

13 November 2022

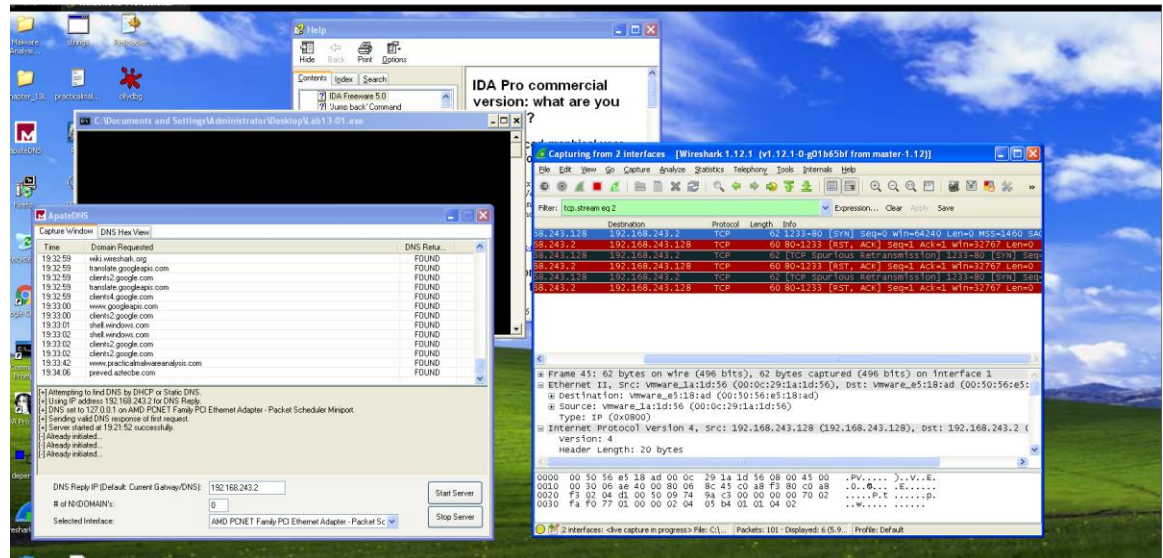
ASSIGNMENT 10

LAB 13-1

Analyze the malware found in the file Lab13-01.exe.

Questions

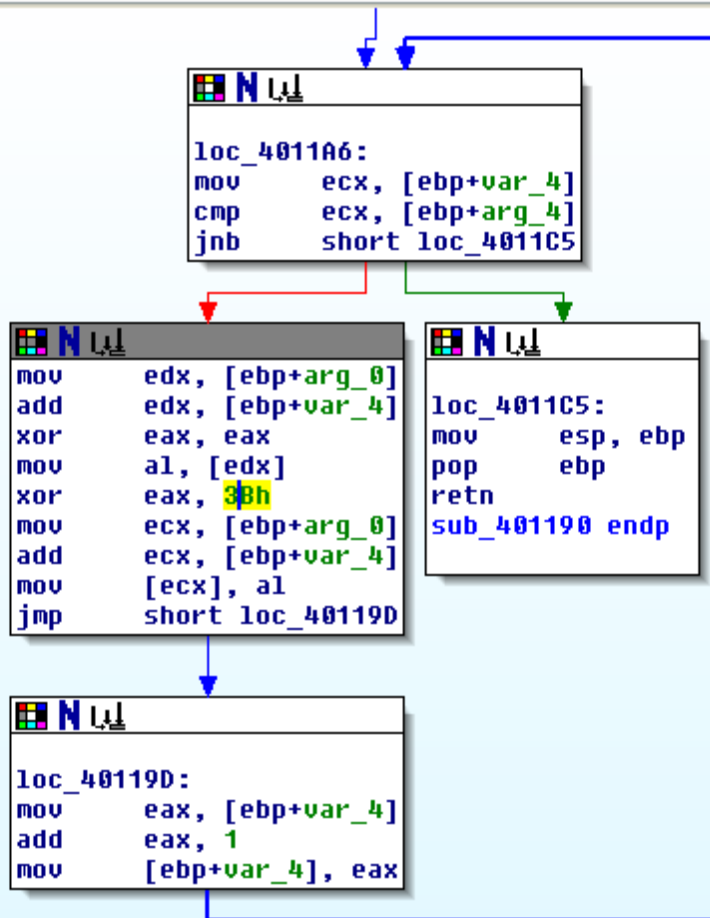
- Compare the strings in the malware (from the output of the strings command) with the information available via dynamic analysis. Based on this comparison, which elements might be encoded?



