Malicious document targets Vietnamese officials

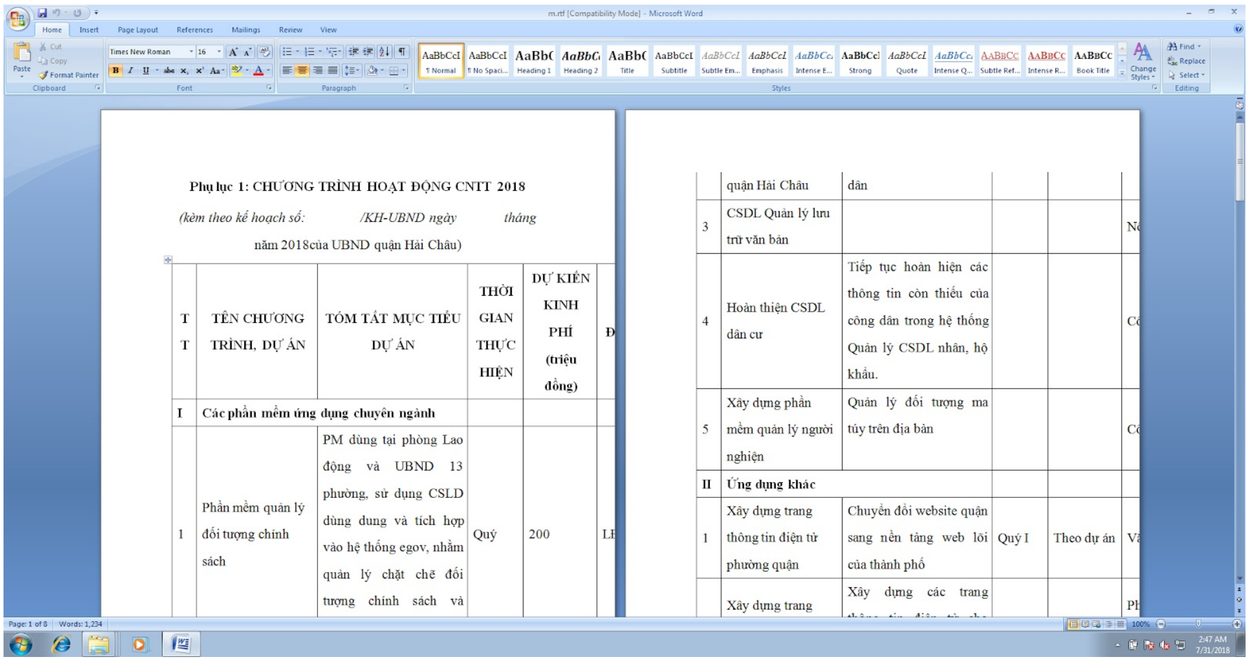
medium.com/@Sebdraven/malicious-document-targets-vietnamese-officials-acb3b9d8b80a

Sebdraven

July 31, 2018

After our investigation of APT SideWinder, we've done a yara rule for hunting RTF document exploiting the CVE-2017–11882.

We found a document written in Vietnamese dealing with a summary about differents projects in the district Hải Châu of Đà Nẵng.



Phụ lục 1: CHƯƠNG TRÌNH HOẠT ĐỘNG CNTT 2018

(kèm theo kế hoạch số: /KH-UBND ngày tháng

năm 2018của UBND quận Hải Châu)

TT	TÊN CHƯƠNG TRÌNH, DỰ ÁN	TÓM TẮT MỤC TIÊU DỰ ÁN	THỜI GIAN THỰC HIỆN	DỰ KIẾN KINH PHÍ (triệu đồng)	Đ
I	**Các phần mềm ứng dụng chuyên ngành**				
1	Phần mềm quản lý đối tượng chính sách	PM dùng tại phòng Lao động và UBND 13 phường, sử dụng CSLD dùng dung và tích hợp vào hệ thống egov, nhằm quản lý chặt chẽ đối tượng chính sách và	Quý	200	L
	quận Hải Châu	dân			
3	CSDL Quản lý lưu trữ văn bản				N
4	Hoàn thiện CSDL dân cư	Tiếp tục hoàn hiện các thông tin còn thiếu của công dân trong hệ thống Quản lý CSDL nhân, hộ khẩu.			C
5	Xây dựng phần mềm quản lý người nghiện	Quản lý đối tượng ma túy trên địa bàn			C
II	**Ứng dụng khác**				
1	Xây dựng trang thông tin điện tử phường quận	Chuyển đổi website quận sang nền tảng web lõi của thành phố	Quý I	Theo dự án	V
	Xây dựng trang	Xây dựng các trang			Ph

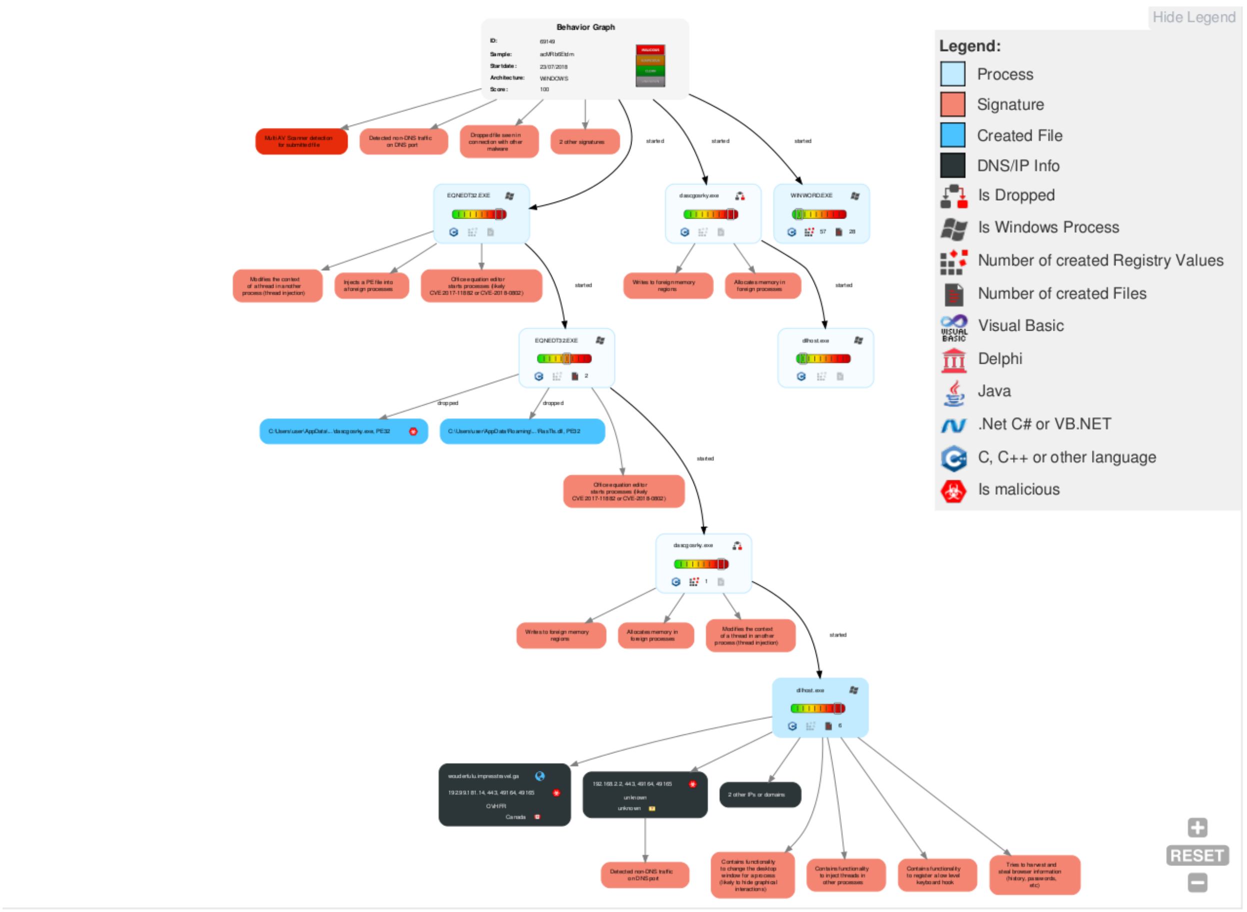
RTF document

In this article, we'll detail the infection chains and the infrastructures of the attackers and the TTPs of this campaign.

The infrastructures and TTPs during this campaign seem to the Chinese hacking group 1937CN.

Infection chains

Joe sandbox has a good representation of the behaviour of the infection.



This rtf document is really malicious and it exploits the equation vulnerability to write two files in the system:

1. A dll named RasTls.dll
2. A executable file named dascgosrky.exe

This document is interesting to analyze so let'go !

RTF analysis

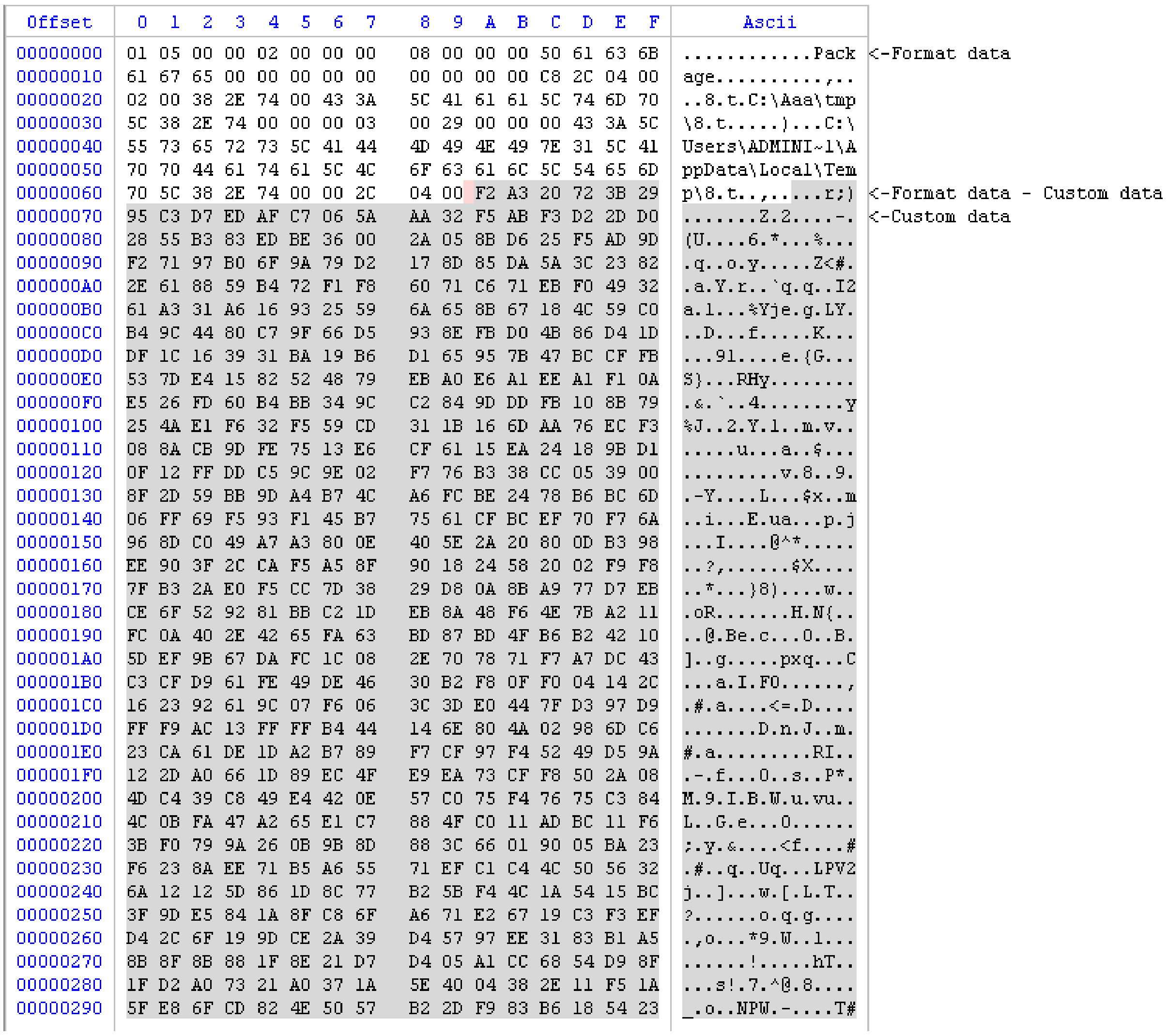
With rtfobj, we found three ole objects in the document:

two non well formed ole object and a third named package ole object.

```
===============================================================================
File: '42162c495e835cdf28670661a53d47d12255d9c791c1c5653673b25fb587ffed' - size: 765121 bytes
---+----------+---------------------------------------------------------------
id |index     |OLE Object
---+----------+---------------------------------------------------------------
0  |000305E8h |format_id: 2 (Embedded)
   |          |class name: 'Package'
   |          |data size: 273608
   |          |OLE Package object:
   |          |Filename: u'8.t'
   |          |Source path: u'C:\\Aaa\\tmp\\8.t'
   |          |Temp path = u'C:\\Users\\ADMINI~1\\AppData\\Local\\Temp\\8.t'
---+----------+---------------------------------------------------------------
1  |000B6010h |Not a well-formed OLE object
---+----------+---------------------------------------------------------------
2  |000B5FFEh |Not a well-formed OLE object
---+----------+---------------------------------------------------------------
```

The package ole object is used to write a file in the disk when the document is opened at the destination described by the ole object.

That's why, there is a path and a name in the ole object.



```
Offset    0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F    Ascii
00000000  01 05 00 00 02 00 00 00  08 00 00 00 50 61 63 6B  ............Pack <-Format data
00000010  61 67 65 00 00 00 00 00  00 00 00 00 C8 2C 04 00  age.........,..
00000020  02 00 38 2E 74 00 43 3A  5C 41 61 61 5C 74 6D 70  ..8.t.C:\Aaa\tmp
00000030  5C 38 2E 74 00 00 00 03  00 29 00 00 00 43 3A 5C  \8.t.....)...C:\
00000040  55 73 65 72 73 5C 41 44  4D 49 4E 49 7E 31 5C 41  Users\ADMINI~1\A
00000050  70 70 44 61 74 61 5C 4C  6F 63 61 6C 5C 54 65 6D  ppData\Local\Tem
00000060  70 5C 38 2E 74 00 00 2C  04 00 F2 A3 20 72 3B 29  p\8.t..,.....r;) <-Format data - Custom data
00000070  95 C3 D7 ED AF C7 06 5A  AA 32 F5 AB F3 D2 2D D0  .......Z.2....-. <-Custom data
00000080  28 55 B3 83 ED BE 36 00  2A 05 8B D6 25 F5 AD 9D  (U....6.*...%...
00000090  F2 71 97 B0 6F 9A 79 D2  17 8D 85 DA 5A 3C 23 82  .q..o.y.....Z<#.
000000A0  2E 61 88 59 B4 72 F1 F8  60 71 C6 71 EB F0 49 32  .a.Y.r..`q.q..I2
000000B0  61 A3 31 A6 16 93 25 59  6A 65 8B 67 18 4C 59 C0  a.1...%Yje.g.LY.
000000C0  B4 9C 44 80 C7 9F 66 D5  93 8E FB D0 4B 86 D4 1D  ..D...f.....K...
000000D0  DF 1C 16 39 31 BA 19 B6  D1 65 95 7B 47 BC CF FB  ...91....e.{G...
000000E0  53 7D E4 15 82 52 48 79  EB A0 E6 A1 EE A1 F1 0A  S}...RHy........
000000F0  E5 26 FD 60 B4 BB 34 9C  C2 84 9D DD FB 10 8B 79  .&.`..4........y
00000100  25 4A E1 F6 32 F5 59 CD  31 1B 16 6D AA 76 EC F3  %J..2.Y.1..m.v..
00000110  08 8A CB 9D FE 75 13 E6  CF 61 15 EA 24 18 9B D1  .....u...a..$...
00000120  0F 12 FF DD C5 9C 9E 02  F7 76 B3 38 CC 05 39 00  .........v.8..9.
00000130  8F 2D 59 BB 9D A4 B7 4C  A6 FC BE 24 78 B6 BC 6D  .-Y....L...$x..m
00000140  06 FF 69 F5 93 F1 45 B7  75 61 CF BC EF 70 F7 6A  ..i...E.ua...p.j
00000150  96 8D C0 49 A7 A3 80 0E  40 5E 2A 20 80 0D B3 98  ...I....@^*.....
00000160  EE 90 3F 2C CA F5 A5 8F  90 18 24 58 20 02 F9 F8  ..?,......$X....
00000170  7F B3 2A E0 F5 CC 7D 38  29 D8 0A 8B A9 77 D7 EB  ..*...}8)....w..
00000180  CE 6F 52 92 81 BB C2 1D  EB 8A 48 F6 4E 7B A2 11  .oR.......H.N{..
00000190  FC 0A 40 2E 42 65 FA 63  BD 87 BD 4F B6 B2 42 10  ..@.Be.c...O..B.
000001A0  5D EF 9B 67 DA FC 1C 08  2E 70 78 71 F7 A7 DC 43  ]..g.....pxq...C
000001B0  C3 CF D9 61 FE 49 DE 46  30 B2 F8 0F F0 04 14 2C  ...a.I.F0......,
000001C0  16 23 92 61 9C 07 F6 06  3C 3D E0 44 7F D3 97 D9  .#.a....<=.D....
000001D0  FF F9 AC 13 FF FF B4 44  14 6E 80 4A 02 98 6D C6  .......D.n.J..m.
000001E0  23 CA 61 DE 1D A2 B7 89  F7 CF 97 F4 52 49 D5 9A  #.a.........RI..
000001F0  12 2D A0 66 1D 89 EC 4F  E9 EA 73 CF F8 50 2A 08  .-.f...O..s..P*.
00000200  4D C4 39 C8 49 E4 42 0E  57 C0 75 F4 76 75 C3 84  M.9.I.B.W.u.vu..
00000210  4C 0B FA 47 A2 65 E1 C7  88 4F C0 11 AD BC 11 F6  L..G.e...O......
00000220  3B F0 79 9A 26 0B 9B 8D  88 3C 66 01 90 05 BA 23  ;.y.&....<f....#
00000230  F6 23 8A EE 71 B5 A6 55  71 EF C1 C4 4C 50 56 32  .#..q..Uq...LPV2
00000240  6A 12 12 5D 86 1D 8C 77  B2 5B F4 4C 1A 54 15 BC  j..]...w.[.L.T..
00000250  3F 9D E5 84 1A 8F C8 6F  A6 71 E2 67 19 C3 F3 EF  ?......o.q.g....
00000260  D4 2C 6F 19 9D CE 2A 39  D4 57 97 EE 31 83 B1 A5  .,o...*9.W..1...
00000270  8B 8F 8B 88 1F 8E 21 D7  D4 05 A1 CC 68 54 D9 8F  ......!.....hT..
00000280  1F D2 A0 73 21 A0 37 1A  5E 40 04 38 2E 11 F5 1A  ...s!.7.^@.8....
00000290  5F E8 6F CD 82 4E 50 57  B2 2D F9 83 B6 18 54 23  _.o..NPW.-....T#
```

