A Detailed Analysis of The LockBit Ransomware

Prepared by: Vlad Pasca, LIFARS, LLC

Date: 02/14/2022



www.LIFARS.com

info@lifars.com

©2022 SecurityScorecard Inc. 244 Fifth Avenue, Suite 2035, New York, NY 10001 1.212.222.7061

Table of Contents

Executive Summary	2
Analysis and Findings	2
Thread activity – sub_4DF310 function	10
Thread activity – sub_4C3430 function	14
Thread activity – sub_4A2EC0 function	19
Thread activity – sub_45C960 function	
Thread activity – sub_497060 function	34
Thread activity – sub_49E730 function	
Printing ransom notes	44
LockBit Wallpaper Setup	46
Extract and save the HTA ransom note to Desktop	52
Indicators of Compromise	59
Registry Keys	59
Files Created	59
Processes spawned	59
Mutex	60
LockBit 2.0 Extension	60
LockBit 2.0 Ransom Note	60
Appendix	61
List of processes to be killed	61
List of services to be stopped	61



Executive Summary

LockBit 2.0 ransomware is one of the most active families in the wild and pretends to implement the fastest encryption algorithms using multithreading with I/O completion ports. The malware doesn't encrypt systems from CIS countries and can perform UAC bypass on older Windows versions if running with insufficient privileges. A hidden window that logs different actions performed by LockBit is created and might be activated using the Shift+F1 shortcut. The ransomware mounts all hidden volumes and stops a list of targeted processes and services. The malware generates a pair of ECC (Curve25519) session keys, with the private key being encrypted using a hard-coded ECC public key and stored in the registry. The binary deletes all Volume Shadow Copies using vssadmin and clears the Windows security application and system logs. LockBit obtains a list of physical printers used to print multiple ransom notes. The encrypted files have the ".lockbit" extension, and only the first 4KB of the file will be encrypted using the AES algorithm. A unique AES key is generated for each file, encrypted using the session ECC public key, and stored in each encrypted file.



Analysis and Findings

SHA256: 9feed0c7fa8c1d32390e1c168051267df61f11b048ec62aa5b8e66f60e8083af

The malware verifies whether it's being debugged by checking the NtGlobalFlag field from the PEB (process environment block). If the debugger is detected, the process jumps to an infinite loop:





The encrypted strings are stored as stack strings and will be decrypted using the XOR operator. An example of a decryption algorithm is shown in figure 2, along with the decrypted DLL name:

EIP	0048FFB2	C7	84	24	E8	00	00	00	20	00	mov dword ptr ss:[esp+E8],20	20: '
	004BFFBD	33	F6								xor esi,esi	esi:En
	0048FF8F	CG	84	24	EC	00	00	00	47		mov byte ptr ss: esp+EC1,47	47: 'G'
	0048FFC7	CG	84	24	ED	00	00	00	44		mov byte ptr ss: esp+ED1,44	44: 'D'
	004BFFCF	CG	84	24	EE	00	00	00	49		mov byte ptr ss: esp+EE1,49	49: 'I'
	0048FFD7	CG	84	24	EF	00	00	00	50		mov byte ptr ss: esp+EF1.50	50: 'P'
	004BFFDF	CG	84	24	FO	00	00	00	4C		mov byte ptr ss: [esp+F0],4C	4C:'L'
	0048FFE7	CG	84	24	F1	00	00	00	55		mov byte ptr ss: [esp+F1],55	55: 'U'
	004BFFEF	CG	84	24	F2	00	00	00	53		mov byte ptr ss:[esp+F2],53	53: 'S'
	0048FFF7	CG	84	24	F3	00	00	00	0E		mov byte ptr ss: esp+F3],E	
	004BFFFF	CG	84	24	F4	00	00	00	44		mov byte ptr ss: esp+F4],44	44: 'D'
	004C0007	CG	84	24	F5	00	00	00	4C		mov byte ptr ss:[esp+F5],4C	4C: 'L'
	004C000F	CG	84	24	F6	00	00	00	4C		mov byte ptr ss:[esp+F6],4C	4C: 'L'
	• 004C0017	8A	84	24	EC	00	00	00			mov al, byte ptr ss:[esp+EC]	
	004C001E	CG	84	24	F7	00	00	00	00		mov byte ptr ss:[esp+F7],0	
	004C0026	66	66	OF	1F	84	00	00	00	00	nop word ptr ds:[eax+eax],ax	
r	-> 004C0030	SA	94	34	EC	00	00	00			mov dl,byte ptr ss:[esp+esi+EC]	
	004C0037	8B	84	24	E8	00	00	00			mov eax,dword ptr ss:[esp+E8]	1-2200-00
	004C003E	OF	BE	CS							movsx ecx,al	ecx:En
1	004C0041	OF	BE	C2							movsx eax,dl	
1	004C0044	33	C8								xor ecx,eax	ecx:En
	004C0046	88	8C	34	EC	00	00	00			mov byte ptr ss:[esp+esi+EC],cl	1000 C
	004C004D	46									inc esi	esi:En
1	004C004E	83	FE	OB							cmp esi,B	esi:En
1	0 004C0051	^ 72	DD								jb lockbit.4C0030	
	004C0053	A1	10	08	4F	00					mov eax, dword ptr ds: [4F081C]	
	004C0058	CG	84	24	F7	00	00	00	00		mov byte ptr ss:[esp+F7],0	
	004C0060	85	CO								test eax,eax	
·	• 004C0062	× 75	0A								jne lockbit.4C006E	
	004C0064	E8	A7	19	F5	FF			_		call lockbit.411A10	
Address Hex	<u>.</u>										ASCII	
0019FBE0 20	00 00 00 67 6	4 69 70	0 60	2 75	73	2E	64	6C	60	00	odiplus.dll.	

Figure 2

The binary implements the API hashing technique to hide the API functions used. As we can see below, the malware computes a 4-byte hash value and compares it with a hard-coded one (0xA3E6F6C3 in this case):

text:00411030	
text:00411030 loc 411030:	
text:00411A30 mov7x edx.	word ntr [edi+2Ch]
text:00411A34 xor esi.	esi
.text:00411A36 mov eax.	[edi+30h]
.text:00411A39 xor ebx.	ebx
.text:00411A3B shr edx. 1	1
.text:00411A3D mov [ebp+)	var 4], 811C9DC5h
.text:00411A44 lea ecx,	[eax+edx*2]
.text:00411A47 cmp eax, e	ecx
.text:00411A49 cmova edx, e	esi
.text:00411A4C mov [ebp+v	var C], edx
.text:00411A4F test edx, e	edx
.text:00411A51 jz short	loc_411A87
	•
🗾 📬 🖙	
have 100 (11)	AF2 may add adu
.text:0041	LASS mov ed1, edx
-	L ₁
	V V
🔛 🚵	
.text:00411A55	
.text:00411A55	loc_411A55:
.text:00411A55	mov dl, [eax]
.text:00411A57	lea eax, [eax+2]
.text:00411A5A	movsx esi, dl
.text:00411A5D	sub dl, 41h ; 'A'
.text:00411A60	mov ecx, esi
.text:00411A62	or ecx, 20h
.text:00411A65	cmp dl, 19h
.text:00411A68	cmova ecx, esi
.text:00411A6B	inc ebx
.text:00411A6C	<pre>xor ecx, [ebp+var_4]</pre>
.text:00411A6F	imul ecx, 1000193h
.text:00411A75	mov [ebp+var_4], ecx
.text:00411A78	cmp ebx, edi
.text:00411A7A	jnz short loc_411A55
🔜 🛋 🖼	
.text:00411A7C	mov edi, [ebp+var 8]
.text:00411A7F	cmp ecx, 0A3E6F6C3h
.text:00411A85	iz short loc 411AA2

Figure 3

The malicious executable loads multiple DLLs into the address space of the process using the LoadLibraryA API:



• 004C008E 50		50	push eax	push eax eax: "g			
EIP		FF D1	Call ecx	lecx:re ~	Default (stdcall)	👻 💈 🗖 Unlock	
ecx= <ker< td=""><td>nel32.LoadLibraryA> 4C008F lockbit.exe:</td><td>(76A75980) \$C008F #BF48F</td><td></td><td></td><td>1: [esp] 0019F8E4 'gd 2: [esp+4] 0048FF90 < 3: [esp+8] 0048FF90 < 4: [esp+C] 76FE0000 n 5: [esp+10] 00000942</td><td>lockbit.EntryPoint> lockbit.EntryPoint> tdll.76FE0000</td></ker<>	nel32.LoadLibraryA> 4C008F lockbit.exe:	(76A75980) \$C008F #BF48F			1: [esp] 0019F8E4 'gd 2: [esp+4] 0048FF90 < 3: [esp+8] 0048FF90 < 4: [esp+C] 76FE0000 n 5: [esp+10] 00000942	lockbit.EntryPoint> lockbit.EntryPoint> tdll.76FE0000	
Sill Comm			n n Manual taka 1 Gan a	0019FAF4 0019	FBE4 "gdiplus.dll"		

The following DLLs have been loaded: "gdiplus.dll", "ws2_32.dll", "shell32.dll", "advapi32.dll", "user32.dll", "ole32.dll", "netapi32.dll", "gpedit.dll", "oleaut32.dll", "shlwapi.dll", "msvcrt.dll", "activeds.dll", "mpr.dll", "bcrypt.dll", "crypt32.dll", "iphlpapi.dll", "wtsapi32.dll", "win32u.dll", "Comdlg32.dll", "cryptbase.dll", "combase.dll", "Winspool.drv".

GetSystemDefaultUILanguage is utilized to retrieve the language identifier for the system default UI language of the OS. The return value is compared with multiple identifiers that correspond to CIS countries (LockBit doesn't encrypt these systems):

CIT C	0049B2FE	FF DO	Call cax	eaxige	[™] Default (stdcall)
eaver/ernel3	<	aulturi anguages (7647)	(250)	>	1: [esp] 0048FF90 <lockbit.entrypoint></lockbit.entrypoint>
		unicorcungunger (1997)			3: [esp+8] 00223000
.text:004982	FE lockbit.exe	\$982FE #9A6FE			4: [esp+C] 00000000 5: [esp+10] 00000850
dill Come t			<u>.</u>	0019FAD0 00	48FF90 lockbit.EntryPoint
			Figure 5		
			rigare 5		
			.text:00498300 mov ecx, 42	_n	
			.text:0049B308 mov [ebp+va	[10], 82Ch	
			.text:0049B30F lea edx, [e	cx-1]	
			.text:0049B312 lea ebx, [e	cx-9]	
			.text:00498318 lea edi, [e	cx-0Ah]	
			.text:0049B31B cmp ax, wor	d ptr [ebp+var_10]	
			.text:0049B31F jz short l	oc_49B38E	
			🛄 🛃 🖼		
			.text:0049B321 cmp ax,	cx	
			.text:0049B324 jz sho	int loc_49B38E	
			.text:0049B326 cmp ax,	dx	
			.text:0049B329 jz sho	rt loc_49838E	
			.text:0049B32B cmp ax	bx	
			.text:0049B32E jz sho	ort loc_49B38E	
			.text:0049B330 cmp ax	, și	
			.text:0049B333 jz sh	ort loc_49B38E	
			text:00498335 mov ec	x. 43Fh	
			.text:0049B33A cmp ax	., cx	
			.text:0049833D jz sł	ort loc_49B38E	
			.text:0049B33F mov e	cx, 440h	
			.text:0049B344 cmp a	ix, cx	
			.text:00498347 jz s	NOTE 10C_49838E	
			Figure 6		
			J		

The following language identifiers have been found:

- 0x82c Azerbaijani (Cyrillic)
- 0x42c Azerbaijani (Latin)
- 0x42b Armenian



- 0x423 Belarusian
- 0x437 Georgian
- 0x43F Kazakh
- 0x440 Kyrgyz
- 0x819 Russian (Moldova)
- 0x419 Russian
- 0x428 Tajik
- 0x442 Turkmen
- 0x843 Uzbek (Cyrillic)
- 0x443 Uzbek (Latin)
- 0x422 Ukrainian

The GetUserDefaultUILanguage routine extracts the language identifier for the user UI language for the current user. The extracted value is compared with the same identifiers from above:

004985F3	FF DO	call eax	eax:Ge v	Default (stdcall)	▼ 5 ¢ Unlock
eax= <kernel32.getuserdefaul< th=""><th>tUILanguage> (76A7</th><th>5580)</th><th></th><th>1: [esp] 0048FF90 cli 2: [esp+4] 00000008 3: [esp+8] 00223000 4: [esp+C] 00000000</th><th>OCKDIT.ENTRYPOINT></th></kernel32.getuserdefaul<>	tUILanguage> (76A7	5580)		1: [esp] 0048FF90 cli 2: [esp+4] 00000008 3: [esp+8] 00223000 4: [esp+C] 00000000	OCKDIT.ENTRYPOINT>
fill Dump 1 fill Dump 2 fills	3965F3 #9A9F3	ill come to Manual to Industria 9 cause	0019FAD0 0048	5: [esp+10] 0000082C BFF90 lockbit.EntryPo	int

Figure 7

The NtQuerySystemInformation function is utilized to retrieve the number of processors in the system (0x0 = **SystemBasicInformation**):

	 004C10E7 004C10E9 004C10EB 004C10F0 004C10F2 	6A 00 6A 2C 65 70 8A 4F 00 6A 00 E8 D9 21 F5 FF	push 0 push 2C push lockbit.4F8A70 push 0 call lockbit.4132D0			x875tatusword 0000 x875w_E8 x875w_C3 0 x875w_C2 0 x875w_C1 x875w_C4 0 x875w_E5 0 x875w_S5 x875w_P 0 x875w_U4 0
EIP	004C10F7	FF DO	call eax		eax:N	Default (stdcall)
eax= <ntdll< th=""><th>NtQuerySystemIn</th><th>formation> (7704EAEO) \$C10F7 #C04F7</th><th></th><th></th><th></th><th>1: [esp1 00000000 2: [esp+4] 004F8A70 lockbit.004F8A70 3: [esp+6] 0000002 4: [esp+C] 00000000 5: [esp+10] 0048FF90 <lockbit.entrypoint></lockbit.entrypoint></th></ntdll<>	NtQuerySystemIn	formation> (7704EAEO) \$C10F7 #C04F7				1: [esp1 00000000 2: [esp+4] 004F8A70 lockbit.004F8A70 3: [esp+6] 0000002 4: [esp+C] 00000000 5: [esp+10] 0048FF90 <lockbit.entrypoint></lockbit.entrypoint>
Ump 1	U Dump 2	Dump 3 💭 Dump 4 🐖	Dump 5 🛛 💮 Watch 1 🛛 🕅 🕬	als 🖉 Struct	0019FAE8 (00000000 004F8A70 004F8A70
Address H	ex		ASCII		^ 0019FAF0 0	0000002C

Figure 8

The binary opens a handle to the current process (0x60000 = WRITE_DAC | READ_CONTROL):

<pre></pre>	1110 50 1111 50 84 24 60 02 00 00 101 1113 57 84 24 30 00 00 00 00 101 1123 50 84 24 34 00 00 00 00 101 1124 50 84 24 34 00 00 00 00 101 1130 56 50 50 50 50 11 101 00 00 00 00 101 <td< th=""><th>push eax lea eax,dword ptr ss:[esp+206] nov dword ptr ss:[esp+206] lea eax,dword ptr ss:[esp+204] lea eax,dword ptr ss:[esp+204] push eax nov dword ptr ss:[esp+130] nov dword ptr ss:[esp+130] nov dword ptr ss:[esp+130] nov dword ptr ss:[esp+264].0 nov dword ptr ss:[esp+265].0 nov dword ptr ss:[</th><th>eax:Nt eax:Nt eax:Nt eax:Nt eax:Nt eax:Nt</th><th>x87r5 00000000000000000 5TS Empty 0.00000000 x87r6 3FF600000000000000 5TS Empty 0.00000000 x87r 3678000000000000000 5TS Empty 0.00000000 x87Tagword FFFF x87Tw_0 8 (Empty) x87Tw_1 3 (Empty) x87Tw_2 8 (Empty) x87Tw_3 3 (Empty) x87Tw_4 8 (Empty) x87Tw_7 3 (Empty) x87Tw_4 8 (Empty) x87Tw_7 3 (Empty) x87Tw_5 8 (Empty) x87Tw_7 8 (Empty) x87Tw_5 8 (Empty) x87Tw_7 9 (Empty) x87Tw_7 9 (Empty) x87Tw_5 8 (Empty) x87Tw_7 9 (Empty) x8</th></td<>	push eax lea eax,dword ptr ss:[esp+206] nov dword ptr ss:[esp+206] lea eax,dword ptr ss:[esp+204] lea eax,dword ptr ss:[esp+204] push eax nov dword ptr ss:[esp+130] nov dword ptr ss:[esp+130] nov dword ptr ss:[esp+130] nov dword ptr ss:[esp+264].0 nov dword ptr ss:[esp+265].0 nov dword ptr ss:[eax:Nt eax:Nt eax:Nt eax:Nt eax:Nt eax:Nt	x87r5 00000000000000000 5TS Empty 0.00000000 x87r6 3FF600000000000000 5TS Empty 0.00000000 x87r 3678000000000000000 5TS Empty 0.00000000 x87Tagword FFFF x87Tw_0 8 (Empty) x87Tw_1 3 (Empty) x87Tw_2 8 (Empty) x87Tw_3 3 (Empty) x87Tw_4 8 (Empty) x87Tw_7 3 (Empty) x87Tw_4 8 (Empty) x87Tw_7 3 (Empty) x87Tw_5 8 (Empty) x87Tw_7 8 (Empty) x87Tw_5 8 (Empty) x87Tw_7 9 (Empty) x87Tw_7 9 (Empty) x87Tw_5 8 (Empty) x87Tw_7 9 (Empty) x8
Ump 1 Dump 2 Address Hex	💭 Dump 3 👹 Dump 4 👹 Dump 5	Watch 1 x= Locals Struct ASCII	0019FAE8 0019 0019FAEC 0006 0019FAF0 0019 0019FAF4 0019	FB24 0000 FD60 FD4C "(\f"



The GetSecurityInfo API is utilized to retrieve a pointer to the DACL in the returned security descriptor (0x6 = **SE_KERNEL_OBJECT**, 0x4 = **DACL_SECURITY_INFORMATION**):



Figure 10

RtlAllocateAndInitializeSid is used to allocate and initialize a SID (security identifier) structure:



Figure 11

The file extracts the ACL size information via a function call to RtlQueryInformationAcl (0x2 = **AclSizeInformation**):

Address Hex		ASCII	0019FAF0	0000000C	
Dump 1	np 2 🗱 Dump 3 🔛 Dump 4	💷 Dump 5 🛞 Watch 1 🔤 Locals	Struct 0019FAE8	0253FFEC 0019FD54	
eax= <ntdll.rtlquer< th=""><th>yInformationAcl> (7704A220</th><th>)</th><th>eax:</th><th>> Defau 2: 3: 4: 5:</th><th>At (stdcal) (sp1 0253FFEC (sp+4) 0019FD54 (sp+6) 0000000C (ssp+6) 0000000C (ssp+6) 0000000C (ssp+10) 0048FF90 <lockbit.entrypoint> (sp+10) 0048FF90 </lockbit.entrypoint> (sp+10) 0048FF90 (sp+1</th></ntdll.rtlquer<>	yInformationAcl> (7704A220)	eax:	> Defau 2: 3: 4: 5:	At (stdcal) (sp1 0253FFEC (sp+4) 0019FD54 (sp+6) 0000000C (ssp+6) 0000000C (ssp+6) 0000000C (ssp+10) 0048FF90 <lockbit.entrypoint> (sp+10) 0048FF90 </lockbit.entrypoint> (sp+10) 0048FF90 (sp+1
• 0 • 0 • 0 • 0 • 0	Ac1220 GA 02 Ac1222 GA 0C Ac1224 SD 84 24 64 02 Ac1222 FF B4 24 08 02 Ac1223 E8 78 7F F6 FF	00 00 lea eax,dword ptr ss:[esp push c push eax 00 00 push dword ptr ss:[esp+20 cml] lockbit.429180	+264] eax: eax: eax:	Rt x87 Rt x87 p+2 x87	StatusWord 0000 SM_B 0 x875M_C3 0 x875M_C2 0 SM_S75N_C0 0 x875M_E5 0 SM_SF 0 x875M_P 0 x875M_U 0

Figure 12

The executable allocates memory by calling the ZwAllocateVirtualMemory routine (0x3000 = **MEM_COMMIT** | **MEM_RESERVE**, 0x4 = **PAGE_READWRITE**). It's also important to mention that LockBit frees memory previously allocated using ZwFreeVirtualMemory:

	6A 04 68 00 30 00 00 8D 4C 24 2C 51 6A 00 8D 4C 24 28 51 6A FF FF D0	push 4 push 3000 lea ecx,dword ptr ss:[esp+2C] push ecx push ecx push fereffere push FFFFFFF call eax	R8X.028 V	x87Tw_6 5 (Empty) x87Tw_7 3 (Empty) x87StatusWord 0000 x87Sw_5 0 x87Sw_C2 0 x87Sw_C2 0 x87Sw_5 0 x87Sw_C 0 x87Sw_5 0 x87Sw_5 0 x87Sw_P 0 x87Sw_5 0 x87Sw_5 0 x87Sw_P 0 x87Sw_0 0 x87Sw_5 0 x87Sw_5 0 x87Sw_1 0 x87Sw_5 0 x87Sw_5
eax= <ntd11.zwallocatevirtua .text:004BACAS lockbit.exe:</ntd11.zwallocatevirtua 	Memory> (7704E8E0) \$BACAS #BAOAS			1: [esp] FFFFFFF 2: [esp+4] 0019FAA0 3: [esp+5] 00000000 4: [esp+C] 0019FAAC 5: [esp+10] 0003000
Dump 1 Dump 2	Dump 3 🗱 Dump 4 👪 Dump	5 🥳 Watch 1 🕅 🕬 Locals 🍃 Struct	0019FA70 FFFF 0019FA74 0019	IFFFF IFAA0
Address Hex 0019FAA0 00 00 00 00 F0 0B 0019FAB0 00 00 14 00 0C 00	0E 77 C8 03 04 77 78 00 00 00 00 90 FF 48 00 EC FF 53	ASCII 00	0019FA78 0000 0019FA7C 0019 0019FA80 0000 0019FA84 0000	0000 JFAAC 3000 0004





The RtlCreateAcl function is utilized to create and initialize an access control list (0x4 = **ACL_REVISION_DS**):

	 004C120 004C120 004C120 004C120 004C120 	66 6A 0 68 56 69 57 6A E8 2	4 1 81 F6 FF		push 4 push esi push edi call lockbit	. 429390			X875W_B 0 X875W_C3 0 X875W_C2 0 X875W_C1 0 X875W_C0 0 X875W_E5 0 X875W_5F 0 X875W_P 0 X875W_U 0
ELP	004C12	GF FF D	0		call eax			eax:Rt v	Default (stdcall)
eax= <ntdll.< th=""><th>RtlCreateAcl</th><th><pre>l> (7700A8C) .exe:\$C126F</pre></th><th>#C066F</th><th></th><th></th><th></th><th></th><th></th><th>1: [csp+4] 00000078 3: [csp+4] 000000078 4: [csp+6] 00000004 5: [csp+10] 0048FF90 <lockbit.entrypoint> 5: [csp+10] 0048FF90 <lockbit.entrypoint></lockbit.entrypoint></lockbit.entrypoint></th></ntdll.<>	RtlCreateAcl	<pre>l> (7700A8C) .exe:\$C126F</pre>	#C066F						1: [csp+4] 00000078 3: [csp+4] 000000078 4: [csp+6] 00000004 5: [csp+10] 0048FF90 <lockbit.entrypoint> 5: [csp+10] 0048FF90 <lockbit.entrypoint></lockbit.entrypoint></lockbit.entrypoint>
Dump 1	Ump 2	Dump 3	Dump 4	Dump 5	💮 Watch 1	[x=] Locals	2 Struct	0019FAEC 0431 0019FAF0 0000	F0000 00078
Address 14	av l				ASCTT	1		. 0019FAF4 000	00004

Figure 14

The RtlAddAccessDeniedAce routine is used to add an access-denied access control entry (ACE) to the ACL created earlier (0x4 = **ACL_REVISION_DS**, 0x1 = **FILE_READ_DATA**):

	 004C1279 004C1280 004C1282 004C1284 004C1284 004C1285 	FF B4 24 B4 00 0 6A 01 6A 04 57 E8 F6 81 F6 FF	00 00 push dword push 1 push 4 push edi call lockbi	otr ss:[esp	+84			x8/5t x87SW x87SW x87SW	atusword 0000 (_B 0 x87SW_C3 (_C1 0 x87SW_C0 (_SF 0 x87SW_P	0 x87SW_C2 0 x87SW_ES 0 x87SW_U	0
eax= <ntd11.< th=""><th>RtlAddAccessDeni</th><th>EF D0 edAce> (770A2940)</th><th>call eax</th><th>5</th><th></th><th></th><th>eax:RI v</th><th>Default 1: [e: 2: [e: 3: [e: 4: [e:</th><th>(stdcal) sp] 043F0000 sp+4] 00000004 sp+8] 0000001 sp+C] 0255E590</th><th></th><th>S C Unlod</th></ntd11.<>	RtlAddAccessDeni	EF D0 edAce> (770A2940)	call eax	5			eax:RI v	Default 1: [e: 2: [e: 3: [e: 4: [e:	(stdcal) sp] 043F0000 sp+4] 00000004 sp+8] 0000001 sp+C] 0255E590		S C Unlod
	ton formationene.	JELEON PEODON						s: Les	SD+10] 0048FF90	<iockdit.entr< th=""><th>yP01nt></th></iockdit.entr<>	yP01nt>
Ump 1	Dump 2	Dump 3 🔛 Dump 4	📖 Dump 5 🛛 💮 Watch 1	x= Locals	2 Struct	001	9FAES 043F	0000			
Address He 0255E590 01	x 01 00 00 00 00	00 01 00 00 00 00	ASCII			^ 000 000	9FAF0 0000	0001 E590	lacht EntruPai		

Figure 15

The malicious file obtains a pointer to the first ACE in the ACL via a function call to RtlGetAce:

	 004C12B2 004C12B3 004C12B4 004C12B8 	50 56 FF B4 24 04 02 E8 B0 82 F6 FF	00 00	push eax push esi push dword p call lockbit	tr_ss: [esp+ .429570	204		eax:Rt [esp+2	×87 ×87 ×87	SW_8 0 SW_C1 0 SW_SF 0	x875W_C3 0 x875W_C0 0 x875W_P 0	x87SW_C2 x87SW_ES x87SW_U	0
LILL	<	FF DO		call eax			_	eax:Rt v	Defa	ult (stdcall)			• S 🗘 🗌 Unlock
eax= <ntdll.r< td=""><td>tlGetAce> (7704</td><td>6A10) \$C12C0 #C06C0</td><td></td><td></td><td></td><td></td><td></td><td></td><td>2: 3: 4: 5:</td><td>[esp+4] [esp+8] [esp+C] [esp+10]</td><td>00000000 0019FD38 004BFF90 <10 004BFF90 <10</td><td>:kbit.Entry ockbit.Entr</td><td>Point> yPoint></td></ntdll.r<>	tlGetAce> (7704	6A10) \$C12C0 #C06C0							2: 3: 4: 5:	[esp+4] [esp+8] [esp+C] [esp+10]	00000000 0019FD38 004BFF90 <10 004BFF90 <10	:kbit.Entry ockbit.Entr	Point> yPoint>
Dump 1	Dump 2	Cump 3 💷 Dump 4	💷 Dump 5	👹 Watch 1	[x=] Locals	2 Struct	001	9FAEC 0253 9FAF0 0000	FFEC				
Address Hex 0253FFEC 02	00 50 00 03 00	00 00 00 00 18 00	FF FF 1F 00	ASCII	· yy · · ·		001 001	9FAF4 0019 9FAF8 0048	FD38	1ockbi	t.EntryPoint		

Figure 16

The process adds an ACE to the ACL previously created using RtIAddAce (0x4 = **ACL_REVISION_DS**):



Figure 17

LockBit sets the DACL of the current process to the ACL modified earlier by calling the SetSecurityInfo API (0x6 = **SE_KERNEL_OBJECT**, 0x4 = **DACL_SECURITY_INFORMATION**):



	O04C131 O04C131 O04C131 O04C131 O04C131 O04C132 O04C132	9 6A 0 8 57 6A 0 6A 0 0 6A 0 2 6A 0 4 FF 7 8 FF D	0 0 4 6 6 4 24 44		oush 0 oush edi oush 0 oush 0 oush 4 oush 6 oush dword call ecx	ptr ss:[esp	D+44]			ecxi Se v	x8 x8 x8 x8 x8	/IW_6 3 (Empty) X: 75tatusWord 0000 75W_8 0 x875W_C3 75W_5F 0 x875W_C 75W_5F 0 x875W_P 20050000000	0 x875W_C2 0 0 x875W_C2 0 0 x875W_E5 0 0 x875W_U 0 ▼ 5	Unlod
ecx= <advap< th=""><th>132.SetSecuri</th><th>tyInfo> (7)</th><th>#C0728</th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th>2: 3: 4: 5:</th><th>[esp+4] 00000006 [esp+8] 00000004 [esp+C] 00000000 [esp+10] 00000000</th><th></th><th></th></advap<>	132.SetSecuri	tyInfo> (7)	#C0728								2: 3: 4: 5:	[esp+4] 00000006 [esp+8] 00000004 [esp+C] 00000000 [esp+10] 00000000		
Dump 1	Dump 2	Dump 3	Dump 4	Dump 5	🛞 Watch 1	(x=) Locals	2 Struct		0019F	ADC 000 AE0 000	0029	8		
Address H 043F0000 0 043F0010 0 043F0020 E 043F0030 2 043F0040 0 043F0050 0 043F0050 0 043F0050 0	ex 4 00 78 00 04 1 01 00 00 00 F FF 1F 00 01 0 02 00 00 05 12 1 03 00 00 00 0 FC 01 00 00 0 00 00 00 00	00 00 00 00 00 00 01 02 00 01 00 14 00 00 00 05 00 00 05 00 00 05 00 00 00	01 00 14 00 00 00 00 00 00 00 00 05 FF FF 1F 00 00 00 1C 00 05 00 00 00 00 00 00 00 00 00 00 00	01 00 00 00 00 00 18 00 20 00 00 00 01 01 00 00 11 14 12 00 00 00 00 00 00 00 00 00 00 00 00 00	ASCII , x , yy ,			A	0019F 0019F 0019F 0019F 0019F 0019F 0019F 0019F 0019F	AE4 000 AE8 000 AEC 000 AF0 043 AF4 000 AF8 004 AF6 004 AFC 004 800 000 804 797 808 642	0000 0000 0000 0000 0000 0000 0000 0000 0000	4 0 0 0 0 lockbit.EntryPoin 0 lockbit.EntryPoin 3 2 0	t	

The malware modifies the hard error mode in a way that some error types are not displayed to the user (0xC = ProcessDefaultHardErrorMode, 0x7 = SEM_FAILCRITICALERRORS | SEM_NOGPFAULTERRORBOX | SEM_NOALIGNMENTFAULTEXCEPT):

EIP	> 004C137D 004C137F 004C1386 004C1386 004C1391 004C1392 004C1394 004C1396 004C1398	6A 04 8D 84 24 1 C7 84 24 1 50 6A 0C 6A FF E8 A5 84 FF D0	50 02 00 50 02 00	00 07 00	push 4 lea eax,dwor mov dword pt push eax push C push FFFFFFF call lockbit call eax	d ptr ss: r ss: esp+	esp+250] 250],7		eax:Zw eax:Zw eax:Zb v	x875tatu x875tatu x875W_B x875W_C1 x875W_SF	s (Empty) > isword 0000 0 x875w_C3 0 x875w_C0 0 x875w_P	0 x875W_7 3 (Em 0 x875W_C2 0 x875W_E5 0 x875W_U	
eax= <ntdll. .text:004C1</ntdll. 	ZwSetInformation	Process> (77	04E940) 98						>	1: [esp] 2: [esp+ 3: [esp+ 4: [esp+ 5: [esp+	FFFFFFF 4] 0000000C 8] 0019FD44 C] 00000004 10] 004BFF90	<lockbit.ent< th=""><th>ryPoint></th></lockbit.ent<>	ryPoint>
Dump 1	Dump 2	Dump 3 👯 D	ump 4	Dump 5	👹 Watch 1	x= Locals	2 Struct	00	019FAE8 FFFF	FFF			
Address He 0019FD44 07	x 00 00 00 00 00	3F 04 28 0C	00 00 0	0 00 00 00	ASCII			^ 00	019FAF0 0019F 019FAF4 00000	D44 1004	bit EntryPai	nt	-

Figure 19

The ransomware enables the SeTakeOwnershipPrivilege privilege in the current process token (0x9 = **SeTakeOwnershipPrivilege**):



Figure 20

LockBit decrypts a list of processes and services that will be stopped during the infection (the entire list can be found in the appendix):

Address	He	<															ASCII
04460000	77	78	53	65	72	76	65	72	2C	77	78	53	65	72	76	65	WxServer,wxServe
04460010	72	56	69	65	77	2C	73	71	6C	6D	61	6E	67	72	2C	52	rView, sqlmangr, R
04460020	41	67	75	69	2C	73	75	70	65	72	76	69	73	65	2C	43	Agui, supervise, C
04460030	75	6C	74	75	72	65	2C	44	65	66	77	61	74	63	68	2C	ulture,Defwatch,
04460040	77	69	6E	77	6F	72	64	2C	51	42	57	33	32	2C	51	42	winword,QBW32,QB
04460050	44	42	4D	67	72	2C	71	62	75	70	64	61	74	65	2C	61	DBMgr, gbupdate, a
04460060	78	6C	62	72	69	64	67	65	20	68	74	74	70	64	2C	66	xlbridge, httpd, f
04460070	64	6C	61	75	6E	63	68	65	72	2C	4D	73	44	74	53	72	dlauncher,MsDtSr
04460080	76	72	2C	6A	61	76	61	2C	33	36	30	73	65	2C	33	36	vr,java,360se,36
04460090	30	64	6F	63	74	6F	72	2C	77	64	73	77	66	73	61	66	Odoctor,wdswfsaf
044600A0	65	2C	66	64	68	6F	73	74	2C	47	44	73	63	61	6E	2C	e,fdhost,GDscan,
044600B0	5A	68	75	44	6F	6E	67	46	61	6E	67	59	75	2C	51	42	ZhuDongFangYu,QB
044600C0	44	42	4D	67	72	4E	2C	6D	79	73	71	6C	64	20	41	75	DBMgrN, mysqld, Au
044600D0	74	GF	64	65	73	6B	44	65	73	68	74	6F	70	41	70	70	todeskDesktopApp
044600E0	2C	61	63	77	65	62	62	72	6F	77	73	65	72	2C	43	72	,acwebbrowser,Cr
044600F0	65	61	74	69	76	65	20	43	6C	6F	75	64	2C	41	64	6F	eative Cloud, Ado
04460100	62	65	20	44	65	73	68	74	6F	70	20	53	65	72	76	69	be Desktop Servi
04460110	63	65	2C	43	6F	72	65	53	79	6E	63	2C	41	64	6F	62	ce,CoreSync,Adob
04460120	65	20	43	45	46	2C	48	65	6C	70	65	72	2C	GE	6F	64	e CEF, Helper, nod



Address	He	ĸ															ASCII
04460000	77	72	61	70	70	65	72	2C	44	65	66	57	61	74	63	68	wrapper.DefWatch
04460010	2C	63	63	45	76	74	4D	67	72	2C	63	63	53	65	74	4D	, ccEvtMgr, ccSetM
04460020	67	72	2C	53	61	76	52	6F	61	GD	2C	53	71	6C	73	65	gr, SavRoam, Sqlse
04460030	72	76	72	2C	73	71	6C	61	67	65	6E	74	20	73	71	6C	rvr, sqlagent, sql
04460040	61	64	68	6C	70	20	43	75	6C	73	65	72	76	65	72	2C	adhlp,Culserver,
04460050	52	54	56	73	63	61	6E	2C	73	71	6C	62	72	6F	77	73	RTVscan, sqlbrows
04460060	65	72	2C	53	51	4C	41	44	48	4C	50	2C	51	42	49	44	er, SQLADHLP, QBID
04460070	50	53	65	72	76	69	63	65	2C	49	6E	74	75	69	74	2E	PService, Intuit.
04460080	51	75	69	63	6B	42	6F	6F	68	73	2E	46	43	53	2C	51	QuickBooks.FCS,Q
04460090	42	43	46	4D	6F	6E	69	74	6F	72	53	65	72	76	69	63	BCFMonitorServic
044600A0	65	2C	20	6D	73	GD	64	73	72	76	2C	74	6F	6D	63	61	e, msmdsrv,tomca
044600B0	74	36	2C	7A	68	75	64	6F	6E	67	66	61	6E	67	79	75	t6,zhudongfangyu
044600C0	2C	76	6D	77	61	72	65	2D	75	73	62	61	72	62	69	7.4	,vmware-usbarbit
044600D0	61	74	6F	72	36	34	2C	76	GD	77	61	72	65	2D	63	6F	ator 64, vmware-co
044600E0	6E	76	65	72	74	65	72	2C	64	62	73	72	76	31	32	2C	nverter,dbsrv12,
044600F0	64	62	65	6E	67	38	2C	4D	53	53	51	4C	24	4D	49	43	dbeng8,MSSQL\$MIC
04460100	52	4F	53	4F	46	54	23	23	57	49	44	2C	4D	53	53	51	ROSOFT##WID, MSSQ
04460110	4C	24	56	45	45	41	4D	53	51	4C	32	30	31	32	2C	53	L\$VEEAMSQL2012,S
04460120	51	4C	41	67	65	6E	74	24	56	45	45	41	4D	53	51	4C	QLAgent \$VEEAMSQL

The malware calls the ZwOpenProcessToken API in order to open the access token associated with the current process (0x8 = **TOKEN_QUERY**):



Figure 23

GetTokenInformation is utilized to extract the user account of the token (0x1 = TokenUser):

 0048829D 048829E 048824E 0488240 80 40 80 00488243 11 00488245 6A 01 00488246 F75 F8 	push eck push 4C lea eck, dword ptr ss:[ebp-80] push eck push 1 push dword ptr ss:[ebp-8]	x875tatusWc x875W_B 0 x875W_C1 0 x875W_SF 0	ord 0000 x875%_C3 0 x875%_C2 0 x875%_C0 0 x875%_ES 0 x875%_P 0 x875%_U 0
eax= <advapi32.gettokeninformation> (73ASF7F0)</advapi32.gettokeninformation>	[call eax	eax:Ge Default (stdcal) 1: [esp] 00 2: [esp+4]	▼ 5 🗘 Unlod 000298 00000001 00185470
.text:004882A9 lockbit.exe:\$882A9 #8A6A9		4: [esp+C] 5: [esp+10]	0000004C 0019FACC
Ump 1 Ump 2 Ump 3 Ump 4	Dump 5 👹 Watch 1 🛛 🕸 I Locals 🎾 Struct	0019FA4C 00000298 0019FA50 00000001	
Address Hex 0019FA70 4E 00 00 00 50 00 00 00 00 05 05 12 0019FA70 4E 00 00 00 50 00 00 00 00 00 55 05 12	ASCII	0019FA54 0019FA70 0019FA58 0000004C 0019FA5C 0019FACC	

Figure 24

The AllocateAndInitializeSid routine is used to allocate and initialize a security identifier (SID) with a single subauthority:





The executable compares two security identifier (SID) values using the EqualSid API:



0048856E FF 75 EC 00488571 FF 75 80	push dword ptr ss: ebp-14 push dword ptr ss: ebp-80	x875W_SF 0	x875W_P 0 x875W_U 0
EIF > 00488574 FF D0 < <	call eax	Default (stdcall)	▼ 5 ↓ Unlock 9FA78
eax= <advap132.equa151d> (73A60B70) .text:004BB574 lockbit.exe:\$BB574 #BA974</advap132.equa151d>		2: [esp+4] 0 3: [esp+3] 0 4: [esp+5] 0 5: [esp+1] 0	255E5F0 43F0000 0000003 00223000
Ump 1 Ump 2 Ump 3 Ump 4 Um Dump 4	np 5 👹 Watch 1 🛛 🕅 Locals 🎾 Struct	0019FA58 0019FA78 0019FA5C 0255E5F0	
Address Hex	ASCII	0019FA60 043F0000 0019FA64 00000003	
Address Hex 025555F0 01 01 00 00 00 00 00 05 12 00 00 00 00 00 00	ASCII	1 0019E468 00223000	

There is a recurrent function call to GlobalMemoryStatusEx that retrieves information about the current usage of both physical and virtual memory:

• 0048A446 51	push ecx			
(B12) 004BA447 FF D0	call eax	eax:GI V	Default (stdcall)	▼ 5 🗘 🗌 Unloc
<pre>eax=<kernel32.globalmemorystatusex> (76A75 .text:004BA447 lockbit.exe:\$BA447 #B9847</kernel32.globalmemorystatusex></pre>	570)		1: [esp] 0019FA90 2: [esp+4] 043F0000 3: [esp+8] 0000003 4: [esp+C] 00223000 5: [esp+10] 5E1FB3E5	
	· · · · · · · · · · · · · · · · · · ·	0019FA7C 001	9FA90	

Figure 27

LockBit creates a new thread using the CreateThread API, which will run the sub_4DF310 function:

	 0048A73D 0048A73E 0048A740 0048A743 0048A746 0048A748 	51 6A 00 FF 75 FF 75 6A 00 6A 00	E8 E4		push ecx push 0 push dword p push dword p push 0 push 0	tr ss: ebp- tr ss: ebp-	18 10			X8 X8 X8 X8	75tatusword 0000 75w_B 0 x875w_C3 0 x875w_C2 0 75w_SF 0 x875w_C 0 x875w_ES 0 75w_SF 0 x875w_P 0 x875w_U 0
EIP	004BA74A	FF DO			call eax				eax:Cr	Def	ault (stdcall) 💌 5 🔅 🗌 Linlock
eax= <kernel< th=""><th>32.CreateThrea</th><th>d> (76A749 e:\$BA74A (</th><th>(BO) (B9B4A</th><th></th><th></th><th></th><th></th><th></th><th></th><th>2: 3: 4: 5:</th><th>[esp+4] 00000000 [esp+8] 0040F310 lockbit.004DF310 [esp+C] 0000000 [esp+10] 00000000</th></kernel<>	32.CreateThrea	d> (76A749 e:\$BA74A ((BO) (B9B4A							2: 3: 4: 5:	[esp+4] 00000000 [esp+8] 0040F310 lockbit.004DF310 [esp+C] 0000000 [esp+10] 00000000
Dump 1	Ump 2	Dump 3	Dump 4	Dump 5	💮 Watch 1	[x=] Locals	2 Struct	00	19FA68 00	00000	0
Address H	ex				ASCII			^ 00	19FA70 00	4DF31	0 lockbit.004DF310
0019FA90 40 0019FAA0 00	00 00 00 20 0	0 00 00 00 00 00 00 00 00 00 00 00 00 0	AO EF FF	00 00 00 00 00	@D+	ý		00	19FA78 00 19FA7C 00	00000 19FAD	0

Figure 28

ZwSetInformationThread is used to hide the thread from our debugger however, the x32dbg's plugin called ScyllaHide can circumvent its effect (0x11 = **HideThreadFromDebugger**):

	 004BA75A 004BA75C 004BA75C 004BA760 004BA761 	6A 00 6A 00 6A 11 56 E8 8A AD F5	FF	ush 0 ush 0 ush 11 ush esi all lockbit	.4154F0				x87 x87 x87 x87	Statusword 0000 75W_B 0 x875W_C3 0 75W_C1 0 x875W_C0 0 75W_SF 0 x875W_P 0	x875W_C2 x875W_E5 x875W_U	0 0
eax= <ntd]].< th=""><th>2wSetInformatic</th><th>FF D0</th><th>E830)</th><th>all eax</th><th></th><th></th><th></th><th>eax:Zm ~</th><th>Defa</th><th>ult (stdcall) [esp] 00000298</th><th></th><th>5 🗘 🗌 Unlock</th></ntd]].<>	2wSetInformatic	FF D0	E830)	all eax				eax:Zm ~	Defa	ult (stdcall) [esp] 00000298		5 🗘 🗌 Unlock
.text:004BA	1766 lockbit.ex	e:\$BA766 #B9B66							2: 3: 4: 5:	[esp+4] 00000011 [esp+8] 00000000 [esp+C] 00000000 [esp+10] 043F0000		
Dump 1	Ump 2	Dump 3 🛛 💭 Du	mp 4 👹 Dump 5	💮 Watch 1	[x=] Locals	2 Struct	0019	FA70 0000 FA74 0000	0298	3		
Address He	ex			ASCII			^ 0019 0019	FA78 0000 FA7C 0000	00000			

Figure 29

Thread activity – sub_4DF310 function

The shutdown priority for the current process relative to other processes in the system is set to 0, which means that it's set to be the last process to be shut down:

004DF443 6A 00 004DF445 6A 00	push 0 push 0	x875W_SF 0 x875W_P 0 x875W_U 0	
0040F447 FF D0 <	call cax	eax:Se → Default (stdcall)	Unlock
eax= <kernel32.setprocessshutdownparameters> .text:004DF447 lockbit.exe:SDF447 #DE847</kernel32.setprocessshutdownparameters>	(76A764E0)	1: [esp] 00000000 2: [esp+4] 0000000 3: [esp+4] 0040F310 lockbit.0040F310 4: [esp+5] 0040F310 lockbit.004DF310 5: [esp+10] 0000000	
Ump 1 Ump 2 Ump 3 Ump 4	🗱 Dump 5 👹 Watch 1 🛛 🕼 I Locals 🎾 Struct	05C3FF3C 00000000 05C3FF40 00000000	





GetSystemDirectoryW is utilized to retrieve the path of the system directory:



Figure 31

The process creates an activation context and activates it using the CreateActCtxW and ActivateActCtx routines:

-	004C3E	A7 51	-		push ecx					THE OWNER			-	
ETA	** 004C3E	AS PF D	0		call eax				eax:cr v	Defa	ault (stdcall)		- 5	Cunloc
eax= <kernel< td=""><td>I32.CreateAc</td><td>tCtxw> (76A .exe:\$C3EA8</td><td>77740) #C32A8</td><td></td><td></td><td></td><td></td><td></td><td></td><td>1: 2: 3: 4: 5:</td><td>[esp] 05C: [esp+4] 05 [esp+8] 03 [esp+C] 70 [esp+10] 0</td><td>8FD10 5C3FD0C 17CCBA2 5A60000 kernel32.76A6 00000000</td><td>0000</td><td></td></kernel<>	I32.CreateAc	tCtxw> (76A .exe:\$C3EA8	77740) #C32A8							1: 2: 3: 4: 5:	[esp] 05C: [esp+4] 05 [esp+8] 03 [esp+C] 70 [esp+10] 0	8FD10 5C3FD0C 17CCBA2 5A60000 kernel32.76A6 00000000	0000	
Dump 1	Dump 2	Dump 3	Dump 4	Dump 5	🛞 Watch 1	[x=] Locals	3 Struct	050	3FC58 05C 3FC5C 05C	3FD1 3FD0	0			
Address He 05C3FD10 20 05C3FD20 30	EX 0 00 00 00 1 0 FD C3 05 7	C 00 00 00 C 00 00 00	DC FC C3 05	00 00 00 00 00	ASCII	A		▲ 050 050 050	3FC60 017 3FC64 76A 3FC68 000 3FC66 000	CCBA 6000 0000	2 0 kernel32 0 8 return t	2.76A60000	740	
							Figure	32						
	 004C38 004C38 004C38 004C38 	A7 51 A8 FF 6 AA 50	00		push ecx call eax push eax					x8 x8	7SW_C1 0 7SW_SF 0	x875W_C0 0 x875W_E5 x875W_P 0 x875W_U	0	
EIP	004C38	AB FF (06		call esi				est:Ac v	Def	ault (stdcall)		- 5	🗘 🗌 Unior
esi= <kerne< td=""><td>132.Activate BEAB lockbit</td><td>ACTCTX> (76</td><td>A776E0) #C32AB</td><td></td><td></td><td></td><td></td><td></td><td></td><td>1: 2: 3: 4: 5:</td><td>[esp] 0000 [esp+4] 05 [esp+8] 03 [esp+C] 70 [esp+10] 0</td><td>00000 SC3FD0C 17CCBA2 SA60000 kernel32.76A6 00000000</td><td>0000</td><td></td></kerne<>	132.Activate BEAB lockbit	ACTCTX> (76	A776E0) #C32AB							1: 2: 3: 4: 5:	[esp] 0000 [esp+4] 05 [esp+8] 03 [esp+C] 70 [esp+10] 0	00000 SC3FD0C 17CCBA2 SA60000 kernel32.76A6 00000000	0000	
Dump 1	Dump 2	Ump 3	Ump 4	Dump 5	🛞 Watch 1	[x=] Locals	Struct	050	3FC58 000 3FC5C 05C	00000 3FD0	o c			

Figure 33

The binary registers and initializes specific common control window classes using the InitCommonControls API:

ETE 004C4ED7 FF D0	call eax	eax: In V	Default (stdcall)	▼ 5 \$ Unlock
<pre>eax=<comct132.initcommoncontrols> (6FD50AE0) .text:004C43D7 lockbit.exe:\$C43D7 #C37D7</comct132.initcommoncontrols></pre>			1: [esp] 017CCBA2 2: [esp+4] 76A60000 kei 3: [esp+8] 00000000 4: [esp+C] 000317F8 &" 5: [esp+10] 000002AA	rnel32.76A60000 nfow"
	Barris Manual Indiana () and	05C3FC60 017C	CBA2	

Figure 34

GdiplusStartup is used to initialize Windows GDI+:

	004DES 004DES 004DES 004DES 004DES	047 6A 049 80 040 51 04E 68	00 4C 24 34 F0 8F 4F 00		push 0 lea ecx,dwor push ecx push lockbit	d ptr ss:	esp+34]		×87 ×87 ×87	75W_B 0 x875W_C3 75W_C1 0 x875W_C0 75W_SF 0 x875W_P	0 x875W_C2 0 x875W_E5 0 x875W_U	000	
EIP	004DE9	53 FF	DO		call eax			eax:Gd ~	Defa	ault (stdcall)		- 5	Unlock
eax= <gdipl< th=""><th>us.GdiplusSt E953 lockbit</th><th>artup> (6C</th><th>580850) 3 #DDD53</th><th></th><th></th><th></th><th></th><th></th><th>1: 2: 3: 4: 5:</th><th>[esp] 004F8FF0 lock [esp+4] 05C3FF28 [esp+8] 0000000 [esp+C] 017CCBA2 [esp+10] 76A60000 k</th><th>ernel32.76A6</th><th>0000</th><th>0</th></gdipl<>	us.GdiplusSt E953 lockbit	artup> (6C	580850) 3 #DDD53						1: 2: 3: 4: 5:	[esp] 004F8FF0 lock [esp+4] 05C3FF28 [esp+8] 0000000 [esp+C] 017CCBA2 [esp+10] 76A60000 k	ernel32.76A6	0000	0
UIII Dump 1	Dump 2	🕮 Dump 3	Dump 4	💷 Dump 5	🛞 Watch 1	x= Locals	2 Struct	OSC3FEEC 004 OSC3FEFO 05C	F8FFC	0 lockbit.004F8FF0			
A status area of 1 and	Land 1				LACCET			05C3FEF4 000	00000	0.			

Figure 35

The malicious file initializes the COM library on the current thread:

	push 0		
CIP 004DEA9A FF D0	call eax	eax:Co Y	Default (stdcall) - 5 🗘 Unlod
eax= <ole32.coinitialize> (76756630)</ole32.coinitialize>			1: [esp] 00000000 2: [esp+4] 017CCBA2 3: [esp+8] 76A60000 kernel32.76A60000
.text:004DEA9A lockbit.exe:\$DEA9A #DDE9A			4: [esp+C] 00000000 5: [esp+10] 00000000
and class class class class	a and a a a a a a a a a a a a a a a a a	05C3FEF4 0000	0000

Figure 36

The GetVersion routine is used to retrieve the operating system version:



004DEBE4 FF D0	call eax	1	eax:Ge V	Default ((stdcall)	•	5 🔹 🗌 Unlock
eax= <kernel32.getversion> (76A755C0)</kernel32.getversion>				1: [es 2: [es 3: [es	<pre>sp] 017CCBA2 sp+4] 76A60000 sp+8] 00000000</pre>	kerne132.76A6000	00
.text:004DEBE4 lockbit.exe:\$DEBE4 #DDFE4				4: [es 5: [es	sp+C] 00000000 sp+10] 0000031	5	
Millionest Millionest Millionest Millionest	r 🦀 minada ti iyati anala	9) charact	05C3FEF8 0170	CBA2			

CreateStreamOnHGlobal is utilized to create a stream object that uses an HGLOBAL memory handle to store the content:

	 004DED 004DED 004DED 004DED 	38 51 39 6A 0 38 6A 0	01		push ecx push 1 push 0				x8 x8	75W_C1 0 x875W_C0 75W_SF 0 x875W_P	1 x87SW_ES 0 1 x87SW_U 0	
EIP	→• 004DED	3D FF D	0		call eax			eax:Cr ~	Defa	ault (stdcall)	▼ 5 \$	Unloci
eax= <combas< th=""><th>D3D lockbit</th><th>exe:\$DED3D</th><th><pre>1> (76C007B0 #DE13D</pre></th><th>)</th><th></th><th></th><th></th><th></th><th>2: 3: 4: 5:</th><th>[esp+4] 00000000 [esp+8] 05C3FF04 [esp+C] 017CCBA2 [esp+10] 76A60000 k</th><th>erne132,76A60000</th><th></th></combas<>	D3D lockbit	exe:\$DED3D	<pre>1> (76C007B0 #DE13D</pre>)					2: 3: 4: 5:	[esp+4] 00000000 [esp+8] 05C3FF04 [esp+C] 017CCBA2 [esp+10] 76A60000 k	erne132,76A60000	
Dump 1	Dump 2	Ump 3	Dump 4	Dump 5	🛞 Watch 1	[x=] Locals	2 Struct	05C3FEEC 000 05C3FEF0 000	0000	00		1
Address W	ev.				ASCTT	1		05C3FEF4 05C	3FF0)4		

Figure 38

The stream content is modified, and the process uses the GdipCreateBitmapFromStream function to create a Bitmap object based on the stream:

	 004DF0 004DF0 	0E0 68 0	C 8F 4F 00 74 24 10		push lockbit	.4F8FCC tr ss:[esp	+10			x8	75W_SF 0 x87SW_P 1 x87SW_U 0
EIP	* 004DF0	DE9 FF I	00		call eax				eax:Gd1	> Def	ault (stdcall) - S S Unlock
eax= <gdipli< th=""><th>FOE9 lockbit</th><th>eBitmapFrom</th><th>#DE4E9</th><th>580810)</th><th></th><th></th><th></th><th></th><th></th><th>2: 3: 4: 5:</th><th>[esp+3] 00478FCC lockbit.004F8FCC [esp+8] 017CCBA2 [esp+C] 76A60000 kernel32.76A60000 [esp+10] 00000000</th></gdipli<>	FOE9 lockbit	eBitmapFrom	#DE4E9	580810)						2: 3: 4: 5:	[esp+3] 00478FCC lockbit.004F8FCC [esp+8] 017CCBA2 [esp+C] 76A60000 kernel32.76A60000 [esp+10] 00000000
Dump 1	Dump 2	Dump 3	Ump 4	Dump 5	🛞 Watch 1	x= Locals	2 Struct	05	C3FEF4	0255CB7 004F8FC0	0 C lockbit.004F8FCC
Address H	ex 0 58 83 76 5	3 54 52 4D	01 00 00 00	00 00 00 00	ASCII			^ 05 05	C3FEF8 C3FEFC	017CCBA	2 kernel32.76A60000

Figure 39

The malware loads the standard arrow cursor resource via a function call to LoadCursorW (0x7F00 = **IDC_ARROW**):

	 004C47 004C47 	97 68 0 90 6A 0	00 7F 00 00		push 7F00 push 0				XS	87SW_SF 0 x87SW_P	0 x875W_U 0	
EIP	004C47 <	9E FF (0		call eax			eax:Load(>	Def	fault (stdcall)	▼ 5	Cunlock
eax= <user3:< td=""><td>79E lockbit.</td><td>(75D0CF6) exe: \$C479E</td><td>#C3B9E</td><td></td><td></td><td></td><td></td><td></td><td>2: 3: 4: 5:</td><td>[esp+4] 00007F00 [esp+4] 0007F00 [esp+8] 017CCBA2 [esp+C] 76A60000 k [esp+10] 00000000</td><td>ernel32.76A60000</td><td></td></user3:<>	79E lockbit.	(75D0CF6) exe: \$C479E	#C3B9E						2: 3: 4: 5:	[esp+4] 00007F00 [esp+4] 0007F00 [esp+8] 017CCBA2 [esp+C] 76A60000 k [esp+10] 00000000	ernel32.76A60000	
Ump 1	Ump 2	Dump 3	Dump 4	Ump 5	🛞 Watch 1	[x=] Locals	2 Struct	05C3FDE8 0000 05C3FDEC 0000	0000 7F0	0		

Figure 40

GdipAlloc is utilized to allocate memory for a Windows GDI+ object:

eax= <gdip< th=""><th>us.GdipAlloc> (6C</th><th>575A40) 51676E #15B6E</th><th></th><th></th><th>1: [esp] 00000010 2: [esp+4] 00FLCEA 3: [esp+8] 025SC988 4: [esp+c] 76830000 combase.76830000 5: [esp+10] 76830000 combase.76830000</th></gdip<>	us.GdipAlloc> (6C	575A40) 51676E #15B6E			1: [esp] 00000010 2: [esp+4] 00FLCEA 3: [esp+8] 025SC988 4: [esp+c] 76830000 combase.76830000 5: [esp+10] 76830000 combase.76830000
EIP	• 00416767 • 00416769 • 00416765	03 C6 A3 A4 8D 4F 00 FF D0	add eax,es1 mov dword ptr ds:[<&Gd1pAlloc>],eax call eax	eax:Gdip4 004F8DA4: eax:Gdip4 ~	x875W_SF 0 x875W_P 0 x875W_U 0
	 00416754 00416757 0041675A 0041675E 00416761 00416764 	FF 75 E8 8D 04 48 0F 87 0C 30 8B 42 1C 8D 04 88 8B 04 30	<pre>push dword ptr ss:@ebp-18] lea eax.dword ptr ds:[eax+ecx*2] movz x ecx,word ptr ds:[eax+ess1] mov eax.dword ptr ds:[eax+ess1] lea eax.dword ptr ds:[eax+ecx*4] mov eax.dword ptr ds:[eax+ecs1]</pre>	eax: Gdipa eax: Gdipa eax: Gdipa	x87TW_6 3 (Empty) x87TW_7 3 (Empty) x87TS_6 3 (Empty) x87StatusWord 4000 x87SW_B 0 x87SW_C3 1 x87SW_C2 0 x87SW_C1 0 x87SW_C3 0 x87SW_C5 0

Figure 41

There is another call to GdipCreateBitmapFromStream followed by a call to GdipDisposeImage, which releases resources used by the Image object:

EIP	0041A851 FF 71 04 0041A854 FF D0	push dword ptr ds:[ecx+4]	eax: Gd1p0 v	
	• <		>	efault (stdcall) 🔻 💈 🗘 Unlodo
eax= <gdi< td=""><td>plus.GdipDisposeImage> (6C67F380) 41A854 lockbit.exe:\$1A854 #19C54</td><td></td><td></td><td>: [esp+] 00000000 : [esp+4] 00FLCEA : [esp+6] 05CE2858 : [esp+C] 76B30000 combase.76B30000 : [esp+L0] 05C3FDA4</td></gdi<>	plus.GdipDisposeImage> (6C67F380) 41A854 lockbit.exe:\$1A854 #19C54			: [esp+] 00000000 : [esp+4] 00FLCEA : [esp+6] 05CE2858 : [esp+C] 76B30000 combase.76B30000 : [esp+L0] 05C3FDA4
till Dump		the prover Manual to trail and Strange	05C3FD6C 00000	000





LockBit registers a window class called "LockBit_2_0_Ransom" using the RegisterClassExW API:

004C4D88 51	push ecx				
EIP → 004C4DBC FF D0	call eax		eax:Regis ¥	Default (stdcall)	💌 💈 🗘 Unlock
eax= <user32.registerclassexw> (75D00 .text:004C4DBC lockbit.exe:\$C4DBC #</user32.registerclassexw>	0570) C41BC			1: [esp] 05C3FF08 2: [esp+4] 017CCBA2 3: [esp+8] 76A60000 4: [esp+C] 00000000 5: [esp+10] 00000205	kerne132.76A60000
Address Hex 05C3FE20 4C 00 6F 00 63 00 6B 00 42 05C3FE30 32 00 5F 00 30 00 5F 00 52 05C3FE40 6F 00 6D 00 00 00 00 4C	ASCII 00 69 00 74 00 5F 00 L.o.c.k.8.1 00 61 00 6E 00 73 00 20k.8 00 6F 00 63 00 68 00 8	. t	OSC3FDEC OSC3F	F08	

Figure 43

CreateWindowExW is used to create a window called "LockBit 2.0 Ransom" that will track the progress of the ransomware, such as the identified drives and different logs:

	push 0 push dword ptr ss:[esp+28] lea ecx,dword ptr ss:[esp+88] push 0 push 2 lea ecx,dword ptr ss:[esp+F4] push 1 eax	ecx:L"Loc ecx:L"Loc eaxiCreat V >	<pre>k87r7 00000000000000000 ST7 Empty 0.000000000 k87TagWord FFFF k87Tw_3 (Empty) x87Tw_3 (Empty) k87Tw_2 3 (Empty) x87Tw_3 3 (Empty) k87Tw_6 3 (Empty) x87Tw_5 3 (Empty) k87Tw_6 3 (Empty) x87Tw_5 3 (Empty) k87Tw_6 3 (Empty) x87Tw_5 0 k87Sw_5 0 x87Sw_C 0 x87Sw_C 0 k87Sw_5 0 x87Sw_C 0 x87Sw_C 0 k87Sw_5 0 x87Sw_C 0 x87Sw_U 0 efwt(gtdad)</pre>
The Dame 1 the Dame 2 the Dame 2 the Dame 4 the Dame 5	Winteds to Irail Lands 9 Struct	05C3FDC0 000000	: [esp+10] 80000000 001
ear comp 1 ear comp 2 ear comp 3 ear ear< ear< ear	Watch 1 M=10Coss 2 struct ASCII	05C3PDC4 05C3PF 05C3PDC8 05C3PF 05C3PDC8 05C3PF 05C3PDC0 00CF0 05C3PDD0 00000 05C3PDD8 00000 05C3PDD8 00000 05C3PDE8 004000 05C3PDE8 004000 05C3PDE2 000000	BC L"LockBit_2_0_Ransom" 70 L"LockBit_2.0 Ransom" 80 80 80 80 80 80 80 80 80 80

Figure 44

The new window is hidden using the ShowWindow routine (0x0 = **SW_HIDE**):

004C5624 6A 00 push 0 004C5626 FF 35 00 8B 4F 00 push dword ptr ds:[4F8800]		×s	x875W_SF 0 x875W_P 1 x875W_U 0			
€19 004C562C FF D0 <	call eax		eax: Showi V	fault (stdcall)	🔻 💈 🗘 Unlock	
eax= <user32.showwindow> (75D0F4A0) .text:004C562C lockbit.exe:\$C562C #C</user32.showwindow>	4A2C		1: 2: 3: 4: 5:	[esp] 034F024A [esp+4] 0000000 [esp+8] 017CCBA2 [esp+C] 76A60000 ke [esp+10] 00000000	rne132.76A60000	
🚛 Dump 1 🚛 Dump 2 🚛 Dump 3 👹	🗒 Dump 4 🛛 👹 Dump 5 🛛 🛞 Watch 1 🛛 🗠	-I Locals 🛛 🖉 Struct	05C3FDE8 034F024 05C3FDEC 0000000	A		

Figure 45

The UpdateWindow function is utilized to update the client area of the specified window by sending a WM_PAINT message to the window:

004C5770 FF 35 00	88 4F 00 push dword ptr ds: [4F8800]	apy tindat y		
	call cax	> De	efault (stdcall)	👻 5 🗘 🗌 Unlock
<pre>eax=<user32.updatewindow> (75D008D0) .text:004C5776 lockbit.exe:\$C5776 #C483</user32.updatewindow></pre>	76	1	<pre>1 [esp] 034F024A 1 [esp+4] 017CCBA2 2 [esp+8] 76A60000 kernel32 1 [esp+C] 00000000 2 [esp+10] 75D6A9D0 user32.</pre>	2.76A60000 75D6A9D0
filliounal filliouna filliouna fillio	and Million of Manada Indiana Grand	OSC3FDEC 034F02	44	

Figure 46

The process creates a new thread by calling the CreateThread function:

	 004DF59E 004DF5A0 004DF5A2 004DF5A4 004DF5A4 004DF5A9 004DF5A8 	0040F595 6A 00 puth 0 0040F5A0 6A 00 puth 0 0040F5A2 6A 00 puth 0 0040F5A2 6A 00 puth 0 0040F5A3 6A 00 puth 10ckb1t,4C3430 0040F5A8 6A 00 puth 0 0040F5A9 6A 00 puth 0					X8 X8 X8 X8	75tatusword 4020 75w_B 0 x875w_(75w_C1 0 x875w_(75w_SF 0 x875w_)	3 1 0 0 1	x875W_C2 x875W_E5 x875W_U	000			
eax= <kernel< th=""><th>9 < 32.CreateThread> SAD lockbit.exe:</th><th>(76A745B0) \$DF5AD #DE9AD</th><th></th><th></th><th></th><th></th><th></th><th>></th><th>Defi 1: 2: 3: 4: 5:</th><th>ault (stdcall) [esp] 00000000 [esp+4] 00000000 [esp+8] 004C3430 [esp+C] 0000000 [esp+10] 0000000</th><th>1ock</th><th>bit.004C34</th><th>▼ 5130</th><th>Unloc</th></kernel<>	9 < 32.CreateThread> SAD lockbit.exe:	(76A745B0) \$DF5AD #DE9AD						>	Defi 1: 2: 3: 4: 5:	ault (stdcall) [esp] 00000000 [esp+4] 00000000 [esp+8] 004C3430 [esp+C] 0000000 [esp+10] 0000000	1ock	bit.004C34	▼ 5130	Unloc
Dump 1	Dump 2	Cump 3 🗰 Dump 4	Dump 5	🗑 Watch 1	[x=] Locals	3 Struct		SC3FF2C 0000	0000	}				
Address He OSC3FDE8 78 OSC3FDF8 00	x 57 4C 00 4A 02 00 00 00 00 00 A9	4F 03 A2 CB 7C 01 06 75 B2 03 00 00	00 00 A6 76 XV B0 CB D6 75	SCII WL.J.O.¢E D8Ou*	- Eou		10000	05C3FF34 004C 05C3FF38 0000 05C3FF3C 0000 05C3FF40 0000	3430 0000 0000	lockbit.004C34	30			





LockBit defines a Shift+F1 hot key for the new window that can be used to unhide it (0x70 = **VK_F1**, 0x4 = **MOD_SHIFT**):

Address	Hex				ASCII			^	05C3FF3C	0000000	4				
Dump 1	Dump 2	Dump 3	Dump 4	Ump 5	👹 Watch 1	Ix=I Locals	2 Struct		05C3FF34 05C3FF38	034F024 0000000	A 1				
eax= <user< th=""><th>0040F(0040F) 0040F(232.RegisterHo DF6EA lockbit</th><th>E2 GA E4 FF EA FF tKey> (75D0 exe: \$DF6EA</th><th>01 35 00 88 4F D0 F070)</th><th>00</th><th>push 1 push dword call cax</th><th>otr ds:[4F8</th><th>5800]</th><th>•</th><th>eaxtRe</th><th>De 211 211 211 211 211 211 211 21</th><th>fault (std [esp] [esp+ [esp+ [esp+ [esp+ [esp+</th><th>call) 034F024A 4] 0000000 8] 0000000 6] 0000007 10] 004DF31</th><th>P 1 x875W_ 1 x8</th><th>▼ 5 : DF310</th><th>Unlod</th></user<>	0040F(0040F) 0040F(232.RegisterHo DF6EA lockbit	E2 GA E4 FF EA FF tKey> (75D0 exe: \$DF6EA	01 35 00 88 4F D0 F070)	00	push 1 push dword call cax	otr ds:[4F8	5800]	•	eaxtRe	De 211 211 211 211 211 211 211 21	fault (std [esp] [esp+ [esp+ [esp+ [esp+ [esp+	call) 034F024A 4] 0000000 8] 0000000 6] 0000007 10] 004DF31	P 1 x875W_ 1 x8	▼ 5 : DF310	Unlod
	004DF0 004DF0	DE 6A 50 6A	70		push 70 push 4					X	375W_B	0 x875W_	C3 1 X875W_0	2 0	

Figure 48

	 004DF81E 004DF820 004DF822 004DF824 	6A 70 6A 00 6A 02 FF 35 0	0 88 4F 00	pus pus pus	h 70 h 0 h 2 h dword p	tr ds:[4F8	1800]				x879 x879 x879	W_B 0 W_C1 0 W_SF 0	x875W_C x875W_C x875W_P	3 1 0 0 1	x87SW_C2 x87SW_ES x87SW_U	0 0 0	
EIP	004DF82A	FF DO		cal	eax				eaxiReg	× 1	Defau	it (stdcall)				- 5	Unloc
eax= <user3< th=""><th>2.RegisterHotKey F82A lockbit.exe</th><th>> (75D0F070 :\$DF82A #DE</th><th>) C2A</th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th>2: 4: 5:</th><th>espj 0 esp+4] esp+8] esp+C] esp+10</th><th>00000002 00000000 00000000 00000070 004DF310</th><th>) lock</th><th>bit.004DF</th><th>F310</th><th></th></user3<>	2.RegisterHotKey F82A lockbit.exe	> (75D0F070 :\$DF82A #DE) C2A								2: 4: 5:	espj 0 esp+4] esp+8] esp+C] esp+10	00000002 00000000 00000000 00000070 004DF310) lock	bit.004DF	F310	
Dump 1	Dump 2	Dump 3	Dump 4	Dump 5	Watch 1	x= Locals	2 Struct		OSC3FF34 OSC3FF38	034F0 00000	24A 002						
Address H	ex			AS	II			^	05C3FF3C 05C3FF40	00000	000						



GetMessageW is used to retrieve a message from the thread's message queue:

ar	 004DF95F 004DF961 004DF963 004DF965 004DF965 	6A 00 6A 00 6A 00 8D 4D D0 51	push o push o push o lea ecx,d push ecx	word ptr ss: [eb	p-30)		x875W_8 x875W_2 x875W_2 x875W_5	0 x875W_C3 1 0 x875W_C3 5 0 x875W_C0 F 0 x875W_P	1 x87SW_C2 0 x87SW_ES 1 x87SW_U	0
eax= <user32< th=""><th><pre>closed 2.GetMessagew> (F969 lockbit.exe</pre></th><th>(75D0B240) 2: \$DF969 #DED69</th><th>12411-200</th><th></th><th></th><th></th><th>Default (str 1: [esp] 2: [esp- 3: [esp- 4: [esp- 5: [esp-</th><th>cal) 05C3FF50 -4] 00000000 8] 00000000 C] 00000000 -10] 004DF310</th><th>lockbit.004DF3</th><th>• 5 🔄 🗌 Unlock</th></user32<>	<pre>closed 2.GetMessagew> (F969 lockbit.exe</pre>	(75D0B240) 2: \$DF969 #DED69	12411-200				Default (str 1: [esp] 2: [esp- 3: [esp- 4: [esp- 5: [esp-	cal) 05C3FF50 -4] 00000000 8] 00000000 C] 00000000 -10] 004DF310	lockbit.004DF3	• 5 🔄 🗌 Unlock
Dump 1	Dump 2	Dump 3 Ump 4	📖 Dump 5 🛛 🥶 Watch	1 Ix=I Locals	2 Struct	05C3FF34 05C 05C3FF38 000	3FF50 00000			
Address He	ex	CO 00174 00 FF 00	ASCII			OSC3FF3C 000 05C3FF40 000	00000			

Figure 50

The malicious file translates virtual-key messages into character messages via a call to TranslateMessage:

EIP	● 004DFA81 51 →● 004DFA82 FF D0	call eax	eax: Trans v	
	• <		>	Default (stdcall)
eax= <user32.< td=""><td>.TranslateMessage> (75D0A4A0) AB2 lockbit.exe:\$DFAB2 #DEEB2</td><td></td><td></td><td>1: [esp] 05C3FF50 2: [esp+4] 004DF310 lockbit.004DF310 3: [esp+8] 004DF310 lockbit.004DF310 4: [esp-C] 0000000 5: [esp-10] 034F024A</td></user32.<>	.TranslateMessage> (75D0A4A0) AB2 lockbit.exe:\$DFAB2 #DEEB2			1: [esp] 05C3FF50 2: [esp+4] 004DF310 lockbit.004DF310 3: [esp+8] 004DF310 lockbit.004DF310 4: [esp-C] 0000000 5: [esp-10] 034F024A
fill Dump 1	diama diama diama	and the second state of th	05C3FF40 05C3F	F50

Figure 51

DispatchMessageW is utilized to dispatch a message retrieved by the GetMessage function:

004DFD0F 004DFD10	S1 A3 38 90 4F 00	<pre>push ecx mov dword ptr ds:[<&DispatchMessagew>],eax</pre>	eax:Dispa	x875W_SF 0 x875W_P	1 x875W_U 0
EIP 0040FD15	FF DO	call eax	eax:Dispa ~	Default (stdcall)	▼ 5 🗘 Unlock
eax= <user32.dispatchmessag .text:004DFD15 lockbit.exe</user32.dispatchmessag 	ew> (75D08C10) :\$DFD15 #DF115			1: [esp] 05C3FF50 2: [esp+4] 004DF310 loc 3: [esp+8] 004DF310 loc 4: [esp+C] 00000000 5: [esp+10] 034F024A	ckbit.004DF310 ckbit.004DF310
fill Care 1 fill Care 2 fill	own a life own a life	anna e 🚵 anna e I faile anta 🗐 earait	05C3FF40 05C3	BFF50	1



Thread activity - sub_4C3430 function

The process sends the **LVM_GETITEMCOUNT** message to the newly created window (0x1004 = **LVM_GETITEMCOUNT**):



Address H	ex				ASCII			△ 0688FEC0 00 0688FEC4 00		0			
Dump 1	Dump 2	Dump 3	Dump 4	Ump 5	👹 Watch 1	[x:=] Locals	2 Struct	OGBBFEBS 00 OGBBFEBC 00	DE6024	2			
eax= <user3< th=""><th>2.SendMessageW 0D7D lockbit.e</th><th>> (75CFB90 xe:\$D007D</th><th>#D017D</th><th></th><th></th><th></th><th></th><th></th><th>1</th><th>[esp] 00 [esp+4] [esp+8] [esp+C] [esp+10]</th><th>00001004 00000000 00000000 004C3430</th><th>lockbit.004c</th><th>3430</th></user3<>	2.SendMessageW 0D7D lockbit.e	> (75CFB90 xe:\$D007D	#D017D						1	[esp] 00 [esp+4] [esp+8] [esp+C] [esp+10]	00001004 00000000 00000000 004C3430	lockbit.004c	3430
11 2	● 004D0D7 ● 004D0D7 ● 004D0D7 ● 004D0D7	6A 0 68 0 FF 3	4 10 00 00 5 80 8A 4F	00	push 0 push 1004 push dword p call eax	otr ds:[<mark>4F8</mark>	ABO]	eax: Sendly	v De	67SW_C1 0 87SW_SF 0	x875W_C0 x875W_P	0 x875W_E	S 0 0 ▼ 5 0 □ Unloc
	004D0D6i	6A 0	0		push 0	185	10 M		×	875W_B 0	x875W_C3	0 x875W_C	2 0

Figure 53

The malware calls the InvalidateRect API many times to add multiple rectangles to the window's update region:

	 004C358 004C358 004C358 	4 6A 0 6 6A 0 6 FF 3	0 10 15 D4 8A 4F	00	push 0 push 0 push dword	ptr ds:[4F8	SAD 4]				x875W_C1 0 x875W_C0 0 x875W_E5 0 x875W_SF 0 x875W_P 0 x875W_U 0				
EIP	004C358	FF D	10		call eax			•	eax:Inval ~	Defa	ault (stdcall)	• [5 🗘 🗌 Unlock		
eax= <user32< th=""><th>.InvalidateRe 358E lockbit.e</th><th>ct> (75D08 xe:\$C358E</th><th>#C298E</th><th></th><th></th><th></th><th></th><th></th><th></th><th>1: 2: 3: 4: 5:</th><th>[esp] 00/E02E6 [esp+4] 00000000 [esp+8] 00000000 [esp+C] 004C3430 [esp+10] 004C3433</th><th>lockbit.004C3430 0 lockbit.004C3430</th><th></th></user32<>	.InvalidateRe 358E lockbit.e	ct> (75D08 xe:\$C358E	#C298E							1: 2: 3: 4: 5:	[esp] 00/E02E6 [esp+4] 00000000 [esp+8] 00000000 [esp+C] 004C3430 [esp+10] 004C3433	lockbit.004C3430 0 lockbit.004C3430			
Dump 1	Dump 2	Dump 3	📖 Dump 4	Dump 5	👹 Watch 1	(x=) Locals	Struct		0688FF54 007 0688FF58 000	E02E6	5				
Address He	av				ASCTT				0688FF5C 000	00000					



We continue with the analysis of the main thread.

The CommandLineToArgvW routine obtains an array of pointers to the command line arguments:

	 0045985 0045984 0045984 0045984 0045984 0045984 	5F 50 50 88 4 53 88 4 56 FF 2	45 F4 40 10 70 44		push eax nov eax,dwor nov eax,dwor push dword p	d ptr ss: d ptr ds:[otr ds:[eax	ebp-C] eax+10] +44]	[eax+44]	x875 x875 x875 x875	tatusWo W_B 0 W_C1 0 W_SF 0	rd 0000 x87SW_C3 x87SW_C0 x87SW_P	0 x875W_0 0 x875W_0 0 x875W_0	2 0 S 0
ecx= <shell< th=""><th>0045980 < 32.CommandLin 9869 lockbit.</th><th>eToArgvw> exe:\$59869</th><th>70 DC 02 (743505E0) #58C69</th><th></th><th>tmo dword ot</th><th>r ss:febo-</th><th>241.2</th><th>></th><th>Defaul 1: [4 3: [4 5: [4]</th><th>t (stdcall) esp] 02 esp+4] esp+8] esp+C] esp+C]</th><th>532226 L"\ 0019FACC 043F0000 00000003 00223000</th><th>'C:\\Users\</th><th>▼ 5 文 🗌 Unlock</th></shell<>	0045980 < 32.CommandLin 9869 lockbit.	eToArgvw> exe:\$59869	70 DC 02 (743505E0) #58C69		tmo dword ot	r ss: f ebo-	241.2	>	Defaul 1: [4 3: [4 5: [4]	t (stdcall) esp] 02 esp+4] esp+8] esp+C] esp+C]	532226 L"\ 0019FACC 043F0000 00000003 00223000	'C:\\Users\	▼ 5 文 🗌 Unlock
Ump 1	Ump 2	🖏 Dump 3	Ump 4	Ump 5	💮 Watch 1	[x=] Locals	Struct	0019F650 025 0019F654 001	32226 9FACC	L"\"C:\	\users\\	\\Desktop	<pre>\lockbit.exe\""</pre>

Figure 55

The file tries to see if the access token is elevated by calling the NtQueryInformationToken API (0x14 = **TokenElevation**):

	 004819F9 004819FA 004819FC 004819FC 00481400 00481A00 00481A02 00481A05 	50 6A 04 8D 45 F4 50 6A 14 FF 75 FC E8 16 F8 F5 FF	push eax push 4 lea eax,dword ptr s push eax push 14 push dword ptr ss: call lockbit.411220	ebp-C]	eax:NtC eax:NtC	u x87 u x87 x87 x87 x87	Theory Statusword 0000 SW_B 0 x875W_C3 0 SW_C1 0 x875W_C3 0 SW_SF 0 x875W_C9 0	0 x875w_C2 0 0 x875w_E2 0 0 x875w_ES 0 0 x875w_U 0
eax= <ntd11.< th=""><th>NtQueryInformati</th><th>FF D0 onToken> (7704E990)</th><th>call eax</th><th></th><th>eax:NC</th><th>Defau</th><th>lt (stdcall) esp] 00000334 esp+4] 0000014</th><th>▼ 5 \$ Unlod</th></ntd11.<>	NtQueryInformati	FF D0 onToken> (7704E990)	call eax		eax:NC	Defau	lt (stdcall) esp] 00000334 esp+4] 0000014	▼ 5 \$ Unlod
.text:00481	AOA lockbit.exe:	\$B1AOA #BOEOA				4:	esp+C] 00000004 esp+10] 0019FAE8	
Dump 1	Ump 2	Dump 3 📲 Dump 4	💭 Dump 5 🛛 💮 Watch 1 🛛 🕅 🖉	als 🖉 Struct	0019FACC 0019FAD0	00000334		
Address He 0019FAE4 00	x 30 22 00 04 00	00 00 34 03 00 00 8	ASCII		0019FAD4 0019FAD8 0019FADC	0019FAE4 000000004 0019FAE8		

Figure 56

Depending on the result, the malware proceeds by decrypting the "[+] Process created with admin rights" or "[-] Process created with limited rights" strings. We know that this sample is supposed to perform UAC bypass in the case of low-level privileges however, this method wasn't employed on our Windows 10 analysis machine (it's supposed to be used on older Windows versions).

