

Malicious document targets Vietnamese officials

By Sebdraven

Published: 2018-08-13 · Archived: 2026-05-01 02:01:02 UTC



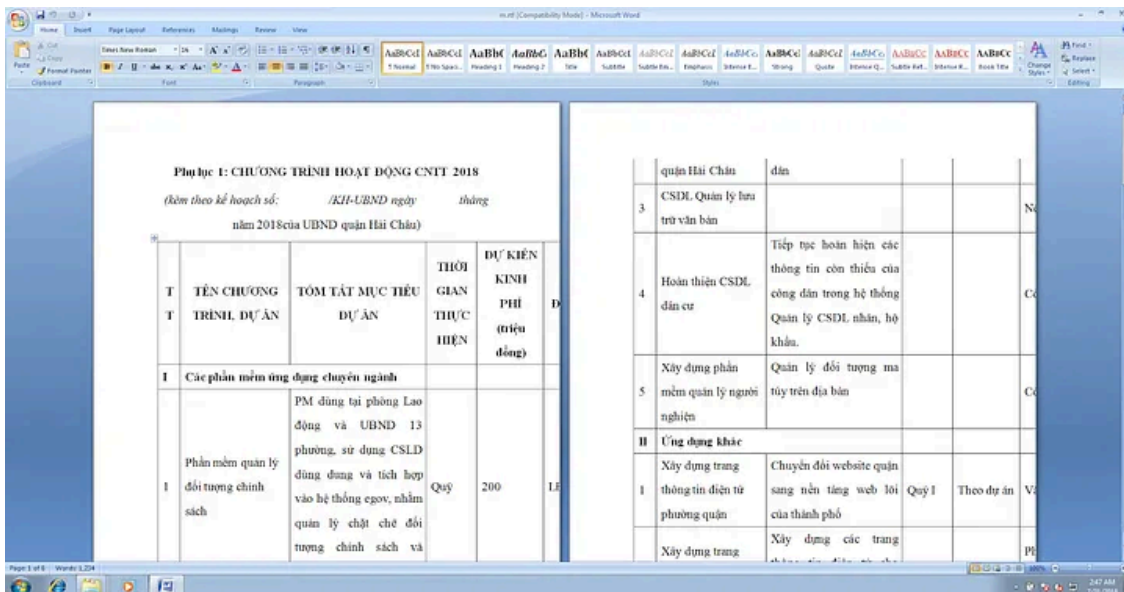
9 min read

Jul 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 different projects in the district Hải Châu of Đà Nẵng.

Press enter or click to view image in full size



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 be the Chinese hacking group 1937CN.

