VB2021
localhost

7 - 8 October, 2021 / vblocalhost.com

WHERE IS THE CUCKOO EGG?

Ryuichi Tanabe, Hajime Takai & Rintaro Koike

NTT Security (Japan) KK, Japan

ryuichi.tanabe@global.ntt

hajime.takai@global.ntt

rintaro.koike@global.ntt

ABSTRACT

In 2020 we observed that TA428, which might belong to China, had used a new piece of unknown malware. We named it 'Tmanger'. We analysed it in detail, and we found that there had also been some other Tmanger-like malware. These pieces of malware, called 'Albaniiutas' and 'Smanager', have been reported to have been used in two supply chain attacks.

At the start of this presentation, we provide a detailed analysis of each Tmanger malware family. In particular, we focus on the relationships between the Tmanger family and the timeline of those pieces of malware. We also introduce how the supply chain attacks that used the Tmanger family occurred by sharing an intrusion case.

Next, we describe how to find Tmanger malware and how to research it robustly. In this section, you will learn how to detect the Tmanger family effectively.

Finally, we consider the relationship between TA428 and other APT groups by showing relationships between malware builders and infrastructures and by comparing the shared cases such as the Royal Road RTF weaponizer and ShadowPad. The Tmanger family was used by TA428 at first, but other APT groups such as Lucky Mouse also started using it later. This can be considered as the malware being shared between TA428 and the other APT groups.

Through this presentation we will share various information (details of the campaign, the toolsets, the TTPs, the infrastructure, and the group's information), allowing SOCs, CSIRTs and security researchers who research APT groups which might belong to China to gain a deeper understanding of their attacks and of how to take measures against them.

INTRODUCTION

Around 2019, a lot of researchers reported on the Royal Road RTF weaponizer, which is a shared tool among Chinese APT groups [1, 2, 3]. Last year, we presented Operation LagTime IT, which had been started by Royal Road [4]. In the research, we discovered an unknown piece of malware called Tmanger.

Tmanger is a typical RAT, and it has various versions [5]. Some Tmanger variants have been discovered such as Albaniiutas and Smanager [6, 7]. Albaniiutas has similar characteristics to Tmanger (target, timestamp, data in resource section, etc.), however there are differences in detail such as the handles of C&C communications, commands, and so on. On the other hand, because of its target and some differences, Smanager is a little bit different. However, the names of export functions, the feature of the encryption key, the common processes and other points match Tmanger and Albaniiutas. Since it is difficult to say that these are just coincidental matches, we consider Smanager to be a Tmanger variant.

Other than Operation LagTime IT, threat groups used Tmanger family malware in supply-chain attacks such as Operation StealthyTrident [8] and Operation SignSight [9]. Furthermore, Tmanger communicated with one of the ShadowPad C&C servers used by LuckyMouse in Operation StealthyTrident. This indicates the possibility of relationships amongst these APT groups.

We consider that Tmanger has become one of the most influential malware families in East and Southeast Asian countries. In this paper we provide detailed analysis of these Tmanger variants and the relationship between them. In addition, we will show the relationship of other APT groups with TA428.

MALWARE ANALYSIS

In this section, we will describe the analysis results of Tmanger, Albaniiutas and Smanager. We found that these three pieces of malware have some common elements. We named each element based on the name of an EXE, the internal name of a DLL and the PDB path. Developers may categorize them like us as follows:

Classification of Tmanger	Description
Setup	Open and execute MloadDll
MloadDll	Open and execute Client
Client	Fileless RAT

Table 1: Classification of Tmanger.

Tmanger

Tmanger is a RAT used by TA428. Based on the PDB file path, we named this RAT 'Tmanger'. We guess that 'Tmanger' is typo of 'Tmanager'. We do not know whether the developer of this malware intended it or not, but we found similar typos in other locations – for instance, strings such as 'Entery', 'Waston', etc.

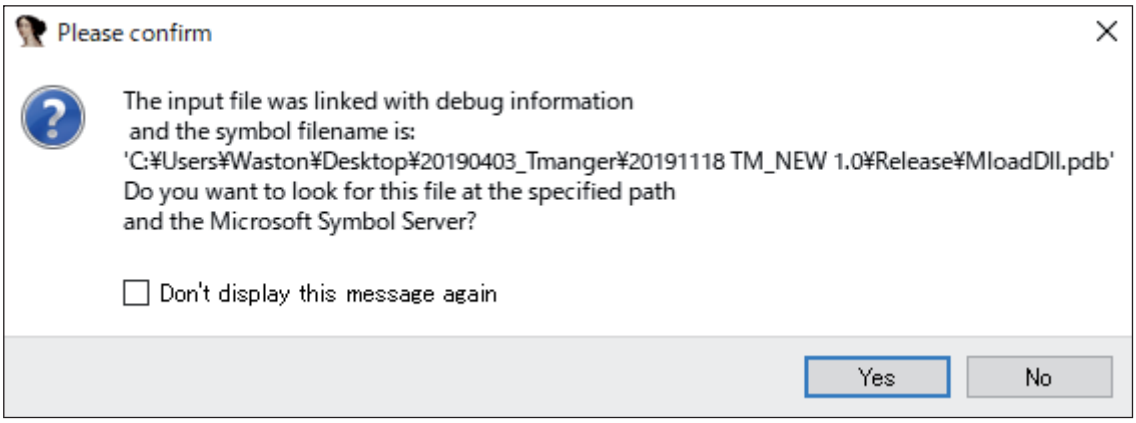


Figure 1: PDB path of Tmanger.

Attack flow

Figure 2 shows the Tmanger attack flow. We have observed Tmanger versions 1.0 – 6.2 and they all consist of Setup, MloadDll and Client. Setup establishes the persistence of MloadDll, then MloadDll executes Client. Client is a fileless RAT which executes commands received from C&C servers.

Figure 2: Attack flow of Tmanger.

Setup

Evade multiple launching

Setup is implemented as a process to stop itself if it fails to create a specific name of event. Thus, Setup creates the specific named event using CreateEvent() from WINAPI when it first runs. We guess that this process prevents multiple executions of Setup. We have confirmed that MloadDll and Client also contain this kind of process using CreateEvent() to evade multiple launching. We observed that the name of the created event fulfils the following regex condition:

```
/[0-9a-f]{8}-[0-9a-f]{4}-4551-8f84-08e738aec[0-9a-f]{3}/
```

Figure 3: Name of event created using CreateEvent().

Check admin privileges

Next, Setup checks the privilege of the user by using IsUserAdmin() from WINAPI. The persistence method of MloadDll to the infected host depends on the privilege level of the user.

Establish persistence of MloadDll (with admin privileges)

First, Setup decrypts the XOR'd strings with the value 0x88. The decrypted strings are as follows (they are used for the service registration later):

- DFS Replication
- FTP Publishing Service
- ReadyBoost
- Software Licensing
- SL UI Notification Service

- Terminal Services Configuration
- Windows Media Center Extender Service
- Windows Media Center Service Launcher
- SOFTWARE\Microsoft\Windows NT\CurrentVersion\Svchost
- netsvcs
- %SystemRoot%\System32\svchost.exe -k netsvcs
- MACHINE\SYSTEM\CurrentControlSet\Services\

Then, Setup decrypts other XOR'd strings also using the value '0x88' as in the previous step. The obtained strings are used subsequently for resolving WINAPIs:

- RegOpenKeyEx
- RegQueryValueEx
- OpenSCManager
- RegOpenKeyExA
- RegQueryValueExA
- RegSetValueExA
- GetSystemDirectoryA
- RegCloseKey

Next, Setup inflates the deflated data and stores a DLL file with a four random character name under System32 directory. This DLL is 'MloadDll'.

Again, Setup decrypts the following XOR'd strings using the value 0x88:

- SYSTEM\CurrentControlSet\Services\
- Description
- DisplayName
- ServiceDll
- \Parameters
- CreateServiceA

Setup registers MloadDll (which was created in the System32 directory before) as a service and runs it.

Establish persistence of MloadDll (without admin privileges)

First, Setup confirms if there is a file named 'Rahoto.exe' in the %TEMP% directory. If it does not exist, Setup copies itself to %TEMP% as 'Rahoto.exe' and creates an entry in the 'CurrentVersion\Run' registry key to run automatically. Afterwards, MloadDll runs as Client.

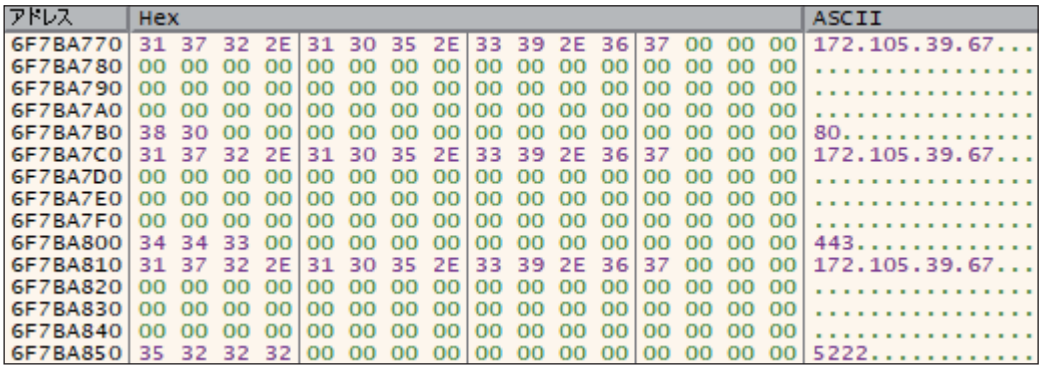
MloadDll

MloadDll implements export functions named 'Entery' and 'ServiceMain'. Since Entery will execute in the end, it does not matter whether the user has admin privileges or not.

