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: <u>@GeeksCyber</u>

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:

hall har 🗋 🖬 pix hey 🗋 🚽 pub hey 🖸	
ADMALAMAAAAAAAAAAAAAAAAAAAAAAAAAAAAA	ť.
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:

0040384 0040384	4 53	push eax push ebx	ebx:L"	x875W_5F 0 x875W_P 0 x875W_U 0
		call <builder.commandlinetoargvw></builder.commandlinetoargvw>	· · · · · · · · · · · · · · · · · · ·	Default (stdcall)
<builder.commandlinetoa .text:00403B45 builder.</builder.commandlinetoa 				2: [esp+4] 0019FF54 3: [esp+8] 00000000 4: [esp+C] 00000000 5: [esp+10] 00000000
Dump 1 Dump 2	🕮 Dump 3 📲 Dump 4 👭	Dump 5 💮 Watch 1 🛛 🕅 Locals 🖉 Struct		9FF4C 02532246 L"\"C:\\Users\\\\Desktop\\bu1ld 9FF50 0019FF54

Figure 3

Running with the **-type dec -privkey privkey -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:

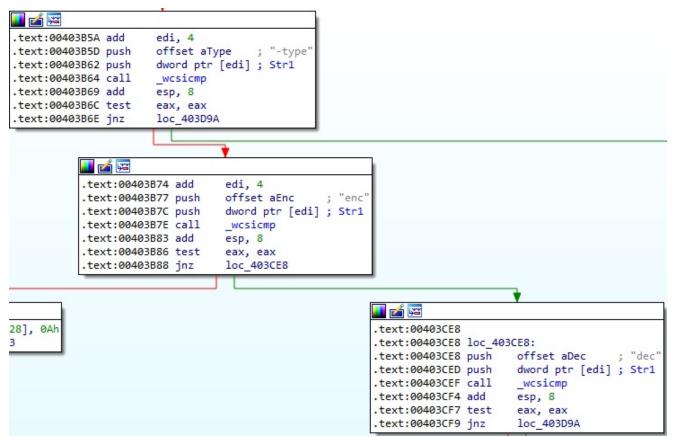


Figure 4

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):

<pre></pre>	6A 00 6A 03 6A 03 6A 03 6A 01 6B 00 00 00 80 FF 75 08 E8 1F 3C 00 00 5159F #99F	push 80 push 80 push 3 push 3 push 4000000 push dword ptr ss: ebp+3 call chulder.createfilew	[ebp+8 >	A\$71m_0 5 (Emp/y) A\$71m_r 5 (Emp/y) X875tatusWord 0000 X875w_C0 0 X875w_C2 0 X875w_C1 0 X875w_C0 0 X875w_E2 0 X875w_S0 0 X875w_P 0 X875w_U 0 Default (stdcal) ▼ 5 ↓ Unlock 1: [esp1 02535C 90 L"pr1v_key" 2: [esp+4] 80000000 3: [esp+6] 00000001 4: [esp-C] 000000001 4: [esp-C] 000000003
Dump 1 0 Dump 2 0 0	Dump 3 🔛 Dump 4 🔛 0	ump 5 👹 Watch 1 🕼 I Locals 🖉 Struct		ECC 02535C90 L"priv.key"
Address Hex 0019FEC4 20 34 C3 75 A0 5A 0019FED4 01 00 00 00 00 00		ASCII 00 80 4ŵ 25\5	00199 00199 00199 00199	FED 80000001 FED 8000001 FED 8000000 FED 8000003 FEE 00000003 FEE 40000000

