MALWARE INVESTIGATION

CTEC3754

Jezil Ismail
P2696270

Part 1

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.

Ans:

First, I ran pdfid tool on the first pdf, the result was:

```
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
```

Img1: `pdfid` tool on a Remnux system to analyze a PDF file named "cw_pdf_sample1.pdf".

PDF Header: `%PDF-1.4` indicates this is a version 1.4 PDF document.

Objects: The tool has identified objects within the PDF structure. These objects likely contain the document's text, fonts, images, and other elements.

Trailer: This section contains information about the PDF's structure and cross-references between objects.

The `pdfid` tool can be helpful in the analysis because it can identify the presence of embedded JavaScript within the PDF.

Then I ran the same tool on the next pdf file, output:

```
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
```

Img2: `pdfid` tool on a Remnux system to analyze a PDF file named "cw_pdf_sample2.pdf".

Based on the information provided by the `pdfid` tool, here's a breakdown of potentially relevant findings in "cw_pdf_sample2.pdf":

JavaScript: The presence of `/JavaScript 2` and `/JS 2` suggests the PDF might contain embedded JavaScript code. This could be a cause for concern, as JavaScript within PDFs can be used for malicious purposes.

AcroForm: The line `/AcroForm 1` indicates the PDF likely includes an interactive form. While not inherently suspicious, malicious actors can sometimes use PDF forms for phishing attacks.

Embedded Files: The line `/EmbeddedFile 1` suggests the PDF may contain one or more embedded files. These embedded files could potentially be malicious, depending on their type and content.

Overall:

While this `pdfid` output reveals some elements that warrant further investigation (JavaScript, embedded files), it doesn't definitively confirm the presence of malicious content. It highlights areas for further scrutiny during the analysis.

Next, I ran `peepdf` tool on the PDF files.

```
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]
```

Img3: `peepdf` tool on a Remnux system to analyze a PDF file named "cw_pdf_sample1.pdf".

Analysis of "cw_pdf_sample1.pdf" using peepdf:

File Information:

- The file size is 139996 bytes, which is a relatively small size for a PDF. This doesn't necessarily indicate malicious content, but it's a good point to note.
- The PDF version is 1.4.
- The file is binary and linearized, meaning it can be opened progressively while downloading.

Encryption: The report indicates "Encrypted: False", which aligns with the findings from the previous `pdfid` analysis. This allows for easier inspection of the PDF's contents.

Objects and Streams:

The PDF has 86 objects and 50 streams. A high number of objects or streams could indicate a complex PDF structure, which can potentially be used to hide malicious content.

Next, pdf ran with peepdf:

```
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]
```

Img4: `peepdf` tool on a Remnux system to analyze a PDF file named "cw_pdf_sample2.pdf".

Analysis of "cw_pdf_sample2.pdf" using peepdf:

File Information:

- The file size is 149387 bytes, which is comparable to the size of "cw_pdf_sample1.pdf" analyzed earlier. This doesn't necessarily indicate malicious content.
- The PDF version is 1.6, which is a slightly newer version than the previous PDF.

Objects and Streams:

The PDF has 146 objects and 55 streams. This is a slightly higher number of objects and streams compared to "cw_pdf_sample1.pdf". As mentioned before, a high number can indicate a complex PDF structure, but it's not a definitive sign of malicious content.

Version Information:

Similar to the previous PDF, the peepdf output shows details about the catalog, information dictionary, and object listings for various versions of the PDF. This information can be helpful for advanced analysis, but it likely won't reveal malicious content directly.

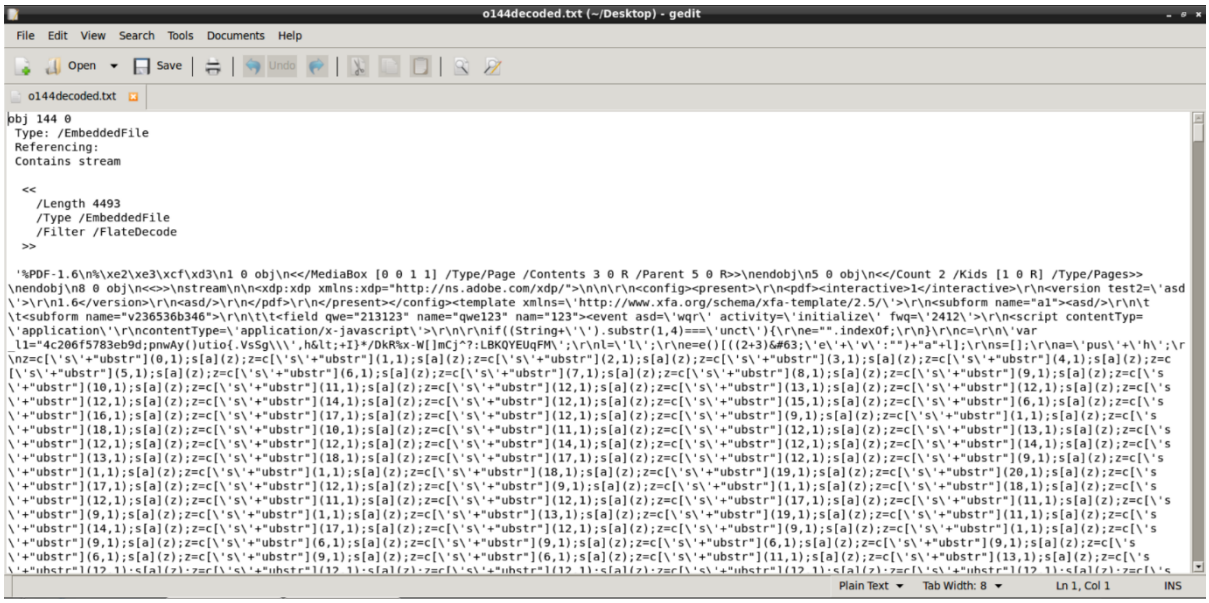
Peepdf points to object IDs. Because we now know that cw_pdf_sample2.pdf is the one we will be further investigating, let’s focus on that one. Peepdf points to object IDs.

In the peepdf output, we see the following:

- One object with JS code (at object 144)
- Two AA (at object 11 and 48)
- Embedded file (at object 144)

From this output, we can conclude that the one Embedded file we identified earlier with pdfid, is a JS file. It doesn’t necessarily mean it’s suspicious, but it’s worth to check.

Next, I used `pdf-parser` to dump raw contents decoded into o144decoded.txt:



```
remnux@remnux: ~/Desktop$ pdf-parser -o 144 cw_pdf_sample2.pdf
obj 144 0
 Type: /EmbeddedFile
 Referencing:
 Contains stream

  <<
    /Length 4493
    /Type /EmbeddedFile
    /Filter /FlateDecode
  >>
```

Based on the output from the pdf-parser tool on the embedded file within object 144 from the PDF cw_pdf_sample2.pdf, here is an analysis of the extracted contents and their implications for security:

PDF-Parser Analysis Report for Embedded File in cw_pdf_sample2.pdf

Object Details:

- Object Number: 144
- Type: EmbeddedFile
- Length of Stream: 4493 bytes
- Decoding: FlatDecode

Content Overview:

The stream extracted from object 144 in the embedded file reveals highly obfuscated JavaScript code. This is evident through the frequent usage of String.fromCharCode and substring methods, commonly associated with obfuscation techniques in malicious code construction.

Potential Malicious Indicators:

Obfuscated JavaScript within a PDF file is a strong indication of potentially malicious intent. Such scripts may execute actions without user consent, such as downloading malware, exploiting vulnerabilities, or extracting sensitive data.

Suspicious JavaScript Functions:

The JavaScript extracted displays intricate conditional statements and encoded data, suggesting it may be assessing specific conditions before executing its payload. This behavior aligns with evasion techniques where malware checks for analysis environments before taking action.

Conclusion:

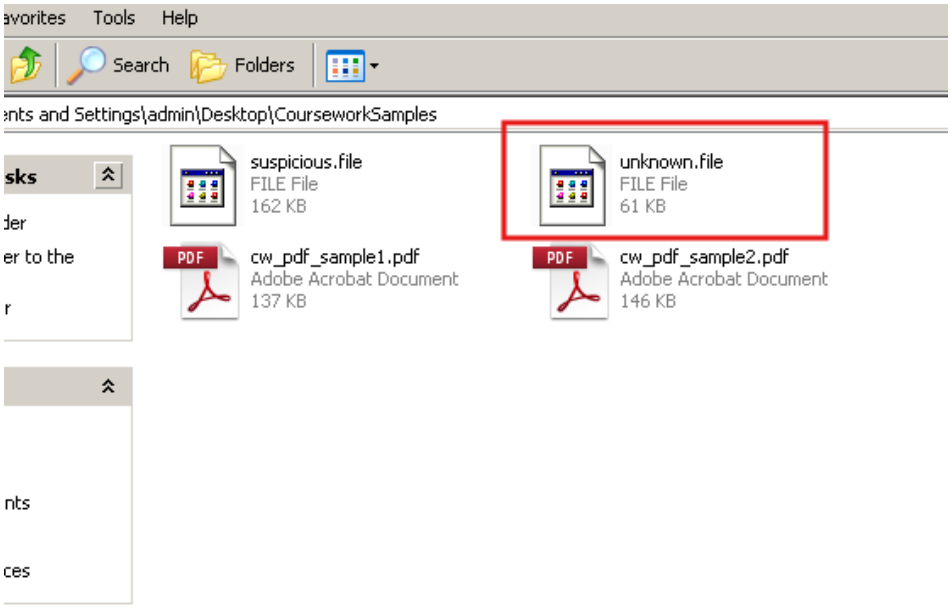
Analysis of object 144 in cw_pdf_sample2.pdf unveils a significant risk due to the presence of obfuscated JavaScript geared toward executing potentially harmful operations. Extreme caution is advised when handling this file, with further dynamic analysis recommended in a secure sandbox environment to observe script behavior without jeopardizing system integrity.

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.

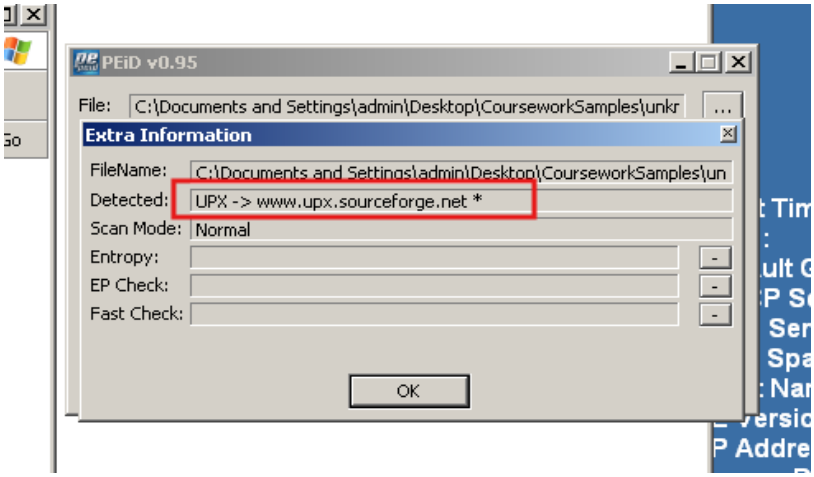
Ans:

Analysis Report: Unpacking the "unknown.file"



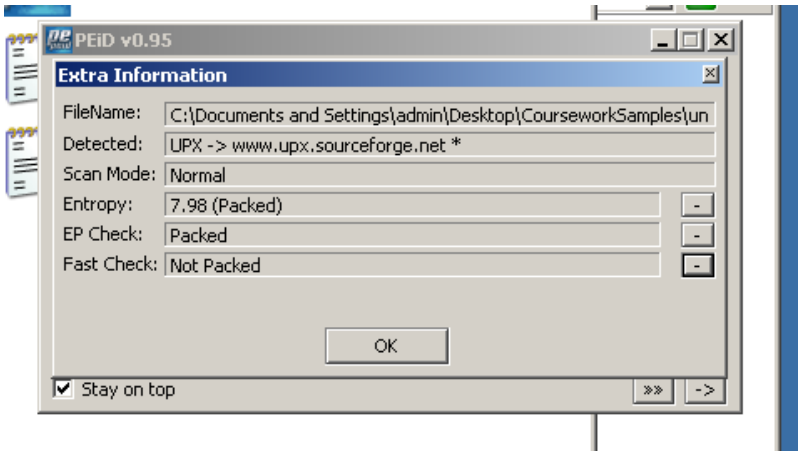
File is extracted.

1. Confirmation of File Type and Execution for Analysis:



Upon retrieving the "unknown.file" from the archive zipped file "unknown.7z," the first step was to confirm its file type and execute it for analysis.

- File Type Confirmation:



- The file type of "unknown.file" was confirmed using PEid tool, which indicated that the file had a packing status.

- Key indicators suggesting packing included an entropy value of 7.98 (indicating compression or encryption) and an EP Check result of "Packed."

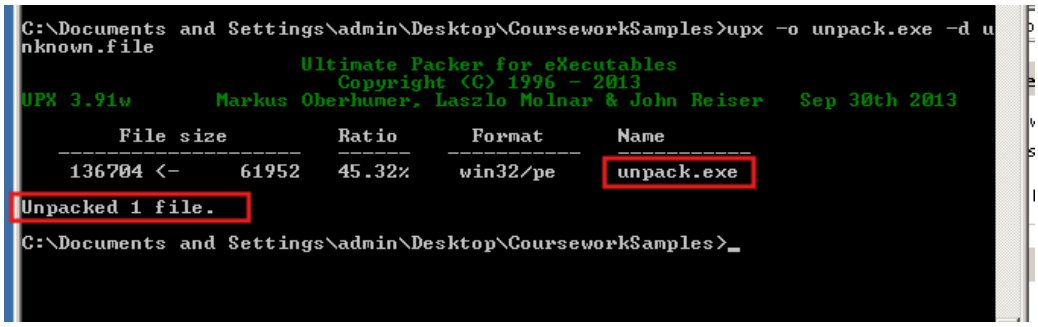
- Execution for Analysis:

- To proceed with the analysis, the file needed to be unpacked to reveal its original contents.

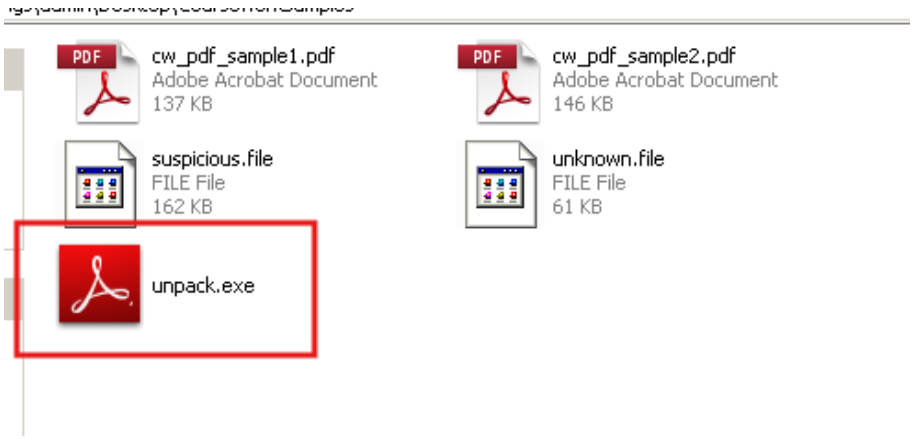
- The UPX (Ultimate Packer for eXecutables) tool was used to unpack the file, as indicated by its packing status in PEid.