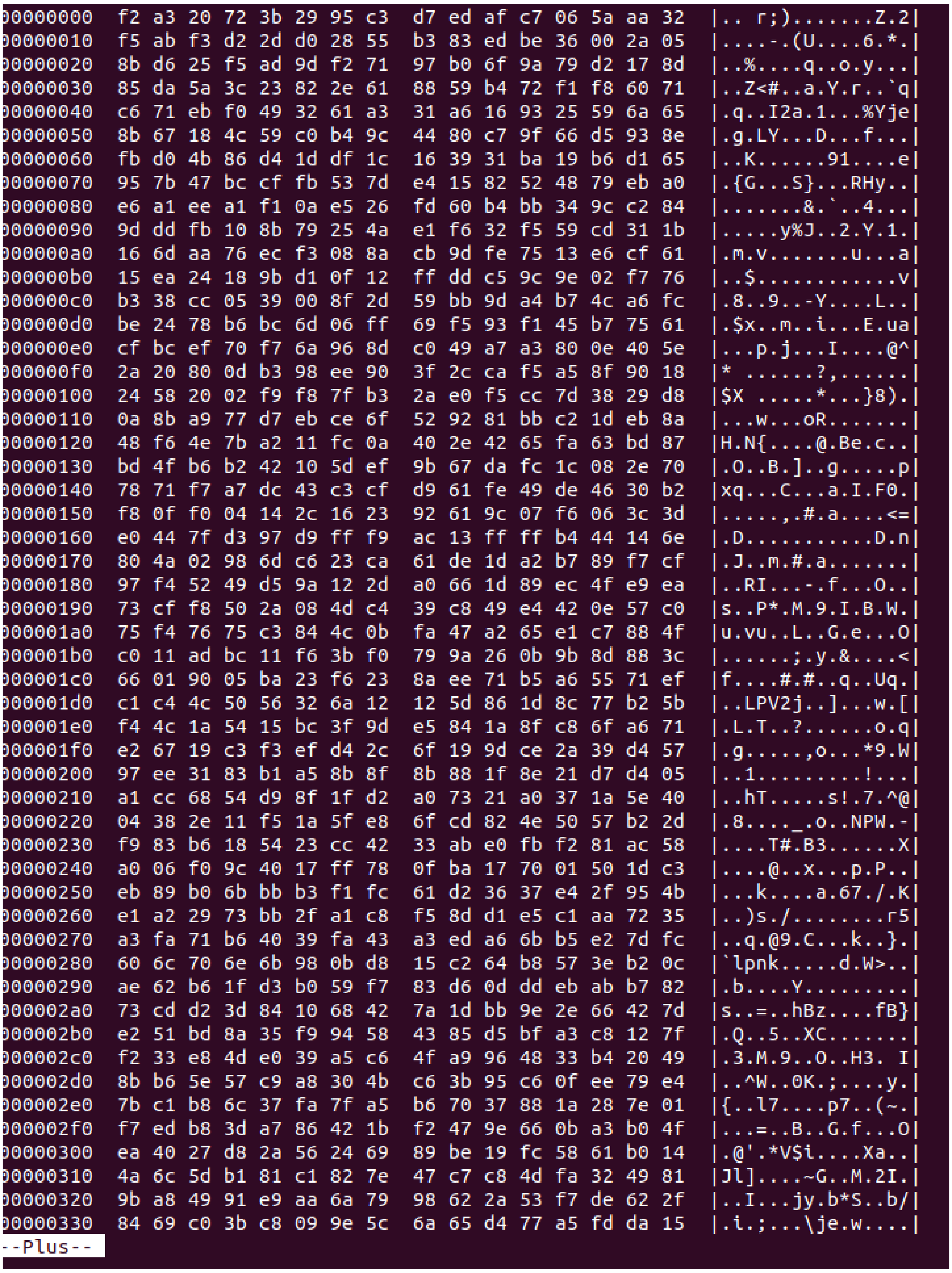
Package OLE Object

This technique is used to execute code like sct file to download an executable on the operating system. McAfee labs has detailed all this stuff with sct file: https://securingtomorrow.mcafee.com/mcafee-labs/dropping-files-temp-folder-raises-security-concerns/

Many attackers use it in the wild because it' very easy to use and it' supported by the office software with RTF files.

So, in our case, a file named 8.t is dropped on %TMP% folder.

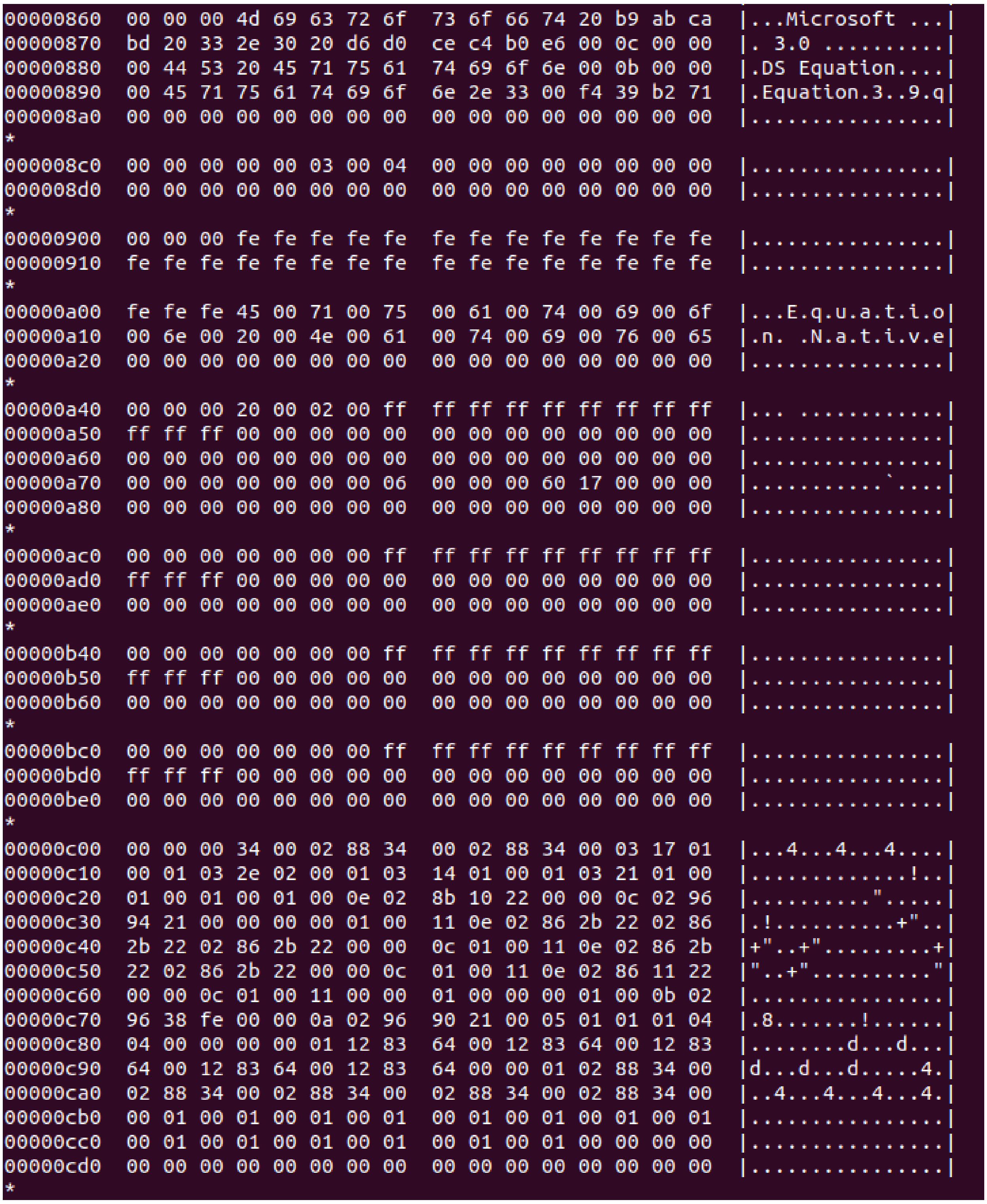
If we check it, it's clearly encrypted.



```
00000000  f2 a3 20 72 3b 29 95 c3  d7 ed af c7 06 5a aa 32  |.. r;).......Z.2|
00000010  f5 ab f3 d2 2d d0 28 55  b3 83 ed be 36 00 2a 05  |....-.(U....6.*.|
00000020  8b d6 25 f5 ad 9d f2 71  97 b0 6f 9a 79 d2 17 8d  |..%....q..o.y...|
00000030  85 da 5a 3c 23 82 2e 61  88 59 b4 72 f1 f8 60 71  |..Z<#..a.Y.r..`q|
00000040  c6 71 eb f0 49 32 61 a3  31 a6 16 93 25 59 6a 65  |.q..I2a.1...%Yje|
00000050  8b 67 18 4c 59 c0 b4 9c  44 80 c7 9f 66 d5 93 8e  |.g.LY...D...f...|
00000060  fb d0 4b 86 d4 1d df 1c  16 39 31 ba 19 b6 d1 65  |..K......91....e|
00000070  95 7b 47 bc cf fb 53 7d  e4 15 82 52 48 79 eb a0  |.{G...S}...RHy..|
00000080  e6 a1 ee a1 f1 0a e5 26  fd 60 b4 bb 34 9c c2 84  |.......&.`..4...|
00000090  9d dd fb 10 8b 79 25 4a  e1 f6 32 f5 59 cd 31 1b  |.....y%J..2.Y.1.|
000000a0  16 6d aa 76 ec f3 08 8a  cb 9d fe 75 13 e6 cf 61  |.m.v.......u...a|
000000b0  15 ea 24 18 9b d1 0f 12  ff dd c5 9c 9e 02 f7 76  |..$............v|
000000c0  b3 38 cc 05 39 00 8f 2d  59 bb 9d a4 b7 4c a6 fc  |.8..9..-Y....L..|
000000d0  be 24 78 b6 bc 6d 06 ff  69 f5 93 f1 45 b7 75 61  |.$x..m..i...E.ua|
000000e0  cf bc ef 70 f7 6a 96 8d  c0 49 a7 a3 80 0e 40 5e  |...p.j...I....@^|
000000f0  2a 20 80 0d b3 98 ee 90  3f 2c ca f5 a5 8f 90 18  |* ......?,......|
00000100  24 58 20 02 f9 f8 7f b3  2a e0 f5 cc 7d 38 29 d8  |$X .....*...}8).|
00000110  0a 8b a9 77 d7 eb ce 6f  52 92 81 bb c2 1d eb 8a  |...w...oR.......|
00000120  48 f6 4e 7b a2 11 fc 0a  40 2e 42 65 fa 63 bd 87  |H.N{....@.Be.c..|
00000130  bd 4f b6 b2 42 10 5d ef  9b 67 da fc 1c 08 2e 70  |.O..B.]..g.....p|
00000140  78 71 f7 a7 dc 43 c3 cf  d9 61 fe 49 de 46 30 b2  |xq...C...a.I.F0.|
00000150  f8 0f f0 04 14 2c 16 23  92 61 9c 07 f6 06 3c 3d  |.....,.#.a....<=|
00000160  e0 44 7f d3 97 d9 ff f9  ac 13 ff ff b4 44 14 6e  |.D...........D.n|
00000170  80 4a 02 98 6d c6 23 ca  61 de 1d a2 b7 89 f7 cf  |.J..m.#.a.......|
00000180  97 f4 52 49 d5 9a 12 2d  a0 66 1d 89 ec 4f e9 ea  |..RI...-.f...O..|
00000190  73 cf f8 50 2a 08 4d c4  39 c8 49 e4 42 0e 57 c0  |s..P*.M.9.I.B.W.|
000001a0  75 f4 76 75 c3 84 4c 0b  fa 47 a2 65 e1 c7 88 4f  |u.vu..L..G.e...O|
000001b0  c0 11 ad bc 11 f6 3b f0  79 9a 26 0b 9b 8d 88 3c  |......;.y.&....<|
000001c0  66 01 90 05 ba 23 f6 23  8a ee 71 b5 a6 55 71 ef  |f....#.#..q..Uq.|
000001d0  c1 c4 4c 50 56 32 6a 12  12 5d 86 1d 8c 77 b2 5b  |..LPV2j..]...w.[|
000001e0  f4 4c 1a 54 15 bc 3f 9d  e5 84 1a 8f c8 6f a6 71  |.L.T..?......o.q|
000001f0  e2 67 19 c3 f3 ef d4 2c  6f 19 9d ce 2a 39 d4 57  |.g.....,o...*9.W|
00000200  97 ee 31 83 b1 a5 8b 8f  8b 88 1f 8e 21 d7 d4 05  |..1.........!...|
00000210  a1 cc 68 54 d9 8f 1f d2  a0 73 21 a0 37 1a 5e 40  |..hT.....s!.7.^@|
00000220  04 38 2e 11 f5 1a 5f e8  6f cd 82 4e 50 57 b2 2d  |.8...._.o..NPW.-|
00000230  f9 83 b6 18 54 23 cc 42  33 ab e0 fb f2 81 ac 58  |....T#.B3......X|
00000240  a0 06 f0 9c 40 17 ff 78  0f ba 17 70 01 50 1d c3  |....@..x...p.P..|
00000250  eb 89 b0 6b bb b3 f1 fc  61 d2 36 37 e4 2f 95 4b  |...k....a.67./.K|
00000260  e1 a2 29 73 bb 2f a1 c8  f5 8d d1 e5 c1 aa 72 35  |..)s./........r5|
00000270  a3 fa 71 b6 40 39 fa 43  a3 ed a6 6b b5 e2 7d fc  |..q.@9.C...k..}.|
00000280  60 6c 70 6e 6b 98 0b d8  15 c2 64 b8 57 3e b2 0c  |`lpnk.....d.W>..|
00000290  ae 62 b6 1f d3 b0 59 f7  83 d6 0d dd eb ab b7 82  |.b....Y.........|
000002a0  73 cd d2 3d 84 10 68 42  7a 1d bb 9e 2e 66 42 7d  |s..=..hBz....fB}|
000002b0  e2 51 bd 8a 35 f9 94 58  43 85 d5 bf a3 c8 12 7f  |.Q..5..XC.......|
000002c0  f2 33 e8 4d e0 39 a5 c6  4f a9 96 48 33 b4 20 49  |.3.M.9..O..H3. I|
000002d0  8b b6 5e 57 c9 a8 30 4b  c6 3b 95 c6 0f ee 79 e4  |..^W..0K.;....y.|
000002e0  7b c1 b8 6c 37 fa 7f a5  b6 70 37 88 1a 28 7e 01  |{..l7....p7..(~.|
000002f0  f7 ed b8 3d a7 86 42 1b  f2 47 9e 66 0b a3 b0 4f  |...=..B..G.f...O|
00000300  ea 40 27 d8 2a 56 24 69  89 be 19 fc 58 61 b0 14  |.@'.*V$i....Xa..|
00000310  4a 6c 5d b1 81 c1 82 7e  47 c7 c8 4d fa 32 49 81  |Jl]....~G..M.2I.|
00000320  9b a8 49 91 e9 aa 6a 79  98 62 2a 53 f7 de 62 2f  |..I...jy.b*S..b/|
00000330  84 69 c0 3b c8 09 9e 5c  6a 65 d4 77 a5 fd da 15  |.i.;...\je.w....|
--Plus--
```

8.t encrypted

The others object ole seem to the exploit of CVE-2017–11882.

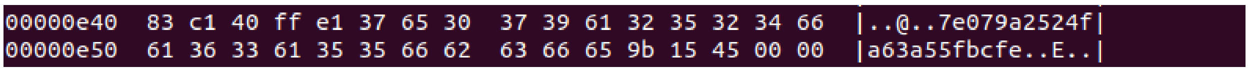


```
00000860  00 00 00 4d 69 63 72 6f  73 6f 66 74 20 b9 ab ca  |...Microsoft ...|
00000870  bd 20 33 2e 30 20 d6 d0  ce c4 b0 e6 00 0c 00 00  |. 3.0 ..........|
00000880  00 44 53 20 45 71 75 61  74 69 6f 6e 00 0b 00 00  |.DS Equation....|
00000890  00 45 71 75 61 74 69 6f  6e 2e 33 00 f4 39 b2 71  |.Equation.3..9.q|
000008a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
000008c0  00 00 00 00 00 03 00 04  00 00 00 00 00 00 00 00  |................|
000008d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000900  00 00 00 fe fe fe fe fe  fe fe fe fe fe fe fe fe  |................|
00000910  fe fe fe fe fe fe fe fe  fe fe fe fe fe fe fe fe  |................|
*
00000a00  fe fe fe 45 00 71 00 75  00 61 00 74 00 69 00 6f  |...E.q.u.a.t.i.o|
00000a10  00 6e 00 20 00 4e 00 61  00 74 00 69 00 76 00 65  |.n. .N.a.t.i.v.e|
00000a20  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000a40  00 00 00 20 00 02 00 ff  ff ff ff ff ff ff ff ff  |... ............|
00000a50  ff ff ff 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000a60  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000a70  00 00 00 00 00 00 00 06  00 00 00 60 17 00 00 00  |...........`....|
00000a80  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000ac0  00 00 00 00 00 00 00 ff  ff ff ff ff ff ff ff ff  |................|
00000ad0  ff ff ff 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000ae0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000b40  00 00 00 00 00 00 00 ff  ff ff ff ff ff ff ff ff  |................|
00000b50  ff ff ff 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000b60  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000bc0  00 00 00 00 00 00 00 ff  ff ff ff ff ff ff ff ff  |................|
00000bd0  ff ff ff 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000be0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000c00  00 00 00 34 00 02 88 34  00 02 88 34 00 03 17 01  |...4...4...4....|
00000c10  00 01 03 2e 02 00 01 03  14 01 00 01 03 21 01 00  |.............!..|
00000c20  01 00 01 00 01 00 0e 02  8b 10 22 00 00 0c 02 96  |..........".....|
00000c30  94 21 00 00 00 00 01 00  11 0e 02 86 2b 22 02 86  |.!..........+"..|
00000c40  2b 22 02 86 2b 22 00 00  0c 01 00 11 0e 02 86 2b  |+"..+".........+|
00000c50  22 02 86 2b 22 00 00 0c  01 00 11 0e 02 86 11 22  |"..+".........."|
00000c60  00 00 0c 01 00 11 00 00  01 00 00 00 01 00 0b 02  |................|
00000c70  96 38 fe 00 00 0a 02 96  90 21 00 05 01 01 01 04  |.8.......!......|
00000c80  04 00 00 00 00 01 12 83  64 00 12 83 64 00 12 83  |........d...d...|
00000c90  64 00 12 83 64 00 12 83  64 00 00 01 02 88 34 00  |d...d...d.....4.|
00000ca0  02 88 34 00 02 88 34 00  02 88 34 00 02 88 34 00  |..4...4...4...4.|
00000cb0  00 01 00 01 00 01 00 01  00 01 00 01 00 01 00 01  |................|
00000cc0  00 01 00 01 00 01 00 01  00 01 00 01 00 00 00 00  |................|
00000cd0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
```

Equation Ole Object

At the end of the object ole, we have differents API functions to make a runPE.