Infection chains

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


```

=====
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.

Press enter or click to view image in full size

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	6BPack
00000010	61	67	65	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:)
00000070	95	C3	D7	ED	AF	C7	06	5A	AA	32	F5	AB	F3	D2	2D	D0Z.2....-
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.l...%Yj.e.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	\$)...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.l..m.v..
00000110	08	8A	CB	9D	FE	75	13	E6	CF	61	15	EA	24	18	9B	D1u...a..\$...
00000120	0F	12	FF	DD	C5	9C	9E	02	F7	76	B3	38	CC	05	39	00v.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...0..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.FO.....,
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	C6D.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...0..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...0.....
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.l...<
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

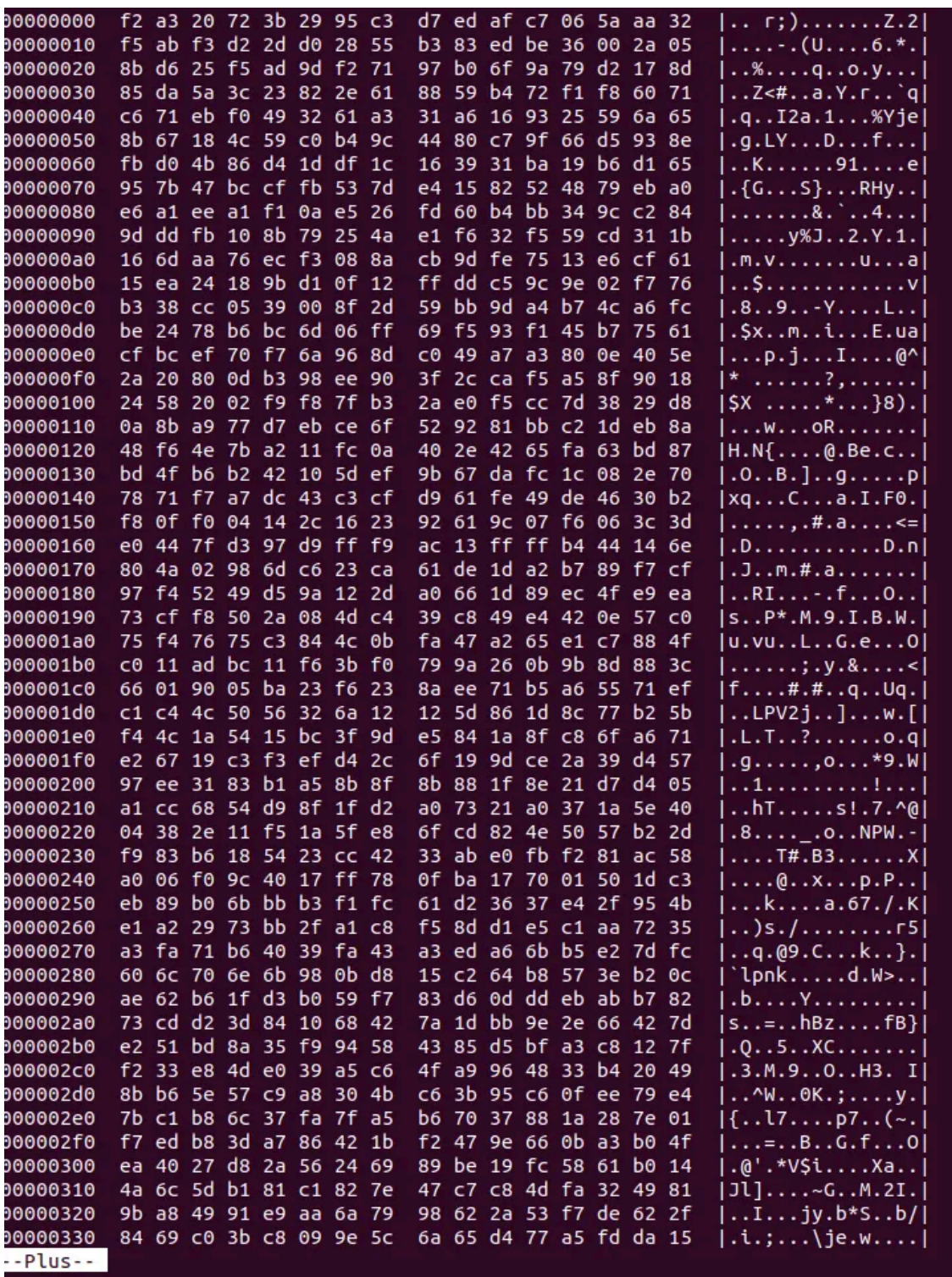
This technique is used to execute code like sct file to download an executable on the operating system. McAfee labs has detailed all this stuff with sct file: <https://securingtomorrow.mcafee.com/mcafee-labs/dropping-files-temp-folder-raises-security-concerns/>

Many attackers use it in the wild because it's very easy to use and it's 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.

Press enter or click to view image in full size



8.t encrypted

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.

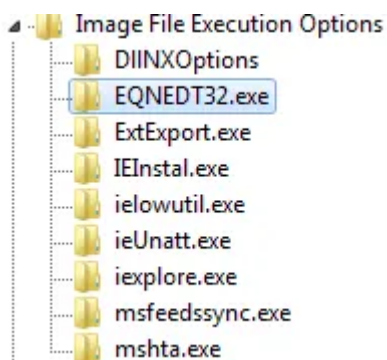
Debugging of the shellcode

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 .

Press enter or click to view image in full size

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 debugger when EQNEDT32.exe is executed

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

Press enter or click to view image in full size

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		
proceexp.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

Press enter or click to view image in full size

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		

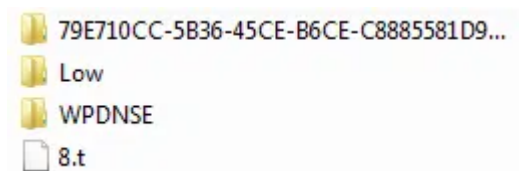
EQNEDT32.exe process attached by the debugger

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].

Press enter or click to view image in full size

```

00402000 FF 75 20      push dword ptr ss:[ebp+20]
00402003 FF 75 1C      push dword ptr ss:[ebp+1C]
00402006 FF 75 18      push dword ptr ss:[ebp+18]
00402009 FF 75 14      push dword ptr ss:[ebp+14]
0040200C FF 75 10      push dword ptr ss:[ebp+10]
0040200F FF 75 0C      push dword ptr ss:[ebp+C]
00402012 FF 75 08      push dword ptr ss:[ebp+8]
00402015 FF 75 04      push dword ptr ss:[ebp+4]
00402018 E8 12 FE FF FF call kernel32.CreateFileEx
    
```

Press enter or click to view image in full size

```

dword ptr [ebp-4]=[0012F0A4 &"C:\Users\IEUser\AppData\Local\Temp\8.t"]=001B6C3D L"C:\Users\IEUser\AppData\Local\Temp\8.t"
.text:75D2E8BF kernel32.dll:4E8BF #4E8BF
    
