LIFARS
your digital world, **secured**

A Detailed Analysis of The Last Version of Conti Ransomware

Prepared by: LIFARS, LLC
Date: 10/08/2021

EXECUTIVE SUMMARY

Conti ransomware has been sold as a RaaS (Ransomware as a Service) in underground forums and it's usually deployed by other malware such as TrickBot and BazaLoader/BazarLoader. It can run with one of the following parameters: "-p", "-m", "-size", "-log" and "-nomutex". A new mutex called "YUIOGHJKCVVBNMFGHJKTYQUWIETASKDHGZBDGSKL237782321344" can be created to ensure that only one instance of ransomware is running at a single time. The malware has the ability to only encrypt network shares ("-m net" parameter), local drives ("-m local" parameter), or both of them ("-m all" parameter). The volume shadow copies are deleted using wmic and COM objects. The algorithm used to encrypt files is ChaCha8, with the key and nonce being encrypted using an RSA public key.

LIFARS your digital world, secured
244 Fifth Avenue, Suite 2035, New York, NY 10001
LIFARS.com (212) 222-7061 info@lifars.com

ANALYSIS AND FINDIGS

SHA256: 4bfd58d4e4a6fe5e91b408bc190a24d352124902085f9c2da948ad7d79b72618

The malware obfuscates the stack strings and implements multiple custom algorithms to decrypt them. An example of a decryption algorithm is shown below, along with the decrypted string:

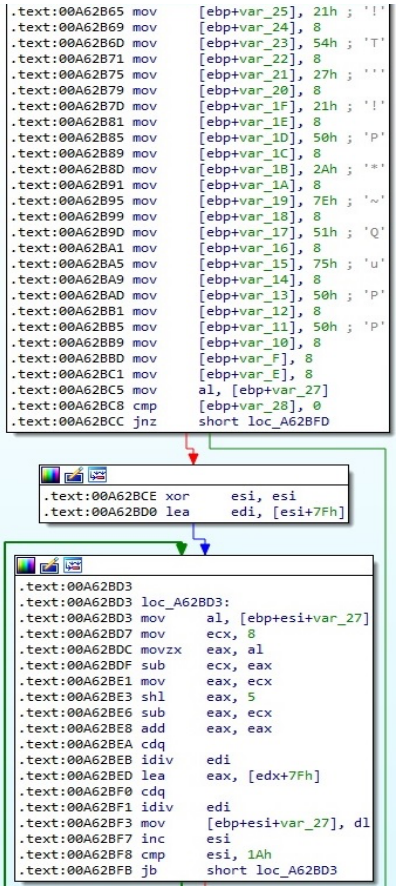


Figure 1

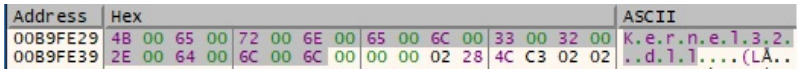


Figure 2

The relevant APIs are imported dynamically at runtime using some hashing algorithms (the first parameter is a hash value; the second parameter is an offset). The return value is placed into the EAX register:

LIFARS your digital world, secured
244 Fifth Avenue, Suite 2035, New York, NY 10001
LIFARS.com (212) 222-7061 info@lifars.com

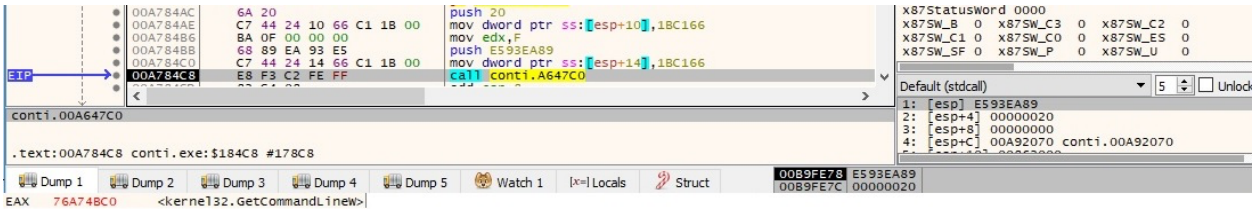


Figure 3

The binary retrieves the command-line string for the process by calling the GetCommandLineW API:

Figure 4

CommandLineToArgvW is utilized to extract an array of pointers to the command line arguments, as shown in figure 5:

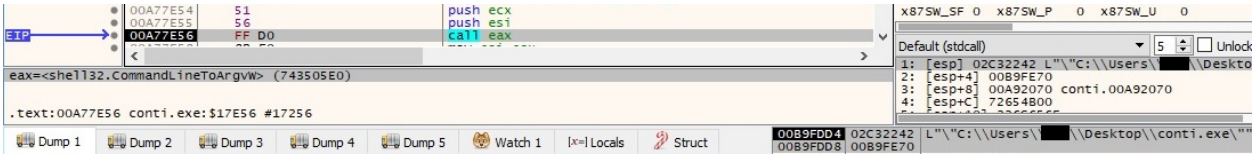


Figure 5

The following strings have been decrypted using algorithms like the one presented in figure 1:

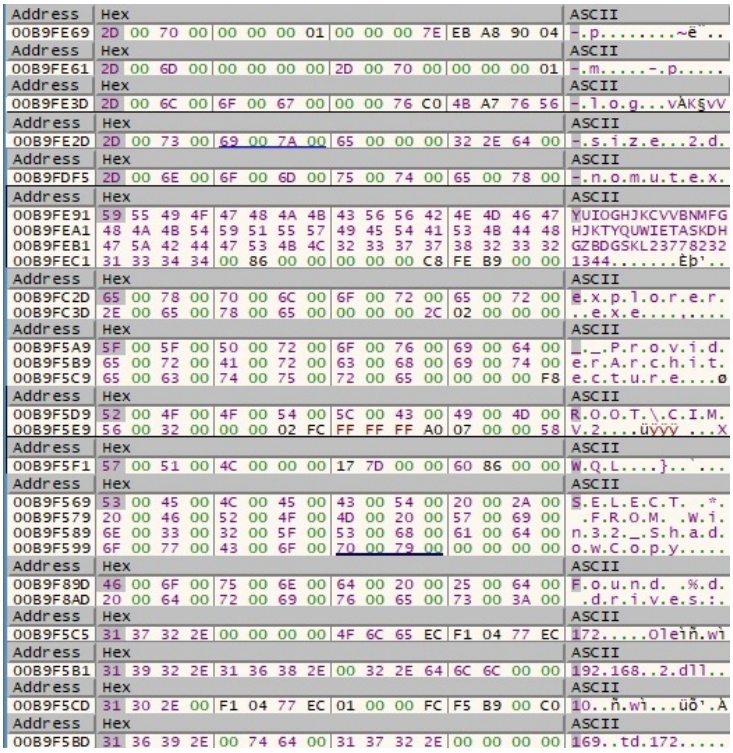


Figure 6

The executable creates a mutex called "YUIOGHJKCVVBNMFGHJKTYQUWIETASKDHGZBDGSKL237782321344" (if the malware runs with the "-nomutex" parameter, then no mutex is created):

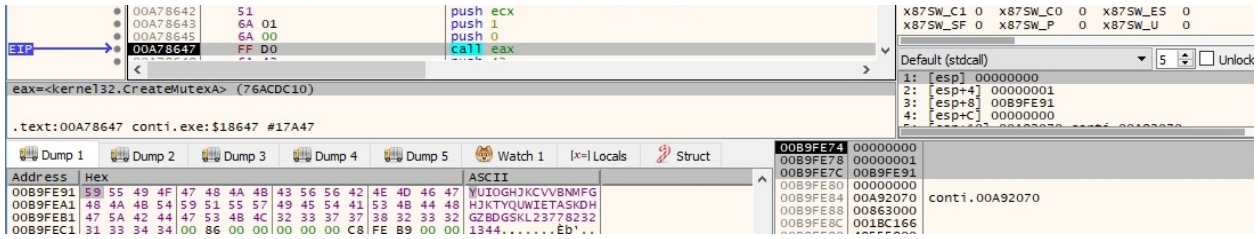


Figure 7

GetNativeSystemInfo is used to retrieve information about the system:

Figure 8

The malicious file creates 2 (which is the number of processors) threads that will handle the files encryption, as we'll describe in the upcoming paragraphs:

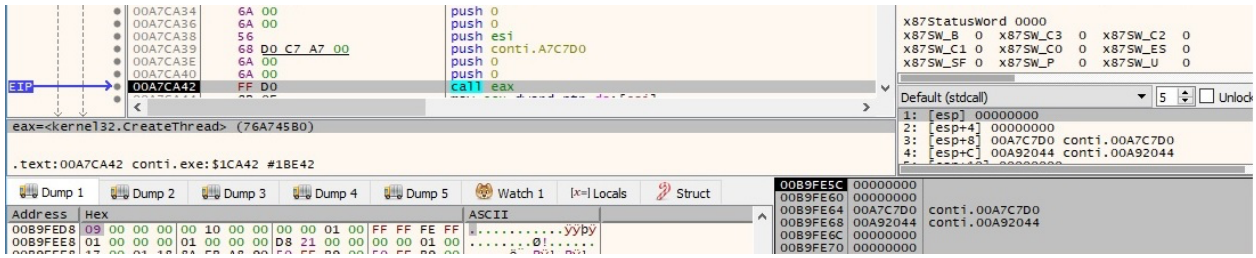


Figure 9

The executable takes a snapshot of all processes in the system by calling the CreateToolhelp32Snapshot routine (0x2 = **TH32CS_SNAPPROCESS**):

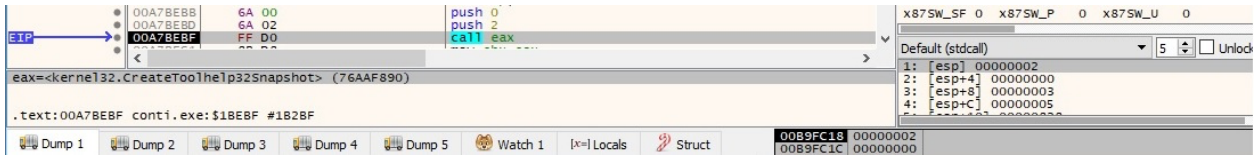


Figure 10

The processes are enumerated using the Process32FirstW and Process32NextW APIs:

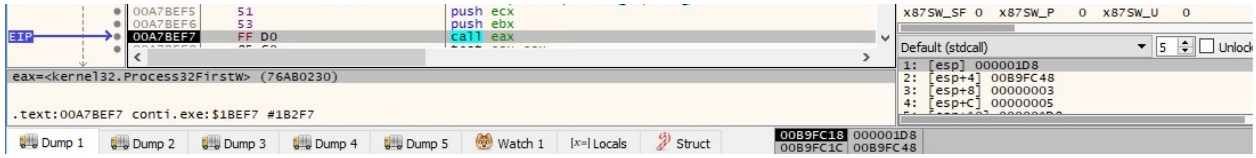


Figure 11

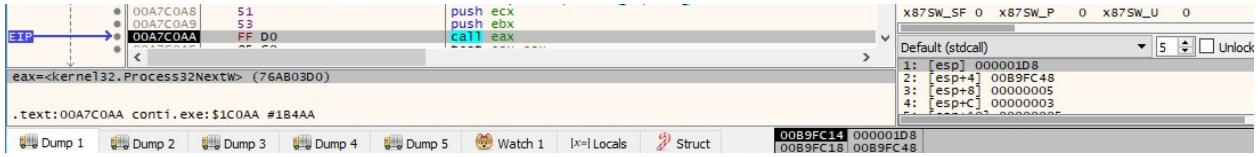


Figure 12

The malware searches for the "explorer.exe" process and saves its ID into a buffer for later use. The CoInitializeEx function is utilized to initialize the COM library for use by the thread, as highlighted below:

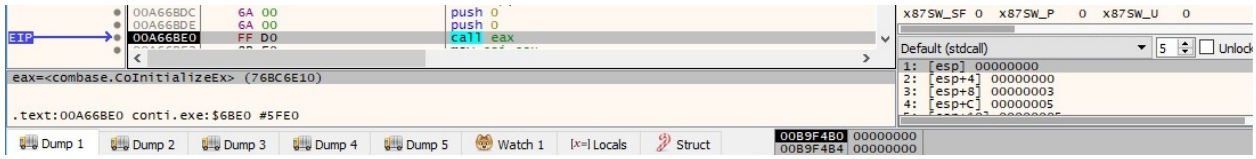


Figure 13

The CoInitializeSecurity API is used to register security and set the default security values for the current process (0x3 = **RPC_C_IMP_LEVEL_IMPERSONATE**):

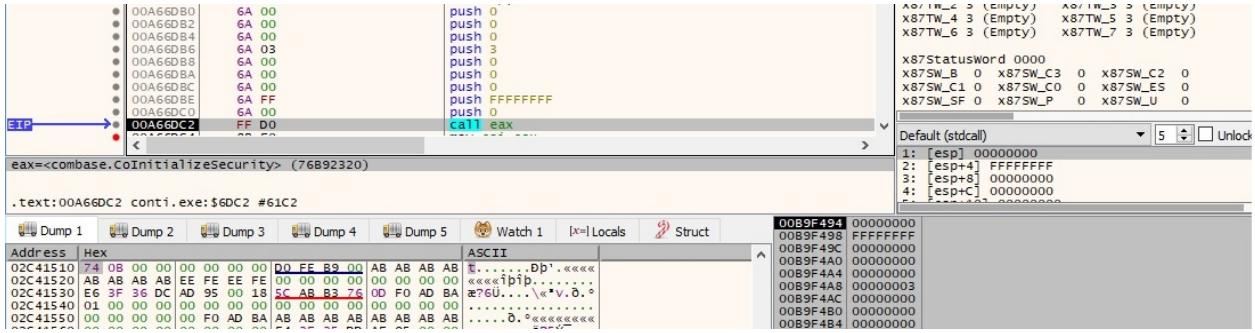


Figure 14

The malware uses COM objects and wmic in order to delete the volume shadow copies on the system. It calls the CoCreateInstance function with the CLSID {4590F812-1D3A-11D0-891F-00AA004B2E24}, which creates an IWbemLocator object:

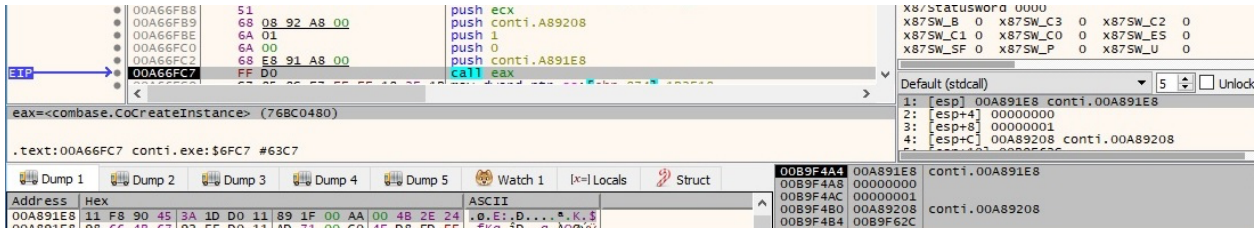


Figure 15

A new IWbemContext object is created with the CLSID {674B6698-EE92-11D0-AD71-00C04FD8FDFF}:

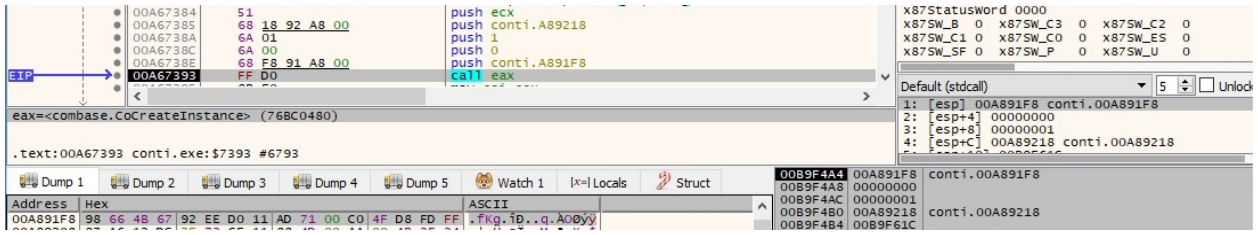


Figure 16

The ConnectServer method is utilized to connect to the "ROOT\CIMV2" namespace:

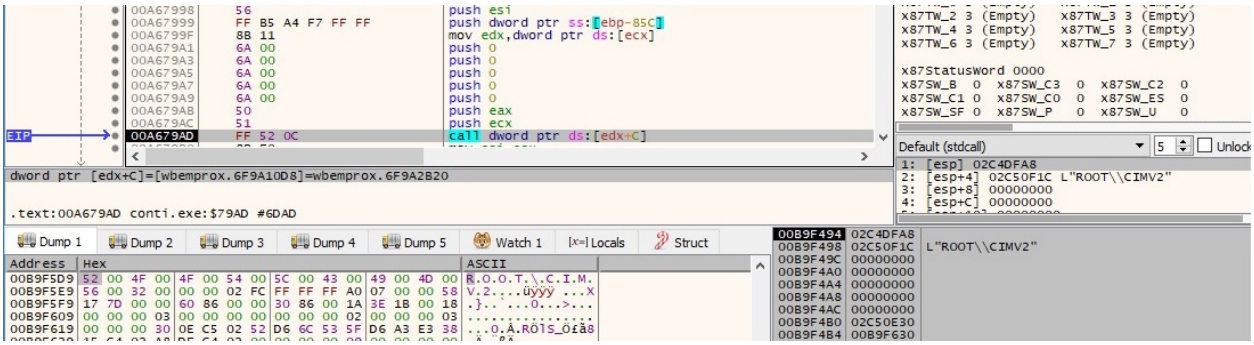


Figure 17

The binary sets the authentication information that is used to make calls on a proxy via a CoSetProxyBlanket API call (0xA = **RPC_C_AUTHN_WINNT**, 0x3 = **RPC_C_AUTHN_LEVEL_CALL**, 0x3 = **RPC_C_IMP_LEVEL_IMPERSONATE**):

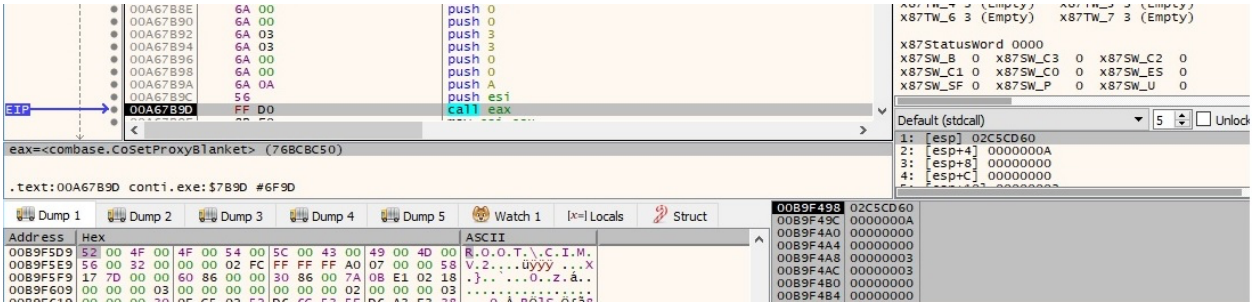


Figure 18

The following WQL (SQL for WMI) query is executed by the ransomware:

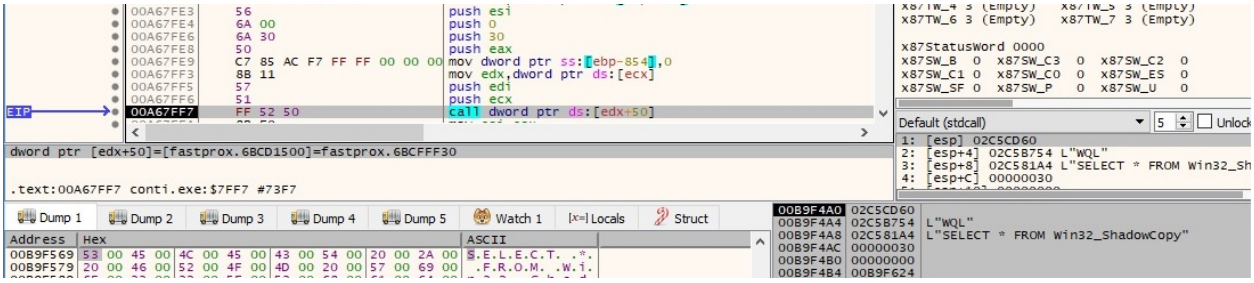


Figure 19

For each volume shadow copy, the binary extracts its ID using the Get method:

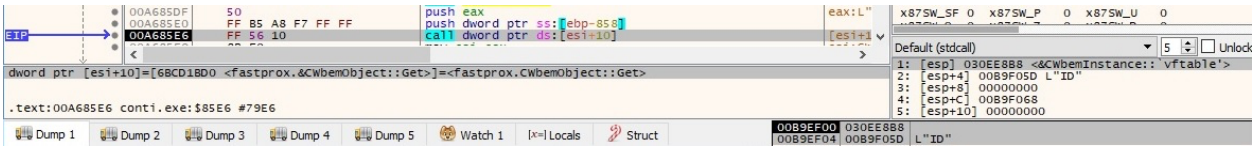


Figure 20

The following string that contains a process name with parameters is decrypted:

Address	Hex	ASCII
00B9EF29	63 00 6D 00 64 00 2E 00 65 00 78 00 65 00 20 00	c.m.d...e.x.e. .
00B9EF39	2F 00 63 00 20 00 43 00 3A 00 5C 00 57 00 69 00	/.c. .C.:.\.W.i.
00B9EF49	6E 00 64 00 6F 00 77 00 73 00 5C 00 53 00 79 00	n.d.o.w.s.\.S.y.
00B9EF59	73 00 74 00 65 00 6D 00 33 00 32 00 5C 00 77 00	s.t.e.m.3.2.\.w.
00B9EF69	62 00 65 00 6D 00 5C 00 57 00 4D 00 49 00 43 00	b.e.m.\.W.M.I.C.
00B9EF79	2E 00 65 00 78 00 65 00 20 00 73 00 68 00 61 00	..e.x.e. .s.h.a.
00B9EF89	64 00 6F 00 77 00 63 00 6F 00 70 00 79 00 20 00	d.o.w.c.o.p.y. .
00B9EF99	77 00 68 00 65 00 72 00 65 00 20 00 22 00 49 00	w.h.e.r.e. .".I.
00B9EFA9	44 00 3D 00 27 00 25 00 73 00 27 00 22 00 20 00	D.=.'.%.s.'.". .
00B9EFB9	64 00 65 00 6C 00 65 00 74 00 65 00 00 00 00 00	d.e.l.e.t.e.....

Figure 21

Wow64DisableWow64FsRedirection is utilized to disable file system redirection for the current thread:

Figure 22

The executable deletes each volume shadow copy that corresponds to the ID extracted above using the CreateProcessW API (0x08000000 = **CREATE_NO_WINDOW**):

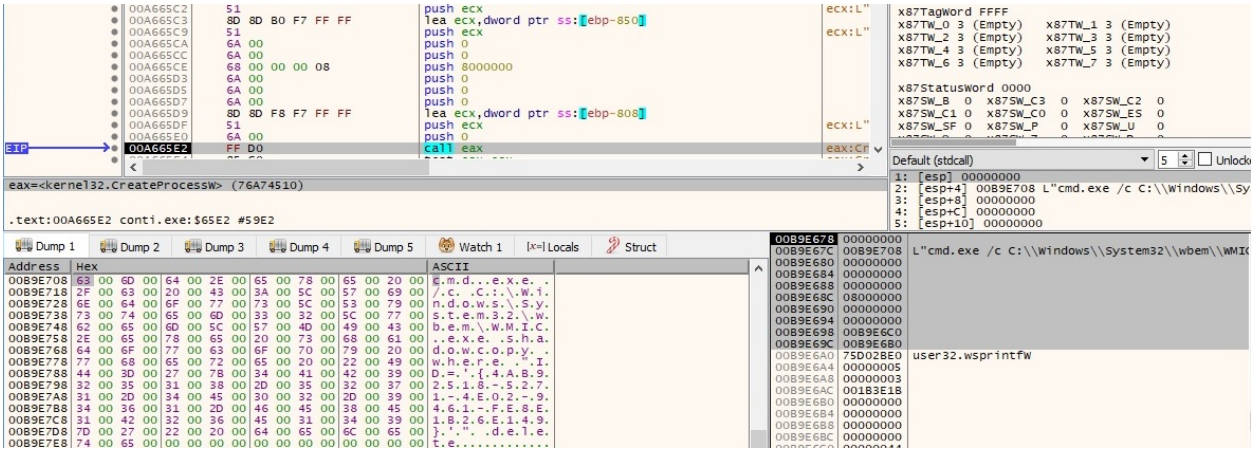


Figure 23

The malware restores file system redirection by calling the Wow64RevertWow64FsRedirection routine:

Figure 24

The valid drives on the system are retrieved by calling the GetLogicalDriveStringsW function:

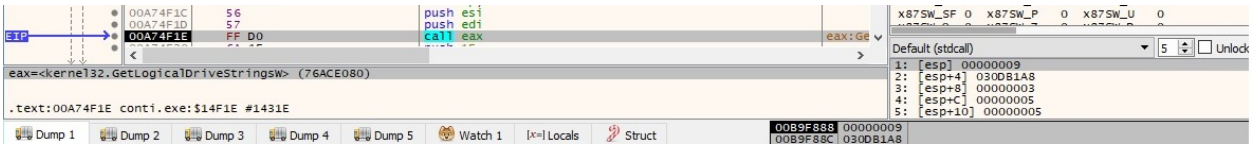


Figure 25

There is a function call to WSAStartup that initiates the use of the Winsock DLL:

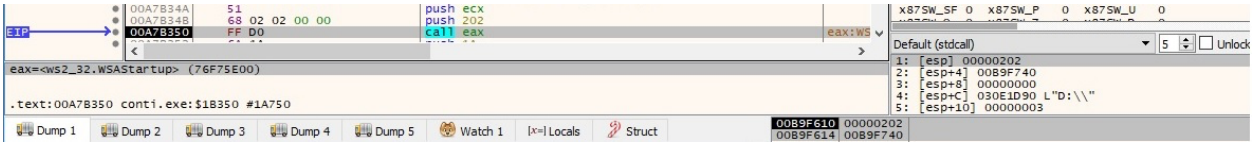


Figure 26

A new socket is created by the process (0x2 = **AF_INET**, 0x1 = **SOCK_STREAM**, 0x6 = **IPPROTO_TCP**):

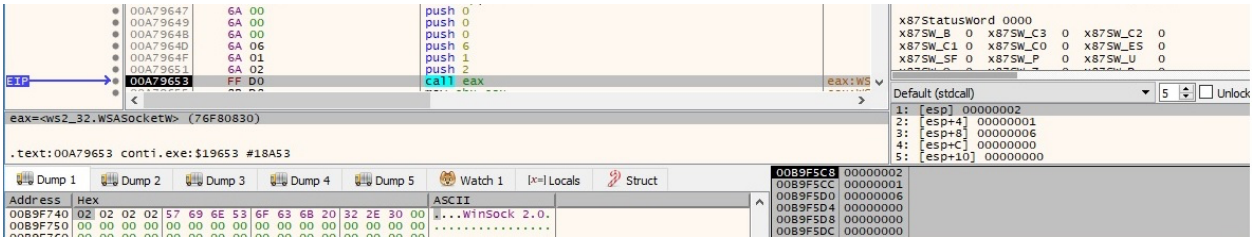


Figure 27

The malicious process calls the WSAIoctl function with the **SIO_GET_EXTENSION_FUNCTION_POINTER** command code in order to invoke an extension function, as shown in figure 28:

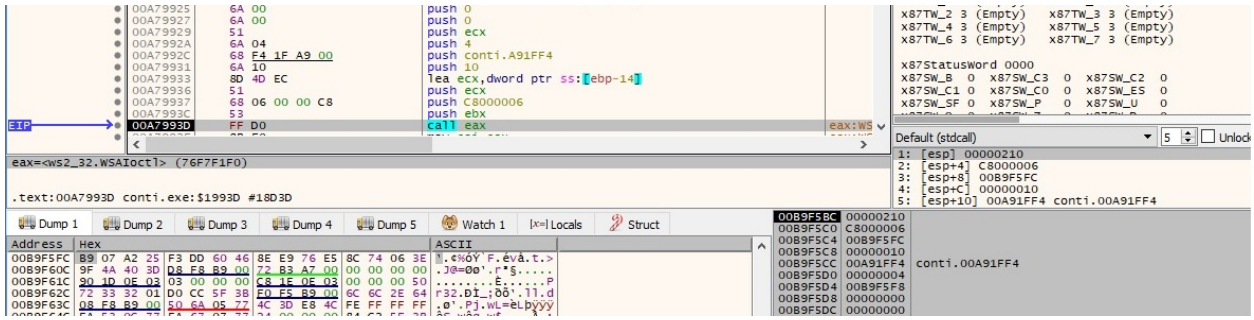


Figure 28

The gethostname routine is utilized to retrieve the host name for the local computer:

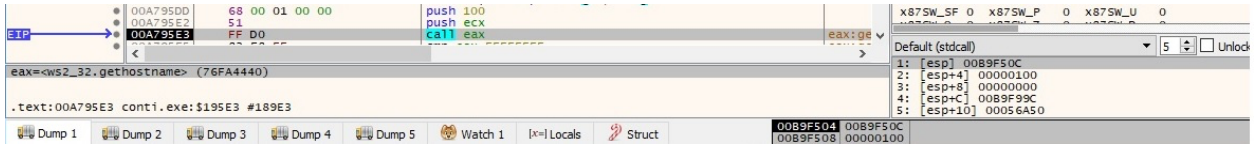


Figure 29

The malicious file retrieves host information that corresponds to the host extracted above:

Figure 30

The CreateIoCompletionPort API is used to create an I/O (input/output) completion port that is not yet associated with a file handle (0xFFFFFFFF = **INVALID_HANDLE_VALUE**):

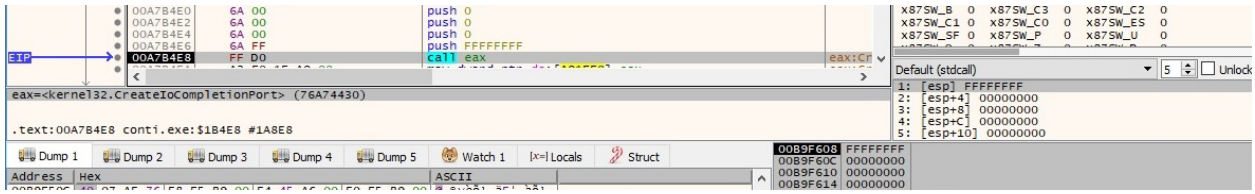


Figure 31

The ARP table is extracted by calling the GetIpNetTable routine, and the result is stored in a MIB_IPNETTABLE structure:

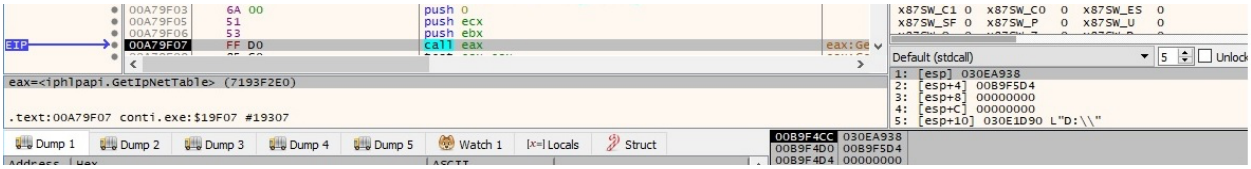


Figure 32

Each IP address extracted above is converted into a string (dotted-decimal format):

Figure 33

The malware is only interested in local IP addresses because it compares every IP address with the prefixes "172.", "192.168.", "10." and "169.". The binary creates 2 new threads via a function call to CreateThread:

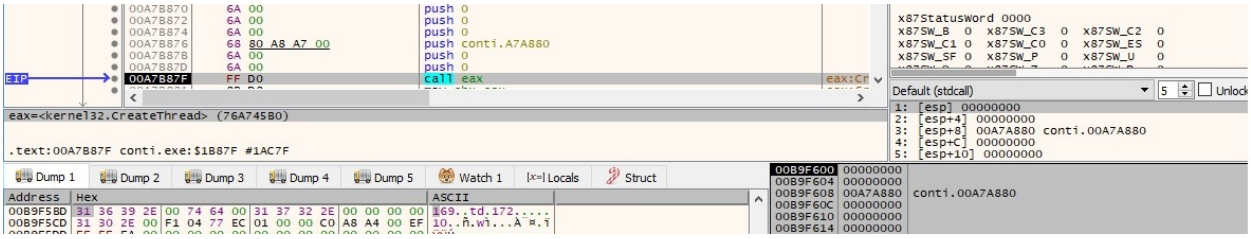


Figure 34

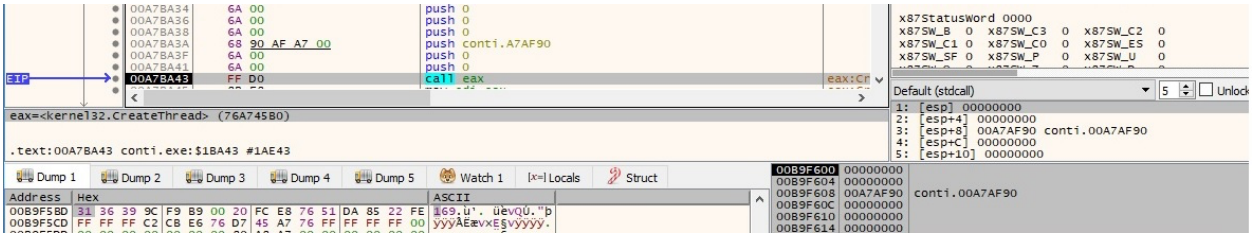


Figure 35

PostQueuedCompletionStatus is utilized to send an I/O completion packet to the completion port created earlier (**dwCompletionKey** = 0x1):

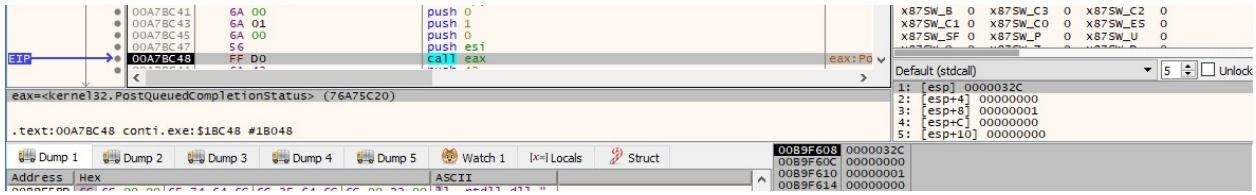


Figure 36

THREAD ACTIVITY – SUB_A7AF90 FUNCTION

The file creates a queue for timers (which are objects that allow the user to specify a function that will be called at a particular time):

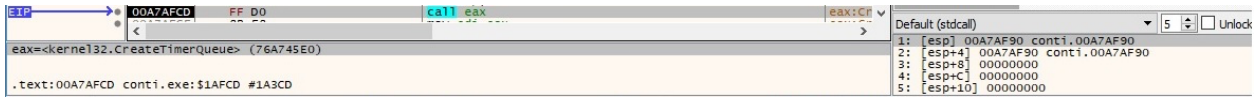


Figure 37

The ransomware attempts to extract the I/O completion packet from the I/O completion port (sent by the main thread) by calling the GetQueuedCompletionStatus routine:

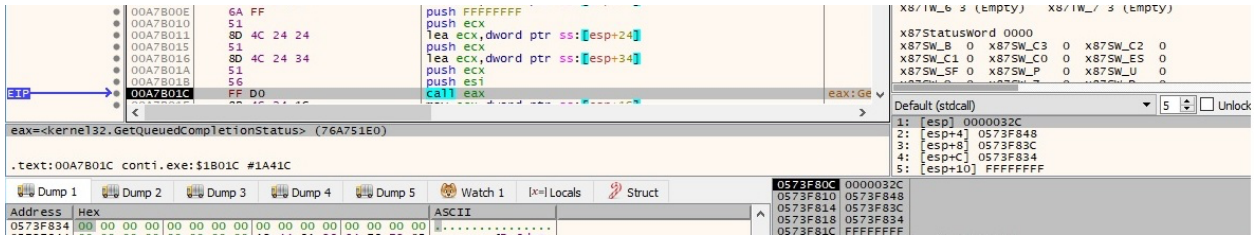


Figure 38

A new socket is created by calling the WSASocketW API (0x2 = **AF_INET**, 0x1 = **SOCK_STREAM**, 0x6 = **IPPROTO_TCP**, 0x1 = **WSA_FLAG_OVERLAPPED**):

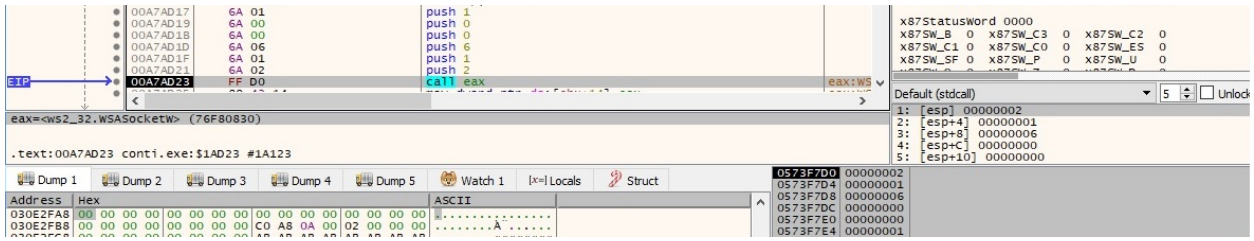


Figure 39

The bind routine associates the local address with the above socket:

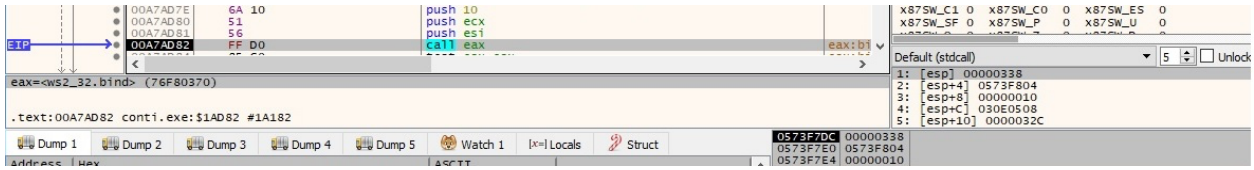


Figure 40

CreateIoCompletionPort is utilized to associate the socket created above with the I/O completion port. After this operation is complete, the process can receive notifications of the completion of I/O operations involving the socket handle (**CompletionKey** = 0x2):

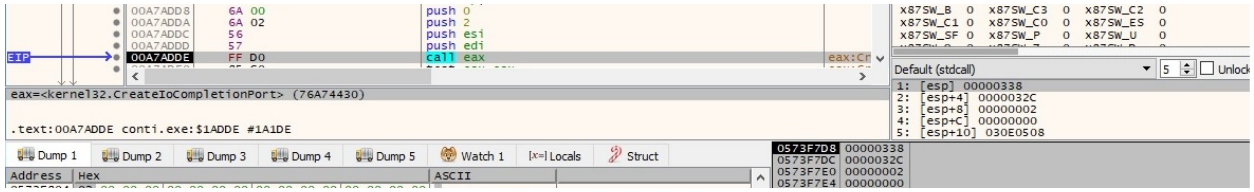


Figure 41

The binary converts a port number (445) from network byte order to host byte order:

Figure 42

The malware tries to connect to different IP addresses on port 445 (192.168.10.x and 192.168.164.x) using the LPFN_CONNECTEX function, as described below:

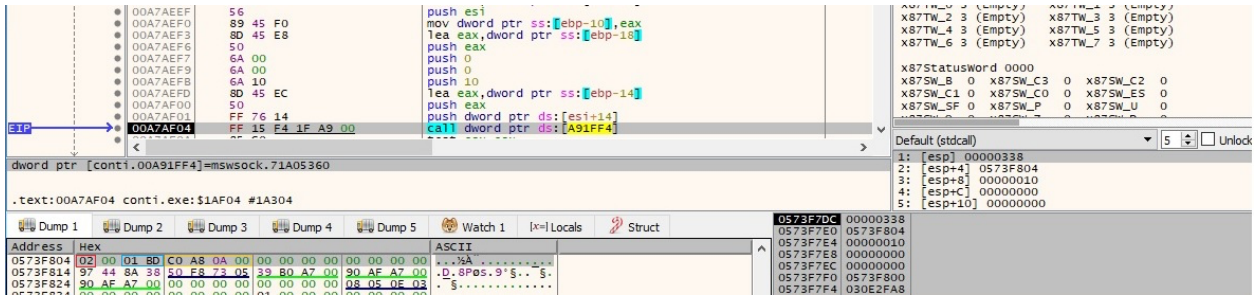


Figure 43

The CreateTimerQueueTimer routine is used to create a timer-queue timer, which expires at a specific time (0x7530 = 30000ms = 30 seconds) and then a callback function is called:

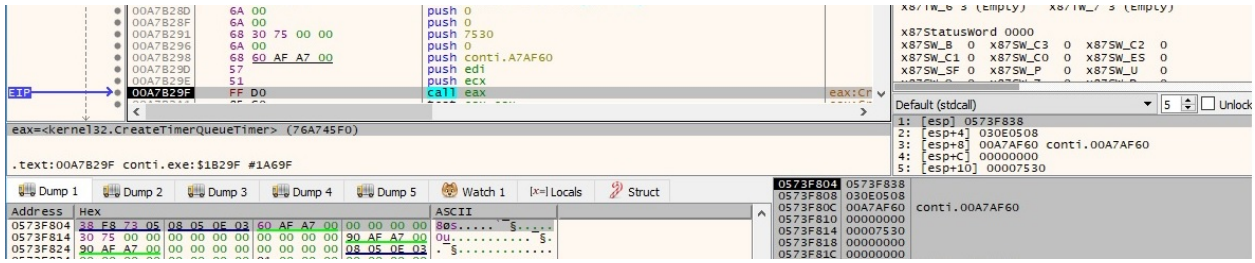


Figure 44

The setsockopt API is utilized to set the **SO_UPDATE_CONNECT_CONTEXT** option, which updates the properties of the socket after a connection is established (0xFFFF = **SOL_SOCKET**, 0x7010 = **SO_UPDATE_CONNECT_CONTEXT**):

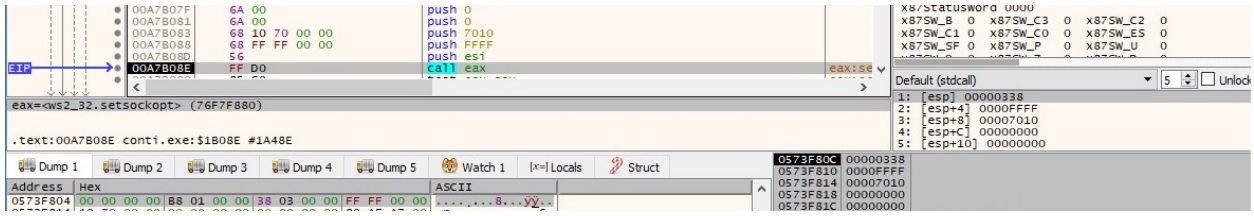


Figure 45

The file retrieves the **SO_CONNECT_TIME** option, which represents the number of seconds a socket was connected (0xFFFF = **SOL_SOCKET**, 0x700C = **SO_CONNECT_TIME**):

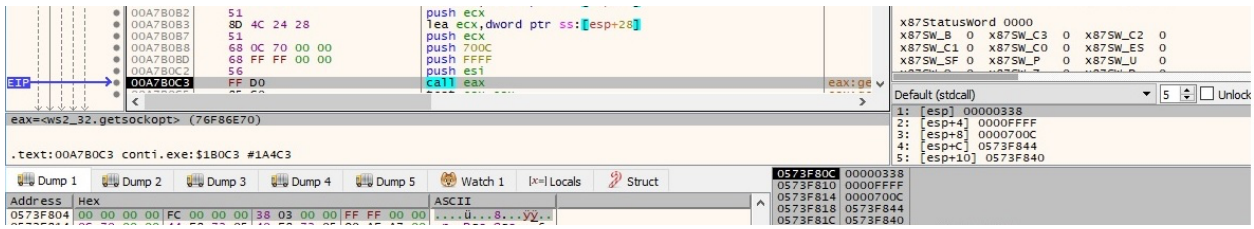


Figure 46

Whether the sample has successfully established a connection to a particular IP address, then it calls the WSAAddressToStringW routine to convert the components of that sockaddr structure into a human-readable string:

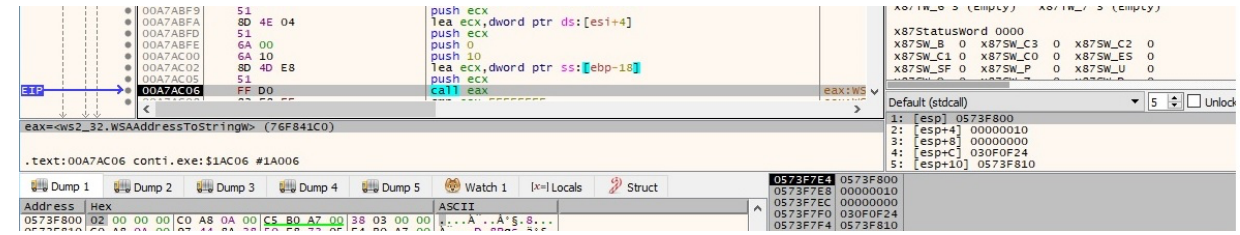


Figure 47

PostQueuedCompletionStatus is utilized to send an I/O completion packet to the completion port created before (**dwCompletionKey** = 0x3):

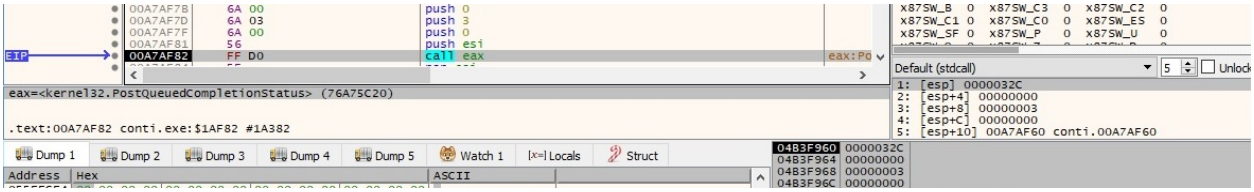


Figure 48

The binary shuts down send operations for the socket (0x1 = **SD_SEND**):

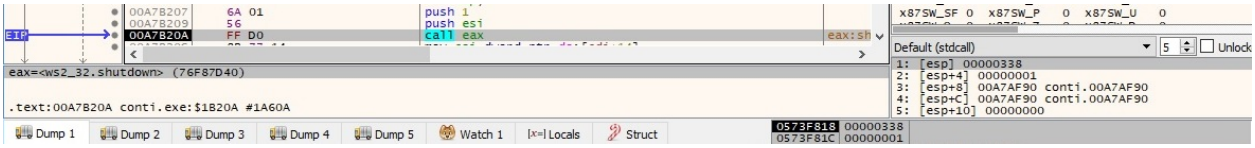


Figure 49

THREAD ACTIVITY – SUB_A7A880 FUNCTION

The NetShareEnum function is utilized to retrieve information about the network shares available on other computers:

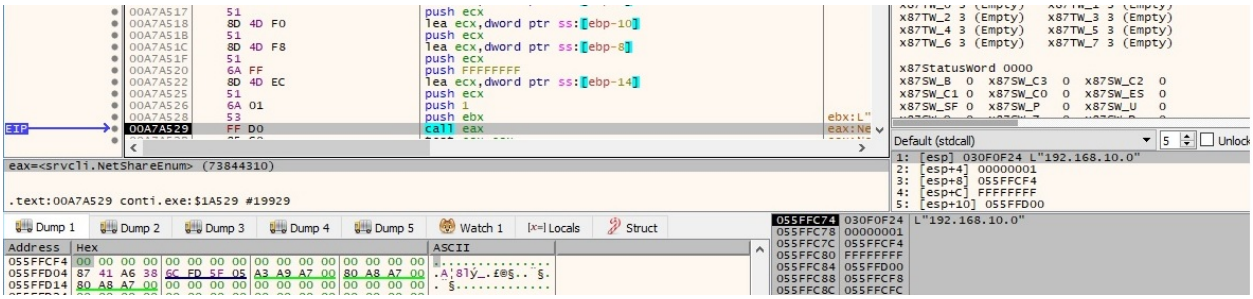


Figure 50

Some strings that will be written in the log file (if logging mode is enabled) are also decrypted using custom algorithms:

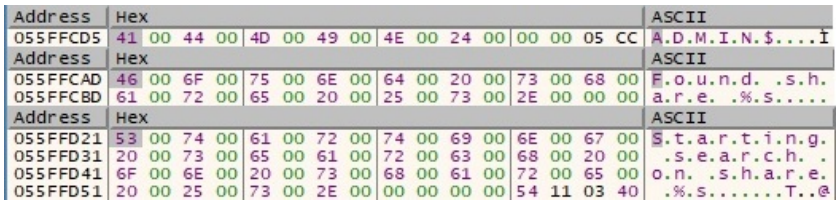


Figure 51

The "ADMIN$" share will not be targeted by the malware (the others will be encrypted):

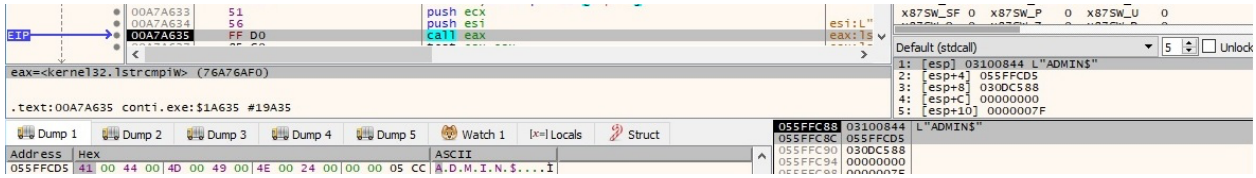


Figure 52

THREAD ACTIVITY – SUB_A7C7D0 FUNCTION

CryptAcquireContextA is used to obtain a handle to a key container within a CSP (0x18 = **PROV_RSA_AES**, 0xF0000000 = **CRYPT_VERIFYCONTEXT**):

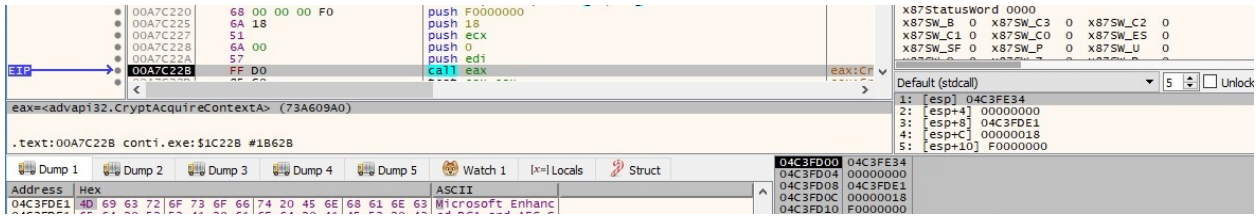


Figure 53

An RSA public key is imported via a CryptImportKey function call:

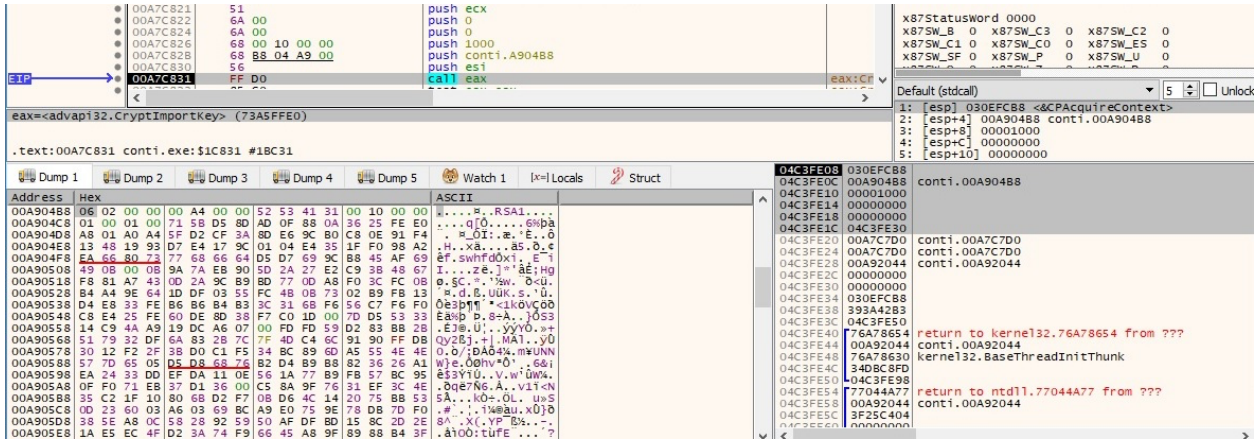


Figure 54

The process creates a file called "readme.txt" in every folder that it encrypts (0x40000000 = **GENERIC_WRITE**, 0x2 = **CREATE_ALWAYS**):

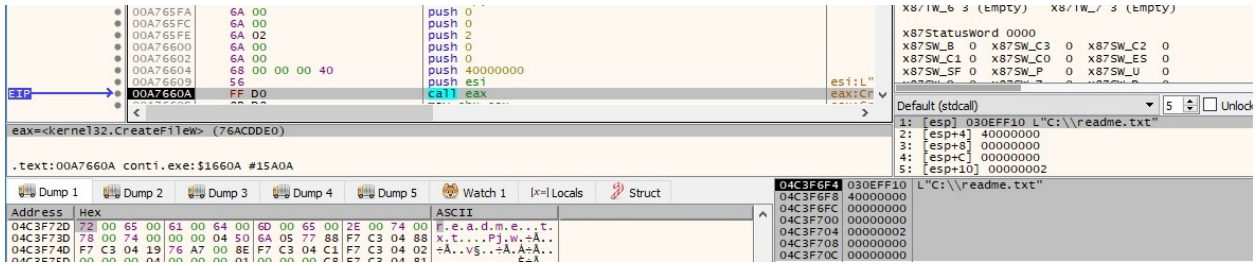


Figure 55

The following 4-byte values suggest that the encryption algorithm is a ChaCha cipher (Ref. https://arxiv.org/pdf/1907.11941.pdf):

```
.text:00A766EA push    esi
.text:00A766EB push    edi
.text:00A766EC mov     [ebp+var_80], 61707865h
.text:00A766F3 mov     [ebp+var_7C], 3320646Eh
.text:00A766FA mov     [ebp+var_78], 79622D32h
.text:00A76701 mov     [ebp+var_74], 6B206574h
.text:00A76708 mov     [ebp+var_44], eax
.text:00A7670B call    sub_A65AF0
```

Figure 56

The encrypted content of the ransom note is decrypted using the ChaCha algorithm, and the file is populated by calling the WriteFile routine, as highlighted in figure 57.

