

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. Introduction3
- 2. Specimen3
- 3. Activities in environments where name resolution is available9
- 4. Encryption process12
- 5. Special behavior in domain controllers20
- 6. How to detect domain controllers21
- 7. Summary26
- 8. About us26

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”

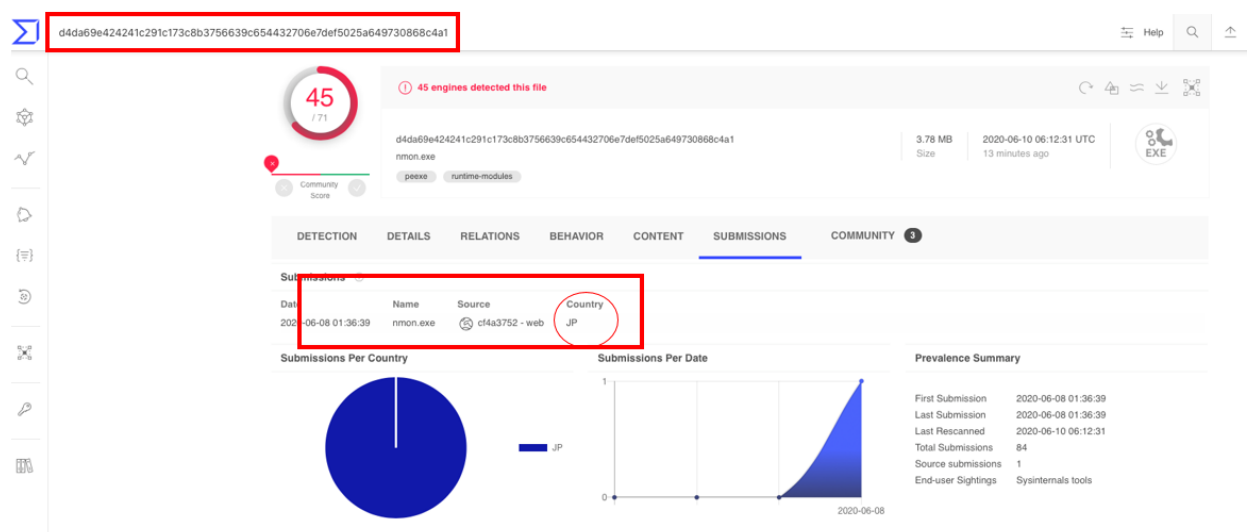


Fig. 1 - Uploaded to VirusTotal from Japan

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.

※ Information on the development environment that can be extracted from the binaries of the malware body

```

.text:007B0FA4 0000001F C C:/Go/src/crypto/cipher/gcm.go
.text:007B0FC4 0000001F C C:/Go/src/crypto/cipher/xor.go
.text:007B0FE4 00000022 C C:/Go/src/crypto/cipher/cipher.go
.text:007B1008 0000001E C C:/Go/src/crypto/cipher/ctr.go
.text:007B1028 0000009F C ;/Users/Admin3/go/src/agfkbpbpbmhpjgfgmf/ocldgdobgbcgababki/pdllhaickabprhmjmcda/apbnkncjkhoefmldne/cnfadkapheieagpbjk/bbcoocpkdokenbefinbhc/service.go
.text:007B10C8 0000009C C C:/Users/Admin3/go/src/agfkbpbpbmhpjgfgmf/ocldgdobgbcgababki/pdllhaickabprhmjmcda/apbnkncjkhoefmldne/cnfadkapheieagpbjk/bbcoocpkdokenbefinbhc/mgr.go
.text:007B1165 0000008B C C:/Users/Admin3/go/src/agfkbpbpbmhpjgfgmf/ocldgdobgbcgababki/pdllhaickabprhmjmcda/apbnkncjkhoefmldne/cnfadkapheieagpbjk/service.go
.text:007B11F1 0000007C C C:/Users/Admin3/go/src/agfkbpbpbmhpjgfgmf/ocldgdobgbcgababki/pdllhaickabprhmjmcda/apbnkncjkhoefmldne/types_windows.go
.text:007B126E 0000007F C C:/Users/Admin3/go/src/agfkbpbpbmhpjgfgmf/ocldgdobgbcgababki/pdllhaickabprhmjmcda/apbnkncjkhoefmldne/zsyscall_windows.go
.text:007B12EE 0000007E C C:/Users/Admin3/go/src/agfkbpbpbmhpjgfgmf/ocldgdobgbcgababki/pdllhaickabprhmjmcda/apbnkncjkhoefmldne/syscall_windows.go
.text:007B136D 00000076 C C:/Users/Admin3/go/src/agfkbpbpbmhpjgfgmf/ocldgdobgbcgababki/pdllhaickabprhmjmcda/apbnkncjkhoefmldne/syscall.go
.text:007B13E4 00000072 C C:/Users/Admin3/go/src/agfkbpbpbmhpjgfgmf/ocldgdobgbcgababki/pdllhaickabprhmjmcda/apbnkncjkhoefmldne/str.go
.text:007B1458 0000007E C ;/Users/Admin3/go/src/agfkbpbpbmhpjgfgmf/ocldgdobgbcgababki/pdllhaickabprhmjmcda/apbnkncjkhoefmldne/security_windows.go
.text:007B14D8 0000007A C ;/Users/Admin3/go/src/agfkbpbpbmhpjgfgmf/ocldgdobgbcgababki/pdllhaickabprhmjmcda/apbnkncjkhoefmldne/exec_windows.go
.text:007B1554 00000079 C ;/Users/Admin3/go/src/agfkbpbpbmhpjgfgmf/ocldgdobgbcgababki/pdllhaickabprhmjmcda/apbnkncjkhoefmldne/dll_windows.go
.text:007B15CE 00000016 C C:/Go/src/net/hook.go
.text:007B15E5 0000001E C C:/Go/src/net/sock_windows.go
.text:007B1604 00000017 C C:/Go/src/net/parse.go
.text:007B161C 00000015 C C:/Go/src/net/mac.go
.text:007B1632 00000020 C C:/Go/src/net/lookup_windows.go

```

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.)

HEADERS INFO			
Address of Entry Point: 0044B7B0		Real Image Checksum: 003D47DAh	
Field Name	Data Value	Description	
Machine	014Ch	i386	
Number of Sections	0004h		
Time Date Stamp	00000000h	01/01/1970 00:00:00	
Pointer to Symbol Table	003C8200h		
Number of Symbols	00000000h		
Size of Optional Header	00E0h		
Characteristics	0303h		
Magic	010Bh	PE32	
Linker Version	0003h	3.0	
Size of Code	003B1E00h		
Size of Initialized Data	00015C00h		
Size of Uninitialized Data	00000000h		
Address of Entry Point	0044B7B0h		
Base of Code	00001000h		
Base of Data	003B3000h		
Image Base	00400000h		
Field Name	Data Value	Description	
Section Alignment	00001000h		
File Alignment	00000200h		
Operating System Version	00000004h	4.0	
Image Version	00000001h	1.0	
Subsystem Version	00000004h	4.0	
Win32 Version Value	00000000h	Reserved	
Size of Image	003E0000h	4063232 bytes	
Size of Headers	00000400h		
Checksum	00000000h		
Subsystem	0003h	Win32 Console	
Dll Characteristics	0000h		
Size of Stack Reserve	00100000h		
Size of Stack Commit	00001000h		
Size of Heap Reserve	00100000h		
Size of Heap Commit	00001000h		
Loader Flags	00000000h	Obsolete	
Number of Data Directories	00000010h		

