A Deep Dive Into Black Basta 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

Black Basta ransomware is a recent threat that compiled its first malware samples in February 2022. The ransomware deletes all Volume Shadow Copies, creates a new JPG image set as the Desktop Wallpaper and an ICO file representing the encrypted files. Unlike other ransomware families, the malware doesn't skip files based on their extensions. However, it doesn't encrypt critical folders that would make the system inoperable.

The files are encrypted using the ChaCha20 algorithm, with the key and nonce being encrypted using the RSA public key that is hard-coded in the sample. The malware can fully or partially encrypt a file depending on its size. The extension of the encrypted files is changed to .basta by the ransomware.

Analysis and findings

SHA256: ae7c868713e1d02b4db60128c651eb1e3f6a33c02544cc4cb57c3aa6c6581b6e

The process displays "ENCRYPTION" in the program window using WriteFile:

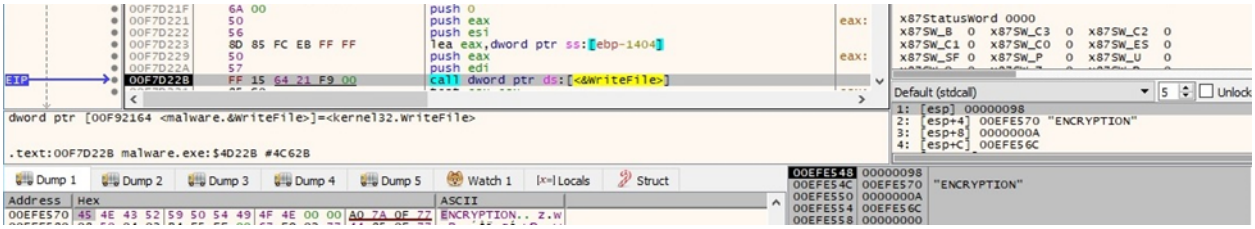

Figure 1

Figure 2

The binary retrieves the process ID via a function call to GetCurrentProcessId:

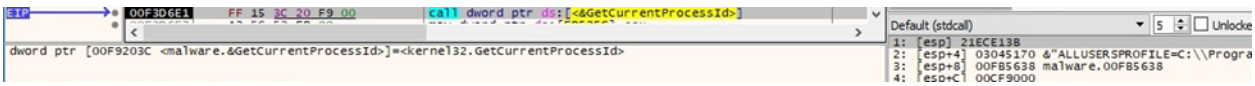

Figure 3

The malicious process detaches itself from its console by calling the FreeConsole API:

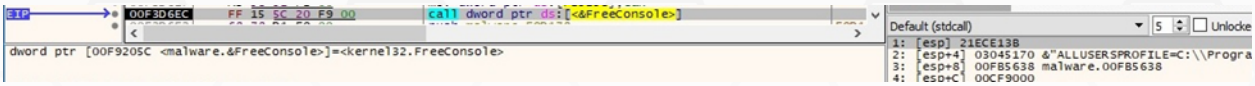

Figure 4

The executable obtains the "COMSPEC" environment variable value, which points to the command line:

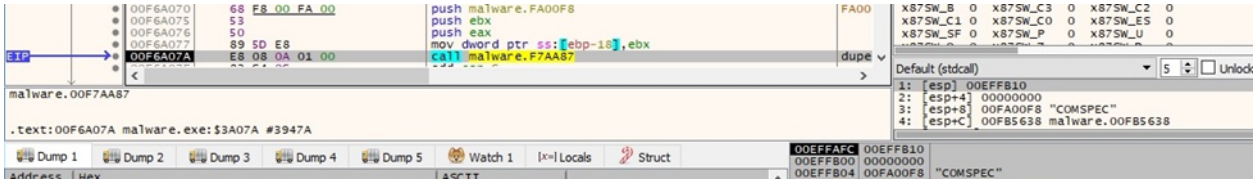

Figure 5

The ransomware deletes all Volume Shadow Copies by running the “C:\Windows\SysNative\vssadmin.exe delete shadows /all /quiet” command, as highlighted below:

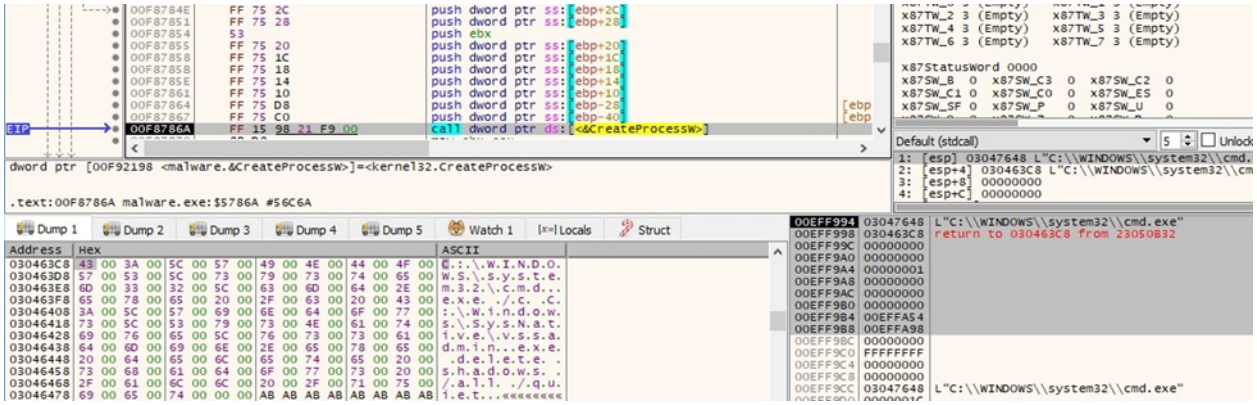

Figure 6

The sample waits until the spawned process finishes using the WaitForSingleObject routine:

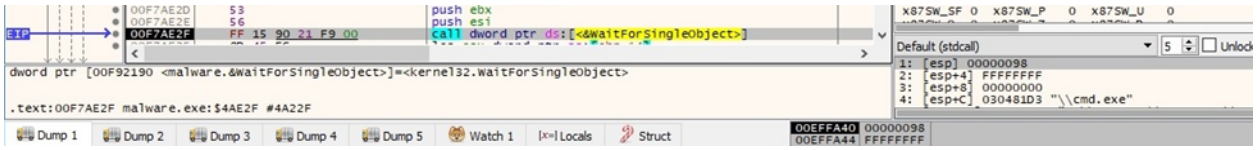

Figure 7

A similar process as above that deletes the Volume Shadow Copies is spawned:

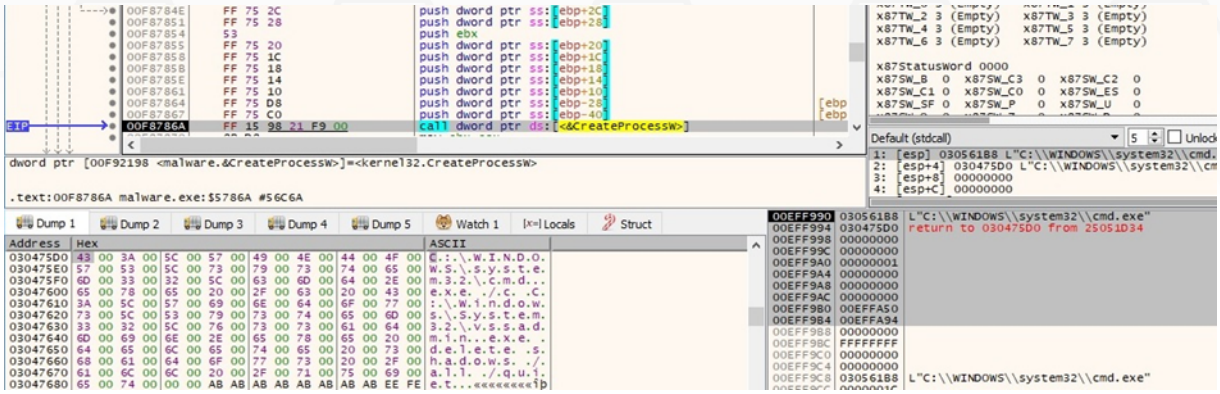

Figure 8

The binary extracts the path of the executable of the current process via a call to GetModuleFileNameW:

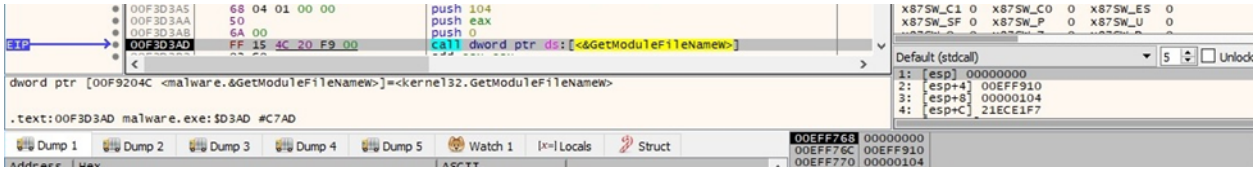

Figure 9

The GetTempPathW API is utilized to retrieve the path of the Temp directory:

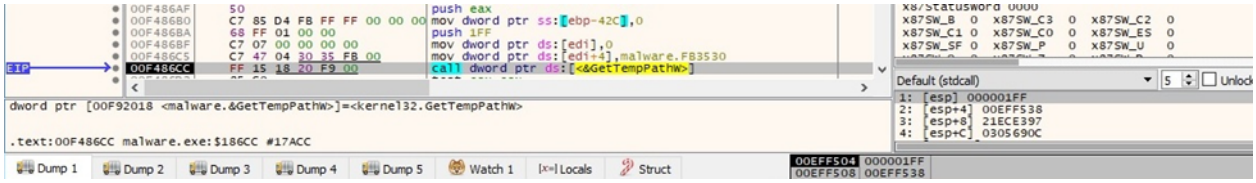

Figure 10

A file called “dlaksjdoiwq.jpg” is created in the Temp directory (0x40 = **_SH_DENYNO**):

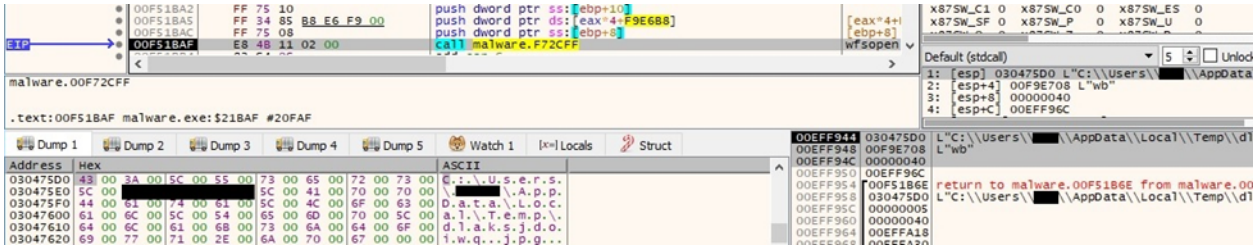

Figure 11

The process moves the file position indicator to the beginning of the file using the fsetpos function:

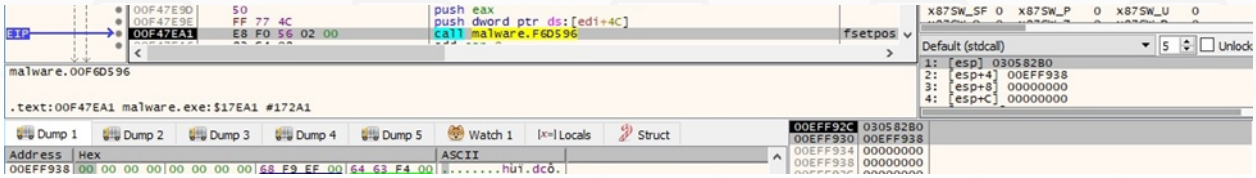

Figure 12

The WriteFile routine is used to populate the JPG file, which contains instructions from the threat actor:

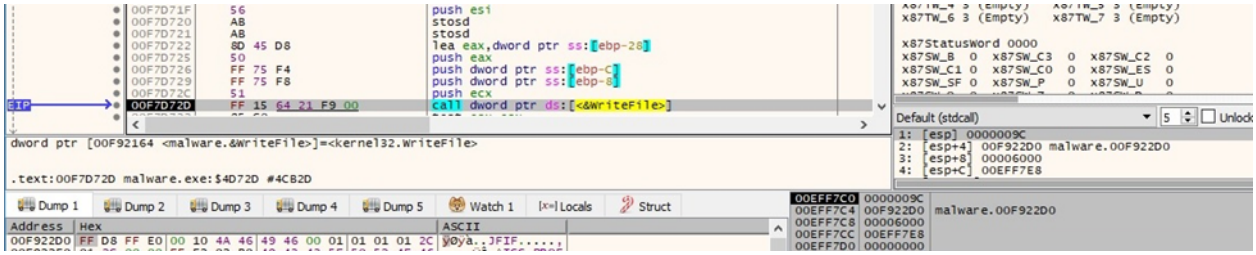

Figure 13

Figure 14

The newly created image is set as the Desktop Wallpaper using SystemParametersInfoW (0x14 = **SPI_SETDESKWALLPAPER**, 0x1 = **SPIF_UPDATEINIFILE**):

Figure 15

The executable creates an ICO file called “fkdjsadasd.ico” in the Temp directory:

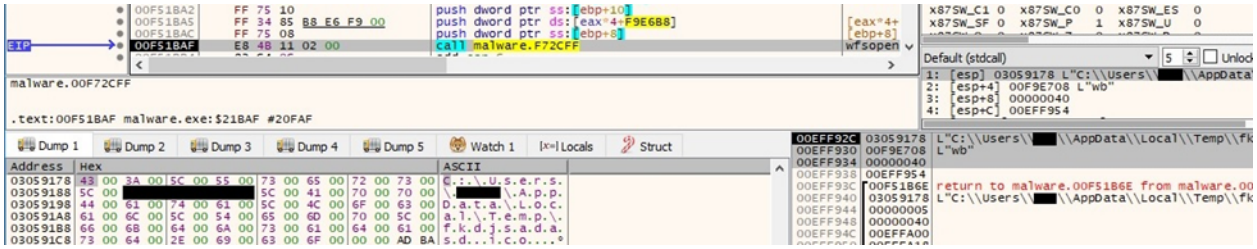

Figure 16

The ransomware writes content to the ICO file, which will represent the icon of the encrypted files:

Figure 17

Figure 18

Black Basta ransomware creates the ".basta\DefaultIcon" registry key using RegCreateKeyExW

(0x80000000 = **HKEY_CLASSES_ROOT**, 0x103 = **KEY_WOW64_64KEY** | **KEY_SET_VALUE** | **KEY_QUERY_VALUE**):

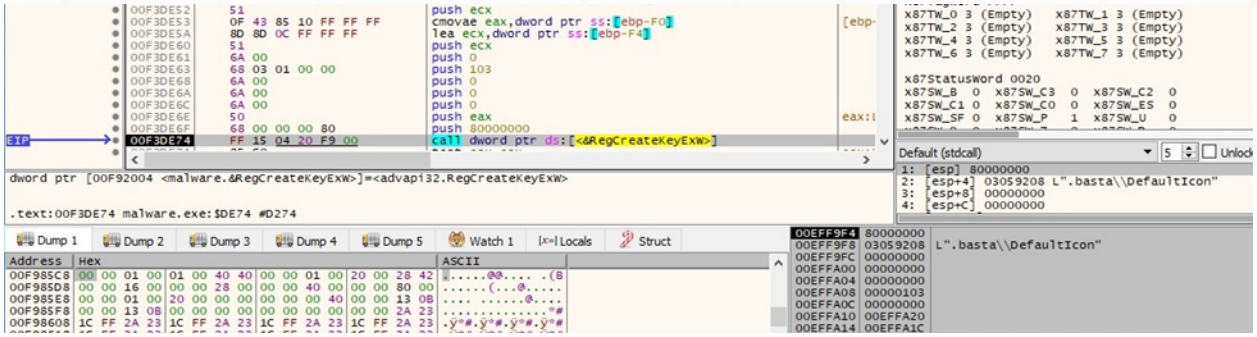

Figure 19

The “(Default)” value of the above key is set to the path of the ICO file:

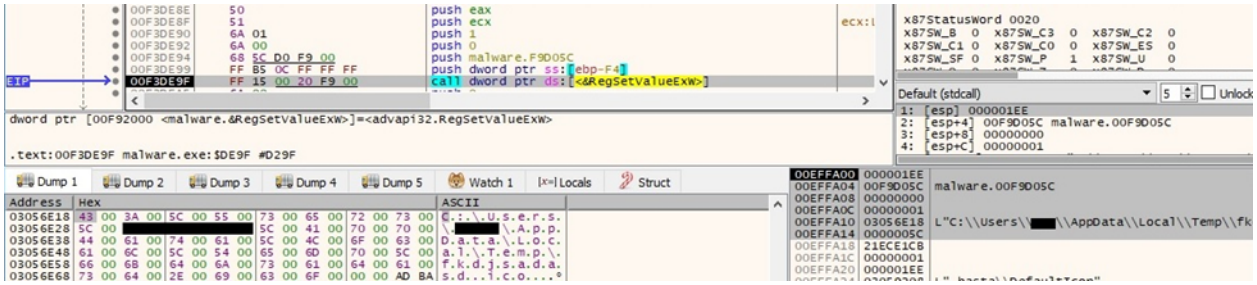

Figure 20

Figure 21

The malicious binary notifies the system that the icon has been changed by calling the SHChangeNotify function (0x08000000 = **SHCNE_ASSOCCHANGED**, 0x3000 = **SHCNF_FLUSHNOWAIT**):

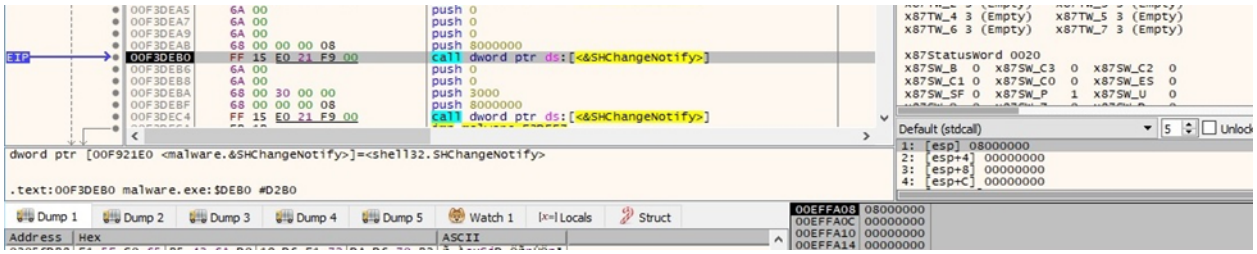

Figure 22

The malware starts scanning for volumes on the system using FindFirstVolumeW:

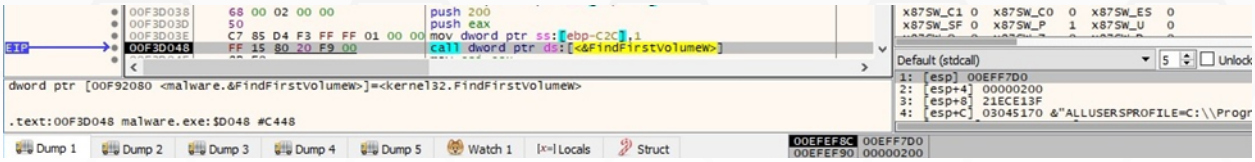

Figure 23

GetVolumePathNamesForVolumeNameW is utilized to obtain the list of drive letters and mounted folder paths for the volume:

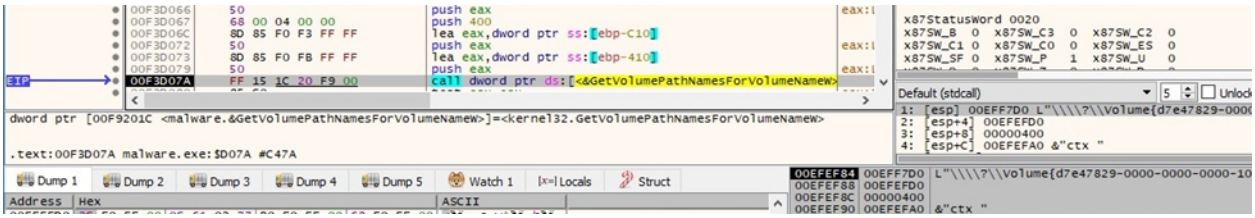

Figure 24

For each drive found, the process performs a call to the GetVolumeInformationW API (see figure 25). As opposed to other ransomware families, Black Basta only targets the mounted volumes and doesn't mount the hidden volumes.

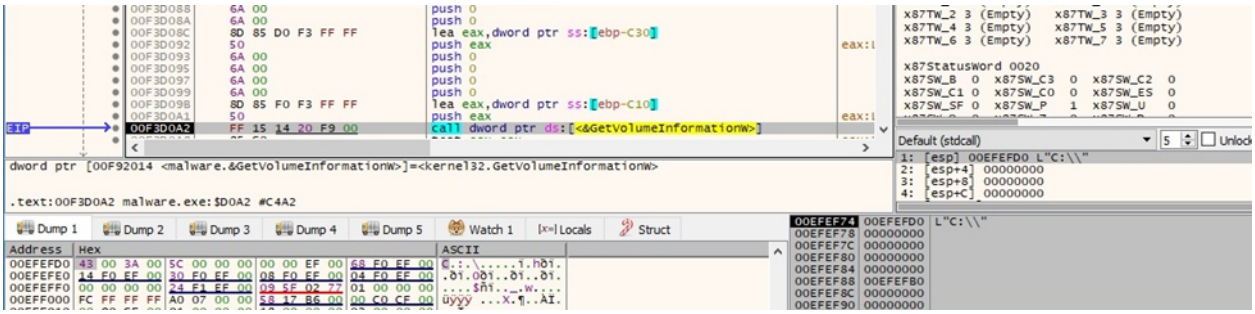

Figure 25

The volume's enumeration continues by calling the FindNextVolumeW routine:

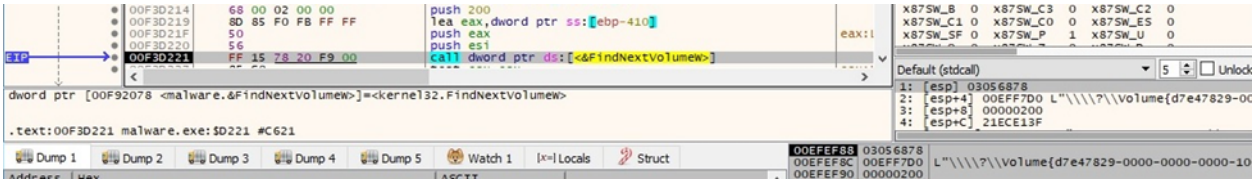

Figure 26

The ransomware extracts a standard set of attribute information from the drives found via a function call to GetFileAttributesExW (0x0 = **GetFileExInfoStandard**):

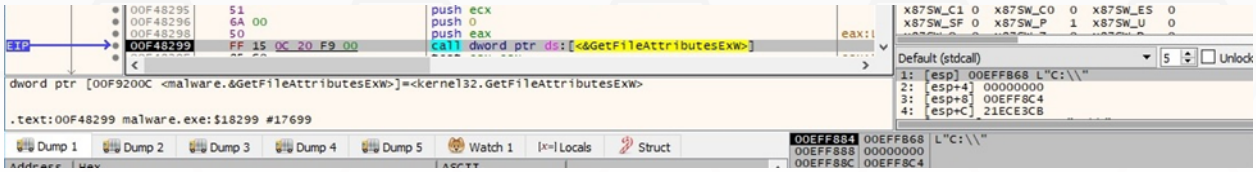

Figure 27

The ransomware creates a ransom note called “readme.txt” in every directory that is traversed, as highlighted in figure 28:

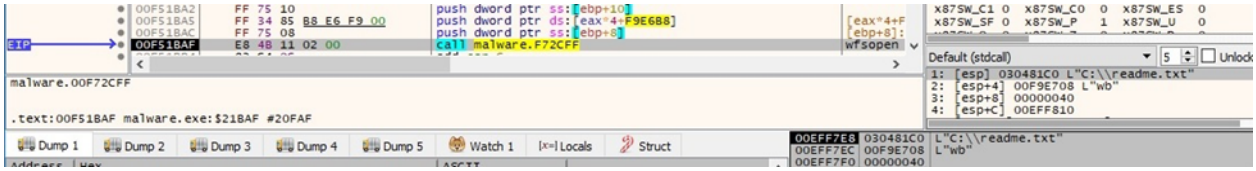

Figure 28

WriteFile is used to populate the ransom note:

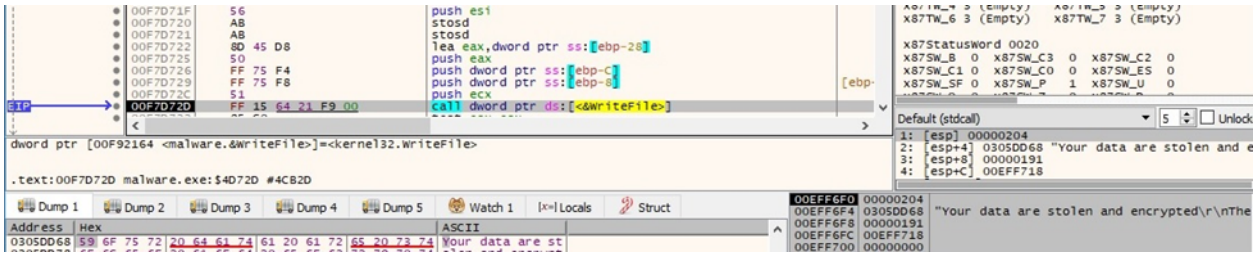

Figure 29

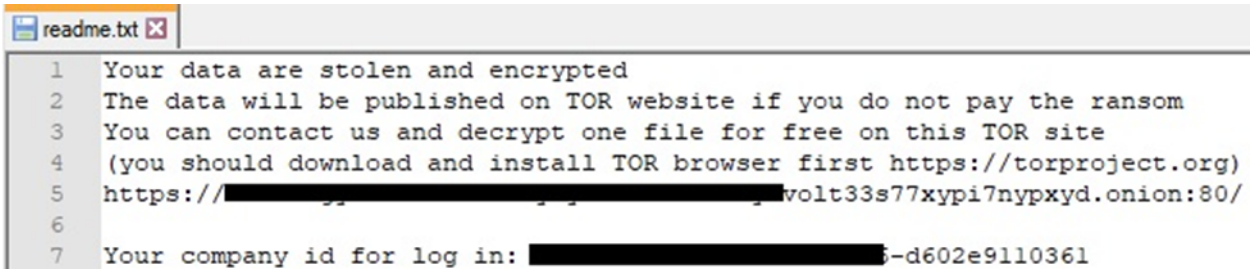

Figure 30

The binary retrieves information about the current system by calling the GetNativeSystemInfo function:

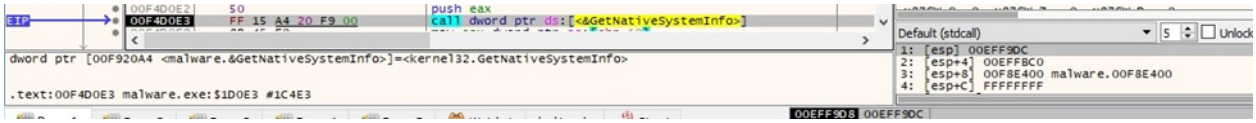

Figure 31

The malware creates multiple threads that will handle the file encryption. The function responsible for encryption is sub_F33DA0 and not the starting address of the thread:

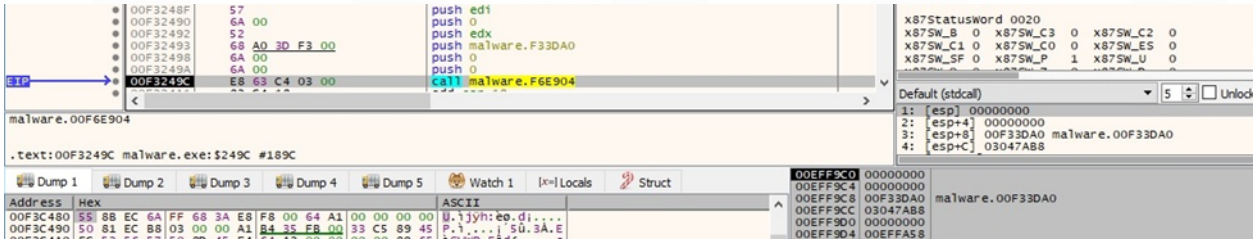

Figure 32

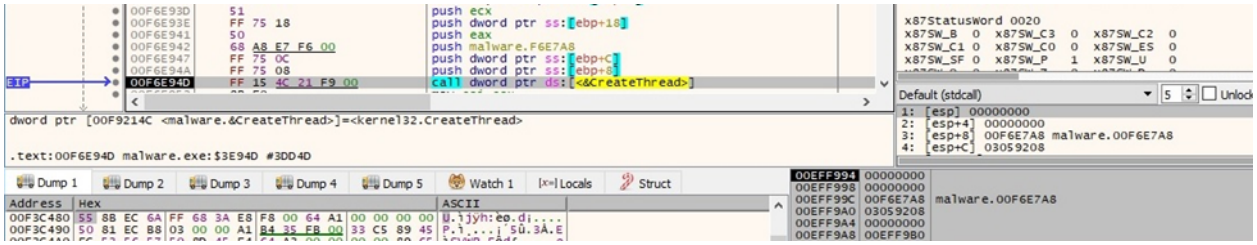

Figure 33

The malicious process starts enumerating the files on the drive using FindFirstFileW:

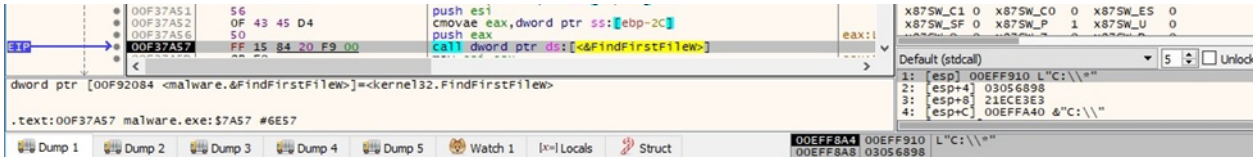

Figure 34

As shown in figure 35, the following files/directories will be skipped:

- $Recycle.Bin
- Windows
- boot
- readme.txt
- dlaksjdoiwq.jpg
- NTUSER.DAT
- fkdjsadasd.ico

```
.rdata:00F9D0A8                 text "UTF-16LE", '$Recycle.Bin',0
.rdata:00F9D0C2                 align 4
.rdata:00F9D0C4 aWindows:                               ; DATA XREF: sub_F3BBE0:loc_F3BF44to
.rdata:00F9D0C4                 text "UTF-16LE", 'Windows',0
.rdata:00F9D0D4 aBoot:                                  ; DATA XREF: sub_F3BBE0:loc_F3BF8Bto
.rdata:00F9D0D4                 text "UTF-16LE", 'boot',0
.rdata:00F9D0DE                 align 10h
.rdata:00F9D0E0 aReadmeTxt:                             ; DATA XREF: sub_F3B3D0+3Cto
.rdata:00F9D0E0                                         ; sub_F3BBE0:loc_F3BFD5to
.rdata:00F9D0E0                 text "UTF-16LE", 'readme.txt',0
.rdata:00F9D0F6                 align 4
.rdata:00F9D0F8 aDlaksjdoiwqJpg:                        ; DATA XREF: sub_F3BBE0:loc_F3C01Fto
.rdata:00F9D0F8                                         ; sub_F3DCA0+5Cto
.rdata:00F9D0F8                 text "UTF-16LE", 'dlaksjdoiwq.jpg',0
.rdata:00F9D118 aNtuserDat:                             ; DATA XREF: sub_F3BBE0:loc_F3C069to
.rdata:00F9D118                 text "UTF-16LE", 'NTUSER.DAT',0
.rdata:00F9D12E                 align 10h
.rdata:00F9D130 aError755       db 'Error 755: ',0      ; DATA XREF: sub_F3BBE0:loc_F3C29Dto
.rdata:00F9D13C aFkdjsadasdIco:                         ; DATA XREF: sub_F3DB50+5Cto
.rdata:00F9D13C                 text "UTF-16LE", 'fkdjsadasd.ico',0
.rdata:00F9D15A                 align 4
```