Figure 5

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

<pre>>004015D1 8D 45 F4 004015D4 50 004015D5 50 004015D5 53 004015D5 F7 55 F8 004015D0 FF 75 F8 004015D0 FF 75 F8 004015D0 FF 75 F8 </pre>	push 0 lea eax,dword ptr ss:[ebp-C] push eax push dword ptr ss:[ebp-6] push dword ptr ss:[ebp-6] call dword ptr ss:[ebp-4] call chuilder.ReadFilex	x875xtausword 0000 x875w_B 0 x875w_C3 0 x875w_C2 0 x875w_C1 0 x875w_C0 0 x875w_E5 0 x875w_5 0 x875w_P 0 x875w_U 0 y875w_5 0 x875w_P 0 x875w_U 0 x875w_5 0 x875w_V 0 x875w_U 0 x875w_5 0 x875w_C 0 x875w_C 0 x875w_U 0 x875w_5 0 x875w_C 0 x
.text:004015DC builder.exe:\$15DC #9DC		5: [esp+10] 00000000
.text:004015DC builder.exe:\$15DC #9DC	un f Martin Indiana Securi	5: [esp+10] 00000000 0019FED4 000001CC
🕮 Dump 1 👹 Dump 2 👹 Dump 3 👹 Dump 4 👹 D	ump 5 👹 Watch 1 🕼 Locals 🎾 Struct	0019FED4 000001CC 0019FED8 02541D28
	ump 5 👹 Watch 1 💷 Locals 🌮 Struct	0019FED4 000001CC

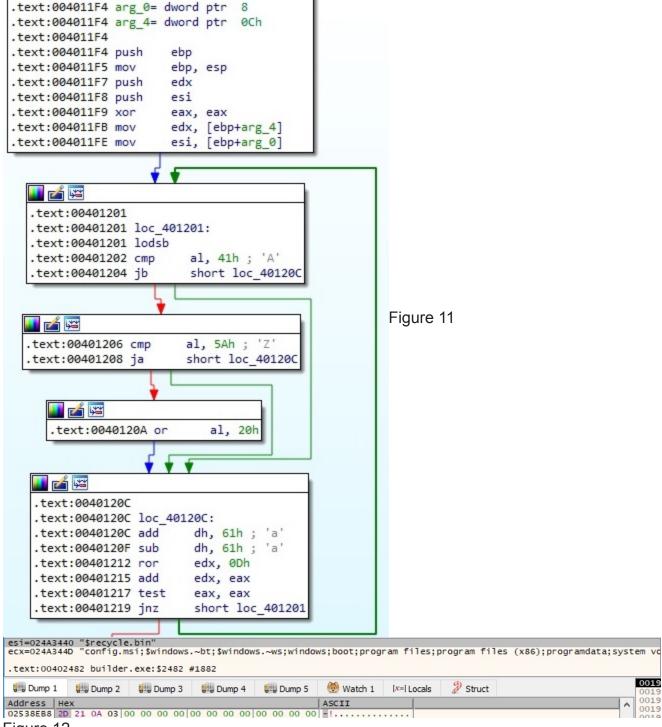
The RSA private key is Base64-decoded by the malicious process:

L			
	/		
.text:004013C3	262.		
.text:004013C3 loc_401			
.text:004013C3 mov .text:004013C5 movzx	ecx, [esi] 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]	<u> </u>	
.text:004013EA pop	edi	Figure 7	
.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		
	-		
Address Hex		100	ASCII
02541EA0 95 B4 82 C7 87			
02541EB0 46 65 22 D6 C3 02541EC0 E1 5F DA AC D2			
02541ED0 13 18 EF B3 C7	1D 64 F5 3E 73 72 D.	A 7E 40 AA CD	ï⁼C.dő>srú~@≏Í
02541EE0 2A 1E 2D 52 9B 02541EF0 5F 73 24 FF FD		A AD 83 31 1D	*Râ4©1. _s\$ÿý.W3ï. âZ.JN
02541F00 D8 2A 1B A1 ED	9F EF 73 EC 81 80 6	C E0 40 7E 2D	Ø*. 11. 151 la@~-
02541F10 C7 E5 3A CD 78	88 94 99 35 93 F4 2	A 9D E1 D4 48	Cå: 1x5. ô*. áôh FIQUre 8
02541F20 9D A6 CB 39 7D 02541F30 C8 08 9D 41 90	C8 D9 /B 4D // 6E 5 DA 55 FE 8B 7F EA 6	2 01 9A 88 7A 9 D5 6F 0F 76	.¦Ë9}ÈÙ{MwnRz ÈA.ÚUþêiÕo.v
02541F40 AC 77 36 5D 13	70 D3 7C 97 97 6B E	0 A4 58 E2 37	¬w6].póL.ka¤xâ7
02541F50 32 13 5D F3 35 02541F60 B8 BA 77 09 CF	27 6D AF 7F AF B5 1	B 14 F1 3D 53	2.]ó5'm ⁻ . µñ=S .°w.ïæ.∨.ß. s.î
02541F60 88 BA 77 09 CF 02541F70 38 FF B4 18 A0	34 0A 4B BA E8 2E 5	6 70 5E 5F EO	8ÿ′. 4.K°è.Vp^_à
02541F80 B5 82 OF 3C 1B	C4 6D B6 8F DA 5F B	F 5C E0 EF 34	μ. <. Äm¶. Ú_¿\àï4
02541F90 57 73 CA D3 DB	3C F2 80 AA D7 76 2	1 23 3A D2 58	WSEOU<0. XV!#:OX