First, MloadDll generates an RC4 key. It decrypts the config data using the generated key. This config data includes IP addresses and port numbers of the C&C servers.

```
0x004010f0      movzx eax, byte [edx - 0x40]
0x004010f4      lea edx, [edx + 4]
0x004010f7      xor byte [edx - 4], al
0x004010fa      movzx eax, byte [edx - 0x43]
0x004010fe      xor byte [edx - 3], al
0x00401101      movzx eax, byte [edx - 0x42]
0x00401105      xor byte [edx - 2], al
0x00401108      movzx eax, byte [edx - 0x41]
0x0040110c      xor byte [edx - 1], al
0x0040110f      sub esi, 1
0x00401112      jne 0x4010f0
```

Figure 4: Generating RC4 encryption key.



アドレス	Hex																ASCII
6F7BA770	31	37	32	2E	31	30	35	2E	33	39	2E	36	37	00	00	00	172.105.39.67...
6F7BA780	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
6F7BA790	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
6F7BA7A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
6F7BA7B0	38	30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	80..............
6F7BA7C0	31	37	32	2E	31	30	35	2E	33	39	2E	36	37	00	00	00	172.105.39.67...
6F7BA7D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
6F7BA7E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
6F7BA7F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
6F7BA800	34	34	33	00	00	00	00	00	00	00	00	00	00	00	00	00	443.............
6F7BA810	31	37	32	2E	31	30	35	2E	33	39	2E	36	37	00	00	00	172.105.39.67...
6F7BA820	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
6F7BA830	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
6F7BA840	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
6F7BA850	35	32	32	32	00	00	00	00	00	00	00	00	00	00	00	00	5222............

Figure 5: Decrypted config data.

After that, MloadDll inflates the deflated data. The obtained data is Client. MloadDll calls 'callfunc', which is one of the export functions of Client.

Client

Collect information about the infected host

After executing CreateEvent() and other functions, Client collects the following information about the infected host:

- OS and architecture information
- Drive information
- Host information
- User information

Create mutex

Next, Client creates a thread and repeats a set of loops. In the loop process, Client first creates a socket. If the socket is successfully created, Client creates mutexes using CreateMutex(), as follows:

- sock_hmutex
- cmd_hmutex

Command list

In the next step, Client checks if it can connect with C&C servers. After making a connection, Client waits to receive commands from the C&C servers.

Command ID	**Description**
1, 17	Start specific process
2	Get directory information
3, 19, 35	Send file to C&C server from client
4	Get file information
18	File delete
20, 52	Clean up memory etc.
34	Start process by CreateProcess()
36	File write
50	File copy
80, 81	Get keylog
96	Get screen capture
Others	Sleep

Table 2: Client command list.

The traffic is encrypted by RC4. The decrypted traffic structure is shown in Figure 6.

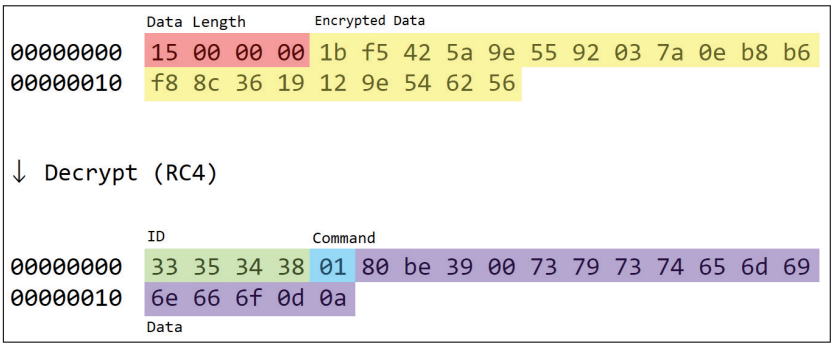


Figure 6: Traffic structure.

The ID that exists at the head of the traffic data after decrypting is generated from the process ID (PID). The processes and the traffic are managed by this ID. The ID is generated as follows:

$$\{(\text{ProcessID} \% 9) \times 1000\} + \{((\text{ProcessID} \% 1000) + 1000\}$$

Figure 7: Algorithm to derive ID.

Albaniiutas

In July 2020 we found a piece of malware that behaved a little differently from the Tmanger malware we had seen until then. The Tmanger malware that we had found until that point had similar implementations, but this one was obviously different. We call this malware 'Albaniiutas' from the file name we found.

Like Tmanger, Albaniiutas consists of Setup, MloadDll and Client. There are a lot of other similarities, such as its target, existing data in the resource section, and so on. From these points, we consider Albaniiutas to be a variant of Tmanger, or at least malware made by same developer as Tmanger.

Attack flow

Like Tmanger, Albaniiutas uses Service to establish persistence if the user has admin privileges. However, if the user does not have admin privileges, Albaniiutas creates an entry in the 'CurrentVersion\Run' registry key. Figures 8 and 9 indicate the attack flow of Albaniiutas.

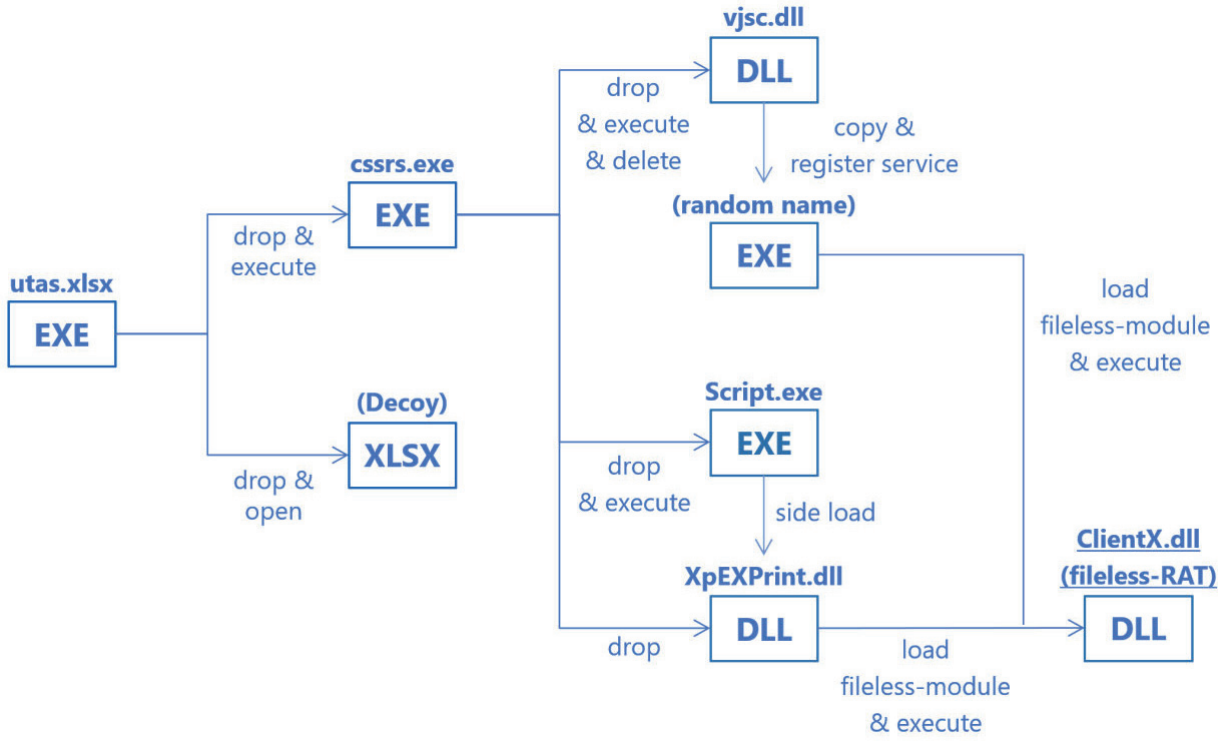


Figure 8: Attack flow of Albaniiutas (with admin privileges).

