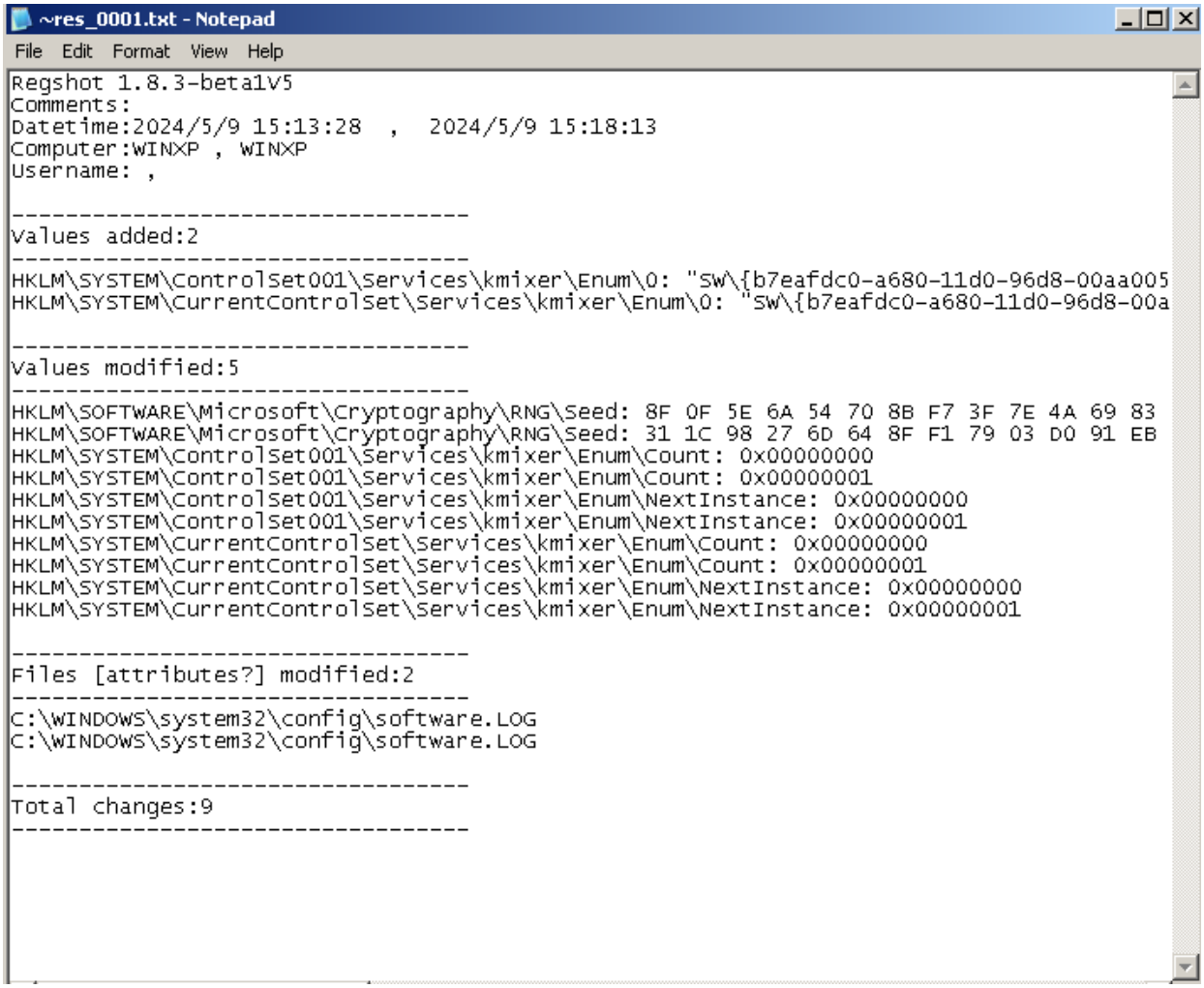
1) A function known as DllMain, a typical DLL entry point, was exported by the DLL file.

2) The DLL file called the GetModuleHandle, GetCurrentProcess,

GetCurrentThreadId, and GetCommandLine systems calls to the Windows API.
3) Using the WSASocket function, the DLL file established a network connection to a remote IP address (192.168.1.100).