The beacon contains two strings that are absent from the virus. (The strings are not printed when the strings command is executed.) The domain `www.practicalmalwareanalysis.com` is one example. The second is the GET request route, which is composed of random letters.

- Use IDA Pro to look for potential encoding by searching for the string `xor`. What type of encoding do you find?

```
• .text:00402BE5          inc     ebx
• .text:00402BE6          xor     [eax], dh
• .text:00402BE8 : 7000000BD BYTES: COLLAPSED FUNCTION un
```



```

sub_401000 proc near
var_2= byte ptr -2
var_1= byte ptr -1
arg_0= dword ptr 8
arg_4= dword ptr 0Ch
arg_8= dword ptr 10h

push    ebp
mov     ebp, esp
push    ecx
mov     eax, [ebp+arg_0]
xor     ecx, ecx
mov     cl, [eax]
sar     ecx, 2
mov     edx, [ebp+arg_4]
mov     al, ds:byte_4050E8[ecx]
mov     [edx], al
mov     ecx, [ebp+arg_0]
xor     edx, edx
mov     dl, [ecx]
and     edx, 3
shl     edx, 4
mov     eax, [ebp+arg_0]
xor     ecx, ecx
mov     cl, [eax+1]
and     ecx, 0F0h
sar     ecx, 4
or      edx, ecx
mov     eax, [ebp+arg_4]
mov     cl, ds:byte_4050E8[edx]
mov     [eax+1], cl
cmp     [ebp+arg_8], 1
jle     short loc_401077

```

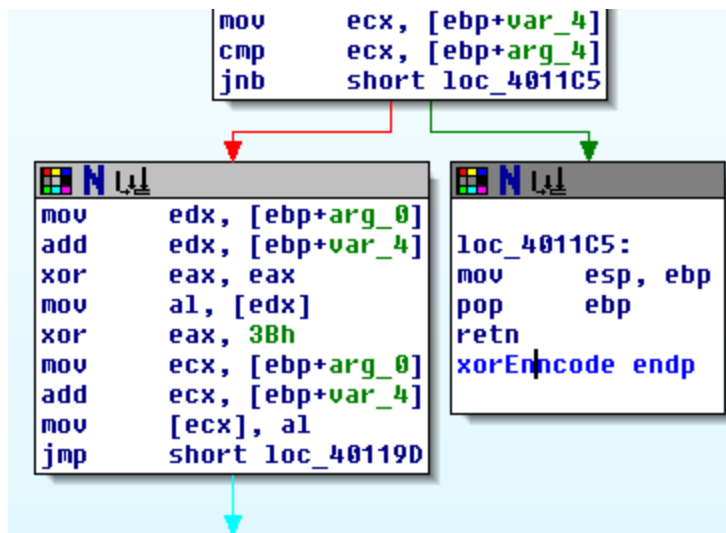
The xor eax,3Bh instruction is found in sub 401190 , as seen in the above screenshot. As there are several xor strings in IDAPro's text data. This is part of a single-byte Xor decoding algorithm that decodes the to-be-contacted domain.