Another interesting thing is this string at the begin of the object: 7e079a2524fa63a55fbcfe



```
00000e40  83 c1 40 ff e1 37 65 30  37 39 61 32 35 32 34 66  |..@..7e079a2524f|
00000e50  61 36 33 61 35 35 66 62  63 66 65 9b 15 45 00 00  |a63a55fbcfe..E..|
```

String found in many exploits of CVE-2017–11882

We have the same string used by APT SideWinder in the equation object ole.

It’s the same toolset to create the malicious document.

So now, we have to debug the malicious document to find how the file 8.t is used and find this runPE.

Debugging of the shellcode

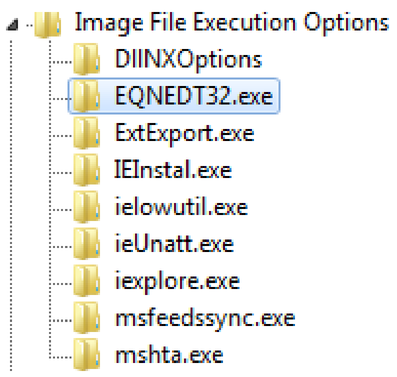
At the start of the analysis, we think the process EQNEDT32.exe is created by Winword.exe using the function CreateProcess. So we decided to set a breakpoint at the call of his function.

But EQNEDT32.exe is invoked by Winword.exe using COM Object. It’s not CreateProcess that used and Winword.exe is not the parent process of EQNEDT32.exe. So we have to attach the debugger when EQNEDT32.exe is launched.

For that, we used a technique named Image File Execution Options that was documented by Microsoft. https://blogs.msdn.microsoft.com/mithuns/2010/03/24/image-file-execution-options-ifeo/

We create a key EQNEDT32.exe.

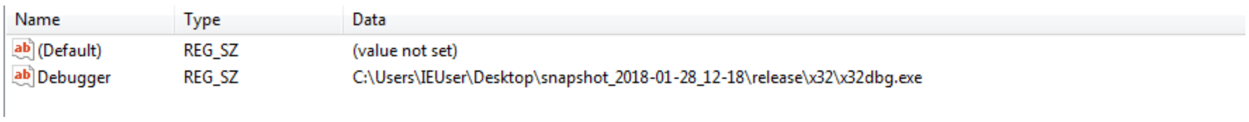
Registry



```
Image File Execution Options
    DllNXOptions
    EQNEDT32.exe
    ExtExport.exe
    IEInstal.exe
    ielowutil.exe
    ieUnatt.exe
    iexplore.exe
    msfeedssync.exe
    mshta.exe
```

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options

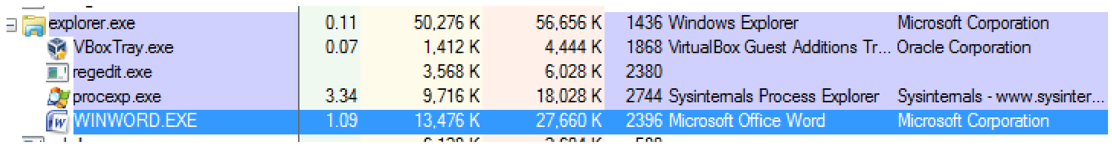
And we set a value string for launching the debugger when EQNEDT32.exe is executed and attaching the debugger to the process .



Name	Type	Data
(Default)	REG_SZ	(value not set)
Debugger	REG_SZ	C:\Users\IEUser\Desktop\snapshot_2018-01-28_12-18\release\x32\x32dbg.exe

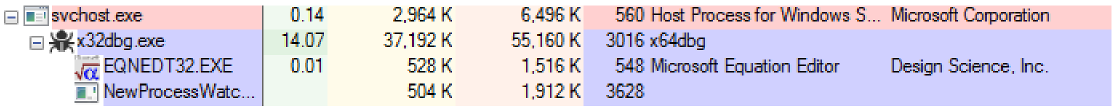
Value to set the debuuger when EQNEDT32.exe is executed

When we open the rtf document, Winword is launched and EQNEDT32.exe also.



explorer.exe	0.11	50,276 K	56,656 K	1436 Windows Explorer	Microsoft Corporation
VBoxTray.exe	0.07	1,412 K	4,444 K	1868 VirtualBox Guest Additions Tr...	Oracle Corporation
regedit.exe		3,568 K	6,028 K	2380	
procexp.exe	3.34	9,716 K	18,028 K	2744 Sysinternals Process Explorer	Sysinternals - www.sysinter...
WINWORD.EXE	1.09	13,476 K	27,660 K	2396 Microsoft Office Word	Microsoft Corporation

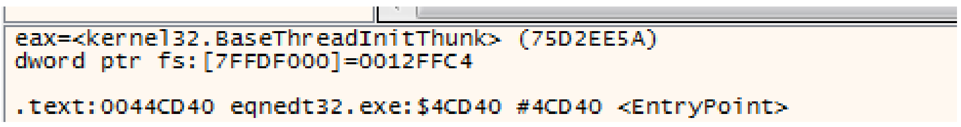
Winword process



```
svchost.exe          0.14    2,964 K    6,496 K   560 Host Process for Windows S...  Microsoft Corporation
  x32dbg.exe        14.07   37,192 K   55,160 K  3016 x64dbg
    EQNEDT32.EXE     0.01      528 K    1,516 K   548 Microsoft Equation Editor      Design Science, Inc.
    NewProcessWatc...          504 K    1,912 K  3628
```

EQNEDT32.exe process attached by the debugger

And the debugger is attached at the entrypoint of EQNEDT32.exe.

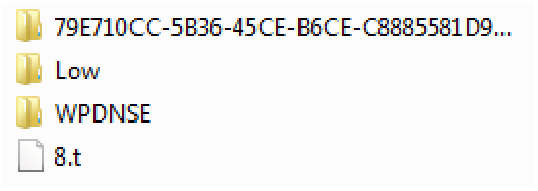


```
eax=<kernel32.BaseThreadInitThunk> (75D2EE5A)
dword ptr fs:[7FFDF000]=0012FFC4

.text:0044CD40 eqnedt32.exe:$4CD40 #4CD40 <EntryPoint>
```

We check if it's 8.t is correctly created in the %TMP% folder.

8.t dropped on disk

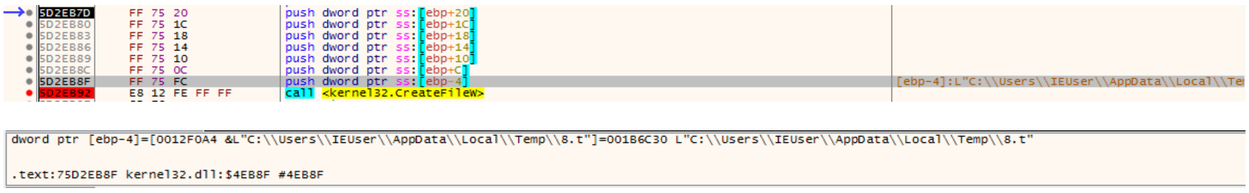
Now we set a breakpoint at the createFile to check if the shellcode of the exploit reads the file 8.t.



```
79E710CC-5B36-45CE-B6CE-C8885581D9...
Low
WPDNSE
8.t
```

CreateFile is called at call eqnedt32.41E5EE.

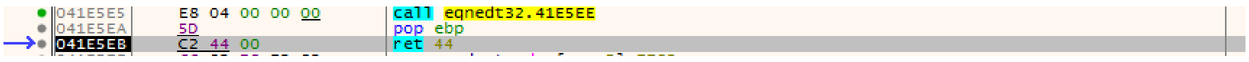
The param of the path of file is pushed on the stack push dword ptr ss:[ebp-4].



```
5D2EB7D   FF 75 20        push dword ptr ss:[ebp+20]
5D2EB80   FF 75 1C        push dword ptr ss:[ebp+1C]
5D2EB83   FF 75 18        push dword ptr ss:[ebp+18]
5D2EB86   FF 75 14        push dword ptr ss:[ebp+14]
5D2EB89   FF 75 10        push dword ptr ss:[ebp+10]
5D2EB8C   FF 75 0C        push dword ptr ss:[ebp+C]
5D2EB8F   FF 75 FC        push dword ptr ss:[ebp-4]      [ebp-4]:L"C:\\Users\\IEUser\\AppData\\Local\\Te
5D2EB92   E8 12 FE FF FF  call <kernel32.CreateFileW>

dword ptr [ebp-4]=[0012F0A4 &L"C:\\Users\\IEUser\\AppData\\Local\\Temp\\8.t"]=001B6C30 L"C:\\Users\\IEUser\\AppData\\Local\\Temp\\8.t"

.text:75D2EB8F kernel32.dll:$4EB8F #4EB8F
```

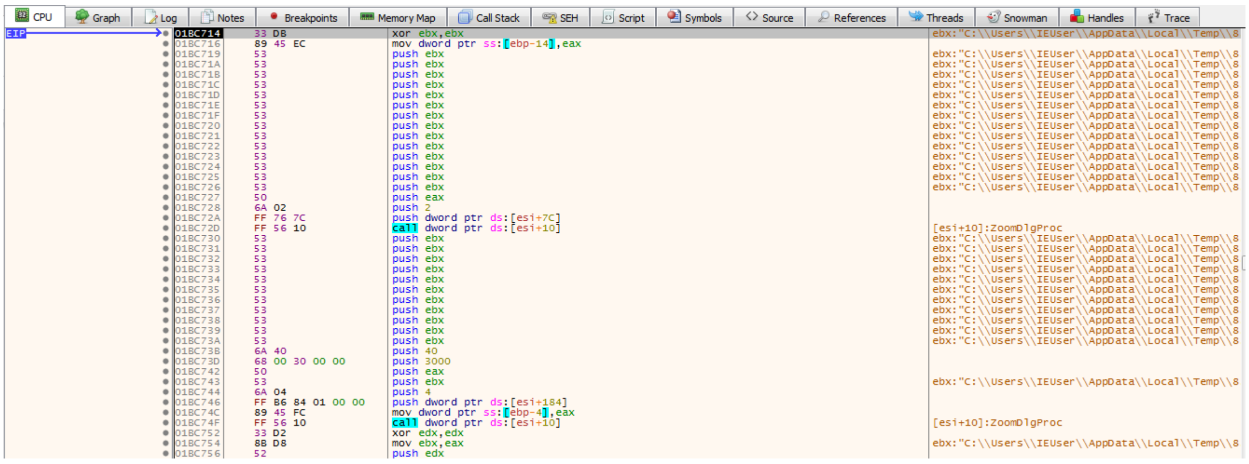
The shellcode uses CreateFile to the 8.t in the %TMP% folder

So now, we can return of the user code at the calling function.



```
041E5E5   E8 04 00 00 00   call eqnedt32.41E5EE
041E5EA   5D               pop ebp
041E5EB   C2 44 00         ret 44
```

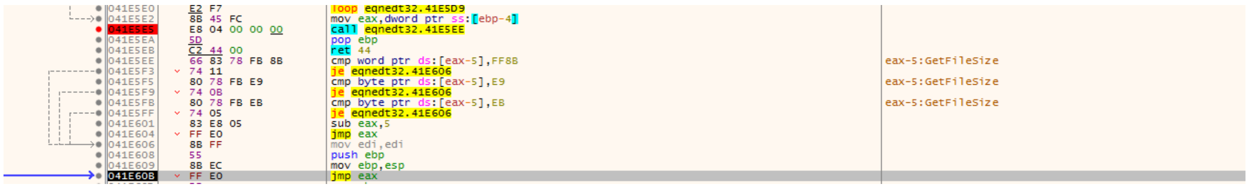
After a step into, we enter in the shellcode, the address space has changed:



```
EIP  01BC714   33 DB               xor ebx,ebx                      ebx:"C:\\Users\\IEUser\\AppData\\Local\\Temp\\8
     01BC716   89 45 EC            mov dword ptr ss:[ebp-14],eax
     01BC719   53                  push ebx                         ebx:"C:\\Users\\IEUser\\AppData\\Local\\Temp\\8
     01BC71A   53                  push ebx                         ebx:"C:\\Users\\IEUser\\AppData\\Local\\Temp\\8
     01BC71B   53                  push ebx                         ebx:"C:\\Users\\IEUser\\AppData\\Local\\Temp\\8
     01BC71C   53                  push ebx                         ebx:"C:\\Users\\IEUser\\AppData\\Local\\Temp\\8
     01BC71D   53                  push ebx                         ebx:"C:\\Users\\IEUser\\AppData\\Local\\Temp\\8
     01BC71E   53                  push ebx                         ebx:"C:\\Users\\IEUser\\AppData\\Local\\Temp\\8
     01BC71F   53                  push ebx                         ebx:"C:\\Users\\IEUser\\AppData\\Local\\Temp\\8
     01BC720   53                  push ebx                         ebx:"C:\\Users\\IEUser\\AppData\\Local\\Temp\\8
     01BC721   53                  push ebx                         ebx:"C:\\Users\\IEUser\\AppData\\Local\\Temp\\8
     01BC722   53                  push ebx                         ebx:"C:\\Users\\IEUser\\AppData\\Local\\Temp\\8
     01BC723   53                  push ebx                         ebx:"C:\\Users\\IEUser\\AppData\\Local\\Temp\\8
     01BC724   53                  push ebx                         ebx:"C:\\Users\\IEUser\\AppData\\Local\\Temp\\8
     01BC725   53                  push ebx                         ebx:"C:\\Users\\IEUser\\AppData\\Local\\Temp\\8
     01BC726   53                  push ebx                         ebx:"C:\\Users\\IEUser\\AppData\\Local\\Temp\\8
     01BC727   50                  push eax
     01BC728   6A 02               push 2
     01BC72A   FF 76 7C            push dword ptr ds:[esi+7C]
     01BC72D   FF 56 10            call dword ptr ds:[esi+10]       [esi+10]:ZoomDlgProc
     01BC730   53                  push ebx                         ebx:"C:\\Users\\IEUser\\AppData\\Local\\Temp\\8
     01BC731   53                  push ebx                         ebx:"C:\\Users\\IEUser\\AppData\\Local\\Temp\\8
     01BC732   53                  push ebx                         ebx:"C:\\Users\\IEUser\\AppData\\Local\\Temp\\8
     01BC733   53                  push ebx                         ebx:"C:\\Users\\IEUser\\AppData\\Local\\Temp\\8
     01BC734   53                  push ebx                         ebx:"C:\\Users\\IEUser\\AppData\\Local\\Temp\\8
     01BC735   53                  push ebx                         ebx:"C:\\Users\\IEUser\\AppData\\Local\\Temp\\8
     01BC736   53                  push ebx                         ebx:"C:\\Users\\IEUser\\AppData\\Local\\Temp\\8
     01BC737   53                  push ebx                         ebx:"C:\\Users\\IEUser\\AppData\\Local\\Temp\\8
     01BC738   53                  push ebx                         ebx:"C:\\Users\\IEUser\\AppData\\Local\\Temp\\8
     01BC739   53                  push ebx                         ebx:"C:\\Users\\IEUser\\AppData\\Local\\Temp\\8
     01BC73A   53                  push ebx                         ebx:"C:\\Users\\IEUser\\AppData\\Local\\Temp\\8
     01BC73B   6A 40               push 40
     01BC73D   68 00 30 00 00      push 3000
     01BC742   50                  push eax
     01BC743   53                  push ebx                         ebx:"C:\\Users\\IEUser\\AppData\\Local\\Temp\\8
     01BC744   6A 04               push 4
     01BC746   FF B6 84 01 00 00   push dword ptr ds:[esi+184]
     01BC74C   89 45 FC            mov dword ptr ss:[ebp-4],eax
     01BC74F   FF 56 10            call dword ptr ds:[esi+10]       [esi+10]:ZoomDlgProc
     01BC752   33 D2               xor edx,edx
     01BC754   8B D8               mov ebx,eax                      ebx:"C:\\Users\\IEUser\\AppData\\Local\\Temp\\8
     01BC756   52                  push edx
```

Shellcode of the exploit

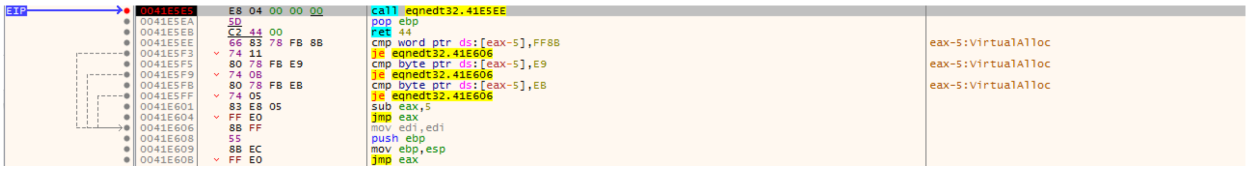
After CreateFile, GetFileSize is called to have the size of the file



```
041E5E0   E2 F7            loop eqnedt32.41E5D9
041E5E2   8B 45 FC         mov eax,dword ptr ss:[ebp-4]
041E5E5   E8 04 00 00 00   call eqnedt32.41E5EE
041E5EA   5D               pop ebp
041E5EB   C2 44 00         ret 44
041E5EE   66 83 78 FB 8B   cmp word ptr ds:[eax-5],FF8B     eax-5:GetFileSize
041E5F3   74 11            je eqnedt32.41E606
041E5F5   80 78 FB E9      cmp byte ptr ds:[eax-5],E9       eax-5:GetFileSize
041E5F9   74 0B            je eqnedt32.41E606
041E5FB   80 78 FB EB      cmp byte ptr ds:[eax-5],EB       eax-5:GetFileSize
041E5FF   74 05            je eqnedt32.41E606
041E601   83 E8 05         sub eax,5
041E604   FF E0            jmp eax
041E606   8B FF            mov edi,edi
041E608   55               push ebp
041E609   8B EC            mov ebp,esp
041E60B   FF E0            jmp eax
```

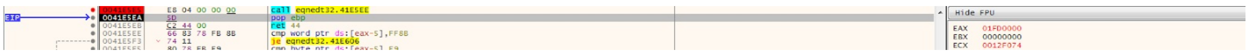
Get the size of the file

After is Virtualloc, and it create a memory page at 1FD0000 (eax value)



