BRANDEFENSE

CYBER THREAT INTELLIGENCE

Lockbit 3.0 Technical Analysis

Author: Threat Intelligence Team

Release Date: 16.06.2022

Report ID: BD160622SM

Overview

With the group's return, Lockbit introduced Lockbit3.0, a new variant of Lockbit 2.0. LockBit 3.0 ransomware (aka LockBit Black) Based on the BlackMatter group and adopting Ransomware-as-a-Service, LockBit is an advanced version of the RaaS family. The ransomware, also called Lockbit Black, has developed itself with new extortion techniques and added the option to pay with Zcash and the existing Bitcoin and Monero crypto payment methods.

As a result of critical bugs discovered in Lockbit 2.0 in the first quarter of 2022, malware authors began adding new features to improve encryption processes and thwart security researchers.

In addition to these developments, Lockbit announced the Bug Bounty program, breaking new ground among cybercriminal gangs. For many other cybercriminals, the program promises rewards between $1000 and $1,000,000 for the idea of bug fixing or improving existing features.

Features Changed with Lockbit 3.0

With the introduction of Lockbit 3.0 by the Lockbit gang, Lockbit operators and their gang-affiliated collaborators started to adopt Lockbit 3.0 quickly. As a result, as of June 2022, affected organizations and many affected organizations have been identified on the “Version 3.0” leak site of the leaked data.

(hxxp://lockbitapt[REDACTED]ead[.]onion)

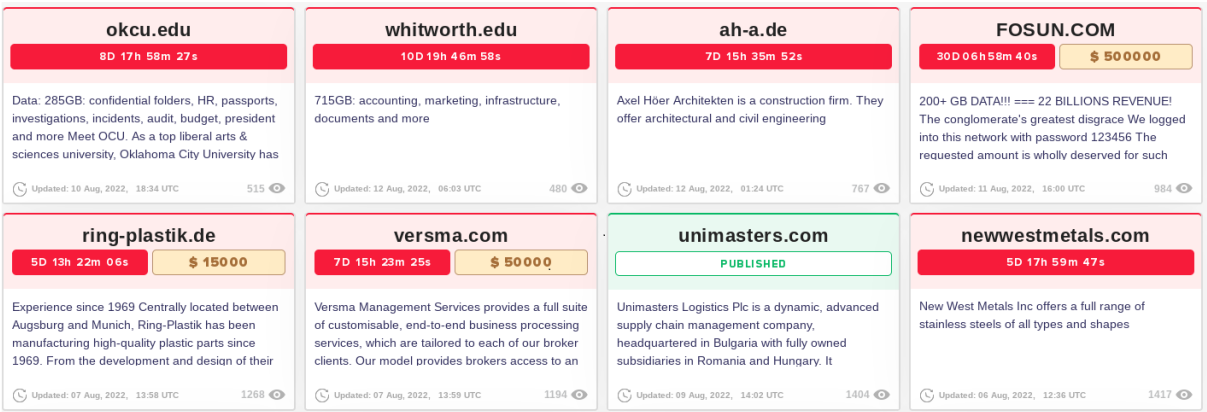


Figure 1: Organizations most recently affected by Lockbit 3.0

Lockbit is also aggressively releasing alternate Onion addresses with copies of the data they managed to leak to ensure continuity of announcements of leaked data and to increase resilience to interception efforts.

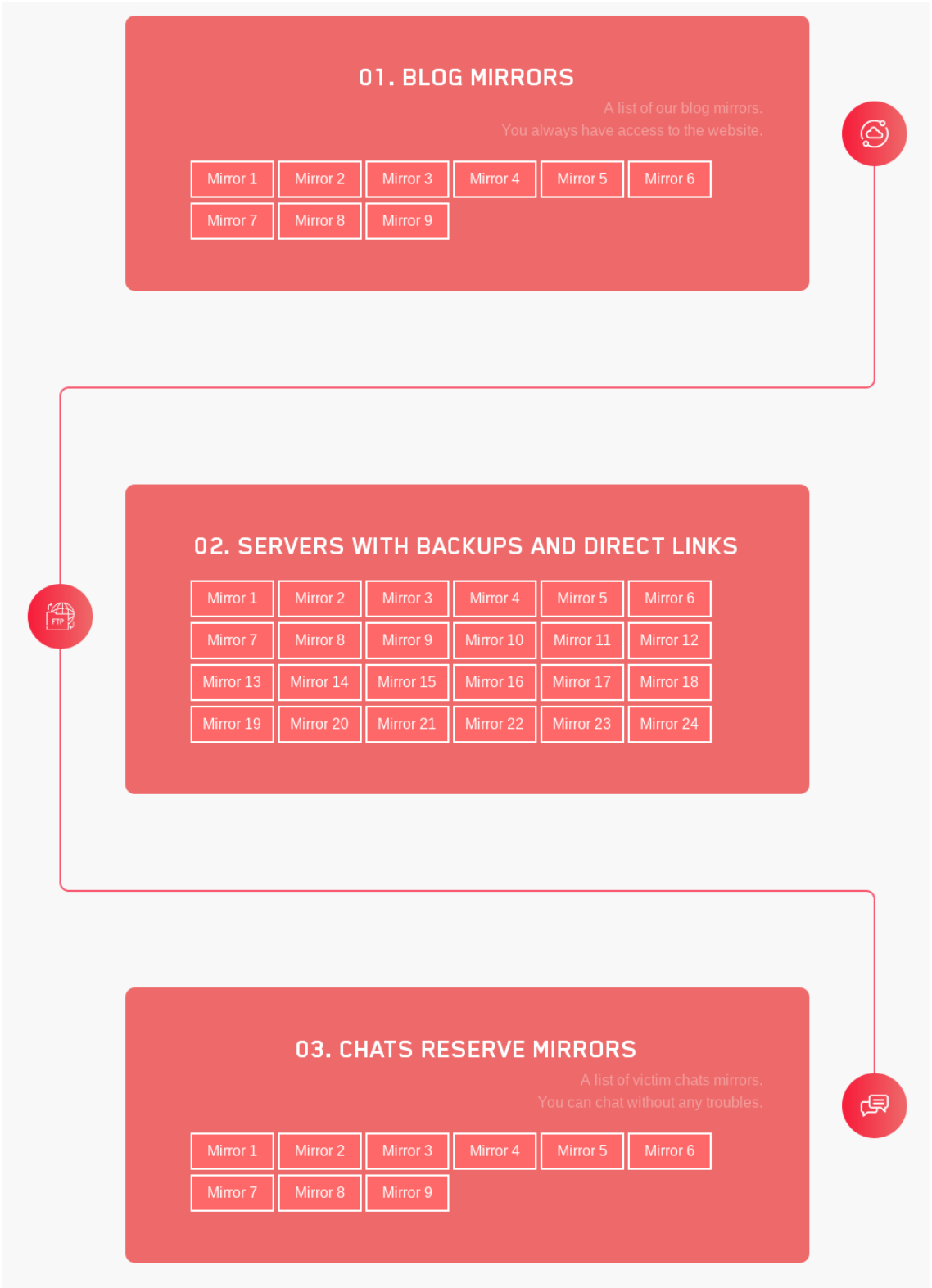


Figure 2: Mirror sites (.onion) used for data leaks, backups and communication

Another change with Lockbit 3.0 is adding a search feature that will allow a targeted organization to browse publicly published data without downloading it instantly.

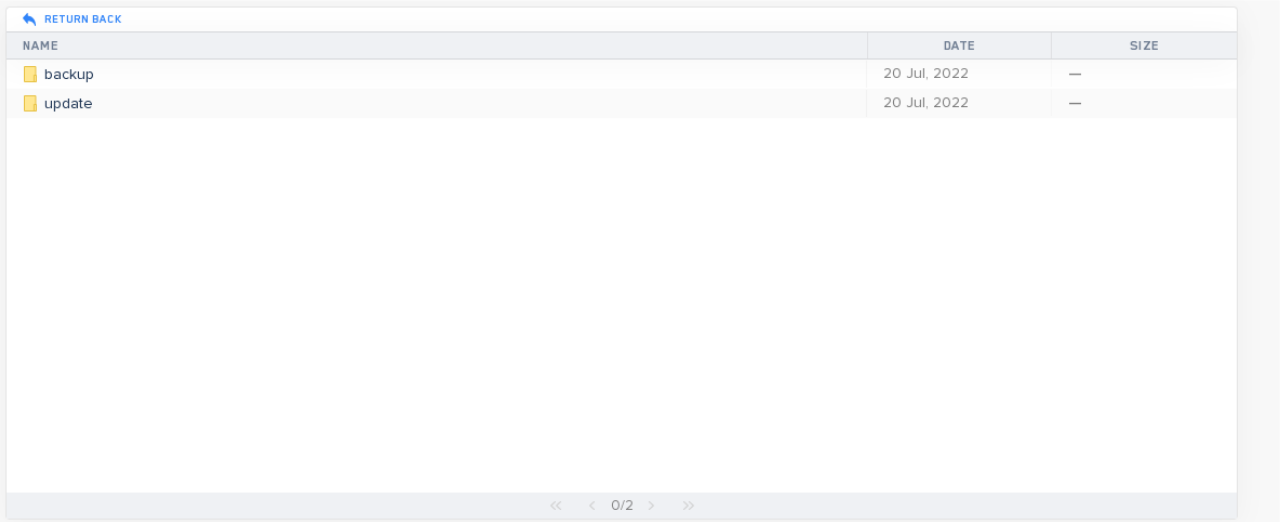


Figure 3: Added feature for instant search of leaked data

In addition to its existing payment methods Bitcoin and Monero, Lockbit has added the ability to pay with Zcash crypto. On the other hand, one of the most remarkable developments was the bug bounty program they announced.

Among the topics in the scope of the program is the detection of errors that may occur during encryption, XSS, shell, etc., found on the website. All kinds of ideas can make Lockbit more dangerous and functional, such as injections, detection of situations that may reveal the identities of collaborators, TOX Messenger vulnerabilities used for messaging, IP address learning for servers in the TOR network, root access, database dumping, and all kinds of ideas that can make Lockbit more dangerous and functional.

Bug Bounty Program

— # —

We invite all security researchers, ethical and unethical hackers on the planet to participate in our bug bounty program.
The amount of remuneration varies from $1000 to $1 million.

Web Site Bugs

XSS vulnerabilities, mysql injections, getting a shell to the site and more, will be paid depending on the severity of the bug, the main direction is to get a decryptor through bugs web site, as well as access to the history of correspondence with encrypted companies.

Locker Bugs

Any errors during encryption by lockers that lead to corrupted files or to the possibility of decrypting files without getting a decryptor.