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)

1	nmon.exe	LoadLibraryExW ("kernel32.dll", NULL, LOAD_LIBRARY_SEARCH_SYSTEM32)
1	nmon.exe	GetProcAddress (0x77ce0000, "CreateMutexW")
1	nmon.exe	CreateMutexW (NULL, FALSE, "Global\EKANS")
2	nmon.exe	NtWaitForSingleObject (0xffffffff, FALSE, 0x32a2fecc)

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))

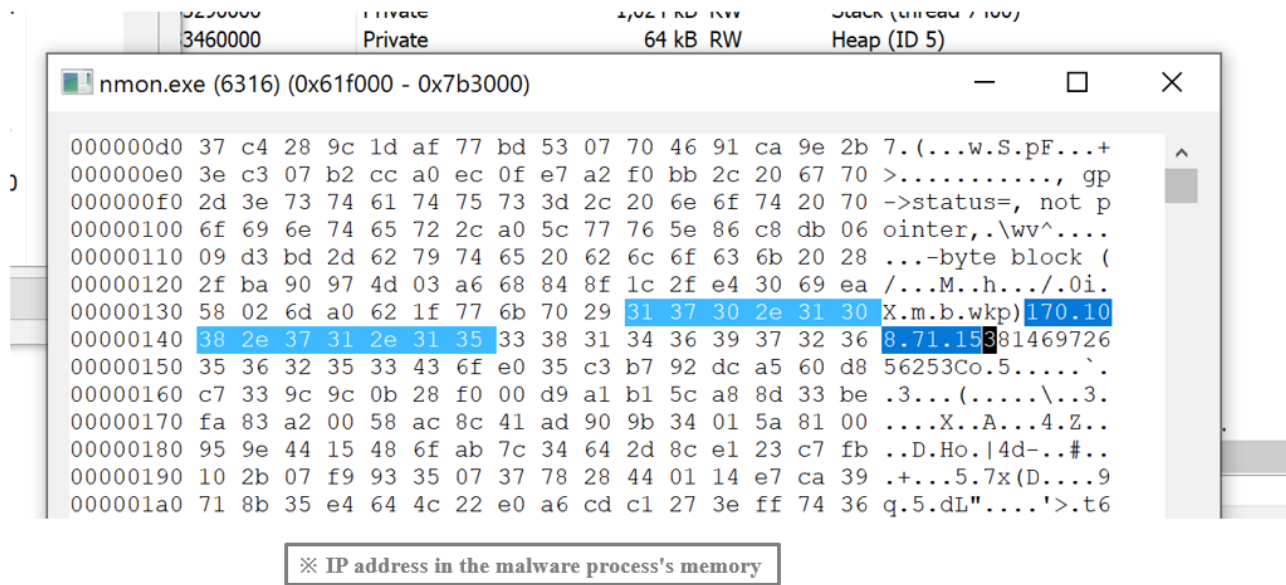


Fig. 5 - Checking specific IP address

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.

※ On this screen, the function called by malware is recorded in chronological order from top to bottom.

#	Time of Day	Thread	Module	API	Return Value	Error	Duration
1532	3:36:43.764 PM	1	apphelp.dll	RtlCaptureStackBackTrace (0, 16, 0x0019fbc4, NULL)	2		0.0000004
1533	3:36:43.764 PM	1	apphelp.dll	RtlLeaveCriticalSection (0x745d1560)	STATUS_SUCCESS		0.0000004
1534	3:36:43.764 PM	2	nmon.exe	NtWaitForSingleObject (0xffffffff, FALSE, 0x0252fcdc)	STATUS_TIMEOUT		0.0001138
1535	3:36:43.765 PM	2	nmon.exe	NtWaitForSingleObject (0xffffffff, FALSE, 0x0252fcdc)	STATUS_TIMEOUT		0.0004153
1536	3:36:43.765 PM	2	nmon.exe	NtWaitForSingleObject (0xffffffff, FALSE, 0x0252fcdc)	STATUS_TIMEOUT		0.0006804
1537	3:36:43.766 PM	2	nmon.exe	NtWaitForSingleObject (0xffffffff, FALSE, 0x0252fcdc)	STATUS_TIMEOUT		0.0003349
1538	3:36:43.766 PM	2	nmon.exe	NtWaitForSingleObject (0xffffffff, FALSE, 0x0252fcdc)	STATUS_TIMEOUT		0.0005213
1539	3:36:43.767 PM	2	nmon.exe	NtWaitForSingleObject (0xffffffff, FALSE, 0x0252fcdc)	STATUS_TIMEOUT		0.0018248
1540	3:36:43.768 PM	2	nmon.exe	NtWaitForSingleObject (0xffffffff, FALSE, 0x0252fcdc)	STATUS_TIMEOUT		0.0014822
1541	3:36:43.769 PM	1	nmon.exe	GetProcAddress (0x77310000, "GetAddrInfoW")	0x77322180		0.0000090
1542	3:36:43.769 PM	1	KERNELBASE.dll	RtlInitString (0x0019fd04, "GetAddrInfoW")			0.0000004
1543	3:36:43.769 PM	1	apphelp.dll	memset (0x0019fd04, 0, 128)	0x0019fd04		0.0000004
1544	3:36:43.769 PM	1	apphelp.dll	RtlEnterCriticalSection (0x745d1560)	STATUS_SUCCESS		0.0000000
1545	3:36:43.769 PM	1	apphelp.dll	RtlCaptureStackBackTrace (0, 16, 0x0019fbc4, NULL)	2		0.0000004
1547	3:36:43.769 PM	1	nmon.exe	GetAddrInfoW ("MDS.HONDA.COM", NULL, 0x1345efbc, 0x1345ef30)	WSAHOST_NOT_FOUND	11001 = そのようなホストは不明です。	0.0003150
1549	3:36:43.769 PM	1	ws2_32.dll	RtlIpv4StringToAddressW ("MDS.HONDA.COM", TRUE, 0x0019fd04, 0x00000000)	STATUS_INVALID_PARAMETER_1	0x00000004 = 無効なパラメーターまたは値が渡されました。	0.0000004
1550	3:36:43.769 PM	1	KERNELBASE.dll	NtWaitForSingleObject (0x00000000, FALSE, 0x0019fd04)	STATUS_TIMEOUT		0.0000017
1551	3:36:43.769 PM	1	KERNELBASE.dll	RtlExpandString (0x00000000, 0x00000000, 0x00000000, 0x00000000)	STATUS_SUCCESS		0.0000009
1552	3:36:43.769 PM	1	KERNELBASE.dll	RtlInitUnicodeString (0x00000000, 0x00000000)	STATUS_SUCCESS		0.0000004
1553	3:36:43.769 PM	1	KERNELBASE.dll	GetLoadTime (0x00000000, 0x00000000)	STATUS_SUCCESS		0.0004268
1554	3:36:43.769 PM	1	KERNEL32.DLL	GetAddrInfoEx (0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000)	STATUS_RESOLVE_NAME	0x0000000a = 指定されたリソースの種類がイメージファイルから見つかりません。	0.0000034
1546	3:36:43.769 PM	1	apphelp.dll	RtlLeaveCriticalSection (0x745d1560)	STATUS_SUCCESS		0.0000000
1547	3:36:43.769 PM	1	nmon.exe	GetAddrInfoW ("MDS.HONDA.COM", NULL, 0x1345efbc, 0x1345ef30)	WSAHOST_NOT_FOUND	11001 = そのようなホストは不明です。	0.0000000
1548	3:36:43.769 PM	1	ws2_32.dll	RtlIpv4StringToAddressW ("MDS.HONDA.COM", TRUE, 0x0019fd04, 0x00000000)	STATUS_INVALID_PARAMETER_1	0x00000004 = 無効なパラメーターまたは値が渡されました。	0.0000004
1561	3:36:43.769 PM	1	apphelp.dll	_wcsicmp ("SysWow64\InstalShield", "system32\mswsock.dll", 24)	3		0.0000000
1562	3:36:43.769 PM	1	apphelp.dll	_wcsicmp ("System32", "system32\mswsock.dll", 10)	12		0.0000004
1563	3:36:43.769 PM	1	apphelp.dll	_wcsicmp ("mswsock.dll", "apphelp.dll")	10		0.0000000
1564	3:36:43.769 PM	1	apphelp.dll	_wcsicmp ("mswsock.dll", "cmd.exe")	10		0.0000004
1565	3:36:43.769 PM	1	apphelp.dll	_wcsicmp ("mswsock.dll", "csrss.exe")	10		0.0000004
1566	3:36:43.769 PM	1	apphelp.dll	_wcsicmp ("mswsock.dll", "java.exe")	3		0.0000004
1567	3:36:43.769 PM	1	apphelp.dll	_wcsicmp ("mswsock.dll", "javaw.exe")	3		0.0000000
1568	3:36:43.769 PM	1	apphelp.dll	_wcsicmp ("mswsock.dll", "javaws.exe")	3		0.0000004