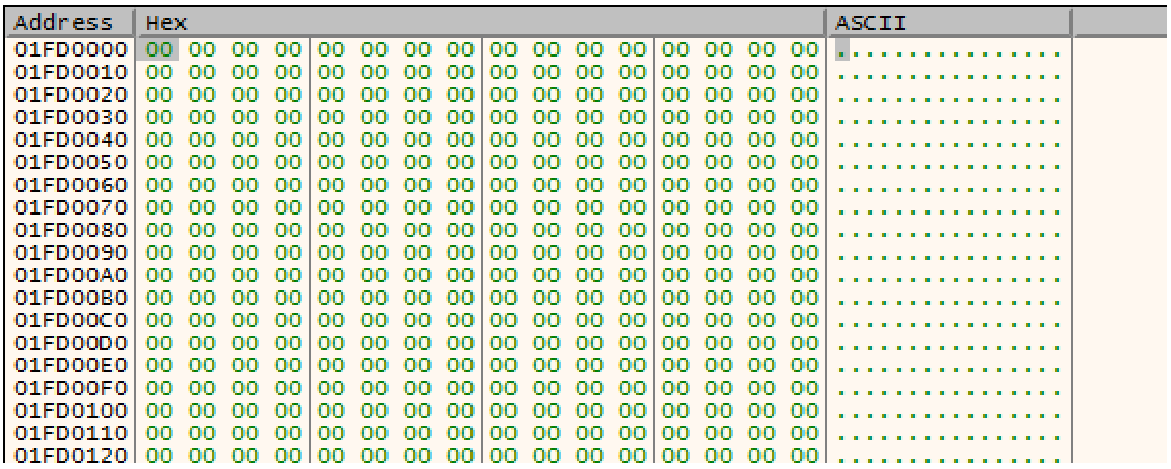
```
EIP  0041E5E5   E8 04 00 00 00   call eqnedt32.41E5EE
     0041E5EA   5D               pop ebp
     0041E5EB   C2 44 00         ret 44
     0041E5EE   66 83 78 FB 8B   cmp word ptr ds:[eax-5],FF8B     eax-5:VirtualAlloc
     0041E5F3   74 11            je eqnedt32.41E606
     0041E5F5   80 78 FB E9      cmp byte ptr ds:[eax-5],E9       eax-5:VirtualAlloc
     0041E5F9   74 0B            je eqnedt32.41E606
     0041E5FB   80 78 FB EB      cmp byte ptr ds:[eax-5],EB       eax-5:VirtualAlloc
     0041E5FF   74 05            je eqnedt32.41E606
     0041E601   83 E8 05         sub eax,5
     0041E604   FF E0            jmp eax
     0041E606   8B FF            mov edi,edi
     0041E608   55               push ebp
     0041E609   8B EC            mov ebp,esp
     0041E60B   FF E0            jmp eax
```

VirtualAlloc memory page to load 8.t



```
     0041E5E5   E8 04 00 00 00   call eqnedt32.41E5EE
EIP  0041E5EA   5D               pop ebp
     0041E5EB   C2 44 00         ret 44
     0041E5EE   66 83 78 FB 8B   cmp word ptr ds:[eax-5],FF8B
     0041E5F3   74 11            je eqnedt32.41E606

Hide FPU
EAX   01FD0000
EBX   00000000
ECX   0012F074
```

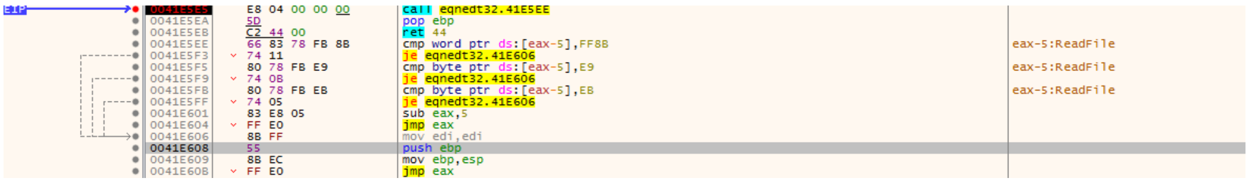
After virtualAlloc, the memory page is pointed by EAX



```
Address   Hex                                              ASCII
01FD0000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01FD0010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01FD0020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01FD0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01FD0040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01FD0050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01FD0060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01FD0070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01FD0080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01FD0090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01FD00A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01FD00B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01FD00C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01FD00D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01FD00E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01FD00F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01FD0100  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01FD0110  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01FD0120  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

The page allocated

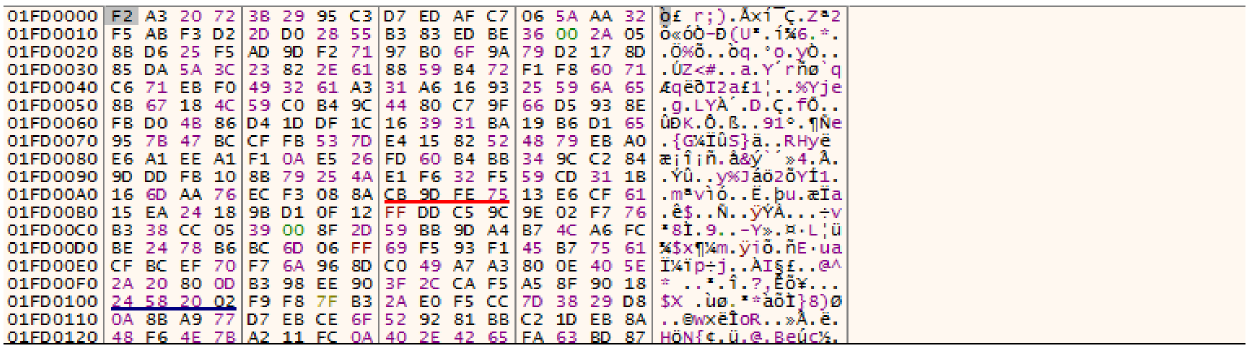
ReadFile is called:



```
EIP  0041E5E5   E8 04 00 00 00   call eqnedt32.41E5EE
     0041E5EA   5D               pop ebp
     0041E5EB   C2 44 00         ret 44
     0041E5EE   66 83 78 FB 8B   cmp word ptr ds:[eax-5],FF8B     eax-5:ReadFile
     0041E5F3   74 11            je eqnedt32.41E606
     0041E5F5   80 78 FB E9      cmp byte ptr ds:[eax-5],E9       eax-5:ReadFile
     0041E5F9   74 0B            je eqnedt32.41E606
     0041E5FB   80 78 FB EB      cmp byte ptr ds:[eax-5],EB       eax-5:ReadFile
     0041E5FF   74 05            je eqnedt32.41E606
     0041E601   83 E8 05         sub eax,5
     0041E604   FF E0            jmp eax
     0041E606   8B FF            mov edi,edi
     0041E608   55               push ebp
     0041E609   8B EC            mov ebp,esp
     0041E60B   FF E0            jmp eax
```

Readfile 8.t

And 8.t is loaded at 1FD0000:



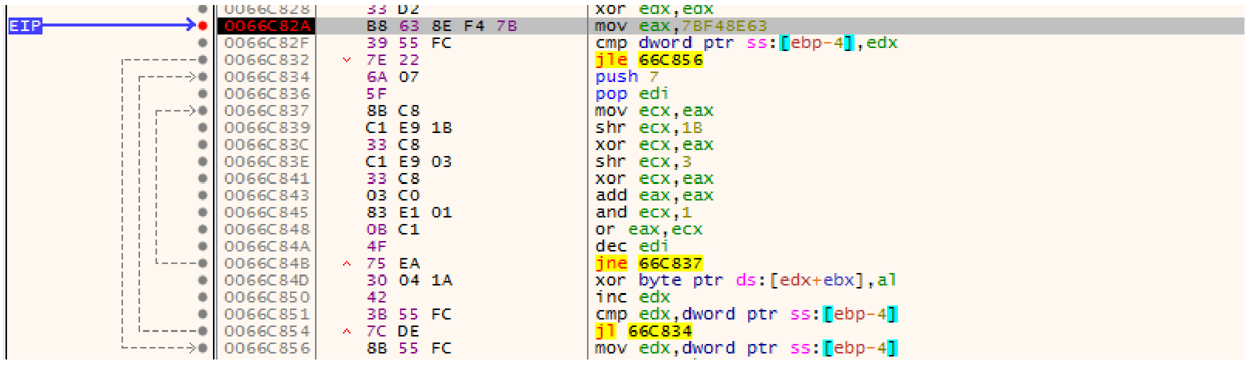
```
01FD0000 F2 A3 20 72 3B 29 95 C3 D7 ED AF C7 06 5A AA 32 ò£ r;).Àxí¯Ç.Zª2
01FD0010 F5 AB F3 D2 2D D0 28 55 B3 83 ED BE 36 00 2A 05 õ«óÒ-Ð(U³.í¾6.*.
01FD0020 8B D6 25 F5 AD 9D F2 71 97 B0 6F 9A 79 D2 17 8D .Ö%õ..òq.°o.yÒ..
01FD0030 85 DA 5A 3C 23 82 2E 61 88 59 B4 72 F1 F8 60 71 .ÚZ<#..a.Y´rñø`q
01FD0040 C6 71 EB F0 49 32 61 A3 31 A6 16 93 25 59 6A 65 ÆqëðI2a£1¦..%Yje
01FD0050 8B 67 18 4C 59 C0 B4 9C 44 80 C7 9F 66 D5 93 8E .g.LYÀ´.D.Ç.fÕ..
01FD0060 FB D0 4B 86 D4 1D DF 1C 16 39 31 BA 19 B6 D1 65 ûÐK.Ô.ß..91º.¶Ñe
01FD0070 95 7B 47 BC CF FB 53 7D E4 15 82 52 48 79 EB A0 .{G¼ÏûS}ä..RHyë
01FD0080 E6 A1 EE A1 F1 0A E5 26 FD 60 B4 BB 34 9C C2 84 æ¡î¡ñ.å&ý`´»4.Â.
01FD0090 9D DD FB 10 8B 79 25 4A E1 F6 32 F5 59 CD 31 1B .Ýû..y%Jáö2õYÍ1.
01FD00A0 16 6D AA 76 EC F3 08 8A CB 9D FE 75 13 E6 CF 61 .mªvìó..Ë.þu.æÏa
01FD00B0 15 EA 24 18 9B D1 0F 12 FF DD C5 9C 9E 02 F7 76 .ê$..Ñ..ÿÝÅ...÷v
01FD00C0 B3 38 CC 05 39 00 8F 2D 59 BB 9D A4 B7 4C A6 FC ³8Ì.9..-Y».¤·L¦ü
01FD00D0 BE 24 78 B6 BC 6D 06 FF 69 F5 93 F1 45 B7 75 61 ¾$x¶¼m.ÿiõ.ñE·ua
01FD00E0 CF BC EF 70 F7 6A 96 8D C0 49 A7 A3 80 0E 40 5E Ï¼ïp÷j..ÀI§£..@^
01FD00F0 2A 20 80 0D B3 98 EE 90 3F 2C CA F5 A5 8F 90 18 * ..³.î.?,Êõ¥...
01FD0100 24 58 20 02 F9 F8 7F B3 2A E0 F5 CC 7D 38 29 D8 $X .ùø.³*àõÌ}8)Ø
01FD0110 0A 8B A9 77 D7 EB CE 6F 52 92 81 BB C2 1D EB 8A ..©w×ëÎoR..»Â.ë.
01FD0120 48 F6 4E 7B A2 11 FC 0A 40 2E 42 65 FA 63 BD 87 HöN{¢.ü.@.Beúc½.
```

8.t in memory

And the shellcode decrypts the 8.t file in memory at 0066C82A.

The loop of decryption is a xoring with different manipulations on the decryption key.

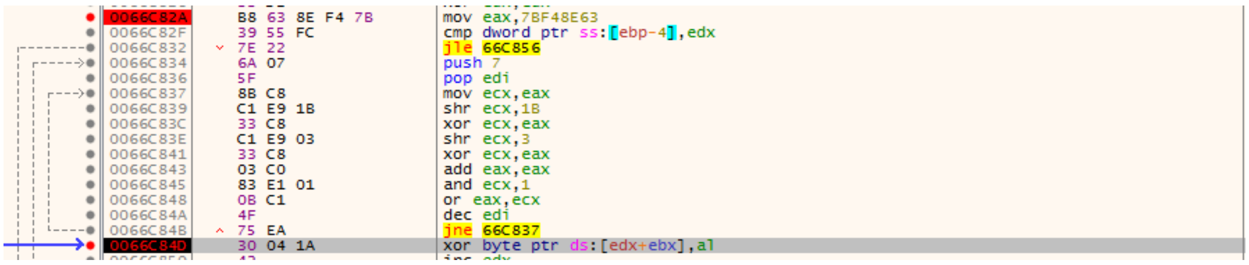
At the start of the decryption the key is set to 7BF48E63.



```
      0066C828   33 D2         xor edx,edx
EIP   0066C82A   B8 63 8E F4 7B   mov eax,7BF48E63
      0066C82F   39 55 FC      cmp dword ptr ss:[ebp-4],edx
      0066C832   7E 22         jle 66C856
      0066C834   6A 07         push 7
      0066C836   5F            pop edi
      0066C837   8B C8         mov ecx,eax
      0066C839   C1 E9 1B      shr ecx,1B
      0066C83C   33 C8         xor ecx,eax
      0066C83E   C1 E9 03      shr ecx,3
      0066C841   33 C8         xor ecx,eax
      0066C843   03 C0         add eax,eax
      0066C845   83 E1 01      and ecx,1
      0066C848   0B C1         or eax,ecx
      0066C84A   4F            dec edi
      0066C84B   75 EA         jne 66C837
      0066C84D   30 04 1A      xor byte ptr ds:[edx+ebx],al
      0066C850   42            inc edx
      0066C851   3B 55 FC      cmp edx,dword ptr ss:[ebp-4]
      0066C854   7C DE         jl 66C834
      0066C856   8B 55 FC      mov edx,dword ptr ss:[ebp-4]
```

Decryption loop

And the xor is made after key manipulation.

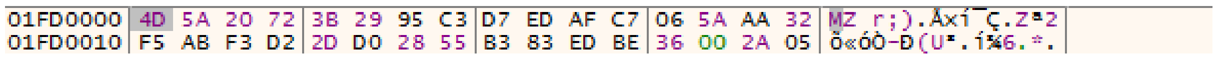


```
0066C82A   B8 63 8E F4 7B   mov eax,7BF48E63
0066C82F   39 55 FC      cmp dword ptr ss:[ebp-4],edx
0066C832   7E 22         jle 66C856
0066C834   6A 07         push 7
0066C836   5F            pop edi
0066C837   8B C8         mov ecx,eax
0066C839   C1 E9 1B      shr ecx,1B
0066C83C   33 C8         xor ecx,eax
0066C83E   C1 E9 03      shr ecx,3
0066C841   33 C8         xor ecx,eax
0066C843   03 C0         add eax,eax
0066C845   83 E1 01      and ecx,1
0066C848   0B C1         or eax,ecx
0066C84A   4F            dec edi
0066C84B   75 EA         jne 66C837
0066C84D   30 04 1A      xor byte ptr ds:[edx+ebx],al
```

Set the decryption key in EAX

If we check the destination of the result of the xoring (here edx + ebx), we find 01FD0000 where 8.t is loaded.

After two step of the loop, we can see the magic number MZ set at the begin of memory section.

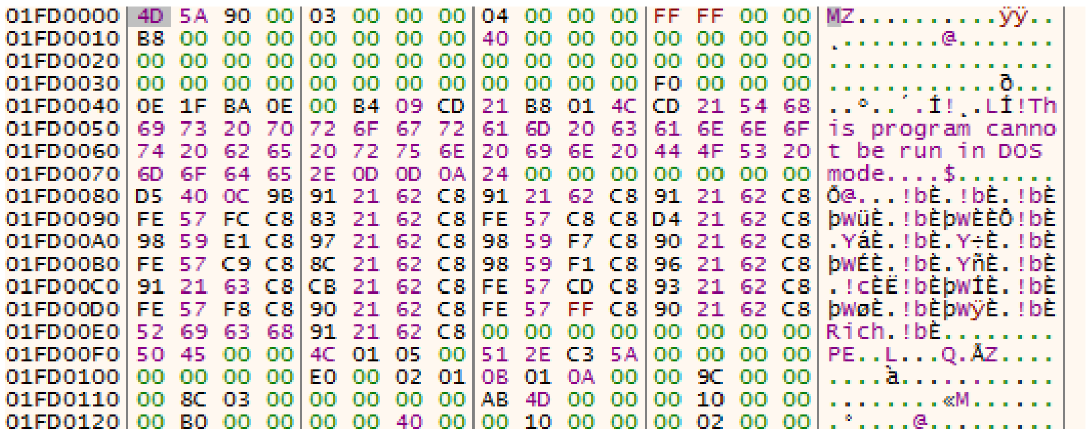


```
01FD0000 4D 5A 20 72 3B 29 95 C3 D7 ED AF C7 06 5A AA 32 MZ r;).Àxí¯Ç.Zª2
01FD0010 F5 AB F3 D2 2D D0 28 55 B3 83 ED BE 36 00 2A 05 õ«óÒ-Ð(U³.í¾6.*.
```

MZ magic number

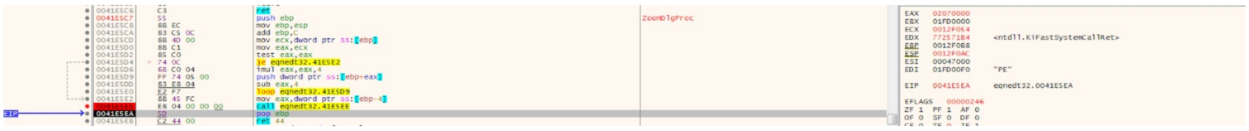
At the end of the decryption loop, we have a PE in memory at 01FD0000.

the file 8.t has been decrypted.

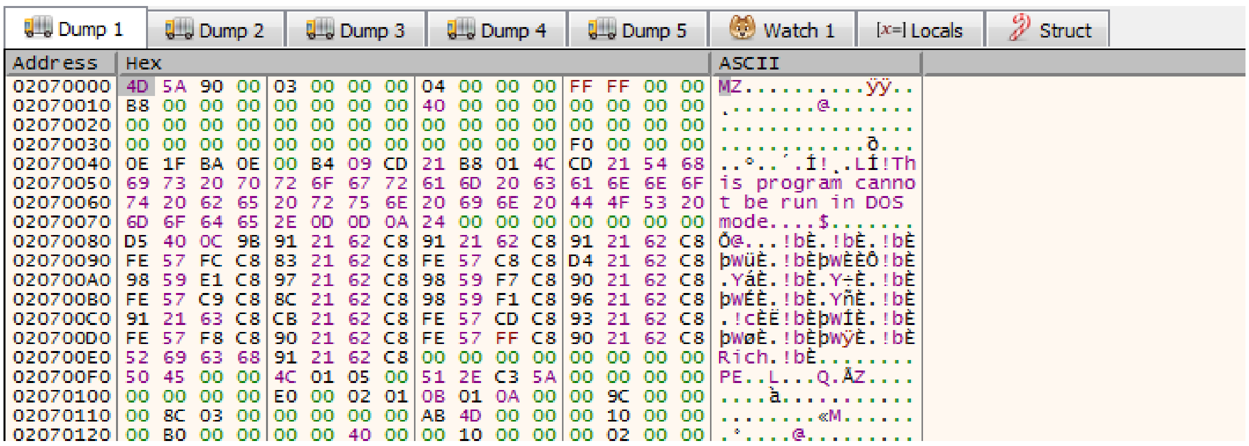

```
01FD0000  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
01FD0010  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ¸.......@.......
01FD0020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01FD0030  00 00 00 00 00 00 00 00 00 00 00 00 F0 00 00 00  ............ð...
01FD0040  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
01FD0050  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
01FD0060  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS 
01FD0070  6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
01FD0080  D5 40 0C 9B 91 21 62 C8 91 21 62 C8 91 21 62 C8  Õ@...!bÈ.!bÈ.!bÈ
01FD0090  FE 57 FC C8 83 21 62 C8 FE 57 C8 C8 D4 21 62 C8  þWüÈ.!bÈþWÈÈÔ!bÈ
01FD00A0  98 59 E1 C8 97 21 62 C8 98 59 F7 C8 90 21 62 C8  .YáÈ.!bÈ.Y÷È.!bÈ
01FD00B0  FE 57 C9 C8 8C 21 62 C8 98 59 F1 C8 96 21 62 C8  þWÉÈ.!bÈ.YñÈ.!bÈ
01FD00C0  91 21 63 C8 CB 21 62 C8 FE 57 CD C8 93 21 62 C8  .!cÈË!bÈþWÍÈ.!bÈ
01FD00D0  FE 57 F8 C8 90 21 62 C8 FE 57 FF C8 90 21 62 C8  þWøÈ.!bÈþWÿÈ.!bÈ
01FD00E0  52 69 63 68 91 21 62 C8 00 00 00 00 00 00 00 00  Rich.!bÈ........
01FD00F0  50 45 00 00 4C 01 05 00 51 2E C3 5A 00 00 00 00  PE..L...Q.ÃZ....
01FD0100  00 00 00 00 E0 00 02 01 0B 01 0A 00 00 9C 00 00  ....à...........
01FD0110  00 8C 03 00 00 00 00 00 AB 4D 00 00 00 10 00 00  ........«M......
01FD0120  00 B0 00 00 00 00 40 00 00 10 00 00 00 02 00 00  .°....@.........
```

8.t fully decrypted

Then, the shellcode uses the VirtualAlloc and create a memory page at 02070000.


