Sogeti ESEC Lab

TL;DR

This article explains why it is still worth trying to reverse engineer a ransomware in order to retrieve your encrypted files. You may find a tool to decrypt the files modified by this specific ransomware at the end of the article.

Context

A few weeks ago, Sogeti ESEC was called on an incident that took place in a client information system.

The client noticed some strange activities ongoing on one computer within their network. Indeed, some files on that computer, including the ones available on network shares, were unreadable. Unfortunately most of those files included vital documents for our client, such as financial information, strategic information and so on. This data were the achievement of a couple dozen years of work, thus could not be allowed to be lost. That is the reason why several backups of those data were done.

After some investigations, they quickly understood that a malware was running on it. On their own, they suspected the presence of a ransomware.

Their moves were to:

- end the process that was running
- delete all the encrypted files
- reboot
- plug the backup storage and copy original files to the infected computer

Obviously the malware was still present on the system and had executed itself at boot time, infecting all freshly mounted backups, ending up with the loss of their three (3) backups (i.e.: averaging 1To worth of data).

Through a private company they were able to retrieve 40Go of encrypted data. Our mission was to understand how that incident happened and if we were able to retrieve their data.

This mission has been conducted in collaboration with the Pentest and R&D teams of Sogeti/ESEC.

Introduction

Wikipedia well defines what a ransomware is:

```
A ransomware is a type of malware which restricts access to a machine data in
some way. Most of the time, the malware encrypts the content of all disks
available locally and remotely. Then the encryption key is sent to the
attacker and destroyed.
The instructions to recover the files are also dropped by the malware. The
attacker commonly asks for a ransom to get the files decrypted.
```

Ransomware is the easiest and quickest way of preventing a company to work, creating a leverage to get paid, or "ransom". Of course there is no guarantee of being able to retrieve the lost data. On the top of that, cryptocurrencies - like Bitcoin - allow fast transactions and are more or less easy to cash out. That is one of the reasons this kind of malware is on the rise.

A 2015 study published by Microsoft [1] shows this increase:

```
2014 saw exponential infections from ransomware, with families such as Ransom:JS/Krypterade,
Win32/Crowti,
Win32/Reveton, and Win32/Teerac garnering more than 4 million infections. The start of 2015
introduced new
characters such as Win32/Tescrypt and Win32/Troldesh.
```

Due to their nature, most of the files encrypted with a ransomware are impossible to retrieve. Microsoft agrees [2]:

```
Due to the encryption of the files, it can be practically impossible to reverse-engineer the
encryption or “crack”
the files without the original encryption key – which only the attackers will have access
to.
```

Compromise

The discovery of the so-called **Patient Zero** is really important in order to understand how an incident happened. In our case, we easily identified a RDP service opened on the Internet. Through the computer's logs, we identified that a successful authentication of the *Administrator* user on that service was the root of that compromise. Indeed we then noticed that the password used was really weak and could be found in any dictionaries.

From there we were able to locate the malware (that was still running on the infected computer) and decided to analyze it.

Analysis

In this section, we describe the main steps of the ransomware which are common to this kind of malware.

Configuration decryption

The first task done by the ransomware is to decrypt a block of 73476 bytes located at the address *0x00413FF8*:

```
i = 0;
do
{
  if ( i * 4 & 1 )
    config[i] ^= 0xAF67D12E;
  else
    config[i] ^= 0xF76742E2;
  ++i;
}
while ( i < 18369 );
```

This block contains its configuration. Here is a list of the data contained in it:

- Size (on 4 bytes) of the RSA key in bytes
- The RSA modulus

```
00413FF8  80 00 00 00 B5 EF D6 45  33 3D 7B 75 57 96 FD 0A  Ç...Á´ÍE3={uWû².
00414008  2A E3 DB 56 78 38 57 CA  2C E3 64 33 AC 8A 75 57  *Ò¦Vx8W-,Òd3¼èuW
00414018  6A 71 64 A0 62 53 1B AC  C7 05 A7 23 60 CA 00 47  jqdábS.¼Ã.º#`-.G
00414028  C2 62 58 EC C2 34 AA D1  AE FD 86 F9 EA 78 56 6D  -bXý-4¬Ð«²å¨ÛxVm
00414038  C9 83 A4 7D A1 59 C2 29  F7 07 42 F1 C0 40 AE E6  +âñ}íY-)¸.B±+@«µ
00414048  CB F4 E3 EA 47 36 56 03  7F 99 31 D6 CE E8 AB 94  -¶ÒÛG6V..Ö1Í+Þ½ö
00414058  F5 B9 AC 64 F1 0D C8 17  ED 0A 90 24 DB CA 71 00  §¦¼d±.+.Ý..$¦-q.
00414068  30 E8 80 93 3B 93 24 E9  F6 8F 8C 5B BB D3 78 76  0ÞÇô;ô$Ú÷.î[+Ëxv
00414078  CE 5F FB C3 00 00 00 00  00 00 00 00 00 00 00 00  +_¹+............
```

The RSA exponent

```
004141F2  00 00 00 00 00 00 04 00  00 00 00 01 00 01 00 00  ................
```

List of drive letter to infect

```
00414472  00 00 00 00 00 00 41 00  42 00 43 00 44 00 45 00  ......A.B.C.D.E.
00414482  46 00 47 00 48 00 49 00  4A 00 4B 00 4C 00 4D 00  F.G.H.I.J.K.L.M.
00414492  4E 00 4F 00 50 00 51 00  52 00 53 00 54 00 55 00  N.O.P.Q.R.S.T.U.
004144A2  56 00 57 00 58 00 59 00  5A 00 00 00 00 00 00 00  V.W.X.Y.Z.......
```

The magic number for the encrypted file's footer

```
004144F8  30 30 35 50 52 5A 00 00  00 00 00 00 00 00 00 00  005PRZ..........
```

The email address to ask the ransom from

```
00414578  2E 00 7B 00 61 00 5F 00  70 00 72 00 69 00 6E 00  ..{.a._.p.r.i.n.
00414588  63 00 40 00 61 00 6F 00  6C 00 2E 00 63 00 6F 00  c.@.a.o.l...c.o.
00414598  6D 00 7D 00 00 00 00 00  00 00 00 00 00 00 00 00  m.}.............
```

The extension for the encrypted file

```
004145F8  2E 00 78 00 74 00 62 00  6C 00 00 00 00 00 00 00  ..x.t.b.l.......
```

The name of the mutex

```
00414678  47 00 6C 00 6F 00 62 00  61 00 6C 00 5C 00 73 00  G.l.o.b.a.l.\.s.
00414688  6E 00 63 00 5F 00 00 00  00 00 00 00 00 00 00 00  n.c._...........
```

The folder to exclude from encryption

```
00415678  25 00 77 00 69 00 6E 00  64 00 69 00 72 00 25 00  %.w.i.n.d.i.r.%.
```

The registry key for persistence

```
004147F8  53 00 6F 00 66 00 74 00  77 00 61 00 72 00 65 00  S.o.f.t.w.a.r.e.
00414808  5C 00 4D 00 69 00 63 00  72 00 6F 00 73 00 6F 00  \.M.i.c.r.o.s.o.
00414818  66 00 74 00 5C 00 57 00  69 00 6E 00 64 00 6F 00  f.t.\.W.i.n.d.o.
00414828  77 00 73 00 5C 00 43 00  75 00 72 00 72 00 65 00  w.s.\.C.u.r.r.e.
00414838  6E 00 74 00 56 00 65 00  72 00 73 00 69 00 6F 00  n.t.V.e.r.s.i.o.
00414848  6E 00 5C 00 52 00 75 00  6E 00 00 00 00 00 00 00  n.\.R.u.n.......
```

