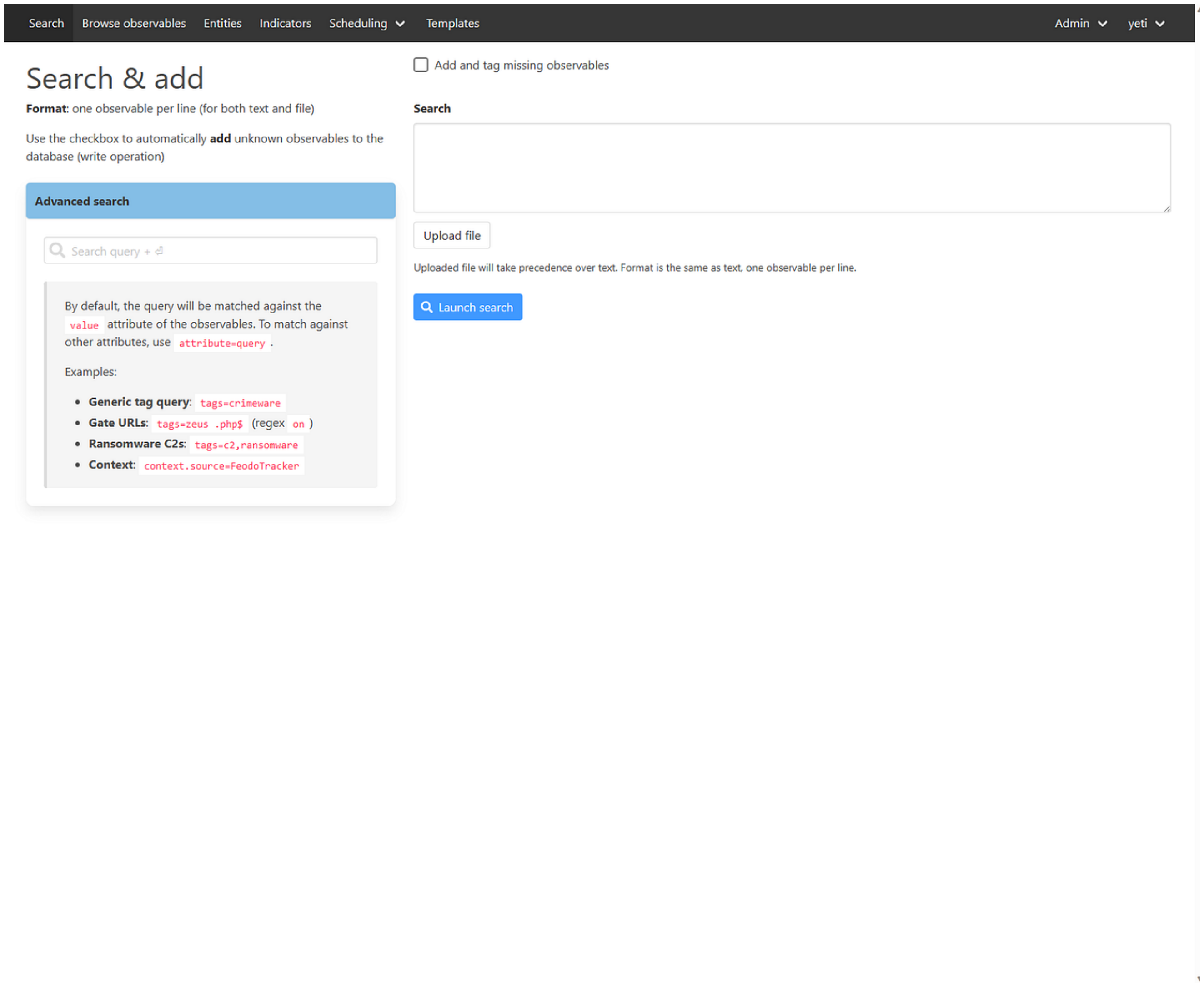
Malicious document targets Vietnamese officials

medium.com/@Sebdraven/malicious-document-targets-vietnamese-officials-acb3b9d8b80a

Sebdraven August 13, 2018



Search & add

Format: one observable per line (for both text and file)

Use the checkbox to automatically **add** unknown observables to the database (write operation)

Advanced search

Search query + ⏎

By default, the query will be matched against the `value` attribute of the observables. To match against other attributes, use `attribute=query` .

