Practical2 Report

Name: Rachana Nitinkumar Gugale
UFID: 6353-2454
Email: rgugale@ufl.edu

Date: 20th March 2023
Class: CAP6137
Assignment: Practical 2

Executive Summary

The malware is a **ransomware**. Its behavior is very apparent when sample2.exe is run. Once the malware is run, it hijacks the whole system. It changes the desktop background to an image stating the system has been infected with **Lockbit 2.0 Ransomware** and the user's data is stolen and encrypted. The background also has details and links to get the stolen data back. It creates a file **LockBit_Ransomware.hta** on the desktop.

The malware recursively goes through each directory on the C drive, encrypting all non-executable files (.exe and .dll files are not encrypted) and renaming encrypted files by changing their extension to **.lockbit**. The encrypted files are easily recognizable as their icon is also changed. The malware skips over C:\Windows and C:\PerfLog directories and doesn't encrypt their contents. The program also mounts a Z drive but it doesn't populate it with any data.

In every directory where the malware encrypts files, it creates a new file **Restore-My-Files.txt** with details about the hackers and how to contact them to get the stolen data back. It also contains a unique "Decryption ID" for every run of the malware.

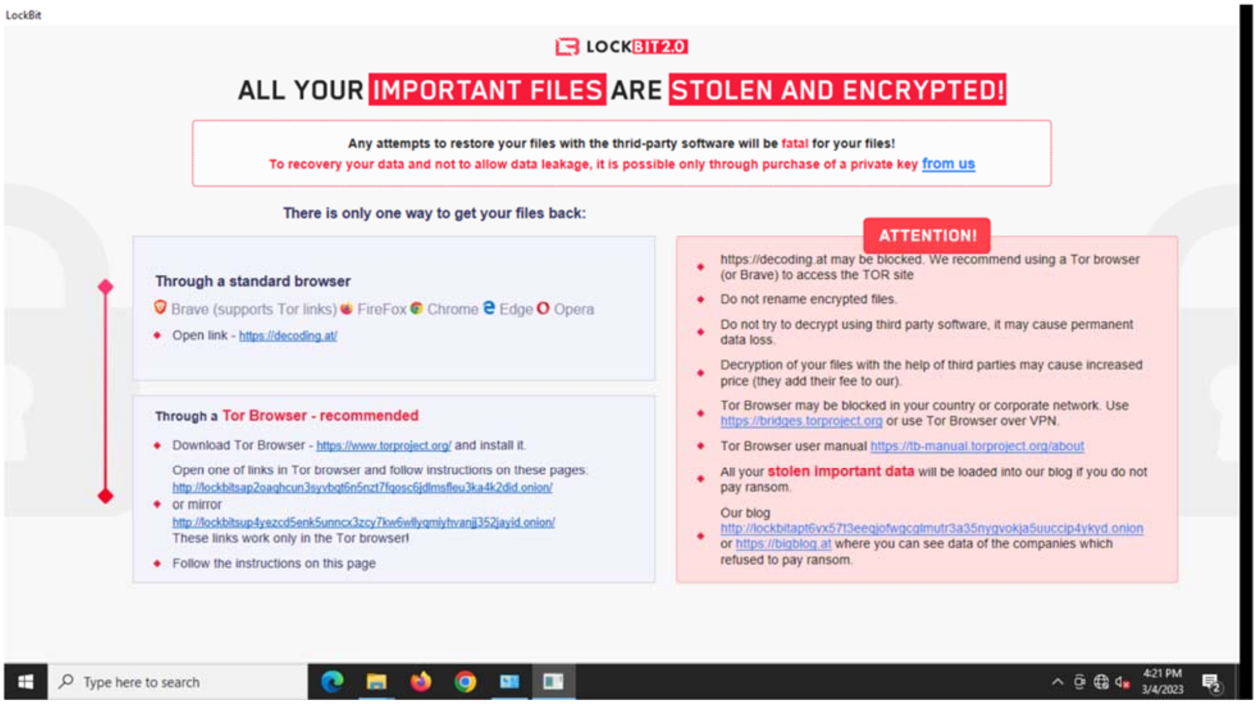
The program kills any running instances of Process Monitor and Process Explorer to thwart dynamic analysis. The malware uses a method to prevent debugging by inspecting the value located at the offset 0x68 of the PEB.

At the end of its execution, the Malware opens OneNote. The malware probably originated in a Slavic country as it does not infect computers whose default language is set to a Slavic language. This malware could be a part of a politically motivated cyberattack.

sample2.exe isn't persistent. Once it runs to completion and encrypts all the files available on the system, sample2.exe deletes itself. If new files are created after the malware finishes its execution, they are not encrypted. The encrypted files and wallpaper persist after logoffs and restarts. The malware adds LockBit_Ransomware.hta to **SOFTWARE\Microsoft\Windows\CurrentVersion\Run** to run it each time the machine is started.

Some host-based indicators: Presence of Restore-My-Files.txt, LockBit_Ransomware.hta, files with .lockbit extension, mutant \BaseNamedObjects\{A6E8DCE4-A6E8-7875-0E52-0E52-236D6DD023EE}

Network-based indicators: The malware doesn't perform any network activity on a normal Windows machine. It might use the Active Directory and LDAP on a Windows Server to find other machines in the network to infect them but this couldn't be observed and verified as I didn't have access to a Windows Server.



ALL YOUR IMPORTANT FILES ARE STOLEN AND ENCRYPTED!

Any attempts to restore your files with the thrid-party software will be fatal for your files!
To recovery your data and not to allow data leakage, it is possible only through purchase of a private key from us

There is only one way to get your files back:

Through a standard browser

Brave (supports Tor links) FireFox Chrome Edge Opera

- Open link - https://decoding.at/

Through a Tor Browser - recommended

- Download Tor Browser - https://www.torproject.org/ and install it.
- Open one of links in Tor browser and follow instructions on these pages:
  http://lockbitsap2oaghcun3syvbqt6n5nzt7fqosc6jdlmsfleu3ka4k2did.onion/
  or mirror
  http://lockbitsup4yezcd5enk5unncx3zcy7kw6wllyqmiyhvanjj352jayid.onion/
  These links work only in the Tor browser!
- Follow the instructions on this page

ATTENTION!

- https://decoding.at may be blocked. We recommend using a Tor browser (or Brave) to access the TOR site
- Do not rename encrypted files.
- Do not try to decrypt using third party software, it may cause permanent data loss.
- Decryption of your files with the help of third parties may cause increased price (they add their fee to our).
- Tor Browser may be blocked in your country or corporate network. Use https://bridges.torproject.org or use Tor Browser over VPN.
- Tor Browser user manual https://tb-manual.torproject.org/about
- All your stolen important data will be loaded into our blog if you do not pay ransom.
- Our blog
  http://lockbitapt6vx57t3eeqjofwgcglmutr3a35nygvokja5uuccip4ykyd.onion
  or https://bigblog.at where you can see data of the companies which refused to pay ransom.

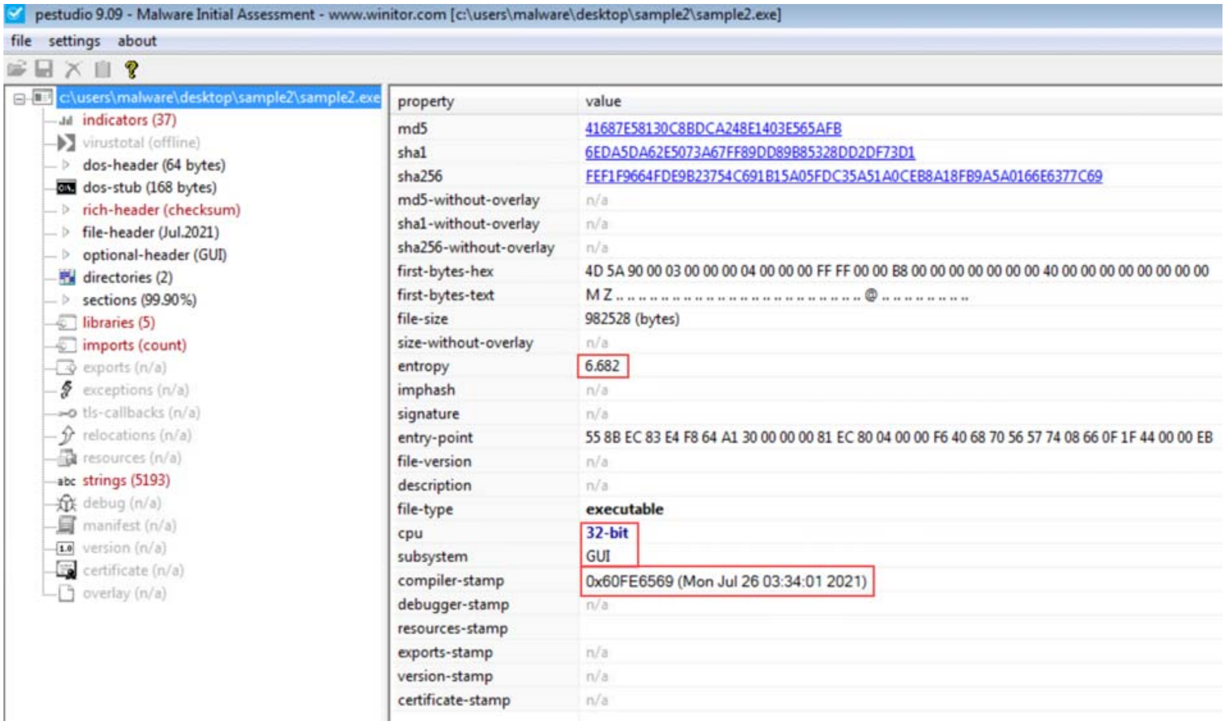
Figure 1 Wallpaper after running the malware

Static Analysis

PE Studio

PEStudio helped glean the following information:

1. **Compilation Date:** Jul 26, 2021 at 03:34:01
2. The malware is a 32-bit **Windows GUI** program as can be seen from the "subsystem" value.

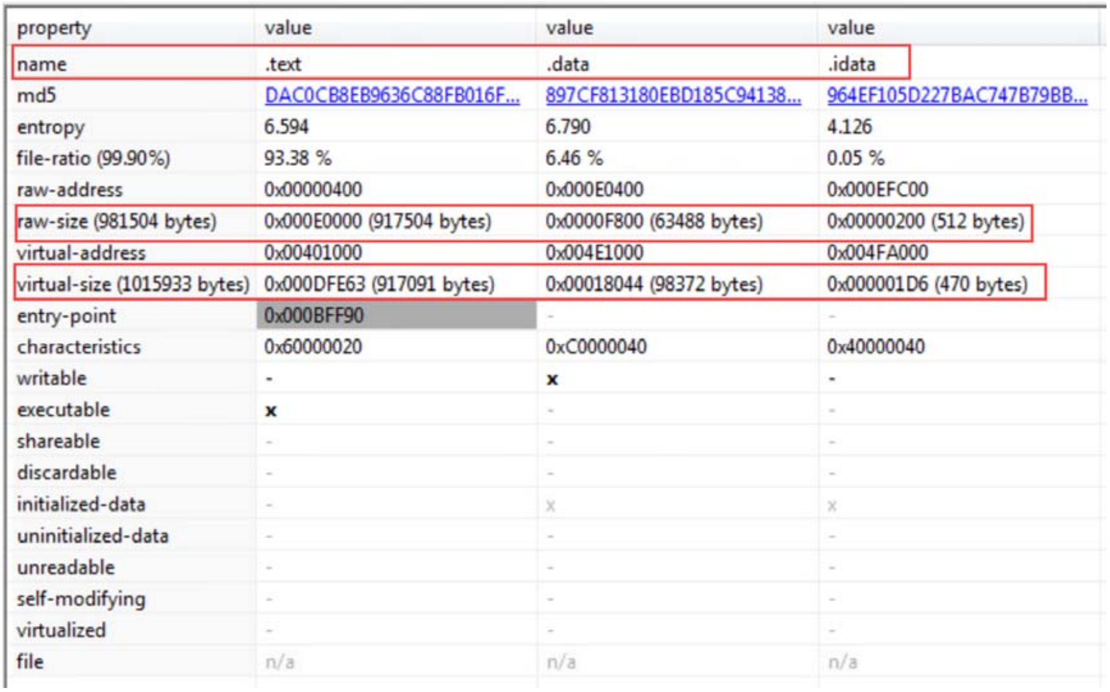

pestudio 9.09 - Malware Initial Assessment - www.winitor.com [c:\users\malware\desktop\sample2\sample2.exe]
file
settings
about

c:\users\malware\desktop\sample2\sample2.exe
- indicators (37)
- virustotal (offline)
- dos-header (64 bytes)
- dos-stub (168 bytes)
- rich-header (checksum)
- file-header (Jul.2021)
- optional-header (GUI)
- directories (2)
- sections (99.90%)
- libraries (5)
- imports (count)
- exports (n/a)
- exceptions (n/a)
- tls-callbacks (n/a)
- relocations (n/a)
- resources (n/a)
- strings (5193)
- debug (n/a)
- manifest (n/a)
- version (n/a)
- certificate (n/a)
- overlay (n/a)

property	value
md5	41687E58130C8BDCA248E1403E565AFB
sha1	6EDA5DA62E5073A67FF89DD89B85328DD2DF73D1
sha256	FEF1F9664FDE9B23754C691B15A05FDC35A51A0CEB8A18FB9A5A0166E6377C69
md5-without-overlay	n/a
sha1-without-overlay	n/a
sha256-without-overlay	n/a
first-bytes-hex	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00
first-bytes-text	M Z .. .. @ .. ..
file-size	982528 (bytes)
size-without-overlay	n/a
entropy	6.682
imphash	n/a
signature	n/a
entry-point	55 8B EC 83 E4 F8 64 A1 30 00 00 00 81 EC 80 04 00 00 F6 40 68 70 56 57 74 08 66 0F 1F 44 00 00 EB
file-version	n/a
description	n/a
file-type	**executable**
cpu	32-bit
subsystem	GUI
compiler-stamp	0x60FE6569 (Mon Jul 26 03:34:01 2021)
debugger-stamp	n/a
resources-stamp	
exports-stamp	n/a
version-stamp	n/a
certificate-stamp	n/a

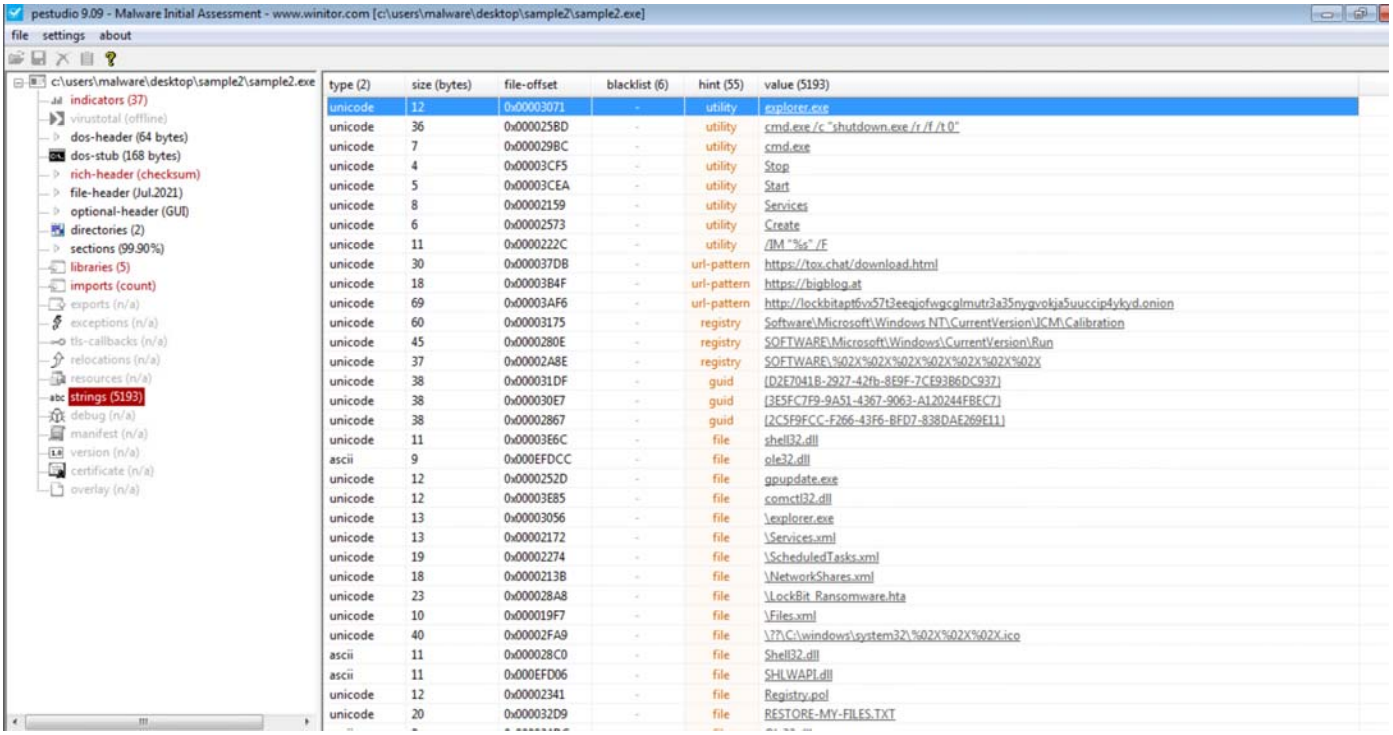
Figure 2 Compilation Date, Subsystem and Entropy

3. The program is **not packed**. The following indicators were used to reach this conclusion:
    a. **Entropy:** The entropy of a program or file is the measure of diversity in it. The entropy values range from 0 to 8 with 0 being no entropy (all bytes in the file are the same) to 8 (All bytes in the file are different). Normal executable programs have an entropy of around 6. Since our sample's entropy is 6.682, the entropy does not show any signs of the malware being packed.
    b. **Names of PE Sections:** The program has PE sections named .text, .data and .idata that are standard in all PE files. Since the names of the sections aren't obfuscated, the malware might not be packed.
    c. **Raw and Virtual Sizes of PE Sections:** If the raw size of a section is much smaller than its virtual size, it indicates that the malware might be doing some extraction and unpacking to populate new data. The raw and virtual sizes of .text and .idata section are not very far from each other. The .data section has a larger virtual size but this is standard for Windows executables. This provides further evidence that the malware isn't packed.
    d. **Strings:** The presence of many readable strings can provide some proof of the malware being unpacked. Since a large number of strings were found by PE Studio and most of them are readable, our program is not obfuscated.
    e. **Imports:** The program imports some functions from kernel32.dll, advapi32.dll, ole32.dll, activeeds.dll and shlwapi.dll. Though the presence of these imported functions is insufficient to conclude that the malware is not packed, combined with the other indicators, it provides us with sufficient evidence to establish that the malware is obfuscated.