The process sends the "[+] Process created with admin rights" message to the hidden window by calling the SendMessageA API:





The binary creates a mutex called "\\BaseNamedObjects\\{3FE573D4-3FE5-DD38-399C-886767BD8875}" to ensure that only one instance of the malware is running at one time (0x1F0001 = **MUTEX_ALL_ACCESS**):

O048D97C 6A 01 push 1 push 2 push 2	A0/Im_U 5 (cmpty) A0/Im_L 5 (cmpty) X87TW_3 2 (cmpty) X87TW_3 5 (cmpty) X87TW_4 3 (cmpty) X87TW_5 3 (cmpty) X87TW_6 3 (cmpty) X87TW_7 3 (cmpty) X87TW_6 8 (cmpty) X87TW_7 3 (cmpty) X87TW_6 8 (cmpty) X87TW_7 3 (cmpty) X87TW_6 8 (cmpty) X87TW_7 2 (cmpty) X87SW_6 0 X87SW_C3 0 X87SW_C2 0 X87SW_C2 0 X87SW_5 0 X87SW_C3 0 X87SW_C2 0 X87SW_5 0 X87SW_5 0 X87SW_P 0 X87SW_U 0 0
BIG > Call eax eax:NCCH ((((eax= <ntdll.ntcreatemutant> (7704F250) > .text:004ED985 lockbit.exe:\$ED985 #ECD85</ntdll.ntcreatemutant>	Default (stdcal) Equal (stdcal) Default (stdcal) Equal (stdcal) Default (stdcal)
🗱 Dump 1 🗱 Dump 2 🗱 Dump 3 🗱 Dump 4 👯 Dump 5 🛞 Watch 1 🖾 Locals 🎾 Struct 0019573C 001	19F968 1F0001
Address Hex ASCII 00197748 001974748 0019747748 0019747748 <th< th=""><th>19796C 1000012 197000 197000 197000 197000 197000 197000 197000 197000 1970000</th></th<>	19796C 1000012 197000 197000 197000 197000 197000 197000 197000 197000 1970000

Figure 58

The NetBIOS name of the local computer is extracted using GetComputerNameW:

	00453 00453	2F1 51 2F2 FF	75 F8		push ecx	tr ssileho	-81		×8	7SW_SF 0 x	(875W_P 0 x875W_U 0
EIP	→• 00453 <	ZES FF	00		call eax			eax:GetCi V	Def	ault (stdcall)	▼ 5 C Unlock
eax= <kerne< th=""><th>132.GetCompu</th><th>terNamew> (</th><th>#526F5</th><th></th><th></th><th></th><th></th><th></th><th>2: 3: 4: 5:</th><th>[esp+4] 00: [esp+8] 76: [esp+C] 76: [esp+10] 71</th><th>0000 19F01C 756630 <ole32.coinitialize> 720000 ole32.76720000 68C6E60 <combase.couninitialize></combase.couninitialize></ole32.coinitialize></th></kerne<>	132.GetCompu	terNamew> (#526F5						2: 3: 4: 5:	[esp+4] 00: [esp+8] 76: [esp+C] 76: [esp+10] 71	0000 19F01C 756630 <ole32.coinitialize> 720000 ole32.76720000 68C6E60 <combase.couninitialize></combase.couninitialize></ole32.coinitialize>
Ump 1	Dump 2	Dump 3	Ump 4	🗱 Dump 5	😸 Watch 1	[x=] Locals	2 Struct	0019EFDS 050 0019EFDC 001	1000 9F01	0	

Figure 59

The malicious executable retrieves the name of the primary domain controller by calling the NetGetDCName function. LockBit has the ability to propagate on the network and kill processes and services via malicious GPOs (group policy objects); however, these features weren't activated in this sample:

	00453 00453 00453	632 53 633 6A (00		push ebx push 0	8			x879 x879	SW_C1 0 x87SW_ SW_SF 0 x87SW_	CO 0 x87SW_ES P 0 x87SW_U	0
	→• 00453 <	6337 FF 0	0		call eax			eax:NetG ~	Defaul	t (stdcall)		▼ 5 ¢ □ Unlod
eax= <logon< th=""><th>cli.NetGetDC 3637 lockbit</th><th>Name> (6883</th><th>#52A37</th><th></th><th></th><th></th><th></th><th></th><th>2: 2: 2: 2: 2: 2: 2: 2: 2: 2: 2: 2: 2: 2</th><th>esp+4] 0000000 esp+8] 0019F750 esp+C] 76756630 esp+10] 7672000</th><th>0 c 0 <ole32.coiniti 00 ole32.7672000</ole32.coiniti </th><th>alize></th></logon<>	cli.NetGetDC 3637 lockbit	Name> (6883	#52A37						2: 2: 2: 2: 2: 2: 2: 2: 2: 2: 2: 2: 2: 2	esp+4] 0000000 esp+8] 0019F750 esp+C] 76756630 esp+10] 7672000	0 c 0 <ole32.coiniti 00 ole32.7672000</ole32.coiniti 	alize>
Dump 1	Dump 2	Dump 3	Dump 4	Dump 5	💮 Watch 1	(x=) Locals	2 Struct	0019EFD4 0000 0019EFD8 0000	00000			
Address H	PY				ASCTT	1		0019EFDC 0019	9F75C			

Figure 60

The process opens the Run registry key using RegCreateKeyExA (0x80000001 = **HKEY_CURRENT_USER**, 0x2001F = **KEY_READ** | **KEY_WRITE**):



	 00498A94 00498A95 80 4D 00498A95 00498A98 00498A98 00498A98 00498A98 00498A94 6A 00 00498A04 6A 00 00498A2 6A 00 00498A4 6A 01 	D8 00 02 00 92	push eck lea ecx,dword ptr ss: push ecx push 0 push 0 push 0 push 0 push 0 push 0 lea ecx,dword ptr ss: push ecx push ecx push ecx	:bp-23] ebp-65]	ec ec	x: "SOF" x: "SOF" x: "SOF"	x877W_0 3 (Empty) x67TW_1 3 (Empty) x877W_2 3 (Empty) x67TW_3 3 (Empty) x877W_4 3 (Empty) x67TW_5 3 (Empty) x877W_6 3 (Empty) x67TW_7 3 (Empty) x875tatusWord 0000 x875W_6 0 x875W_C3 0 x875W_C2 0 x875W_5 0 x875W_C0 0 x875W_5 0 x875W_5 0 x875W_0 0 x875W_2 0
eax= <advap13< th=""><th>→ 00450BAAF FF D0 < 2.RegCreateKeyExA> (73A6 AF lockbit.exe:\$9BAAF #5</th><th>50020) MAEAF</th><th>call eax</th><th></th><th>ea</th><th>XIREOCI Y</th><th>Default (stdcal) 1: [esp] 80000001 2: [esp+4] 0019FA82 "SOFTWARE\\Microsoft\\Windows 3: [esp+6] 00000000 4: [esp+C] 00000000 5: [esp+10] 00000000</th></advap13<>	→ 00450BAAF FF D0 < 2.RegCreateKeyExA> (73A6 AF lockbit.exe:\$9BAAF #5	50020) MAEAF	call eax		ea	XIREOCI Y	Default (stdcal) 1: [esp] 80000001 2: [esp+4] 0019FA82 "SOFTWARE\\Microsoft\\Windows 3: [esp+6] 00000000 4: [esp+C] 00000000 5: [esp+10] 00000000
Ump 1	Dump 2 👹 Dump 3	Dump 4 🛛 🕮 Dump 5	🛞 Watch 1 🛛 🕅 🖉 🖉	2 Struct	0019	704 80000 708 0019F	001 A82 "SOFTWARE\\Microsoft\\Windows\\CurrentVersion
Address Hex 0019FA82 53 0019FA92 66 0019FA92 66 0019FA82 00 0019FA82 00 0019FA82 00	4F 46 54 57 41 52 45 5C 74 5C 57 65 6E 64 6F 77 74 56 65 72 73 69 6F 6E 00 D4 FA 19 00 3D 18 03 00 16 00 88 02 5B 18 03	4D 69 63 72 6F 73 6 73 5C 43 75 72 72 6 5 <u>C 52 75</u> 6E 00 10 0 00 38 03 00 00 00 4 00 50 50 10 10 0	ASCII SOFTWARE\Microso S ft\Windows\Curre ontVersion\Run 		00199 0019 0019 0019 0019 0019 0019 0019 0019 0019	70C 00000 710 00000 714 00000 718 00020 71C 00000 720 0019F 724 0019F	000 000 000 000 000 AC8 540

The file is looking for a registry value called "{9FD872D4-E5E5-DDC5-399C-396785BDC975}":



Figure 62

The malware establishes persistence by creating the above registry value:

	 0049CD30 0049CD31 0049CD32 0049CD34 0049CD36 0049CD36 	50 53 6A 01 6A 00 FF 75 EC FF 75 D8	push eax push ebx push 1 push 0 push dword ptr push dword ptr	ss: ebp-14 ss: ebp-28	ebx:L"\\ [ebp-14]	x8 x8 x8 x8	75tatusword 0000 75w_B 0 x875w_C3 0 x875w_C2 0 75w_C1 0 x875w_C0 0 x875w_E5 0 75w_SF 0 x875w_P 0 x875w_U 0
ENG	→• 0049CD3C	FF D2	call edx		edx:Reg	Defa	ult (stdcall) 👻 5 🗘 🗌 Unlock
edx= <advap1< td=""><td>32.RegSetValueEx</td><td>W> (73ASFE80) \$9CD3C #9C13C</td><td></td><td></td><td></td><td>1: 2: 3: 4: 5:</td><td><pre>[esp+1 00000350 [esp+4] 05040000 L"[9FD872D4-E5E5-DDC5-399C-39 [esp+8] 00000000 [esp+0] 06220000 L"\"C:\\Users\\"\Desktop\</pre></td></advap1<>	32.RegSetValueEx	W> (73ASFE80) \$9CD3C #9C13C				1: 2: 3: 4: 5:	<pre>[esp+1 00000350 [esp+4] 05040000 L"[9FD872D4-E5E5-DDC5-399C-39 [esp+8] 00000000 [esp+0] 06220000 L"\"C:\\Users\\"\Desktop\</pre>
Dump 1	Bump 2 Bal	Dump 3 Ump 4	🚛 Dump 5 💮 Watch 1 🕅	-I Locals 🌮 Struct	0019F710 0 0019F714 0	0000350 5D40000	UL"{9FD 87 2D 4~E5 E5 - DD C5 - 399C - 3967 85 BD C 975 }"
Address He	x		ASCII		▲ 0019F718 0	0000000	
06220060 00 06220070 00	00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00		0019F720 (0019F720 (0019F724 (6220000	L"\"C:\\Users\\



Registry Editor				– 🗆 ×
File Edit View Favorites Help				
Computer\HKEY_CURRENT_USER\Software\Microsoft\Windo	ws\CurrentVersion\l	Run		
> DeviceAccess	^	Name	Туре	Data
> DeviceCapabilities		ab (Default)	REG SZ	(value not set)
> Devices		10 (9ED872D4-E5E5-DDC5-399C-396785BDC975)	REG SZ	C:\Users\Desktop\lockbit.ex
Disgooffice				

Figure 64

CreateThread is used to create a new thread within the address space of the process:

0048A73D 0048A73E 0048A73E 0048A740 0048A743 0048A743 0048A743 0048A746 0048A746 0048A746 0048A748 0048A748	51 6A 00 FF 75 E8 FF 75 E4 6A 00 6A 00	push ecx push 0 push dword ptr ss: ebp-18 push dword ptr ss: ebp-1C push 0 push 0			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
eax= <kernel32.createthread< td=""><td>PF 00 (76A745B0) ** \$Ra74a #89R4a</td><td></td><td></td><td>></td><td>Default(stdcall) ▼ 5 ↓ Unled 1: [esp+4] 00000000 3: [esp+4] 00000000 3: [esp+4] 00000000 3: [esp+4] 00000000 3: [esp+4] 00000000 3: [esp+4] 00000000</td></kernel32.createthread<>	PF 00 (76A745B0) ** \$Ra74a #89R4a			>	Default(stdcall) ▼ 5 ↓ Unled 1: [esp+4] 00000000 3: [esp+4] 00000000 3: [esp+4] 00000000 3: [esp+4] 00000000 3: [esp+4] 00000000 3: [esp+4] 00000000
atilities a source of sour		M	10010 T	0019FA68 000	00000
Ump 1 Ump 2	Dump 3 Ump 4 Ump 9	Watch 1 🛛 🕅 🖉 S	ruct	0019FA6C 0000	00000
Address Hex		ASCII	~	0019FA70 004	A2EC0 10CKD11.004A2EC0
0019FAD0 01 00 00 00 C0 2 0019FAE0 DC 06 AA 73 00 0 0019FAE0 20 EE 19 00 E9 1	4A 00 00 00 00 00 00 00 00 04 22 06 E0 EF A9 73 58 02 00 46 00 80 EF A9 73 58 02 00	05 U. A.J		0019FA78 000 0019FA78 000	00000 9FAD0





As in the case of every thread creation, the binary tries to hide it from the debugger using the ZwSetInformationThread API.

A file called "C:\windows\system32\2ED873.ico" is created via a function call to ZwCreateFile (0x40000000 = GENERIC_WRITE, 0x80 = FILE_ATTRIBUTE_NORMAL, 0x5 = FILE_OVERWRITE_IF):



Figure 66

The ICO file is populated using the ZwWriteFile routine:

	 004A8937 004A8939 004A8937 004A8937 004A8937 004A8934 004A8945 004A8950 004A8951 004A8951 004A8952 004A8957 004A8957 004A8957 004A8957 	6A 00 8D 85 A0 F 50 68 EE 2C 0 68 28 25 4 8D 85 98 F 50 6A 00 6A 00 6A 00 6A 00 FF 85 68 F E8 AE 99 F	FC FF FF 00 00 4E 00 FC FF FF FE FF FF	push 0 lea eax,dword ptr ss: push eax push 2CEE push lockbit,4E2528 lea eax,dword ptr ss: push eax push 0 push 0 push 0 push dword ptr ss: push dw	ebp-360] ebp-368] -198]		eax:ZwWr eax:ZwWr eax:ZwWr eax:ZwWr	x87TagWord FFFF x87Tw_0 3 (Empt; x87Tw_2 3 (Empt; x87Tw_4 3 (Empt; x87Tw_6 3 (Empt; x87StatusWord 0 x87Sw_B 0 x87; x87Sw_C1 0 x87; x87Sw_SF 0 x87;	y) x87TW_1 3 (Em y) x87TW_3 3 (Em y) x87TW_5 3 (Em y) x87TW_7 3 (Em y) x87TW_7 3 (Em 000 SW_C3 0 x87SW_C2 SW_C0 0 x87SW_C2 SW_C 0 x87SW_U	15y) 15y) 15y) 15y) 15y) 0 0
eax= <ntd11.zww< th=""><th><pre>004A8962 < riteFile> (77) lockbit.exe:</pre></th><th>FF D0 04E7E0) \$AB962 #AAD6;</th><th>2</th><th>call eax</th><th></th><th>4</th><th>eax:Zwwni *</th><th>Default (stdcall) 1: [esp] 0000037 2: [esp+4] 00000 3: [esp+8] 00000 4: [esp+6] 00000 6: [esp+0] 00000</th><th>14 0000 0000 0000</th><th>▼ 5 € 🗆 Uni</th></ntd11.zww<>	<pre>004A8962 < riteFile> (77) lockbit.exe:</pre>	FF D0 04E7E0) \$AB962 #AAD6;	2	call eax		4	eax:Zwwni *	Default (stdcall) 1: [esp] 0000037 2: [esp+4] 00000 3: [esp+8] 00000 4: [esp+6] 00000 6: [esp+0] 00000	14 0000 0000 0000	▼ 5 € 🗆 Uni
🕮 Dump 1 👔	Dump 2	Cump 3 🗰 Du	ump 4 🛛 🕮 Dump	5 👹 Watch 1 🕅 🕬 Locals	2 Struct	00	19F33C 0000	0374		
Address Hex 004E2528 00 00 004E2538 00 00 004E2548 00 00 004E2558 00 00 004E2558 00 00	01 00 03 00 36 00 00 00 9E 03 00 00 46 10 00 00 01 00 18 00	10 10 00 00 10 20 20 00 00 10 30 30 00 00 00 28 00 00 00 00 00 00 00 00 00	00 00 18 00 68 00 00 18 00 A8 00 00 18 00 A8 10 00 00 00 20 00 03 00 00 00 00	ASCII 03 0		^ 000 000 000 000 000 000 000	19F344 0000 19F348 0000 19F34C 0019 19F350 004E 19F354 0000 19F358 0019 19F358 0019 19F35C 0000	0000 0000 F788 2528 lockbit.004 2CEE F790 0000	E2528	

Figure 67

The executable creates the "HKCR\.lockbit" registry key using ZwCreateKey (0x2000000 = **MAXIMUM_ALLOWED**):





LockBit creates the DefaultIcon subkey and sets its value to the newly created ICO file, as highlighted below:



Figure 69

Registry Editor					- 0	×
File Edit View Favorites Help						
Computer\HKEY_CLASSES_ROOT\.lockbit\Defau	ulticon					
>jtx	^	Name	Туре	Data		
.jpr		(Default)	REG_SZ	C:\windows	SysWow64\2	ED873.ic

Figure 70

Thread activity – sub_4A2EC0 function

The FindFirstVolumeW API is utilized to begin scanning the volumes of the computer:

	 004A30 004A30 004A30 	0A 68 0F 8D	04 01 00 00 8C 24 6C 01	00 00	push 104 lea ecx, dwor	d ptr ss:	esp+16C]		x875w_C1 0 x875w_C0 0 x875w_E5 0 x875w_SF 0 x875w_P 0 x875w_U 0				
EIP	→• 004AB0 <	17 FF	00		call eax			eax:Find ~	Def	ault (stdcall)		• 5	🗘 🗌 Unlock
eax= <kerne< th=""><th>132.FindFirst 3017 lockbit.</th><th><pre>exe:\$A3017</pre></th><th>#A2417</th><th></th><th></th><th></th><th></th><th></th><th>2: 3: 4: 5:</th><th>[esp+4] 00 [esp+8] 00 [esp+C] 00 [esp+10] 0</th><th>0000104 0000104 04A2EC0 lockbit.00 04A2EC0 lockbit.00 00000000</th><th>4A2EC0 4A2EC0</th><th></th></kerne<>	132.FindFirst 3017 lockbit.	<pre>exe:\$A3017</pre>	#A2417						2: 3: 4: 5:	[esp+4] 00 [esp+8] 00 [esp+C] 00 [esp+10] 0	0000104 0000104 04A2EC0 lockbit.00 04A2EC0 lockbit.00 00000000	4A2EC0 4A2EC0	
Ump 1	Dump 2	Dump 3	Dump 4	Dump 5	🛞 Watch 1	{x= Locals	2 Struct	0643EFF0 0643 0643EFF4 0000	BF16	0			

Figure 71

QueryDosDeviceW is used to obtain the current mapping for the above volume:

	004A31D9 68 004A31DE 8D 004A31E5 51 004A31E5 51 004A31E6 55	04 01 00 00 8C 24 74 03 00 00 8C 24 78 01 00 00	push 104 lea ecx,dwo push ecx lea ecx,dwo	rd ptr ss: rd ptr ss:	esp+374]	ecx:L"Vo	×8 ×8 ×8 ×8	/Statusword 0000 75%_B 0 x875%_C3 0 x875%_C2 0 75%_C1 0 x875%_C0 0 x875%_E5 0 75%_SF 0 x875%_P 0 x875%_U 0
	ODAABIEE FF	DO	call eax			eax:Quer! ~	Def	ault (stdcall) 🔹 🗧 Unlock
eax= <kernel32.0< th=""><th>QueryDosDevicew> (lockbit.exe:\$A31E</th><th>76ACE150) E #A25EE</th><th></th><th></th><th></th><th></th><th>2: 3: 4: 5:</th><th>[esp+3] 06437168 L V0100402647529-0000-0000-000 [esp+4] 06437368 [esp+8] 0000104 [esp+c] 004A2EC0 lockbit.004A2EC0 [esp+10] 004A2EC0 lockbit.004A2EC0</th></kernel32.0<>	QueryDosDevicew> (lockbit.exe:\$A31E	76ACE150) E #A25EE					2: 3: 4: 5:	[esp+3] 06437168 L V0100402647529-0000-0000-000 [esp+4] 06437368 [esp+8] 0000104 [esp+c] 004A2EC0 lockbit.004A2EC0 [esp+10] 004A2EC0 lockbit.004A2EC0
Ump 1	Dump 2 🛛 Dump 3	🗱 Dump 4 🛛 🗱 D	ump 5 🛛 🛞 Watch 1	(x=) Locals	Struct	0643EFEC 0643 0643EFF0 0643	F16	8 L"Volume{d7e47829-0000-0000-0000-10000000000
Address Hex 0643F368 5C 00 0643F378 48 00 0643F388 56 00	44 00 65 00 76 00 61 00 72 00 64 00 6F 00 6C 00 75 00	69 00 63 00 65 00 64 00 69 00 73 00 6D 00 65 00 31 00	ASCII 5C 00 N.D.e.v.1. 68 00 H.a.r.d.d. 00 00 V.o.1.u.m.	c.e.\. i.s.k. e.1		1 1 06432674 0000	1010	•

Figure 72

The malware retrieves a list of drive letters for the specified volume via a call to GetVolumePathNamesForVolumeNameW:

O0443367 FF 74 24 24 O0443370 FF 74 24 24 O0443374 S0 8C 24 70 01 00 00 O0443375 S1 O0443375 O0443375 S1 O0443375 O144 O14	
.text:004A337D lockbit.exe:\$A337D #A277D	4: [esp+C] 0643F018 5: [esp+10] 004A2EC0 lockbit.004A2EC0
💷 Dump 1 💭 Dump 2 💭 Dump 3 💭 Dump 4 💭 Dump 5 🤴 Watch 1 💷 Locals	Struct 0643EFE8 0643F160 L"\\\\7\d7e47829-0000-0000-0000-10000
Address Hex ASCII	0643EFF0 00000209 0643EFF4 0643F018
064371100 350 00 37 00 56 00 67 00 N, \7,	0643EFF8 004A2EC0 10ckb1t.004A2EC0 0643EFC0 004A2EC0 10ckb1t.004A2EC0 0643F000 00000000 0643F000 00000000 0643F008 00000320 0643F008 0000320 0643F000 766E2C04 kernel32.76AE2CD4





The drive type of the volume is extracted using GetDriveTypeW:

	→• 004A3588 51 004A3588 FF D0	push ecx call eax	ecx:L"\\ eax:GetD	Default (stdcall)	- S I Unlock
eax= <kernel< td=""><td>132.GetDr1veTypew> (76ACDFAO) 358C lockbit.exe:\$A358C #A298C</td><td></td><td></td><td>2: [esp+4] 004A2EC0 3: [esp+8] 004A2EC0 4: [esp+C] 0000000 5: [esp+10] 0000000</td><td>lockbit.004A2EC0 lockbit.004A2EC0 0</td></kernel<>	132.GetDr1veTypew> (76ACDFAO) 358C lockbit.exe:\$A358C #A298C			2: [esp+4] 004A2EC0 3: [esp+8] 004A2EC0 4: [esp+C] 0000000 5: [esp+10] 0000000	lockbit.004A2EC0 lockbit.004A2EC0 0
fill Dans 1	tille a tille a tille a	an e Maria la la la dia a	0643EFF4 064	3F160 L"\\\\?\\Volume	e{d7e47829-0000-0000-0000-10000

Figure 74

The malicious process sends a message regarding the identified volume to the LockBit hidden window, as displayed in figure 75.



Figure 75

The malicious file continues the volume search via a function call to FindNextVolumeW:

	e 00444944 68 04 01 00 00 push 104 e 00444949 80 65 24 65 01 00 00 push cct e 00444980 51 push cct e 00444981 5F 74 24 32 push doord ptr ssi@csp+16C] push cct push doord ptr ssi@csp+16C] ecxiL ^m					ecx:L"\\'	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						
EIP	->* 004A49	BS FF I	00		call eax	ne-contractor	100 B	eax:Find	Def	ault (stdcall)		5	Uniock
eax= <kerne< th=""><th>132.FindNextV 4985 lockbit.</th><th>/olumew> (7 exe:\$A4985</th><th>#A3DB5</th><th></th><th></th><th></th><th></th><th></th><th>1: 2: 3: 4: 5:</th><th>[esp] 025871C8 [esp+4] 0643F160 L [esp+8] 00000104 [esp+C] 004A2EC0 1 [esp+10] 004A2EC0</th><th>"\\\\?\\Volume ockbit.004A2EC lockbit.004A2E</th><th>{d7e4 0 C0</th><th>7829-0000</th></kerne<>	132.FindNextV 4985 lockbit.	/olumew> (7 exe:\$A4985	#A3DB5						1: 2: 3: 4: 5:	[esp] 025871C8 [esp+4] 0643F160 L [esp+8] 00000104 [esp+C] 004A2EC0 1 [esp+10] 004A2EC0	"\\\\?\\Volume ockbit.004A2EC lockbit.004A2E	{d7e4 0 C0	7829-0000
Ump 1	Dump 2	Dump 3	Ump 4	🕮 Dump 5	👹 Watch 1	(x=) Locals	2 Struct	0643EFEC 025 0643EFF0 064	871C	8 0 L"\\\\7\	d7e47829-0000-0	0000-0	0000-10000
Address H	ev.				ASCIT			0643EFF4 000	0010-	4			

Figure 76

The purpose of the malware is to find unmounted volumes and mount them.

LockBit tries to open the BOOTMGR file from the volume (0x80000000 = **GENERIC_READ**, 0x3 = **FILE_SHARE_READ** | **FILE_SHARE_WRITE**, 0x3 = **OPEN_EXISTING**, 0x80 = **FILE_ATTRIBUTE_NORMAL**):





An unmounted volume is mounted by calling the SetVolumeMountPointW routine:





LockBit sends a message regarding the successful mount operation to the hidden window (see figure 80). After the enumeration is complete, the thread exits by calling the RtlExitUserThread function.

30	● 004E0914 51 004E0915 6A 01 004E0915 68 02 04 00 00 004E0917 68 02 04 00 00 0004E091C FF 35 00 88 4F 00 CO110522 FF D0 <	push ecx push 1 push 402 push 402 call eax	eax: Send V	x875 x875 x875 Defaul	w_B 0 x875w_C2 0 x875w_C2 0 w_C1 0 x875w_C2 0 x875w_E5 0 w_SF 0 x875w_P 0 x875W_U 0 t (stdcall)
eax= <user3< th=""><th>:.SendMessagew> (75CF8900) 9922 lockbit.exe:\$E0922 #DFD22</th><th></th><th></th><th>1: [4]</th><th>esp1 00180214 esp+4] 00000402 esp+8] 0000001 esp+6] 0643E7A8 esp+10] 62518C49</th></user3<>	:.SendMessagew> (75CF8900) 9922 lockbit.exe:\$E0922 #DFD22			1: [4]	esp1 00180214 esp+4] 00000402 esp+8] 0000001 esp+6] 0643E7A8 esp+10] 62518C49
Address H 0643EBA8 5 0643EBA8 5 0643EBA8 5 0643EBA8 5 0643EBA8 5 0643EBA8 5 0643EBA8 5 0643EBA8 5 0643EC8 5 0643EBA8 5 0645EBA8 5	U Dump 2 U Dump 3 U Dump 4 U Dump 4 EX 00 6F 00 6C 00 75 00 6D 00 6F 00 6C 00 75 00 6D 00 6F 00 6C 00 75 00 6D 05 00 20 00 75 00 6D 00 6D 00 75 00 6D 00 75 00 6D 00 70 00 30	S Watch 1 Ise Locals ŷ Struct ASCII 00 V.o.1. u.m. e	Odd 12 F420 OD1 06438730 000 06438730 000 06438730 000 06438730 000 06438730 020 06438730 020 06438730 020 06438730 764 06438730 020 06438730 020 06438730 020 06438730 020 06438730 020 06438730 020 06438730 020 06438730 020 06438730 020 06438730 020 06438730 020 06438730 020 06438730 020 06438730 020 06438730 020 06438730 020 06438730 020 06438730 020	80214 00402 00001 8E7A8 1BC49 ECCE1 F0000 8E7F8 00000 00002 5201D 8EA8C 8E7F8	"SetvolumeMountPointwStub" user32.ordinal2648 return to ntdll.7706201D from ntdll.770508A0 lockbit ongazzza

Figure 80

The binary calls the SHChangeNotify API with the SHCNE_ASSOCCHANGED parameter (0x8000000 = **SHCNE_ASSOCCHANGED**):

	 004ACAE4 004ACAE6 004ACAE8 004ACAE8 	6A 00 6A 00 6A 00 6B 00 00 00 08	push 0 push 0 push 0			X8/5W_B 0 X8/5W_C3 0 X8/5W_C2 0 X875W_C1 0 X875W_C0 0 X875W_E5 0 X875W_SF 0 X875W_P 0 X875W_U 0					
EIP	→• OO4ACAEE	FF DO	call eax		eax: SHCh Y	Default (stdcall)	▼ 5 ¢ Unlock				
eax= <shell:< th=""><th>AEF lockbit.exe</th><th>V> (7438A9CO) :\$ACAEF #ABEEF</th><th></th><th></th><th></th><th>2: [esp+4] 0000000 3: [esp+8] 0000000 4: [esp+C] 0000000 5: [esp+10] 043F0000</th><th></th></shell:<>	AEF lockbit.exe	V> (7438A9CO) :\$ACAEF #ABEEF				2: [esp+4] 0000000 3: [esp+8] 0000000 4: [esp+C] 0000000 5: [esp+10] 043F0000					
Dump 1	Ump 2	Dump 3 🗰 Dump 4	💷 Dump 5 🛛 👹 Watch 1 🛛	x=l Locals 🧳 Struct	0019F350 0800 0019F354 0000	00000					
Address He	ex		ASCII		↑ 0019F358 0000 0019F35C 0000	00000					

Figure 81

A new thread is created by the malware using CreateThread:

312	 0048A730 0048A73E 0048A740 0048A743 0048A743 0048A746 0048A748 0048A748 	51 6A 00 FF 75 E8 FF 75 E4 6A 00 6A 00 FF D0		push ecx push 0 push dword p push dword p push 0 push 0 call eax	tr ss: ebp tr ss: ebp	-18 -10	eax:Cre		175tatusWord 0000_ 175w_B 0 x875w_C3 0 x875w_C2 0 175w_S4 0 x875w_C4 0 x875w_E5 0 175w_S5 0 x875w_P 0 x875w_U 0 150w_S5 0 x875w_P 0 x875w_U 0 100000000000000000000000000000000000
eax= <kernel< th=""><th><pre>I32.CreateThread> A74A lockbit.exe::</pre></th><th>(76A745B0) \$8A74A #8984A</th><th></th><th></th><th></th><th></th><th></th><th>> 1: 2: 3: 4: 5:</th><th>(esp) 00000000 (esp+4) 00000000</th></kernel<>	<pre>I32.CreateThread> A74A lockbit.exe::</pre>	(76A745B0) \$8A74A #8984A						> 1: 2: 3: 4: 5:	(esp) 00000000 (esp+4) 00000000
Dump 1	Ump 2 Ump 2	Cump 3 💭 Dump 4	Ump 5	🛞 Watch 1	[x=] Locals	2 Struct	 0019FA68 0019FA6C	0000000	0
Address He	ex			ASCII			0019FA70	0049706	0 lockbit.00497060
0019FAE0 5 0019FAF0 8	48 43 68 61 6E 0 FF 19 00 1E 1A	67 65 4E 6F 74 69 4C 00 90 FF 48 00	66 79 00 00 90 FF 48 00	SHChangeNot	ify ÿK.		0019FA78 0019FA78 0019FA7C	0000000 0019FAD	0

Figure 82

Intel and AMD CPUs implement a functionality called "AES-NI" (Advanced Encryption Standard New Instructions), which can be used for high-speed AES encryption processing. The binary uses the cpuid instruction in order to retrieve the CPU type of the machine and the vendor of the CPU:





Figure 83

Whether the CPU supports "AES-NI" the process sends the "[+] AES-NI enabled" message to the hidden window using SendMessageA.

The malicious process generates 16 random bytes by calling the BCryptGenRandom routine (0x2 = BCRYPT_USE_SYSTEM_PREFERRED_RNG):



Figure 84

The ransom note is also stored in an encrypted form as a stack string that will be decrypted using a custom algorithm:





Address	He	ĸ															ASCII
0019F7C3	4C	6F	63	68	42	69	74	20	32	2E	30	20	52	61	6E	73	LockBit 2.0 Rans
0019F7D3	6F	6D	77	61	72	65	0D	0A	OD	0A	59	6F	75	72	20	64	omwareYour d
0019F7E3	61	74	61	20	61	72	65	20	73	74	6F	6C	65	6E	20	61	ata are stolen a
0019F7F3	6E	64	20	65	6E	63	72	79	70	74	65	64	OD	0A	54	68	nd encryptedTh
0019F803	65	20	64	61	74	61	20	77	69	6C	6C	20	62	65	20	70	e data will be p
0019F813	75	62	6C	69	73	68	65	64	20	6F	6E	20	54	4F	52	20	ublished on TOR
0019F823	77	65	62	73	69	74	65	20	68	74	74	70	3A	2F	2F	6C	website http://l
0019F833	6F	63	68	62	69	74	61	70	74	36	76	78	35	37	74	33	ockbitapt6vx57t3
0019F843	65	65	71	6A	6F	66	77	67	63	67	6C	GD	75	74	72	33	eeqjofwgcg1mutr3
0019F853	61	33	35	6E	79	67	76	6F	6B	6A	61	35	75	75	63	63	a35nygvokja5uuco
0019F863	69	70	34	79	68	79	64	2E	6F	6E	69	6F	6E	20	61	6E	ip4ykyd.onion ar
0019F873	64	20	68	74	74	70	73	3A	2F	2F	62	69	67	62	6C	6F	d https://bigblo
0019F883	67	2E	61	74	20	69	66	20	79	6F	75	20	64	6F	20	6E	g.at if you do r
0019F893	6F	74	20	70	61	79	20	74	68	65	20	72	61	6E	73	6F	ot pay the ranso
0019F8A3	GD	OD	0A	59	6F	75	20	63	61	6E	20	63	6F	6E	74	61	mYou can conta
0019F8B3	63	74	20	75	73	20	61	6E	64	20	64	65	63	72	79	70	ct us and decryp
0019F8C3	74	20	6F	6E	65	20	66	69	6C	65	20	66	6F	72	20	66	t one file for f
0019F8D3	72	65	65	20	6F	6E	20	74	68	65	73	65	20	54	4F	52	ree on these TOR
0019F8E3	20	73	69	74	65	73	0D	0A	68	74	74	70	3A	2F	2F	6C	siteshttp://l
0019F8F3	6F	63	6B	62	69	74	73	75	70	34	79	65	7A	63	64	35	ockbitsup4yezcd5
00105003	65	00	00	25	75	10	00	6.2	70	22	7.4	63	70	37	00	77	ant/Funney 37 cu7los

The process creates a registry key called "HKCU\SOFTWARE\2ED873D4E5389C" (0x80000001 = **HKEY_CURRENT_USER**, 0xF003F = **KEY_ALL_ACCESS**):



Figure 87

LockBit is looking for two registry values called "Private" and "Public" under the registry key above, which don't exist at this time:

	00 push dword ptr ds: [4FBC90] FF lea eax, dword ptr ss: [ebp-336] push eax push 0 lea eax, dword ptr ss: [ebp-44] push eax push dword ptr ss: [ebp-36]	eax:"Pr1: X8/TM_43 (Empty) X8/TM_73 (Empty) eax:"Pr1: X87TM_63 (Empty) X87TM_73 (Empty) x87SM_80 x87SM_63 (D000 x87SM_80 x87SM_60 x87SM_62 0 x87SM_610 x87SM_60 x87SM_65 0 x87SM_510 x87SM_9 0 x87SM_65 0 x87SM_510 x87SM_9 0 x87SM_10 0
004A05A6 FF D1	call ecx	Default (stdcall)
ecx= <advap132.regqueryvalueexa> (73A5FA00) .text:004A05A6 lockbit.exe:\$A05A6 #9F9A6</advap132.regqueryvalueexa>		11 (25) 000038C "Private" 31 (25) 000000 41 (25) 00137788 51 (25)+10] 04540000
Ump 1 Ump 2 Ump 3 Ump 4	🗱 Dump 5 👹 Watch 1 🕅 🕬 Locals 🎾 Struct	0019F558 0000038C 0019F55C 0019FAAC "Private"
Address Hex 0019F768 03 00 00 00 70 00 00 00 3C 00 3E 4C 0019F7C8 69 74 20 32 2E 30 20 52 61 6E 73 6F	ASCII 6F 63 6B 42p<.>LOCKB 6D 77 61 72 1t 2.0 Ransomwar	O19F560 0000000 O019F564 0019F788 O019F568 06450000 O19F568 06450000 O19F56C O19F76C

Figure 88

	 004A0635 004A0636 004A0637 004A0632 004A0632 004A0643 004A0643 004A0643 	50 50 80 85 C8 FC FF FF 50 6A 00 80 45 80 50 FF 75 C4	push eax push esi lea eax,dword ptr ss: push eax push 0 lea eax,dword ptr ss: push eax push dword ptr ss: push dword ptr ss:	[ebp-338] [ebp-50] pp-30]	eax:&"C eax:&"C [ebp-SC eax:&"C	x87 x87 x87 x87 x87 x87	TW_6 3 (Empty) Statusword 0000 SW_8 0 x87SW_C SW_C1 0 x87SW_C SW_SF 0 x87SW_F	x87TW_7 3 (Empty) 3 0 x87SW_C2 0 0 x87SW_ES 0 0 x87SW_U 0
ecx= <advap1< th=""><th>→ 004A0547 < 32.RegQueryValu 647 lockbit.exe</th><th>FF D1 eEXA> (73A5FA00) :\$A0647 #9FA47</th><th>call ecx</th><th></th><th>ecx:Rec</th><th>Defat 2: 3: 4: 5:</th><th>lt (stdcall) esp+4] 0019FAA0 esp+8] 0000038C esp+8] 0000000 esp+C] 0019F788 esp+10] 0646000</th><th>▼ 5 ♥ Unlock &"0 setlinejoin\r\n" 0</th></advap1<>	→ 004A0547 < 32.RegQueryValu 647 lockbit.exe	FF D1 eEXA> (73A5FA00) :\$A0647 #9FA47	call ecx		ecx:Rec	Defat 2: 3: 4: 5:	lt (stdcall) esp+4] 0019FAA0 esp+8] 0000038C esp+8] 0000000 esp+C] 0019F788 esp+10] 0646000	▼ 5 ♥ Unlock &"0 setlinejoin\r\n" 0
Dump 1	Dump 2	Dump 3 👹 Dump 4 🕻	🗒 Dump 5 🛛 👹 Watch 1 🛛 🕸 🕸 I Locals	2 Struct	0019F558 0019F55C	0000038C 0019FAA0	&"O setlinejoir	1/r/n"
Address He 0019FAA0 50 0019FAB0 61	x 75 62 6C 69 63 74 65 00 8C 03 40 40 7C 80 8C	00 00 10 00 00 00 50 00 00 00 00 FE 7E 00	ASCII 72 69 76 FublicPriv 00 00 00 ate		0019F560 0019F564 0019F568 0019F568	00000000 0019F7B8 06460000 0019F7BC	-	

Figure 89

The malware sends the "[+] Generate session keys" message to the hidden window. It will compute a public ECC (Curve25519) key and a private ECC (Curve25519) key.