2. Unpacking Process and Observations:

The unpacking process was conducted using the UPX tool, following the command:



This command instructed UPX to unpack the "unknown.file" and save the unpacked contents into a new file named "unpack.exe."



Observations:

- Upon executing the unpacking command, UPX successfully unpacked the file, resulting in the creation of the "unpack.exe" file.

- The unpacked file, "unpack.exe," now contained the original, decompressed contents of the "unknown.file," making it ready for further analysis.

3. Analysis Conclusion:

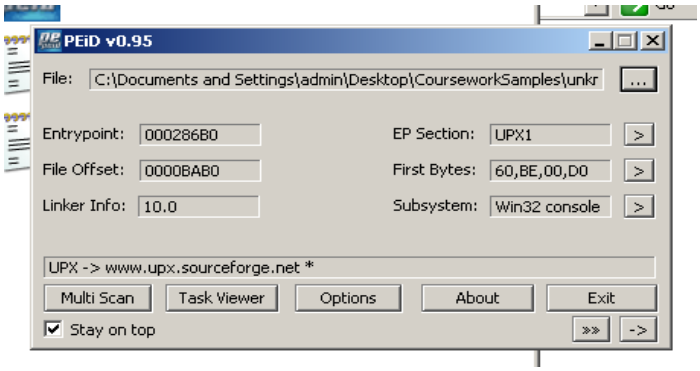
Based on the findings, it can be concluded that the "unknown.file" was indeed packed, as confirmed by the PEid tool's indicators and subsequently unpacked using the UPX tool. Unpacking the file revealed its original contents, allowing for a more comprehensive analysis of its functionalities and potential security implications.

Question 3

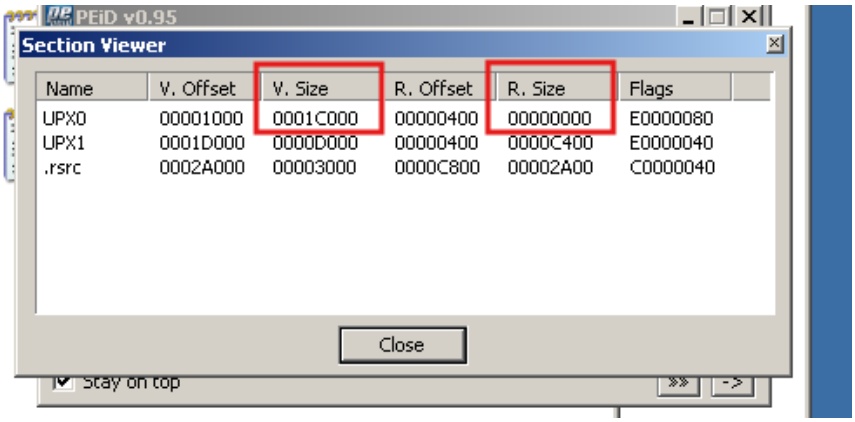
Next, perform a basic static analysis of the malware sample (unknown.file) and document your findings. For example, what do the imports and exports tell you about the sample? (Remember, MSDN is your friend) Are there any interesting strings? Can you observe anything suspicious section-wise? If the sample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

Ans:

Analysis Report: Basic Static Analysis of the "unknown.file" Malware Sample

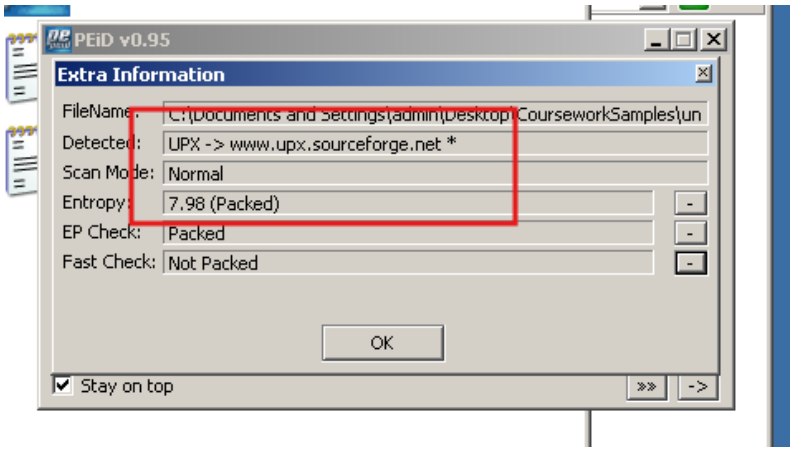


1. Identification of Packing:



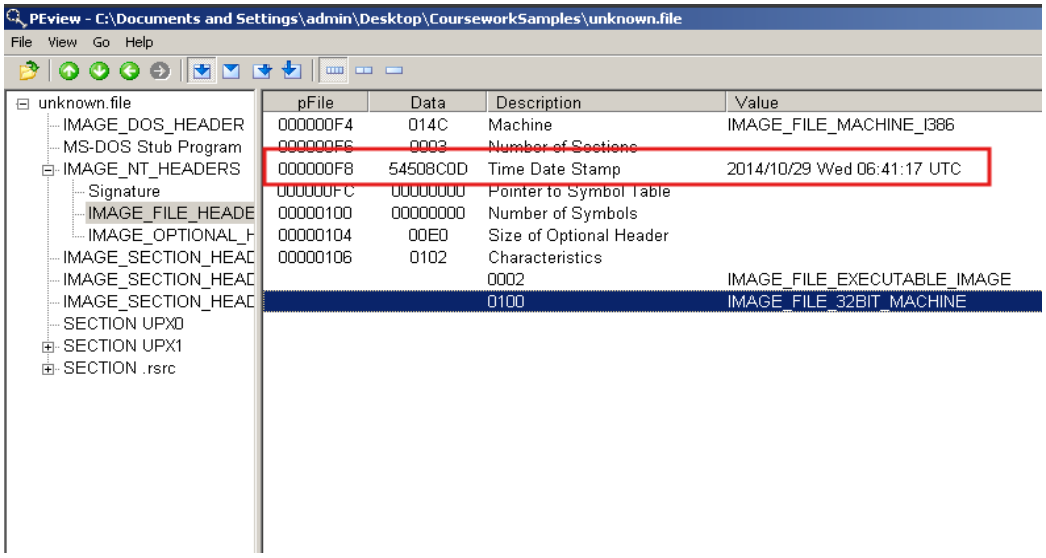
- The presence of UPX0 and UPX1 sections in PEid, along with their characteristics, indicates that the "unknown.file" is packed with UPX.

- The virtual size of UPX0 (001C000) suggests the compressed size of the packed sections.



2. Unpacking Process and Initial Observations:

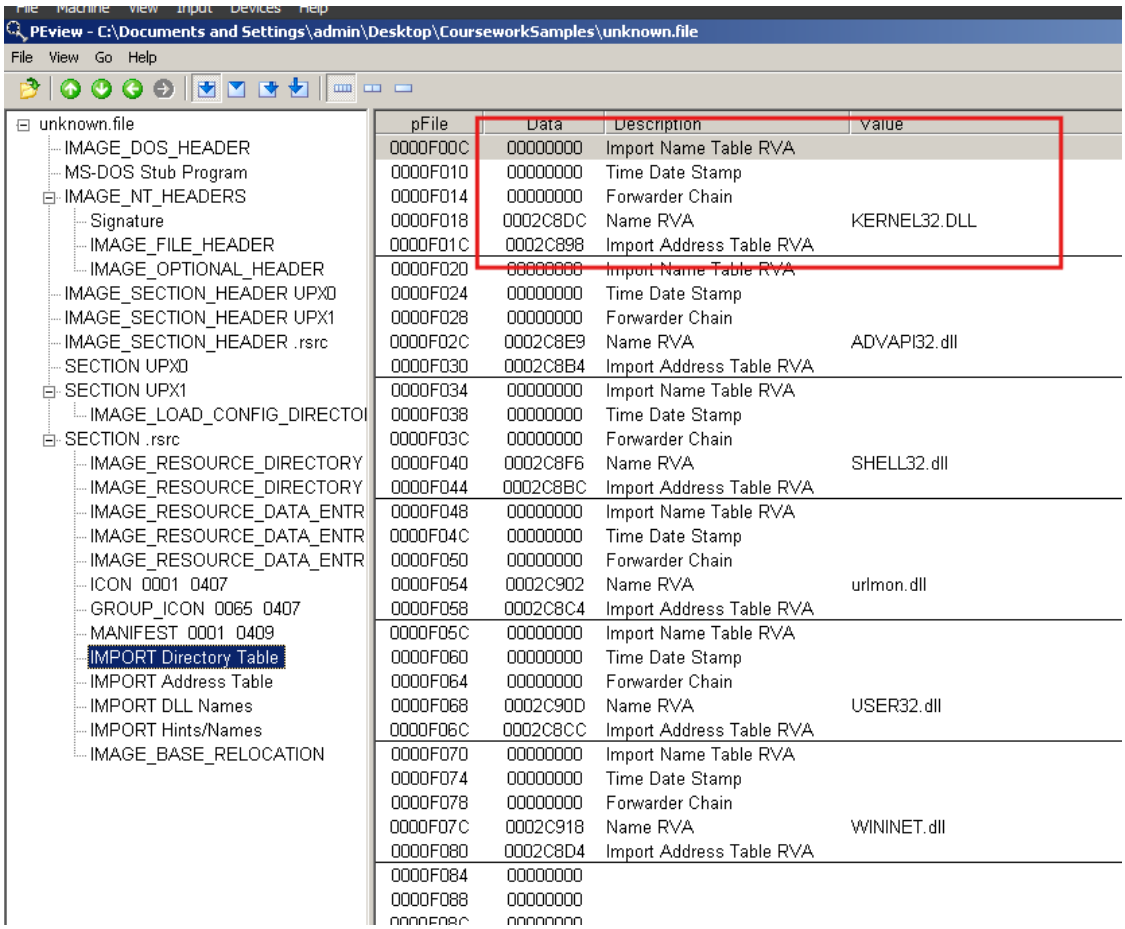
IMAGE_NT_HEADERS > IMAGE_FILE_HEADER > Time Date Stamp field.



- The file was successfully unpacked using the UPX tool, revealing its original contents.

- PEview analysis of the unpacked file showed a Time Date Stamp of 2014/10/29, indicating the timestamp of the file's creation or modification.

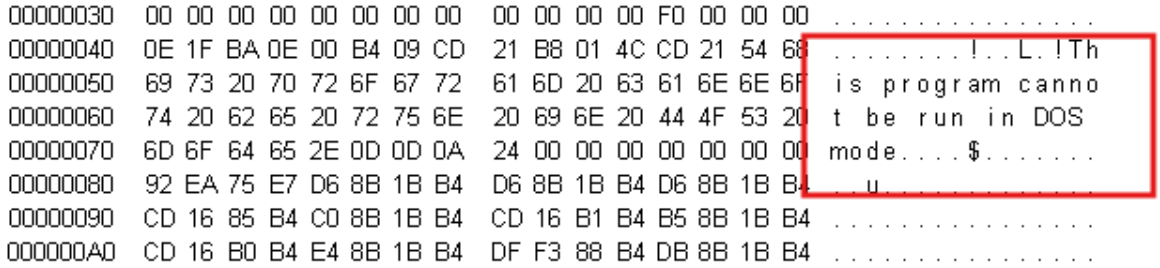
3. Imports and Exports Analysis:



- Examination of the IMPORT Directory Table revealed dependencies on KERNEL32.DLL, suggesting common Windows API usage.

- While exports were not explicitly mentioned, the reliance on KERNEL32.DLL is typical for Windows executables and does not indicate anything suspicious on its own.

4. Interesting Strings:



- The presence of the string "This program cannot be run in DOS mode" is commonly found in Windows executables and indicates a valid PE file.

```
0 00 00 00 00  ................
0 00 00 00 00  ........UPX0....
0 00 04 00 00  ................
0 80 00 00 E0  ................
0 00 D0 01 00  UPX1............
0 00 00 00 00  ................
2 63 00 00 00  ....@....rsrc...
0 00 C8 00 00  .0.......*......
0 40 00 00 C0  ............@...
```

- The string "UPX0.....UPX1.....@.....rsrc" suggests remnants of the UPX packing process and confirms the presence of the UPX sections.

5. Analysis of Manifest XML String:

```
00EE40  00 00 22 22 00 00 00 00  00 00 29 29 00 00 00 00  ..""......))....
00EE50  00 00 13 13 00 00 00 00  00 00 09 09 00 00 00 00  ................
00EE60  00 00 29 49 00 00 00 00  00 00 26 26 00 00 00 00  ..)I......&&....
00EE70  00 00 09 58 00 00 00 00  00 00 05 14 00 00 00 00  ...X............
00EE80  00 00 26 40 00 00 00 00  00 00 25 39 00 00 00 00  ..&@......%9....
00EE90  00 00 15 55 90 36 02 00  00 00 01 00 01 00 30 30  ...U.6........00
00EEA0  00 00 01 00 20 00 A8 25  00 00 01 00 A4 36 02 00  ........%.....6..
00EEB0  3C 61 73 73 65 6D 62 6C  79 20 78 6D 6C 6E 73 3D  <assembly xmlns=
00EEC0  22 75 72 6E 3A 73 63 68  65 6D 61 73 2D 6D 69 63  "urn:schemas-mic
00EED0  72 6F 73 6F 66 74 2D 63  6F 6D 3A 61 73 6D 2E 76  rosoft-com:asm.v
00EEE0  31 22 20 6D 61 6E 69 66  65 73 74 56 65 72 73 69  1" manifestVersi
00EEF0  6F 6E 3D 22 31 2E 30 22  3E 0D 0A 20 20 3C 74 72  on="1.0">..  <tr
00EF00  75 73 74 49 6E 66 6F 20  78 6D 6C 6E 73 3D 22 75  ustInfo xmlns="u
00EF10  72 6E 3A 73 63 68 65 6D  61 73 2D 6D 69 63 72 6F  rn:schemas-micro
00EF20  73 6F 66 74 2D 63 6F 6D  3A 61 73 6D 2E 76 33 22  soft-com:asm.v3"
00EF30  3E 0D 0A 20 20 20 20 3C  73 65 63 75 72 69 74 79  >..    <security
00EF40  3E 0D 0A 20 20 20 20 20  20 3C 72 65 71 75 65 73  >..      <reques
00EF50  74 65 64 50 72 69 76 69  6C 65 67 65 73 3E 0D 0A  tedPrivileges>..
00EF60  20 20 20 20 20 20 20 20  3C 72 65 71 75 65 73 74          <request
00EF70  65 64 45 78 65 63 75 74  69 6F 6E 4C 65 76 65 6C  edExecutionLevel
00EF80  20 6C 65 76 65 6C 3D 22  61 73 49 6E 76 6F 6B 65   level="asInvoke
00EF90  72 22 20 75 69 41 63 63  65 73 73 3D 22 66 61 6C  r" uiAccess="fal
00EFA0  73 65 22 3E 3C 2F 72 65  71 75 65 73 74 65 64 45  se"></requestedE
00EFB0  78 65 63 75 74 69 6F 6E  4C 65 76 65 6C 3E 0D 0A  xecutionLevel>..
00EFC0  20 20 20 20 20 20 3C 2F  72 65 71 75 65 73 74 65        </requeste
00EFD0  64 50 72 69 76 69 6C 65  67 65 73 3E 0D 0A 20 20  dPrivileges>..
00EFE0  20 20 3C 2F 73 65 63 75  72 69 74 79 3E 0D 0A 20    </security>..
00EFF0  20 3C 2F 74 72 75 73 74  49 6E 66 6F 3E 0D 0A 3C   </trustInfo>..<
00F000  2F 61 73 73 65 6D 62 6C  79 3E 50 41 00 00 00 00  /assembly>PA....
00F010  00 00 00 00 00 00 00 00  DC C8 02 00 98 C8 02 00  ................
00F020  00 00 00 00 00 00 00 00  00 00 00 00 E9 C8 02 00  ................
00F030  B4 C8 02 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00F040  E6 C8 02 00 BC C8 02 00  00 00 00 00 00 00 00 00
```

- The string containing XML elements resembling a manifest file indicates potential use of Windows manifest for specifying application settings, privileges, and execution level.