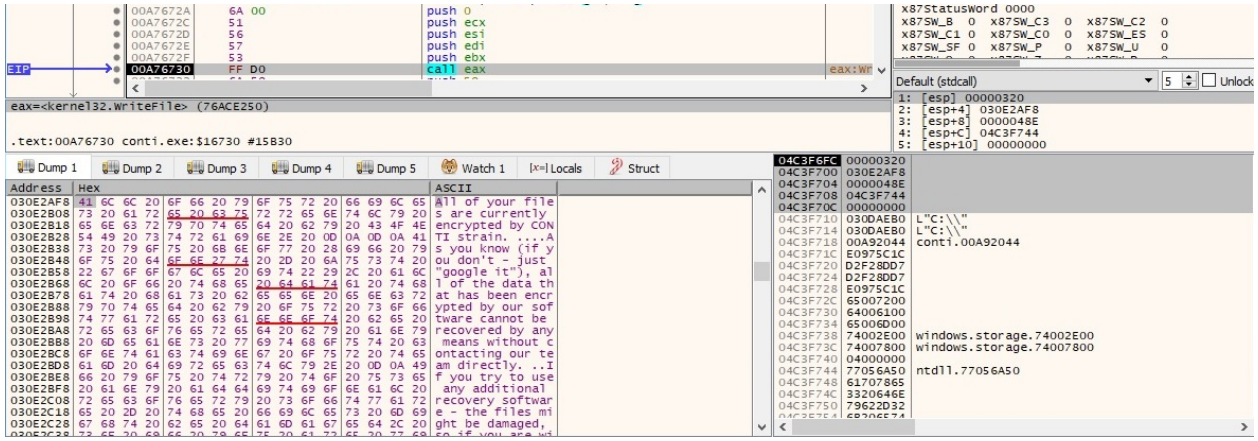


Figure 57

The files are enumerated in the targeted directory using the FindFirstFileW and FindNextFileW APIs:

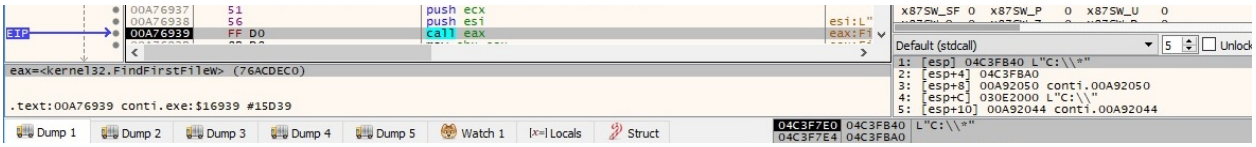


Figure 58

Figure 59

There is a comparison between the directory name and a list of directories that will be skipped by the ransomware:

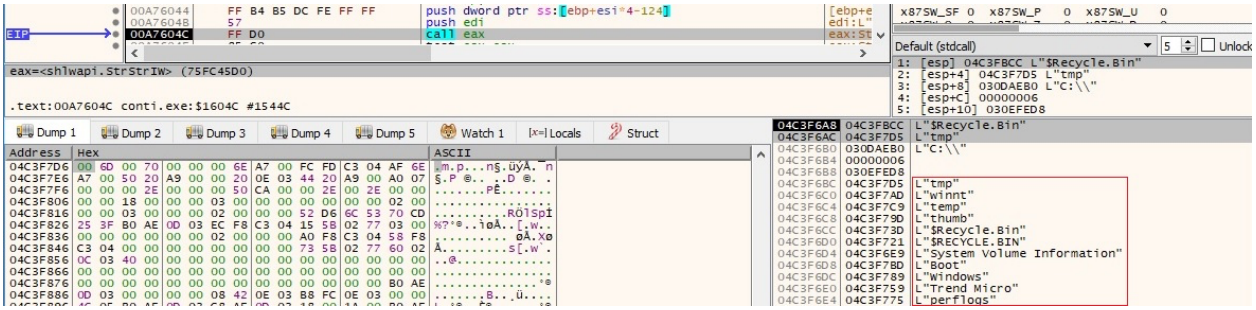


Figure 60

The PathIsDirectoryW routine is utilized to determine whether a path is a valid directory:

Figure 61

The following files/files extensions will also be skipped by Conti:

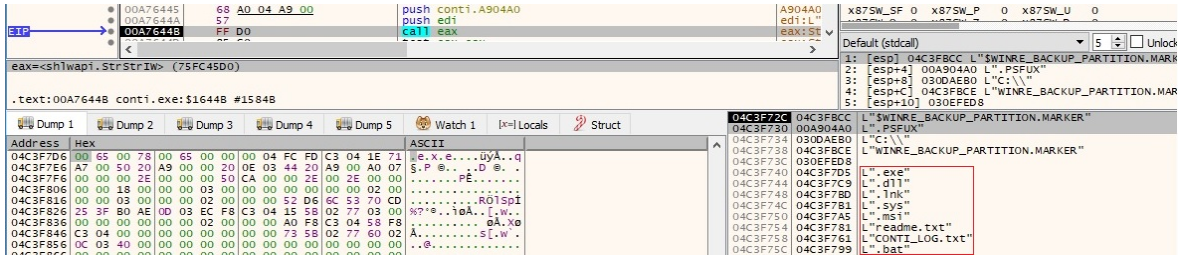


Figure 62

The sample descrypts the following DLL names: OleAut32.dll, Rstrtmgr.dll, Iphlpapi.dll, Netapi32.dll, Advapi32.dll, Kernel32.dll, Shell32.dll, Shlwapi.dll, ws2_32.dll, User32.dll, ntdll.dll, Ole32.dll. The GetModuleHandleA function is utilized to retrieve a handle for these DLLs. The malware generates 32 random bytes by calling the CryptGenRandom routine (this will be used as the ChaCha key):

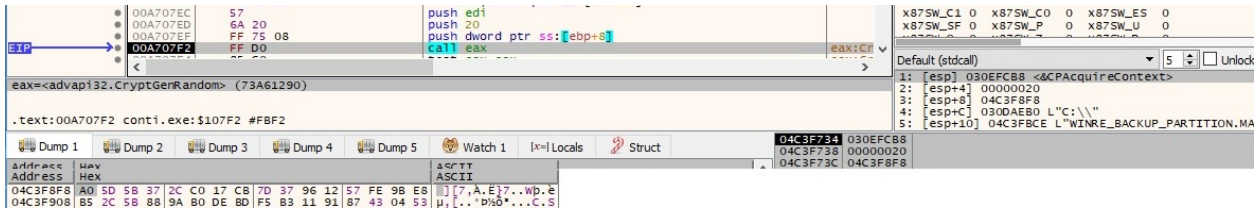


Figure 63

There is also a call to CryptGenRandom that generates 8 random bytes, which will be used as the ChaCha8 nonce (this is the moment when we can tell for sure that the encryption algorithm for files is ChaCha8):

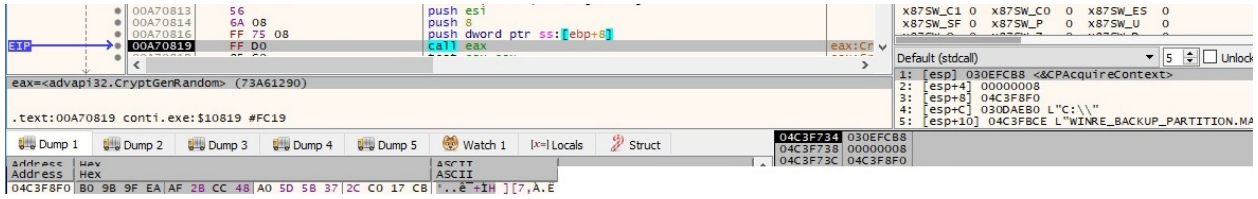


Figure 64

The ChaCha8 key and nonce are encrypted using the RSA public key:

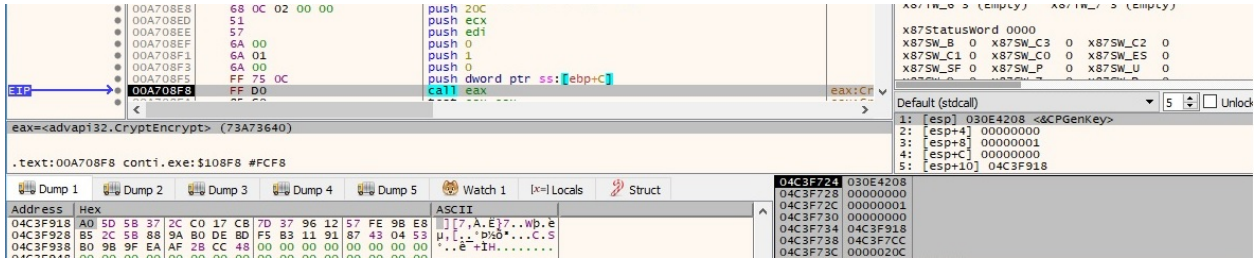


Figure 65

The ransomware retrieves file system attributes for the targeted file:

Figure 66

The CreateFileW API is utilized to open the targeted file (0xC0000000 = **GENERIC_READ** | **GENERIC_WRITE**, 0x3 = **OPEN_EXISTING**):

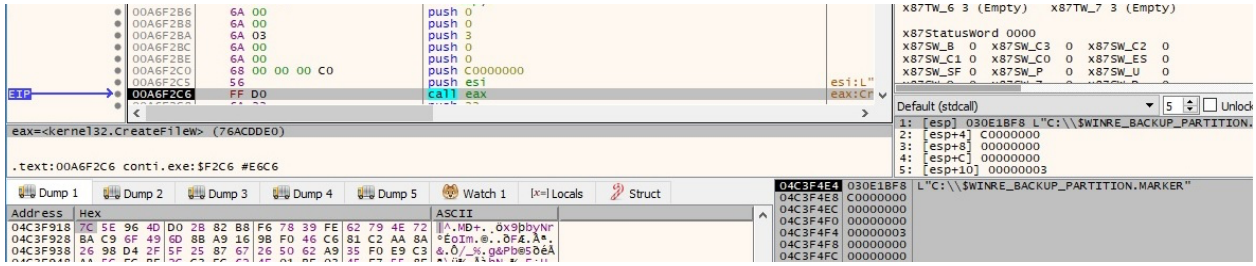


Figure 67

The malware comes with two hard-coded lists of file extensions that will be encrypted. It's important to mention that if the file extension doesn't belong to these lists, it will be partially encrypted using a different execution flow that will be explained later (the full lists are available in the Appendix):

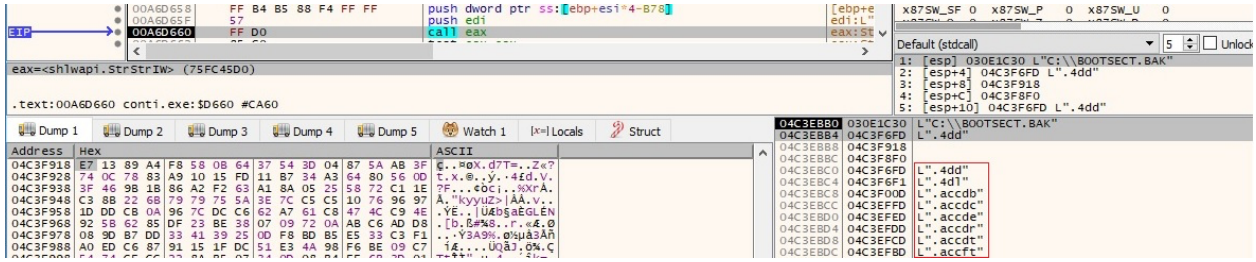


Figure 68

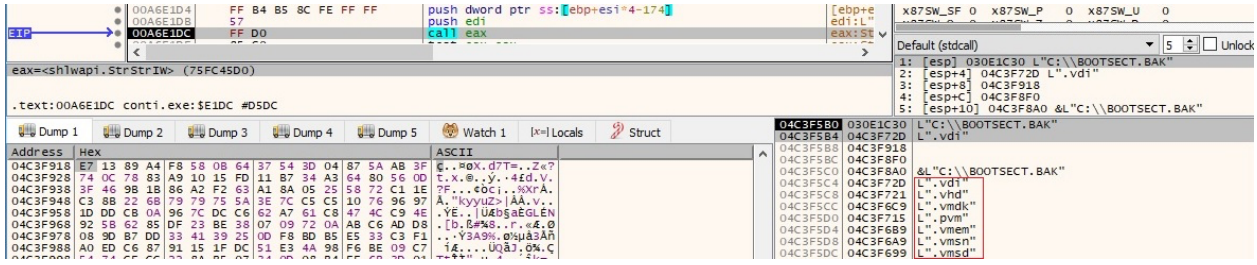


Figure 69

The process writes the encrypted ChaCha8 key and nonce to the encrypted file:

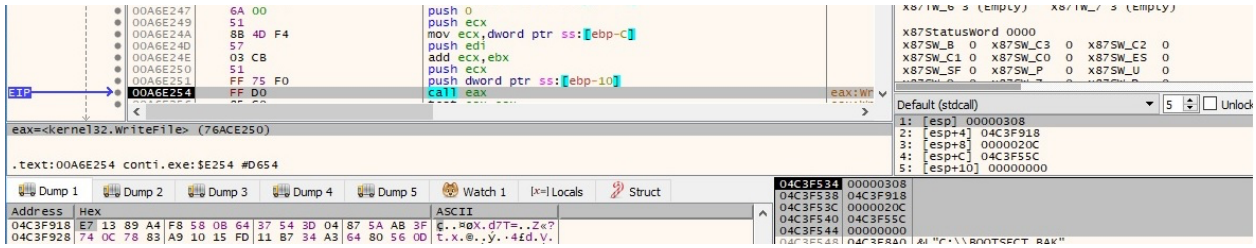


Figure 70

There are 3 different cases depending on the file size: small files (< 1MB), medium files (between 1MB and 5MB), and large files (> 5MB). In the case of medium and large files, there exist 2 sub-cases depending on the file extension (if it belongs to the targeted lists or not). The following 10-byte buffer that contains a marker (0x24) and the file size (0x2000) is appended to the encrypted file:

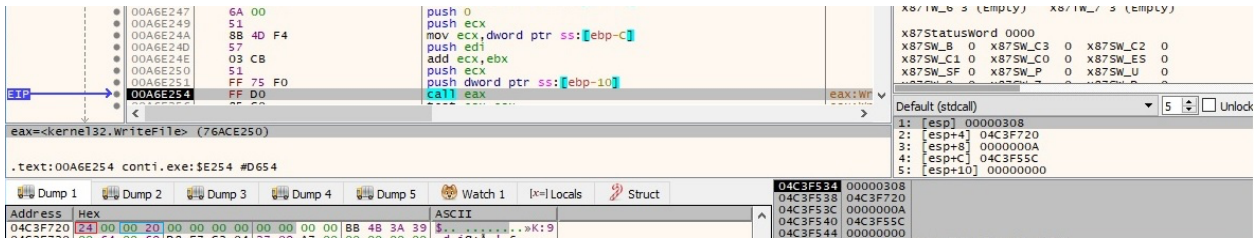


Figure 71

The binary reads the file content using the ReadFile function:

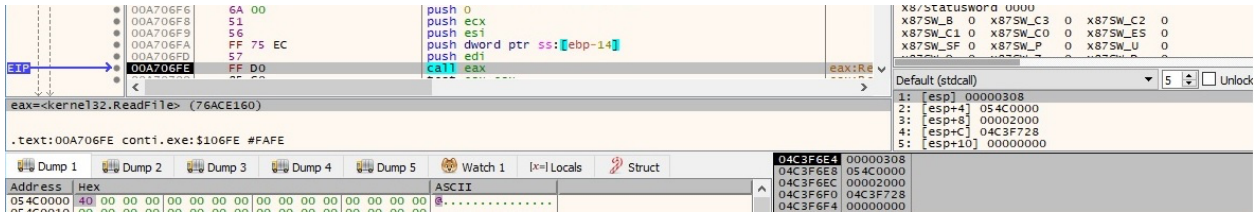


Figure 72

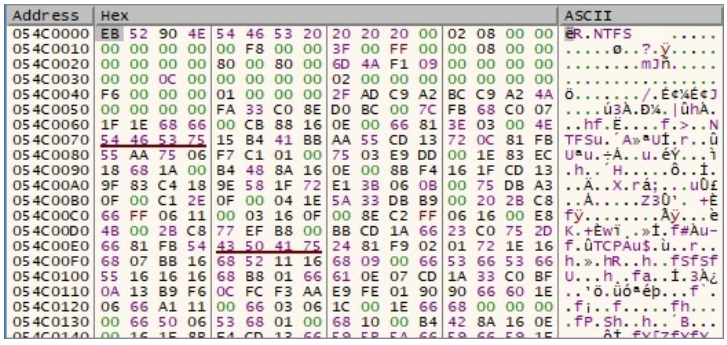


Figure 73

The content is encrypted using the ChaCha8 algorithm implemented by Conti:

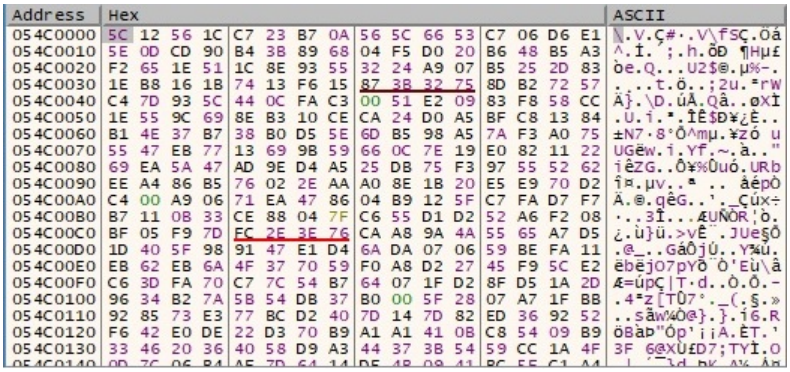


Figure 74

A snippet of the ChaCha8 algorithm developed by the ransomware is presented in figure 75.

```
.text:00A65C83
.text:00A65C83 loc_A65C83:
.text:00A65C83 add     edi, [ebp+var_60]
.text:00A65C86 mov     eax, [ebp+var_58]
.text:00A65C89 xor     ebx, edi
.text:00A65C8B add     ecx, [ebp+var_78]
.text:00A65C8E rol     ebx, 10h
.text:00A65C91 xor     edx, ecx
.text:00A65C93 add     eax, ebx
.text:00A65C95 mov     [ebp+var_5C], edi
.text:00A65C98 mov     [ebp+var_58], eax
.text:00A65C9B xor     eax, [ebp+var_60]
.text:00A65C9E rol     eax, 0Ch
.text:00A65CA1 add     edi, eax
.text:00A65CA3 rol     edx, 10h
.text:00A65CA6 xor     ebx, edi
.text:00A65CA8 mov     [ebp+var_5C], edi
.text:00A65CAB mov     edi, [ebp+var_58]
.text:00A65CAE rol     ebx, 8
.text:00A65CB1 add     edi, ebx
.text:00A65CB3 mov     [ebp+var_58], edi
.text:00A65CB6 xor     edi, eax
.text:00A65CB8 mov     eax, [ebp+var_4C]
.text:00A65CBB add     eax, edx
.text:00A65CBD rol     edi, 7
.text:00A65CC0 mov     [ebp+var_4C], eax
.text:00A65CC3 xor     eax, [ebp+var_78]
.text:00A65CC6 rol     eax, 0Ch
.text:00A65CC9 add     ecx, eax
.text:00A65CCB xor     edx, ecx
.text:00A65CCD mov     [ebp+var_84], ecx
.text:00A65CD3 mov     ecx, [ebp+var_4C]
.text:00A65CD6 rol     edx, 8
.text:00A65CD9 add     ecx, edx
.text:00A65CDB mov     [ebp+var_60], edx
.text:00A65CDE mov     edx, [ebp+var_64]
.text:00A65CE1 add     edx, [ebp+var_7C]
.text:00A65CE4 xor     esi, edx
.text:00A65CE6 mov     [ebp+var_4C], ecx
.text:00A65CE9 xor     ecx, eax
.text:00A65CEB rol     esi, 10h
.text:00A65CEE mov     eax, [ebp+var_48]
```

Figure 75

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

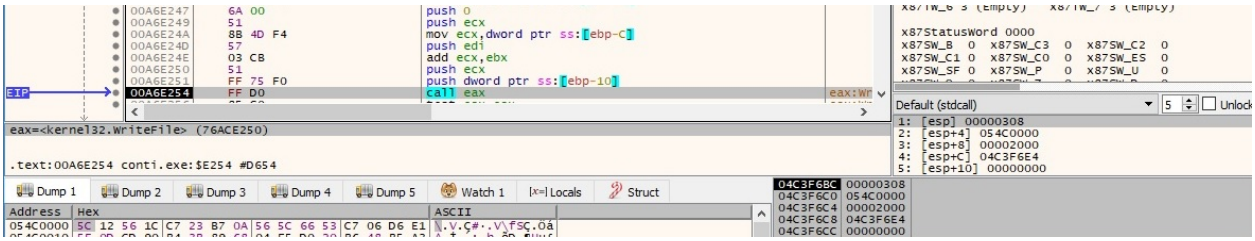


Figure 76

The ".PSFUX" extension is added to the file name:

Figure 77

The ransom note that is created in every encrypted directory is displayed below:

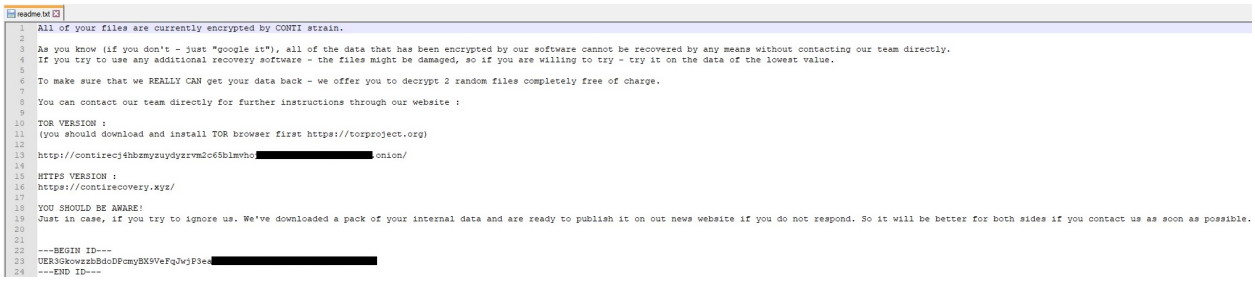


Figure 78

An example of an encrypted file (file size < 1MB) is highlighted in the next 2 pictures:

BOOTSECT.BAK.PSFUX

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	5C	12	56	1C	C7	23	B7	0A	56	5C	66	53	C7	06	D6	E1
00000010	5E	0D	CD	90	B4	3B	89	68	04	F5	D0	20	B6	48	B5	A3
00000020	F2	65	1E	51	1C	8E	93	55	32	24	A9	07	B5	25	2D	83
00000030	1E	B8	16	1B	74	13	F6	15	87	3B	32	75	8D	B2	72	57
00000040	C4	7D	93	5C	44	0C	FA	C3	00	51	E2	09	83	F8	58	CC
00000050	1E	55	9C	69	8E	B3	10	CE	CA	24	D0	A5	BF	C8	13	84
00000060	B1	4E	37	B7	38	B0	D5	5E	6D	B5	98	A5	7A	F3	A0	75
00000070	55	47	EB	77	13	69	9B	59	66	0C	7E	19	E0	82	11	22
00000080	69	EA	5A	47	AD	9E	D4	A5	25	DB	75	F3	97	55	52	62
00000090	EE	A4	86	B5	76	02	2E	AA	A0	8E	1B	20	E5	E9	70	D2
000000A0	C4	00	A9	06	71	EA	47	86	04	B9	12	5F	C7	FA	D7	F7
000000B0	B7	11	0B	33	CE	88	04	7F	C6	55	D1	D2	52	A6	F2	08
000000C0	BF	05	F9	7D	FC	2E	3E	76	CA	A8	9A	4A	55	65	A7	D5
000000D0	1D	40	5F	98	91	47	E1	D4	6A	DA	07	06	59	BE	FA	11
000000E0	EB	62	EB	6A	4F	37	70	59	F0	A8	D2	27	45	F9	5C	E2
000000F0	C6	3D	FA	70	C7	7C	54	B7	64	07	1F	D2	8F	D5	1A	2D
00000100	96	34	B2	7A	5B	54	DB	37	B0	00	5F	28	07	A7	1F	BB
00000110	92	85	73	E3	77	BC	D2	40	7D	14	7D	82	ED	36	92	52
00000120	F6	42	E0	DE	22	D3	70	B9	A1	A1	41	0B	C8	54	09	B9
00000130	33	46	20	36	40	58	D9	A3	44	37	3B	54	59	CC	1A	4F
00000140	0D	7C	06	B4	AF	7D	64	14	DE	4B	09	41	BC	5F	C1	A4
00000150	C8	DC	EA	D2	5A	3F	10	32	56	79	D4	F2	ED	9E	E5	AB
00000160	A7	6B	1B	1B	0F	CA	E5	78	57	DF	2E	19	FC	1E	90	37
00000170	F3	59	D6	DF	F5	48	7E	95	14	7C	16	CA	82	65	79	AD
00000180	C5	98	52	DC	F5	C2	C4	45	1B	C2	88	49	09	1D	A4	53
00000190	E0	F2	B2	C8	DD	51	75	4E	26	C2	2B	D2	A1	F3	F6	64

Figure 79

BOOTSECT.BAK.PSFUX

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00002000	E7	13	89	A4	F8	58	0B	64	37	54	3D	04	87	5A	AB	3F
00002010	74	0C	78	83	A9	10	15	FD	11	B7	34	A3	64	80	56	0D
00002020	3F	46	9B	1B	86	A2	F2	63	A1	8A	05	25	58	72	C1	1E
00002030	C3	8B	22	6B	79	79	75	5A	3E	7C	C5	C5	10	76	96	97
00002040	1D	DD	CB	0A	96	7C	DC	C6	62	A7	61	C8	47	4C	C9	4E
00002050	92	5B	62	85	DF	23	BE	38	07	09	72	0A	AB	C6	AD	D8
00002060	08	9D	B7	DD	33	41	39	25	0D	F8	BD	B5	E5	33	C3	F1
00002070	A0	ED	C6	87	91	15	1F	DC	51	E3	4A	98	F6	BE	09	C7
00002080	54	74	CE	CC	22	8A	B5	07	34	0D	08	B4	EE	6B	3D	01
00002090	7D	1D	F2	D4	7A	FF	F1	50	EA	61	BF	FB	D1	BE	EC	63
000020A0	2C	4B	F8	B5	96	EB	34	0E	42	EC	EB	38	D8	D4	98	6A
000020B0	02	20	AC	08	59	C8	21	60	47	98	8E	5B	7E	AE	B6	BF
000020C0	1C	91	9A	98	1F	50	B1	0D	97	EC	51	FE	E2	1D	9A	D9
000020D0	EA	CC	D3	F4	28	B3	7A	53	23	C1	C9	9D	49	BB	51	37
000020E0	0E	65	BF	E1	90	E9	1F	5F	99	68	C8	A1	86	54	AB	E6
000020F0	AA	E8	1D	F9	F4	77	19	0E	C1	24	0A	89	E5	14	BD	09
00002100	15	5D	CA	E9	F2	0F	F5	58	58	D4	4F	96	C5	30	94	BF
00002110	AE	48	4A	8E	50	7C	D6	60	3E	B0	8D	38	94	4B	DA	CD
00002120	9F	74	A5	28	FE	B1	14	DB	E3	D9	3A	A8	C1	34	BF	A9
00002130	12	DC	57	C7	95	8E	E5	8A	E5	9D	7C	63	CA	9F	DE	D6
00002140	66	21	09	38	C1	CF	CA	66	1D	9A	64	BA	3E	13	0A	51
00002150	6D	40	2E	F7	BB	03	0A	85	40	A6	DF	33	9B	60	C6	57
00002160	58	0D	F9	BF	F5	4B	94	4B	F6	64	58	02	9B	52	EA	CD
00002170	22	59	40	F2	22	14	A0	0F	F2	B3	B3	B5	A1	26	67	BB
00002180	11	DE	D0	A7	D2	96	FC	EE	2D	66	77	30	C4	0E	5F	A8
00002190	ED	DB	84	69	5E	75	3A	08	9A	E4	15	50	B5	79	AA	E5
000021A0	91	0E	AE	54	D3	4E	0F	DE	1E	19	91	3E	4C	C7	E8	30
000021B0	BA	86	60	CB	68	03	D0	F4	6F	B8	E3	29	EF	7D	3F	4D
000021C0	98	8D	D6	25	DC	15	FB	0C	22	8E	C0	DE	27	80	32	3A
000021D0	F9	7A	1A	E0	3C	47	B8	54	80	A9	87	8F	DA	C8	39	84
000021E0	5A	11	7E	65	19	97	90	0F	24	C5	38	FC	EE	CF	98	36
000021F0	B9	F0	71	BE	AD	64	16	13	60	47	E8	B9	76	9B	DC	8A
00002200	00	00	00	00	00	00	00	00	00	00	00	00	24	00	00	20
00002210	00	00	00	00	00	00										