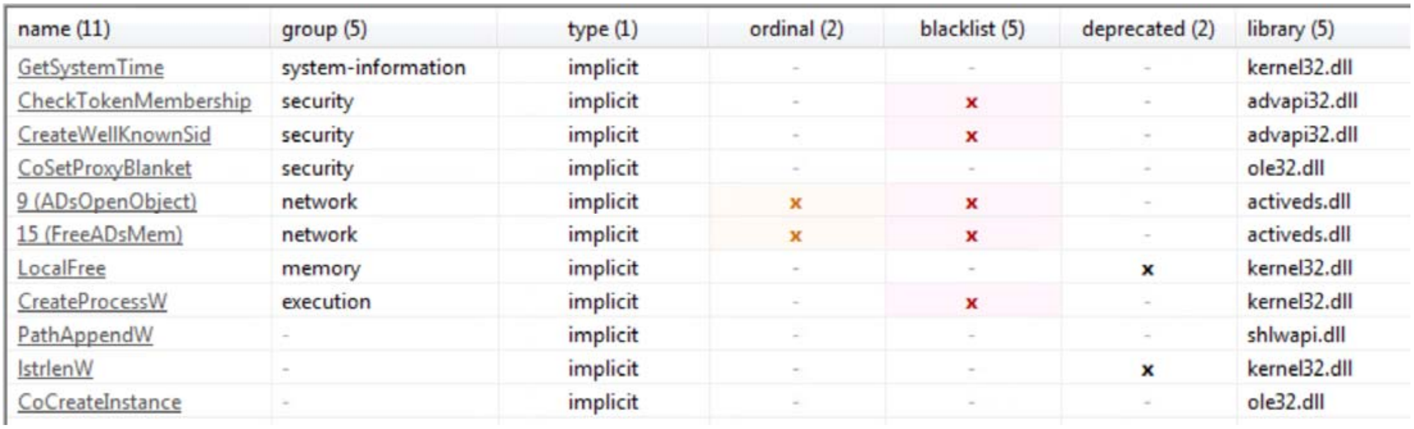
property	value	value	value
name	.text	.data	.idata
md5	DAC0CB8EB9636C88FB016F...	897CF813180EBD185C94138...	964EF105D227BAC747B79BB...
entropy	6.594	6.790	4.126
file-ratio (99.90%)	93.38 %	6.46 %	0.05 %
raw-address	0x00000400	0x000E0400	0x000EFC00
raw-size (981504 bytes)	0x000E0000 (917504 bytes)	0x0000F800 (63488 bytes)	0x00000200 (512 bytes)
virtual-address	0x00401000	0x004E1000	0x004FA000
virtual-size (1015933 bytes)	0x000DFE63 (917091 bytes)	0x00018044 (98372 bytes)	0x000001D6 (470 bytes)
entry-point	0x000BFF90	-	-
characteristics	0x60000020	0xC0000040	0x40000040
writable	-	x	-
executable	x	-	-
shareable	-	-	-
discardable	-	-	-
initialized-data	-	x	x
uninitialized-data	-	-	-
unreadable	-	-	-
self-modifying	-	-	-
virtualized	-	-	-
file	n/a	n/a	n/a

Figure 3 Virtual sizes of PE sections are not much larger than raw sizes



type (2)	size (bytes)	file-offset	blacklist (6)	hint (55)	value (5193)
unicode	12	0x00003071	-	utility	explorer.exe
unicode	36	0x000025BD	-	utility	cmd.exe /c "shutdown.exe /r /f /t 0"
unicode	7	0x000029BC	-	utility	cmd.exe
unicode	4	0x00003CF5	-	utility	Stop
unicode	5	0x00003CEA	-	utility	Start
unicode	8	0x00002159	-	utility	Services
unicode	6	0x00002573	-	utility	Create
unicode	11	0x0000222C	-	utility	/IM "%s" /F
unicode	30	0x000037DB	-	url-pattern	https://tox.chat/download.html
unicode	18	0x00003B4F	-	url-pattern	https://bigblog.at
unicode	69	0x00003AF6	-	url-pattern	http://lockbitapt6vx57t3eeqjofwgcglmutr3a35nygvokja5uuccip4ykyd.onion
unicode	60	0x00003175	-	registry	Software\Microsoft\Windows NT\CurrentVersion\ICM\Calibration
unicode	45	0x0000280E	-	registry	SOFTWARE\Microsoft\Windows\CurrentVersion\Run
unicode	37	0x00002A8E	-	registry	SOFTWARE\%02X%02X%02X%02X%02X%02X%02X
unicode	38	0x000031DF	-	guid	{D2E7041B-2927-42fb-8E9F-7CE93B6DC937}
unicode	38	0x000030E7	-	guid	{3E5FC7F9-9A51-4367-9063-A120244FBEC7}
unicode	38	0x00002867	-	guid	{2C5F9FCC-F266-43F6-BFD7-838DAE269E11}
unicode	11	0x00003E6C	-	file	shell32.dll
ascii	9	0x000EFDCC	-	file	ole32.dll
unicode	12	0x0000252D	-	file	gpupdate.exe
unicode	12	0x00003E85	-	file	comctl32.dll
unicode	13	0x00003056	-	file	\explorer.exe
unicode	13	0x00002172	-	file	\Services.xml
unicode	19	0x00002274	-	file	\ScheduledTasks.xml
unicode	18	0x0000213B	-	file	\NetworkShares.xml
unicode	23	0x000028A8	-	file	\LockBit_Ransomware.hta
unicode	10	0x000019F7	-	file	\Files.xml
unicode	40	0x00002FA9	-	file	\??\C:\windows\system32\%02X%02X%02X.ico
ascii	11	0x000028C0	-	file	Shell32.dll
ascii	11	0x000EFD06	-	file	SHLWAPI.dll
unicode	12	0x00002341	-	file	Registry.pol
unicode	20	0x000032D9	-	file	RESTORE-MY-FILES.TXT

Figure 4 Static analysis shows many readable strings



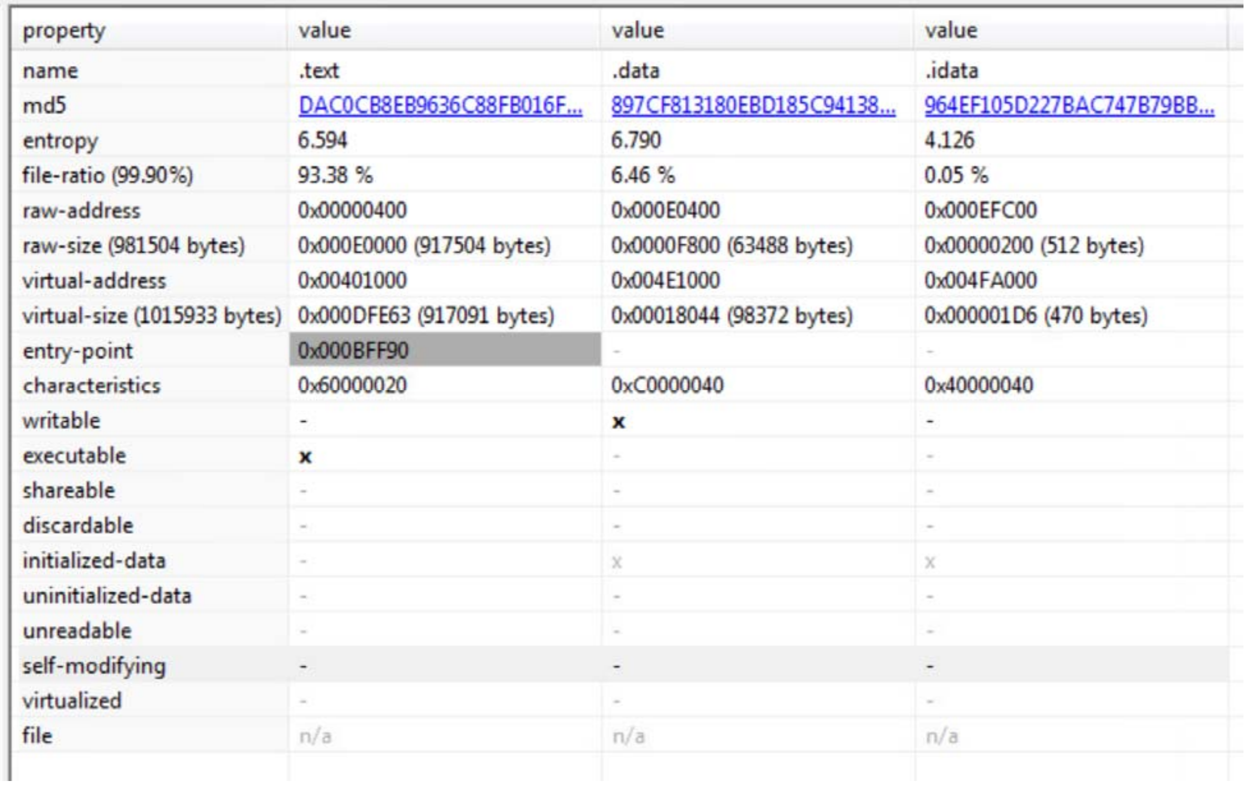
name (11)	group (5)	type (1)	ordinal (2)	blacklist (5)	deprecated (2)	library (5)
GetSystemTime	system-information	implicit	-	-	-	kernel32.dll
CheckTokenMembership	security	implicit	-	x	-	advapi32.dll
CreateWellKnownSid	security	implicit	-	x	-	advapi32.dll
CoSetProxyBlanket	security	implicit	-	-	-	ole32.dll
9 (ADsOpenObject)	network	implicit	x	x	-	activeds.dll
15 (FreeADsMem)	network	implicit	x	x	-	activeds.dll
LocalFree	memory	implicit	-	-	x	kernel32.dll
CreateProcessW	execution	implicit	-	x	-	kernel32.dll
PathAppendW	-	implicit	-	-	-	shlwapi.dll
lstrlenW	-	implicit	-	-	x	kernel32.dll
CoCreateInstance	-	implicit	-	-	-	ole32.dll

Figure 5 Static imports shown by PEStudio

PE Sections

The program has 3 PE sections:

1. **.text section:** This section stores the actual instructions of the program that get executed. This is the only section which has executable permission. The **entry point** of this section is the entry point of the program code. The execution of the program begins at the entry point of the .text section.
2. **.data section:** This section contains globally accessible data. It can contain arrays, structs or strings which can be read and modified from anywhere in the program. This section does not contain any executable instructions so it does not have the executable permission.
3. **.idata section:** This section contains information about the libraries and functions the program imports. This information is stored in tables called **import address table (IAT)** and **import directory table (IDT).** The IDT gets loaded first and contains details and addresses of the imported libraries. The IAT gets loaded next and contains details and addresses of imported functions (It uses the library addresses from IDT to figure out the function addresses). .idata is a **read-only** section. It does not contain any executable information so it doesn't have executable permission.

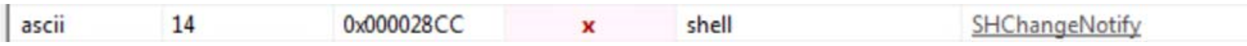


property	value	value	value
name	.text	.data	.idata
md5	DAC0CB8EB9636C88FB016F...	897CF813180EBD185C94138...	964EF105D227BAC747B79BB...
entropy	6.594	6.790	4.126
file-ratio (99.90%)	93.38 %	6.46 %	0.05 %
raw-address	0x00000400	0x000E0400	0x000EFC00
raw-size (981504 bytes)	0x000E0000 (917504 bytes)	0x0000F800 (63488 bytes)	0x00000200 (512 bytes)
virtual-address	0x00401000	0x004E1000	0x004FA000
virtual-size (1015933 bytes)	0x000DFE63 (917091 bytes)	0x00018044 (98372 bytes)	0x000001D6 (470 bytes)
entry-point	0x000BFF90	-	-
characteristics	0x60000020	0xC0000040	0x40000040
writable	-	x	-
executable	x	-	-
shareable	-	-	-
discardable	-	-	-
initialized-data	-	x	x
uninitialized-data	-	-	-
unreadable	-	-	-
self-modifying	-	-	-
virtualized	-	-	-
file	n/a	n/a	n/a

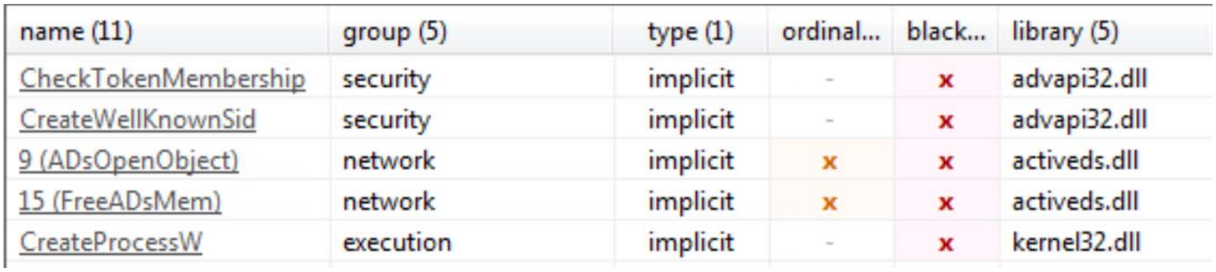
Figure 6 PE Sections

Suspicious Imports

1. **SHChangeNotify:** This Windows API function is used inform the Windows Shell of a change that has taken place on the system. The Windows Shell is the UI component responsible for displaying GUI elements like windows and menu bars on the screen. The malware could be using SHChangeNotify to refresh the Windows Shell's view of the system after some change like creation or deletion of a file has taken place.
2. **CheckTokenMembership:** This is a WindowsAPI function to check if the current user belongs to a particular user group like Administrators. The malware might be checking to see if the user is a privileged user. If not, the malware might elevate its privileges. There's a string "*Elevation:Administrator!new*" which might be related to this.
3. **CreateWellKnownSid**
4. **CreateProcess:** The malware probably spawns processes.
5. **ADsOpenObject and FreeADsMem:** Active directory related functions. Malware uses the active directory to discover other computers in the network and infect them.

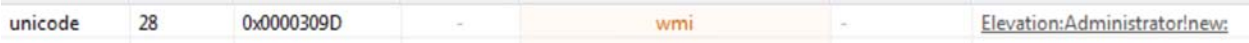


ascii	14	0x000028CC	x	shell	SHChangeNotify



name (11)	group (5)	type (1)	ordinal...	black...	library (5)
CheckTokenMembership	security	implicit	-	✗	advapi32.dll
CreateWellKnownSid	security	implicit	-	✗	advapi32.dll
9 (ADsOpenObject)	network	implicit	✗	✗	activeds.dll
15 (FreeADsMem)	network	implicit	✗	✗	activeds.dll
CreateProcessW	execution	implicit	-	✗	kernel32.dll

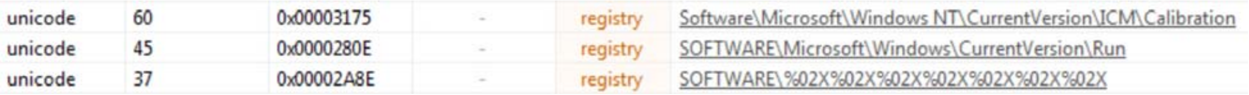
Figure 7 Suspicious Imports



unicode	28	0x0000309D	-	wmi	-	Elevation:Administrator!new:

Suspicious Strings

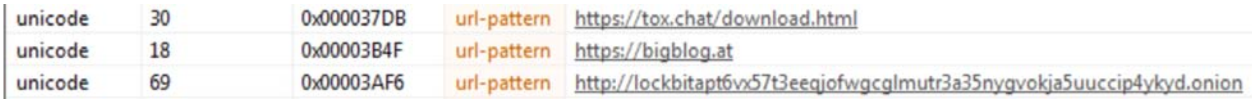
1. **Registry related strings:** The following registry keys are found during static analysis. The **SOFTWARE\Microsoft\Windows\CurrentVersion\Run** key is suspicious as it is a common mechanism malware sets up persistence. The **Software\Microsoft\Windows NT\CurrentVersion\ICM\Calibration** key stores color profile information for IO devices like monitors and printers.



unicode	60	0x00003175	-	registry	Software\Microsoft\Windows NT\CurrentVersion\ICM\Calibration
unicode	45	0x0000280E	-	registry	SOFTWARE\Microsoft\Windows\CurrentVersion\Run
unicode	37	0x00002A8E	-	registry	SOFTWARE\%02X%02X%02X%02X%02X%02X%02X

Figure 8 Registry related strings

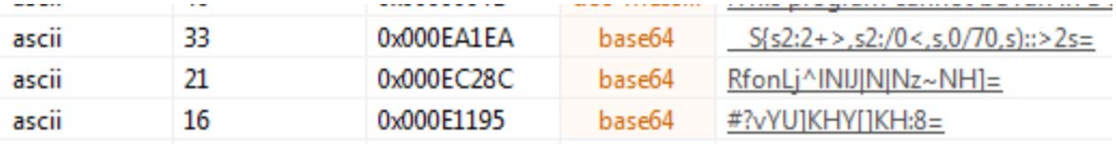
2. **URL strings:** The "lockbit" URL can give some clue about the malware being Lockbit ransomware. The "tox.chat" URL is also suspicious. Googling "Tox messenger" brings up search results showing how this peer-to-peer, encrypted messaging app was used as a command-and-control server in cyber-attacks.



unicode	30	0x000037DB	url-pattern	https://tox.chat/download.html
unicode	18	0x00003B4F	url-pattern	https://bigblog.at
unicode	69	0x00003AF6	url-pattern	http://lockbitapt6vx57t3eeqjofwgcglmutr3a35nygvokja5uuccip4ykyd.onion

Figure 9 URL strings

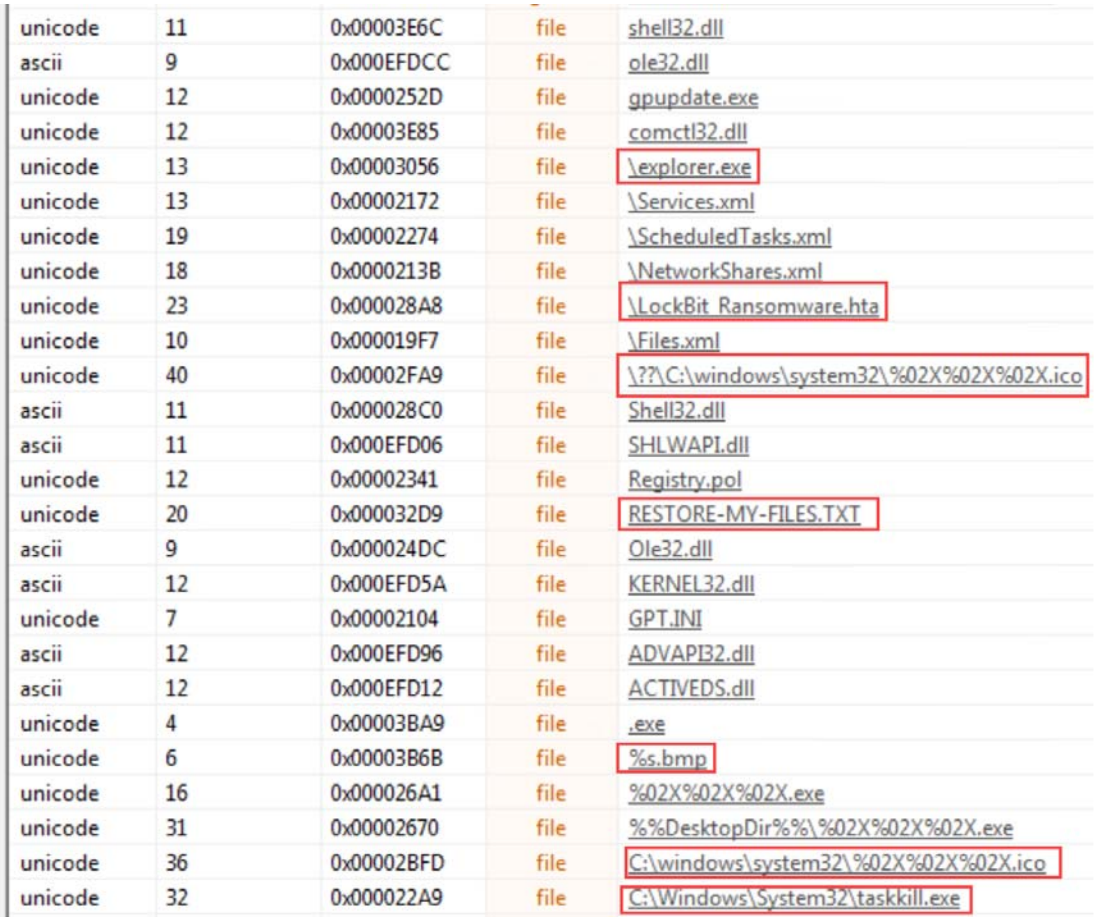
3. **Base64 strings:** PE Studio recognizes some base64 encoded strings suggesting that the malware might be using base64 encoding.



ascii	33	0x000EA1EA	base64	S{s2:2+>,s2:/0<,s,0/70,s)::>2s=
ascii	21	0x000EC28C	base64	RfonLj^INIJ|N|Nz~NH]=
ascii	16	0x000E1195	base64	#?vYU]KHY[]KH:8=

Figure 10 Base64 encoded strings

4. **Filename strings:** Some of the interesting file strings recognized by PE Studio are –
    a. **explorer.exe** – The malware uses the file explorer process to manipulate files.
    b. **Lockbit_Ransomware.hta** – HTA files are HTML executable files and are frequently used to spread malware through email attachments. An HTA file can execute code on a machine with much higher privileges than a normal HTML file which makes this file highly suspicious.
    c. **.ico and .bmp files** – These could be the path of the icon files the malware creates.
    d. **RESTORE-MY-FILES.TXT:** This is the name of the ransom note created in each folder where the malware encrypts files.
    e. **Taskkill.exe** – This Windows program is used to kill a process by its PID or name. The malware might be using taskkill to kill other processes.



