Sebdraven Follow

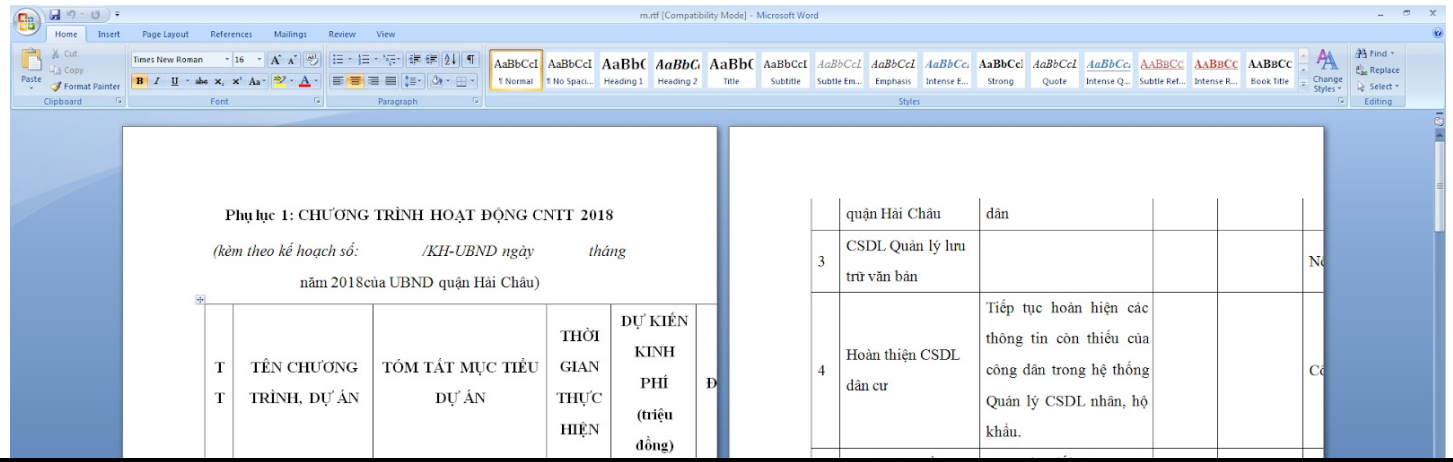
OSINT, Python,Malware Analysis, Botnet Tracker, SIEM and IPS/IDS and Threats Expert / co-organizer #BotConf / co-creator of #FastIR

Jul 31 · 9 min read

Malicious document targets Vietnamese officials

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.



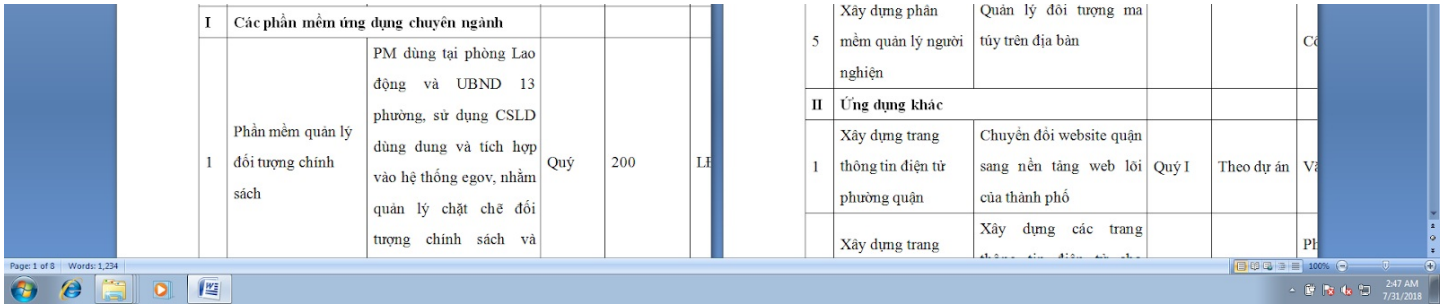
Phụ lục 1: CHƯƠNG TRÌNH HOẠT ĐỘNG CNTT 2018

(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

	quận Hải Châu	dân			
3	CSDL Quản lý lưu trữ văn bản				Nơ
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.			Có



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	LH
5	Xây dựng phần mềm quản lý người nghiện	Quản lý đối tượng ma túy trên địa bàn			Có
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	Và
	Xây dựng trang	Xây dựng các trang			Ph

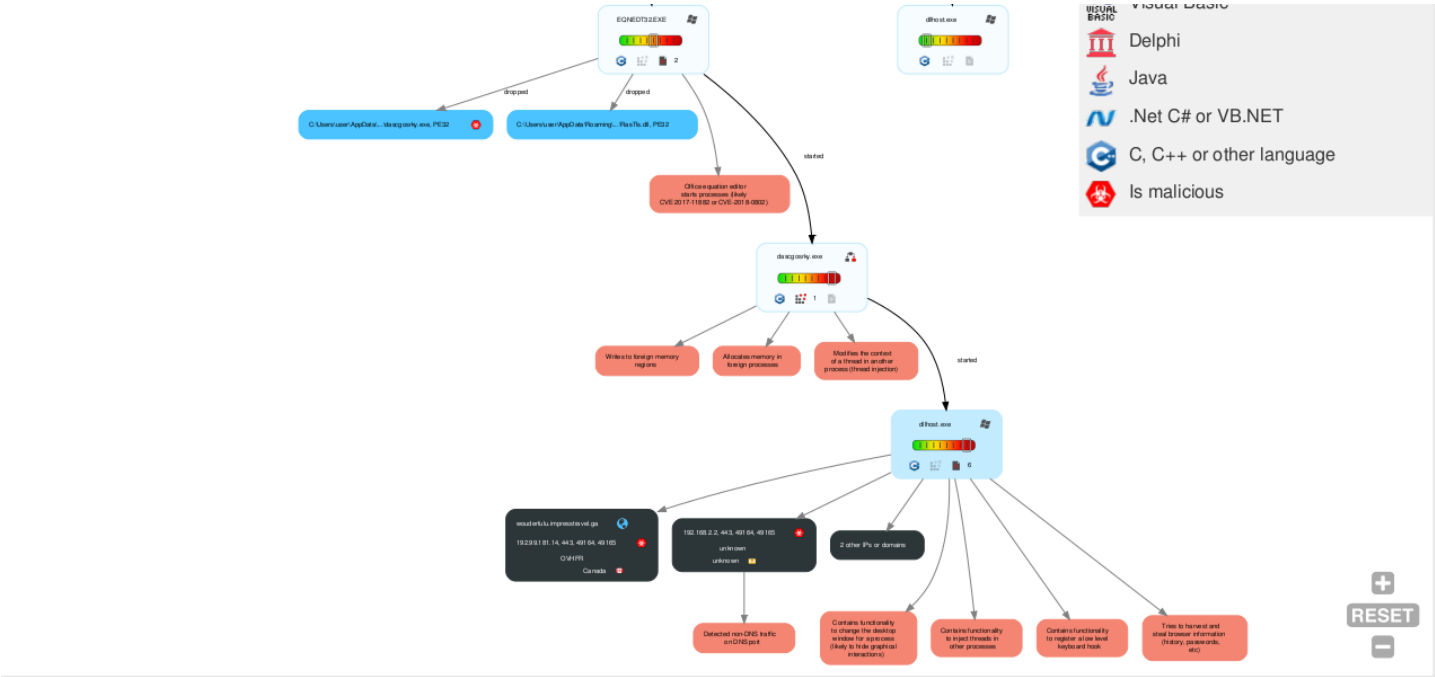
RTF document

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:

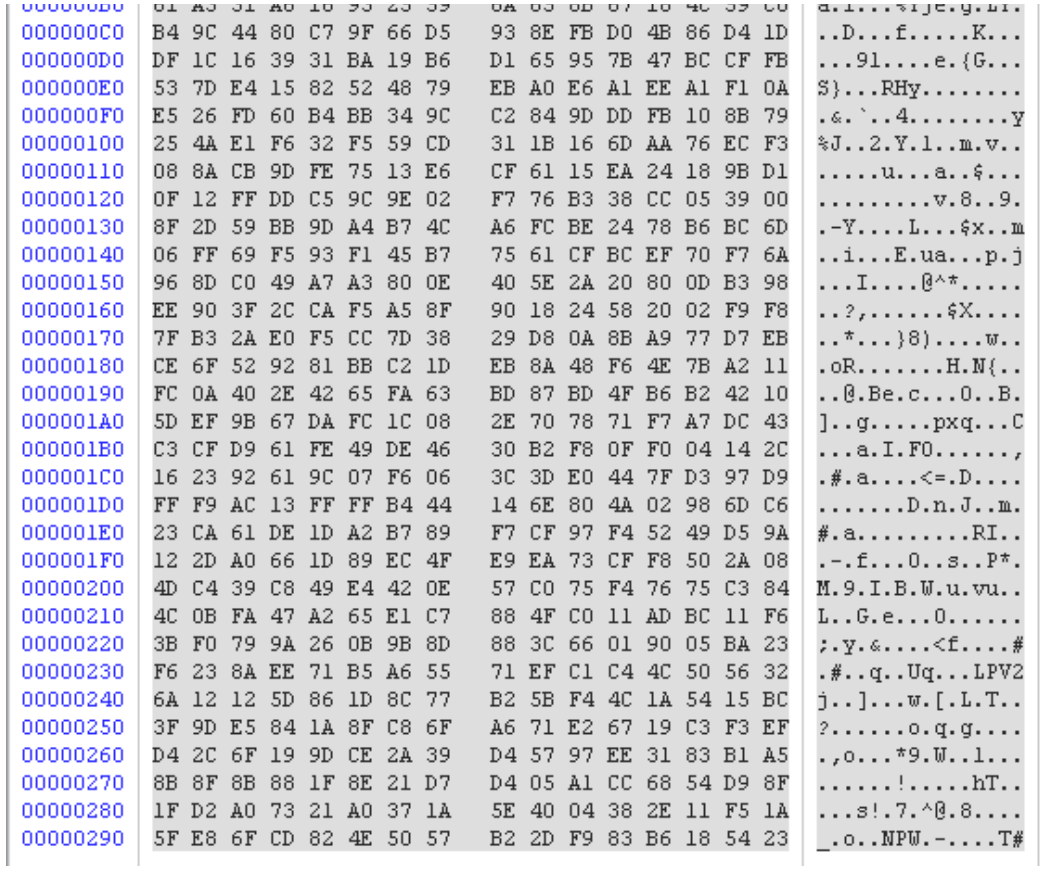
two non well formed ole object and a third named package ole object.

```
================================================================================
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 39   6A 65 6B 67 16 4C 39 C6  a.1...%9je.g.L9.
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.

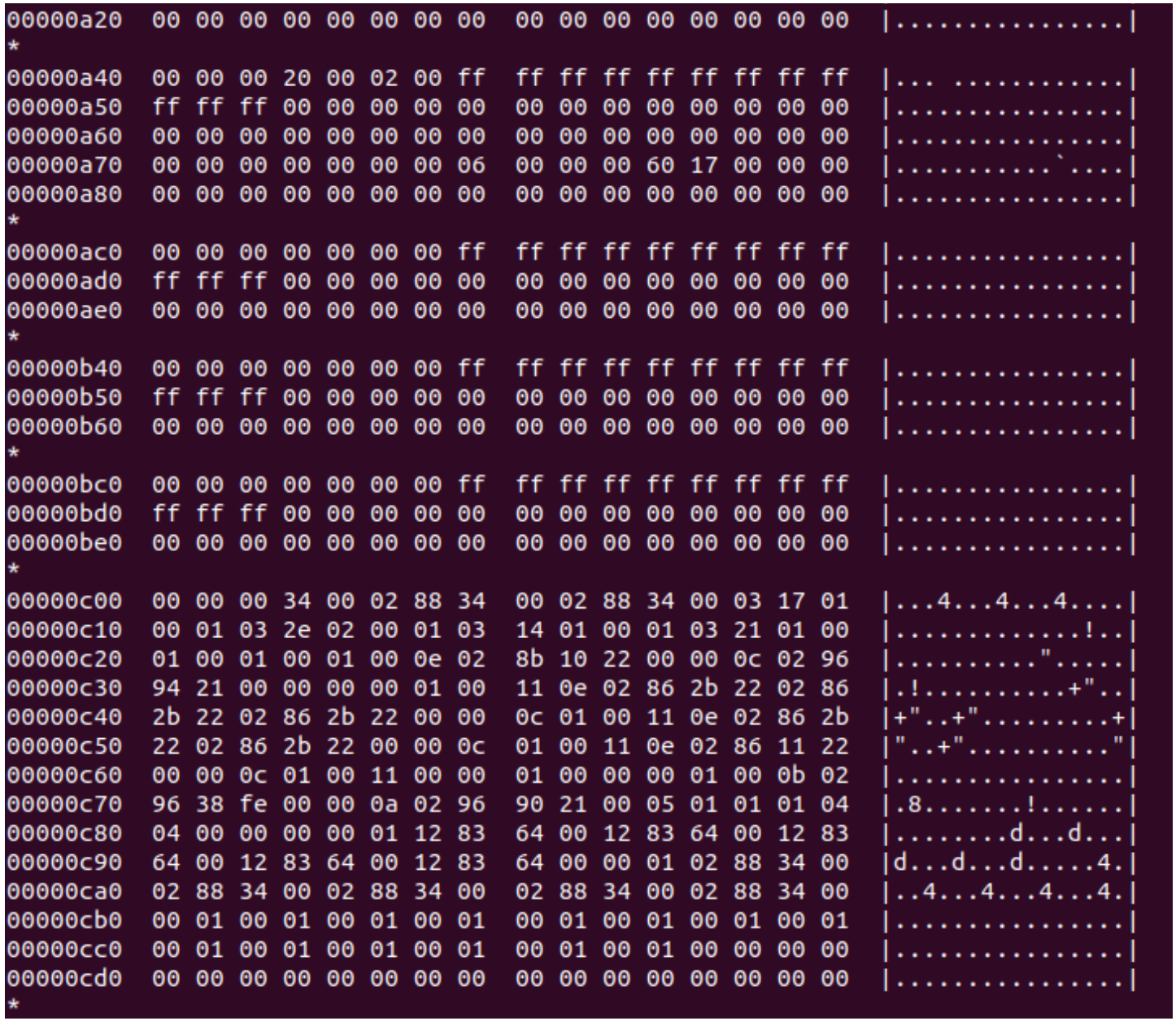
If we check it, it's clearly encrypted.

```
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

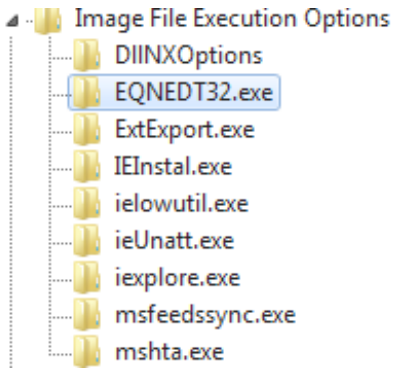
At the start of the analysis, we think the process EQNEDT32.exe is created by Winword.exe using the function CreateProcess. So we decided to set a breakpoint at the call of his function.

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 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		

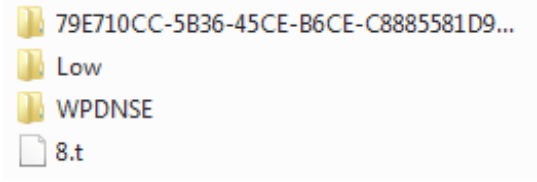
EQNEDT32.exe process attached by the debugger

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.

CreateFile is called at call eqnedt32.41E5EE.