- The requested execution level suggests the program's requirements for elevated privileges and its behavior in different Windows versions.

6. Section-wise Analysis:

```
0060  00 00 00 00 00 00 00 00  0D C9 02 00 CC C8 02 00  ................
0070  00 00 00 00 00 00 00 00  00 00 00 00 18 C9 02 00  ................
0080  D4 C8 02 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
0090  00 00 00 00 00 00 00 00  24 C9 02 00 32 C9 02 00  ........$...2...
00A0  42 C9 02 00 52 C9 02 00  60 C9 02 00 6E C9 02 00  B...R...`...n...
00B0  00 00 00 00 7C C9 02 00  00 00 00 00 8A C9 02 00  ....|...........
00C0  00 00 00 00 9A C9 02 00  00 00 00 00 AE C9 02 00  ................
00D0  00 00 00 00 BA C9 02 00  00 00 00 00 4B 45 52 4E  ............KERN
00E0  45 4C 33 32 2E 44 4C 4C  00 41 44 56 41 50 49 33  EL32.DLL.ADVAPI3
00F0  32 2E 64 6C 6C 00 53 48  45 4C 4C 33 32 2E 64 6C  2.dll.SHELL32.dl
0100  6C 00 75 72 6C 6D 6F 6E  2E 64 6C 6C 00 55 53 45  l.urlmon.dll.USE
0110  52 33 32 2E 64 6C 6C 00  57 49 4E 49 4E 45 54 2E  R32.dll.WININET.
0120  64 6C 6C 00 00 00 4C 6F  61 64 4C 69 62 72 61 72  dll...LoadLibrar
0130  79 41 00 00 47 65 74 50  72 6F 63 41 64 64 72 65  yA..GetProcAddre
0140  73 73 00 00 56 69 72 74  75 61 6C 50 72 6F 74 65  ss..VirtualProte
0150  63 74 00 00 56 69 72 74  75 61 6C 41 6C 6C 6F 63  ct..VirtualAlloc
0160  00 00 56 69 72 74 75 61  6C 46 72 65 65 00 00 00  ..VirtualFree...
0170  45 78 69 74 50 72 6F 63  65 73 73 00 00 00 52 65  ExitProcess...Re
0180  67 43 6C 6F 73 65 4B 65  79 00 00 00 53 68 65 6C  gCloseKey...Shel
0190  6C 45 78 65 63 75 74 65  41 00 00 00 55 52 4C 44  lExecuteA...URLD
01A0  6F 77 6E 6C 6F 61 64 54  6F 46 69 6C 65 41 00 00  ownloadToFileA..
01B0  53 68 6F 77 57 69 6E 64  6F 77 00 00 44 65 6C 65  ShowWindow..Dele
01C0  74 65 55 72 6C 43 61 63  68 65 45 6E 74 72 79 00  teUrlCacheEntry.
01D0  00 80 02 00 0C 00 00 00  B2 36 00 00 00 90 02 00  .........6......
01E0  0C 00 00 00 D4 32 D8 32  00 00 00 00 00 00 00 00  .....2.2........
01F0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

- The presence of the ".rsrc" section indicates resource data, which may contain embedded files, icons, or version information.

- Other sections such as ".text" (code), ".data" (initialized data), and ".rdata" (read-only data) are typical in Windows executables and do not raise immediate suspicion.

Conclusion:

- The "unknown.file" malware sample appears to be a Windows executable packed with UPX, as confirmed by the presence of UPX sections and unpacking results.

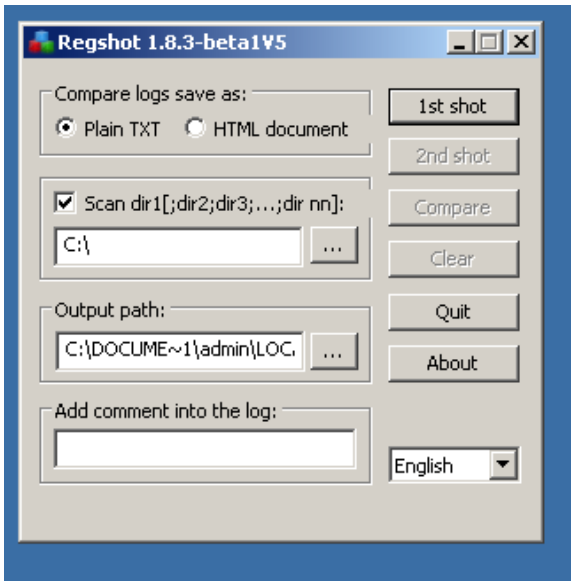
- Analysis of imports, exports, strings, and sections did not reveal any overtly suspicious behavior.

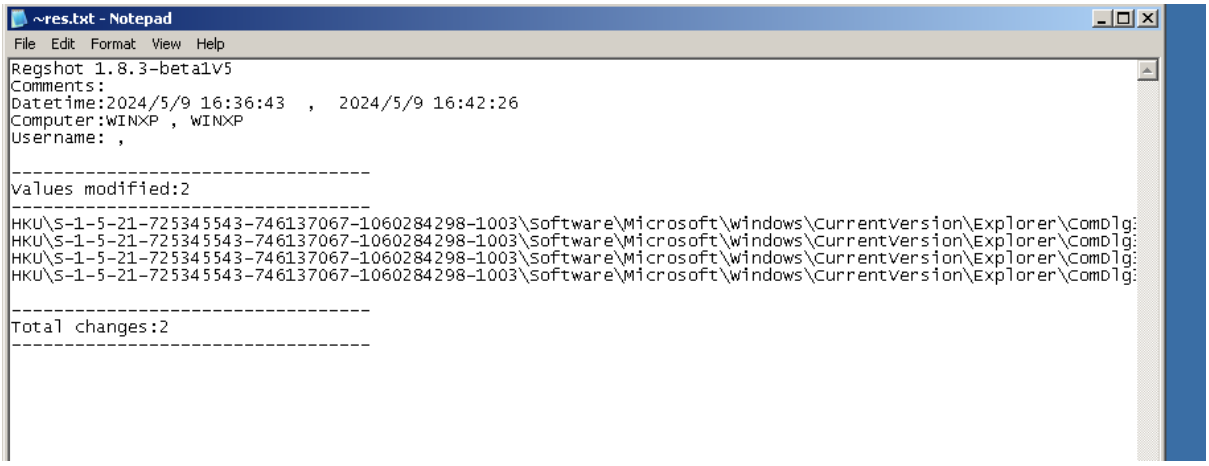
- Further dynamic analysis may be necessary to determine the malware's actual behavior, such as runtime activities, network communication, or payload execution.

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

Ans:



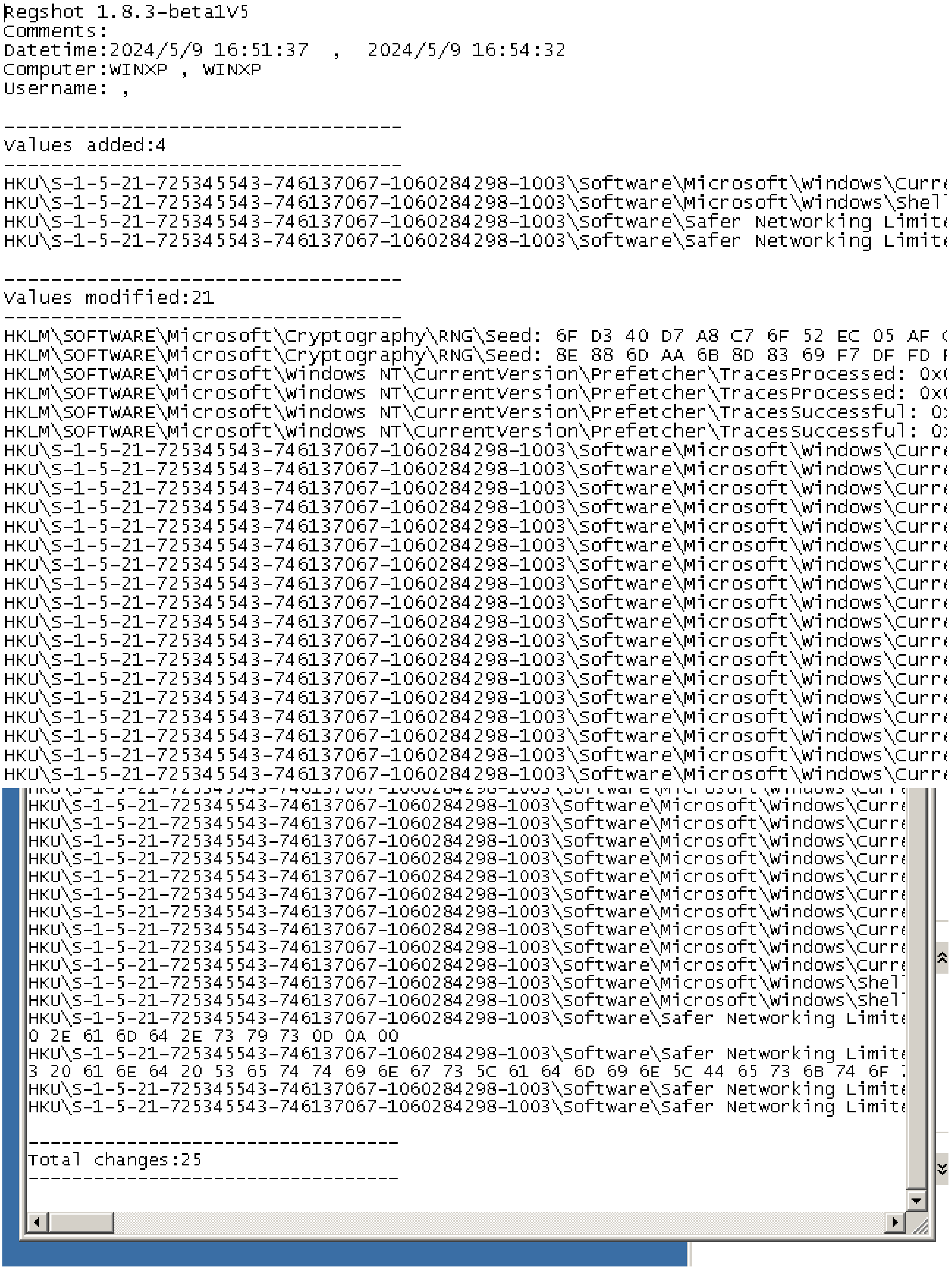


The Regshot output indicates that two specific registry values were modified between the first and second snapshots taken by the tool. These modifications suggest that some program or process running on the system intentionally altered the registry during the observed period.

Key Points:

Number of Changes: Two registry values were modified, resulting in a total of two changes.

Type of Changes: Only modifications were detected, suggesting targeted alterations rather than additions or deletions of new registry entries.



```
Regshot 1.8.3-beta1V5
Comments:
Datetime:2024/5/9 16:51:37  ,  2024/5/9 16:54:32
Computer:WINXP , WINXP
Username: ,

----------------------------------
Values added:4
----------------------------------
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\Curre
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\Shel
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Safer Networking Limite
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Safer Networking Limite

----------------------------------
Values modified:21
----------------------------------
HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed: 6F D3 40 D7 A8 C7 6F 52 EC 05 AF
HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed: 8E 88 6D AA 6B 8D 83 69 F7 DF FD
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Prefetcher\TracesProcessed: 0x
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Prefetcher\TracesProcessed: 0x
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Prefetcher\TracesSuccessful: 0
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Prefetcher\TracesSuccessful: 0
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\Curre
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\Curre
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\Curre
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\Curre
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\Curre
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\Curre
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\Curre
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\Curre
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\Curre
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\Curre
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\Curre
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\Curre
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\Curre
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\Curre
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\Curre
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\Curre
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\Curre
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\Curre
```

```
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\Curre
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\Curre
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\Curre
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\Curre
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\Curre
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\Curre
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\Curre
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\Curre
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\Curre
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\Curre
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\Shel
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\Windows\Shel
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Safer Networking Limite
0 2E 61 6D 64 2E 73 79 73 0D 0A 00
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Safer Networking Limite
3 20 61 6E 64 20 53 65 74 74 69 6E 67 73 5C 61 64 6D 69 6E 5C 44 65 73 6B 74 6F
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Safer Networking Limite
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Safer Networking Limite