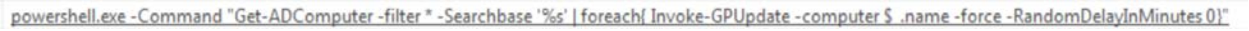
unicode	11	0x00003E6C	file	shell32.dll
ascii	9	0x000EFDCC	file	ole32.dll
unicode	12	0x0000252D	file	gpupdate.exe
unicode	12	0x00003E85	file	comctl32.dll
unicode	13	0x00003056	file	\explorer.exe
unicode	13	0x00002172	file	\Services.xml
unicode	19	0x00002274	file	\ScheduledTasks.xml
unicode	18	0x0000213B	file	\NetworkShares.xml
unicode	23	0x000028A8	file	\LockBit_Ransomware.hta
unicode	10	0x000019F7	file	\Files.xml
unicode	40	0x00002FA9	file	\??\C:\windows\system32\%02X%02X%02X.ico
ascii	11	0x000028C0	file	Shell32.dll
ascii	11	0x000EFD06	file	SHLWAPI.dll
unicode	12	0x00002341	file	Registry.pol
unicode	20	0x000032D9	file	RESTORE-MY-FILES.TXT
ascii	9	0x000024DC	file	Ole32.dll
ascii	12	0x000EFD5A	file	KERNEL32.dll
unicode	7	0x00002104	file	GPT.INI
ascii	12	0x000EFD96	file	ADVAPI32.dll
ascii	12	0x000EFD12	file	ACTIVEDS.dll
unicode	4	0x00003BA9	file	.exe
unicode	6	0x00003B6B	file	%s.bmp
unicode	16	0x000026A1	file	%02X%02X%02X.exe
unicode	31	0x00002670	file	%%DesktopDir%%\%02X%02X%02X.exe
unicode	36	0x00002BFD	file	C:\windows\system32\%02X%02X%02X.ico
unicode	32	0x000022A9	file	C:\Windows\System32\taskkill.exe

Figure 11 Filenames found by PEStudio

5. **Active Directory and LDAP related strings**:
Microsoft's Active Directory is a service which stores information about users, groups and computers in network in a hierarchical database. Active directory is often used for authenticating and authorizing users. AD is based on Lightweight Directory Access Protocol (LDAP). With AD, multiple computers in an organization can be controlled from a central

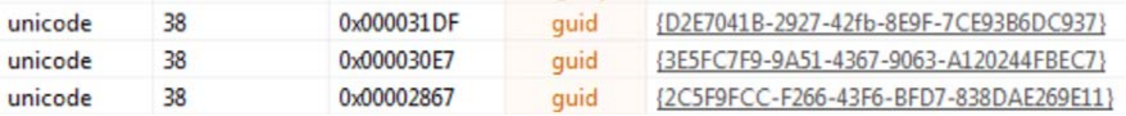
controller machine. If the malware infects a Windows Server machine, it might be using functions related to AD to search for other machines to infect within the network.

i. The strings contain a Powershell command that gets computers from the active directory and forces a Group Policy refresh (using **Invoke-GPUpdate**) on them. The malware might be changing the group policies and forcing new group policies on other computers in the network to infiltrate them.



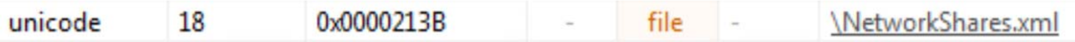
```
powershell.exe -Command "Get-ADComputer -filter * -Searchbase '%s' | foreach{ Invoke-GPUpdate -computer $ .name -force -RandomDelayInMinutes 0}"
```

ii. The malware has strings containing GUID (Globally Unique Identifier) values. GUID are 128-bit values used to identify objects in Active Directory, such as users, groups, and computers.



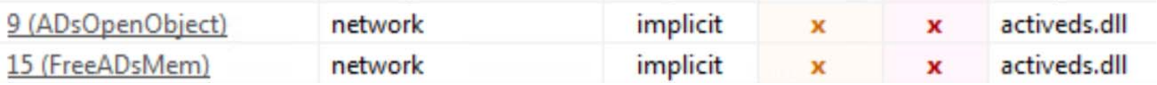
unicode	38	0x000031DF	guid	{D2E7041B-2927-42fb-8E9F-7CE93B6DC937}
unicode	38	0x000030E7	guid	{3E5FC7F9-9A51-4367-9063-A120244FBEC7}
unicode	38	0x00002867	guid	{2C5F9FCC-F266-43F6-BFD7-838DAE269E11}

iii. **NetworkShares.xml** – This file stores the network sharing preferences for an active directory group. The malware might be modifying this file to change the sharing preferences for a group to infect other computers in the group.



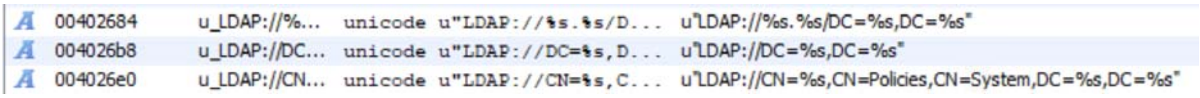
unicode	18	0x0000213B	-	file	-	\NetworkShares.xml

iv. **activeeds.dll** – The malware loads activeds.dll and probably calls the ADsOpenObject and FreeADsMem functions from it.



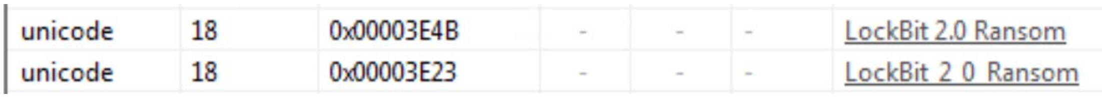
9 (ADsOpenObject)	network	implicit	x	x	activeds.dll
15 (FreeADsMem)	network	implicit	x	x	activeds.dll

v. **LDAP strings:**

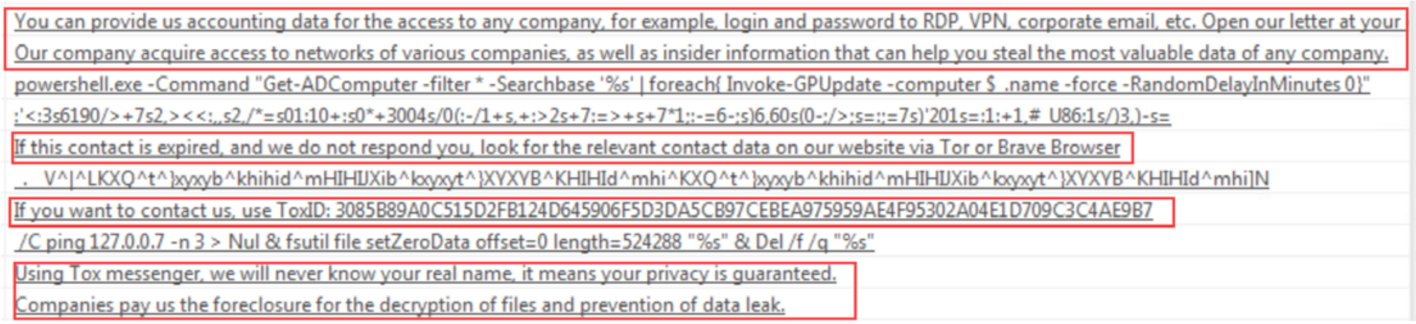


```
A  00402684   u_LDAP://%...   unicode u"LDAP://%s.%s/D...   u"LDAP://%s.%s/DC=%s,DC=%s"
A  004026b8   u_LDAP://DC...  unicode u"LDAP://DC=%s,D...   u"LDAP://DC=%s,DC=%s"
A  004026e0   u_LDAP://CN...  unicode u"LDAP://CN=%s,C...   u"LDAP://CN=%s,CN=Policies,CN=System,DC=%s,DC=%s"
```

6. **Ransom note and Lockbit strings:** Strings telling the user their files have been stolen and ways to contact the hacker to get the files back. These messages encourage the user to contact the malware authors using Tox messenger on Tor or Brave Browser.



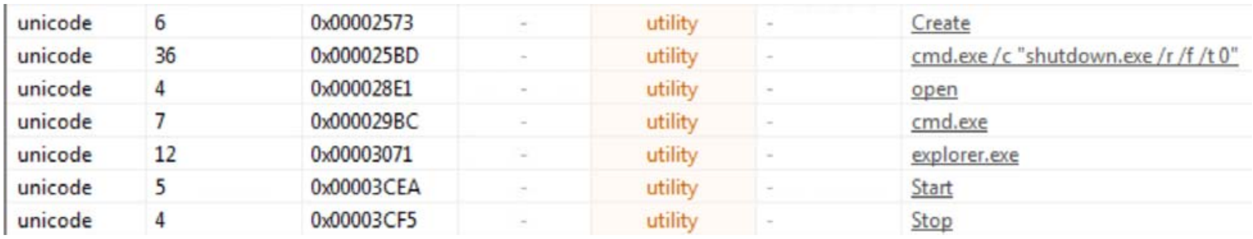
unicode	18	0x00003E4B	-	-	-	LockBit 2.0 Ransom
unicode	18	0x00003E23	-	-	-	LockBit 2 0 Ransom



```
You can provide us accounting data for the access to any company, for example, login and password to RDP, VPN, corporate email, etc. Open our letter at your
Our company acquire access to networks of various companies, as well as insider information that can help you steal the most valuable data of any company.
powershell.exe -Command "Get-ADComputer -filter * -Searchbase '%s' | foreach{ Invoke-GPUpdate -computer $_.name -force -RandomDelayInMinutes 0}"
:'<:3s6190/>+7s2,><<:,,s2,/*=s01:10+:s0*+3004s/0(:-/1+s,+:>2s+7:=>+s+7*1;:-=6-;s)6,60s(0-;/>;s=:;=7s)'201s=:1:+1,#_U86:1s/)3,)-s=
If this contact is expired, and we do not respond you, look for the relevant contact data on our website via Tor or Brave Browser
 .   V^|^LKXQ^t^}xyxyb^khihid^mHIHIJXib^kxyxyt^}XYXYB^KHIHId^mhi^KXQ^t^}xyxyb^khihid^mHIHIJXib^kxyxyt^}XYXYB^KHIHId^mhi]N
If you want to contact us, use ToxID: 3085B89A0C515D2FB124D645906F5D3DA5CB97CEBEA975959AE4F95302A04E1D709C3C4AE9B7
/C ping 127.0.0.7 -n 3 > Nul & fsutil file setZeroData offset=0 length=524288 "%s" & Del /f /q "%s"
Using Tox messenger, we will never know your real name, it means your privacy is guaranteed.
Companies pay us the foreclosure for the decryption of files and prevention of data leak.
```

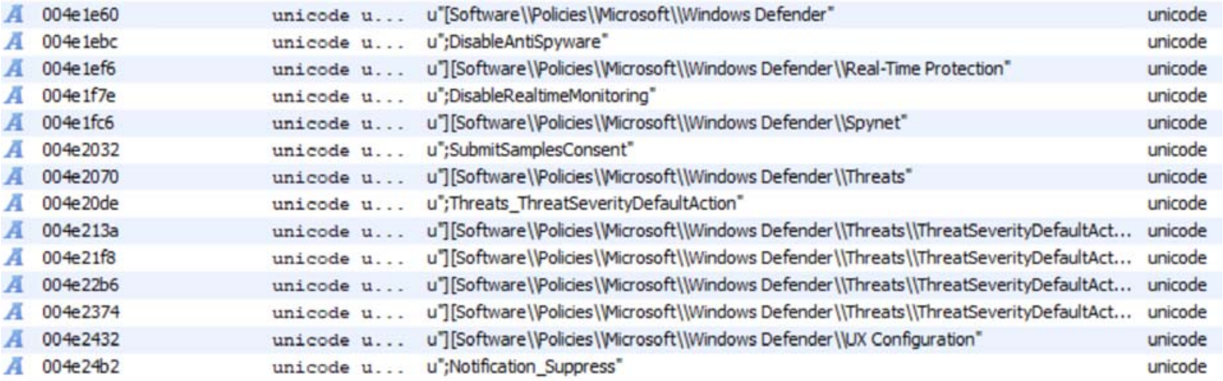
Figure 12 Ransom note strings

7. **Other suspicious strings:**
    a. **Shutdown command:** The *cmd.exe /c "shutdown.exe /r /f /t 0"* command restarts the machine after a delay of 0ms. The malware might be restarting the machine if some conditions are matched.
    b. **cmd.exe and explorer.exe** – The malware makes use of the command line and Windows Explorer to infect the system.
    c. **Windows Defender related registry keys** – The malware might be changing these keys to prevent Windows Defender from detecting it or taking any action against it.
    d. **Printing related strings**: The malware might be creating a PDF file but no such file was found during dynamic analysis.



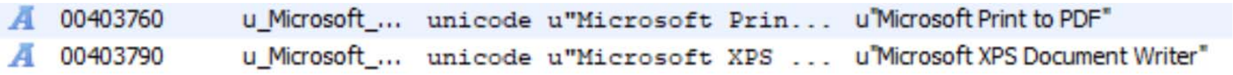
unicode	6	0x00002573	-	utility	-	Create
unicode	36	0x000025BD	-	utility	-	cmd.exe /c "shutdown.exe /r /f /t 0"
unicode	4	0x000028E1	-	utility	-	open
unicode	7	0x000029BC	-	utility	-	cmd.exe
unicode	12	0x00003071	-	utility	-	explorer.exe
unicode	5	0x00003CEA	-	utility	-	Start
unicode	4	0x00003CF5	-	utility	-	Stop

Figure 13 Some more suspicious strings



004e1e60	unicode u...	u"[Software\\Policies\\Microsoft\\Windows Defender"	unicode
004e1ebc	unicode u...	u";DisableAntiSpyware"	unicode
004e1ef6	unicode u...	u"][Software\\Policies\\Microsoft\\Windows Defender\\Real-Time Protection"	unicode
004e1f7e	unicode u...	u";DisableRealtimeMonitoring"	unicode
004e1fc6	unicode u...	u"][Software\\Policies\\Microsoft\\Windows Defender\\Spynet"	unicode
004e2032	unicode u...	u";SubmitSamplesConsent"	unicode
004e2070	unicode u...	u"][Software\\Policies\\Microsoft\\Windows Defender\\Threats"	unicode
004e20de	unicode u...	u";Threats_ThreatSeverityDefaultAction"	unicode
004e213a	unicode u...	u"][Software\\Policies\\Microsoft\\Windows Defender\\Threats\\ThreatSeverityDefaultAct...	unicode
004e21f8	unicode u...	u"][Software\\Policies\\Microsoft\\Windows Defender\\Threats\\ThreatSeverityDefaultAct...	unicode
004e22b6	unicode u...	u"][Software\\Policies\\Microsoft\\Windows Defender\\Threats\\ThreatSeverityDefaultAct...	unicode
004e2374	unicode u...	u"][Software\\Policies\\Microsoft\\Windows Defender\\Threats\\ThreatSeverityDefaultAct...	unicode
004e2432	unicode u...	u"][Software\\Policies\\Microsoft\\Windows Defender\\UX Configuration"	unicode
004e24b2	unicode u...	u";Notification_Suppress"	unicode

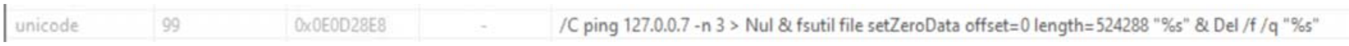
Figure 14 Windows Defender related strings



```
00403760    u_Microsoft_...    unicode u"Microsoft Prin...    u"Microsoft Print to PDF"
00403790    u_Microsoft_...    unicode u"Microsoft XPS ...    u"Microsoft XPS Document Writer"
```

Figure 15 Printing related strings

e. **Ping command:** There is a ping command the malware contains to ping localhost's loopback address and set 524288 bytes of the file whose name would be supplied at runtime to 0. It then has a command to delete the file whose name is supplied in the command. Looking for the references of this string in Ghidra doesn't show anything so it might not be being used and the malware author might have included it to mislead the analyst.

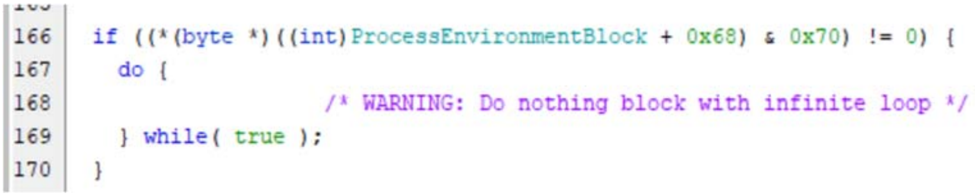


```
unicode    99    0x0E0D28E8    -    /C ping 127.0.0.7 -n 3 > Nul & fsutil file setZeroData offset=0 length=524288 "%s" & Del /f /q "%s"
```

Figure 16 Ping command

Ghidra Analysis

1. **Anti-debugging:** The malware employs an anti-debugging technique where it checks the value at the offset 0x68 of the Process Environment Block (PEB). It compares this value against 0x70. If the value at the offset is 0x70, the program goes in an infinite loop. This anti-debugging strategy was at the beginning of the entry function. The ScyllaHide plugin in x32dbg takes care of thwarting this anti-debugging strategy.



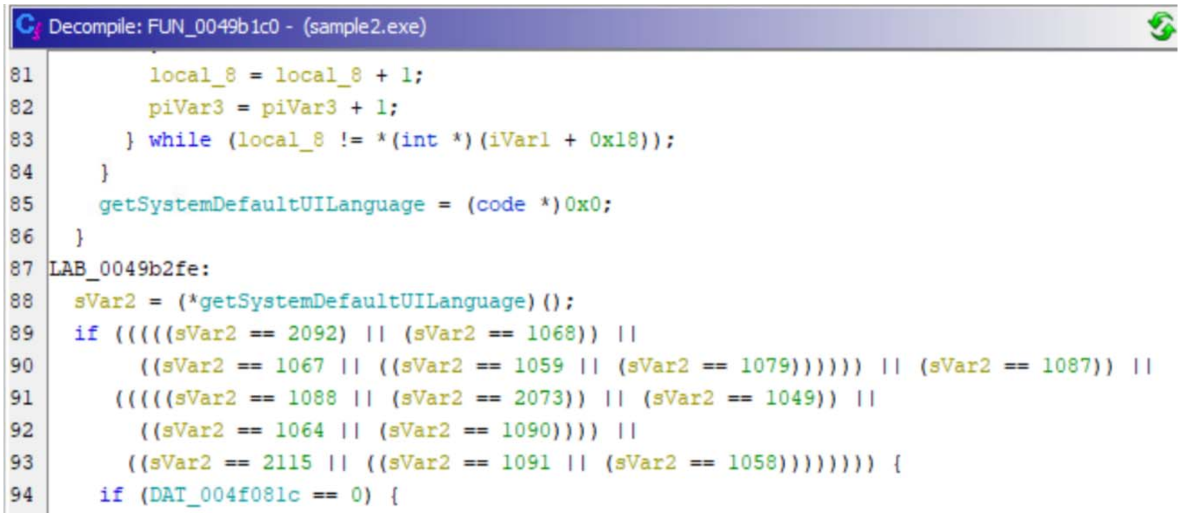
```
  if ((*(byte *)((int)ProcessEnvironmentBlock + 0x68) & 0x70) != 0) {
    do {
                    /* WARNING: Do nothing block with infinite loop */
    } while( true );
  }
```

Figure 17 Anti-Debugging Technique in entry()

2. **Checking system default language:** In the function starting at location 49b1c0, the malware gets system's default UI language using kernel32.dll's *GetSystemDefaultUILanguage* function. It then compares the return value of this function with a bunch of Slavic languages like the following:

Language	Language Code
Russian	1049
Azerbaijani	2082
Armenian	1067
Belarusian	1059

If the default language on the system is set to any of the Slavic languages, the program exits and doesn't cause any harm to the machine. This could indicate that the malware originated in Russia or one of the Slavic countries.



```
        local_8 = local_8 + 1;
        piVar3 = piVar3 + 1;
      } while (local_8 != *(int *)(iVar1 + 0x18));
    }
    getSystemDefaultUILanguage = (code *)0x0;
  }
LAB_0049b2fe:
  sVar2 = (*getSystemDefaultUILanguage)();
  if (((((sVar2 == 2092) || (sVar2 == 1068)) ||
       ((sVar2 == 1067 || ((sVar2 == 1059 || (sVar2 == 1079)))))) || (sVar2 == 1087)) ||
     (((((sVar2 == 1088 || (sVar2 == 2073)) || (sVar2 == 1049)) ||
       ((sVar2 == 1064 || (sVar2 == 1090)))) ||
      ((sVar2 == 2115 || ((sVar2 == 1091 || (sVar2 == 1058)))))))) {
    if (DAT_004f081c == 0) {
```

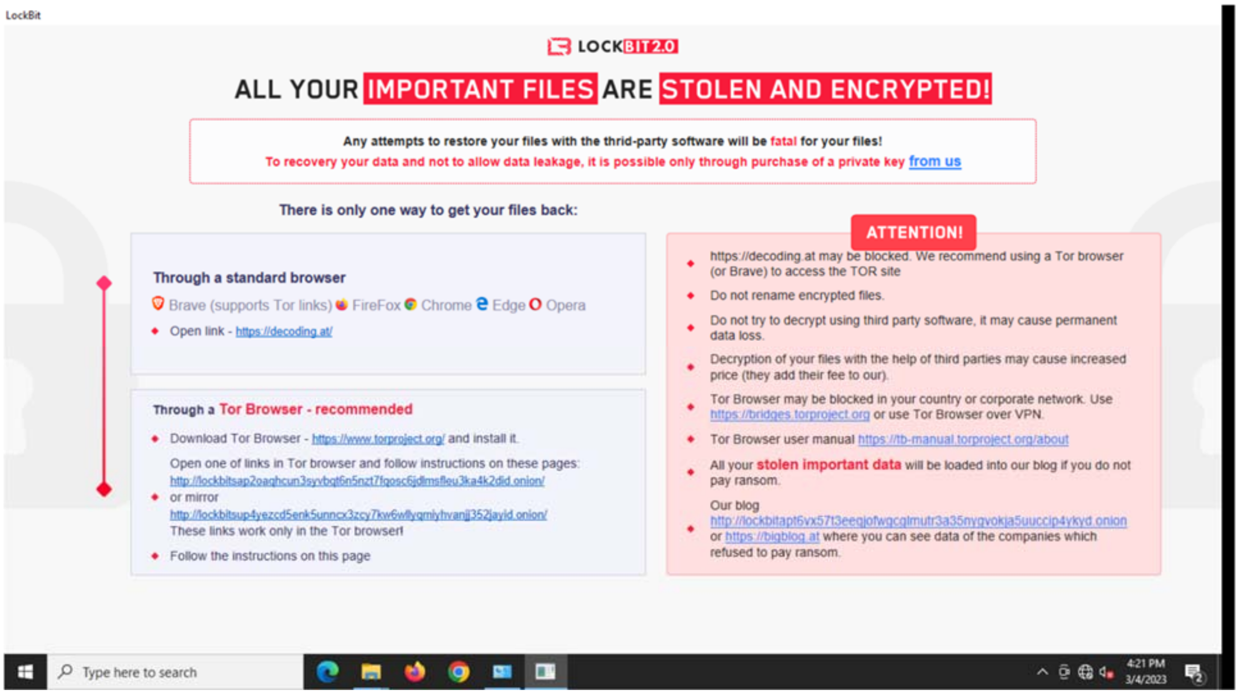
Figure 18 Checking if the default system UI language is Slavic

3. Not a lot of information could be obtained by static analysis with Ghidra in isolation. It proved to be useful to lookup function code while doing dynamic analysis with x32dbg.

