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 49th Street
Suite 15-001
New York, NY 10017
1.800.682.1707

Table of contents

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.

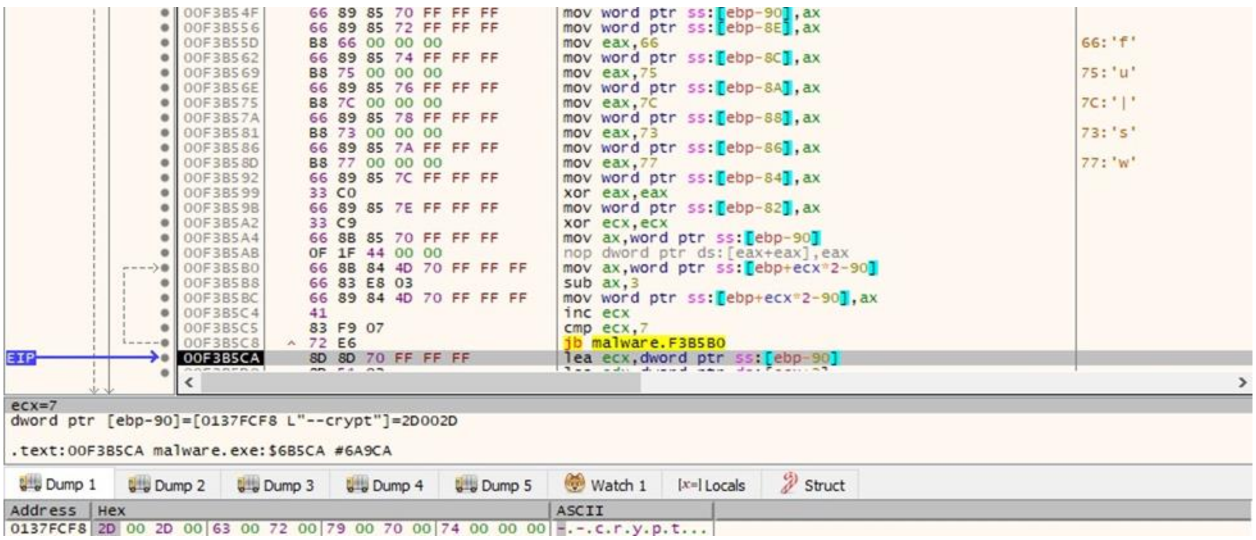


Figure 1

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

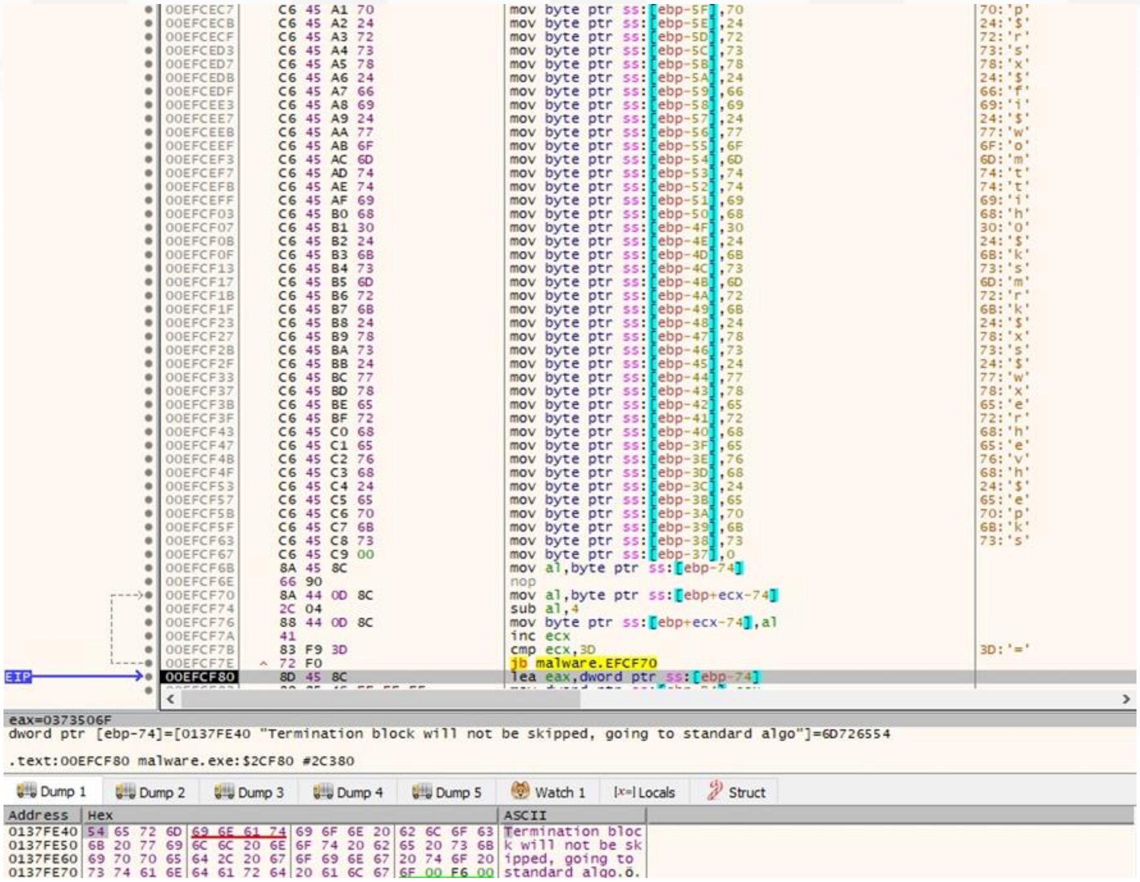


Figure 2

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

Figure 3

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