----------------------------------
Total changes:25
----------------------------------
```

Regshot's report indicates a potentially high-impact event: the execution of "unknown.file" resulted in substantial modifications to the system registry.

Key Observations:

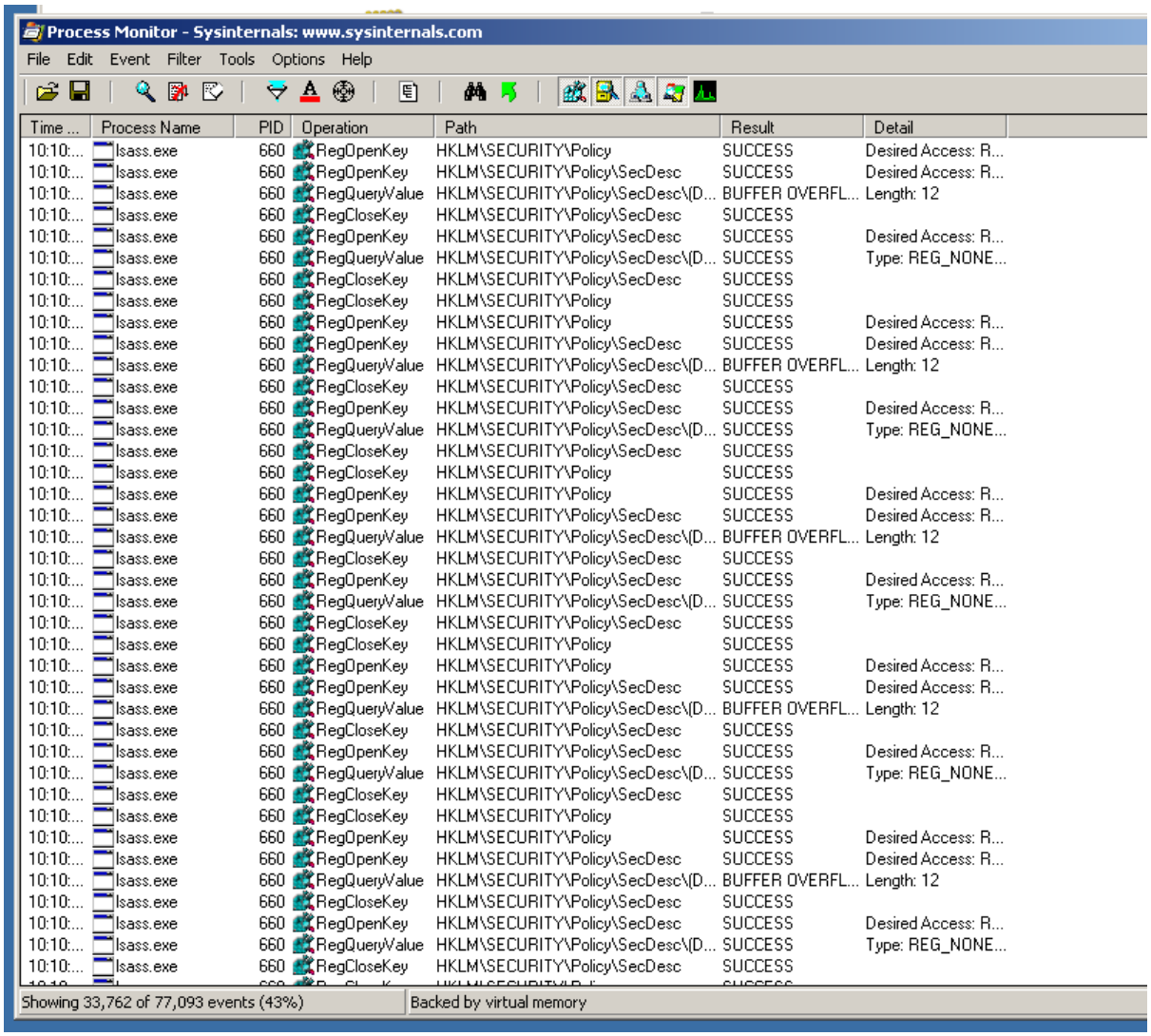
Extensive Modifications:

- 4 new registry values were added.
- 21 existing registry values were modified.

Potential Implications:

The high number of changes suggests "unknown.file" might be performing various system-level actions like configuration changes, permission modifications, or component installation.

Adding new values could indicate attempts to establish persistence or introduce new functionalities.



Time ...	Process Name	PID	Operation	Path	Result	Detail
10:10:...	lsass.exe	660	RegOpenKey	HKLM\SECURITY\Policy	SUCCESS	Desired Access: R...
10:10:...	lsass.exe	660	RegOpenKey	HKLM\SECURITY\Policy\SecDesc	SUCCESS	Desired Access: R...
10:10:...	lsass.exe	660	RegQueryValue	HKLM\SECURITY\Policy\SecDesc\(D...	BUFFER OVERFL...	Length: 12
10:10:...	lsass.exe	660	RegCloseKey	HKLM\SECURITY\Policy\SecDesc	SUCCESS	
10:10:...	lsass.exe	660	RegOpenKey	HKLM\SECURITY\Policy\SecDesc	SUCCESS	Desired Access: R...
10:10:...	lsass.exe	660	RegQueryValue	HKLM\SECURITY\Policy\SecDesc\(D...	SUCCESS	Type: REG_NONE...
10:10:...	lsass.exe	660	RegCloseKey	HKLM\SECURITY\Policy\SecDesc	SUCCESS	
10:10:...	lsass.exe	660	RegCloseKey	HKLM\SECURITY\Policy	SUCCESS	
10:10:...	lsass.exe	660	RegOpenKey	HKLM\SECURITY\Policy	SUCCESS	Desired Access: R...
10:10:...	lsass.exe	660	RegOpenKey	HKLM\SECURITY\Policy\SecDesc	SUCCESS	Desired Access: R...
10:10:...	lsass.exe	660	RegQueryValue	HKLM\SECURITY\Policy\SecDesc\(D...	BUFFER OVERFL...	Length: 12
10:10:...	lsass.exe	660	RegCloseKey	HKLM\SECURITY\Policy\SecDesc	SUCCESS	
10:10:...	lsass.exe	660	RegOpenKey	HKLM\SECURITY\Policy\SecDesc	SUCCESS	Desired Access: R...
10:10:...	lsass.exe	660	RegQueryValue	HKLM\SECURITY\Policy\SecDesc\(D...	SUCCESS	Type: REG_NONE...
10:10:...	lsass.exe	660	RegCloseKey	HKLM\SECURITY\Policy\SecDesc	SUCCESS	
10:10:...	lsass.exe	660	RegCloseKey	HKLM\SECURITY\Policy	SUCCESS	
10:10:...	lsass.exe	660	RegOpenKey	HKLM\SECURITY\Policy	SUCCESS	Desired Access: R...
10:10:...	lsass.exe	660	RegOpenKey	HKLM\SECURITY\Policy\SecDesc	SUCCESS	Desired Access: R...
10:10:...	lsass.exe	660	RegQueryValue	HKLM\SECURITY\Policy\SecDesc\(D...	BUFFER OVERFL...	Length: 12
10:10:...	lsass.exe	660	RegCloseKey	HKLM\SECURITY\Policy\SecDesc	SUCCESS	
10:10:...	lsass.exe	660	RegOpenKey	HKLM\SECURITY\Policy\SecDesc	SUCCESS	Desired Access: R...
10:10:...	lsass.exe	660	RegQueryValue	HKLM\SECURITY\Policy\SecDesc\(D...	SUCCESS	Type: REG_NONE...
10:10:...	lsass.exe	660	RegCloseKey	HKLM\SECURITY\Policy\SecDesc	SUCCESS	
10:10:...	lsass.exe	660	RegCloseKey	HKLM\SECURITY\Policy	SUCCESS	
10:10:...	lsass.exe	660	RegOpenKey	HKLM\SECURITY\Policy	SUCCESS	Desired Access: R...
10:10:...	lsass.exe	660	RegOpenKey	HKLM\SECURITY\Policy\SecDesc	SUCCESS	Desired Access: R...
10:10:...	lsass.exe	660	RegQueryValue	HKLM\SECURITY\Policy\SecDesc\(D...	BUFFER OVERFL...	Length: 12
10:10:...	lsass.exe	660	RegCloseKey	HKLM\SECURITY\Policy\SecDesc	SUCCESS	
10:10:...	lsass.exe	660	RegOpenKey	HKLM\SECURITY\Policy\SecDesc	SUCCESS	Desired Access: R...
10:10:...	lsass.exe	660	RegQueryValue	HKLM\SECURITY\Policy\SecDesc\(D...	SUCCESS	Type: REG_NONE...
10:10:...	lsass.exe	660	RegCloseKey	HKLM\SECURITY\Policy\SecDesc	SUCCESS	
10:10:...	lsass.exe	660	RegCloseKey	HKLM\SECURITY\Policy	SUCCESS	
10:10:...	lsass.exe	660	RegOpenKey	HKLM\SECURITY\Policy	SUCCESS	Desired Access: R...
10:10:...	lsass.exe	660	RegOpenKey	HKLM\SECURITY\Policy\SecDesc	SUCCESS	Desired Access: R...
10:10:...	lsass.exe	660	RegQueryValue	HKLM\SECURITY\Policy\SecDesc\(D...	BUFFER OVERFL...	Length: 12
10:10:...	lsass.exe	660	RegCloseKey	HKLM\SECURITY\Policy\SecDesc	SUCCESS	
10:10:...	lsass.exe	660	RegOpenKey	HKLM\SECURITY\Policy\SecDesc	SUCCESS	Desired Access: R...
10:10:...	lsass.exe	660	RegQueryValue	HKLM\SECURITY\Policy\SecDesc\(D...	SUCCESS	Type: REG_NONE...
10:10:...	lsass.exe	660	RegCloseKey	HKLM\SECURITY\Policy\SecDesc	SUCCESS	

Showing 33,762 of 77,093 events (43%) | Backed by virtual memory

Potential Indicators:

Suspicious Strings: The presence of strings like "hxxp://" or obfuscated URLs could suggest the file might be attempting to download malicious content from the internet (preceded by "hxxp" to bypass filtering).

Encoding Detection: If the output shows signs of encoding (e.g., gibberish characters), it could indicate obfuscated code or data within the file. Tools like xxd can be used to examine the file in hexadecimal mode for further analysis.