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.

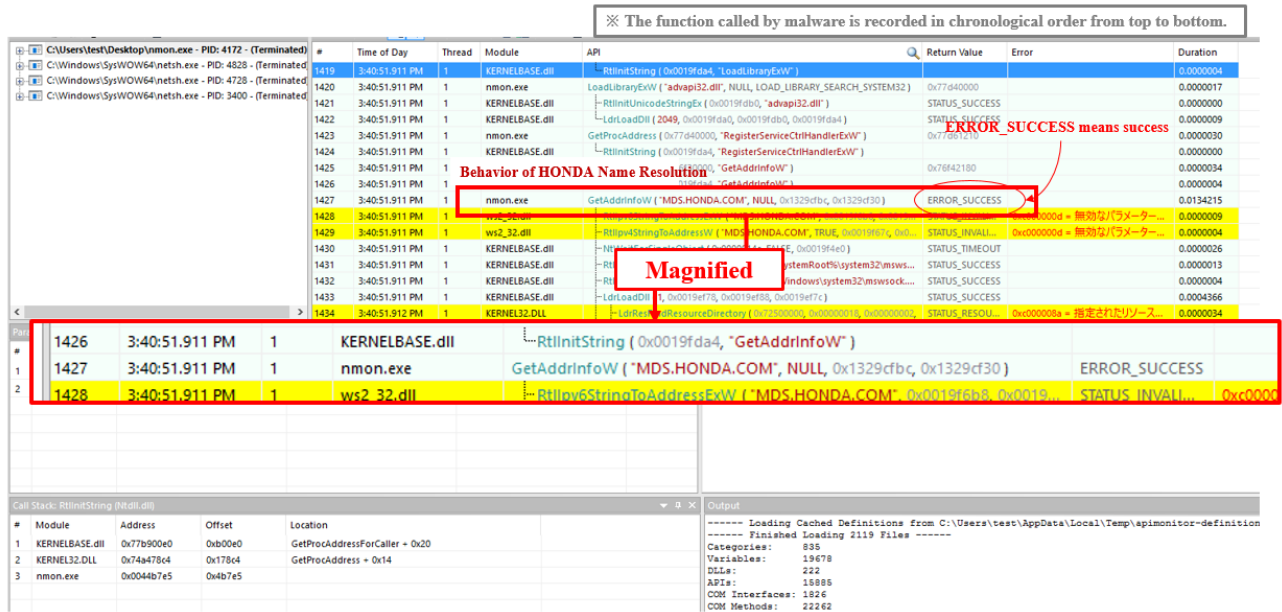


Fig. 7 - Operation continues when name resolution is possible for a specific IP address

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)

```

^ 0014FE80 0044B7E5 CALL to CreateProcessW from nmon.0044B7E3
0014FE84 10F4C740 ModuleFileName = "C:\Windows\system32\netsh.exe"
0014FE88 10FB21E0 CommandLine = "netsh advfirewall set allprofiles firewallpolicy blockinbound,blockoutbound"
0014FE8C 00000000 pProcessSecurity = NULL
0014FE90 00000000 pThreadSecurity = NULL
0014FE94 00000001 InheritHandles = TRUE
0014FE98 00000400 CreationFlags = CREATE_UNICODE_ENVIRONMENT
0014FE9C 10FC8000 pEnvironment = 10FC8000
0014FEA0 00000000 CurrentDir = NULL
0014FEA4 10F67D3C pStartupInfo = 10F67D3C
0014FEA8 10F67D18 pProcessInfo = 10F67D18
0014FEAC 0044A744 RETURN to nmon.0044A744

```

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.

```

^ 0014FE80 0044B7E5 CALL to CreateProcessW from nmon.0044B7E3
0014FE84 10F1C480 ModuleFileName = "C:\Windows\system32\netsh.exe"
0014FE88 10FE41E0 CommandLine = "netsh advfirewall set allprofiles state on"
0014FE8C 00000000 pProcessSecurity = NULL
0014FE90 00000000 pThreadSecurity = NULL
0014FE94 00000001 InheritHandles = TRUE
0014FE98 00000400 CreationFlags = CREATE_UNICODE_ENVIRONMENT
0014FE9C 10FFC000 pEnvironment = 10FFC000
0014FEA0 00000000 CurrentDir = NULL
0014FEA4 10F67D3C pStartupInfo = 10F67D3C
0014FEA8 10F67D18 pProcessInfo = 10F67D18
0014FEAC 0044A744 RETURN to nmon.0044A744

```

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	jg 8A551C3A	
898424 50020000	mov dword ptr ss:[esp+250],esi	
80BCF4 4C0A0000	lea edi,dword ptr ss:[esp+esi*8+A4C]	
8837	mov esi,dword ptr ds:[edi]	esi:"Sophos Health Service"
887F 04	mov edi,dword ptr ds:[edi+4]	
39CD	cmp ebp,ecx	
0F83 35050000	jae 8A551CF7	
898424 E4040000	mov dword ptr esi:[esp+4E4],esi	[esp+4E4]:"Sophos File Scanner Service"
897C24 28	mov dword ptr ss:[esp+28],edi	
80DCE8	lea ecx,dword ptr ds:[eax+ebp*8]	[eax+ebp*8]:"AJRouter"
898C24 1C070000	mov dword ptr ss:[esp+71C],ecx	[esp+71C]:&"AJRouter"
8811	mov edx,dword ptr ds:[ecx]	edx:&"AllJoyn Router Service"
8859 04	mov ebx,dword ptr ds:[ecx+4]	
891424	mov dword ptr ss:[esp],edx	[esp]:"AllJoyn 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.

