

A detailed analysis of the Money Message Ransomware

Prepared by: Vlad Pasca, Senior Malware &
Threat Analyst



[SecurityScorecard.com](https://www.securityscorecard.com)
info@securityscorecard.com

Tower 49
12 E 49th Street
Suite 15-001
New York, NY 10017
[1.800.682.1707](tel:18006821707)

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 the --crypt parameter	17
Running with the -d parameter	17
Running with the -l parameter	18
Running with the -v parameter	18
Indicators of Compromise	19

Executive summary

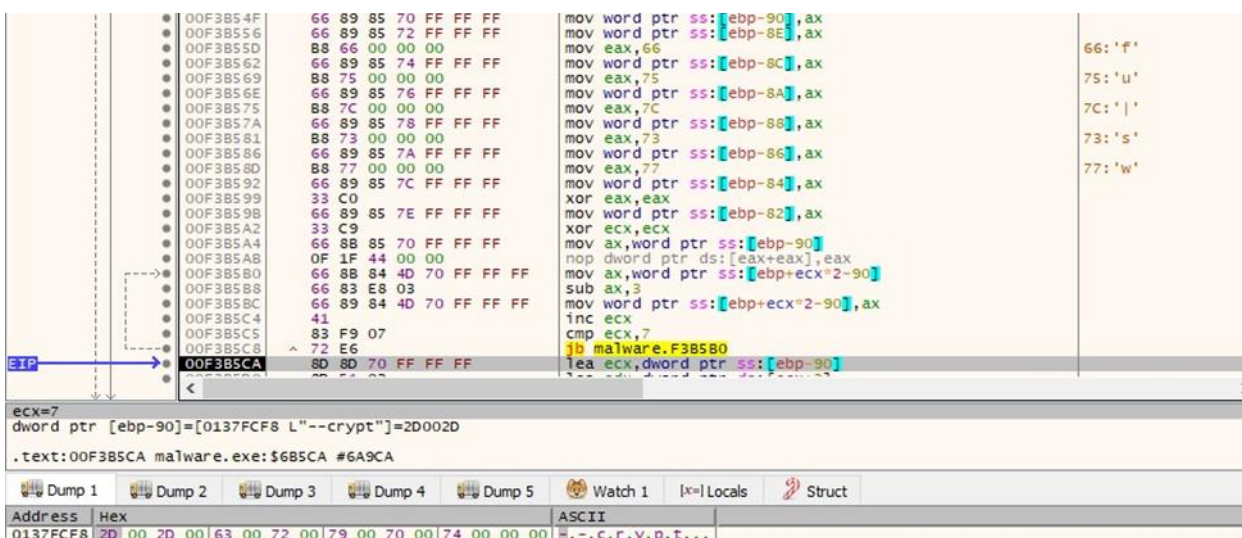
The threat actor group, Money Message ransomware, first appeared in March 2023, demanding million-dollar ransoms from its targets. Its configuration, which contains the services and processes to stop a ransomware attack, can be found at the end of the executable. The ransomware creates a mutex and deletes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



```
00F3B54F 66 89 85 70 FF FF FF mov word ptr ss:[ebp-90],ax
00F3B556 66 89 85 72 FF FF FF mov word ptr ss:[ebp-8E],ax
00F3B55D 88 66 00 00 00 00 mov eax,66
00F3B562 66 89 85 74 FF FF FF mov word ptr ss:[ebp-8C],ax
00F3B569 88 75 00 00 00 00 mov eax,75
00F3B56E 66 89 85 76 FF FF FF mov word ptr ss:[ebp-8A],ax
00F3B575 88 7C 00 00 00 00 mov eax,7C
00F3B57A 66 89 85 78 FF FF FF mov word ptr ss:[ebp-88],ax
00F3B581 88 73 00 00 00 00 mov eax,73
00F3B586 66 89 85 7A FF FF FF mov word ptr ss:[ebp-86],ax
00F3B58D 88 77 00 00 00 00 mov eax,77
00F3B592 66 89 85 7C FF FF FF mov word ptr ss:[ebp-84],ax
00F3B599 33 C0 xor eax,eax
00F3B59B 66 89 85 7E FF FF FF mov word ptr ss:[ebp-82],ax
00F3B5A2 33 C9 xor ecx,ecx
00F3B5A4 66 8B 85 70 FF FF FF mov ax,word ptr ss:[ebp-90]
00F3B5AB 0F 1F 44 00 00 00 nop dword ptr ds:[eax+eax],eax
00F3B5B0 66 8B 84 4D 70 FF FF FF mov ax,word ptr ss:[ebp+ecx*2-90]
00F3B5B8 66 83 E8 03 sub ax,3
00F3B5BC 66 89 84 4D 70 FF FF FF mov word ptr ss:[ebp+ecx*2-90],ax
00F3B5C4 41 inc ecx
00F3B5C5 83 F9 07 cmp ecx,7
00F3B5C8 72 E6 jb malware.F3B5B0
00F3B5CA 8D 8D 70 FF FF FF lea ecx,dword ptr ss:[ebp-90]
```

ecx=7
dword ptr [ebp-90]=[0137FCF8 L"--crypt"]=2D002D
.text:00F3B5CA malware.exe:\$6B5CA #6A9CA