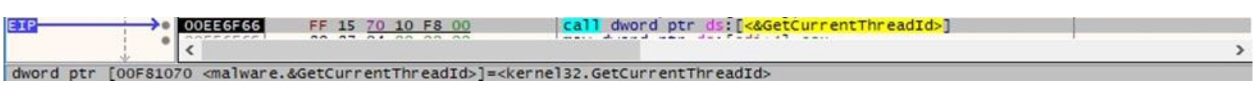


Figure 4

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:

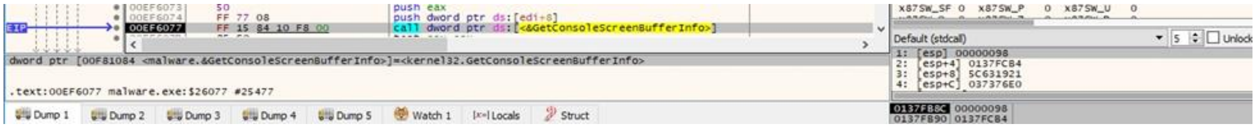


Figure 6

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 wfsopen 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:

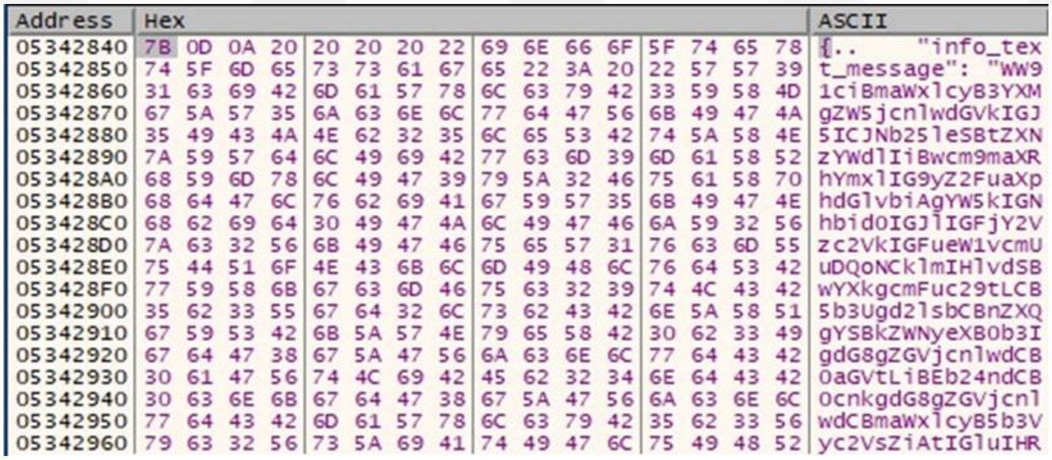


Figure 10

Figure 11

The directories extracted from the “skip_directories” field are Base64-decoded (Figure 12).

