# A detailed analysis of the Money Message Ransomware

**Prepared by:** Vlad Pasca, Senior Malware & Threat Analyst

SecurityScorecard

SecurityScorecard.com info@securityscorecard.com

Tower 49 12 E 49<sup>th</sup> Street Suite 15-001 New York, NY 10017 1.800.682.1707

# **Table of contents**

Executive summary	2
Analysis and findings	2
Thread activity – sub_ED2770 function	9
Thread activity – sub_F179D0 function	10
Thread activity - sub_F1D1C0 function	12
Running with thecrypt parameter	17
Running with the -d parameter	17
Running with the -I parameter	18
Running with the -v parameter	18
Indicators of Compromise	19

### **Executive summary**

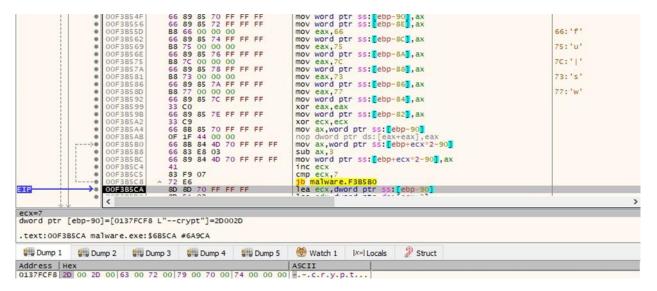
The threat actor group, Money Message ransomware, first appeared in <u>March 2023</u>, demanding million-dollar ransoms from its targets. Its configuration, which contains the services and processes to stop a ransomware attack, can be found at the end of the executable. The ransomware creates a mutex and deletes the Volume Shadow Copies using vssadmin.exe.

The files are encrypted using the ChaCha20 algorithm, with the key being encrypted using ECDH (Elliptic-curve Diffie-Hellman). The extension of the encrypted files isn't changed, however the structure of the files indicates they were encrypted.

# **Analysis and findings**

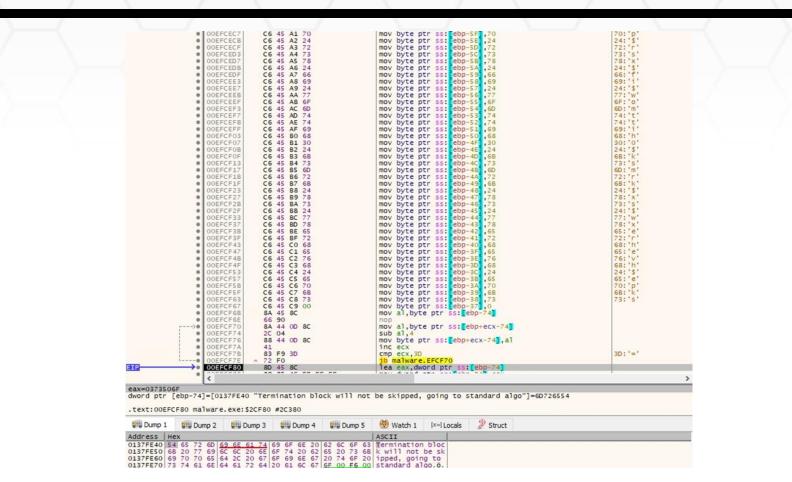
SHA256: 8be41efd6e6ace53b8c59344be2ba91fe41003987a8e38484b20760d7c400a42

The malware decrypts a list of arguments that it can run with: "--crypt", "-d", "-l", and "-v". We'll explain the purpose of each argument in our analysis.



#### Figure 1

The messages that would be displayed in the console are decrypted using the same SUB instruction (see Figure 2).



#### Figure 2

The ransomware retrieves the current system date and time via a function call to GetSystemTimePreciseAsFileTime:

Ump 1	Dump 2 Dump 3 Dump 4 Dump 5	Se motoria (* recebs 2/ sect		0137FD4C 0137FD60
		👹 Watch 1 🔤 Locals 🍃 Struct		0137FD48 000003D
	OOF4171C	mov ecx,es1 Cell decd ptr ds:[F51295] Cell es1 Cell es1	ecx:GetSystemTimeP esilGetSystemTimeP >	x875%_SF 0 x875%_P 0 x875%_U 0 Default (stdcal) 12 [esp] 0137FD58 13 [esp] 0137FD58 14 [esp] 0137FD69 14 [esp+C]_00F3CE9A ma1ware.00F3CE9A
	OOF41715     OOF41718     SF 75 08     OOF41718     S5 F6     OOF4171A     Y74 0C	push dword ptr ss:[ebp+6] test es1,es1 je malware.F41728	esi:GetSystemTimeP	x875xatusword 0000 x875w_B 0 x875w_C3 0 x875w_C2 0 x875w_C1 0 x875w_C0 0 x875w_ES 0

Figure 3

The GetCurrentThreadId API is utilized to obtain the ID of the calling thread:

EIP		00EE6F66			0 F8 00	call dword ptr ds: [<&GetCurrentThreadId>]	>
dword pt	tr [00F810	070 <malware.< td=""><td>&amp;GetCur</td><td>rentTh</td><td>nreadId&gt;]=&lt;</td><td><pre>ckernel32.GetCurrentThreadId&gt;</pre></td><td></td></malware.<>	&GetCur	rentTh	nreadId>]=<	<pre>ckernel32.GetCurrentThreadId&gt;</pre>	

Figure 4

The process writes relevant strings in the console by calling the WriteConsoleA method:

Dump 1 Dump 2 Dump 2	ump 3 👹 Dump 4 👹 Dump	5 👹 Watch 1 Ix-I Locals 🤌 Struct		0137F880 00000098 0137F884 0137F888 "[2023-08-17 07:28:23.996] [info] T 0137F886 00000018 0137F886 00000000
dword ptr [OOF51050 <malware .text:00EF605C malware.exe:5</malware 		<b>Call dword producted and teconsoleA</b> >]	*	Default (stdcal)         *         S         Interpretation           12         [esp]         0030098         [s]         [s] </th
OOEF604E     OOEF6050     OOEF6052     OOEF6053     OOEF6053     OOEF6059	6A 00 6A 00 50 FF 85 D0 FE FF FF FF 77 08	push 0 push 0 push eax push dword ptr ss:[ebp-130] push dword ptr ds:[edi+8]	[ebp-130]:"[2023-0	x875%L2U3W0rd 0000 x875%L20 0 x875%LC3 0 x875%LC2 0 x875%LC1 0 x875%LC0 0 x875%LES 0 x875%L50 x875%L20 0 x875%L2 0

Figure 5

It extracts information about the console screen buffer using GetConsoleScreenBufferInfo:

OOEF6073 50     ODEF6074 FF 77 08     Dush dword ptr ds:[ed1+8]		x875W_SF 0 x875W_P 0 x87	SW_U 0
313     →     001550022     FF 15 84 10 FB 00     Call dword pir dsi[ <sdetconsolescreen8ufferinfo>]       36word pir [00F81084 -malware.dgetConsoleScreen8ufferInfo&gt;]=<kernel32.getconsolescreen8ufferinfo>       .text:00EF6077 malware.exe:\$26077 #25477</kernel32.getconsolescreen8ufferinfo></sdetconsolescreen8ufferinfo>	>	Default (stdcal) 1: [esp] 00000098 2: [esp+4] 0137FCB4 3: [esp+4] 5C631921 4: [esp+C] 03737E0	▼ S C Unlock
🗱 Dump 1 🗱 Dump 2 🗱 Dump 3 🗱 Dump 4 🗱 Dump 5 👹 Watch 1 💷 Locals 🦻 Struct		0137F88C 00000098 0137F890 0137FC84	

Figure 6

The malware changes the text color for output using the SetConsoleTextAttribute function (0x2 = **FOREGROUND\_GREEN**):

OOEF6098 50     ODEF609C FF 77 08     push dword ptr ds:[edi+8]			875W_U 0
CONSEGUE FF 15 85 10 F8 00     Constant for dir dir dir dir dir dir dir dir dir di	,	Default (stdcall) 1: [esp] 00000098 2: [esp+4] 0000002 3: [esp+8] 5C631921 4: [esp+C] 037376E0	▼ S 🗘 🗆 Unloci
🗱 Dump 1 📲 Dump 2 🚝 Dump 3 🚝 Dump 4 💭 Dump 5 🥮 Watch 1 💷 Locals 🎾 Struct		0137F88C 00000098 0137F890 00000002	

Figure 7

The executable file is opened by calling the wfsopen method (0x40 = **\_SH\_DENYNO**):

	<ul> <li>00F3E78A</li> <li>00F3E78D</li> <li>00F3E794</li> </ul>	FF 75 10 FF 34 85 FO 62 FF 75 08	push dword ptr ss:	ax 4+F862F0]	[eax*4+F862F0]:L"r [ebp+8]:L"C:\\User	x875w_C1 0 x875w_C0 x875w_SF 0 x875w_P	
malware.00F	58ABF	E8 23 A3 01 00	call maiware, FS8ABF		wfsopen v	Default (stdcal) 1: [esp] 03737138 L" 2: [esp+4] 00F86334 3: [esp+8] 0000040 4: [esp+C] 0137EE84	▼ 5 ♀ □ Unlock C:\\Users\\ □\\Desktop\\malware.e L"rb"
UIL Dump 1		ump 3 11 Dump 4	Ump 5 👹 Watch 1 🔤 Locals	2 Struct		0137EE5C 03737138 L" 0137EE60 00F86334 L"	C://Users// Desktop//malware.e
Addrare 10			ASCTT I			0137EE64 00000040	



The binary is looking for its configuration by reading 4096 bytes at a time:

ODFF04D01 6A 00     ODFF04D02 80 45 EC     ODFF04D5 50     ODFF04D5 50     ODFF04D6 53     ODFF04D7 88 5D F4     ODFF04D7 55 E8     ODFF04D7 F7 5E 8     ODFF04D1 FF 15 24 11 F6 10        S15      ODFF04D1 FF 15 24 11 F8 10        Odf04D1 FF 15 24 11 F8 10        Odf04D1 FF 15 24 11 F8 10	<pre>push 0 lea eax_dword ptr ss:[ebp-14] push eax mov ebx,dword ptr ss:[ebp-C] push ebx push dword ptr ss:[ebp-18] call dword ptr ds:[caReadfiles] call dword ptr ds:[caReadfiles]</pre>	Adv: Im_G 3         CEMPLY)         Adv: Im_G 4           x8757tatusword 0000         x875W_C3         0         x875           x875W_C1         0         x875W_C3         0         x875           x875W_55         0         x875W_F9         x875         x875W_F9         x875           Default (stdcal)         1:         [esp] 0000020C         1:         [esp] 000020C         1:         [esp] 000020C         1:         [esp] 0000020C         1:         [esp] 000020C         1:         1:         [esp] 000020C         1:         1:         [esp] 000020C         1:         1:         [esp] 000020C         1:	5W_C2 0 5W_ES 0 5W_U 0
.text:00F604DE malware.exe:\$904DE #8F8DE		2: [esp+4] 03747DD0 3: [esp+8] 00001000 4: [esp+C] 0137ED30	
Ump 1 Ump 2 Ump 3 Ump 4 Ump Dump 4	6 👹 Watch 1 🛛 🕸 I Locals 🎾 Struct	0137E000 0000020C 0137E004 03747DD0	
Address Hex 03747DD0 00 00 00 00 00 00 00 00 00 00 00 00	ASCII	0137ED08 00001000 0137ED0C 0137ED30 0137ED10 00000000	

### Figure 9

The configuration contains the ransom note content, the mutex name, a list of directories that will be skipped, the public key, a list of processes and services to stop, a list of corporate credentials previously extracted from the victim, and the temporary extension:

Address	He	×															ASCII
05342840	7B	0D	0A	20	20	20	20	22	69	6E	66	6F	5F	74	65	78	<pre>[ "info_tex</pre>
5342850	74	5F	6D	65	73	73	61	67	65	22	3A	20	22	57	57	39	t_message": "WWS
5342860	31	63	69	42	GD	61	57	78	6C	63	79	42	33	59	58	4D	1ciBmaWx1cyB3YXM
5342870	67	5A	57	35	6A	63	6E	6C	77	64	47	56	6B	49	47	4A	gZW5jcn1wdGVkIG:
5342880	35	49	43	4A	4E	62	32	35	6C	65	53	42	74	5A	58	4E	5ICJNb25leSBtZXM
5342890	7A	59	57	64	6C	49	69	42	77	63	GD	39	6D	61	58	52	zYWdlIiBwcm9maXF
53428A0	68	59	6D	78	6C	49	47	39	79	5A	32	46	75	61	58	70	hYmx1IG9yZ2FuaXp
53428B0	68	64	47	6C	76	62	69	41	67	59	57	35	6B	49	47	4E	hdGlvbiAgYW5kIGM
053428C0	68	62	69	64	30	49	47	4A	6C	49	47	46	6A	59	32	56	hbid0IGJ1IGFjY2\
053428D0	7A	63	32	56	6B	49	47	46	75	65	57	31	76	63	GD	55	zc2VkIGFueW1vcml
053428E0	75	44	51	6F	4E	43	6B	6C	6D	49	48	6C	76	64	53	42	uDQoNCklmIHlvdSE
053428F0	77	59	58	68	67	63	GD	46	75	63	32	39	74	4C	43	42	wYXkgcmFuc29tLCE
5342900	35	62	33	55	67	64	32	6C	73	62	43	42	6E	5A	58	51	5b3Ugd21sbCBnZX0
5342910	67	59	53	42	6B	5A	57	4E	79	65	58	42	30	62	33	49	gYSBkZWNyeXB0b31
5342920	67	64	47	38	67	5A	47	56	6A	63	6E	6C	77	64	43	42	gdG8gZGVjcn1wdCE
5342930	30	61	47	56	74	4C	69	42	45	62	32	34	6E	64	43	42	
5342940			6E	6B	67	64	47	38	67	5A	47	56	6A	63	6E	6C	
5342950	77	64	43	42	GD	61	57	78	6C	63	79	42		62	33	56	
05342960	79	63	32	56	73	5A	69	41	74	49	47	6C	75	49	48	52	yc2VsZiAtIGluIHF
									Fig	ure	10						
																	Jgd2lsbCBnZXQgY3BkZWByeXB0b3IgdG8gZGVjcnlwd onRid2U2emFwcHEyMnZnbJNudT24b2IrDNR0MmVzZW
BocD93aGF05W09YmFnN5	NANTQxOD LCBpb1Bv	AzHTF123	ZzoNOg08	CH2OWZ12	NTINDQNC N3RgeWp	NVzaWSnI 2c220a09	HRVC1810 1dHR62TU	mp3c2Vy2 lydzH2d3e	tGhOdHBz tpejHOaH	Oldvd3d3 JmY21vcm	LnRvonBy d2ZSpiNt	b2p1Y3Q	ab3JnL2R	vd25sb2F 25pb24NC	kLwOKDQs gOKRW530	JbiBjTX	N1IH1vd5ByIN21c2OgdG4gcGF5LCB3I5B3aWxaIHBvc IGZpbGVzIGNbb1d0IGJ1IGR1Y3J5cHR12CB3aXRob3V
345-12235-12354",																	
<pre>spokXNvY2FjaG0=","Og "Qzpod21u2093cw==","</pre>						VtISBpbm	ZvomlhdG	ilvbg=="	*Ospecci	WyZmawZ3	N=", "Ozj	oceBJv23	ThewRhos	E-*,*Qap	ocHJvZ3	Thb58na¥a	<pre>klcyAceDg2RQ****,*QzpocHJv23JhbSBmaHklow****</pre>
an a	85 (R. 41	11-3-5-5		1000		39305262	570bcbf4	7aa9055:	(b)eccic	82be053c	:8d952974	alf2e32	18005e70	59461ble	558feod	dbc1fa34	6bf9f9bc5746d431902fe990772d16bable779dbe63
\$1a74670dd05eb719678	d62d150e	dee22704	t",														
:3FsLmV4ZQ==*, "b3JhY	2x11mV4Z	Q	Nzc2Qu23	ch1", "ZG	Izbalwim	V4ZQmm",	"c31uY39	pbWUuZX3	11" TWO	lodH3R2Yy5	1eG0=","	aXNxbHB:	theb z RXD e	uZXhl","	eGZzc3Z;	Y29uLmV4	<pre>#20==","bX1k2XNzdG9wc2Vydmlj2S51eGD=","b2Nk BhdGgu2Xh1","bXNkY2N1c3Hu2Xh1","bXNwdWIu2Xh</pre>

Figure 11

The directories extracted from the "skip\_directories" field are Base64-decoded (Figure 12).

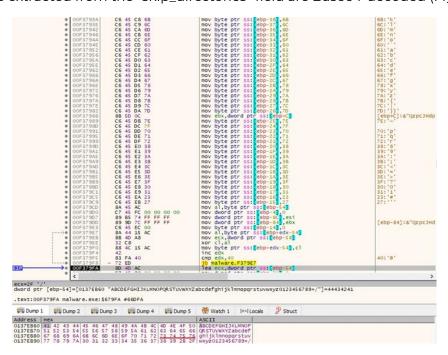


Figure 12

("info "WW91c hkgd00 kgd00 HN0b1a uIH0vv "mutea "sutea "skip "Qipos "netwo "fl820

/ these directories": {
ize\_for\_partial" : 100,
extension": "cbgnfvn")

The ransomware creates a mutex called "12345-12345-12235-12354", which ensures that only one copy is running at a single time:

OOEF9AEA 50     OOEF9AEB 6A 00     OGEF9AED 6A 00	push eax push o push o	eax: "12345-12345-1	x875W_C1 0 x875W_C0 0 x875W_ES 0 x875W_SF 0 x875W_P 0 x875W_U 0
dword ptr [ODF81064 «malware.4Create4utexAs]»	1	······································	Default (stdcal) ▼ 5 C □ Unlock 11 [esp] 00000000 22 [esp+4] 00000000 33 [esp+8] 05163188 "12345-12345-12235-12354" 44 [esp+4] 05163188 "12345-12345-12235-12354"
Ump 1 Ump 2 Ump 3 Ump 4	Ump 5 👹 Watch 1 Ix=ILocals 🎾 Struct	500	01372574 013725758 00000000 01372575 05163388 "12345-12345-12235-12354"

#### Figure 13

It opens the "ServicesActive" database using the OpenSCManagerW API (0x80000000 = **GENERIC\_READ**):

	<ul> <li>00F00E4E</li> <li>00F00E53</li> <li>00F00E58</li> <li>00F00E5A</li> <li>00F00E61</li> </ul>	68 00 00 00 80 68 <u>14 32 F8 00</u> 6A 00 C7 45 FC 00 00 00 00 C7 45 BC 01 00 00 00	push 8000000 push malware.F83214 push of ptr ss:[ebp-4],0 mov dword ptr ss:[ebp-44],1	F83214:L"ServicesA		
		<u>FF 15 40 10 F8 00</u> &OpenSCManagerW>]= <advapi< th=""><th>call dword ptr ds:[c&amp;OpenSCManagerw&gt;] 32.0penSCManagerw&gt;</th><th>,</th><th>Default (stdcall) 1: [esp] 00000000 2: [esp+4] 00F83214 L"Serv 3: [esp+5] 8000000 4: [esp+6] 5C6308A9</th><th>▼ 5 € Unlod</th></advapi<>	call dword ptr ds:[c&OpenSCManagerw>] 32.0penSCManagerw>	,	Default (stdcall) 1: [esp] 00000000 2: [esp+4] 00F83214 L"Serv 3: [esp+5] 8000000 4: [esp+6] 5C6308A9	▼ 5 € Unlod
.text:00F00	E68 malware.exe:\$3		5 👼 Watch 1 🔤 Locals 🎾 Struct		0137ECE0 00000000	
www.bumpii	Ump 2 Ump 2	mp 3 📲 Dump 4 📲 Dump !	watch 1  X=1Locals 2 Struct		0137ECE4 00F83214 L"Servic 0137ECE8 80000000	esActive"

