A technical analysis of the BackMyData ransomware used to attack hospitals in Romania

cybergeeks.tech/a-technical-analysis-of-the-backmydata-ransomware-used-to-attack-hospitals-in-romania/

Summary

According to BleepingComputer, a ransomware attack that occurred starting 0n February 11 forced 100 hospitals across Romania to take their systems offline. BackMyData ransomware, which took credit for it, belongs to the Phobos family. The malware embedded an AES key that is used to decrypt its configuration containing whitelisted extensions, files, and directories, a public RSA key that is used to encrypt AES keys used for files' encryption, and other information. Persistence is achieved by creating an entry under the Run registry key and copying the malware to the Startup folder. The ransomware encrypts the local drives as well as the network shares. It deletes all Volume Shadow Copies and runs commands to disable the firewall.

The files are encrypted using the AES256 algorithm, with the AES key being encrypted using the public RSA key decrypted from the configuration. The malware appends 6 custom bytes at the end of every encrypted file. In the end, the ransomware drops two ransom notes called "info.txt" and "info.hta" that contain information about how to contact the threat actor.

Technical analysis

SHA256: 396a2f2dd09c936e93d250e8467ac7a9c0a923ea7f9a395e63c375b877a399a6

The ransomware comes with an encrypted configuration that is decrypted using a hard-coded AES key:

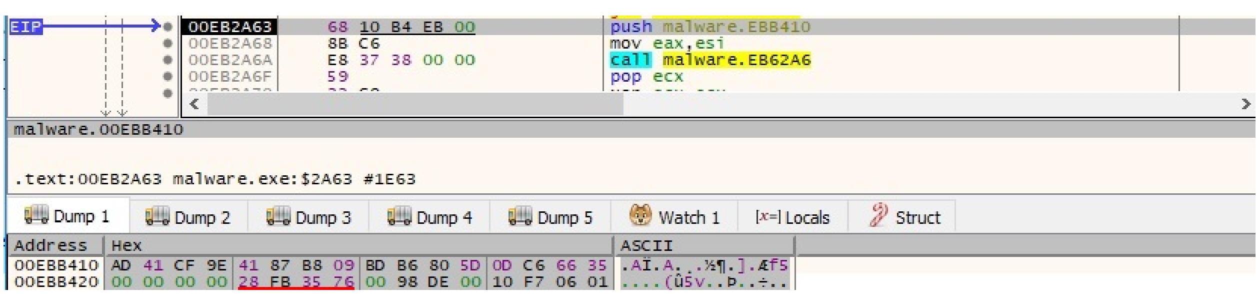


Figure 1

As we can see below, the configuration is stored in an encrypted form:

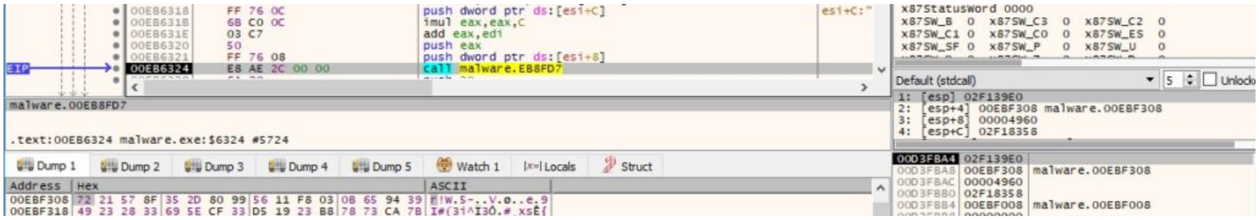


Figure 2

The malware implements the AES algorithm in its code and doesn't rely on Windows APIs:

