A technical analysis of the leaked LockBit 3.0 builder

cybergeeks.tech/a-technical-analysis-of-the-leaked-lockbit-3-0-builder/

Summary

This is our analysis of the LockBit 3.0 builder that was leaked online on September 21, 2022. The executable called “keygen.exe” can be used to generate the RSA public and private keys that are embedded in the encryptor and decryptor, respectively. The builder embedded 4 resources used to create executables or DLL files according to the command line parameters. As in the case of Conti leaks, we’ll probably encounter LockBit-forked ransomware because of the builder’s availability.

Analyst: @GeeksCyber

Technical analysis

SHA256:
A736269F5F3A9F2E11DD776E352E1801BC28BB699E47876784B8EF761E0062DB

The builder (“builder.exe”) was compiled on September 13, 2022. The executable “keygen.exe” can be used to generate RSA public and private keys that are saved as “pub.key” and “priv.key”.

The RSA public/private key is Base64-encoded, as highlighted below:

Figure 1
The process retrieves the command-line string using GetCommandLineW:

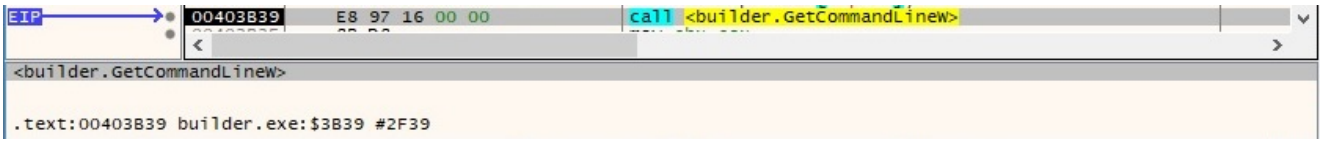


Figure 2
The CommandLineToArgvW API is utilized to obtain an array of pointers to the command line arguments:

Figure 3
Running with the **-type dec -privkey priv.key -config config.json -ofile LB3Decryptor.exe** parameters

The malware compares the parameters with “-type enc” (encryptor) and “-type dec” (decryptor) to decide which executable to generate:

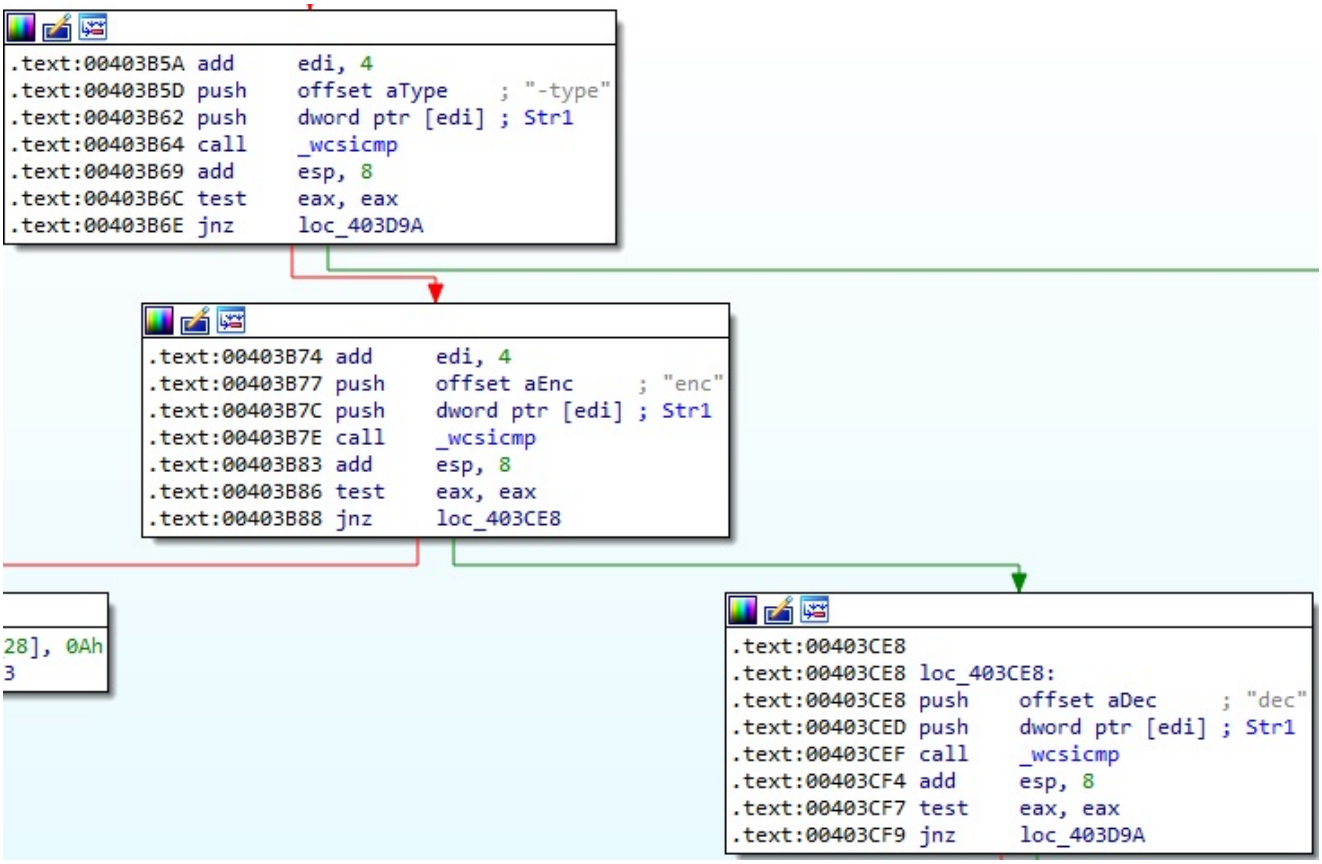


Figure 4

The builder opens the RSA private key file by calling the CreateFileW function (0x80000000 = **GENERIC_READ**, 0x1 = **FILE_SHARE_READ**, 0x3 = **OPEN_EXISTING**, 0x80 = **FILE_ATTRIBUTE_NORMAL**):