#### Figure 14

The malicious binary obtains a list of all services that run in their own processes using EnumServicesStatusExW (0x10 = SERVICE\_WIN32\_OWN\_PROCESS, 0x3 = SERVICE\_STATE\_ALL):

OOFOREA2     SD 45 C4     OOFOREA2     SD 45 C4     OOFOREA2     SO 45 D4     OOFOREA     SO 45 D4     OOFOREA     SO 45 D4     OOFOREA     SO 40     OOFOREA     SO 40	push 0 lea eavid of pr ss:[ebp-4],2 push eax push eax push eax push eavid of pr ss:[ebp-40] push eavid pr ss:[ebp-40] push eavid pr ss:[ebp-30] push 0 push 0 push 0 push est push eavid pr ss:[ebp-30] call dewng ptr ss:[ebp-30] call dewng pt	x8777 0000000000000000000000000000000000
.text:00F00EC4 malware.exe:\$30EC4 #302C4		3: [esp+8] 00000010 4: [esp+C] 00000003
	9 5 🥮 Watch 1 🕬 Locals 🤌 Struct	3: [esp+8] 00000010

Figure 15

The following services will be stopped:

• "vss" "sql" "svc\$" "memtas" "mepocs" "sophos" "veeam" "backup" "vmms"

🗾 🚄 🖼		
.text:00F015DB		
.text:00F015DB	loc_F01	5DB:
.text:00F015DB	lea	eax, [ebp+ServiceStatus]
.text:00F015DE	mov	[ebp+var_2C], 0
.text:00F015E5	push	eax ; lpServiceStatus
.text:00F015E6	mov	eax, [ebp+var_50]
.text:00F015E9	xorps	xmm0, xmm0
.text:00F015EC	push	SERVICE_CONTROL_STOP ; dwControl
.text:00F015EE	movups	xmmword ptr [ebp+ServiceStatus.dwServiceType], xmm0
.text:00F015F2	movups	xmmword ptr [ebp+ServiceStatus.dwServiceSpecificExitCode], xmm0
.text:00F015F6	push	dword ptr [eax] ; hService
.text:00F015F8	call	ds:ControlService
.text:00F015FE	test	eax, eax
.text:00F01600	jnz	short loc_F01617



The malware takes a snapshot of all processes via a call to CreateToolhelp32Snapshot (0x2 = **TH32CS\_SNAPPROCESS**):

	<pre>00F0110E 00F01110</pre>	6A 00 6A 02	push 0 push 2			875W_U 0
EIP	→• 00±01112 <	E8 93 88 03 00	call smalware.CreateToolhelp32Snapshot>	>	Default (stdcall) 1: [esp] 00000002	👻 💈 💭 Unloci
	teToolhelp32Snap 2 malware.exe:\$3				2: [esp+4] 00000000 3: [esp+8] 5C630AED 4: [esp+C] 00007A68	
Ump 1	Dump 2 🔛 Du	np 3 🗱 Dump 4 👯 Du	mp 5 👹 Watch 1 🛛 Ix=I Locals 🦻 Struct		0137EC74 00000002 0137EC78 00000000	

Figure 17

The processes are enumerated using the Process32FirstW and Process32NextW APIs, as shown below:

COFOLIAE     SO     OUTINE     OUTINE     OUTINE     SO     OUTINE     OUTIN	
👜 Dump 1 💭 Dump 2 🕮 Dump 3 🕮 Dump 4 🕮 Dump 5 👹 Watch 1 IrelLocals 🤌 Struct Figure 18	035565641 00000228 01376678 03378686
COFFOLISE 50     COFFOLISE 57     COFFOLISE 57     COFFOLISE 57     COFFOLISE 57     COFFOLISE 54 88 03 00     Call enalware, Process32NextWr.	v 27.5%_5F 0 x87.5%_P 0 x87.5%_U 0 Default (stdcall) ▼ 5 € Utrice
cmalware.Process32Nextm> .text:00F0115D malware.exe:\$3115D #3055D	1: [esp] 00000228 2: [esp+4] 0.37€CBC 3: [esp+8] 5C630AED 4: [esp+C] 00007A6B

Figure 19

The ransomware opens the target processes using the OpenProcess method (0x1 = **PROCESS\_TERMINATE**):

	<ul> <li>00F01287</li> <li>00F01280</li> <li>00F0128F</li> </ul>	FF BS B4 FD FF 6A 00 6A 01	push 0 push 1	d ptr ss:[eb			x875W_C1 0 x875W_C0 x875W_SF 0 x875W_P	0 x875W_ES 0 0 x875W_U 0
	ODF012C1		oo call dwo	d ptr ds:[<&	OpenProcess>]	,	Default (stdcal) 1: [esp] 00000001 2: [esp+4] 00000000 3: [esp+8] 00000558 4: [esp+c] 5C630AED	▼ S C Unlock
-					di		01375670 00000001	
Address Hey	Dump 2	Cump 3 📲 Dump 4	Ump 5 💮 Watch :	x=  Locals	2 Struct		0137EC74 00000000 0137EC78 00000558	

Figure 20

The following processes will be killed:

 "sql.exe" "oracle.exe" "ocssd.exe" "dbsnmp.exe" "synctime.exe" "agntsvc.exe" "isqlplussvc.exe" "xfssvccon.exe" "mydesktopservice.exe" "ocautoupds.exe" "encsvc.exe" "firefox.exe" "tbirdconfig.exe" "mdesktopqos.exe" "ocomm.exe" "dbeng50.exe" "sqbcoreservice.exe" "excel.exe" "infopath.exe" "msaccess.exe" "mspub.exe" "onenote.exe" "outlook.exe" "powerpnt.exe" "steam.exe" "thebat.exe" "thunderbird.exe" "visio.exe" "winword.exe" "wordpad.exe" "vmms.exe" "vmwp.exe"

OOF01201 6A 00 push 0     OOF01203 57 push ed1		x875w_SF 0 x875w_P 0	x875W_U 0
312 → 00501203 FF 15 90 10 F8 00 call dword ptr ds:[<&TerminateProcess>]	>	Default (stdcall) 1: [esp] 0000024C	▼ S C Unlock
dword ptr [00F81090 «malware.&TerminateProcess>]= <kernel32.terminateprocess> .text:00F012D4 malware.exe:\$312D4 #306D4</kernel32.terminateprocess>		2: [esp+4] 00000000 3: [esp+8] 5C630AED 4: [esp+C] 00007A6B	
🗊 Dump 1 💷 Dump 2 🐖 Dump 3 🕮 Dump 4 🐖 Dump 5 👹 Watch 1 🕬 Locals 🎾 Struct		0137EC74 0000024C 0137EC78 00000000	

Figure 21



The following directories will not be encrypted:

 "C:\msocache" "C:\\$windows.~ws" "C:\system volume information" "C:\perflogs" "C:\programdata" "C:\program files (x86)" "C:\program files" "C:\\$windows.~bt" "C:\windows" "C:\windows.old" "C:\boot"

The executable retrieves a pseudo handle for the process, as highlighted in Figure 22.

EIP	00F376BF <	FF 15 AC 10 F8 00	call dword ptr ds:[<&GetCurrentProcess>]	>
dword ptr [00F8	10AC <malware.< td=""><td>GetCurrentProcess&gt;]=<ke< td=""><td>ernel32.GetCurrentProcess&gt;</td><td></td></ke<></td></malware.<>	GetCurrentProcess>]= <ke< td=""><td>ernel32.GetCurrentProcess&gt;</td><td></td></ke<>	ernel32.GetCurrentProcess>	

#### Figure 22

IsWow64Process is used to determine if the process is running on a 64-bit architecture:

• 00F376C5	50	push eax call esi		-		
	FF D6	Call esi	es1:IsWow64Process	Defa	ult (stdcall)	▼ 5 🗢 🗌 Unlocke
esi= <kernel32.iswow64process></kernel32.iswow64process>	(76A75920)			1: 2:	[esp] FFFFFFF [esp+4] 0089E760	

#### Figure 23

The ransomware disables file system redirection for the current thread (see Figure 24).

OF382FD 8D 85 98 FE FF FF lea eax,dword ptr ss: ebp-168     OF38303 50 push eax		x875W_SF 0 x875W_P 0 x	875W_U 0
Call est	es1:wow64D1sabTewow64	Default (stdcall)	▼ 5 C Unlocke
lesiw-ckernel32.wow5401sablewow64FSRedirection> (76A76A30) .text:00F38304 malware.exe:\$68304 #67704		1: [esp] 00895810 2: [esp+4] D57088F8 3: [esp+8] 048A5918 4: [esp+C] 048A5918 5: [esp+10] 0089F714	

Figure 24

It deletes all Volume Shadow Copies using the vssadmin.exe tool:

dword ptr [OOF8126C «malware .text:00F38785 malware.exe:5		:.ShellExecutew>	,	1: [esp] 00000000 2: [esp+4] 0089E8A8 L"open" 3: [esp+6] 0089E8A8 L"cmd.exe" 4: [esp+C] 0089E8AC L"/c vssadmin.exe delete shadows /all /qu
OOF38735     OOF38736     OOF38736     OOF38737     OOF3874     OOF3874     OOF38748     OO	50 50 50 50 85 85 85 78 FF FF FF 50 85 D4 FE FF FF 50 85 C0 FE FF FF 50 85 30 FF FF FF 50 85 30 FF FF FF 50 65 00 FF F5 85 12 F8 00	Jush eax push eax mov word ptr ss:[ebp-120] push eax heax heax heax heak eax, dword ptr ss:[ebp-140] hush eax heak push eax push eax push eax push eax push eax heat	eax:L"open" eax:L"open" eax:L"open" eax:L"open" eax:L"open" eax:L"open"	x87TW_2 3 (Empty) x87TW_3 5 (Empty) x87TW_4 3 (Empty) x87TW_5 5 (Empty) x87TW_6 3 (Empty) x87TW_5 (Empty) x875W_6 0 x875W_C 0 0 x875W_C 2 0 x875W_6 0 x875W_C 0 0 x875W_C 2 0 x875W_5 0 x875W_P 0 x875W_E 0 x875W_5 0 x875W_P 0 x875W_U 0 v875W_5 0 x875W_P 0 x875W_U 0 x875W_5 0 x875W_P 0 x875W_U 0 x875W_5 0 x875W_P 0 x875W_U 0 x875W_5 0 x875W_F 0 x875W_U 0 x875W_5 0 x875W_F 0 x875W_U 0 x875W_5 0 x875W_5 0 x875W_F 0 x875W_U 0 x875W_5 0 x875W_5 0 x875W_F 0 x875W_U 0 x875W_5 0 x875W_F 0 x875W_F 0 x875W_U 0 x875W_5 0 x875W_F 0 x8