Brilliant ideas

We pay for ideas, please write us how to improve our site and our software, the best ideas will be paid. What is so interesting about our competitors that we don't have?

Doxing

We pay exactly one million dollars, no more and no less, for doxing the affiliate program boss. Whether you're an FBI agent or a very clever hacker who knows how to find anyone, you can write us a TOX messenger, give us your boss's name, and get $1 million in bitcoin or monero for it.

TOX messenger

Vulnerabilities of TOX messenger that allow you to intercept correspondence, run malware, determine the IP address of the interlocutorand other interesting vulnerabilities.

Tor network

Any vulnerabilities which help to get the IP address of the server where the site is installed on the onion domain, as well as getting root access to our servers, followed by a database dump and onion domains.

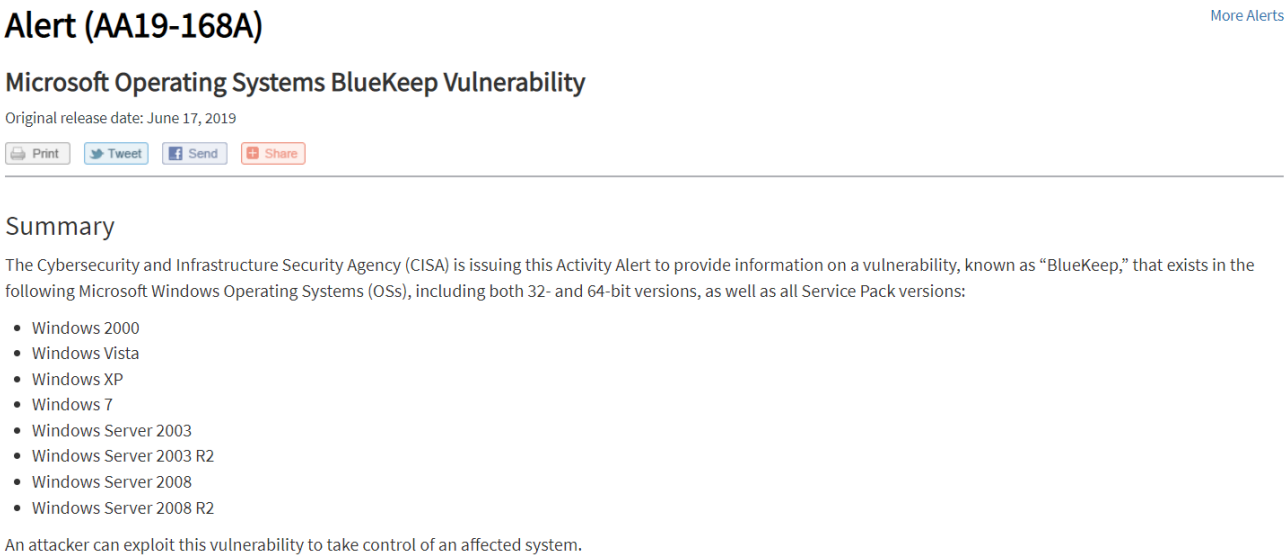
Figure 4: Bounty hunting program announced with Lockbit 3.0

Technical Analysis

Initial Access and First Execution

Methods found to be used by Lockbit 3.0 members to gain initial access by targeting organizations include valid Local Admin account information that has been compromised to gain access to the organization's network and the CVE-2019-0708 BlueKeep vulnerability.

Below are the systems affected by the BlueKeep vulnerability.



Alert (AA19-168A)

More Alerts

Microsoft Operating Systems BlueKeep Vulnerability

Original release date: June 17, 2019

Print Tweet Send Share

Summary

The Cybersecurity and Infrastructure Security Agency (CISA) is issuing this Activity Alert to provide information on a vulnerability, known as "BlueKeep," that exists in the following Microsoft Windows Operating Systems (OSs), including both 32- and 64-bit versions, as well as all Service Pack versions:

- Windows 2000
- Windows Vista
- Windows XP
- Windows 7
- Windows Server 2003
- Windows Server 2003 R2
- Windows Server 2008
- Windows Server 2008 R2

An attacker can exploit this vulnerability to take control of an affected system.

Figure 5: Systems affected by the BlueKeep vulnerability used by Lockbit

The BlueKeep vulnerability resides in the Remote Desktop Protocol (RDP) used by the Microsoft Windows operating systems listed above.

After obtaining the first access, the Lockbit member finds the appropriate environment to run the ransomware in the target environment. Still, the change introduced with Lockbit 3.0 cannot be run with standard user interaction.

For the Lockbit ransomware to run for the first time, the member who gains access to the target system must run a command similar to the one below via the Windows Command Line (CMD).

filename>.exe -k LocalServiceNetworkRestricted -pass db66023ab2abcb9957fb01ed50cdfa6a

With such an execution method, attackers aim to:

If the Lockbit 3.0 executable comes under the scrutiny of a security researcher, it is to prevent analysis. The parameter passed to the executable with *-pass* is used to decode the program and make it run and may differ between different Lockbit members. Under normal circumstances, no one other than the Lockbit member knows the parameter needed to decode the program.

File System Changes

When the attacker runs the program, it leaves a ransom note named HLJkNskOq.README.txt, containing the ransom note and Onion URLs, in all the directories where encryption is made. The Lockbit replaces the file extension of encrypted files with HLJkNskOq.

HLJkNskOq.README.txt - Notepad

File Edit Format View Help

```
~~~ LockBit 3.0 the world's fastest and most stable ransomware from 2019~~~

>>>>> Your data is stolen and encrypted.
If you don't pay the ransom, the data will be published on our TOR darknet sites. Keep in mind that once your data appears on our leak site, it could be bought b

Tor Browser Links:
http://lockbitapt2d73krlbewgv27tquljgxr33xbwwsp6rkyieto7u4ncead.onion
http://lockbitapt2yfbt7lchxejug47kmqvqqxvvjpqkmevv4l3azl3gy6pyd.onion
http://lockbitapt34kvrip6xojylohhxrwsvpzdffgs5z4pbbsywnzsbdguqd.onion
http://lockbitapt5x4zkjbcqmz6frdhecqqgadevyiwqxukksspnlidyvd7qd.onion
http://lockbitapt6vx57t3eeqjofwgcglmutr3a35nygvokja5uuccip4ykyd.onion
http://lockbitapt72iw55njgnqpymggskg5yp75ry7rirtdg4m7i42artsbqd.onion
http://lockbitaptawjl6udhpd323uehekiyatj6ftcxmkwe5sezs4fqgpjpid.onion
http://lockbitaptbdiajqtplcrigzgdjprwugkkut63nbvy2d5r4w2agyekqd.onion
http://lockbitaptc2iq4atewz2ise62q63wfktyrl4qtwuk5qax262kgtzjqd.onion

Links for normal browser:
http://lockbitapt2d73krlbewgv27tquljgxr33xbwwsp6rkyieto7u4ncead.onion.ly
http://lockbitapt2yfbt7lchxejug47kmqvqqxvvjpqkmevv4l3azl3gy6pyd.onion.ly
http://lockbitapt34kvrip6xojylohhxrwsvpzdffgs5z4pbbsywnzsbdguqd.onion.ly
http://lockbitapt5x4zkjbcqmz6frdhecqqgadevyiwqxukksspnlidyvd7qd.onion.ly
http://lockbitapt6vx57t3eeqjofwgcglmutr3a35nygvokja5uuccip4ykyd.onion.ly
http://lockbitapt72iw55njgnqpymggskg5yp75ry7rirtdg4m7i42artsbqd.onion.ly
http://lockbitaptawjl6udhpd323uehekiyatj6ftcxmkwe5sezs4fqgpjpid.onion.ly
http://lockbitaptbdiajqtplcrigzgdjprwugkkut63nbvy2d5r4w2agyekqd.onion.ly
http://lockbitaptc2iq4atewz2ise62q63wfktyrl4qtwuk5qax262kgtzjqd.onion.ly

>>>>> What guarantee is there that we won't cheat you?
We are the oldest ransomware affiliate program on the planet, nothing is more important than our reputation. We are not a politically motivated group and we want