Dynamic Analysis

Behavior after running the malware

1. After the malware is run, it displays the following window alerting the user that their files have been stolen and how to get them back. The malware authors ask the user to use the Tor browser and navigate to one of the provided links. The window displayed is an HTA (HTML Application) that is stored in a file called **Lockbit_Ransomware.hta** on the desktop that the malware creates.



LockBit

LOCKBIT 2.0

ALL YOUR IMPORTANT FILES ARE STOLEN AND ENCRYPTED!

Any attempts to restore your files with the thrid-party software will be fatal for your files!
To recovery your data and not to allow data leakage, it is possible only through purchase of a private key from us

There is only one way to get your files back:

Through a standard browser

Brave (supports Tor links) FireFox Chrome Edge Opera

- Open link - https://decoding.at/

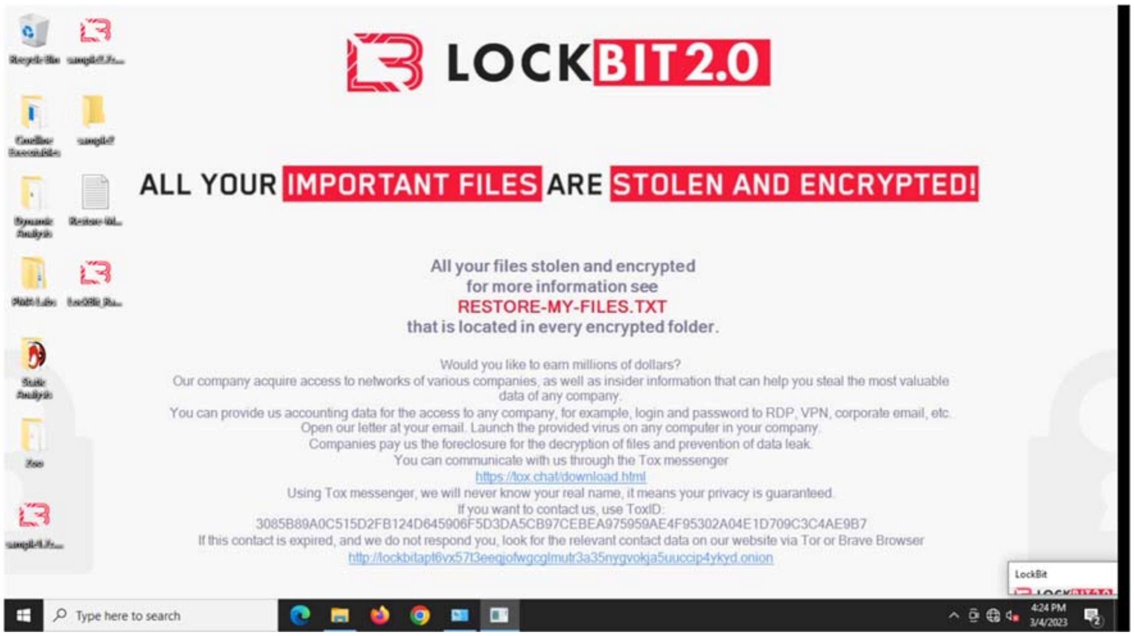
Through a Tor Browser - recommended

- Download Tor Browser - https://www.torproject.org/ and install it.
- Open one of links in Tor browser and follow instructions on these pages:
  http://lockbitsap2oaghcun3syvbqt6n5nzt7fqosc6jdlmsfleu3ka4k2did.onion/
  or mirror
  http://lockbitsup4yezcd5enk5unncx3zcy7kw6wllyqmiyhvanjj352jayid.onion/
  These links work only in the Tor browser!
- Follow the instructions on this page

ATTENTION!

- https://decoding.at may be blocked. We recommend using a Tor browser (or Brave) to access the TOR site
- Do not rename encrypted files.
- Do not try to decrypt using third party software, it may cause permanent data loss.
- Decryption of your files with the help of third parties may cause increased price (they add their fee to our).
- Tor Browser may be blocked in your country or corporate network. Use https://bridges.torproject.org or use Tor Browser over VPN.
- Tor Browser user manual https://tb-manual.torproject.org/about
- All your stolen important data will be loaded into our blog if you do not pay ransom.
- Our blog http://lockbitapt6vx57t3eeqjofwgcglmutr3a35nygvokja5uuccip4ykyd.onion or https://bigblog.at where you can see data of the companies which refused to pay ransom.

2. Minimizing the above window shows the desktop. The desktop wallpaper is changed by the malware. All the files other than .exe and .dll from all folders (except the C:\Windows) get encrypted by the malware. The encrypted files have an extension of .lockbit. Files with a lockbit extension have a distinct icon that the malware generates.



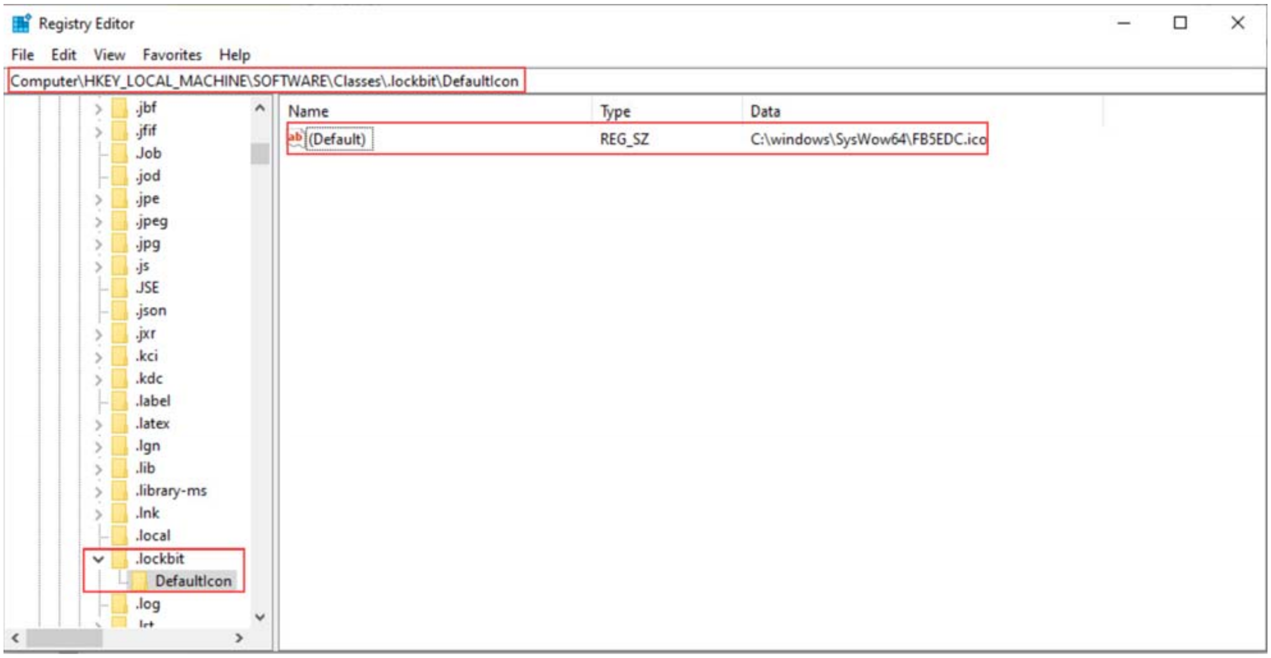
ALL YOUR IMPORTANT FILES ARE STOLEN AND ENCRYPTED!

All your files stolen and encrypted
for more information see
RESTORE-MY-FILES.TXT
that is located in every encrypted folder.

Would you like to earn millions of dollars?
Our company acquire access to networks of various companies, as well as insider information that can help you steal the most valuable data of any company.
You can provide us accounting data for the access to any company, for example, login and password to RDP, VPN, corporate email, etc.
Open our letter at your email. Launch the provided virus on any computer in your company.
Companies pay us the foreclosure for the decryption of files and prevention of data leak.
You can communicate with us through the Tox messenger
https://tox.chat/download.html
Using Tox messenger, we will never know your real name, it means your privacy is guaranteed.
If you want to contact us, use ToxID:
3085B89A0C515D2FB124D645906F5D3DA5CB97CEBEA975959AE4F95302A04E1D709C3C4AE9B7
If this contact is expired, and we do not respond you, look for the relevant contact data on our website via Tor or Brave Browser
http://lockbitapt6vx57t3eeqjofwgcglmutr3a35nygvokja5uuccip4ykyd.onion

Figure 19 Desktop after executing the malware

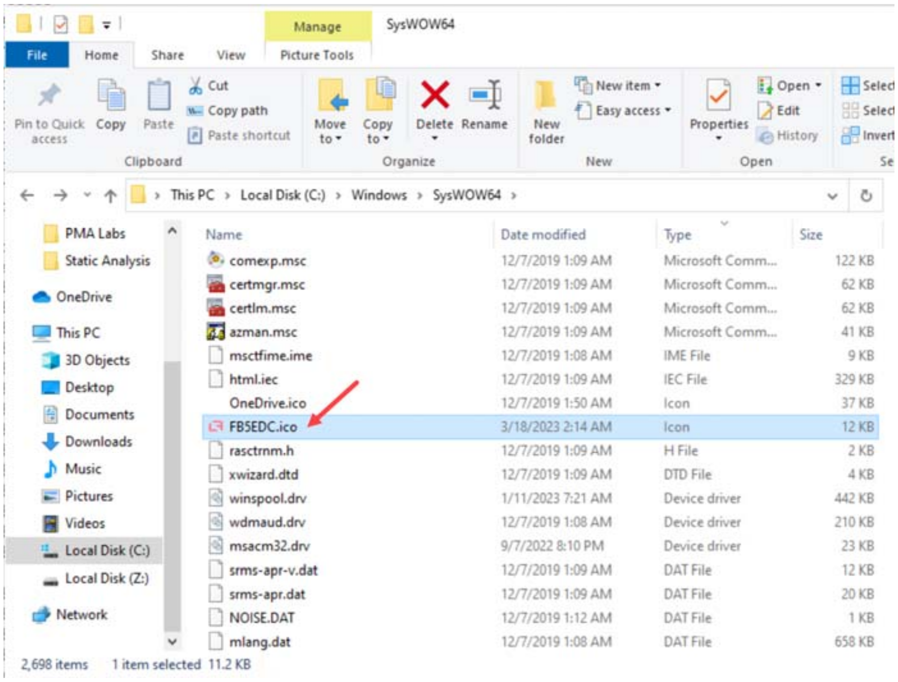
3. The malware creates and stores the icon associated with .lockbit extension in the **C:\Windows\SysWow64** folder. It also adds a registry entry at **HKEY_LOCAL_MACHINE\SOFTWARE\Classes\.lockbit\DefaultIcon** for to associate this icon with the lockbit extension.



Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Classes\.lockbit\DefaultIcon

Name	Type	Data
(Default)	REG_SZ	C:\windows\SysWow64\FB5EDC.ico

Figure 20 Registry entry for default icon associated with .lockbit extension



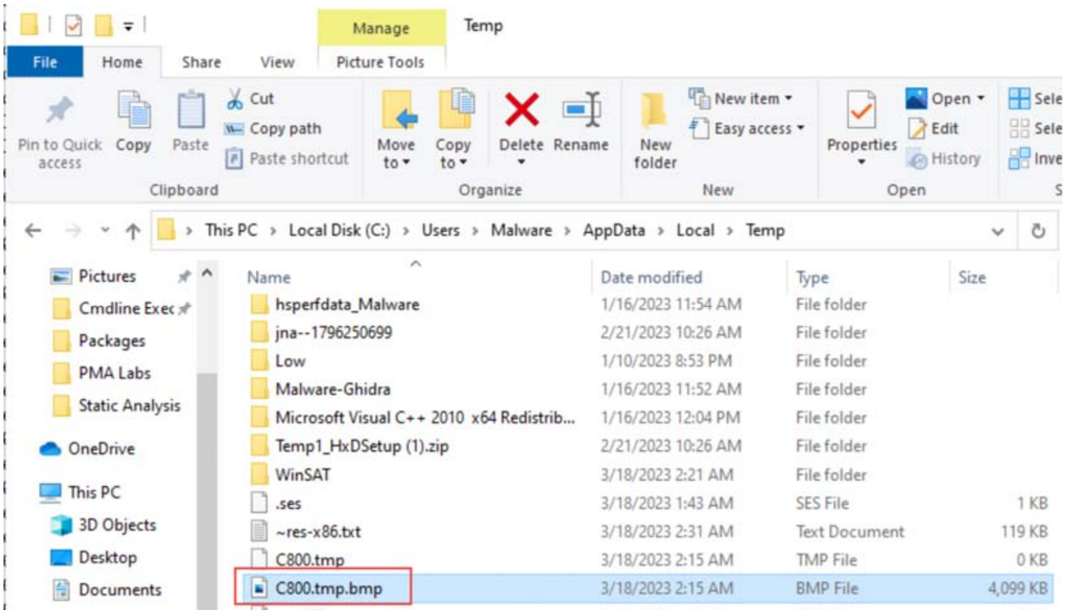
This PC › Local Disk (C:) › Windows › SysWOW64 ›

Name	Date modified	Type	Size
comexp.msc	12/7/2019 1:09 AM	Microsoft Comm...	122 KB
certmgr.msc	12/7/2019 1:09 AM	Microsoft Comm...	62 KB
certlm.msc	12/7/2019 1:09 AM	Microsoft Comm...	62 KB
azman.msc	12/7/2019 1:09 AM	Microsoft Comm...	41 KB
msctfime.ime	12/7/2019 1:08 AM	IME File	9 KB
html.iec	12/7/2019 1:09 AM	IEC File	329 KB
OneDrive.ico	12/7/2019 1:50 AM	Icon	37 KB
FB5EDC.ico	3/18/2023 2:14 AM	Icon	12 KB
rasctrnm.h	12/7/2019 1:09 AM	H File	2 KB
xwizard.dtd	12/7/2019 1:09 AM	DTD File	4 KB
winspool.drv	1/11/2023 7:21 AM	Device driver	442 KB
wdmaud.drv	12/7/2019 1:08 AM	Device driver	210 KB
msacm32.drv	9/7/2022 8:10 PM	Device driver	23 KB
srms-apr-v.dat	12/7/2019 1:09 AM	DAT File	12 KB
srms-apr.dat	12/7/2019 1:09 AM	DAT File	20 KB
NOISE.DAT	12/7/2019 1:12 AM	DAT File	1 KB
mlang.dat	12/7/2019 1:08 AM	DAT File	658 KB

2,698 items 1 item selected 11.2 KB

Figure 21 Icon associated with .lockbit extension

4. The desktop wallpaper is an image named **C800.tmp.bmp** created and stored by the malware at **C:\Users\Malware\AppData\Local\Temp**

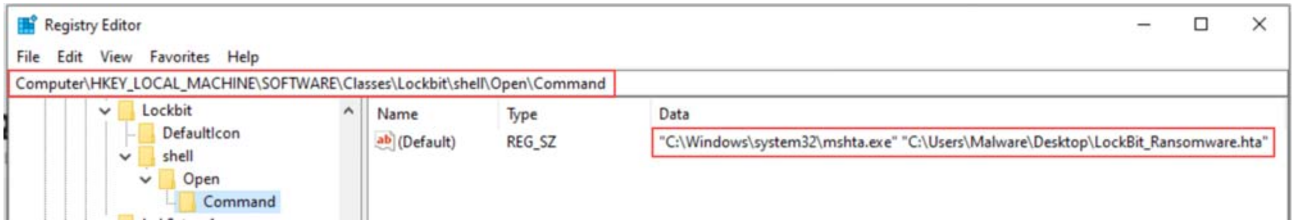


This PC › Local Disk (C:) › Users › Malware › AppData › Local › Temp

Name	Date modified	Type	Size
hsperfdata_Malware	1/16/2023 11:54 AM	File folder	
jna--1796250699	2/21/2023 10:26 AM	File folder	
Low	1/10/2023 8:53 PM	File folder	
Malware-Ghidra	1/16/2023 11:52 AM	File folder	
Microsoft Visual C++ 2010 x64 Redistrib...	1/16/2023 12:04 PM	File folder	
Temp1_HxDSetup (1).zip	2/21/2023 10:26 AM	File folder	
WinSAT	3/18/2023 2:21 AM	File folder	
.ses	3/18/2023 1:43 AM	SES File	1 KB
~res-x86.txt	3/18/2023 2:31 AM	Text Document	119 KB
C800.tmp	3/18/2023 2:15 AM	TMP File	0 KB
C800.tmp.bmp	3/18/2023 2:15 AM	BMP File	4,099 KB

Figure 22 Desktop wallpaper image

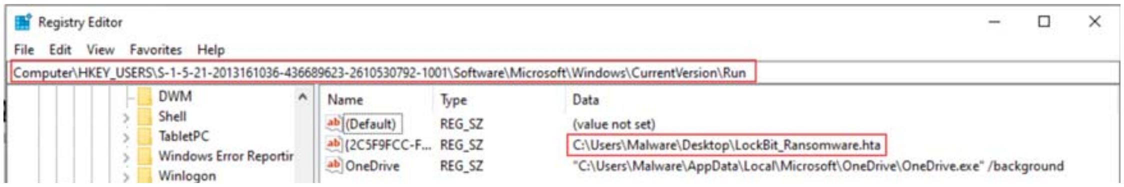
5. The malware adds another key to the registry **Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Classes\Lockbit** and sets its value to the path of mshta.exe (The program used to run HTA files) and LockBit_Ransomware.hta.



Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Classes\Lockbit\shell\Open\Command

Name	Type	Data
(Default)	REG_SZ	"C:\Windows\system32\mshta.exe" "C:\Users\Malware\Desktop\LockBit_Ransomware.hta"

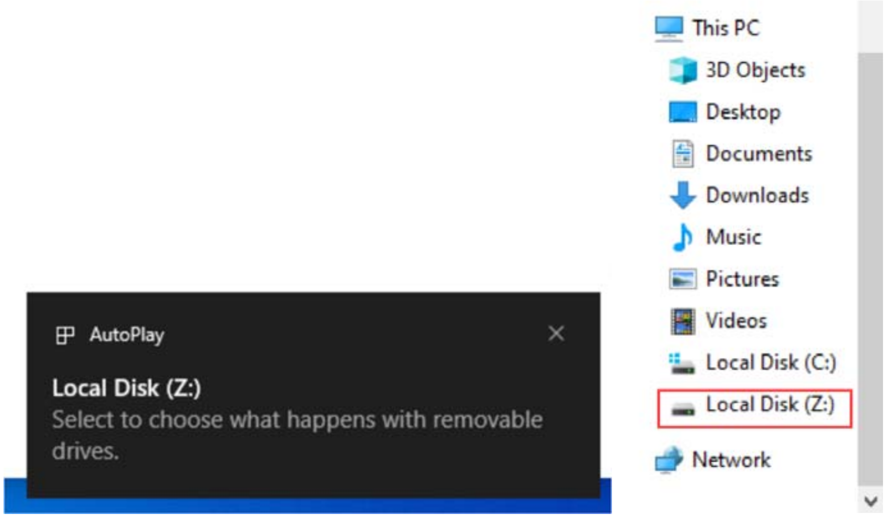
6. The malware adds the HTA file it created to **Computer\HKEY_USERS\<UserID>\Software\Microsoft\Windows\CurrentVersion\Run** so that it runs on startup.



Computer\HKEY_USERS\S-1-5-21-2013161036-436689623-2610530792-1001\Software\Microsoft\Windows\CurrentVersion\Run

Name	Type	Data
(Default)	REG_SZ	(value not set)
{2C5F9FCC-F...	REG_SZ	C:\Users\Malware\Desktop\LockBit_Ransomware.hta
OneDrive	REG_SZ	"C:\Users\Malware\AppData\Local\Microsoft\OneDrive\OneDrive.exe" /background

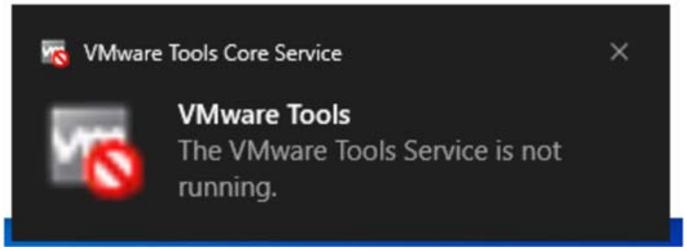
7. Once the malware it run, it mounts a Z drive on the machine. The Z drive contains no data.



Local Disk (Z:)
Select to choose what happens with removable drives.

Figure 23 Malware mounts Z drive

8. Once the malware runs, the following notification is displayed which hints that the malware might be employing an anti-VMware technique to check if it is running on a VMWare machine. Since it doesn't find VMWare Tools running, it proceeds to infect the machine.

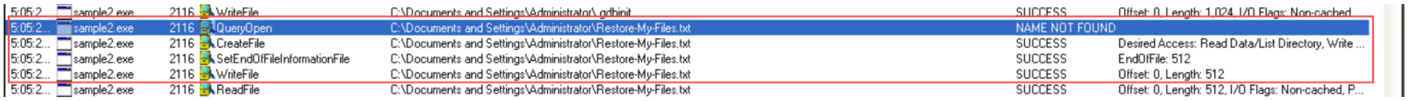


VMware Tools
The VMware Tools Service is not running.

Figure 24 Malware checks VMWare Tools Process

Process Monitor

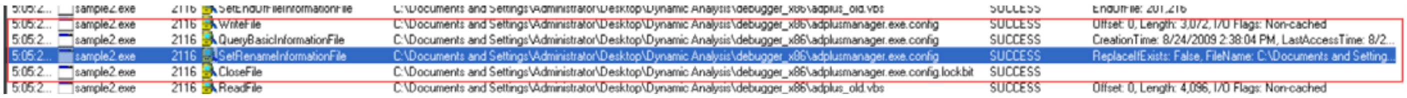
1. The malware recursively goes to every directory and tries to open a file called "Restore-My-Files.txt". If the file doesn't exist in the directory, the malware creates it and writes 512 bytes of content to it. Opening the file and inspecting the contents shows us that the system was infected by Lockbit 2.0 Ransomware. The malware authors provide details about how to get the data back and along with a decryption ID.



```
5:05:2...  sample2.exe  2116  WriteFile                    C:\Documents and Settings\Administrator\.gdbinit              SUCCESS         Offset: 0, Length: 1,024, I/O Flags: Non-cached
5:05:2...  sample2.exe  2116  QueryOpen                    C:\Documents and Settings\Administrator\Restore-My-Files.txt  NAME NOT FOUND
5:05:2...  sample2.exe  2116  CreateFile                   C:\Documents and Settings\Administrator\Restore-My-Files.txt  SUCCESS         Desired Access: Read Data/List Directory, Write ...
5:05:2...  sample2.exe  2116  SetEndOfFileInformationFile  C:\Documents and Settings\Administrator\Restore-My-Files.txt  SUCCESS         EndOfFile: 512
5:05:2...  sample2.exe  2116  WriteFile                    C:\Documents and Settings\Administrator\Restore-My-Files.txt  SUCCESS         Offset: 0, Length: 512
5:05:2...  sample2.exe  2116  ReadFile                     C:\Documents and Settings\Administrator\Restore-My-Files.txt  SUCCESS         Offset: 0, Length: 512, I/O Flags: Non-cached, P...
```

Figure 25 Creation of Restore-My-Files.txt

2. The malware renames the file and adds a .**lockbit** extension by using the *SetRenameInformationFile* function. A *WriteFile* call precedes the rename function call suggesting that this call might be responsible for encrypting the file contents.

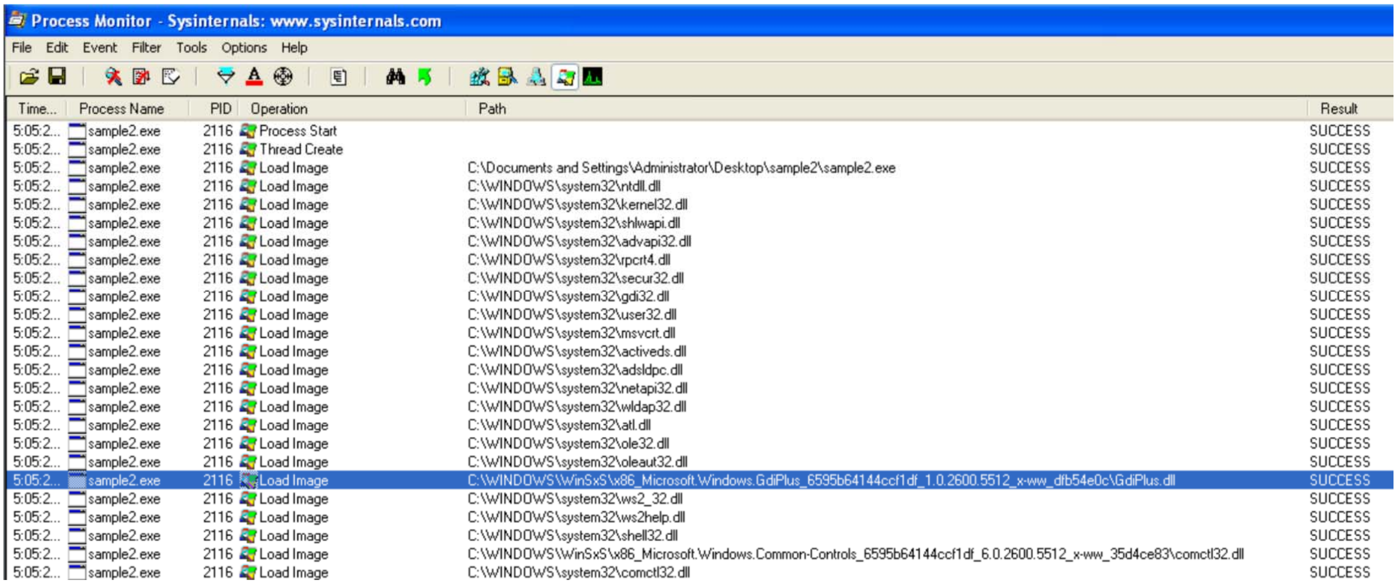


```
5:05:2...  sample2.exe  2116  SetEndOfFileInformationFile  C:\Documents and Settings\Administrator\Desktop\Dynamic Analysis\debugger_x86\adplus_old.vbs                  SUCCESS  EndOfFile: 201,216
5:05:2...  sample2.exe  2116  WriteFile                    C:\Documents and Settings\Administrator\Desktop\Dynamic Analysis\debugger_x86\adplusmanager.exe.config          SUCCESS  Offset: 0, Length: 3,072, I/O Flags: Non-cached
5:05:2...  sample2.exe  2116  QueryBasicInformationFile    C:\Documents and Settings\Administrator\Desktop\Dynamic Analysis\debugger_x86\adplusmanager.exe.config          SUCCESS  CreationTime: 8/24/2009 2:38:04 PM, LastAccessTime: 8/2...
5:05:2...  sample2.exe  2116  SetRenameInformationFile     C:\Documents and Settings\Administrator\Desktop\Dynamic Analysis\debugger_x86\adplusmanager.exe.config          SUCCESS  ReplaceIfExists: False, FileName: C:\Documents and Setting...
5:05:2...  sample2.exe  2116  CloseFile                    C:\Documents and Settings\Administrator\Desktop\Dynamic Analysis\debugger_x86\adplusmanager.exe.config.lockbit  SUCCESS
5:05:2...  sample2.exe  2116  ReadFile                     C:\Documents and Settings\Administrator\Desktop\Dynamic Analysis\debugger_x86\adplus_old.vbs                  SUCCESS  Offset: 0, Length: 4,096, I/O Flags: Non-cached
```

Figure 26 Changing the extension of the encrypted file

3. The malware loads the multiple DLLs at runtime. Some of the DLLs loaded by the malware are:
   - **Ntdll.dll** – This DLL contains Native API functions to access the kernel directly. Presence of this DLL is suspicious as legitimate programs don't use it.
   - **Gdi32.dll** – This library contains functions to draw figures and generate graphical images. The malware presumably uses this DLL's services to generate the wallpaper and icons.
   - **Shell32.dll** and **shlwapi.dll** – These DLLs contain Windows Shell manipulation functions which show that the malware changes GUI elements.
   - **Bcrypt.dll, crypt32.dll, cryptbase.dll** – The malware must be using functions from these DLLs to encrypt files on the machine.
   - **Netapi32.dll** – This DLL provides network management related functions for tasks like user, group and domain management. The malware might be using these to gain access to other computers in the network.
   - **Activeds**.dll – This DLL provides functions to work with the Active Directory. The malware uses functions ADsOpenObject and FreeADsMem which it might be using to manipulate Active Directory objects like users, groups and domains.
   - **kernel32.dll** – This is a standard DLL imported by almost all Windows programs to access Windows APIs related to file and memory manipulation.
   - **advapi32.dll** – The malware uses registry manipulation functions from this DLL.

- **User32.dll** – The malware uses *ActivateKeyBoardLayout* and *SendMessage* functions from this DLL. It also probably uses the CreateWindowEx function from this DLL to create a window to display the ransom message.
- **Ole32.dll, oleaut32.dll** – These DLLs provide functions to interact with COM objects. This indicates that the malware uses the Component Object Model (COM).
- **Ws2_32.dll** – The malware might be doing some network activity using the exports from this socket library.

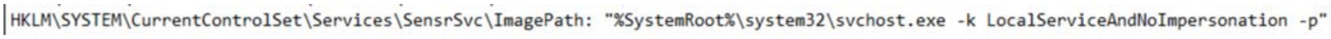

Process Monitor - Sysinternals: www.sysinternals.com
File Edit Event Filter Tools Options Help

Time...	Process Name	PID	Operation	Path	Result
5:05:2...	sample2.exe	2116	Process Start		SUCCESS
5:05:2...	sample2.exe	2116	Thread Create		SUCCESS
5:05:2...	sample2.exe	2116	Load Image	C:\Documents and Settings\Administrator\Desktop\sample2\sample2.exe	SUCCESS
5:05:2...	sample2.exe	2116	Load Image	C:\WINDOWS\system32\ntdll.dll	SUCCESS
5:05:2...	sample2.exe	2116	Load Image	C:\WINDOWS\system32\kernel32.dll	SUCCESS
5:05:2...	sample2.exe	2116	Load Image	C:\WINDOWS\system32\shlwapi.dll	SUCCESS
5:05:2...	sample2.exe	2116	Load Image	C:\WINDOWS\system32\advapi32.dll	SUCCESS
5:05:2...	sample2.exe	2116	Load Image	C:\WINDOWS\system32\rpcrt4.dll	SUCCESS
5:05:2...	sample2.exe	2116	Load Image	C:\WINDOWS\system32\secur32.dll	SUCCESS
5:05:2...	sample2.exe	2116	Load Image	C:\WINDOWS\system32\gdi32.dll	SUCCESS
5:05:2...	sample2.exe	2116	Load Image	C:\WINDOWS\system32\user32.dll	SUCCESS
5:05:2...	sample2.exe	2116	Load Image	C:\WINDOWS\system32\msvcrt.dll	SUCCESS
5:05:2...	sample2.exe	2116	Load Image	C:\WINDOWS\system32\activeds.dll	SUCCESS
5:05:2...	sample2.exe	2116	Load Image	C:\WINDOWS\system32\adsldpc.dll	SUCCESS
5:05:2...	sample2.exe	2116	Load Image	C:\WINDOWS\system32\netapi32.dll	SUCCESS
5:05:2...	sample2.exe	2116	Load Image	C:\WINDOWS\system32\wldap32.dll	SUCCESS
5:05:2...	sample2.exe	2116	Load Image	C:\WINDOWS\system32\atl.dll	SUCCESS
5:05:2...	sample2.exe	2116	Load Image	C:\WINDOWS\system32\ole32.dll	SUCCESS
5:05:2...	sample2.exe	2116	Load Image	C:\WINDOWS\system32\oleaut32.dll	SUCCESS
5:05:2...	sample2.exe	2116	Load Image	C:\WINDOWS\WinSxS\x86_Microsoft.Windows.GdiPlus_6595b64144ccf1df_1.0.2600.5512_x-ww_dfb54e0c\GdiPlus.dll	SUCCESS
5:05:2...	sample2.exe	2116	Load Image	C:\WINDOWS\system32\ws2_32.dll	SUCCESS
5:05:2...	sample2.exe	2116	Load Image	C:\WINDOWS\system32\ws2help.dll	SUCCESS
5:05:2...	sample2.exe	2116	Load Image	C:\WINDOWS\system32\shell32.dll	SUCCESS
5:05:2...	sample2.exe	2116	Load Image	C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.2600.5512_x-ww_35d4ce83\comctl32.dll	SUCCESS
5:05:2...	sample2.exe	2116	Load Image	C:\WINDOWS\system32\comctl32.dll	SUCCESS

Figure 27 DLLs loaded by the malware

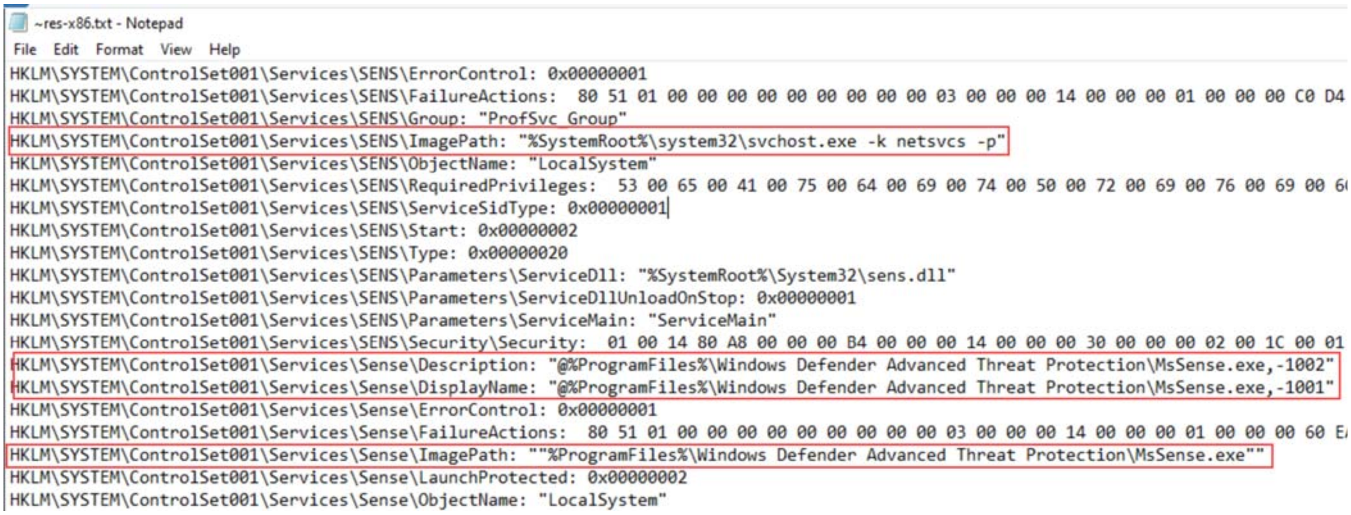
Regshot

- The malware encrypts all files including .hiv files generated by RegShot. The workaround for this is to rename the file containing the first shot with a .exe or .dll extension before running the malware and changing the extension back to the original one once the malware finishes executing. This method works as the malware doesn’t encrypt PE files.



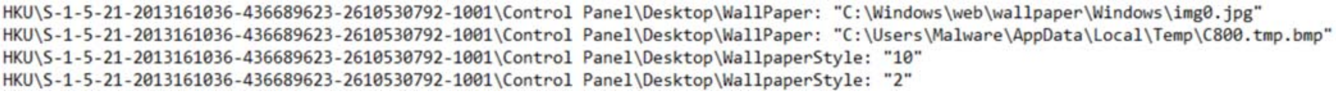
```
HKLM\SYSTEM\CurrentControlSet\Services\SensrSvc\ImagePath: "%SystemRoot%\system32\svchost.exe -k LocalServiceAndNoImpersonation -p"
```

- The malware edits registry keys related to Windows Defender probably to avoid detection.