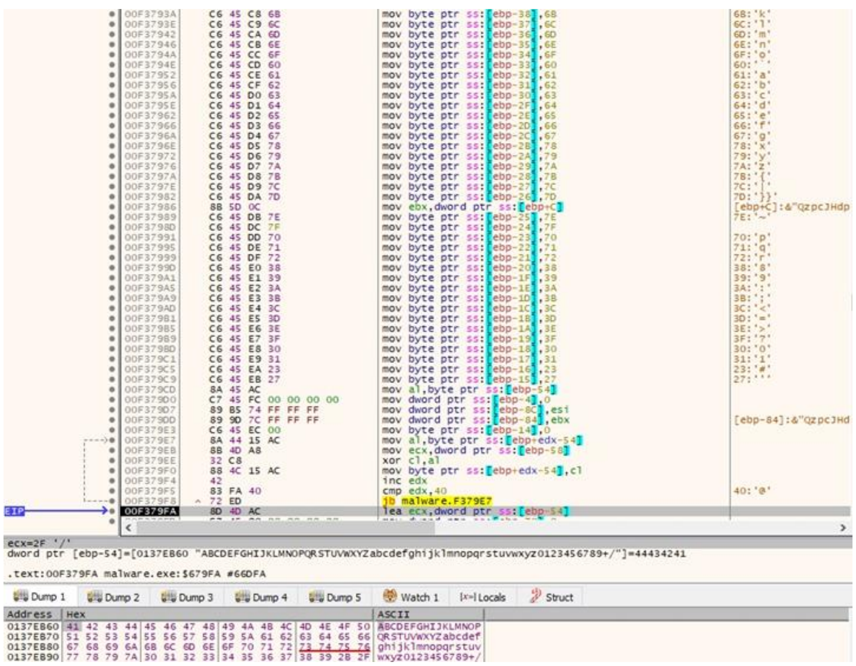


Figure 12

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**):

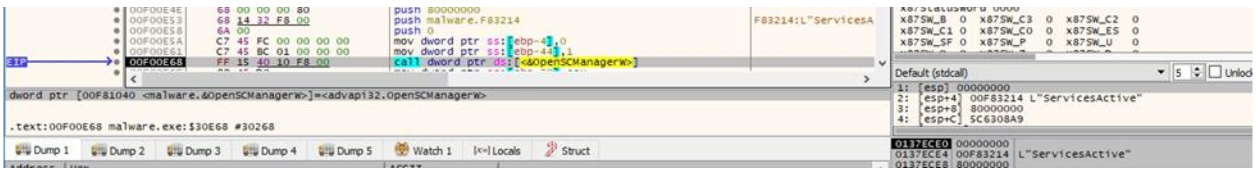


Figure 14

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**):

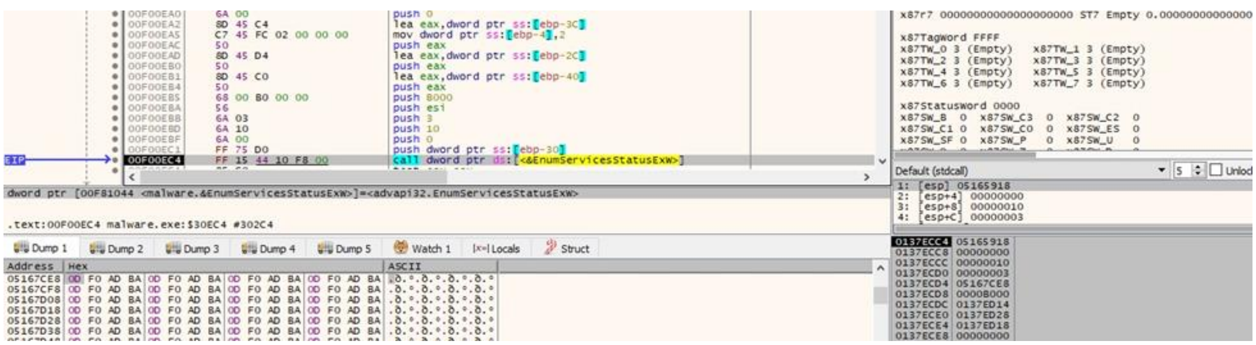


Figure 15

The following services will be stopped:

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

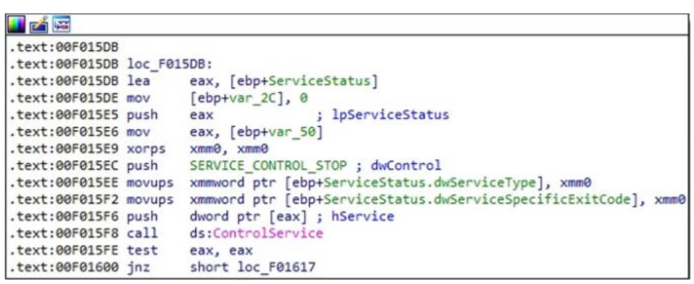


Figure 16

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:

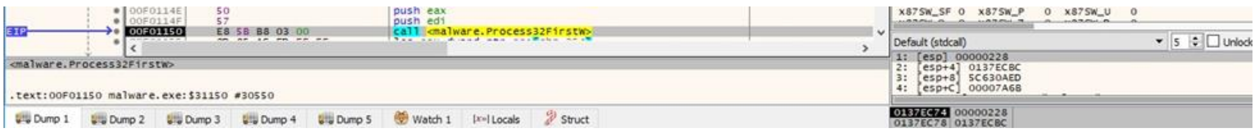


Figure 18

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" "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"

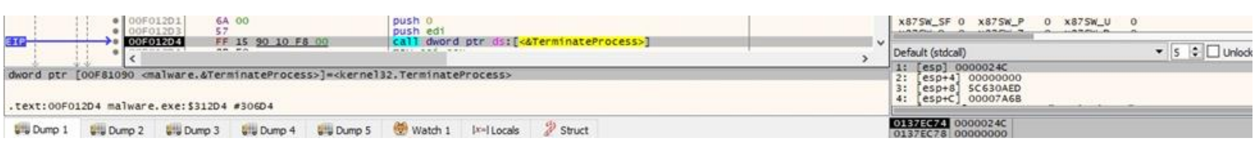


Figure 21

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.