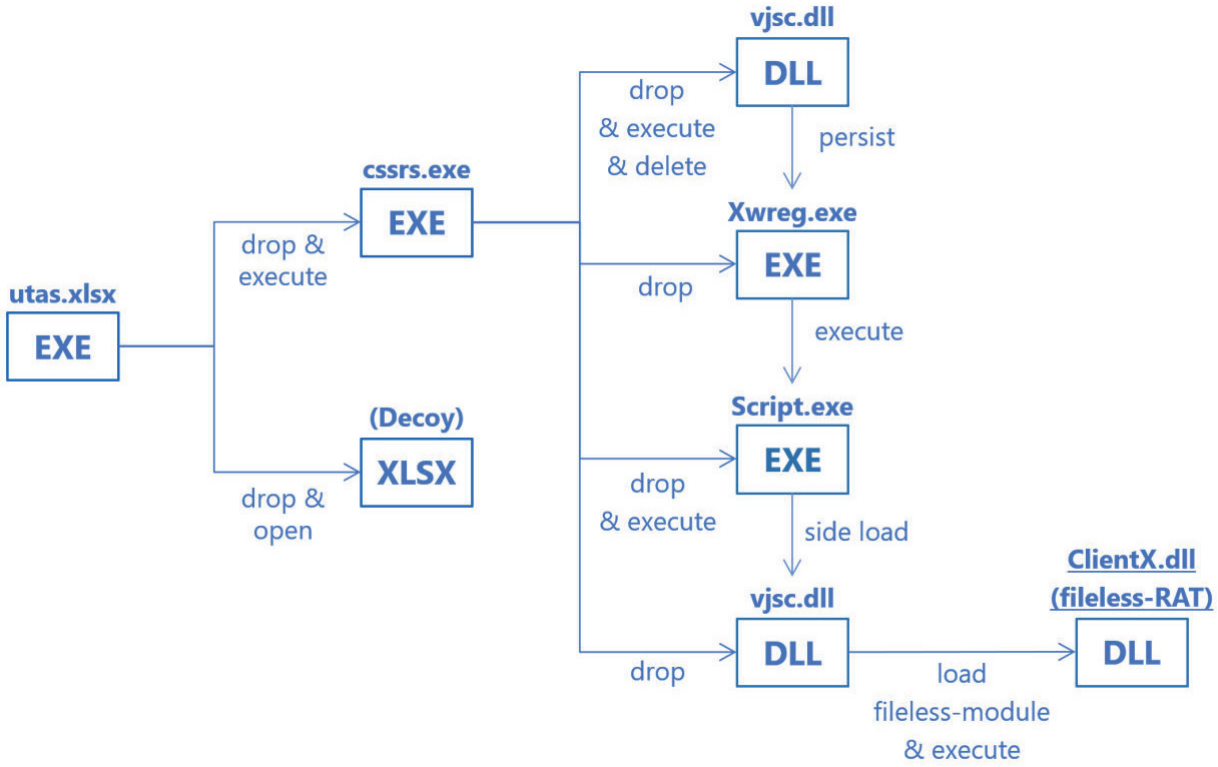


Figure 9: Attack flow of Albaniiutas (without admin privileges).

The attack begins with a RAR file named albaniiutas.rar. From this file, a file named utas.xlsx.exe is extracted. As you can see, this EXE file is disguised as an XLSX file. If this file is executed, a stored XLSX file and EXE file will be created and executed in the resource section.

The XLSX file is a contact list of the members of the Mongolian Citizens' Representative Hural. This indicates that this threat actor aims to attack Mongolian political organizations. This is the same target as TA428.

Classification of Tmanger	**Albaniiutas**
Setup	cssrs.exe, vjsc.dll
MloadDll	• Admin privileges (random name), Script.exe, XpEXPrint.dll • No Admin privileges Xwreg.exe, Script.exe, vjsc.dll
Client	ClientX.dll

Table 3: Corresponding Albaniiutas elements to Tmanger classification.

The name of the EXE file is cssrs.exe and it generates some other files. Since cssrs.exe contains roles similar to Setup, MloadDll and Client, it is extremely like Tmanger.

cssrs.exe

Evade multiple launching

Using CreateEventW(), cssrs.exe creates an event which it names{F14E0EF3-E26A-4551-8F84-08E738AEC912}. This event name corresponds to the rule of Tmanger event creation.

Check admin privileges

Like Tmanger, cssrs.exe checks the privilege of the user by using IsUserAdmin() from WINAPI. The persistence method of Albaniiutas depends on the privilege of that user.

Behaviour with admin privileges

First, cssrs.exe decrypts the data with the key value 'L!Q@W#E$R%T^Y&U*Al}t~k' by using RC4. The decrypted data is 'XpEXPrint.dll' and is the name of the DLL file in the next step.

Subsequently, cssrs.exe loads the data whose ID is 162 and whose resource type is 'T', from the resource section. After AES-256 decrypting the loaded data, it becomes deflated data. Afterwards, cssrs.exe inflates the deflated data, and stores it as XpEXPrint.dll in the System32 directory.

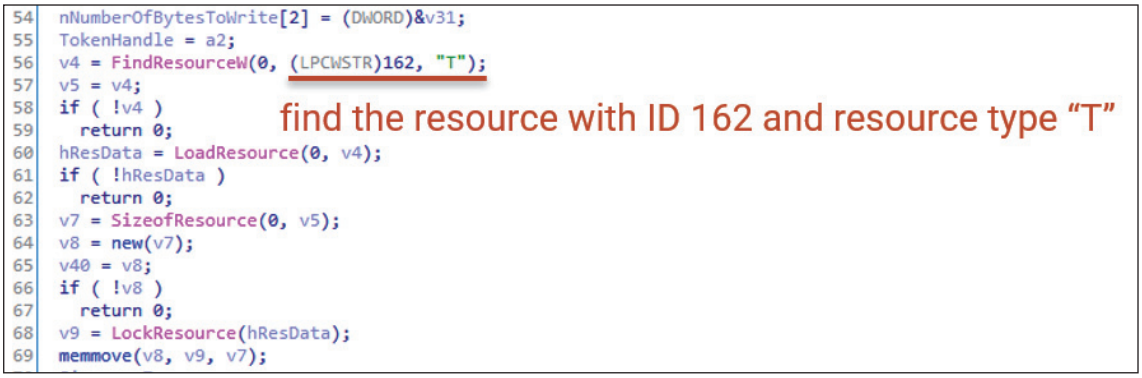


Figure 10: Loading data from the resource section.

'T' is a resource type defined by the developer and is not a *Windows* standard resource type. Albaniiutas loads resources several times later, and in each case their resource type is 'T'. The AES-256 decryption key is created by a hash generated using the CryptHashData() function. The second argument of CryptHashData() is set to a characteristic string, 'e4e5276c00001ff5', as shown in Figure 11.

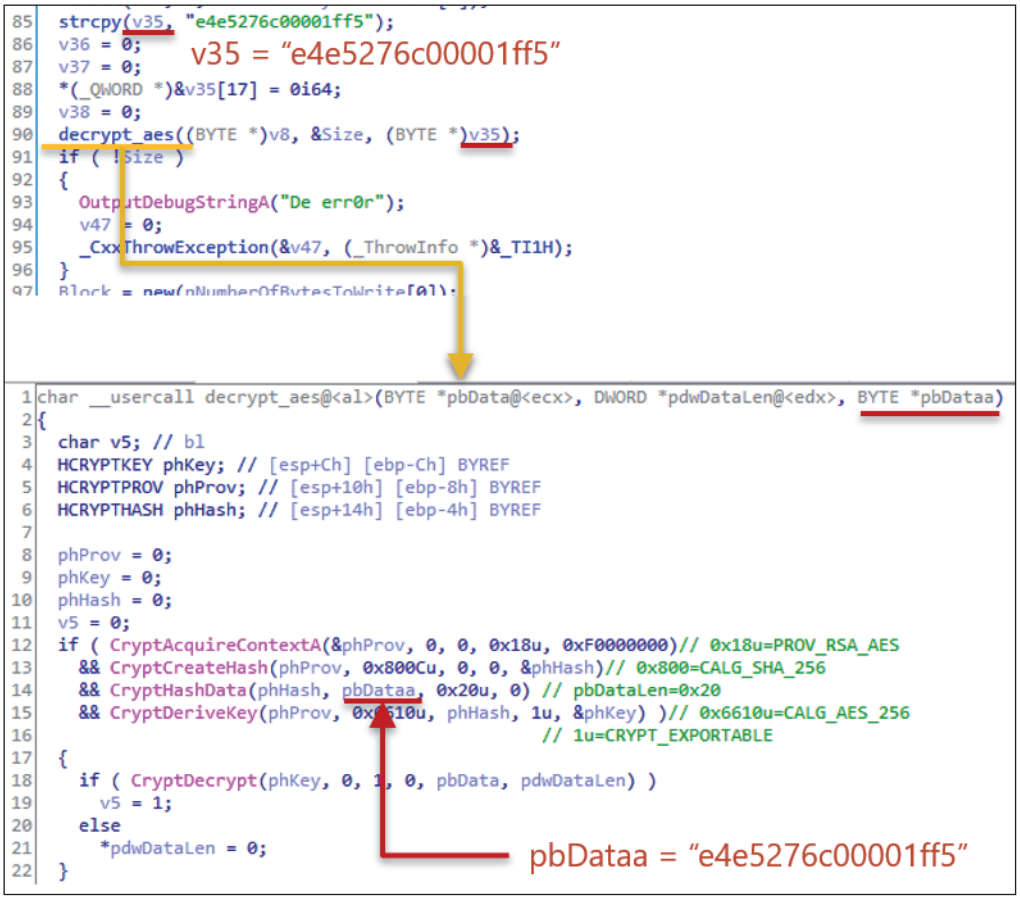


Figure 11: AES-256 decrypt data in resource section.

Next, cssrs.exe loads the data from the resource section as well. The IDs are 163 and 165. The obtained data is decrypted using the value 'L!Q@W#E$R%T^Y&U*Al}t~k' as an RC4 key. The data of ID 163 will be stored as 'vjsc.dll' and ID 165 as 'Scrpt.exe' in the System32 folder.

Cssrs.exe stores binary data from the resource section to '%SystemRoot%\system32\vjsc.dll'. When csssrs.exe stores vjsc.dll as a local file, it does not just store the data in the resource section. After RC4 decrypting the data in the resource section, cssrs.exe translates the binary data 'C:\Users\power\AppData\Local\Microsoft\Internet Explorer\CXXX.dll' to

'%SystemRoot%\system32\vjsc.dll'. The developer may use this file path to create malware. Subsequently, cssrs.exe sets the modified datetime value of the file to 10 years ago. This may be an attempt to prevent the file from standing out in the System32 folder.

Afterwards, cssrs.exe executes 'Scrpt.exe' using ShellExecuteExW. Scrpt.exe is a Microsoft Visual J# command-line tool and is a legitimate signed binary file. Scrpt.exe side-loads vjsc.dll, which is in the same directory, and executes an export function, 'VJSCCommandLineCompile'.