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:

•	0040158C 61 00401591 66 00401593 66 00401595 66 00401597 61 0040159C F1 0040159F E5 <	A 00 3 80 00 00 0 A 03 A 00 A 01 5 00 00 00 8 75 08 8 1F 3C 00 0	0		oooooo ord ptr ss uilder.Cre		-	_		[e	op+8 >	x875tatusWord 0000 x875W_CB 0 x875W_CB 0 x875W_C2 0 x875W_CL 0 x875W_CC3 0 x875W_C5 0 x875W_C5 0 x875W_C9 0 x875W_U 0 Default (stdcall)
<builder.create< td=""><td>Filew> builder.exe:\$159</td><td>F #99F</td><td>_</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>2: [esp+4] 8000000 3: [esp+8] 0000001 4: [esp+C] 00000000 5: [esp+10] 00000003</td></builder.create<>	Filew> builder.exe:\$159	F #99F	_									2: [esp+4] 8000000 3: [esp+8] 0000001 4: [esp+C] 00000000 5: [esp+10] 00000003
Ump 1	Dump 2	3 💷 Dump	4 📖 Dum	p 5 🛞 Wal	tch 1 [x=]	Locals 🐉	Struct					9FE85 02535CB2 L"config.json" 9FE8C 80000000
02541EB0 46 65	82 C7 87 E5 4A 2 22 D6 C3 34 68 8 DA AC D2 EA 62 E	1 61 B1 B7	59 98 6D B	9 78 Fe"OA4	J∘ZüQYk∧ů` h.a±.i.m'x bçÝ.3ZYÝ.w						001 001 001 001	9FE90 00000001 9FE94 00000000 9FE98 00000000 9FE95 00000000 9FE96 00000000
	Address	Hex									1	ASCII
Figure 9	02541FC0 02541FE0 02541FE0 02542000 02542000 02542010 02542020 02542030 02542040 02542040 02542060 02542060 02542080 02542080 02542080 02542080	20 20 30 30 30 30 20 20 30 30 30 30 7D 2C 0A 20 20 7B 70 74 2C 0D 74 5F 73 65 72 73 2C 0D	0A 20 20 22 30 30 30 30 20 22 30 30 30 30 0D 0A 20 20 0D 0A 20 20 0D 0A 20 20 0D 0A 20 20 0D 0A 20 20 0D 0A 20 20 0D 0A 20 20 20 0D 0A 20 20 20 0D 0A 20 20 20 20 20 30 30 40 40 40 40 40 40 40 40 40 40 40 40 40 4	75 69 30 30 30 30 68 65 30 30 30 30 22 63 20 22 20 20 6F 64 20 20 6C 65 0A 20 61 74	64 2: 30 30 30 30 79 2: 30 30 30 30 6F 60 73 65 20 20 65 2: 20 20 6E 65 20 20 69 65	2 3A 20 30 30 30 2 3A 20 30 30 30 2 3A 20 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 2 3A 20 2 2 2 2 3A 20 2 2 2 2 3A 20 2 0 20 2 2 2 3A 20 2 0 0 20 2 0 0 0 0	0 22 0 30 0 22 0 30 0 22 0 30 0 30 9 67 4 69 0 22 0 22 2 65 5 22 0 20 2 3A 2 73	20 30 22 30 22 22 65 61 62 20 65 61 22 20 67 20 67	7B 00 30 3 32C 00 30 3 30D 3 30D 3 30D 3 30D 3 30D 3 4 67 7 62 6 75 7 63 7 62 6 69 6 74 7 69 7 69 7 73 2	0 30 0 30 0 0A 0 30 0 30 0 30 0 30 0 30	20 30 20 20 30 20 30 20 30 20 20 30 20 20 20 20 20 30 20 20 20 20 20 20 20 20 20 20 20 20 20	<pre>I "bot": { "uid": "00000 000000000000000000000000000000</pre>
	025420D0	66 61	6C 73	65 2C	0D 0/	20 20	0 20	20	20 2	0 22	6C	false, "1
	025420E0 025420F0		67 75 6C 73			20 20	0 20	63 20	6B 2 20 2	0 22	20 6C	anguage_check": false, "1 ocal disks": tru