iii. What is the key used for encoding and what content does it encode?

```

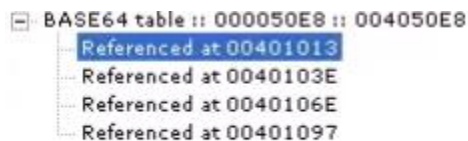
LPVOID __stdcall LockResource(HGLOBAL hResData)
    extrn LockResource:dword ; DATA XREF: sub_401300+96Tr
HGLOBAL __stdcall LoadResource(HMODULE hModule,HRSRC hResInfo)
    extrn LoadResource:dword ; DATA XREF: sub_401300+7FTr
HGLOBAL __stdcall GlobalAlloc(UINT uFlags,DWORD dwBytes)
    extrn GlobalAlloc:dword ; DATA XREF: sub_401300+6ETr
DWORD __stdcall SizeofResource(HMODULE hModule,HRSRC hResInfo)
    extrn SizeofResource:dword ; DATA XREF: sub_401300+5FTr
HRSRC __stdcall FindResourceA(HMODULE hModule,LPCSTR lpName,LPCSTR lpType)
    extrn FindResourceA:dword ; DATA XREF: sub_401300+41Tr
HMODULE __stdcall GetModuleHandleA(LPCSTR lpModuleName)
    extrn GetModuleHandleA:dword ; DATA XREF: sub_401300+16Tr
void __stdcall Sleep(DWORD dwMilliseconds)
    extrn Sleep:dword ; DATA XREF: _main+46Tr
    ; _main+63Tr
BOOL __stdcall GetStringTypeW(DWORD dwInfoType,LPCWSTR lpSrcStr,int cchSrc,LPWORD lpCharType)
    extrn GetStringTypeW:dword
    ; DATA XREF: __crtGetStringTypeA+3FTr
    ; __crtGetStringTypeA+12DTr
BOOL __stdcall GetStringTypeA(LCID Locale,DWORD dwInfoType,LPCSTR lpSrcStr,int cchSrc,LPWORD lpCharType)
    extrn GetStringTypeA:dword
    ; DATA XREF: __crtGetStringTypeA+59Tr
    ; __crtGetStringTypeA+8DTr
int __stdcall LCMMapStringW(LCID Locale,DWORD dwMapFlags,LPCWSTR lpSrcStr,int cchSrc,LPWSTR lpDestStr,int cchDest)
    extrn LCMMapStringW:dword ; DATA XREF: __crtLCMapStringA+42Tr
    ; __crtLCMapStringA+14DTr ...
int __stdcall LCMMapStringA(LCID Locale,DWORD dwMapFlags,LPCSTR lpSrcStr,int cchSrc,LPSTR lpDestStr,int cchDest)
    extrn LCMMapStringA:dword ; DATA XREF: __crtLCMapStringA+5ETr
    ; __crtLCMapStringA+77Tr
int __stdcall MultiByteToWideChar(UINT CodePage,DWORD dwFlags,LPCSTR lpMultiByteStr,int cchMultiByte,LPWSTR lpWideCharStr

```



For a further analysis one can rename sub_401190 to *xorEnncode*. The byte 0x3B is used for single-byte XOR encoding. The raw data resource with index 101 is a buffer that decodes to www.practicalmalwareanalysis.com.

- iv. Use the static tools FindCrypt2, Krypto ANALyzer (KANAL), and the IDA Entropy Plugin to identify any other encoding mechanisms. What do you find?



FindCrypt2 was unable to locate any encoding techniques. The Base64 table described above is located at 004050E8 and is used four times. The base64 table can be found within the Kanal plugin provided within PIED. With IDA Entropy, one may test for a high level of entropy. After configuring the chunk size to 64 and the maximum entropy to 5.95. One can conclude that the .rdata section has a high entropy. Therefore, one can find and conclude the Base64 encoding string of

"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"

- v. What type of encoding is used for a portion of the network traffic sent by the malware?

As the URI Path, the request to the server comprises a Base64-encoded portion of the hostname.

- vi. Where is the Base64 function in the disassembly?

The Base64 function is located at sub 4010B1 in the disassembly. This information can be found within the IDAPro application. When analyzing in the application one can find through the method of searching for the sub manually or in graph mode.

- vii. What is the maximum length of the Base64-encoded data that is sent? What is encoded?

Maximum length is sixteen bytes. Before Base64 encoding the hostname, Lab13-01.exe copies a maximum of 12 bytes from it. A 12-byte string yields a 16-byte base64 string.

- viii. In this malware, would you ever see the padding characters (= or ==) in the Base64-encoded data?

If the hostname length is less than 12 bytes and is not evenly divisible by 3, padding characters may be utilized.

- ix. What does this malware do?

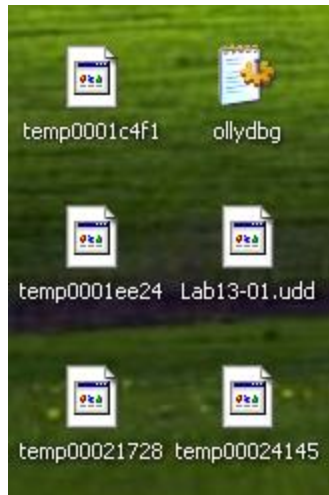
This virus sends an GET request with its hostname every 30 seconds to www.practicalmalwareanalysis.com until it returns a response beginning with the letter o. This might be a bot communicating to a command-and-control server that it is online.