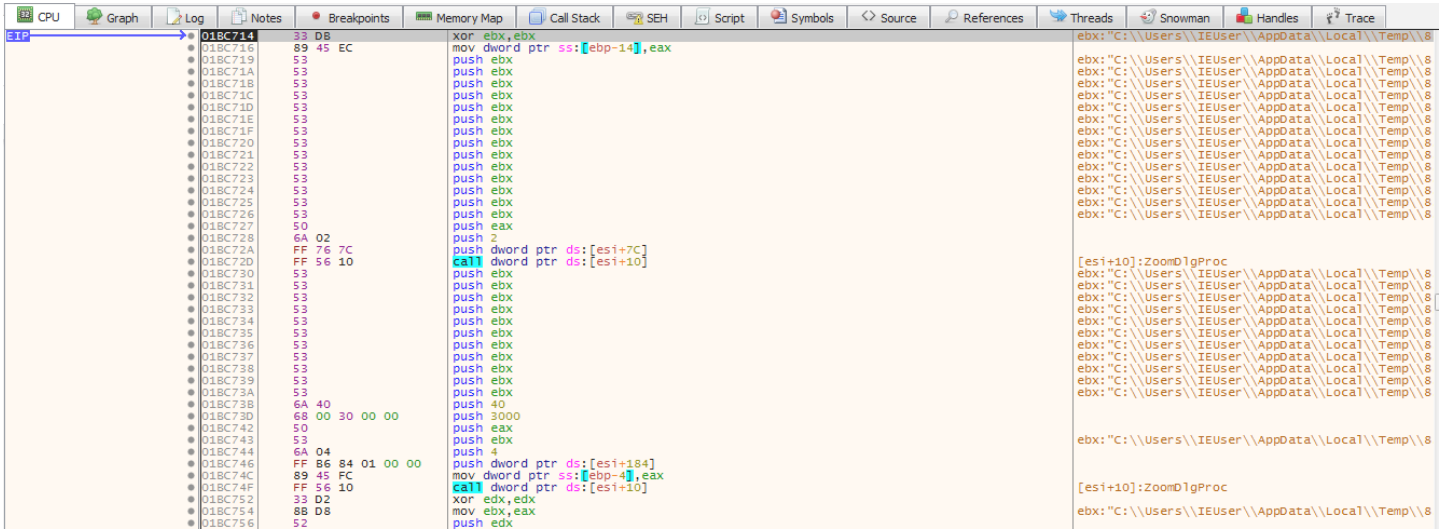
The param of the path of file is pushed on the stack push dword ptr ss:[ebp-4].

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.

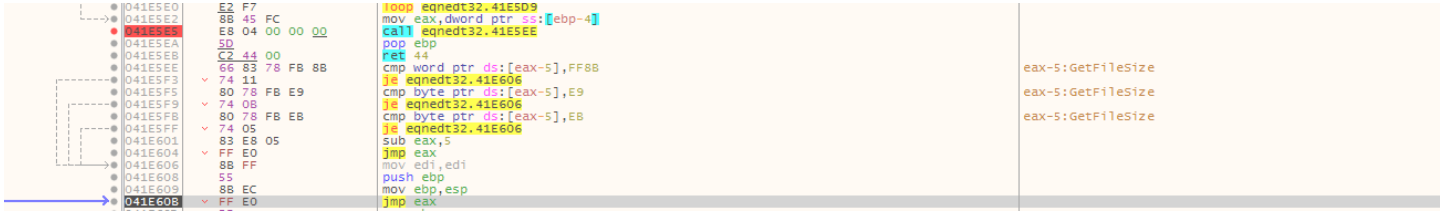
```
041E5E5    E8 04 00 00 00    call eqnedt32.41E5EE
041E5EA    5D                pop ebp
041E5EB    C2 44 00          ret 44
```

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



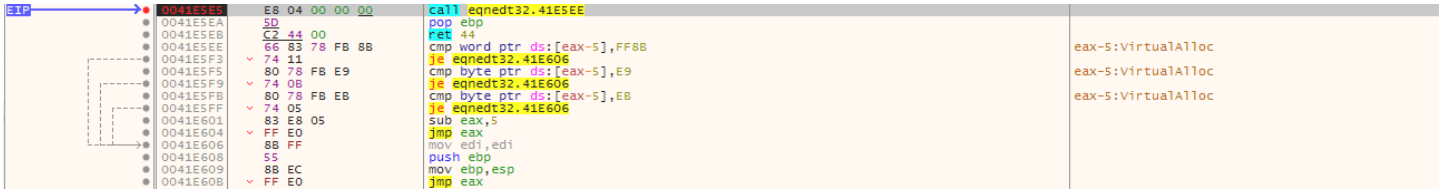
Shellcode of the exploit

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



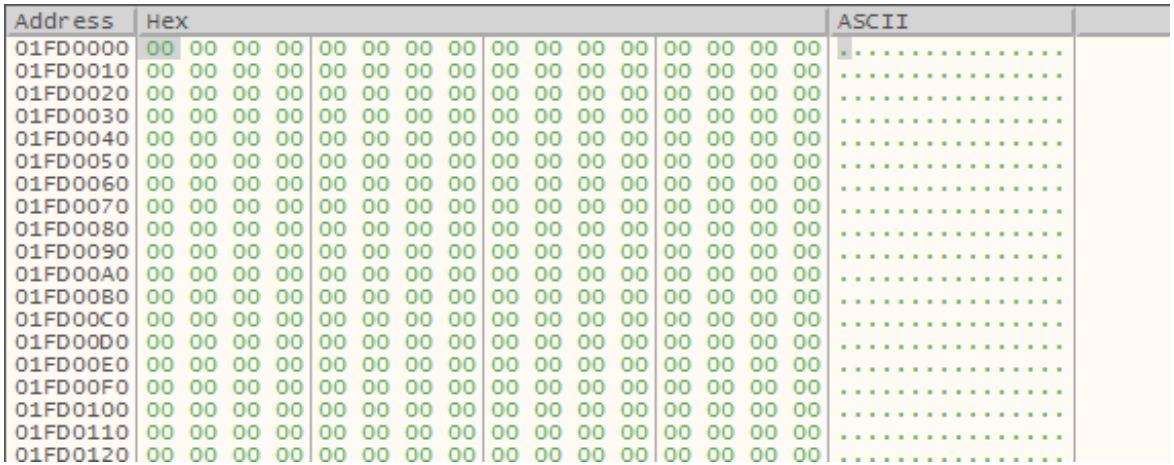
Get the size of the file

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



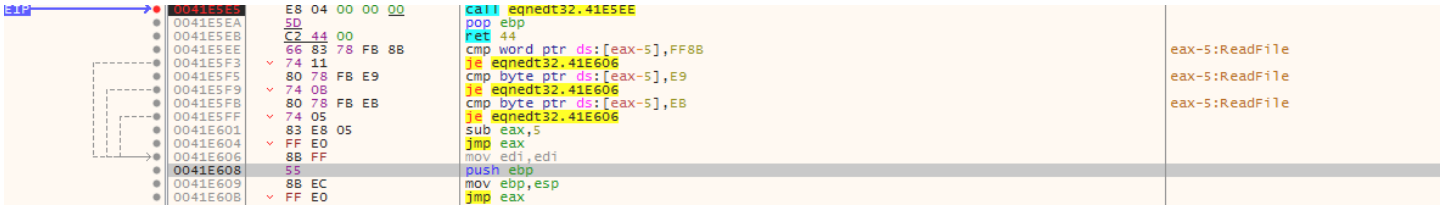
VirtualAlloc memory page to load 8.t

After virtualAlloc, the memory page is pointed by EAX



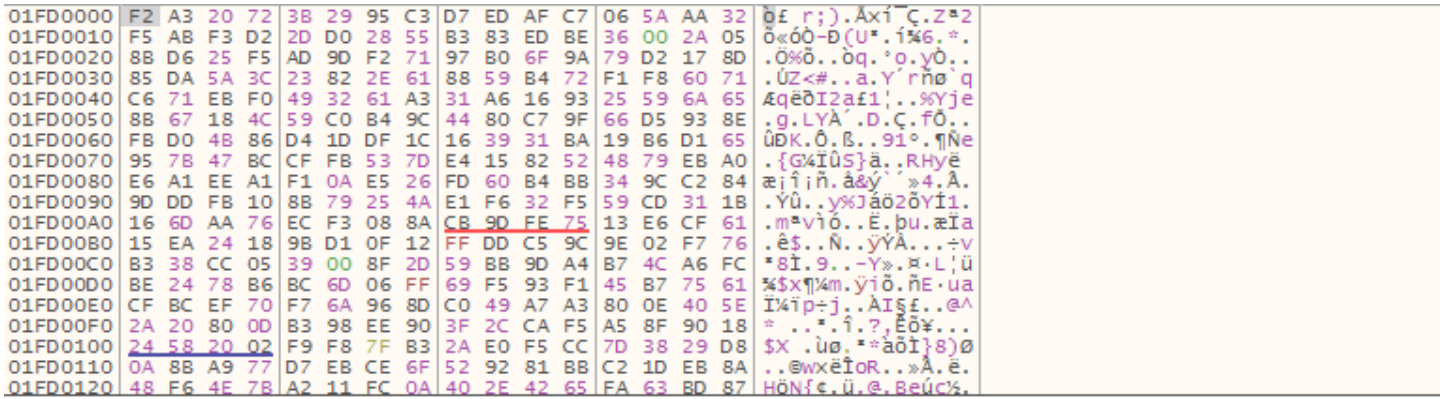
The page allocated

ReadFile is called:



Readfile 8.t

And 8.t is loaded at 1FD0000:

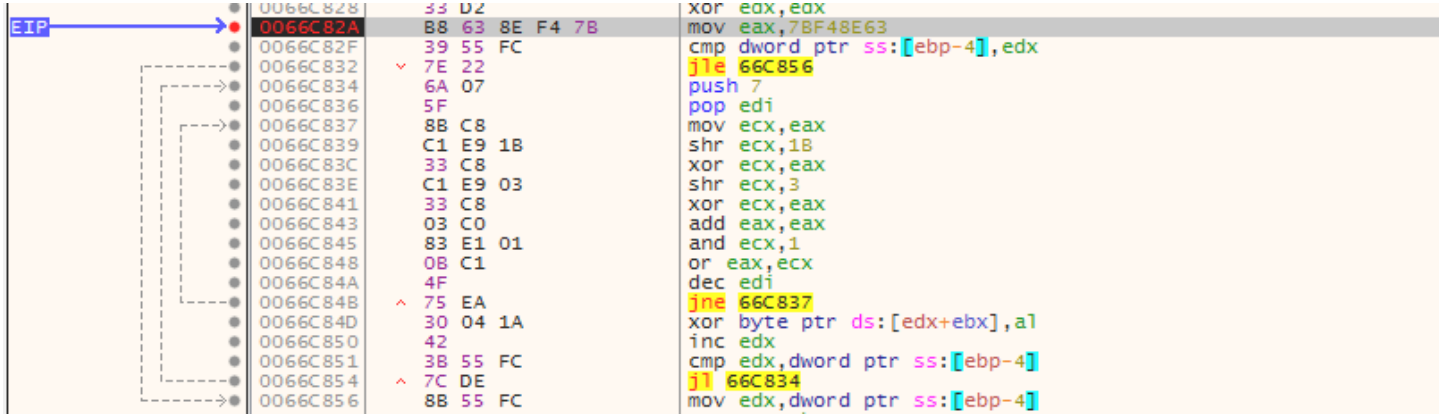


8.t in memory

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

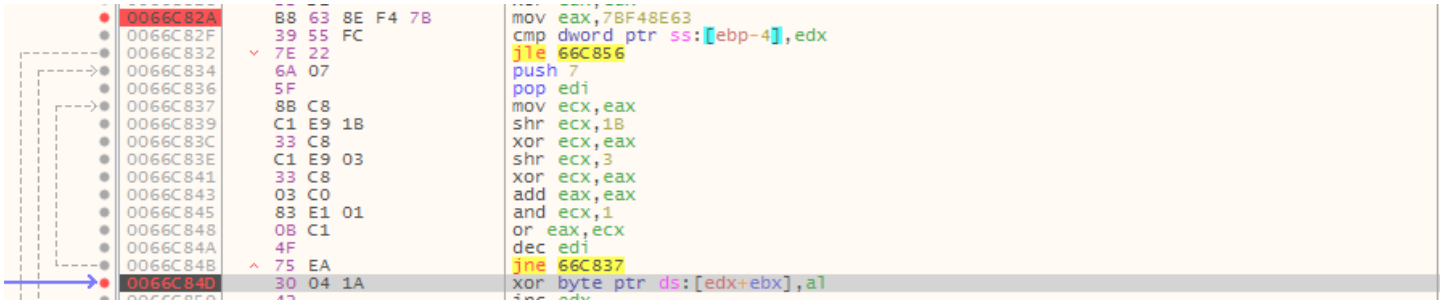
The loop of decryption is a xoring with different manipulations on the decryption key.

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



Decryption loop

And the xor is made after key manipulation.



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;).Ãx¯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.