```

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.

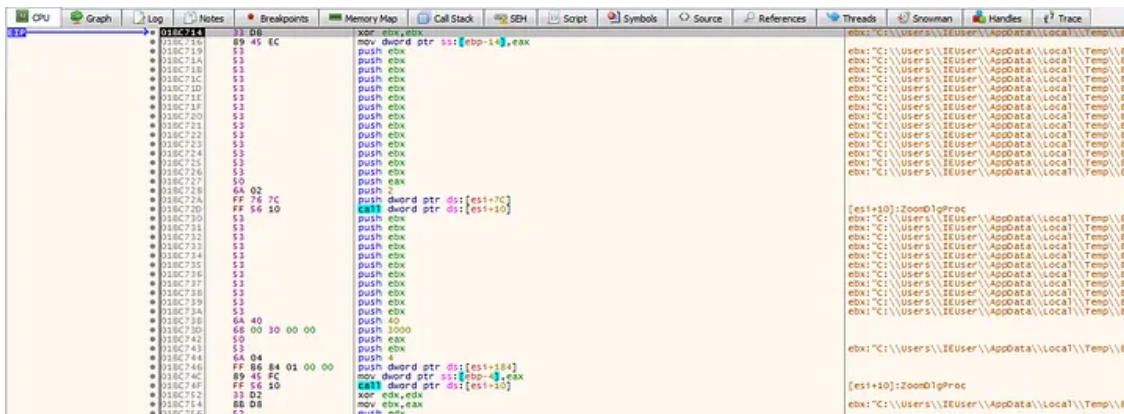
Press enter or click to view image in full size

```

0041E5E5 E8 04 00 00 00 call eqnedt32.41E5EE
0041E5EA 5D      pop ebp
0041E5EB C2 44 00 00 ret 44
    
```

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

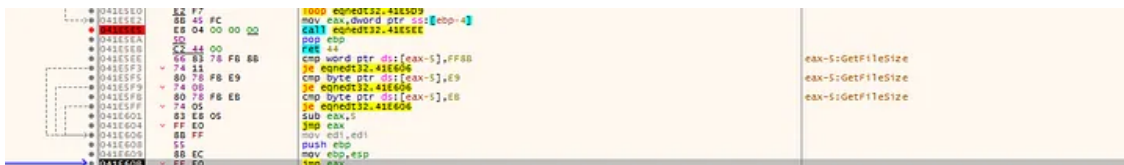
Press enter or click to view image in full size



Shellcode of the exploit

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

Press enter or click to view image in full size



Get the size of the file

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

Press enter or click to view image in full size



VirtualAlloc memory page to load 8.t

Press enter or click to view image in full size



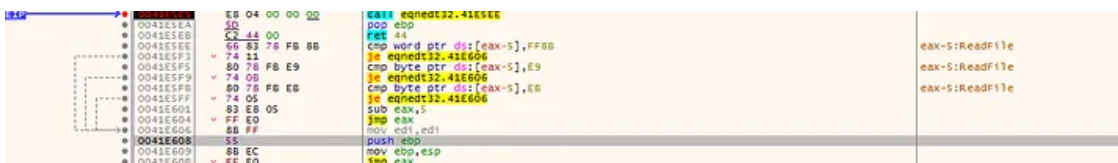
After virtualAlloc, the memory page is pointed by EAX

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:

Press enter or click to view image in full size



Readfile 8.t

And 8.t is loaded at 1FD0000:

Press enter or click to view image in full size

01FD0000	F2 A3 20 72 38 29 95 C3 D7 ED AF C7 06 5A AA 32	bf r;).Ax i C.Z*2
01FD0010	F5 AB F3 D2 2D D0 28 55 B3 83 ED BE 36 00 2A 05	0<0-D(U'.i%6.=".
01FD0020	8B D6 25 F5 AD 9D F2 71 97 80 6F 9A 79 D2 17 8D	.0%0..0q.'o.y0..
01FD0030	85 DA 5A 3C 23 82 2E 61 88 59 B4 72 F1 F8 60 71	.Uz<#.a.Y rno'q
01FD0040	C6 71 EB F0 49 32 61 A3 31 A6 16 93 25 59 6A 65	Aqe0I2af1!..%Yje
01FD0050	8B 67 18 4C 59 C0 B4 9C 44 80 C7 9F 66 D5 93 8E	.g.LYA'.D.C'.f0..
01FD0060	FB D0 48 86 D4 1D DF 1C 16 39 31 BA 19 B6 D1 65	0DK.O.B..9i°.qNe
01FD0070	95 78 47 BC CF FB 53 7D E4 15 82 52 48 79 E8 A0	.(G%IUS}a..RHyè
01FD0080	E6 A1 EE A1 F1 0A E5 26 FD 60 B4 BB 34 9C C2 84	æijñ.â&y »4.Å.
01FD0090	9D DD FB 10 8B 79 25 4A E1 F6 32 F5 59 CD 31 18	.Yu..y%}a020vI1.
01FD00A0	16 6D AA 76 EC F3 08 8A CB 9D FE 75 13 E6 CF 61	.m^vì0..E.bu.æIa
01FD00B0	15 EA 24 18 9B D1 0F 12 FF DD C5 9C 9E 02 F7 76	.è\$.N..yYA...+v
01FD00C0	B3 38 CC 05 39 00 8F 2D 59 BB 9D A4 B7 4C A6 FC	*SI.9..Y».a.L;ù
01FD00D0	8E 24 78 B6 BC 6D 06 FF 69 F5 93 F1 45 B7 75 61	%\$xq%4m.yi0.ñE.ua
01FD00E0	CF BC EF 70 F7 6A 96 8D C0 49 A7 A3 80 0E 40 5E	I%ip=j..AISf..@^
01FD00F0	2A 20 80 0D B3 98 EE 90 3F 2C CA F5 A5 8F 90 18	" ..'.i'?.E0%...
01FD0100	24 58 20 02 F9 F8 7F B3 2A E0 F5 CC 7D 38 29 D8	\$X .uo.'*a0i}s)0
01FD0110	0A 8B A9 77 D7 EB CE 6F 52 92 81 BB C2 1D EB 8A	..@wxèI0R...»A.è.
01FD0120	48 F6 4E 78 A2 11 FC 0A 40 2E 42 65 FA 63 BD 87	H0Nfè.ù.@.BeúC%.

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.

Press enter or click to view image in full size

