Sheridan College

Course	**INFO43921** **Malicious Code Design and Defense**
Activity Title	**Static and Dynamic Analysis**
Student Name	Jackson Yuan
Lab performed on (Date):	March 1, 2023

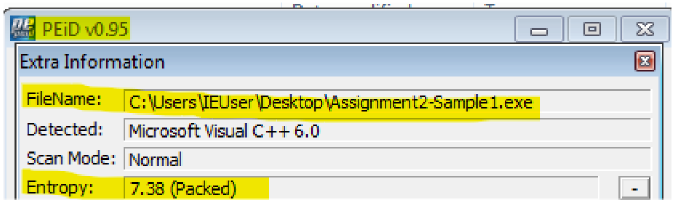
Objectives

- To utilize tools that analyze malware statically and dynamically
- Be able to determine a malwares type, format, packed or unpacked, entropy, etc.
- Be able to use API Miner and observer what kind of API calls were made during execution (Log observations)
- Be able to explain the type of family, attributes and artifacts of the malware
- Be able to dynamically observe the strings of the malware
- Use tools such as ProcMon to observer changes within the system

Output Section for Sample #1

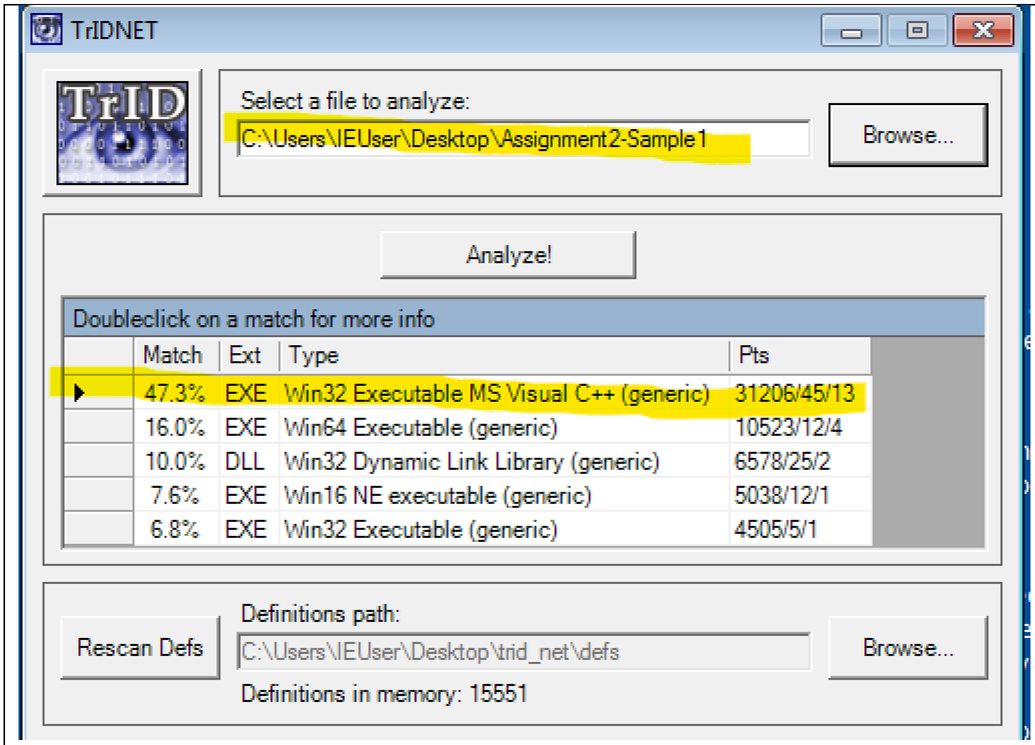
Output#1: Static Analysis

Entropy: To determine the entropy of sample #1, we use PEiD and we conclude that it came up to 7.38.



FileName: C:\Users\IEUser\Desktop\Assignment2-Sample1.exe
Detected: Microsoft Visual C++ 6.0
Scan Mode: Normal
Entropy: 7.38 (Packed)

File Type/Format: To find this out, we use TrID to scan it. After scanning, we find out that it is a Windows 32 executable as shown below. To double check its legitimacy, we also use CFF Explorer and check the magic bytes which confirms our initial results as shown below in the screenshots.



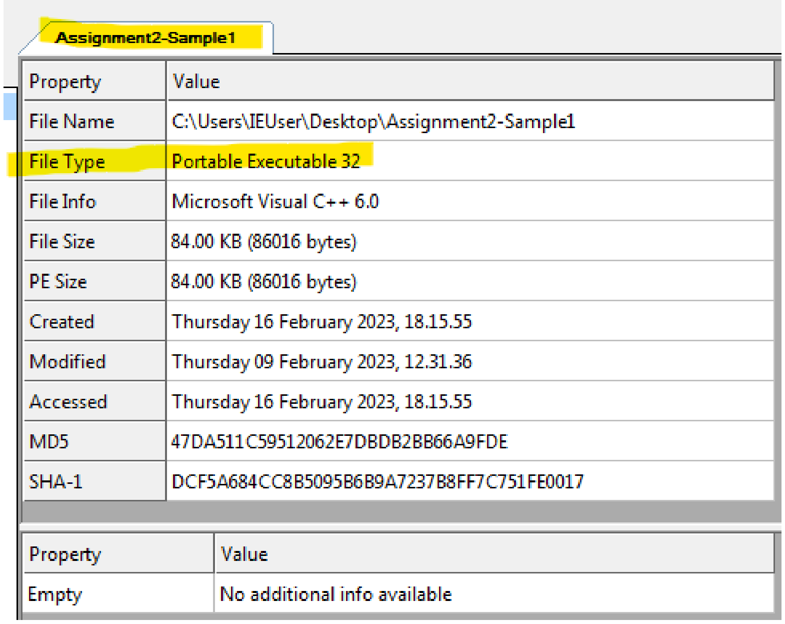
Select a file to analyze: C:\Users\IEUser\Desktop\Assignment2-Sample1

Doubleclick on a match for more info

Match	Ext	Type	Pts
47.3%	EXE	Win32 Executable MS Visual C++ (generic)	31206/45/13
16.0%	EXE	Win64 Executable (generic)	10523/12/4
10.0%	DLL	Win32 Dynamic Link Library (generic)	6578/25/2
7.6%	EXE	Win16 NE executable (generic)	5038/12/1
6.8%	EXE	Win32 Executable (generic)	4505/5/1

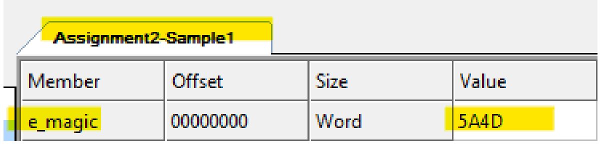
Definitions path: C:\Users\IEUser\Desktop\trid_net\defs

Definitions in memory: 15551

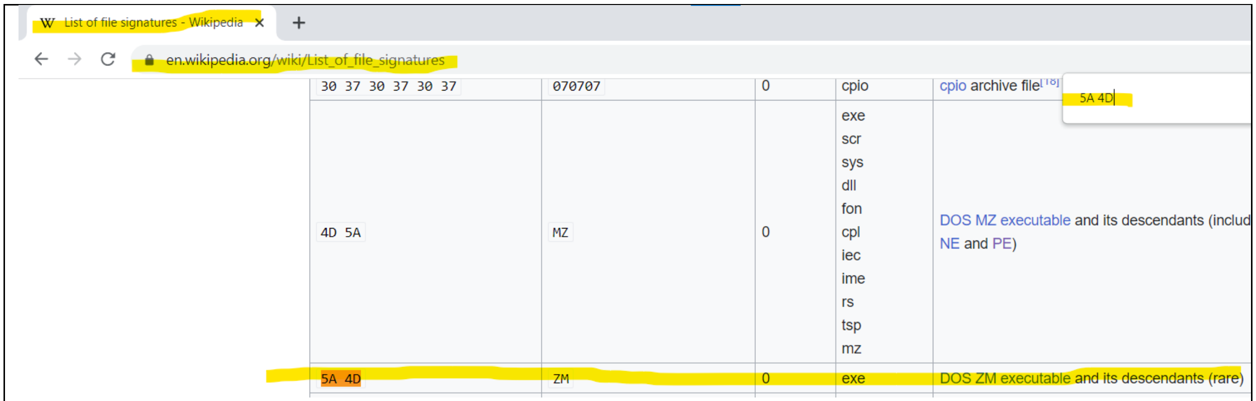


Property	Value
File Name	C:\Users\IEUser\Desktop\Assignment2-Sample1
File Type	Portable Executable 32
File Info	Microsoft Visual C++ 6.0
File Size	84.00 KB (86016 bytes)
PE Size	84.00 KB (86016 bytes)
Created	Thursday 16 February 2023, 18.15.55
Modified	Thursday 09 February 2023, 12.31.36
Accessed	Thursday 16 February 2023, 18.15.55
MD5	47DA511C59512062E7DBDB2BB66A9FDE
SHA-1	DCF5A684CC8B5095B6B9A7237B8FF7C751FE0017

Property	Value
Empty	No additional info available



Member	Offset	Size	Value
e_magic	00000000	Word	5A4D

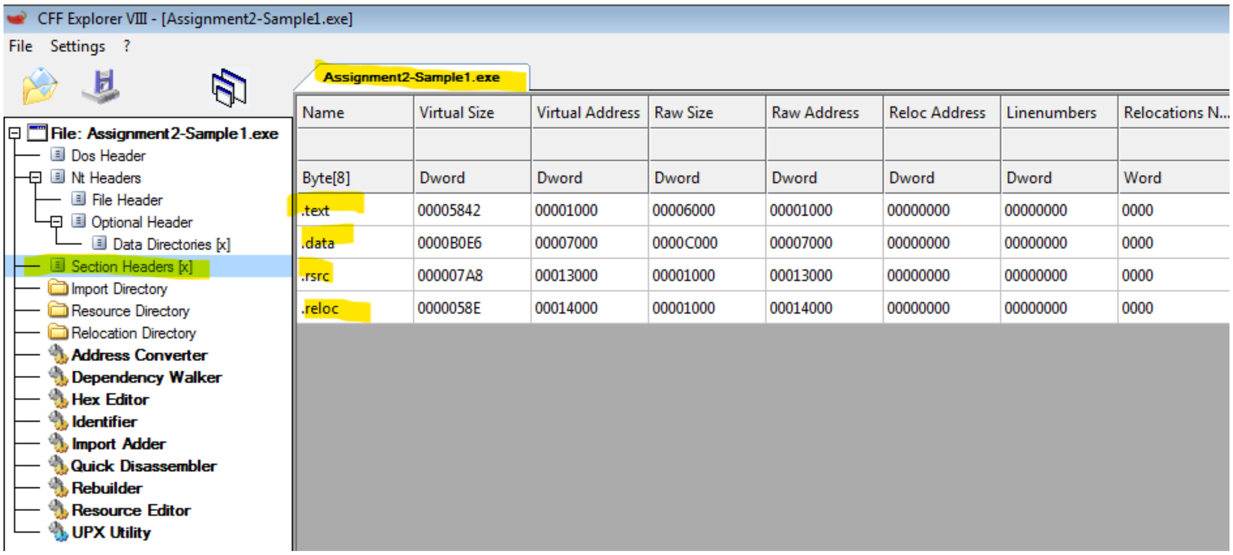

List of file signatures - Wikipedia
en.wikipedia.org/wiki/List_of_file_signatures
5A 4D

30 37 30 37 30 37	070707	0	cpio	cpio archive file[18]
4D 5A	MZ	0	exe scr sys dll fon cpl iec ime rs tsp mz	DOS MZ executable and its descendants (includ NE and PE)
5A 4D	ZM	0	exe	DOS ZM executable and its descendants (rare)

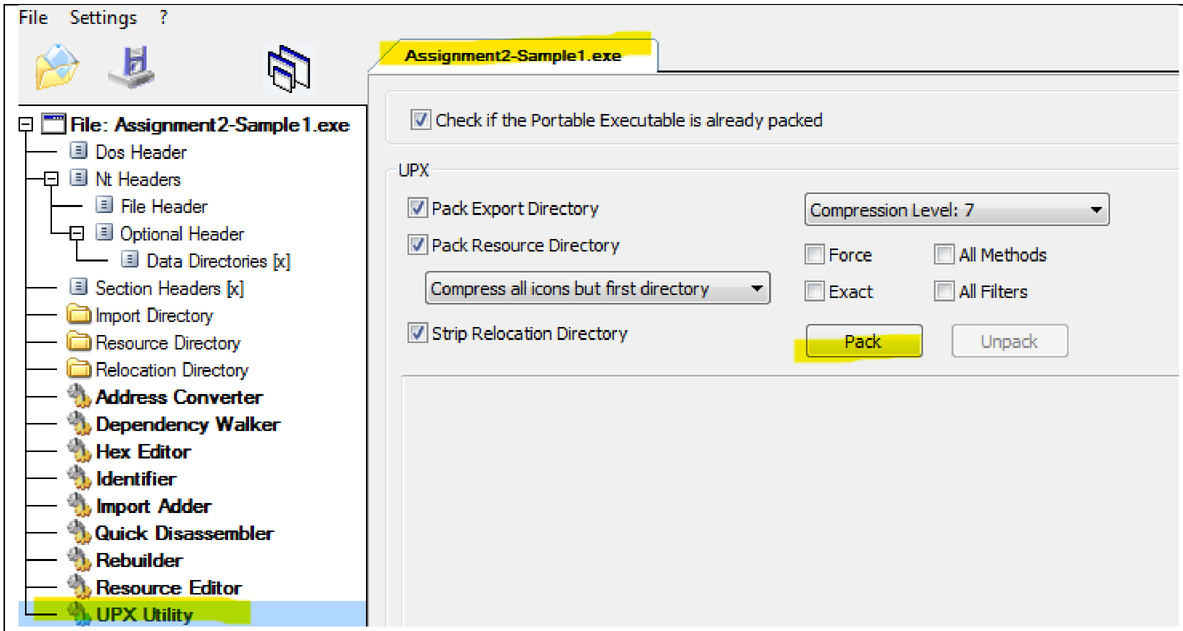
Figure 1: Wikipedia page; list of file signatures

Source: [1]

Packing: To determine if sample #1 is packed, we check the section headers of the sample through CFF Explorer. The reason for this is because if the sample was packed, you wouldn't be able to see the specific sections. Then again, to verify, we check again in the UPX Utility as this will show if the sample has been packed or not. In this case it shows if the user wants to "pack" the sample thus its unpacked.



Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word
.text	00005842	00001000	00006000	00001000	00000000	00000000	0000
.data	0000B0E6	00007000	0000C000	00007000	00000000	00000000	0000
.rsrc	000007A8	00013000	00001000	00013000	00000000	00000000	0000
.reloc	0000058E	00014000	00001000	00014000	00000000	00000000	0000



☑ Check if the Portable Executable is already packed

☑ Pack Export Directory

☑ Pack Resource Directory

Compress all icons but first directory

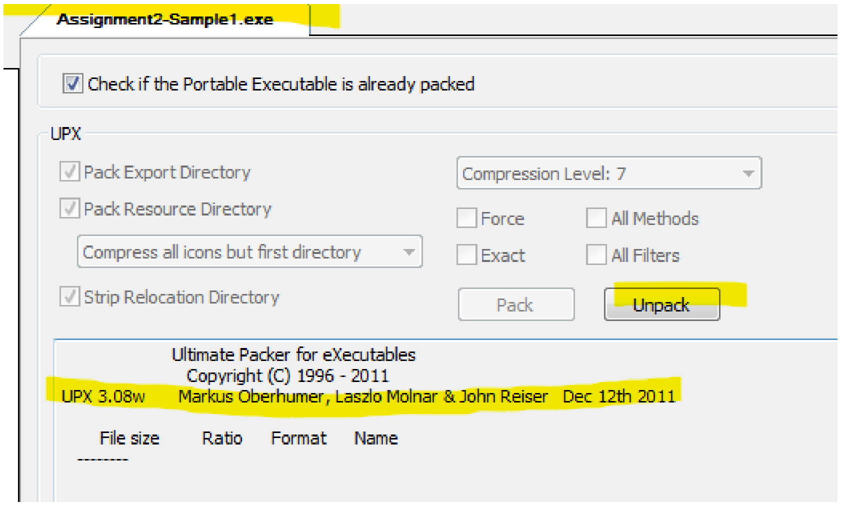
☑ Strip Relocation Directory

Compression Level: 7

□ Force □ All Methods

□ Exact □ All Filters

(Before clicking packed ↑)



☑ Check if the Portable Executable is already packed

☑ Pack Export Directory

☑ Pack Resource Directory

Compress all icons but first directory

☑ Strip Relocation Directory

Compression Level: 7

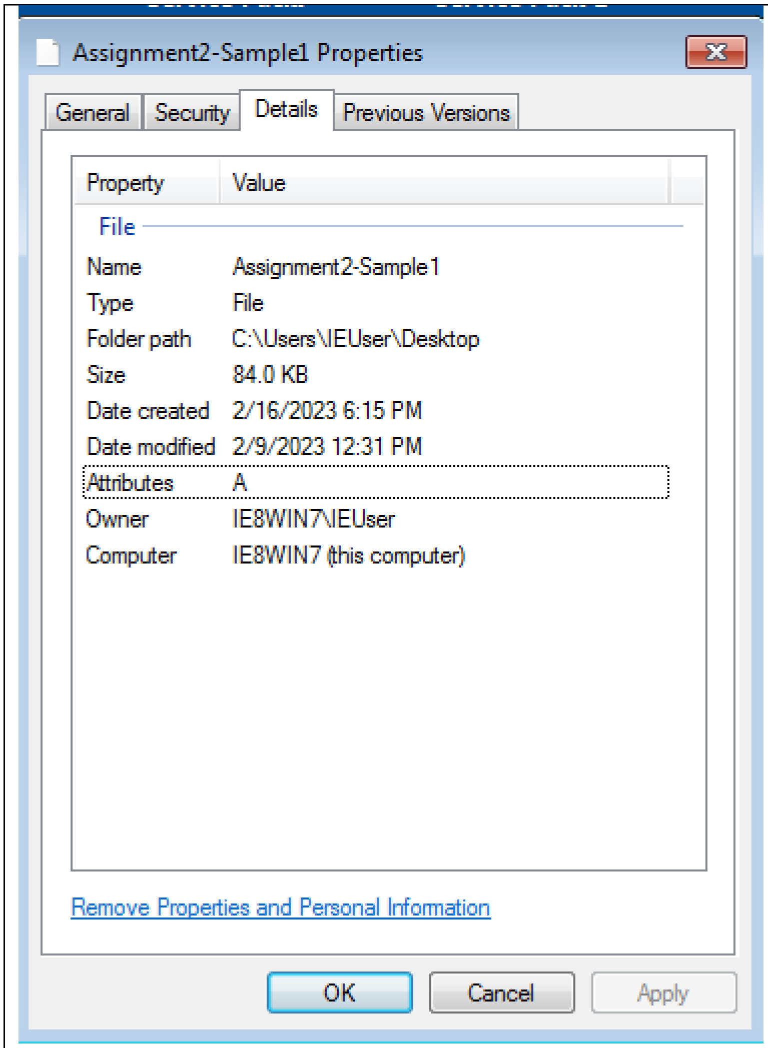
□ Force □ All Methods

□ Exact □ All Filters

```
                    Ultimate Packer for eXecutables
                      Copyright (C) 1996 - 2011
UPX 3.08w      Markus Oberhumer, Laszlo Molnar & John Reiser   Dec 12th 2011

        File size         Ratio      Format      Name
   --------
```

(After clicking “packed” ↑)

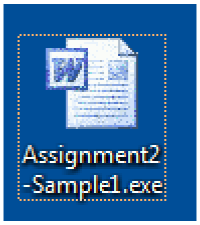
File Properties: Some of the file properties I have observed was that sample had no digital signature to it as well as thumbnail faking as a MS Word document. It used several dll libraries as shown in the screenshot below.