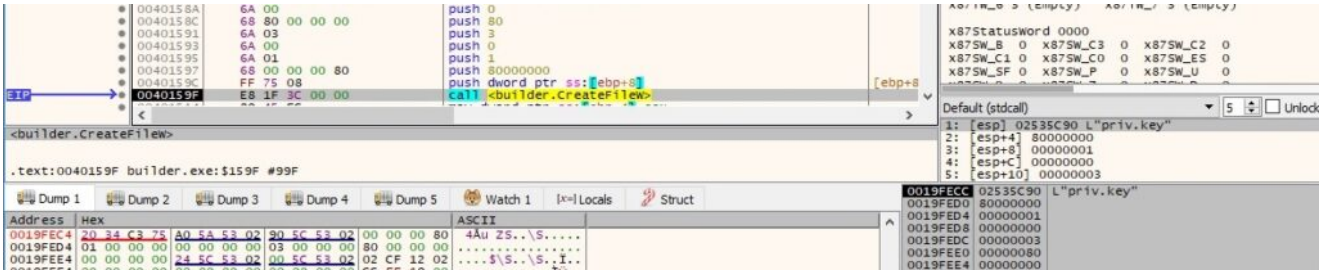


Figure 5

The process reads the above file content using the ReadFile API:

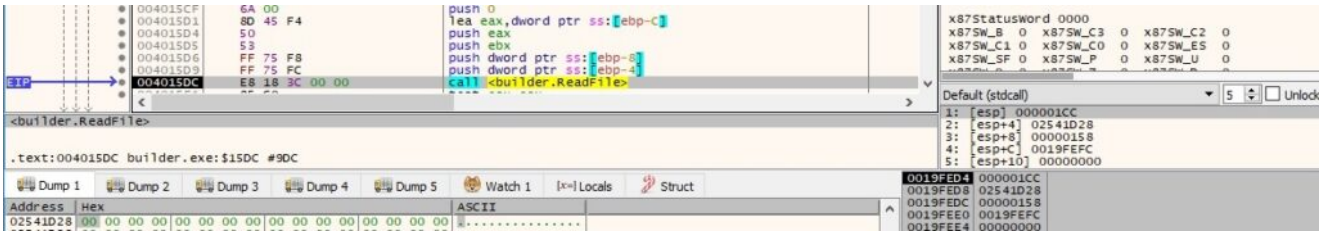
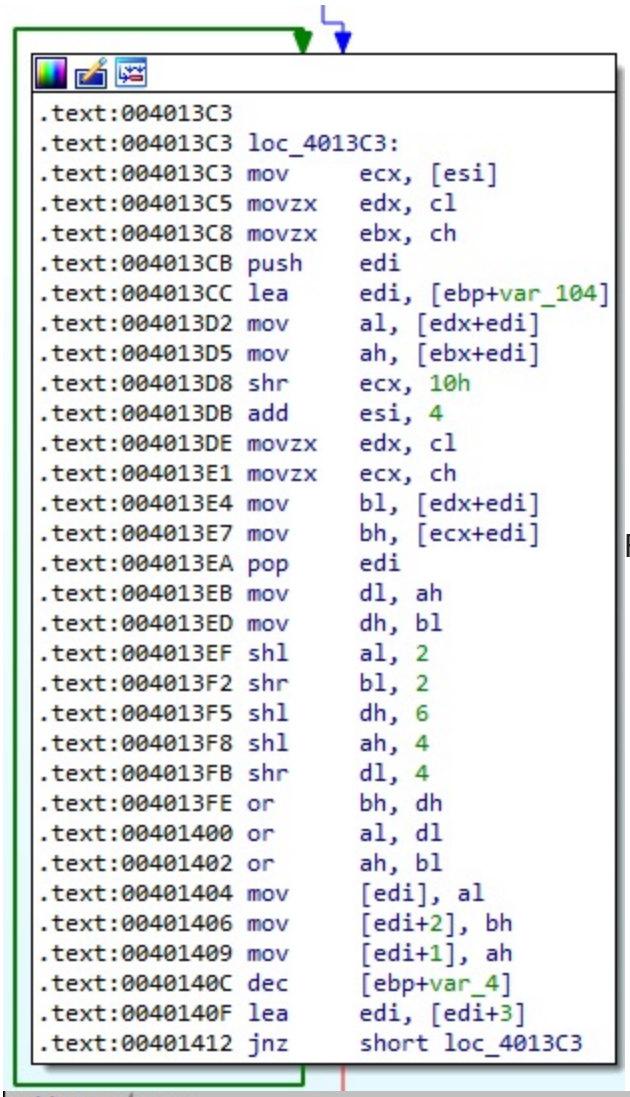