>>>>> You need to contact us and decrypt one file for free on TOR darknet sites with your personal ID
```

Figure 6: A portion of the ransom note left on the file system

HLJkNskOq.bmp for the background image, HLJkNskOq.ico, and D6AA.tmp for encrypted file icons are dropped into the C:\ProgramData directory. D6AA.tmp is the Windows executable.

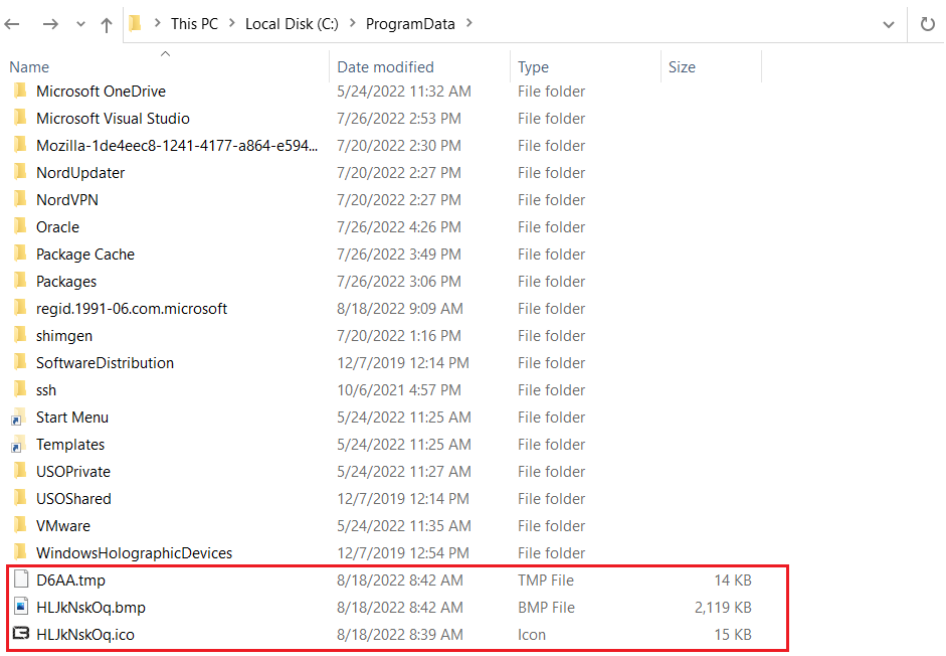


Figure 7: Files dropped in C:\ProgramData directory

The D6AA.tmp file changes every time the program is run, but the filename length is a fixed four characters long.

Process Activity

When the first stage program file is run via the command line, it terminates itself and restarts it under the same name under the dllhost.exe process. The process started under dllhost.exe and runs the D6AA.tmp file in the C:\ProgramData directory. After the first stage file is run, it is deleted for privacy purposes.

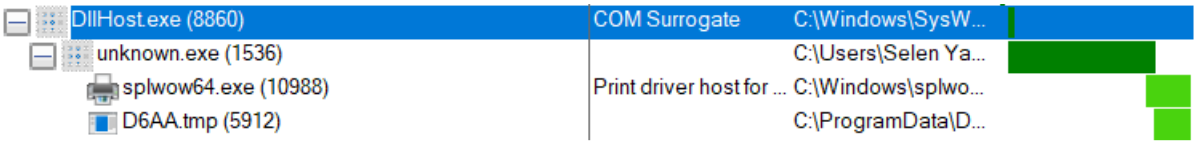


Figure 8: Process tree formed after the first stage program file is finalized

When the D6AA.tmp file is run, the C:\AF485E4C directory is created, and the D7F1.tmp file is left in this directory.

Windows Defender was disabled when the Lockbit had run.

The new process started under dllhost.exe also starts the splwow64.exe process. splwow64.exe is a Windows process that runs when using 32-bit printer drivers on 64-bit Windows operating systems. This process is executed when print jobs are sent. In this case, Lockbit 3.0 may attempt to print the ransom note via printers connected to the computer.

Registry Changes

The started process under dllhost.exe
Sets their values to 0 to disable many registry keys associated with the Windows Event Log mechanism under the
HKLM\Software\Microsoft\Windows\CurrentVersion\WINEVT\Channels registry.

For this, the Enabled subkey is set to 0 for each log type under
HKLM\Software\Microsoft\Windows\CurrentVersion\WINEVT\Channels\<Log Name>.

You can make these registry changes like Windows Defender, Volume Shadow Copy, etc., on the target system. We think that they are implemented to prevent the recordings that will occur during the deactivation of features.

Analysis of Program Code

The program starts running at address 41B0000. The loop in the sub_41B248 routine parses the "-pass" statement from the statement used as the command line to run the program file.

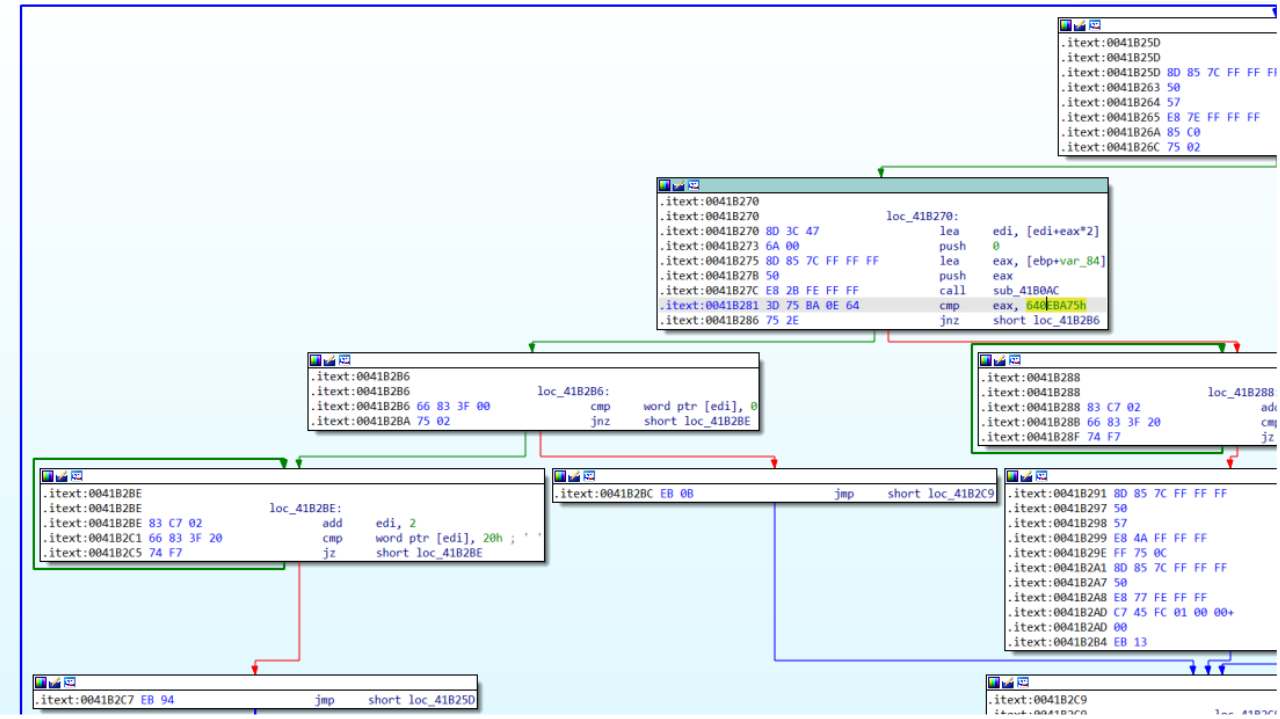


Figure 9: Check for -pass argument name at 0x41B248

The result of parsing the -pass parameter name in the sub_41B0AC routine with the mathematical combination of add, sub, mov, and ror commands are checked against the value of 640EBA75h. The success of this check indicates that the -pass argument name is correctly given on the command line to run the program.

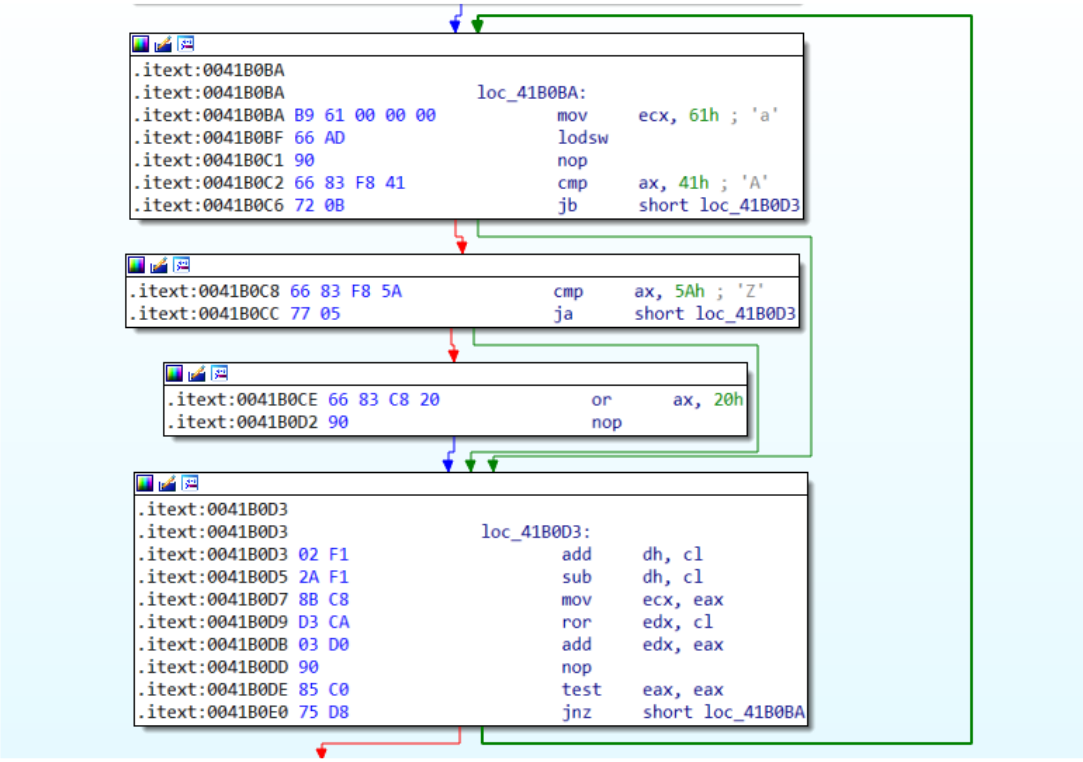


Figure 10: Calculating the hexadecimal value for the -pass argument name

Then the expression used as the value of the -pass parameter is checked against 32 characters long. When all required command line arguments are passed correctly, the EAX register is set to 1 and is used to check if the check is successful.

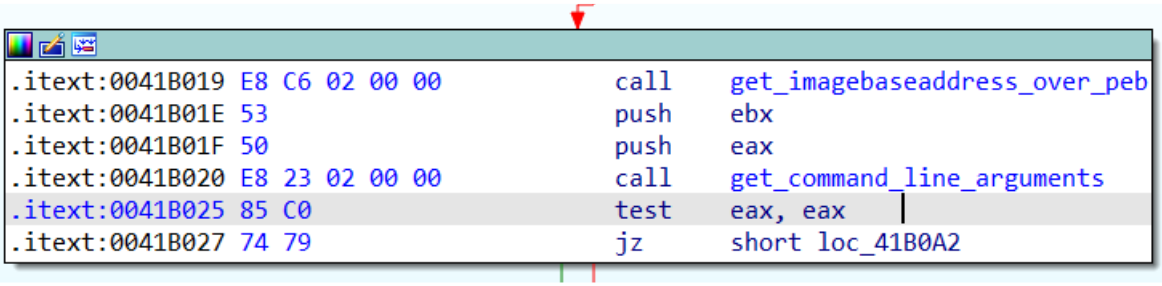


Figure 11: The function responsible for obtaining the command line parameters necessary for the program file to run

When we examined the program with Detect it Easy, we found that the entropy value of the .text section was higher than usual. This may indicate that additional executable code is packaged within the program.