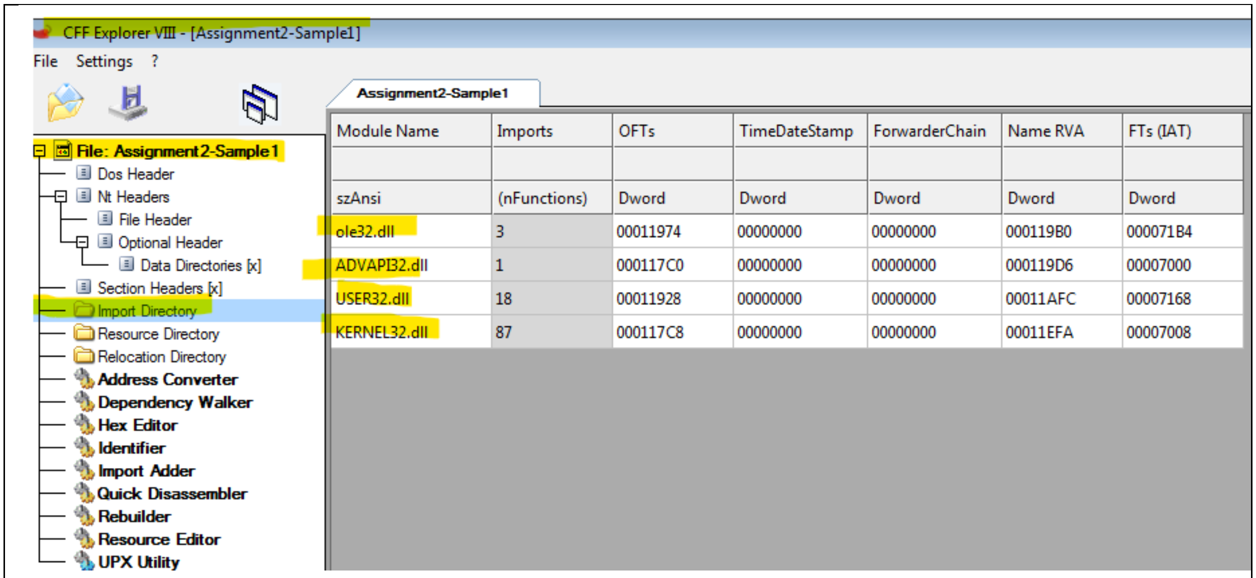


Assignment2-Sample1 Properties
General
Security
Details
Previous Versions

Property	Value
File	
Name	Assignment2-Sample1
Type	File
Folder path	C:\Users\IEUser\Desktop
Size	84.0 KB
Date created	2/16/2023 6:15 PM
Date modified	2/9/2023 12:31 PM
Attributes	A
Owner	IE8WIN7\IEUser
Computer	IE8WIN7 (this computer)

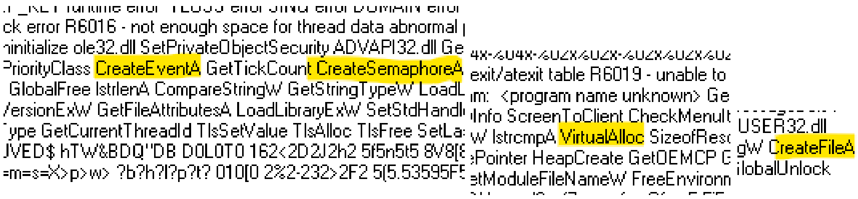
Remove Properties and Personal Information
OK
Cancel
Apply





Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
ole32.dll	3	00011974	00000000	00000000	000119B0	000071B4
ADVAPI32.dll	1	000117C0	00000000	00000000	000119D6	00007000
USER32.dll	18	00011928	00000000	00000000	00011AFC	00007168
KERNEL32.dll	87	000117C8	00000000	00000000	00011EFA	00007008

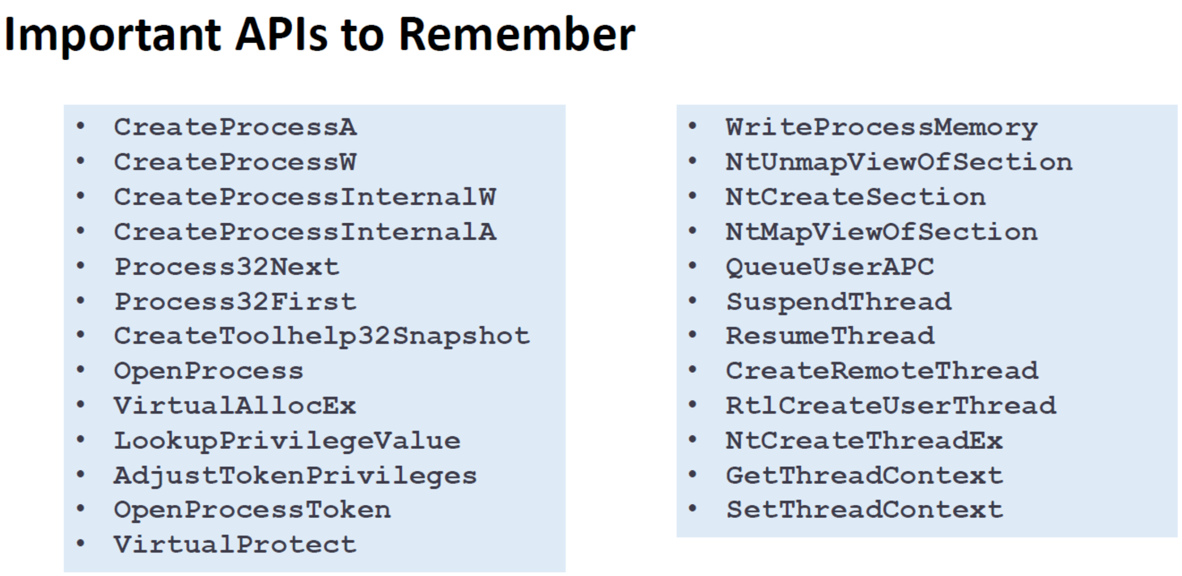
Static String Analysis: During the static string analysis, I observed suspicious API calls such as "CreateEventA", "CreateSemaphoreA", "VirtualAlloc", etc. which were it exhibits traits of a user-mode code injection as discussed in class.



```
ck error R6016 - not enough space for thread data abnormal
ninitialize ole32.dll SetPrivateObjectSecurity ADVAPI32.dll Ge
PriorityClass CreateEventA GetTickCount CreateSemaphoreA
 GlobalFree lstrlenA CompareStringW GetStringTypeW LoadL
/ersionExW GetFileAttributesA LoadLibraryExW SetStdHandl
Type GetCurrentThreadId TlsSetValue TlsAlloc TlsFree SetLa
JVED$ hTW&BDQ''DB D0L0T0 162<2D2J2h2 5f5n5t5 8V8[
=m=s=X>p>w> ?b?h?l?p?t? 010[0 2%2-232>2F2 5(5.53595F
```

```
4x-%U4x-%U2x%U2x-%U2x%U2x%U
exit/atexit table R6019 - unable to
m:  <program name unknown> Ge
Info ScreenToClient CheckMenuIt
W lstrcmpA VirtualAlloc SizeofRes
Pointer HeapCreate GetOEMCP G
etModuleFileNameW FreeEnvironn
```

```
USER32.dll
gW CreateFileA
ilobalUnlock
```



Important APIs to Remember

- CreateProcessA
- CreateProcessW
- CreateProcessInternalW
- CreateProcessInternalA
- Process32Next
- Process32First
- CreateToolhelp32Snapshot
- OpenProcess
- VirtualAllocEx
- LookupPrivilegeValue
- AdjustTokenPrivileges
- OpenProcessToken
- VirtualProtect

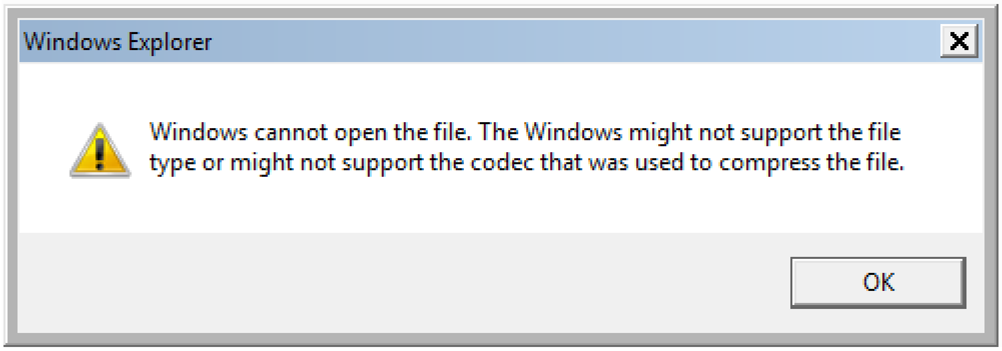
- WriteProcessMemory
- NtUnmapViewOfSection
- NtCreateSection
- NtMapViewOfSection
- QueueUserAPC
- SuspendThread
- ResumeThread
- CreateRemoteThread
- RtlCreateUserThread
- NtCreateThreadEx
- GetThreadContext
- SetThreadContext

Figure 2: API listings from class lectures

Source: [2]

Output#2: System Observation Before & After

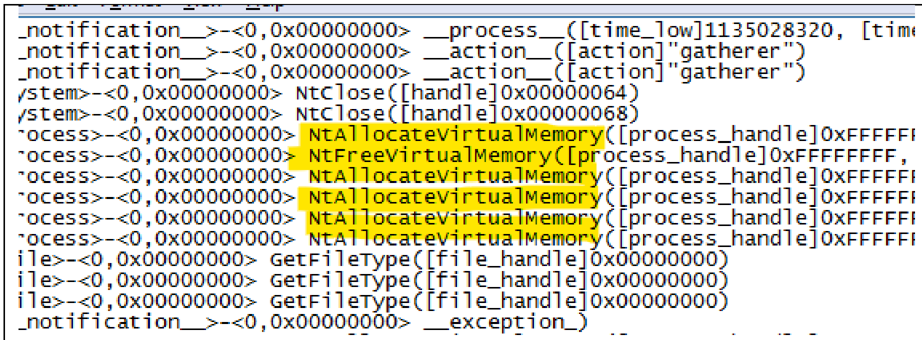
When I ran the file in administrative mode, the error that popped stating that it was unable to open the file due to an unsupported file/codec type as shown below. The pop up looked suspicious as there does not exist such a pop up with the that message. The closest thing to it was a Windows Media Player error [3]. Moreover, the pop-up windows title was also not real as "Windows Explorer" warning messages does not exist.



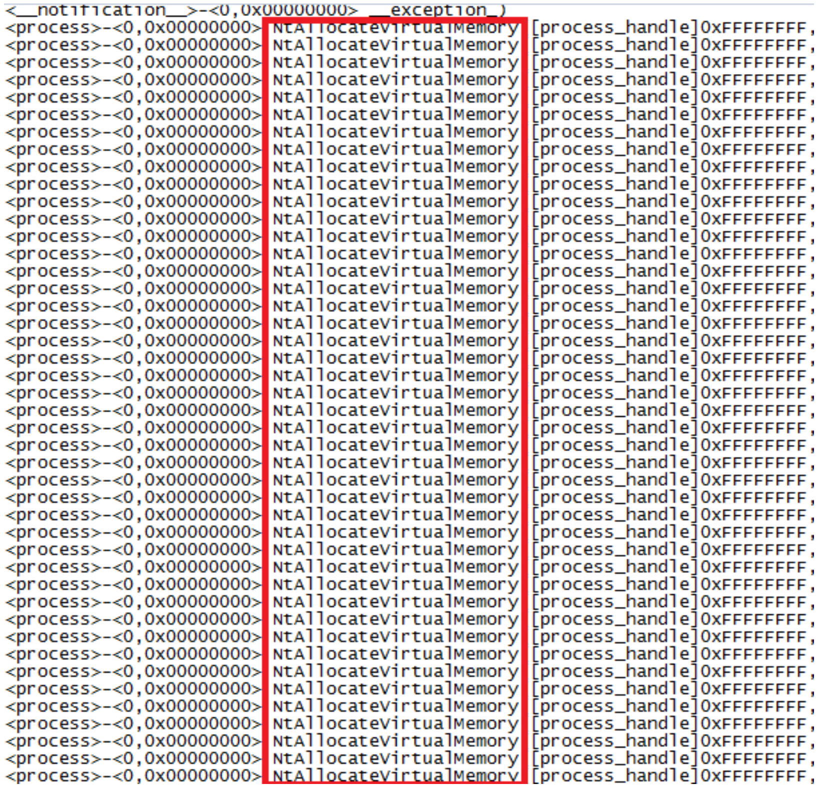
Windows cannot open the file. The Windows might not support the file type or might not support the codec that was used to compress the file.

Output#3: Analyzing API Logs

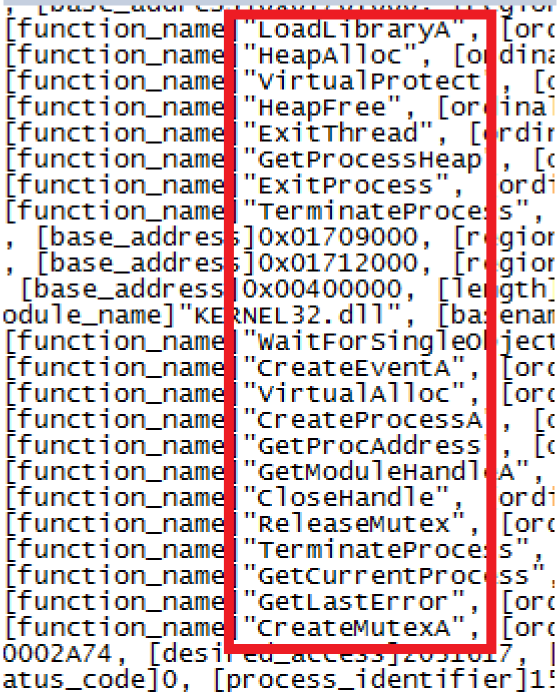
After running the sample through API Miner, the API logs show that malware first uses the NtAllocateVirtualMemory to allocate memory for storing malicious code or data. Next it calls NtFreeVirtualMemory to free some space before creating more memory by calling NtAllocateVirtualMemory again. It then loads it a library called "LoadLibraryA" in which "HeapAlloc" is then called from memory to store this malicious library. "VirtualProtect" is then called afterwards to change the permission in that region of memory in order to write or execute. Finally, the sample calls dll libraries such as "CreateEventA", and "CreateProcessA". Finally, it closes everything by "CloseHandle", "ReleaseMutex", "TerminateProcess" to minimize detection.



```
_notification__>-<0,0x00000000> __process__([time_low]1135028320, [time
_notification__>-<0,0x00000000> __action__([action]"gatherer")
_notification__>-<0,0x00000000> __action__([action]"gatherer")
ystem>-<0,0x00000000> NtClose([handle]0x00000064)
ystem>-<0,0x00000000> NtClose([handle]0x00000068)
ocess>-<0,0x00000000> NtAllocateVirtualMemory([process_handle]0xFFFFFF
ocess>-<0,0x00000000> NtFreeVirtualMemory([process_handle]0xFFFFFFFF,
ocess>-<0,0x00000000> NtAllocateVirtualMemory([process_handle]0xFFFFFF
ocess>-<0,0x00000000> NtAllocateVirtualMemory([process_handle]0xFFFFFF
ocess>-<0,0x00000000> NtAllocateVirtualMemory([process_handle]0xFFFFFF
ocess>-<0,0x00000000> NtAllocateVirtualMemory([process_handle]0xFFFFFF
ile>-<0,0x00000000> GetFileType([file_handle]0x00000000)
ile>-<0,0x00000000> GetFileType([file_handle]0x00000000)
ile>-<0,0x00000000> GetFileType([file_handle]0x00000000)
_notification__>-<0,0x00000000> __exception_)
```



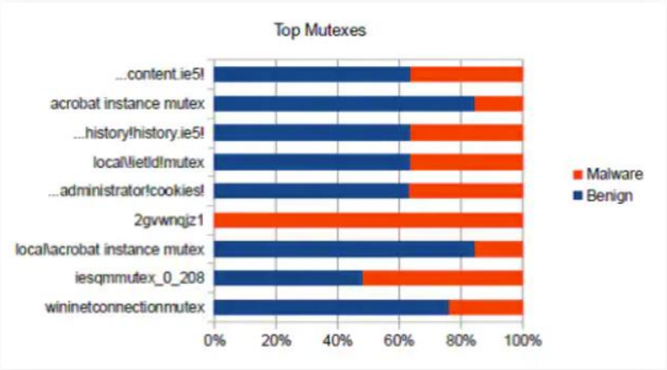
```
<__notification__>-<0,0x00000000> __exception_)
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
<process>-<0,0x00000000> NtAllocateVirtualMemory [process_handle]0xFFFFFFFF,
```



```
[function_name]"LoadLibraryA", [ord
[function_name]"HeapAlloc", [ordina
[function_name]"VirtualProtect", [o
[function_name]"HeapFree", [ordinal
[function_name]"ExitThread", [ordin
[function_name]"GetProcessHeap", [o
[function_name]"ExitProcess", [ordi
[function_name]"TerminateProcess",
, [base_address]0x01709000, [region
, [base_address]0x01712000, [region
 [base_address]0x00400000, [length]
odule_name]"KERNEL32.dll", [basenam
[function_name]"WaitForSingleObject
[function_name]"CreateEventA", [ord
[function_name]"VirtualAlloc", [ord
[function_name]"CreateProcessA", [o
[function_name]"GetProcAddress", [o
[function_name]"GetModuleHandleA",
[function_name]"CloseHandle", [ordi
[function_name]"ReleaseMutex", [ord
[function_name]"TerminateProcess",
[function_name]"GetCurrentProcess",
[function_name]"GetLastError", [ord
[function_name]"CreateMutexA", [ord
0002A74, [desired_access]2031617, [
atus_code]0, [process_identifier]15
```

Output#4: Artifacts Analysis

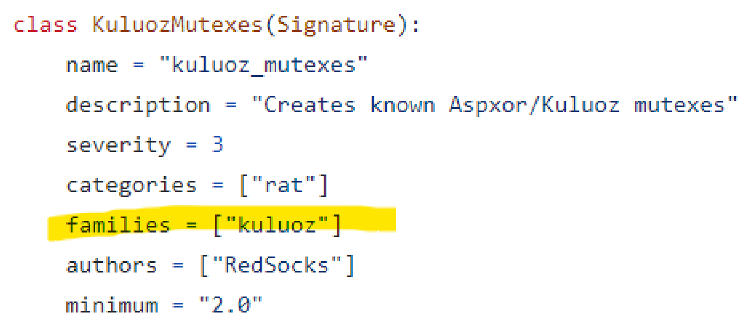
Upon further inspection, the suspicious name is identified as a mutex which is created by the malware called “2GVWNQJz1”. When googling, it is then verified that this is truly a malware as shown in the screenshots below confirmed by the online community [4]. The family of this malware is "kuluoz" [5] shown below in the screenshot.



