MBSD Blog

Understanding internal structure of the SNAKE (EKANS) ransomware

Takashi Yoshikawa
Senior Malware Analyst, Cyber Intelligence Group
Mitsu Bussan Secure Directions, Inc.

Kei Sugawara
Senior Malware Analyst, Cyber Intelligence Group
Mitsu Bussan Secure Directions, Inc.

June 2020

Table of Contents

1. Introduction

The news has been running around the world that Honda has been suffered by cyber attacks in the past few days.
In this article, we analyze the sample (*) of SNAKE ransomware that was uploaded to VirusTotal, and would like to share the information we found through our analysis. Although some information has already been published outside Japan, it seems that there are little information available in Japanese, so I would appreciate it if you could use it as a reference.

※ Hash value : d4da69e424241c291c173c8b3756639c654432706e7def5025a649730868c4a1

We do not know if this sample is related to the cyber attack against Honda. Please note that this article only describes analysis results of the sample with the above hash value.

2. Specimen

First of all, the sample in question was uploaded from Japan to VirusTotal around June 8th as follows.

It was uploaded from Japan to VirusTotal, because the SUBMISSIONS Country indicates "JP"

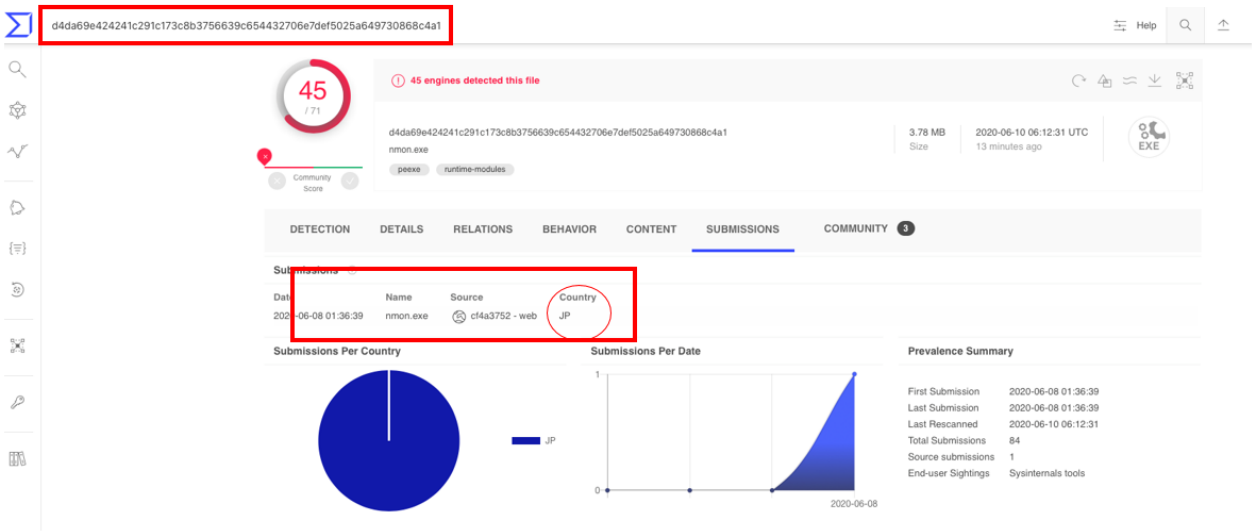


Fig. 1 - Uploaded to VirusTotal from Japan

SNAKE ransomware, also known as EKANS(SNAKE reverse reading), is a new ransomware that emerged in December 2019.

SNAKE body was developed in the GO-language, a relatively new open-source development language developed by google about a decade ago, and compiled into the EXE-format. One of the backgrounds created in the GO language is that it can be developed on multiple platforms, which is considered to be an advantage of the developer.

The following illustration shows the strings inside malware, that indicates that it was developed in the GO language with an account name of Admin3 has been developed in Windows environment.

SNAKE core is written in GO language, and developed by an attacker with an account name of Admin3.

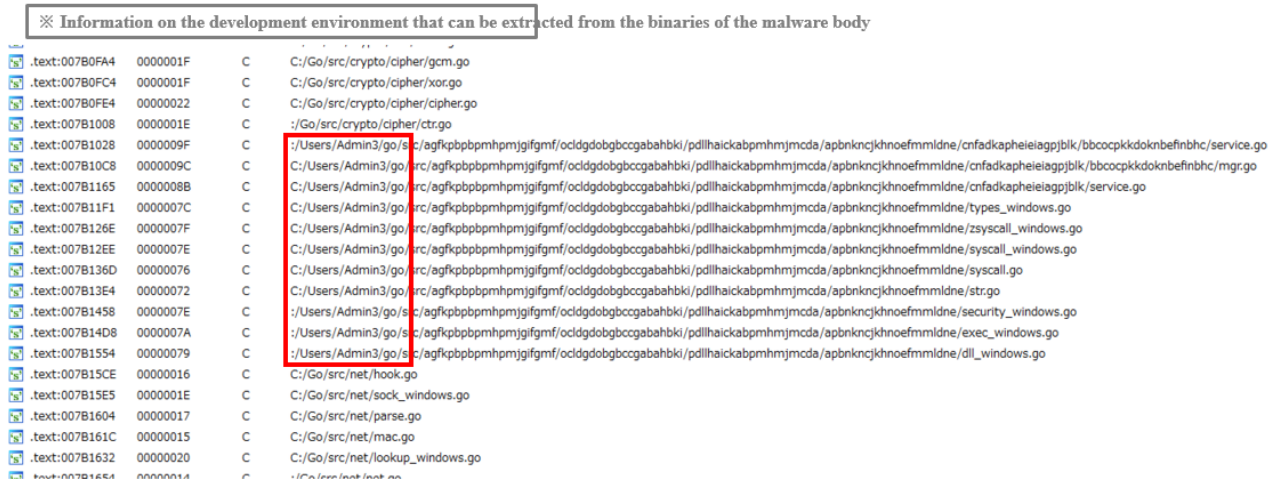


Fig. 2 - SNAKE was developed in GO language on Windows platform

(note) PE files developed in GO have fixed compilation date and time value of 1970/1/1 because Time Date Stamp is set to 0x0.

If it was written in GO language, Time Date Stamp (*) is set to 0x0, so the compilation date and time is fixed at 1970/1/1. (* Time Date Stamp: Normally, the date and time when the EXE file was compiled.)

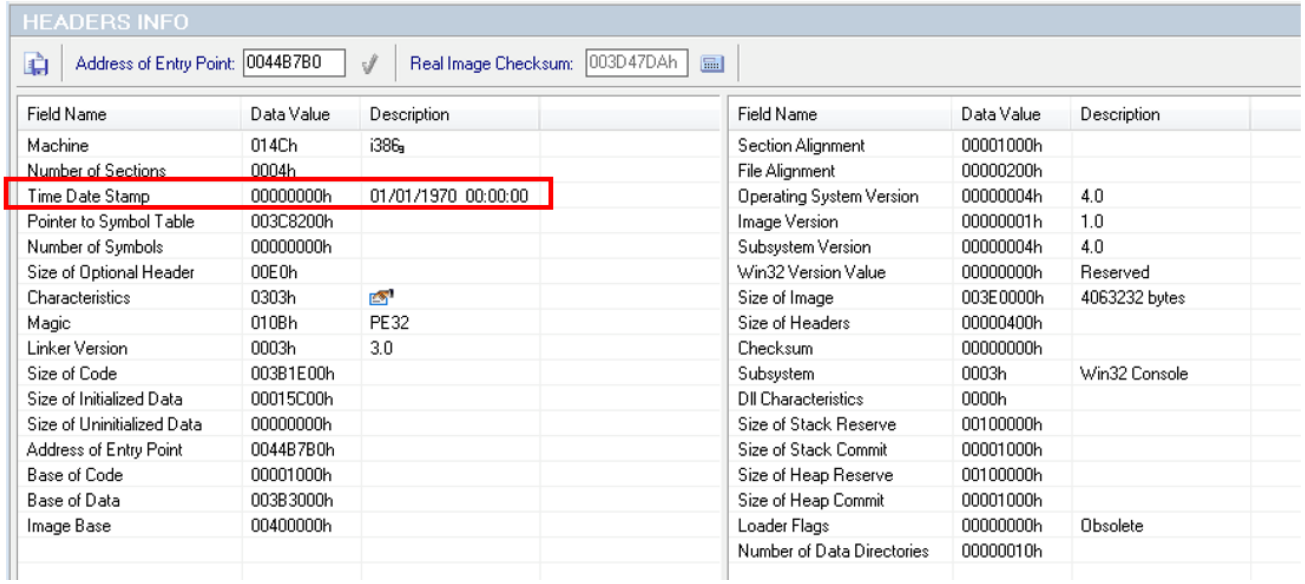


Fig. 3 - A PE-file developed in GO language includes fixed Time Date Stamp

When SNAKE is started, it registers a mutex called "EKANS" to the system.
This prevents the activation of other EKANS while SNAKE is already in operation, preventing unintentional multiple infections of SNAKE in the same PC.

Register a mutex called "EKANS" to the system so that the ransomware does not start multiple times after startup.
(This operation prevents from multiple infection to the same ransomware by mistake)

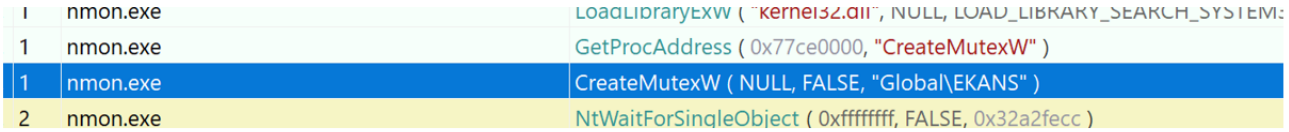


Fig. 4 - Registering the mutex "EKANS" to prevent multiple infections

Next, the following specific-domain name resolution is checked as a unique behavior of this particular SNAKE:

- **MDS[.]HONDA[.]COM**

In addition, check if name resolution for the above domain is equal to the following IP address, if not, terminate the operation, if it is equal, continue the operation.

- **170[.]108[.]71[.]15**

In other words, this specimen is a ransomware specially built for a unique set of name resolution and IP address.
As we've said in our previous blogs, targeted ransomware in recent years tends to create behaviors tailored to the target, so even seemingly the same type of ransomware often vary in motion.

Incidentally, at the time of the survey, 170[.]108[.]71[.]15 can be reverse-looked up as the following host name:

- **Unspec170108[.]amerhonda[.]com**
  (Organization: American Honda Motor Company, Inc. (AHMC-Z))

Check if the result of name resolution of MDS.HONDA.COM is 170.108.71.15. If not, terminate the process.