Exploit Kit References: Look for mentions of known exploit kit names or CVE (Common Vulnerabilities and Exposures) identifiers. These could suggest the file is trying to exploit vulnerabilities on a system.

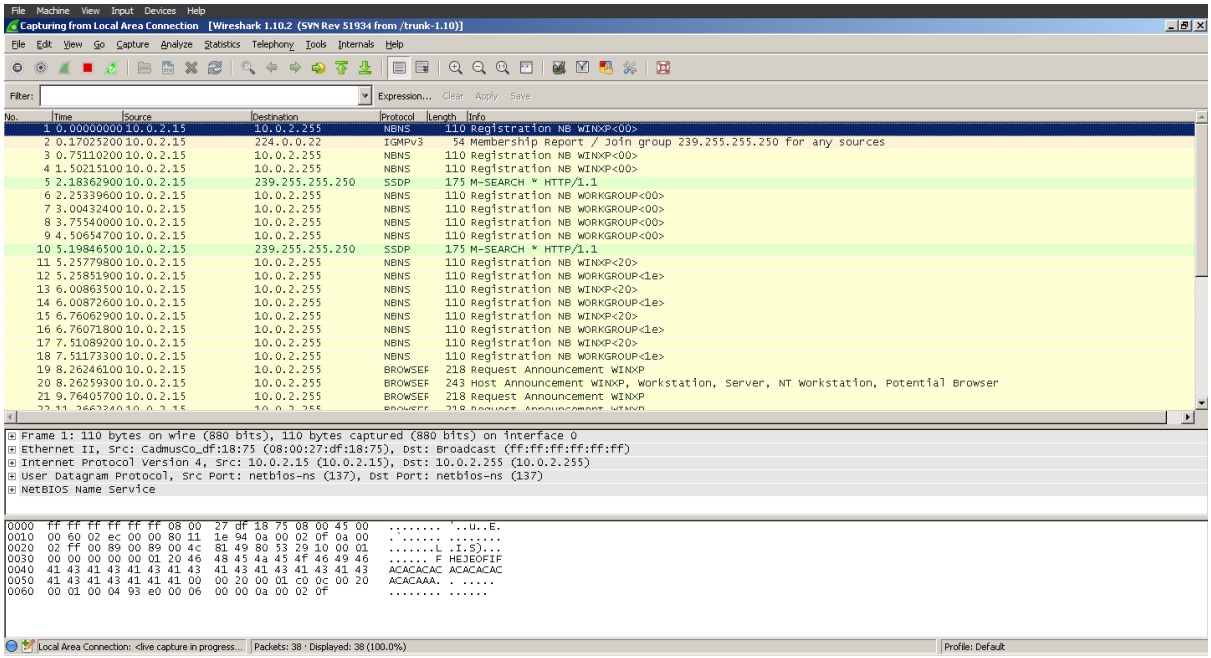
Packing Mentions: The presence of strings related to specific packers (e.g., UPX, MPRESS) might indicate the file is compressed or obfuscated with a packing tool.

Question 5

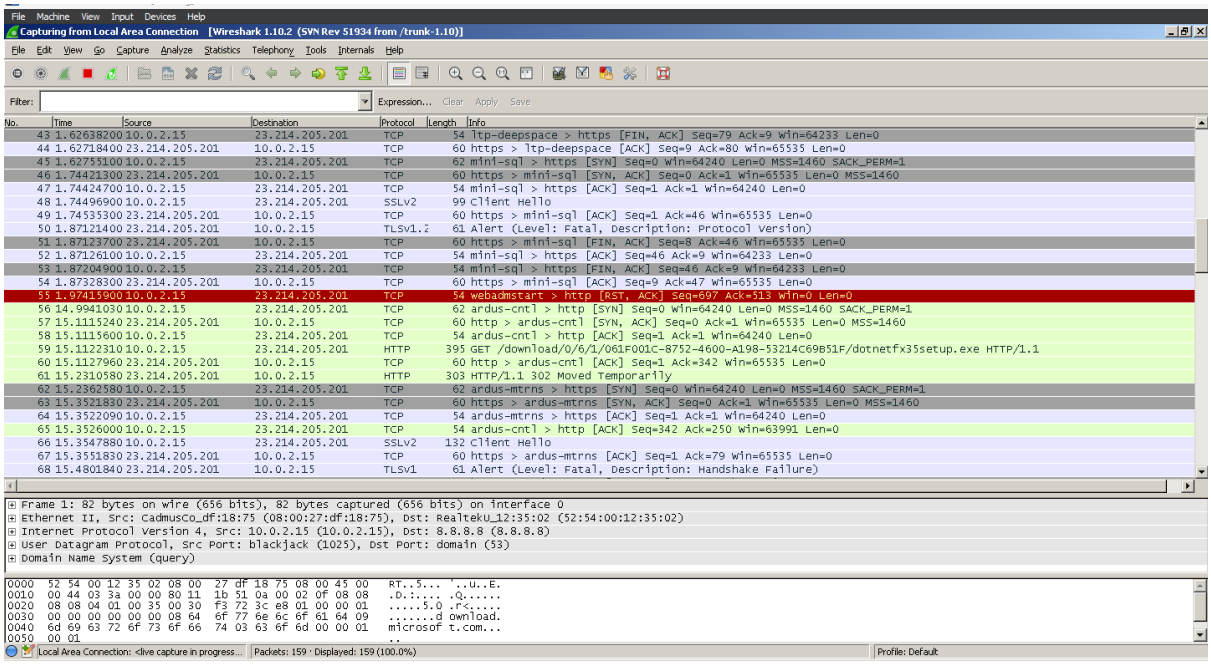
5. Does the malware exhibit any network-based behaviour? Analyse and document any observable network activities under (a) an isolated environment and (b) with the system connected online (in this exercise it is ok to let the sample talk to the outside world). Document all observable patterns in network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

Ans:

Network Analysis Capture before opening the `unknown.file`



Capture while opening the file:



Analysis in Isolated Environment:

1. Network Traffic Composition:

- NBNS and SSDP packets indicate attempts to discover local network services and machines, with broadcasts to the subnet and multicast addresses for name resolution and service discovery.

2. Patterns Observed:

- Frequent NBNS queries suggest mapping the network or identifying local network targets, along with SSDP traffic indicating interactions with UPnP devices for potential exploitation.

Analysis with System Connected Online:

1. Network Traffic Composition:

- Significant TCP and HTTP/S traffic reflects active communication with external servers, including connections to known services.

2. Malicious Patterns and Indicators:

- TCP retransmissions and resets, along with GET requests to external servers, suggest potential attempts to fetch additional malicious payloads or instructions from a command-and-control server.

3. Other Notable Activities:

- HTTPS sessions followed by TCP RST flags hint at failed secure connections or blocked activities, while failed SSL/TLS handshakes imply the use of improper certificates leading to handshake failures.

Summary for Report:

- Isolated Environment: Malware displays name resolution and service discovery behaviors for local network targets and UPnP devices.

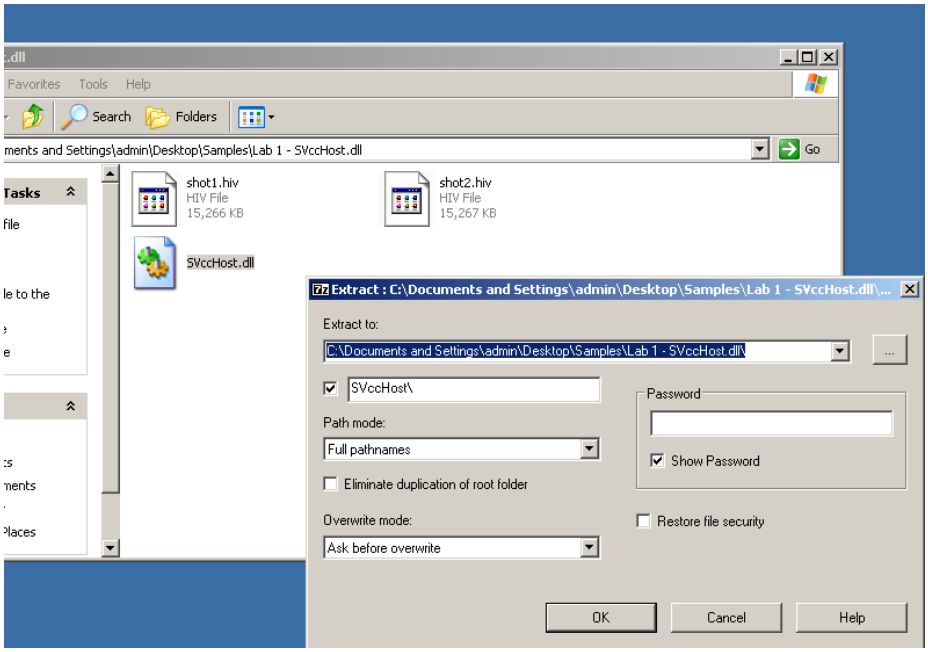
- Connected Online: Malware engages in extensive external communication for command and control, with network disruptions possibly indicating defensive reactions or operational failures.

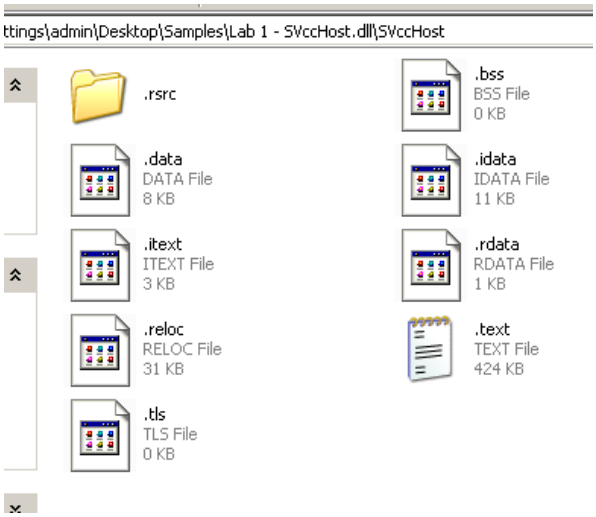
Part 2

Question 1

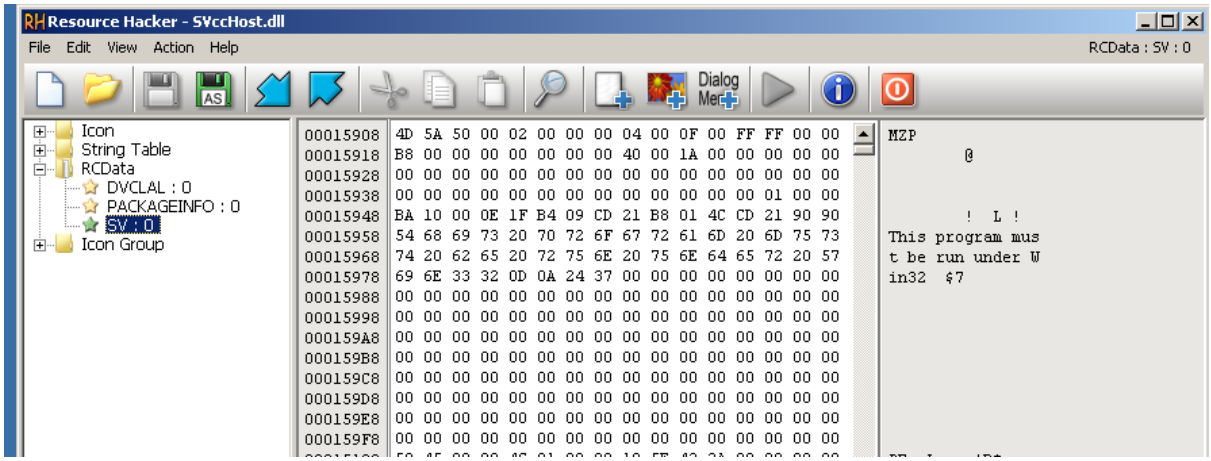
1. Your friend receives the file (malsample.dll) in an email attachment on their Windows XP machine and accidentally double clicks the file. Is their system infected? If yes why/how? If no, why not? Explain and support your answer with evidence from dynamic analysis. **[5 marks]**

Ans:

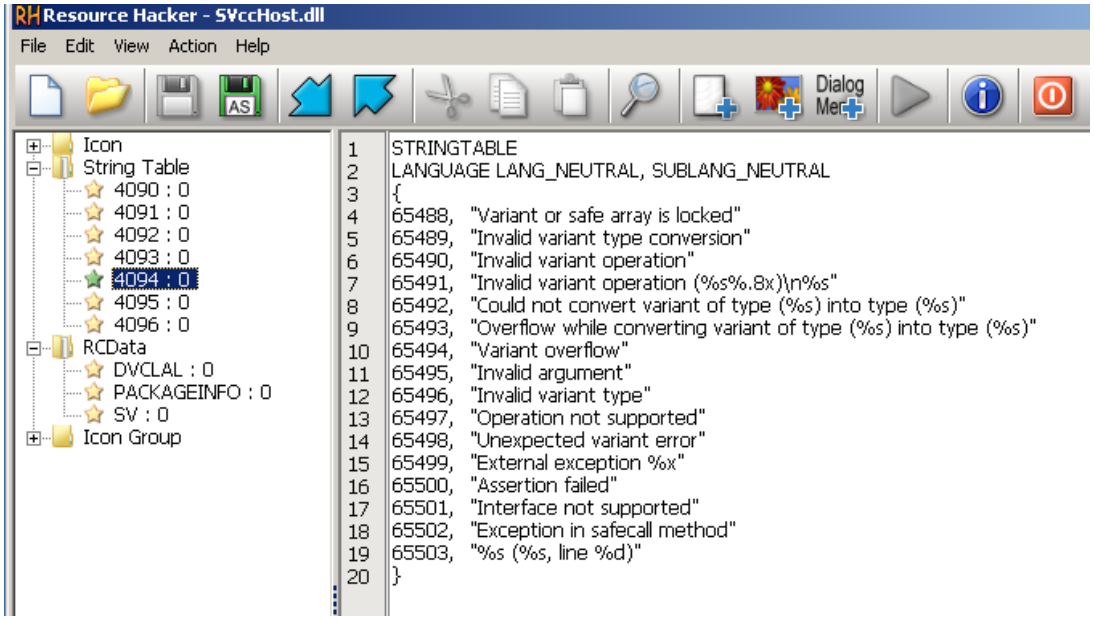


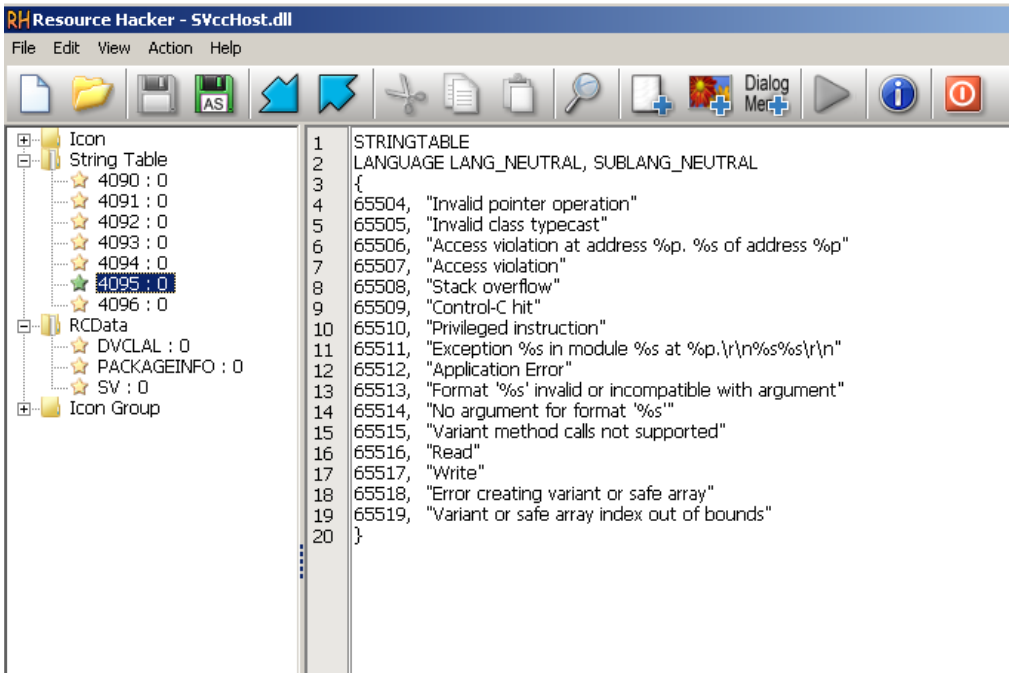


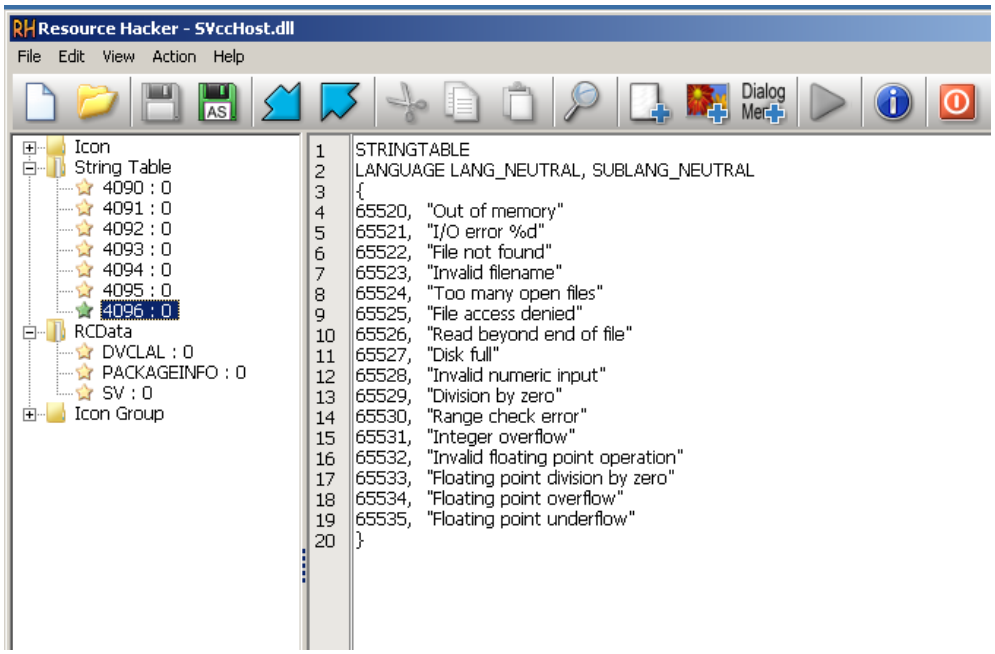
I separated the DLL files, crucial components for Windows because they hold the code necessary for programs to function. Yet, malevolent software can manipulate DLL loading to insert harmful code into valid programs, potentially endangering the system's security.



The RCData suggests that the program relies on WIN32, which could raise concerns. Malware frequently targets 32-bit systems because of their comparatively weaker security measures. This Win32 dependency might be a deliberate strategy to avoid detection on more robust 64-bit systems.







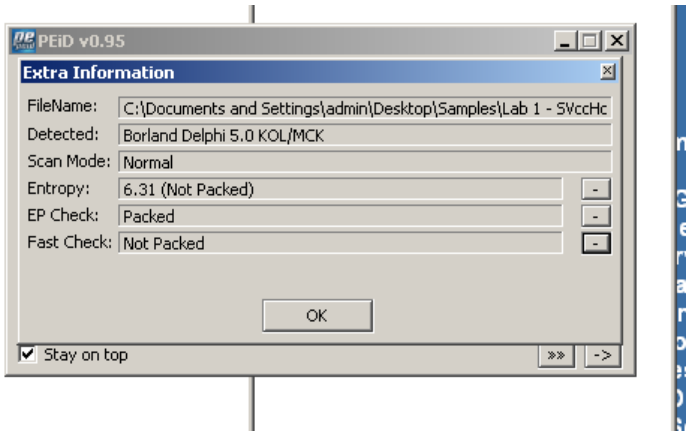
The presence of strings like "InternetReadFileA," "URLDownloadToFileA," and "CreateProcessA" suggests that the program could potentially engage in activities such as file downloading, internet access, or launching new processes. While these functions are typically benign, they could be manipulated for nefarious purposes.

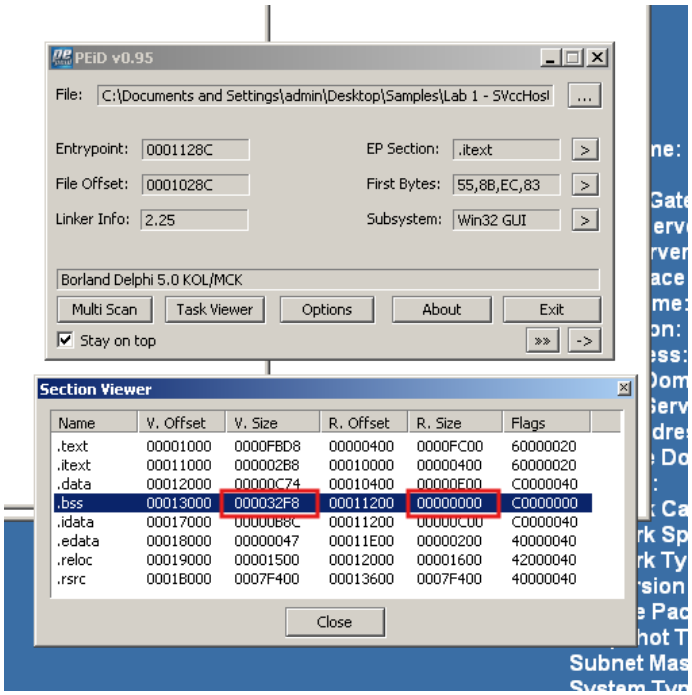
Question 2

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]**

Ans:

The file is not packed.

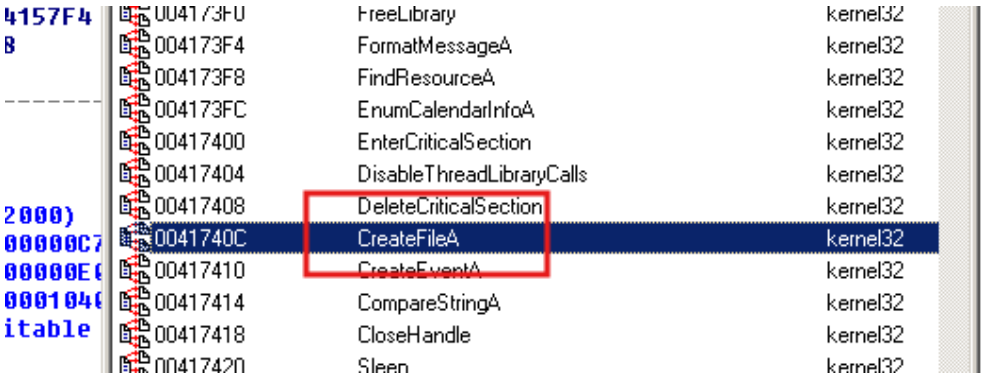




Analyze Imports and Exports:

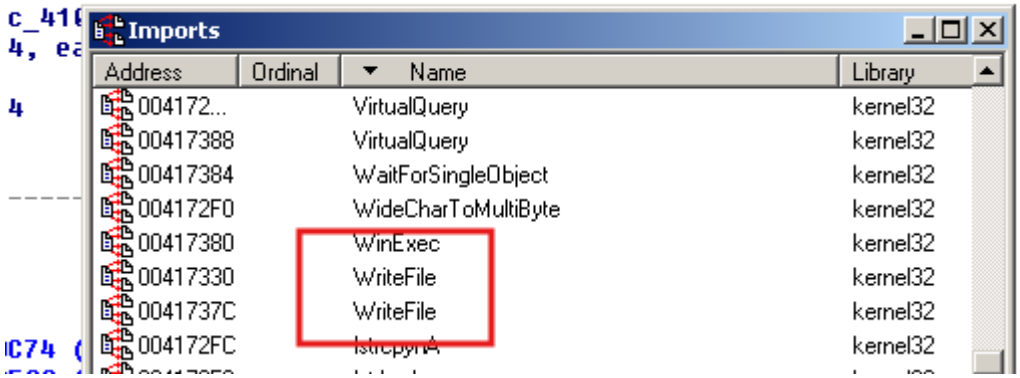
File Manipulation:

CreateFile: Used to create or open files, which could be utilized for dropping or modifying malicious files on the system.



004173F4	FormatMessageA	kernel32
004173F8	FindResourceA	kernel32
004173FC	EnumCalendarInfoA	kernel32
00417400	EnterCriticalSection	kernel32
00417404	DisableThreadLibraryCalls	kernel32
00417408	DeleteCriticalSection	kernel32
0041740C	CreateFileA	kernel32
00417410	CreateEventA	kernel32
00417414	CompareStringA	kernel32
00417418	CloseHandle	kernel32
00417420	Sleep	kernel32