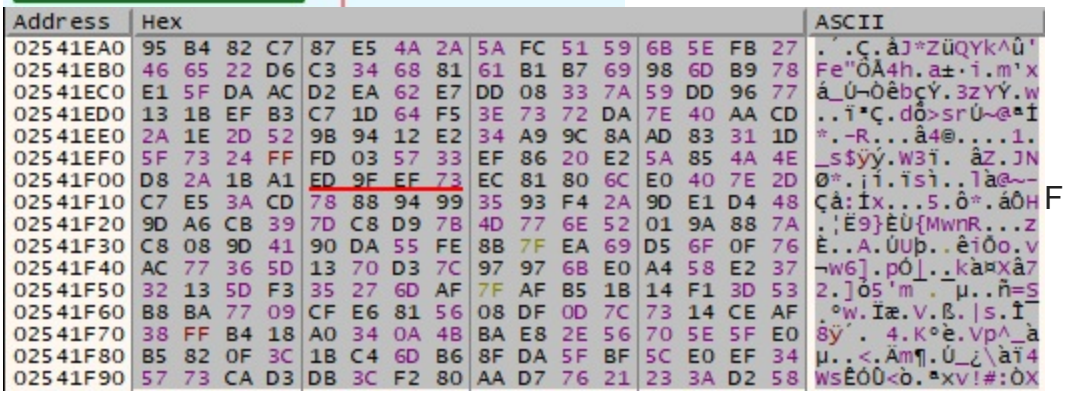
Figure 6

The RSA private key is Base64-decoded by the malicious process:



```
.text:004013C3
.text:004013C3 loc_4013C3:
.text:004013C3 mov     ecx, [esi]
.text:004013C5 movzx   edx, cl
.text:004013C8 movzx   ebx, ch
.text:004013CB push    edi
.text:004013CC lea     edi, [ebp+var_104]
.text:004013D2 mov     al, [edx+edi]
.text:004013D5 mov     ah, [ebx+edi]
.text:004013D8 shr     ecx, 10h
.text:004013DB add     esi, 4
.text:004013DE movzx   edx, cl
.text:004013E1 movzx   ecx, ch
.text:004013E4 mov     bl, [edx+edi]
.text:004013E7 mov     bh, [ecx+edi]
.text:004013EA pop     edi
.text:004013EB mov     dl, ah
.text:004013ED mov     dh, bl
.text:004013EF shl     al, 2
.text:004013F2 shr     bl, 2
.text:004013F5 shl     dh, 6
.text:004013F8 shl     ah, 4
.text:004013FB shr     dl, 4
.text:004013FE or      bh, dh
.text:00401400 or      al, dl
.text:00401402 or      ah, bl
.text:00401404 mov     [edi], al
.text:00401406 mov     [edi+2], bh
.text:00401409 mov     [edi+1], ah
.text:0040140C dec     [ebp+var_4]
.text:0040140F lea     edi, [edi+3]
.text:00401412 jnz     short loc_4013C3
```

Figure 7



Address	Hex	ASCII
02541EA0	95 B4 82 C7 87 E5 4A 2A 5A FC 51 59 6B 5E FB 27	.´.Ç.åJ*ZüQYk^û'
02541EB0	46 65 22 D6 C3 34 68 81 61 B1 B7 69 98 6D B9 78	Fe"ÖÃ4h.a±·i.m'x
02541EC0	E1 5F DA AC D2 EA 62 E7 DD 08 33 7A 59 DD 96 77	á_Ú¬ÒêbçÝ.3zYÝ.w
02541ED0	13 1B EF B3 C7 1D 64 F5 3E 73 72 DA 7E 40 AA CD	..ï³Ç.dõ>srÚ~@ªÍ
02541EE0	2A 1E 2D 52 9B 94 12 E2 34 A9 9C 8A AD 83 31 1D	*.-R...â4©.....1.
02541EF0	5F 73 24 FF FD 03 57 33 EF 86 20 E2 5A 85 4A 4E	_s$ÿý.W3ï. âZ.JN
02541F00	D8 2A 1B A1 ED 9F EF 73 EC 81 80 6C E0 40 7E 2D	Ø*.¡í.ïsì..là@~-
02541F10	C7 E5 3A CD 78 88 94 99 35 93 F4 2A 9D E1 D4 48	Çå:Íx...5.ô*.áÔH
02541F20	9D A6 CB 39 7D C8 D9 7B 4D 77 6E 52 01 9A 88 7A	.¦Ë9}ÈÙ{MwnR...z
02541F30	C8 08 9D 41 90 DA 55 FE 8B 7F EA 69 D5 6F 0F 76	È..A.ÚUþ..êiÕo.v
02541F40	AC 77 36 5D 13 70 D3 7C 97 97 6B E0 A4 58 E2 37	¬w6].pÓ|..kà¤Xâ7
02541F50	32 13 5D F3 35 27 6D AF 7F AF B5 1B 14 F1 3D 53	2.]ó5'm¯.¯µ..ñ=S
02541F60	B8 BA 77 09 CF E6 81 56 08 DF 0D 7C 73 14 CE AF	¸ºw.Ïæ.V.ß.|s.Î¯
02541F70	38 FF B4 18 A0 34 0A 4B BA E8 2E 56 70 5E 5F E0	8ÿ´. 4.Kºè.Vp^_à
02541F80	B5 82 0F 3C 1B C4 6D B6 8F DA 5F BF 5C E0 EF 34	µ..<.Äm¶.Ú_¿\àï4
02541F90	57 73 CA D3 DB 3C F2 80 AA D7 76 21 23 3A D2 58	WsÊÓÛ<ò.ª×v!#:ÒX

Figure 8

The executable parses the LockBit configuration file “config.json” that contains information such as the whitelisted folders/files/extensions, the processes and services to stop, and the ransom note content:

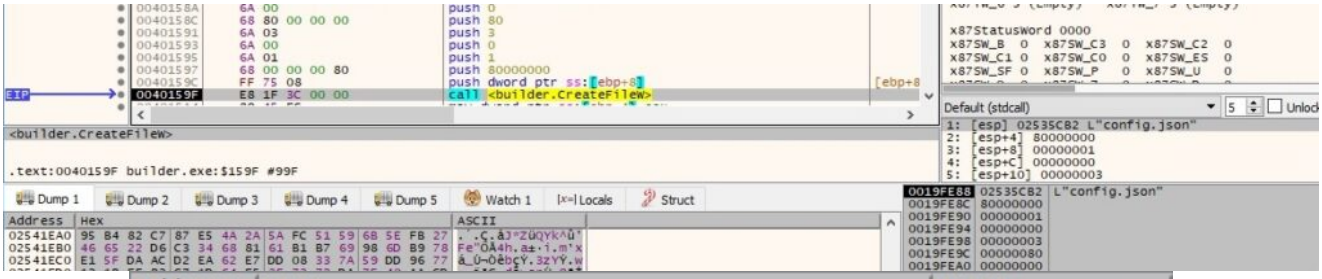
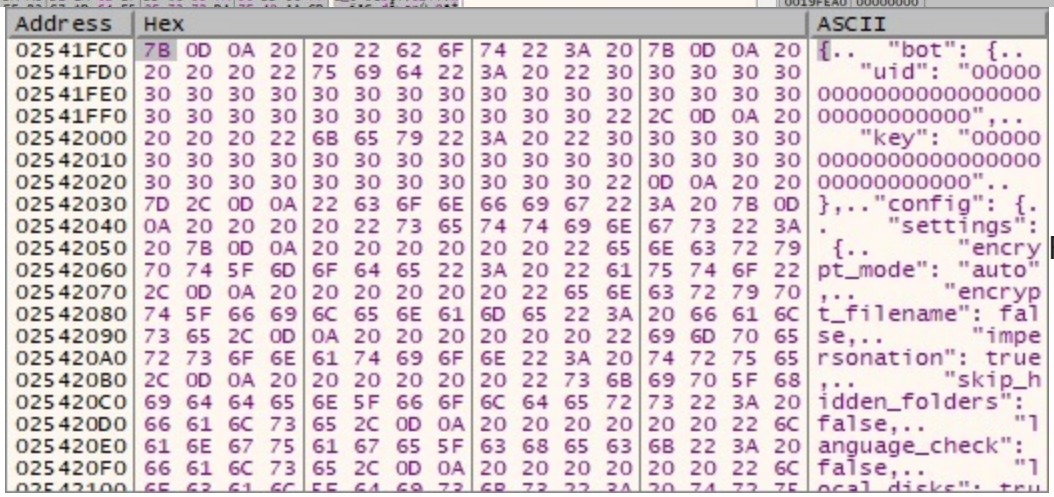