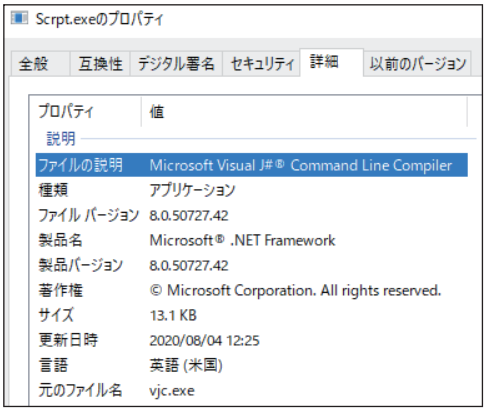


Figure 12: Property of Scrpt.exe.

First, vjsc.dll decrypts XOR'd data with the value 0x88. The decrypted strings are as follows:

- DFS Replication
- FTP Publishing Service
- ReadyBoost
- Software Licensing
- SL UI Notification Service
- Terminal Services Configuration
- Windows Media Center Extender Service
- Windows Media Center Service Launcher
- SOFTWARE\Microsoft\Windows NT\CurrentVersion\Svchost
- RegOpenKeyExA
- netsvcs
- RegQueryValueExA
- OpenSCManagerA
- %SystemRoot%\System32\svchost.exe -k netsvcs
- MACHINE\SYSTEM\CurrentControlSet\Services\
- RegSetValueExA
- GetSystemDirectoryA

After that, vjsc.dll decrypts XpEXPrint.dll with a random four characters. Then, it registers a service by using the decrypted strings above and runs.

Behaviour without admin privileges

Cssrs.exe loads the data (ID 161, 164 and 165) from the resource section and decrypts it with the key value 'L!Q@W#E$R%T^Y&U*A|}t~k' by using RC4. Each file is written to the folder 'AppData\Local\Microsoft\Internet Explorer'. The file names are as follows: ID 161 as vjsc.dll, ID 164 as Xwreg.exe, and ID 165 as Scrpt.exe.

The file store technique is the same as when cssrs.exe has admin privileges. Vjsc.dll has the string 'C:\Users\Waston\AppData\Local\Microsoft\Internet Explorer\FindX.exe' and Xwreg.exe has 'C:\Users\Waston\AppData\Local\Microsoft\Internet Explorer\WSMprovhost.exe' in each resource section, but they translate the paths to where they exist at that moment. Since 'C:\Users\Waston' is equivalent to the name of the Tmanger PDB path, it may be that the developer uses that environment when he/she creates malware. After that, cssrs.exe sets the modified datetime value of the file to 10 years ago.

Then, cssrs.exe executes ‘Scrpt.exe’ with ShellExecuteExW and side-loads vjsc.dll, which is in the same directory as cssrs.exe.

The internal name of vjsc.dll is RegAdd.dll. It creates an entry in the ‘CurrentVersion\Run’ registry key, and it registers Xwreg.exe, which is in the same directory as vjsc.dll, to launch automatically.

Afterwards, cssrs.exe deletes vjsc.dll and it loads the data of ID 162 from the resource section. The data is AES-256 decrypted using CryptDecrypt() from WINAPI. The key-generating string is ‘e4e5276c00001ff5’ and the decrypted data will be written into ‘AppData\Local\Microsoft\Internet Explorer’ under the name ‘vjsc.dll’.

Finally, cssrs.exe executes Scrpt.exe again with ShellExecuteExW and side-loads vjsc.dll, which is in the same directory.

XpEXPrint.dll

XpEXPrint.dll, which is launched with admin privileges, and vjsc.dll, which is launched without admin privileges via Scrpt.exe, are the same file.

In the case of launching XpEXPrint.dll as a service, it will create an event named ‘{A24D0DC3-E26A-4551-8F84-08E738AEC718}’. Their next behaviour is the same.

First, XpEXPrint.dll loads the data of ID 104 from the resource section. Then it AES-256 decrypts the loaded data. The decrypted data is ClientX. XpEXPrint.dll loads and executes the decrypted ClientX in memory. The decryption technique is the same as we described in the cssrs.exe section with Figure 11.

ClientX.dll

ClientX.dll is a RAT body like Tmanger’s Client. The name ‘ClientX.dll’ comes from the internal name of the DLL file.

Decrypt config

ClientX.dll decrypts the config data with the key value ‘L!Q@W#E$R%T^Y&U*Al}t~k’ by using RC4. The obtained config data is as follows:

- http[:]//go.vegispaceshop[.]org/shop.htm
- 0
- AppData
- Roaming
- 0.0.0.0:

It decrypts the User-Agent using RC4 as well. Afterwards, it gets the host name with gethostname and adds it to the User-Agent. The User-Agent is generated as shown below:

```
Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:71.0) Gecko/20100101 Firefox/71.0 [HOSTNAME])
```

Figure 13: Decrypted User-Agent.

Get C&C server IP address

After finishing RC4 decrypting, ClientX.dll accesses the decrypted URL to download the HTML file. Then ClientX.dll decrypts it to get new C&C server IP addresses (see Appendix 1). These decrypted IP addresses will be used for sending information about the infected host and executing commands.

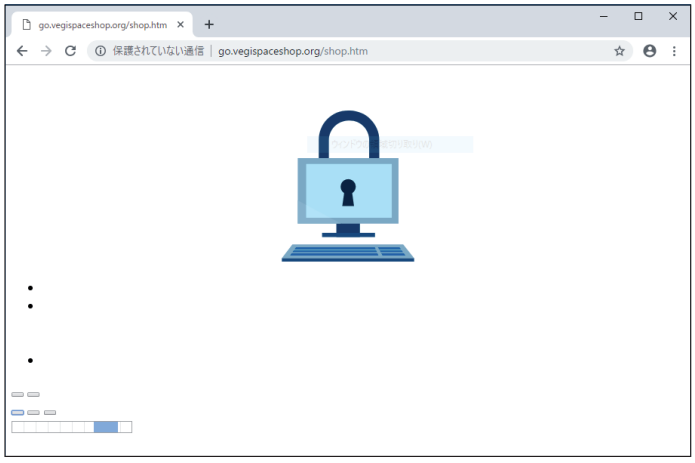


Figure 14: Downloaded HTML file from C&C server.

Steal information of the infected host

After decrypting IP addresses, ClientX.dll sends information to the C&C servers. The sending URL format is as shown in Figures 15 and 16.

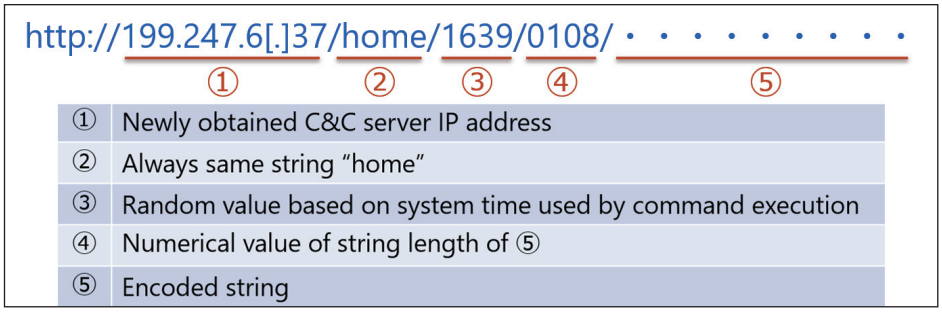


Figure 15: URL format.

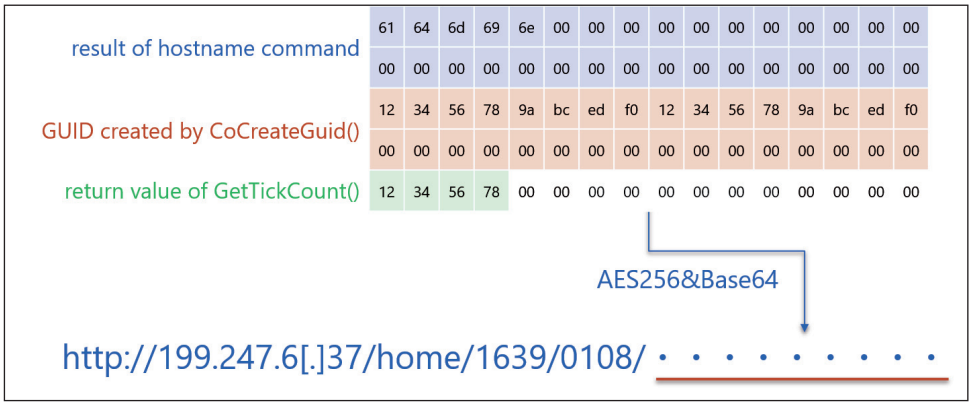


Figure 16: Encrypted string included in the path part of the URL.

The URL path contains the information about the infected host and the information that is needed for the traffic encryption command process. At the end of the URL path there is an encrypted string. To generate it, ClientX.dll encrypts the combined value (the result of the hostname command, the GUID generated by CoCreateGuid() and the return value of GetTickCount()) with AES-256 and encodes it with Base64. The encrypted string is added to the end of the URL path. (You can decrypt the encrypted string with our code in Appendix 2.)

Execute command

Next, ClientX.dll executes commands based on the receiving data from C&C servers. It can execute cmd.exe, and it uploads and downloads files.

Command ID	**Option**	**Description**
(command)	Argument(s)	Execute command by using cmd.exe and return the result to the C&C server
-upload	File Path of the infected host Uploading path of the URL	Upload file
-download	Download URL Stored file path	Download file
-exit		Do nothing

Table 4: ClientX command list.