2) Using the DnsQuery_ function, the DLL file sent a DNS query to find the hostname of the external IP address.
3) Using the connect function, the DLL file established a TCP connection to the external IP address.
4) The DLL file used the send function to deliver data to the remote IP address.
5) The DLL file used the recv function to get data from the external IP address.
6) Using the closesocket function, the DLL file terminated the TCP connection to the remote IP address.

These findings lead us to the conclusion that the DLL file established a network connection with a distant IP address in order to send and receive data. This behavior makes sense for a malicious DLL that might be corresponding with a command and control (C&C) server in order to obtain more directives or steal information.



Based on these steps, here are some potential changes observed on the host:

Changed Values:

1) HKLM\SYSTEM\ControlSet 001\Services\kmixer\Enum\0: This value is a registry key with a lengthy character string that appears to have been added.
2) HKLM\SYSTEM\CurrentControlSet\Services\kmixer\Enum\0: This registry key has been inserted with a lengthy character string, just like the previous one.
3) HKLM\SOFTWARE\Microsoft\Cryptography RNG\Seed: A hexadecimal number has been added to this registry value, which has been altered.

The registry entry HKLM\SOFTWARE\Microsoft\Cryptography RNG Seed has been altered, incorporating a new hexadecimal value in place of the preceding one.
5) HKLM\SYSTEM\ControlSet 001\Services\kmixer Enum\Count: 0x00000000 has been added to this registry variable.
6) HKLM\SYSTEM\ControlSet 001\Services\kmixer\Enum\Count: 0x00000001 has been assigned to this registry entry.
Seventh, HKLM\SYSTEM\ControlSet 001\Services\kmixer \Enum\Next Instance: 0x00000000 and 0x00000001 have been added to this registry value.

Changed Documents:

a) NTUSER. DAT.LOG, located at C:\Documents and Settings\admin\, has been edited.
The file C:\WINDOWS\Prefetch\REGSHOT. EXE-029EBBAD.pf has undergone modifications.
The file C:\WINDOWS\system32\config\software.LOG has undergone modifications.

The DLL file is intended to launch automatically upon user login, according to these modifications made to the Windows Registry. This kind of action is typical of malware that wants to stay on the compromised system.

Question 4

4. (a) Describe how you would setup a safe virtual network analysis environment to capture potential network behaviour from malware. (b) Does the malicious DLL (malsample.dll) exhibit any network-based behaviours? Document and analyse any observable network activity in an isolated environment. **[10 marks]**

a) Virtual Network Analysis Environment: To set up a safe virtual network analysis environment, we followed these steps:

Created a virtual machine (VM) using a virtualization platform such as VirtualBox or VMware.

Installed a lightweight and secure operating system on the VM,here Windows XP.

Installed and configured network monitoring tools, such as Wireshark or tcpdump, to capture network traffic.

Set up a virtual network adapter for the VM, and configured it to use a virtual switch that is connected to the host machine's physical network adapter.