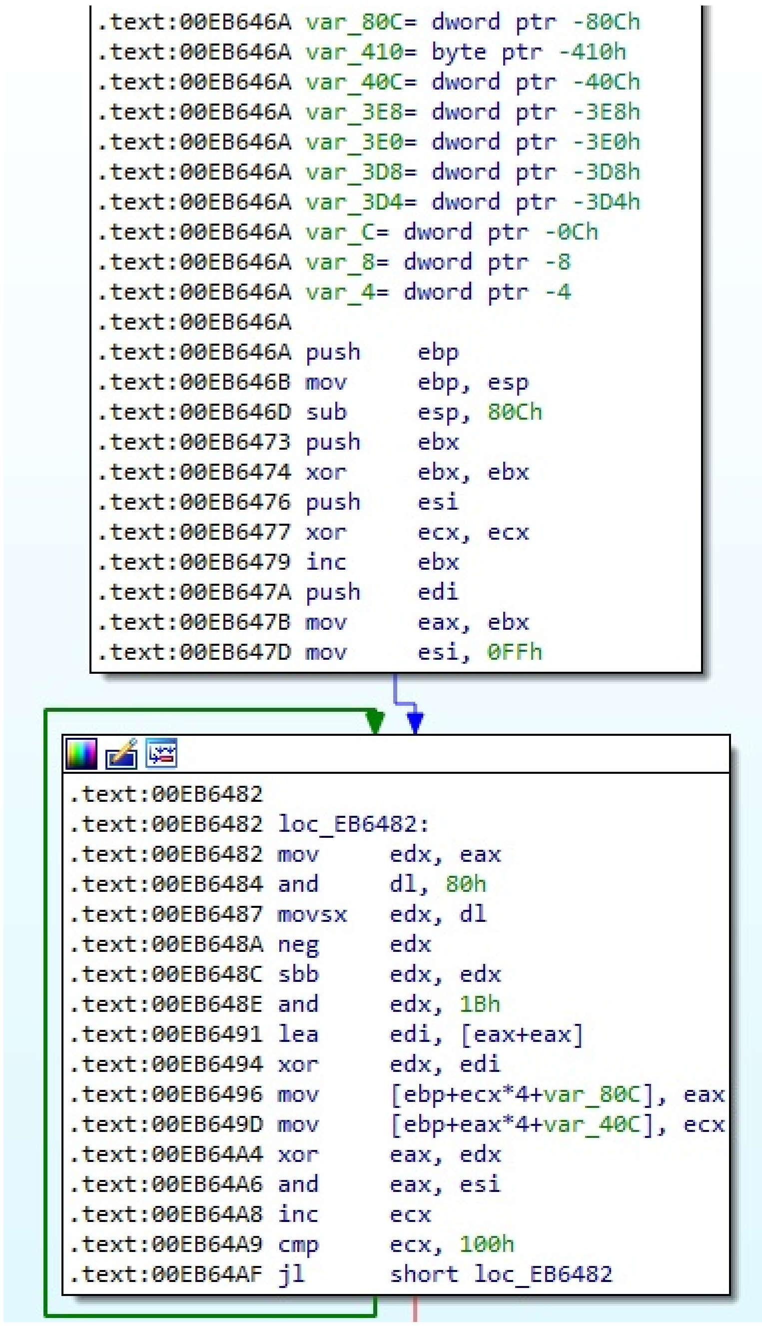


Figure 3

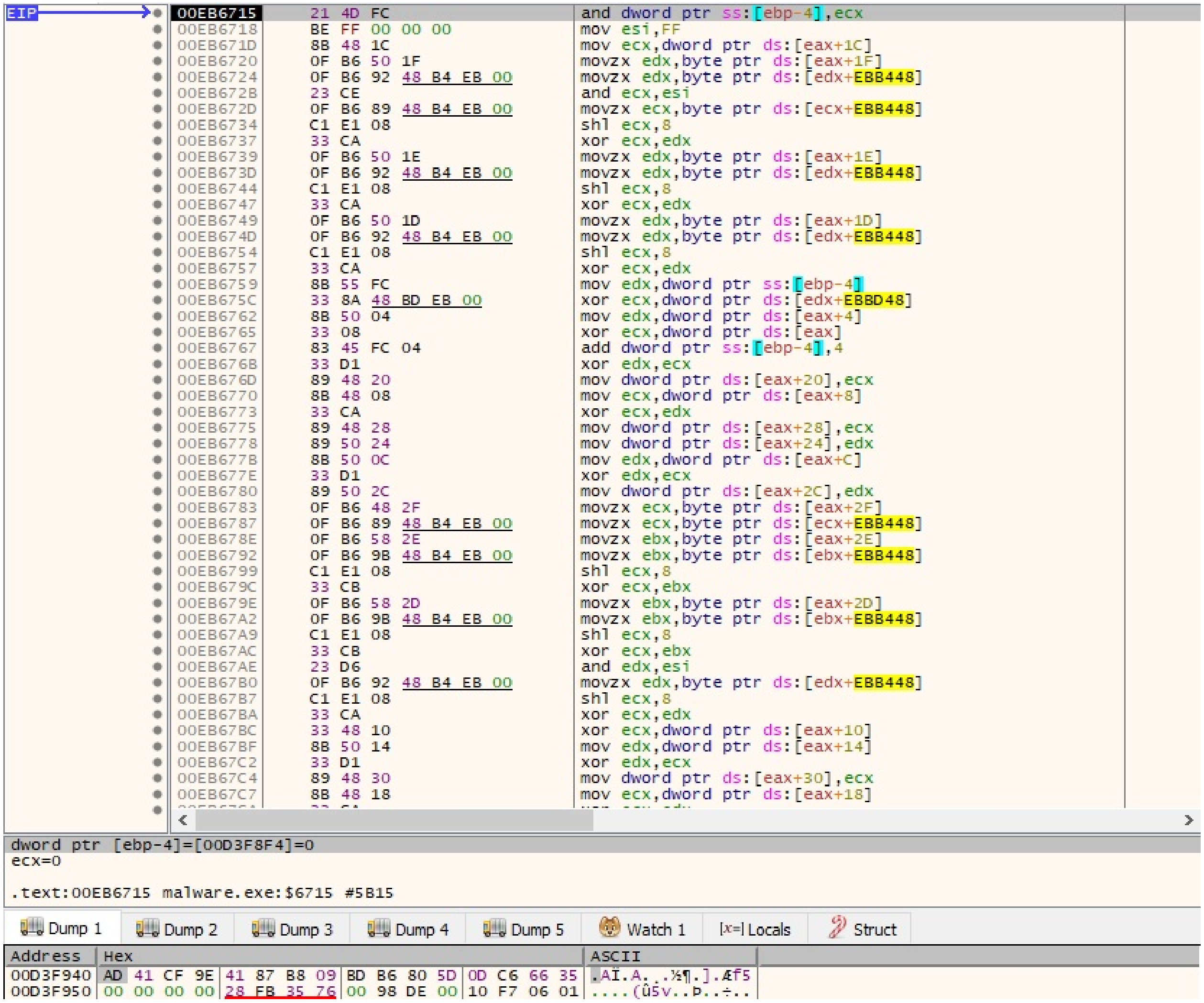


Figure 4

The malicious process retrieves the number of milliseconds elapsed since the system was started using GetTickCount:

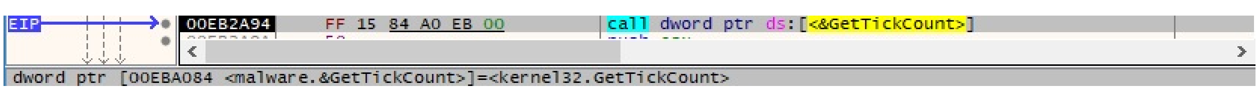


Figure 5

The GetLocaleInfoW function is used to obtain the default locale for the operating system (0x800 = **LOCALE_SYSTEM_DEFAULT**, 0x58 = **LOCALE_FONTSIGNATURE**). The binary verifies whether the 9th bit, which represents Cyrillic alphabets, is cleared. This technique of avoiding systems that have this setting as default was also documented by Malwarebytes in their article about Phobos ransomware.

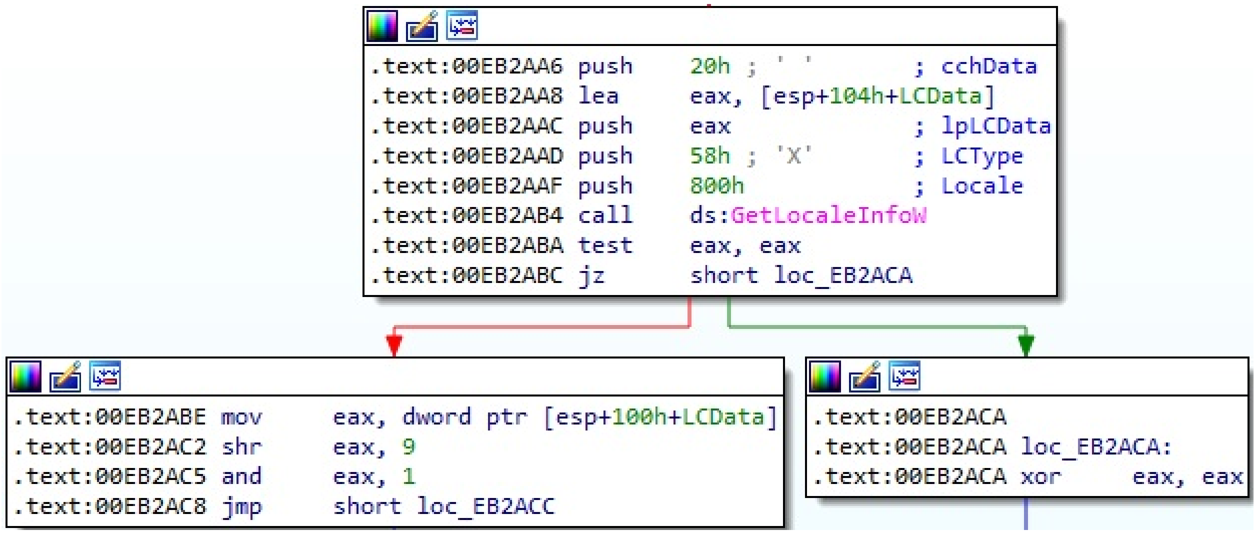


Figure 6

An example of decrypting values from configuration is highlighted below:

Figure 7

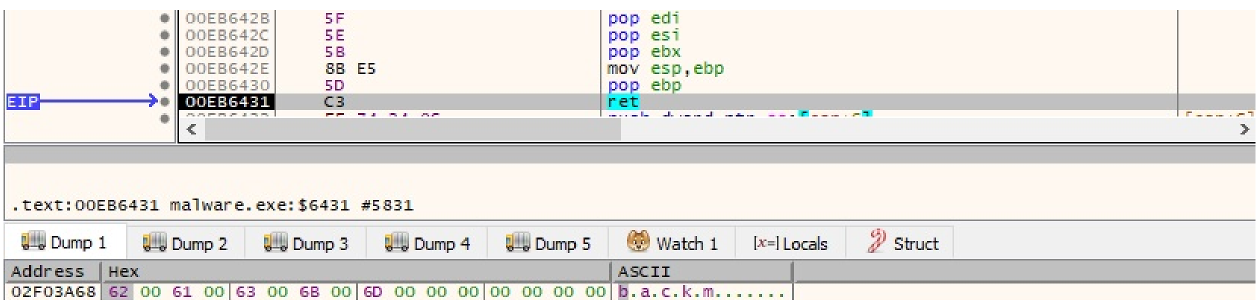


Figure 8

The binary retrieves the path of the executable file of the current process via a function call to GetModuleFileNameW (see Figure 9).

Figure 9

Interestingly, the process is looking for a file called “backm” that wasn’t previously created by the ransomware (0x80000000 = **GENERIC_READ**, 0x3 = **OPEN_EXISTING**):

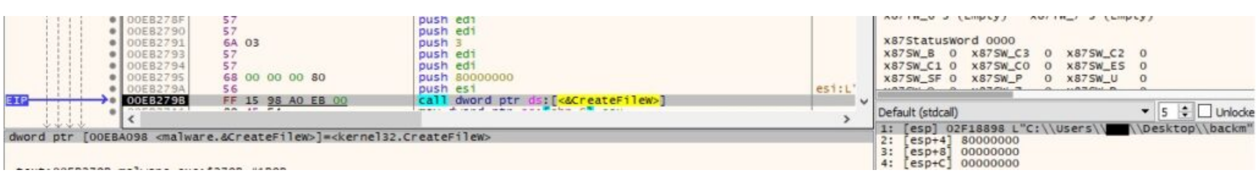


Figure 10

The malware extracts the major and minor version numbers of the operating system using the GetVersion method:

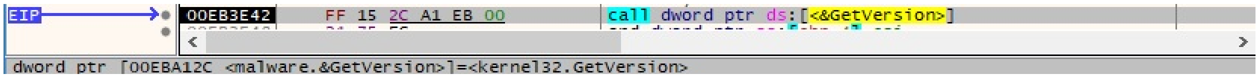


Figure 11

It opens the access token associated with the current process by calling the OpenProcessToken API (0x8 = **TOKEN_QUERY**):

Figure 12

The malicious process verifies if the token is elevated using GetTokenInformation (0x14 = **TokenElevation**):