The host name of the applicable IP address is unspec170108.amerhonda.com (Organization: American Honda Motor Company, Inc. (AHMC-Z))

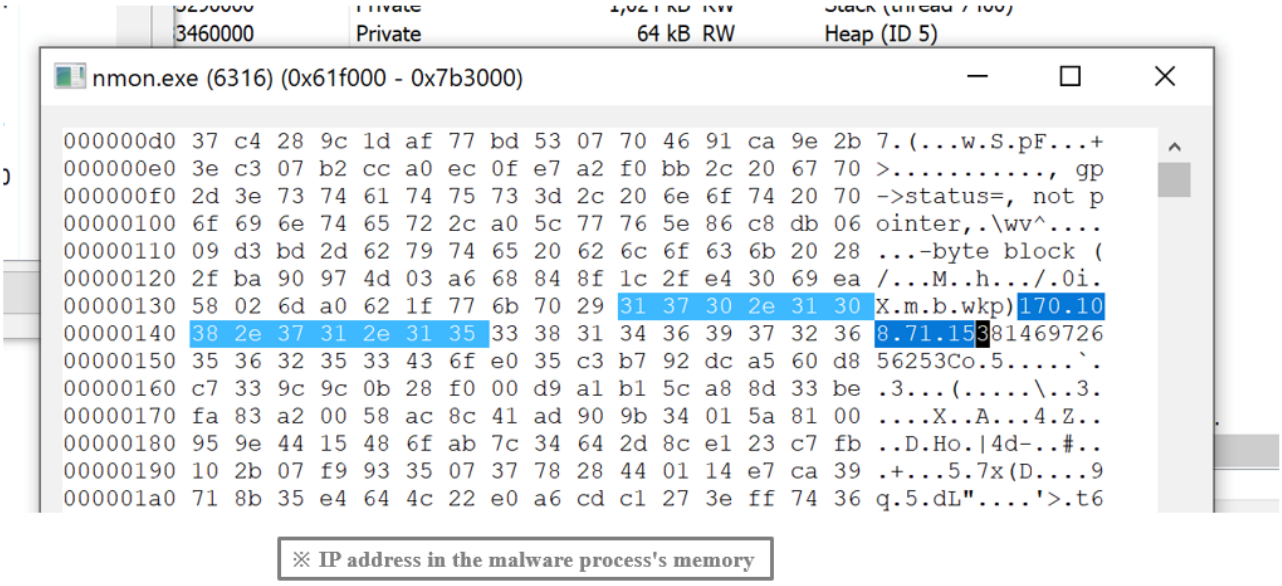


Fig. 5 - Checking specific IP address

Following discusses more detail around name resolution.

Following shows the behavior of SNAKE in case name resolution failed to match to the name of the relevant domain.
(SNAKE behavior is shown in time order from top to bottom in the following picture)

The part with red frame is a process resolving the name of the domain. In case the domain cannot be resolved, errors are returned to GetAddrInfoW, and SNAKE stops working and terminates. In another words, it does not work on PCs that cannot resolve MDS[.]HONDA[.]COM.

Behavior of the SNAKE when name resolution (MDS.HONDA.COM) was not possible.
A function GetAddrInfoW that subtracts the IP address from the domain results in an error. Then SNAKE terminates.

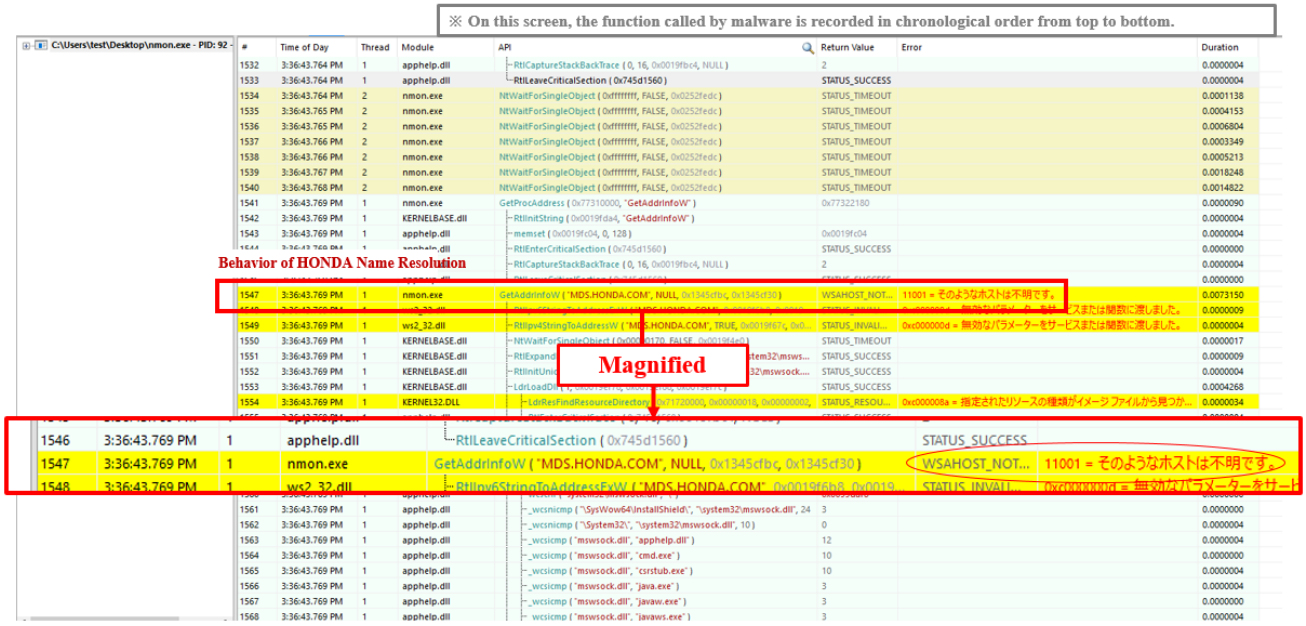


Fig. 6 - Immediately finished when name resolution fails

On the other hand, If you run SNAKE in a spoofed environment so that 170[.]108[.]71[.]15 is responded against MDS[.]HONDA[.]COM name resolution, a successful (= ERROR_SUCCESS) reply is returned and SNAKE continues to operate afterwards, as follows:
In other words, this ransomware is made to work only in the particular environment in which 170[.]108[.]71[.]15 is responded against MDS[.]HONDA[.]COM name resolution.

Behavior in a HONDA emulated environment so that MDS.name.COM can be resolved to a specific IP address.
A function GetAddrInfoW that translates from the domain name to the IP address succeeds, and then SNAKE continues to operate.

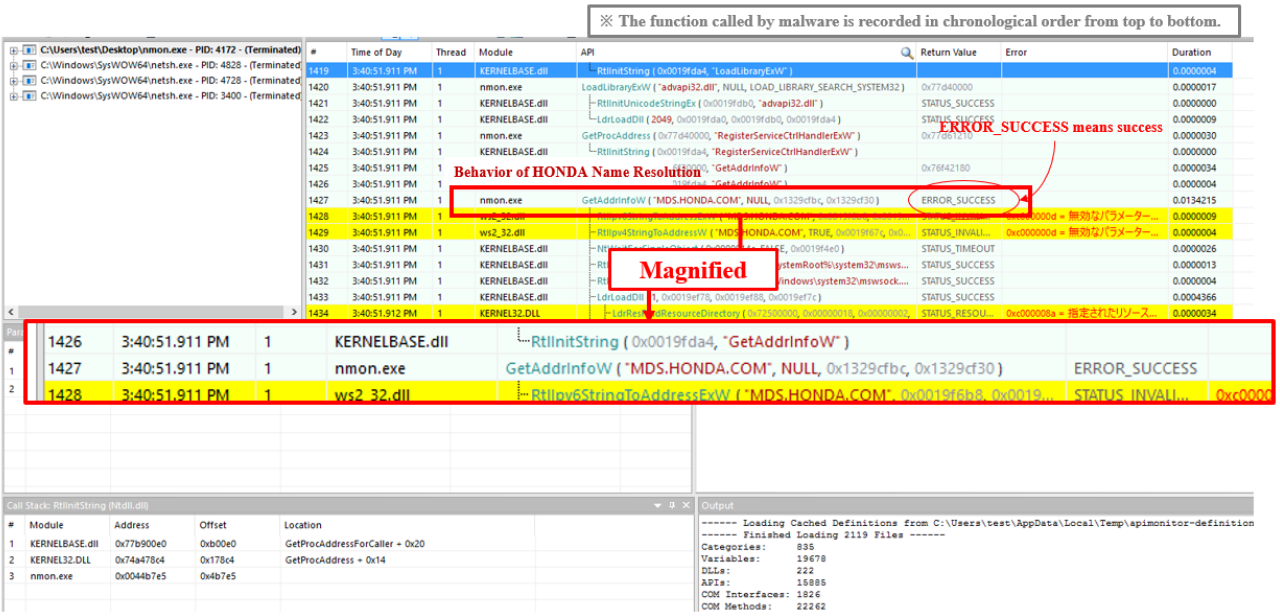


Fig. 7 - Operation continues when name resolution is possible for a specific IP address

These approaches make it impossible to grasp the behavior of this specimen in an automated analysis environment or an immediate inspection environment where the name resolution fails, and thus it could be one of the effective analysis/survey jamming.

3. Activities in environments where name resolution is available

Once SNAKE realizes it is a good operating environment (that is, an environment where name resolution stated above is succeeded), now SNAKE changes Windows firewall settings.
Specifically, following command is passed to netsh.exe (network tool, comes with Windows):

```
Netsh advfirewall set allprofiles firewallpolicy blockinbound,blockoutbound
```

This will block all incoming and outgoing connections for all profiles that do not match the ingress and egress rules using Windows firewall function.

Use netsh.exe(*) to change the firewall settings.
(* netsh.exe: a program that performs network control that comes with Windows)

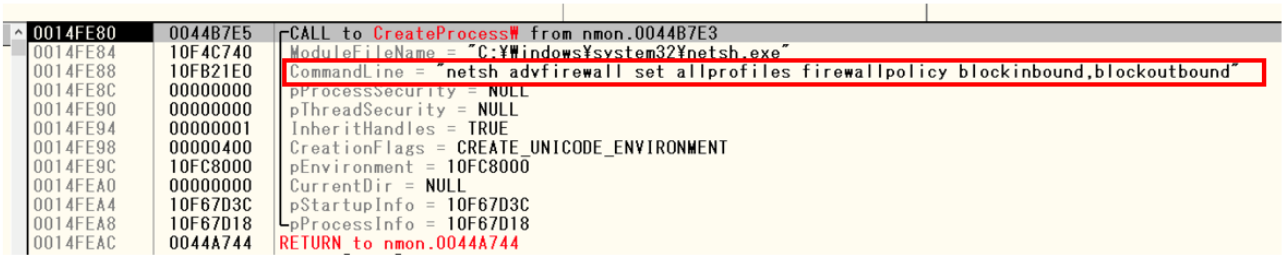
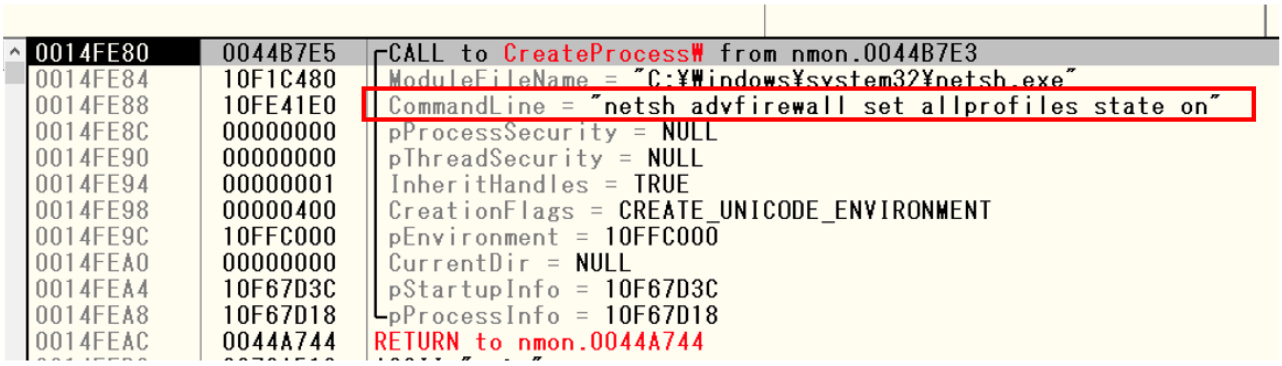


Fig. 8 - Modifying Windows firewall settings

Then enable the firewall configuration by passing the following command to netsh.exe:

```
Netsh advfirewall set allprofiles state on
```

Use netsh.exe to activate the firewall settings.



This makes it impossible to send and receive all communications that do not match the firewall's rules.
(that is, many applications become unable to communicate)

Fig. 9 - Enabling Windows firewall settings

As a result, all communications that do not match the firewall's existing rules are blocked, making many applications unable to communicate. This mechanism, which uses legitimate Windows firewalls to block network communication during encryption, is a unique phenomenon of this ransomware.