As can be clearly seen, mutex 2gvwnqjz1 is strongly associated with malware. In fact, we have only seen it in malware.

Figure 3: Online user explain what "2GVWNQJz1" is

Source: [4]



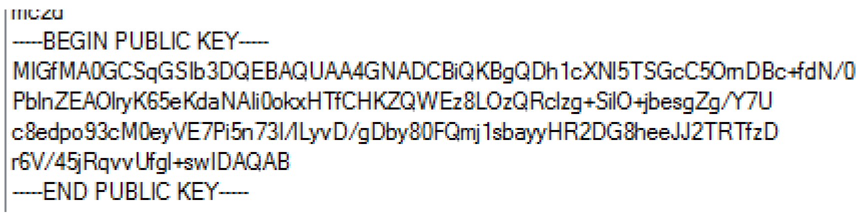
```
class KuluozMutexes(Signature):
    name = "kuluoz_mutexes"
    description = "Creates known Aspxor/Kuluoz mutexes"
    severity = 3
    categories = ["rat"]
    families = ["kuluoz"]
    authors = ["RedSocks"]
    minimum = "2.0"
```

Figure 4: Family name of malware

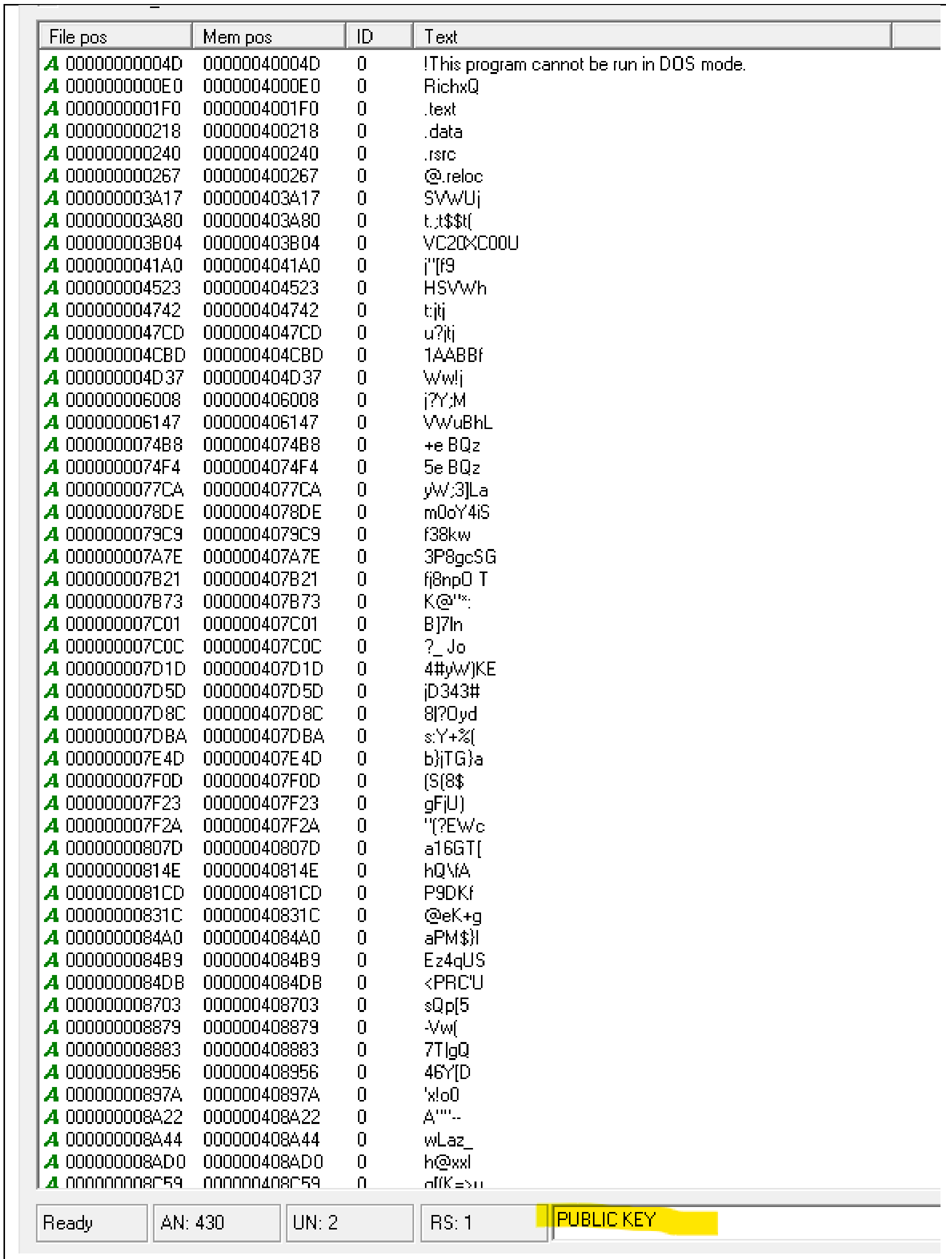
Source: [5]

Output#5: Dynamic Analysis

When running the sample dynamically, there was more text being generated such as public key generation while this was not the case for the static analysis as shown below. The malware is also trying to stay persistence by

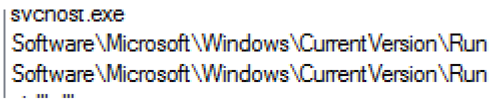


```
mc2d
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDh1cXNI5TSGcC5OmDBc+fdN/0
PbInZEAOlryK65eKdaNAli0okxHTfCHKZQWEz8LOzQRclzg+SilO+jbesgZg/Y7U
c8edpo93cM0eyVE7Pi5n73l/lLyvD/gDby80FQmj1sbayyHR2DG8heeJJ2TRTfzD
r6V/45jRqvvUfgl+swIDAQAB
-----END PUBLIC KEY-----
```



```
File pos          Mem pos           ID   Text
A 00000000004D    00000040004D      0    !This program cannot be run in DOS mode.
A 0000000000E0    0000004000E0      0    RichxQ
A 0000000001F0    0000004001F0      0    .text
A 000000000218    000000400218      0    .data
A 000000000240    000000400240      0    .rsrc
A 000000000267    000000400267      0    @.reloc
A 000000003A17    000000403A17      0    SVWUj
A 000000003A80    000000403A80      0    t.;t$$t(
A 000000003B04    000000403B04      0    VC20XC00U
A 0000000041A0    0000004041A0      0    j"[f9
A 000000004523    000000404523      0    HSVWh
A 000000004742    000000404742      0    t:jtj
A 0000000047CD    0000004047CD      0    u?jtj
A 000000004CBD    000000404CBD      0    1AABBf
A 000000004D37    000000404D37      0    Ww!j
A 000000006008    000000406008      0    j?Y;M
A 000000006147    000000406147      0    VWuBhL
A 0000000074B8    0000004074B8      0    +e BQz
A 0000000074F4    0000004074F4      0    5e BQz
A 0000000077CA    0000004077CA      0    yW;3]La
A 0000000078DE    0000004078DE      0    m0oY4iS
A 0000000079C9    0000004079C9      0    f38kw
A 000000007A7E    000000407A7E      0    3P8gcSG
A 000000007B21    000000407B21      0    fj8np0 T
A 000000007B73    000000407B73      0    K@"*:
A 000000007C01    000000407C01      0    B]7ln
A 000000007C0C    000000407C0C      0    ?_ Jo
A 000000007D1D    000000407D1D      0    4#yW)KE
A 000000007D5D    000000407D5D      0    jD343#
A 000000007D8C    000000407D8C      0    8|?0yd
A 000000007DBA    000000407DBA      0    s:Y+%(
A 000000007E4D    000000407E4D      0    b}jTG}a
A 000000007F0D    000000407F0D      0    [S(8$
A 000000007F23    000000407F23      0    gFjU)
A 000000007F2A    000000407F2A      0    "(?EWc
A 00000000807D    00000040807D      0    a16GT[
A 00000000814E    00000040814E      0    hQ\fA
A 0000000081CD    0000004081CD      0    P9DKf
A 00000000831C    00000040831C      0    @eK+g
A 0000000084A0    0000004084A0      0    aPM$}l
A 0000000084B9    0000004084B9      0    Ez4qUS
A 0000000084DB    0000004084DB      0    <PRC'U
A 000000008703    000000408703      0    sQp[5
A 000000008879    000000408879      0    -Vw(
A 000000008883    000000408883      0    7TlgQ
A 000000008956    000000408956      0    46Y[D
A 00000000897A    00000040897A      0    'x!o0
A 000000008A22    000000408A22      0    A""--
A 000000008A44    000000408A44      0    wLaz_
A 000000008AD0    000000408AD0      0    h@xxl
A 000000008C59    000000408C59      0    q[(K=>u
```

(I've tried searching for the term "PUBLIC KEY" in bintext but nothing was found.

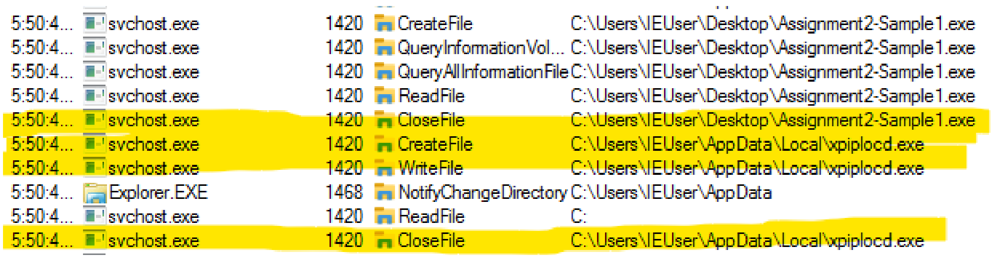


```
svchost.exe
Software\Microsoft\Windows\CurrentVersion\Run
Software\Microsoft\Windows\CurrentVersion\Run
```

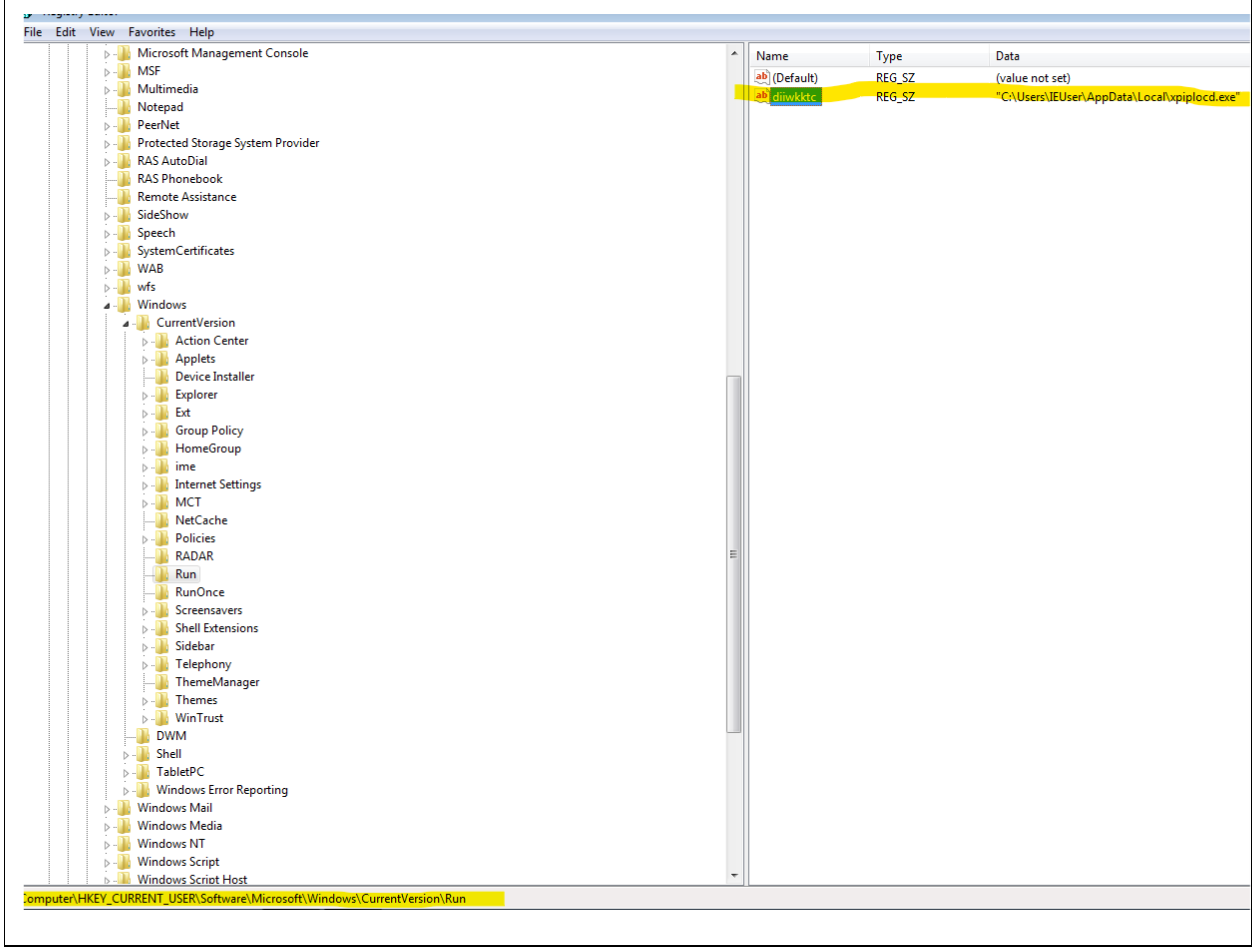
Output#6: ProcMon Analysis

During my analysis, shortly after "Assignment2-Sample1.exe" was ran, it created a random file in my user's directory with a random string as shown below in ProcMon. Moreover, when

I googled this location with a random generation of a file, I found that another analysis confirmed this as the “backdoor drops the following copies of itself into the affected system (C:\Users\{user name}\AppData) . . . [and] . . .adds the following registry entries to enable its automatic execution at every system startup” [6]. According to the author, this should also be created at “HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run” [6] which again, confirmed true as shown in the screenshot below.



```
5:50:4...  svchost.exe   1420  CreateFile              C:\Users\IEUser\Desktop\Assignment2-Sample1.exe
5:50:4...  svchost.exe   1420  QueryInformationVol...  C:\Users\IEUser\Desktop\Assignment2-Sample1.exe
5:50:4...  svchost.exe   1420  QueryAllInformationFile C:\Users\IEUser\Desktop\Assignment2-Sample1.exe
5:50:4...  svchost.exe   1420  ReadFile                C:\Users\IEUser\Desktop\Assignment2-Sample1.exe
5:50:4...  svchost.exe   1420  CloseFile               C:\Users\IEUser\Desktop\Assignment2-Sample1.exe
5:50:4...  svchost.exe   1420  CreateFile              C:\Users\IEUser\AppData\Local\xpiplocd.exe
5:50:4...  svchost.exe   1420  WriteFile               C:\Users\IEUser\AppData\Local\xpiplocd.exe
5:50:4...  Explorer.EXE  1468  NotifyChangeDirectory   C:\Users\IEUser\AppData
5:50:4...  svchost.exe   1420  ReadFile                C:
5:50:4...  svchost.exe   1420  CloseFile               C:\Users\IEUser\AppData\Local\xpiplocd.exe
```



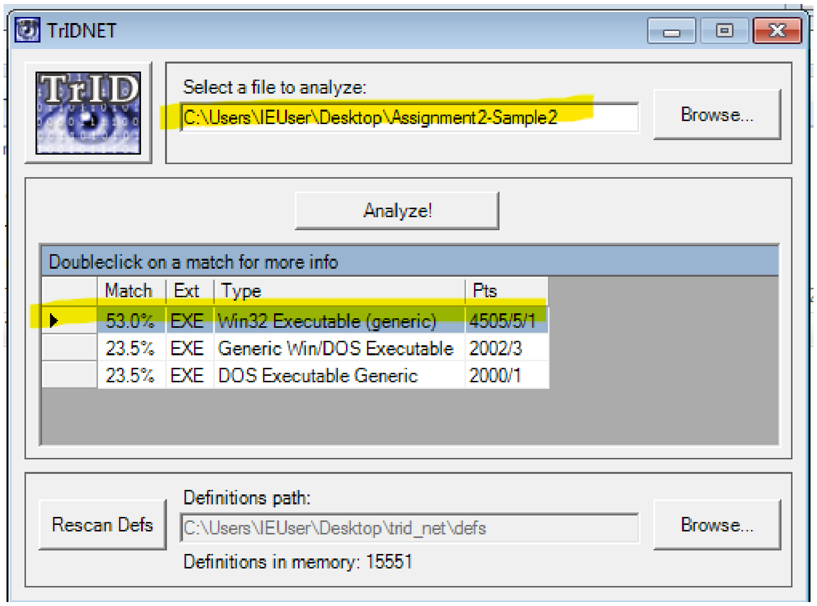
Name	Type	Data
(Default)	REG_SZ	(value not set)
diiwkktc	REG_SZ	"C:\Users\IEUser\AppData\Local\xpiplocd.exe"

Computer\HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run

Output Section for Sample #2

Output#1: Static Analysis

File Type/Format: To see the file type we perform a scan with TrID which states that it is a Win32 Executable as shown in the screenshot below. The file is also an executable as cross referenced with CFF Explorer.