Figure 35

The FindNextFileW routine is utilized to continue the files enumeration:

Figure 36

Black Basta ransomware calls the GetFullPathNameW API with a targeted file as a parameter:

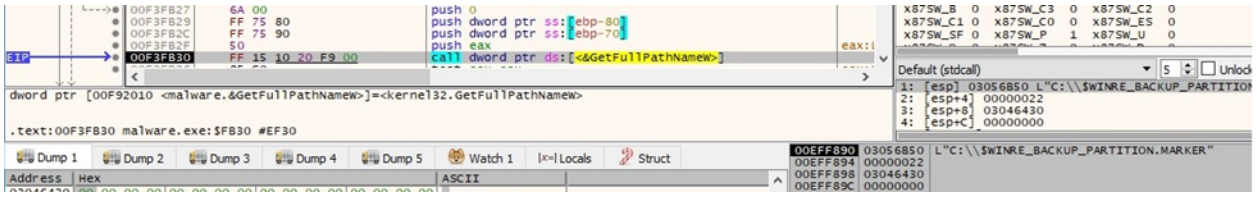

Figure 37

The process obtains a standard set of attribute information for the file via a call to GetFileAttributesExW:

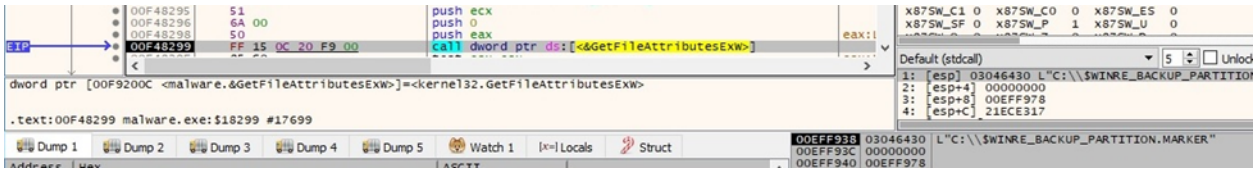

Figure 38

The ransomware has embedded a list of extensions (.exe, .cmd, .bat, and .com) in a section; however, it still encrypts these file extensions.

The executable retrieves the thread identifier of the calling thread using GetCurrentThreadId:

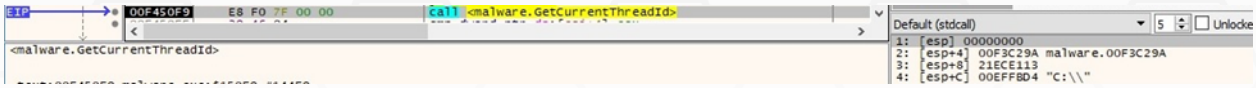

Figure 39

The malicious process blocks the main thread until all encryption threads finish execution (see figure 40).

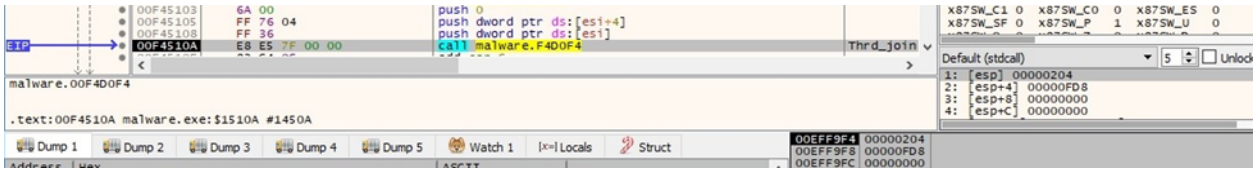

Figure 40

Thread activity – sub_F33DA0 function

The GetFileAttributesW API is utilized to retrieve file system attributes for a targeted file:

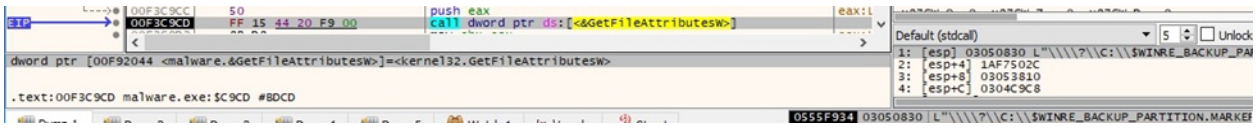

Figure 41

The malicious process opens a file for reading using wfsopen:

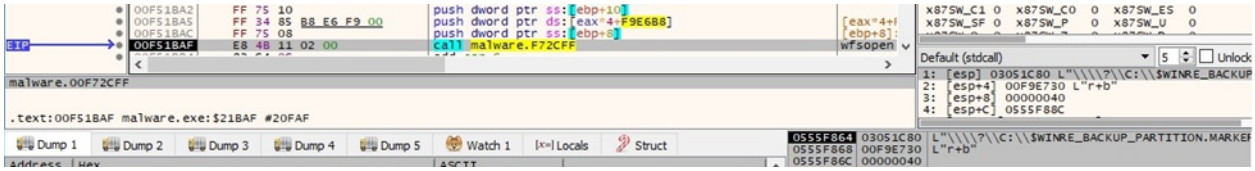

Figure 42

The ransomware moves the file pointer to the position of the last 4 bytes. Whether the file would be encrypted, these would represent the length of the encrypted ChaCha20 key and nonce, as we'll see later on:

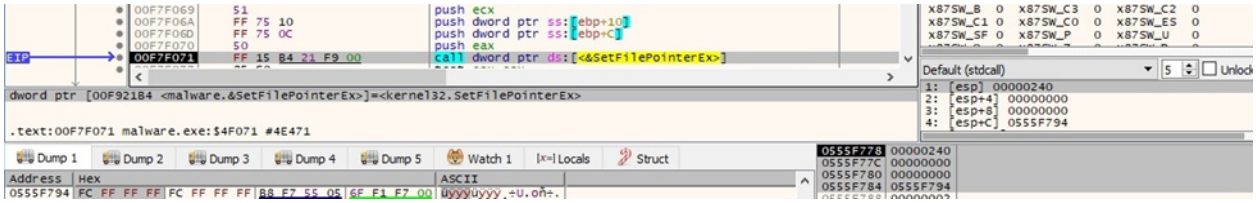

Figure 43

Black Basta ransomware generates 32 random bytes representing the ChaCha20 key and then

8 bytes representing the nonce using rand_s:

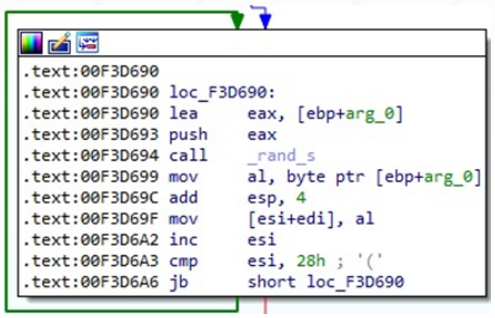

Figure 44

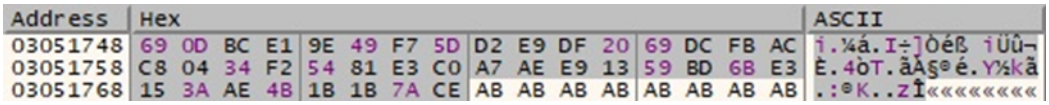

Figure 45

The binary implements the RSA algorithm using the Mini-GMP library, which is fully available on Github:

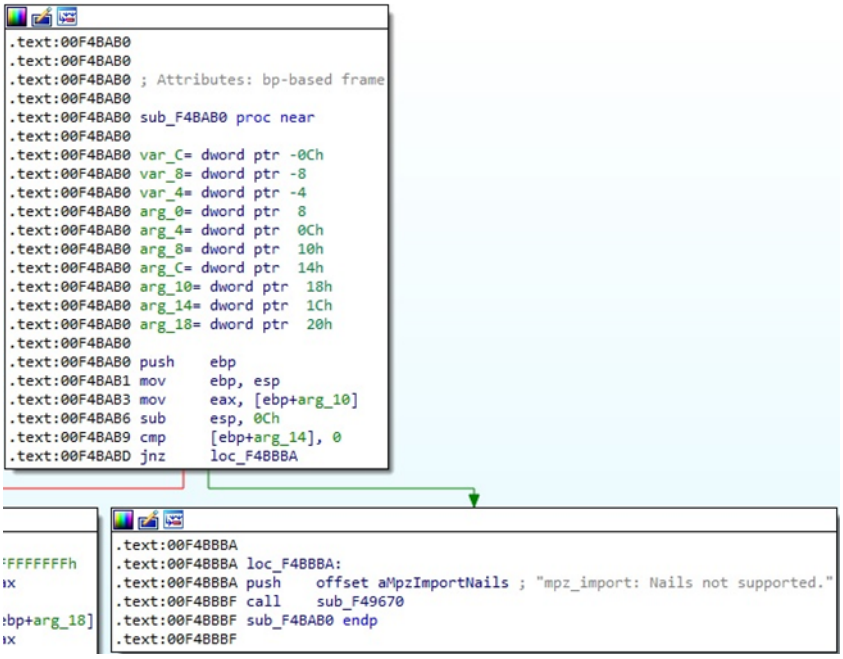

Figure 46

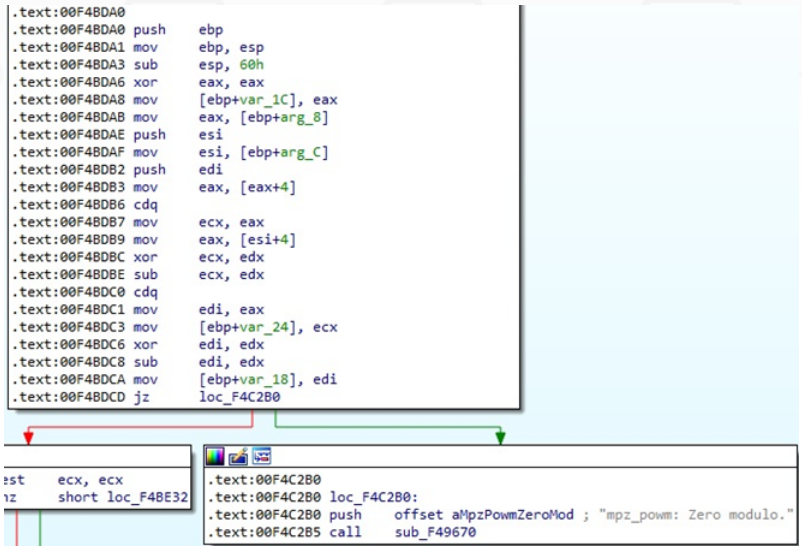

Figure 47

The RSA public key used to encrypt the randomly generated ChaCha20 key and the nonce is presented in the figure below:

```
.rdata:00F9D1F8 aZz11ttcaoj0zrc db 'zz11tTCaoj0ZRc3xITYjF3g8OU8OBkMvQR3vA/EVuVXFMg+jdmyjEhLhEqLATJKqg'
.rdata:00F9D1F8                                 ; DATA XREF: sub_F3D6B0+188To
.rdata:00F9D1F8                 db '/BnWIq2T6dpvX6ycqNxo6FYbjbmS2nmsznwRNN6e04vyXIo7c2gbWh0rS51qSIVPs'
.rdata:00F9D1F8                 db '0r2kF0mj0ES6ukt9/7gXUB7qAfQp2eY2iraxqpI4YUM5A2EK+AYNbXYmv2qqABYbB'
.rdata:00F9D1F8                 db 'QuhX0yHu6z24cC4GrNRVKtL0wk1FeY6JFSzG7OzcfHZJxo23oArVb/c0ZGZyMhcrN'
.rdata:00F9D1F8                 db 'Xl7bGLTPIu9ZGz+TV1jo76cvi/DF5qPfh7jq5VBzHMNXEYfdedxMe9rate17YZMAI'
.rdata:00F9D1F8                 db 'EUhJPeb9oGAeN9n8jf6MHTASx48+bU6Vn+EFG4a1JHoEj/KGY0nw6FwN8Wex5Ov1I'
.rdata:00F9D1F8                 db 'a/USqQtOURCJOo3EaOHuPGO2eVUVMIeOS6iabcbIVR3POrdNEi5Bt6bTgZs01Si1Q'
.rdata:00F9D1F8                 db 'XRu4eT3jCTSjlFxU8VRruAinyOOLi6vI2pUgcRJaxFRwldzyUuX5kd67/0JyXHTuw'
.rdata:00F9D1F8                 db 'VkiMTuFtpCnyj+7bMfg46LXIXC4PtzQvgxDewRppyzmT5Sdx28TXOwTwaGvMiZhll'
.rdata:00F9D1F8                 db 'y+66Nu+wmepHZ7u/WTDJg9H8V/AZbVDatAZ1vFy2tZws3IFVNE6dqI6lvIzgPca4o'
.rdata:00F9D1F8                 db 'xyi8+5JIhoZdSRpW05+A8vk0AkMMPNgGU=',0
```

Figure 48

The process constructs the initial state of ChaCha20 using the key, the nonce, and some constant values:

```
.text:00F368D0
.text:00F368D0
.text:00F368D0 ; Attributes: bp-based frame
.text:00F368D0
.text:00F368D0 sub_F368D0 proc near
.text:00F368D0
.text:00F368D0 arg_0= dword ptr  8
.text:00F368D0 arg_4= dword ptr  0Ch
.text:00F368D0
.text:00F368D0 push    ebp
.text:00F368D1 mov     ebp, esp
.text:00F368D3 push    esi
.text:00F368D4 mov     esi, [ebp+arg_0]
.text:00F368D7 push    edi
.text:00F368D8 mov     edi, ecx
.text:00F368DA mov     dword ptr [edi], 'apxe'
.text:00F368E0 mov     dword ptr [edi+4], '3 dn'
.text:00F368E7 mov     dword ptr [edi+8], 'yb-2'
.text:00F368EE mov     dword ptr [edi+0Ch], 'k et'
.text:00F368F5 movzx   edx, byte ptr [esi+3]
.text:00F368F9 movzx   eax, byte ptr [esi+2]
.text:00F368FD shl     edx, 8
.text:00F36900 or      edx, eax
.text:00F36902 movzx   eax, byte ptr [esi+1]
```

Figure 49

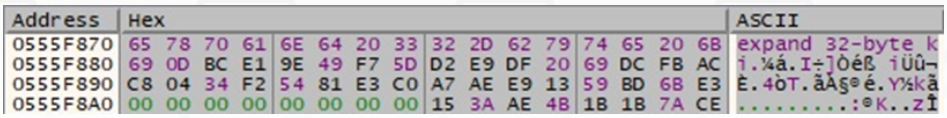

Figure 50

The sample obtains the current position in the targeted file by calling the fgetpos function:

Figure 51

The file content is read by the process via a call to the _read function:

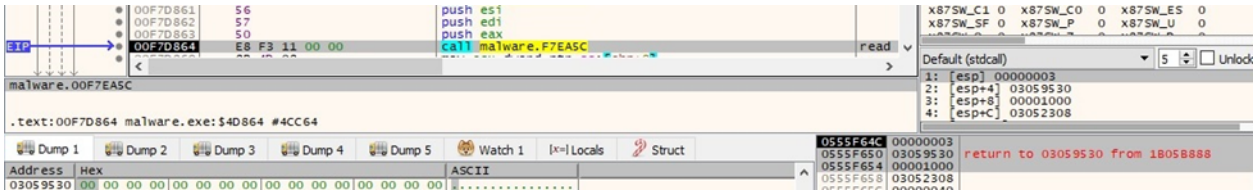

Figure 52

The content is encrypted by the ChaCha20 algorithm 64 bytes at a time:

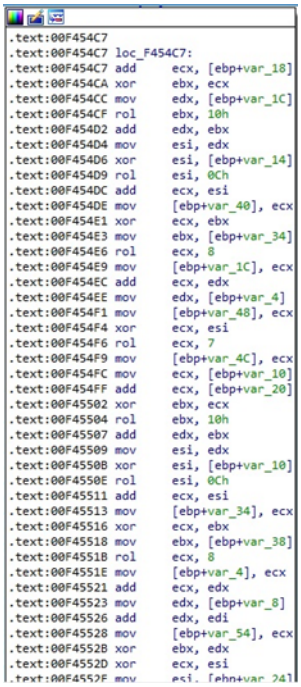

Figure 53

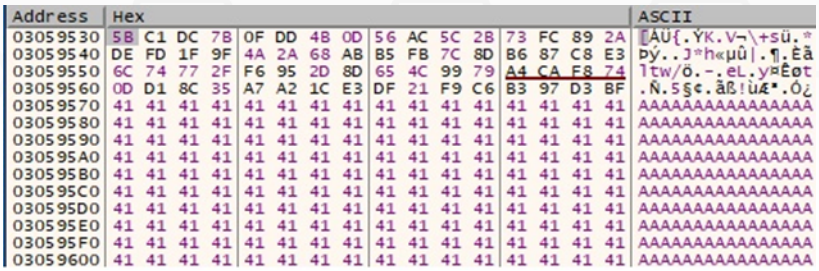

Figure 54

The encrypted data is written back to the file using the WriteFile API:

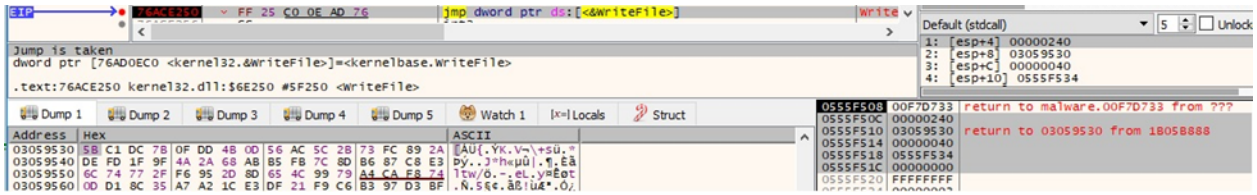

Figure 55

The buffer containing the RSA encrypted ChaCha20 key and nonce is appended to the encrypted file. The length of the encrypted information (0x200 = 512) is added as well:

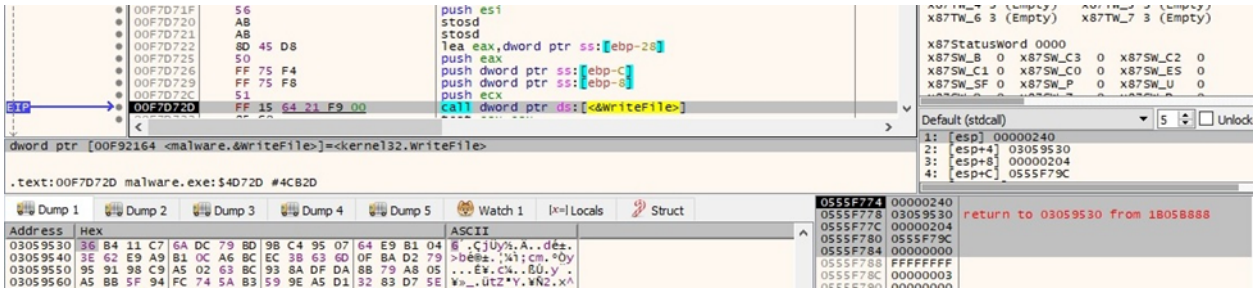

Figure 56

The encrypted file extension is changed to ".basta" using MoveFileW:

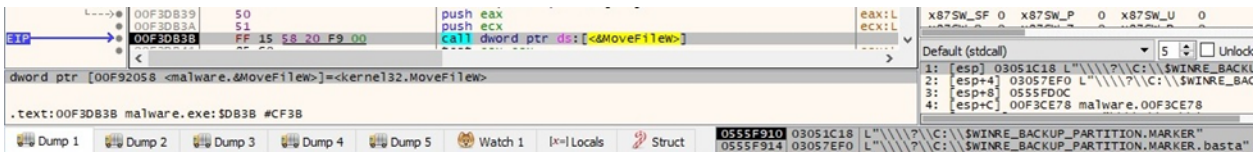

Figure 57

Case 1 – File size < 704 bytes

In this case, the entire file content is encrypted by the ransomware:

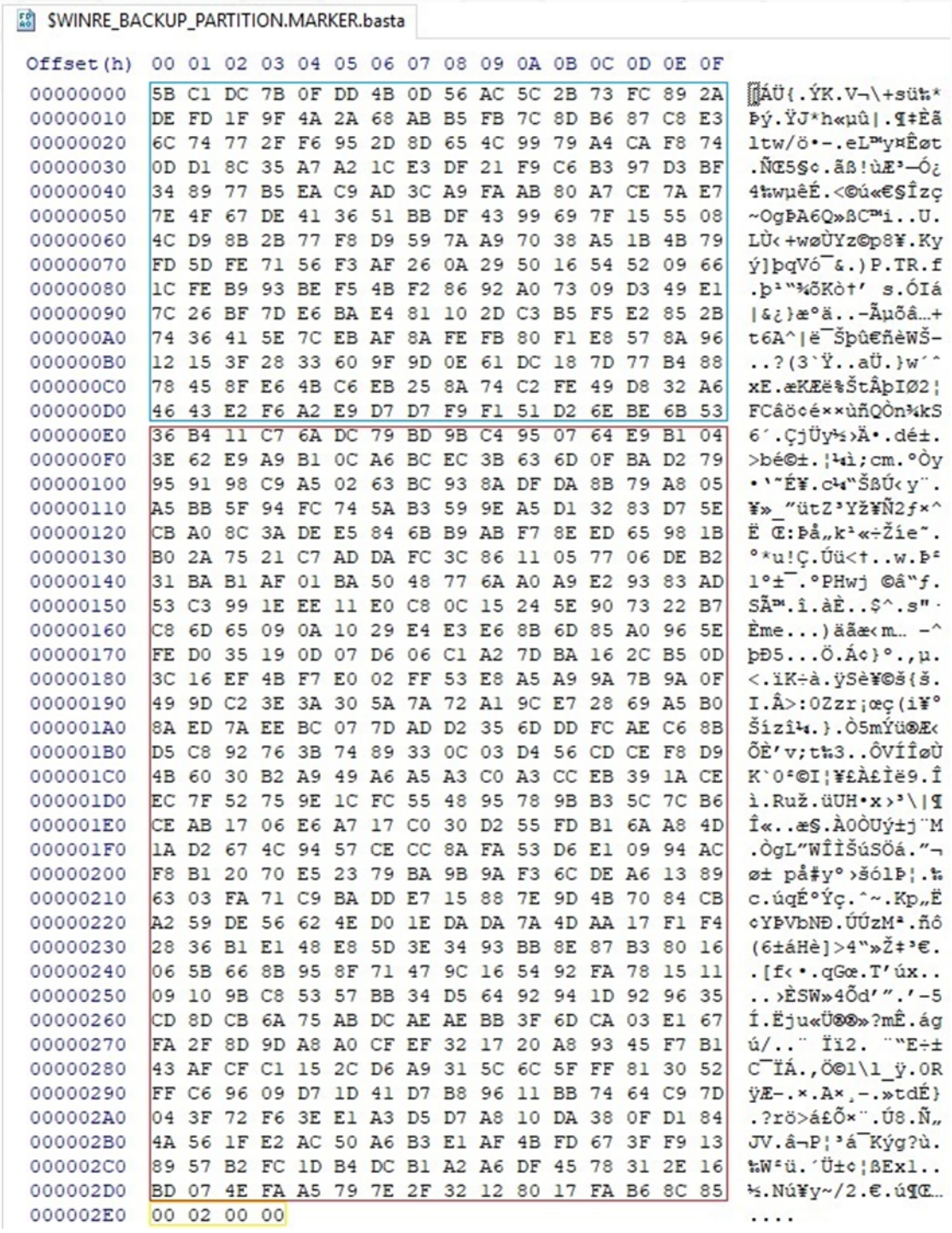

Figure 58

Case 2 – File size < 4KB

In this case, the file is partially encrypted. The ransomware encrypts 64 bytes, skips 192 bytes,

encrypts 64 bytes again, and so on.