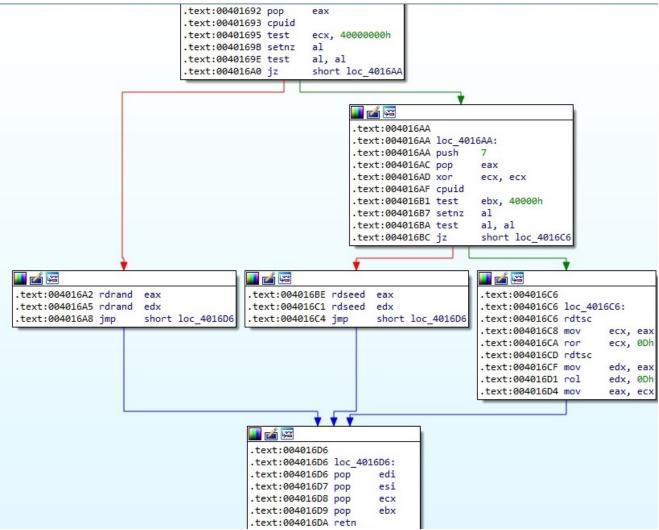
The malware implements a custom "hashing" function that computes a 4-byte value for each whitelisted directory/file/extension/host. An example of a function result is shown in figure 12.