A list of files to exclude from encryption

```
004151F8  45 00 78 00 70 00 6C 00  6F 00 72 00 65 00 72 00  E.x.p.l.o.r.e.r.
00415208  2E 00 65 00 78 00 65 00  3B 00 53 00 76 00 63 00  ..e.x.e.;.S.v.c.
00415218  68 00 6F 00 73 00 74 00  2E 00 65 00 78 00 65 00  h.o.s.t...e.x.e.
```

The name of the files that contain decryption instructions to follow in order to pay the ransom (dropped on disk):

```
00414978  44 00 45 00 43 00 52 00  59 00 50 00 54 00 2E 00  D.E.C.R.Y.P.T...
00414988  6A 00 70 00 67 00 00 00  00 00 00 00 00 00 00 00  j.p.g..........

00414A78  44 00 45 00 43 00 50 00  59 00 50 00 54 00 20 00  D.E.C.P.Y.P.T. .
00414A88  46 00 49 00 4C 00 45 00  53 00 2E 00 74 00 78 00  F.I.L.E.S...t.x.
00414A98  74 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  t..............
```

The file `DECPYPT FILES.txt` obviously contain a typo (*P* instead of *R*). The only reason we found is the fact the *P* and *R* are on the same keyboard key on russian keyboard layout..

The instructions picture looks like that:

Thanks to a famous search engine and a russian OCR tool, this is the text in english:

```
Your data is encrypted using the latest encryption algorithm.

If you want to restore the data, send 1 encrypted file by e-mail to

a_princ@aol.com

You have 48 hours otherwise keys are removed
```

He wants one encrypted file, probably to be sure that we have been infected by his cryptolocker. Ransom demand probably comes as a response to the email.

Persistence

The ransomware tries to copy itself in `%windir%` then `%appdata%`.

Once done, it creates a registry key `System Service` under `Software\Microsoft\Windows\CurrentVersion\Run` in order to be run at startup, thus achieving persistence

```
if (v9 && GetModuleFileNameW(0, v0, 0x7FFFu))
{
    if ((wcscpy_s(v1, 0x7FFFu, "%windir%\\System32"), wcscat_s(v1, 0x7FFFu, L"\\"),
wcscat_s(v1, 0x7FFFu, CryptolockerName),
      ExpandEnvString(v1)) && CopyFileTo(v0, v1)   ||
      (wcscpy_s(v1, 0x7FFFu, "%localappdata%"), wcscat_s(v1, 0x7FFFu, L"\\"), wcscat_s(v1,
0x7FFFu, CryptolockerName),
      ExpandEnvString(v1)) && CopyFileTo(v0, v1))
    {
        if (RegOpenKeyExW(HKEY_LOCAL_MACHINE, &SubKey, 0, 0x20106u, &phkResult) ||
        (v5 = RegSetValueExW(phkResult, "System Service", 0, 1u, v1, 2 * wcslen(v1)) == 0,
        RegCloseKey(phkResult), !v5))
        {
            if (!RegOpenKeyExW(HKEY_CURRENT_USER, &SubKey, 0, 0x20106u, &phkResult))
            {
                RegSetValueExW(phkResult, "System Service", 0, 1u, v1, 2 * wcslen(v1));
                RegCloseKey(phkResult);
            }
        }
    }
}
```

Note: The name of this registry key is contained in the configuration data inside the binary and may change.