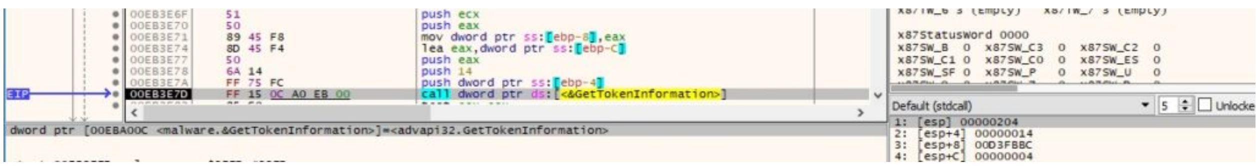


Figure 13

The environment variable “%systemdrive%” is expanded, which reveals the drive that contains the Windows directory:

Figure 14

GetVolumeInformationW is used to obtain the volume serial number:

Figure 15

The ransomware tries to open two mutexes called “Global\\<<BID>><Volume serial number>00000001” and “Global\\<<BID>><Volume serial number>00000000”, and then creates them:

Figure 16

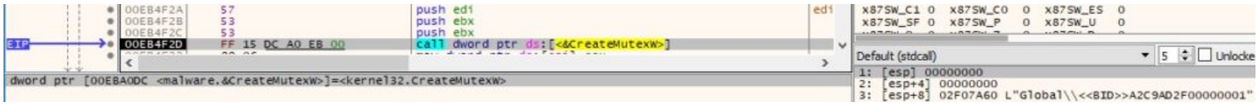


Figure 17

The DLLs and functions necessary to perform some activities are also decrypted from the configuration. The binary obtains the module handle for a DLL using GetModuleHandleA:

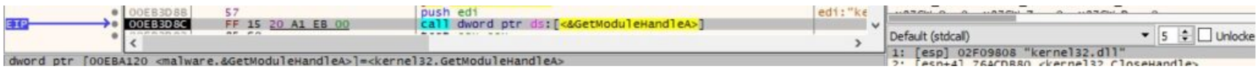


Figure 18

The address of the exported functions is retrieved by calling the GetProcAddress API:

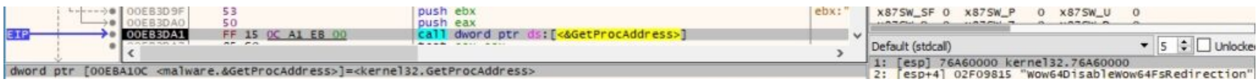


Figure 19

The malware disables file system redirection for the calling thread:

Figure 20

It obtains a handle to the Shell’s desktop window via a function call to GetShellWindow (Figure 21).

Figure 21

Using the above handle, the process calls the GetWindowThreadProcessId function to retrieve the identifier of the process that created the window (explorer.exe):

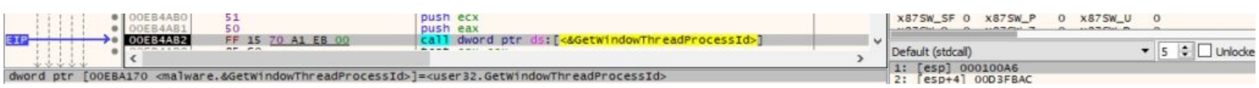


Figure 22

The binary opens the “explorer.exe” process using the OpenProcess method (0x400 = **PROCESS_QUERY_INFORMATION**):

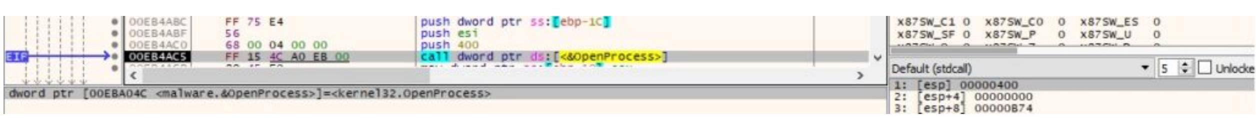


Figure 23

OpenProcessToken is used to open the access token associated with the above process (0x02000000 = **MAXIMUM_ALLOWED**):

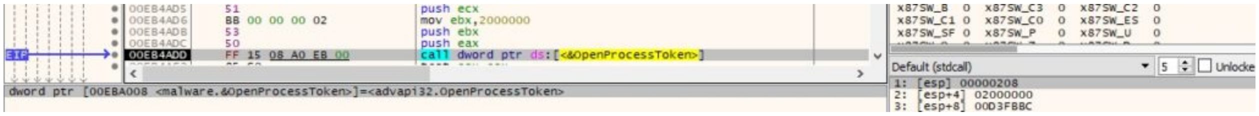


Figure 24

The DuplicateTokenEx API is utilized to create a new access token that duplicates the token mentioned above (0x2 = **SecurityImpersonation**, 0x1 = **TokenPrimary**):

Figure 25

The ransomware spawns itself running in the security context of the newly created token (Figure 26).

Figure 26

It creates a new thread that will run the following commands in the sub_EB4B85 function:

- vssadmin delete shadows /all /quiet – delete all Volume Shadow Copies
- wmic shadowcopy delete – delete all Volume Shadow Copies
- bcdedit /set {default} bootstatuspolicy ignoreallfailures – ignore errors if there is a failed boot, shutdown, or checkpoint
- bcdedit /set {default} recoveryenabled no – disable automatic repair
- wbadmin delete catalog -quiet – delete the backup catalog on the machine
- netsh advfirewall set currentprofile state off – disable the firewall for the current network profile
- netsh firewall set opmode mode=disable – disable the firewall

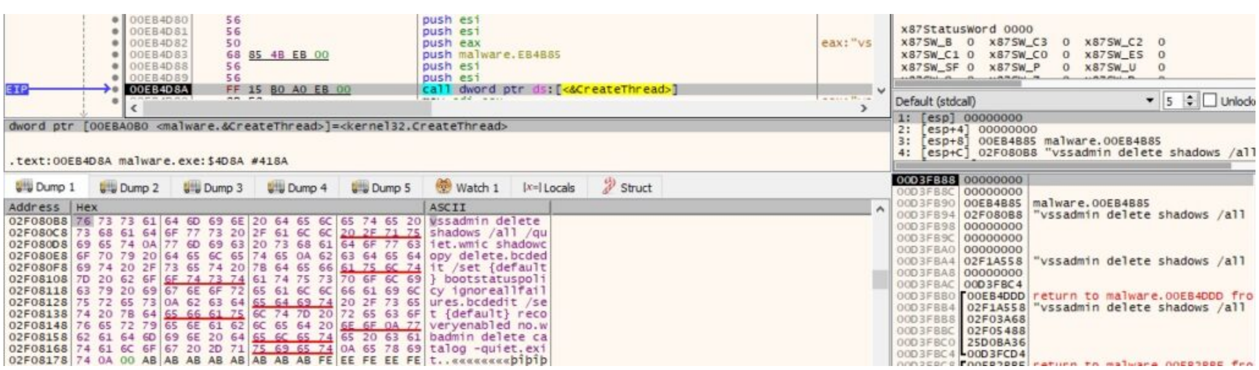


Figure 27

The process copies its executable to the “%AppData%\Local” directory, as highlighted in Figure 28.

Figure 28

RegOpenKeyExW is used to open the Run registry key (0x80000002 = **HKEY_LOCAL_MACHINE**, 0x20106 = **KEY_WRITE** | **KEY_WOW64_64KEY**):

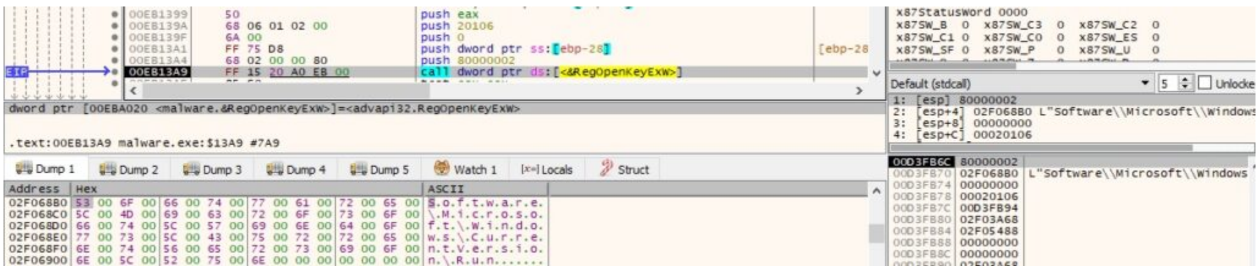


Figure 29

The ransomware establishes persistence by creating an entry named based on the executable name, which points to the newly created executable:

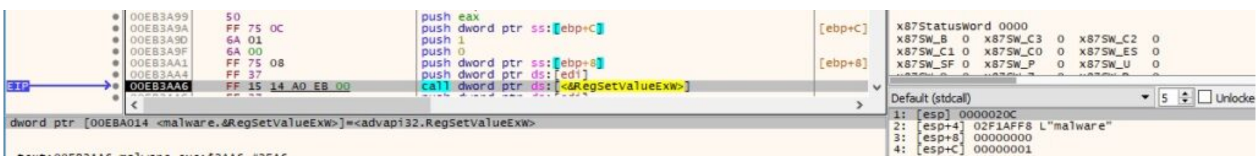


Figure 30

The malicious binary tries to copy the non-existent file called “backm” to the same directory:

Figure 31

The second persistence mechanism consists of copying the executable to the Startup folder.