Then, attempt to stop multiple services, such as system or security system.

Attempt to stop several services, such as EventLog.

```
CloseServiceHandle ( 0x008366d0 )                                                                                    TRUE
OpenSCManagerW ( NULL, NULL, SC_MANAGER_ALL_ACCESS )                                                                 0x008364f0
NtWaitForSingleObject ( 0xffffffff, FALSE, 0x32a2fecc )                                                              STATUS_TIMEOUT
OpenServiceW ( 0x008364f0, "EventLog", SERVICE_ALL_ACCESS )                                                          0x008361f8
ControlService ( 0x008361f8, SERVICE_CONTROL_STOP, 0x12976aa8 )                                                      FALSE
FormatMessageW ( FORMAT_MESSAGE_ARGUMENT_ARRAY | FORMAT_MESSAGE_FROM_SYSTEM | FORMAT_MESSAGE_IGNORE_INSERTS , NULL, 1051, 1033, 0x12...  89
NtWaitForSingleObject ( 0xffffffff, FALSE, 0x32a2fecc )                                                              STATUS_TIMEOUT
CloseServiceHandle ( 0x008361f8 )                                                                                    TRUE
CloseServiceHandle ( 0x008364f0 )                                                                                    TRUE
OpenSCManagerW ( NULL, NULL, SC_MANAGER_ALL_ACCESS )                                                                 0x00836518
OpenServiceW ( 0x00836518, "SamSs", SERVICE_ALL_ACCESS )                                                             0x008362c0
ControlService ( 0x008362c0, SERVICE_CONTROL_STOP, 0x12976aa8 )                                                      FALSE
NtWaitForSingleObject ( 0xffffffff, FALSE, 0x32a2fecc )                                                              STATUS_TIMEOUT
FormatMessageW ( FORMAT_MESSAGE_ARGUMENT_ARRAY | FORMAT_MESSAGE_FROM_SYSTEM | FORMAT_MESSAGE_IGNORE_INSERTS , NULL, 1052, 1033, 0x12...  54
CloseServiceHandle ( 0x008362c0 )                                                                                    TRUE
CloseServiceHandle ( 0x00836518 )                                                                                    TRUE
```

Service of security products is also subject to stop.

```
8B6C24 24          mov ebp,dword ptr ss:[esp+24]
81FE 2C010000      cmp esi,12C                                  esi:"Sophos Health Service"
0F8D 93040000      jge nmon.551C3A
89B424 50020000    mov dword ptr ss:[esp+250],esi
8DBCF4 4C0A0000    lea edi,dword ptr ss:[esp+esi*8+A4C]
8B37               mov esi,dword ptr ds:[edi]                   esi:"Sophos Health Service"
8B7F 04            mov edi,dword ptr ds:[edi+4]
39CD               cmp ebp,ecx
0F83 35050000      jae nmon.551CF7
89B424 E4040000    mov dword ptr ss:[esp+4E4],esi               [esp+4E4]:"Sophos File Scanner Service"
897C24 28          mov dword ptr ss:[esp+28],edi
8D0CE8             lea ecx,dword ptr ds:[eax+ebp*8]             [eax+ebp*8]]:"AJRouter"
898C24 1C070000    mov dword ptr ss:[esp+71C],ecx               [esp+71C]:&"AJRouter"
8B11               mov edx,dword ptr ds:[ecx]                   edx:&"AllJoyn Router Service"
8B59 04            mov ebx,dword ptr ds:[ecx+4]
891424             mov dword ptr ss:[esp],edx                   [esp]:"llJoyn Router Service"
895C24 04          mov dword ptr ss:[esp+4],ebx
897424 08          mov dword ptr ss:[esp+8],esi                 [esp+8]:"Sophos File Scanner Service"
897C24 0C          mov dword ptr ss:[esp+C],edi
```

Fig. 10 - Attempts to stop various services

In addition, multiple legitimate processes are forcibly terminated, which may interfere with encryption and recovery activities.

Forcibly terminate multiple regular processes, because they may interfere with encryption or recovery activities.

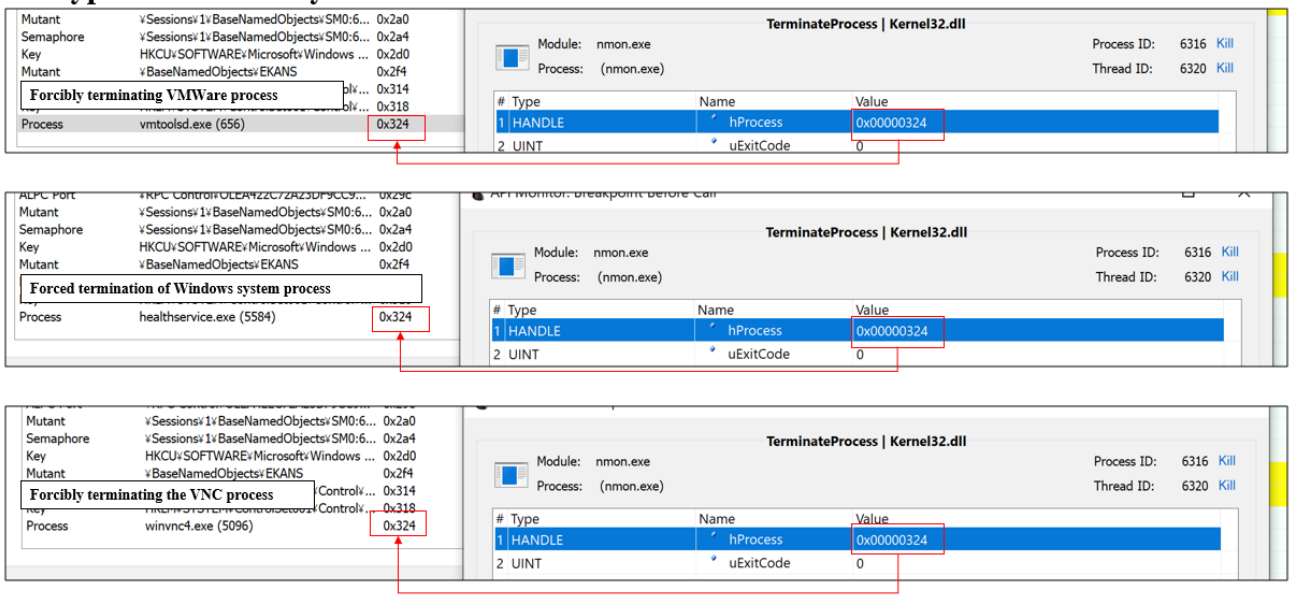


Fig. 11 - Forcibly terminates various legitimate processes

4. Encryption process

After the environment settings are ready, encryption of the files that is the main process of the ransomware begins.
File encryption takes all drives available in the system and encrypts them in order from the beginning of the drive.
The following image shows how the encryption process begins, and it is searching to encrypt the files in Recycle.Bin (Trash folder) found at the beginning of the C drive. After that, the accessible files contained in each folder of the PC are encrypted one after the other.

All available drives in the system are targeted. The encryption starts in order from the beginning of the drive.

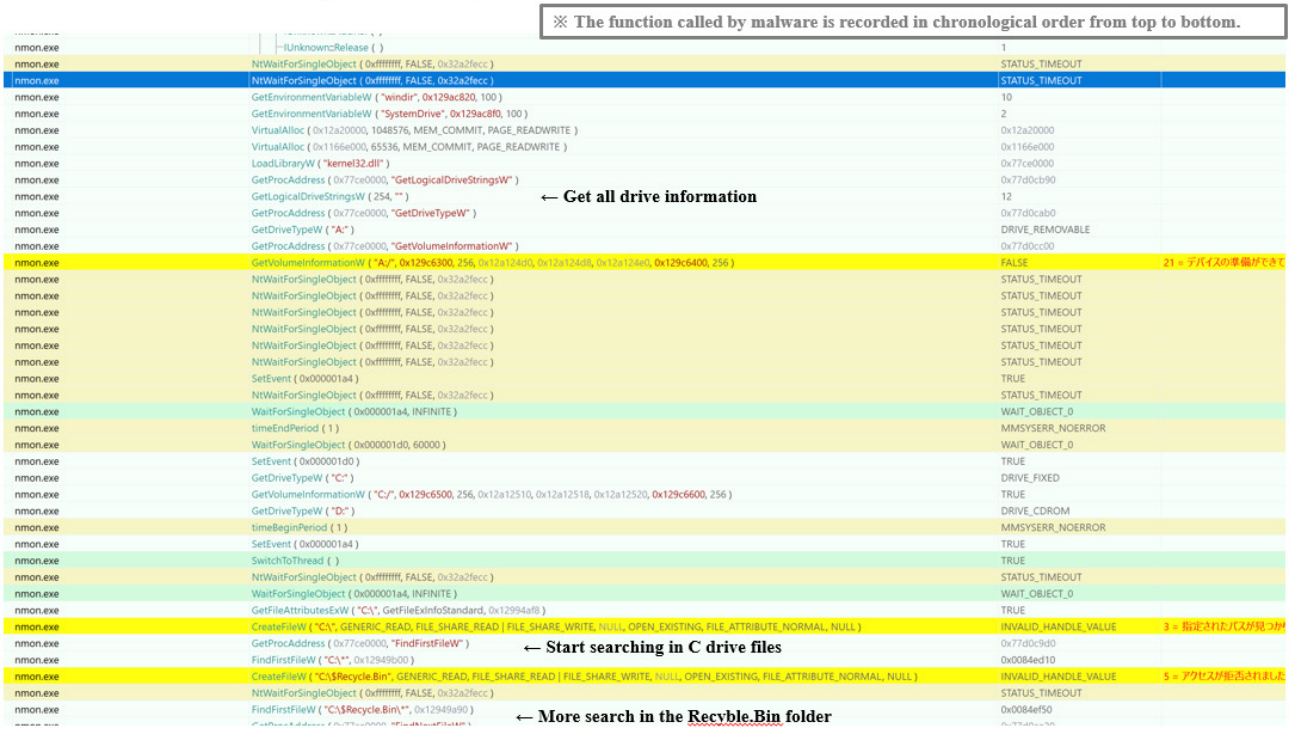
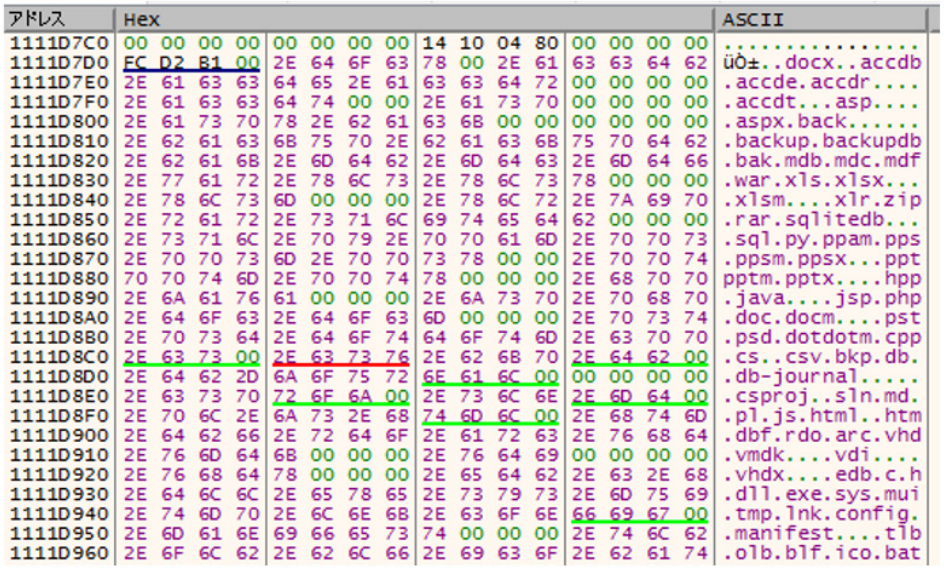


Fig. 12 - Flow of file encryption (portion)

The following file extensions (portion) for encryption are pre-determined.