```

EIP 0066C82A 8B 63 8E F4 7B xor eax,edx
0066C82F 39 55 FC mov eax,7BF48E63
0066C832 7E 22 cmp dword ptr ss:[ebp-4],edx
0066C834 6A 07 jle 66C856
0066C836 5F push 7
0066C837 8B C8 pop edi
0066C839 C1 E9 1B mov ecx,eax
0066C83C 33 C8 shr ecx,1B
0066C83E C1 E9 03 xor ecx,eax
0066C841 33 C8 shr ecx,3
0066C843 03 C0 xor ecx,eax
0066C845 83 E1 01 add eax,eax
0066C848 0B C1 and ecx,1
0066C84A 4F or eax,ecx
0066C84B ^ 75 EA dec edi
0066C84D 30 04 1A jne 66C837
0066C850 42 xor byte ptr ds:[edx+ebx],al
0066C851 3B 55 FC inc edx
0066C854 ^ 7C DE cmp edx,dword ptr ss:[ebp-4]
0066C856 8B 55 FC j1 66C834
mov edx,dword ptr ss:[ebp-4]
    
```

Decryption loop

And the xor is made after key manipulation.

Press enter or click to view image in full size

```

0066C82A 8B 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;).Ax1 C.Z*2
01FD0010 F5 AB F3 D2 2D D0 28 55 B3 83 ED BE 36 00 2A 05 ô«óó-Ð(U*. í%6.*.
    