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 -!1.o.oxp&5~h& 02538EC8 D7 6C 42 E3 C0 3B A6 E1 4E 4E 00 36 95 65 08 AB xlBàÀ; 'áNN.6.e.« 02538ED8 94 E3 75 2E AE 8E 01 AE D4 25 4B 4C 35 79 F0 07 .ãu.eeô%KL5yð. 02538EE8 75 F9 66 6B 92 38 EA 87 94 E6 66 53 9B 58 1B CD uùfk.8ê.æf5.X.1 02538EF8 7B 3A DE 5C 3B 62 22 BA B3 37 3A EF 18 BE 72 CC {:p\;b"e*7:ï.¾r1 .text:0040142B lea edi, [ebp+var_44] .text:0040142E mov eax, 'DCBA'
02538EC8 D7 6C 42 E3 C0 3B A6 E1 4E 4E 00 36 95 65 08 AB x1BâA; 'áNN.6.e.« Figure 13 02538ED8 94 E3 75 2E AE 8E 01 AE D4 25 4B 4C 35 79 F0 07 .ãu.ee0%KL5y0. 02538EE8 75 F9 66 6B 92 38 EA B7 94 E6 66 53 9B 58 1B CD uufk.8ê.æf5.X.1 02538EF8 7B 3A DE 5C 3B 62 22 BA B3 37 3A EF 18 BE 72 CC {:p\;b"e*7:i.%r1 .text:0040142B lea edi, [ebp+var_44] .text:0040142E mov eax, 'DCBA'
02538ED8 94 E3 75 2E AE 8E 01 AE D4 25 4B 4C 35 79 F0 07 .au.ee0%KL5y0. 02538EE8 75 F9 66 6B 92 38 EA B7 94 E6 66 53 9B 58 1B CD uùfk.8ê.æfs.X.1 02538EF8 7B 3A DE 5C 3B 62 22 BA B3 37 3A EF 18 BE 72 CC {:p\;b"°*7:ï.%r1 .text:0040142B lea edi, [ebp+var_44] .text:0040142E mov eax, 'DCBA'
02538EF8 7B 3A DE 5C 3B 62 22 BA B3 37 3A EF 18 BE 72 CC {:Þ\;b"°*7:ï.%rì .text:0040142B lea edi, [ebp+var_44] .text:0040142E mov eax, 'DCBA'
.text:0040142B lea edi, [ebp+var_44] .text:0040142E mov eax, 'DCBA'
.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 Figure 14
.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]
Address Hex ASCII
02544190 4C 53 45 48 41 38 32 42 38 6F 7A 31 65 48 41 6D LSEKA82B8oz1eHAm
025441A0 4F 58 35 6F 44 74 64 73 51 75 50 41 4F 36 62 68 NX501tds0uPA06bh
025441B0 54 6B 34 41 4E 70 56 6C 43 4B 75 55 <u>34 33 55 75</u> Tk4ANpVlCkuU43Uu Figure 15 025441C0 72 6F 34 42 72 74 51 6C 53 30 77 31 65 66 41 48 ro4BrtQlSow1efAH
025441D0 64 66 6C 6D 61 35 49 34 36 72 65 55 35 6D 5A 54 df1ma5I46reU5mZT
025441E0 6D 31 67 62 7A 58 73 36 33 6C 77 37 59 69 4B 36 m1gbzXs631w7YiK6
025441F0 73 7A 63 36 37 78 69 28 63 73 77 41 41 41 41 41 41 szc67xi+cswAAAAA The malicious executable can use two instructions to generate 2 random 4-byte value

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 <u>DarkSide ransomware</u>, which could mean that the two groups borrowed the code from the same place:



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



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:

				-										1					
EIP		00401			0 06						c es		Tr	ds:[esi	, a 1			
		00401			в						c eb								
		00401		8	5 DB					tes	ste	bx,e	bx						
	[•	<		-						-			-	~ ~ ~ ~	-				
byte ptr [esil=	[02544	0901=	95						_									
al=D4 '0'																			
.text:0040	1774	builder	exe	\$17	74 #R	74													
Teckeroon		barrac	Texe		11	-		-		1.0							10		
Dump 1	00	Dump 2	00	Dump	3	🚽 Dur	mp 4	D	ump 5		👽 W	atch	1	[<i>x</i> =]	Local	S	2 Struct		
	Hex									_	SCII	-							
							1 59							'k^û'					
			C3 34			B1 B		98 GD						.m'x					
025440B0 8 025440C0 1				64				9 DD 7E 40						YÝ.w					
025440D0			9B 94						31 1			â4							
025440E0			FD 03			86 2			4A 4					Z.JN	4				
025440F0 E			ED 9F		73 EC			0 40						ac					
02544100 0	C7 E5	3A CD	78 88	94	99 35	93 F	4 2A	D E1	D4 4					. áÔH					
02544110					7B 4D			01 9A	88 7	Α.				Z					
02544120 0			90 DA				A 69 [OF 7					00.V					
02544130			13 70				BEO							AXâ7					
02544140			35 27		AF 7F			L4 F1						. n=S	2				
02544150 8		77 09	A0 34	81		DF 0		73 14 70 5E				æ. V.		p^_a					
02544170			1B C4				FBF							\aï4					
02544180		CA D3				D7 7			D2 5					#:0X					
			41 38						41 6					eHAm					
025441A0 4	4E 58	35 GF	4A 74	64	73 51	75 5	0 41	4F 36	62 6	8 N	X50J	tdsQ	UPA	O6bh	1				
025441B0 9									55 7					143Uu					
02544100			72 74	51		30 7	7 31	55 66	41 4	8 r				efAH					
		dress	He		271212							- 1712		2 00 7 1		1	ASCII		
	-	544090	-	-	A0 1	8 01	B0 1	4 65	E1	76	39	DC	12	2B	86	10	And in case of the local division of the loc	eáv9Ü.+	
		5440A0			69 D		F2 8			93				CB				=0. = EBu	
																		["Ù*v0.mº	
		5440B0			D9 A	_		2 58		D9			4F	_		AE			
		5440C0			DF 4	_		A 43		44	22		OC			5B		CGD"5L[
		5440D0			64 B	_	89 0			5A			FB			21		o.ZEøüOg!	
		5440E0		09			25 A						62			7F		Er. j b.t.	
		5440F0			D2 C		CB C	_			E5		EA	E5		OC		ï«®åIêå¾.	
	02	544100			19 9		A9 9				89		7F			3E		x4'.W@>	
		544110	39	49	SE A	0 DF	E7 6	D 4D	06	4D	A9	61	7F	D8		80	91^ Bcm	M.M@a.ØØ.	L
igure 19	9 02!	544120) 6E	8F	C7 8	6 98	DC 3	8 22	2E	C4	8C	AO	9D	89	76	4C	n.ÇÛ8	".ÄVL	Figure
	025	544130	D4	47	35 B	EFA	10 0	6 B7	18	97	2A	DS	64	1D	D4	10	ÔG5%ú. Æ	*Ød.Ô.	
		544140			C9 C		62 D				17				E7			. öK. õh çá	
		544150		59				0 08			D5	_	6C		9A			£0ã1a	
		544160		B4			F8 1			DD			9F		99			ā.Ýús.3.#	
		544170			5D 1		OD 5			E4			AA			4A		,xäªChJ	
		544180		E4			9D 5			3F	97		F7			50		ü»?.8÷4.]	
		544190		B3			DA				D3	_	CF			83		.ã.Ó.Ïin.	
		5441A0			A8 0			E AS		AO						06	< .%9n		
		5441B0			63 D			3 86		22	58			DO		84	CDQ§3	.\"XmADE.	
		5441C0			58 F	7 7E	BB 7			82			60			8E	. mx+~»}	.E./\$AS¢.	
	0.00	44107		ALC: N		1 100	ar /				213		1.14	11		12.2		AND SET	

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**):

0040390D 6A 0A 0040390F 6A 64 004039E1 50	push A push 64 push eax	x875w_C1 0 x875w_C0 0 x875w_E5 0 x875w_SF 0 x875w_P 0 x875w_U 0
SIP 00403922 E8 E8 17 00 00 < c south of the sourcews	call 	> Default (stdcall) 5 Unlock 1: [esp] 00400000 bu11der.00400000
<pre>.text:004039E2 builder.exe:\$39E2 #2DE2</pre>		2: [csp+4] 0000064 3: [csp+6] 000000A 4: [csp+C] 02535C24 5: [csp+10] 02535C00 &L"C:\\Users\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
Ump 1 Ump 2 Ump 3 Ump 4	🕮 Dump 5 🛛 👹 Watch 1 🛛 🕸 🖉 Struct	0019FF0C 00400000 builder.00400000 0019FF10 00000064 0019FF14 0000000A