Target File Extensions list (part of the file)



アドレス	Hex												ASCII
1111D7C0	00	00	00	00	00	00	00	00	14	10	04	80	00	00	00	00	.....
1111D7D0	FC	D2	B1	00	2E	64	6F	63	78	00	2E	61	63	63	64	62	üÒ±..docx..accdb
1111D7E0	2E	61	63	63	64	65	2E	61	63	63	64	72	00	00	00	00	.accde.accdr....
1111D7F0	2E	61	63	63	64	74	00	00	2E	61	73	70	00	00	00	00	.accdt...asp....
1111D800	2E	61	73	70	78	2E	62	61	63	6B	00	00	00	00	00	00	.aspx.back.....
1111D810	2E	62	61	63	6B	75	70	2E	62	61	63	6B	75	70	64	62	.backup.backupdb
1111D820	2E	62	61	6B	2E	6D	64	62	2E	6D	64	63	2E	6D	64	66	.bak.mdb.mdc.mdf
1111D830	2E	77	61	72	2E	78	6C	73	2E	78	6C	73	78	00	00	00	.war.xls.xlsx...
1111D840	2E	78	6C	73	6D	00	00	00	2E	78	6C	72	2E	7A	69	70	.xlsm....xlr.zip
1111D850	2E	72	61	72	2E	73	71	6C	69	74	65	64	62	00	00	00	.rar.sqlitedb...
1111D860	2E	73	71	6C	2E	70	79	2E	70	70	61	6D	2E	70	70	73	.sql.py.ppam.pps
1111D870	2E	70	70	73	6D	2E	70	70	73	78	00	00	2E	70	70	74	.ppsm.ppsx...ppt
1111D880	70	70	74	6D	2E	70	70	74	78	00	00	00	2E	68	70	70	pptm.pptx....hpp
1111D890	2E	6A	61	76	61	00	00	00	2E	6A	73	70	2E	70	68	70	.java....jsp.php
1111D8A0	2E	64	6F	63	2E	64	6F	63	6D	00	00	00	2E	70	73	74	.doc.docm....pst
1111D8B0	2E	70	73	64	2E	64	6F	74	64	6F	74	6D	2E	63	70	70	.psd.dotdotm.cpp
1111D8C0	2E	63	73	00	2E	63	73	76	2E	62	6B	70	2E	64	62	00	.cs..csv.bkp.db.
1111D8D0	2E	64	62	2D	6A	6F	75	72	6E	61	6C	00	00	00	00	00	.db-journal.....
1111D8E0	2E	63	73	70	72	6F	6A	00	2E	73	6C	6E	2E	6D	64	00	.csproj..sln.md.
1111D8F0	2E	70	6C	2E	6A	73	2E	68	74	6D	6C	00	2E	68	74	6D	.pl.js.html..htm
1111D900	2E	64	62	66	2E	72	64	6F	2E	61	72	63	2E	76	68	64	.dbf.rdo.arc.vhd
1111D910	2E	76	6D	64	6B	00	00	00	2E	76	64	69	00	00	00	00	.vmdk....vdi....
1111D920	2E	76	68	64	78	00	00	00	2E	65	64	62	2E	63	2E	68	.vhdx....edb.c.h
1111D930	2E	64	6C	6C	2E	65	78	65	2E	73	79	73	2E	6D	75	69	.dll.exe.sys.mui
1111D940	2E	74	6D	70	2E	6C	6E	6B	2E	63	6F	6E	66	69	67	00	.tmp.lnk.config.
1111D950	2E	6D	61	6E	69	66	65	73	74	00	00	00	2E	74	6C	62	.manifest....tlb
1111D960	2E	6F	6C	62	2E	62	6C	66	2E	69	63	6F	2E	62	61	74	.olb.blf.ico.bat

Fig. 13 - Extensions of the file to be encrypted (part)

SNAKE does not handle multiple-process encryption, such as MegaCortex and LockerGoga, but uses a single process to encrypt files.

SNAKE (nmon.exe) is performing file encryption.

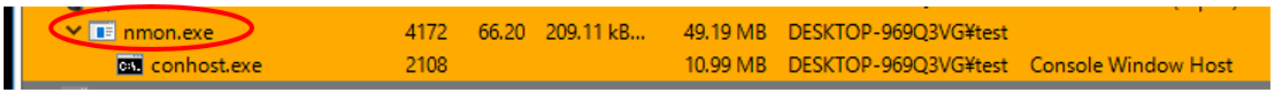


Fig. 14 - SNAKE encrypting files

During the encryption process, Windows system folders (such as system-based files and Windows folders) are excluded from encryption, and Windows remains operational even when infected with SNAKE. However, as described above, various services and processes are stopped, and many programs including EXE files are also encrypted, so that the system of the terminal that has been infected to this specimen will be totally unstable.

List of excluded files from encryption (partial)

```
11057F04 1111DA00 "bootnxt"
11057F08 1111D988 "bootmgr"
11057F0C 1111A920 "usrclass.dat.log2"
11057F10 1111A900 "usrclass.dat.log1"
11057F14 1111D9F0 "usrclass.dat"
11057F18 1111D9E0 "ntuser.dat.log2"
11057F1C 1111D9D0 "ntuser.dat.log1"
11057F20 1111D9C0 "ntuser.ini"
11057F24 1111D9B0 "ntuser.dat"
11057F28 1111D9A0 "iconcache.db"
11057F2C 1111D990 "desktop.ini"
11057F30 1111D984 ".ps1bootmgr"
11057F34 1111D980 ".cmd.ps1bootmgr"
11057F38 1111D96C ".bat.regtrans-ms"
11057F3C 1111A8E0 ".settingcontent-ms"
11057F40 1111A8C0 ".devicemetadata-ms"
11057F44 1111D970 ".regtrans-ms"
11057F48 1111DB50 "thumbs.db"
11057F4C 1111DB40 "ntuser.ini"
11057F50 1111DB30 "ntuser.dat.log"
11057F54 1111DB20 "ntuser.dat"
11057F58 1111DB10 "iconcache.db"
11057F5C 1111DB00 "ctfmon.exe"
11057F60 1111DAF0 "desktop.ini"
11057F64 1111DAE0 "bootsect.bak"
11057F68 1111DAD0 "bootfont.bin"
11057F6C 1111DAC0 "boot.ini"
11057F70 1111DAB0 "NTDETECT.COM"
11057F74 1111DAA9 "ntldr"
```

Fig. 15 - Files that SNAKE excludes from encryption (portion)

Common ransomware often changes or adds encrypted file extensions to a particular string (for example, .locked), but all files encrypted by SNAKE are appended with a random string at the end of the original extension.

Before and after encryption by SNAKE.

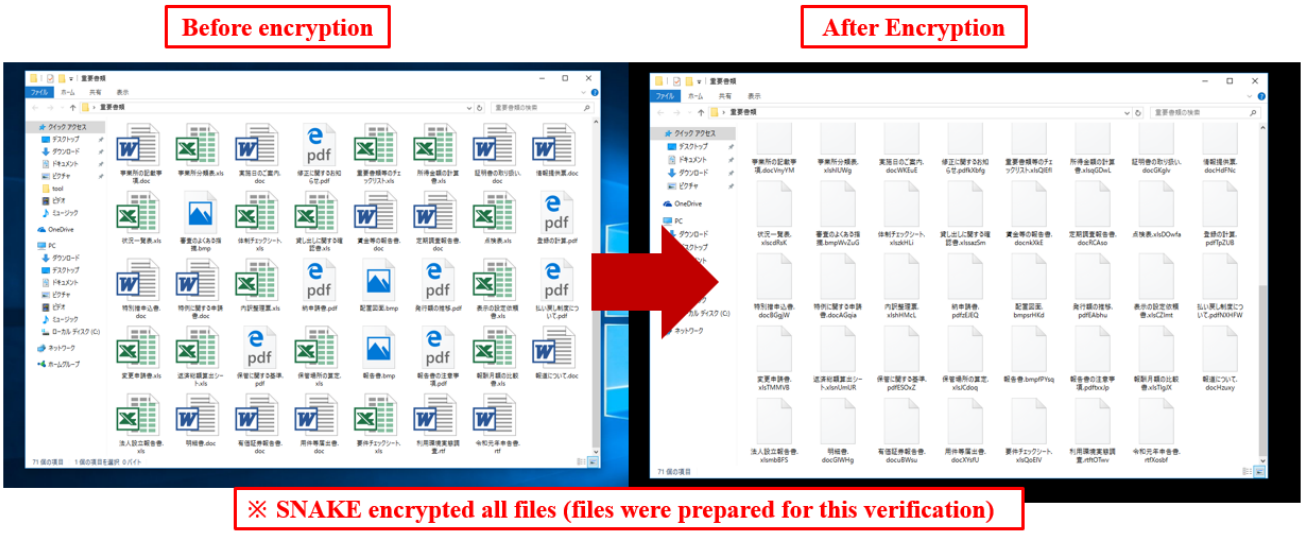
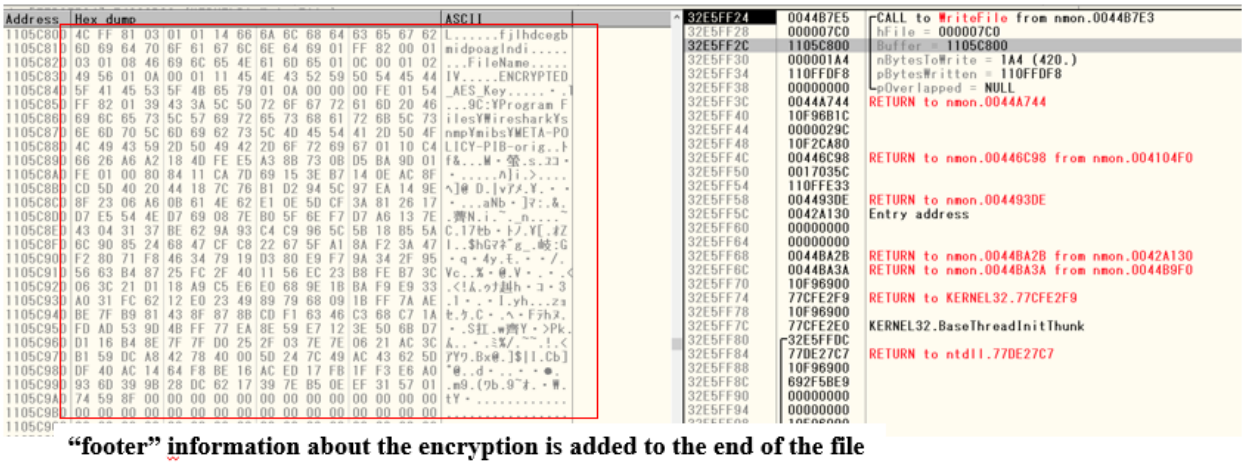


Fig. 16 - Comparing before and after SNAKE encryption

A footer is added to the end of SNAKE encrypted file, including the original file name and the AES key encrypted with the RSA public key, and a "EKANS" marker is added to the last 5 bytes to indicate that the file was encrypted with SNAKE.

Contents in RAM Memory when encrypting a file.
First, SNAKE encrypts entire file, followed by adding "footer" and "marker" to the end of the file.