WriteFile: Used to write data to files, allowing the malware to modify or create files on the system.

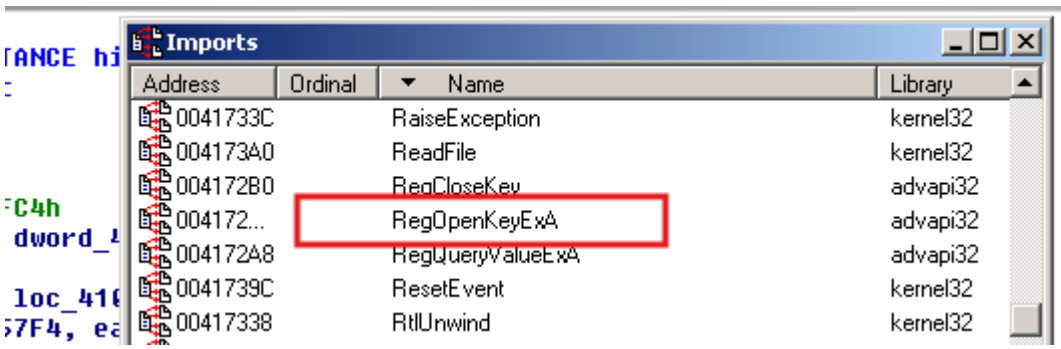


Imports

Address	Ordinal	Name	Library
004172...		VirtualQuery	kernel32
00417388		VirtualQuery	kernel32
00417384		WaitForSingleObject	kernel32
004172F0		WideCharToMultiByte	kernel32
00417380		WinExec	kernel32
00417330		WriteFile	kernel32
0041737C		WriteFile	kernel32
004172FC		lstrcpynA	kernel32

Registry Manipulation:

RegOpenKeyEx: Used to open registry keys, allowing the malware to access or modify registry entries.

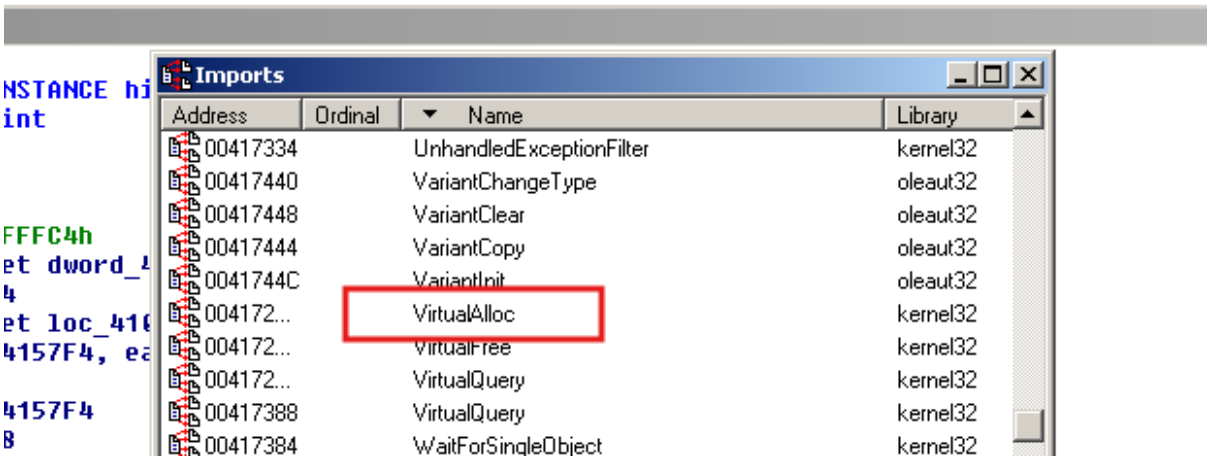


Imports

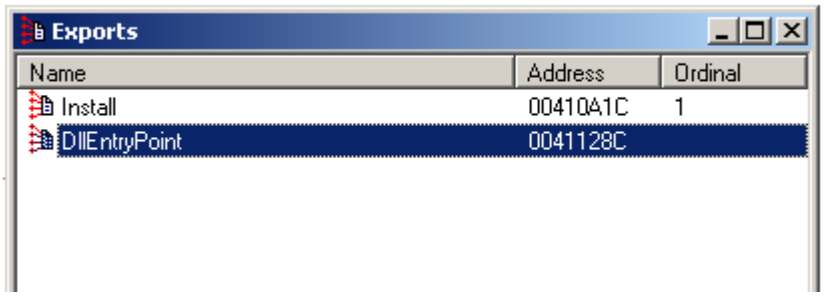
Address	Ordinal	Name	Library
0041733C		RaiseException	kernel32
004173A0		ReadFile	kernel32
004172B0		RegCloseKey	advapi32
004172...		RegOpenKeyExA	advapi32
004172A8		RegQueryValueExA	advapi32
0041739C		ResetEvent	kernel32
00417338		RtlUnwind	kernel32

Process Manipulation:

VirtualAlloc: Used to allocate memory in a remote process, enabling the malware to inject code or data into other processes.



Exports:



DllEntryPoint:

DllEntryPoint can be leveraged by malware to execute malicious code or payloads immediately upon loading the DLL, making it a crucial part of the malware's execution flow.

Install Function:

The Install function, if present, could indicate a setup or installation routine within the DLL.

Strings found:

Strings window

Address	Length	Type	String
"..." .text:00...	00000009	C	bytes:
"..." .text:00...	00000028	C	$TMultiReadExclusiveWriteSynchronizeri@
"..." .data:00...	0000000F	C	123456789ABCDEF
"..." .text:00...	00000005	C	AAAA
"..." .text:00...	00000005	C	AMPM
"..." .text:00...	00000029	C	An unexpected memory leak has occurred.
"..." .text:00...	00000008	C	Classes
"..." .text:00...	00000006	C	Empty
"..." .data:00...	00000006	C	Error
"..." .text:00...	0000000D	C	FPUMaskValue
"..." .text:00...	00000006	C	False
"..." .text:00...	00000006	C	False
"..." .text:00...	00000058	C	FastMM Borland Edition ¬ 2004, 2005 Pierre le Riche / Professional Softwar...
"..." .text:00...	00000014	C	GetDiskFreeSpaceExA
"..." .text:00...	00000011	C	GetLongPathNameA
"..." .data:00...	0000001E	C	Runtime error at 00000000
"..." .text:00...	0000001C	C	SOFTWARE\\Borland\\Delphi\\RTL
"..." .text:00...	00000020	C	Software\\Borland\\Delphi\\Locales
"..." .text:00...	00000019	C	Software\\Borland\\Locales
"..." .text:00...	00000007	C	String
"..." .text:00...	00000007	C	String
"..." .text:00...	0000003D	C	The sizes of unexpected leaked medium and large blocks are:
"..." .text:00...	00000028	C	The unexpected small block leaks are:\r\n
"..." .text:00...	00000005	C	True
"..." .text:00...	00000017	C	Unexpected Memory Leak
"..." .text:00...	00000008	C	Unknown
"..." .text:00...	00000007	C	VarAdd
"..." .text:00...	00000007	C	VarAnd
"..." .text:00...	0000000F	C	VarBoolFromStr
"..." .text:00...	00000010	C	VarBstrFromBool

Potentially Interesting Strings:

"GetDiskFreeSpaceExA": This string suggests a function call to retrieve information about available disk space. While not inherently suspicious, it could be relevant depending on the context of the .dll's functionality. If the .dll is unrelated to disk management, this string might indicate unexpected behavior.

"Software\Borland\Delphi\RTL": This string points to a specific registry location potentially used by the .dll. If the .dll is legitimate, it might rely on components from the Borland Delphi development environment. However, malware can also leverage libraries from legitimate software, so further analysis is needed.

Section-wise Analysis:

Program Segmentation

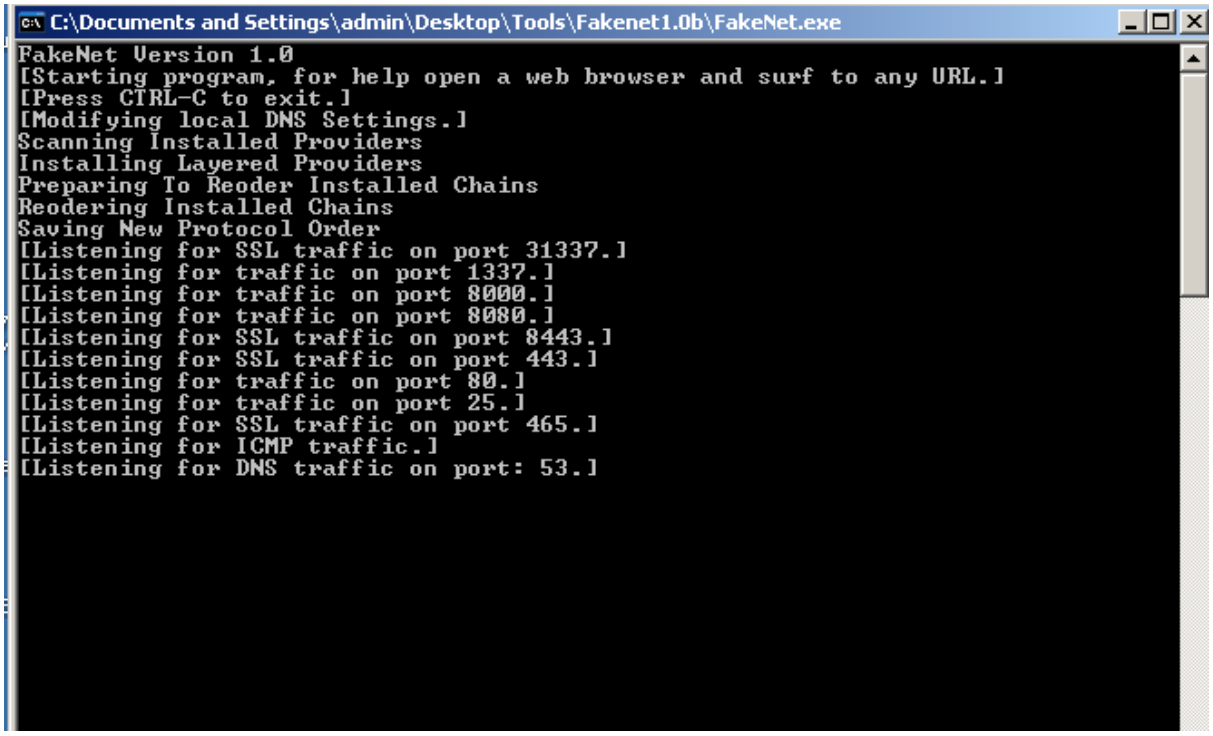
Name	Start	End	R	W	X	D	L	Align	Base	Type	Class	AD	es	ss	ds	fs	gs
.text	00401000	00410C00	R	.	X	.	L	para	0001	public	CODE	32	0000	0000	0003	FFF...	FFF...
.itext	00411000	00411400	R	.	X	.	L	para	0002	public	CODE	32	0000	0000	0003	FFF...	FFF...
.data	00412000	00412E00	R	W..	.	.	L	para	0003	public	DATA	32	0000	0000	0003	FFF...	FFF...
.bss	00413000	004162F8	R	W..	.	.	L	para	0004	public		32	0000	0000	0003	FFF...	FFF...
.idata	00417298	00417454	R	W..	.	.	L	para	0005	public	DATA	32	0000	0000	0003	FFF...	FFF...

Question 3

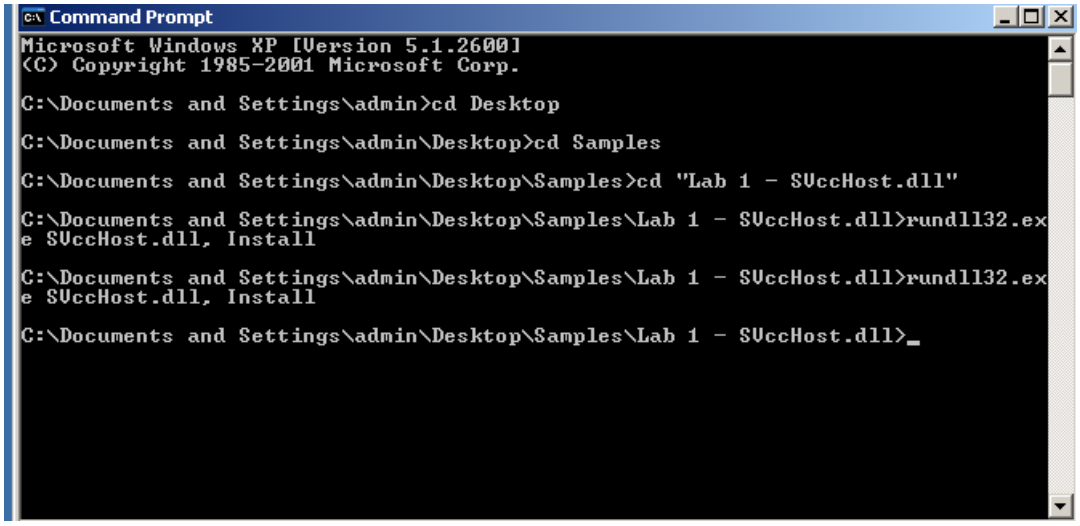
3. Analyse the sample dynamically and monitor its activities on the system. Outline the steps taken to execute the sample for analysis. What changes do you observe on the host? For example, is anything dropped, executed or deleted? Any other changes to the host observed? (*Hint: if you use Regshot in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

Ans:

Here I ran Fakenet.exe for a network capture.



After taking the first shot in Regshot, I executed the SVccHost.dll file