The following extensions are targeted, but the ransomware will encrypt other extensions as well:

> fdb sql 4dd 4dl abs abx accdb accdc accde adb adf ckp db db-journal db-shm db-wal db2 db3 dbc dbf dbs dbt dbv dcb dp1 eco edb epim fcd gdb mdb mdf ldf myd ndf nwdb nyf sqlitedb sqlite3 sqlite

Also, the ransomware doesn’t encrypt files that were previously encrypted by other ransomware families:

backmydata actin DIKE Acton actor Acuff FILE Acuna fullz MMXXII GrafGrafel monero n3on jopanaxye 2700 DEVOS kmrox s0m1n qos cg ext rdptest S0va 6y8dghklp SHTORM NURRI GHOST FF6OM6 blue NX BACKJOHN OWN FS23 2QZ3 top blackrock CHCRBO G-STARS faust unknown STEEL worry WIN duck fopra unique acute adage make Adair MLF magic Adame banhu banjo Banks Banta Barak Caleb Cales Caley calix Calle Calum Calvo deuce Dever devil Devoe Devon Devos dewar eight eject eking Elbie elbow elder phobos help blend bqux com mamba KARLOS DDoS phoenix PLUT karma bbc CAPITAL WALLET LKS tech s1g2n3a4l MURK makop ebaka jook LOGAN FIASKO GUCCI decrypt OOH Non grt LIZARD FLSCRYPT SDK 2023 vhdv

The following files and directories will also be skipped during the encryption process:

- info.hta info.txt boot.ini bootfont.bin ntldr ntdetect.com io.sys backm
- C:\WINDOWS C:\ProgramData\microsoft\windows\caches

The process splits further malicious activity into multiple threads that will be described in the following paragraphs. The following functions will be executed: sub_EB22EE, sub_EB239A, sub_EB2161, sub_EB1A76, and sub_EB1CC5.

Figure 32

Thread activity – sub_EB22EE function

The malware opens the access token associated with the current process (0x20 = **TOKEN_ADJUST_PRIVILEGES**):

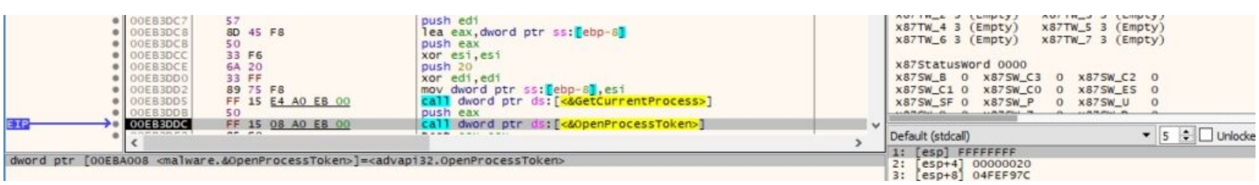


Figure 33

The LookupPrivilegeValueW method is used to extract the locally unique identifier (LUID) that represents the “SeDebugPrivilege” privilege:

Figure 34

The malicious process enables the above privilege via a call to AdjustTokenPrivileges:

Figure 35

The following processes will be killed because they could lock files to be encrypted:

> msftesql.exe sqlagent.exe sqlbrowser.exe sqlservr.exe sqlwriter.exe oracle.exe ocssd.exe dbsnmp.exe synctime.exe agntsvc.exe mydesktopqos.exe isqlplussvc.exe xfssvccon.exe mydesktopservice.exe ocautoupds.exe agntsvc.exe agntsvc.exe agntsvc.exe encsvc.exe firefoxconfig.exe tbirdconfig.exe ocomm.exe mysqld.exe mysqld-nt.exe mysqld-opt.exe dbeng50.exe sqbcoreservice.exe excel.exe infopath.exe msaccess.exe mspub.exe onenote.exe outlook.exe powerpnt.exe steam.exe thebat.exe thebat64.exe thunderbird.exe visio.exe winword.exe wordpad.exe

The malware takes a snapshot of all processes in the system, as displayed in the figure below.

Figure 36

The processes are enumerated using the Process32FirstW and Process32NextW APIs:

Figure 37

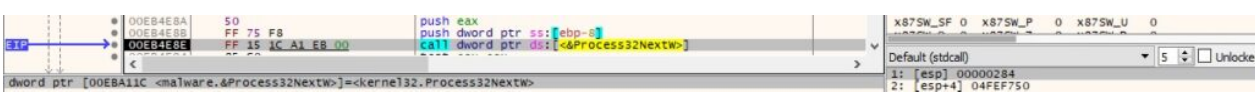


Figure 38

Any target process is stopped using the TerminateProcess method:

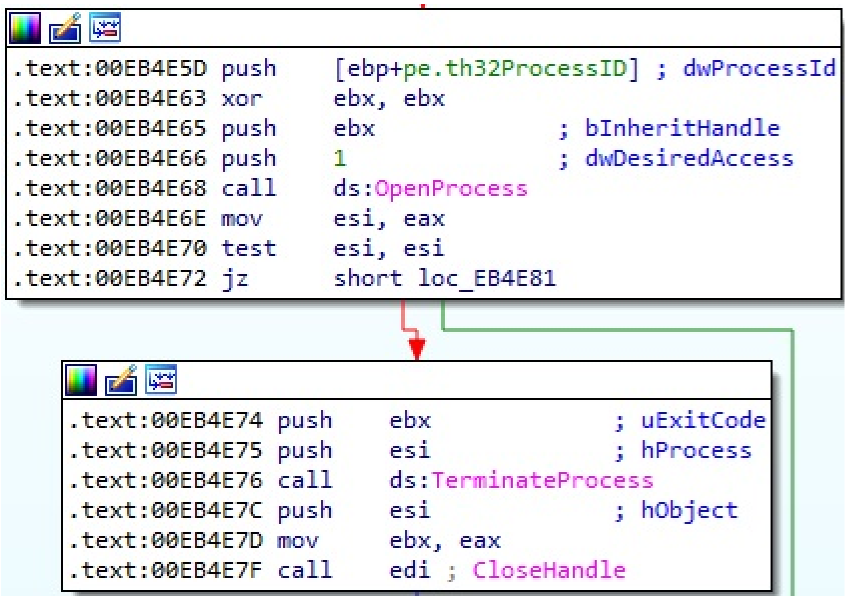


Figure 39

Thread activity – sub_EB239A function

OpenProcessToken is utilized to open the access token associated with the process (0x8 = **TOKEN_QUERY**):

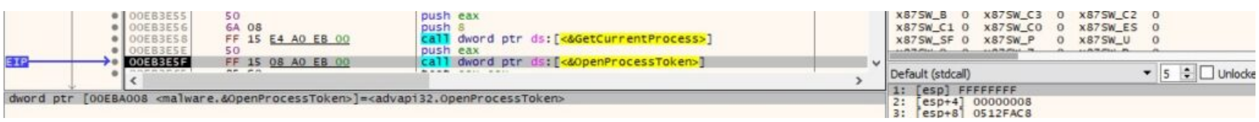


Figure 40

The binary verifies again if the token is elevated by calling the GetTokenInformation API (0x14 = **TokenElevation**):

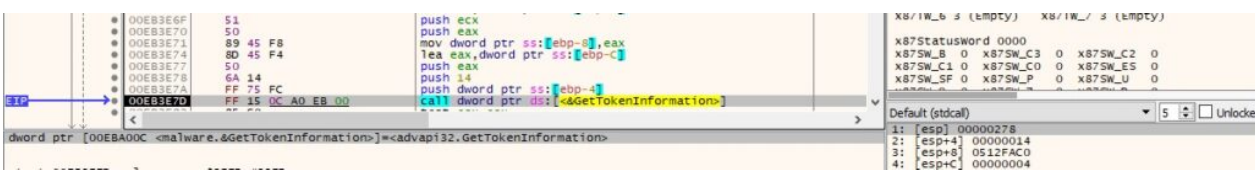


Figure 41

It calls again the OpenMutexW and CreateMutexW methods with the “Global\\<<BID>><Volume serial number>00000000” mutex name:

Figure 42

Thread activity – sub_EB2161 function

The ransomware uses events to synchronize threads. It creates two unnamed event objects using CreateEventW:

Figure 43

The NetBIOS name of the local machine is extracted (Figure 44).