"footer" information about the encryption is added to the end of the file

Finally, "marker" which is a 5 byte string (EKANS), is added to the end of the file

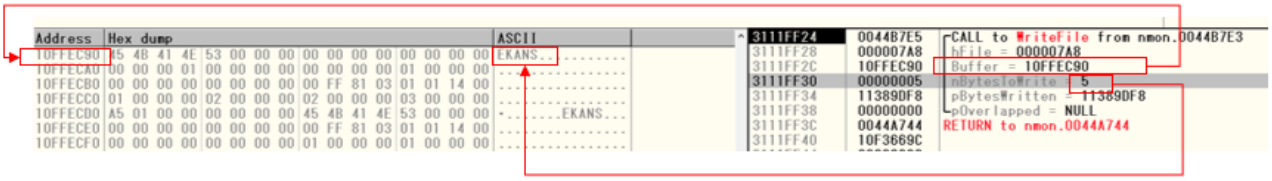


Fig. 17 - SNAKE added data to the end of an encrypted file

Previous process added the "EKANS" marker at the end of the file

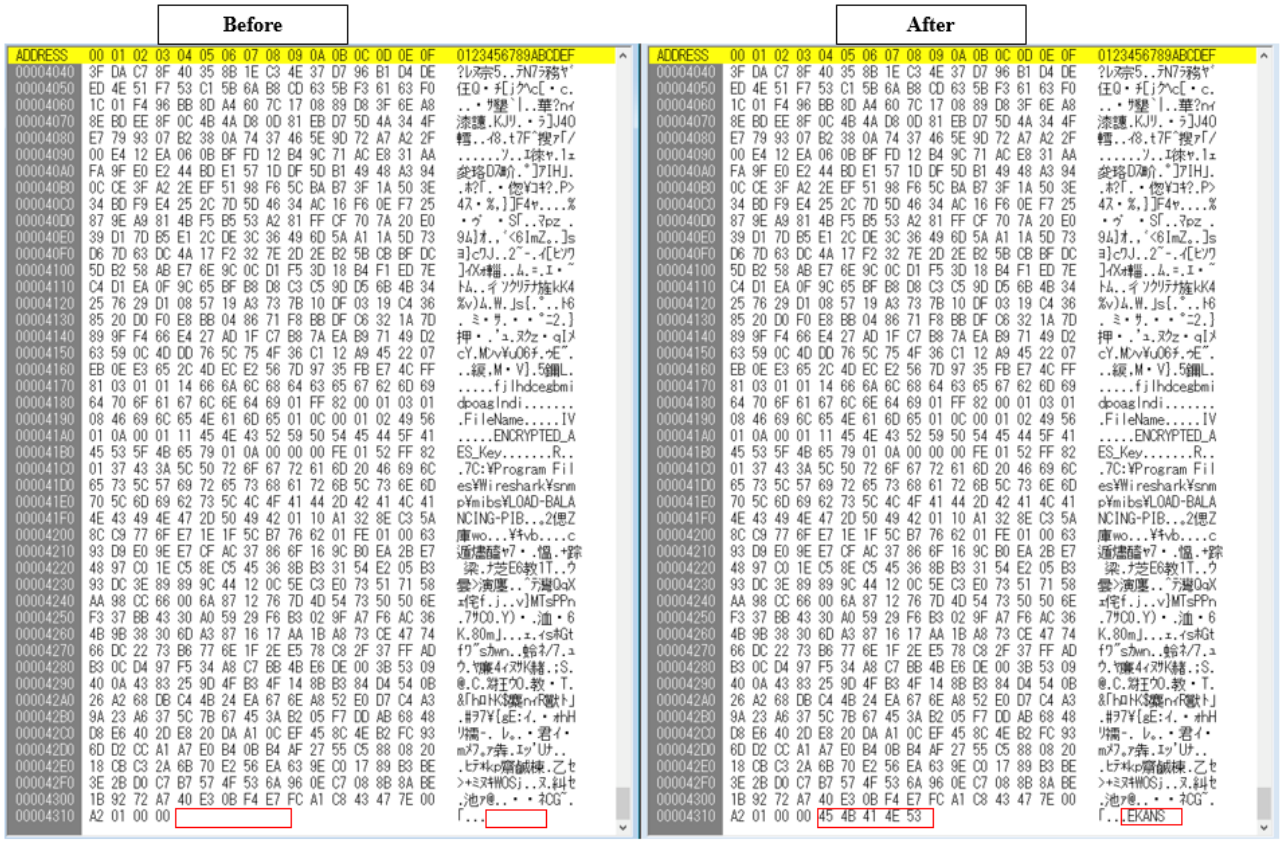


Fig. 18 - Comparing before and after SNAKE. "EKANS" marker added at the end of the encrypted file

Encrypted file is shown in below, and you can determine if it is a SNAKE encrypted file by looking at the last 5 bytes of the file.

At the end of SNAKE encrypted file,
A “footer” containing AES encryption key with RSA public key and the original file name,
A "marker" is added to indicate that it has been encrypted by SNAKE.

※ The end of an image file encrypted by SNAKE is displayed in a binary editor.

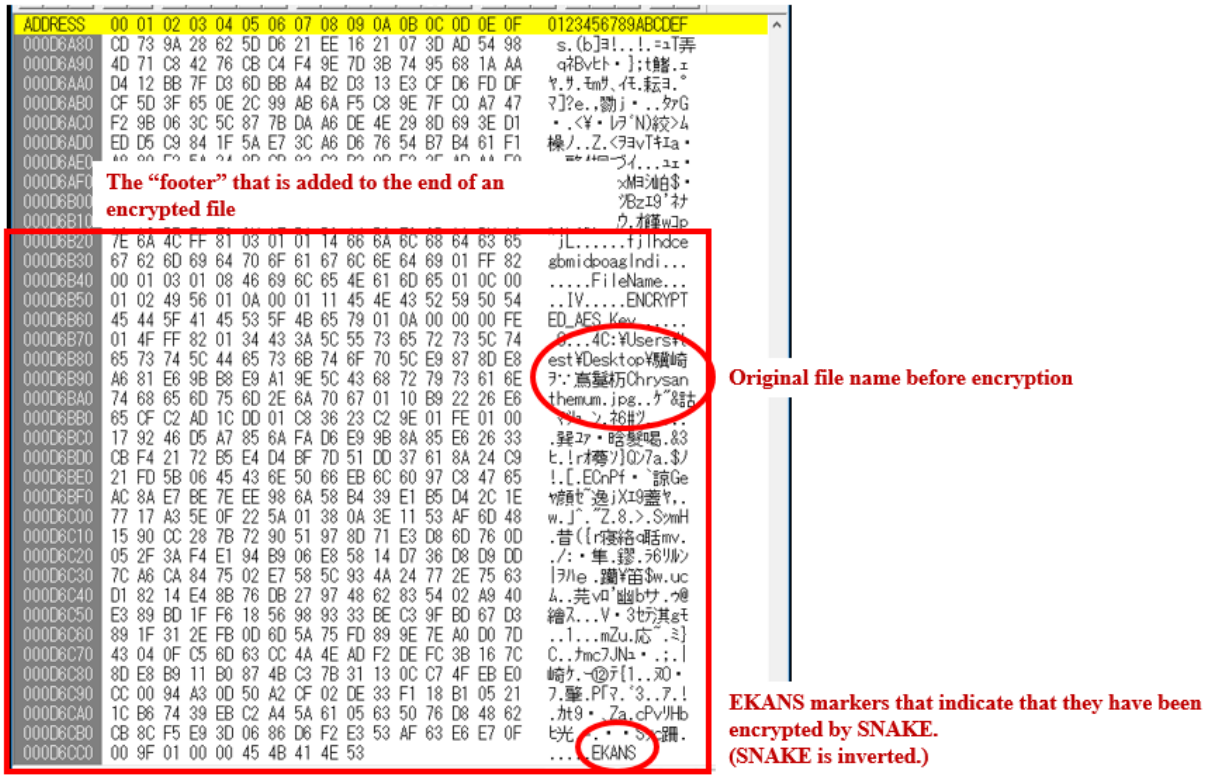


Fig. 19 - Structure of end of file, encrypted by SNAKE

The RSA public key included in the ransomware is shown below.

```
----- BEGIN RSA PUBLIC KEY-----
MIIBCgKCAQEAt1GCKUHXITsiWc1d8V0vo1Y9Jm18RDZEmMS6OkHI7pZT0RHAThlR
BFITZY9bXrl6RFdUwmIX0WYn5ZqIlhLAEe1cqd8RpJ/KK2OeiTn0CJ1CGmOOJvfm
5rFa8whVAU9cnh/iVCcf+aEHJVcHhzB5tTtiT3lBIdfzaLL6GR5EmytbQ3V3O1Uk
Y4FCKxYOMVoPzPtRG3vo3688uUWpZIKBV7e6dht/mAhuCEIlRGcdpAEf6f4zUUYf
DtHcDafMVEA4Sy/DDsd76wAyBIM0XKLv1+vH476TN1K1tIRBrR98QFI5mlXkgqz6
h+Wpb/5KYWWvG0ZLZcu6eWOCGmLEmorvWQIDAQAB
----- END RSA PUBLIC KEY-----
```

The file operation during encryption is implemented in the following flow. Even if it is developed in GO language, from Windows API point of view, it is the same file manipulation flow found in the normal ransomware, that is eventually utilizing WriteFile and ReadFile.

The file operation when encrypting files is implemented in the following flow.
Even when developed in GO language, from Windows API point of view,
common file manipulation by WriteFile and ReadFile is eventually called.

※ The function called by malware is recorded in chronological order from top to bottom.

nmon.exe	NtWaitForSingleObject (0xffffffff, FALSE, 0x32a2fecc)	STATUS_TIMEOUT
nmon.exe	GetConsoleMode (0x00000324, 0x12a7fe7c)	FALSE
nmon.exe	GetFileType (0x00000324)	FILE_TYPE_DISK
nmon.exe	SetFilePointerEx (0x00000324, { u = { LowPart = 4294967291, HighPart = -1 }, QuadPart = -5 }, 0x12a7fe44, FILE_END)	TRUE
nmon.exe	ReadFile (0x00000324, 0x12b198e4, 5, 0x12a7fe2c, NULL) ← Check if the last 5 bytes of the file are EKANS (check if encrypted)	TRUE
nmon.exe	GetFileType (0x00000324)	FILE_TYPE_DISK
nmon.exe	SetFilePointerEx (0x00000324, { u = { LowPart = 0, HighPart = 0 }, QuadPart = 0 }, 0x12a7fe44, FILE_BEGIN)	TRUE
nmon.exe	CryptGenRandom (0x0085bff8, 16, 0x12b198f0)	TRUE
nmon.exe	CryptGenRandom (0x0085bff8, 32, 0x12b7d280)	TRUE
nmon.exe	ReadFile (0x00000324, 0x12bf6000, 102400, 0x12a7fdf8, NULL) ← Read the file to be encrypted	TRUE
nmon.exe	GetFileType (0x00000324)	FILE_TYPE_DISK
nmon.exe	SetFilePointerEx (0x00000324, { u = { LowPart = 0, HighPart = 0 }, QuadPart = 0 }, 0x12a7fde0, FILE_CURRENT)	TRUE
nmon.exe	WriteFile (0x00000324, 0x12c10000, 10210, 0x12a7fdf8, 0x12a7fdfc) ← Write encrypted data to a file	TRUE
nmon.exe	GetFileType (0x00000324)	FILE_TYPE_DISK
nmon.exe	SetFilePointerEx (0x00000324, { u = { LowPart = 10210, HighPart = 0 }, QuadPart = 10210 }, 0x12a7fde0, FILE_BEGIN)	TRUE
nmon.exe	ReadFile (0x00000324, 0x12bf6000, 102400, 0x12a7fdf8, NULL)	TRUE
nmon.exe	CryptGenRandom (0x0085bff8, 20, 0x12b38901)	TRUE
nmon.exe	GetFileType (0x00000324)	FILE_TYPE_DISK
nmon.exe	SetFilePointerEx (0x00000324, { u = { LowPart = 0, HighPart = 0 }, QuadPart = 0 }, 0x12a7fe10, FILE_END)	TRUE
nmon.exe	WriteFile (0x00000324, 0x12c1a600, 428, 0x12a7fdf8, NULL) ← Write footer to end of file	TRUE
nmon.exe	WriteFile (0x00000324, 0x12b19940, 4, 0x12a7fdf8, NULL)	TRUE
nmon.exe	WriteFile (0x00000324, 0x12b19948, 5, 0x12a7fdf8, NULL) ← Writing EKANS Markers to the End of a File	TRUE
nmon.exe	CloseHandle (0x00000324)	TRUE
nmon.exe	CreateFileW ("C:\Program Files\Common Files\VMware\Drivers\mouse\Win8\vmusbmouse.cat", GENERIC_READ | GENERIC_WRITE, FILE_SHARE_READ | FILE_SHAR...	0x00000324
nmon.exe	SetEvent (0x0000019c)	TRUE
nmon.exe	WaitForSingleObject (0x000002cc, INFINITE)	WAIT_OBJECT_0
nmon.exe	NtWaitForSingleObject (0xffffffff, FALSE, 0x32a2fecc)	STATUS_TIMEOUT
nmon.exe	FindNextFileW (0x0084ecd0, 0x12949a0c)	TRUE

Fig. 20 - Details of file operations when encrypting a single file

Note that SNAKE does not take the recent encryption trend found in the other ransomware (encrypt a file, then change its file extension, and repeat again and again on other files). SNAKE first encrypts all files (without changing file extensions). Once all encryptions are done, it changes all file extensions at a time. For the hacker, there are benefits in this method. Since the file extension is not changed while the ransomware is being encrypted, it is difficult for the user to notice that the file is encrypted halfway, because the behavior is similar to the normal renaming process. As a result, it is possible to escape without being detected by the behavior detection (encrypt a file, then change its file extension, and repeat again and again on other files) .