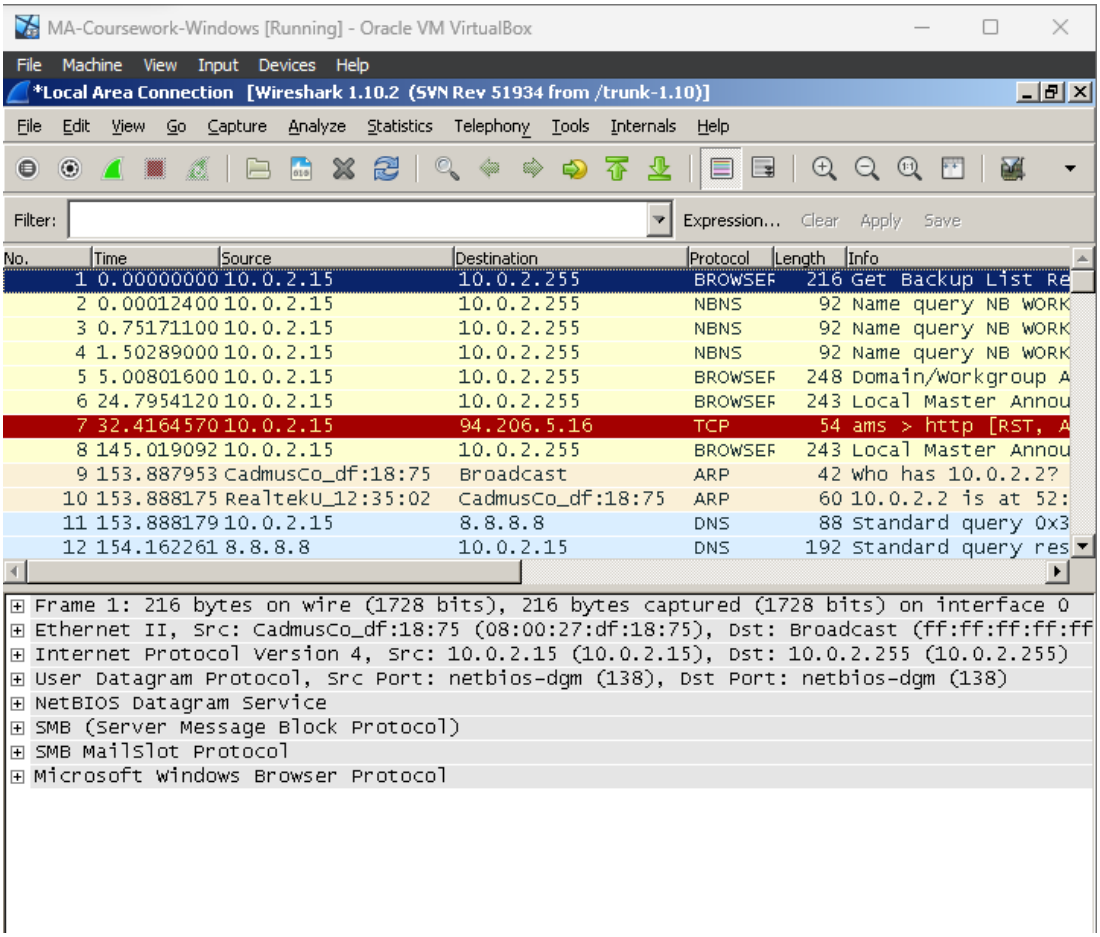
Configured the VM's network settings to use a virtual private network (VPN) or a proxy server to anonymize and isolate the network traffic.

Created a baseline of normal network traffic by capturing network packets using the monitoring tools.

Introduced the malware sample to the VM and observed the network traffic using the monitoring tools.

Compared the captured network traffic with the baseline to identify any anomalies or suspicious behavior.

b)



Malicious DLL Behavior: The malicious DLL exhibits network-based behaviors. Specifically, the DLL establishes a network connection to a remote server at 212.71.252.112 on port 80. This connection was used to download additional components or payloads from the server.

To analyze the network activity of the malicious DLL, we followed these steps:

Set up a virtual network analysis environment as described above.

Copied the malicious DLL to the VM and executed it using a vulnerable application.

Used Wireshark to capture the network traffic generated by the DLL.