Figure 80

Whether the file size is between 1MB and 5MB and the extension is not in the targeted lists, the ransomware only encrypts the first MB of the file, and the encrypted file has the following structure:

test.txt.PSFUX

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00100000	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
00100010	F5	39	A3	4F	E1	25	32	0E	33	BF	FD	C9	3E	FA	C9	09	õ9£Oá%2.3¿ýÉ>úÉ.
00100020	AF	FE	BB	C6	A7	34	5E	7D	85	F7	3C	10	80	49	D8	39	¯þ»Æ§4^}…÷<.€IØ9
00100030	99	C8	E1	01	4A	E5	95	B4	A5	EB	50	6A	B5	32	79	2B	™Èá.Jå•´¥ëPjµ2y+
00100040	3A	FB	64	73	28	0A	7D	15	FD	92	EE	C9	B3	EB	B2	67	:ûds(.}.ý'îÉ³ë²g
00100050	99	8B	15	77	A7	DB	94	DE	9D	56	7B	D3	9E	89	AE	19	™‹.w§Û"Þ.V{Óž‰®.
00100060	69	FF	02	49	38	41	D0	DF	DE	28	20	98	A6	30	5E	12	iÿ.I8AÐßÞ(˜¦0^.
00100070	1B	05	C3	8E	FA	0D	06	91	FE	2D	58	AA	E2	80	74	2C	..ÃŽú..'þ-X ªâ€t,
00100080	A1	2A	F5	7D	48	96	0A	60	81	40	AA	40	0C	8C	37	06	¡*õ}H-.`.@ª@.Œ7.
00100090	5C	10	1E	76	05	43	EF	2A	A9	1D	4A	27	A6	CF	E9	DB	\..v.Cï*©.J'¦ÏéÛ
001000A0	74	3F	FE	8E	19	62	4E	58	06	0E	8B	31	C5	3B	DC	03	t?þŽ.bNX..‹1Å;Ü.
001000B0	A4	5F	B8	B8	E3	28	19	83	C2	B3	B3	89	FD	D0	E5	BC	¤_¸¸ã(.ƒÂ³³‰ýÐå¼
001000C0	7C	FA	6B	9E	CE	2E	17	7E	82	4A	9F	00	57	06	31	BC	|úkžÎ..~‚JŸ.W.1¼
001000D0	F8	89	77	F4	EB	90	8C	0F	2F	71	99	37	B8	91	BD	90	ø‰wôë.Œ./q™7¸'½.
001000E0	53	4D	DE	9C	5A	DB	81	0B	B8	3F	E2	0C	20	C2	0E	E8	SMÞœZÛ..¸?â. Â.è
001000F0	32	B9	78	E5	3F	AA	52	E4	74	02	1F	B3	B3	6C	43	FD	2¹xå?ªRät..³³lCý
00100100	70	FD	DF	EB	04	2E	B3	08	C7	13	02	1D	23	46	51	7F	pýßë..³.Ç...#FQ.
00100110	31	1B	89	BC	48	03	E5	00	50	2F	0B	DF	AF	C4	E8	A0	1.‰¼H.å.P/.ß¯Äè
00100120	73	3A	2A	38	99	67	6C	B1	68	14	FF	EC	73	EE	AD	2F	s:*8™gl±h.ÿìsî./
00100130	DC	A1	30	7B	71	28	D7	52	32	13	3A	A1	B8	12	A9	01	Ü¡0{q(×R2.:¡¸.©.
00100140	9F	46	05	25	2D	E8	B6	AE	2B	89	DE	DB	29	26	36	7A	ŸF.%-è¶®+‰ÞÛ)&6z
00100150	09	51	C7	87	65	E7	19	F5	0D	F0	C4	1C	E2	06	A6	60	.QÇ‡eç.õ.ðÄ.â.¦`
00100160	1F	F3	AF	89	B2	ED	CA	9B	BD	30	D9	D5	9E	B4	48	72	.ó¯‰²íÊ›½0ÙÕž´Hr
00100170	08	91	1A	EC	F0	36	5C	B9	D6	8B	11	F9	36	7F	A3	39	.'.ìð6\¹Ö‹.ù6.£9
00100180	0A	74	6E	AF	7A	50	DA	14	6B	7E	E5	97	2D	63	CF	2E	.tn¯zPÚ.k~å—-cÏ.
00100190	B4	7A	A0	A2	56	BD	A6	0C	0E	64	9C	9E	E1	28	6A	E3	´z ¢V½¦..dœžá(jã
001001A0	BA	4E	16	DF	46	C3	93	C5	4C	27	C7	F4	21	79	1F	95	ºN.ßFÃ"ÅL'Çô!y.•
001001B0	46	F8	98	4C	B7	43	E5	53	3D	86	84	5A	F6	32	8E	82	Fø˜L·CåS=†„Zö2Ž,
001001C0	23	09	68	ED	DF	41	1B	4B	85	B1	DC	5A	C6	FD	04	70	#.híßA.K…±ÜZÆý.p
001001D0	AC	8A	72	52	57	FE	59	57	86	62	C1	63	C8	2C	17	7D	¬ŠrRWþYW†bÁcÈ,.}
001001E0	4E	38	0C	48	26	6B	AD	3D	FC	C6	B4	A9	B1	90	1D	DD	N8.H&k.=üÆ´©±..Ý
001001F0	2F	D1	E8	F1	DE	88	FF	E4	70	DE	36	3C	F5	39	2A	2E	/ÑèñÞˆÿäpÞ6<õ9*.
00100200	51	5E	07	53	38	51	81	2B	7F	93	F4	A4	DC	20	89	5B	Q^.S8Q.+.“ô¤Ü ‰[
00100210	00	00	00	00	00	00	00	00	00	00	00	00	26	00	10	00	..........&....
00100220	10	00	00	00	00	00											.....

Figure 81

Whether the file size is between 1MB and 5MB and the extension is in the targeted lists, the ransomware encrypts the entire content, and the encrypted file has the following structure:

test.sql.PSFUX

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00100000	AA	E6	32	1D	07	8C	04	D8	3C	BA	07	A4	41	DE	CA	EC	ªæ2..Œ.Ø<º.¤AÞÊì
00100010	D8	A9	53	62	C0	57	03	83	F1	83	AE	FE	40	22	B8	E6	Ø©SbÀW.ƒñƒ®þ@"¸æ
00100020	DC	C6	D4	A9	B3	B3	F1	A2	8F	07	3A	1B	F8	75	80	65	ÜÆÔ©³³ñ¢..:.øu€e
00100030	35	B3	82	17	95	2A	A7	34	08	BA	D8	A3	5A	8F	DE	C5	5³‚.•*§4.ºØ£Z.ÞÅ
00100040	22	D4	50	60	25	4E	AC	05	AA	75	52	0A	3F	4D	5D	9B	"ÔP`%N¬.ªuR.?M]›
00100050	4C	BE	80	23	14	E0	AA	8E	9D	68	94	21	73	35	19	DD	L¾€#.àªŽ.h”!s5.Ý
00100060	59	CC	49	26	B8	6B	8F	FE	1C	F6	C4	26	7F	4C	B3	76	YÌI&¸k.þ.öÄ&.L³v
00100070	B6	D3	B5	41	C4	4D	7D	80	35	56	45	CE	3C	68	2B	59	¶ÓµAÄM}€5VEÎ<h+Y
00100080	C0	90	D8	A3	CD	81	0A	B7	ED	EB	71	A2	94	24	51	A6	À.Ø£Í..·íëq¢”$Q¦
00100090	BE	94	A0	20	97	DE	87	F3	8F	99	B3	8E	A1	78	B2	0E	¾” —Þ‡ó.™³Ž¡x².
001000A0	80	ED	E8	12	D4	77	6D	66	1C	AD	E5	11	D0	5D	F2	9F	€íè.Ôwmf..å.Ð]òŸ
001000B0	4C	5C	FB	36	E4	B5	45	1D	B9	23	F4	52	E3	74	30	09	L\û6äµE.¹#ôRãt0.
001000C0	4A	BD	6D	24	DB	DF	9F	A9	B7	AD	D9	A7	92	08	DF	8D	J½m$ÛßŸ©·.Ù§'.ß.
001000D0	92	DD	2A	11	66	63	B2	0A	92	FD	CC	43	1E	22	D0	87	'Ý*.fc².'ýÌC."Ð‡
001000E0	61	C9	95	3A	8A	E5	3B	25	FA	8D	08	E3	E8	48	4E	60	aÉ•:Šå;%ú..ãèHN`
001000F0	EB	BD	B2	EF	3D	95	0C	58	1B	4D	42	49	C0	28	45	A5	ë½²ï=•.X.MBIÀ(E¥
00100100	D2	1C	B7	37	E1	57	9A	D3	6E	42	DC	DA	1D	0C	9F	6B	Ò.·7áWšÓnBÜÚ..Ÿk
00100110	D8	56	A4	8B	35	AA	B2	B4	BC	63	38	44	F7	14	56	CF	ØV¤‹5ª²´¼c8D÷.VÏ
00100120	54	45	32	9C	A9	F0	39	7B	51	F3	7D	28	EF	2B	D3	36	TE2œ©ð9{Qó}(ï+Ó6
00100130	12	F6	42	28	F0	32	70	89	57	FB	AF	F5	9E	64	C7	E4	.öB(ð2p‰Wû¯õždÇä
00100140	0B	BD	53	7F	56	F6	DE	E3	84	B4	88	1B	9F	A9	33	D7	.½S.VöÞã„´ˆ.Ÿ©3×
00100150	5C	82	1B	1A	4B	CD	29	01	FD	22	F9	D9	98	5F	7E	D7	\‚..KÍ).ý"ùÙ˜_~×
00100160	16	E2	66	49	E0	BC	93	D7	99	B0	35	CC	B0	69	4D	FA	.âfIà¼“×™°5Ì°iMú
00100170	C3	91	8D	C6	E5	F7	72	FB	37	BB	A7	BC	91	B3	0C	7F	Ã‘.Æå÷rû7»§¼‘³..
00100180	23	10	14	64	7B	3A	05	11	3B	97	E7	AC	9F	31	9C	A0	#..d{:..;—ç¬Ÿ1œ
00100190	B8	98	FA	61	B0	B1	E7	9A	48	D7	4B	9D	9D	16	A9	27	¸˜úa°±çšH×K...©'
001001A0	DD	54	4D	6B	D7	0A	92	D4	E0	EA	26	B1	A3	56	C0	2E	ÝTMk×.'Ôàê&±£VÀ.
001001B0	84	28	E2	FE	43	30	17	64	DE	87	FB	1A	30	E3	18	37	„(âþC0.dÞ‡û.0ã.7
001001C0	35	00	7C	94	C8	A6	77	5E	33	A7	A8	5F	35	84	85	09	5.|”È¦w^3§¨_5„….
001001D0	29	FD	CD	52	08	A3	CC	95	60	15	E2	1B	AA	68	17	37	)ýÍR.£Ì•`.â.ªh.7
001001E0	04	2E	D2	77	F8	FF	14	13	CD	DE	0C	B2	CC	33	CA	34	..Òwøÿ..ÍÞ.²Ì3Ê4
001001F0	15	31	B2	0A	73	B5	AD	A5	12	61	66	75	70	55	87	95	.1².sµ.¥.afupU‡•
00100200	AC	1A	9A	86	2C	80	17	B3	46	D3	84	A8	DA	0B	B8	BB	¬.š†,€.³FÓ„¨Ú.¸»
00100210	00	00	00	00	00	00	00	00	00	00	00	00	24	00	10	00	..........$....
00100220	10	00	00	00	00	00											.....