Address	Hex	ASCII
0137FCF8	2D 00 2D 00 63 00 72 00 79 00 70 00 74 00 00 00	m.-.c.r.y.p.t...

Figure 1

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

```

00EFCCE7  C6 45 A1 70      mov byte ptr ss:[ebp-5F],70
00EFCCEB  C6 45 A2 24      mov byte ptr ss:[ebp-5E],24
00EFCCEC  C6 45 A3 72      mov byte ptr ss:[ebp-5D],72
00EFCCE3  C6 45 A4 73      mov byte ptr ss:[ebp-5C],73
00EFCED7  C6 45 A5 78      mov byte ptr ss:[ebp-5B],78
00EFCEDB  C6 45 A6 24      mov byte ptr ss:[ebp-5A],24
00EFCEDF  C6 45 A7 66      mov byte ptr ss:[ebp-59],66
00EFCCE3  C6 45 A8 69      mov byte ptr ss:[ebp-58],69
00EFCCE7  C6 45 A9 24      mov byte ptr ss:[ebp-57],24
00EFCCEB  C6 45 AA 77      mov byte ptr ss:[ebp-56],77
00EFCCE7  C6 45 AB 6F      mov byte ptr ss:[ebp-55],6F
00EFCCE3  C6 45 AC 6D      mov byte ptr ss:[ebp-54],6D
00EFCCE7  C6 45 AD 74      mov byte ptr ss:[ebp-53],74
00EFCCEB  C6 45 AE 74      mov byte ptr ss:[ebp-52],74
00EFCCE7  C6 45 AF 69      mov byte ptr ss:[ebp-51],69
00EFCF03  C6 45 B0 68      mov byte ptr ss:[ebp-50],68
00EFCF07  C6 45 B1 30      mov byte ptr ss:[ebp-4F],30
00EFCF0B  C6 45 B2 24      mov byte ptr ss:[ebp-4E],24
00EFCF0F  C6 45 B3 6B      mov byte ptr ss:[ebp-4D],6B
00EFCF13  C6 45 B4 73      mov byte ptr ss:[ebp-4C],73
00EFCF17  C6 45 B5 6D      mov byte ptr ss:[ebp-4B],6D
00EFCF1B  C6 45 B6 72      mov byte ptr ss:[ebp-4A],72
00EFCF1F  C6 45 B7 6B      mov byte ptr ss:[ebp-49],6B
00EFCF23  C6 45 B8 24      mov byte ptr ss:[ebp-48],24
00EFCF27  C6 45 B9 78      mov byte ptr ss:[ebp-47],78
00EFCF2B  C6 45 BA 73      mov byte ptr ss:[ebp-46],73
00EFCF2F  C6 45 BB 24      mov byte ptr ss:[ebp-45],24
00EFCF33  C6 45 BC 77      mov byte ptr ss:[ebp-44],77
00EFCF37  C6 45 BD 78      mov byte ptr ss:[ebp-43],78
00EFCF3B  C6 45 BE 65      mov byte ptr ss:[ebp-42],65
00EFCF3F  C6 45 BF 72      mov byte ptr ss:[ebp-41],72
00EFCF43  C6 45 C0 68      mov byte ptr ss:[ebp-40],68
00EFCF47  C6 45 C1 65      mov byte ptr ss:[ebp-3F],65
00EFCF4B  C6 45 C2 76      mov byte ptr ss:[ebp-3E],76
00EFCF4F  C6 45 C3 68      mov byte ptr ss:[ebp-3D],68
00EFCF53  C6 45 C4 24      mov byte ptr ss:[ebp-3C],24
00EFCF57  C6 45 C5 65      mov byte ptr ss:[ebp-3B],65
00EFCF5B  C6 45 C6 70      mov byte ptr ss:[ebp-3A],70
00EFCF5F  C6 45 C7 6B      mov byte ptr ss:[ebp-39],6B
00EFCF63  C6 45 C8 73      mov byte ptr ss:[ebp-38],73
00EFCF67  C6 45 C9 00      mov byte ptr ss:[ebp-37],0
00EFCF6B  BA 45 8C          mov al,byte ptr ss:[ebp-74]
00EFCF6E  66 90            nop
00EFCF70  8A 44 0D 8C      mov al,byte ptr ss:[ebp+ecx-74]
00EFCF74  2C 04            sub al,4
00EFCF76  88 44 0D 8C      mov byte ptr ss:[ebp+ecx-74],al
00EFCF7A  41              inc ecx
00EFCF7B  83 F9 3D        cmp ecx,3D
00EFCF7B  8D 45 8C        jnb malware.EFCF70
00EFCF80  8D 45 8C        lea eax,dword ptr ss:[ebp-74]