```

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.....yy..
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 84 09 CD 21 B8 01 4C CD 21 54 68 ..°.!.Li!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 98 91 21 62 C8 91 21 62 C8 91 21 62 C8 Ô@...!bÈ.!bÈ.!bÈ
01FD0090 FE 57 FC C8 83 21 62 C8 FE 57 C8 C8 D4 21 62 C8 pwüÈ.!bÈpwÈÈ!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 pwÈÈ.!bÈ.YñÈ.!bÈ
01FD00C0 91 21 63 C8 CB 21 62 C8 FE 57 CD C8 93 21 62 C8 .!cÈÈ!bÈpwIÈ.!bÈ
01FD00D0 FE 57 F8 C8 90 21 62 C8 FE 57 FF C8 90 21 62 C8 pwøÈ.!bÈpwÿÈ.!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.AZ
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

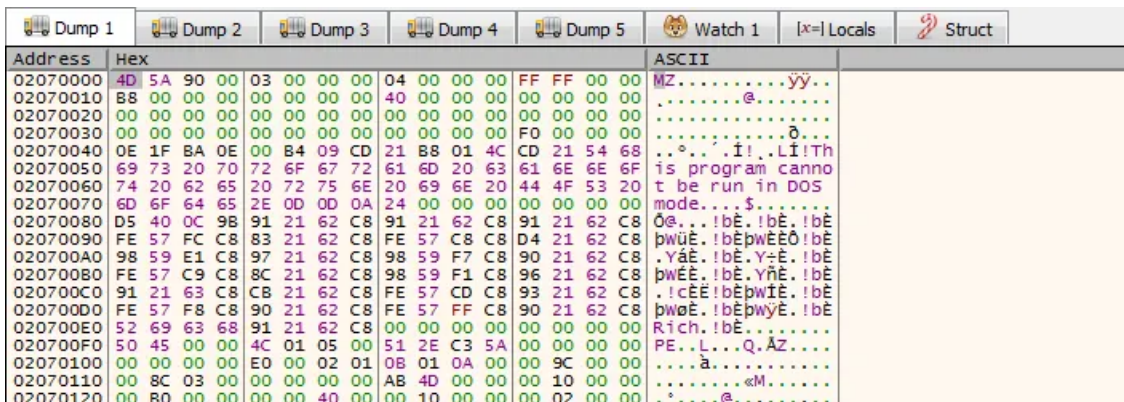
Then, the shellcode uses the VirtualAlloc and create a memory page at 02070000.

Press enter or click to view image in full size



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

Press enter or click to view image in full size

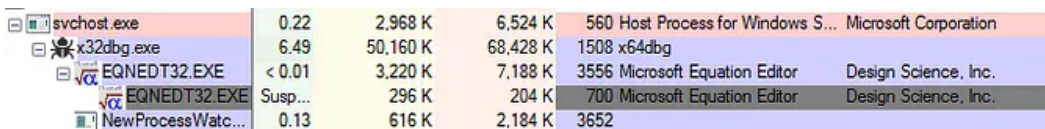


the PE decrypted is copied in the new memory page

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

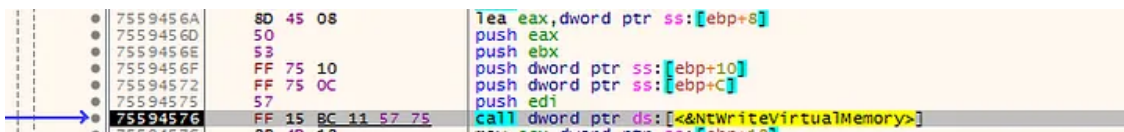
And EQNEDT32.exe is forked in suspend status by a CreateProcess and the shellcode overwrite it by the PE at the address 02070000

Press enter or click to view image in full size

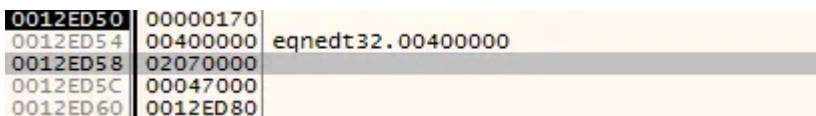


Fork of EQNEDT32.exe

Press enter or click to view image in full size



Overwriting of EQNEDT32.exe



Stack used by NTWriteVirtualMemory

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

Get Sebdraven's stories in your inbox