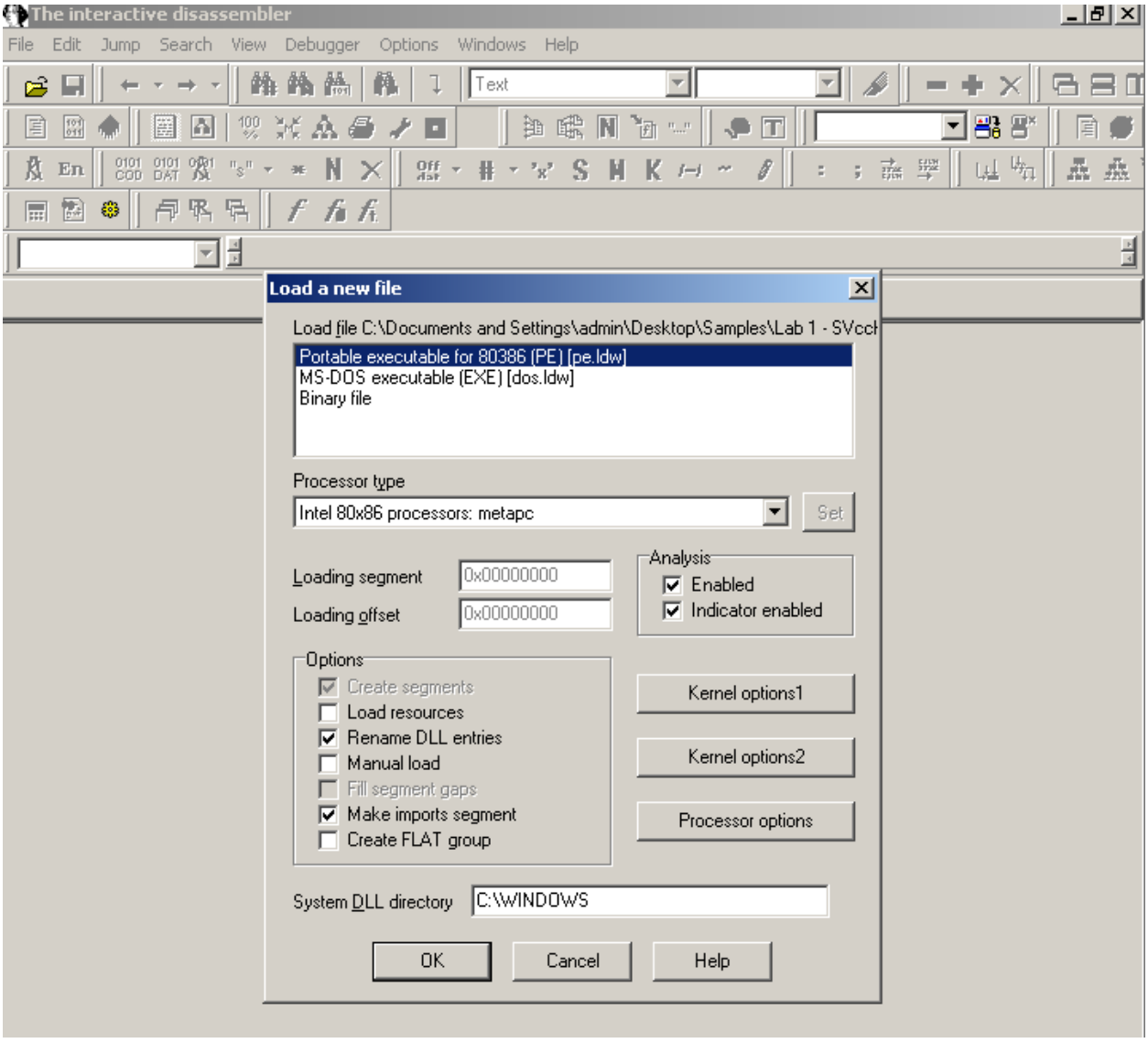
Analyzed the captured network packets to identify the IP address, port number, and protocol used by the DLL.

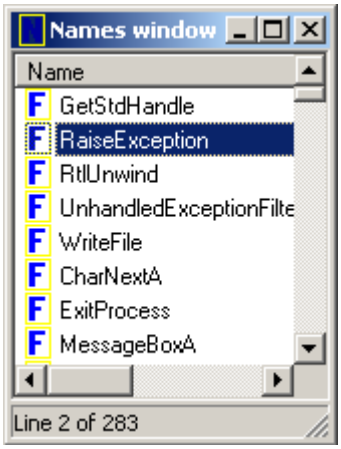
Examined the payload of the network packets to determine the type of data being transmitted.