```

eax=0373506F
dword ptr [ebp-74]=[0137FE40 "Termination block will not be skipped, going to standard algo"]=6D726554
.text:00EFCF80 malware.exe:\$2CF80 #2C380

Address	Hex	ASCII
0137FE40	54 65 72 6D 69 6E 61 74	Termination bloc
0137FE50	68 20 77 69 6C 6C 20 6E	k will not be sk
0137FE60	69 70 70 65 64 2C 20 67	ipped, going to
0137FE70	73 74 61 6E 64 61 72 64	standard algo.o.

Figure 2

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

```

00F41715  FF 75 08      push dword ptr ss:[ebp+8]
00F41718  85 F6        test esi,esi
00F4171A  3E malware.F41728
00F4171C  8B CE        mov ecx,esi
00F4171E  FF 15 28 12 F8 00
00F41724  FF D6        call dword ptr ds:[F81298]

```

esi=kernel32.GetSystemTimePreciseAsFileTime- (76ACE280)
.text:00F41724 malware.exe:\$71724 #70B24

Address	Hex	ASCII
0137FD58	E0 76 73 03 E0 76 73 03	Bvs. Avs. Eyr. U.1.

Figure 3

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

```

00EE6F66  FF 15 70 10 F8 00
call dword ptr ds:[<GetCurrentThreadId>]

```

dword ptr [00F81070 <malware.&GetCurrentThreadId>]=<kernel32.GetCurrentThreadId>

Figure 4

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

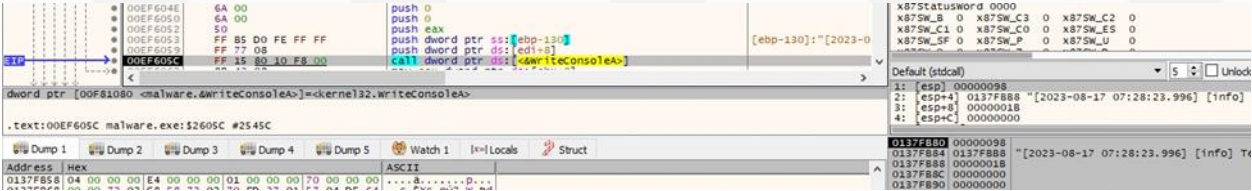


Figure 5

It extracts information about the console screen buffer using GetConsoleScreenBufferInfo:

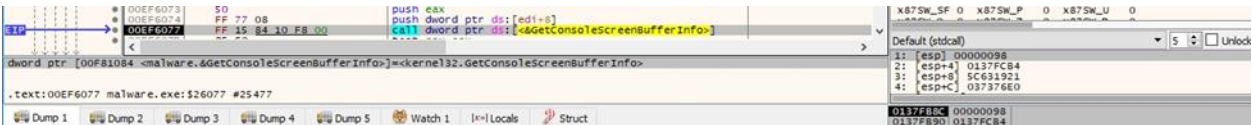


Figure 6

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



Figure 7

The executable file is opened by calling the wflopen method (0x40 = **_SH_DENYNO**):



Figure 8

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

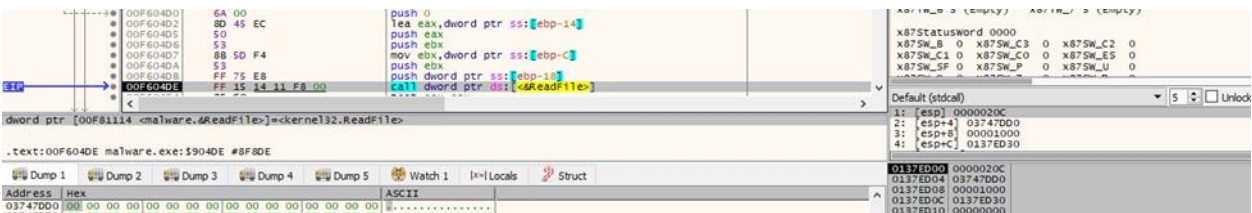


Figure 9

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:

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



Figure 13

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



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**):

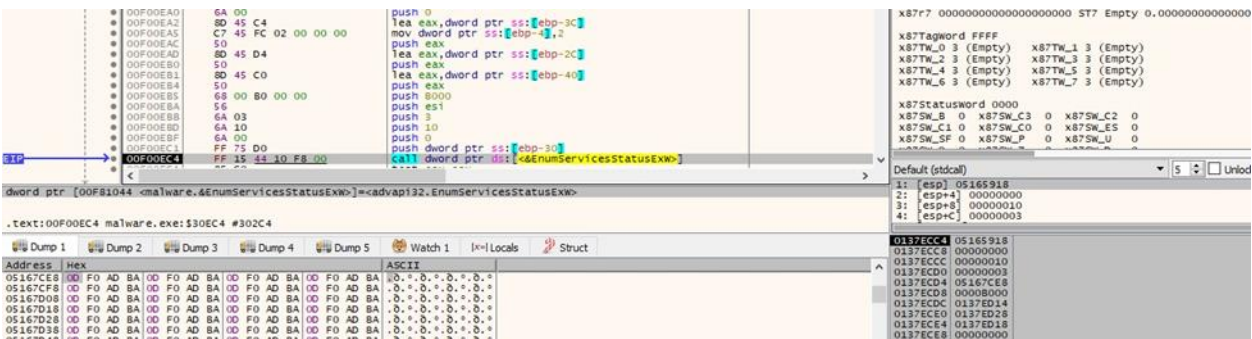


Figure 15

The following services will be stopped:

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



Figure 16

The malware takes a snapshot of all processes via a call to CreateToolhelp32Snapshot (0x2 = TH32CS_SNAPPROCESS):

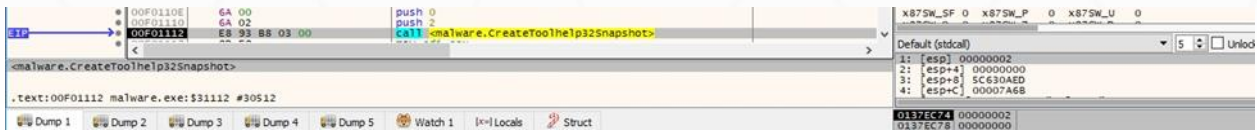


Figure 17

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



Figure 18



Figure 19

The ransomware opens the target processes using the OpenProcess method (0x1 = PROCESS_TERMINATE):



Figure 20

The following processes will be killed:

- "sql.exe" "oracle.exe" "ocssd.exe" "dbnmp.exe" "synctime.exe" "agntsvc.exe" "isqlplussvc.exe" "xfssvcon.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"



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.

Figure 22

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

Figure 23

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

Figure 24

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

Figure 25

A new thread is created, which runs the sub_ED2770 function even if the argument passed to CreateThread is the StartAddress function:

Figure 26

Figure 27

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:

Figure 30

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

Figure 31

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

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.

Figure 34

Thread activity – sub_F179D0 function

The executable extracts a pseudo handle for the current thread:

Figure 35

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

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.



Figure 38

We continue to analyze the main thread.

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

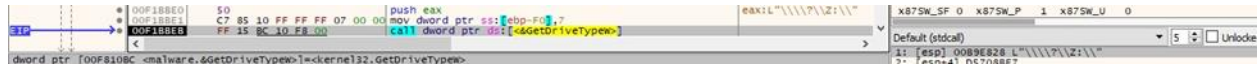


Figure 39

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**):

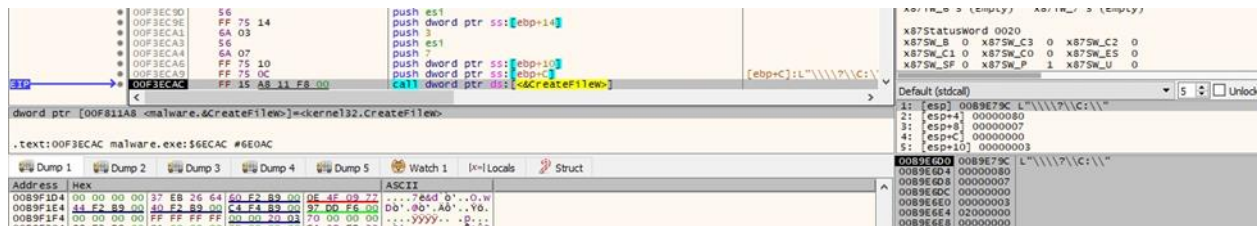


Figure 40

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



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:

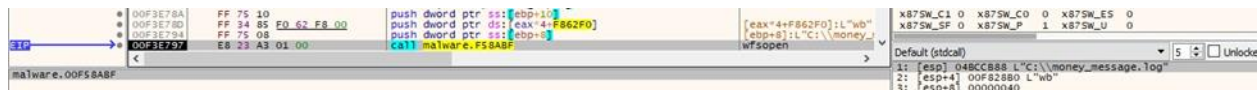


Figure 43

```

1 Your files was encrypted by "Money message" profitable organization and can't be accessed anymore.
2
3 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.
4
5 For further negotiations open this client: [REDACTED].onion/chat.php?chatId=[REDACTED]
6 using tor browser https://www.torproject.org/download/
7
8 In case you refuse to pay, we will post the files we stole from your internal network, in our blog:
9
10 blog: [REDACTED].onion
11
12 Encrypted files can't be decrypted without our decryption software.
13
14 client: [REDACTED].onion/chat.php?chatId=[REDACTED]

```

Figure 44

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

Figure 45

Thread activity – sub_FID1C0 function

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

Figure 46

The files are enumerated using the FindFirstFileExW and FindNextFileW functions:

Figure 47

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:



Figure 49

The ECDH public key used is hard-coded in the executable "71828bcd92dd7f5950841835ed0fc926a7f6888be7af4fc07ddc06dd8a1e6400bf3ad4cc27083aad e393c5262570bcbf47aa9855fb9ecc0c82be053c8d95f974a1f2e32d8005e7059461b1e958fec7dbc 1fa36bf9fbc8746d431902fe990772d16bab1e779dbe6265444010011708d1091df3838c0a15b0ccf9 9db70e951a74670dd05eb719678d62d150edee22706".

CryptAcquireContextA is utilized to acquire a handle to a key container within a cryptographic service provider (0x1 = **PROV_RSA_FULL**, 0xF000040 = **CRYPT_VERIFYCONTEXT** | **CRYPT_SILENT**):



Figure 50

[CSPRNG](#) is used to generate 0x48 random bytes. These bytes, together with the ECDH public key, will be used to generate the shared secret.



Figure 51

```

.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 lea    edi, [esi+edx]
.text:00F0D7D7 lea    esi, [edx+eax]
.text:00F0D7DA lea    edx, [ecx+eax]
.text:00F0D7DD jnz    short loc_F0D7C0

.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

```

Figure 52

The shared secret generated between the public key and the random byte is 144 bytes long. The elliptic curve is [P-384](#) for the ECDH algorithm.

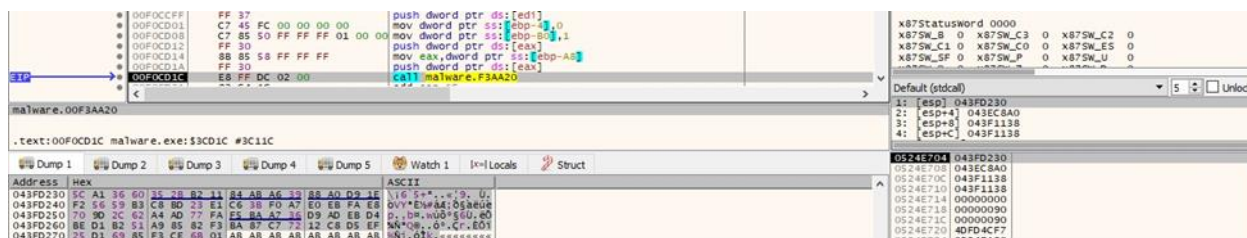


Figure 53

Address	Hex	ASCII
043EC8A0	71 82 8B CD 92 DD 7F 59 50 84 18 35 ED 0F C9 26	q..I.Y.YP..5f.E&
043EC8B0	A7 F6 88 8B E7 AF 4F C0 7D DC 06 DD 8A 1E 64 00	\$ö..ç'OA'Ü.Y..d
043EC8C0	BF 3A D4 C8 27 08 3A AD E3 93 C5 26 25 70 BC BF	?:öI'...ä.A&#p%ç
043EC8D0	47 AA 98 55 FB 9E CC 0C 82 BE 05 3C 8D 95 F9 74	G^..Uü.I..%.<..üt
043EC8E0	A1 F2 E3 2D 80 05 E7 05 94 61 B1 E9 58 FE CD 7D	iöä..ç..æéXpI}
043EC8F0	BC 1F A3 68 F9 F9 BC 87 46 D4 31 90 2F E9 90 77	%..Éküü%..F01./é.w
043EC900	2D 16 BA 81 E7 79 DB E6 26 54 44 01 00 11 70 8D	-.?+çyÜæ&TD...p.
043EC910	10 91 DF 38 38 C0 A1 58 0C CF 99 DB 70 E9 51 A7	..888A [.I.ÜpéQ\$
043EC920	46 70 DD 05 EB 71 96 78 D6 2D 15 0E DE E2 27 06	FpY..éq.Xö-..pä'.