Figure 9



Address	Hex	ASCII
02541FC0	7B 0D 0A 20 20 22 62 6F 74 22 3A 20 7B 0D 0A 20	{.. "bot": {..
02541FD0	20 20 20 22 75 69 64 22 3A 20 22 30 30 30 30 30	"uid": "00000
02541FE0	30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30	0000000000000000
02541FF0	30 30 30 30 30 30 30 30 30 30 30 22 2C 0D 0A 20	00000000000",..
02542000	20 20 20 22 6B 65 79 22 3A 20 22 30 30 30 30 30	"key": "00000
02542010	30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30	0000000000000000
02542020	30 30 30 30 30 30 30 30 30 30 30 22 0D 0A 20 20	00000000000"..
02542030	7D 2C 0D 0A 22 63 6F 6E 66 69 67 22 3A 20 7B 0D	},.."config": {.
02542040	0A 20 20 20 20 22 73 65 74 74 69 6E 67 73 22 3A	. "settings":
02542050	20 7B 0D 0A 20 20 20 20 20 20 22 65 6E 63 72 79	{.. "encry
02542060	70 74 5F 6D 6F 64 65 22 3A 20 22 61 75 74 6F 22	pt_mode": "auto"
02542070	2C 0D 0A 20 20 20 20 20 20 22 65 6E 63 72 79 70	,.. "encryp
02542080	74 5F 66 69 6C 65 6E 61 6D 65 22 3A 20 66 61 6C	t_filename": fal
02542090	73 65 2C 0D 0A 20 20 20 20 20 20 22 69 6D 70 65	se,.. "impe
025420A0	72 73 6F 6E 61 74 69 6F 6E 22 3A 20 74 72 75 65	rsonation": true
025420B0	2C 0D 0A 20 20 20 20 20 20 22 73 6B 69 70 5F 68	,.. "skip_h
025420C0	69 64 64 65 6E 5F 66 6F 6C 64 65 72 73 22 3A 20	idden_folders":
025420D0	66 61 6C 73 65 2C 0D 0A 20 20 20 20 20 20 22 6C	false,.. "l
025420E0	61 6E 67 75 61 67 65 5F 63 68 65 63 6B 22 3A 20	anguage_check":
025420F0	66 61 6C 73 65 2C 0D 0A 20 20 20 20 20 20 22 6C	false,.. "l

Figure 10

The malware implements a custom “hashing” function that computes a 4-byte value for each whitelisted directory/file/extension/host. An example of a function result is shown in figure 12.