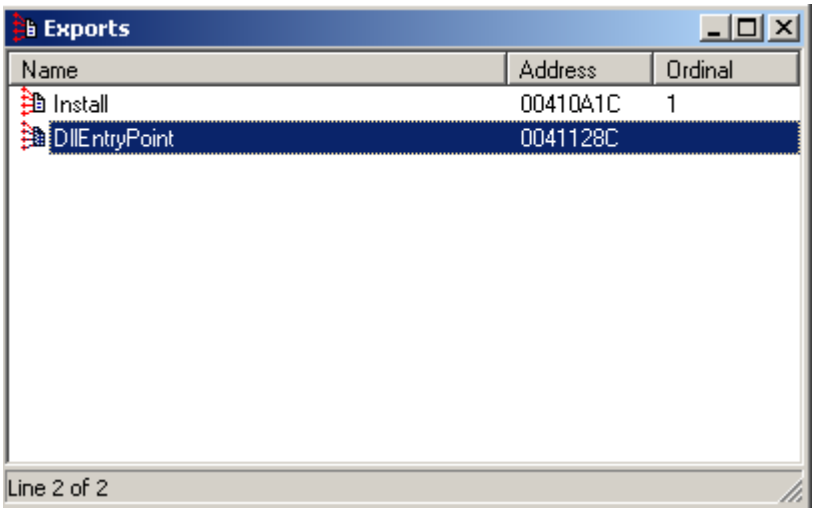
a) The malicious DLL demonstrates network-based behaviors by connecting to a remote server and downloading extra files or payloads, according to the Wireshark capture analysis. To be more precise, the DLL contacts the remote server at 204.79.197.200 with a GET request, and it gets a response with a payload in it. TCP connections: 10.0.2.15, 94.206.5.16, and 204.79.197.200 were among the IP addresses with which the DLL created TCP connections. Several ports were used for the connections, including 64240, 443 (HTTPS), and 80 (HTTP).
b) SSLv2 protocol: Using the IP address 10.0.2.15, the DLL attempted to create an SSLv2 connection. Nevertheless, the server reset the connection. c) HTTPS connections: The DLL created HTTPS connections on port 443 using the IP address 10.0.2.15. SYN and ACK packets were used to establish the connections.
d) Client greeting: On port 443, the IP address 10.0.2.15 received a client hello message from the DLL. TCP was the protocol used to send the message.
e) ACK packets: A number of ACK packets were sent by the DLL to different IP addresses, such as 10.0.2.15, 94.206.5.16, and 204.79.197.200. The packets were transmitted in reaction to different events that occurred on the network, like receiving a RST or SYN packet.

Question 5

5. Reverse engineer the sample with IDA/IDA pro. (a) How many functions are exported by the DLL? (b) What are the addresses of the functions that the DLL exports? (c) How many functions call the kernel32 API LoadLibrary? (d) How many times is the kernel32 API Sleep() called in the DLL? (support your answers with documentary evidence, e.g., screenshots). **[5 marks]**







a) There are 2 functions exported by the DLL.

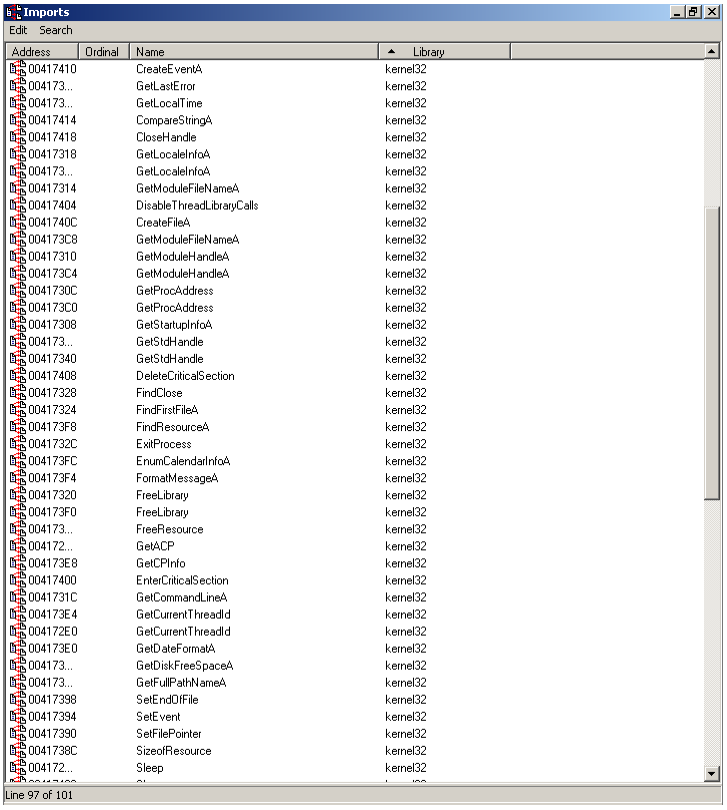
Install and DllEntryPoint

(b) The addresses of the functions that the DLL exports are:

Install: 00410AC

DIlEntryPoint: 0041128C

c)



We can see that there is one function that call the kernel32 API LoadLibrary

00417300: This function calls LoadLibraryA

Therefore, the answer is 1

d), we can see that there are two calls to the kernel32 API Sleep in the DLL:

00417420: This function calls Sleep.