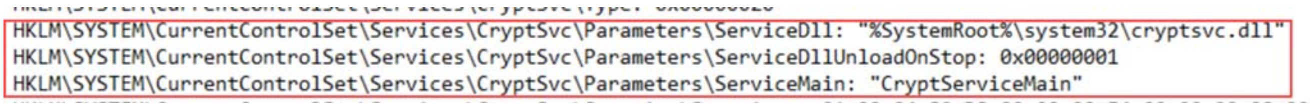
```
HKLM\SYSTEM\ControlSet001\Services\SENS\ErrorControl: 0x00000001
HKLM\SYSTEM\ControlSet001\Services\SENS\FailureActions:  80 51 01 00 00 00 00 00 00 00 00 00 03 00 00 00 14 00 00 00 01 00 00 00 C0 D4
HKLM\SYSTEM\ControlSet001\Services\SENS\Group: "ProfSvc_Group"
HKLM\SYSTEM\ControlSet001\Services\SENS\ImagePath: "%SystemRoot%\system32\svchost.exe -k netsvcs -p"
HKLM\SYSTEM\ControlSet001\Services\SENS\ObjectName: "LocalSystem"
HKLM\SYSTEM\ControlSet001\Services\SENS\RequiredPrivileges:  53 00 65 00 41 00 75 00 64 00 69 00 74 00 50 00 72 00 69 00 76 00 69 00 6
HKLM\SYSTEM\ControlSet001\Services\SENS\ServiceSidType: 0x00000001
HKLM\SYSTEM\ControlSet001\Services\SENS\Start: 0x00000002
HKLM\SYSTEM\ControlSet001\Services\SENS\Type: 0x00000020
HKLM\SYSTEM\ControlSet001\Services\SENS\Parameters\ServiceDll: "%SystemRoot%\System32\sens.dll"
HKLM\SYSTEM\ControlSet001\Services\SENS\Parameters\ServiceDllUnloadOnStop: 0x00000001
HKLM\SYSTEM\ControlSet001\Services\SENS\Parameters\ServiceMain: "ServiceMain"
HKLM\SYSTEM\ControlSet001\Services\SENS\Security\Security:  01 00 14 80 A8 00 00 00 B4 00 00 00 14 00 00 00 30 00 00 00 02 00 1C 00 01
HKLM\SYSTEM\ControlSet001\Services\Sense\Description: "@%ProgramFiles%\Windows Defender Advanced Threat Protection\MsSense.exe,-1002"
HKLM\SYSTEM\ControlSet001\Services\Sense\DisplayName: "@%ProgramFiles%\Windows Defender Advanced Threat Protection\MsSense.exe,-1001"
HKLM\SYSTEM\ControlSet001\Services\Sense\ErrorControl: 0x00000001
HKLM\SYSTEM\ControlSet001\Services\Sense\FailureActions:  80 51 01 00 00 00 00 00 00 00 00 00 03 00 00 00 14 00 00 00 01 00 00 00 60 E
HKLM\SYSTEM\ControlSet001\Services\Sense\ImagePath: ""%ProgramFiles%\Windows Defender Advanced Threat Protection\MsSense.exe""
HKLM\SYSTEM\ControlSet001\Services\Sense\LaunchProtected: 0x00000002
HKLM\SYSTEM\ControlSet001\Services\Sense\ObjectName: "LocalSystem"
```

- Registry keys related to desktop wallpaper are also modified by the malware. This might be how the malware sets the new wallpaper.



```
HKU\S-1-5-21-2013161036-436689623-2610530792-1001\Control Panel\Desktop\WallPaper: "C:\Windows\web\wallpaper\Windows\img0.jpg"
HKU\S-1-5-21-2013161036-436689623-2610530792-1001\Control Panel\Desktop\WallPaper: "C:\Users\Malware\AppData\Local\Temp\C800.tmp.bmp"
HKU\S-1-5-21-2013161036-436689623-2610530792-1001\Control Panel\Desktop\WallpaperStyle: "10"
HKU\S-1-5-21-2013161036-436689623-2610530792-1001\Control Panel\Desktop\WallpaperStyle: "2"
```

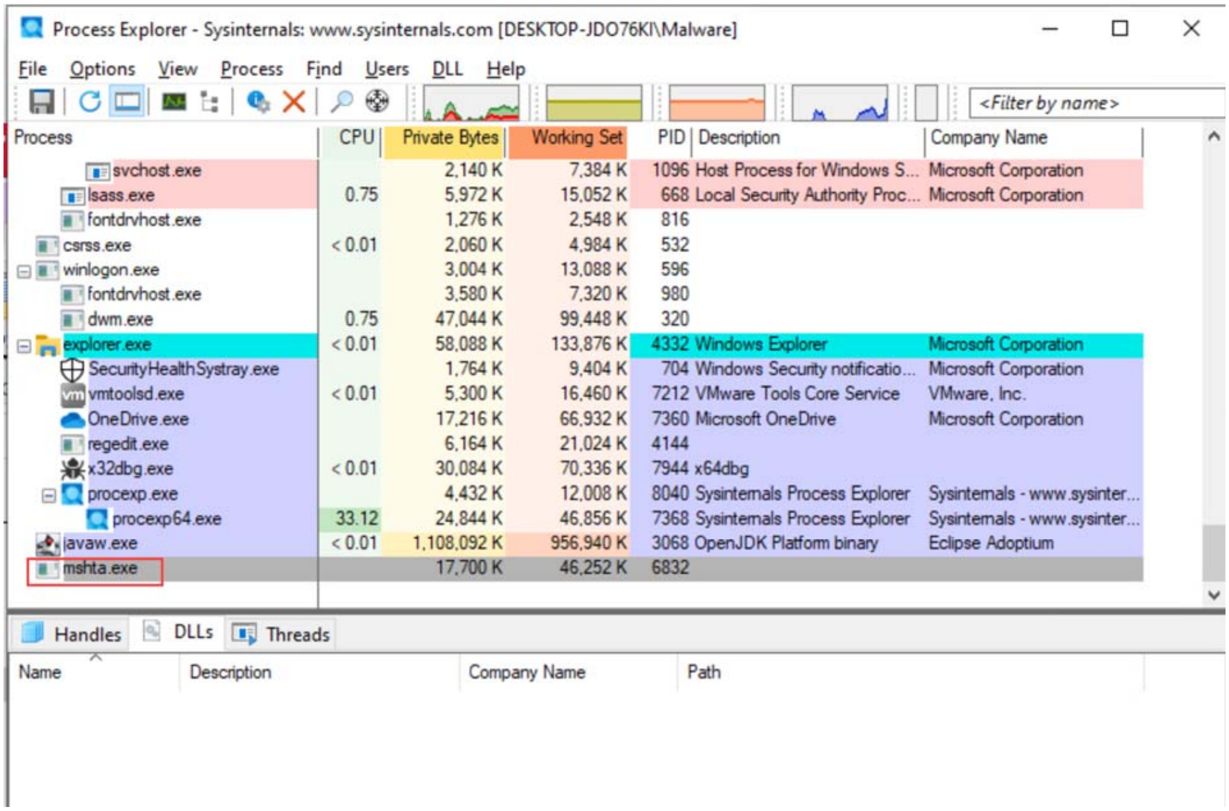
- The malware seems to be installing cryptsvc.dll as a service.



```
HKLM\SYSTEM\CurrentControlSet\Services\CryptSvc\Parameters\ServiceDll: "%SystemRoot%\system32\cryptsvc.dll"
HKLM\SYSTEM\CurrentControlSet\Services\CryptSvc\Parameters\ServiceDllUnloadOnStop: 0x00000001
HKLM\SYSTEM\CurrentControlSet\Services\CryptSvc\Parameters\ServiceMain: "CryptServiceMain"
```

Process Explorer

- The malware kills the Process Explorer process immediately after executing on Windows 7 and 10. On WindowsXP, it takes about a minute to execute and kill Process Explorer.
- Once the malware executes, it runs LockBit_Ransomware.hta as an mshta.exe process. Analyzing this process in Process Explorer shows that it doesn't use any DLLs and doesn't have any handles.



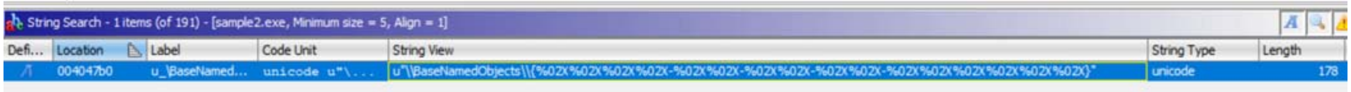
Process Explorer - Sysinternals: www.sysinternals.com [DESKTOP-JDO76KI\Malware]

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
svchost.exe		2,140 K	7,384 K	1096	Host Process for Windows S...	Microsoft Corporation
lsass.exe	0.75	5,972 K	15,052 K	668	Local Security Authority Proc...	Microsoft Corporation
fontdrvhost.exe		1,276 K	2,548 K	816		
csrss.exe	< 0.01	2,060 K	4,984 K	532		
winlogon.exe		3,004 K	13,088 K	596		
fontdrvhost.exe		3,580 K	7,320 K	980		
dwm.exe	0.75	47,044 K	99,448 K	320		
explorer.exe	< 0.01	58,088 K	133,876 K	4332	Windows Explorer	Microsoft Corporation
SecurityHealthSystray.exe		1,764 K	9,404 K	704	Windows Security notificatio...	Microsoft Corporation
vmtoolsd.exe	< 0.01	5,300 K	16,460 K	7212	VMware Tools Core Service	VMware, Inc.
OneDrive.exe		17,216 K	66,932 K	7360	Microsoft OneDrive	Microsoft Corporation
regedit.exe		6,164 K	21,024 K	4144		
x32dbg.exe	< 0.01	30,084 K	70,336 K	7944	x64dbg	
procexp.exe		4,432 K	12,008 K	8040	Sysinternals Process Explorer	Sysinternals - www.sysinter...
procexp64.exe	33.12	24,844 K	46,856 K	7368	Sysinternals Process Explorer	Sysinternals - www.sysinter...
javaw.exe	< 0.01	1,108,092 K	956,940 K	3068	OpenJDK Platform binary	Eclipse Adoptium
mshta.exe		17,700 K	46,252 K	6832		

Name	Description	Company Name	Path

Figure 28 LockBit_Ransomware.hta as an mshta.exe process

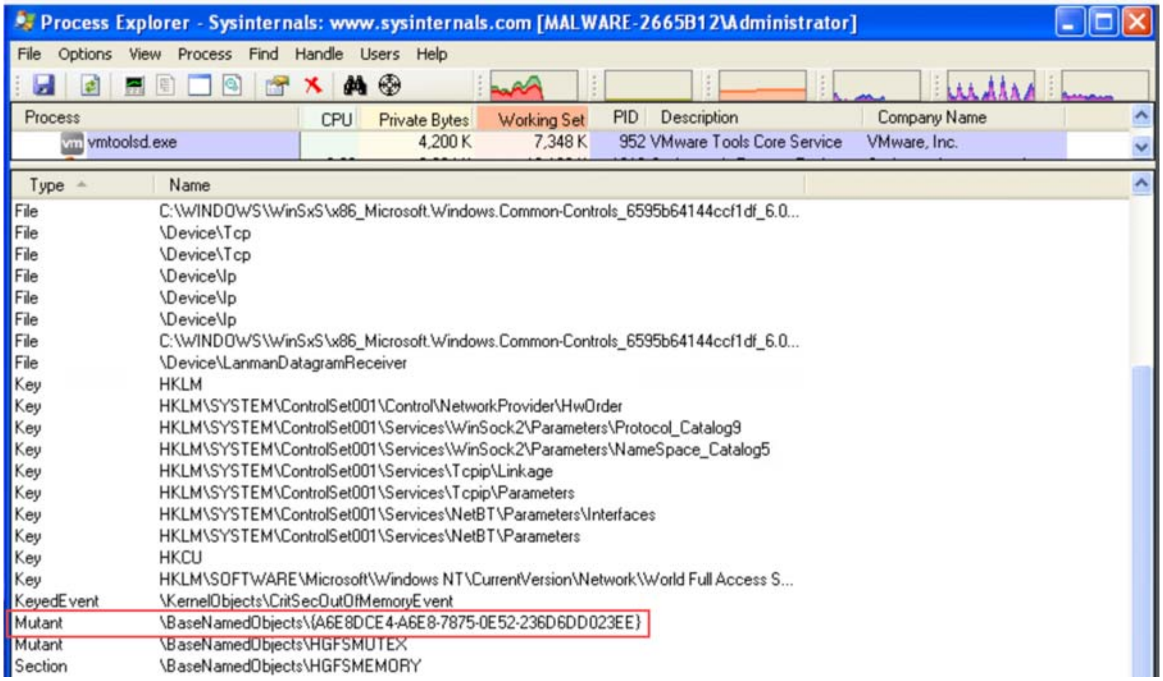
- **Mutant:** The malware creates a mutant which might be being used to ensure that only one instance of malware is running at a time.
    - The name of the mutant appears as a formatted string when looking for strings during static analysis.
    - At runtime, this formatted string always gets populated with the same values each time to generate the string **"\BaseNamedObjects\{A6E8DCE4-A6E8-7875-0E52-0E52-236D6DD023EE}"** which is the name of the mutant as can be seen in Process Explorer.



String Search - 1 items (of 191) - [sample2.exe, Minimum size = 5, Align = 1]

Defi...	Location	Label	Code Unit	String View	String Type	Length
	004047b0	u_\BaseNamed...	unicode u"\...	u"\\BaseNamedObjects\\{%02X%02X%02X%02X-%02X%02X-%02X%02X-%02X%02X-%02X%02X%02X%02X%02X%02X}"	unicode	178

Figure 29 Name of the mutant as a formatted string


Process Explorer - Sysinternals: www.sysinternals.com [MALWARE-2665B12\Administrator]
File
Options
View
Process
Find
Handle
Users
Help

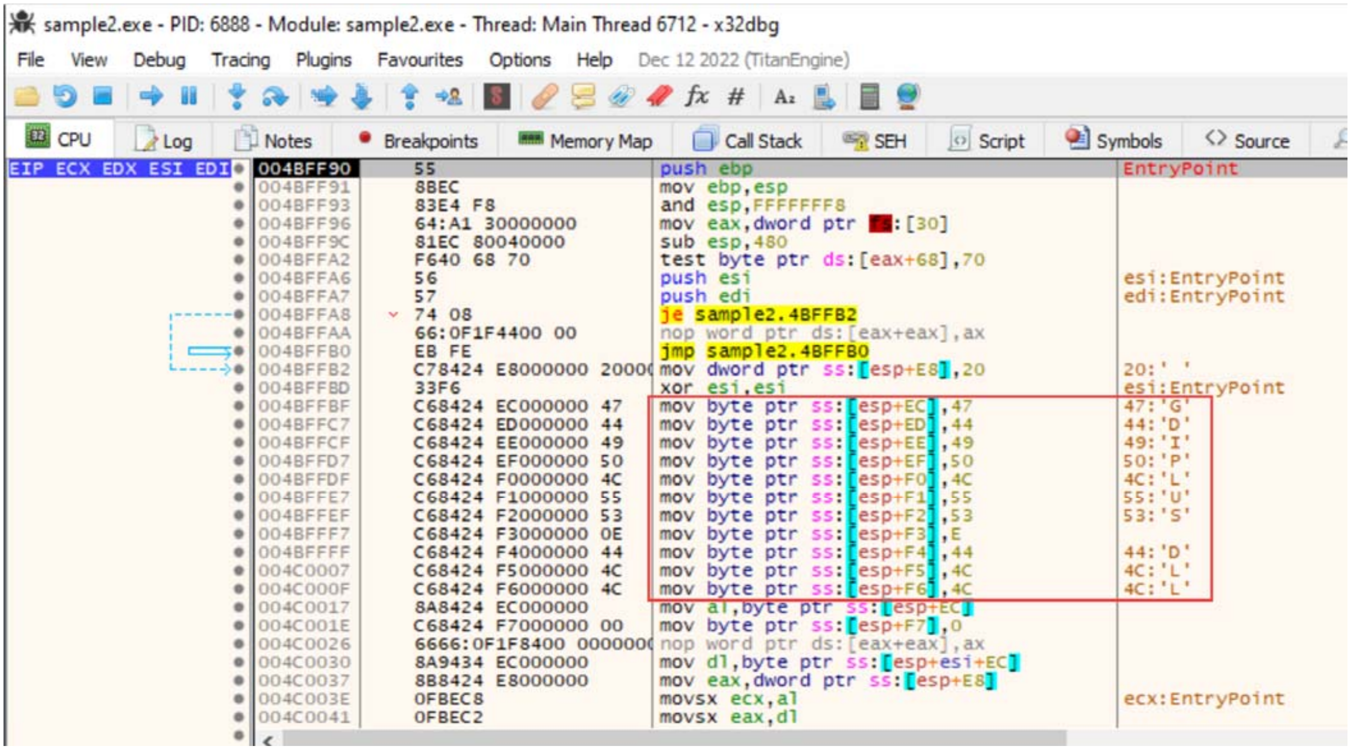
Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
vmtoolsd.exe		4,200 K	7,348 K	952	VMware Tools Core Service	VMware, Inc.

Type	Name
File	C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0...
File	\Device\Tcp
File	\Device\Tcp
File	\Device\Ip
File	\Device\Ip
File	\Device\Ip
File	C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0...
File	\Device\LanmanDatagramReceiver
Key	HKLM
Key	HKLM\SYSTEM\ControlSet001\Control\NetworkProvider\HwOrder
Key	HKLM\SYSTEM\ControlSet001\Services\WinSock2\Parameters\Protocol_Catalog9
Key	HKLM\SYSTEM\ControlSet001\Services\WinSock2\Parameters\NameSpace_Catalog5
Key	HKLM\SYSTEM\ControlSet001\Services\Tcpip\Linkage
Key	HKLM\SYSTEM\ControlSet001\Services\Tcpip\Parameters
Key	HKLM\SYSTEM\ControlSet001\Services\NetBT\Parameters\Interfaces
Key	HKLM\SYSTEM\ControlSet001\Services\NetBT\Parameters
Key	HKCU
Key	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Network\World Full Access S...
KeyedEvent	\KernelObjects\CritSecOutOfMemoryEvent
Mutant	\BaseNamedObjects\{A6E8DCE4-A6E8-7875-0E52-236D6DD023EE}
Mutant	\BaseNamedObjects\HGFSMUTEX
Section	\BaseNamedObjects\HGFSMEMORY

Figure 30 Mutant created by malware

X32dbg

1. The malware stores the names of all the DLLs it loads dynamically in the code of the entry function.

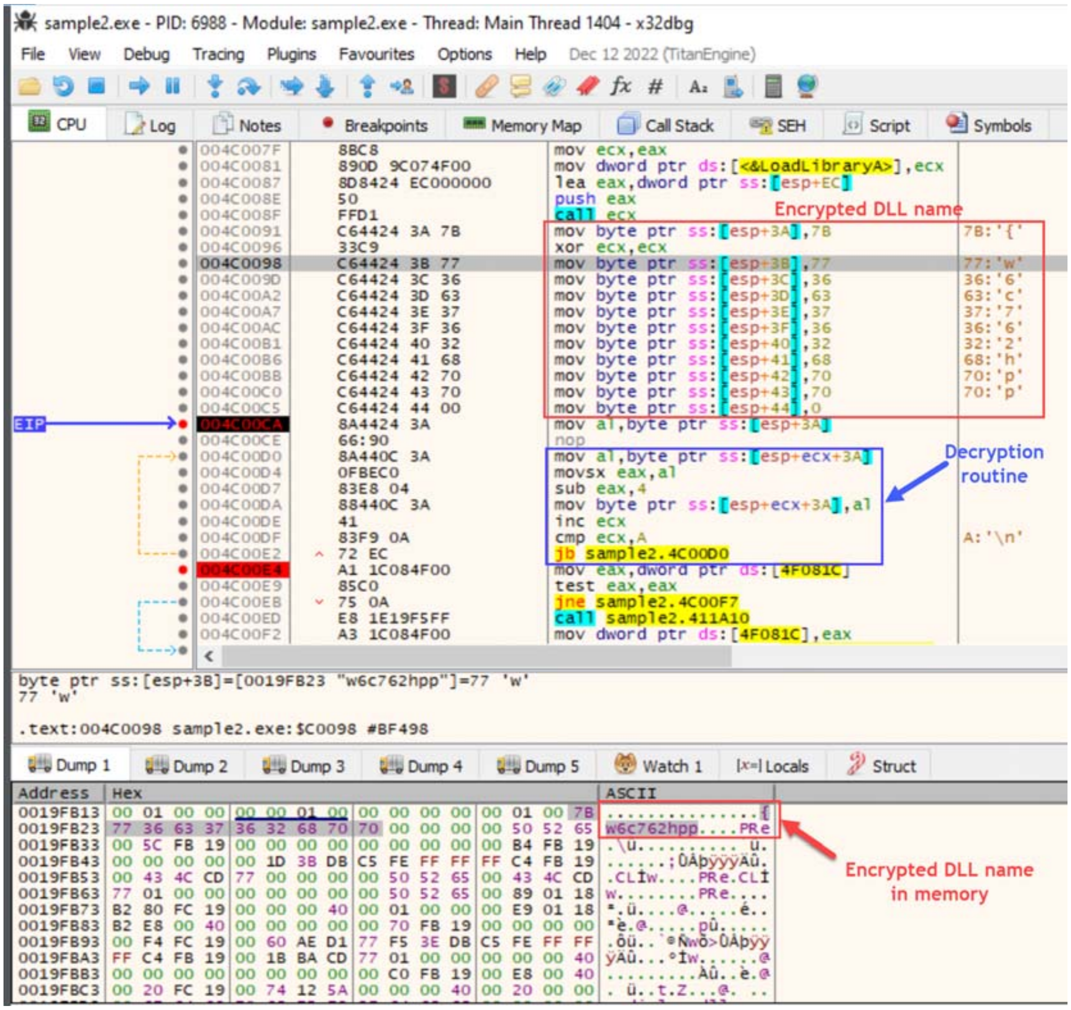


sample2.exe - PID: 6888 - Module: sample2.exe - Thread: Main Thread 6712 - x32dbg
File View Debug Tracing Plugins Favourites Options Help Dec 12 2022 (TitanEngine)
CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source
EIP ECX EDX ESI EDI

```
004BFF90    55                      push ebp                            EntryPoint
004BFF91    8BEC                    mov ebp,esp
004BFF93    83E4 F8                 and esp,FFFFFFF8
004BFF96    64:A1 30000000          mov eax,dword ptr fs:[30]
004BFF9C    81EC 80040000           sub esp,480
004BFFA2    F640 68 70              test byte ptr ds:[eax+68],70
004BFFA6    56                      push esi                            esi:EntryPoint
004BFFA7    57                      push edi                            edi:EntryPoint
004BFFA8    74 08                   je sample2.4BFFB2
004BFFAA    66:0F1F4400 00          nop word ptr ds:[eax+eax],ax
004BFFB0    EB FE                   jmp sample2.4BFFB0
004BFFB2    C78424 E8000000 2000(   mov dword ptr ss:[esp+E8],20        20:' '
004BFFBD    33F6                    xor esi,esi                         esi:EntryPoint
004BFFBF    C68424 EC000000 47      mov byte ptr ss:[esp+EC],47         47:'G'
004BFFC7    C68424 ED000000 44      mov byte ptr ss:[esp+ED],44         44:'D'
004BFFCF    C68424 EE000000 49      mov byte ptr ss:[esp+EE],49         49:'I'
004BFFD7    C68424 EF000000 50      mov byte ptr ss:[esp+EF],50         50:'P'
004BFFDF    C68424 F0000000 4C      mov byte ptr ss:[esp+F0],4C         4C:'L'
004BFFE7    C68424 F1000000 55      mov byte ptr ss:[esp+F1],55         55:'U'
004BFFEF    C68424 F2000000 53      mov byte ptr ss:[esp+F2],53         53:'S'
004BFFF7    C68424 F3000000 0E      mov byte ptr ss:[esp+F3],E
004BFFFF    C68424 F4000000 44      mov byte ptr ss:[esp+F4],44         44:'D'
004C0007    C68424 F5000000 4C      mov byte ptr ss:[esp+F5],4C         4C:'L'
004C000F    C68424 F6000000 4C      mov byte ptr ss:[esp+F6],4C         4C:'L'
004C0017    8A8424 EC000000         mov al,byte ptr ss:[esp+EC]
004C001E    C68424 F7000000 00      mov byte ptr ss:[esp+F7],0
004C0026    6666:0F1F8400 000000(   nop word ptr ds:[eax+eax],ax
004C0030    8A9434 EC000000         mov dl,byte ptr ss:[esp+esi+EC]
004C0037    8B8424 E8000000         mov eax,dword ptr ss:[esp+E8]
004C003E    0FBEC8                  movsx ecx,al                        ecx:EntryPoint
004C0041    0FBEC2                  movsx eax,dl
```