Figure 44

WNetOpenEnumW is used to start an enumeration of all currently connected resources (0x1 = **RESOURCE_CONNECTED**):

Figure 45

The enumeration continues by calling the WNetEnumResourceW function:

Figure 46

The process obtains the interface–to–IPv4 address mapping table via a function call to GetIpAddrTable, as shown below:

Figure 47

Every IP address extracted above is converted from network order to host byte order using ntohl:

Figure 48

The malware creates a TCP socket (0x2 = **AF_INET**, 0x1 = **SOCK_STREAM**, 0x6 = **IPPROTO_TCP**):

Figure 49

It tries to connect to every host on the network on port 445 in order to encrypt every available network share:

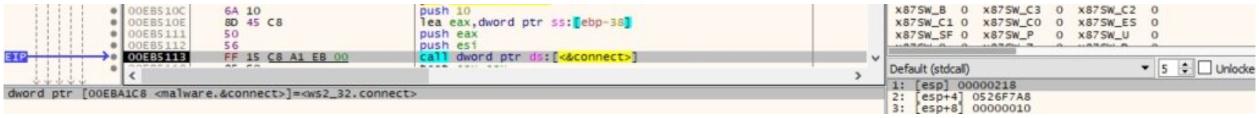


Figure 50

Thread activity – sub_EB4B85 function

The process creates two anonymous pipes by calling the CreatePipe method (see Figure 51).

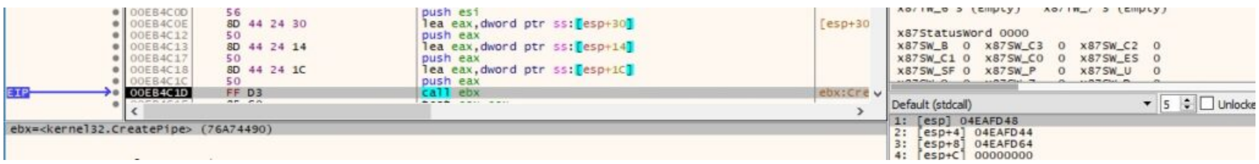


Figure 51

The read handles are made inheritable using SetHandleInformation (0x1 = **HANDLE_FLAG_INHERIT**):

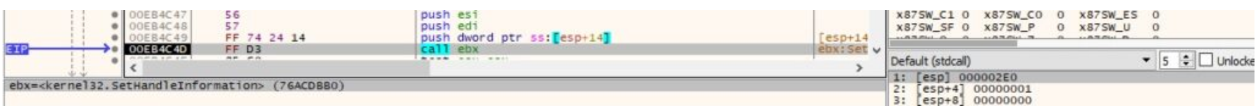


Figure 52

The ransomware creates a “cmd.exe” process that will execute multiple commands:

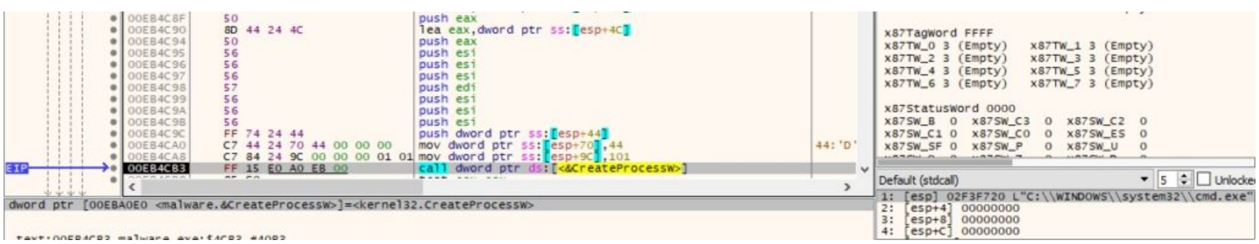


Figure 53

The commands responsible for disabling the firewall, deleting all Volume Shadow Copies, and so on, are transmitted to the newly created process via pipes:

Figure 54

Thread activity – sub_EB1CC5 function

This thread keeps extracting a bitmask representing the currently available disk drives using the GetLogicalDrives API:

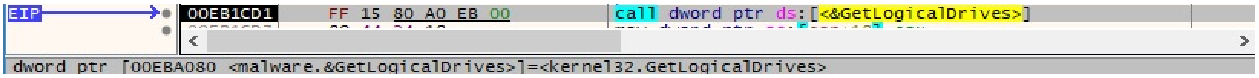


Figure 55

Thread activity – sub_EB1A76 function

The malware decrypts the public RSA key that will be used to encrypt the AES256 key used for file's encryption. The same key was used by Phobos ransomware since 2019 according to Talos.

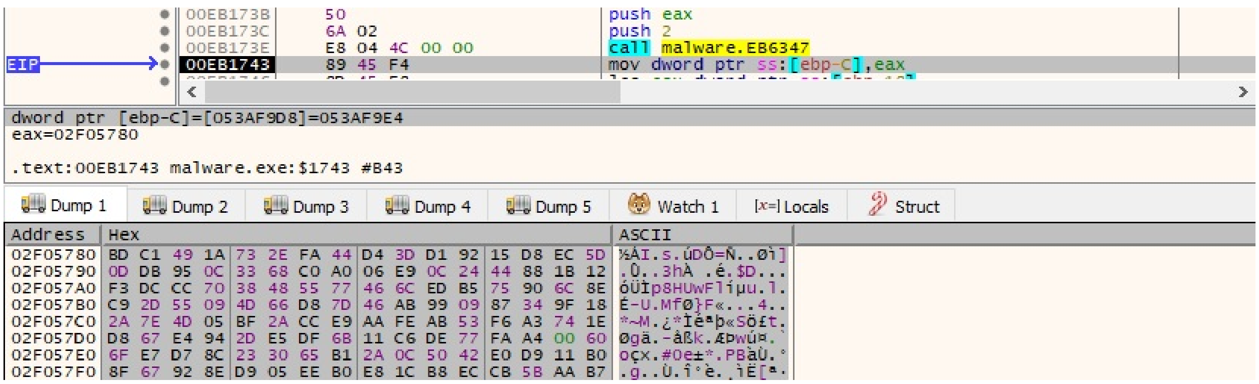


Figure 56

It extracts the current local date and time, the current process and thread IDs, and other information using multiple functions:

```
.text:00EB77FF mov     edi, ds:QueryPerformanceCounter
.text:00EB7805 lea     eax, [ebp+PerformanceCount]
.text:00EB7808 push    eax             ; lpPerformanceCount
.text:00EB7809 call    edi ; QueryPerformanceCounter
.text:00EB780B call    ds:GetTickCount
.text:00EB7811 xor     dword_EBD680, eax
.text:00EB7817 mov     eax, dword ptr [ebp+PerformanceCount+4]
.text:00EB781A xor     dword_EBD684, eax
.text:00EB7820 mov     eax, dword ptr [ebp+PerformanceCount]
.text:00EB7823 xor     dword_EBD688, eax
.text:00EB7829 call    ds:GetCurrentProcessId
.text:00EB782F xor     dword_EBD68C, eax
.text:00EB7835 call    ds:GetCurrentThreadId
.text:00EB783B xor     dword_EBD690, eax
.text:00EB7841 lea     eax, [ebp+SystemTime]
.text:00EB7844 push    eax             ; lpSystemTime
.text:00EB7845 call    ds:GetLocalTime
.text:00EB784B lea     eax, [ebp+FileTime]
.text:00EB784E push    eax             ; lpFileTime
.text:00EB784F lea     eax, [ebp+SystemTime]
.text:00EB7852 push    eax             ; lpSystemTime
.text:00EB7853 call    ds:SystemTimeToFileTime
.text:00EB7859 mov     eax, [ebp+FileTime.dwHighDateTime]
.text:00EB785C xor     dword_EBD694, eax
.text:00EB7862 mov     eax, [ebp+FileTime.dwLowDateTime]
.text:00EB7865 xor     dword_EBD698, eax
.text:00EB786B lea     eax, [ebp+PerformanceCount]
.text:00EB786E push    eax             ; lpPerformanceCount
.text:00EB786F call    edi ; QueryPerformanceCounter
.text:00EB7871 mov     eax, dword ptr [ebp+PerformanceCount+4]
.text:00EB7874 xor     dword_EBD69C, eax
.text:00EB787A mov     eax, dword ptr [ebp+PerformanceCount]
.text:00EB787D xor     dword_EBD680, eax
```