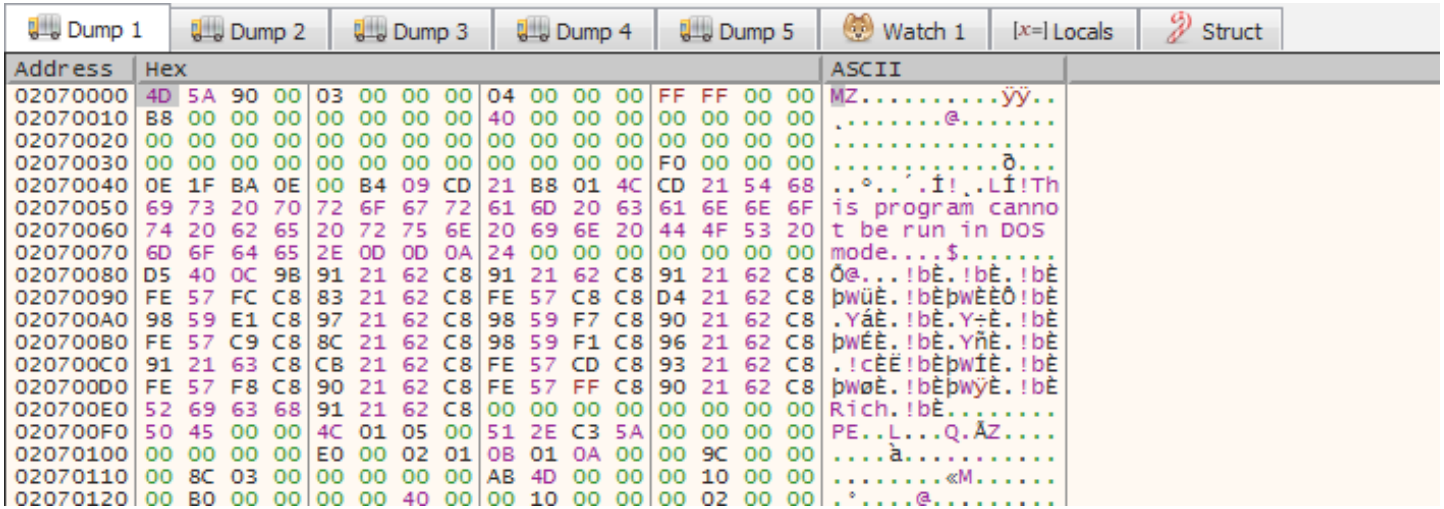
the file 8.t has been decrypted.

```
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.

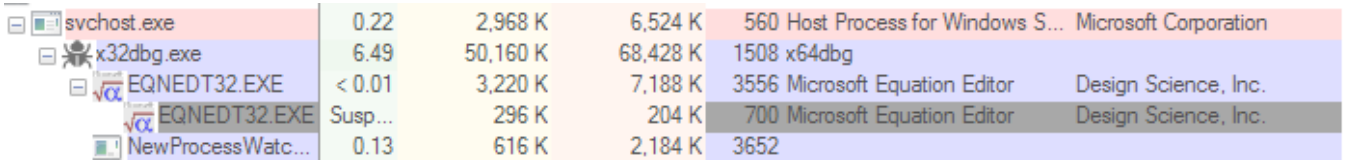
And the new PE at 01FD0000 is copied at this address.



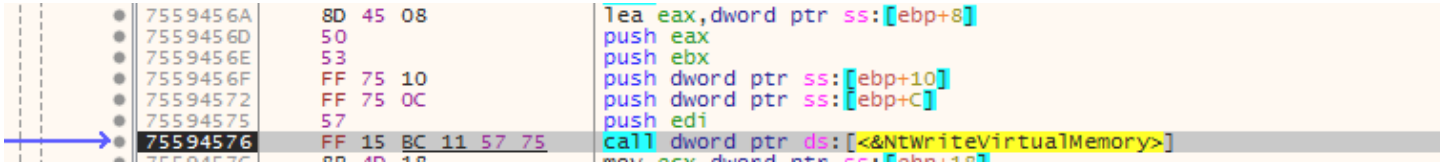
the PE decrypted is copied in the new memory page

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

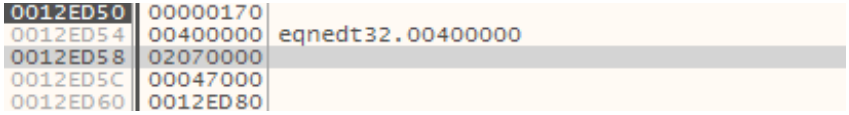
And EQNEDT32.exe is forked in suspend status by a CreateProcess and the shellcode overwrite it by the PE at the address 02070000



Fork of EQNEDT32.exe



Overwritting of EQNEDT32.exe



Stack used by NTWriteVirtualMemory

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

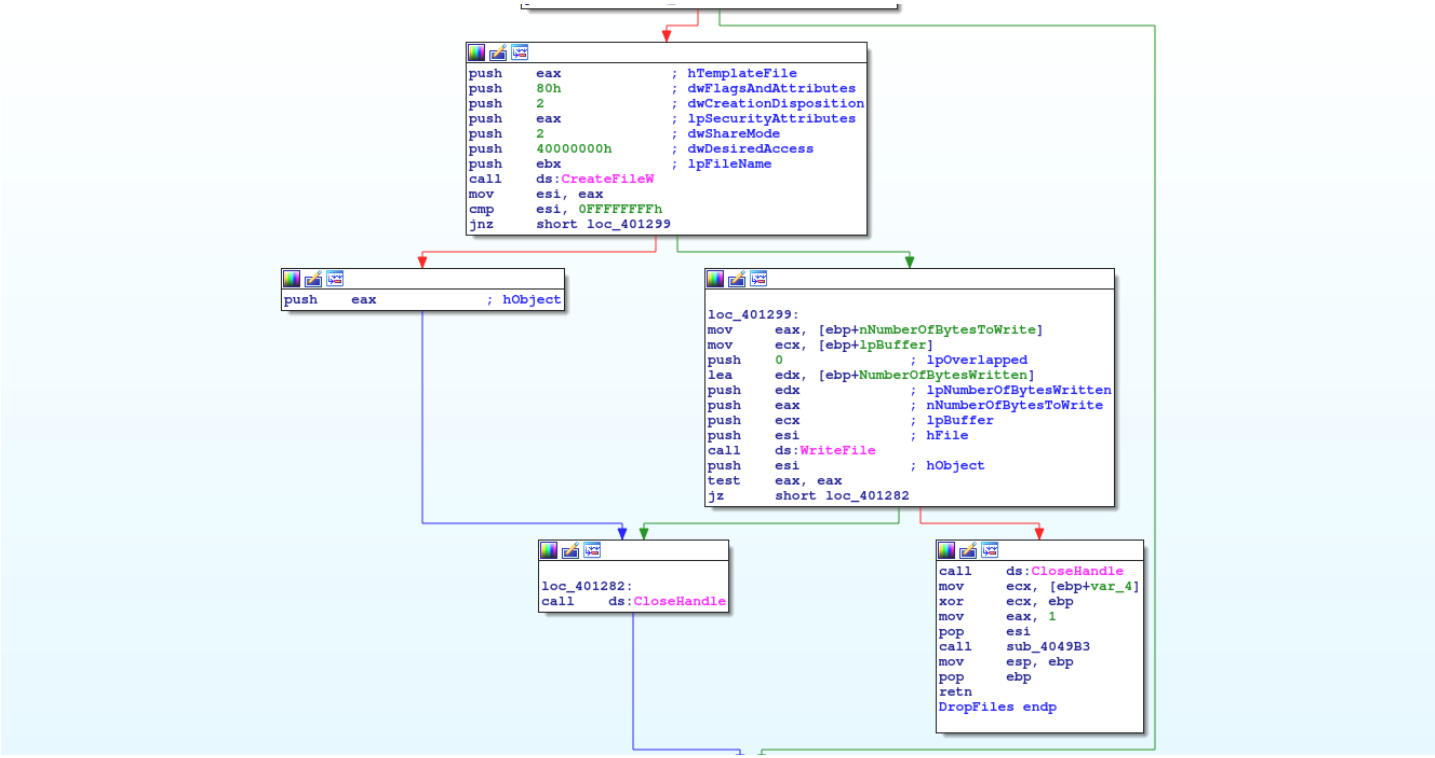
So, We've found all API Calls in the object ole at the beginning and we have a runPE to launch the new EQNEDT32.exe overwritten.

Analysing the fork of EQNEDT32.exe

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

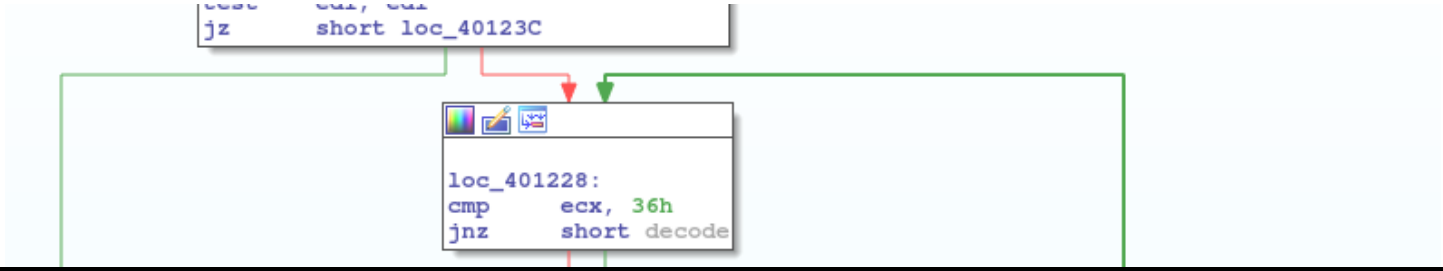
- A dll named RasTls.dll
- A executable file named dascgosrky.exe

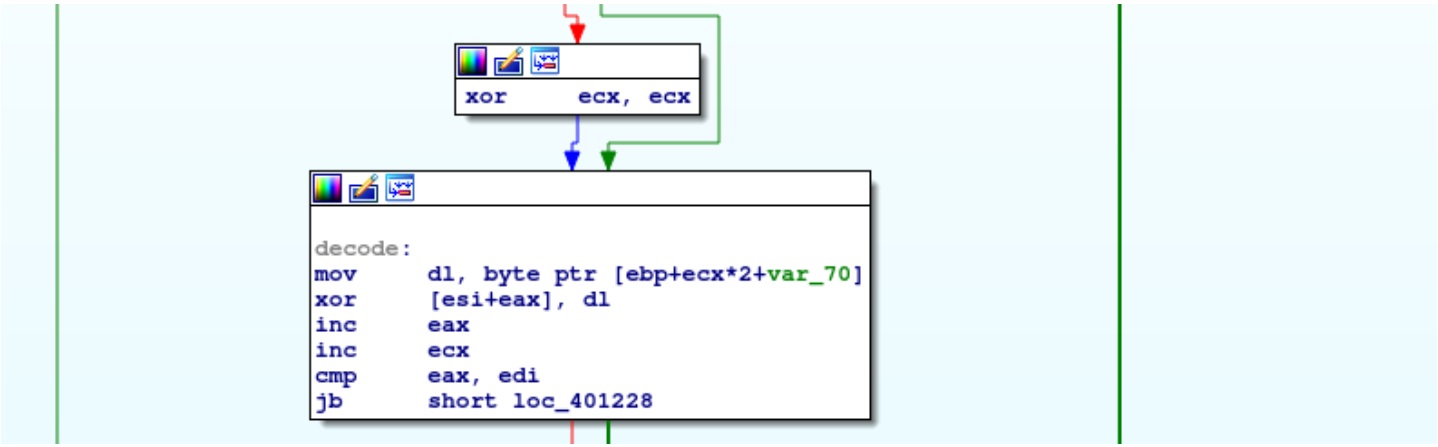
If we dump EQNEDT32.exe and we put in IDA, we found quickly the function that drops the files on disk (sub_00401150) renamed dropFiles.