After all files have been encrypted, rename them all at once with "MoveFileEx"
adding random strings to its file extension.
(Using this technique, it may be able to escape from being detected, because this process is commonly used)

nmon.exe	MoveFileExW ("C:\Users\hanako\AppData\Local\Microsoft\Internet Explorer\CacheStorage\.inUse", "C:\Users\hanako\AppData\Local\Microsoft\Internet Explorer\CacheStorage\.inUseHlqzC", MOVEFILE_REPLACE_EXISTING)	TRUE
nmon.exe	MoveFileExW ("C:\Users\hanako\AppData\Local\Microsoft\Internet Explorer\CacheStorage\edb.chk", "C:\Users\hanako\AppData\Local\Microsoft\Internet Explorer\CacheStorage\edb.chkfyvuj", MOVEFILE_REPLACE_EXISTING)	TRUE
nmon.exe	MoveFileExW ("C:\Users\hanako\AppData\Local\Microsoft\Internet Explorer\CacheStorage\edb.log", "C:\Users\hanako\AppData\Local\Microsoft\Internet Explorer\CacheStorage\edb.logWCgwN", MOVEFILE_REPLACE_EXISTING)	TRUE
nmon.exe	MoveFileExW ("C:\Users\hanako\AppData\Local\Microsoft\Internet Explorer\CacheStorage\edbres00001.jrs", "C:\Users\hanako\AppData\Local\Microsoft\Internet Explorer\CacheStorage\edbres00001.jrsRxDAF", MOVEFILE_REPLACE_EXISTING)	TRUE
nmon.exe	MoveFileExW ("C:\Users\hanako\AppData\Local\Microsoft\Internet Explorer\CacheStorage\edbres00002.jrs", "C:\Users\hanako\AppData\Local\Microsoft\Internet Explorer\CacheStorage\edbres00002.jrsUGRmS", MOVEFILE_REPLACE_EXISTING)	TRUE
nmon.exe	MoveFileExW ("C:\Users\hanako\AppData\Local\Microsoft\Internet Explorer\CacheStorage\edbtmp.log", "C:\Users\hanako\AppData\Local\Microsoft\Internet Explorer\CacheStorage\edbtmp.logmyFJy", MOVEFILE_REPLACE_EXISTING)	TRUE
nmon.exe	MoveFileExW ("C:\Users\hanako\AppData\Local\Microsoft\Internet Explorer\ie4uinit-ClearIconCache.log", "C:\Users\hanako\AppData\Local\Microsoft\Internet Explorer\ie4uinit-ClearIconCache.loggJUk", MOVEFILE_REPLACE_EXISTING)	TRUE
nmon.exe	MoveFileExW ("C:\Users\hanako\AppData\Local\Microsoft\Internet Explorer\ie4uinit-UserConfig.log", "C:\Users\hanako\AppData\Local\Microsoft\Internet Explorer\ie4uinit-UserConfig.logpabJt", MOVEFILE_REPLACE_EXISTING)	TRUE
nmon.exe	MoveFileExW ("C:\Users\hanako\AppData\Local\Microsoft\Internet Explorer\IECompatData\iecompatdata.xml", "C:\Users\hanako\AppData\Local\Microsoft\Internet Explorer\IECompatData\iecompatdata.xmlkfiWy", MOVEFILE_REPLACE_EXISTING)	TRUE
nmon.exe	MoveFileExW ("C:\Users\hanako\AppData\Local\Microsoft\Internet Explorer\MSIMGSIZ.DAT", "C:\Users\hanako\AppData\Local\Microsoft\Internet Explorer\MSIMGSIZ.DATMFSmk", MOVEFILE_REPLACE_EXISTING)	TRUE
nmon.exe	MoveFileExW ("C:\Users\hanako\AppData\Local\Microsoft\Media Player\Sync Playlists\ja-JP\00039367\01_Music_auto_rated_at_5_stars.wpl", "C:\Users\hanako\AppData\Local\Microsoft\Media Player\Sync Playlists\ja-JP\00039367\01_Music_auto...	TRUE
nmon.exe	MoveFileExW ("C:\Users\hanako\AppData\Local\Microsoft\Media Player\Sync Playlists\ja-JP\00039367\02_Music_added_in_the_last_month.wpl", "C:\Users\hanako\AppData\Local\Microsoft\Media Player\Sync Playlists\ja-JP\00039367\02_Music_a...	TRUE

Fig. 21 - Renaming all files at once after they have been encrypted

When all encryption are completed, SNAKE disables all firewalls with the following command:

```
Netsh advfirewall set allprofiles state off
```

Once encryption is completed, SNAKE disables all firewall settings

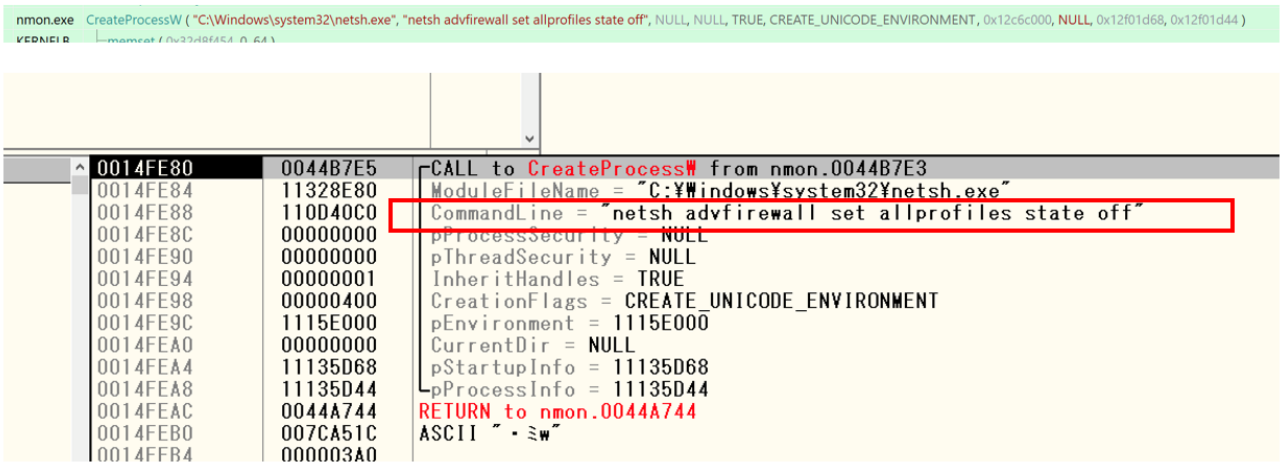


Fig. 22 - Disabling the firewall after the encryption process

In other words, SNAKE performs a series of tasks, such as blocking network communications from being sent to and from Windows firewall prior to encrypting files, preventing recovery activities and monitoring across the network during file encryption, and after file encryption is complete, breaking those blocks.

5. Special behavior in domain controllers

A unique feature of this specimen is that it specially works when the work environment is a domain controller.

Typical ransomware basically creates threatening letter after encryption. However, this SNAKE does not create any threatening letter on infected user PC or server, although it does encrypt files.

However, if it determines that the infected environment is a domain controller, it does not encrypt any files. Instead, it creates threatening letters on Windows desktop of Public User and under the root directory of C drive (C:¥).

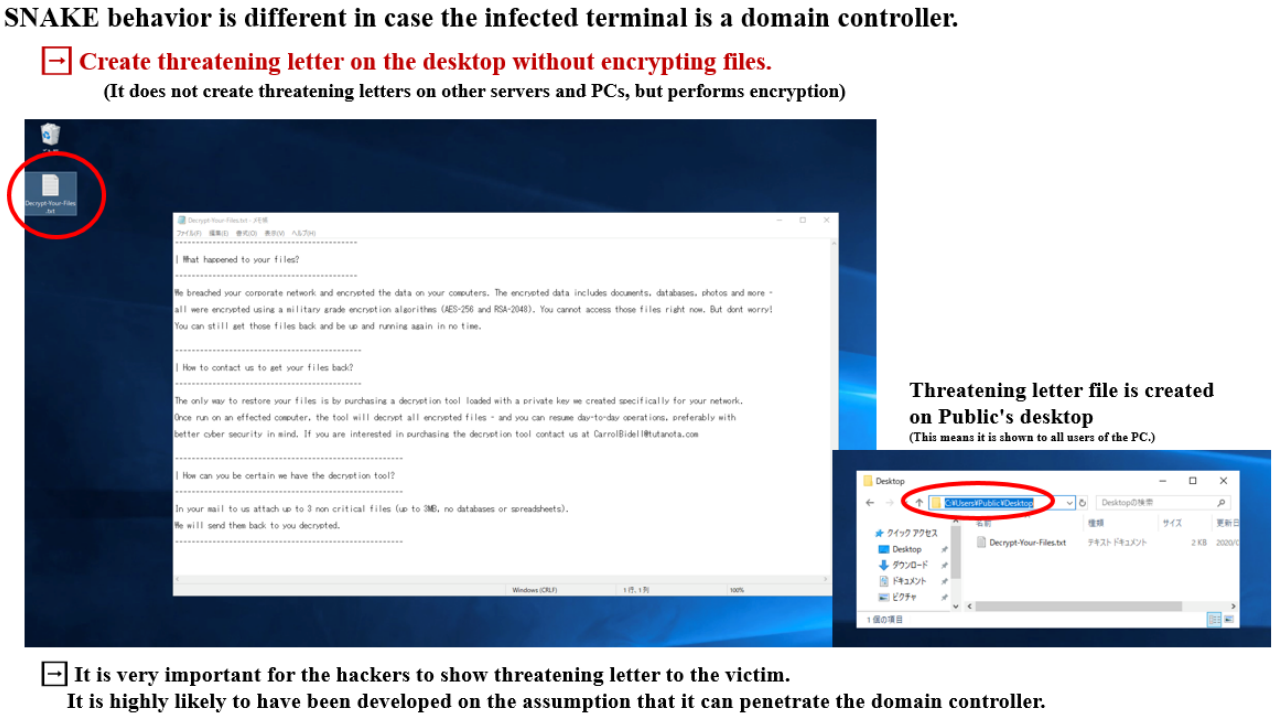


Fig. 23 - Create a threatening letter only on domain controller

6. How to detect domain controllers

In this section, I will explain more in detail about the mechanism of identifying domain controllers.