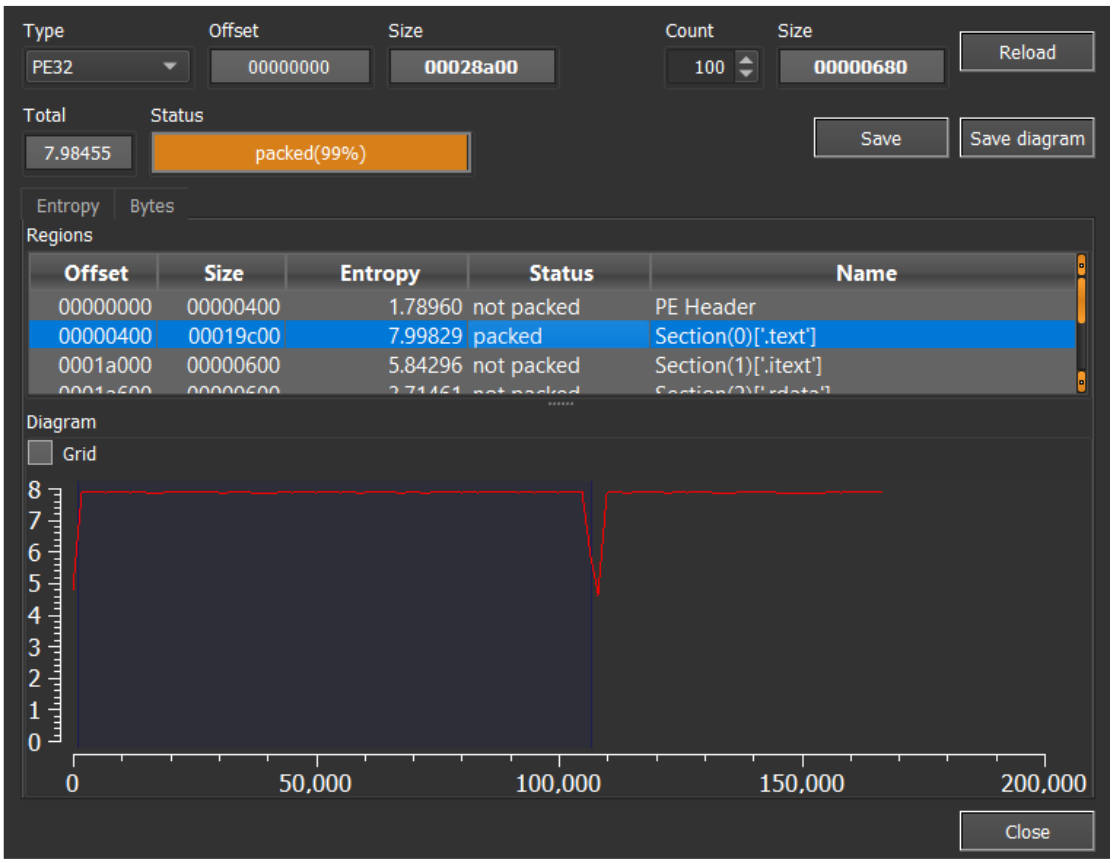


Figure 12: Entropy value showing that the .text part of the program may contain additional packed data

During the program's execution, using the -pass parameter transferred over the command line, the encrypted part of the code is decrypted and rewritten to the .text part of the running program file.

The data kept embedded in the program in the .text, .data, and .pdata parts of the program that was run at the 41B000 address in the first stage was written as new data, possibly passing through a decryption algorithm. The routine responsible for writing the data to the program sections by XORing is located at 41B095. After writing to program sections is complete, the control stream branches to address 408254.

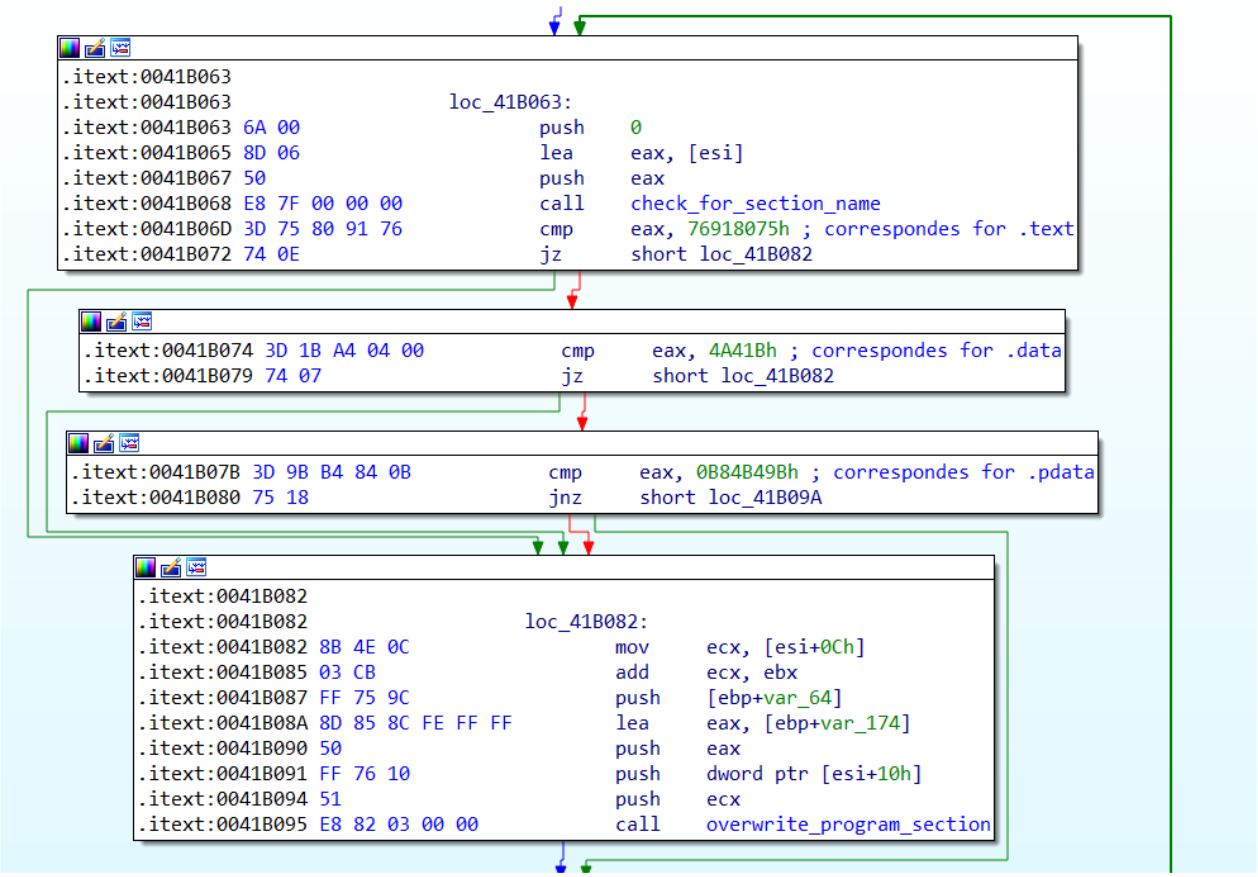


Figure 13: Code block responsible for writing data to program parts

The memory dump of the state of the current running program before and after the new data is written to the .text section is given below.

Adres	Hex																ASCII
00401000	B8	7A	F2	C0	29	21	A2	2E	AD	28	55	6B	C1	AE	27	65	¸zòÀ)!¢..(UkÁ®'e
00401010	BC	EE	2B	04	1D	78	95	C0	71	EE	B4	83	DD	FF	76	28	¼î+..x.Àqî´.Ýÿv(
00401020	15	F0	75	5D	CA	05	DF	9E	13	09	3E	E4	07	13	E9	17	.ðu]Ê.ß...>ä..é.
00401030	2F	02	B1	74	E5	EB	57	EC	32	6E	83	BD	4B	90	72	F2	/.±tåëWì2n.½K.rò
00401040	F9	E5	E0	7A	B4	3C	39	19	B9	42	D8	94	8C	E7	F2	EA	ùåàz´<9.'BØ..çòê
00401050	30	D8	01	96	D3	34	1E	E3	A6	5D	76	14	97	D7	C3	84	0Ø..Ó4.ã¦]v..×Ã.
00401060	D3	88	24	4D	BD	9F	70	47	C4	E9	1F	F6	BD	F6	DE	7C	Ó.$M½.pGÄé.ö½öÞ|
00401070	5C	B5	88	64	C0	31	DB	ED	D7	2B	FF	46	17	FD	06	E4	\µ.dÀ1Ûí×+ÿF.ý.ä
00401080	52	32	FC	07	00	60	FD	3E	BA	92	2E	3A	EA	6E	05	43	R2ü..`ý>º..:ên.C
00401090	16	5F	CF	EF	50	48	C7	C0	53	50	84	73	8D	1E	19	94	._ÏïPHÇÀSP.s....
004010A0	CF	30	F1	9E	43	99	40	24	B3	66	78	FB	0E	F3	F2	EF	Ï0ñ.C.@$³fxû.óòï
004010B0	0C	3B	4B	93	48	03	2D	6A	D0	C3	8A	DA	39	CD	5F	F9	.;K.H.-jÐÃ.Ú9Í_ù
004010C0	9A	01	D7	73	ED	A3	5E	58	55	2B	F7	3E	61	1A	F1	11	..×sí£^XU+÷>a.ñ.
004010D0	0F	66	EF	14	B8	29	33	46	CC	8C	33	5D	FB	48	93	F4	.fï.¸)3FÌ.3]ûH.ô
004010E0	14	57	8F	60	07	1B	E4	8E	42	52	F0	FF	9B	37	93	99	.W.`..ä.BRðÿ.7..
004010F0	2E	63	F4	95	BA	45	0E	B1	D4	DE	8D	6B	1A	C0	1B	84	.cô.ºE.±ÔÞ.k.À..
00401100	4F	34	DB	53	7B	96	00	61	C7	F9	D3	A9	2F	95	5F	A4	O4ÛS{..aÇùÓ©/._¤

Figure 14: Before writing new data

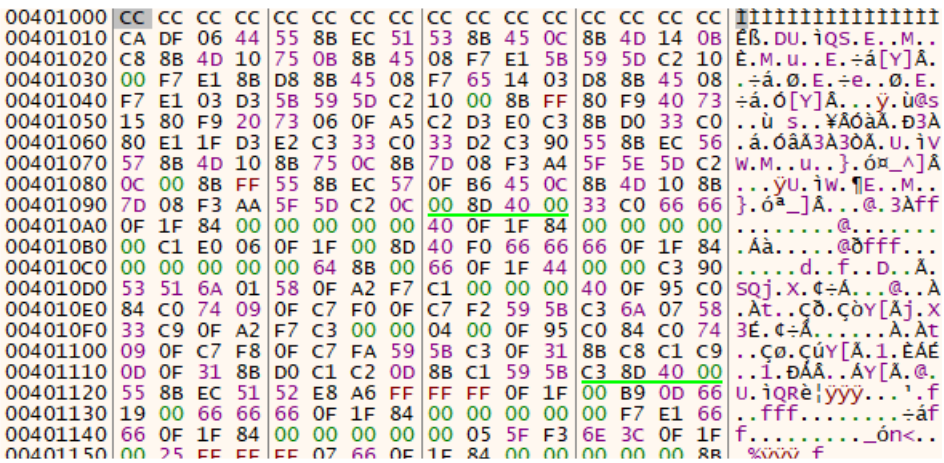


Figure 15: After the new data is written

At this stage, values displayed by the debugger and disassembler will be different because the content of the .text part of the code has changed. Because the changes made in the program sections are reflected the debugger instantly, and the execution continues on the changed sections. To view the current control flow on the disassembler, we can dump the program can be obtained as a new OEP at address 408254.

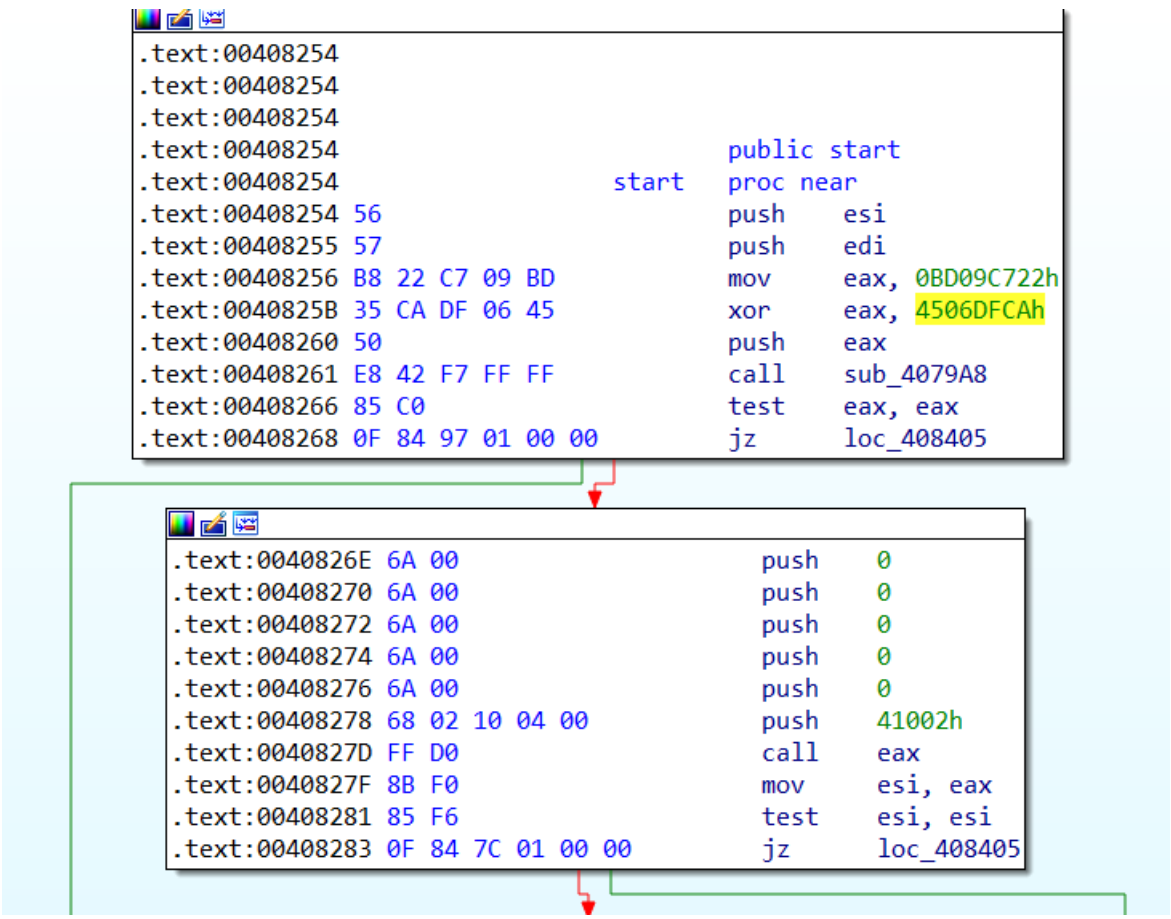


Figure 16: Continuing control flow start at 408254

Resolving of API Functions

After the program starts working in the new control flow, it resolves the LdtGetProcedureAddress and LdrLoadDll API functions from the NTDLL library. Function addresses are determined at address 4079A8. These functions are used to find other API calls from the relevant libraries to be analyzed in the future. Lockbit uses a 4-bytes 4506DFCA XOR key when resolving API calls. All processing of API call resolution is done at 407C5C.