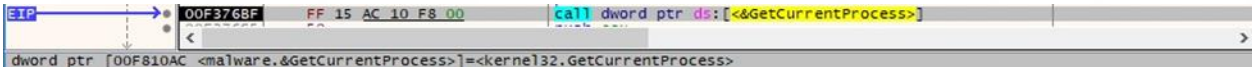


Figure 22

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).

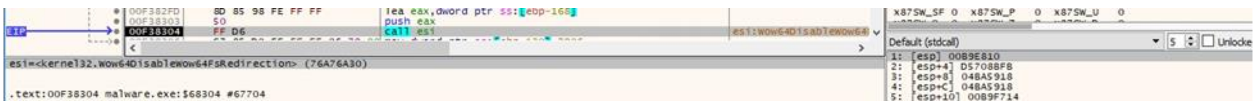


Figure 24

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

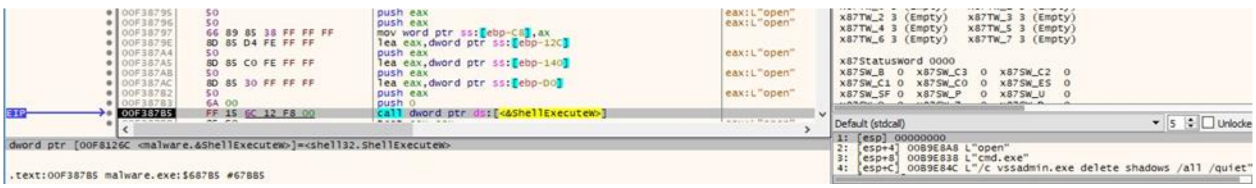


Figure 25

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

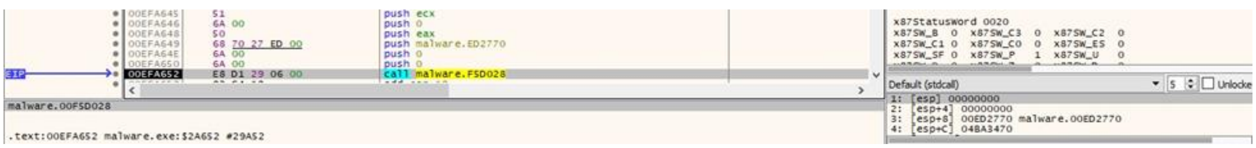


Figure 26

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).

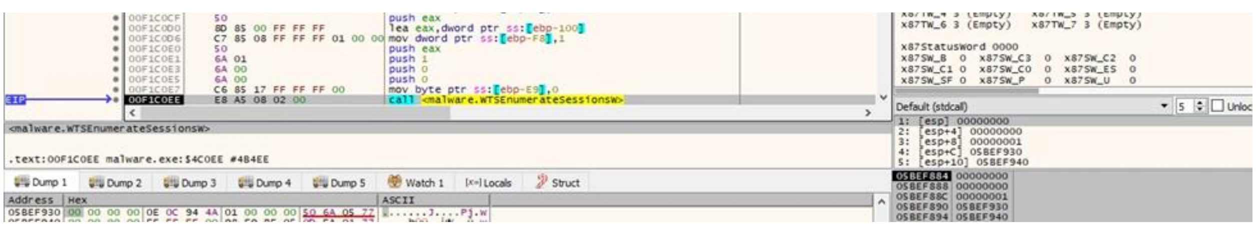


Figure 31

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:

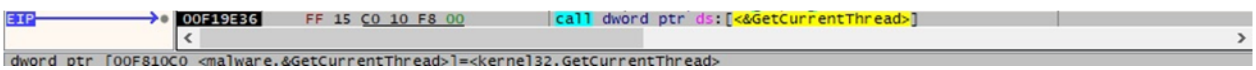


Figure 35

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**):

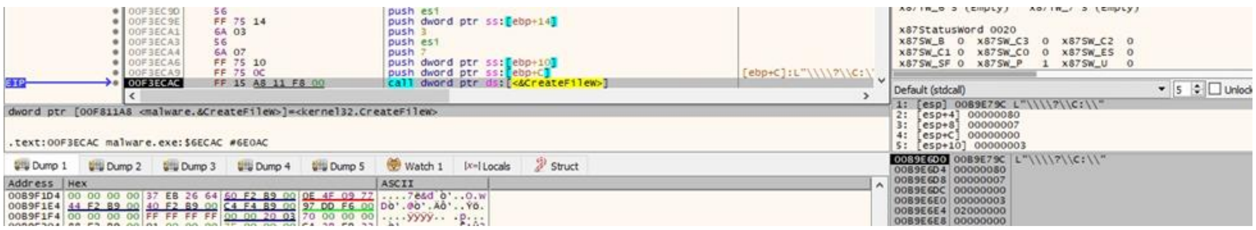


Figure 40

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).