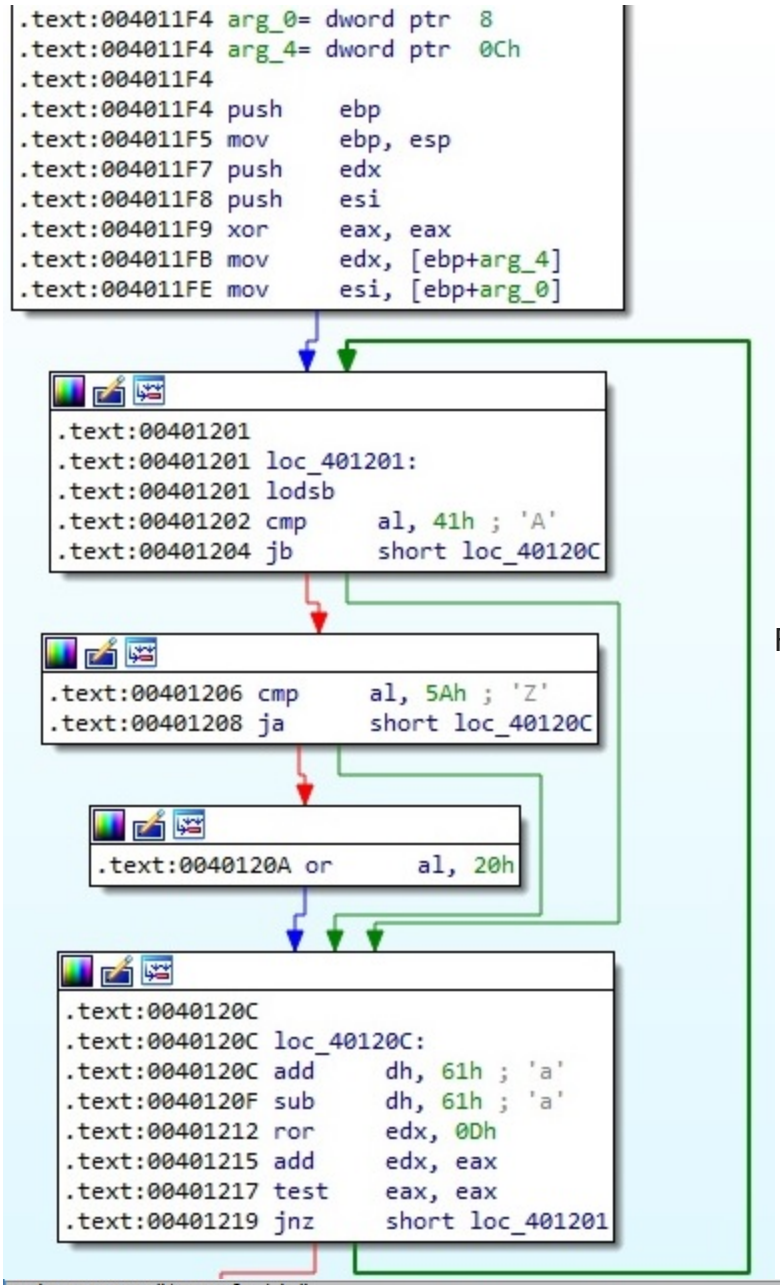


Figure 11

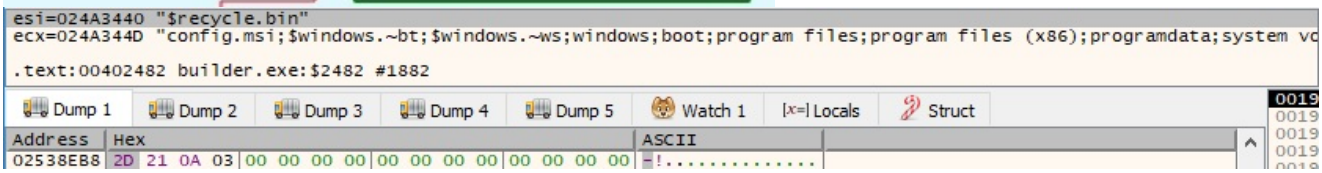
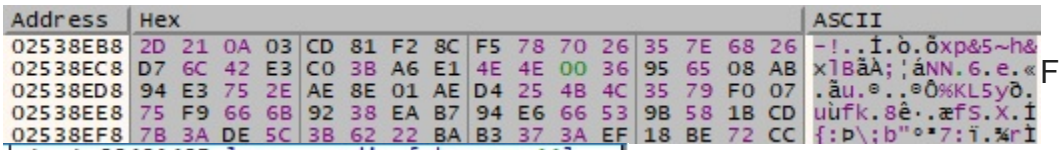


Figure 12

The resulting buffer containing the hashes is Base64-encoded by the builder, as shown in the figure below.

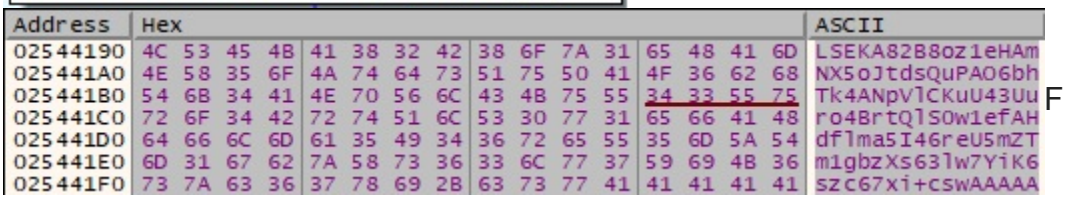


Address	Hex	ASCII
02538EB8	2D 21 0A 03 CD 81 F2 8C F5 78 70 26 35 7E 68 26	-!..Í.ò.õxp&5~h&
02538EC8	D7 6C 42 E3 C0 3B A6 E1 4E 4E 00 36 95 65 08 AB	xlBãÀ;¦áNN.6.e.«
02538ED8	94 E3 75 2E AE 8E 01 AE D4 25 4B 4C 35 79 F0 07	.ãu.®..®Ô%KL5yð.
02538EE8	75 F9 66 6B 92 38 EA B7 94 E6 66 53 9B 58 1B CD	uùfk.8ê·.æfS.X.Í
02538EF8	7B 3A DE 5C 3B 62 22 BA B3 37 3A EF 18 BE 72 CC	{:Þ\;b"º³7:ï.¾rÌ

Figure 13

```
.text:0040142B lea     edi, [ebp+var_44]
.text:0040142E mov     eax, 'DCBA'
.text:00401433 stosd
.text:00401434 mov     eax, 'HGFE'
.text:00401439 stosd
.text:0040143A mov     eax, 'LKJI'
.text:0040143F stosd
.text:00401440 mov     eax, 'PONM'
.text:00401445 stosd
.text:00401446 mov     eax, 'TSRQ'
.text:0040144B stosd
.text:0040144C mov     eax, 'XWVU'
.text:00401451 stosd
.text:00401452 mov     eax, 'baZY'
.text:00401457 stosd
.text:00401458 mov     eax, 'fedc'
.text:0040145D stosd
.text:0040145E mov     eax, 'jihg'
.text:00401463 stosd
.text:00401464 mov     eax, 'nmlk'
.text:00401469 stosd
.text:0040146A mov     eax, 'rqpo'
.text:0040146F stosd
.text:00401470 mov     eax, 'vuts'
.text:00401475 stosd
.text:00401476 mov     eax, 'zyxw'
.text:0040147B stosd
.text:0040147C mov     eax, '3210'
.text:00401481 stosd
.text:00401482 mov     eax, '7654'
.text:00401487 stosd
.text:00401488 mov     eax, '/+98'
.text:0040148D stosd
.text:0040148E mov     esi, [ebp+arg_0]
.text:00401491 mov     eax, [ebp+arg_4]
.text:00401494 mov     [ebp+var_4], eax
.text:00401497 mov     edi, [ebp+arg_8]
```

Figure 14



Address	Hex	ASCII
02544190	4C 53 45 4B 41 38 32 42 38 6F 7A 31 65 48 41 6D	LSEKA82B8oz1eHAm
025441A0	4E 58 35 6F 4A 74 64 73 51 75 50 41 4F 36 62 68	NX5oJtdsQuPAO6bh
025441B0	54 6B 34 41 4E 70 56 6C 43 4B 75 55 34 33 55 75	Tk4ANpVlCKuU43Uu
025441C0	72 6F 34 42 72 74 51 6C 53 30 77 31 65 66 41 48	ro4BrtQlS0w1efAH
025441D0	64 66 6C 6D 61 35 49 34 36 72 65 55 35 6D 5A 54	dflma5I46reU5mZT
025441E0	6D 31 67 62 7A 58 73 36 33 6C 77 37 59 69 4B 36	m1gbzXs63lw7YiK6
025441F0	73 7A 63 36 37 78 69 2B 63 73 77 41 41 41 41 41	szc67xi+cswAAAAA

Figure 15

The malicious executable can use two instructions to generate 2 random 4-byte values: RDRAND and RDSEED. Firstly, it checks if these instructions are supported by the processor and then generates the random bytes. An identical implementation was also used by DarkSide ransomware, which could mean that the two groups borrowed the code from the same place:

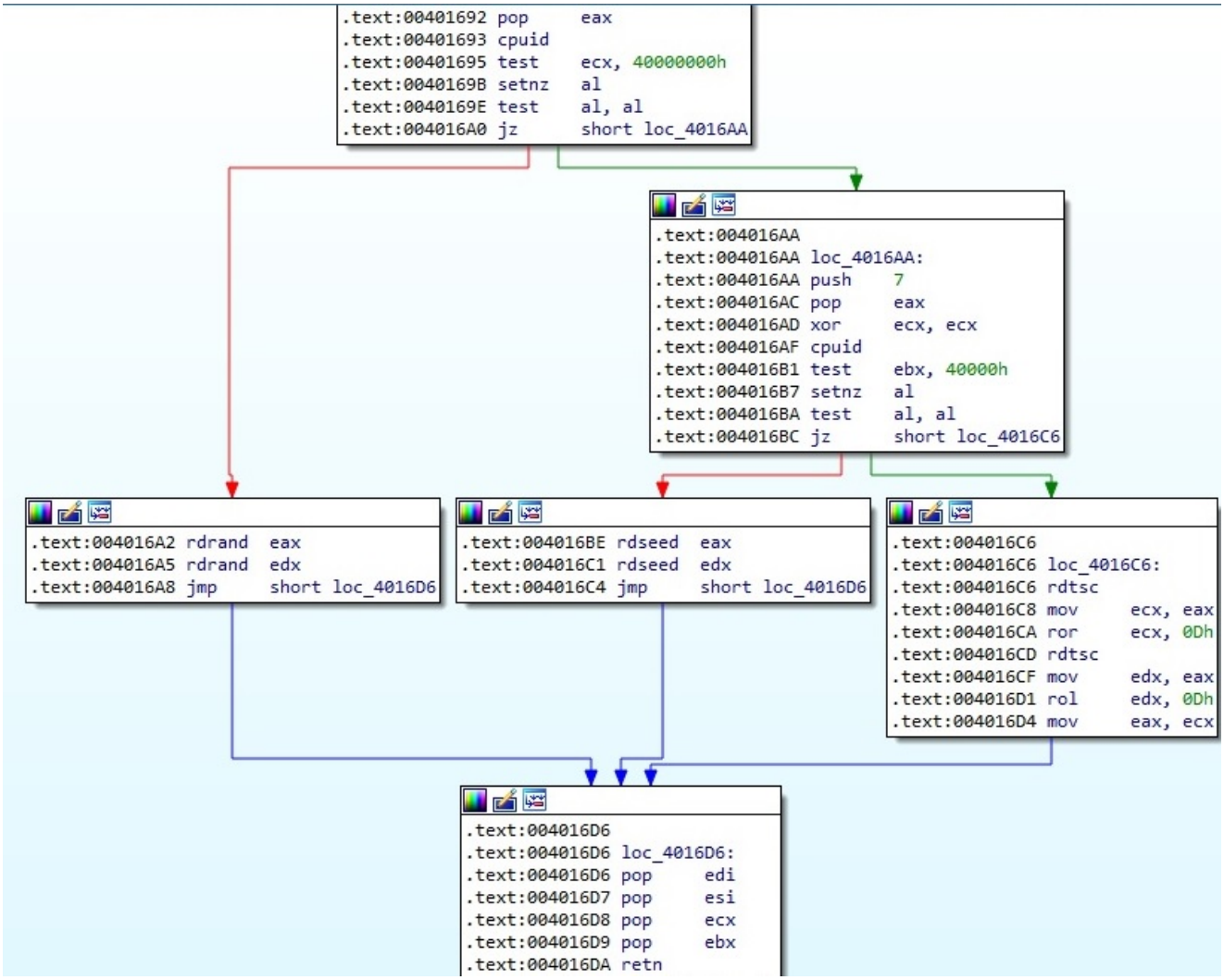


Figure 16

The random values are combined with two hard-coded values, which are modified using simple operations such as OR:

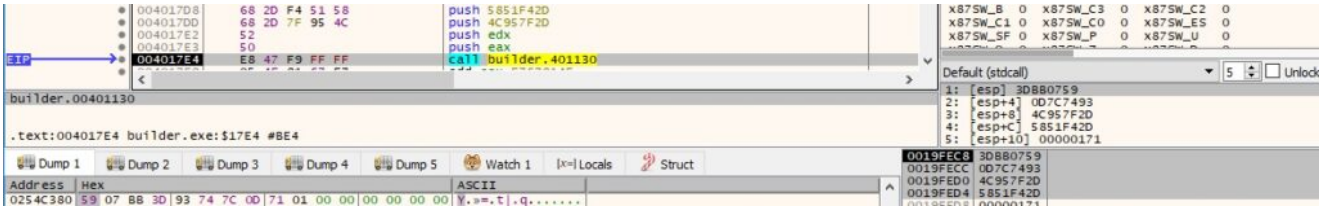


Figure 17

Figure 18

A buffer containing the RSA private key and the Base64-encoded string computed above is XOR-ed with the values generated using the 4-byte random values:

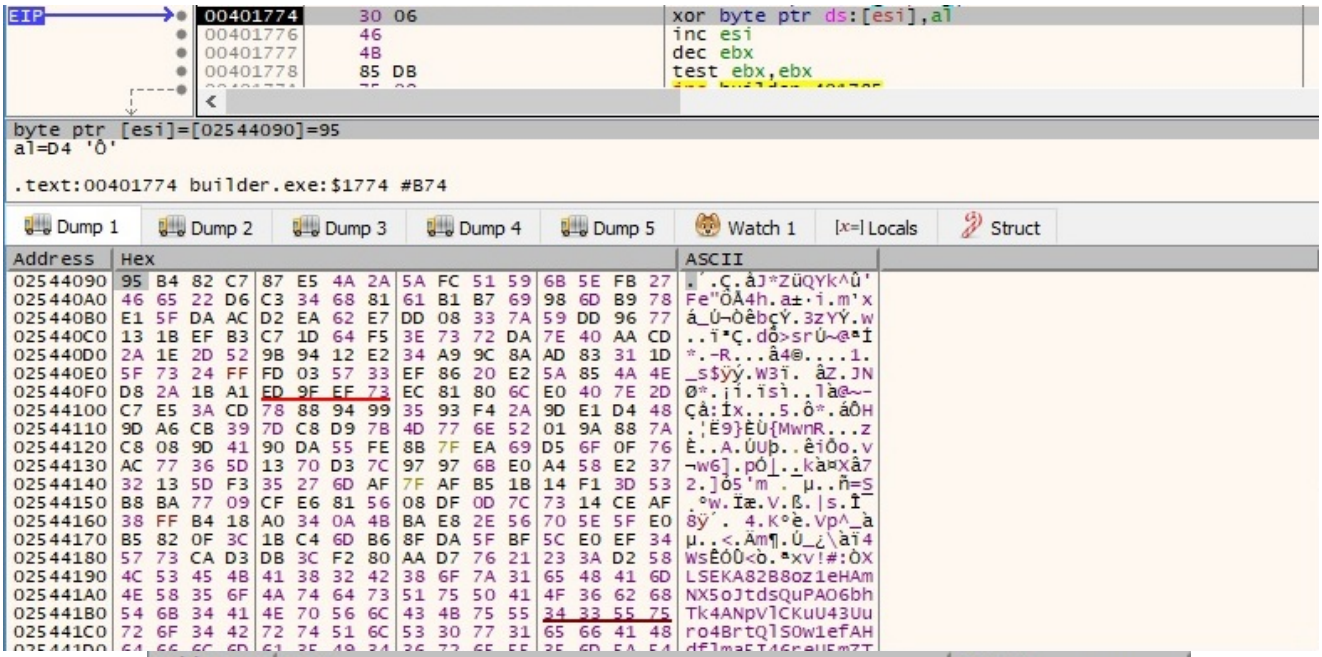


Figure 19

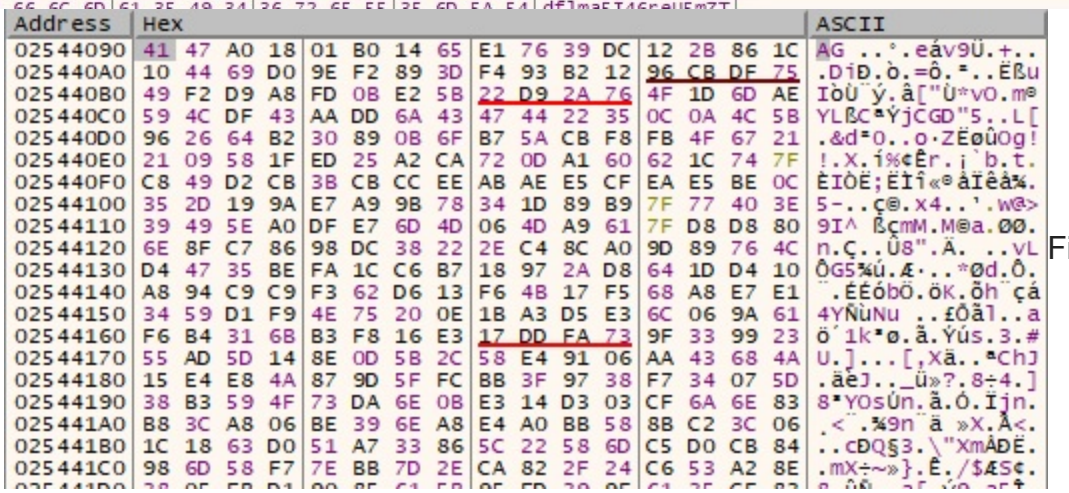


Figure 20

The encrypted data will be embedded in the final decryptor.

The malware determines the location of the resource with ID = 100 using FindResourceW (0xA = **RT_RCDATA**):

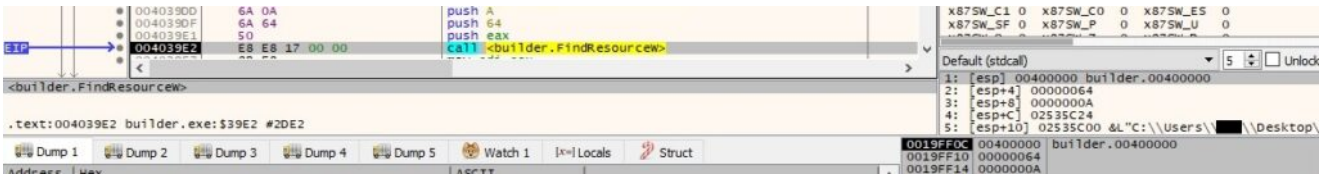


Figure 21

The resource is loaded into memory via a function call to LoadResource:

Figure 22

The builder has embedded 4 resources in the ".rsrc" section. We'll give the details about the other resources in the following paragraphs:

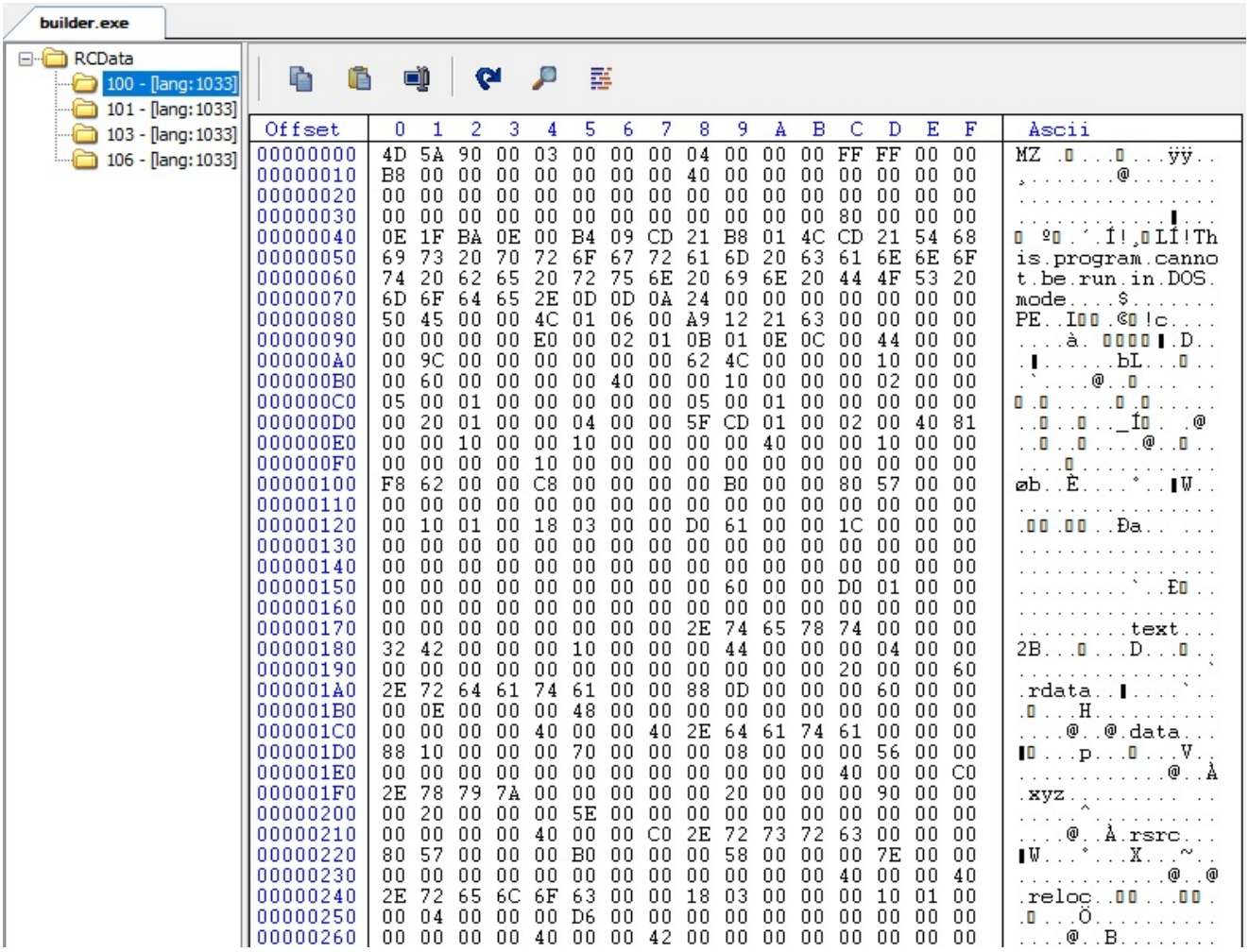


Figure 23

The binary uses the undocumented RtlImageNtHeader function to retrieve the NT header of the resource:

Figure 24

The section name called ".xyz" is replaced with ".data" by the process:

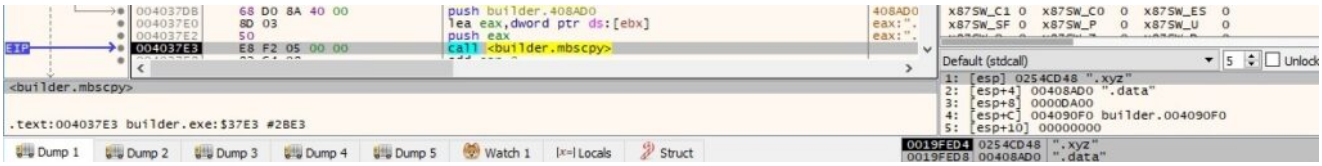


Figure 25

The CheckSumMappedFile method is used to compute the checksum of the extracted resource. The value will populate the PE checksum field in the header:

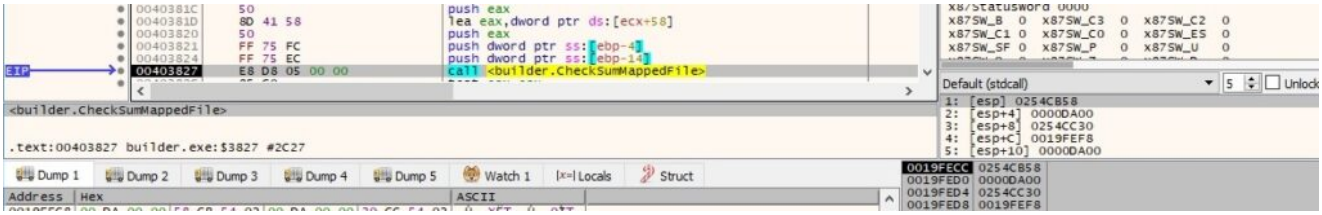


Figure 26

The builder creates the decryptor file called “LB3Decryptor.exe” using CreateFileW:

Figure 27

The process writes the modified resource to the decryptor executable via a call to WriteFile:

Figure 28

Running with the **-type enc -exe (-pass) -pubkey pub.key -config config.json -ofile LB3.exe** parameters

We only highlight the differences between this case and the first one. The builder extracts the resource with ID = 101, and the encryptor will contain the RSA public key and the ransom note content. If it’s running with the “-pass” parameter, the ransomware avoids sandboxes and increases the difficulty of the dynamic analysis. SentinelOne also analyzed the LockBit 3.0 ransomware and mentioned the “-pass” parameter.

Running with the **-type enc -dll (-pass) -pubkey pub.key -config config.json -ofile LB3_Rundll32.dll** parameters

The builder extracts the resource with ID = 103, and the encryptor will be a DLL file with multiple export functions (see figure 29).

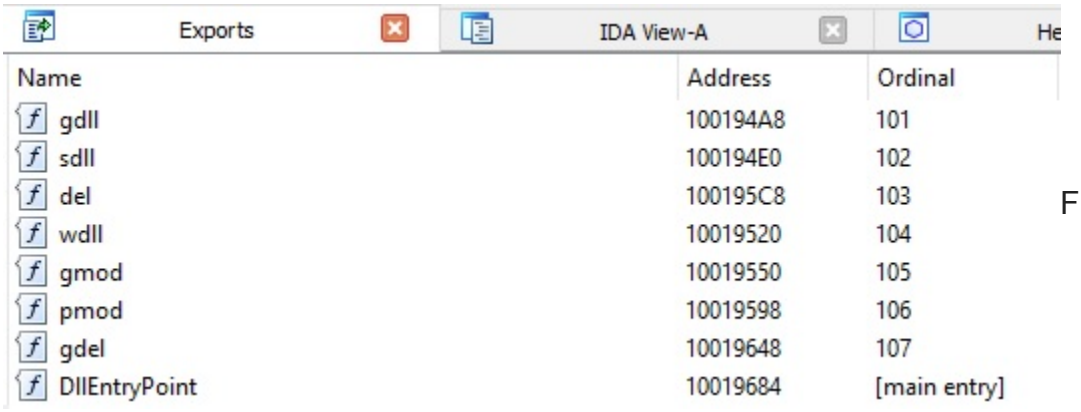


Figure 29

Running with the **-type enc -ref -pubkey pub.key -config config.json -ofile LB3_ReflectiveDll_DllMain.dll** parameters

The builder extracts the resource with ID = 106, and the encryptor will be a DLL file with a single export function. The execution flows of the two different DLLs are similar, as highlighted in the figure below.

Figure 30

According to our preliminary analysis of the LockBit 3.0 encryptor, the builder is legit and, unfortunately, can represent a gold mine for cybercriminals. Please do not use the builder for malicious purposes because you'll be persecuted according to the law.