A new thread is created, which runs the sub\_ED2770 function even if the argument passed to CreateThread is the StartAddress function:

<u>E19</u>	ODEFA645     ODEFA645     ODEFA648     ODEFA648     ODEFA649     ODEFA650     ODEFA650     ODEFA652	51 6A 00 6A 00 6A 00 6A 00 6A 00 6B D1 29 06 00	push eck push eak push eak push malware.502770 push 0 push 0 call malware.50028	 x875tatusword 0020 x875m_6 0 x875m_C3 0 x875m_C2 0 x875m_C10 x875m_C0 0 x875m_E5 0 x875m_57 0 x875m_P 1 x875m_U 0 Default (stdcall) v [5 0]
malware.00	F5D028 A652 malware.exe:\$2/	A652 #29A52		1: [esp] 00000000 2: [esp+4] 00000000 3: [esp+5] 00ED2770 malware.00ED2770 4: [esp+C] 04BA3470
			Figure 26	
0.C	OOF5D061     OOF5D065     OOF5D065     OOF5D068     OOF5D068	51 FF 75 18 50 68 <u>CC CE F5 00</u> FF 75 08 FF 75 08 FF 15 <u>X 11 F8 00</u>	push ess push doord ptr ss:[ebp+18] push mainare.FSCECC push dword ptr ss:[ebp+2] call dword ptr ds:[esCreateThread>]	 x875tatusword         0020           x875tatusword         0x875kL2         0x875kL2           x875kL2         0x875kL2         0x875kL2           Default (stdcall)         v         5
	[OOF8119C «malware.a	LCreateThread>]≪kernel3 0071 #8C471	:CreateThread>	 1: [esp] 00000000 2: [esp+4] 00000000 3: [esp+8] 00F5CECC malware.00F5CECC 4: [esp+C]_04B87720



# Thread activity - sub\_ED2770 function

The LookupPrivilegeValueA API is utilized to obtain the LUID for the following privileges: "SeAssignPrimaryTokenPrivilege", "SeRestorePrivilege", and "SeTakeOwnershipPrivilege":



#### Figure 28

The executable opens the access token associated with the current process (0xF01FF = **TOKEN\_ALL\_ACCESS**):



#### Figure 29

The process enables all the privileges mentioned above by calling the AdjustTokenPrivileges method, as highlighted below:

OOF3ABD3 6A 00     OOF3ABD3 6A 00     OOF3ABD5 6A 00     OOF3ABD7 6A 10     OOF3ABD7 6A 55 EC     OOF3ABD7 FF 75 56     OOF3ABD7 FF 75 56     OOF3ABD7 FF 75 56     OOF3ABTF FF 75 50     OOF3ABTF FF 75 50     OOF3ABTF FF 75     OOF3ABTF	push 0 push 0 push 10 lea eax,dword ptr ss:[ebp-14] push dword ptr ss:[ebp-18] call dword ptr ss:[ebp-18] call dword ptr ss:[edp-18]	x8/1m_0 5 (cmu/y) x8/1m_0 5 (cmu/y) x875m_6 0 x875m_6 0 0 x875m_6 0 x875m_
dword ptr [00F81030 «malware.&AdjustTokenPrivileges>]= .text:00F3ABE2 malware.exe:\$6ABE2 #69FE2	advapi32.AdjustTokenPrivileges>	11 [ESD] 00000204 21 [ESD+4] 0000000 31 [ESD+4] 00895648 45 [ESD+6] 0000010
Dump 1 Dump 2 Dump 3 Dump 4 Dump 5	👹 Watch 1 🔤 Locals 🦻 Struct	00591618 00000204 00595616 00000000
Address Hex 0089E644 D4 02 00 00 01 00 00 00 03 00 00 00 00 00 00 00 0089E654 02 00 00 00 DF 84 70 D5 88 E7 89 00 AC C1 EE	ASCII	O0896620 00090648     O0896624 0000010     O0896624 0000000     O0896623 00000000     O0896623 00000000

Figure 30

The ransomware retrieves a list of sessions on the machine using WTSEnumerateSessionsW (Figure 31).

.text:00F1C0EE ma1ware.exe:54C0EE #484EE ## Dump 1 ## Dump 2 ## Dump 3 ## Dump 4 ## Dump 5 @ Watch 1 [r=]Locals 2 Struct Address   Hex   ASCII	42 (ssyc1) 058E7930 51 (ssyc1) 058E7940 058E7840 058E7885 (0000000 ▲ 058E7830 058E7930
• ODFICOE7     C6 85 17 FF FF FF 00     mov byte ptr ssi tebp-teg 0       • ODFICOEF     E8 A5 08 02 00     call call call call call call call call	
<pre>     OOFLOCF 50 00 FF FF FF push eax     OutlocFF 50 00 FF FF FF 00 00 00 00 00 00 00 00 00</pre>	∧0:1m_7         LCmpty)         ×0:1m_7         LCmpty)           ×87m_6         LCmpty)         ×87m_7         LCmpty)           ×87m_6         LCmpty)         ×87m_7         LCmpty)           ×875m_8         0         ×875m_6         0           ×875m_6         0         ×875m_6         0           ×875m_6         0         ×875m_6         0           ×875m_5         0         ×875m_6         0



For each of the identified sessions, the malware obtains the access token of the logged-on user:

OOF1C203 51     OOF1C204 50	push ecx push eax	x875W_SF 0	x875W_P 0 x875W_U 0
STE →• 00F1C2D5 E8 CA 06 02 00	call «malware.wTSQueryUserToken»	Default (stdcall)	▼ 5 C □ Unlocke
<malware.wtsqueryusertoken></malware.wtsqueryusertoken>		1: [esp] 000 2: [esp+4] 0	000001 )\$8EF920

Figure 32

The DuplicateTokenEx method is used to create an impersonation token that duplicates the above token (0x2 = **SecurityImpersonation**, 0x2 = **TokenImpersonation**):



#### Figure 33

The malicious binary creates a new thread that will identify the shared resources. The credentials extracted from the configuration will be used to access those shares.

00F1C43F 51     00F1C440 6A 00     00F1C442 50     00F1C443 6A 00     00F1C443 6A 00     00F1C443 6A 00     00F1C448 6A 00	Dush ecx Dush 0 Dush eax Dush matware.F179D0 Dush 0 Dush 0	x875tatusword 0000 x875w_B 0 x875w_C3 0 x875w_C2 0 x875w_C10 x875w_C2 0 x875w_E5 0 x875w_SF 0 x875w_P 0 x875w_U 0
EXE →• 00F1C44C E8 D7 08 04 00	call malware.FSD028	V Default (stdcall) V S C Unlod
malware.00F5D028 .text:00F1C44C malware.exe:\$4C44C #4884C		1: [csp] 0000000 2: [csp+0] 00000000 malware.00F17900 4: [csp+C] 032084A8 5: [csp+C] 032084A8
1 Dump 1 1 Dump 2 1 Dump 3 1 Dump 4 1 Dump 4	5 👹 Watch 1 🛛 🕸 Locals 🎾 Struct	058553880 0000000 05855884 00000000
Address   Hex	ASCII	OSBEF888 00F17900 malware.00F17900 OSBEF88C 032084A8
032084A8 CC 03 00 00 <u>68 FA BE 05</u> <u>47 F9 BE 05</u> AB AB A6 032084B8 AB AB AB AB AB EE FE EE FE 00 00 00 00 00 00 00	A8 1hús.Gus.eeee 00 eeetipip	058EF890 0000000 058EF890 058EF92C

#### Figure 34

# Thread activity – sub\_F179D0 function

The executable extracts a pseudo handle for the current thread:

EIP 00F19E36	FF 15 <u>CO 10 F8 00</u>	call dword ptr ds: [<&GetCurrentThread>]	>				
dword ptr f00F810C0 <malware.&getcurrentthread>l=<kernel32.getcurrentthread></kernel32.getcurrentthread></malware.&getcurrentthread>							

#### Figure 35

SetThreadToken is utilized to assign the impersonation token to the current thread, as displayed in Figure 36.

	<ul> <li>00F19E48</li> <li>00F19E49</li> </ul>	56	push est push eax		x875W_SF 0 x875W_P 0 x87	SWU 0
EIP	>= 00F19E4A	FF 15 18 10 F8 00	call dword ptr ds: [<&SetThreadToken>]	, v	Default (stdcall)	👻 5 🗘 🗌 Unlocke
dword ptr	dword ptr [00F81018 <maiware.&setthreadtoken>]=<advapi32.setthreadtoken></advapi32.setthreadtoken></maiware.&setthreadtoken>				1: [esp] 45D3F848 2: [esp+4] 000003CC	

#### Figure 36

The ransomware starts enumerating all currently connected resources via a function call to WNetOpenEnumW (0x1 = **RESOURCE\_CONNECTED**):



#### Figure 37

The enumeration continues using the WNetEnumResourceW API (see Figure 38). The malware is looking for files to encrypt in these network resources.