Join Medium for free to get updates from this writer.

Remember me for faster sign in

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.

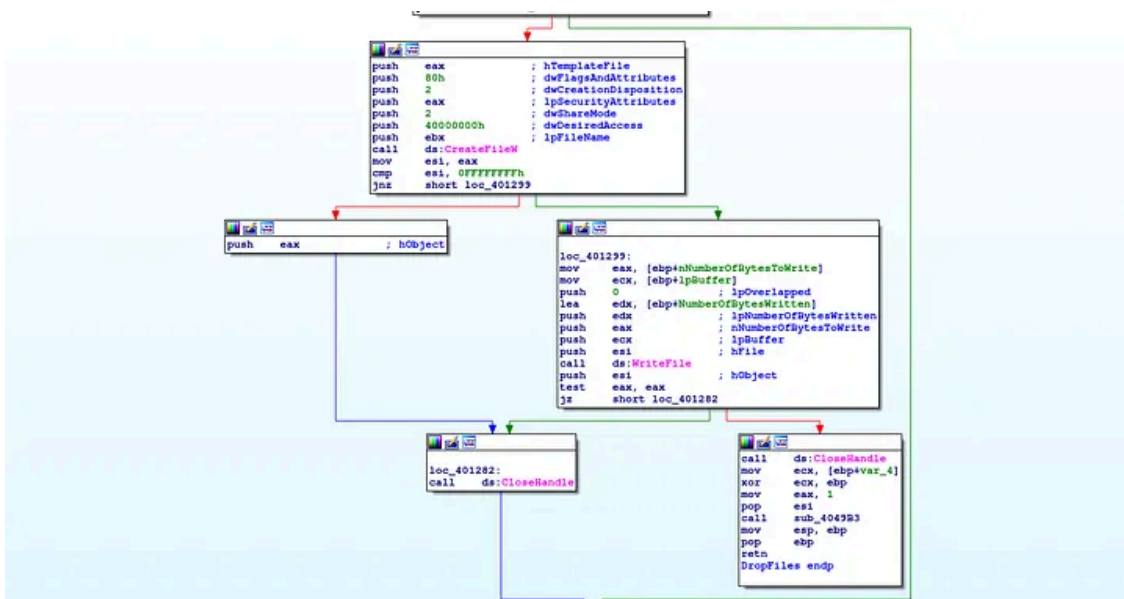
Analysing the fork of EQNEDT32.exe

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.

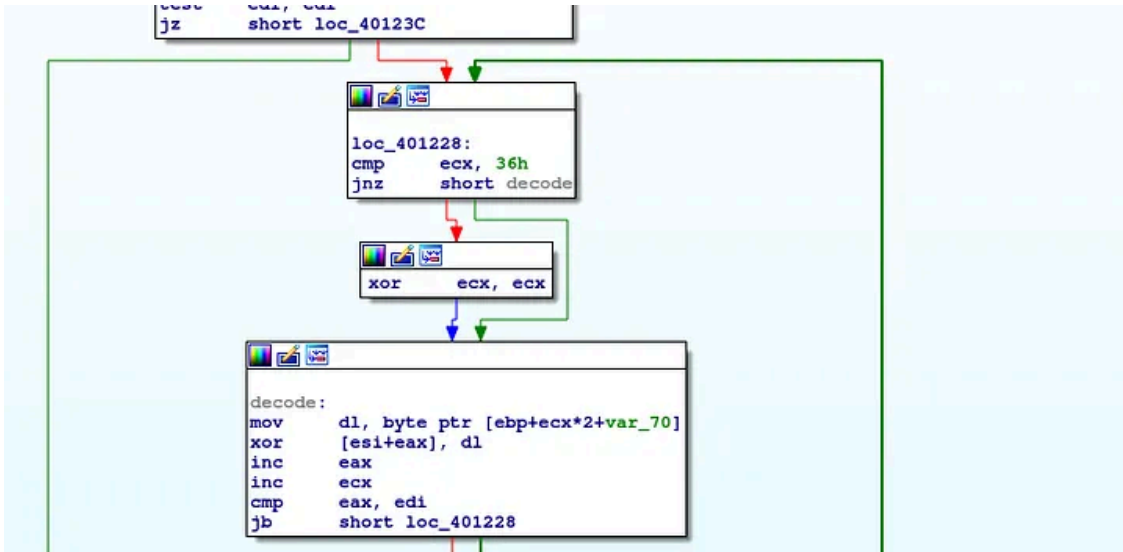
Press enter or click to view image in full size



DropFiles Fuction

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

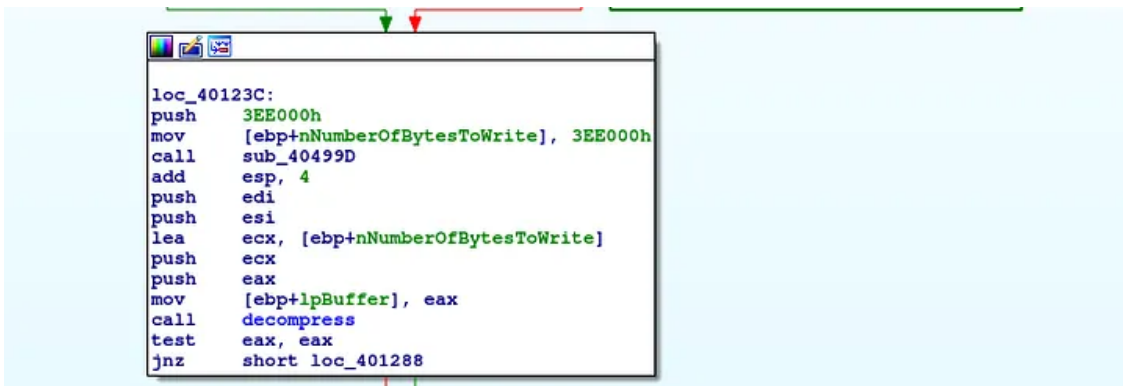
Press enter or click to view image in full size



Second loop of decryption

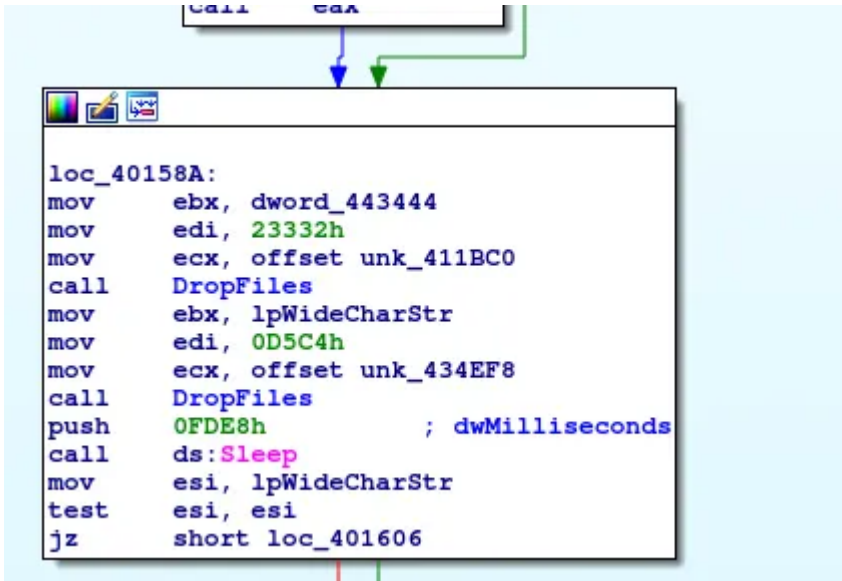
And just after we have a call of the decompression function.

Press enter or click to view image in full size



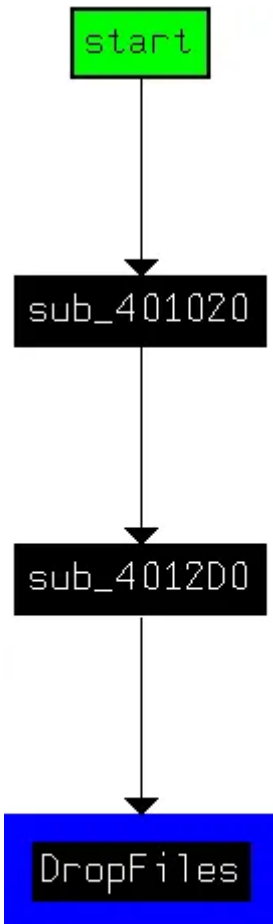
Decompression function used zlib

The function dropFiles is called twice by the sub_4012D0.



Drop the dll and the executable

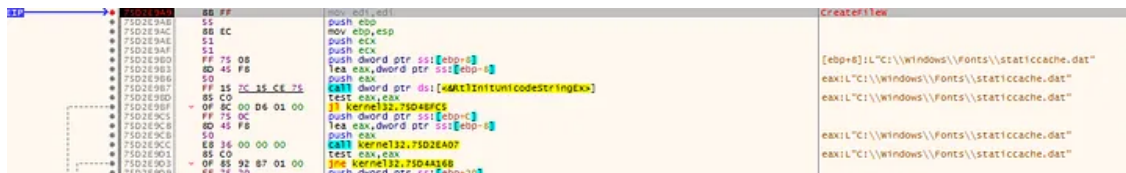
If we check the call graph, DropFiles is called only by the function sub_4012D0.



Functions using DropFiles function

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

Press enter or click to view image in full size

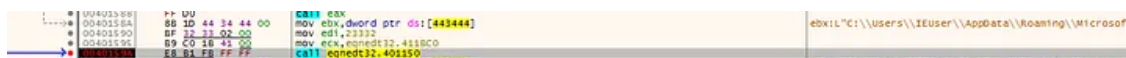


Breakpoint to createfile

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.

Press enter or click to view image in full size



Breakpoint to the first call of DropFiles

And now we can analyse the second loop of decryption.

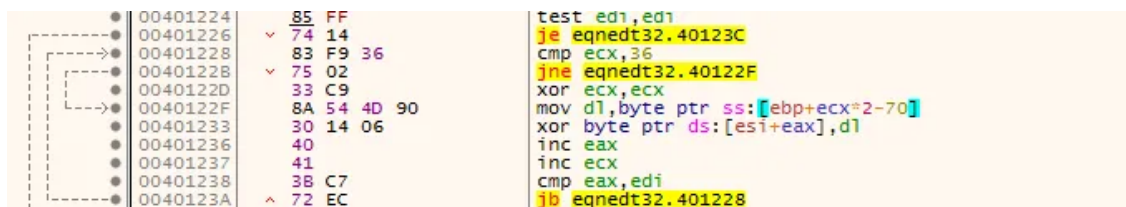
The first step is the initialization of the decryption function.

Press enter or click to view image in full size



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.