```
0041E5C6  C3              ret
0041E5C7  55              push ebp                          ZoomDlgProc
0041E5C8  8B EC           mov ebp,esp
0041E5CA  83 C5 0C        add ebp,C
0041E5CD  8B 4D 00        mov ecx,dword ptr ss:[ebp]
0041E5D0  8B C1           mov eax,ecx
0041E5D2  85 C0           test eax,eax
0041E5D4  74 0C           je eqnedt32.41E5E2
0041E5D6  6B C0 04        imul eax,eax,4
0041E5D9  FF 74 05 00     push dword ptr ss:[ebp+eax]
0041E5DD  83 E8 04        sub eax,4
0041E5E0  E2 F7           loop eqnedt32.41E5D9
0041E5E2  8B 45 FC        mov eax,dword ptr ss:[ebp-4]
0041E5E5  E8 04 00 00 00  call eqnedt32.41E5EE
0041E5EA  5D              pop ebp
0041E5EB  C2 44 00        ret 44

EAX     02070000
EBX     01FD0000
ECX     0012F054
EDX     772571B4     <ntdll.KiFastSystemCallRet>
EBP     0012F0B8
ESP     0012F0AC
ESI     00047000
EDI     01FD00F0     "PE"

EIP     0041E5EA     eqnedt32.0041E5EA

EFLAGS    00000246
ZF 1  PF 1  AF 0
OF 0  SF 0  DF 0
```

And the new PE at 01FD0000 is copied at this address.

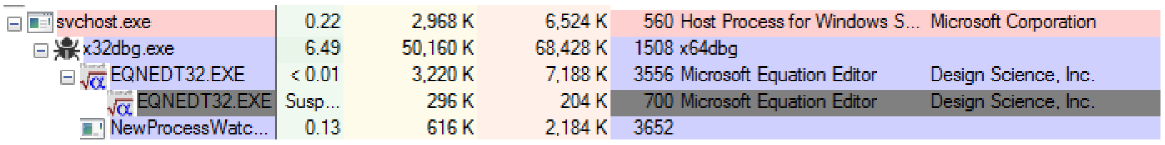


```
Address   Hex                                              ASCII
02070000  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
02070010  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ¸.......@.......
02070020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
02070030  00 00 00 00 00 00 00 00 00 00 00 00 F0 00 00 00  ............ð...
02070040  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
02070050  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
02070060  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS 
02070070  6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
02070080  D5 40 0C 9B 91 21 62 C8 91 21 62 C8 91 21 62 C8  Õ@...!bÈ.!bÈ.!bÈ
02070090  FE 57 FC C8 83 21 62 C8 FE 57 C8 C8 D4 21 62 C8  þWüÈ.!bÈþWÈÈÔ!bÈ
020700A0  98 59 E1 C8 97 21 62 C8 98 59 F7 C8 90 21 62 C8  .YáÈ.!bÈ.Y÷È.!bÈ
020700B0  FE 57 C9 C8 8C 21 62 C8 98 59 F1 C8 96 21 62 C8  þWÉÈ.!bÈ.YñÈ.!bÈ
020700C0  91 21 63 C8 CB 21 62 C8 FE 57 CD C8 93 21 62 C8  .!cÈË!bÈþWÍÈ.!bÈ
020700D0  FE 57 F8 C8 90 21 62 C8 FE 57 FF C8 90 21 62 C8  þWøÈ.!bÈþWÿÈ.!bÈ
020700E0  52 69 63 68 91 21 62 C8 00 00 00 00 00 00 00 00  Rich.!bÈ........
020700F0  50 45 00 00 4C 01 05 00 51 2E C3 5A 00 00 00 00  PE..L...Q.ÃZ....
02070100  00 00 00 00 E0 00 02 01 0B 01 0A 00 00 9C 00 00  ....à...........
02070110  00 8C 03 00 00 00 00 00 AB 4D 00 00 00 10 00 00  ........«M......
02070120  00 B0 00 00 00 00 40 00 00 10 00 00 00 02 00 00  .°....@.........
```

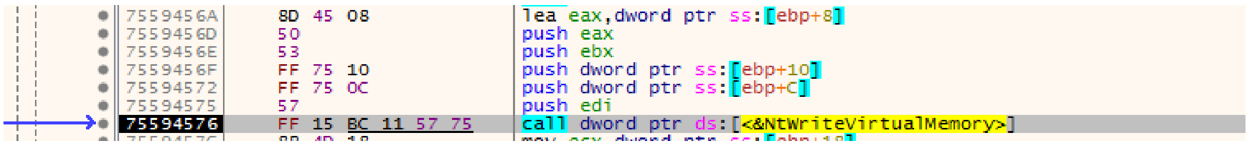
the PE decrypted is copied in the new memory page

After GetModuleFileNameA is called to have the path of EQNEDT32.exe

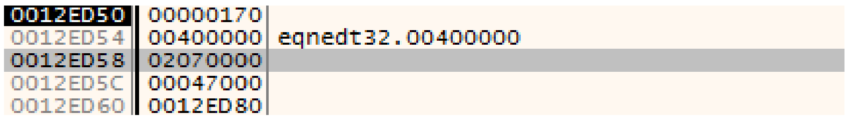
And EQNEDT32.exe is forked in suspend status by a CreateProcess and the shellcode overwrite it by the PE at the address 02070000


```
svchost.exe             0.22    2,968 K    6,524 K   560 Host Process for Windows S...  Microsoft Corporation
  x32dbg.exe            6.49   50,160 K   68,428 K  1508 x64dbg
    EQNEDT32.EXE      < 0.01    3,220 K    7,188 K  3556 Microsoft Equation Editor    Design Science, Inc.
      EQNEDT32.EXE      Susp...   296 K      204 K   700 Microsoft Equation Editor    Design Science, Inc.
    NewProcessWatc...   0.13      616 K    2,184 K  3652
```

Fork of EQNEDT32.exe


```
7559456A  8D 45 08           lea eax,dword ptr ss:[ebp+8]
7559456D  50                 push eax
7559456E  53                 push ebx
7559456F  FF 75 10           push dword ptr ss:[ebp+10]
75594572  FF 75 0C           push dword ptr ss:[ebp+C]
75594575  57                 push edi
75594576  FF 15 BC 11 57 75  call dword ptr ds:[<&NtWriteVirtualMemory>]
```

Overwritting of EQNEDT32.exe



```
0012ED50  00000170
0012ED54  00400000  eqnedt32.00400000
0012ED58  02070000
0012ED5C  00047000
0012ED60  0012ED80
```

Stack used by NTWriteVirtualMemory

And the shellcode does a ResumeThread to launch the new PE.

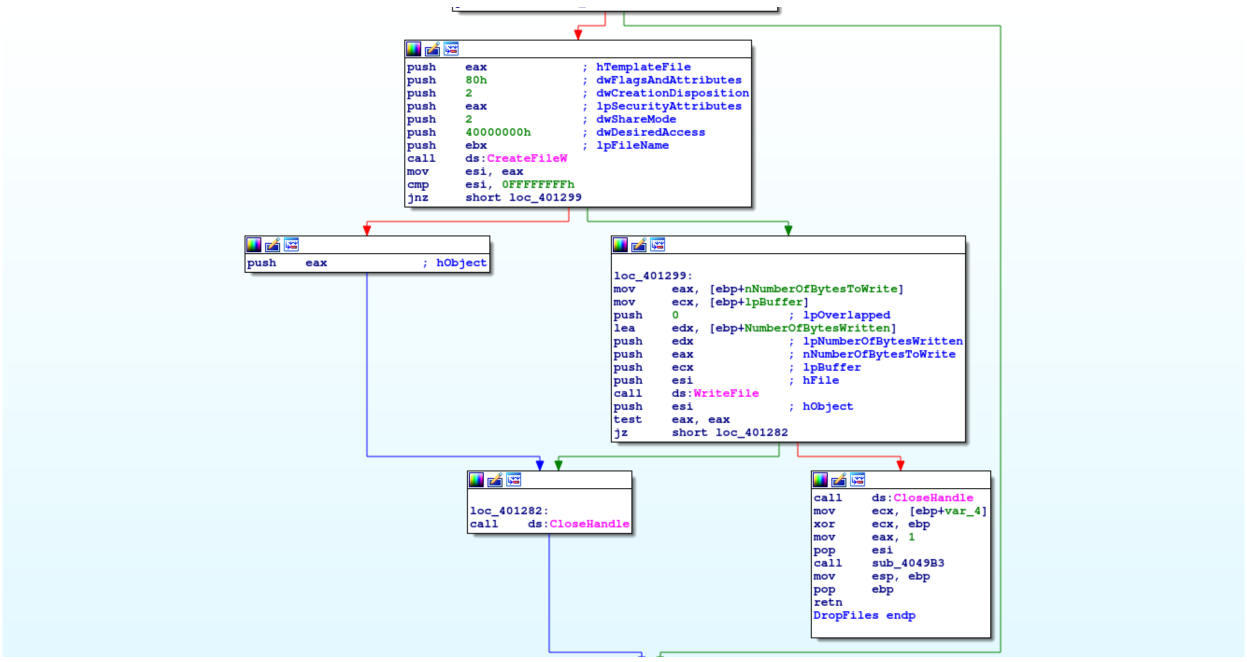
So, We've found all API Calls in the object ole at the beginning and we have a runPE to launch the new EQNEDT32.exe overwritten.

Analysing the fork of EQNEDT32.exe

We know that this process has to create on disk two files following the Joe SandBox Analysis:

- A dll named RasTls.dll
- A executable file named dascgosrky.exe

If we dump EQNEDT32.exe and we put in IDA, we found quickly the function that drops the files on disk (sub_00401150) renamed dropFiles.



```
push    eax             ; hTemplateFile
push    80h             ; dwFlagsAndAttributes
push    2               ; dwCreationDisposition
push    eax             ; lpSecurityAttributes
push    2               ; dwShareMode
push    40000000h       ; dwDesiredAccess
push    ebx             ; lpFileName
call    ds:CreateFileW
mov     esi, eax
cmp     esi, 0FFFFFFFFh
jnz     short loc_401299
```

```
push    eax                     ; hObject
```

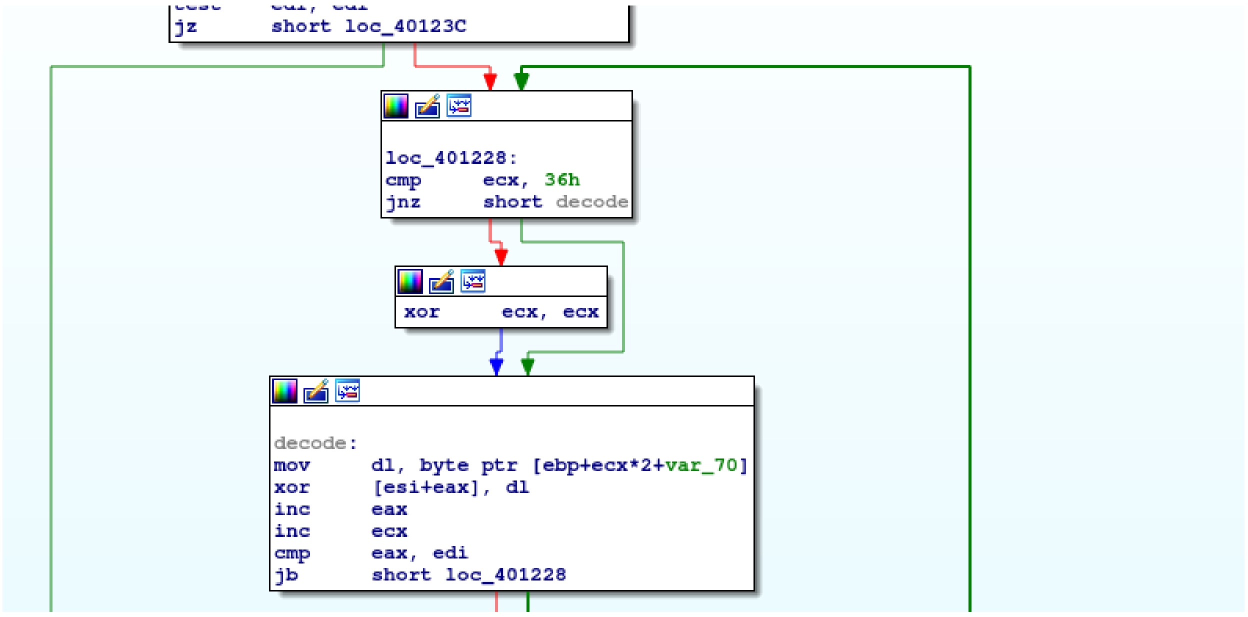
```
loc_401299:
mov     eax, [ebp+nNumberOfBytesToWrite]
mov     ecx, [ebp+lpBuffer]
push    0                       ; lpOverlapped
lea     edx, [ebp+NumberOfBytesWritten]
push    edx                     ; lpNumberOfBytesWritten
push    eax                     ; nNumberOfBytesToWrite
push    ecx                     ; lpBuffer
push    esi                     ; hFile
call    ds:WriteFile
push    esi                     ; hObject
test    eax, eax
jz      short loc_401282
```

```
loc_401282:
call    ds:CloseHandle
```

```
call    ds:CloseHandle
mov     ecx, [ebp+var_4]
xor     ecx, ebp
mov     eax, 1
pop     esi
call    sub_4049B3
mov     esp, ebp
pop     ebp
retn
DropFiles endp
```

DropFiles Fucntion

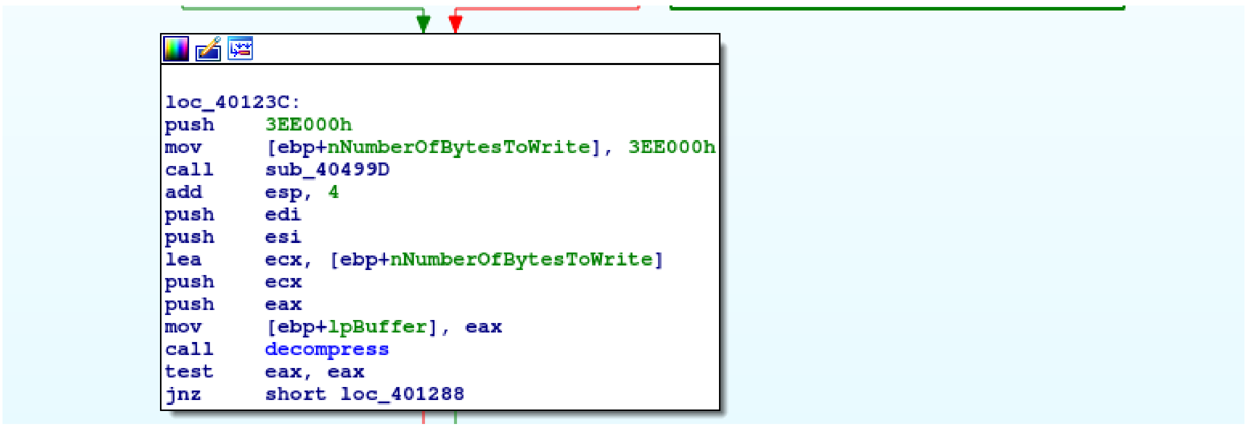
And at the start of this functions, we have a loop with a xor.