```
.text:00408254                          public start
.text:00408254                  start   proc near
.text:00408254 56                       push    esi
.text:00408255 57                       push    edi
.text:00408256 B8 22 C7 09 BD           mov     eax, 0BD09C722h
.text:0040825B 35 CA DF 06 45           xor     eax, 4506DFCAh
.text:00408260 50                       push    eax
.text:00408261 E8 42 F7 FF FF           call    locate_function_address ; RtlCreateHeap
.text:00408266 85 C0                    test    eax, eax
.text:00408268 0F 84 97 01 00 00        jz      loc_408405
```

```
.text:0040826E 6A 00                    push    0
.text:00408270 6A 00                    push    0
.text:00408272 6A 00                    push    0
.text:00408274 6A 00                    push    0
.text:00408276 6A 00                    push    0
.text:00408278 68 02 10 04 00           push    41002h
.text:0040827D FF D0                    call    eax
.text:0040827F 8B F0                    mov     esi, eax
.text:00408281 85 F6                    test    esi, esi
.text:00408283 0F 84 7C 01 00 00        jz      loc_408405
```

Figure 17: Routine that determines the function address

The first resolved API calls are the FindFirstFile, FindNextFile, and FindClose functions used to search for files. The functions are resolved by loading the like ntdll.dll and kernel32.dll libraries with calls made to the function at 407C5C. For example, the expressions specifying the System32 file path (C:\Windows\System32) and the DLL file extension are parsed and combined into the FindFirstFile API function (C:\Windows\System32*.dll). Next, it finds the ntdll.dll library from the System32 directory and loads it with the call to LdrLoadDll.

Libraries the program needs (ntdll.dll, kernel32.dll, etc.) and API functions are loaded with consecutive calls to the routine at 407C5C.

```
00407BB2    50                  push eax                                    eax:L"userenv.dll"
00407BB3    FF15 FC734200       call dword ptr ds:[<&FindFirstFileW>
00407BB9    8945 F8             mov dword ptr ss:[ebp-8],eax
00407BBC    837D F8 FF          cmp dword ptr ss:[ebp-8],FFFFFFFF
00407BC0    74 4E               je unknown.407C10
00407BC2    6A 00               push 0
00407BC4    8D85 D4FDFFFF       lea eax,dword ptr ss:[ebp-22C]
00407BCA    50                  push eax                                    eax:L"userenv.dll"
00407BCB    E8 1496FFFF         call unknown.4011E4
00407BD0    3B45 08             cmp eax,dword ptr ss:[ebp+8]
00407BD3    75 1E               jne unknown.407BF3
00407BD5    FF75 F8             push dword ptr ss:[ebp-8]
00407BD8    FF15 04744200       call dword ptr ds:[<&FindClose>]
00407BDE    6A 01               push 1
00407BE0    8D85 D4FDFFFF       lea eax,dword ptr ss:[ebp-22C]
00407BE6    50                  push eax                                    eax:L"userenv.dll"
00407BE7    E8 F0FCFFFF         call unknown.4078DC
00407BEC    5B                  pop ebx                                     ebx:L"\\*.dll"
00407BED    8BE5                mov esp,ebp
00407BEF    5D                  pop ebp
00407BF0    C2 0400             ret 4
```

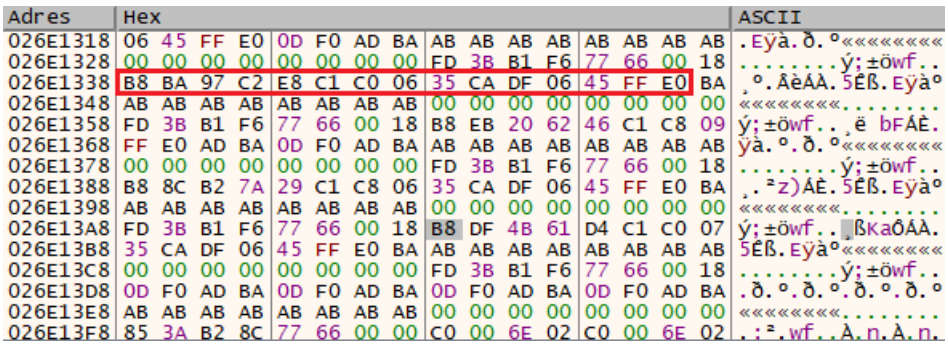
Figure 18: Code snippet that searches for DLLs under System32

```
.text:004082AF 57                      push    edi
.text:004082B0 56                      push    esi
.text:004082B1 68 A4 7D 40 00          push    offset dword_407DA4
.text:004082B6 68 08 74 42 00          push    offset unk_427408
.text:004082BB E8 9C F9 FF FF          call    sub_407C5C
.text:004082C0 57                      push    edi
.text:004082C1 56                      push    esi
.text:004082C2 68 94 7E 40 00          push    offset dword_407E94
.text:004082C7 68 F4 74 42 00          push    offset unk_4274F4
.text:004082CC E8 8B F9 FF FF          call    sub_407C5C
.text:004082D1 57                      push    edi
.text:004082D2 56                      push    esi
.text:004082D3 68 88 7F 40 00          push    offset dword_407F88
.text:004082D8 68 E4 75 42 00          push    offset unk_4275E4
.text:004082DD E8 7A F9 FF FF          call    sub_407C5C
.text:004082E2 57                      push    edi
.text:004082E3 56                      push    esi
.text:004082E4 68 2C 80 40 00          push    offset dword_40802C
.text:004082E9 68 84 76 42 00          push    offset unk_427684
.text:004082EE E8 69 F9 FF FF          call    sub_407C5C
.text:004082F3 57                      push    edi
.text:004082F4 56                      push    esi
.text:004082F5 68 40 80 40 00          push    offset dword_408040
.text:004082FA 68 94 76 42 00          push    offset unk_427694
.text:004082FF E8 58 F9 FF FF          call    sub_407C5C
.text:00408304 57                      push    edi
.text:00408305 56                      push    esi
.text:00408306 68 7C 80 40 00          push    offset dword_40807C
.text:0040830B 68 CC 76 42 00          push    offset unk_4276CC
.text:00408310 E8 47 F9 FF FF          call    sub_407C5C
```

Figure 19: Consecutive API analysis

Each address range in the address range of the .text part of the program is transferred at address 407C5C. These data correspond to the 3rd parameter passed to the sub_407C5C call. A value of 0xCCCCCCC separates each piece of code. For each code block, 4-byte data is XORed, and the value obtained is transferred to the function at address 4079A8 to get the API function address. Each data block passed as a parameter to the call at address 407C5C corresponds to a different library and functions included in that library. The code used in the first call to sub_407C5C is labeled dword_407DA4, where each 4-byte data corresponds to the unresolved API call to be used by Lockbit in the ntdll.dll library.

Resolved API addresses are not called directly. Instead, it creates a heap object that can be used in the process's virtual address space with the RtlCreateHeap call and allocates storage space. With the RtlAllocateHeap call, memory is allocated from the previously created heap region, and information about API resolution is written to the allocated heap memory region. This method is called Trampoline Code.



Adres	Hex	ASCII
026E1318	06 45 FF E0 0D F0 AD BA AB AB AB AB AB AB AB AB	.Eÿà.ð.º««««««««
026E1328	00 00 00 00 00 00 00 00 FD 3B B1 F6 77 66 00 18	........ý;±öwf..
026E1338	B8 BA 97 C2 E8 C1 C0 06 35 CA DF 06 45 FF E0 BA	¸º.ÂèÁÀ.5Êß.Eÿà°
026E1348	AB AB AB AB AB AB AB AB 00 00 00 00 00 00 00 00	«««««««««.......
026E1358	FD 3B B1 F6 77 66 00 18 B8 EB 20 62 46 C1 C8 09	ý;±öwf..¸ë bFÁÈ.
026E1368	FF E0 AD BA 0D F0 AD BA AB AB AB AB AB AB AB AB	ÿà.º.ð.º««««««««
026E1378	00 00 00 00 00 00 00 00 FD 3B B1 F6 77 66 00 18	........ý;±öwf..
026E1388	B8 8C B2 7A 29 C1 C8 06 35 CA DF 06 45 FF E0 BA	¸.²z)ÁÈ.5Êß.Eÿà°
026E1398	AB AB AB AB AB AB AB AB 00 00 00 00 00 00 00 00	«««««««««.......
026E13A8	FD 3B B1 F6 77 66 00 18 B8 DF 4B 61 D4 C1 C0 07	ý;±öwf..¸ßKaÔÁÀ.
026E13B8	35 CA DF 06 45 FF E0 BA AB AB AB AB AB AB AB AB	5Êß.Eÿà°««««««««
026E13C8	00 00 00 00 00 00 00 00 FD 3B B1 F6 77 66 00 18	........ý;±öwf..
026E13D8	0D F0 AD BA 0D F0 AD BA 0D F0 AD BA 0D F0 AD BA	.ð.º.ð.º.ð.º.ð.º
026E13E8	AB AB AB AB AB AB AB AB 00 00 00 00 00 00 00 00	«««««««««.......
026E13F8	85 3A B2 8C 77 66 00 00 C0 00 6E 02 C0 00 6E 02	.:².wf..À.n.À.n.

Figure 20: Code snippet written to heap memory region for API call

The byte array used to resolve and call the addresses of the API functions to be used by Lockbit is indicated in the image above.