```
00411BC0 7C 9D E5 7A 08 55 57 D2 D2 FA 98 1F 0A 3D 3A AA |.âz.Uw000ú...=:ª
00411BD0 85 60 A1 8F 4F 61 E7 81 43 51 1B 59 4D 07 B7 47 |.i.Oaç.CQ.YM.·G
```

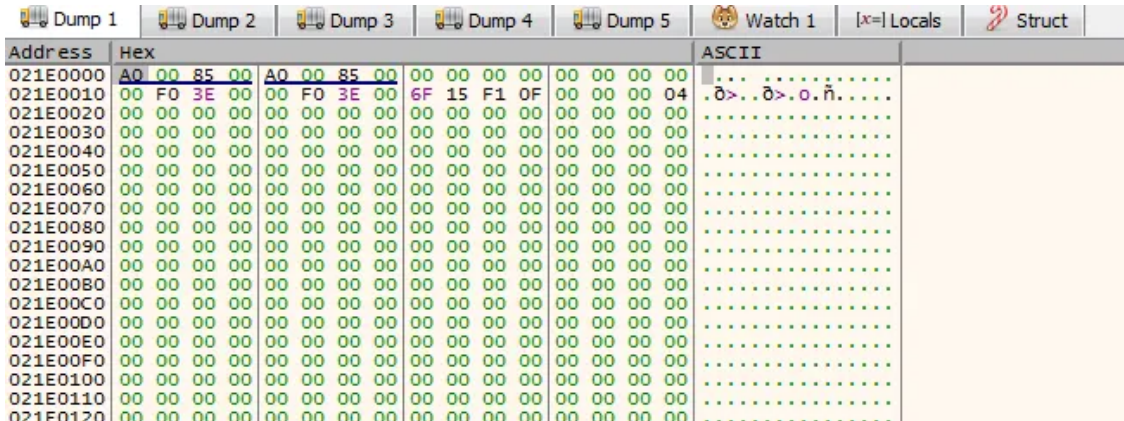
Before the decryption

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.iz.Uw000ú...=:ª
00411BD0 85 60 A1 8F 4F 61 E7 81 43 51 1B 59 4D 07 B7 47 |.i.Oaç.CQ.YM.·G
```