DropFiles Fucntion

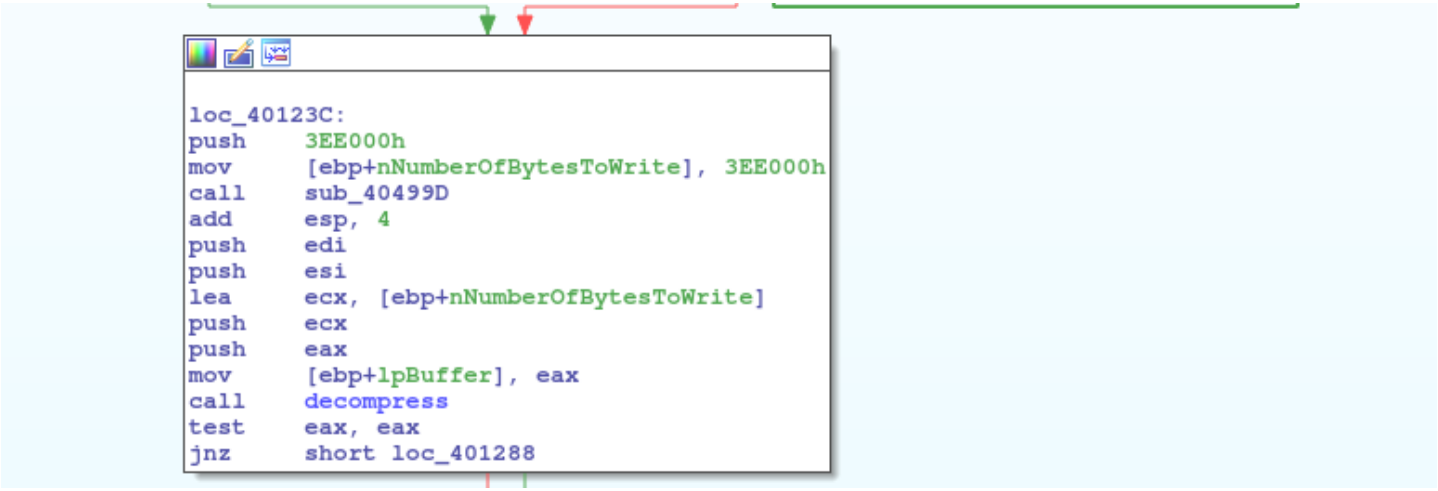
And at the start of this functions, we have a loop with a xor.





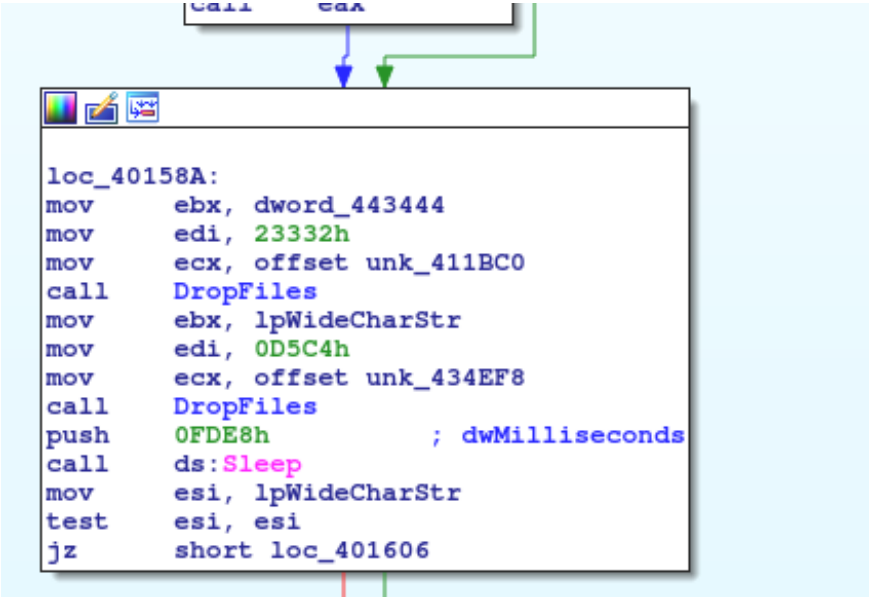
Second loop of decryption

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



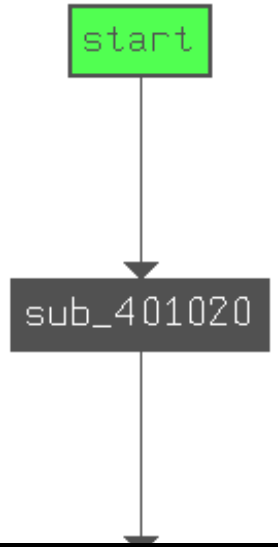
Decompression function used zlib

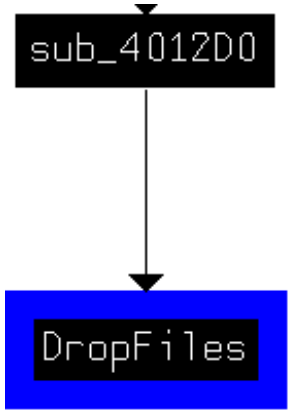
The function dropFiles is called twice by the sub_4012D0.



Drop the dll and the executable

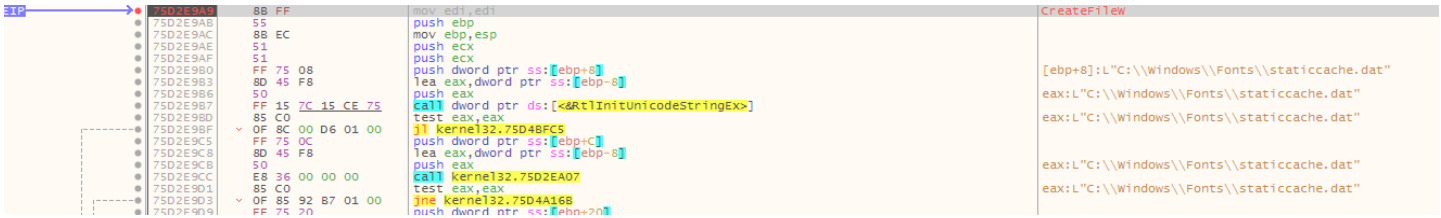
If we check the call graph, DropFiles is called only by the function sub_4012D0.





Functions using DropFiles function

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



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 00401504 when DropFiles is called

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

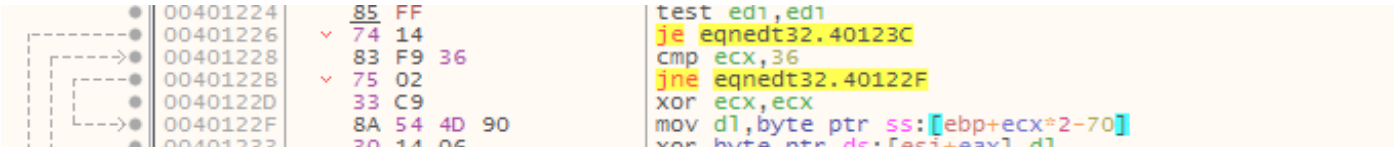
Breakpoint to the first call of DropFiles

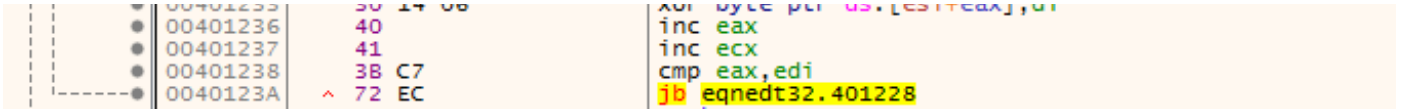
And now we can analyse the second loop of decryption.

The first step is the initialization of the decryption function.

Set for the second loop encryption

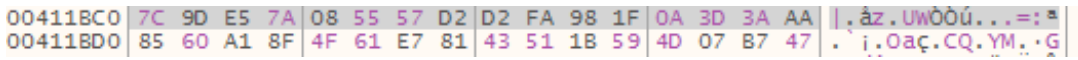
