LIFARS
your digital world, **secured**

Makop Ransomware

Prepared by: LIFARS, LLC
Date: 8/13/2021

EXECUTIVE SUMMARY

Makop ransomware encrypts user's files using the AES256 algorithm and advises the victims to contact the attackers via Tox (P2P instant-messaging protocol). The ransomware imports an AES256 key that is used to decrypt a lot of strings, including an RSA public key. There is a mutex called "m23071644" created by the process to ensure that only one instance is running at a single time and a new process spawned by the malware that encrypts network shares. The Windows Product ID is extracted from the registry and is used to generate a personal ID that will also be present in the ransom note. The ransomware deletes all volume shadow copies and kills specific processes that could lock different targeted file types. The malware operators are aware of other ransomware families because they don't encrypt possibly encrypted files by ransomware such as Shootlock, RAGA and origami. Two new AES256 keys are generated by the ransomware, which will be used interchangeably to encrypt the content of the files. A new initialization vector (IV) that consists of 16 bytes is generated and stored in the encrypted file, and the AES key used for encryption is encrypted using the RSA public key. There is no possibility to decrypt the files without knowing the RSA private key that corresponds to the hard-coded public one. Even if the operators pretend that they exfiltrate data from the network, we didn't observe any network communications.

ANALYSIS AND FINDINGS

SHA256:9D90919B4434B9CAC736945384857209103FDF1A749671F190C947FDA8CC1681

The malware uses the GetVersion function to retrieve the major and minor version numbers of the OS along with other information:

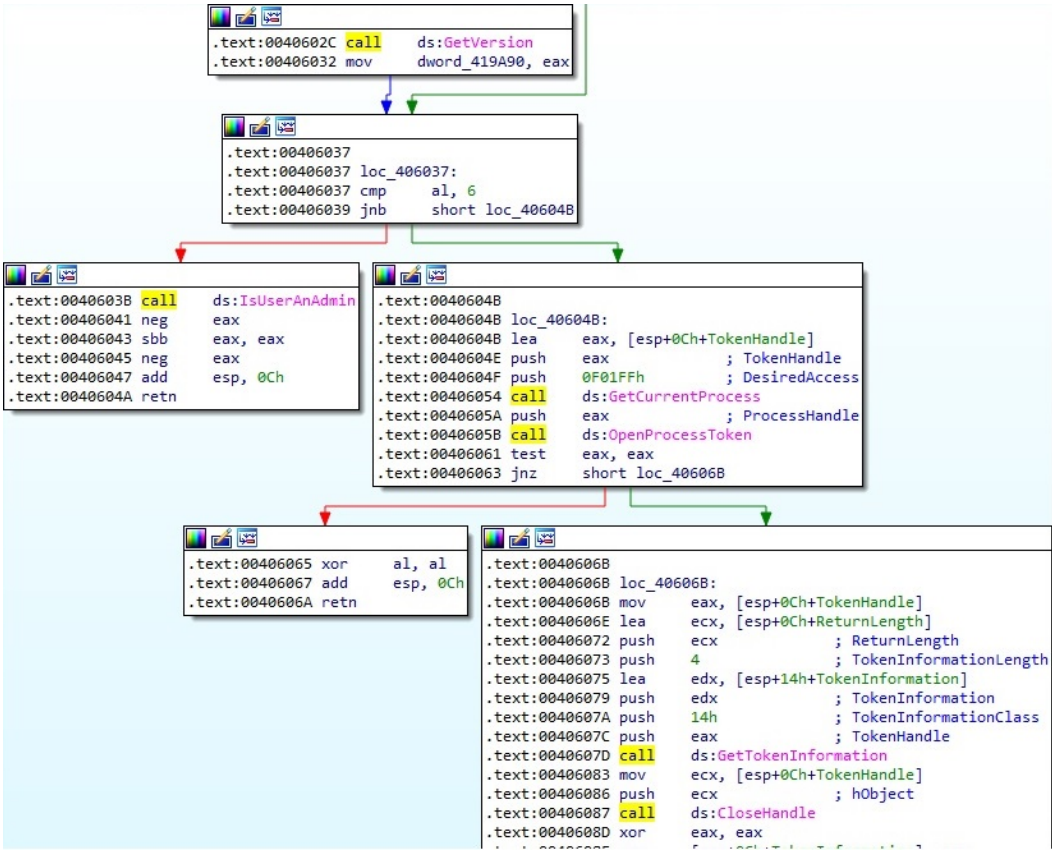


Figure 1

The Get TokenInformation API is used to determine the elevation level of the token (0x14 = **TokenElevationType**):

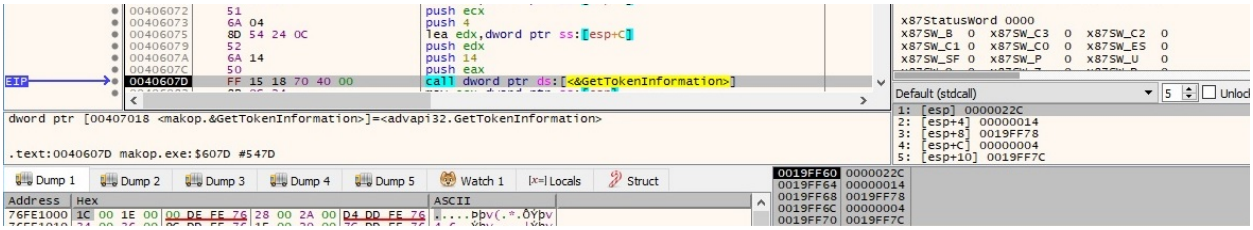


Figure 2

The ransomware retrieves the command-line string for the current process and compares the number of arguments with 2:

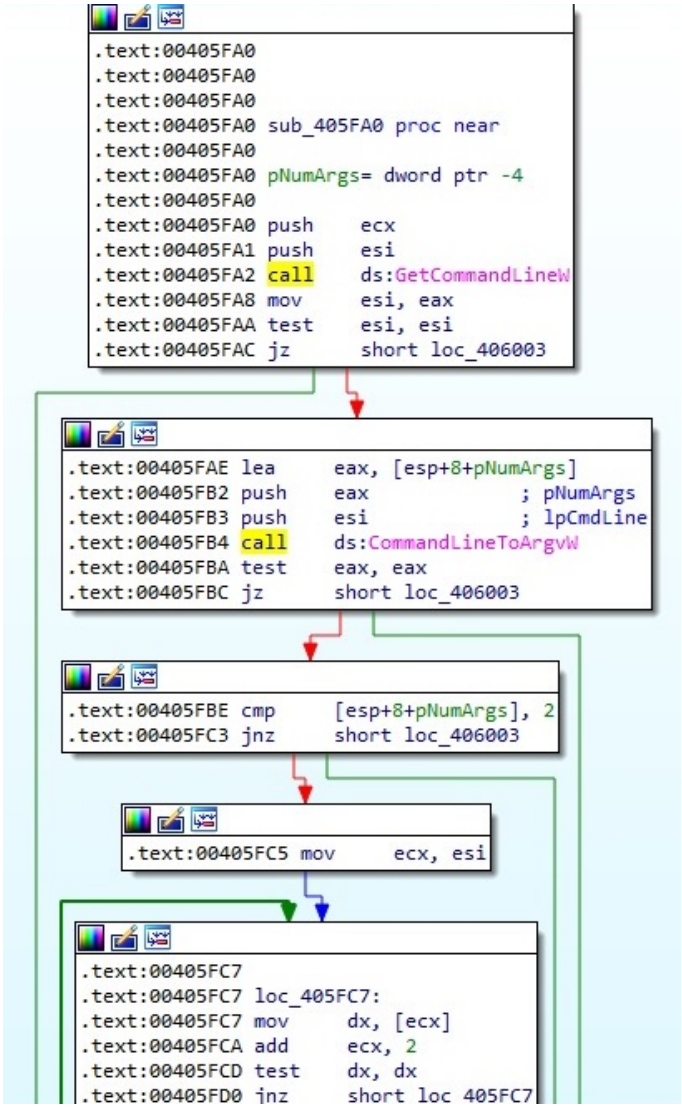


Figure 3

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

The CryptAcquireContextW routine is utilized to acquire a handle to a key container within a cryptographic service provider (0x18 = **PROV_RSA_AES**):

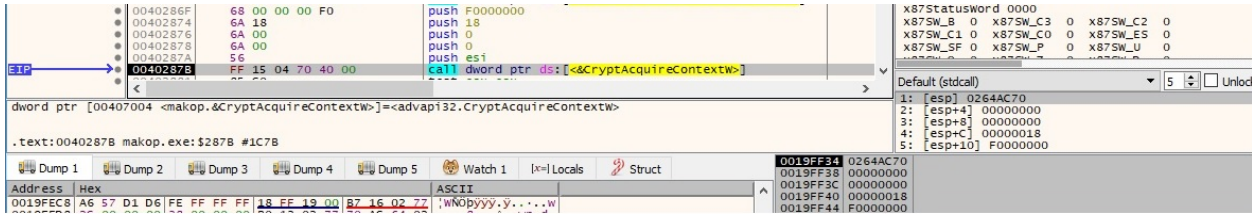


Figure 4

The following 32 bytes represent an AES256 key that will be used to decrypt a lot of strings at runtime:

```
.text:004029E8 mov     [esp+4Ch+var_4C], 52h ; 'R'
.text:004029EC mov     [esp+4Ch+var_4B], 1Fh
.text:004029F1 mov     [esp+4Ch+var_4A], 0CBh ; 'Ë'
.text:004029F6 mov     [esp+4Ch+var_49], 24h ; '$'
.text:004029FB mov     [esp+4Ch+var_48], 54h ; 'T'
.text:00402A00 mov     [esp+4Ch+var_47], al
.text:00402A04 mov     [esp+4Ch+var_46], 1Eh
.text:00402A09 mov     [esp+4Ch+var_45], 7Fh
.text:00402A0E mov     [esp+4Ch+var_44], 0A9h ; '©'
.text:00402A13 mov     [esp+4Ch+var_43], 0D7h ; '×'
.text:00402A18 mov     [esp+4Ch+var_42], 6Bh ; 'k'
.text:00402A1D mov     [esp+4Ch+var_41], 0DAh ; 'Ú'
.text:00402A22 mov     [esp+4Ch+var_40], 0D5h ; 'Õ'
.text:00402A27 mov     [esp+4Ch+var_3F], 0A6h ; '¦'
.text:00402A2C mov     [esp+4Ch+var_3E], 7Eh ; '~'
.text:00402A31 mov     [esp+4Ch+var_3D], 0C6h ; 'Æ'
.text:00402A36 mov     [esp+4Ch+var_3C], 91h ; '‘'
.text:00402A3B mov     [esp+4Ch+var_3B], 0CAh ; 'Ê'
.text:00402A40 mov     [esp+4Ch+var_3A], 0B4h ; '´'
.text:00402A45 mov     [esp+4Ch+var_39], 1Ch
.text:00402A4A mov     [esp+4Ch+var_38], 6
.text:00402A4F mov     [esp+4Ch+var_37], al
.text:00402A53 mov     [esp+4Ch+var_36], 87h ; '‡'
.text:00402A58 mov     [esp+4Ch+var_35], 0BEh ; '¾'
.text:00402A5D mov     [esp+4Ch+var_34], 44h ; 'D'
.text:00402A62 mov     [esp+4Ch+var_33], 20h ; ' '
.text:00402A67 mov     [esp+4Ch+var_32], 0BFh ; '¿'
.text:00402A6C mov     [esp+4Ch+var_31], 0Ch
.text:00402A71 mov     [esp+4Ch+var_30], 0B5h ; 'µ'
.text:00402A76 mov     [esp+4Ch+var_2F], 48h ; 'H'
.text:00402A7B mov     [esp+4Ch+var_2E], 59h ; 'Y'
.text:00402A80 mov     [esp+4Ch+var_2D], 6Dh ; 'm'
```

Figure 5

The AES key constructed earlier is imported by calling the CryptImportKey API, as shown in figure 6:

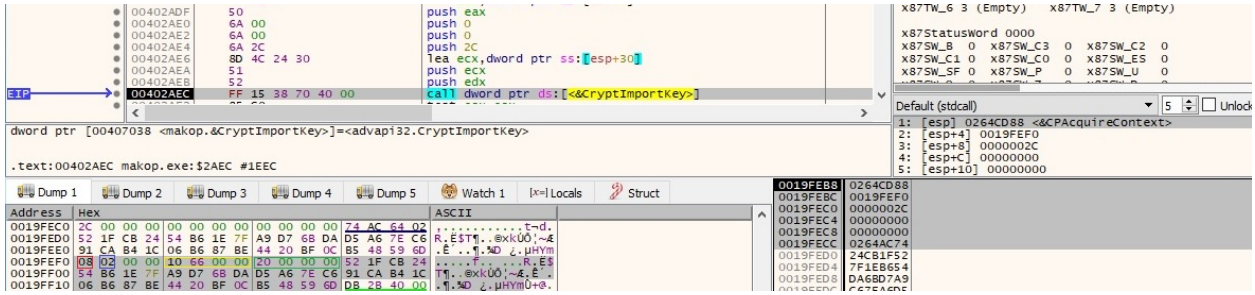


Figure 6

The parameters of the blob are explained below:

- 08 - **PLAINTEXTKEYBLOB** – the key is a session key
- 02 – **CUR_BLOB_VERSION**
- 0x6610 – **CALG_AES_256**
- 0x20 – key size

Using the AES key, the binary decrypts data by calling the CryptDecrypt function:

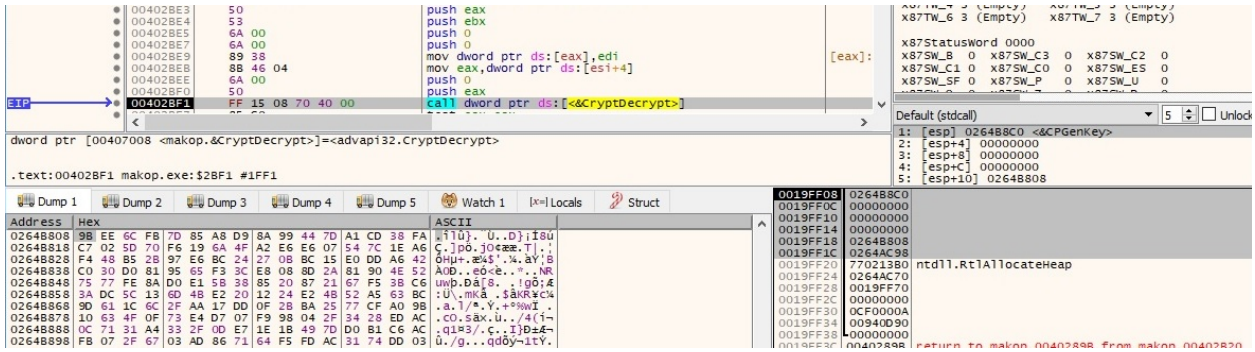
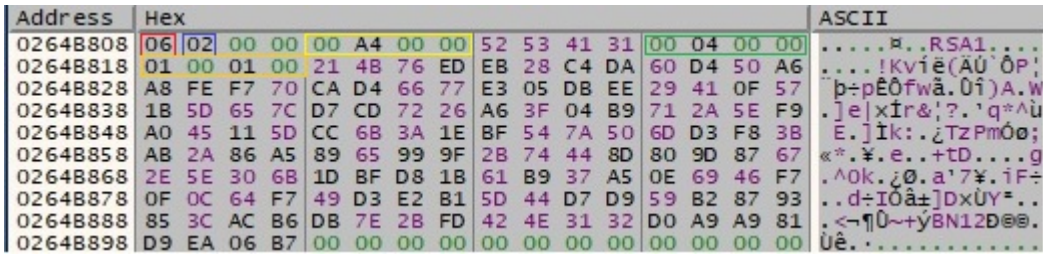


Figure 7

The result of the decryption is an RSA public key:



Address	Hex	ASCII
0264B808	06 02 00 00 00 A4 00 00 52 53 41 31 00 04 00 00	.....¤..RSA1....
0264B818	01 00 01 00 21 4B 76 ED EB 28 C4 DA 60 D4 50 A6	....!Kvíë(ÄÚ`ÔP¦
0264B828	A8 FE F7 70 CA D4 66 77 E3 05 DB EE 29 41 0F 57	¨þ÷pÊÔfwã.Ûî)A.W
0264B838	1B 5D 65 7C D7 CD 72 26 A6 3F 04 B9 71 2A 5E F9	.]e|×Ír&¦?.¹q*^ù
0264B848	A0 45 11 5D CC 6B 3A 1E BF 54 7A 50 6D D3 F8 3B	E.]Ik:.¿TzPmÓø;
0264B858	AB 2A 86 A5 89 65 99 9F 2B 74 44 8D 80 9D 87 67	«*.¥.e..+tD....g
0264B868	2E 5E 30 6B 1D BF D8 1B 61 B9 37 A5 0E 69 46 F7	.^0k.¿Ø.a'7¥.iF÷
0264B878	0F 0C 64 F7 49 D3 E2 B1 5D 44 D7 D9 59 B2 87 93	..d÷IÓâ±]DxÙY²..
0264B888	85 3C AC B6 DB 7E 2B FD 42 4E 31 32 D0 A9 A9 81	.<¬¶Û~+ýBN12Ð©©.
0264B898	D9 EA 06 B7 00 00 00 00 00 00 00 00 00 00 00 00	Ùê.·............