```
jz      short loc_40123C

loc_401228:
cmp     ecx, 36h
jnz     short decode

xor     ecx, ecx

decode:
mov     dl, byte ptr [ebp+ecx*2+var_70]
xor     [esi+eax], dl
inc     eax
inc     ecx
cmp     eax, edi
jb      short loc_401228
```

Second loop of decryption

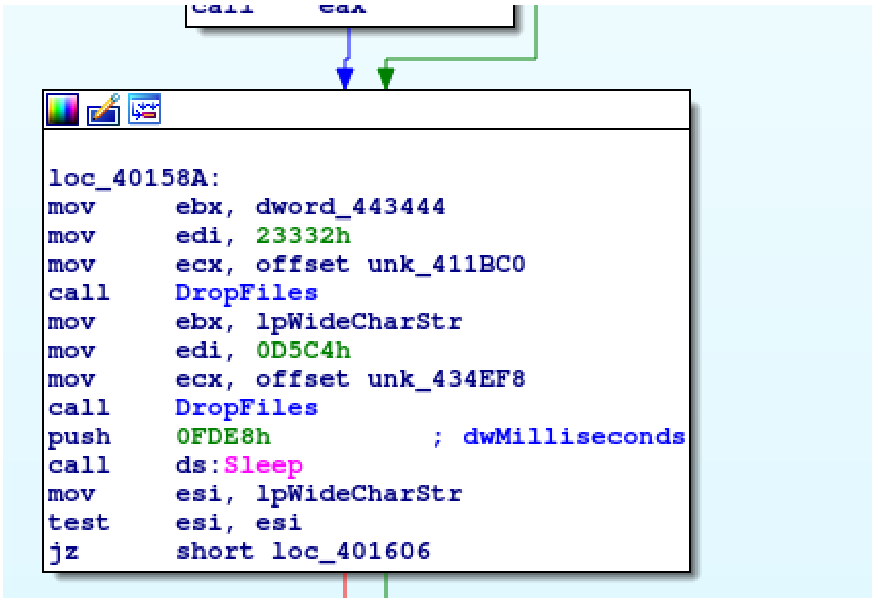
And just after we have a call of the decompression function.



```
loc_40123C:
push    3EE000h
mov     [ebp+nNumberOfBytesToWrite], 3EE000h
call    sub_40499D
add     esp, 4
push    edi
push    esi
lea     ecx, [ebp+nNumberOfBytesToWrite]
push    ecx
push    eax
mov     [ebp+lpBuffer], eax
call    decompress
test    eax, eax
jnz     short loc_401288
```

Decompression function used zlib

The function dropFiles is called twice by the sub_4012D0.

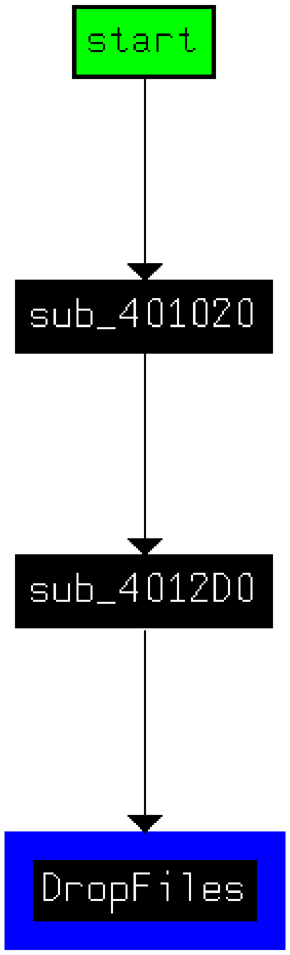


```
loc_40158A:
mov     ebx, dword_443444
mov     edi, 23332h
mov     ecx, offset unk_411BC0
call    DropFiles
mov     ebx, lpWideCharStr
mov     edi, 0D5C4h
mov     ecx, offset unk_434EF8
call    DropFiles
push    0FDE8h          ; dwMilliseconds
call    ds:Sleep
mov     esi, lpWideCharStr
test    esi, esi
jz      short loc_401606
```

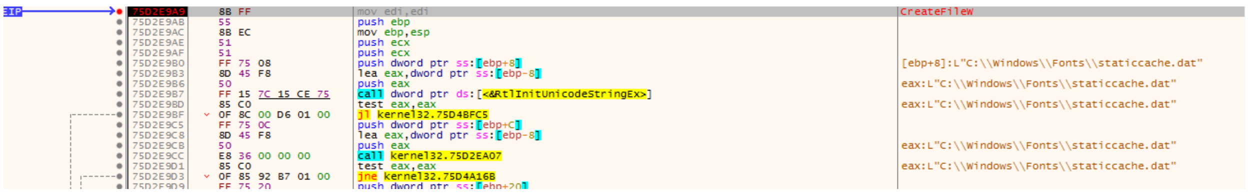
Drop the dll and the executable

If we check the call graph, DropFiles is called only by the function sub_4012D0.

Functions using DropFiles function



So we set a breakpoint on CreateFile because at each execution, EQNEDT32.exe starts by CreateFile onstaticcache.dat.

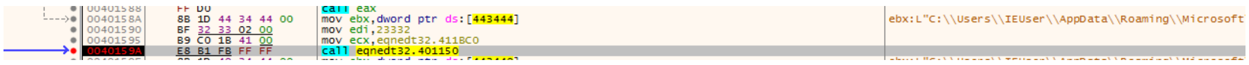


```
EIP ──→ 75D2E9A9   8B FF              mov edi,edi                                   CreateFileW
        75D2E9AB   55                 push ebp
        75D2E9AC   8B EC              mov ebp,esp
        75D2E9AE   51                 push ecx
        75D2E9AF   51                 push ecx
        75D2E9B0   FF 75 08           push dword ptr ss:[ebp+8]                     [ebp+8]:L"C:\\Windows\\Fonts\\staticcache.dat"
        75D2E9B3   8D 45 F8           lea eax,dword ptr ss:[ebp-8]
        75D2E9B6   50                 push eax                                      eax:L"C:\\Windows\\Fonts\\staticcache.dat"
        75D2E9B7   FF 15 7C 15 CE 75  call dword ptr ds:[<&RtlInitUnicodeStringEx>]
        75D2E9BD   85 C0              test eax,eax                                  eax:L"C:\\Windows\\Fonts\\staticcache.dat"
        75D2E9BF   0F 8C 00 D6 01 00  jl kernel32.75D4BFC5
        75D2E9C5   FF 75 0C           push dword ptr ss:[ebp+C]
        75D2E9C8   8D 45 F8           lea eax,dword ptr ss:[ebp-8]
        75D2E9CB   50                 push eax                                      eax:L"C:\\Windows\\Fonts\\staticcache.dat"
        75D2E9CC   E8 36 00 00 00     call kernel32.75D2EA07
        75D2E9D1   85 C0              test eax,eax                                  eax:L"C:\\Windows\\Fonts\\staticcache.dat"
        75D2E9D3   0F 85 92 B7 01 00  jne kernel32.75D4A16B
        75D2E9D9   FF 75 20           push dword ptr ss:[ebp+20]
```

Breakpoint to createfile

And we return at the user code to set a new breakpoint to check the static analysis.

So we set a breakpoint at 0040159A when DropFiles is called.

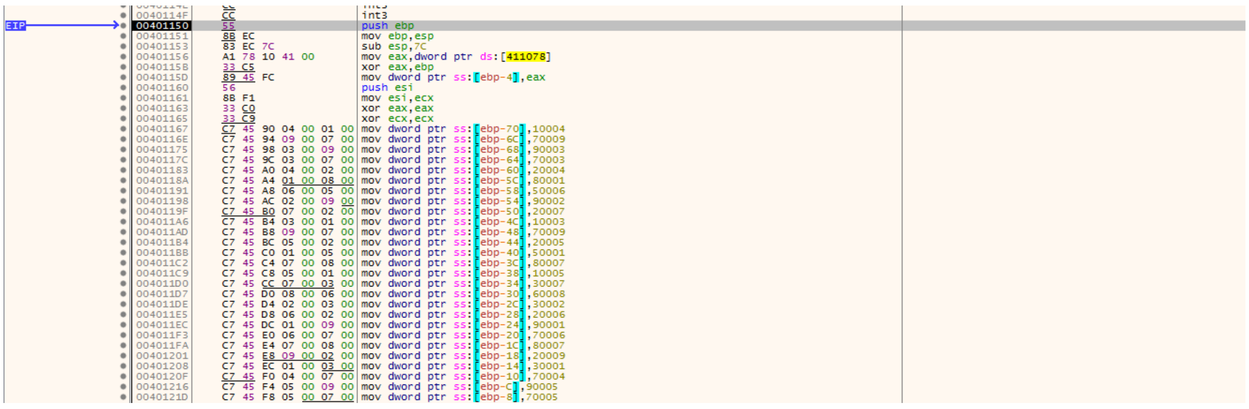


```
        00401588   FF D0              call eax
    --→ 0040158A   8B 1D 44 34 44 00  mov ebx,dword ptr ds:[443444]                 ebx:L"C:\\Users\\IEUser\\AppData\\Roaming\\Microsoft
        00401590   BF 32 33 02 00     mov edi,23332
        00401595   B9 C0 1B 41 00     mov ecx,eqnedt32.411BC0
──→     0040159A   E8 B1 FB FF FF     call eqnedt32.401150
```

Breakpoint to the first call of DropFiles

And now we can analyse the second loop of decryption.

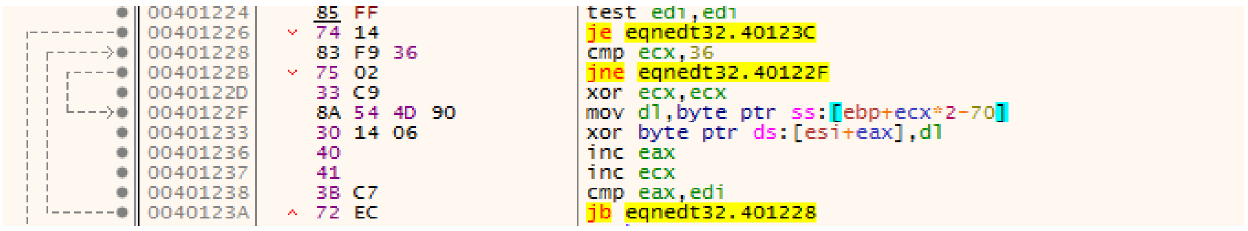
The first step is the initialization of the decryption function.



```
0040114E  CC                    int3
0040114F  CC                    int3
00401150  55                    push ebp
00401151  8B EC                 mov ebp,esp
00401153  83 EC 7C              sub esp,7C
00401156  A1 78 10 41 00        mov eax,dword ptr ds:[411078]
0040115B  33 C5                 xor eax,ebp
0040115D  89 45 FC              mov dword ptr ss:[ebp-4],eax
00401160  56                    push esi
00401161  8B F1                 mov esi,ecx
00401163  33 C0                 xor eax,eax
00401165  33 C9                 xor ecx,ecx
00401167  C7 45 90 04 00 01 00  mov dword ptr ss:[ebp-70],10004
0040116E  C7 45 94 09 00 07 00  mov dword ptr ss:[ebp-6C],70009
00401175  C7 45 98 03 00 09 00  mov dword ptr ss:[ebp-68],90003
0040117C  C7 45 9C 03 00 07 00  mov dword ptr ss:[ebp-64],70003
00401183  C7 45 A0 04 00 02 00  mov dword ptr ss:[ebp-60],20004
0040118A  C7 45 A4 01 00 08 00  mov dword ptr ss:[ebp-5C],80001
00401191  C7 45 A8 06 00 05 00  mov dword ptr ss:[ebp-58],50006
00401198  C7 45 AC 02 00 09 00  mov dword ptr ss:[ebp-54],90002
0040119F  C7 45 B0 07 00 02 00  mov dword ptr ss:[ebp-50],20007
004011A6  C7 45 B4 03 00 01 00  mov dword ptr ss:[ebp-4C],10003
004011AD  C7 45 B8 09 00 07 00  mov dword ptr ss:[ebp-48],70009
004011B4  C7 45 BC 05 00 02 00  mov dword ptr ss:[ebp-44],20005
004011BB  C7 45 C0 01 00 05 00  mov dword ptr ss:[ebp-40],50001
004011C2  C7 45 C4 07 00 08 00  mov dword ptr ss:[ebp-3C],80007
004011C9  C7 45 C8 05 00 01 00  mov dword ptr ss:[ebp-38],10005
004011D0  C7 45 CC 07 00 03 00  mov dword ptr ss:[ebp-34],30007
004011D7  C7 45 D0 08 00 06 00  mov dword ptr ss:[ebp-30],60008
004011DE  C7 45 D4 02 00 03 00  mov dword ptr ss:[ebp-2C],30002
004011E5  C7 45 D8 06 00 02 00  mov dword ptr ss:[ebp-28],20006
004011EC  C7 45 DC 01 00 09 00  mov dword ptr ss:[ebp-24],90001
004011F3  C7 45 E0 06 00 07 00  mov dword ptr ss:[ebp-20],70006
004011FA  C7 45 E4 07 00 08 00  mov dword ptr ss:[ebp-1C],80007
00401201  C7 45 E8 09 00 02 00  mov dword ptr ss:[ebp-18],20009
00401208  C7 45 EC 01 00 03 00  mov dword ptr ss:[ebp-14],30001
0040120F  C7 45 F0 04 00 07 00  mov dword ptr ss:[ebp-10],70004
00401216  C7 45 F4 05 00 09 00  mov dword ptr ss:[ebp-C],90005
0040121D  C7 45 F8 05 00 07 00  mov dword ptr ss:[ebp-8],70005
```

Set for the second loop encryption

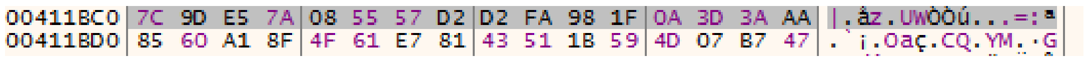
And after we find the xor and store the result in esi+eax.



```
00401224   85 FF         test edi,edi
00401226   74 14         je eqnedt32.40123C
00401228   83 F9 36      cmp ecx,36
0040122B   75 02         jne eqnedt32.40122F
0040122D   33 C9         xor ecx,ecx
0040122F   8A 54 4D 90   mov dl,byte ptr ss:[ebp+ecx*2-70]
00401233   30 14 06      xor byte ptr ds:[esi+eax],dl
00401236   40            inc eax
00401237   41            inc ecx
00401238   3B C7         cmp eax,edi
0040123A   72 EC         jb eqnedt32.401228
```

Decryption loop

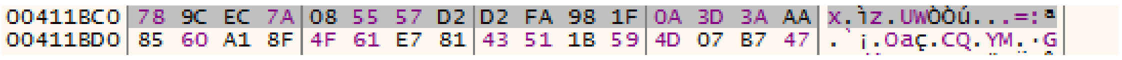
In the first step of the decryption loop, the result is written to 411BC0 in the address space of EQNEDT32.exe.



```
00411BC0  7C 9D E5 7A 08 55 57 D2 D2 FA 98 1F 0A 3D 3A AA  |.åz.UWÒÒú...=:ª
00411BD0  85 60 A1 8F 4F 61 E7 81 43 51 1B 59 4D 07 B7 47  .`¡.Oaç.CQ.YM.·G
```

Before the decryption

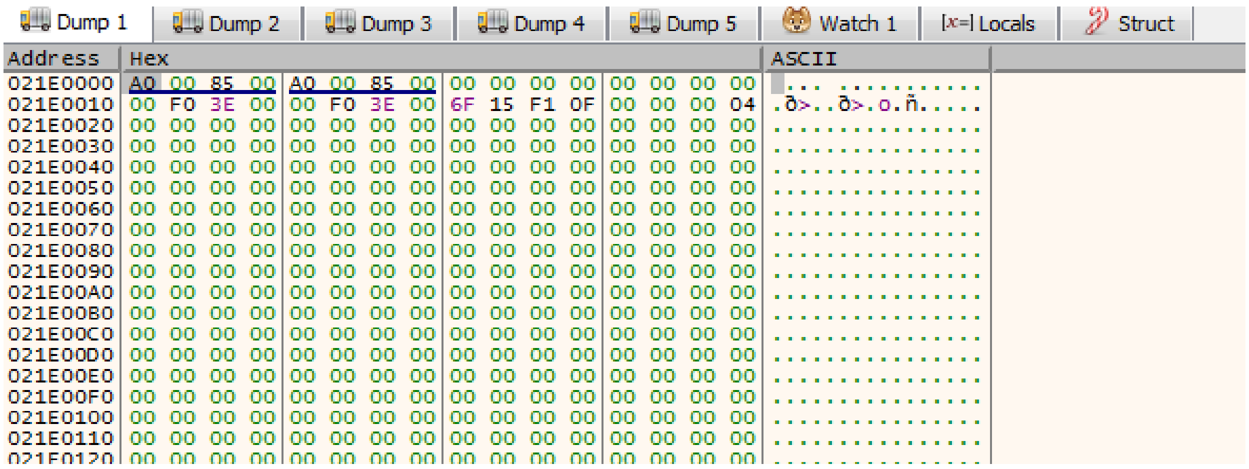
After tree loops, we obtains the header of zlib compressed object.