The file generates 32 random bytes via a function call to BcryptGenRandom:





The malicious process implements a Curve25519 wrapper in the sub_4300C0 function. Based on the above buffer, it generates a session ECC public key:

			<u>8</u>	1
	.text:004300EE	mov	al, [edi+1Fh]	
	.text:004300F1	lea	ecx, [esp+120h+var_A0]	
	.text:004300F8	and	byte ptr [edi], 0F8h	
	.text:004300FB	and	al, 3Fh	
	.text:004300FD	or	al, 40h	
	.text:004300FF	mov	edx, edi	
	.text:00430101	mov	[edi+1Fh], al	
	.text:00430104	call	sub_43CF20	
	.text:00430109	lea	eax, [esp+120h+var_78]	
	.text:00430110	push	eax	
	.text:00430111	lea	edx, [esp+124h+var_50]	
	.text:00430118	lea	ecx, [esp+124h+var_F0]	
	.text:0043011C	call	sub_42E790	
	.text:00430121	add	esp, 4	
	.text:00430124	lea	eax, [esp+120h+var_78]	
	.text:0043012B	lea	edx, [esp+120h+var_50]	
	.text:00430132	lea	ecx, [esp+120h+var_118]	
	.text:00430136	push	eax	
	.text:00430137	call	sub_42E830	
	.text:0043013C	add	esp, 4	
	.text:0043013F	lea	edx, [esp+120h+var_118]	
	.text:00430143	mov	ecx, edx	
	.text:00430145	call	sub_43C980	
	.text:0043014A	lea	eax, [esp+120h+var_118]	
	.text:0043014E	push	eax	
	.text:0043014F	lea	edx, [esp+124h+var_F0]	
	.text:00430153	lea	ecx, [esp+124h+var_C8]	
	.text:00430157	call	sub 42EA30	
	.text:0043015C	add	esp, 4	
	.text:0043015F	lea	edx, [esp+120h+var C8]	
	.text:00430163	mov	ecx, edi	
	.text:00430165	call	sub 43C830	
	.text:0043016A	рор	edi	
	.text:0043016B	xor	eax, eax	
	.text:0043016D	рор	esi	
	.text:0043016E	mov	esp, ebp	
	.text:00430170	рор	ebp	
	.text:00430171	retn		
	.text:00430171	sub 4300	OCO endp	
	.text:00430171	-		
Address He	x			ASCII
004F88A0 A5	27 53 28 E9 DO	F0 C9 2	4 B3 0B 74 66 F4 FC 48	*'S+éDðÉ\$".tfôuk
004F88B0 DA	5F 25 A8 37 D8	B DC D7 4	B 50 C5 86 71 DA D5 3E	J_% 700×KPA. q00>
004F88C0 17	24 89 15 1E 89 98 88 7A D7 59	D1 7F	C 7F 51 27 53 3A FF D1	. ZXYNO'S:VN
			a sector of the	and the second sec

Figure 91

The above operation of generating random bytes is repeated one more time:





The same Curve25519 wrapper is used again to transform the above buffer:

Address	He	ĸ															ASCII
0019F530	73	9C	00	80	3C	E1	E2	91	A9	AF	EF	DA	53	76	8D	11	s<áâ.@ ïÚSv
0019F540	OB	35	23	13	26	OA	51	17	43	1F	50	DC	CA	42	12	7D	.5#.&.Q.C.PÜÊB.}

Figure 93

The executable embedded an ECC public key that we call Master ECC public key (highlighted in figure 94). Based on the implementation of the Curve25519 algorithm, it is used to generate a shared secret (32-byte value):

	 0042E6E6 0042E6E7 	57			push edi push esi					×	875W_SF 0 x	875W_P 0 x875W	_U 0
EIP	→• 10042E6E8	FF 1	5 90 06 4F	00	call dword p	otr ds:[4F0	690]			> De	fault (stdcall)		👻 5 🗘 🗌 Unlock
dword ptr ([lockbit.004F06	90]=1ockb e:\$2E6E8	#2DAE8							12345	[esp] 00199 [esp+4] 001 [esp+8] 001 [esp+C] 001 [esp+10] 001	2C0 9F320 9F518 9F530 19F49C	
Ump 1	Dump 2	Dump 3	Dump 4	Dump 5	🛞 Watch 1	[x=] Locals	Struct		0019F218 0019F21C	0019F2	0		
Address He 0019F320 7 0019F330 00 0019F340 21 0019F350 55	ex 3 9C 00 80 3C E 8 35 23 13 26 0 E D8 73 D4 3F E 0 9F 6C CE D6 E	1 E2 91 A A 51 17 4 5 DD 38 3 B C5 85 0	A9 AF EF DA 3 1F 50 DC 9 9C 88 67 2 0C F7 F3	53 76 8D 11 CA 42 12 7D 72 8D C9 75 0E 91 D6 61	ASCII 5<à.@i .5#.&.Q.C.P .ØsÔ?àÝ89].1ÌŎĖĂ+	ÚSV ÚĚ8.} gr%Eu 00a		^	0019F220 0019F224 0019F228 0019F22C 0019F230	0019F5 0019F5 0019F4 000000 0042E6	L8 30 50 10 10 10 10 10 10 10 10	lockb1t.0042E61C	from lockbit.00400

Figure 94

The Master ECC public key is utilized to encrypt the session ECC private key computed above:



Figure 95

We have utilized the capa tool in order to confirm that the above function is used to encrypt data using Curve25519:

encrypt dat	a using Curve25519 (2 matches)
namespace	data-manipulation/encryption/elliptic-curve
author	dimiter.andonov@mandiant.com
scope	basic block
attack	Defense Evasion::Obfuscated Files or Information [T1027]
examples	0a0882b8da225406cc838991b5f67d11:0x4135f6, 0a0882b8da225406cc838991b5f67d11:0x416f51,
basic block	0 x42F89E in function 0x42F6E0
and:	
and:	
numbe	r: 0xF8 @ 0x42F8AD
mnemo	nic: and @ 0x42F8AB, 0x42F8AD
and:	
numbe	r: 0x3F @ 0x42F8AB
mnemo	nic: and @ 0x42F8AB, 0x42F8AD
and:	
numbe	r: 0x40 @ 0x42F8B0
mnemo	nic: or @ 0x42F8B0
basic block	0 0x4300EE in function 0x4300C0
and:	
and:	
numbe	r: 0xF8 @ 0x4300F8
mnemo	nic: and @ 0x4300F8, 0x4300FB
and:	
numbe	r: 0x3F @ 0x4300FB
mnemo	nic: and @ 0x4300F8, 0x4300FB
and:	
numbe	r: 0x40 @ 0x4300FD
mnemo	nic: or @ 0x4300FD



LockBit stores the encrypted session ECC private key in the "HKCU\Software\2ED873D4E5389C\Private" registry value:

ETP	00440A48 00440A48 00440A50 00440A53 00440A53 00440A55 00440A55 00440A58 00440A58 00440A58 00440A58	6A 70 FF 35 90 8C 4F 00 8D 45 CC 6A 03 6A 00 FF 75 C4 FF 01	push 70 push dword ptr ds: [4F8C90] lea eax, dword ptr ss: [ebp-34] push 3 push eax push dword ptr ss: [ebp-3C]	eax:"Priv	×8 ×8 ×8 ×8	//w_b 3 (EmpLy) X8//w_/ 3 (EmpLy) 75tatusword 0000 75w.8 0 X875w.C2 0 75w.C1 0 X875w.C2 0 X875w.E5 0 75w.C1 0 X875w_C0 0 X875w_E5 0 75w.SF 0 X875w_P 0 X875w_U 0
	<			>	Defa	ault (stdcall) 🔻 🛛 🖓 🛄 Unlock
ecx= <adva< td=""><td>AOASB lockbit.exe:</td><td>A> (73A60F60) \$A0A5B #9FE58</td><td></td><td></td><td>2: 3: 4: 5:</td><td>[esp+8] 0019FABC "Private" [esp+8] 00000000 [esp+C] 00000003 [esp+10] 06450000</td></adva<>	AOASB lockbit.exe:	A> (73A60F60) \$A0A5B #9FE58			2: 3: 4: 5:	[esp+8] 0019FABC "Private" [esp+8] 00000000 [esp+C] 00000003 [esp+10] 06450000
Dump 1	Dump 2	Dump 3 💭 Dump 4 💭 Dump	5 💮 Watch 1 🛛 🕬 Struct	0019F556 00 0019F55C 00	000380 19FAB0	"Private"
Address 06450000 06450020 06450030 06450040 06450050 06450050	Hex 24 6C F9 D0 D9 OA EC 6C C0 2A E3 39 48 9C B6 27 E5 18 5B A2 E5 64 C0 15 DD 01 71 CA 95 43 E1 D0 87 54 A0 63 8F 34 A9 A1 C9 8A	EE E1 CF 92 65 A0 32 AS C8 EE A8 0C 6F 81 29 CC 7F 71 26 D5 A8 A8 F2 63 5A 8D AC 48 18 2F 27 58 6F 6A ED 5A 31 C2 FB D1 58 7C 9B 52 0A A8 2F 41 58 55 24 11 91 08 6A 31 82 15 3A 38 <u>41 55 E8</u>	ASCII 97 \$100.1 åt.e 2¥E. \$111 År å910.1 x.q. \$100.1 åt.e 22* \$100.0 gc.e.* \$31 År å910.1 x.q. \$100.1 gc.e.* \$100.0 gc.e.* \$34 År åqda.k.,''ojiz. \$100 Y.q.e.* \$100 Y.q.e.* \$19 Åb.t. c. /AXUS \$21* \$100 Y.q.e.*	O0197560 00 00197568 06 00197568 06 00197570 00 00197570 00 00197577 00 00197577 00 00197577 00	000000 000003 450000 3F0000 223000 223000 770050 45005	Tarbit oneones

Figure 97

LockBit stores the session ECC public key in the "HKCU\Software\2ED873D4E5389C\Public" registry value:

	OO4A0ACE OO4A0AD0 OO4A0AD0 OO4A0AD5 OO4A0AD7 OO4A0AD7 OO4A0AD9 OO4A0AD0 OO4A0AD0	6A 20 68 A0 88 4F 00 6A 03 6A 00 8D 45 A4 50 55 75 C4	push 20 push lockbit.4F88A0 push 3 push 0 lea eax,dword ptr ss:[ebp-5C] push eax		[ebp-5C] eax:&"0:	X8/IW_6 3 (LMDTY) X8/IW_/ 3 (LMDTY) X875XLa0000 X875W_8 0 X875W_C3 0 X875W_C2 0 X875W_8 0 X875W_C5 0 X875W_C5 0 X875W_5 0 X875W_C 0 X875W_U 0
EIP	>• 004A0AE0 <	FF D1	call ecx		ecx:RegS *	Default (stdcall)
ecx= <advap< th=""><th>0132.RegSetValueEx</th><th>(A> (73A60F60) \$A0AE0 #9FEE0</th><th></th><th></th><th></th><th>2: [esp+4] 0019FA94 &"0 setlinejoin\r\n" 3: [esp+8] 00000000 4: [esp+0] 00000003 5: [esp+10] 004F88A0 lockbit.004F88A0</th></advap<>	0132.RegSetValueEx	(A> (73A60F60) \$A0AE0 #9FEE0				2: [esp+4] 0019FA94 &"0 setlinejoin\r\n" 3: [esp+8] 00000000 4: [esp+0] 00000003 5: [esp+10] 004F88A0 lockbit.004F88A0
Ump 1	Dump 2	Dump 3 👹 Dump 4 👹 D	ump 5 🛛 🧐 Watch 1 🛛 🕸 🖉 Struct	00	19F558 00000 19F55C 00190	038C FA94 &"O setlinejoin\r\n"
Address 004F88A0 004F88B0 0	Hex AS 27 53 28 E9 D0 DA 5F 25 A8 37 D8	F0 C9 24 B3 08 74 66 F4 DC D7 48 50 C5 86 71 DA	ASCII FC 48 ¥`S+600£\$*.tfôux D5 3E 0_% 700×KPÅ.q00>	^ 00 00 00	19F560 0000 19F564 0000 19F568 004F 19F56C 0000	0000 0003 88A0 lockbit.004F88A0 0020

Figure 98

Figure 99 reveals both registry values with their content:

Y .	Software	^ Name	Type	Data
2	10-Strike Software	ab (Default)	REG_SZ	(value not set)
	2ED873D4E5389C	# Private	REG_BINARY	24 6c f9 d0 d9 0a ee e1 cf 92 65 a0 32 a5 c8 97 ec 6c c0 2a e3 39 ee ab 0c 6f 81 29 cc 7f 71 51 4b 9c b6 27 e5
2	7-Zip	24 Public	REG_BINARY	a5 27 53 2b e9 d0 f0 c9 24 b3 0b 74 66 f4 fc 4b da 5f 25 a8 37 d8 dc d7 4b 50 c5 86 71 da d5 3e

Figure 99

The malware uses I/O completion ports to improve the encryption speed. It creates an I/O completion object by calling the NtCreateloCompletion API (0x1F0003 = **IO_COMPLETION_ALL_ACCESS**):

Dump 1	Dump 2	ump 3 Sta Dump 4	ala nomb n	and the second second		Martin Contract	00196304	00110000	5
	and a state of	and a star from 4	titl Dumo 5	Watch 1	Ixel Locals	Struct	0019F560	004E252	0 lockbit.004E2520
eax= <ntdll.n< th=""><th>004A0D03 004A0D05 004A0D05 004A0D05 004A000F 004A001F 004A001F c<cccccccccccccccccccccccccccccccc< th=""><th>64 00 68 03 00 1F 00 68 20 25 4E 00 A3 98 88 4F 00 E8 E7 46 F7 FF FF D0 Cion> (7704F1E0) SA0D19 #A0119</th><th></th><th>oush 0 oush 1F0003 push lockbit nov dword pt call lockbit call eax</th><th>.4E2520 r ds:[4F88 .415400</th><th>98],eax</th><th>eax:NtC eax:NtC</th><th>x8 x8 x8 x8 x8 x8 x8 x8 x8 x8 x8 x8 x8 x</th><th>75Tatusword 0000 75W_B 0 x87SW_C2 0 x87SW_C2 0 75W_C1 0 x87SW_C0 0 x87SW_E5 0 75W_S5 0 x87SW_P 0 x87SW_D 0 sult(stdcall) ▼ 5 0 Unlock [esp1 004E2520 lockbit.004E2520 [esp+1] 0004E003 [esp+1] 004E0000 [esp+1] 00450000</th></cccccccccccccccccccccccccccccccc<></th></ntdll.n<>	004A0D03 004A0D05 004A0D05 004A0D05 004A000F 004A001F 004A001F c <cccccccccccccccccccccccccccccccc< th=""><th>64 00 68 03 00 1F 00 68 20 25 4E 00 A3 98 88 4F 00 E8 E7 46 F7 FF FF D0 Cion> (7704F1E0) SA0D19 #A0119</th><th></th><th>oush 0 oush 1F0003 push lockbit nov dword pt call lockbit call eax</th><th>.4E2520 r ds:[4F88 .415400</th><th>98],eax</th><th>eax:NtC eax:NtC</th><th>x8 x8 x8 x8 x8 x8 x8 x8 x8 x8 x8 x8 x8 x</th><th>75Tatusword 0000 75W_B 0 x87SW_C2 0 x87SW_C2 0 75W_C1 0 x87SW_C0 0 x87SW_E5 0 75W_S5 0 x87SW_P 0 x87SW_D 0 sult(stdcall) ▼ 5 0 Unlock [esp1 004E2520 lockbit.004E2520 [esp+1] 0004E003 [esp+1] 004E0000 [esp+1] 00450000</th></cccccccccccccccccccccccccccccccc<>	64 00 68 03 00 1F 00 68 20 25 4E 00 A3 98 88 4F 00 E8 E7 46 F7 FF FF D0 Cion> (7704F1E0) SA0D19 #A0119		oush 0 oush 1F0003 push lockbit nov dword pt call lockbit call eax	.4E2520 r ds:[4F88 .415400	98],eax	eax:NtC eax:NtC	x8 x8 x8 x8 x8 x8 x8 x8 x8 x8 x8 x8 x8 x	75Tatusword 0000 75W_B 0 x87SW_C2 0 x87SW_C2 0 75W_C1 0 x87SW_C0 0 x87SW_E5 0 75W_S5 0 x87SW_P 0 x87SW_D 0 sult(stdcall) ▼ 5 0 Unlock [esp1 004E2520 lockbit.004E2520 [esp+1] 0004E003 [esp+1] 004E0000 [esp+1] 00450000

Figure 100

The binary creates 2 (# of processors/cores) that will handle the files encryption:



EIP	● 0048A73D ● 0048A73E ● 0048A740 ● 0048A743 ● 0048A743 ● 0048A745 ● 0048A746 ● 0048A748	51 6A 00 FF 75 E8 FF 75 E4 6A 00 6A 00 FF D0	push ecx push o push dword g push dword g push o call eax	tr ss:[ebp tr ss:[ebp	9-18 9-10	iez	IXICITEA Y	x8 x8 x8	175 tatusword 0000 175 w_B 0 x875 w_C3 0 x875 w_C2 0 175 w_E 0 x875 w_C3 0 x875 w_E5 0 175 w_SF 0 x875 w_P 0 x875 w_U 0 fault (stdcall) ▼ 5 0 Unicol
eax= <kernel< th=""><th>132.CreateThreads</th><th>(76A745B0) \$8A74A #8984A</th><th></th><th></th><th></th><th></th><th></th><th>1: 2: 3: 4: 5:</th><th>[esp] 00000000 [esp+8] 00000000 [esp+8] 0049E730 lockbit.0049E730 [esp+c] 00000000 [esp+t0] 00000000</th></kernel<>	132.CreateThreads	(76A745B0) \$8A74A #8984A						1: 2: 3: 4: 5:	[esp] 00000000 [esp+8] 00000000 [esp+8] 0049E730 lockbit.0049E730 [esp+c] 00000000 [esp+t0] 00000000
Ump 1	Ump 2	Dump 3 🚦 Dump 4	🗰 Dump 5 🛛 🔞 Watch 1	x= Locals	2 Struct	0019	F4E0 000	0000	0
Address H 0019F548 0 0019F558 E	ex 2 00 00 00 <u>30 E7</u> C F1 04 77 1B 00	49 00 00 00 00 00 4A 00 20 25 4E 00	ASCII A6 00 00 000cI 03 00 1F 00 1ñ.w			0019 0019 0019 0019	F4E8 004 F4EC 000 F4F0 000 F4F4 001	9E73 0000 0000 9F54	0 lockbit.0049E730 0 8

The thread affinity mask is set to 1 via a function call to ZwSetInformationThread (0x4 = **ThreadAffinityMask**):

	 004A0D 004A0D 004A0D 004A0D 004A0D 004A0D 	87 6A 0 89 50 8A 6A 0 8C 52 8D E8 5	6A 04 push 4 50 push eax 64 push eax 64 push eax 65 st 47 F7 FF call lockbit.4154F0						eax:ZwS	e x8 x8 x8	875W_B 0 X875W_C3 0 X875W_C2 0 875W_B 0 X875W_C0 0 X875W_E5 0 875W_C1 0 X875W_C0 0 X875W_E5 0 875W_SF 0 X875W_P 0 X875W_U 0			0 0		
	→• 004A00 <	92 FF D	0		call eax				eax:Zw5	Def	ault (stdc) (Ile			▼ 5 \$ U	nlock
eax= <ntdll. .text:004A0</ntdll. 	ZwSetInforma D92 lockbit.	exe: \$A0D92	* (7704E830) #A0192							2: 3: 4: 5:	esp+4 esp+8 esp+0 esp+1] 000000] 0019F7] 000000 0] 043F0	04 9C 04 000			
Dump 1	Ump 2	Dump 3	💷 Dump 4	Dump 5	💮 Watch 1	[x=] Locals	2 Struct		0019F560 0019F564	0000039	4					
Address He	x		and the second		ASCII			A 2	0019F568	0019F79						
0019F79C 01	00 00 00 00	00 00 00	00 00 46 06	2E 00 25 00	F	96.		1 12	0185570	0.425000						

Figure 102

GetLogicalDrives is used to retrieve the available disk drives:

GIG →•	OO45BEBE FF DO	call eax	a.	eax:Get	ľ	Default (stdcall)	▼ 5 🗘 🗆 Unlod
eax= <kernel32.g< th=""><th>etLogicalDrives> (76470CDO) lockbit.exe:\$58E8E #5828E</th><th></th><th></th><th></th><th></th><th>1: [esp] 043P0000 2: [esp+4] 00223000 3: [esp+8] 00223000 4: [esp+C] 00223000 5: [esp+10] 00223000</th><th></th></kernel32.g<>	etLogicalDrives> (76470CDO) lockbit.exe:\$58E8E #5828E					1: [esp] 043P0000 2: [esp+4] 00223000 3: [esp+8] 00223000 4: [esp+C] 00223000 5: [esp+10] 00223000	
till Dama 1	own fillows fillows	and Printer P. Martin Strate and	(3) Charles	0019FA78	043F	0000	

Figure 103

The malicious binary determines the disk drive type using the GetDriveTypeW routine:

312	• 00458FF5 51 00539FF26 FF D0 <	call eax	ecx:L"Z: eax:GetDi	Default (stdcall)	- ▼ 5 € Unlock
eax= <kernel< td=""><td>32.GetDriveTypew> (76ACDFA0) FF6 lockbit.exe:\$58FF6 #583F6</td><td></td><td></td><td>1: [esp+4] 043F0000 3: [esp+8] 00223000 4: [esp+6] 00223000 5: [esp+10] 00223000</td><td></td></kernel<>	32.GetDriveTypew> (76ACDFA0) FF6 lockbit.exe:\$58FF6 #583F6			1: [esp+4] 043F0000 3: [esp+8] 00223000 4: [esp+6] 00223000 5: [esp+10] 00223000	
fill come t	and a second second		0019FA74 001	9FAC4 L"Z:"	

Figure 104

The process is looking for type 2 (**DRIVE_REMOVABLE**), type 3 (**DRIVE_FIXED**) and type 6 (**DRIVE_RAMDISK**) drives:





For each targeted drive, the malware creates a new thread that will traverse it and locate all files selected for encryption:

	 0048A73D 0048A73E 0048A740 0048A743 0048A743 0048A746 0048A748 	51 6A 00 FF 75 E8 FF 75 E4 6A 00 6A 00		ush ecx oush 0 oush dword p oush dword p oush 0 oush 0	tr ss: ebp tr ss: ebp	-18 -10	[ebp-18]] x87 x87 x87	Statusword 0000 Sw.B 0 x875w_C3 0 x875w_C2 0 Sw.C1 0 x875w_C0 0 x875w_E5 0 75w_SF 0 x875w_P 0 x875w_U 0
EIP		FF DO	6	all eax			eax:Crei	a Defa	ult (stdcail) 👻 💈 💭 Unlock
eax= <kerne< th=""><th>132.CreateThread></th><th>(76A745B0) \$BA74A #89B4A</th><th></th><th></th><th></th><th></th><th></th><th>2: 3: 4: 5:</th><th>[esp+4] 0000000 [esp+8] 0045C960 lockbit.0045C960 [esp+C] 06700000 L"Z:\\" [esp+10] 00000000</th></kerne<>	132.CreateThread>	(76A745B0) \$BA74A #89B4A						2: 3: 4: 5:	[esp+4] 0000000 [esp+8] 0045C960 lockbit.0045C960 [esp+C] 06700000 L"Z:\\" [esp+10] 00000000
Dump 1	Ump 2 Ump 2	Cump 3 Ully Dump 4	Ump 5	💮 Watch 1	[x=] Locals	Struct	 0019F9E8 0	00000000	
Address H	ex			ASCII	1		0019F9F0 0	00450960	lockb1t.0045C960
06700000 5 06700010 0	A 00 3A 00 5C 00 0 0 00 00 00 00 00 0	00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00	Z.:.\			0019F9F8 0 0019F9F8 0	000000000 0019FA50	



Thread activity – sub_45C960 function

The file compares the drive name with the tsclient (Terminal Server Client) share:

	0045C8F4 FF 75 08 0045C8F7 80 4C 24 50 0045C8F7	push dword ptr ss: [ebp+8] lea ecx,dword ptr ss: [esp+50]	[ebp+8]:1	x87SW_C1 0 x87SW_C0 0 x87SW_E5 0 x87SW_SF 0 x87SW_P 0 x87SW_U 0		
BIB	→• 0045CBEC FF D0	call eax	eax:lstr ~	Default (stdcall)		
eax= <kerne< th=""><th>132.1strcmpiw> (76A76AF0) :BFC lockbit.exe:\$5CBFC #5BFFC</th><th></th><th></th><th>1: [csp] 00397A8+ C (\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\</th></kerne<>	132.1strcmpiw> (76A76AF0) :BFC lockbit.exe:\$5CBFC #5BFFC			1: [csp] 00397A8+ C (\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\		
Dump 1	Dump 2 Dump 3 Dump 4 Dump 4 00450085 FF 75 08 008 224 E8 00 00 00 00450025 51 008 51 008	ump 5 🛞 Watch 1 [x=]Locals 🌮 Struct push dword ptr ss:[esp+8] lea ecx, dword ptr ss:[esp+83] push ecx	0599FA60 069 0699FA64 068 (ebp+8]:1 ecx:L"M1	FA84 L"\\\tsclient" 0000 L"2"\" x875W_C1 0 x875W_C0 0 x875W_E5 0 x875W_SF 0 x875W_P 0 x875W_U 0		
	< PP 00	Call Cax	>	Default (stdcall)		
eax= <kernel< td=""><td>132.1strcmp1₩> (76A76AF0) 00C3 lockbit.exe:\$SD0C3 #SC4C3</td><td></td><td></td><td>2: esp+10 06850000 L"2:\\" 3: esp+10 06850000 L"2:\\"</td></kernel<>	132.1strcmp1₩> (76A76AF0) 00C3 lockbit.exe:\$SD0C3 #SC4C3			2: esp+10 06850000 L"2:\\" 3: esp+10 06850000 L"2:\\"		
Dump 1	💭 Dump 2 💭 Dump 3 💭 Dump 4 🕬 0	ump 5 👹 Watch 1 🛛 🕸 🖉 Struct	0699FA60 0699 0699FA64 0685	DFB4C L"Microsoft Terminal Services" 00000 L"2:\\"		

Figure 107

The CreateFileW function is utilized to create a file called "2ED873D4.lock" (0xC0000000 = GENERIC_READ | GENERIC_WRITE, 0x1 = CREATE_NEW, 0x04000100 = FILE_FLAG_DELETE_ON_CLOSE | FILE_ATTRIBUTE_TEMPORARY):



	 0045DSD9 0045D8D8 0045D8D8 0045D8E0 0045D8E4 0045D8E4 0045D8E6 0045D8E8 0045D8F2 	6A 00 68 00 01 00 04 6A 01 6A 00 6A 00 6S 00 00 00 C0 8D 8C 24 20 03 00 00 51	push 0 push 4000100 push 1 push 0 push C0000000 lea ecx,dword ptr ss:[esp+3 push ecx	320]		As (n* 3 (Empty) As (n* 3 (Empty) x87TW_6 3 (Empty) x87TW_7 3 (Empty) x875tutsword 0000 x875W_8 0 x875W_C3 0 x875W_C2 0 x875W_5 0 x875W_C0 0 x875W_C5 0 x875W_5 0 x875W_0 0 x875W_U 0	
EIP	→• 0045D8F3	FF DO	call eax		eax:Crea	Default (stdcall) 🔹 💈	Unlock
eax= <kerne< td=""><td>08F3 lockbit.exe:</td><td>(76ACDDEO) SSD8F3 #SCCF3</td><td></td><td></td><td></td><td>1: [esp+4] C000000 3: [esp+4] C000000 4: [esp+4] C000000 5: [esp+10] 00000001</td><td></td></kerne<>	08F3 lockbit.exe:	(76ACDDEO) SSD8F3 #SCCF3				1: [esp+4] C000000 3: [esp+4] C000000 4: [esp+4] C000000 5: [esp+10] 00000001	
Dump 1	Ump 2 Ump 2	ump 3 👹 Dump 4 👹 Du	np 5 💮 Watch 1 🛛 🕅 🖉	Struct	0699FA4C 0699 0699FA50 C000	9FD70 00000	
Address H 0699FD70 5 0699FD80 3 0699FD90 6	ex A 00 3A 00 5C 00 1 7 00 33 00 44 00 5 8 00 00 00 00 00 00	5C 00 <u>32 00 45 00</u> 44 00 34 00 2E 00 6C 00 6F 00 30 00 00 00 00 00 00 00	ASCII 8 00 Z.:.\.\.2.E.D.8. 3 00 7.3.D.41.0.C. 0 00 k	^	0699FA54 0000 0699FA58 0000 0699FA5C 0000 0699FA60 0400 0699FA64 0000	00000 00000 00001 00100 00000	

SHEmptyRecycleBinW is used to empty the Recycle Bin on the drive (0x7 = **SHERB_NOCONFIRMATION** | **SHERB_NOPROGRESSUI** | **SHERB_NOSOUND**):

	 00450/ 00450/ 00450/ 	AE7 6A 6 AE9 FF 5	E7 6A 07 push 7 59 FF 75 08 push dword ptr ss:[ebp+6] 6A 00 push dword ptr ss:[ebp+6]		[ebp+8]:1	x87SW_C1 0 x87SW_C0 0 x87SW_ES 0 x87SW_SF 0 x87SW_P 0 x87SW_U 0				
	→• 0045D <	SEE FF	00		call eax			eax:SHEm ~	Default (stdcall)	▼ 5 🗘 🗌 Unlod
eax= <shell:< th=""><th>32.SHEmptyRe</th><th>cycleBinW></th><th>(744E6C40) #SCEEE</th><th></th><th></th><th></th><th></th><th></th><th>2: [esp+4] 06850 3: [esp+8] 00000 4: [esp+6] 00450 5: [esp+10] 00450</th><th>0000 L"Z:\\" 0007 :960 lockbit.0045C960 (C960 lockbit.0045C960</th></shell:<>	32.SHEmptyRe	cycleBinW>	(744E6C40) #SCEEE						2: [esp+4] 06850 3: [esp+8] 00000 4: [esp+6] 00450 5: [esp+10] 00450	0000 L"Z:\\" 0007 :960 lockbit.0045C960 (C960 lockbit.0045C960
Dump 1	Dump 2	🗱 Dump 3	Dump 4	📖 Dump 5	🛞 Watch 1	(x=) Locals	Struct	0699FASC 0000 0699FA60 0685	00000 50000 L"Z:\\"	
Address IN	av				LASCIT			0699FA64 0000	00007	

Figure 109

The executable retrieves information about the total amount of space and the total amount of free space on the drive by calling the GetDiskFreeSpaceW and GetDiskFreeSpaceExW APIs:

	 0045DC58 0045DC5C 0045DC63 0045DC64 0045DC68 0045DC69 0045DC70 0045DC71 	51 80 8C 24 6C 01 00 00 51 80 4C 24 38 51 80 8C 24 78 01 00 00 51 FF 75 08	<pre>push ecx hea ecx, dword ptr ss:[esp+16C] bea ecx, dword ptr ss:[esp+30] push ecx hea ecx, dword ptr ss:[esp+30] push ecx push decx push decx</pre>	[ebo+8]:(Av:Im3 (Empty) Av:Im3 (Empty) x87TM_6 3 (Empty) x87TM_7 3 (Empty) x87Statusword 0000 x87SM_B 0 x87SM_C2 0 x87SM_C2 0 x87SM_E5 0 x87SM_C0 0 x87SM_E5 0 x87SM_E5 0 x87SM_P 0 x87SM_U 0
EIP	> 00450C74	FF DO	call eax	eax:GetD Y	Default (stdcall)
eax= <kernel< th=""><th>132.GetDiskFreeSp 0C74 lockbit.exe:</th><th>acew> (76ACDF80) \$SDC74 #SD074</th><th></th><th></th><th>1: [esp] 06850000 L'2:\\" 2: esp+4) 06995B04 3: [esp+4] 06995B04 4: [esp+C] 0699F8D0 5: [esp+10] 0699F8DC</th></kernel<>	132.GetDiskFreeSp 0C74 lockbit.exe:	acew> (76ACDF80) \$SDC74 #SD074			1: [esp] 06850000 L'2:\\" 2: esp+4) 06995B04 3: [esp+4] 06995B04 4: [esp+C] 0699F8D0 5: [esp+10] 0699F8DC
Dump 1	Dump 2	Dump 3 🗱 Dump 4 🗱 Dump	5 👹 Watch 1 🛛 🕸 🖉 Struct	0699FA54 0685 0699FA58 0699	0000 L"Z:\\" FBD4
Address He 0699FBD4 00	ex 0 00 00 00 00 00 00	00 00 00 00 00 00 00 00	ASCII	0699FA5C 0699 0699FA60 0699 0699FA64 0699	FA98 F8D0 F8CC

Figure 110

	 00450DDC 00450DDD 00450DE1 00450DE2 00450DE9 00450DEA 	51 8D 4C 24 44 51 8D 8C 24 78 01 00 00 51 FF 75 08	<pre>push ecx lea ecx,dword ptr ss:[esp+44] push ecx lea ecx,dword ptr ss:[esp+178] push ecx push ecx push ecx</pre>	[ebp+6]:	x875tatusword 0000 x875w_B 0 x875w_C3 0 x875w_C2 0 x875w_C10 x875w_C0 0 x875w_E5 0 x875w_SF 0 x875w_P 0 x875w_U 0		
EIP	>• 004500ED <	FF DO	call eax	eax:Get0 ~	Default (stdcall)		
eax= <kerne< th=""><th>DED lockbit.exe:</th><th>SSDDED #SD1ED</th><th></th><th></th><th>1: (c5) 05395000 L 2:\\ 2: (c5)+4) 05395B88 3: [c5)+8] 05395A88 4: [c5)+2] 06395BC0 5: [c5)+10] 0045C360 lockbit.0045C360</th></kerne<>	DED lockbit.exe:	SSDDED #SD1ED			1: (c5) 05395000 L 2:\\ 2: (c5)+4) 05395B88 3: [c5)+8] 05395A88 4: [c5)+2] 06395BC0 5: [c5)+10] 0045C360 lockbit.0045C360		
Dump 1	Dump 2	Dump 3 👹 Dump 4 👹 I	Dump 5 👹 Watch 1 🛛 🕅 Locals 🦻 Struct	0699FA58 0685 0699FA5C 0695	0000 L"Z:\\" 9F8D8		
Address H	ex		ASCII	↑ 0699FA60 0699 0699FA64 0699	IFAA8 9FBC0		

Figure 111

The user interface language for the current thread is set to "English - United States":

0045DF34 0045DF39 <	68 09 04 00 00 FF D0	call eax	eax:SetTI ~	Default (stdcall)	▼ S 🗘 🗆 Unlock
<pre>eax=<kernel32.setthreaduilar .text:0045df39="" lockbit.exe:<="" pre=""></kernel32.setthreaduilar></pre>	1guage> (76A74E70) 15DF39 #5D339			2: [esp+4] 0045C960 lo 3: [esp+8] 0045C960 lo 4: [esp+6] 06850000 L" 5: [esp+10] 0699FA94	ockbit.0045C960 ockbit.0045C960 'Z:\\"
fill Canada dillo a dillo		n n 26. mart a futa t 🕲 na a	0699FA64 0000	00409	

Figure 112

The numeric values extracted above are converted into a string that represents the size values in bytes, kilobytes, megabytes, or gigabytes, depending on their size:





The drive name and the information regarding its size are sent to the hidden window via SendMessageW.

The FindFirstFileExW API is utilized to enumerate the drive:

ecx= <kerne< th=""><th>0045 294 0045 294 0045 294 0045 295 0045 295 0045</th><th>6A 00 6A 00 6A 00 8D 85 0 6A 00 6A 00 8D 85 0 70 6A 00 70 6A 00 70 8D 85 0 70 8D 85 0 8D 85 0</th><th>4 F7 FF FF 4 F2 FF FF DEA0)</th><th>push 0 push 0 push 0 push 0 push eax push 0 lea eax,dword push eax call ecx</th><th>ptr ss:[</th><th>ebp-8FC] ebp-D1C]</th><th> eax:L" eax:L" ecx:F1</th><th>z:' x8 x8 x8 x8 x8 x8 x8 x8 x8 x8 x8 x8 x8 x</th><th>Imposite 1 Lemps/j X87TV 75tatusword 0000 75w.87TW 75w.21 0 X87Sw.20 0 75w.55 0 X87Sw.20 0 75w.55 0 X87Sw.20 0 aut (stdcal) (stdcal) (stdcal) (stdp4) 6599E140 LT2:\\(stdp4) (stdp4) (stdp4) (stdp4)</th><th>x875W_C2 0 x875W_E5 0 x875W_U 0</th><th>Unlock</th></kerne<>	0045 294 0045 294 0045 294 0045 295 0045	6A 00 6A 00 6A 00 8D 85 0 6A 00 6A 00 8D 85 0 70 6A 00 70 6A 00 70 8D 85 0 70 8D 85 0 8D 85 0	4 F7 FF FF 4 F2 FF FF DEA0)	push 0 push 0 push 0 push 0 push eax push 0 lea eax,dword push eax call ecx	ptr ss:[ebp-8FC] ebp-D1C]	 eax:L" eax:L" ecx:F1	z:' x8 x8 x8 x8 x8 x8 x8 x8 x8 x8 x8 x8 x8 x	Imposite 1 Lemps/j X87TV 75tatusword 0000 75w.87TW 75w.21 0 X87Sw.20 0 75w.55 0 X87Sw.20 0 75w.55 0 X87Sw.20 0 aut (stdcal) (stdcal) (stdcal) (stdp4) 6599E140 LT2:\\(stdp4) (stdp4) (stdp4) (stdp4)	x875W_C2 0 x875W_E5 0 x875W_U 0	Unlock
							 	31	[esp+10] 00000000		
Dump 1	Ully Dump 2	Dump 3	Dump 4 Ump 5	💮 Watch 1	[x=] Locals	2 Struct	0699ED18 0699ED1C	0699ED40 00000000	0 L"2:*"		
Address	lex			ASCII	-		0699ED20	0699F160	0		
0699ED40	A 00 3A 00 5C 00 00 00 00 00	00 5C 00 2A 0 00 00 00 00 00 0	00 00 00 00 00 00 00 00 00 00 00 00 00	Z.:.\.\.°			0699ED28 0699ED2C	00000000	0		



The following directories will be skipped:

- system volume information
- windows photo viewer
- windows powershell
- internet explorer
- windows security
- windows defender
- microsoft shared
- application data
- windows journal
- \$recycle.bin
- \$windows~bt
- windows.old

The files enumeration is continued via a function call to FindNextFileW:



00461296 50 push eax 00461297 52 00461297							x875W_SF 0 x875W_P 0 x875W_U 0				
EIP	→• 00461 <	298 FF (01		call ecx			ecx:Findl ~	Def	fault (stdcall)	🔻 💈 🗘 Unioc
ecx= <kerne< th=""><th>132.FindNext</th><th>Filew⊳ (76A .exe:\$61298</th><th>#60698</th><th></th><th></th><th></th><th></th><th></th><th>1: 2: 3: 4: 5:</th><th>[esp] 02 [esp+4] [esp+8] [esp+C] [esp+10]</th><th>155C4A0 0699F160 8705F568 76AE7AFC "GetCurrentThreadId" 75CF0000 <user32.ordinal2648></user32.ordinal2648></th></kerne<>	132.FindNext	Filew⊳ (76A .exe:\$61298	#60698						1: 2: 3: 4: 5:	[esp] 02 [esp+4] [esp+8] [esp+C] [esp+10]	155C4A0 0699F160 8705F568 76AE7AFC "GetCurrentThreadId" 75CF0000 <user32.ordinal2648></user32.ordinal2648>
Sty Dump 1	Ump 2	Ump 3	Ump 4	UIII Dump 5	👹 Watch 1	[x=] Locals	2 Struct	0699ED28 0255 0699ED2C 0699	C44	0	

File extensions are extracted using the PathFindExtensionW routine:

EIP	→ 0046007E 56 0046007E FF D0	push esi call eax	est:L"2e eax:Path	Default (stdcall)	▼ 5 🗘 🗌 Unlod
eax= <sh1wa< td=""><td>pi.PathFindExtensionw> (75FC3E60) 007F lockbit.exe:\$6007F #5F47F</td><td></td><td></td><td>2: [esp+4] 87D5F568 3: [esp+8] 76AE7AFC "G 4: [esp+C] 75CF0000 <u 5: [esp+10] 00000000</u </td><td>etCurrentThreadId" ser32.0rdinal2648></td></sh1wa<>	pi.PathFindExtensionw> (75FC3E60) 007F lockbit.exe:\$6007F #5F47F			2: [esp+4] 87D5F568 3: [esp+8] 76AE7AFC "G 4: [esp+C] 75CF0000 <u 5: [esp+10] 00000000</u 	etCurrentThreadId" ser32.0rdinal2648>
	the states a the	aller a Manara ante a Gara	0699ED2C 0699	9F18C L"2ed873d4.lock"	

Figure 116

The binary is looking for a ".lockbit" file that would suggest the targeted file has already been encrypted:



Figure 117

ZwCreateFile is utilized to open the targeted file (0x10003 = **FILE_READ_DATA** | **FILE_WRITE_DATA** | **DELETE**, 0x80 = **FILE_ATTRIBUTE_NORMAL**, 0x1 = **FILE_OPEN**, 0x48 = **FILE_NON_DIRECTORY_FILE** | **FILE_NO_INTERMEDIATE_BUFFERING**):

00440EBD 6A 00 0 00440EBF 6A 00 0 00440EC1 CF 75 00 00 00 00 0040EC1 CF 75 00 00 00 0040EC1 CF 75 00 00 00 0040EC1 CF 75 00 00 0040EC1 CF 75 00 00 00440EC2 CF 40 01 00 00440ED0 6A 00 00 00440ED2 SF 80 00 00 00440ED2 SF 07 FD 00 00 00440ED2 SF 330 1 00440ED2 SF 07 C0 FF FF 00440ED2 SF 05 70 FD FF FF 1 00440EE5 SF 35 85 AC FD FF FF 1 0 00440EF4 GF 56 65 FD FF FF 13 00 00 FF 00440F74 CF 85 56 FD FF FF 18 00 00 FF 00440F74 GF 56 00 00 00 FF 00440F74 SF 57 C0 FF FF 40 00 00 FF 00440F76 SF 75 75 75 FF FF FF 40 00 00 FF 00440F76 SF 75 75 76 FF FF FF 00 00 00 FF 00440F76 SF 75 76 FF FF FF 00 00 00 FF 00	<pre>ush 0 ush dword ptr ds:[es1+48],1 ush dword ptr ds:[es1+48],1 ush 1 ush 1 ush 0 vsh 80 vecx.FFF0 vecx.FF0</pre>	GS 0028 FS 0053 ES 0028 DS 0028 C3 0023 ES 0028 eax:2wCr: x37r10 00000000000000000000000000000000000
004A0540 FF D0	all eax	eax:ZWCF) VDefault (stdcall)
eax= <ntdll.zwcreatefile> (7704ECD0) .text:004A0F40 lockbit.exe:\$A0F40 #A0340</ntdll.zwcreatefile>	~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~	11: [esp] 06A4030 2: [esp+4] 0001003 3: [esp+8] 0699EA88 4: [esp-(0699EACC 5: [esp+10] 0000000 201910 (esp=10]
Ump 1 Ump 2 Ump 3 Ump 4 Um Dump 5	🧐 Watch 1 🛛 🗐 Locals 🦉 Struct	69927F0 00010003 69927F4 0692688
Data Data <thdata< th=""> Data Data <thd< td=""><td>Aude Aude Aude Aude Aude<td>6992783 06995ACC 699578C 0000000 6995800 00000000 6995808 00000000 6995808 00000001 6995802 00000048 6995810 00000000</td></td></thd<></thdata<>	Aude Aude Aude <td>6992783 06995ACC 699578C 0000000 6995800 00000000 6995808 00000000 6995808 00000001 6995802 00000048 6995810 00000000</td>	6992783 06995ACC 699578C 0000000 6995800 00000000 6995808 00000000 6995808 00000001 6995802 00000048 6995810 00000000

Figure 118

The targeted file is bound to the I/O completion port created earlier via a function call to NtSetInformationFile (0x1E = **FileCompletionInformation**):





The NtQueryInformationFile routine is used to query file information (0x5 = **FileStandardInformation**):

ale and a	 004A10F8 004A10FD 004A10FF 004A1100 004A1106 004A1107 004A1107 	6A 05 6A 18 50 8D 85 48 FD FF F 50 FF 33 E8 B2 22 F7 FF	F lea eax, dword push seax push eax push eax push dword p call lockbit	1 ptr ss:[ebp-288] :r ds:[ebx] 4133C0	eax:NTQu eax:NTQu eax:NTQu	×87 ×87 ×87 ×87	Statusword 0000 Sw_B 0 x87Sw_C3 0 Sw_C1 0 x87Sw_C0 0 Sw_SF 0 x87Sw_P 0	x87SW_C2 0 x87SW_ES 0 x87SW_U 0	
EIR	> 004A110E	FF DO	call eax		eax:NTQu	Defa	ult (stdcall)	•	5 🗘 🗌 Unlock
eax= <ntdll.< th=""><th>NtQueryInformat</th><th>ionFile> (7704E870) :\$A110E #A050E</th><th></th><th></th><th></th><th>1: 2: 3: 4: 5:</th><th>[esp] 000003A8 [esp+4] 0699EA68 [esp+8] 0699EA38 [esp+C] 00000018 [esp+10] 00000005</th><th></th><th></th></ntdll.<>	NtQueryInformat	ionFile> (7704E870) :\$A110E #A050E				1: 2: 3: 4: 5:	[esp] 000003A8 [esp+4] 0699EA68 [esp+8] 0699EA38 [esp+C] 00000018 [esp+10] 00000005		
Dump 1	Dump 2	Dump 3 👹 Dump 4	🕮 Dump 5 🛛 🕙 Watch 1	Ix=I Locals 🖉 Struct	0699E804 0 0699E808 0	00003A8			
Address He 0699EA38 00	ex 00 00 00 00 00 00	00 00 00 00 00 00	ASCII		0699E80C 0 0699E810 0 0699E814 0	699EA38 0000018 0000005			