Figure 54

Address	Hex	ASCII
043F1138	98 B1 14 5A FB 7F EB 12 58 D3 38 A8 38 07 89 F9	.±.ZÜ-è.[08 ;..u
043F1148	28 45 F8 14 7F 37 7F 48 5F 8A 62 B2 F3 F8 B4 E8	(Eo..7.K..b°ö'e
043F1158	36 6E 68 26 9A 8D C5 49 97 D9 CC 6E 67 F2 80 71	6nk&..AI.ÜIngò.q
043F1168	82 1C DA C0 98 F5 D5 0E 73 74 1F 02 E5 12 07 39	..ÜA..ö0.st..ä..9
043F1178	07 B7 0E F9 02 D0 9E 01 26 40 4A 9C 37 79 C0 FF	..ü.D..&@j.7yAy
043F1188	1C 13 24 73 BA 26 1D 37 58 03 CF C9 09 75 60 72	..\$\$°&.7'.IE..u r
043F1198	85 BE 92 A4 B9 63 64 FB 6A 6C 92 D4 78 39 E6 8D	%.M'cdüj].Öx9æ.
043F11A8	84 E9 B7 8E 6D 00 25 D1 F3 8A C6 BA 84 A5 42 CF	.é..m.%No.£°.¥BI
043F11B8	62 44 AA 7F FD E5 E8 BE EA 84 E5 73 29 10 9E 05	bD^..yâé%è..às)...