First, SNAKE uses WMI queries to refer the domain role value.

Checking Domain Roles using WMI queries

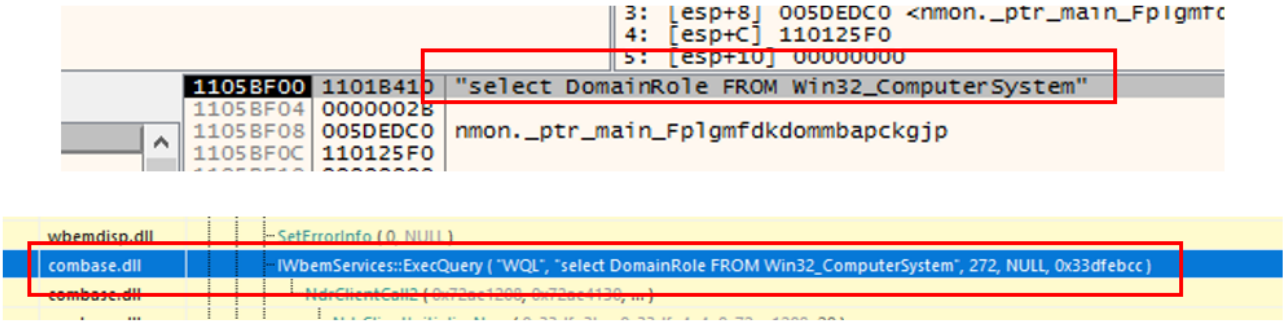


Fig. 24 - Checking the domain role value with WMI Query

The domain role value is defined by the following number, and the domain role value for the domain controller is "4" or "5".

In the domain controller, the domain role value becomes "4" or "5".

Table 9.4 DomainRole Property Values

Value	Description
0	Stand-alone workstation (the computer is not a member of a domain)
1	Member workstation
2	Stand-alone server (the computer is not a member of a domain)
3	Member server
4	Backup domain controller
5	Primary domain controller

Numerical value indicating the type of domain role (quoted from the Microsoft site)
(https://docs.microsoft.com/en-us/previous-versions/tn-archive/ee198796(v=technet.10)?redirectedfrom=MSDN)

Fig. 25 – Definition of the domain role value

SNAKE checks whether the domain role value is less than or equal to 3, and encrypts the file if it is less than or equal to 3, but does not create threatening letter. On the other hand, if it is not less than 3, it does not encrypt the file and creates threatening letters.
The following example compares the behavior of SNAKE in a non-domain controller server (StandaloneServer) and domain controller (Primary domain controller) environment.

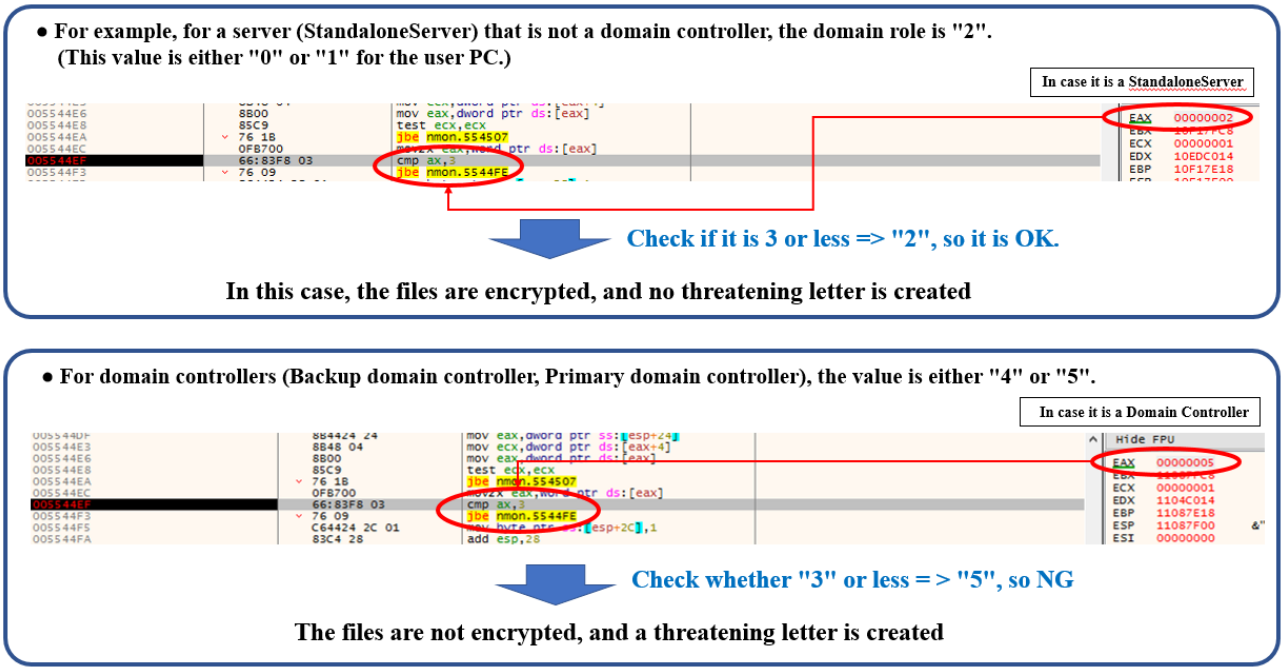


Fig. 26 – How to identify domain role

The following is the process by which SNAKE branches its behavior depending on the domain role value.

The process branches according to the domain role

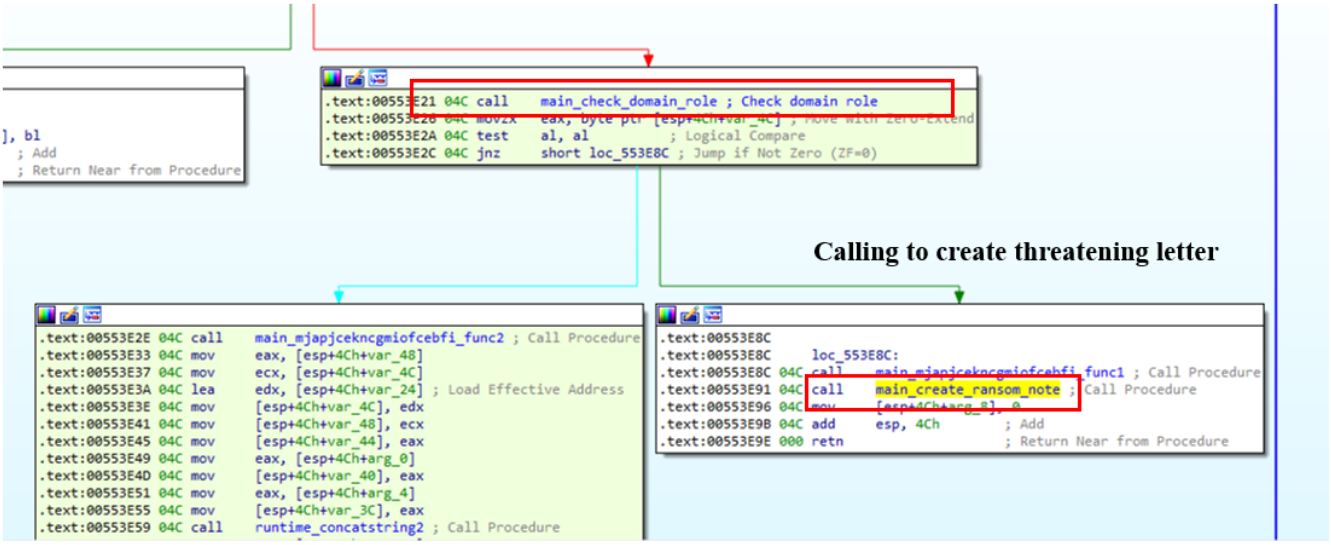


Fig. 27 - Branch by domain role value

If SNAKE identifies it is a domain controller, then it obtains the path to Windows desktop of Public User.

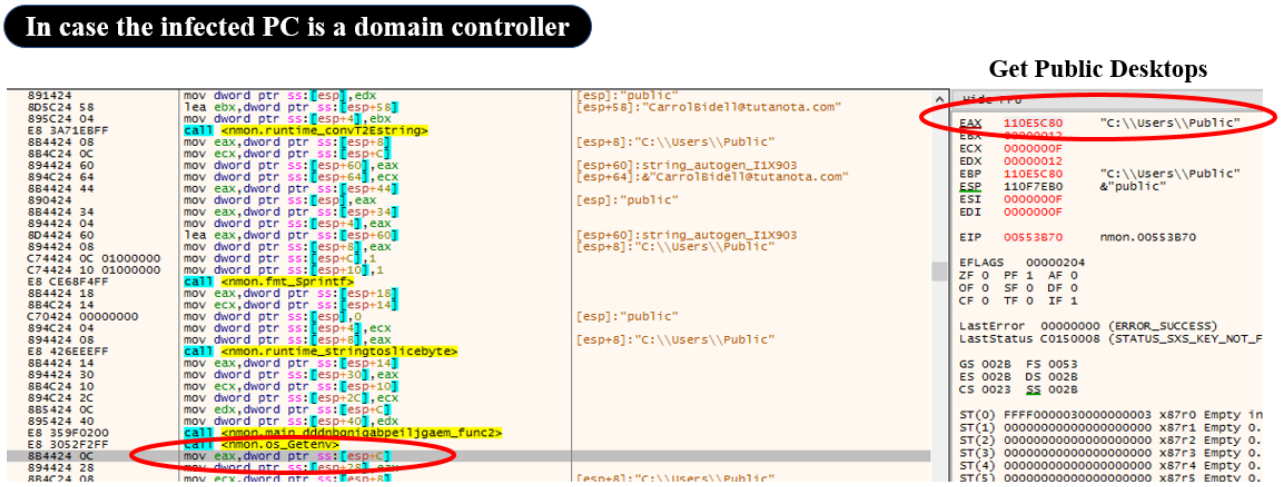
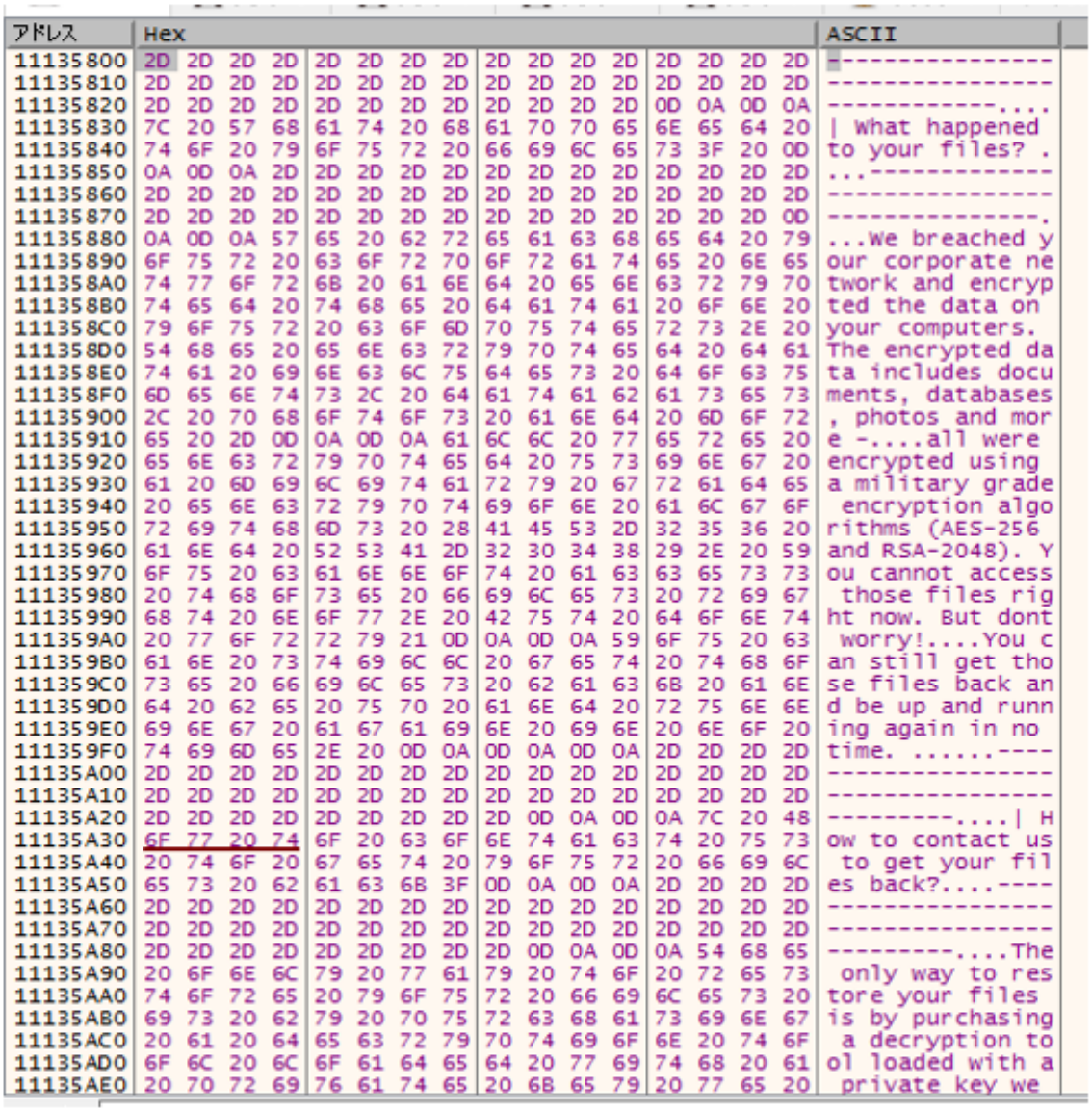