Figure 120

NtSetInformationFile is utilized to set end-of-file information for the file (0x14 = **FileEndOfFileInformation**):

	and the second s		Lange to the second		
	004A113F 004A11A1	6A 14 13 C2 83 E9 01 64 00 189 C2 23 45 DC 23 45 DC 23 45 DC 23 45 DC 23 55 DC 23 75 DC 23 75 DC 23 75 DC 23 75 DC 23 75 DC 23 75 DC 25 75 DC 26 75 PF FF 50 80 85 40 PD FF FF 50 80 85 40 PD FF FF 50 80 95 70 PF FF 50 50 50 PF FF 50 50 PF FF 50 50 PF FF	<pre>push 14 adc eax,edx sub ecx,1 push 8 and ecx,edx ptr ss:[ebp-34] and ecx,edx ptr ss:[ebp-34] and ecx,edx ptr ss:[ebp-34] and edx,edx is[es1+44],eax adc edx,edx and edx,edx is[es1+44],eax adc edx,edx is[ebp-24] and edt,dword ptr ss:[ebp-24] add edt,dword ptr ss:[ebp-24] add edt,eax lea eax,edword ptr ss:[ebp-268],ed1 adc edx,eax lea eax,edword ptr ss:[ebp-268] push eax push eax push eax push eax</pre>	eax: NtSe eax: NtSe eax: NtSe eax: NtSe eax: NtSe eax: NtSe eax: NtSe eax: NtSe eax: NtSe	 x87r0 00000000000000000000000000000000000
EIP	>= 004A11E9	FF DO	call eax	eax:NtSe Y	Dafa dt (stdeall)
	<			>	Lecal 0000234
eax= <ntdll< td=""><td>.NtSetInformation</td><td>File> (7704E9F0) \$A11E9 #A05E9</td><td></td><td></td><td>2: esp+4) 0699EA60 3: esp+8) 0699EA86 4: esp+C) 00000008 5: esp+10] 00000008</td></ntdll<>	.NtSetInformation	File> (7704E9F0) \$A11E9 #A05E9			2: esp+4) 0699EA60 3: esp+8) 0699EA86 4: esp+C) 00000008 5: esp+10] 00000008
Ump 1	Ump 2	Dump 3 👹 Dump 4 👹 Dur	np 5 👹 Watch 1 🛛 🕸 Locals 🎾 Struct	0699E804 0000 0699E808 0699	003A8 0EA60
Address H	iex		ASCII	▲ 0699E80C 0699	PEABS
0699EAB8 0	0 2A 00 00 00 00	00 00 00 00 00 00 SC 03 0	0 00 10	06995810 0000	00014



The following extensions list has been found:

- ".rar" ".zip" ".ckp" ".db3" ".dbf" ".dbc" ".dbs" ".dbt" ".dbv" ".frm" ".mdf"
- ".mrg" ".mwb" ".myd" ".ndf" ".qry" ".sdb" ".sdf" ".sql" ".tmd" ".wdb" ".bz2"
- ".tgz" ".lzo" ".db" ".7z" ".sqlite" ".accdb" ".sqlite3" ".sqlitedb" ".db-shm"
- ".db-wal" ".dacpac" ".zipx" ".lzma"

LockBit only encrypts the first 4KB of the file. It uses the ZwReadFile API in order to read 0x1000 (4096) bytes:



	push 0 eax, dword ptr ds:[ed1-C] eax push dword ptr ds:[ed1-26] eax	:ZWR6. x87Tagword FFFF :ZWR6. x87Tw_0 3 (Empty) x87Tw_1 3 (Empty) x87Tw_2 3 (Empty) x87Tw_4 3 (Empty) x87Tw_5 3 (Empty) x87Tw_4 4 (Empty) x87Tw_5 3 (Empty) x
eax= <ntd11.zwreadfile> (7704E7C0) .text:004A20D5 lockbit.exe:\$A20D5 #A14D5</ntd11.zwreadfile>		> 116 (90000) 21 (esp) 000003A8 21 (esp+4) 00000000 31 (esp+4) 00000000 41 (esp+C) 06A40050 51 (esp+10) 06A40054
🚛 Dump 1 🚛 Dump 2 🚛 Dump 3 🚛 Dump 4 🚛 Dump 5		774 000003A8 778 00000000
Address Hex 064800000 000 00	06995 0000000000000000000000000000000000	7FC 10000000 800 106A0030 800 106A0030 800 106A0030 800 106A0030 810 106A0070 814 10000000

The GetFileAttributesW function is used to get file system attributes for the ransom note called "Restore-My-Files.txt":

Image: Signature Image: Optimized state Signature Image: Signature </th <th>eax:Get# befault (stdcall)</th> <th>Unloci</th>	eax:Get# befault (stdcall)	Unloci
eax= <kernel32.getfileattributesw> (76ACDFE0) .text:004A2606 lockbit.exe:\$A2606 #A1AD6</kernel32.getfileattributesw>	1: (ESP) 0699628 L 2: \\\\KESDFEA 2: (ESP+4) 0699676A 3: (ESP+5) 0000000 4: (ESP+5) 0000000 4: (ESP+5) 0000000	xt
Mont Ma a Ma a Ma a Ma a Mutte Mut (0.0.1	0699E814 0699E828 [L"Z:\\\Restore-My-Files.txt"	

Figure 123

The ransomware creates the ransom note via a call to ZwCreateFile (0x10003 = **FILE_READ_DATA** | **FILE_WRITE_DATA** | **DELETE**, 0x80 = **FILE_ATTRIBUTE_NORMAL**, 0x2 = **FILE_CREATE**, 0x40 = **FILE_NON_DIRECTORY_FILE**):

eax=cntd]]	004427A3 004427A3 004427A3 004427A9 004427A9 004427A9 004427A9 004427B5 0004427B5 000485 0000000000000000000000000000000	6A 00 6A 00 6A 40 6A 40 6B 80 00 00 00 6F 56 FF 50 00 6F 57 6F 50 77 85 50 PD FF FF 50 85 50 PD FF FF 50 00 6F 07 00 6F 07 00 6F 07 00 6F 07 00 6F 07 7 85 00 PF FF 50 00 6F 07 7 85 00 PF FF 50 00 77 85 50 PF FF 50 00 78 50 PF FF 50 00 77 85 50 PF FF 50 00 70 70 70 00 70 00 70 00 7	push 0 push 0 push 40 push 40 push 80 push 80	es1+48],1 d1] ax s:[es1+30] ss:[es1+30] ss:[es1+30] ebp-2x0],10 ebp-2x0],0 ebp-2x4],40 ebp-2x0],0 ebp-2x4],40 ebp-2x0,0 ebp-2x0,0 s:[ebp-278],xmm0	eax:2wCr· eax:2wCr· eax:2wCr· eax:2wCr· eax:2wCr· 40:'@' eax:2wCr	GS 0028 FS 0053 ES 0028 DS 0028 CS 0023 <u>SS</u> 0028 K37r0 000000000000 x87r1 0000000000000 x87r2 0000000000000 x87r4 0000000000000 x87r6 0000000000000 x87r6 0000000000000 x87r6 0000000000000 x87r5 00000000000000 x87r5 0 0000000000000 x87r5 0 0000000000000 x87r5 0 0000000000000 x87r5 100000000000000 x87r5 100000000000000 x87r5 1000000000000000 x87r5 1000000000000000 x87r5 1000000000000000 x87r5 1000000000000000 x87r5 10000000000000000 x87r5 1000000000000000 x87r5 10000000000000000 x87r5 1000000000000000 x87r5 10000000000000000 x87r5 100000000000000000 x87r5 100000000000000000 x87r5 1000000000000000000 x87r5 100000000000000000000000000 x87r5 0 00000000000000000000 x87r5 0 0000000000000000000000000000000000	00000000 STO Empty 0.00000000 00000000 STI Empty 0.000000000 00000000 STI Empty 0.000000000 00000000 STI Empty 0.000000000 00000000 STF Empty 0.000000000 00000000 STF Empty 0.000000000 x87TW_3 3 (Empty) x87TW_3 3 (Empty) x87TW_3 3 (Empty) x87TW_7 3 (Empty) x87TW_5 (Empty) x87TW_5 3 (Empty) x87TW_5 4 (Empty) x87TW_5 4 (Empty) x87TW_5 5 C unled
Ump 1	Dump 2	Dump 3 👹 Dump 4 👹	Dump 5 👹 Watch 1 🛛 🕬	cals 🦻 Struct	0699E7EC 064 0699E7F0 000	C0030	
Address He	ex		ASCII		↑ 0699E7F4 069 0699E7F8 069	9EA70 9EAA8	
06AD0000 00 06AD0020 00 06AD0020 00 06AD0030 00 06AD0040 00 06AD0050 00 06AD0050 00	0 00 </th <th>00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00</th> <th>00 00 00 00</th> <th></th> <th>0699E7FC 000 0699E800 000 0699E804 000 0699E806 000 0699E810 000 0699E810 000 0699E814 000</th> <th>000000 00080 000000 000002 000000 000000</th> <th></th>	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00		0699E7FC 000 0699E800 000 0699E804 000 0699E806 000 0699E810 000 0699E810 000 0699E814 000	000000 00080 000000 000002 000000 000000	

Figure 124

The ransom note is bound to the I/O completion port previously created via a function call to NtSetInformationFile (0x1E = **FileCompletionInformation**):







The note is populated using the ZwWriteFile routine:

	 00442ABB 00442ACB 00442AC1 00442AC2 00442AC2 00442AC2 00442AC3 00442AC3 00442AC3 00442AC3 00442AC3 00442AE3 00442AE4 00442AE3 00442AE3<	6A 00 C7 46 2C 00 00 00 00 S1 42 2C A1 78 8C 4F 00 S9 42 28 C7 02 03 00 00 00 C7 41 04 00 00 00 C7 41 04 00 00 00 FF 72 2C S0 42 04 S0 S0 S1 S4 00 S4 00 FF 33 E8 LC 28 F7 FF	<pre>push 0 mov dword ptr ds:[est+2C],0 push ecx mov dword ptr ds:[edx+2C],eax mov dword ptr ds:[edx+2B],eax mov dword ptr ds:[edx+2B],eax mov dword ptr ds:[edx+2B],0 push dword ptr ds:[edx+2C] push dword ptr ds:[edx+2C] push eax push dword ptr ds:[edx+4] push eax push dword ptr ds:[ebx] Cabl 0ccbst.415310</pre>	eax: Zwwr eax: Zwwr eax: Zwwr eax: Zwwr eax: Zwwr eax: Zwwr	X8/74 000000000000000000000000000000000000
ETP	>004A2AF4	FF DO	call eax	eax:Zwwr Y	Default (stdcall) 🔹 🔽 Unlock
eax= <ntdll .text:004A</ntdll 	.ZwwriteFile> (77 2AF4 lockbit.exe:	04E7E0) \$A2AF4 #A1EF4 Dump 3 👹 Dump 4 👹 Dump	15 👹 Watch 1 I∞ Locals 🎾 Struct	06998754 0000	1: [esp] 00000494 2: [esp+8] 00000000 3: [esp+8] 00000000 5: [esp-() 66420054 00494
Address + 06440000 4 06440010 6 06440020 6 06440030 6 06440040 6	Image: Non-State Image: Non-State<	74 20 32 2E 30 20 52 61 6E 00 0A 00 0A 59 6F 75 72 20 65 20 73 74 6F 6C 65 6E 20 73 74 65 64 00 0A 54 20 77 69 6C 6C 20 62 65 20 65 61 00 64 65 62 20	ASCII 73 LockBit 2.0 Rans 64 omware		00000 C0050 40000 00200 C0070 00000

Figure 126

The ".lock" file created earlier is deleted after the drive enumeration is complete:

	• 0045E60F 51 • 0045E610 FF D0 <	push ecx call eax	ecx:L"21' eax:Dele ~	Default (stdcall)	- - 5 🗘 🗋 Unlock
eax= <kernel< td=""><td>132.DeleteFilew> (76ACDE10) E610 lockbit.exe:\$5E610 #5DA10</td><td></td><td></td><td>2: [esp+4] 0045C960 3: [esp+8] 0045C960 4: [esp+8] 0045C960 4: [esp+C] 06850000 5: [esp+10] 0699FA94</td><td>lockbit.0045C960 lockbit.0045C960 L"Z:\\"</td></kernel<>	132.DeleteFilew> (76ACDE10) E610 lockbit.exe:\$5E610 #5DA10			2: [esp+4] 0045C960 3: [esp+8] 0045C960 4: [esp+8] 0045C960 4: [esp+C] 06850000 5: [esp+10] 0699FA94	lockbit.0045C960 lockbit.0045C960 L"Z:\\"
fill Dama 1	Minuna Minuna Minuna	and the second of the second s	0699FA64 0699	9FD70 L"Z: \\\\2ED873D4	.lock"

Figure 127

The content of the ransom note is displayed below:

11100	north my a more that
1	LockBit 2.0 Ransomware
2	
з	Your data are stolen and encrypted
4	The data will be published on TOR website http://lockbitapt6vx57t3eeqjofwgcglmutr3a35nygvokja5uuccip4ykyd.onion and https://bigblog.at if you do not pay the ransom
5	You can contact us and decrypt one file for free on these TOR sites
6	http://lockbitsup4yezcd5enk5unncx3zcy7kw6wllygmiyhvanjj352jayid.onion
2	http://lockbitsap2oaqhcun3syvbqt6n5nzt7fqosc6jdlmsfleu3ka4k2did.onion
8	OR
9	https://decoding.at
10	
11	Decryption ID: 2ED873D43FE5DD38A527532BE9D0F0C9

Figure 128

The main thread sends the "Scan done, waiting handles..." message to the hidden window.

Thread activity – sub_497060 function

The malware retrieves the locally unique identifier (LUID) for the SeDebugPrivilege privilege using the LookupPrivilegeValueA routine:

0048ECD1 51 0048ECD2 80 40 BF 0048ECD5 51 0048ECD5 51 0048ECD5 54 00					push ecx lea ecx,dwor push ecx push o	rd ptr ss:	ebp-41	ecx: "SeD	x87Sw_B 0 x87Sw_C3 0 x87Sw_C2 0 x87Sw_C1 0 x87Sw_C0 0 x87Sw_E5 0 x87Sw_SF 0 x87Sw_P 0 x87Sw_U 0					
EIP	→• 004BEG	DS FF	DO		call eax			eax:Look) ~	Defa	ult (stdcall)			• 5	🗘 🗌 Unlock
eax= <advapi< th=""><th>CD8 lockbit</th><th>ivilegevalu .exe:\$BECD8</th><th>eA> (73A5949 #BE0D8</th><th>90)</th><th></th><th></th><th></th><th></th><th>2: 3: 4: 5:</th><th>[esp+4] 08 [esp+8] 08 [esp+6] 00 [esp+10] 00</th><th>24F7FF "Se 24F820 0497060 100 00497060 100</th><th>DebugPrivil kbit.004970 ockbit.00497</th><th>ege" 60 060</th><th></th></advapi<>	CD8 lockbit	ivilegevalu .exe:\$BECD8	eA> (73A5949 #BE0D8	90)					2: 3: 4: 5:	[esp+4] 08 [esp+8] 08 [esp+6] 00 [esp+10] 00	24F7FF "Se 24F820 0497060 100 00497060 100	DebugPrivil kbit.004970 ockbit.00497	ege" 60 060	
Ump 1	Dump 2	Dump 3	Dump 4	Dump 5	👹 Watch 1	[x=] Locals	2 Struct	0824F7CC 0000 0824F7D0 082	00000 4F7FF	seDebug	Privilege"			
Address He	AY				ASCTT			0824F7D4 082	4F820					

Figure 129

The privileges of the access token are adjusted to include the SeDebugPrivilege privilege via a function call to ZwAdjustPrivilegesToken:



	0048EDEF 0048EDF1 0048EDF3 0048EDF5 0048EDF5 0048EDF8 0048EDF9 0048EDF8	6A 00 6A 00 6A 10 8D 4D D0 51 6A 00 6F 75 50	push push lea push push	0 10 tcx,dword ecx 0 dword p	d ptr ss:[ebp-30]			×87 ×87 ×87 ×87	Statusw SW_B 0 SW_C1 0 SW_SF 0	ord 0000 x875W_C3 x875W_C0 x875W_P	0 x875W_C 0 x875W_E 0 x875W_U	2 0 5 0 0	
eax= <ntd11.zwa< th=""><th>004BEDFE</th><th>FF D0 Foken> (7704EB90) EDFE #BE1FE</th><th>call</th><th>eax</th><th></th><th></th><th></th><th>eax:ZwAd ~</th><th>Defa 1: 2: 3: 4: 5:</th><th>ult (stdcall) [esp] 00 [esp+4] [esp+8] [esp+C] [esp+10]</th><th>00003A4 00000000 0824F810 00000010</th><th></th><th>• 5</th><th>🗘 🗌 Unlod</th></ntd11.zwa<>	004BEDFE	FF D0 Foken> (7704EB90) EDFE #BE1FE	call	eax				eax:ZwAd ~	Defa 1: 2: 3: 4: 5:	ult (stdcall) [esp] 00 [esp+4] [esp+8] [esp+C] [esp+10]	00003A4 00000000 0824F810 00000010		• 5	🗘 🗌 Unlod
Ump 1	Dump 2	np 3 👹 Dump 4	🖏 Dump 5 🛛 💮	Watch 1	[x=] Locals	2 Struct	0	824F7C0 0000 824F7C4 0000	003A4					
Address Hex 0824F810 01 00 0824F820 14 00	00 00 14 00 00 00 00 00 00 00	00 00 00 00 00 00 00 <u>21 8A E6 76</u>	ASC1		(B		^ 0000	824F7C8 0824 824F7CC 0000 824F7D0 0000 824F7D4 0000	4F810 00010 00000					

OpenSCManagerA is used to establish a connection to the service control manager and to open the service control manager database (0xF003F = **SC_MANAGER_ALL_ACCESS**):

10	00497 00497 00497 00497	4E6 68 4E8 6A	68 3F 00 0F 00 6A 00 6A 00	0 push F003F push 0 push 0					x875 x875	0	
EIP	→• 00497 <	IEF FF I	00		call eax			eax:Open ~	Defau	ilt (stdcall) 🔻	5 🗘 🗌 Unlod
eax= <advap< th=""><th>132.OpenSCMa 74EF lockbit</th><th>nagerA> (73 .exe:\$974EF</th><th>A61350) #968EF</th><th></th><th></th><th></th><th></th><th></th><th>2: 2: 2: 2: 2: 2: 2: 2: 2: 2: 2: 2: 2: 2</th><th>esp+4] 00000000 esp+8] 00000000 esp+8] 000F003F esp+C] 00497060 lockbit.00497060 esp+10] 00497060 lockbit.0049706</th><th>0</th></advap<>	132.OpenSCMa 74EF lockbit	nagerA> (73 .exe:\$974EF	A61350) #968EF						2: 2: 2: 2: 2: 2: 2: 2: 2: 2: 2: 2: 2: 2	esp+4] 00000000 esp+8] 00000000 esp+8] 000F003F esp+C] 00497060 lockbit.00497060 esp+10] 00497060 lockbit.0049706	0
Dump 1	Dump 2	Dump 3	Dump 4	Dump 5	🛞 Watch 1	Ix=I Locals	Struct	0824F83C 0000 0824F840 0000	00000		
Address W	AV				ACCTT	1		0824F844 000F	FOO3F		

Figure 131

А	targeted	service	is	opened	using	the	Opens	ServiceA	API	(0x2c	=
SC_	MANAGEF	R_MODIFY_	BOOT_C	ONFIG			S	C_MANA	SER_LO	СК	
SC_	MANAGEF	R_ENUMER	ATE_SE	RVICE):							
<u>aip</u>	00497580 00497582 00497585 00497586 00497586 00497586	64 2C FF 34 B1 53 FF D2	push push call	dword ptr ds:[ect ebx edx	x+esi=4]	_	[ecx+es1 edx:0pen ~	x875W_C1 0 x875W_ x875W_SF 0 x875W_ wefault (stdcall)	CO 0 X87SW_E P 0 X87SW_U	s 0 0 ▼ s ≎ □ Unio	ock
edx= <ad< th=""><th>dvapi32.OpenService</th><th>A> (73A740E0) (e:\$97586 #96986</th><th></th><th></th><th></th><th></th><th></th><th>1: [esp] 025931A8 1: [esp+4] 0560000 1: [esp+8] 000002 1: [esp+C] 0049706 1: [esp+10] 004970</th><th>0 "wrapper" C 0 lockbit.0049 60 lockbit.004</th><th>7060 97060</th><th></th></ad<>	dvapi32.OpenService	A> (73A740E0) (e:\$97586 #96986						1: [esp] 025931A8 1: [esp+4] 0560000 1: [esp+8] 000002 1: [esp+C] 0049706 1: [esp+10] 004970	0 "wrapper" C 0 lockbit.0049 60 lockbit.004	7060 97060	
Ul Dun	np 1 👹 Dump 2 👹	🖞 Dump 3 🛛 🗱 Dump 4	Dump 5	Watch 1 x= Locals	Struct		025931 0824F840 056000 0824F844 000000	LAS 000 "wrapper" 020			

Figure 132

QueryServiceStatusEx is used to extract the current status of the service:

004975C5 50 004975C5 6A 24 004975C6 6A 24 004975C8 8D 84 24 08 03 00 00 004975CF 50 004975D0 6A 00 004975D0 53 3	push eax push 24 lea eax,dword ptr ss:[esp+308] push eax push 0 push ebx		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
EIP 00497503 FF D1	call ecx	ecx:quer: *	Default (stdcall)
.text:004975D3 lockbit.exe:\$975D3 #969D3			2: [esp+4] 00000000 3: [esp+8] 0643FB48 4: [esp+C] 00000024 5: [esp+10] 0643FB3C
Ump 1 Ump 2 Ump 3 Ump 4 Ump 5	🛞 Watch 1 🛛 🕸 🖉 Struct	0643F834 0255 0643F838 0000	3018
Address Hex	ASCII	A 0643F83C 0643	F848
0643FB48 02 00 00 00 00 00 00 00 02 00 00 00 00	and a second sec	0643F844 0643	FB3C

Figure 133

The EnumDependentServicesA routine is utilized to retrieve the name and status of each service that depends on the targeted service (see figure 134). These services will be stopped as well (0x1 = **SERVICE_ACTIVE**):

	 0049A466 0049A467 0049A467 0049A46A 0049A46B 0049A46D 0049A46F 0049A46F 0049A46F 0049A471 	51 80 4D C8 51 6A 00 6A 00 6A 01 FF 75 F4	push ec lea ecx push o push 0 push 1 push 1	dword ptr ss:	[ebp-38]			x87 x87 x87 x87 x87	StatusWord 0000 SW_B 0 x87SW_C3 SW_C1 0 x87SW_C0 SW_SF 0 x87SW_P	0 x87SW_C2 0 x87SW_ES 0 x87SW_U	0 0 0
EIP	→• 0049A474	FF DO	call ea				eax:Enu	Defa	ult (stdcall)		• 5 C Unlock
eax= <advapi< td=""><td>32.EnumDependen 474 lockbit.exe</td><td>tServicesA> (73A83F :\$9A474 #99874</td><td>80)</td><td></td><td></td><td></td><td></td><td>1: 2: 3: 4: 5:</td><td>[esp] 02593018 [esp+4] 00000001 [esp+8] 00000000 [esp+C] 00000000 [esp+10] 0643F808</td><td></td><td></td></advapi<>	32.EnumDependen 474 lockbit.exe	tServicesA> (73A83F :\$9A474 #99874	80)					1: 2: 3: 4: 5:	[esp] 02593018 [esp+4] 00000001 [esp+8] 00000000 [esp+C] 00000000 [esp+10] 0643F808		
Dump 1	Dump 2	Dump 3 Dump 4	🕮 Dump 5 🛛 🥮 Wat	h 1 [x=] Locals	Struct	0	643F678 643F67C	02593018			
Address He	x		ASCII			A 0	643F680	00000000			
0049ACC0 88 0049ACD0 0F	45 F8 88 55 FC 87 04 50 88 04	88 48 1C 88 40 24 81 03 C3 E9 80 F7	03 CB 03 C3 .E0.UU. FF FF 88 45P	H@S.E.Å .Åé.+99.E		000	643F688 643F68C	0643F808 0643F800			





Every chosen service is stopped by calling the ControlService function (0x1 = **SERVICE_CONTROL_STOP**):

	 00497 00497 00497 00497 	97AC5 51 97AC6 6A 01 97AC8 53	push ecx push 1 push ebx						X875W_C1 0 X875W_C0 0 X875W_E5 0 X875W_SF 0 X875W_P 0 X875W_U 0	
EIP	→• 00497 <	AC9 FF I	DO		call eax			eax:Co	> (111)	Default (stdcall)
eax= <advap< th=""><th>132.Controls 7AC9 lockbit</th><th>ervice> (73</th><th>A73020) #96EC9</th><th></th><th></th><th></th><th></th><th></th><th></th><th>1: [esp] 02593018 2: [esp+4] 00000001 3: [esp+8] 0643FB48 4: [esp+C] 00497060 lockbit.00497060 5: [esp+10] 00497060 lockbit.00497060</th></advap<>	132.Controls 7AC9 lockbit	ervice> (73	A73020) #96EC9							1: [esp] 02593018 2: [esp+4] 00000001 3: [esp+8] 0643FB48 4: [esp+C] 00497060 lockbit.00497060 5: [esp+10] 00497060 lockbit.00497060
Ump 1	Dump 2	Dump 3	Dump 4	Dump 5	👹 Watch 1	[x=] Locals	Struct	0643F83C 0643F840	0259	3018 0001
Address U	da be				ACCTT	1		0643F844	0643	FB48

Figure 135

A confirmation message that the service was successfully stopped is sent to the hidden window:

	 004E0DD4 004E0DD5 004E0DD7 004E0DDC 	51 6A 01 68 01 FF 35	04 00 00 00 88 4F	00	push ecx push 1 push 401 push dword p	otr ds:[4F8	800]		_	x8 x8 x8	/SW_B 7SW_C1 (7SW_SF (x875W_C3 x875W_C0 x875W_P	0 x8/5W_0 0 x875W_0 0 x875W_0	5 0 0	
EIP	> 004E00E2	FF DO			call eax				eax: Send	Def	ult (stdcal)		• 5	Cunlock
eax= <user32< th=""><th>DE2 lockbit.ex</th><th>(75CFAB00 e:\$E0DE2 #</th><th>) E01E2</th><th></th><th></th><th></th><th></th><th></th><th></th><th>2: 3: 4: 5:</th><th>[esp+4] [esp+8] [esp+C] [esp+10</th><th>000000401 000000001 0643F408] 00000012</th><th></th><th></th><th></th></user32<>	DE2 lockbit.ex	(75CFAB00 e:\$E0DE2 #) E01E2							2: 3: 4: 5:	[esp+4] [esp+8] [esp+C] [esp+10	000000401 000000001 0643F408] 00000012			
Ump 1	Ump 2	Dump 3	Dump 4	Dump 5	💮 Watch 1	[x=] Locals	2 Struct	00	43F3ES 00	18021-					
Address He 0643F608 51 0643F618 21	ex 3 65 72 76 69 6 2 20 73 74 6F 7	3 65 20 53 0 70 65 64	51 4C 57 00 65 00	72 69 74 65 00 F8 43 06	ASCII Service SQL r stopped.e	Write		^ 00	543F3F0 000 543F3F4 06 543F3F8 000 543F3F8 000	000000 43F40 00001	L B				

Figure 136

The ransomware takes a snapshot of all processes in the system (0x2 = TH32CS_SNAPPROCESS):

	0049978E 00499760	6A 00 6A 02		x875W_SF 0 x875W_P 0 >	x875W_SF 0 x875W_P 0 x875W_U 0			
EIP	→• 004997C2	FF DO	call eax		eaxtCrea V	Default (stdcall)	👻 5 💠 🗌 Unlock	
eax= <kerne< th=""><th>132.CreateToolhelp 97C2 lockbit.exe:\$</th><th>325napshot> (76AAF 997C2 #988C2</th><th>890)</th><th></th><th></th><th>1: [esp] 00000002 2: [esp+4] 0000000 3: [esp+8] 00791835 4: [esp+C] 0000001C 5: [esp+10] 0000000</th><th></th></kerne<>	132.CreateToolhelp 97C2 lockbit.exe:\$	325napshot> (76AAF 997C2 #988C2	890)			1: [esp] 00000002 2: [esp+4] 0000000 3: [esp+8] 00791835 4: [esp+C] 0000001C 5: [esp+10] 0000000		
Dump 1	Dump 2	mp 3 🕮 Dump 4	🕮 Dump 5 🛛 👹 Watch 1	Ix=I Locals 2 Struct	0643F3A8 000	00002		

Figure 137

The malicious file retrieves information about the first process from the snapshot via a function call to Process32First:

	004999 004999	09 51 0A FF	75 EC		push ecx	tr ss:Tebp	-14		×8	375W_SF0 x875W_P 0 x8	17SW_U 0
	→• 004999 <	CO FF (00		call eax			eax: Proc *	Def	fault (stdcall)	👻 5 💠 🗌 Unlock
eax= <kernel< th=""><th>132.Process3</th><th>2First> (76 .exe:\$99900</th><th>AB02F0) #98D0D</th><th></th><th></th><th></th><th></th><th></th><th>2:3:4:5:</th><th>[esp+4] 0643F6D0 [esp+8] 00791835 [esp+C] 000001C [esp+10] 00000000</th><th></th></kernel<>	132.Process3	2First> (76 .exe:\$99900	AB02F0) #98D0D						2:3:4:5:	[esp+4] 0643F6D0 [esp+8] 00791835 [esp+C] 000001C [esp+10] 00000000	
Dump 1	Ump 2	Dump 3	Dump 4	Ump 5	🛞 Watch 1	[x=] Locals	2 Struct	0643F3A8 000 0643F3AC 064	004A 3F6D	8	

Figure 138

Interestingly, the malware removes the extension of the process name (if present) before the comparison with the targeted list:

	004998C4 51 004998C5 FF D0	push ecx call eax	ecx:"[Sy eax:Path	Default (stdcall)
eax= <sh]wapi.f< td=""><td>PathRemoveExtensionA> (75FCCE90) 5 lockbit.exe:\$998C5 #98FC5</td><td></td><td></td><td>1: [esp] 0643FSC8 "[System Process]" 2: [esp+4] 0079182 3: [esp+8] 0000001C 4: [esp+C] 0000000 5: [esp+10] 00000000</td></sh]wapi.f<>	PathRemoveExtensionA> (75FCCE90) 5 lockbit.exe:\$998C5 #98FC5			1: [esp] 0643FSC8 "[System Process]" 2: [esp+4] 0079182 3: [esp+8] 0000001C 4: [esp+C] 0000000 5: [esp+10] 00000000
all a la	and complete complete com	and a second second second second	0643F3AC 0643	FSC8 "[System Process]"

Figure 139

An example of such a comparison is shown in figure 140.



	 00499021 00499024 00499024 	FF 34 0A 8D 8D 88 FD FF FF 51	push dword p lea ecx,dwor	edx+ecx*	x875W_C1 0 x875W_C0 0 x875W_E5 0 x875W_SF 0 x875W_P 0 x875W_U 0			
EIP	>• 00499028 <	FF DO	call eax		eax:1str Y	efault (stdcall)	🝷 💈 🗖 Unlock	
eax= <kerne< th=""><th>132.1strcmpi> (7) 9D2B lockbit.exe</th><th>6A76ADO) :\$99D28 #99128</th><th></th><th></th><th>233</th><th>L: [esp] 0643F5C8 "[Sys 2: [esp+4] 04480000 "wx 3: [esp+8] 00791835 3: [esp+C] 0000001C 5: [esp+10] 00000000</th><th>tem Process]" Server"</th></kerne<>	132.1strcmpi> (7) 9D2B lockbit.exe	6A76ADO) :\$99D28 #99128			233	L: [esp] 0643F5C8 "[Sys 2: [esp+4] 04480000 "wx 3: [esp+8] 00791835 3: [esp+C] 0000001C 5: [esp+10] 00000000	tem Process]" Server"	
tilli Dumo 1	SHE Dump 2 SHE	Dumo 3 Elli Dumo 4	Dumo S 🚳 Watch 1	Ixal Locals Struct	0643F3A8 0643F5	C8 "[System Process]"		

The process enumeration continues by calling the Process32Next routine:

	• 0049A	0A7 51			push ecx	8. Ø	80 B		x875W_SF 0 x875W_P 0 x875W_U 0				
	→• 0049A <	DA9 FF C	00		call eax			eax:Proc *	Def	ault (stdcall)	▼ 5 🗘 Unlock		
eax= <kerne< th=""><th>132.Process3 A0A9 lockbit</th><th>2Next> (76A</th><th>#994A9</th><th></th><th></th><th></th><th></th><th></th><th>2: 3: 4: 5:</th><th>[esp+4] 0643F6D0 [esp+8] 00791835 [esp+C] 0000001C [esp+10] 00000000</th><th></th></kerne<>	132.Process3 A0A9 lockbit	2Next> (76A	#994A9						2: 3: 4: 5:	[esp+4] 0643F6D0 [esp+8] 00791835 [esp+C] 0000001C [esp+10] 00000000			
Dump 1	Dump 2	Dump 3	Dump 4	Dump 5	💮 Watch 1	[x=] Locals	2 Struct	0643F3A8 0000 0643F3AC 064	004A 3F6D	8			

Figure 141

OpenProcess is used to open a targeted process (0x1FFFFF = **PROCESS_ALL_ACCESS**):

	 0049E 0049E 0049E 	BBE FF	35 C8 FD FF	FF	push dword push 1	otr ss: [ebp	-238		x8 x8	x875W_C1 0 x875W_C0 0 x875W_E5 0 x875W_SF 0 x875W_P 0 x875W_U 0				
	→• 0049E	SCB FF I	50		call eax			eax:Open	Def	ault (stdcall)	•][5 🗘 🗌 Unlock		
eax= <kernel< th=""><th>132.OpenProc E3CB lockbit</th><th>ess> (76A75</th><th>#9D7CB</th><th></th><th></th><th></th><th></th><th></th><th>2: 3: 4: 5:</th><th>[esp+4] 0000000 [esp+8] 0000014 [esp+C] DAB098F [esp+10] 76A600</th><th>1 4 2 000 kernel32.76A6000</th><th>,</th></kernel<>	132.OpenProc E3CB lockbit	ess> (76A75	#9D7CB						2: 3: 4: 5:	[esp+4] 0000000 [esp+8] 0000014 [esp+C] DAB098F [esp+10] 76A600	1 4 2 000 kernel32.76A6000	,		
U Dump 1	Dump 2	Dump 3	Ump 4	Dump 5	👹 Watch 1	[x=] Locals	Struct	0643F13C 00 0643F140 00	01FFFF	F				
Address H	AV.				LASCIT			0643F144 00	00001A	4				

Figure 142

A process is killed by calling the NtTerminateProcess API:

	 0049E3D3 0049E3D5 0049E3D5 	GA 01 push 1 SG push c2 c7 c5 push c3 c4 c4 c5 c			x875W_C1 0 x87 x875W_SF 0 x87	SW_CO 0 x87SW_ES 0 SW_P 0 x87SW_U 0		
BIB	→• 0049E3DB	FF DO	call eax		eax:NtTel *	Default (stdcall)	▼ 5	Unlock
eax= <ntdll. .text:0049E</ntdll. 	NtTerminateProce	ess> (7704EA40) :\$9E3DB #907DB				1: [esp] 000004 2: [esp+4] 0000 3: [esp+8] DAB0 4: [esp+C] 76A6 5: [esp+10] 75F	80 0001 98F2 0000 kernel32.76A60000 80000 shlwap1.75F80000	
Ump 1	Ump 2	Dump 3 🛛 💭 Dump 4	💭 Dump 5 🛛 👹 Watch 1 🛛 💷 Lo	ocals 🖉 Struct	0643F140 00000 0643F144 00000	0480		

Figure 143

LockBit initializes the COM library for apartment threading using the CoInitializeEx function (0x6 = **COINIT_APARTMENTTHREADED** | **COINIT_DISABLE_OLEIDDE**):

	00498108 6A 06 00498100 6A 00				push 6				x875w_SF 0 x875w_P 0 x875w_U 0			
ETP	→• <u>00498</u>	IOF FF I	57		call edi			edi:CoIn ~	Defa	ault (stdcall) 🔻 🔽 Unlock		
ed1= <combas< th=""><th>se.CoInitial</th><th>12eEx> (76B .exe:\$9810F</th><th>#9750F</th><th></th><th></th><th></th><th></th><th></th><th>2: 3: 4: 5:</th><th>[esp1] 00000000 [esp14] 00000006 [esp14] 00497060 lockbit.00497060 [esp14] 00497060 lockbit.00497060 [esp10] 00000000</th></combas<>	se.CoInitial	12eEx> (76B .exe:\$9810F	#9750F						2: 3: 4: 5:	[esp1] 00000000 [esp14] 00000006 [esp14] 00497060 lockbit.00497060 [esp14] 00497060 lockbit.00497060 [esp10] 00000000		
Dump 1	Dump 2	Dump 3	💷 Dump 4	Dump 5	👹 Watch 1	[x=] Locals	2 Struct	0643F840 0000 0643F844 0000	00000	0		

Figure 144

The ransomware deletes all volume shadow copies on the system by calling the ShellExecuteEx function and running the commands shown below:

	eax:She) V Default (stdcall) V [5] Unlock
eax= <shell32.shellexecuteex> (74439C50) .text:00498890 lockbit.exe:\$98890 #97C90</shell32.shellexecuteex>	21 (#50+41 00497060 lockbit.00497060 31 (#50+8) 00497060 lockbit.00497060 42 (#50+C) 00000000 51 (#50+C) 00000000
Allowed allo a allo a allo e allo e Allowed bury	(d) 0643F844 0643F890



Address	He	<															ASCII
0643FA14	2F	63	20	76	73	73	61	64	6D	69	6E	20	64	65	6C	65	/c vssadmin dele
0643FA24	74	65	20	73	68	61	64	6F	77	73	20	2F	61	6C	6C	20	te shadows /all
0643FA34	2F	71	75	69	65	74	20	26	20	77	6D	69	63	20	73	68	/quiet & wmic sh
0643FA44	61	64	6F	77	63	6F	70	79	20	64	65	6C	65	74	65	20	adowcopy delete
0643FA54	26	20	62	63	64	65	64	69	74	20	2F	73	65	74	20	7B	& bcdedit /set {
0643FA64	64	65	66	61	75	6C	74	7D	20	62	6F	6F	74	73	74	61	default} bootsta
0643FA74	74	75	73	70	6F	6C	69	63	79	20	69	67	6E	6F	72	65	tuspolicy ignore
0643FA84	61	6C	6C	66	61	69	6C	75	72	65	73	20	26	20	62	63	allfailures & bo
0643FA94	64	65	64	69	74	20	2F	73	65	74	20	7B	64	65	66	61	dedit /set {defa
0643FAA4	75	6C	74	7D	20	72	65	63	6F	76	65	72	79	65	6E	61	ult} recoveryena
0643FAB4	62	6C	65	64	20	6E	6F	00	19	18	79	00	24	FB	43	06	bled noy.\$ûC.