Examples:

- **Generic tag query**: `tags=crimeware`
- **Gate URLs**: `tags=zeus .php$` (regex `on`)
- **Ransomware C2s**: `tags=c2,ransomware`
- **Context**: `context.source=FeodoTracker`

☐ Add and tag missing observables

Search

Upload file

Uploaded file will take precedence over text. Format is the same as text. one observable per line.

Launch search

Sebdraven

--

After our investigation of APT SideWinder, we've done a yara rule for hunting RTF document exploiting the CVE-2017–11882.

We found a document written in Vietnamese dealing with a summary about differents projects in the district Hải Châu of Đà Nẵng.

RTF document
In this article, we'll detail the infection chains and the infrastructures of the attackers and the TTPs of this campaign.

The infrastructures and TTPs during this campaign seem to the Chinese hacking group 1937CN.

Infection chains

Joe sandbox has a good representation of the behaviour of the infection.

This rtf document is really malicious and it exploits the equation vulnerability to write two files in the system:

1. A dll named RasTls.dll
2. A executable file named dascgosrky.exe

This document is interesting to analyze so let'go !

RTF analysis

With rtfobj, we found three ole objects in the document:

two non well formed ole object and a third named package ole object.

The package ole object is used to write a file in the disk when the document is opened at the destination described by the ole object.

That's why, there is a path and a name in the ole object.

Package OLE Object
This technique is used to execute code like sct file to download an executable on the operating system. McAfee labs has detailed all this stuff with sct file: https://securingtomorrow.mcafee.com/mcafee-labs/dropping-files-temp-folder-raises-security-concerns/

Many attackers use it in the wild because it' very easy to use and it' supported by the office software with RTF files.

So, in our case, a file named 8.t is dropped on %TMP% folder.

If we check it, it's clearly encrypted.

8.t encrypted
The others object ole seem to the exploit of CVE-2017–11882.

Equation Ole Object
At the end of the object ole, we have differents API functions to make a runPE.

Another interesting thing is this string at the begin of the object: 7e079a2524fa63a55fbcfe

String found in many exploits of CVE-2017–11882
We have the same string used by APT SideWinder in the equation object ole.

It’s the same toolset to create the malicious document.

So now, we have to debug the malicious document to find how the file 8.t is used and find this runPE.

Debugging of the shellcode

At the start of the analysis, we think the process EQNEDT32.exe is created by Winword.exe using the function CreateProcess. So we decided to set a breakpoint at the call of his function.

But EQNEDT32.exe is invoked by Winword.exe using COM Object. It’s not CreateProcess that used and Winword.exe is not the parent process of EQNEDT32.exe. So we have to attach the debugger when EQNEDT32.exe is launched.

For that, we used a technique named Image File Execution Options that was documented by Microsoft. https://blogs.msdn.microsoft.com/mithuns/2010/03/24/image-file-execution-options-ifeo/

We create a key EQNEDT32.exe.

Registry HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options
And we set a value string for launching the debugger when EQNEDT32.exe is executed and attaching the debugger to the process .

Value to set the debuuger when EQNEDT32.exe is executed
When we open the rtf document, Winword is launched and EQNEDT32.exe also.

Winword process
EQNEDT32.exe process attached by the debugger
And the debugger is attached at the entrypoint of EQNEDT32.exe.

We check if it’s 8.t is correctly created in the %TMP% folder.

8.t dropped on disk

Now we set a breakpoint at the createFile to check if the shellcode of the exploit reads the file 8.t.

CreateFile is called at call eqnedt32.41E5EE.

The param of the path of file is pushed on the stack push dword ptr ss:[ebp-4].

The shellcode uses CreateFile to the 8.t in the %TMP% folder

So now, we can return of the user code at the calling function.