Figure 31 DLL names stored in code

2. Some DLL names are hard-coded in the program as encrypted values. These are decrypted at runtime by a convoluted decryption routine. For example, as shown in the figures below, ws2_32.dll’s name is stored as encrypted characters in the program which gets decrypted at runtime.



```
004C007F   8BC8             mov ecx,eax
004C0081   890D 9C074F00    mov dword ptr ds:[<&LoadLibraryA>],ecx
004C0087   8D8424 EC000000  lea eax,dword ptr ss:[esp+EC]
004C008E   50               push eax
004C008F   FFD1             call ecx
004C0091   C64424 3A 7B     mov byte ptr ss:[esp+3A],7B         7B:'{'
004C0096   33C9             xor ecx,ecx
004C0098   C64424 3B 77     mov byte ptr ss:[esp+3B],77         77:'w'
004C009D   C64424 3C 36     mov byte ptr ss:[esp+3C],36         36:'6'
004C00A2   C64424 3D 63     mov byte ptr ss:[esp+3D],63         63:'c'
004C00A7   C64424 3E 37     mov byte ptr ss:[esp+3E],37         37:'7'
004C00AC   C64424 3F 36     mov byte ptr ss:[esp+3F],36         36:'6'
004C00B1   C64424 40 32     mov byte ptr ss:[esp+40],32         32:'2'
004C00B6   C64424 41 68     mov byte ptr ss:[esp+41],68         68:'h'
004C00BB   C64424 42 70     mov byte ptr ss:[esp+42],70         70:'p'
004C00C0   C64424 43 70     mov byte ptr ss:[esp+43],70         70:'p'
004C00C5   C64424 44 00     mov byte ptr ss:[esp+44],0
004C00CA   8A4424 3A        mov al,byte ptr ss:[esp+3A]
004C00CE   66:90            nop
004C00D0   8A440C 3A        mov al,byte ptr ss:[esp+ecx+3A]
004C00D4   0FBEC0           movsx eax,al
004C00D7   83E8 04          sub eax,4
004C00DA   88440C 3A        mov byte ptr ss:[esp+ecx+3A],al
004C00DE   41               inc ecx
004C00DF   83F9 0A          cmp ecx,A                           A:'\n'
004C00E2 ^ 72 EC            jb sample2.4C00D0
004C00E4   A1 1C084F00      mov eax,dword ptr ds:[4F081C]
004C00E9   85C0             test eax,eax
004C00EB v 75 0A            jne sample2.4C00F7
004C00ED   E8 1E19F5FF      call sample2.411A10
004C00F2   A3 1C084F00      mov dword ptr ds:[4F081C],eax
```

Encrypted DLL name

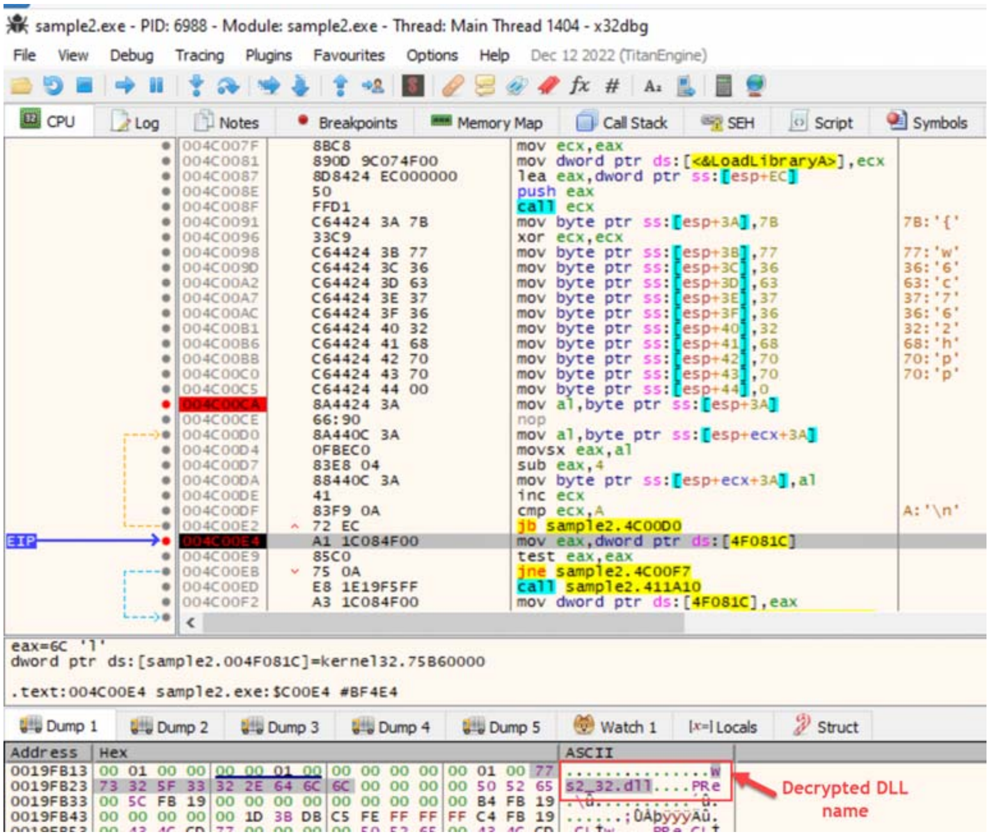
Decryption routine

```
byte ptr ss:[esp+3B]=[0019FB23 "w6c762hpp"]=77 'w'
77 'w'

.text:004C0098 sample2.exe:$C0098 #BF498
```

```
Address   Hex                                              ASCII
0019FB13  00 01 00 00 00 00 01 00 00 00 00 00 00 01 00 7B  ...............{
0019FB23  77 36 63 37 36 32 68 70 70 00 00 00 00 50 52 65  w6c762hpp....PRe
0019FB33  00 5C FB 19 00 00 00 00 00 00 00 00 00 B4 FB 19  .\û..........´û.
0019FB43  00 00 00 00 00 1D 3B DB C5 FE FF FF FF C4 FB 19  ......;ÛÅþÿÿÿÄû.
0019FB53  00 43 4C CD 77 00 00 00 00 50 52 65 00 43 4C CD  .CLÍw....PRe.CLÍ
0019FB63  77 01 00 00 00 00 00 00 00 50 52 65 00 89 01 18  w........PRe....
0019FB73  B2 80 FC 19 00 00 00 40 00 01 00 00 00 E9 01 18  ².ü....@.....é..
0019FB83  B2 E8 00 40 00 00 00 00 00 70 FB 19 00 00 00 00  ²è.@.....pû.....
0019FB93  00 F4 FC 19 00 60 AE D1 77 F5 3E DB C5 FE FF FF  .ôü..`®Ñwõ>ÛÅþÿÿ
0019FBA3  FF C4 FB 19 00 1B BA CD 77 01 00 00 00 00 00 40  ÿÄû...ºÍw......@
0019FBB3  00 00 00 00 00 00 00 00 00 C0 FB 19 00 E8 00 40  .........Àû..è.@
0019FBC3  00 20 FC 19 00 74 12 5A 00 00 00 40 00 20 00 00  . ü..t.Z...@. ..
```

Encrypted DLL name in memory

Figure 32 Encrypted DLL name and decryption routine



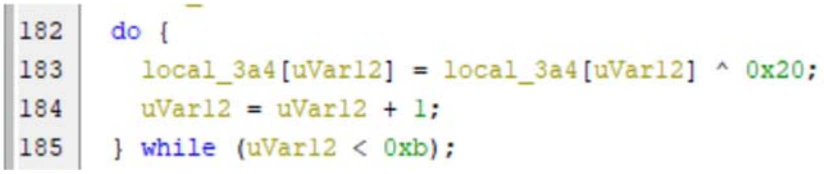
sample2.exe - PID: 6988 - Module: sample2.exe - Thread: Main Thread 1404 - x32dbg
File View Debug Tracing Plugins Favourites Options Help Dec 12 2022 (TitanEngine)
CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols

```
004C007F   8BC8              mov ecx,eax
004C0081   890D 9C074F00     mov dword ptr ds:[<&LoadLibraryA>],ecx
004C0087   8D8424 EC000000   lea eax,dword ptr ss:[esp+EC]
004C008E   50                push eax
004C008F   FFD1              call ecx
004C0091   C64424 3A 7B      mov byte ptr ss:[esp+3A],7B          7B:'{'
004C0096   33C9              xor ecx,ecx
004C0098   C64424 3B 77      mov byte ptr ss:[esp+3B],77          77:'w'
004C009D   C64424 3C 36      mov byte ptr ss:[esp+3C],36          36:'6'
004C00A2   C64424 3D 63      mov byte ptr ss:[esp+3D],63          63:'c'
004C00A7   C64424 3E 37      mov byte ptr ss:[esp+3E],37          37:'7'
004C00AC   C64424 3F 36      mov byte ptr ss:[esp+3F],36          36:'6'
004C00B1   C64424 40 32      mov byte ptr ss:[esp+40],32          32:'2'
004C00B6   C64424 41 68      mov byte ptr ss:[esp+41],68          68:'h'
004C00BB   C64424 42 70      mov byte ptr ss:[esp+42],70          70:'p'
004C00C0   C64424 43 70      mov byte ptr ss:[esp+43],70          70:'p'
004C00C5   C64424 44 00      mov byte ptr ss:[esp+44],0
004C00CA   8A4424 3A         mov al,byte ptr ss:[esp+3A]
004C00CE   66:90             nop
004C00D0   8A440C 3A         mov al,byte ptr ss:[esp+ecx+3A]
004C00D4   0FBEC0            movsx eax,al
004C00D7   83E8 04           sub eax,4
004C00DA   88440C 3A         mov byte ptr ss:[esp+ecx+3A],al
004C00DE   41                inc ecx
004C00DF   83F9 0A           cmp ecx,A                            A:'\n'
004C00E2 ^ 72 EC             jb sample2.4C00D0
EIP 004C00E4   A1 1C084F00   mov eax,dword ptr ds:[4F081C]
004C00E9   85C0              test eax,eax
004C00EB v 75 0A             jne sample2.4C00F7
004C00ED   E8 1E19F5FF       call sample2.411A10
004C00F2   A3 1C084F00       mov dword ptr ds:[4F081C],eax

eax=6C 'l'
dword ptr ds:[sample2.004F081C]=kernel32.75B60000

.text:004C00E4 sample2.exe:$C00E4 #BF4E4
```

```
Address   Hex                                              ASCII
0019FB13  00 01 00 00 00 00 01 00 00 00 00 00 00 01 00 77  ...............w
0019FB23  73 32 5F 33 32 2E 64 6C 6C 00 00 00 00 50 52 65  s2_32.dll....PRe
0019FB33  00 5C FB 19 00 00 00 00 00 00 00 00 00 B4 FB 19  .\û.........´û.
0019FB43  00 00 00 00 00 1D 3B DB C5 FE FF FF FF C4 FB 19  ......;ÛÅþÿÿÿÄû.
```

Figure 33 Decrypted DLL name

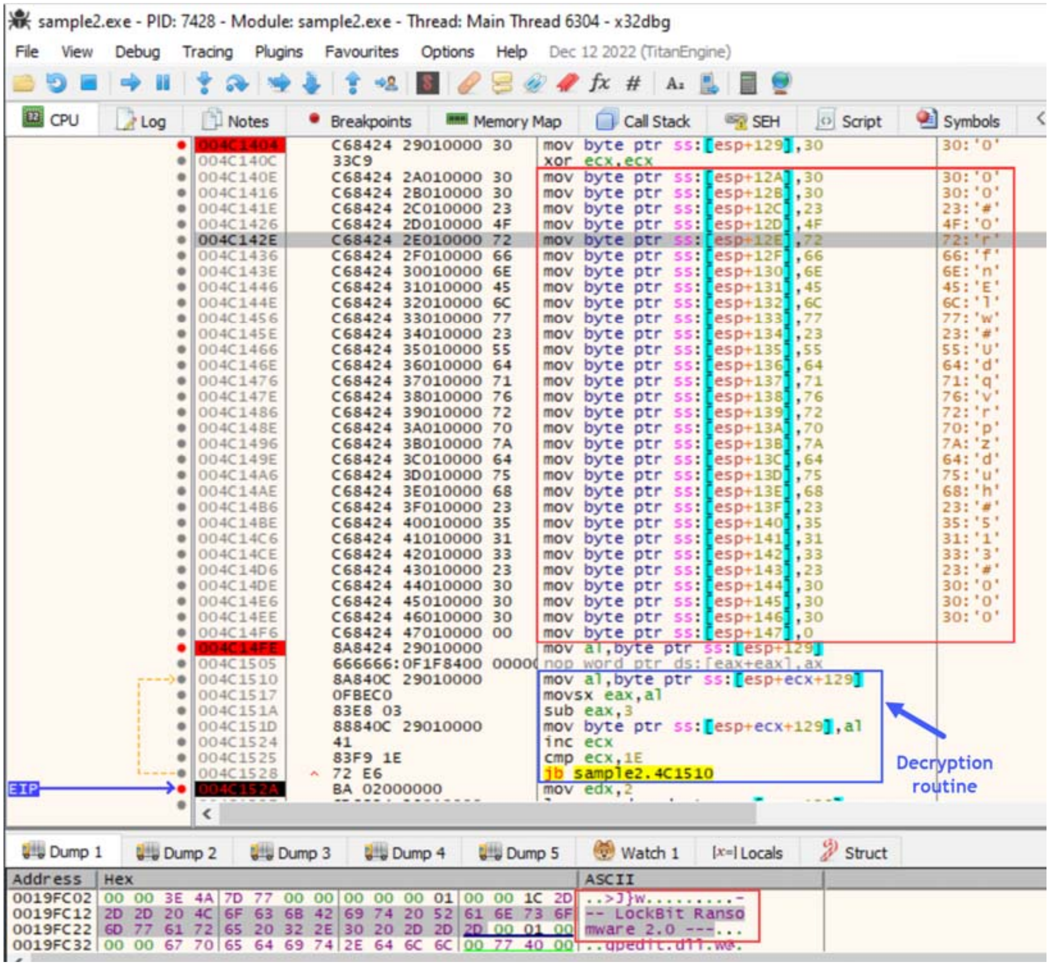
3. **Changing character case:** While single-stepping through the program, I found a cool trick the malware uses to convert lowercase characters to uppercase (and vice-versa) by XORing the character's ASCII value with 0x20 (ASCII value of space).



```
do {
  local_3a4[uVar12] = local_3a4[uVar12] ^ 0x20;
  uVar12 = uVar12 + 1;
} while (uVar12 < 0xb);
```

Figure 34 Converting characters to a different case

4. **Lockbit string:** The name of the ransomware, *"LockBit Ransomware 2.0"* is stored as an encrypted string in the code and is decrypted at runtime by non-standard decryption routine which decrypts each character in a loop.