1KB.exe.basta

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000000	6C	FE	7A	E5	94	39	63	A1	A6	19	45	42	65	B3	1F	83	lþzå”9c¡¦.EBe³.ƒ
00000010	F4	A2	25	4E	22	2F	62	E9	6A	E2	19	5E	97	AF	91	40	ô¢%N"/béjâ.^—¯'@
00000020	5C	45	A0	96	2B	36	C6	69	1A	74	D3	EE	1A	73	71	F2	\E –+6Æi.tÓî.sqò
00000030	97	5C	44	08	E2	CF	2E	98	99	DC	BF	57	DC	E1	58	65	—\D.âÏ.˜™Ü¿WÜáXe
00000040	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
00000050	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
00000060	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
00000070	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
00000080	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
00000090	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
000000A0	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
000000B0	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
000000C0	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
000000D0	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
000000E0	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
000000F0	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
00000100	35	08	A4	ED	AB	D6	81	C7	3B	8A	04	BA	65	2C	25	13	5.¤í«Ö.Ç;Š.ºe,%.
00000110	94	7B	E4	A1	CE	AD	9B	D0	1F	6C	9F	DA	66	7F	66	D0	”{ä¡Î.›Ð.lŸÚf.fÐ
00000120	3D	1A	54	D4	4F	95	A4	31	D6	FC	FA	9F	B3	AB	F3	03	=.TÔO•¤1ÖüúŸ³«ó.
00000130	15	1D	B8	62	3F	9D	1B	F0	DD	29	16	13	76	5E	19	FE	..¸b?..ðÝ)..v^.þ
00000140	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
00000150	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
00000160	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
00000170	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
00000180	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
00000190	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
000001A0	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
000001B0	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
000001C0	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
000001D0	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
000001E0	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
000001F0	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
00000200	60	4D	0B	33	40	DF	D4	98	1D	10	E1	C5	D3	0F	AC	60	`M.3@ßÔ˜..áÅÓ.¬`
00000210	7A	99	98	45	35	75	AD	4E	24	37	49	5C	49	5E	FE	45	z™˜E5u.N$7I\I^þE
00000220	E1	89	BE	E4	08	EF	FC	22	10	DA	62	F1	B5	A0	80	E7	á‰¾ä.ïü".Úbñµ €ç
00000230	AE	BF	91	27	84	39	D5	BA	65	26	85	6A	52	FE	C0	1B	®¿'‘„9Õºe&…jRþÀ.

Figure 59

Case 3 – File size > 4KB

In this case, the file is partially encrypted. The ransomware encrypts 64 bytes, skips 128 bytes, encrypts 64 bytes again, and so on.

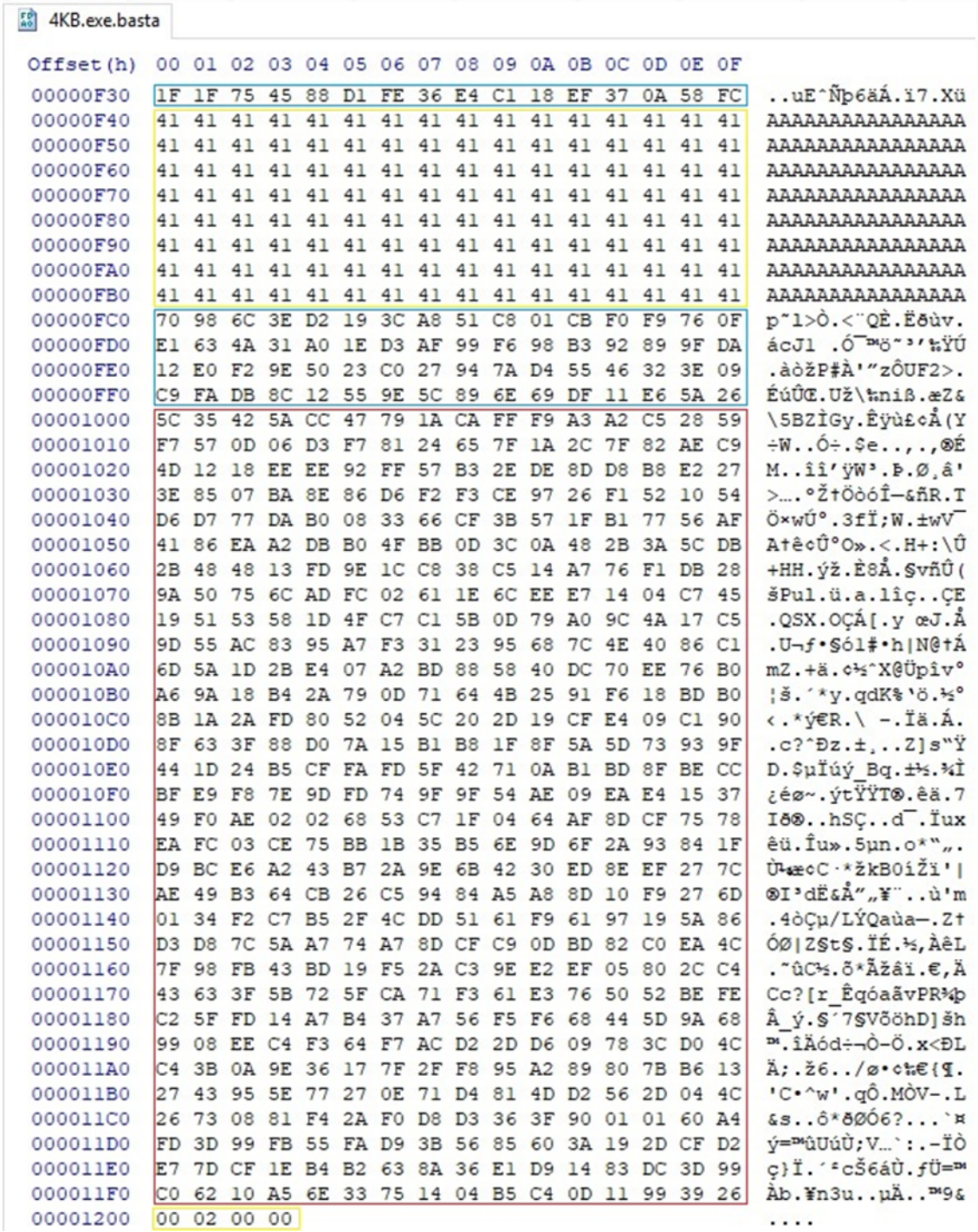

Figure 60

Finally, the ransomware tries to write the time spent during the execution and the total size of

encrypted files to the console; however, it raises an error because the process was detached from its console:

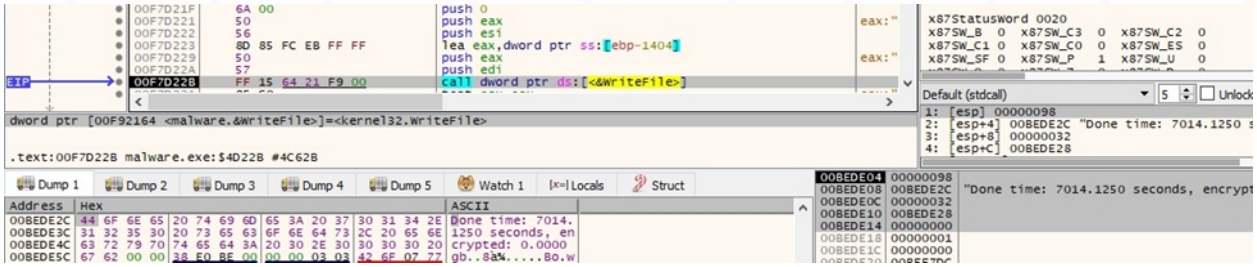

Figure 61

Indicators of Compromise

Black Basta Ransom Note

readme.txt

Files created

%Temp%\fkdjsadasd.ico

%Temp%\dlaksjdoiwq.jpg

Processes spawned

cmd.exe /c “C:\Windows\SysNative\vssadmin.exe delete shadows /all /quiet”

cmd.exe /c “C:\Windows\System32\vssadmin.exe delete shadows /all /quiet”

Registry key created

HKEY_CLASSES_ROOT\.basta