Figure 57

The binary creates a new thread that will traverse the network shares and drives in order to extract files to be encrypted:

Figure 58

Thread activity – sub_EB56B3 function

Two new threads, which will be responsible for file’s encryption, are created:

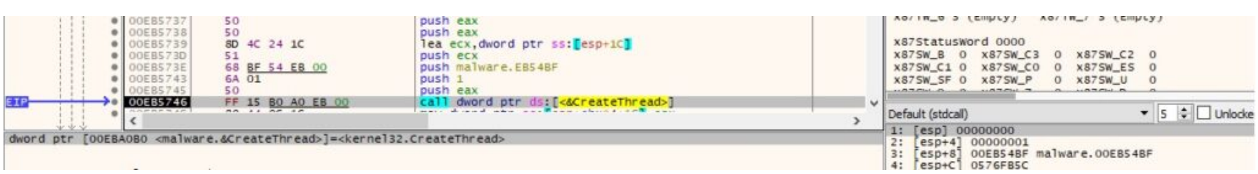


Figure 59

The files are enumerated using the FindFirstFileW and FindNextFileW methods:

Figure 60

Figure 61

The malware sets the event objects to the signaled state via a function call to SetEvent:

```
.text:00EB33C5
.text:00EB33C5
.text:00EB33C5
.text:00EB33C5 sub_EB33C5 proc near
.text:00EB33C5 push    ebx
.text:00EB33C6 push    edi
.text:00EB33C7 lea     ebx, [esi+0Ch]
.text:00EB33CA push    ebx             ; lpCriticalSection
.text:00EB33CB call    ds:EnterCriticalSection
.text:00EB33D1 push    dword ptr [esi+4] ; hEvent
.text:00EB33D4 mov     edi, ds:SetEvent
.text:00EB33DA or      dword ptr [esi+28h], 1
.text:00EB33DE call    edi ; SetEvent
.text:00EB33E0 push    dword ptr [esi] ; hEvent
.text:00EB33E2 call    edi ; SetEvent
.text:00EB33E4 push    dword ptr [esi+8] ; hEvent
.text:00EB33E7 call    edi ; SetEvent
```

Figure 62

The process waits until the new threads finish their execution using the WaitForMultipleObjects function.

Thread activity – sub_EB54BF function

The ransomware opens a file to be encrypted in reading mode (0x80000000 = **GENERIC_READ**, 0x7 = **FILE_SHARE_DELETE** | **FILE_SHARE_WRITE** | **FILE_SHARE_READ**, 0x3 = **OPEN_EXISTING**):

Figure 63

It retrieves the size of the file using the GetFileSizeEx API:

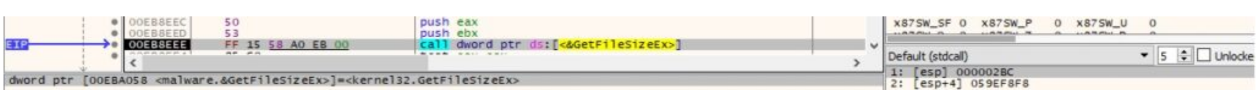


Figure 64

The size is compared with 0x180000 bytes (1.5MB), and the files having more bytes are partially encrypted. The rest of the files are totally encrypted:

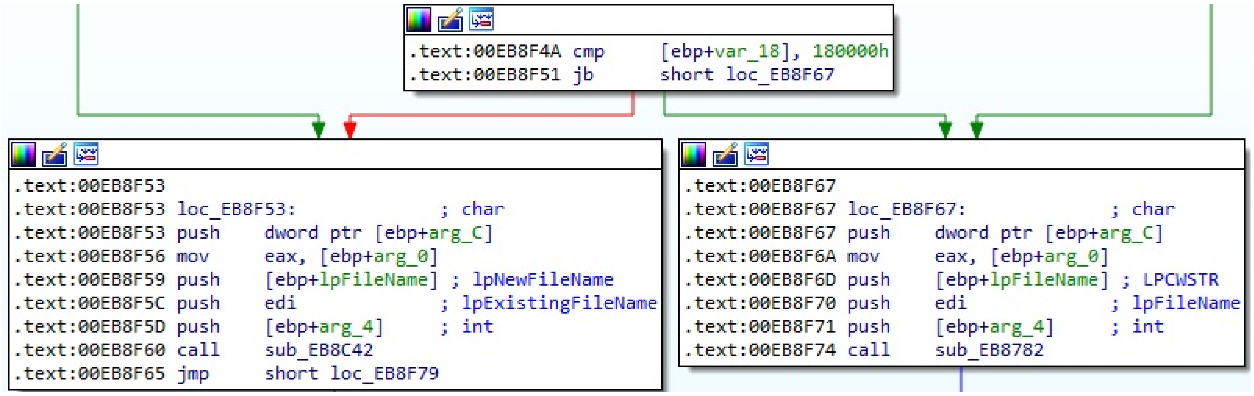


Figure 65

As we can see below, not only the ".backmydata" extension will be added to an encrypted file, but also the volume serial number and the threat actor's email address:

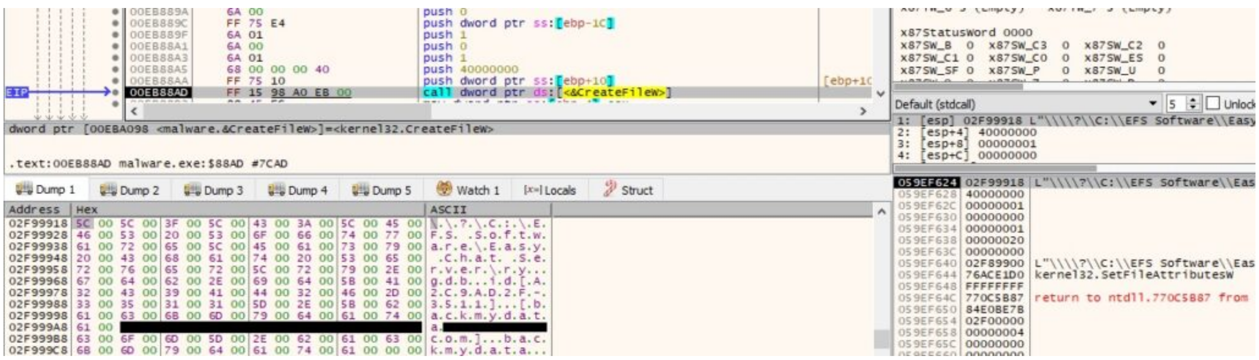


Figure 66

The file's content is read using the ReadFile method (see Figure 67).