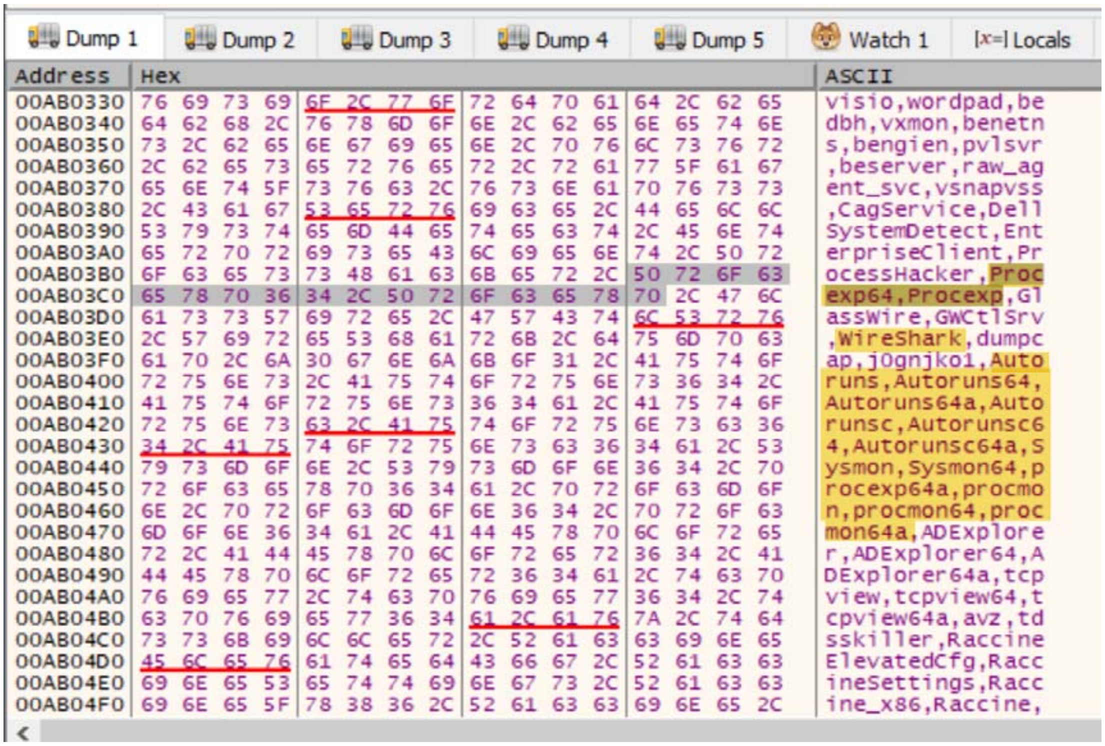


```
004C1404   C68424 29010000 30    mov byte ptr ss:[esp+129],30     30:'0'
004C140C   33C9                  xor ecx,ecx
004C140E   C68424 2A010000 30    mov byte ptr ss:[esp+12A],30     30:'0'
004C1416   C68424 2B010000 30    mov byte ptr ss:[esp+12B],30     30:'0'
004C141E   C68424 2C010000 23    mov byte ptr ss:[esp+12C],23     23:'#'
004C1426   C68424 2D010000 4F    mov byte ptr ss:[esp+12D],4F     4F:'O'
004C142E   C68424 2E010000 72    mov byte ptr ss:[esp+12E],72     72:'r'
004C1436   C68424 2F010000 66    mov byte ptr ss:[esp+12F],66     66:'f'
004C143E   C68424 30010000 6E    mov byte ptr ss:[esp+130],6E     6E:'n'
004C1446   C68424 31010000 45    mov byte ptr ss:[esp+131],45     45:'E'
004C144E   C68424 32010000 6C    mov byte ptr ss:[esp+132],6C     6C:'l'
004C1456   C68424 33010000 77    mov byte ptr ss:[esp+133],77     77:'w'
004C145E   C68424 34010000 23    mov byte ptr ss:[esp+134],23     23:'#'
004C1466   C68424 35010000 55    mov byte ptr ss:[esp+135],55     55:'U'
004C146E   C68424 36010000 64    mov byte ptr ss:[esp+136],64     64:'d'
004C1476   C68424 37010000 71    mov byte ptr ss:[esp+137],71     71:'q'
004C147E   C68424 38010000 76    mov byte ptr ss:[esp+138],76     76:'v'
004C1486   C68424 39010000 72    mov byte ptr ss:[esp+139],72     72:'r'
004C148E   C68424 3A010000 70    mov byte ptr ss:[esp+13A],70     70:'p'
004C1496   C68424 3B010000 7A    mov byte ptr ss:[esp+13B],7A     7A:'z'
004C149E   C68424 3C010000 64    mov byte ptr ss:[esp+13C],64     64:'d'
004C14A6   C68424 3D010000 75    mov byte ptr ss:[esp+13D],75     75:'u'
004C14AE   C68424 3E010000 68    mov byte ptr ss:[esp+13E],68     68:'h'
004C14B6   C68424 3F010000 23    mov byte ptr ss:[esp+13F],23     23:'#'
004C14BE   C68424 40010000 35    mov byte ptr ss:[esp+140],35     35:'5'
004C14C6   C68424 41010000 31    mov byte ptr ss:[esp+141],31     31:'1'
004C14CE   C68424 42010000 33    mov byte ptr ss:[esp+142],33     33:'3'
004C14D6   C68424 43010000 23    mov byte ptr ss:[esp+143],23     23:'#'
004C14DE   C68424 44010000 30    mov byte ptr ss:[esp+144],30     30:'0'
004C14E6   C68424 45010000 30    mov byte ptr ss:[esp+145],30     30:'0'
004C14EE   C68424 46010000 30    mov byte ptr ss:[esp+146],30     30:'0'
004C14F6   C68424 47010000 00    mov byte ptr ss:[esp+147],0
004C14FE   8A8424 29010000       mov al,byte ptr ss:[esp+129]
004C1505   666666:0F1F8400 0000( nop word ptr ds:[eax+eax],ax
004C1510   8A840C 29010000       mov al,byte ptr ss:[esp+ecx+129]
004C1517   0FBEC0                movsx eax,al
004C151A   83E8 03               sub eax,3
004C151D   88840C 29010000       mov byte ptr ss:[esp+ecx+129],al
004C1524   41                    inc ecx
004C1525   83F9 1E               cmp ecx,1E
004C1528 ^ 72 E6                 jb sample2.4C1510
004C152A   BA 02000000           mov edx,2
```

```
Address   Hex                                              ASCII
0019FC02  00 00 3E 4A 7D 77 00 00 00 00 00 01 00 00 1C 2D  ..>J}w.........-
0019FC12  2D 2D 20 4C 6F 63 6B 42 69 74 20 52 61 6E 73 6F  -- LockBit Ranso
0019FC22  6D 77 61 72 65 20 32 2E 30 20 2D 2D 2D 00 01 00  mware 2.0 ---...
0019FC32  00 00 67 70 65 64 69 74 2E 64 6C 6C 00 77 40 00  ..gpedit.dll.w@.
```

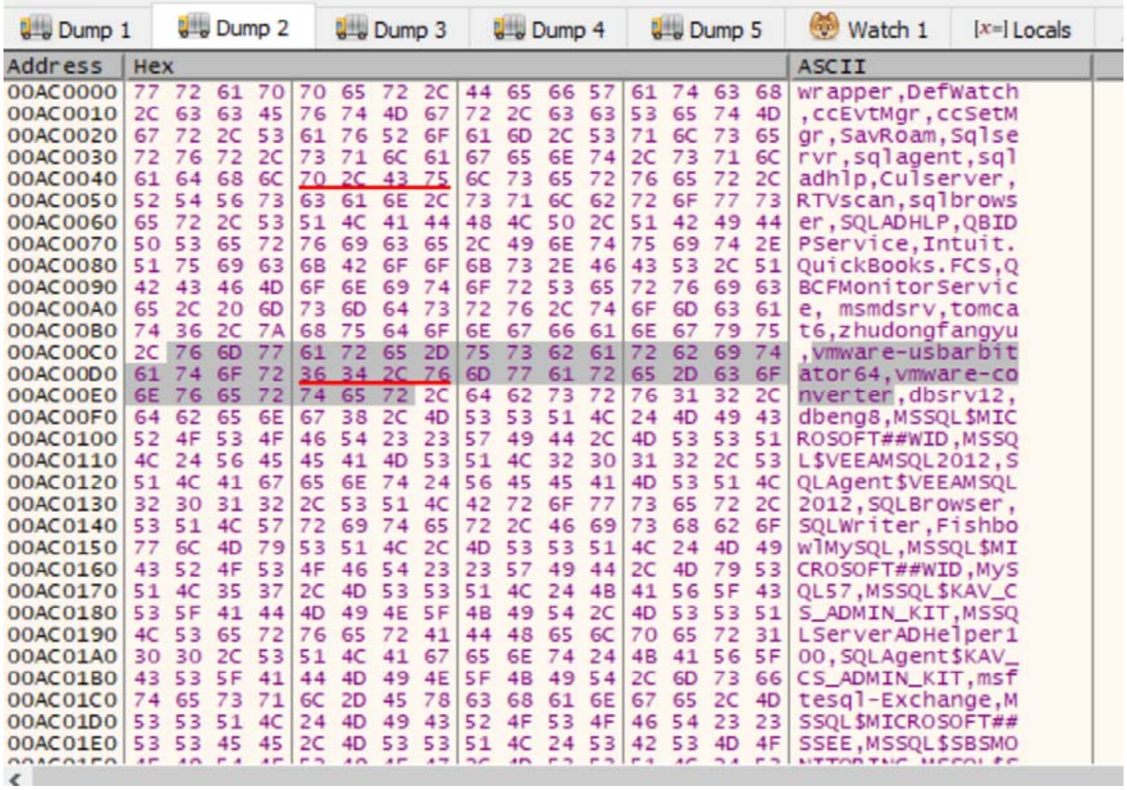
Figure 35 Decrypting "LockBit Ransomware 2.0" string

5. **Kills processes and services:** The malware has a list of processes and services it kills if they are running when the malware is executed. This names of processes and services are not visible during static analysis and are decoded at runtime. The list of processes to kill includes processes like ProcMon, Process Explorer, Autoruns and Wireshark which make dynamic analysis challenging.



```
Address   Hex                                                ASCII
00AB0330  76 69 73 69 6F 2C 77 6F 72 64 70 61 64 2C 62 65  visio,wordpad,be
00AB0340  64 62 68 2C 76 78 6D 6F 6E 2C 62 65 6E 65 74 6E  dbh,vxmon,benetn
00AB0350  73 2C 62 65 6E 67 69 65 6E 2C 70 76 6C 73 76 72  s,bengien,pvlsvr
00AB0360  2C 62 65 73 65 72 76 65 72 2C 72 61 77 5F 61 67  ,beserver,raw_ag
00AB0370  65 6E 74 5F 73 76 63 2C 76 73 6E 61 70 76 73 73  ent_svc,vsnapvss
00AB0380  2C 43 61 67 53 65 72 76 69 63 65 2C 44 65 6C 6C  ,CagService,Dell
00AB0390  53 79 73 74 65 6D 44 65 74 65 63 74 2C 45 6E 74  SystemDetect,Ent
00AB03A0  65 72 70 72 69 73 65 43 6C 69 65 6E 74 2C 50 72  erpriseClient,Pr
00AB03B0  6F 63 65 73 73 48 61 63 6B 65 72 2C 50 72 6F 63  ocessHacker,Proc
00AB03C0  65 78 70 36 34 2C 50 72 6F 63 65 78 70 2C 47 6C  exp64,Procexp,Gl
00AB03D0  61 73 73 57 69 72 65 2C 47 57 43 74 6C 53 72 76  assWire,GWCtlSrv
00AB03E0  2C 57 69 72 65 53 68 61 72 6B 2C 64 75 6D 70 63  ,WireShark,dumpc
00AB03F0  61 70 2C 6A 30 67 6E 6A 6B 6F 31 2C 41 75 74 6F  ap,j0gnjko1,Auto
00AB0400  72 75 6E 73 2C 41 75 74 6F 72 75 6E 73 36 34 2C  runs,Autoruns64,
00AB0410  41 75 74 6F 72 75 6E 73 36 34 61 2C 41 75 74 6F  Autoruns64a,Auto
00AB0420  72 75 6E 73 63 2C 41 75 74 6F 72 75 6E 73 63 36  runsc,Autorunsc6
00AB0430  34 2C 41 75 74 6F 72 75 6E 73 63 36 34 61 2C 53  4,Autorunsc64a,S
00AB0440  79 73 6D 6F 6E 2C 53 79 73 6D 6F 6E 36 34 2C 70  ysmon,Sysmon64,p
00AB0450  72 6F 63 65 78 70 36 34 61 2C 70 72 6F 63 6D 6F  rocexp64a,procmo
00AB0460  6E 2C 70 72 6F 63 6D 6F 6E 36 34 2C 70 72 6F 63  n,procmon64,proc
00AB0470  6D 6F 6E 36 34 61 2C 41 44 45 78 70 6C 6F 72 65  mon64a,ADExplore
00AB0480  72 2C 41 44 45 78 70 6C 6F 72 65 72 36 34 2C 41  r,ADExplorer64,A
00AB0490  44 45 78 70 6C 6F 72 65 72 36 34 61 2C 74 63 70  DExplorer64a,tcp
00AB04A0  76 69 65 77 2C 74 63 70 76 69 65 77 36 34 2C 74  view,tcpview64,t
00AB04B0  63 70 76 69 65 77 36 34 61 2C 61 76 7A 2C 74 64  cpview64a,avz,td
00AB04C0  73 73 6B 69 6C 6C 65 72 2C 52 61 63 63 69 6E 65  sskiller,Raccine
00AB04D0  45 6C 65 76 61 74 65 64 43 66 67 2C 52 61 63 63  ElevatedCfg,Racc
00AB04E0  69 6E 65 53 65 74 74 69 6E 67 73 2C 52 61 63 63  ineSettings,Racc
00AB04F0  69 6E 65 5F 78 38 36 2C 52 61 63 63 69 6E 65 2C  ine_x86,Raccine,
```

Figure 36 List of processes to kill



```
Address   Hex                                                ASCII
00AC0000  77 72 61 70 70 65 72 2C 44 65 66 57 61 74 63 68  wrapper,DefWatch
00AC0010  2C 63 63 45 76 74 4D 67 72 2C 63 63 53 65 74 4D  ,ccEvtMgr,ccSetM
00AC0020  67 72 2C 53 61 76 52 6F 61 6D 2C 53 71 6C 73 65  gr,SavRoam,Sqlse
00AC0030  72 76 72 2C 73 71 6C 61 67 65 6E 74 2C 73 71 6C  rvr,sqlagent,sql
00AC0040  61 64 68 6C 70 2C 43 75 6C 73 65 72 76 65 72 2C  adhlp,Culserver,
00AC0050  52 54 56 73 63 61 6E 2C 73 71 6C 62 72 6F 77 73  RTVscan,sqlbrows
00AC0060  65 72 2C 53 51 4C 41 44 48 4C 50 2C 51 42 49 44  er,SQLADHLP,QBID
00AC0070  50 53 65 72 76 69 63 65 2C 49 6E 74 75 69 74 2E  PService,Intuit.
00AC0080  51 75 69 63 6B 42 6F 6F 6B 73 2E 46 43 53 2C 51  QuickBooks.FCS,Q
00AC0090  42 43 46 4D 6F 6E 69 74 6F 72 53 65 72 76 69 63  BCFMonitorServic
00AC00A0  65 2C 20 6D 73 6D 64 73 72 76 2C 74 6F 6D 63 61  e, msmdsrv,tomca
00AC00B0  74 36 2C 7A 68 75 64 6F 6E 67 66 61 6E 67 79 75  t6,zhudongfangyu
00AC00C0  2C 76 6D 77 61 72 65 2D 75 73 62 61 72 62 69 74  ,vmware-usbarbit
00AC00D0  61 74 6F 72 36 34 2C 76 6D 77 61 72 65 2D 63 6F  ator64,vmware-co
00AC00E0  6E 76 65 72 74 65 72 2C 64 62 73 72 76 31 32 2C  nverter,dbsrv12,
00AC00F0  64 62 65 6E 67 38 2C 4D 53 53 51 4C 24 4D 49 43  dbeng8,MSSQL$MIC
00AC0100  52 4F 53 4F 46 54 23 23 57 49 44 2C 4D 53 53 51  ROSOFT##WID,MSSQ
00AC0110  4C 24 56 45 45 41 4D 53 51 4C 32 30 31 32 2C 53  L$VEEAMSQL2012,S
00AC0120  51 4C 41 67 65 6E 74 24 56 45 45 41 4D 53 51 4C  QLAgent$VEEAMSQL
00AC0130  32 30 31 32 2C 53 51 4C 42 72 6F 77 73 65 72 2C  2012,SQLBrowser,
00AC0140  53 51 4C 57 72 69 74 65 72 2C 46 69 73 68 62 6F  SQLWriter,Fishbo
00AC0150  77 6C 4D 79 53 51 4C 2C 4D 53 53 51 4C 24 4D 49  wlMySQL,MSSQL$MI
00AC0160  43 52 4F 53 4F 46 54 23 23 57 49 44 2C 4D 79 53  CROSOFT##WID,MyS
00AC0170  51 4C 35 37 2C 4D 53 53 51 4C 24 4B 41 56 5F 43  QL57,MSSQL$KAV_C
00AC0180  53 5F 41 44 4D 49 4E 5F 4B 49 54 2C 4D 53 53 51  S_ADMIN_KIT,MSSQ
00AC0190  4C 53 65 72 76 65 72 41 44 48 65 6C 70 65 72 31  LServerADHelper1
00AC01A0  30 30 2C 53 51 4C 41 67 65 6E 74 24 4B 41 56 5F  00,SQLAgent$KAV_
00AC01B0  43 53 5F 41 44 4D 49 4E 5F 4B 49 54 2C 6D 73 66  CS_ADMIN_KIT,msf
00AC01C0  74 65 73 71 6C 2D 45 78 63 68 61 6E 67 65 2C 4D  tesql-Exchange,M
00AC01D0  53 53 51 4C 24 4D 49 43 52 4F 53 4F 46 54 23 23  SSQL$MICROSOFT##
00AC01E0  53 53 45 45 2C 4D 53 53 51 4C 24 53 42 53 4D 4F  SSEE,MSSQL$SBSMO
```

Figure 37 List of services to kill

Indicators of compromise

- **Host based indicators:**
  - Desktop background
  - Presence of files with .lockbit extension
  - Presence of LockBit_Ransomware.hta file on Desktop
  - Presence of Restore-My-Files.txt in any folder
  - Presence of mutant \BaseNamedObjects\{A6E8DCE4-A6E8-7875-0E52-0E52-236D6DD023EE}
  - Presence of the following registry keys:
    - Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Classes\.lockbit
    - Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Classes\Lockbit
    - Presence of value C:\Users\<Username>\Desktop\LockBit_Ransomware.hta in the key Computer\HKEY_USERS\<UserID>\Software\Microsoft\Windows\CurrentVersion\Run
- **Network based indicators:** The malware doesn't do any network communication, so it does not have any network based indicators. There were some LDAP and Active Directory related strings encountered during static analysis but as the malware wasn't run on a Windows Server machine, there wasn't a way to verify the use of these strings dynamically.

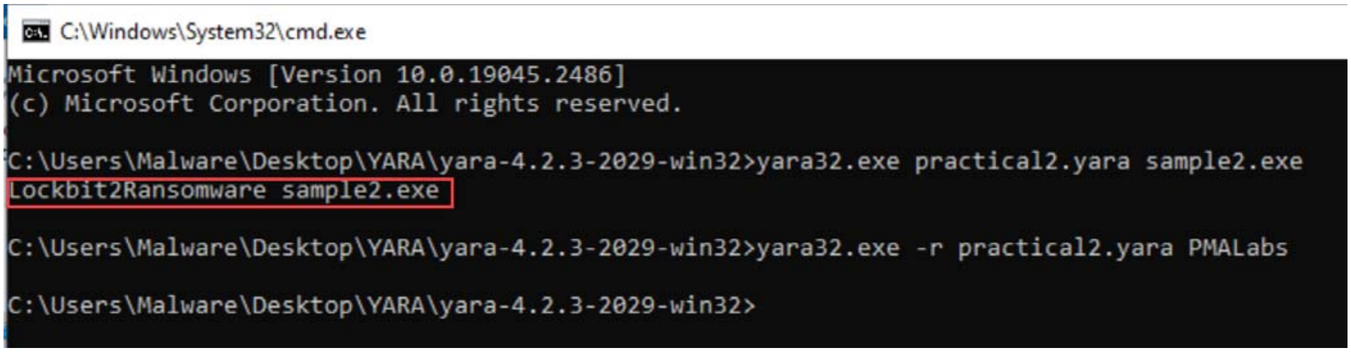
Yara Rule:

My YARA rule to detect Lockbit 2.0 Ransomware checks if the file is a PE file by comparing the first 2 bytes of the file against 4D 5A. Since the sample2.exe's size is 960KB, I put in a condition to not check any file with a size greater than 1000KB to speed up the search. The rule then checks if atleast 4 of the 7 string variables starting with s are found and atleast 3 of the 6 string variables starting with x are found. Though the sample contains a lot of strings which could be used as indicators, I chose 13 of them that seemed unique enough. The rule isn't case sensitive while comparing the strings and it checks for ASCII as well as Wide/Unicode strings.

```
rule Lockbit2Ransomware {

  meta:
    description = "Rule to detect Lockbit 2.0 Ransomware"
    author = "Rachana"
    date = "3/15/2023"


  strings:

    $s1 = "Tox messenger" ascii wide nocase
    $s2 = "Would you like to earn millions of dollars?" ascii wide nocase
```

```
        $s3 = "All your files stolen and encrypted" ascii wide nocase
        $s4 = "https://tox.chat/download.html" ascii wide nocase
        $s5 = "Using Tox messenger, we will never know your real name, it means your privacy is guaranteed." ascii wide nocase
        $s6 = "If this contact is expired, and we do not respond you, look for the relevant contact data on our website via Tor or Brave Browser" ascii wide nocase
        $s7 = "Get-ADComputer -filter * -Searchbase '%s' | foreach{ Invoke-GPUpdate -computer $_.name -force -RandomDelayInMinutes 0}" ascii wide nocase

        $x1 = "\\Registry\\Machine\\Software\\Classes\\.lockbit\\DefaultIcon" ascii wide nocase
        $x2 = "\\Registry\\Machine\\Software\\Classes\\Lockbit\\shell" ascii wide nocase
        $x3 = "LockBit_Ransomware.hta" ascii wide nocase
        $x4 = "LockBit 2.0 Ransom" ascii wide nocase
        $x5 = "LockBit_2_0_Ransom" ascii wide nocase
        $x6 = "lockbit" ascii wide nocase

 condition:
        uint16(0) == 0x5a4d and filesize < 1000KB and (4 of ($s*) and 3 of ($x*))
}
```

The rule only fires on sample2.exe. It doesn’t fire on any other samples from Practical Malware Analysis.



```
Microsoft Windows [Version 10.0.19045.2486]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Malware\Desktop\YARA\yara-4.2.3-2029-win32>yara32.exe practical2.yara sample2.exe
Lockbit2Ransomware sample2.exe

C:\Users\Malware\Desktop\YARA\yara-4.2.3-2029-win32>yara32.exe -r practical2.yara PMALabs

C:\Users\Malware\Desktop\YARA\yara-4.2.3-2029-win32>
```

Figure 38 Testing the rule on sample2.exe and PMA labs

References

1. https://www.windows-active-directory.com/active-directory-ad-fundamentals.html
2. https://sdmsoftware.com/whitepapers/understanding-group-policy-storage/