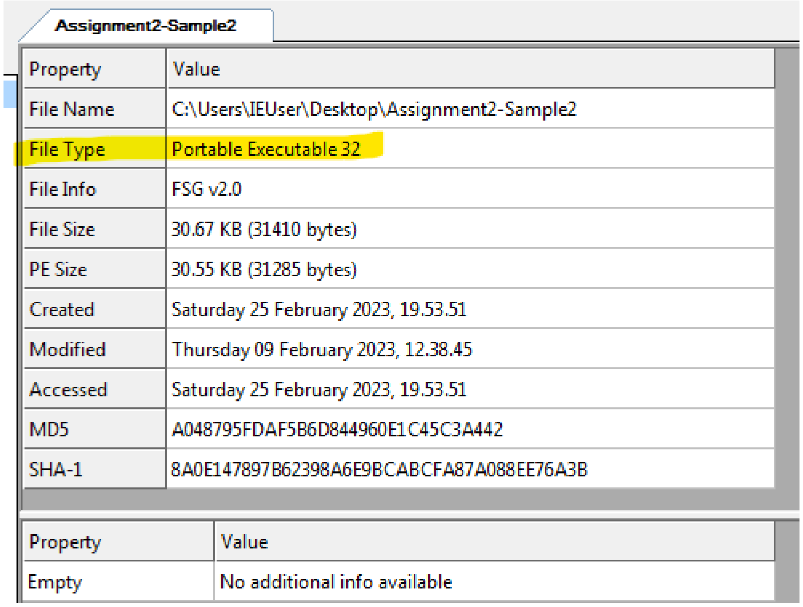
Select a file to analyze: C:\Users\IEUser\Desktop\Assignment2-Sample2

Doubleclick on a match for more info

Match	Ext	Type	Pts
53.0%	EXE	Win32 Executable (generic)	4505/5/1
23.5%	EXE	Generic Win/DOS Executable	2002/3
23.5%	EXE	DOS Executable Generic	2000/1

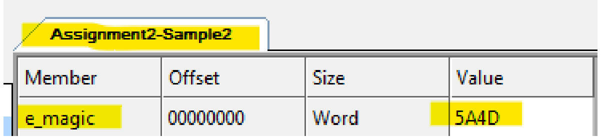
Definitions path: C:\Users\IEUser\Desktop\trid_net\defs

Definitions in memory: 15551



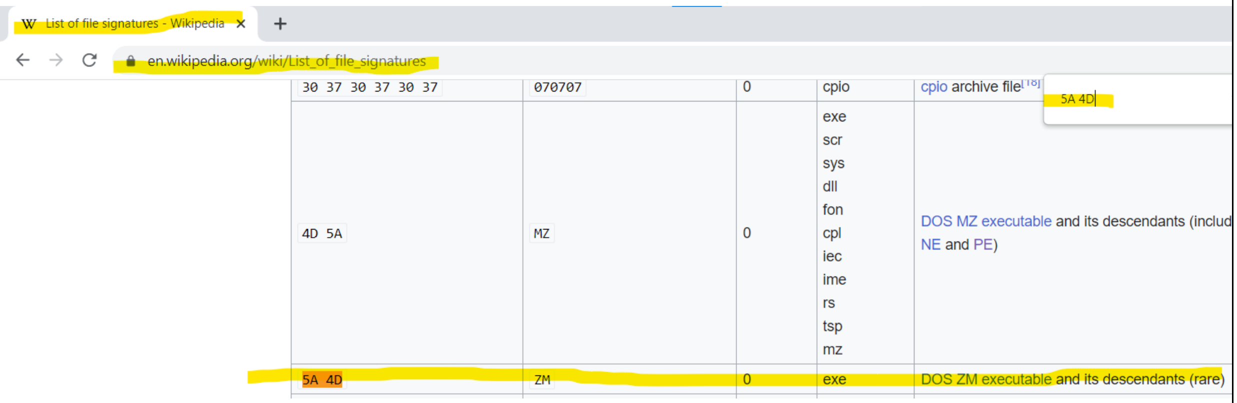
Property	Value
File Name	C:\Users\IEUser\Desktop\Assignment2-Sample2
File Type	Portable Executable 32
File Info	FSG v2.0
File Size	30.67 KB (31410 bytes)
PE Size	30.55 KB (31285 bytes)
Created	Saturday 25 February 2023, 19.53.51
Modified	Thursday 09 February 2023, 12.38.45
Accessed	Saturday 25 February 2023, 19.53.51
MD5	A048795FDAF5B6D844960E1C45C3A442
SHA-1	8A0E147897B62398A6E9BCABCFA87A088EE76A3B

Property	Value
Empty	No additional info available



Assignment2-Sample2

Member	Offset	Size	Value
e_magic	00000000	Word	5A4D



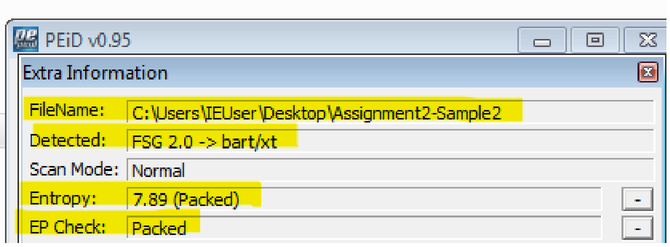
W List of file signatures - Wikipedia ×
+
en.wikipedia.org/wiki/List_of_file_signatures
5A 4D

30 37 30 37 30 37	070707	0	cpio	cpio archive file[18]
4D 5A	MZ	0	exe scr sys dll fon cpl iec ime rs tsp mz	DOS MZ executable and its descendants (includ NE and PE)
5A 4D	ZM	0	exe	DOS ZM executable and its descendants (rare)

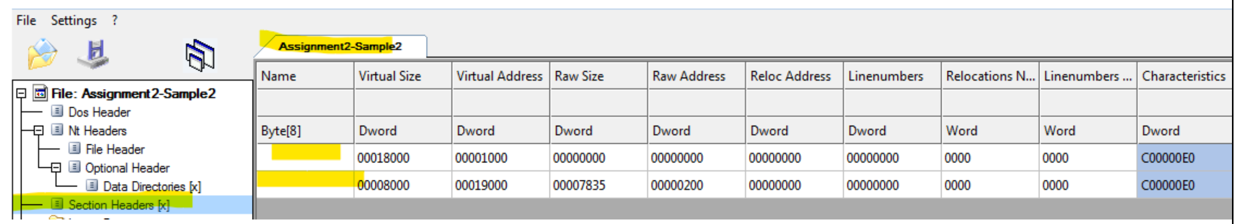
Figure 1: Wikipedia page; list of file signatures

Source: [1]

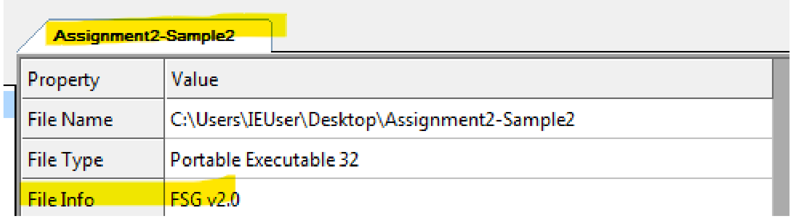
Packing: This file is packed as shown by PeiD and CFF Explorer as we cannot see any of the section headers and we can see the packers name is "FSG 2.0" as shown in the screenshot below. Finally, to drive home the point, we try packing it in CFF Explorer and we get an error.



FileName: C:\Users\IEUser\Desktop\Assignment2-Sample2
Detected: FSG 2.0 -> bart/xt
Scan Mode: Normal
Entropy: 7.89 (Packed)
EP Check: Packed

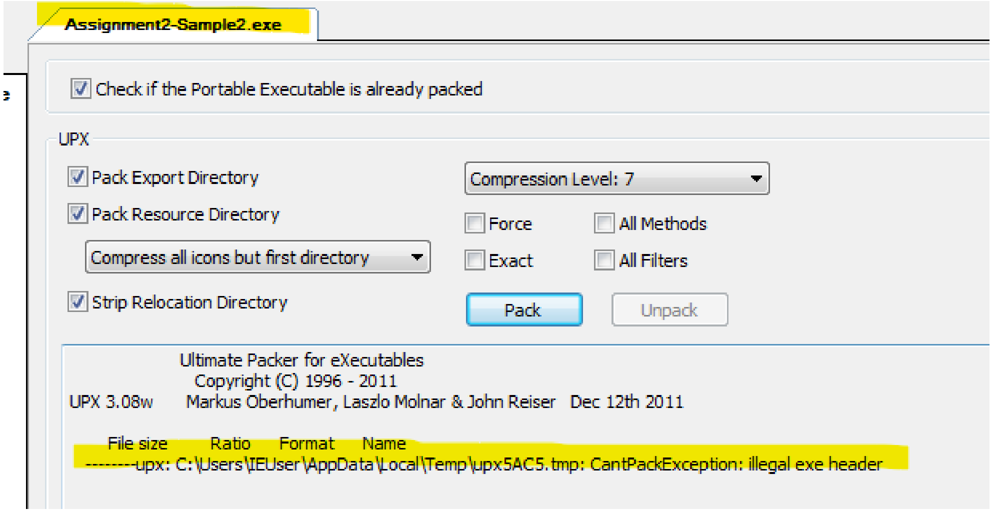


Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
	00018000	00001000	00000000	00000000	00000000	00000000	0000	0000	C00000E0
	00008000	00019000	00007835	00000200	00000000	00000000	0000	0000	C00000E0



Assignment2-Sample2

Property	Value
File Name	C:\Users\IEUser\Desktop\Assignment2-Sample2
File Type	Portable Executable 32
File Info	FSG v2.0



Assignment2-Sample2.exe

☑ Check if the Portable Executable is already packed

UPX

☑ Pack Export Directory

☑ Pack Resource Directory

Compress all icons but first directory

☑ Strip Relocation Directory

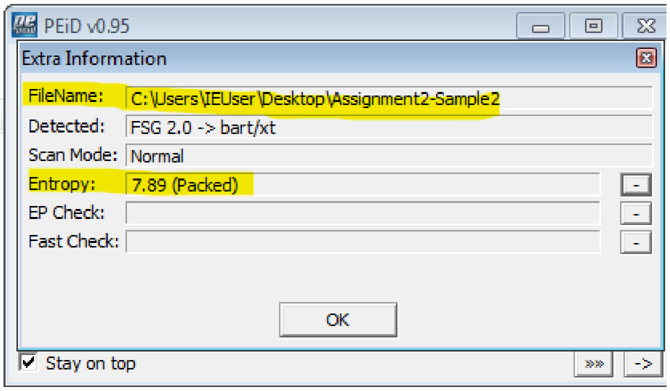
Compression Level: 7

□ Force □ All Methods

□ Exact □ All Filters

```
                 Ultimate Packer for eXecutables
                    Copyright (C) 1996 - 2011
UPX 3.08w    Markus Oberhumer, Laszlo Molnar & John Reiser   Dec 12th 2011

     File size        Ratio      Format      Name
   --------upx: C:\Users\IEUser\AppData\Local\Temp\upx5AC5.tmp: CantPackException: illegal exe header
```

Entropy: The entropy is calculated by running PEiD which is shown to be 7.89.



FileName: C:\Users\IEUser\Desktop\Assignment2-Sample2

Detected: FSG 2.0 -> bart/xt

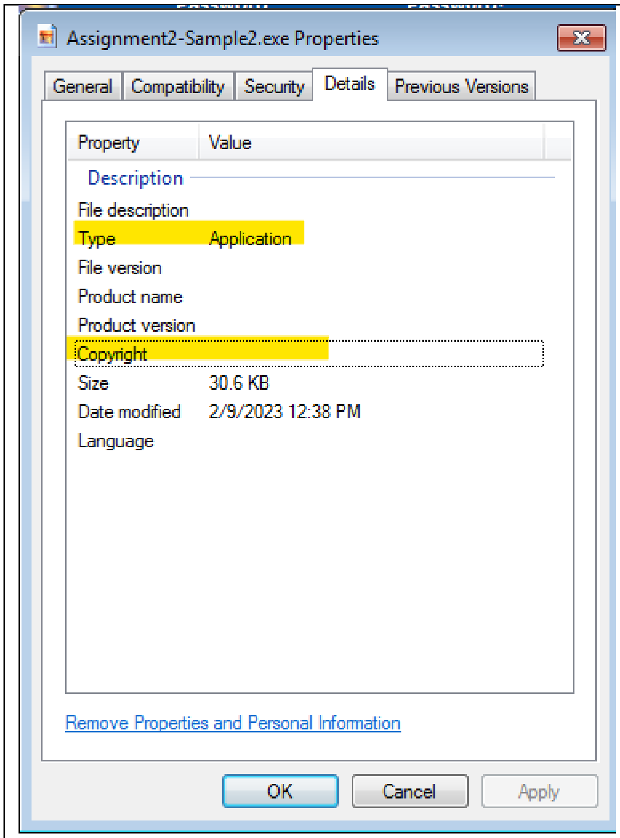
Scan Mode: Normal

Entropy: 7.89 (Packed)

EP Check:

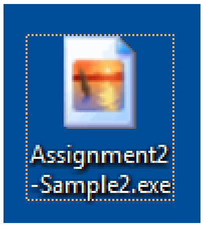
Fast Check:

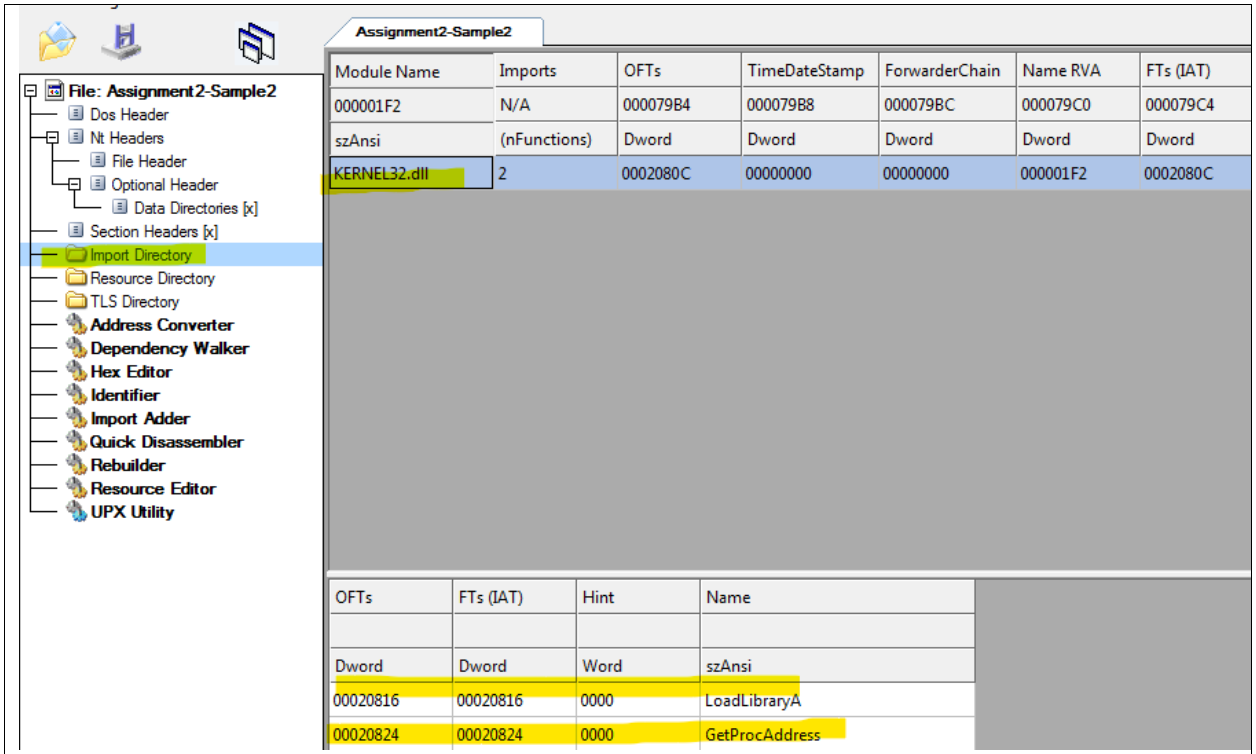
File Properties: In terms of file properties, there is no copyright on this file and it is tagged as an "application" once the .exe is added at the end. This file also has thumbnail faking as an image and also has 1 DLL as shown below.



Assignment2-Sample2.exe Properties

Property	Value
Description	
File description	
Type	Application
File version	
Product name	
Product version	
Copyright	
Size	30.6 KB
Date modified	2/9/2023 12:38 PM
Language	





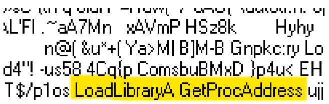
Assignment2-Sample2

- File: Assignment2-Sample2
  - Dos Header
  - Nt Headers
    - File Header
    - Optional Header
      - Data Directories [x]
  - Section Headers [x]
  - Import Directory
  - Resource Directory
  - TLS Directory
  - Address Converter
  - Dependency Walker
  - Hex Editor
  - Identifier
  - Import Adder
  - Quick Disassembler
  - Rebuilder
  - Resource Editor
  - UPX Utility

Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
000001F2	N/A	000079B4	000079B8	000079BC	000079C0	000079C4
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	2	0002080C	00000000	00000000	000001F2	0002080C

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
00020816	00020816	0000	LoadLibraryA
00020824	00020824	0000	GetProcAddress

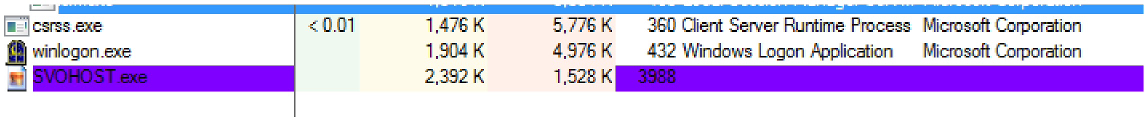
Static String Analysis: During static analysis, I've used BinText to analyze the strings however, because the malware was well packed, we aren't able to view much at all other than 2 suspicious API calls in which it calls "LoadLibraryA" and then "GetProcAddress".



```
\L'FI .~aA7Mn  xAVmP HSz8k      Hyhy
      n@[ &u*+[ Ya>M| B]M-B Gnpkc:ry Lo
d4"! -us58 4Cq{p ComsbuBMxD }p4u< EH
T$/p1os LoadLibraryA GetProcAddress uji
```

Output#2: System Observation Before & After

Before executing the sample, the file was there however, after running it as an administrator, we were unable to view the file as it just disappears. Under processor explorer, we can observer that a new exe process is still running as shown below as "SVOHOST.exe"



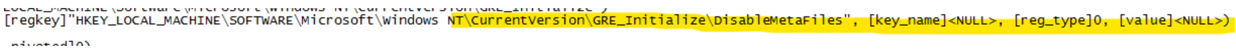
```
csrss.exe       < 0.01   1,476 K   5,776 K   360 Client Server Runtime Process   Microsoft Corporation
winlogon.exe             1,904 K   4,976 K   432 Windows Logon Application       Microsoft Corporation
SVOHOST.exe              2,392 K   1,528 K   3988
```

Output#3: Analyzing API Logs

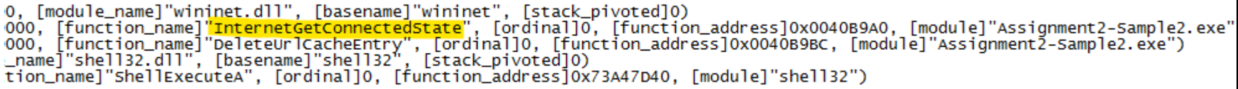
After running API Miner, we find that the malware tries to load its own libraries by invoking “LdrGetProcedureAddress” [12]. Then after, we see that the malware is playing around with the critical section by first invoking the “DeleteCriticalSection” which “releases all resources used by an unowned critical section object” [13] and then afterwards, release the ownership of that object by calling the “LeaveCriticalSection function” [14]. The malware finally then waits for the ownership to be granted once their library has been successfully loaded and then starts freeing up memory. After, the malware then obtains the location of “Imm32.dll” by using an undocumented function “LdrGetDllHandle”. “Imm32.dll” contains “a set of procedures and driver functions” [16] which is why probably the malware wants to mess around with it. Then it opens up the registry keys and makes adjustments to “LoadAppInit_DLLs” by allowing itself to load DLLs’s via AppInit [17]. The malware then checks for a remote session by checking the terminal server “TSAppCompat” then it enables RDP by changing the registry key “TSUserEnabled” [18]. Finally, it tries to contact C&C by checking the state of the internet by invoking the “wininet.dll” and “InternetGetConnectedState” [19], [9].