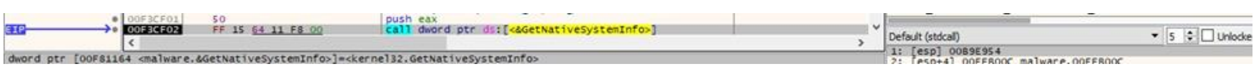
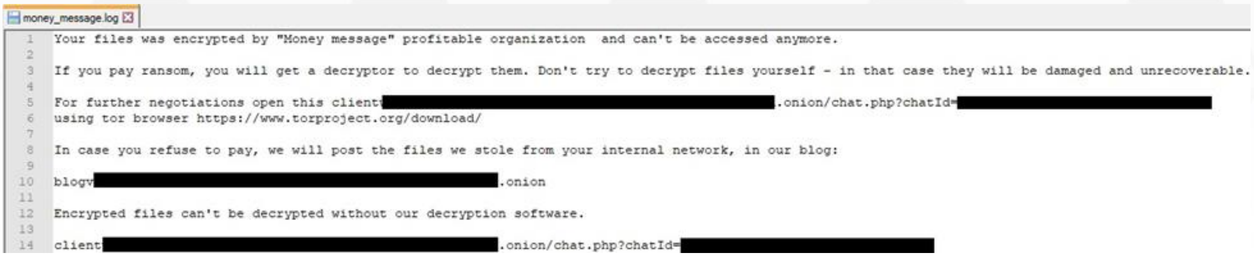


Figure 42

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

money_message.log


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.

For further negotiations open this client[redacted].onion/chat.php?chatId=[redacted]
using tor browser https://www.torproject.org/download/

In case you refuse to pay, we will post the files we stole from your internal network, in our blog:

blogv[redacted].onion

Encrypted files can't be decrypted without our decryption software.

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_F1D1C0 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 “71828bcd92dd7f5950841835ed0fc926a7f6888be7af4fc07ddc06dd8a1e6400bf3ad4cc27083aade393c5262570bcbf47aa9855fb9ecc0c82be053c8d95f974a1f2e32d8005e7059461b1e958fecd7dbc1fa36bf9f9bc8746d431902fe990772d16bab1e779dbe626544401001170 8d1091df3838c0a15b0ccf99db70e951a74670dd05eb719678d62d150edee22706”.

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**):