When executing the commands the traffic is encrypted with AES-256. ClientX.dll decrypts the data as shown in Figure 17.

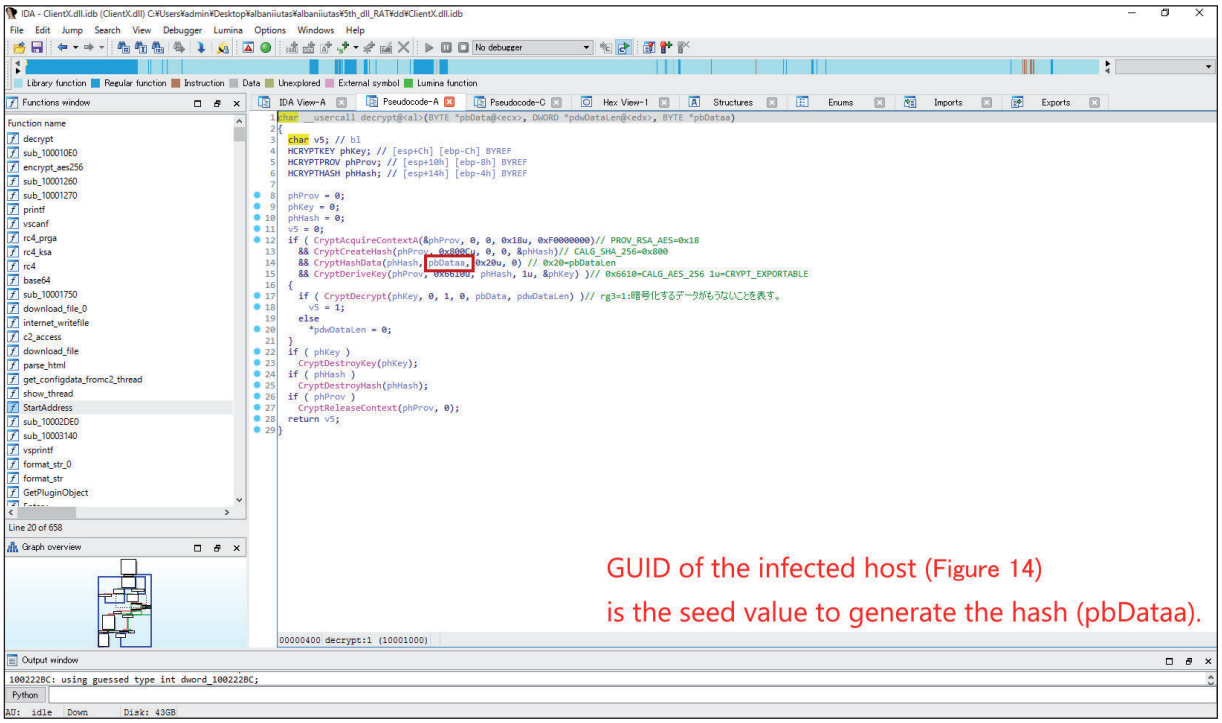


Figure 17: Decryption process of command data.

The receiving data format for the commands is as follows:

	①			②	③			②	
12	34	56	78	0b	31	31	31	31	0b
2d	75	70	6c	6f	61	64	20	･･	･･	･･	0b

④

①	In case of executing commands more than once, this value must be different one from previous time
②	Delimiter
③	If the value in not equal to ③ of the Figure 15, the command will not execute.
④	It means command ID and command option(s), and they are separated with space.

Figure 18: Data format received from C&C server.

Examples of received data for each command are shown in Figures 19 to 22.

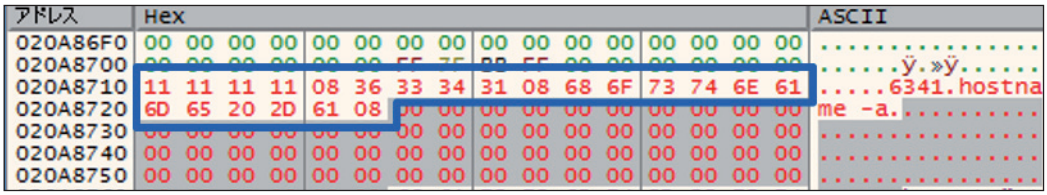


Figure 19: Example of received data for executing cmd.exe.

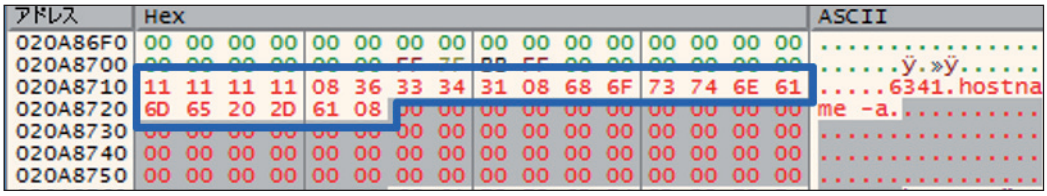


Figure 20: Example of received data for uploading file.

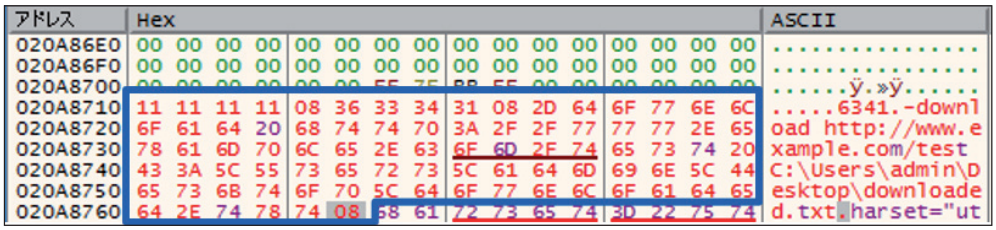


Figure 21: Example of received data for downloading file.

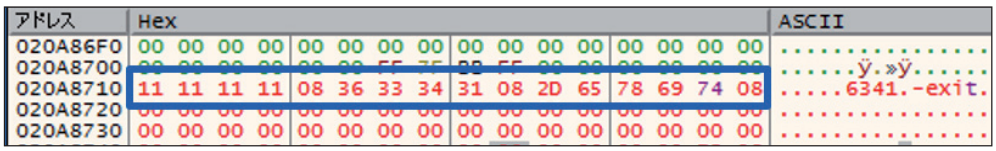


Figure 22: Example of received data for doing nothing.

Smanager

Since Smanager is often found in Vietnam, and it may be used in attacks against Vietnam-related organizations, it is different from Tmanger and Albaniiutas. However, it has a lot of similarities with Tmanger and Albaniiutas.

Attack flow

Like Tmanger, Setup, MloadDll and Client-like files exist in Smanager. We did not find a file that corresponds to Client in Tmanger, but we confirmed that MloadDll has functions to download and execute an executable file. We guess that this downloaded file is equivalent to Client. The Smanager attack flow is shown in Figure 23.

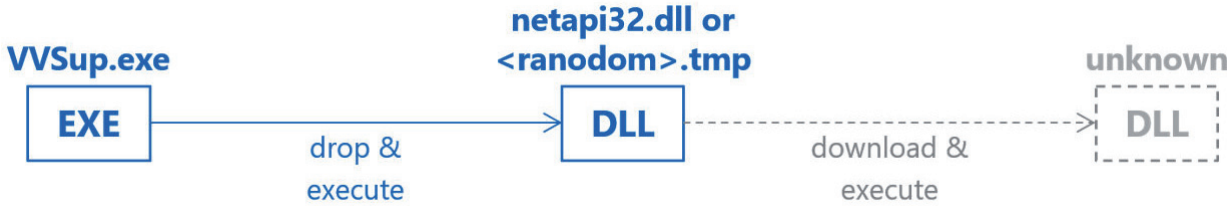


Figure 23: Attack flow of Smanager.

Classification of Tmanger	Smanager
Setup	VVSup.exe
MloadDll	netapi32.dll or <random>.tmp
Client	Unknown

Table 5: Corresponding Smanager elements to Tmanger classification.

VVSup.exe

VVSup.exe, which corresponds to the Setup file in Tmanger, opens and executes subsequent files. When VVSup.exe executes, it writes a CAB file to '%USERPROFILE%\test\7z.cab'. Afterwards, if it is executed as admin, it decompresses 7z.cab as 'C:\windows\apppatch\netapi32.dll', if not, '%TEMP%\\WMedia\[GetTickCount()].tmp' is extracted from 7z.cab. This file is in fact a DLL file, and its internal name is 'Smanager_ssl.dll'. The malware name 'Smanager' comes from the internal name of this DLL file.

Next, VVSup.exe searches for the data '192.168' in the DLL file, and determines the location of the Config data. Then it overwrites dummy data to the actual config data, with the exception of the string to generate the encrypt key. The encrypt key value is overwritten to the same value.

Dummy data	Actual config data
192.168.0.107:8888	vgca.homeunix[.]org:443
(null)	office365.blogdns[.]com:443
(null)	10[.]0.14.196:53
f4f5276c00001ff5	f4f5276c00001ff5

Table 6: Dummy data and actual config data.

If VVSup.exe is executed with admin privileges, it writes several registry keys like 'HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Svchost', registers the DLL file as a service and executes ServiceMain. If VVSup.exe is executed without admin privileges, 'Entery', which is one of the export functions in the DLL file, is executed by using rundll32.exe with WinExec.