Figure 67

There is a custom implementation of the AES256 algorithm, as highlighted in the figure below.

```
.text:00EB6ABB
.text:00EB6ABB loc_EB6ABB:
.text:00EB6ABB shr     edi, 8
.text:00EB6ABE and     edi, eax
.text:00EB6AC0 shr     edx, 10h
.text:00EB6AC3 shr     ebx, 8
.text:00EB6AC6 and     ebx, eax
.text:00EB6AC8 and     edx, eax
.text:00EB6ACA mov     edx, dword_EBC970[edx*4]
.text:00EB6AD1 xor     edx, dword_EBCD70[edi*4]
.text:00EB6AD8 mov     edi, [ebp+var_C]
.text:00EB6ADB shr     edi, 18h
.text:00EB6ADE xor     edx, dword_EBC570[edi*4]
.text:00EB6AE5 mov     edi, [ebp+var_4]
.text:00EB6AE8 and     edi, eax
.text:00EB6AEA xor     edx, dword_EBD170[edi*4]
.text:00EB6AF1 mov     edi, [ebp+var_C]
.text:00EB6AF4 shr     edi, 10h
.text:00EB6AF7 and     edi, eax
.text:00EB6AF9 mov     edi, dword_EBC970[edi*4]
.text:00EB6B00 xor     edi, dword_EBCD70[ebx*4]
.text:00EB6B07 mov     ebx, [ebp+var_4]
.text:00EB6B0A shr     ebx, 18h
.text:00EB6B0D xor     edi, dword_EBC570[ebx*4]
.text:00EB6B14 mov     ebx, [ebp+var_8]
.text:00EB6B17 and     ebx, eax
.text:00EB6B19 xor     edi, dword_EBD170[ebx*4]
.text:00EB6B20 mov     ebx, [ebp+var_4]
.text:00EB6B23 xor     edi, [ecx+4]
.text:00EB6B26 shr     ebx, 10h
.text:00EB6B29 mov     [ebp+arg_0], edi
.text:00EB6B2C mov     edi, [ebp+var_C]
.text:00EB6B2F shr     edi, 8
.text:00EB6B32 and     edi, eax
.text:00EB6B34 mov     edi, dword_EBCD70[edi*4]
.text:00EB6B3B and     ebx, eax
.text:00EB6B3D xor     edi, dword_EBC970[ebx*4]
.text:00EB6B44 mov     ebx, [ebp+var_8]
.text:00EB6B47 shr     ebx, 18h
.text:00EB6B4A xor     edi, dword_EBC570[ebx*4]
.text:00EB6B51 mov     ebx, [ebp+var_14]
.text:00EB6B54 and     ebx, eax
```

```
.text:00EB6CA7 shr     edx, 10h
.text:00EB6CAA and     edx, eax
.text:00EB6CAC shr     edi, 8
.text:00EB6CAF and     edi, eax
.text:00EB6CB1 shr     ebx, 8
.text:00EB6CB4 mov     [ebp+arg_0], edi
.text:00EB6CB7 mov     edi, dword_EBC970[edx*4]
.text:00EB6CBE mov     edx, [ebp+arg_0]
.text:00EB6CC1 xor     edi, dword_EBCD70[edx*4]
.text:00EB6CC8 mov     edx, [ebp+var_C]
.text:00EB6CCB shr     edx, 18h
.text:00EB6CCE xor     edi, dword_EBC570[edx*4]
.text:00EB6CD5 mov     edx, [ebp+var_4]
.text:00EB6CD8 and     edx, eax
.text:00EB6CDA xor     edi, dword_EBD170[edx*4]
.text:00EB6CE1 mov     edx, [ebp+var_C]
.text:00EB6CE4 shr     edx, 10h
.text:00EB6CE7 and     edx, eax
.text:00EB6CE9 mov     edx, dword_EBC970[edx*4]
.text:00EB6CF0 and     ebx, eax
.text:00EB6CF2 xor     edx, dword_EBCD70[ebx*4]
.text:00EB6CF9 mov     ebx, [ebp+var_4]
.text:00EB6CFC shr     ebx, 18h
.text:00EB6CFF xor     edx, dword_EBC570[ebx*4]
.text:00EB6D06 mov     ebx, [ebp+var_8]
.text:00EB6D09 and     ebx, eax
.text:00EB6D0B xor     edx, dword_EBD170[ebx*4]
.text:00EB6D12 mov     ebx, [ebp+var_4]
.text:00EB6D15 xor     edx, [ecx+4]
.text:00EB6D18 shr     ebx, 10h
.text:00EB6D1B mov     [ebp+arg_0], edx
.text:00EB6D1E mov     edx, [ebp+var_C]
.text:00EB6D21 shr     edx, 8
.text:00EB6D24 and     edx, eax
.text:00EB6D26 mov     edx, dword_EBCD70[edx*4]
.text:00EB6D2D and     ebx, eax
.text:00EB6D2F xor     edx, dword_EBC970[ebx*4]
.text:00EB6D36 mov     ebx, [ebp+var_8]
.text:00EB6D39 shr     ebx, 18h
.text:00EB6D3C xor     edx, dword_EBC570[ebx*4]
.text:00EB6D43 mov     ebx, [ebp+var_14]
.text:00EB6D46 and     ebx, eax
```

Figure 68

The content is encrypted using the AES256 algorithm and written to the newly created file:

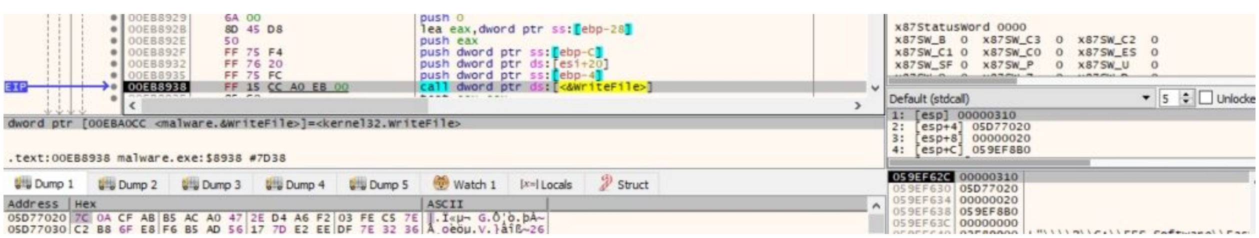


Figure 69

The file’s name is encrypted as well and will appear in the encrypted file:

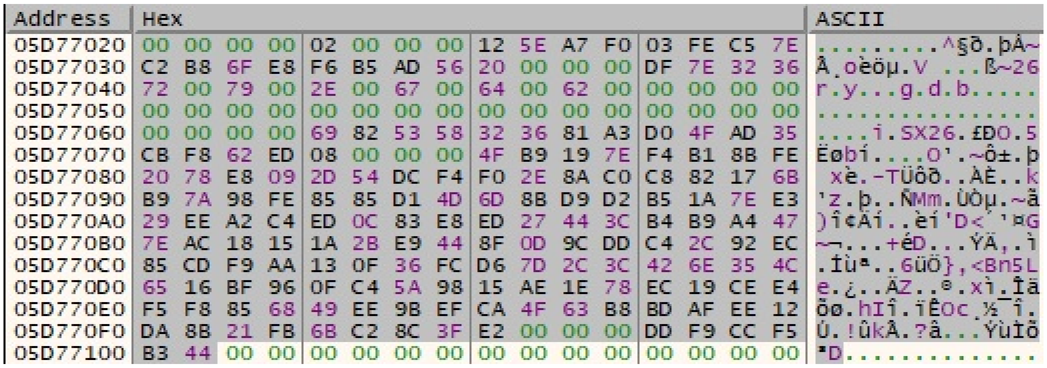


Figure 70

The following information is also written to the encrypted file: unencrypted 16-byte IV, RSA-encrypted AES256 key, and 6 bytes decrypted from the config that identifies the ransomware “DD F9 CC F5 B3 44”:

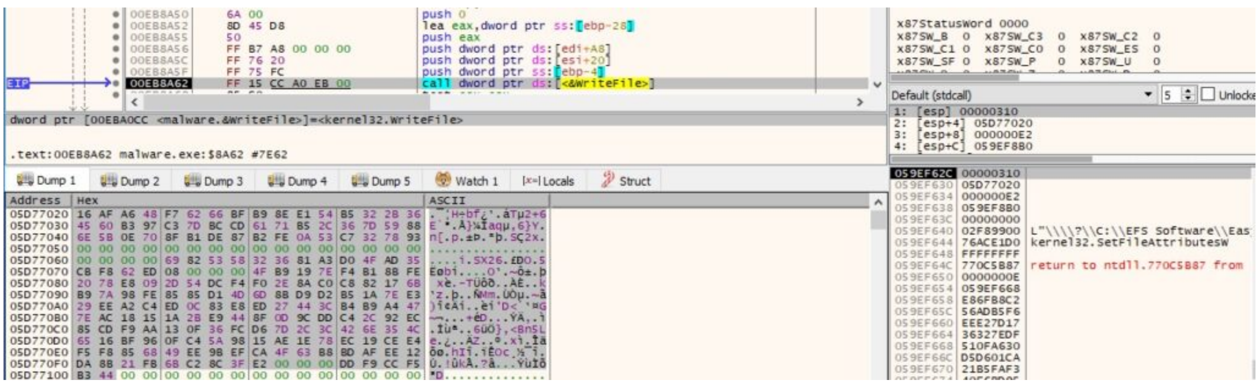


Figure 71

The unencrypted file is overwritten with zeros and deleted afterwards:

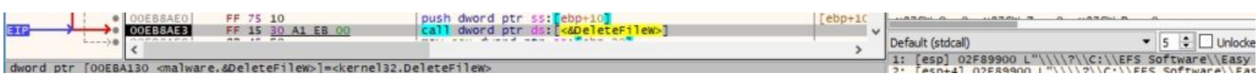


Figure 72

The structure of an encrypted file is displayed below. We’ve already described the meaning of the buffers.