Figure 82

Whether the file size is greater than 5MB and the extension is not in the targeted lists, the ransomware encrypts 5 chunks of (file size/100 * 10) bytes. In this case, this value is (0x500010/0x64 * 0xa) = 0x7FFF8 bytes (basically, the malware encrypts 0x7FFF8 bytes, then skips some bytes, and then encrypts 0x7FFF8 bytes again and so on). The structure of the encrypted file is presented below:

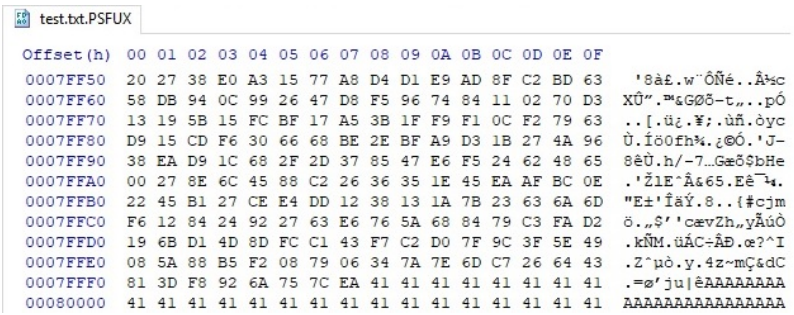


Figure 83

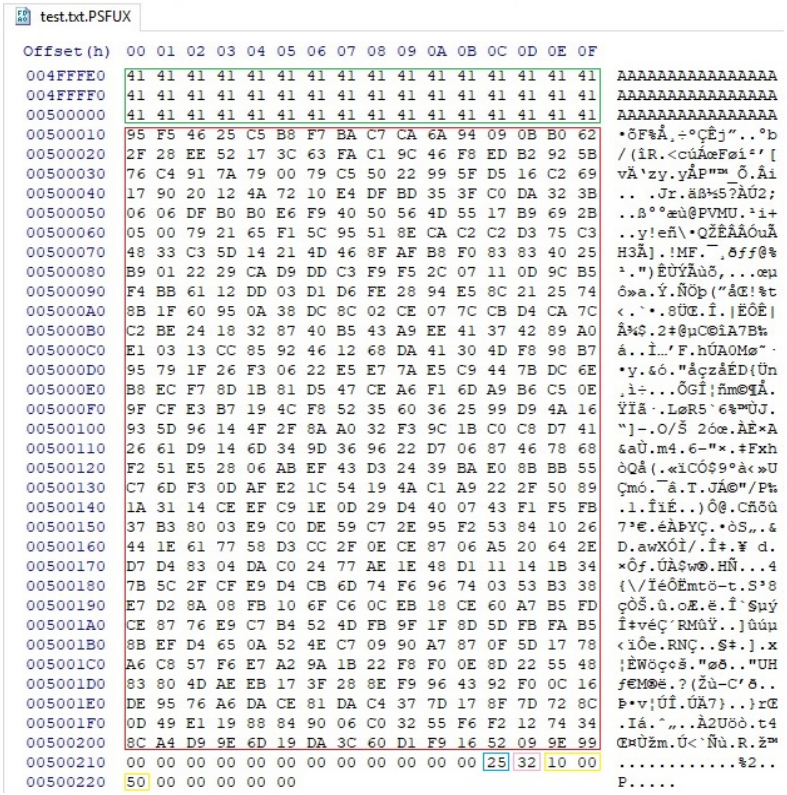
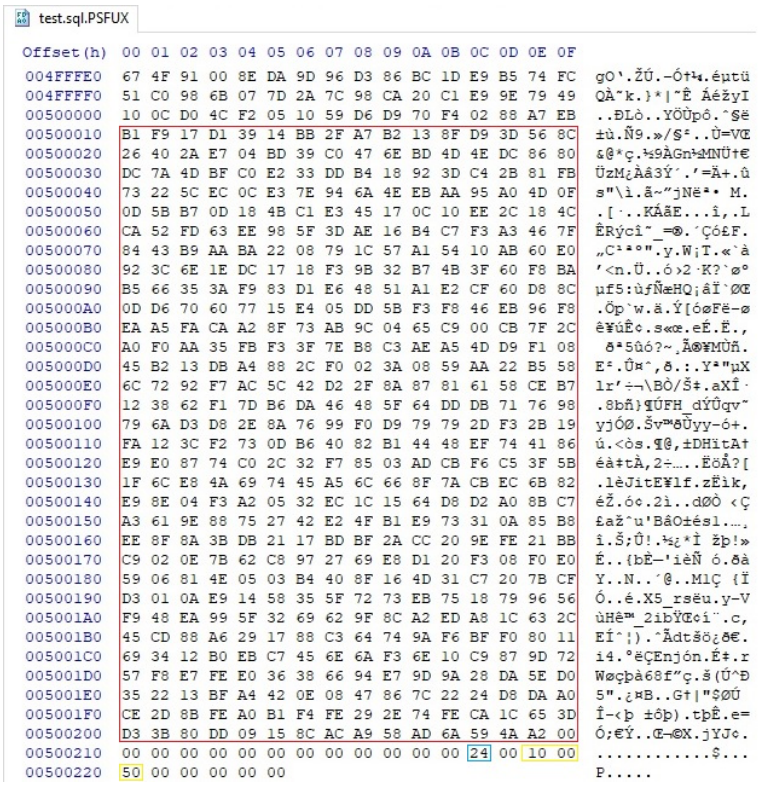


Figure 84

Whether the file size is greater than 5MB and the extension is in the targeted lists, the ransomware encrypts the entire content, and the encrypted file has the following structure:

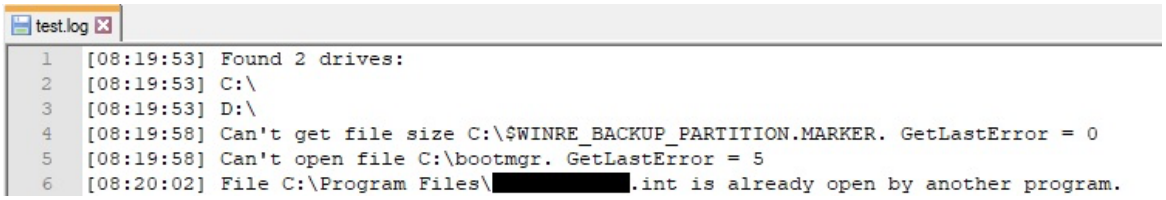


test.sql.PSFUX

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
004FFFE0   67 4F 91 00 8E DA 9D 96 D3 86 BC 1D E9 B5 74 FC  gO‘.ŽÚ.-Ó†¼.éµtü
004FFFF0   51 C0 98 6B 07 7D 2A 7C 98 CA 20 C1 E9 9E 79 49  QÀ˜k.}*|˜Ê Áéžy I
00500000   10 0C D0 4C F2 05 10 59 D6 D9 70 F4 02 88 A7 EB  ..ÐLò..YÖÙpô.^§ë
00500010   B1 F9 17 D1 39 14 BB 2F A7 B2 13 8F D9 3D 56 8C  ±ù.Ñ9.»/§².. Ù=VŒ
00500020   26 40 2A E7 04 BD 39 C0 47 6E BD 4D 4E DC 86 80  &@*ç.½9ÀGn½MNÜt€
00500030   DC 7A 4D BF C0 E2 33 DD B4 18 92 3D C4 2B 81 FB  ÜzM¿Àâ3Ý´.’=Ä+.û
00500040   73 22 5C EC 0C E3 7E 94 6A 4E EB AA 95 A0 4D 0F  s"\ì.ã~”jNë²• M.
00500050   0D 5B B7 0D 18 4B C1 E3 45 17 0C 10 EE 2C 18 4C  .[·..KÁãE...î,.L
00500060   CA 52 FD 63 EE 98 5F 3D AE 16 B4 C7 F3 A3 46 7F  ÊRýcî˜_=®.´Çó£F.
00500070   84 43 B9 AA BA 22 08 79 1C 57 A1 54 10 AB 60 E0  „C¹ª°".y.W¡T.«`à
00500080   92 3C 6E 1E DC 17 18 F3 9B 32 B7 4B 3F 60 F8 BA  ’<n.Ü..ó›2·K?`ø°
00500090   B5 66 35 3A F9 83 D1 E6 48 51 A1 E2 CF 60 D8 8C  µf5:ùƒÑæHQ¡âÏ`ØŒ
005000A0   0D D6 70 60 77 15 E4 05 DD 5B F3 F8 46 EB 96 F8  .Öp`w.ä.Ý[óøFë–ø
005000B0   EA A5 FA CA A2 8F 73 AB 9C 04 65 C9 00 CB 7F 2C  ê¥úÊ¢.s«œ.eÉ.Ë.,
005000C0   A0 F0 AA 35 FB F3 3F 7E B8 C3 AE A5 4D D9 F1 08  ð²5ûó?~¸Ã®¥MÙñ.
005000D0   45 B2 13 DB A4 88 2C F0 02 3A 08 59 AA 22 B5 58  E².Û¤ˆ,ð.:.Yª"µX
005000E0   6C 72 92 F7 AC 5C 42 D2 2F 8A 87 81 61 58 CE B7  lr’÷¬\BÒ/Š‡.aXÎ·
005000F0   12 38 62 F1 7D B6 DA 46 48 5F 64 DD DB 71 76 98  .8bñ}¶ÚFH_dÝÛqv˜
00500100   79 6A D3 D8 2E 8A 76 99 F0 D9 79 79 2D F3 2B 19  yjÓØ.Šv™ðÙyy-ó+.
00500110   FA 12 3C F2 73 0D B6 40 82 B1 44 48 EF 74 41 86  ú.<òs.¶@‚±DHïtA†
00500120   E9 E0 87 74 C0 2C 32 F7 85 03 AD CB F6 C5 3F 5B  éà‡tÀ,2÷….ËöÅ?[
00500130   1F 6C E8 4A 69 74 45 A5 6C 66 8F 7A CB EC 6B 82  .lèJitE¥lf.zËìk‚
00500140   E9 8E 04 F3 A2 05 32 EC 1C 15 64 D8 D2 A0 8B C7  éŽ.ó¢.2ì..dØÒ ‹Ç
00500150   A3 61 9E 88 75 27 42 E2 4F B1 E9 73 31 0A 85 B8  £až‹u'BâO±és1.…¸
00500160   EE 8F 8A 3B DB 21 17 BD BF 2A CC 20 9E FE 21 BB  î.Š;Û!.½¿*Ì žþ!»
00500170   C9 02 0E 7B 62 C8 97 27 69 E8 D1 20 F3 08 F0 E0  É..{bÈ—'ièÑ ó.ðà
00500180   59 06 81 4E 05 03 B4 40 8F 16 4D 31 C7 20 7B CF  Y..N..´@..M1Ç {Ï
00500190   D3 01 0A E9 14 58 35 5F 72 73 EB 75 18 79 96 56  Ó..é.X5_rsëu.y–V
005001A0   F9 48 EA 99 5F 32 69 62 9F 8C A2 ED A8 1C 63 2C  ùHê™_2ibŸŒ¢í¨.c,
005001B0   45 CD 88 A6 29 17 88 C3 64 74 9A F6 BF F0 80 11  EÍˆ¦).ˆÃdtšö¿ð€.
005001C0   69 34 12 B0 EB C7 45 6E 6A F3 6E 10 C9 87 9D 72  i4.°ëÇEnjón.É‡.r
005001D0   57 F8 E7 FE E0 36 38 66 94 E7 9D 9A 28 DA 5E D0  Wøçþà68f”ç.š(Ú^Ð
005001E0   35 22 13 BF A4 42 0E 08 47 86 7C 22 24 D8 DA A0  5".¿¤B..G†|"$ØÚ
005001F0   CE 2D 8B FE A0 B1 F4 FE 29 2E 74 FE CA 1C 65 3D  Î-‹þ ±ôþ).tþÊ.e=
00500200   D3 3B 80 DD 09 15 8C AC A9 58 AD 6A 59 4A A2 00  Ó;€Ý..Œ¬©X.jYJ¢.
00500210   00 00 00 00 00 00 00 00 00 00 00 00 24 00 10 00  ............$...
00500220   50 00 00 00 00 00                                 P.....
```

Figure 85

When the malware runs with the "-log" parameter, then the list of actions is logged in a file:



test.log

```
1  [08:19:53] Found 2 drives:
2  [08:19:53] C:\
3  [08:19:53] D:\
4  [08:19:58] Can't get file size C:\$WINRE_BACKUP_PARTITION.MARKER. GetLastError = 0
5  [08:19:58] Can't open file C:\bootmgr. GetLastError = 5
6  [08:20:02] File C:\Program Files\[redacted].int is already open by another program.
```

Figure 86

APPENDIX

Lists of targeted extensions:

- .4dd .4dl .accdb .accdc .accde .accdr .accdt .accft .adb .ade .adf .adp .arc .ora .alf .ask .btr .bdf .cat .cdb .ckp .cma .cpd .dacpac .dad .dadiagrams .daschema .db .db-shm .db-wal .db3 .dbc .dbf .dbs .dbt .dbv .dbx .dcb .dct .dcx .ddl .dlis .dp1 .dqy .dsk .dsn .dtsx .dxl .eco .ecx .edb .epim .exb .fcd .fdb .fic .fmp .fmp12 .fmpsl .fol .fp3 .fp4 .fp5 .fp7 .fpt .frm .gdb .grdb .gwi .hdb .his .ib .idb .ihx .itdb .itw .jet .jtx .kdb .kexi .kexic .kexis .lgc .lwx .maf .maq .mar .mas .mav .mdb .mdf .mpd .mrg .mud .mwb .myd .ndf .nnt .nrmlib .ns2 .ns3 .ns4 .nsf .nv .nv2 .nwdb .nyf .odb .oqy .orx .owc .p96 .p97 .pan .pdb .pdm .pnz .qry .qvd .rbf .rctd .rod .rodx .rpd .rsd .sas7bdat .sbf .scx .sdb .sdc .sdf .sis .spq .sql .sqlite .sqlite3 .sqlitedb .te .temx .tmd .tps .trc .trm .udb .udl .usr .v12 .vis .vpd .vvv .wdb .wmdb .wrk .xdb .xld .xmlff .abcddb .abs .abx .accdw .adn .db2 .fm5 .hjt .icg .icr .kdb .lut .maw .mdn .mdt
- .vdi .vhd .vmdk .pvm .vmem .vmsn .vmsd .nvram .vmx .raw .qcow2 .subvol .bin .vsv .avhd .vmrs .vhdx .avdx .vmcx .iso

LIFARS
your digital world, secured
244 Fifth Avenue, Suite 2035, New York, NY 10001
LIFARS.com (212) 222-7061 info@lifars.com