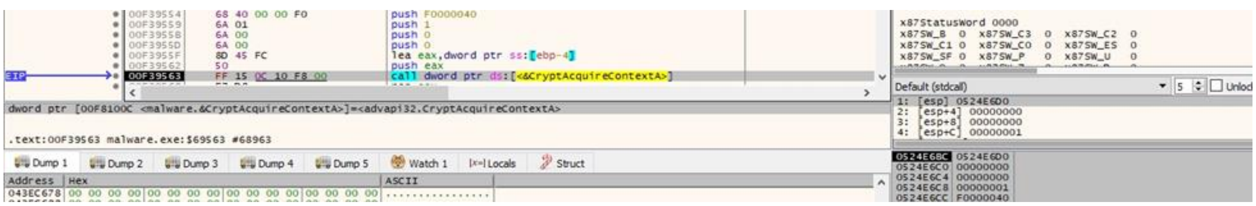


Figure 50

[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

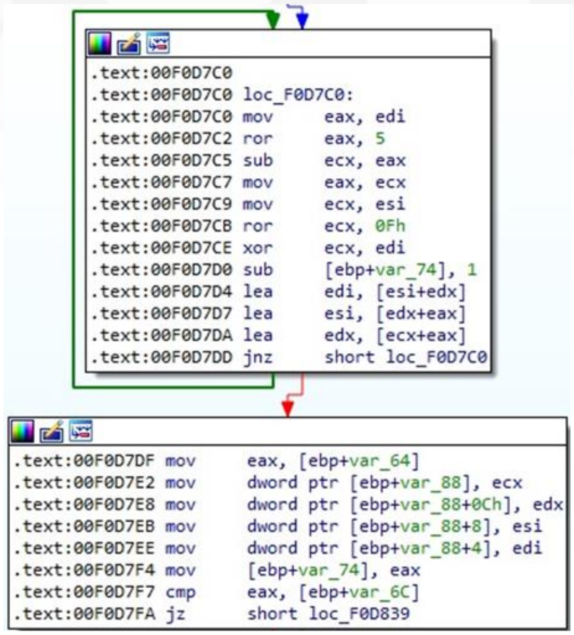


Figure 52

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

Figure 53

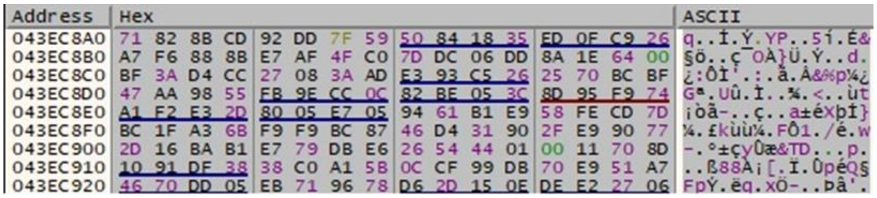


Figure 54

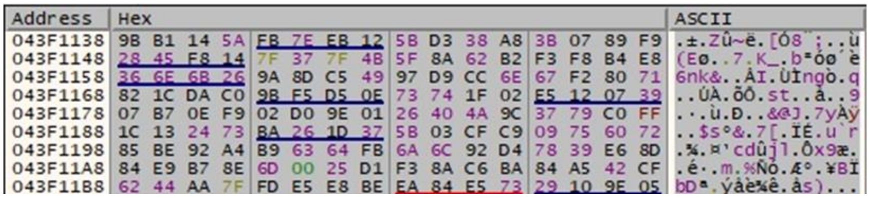


Figure 55

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], 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 ; 'ˆ'
.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], 8000808Bh
.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], 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], 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
```

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**):

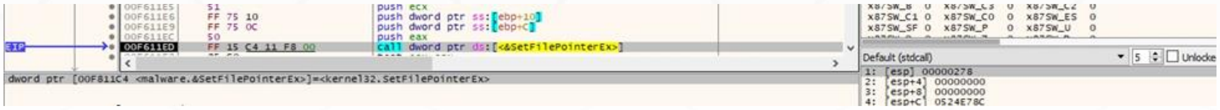


Figure 59

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

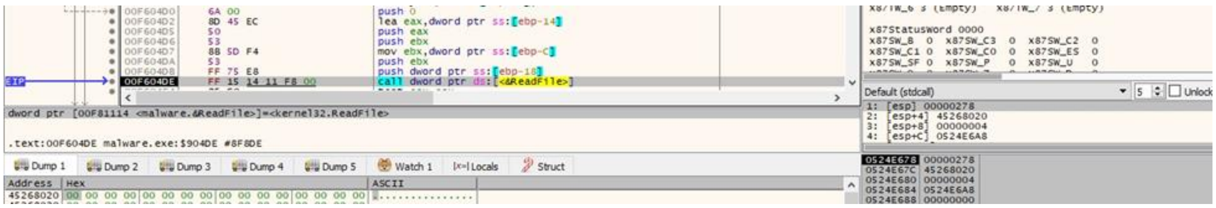


Figure 60

The content is encrypted using the ChaCha20 algorithm:

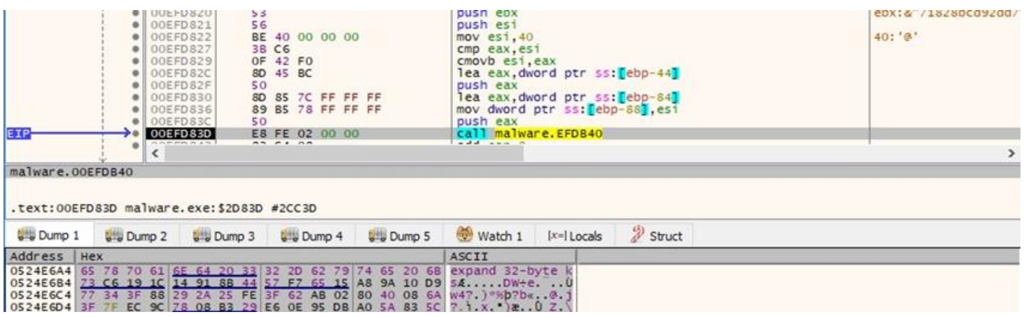


Figure 61

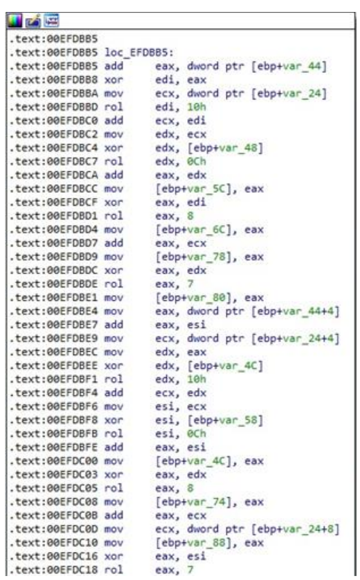


Figure 62

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**):

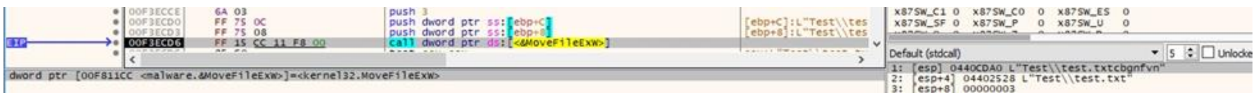


Figure 64

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

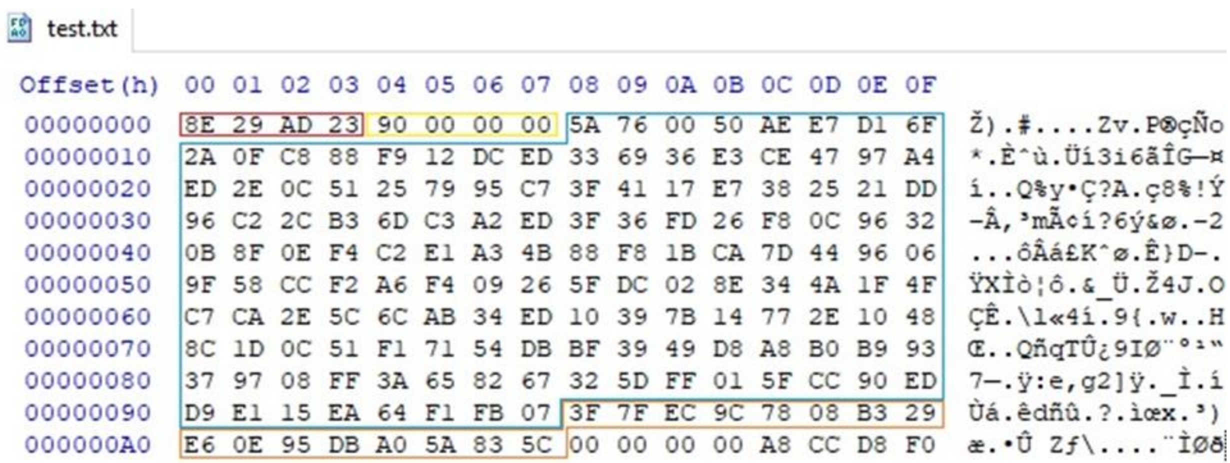


Figure 65

Running with the --crypt parameter

The malware only encrypts the directory passed as the argument.

Running with the -d parameter

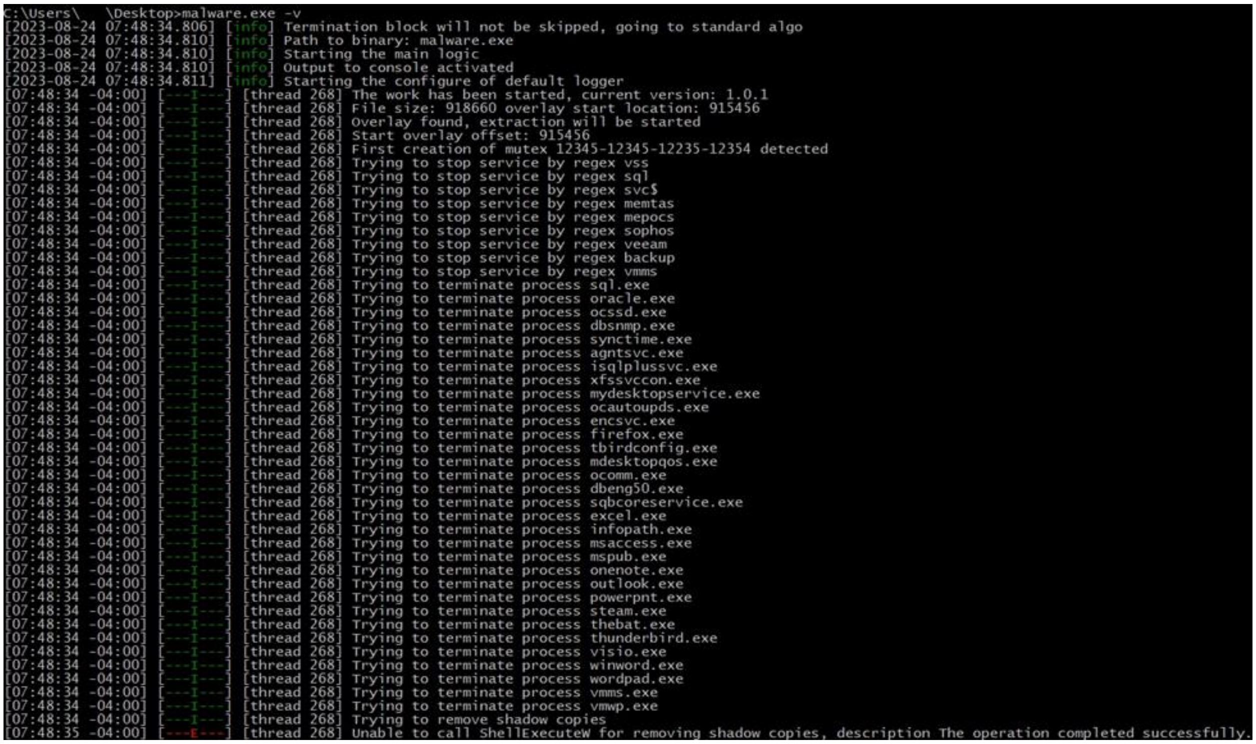
In this case, the ransomware doesn’t stop the target services and processes and doesn’t delete the Volume Shadow Copies.

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 07:48:34.810] [info] Output to console activated