Figure 8

The parameters of the blob are detailed below:

- 06 – **PUBLICKEYBLOB –** the key is a public key
- 02 – **CUR_BLOB_VERSION**
- 0xa400 – **CALG_RSA_KEYX**
- 0x0400 – key size
- 0x010001 – public key exponent

Other strings are decrypted by the malicious process using the same hard-coded AES key:

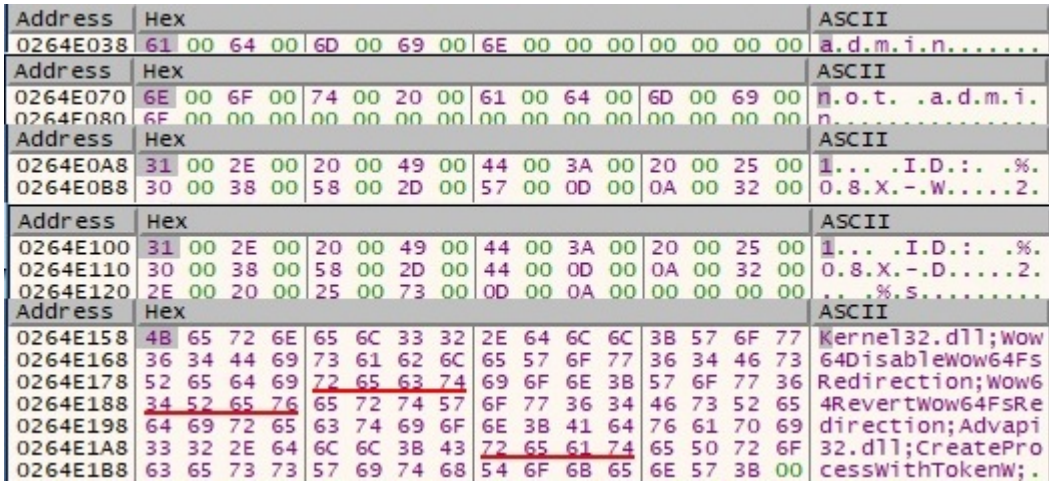


Figure 9

The malware retrieves the address of the following export functions by calling the GetProcAddress routine: Wow64DisableWow64FsRedirection, Wow64RevertWow64FsRedirection and CreateProcessWithTokenW. GetLocaleInfoW is used to retrieve the **LOCALE_FONTSIGNATURE** value for the default locale of the OS (0x800 = **LOCALE_SYSTEM_DEFAULT** and 0x58 = **LOCALE_FONTSIGNATURE**):

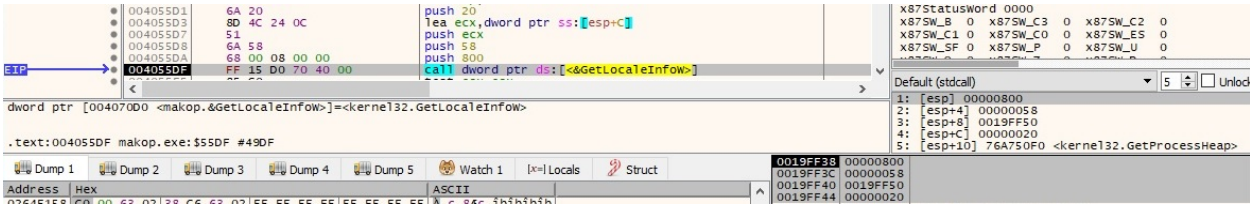
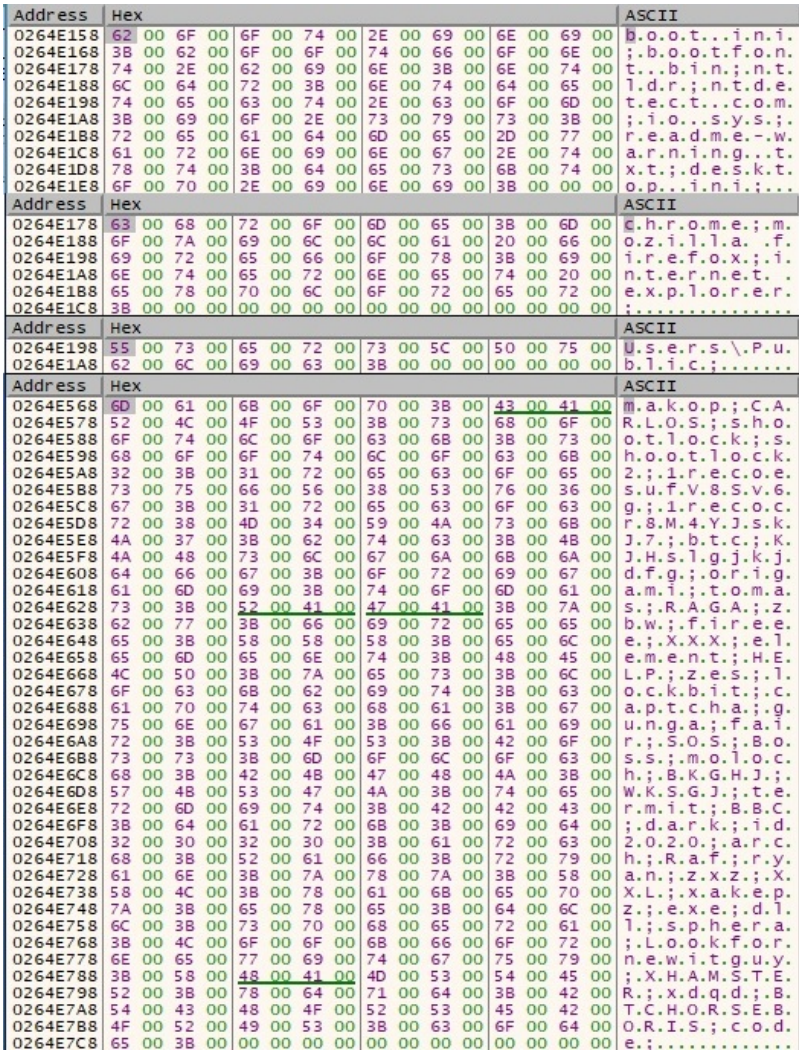


Figure 10

The ransomware decrypts even more strings, and their purpose will be explained later on:



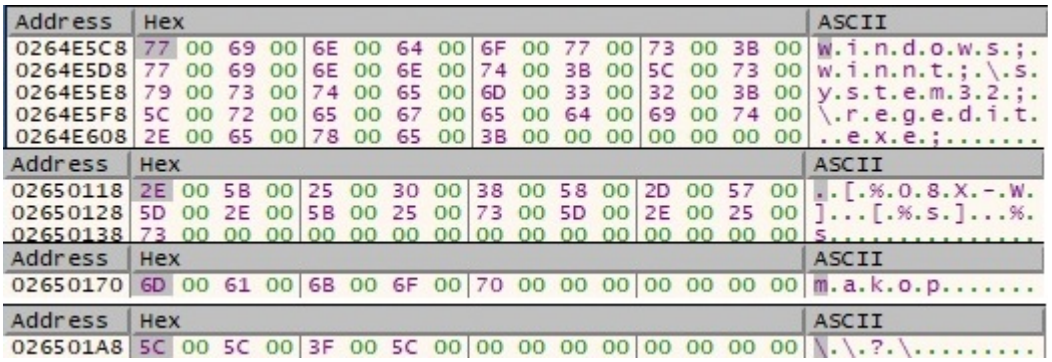
Address	Hex	ASCII
0264E158	62 00 6F 00 6F 00 74 00 2E 00 69 00 6E 00 69 00	b.o.o.t...i.n.i.
0264E168	3B 00 62 00 6F 00 6F 00 74 00 66 00 6F 00 6E 00	;.b.o.o.t.f.o.n.
0264E178	74 00 2E 00 62 00 69 00 6E 00 3B 00 6E 00 74 00	t...b.i.n.;.n.t.
0264E188	6C 00 64 00 72 00 3B 00 6E 00 74 00 64 00 65 00	l.d.r.;.n.t.d.e.
0264E198	74 00 65 00 63 00 74 00 2E 00 63 00 6F 00 6D 00	t.e.c.t...c.o.m.
0264E1A8	3B 00 69 00 6F 00 2E 00 73 00 79 00 73 00 3B 00	;.i.o...s.y.s.;.
0264E1B8	72 00 65 00 61 00 64 00 6D 00 65 00 2D 00 77 00	r.e.a.d.m.e.-.w.
0264E1C8	61 00 72 00 6E 00 69 00 6E 00 67 00 2E 00 74 00	a.r.n.i.n.g...t.
0264E1D8	78 00 74 00 3B 00 64 00 65 00 73 00 6B 00 74 00	x.t.;.d.e.s.k.t.
0264E1E8	6F 00 70 00 2E 00 69 00 6E 00 69 00 3B 00 00 00	o.p...i.n.i.;...

Address	Hex	ASCII
0264E178	63 00 68 00 72 00 6F 00 6D 00 65 00 3B 00 6D 00	c.h.r.o.m.e.;.m.
0264E188	6F 00 7A 00 69 00 6C 00 6C 00 61 00 20 00 66 00	o.z.i.l.l.a. .f.
0264E198	69 00 72 00 65 00 66 00 6F 00 78 00 3B 00 69 00	i.r.e.f.o.x.;.i.
0264E1A8	6E 00 74 00 65 00 72 00 6E 00 65 00 74 00 20 00	n.t.e.r.n.e.t. .
0264E1B8	65 00 78 00 70 00 6C 00 6F 00 72 00 65 00 72 00	e.x.p.l.o.r.e.r.
0264E1C8	3B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	;...............

Address	Hex	ASCII
0264E198	55 00 73 00 65 00 72 00 73 00 5C 00 50 00 75 00	U.s.e.r.s.\.P.u.
0264E1A8	62 00 6C 00 69 00 63 00 3B 00 00 00 00 00 00 00	b.l.i.c.;.......

Address	Hex	ASCII
0264E568	6D 00 61 00 6B 00 6F 00 70 00 3B 00 43 00 41 00	m.a.k.o.p.;.C.A.
0264E578	52 00 4C 00 4F 00 53 00 3B 00 73 00 68 00 6F 00	R.L.O.S.;.s.h.o.
0264E588	6F 00 74 00 6C 00 6F 00 63 00 6B 00 3B 00 73 00	o.t.l.o.c.k.;.s.
0264E598	68 00 6F 00 6F 00 74 00 6C 00 6F 00 63 00 6B 00	h.o.o.t.l.o.c.k.
0264E5A8	32 00 3B 00 31 00 72 00 65 00 63 00 6F 00 65 00	2.;.1.r.e.c.o.e.
0264E5B8	73 00 75 00 66 00 56 00 38 00 53 00 76 00 36 00	s.u.f.V.8.S.v.6.
0264E5C8	67 00 3B 00 31 00 72 00 65 00 63 00 6F 00 63 00	g.;.1.r.e.c.o.c.
0264E5D8	72 00 38 00 4D 00 34 00 59 00 4A 00 73 00 6B 00	r.8.M.4.Y.J.s.k.
0264E5E8	4A 00 37 00 3B 00 62 00 74 00 63 00 3B 00 4B 00	J.7.;.b.t.c.;.K.
0264E5F8	4A 00 48 00 73 00 6C 00 67 00 6A 00 6B 00 6A 00	J.H.s.l.g.j.k.j.
0264E608	64 00 66 00 67 00 3B 00 6F 00 72 00 69 00 67 00	d.f.g.;.o.r.i.g.
0264E618	61 00 6D 00 69 00 3B 00 74 00 6F 00 6D 00 61 00	a.m.i.;.t.o.m.a.
0264E628	73 00 3B 00 52 00 41 00 47 00 41 00 3B 00 7A 00	s.;.R.A.G.A.;.z.
0264E638	62 00 77 00 3B 00 66 00 69 00 72 00 65 00 65 00	b.w.;.f.i.r.e.e.
0264E648	65 00 3B 00 58 00 58 00 58 00 3B 00 65 00 6C 00	e.;.X.X.X.;.e.l.
0264E658	65 00 6D 00 65 00 6E 00 74 00 3B 00 48 00 45 00	e.m.e.n.t.;.H.E.
0264E668	4C 00 50 00 3B 00 7A 00 65 00 73 00 3B 00 6C 00	L.P.;.z.e.s.;.l.
0264E678	6F 00 63 00 6B 00 62 00 69 00 74 00 3B 00 63 00	o.c.k.b.i.t.;.c.
0264E688	61 00 70 00 74 00 63 00 68 00 61 00 3B 00 67 00	a.p.t.c.h.a.;.g.
0264E698	75 00 6E 00 67 00 61 00 3B 00 66 00 61 00 69 00	u.n.g.a.;.f.a.i.
0264E6A8	72 00 3B 00 53 00 4F 00 53 00 3B 00 42 00 6F 00	r.;.S.O.S.;.B.o.
0264E6B8	73 00 73 00 3B 00 6D 00 6F 00 6C 00 6F 00 63 00	s.s.;.m.o.l.o.c.
0264E6C8	68 00 3B 00 42 00 4B 00 47 00 48 00 4A 00 3B 00	h.;.B.K.G.H.J.;.
0264E6D8	57 00 4B 00 53 00 47 00 4A 00 3B 00 74 00 65 00	W.K.S.G.J.;.t.e.
0264E6E8	72 00 6D 00 69 00 74 00 3B 00 42 00 42 00 43 00	r.m.i.t.;.B.B.C.
0264E6F8	3B 00 64 00 61 00 72 00 6B 00 3B 00 69 00 64 00	;.d.a.r.k.;.i.d.
0264E708	32 00 30 00 32 00 30 00 3B 00 61 00 72 00 63 00	2.0.2.0.;.a.r.c.
0264E718	68 00 3B 00 52 00 61 00 66 00 3B 00 72 00 79 00	h.;.R.a.f.;.r.y.
0264E728	61 00 6E 00 3B 00 7A 00 78 00 7A 00 3B 00 58 00	a.n.;.z.x.z.;.X.
0264E738	58 00 4C 00 3B 00 78 00 61 00 6B 00 65 00 70 00	X.L.;.x.a.k.e.p.
0264E748	7A 00 3B 00 65 00 78 00 65 00 3B 00 64 00 6C 00	z.;.e.x.e.;.d.l.
0264E758	6C 00 3B 00 73 00 70 00 68 00 65 00 72 00 61 00	l.;.s.p.h.e.r.a.
0264E768	3B 00 4C 00 6F 00 6F 00 6B 00 66 00 6F 00 72 00	;.L.o.o.k.f.o.r.
0264E778	6E 00 65 00 77 00 69 00 74 00 67 00 75 00 79 00	n.e.w.i.t.g.u.y.
0264E788	3B 00 58 00 48 00 41 00 4D 00 53 00 54 00 45 00	;.X.H.A.M.S.T.E.
0264E798	52 00 3B 00 78 00 64 00 71 00 64 00 3B 00 42 00	R.;.x.d.q.d.;.B.
0264E7A8	54 00 43 00 48 00 4F 00 52 00 53 00 45 00 42 00	T.C.H.O.R.S.E.B.
0264E7B8	4F 00 52 00 49 00 53 00 3B 00 63 00 6F 00 64 00	O.R.I.S.;.c.o.d.
0264E7C8	65 00 3B 00 00 00 00 00 00 00 00 00 00 00 00 00	e.;.............