Figure 21

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



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

builder.exe											
RCData 100 - [lang: 1033] 101 - [lang: 1033]	6		6	P :	¥						
103 - [lang: 1033]	Offset	0 1	23	4 5	6	78	9 A	B C	D	E F	Ascii
	00000000 0000010 0000030 0000040 0000050 0000050 00000050 00000080 00000080 00000080 00000080 000000	$\begin{array}{cccccccc} 4D & 5A & 9 \\ B8 & 00 & 0 \\ 00 & 00 & 0 \\ 00 & 00 & 0 \\ 00 & 01 & 0 \\ 00 & 01 & 0 \\ 69 & 73 & 2 \\ 74 & 20 & 6 \\ 6D & 6F & 6 \\ 50 & 45 & 0 \\ 00 & 00 & 0 \\ 2E & 72 & 6 \\ 00 & 00 & 0 \\ 00 & 00 & 0 \\ 2E & 72 & 6 \\ 00 & 04 & 0 \\ 00 & 00 & 0 \\ 00 & 00 &$	$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	$ar{0}0$ $ar{0}0$ $ar{0}0$ $ar{0}0$ $ar{0}00$ $B4$ $C20$ $C22$ $C20$ $C22$ $C22$ $C22$ $C20$ $C22$ $C20$ $C20$ $C20$ $C20$ $C20$ $C20$ $C20$ $C20$ $C00$ $C0$	$\begin{array}{c} 00 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ $	$ \begin{array}{c} 0 & 40 \\ 0 & 00 \\ 0 & 00 \\ 0 & 00 \\ 0 & 00 \\ 0 & 21 \\ 0 & 24 \\ 0 & 00 \\ 0 & 0 \\ 0 & 0$	$\begin{array}{c} \textbf{B8} & 0.1\\ \textbf{6D} & 20\\ \textbf{69} & \textbf{6E}\\ 000 & 021\\ 012 & 012 & 012\\ 010 & 000 & 012\\ 000 & 010 & 000\\ 000 & 000 & 000\\$	$\begin{array}{cccccccccccccccccccccccccccccccccccc$	$\begin{smallmatrix} 00 & 0 \\ 00 & 0 \\ 00 & 0 \\ 21 & 60 \\ 4F & 5 \\ 00 & 0 \\ 0 & 0$	$egin{array}{cccccccccccccccccccccccccccccccccccc$	MZ

Figure 23

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

O0403782 FF 75 EC O0403783 E8 99 1A 00 00	push dword ptr ssilebp-141 call <builder.rtlimagentheader></builder.rtlimagentheader>	~	Default (stdcall) 🔻 5 🜩 🗌 Unlod
 <builder.rtlimagentheader> .text:00403785 builder.exe:\$3785 #2885</builder.rtlimagentheader>			1: [esp] 0254C858 2: [esp+4] 0000A00 3: [esp+4] 004090F0 bu11der.004090F0 4: [esp+C] 00000000 5: [esp+10] 00000000
Image: Weight of the second	Dump 5 👹 Watch 1 [x= Locals 🦻 Struct ASCII F 00 00 MZ	0015 0015 0015	DFED3 0254C858 DFED0 00000A00 DFEE0 004090F0 DFEE4 00000000 DFEE4 00000000 DFEE4 00000000

Figure 24

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

004037DB 68 D0 8A 40 00 004037E0 8D 03 004037E2 50	<pre>push builder.408AD0 lea eax,dword ptr ds:[ebx] push eax</pre>	408AD0 eax1", x875w_C1 0 x875w_C0 0 x875w_E5 0 eax1", x875w_SF 0 x875w_P 0 x875w_U 0 eax1", x875w_SF 0 x875w_P 0 x875w_U 0
610 002103768 E8 F2 05 00 00 	call call score based of the score scor	> Default (stdcall)
.text:004037E3 builder.exe:\$37E3 #2BE3		3: [esp+8] 0000DA00 4: [esp+c] 004090F0 builder.004090F0 5: [esp+10] 0000000
Dump 1 Dump 2 Dump 3 Dump 4 Dump 4	mp 5 👹 Watch 1 🛛 🛛 Ix=l Locals 🖉 Struct	0019FED1 0254C048 ".xyz" 0019FED8 00408AD0 ".data"

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

Address Hex	ASCI	I			0254CC30 0FED8 0019FEF8		
Ump 1 Ump 2 Ump 3	🕮 Dump 4 🛛 🗰 Dump 5 👹	Watch 1 x= Locals	Struct	0019	FECC 0254CB58 0FED0 0000DA00		
 kuilder.CheckSumMappedFile> .text:00403827 builder.exe:\$3827 #	2C27				2: [esp+4] 0000DA00 3: [esp+8] 0254CC30 4: [esp+C] 0019FEF8 5: [esp+10] 0000DA00		
• <				>	Default (stdcall) 1: [esp] 0254CB58	▼ 5 🜩 🗌 Unloc	
0040381C 50 0040381D 80 41 00403820 50 00403821 FF 75 00403824 FF 75 00403824 EF 75	.58 lea e push FC push EC push	push eax tea eax, dword ptr ds:[ecx+58] push eax push dword ptr ss:[ebp-4] push dword ptr ss:[ebp-14] call dbullder.CheckSumMappedFile>			X#/>tatusword 0000 X875W_B0 X875W_C2 0 X875W_C2 0 X875W_C1 0 X875W_C0 0 X875W_E5 0 X875W_SF 0 X875W_P 0 X875W_U 0		