Figure 55

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], 80000000h
.text:00F397FA movaps  xmm0, ds:xmmword_F84C40
.text:00F39801 mov     [ebp+var_D4], 80008000h
.text:00F3980B mov     [ebp+var_D0], 80000000h
.text:00F39815 mov     [ebp+var_CC], 8088h
.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], 8008081h
.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 ; 'S'
.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], 8008009h
.text:00F39897 mov     [ebp+var_98], 0
.text:00F398A1 mov     [ebp+var_94], 800000Ah
.text:00F398AB mov     [ebp+var_90], 0
.text:00F398B5 mov     [ebp+var_8C], 8008088h
.text:00F398BF mov     [ebp+var_88], 0
.text:00F398C9 mov     [ebp+var_84], 88h ; '<'
.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], 80000000h
.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], 800000Ah
.text:00F39927 mov     [ebp+var_50], 80000000h
.text:00F3992E mov     [ebp+var_4C], 8008081h
.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], 80080000h
.text:00F3995F mov     [ebp+var_30], 80000000h

```

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**):



Figure 57

The `GetFileType` method is utilized to obtain the file type:

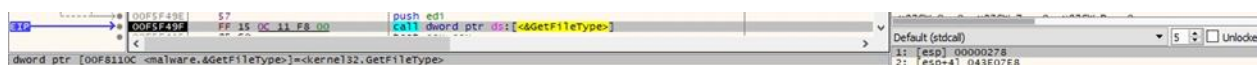


Figure 58

The ransomware moves the file pointer to the beginning of the file (0x0 = **FILE_BEGIN**):

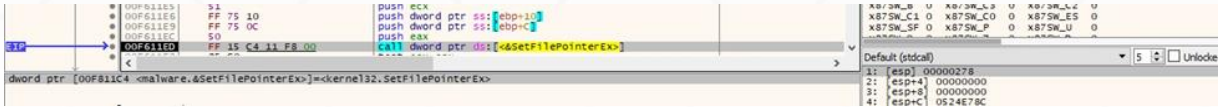


Figure 59

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

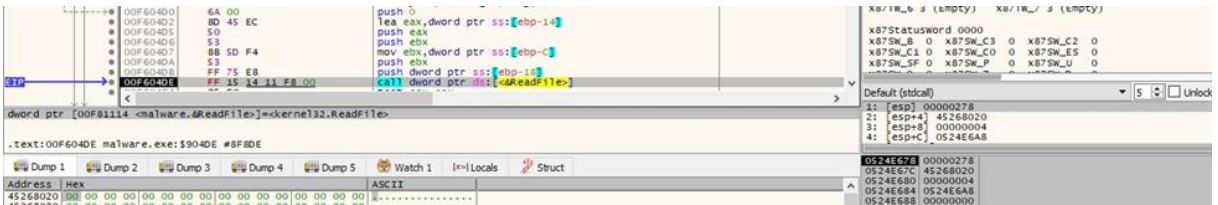


Figure 60

The content is encrypted using the ChaCha20 algorithm:

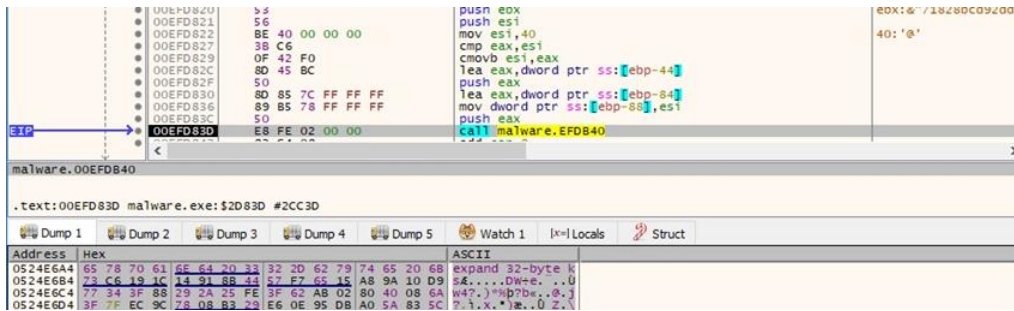


Figure 61

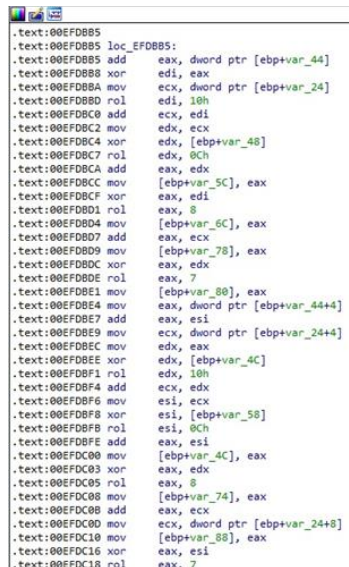


Figure 62

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



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:

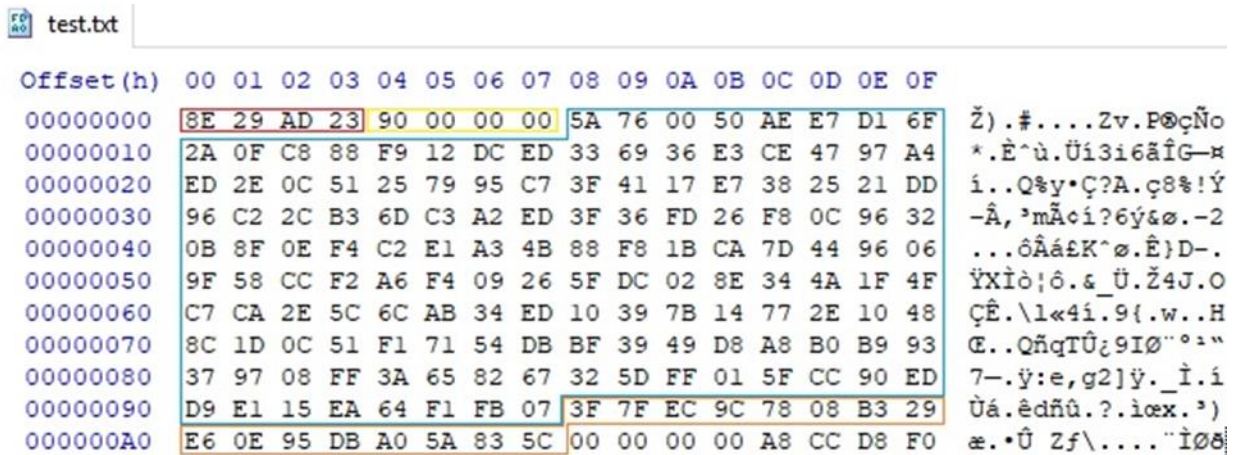


Figure 65

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 -l 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
[2023-08-24 07:48:34.806] [info] Termination block will not be skipped, going to standard algo
[2023-08-24 07:48:34.810] [info] Path to binary: malware.exe
[2023-08-24 07:48:34.810] [info] Starting the main logic
[2023-08-24 07:48:34.811] [info] Output to console activated
[2023-08-24 07:48:34.811] [info] Starting the configure of default logger
[07:48:34 -04:00] [info] [thread 268] The work has been started, current version: 1.0.1
[07:48:34 -04:00] [info] [thread 268] File size: 918660 overlay start location: 915456
[07:48:34 -04:00] [info] [thread 268] Overlay Found, extraction will be started
[07:48:34 -04:00] [info] [thread 268] Start overlay offset: 915456
[07:48:34 -04:00] [info] [thread 268] First creation of mutex 12345-12345-12235-12354 detected
[07:48:34 -04:00] [info] [thread 268] Trying to stop service by regex vss
[07:48:34 -04:00] [info] [thread 268] Trying to stop service by regex sql
[07:48:34 -04:00] [info] [thread 268] Trying to stop service by regex svc$
[07:48:34 -04:00] [info] [thread 268] Trying to stop service by regex memtas
[07:48:34 -04:00] [info] [thread 268] Trying to stop service by regex mepocs