LAB 13-2

Analyze the malware found in the file Lab13-02.exe.

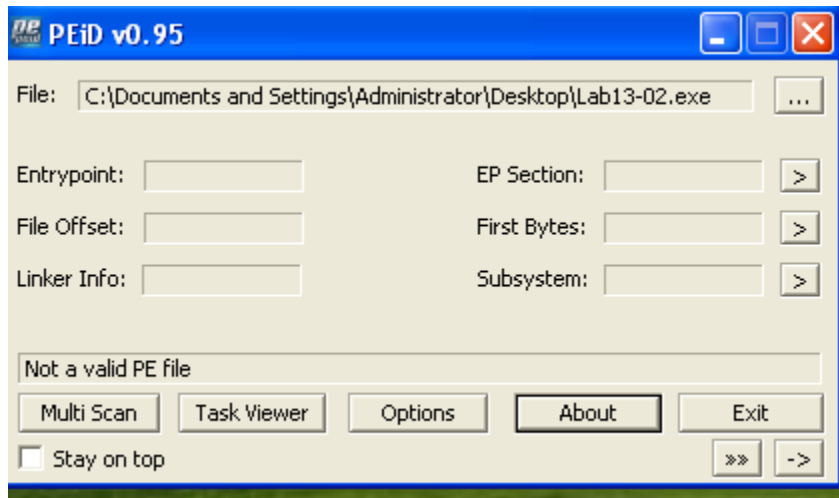
Questions

- i. Using dynamic analysis, determine what this malware creates.



The malware generates enormous, apparently random files with names beginning with temp and ending with eight digits in the current directory. Finally, the infection generates unreadable files. One can find these temp files either on their desktop after opening the malware or analyzing the malware under the application Process Monitor.

- ii. Use static techniques such as an xor search, FindCrypt2, KANAL, and the IDA Entropy Plugin to look for potential encoding. What do you find?



The xor search approach reveals probable functions associated with encoding at sub_401570 and sub_401739. Though when finding the potential of encoding the plugins failed to identify any. Provided above is the screenshot of PEiD could not find a response of the malware, therefore, one cannot use the plugin the analyze.

- iii. Based on your answer to question 1, which imported function would be a good prospect for finding the encoding functions? Based on your answer to question

1, which imported function would be a good prospect for finding the encoding functions?

WriteFile would be an excellent starting point. Typically, data is encrypted just before being output to a file or network. The function responsible for encoding the material may be identified.

iv. Where is the encoding function in the disassembly?

The function of encoding is sub 40181F. While some of the instructions displayed a succession of three more functions that call each other, others depict a single function. Each has what seem to be encoding instructions, XORs, and shifts.

v. Trace from the encoding function to the source of the encoded content. What is the content?

```

[ebp+hdc], 0
0 ; nIndex
ds:GetSystemMetrics
[ebp+var_1C], eax
1 ; nIndex
ds:GetSystemMetrics
[ebp+cy], eax
ds:GetDesktopWindow
hWnd, eax
eax, hWnd
eax ; hWnd
ds:GetDC
hDC, eax
ecx, hDC
ecx ; hdc
ds:CreateCompatibleDC
[ebp+hdc], eax
edx, [ebp+cy]
edx ; cy
eax, [ebp+var_1C]
eax ; cx
ecx, hDC
ecx ; hdc
ds:CreateCompatibleBitmap
[ebp+h], eax
edx, [ebp+h]
edx ; h
eax, [ebp+hdc]
eax ; hdc
ds:SelectObject
0CC0020h ; rop
0 ; y1
0 ; x1
ecx, hDC
ecx ; hdcSrc
edx, [ebp+cy]
edx ; cy
eax, [ebp+var_1C]
eax ; cx
0 ; y
0 ; x
ecx, [ebp+hdc]
ecx ; hdc
ds:BitBlt
edx, [ebp+pv]
edx ; pv
18h ; c
eax, [ebp+h]
eax ; h
ds:GetObjectA

```

The content consists of a desktop bitmap. Every 10 seconds, this software captures screenshots, encodes them, and saves them to a file. It comprises a set of function calls for capturing a desktop screenshot and returning it as a bitmap.

- vi. Can you find the algorithm used for encoding? If not, how can you decode the content?