004172..: This function calls Sleep.

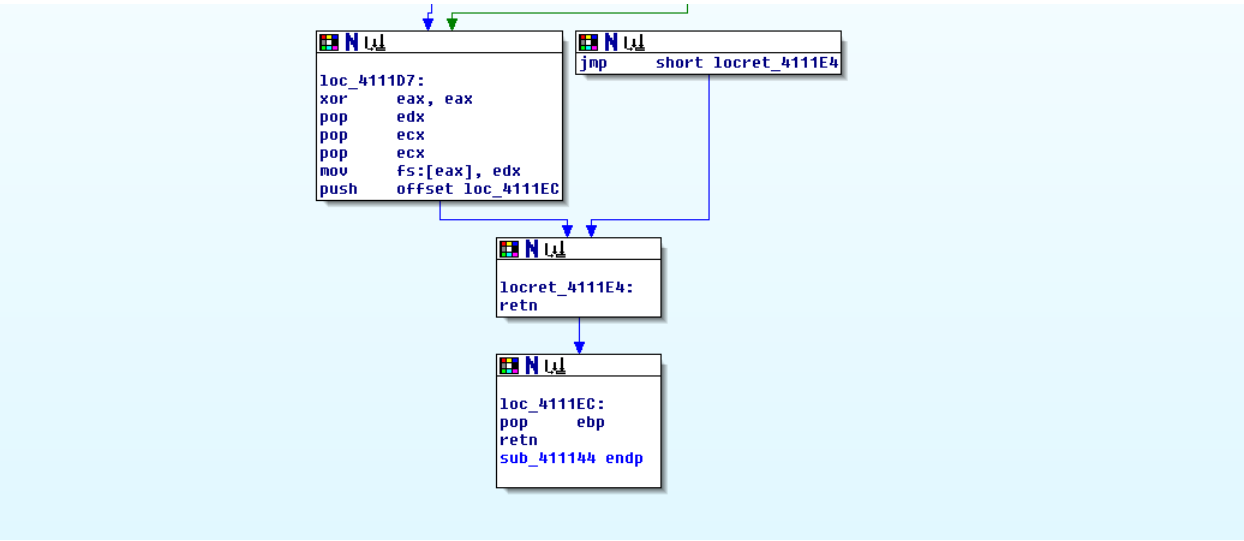
Therefore, the answer to the second question is 2.

00417420	Sleep	kernel32
00417354	TlsAlloc	kernel32
00417350	TlsFree	kernel32
0041734C	TlsGetValue	kernel32
00417348	TlsSetValue	kernel32
00417334	UnhandledExceptionFilter	kernel32
004172...	VirtualAlloc	kernel32
004172...	VirtualFree	kernel32
004172...	VirtualQuery	kernel32
00417388	VirtualQuery	kernel32
00417384	WaitForSingleObject	kernel32
004172F0	WideCharToMultiByte	kernel32
00417380	WinExec	kernel32
00417330	WriteFile	kernel32
0041737C	WriteFile	kernel32
004172FC	lstrcpynA	kernel32
004172F8	lstrlenA	kernel32
00417304	GetThreadLocale	kernel32
004173B8	GetThreadLocale	kernel32
004173B4	GetVersionExA	kernel32
004173B0	InitializeCriticalSection	kernel32
004172E4	InterlockedDecrement	kernel32
004172E8	InterlockedIncrement	kernel32
004173...	LeaveCriticalSection	kernel32
00417300	LoadLibraryExA	kernel32
004173A8	LoadResource	kernel32
0041735C	LocalAlloc	kernel32
00417358	LocalFree	kernel32
004173A4	LockResource	kernel32
004172F4	MultiByteToWideChar	kernel32
0041733C	RaiseException	kernel32
004173A0	ReadFile	kernel32
0041739C	ResetEvent	kernel32
00417338	RtlUnwind	kernel32