```
00411BC0  78 9C EC 7A 08 55 57 D2 D2 FA 98 1F 0A 3D 3A AA  x.ìz.UWÒÒú...=:ª
00411BD0  85 60 A1 8F 4F 61 E7 81 43 51 1B 59 4D 07 B7 47  .`¡.Oaç.CQ.YM.·G
```

After the decryption

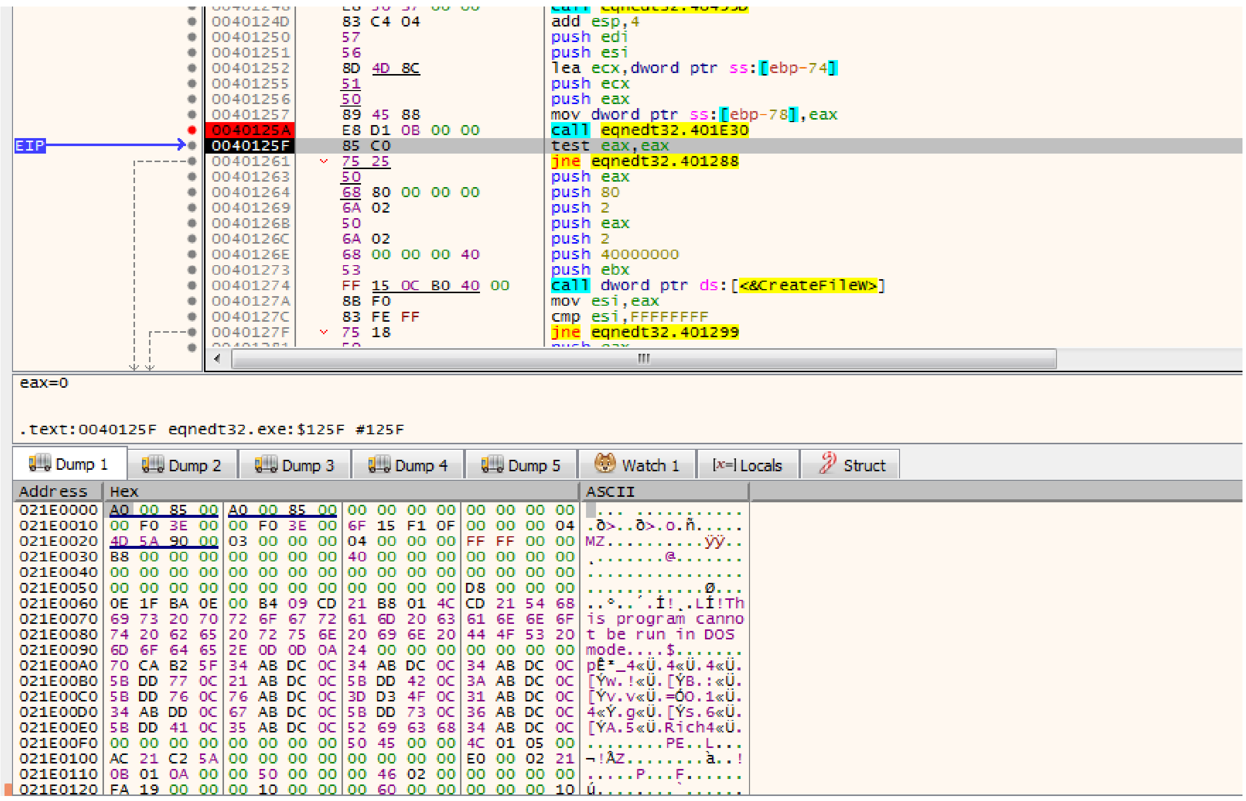
And at the memory page 021E0000, a PE is decompressed.



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

```
Address   Hex                                              ASCII
021E0000  A0 00 85 00 A0 00 85 00 00 00 00 00 00 00 00 00  ... ............
021E0010  00 F0 3E 00 00 F0 3E 00 6F 15 F1 0F 00 00 00 04  .ð>..ð>.o.ñ.....
021E0020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
021E0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
021E0040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
021E0050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
021E0060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
021E0070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
021E0080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
021E0090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
021E00A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
021E00B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
021E00C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
021E00D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
021E00E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
021E00F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
021E0100  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
021E0110  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
021E0120  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Page memory allocated to store the dll



```
0040124D   83 C4 04          add esp,4
00401250   57                push edi
00401251   56                push esi
00401252   8D 4D 8C          lea ecx,dword ptr ss:[ebp-74]
00401255   51                push ecx
00401256   50                push eax
00401257   89 45 88          mov dword ptr ss:[ebp-78],eax
0040125A   E8 D1 0B 00 00    call eqnedt32.401E30
0040125F   85 C0             test eax,eax
00401261   75 25             jne eqnedt32.401288
00401263   50                push eax
00401264   68 80 00 00 00    push 80
00401269   6A 02             push 2
0040126B   50                push eax
0040126C   6A 02             push 2
0040126E   68 00 00 00 40    push 40000000
00401273   53                push ebx
00401274   FF 15 0C B0 40 00 call dword ptr ds:[<&CreateFileW>]
0040127A   8B F0             mov esi,eax
0040127C   83 FE FF          cmp esi,FFFFFFFF
0040127F   75 18             jne eqnedt32.401299

eax=0

.text:0040125F eqnedt32.exe:$125F #125F
```

```
Address   Hex                                              ASCII
021E0000  A0 00 85 00 A0 00 85 00 00 00 00 00 00 00 00 00  ... ............
021E0010  00 F0 3E 00 00 F0 3E 00 6F 15 F1 0F 00 00 00 04  .ð>..ð>.o.ñ.....
021E0020  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
021E0030  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ........@.......
021E0040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
021E0050  00 00 00 00 00 00 00 00 00 00 00 00 D8 00 00 00  ............Ø...
021E0060  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
021E0070  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
021E0080  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS 
021E0090  6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
021E00A0  70 CA B2 5F 34 AB DC 0C 34 AB DC 0C 34 AB DC 0C  pÊ²_4«Ü.4«Ü.4«Ü.
021E00B0  5B DD 77 0C 21 AB DC 0C 5B DD 42 0C 3A AB DC 0C  [Ýw.!«Ü.[ÝB.:«Ü.
021E00C0  5B DD 76 0C 76 AB DC 0C 3D D3 4F 0C 31 AB DC 0C  [Ýv.v«Ü.=ÓO.1«Ü.
021E00D0  34 AB DD 0C 67 AB DC 0C 5B DD 73 0C 36 AB DC 0C  4«Ý.g«Ü.[Ýs.6«Ü.
021E00E0  5B DD 41 0C 35 AB DC 0C 52 69 63 68 34 AB DC 0C  [ÝA.5«Ü.Rich4«Ü.
021E00F0  00 00 00 00 00 00 00 00 50 45 00 00 4C 01 05 00  ........PE..L...
021E0100  AC 21 C2 5A 00 00 00 00 00 00 00 00 E0 00 02 21  ¬!ÂZ........à..!
021E0110  0B 01 0A 00 00 50 00 00 00 46 02 00 00 00 00 00  .....P...F......
021E0120  FA 19 00 00 00 10 00 00 00 60 00 00 00 00 00 10  ú........`......
```

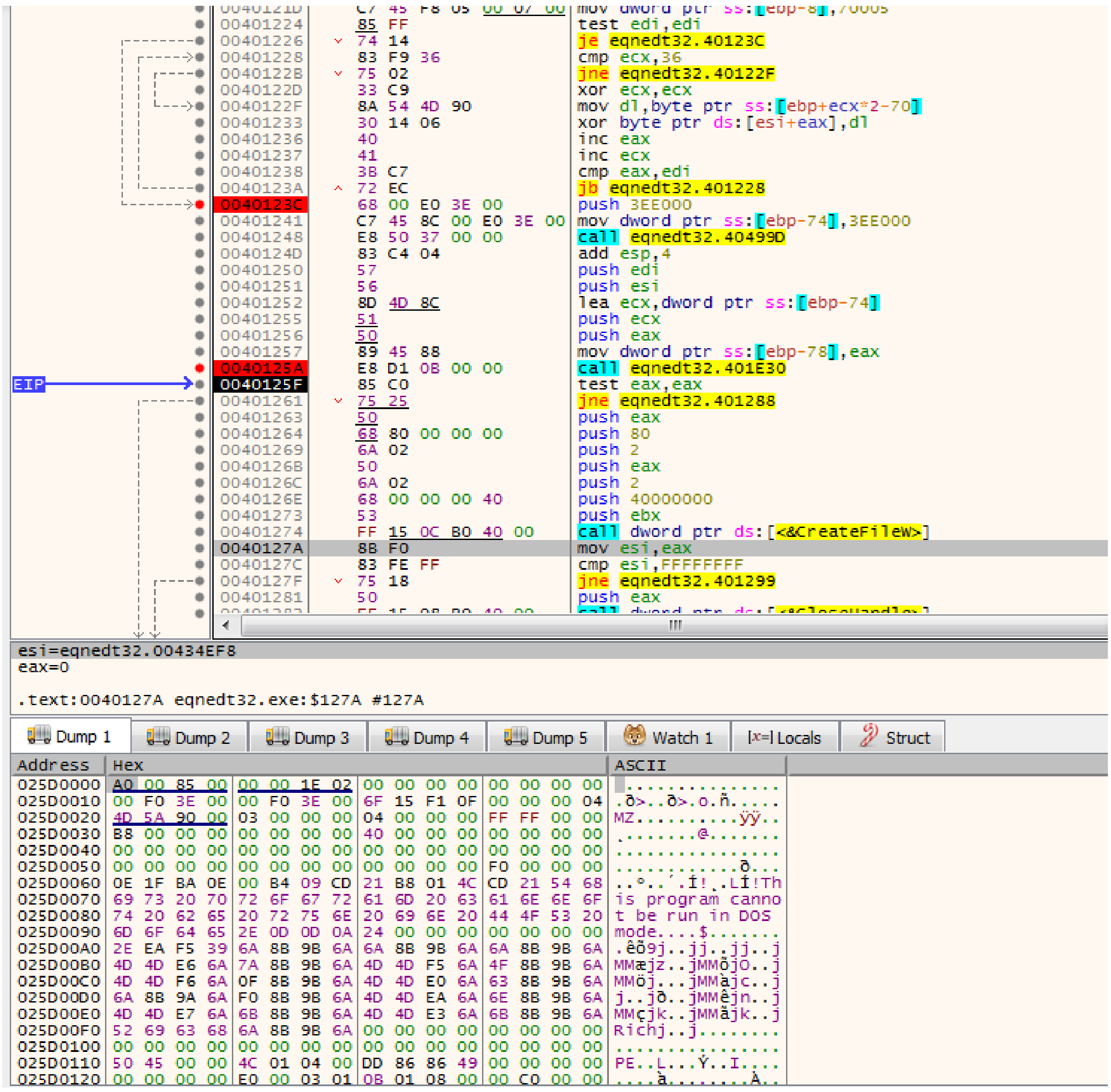
After decompression

And after the file is created with the following path: L”C:\\Users\\IEUser\\AppData\\Roaming\\Microsoft\\Windows\\Network Shortcuts\\RasTls.dll”

Stored by ebx.

DropFiles is called a twice to decrypt and decompress the executable file.

The offset where store the file is 00434EF8 and the pe decompressed is stored at 025D0020



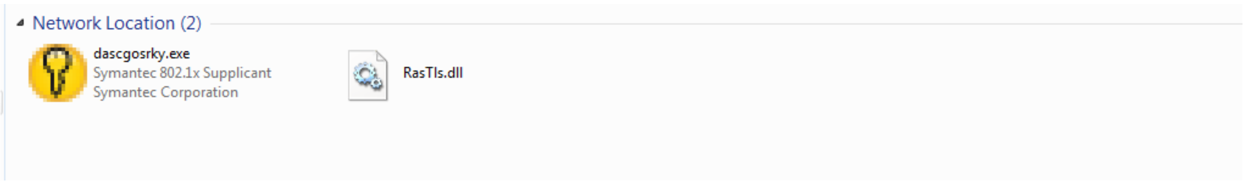
```
0040121D   C7 45 F8 05 00 07 00   mov dword ptr ss:[ebp-8],70005
00401224   85 FF                  test edi,edi
00401226   74 14                  je eqnedt32.40123C
00401228   83 F9 36               cmp ecx,36
0040122B   75 02                  jne eqnedt32.40122F
0040122D   33 C9                  xor ecx,ecx
0040122F   8A 54 4D 90            mov dl,byte ptr ss:[ebp+ecx*2-70]
00401233   30 14 06               xor byte ptr ds:[esi+eax],dl
00401236   40                     inc eax
00401237   41                     inc ecx
00401238   3B C7                  cmp eax,edi
0040123A   72 EC                  jb eqnedt32.401228
0040123C   68 00 E0 3E 00         push 3EE000
00401241   C7 45 8C 00 E0 3E 00   mov dword ptr ss:[ebp-74],3EE000
00401248   E8 50 37 00 00         call eqnedt32.40499D
0040124D   83 C4 04               add esp,4
00401250   57                     push edi
00401251   56                     push esi
00401252   8D 4D 8C               lea ecx,dword ptr ss:[ebp-74]
00401255   51                     push ecx
00401256   50                     push eax
00401257   89 45 88               mov dword ptr ss:[ebp-78],eax
0040125A   E8 D1 0B 00 00         call eqnedt32.401E30
EIP  0040125F   85 C0             test eax,eax
00401261   75 25                  jne eqnedt32.401288
00401263   50                     push eax
00401264   68 80 00 00 00         push 80
00401269   6A 02                  push 2
0040126B   50                     push eax
0040126C   6A 02                  push 2
0040126E   68 00 00 00 40         push 40000000
00401273   53                     push ebx
00401274   FF 15 0C B0 40 00      call dword ptr ds:[<&CreateFileW>]
0040127A   8B F0                  mov esi,eax
0040127C   83 FE FF               cmp esi,FFFFFFFF
0040127F   75 18                  jne eqnedt32.401299
00401281   50                     push eax

esi=eqnedt32.00434EF8
eax=0

.text:0040127A eqnedt32.exe:$127A #127A
```

```
Address   Hex                                               ASCII
025D0000  A0 00 85 00 00 00 1E 02 00 00 00 00 00 00 00 00  ................
025D0010  00 F0 3E 00 00 F0 3E 00 6F 15 F1 0F 00 00 00 04  .ð>..ð>.o.ñ.....
025D0020  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
025D0030  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ,.......@.......
025D0040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
025D0050  00 00 00 00 00 00 00 00 00 00 00 00 F0 00 00 00  ............ð...
025D0060  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
025D0070  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
025D0080  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS 
025D0090  6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
025D00A0  2E EA F5 39 6A 8B 9B 6A 6A 8B 9B 6A 6A 8B 9B 6A  .êõ9j..jj..jj..j
025D00B0  4D 4D E6 6A 7A 8B 9B 6A 4D 4D F5 6A 4F 8B 9B 6A  MMæjz..jMMõjO..j
025D00C0  4D 4D F6 6A 0F 8B 9B 6A 4D 4D E0 6A 63 8B 9B 6A  MMöj...jMMàjc..j
025D00D0  6A 8B 9A 6A F0 8B 9B 6A 4D 4D EA 6A 6E 8B 9B 6A  j..jð..jMMêjn..j
025D00E0  4D 4D E7 6A 6B 8B 9B 6A 4D 4D E3 6A 6B 8B 9B 6A  MMçjk..jMMãjk..j
025D00F0  52 69 63 68 6A 8B 9B 6A 00 00 00 00 00 00 00 00  Richj..j........
025D0100  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
025D0110  50 45 00 00 4C 01 04 00 DD 86 86 49 00 00 00 00  PE..L...Ý..I....
025D0120  00 00 00 00 E0 00 03 01 0B 01 08 00 00 C0 00 00  ....à.......À..
```

Decryption of the executable dascgosrky.exe

And the path of the new file is : ebx=005DA228 L"C:\\Users\\IEUser\\AppData\\Roaming\\Microsoft\\Windows\\Network Shortcuts\\dascgosrky.exe"

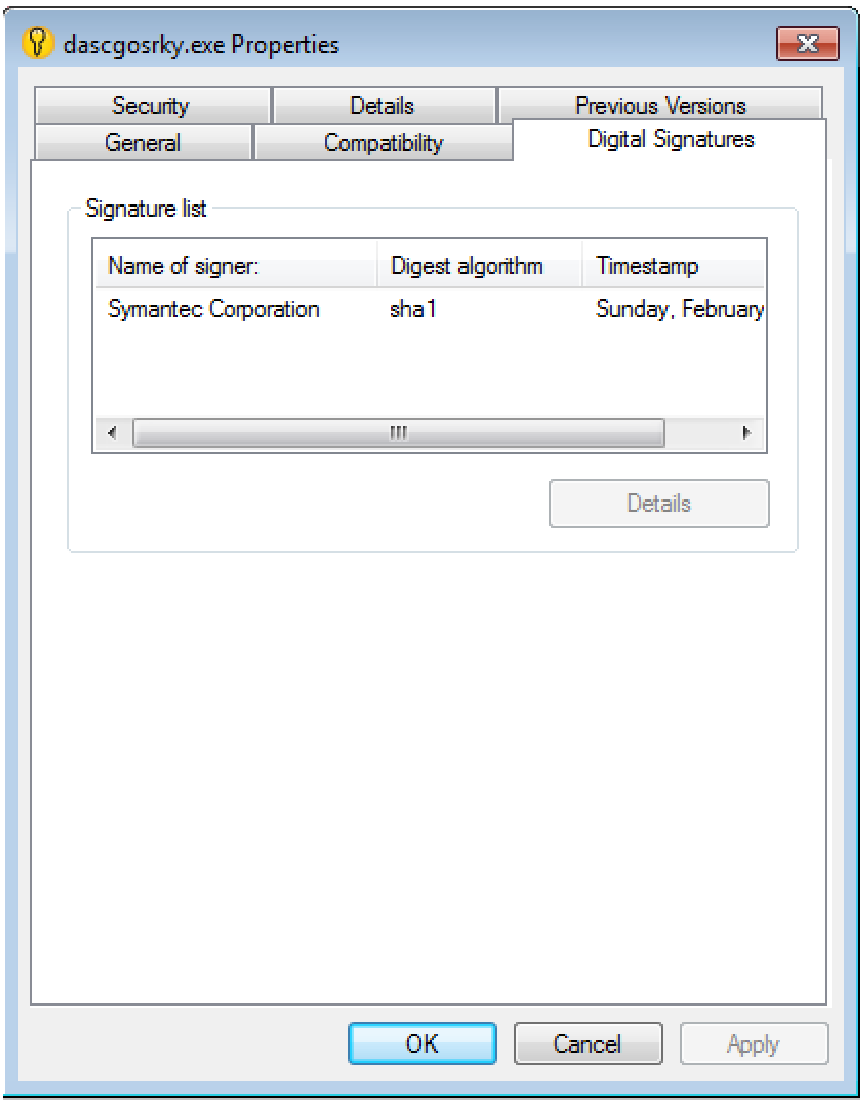
So we have two files in networks shortcuts of Windows.



Files drops on disk

dll hijacking

Dascgosrky.exe is a legit and trusted software develop by Symantec.