Smanager_ssl.dll

Smanager_ssl.dll may correspond to MloadDll in Tmanger. It is opened and executed by VVSup.exe. Once executed, Smanager_ssl.dll establishes a connection to a C&C server. After establishing the connection, it authenticates and encrypts the connection by using the *Microsoft Security Service Provider Interface*.

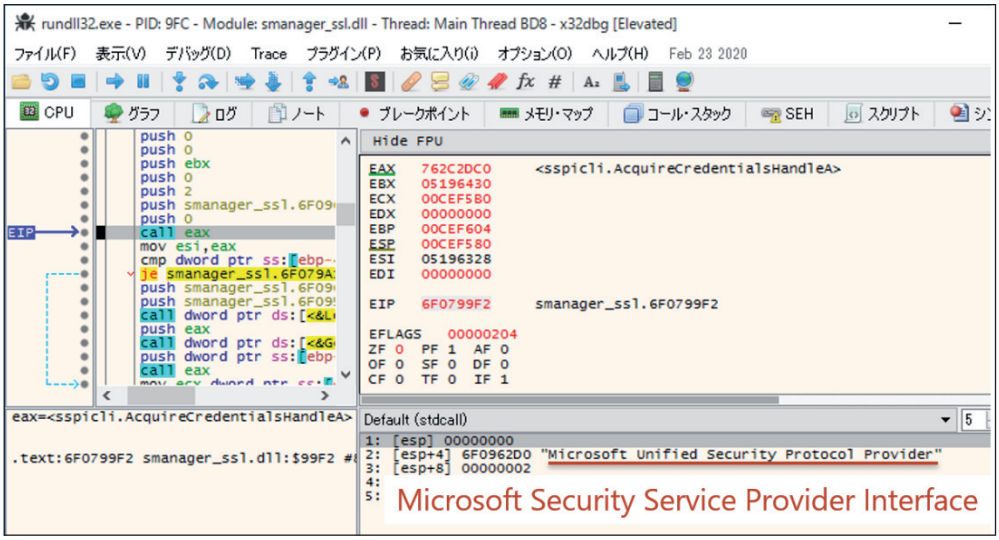


Figure 24: Established connection to a C&C server by Smanager_ssl.dll.

After establishing the connection with the C&C server, Smanager_ssl.dll executes commands it receives from the C&C server. The implemented commands are as followings:

- Send information about the infected host to the C&C server
- Download an executable file and execute it

The send command collects the following information from the infected host:

- Computer name
- Host name
- IP address
- OS version
- Language information
- Username
- Default browser
- Presence or absence of admin privileges

Smanager_ssl.dll is a RAT but it has only a few commands (send the infected host information, and download and execute an executable file). There is a lack of commands for intrusion activity. For this reason, we consider that the file downloaded by Smanager_ssl.dll has a role corresponding to Client in Tmanger.

Smanagerx64_release_tcp.dll

We found another file named Smanagerx64_release_tcp.dll. The behaviour of this file is like that of Smanager_ssl.dll (send the infected host information, and download and execute an executable file). In addition, it has export functions such as Entery and ServiceMain, like Smanager_ssl.dll. Therefore, it may be executed using the same method as Smanager_ssl.dll. However, since the functions such as authentication and encryption of traffic are not implemented, it is a little different from Smanager_ssl.dll. The Config of Smanagerx64_release_tcp.dll is as follows:

- coms.documentmeda[.]com:443
- f4f5276c00001ff5

FAMILY TREE

In this section, we will discuss the Tmanger family tree. In particular, we will focus on Tmanger, Albaniiutas and Smanager, which have been used since 2020. In order to trace back the relationships of the Tmanger family, we must introduce their aliases, which we summarize in Table 7. They are not strictly equal, because researchers have different opinions about the names of malware and attribution. However, we consider that following aliases are equivalent in many cases:

Our definition	**Alias**
Tmanger	LuckyBack
Albaniiutas	(BlueTraveller)
Smanager	PhantomNet, CoughingDown

Table 7: Malware aliases.

One thing that we should note as exceptional in the above table is the sample which we call 'Albaniiutas'. *Avast* labelled it as 'BlueTraveller' [10]. BlueTraveller was reported by *Kaspersky* in 2016. However, since we did not obtain a concrete sample, we do not have a definitive opinion about its origin and relation. For this reason, we will not discuss BlueTraveller in this paper.

ESET handles PhantomNet and Smanager as the same malware, and we agree with them. Yet, since we would like to introduce the transition of PhantomNet and Smanager in this chapter, we will strictly treat these as separate entities in this paper.

Table 8 and Figure 25 describe the relationships of the similar points of implementation and timeline among Tmanger, Albaniiutas and Smanager.

	Tmanger	**Albaniiutas**	**Smanager**
Common items through multi part			
Target	Mongolia	Mongolia	Vietnam
Output of debug message	True	True	True
Compile time is around 2025	True	True	False
Include 'Waston' in user path	True	True	False
Overwrite config data	True	True	True
Setup			
Check admin privileges	True	True	True
Compression algorithm	Deflate	Deflate	Cab
String to generate encrypt key	N/A	Including '276c00001ff5'	Including '276c00001ff5'
MloadDll			
Function 'Entery' is exported	True	True	True
Call export function 'GetPluginObject' from Client	False	True	True

Table 8: Similarities of Tmanger family.

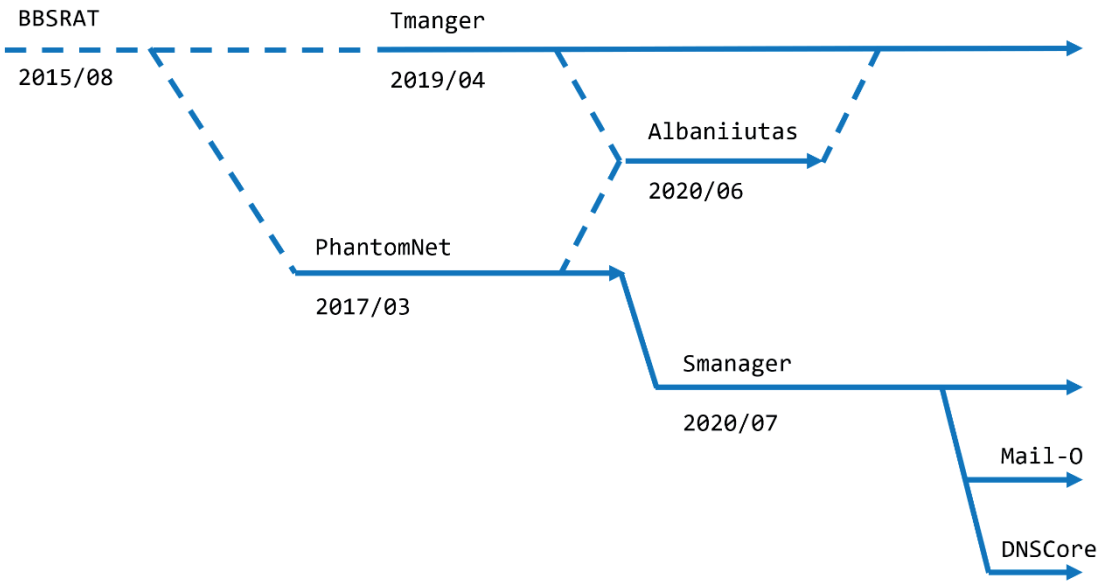


Figure 25: Tmanger family tree.

The oldest file sample that we found was PhantomNet from March 2017. At that time, PhantomNet might have been used to target organizations in the US or Singapore, until June 2020. After Smanager appeared in July 2020, PhantomNet disappeared. When you compare their implementations, you can clearly see that PhantomNet and Smanager are extremely similar. For this reason, we consider Smanager to be the successor of PhantomNet.

Now we will consider Tmanger and Albaniiutas. Table 9 shows the timestamps and PDB paths of Tmanger and Albaniiutas.

Name	Timestamp	Version	PDB
Tmanger	2025-08-19	1.0	C:\Users\Waston\Desktop\20190403_Tmanger\20191118 TM_NEW 1.0\Release\MloadDll.pdb
Tmanger	2025-10-05	4.4	C:\Users\Waston\Desktop\20190403_Tmanger\20191118 TM_NEW 4.4\Release\MloadDll_REG.pdb
Tmanger	2025-10-06	4.5	C:\Users\Waston\Desktop\20190403_Tmanger\20191118 TM_NEW 4.5\Release\MloadDll_REG_DLL.pdb
Tmanger	2020-03-16	4.5	C:\Users\sxpolaris\Desktop\2020\TM VS2015\TM_NEW 4.5\ Release\MloadDll.pdb
Albaniiutas	2025-12-10		
Tmanger	2026-04-23	6.2	

Table 9: Malware properties.

Since the timestamps in the samples of both Tmanger and Albaniiutas are around 2025, we guess that they have been compiled in the same environment. In addition, if the PDB paths are true, Tmanger started being created around April 2019. Furthermore, Albaniiutas was created between the creation times of Tmanger v4.5 and v6.2. In fact, looking at Tmanger v4.5 and v6.2, v6.2 has a lot of similarity with Albaniiutas, thus we can confirm the relationship between Tmanger and Albaniiutas more clearly. There is also a report that Tmanger and Albaniiutas were used in the same attack campaign. For those reasons, we consider Tmanger and Albaniiutas to be extremely close.

There is a report that Albaniiutas was used in June 2020 [10]. This is close to the term switching from PhantomNet to Smanager. The relationship between Albaniiutas and Tmanger is already clear, but Smanager has some similar implementations too. For those reasons, we consider that Albaniiutas, which originated from both Tmanger and Smanager, had been created in June 2020, then its features were merged to Tmanger.