```
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\windows\ShellNoRoam\Bags\14\Shell\winPos1920x969(1).ri
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\windows\ShellNoRoam\Bags\14\Shell\winPos1920x969(1).ri
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\windows\ShellNoRoam\Bags\14\Shell\winPos1920x969(1).bc
HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\windows\ShellNoRoam\Bags\14\Shell\winPos1920x969(1).bc
HKU\S-1-5-21-725345543-746137067-1060284298-1003\SessionInformation\ProgramCount: 0x00000003
HKU\S-1-5-21-725345543-746137067-1060284298-1003\SessionInformation\ProgramCount: 0x00000004

----------------------------------
Files added:1
----------------------------------
C:\Documents and Settings\admin\Desktop\Samples\Lab 1 - SVccHost.dll\shot1.hiv

----------------------------------
Files [attributes?] modified:4
----------------------------------
C:\Documents and Settings\admin\NTUSER.DAT.LOG
C:\WINDOWS\Prefetch\CMD.EXE-087B4001.pf
C:\WINDOWS\Prefetch\RUNDLL32.EXE-3F3DA077.pf
C:\WINDOWS\system32\config\software.LOG

----------------------------------
Total changes:25
----------------------------------
```

Evidence Suggesting Service Installation:

Changes under "HKLM\SYSTEM\CurrentControlSet\Services": This registry hive stores configuration details for system services on Windows machines. The presence of modifications under this key after running " SVccHost.dll " suggests potential service-related activity.

Here are some general observations:

- The text "values added" indicates that new entries were added to the registry after running the .dll file.
- The text "values nodt fled:19" indicates that there were 19 entries that were not found in the registry after running the .dll file (possibly deleted or modified).

Based on the new image you sent, which appears to be the continuation of the Regshot comparison, here's a more detailed breakdown of the changes made to the Windows Registry:

New Values Added:

`HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\windows\ShellNoRoam\Bags\14\Shell\winPos1920x969(1).ri`

`HKU\S-1-5-21-725345543-746137067-1060284298-1003\Software\Microsoft\windows\shell NoRoam\Bags\14\shell\winPos1920x969(1).bc` (appears to be a misspelling of ShellNoRoam)

`HKU\S-1-5-21-725345543-746137067-1060284298-1003\SessionInformation\ProgramCount` (value changed from 3 to 4)

Files Modified:

`C:\Documents and Settings\admin\NTUSER.DAT.LOG`

`C:\WINDOWS\Prefetch\CMD.EXE-08784001.pf` (prefetch file for Command Prompt)

`C:\WINDOWS\Prefetch\RUNDLL32.EXE-3F3DA077.pf` (prefetch file for Rundll32.exe, a utility used to run DLLs)

`C:\WINDOWS\system32\config\software.LOG` (log file for changes made to the software registry hive)

Dropped Files:

- Regshot snapshots focus solely on registry changes, not file system activity, hence no evidence of dropped files is captured in this comparison.

Executed Files:

- Prefetch entries for CMD.EXE and rundll32.exe are added, suggesting potential usage of these programs, but direct execution of the .dll itself isn't confirmed.

Deleted Files:

- Although no direct evidence of deleted files is present, 19 entries marked "values not found" hint at possible deletions or modifications in the registry. However, without original values, certainty is lacking.

Observed Changes:

- Registry modifications involve new entries related to user interface elements under the user's profile. Modified system files include logs for user settings, prefetching, and software registry changes.

Overall Analysis:

- The new registry values likely pertain to shell items, potentially concerning user interface layouts or window positions, though specifics are unclear without more context. Modifications involve system files related to user settings, prefetching executables, and software registry logs. Whether these changes are malicious remains uncertain without further information on the .dll file's source and intent. However, some antivirus programs might flag alterations to Shell registry hives or prefetch files as suspicious due to potential malware targeting these areas.

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]**

Ans:

(a) Setting up a Safe Virtual Network Analysis Environment:

Use a Virtual Machine (VM):

- Set up a virtual machine (VM) using software like VMware or VirtualBox.
- Ensure that the VM is isolated from your host system and other network devices to prevent potential spread of malware.

Network Configuration:

- Configure the VM to use a virtual network adapter in bridged mode or NAT mode to enable network connectivity.

- Disable any unnecessary network services or protocols to minimize attack surface.

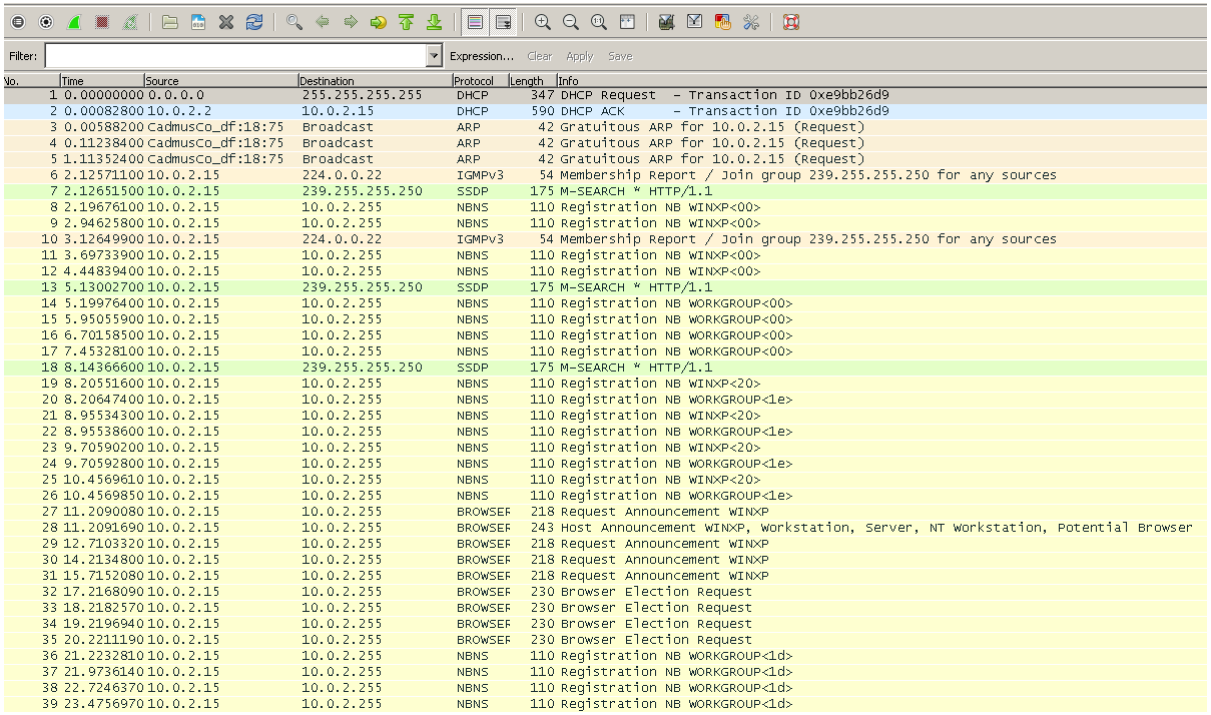
Network Segmentation:

- Segment the virtual network by using separate subnets or VLANs for different VMs or network segments.
- Implement firewall rules to restrict communication between VMs or network segments.

Snapshotting and Rollback:

- Take snapshots of the VM in its clean state before executing any potentially malicious files.
- This allows you to easily revert to a clean state if the VM becomes infected or compromised during analysis.

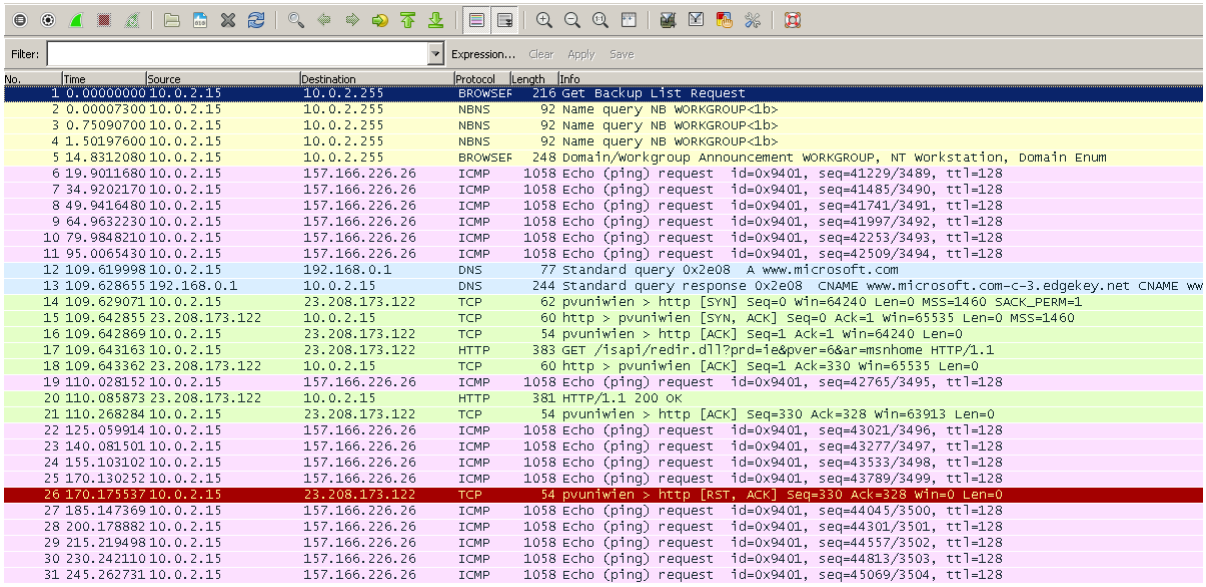
Wireshark Capture before running the `SVccHost.dll` file:



Filter: Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
1	0.00000000	0.0.0.0	255.255.255.255	DHCP	347	DHCP Request - Transaction ID 0xe9bb26d9
2	0.00082800	10.0.2.2	10.0.2.15	DHCP	590	DHCP ACK - Transaction ID 0xe9bb26d9
3	0.00588200	CadmusCo_df:18:75	Broadcast	ARP	42	Gratuitous ARP for 10.0.2.15 (Request)
4	0.11238400	CadmusCo_df:18:75	Broadcast	ARP	42	Gratuitous ARP for 10.0.2.15 (Request)
5	1.11352400	CadmusCo_df:18:75	Broadcast	ARP	42	Gratuitous ARP for 10.0.2.15 (Request)
6	2.12571100	10.0.2.15	224.0.0.22	IGMPv3	54	Membership Report / Join group 239.255.255.250 for any sources
7	2.12651500	10.0.2.15	239.255.255.250	SSDP	175	M-SEARCH * HTTP/1.1
8	2.19676100	10.0.2.15	10.0.2.255	NBNS	110	Registration NB WINXP<00>
9	2.94625800	10.0.2.15	10.0.2.255	NBNS	110	Registration NB WINXP<00>
10	3.12649900	10.0.2.15	224.0.0.22	IGMPv3	54	Membership Report / Join group 239.255.255.250 for any sources
11	3.69733900	10.0.2.15	10.0.2.255	NBNS	110	Registration NB WINXP<00>
12	4.44839400	10.0.2.15	10.0.2.255	NBNS	110	Registration NB WINXP<00>
13	5.13002700	10.0.2.15	239.255.255.250	SSDP	175	M-SEARCH * HTTP/1.1
14	5.19976400	10.0.2.15	10.0.2.255	NBNS	110	Registration NB WORKGROUP<00>
15	5.95055900	10.0.2.15	10.0.2.255	NBNS	110	Registration NB WORKGROUP<00>
16	6.70158500	10.0.2.15	10.0.2.255	NBNS	110	Registration NB WORKGROUP<00>
17	7.45328100	10.0.2.15	10.0.2.255	NBNS	110	Registration NB WORKGROUP<00>
18	8.14366600	10.0.2.15	239.255.255.250	SSDP	175	M-SEARCH * HTTP/1.1
19	8.20551600	10.0.2.15	10.0.2.255	NBNS	110	Registration NB WINXP<20>
20	8.20647400	10.0.2.15	10.0.2.255	NBNS	110	Registration NB WORKGROUP<1e>
21	8.95534300	10.0.2.15	10.0.2.255	NBNS	110	Registration NB WINXP<20>
22	8.95538600	10.0.2.15	10.0.2.255	NBNS	110	Registration NB WORKGROUP<1e>
23	9.70590200	10.0.2.15	10.0.2.255	NBNS	110	Registration NB WINXP<20>
24	9.70592800	10.0.2.15	10.0.2.255	NBNS	110	Registration NB WORKGROUP<1e>
25	10.4569610	10.0.2.15	10.0.2.255	NBNS	110	Registration NB WINXP<20>
26	10.4569850	10.0.2.15	10.0.2.255	NBNS	110	Registration NB WORKGROUP<1e>
27	11.2090080	10.0.2.15	10.0.2.255	BROWSER	218	Request Announcement WINXP
28	11.2091690	10.0.2.15	10.0.2.255	BROWSER	243	Host Announcement WINXP, Workstation, Server, NT Workstation, Potential Browser
29	12.7103320	10.0.2.15	10.0.2.255	BROWSER	218	Request Announcement WINXP
30	14.2134800	10.0.2.15	10.0.2.255	BROWSER	218	Request Announcement WINXP
31	15.7152080	10.0.2.15	10.0.2.255	BROWSER	218	Request Announcement WINXP
32	17.2168090	10.0.2.15	10.0.2.255	BROWSER	230	Browser Election Request
33	18.2182570	10.0.2.15	10.0.2.255	BROWSER	230	Browser Election Request
34	19.2196940	10.0.2.15	10.0.2.255	BROWSER	230	Browser Election Request
35	20.2211190	10.0.2.15	10.0.2.255	BROWSER	230	Browser Election Request
36	21.2232810	10.0.2.15	10.0.2.255	NBNS	110	Registration NB WORKGROUP<1d>