Output#4: Artifacts Analysis

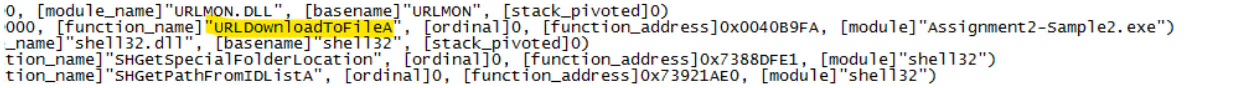
When examining the API calls, there was a particular spot where the malware removes the thumbnail. When disabling metafiles, it affects “all drawing operations required to create the picture” [7][8]. Hence, this is why the picture thumbnail disappeared as shown below in the screenshot. Another suspicious spot is the malware is trying to communicate with C&C and download additional resources. It checks if there’s a connected state by invoking the “InternetGetConnectedState function” [9] then establishes with C&C by calling the “URLDownloadToFile” function [10]. Armed with the knowledge in output #3 and #4, I searched google along with the hash of the file and I found the family name of this malware using the Microsoft malware naming convention is “Hupigon” [11].



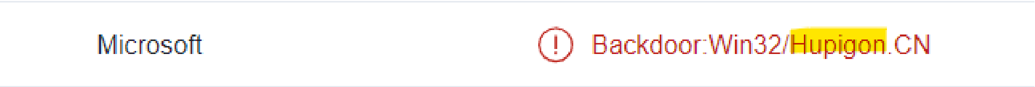
```
[regkey]"HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\GRE_Initialize\DisableMetaFiles", [key_name]<NULL>, [reg_type]0, [value]<NULL>)
```



```
0, [module_name]"wininet.dll", [basename]"wininet", [stack_pivoted]0)
000, [function_name]"InternetGetConnectedState", [ordinal]0, [function_address]0x0040B9A0, [module]"Assignment2-Sample2.exe"
000, [function_name]"DeleteUrlCacheEntry", [ordinal]0, [function_address]0x0040B9BC, [module]"Assignment2-Sample2.exe")
_name]"shell32.dll", [basename]"shell32", [stack_pivoted]0)
tion_name]"ShellExecuteA", [ordinal]0, [function_address]0x73A47D40, [module]"shell32")
```



```
0, [module_name]"URLMON.DLL", [basename]"URLMON", [stack_pivoted]0)
000, [function_name]"URLDownloadToFileA", [ordinal]0, [function_address]0x0040B9FA, [module]"Assignment2-Sample2.exe")
_name]"shell32.dll", [basename]"shell32", [stack_pivoted]0)
tion_name]"SHGetSpecialFolderLocation", [ordinal]0, [function_address]0x7388DFE1, [module]"shell32")
tion_name]"SHGetPathFromIDListA", [ordinal]0, [function_address]0x73921AE0, [module]"shell32")
```

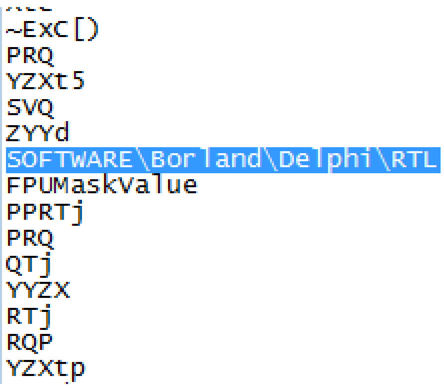


Microsoft Backdoor:Win32/Hupigon.CN

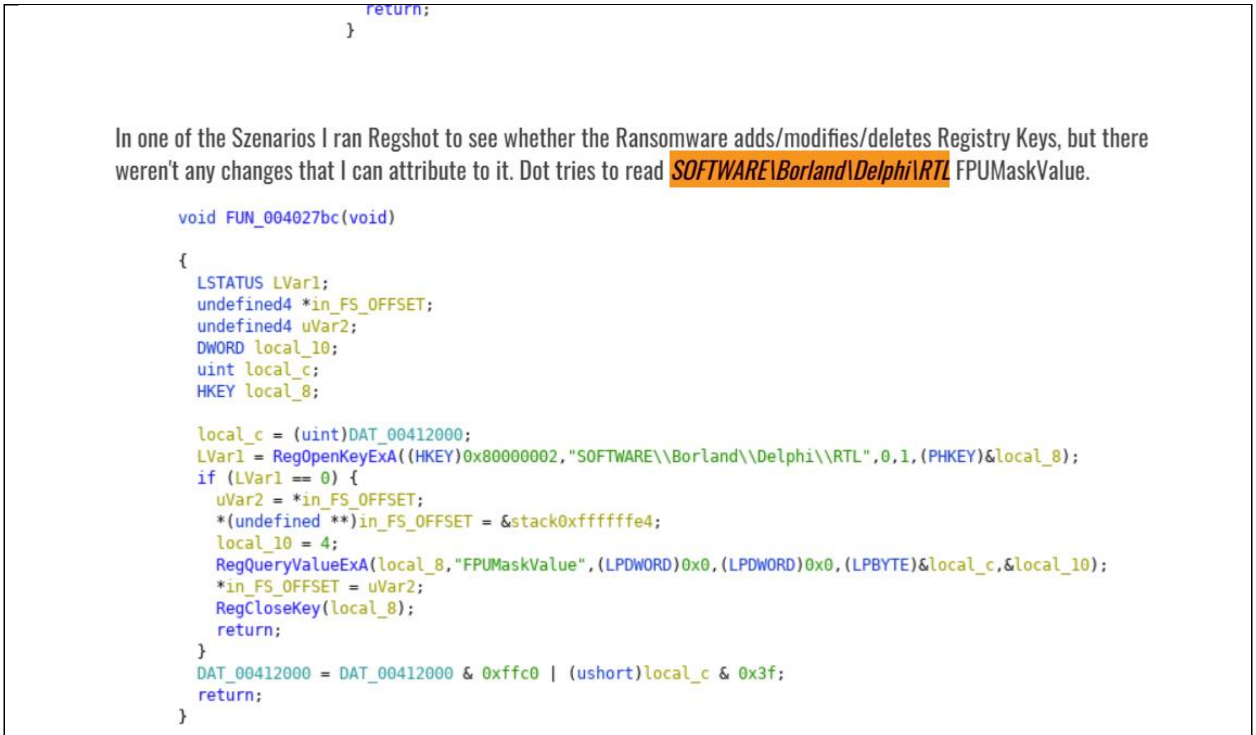
Source: [11]

Output#5: Dynamic Analysis

When observing the strings in process explorer we find a suspicious string “SOFTWARE\Borland\Delphi\RTL”. Doing some googling, it linked me to a virus that was also infecting other people hence giving us a further refinement that this is indeed a malware [20]. There were also suspicious API calls that was a sign of user mode code injection as shown below in the screenshot. Finally, another suspicious string was “config KVWSC start= disabled”. After googling this, it was confirmed again, it’s a malware as shown in the screenshot below.



```
~ExC[)
PRQ
YZXt5
SVQ
ZYYd
SOFTWARE\Borland\Delphi\RTL
FPUMaskValue
PPRTj
PRQ
QTj
YYZX
RTj
RQP
YZXtp
```

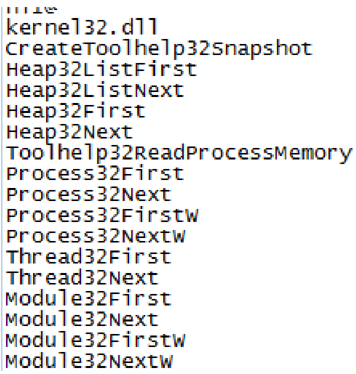


```
                              return;
                            }
```

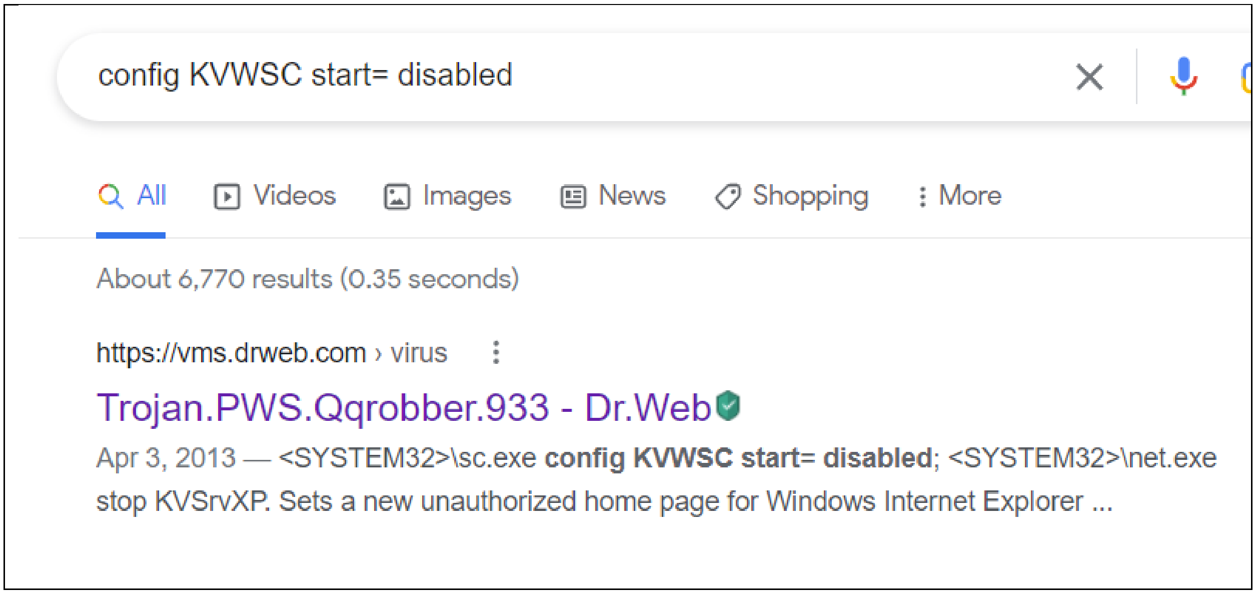
In one of the Szenarios I ran Regshot to see whether the Ransomware adds/modifies/deletes Registry Keys, but there weren't any changes that I can attribute to it. Dot tries to read *SOFTWARE\Borland\Delphi\RTL* FPUMaskValue.

```
void FUN_004027bc(void)

{
  LSTATUS LVar1;
  undefined4 *in_FS_OFFSET;
  undefined4 uVar2;
  DWORD local_10;
  uint local_c;
  HKEY local_8;

  local_c = (uint)DAT_00412000;
  LVar1 = RegOpenKeyExA((HKEY)0x80000002,"SOFTWARE\\Borland\\Delphi\\RTL",0,1,(PHKEY)&local_8);
  if (LVar1 == 0) {
    uVar2 = *in_FS_OFFSET;
    *(undefined **)in_FS_OFFSET = &stack0xffffffe4;
    local_10 = 4;
    RegQueryValueExA(local_8,"FPUMaskValue",(LPDWORD)0x0,(LPDWORD)0x0,(LPBYTE)&local_c,&local_10);
    *in_FS_OFFSET = uVar2;
    RegCloseKey(local_8);
    return;
  }
  DAT_00412000 = DAT_00412000 & 0xffc0 | (ushort)local_c & 0x3f;
  return;
}
```

Source: [20]



```
kernel32.dll
CreateToolhelp32Snapshot
Heap32ListFirst
Heap32ListNext
Heap32First
Heap32Next
Toolhelp32ReadProcessMemory
Process32First
Process32Next
Process32FirstW
Process32NextW
Thread32First
Thread32Next
Module32First
Module32Next
Module32FirstW
Module32NextW
```



About 6,770 results (0.35 seconds)

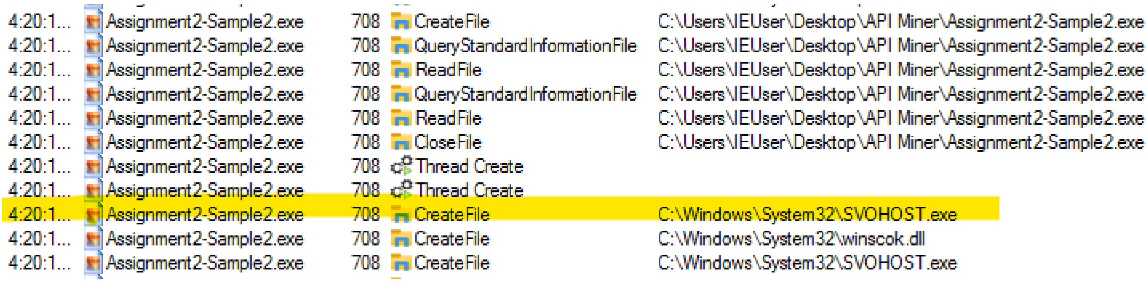
https://vms.drweb.com › virus

Trojan.PWS.Qqrobber.933 - Dr.Web

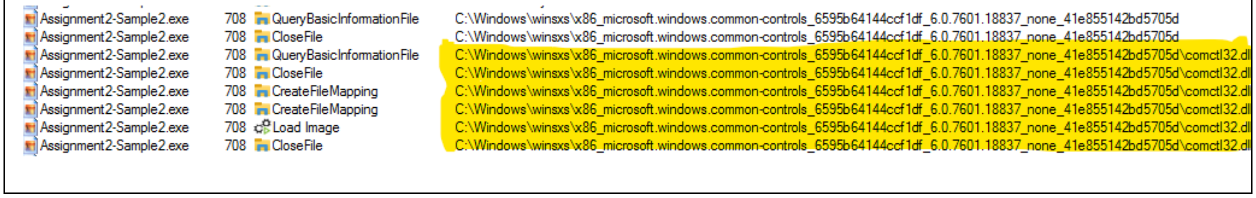
Apr 3, 2013 — <SYSTEM32>\sc.exe **config KVWSC start= disabled**; <SYSTEM32>\net.exe stop KVSrvXP. Sets a new unauthorized home page for Windows Internet Explorer ...

Output#6: ProcMon Analysis

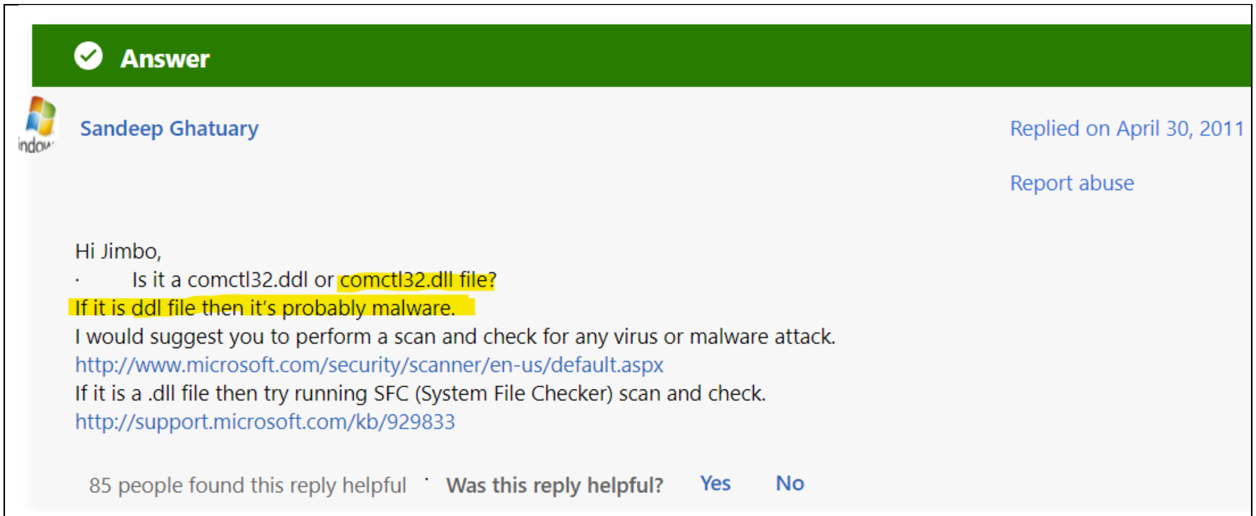
When we examine ProcMon, we find that a process called “SVOHOST.exe” was created after running the initial .exe file as shown below in the screenshot. Finally, we noticed that a folder with a dll being created as “C:\Windows\winsxs\x86_microsoft.windows.common-controls_6595b64144ccf1df_6.0.7601.18837_none_41e855142bd5705d\comctl32.dll” in the screenshot below. According to a user on the Microsoft answer board, it is definitively concluded that it’s a malware [21].



```
4:20:1...  Assignment2-Sample2.exe   708  CreateFile                     C:\Users\IEUser\Desktop\API Miner\Assignment2-Sample2.exe
4:20:1...  Assignment2-Sample2.exe   708  QueryStandardInformationFile   C:\Users\IEUser\Desktop\API Miner\Assignment2-Sample2.exe
4:20:1...  Assignment2-Sample2.exe   708  ReadFile                       C:\Users\IEUser\Desktop\API Miner\Assignment2-Sample2.exe
4:20:1...  Assignment2-Sample2.exe   708  QueryStandardInformationFile   C:\Users\IEUser\Desktop\API Miner\Assignment2-Sample2.exe
4:20:1...  Assignment2-Sample2.exe   708  ReadFile                       C:\Users\IEUser\Desktop\API Miner\Assignment2-Sample2.exe
4:20:1...  Assignment2-Sample2.exe   708  CloseFile                      C:\Users\IEUser\Desktop\API Miner\Assignment2-Sample2.exe
4:20:1...  Assignment2-Sample2.exe   708  Thread Create
4:20:1...  Assignment2-Sample2.exe   708  Thread Create
4:20:1...  Assignment2-Sample2.exe   708  CreateFile                     C:\Windows\System32\SVOHOST.exe
4:20:1...  Assignment2-Sample2.exe   708  CreateFile                     C:\Windows\System32\winscok.dll
4:20:1...  Assignment2-Sample2.exe   708  CreateFile                     C:\Windows\System32\SVOHOST.exe
```



```
Assignment2-Sample2.exe   708  QueryBasicInformationFile   C:\Windows\winsxs\x86_microsoft.windows.common-controls_6595b64144ccf1df_6.0.7601.18837_none_41e855142bd5705d
Assignment2-Sample2.exe   708  CloseFile                   C:\Windows\winsxs\x86_microsoft.windows.common-controls_6595b64144ccf1df_6.0.7601.18837_none_41e855142bd5705d
Assignment2-Sample2.exe   708  QueryBasicInformationFile   C:\Windows\winsxs\x86_microsoft.windows.common-controls_6595b64144ccf1df_6.0.7601.18837_none_41e855142bd5705d\comctl32.dl
Assignment2-Sample2.exe   708  CloseFile                   C:\Windows\winsxs\x86_microsoft.windows.common-controls_6595b64144ccf1df_6.0.7601.18837_none_41e855142bd5705d\comctl32.dl
Assignment2-Sample2.exe   708  CreateFileMapping           C:\Windows\winsxs\x86_microsoft.windows.common-controls_6595b64144ccf1df_6.0.7601.18837_none_41e855142bd5705d\comctl32.dl
Assignment2-Sample2.exe   708  CreateFileMapping           C:\Windows\winsxs\x86_microsoft.windows.common-controls_6595b64144ccf1df_6.0.7601.18837_none_41e855142bd5705d\comctl32.dl
Assignment2-Sample2.exe   708  Load Image                  C:\Windows\winsxs\x86_microsoft.windows.common-controls_6595b64144ccf1df_6.0.7601.18837_none_41e855142bd5705d\comctl32.dl
Assignment2-Sample2.exe   708  CloseFile                   C:\Windows\winsxs\x86_microsoft.windows.common-controls_6595b64144ccf1df_6.0.7601.18837_none_41e855142bd5705d\comctl32.dl
```


Answer
Sandeep Ghatuary
Replied on April 30, 2011
Report abuse

Hi Jimbo,

- Is it a comctl32.ddl or comctl32.dll file?