Because algorithms are nonstandard and difficult to establish, instrumentation is the most efficient method for decoding communications. There is just an encoding procedure, not a decoding process. Reconstructing the inverse of the encoding function or using instrumentation would be necessary to decode the material.

- vii. Using instrumentation, can you recover the original source of one of the encoded files?

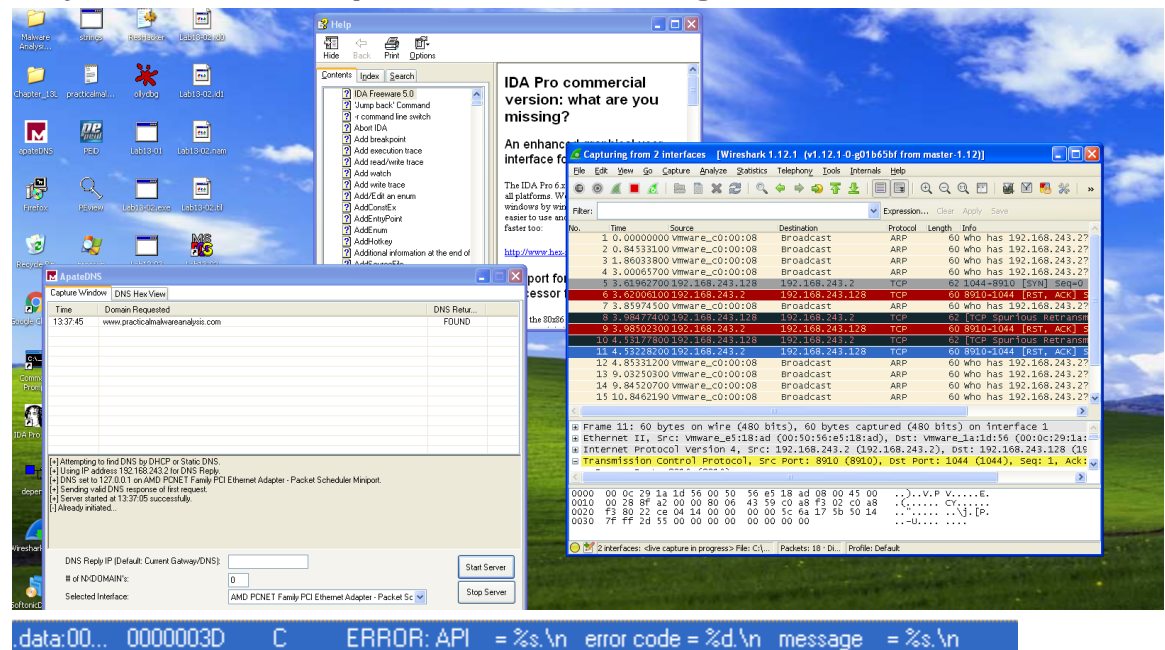
It is possible to recover the original source by creating a python script to decrypt. In the script one should created and have in mind open the e encrypted file that has already been written to the filesystem. As well as reading the file into the memory, calculating the size, etc. In the end one can run this script to find the original interface of the malware.

LAB 13-3

Analyze the malware found in the file Lab13-03.exe.

Questions

- i. Compare the output of strings with the information available via dynamic analysis. Based on this comparison, which elements might be encoded?



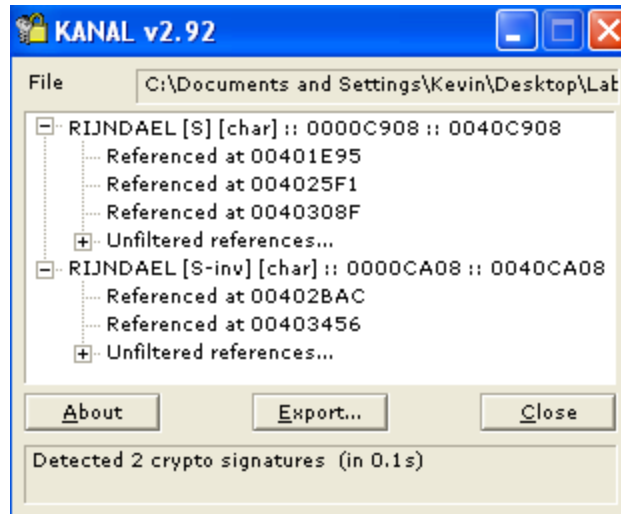
The results of a dynamic analysis may disclose encoded material that seems random. There are no discernible strings in the output of the software, and nothing else implies encoding. There were not any strings that stood out when analyzing using Wireshark and IDAPro.