The malware also creates multiple processes twice in order to delete (again) all shadow copies and Windows logs. An example of process creation is shown in figure 147 (0x08000000 = **CREATE_NO_WINDOW**):

<pre> 00499402 51 00499403 80 8C 24 94 03 00 00 00499405 6A 00 00499406 6A 00 00499406 6A 00 00499406 6A 00 00499414 6A 01 00499414 6A 01 00499416 6A 00 00499416 6A 00 00499418 6A 00 00499418 6B 08 C 4F 80 03 00 00 00499418 6B 08 08 C 4F 80 03 00 00 0049418 6B 08 C 4F 80 03 00 00 00499418 6B 08 C 4F 80 03 00 00 00499418 6B 08 08 C 4F 80 03 00 00 00499418 6B 08 08 C 4F 80 03 00 00 00499418 6B 08 08 C 4F 80 03 00 00 00499418 6B 08 08 08 08 08 08 08 08 08 08 08 08 08</pre>	push ecx lea ecx,dword ptr ss:[esp+394] push 0 push 0 push 1 push 0 push 1 push 0 push 2 push 2 push cx,dword ptr ss:[esp+3F4] push cx,dword ptr ss:[esp+3F4]	ecx:"cmd ecx:"cmd	(87Tagword FFFF (87Tw_0.3 (Empty)) (87Tw_2.3 (Empty)) (87Tw_2.3 (Empty)) (87Tw_4.3 (Empty)) (87Tw_4.3 (Empty)) (87Tw_4.3 (Empty)) (87Tw_6.3 (Empty)) (87Tw_6.3 (Empty)) (87Tw_6.3 (Empty)) (87Tw_6.4 (Empty)) (87Tw_6.6 (Empty)) (875w_6.6 0) (875w_6.6 0)
	push eck call eax	ecx: "cmd eax:Creal" 3	efault (stdcal) = [esp] 0643F85C "cmd.exe" : [esp+3] 0643F25C "/c vssadmin Delete Shadows / : [esp+2] 00000000 : [esp+2] 00000001
Ump 1 Ump 2 Ump 3 Ump 4 Ump 5 Address Hex 0643FC20 28 51 73 73 61 64 60 69 62 20 43 64 60 69 62 20 44 65 67 67 73 75 61 64 60 69 62 20 44 65 67 67 73 75 61 64 60 69 62 20 44 65 67 67 73 75 61 64 60 69 62 20 44 65 67 60 60 60 60 66 67 73 75 61 65 65 60 60 60 60 60 60 66 67 67 73 67 65 76 60 60 66 66 67 67 67 67 67 67 67 67 <	Watch 1 Inclucals Struct ASCII Inclucals Struct If Control is the state of	0643F824 0643F 0643F824 0643F 0643F824 0643F 0643F825 000000 0643F823 000000 0643F833 000000 0643F838 00000 0643F838 00000 0643F840 0643F 0643F844 0643F	SC "rond.exe" 20 "/c vssadmin Delete Shadows /All /Quiet" 00 00 00 00 00 00 00 88 85



The following processes have been spawned:

- cmd.exe /c vssadmin Delete Shadows /All /Quiet delete all shadow copies
- cmd.exe /c bcdedit /set {default} recoveryenabled No disable automatic repair
- cmd.exe c bcdedit set {default} bootstatuspolicy ignoreallfailures ignore errors in the case of a failed boot / shutdown / checkpoint
- cmd.exe /c wmic SHADOWCOPY /nointeractive invalid syntax
- cmd.exe /c wevtutil cl security clear security log
- cmd.exe /c wevtutil cl system clear system log
- cmd.exe /c wevtutil cl application clear application log

The ransomware forwards the "Volume Shadow Copy & Event log clean" message to the hidden window:

004E0DD4 004E0DD5 004E0DD5 004E0DD7 004E0DDC	51 6A 01 68 01 04 00 00 FF 35 00 88 4F 00	push ecx push 1 push 401 push dword ptr ds:[4F8B00]		X8/SW_B 0 X8/SW_C3 0 X87SW_C1 0 X87SW_C0 0 X87SW_SF 0 X87SW_P 0	x875W_ES 0 x875W_U 0
ETP →• 004E00E2 <	FF DO	call eax	eax: Send	Default (stdcall)	🕶 💈 🖨 Unlock
<pre>eax=<user32.sendmessagea> .text:004E0DE2 lockbit.exe</user32.sendmessagea></pre>	(75CFA800) ::\$E0DE2 #E01E2			2: [esp+4] 00000401 3: [esp+8] 0000001 4: [esp+6] 0643F408 5: [esp+10] 00000002	
Ump 1 Ump 2	Dump 3 🗰 Dump 4 👹 Dum	p 5 👹 Watch 1 🛛 🕸 Locals 🎾 Struct	0643F3E8 0018 0643F3EC 0000	B0214 00401	
Address Hex		ASCII	↑ 0643F3F0 0000 0643F3F4 0643	00001 3F408	
0643F608 56 6F 6C 75 6D 65 0643F618 70 79 20 26 20 45 0643F628 6C 65 61 6F 00 00	20 53 68 61 64 6F 77 20 4 76 65 6E 74 20 6C 6F 67 2 00 00 20 09 97 02 00 00 0	6F Volume Shadow Co 63 py & Event log c	0643F3F8 0000 0643F3FC 0000	00002 00024	



Thread activity – sub_49E730 function

The NtRemoveloCompletion function is utilized to wait for at least a file to be available for encryption:

	 0049E812 0049E814 0049E818 0049E819 0049E819 0049E810 0049E810 0049E812 0049E822 0049E823 	6A 00 8D 4C 24 34 51 8D 4C 24 20 51 8D 4C 24 20 51 8D 4C 24 28 51 5F 35 20 25 4F 00	push 0 lea ecx,dword ptr ss:[esp+34] push ecx lea ecx,dword ptr ss:[esp+20] push ecx lea ecx,dword ptr ss:[esp+28] push ecx push dword ptr ds:[459520]		x87TW_6 3 (Empty) x87TW_7 3 (Empty) x87Statusword 0000 x87Sw_2 0 x87Sw_8 0 x87Sw_2 0 x87Sw_2 0 x87Sw_8 0 x87Sw_9 0 x87Sw_2 0 x87Sw_9 0 x87Sw_2 0
EIP	→• 0049E829 <	FF DO	call eax	eax:NTRe ~	Default (stdcall)
eax= <ntd11.< td=""><td>NtRemoveIoComple</td><td>tion> (7704E7F0) \$9E829 #90C29</td><td></td><td></td><td>1: [esp] 0000038C 2: [esp+4] 0658FB3C 3: [esp+6] 0658FB38 4: [esp+C] 0658FB30 5: [esp+10] 00000000</td></ntd11.<>	NtRemoveIoComple	tion> (7704E7F0) \$9E829 #90C29			1: [esp] 0000038C 2: [esp+4] 0658FB3C 3: [esp+6] 0658FB38 4: [esp+C] 0658FB30 5: [esp+10] 00000000
Ump 1	Ump 2 Ump 2	Dump 3 👹 Dump 4 👹 Dur	np 5 👹 Watch 1 🕼 Locals 🎾 Struct	0658F80C 0000 0658F810 065	0038C 3F83C
Address He 0658FB38 00	x	00 00 08 02 00 00 FD 08 0	ASCII	O658F814 0658 O658F818 0658 O658F818 0658 O658F81C 000	3F 838 8F 85 0 30000



The following file extensions will be skipped:

- .386 .cmd .ani .adv .msi .msp .com .nls .ocx .mpa .cpl .mod .hta
- .prf.rtp.rdp.bin.hlp.shs.drv.wpx.bat.rom.msc.spl.msu
- .ics .key .exe .dll .lnk .ico .hlp .sys .drv .cur .idx .ini .reg
- .mp3 .mp4 .apk .ttf .otf .fon .fnt .dmp .tmp .pif .wav .wma .dmg
- .iso .app .ipa .xex .wad .msu .icns .lock .lockbit .theme .diagcfg
- .diagcab .diagpkg .msstyles .gadget .woff .part .sfcache .winmd

The files that can be found in the following directories will not be encrypted:

- "\$windows.~bt" "intel" "\$recycle.bin" "to.msstyles" "boot" "msbuild" "system volume information"
- "google" "application data" "windows" "windows.old" "appdata" "mozilla" "microsoft shared" "internet explorer"
- "opera" "windows journal" "windows defender" "windowspowershell" "windows security" "windows photo viewer"

The following specific files will also be skipped:

 "iconcache.db" "ntuser.dat.log" "restore-my-files.txt" "autorun.inf" "bootsect.bak" "thumbs.db"

LockBit uses multiple aeskeygenassist operations in order to assist in AES round key generation, as we can see below:



.text:0043D970	sub_43D	970 proc near
.text:0043D970	movups	xmm1, xmmword ptr [edx]
.text:0043D973	aeskeyg	enassist xmm0, xmm1, 1
.text:0043D979	pshufd	xmm3, xmm0, 0FFh
.text:0043D97E	movaps	xmm0, xmm1
.text:0043D981	pslldq	xmm0, 4
.text:0043D986	pxor	xmm0, xmm1
.text:0043D98A	movups	xmmword ptr [ecx], xmm1
.text:0043D98D	movaps	xmm1, xmm0
.text:0043D990	psllda	xmm1, 4
.text:0043D995	pxor	xmm1, xmm0
text:00430999	movans	xmm2, xmm1
text:00430990	nsllda	vmm2 4
text:00430901	pyor	vmm2 vmm1
tout 004309A1	pxor	XIIIIZ, XIIIIZ
.text.004303A3	pxor	Anniz, Annis
.LEX1:004309A9	deskeyg	endssist xinno, xinnz, z
.text:0043D9AF	рѕпита	xmm3, xmm0, 0FFN
.text:0043D9B4	movaps	xmm0, xmm2
.text:0043D9B7	pslldq	xmm0, 4
.text:0043D9BC	pxor	xmm0, xmm2
.text:0043D9C0	movups	xmmword ptr [ecx+10h], xmm2
.text:0043D9C4	movaps	xmm1, xmm0
.text:0043D9C7	pslldq	xmm1, 4
.text:0043D9CC	pxor	xmm1, xmm0
.text:0043D9D0	movaps	xmm2, xmm1
.text:0043D9D3	pslldq	xmm2, 4
.text:0043D9D8	pxor	xmm2, xmm1
.text:0043D9DC	pxor	xmm2, xmm3
.text:0043D9E0	aeskeyg	enassist xmm0, xmm2, 4
.text:0043D9E6	pshufd	xmm3, xmm0, 0FFh
.text:0043D9EB	movaps	xmm0, xmm2
.text:0043D9EE	pslldq	xmm0, 4
.text:0043D9F3	pxor	xmm0, xmm2
.text:0043D9F7	movups	xmmword ptr [ecx+20h], xmm2
.text:0043D9FB	movaps	xmm1, xmm0
.text:0043D9FE	psllda	xmm1, 4
.text:0043DA03	pxor	xmm1, xmm0
.text:0043DA07	movaps	xmm2, xmm1
.text:0043DA0A	psllda	xmm2, 4
.text:0043DA0F	nxor	xmm2, xmm1
.text:0043D413	pxor	xmm2, xmm3
text:00430417	aeskeve	enassist xmm0, xmm2, 8
text:00430410	nshufd	vmm3 vmm0 0FFh
text:00430422	movans	vmmQ vmm2
text:00430422	nellde	vmmQ 4
text:00430A25	Parred	
. LEX1:00430A2A	pxor	XIIIIIO, XIIIIIZ

Figure 150

Address	He	Č.														- 21	ASCII
065BFE40	BC	77	43	88	2F	F4	A2	CO	63	3B	F3	68	85	17	37	FC	WWC./ô¢Ac;ók7ü
065BFE50	4D	ED	F3	1F	62	19	51	DF	01	22	A2	B4	84	35	95	48	Mió.b.QB. "¢ .5.H
065BFE60	D9	C7	A1	40	BB	DE	FO	9F	BA	FC	52	2B	3E	C9	C7	63	ÙÇį@≫Þð.°üR+>ÉÇc
065BFE70	00	01	5A	F2	BB	DF	AA	GD	01	23	F8	46	3F	EA	3F	25	Zo»Bªm.#øF?ê?%
065BFE80	8F	74	65	87	34	AB	CF	EA	35	88	37	AC	OA	62	08	89	.te.4«Iê5.7¬.b
065BFE90	35	44	C2	EO	01	EF	OD	0A	34	67	ЗA	A6	3E	05	32	2F	5DAa.i4g: :>.2/
065BFEA0	7E	67	D7	52	7F	88	DA	58	4B	EF	EO	FE	75	EA	D2	D1	~gxRÚXKïaþuêÓŇ
065BFEB0	B9	D2	E9	CF	C6	5A	33	97	8D	B5	D3	69	F8	5F	01	B 8	'OéIÆZ3µOiø
065BFEC0	F6	AE	85	8E	30	F4	B6	19	BD	41	65	70	45	1E	64	C8	ö®0ô¶.%AepE.dÈ
065BFED0	9F	ED	6D	EO	AF	19	DB	F9	12	58	BE	89	57	46	DA	41	.ima .Où. X%. WFÚA
065BFEE0	F3	BA	EE	BB	5C	A3	35	42	4E	FB	8B	CB	19	BD	51	8A	Ó°î»∖£5BNû.E.½Q.

The file content is encrypted using the AES128 algorithm. Basically, the malware uses aesenc instructions to perform one round of an AES encryption flow:

🗾 🚄 🖼		
.text:0043D8E0		
.text:0043D8E0	loc_43D	8E0:
.text:0043D8E0	lea	eax, [eax+10h]
.text:0043D8E3	movups	xmm0, xmmword ptr [esi+eax-10h]
.text:0043D8E8	pxor	xmm1, xmm0
.text:0043D8EC	pxor	xmm1, xmmword ptr [ecx]
.text:0043D8F0	aesenc	xmm1, xmmword ptr [ecx+10h]
.text:0043D8F6	aesenc	xmm1, xmmword ptr [ecx+20h]
.text:0043D8FC	aesenc	xmm1, xmmword ptr [ecx+30h]
.text:0043D902	aesenc	xmm1, xmmword ptr [ecx+40h]
.text:0043D908	aesenc	xmm1, xmmword ptr [ecx+50h]
.text:0043D90E	aesenc	xmm1, xmmword ptr [ecx+60h]
.text:0043D914	aesenc	xmm1, xmmword ptr [ecx+70h]
.text:0043D91A	aesenc	xmm1, xmmword ptr [ecx+80h]
.text:0043D923	aesenc	xmm1, xmmword ptr [ecx+90h]
.text:0043D92C	aesencl	ast xmm1, xmmword ptr [ecx+0A0h]
.text:0043D935	movups	xmmword ptr [eax-10h], xmm1
.text:0043D939	sub	edx, 1
.text:0043D93C	jnz	short loc 43D8E0

Figure 152



Address	He	¢															ASCII
06AB0000	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
06AB0010	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAAAAA
06AB0020	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAAAAA
06AB0030	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAAAAA
06AB0040	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAAAAA
06AB0050	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAAAAA
06AB0060	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAAAA
06AB0070	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAAAAA
06AB0080	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAAAAA
06AB0090	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAAAAAA
06AB00A0	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAAAAA
06AB00B0	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAAAAA
06AB00C0	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAAAAA
06AB00D0	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAAAAAAA
06AB00E0	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAAAAA
06AB00F0	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAAAAA
06AB0100	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAAAAA
06AB0110	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAAAA
06AB0120	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAAAAA
06AB0130	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAAAAA

Address	He	<															ASCII
06AB0000	C8	89	88	60	3E	F8	64	12	B 3	F1	7E	D7	2A	1A	8A	EE	È`>ød."n~x*î
06AB0010	6C	0B	52	F5	GD	FD	6A	3A	BO	EF	D5	99	C1	D7	24	F2	1.Rômýj:°ïŐ.Ax\$ò
06AB0020	72	3E	B1	2C	60	EA	4C	71	A1	16	DB	FF	85	AO	29	67	r>±, eLqi.0ÿ.)g
06AB0030	09	AO	97	A2	C1	1E	EO	8A	5D	47	9D	6E	5E	F5	C5	76	¢A.a.]G.n^õAv
06AB0040	94	64	92	12	FE	D0	7F	F2	A2	B2	B 7	6F	03	D2	C8	A2	.dbD.oc=.o.OE¢
06AB0050	80	56	22	43	D3	D6	F6	B5	3D	3B	SC	14	31	A1	E1	E3	.V"COOöµ=;1;áã
06AB0060	75	66	92	5A	EO	68	95	CO	B6	00	94	93	D4	4C	7D	BO	uf.Zak.A¶ÔL}*
06AB0070	C1	CD	8F	13	52	E8	A3	F2	05	70	61	2D	35	00	BD	8D	A1Refo.pa-5.%.
06AB0080	30	03	8B	SE	CB	06	DO	BC	99	EO	31	26	D9	82	08	88	0E.D%.a1&U
06AB0090	39	E9	4C	53	A4	5F	43	8D	BF	FF	05	9B	41	72	E3	BE	9éLS¤_C.¿ÿ. Arãa
06AB00A0	4C	A6	1D	AA	OE	75	43	A1	OD	AF	12	DF	D2	2A	25	70	L uC ;
06AB00B0	29	CA	EE	DF	F5	9F	DO	17	A1	57	01	C5	2E	F4	D7	C5)ÊîBÕ.Đ.;W.Â.ôxÂ
06AB00C0	06	50	B 5	43	59	C8	44	AB	86	E6	00	A4	AA	31	93	C8	.PµCYÈD«.æ.¤ª1.È
06AB00D0	19	2F	71	B1	43	D1	1A	41	17	DA	6D	B1	DD	9B	F1	8F	./q±CÑ.A.Úm±Ý.ñ.
06AB00E0	E5	4E	35	E7	53	2E	7C	71	20	7F	A7	69	CA	55	79	BE	aN5c5. q .§iÊUy%
06AB00F0	91	6C	28	BS	EA	E6	3E	13	C7	OD	22	95	AC	2D	9F	F3	.1(_êæ>.Ç.".¬ó
06AB0100	E9	47	BC	77	70	80	DE	C9	20	3F	80	44	OF	B 8	69	C2	éG¼wp.ÞÉ ?.D. iÅ
06AB0110	64	38	44	02	DO	FB	F6	AF	C4	C5	38	56	BC	99	31	AD	dSD.Dûö AASV%.1.
06AB0120	9A	3A	40	0C	66	68	AF	5 B	CC	A9	9E	36	D9	82	87	C4	.:@.fk [1@.60A
06AB0130	24	6E	E0	A7	FC	F2	0E	D3	9B	92	65	B 2	33	2D	44	5E	\$nà§üò.Óe=3-D^

Figure 154

As we mentioned before, only the first 4KB of the file is encrypted. The encrypted content is written to the file using ZwWriteFile:



Figure 155

The BcryptGenRandom routine is utilized to generate 32 random bytes:

0048E1AA 6A 02 0048E1AC FF 75 0C 0048E1AF FF 75 08 0048E1AF FF 75 08	push 2 push dword ptr ss: [ebp+C] push dword ptr ss: [ebp+8]	x875W_B 0 x875W_C3 0 x x875W_C1 0 x875W_C0 0 x x875W_SF 0 x875W_P 0 x	(875W_C2 0 (875W_ES 0 x875W_U 0
COABELISA FF DO	call eax	eax:BCFY > Default (stdcall) 1: [esp] 00000000	▼ 5 🗘 🗆 Unlod
eax= <bcrypt.bcryptgenrandom> (73817DE0) .text:004BE1B4 lockbit.exe:\$BE1B4 #8D584</bcrypt.bcryptgenrandom>		2: [esp+4] 0658FAA8 3: [esp+8] 0000020 4: [esp+C] 00000002 5: [esp+10] 06AA0050	
Ump 1 Ump 2 Ump 3 Ump 4 Ump 5	5 👹 Watch 1 🛛 🕅 I.ocals 🎾 Struct	0658FA08 00000000 0658FA0C 0658FAA8	
Address Hex	ASCII	0658FA10 00000020 0658FA14 00000002	
Address Hex 065BFAA8 54 1F 11 C0 C0 0F F1 28 11 48 11 51 42 56 D7 065BFAB8 02 A6 F5 84 30 5E BC 72 FC 01 5A BF E5 F8 01	ASCII EE TAA.ñ(.K.QBVX1 C3 .¦ő'oA%rü.Z2áo.A		



The buffer generated above is transformed using the Curve25519 wrapper and then copied to a new buffer together with the session ECC public key (see figure 157). Based on the implementation of the Curve25519 algorithm, it is used to generate a shared secret (32-byte value).

Address	He	ĸ															ASCII
065BF8E0	39	23	1A	E5	80	F7	25	91	20	63	11	0A	F7	98	91	56	9#. a. ÷%. c÷V
065BF8F0	05	C5	A3	C4	28	56	41	B5	EA	DO	CD	2E	F1	83	DO	76	.£Ä(VAµêĐÍ.ñ*Đv
065BF900	A5	27	53	2B	E9	DO	FO	C9	24	B3	OB	74	66	F4	FC	4B	¥'S+éDÖÉ\$*.tfôüK
065BF910	DA	5F	25	AS	37	D8	DC	D7	4B	50	C5	86	71	DA	D5	3E	Ú_% 70ÜxKPÅ.qÚŐ>

Figure 157

The AES128 key and IV (initialization vector) are encrypted using Curve25519 with the session ECC public key, as highlighted below:

Address Luev	ACCTT	0658FAS4 004F88A0 lockbit.004F88A0
Dump 1 Dump 2 Dump 3 Dump 4	Dump 5 🛞 Watch 1 🕅 Locals 🤌 Struct	055BFA4C 0658FA88 0658FA50 0658FA48
eax=lockbit.0042F6E0 .text:0042AD05 lockbit.exe:\$2AD05 #2A105		1: [esp] 0658FA88 2: [esp+4] 0658FA88 3: [esp+8] 004F88A0 lockbit.004F88A0 4: [esp+6] 06AA0050 5: [esp+10] 06AA0050
• 00424CEE 57 • 00424CEF 80 45 9C • 00424CF2 80 45 9C • 00424CF2 80 45 9C • 00424CF5 50 • 00424CF7 80 85 7C FF FF • 00424CF7 50 • 00424CF2 50 • 00424CF3 50 • 00424CF3 50 • 00424CF5 50 • 00424CF • 00424CF • 00424CF • 00 • 00424CF • 00	push edi lea eax.dword ptr ss:[ebp-64] mov byte ptr ss:[ebp-5].0 push eax lea eax.dword ptr ss:[ebp-84] push eax. mov eax.dword ptr ds:[460670] mov eax.dword ptr ds:[eax] cal eax	X877W_6 3 (Empty) X87TW_7 3 (Empty) X877W_6 3 (Empty) X87TW_7 3 (Empty) X875YL5000 X875W_28 0 X875W_20 0 X875W_22 0 X875W_210 X875W_20 0 X875W_25 0 X875W_25 0 X875W_20 0 X875W_25 0 X875W_20 0 V Default (stdcall) V 5 5 0 Unio

Figure 158

Each encrypted file has a 512-byte footer that will be explained in detail. It's written to the encrypted file by calling the ZwWriteFile API:

STP	 00495533 00495955 00495955 00495955 00495956 00495956 00495956 00495963 00495964 00495966 00495968 00495968 	6A 00 8D 41 20 50 FF 71 2C 8D 41 04 FF 71 28 51 6A 00 66 00 FF 76 30 FF 76 30 FF 75 30 FF 75	<pre>push 0 lea eax,dword ptr ds:[ecx+20] push eax push dword ptr ds:[ecx+20] lea eax,dword ptr ds:[ecx+4] push dword ptr ds:[ecx+8] push eax push ecx push 0 push</pre>	eax: 2wWr eax: 2wWr eax: 2wWr eax: 2wWr	x877 m_03 (Empty) x877 m_13 (Empty) x877 m_23 (Empty) x877 m_13 (Empty) x877 m_43 (Empty) x877 m_53 (Empty) x877 m_63 (Empty) x877 m_53 (Empty) x877 m_63 (Empty) x877 m_53 (Empty) x877 m_63 (Empty) x877 m_53 (Empty) x875 tatusword 0000 x875 m_63 (Empty) x875 m_8 0 x875 m_63 0 x875 m_65 0 x875 m_65 0 x875 m_56 0 x875 m_7 0 x875 m_20 0
Carlos .	<	FF DU	I can cax	CdA Chill	Default (stdcall) 🔹 5 🔹 🗌 Unlock
eax= <ntdl< th=""><th>1.ZwwriteFile> (77 DE970 lockbit.exe:</th><th>04E7E0) \$9E970 #90D70</th><th></th><th></th><th>1: [csp1 0000000 2: [csp+4] 0000000 3: [csp+6] 0000000 4: [csp+C] 06AA0050 5: [csp+10] 06AA0054</th></ntdl<>	1.ZwwriteFile> (77 DE970 lockbit.exe:	04E7E0) \$9E970 #90D70			1: [csp1 0000000 2: [csp+4] 0000000 3: [csp+6] 0000000 4: [csp+C] 06AA0050 5: [csp+10] 06AA0054
Dump 1	Ump 2	Dump 3 📲 Dump 4 🖏 Du	mp 5 👹 Watch 1 🛛 🕸 I Locals 🤌 Struct	0658FAFC 000	03A8
Address	Hex		ASCII	▲ 0658F804 0000 0658F808 06A	10000 40050
06AB0000 06AB0010 06AB0020 06AB0030 06AB0040	E1 E4 BD 09 16 50 5F 7A A0 F7 6E 42 20 C1 E4 54 86 D3 EB 1E 73 E2 A7 C0 7 F0 02 3E 11 F6	80 6F C0 F5 94 32 CD 63 95 8C 86 C7 19 61 EB 81 2A 25 D4 CF 0A 1B 68 79 43 F8 44 EA 0A 35 03 37 38 2A CC 93 34 7B 98 BE	12 90 (00:) 040.210. 57 03 2 - 08.%, C actg. 15 70 Aat.0=%01.hyup 02 28 0.834/c000.5.7. EF EC (x0.>.0;*1.4[.%1]	0658FB0C 06A 0658FB10 06A 0658FB14 000 0658FB16 06A 0658FB16 06A	0054 50000 10200 40070 10000

Figure 159

NtSetInformationFile is used to append the ".lockbit" extension to encrypted files (0xA = **FileRenameInformation**):

 0049ECFD 0049ECFF 0049ECFF FF 74 24 18 0049ED03 66 0F 13 44 24 48 0049ED04 0049ED04 0049ED04 0049ED04 0049ED05 FF 74 24 34 0049ED06 FF 74 24 34 	push A push dword ptr ss:[esp+18] movlpd qword ptr ss:[esp+48],xmm0 push edi push eax push dword ptr ss:[esp+34] call lockbir.43130	eax:Nt5	x87 x87 x87 x87 x87	TW_6 3 (Empty) > 'StatusWord 0000 'SW_8 0 x87SW_C3 'SW_C1 0 x87SW_C0 'SW_SF 0 x87SW_P	0 x87SW_C2 0 x87SW_C2 0 x87SW_ES 0 x87SW_U	0 0 0
EIE 0049E014 FF D0	call eax	eax:NtS	e V Defa	ult (stdcall)		▼ 5 \$ Unlock
eax= <ntd]].ntsetinformationfile> (7704E9F0) .text:0049ED14 lockbit.exe:\$9ED14 #9E114</ntd]].ntsetinformationfile>			2: 3: 4: 5:	[esp+4] 065BFB60 [esp+8] 06710000 [esp+C] 0000003E [esp+10] 0000000A		
5 Dump 1 5 Dump 2 5 Dump 3 5 Dump 4 5 Dump	5 👹 Watch 1 🛛 🕅 Locals 🖉 Struct	065BFB0C 065BFB10	000003A8			
Address Hex 06710000 00 00 00 00 00 00 00 00 2E 00 00 00 5C 00 3F	ASCII	065BFB14 065BFB18 065BFB1C	06710000 0000003E 0000000A			
06710020 74 00 56 00 74 00 76 00 74 00 26 00 74 00 65 00 75 00 74 00 65 00 75	00 tt.x.t 00 c.k.b.i.t	0658F820 0658F824 0658F828	0049E730 0049E730 6E614DRE	lockbit.0049E730 lockbit.0049E730	SEGIADRE fro	a dwaani Ordina'

Figure 160

As we can see below, the files are partially encrypted, which is enough to make them useless without decrypting them:



₩ HxD - [Z:\te	HxD - [Z:\test.txt.lockbit]																
📓 File Edit S	Searc	h V	iew	Ana	lysis	Ext	ras	Win	dow	?							
🗋 👌 - 🗐	hum	3	+ +	16		~	AN	SI		~	he	x	~				
📓 test.txt.lock	bit																
Offset(h)	00	01	02	03	04	05	06	07	08	09	OA	0B	0C	OD	0E	OF	
00000F30	A6	CO	91	AF	FD	A6	D3	4B	FB	38	Al	D2	D5	AF	BC	AA	¦À `¯ý¦ÓKû8;ÒÕ¯4ª
00000F40	DO	03	22	9D	3B	6D	ED	8B	EO	71	41	54	07	EO	7E	4B	Ð.".;mí<àqAT.à~K
00000F50	CE	5A	B6	19	AE	F2	05	DO	4A	39	CE	77	04	BD	F9	ED	ÎZ¶.®ò.ĐJ9Îw.%ùi
00000F60	CC	D8	43	22	15	31	FC	74	1B	BF	37	04	62	4A	90	06	ÌØC".lüt.¿7.bJ
00000F70	96	61	FO	C5	66	D4	3B	CA	CA	5A	D8	CD	A2	06	9A	0B	-aðÅfÔ;ÊÊZØÍ¢.š.
00000F80	F4	E9	B9	1B	D2	D2	0C	FO	97	53	DO	C2	68	D8	59	75	ôé¹.ÒÒ.ð—SĐÂhØYu
00000F90	7A	AO	B7	A3	EF	D5	El	A4	D5	05	5D	FE	72	61	15	F4	z £ïŐá¤Õ.]þra.ô
00000FA0	C3	6B	01	A7	9E	lD	C2	3F	B8	CE	81	5C	23	95	2E	85	Ãk.§ž.Â?,Î.\#•
00000FB0	1B	39	BC	7D	36	DO	C9	37	F3	D8	A9	C7	1F	51	18	31	.94)6ĐÉ7óØ©Ç.Q.1
00000FC0	07	0A	B6	4C	46	89	7C	Al	4F	F8	77	E6	F4	03	20	74	ILF% ; Ogwæô. t
00000FD0	91	36	A5	6F	23	Dl	DE	DD	2A	F4	FB	7A	5D	10	2F	AE	`6¥o‡ÑÞÝ*ôûz]./⊗
00000FE0	CD	32	7E	85	83	0B	A2	25	D4	4C	85	93	7A	8C	D6	01	12~f.¢%ÔL"zŒÖ.
00000FF0	04	FD	C6	F5	05	D4	96	CF	Al	47	85	6E	E5	F8	BC	28	.ýÆõ.Ô−Ï;G…nåø4(
00001000	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	аааааааааааааааа

Out of the 512 bytes from the footer, we can highlight the following bytes:

- last 8 bytes first 8 bytes from the session ECC public key
- previous 8 bytes hard-coded bytes that correspond to this particular LockBit sample
- 112 bytes session ECC private key that was encrypted using the Master ECC public key (also stored in the Private registry value)
- 96 bytes AES key + IV that were encrypted using the session ECC public key

📓 test.txt.lock	bit																
Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	OD	0E	OF	
000027C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000027D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000027E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000027F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00002800	81	E4	BD	09	16	50	BO	6F	CO	F5	94	32	CD	63	B2	90	.ä≒P°oÀõ″2Íc⁵.
00002810	5F	7A	AO	F7	6E	42	95	BC	86	C7	19	61	EB	Bl	67	03	z ÷nB·4tC.aë±g.
00002820	20	C1	E4	54	86	D3	2A	25	D4	CF	OA	18	68	79	55	70	ÁäTtó*%ÔÏhyUp
00002830	EB	1E	73	E2	A7	CO	43	F8	44	EA	OA	35	03	37	0C	2E	ë.sâ§ÀCøDê.5.7
00002840	D7	FO	02	3E	11	F6	3B	2A	CC	93	34	7B	9B	BE	EF	EC	*8.>.ö;*Ì"4{>%ïì
00002850	82	3F	56	6E	3D	F6	53	56	EB	03	AO	55	7E	CA	DO	19	,?Vn=öSVë. U~ÊĐ.
00002860	B6	D9	FO	DA	69	DA	DC	9B	7A	EE	65	AB	DB	D2	8D	Cl	¶ÙðÚiÚÜ>zîe≪ÛÒ.Á
00002870	AD	8E	30	CE	6E	D6	BC	81	AO	7A	80	53	D6	6D	2A	BO	.Ž<ÎnÖ4. zŒSÖm*°
00002880	C7	81	F7	7B	C3	7B	63	7E	FE	CO	9A	F8	4B	EE	5D	E4	Ç.÷{Ã{c~þÀšøKî]ä
00002890	5C	OF	CO	86	B5	CB	89	CF	3E	1E	11	CB	08	D3	39	F6	\.ÀtµË‰Ï>Ë.Ó9ö
000028A0	Cl	5F	E3	0B	52	F6	71	DB	53	39	74	E8	09	66	45	7D	Á ã.RögÛS9tè.fE}
000028B0	1E	24	45	02	21	C4	E3	33	FA	AA	FA	13	4A	DO	DF	FD	.ŞE.!Äã3úªú.JĐBý
000028C0	CO	BC	9E	C4	7A	53	1E	41	EB	6A	81	A2	06	07	C2	64	À4žÄzS.Aëj.¢Âd
000028D0	BE	58	82	0B	47	4C	C4	4C	CD	2E	65	40	C8	C7	26	85	₩X,.GLÄLÍ.e@ÈÇ&
000028E0	45	2C	90	41	37	BO	E9	5D	99	A7	55	BA	FA	DE	78	F9	E,œA7°é]™§U°úÞxù
000028F0	56	A9	6C	D3	10	EF	FO	17	02	EO	40	DE	FO	OF	21	32	V©ló.ïðà@Þð.!2
00002900	3C	67	55	FB	A9	44	BC	C6	35	A3	54	F2	4A	41	C4	65	<guû©d4£5£tòjaäe< td=""></guû©d4£5£tòjaäe<>
00002910	28	D9	3A	FA	AC	B2	D5	FO	9F	3A	36	86	35	11	DC	5B	(Ù:ú-*ÕðŸ:6†5.Ü[
00002920	39	23	1A	E5	80	F7	25	91	20	63	11	0A	F7	98	91	56	9 #.å€÷%`c÷~` V
00002930	05	C5	A3	C4	28	56	41	B 5	EA	DO	CD	2E	Fl	B3	DO	76	.Å£Ä (VAµêÐÍ.ñ'Đv
00002940	1B	B6	AE	05	56	D8	99	BD	5F	EB	A3	CF	82	8F	A8	89	.¶@.Vؤ%s_ë£Ï,.~%
00002950	49	D2	11	DD	6D	4B	AD	66	75	A2	E8	63	OD	AF	F7	01	IÒ.ÝmK.fu¢èc. ÷.
00002960	69	70	74	92	A8	C3	39	FE	9F	99	2C	2F	OD	9E	59	9F	ipt' Ä9þŸ™,/.žYŸ
00002970	42	15	4E	AE	13	8E	EF	6F	Cl	A9	19	7A	CA	4C	Al	4C	B.NO.ŽioÁO.zÊL;L
00002980	24	6C	F9	DO	D9	OA	EE	E1	CF	92	65	AO	32	A5	C8	97	\$lùĐÙ.îáÏ'e 2¥È-
00002990	EC	6C	CO	2A	E3	39	EE	AB	0C	6F	81	29	CC	7F	71	51	ìlÀ*ã9î≪.o.)Ì.qQ
000029A0	4B	90	B6	27	E5	18	26	D5	AB	A8	F2	63	5A	8D	AC	A5	Kœ¶'å.≨Õ≪¨òcZ.¬¥
000029B0	5B	A2	E5	64	CO	15	4B	18	2F	27	5B	6F	6A	ED	5A	8A	[¢ådÀ.K./'[ojíZŠ
00002900	DD	01	71	CA	95	43	31	C2	FB	Dl	58	7C	9B	52	OA	90	Ý.qÊ•ClÂûÑX >R
000029D0	E1	DO	87	54	AO	63	AS	2F	41	58	55	24	11	91	08	19	áЇT c″/AXU\$.`
000029E0	8F	34	A9	Al	C9	8A	6A	31	82	15	3A	3B	Al	55	EB	76	.4©;ÉŠjl,.:;;Uëv
000029F0	2E	D8	73	D4	3F	E5	DD	38	A5	27	53	2B	E9	DO	FO	C9	.ØsÔ?åÝ8¥'S+éĐðÉ



We can observe the icon of the encrypted files in figure 163:

Local Disk (Z:)				~	õ
Name	Date modified	Туре	Size		
📓 Restore-My-Files.txt	2/7/2022 4:24 AM	TXT File	1 KB		
🖾 test.txt.lockbit	2/7/2022 4:52 AM	LOCKBIT File	11 KB		

Figure 163

We continue with the analysis of the main thread.

The binary sends the "Cleanup" message to the hidden window via a function call to SendMessageA.

Printing ransom notes

The process enumerates the local printers using the EnumPrintersW function (0x2 = **PRINTER_ENUM_LOCAL**):

	 004A93E9 004A93EA 004A93ED 004A93ED 004A93F1 004A93F2 004A93F2 004A93F4 004A93F6 	51 80 40 F8 51 FF 75 F8 53 6A 01 6A 00 FF 75 E8	push ecx lea ecx,dw push ecx push dword push ebx push 1 push 0 push dword	ord ptr ss:[ebp-1 ptr ss:[ebp-1 ptr ss:[ebp-1	80 - 80 80			x87TW_6 3 (Empty) x87StatusWord 0000 x87StatusWord 0000 x87SW_80 0 x87SW_C3 x87SW_C1 0 x87SW_C0 x87SW_SF 0 x87SW_P	0 x875W_C2 0 0 x875W_ES 0 0 x875W_U 0
E10	004A93F9 <	FF D0	call eax			ea	KEEDUMP) V	Default (stdcall)	▼ 5 🗘 🗌 Unlock
eax= <winspo< th=""><th>ol.EnumPrintersw></th><th>(689F3850) 93F9 #A87F9</th><th></th><th></th><th></th><th></th><th></th><th>2: [esp+4] 00000000 3: [esp+8] 00000001 4: [esp+C] 06300000</th><th></th></winspo<>	ol.EnumPrintersw>	(689F3850) 93F9 #A87F9						2: [esp+4] 00000000 3: [esp+8] 00000001 4: [esp+C] 06300000	
Dump 1	U Dump 2 U Du	mp 3 📖 Dump 4	🚛 Dump 5 🛛 👹 Watch 1	x= Locals	Struct	0019	FASC 0000	0002	
Address He 06300000 00 06300010 00 06300020 00	x 00 00 00 00 00 00 00 00 00 00 00 00 00	0 00 00 00 00 00 0 00 00 00 00 00 0 00 0	ASCII 00 00 00 00 00 00 00 00			^ 0019 0019 0019 0019 0019	FA94 0000 FA98 0630 FA9C 0000 FAA0 0019 FAA4 0019	0001 0000 01F8 FADC FAD8	

Figure 164

The ransomware avoids the following values that don't correspond to physical printers: "Microsoft XPS Document Writer" and "Microsoft Print to PDF".

The OpenPrinterW routine is utilized to retrieve a handle to the printer:

004AA1E4 6A 0 004AA1E6 51 004AA1E7 53			00		push 0 push ecx push ebx			ebx:L"Fax	x875w_C1 0 x875w_C0 0 x875w_E5 0 x875w_SF 0 x875w_P 0 x875w_U 0			
	004AA	LEGI FF	00		call eax			eax:OpenP) ~	✓ Default (stdcall)			
.text:004A	A1E8 lockbit	.exe:\$AA1E8	#A95E8						2: [esp+4] 0019FA78 3: [esp+8] 0000000 4: [esp+C] 06300028 &L"Fax"			
Ump 1	Dump 2	Ump 3	Dump 4	Ump 5	🛞 Watch 1	(x=) Locals	Struct	0019F954 0630 0019F958 0019	300062 L"Fax" L9FA78			
Address H	ev				ASCTT			0019F95C 0000	300000			

Figure 165

StartDocPrinterW is used to notify the print spooler that a document is to be spooled for printing:

 004AA3F8 004AA3F9 004AA3F8 		F8 51 F9 6A F8 FF	51 push ecx 6A 01 push 1 FF 75 DC push dword ptr ss:				ecx:&L"Fa: [ebp-24]:			x875w_C1 0 x875w_C0 0 x875w_E5 0 x875w_SF 0 x875w_P 0 x875w_U 0			
eax= <winsp< th=""><th>ool.StartDocP</th><th>erinterw> (</th><th>689EF730)</th><th></th><th>call eax</th><th></th><th></th><th>eax: Starti</th><th>* De 123</th><th>efault (stdcall) : [esp] 02556BC4 "gg" : [esp+4] 00000001 : [esp+8] 0019F970 &L"Fax"</th><th>▼ 5 🗘 🗌 Unlock</th></winsp<>	ool.StartDocP	erinterw> (689EF730)		call eax			eax: Starti	* De 123	efault (stdcall) : [esp] 02556BC4 "gg" : [esp+4] 00000001 : [esp+8] 0019F970 &L"Fax"	▼ 5 🗘 🗌 Unlock		
.text:004A	A3FE lockbit.	exe: \$AA3FE	#A97FE						4	: [esp+C]_06300028 &L"Fax"			
Dump 1	Ump 2	Ump 3	Ump 4	Dump 5	👹 Watch 1	Ix=I Locals	Struct	0019F954 02 0019F958 00	556B	C4 "99" 01			
Addeser H	av.				ACCTT	T.		0019F95C 00	19F9	70 &L"Fax"			



The StartPagePrinter API notifies the spooler that a page will be printed on the printer:

004AAS4E FF 75 DC	push dword ptr ss:[ebp-24]	[ebp-24]:			
BIC OUTAASSI FF DO	call eax,	eax:starti v	Default (stdcall)	🔻 💈 🗘 Unlock	
eax= <winspool.startpageprinter> (689F)</winspool.startpageprinter>	8A90) 951		1: [esp] 025568.4 00 2: [esp+4] 06300028 &L" 3: [esp+8] 00000002 4: [esp+C] 00000200	"Fax"	
	num 4 - Minner 6 - Minner 6 - North and - State	0019F95C 0255	68C4 "gg"		

Figure 167

The ransom note is printed via a function call to WritePrinter:

	 004AA698 004AA699 004AA690 004AA690 004AA642 	51 FF 75 08 FF 85 FC FE FF FF FF 75 DC	push dword ptr ss: ebp+8 push dword ptr ss: ebp-104 push dword ptr ss: ebp-24	[ebp-24]:	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		
EIR	004AAGAS <	FF DO	call eax	eax:write ~	Default (stdcall)	▼ 5 C Unloc	
eax= <winspo< th=""><th>AGAS lockbit.exe:</th><th>(689F52A0) \$AA6A5 #A9AA5</th><th></th><th></th><th>2: [esp+4] 06440000 3: [esp+8] 00000200 4: [esp+C] 0019F97C</th><th></th></winspo<>	AGAS lockbit.exe:	(689F52A0) \$AA6A5 #A9AA5			2: [esp+4] 06440000 3: [esp+8] 00000200 4: [esp+C] 0019F97C		
Dump 1	Ump 2	Dump 3 👹 Dump 4 👹 Du	mp 5 Watch 1 🛛 🕸 I Locals 🎾 Struct	0019F950 0255 0019F954 0644	68C4 "gg" 40000		
Address He 06440000 40 06440010 6F 06440020 61 06440030 6E	ex 6F 63 6B 42 69 6D 77 61 72 65 1 74 61 20 61 72 64 20 65 6E 63	74 20 32 2E 30 20 52 61 0D 0A 0D 0A 59 6F 75 72 72 79 70 74 65 64 0D 0A	ASCII 67 73 LockBit 2.0 Rans 20 64 OmwareYour d 20 61 ata are stolen a 54 68 nd encryptedTh	0019F958 0000 0019F950 0013 0019F950 0633 0019F964 0000 0019F968 0000	00200 9F97C 00028 &L"Fax" 00020 00200		

Figure 168

The EndPagePrinter routine notifies the print spooler that the application is at the end of a page in the print job:

 004AA7ED FF 75 DC 	push dword ptr ss: [ebp-24]	[ebp-24]:			
C C C C C C C C C C C C C C C C C C C	call eax.	eax:EndPai V	Default (stdcall)		
eax= <vinspool.endpageprinter> (689F3C00)</vinspool.endpageprinter>			1: [esp] 0256864 90 2: [esp+4] 06300028 &L"Fax" 3: [esp+8] 0000002 4: [esp+C] 0000020		
	- Marine 1	0019F95C 0255	68C4 "gg"		

Figure 169

The printing operation is effected 10000 times, as displayed in figure 170:

🚺 🛃 🖼		
.text:004AA7ED	loc 444	760.
.text:004AA7ED	push	[ebp+var_24]
.text:004AA7F0 .text:004AA7F2	call mov	<pre>eax ; dword_4F8D3C eax, [ebp+var_1C]</pre>
.text:004AA7F5	inc	eax
.text:004AA7F6	cmp	[edp+var_10], eax eax, <mark>9999</mark>
.text:004AA7FE	jl	loc_4AA410

Figure 170

The print job operation is completed by calling the EndDocPrinter and ClosePrinter APIs.