If it is ddl file then it's probably malware.
I would suggest you to perform a scan and check for any virus or malware attack.
http://www.microsoft.com/security/scanner/en-us/default.aspx
If it is a .dll file then try running SFC (System File Checker) scan and check.
http://support.microsoft.com/kb/929833

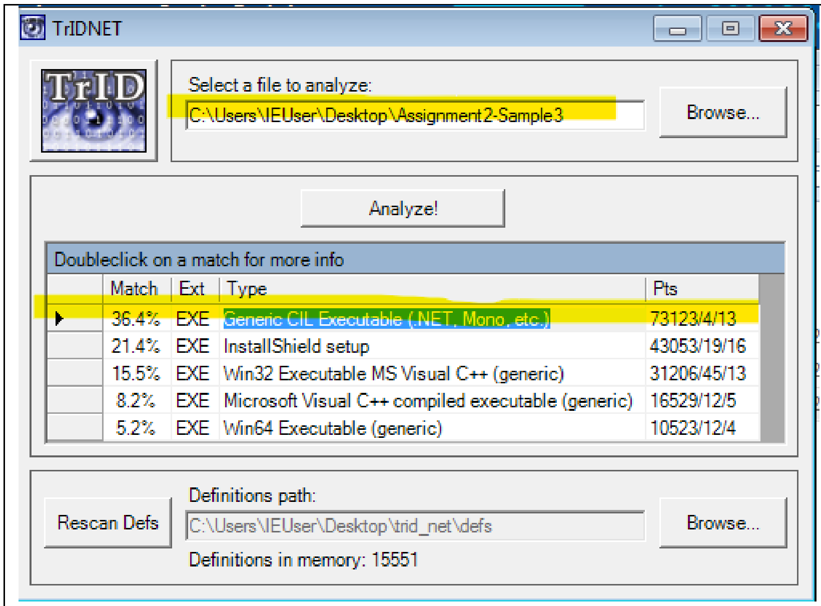
85 people found this reply helpful · Was this reply helpful? Yes No

Source: [21]

Output Section for Sample #3

Output#1: Static Analysis

File type/format: After running it with TrID, we find out that it has a type of "Generic CIL Executable (.NET, Mono, etc.)" and most likely written in assembly as shown by EXEinfo PE and it is a .exe file format as shown below.



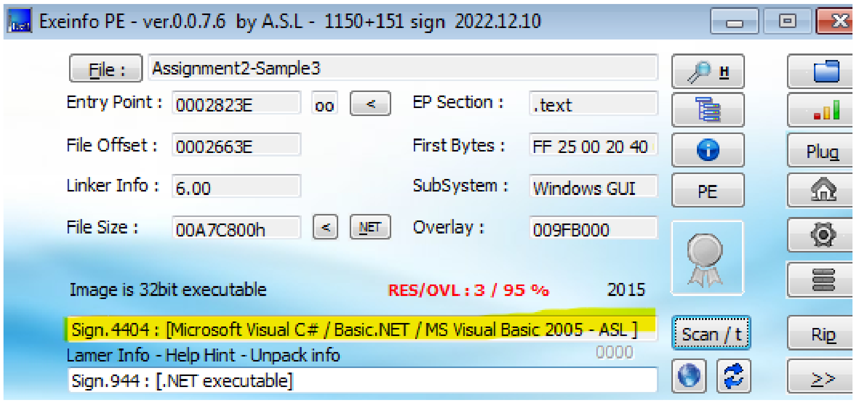
Select a file to analyze: C:\Users\IEUser\Desktop\Assignment2-Sample3

Doubleclick on a match for more info

Match	Ext	Type	Pts
36.4%	EXE	Generic CIL Executable (.NET, Mono, etc.)	73123/4/13
21.4%	EXE	InstallShield setup	43053/19/16
15.5%	EXE	Win32 Executable MS Visual C++ (generic)	31206/45/13
8.2%	EXE	Microsoft Visual C++ compiled executable (generic)	16529/12/5
5.2%	EXE	Win64 Executable (generic)	10523/12/4

Definitions path: C:\Users\IEUser\Desktop\trid_net\defs

Definitions in memory: 15551


Exeinfo PE - ver.0.0.7.6 by A.S.L - 1150+151 sign 2022.12.10
File :
PE
Plug
Scan / t
Rip
>>

File : Assignment2-Sample3

Entry Point : 0002823E oo <

EP Section : .text

File Offset : 0002663E

First Bytes : FF 25 00 20 40

Linker Info : 6.00

SubSystem : Windows GUI

File Size : 00A7C800h < NET

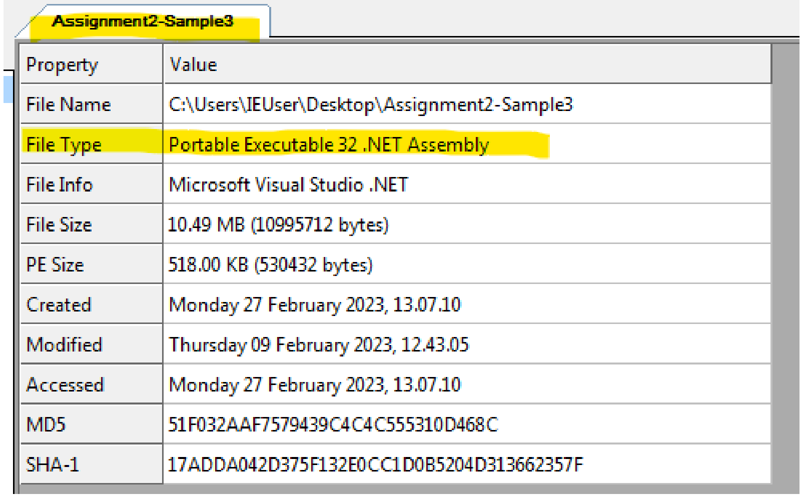
Overlay : 009FB000

Image is 32bit executable RES/OVL : 3 / 95 % 2015

Sign.4404 : [Microsoft Visual C# / Basic.NET / MS Visual Basic 2005 - ASL]

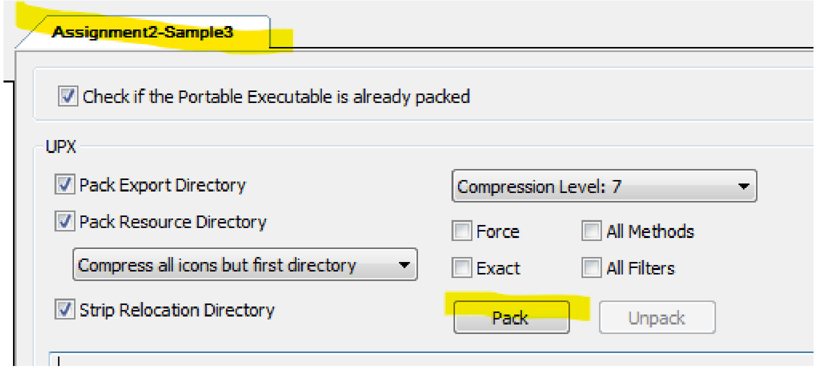
Lamer Info - Help Hint - Unpack info 0000

Sign.944 : [.NET executable]


Assignment2-Sample3

Property	Value
File Name	C:\Users\IEUser\Desktop\Assignment2-Sample3
File Type	Portable Executable 32 .NET Assembly
File Info	Microsoft Visual Studio .NET
File Size	10.49 MB (10995712 bytes)
PE Size	518.00 KB (530432 bytes)
Created	Monday 27 February 2023, 13.07.10
Modified	Thursday 09 February 2023, 12.43.05
Accessed	Monday 27 February 2023, 13.07.10
MD5	51F032AAF7579439C4C4C555310D468C
SHA-1	17ADDA042D375F132E0CC1D0B5204D313662357F

<u>Packing:</u> To check if it packed, we check it in CFF Explorer as we can clearly see that when we try to pack it, it gives us an error. We can also confirm this with PEiD which says "packed". Thus, this sample is indeed packed.



Assignment2-Sample3

☑ Check if the Portable Executable is already packed

UPX

☑ Pack Export Directory

☑ Pack Resource Directory

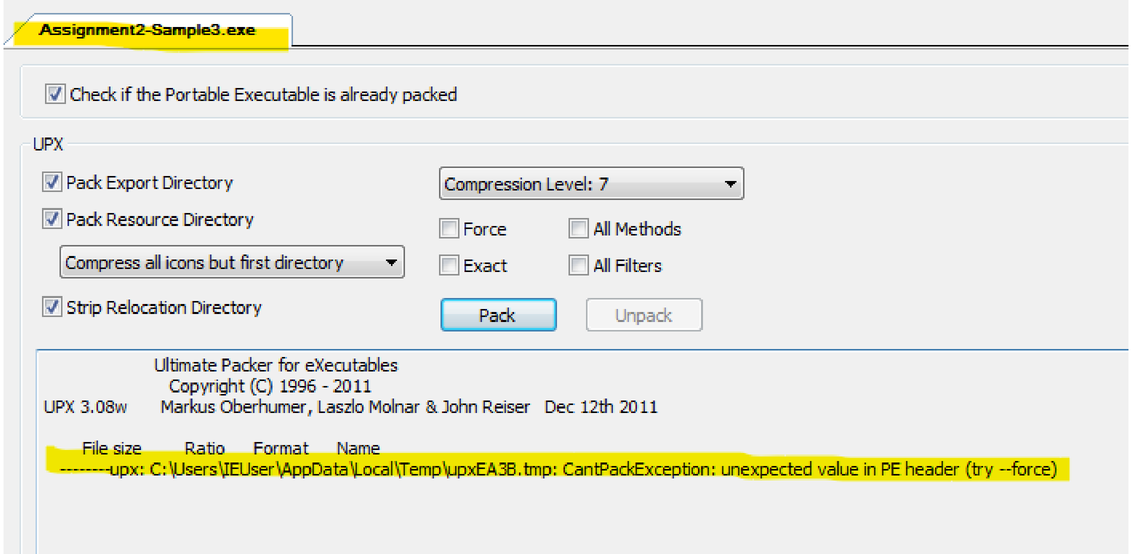
Compress all icons but first directory

☑ Strip Relocation Directory

Compression Level: 7

□ Force □ All Methods

□ Exact □ All Filters



Assignment2-Sample3.exe

☑ Check if the Portable Executable is already packed

UPX

☑ Pack Export Directory

☑ Pack Resource Directory

Compress all icons but first directory

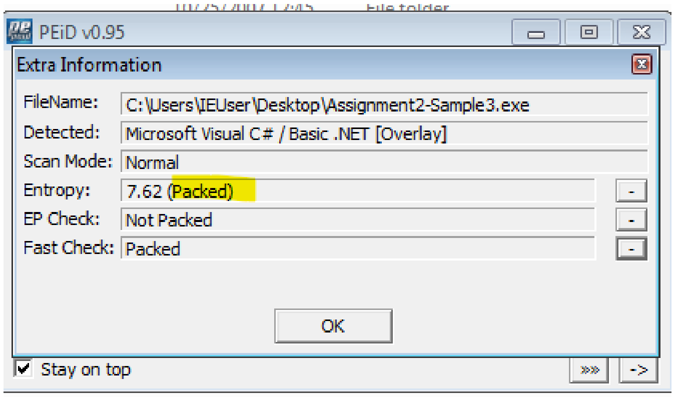
☑ Strip Relocation Directory

Compression Level: 7

□ Force □ All Methods

□ Exact □ All Filters

```
                    Ultimate Packer for eXecutables
                       Copyright (C) 1996 - 2011
UPX 3.08w       Markus Oberhumer, Laszlo Molnar & John Reiser   Dec 12th 2011

        File size         Ratio      Format      Name
   --------upx: C:\Users\IEUser\AppData\Local\Temp\upxEA3B.tmp: CantPackException: unexpected value in PE header (try --force)
```



FileName: C:\Users\IEUser\Desktop\Assignment2-Sample3.exe

Detected: Microsoft Visual C# / Basic .NET [Overlay]

Scan Mode: Normal

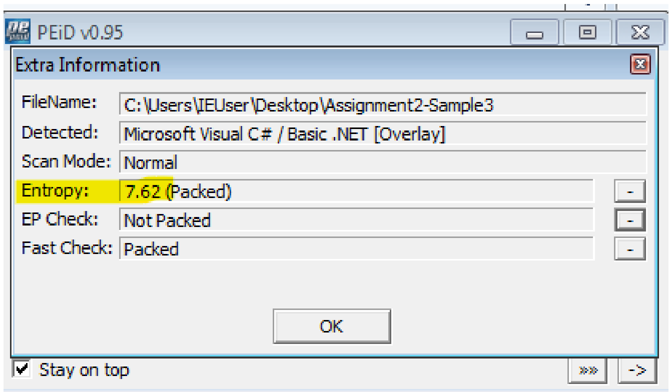
Entropy: 7.62 (Packed)

EP Check: Not Packed

Fast Check: Packed

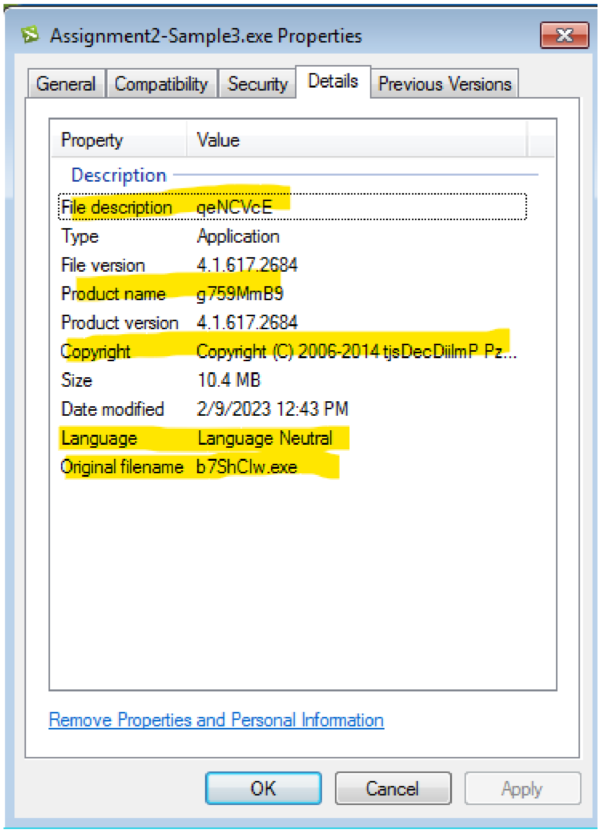
☑ Stay on top

Entropy: The entropy for this file is listed as 7.62 in accordance with PEiD.



