Following the RTM

Archived: 2026-05-01 02:25:18 UTC

Where did it all start?

Researchers became aware of the activities of the RTM group in December 2015. Since then, phishing emails distributing the trojan have been sent to potential victims with admirable persistence.

From September to December 2018 **the RTM group sent out more than 11,000 malicious emails**. The cybercriminals, however, are not going to stop there, as evidenced by the new malicious campaigns that we track as part of our ongoing threat intelligence activities.

In this article, I am going to show **how to perform forensic analysis of an image of a computer infected with the RTM banking trojan.**

Required background

Let’s imagine that we are not aware of the RTM infection on the computer. The only evidence we have is the fact of compromise, which resulted in a theft of funds. This will make the analysis process more interesting and applicable to other cases as well. I would also like to draw your attention to the fact that in this article I will not focus on reverse engineering of the trojan.

So, all we have is a computer drive image in E01 format (Encase Image File Format). For a start, it would be good to know what is inside, at least the operating system, since it is the operating system and its version that determine the presence of certain forensic artefacts that we have to analyse.

1. Let’s use the mmls utility from Brian Carrier’s Sleuth Kit:

```
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

      Slot      Start        End          Length       Description
000:  Meta      0000000000   0000000000   0000000001   Primary Table (#0)
001:  -------   0000000000   0000002047   0000002048   Unallocated
002:  000:000   0000002048   0000718847   0000716800   NTFS / exFAT (0x07)
003:  000:001   0000718848   0527046655   0526327808   NTFS / exFAT (0x07)
004:  000:002   0527046656   0976769023   0449722368   NTFS / exFAT (0x07)
005:  -------   0976769024   0976773167   0000004144   Unallocated
```

Let’s see what we have here. Several NTFS partitions that look like Windows partitions. We need to be sure of that. Let’s try to find registry files, such as SOFTWARE, for example.

2. Let's use the fls (from Sleuth Kit) and findstr utilities to find the corresponding entry number in the master file table (MFT):

```
C:\Users\0136\Desktop\sleuthkit-4.6.2-win32\bin>fls.exe -o 718848 -r E:\RTM.E01 | findstr "SOFTWARE"
++++ r/r 234782-128-3:  SOFTWARE
++++ r/r 65457-128-1:   SOFTWARE.LOG1
++++ r/r 65458-128-1:   SOFTWARE.LOG2
+++ r/r 37662-128-3:    SOFTWARE.LOG
```

Now we can copy the file we need for further analysis using icat (from Sleuth Kit):

icat -o 718848 E:\RTM.E01 234782 > SOFTWARE

We have a SOFTWARE registry file from which we can extract the most relevant information using **Harlan Carvey's RegRipper**, for example. At the moment, we are interested in the contents of the *Microsoft\Windows NT\CurrentVersion* section:

```
SystemRoot : C:\WINDOWS
PathName : C:\WINDOWS
EditionID : Professional
CSDVersion : Service Pack 1
CurrentType : Multiprocessor Free
ProductName : Windows 7 Professional
```

We now know that the computer in question ran Windows 7 Professional SP1, which means that we know what forensic artefacts we may encounter and which ones we may need.

Where do we begin our search? Let's recall Jesse Kornblum's paradox: "Malware can hide, but it must run". A good start will be to look for potential persistence mechanisms that can be used by the malware to restart after reboot.

Let's start with simple things: we will take the **NTUSER.DAT** registry file with the latest modification date from the user directory (C:\Users\%username%\), and extract data from it using RegRipper. If we want to get the record number of the file we need by means of fls and findstr again, we should add the -p parameter for fls — this will allow the utility to display the full paths to the files. Why do we need that? There is an NTUSER.DAT file in each user's directory, while there is only one SOFTWARE for the entire system, so in this case it is important to get the record number of a particular file. In general, you do not have to stick to the Sleuth Kit at all; there are more convenient tools like **FTK Imager**, a free tool, which can be used not only to create forensic images, but also to examine their contents.

Name	Size
$TXF_DATA	1
ntuser.dat	7 680
ntuser.dat.LOG1	256
ntuser.dat.LOG2	0
ntuser.dat{0c128f82-bbce-11e8-97c4-e03f49540b69}.TM.blf	64
ntuser.dat{0c128f82-bbce-11e8-97c4-e03f49540b69}.TMContaine...	512
ntuser.dat{0c128f82-bbce-11e8-97c4-e03f49540b69}.TMContaine...	512
NTUSER.DAT{6cced2f1-6e01-11de-8bed-001e0bcd1824}.TM.blf	64
NTUSER.DAT{6cced2f1-6e01-11de-8bed-001e0bcd1824}.TMCont...	512
NTUSER.DAT{6cced2f1-6e01-11de-8bed-001e0bcd1824}.TMCont...	512
ntuser.ini	1
ntuser.pol	1