And after we find the xor and store the result in esi+eax.





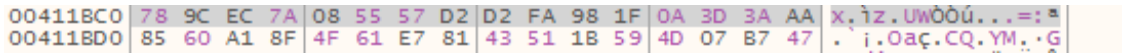
Decryption loop

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



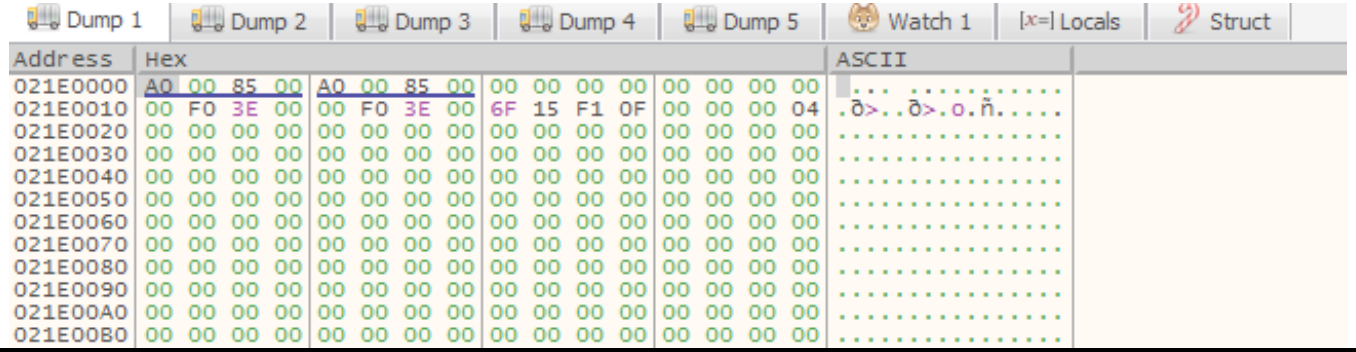
Before the decryption

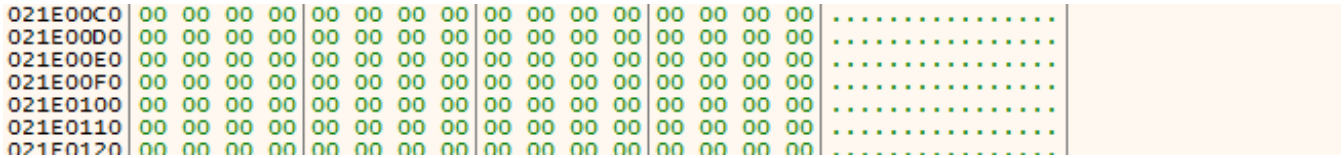
After tree loops, we obtains the header of zlib compressed object.



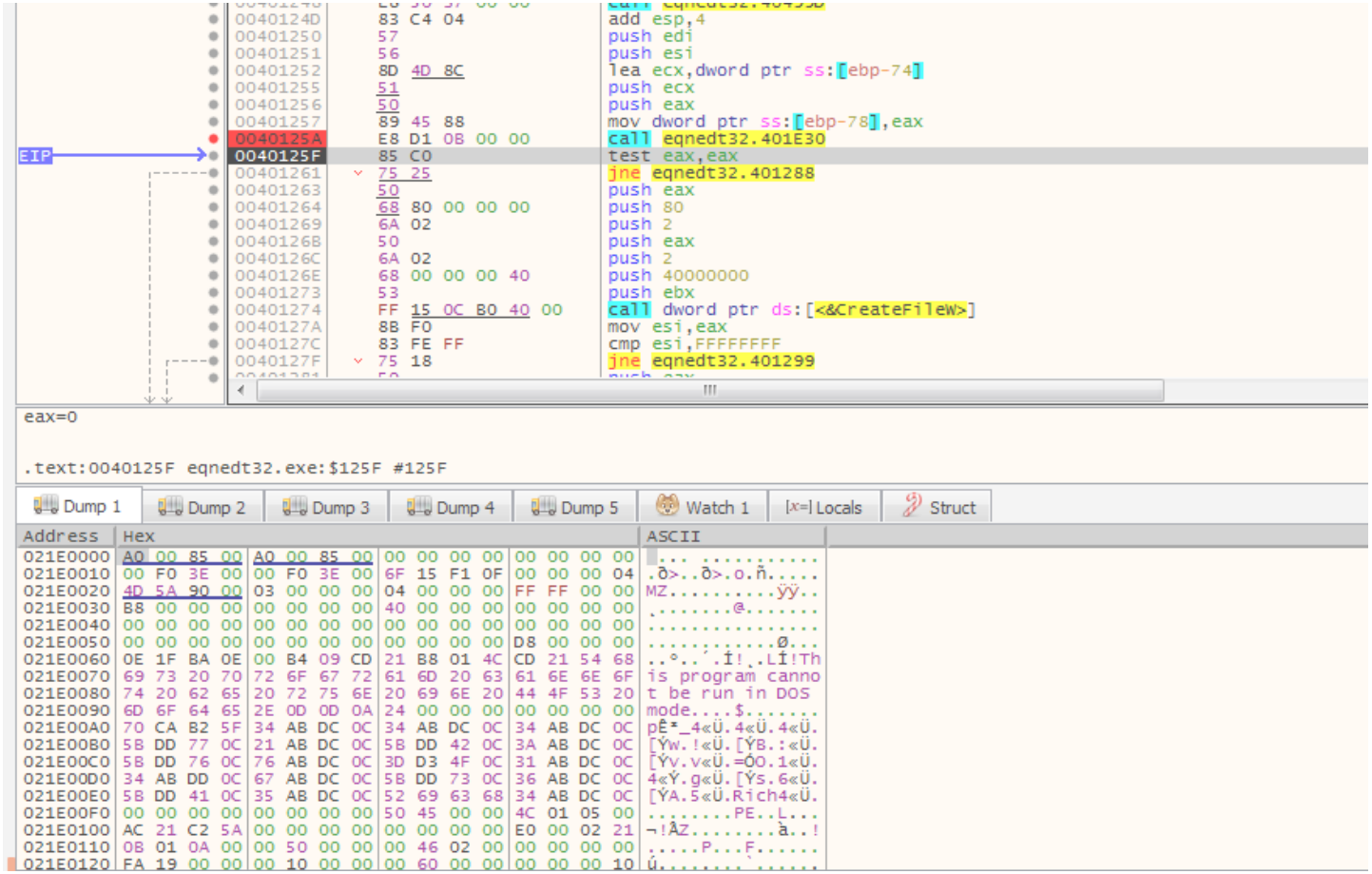
After the decryption

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





Page memory allocated to store the dll



After decompression

And after the file is created with the following path:

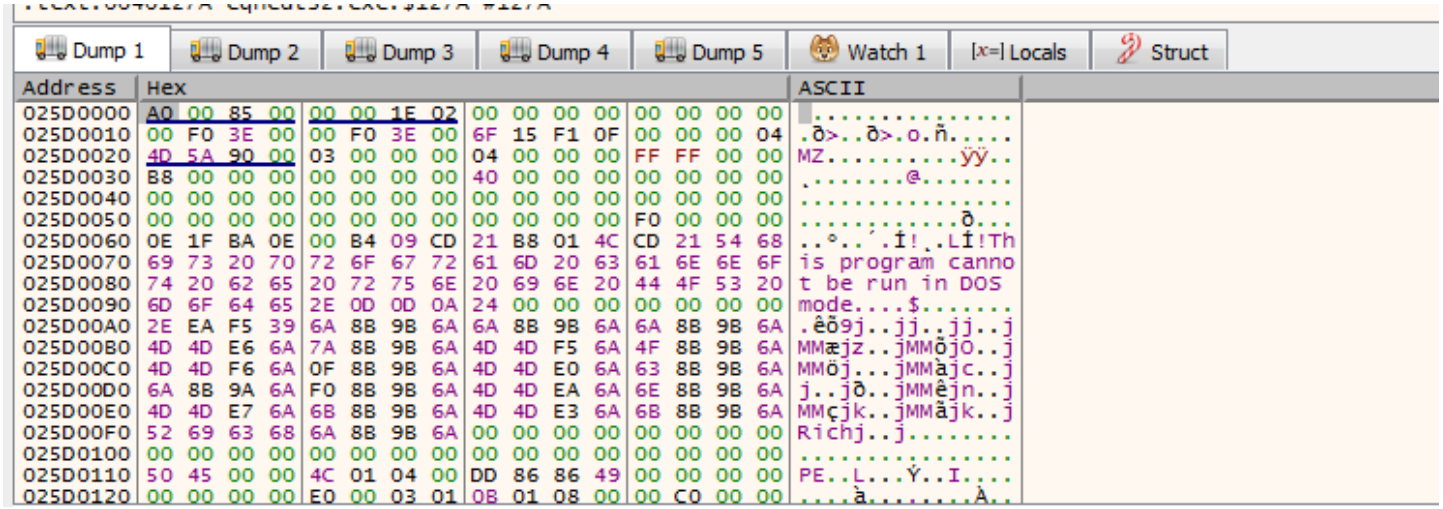
L”C:\\Users\\IEUser\\AppData\\Roaming\\Microsoft\\Windows\\Netw

ork Shortcuts\\RasTls.dll”

Stored by ebx.

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

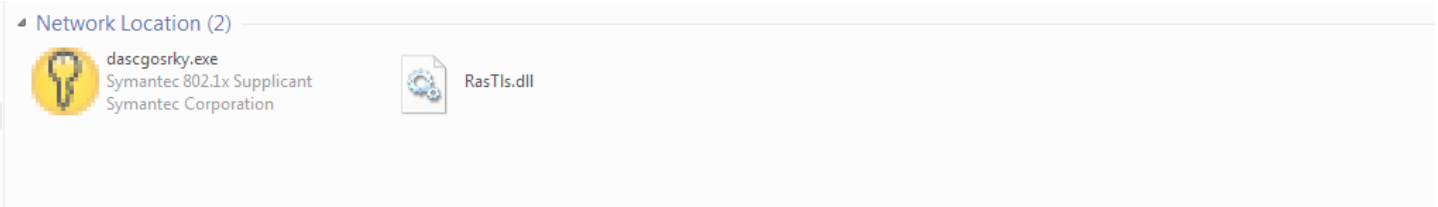
The offset where store the file is 00434EF8 and the pe decompressed is stored at 025D0020



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”

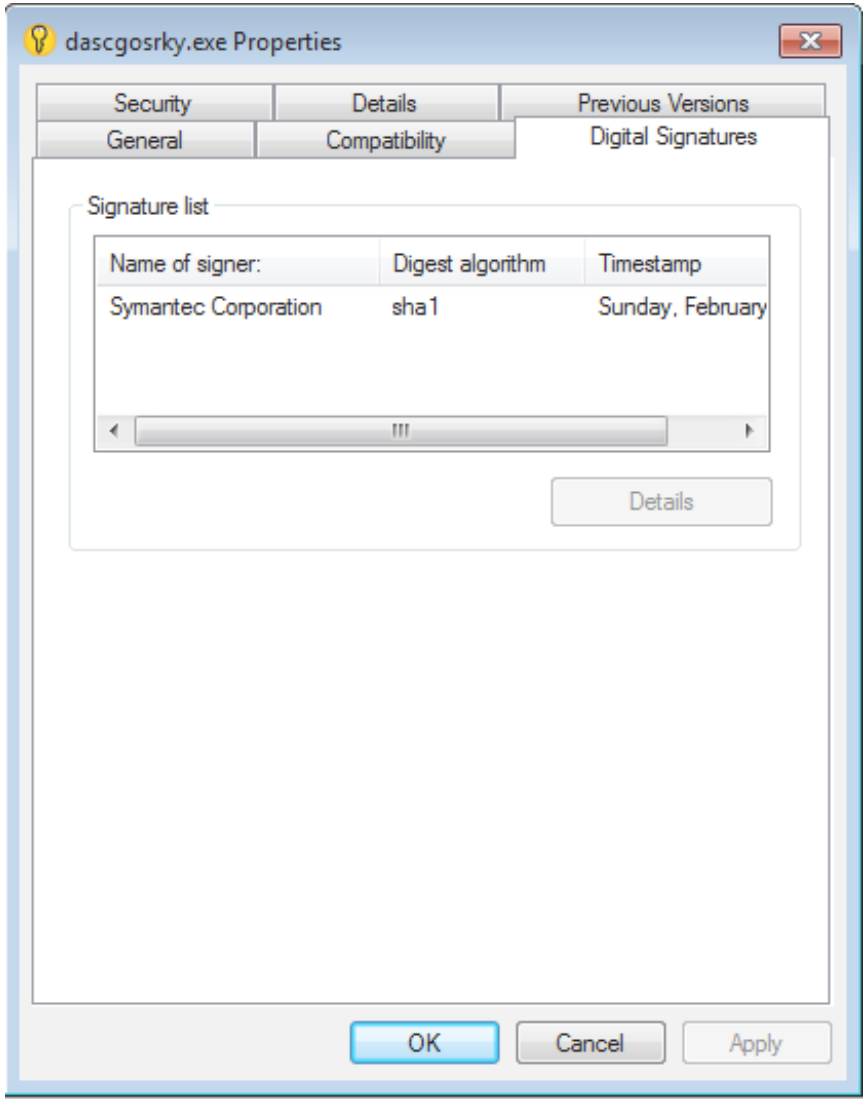
So we have two files in networks shortcuts of Windows.



Files drops on disk

dll hijacking

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



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