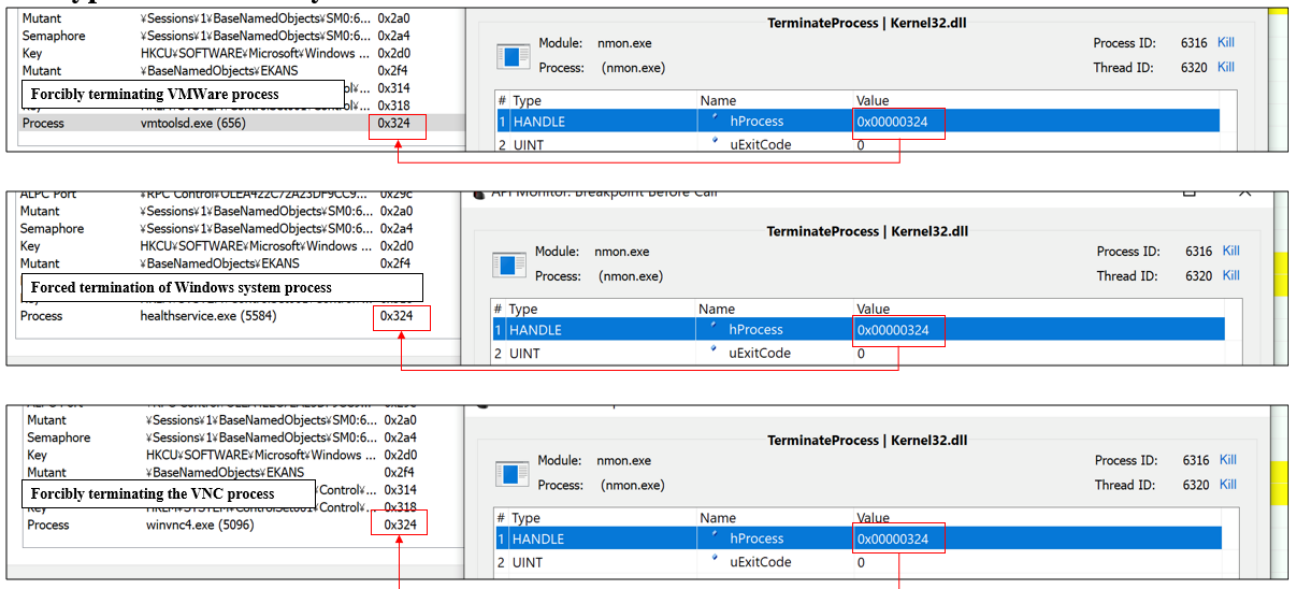


Fig. 11 - Forcibly terminates various legitimate processes

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.

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

nmon.exe	!Unknown:Release ()	1	
nmon.exe	NtWaitForSingleObject (0x00000000, FALSE, 0x32a2fccc)	STATUS_TIMEOUT	
nmon.exe	NtWaitForSingleObject (0x00000000, FALSE, 0x32a2fccc)	STATUS_TIMEOUT	
nmon.exe	GetEnvironmentVariableW ("windir", 0x129ac820, 100)	10	
nmon.exe	GetEnvironmentVariableW ("SystemDrive", 0x129ac880, 100)	2	
nmon.exe	VirtualAlloc (0x12a20000, 1048576, MEM_COMMIT, PAGE_READWRITE)	0x12a20000	
nmon.exe	VirtualAlloc (0x1166e000, 65536, MEM_COMMIT, PAGE_READWRITE)	0x1166e000	
nmon.exe	LoadLibraryW ("kernel32.dll")	0x77c00000	
nmon.exe	GetProcAddress (0x77c00000, "GetLogicalDriveStringsW")	0x77d0cb90	
nmon.exe	GetLogicalDriveStringsW (254, "")	12	← Get all drive information
nmon.exe	GetProcAddress (0x77c00000, "GetDriveTypeW")	0x77d0cab0	
nmon.exe	GetDriveTypeW ("A:")	DRIVE_REMOVABLE	
nmon.exe	GetProcAddress (0x77c00000, "GetVolumeInformationW")	0x77d0cc00	
nmon.exe	GetVolumeInformationW ("A:", 0x129c6300, 256, 0x12a12b00, 0x12a12b00, 0x12a12b00, 0x129c6400, 256)	FALSE	21 = フォルダの準備ができ
nmon.exe	NtWaitForSingleObject (0x00000000, FALSE, 0x32a2fccc)	STATUS_TIMEOUT	
nmon.exe	NtWaitForSingleObject (0x00000000, FALSE, 0x32a2fccc)	STATUS_TIMEOUT	
nmon.exe	NtWaitForSingleObject (0x00000000, FALSE, 0x32a2fccc)	STATUS_TIMEOUT	
nmon.exe	NtWaitForSingleObject (0x00000000, FALSE, 0x32a2fccc)	STATUS_TIMEOUT	
nmon.exe	NtWaitForSingleObject (0x00000000, FALSE, 0x32a2fccc)	STATUS_TIMEOUT	
nmon.exe	NtWaitForSingleObject (0x00000000, FALSE, 0x32a2fccc)	STATUS_TIMEOUT	
nmon.exe	SetEvent (0x000001a4)	TRUE	
nmon.exe	NtWaitForSingleObject (0x00000000, FALSE, 0x32a2fccc)	STATUS_TIMEOUT	
nmon.exe	WaitForSingleObject (0x000001a4, INFINITE)	WAIT_OBJECT_0	
nmon.exe	timeEndPeriod (1)	MMSYSERR_NOERROR	
nmon.exe	WaitForSingleObject (0x000001d0, 60000)	WAIT_OBJECT_0	
nmon.exe	SetEvent (0x000001d0)	TRUE	
nmon.exe	GetDriveTypeW ("C:")	DRIVE_FIXED	
nmon.exe	GetVolumeInformationW ("C:", 0x129c6500, 256, 0x12a12510, 0x12a12510, 0x12a12520, 0x129c6600, 256)	TRUE	
nmon.exe	GetDriveTypeW ("D:")	DRIVE_CDROM	
nmon.exe	timeBeginPeriod (1)	MMSYSERR_NOERROR	
nmon.exe	SetEvent (0x000001a4)	TRUE	
nmon.exe	SwitchToThread ()	TRUE	
nmon.exe	NtWaitForSingleObject (0x00000000, FALSE, 0x32a2fccc)	STATUS_TIMEOUT	
nmon.exe	WaitForSingleObject (0x000001a4, INFINITE)	WAIT_OBJECT_0	
nmon.exe	GetFileAttributesExW ("C:", GetFileExInfoStandard, 0x12994a00)	TRUE	
nmon.exe	CreateFileW ("C:\", GENERIC_READ, FILE_SHARE_READ FILE_SHARE_WRITE, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL)	INVALID_HANDLE_VALUE	3 = 指定されたパスが見つ
nmon.exe	GetProcAddress (0x77c00000, "FindFirstFileW")	0x77d0c900	
nmon.exe	FindFirstFileW ("C:*", 0x12949b00)	0x0084ed10	
nmon.exe	CreateFileW ("C:\\$Recycle.Bin", GENERIC_READ, FILE_SHARE_READ FILE_SHARE_WRITE, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL)	INVALID_HANDLE_VALUE	5 = アクセスが拒否され
nmon.exe	NtWaitForSingleObject (0x00000000, FALSE, 0x32a2fccc)	STATUS_TIMEOUT	
nmon.exe	FindFirstFileW ("C:\\$Recycle.Bin*", 0x12949b00)	0x0084ef50	
nmon.exe	GetProcAddress (0x77c00000, "FindFirstFileW")	0x77d0c900	