Let’s start with low-hanging fruits, the so-called **run keys**:

```
Software\Microsoft\Windows\CurrentVersion\Run
LastWrite Time Wed Nov  7 10:59:41 2018 (UTC)
  apg: C:\Users\            \AppData\Roaming\b7mg81\apg.exe
  OfficeSyncProcess: "C:\Program Files\Microsoft Office\Office14\MSOSYNC.EXE"
  BssPluginHost32: C:\Users\                \AppData\Roaming\BSS\BSSPlugin\BssPluginHost.exe
```

The partition was last modified on November 7th, and we see that when a user logs in, the apg.exe file is executed from a very suspicious location. Let’s see what else we can find in the b7mg81 directory:

Name	Size	Type	Date Modif...
x64	1	Directory	03.10.2018 ...
x86	1	Directory	03.10.2018 ...
$I30	4	NTFS Index ...	07.11.2018 ...
apg.exe	10 372	Regular File	03.06.2015 ...
apg.exe.FileSlack	1	File Slack	
itdzsl.cfg	1	Regular File	07.11.2018 ...
msi.dll	33	Regular File	14.10.2018 ...
msi.dll.FileSlack	3	File Slack	
TeamViewer.ini	1	Regular File	25.12.2017 ...
TeamViewer_Des...	4 439	Regular File	03.06.2015 ...
TeamViewer_Des...	2	File Slack	
TeamViewer_Res...	268	Regular File	03.06.2015 ...
TeamViewer_Stati...	2 617	Regular File	03.06.2015 ...
TeamViewer_Stati...	4	File Slack	
tv_w32.dll	94	Regular File	03.06.2015 ...
tv_w32.dll.FileSlack	3	File Slack	
tv_w32.exe	192	Regular File	03.06.2015 ...
tv_w32.exe.FileSla...	1	File Slack	
tv_x64.dll	112	Regular File	03.06.2015 ...
tv_x64.dll.FileSlack	1	File Slack	
tv_x64.exe	228	Regular File	03.06.2015 ...
tv_x64.exe.FileSla...	1	File Slack	
x86		$I30 INDX E...	

TeamViewer? That is interesting. Let’s take a closer look at apg.exe and **use PPEE**:

Member	Value	Comment
Length	1910	
Revision	0200	
CertificateType	0002	PKCS_SIGNED_DATA
Certificate	30	

Type to filter...

Member	Value
Validity	SIGNED & VERIFIED
Program Name	TeamViewer
MoreInfo Link	http://www.teamviewer.com
Signer Certificate:	
Serial Number	56 72 93 00 C7 83 06 C4 26 7C A4 4A 10 AD CD 03
Issuer Name	VeriSign Class 3 Code Signing 2010 CA
Subject Name	TeamViewer
TimeStamp Certificate:	
Serial Number	0E CF F4 38 C8 FE BF 35 6E 04 D8 6A 98 1B 1A 50
Issuer Name	Symantec Time Stamping Services CA - G2
Subject Name	Symantec Time Stamping Services Signer - G4
TimeDateStamp	2015/06/03 19:11:04

This looks like TeamViewer and is signed as TeamViewer, so does this mean it indeed is TeamViewer? Seems so, but it’s not that simple. Let’s take a look at the import table:

Name RVA	Name	OriginalFirstThunk	TimeDate Stamp	ForwarderChain	FirstThunk	Description (Read from file)
00908F26	KERNEL32.dll	00908434	00000000	00000000	0072A000	Библиотека клиента Windows NT BASE API
00908F34	msi.dll	0090879C	00000000	00000000	0072A368	Windows Installer

Right, msi.dll — a file we have already seen somewhere. This is still the b7mg81 directory, not C:\Windows\System32. Judging by the file’s size, it has nothing to do with the original msi.dll, so it is clearly **DLL Search Order Hijacking**. The operating system starts searching for the necessary libraries from the current directory, which means that instead of the legitimate msi.dll, the one located in b7mg81 will be loaded.

Another interesting file is **TeamViewer.ini:**

```
[Settings]
nosave=1
importsettings=1
LogIncomingConnections=0
LogOutgoingConnections=0
writeconnectionlog=0
[License]
code=00-00000-000000-000000
```

Here is anti-forensics: according to the configuration file, our “TeamViewer” did not keep any logs, and was apparently used as a RAT (Remote Access Trojan). Well, not bad. It is time to find out if it ever started at all.

There are quite a lot of artefacts in Windows that can indicate that executable files have been run. Let’s keep working with the registry, this time with the **SYSTEM** file. To extract data from it, you can use RegRipper again.

We are interested in ControlSet001\Control\Session Manager\AppCompatCache. Here we will find a list of executable files with paths to them, the dates when they were last modified (according to the $STANDARD_INFORMATION attribute), and a flag indicating whether the file was launched or not:

Source: https://www.group-ib.com/blog/rtm