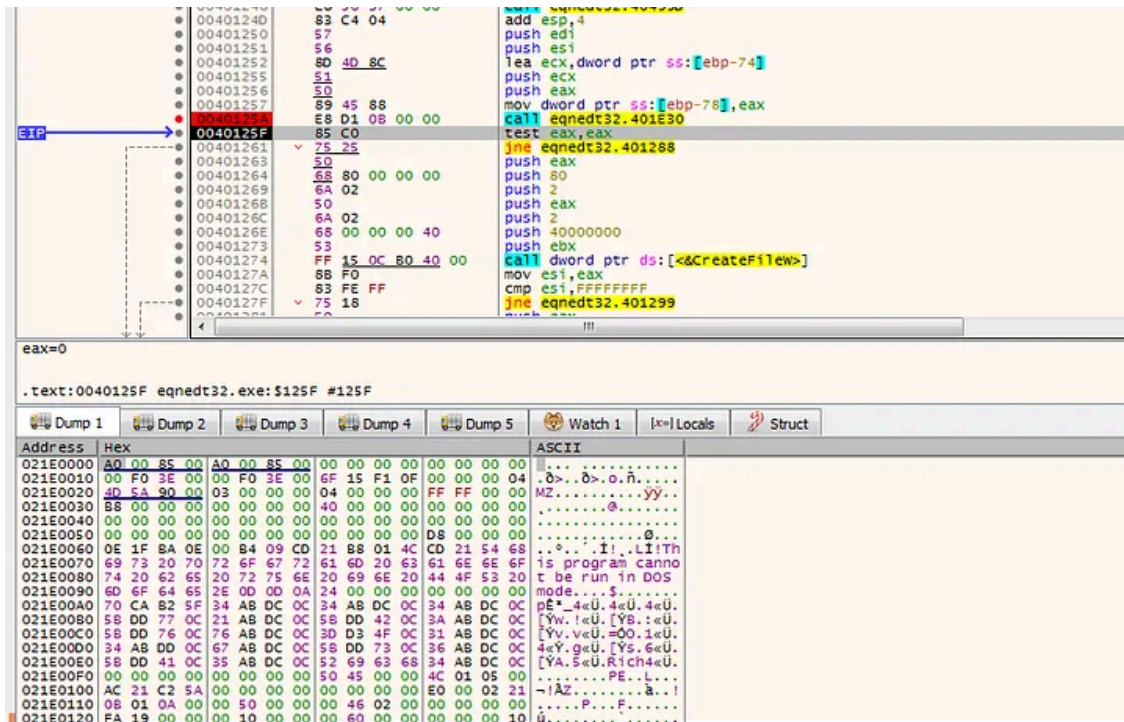
After the decryption

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



Page memory allocated to store the dll

Press enter or click to view image in full size



After decompression

And after the file is created with the following path:

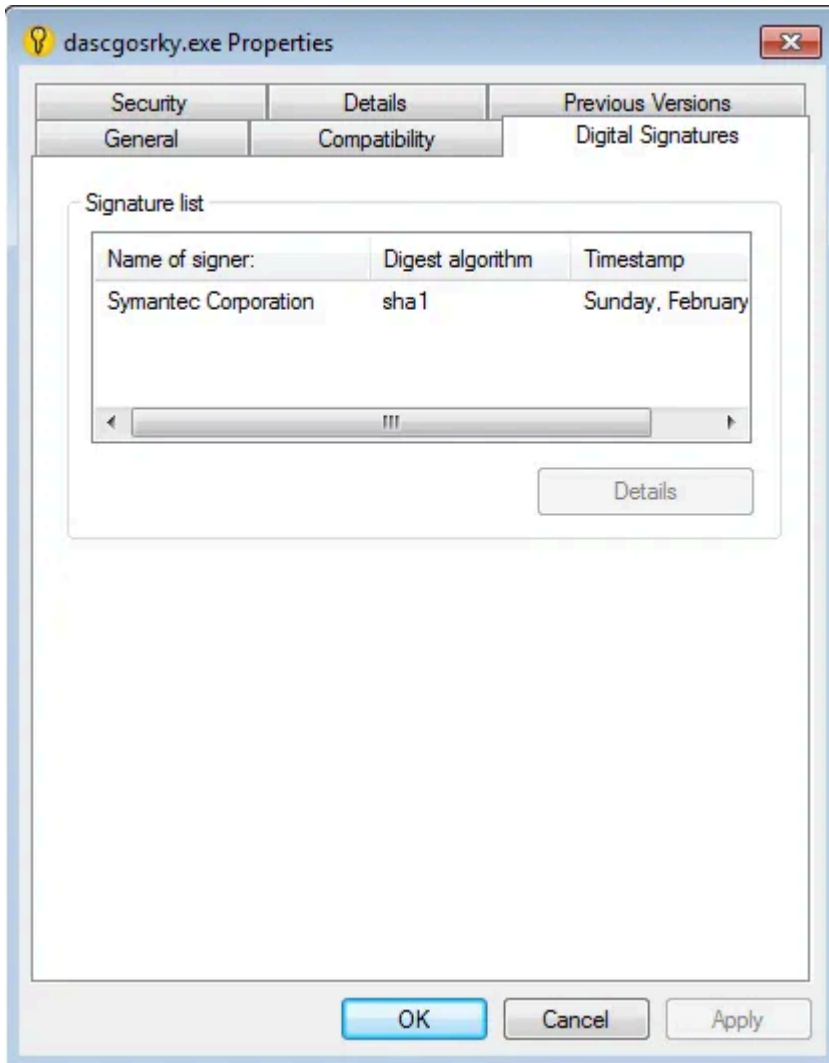
L”C:\Users\IEUser\AppData\Roaming\Microsoft\Windows\Network Shortcuts\RasTls.dll”



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

```
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



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.

Press enter or click to view image in full size

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 campaigns.

The second one is:

Cat.toonganuh.com is a subdomain of toonganuh.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 property 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)

9f5da7524817736cd85d87dae93fdbbe478385baac1c0aa3102b6ad50d7e5e368 (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 ScnCf.exe

Source: <https://medium.com/@Sebdraaven/malicious-document-targets-vietnamese-officials-acb3b9d8b80a>