- ii. Use static analysis to look for potential encoding by searching for the string xor. What type of encoding do you find?

.text:00401135	xor	eax, eax
.text:0040123C	xor	eax, eax
.text:00401310	xor	ecx, ecx
.text:00401341	xor	eax, eax
.text:00401357	xor	eax, eax
.text:0040136D	xor	eax, eax
.text:004014A5	xor	eax, eax
.text:004014BB	xor	eax, eax
.text:00401873	xor	eax, eax
.text:004019A5	xor	eax, eax
.text:00401A53	xor	eax, eax
.text:00401D51	xor	edx, edx
.text:00401D69	xor	eax, eax
.text:00401D88	xor	eax, eax
.text:00401DA7	xor	eax, eax
.text:00401EBD	xor	eax, edx
.text:00401ED8	xor	eax, edx
.text:00401EF3	xor	eax, edx
.text:00401F08	xor	eax, edx
.text:00401F13	xor	edx, eax
.text:00401F56	xor	eax, [esi+edx*4+414h]
.text:00401FB1	xor	edx, [esi+ecx*4+414h]
.text:00402028	xor	edx, ecx
.text:00402046	xor	edx, ecx
.text:00402064	xor	edx, ecx
.text:00402070	xor	ecx, edx
.text:004020B3	xor	edx, [esi+ecx*4+414h]
.text:004021E3	xor	ecx, ds:dword_40EF08[edx*4]
.text:004021F6	xor	ecx, ds:dword_40F308[edx*4]

The search for xor instructions shows six distinct functions that may be related with encoding, although the kind of encoding is unclear. Even if there are several xor calls being executed in the malware's .text files, this is not the case.

- iii. Use static tools like FindCrypt2, KANAL, and the IDA Entropy Plugin to identify any other encoding mechanisms. How do these findings compare with the XOR findings?



All three methods identify the Advanced Encryption Standard algorithm, which relates to each of the six discovered XOR functions. In addition, the IDA Entropy Plugin discovers a bespoke Base64 indexing string that is unrelated to xor operations.

- iv. Which two encoding techniques are used in this malware?

Malware employs Base64 encoding and AES ciphering.

- v. For each encoding technique, what is the key?

```
'DEFGHIJKLMNOPQRSTUVWXYZABCdefghijklmnopqrstuvwxyzab0123456789+/',0
db 'ijklmnopqrstuvwxyz',0 ; DA
align 4
```

For Base64 encoding, Malware need simply the Base64 Table. In this particular instance, it is a bespoke Base64 Table. The AES encryption key is `ijklmnopqrstuvwxyz`. The index string is the key for the custom Base64 cipher: `"DEFGHIJKLMNOPQRSTUVWXYZABCdefghijklmnopqrstuvwxyzab01223456789+/"`

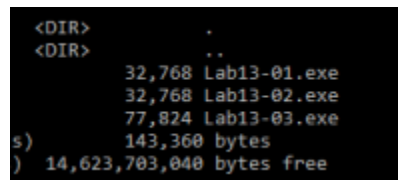
- vi. For the cryptographic encryption algorithm, is the key sufficient? What else must be known?

Obviously, the AES's key is crucial. However, other properties, like block cipher mode, block size, and key length, are required. We already know that the Cipher mode is AES Cipher Block Chaining based on the study. The initialization vector is necessary to decode the first block in this instance.

- vii. What does this malware do? What does this malware do?

This malware establishes a remote shell connection to a C2 server. The instructions are encoded using a bespoke Base64 encoding, and the output is encrypted with AES before being sent to the socket. Once the command has been decrypted, it is sent to cmd.exe for execution. The return results are encrypted using AES and sent to the attacker's server.

- viii. Create code to decrypt some of the content produced during dynamic analysis.
What is this content?



```
<DIR>      .
<DIR>      ..
             32,768 Lab13-01.exe
             32,768 Lab13-02.exe
             77,824 Lab13-03.exe
s)          143,360 bytes
)  14,623,703,040 bytes free
```

After the decryption of the content during dynamic analysis one can find the network content. The output of the entire decryption script gives an out put of the information of the following host :

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.