```
026B05F0    B8 F44ED064     mov eax,64D04EF4
026B05F5    C1C8 01         ror eax,1
026B05F8    35 CADF0645     xor eax,4506DFCA
026B05FD    FFE0            jmp eax
026B05FF    BA ABABABAB     mov edx,ABABABAB
```

Figure 21: Code block used for API address resolution

When ROR and XOR operations are applied to the value transferred to the EAX register, the API function's address in the relevant library is obtained.

roeax, 1 ; eax = 3268277A
xor eax, 4506DFCA ; eax = 776EF8B0

To see the API call corresponding to the value of 0x64D04EF4, when we check the value of the EAX register (776EF8B0) obtained as a result of the arithmetic operations performed, with the help of the debugger, it corresponds to the RtlDestroyHeap API function. Note that these addresses may change on a different computer and subsequent runs.

```
776EF8B0    8BFF            mov edi,edi                         RtlDestroyHeap
776EF8B2    55              push ebp
776EF8B3    8BEC            mov ebp,esp
776EF8B5    83E4 F8         and esp,FFFFFFF8
776EF8B8    83EC 64         sub esp,64
776EF8BB    A1 70B37E77     mov eax,dword ptr ds:[777EB370]
776EF8C0    33C4            xor eax,esp
776EF8C2    894424 60       mov dword ptr ss:[esp+60],eax
776EF8C6    53              push ebx
776EF8C7    56              push esi
776EF8C8    57              push edi
776EF8C9    8B7D 08         mov edi,dword ptr ss:[ebp+8]
776EF8CC    85FF            test edi,edi
```

Figure 22: Function corresponding to the resolved API address

The data written to the heap memory region may not always contain the arithmetic operations shown above. Therefore, when choosing the data to be written to the heap, Lockbit selects the byte sequences to be written by generating random numbers between 1-4.

The byte array to be written is determined by checking the random value produced in the sub_401120 function. We have shown code pieces that show the byte arrays that can be written for values from 1 to 4 in the images below. Note that the bytes shown here begin with the opcode value B8 corresponding to the MOV instruction before being written to memory.

```
.text:00407CEF 6A 09                    push    9
.text:00407CF1 6A 01                    push    1
.text:00407CF3 E8 28 94 FF FF           call    sub_401120
.text:00407CF8 8B C8                    mov     ecx, eax
.text:00407CFA D3 CB                    ror     ebx, cl
.text:00407CFC 89 5A 01                 mov     [edx+1], ebx
.text:00407CFF 66 C7 42 05 C1 C0        mov     word ptr [edx+5], 0C0C1h
.text:00407D05 88 4A 07                 mov     [edx+7], cl
.text:00407D08 66 C7 42 08 FF E0        mov     word ptr [edx+8], 0E0FFh
.text:00407D0E E9 84 00 00 00           jmp     loc_407D97
```

Figure 23: Byte array to be used if the randomly generated value is 1

```
.text:00407D18 B8 CA DF 06 45           mov     eax, 4506DFCAh
.text:00407D1D 33 D8                    xor     ebx, eax
.text:00407D1F 89 5A 01                 mov     [edx+1], ebx
.text:00407D22 C6 42 05 35              mov     byte ptr [edx+5], 35h ; '5'
.text:00407D26 89 42 06                 mov     [edx+6], eax
.text:00407D29 66 C7 42 0A FF E0        mov     word ptr [edx+0Ah], 0E0FFh
.text:00407D2F EB 66                    jmp     short loc_407D97
```

Figure 24: Byte array to be used if the randomly generated value is 2

```
.text:00407D36 6A 09                    push    9
.text:00407D38 6A 01                    push    1
.text:00407D3A E8 E1 93 FF FF           call    sub_401120
.text:00407D3F 8B C8                    mov     ecx, eax
.text:00407D41 B8 CA DF 06 45           mov     eax, 4506DFCAh
.text:00407D46 33 D8                    xor     ebx, eax
.text:00407D48 D3 C3                    rol     ebx, cl
.text:00407D4A 89 5A 01                 mov     [edx+1], ebx
.text:00407D4D 66 C7 42 05 C1 C8        mov     word ptr [edx+5], 0C8C1h
.text:00407D53 88 4A 07                 mov     [edx+7], cl
.text:00407D56 C6 42 08 35              mov     byte ptr [edx+8], 35h ; '5'
.text:00407D5A 89 42 09                 mov     [edx+9], eax
.text:00407D5D 66 C7 42 0D FF E0        mov     word ptr [edx+0Dh], 0E0FFh
.text:00407D63 EB 32                    jmp     short loc_407D97
```

Figure 25: Byte array to be used if the randomly generated value is 3

```
.text:00407D6A 6A 09                   push    9
.text:00407D6C 6A 01                   push    1
.text:00407D6E E8 AD 93 FF FF          call    sub_401120
.text:00407D73 8B C8                   mov     ecx, eax
.text:00407D75 B8 CA DF 06 45          mov     eax, 4506DFCAh
.text:00407D7A 33 D8                   xor     ebx, eax
.text:00407D7C D3 CB                   ror     ebx, cl
.text:00407D7E 89 5A 01                mov     [edx+1], ebx
.text:00407D81 66 C7 42 05 C1 C0       mov     word ptr [edx+5], 0C0C1h
.text:00407D87 88 4A 07                mov     [edx+7], cl
.text:00407D8A C6 42 08 35             mov     byte ptr [edx+8], 35h ; '5'
.text:00407D8E 89 42 09                mov     [edx+9], eax
.text:00407D91 66 C7 42 0D FF E0       mov     word ptr [edx+0Dh], 0E0FFh
```

Figure 26: Byte array to be used if the randomly generated value is 4

Questionable API functions resolved by Lockbit 3.0

NTDLL

- RtlReAllocateHeap
- NtOpenProcess
- ZwSetThreadExecutionState
- ZwSetInformationProcess
- ZwQuerySystemInformation
- NtQuerySystemInformationProcess
- ZwQueryInformationToken
- NtSetInformationToken
- NtSetInformationThread
- NtOpenProcessToken
- NtShutdownSystem
- RtlAdjustPrivilege
- LdrEnumerateLoadedModules
- ZwTerminateProcess
- ZwTerminateThread
- NtPrivilegeCheck
- ZwWriteVirtualMemory
- NtReadVirtualMemory
- ZwProtectVirtualMemory
- NtAllocateVirtualMemory
- NtQueryInstallUILanguage
- ZwQueryDefaultUILanguage

KERNEL32

- FindFirstFileExW
- FindNextFileW
- SetFileAttributesW
- CopyFileW
- MoveFileExW
- CreateThread
- CreateRemoteThread
- ResumeThread
- CreateFileW
- WriteFile
- ReadFile
- WinExec
- Sleep
- SetFilePointerEx
- GetLogicalDriveStringsW
- GetDriveTypeW
- GetDiskFreeSpaceExW
- DeleteFileW
- CreateDirectoryW
- RemoveDirectoryW
- OpenMutexW
- CreateMutexW
- GetCurrentDirectoryW
- SetCurrentDirectoryW
- GetTickCount
- GetComputerNameW
- SetVolumeMountPointW
- SetThreadPriority
- GetVolumePathNameW
- FindFirstVolumeW
- FindNextVolumeW
- DeviceIoControl
- GetVolumePathNamesForVolumeNameW
- GetVolumeNameForVolumeMountPointW
- CreateProcessW
- CreateNamedPipeW
- ConnectNamedPipeW
- GetTempFileNameW

ADVAPI32

- RegCreateKeyExW
- RegSetValueExW
- RegQueryValueExW
- RegDeleteKeyExW
- RegDeleteKeyW
- RegEnumKeyW
- OpenSCManagerW
- EnumServicesStatusExW
- OpenServiceW
- CreateServiceW
- StartServiceW
- SetServiceStatus
- LogonUserW
- GetUserNameW
- ControlService
- DeleteService
- LsaOpenPolicy
- LsaStorePrivateData

WININET

- InternetOpenW
- InternetConnectW
- InternetSetOptionW
- InternetQueryOptionW
- InternetCloseHandle
- HttpQueryInfoW
- HttpOpenRequestW
- HttpSendRequestW
- InternetQueryDataAvailable
- InternetReadFile

COMBASE

- CoInitializeSecurity
- CoCreateInstance
- CoCreateInstanceEx
- CoInitialize
- CoInitializeEx
- CoGetObject

WS2_32

- WSAStartup
- WSACleanup
- gethostbyname

WINSPOOL

- OpenPrinterW
- ClosePrinter
- EnumPrintersW
- DocumentPropertiesW

Anti-Analysis Techniques

After the API addresses are to be called and the arithmetic operations required for resolution are written to the heap, the sub_40D2D5 routine is responsible for calling the functions in a heap. The first API function called is NtSetInformationThread. The value used for the ThreadInformationClass parameter of this function is 0x11 with the symbolic name ThreadHideFromDebugger. That indicates that the NtSetInformationThread API function has been used to hide thread activities from the debugger.

Another control Lockbit has implemented to prevent analysis is the ProcessHeap field of the PEB data structure. There are two areas where the heap is affected in the presence of a debugger. The values they get depend on the version of Windows. These fields are Flags and ForceFlags. Under normal conditions (no debugger), the Flags and ForceFlags fields take values HEAP_GROWABLE and 0, respectively.

In the case of a debugger found, Flags and ForceFlags respectively can have a combination of the following values.

Flags

HEAP_GROWABLE (2)
HEAP_TAIL_CHECKING_ENABLED (0x20)
HEAP_FREE_CHECKING_ENABLED (0x40)
HEAP_VALIDATE_PARAMETERS_ENABLED (0x40000000)

ForceFlags

0
HEAP_TAIL_CHECKING_ENABLED (0x20)
HEAP_FREE_CHECKING_ENABLED (0x40)
HEAP_VALIDATE_PARAMETERS_ENABLED (0x40000000)