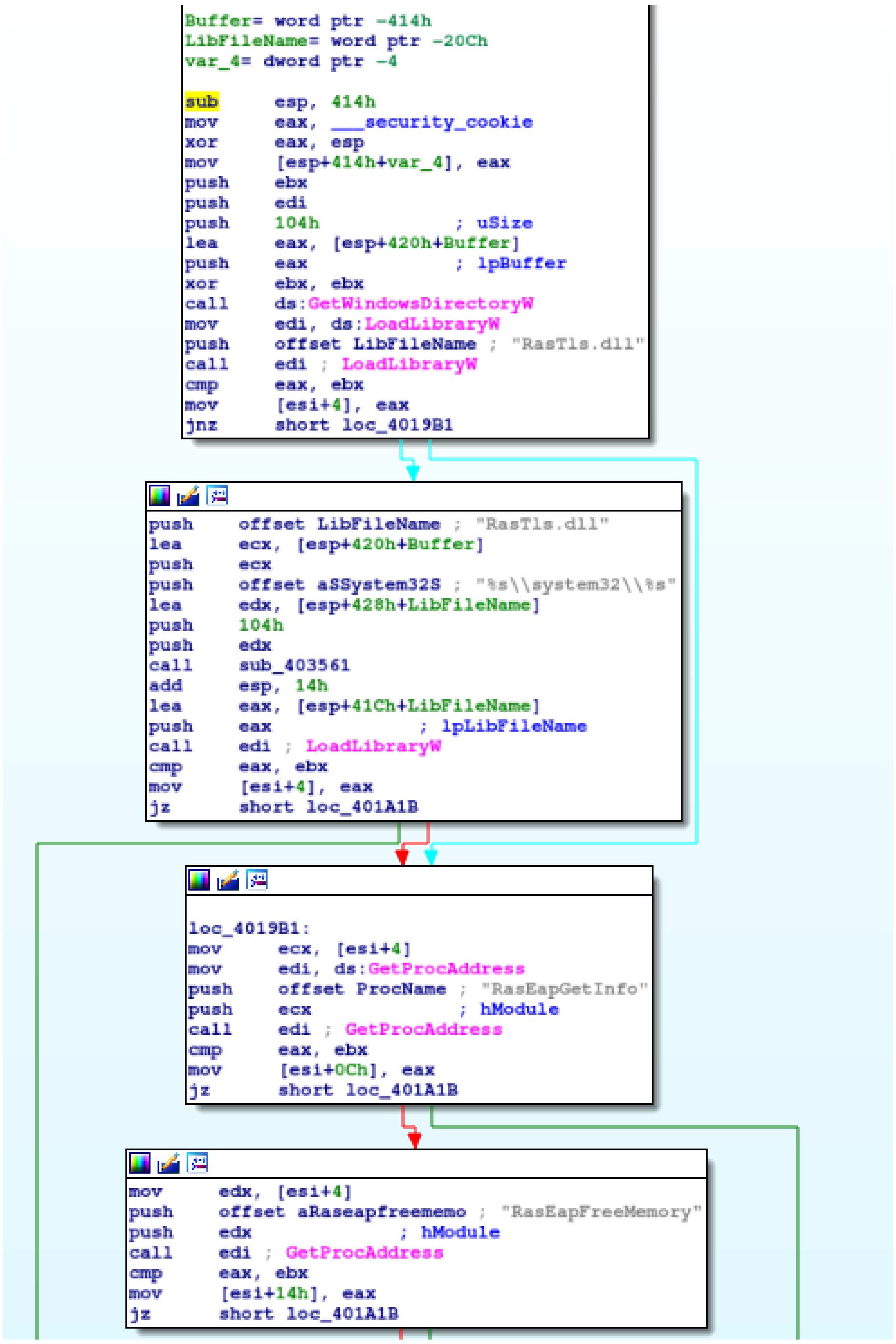

dascgosrky.exe Properties

Security | Details | Previous Versions | General | Compatibility | Digital Signatures

Signature list

Name of signer:	Digest algorithm	Timestamp
Symantec Corporation	sha1	Sunday, February

To load the library RasTls.dll, the executable calls LoadLibrary and GetProcaddress in sub_401940 to execute the malicious functions



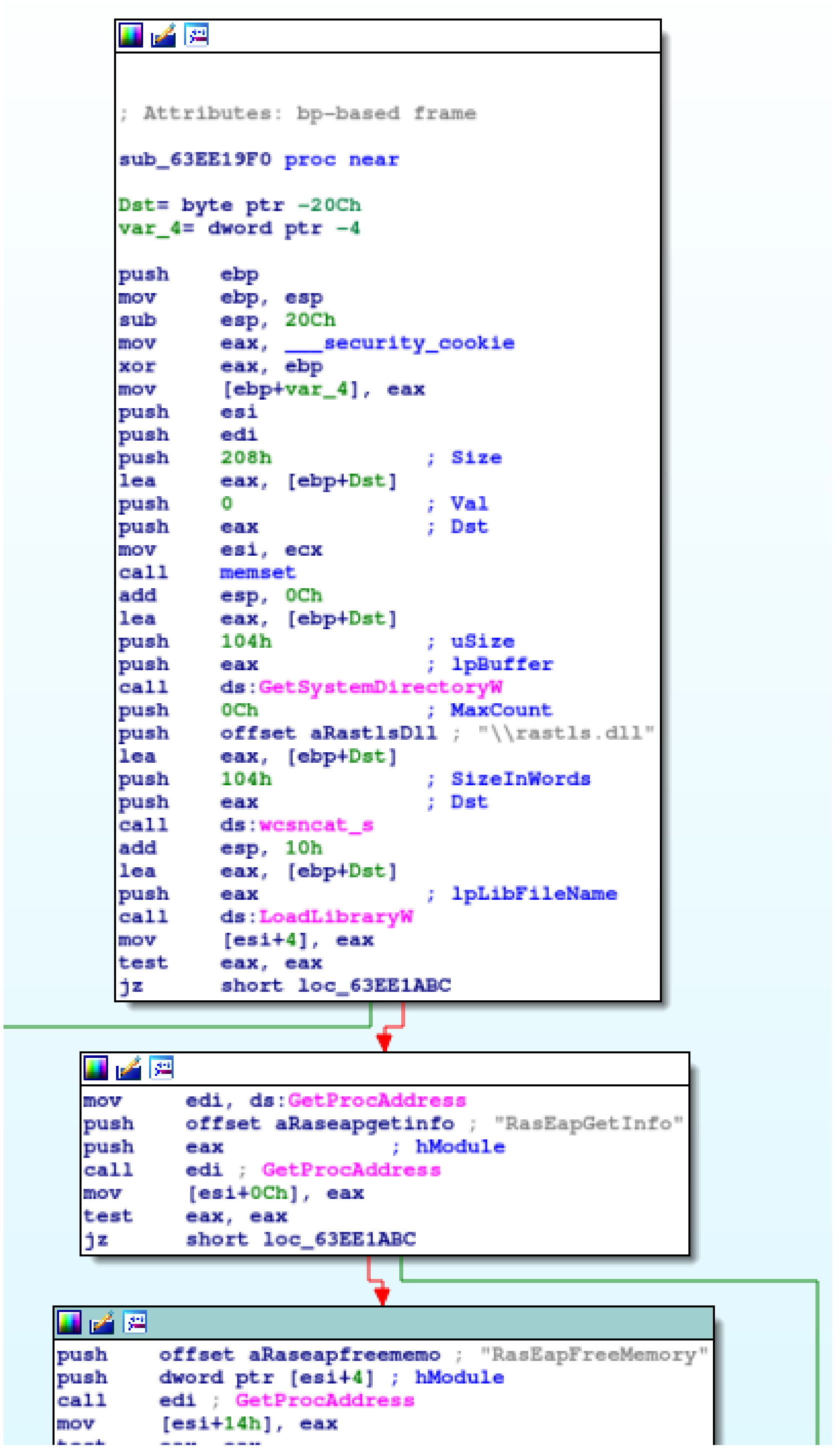
```
Buffer= word ptr -414h
LibFileName= word ptr -20Ch
var_4= dword ptr -4

sub     esp, 414h
mov     eax, ___security_cookie
xor     eax, esp
mov     [esp+414h+var_4], eax
push    ebx
push    edi
push    104h                ; uSize
lea     eax, [esp+420h+Buffer]
push    eax                 ; lpBuffer
xor     ebx, ebx
call    ds:GetWindowsDirectoryW
mov     edi, ds:LoadLibraryW
push    offset LibFileName ; "RasTls.dll"
call    edi ; LoadLibraryW
cmp     eax, ebx
mov     [esi+4], eax
jnz     short loc_4019B1
```

```
push    offset LibFileName ; "RasTls.dll"
lea     ecx, [esp+420h+Buffer]
push    ecx
push    offset aSSystem32S ; "%s\\system32\\%s"
lea     edx, [esp+428h+LibFileName]
push    104h
push    edx
call    sub_403561
add     esp, 14h
lea     eax, [esp+41Ch+LibFileName]
push    eax                 ; lpLibFileName
call    edi ; LoadLibraryW
cmp     eax, ebx
mov     [esi+4], eax
jz      short loc_401A1B
```

```
loc_4019B1:
mov     ecx, [esi+4]
mov     edi, ds:GetProcAddress
push    offset ProcName ; "RasEapGetInfo"
push    ecx                 ; hModule
call    edi ; GetProcAddress
cmp     eax, ebx
mov     [esi+0Ch], eax
jz      short loc_401A1B
```

```
mov     edx, [esi+4]
push    offset aRaseapfreememo ; "RasEapFreeMemory"
push    edx                 ; hModule
call    edi ; GetProcAddress
cmp     eax, ebx
mov     [esi+14h], eax
jz      short loc_401A1B
```

Dascgosrky.exe loading the malicious



```
; Attributes: bp-based frame

sub_63EE19F0 proc near

Dst= byte ptr -20Ch
var_4= dword ptr -4

push    ebp
mov     ebp, esp
sub     esp, 20Ch
mov     eax, ___security_cookie
xor     eax, ebp
mov     [ebp+var_4], eax
push    esi
push    edi
push    208h            ; Size
lea     eax, [ebp+Dst]
push    0               ; Val
push    eax             ; Dst
mov     esi, ecx
call    memset
add     esp, 0Ch
lea     eax, [ebp+Dst]
push    104h            ; uSize
push    eax             ; lpBuffer
call    ds:GetSystemDirectoryW
push    0Ch             ; MaxCount
push    offset aRastlsDll ; "\\rastls.dll"
lea     eax, [ebp+Dst]
push    104h            ; SizeInWords
push    eax             ; Dst
call    ds:wcsncat_s
add     esp, 10h
lea     eax, [ebp+Dst]
push    eax             ; lpLibFileName
call    ds:LoadLibraryW
mov     [esi+4], eax
test    eax, eax
jz      short loc_63EE1ABC
```

```
mov     edi, ds:GetProcAddress
push    offset aRaseapgetinfo ; "RasEapGetInfo"
push    eax             ; hModule
call    edi ; GetProcAddress
mov     [esi+0Ch], eax
test    eax, eax
jz      short loc_63EE1ABC
```

```
push    offset aRaseapfreememo ; "RasEapFreeMemory"
push    dword ptr [esi+4] ; hModule
call    edi ; GetProcAddress
mov     [esi+14h], eax
```

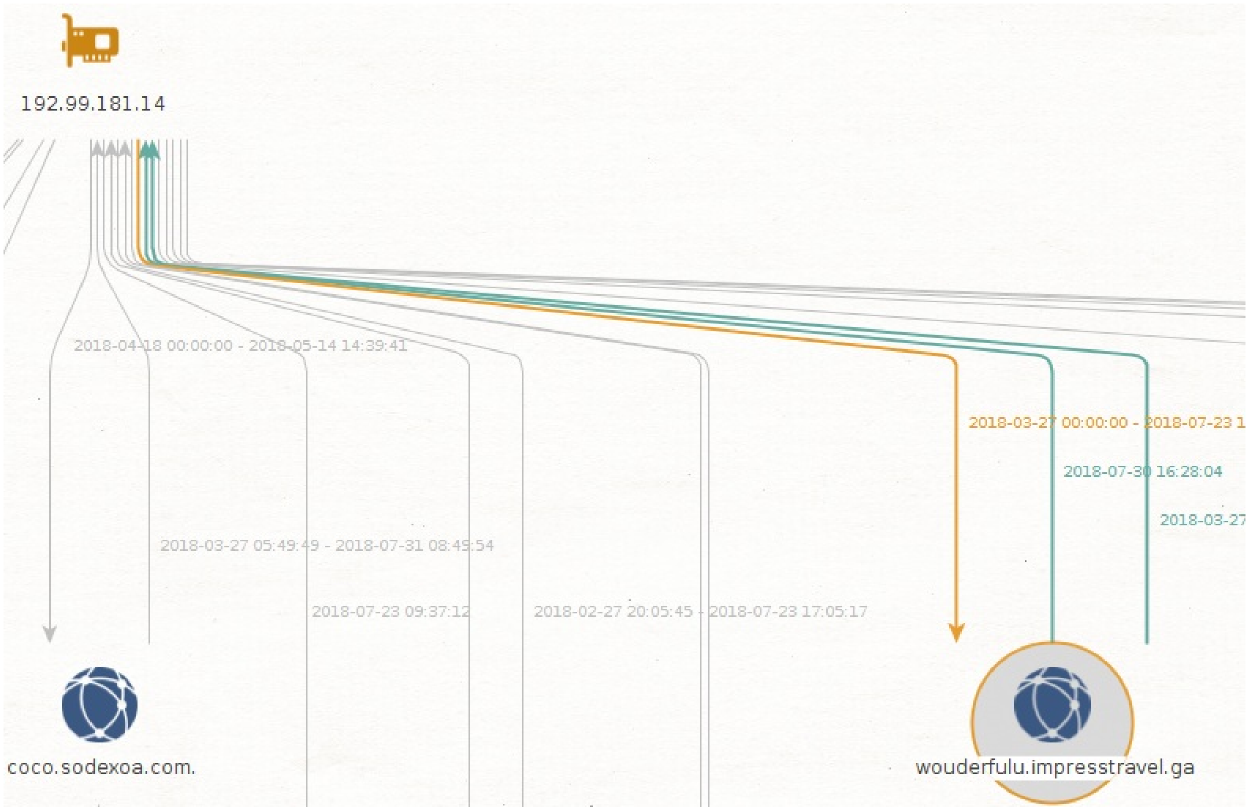
The original file

If we check the exports in IDA, we just have a dllentrypoint. The dll is executed like this.

We'll analyse the RAT in the second Part.

Infrastructure of Attackers

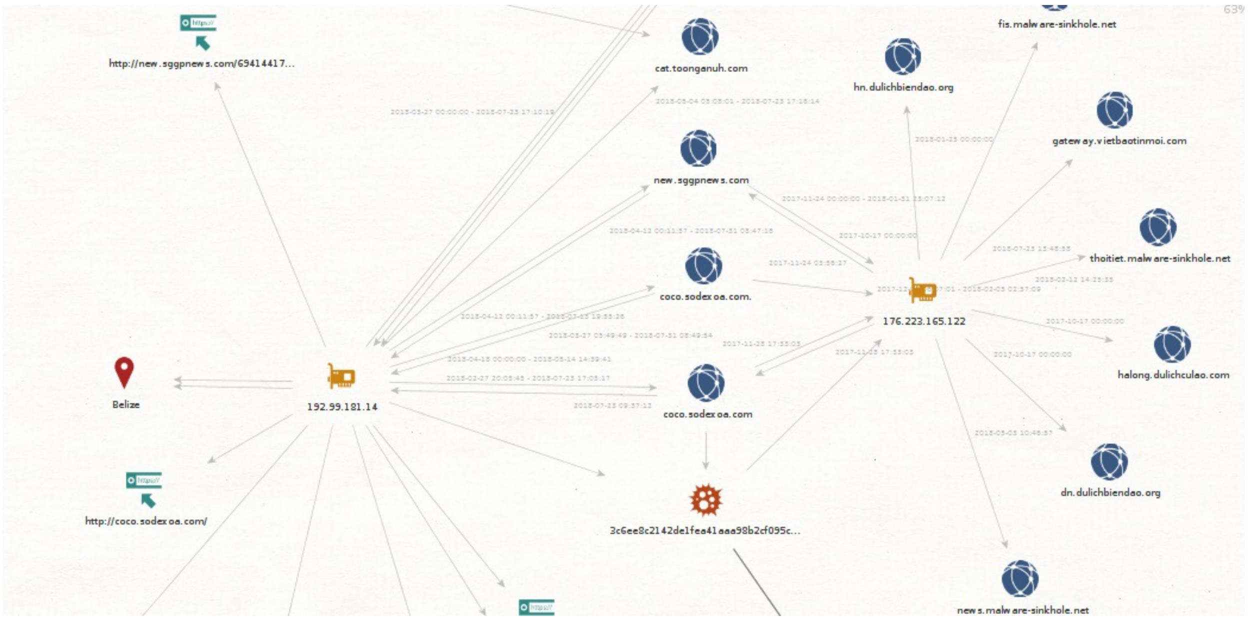
The domain contacted is wouderfulu.impresstravel.ga and this domain resolved on 192.99.181.14.



Domain wouderfulu.impresstravel.ga

This IP has differents domains found with PassiveTotal and theses domains is recorded in the IP 176.223.165.122.

Many domain names is used for Vietnameses people.



Expansion of domains

There are two domains really interesting:

Halong.dulichculao.com is already used in the campaign targeting Vietnameses organizations.

https://www.fortinet.com/blog/threat-research/rehashed-rat-used-in-apt-campaign-against-vietnamese-organizations.html

For Fortinet is the Chinese hacking group 1937CN.

If we compare the TTPs, it’s really similar. They used RTFs to make the intrusion and dll hijacking to load the real payload.

And the name of domains are really similar between the campaings.

The second one is:

Cat.toonganuh.com is a subdomain of tooganuh.com recorded by florence1972@scryptmail.com

Conclusion

The Chinese hacking group 1937CN continues to target Vietnam officials with the same TTPs with a refreshing on the tools used. The toolset used by this group to create RTF malicious document has the same properpy of the SideWinder.

I want to thank my buddies on “Zone de Confort”. It’s with this dreamteam, I can finalize correctly this analyses.

In the second part, we analyze the RAT using in this campaign. Or if another reverse can make that, I’ll paid a beer ;)

IOCs for the paper:

domains:
dn.dulichbiendao.org
gateway.vietbaotinmoi.com

web.thoitietvietnam.org
hn.dulichbiendao.org
halong.dulichculao.com
cat.toonganuh.com
new.sggpnews.com
dulichculao.com
coco.sodexoa.com.
thoitiet.malware-sinkhole.net
wouderfulu.impresstravel.ga
toonganuh.com
coco.sodexoa.com

IPs:
192.99.181.14
176.223.165.122

RTFs:

42162c495e835cdf28670661a53d47d12255d9c791c1c5653673b25fb587ffed

8.t:

2c60d4312e4416745e56048ee35e694a79e1bc77e7e4d0b5811e64c84a72d2d7

PE:

f9ebf6aeb3f0fb0c29bd8f3d652476cd1fe8bd9a0c11cb15c43de33bbce0bf68 (exe)

9f5da7524817736cd85d87dae93fdbe478385baac1c0aa3102b6ad50d7e5e368 (dll)

Update:

The payload is PlugX. Thanks to Gabor Szappanos
https://twitter.com/GaborSzappanos/status/1024622354582908928

Update IOCs:

597c0c6f397eefb06155abdf5aa9a7476c977c44ef8bd9575b01359e96273486 59.rtf
11f38b6a69978dad95c9b1479db9a8729ca57329855998bd41befc364657d654 RasTls.dll
f9ebf6aeb3f0fb0c29bd8f3d652476cd1fe8bd9a0c11cb15c43de33bbce0bf68 RasTls.exe

b70069e1c8e829bfd7090ba3dfbf0e256fc7dfcefc6acafb3b53abcf2caa2253 b7.rtf
77361b1ca09d6857d68cea052a0bb857e03d776d3e1943897315a80a19f20fc2
spoolsver.exe
9fba998ab2c1b7fec39da9817b27768ba7892c0613c4be7c525989161981d2e2 vsodscpl.dll

9d239ddd4c925d14e00b5a95827e9191bfda7d59858f141f6f5dcc52329838f0 9d.rtf
087d8bee1db61273a7cd533d52b63265d3a8a8b897526d7849c48bcdba4b22ec RasTls.dll
f9ebf6aeb3f0fb0c29bd8f3d652476cd1fe8bd9a0c11cb15c43de33bbce0bf68 RasTls.exe

332aa26d719a20f3a26b2b00a9ca5d2e090b33f5070b057f4950d4f088201ab9 rtf

93aa353320a8e27923880401a4a0f3760374b4d17dcd709d351e612d589b969d
vsodscpl.dll

77361b1ca09d6857d68cea052a0bb857e03d776d3e1943897315a80a19f20fc2 ScnCfg.exe