Operation

After this, the ransomware adds three files to the exception list:

- `DECRYPT.jpg`
- `DECPYPT FILES.txt`
- the name of the running binary

It then uses the `GetLogicalDrives` function to get the list of drives present on the machine.

```
while (1){
    v2 = GetLogicalDrives();
    v14 = v2;
    if ( v1 != v2 ){
        driveIndex = 0;
        v13 = v2 & ~v1 & 0xFFFFFFFC;
        v16 = 1;
        v15 = 0;
        do{
            if (v13 & v16){
                Src = *&aAbcdefghijklmn[driveIndex];
                v10 = malloc(0x1002Cu);
                *v10 = *lpThreadParameter;
                *(v10 + 16385) = *(lpThreadParameter + 16385);
                *(v10 + 16386) = *(lpThreadParameter + 16386);
                *(v10 + 16387) = *(lpThreadParameter + 16387);
                wcscpy_s(v10 + 2, 0x7FFFu, &Src);
                CreateThread(0, 0, sub_401A90, v10, 0, 0);
                driveIndex = v15;
                v2 = v14;
            }
            driveIndex += 2;
            v16 = __ROL4__(v16, 1);
            v15 = driveIndex;
        }
        while ( driveIndex < '@' );
        v1 = v2;
    }
    Sleep(0x64u);
}
```

On each of these logical drives, a thread is launched. Inside each of those threads, four other threads are launched

```
do
    *(&Handles + v1++) = CreateThread(0, 0, StartAddress, &Parameter, 0, 0);
while ( v1 < 4 );
```

Each of these previous threads will walk through folders recursively and will encrypt each file.

Encryption

In this section, we analyzed the different steps involving the encryption scheme used by the ransomware.

Key generation

The key is the most important operation of the ransomware as all the files will be encrypted with it.

The ransomware uses two (2) different sources of data, more or less random, to generate the 256 bits of the key.

First, it calls the `_ftime64` function to get a 64-bit time value by default. Only the epochs time in seconds and the current number of milliseconds will be used by the ransomware. This data represents 8 bytes of data stored at *0x00426c44*.

```
_ftime64(&data);
dword_426C44 ^= 1000 * ms;
dword_426C48 ^= ((1000 * ms) >> 32) | data;
```

The second source of data is the result of a call to the `rand` function of the libc. This call will add 4 bytes of data.

```
qword_426C4C = rand() ^ qword_426C4C; // qword_426C4C = 0
```

A 256 bits memory space is allocated for the key and then filled with zeroes. The number of milliseconds multiplied by 1000 is copied. Then the epochs time in seconds. And finally, the rand value.

```
00426C44  08 3A 04 00 FE 56 4F 57  29 00 00 00 00 00 00 00  .:..¦VOW).......
00426C54  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

As shown on the previous picture, more than 50% of the space is not updated with random or selected values and stay filled with zeroes.

From there, the ransomware computes the MD5 hash of these 256 bits to get 16 bytes of data.

```
md5CTX[0] = 0x67452301;
md5CTX[1] = 0xEFCDAB89;
md5CTX[2] = 0x98BADCFE;
md5CTX[3] = 0x10325476;
md5_update(32, md5CTX, &dword_426C44);
md5_final(md5CTX);
```

These 16 bytes are then used as a key to encrypt the 256 bits using the RC4 algorithm.

```
rc4_init(&rc4CTX, &data);
...
memcpy(a1, &data, lenToGenerated);
rc4_encrypt(&rc4CTX, a1, 32);
```

The resulting encrypted data is the key used to encrypt the files of the system.

To sum up:

```
KEY  = RC4(MD5(data), data)
data = [epochs][milliseconds*1000][rand()][0000][0000][0000][0000][0000]

(each [] = 4 bytes)
```

We can consider that the key is computed from 96 bits of data or only 74 bits if we consider the maximal millisecond value being 999.

Weaknesses

You may have already found the weaknesses in this encryption scheme.

- The epochs time in seconds can be guessed from the date of last modification of the encrypted files. Even if the timestamp has been corrupted, it is still possible to bruteforce the value over a few days.
- Regarding the number of milliseconds, there is no real weakness other than the number of milliseconds itself. Only 1000 possible values.
- The third weakness is the call of the `rand` function. As it is not securely seeded before, and this call is always the first call to this function, the value returned by `rand` is always the same (**0x00000029**).
- Finally, if we consider that we want to bruteforce the timestamp on 10 days, the key generation algorithm takes around 20 bits for the timestamp and 10 bits for the number of milliseconds.

A bruteforce should then require only a few minutes to find the key.

Key encryption

After the key generation, the key is encrypted using RSA and a 1024 bits key located in the config at **0x00413ffc**. Exponent is also in the config at **0x004141fc**.