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
1111D7D0	FC D2 B1 00	û±..docx..accdb
1111D7E0	2E 61 63 63	.accde.accdr....
1111D7F0	2E 61 63 63	.accdt...asp....
1111D800	2E 61 73 70	.aspx.back.....
1111D810	2E 62 61 63	.backup.backupdb
1111D820	2E 62 61 68	.bak.mdb.mdc.mdf
1111D830	2E 77 61 72	.war.xls.xlsx...
1111D840	2E 78 6C 73	.xism...xlr.zip
1111D850	2E 72 61 72	.rar.sqlitedb...
1111D860	2E 73 71 6C	.sql.py.ppam.pps
1111D870	2E 70 70 73	.ppsm.ppsx...ppt
1111D880	70 70 74 6D	pptm.pptx...hpp
1111D890	2E 6A 61 76	.java...jsp.php
1111D8A0	2E 64 6F 63	.doc.docm...pst
1111D8B0	2E 70 73 64	.psd.dotdotm.cpp
1111D8C0	2E 63 73 00	.cs..csv.bkp.db.
1111D8D0	2E 64 62 2D	.db-journal.....
1111D8E0	2E 63 73 70	.csproj...sln.md.
1111D8F0	2E 70 6C 2E	.pl.js.html..htm
1111D900	2E 64 62 66	.dbf.rdo.arc.vhd
1111D910	2E 76 6D 64	.vmdk...vdi....
1111D920	2E 76 68 64	.vhdx...edb.c.h
1111D930	2E 64 6C 6C	.dll.exe.sys.mui
1111D940	2E 74 6D 70	.tmp.lnk.config.
1111D950	2E 6D 61 6E	.manifest...t1b
1111D960	2E 6F 6C 62	.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.

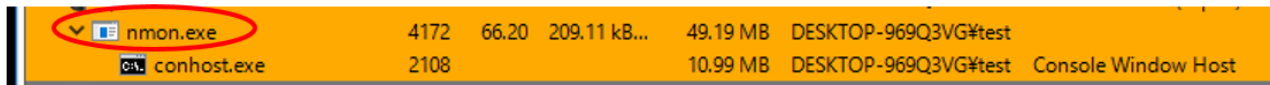


Fig. 14 - SNAKE encrypting files

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	1111D980	"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	1111D850	"thumbs.db"
11057F4C	1111D840	"ntuser.ini"
11057F50	1111D830	"ntuser.dat.log"
11057F54	1111D820	"ntuser.dat"
11057F58	1111D810	"iconcache.db"
11057F5C	1111D800	"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.

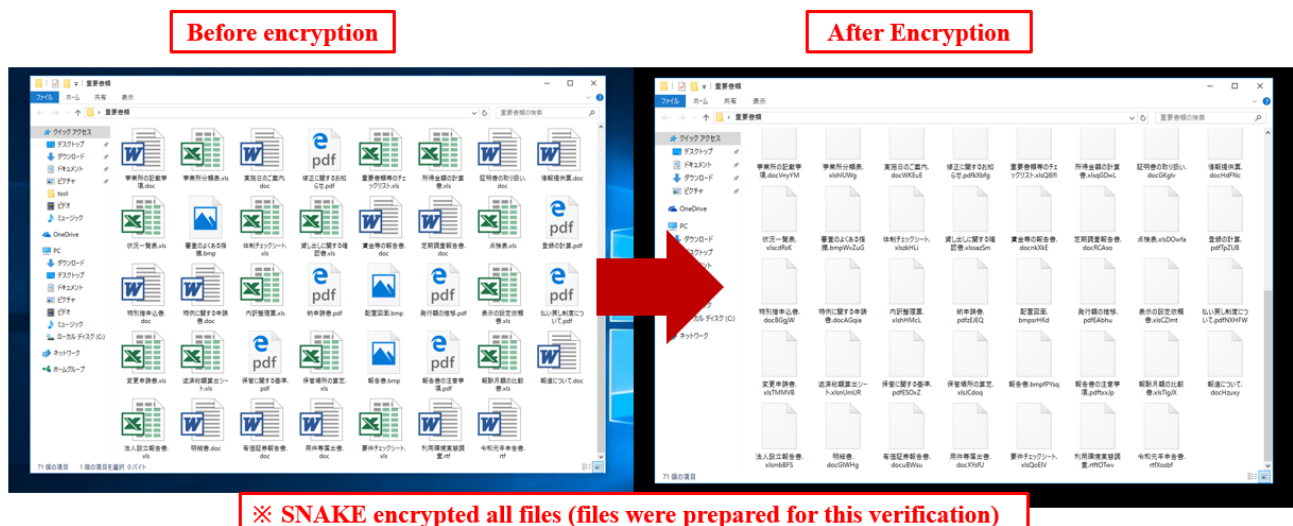


Fig. 16 - Comparing before and after SNAKE encryption

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.