After June 2020, their movement is as follows: since Tmanger v6.2 was released, Tmanger continues to be developed. However, we have not found any report of it being used in an actual attack. *Rostelecom* and *NKTsKI* describe Smanager-like malware named Mail-O in their reports in May 2021 [11, 12]. Furthermore, another piece of malware that may be based on Smanager (we call this malware 'DNSCore') appeared to use APT attacks against East Asia and Oceania in June 2021. For those reasons, we believe that the Tmanger and Smanager lineages may continue to be developed. Therefore, we need to pay attention to their movement.

ATTRIBUTION AND RELATIONSHIP

Some concrete incidents using Tmanger family malware were reported as shown in Table 10.

When	Group	Campaign	Target	Attack vector	Malware
February 2020	TA428	Operation LagTime IT	Mongolia	Spear phishing-> Royal Road RTF	Tmanger
June 2020	LuckyMouse		Mongolia	Spear phishing -> fake software	Tmanger, Albaniiutas
June 2020	LuckyMouse	Operation StealthyTrident	Mongolia	Supply chain	Tmanger
July 2020		Operation SignSight	Vietnam	Supply chain	Smanager

Table 10: Campaign information.

It is possible to say that the TA428 and LuckyMouse attacks in the above table have middle to high certainty. However, the threat groups that attacked Vietnam with supply chain attacks using Smanager have not been clarified.

As we mentioned earlier, Smanager may share code with Tmanger and Albaniiutas. However, Tmanger and Albaniiutas were used by TA428, which targets Russia and Mongolia, whereas Smanager was used to attack Vietnam. There is often some confusion between TA428 and TA413 (also known as KeyBoy and Tropic Trooper) – which has close affinity with TA428. TA413 targets Southeast Asia, including Vietnam. For this reason, we consider that this attack may relate to TA413.

Furthermore, FunnyDream – a group that overlaps some part of TA413 – has an association with PhantomNet, the predecessor to Smanager. Based on the data from an online sandbox, it appears that the FunnyDream backdoor, which is used by FunnyDream, might download PhantomNet.

A report [13] states that old PhantomNet samples use the same mutex as BBSRAT used by Roaming Tiger, as shown in Figure 24. In addition, 'Entery', which is one of the characteristic export functions used by Tmanger family malware, may come from 'Enter' in BBSRAT. Roaming Tiger is a threat group that may target Russia, Mongolia, etc., like TA428. Thus, it is possible to say that they are close to TA428.

```
int32_t main (void) {
    al = fcn_00402b80 ();
    if (al == 0) {
        goto label_1;
    }
    eax = uint32_t (*CreateMutexA)(void, void, char*) (0, 0, "Global\\GlobalAcprotectMutex");
    if (eax == 0) {
        goto label_1;
    }
    eax = uint32_t (*GetLastError)() ();
    if (eax == 0xb7) {
        goto label_1;
    }
```

Figure 26: Mutex of old PhantomNet.

Based on some attack cases against Russia and Mongolia, some researchers mention an association between Tmanger family malware and IronHusky (also known as Vicious Panda) [12]. IronHusky uses Royal Road RTF Weaponizer like TA428, but their detail such as the relationship with TA428 has not become clear yet.

In addition, since Tmanger family malware might be used in attacks against organizations related to East Asian and Oceanian countries, there is an indication that those pieces of malware may overlap with threat groups targeting East Asian countries, such as Tonto Team and Tick, which may be associated with TA428.

There are some pieces of malware and tools that may be shared not only by a single group but among several groups, like Tmanger family malware. Recent examples are Microsoft Exchange Exploit code, Royal Road RTF Weaponizer and ShadowPad [14, 15]. Threat groups that share these pieces of malware will be able to share other pieces of malware and tools among them. Therefore, it can be considered that Tmanger family malware may be shared more widely than now in the future. Table 11 shows the commonly known shared tools among multiple groups.

	Tmanger family	ShadowPad	Microsoft Exchange Exploit	Royal Road RTF Weaponizer
TA428	○	○		○
LuckyMouse	○		○	
TA413	△	○		
IronHusky	△			○
FunnyDream	△			○
Winnti		○	○	
Tonto		○	○	○
IceFog		○		○
Tick			○	○
Calypso			○	
Websiic			○	
Hellsing				○
Leviathan				○
Rancor				○
Naikon				○
Higaisa				○
TA410				○
Sharp Panda				○

Table 11: Threat groups that share tools/malware.

HUNTING

When you research an attack using Tmanger family malware, you need to pay particular attention to supply chain attack cases. Operation StealthyTrident and Operation SignSight are supply chain attacks which do not target famous software like *SolarWinds*, but a limited specific software. More in concrete, those threat groups targeted at the political or related organizations of Mongolia and Vietnam. These kinds of region-specific supply chain attacks have more limited impact and visibility than worldwide supply chain attacks, thus it is rare to discover these kinds of cases. Threat groups carry out attacks against their targets rather effectively. It is extremely difficult to detect an initial phase of intrusions like this.

When you research APT groups that might belong to China, it is effective to focus on their attributes. In particular, one of the most important points is to check the tools which share cases, as we mentioned earlier. We successfully extracted some related examples at this time. The first case in which we observed Tmanger was when we researched a sample of Royal Road RTF Weaponizer from TA428. In the process of researching a shared tool, we discovered another shared tool. A lot of APT groups use unique tools or malware. In particular, we should pay attention to the malware that is used in the core of the attack, as we found.

How could we trace back that family from Tmanger? First, we analysed Tmanger in detail, and then we searched for a database with its characteristic behaviours, strings and export functions. The important points for the research are as follows:

- Debug strings
- Event names
- Path names
- Encryption keys
- Export functions
- Load process of additional modules

CONCLUSION

As we explained in this paper, Tmanger family malware mainly consist of three roles (Setup, MloadDll and Client), and there are several shared points between Tmanger, Albaniiutas and Smanager.

Albaniiutas has some characteristic similarities ,such as future compile time, common PDB paths, etc., and its target is Mongolia, like Tmanger. However, we consider that Albaniiutas is an incoming or latest version of Tmanger, because it has some differences in some detailed parts of processes (connection handling to C2 servers, commands handling, etc.) and the timestamp information.

Smanager was found in Vietnam and its targets may be organizations related to Vietnam. However, Smanager has the characteristic features of Tmanger family malware, Setup and MloadDll, and it is very similar to Tmanger and Albaniiutas in some detailed points (the structure of the Config file, setting the value to permanent by using Service, etc.). Based on the compile times, Smanager might have been created between Tmanger and Albaniiutas.

Tracing back Tmanger family malware, PhantomNet, which may be the predecessor of Smanager, and BBSRAT, which has some similar characteristics to old PhantomNet, might be related, with Tmanger, Albaniiutas and Smanager appearing later. Tmanger v6.2 was observed in November 2020, and recently, Mail-O and DNSCore, which might be from Smanager, have also appeared.

Since Tmanger family malware seems to be developing day by day, it is possible that threat groups will use them continuously in the future. Furthermore, Tmanger family malware may be shared with other threat groups like Royal Road RTF Weaponizer and ShadowPad. Therefore, we should pay attention to their trends.

To protect yourself from Tmanger family malware attacks, we recommend leveraging the information that we have proposed in this paper for detecting and defending.

REFERENCES

[1] Raggi, M.; Saad, G. Attribution is in the object: using RTF object dimensions to track APT phishing weaponizers. VB2019. https://www.virusbulletin.com/conference/vb2019/abstracts/attribution-object-using-rtf-object-dimensions-track-apt-phishing-weaponizers.

[2] Jayanand, N.; Macalintal, I.; Ghosh, D. Curious tale of 8.t used by multiple campaigns against South Asia. VB2019. https://www.virusbulletin.com/conference/vb2019/abstracts/curious-tale-8t-used-multiple-campaigns-against-south-asia.

[3] Thomasen, T.; Kharouni, L. Chinese cyber espionage and the Belt & Road Initiative. VB2019. https://www.virusbulletin.com/conference/vb2019/abstracts/chinese-cyber-espionage-and-belt-road-initiative.

[4] Ozawa, F.; Hayashi, S.; Koike, R. Operation LagTime IT: colourful Panda footprint. VB2020. https://vb2020.vblocalhost.com/conference/presentations/operation-lagtime-it-colourful-panda-footprint/.

[5] Hada, H. Panda's New Arsenal: Part 1 Tmanger. NTT Security (Japan) KK. December 2020. https://insight-jp.nttsecurity.com/post/102gi9b/pandas-new-arsenal-part-1-tmanger.

[6] Hada, H. Panda's New Arsenal: Part 2 Albaniiutas. NTT Security (Japan) KK. December 2020. https://insight-jp.nttsecurity.com/post/102gkfp/pandas-new-arsenal-part-2-albaniiutas.

[7] Hada, H. Panda's New Arsenal: Part 3 Smanager. NTT Security (Japan) KK. December 2020. https://insight-jp.nttsecurity.com/post/102glv5/pandas-new-arsenal-part-3-smanager.

[8] Tartare, M. Operation StealthyTrident: corporate software under attack. ESET. December 2020. https://www.welivesecurity.com/2020/12/10/luckymouse-ta428-compromise-able-desktop/.

[9] Sanmillan, I.; Faou, M. Operation SignSight: Supply chain attack against a certification authority in Southeast Asia. ESET. December 2020. https://www.welivesecurity.com/2020/12/17/operation-signsight-supply-chain-attack-southeast-asia/.