Question 6

6. Navigate to the ServiceMain function. (a) Show the graph view of the function (b) The main subroutine (of the ServiceMain function) jumps to a location where the code calls the kernel32 API *Sleep()* right after the JZ assembly instruction. What is the value of the parameter used by this *Sleep()* call? **[5 marks]**

a) GRAPH



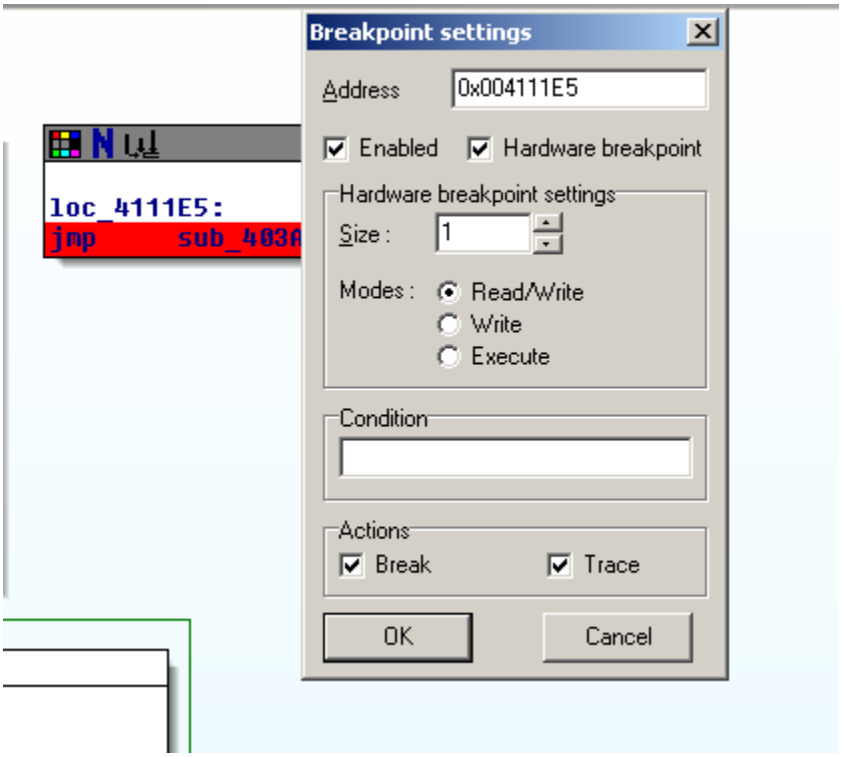
B)

By using the jmp instruction to skip over the conditional branch instruction, the code ensures that the Sleep function call is always executed, regardless of the value of the zero flag. This is useful in situations where the code needs to pause for a certain amount of time, regardless of whether certain conditions are met or not.

In summary, the jmp instruction in the given image is used to transfer control to the Sleep function call, skipping over any conditional branch instructions that come before it. This ensures that the Sleep function call is always executed, regardless of the value of any flags or conditions.

```
.text:0041083F locret_41083F:                          ; CODE XREF: sub_41079C+A9↓j
.text:0041083F                 retn
.text:00410840 ; ---------------------------------------------------------------------------
.text:00410840
.text:00410840 loc_410840:                             ; DATA XREF: sub_41079C+6↑o
.text:00410840                 jmp     sub_403A14
.text:00410845 ; ---------------------------------------------------------------------------
.text:00410845                 jmp     short locret_41083F
.text:00410847 ; ---------------------------------------------------------------------------
.text:00410847
.text:00410847 loc_410847:                             ; CODE XREF: sub_41079C:locret_41083F↑j
.text:00410847                                         ; DATA XREF: sub_41079C+9E↑o
.text:00410847                 pop     ebp
.text:00410848                 retn
.text:00410848 sub_41079C      endp ; sp = -4
.text:00410848
.text:00410848 ; ---------------------------------------------------------------------------
.text:00410849                 align 4
.text:0041084C
.text:0041084C ; ||||||||||||||| S U B R O U T I N E |||||||||||||||||||||||||||||||||||||||
.text:0041084C
.text:0041084C
.text:0041084C sub_41084C      proc near               ; CODE XREF: sub_4108F0+64↓p
.text:0041084C                                         ; Install+26↓p ...
.text:0041084C                 push    ebx
```

To determine the value of the parameter used by the Sleep() call, we need to look at the instruction right before the JZ instruction. Specifically, we need to look at the instruction that sets the value of the EAX register, since the JZ instruction checks the zero flag, which is set or cleared based on the result of the previous arithmetic or logical instruction.



Service main function could not be found in the sample.