Address	Hex dump	ASCII	CALL to WriteFile from nmon.0044B7E3
1105C800	4C FF 81 03 01 01 14 66 6A 6C 68 64 63 65 67 62	L.....fjhdcegb	hFile = 000007C0
1105C810	6D 69 64 70 6F 61 67 6C 6E 64 69 01 FF 82 00 01	midpoagIndi.....	Buffer = 1105C800
1105C820	03 01 08 46 69 6C 65 4E 61 60 65 01 0C 00 01 02	...FileName....	nBytesWritten = 1A4 (420.)
1105C830	49 58 01 0A 00 01 11 45 4E 43 52 59 50 54 45 44	IV.....ENCRYPTE	pBytesWritten = 110FFDF8
1105C840	5F 41 45 53 5F 48 65 79 01 0A 00 00 00 FE 01 54	...AES.Key.....	pOverlapped = NULL
1105C850	FF 82 01 39 43 3A 5C 50 72 6F 67 72 61 60 2D 46	...9C:YProgram F	RETURN to nmon.0044A744
1105C860	69 6C 65 73 5C 57 69 72 65 73 68 61 72 68 5C 73	ilesYmiresharkFs	10F96B1C
1105C870	6E 6D 70 5C 6D 69 62 73 5C 4D 45 54 41 2D 50 4F	nmpYmibsYMETIA-PO	0000029C
1105C880	4C 49 43 59 2D 50 49 42 2D 6F 72 69 67 01 10 C4	LICY-PIB-orig.ト	10F2CA80
1105C890	66 26 A6 A2 18 4D FE E5 A3 8B 73 08 05 8A 9D 01	!&...W・登.s.コ.	00446C98
1105C8A0	FE 01 00 80 84 11 CA 7D 69 15 3E 67 14 0E AC 8F	*....n]i>....	RETURN to nmon.00446C98 from nmon.004104F0
1105C8B0	CD 5D 40 2D 44 18 7C 76 B1 02 94 5C 97 EA 14 9E	~ 0 D. v f Y. .	0017036C
1105C8C0	8F 23 06 A8 0B 61 4E 62 E1 0E 5D CF 3A 81 28 17	*...aNB・ j;.k.	110FFE33
1105C8D0	D7 E5 54 4E 07 69 08 7E 80 5F 6E F7 07 A6 13 7E	舞.N.i.~.p.....	004493DE
1105C8E0	43 04 31 37 BE 62 9A 93 C4 C9 96 5C 5B 18 85 5A	C.17bb.トJ.V.4Z	Entry address
1105C8F0	6C 90 85 24 68 47 CF C8 22 67 5F A1 8A F2 3A 47	l..\$hG7?g. 破.G	00000000
1105C900	F2 80 71 F8 46 34 79 19 03 80 E9 F7 9A 34 2F 95	*q・4y.t. /.	00000000
1105C910	56 63 B4 87 25 FC 2F 40 11 56 EC 23 88 FE B7 3C	Ve. % . @ . V . . <	0044BA2B
1105C920	06 3C 21 01 18 A8 C5 E6 0E 68 9E 1B BA F9 E9 33	<1A.?? . .3.3	RETURN to nmon.0044BA2B from nmon.0042A130
1105C930	A0 31 FC 62 12 E0 23 49 89 79 68 09 1B FF 7A AE	.l. . . .yh. .zs	RETURN to nmon.0044BA3A from nmon.0044B9F0
1105C940	BE 7F B9 81 43 8F 87 8B CD F1 63 46 C3 68 C7 1A	t. . C . . . FjhZ.	10F96900
1105C950	FD AD 53 9D 48 FF 77 EA 8E 59 E7 12 3E 50 6B D7	*. 	77CFE2F9
1105C960	D1 16 B4 8E 7F 7F 0D 25 2F 03 7E 7E 06 21 AC 3C	A. . . . % / . ! <	10F96900
1105C970	B1 59 DC A8 42 78 40 00 5D 24 7C 49 AC 43 82 5D	Y?Y.Bx@.] .Cb	77CFE2E0
1105C980	DF 40 AC 14 64 F8 BE 16 AC ED 17 FB 1F F3 E6 A0	*. . d *	32E5FFDC
1105C990	93 6D 39 9B 28 DC 62 17 39 7E 85 0E EF 31 57 01	.m0.(?b.0? . . .	77DE27C7
1105C9A0	74 59 8F 00 00 00 00 00 00 00 00 00 00 00 00	tY*.....	32E5FF80
1105C9B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		32E5FF84
1105C9C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		10F96900
1105C9D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		6825BE69
1105C9E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		00000000
1105C9F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		00000000
1105C9FF	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		00000000

"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

Address	Hex dump	ASCII	CALL to WriteFile from nmon.0044B7E3
10FFEC90	75 4B 41 4E 53 00 00 00 00 00 00 00 00 00 00	EKANS.....	hFile = 000007A8
10FFEC9D	00 00 00 01 00 00 00 00 00 00 00 01 00 00 00		Buffer = 10FFEC90
10FFECB0	00 00 00 00 00 00 00 00 00 FF 81 03 01 01 14 00		nBytesWritten = 5
10FFEC90	01 00 00 00 02 00 00 00 02 00 00 00 03 00 00		pBytesWritten = 11389DF8
10FFECB0	A5 01 00 00 00 00 00 00 45 4B 41 4E 53 00 00 00EKANS.....	pOverlapped = NULL
10FFECED	00 00 00 00 00 00 00 00 FF 81 03 01 01 14 00		RETURN to nmon.0044A744
10FFECF0	00 00 00 00 00 00 00 00 01 00 00 00 01 00 00		10F3669C

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

The "footer" that is added to the end of an encrypted file

Original file name before encryption

EKANS markers that indicate that they have been encrypted by SNAKE. (SNAKE is inverted.)

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-----
MIIBCgKCAQEA1GCKUHXITsiWc1d8V0vo1Y9Jm18RDZEmMS6OkHI7pZT0RHATHIR
BFITZY9bXrl6RFdUwmIX0WYn5ZqllhLAEe1cq48RpJ/KK2OeiTn0CJ1CGmOOJvfm
5rFa8whVAU9cnh/iVCcf+aEHJVcHhzB5tTtI3tIBldfzaLL6GR5EmytbQ3V3O1Uk
Y4FCkxYOMVoPzPtRG3vo3688uUWpZIKBV7e6dht/mAhuCEIIRGcdpAEf6f4zUUYf
DtHcDafMVEA4Sy/DDsd76wAyBIM0XKLv1+vH476TN1K1tIRBrR98QF15mIXkgqz6
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 (0x0000324, 0x12a7fe7c)	FALSE
nmon.exe	GetFileType (0x0000324)	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)	TRUE ← Check if the last 5 bytes of the file are EKANS (check if encrypted)
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 (0x0085bf8, 16, 0x12b198f0)	TRUE
nmon.exe	CryptGenRandom (0x0085bf8, 32, 0x12b7d280)	TRUE
nmon.exe	ReadFile (0x00000324, 0x12b198e4, 102400, 0x12a7fd8, NULL)	TRUE ← Read the file to be encrypted
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, 0x12a7fd8, 0x12a7dfc)	TRUE ← Write encrypted data to a file
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, 0x12b198e4, 102400, 0x12a7fd8, NULL)	TRUE
nmon.exe	CryptGenRandom (0x0085bf8, 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, 0x12a7fd8, NULL)	TRUE ← Write footer to end of file
nmon.exe	WriteFile (0x00000324, 0x12b19940, 4, 0x12a7fd8, NULL)	TRUE
nmon.exe	WriteFile (0x00000324, 0x12b19948, 5, 0x12a7fd8, NULL)	TRUE ← Writing EKANS Markers to the End of a File
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_SHARE_WRITE, NULL, CREATE_NEW, FILE_ATTRIBUTE_NORMAL, NULL)	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 (0x0084ecc0, 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.chkfyuy", 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.logWcgwH", MOVEFILE_REPLACE_EXISTING)	TRUE
nmon.exe	MoveFileExW ("C:\Users\hanako\AppData\Local\Microsoft\Internet Explorer\CacheStorage\edbres0001jrs", "C:\Users\hanako\AppData\Local\Microsoft\Internet Explorer\CacheStorage\edbres0001jrsRDAP", MOVEFILE_REPLACE_EXISTING)	TRUE
nmon.exe	MoveFileExW ("C:\Users\hanako\AppData\Local\Microsoft\Internet Explorer\CacheStorage\edbres0002jrs", "C:\Users\hanako\AppData\Local\Microsoft\Internet Explorer\CacheStorage\edbres0002jrsUGRmS", MOVEFILE_REPLACE_EXISTING)	TRUE
nmon.exe	MoveFileExW ("C:\Users\hanako\AppData\Local\Microsoft\Internet Explorer\CacheStorage\edbtmptmp.log", "C:\Users\hanako\AppData\Local\Microsoft\Internet Explorer\CacheStorage\edbtmptmp.logmyf3y", MOVEFILE_REPLACE_EXISTING)	TRUE
nmon.exe	MoveFileExW ("C:\Users\hanako\AppData\Local\Microsoft\Internet Explorer\ie4unit-CleariconCache.log", "C:\Users\hanako\AppData\Local\Microsoft\Internet Explorer\ie4unit-CleariconCache.loggJlJK", MOVEFILE_REPLACE_EXISTING)	TRUE
nmon.exe	MoveFileExW ("C:\Users\hanako\AppData\Local\Microsoft\Internet Explorer\ie4unit-UserConfig.log", "C:\Users\hanako\AppData\Local\Microsoft\Internet Explorer\ie4unit-UserConfig.loggabab", 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.xmlkflWY", MOVEFILE_REPLACE_EXISTING)	TRUE
nmon.exe	MoveFileExW ("C:\Users\hanako\AppData\Local\Microsoft\Internet Explorer\MSIMGSZ.DAT", "C:\Users\hanako\AppData\Local\Microsoft\Internet Explorer\MSIMGSZ.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_rated_at_5_stars.wpl", MOVEFILE_REPLACE_EXISTING)	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_added_in_the_last_month.wpl", MOVEFILE_REPLACE_EXISTING)	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