EIP	• 00F197A7     • 00F197A8     • 00F197A9     • 00F197A9     • 00F197AF     • 00F197B0     • 00F197B6     <	50 50 80 85 85 85 85 85 85 85 85 85 85	push eax push esi lea eax,dword ptr ss:[ebp-114] push eax push dword ptr ss:[ebp-130] call dwalware,mwetenumkesourcew	[ebp-130]:"conn"	xa/Statusmura 0000 xa/Statusmura 0000 xa/Stw_C1 0 xa/Stw_C3 0 xa/S xa/Stw_C1 0 xa/Stw_C 0 0 xa/S xa/Stw_SF 0 xa/Stw_P 0 xa/St Default (stdcal)	W_ES 0
<malware.w< td=""><td>NetEnunResourcew&gt;</td><td></td><td></td><td></td><td>1: [esp] 04889208 "c0nn" 2: [esp+4] 4503F6A8 3: [esp+6] 04800508 4: [esp+C] 4503F680</td><td></td></malware.w<>	NetEnunResourcew>				1: [esp] 04889208 "c0nn" 2: [esp+4] 4503F6A8 3: [esp+6] 04800508 4: [esp+C] 4503F680	
			Figure 3	38		

We continue to analyze the main thread.

The process iterates over drives in the range "Z:" to "A:" and calls the GetDriveTypeW method:

OOF188E0 50 push eax     OOF188E1 C7 85 10 FF FF FF 07 00 00 may dward ptr ss: ebo-F01.7	eax:L"\\\\?\\Z:\\"	x875W_SF 0 x875W_P 1 x87	75W_U 0
ODF18851 C7 85 10 FF FF FF 07 00 00 mov dword ptr ss:[cbp=F0],7     ODF18853 FF 15 8C 10 F8 00     call dword ptr ds:[c&GetDr1veTypew>]     call dword ptr ds:[c&GetDr1veTypew>]		Default (stdcall)	★ 5 \$ Unlocke
dword ptr [00F8108C <maiware.dgetdrivetypew>]=<kernel32.getdrivetypew></kernel32.getdrivetypew></maiware.dgetdrivetypew>		1: [esp] 0089E828 L"\\\\?\\Z	://"

### Figure 39

For each of the identified drives, the ransomware calls the CreateFileW function (0x80 = FILE\_READ\_ATTRIBUTES, 0x7 = FILE\_SHARE\_DELETE | FILE\_SHARE\_WRITE | FILE\_SHARE\_READ, 0x3 = OPEN\_EXISTING, 0x02000000 = FILE\_FLAG\_BACKUP\_SEMANTICS):

<ul> <li>OOF3ECS0</li> <li>OOF3ECS0</li> <li>OPF3ECA3</li> <li>OOF3ECA3</li> <li>OOF3ECA4</li> <li>OOF3ECA4</li> <li>OOF3ECA6</li> <li>OOF3ECA6</li> <li>FF7</li> <li>OOF3ECA6</li> </ul>	3 push 3 push es1 7 push 7 5 10 push dwo 5 0C push dwo	rd ptr ss:[ebp+14]	[ebp+C]:L"\\\\?\\C:\`	x87StatusWord 0020 x87Sw_B 0 x87Sw_C x87Sw_C1 0 x87Sw_C x87Sw_SF 0 x87Sw_F	0 0 x875W_ES 0
SIR OOFSECAC FF 1	5 A8 11 F8 00 call dwo	rd ptr ds:[<&CreateFilew>]		Default (stdcall)	▼ 5 🗘 🗆 Unlock
dword ptr [00F811A8 <malware.&creat .text:00F3ECAC malware.exe:\$6ECAC #</malware.&creat 				1: [esp] 0089E79C L 2: [esp+4] 0000080 3: [esp+8] 00000007 4: [esp+C] 0000000 5: [esp+10] 0000000	3
Ump 1 Ump 2 Ump 3	🖽 Dump 4 🛛 👹 Dump 5 🛛 🥮 Watch 1	I Ix=I Locals 🎾 Struct		0089E600 0089E79C L 0089E604 00000080	."////?//C://"
Address   Hex	ASCII		A	0089E6D8 00000007 0089E6DC 00000000	
0089F1D4 00 00 00 00 37 E8 26 64 50 0089F1E4 44 F2 89 00 40 F2 89 00 C4 0089F1F4 00 00 00 00 FF FF FF FF 00 0089F1F4 00 00 00 00 FF FF FF FF 00	F4 89 00 97 DD F6 00 Do'. 00'. A	ô'ŶŎ.		0089E6E0 00000003 0089E6E4 02000000 0089E6E8 00000000	

#### Figure 40

The binary obtains the final path for the drives by calling the GetFinalPathNameByHandleW API:

	00F3EA7D FF 75 14     00F3EA80 FF 75 10     00F3EA83 FF 75 0C     00F3EA86 FF 75 08	push dword ptr ss: ebp+14 push dword ptr ss: ebp+10 push dword ptr ss: ebp+0 push dword ptr ss: ebp+6		x875w_B 0 x875w_C3 0 x8 x875w_C1 0 x875w_C0 0 x8 x875w_SF 0 x875w_P 1 x8	75W_ES O
EIP	COFFEASS FF 15 A4 10 F8 00	call dword ptr ds:[<&GetFinalPathNameByHandlew>]	×	Default (stdcall)	▼ 5 ¢ 🗌 Unlocke
	[OOF810A4 <malware.&getfinalpathnamebyhand]< td=""><td>ew&gt;]=<kernel32.getfinalpathnamebyhandlew></kernel32.getfinalpathnamebyhandlew></td><td></td><td>1: [esp] 00000398 2: [esp+4] 048DC368 3: [esp+8] 00000104 4: [esp+C] 00000000</td><td></td></malware.&getfinalpathnamebyhand]<>	ew>]= <kernel32.getfinalpathnamebyhandlew></kernel32.getfinalpathnamebyhandlew>		1: [esp] 00000398 2: [esp+4] 048DC368 3: [esp+8] 00000104 4: [esp+C] 00000000	

#### Figure 41

It extracts information about the current system using GetNativeSystemInfo (see Figure 42).



#### Figure 42

The ransomware creates the ransom note called "money\_message.log" in every drive. The file contains the chat ID that is specific to a victim and can be used to contact the threat actor:





	Your files was encrypted by "Money message" profitable organization and can't be accessed anymore.
	If you pay ransom, you will get a decryptor to decrypt them. Don't try to decrypt files yourself - in that case they will be damaged and unrecoverable
5	For further negotiations open this client
6	using tor browser https://www.torproject.org/download/
3	In case you refuse to pay, we will post the files we stole from your internal network, in our blog:
0	blogv
2	Encrypted files can't be decrypted without our decryption software.
4	.onion/chat.php?chatId=

Figure 44

The malware creates a new thread that handles the files encryption:

	00F24427 51     00F24428 6A 00     00F24428 6A 00     00F24428 68 <u>C0 D1 F1 00</u> 00F24428 6A 00     00F24432 6A 00	push eck push 0 push 2 push eak push 2 push 2 push 0 cal malware.FDDC0 push 0 cal malware.FSD026		x875tatusword 0020 x875w_50 0 x875w_C3 0 x875w_C2 0 x875w_C1 0 x875w_C0 0 x875w_E5 0 x875w_5F 0 x875w_P 1 x875w_U 0	
CHLA	→• 00F24434 E8 EF 88 03 00 <	Call maiware. FSD028	>	Default (stdcall) 🔻 5	C Unlocke
malware.00F	\$5028 4434 malware.exe:\$\$4434 #53834			11: [esp] 0000000 21: [esp+4] 0000000 31: [esp+4] 00FIDICO malware.00FIDICO 42: [esp+4] 048C7418 51: [esp+10] 00000000	

Figure 45

# Thread activity – sub\_F1D1C0 function

The ransomware opens the directory to encrypt using CreateFileW, as shown in the figure below.

OOP3EC30     S6     OOP3EC4     OOP3EC4     OOP3EC4     GA     OOP3ECA     GA     OOP3ECA     S6     OOP3ECA     S6     OOP3ECA     GA     OOP3ECA     S6     OOP3ECA     S6     OOP3ECA     S6     OO     OO     S     OO     O	push esi push dword ptr ss:[ebp+14] push 3 push 7 push 7 push 7 push dword ptr ss:[ebp+10] push dword ptr ss:[ebp+10] push dword ptr ds: {edgreateFilex}]	[ebp+C]:L"Test"	x87StatusWord 0000 x87SW_B 0 x87SW_C3 0 x8 x87SW_C1 0 x87SW_C0 0 x8 x87SW_SF 0 x87SW_P 0 x8	s (Emply) 575W_C2 0 575W_E5 0 575W_U 0 ▼ 5 € Uniod
<pre>dword ptr [00F811A6 <malware.&createfilew>]=<kern #6e0ac<="" .text:00f3ecac="" malware.exe:\$6ecac="" pre=""></kern></malware.&createfilew></pre>	el32.CreateFilex>	>	1: [esp] 0524F74C L"Test" 2: [esp+4] 0000080 3: [esp+8] 0000007 4: [esp+C] 0000000	
Address   Hex	Dump 5 🛞 Watch 1 I≈-ILocals 🎾 Struct ASCII 0 61 00 ¥.a.1.u.ew.a.		0524F680 0524F74C L"Test" 0524F684 00000080 0524F688 00000007 0524F688 00000000 0524F690 00000000 0524F690 00000003	



The files are enumerated using the FindFirstFileExW and FindNextFileW functions:

	0073EA2A 57     0073EA2B 57     0073EA2C 57     0073EA2C FF 75 10     0073EA2C FF 75 10     0073EA32 FF 75 08     0073EA32 FF 75 08     0073EA32 FF 75 08	push edi push edi push dord ptr ss:[ebp+10] push a push a push dword ptr ss:[ebp+0] call dword ptr ds:[casindirstFileExw>]	[ebp+8]:L"Test\\*"	x875tatusword         0000           x875tw_5         0         x875W_C2         0           x875W_C1         0         x875W_C2         0           x875W_C5         0         x875W_C2         0           x875W_C5         0         x875W_U         0           x875W_C5         0         x875W_U         0           x875W_C5         0         x875W_U         0
C.L.P.	• <	carl dword per ossessmentsterretaws	······ · · · · · · · · · · · · · · · ·	Default (stdcall) - 5 🗘 🗌 Unlock
	[OOF811B0 <malware.dfindfirstfileexw>]=<kerne 3EA35 malware.exe:\$6EA35 #6DE35</kerne </malware.dfindfirstfileexw>	132.FindFirstFileExw>		1: [esp] 0524EEE4 L"Test\\*" 2: [esp+4] 0000001 3: [esp+6] 0524EF00 4: [esp+C] 0000000
Address		ASCII E 8>.x*7.ipipipip	^	0524EE43 0524EE4   L"Test\\*" 0524EE40 00000001 0524EE50 0524EF00 0524EE54 00000000 0524EE54 00000000
043FC300	<u>66 F6 66 F</u>			0524EESC   00000000
		Figure 47		
372	OOF3E9DB FF 75 0C     OOF3E9DE FF 75 08     OOF3E9DE FF 75 08     OOF3E9E1 FF 15 84 11 F8 00	push dword ptr ss: ebp+C push dword ptr ss: ebp+8 call dword ptr ds: <&FindNextFileW>]		x875W_SF 0 x875W_P 0 x875W_U 0
	• <		, , , , , , , , , , , , , , , , , , ,	Default (stdcall)
oword ptr	[OOF81184 <malware.&findnextfilew>]=<kernel32.< td=""><td>PINONEXTRIIEW&gt;</td><td></td><td>2: [esp+4] 0524EF00</td></kernel32.<></malware.&findnextfilew>	PINONEXTRIIEW>		2: [esp+4] 0524EF00

Figure 48

The following files will not be encrypted:

• "desktop.ini" "ntuser.dat" "thumbs.db" "iconcache.db" "ntuser.ini" "ntldr" "bootfont.bin" "ntuser.dat.log" "bootsect.bak" "boot.ini" "autorun.inf"

The malware retrieves attributes for a file to be encrypted by calling the GetFileAttributesExW method:

	<ul> <li>OOF3EB46</li> <li>OOF3EB47</li> <li>OOF3EB49</li> </ul>	50 6A 00 56	push eax push o push esi	es1:L"Test\\test.tx	x875W_C1 0 x675W_C0 0 x875W_ x875W_SF 0 x875W_P 0 x875W_	u o
312	ODE3EB4A	FF 15 88 11 F8 00	call dword ptr ds:[<&GetFileAttributesExw>]	×	Default (stdcall)	▼ 5 C Unlocke
dword ptr [00	F81188 <malware.< td=""><td>&amp;GetFileAttributesExw&gt;]=</td><td><pre><kernel32.getfileattributesexw></kernel32.getfileattributesexw></pre></td><td></td><td>1: [esp] 04402528 L"Test\\test.1 2: [esp+4] 00000000 3: [esp+8] 0524F06C</td><td>txt"</td></malware.<>	&GetFileAttributesExw>]=	<pre><kernel32.getfileattributesexw></kernel32.getfileattributesexw></pre>		1: [esp] 04402528 L"Test\\test.1 2: [esp+4] 00000000 3: [esp+8] 0524F06C	txt"

#### Figure 49

The ECDH public key used is hard-coded in the executable "71828bcd92dd7f5950841835ed0fc926a7f6888be7af4fc07ddc06dd8a1e6400bf3ad4cc27083aad e393c5262570bcbf47aa9855fb9ecc0c82be053c8d95f974a1f2e32d8005e7059461b1e958fecd7dbc 1fa36bf9f9bc8746d431902fe990772d16bab1e779dbe6265444010011708d1091df3838c0a15b0ccf9 9db70e951a74670dd05eb719678d62d150edee22706".

CryptAcquireContextA is utilized to acquire a handle to a key container within a cryptographic service provider (0x1 = **PROV\_RSA\_FULL**, 0xF0000040 = **CRYPT\_VERIFYCONTEXT** | **CRYPT\_SILENT**):

• 00F39554 68 40 00 00 F0 push F0000040 • 00F39556 6A 00 push 1 • 00F39556 6A 00 push 0 • 00F39556 6A 00 push 0 • 00F39557 6A 00 push 0 • 00F39557 8D 45 FC lea eax, dword ptr ss:[[ebp-4]]						x875W_C1 0 x875W_C0 0 x	875W_C2 0 875W_ES 0 875W_U 0
	COF8100C <malware< th=""><th></th><th></th><th>api32.CryptAcquireContextA&gt;</th><th>&gt;</th><th>Default (stdcal) 1: [esp] 0524E6D0 2: [esp+4] 0000000 3: [esp+6] 0000000 4: [esp+C] 0000001</th><th>▼ S C Uniod</th></malware<>			api32.CryptAcquireContextA>	>	Default (stdcal) 1: [esp] 0524E6D0 2: [esp+4] 0000000 3: [esp+6] 0000000 4: [esp+C] 0000001	▼ S C Uniod
Ump 1	Ump 2	ump 3 🗱 Dump 4	Dump 5	😻 Watch 1 🛛 🖉 Struct		0524E68C 0524E6D0 0524E6C0 00000000	
Address H 043EC678 0			00 00 00 00	ASCII	^	0524E6C4 00000000 0524E6C8 00000001 0524E6CC F0000040	

#### Figure 50

<u>CSPRNG</u> is used to generate 0x48 random bytes. These bytes, together with the ECDH public key, will be used to generate the shared secret.





🗾 🛃 🖼	
.text:00F0D7C0	
.text:00F0D7C0	loc F0D7C0:
.text:00F0D7C0	mov eax, edi
.text:00F0D7C2	ror eax, 5
.text:00F0D7C5	sub ecx, eax
.text:00F0D7C7	mov eax, ecx
.text:00F0D7C9	mov ecx, esi
.text:00F0D7CB	ror ecx, 0Fh
.text:00F0D7CE	xor ecx, edi
.text:00F0D7D0	sub [ebp+var 74], 1
.text:00F0D7D4	<pre>lea edi, [esi+edx]</pre>
.text:00F0D7D7	<pre>lea esi, [edx+eax]</pre>
.text:00F0D7DA	<pre>lea edx, [ecx+eax]</pre>
.text:00F0D7DD	jnz short loc_F0D7C0
	<b>F</b>
💶 🚄 🖼	
.text:00F0D7DF mov	eax, [ebp+var_64]
.text:00F0D7E2 mov	dword ptr [ebp+var_88], ecx
.text:00F0D7E8 mov	dword ptr [ebp+var_88+0Ch], edx
.text:00F0D7EB mov	dword ptr [ebp+var_88+8], esi
.text:00F0D7EE mov	dword ptr [ebp+var_88+4], edi
.text:00F0D7F4 mov	[ebp+var_74], eax
.text:00F0D7F7 cmp	eax, [ebp+var_6C]
.text:00F0D7FA jz	short loc_F0D839



The shared secret generated between the public key and the random bytes is 144 bytes long. The elliptic curve is P-384 for the ECDH algorithm.

BOFFOCCFF FF 37     DUED doord ptr ds:[ed1]     OFFCD03 C7 45 FC 00 00 00 00     BOY doord ptr ds:[ed2]     OFFCD03 C7 45 50 FF FF FF 01 00 00 BOY doord ptr ds:[ed2].     OFFCD14 B8 85 59 FF FF FF     BOY eax,dword ptr ds:[ed3]     OFFCD14 FF 30     DOFFCD14	x875tatusword 0000 x875m_8 0 x875m_C3 0 x875m_C2 0 x875m_5C1 0 x875m_C0 0 x875m_E5 0 x875m_5F 0 x875m_P 0 x875m_U 0
	>         Default (stocal)         ▼         5         □         Unlocke           11         [Esp14]         043PD230         22         (Esp14)         043PD230         23         23         (Esp14)         043PD230         33         (Esp14)         043PD230         34         34         34         (Esp14)         043PD230         34
Cestrolovout maintare exertatudi #JLIIC     Forman 1 #JDump 2 #JDump 3 #JDump 4 #JDump 5      Walch 1  i~ Locals      Struct     Ascri     O43P0230 SC A1 36 60 13 28 EC 18 84 A8 A6 33 88 A0 D9 28     O43P0230 SC A3 36 60 13 28 EC 18 64 A8 A6 33 88 A0 D9 28     O43P0230 SC A3 36 60 13 28 EC 18 54 A8     O43P0230 SC A3 36 50 38 28 C6 85 88 C6 84     O43P0230 SC A3 36 50 38 28 C6 85 88 C6 84     O43P0230 SC A3 36 50 38 28 C6 85 88 C6 84     O43P0230 SC A3 36 50 38 28 C6 85 88 C6 84     O43P0230 SC A3 36 50 38 28 C6 85 88 C6 84     O43P0230 SC A3 50 50 88 C6 85 88 C6 84     O43P0230 SC A3 50 50 88 C6 85 80 A8	03447708 04370230 05.447703 0435C3A0 05.447703 0435C3A0 05.447703 0435C3A0 05.242714 00000000 05242714 00000000 05242714 00000090 05242724 04000090 05242724 04000090

Figure 53

Address	He	ĸ															ASCII
043EC8A0	71	82	8B	CD	92	DD	7F	59	50	84	18	35	ED	OF	C9	26	q1.Ý.YP51.É&
043EC8B0	A7	F6	88	8B	E7	AF	4F	CO	7D	DC	06	DD	8A	1E	64	00	§öç OA}Ü.Ýd.
043EC8C0	BF	3A	D4	CC	27	08	3A	AD	E3	93	C5	26	25	70	BC	BF	¿:01'.:. a. A&%p¼¿
043EC8D0	47	AA	98	55	FB	9E	CC	0C	82	BE	05	3C	8D	95	F9	74	Gª.Uû.1%.<ùt
																	¡òãça±éxþ1}
043EC8F0	BC	1F	A3	68	F9	F9	BC	87	46	D4	31	90	2F	E9	90	77	14. £kùù14. FÔ1. /é. w
																	<ul> <li>°±çyÛæ&amp;TDp.</li> </ul>
043EC910	10	91	DF	38	38	CO	A1	5B	OC	CF	99	DB	70	E9	51	A7	B88A;[.I.OpéQ§
043EC920	46	70	DD	05	EB	71	96	78	D6	2D	15	OE	DE	E2	27	06	Fpý.eq.x0þâ'.