[10] Camastra, L.; Morgenstern, I. APT Group Targeting Governmental Agencies in East Asia. Avast. December 2020. https://decoded.avast.io/luigicamastra/apt-group-targeting-governmental-agencies-in-east-asia/.

[11] Отчетоб исследованиисериикибератак на органыгосударственнойвластиРоссийской Федерации. Rosteecom and NKTsKI. https://rt-solar.ru/upload/iblock/b55/Ataki-na-FOIV_otchet-NKTSKI-i-Rostelekom_Solar_otkrytyy.pdf.

[12] Guerrero-Saade, J.A. ThunderCats Hack the FSB | Your Taxes Didn't Pay For This Op. Sentinel Labs. June 2021. https://labs.sentinelone.com/thundercats-hack-the-fsb-your-taxes-didnt-pay-for-this-op/.

[13] Lee, B.; Grunzweig, J. BBSRAT Attacks Targeting Russian Organizations Linked to Roaming Tiger. Palo Alto Networks. December 2015. https://unit42.paloaltonetworks.com/bbsrat-attacks-targeting-russian-organizations-linked-to-roaming-tiger/.

[14] Faou, M.; Tartare, M.; Dupuy, T. Exchange servers under siege from at least 10 APT groups. ESET. March 2021. https://www.welivesecurity.com/2021/03/10/exchange-servers-under-siege-10-apt-groups/.

[15] Koike, R.; Nakajima, S. Royal Road! Re:Dive. nao_sec. January 2021. https://nao-sec.org/2021/01/royal-road-redive.html.

IOCs

Family	SHA-256
Tmanger	977bd4b7e054b84b4b62e84875ff3277dd8c039cf3ee0ded435b41025d0d2b21
Tmanger	88ffb081f6924261df32322f343ccb9078ee45eaa369660892585037baf59078
Tmanger	8987b9587c1d4f6fbf2fa49eb11bb20b8b30b82d5bc988f5c882501b1f76b82a
Tmanger	85a53a2525643a84509b10d439734509203a2a74e1a167d5c3494e37a47c8c8c
Tmanger	86297be195acaa36ec042523a5484d9e14fd9fb4cbd977f709e75207358a3f86
Tmanger	5d3db73458eeeb6439ab921159ba447b01c7a12f7291eb4b5cf510e29a8137c6
Tmanger	ebe05801d32985dc954e754aed63b5cee6e889f26533b1635c1f47e42bcb483a
Tmanger	c60490f6fbda0a2cf1a8cd401b2f3ce9262e600268264229122a4d80e327ed4b
Tmanger	6fdd004d0835577749e8742c91e9f1720953faa8ecd55d3b203eddd4d6db5568
Tmanger	6fdd004d0835577749e8742c91e9f1720953faa8ecd55d3b203eddd4d6db5568
Tmanger	71fe3edbee0c27121386f9c01b723e1cfb416b7af093296bd967bbabdc706393
Tmanger	8109a33c573e00e7849ba2d63714703e2e7bd65dee1c2c6454951f7fc4b2f275
Tmanger	7807c0177cf37bce6e38ef534f804935f505a24d735baa53a18e2da766ec136b
Tmanger	4fcb79a73f5286ed8f2bc671b64c76dac4971a0cce10936f63d210e8e17c5fd5
Tmanger	e494c8916e93295338a7368f86c42fce0916b559e63d462bd1b3265b6009bf9b
Tmanger	d4b339f502119d4cf10d48c8c7297bbaebb22387eb7cc4447540b666d27ba166
Tmanger	078498d02775b64c5660ccbfdf12f31f3b810ed612e10c3dd50660cfa03ad470
Tmanger	afd457592715bdef21d02c4e4d0e80dd70cf801a9d4d9afed795494012994372
Tmanger	772e69b3d66ef5b4fdda49d3ca39a5459b8c3afce77c24ebda698aef5bdbc5c3
Tmanger	8e9fc7bd0673a88a04583dda7d42f278013aa7abc4e26de86e953cc4a6825708
Tmanger	2999e5209cf1d7fb484832278e11e4c4950ef40e8f52a44329ed4230135f9b64
Albaniiutas	5eb4a19fbd25ecdabf2a456a23251f13fa938400cb32cfe87a62e8c168f9b841
Albaniiutas	29152de94199d77b0da9fc89d5b80bd4692f4aadf9e8362a2aee0a3b455c4e76
Albaniiutas	fd43fa2e70bcc3b602363667560494229287bf4716638477889ae3f816efc705
Albaniiutas	cf36344673a036f5a96c1c63230c9c15bb5e4f440eafd4ba0dc01d44bb1df3bf
Albaniiutas	d94f404b2b5bafa0d9ce66219b2684186715f5ef20a69f036a06d465177d5769
Albaniiutas	71750c58eee35107db1a8e4d583f3b1a918dbffbd42a6c870b100a98fd0342e0
Smanager/PhantomNet	f659b269fbe4128588f7a2fa4d6022cc74e508d28eee05c5aff26cc23b7bd1a5
Smanager/PhantomNet	1d9bc6939e2eceb3e912f158e05e04cadc1965849c4eb2c96e37e51a7d4f7aa5
Smanager/PhantomNet	97a5fe1d2174e9d34cee8c1d6751bf01f99d8f40b1ae0bce205b8f2f0483225c
Smanager/PhantomNet	02f1244310dd527d407ebcef07c5431306c56c1b28272b8d4e59902b3df537c8
Smanager/PhantomNet	c129d892a5e2d17c38950fdf77a0838edc1fa297a4787414e90906f7cb8f43b8
Smanager/PhantomNet	1fff4faa83678564aefb30363f0cbe2917d2a037d3d8e829a496e8fd1eca24c9
Smanager/PhantomNet	58012504861dee4663ecaa4f2b93ca245521103f4c653b2dd0032a583db8f0af
Smanager/PhantomNet	17bc9b7c7df4acd42e795591731e568cb040d6908d892f853af777d5f05c8806
Smanager/PhantomNet	338502691f6861ae54e651a25a08e62eeca9febc6830978a670d44caf3d5d056
Smanager/PhantomNet	d5f96b3b677ac68e45d4297e392b14a52678c2758a4030d2f6ad158027508c6d
Smanager/PhantomNet	00badf016953ec740b61f4ba27c5886a6460f6abba98819e00bde51574e0ebf4
Smanager/PhantomNet	e8156ec1706716cada6f57b6b8ccc9fb0eb5debe906ac45bdc2b26099695b8f5
Smanager/PhantomNet	feaba29072531b312e3bd0152b9c17c48901db7c8d31019944e453ca9b1572e2

APPENDIX 1

Decoding HTML file to C&C server IP addresses by ClientX.dll:

```
import sys
def main():
    result = decode_bin(extract_data(read_htmlfile()))
    with open("result.bin","wb") as f:
       for b in result:
           f.write(b.to_bytes(1,byteorder="big"))

def extract_data(input_bin):
    index = 0
    data = list()
    while index+2 < len(input_bin):
       if input_bin[index] != 0x3E or input_bin[index+1] != 0x9:
           index = index + 1
           continue
       index = index + 2
       sub_bin_ary = get_unitl_cr(input_bin[index:])
       if len(sub_bin_ary) == 0:
           return list()
       data = data + sub_bin_ary
       index = index + len(sub_bin_ary) + 1
    return data

def get_unitl_cr(bin_ary):
    if len(bin_ary) == 0 or bin_ary[0] == 0x0d:
       return list()
    last_index = len(bin_ary) -1
    ret = list()
    for ii in range(0,len(bin_ary)-1):
       if bin_ary[ii] == 0x0d:
           break
       if bin_ary[ii] == 0x0a:
           return list()
       ret.append(bin_ary[ii])
    return ret

def decode_bin(bin_ary):
    ret = list()
    for ii in range(0,len(bin_ary)//8):
       ret.append(0)
    start_index = 0
    for ii in range(7,-1,-1):
       kk = 0
       while kk < (len(bin_ary)//8):
           val = bin_ary[kk*8+start_index]
           ret[kk] += ( val & 0x1 ) << ii
           kk += 1
       start_index = start_index + 1
    return ret

def read_htmlfile():
    with open(sys.argv[1],"rb") as f:
       return f.read()

if __name__ == "__main__":
    main()
```

APPENDIX 2

Code to decrypt string added to end of URL in Figure 15:

```
import hashlib
from Crypto.Cipher import AES
from binascii import a2b_hex
import base64
BS = 16
def aes_decrypt(hash_seed,encrypted_data):
    aes_key = hashlib.sha256(hash_seed).hexdigest()
    encrypted_data = pcks_pad(encrypted_data)
    cipher_aes = AES.new(key=a2b_hex(aes_key),mode=AES.MODE_CBC,IV=b"\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00")
    return cipher_aes.decrypt(encrypted_data)

def pcks_pad(raw):
    result = list(raw)
    pad_ch = BS - len(result) % BS
    for ii in range(0,pad_ch):
       result.append(pad_ch)
    return bytes(result)

hash_seed = "303330343237366366346633313334350000000000000000000000000000000000"
encrypted = "426d62484c4d2f495a734a6c444a716d335037784e6451534c6a534761636a6954694864454d463
6776a6347366e71692f43457a52366e5a583472734f4b38545a4479304d6e77556c4c4754444e4936476e6675774b6
c657a72353749494d6b7134513171633469547 64d3d"
encrypted = a2b_hex(encrypted)
plain = aes_decrypt(a2b_hex(hash_seed),base64.b64decode(encrypted))
print("DecryptedData:" + bytes.hex(plain))
```