```
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
```

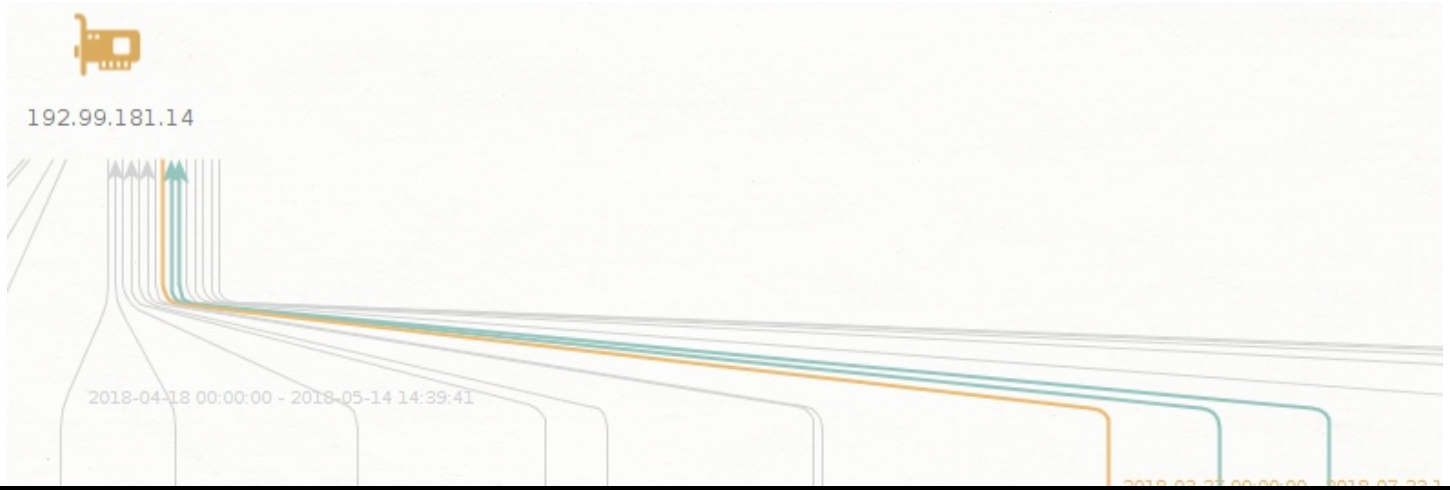
The original file

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

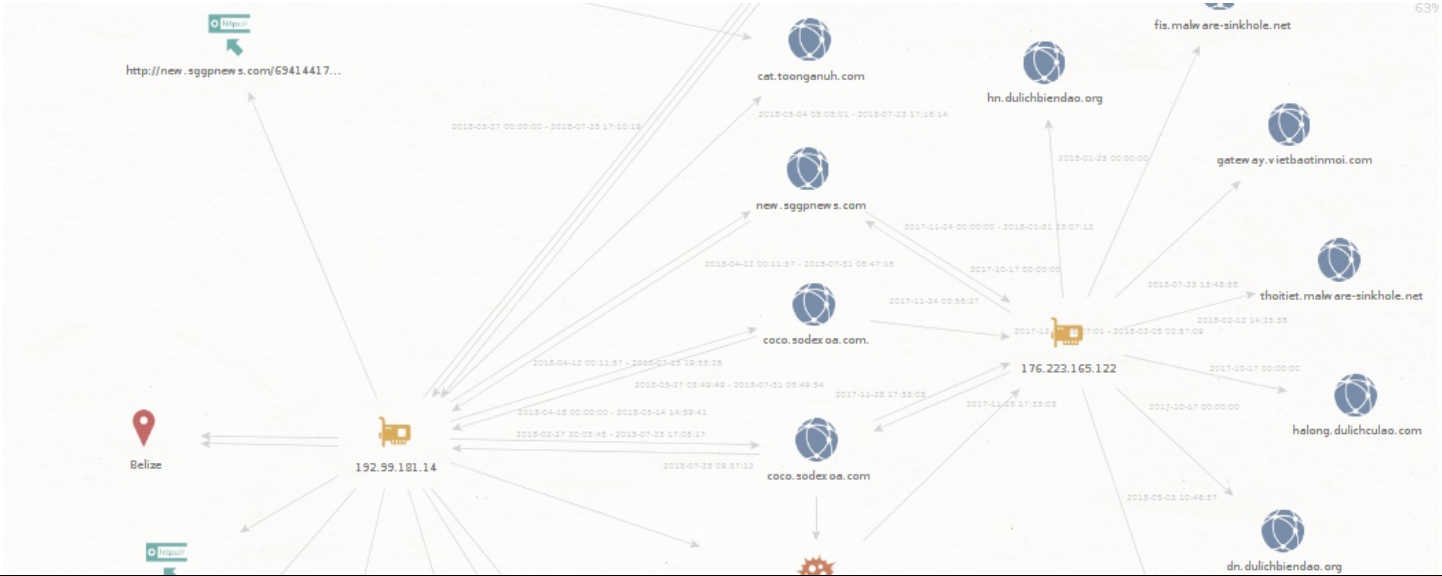
The domain contacted is wouderfulu.impresstravel.ga and this domain resolved on 192.99.181.14.



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 for 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:

domains:
dn.dulichbiendao.org
gateway.vietbaotinmoi.com
fis.malware-sinkhole.net
hn.dulichbiendao.org
halong.dulichculao.com
news.malware-sinkhole.net

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)

Security Malware Analysis Threat Intelligence