Fig. 28 - Get Public desktop path

Then, layouts the threatening letter into memory as follows, and creates threatening letters on Windows desktop of Public User and under the root directory of C drive (C:¥).

In case the infected PC is a domain controller

SNAKE puts threatening letter into RAM memory (which is then created on Public desktop)



アドレス	Hex																ASCII
11135800	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	----------
11135810	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	----------
11135820	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	0D	0A	0D	0A	----------....
11135830	7C	20	57	68	61	74	20	68	61	70	70	65	6E	65	64	20	| What happened
11135840	74	6F	20	79	6F	75	72	20	66	69	6C	65	73	3F	20	0D	to your files? .
11135850	0A	0D	0A	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	...----------
11135860	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	----------
11135870	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	0D	----------.
11135880	0A	0D	0A	57	65	20	62	72	65	61	63	68	65	64	20	79	...We breached y
11135890	6F	75	72	20	63	6F	72	70	6F	72	61	74	65	20	6E	65	our corporate ne
111358A0	74	77	6F	72	6B	20	61	6E	64	20	65	6E	63	72	79	70	twork and encryp
111358B0	74	65	64	20	74	68	65	20	64	61	74	61	20	6F	6E	20	ted the data on
111358C0	79	6F	75	72	20	63	6F	6D	70	75	74	65	72	73	2E	20	your computers.
111358D0	54	68	65	20	65	6E	63	72	79	70	74	65	64	20	64	61	The encrypted da
111358E0	74	61	20	69	6E	63	6C	75	64	65	73	20	64	6F	63	75	ta includes docu
111358F0	6D	65	6E	74	73	2C	20	64	61	74	61	62	61	73	65	73	ments, databases
11135900	2C	20	70	68	6F	74	6F	73	20	61	6E	64	20	6D	6F	72	, photos and mor
11135910	65	20	2D	0D	0A	0D	0A	61	6C	6C	20	77	65	72	65	20	e -....all were
11135920	65	6E	63	72	79	70	74	65	64	20	75	73	69	6E	67	20	encrypted using
11135930	61	20	6D	69	6C	69	74	61	72	79	20	67	72	61	64	65	a military grade
11135940	20	65	6E	63	72	79	70	74	69	6F	6E	20	61	6C	67	6F	encryption algo
11135950	72	69	74	68	6D	73	20	28	41	45	53	2D	32	35	36	20	rithms (AES-256
11135960	61	6E	64	20	52	53	41	2D	32	30	34	38	29	2E	20	59	and RSA-2048). Y
11135970	6F	75	20	63	61	6E	6E	6F	74	20	61	63	63	65	73	73	ou cannot access
11135980	20	74	68	6F	73	65	20	66	69	6C	65	73	20	72	69	67	those files rig
11135990	68	74	20	6E	6F	77	2E	20	42	75	74	20	64	6F	6E	74	ht now. But dont
111359A0	20	77	6F	72	72	79	21	0D	0A	0D	0A	59	6F	75	20	63	worry!....You c
111359B0	61	6E	20	73	74	69	6C	6C	20	67	65	74	20	74	68	6F	an still get tho
111359C0	73	65	20	66	69	6C	65	73	20	62	61	63	6B	20	61	6E	se files back an
111359D0	64	20	62	65	20	75	70	20	61	6E	64	20	72	75	6E	6E	d be up and runn
111359E0	69	6E	67	20	61	67	61	69	6E	20	69	6E	20	6E	6F	20	ing again in no
111359F0	74	69	6D	65	2E	20	0D	0A	0D	0A	0D	0A	2D	2D	2D	2D	time. ......----
11135A00	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	----------
11135A10	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	----------
11135A20	2D	2D	2D	2D	2D	2D	2D	2D	2D	0D	0A	0D	0A	7C	20	48	---------....| H
11135A30	6F	77	20	74	6F	20	63	6F	6E	74	61	63	74	20	75	73	ow to contact us
11135A40	20	74	6F	20	67	65	74	20	79	6F	75	72	20	66	69	6C	to get your fil
11135A50	65	73	20	62	61	63	6B	3F	0D	0A	0D	0A	2D	2D	2D	2D	es back?....----
11135A60	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	----------
11135A70	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	----------
11135A80	2D	2D	2D	2D	2D	2D	2D	2D	2D	0D	0A	0D	0A	54	68	65	---------....The
11135A90	20	6F	6E	6C	79	20	77	61	79	20	74	6F	20	72	65	73	only way to res
11135AA0	74	6F	72	65	20	79	6F	75	72	20	66	69	6C	65	73	20	tore your files
11135AB0	69	73	20	62	79	20	70	75	72	63	68	61	73	69	6E	67	is by purchasing
11135AC0	20	61	20	64	65	63	72	79	70	74	69	6F	6E	20	74	6F	a decryption to
11135AD0	6F	6C	20	6C	6F	61	64	65	64	20	77	69	74	68	20	61	ol loaded with a
11135AE0	20	70	72	69	76	61	74	65	20	6B	65	79	20	77	65	20	private key we

Fig. 29 – Threatening letter layout in SNAKE memory

Other behaviors, such as horizontal expansion and file theft, have not been confirmed at the present time.

Since Windows firewall operation by netsh requires administrator's privileges/system privileges, and there is no behavior requiring privilege promotion when SNAKE is executed with user privileges, this specimen may have been developed under the assumption that it is executed with administrator's privileges/system privileges from the beginning.

Also, because of nature of a ransomware, one of the most important goal is to show the target a threatening letter with the contact information, but this specimen does not create a threatening letter if it is a user terminal or general server, so the ransomware's purpose (getting a ransom money) cannot be achieved without doing so (unless it is a wiper with destructive purpose).
Considering this, the behavior of creating threatening letter only on domain controllers and presenting threatening letter to system administrators suggests that they have been developed under the assumption that they can intrude into domain controllers.

In view of this situation, one of the possibilities for the infection route of this specimen to general user terminals is the distribution of this specimen to each terminal via the domain controller with the supervisor authority/system authority, rather than by a user's double click.

The results of the analysis described above show that the following symptom occurs on the terminal that has been infected to this specimen.

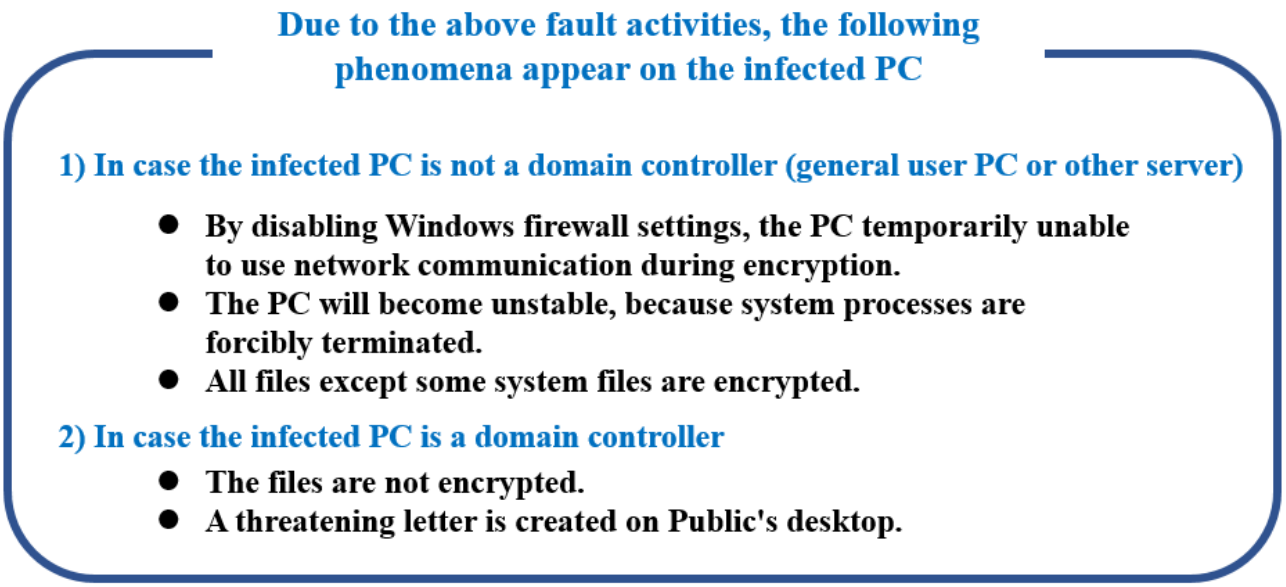


Fig. 30 - Symptoms of the infected PC by this SNAKE

7. Summary

As mentioned at the beginning, we do not have any information other than this specimen, so there is no evidence that this specimen was actually used in Honda's cyber attack. The fact is that, the specimen analyzed this time has been developed so that it works only on terminals under the environment that can resolve (MDS[.]HONDA[.]COM) to specific IP address (170 [.]108[.]71[.]15).

As mentioned in previous blog posts, targeted ransomware in recent years tends to be sent after tuning the behavior according to the attack target organization. As a general intrusion route for targeted ransomware, intrusion routes aimed at RDP, VNC, etc. may be targeted. Therefore, I recommend that you re-check RDP and other services have not been exposed to the Internet.

8. About us

MBSD (Mitsui Bussan Secure Directions, Inc.) is the Japanese leading security company in managed security services, vulnerability assessment and testing, GRC (Governance, Risk, Compliance) consulting, incident response and handling, digital forensics, and secure programming training services. The MBSD services are provided by its personnel including the leading security experts in the field of secure programming, application security, penetration testing and threat analysis who have in-depth knowledge and understanding of attackers' methodologies. MBSD is working for the Internet infrastructure companies, cyber commerce and media giants, financial institutes, global enterprise, and government agencies in Japan to support their strategies against rapidly increasing threats from cyber space.

Company Contact Information:
Mitsui Bussan Secure Directions, Inc.
Yusen Suitengumae Building 6F, 1-14-8, Nihonbashi Ningyo Cho, Chuo-ku, Tokyo, 103-0013, Japan
+81 3 5649 1961 | http://www.mbsd.jp/