Figure 54

Address	He	Hex												ASCII			
043F1138	9B	B1	14	5A	FB	7E	EB	12	5B	D3	38	AS	3B	07	89	F9	.±.Zû~ë.[Ó8 ;ù
043F1148	28	45	F8	14	7F	37	7F	4B	5F	8A	62	<b>B2</b>	F3	F8	<b>B4</b>	E8	(EØ7.Kb=00 e
043F1158	36	6E	<b>6</b> B	26	9A	8D	C5	49	97	D9	CC	6E	67	F2	80	71	6nk&AI.UIngo.c
043F1168	82	10	DA	CO	9B	F5	D5	OE	73	74	1F	02	E5	12	07	39	ÚÀ. ÕÕ. st å 9
043F1178	07	<b>B</b> 7	OE	F9	02	DO	9E	01	26	40	4A	90	37	79	CO	FF	ù.D &@J. 7yAy
																	\$s°&.7[.ÏÉ.u`r
043F1198	85	BE	92	A4	89	63	64	FB	6A	6C	92	D4	78	39	E6	8D	.%.¤'cdûjl.Ôx9æ.
043F11A8	84	E9	<b>B</b> 7	8E	GD	00	25	D1	F3	8A	C6	BA	84	A5	42	CF	. é m. %Ño. Æ . ¥BJ
																	bDª.ýåè%ê.ås)



The SHA384 algorithm implementation is shown in Figure 56. The process computes the hash of the shared secret and copies the first 32 resulting bytes to a new buffer. These bytes represent the ChaCha20 key that will be used to encrypt the file. The nonce (16 bytes) is randomly generated using the same library.

.text:00F397F0	mov	[ebp+var_D8],	8000000h
.text:00F397FA	movaps	xmm0, ds:xmmwd	ord_F84C40
.text:00F39801	mov	[ebp+var_D4],	80008000h
.text:00F3980B	mov	[ebp+var_D0],	8000000h
.text:00F39815	mov	[ebp+var CC],	808Bh
.text:00F3981F	mov	[ebp+var_C8],	0
.text:00F39829	mov	[ebp+var_C4],	80000001h
.text:00F39833	mov	[ebp+var C0],	0
.text:00F3983D	mov	[ebp+var_BC],	80008081h
.text:00F39847	mov	[ebp+var_B8],	80000000h
.text:00F39851	mov	[ebp+var B4],	8009h
.text:00F3985B	mov	[ebp+var_B0],	80000000h
.text:00F39865	mov	[ebp+var AC],	8Ah ; 'Š'
.text:00F3986F	mov	[ebp+var A8],	0
.text:00F39879	mov	[ebp+var_A4],	88h ; '*'
.text:00F39883	mov	[ebp+var A0].	0
.text:00F3988D	mov	[ebp+var 9C],	80008009h
.text:00F39897	mov	[ebp+var_98],	0
.text:00F398A1	mov	[ebp+var 94],	8000000Ah
.text:00F398AB	mov	[ebp+var 90],	0
.text:00F398B5	mov	[ebp+var_8C],	80008088h
.text:00F398BF	mov	[ebp+var 88],	0
.text:00F398C9	mov	[ebp+var 84],	8Bh ; '<'
.text:00F398D3	mov	[ebp+var_80],	80000000h
.text:00F398DA	mov	[ebp+var_7C],	8089h
.text:00F398E1	mov	[ebp+var 78],	80000000h
.text:00F398E8	mov	[ebp+var 74],	8003h
.text:00F398EF	mov	[ebp+var 70],	8000000h
.text:00F398F6	mov	[ebp+var 6C],	8002h
.text:00F398FD	mov	[ebp+var 68],	80000000h
.text:00F39904	mov	[ebp+var 64],	80h ; '€'
.text:00F3990B	mov	[ebp+var_60],	80000000h
.text:00F39912	mov	[ebp+var_5C],	800Ah
.text:00F39919	mov	[ebp+var 58],	0
.text:00F39920	mov	[ebp+var 54],	8000000Ah
.text:00F39927	mov	[ebp+var_50],	80000000h
.text:00F3992E	mov	[ebp+var 4C].	80008081h
.text:00F39935	mov	[ebp+var_48],	80000000h
.text:00F3993C	mov	[ebp+var_44],	8080h
.text:00F39943	mov	[ebp+var 40],	80000000h
.text:00F3994A	mov	[ebp+var 3C],	80000001h
.text:00F39951	mov	[ebp+var 38],	0
.text:00F39958	mov	[ebp+var 34].	80008008h
.text:00F3995F	mov	[ebp+var_30],	80000000h
		[b20]}	

Figure 56

The binary creates an intermediary file by adding the "cbgnfvn" string at the end of the filename (0xC0000000 = GENERIC\_READ | GENERIC\_WRITE, 0x3 = FILE\_SHARE\_READ | FILE\_SHARE\_WRITE, 0x3 = OPEN\_EXISTING, 0x80 = FILE\_ATTRIBUTE\_NORMAL):

OOFSPOIF     GA     OOFSPOIF     GO     OOFSPOIL     SO     OOFSPOIL     SO     OOFSPOIL     SO     OOFSPOIL     SO     OOFSPOIL     SO     OOFSPOIL     FF     S     SO     OOFSPOIL     FF     S     S     S	push 0 push dword ptr ss: ebp+18] push dword ptr ss: ebp+10 push dword ptr ss: ebp+10 push dword ptr ss: ebp+10 push dword ptr ss: ebp+10 call dword ptr ss: ebp-10 call dword ptr sss: ebp-10 call dword	[ebp+8]:L"Test\\tes	Xor/ma_0 a (Lempcy)         Xor/ma_r a (Lempcy)           x875tatusWord 0000         X875M_C2 0           X875M_B 0         X875M_C2 0           X875M_C1 0         X875M_C2 0           X875M_C2 0         X875M_C2 0
dword ptr [00F811A8 <malware.&createfilew>]=<kernel32.cr .text:00F5F0D1 malware.exe:\$8F0D1 #8E4D1</kernel32.cr </malware.&createfilew>	eateF11ew>		1: [esp] 043£C980 L"Test\\test.txtcbgnfvn" 2: [esp+4] 0000000 3: [esp+5] 00000003 4: [esp+C] 0524E4DC
Ump 1         Ump 2         Ump 3         Ump 4         Ump 5           Address         Hex         043714C0         Ber 10         040         102	Watch 1 Ir∝Locals      Struct ASCII    ,,, 10101010     Job Job X.2.5.     Ioo	^	D524E450 0485C980 [LTEst\\test.txtcbgnfvn* 0524E460 00000000 0524E464 00000003 0524E464 0224E400 0524E464 0224E400 0524E470 00000080 0524E474 00000080

Figure 57

The GetFileType method is utilized to obtain the file type:

• 00F5F49E 57	push edi			
STO OFFEASE FF 15 OC 11 F8 00	call dword ptr ds:[<&GetFileType>]	Ň	Default (stdcall)	▼ 5 🗢 🗆 Unlocke
dword ptr [00F5110C <malware.&getfiletype>]=<kernel32< td=""><td>.GetFileType&gt;</td><td></td><td>1: [esp] 00000278 2: [esp+4] 043E07E8</td><td></td></kernel32<></malware.&getfiletype>	.GetFileType>		1: [esp] 00000278 2: [esp+4] 043E07E8	

Figure 58



The ransomware moves the file pointer to the beginning of the file (0x0 = FILE\_BEGIN):



### Figure 59

The file content is read via a function call to ReadFile (see Figure 60).

COFF04D0     C    C    C    C    C    C    C	such b tas eax.dword ptr ss:[ebp-14] push ebx mov ebx.dword ptr ss:[ebp-0] push ebx push ebx push ebx (ebx est ebp-18] call dword ptr ss:[ebp-18] call dword ptr ds:[cdReadFiles]	x8/1m_6 s (Lmpty)         x8/1m_/ s (Lmpty)           x8/57tatusword 0000         x875m_C2           x875m_C1 0         x875m_C2           x875m_C1 0         x875m_C2           x875m_C5 0         x875m_C5           x875m_C1 0         x875m_C9           x875m_C1 0 <td< th=""></td<>
.text:00F604DE malware.exe:\$904DE #8F8DE		2: [esp+4] 45268020 3: [esp+8] 0000004 4: [esp+C]_0524E6A8
Dump 1 Dump 2 Dump 3 Dump 4 Dump 5	🔮 Watch 1 🛛 🖉 Struct	0524E67C 45268020
Address Hex 45268020 00 00 00 00 00 00 00 00 00 00 00 00	ASCII	▲ 0524E680 00000004 0524E684 0524E688 0524E688 0000000

Figure 60

The content is encrypted using the ChaCha20 algorithm:



#### Figure 61

.text:00EFDBB5		
.text:00EFDBB5	loc EFD	885:
.text:00EFDBB5		eax, dword ptr [ebp+var 44]
.text:00EFDBB8	xor	edi, eax
.text:00EFDBBA	mov	ecx, dword ptr [ebp+var 24]
.text:00EFDBBD	rol	edi, 10h
.text:00EFDBC0	add	ecx, edi
.text:00EFDBC2	mov	edx, ecx
.text:00EFDBC4	xor	edx, [ebp+var_48]
.text:00EFDBC7	rol	edx, OCh
.text:00EFDBCA	add	eax, edx
.text:00EFDBCC	mov	[ebp+var_5C], eax
.text:00EFDBCF	xor	eax, edi
.text:00EFDBD1	rol	eax, 8
.text:00EFDBD4	mov	[ebp+var_6C], eax
.text:00EFDBD7	add	eax, ecx
.text:00EFDBD9	mov	[ebp+var_78], eax
.text:00EFDBDC	xor	eax, edx
.text:00EFDBDE	rol	eax, 7
.text:00EFDBE1	mov	[ebp+var_80], eax
.text:00EFDBE4	mov	eax, dword ptr [ebp+var_44+4]
.text:00EFDBE7	add	eax, esi
.text:00EFDBE9	mov	ecx, dword ptr [ebp+var_24+4]
.text:00EFDBEC	mov	edx, eax
.text:00EFDBEE	xor	edx, [ebp+var_4C]
.text:00EFDBF1	rol	edx, 10h
.text:00EFDBF4	add	ecx, edx
.text:00EFDBF6	mov	esi, ecx
.text:00EFDBF8	xor	esi, [ebp+var_58]
.text:00EFDBFB	rol	esi, OCh
.text:00EFDBFE	add	eax, esi
.text:00EFDC00	mov	[ebp+var_4C], eax
.text:00EFDC03	xor	eax, edx
.text:00EFDC05	rol	eax, 8
.text:00EFDC08	mov	[ebp+var_74], eax
.text:00EFDC0B	add	eax, ecx
.text:00EFDC0D	mov	ecx, dword ptr [ebp+var_24+8]
.text:00EFDC10	mov	[ebp+var_88], eax
.text:00EFDC16	xor	eax, esi
.text:00EFDC18	rol	eax, 7



The encrypted file content is written back to the file using WriteFile (Figure 63).