[2023-08-24 07:48:34.811] [info] Starting the configure of default logger
[07:48:34 -04:00] [---I---] [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
[07:48:34 -04:00] [---I---] [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] [---I---] [thread 268] Trying to stop service by regex sql
[07:48:34 -04:00] [---I---] [thread 268] Trying to stop service by regex svc$
[07:48:34 -04:00] [---I---] [thread 268] Trying to stop service by regex memtas
[07:48:34 -04:00] [---I---] [thread 268] Trying to stop service by regex mepocs
[07:48:34 -04:00] [---I---] [thread 268] Trying to stop service by regex sophos
[07:48:34 -04:00] [---I---] [thread 268] Trying to stop service by regex veeam
[07:48:34 -04:00] [---I---] [thread 268] Trying to stop service by regex backup
[07:48:34 -04:00] [---I---] [thread 268] Trying to stop service by regex vmms
[07:48:34 -04:00] [---I---] [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] [---I---] [thread 268] Trying to terminate process ocssd.exe
[07:48:34 -04:00] [---I---] [thread 268] Trying to terminate process dbsnmp.exe
[07:48:34 -04:00] [---I---] [thread 268] Trying to terminate process synctime.exe
[07:48:34 -04:00] [---I---] [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] [---I---] [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---] [thread 268] Trying to terminate process ocautoupds.exe
[07:48:34 -04:00] [---I---] [thread 268] Trying to terminate process encsvc.exe
[07:48:34 -04:00] [---I---] [thread 268] Trying to terminate process firefox.exe
[07:48:34 -04:00] [---I---] [thread 268] Trying to terminate process tbirdconfig.exe