```
004086F8    55              push ebp
004086F9    8BEC            mov ebp,esp
004086FB    E8 9C89FFFF     call unknown.40109C
00408700    8B40 18         mov eax,dword ptr ds:[eax+18]
00408703    F740 44 00000040 test dword ptr ds:[eax+44],40000000
0040870A    74 02           je unknown.40870E
0040870C    D1C8            ror eax,1
0040870E    FF75 08         push dword ptr ss:[ebp+8]
00408711    6A 08           push 8
00408713    50              push eax
00408714    FF15 14744200   call dword ptr ds:[427414]
0040871A    5D              pop ebp
0040871B    C2 0400         ret 4
```

Figure 27: Process Heap Flags anti-debug check

Another anti-analysis technique Lockbit applies is to delete the running malicious program file from the file system.

Mutex Creation

When Lockbit is run on the target system, it creates a Mutex object with the value

"Global\2cae82bd1366f4e0fdc7a9a7c12e2a6b".

Control of Running Services

Lockbit 3.0 attempts to detect and disable running services associated with Windows security and the Event Log mechanism at runtime. The EnumServiceStatusEx function determines the current services on the target system and their status.

If a service is to be disabled, the ControlService function is called using the handle obtained for that service from the OpenServiceW call. The ControlService call uses the control code 0x00000001 with the symbolic name SERVICE_CONTROL_STOP to indicate that the service should stop. Then the Service Control Manager calls the DeleteServiceW function to delete it from the Database.

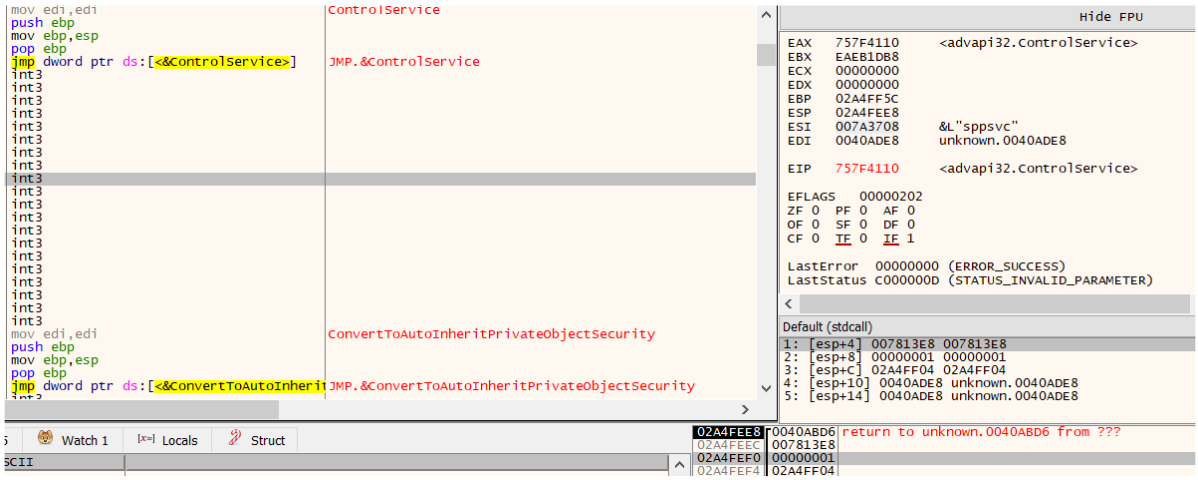


Figure 28: Stopping running service with ControlService API call

Below we have listed the services we found that Lockbit is checking.

- TrustedInstaller
- SecurityHealthService
- sppsvc
- Wdboot
- Wdfilter
- WdNisDrv
- WdNisSvc
- WinDefend
- wscsvc
- vmicvss
- vmvss
- vss
- vsstandardcollectorservice150

Resolving of Strings

During the execution of Lockbit 3.0, the required strings are in the encrypted stack. Strings are parsed in the alt_401260 routine. The string encryption process is completed by repairing the encrypted data with 4506DFCA's and NOT ending immediately.

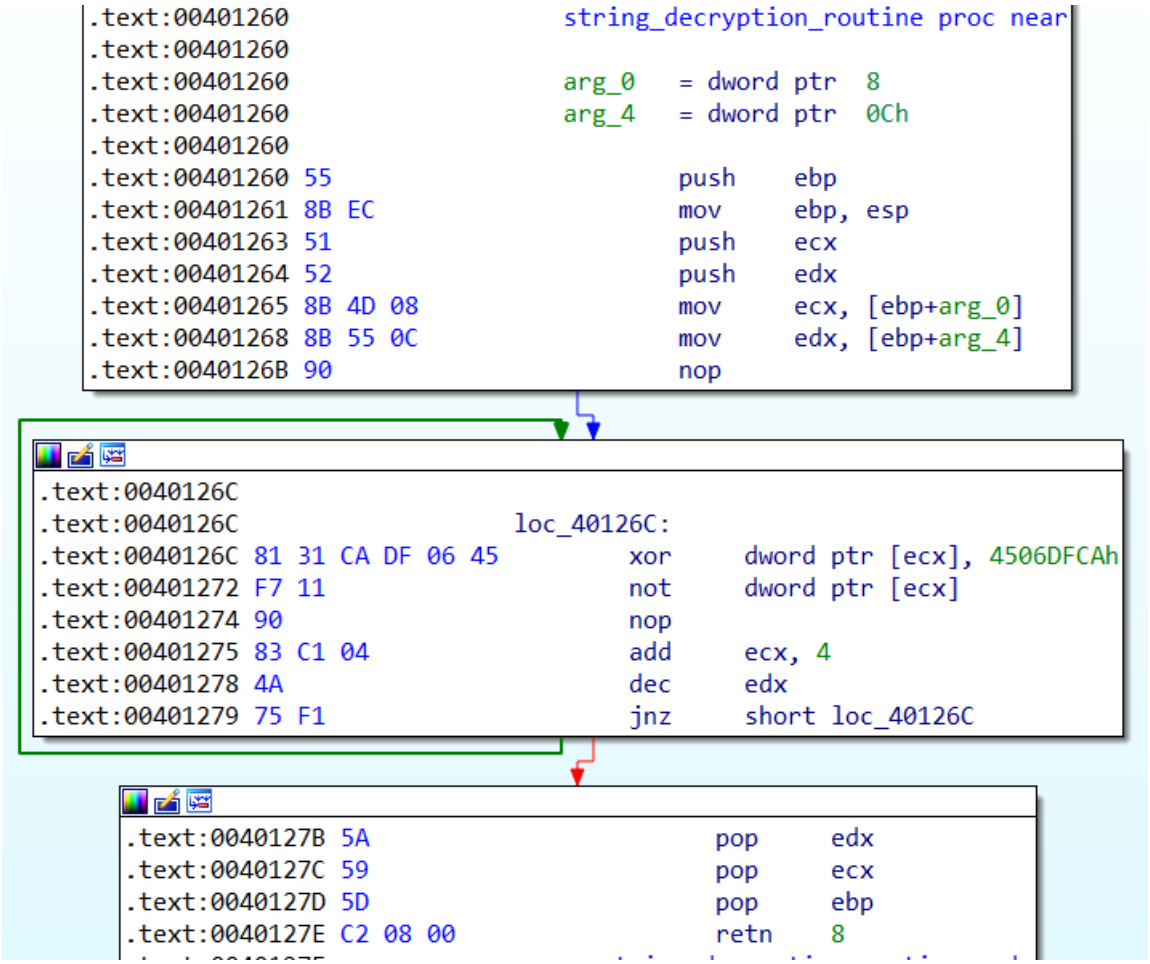


Figure 29: The piece of code responsible for String parsing

Configuration Data

Lockbit 3.0 keeps the important configuration data it needs at runtime in encrypted form.

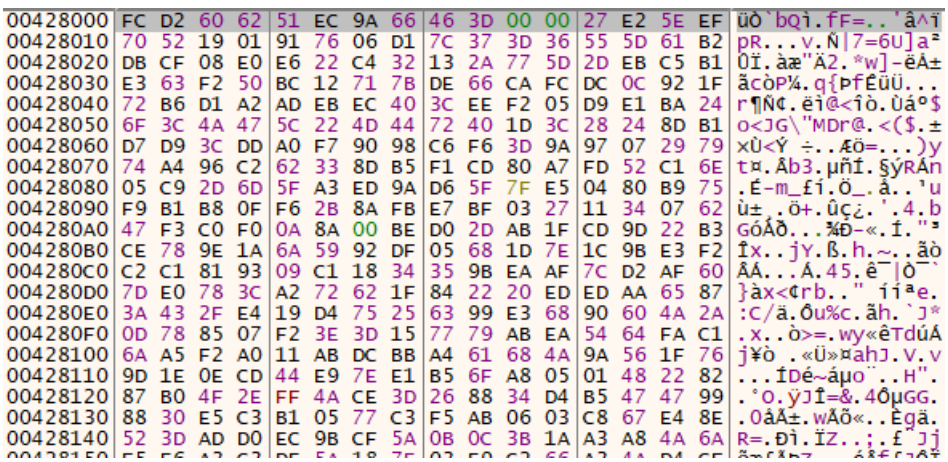


Figure 30: Encrypted configuration data (found in .pdata section)

The configuration data is divided into the following offsets and dimensions.

+00h: RSA-1024 (1024 bit) key (80h bytes)
+80h: NULL bytes (20h bytes)
+A0h: True (01h) or False(00h) Boolean values (24 bytes)
+E0h: Base64 encoded data block

As seen in the +E0h offset, the configuration data includes Base64 data. The Base64 code block contains different types of data, separated by a value of 0x00. Next, the lengths of the read Base64 data are calculated, the heap memory area is allocated accordingly, and the decoded data is written to the memory area allocated with RtlAllocateHeap. The routine is responsible for decoding base64 data and writing to the memory allocated memory region labeled sub_401304.

Below, we have explained the usage purposes we can detect as a result of decoding Base64 code blocks.

- The following expressions are resolved by hash analysis with 4-byte hash values of folders that Lockbit will not include in encryption processes. The piece of code where the check is made is in the sub_00410DD0 routine. We can list MSOCache, PerfLogs, Program Files, Program Files (x86), ProgramData, Python27, Python310, Windows, Default, Public, AppData, Users, etc.

- 4-byte hash values of system files that Lockbit will not include in encryption operations: autorun.inf, desktop.in etc.

- File extensions to be excluded from encryption operations: msi, sys, etc.

- Clear text list of processes that Lockbit will try to terminate at runtime: sql, oracle, ocssd, dbsnmp, synctime, agntsvc, isqlplussvc, xfssvccon, mydesktopservice, ocautoupds, encsvc, firefox, tbirdconfig, mydesktopqos, ocomm, ddbpath, excelservice, msqbeng50 , mspub, onenote, outlook, powerpnt, steam, thebat, thunderbird, visio, winword, wordpad, notepad

- List of services Lockbit looks at to disable and delete:
vss, sql, svc$, memtas, mepocs, msexchange, sophos, veeam, backup, GxVss, GxBlr, GxFWD, GxCVD, GxCIMgr