After a step into, we enter in the shellcode, the address space has changed:

Shellcode of the exploit
After CreateFile, GetFileSize is called to have the size of the file

Get the size of the file
After is Virtualloc, and it create a memory page at 1FD0000 (eax value)

VirtualAlloc memory page to load 8.t
After virtualAlloc, the memory page is pointed by EAX
The page allocated
ReadFile is called:

Readfile 8.t
And 8.t is loaded at 1FD0000:

8.t in memory
And the shellcode decrypts the 8.t file in memory at 0066C82A.

The loop of decryption is a xoring with different manipulations on the decryption key.

At the start of the decryption the key is set to 7BF48E63.

Decryption loop
And the xor is made after key manipulation.

Set the decryption key in EAX
If we check the destination of the result of the xoring (here edx + ebx), we find 01FD0000 where 8.t is loaded.

After two step of the loop, we can see the magic number MZ set at the begin of memory section.

MZ magic number
At the end of the decryption loop, we have a PE in memory at 01FD0000.

the file 8.t has been decrypted.

8.t fully decrypted
Then, the shellcode uses the VirtualAlloc and create a memory page at 02070000.

And the new PE at 01FD0000 is copied at this address.

the PE decrypted is copied in the new memory page
After GetModuleFileNameA is called to have the path of EQNEDT32.exe

And EQNEDT32.exe is forked in suspend status by a CreateProcess and the shellcode overwrite it by the PE at the address 02070000

Fork of EQNEDT32.exe
Overwritting of EQNEDT32.exe
Stack used by NTWriteVirtualMemory
And the shellcode does a ResumeThread to launch the new PE.

So, We've found all API Calls in the object ole at the beginning and we have a runPE to launch the new EQNEDT32.exe overwritten.

Analysing the fork of EQNEDT32.exe

We know that this process has to create on disk two files following the Joe SandBox Analysis:

- A dll named RasTls.dll
- A executable file named dascgosrky.exe

If we dump EQNEDT32.exe and we put in IDA, we found quickly the function that drops the files on disk (sub_00401150) renamed dropFiles.

DropFiles Fucntion
And at the start of this functions, we have a loop with a xor.

Second loop of decryption
And just after we have a call of the decompression function.

Decompression function used zlib
The function dropFiles is called twice by the sub_4012D0.

Drop the dll and the executable
If we check the call graph, DropFiles is called only by the function sub_4012D0.

Functions using DropFiles function

So we set a breakpoint on CreateFile because at each execution, EQNEDT32.exe starts by CreateFile onstaticcache.dat.

Breakpoint to createfile
And we return at the user code to set a new breakpoint to check the static analysis.

So we set a breakpoint at 0040159A when DropFiles is called.

Breakpoint to the first call of DropFiles
And now we can analyse the second loop of decryption.

The first step is the initialization of the decryption function.

Set for the second loop encryption
And after we find the xor and store the result in esi+eax.

Decryption loop
In the first step of the decryption loop, the result is written to 411BC0 in the address space of EQNEDT32.exe.

Before the decryption
After tree loops, we obtains the header of zlib compressed object.

After the decryption
And at the memory page 021E0000, a PE is decompressed.

Page memory allocated to store the dll
After decompression
And after the file is created with the following path:
L"C:\\Users\\IEUser\\AppData\\Roaming\\Microsoft\\Windows\\Network Shortcuts\\RasTls.dll"

Stored by ebx.

DropFiles is called a twice to decrypt and decompress the executable file.

The offset where store the file is 00434EF8 and the pe decompressed is stored at 025D0020

Decryption of the executable dascgosrky.exe
And the path of the new file is : ebx=005DA228
L"C:\\Users\\IEUser\\AppData\\Roaming\\Microsoft\\Windows\\Network Shortcuts\\dascgosrky.exe"

So we have two files in networks shortcuts of Windows.

Files drops on disk

dll hijacking

Dascgosrky.exe is a legit and trusted software develop by Symantec.

To load the library RasTls.dll, the executable calls LoadLibrary and GetProcaddress in sub_401940 to execute the malicious functions

Dascgosrky.exe loading the malicious
The original file
If we check the exports in IDA, we just have a dllentrypoint. The dll is executed like this.