LockBit continues the printer enumeration by searching for network printers in the computer's domain, network printers and print servers in the computer's domain, and the list of printers to which the user has made previous connections. These function calls can be seen below (0x40 = **PRINTER_ENUM_NETWORK**, 0x10 = **PRINTER_ENUM_REMOTE**, 0x4 = **PRINTER_ENUM_CONNECTIONS**):





Figure 171



Figure 172



Figure 173

LockBit Wallpaper Setup

The ransomware sends the "[+] Setup wallpaper" message to the hidden window.

The GdiplusStartup API is utilized to initialize Windows GDI+:

00487395 6A 00 00487340 80 50 34 FF FF FF 00487345 80 50 30 FF FF FF 00487347 80 80 30 FF FF FF 00487340 80 80 30 FF FF FF				push 0 lea ecx,dword ptr ss:[ebp-CC] push ecx lea ecx,dword ptr ss:[ebp-DO] push ecx call eax				x8/s/statusmorg uuuu x875W_EB 0 x875W_C3 0 x875W_C2 0 x875W_C1 0 x875W_C0 0 x875W_C2 0 x875W_SF 0 x875W_C9 0 x875W_E5 0 x875W_SF 0 x875W_P 0 x875W_U0 0				
eax= <gdiplu< th=""><th>s.GdiplusSt</th><th>artup> (6C6 .exe:\$873AE</th><th>80850) #867AE</th><th></th><th></th><th></th><th></th><th>(cax.suring) +</th><th>Default (stdc 1: [esp] 2: [esp+4 3: [esp+6 4: [esp+6]</th><th>all) 0019FA20 4] 0019FA24 8] 00000000 1] 043F0000</th><th></th><th>▼ 5 ¢ Unlock</th></gdiplu<>	s.GdiplusSt	artup> (6C6 .exe:\$873AE	80850) #867AE					(cax.suring) +	Default (stdc 1: [esp] 2: [esp+4 3: [esp+6 4: [esp+6]	all) 0019FA20 4] 0019FA24 8] 00000000 1] 043F0000		▼ 5 ¢ Unlock
Dump 1	Dump 2	Dump 3	Dump 4	Dump 5	💮 Watch 1	[x=] Locals	2 Struct	0019F7E4 0019 0019F7E8 0019	9FA20 9FA24			
Address 14	av				ASCTT	1		0019F7EC 0000	00000			

Figure 174

The file retrieves the width of the screen of the primary display monitor via a function call to GetSystemMetrics:



004874EE 6A 00 push 0	
Call eax	eax:GetSyl ✓ > Default (stdcall)
eax= <user32.getsystemmetrics> (75D08000) .text:004874F0 lockbit.exe:\$874F0 #868F0</user32.getsystemmetrics>	2: [\$594] 0030000 2: [\$594] 043F0000 3: [\$594] 00323000 3: [\$594] 00223000
	0019F7EC 00000000

The malware allocates memory for Windows GDI+ objects using GdipAlloc:

• 0041673A	FF 75 E8	push dword ptr ss:[ebp-18]				
E1P → 0041673D	FF DO	call eax	eax:GdipA v	Default (stdcall)	▼ 5 🗘 🗆 Uniod	
<pre>eax=<gdiplus.gdipalloc> (60 .text:0041673D lockbit.exe:</gdiplus.gdipalloc></pre>	675A40) \$1673D #1583D			1: [esp] 00000010 2: [esp+4] 58857373 3: [esp+8] 00000438 4: [esp+C] 00000780		
Mana Maria Maria			0019F7BC 0000	00010		

Figure 176

A Bitmap object is created based on an array of bytes by calling the GdipCreateBitmapFromScanO function (0x26200a = **PixelFormat32bppARGB**):

 0041800D 00418010 00418015 00418017 00418017 	51 6A 00 68 0A 20 26 00 6A 00 FF 75 0C FF 75 08	push ecx push 0 push 26200A push dword ptr ss: ebp+C push dword ptr ss: ebp+8		x87StatusWord 0000 x87SW_B 0 x87SW_C3 0 x87SW_C1 0 x87SW_C0 0 x87SW_SF 0 x87SW_P 0	x875W_C2 0 x875W_E5 0 x875W_U 0
	FF DO	call eax	eax:gd1pcl♥	Default (stdcall) 1: [esp] 00000780	▼ 5 🗘 Unlock
.text:0041B01D lockbit.ex	e:\$1801D #1A41D			2: [esp+4] 00000438 3: [esp+8] 00000000 4: [esp+C] 0026200A	
Dump 1 Dump 2	🖉 Dump 3 🛛 🖓 Dump 4 🛛 🖓 🛛	Dump 5 👹 Watch 1 🛛 🕅 🖉 Struct	0019F798 0000 0019F79C 0000	00780 00438	
Address Hex 0019F7CC 00 00 00 00 E4 4 0019F7DC E0 FA 19 00 67 7	6 75 6C 50 00 00 00 80 38 6 48 00 80 07 00 00 38 04	ASCII 75 GCòFulP*;ul 00 00 dúgvK8	0019F7A0 0000 0019F7A4 0020 0019F7A8 0000 0019F7AC 0019	00000 5200A 00000 9F7CC	

Figure 177

CreateStreamOnHGlobal is utilized to create a stream object:

	 004E03 004E03 004E03 	251 51 252 6A 254 FF	00 75 FC		push ecx push 0 push dword ptr ss: Tebp-4T				x875w_C1 0 x875w_C0 1 x875w_E5 0 x875w_SF 0 x875w_P 0 x875w_U 0		
ETP	> 004E0	257 FF I	DO		call eax			eax:Creati ~	Defa	ault (stdcall)	▼ 5 🗘 Unlock
eax= <comba< th=""><th>257 lockbit</th><th>eamOnHGloba</th><th>1> (76C007B0 #DF657</th><th>)</th><th></th><th></th><th></th><th></th><th>2: 3: 4:</th><th>[esp+4] 00000000 [esp+8] 0019E2D4 [esp+C] 5BB57373</th><th></th></comba<>	257 lockbit	eamOnHGloba	1> (76C007B0 #DF657)					2: 3: 4:	[esp+4] 00000000 [esp+8] 0019E2D4 [esp+C] 5BB57373	
Dump 1	Ump 2	Dump 3	Dump 4	Ump 5	💮 Watch 1	(x=) Locals	2 Struct	0019E29C 05CA 0019E2A0 0000	A0014	4 D	
Addrare U	av.				ACCTT			0019E2A4 0019	9E2D4	4	

Figure 178

The binary creates a Bitmap object based on the above stream using GdipCreateBitmapFromStream:

0042A2AD 51 0042A2AE FF 75 08	push ecx push dword ptr ss:[ebp+8]	x875w_5F 0 x875w_P 0 x875w_U 0
EIP 0042A2BI FF DO	call eax eax:Gdip	Default (stdcall)
eax= <gdiplus.gdipcreatebitmapfromstream> (6C680810) .text:00424281 lockbit.exe:\$24281 #29681</gdiplus.gdipcreatebitmapfromstream>		2: (esp+4) 00196288 3: (esp+8) 58857373 4: [esp+6] 02593018
Ump 1 Ump 2 Ump 3 Ump 4 Ump 5	Image: Watch 1 IX=I Locals Image: Watch 2 O019E268 O019E266 O019E26C	02593018 0019E288
Address Hex 02593018 8C 58 B3 76 53 54 52 4D 01 00 00 00 00 00 00 00	ASCII 0019E274 X*VSTRM	58857373 02593018 76820000 combine 76820000

Figure 179

A new private font collection is created via a call to GdipNewPrivateFontCollection:

• 00419904	FF 75 E8	push dword ptr ss:[ebp-18]				
	FF DO	Call eax	eax:GdTpN v	Default (stdcall)	👻 💈 💭 Unlock	
<pre>eax=<gdiplus.gdipnewpriv .text:004199cd="" lockbit.ex<="" pre=""></gdiplus.gdipnewpriv></pre>	<pre>steFontCollection> (60 se:\$199CD #18DCD</pre>	681CB0)		1: [esp] 0019EC6C 2: [esp+4] 5B857373 3: [esp+8] 0019FA34 4: [esp+C] 00000780		
Million a la		and a Managara (4) a cl	0019E2B4 0019	EC 6C		

Figure 180

The malicious process adds a memory font to the private font collection:



00419CD 00419CD 00419CD 00419CE 00419CE 00419CE	A FF 75 E8 D 88 75 E0 FF 75 E4 3 FF 76 04	<pre>push dword ptr ss:[ebp-18] mov esi,dword ptr ss:[ebp-20] push dword ptr ss:[ebp-1C] push dword ptr ds:[ebi+4]</pre>	push dword ptr si ebp-180 mov esi, dword ptr si ebp-120 push dword ptr si ebp-140 push dword ptr si ebp-140		
eax= <gdiplus.gdipprivate< th=""><th>G FF D0 AddMemoryFont> (6C682830)</th><th>call eax</th><th> eax:Gd1pPi ↓ ></th><th>Default (stdcal) 1: [esp] 05051820 2: [esp+4] 04410000 3: [esp+6] 00005798 4: [esp+C] 58857373</th><th>▼ S 🗘 🗆 Unloc</th></gdiplus.gdipprivate<>	G FF D0 AddMemoryFont> (6C682830)	call eax	eax:Gd1pPi ↓ >	Default (stdcal) 1: [esp] 05051820 2: [esp+4] 04410000 3: [esp+6] 00005798 4: [esp+C] 58857373	▼ S 🗘 🗆 Unloc
.text.00415ceb Tockort.te	EXE. \$150E0 #150E0				
Dump 1 Dump 2	Dump 3 Dump 4	Dump 5 👹 Watch 1 🕼 Locals 🖉 Struct	0019E29C 05D5 0019E2A0 0441	1820	
Address Hex 04410000 00 01 00 00 00	11 01 00 00 04 00 10 47 4	ASCII 45 46GDEF	▲ 0019E2A4 0000 0019E2A8 5BB5	5798 7373	

The GdipGetImageGraphicsContext function is used to create a Graphics object that is associated with an image object:

	00418880 51 0041888E 88 40 08 004188CI FF 71 04			push dword ptr ss:[ebp+3] push dword ptr ds:[ecx+4]					x875w_C1 0 x875w_C0 0 x875w_E5 0 x875w_SF 0 x875w_P 0 x875w_U 0			
ETP	00418	EGA FF	D0		call eax.			eax:Gd1p		efault (stdcall) : [esp] 05D51660	▼ 5 🗢 🗆 Unlock	
.text:004	188C4 lockbit	.exe:\$188C4	#17CC4	667430)					234	: [esp+4] 0019E2CC : [esp+8] 58857373 : [esp+C] 0019FA34		
Ump 1	Ump 2	Ump 3	Ump 4	Ump 5	👹 Watch 1	Ix=I Locals	Struct	0019E2A8 0019E2AC	05D516	60 CC		
Address 05D51660	Hex 78 14 61 6C 3	1 49 6D 67	00 00 00 00	01 00 00 00	ASCII X.aliImq			▲ 0019E280 0019E284	588573 0019FA	73		

Figure 182

The malware creates multiple SolidBrush objects based on different colors using the GdipCreateSolidFill routine:

	<pre>00416 00416</pre>	ESD 51 ESE FF 7	75 E8		push ecx push dword p	otr ss: [ebp	-18		x875W_SF 0 x875W_P 0 x8	875W_U 0
STP.	>• 00416 <	3931 FF (0	1	call eax			eax:GdipCi ~	Default (stdcall)	▼ 5 ¢ Unlod
eax= <gdiplu< th=""><th>s.GdipCreat</th><th>eSolidFill></th><th>(6C669370) #16291</th><th></th><th></th><th></th><th></th><th>_</th><th>1: [c5p14] 0019E2D0 3: [c5p+8] 00000780 4: [c5p+6] 0019FA34</th><th></th></gdiplu<>	s.GdipCreat	eSolidFill>	(6C669370) #16291					_	1: [c5p14] 0019E2D0 3: [c5p+8] 00000780 4: [c5p+6] 0019FA34	
Ump 1	Dump 2	Ump 3	Ump 4	🚛 Dump 5	👹 Watch 1	Ixel Locals	2 Struct	0019E2A8 FFF8 0019E2AC 0019	8F8F8 9E2D0	

Figure 183

All SolidBrush objects are used to fill the interior of multiple rectangles using GdipFillRectangle. The GdipSetPageUnit API is utilized to set the unit of measure for a Graphics object:

 0041SBCD FF 75 08 0041SBD0 FF 36 				push dword p push dword p	otr ss: ebp otr ds: [esi	0+8]	x87SW_SF 0 x87SW_P 0 x87SW_U 0			
	* 00418: <	D2 FF (00		call eax			eax:GdipS ~	Default (stdcall)	▼ 5 C Unlock
eax= <gdiplu< td=""><td>us.GdipSetPag</td><td>geUnit> (6C</td><td>67ABB0) #17FD2</td><td></td><td></td><td></td><td></td><td></td><td>2: [esp+4] 00000002 3: [esp+8] 000002 4: [esp+C] 0019F7D8</td><td></td></gdiplu<>	us.GdipSetPag	geUnit> (6C	67ABB0) #17FD2						2: [esp+4] 00000002 3: [esp+8] 000002 4: [esp+C] 0019F7D8	
Ump 1	Ump 2	Ump 3	Dump 4	Dump 5	🛞 Watch 1	(x=) Locals	Struct	0019E18B 0505 0019E18C 0000	593A8 00002	

Figure 184

GdipCreatePen1 is used to create a Pen object:





LockBit creates a GraphicsPath object via a function call to GdipCreatePath:



00417DDD 56 00417DDE 6A 00					push esi push 0					x875W_SF 0 x875W_P 0 x875W_U 0		
	>• 0043170 <	EO FF I	20		call eax			eax:GdipCi ~	Det	fault (stdcall)	▼ 5 C Unlod	
eax= <gdipl< th=""><th>7DE0 lockbit</th><th>ePath> (6C6</th><th>#171E0</th><th></th><th></th><th></th><th></th><th></th><th>2:3:4:</th><th>[esp+4] 0019E218 [esp+8] 000002BD [esp+C] 0019F7D8</th><th></th></gdipl<>	7DE0 lockbit	ePath> (6C6	#171E0						2:3:4:	[esp+4] 0019E218 [esp+8] 000002BD [esp+C] 0019F7D8		
Ump 1	Ump 2	Ump 3	Dump 4	Sump 5	👹 Watch 1	[x=] Locals	2 Struct	0019E1BS 0000 0019E1BC 0019	0000 0E21	8		

The process performs multiple GdipAddPathArcl calls in order to add elliptical arcs to the current figure of the path:

004183EA FF 75 14 004183E0 FF 75 10 004183F0 FF 75 10 004183F0 FF 75 00 004183F3 FF 75 08 004183F6 FF 36	push dword ptr ss: cbp+14 push dword ptr ss: cbp+20 push dword ptr ss: cbp+20 push dword ptr ss: cbp+6 push dword ptr ds: [cbp+6 push dword ptr ds: [cs]]	X875tatusWord 4000 x875W_B 0 x875W_C3 1 x875W_C2 0 x875W_L1 0 x875W_C0 0 x875W_E5 0 x875W_5F 0 x875W_P 0 x875W_U 0
EIC 004183F8 FF DD (c c c c c c c c c c c c c c c c c c c	call eax eaxidip	Image: Control of the state of th
Oump 1 Oump 2 Oump 3 Oump 4 Oump 5 Address Hex 05722F40 E0125 1 50 74 68 00	Image: Watch 1 Ix=lLocals Struct 0019E190 0019E190	06722F40 FFFFF80 00000280 0000014

Figure 187

The ransomware performs function calls such as GdipFillPath and GdipDrawPath in order to transform the path. It creates a FontFamily object based on the Proxima Nova Font family:

	0041967D 56 0041967E FF 75 F4 00419681 FF 75 08		9 9 9	push esi push dword ptr ss: ebp-C push dword ptr ss: ebp+8			[ebp+8]:L	x875w_C1 0 x875w_C0 0 x875w_E5 0 x875w_SF 0 x875w_P 0 x875w_U 0				
EIP	* 00419 * <	84 FF I	DO	c	alleax			eax:GdipCi ~	Default (stdca	l) 019F778 L"Prox	(ima Nova"	▼ 5 🗘 🗌 Unlod
.text:0041	9E84 lockbit	eFontFamily	#19284	C64C800)					2: [esp+4] 3: [esp+8] 4: [esp+C]	05D51820 0019E280 00000780		
Ump 1	Ump 2	Dump 3	Dump 4	Dump 5	👹 Watch 1	x= Locals	2 Struct	0019E25C 0019 0019E260 05D5	9F778 L"Pro 51820	xima Nova"		
Address H	ex				ASCTT	1		_ 0019E264 0019	9E2B0			

Figure 188

A Font object is created based on the above object via GdipCreateFont:

	0041366 56 0041367 6A 02 0041367 6A 02 0041365 1F 75 0C 0041366 51 0041386 51 00413486 51 00413486 00413487 FP 07 11 04 24 00413487 FP 07 15 F8			push esi push dword ptr ss:[ebp+C] push dword ptr ss:[esp],xmm0 push dword ptr ss:[esp],xmm0 push dword ptr ss:[ebp-6]			x87Statusword 4000 x87SwL8 0 x87SwLC3 1 x87SwLC2 0 x87SwLC1 0 x87SwLC0 0 x87SwLES 0 x87SwLSF 0 x87SwLP 0 x87SwLU 0				
eax= <gdiplu< th=""><th><pre>> OOLLABZS </pre> <pre> Collabasz </pre> <pre> Collabasz </pre> <pre> Collabasz </pre> </th><th>FF D0 (6C67A640) \$1A375 #19775</th><th>Call, eax</th><th></th><th></th><th>4</th><th>eax:GdipCi v</th><th>Default (stdcall) 1: [esp] 05D59258 2: [esp+4] 41C00000 3: [esp+8] 00000004 4: [esp+C] 00000002</th><th></th><th>▼ 5 \$ □ 0</th><th>nlod</th></gdiplu<>	<pre>> OOLLABZS </pre> <pre> Collabasz </pre> <pre> Collabasz </pre> <pre> Collabasz </pre>	FF D0 (6C67A640) \$1A375 #19775	Call, eax			4	eax:GdipCi v	Default (stdcall) 1: [esp] 05D59258 2: [esp+4] 41C00000 3: [esp+8] 00000004 4: [esp+C] 00000002		▼ 5 \$ □ 0	nlod
Dump 1	Ump 2 Ump 2	0ump 3 👹 Dump 4	Dump 5 👹 Watch 1	x= Locals	🐉 Struct		19E28C 05D 19E290 41C 19E294 000	\$9258 00000 000004			
0019F778 50	00 72 00 6F 00	78 00 69 00 6D 00	61 00 20 00 E.r.o.x.1.	m.a		000	19E298 000 19E29C 001	00002 9E39C			

Figure 189

The GdipDrawImageRect function is utilized to draw an image:

00419836 FF 75 F4 00419839 FF 36					push dword p push dword p	tr ss: ebp	-c]		x875W_SF 0 x875W_P 0	x875W_SF 0 x875W_P 0 x875W_U 0		
ETP	COLDEDED FF DO COLDEDED FF DO COLDEDED FF DO COLDEDED COLDEDED FF DO COLDEDED COLDED COLDEDED COLDED COLDEDED COLDED COLDEDED							eax:Gd1pD) v	d100 ✓ > Default (stdcall) ▼ 5 \$ 1: [esp] 05D5 93A8 5 \$			
.text:0041	9838 lockbit	.exe:\$19838	#18C3B						2: [esp+4] 05D53198 3: [esp+8] 4413D99A 4: [esp+C] 42480000			
Dump 1	Ump 2	Dump 3	Ump 4	Ump 5	💮 Watch 1	(x=) Locals	Struct	0019E298 05D5 0019E29C 05D5	593A8 53198			

Figure 190

The malware measures the extent of the strings that will appear in the wallpaper by calling the GdipMeasureString API:



Od1961D Od1961D Od19621 GA O0 Od19621 GA Od19621 GA Od1962 GA Od1962 GA Od1962 GA Od1962 GA F7 GA Od1962 GA F7 GA Od1963 F7 GA Od1963 F7 GA				push 0 push 0 push 0 push dword ptr ss:[ebp+18] push dword ptr ss:[ebp+14] push dword ptr ss:[ebp+0] push 0 push 0 push 0 push 0 push 0 push 0 push 0 push 0 push dword ptr ss:[ebp+0] push dword p				x87TW_4 3 (Empty) x87TW_5 3 (Empty) x87TW_6 3 (Empty) x87TW_7 3 (Empty) x87TSW_6 8 (Empty) x87TW_7 3 (Empty) x87SSW_8 0 x87SW_7 0100 x87SW_6 0 x87SW_7 0 x87SW_7 0 x87SW_5 0 x87SW_7 0 x87SW_8 0 x87SW_5 0 x87SW_9 0 x87SW_9 0			
eax= <gdiplus< th=""><th>.GdipMeasureSt</th><th>ring> (6C679990)</th><th>)</th><th>cax.</th><th></th><th>(cax, 30.1</th><th>> Def 1: 2: 3:</th><th>ault (stdcal) [esp] 05D593A8 [esp+4] 0019F62C [esp+8] FFFFFFF</th><th>▪ L"All your files</th><th>5 C Unlock</th></gdiplus<>	.GdipMeasureSt	ring> (6C679990))	cax.		(cax, 30.1	> Def 1: 2: 3:	ault (stdcal) [esp] 05D593A8 [esp+4] 0019F62C [esp+8] FFFFFFF	▪ L"All your files	5 C Unlock	
.text:004196	33 lockbit.exe	:\$19633 #18A33					4:	[esp+C]_OSDSEEEO		_	
US Dump 1	Ump 2	Dump 3 🛛 Dump	0 4 🛛 🖓 Dump 5 👹 V	atch 1 x= Locals	s 🦻 Struct	0019E284 0019E288 0019E286	05D593A	L"All your file	s stolen and encr	ypted"	
Address Hex 0019F62C 41 0019F63C 20 0019F64C 74 0019F65C 64 0019F66C 74	00 6C 00 6C 00 00 66 00 69 00 00 6F 00 6C 00 00 20 00 65 00 00 65 00 64 00	20 00 79 00 6F 0 6C 00 65 00 73 0 65 00 6E 00 20 0 6E 00 66 00 72 0 0 00 66 00 6F 0 0 00 66 00 6F	ASCIJ 00 75 00 72 00 A.1.1 00 20 00 73 00			O019E290 O019E294 O019E298 O019E29C O019E29C O019E2A0 O019E2A4	05D5EEE 0019EC74 00000000 0019E336 00000000 00000000				

Figure 191

The process draws the strings based on a font, a layout rectangle, and a format via a call to GdipDrawString:

0041946F FF 75 F6 00419472 88 75 58 00419473 80 40 D6 00419475 80 40 D6 00419475 64 00 00419478 5F 75 F4 0041947E 64 FF 0041947E 64 FF 00419485 FF 75 08	push dword ptr ss: [ebp-8] mov esi, dword ptr ss: [ebp-18] lea ecx, dword ptr ss: [ebp-28] push dword ptr ss: [ebp-6] push dword ptr ss: [ebp-8] push dword ptr ss: [ebp-8] push dword ptr ss: [ebp-8]	[ebp+8]:L'	x87TW_4 3 (Empty) x87TW_5 3 (Empty) x87TW_6 3 (Empty) x87TW_7 3 (Empty) x87StatusWord 0100 x87SW_C 2 0 x87SW_B 0 x87SW_C 3 0 x87SW_C 2 0 x87SW_C 2 0 x87SW_C 10 x87SW_C 0 1 x87SW_C 5 0 x87SW_C 1 0 x87SW_S 0 x87SW_C 0 1 x87SW_C 1 0 x87SW_C 1 0		
eax= <gdiplus.gdipdrawstring> (6C679B00) .text:00419485 lockbit.exe:\$19485 #18885</gdiplus.gdipdrawstring>		>	Default (stdcal) 1: [esp] 05D593A8 2: [esp+4] 0019F62C L" 3: [esp+8] FFFFFFF 4: [esp+C] 05D5EEE0	▼ 5 C Unlook All your files stolen and er	
Homp 1 Homp 2 Homp 3 Homp 4 Homp 4 Address Hex Dump 2 Homp 3 Homp 4 Homp 4 0019F52C 143 00 6 00 65 00 75 00 20 00 60 00	Dump 5 😻 Watch 1 I= Locals 🦻 Struct ASCII 10 72 00 8.1.1	0019E27C 0505 0019E280 0011 0019E284 FFFF 0019E288 0505 0019E282 0019 0019E290 0000 0019E290 00672	93A8 FF62C L"All your files : FFFF SEEE0 922A4 00000 2F3D0	stolen and encrypted"	

Figure 192

The file extracts the path of the %TEMP% directory:

00487854 51 push ecx 00487855 68 04 01 00 00 push 104					x875W_SF 0 x875W_P 0 x875W_U 0					
EIP	0048785A	FF D	0		call eax			eax:GetTe v	Default (stdcall)	▼ 5 C Unlod
eax= <kerne< th=""><th>132.GetTempPath 785A lockbit.ex</th><th>W> (76ACE)</th><th>0C0) #B6C5A</th><th></th><th></th><th></th><th></th><th></th><th>2: esp+4 0019F800 3: esp+8 043F0000 4: [esp+C] 00223000</th><th></th></kerne<>	132.GetTempPath 785A lockbit.ex	W> (76ACE)	0C0) #B6C5A						2: esp+4 0019F800 3: esp+8 043F0000 4: [esp+C] 00223000	
Dump 1	Dump 2	Dump 3	🖏 Dump 4	🖏 Dump 5	👹 Watch 1	(x=) Locals	2 Struct	0019F7E8 0000 0019F7EC 0019	00104 0F800	

Figure 193

GetTempFileNameW is utilized to create a temporary file:

0048798E FF 75 F0 00487991 80 80 10 FD 00487991 6A 00 00487999 6A 00 00487999 6A 00 00487999 6A 00			FF push dword p lea ecx, dwor push 0 push 0 push ecx	otr ss:[ebp-10] d ptr ss:[ebp-2F0]	ecx:L"C:\'	x875%14U\$W070 4000 x875w_60 x875w_63 1 x875w_62 0 x875w_61 0 x875w_60 0 x875w_65 0 x875w_5F 0 x875w_P 0 x875w_U 0		
eax= <kerne< th=""><th>0048799C</th><th>FF D0 amew> (76ACE0A0)</th><th>call eax</th><th></th><th>eax:GetTei v</th><th>Default (stdcall) 1: [esp] 0019F800 L"C: 2: [esp+4] 00000000</th><th>S Unlod Unlod Unlod Users Unlod Users</th></kerne<>	0048799C	FF D0 amew> (76ACE0A0)	call eax		eax:GetTei v	Default (stdcall) 1: [esp] 0019F800 L"C: 2: [esp+4] 00000000	S Unlod Unlod Unlod Users Unlod Users	
.text:004B	799C lockbit.exe	\$8799C #86D9C				3: [esp+8] 00000000 4: [esp+C] 068C0000		
Dump 1	Ump 2	Dump 3 Ump 4	Dump 5 💮 Watch 1	Ix=I Locals 🛛 🎾 Struct	0019F7E4 0000	00000	(AppData\\Local\\Temp\\"	
Address H	ex		ASCII		▲ 0019F7E8 0000 0019F7EC 0680	00000		

Figure 194

The GdipGetImageEncoders function is used to retrieve an array of ImageCodecInfo objects containing information about the available image encoders:

Dump 1	Dump 2	Ump 3	Dump 4	Dump 5	Watch 1	[x=] Locals	W Struct	0019F7B0 0019F7B4	00000	0410	
eax= <gdiplus.gdipgetimageencoders> (6C648810) .text:00418323 lockbit.exe;\$18323 #14723</gdiplus.gdipgetimageencoders>					0019574	00000	2: [esp+4] 00000410 3: [esp+8] 068D0000 4: [esp+C] 30916789				
OO41831A FF 75 08 push dword ptr ss: [ebp-3] OO41832D FF 75 E8 push dword ptr ss: [ebp-3] OO41832D FF 75 E4 push dword ptr ss: [ebp-3] OO41828 FF D0 call eax						eax:Gd1	25 ~ >	x875W_C1 0 x875W_C0 0 x875W_ x875W_5F 0 x875W_P 0 x875W_ Default (stdcall)	ES 0 U S ◯ Unloc		





The image constructed in memory is saved to the disk in the temporary file created earlier:



Figure 196

Figure 197 shows the wallpaper that will be set:

ALL YOUR IMPORTANT FILES ARE STOLEN AND ENCRYPTED!	
All your files stolen and encrypted for more information see RESTORE. M-VH_LES. TOT that is located in every encrypted folder. Our company acquere access to related in every encrypted folder. The stole and even access to relate the scales to a stolen stolen and the stolen you shall be need related as a stolen acquere. While the scales to an encryption of addition? Our company acquere access to relate the scales to any company. For example, togran acqueres access to the scale of acqueres. The stole acquere access to the scales to any company. The scales the scale of the scale previous of	

Figure 197

The RegOpenKeyA API is utilized to open the "Control Panel\Desktop" registry key (0x80000001 = **HKEY_CURRENT_USER**):



Figure 198

The "WallpaperStyle" registry value is set to 2, and the "TileWallpaper" value is set to 0 by calling the RegSetValueExA routine (0x1 = **REG_SZ**):

 00488225 00488226 80.45 DC 00488229 00488224 6A 01 00488226 6A 02 00488226 6A 05 00488231 50 00488232 57 5 94 	push eax lea eax,dword ptr ss:[ebp-24] push 1 push 1 lea eax,dword ptr ss:[ebp-64] push dword ptr ss:[ebp-64]	eax: "Wall; eax: "Wall; eax: "Wall;	A0rm
ecx= <advapi32.regsetvalueexa> (73A60F60)</advapi32.regsetvalueexa>	(all ex	lecxikegse *	Default (stdcal)
2 Dump 1 2 Dump 2 2 Dump 3 2 Dump 4 2 Dump	😽 🛞 Watch 1 🛛 Ix=l Locals 🛛 🖉 Struct	0019F7D8 0000 0019F7DC 0019	04AC FA8C "WallpaperStyle"
Address Hex 004B8648 88 55 FC 88 42 24 80 04 48 0F 87 0C 18 88 42 004B8658 80 04 88 88 04 18 03 C3 E9 55 FA FF FF 88 45 004B8658 80 04 88 88 04 18 03 C3 E9 55 FA FF FF 88 45	ASCII IC UUU.85.H8. F4	0019F7E0 0000 0019F7E4 0000 0019F7E8 0019 0019F7EC 0000	0000 0001 FACC 0002



0048338 0048338 0048338 0048338 0048338 0048338 0048338 0048339 0048 004839 004839 00483 004839 0048 0048 004839 004			push eax lea eax,dword pt push eax push 1 push 0 lea eax,dword pt push eax push dword ptr s	r ss:[ebp-28] r ss:[ebp-36] s:[ebp-66]	eax: "Tile eax: "Tile eax: "Tile	e Asrm_" > (cmpty) Asrm_7 & (cmpty) e x87m_6 & (cmpty) x87m_7 & (cmpty) x87st_1 & (cmpty) x87m_7 & (cmpty) x87sm_B & x87sm_C3 & x87sm_C2 & x		
(10)	00488399	FF D1	call ecx		ecx:Reqse v	Default (stdcall)	👻 💈 💭 Unlod	
ecx= <adva< td=""><td>ap132.RegSetValueEx</td><td>A> (73A60F60) \$88399 #87799</td><td></td><td></td><td></td><td>1: [esp] 000004AC 2: [esp+4] 0019FABA "Til 3: [esp+8] 00000000 4: [esp+C] 00000001</td><td>ewallpaper"</td></adva<>	ap132.RegSetValueEx	A> (73A60F60) \$88399 #87799				1: [esp] 000004AC 2: [esp+4] 0019FABA "Til 3: [esp+8] 00000000 4: [esp+C] 00000001	ewallpaper"	
St Dump 1	1 Uli Dump 2 Uli C	Dump 3 🙀 Dump 4	🕮 Dump 5 👹 Watch 1 💷	Locals 🖉 Struct	0019F7DS 0000 0019F7DC 0019	004AC 9FABA "TileWallpaper"		
Address 0019FAC8 0019FAD8	Hex 30 00 30 06 32 00 BC FB 19 00 98 07	00 00 A0 94 A2 68 AA 73 00 00 SC 06	ASCII 80 07 00 00 0.0.2¢k 8A 02 00 00 40*s	:	0019F7E0 0000 0019F7E4 0000 0019F7E8 0019 0019F7EC 0000	00000 00001 0FAC8 00002		

The Desktop wallpaper is set by calling the SystemParametersInfoW function (0x14 = **SPI_SETDESKWALLPAPER**, 0x3 = **SPIF_UPDATEINIFILE** | **SPIF_SENDCHANGE**):

Address H	ex	00.55.0017	2 00 55 001	72.00.77.00	ASCII			^	0019F7E8 06 0019F7EC 00	SC 0000 000003	0 L"C:\\Users\\	opData\\Local\\	Temp\\8397.
Ump 1	Ump 2	Dump 3	Dump 4	Dump 5	🥮 Watch 1	[x=] Locals	2 Struct		0019F7E0 00 0019F7E4 00	000014	4		
eax= <user32< th=""><th>2.SystemParame 8627 lockbit.e</th><th>tersInfow> xe:\$88627</th><th>(75D09EE0) #87A27</th><th></th><th></th><th></th><th></th><th></th><th></th><th>1: 2: 3: 4:</th><th>[esp] 000000014 [esp+4] 00000000 [esp+8] 068C0000 L"C: [esp+C] 00000003</th><th>\\Users\\</th><th>AppData\\Lo</th></user32<>	2.SystemParame 8627 lockbit.e	tersInfow> xe:\$88627	(75D09EE0) #87A27							1: 2: 3: 4:	[esp] 000000014 [esp+4] 00000000 [esp+8] 068C0000 L"C: [esp+C] 00000003	\\Users\\	AppData\\Lo
EIP	→• 0048862 <	FF D	9		call eax	· · · · · · · · ·			eax:Syste	♥ Defi	ault (stdcall)	▼ 5	i 🗘 🗌 Unlod
00486415 GA 03 puth 3 00486420 FF 75 FD puth dword ptr ss:[ebp-10] 00485423 GA 00 puth 0 00488625 GA 14 puth 14					[ebp-10]:		x8 x8 x8	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					

Figure 201

0

As we can see in the next picture, the registry values were successfully modified:

aD	lileWallpaper
011	TranscodedImageCache
010 110	TranscodedImageCount
210	UserPreferencesMask
ab	WallPaper
011	WallpaperOriginX
210	WallpaperOriginY
ab	WallpaperStyle

REG	SZ
REG	BINARY
REG	DWORD
REG	BINARY
REG	SZ
REG	DWORD
REG	DWORD
REG	SZ

a c3 01 00 36 90 7e 00 80 07 00 00 38 04 00 00 33 b1 88 97 26 1c d8 01 43 00 3a 00 5c 00 55 00 73 00 65 00 72
x00000001 (1)
e 1e 07 80 12 00 00 00
:\Users\\AppData\Local\Temp\B397.tmp.bmp
x00000000 (0)
x0000000 (0)

Figure 202

Extract and save the HTA ransom note to Desktop

LockBit sends the "[+] Extract *.hta file" message to the hidden window. The HTA ransom note is stored in an encrypted form in the executable. It is decrypted using the XOR operator (key = 0x38).