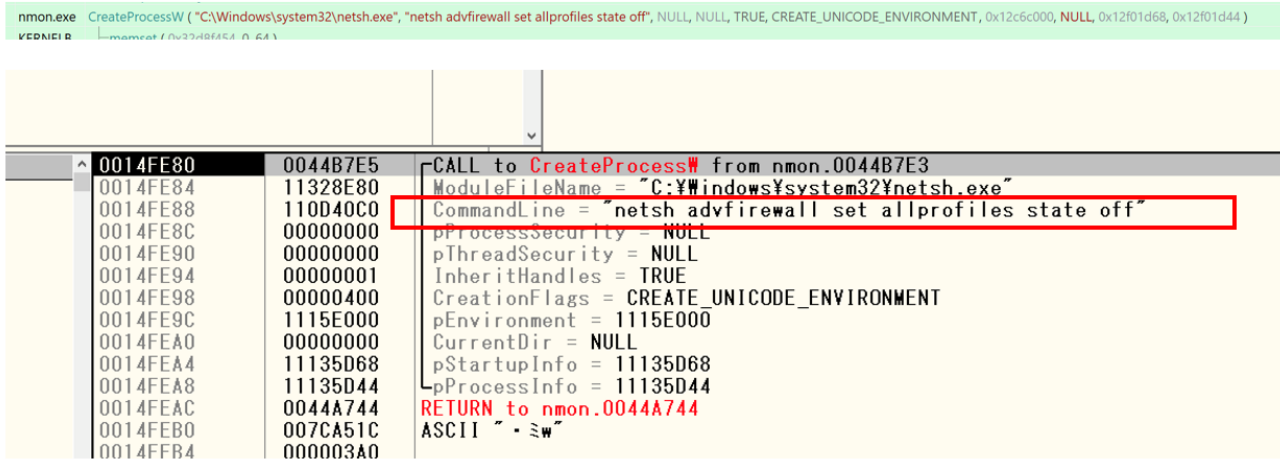


Fig. 22 - Disabling the firewall after the encryption process

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.

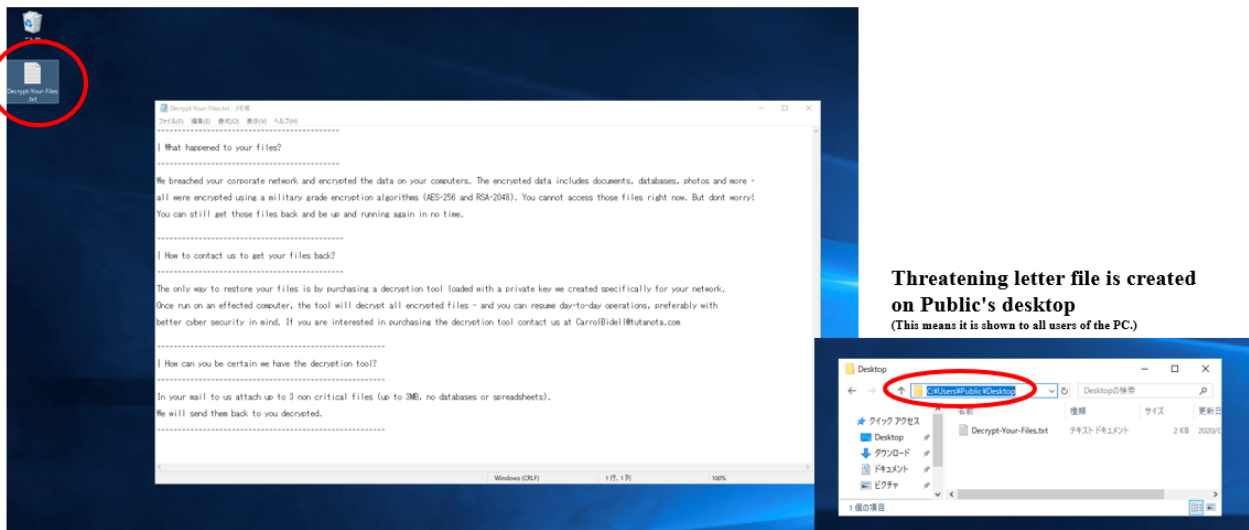
Typical ransomware basically creates threatening letter after encryption. However, this SNAKE does not create any threatening letter on infected user PC or server, although it does encrypt files.

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:¥).

SNAKE behavior is different in case the infected terminal is a domain controller.

☞ **Create threatening letter on the desktop without encrypting files.**

(It does not create threatening letters on other servers and PCs, but performs encryption)



☞ **It is very important for the hackers to show threatening letter to the victim.**

It is highly likely to have been developed on the assumption that it can penetrate the domain controller.

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

```

3: [esp+8] 005DEDC0 <nmon._ptr_main_FpIgmtrc
4: [esp+C] 110125F0
5: [esp+10] 00000000
1105BF00 1101B410 "select DomainRole FROM win32_ComputerSystem"
1105BF04 00000028
1105BF08 005DEDC0 nmon._ptr_main_FpIgmfdkdommbapckgjp
1105BFOC 110125F0

```

```

wbemdisp.dll - SetErrorInfo (0, NULL)
combase.dll IWbemServices::ExecQuery ( "WQL", "select DomainRole FROM Win32_ComputerSystem", 272, NULL, 0x33dfcbcc)
combase.dll NdrClientCall2 (0x72ae1208, 0x72ae1138, in)

```

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".

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](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.

- For example, for a server (StandaloneServer) that is not a domain controller, the domain role is "2". (This value is either "0" or "1" for the user PC.)

In case it is a StandaloneServer

005544E6	8800	mov eax,dword ptr ds:[eax]	EAX 00000002
005544E8	85C9	test ecx,ecx	ECX 00000001
005544EA	76 18	jbe nmon.S54507	ECX 00000003
005544EC	0FB700	movzx eax,word ptr ds:[eax]	EDX 10EDC014
005544EE	66:83F8 03	cmp ax,3	EBP 10F17E18
005544F3	76 09	jbe nmon.S544FE	EBP 10F17E00

Check if it is 3 or less => "2", so it is OK.

In this case, the files are encrypted, and no threatening letter is created

- For domain controllers (Backup domain controller, Primary domain controller), the value is either "4" or "5".