[07:48:34 -04:00] [---I---] [thread 268] Trying to terminate process mdesktopqos.exe
[07:48:34 -04:00] [---I---] [thread 268] Trying to terminate process ocomm.exe
[07:48:34 -04:00] [---I---] [thread 268] Trying to terminate process dbeng50.exe
[07:48:34 -04:00] [---I---] [thread 268] Trying to terminate process sqbcoreservice.exe
[07:48:34 -04:00] [---I---] [thread 268] Trying to terminate process excel.exe
[07:48:34 -04:00] [---I---] [thread 268] Trying to terminate process infopath.exe
[07:48:34 -04:00] [---I---] [thread 268] Trying to terminate process msaccess.exe
[07:48:34 -04:00] [---I---] [thread 268] Trying to terminate process mspub.exe
[07:48:34 -04:00] [---I---] [thread 268] Trying to terminate process onenote.exe
[07:48:34 -04:00] [---I---] [thread 268] Trying to terminate process outlook.exe
[07:48:34 -04:00] [---I---] [thread 268] Trying to terminate process powerpnt.exe
[07:48:34 -04:00] [---I---] [thread 268] Trying to terminate process steam.exe
[07:48:34 -04:00] [---I---] [thread 268] Trying to terminate process thebat.exe
[07:48:34 -04:00] [---I---] [thread 268] Trying to terminate process thunderbird.exe
[07:48:34 -04:00] [---I---] [thread 268] Trying to terminate process visio.exe
[07:48:34 -04:00] [---I---] [thread 268] Trying to terminate process winword.exe
[07:48:34 -04:00] [---I---] [thread 268] Trying to terminate process wordpad.exe
[07:48:34 -04:00] [---I---] [thread 268] Trying to terminate process vmms.exe
[07:48:34 -04:00] [---I---] [thread 268] Trying to terminate process vmwp.exe
[07:48:34 -04:00] [---I---] [thread 268] Trying to remove shadow copies
[07:48:35 -04:00] [---E---] [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