The malicious binary creates a file called "LockBit_Ransomware.hta" on the user Desktop (0x40000000 = **GENERIC_WRITE**, 0x2 = **CREATE_ALWAYS**, 0x80 = **FILE_ATTRIBUTE_NORMAL**):

 00495377 6A 00 00495375 6B 80 00 00 00 00495376 6A 02 00495380 6A 00 00495385 6A 00 00495385 6A 00 00495385 6B 85 C8 7F FF 00495385 00495385 005 05 05 7F FF 00495385 00495385 00495385 00495385 00495385 00495385 00495385 64 4 00 7F FF 	push 0 push 0 push 2 push 2 push 4000000 push 4000000 push 4000000 push 400000 push 4000000 push 40000000 push 400000000 push 400000000 push 400000000 push 4000000000000000000000000000000000000	eax:Creat. eax:Creat. eax:Creat.	X87TW_0 3 (Empty) X87TW_1 3 (Empty) X87TW_2 3 (Empty) X87TW_3 3 (Empty) X87TW_4 3 (Empty) X87TW_3 3 (Empty) X87TW_6 3 (Empty) X87TW_7 3 (Empty) X87TW_6 3 (Empty) X87TW_7 3 (Empty) X875Tw128Word 0000 X875W_1 0 X875W_C 0 X875W_C 0 X875W_5 0 X875W_C 0 X875W_E 0 X875W_5 0 X875W_P 0 X875W_U 0
E312 → 0049539C FF D0	call eax	eax:Creativ	Default (stdcall)
eax= <kernel32.createfilew> (76ACDDE0)</kernel32.createfilew>			1: [esp] 001981FC L"C:\\Users\\ 2: [esp+4] 4000000 3: [esp+4] 0000000 4: [esp+C] 00000000
Pump 1 Dump 2 Dump 3 Dump 4 Dump 5	Watch 1 X=LLocals Struct	00198178 001981FC	L"C:\\Users\\\Desktop\\LockBit_Ransomware.hta"
Address Hex 001981A0 (6F) 00 63 00 66 00 42 00 69 00 74 00 5F 00 52 00 001981B0 (61 00 65 00 72 00 6F 00 60 00 77 00 61 00 72 00 001981B0 (50 00 22 00 69 00 74 00 61 00 00 00 53 66 65 69	ASCII 19.c.k.s.i.t.,.R. a.n.s.o.m.w.a.r. eh.t.aShel	↑ 00198180 0000000 00198184 00000000 00198184 00000000 00198185 00000020 0019818C 00000080 00198190 00000000	



The WriteFile API is used to populate the HTA file:

312	00495380 00495380 00495380 00495387 00495387 00495387 00495387 00495387 00495387 00495387 00495363 00495363 00495363 00495363 00495363 00495363 00495365 00495365 00495365	6A 00 6D 8D 28 89 FF FF 51 88 4D 9C 88 4D 9C 88 51 9C 89 55 9C 88 55 9C 88 55 9C 88 65 A0 50 50 58 58 F7 FF 88 56 50 56 51 F7 FF 88 56 50 51 F7 FF 89 50 50 50 50 50 50 50 50 50 50 50 50 50	<pre>push 0 lea ecx,dword ptr ss:[ebp-7608] push ecx mov ecx,dword ptr ss:[ebp-64] Eall lockbit,48A2E0 push eax outh edx mov eax,dword ptr ss:[ebp-64] push eax call lockbit,410F90 mov ecx,eax call lockbit,4115A0 call eax</pre>	eax:writel eax:writel eax:writel eax:writel eax:writel eax:writel >	x87Tagword FFFF x87Tw_0 3 (Empty) x87Tw_1 3 (Empty) x87Tw_2 3 (Empty) x87Tw_3 3 (Empty) x87Tw_4 3 (Empty) x87Tw_5 3 (Empty) x87Tw_5 3 (Empty) x87Tw_7 3 (Empty) x875tatusword 0000 x875w_B 0 x875w_C3 0 x875w_C2 0 x875w_SP 0 x875w_C9 0 x875w_U 0 Default (tdtcal) ↓ 5 ↓ Unlock
.text:0049	53D4 lockbit.exe:	\$953D4 #947D4			2: [esp+4] 068C0000 3: [esp+8] 0000B909 4: [esp+C] 00198418
Dump 1	Dump 2	Dump 3 👹 Dump 4 👹 Du	mp 5 👹 Watch 1 🛛 🖉 Struct	00198180 000	004AC C0000
Address H 068C0000 3 068C0010 6	ex C 68 74 60 6C 3E 1 20 68 74 74 70	3C 68 65 61 64 3E <u>3C 60</u> 2D 65 <u>71 75 69 76</u> 3D 22	ASCII <u>65 74</u> khtml> <head><met 43 6F a http-equiv="Co</met </head>	00198188 000 0019818C 001 00198190 000	08909 98418 00000

Figure 204

The ZwCreateKey API is utilized to open the "HKCR\.lockbit" registry key (0x2000000 = **MAXIMUM_ALLOWED**):

312	 004ADB13 004ADB15 004ADB15 004ADB15 004ADB17 004ADB19 004ADB19 004ADB26 004ADB26<	6A, 00 6A, 00 6A, 00 8D 85 D8 FD FF FF C7 45 E4 18 00 00 00 83 45 E4 64 00 80 45 E4 65 00 80 70 E8 68 00 00 02 50 C7 45 F0 40 00 00 89 70 F4 50 F6 FF FF D0 89 70 F6 FF FF 00 80 76 FF	push 0 push 0 lea eax,dword ptr ss: ebp-228 mov dword ptr ss: ebp-14, eax lea eax,dword ptr ss: ebp-14, eax lea eax,dword ptr ss: ebp-12 lea eax,dword ptr ss: ebp-12 mov dword ptr ss: ebp-13, ed1 push 200000 push eax mov dword ptr ss: ebp-10, 40 mov dword ptr ss: ebp-1, ed1 mov dword ptr ss: ebp-1, ed1 mov dword ptr ss: ebp-1, ed1 mov dword ptr ss: ebp-1, ed1	eax:ZwCre eax:ZwCre eax:ZwCre eax:ZwCre eax:ZwCre eax:ZwCre 40:'e'	x87r4 3FFF800000000 x87r5 4005C00000000 x87r7 4005C00000000 x87r7 4005C00000000 x8777 4005C0000000000000000000000000000000000	0000000 ST4 Empty 1.000000000 000000 ST5 Empty 96.00000000 000000 ST6 Empty 96.00000000 000000 ST7 Empty 96.000000000 x87TW_13 (Empty) x87TW_53 (Empty) x87TW_53 (Empty) x87TW_53 (Empty) x87TW_55 (Empty) x87TW_55 (Empty) x87TW_55 (Empty)
	• <			· · · · · · · · · · · · · · · · · · ·	Default (stdcall)	▼ 5 ¢ Unlock
eax= <ntd11< td=""><td>.ZwCreateKey> (77 D84D lockbit.exe:</td><td>04E950) \$AD84D #ACC4D</td><td></td><td></td><td>2: [esp+4] 02000000 3: [esp+8] 00198170 4: [esp+C] 0000000</td><td></td></ntd11<>	.ZwCreateKey> (77 D84D lockbit.exe:	04E950) \$AD84D #ACC4D			2: [esp+4] 02000000 3: [esp+8] 00198170 4: [esp+C] 0000000	
Ump 1	1 Dump 2	Dump 3 👹 Dump 4 👹 Dump	5 🥮 Watch 1 🛛 🕬 Locals 🎾 Struct	00197590 001 00197594 020	9816C 00000	
Address H 0019768C 4 001976CC 7 001976DC 6	ex 3 00 3A 00 5C 00 7 00 73 00 5C 00 D 00 33 00 32 00	77 00 69 00 6E 00 64 00 6F 73 00 79 00 73 00 74 00 65 5C 00 <u>32 00 45 00</u> 44 00 38	ASCII 00 0	00197598 001 0019759C 000 001975A0 000 001975A0 000 001975A4 000 001975A8 000	98170 00000 00000 00000 00000	

Figure 205

The (Default) registry value is set to "LockBit" by calling the ZwSetValueKey function (0x1 = **REG_SZ**):

	 004AD978 004AD979 004AD977 004AD975 004AD985 004AD987 004AD987 004AD988 004AD988 004AD988 	50 FF B5 34 F7 FF FF 80 85 D0 FD FF FF 6A 01 6A 00 50 FF 75 E0 E8 CE 7F F6 FF	push eax push dword ptr ss:[ebp-8CC] lea eax,dword ptr ss:[ebp-230] push 0 push 0 push dword,ptr ss:[ebp-20] GBN] lockbit.435960	eax:ZwSet' [ebp-SCC] eax:ZwSet' eax:ZwSet'	X87TW_6 3 (Empty) x87TW_7 3 (Empty) x87StatusWord 0000 x87Sw_8 0 x87Sw_C3 0 x87Sw_C2 0 x87Sw_C10 x87Sw_C0 0 x87Sw_E5 0 x87Sw_SF 0 x87Sw_F 0 x87Sw_E 0	
EIP	004AD 992	FF DO	call eax	eax:ZwSet) v	Default (stdcall) 👻 5 🗘 🗌 Unlock	
eax= <ntd11< th=""><th>.ZwSetValueKey> (</th><th>7704ED80) \$AD992 #ACD92</th><th></th><th></th><th>11 (eSD) 000004AC 21 (eSD+4) 00197F5C 31 (eSD+6) 00000000 42 (eSD+C] 00000001</th></ntd11<>	.ZwSetValueKey> (7704ED80) \$AD992 #ACD92			11 (eSD) 000004AC 21 (eSD+4) 00197F5C 31 (eSD+6) 00000000 42 (eSD+C] 00000001	
Ump 1	US Dump 2 US D	ump 3 👹 Dump 4 👹	Dump 5 👹 Watch 1 🛛 🕸 Locals 🎾 Struct	00197594 000 00197598 001	004AC 97F5C	
Address H 00197F5C 0 00197F6C 0	ex 0 00 02 00 64 81 0 00 00 00 00 00 00 0 00 00 00 00 00	19 00 56 00 58 00 A0 2 00 00 00 00 00 00 00 00	ASCII A 19 00 0 00 00 	0019759C 000 001975A0 000 001975A4 001 001975A8 000	00000 00001 97724 L"LockBit" 0000E	

Figure 206

The malware creates the "HKCR\Lockbit" registry key by calling the ZwCreateKey API (0x2000000 = **MAXIMUM_ALLOWED**):



	push 0 push 0 push 0 push 0 lea eax,dword ptr ss: ebp-228 mov dword ptr ss: ebp-10, 18 mov dword ptr ss: ebp-10, eax push 0 mov dword ptr ss: ebp-20 lea eax,dword ptr ss: ebp-20 lea eax,dword ptr ss: ebp-20 mov dword ptr ss: ebp-10, edi mov dword ptr ss: ebp-10, edi eax:ZwCr eax	x8/r4 #r+80000000000000000 314 mDTy 1.0000000000 x8/r6 4005C000000000000000005 Empty 96.000000000 x8/r6 4005C000000000000005 ST6 Empty 96.000000000 x8/r14 4005C000000000000000000000000000000000
.text:004AF1E8 lockbit.exe:\$AF1E8 #AE5E8		3: [esp+8] 00198170 4: [esp+C] 00000000
Ump 1 Ump 2 Ump 3 Ump 4 Ump 5	Watch 1 Ix=l Locals 2 Struct 00197590 00197594	0019816C 02000000
Address Hex 00137764 54 00 56 00 12 72 19 00	ASCII (0137582 (.R.e.g.1.s.t.r.) (.R.e.g.1.s.t.r.) (0137582 001375A8 001375A8	00198170 00000000 00000000 00000000 00000000

The DefaultIcon registry value is set to "C:\windows\SysWow64\2ED873.ico" using ZwSetValueKey (0x1 = **REG_SZ**):



Figure 208

The process creates the following registry subkeys: "shell", "Open", and "Command". The (Default) value is set to "LockBit Class" using ZwSetValueKey (0x1 = **REG_SZ**):

	 0048005E 50 0048005F FF B5 00480065 80 85 00480066 6A 01 00480067 6A 00 00480077 F7 75 00480073 E8 E8 	C4 FD FF FF D0 FD FF FF E0 58 F6 FF	push dword ptr ss:[ebp-23C] lea eax,dword ptr ss:[ebp-23C] push 1 push 0 push dword ptr ss:[ebp-20] call lockbit.415960	eax:ZwSet' [ebp-22] eax:ZwSet' eax:ZwSet'	X87TW_6 3 (Empty) X87TW_7 3 (Empty) X87Statusword 0000 X87SW_8 0 X87SW_C3 0 X87SW_C2 0 X87SW_5 0 X87SW_C3 0 X87SW_E5 0 X87SW_5F 0 X87SW_P 0 X87SW_U 0	
	00480078 FF D0		call eax	eax:zwset	Default (stdcall)	Unlod
eax= <ntd11.2ws< th=""><th><pre>SetValueKey> (7704ED80] S lockbit.exe:\$80078 #/</pre></th><th>) AF478</th><th></th><th></th><th>2: [esp+4] 00197F5C 3: [esp+8] 0000000 4: [esp+C] 00000001</th><th>_</th></ntd11.2ws<>	<pre>SetValueKey> (7704ED80] S lockbit.exe:\$80078 #/</pre>) AF478			2: [esp+4] 00197F5C 3: [esp+8] 0000000 4: [esp+C] 00000001	_
Dump 1	UDump 2 UU Dump 3	Dump 4 👹 Dump 5	🛞 Watch 1 🛛 🕼 Locals 🖉 Struct	00197594 00 00197598 00	00004AC 0197F5C	
Address Hex 0019815C 00 00 0019816C AC 04	0 00 00 02 00 00 00 00 0 00 00 18 00 00 00 00	00 00 00 74 81 19 00 00 00 00 54 7F 19 00	ASCII	0019759C 00 001975A0 00 001975A4 00 001975A8 00	0000000 0000001 01980E8 L"LockBit Class" 000001A	

Figure 209

The (Default) registry value under the Command key is set to open the HTA ransom note:

eax= <ntd11.< th=""><th>00480542 00480543 00480543 00480549 00480549 00480551 00480551 004805554 00480555 00480555 00480555 00480555 00480555 00480555 0048055 0048055 0048055 0048055 0048055 0048055 0048055 0048055 0048055 0048055 0048055 0048055 0048055 0048055 0048055 0048055 0048055 0048055 004805 004805 004805 004805 004</th><th>50 FF B5 C4 FD FF 80 85 D0 FD FF 6A 00 FF 75 FC E8 04 54 F6 FF FF D0 7704ED80)</th><th>FF push dwor FF lea eax,d push 0 push 0 push eax push dwor call lock call eax</th><th>i ptr ss: [eb word ptr ss: d ptr ss: [eb wit.415960</th><th>p-29C) (ebp-230) p-4)</th><th>eax:2 [ebp- eax:2 eax:2 eax:2</th><th>wset' 23C] wset' wset' 1 1 2 2 3</th><th>x8:71W_6 3 (Empty) x8 x875K_6 3 (Empty) x8 x875K_6 0 x875W_C3 x875W_C3 x875K_5 0 x875W_C4 x875W_C3 x875K_5 0 x875W_C9 x875W_20 x875K_5 0 x875W_2 x875W_21 x875W_20000480 x875W_20 x875W_20000480 x875W_20 x875W_20000480 x875W_200000480 x875W_200000000 x875W_200000000</th><th>7TW_7 3 (Emp 7TW_7 3 (Emp 0 x87SW_C2 0 x87SW_ES 0 x87SW_U</th><th>vy) 0 0 • s ≎ □ Unlock</th></ntd11.<>	00480542 00480543 00480543 00480549 00480549 00480551 00480551 004805554 00480555 00480555 00480555 00480555 00480555 00480555 0048055 0048055 0048055 0048055 0048055 0048055 0048055 0048055 0048055 0048055 0048055 0048055 0048055 0048055 0048055 0048055 0048055 0048055 004805 004805 004805 004805 004	50 FF B5 C4 FD FF 80 85 D0 FD FF 6A 00 FF 75 FC E8 04 54 F6 FF FF D0 7704ED80)	FF push dwor FF lea eax,d push 0 push 0 push eax push dwor call lock call eax	i ptr ss: [eb word ptr ss: d ptr ss: [eb wit.415960	p-29C) (ebp-230) p-4)	eax:2 [ebp- eax:2 eax:2 eax:2	wset' 23C] wset' wset' 1 1 2 2 3	x8:71W_6 3 (Empty) x8 x875K_6 3 (Empty) x8 x875K_6 0 x875W_C3 x875W_C3 x875K_5 0 x875W_C4 x875W_C3 x875K_5 0 x875W_C9 x875W_20 x875K_5 0 x875W_2 x875W_21 x875W_20000480 x875W_20 x875W_20000480 x875W_20 x875W_20000480 x875W_200000480 x875W_200000000 x875W_200000000	7TW_7 3 (Emp 7TW_7 3 (Emp 0 x87SW_C2 0 x87SW_ES 0 x87SW_U	vy) 0 0 • s ≎ □ Unlock
.text:00480	55C lockbit.exe:	\$8055C #AF95C						. [espic] 0000001		
Ump 1	10 Dump 2	Dump 3 🙀 Dump 4	🕮 Dump 5 🛛 👹 Watch	1 x= Locals	2 Struct	001975	94 00000 98 00197	480 F5C		
Address He 02587DD8 22 02587DE8 65	x 00 43 00 3A 00 00 77 00 73 00	5C 00 57 00 69 00 5C 00 73 00 79 00	ASCII 6E 00 64 00 73 00 74 00 0.w.s.\.	.1.n.d. .y.s.t.		^ 001975 001975 001975 001975	9C 00000 A0 00000 A4 025871 A8 00000	000 001 DD8 L"\"C:\\Windows\\: 09A	system32\\ms	hta.exe\" \"C:\'



File	le Edit View Favorites Help							
Com	Computer\HKEY_CLASSES_ROOT\Lockbil\shell\Open\Command							
	~ ~	Lockbit Defaultion shell Command	Name	: Type efault) REG_SZ	Data "C\\Windows\system32\mshta.exe" "C\Users\\Desktop\LockBit_Ransomware.hta"			

The NtOpenKey routine is utilized to open the "HKCR\.hta" registry key (0x2000000 = **MAXIMUM_ALLOWED**):

Addrare Uav			ACCTT	. 001975A8 001	98170
Dump 1	🗒 Dump 2 🛛 🗰 D	ump 3 🛛 🗰 Dump 4 🖉 Du	np 5 🛛 🧐 Watch 1 🛛 🕼 🖉 Struct	001975A0 001 001975A4 020	97F6C 00000
eax= <ntd11.nt< th=""><th>OpenKey> (7704)</th><th>880) 800F6 #800F6</th><th></th><th></th><th>1: [esp] 00197F6C 2: [esp+4] 0200000 3: [esp+8] 00198170 4: [esp+C] 00198254</th></ntd11.nt<>	OpenKey> (7704)	880) 800F6 #800F6			1: [esp] 00197F6C 2: [esp+4] 0200000 3: [esp+8] 00198170 4: [esp+C] 00198254
<u></u>	→ 00480CF6 <	FF DO	call eax	eax:Ntope ~	Default (stdcall) 🔹 5 🗘 🗌 Unio
	 00480CD4 00480CD5 00480CDE 00480CDE 00480CE3 00480CE4 00480CE8 00480CE8 00480CE1 	50 8D 85 E0 FD FF FF 89 7D E8 68 00 00 00 02 50 C7 45 F0 40 00 00 00 89 7D F4 89 7D F8 58 5A 40 56 55	push eax lea eax, dword ptr ss: ebp-220 mov dword ptr ss: ebp-16], ed1 push eax mov dword ptr ss: ebp-10], 40 mov dword ptr ss: ebp-2, ed1 mov dword ptr ss: ebp-3, ed1	eax:NtOpei eax:NtOpei 40:'@'	X87TW_4.8 X87TW_5.3 Lempty) X87TW_6.3 Lempty) X87TW_5.3 Lempty) X87Tw_6.3 Lempty) X87TW_7.3 Lempty) X87StatusWord 0000 X87SW_7.3 Lempty) X87SW_6.4 0 X87SW_6.2 0 X87SW_6.5 0 X87SW_6.2 0 X87SW_6.0 X87SW_6.2 0 X87SW_6.2 0 X87SW_6.5 0 X87SW_6.2 0 X87SW_6.2 0

Figure 212

The malicious binary retrieves the (Default) registry value via a function call to NtQueryValueKey (0x2 = **KeyValuePartialInformation**):

ETP	 00480006 00480007 00480007 00480012 00480013 00480013 00480018 00480018 00480018 004800122 00480022 	50 66 00 01 00 00 8D 85 30 F4 FF FF 50 6A 02 8D 85 D0 FD FF FF 58 50 FD FF FF E8 19 4E F6 FF FF D0	push eax push laa eax,dword ptr ss:[ebp-800] push 2 lea eax,dword ptr ss:[ebp-230] push eax,dword ptr ss:[ebp-220] can lockbit.43280	eax: NtQue: eax: NtQue: eax: NtQue: eax: NtQue: eax: NtQue: eax: NtQue:	X87TW_4 3 (Empty) X87TV X87TW_6 3 (Empty) X87TV X87TW_6 3 (Empty) X87TV X875W_8 0 X87SW_C3 0 X87SW_C1 0 X87SW_C3 0 X87SW_C1 0 X87SW_C 0 X87SW_SF 0 X87SW_P 0	<pre>LS 3 (Empty) L7 3 (Empty) x875W_C2 0 x875W_ES 0 x875W_U 0</pre>
	• <			>	Default (stdcall)	👻 5 🗘 🗌 Unlock
eax= <ntdll.< td=""><td>.NtQueryValueKey> 0D27 lockbit.exe:</td><td>(7704E8D0) \$B0D27 #B0127</td><td></td><td></td><td>2: [esp+4] 00197F5C 3: [esp+8] 00000002 4: [esp+C] 001975BC</td><td></td></ntdll.<>	.NtQueryValueKey> 0D27 lockbit.exe:	(7704E8D0) \$B0D27 #B0127			2: [esp+4] 00197F5C 3: [esp+8] 00000002 4: [esp+C] 001975BC	
Dump 1	Ump 2 Ump 2	oump 3 📲 Dump 4 📲 Dun	np 5 👹 Watch 1 🛛 🕬 Locals 🎾 Struct	00197594 000 00197598 001	00484 97F5C	
Address H 00197F5C 0 00197F6C 8	ex 0 00 02 00 5C 81 4 04 00 00 36 00	19 00 4E 00 50 00 <u>BC 76 1</u> 00 00 5C 00 52 00 65 00 6	ASCII 9.00	0019759C 000 001975A0 001 001975A4 000 001975A8 001	00002 975BC 00100 978CC	

Figure 213

NtOpenKey is used to open the "HKCR\htafile" key (0x2000000 = MAXIMUM_ALLOWED):

 004811ED 004811F1 004811F4 004811F4 004811F3 004811F4 004811F4 004811F4 00481207 00481207 	S0 8D 4S DC 89 70 E8 68 00 00 00 02 50 57 70 F4 89 70 F8 89 70 F8 89 70 F8 89 70 F8 89 70 F8	push eax lea eax,dword ptr ssi ebp-24 mov dword ptr ssi ebp-18, edi push eax mov dword ptr ssi ebp-0, edi mov dword ptr ssi ebp-0, edi mov dword ptr ssi ebp-0, edi	eax:NtOper eax:NtOper 40:'@'	x87TW_4 3 (Empty) x1 x87TW_6 3 (Empty) x1 x87TW_6 3 (Empty) x1 x875W_8 0 x875W_C3 x875W_5 0 x875W_C3 x875W_5 0 x875W_C	o'n= 5 (Empty) S7Tw_5 8 (Empty) S7Tw_7 3 (Empty) 0 x87SW_C2 0 0 x87SW_ES 0 0 x87SW_E 0
<pre>eax=<ntdll.ntopenkey> (7704E .text:004B120C lockbit.exe:\$</ntdll.ntopenkey></pre>	880) 8120C #8060C		,	Default (stdcall) 1: [esp] 00198168 2: [esp+4] 02000000 3: [esp+8] 00198170 4: [esp+C] 00198254	▼ 5 ¢ □ Uniod
Dump 1 Dump 2	ump 3 👹 Dump 4 👹 Dump	5 🛞 Watch 1 🛛 🕸 🖉 Struct	001975A0 001 001975A4 020	98168	
Address Hex 0019768C SC 00 52 00 65 00 6 001976C 79 00 5C 00 4D 00 6 001976C 79 00 5C 00 4D 00 6 001976C 79 00 5C 00 50 00 6 001976C 60 05 00 5C 00 50 00 6 001976C 65 00 5C 00 5C 00 00 00 0 001976C 60 05 00 5C 00 6 001976C 60 00 50 00 00 00 0	17 00 69 00 73 00 74 00 72 1 00 63 00 68 00 69 00 72 1 00 63 00 68 00 69 00 61 16 00 74 00 77 00 61 13 00 6C 00 74 00 73 00 73 8 00 74 00 61 00 73 00 73 90 00 00 00 00 00 00 00 00 00	ASCII 00 N.R.e.g.i.s.t.r. 00 Y.\M.a.C.h.i.n. 00 r.e.\.S.o.f.r.w.a. 00 r.e.\.C.i.a.s.s. 00 r.e.\	001975A8 001 001975B0 001 001975B0 001 001975B4 002 001975B8 000 001975B0 000 001975B0 000 001975C0 000	98170 98254 981CC "Shell32.dll" 23000 00000 00000 00000	

Figure 214

The DefaultIcon registry value is set to "C:\windows\SysWow64\2ED873.ico" (0x1 = **REG_SZ**):



	 004B192 004B192 004B192 004B192 004B193 004B193 004B193 004B193 004B193 004B193 004B193 	1 50 2 FF BS BC FD FF FF 80 85 D0 FD FF FF 6A 01 6A 00 50 50 5 FF 75 DC 6 E8 25 40 F6 FF	<pre>push eax push dword ptr ss:[ebp-244] lea eax,dword ptr ss:[ebp-230] push i push eax push dword ptr ss:[ebp-24] call occbit.43560</pre>	eax:ZwSet' [ebp-244] eax:ZwSet' eax:ZwSet'	Ax/Ha_* S (Empty) Ax/Ha_S S (Empty) X87TW_6 S (Empty) X87TW_7 S (Empty) X87Statusword 0000 X87SW_8 0 x87SW_C2 0 x87SW_C2 0 X87SW_C1 0 x87SW_C0 0 x87SW_C2 0 X87SW_C1 0 x87SW_C0 0 x87SW_E5 0 X87SW_SF 0 x87SW_P 0 x87SW_U 0
EIP	0048193	FF DO	call eax	eax:ZwSet! ~	Default (stdcall)
eax= <ntdll. .text:00481</ntdll. 	.ZwSetValueKey 193B lockbit.e	> (7704ED80) xe:\$81938 #B0D38 #Dump 3 #Dump 4 #Dump 4	ump 5 😻 Watch 1 🕬 Locals 🍃 Struct	00197594 000	1: [csp] 00000145: [csp+c] 0000000 4: [csp+c] 00000001
Address H 0019768C 4 001976CC 7 001976CC 3 001976CC 3 001976C 6 001977CC 6 0019771C 60	ex 3 00 3A 00 5C 0 00 33 00 5C 0 00 33 00 32 7 00 33 00 2E 5 00 73 00 5C 0 00 65 00 5C C 00 74 00 49	00 77 00 69 00 6E 00 64 00 00 73 00 79 00 73 00 74 00 00 5C 00 32 00 45 00 44 00 00 6S 00 6S 00 6F 00 <td>ASCII 65 00 (E.t.\.w.i.n.d.o. 65 00 (W.S.V.S.Z.S.C.C.E. 73 00 7.3i.c.os. 73 00 7.3i.c.os. 69 00 e.s.(h.t.a.f.i. 75 00 1.e.\.o.e.f.a.u. 00 00 1.t.c.o.n</td> <td>0019753C 0000 001975AC 0000 001975A4 0011 001975A4 0011 001975AC 0011 001975AC 0011 001975AC 0011 001975B4 0021 001975B4 0021 001975B4 0021</td> <td>00000 00001 0768C "C:\\windows\\system32\\2ED873.1co" 0035 88254 # 3000 3000 00000</td>	ASCII 65 00 (E.t.\.w.i.n.d.o. 65 00 (W.S.V.S.Z.S.C.C.E. 73 00 7.3i.c.os. 73 00 7.3i.c.os. 69 00 e.s.(h.t.a.f.i. 75 00 1.e.\.o.e.f.a.u. 00 00 1.t.c.o.n	0019753C 0000 001975AC 0000 001975A4 0011 001975A4 0011 001975AC 0011 001975AC 0011 001975AC 0011 001975B4 0021 001975B4 0021 001975B4 0021	00000 00001 0768C "C:\\windows\\system32\\2ED873.1co" 0035 88254 # 3000 3000 00000

Figure 215

The file opens the Run registry key using RegCreateKeyExW (0x80000001 = **HKEY_CURRENT_USER**, 0x2001F = **KEY_READ** | **KEY_WRITE**):

 00496054 00496055 00496055 00496056 00496056 00496056 00496056 00496050 00496050<	push eck lea eck, dword ptr ss:[ebp-24] push eck push 0 push 0 push 0 push 0 push c, dword ptr ss:[ebp-170] lea eck push eck push 8	ecx:L"SOF [ebp-24]: ecx:L"SOF 837 837 837 837 837 837 837 837 837 837	MLO 3 (Empty) x87TM_1 3 (Empty) M_2 3 (Empty) x87TM_3 3 (Empty) M_4 3 (Empty) x87TM_3 3 (Empty) M_6 3 (Empty) x87TM_7 3 (Empty) tatusword 0000 M_8 0 x87SM_50 0 x87SM_50 0 M_550 x87SM_50 0 x87SM_50 0 M_550 x87SM_50 0 x87SM_50 0 M_550 x87SM_50 0 0 M_550 x87SM_50 0 M_550 x8
EIC 00496072 FF D0 ex= <advap132.regcreatekeyexw> (73A5FE50) .text:00496072 lockblt.exe:596072 #95472</advap132.regcreatekeyexw>	call eax	Defaul	t(stdcal) ▼ 5 3 Unlock ssp1 80000001 ssp+4 00138010 L"SOFTWARE\\Microsoft\\Windows ssp+5 00000000
Ump 1 Ump 2 Ump 3 Ump 4 Ump 4 Ump 5 Address Hex 00198010 65 00 44 00 54 00 57 00 41 00 52 00 45 00 50 00 57 00 57 00 50 00 57 00 57 00 50 00 57 00 57 00 50 00 57 00 57 00 50 00 57 00 57 00 50 00 57 00 57 00 50 00 57 00 57 00 50 00 57 00 57 00 50 00 57 00	Watch 1 Isea Locals P Struct ASCII ISO.P.F.T.W.A.R.E. ISO.P.F.T.W.A.R.E. Y.M.I.C.F.P.O.S.O. Y.M.I.C.F.P.O.S.O. W.S.V.C.U.F.F.E. ISO.P.F.S.I.O.	001970C0 80000001 001970C4 00198010 001970C5 0000000 001970C5 0000000 001970D0 0000000 001970D0 0000000 001970D0 0000000 001970D0 00198168 001970E0 00198000	L"SOFTWARE\\Microsoft\\Windows\\CurrentVersid

Figure 216

The ransomware creates a value called " $\{2C5F9FCC-F266-43F6-BFD7-838DAE269E11\}$ ", which contains the path to the HTA note ($0x1 = REG_SZ$):

310	 00496EA4 00496EA5 00496EA8 00496EA8 00496EB0 00496EB2 00496EB3 00496E3 004	50 FF 75 F0 8D 85 40 FF FF FF 6A 01 50 FF 75 DC FF D1	push eax push dword ptr ss:[ebp-10] lea eax,dword ptr ss:[ebp-C0] push i push eax push dword ptr ss:[ebp-24] call ecx	eax:L"[2C] [ebp-10]:1 eax:L"[2C] [ecx:RegSe]	xe/me_bs (tempty) xe/me_rs (tempty) x8755tstUmbord 0000 k8755tstUmbord 0000 k8755tstUmbord 0000 k8755tstUmbord 0000 k8755tstUmbord 0000 x8755tstUmbord 00000 x8755tstUmbord 00000 x8755tstUmbord 00000 x8755tstUmbord 00000 x8755tstUmbord 00000 x8755tstUmbord 00000 x8755tstUmbord 00000 x8755tstUmbord 000000 x8755tstUmbord 000000000000000000000000000000000000
ecx= <adv< th=""><th>ap132.RegSetValueExw 496EB6 lockbit.exe:5</th><th>> (73A5FE80) 96EB6 #962B6</th><th></th><th></th><th>1: esp1 000004AC 2: esp41 001980CC L"{2C5F9FCC-F266-43F6-BFD7-831 3: esp43 0000000 4: esp43 00000001</th></adv<>	ap132.RegSetValueExw 496EB6 lockbit.exe:5	> (73A5FE80) 96EB6 #962B6			1: esp1 000004AC 2: esp41 001980CC L"{2C5F9FCC-F266-43F6-BFD7-831 3: esp43 0000000 4: esp43 00000001
Ump Dump	1 1 Dump 2 10 D	ump 3 📲 Dump 4 📲 Dur	mp 5 🥮 Watch 1 🛛 🕸 I Locals 🎾 Struct	00197DCC 000 00197DD0 001	004AC 980CC L"{2C5F9FCC-F266-43F6-BFD7-838DAE269E11}"
Address 001981FC 0019820C	Hex 43 00 3A 00 5C 00 5 5C 00	5 00 73 00 65 00 72 00 1 00 5C 00 44 00 65 00 7	ASCII 73 00 C.:.\.U.s.e.r.s. 73 00 \	00197DD4 000 00197DD8 000 00197DDC 001 00197DDC 001 00197DDC 000	00000 00001 981FC L"C:\\Users\\ TTT \\Desktop\\LockBit_Ransomware 00056
0019821C 0019822C 0019823C 0019824C	68 00 74 00 6F 00 7 68 00 42 00 69 00 7 73 00 6F 00 6D 00 7 68 00 74 00 61 00 0	74 00 SF 00 52 00 6F 00 7 74 00 SF 00 52 00 61 00 7 70 0 61 00 72 00 65 00 2 00 00 00 00 00 00 00 00 00 00	55 00 K.t.o.p.\.t.o.c. 5E 00 k.B.i.tR.a.n. 2E 00 S.o.m.w.a.r.e 30 00 h.t.a	00197DE4 001 00197DE8 001 00197DEC 002	98254 981CC "Shell32.dll" 23000

Figure 217

ShellExecuteW is utilized to open and display the above ransom note:

	push 1 push 0 hea edx,dword ptr ss:[ebp-76F4] push edx lea eax,dword ptr ss:[ebp-76B6] push eax push eax call lockbit.4133A0 mov ecx,eax call lockbit.4143E0 call eax	edx:Ordin. x877m_0 3 (Empty) x877m_1 3 (Empty) edx:Ordin. x877m_2 3 (Empty) x877m_3 3 (Empty) x877m_4 3 (Empty) x877m_5 3 (Empty) x877m_4 3 (Empty) x877m_5 3 (Empty) x877m_4 3 (Empty) x877m_5 3 (Empty) x877m_5 3 (Empty) x877m_5 3 (Empty) x877m_5 3 (Empty) x877m_5 3 (Empty) x877m_5 0 x875m_5 0 x875
U Dump 1 U Dump 2 U Dump 3 U Dump 4 U Dump 5	😻 Watch 1 🛛 Ix=l Locals 🛛 🖉 Struct	0019317C 00000000 00198180 00198434 L"open"
Address Hex 001981FC 43 00 3A 00 5C 00 73 00 65 00 73 00 65 00 73 00 65 00 73 00 65 00 73 00 65 00 73 00 65 00 73 00 65 00 73 00 65 00 73 00 65 00 73 00 65 00 73 00 65 00 73 00 65 00 73 00 65 00 73 00 65 00 74 00 65 00 74 00 65 00 74 00 65 00 74 00 65 00 74 00 65 00 74 00 65 00 74 00 65 00 74 00 65 00 74 00 00	ASCII 0 C.:.\U.S.C.P.S. 0 \	O0198184 001981FC L/C:\\Users\\■\\Desktop\\LockBit_Ransomwark O0198188 0000000 O0198180 00000001 O0198190 00000001 O0198190 00200001 O0198190 0022000 O0198191 O0198191 O019819 O0198







Figure 219

LockBit deletes the registry value used for persistence named "{9FD872D4-E5E5-DDC5-399C-396785BDC975}". We believe this value was created to resume the encryption process in the case of a reboot:

0049C4AE FF 75 EC 0049C481 FF 75 D8				push dword push dword	ptr ss: ebp	0-14	[ebp-14]:)	XS	x875W_SF 0 x875W_P 0 x875W_U 0			
EIP	004904	B4 FF	DO		call eax			eax:RegDe v	Def	fault (stdcall)		▼ 5 ¢ Unlock
eax= <advap< th=""><th colspan="7">eax=<advap132.regdeletevaluew> (73A61180) .text:0049C484 lockbit.exe:\$9C484 #98884</advap132.regdeletevaluew></th><th></th><th>1: 2: 3: 4:</th><th colspan="3">:: [csp+4] 06980000 L"{9FD872D4-E5E5-DDC5-399C-3 : [csp+8] 043F0000 4: [csp+C]_00223000</th></advap<>	eax= <advap132.regdeletevaluew> (73A61180) .text:0049C484 lockbit.exe:\$9C484 #98884</advap132.regdeletevaluew>								1: 2: 3: 4:	:: [csp+4] 06980000 L"{9FD872D4-E5E5-DDC5-399C-3 : [csp+8] 043F0000 4: [csp+C]_00223000		
Ump 1	Ump 2	Ump 3	💷 Dump 4	Ump 5	🛞 Watch 1	x= Locals	2 Struct	0019F720 000 0019F724 069	0004A	AC L"{9FD872D4-	ESES-DDC5-399C	-396785BDC975}"

Figure 220

The executable sends the "[+] Removed autorun key" message to the hidden window using SendMessageA. There is a call to ZwSetIoCompletion afterward:



Figure 221

The malware deletes itself when the system restarts by calling the MoveFileExW function (0x4 = **MOVEFILE_DELAY_UNTIL_REBOOT**):



	 00490 00490 00490 00490 00490 	C62 6A C64 6A C66 8B C69 FF	04 00 40 10 70 3C		push 4 push 0 mov eax,dwo push dword	rd ptr ds:[ptr ds:[eax	[eax+10] (+3C]	[eax+3C]:1	x875W_B 0 x8 x875W_C1 0 x8 x875W_SF 0 x8	375W_C3 0 x875W 375W_C0 0 x875W 375W_P 0 x875W	C2 0 ES 0 U 0	
ecx= <kerne< th=""><th>132.MoveFile</th><th>EXW> (76A74</th><th>270) #9006C</th><th></th><th>callecx</th><th></th><th></th><th>ecx:MoveP ></th><th>Default (stdcall) 1: [esp] 02532 2: [esp+4] 000 3: [esp+8] 000 4: [esp+C] 043</th><th>1E4 L"C:\\Users\\ 00000 00004 F0000</th><th>√\D</th><th>5 0 Unlock</th></kerne<>	132.MoveFile	EXW> (76A74	270) #9006C		callecx			ecx:MoveP >	Default (stdcall) 1: [esp] 02532 2: [esp+4] 000 3: [esp+8] 000 4: [esp+C] 043	1E4 L"C:\\Users\\ 00000 00004 F0000	√\D	5 0 Unlock
Dump 1	Dump 2	Ump 3	Sii Dump 4	Sta Dump 5	Watch 1	[x=] Locals	Struct	0019F8C4 025 0019F8C8 000 0019F8CC 000	321E4 L"C:\\Use 00000 00004	rs\\ \\Desktop\	\lockb	oit.exe"

There is also a second process that will handle the executable deletion:

"cmd.exe /C ping 127.0.0.7 -n 3 > Nul & fsutil file setZeroData offset=0 length=524288 \"C:\\Users\\<User>\\Desktop\\lockbit.exe\" & Del /f /q \"C:\\Users\\<User>\\Desktop\\lockbit.exe\""

By pressing Shift+F1, we can access the hidden window:

LockBit 2.0 Ransom						-	×
3	LOCK			Active: 0	Completed: 5657		
			Statistics	Log			
Location	Used	Total	Complete	Disk u	50		
C:\	79.0 GB	79.5 GB					
Z:\	384 MB	166 MB	0				
			Figure	e 223			
			Ŭ				
LockBit 2.0 Ransom							×
10							
1							
				Active: 0	Completed: 5657		
			Statistics	Log			
[14:03:01] -	LockBit Ransomware 2.0						
[14:03:01] -	[-] Process created with limited rights						
[14:03:01] -	[+] AES-NI enabled						
[14:03:01] -	[+] Getting session keys from registry						
[14:03:01] -	Found volume \\?\Volume(d7e47829-0000	-0000-0000-	10000000000]\ on \Device\l	larddiskVolume1			
[14:03:01] -	Found volume \\?\Volume(d7e47829-0000	-0000-0000-	b0e213000000]\ on \Device\H	larddiskVolume2			
[14:03:01] -	Found volume \\?\Volume(fce79ce0-b01f-	11e6-b968-80	6e6f6e6963)\ on \Device\Cd	Rom0			
[14:03:16] -	Scan done, waiting handles						



Indicators of Compromise

Registry Keys

Key: HKEY_CLASSES_ROOT\Lockbit\shell\Open\Command

Data: "C:\Windows\system32\mshta.exe" "C:\Users\<User>\Desktop\LockBit_Ransomware.hta"

Key: HKEY_CLASSES_ROOT\Lockbit\DefaultIcon

Key: HKEY_CLASSES_ROOT\.lockbit\DefaultIcon

Key: HKEY_CLASSES_ROOT\htafile\DefaultIcon

Data: C:\windows\SysWow64\2ED873.ico

Key: SOFTWARE\Microsoft\Windows\CurrentVersion\Run\{2C5F9FCC-F266-43F6-BFD7-838DAE269E11}

Data: C:\Users\<User>\Desktop\LockBit_Ransomware.hta

Key: SOFTWARE\Microsoft\Windows\CurrentVersion\Run\{9FD872D4-E5E5-DDC5-399C-396785BDC975}

Data: <LockBit 2.0 file path>

Key: HKCU\Software\2ED873D4E5389C\Private

Key: HKCU\Software\2ED873D4E5389C\Public

Key: HKCU\Control Panel\Desktop

Data: Wallpaper = %AppData%\Local\Temp\<wallpaper>.tmp.bmp

Data: TileWallpaper = 0

Data: WallpaperStyle = 2

Files Created

C:\Users\<User>\Desktop\LockBit_Ransomware.hta

C:\windows\SysWow64\2ED873.ico

C:\Users\<User>\AppData\Local\Temp\<wallpaper>.tmp.bmp

C:\2ED873D4.lock (or any drive)

Processes spawned

cmd.exe /c vssadmin Delete Shadows /All /Quiet cmd.exe /c bcdedit /set {default} recoveryenabled No



cmd.exe /c bcdedit /set {default} bootstatuspolicy ignoreallfailures

cmd.exe /c wmic SHADOWCOPY /nointeractive

cmd.exe /c wevtutil cl security

cmd.exe /c wevtutil cl system

cmd.exe /c wevtutil cl application

cmd.exe /c vssadmin delete shadows /all /quiet & wmic shadowcopy delete & bcdedit /set {default} bootstatuspolicy ignoreallfailures & bcdedit /set {default} recoveryenabled no

cmd.exe /C ping 127.0.0.7 -n 3 > Nul & fsutil file setZeroData offset=0 length=524288 \"C:\Users\<User>\Desktop\lockbit.exe\" & Del /f /q \"C:\Users\<User>\Desktop\lockbit.exe\"

Mutex

\BaseNamedObjects\{3FE573D4-3FE5-DD38-399C-886767BD8875}

LockBit 2.0 Extension

.lockbit

LockBit 2.0 Ransom Note

Restore-My-Files.txt

LockBit_Ransomware.hta



Appendix

List of processes to be killed

wxServer wxServerView sqlmangr RAqui supervise Culture Defwatch winword QBW32 QBDBMgr gbupdate axlbridge httpd fdlauncher MsDtSrvr java 360se 360doctor wdswfsafe fdhost GDscan ZhuDongFangYu QBDBMgrN mysqld AutodeskDesktopApp acwebbrowser Creative Cloud Adobe Desktop Service CoreSync Adobe CEF Helper node AdobeIPCBroker sync-taskbar sync-worker InputPersonalization AdobeCollabSync BrCtrlCntr BrCcUxSys SimplyConnectionManager Simply.SystemTraylcon fbguard fbserver ONENOTEM wsa_service koaly-exp-engine-service TeamViewer_Service TeamViewer tv_w32 tv_x64 TitanV Ssms notepad RdrCEF sam oracle ocssd dbsnmp synctime agntsvc isglplussvc xfssvccon mydesktopservice ocautoupds encsyc tbirdconfig mydesktopgos ocomm dbeng50 sqbcoreservice excel infopath msaccess mspub onenote outlook powerpnt steam thebat thunderbird visio wordpad bedbh vxmon benetns bengien pvlsvr beserver raw_agent_svc vsnapvss CagService DellSystemDetect EnterpriseClient ProcessHacker Procexp64 Procexp GlassWire GWCtlSrv WireShark dumpcap j0gnjkol Autoruns Autoruns64 Autoruns64a Autorunsc Autorunsc64 Autorunsc64a Sysmon Sysmon64 procexp64a procmon procmon64 procmon64a ADExplorer ADExplorer64 ADExplorer64a tcpview tcpview64 tcpview64a avz tdsskiller RaccineElevatedCfg RaccineSettings Raccine_x86 Raccine Sqlservr RTVscan sqlbrowser tomcat6 QBIDPService notepad++ SystemExplorer SystemExplorerService SystemExplorerService64 Totalcmd Totalcmd64 VeeamDeploymentSvc

List of services to be stopped

wrapper DefWatch ccEvtMgr ccSetMgr SavRoam Sqlservr sqlagent sqladhlp Culserver RTVscan sqlbrowser SQLADHLP QBIDPService Intuit.QuickBooks.FCS QBCFMonitorService msmdsrv tomcat6 zhudongfangyu vmware-usbarbitator64 vmware-converter dbsrv12 dbeng8 MSSQL\$MICROSOFT##WID MSSQL\$VEEAMSQL2012 SQLAgent\$VEEAMSQL2012 FishbowlMySQL MSSQL\$MICROSOFT##WID SQLBrowser SQLWriter MySQL57 MSSQL\$KAV_CS_ADMIN_KIT MSSOLServerADHelper100 SOLAgent\$KAV CS ADMIN KIT msftesgl-Exchange MSSQL\$MICROSOFT##SSEE MSSQL\$SBSMONITORING MSSQL\$SHAREPOINT MSSQLFDLauncher\$SBSMONITORING MSSQLFDLauncher\$SHAREPOINT SQLAgent\$SBSMONITORING SQLAgent\$SHAREPOINT QBFCService QBVSS YooBackup YooIT vss sql svc\$ MSSQL MSSQL\$ memtas mepocs sophos veeam backup bedbg PDVFSService BackupExecVSSProvider BackupExecAgentAccelerator BackupExecAgentBrowser BackupExecDiveciMediaService BackupExecJobEngine BackupExecManagementService BackupExecRPCService MVArmor MVarmor64 stc_raw_agent VSNAPVSS VeeamTransportSvc VeeamDeploymentService VeeamNFSSvc AcronisAgent ARSM AcrSch2Svc CASAD2DWebSvc CAARCUpdateSvc WSBExchange MSExchange MSExchange\$