Figure 26

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

 004016F2 68 80 00 00 00 004016F7 6A 02 004016F9 6A 00 004016FB 6A 00 004016FD 68 00 00 40 004016FD 68 00 00 40 00401702 FF 75 08 	push 0 push 80 push 2 push 0 push 0 push 40000000 push 40000000 push 40000000 push 40000000 [elbp+5] [elbp+5] [elbp+5] [elbp+5]	x875tatusWord 00 x875w_B 0 x873 x875w_C1 0 x874 x875w_SF 0 x874	000 SW_C3 0 x87SW_C2 0 SW_C0 0 x87SW_ES 0 SW_P 0 x87SW_U 0
<pre> c c c c c c c c c c c c c c c c c c c</pre>		> Default (stdcall) 1: [esp] 02535CD 2: [esp+4] 40000 3: [esp+8] 00000 4: [esp+C] 00000 5: [esp+10] 0000	000
Image: Second	🛞 Watch 1 💷 Locals 🖉 Struct	0019FECC 02535CD8 L" 0019FED0 40000000 0019FED4 00000000	
Address Hex 0254CC30 ED 43 01 00 02 00 40 81 00 00 10 00 00 10 00 00 0254CC40 00 00 40 00 00 10 00 00 00 00 00 10 00 00	fc@	0019FED8 00000000 0019FEDC 00000002 0019FEE0 00000080	

Figure 27

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

00401713 6A 00 00401715 6D 45 F8 00401715 5D 00401718 5D 00401719 FF 75 10 00401710 FF 75 DC 0040171C FF 75 DC 0040171F FF 75 FC 00400172F E8 DE 3A 00 00 0040172F E8 DE 3A 00 00	push 0 lea eax, dword ptr ss:[ebp-6] push eax push dword ptr ss:[ebp+10] push dword ptr ss:[ebp-4] call dwilder.writeFiles) v	x875tatusword 0000 x875w_B 0 x875w_C3 0 x875w_C2 0 x875w_C1 0 x875w_C0 0 x875w_E5 0 x875w_SF 0 x875w_P 0 x875w_U 0 Default (stdcall)
<pre><builder.writefile> .text:00401722 builder.exe:\$1722 #822</builder.writefile></pre>			1: [esp] 000001D0 2: [esp+4] 0254CB58 3: [esp+6] 0000DA00 4: [esp+C] 0019FEFC 5: [esp+10] 00000000
Address Hex 0254C856 400 5A 90 00 00 00 04 00 00 07		0011 0011 0011 0011 0011 0011 0011 001	DIED: 00000100 PEED: 0124C85 PED: 0124C85 PED: 0124C85 PED: 0124C85 PEED: 0129FEFC PEED: 0129FEFC PEED: 0129FEFC PEED: 00200000 PEEE: 010000000 PEEE: 01000000 PEEE: 01000000 PEEE: 01000000 PEEE: 01000000 PEE: 0100000 PEE: 01000000 P

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. <u>SentinelOne</u> 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).

P	Exports	×	E	IDA View-A	×	Ō	He
Name				Address		Ordinal	
f gdll				100194A8	3	101	
f sdll				100194E0		102	
f del			100195C8	3	103	Figure 29	
f wdll			10019520		104	<u>-</u>	
f gmod				10019550		105	
f pmod				10019598		106	
f gdel				10019648		107	
f DIIEntry	Point			10019684		[main entry]	

Running with the **-type enc -ref -pubkey pub.key -config config.json -ofile** LB3_ReflectiveDII_DIIMain.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.



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.