```
00413FFC  B5 EF D6 45 33 3D 7B 75  57 96 FD 0A 2A E3 DB 56  Á´ÍE3={uWû².*Ò¦V
0041400C  78 38 57 CA 2C E3 64 33  AC 8A 75 57 6A 71 64 A0  x8W-,Òd3¼èuWjqdá
0041401C  62 53 1B AC C7 05 A7 23  60 CA 00 47 C2 62 58 EC  bS.¼Ã.º#`-.G-bXý
0041402C  C2 34 AA D1 AE FD 86 F9  EA 78 56 6D C9 83 A4 7D  -4¬Ð«²å¨ÛxVm+âñ}
0041403C  A1 59 C2 29 F7 07 42 F1  C0 40 AE E6 CB F4 E3 EA  íY-)¸.B±+@«µ-¶ÒÛ
0041404C  47 36 56 03 7F 99 31 D6  CE E8 AB 94 F5 B9 AC 64  G6V..Ö1Í+Þ½ö§¦¼d
0041405C  F1 0D C8 17 ED 0A 90 24  DB CA 71 00 30 E8 80 93  ±.+.Ý..$¦-q.0ÞÇô
0041406C  3B 93 24 E9 F6 8F 8C 5B  BB D3 78 76 CE 5F FB C3  ;ô$Ú÷.î[+Ëxv+_¹+
...
004141FC  00 01 00 01 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

As the size of the data to encrypt is only 32 bytes (256 bits), padding is added to the key before encryption. Padding is generated from the `rand` function, probably following the OAEP padding scheme.

File encryption

The ransomware uses the AES encryption algorithm with a 256 bits key in CBC mode. For each file, a random IV is generated using the same function than the key.

This IV is added in the data at the end of each encrypted file with the magic, the padding length and the encrypted key.

```
WriteFile(f, &magic, 6u, &NumberOfBytesWritten, 0); // 005PRZ
WriteFile(f, &IV, 0x10u, &NumberOfBytesWritten, 0);
WriteFile(f, &paddingLen, 1u, &NumberOfBytesWritten, 0);
WriteFile(f, (RSA + 32), 0x80u, &NumberOfBytesWritten, 0);
```

An example result is depicted below:

```
00103810 30 30 35 50 52 5A D2 5A 22 94 F0 BD DA A1 A0 2B 005PRZ.Z"......+
00103820 3F 97 C4 0F 23 F8 10 57 5D BA C9 61 03 AF 34 0C ?...#..W]..a..4.
00103830 76 4D 73 6B 3A 6D 15 77 DD E6 83 EE FD 3C 9E D1 vMsk:m.w.....<..
00103840 15 BC B9 91 38 BC 16 9E 02 3C 83 F2 42 2F 3F 89 ....8....<..B/?.
00103850 89 33 E0 94 C6 C7 FB EE C2 28 7B 0B 65 69 D8 76 .3.......({.ei.v
00103860 FC 55 E4 E9 FA 74 74 26 6C 01 52 19 6B CC 4F 1D .U...tt&l.R.k.O.
00103870 0B 36 3E 55 63 29 1F 9E 87 F0 90 EB 12 81 5A 08 .6>Uc)........Z.
00103880 E5 15 27 10 28 41 6A 14 6A DF 48 E2 BE 2D FE 71 ..'.(Aj.j.H..-.q
00103890 55 63 22 83 CB C4 BA 4B AE 92 98 73 5C E0 B9 04 Uc"....K...s\...
001038A0 F2 3F 3F CF 26 56                               .??.&V.
```

After the file is encrypted, it is deleted.

```
if ( v8 )
{
  SetFileAttributesW(a4, dwFileAttributes);
  if ( !*(v12 + 172) )
    DeleteFileW(lpFileName);
}
result = v8;
```

Decryption

In this section, we describe the different steps to recover the encryption key and decrypt the files.