We'll analyse the RAT in the second Part.

Infrastructure of Attackers

The domain contacted is wouderfulu.impresstravel.ga and this domain resolved on 192.99.181.14.

Domain wouderfulu.impresstravel.ga
This IP has differents domains found with PassiveTotal and theses domains is recorded in the IP 176.223.165.122.

Many domain names is used for Vietnameses people.

Expansion of domains
There are two domains really interesting:

Halong.dulichculao.com is already used in the campaign targeting Vietnameses organizations.

https://www.fortinet.com/blog/threat-research/rehashed-rat-used-in-apt-campaign-against-vietnamese-organizations.html

For Fortinet is the Chinese hacking group 1937CN.

If we compare the TTPs, it's really similar. They used RTFs to make the intrusion and dll hijacking to load the real payload.

And the name of domains are really similar between the campaings.

The second one is:

Cat.toonganuh.com is a subdomain of tooganuh.com recorded by florence1972@scryptmail.com

Conclusion

The Chinese hacking group 1937CN continues to target Vietnam officials with the same TTPs with a refreshing on the tools used. The toolset used by this group to create RTF malicious document has the same properpy of the SideWinder.

I want to thank my buddies on “Zone de Confort”. It’s with this dreamteam, I can finalize correctly this analyses.

In the second part, we analyze the RAT using in this campaign. Or if another reverse can make that, I’ll paid a beer ;)

IOCs for the paper:

domains:
dn.dulichbiendao.org
gateway.vietbaotinmoi.com

web.thoitietvietnam.org
hn.dulichbiendao.org
halong.dulichculao.com
cat.toonganuh.com
new.sggpnews.com
dulichculao.com
coco.sodexoa.com.
thoitiet.malware-sinkhole.net
wouderfulu.impresstravel.ga
toonganuh.com
coco.sodexoa.com

IPs:
192.99.181.14
176.223.165.122

RTFs:

42162c495e835cdf28670661a53d47d12255d9c791c1c5653673b25fb587ffed

8.t:

2c60d4312e4416745e56048ee35e694a79e1bc77e7e4d0b5811e64c84a72d2d7

PE:

f9ebf6aeb3f0fb0c29bd8f3d652476cd1fe8bd9a0c11cb15c43de33bbce0bf68 (exe)

9f5da7524817736cd85d87dae93fdbe478385baac1c0aa3102b6ad50d7e5e368 (dll)

Update:

The payload is PlugX. Thanks to Gabor Szappanos
https://twitter.com/GaborSzappanos/status/1024622354582908928

Update IOCs:

597c0c6f397eefb06155abdf5aa9a7476c977c44ef8bd9575b01359e96273486 59.rtf
11f38b6a69978dad95c9b1479db9a8729ca57329855998bd41befc364657d654 RasTls.dll
f9ebf6aeb3f0fb0c29bd8f3d652476cd1fe8bd9a0c11cb15c43de33bbce0bf68 RasTls.exe

b70069e1c8e829bfd7090ba3dfbf0e256fc7dfcefc6acafb3b53abcf2caa2253 b7.rtf
77361b1ca09d6857d68cea052a0bb857e03d776d3e1943897315a80a19f20fc2
spoolsver.exe
9fba998ab2c1b7fec39da9817b27768ba7892c0613c4be7c525989161981d2e2 vsodscpl.dll

9d239ddd4c925d14e00b5a95827e9191bfda7d59858f141f6f5dcc52329838f0 9d.rtf
087d8bee1db61273a7cd533d52b63265d3a8a8b897526d7849c48bcdba4b22ec RasTls.dll
f9ebf6aeb3f0fb0c29bd8f3d652476cd1fe8bd9a0c11cb15c43de33bbce0bf68 RasTls.exe

332aa26d719a20f3a26b2b00a9ca5d2e090b33f5070b057f4950d4f088201ab9 rtf

93aa353320a8e27923880401a4a0f3760374b4d17dcd709d351e612d589b969d vsodscpl.dll

77361b1ca09d6857d68cea052a0bb857e03d776d3e1943897315a80a19f20fc2 ScnCfg.exe