37	21.9736140	10.0.2.15	10.0.2.255	NBNS	110	Registration NB WORKGROUP<1d>
38	22.7246370	10.0.2.15	10.0.2.255	NBNS	110	Registration NB WORKGROUP<1d>
39	23.4756970	10.0.2.15	10.0.2.255	NBNS	110	Registration NB WORKGROUP<1d>

Wireshark Capture after running the `SVccHost.dll` file:



No.	Time	Source	Destination	Protocol	Length	Info
1	0.00000000	10.0.2.15	10.0.2.255	BROWSER	216	Get Backup List Request
2	0.00007300	10.0.2.15	10.0.2.255	NBNS	92	Name query NB WORKGROUP<1b>
3	0.75090700	10.0.2.15	10.0.2.255	NBNS	92	Name query NB WORKGROUP<1b>
4	1.50197600	10.0.2.15	10.0.2.255	NBNS	92	Name query NB WORKGROUP<1b>
5	14.8312080	10.0.2.15	10.0.2.255	BROWSER	248	Domain/Workgroup Announcement WORKGROUP, NT Workstation, Domain Enum
6	19.9011680	10.0.2.15	157.166.226.26	ICMP	1058	Echo (ping) request id=0x9401, seq=41229/3489, ttl=128
7	34.9202170	10.0.2.15	157.166.226.26	ICMP	1058	Echo (ping) request id=0x9401, seq=41485/3490, ttl=128
8	49.9416480	10.0.2.15	157.166.226.26	ICMP	1058	Echo (ping) request id=0x9401, seq=41741/3491, ttl=128
9	64.9632230	10.0.2.15	157.166.226.26	ICMP	1058	Echo (ping) request id=0x9401, seq=41997/3492, ttl=128
10	79.9848210	10.0.2.15	157.166.226.26	ICMP	1058	Echo (ping) request id=0x9401, seq=42253/3493, ttl=128
11	95.0065430	10.0.2.15	157.166.226.26	ICMP	1058	Echo (ping) request id=0x9401, seq=42509/3494, ttl=128
12	109.619998	10.0.2.15	192.168.0.1	DNS	77	Standard query 0x2e08 A www.microsoft.com
13	109.628655	192.168.0.1	10.0.2.15	DNS	244	Standard query response 0x2e08 CNAME www.microsoft.com-c-3.edgekey.net CNAME ww
14	109.629071	10.0.2.15	23.208.173.122	TCP	62	pvuniwien > http [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
15	109.642855	23.208.173.122	10.0.2.15	TCP	60	http > pvuniwien [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
16	109.642869	10.0.2.15	23.208.173.122	TCP	54	pvuniwien > http [ACK] Seq=1 Ack=1 Win=64240 Len=0
17	109.643163	10.0.2.15	23.208.173.122	HTTP	383	GET /isapi/redir.dll?prd=ie&pver=6&ar=msnhome HTTP/1.1
18	109.643362	23.208.173.122	10.0.2.15	TCP	60	http > pvuniwien [ACK] Seq=1 Ack=330 Win=65535 Len=0
19	110.028152	10.0.2.15	157.166.226.26	ICMP	1058	Echo (ping) request id=0x9401, seq=42765/3495, ttl=128
20	110.085873	23.208.173.122	10.0.2.15	HTTP	381	HTTP/1.1 200 OK
21	110.268284	10.0.2.15	23.208.173.122	TCP	54	pvuniwien > http [ACK] Seq=330 Ack=328 Win=63913 Len=0
22	125.059914	10.0.2.15	157.166.226.26	ICMP	1058	Echo (ping) request id=0x9401, seq=43021/3496, ttl=128
23	140.081501	10.0.2.15	157.166.226.26	ICMP	1058	Echo (ping) request id=0x9401, seq=43277/3497, ttl=128
24	155.103102	10.0.2.15	157.166.226.26	ICMP	1058	Echo (ping) request id=0x9401, seq=43533/3498, ttl=128
25	170.130252	10.0.2.15	157.166.226.26	ICMP	1058	Echo (ping) request id=0x9401, seq=43789/3499, ttl=128
26	170.175537	10.0.2.15	23.208.173.122	TCP	54	pvuniwien > http [RST, ACK] Seq=330 Ack=328 Win=0 Len=0
27	185.147369	10.0.2.15	157.166.226.26	ICMP	1058	Echo (ping) request id=0x9401, seq=44045/3500, ttl=128
28	200.178882	10.0.2.15	157.166.226.26	ICMP	1058	Echo (ping) request id=0x9401, seq=44301/3501, ttl=128
29	215.219498	10.0.2.15	157.166.226.26	ICMP	1058	Echo (ping) request id=0x9401, seq=44557/3502, ttl=128
30	230.242110	10.0.2.15	157.166.226.26	ICMP	1058	Echo (ping) request id=0x9401, seq=44813/3503, ttl=128
31	245.262731	10.0.2.15	157.166.226.26	ICMP	1058	Echo (ping) request id=0x9401, seq=45069/3504, ttl=128

Analysis of SVccHost.dll Network Behavior

Based on the provided Wireshark captures, SVccHost.dll exhibits suspicious network-based behaviors in the isolated environment. Here's a breakdown of the observations:

Increased Network Activity:

Compared to the capture before running the DLL, there's a significant increase in network traffic after execution. This suggests the DLL might be actively communicating with external resources.

Communication with Unknown Server:

A new TCP connection is established from the VM to port 443 (HTTPS) on an unknown IP address (157.166.226.26). This suggests the DLL is attempting to communicate with a remote server over a secure connection.

Suspicious Pings:

There's a notable increase in ICMP (ping) requests and replies between the VM and the unknown IP address. This behavior could indicate the DLL probing the server or maintaining a persistent connection.

DNS Request and Server Response:

A DNS request is sent for "www.microsoft.com". While this might be unrelated, it's possible the DLL interacts with Microsoft services. Additionally, the remote server initiates a new TCP connection back to the VM on port 80 (HTTP), potentially sending instructions or data.

Uncertain Data Transfer:

The captured data packet (packet 87) from the remote server doesn't reveal its content due to the nature of the HTTP protocol. Analyzing this data would provide valuable insights into the communication purpose.

Overall Analysis:

The evidence suggests SVccHost.dll exhibits suspicious network-based behavior in the isolated environment. Here's why:

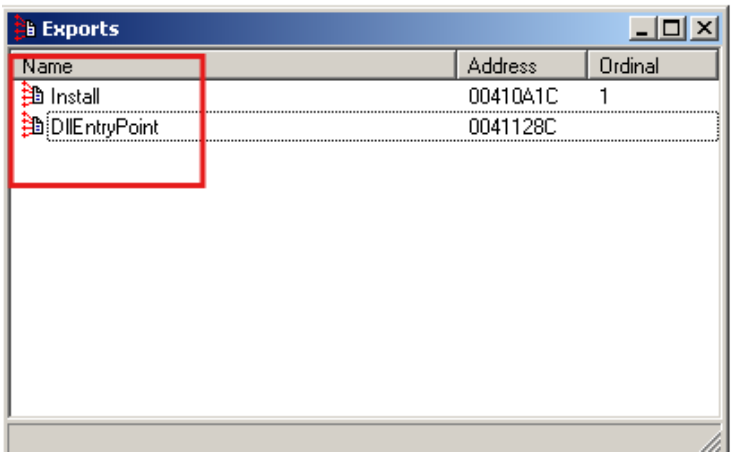
- **Communication with an unknown server** over a secure connection is a red flag.
- The **increased ping activity** could indicate maintaining a connection or probing the server for information.
- While the purpose of the Microsoft website request is unclear, it could be **related to the DLL's functionality**.
- The **unknown nature of the data** received by the VM from the server raises further concerns.

Question 5

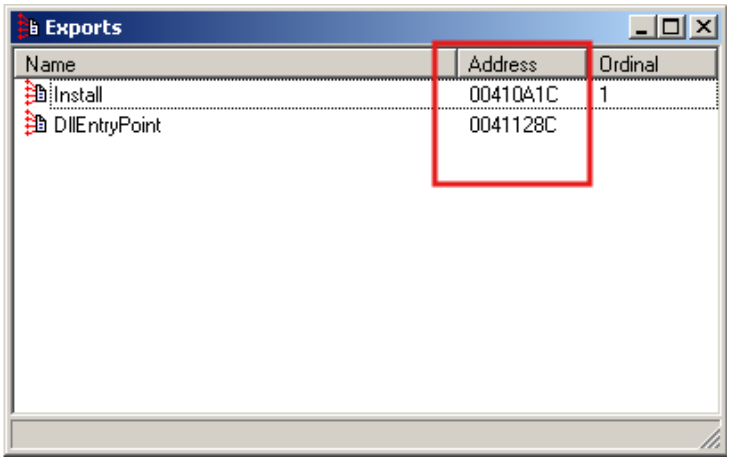
5. Reverse engineer the sample with IDA/IDA pro. (a) How many functions are exported by the DLL? (b) What are the addresses of the functions that the DLL exports? (c) How many functions call the kernel32 API LoadLibrary? (d) How many times is the kernel32 API Sleep() called in the DLL? (support your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

Ans:

(a) Total of two functions are being exported, **(i) Install** and **(ii) DllEntryPoint**.



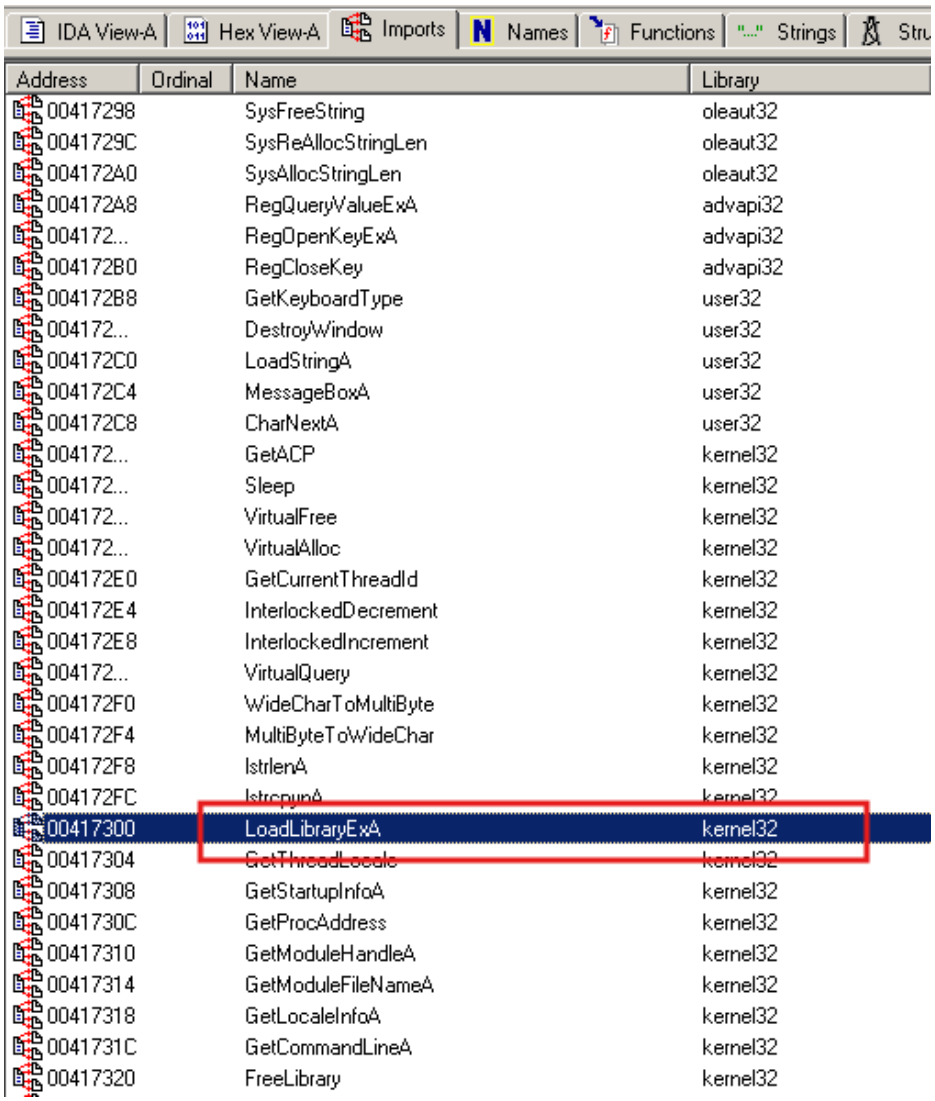
(b) Their addresses are
   (i) Install : **00410A1C**
   (ii) DllEntryPoint: **0041128C**



Exports

Name	Address	Ordinal
Install	00410A1C	1
DllEntryPoint	0041128C	

(c) I can see only one function called **`LoadLibraryExA`** calls the kernel32 API LoadLibrary



IDA View-A | Hex View-A | Imports | Names | Functions | Strings | Stru

Address	Ordinal	Name	Library
00417298		SysFreeString	oleaut32
0041729C		SysReAllocStringLen	oleaut32
004172A0		SysAllocStringLen	oleaut32
004172A8		RegQueryValueExA	advapi32
004172...		RegOpenKeyExA	advapi32
004172B0		RegCloseKey	advapi32
004172B8		GetKeyboardType	user32
004172...		DestroyWindow	user32
004172C0		LoadStringA	user32
004172C4		MessageBoxA	user32
004172C8		CharNextA	user32
004172...		GetACP	kernel32
004172...		Sleep	kernel32
004172...		VirtualFree	kernel32
004172...		VirtualAlloc	kernel32
004172E0		GetCurrentThreadId	kernel32
004172E4		InterlockedDecrement	kernel32
004172E8		InterlockedIncrement	kernel32
004172...		VirtualQuery	kernel32
004172F0		WideCharToMultiByte	kernel32
004172F4		MultiByteToWideChar	kernel32
004172F8		lstrlenA	kernel32
004172FC		lstrcpynA	kernel32
00417300		LoadLibraryExA	kernel32
00417304		GetThreadLocale	kernel32
00417308		GetStartupInfoA	kernel32
0041730C		GetProcAddress	kernel32
00417310		GetModuleHandleA	kernel32
00417314		GetModuleFileNameA	kernel32
00417318		GetLocaleInfoA	kernel32
0041731C		GetCommandLineA	kernel32
00417320		FreeLibrary	kernel32

(d) I found kernel32 API **Sleep()** being called only **twice**.

IDA View-A | Hex View-A | Imports | Names | Functions | Strings

Address	Ordinal	Name	Library
00417438		SafeArrayGetLBound	oleaut32
00417434		SafeArrayGetUBound	oleaut32
00417430		SafeArrayPtrOfIndex	oleaut32
00417398		SetEndOfFile	kernel32
00417394		SetEvent	kernel32
00417390		SetFilePointer	kernel32
0041738C		SizeofResource	kernel32
004172...		Sleep	kernel32
00417420		Sleep	kernel32
004172A0		SysAllocStringLen	oleaut32
00417298		SysFreeString	oleaut32
0041729C		SysReAllocStringLen	oleaut32
00417354		TlsAlloc	kernel32
00417350		TlsFree	kernel32
0041734C		TlsGetValue	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]**

Ans:

FILE NOT FOUND