Bruteforce the secret key

Bruteforcing the key requires a way to identify when we find the correct key. For that, we use the first 4 bytes (magic number) of several known filetypes because their header never changes. So, we need at least one encrypted file with the corresponding filetype.

In the following tables, some of the magic numbers values:

Extension	First 4 bytes
.exe,.dll	4D5A9000
.docx	504b0304
.xlsx	504b0304
.pptx	504b0304
.zip	504b0304
.jpeg	ffd8ffe0
.png	89504E47
.pdf	25504446
.rtf	7b5c7274

The algorithm used to bruteforce the key is :

```
for timestamp in range(start, end):
    for millisecond in range(1000):
        Key = GenerateKey(timestamp, millisecond*1000)
        if Decrypt(encryptedFile[:4]) == header:
            return KEY_FOUND
return KEY_NOT_FOUND
```

Key identification

As the ransomware is designed to be run at startup, it is possible that files on the same machine were encrypted with different keys. If we want to decrypt all files on a machine, we need a way to identify whether we have the right key or not.

For that, we use the encrypted key block located at the end of each encrypted file. For each run, the ransomware will encrypt the key using the RSA 1024 bits key and the result is added at the end of the encrypted file.

When we finally find the right key for a file, we can store the encrypted key separately. And when we need to decrypt a file, we only need to compare the encrypted key stored against the one stored previously.

Key decryption

The ransomware uses a 1024 bits modulus with 0x10001 exponent to encrypt the key. Thus, there is no way of decrypting the key except if you made some progresses in quantum computing.

File decryption

If we have the encryption key, we are able to decrypt the file using the IV stored at the end of each encrypted file with AES256 in CBC mode using the freshly retrieved key.

We also need to remove the last *x* bytes of the decrypted data by using the padding value stored in each encrypted file footer.

Indicator of compromise

Here is a list of IOCs for this ransomware:

- Files with `xtbl` or `{a_princ@aol.com}.xtbl` extension
- Process with mutex starting with `snc_`

- PE files present in `%appdata%` or `%windir%` with that fingerprint:

Algorithm	Fingerprint
MD5	ebcdda10fdfaa38e417d25977546df4f
SHA1	5b58de17843ac44adc91b41883828cf5b3a11744
SHA256	3c4588fe87146b9c1d0c97a5175bc287d8350ace3f4188d3cd2458638fcd8d97
Machoc	https://raw.githubusercontent.com/sogeti-esec-lab/ransomware-xtbl-decrypt-tool/master/machoc-signature.txt

Decryption

We developed a tool that decrypts files encrypted by this malware only. First, the tool will recover the encryption key using one encrypted file.

Please use an encrypted file with the last modification timestamp untouched.

Then the files on each disk of the machine will be decrypted.

Thos tool is available on github:

- Sources
- Compiled executable

Conclusion

Statistics show that the threat of being infected by a ransomware has only begun. Each month, more and more ransomware variants are detected.

Some of them do not use state of the art cryptography yet, or badly use it to encrypt files, such as in our case. But in most cases, there is no way to decrypt the file without having the secret key of the attacker.

Here, the fail comes from the `rand` function call which is not correctly seeded beforehand, the use of the timestamp which can easily be bruteforced and the number of milliseconds which holds a limited space of possibilities.

This post also highlights the good cooperation between the Pentest and the R&D team of Sogeti ESEC. For that, special thanks to **lerobert**, **jbedrine**, **meik**, who also worked on this incident response.

[1] https://www.microsoft.com/security/portal/enterprise/threatreports_september_2015.aspx

[2] https://www.microsoft.com/en-us/security/portal/mmpc/shared/ransomware.aspx