```
00785DC8 73 00 71 00 6C 00 00 00 6F 00 72 00 61 00 63 00 s.q.l...o.r.a.c.
00785DD8 6C 00 65 00 00 00 6F 00 63 00 73 00 73 00 64 00 l.e...o.c.s.s.d.
00785DE8 00 00 64 00 62 00 73 00 6E 00 6D 00 70 00 00 00 ..d.b.s.n.m.p...
00785DF8 73 00 79 00 6E 00 63 00 74 00 69 00 6D 00 65 00 s.y.n.c.t.i.m.e.
00785E08 00 00 61 00 67 00 6E 00 74 00 73 00 76 00 63 00 ..a.g.n.t.s.v.c.
00785E18 00 00 69 00 73 00 71 00 6C 00 70 00 6C 00 75 00 ..i.s.q.l.p.l.u.
00785E28 73 00 73 00 76 00 63 00 00 00 78 00 66 00 73 00 s.s.v.c...x.f.s.
00785E38 73 00 76 00 63 00 63 00 6F 00 6E 00 00 00 6D 00 s.v.c.c.o.n...m.
00785E48 79 00 64 00 65 00 73 00 6B 00 74 00 6F 00 70 00 y.d.e.s.k.t.o.p.
00785E58 73 00 65 00 72 00 76 00 69 00 63 00 65 00 00 00 s.e.r.v.i.c.e...
00785E68 6F 00 63 00 61 00 75 00 74 00 6F 00 75 00 70 00 o.c.a.u.t.o.u.p.
00785E78 64 00 73 00 00 00 65 00 6E 00 63 00 73 00 76 00 d.s...e.n.c.s.v.
00785E88 63 00 00 00 66 00 69 00 72 00 65 00 66 00 6F 00 c...f.i.r.e.f.o.
00785E98 78 00 00 00 74 00 62 00 69 00 72 00 64 00 63 00 x...t.b.i.r.d.c.
00785EA8 6F 00 6E 00 66 00 69 00 67 00 00 00 6D 00 79 00 o.n.f.i.g...m.y.
00785EB8 64 00 65 00 73 00 6B 00 74 00 6F 00 70 00 71 00 d.e.s.k.t.o.p.q.
00785EC8 6F 00 73 00 00 00 6F 00 63 00 6F 00 6D 00 6D 00 o.s...o.c.o.m.m.
00785ED8 00 00 64 00 62 00 65 00 6E 00 67 00 35 00 30 00 ..d.b.e.n.g.5.0.
00785EE8 00 00 73 00 71 00 62 00 63 00 6F 00 72 00 65 00 ..s.q.b.c.o.r.e.
```

Figure 31: Example of decrypted Base64 encoded configuration data - 1

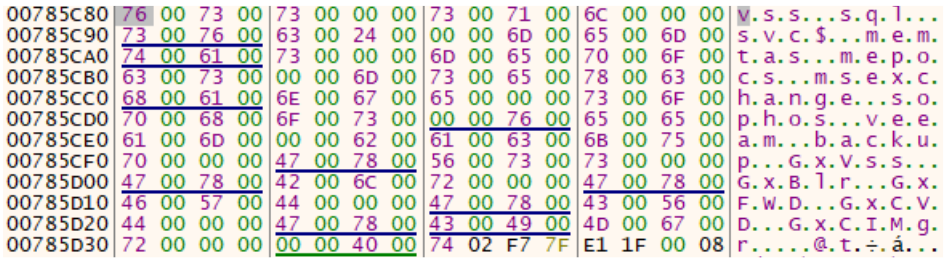


Figure 32: Example of decrypted Base64 encoded configuration data - 2

- Ransom note

The text of the ransom notes left on the target file system by Lockbit is encrypted in the configuration data and decrypted before use. After the ransom note-related operations are completed, the clear-text ransom note in memory is encrypted and hidden again.

Computer Language Control

Like other ransomware, Lockbit checks the computer language used on the infected computer. In addition, Lockbit checks the computer language to avoid infecting computers in Russia and nearby countries. It analyzes the GetSystemDefaultUILanguage and GetUserDefaultUILanguage API functions during API analysis.

Below are the countries excluded by Lockbit 3.0.

- Azerbaijani (Cyrillic Azerbaijan)
- Azerbaijani (Latin Azerbaijani)
- Armenian (Armenia)
- Belarusian (Belarusian)
- Georgian (Georgia)
- Kazakh (Kazakhstan)
- Kyrgyz (Kyrgyzstan)
- Russian (Moldova)
- Russian (Russia)
- Tajik (Cyrillic Tajikistan)
- Turkmen (Turkmenistan)
- Uzbek (Cyrillic Uzbekistan)
- Uzbek (Latin Uzbekistan)
- Ukrainian

MITRE ATT&CK Threat Matrix

The table below contains the techniques, tactics and procedures used by the Lockbit ransomware threat actor.

Tactic Name	ID	Technique Name	ID
Initial Access	TA0001	Drive-by Compromise Valid Accounts	T1189 T1078
Execution	TA0002	Command and Scripting Interpreter: Windows Command Shell Command and Scripting Interpreter: PowerShell System Services: Service Execution Windows Management Instrumentation	T1059.003 T1059.001 T1569.002 T1047
Privilege Escalation	TA0004	Process Injection Abuse Elevation Control Mechanism: Bypass User Account Control	T1055 T1548.002
Defence Evasion	TA0005	Impair Defenses: Disable or Modify Tools Indicator Removal on Host: Clear Windows Event Logs Modify Registry Masquerading Software Packing File Deletion Debugger Evasion	T1562.001 T1070.001 T1112 T1036 T1027.002 T1070.004 T1622
Credential Access	TA0006	Credential API Hooking	T1056.004
Discovery	TA0007	Query Registry Process Discovery System Information Discovery	T1012 T1057 T1082
Collection	TA0009	Automated Collection	T1119
Command and Control	TA0011	Encrypted Channel	T1573
Impact	TA0040	Service Stop Data Destruction Inhibit System Recovery	T1489 T1485 T1490

YARA Rules

```
import "pe"

rule Lockbit30{
        meta:
                description = "Rule detecting Lockbit 3.0 ransomware samples"

        strings:
                $s1 = {50 57 E8 [4] 85 C0 75 ?? EB ?? 8D ?? ?? 6A ?? 8D [5] 50 E8 [4] 3D
                        [4] 75 ?? 83 ?? ??}

        condition:
                uint16(0)==0x5A4D and filesize < 500KB and all of them
}
```

Below are links to YARA rules created by other security providers.

- https://blogs.blackberry.com/en/2022/08/lockbit-3-0-ransomware-abuses-windows-defender-to-load-cobalt-strike

Indicator of Compromises

Table 1: Lockbit 3.0 samples

Hash (MD5/SHA1/SHA256)	Description
0d38f8bf831f1dbbe9a058930127171f24c3df8dae81e6aa66c430a63cbe0509	Lockbit 3.0
9a34909703d679b590d316eb403e12e26f73c8e479812f1d346dcba47b44bc6e	Lockbit 3.0
39c363d01fb5cd0ed3eeb17ca47be0280d93a07dda9bc0236a0f11b20ed95b4c	Lockbit 3.0
80e8defa5377018b093b5b90de0f2957f7062144c83a09a56bba1fe4eda932ce	Lockbit 3.0
391a97a2fe6beb675fe350eb3ca0bc3a995fda43d02a7a6046cd48f042052de5	Lockbit 3.0
80e8defa5377018b093b5b90de0f2957f7062144c83a09a56bba1fe4eda932ce	Lockbit 3.0
391a97a2fe6beb675fe350eb3ca0bc3a995fda43d02a7a6046cd48f042052de5	Lockbit 3.0
506f3b12853375a1fbbf85c82ddf13341cf941c5acd4a39a51d6addf145a7a51	Lockbit 3.0
742489bd828bdcd5caaed00dccdb7a05259986801bfd365492714746cb57eb55	Lockbit 3.0
a56b41a6023f828cccaaef470874571d169fdb8f683a75edd430fbd31a2c3f6e	Lockbit 3.0
b951e30e29d530b4ce998c505f1cb0b8adc96f4ba554c2b325c0bd90914ac944	Lockbit 3.0
c6cf5fd8f71abaf5645b8423f404183b3dea180b69080f53b9678500bab6f0de	Lockbit 3.0
d61af007f6c792b8fb6c677143b7d0e2533394e28c50737588e40da475c040ee	Lockbit 3.0
f9b9d45339db9164a3861bf61758b7f41e6bcfb5bc93404e296e2918e52ccc10	Lockbit 3.0
fd98e75b65d992e0ccc64e512e4e3e78cb2e08ed28de755c2b192e0b7652c80a	Lockbit 3.0