● ODFCOTES     56     push est       ● ODFCOTES     A8     stosd       ● ODFCOTES     A8     stosd       ● ODFCOTES     B0 45 D8     lea eax, dword ptr ssi@ebp-28       ● ODFCOTES     S0 45 D8     push eax       ● ODFCOTES     FT 5 F4     push eax       ● ODFCOTES     FT 5 F8     push eax       ● ODFCOTES     FT 55 C 10 FB 00     Call dword ptr idsi[CauriteFiles]	x8/TM_=*3 (Empty) x8/TM_=*3 (Empty) x8/TM_=6 3 (Empty) x8/TM_7 3 (Empty) x8/TStatusWord 0000 x8/TSM_8 0 x8/TSM_C3 0 x8/TSM_52 0 x8/TSM_510 x8/TSM_20 0 x8/TSM_52 0 x8/TSM_510 x8/TSM_20 0 x8/TSM_52 0 y Default (stdcal) * 5 © Unied
dword ptr [00F8105C «malware.4writeFile>]* <kernel32.writefile> .text:00F60FF4 malware.exe:\$00FF4 #903F4</kernel32.writefile>	2: [esp+4] 45268020 3: [esp+6] 0000004 4: [esp+C] 0524E64C
💭 Dump 1 🐖 Dump 2 🐖 Dump 3 🗱 Dump 4 🐖 Dump 5 🥮 Watch 1 💷 Locals 🎾 Struct	0524E624 00000278 0524E628 45268020
Address Hex ASCII 45265020 <u>8E 22 AD 22</u> 00 00 00 00 00 00 00 00 00 00 00 00 00	

#### Figure 63

The encrypted files extension is changed back to the original after the encryption is complete. The operation is done using MoveFileExW (0x3 = **MOVEFILE\_COPY\_ALLOWED** | **MOVEFILE\_REPLACE\_EXISTING**):



#### Figure 64

The ChaCha20 key is encrypted using ECDH and written to the encrypted file. The ChaCha20 nonce is stored in a non-encrypted form:

FD	test.txt
80	LESLIN

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	OD	0E	OF	
00000000	8E	29	AD	23	90	00	00	00	5A	76	00	50	AE	E7	Dl	6F	Ž). <b>#</b> Zv.₽⊗çÑo
00000010	2A	0F	C8	88	F9	12	DC	ED	33	69	36	E3	CE	47	97	A4	*.È^ù.Üí3i6ãÎG—¤
00000020	ED	2E	0C	51	25	79	95	C7	3F	41	17	E7	38	25	21	DD	1Q%y•Ç?A.ç8%!Ý
00000030	96	C2	2C	<b>B</b> 3	6D	C3	A2	ED	3F	36	FD	26	F8	0C	96	32	-Â, 'mâí?6ý&ø2
00000040	0B	8F	0E	F4	C2	El	A3	4B	88	F8	1B	CA	7D	44	96	06	ôÂá£K^ø.Ê}D
00000050	9F	58	CC	F2	A6	F4	09	26	5F	DC	02	8E	34	4A	lF	4F	ŸXÌò¦ô.&_Ü.Ž4J.O
00000060	C7	CA	2E	5C	6C	AB	34	ED	10	39	7B	14	77	2E	10	48	ÇÊ.\l«41.9{.wH
00000070	8C	lD	0C	51	Fl	71	54	DB	BF	39	49	D8	A8	B0	B9	93	EQñqTÛ;91Ø"°'"
00000080																ED	7ÿ:e,g2]ÿÌ.í
00000090	D9	El	15	EA	64	Fl	FB	07	3F	7F	EC	90	78	08	B3	29	
000000A0	E6	0E	95	DB	AO	5A	83	5C	00	00	00	00	A8	CC	D8	FO	æ.•Û Zf\ïÌØð

Figure 65

### **Running with the --crypt parameter**

The malware only encrypts the directory passed as the argument.

### **Running with the -d parameter**

In this case, the ransomware doesn't stop the target services and processes and doesn't delete the Volume Shadow Copies.

# **Running with the -I parameter**

The process creates a log file called "encrypt\_log.txt" that stores the messages written to the console.

### **Running with the -v parameter**

This is the verbose mode that displays all the intermediary steps during the malware's execution:

C:\Users\ \Desktop>malware.	exe -v
	fo] Termination block will not be skipped, going to standard algo
	of Path to binary: malware.exe
	ifo] Starting the main logic
	ifo] Output to console activated
[2023-08-24 07:48:34.811] [in	ifo] Starting the configure of default logger
[07:48:34 -04:00] [1] [	[thread 268] The work has been started, current version: 1.0.1
[07:48:34 -04:00] [I] [	thread 268] File size: 918660 overlay start location: 915456
[07:48:34 -04:00] [I] [	thread 268] Overlay found, extraction will be started
	[thread 268] Start overlay offset: 915456
[07:48:34 -04:00] [I] [	[thread 268] First creation of mutex 12345-12345-12235-12354 detected
[07:48:34 -04:00] [I] [	[thread 268] Trying to stop service by regex vss
[07:48:34 -04:00] [] [	[thread 268] Trying to stop service by regex sql
[07:48:34 -04:00] [] [	thread 268] Trying to stop service by regex svc\$
[07:48:34 -04:00] [] [	[thread 268] Trying to stop service by regex memtas
[07:48:34 -04:00] [] [	[thread 268] Trying to stop service by regex mepocs
[07:48:34 -04:00] [1] [	[thread 268] Trying to stop service by regex sophos
[07:48:34 -04:00] [] [	[thread 268] Trying to stop service by regex veeam
[07:48:34 -04:00] [] [	[thread 268] Trying to stop service by regex backup
[07:48:34 -04:00] [1] [	[thread 268] Trying to stop service by regex vmms
[07:48:34 -04:00] [] [	[thread 268] Trying to terminate process sql.exe
[07:48:34 -04:00] [I] [	[thread 268] Trying to terminate process oracle.exe
[07:48:34 -04:00] [1] [	thread 268] Trying to terminate process ocssd.exe
[07:48:34 -04:00] [I] [	thread 2681 Trying to terminate process dbsnmp.exe
[07:48:34 -04:00] [1] [	[thread 268] Trying to terminate process synctime.exe
[07:48:34 -04:00] [1] [	thread 268] Trying to terminate process agntsvc.exe
[07:48:34 -04:00] [I] [	thread 268] Trying to terminate process isqlplussvc.exe
[07:48:34 -04:00] [] [	[thread 268] Trying to terminate process xfssvccon.exe
[07:48:34 -04:00] [I] [	[thread 268] Trying to terminate process mydesktopservice.exe
[07:48:34 -04:00] [I] [ [07:48:34 -04:00] [I] [	[thread 268] Trying to terminate process ocautoupds.exe
	thread 268] Trying to terminate process encsyc.exe
	[thread 268] Trying to terminate process firefox.exe
[07:48:34 -04:00] [1] [ [07:48:34 -04:00] [1] [	[thread 268] Trying to terminate process tbirdconfig.exe [thread 268] Trying to terminate process mdesktopqos.exe
	thread 268] Trying to terminate process nocest topqos.exe
	thread 206] Trying to terminate process domaicae
	thread 268] Trying to terminate process selective.exe
	thread 2061 Trying to terminate process successive excelses
	thread 268] Trying to terminate process infopath.exe
	thread 268] Trying to terminate process msaccess.exe
	thread 268] Trying to terminate process matches tere
07:48:34 -04:00] [1] [	thread 2681 Trying to terminate process onenote.exe
	thread 268] Trying to terminate process outlock.exe
07:48:34 -04:00] []	thread 268] Trying to terminate process powerpht.exe
	thread 268] Trying to terminate process steam.exe
	thread 268] Trying to terminate process thebat.exe
[07:48:34 -04:00] [] [	thread 268] Trying to terminate process thunderbird.exe
	thread 268] Trying to terminate process visio.exe
	thread 268] Trying to terminate process winword.exe
[07:48:34 -04:00] [1] [	thread 268] Trying to terminate process wordpad.exe
[07:48:34 -04:00] [1] [	thread 268] Trying to terminate process vmms.exe
	thread 268] Trying to terminate process vmwp.exe
	thread 268] Trying to remove shadow copies
	thread 268] Unable to call ShellExecutew for removing shadow copies, description The operation completed successfully,

Figure 66

SecurityScorecard

# Indicators of Compromise

### **SHA256**

8be41efd6e6ace53b8c59344be2ba91fe41003987a8e38484b20760d7c400a42

### Money Message Ransom Note

money\_message.log

### Mutex

12345-12345-12235-12354

### **Process spawned**

cmd.exe /c vssadmin.exe delete shadows /all /quiet