Figure 11



Address	Hex	ASCII
0264E5C8	77 00 69 00 6E 00 64 00 6F 00 77 00 73 00 3B 00	w.i.n.d.o.w.s.;.
0264E5D8	77 00 69 00 6E 00 6E 00 74 00 3B 00 5C 00 73 00	w.i.n.n.t.;.\.s.
0264E5E8	79 00 73 00 74 00 65 00 6D 00 33 00 32 00 3B 00	y.s.t.e.m.3.2.;.
0264E5F8	5C 00 72 00 65 00 67 00 65 00 64 00 69 00 74 00	\.r.e.g.e.d.i.t.
0264E608	2E 00 65 00 78 00 65 00 3B 00 00 00 00 00 00 00	..e.x.e.;.......

Address	Hex	ASCII
02650118	2E 00 5B 00 25 00 30 00 38 00 58 00 2D 00 57 00	.[.%.0.8.X.-.W.
02650128	5D 00 2E 00 5B 00 25 00 73 00 5D 00 2E 00 25 00	]...[.%.s.]...%.
02650138	73 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	s...............

Address	Hex	ASCII
02650170	6D 00 61 00 6B 00 6F 00 70 00 00 00 00 00 00 00	m.a.k.o.p.......

Address	Hex	ASCII
026501A8	5C 00 5C 00 3F 00 5C 00 00 00 00 00 00 00 00 00	\.\.?.\.........

Figure 12

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

The binary retrieves a handle to the Shell's desktop window using the GetShellWindow API, as shown in the next figure:

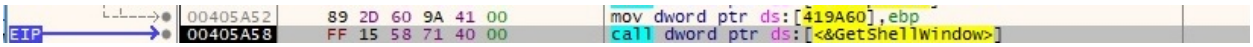


Figure 13

GetWindowThreadProcessId is utilized to extract the identifier of the thread and of the process that created the window from above:

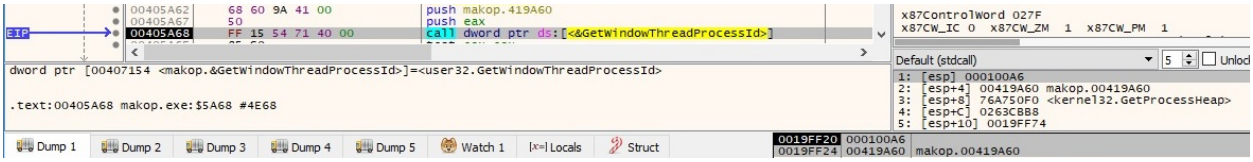


Figure 14

The malware obtains the join status information for the local computer by calling the NetGetJoinInformation function:

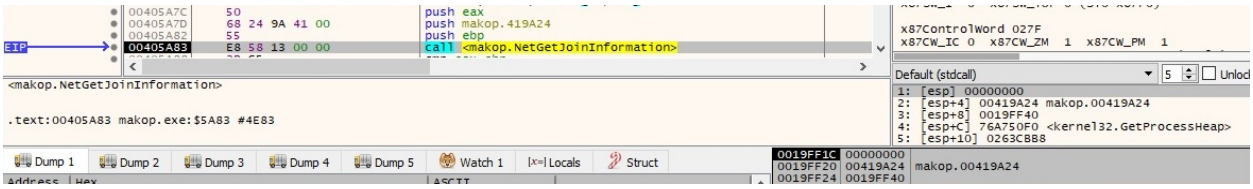


Figure 15

Some directories names and a mutex name are decrypted by the executable, as shown in figure 16:

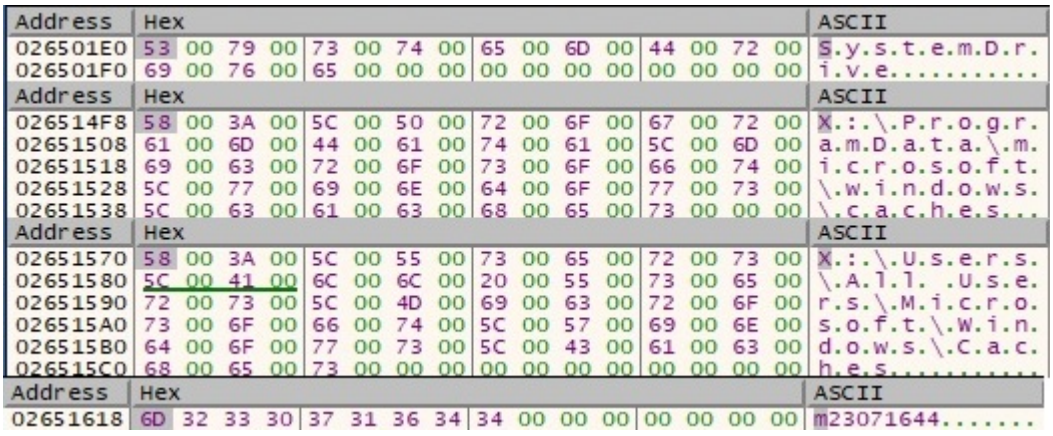


Figure 16

The value of the "SystemDrive" environment variable is retrieved using the GetEnvironmentVariableW API:

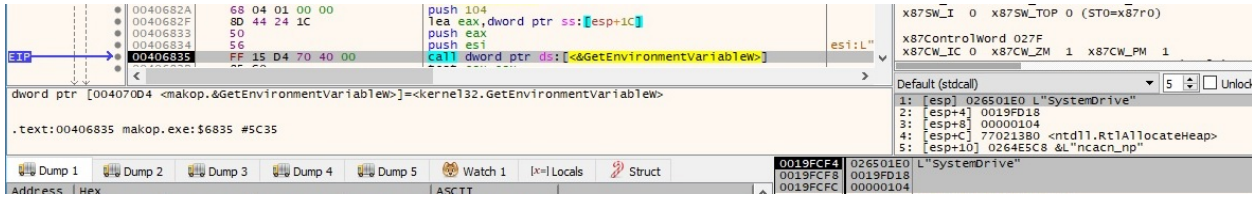


Figure 17

The ransomware creates a mutex called "m23071644" to ensure that only one instance of the executable is running at a single time:

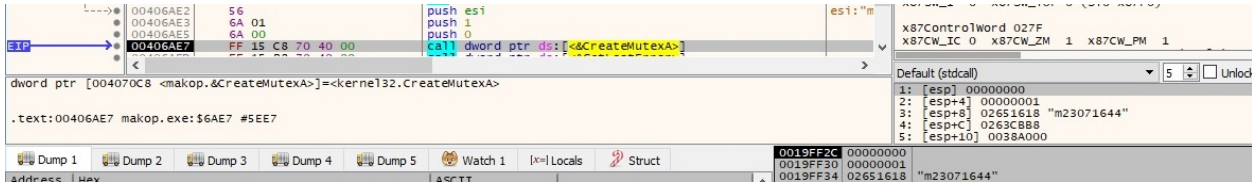


Figure 18

The process opens the "SOFTWARE\Microsoft\Windows NT\CurrentVersion" registry key using the RegOpenKeyExA routine:

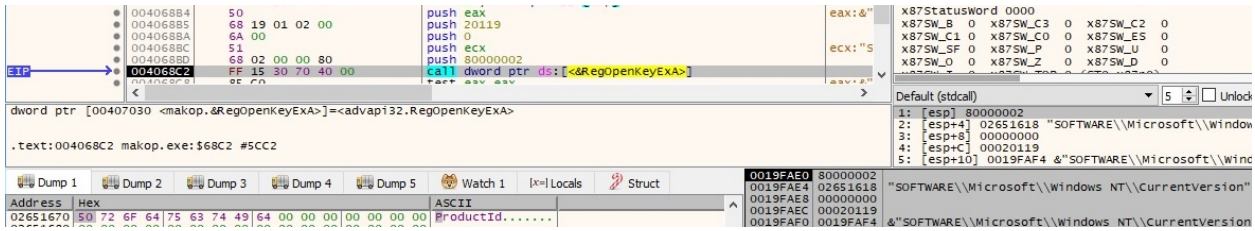


Figure 19

The Windows product ID is extracted from the registry and it will be used to compute a victim ID:

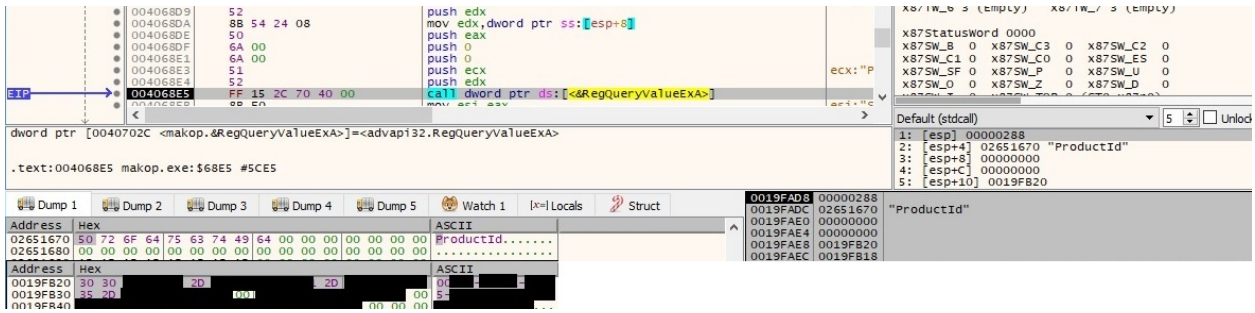


Figure 20

The ransomware extracts the volume serial number of the C drive by calling the GetVolumeInformationW API:

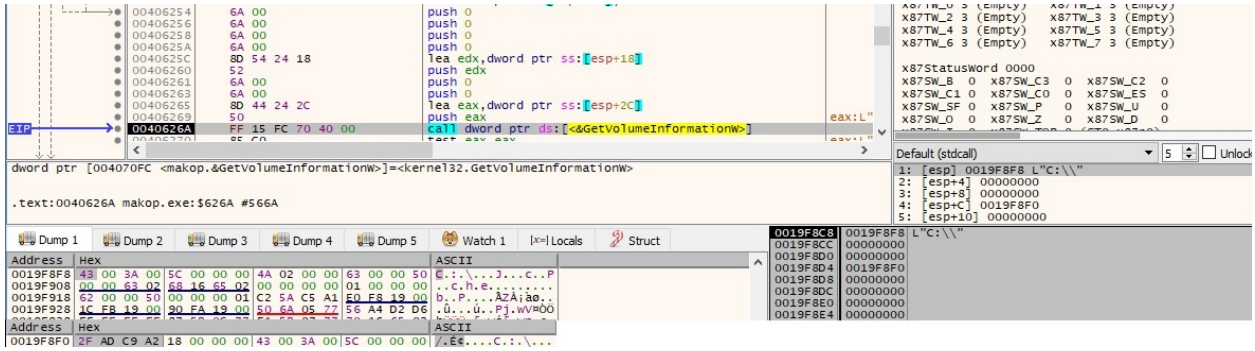


Figure 21

The malware uses a custom "hash" function to compute a 4-byte value that corresponds to the Product ID. A snippet of the implementation is shown below:

Figure 22

Two more strings are decrypted by the ransomware using the AES key imported before:

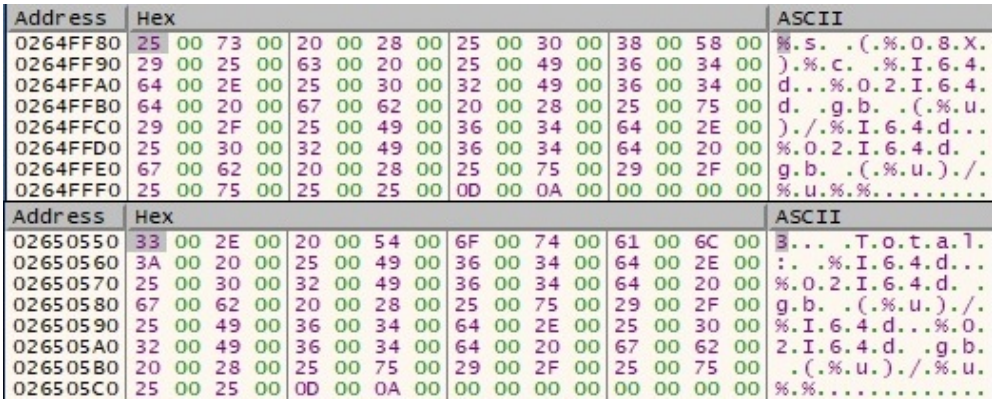


Figure 23

The binary disables the file system redirection for the calling thread using Wow64DisableWow64FsRedirection:

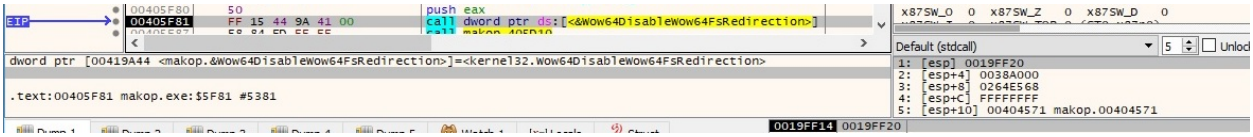


Figure 24

An open handle to the current process is obtained by calling the GetCurrentProcessId and OpenProcess APIs:

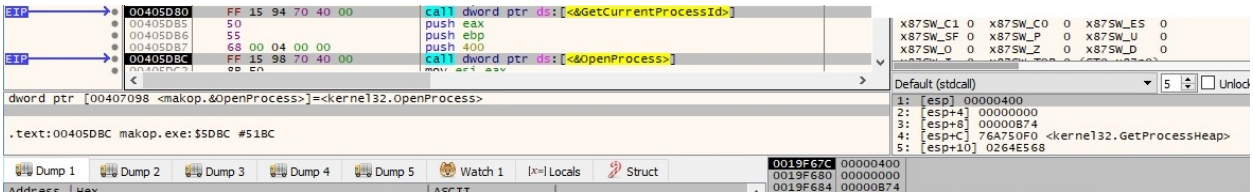


Figure 25

The malicious executable opens the access token associated with the current process:

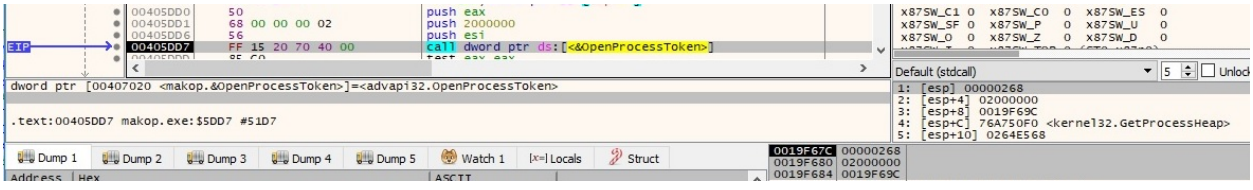


Figure 26

DuplicateTokenEx is utilized to create a new access token that duplicates the existing token:

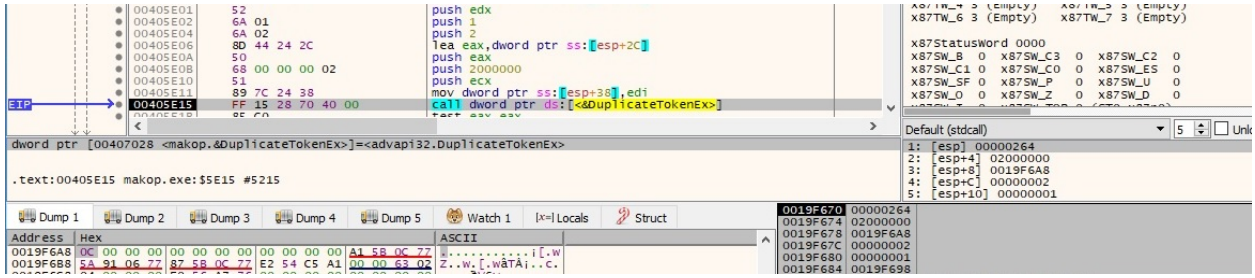


Figure 27

The ransomware creates an anonymous pipe by calling the CreatePipe function:

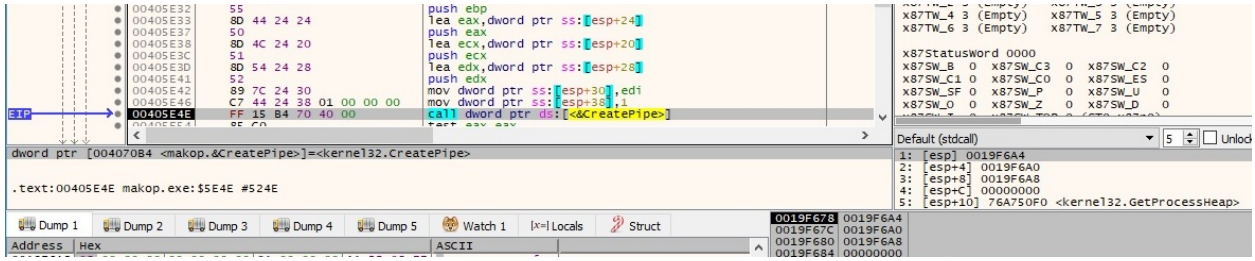


Figure 28

There is a new process spawned by the malware with the "-n<Process ID>" parameter. The new process handles the encryption of the network shares, as will be detailed in the upcoming paragraphs:

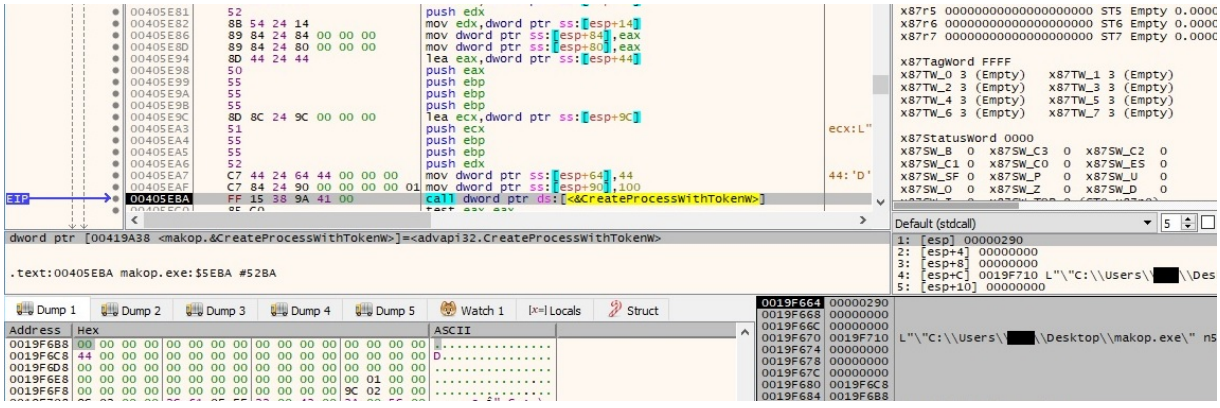


Figure 29

The process restores the file system redirection for the current thread:

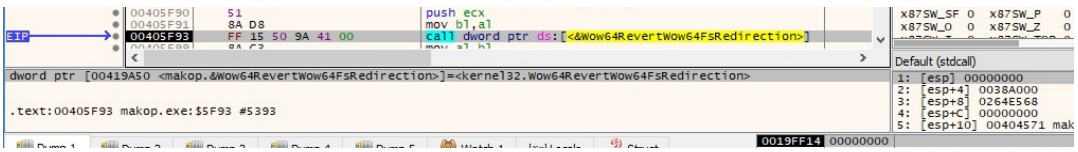


Figure 30

The file enforces the system to send critical errors to the calling process using the SetErrorMode API (0x1 = **SEM_FAILCRITICALERRORS**). It obtains the currently available disk drives using GetLogicalDrives:

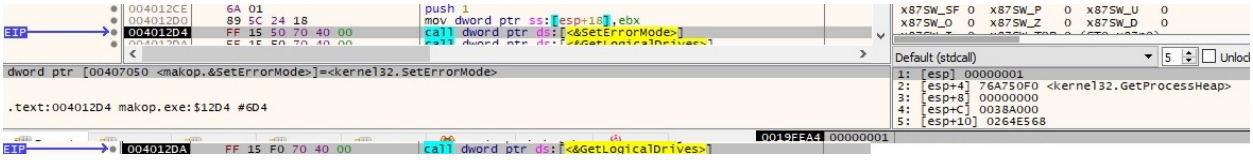


Figure 31

The GetDriveTypeW API determines if a disk drive is a removable, fixed, CD-ROM, RAM or network drive. Makop doesn't target CD-ROM drives and RAM disks:

Figure 32

The malware opens the "C:\" drive using the CreateFileW routine (0x80000000 = **GENERIC_READ,** 0x3 = **FILE_SHARE_READ|FILE_SHARE_WRITE,** 0x3 = **OPEN_EXISTING** and 0x80 = **FILE_ATTRIBUTE_NORMAL**):

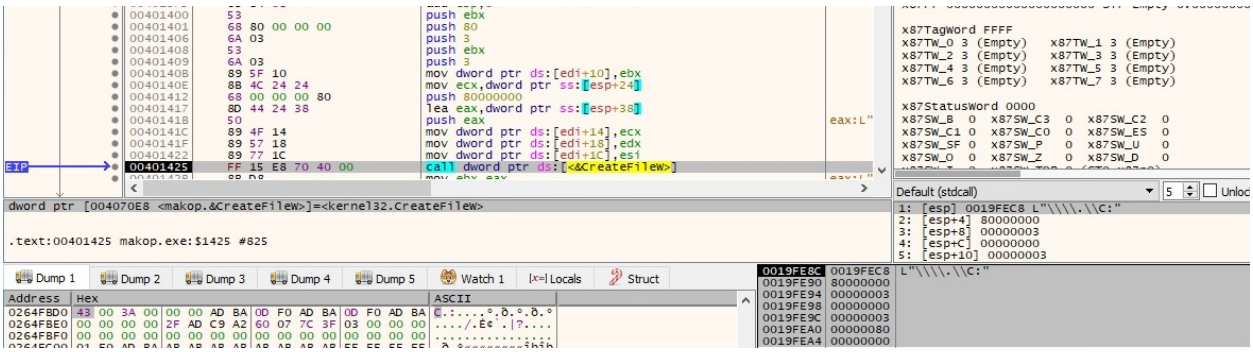


Figure 33

DeviceIoControl is utilized to retrieve the physical location of the specified volume (0x560000 = **IOCTL_VOLUME_GET_VOLUME_DISK_EXTENTS**):

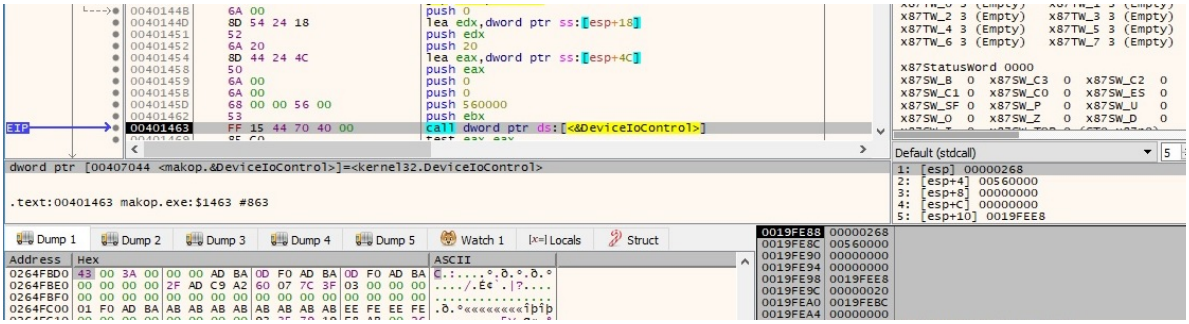


Figure 34

Makop generates 2 buffers of 32 random bytes using the CryptGenRandom API (let's call the first one **AESKey1** and the second one **AESKey2**):

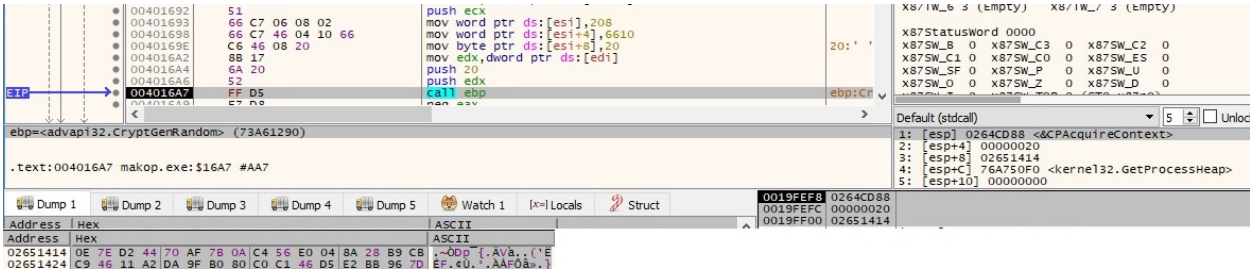


Figure 35

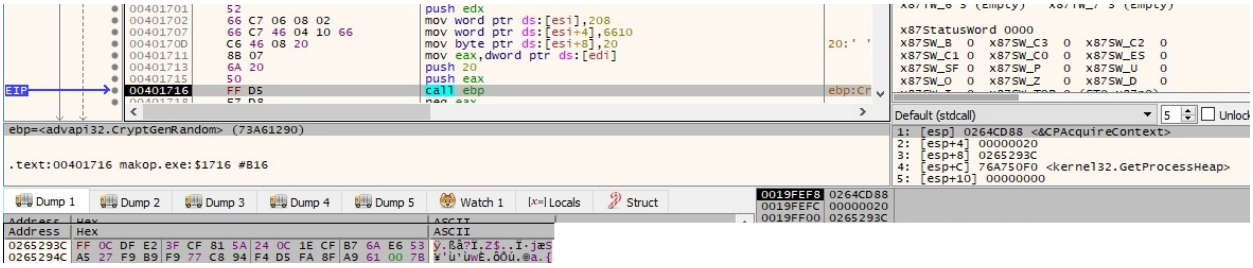


Figure 36

4 bytes that will be used as a marker in the encrypted files are decrypted by the binary: "AD AD 6B A1". The RSA public key is imported using the CryptImportKey function:

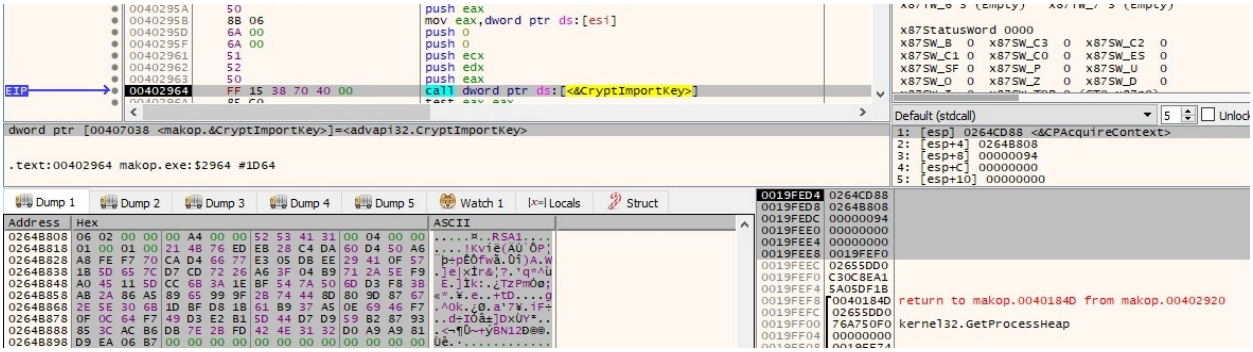


Figure 37

These 2 buffers are encrypted using the RSA public key and contain the following information: 4-byte marker, 4-byte "hash" value of Product ID generated earlier, 4-byte the volume serial number, 4-byte value obtained from the result of the GetLocaleInfoW call, **AESKey1** (or **AESKey2**) and 4-byte "hash" value of this buffer, computed using the same function used for Product ID:

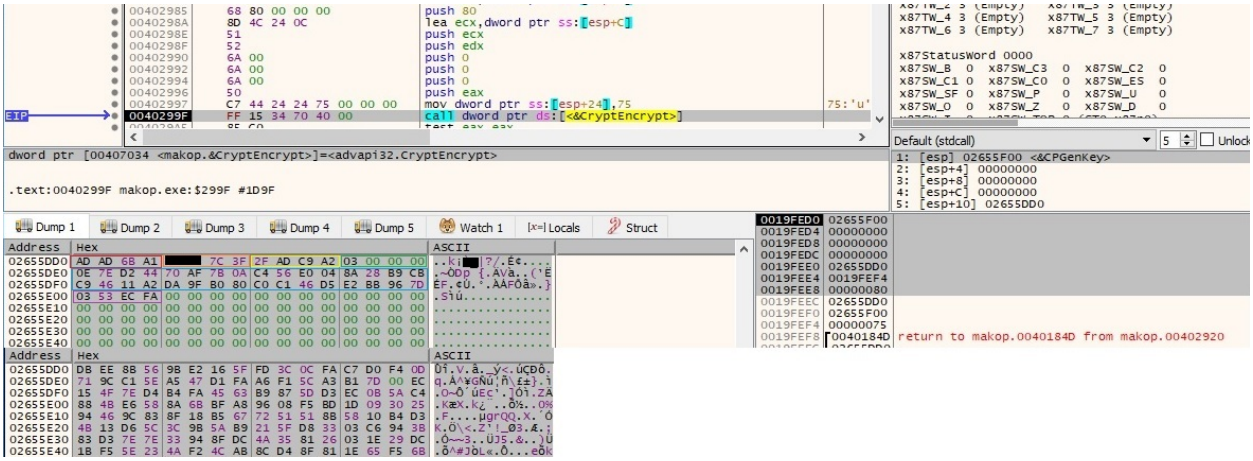


Figure 38

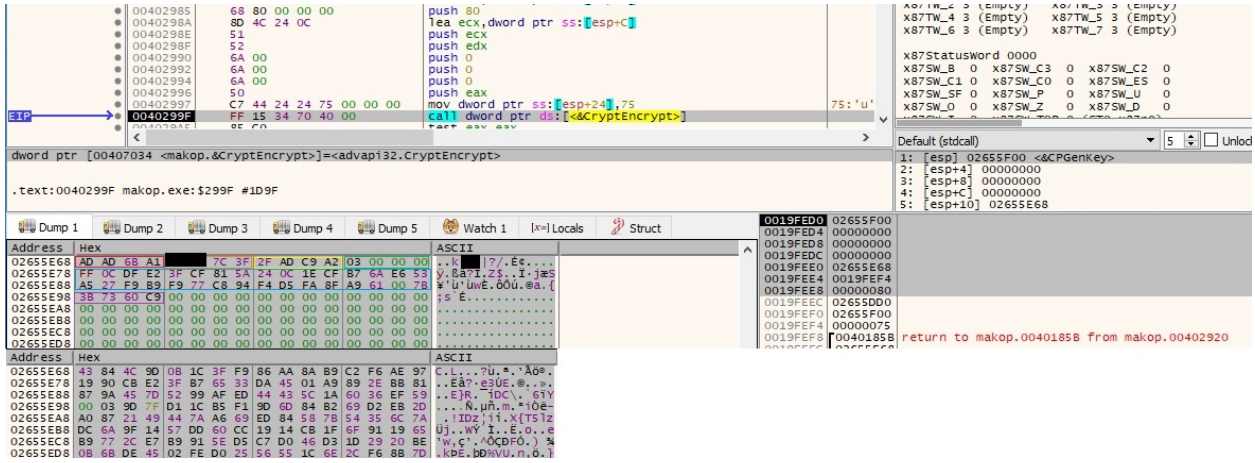
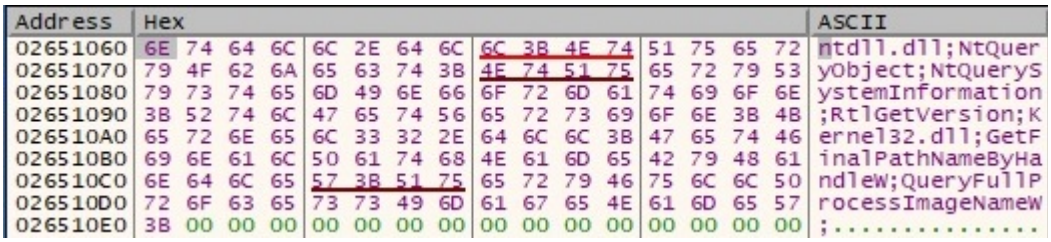


Figure 39

The ransomware decrypts the following strings and uses the GetProcAddress API to obtain the addresses of the export functions:



Address	Hex	ASCII
02651060	6E 74 64 6C 6C 2E 64 6C 6C 3B 4E 74 51 75 65 72	ntdll.dll;NtQuer
02651070	79 4F 62 6A 65 63 74 3B 4E 74 51 75 65 72 79 53	yObject;NtQueryS
02651080	79 73 74 65 6D 49 6E 66 6F 72 6D 61 74 69 6F 6E	ystemInformation
02651090	3B 52 74 6C 47 65 74 56 65 72 73 69 6F 6E 3B 4B	;RtlGetVersion;K
026510A0	65 72 6E 65 6C 33 32 2E 64 6C 6C 3B 47 65 74 46	ernel32.dll;GetF
026510B0	69 6E 61 6C 50 61 74 68 4E 61 6D 65 42 79 48 61	inalPathNameByHa
026510C0	6E 64 6C 65 57 3B 51 75 65 72 79 46 75 6C 6C 50	ndleW;QueryFullP
026510D0	72 6F 63 65 73 73 49 6D 61 67 65 4E 61 6D 65 57	rocessImageNameW
026510E0	3B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	;...

Figure 40

NtQueryObject is used to retrieve information about the system and the current process (0x3 = **ObjectAllTypesInformation**):

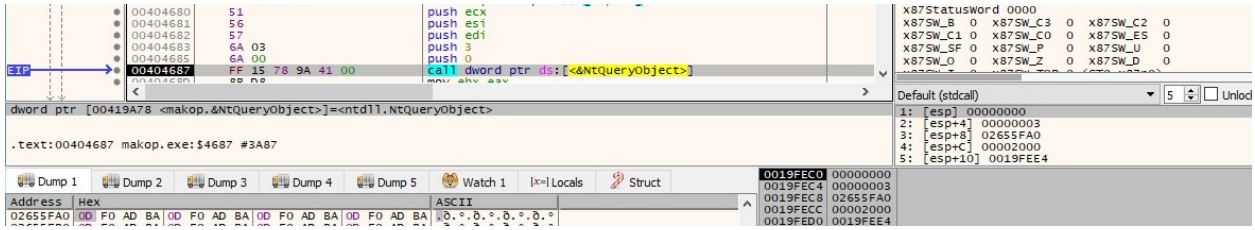


Figure 41

The executable gets information about the OS using the RtlGetVersion API. It compares the major version number (0xa for Windows 10) of the OS with 0x6:

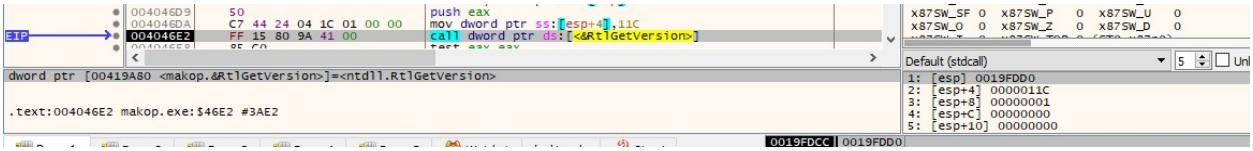


Figure 42

There is a call to GetTokenInformation that obtains information about whether virtualization is enabled for the token (0x18 = **TokenVirtualizationEnabled**):

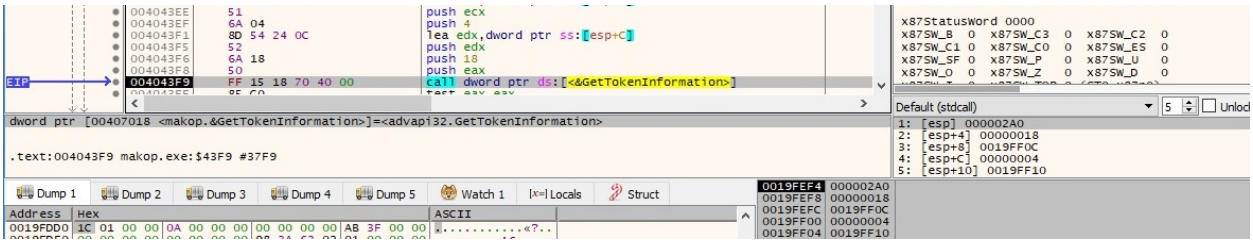
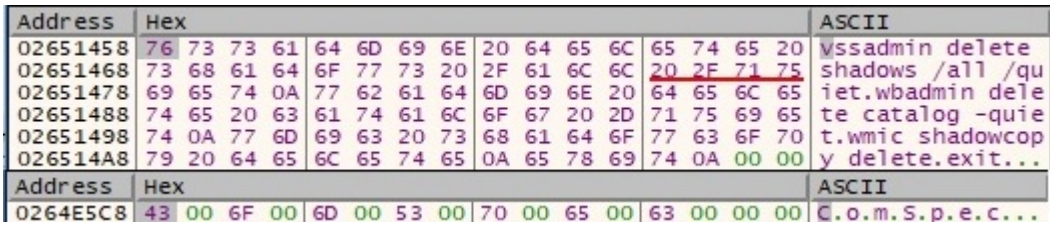


Figure 43

The malware decrypts even more data using the CryptDecrypt routine:



Address	Hex	ASCII
02651458	76 73 73 61 64 6D 69 6E 20 64 65 6C 65 74 65 20	vssadmin delete
02651468	73 68 61 64 6F 77 73 20 2F 61 6C 6C 20 2F 71 75	shadows /all /qu
02651478	69 65 74 0A 77 62 61 64 6D 69 6E 20 64 65 6C 65	iet.wbadmin dele
02651488	74 65 20 63 61 74 61 6C 6F 67 20 2D 71 75 69 65	te catalog -quie
02651498	74 0A 77 6D 69 63 20 73 68 61 64 6F 77 63 6F 70	t.wmic shadowcop
026514A8	79 20 64 65 6C 65 74 65 0A 65 78 69 74 0A 00 00	y delete.exit...

Address	Hex	ASCII
0264E5C8	43 00 6F 00 6D 00 53 00 70 00 65 00 63 00 00 00	C.o.m.S.p.e.c...

Figure 44

The ComSpec environment variable points to the command line interpreter and its content is extracted using the GetEnvironmentVariableW API:

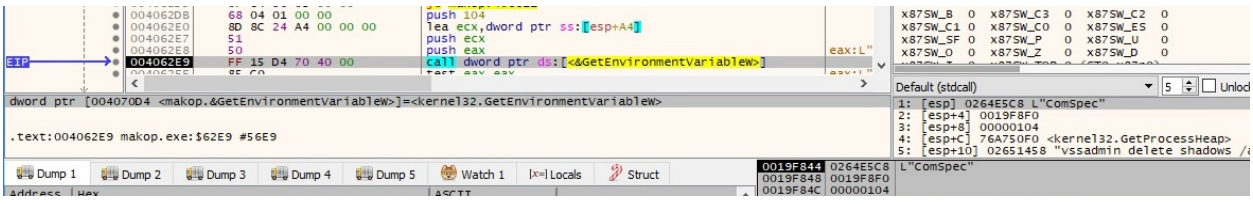


Figure 45

The CreatePipe routine is utilized to create an anonymous pipe that is used as an inter-process communication mechanism:

Figure 46

The pipe created above will be inherited by child processes:

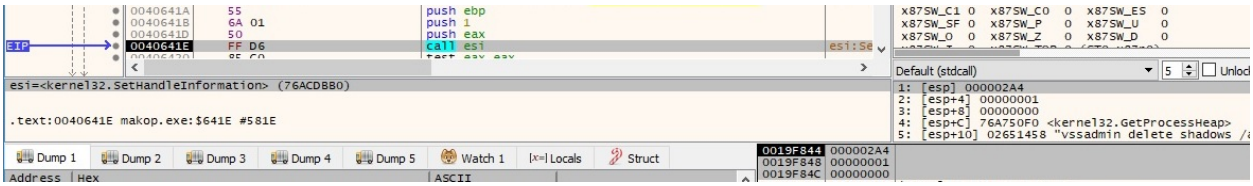


Figure 47

There is a new cmd.exe process created by the ransomware:

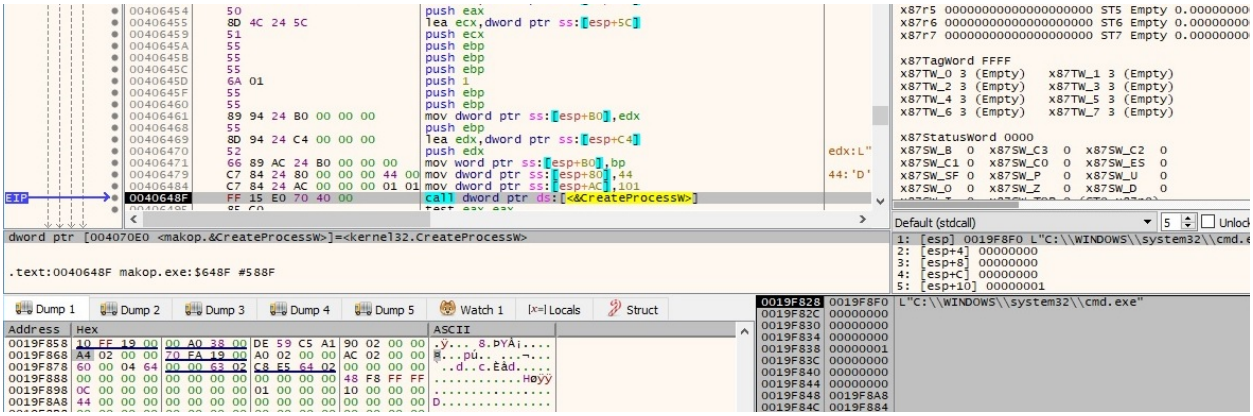


Figure 48

All volume shadow copies are deleted by the cmd.exe process using the following commands:

- vssadmin delete shadows /all /quiet
- wbadmin delete catalog -quiet
- wmic shadowcopy delete

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

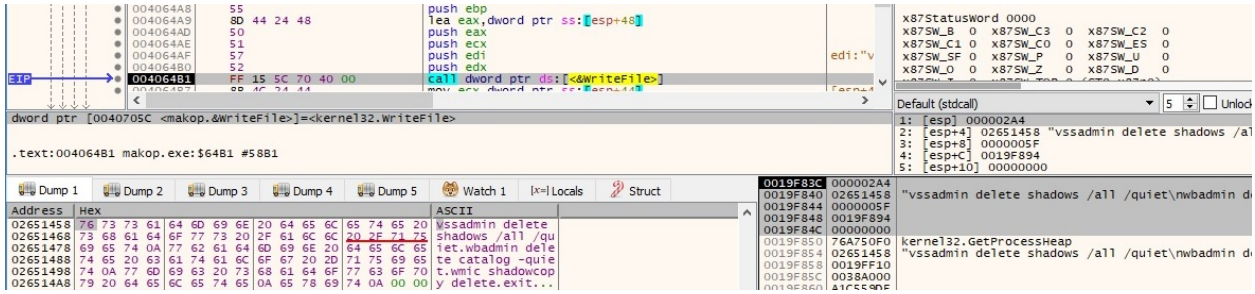


Figure 49

The output of the above operations is transmitted to our initial process via pipes:

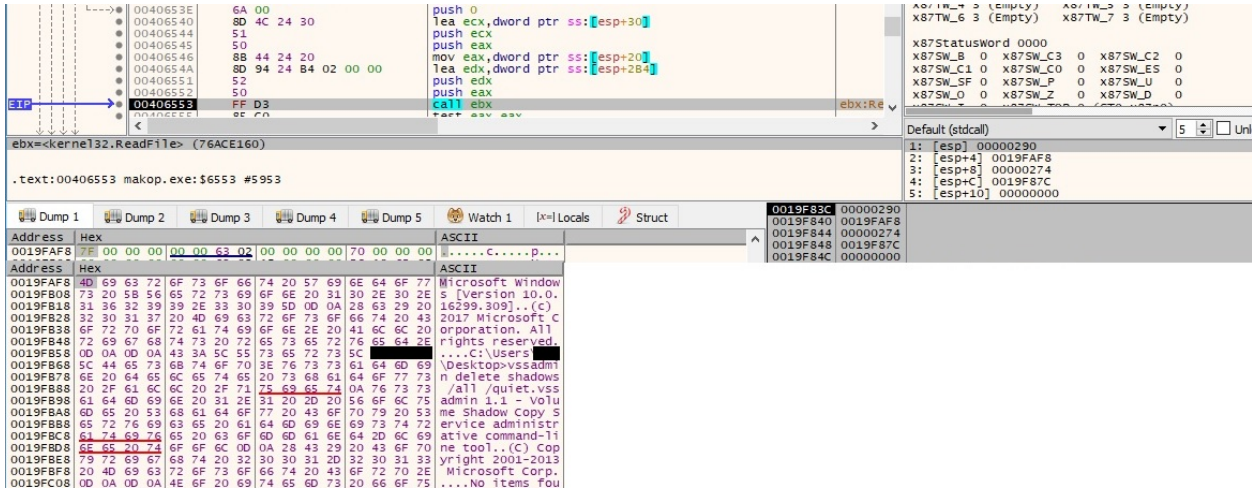


Figure 50

A small part of the processes that will be killed by the malware is presented in figure 51 (the entire list can be found in the appendix):

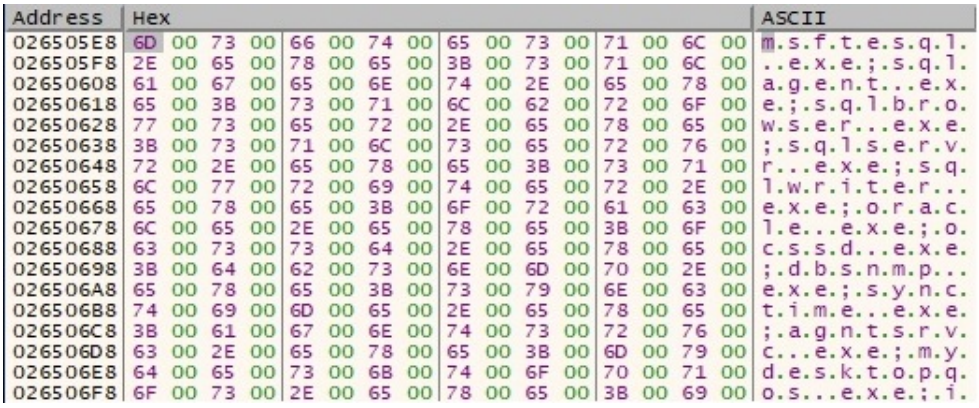


Figure 51

CreateToolhelp32Snapshot is utilized to take a snapshot of the processes (0x2 = **TH32CS_SNAPPROCESS**):

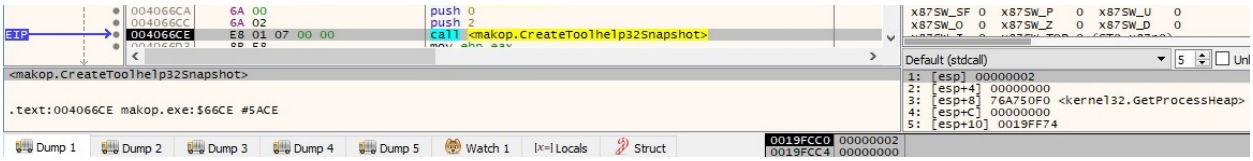


Figure 52

The processes are enumerated using the Process32FirstW and Process32NextW functions:

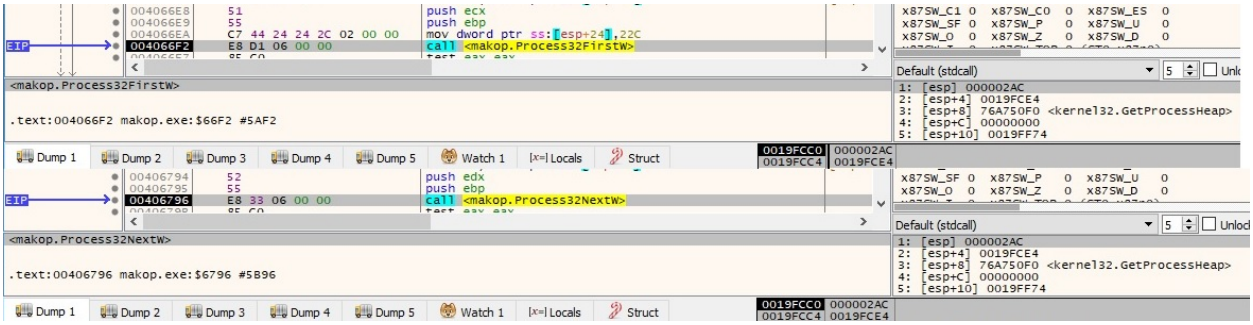


Figure 53

The following function is used to compare processes' names with the targeted list:

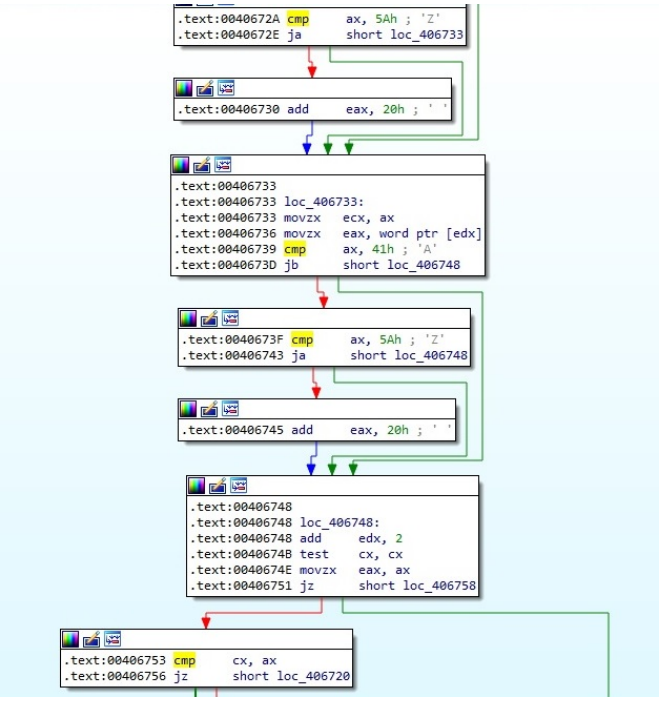


Figure 54

Any targeted process found is killed using the TerminateProcess routine:

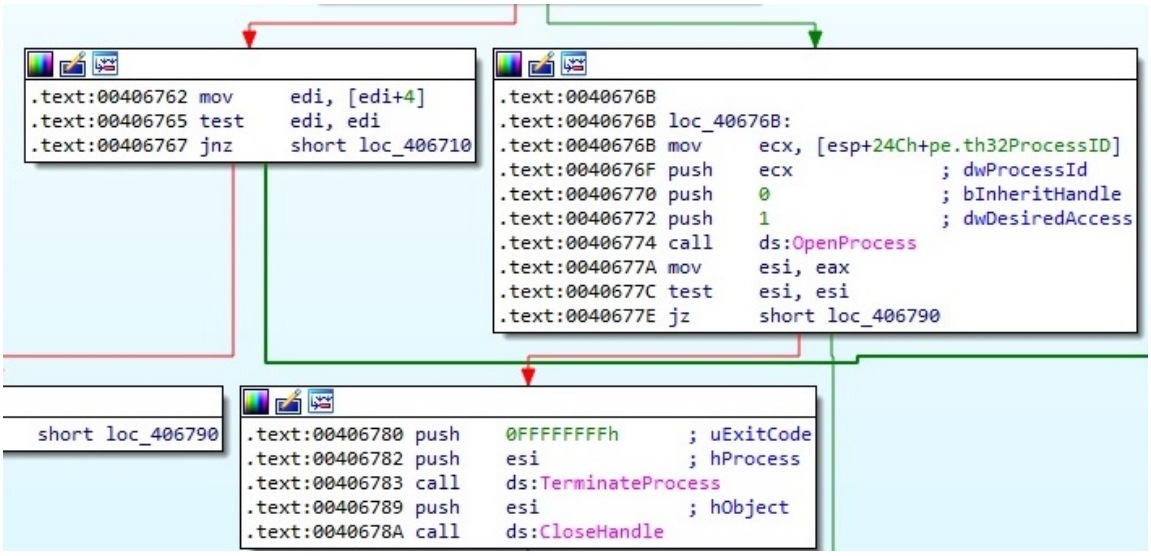


Figure 55

The following extension will be appended to the encrypted files (note the "hash" of the Product ID):

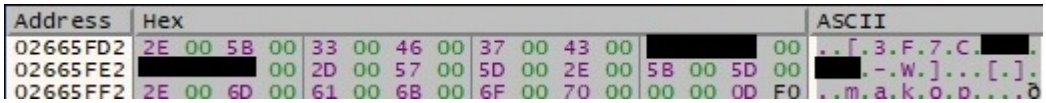


Figure 56

A new thread is created to encrypt the current directory. Similar threads will be created to encrypt the "C:\", "C:\ProgramData" and "C:\Users" directories:

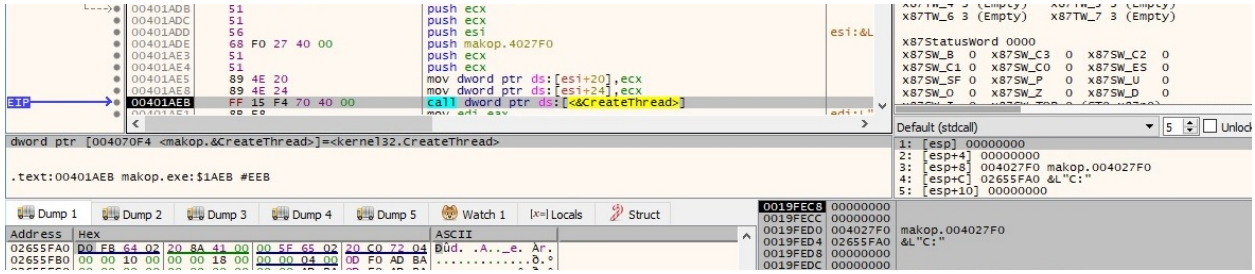


Figure 57

It's important to mention that the following folders will not be encrypted: "C:\WINDOWS", "C:\ProgramData\microsoft\windows\caches", "C:\Users\All Users\Microsoft\Windows\Caches" and "C:\Users\Public". Also, the process doesn't target directories that contain "windows" or "winnt" in their names.

THREAD ACTIVITY – START ADDRESS FUNCTION

The files are enumerated using the FindFirstFileW and FindNextFileW APIs:

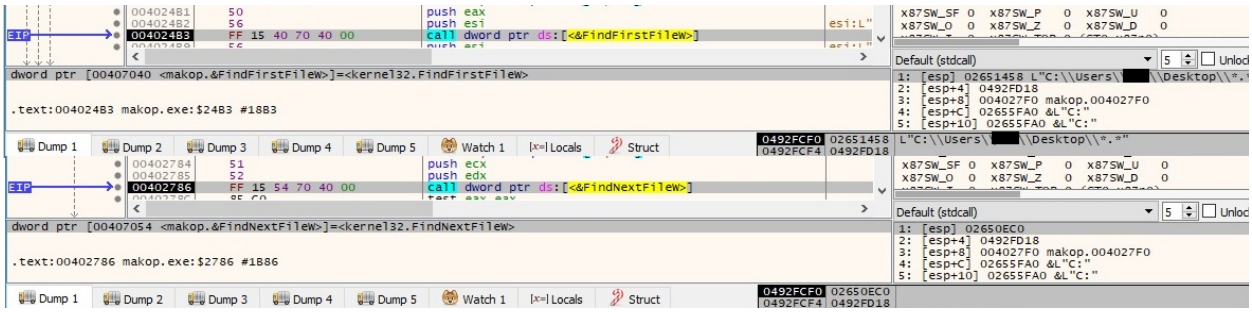


Figure 58

The files that have the following extensions will be skipped: "makop", "CARLOS", "shootlock", "shootlock2", "1recoesufV8Sv6g", "1recocr8M4YJskJ7", "btc", "KJHslgjkjdfg", "origami", "tomas", "RAGA", "zbw", "fireee", "XXX", "element", "HELP", "zes", "lockbit", "captcha", "gunga", "fair", "SOS", "Boss", "moloch", "BKGHJ", "WKSGJ", "termit", "BBC", "dark", "id2020", "arch", "Raf", "ryan", "zxz", "XXL", "xakepz", "exe", "dll", "sphera", "Lookfornewitguy", "XHAMSTER", "xdqd", "BTCHORSEBORIS", "code". Some of these extensions like shootlock, origami, raga and others are the result of other ransomware infections (Shootlock, Origami and Raga ransomware). The following files are not encrypted by Makop: "boot.ini",

"bootfont.bin", "ntldr", "ntdetect.com", "io.sys", "readme-warning.txt", "desktop.ini". The ransomware opens a file for encryption using the CreateFileW API:

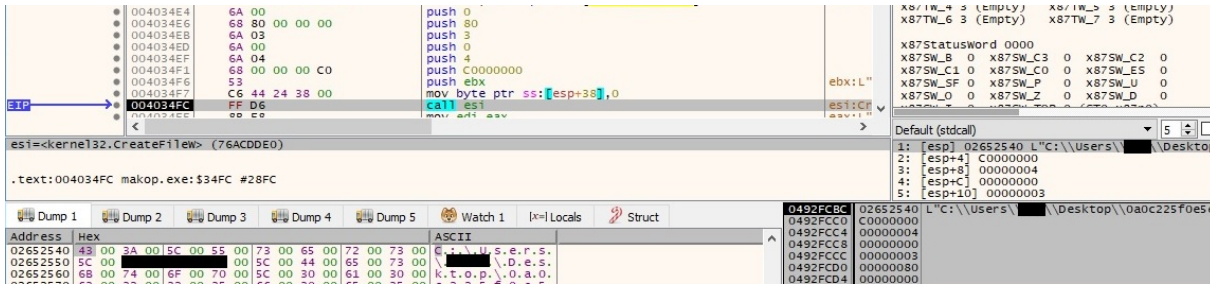


Figure 59

CryptGenRandom is utilized to generate 16 random bytes:

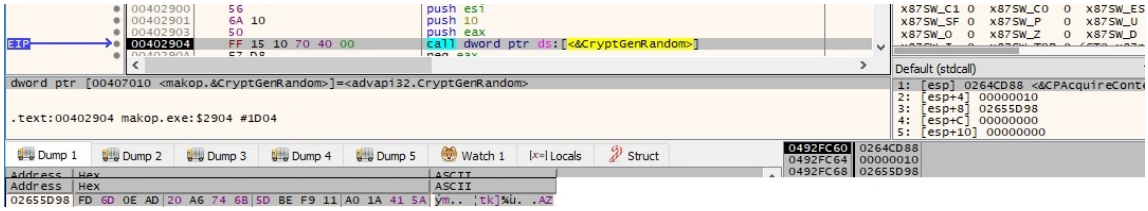


Figure 60

The process imports AESKey1 using the CryptImportKey routine:

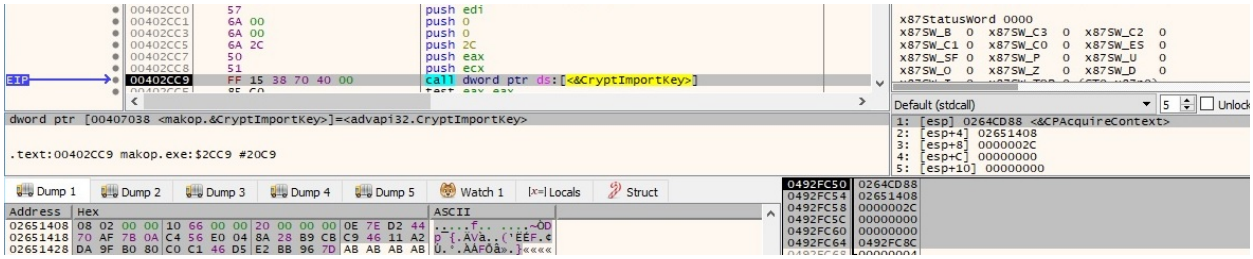


Figure 61

The initialization vector (IV) is set to the 16-byte buffer generated above:

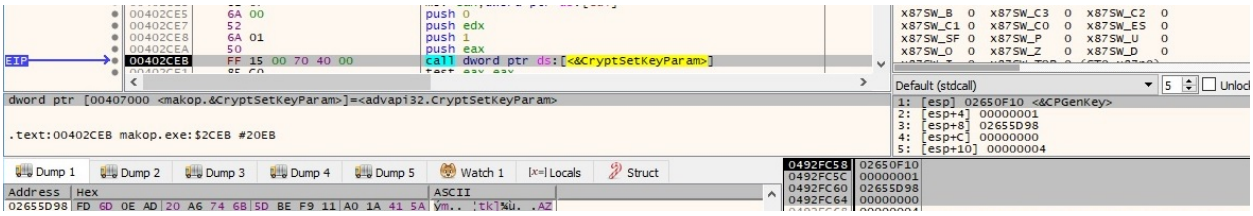


Figure 62

The filename is encrypted using **AESKey1**:

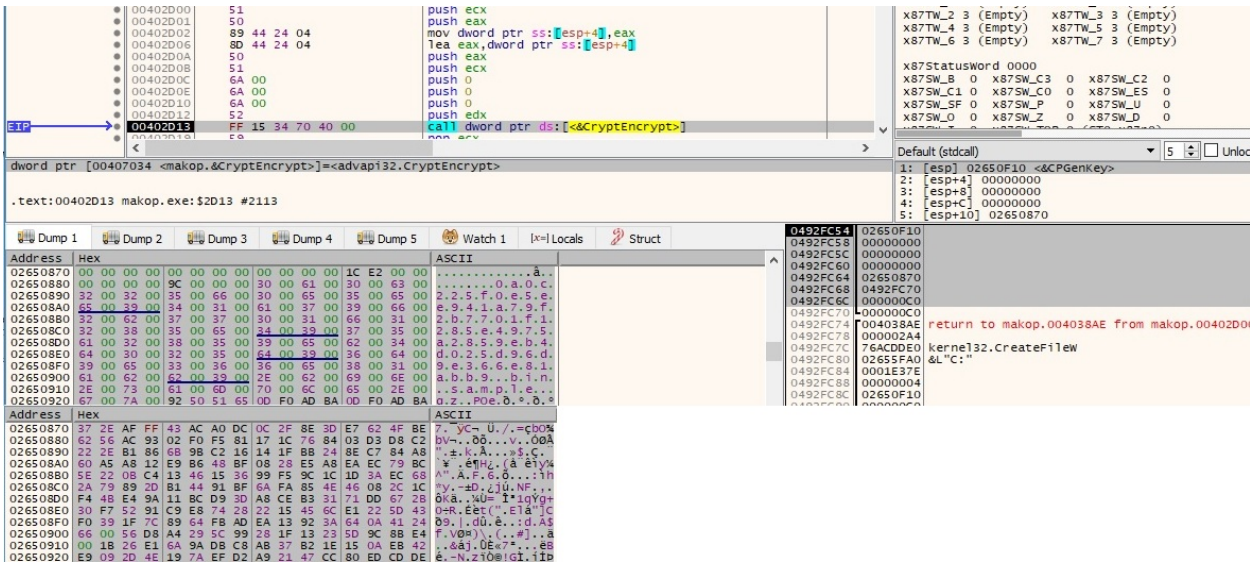


Figure 63

The encrypted filename is written to the file:

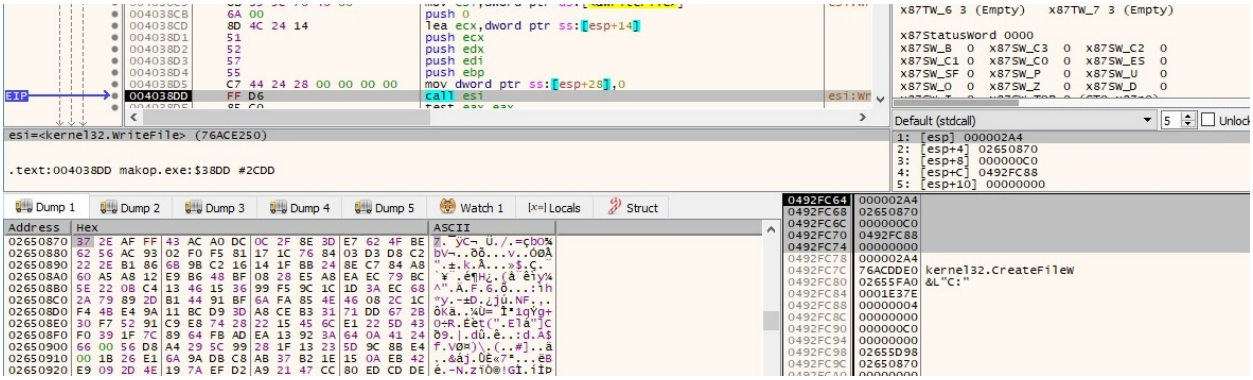


Figure 64

A 4-byte value that represents the encrypted filename size is also written to the file:

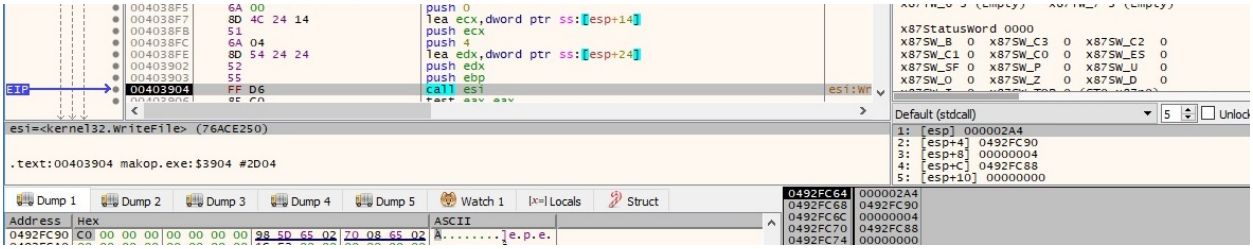


Figure 65

The initialization vector that was generated earlier is added to the encrypted file:

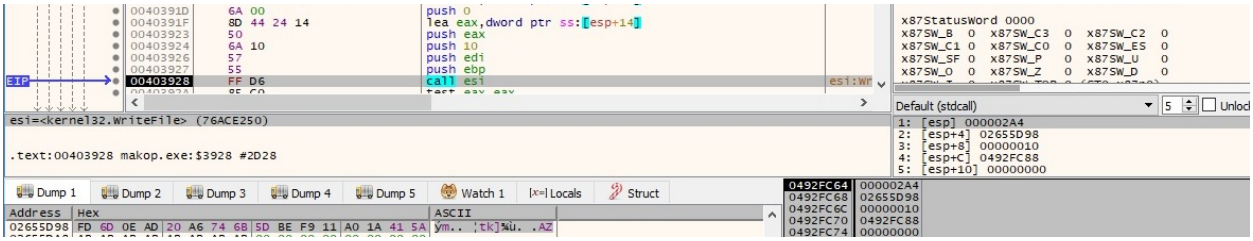


Figure 66

The encrypted buffer that also contains **AESKey1** is written to the file:

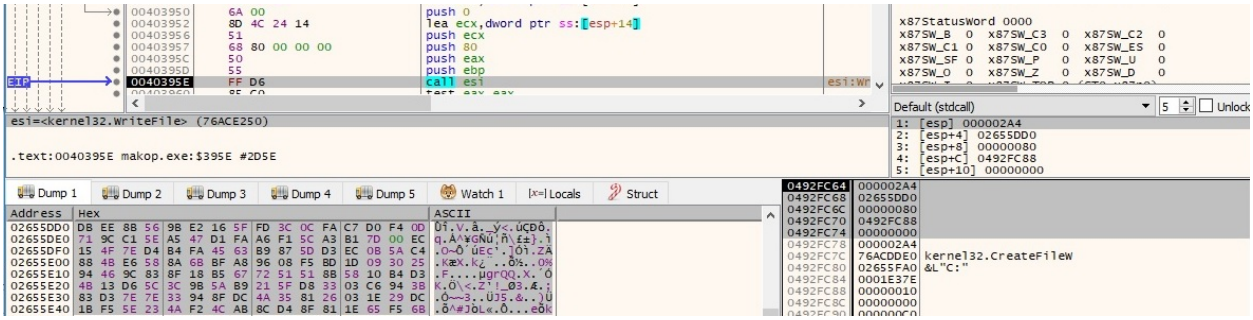


Figure 67

Makop adds 8 NULL bytes after the encrypted content from above:

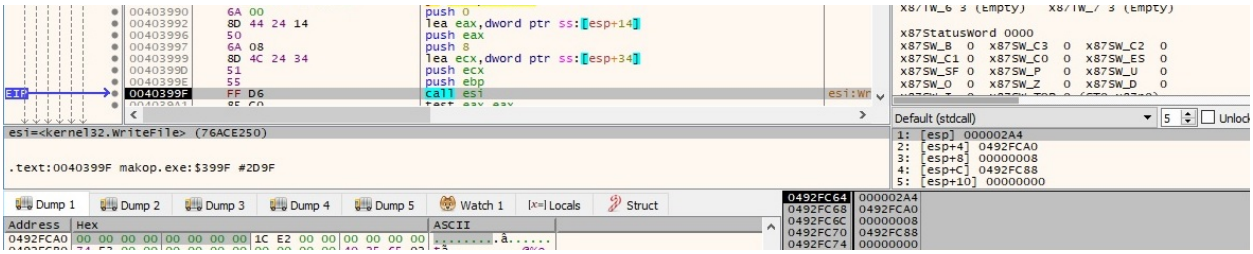


Figure 68

The ransomware reads the content of the file by calling the ReadFile function:

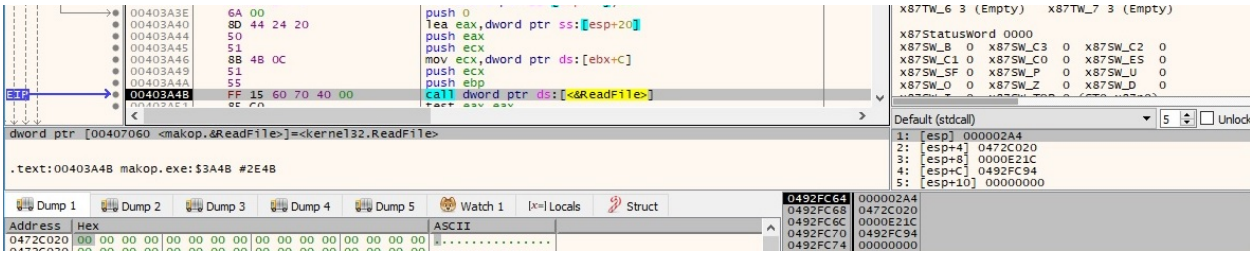


Figure 69

The file content is encrypted using **AESKey1:**

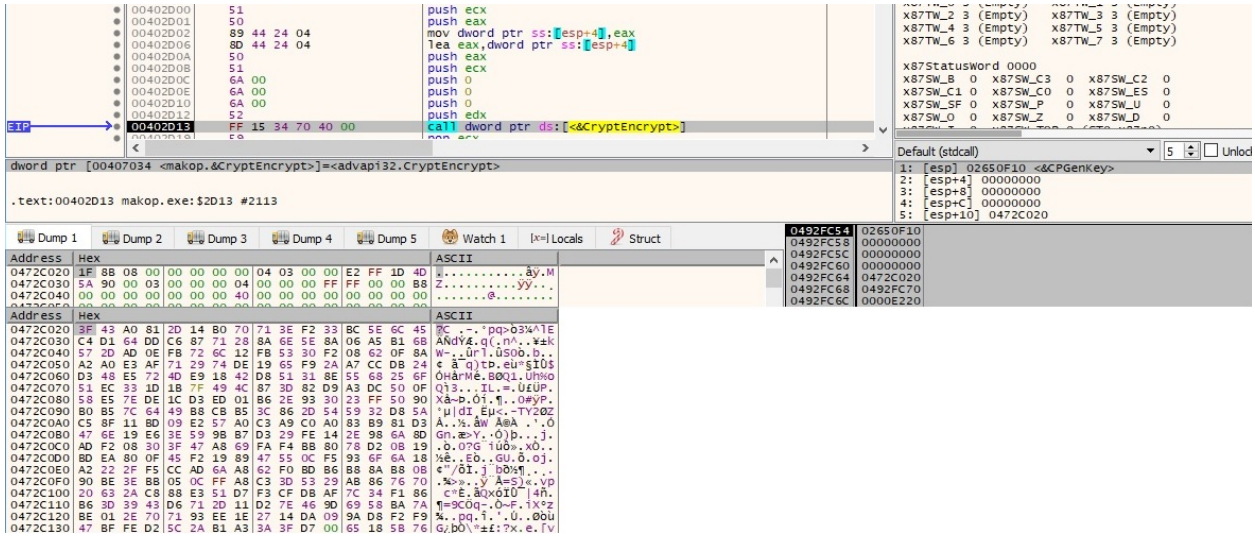


Figure 70

The encrypted content is written to the file:

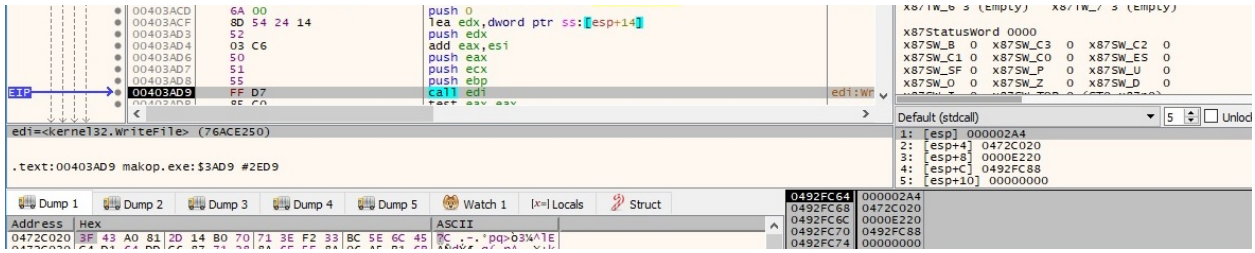


Figure 71

The file extension is changed to show that the file has been encrypted:

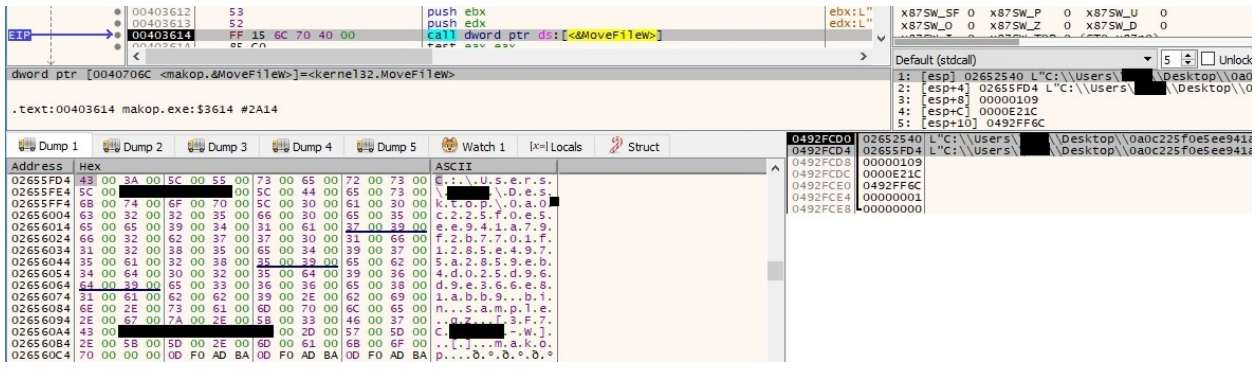
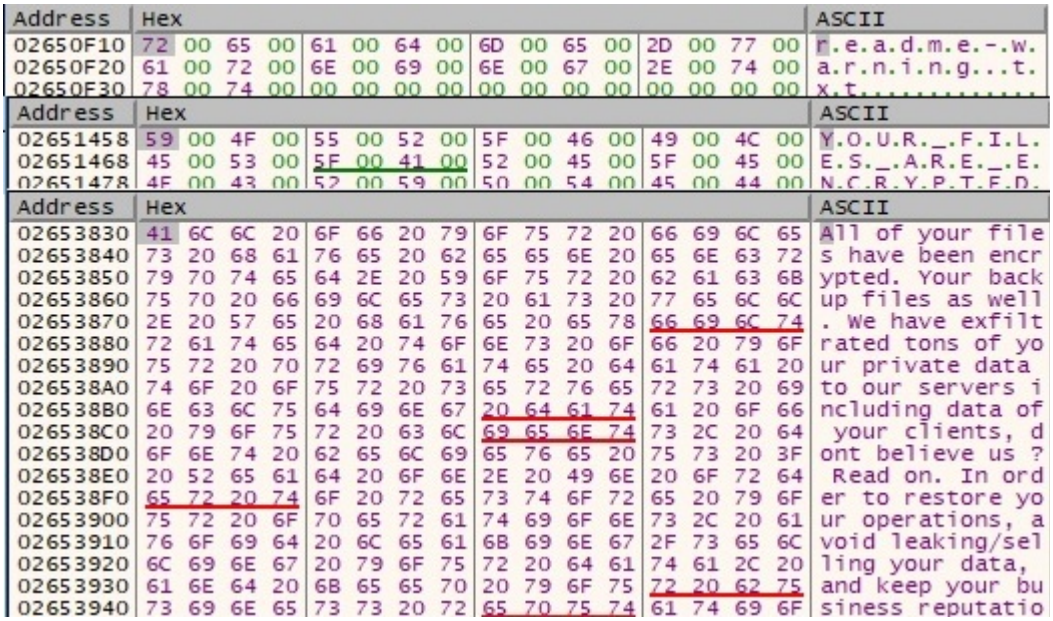


Figure 72

The ransom note name and content are also decrypted at runtime:



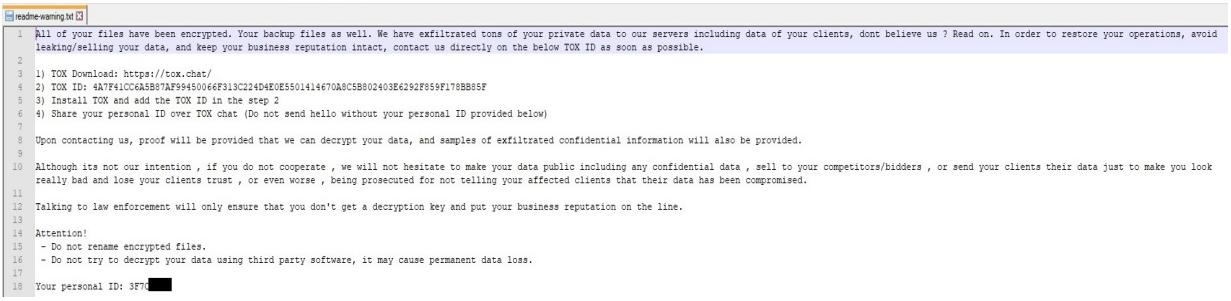
Address	Hex	ASCII
02650F10	72 00 65 00 61 00 64 00 6D 00 65 00 2D 00 77 00	r.e.a.d.m.e.-.w.
02650F20	61 00 72 00 6E 00 69 00 6E 00 67 00 2E 00 74 00	a.r.n.i.n.g...t.
02650F30	78 00 74 00 00 00 00 00 00 00 00 00 00 00 00 00	x.t.............

Address	Hex	ASCII
02651458	59 00 4F 00 55 00 52 00 5F 00 46 00 49 00 4C 00	Y.O.U.R._.F.I.L.
02651468	45 00 53 00 5F 00 41 00 52 00 45 00 5F 00 45 00	E.S._.A.R.E._.E.
02651478	4E 00 43 00 52 00 59 00 50 00 54 00 45 00 44 00	N.C.R.Y.P.T.E.D.

Address	Hex	ASCII
02653830	41 6C 6C 20 6F 66 20 79 6F 75 72 20 66 69 6C 65	All of your file
02653840	73 20 68 61 76 65 20 62 65 65 6E 20 65 6E 63 72	s have been encr
02653850	79 70 74 65 64 2E 20 59 6F 75 72 20 62 61 63 6B	ypted. Your back
02653860	75 70 20 66 69 6C 65 73 20 61 73 20 77 65 6C 6C	up files as well
02653870	2E 20 57 65 20 68 61 76 65 20 65 78 66 69 6C 74	. We have exfilt
02653880	72 61 74 65 64 20 74 6F 6E 73 20 6F 66 20 79 6F	rated tons of yo
02653890	75 72 20 70 72 69 76 61 74 65 20 64 61 74 61 20	ur private data
026538A0	74 6F 20 6F 75 72 20 73 65 72 76 65 72 73 20 69	to our servers i
026538B0	6E 63 6C 75 64 69 6E 67 20 64 61 74 61 20 6F 66	ncluding data of
026538C0	20 79 6F 75 72 20 63 6C 69 65 6E 74 73 2C 20 64	your clients, d
026538D0	6F 6E 74 20 62 65 6C 69 65 76 65 20 75 73 20 3F	ont believe us ?
026538E0	20 52 65 61 64 20 6F 6E 2E 20 49 6E 20 6F 72 64	Read on. In ord
026538F0	65 72 20 74 6F 20 72 65 73 74 6F 72 65 20 79 6F	er to restore yo
02653900	75 72 20 6F 70 65 72 61 74 69 6F 6E 73 2C 20 61	ur operations, a
02653910	76 6F 69 64 20 6C 65 61 6B 69 6E 67 2F 73 65 6C	void leaking/sel
02653920	6C 69 6E 67 20 79 6F 75 72 20 64 61 74 61 2C 20	ling your data,
02653930	61 6E 64 20 6B 65 65 70 20 79 6F 75 72 20 62 75	and keep your bu
02653940	73 69 6E 65 73 73 20 72 65 70 75 74 61 74 69 6F	siness reputatio

Figure 73

The ransom note that contains the personal ID is shown below:



readme-warning.txt

```
All of your files have been encrypted. Your backup files as well. We have exfiltrated tons of your private data to our servers including data of your clients, dont believe us ? Read on. In order to restore your operations, avoid
leaking/selling your data, and keep your business reputation intact, contact us directly on the below TOX ID as soon as possible.

1) TOX Download: https://tox.chat/
2) TOX ID: 4A7F41CC6A5B87AF9945066F313C224D4E0E550141467 0A8C5B802403E6292F859F178BB85F
3) Install TOX and add the TOX ID in the step 2
4) Share your personal ID over TOX chat (Do not send hello without your personal ID provided below)

Upon contacting us, proof will be provided that we can decrypt your data, and samples of exfiltrated confidential information will also be provided.

Although its not our intention , if you do not cooperate , we will not hesitate to make your data public including any confidential data , sell to your competitors/bidders , or send your clients their data just to make you look
really bad and lose your clients trust , or even worse , being prosecuted for not telling your affected clients that their data has been compromised.

Talking to law enforcement will only ensure that you don't get a decryption key and put your business reputation on the line.

Attention!
 - Do not rename encrypted files.
 - Do not try to decrypt your data using third party software, it may cause permanent data loss.

Your personal ID: 3F7C[redacted]
```

Figure 74

It's important to mention that **AESKey1** and **AESKey2** are successively used to encrypt files.

RUNNING WITH "-n" PARAMETER

Makop verifies that the Process ID that comes with the "-n" parameter is composed of digits only:

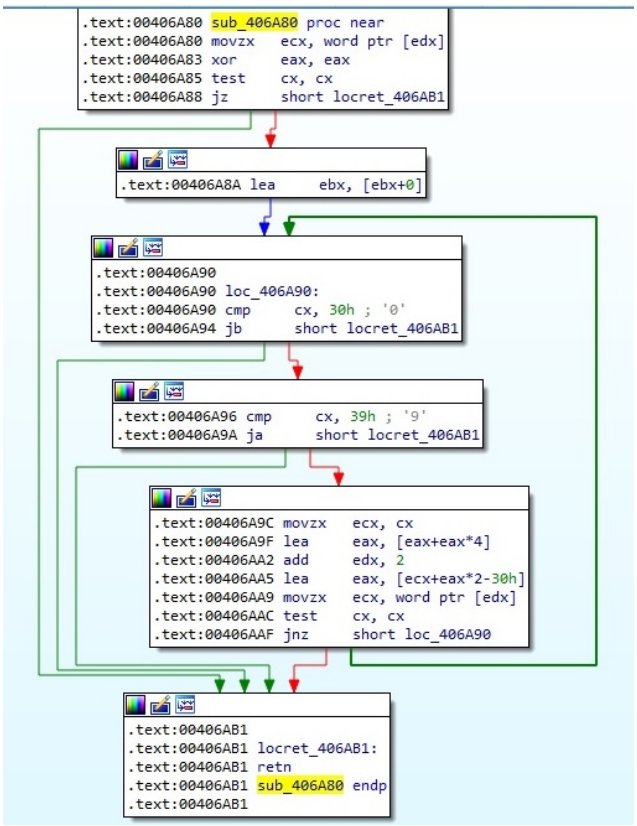


Figure 75

The binary creates a new thread that will enumerate the network resources:

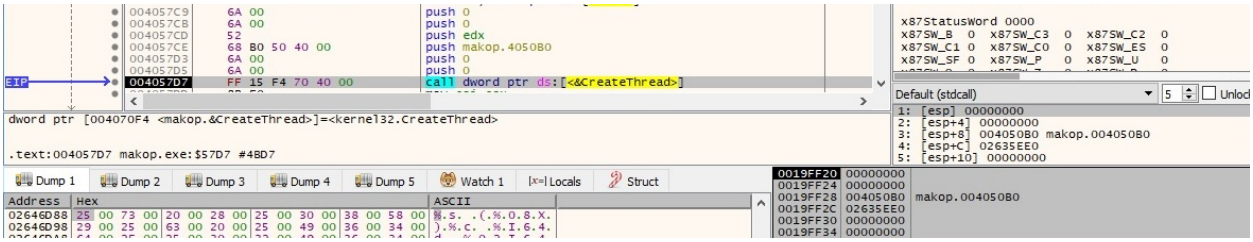


Figure 76

The WNetOpenEnumW and WNetEnumResourceW APIs are used to enumerate the network resources. The malicious executable is looking for network shares that will also be encrypted:

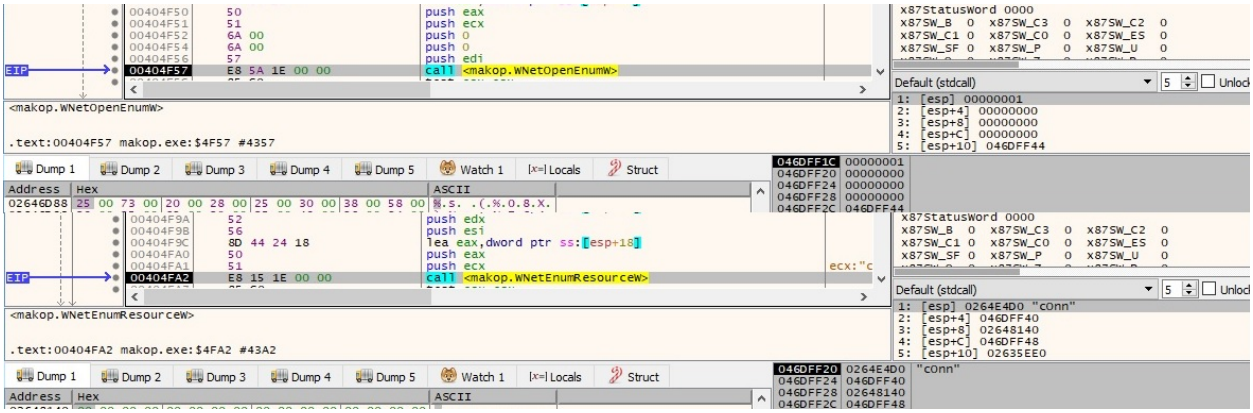


Figure 77

APPENDIX

List of processes to be stopped

msftesql.exe

sqlagent.exe

sqlbrowser.exe

sqlservr.exe

sqlwriter.exe

oracle.exe

ocssd.exe

dbsnmp.exe

synctime.exe

agntsrvc.exe

mydesktopqos.exe

isqlplussvc.exe

xfssvccon.exe

mydesktopservice.exe

ocautoupds.exe

encsvc.exe

firefoxconfig.exe

tbirdconfig.exe

ocomm.exe

mysqld.exe

mysqld-nt.exe

mysqld-opt.exe

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

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

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