In case it is a Domain Controller

00554407	884424 24	mov eax,dword ptr ss:[esp+24]	EAX 00000005
005544E3	8800	mov ecx,dword ptr ds:[eax+1]	ECX 00000003
005544E6	8800	mov eax,dword ptr ds:[eax]	ECX 00000001
005544E8	85C9	test ecx,ecx	EDX 1104C014
005544EA	76 18	jbe nmon.S54507	EBP 11087E18
005544EC	0FB700	movzx eax,word ptr ds:[eax]	ESP 11087F00
005544EE	66:83F8 03	cmp ax,3	ESI 00000000
005544F3	76 09	jbe nmon.S544FE	
005544F5	C64424 2C 01	mov byte ptr ds:[esp+2C],1	
005544FA	83C4 28	add esp,28	

Check whether "3" or less => "5", so NG

The files are not encrypted, and a threatening letter is created

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

```

    .text:00553E21 04C call    main_check_domain_role ; Check domain role
    .text:00553E26 04C movzx  eax, byte ptr [esp+4Ch+var_4C] ; Move with Zero-Extend
    .text:00553E2A 04C test   al, al ; Logical Compare
    .text:00553E2C 04C jnz   short loc_553E8C ; Jump if Not Zero (ZF=0)

    ; Branch 1: Role > 3 (NG)
    .text:00553E8C loc_553E8C:
    .text:00553E8C 04C call    main_mjapjckncgmiofcebfi_func1 ; Call Procedure
    .text:00553E91 04C call    main_create_random_note ; Call Procedure
    .text:00553E96 04C mov    [esp+4Ch+arg_0], 0
    .text:00553E9B 04C add   esp, 4Ch ; Add
    .text:00553E9E 000 retn   ; Return Near from Procedure

    ; Branch 2: Role <= 3 (OK)
    .text:00553E2E 04C call    main_mjapjckncgmiofcebfi_func2 ; Call Procedure
    .text:00553E33 04C mov    eax, [esp+4Ch+var_48]
    .text:00553E37 04C mov    ecx, [esp+4Ch+var_4C]
    .text:00553E3A 04C lea   edx, [esp+4Ch+var_24] ; Load Effective Address
    .text:00553E3E 04C mov    [esp+4Ch+var_4C], edx
    .text:00553E41 04C mov    [esp+4Ch+var_48], ecx
    .text:00553E45 04C mov    [esp+4Ch+var_44], eax
    .text:00553E49 04C mov    eax, [esp+4Ch+arg_0]
    .text:00553E4D 04C mov    [esp+4Ch+var_40], eax
    .text:00553E51 04C mov    eax, [esp+4Ch+arg_4]
    .text:00553E55 04C mov    [esp+4Ch+var_3C], eax
    .text:00553E59 04C call    runtime_concatstring2 ; Call Procedure
  
```

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.

In case the infected PC is a domain controller

		Get Public Desktops	
891424	mov dword ptr ss:[esp],edx	[esp]:"public"	
8D5C24 58	lea ebx,dword ptr ss:[esp+58]	[esp+58]:"Carrol81de1@tutanota.com"	
895C24 04	mov dword ptr ss:[esp],ecx		
E8 2A71E8FF	call <mon.runtime_convert_string>		
8B4424 08	mov eax,dword ptr ss:[esp+8]	[esp+8]:"C:\\Users\\Public"	
8B4C24 0C	mov ecx,dword ptr ss:[esp+C]		
894424 60	mov dword ptr ss:[esp+60],eax	[esp+60]:string_autogen_I1X903	
894C24 64	mov dword ptr ss:[esp+64],ecx	[esp+64]:&"Carrol81de1@tutanota.com"	
8B4424 44	mov eax,dword ptr ss:[esp+44]		
890424	mov dword ptr ss:[esp],eax	[esp]:"public"	
8B4424 34	mov eax,dword ptr ss:[esp+34]		
894424 04	mov dword ptr ss:[esp+4],eax		
8D4424 60	lea eax,dword ptr ss:[esp+60]	[esp+60]:string_autogen_I1X903	
894424 08	mov dword ptr ss:[esp+8],eax	[esp+8]:"C:\\Users\\Public"	
C74424 0C 01000000	mov dword ptr ss:[esp+C],1		
C74424 10 01000000	mov dword ptr ss:[esp+10],1		
E8 CE68F4FF	call <mon.fmt_sprintf>		
8B4424 18	mov eax,dword ptr ss:[esp+18]		
8B4C24 14	mov ecx,dword ptr ss:[esp+14]		
C70424 00000000	mov dword ptr ss:[esp],0	[esp]:"public"	
894C24 04	mov dword ptr ss:[esp+4],ecx		
894424 08	mov dword ptr ss:[esp+8],eax	[esp+8]:"C:\\Users\\Public"	
E8 426EE8FF	call <mon.runtime_stringtos11cebyte>		
8B4424 14	mov eax,dword ptr ss:[esp+14]		
894424 30	mov dword ptr ss:[esp+30],eax		
8B4C24 10	mov ecx,dword ptr ss:[esp+10]		
894C24 2C	mov dword ptr ss:[esp+2C],ecx		
8B5424 0C	mov edx,dword ptr ss:[esp+C]		
895424 40	mov dword ptr ss:[esp+40],edx		
E8 359F0200	call <mon.mta_dddabaniqabpe11jaem_func2>		
E8 3052F2FF	call <mon.os_Getenv>		
8B4424 0C	mov eax,dword ptr ss:[esp+C]		
894424 28	mov dword ptr ss:[esp+28],eax		
8B4C24 08	mov ecx,dword ptr ss:[esp+8]	[esp+8]:"C:\\Users\\Public"	

		Get Public Desktops	
EAX	110E5C80	"C:\\Users\\Public"	
ECX	0000000F		
EDX	00000012		
EBP	110E5C80	"C:\\Users\\Public"	
ESP	110F7E80	&"public"	
ESI	0000000F		
EDI	0000000F		
EIP	00553B70	mmon.00553B70	
EFLAGS	00000204		
ZF	0	PF	1
OF	0	SF	0
CF	0	TF	0
IF	1		
LastError	00000000	(ERROR_SUCCESS)	
LastStatus	C0150008	(STATUS_SXS_KEY_NOT_F	
GS	002B	FS	0053
ES	002B	DS	002B
CS	0023	SS	002B
ST(0)	FFFFFFFF0000300000000003	X87r0	Empty in
ST(1)	000000000000000000000000	X87r1	Empty 0.
ST(2)	000000000000000000000000	X87r2	Empty 0.
ST(3)	000000000000000000000000	X87r3	Empty 0.
ST(4)	000000000000000000000000	X87r4	Empty 0.
ST(5)	000000000000000000000000	X87r5	Empty 0.

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 68 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 68 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 68 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 68 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.

Due to the above fault activities, the following phenomena appear on the infected PC

1) In case the infected PC is not a domain controller (general user PC or other server)

- **By disabling Windows firewall settings, the PC temporarily unable to use network communication during encryption.**
- **The PC will become unstable, because system processes are forcibly terminated.**
- **All files except some system files are encrypted.**

2) In case the infected PC is a domain controller

- **The files are not encrypted.**
- **A threatening letter is created on Public's desktop.**

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/>