FileName: C:\Users\IEUser\Desktop\Assignment2-Sample3
Detected: Microsoft Visual C# / Basic .NET [Overlay]
Scan Mode: Normal
Entropy: 7.62 (Packed)
EP Check: Not Packed
Fast Check: Packed

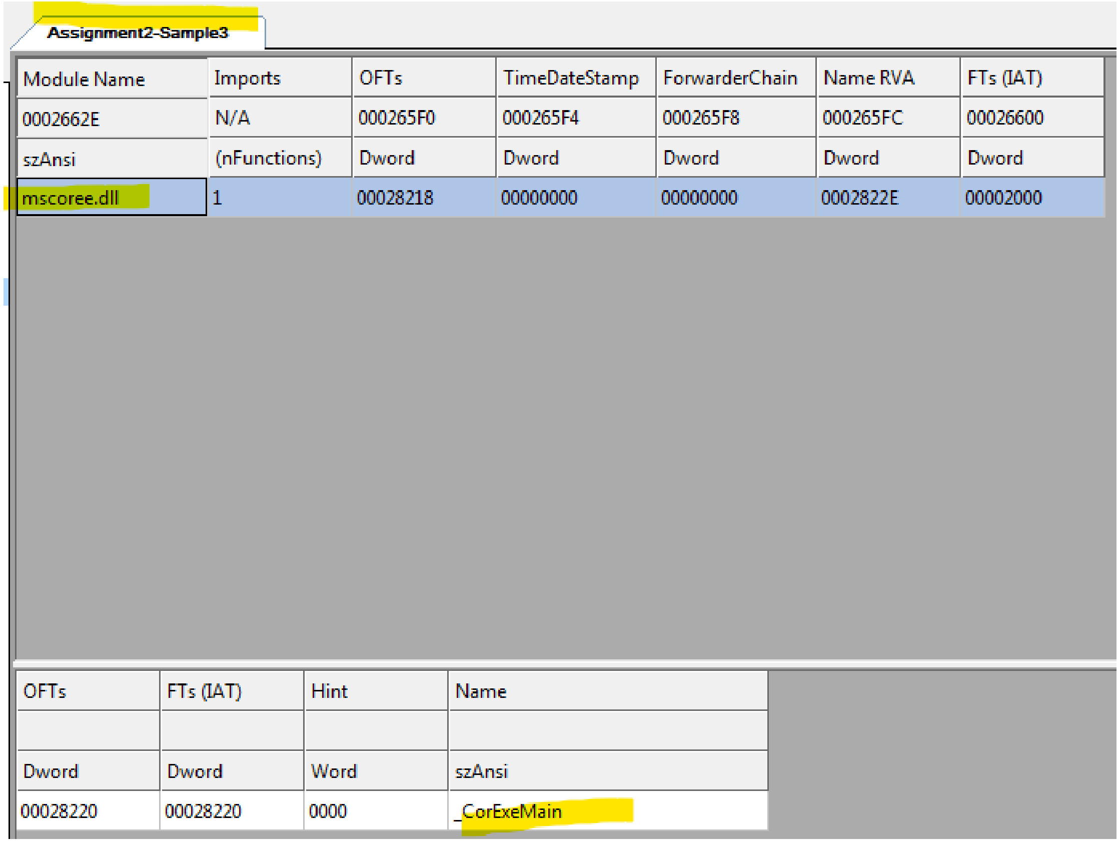
File Properties: The file properties can be found in the details section of the file. We can see that the malware author did put descriptions on it however, it's a random string text; even in the copyright section. It also uses thumbnail faking as shown below.



Assignment2-Sample3.exe Properties
General
Compatibility
Security
Details
Previous Versions

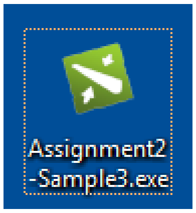
Property	Value
Description	
File description	qeNCVcE
Type	Application
File version	4.1.617.2684
Product name	g759MmB9
Product version	4.1.617.2684
Copyright	Copyright (C) 2006-2014 tjsDecDiilmP Pz...
Size	10.4 MB
Date modified	2/9/2023 12:43 PM
Language	Language Neutral
Original filename	b7ShClw.exe

Remove Properties and Personal Information
OK
Cancel
Apply

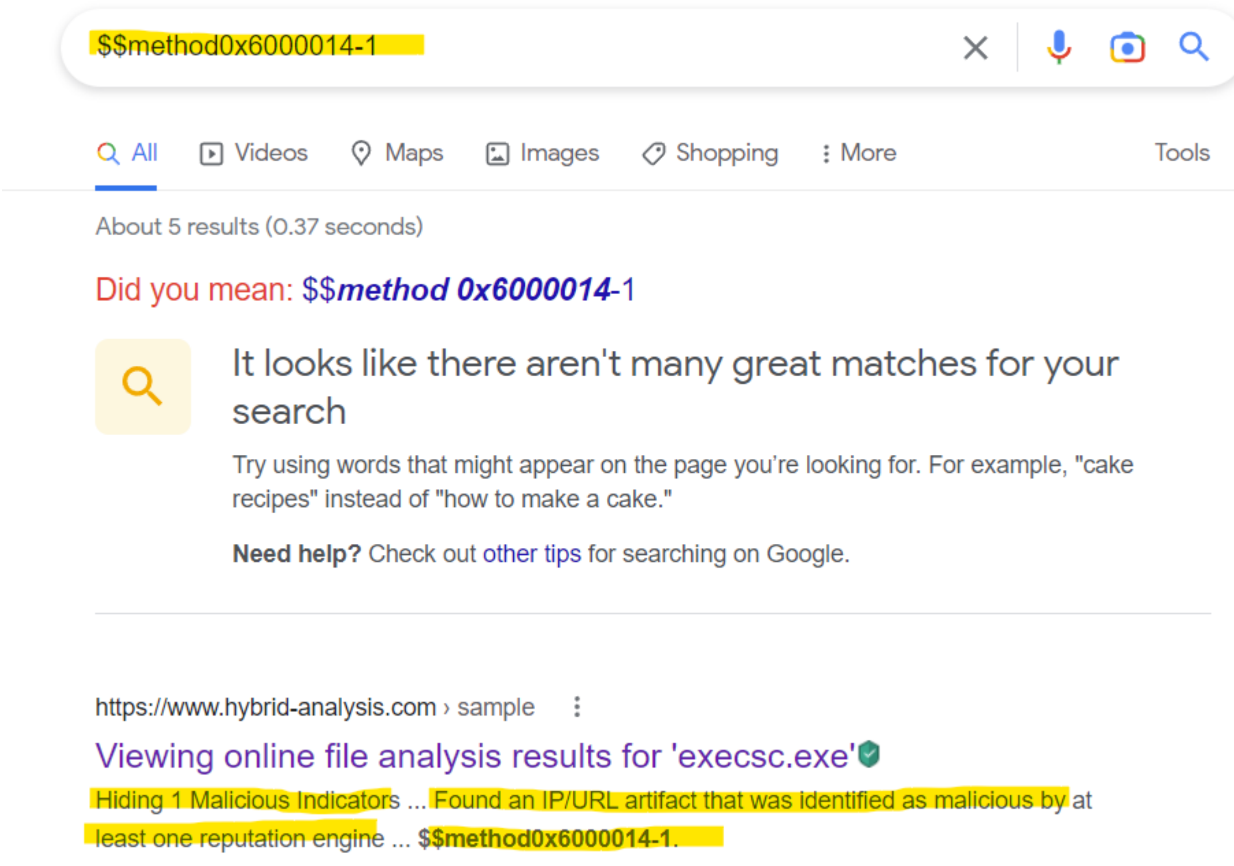


Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
0002662E	N/A	000265F0	000265F4	000265F8	000265FC	00026600
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
mscoree.dll	1	00028218	00000000	00000000	0002822E	00002000

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
00028220	00028220	0000	_CorExeMain



Static String Analysis: During the static string analysis, it was made clear that a string of “$$method0x6000014-1” was suspicious. I decided to google and found a website that stated it was indeed a suspicious string [22]. Moreover, the API calls within the strings are also very suspicious as listed below in screenshot.



$$method0x6000014-1

About 5 results (0.37 seconds)

Did you mean: $$*method 0x6000014*-1

It looks like there aren't many great matches for your search

Try using words that might appear on the page you're looking for. For example, "cake recipes" instead of "how to make a cake."

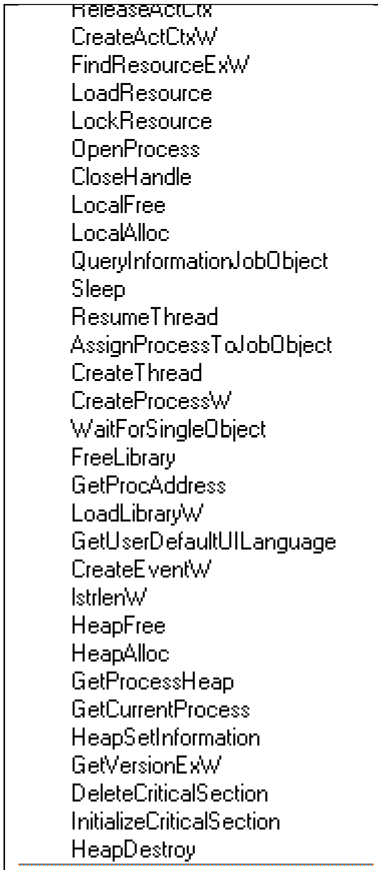
Need help? Check out other tips for searching on Google.

https://www.hybrid-analysis.com › sample

Viewing online file analysis results for 'execsc.exe'

Hiding 1 Malicious Indicators ... Found an IP/URL artifact that was identified as malicious by at least one reputation engine ... **$$method0x6000014-1**.

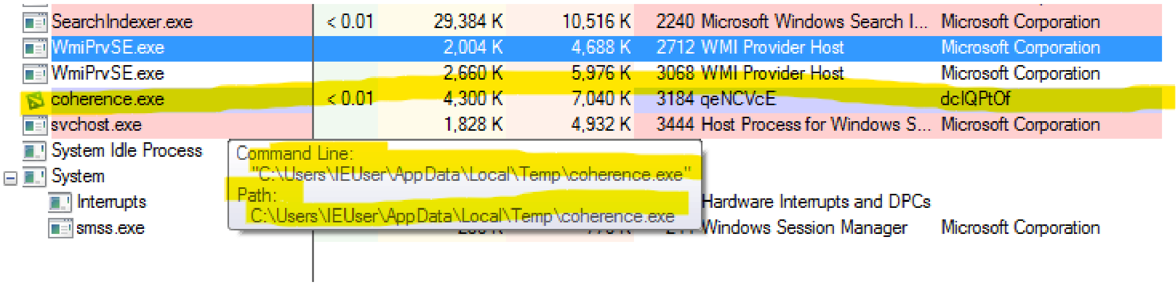
Source: [22]



```
ReleaseActCtx
CreateActCtxW
FindResourceExW
LoadResource
LockResource
OpenProcess
CloseHandle
LocalFree
LocalAlloc
QueryInformationJobObject
Sleep
ResumeThread
AssignProcessToJobObject
CreateThread
CreateProcessW
WaitForSingleObject
FreeLibrary
GetProcAddress
LoadLibraryW
GetUserDefaultUILanguage
CreateEventW
lstrlenW
HeapFree
HeapAlloc
GetProcessHeap
GetCurrentProcess
HeapSetInformation
GetVersionExW
DeleteCriticalSection
InitializeCriticalSection
HeapDestroy
```

Output#2: System Observation Before & After

Once I executed the malware, nothing popped up however, when examining it closely in process monitor, we see that it is running and created a file extension with the name “coherence.exe” as shown below.



```
SearchIndexer.exe        < 0.01   29,384 K   10,516 K   2240 Microsoft Windows Search I...   Microsoft Corporation
WmiPrvSE.exe                       2,004 K    4,688 K   2712 WMI Provider Host               Microsoft Corporation
WmiPrvSE.exe                       2,660 K    5,976 K   3068 WMI Provider Host               Microsoft Corporation
coherence.exe            < 0.01    4,300 K    7,040 K   3184 qeNCVcE                         dcIQPtOf
svchost.exe                        1,828 K    4,932 K   3444 Host Process for Windows S...   Microsoft Corporation
System Idle Process
System
  Interrupts                                                 Hardware Interrupts and DPCs
  smss.exe                                                   Windows Session Manager         Microsoft Corporation
```

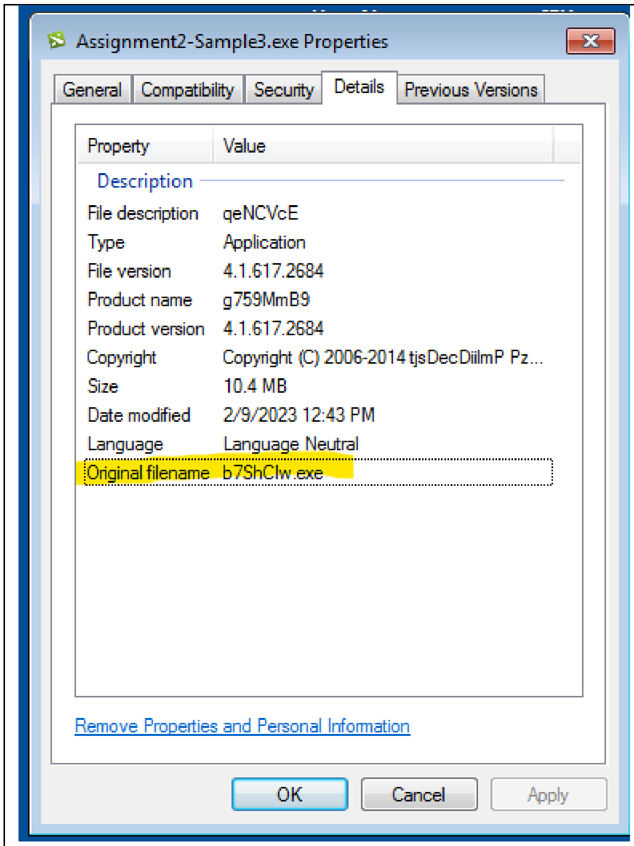
```
Command Line:
  "C:\Users\IEUser\AppData\Local\Temp\coherence.exe"
Path:
  C:\Users\IEUser\AppData\Local\Temp\coherence.exe
```

Output#3: Analyzing API Logs

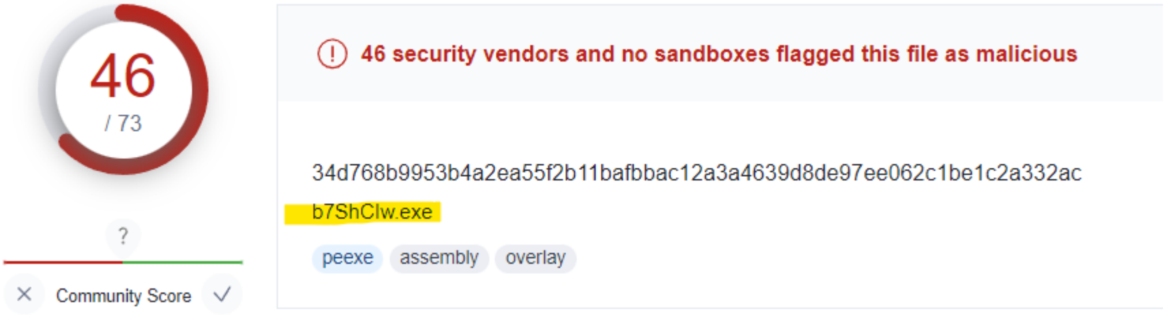
After running API Miner and trying to see the calls we see that initially the malware is trying to get the location of kernel32.dll by invoking “LdrGetDllHandle”. Then it tries to get the pointer to that file by calling “DecodePointer” as this function just returns the pointer that is decoded [24]. Finally, it tries to “EncodePointer” as this will encode any pointer that it returns [25].

Output#4: Artifacts Analysis

One main artifact can be found in the file properties, as it has its original name called “b7ShCIw.exe”. When you search it on google, it is definitively shown as a trojan malware as shown in the screenshot below. As aforementioned, an artifact called “coherence.exe” was created after running. However, upon further inspection and google searching, it seems to be one of the core artifacts that the malware creates. As you can see, it links to “C:\Users\IEUser\AppData\Local\Temp\coherence.exe” however, the application coherence.exe “should run from C:\Program Files\Parallels\Parallels Transporter Agent\ParallelsTransporterAgent.exe and not elsewhere” [26] which is not the case as shown in the screenshot below. To find the family name, we first hash the file, then running a search through TotalVirus, we find that the malware family is “Wacatac” with Microsoft’s naming convention [27].


Assignment2-Sample3.exe Properties
General
Compatibility
Security
Details
Previous Versions
Remove Properties and Personal Information
OK
Cancel
Apply

Property	Value
Description	
File description	qeNCVcE
Type	Application
File version	4.1.617.2684
Product name	g759MmB9
Product version	4.1.617.2684
Copyright	Copyright (C) 2006-2014 tjsDecDiilmP Pz...
Size	10.4 MB
Date modified	2/9/2023 12:43 PM
Language	Language Neutral
Original filename	b7ShClw.exe

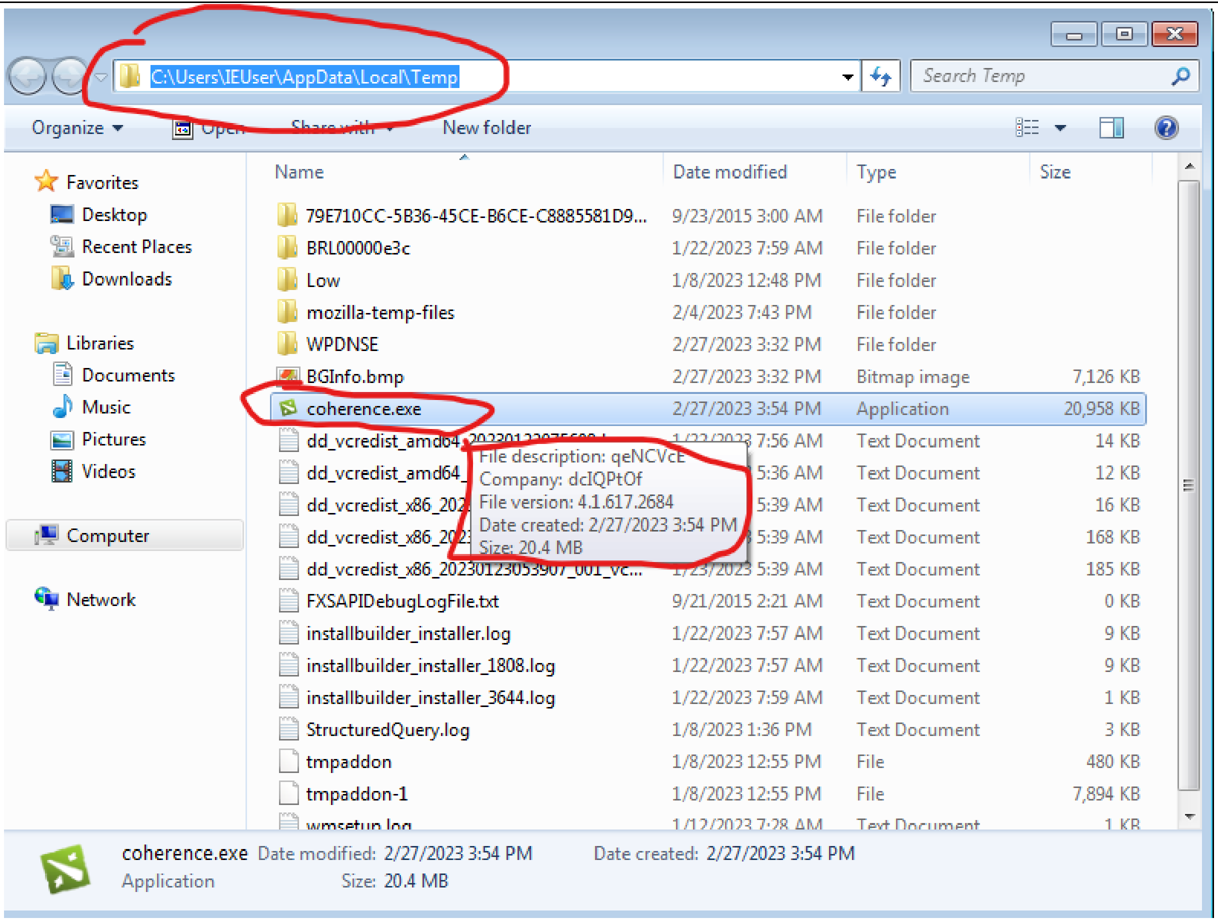


46 security vendors and no sandboxes flagged this file as malicious

34d768b9953b4a2ea55f2b11bafbbac12a3a4639d8de97ee062c1be1c2a332ac

b7ShClw.exe

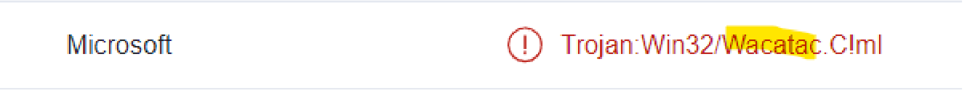
Source: [27]