[07:48:34 -04:00] [info] [thread 268] Trying to stop service by regex sophos
[07:48:34 -04:00] [info] [thread 268] Trying to stop service by regex veeam
[07:48:34 -04:00] [info] [thread 268] Trying to stop service by regex backup
[07:48:34 -04:00] [info] [thread 268] Trying to stop service by regex vmms
[07:48:34 -04:00] [info] [thread 268] Trying to terminate process sql.exe
[07:48:34 -04:00] [info] [thread 268] Trying to terminate process oracle.exe
[07:48:34 -04:00] [info] [thread 268] Trying to terminate process ocsd.exe
[07:48:34 -04:00] [info] [thread 268] Trying to terminate process dbsnmp.exe
[07:48:34 -04:00] [info] [thread 268] Trying to terminate process synctime.exe
[07:48:34 -04:00] [info] [thread 268] Trying to terminate process agnsvcs.exe
[07:48:34 -04:00] [info] [thread 268] Trying to terminate process isqlplusvc.exe
[07:48:34 -04:00] [info] [thread 268] Trying to terminate process xfssvccn.exe
[07:48:34 -04:00] [info] [thread 268] Trying to terminate process mydesktopservice.exe
[07:48:34 -04:00] [info] [thread 268] Trying to terminate process ocautoupds.exe
[07:48:34 -04:00] [info] [thread 268] Trying to terminate process encsvc.exe
[07:48:34 -04:00] [info] [thread 268] Trying to terminate process firefox.exe
[07:48:34 -04:00] [info] [thread 268] Trying to terminate process tbirdconfig.exe
[07:48:34 -04:00] [info] [thread 268] Trying to terminate process mdesktopqos.exe
[07:48:34 -04:00] [info] [thread 268] Trying to terminate process ocomm.exe
[07:48:34 -04:00] [info] [thread 268] Trying to terminate process dbeng50.exe
[07:48:34 -04:00] [info] [thread 268] Trying to terminate process sqbcoreservice.exe
[07:48:34 -04:00] [info] [thread 268] Trying to terminate process excel.exe
[07:48:34 -04:00] [info] [thread 268] Trying to terminate process infopath.exe
[07:48:34 -04:00] [info] [thread 268] Trying to terminate process msaccess.exe
[07:48:34 -04:00] [info] [thread 268] Trying to terminate process mspub.exe
[07:48:34 -04:00] [info] [thread 268] Trying to terminate process onenote.exe
[07:48:34 -04:00] [info] [thread 268] Trying to terminate process outlook.exe
[07:48:34 -04:00] [info] [thread 268] Trying to terminate process powerpnt.exe
[07:48:34 -04:00] [info] [thread 268] Trying to terminate process steam.exe
[07:48:34 -04:00] [info] [thread 268] Trying to terminate process thebat.exe
[07:48:34 -04:00] [info] [thread 268] Trying to terminate process thunderbird.exe
[07:48:34 -04:00] [info] [thread 268] Trying to terminate process visio.exe
[07:48:34 -04:00] [info] [thread 268] Trying to terminate process winword.exe
[07:48:34 -04:00] [info] [thread 268] Trying to terminate process wordpad.exe
[07:48:34 -04:00] [info] [thread 268] Trying to terminate process vmms.exe
[07:48:34 -04:00] [info] [thread 268] Trying to terminate process vmmp.exe
[07:48:34 -04:00] [info] [thread 268] Trying to remove shadow copies
[07:48:35 -04:00] [info] [thread 268] Unable to call ShellExecuteW for removing shadow copies, description The operation completed successfully.
```

Figure 66

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