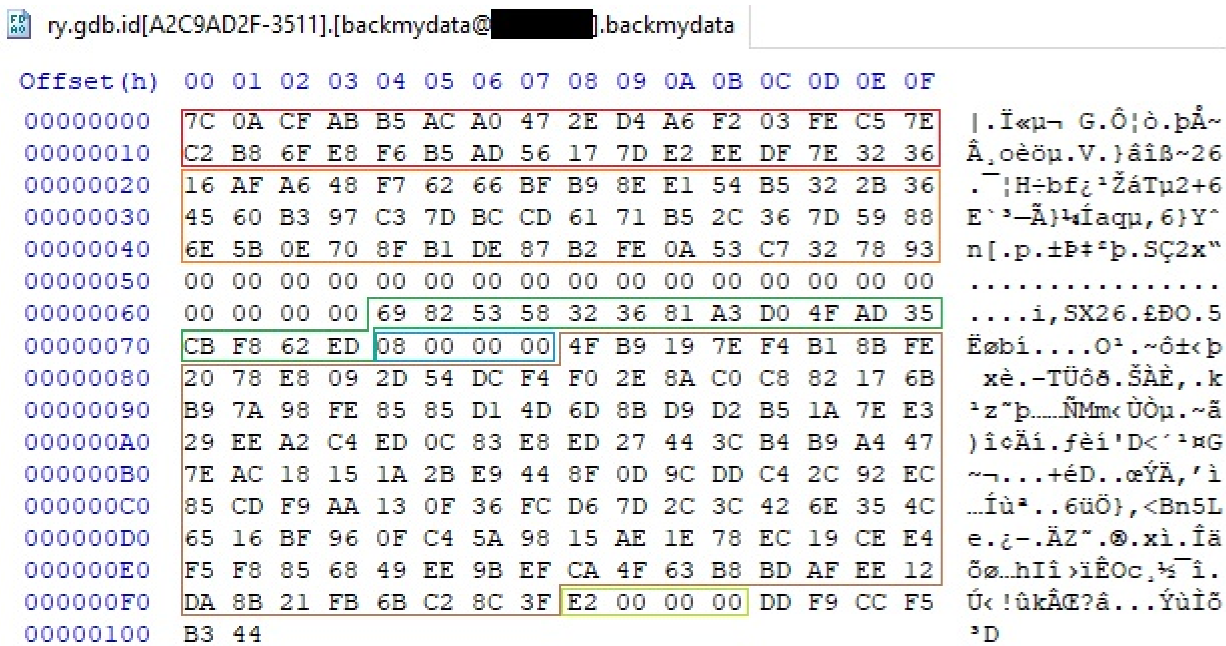
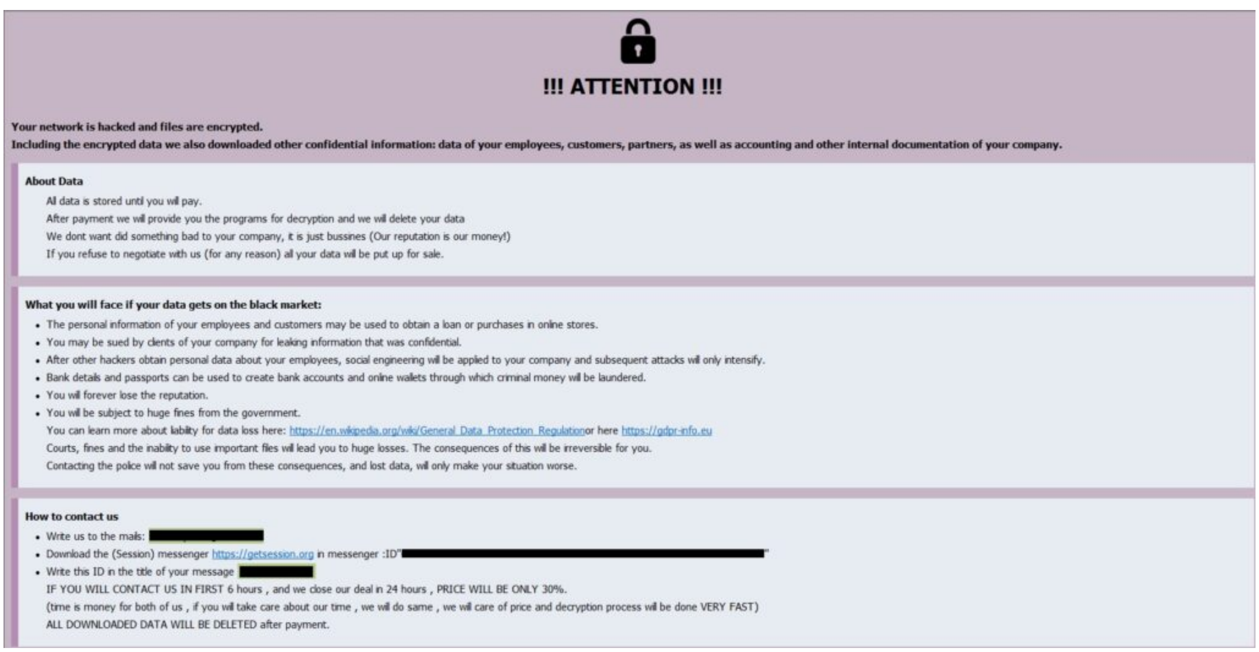


Figure 73

The ransomware drops two ransom notes: “info.txt” and “info.hta”. The communication with the threat actor can be done via email or Session messenger.



!!! ATTENTION !!!

Your network is hacked and files are encrypted.
Including the encrypted data we also downloaded other confidential information: data of your employees, customers, partners, as well as accounting and other internal documentation of your company.

About Data

All data is stored until you will pay.
After payment we will provide you the programs for decryption and we will delete your data
We dont want did something bad to your company, it is just bussines (Our reputation is our money!)
If you refuse to negotiate with us (for any reason) all your data will be put up for sale.

What you will face if your data gets on the black market:

- The personal information of your employees and customers may be used to obtain a loan or purchases in online stores.
- You may be sued by clients of your company for leaking information that was confidential.
- After other hackers obtain personal data about your employees, social engineering will be applied to your company and subsequent attacks will only intensify.
- Bank details and passports can be used to create bank accounts and online wallets through which criminal money will be laundered.
- You will forever lose the reputation.
- You will be subject to huge fines from the government.

You can learn more about liability for data loss here: https://en.wikipedia.org/wiki/General_Data_Protection_Regulationor here https://gdpr-info.eu
Courts, fines and the inability to use important files will lead you to huge losses. The consequences of this will be irreversible for you.
Contacting the police will not save you from these consequences, and lost data, will only make your situation worse.

How to contact us

- Write us to the mails: [redacted]
- Download the (Session) messenger https://getsession.org in messenger :ID"[redacted]"
- Write this ID in the title of your message [redacted]

IF YOU WILL CONTACT US IN FIRST 6 hours , and we close our deal in 24 hours , PRICE WILL BE ONLY 30%.
(time is money for both of us , if you will take care about our time , we will do same , we will care of price and decryption process will be done VERY FAST)
ALL DOWNLOADED DATA WILL BE DELETED after payment.

Figure 74

INDICATORS OF COMPROMISE

SHA256

396a2f2dd09c936e93d250e8467ac7a9c0a923ea7f9a395e63c375b877a399a6

BackMyData ransom notes

info.txt, info.hta

Files created

%AppData%\Local\<Executable name>

C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup\<Executable name>

Registry values

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run\<Executable name>

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\<Executable name>

Processes spawned

vssadmin delete shadows /all /quiet

wmic shadowcopy delete

bcdedit /set {default} bootstatuspolicy ignoreallfailures

bcdedit /set {default} recoveryenabled no

wbadmin delete catalog -quiet

netsh advfirewall set currentprofile state off

netsh firewall set opmode mode=disable

Mutexes

Global\\<<BID>><Volume serial number>00000000

Global\\<<BID>><Volume serial number>00000001

References

https://www.bleepingcomputer.com/news/security/ransomware-attack-forces-100-romanian-hospitals-to-go-offline/

https://www.malwarebytes.com/blog/news/2019/07/a-deep-dive-into-phobos-ransomware

https://blog.talosintelligence.com/deep-dive-into-phobos-ransomware/

https://docs.microsoft.com/en-us/windows/win32/api/