Name	Date modified	Type	Size
79E710CC-5B36-45CE-B6CE-C8885581D9...	9/23/2015 3:00 AM	File folder	
BRL00000e3c	1/22/2023 7:59 AM	File folder	
Low	1/8/2023 12:48 PM	File folder	
mozilla-temp-files	2/4/2023 7:43 PM	File folder	
WPDNSE	2/27/2023 3:32 PM	File folder	
BGInfo.bmp	2/27/2023 3:32 PM	Bitmap image	7,126 KB
coherence.exe	2/27/2023 3:54 PM	Application	20,958 KB
dd_vcredist_amd64_	1/22/2023 7:56 AM	Text Document	14 KB
dd_vcredist_amd64_	5:36 AM	Text Document	12 KB
dd_vcredist_x86_202	5:39 AM	Text Document	16 KB
dd_vcredist_x86_202	5:39 AM	Text Document	168 KB
dd_vcredist_x86_20230123053907_001_vc...	1/23/2023 5:39 AM	Text Document	185 KB
FXSAPIDebugLogFile.txt	9/21/2015 2:21 AM	Text Document	0 KB
installbuilder_installer.log	1/22/2023 7:57 AM	Text Document	9 KB
installbuilder_installer_1808.log	1/22/2023 7:57 AM	Text Document	9 KB
installbuilder_installer_3644.log	1/22/2023 7:59 AM	Text Document	1 KB
StructuredQuery.log	1/8/2023 1:36 PM	Text Document	3 KB
tmpaddon	1/8/2023 12:55 PM	File	480 KB
tmpaddon-1	1/8/2023 12:55 PM	File	7,894 KB
wmsetup.log	1/12/2023 7:28 AM	Text Document	1 KB

File description: qeNCVcE
Company: dcIQPtOf
File version: 4.1.617.2684
Date created: 2/27/2023 3:54 PM
Size: 20.4 MB

coherence.exe Date modified: 2/27/2023 3:54 PM Date created: 2/27/2023 3:54 PM
Application Size: 20.4 MB

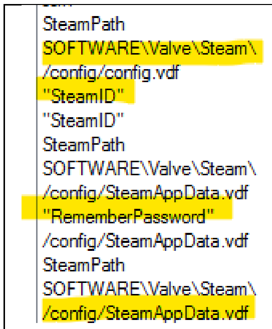


Microsoft ⓘ Trojan:Win32/Wacatac.C!ml

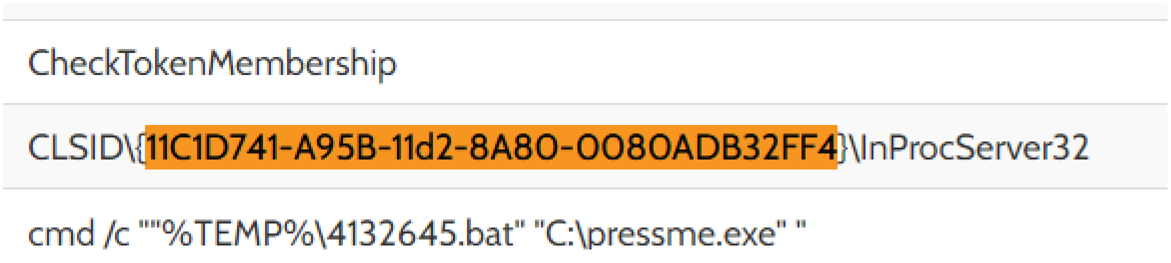
Source: [27]

Output#5: Dynamic Analysis

While the sample is being executed, we observe the strings in process monitor and we find out that it is trying to steal the login ID and password of a application called "Steam" as shown below. Another suspicious string was named "11C1D741-A95B-11d2-8A80-0080ADB32FF4". After some google searching, I found that it is some sort of password stealer trojan as shown below in the screenshot which also verifies the strings, I've found in process explorer [23].

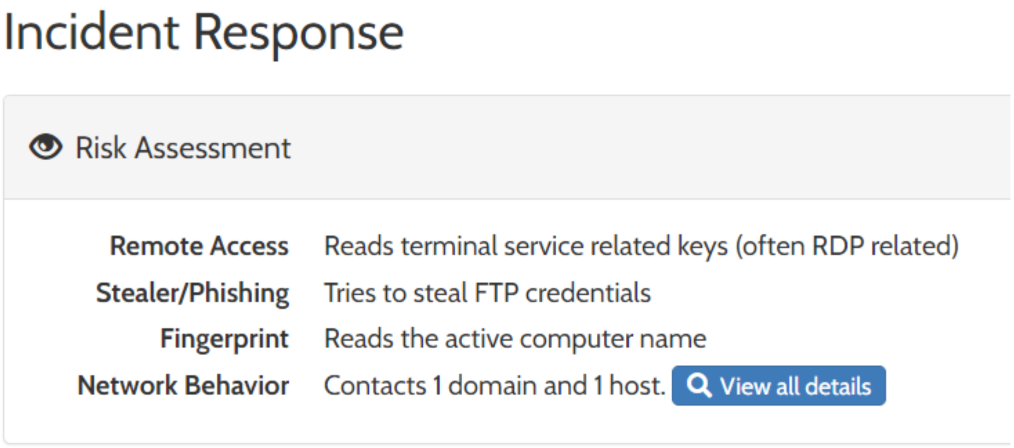


```
SteamPath
SOFTWARE\Valve\Steam\
/config/config.vdf
"SteamID"
"SteamID"
SteamPath
SOFTWARE\Valve\Steam\
/config/SteamAppData.vdf
"RememberPassword"
/config/SteamAppData.vdf
SteamPath
SOFTWARE\Valve\Steam\
/config/SteamAppData.vdf
```



```
CheckTokenMembership
CLSID\{11C1D741-A95B-11d2-8A80-0080ADB32FF4}\InProcServer32
cmd /c ""%TEMP%\4132645.bat" "C:\pressme.exe" "
```

Source: [23]



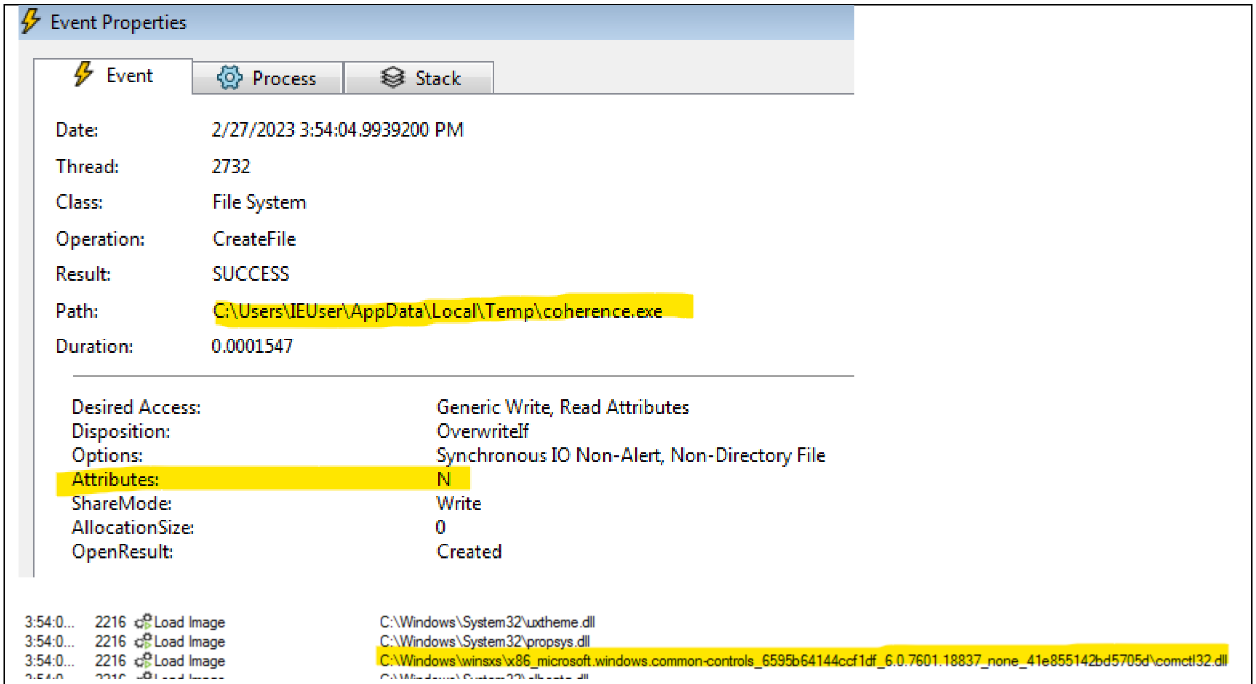
Incident Response

Risk Assessment

Remote Access	Reads terminal service related keys (often RDP related)
Stealer/Phishing	Tries to steal FTP credentials
Fingerprint	Reads the active computer name
Network Behavior	Contacts 1 domain and 1 host.

Output#6: ProcMon Analysis

Upon analysing the file in ProcMon, I noticed that when the file "coherence.exe" was created, it had marked the file attribute as "N". When I googled what this file attribute means, it states that the file was not index during creation. When you don't index a file, it means you cannot search for it in Windows search feature to look for files. Finally, the dead give away was the fact that it loaded an image with "comctl32.dll". As aforementioned, in [21], the author states that if it is a dll file, then it is defined as a malware in that specifc folder as highlighted in the screenshot below.



```
Date:          2/27/2023 3:54:04.9939200 PM
Thread:        2732
Class:         File System
Operation:     CreateFile
Result:        SUCCESS
Path:          C:\Users\IEUser\AppData\Local\Temp\coherence.exe
Duration:      0.0001547

  Desired Access:     Generic Write, Read Attributes
  Disposition:        OverwriteIf
  Options:            Synchronous IO Non-Alert, Non-Directory File
  Attributes:         N
  ShareMode:          Write
  AllocationSize:     0
  OpenResult:         Created
```

```
3:54:0...   2216   Load Image   C:\Windows\System32\uxtheme.dll
3:54:0...   2216   Load Image   C:\Windows\System32\propsys.dll
3:54:0...   2216   Load Image   C:\Windows\winsxs\x86_microsoft.windows.common-controls_6595b64144ccf1df_6.0.7601.18837_none_41e855142bd5705d\comctl32.dll
```

Source: [21]

References

[1] "List of file signatures," *Wikipedia*, 10-Feb-2023. [Online]. Available: https://en.wikipedia.org/wiki/List_of_file_signatures. [Accessed: 16-Feb-2023].

[2] A. Hassan, "Code injection, process hollowing, and API hooking (II)," *Slate*. [Online]. Available: https://slate.sheridancollege.ca/d2l/le/content/1058873/viewContent/13846520/View. [Accessed: 17-Feb-2023].

[3] Deland-Han, "Video doesn't play in Windows Media Player 11 - windows client," *Video doesn't play in Windows Media Player 11 - Windows Client | Microsoft Learn*, 2021. [Online]. Available: https://learn.microsoft.com/en-us/troubleshoot/windows-client/shell-experience/media-player-11-not-play-video. [Accessed: 17-Feb-2023].

[4] P. A. Networks, "Hunting the Mutex," *Unit 42*, 23-Mar-2022. [Online]. Available: https://unit42.paloaltonetworks.com/hunting-mutex/. [Accessed: 17-Feb-2023].

[5] Cuckoosandbox, "Community/rat_kuluoz.py at master · cuckoosandbox/community," *GitHub*, 12-Apr-2018. [Online]. Available: https://github.com/cuckoosandbox/community/blob/master/modules/signatures/windows/rat_kuluoz.py. [Accessed: 17-Feb-2023].

[6] M. J. Manahan, "BKDR_KULUOZ.Ed," *BKDR_KULUOZ.ED - Threat Encyclopedia*. [Online]. Available: https://www.trendmicro.com/vinfo/us/threat-encyclopedia/malware/bkdr_kuluoz.ed. [Accessed: 25-Feb-2023].

[7] Stevewhims, "About metafiles - win32 apps," *Win32 apps | Microsoft Learn*. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/gdi/about-metafiles. [Accessed: 26-Feb-2023].

[8] M. Microsoft, "A registry entry is available to turn off processing of metafiles," *Microsoft Support*. [Online]. Available: https://support.microsoft.com/en-us/topic/a-registry-entry-is-available-to-turn-off-processing-of-metafiles-d217cddb-3aef-2dda-1a01-903f180787d5. [Accessed: 26-Feb-2023].

[9] QuinnRadich, "Internetgetconnectedstate function (wininet.h) - win32 apps," *Win32 apps | Microsoft Learn*, 22-Aug-2022. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/api/wininet/nf-wininet-internetgetconnectedstate. [Accessed: 26-Feb-2023].

[10] "Urldownloadtofile function (windows)," *(Windows) | Microsoft Learn*, 13-Jul-2016. [Online]. Available: https://learn.microsoft.com/en-us/previous-

versions/windows/internet-explorer/ie-developer/platform-apis/ms775123(v=vs.85). [Accessed: 26-Feb-2023].

[11] “Sample-Any-Run-2.exe,” *Virustotal*, 16-Jul-2016. [Online]. Available: https://www.virustotal.com/gui/file/40c5ec744bcf776a3e885a2a88e49ff092155211e8e08ea9576fc98f781f6fc5. [Accessed: 26-Feb-2023].

[12] “R/malware - can the imported functions from windows dlls also be implemented by the malware writer him/herself?,” *reddit*, 17-Mar-2017. [Online]. Available: https://www.reddit.com/r/Malware/comments/5zz9e7/can_the_imported_functions_from_windows_dlls_also/. [Accessed: 26-Feb-2023].

[13] Karl-Bridge-Microsoft, “Deletecriticalsection function (synchapi.h) - win32 apps,” *Win32 apps | Microsoft Learn*, 23-Sep-2022. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/api/synchapi/nf-synchapi-deletecriticalsection. [Accessed: 26-Feb-2023].

[14] Karl-Bridge-Microsoft, “Leavecriticalsection function (synchapi.h) - win32 apps,” *Win32 apps | Microsoft Learn*, 23-Sep-2022. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/api/synchapi/nf-synchapi-leavecriticalsection. [Accessed: 26-Feb-2023].

[15] Karl-Bridge-Microsoft, “Entercriticalsection function (synchapi.h) - win32 apps,” *Win32 apps | Microsoft Learn*, 23-Sep-2022. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/api/synchapi/nf-synchapi-entercriticalsection. [Accessed: 26-Feb-2023].

[16] “Imm32.dll,” *WikiDll.com - Dll Filebase & Encyclopedia*. [Online]. Available: https://wikidll.com/other/imm32-dll. [Accessed: 26-Feb-2023].

[17] A. Administrator, “Persistence – appinit dlls,” *Penetration Testing Lab*, 30-Dec-2019. [Online]. Available: https://pentestlab.blog/2020/01/07/persistence-appinit-dlls/. [Accessed: 26-Feb-2023].

[18] Genlin, “Troubleshoot an RDP general error to a windows VM in Azure - virtual machines,” *Troubleshoot an RDP general error to a Windows VM in Azure - Virtual Machines | Microsoft Learn*, 10-Jul-2022. [Online]. Available: https://learn.microsoft.com/en-us/troubleshoot/azure/virtual-machines/troubleshoot-rdp-general-error. [Accessed: 26-Feb-2023].

[19] S. Stevewhims, “About wininet - win32 apps,” *Win32 apps | Microsoft Learn*, 20-Feb-2020. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/wininet/about-wininet. [Accessed: 26-Feb-2023].

[20] f0wL, “‘nice decorating. let me guess, satan?" - dot / MZP ransomware,” *~Dissecting Malware*, 02-Jan-2020. [Online]. Available: https://dissectingmalwa.re/nice-decorating-let-me-guess-satan-dot-mzp-ransomware.html. [Accessed: 26-Feb-2023].

[21] Jimbo_UK_101, “"‘c:\windows\winsxs\x86_microsoft.windows.common-controls.........\COMCTL32.ddl is either designed not to run on windows or it contains an error’",” *Redirecting*, 29-Apr-2011. [Online]. Available: https://answers.microsoft.com/en-us/windows/forum/all/cwindowswinsxsx86microsoftwindowscommon/ec0064a4-7d72-e011-8dfc-68b599b31bf5. [Accessed: 26-Feb-2023].

[22] Falcon Sandbox, “execsc.exe,” *Free Automated Malware Analysis Service - powered by Falcon Sandbox - viewing online file analysis results for 'execsc.exe'*. [Online]. Available: https://www.hybrid-analysis.com/sample/05462caca9f820d3fc99d589cb21ba4bb7b7db8d4ca63782428f9c7121e6abc0/5ac981137ca3e1684d361e74. [Accessed: 27-Feb-2023].

[23] Falcon Sandbox, “pressme.exe,” *Free Automated Malware Analysis Service - powered by Falcon Sandbox - viewing online file analysis results for 'pressme.exe'*. [Online]. Available: https://www.hybrid-analysis.com/sample/2ccacb57ebda602ccbb2710fe8195d485c11b4f45ce96ba5c379020ee78b5e10/5b23781e7ca3e16d66609367. [Accessed: 27-Feb-2023].

[24] “DecodePointer function (windows),” *(Windows) | Microsoft Learn*, 15-Mar-2018. [Online]. Available: https://learn.microsoft.com/en-us/previous-versions/bb432242(v=vs.85). [Accessed: 27-Feb-2023].

[25] “Encodepointer function (windows),” *(Windows) | Microsoft Learn*, 15-Mar-2018. [Online]. Available: https://learn.microsoft.com/en-us/previous-versions/bb432254(v=vs.85). [Accessed: 27-Feb-2023].

[26] P. Hart, “What is coherence.exe? is it safe or a virus? how to remove or fix it,” *Windows Bulletin Tutorials*, 17-Jul-2019. [Online]. Available: https://windowsbulletin.com/files/exe/parallels-software-international-inc/parallels-transporter-agent/coherence-exe. [Accessed: 27-Feb-2023].

[27] “b7ShCIw.exe,” *Virustotal*. [Online]. Available: https://www.virustotal.com/gui/file/34d768b9953b4a2ea55f2b11bafbbac12a3a4639d8de97ee062c1be1c2a332ac. [Accessed: 27-Feb-2023].