Malware Analysis: Snake Ransomware

By Nishan Maharjan

Published: 2020-05-15 · Archived: 2026-04-05 23:00:34 UTC

7 min read

May 15, 2020

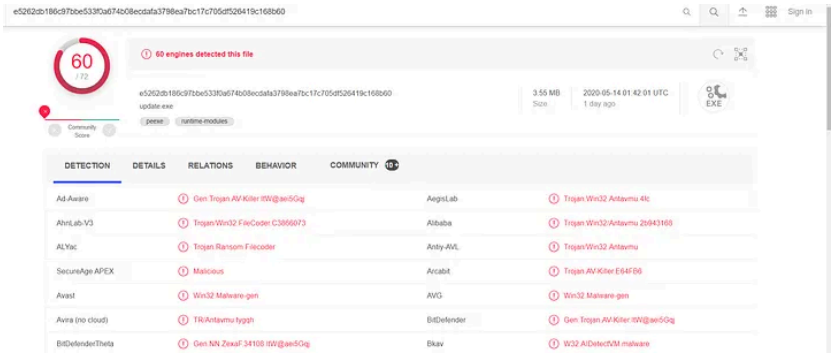
Introduction

This article contains the analysis of snake ransomware. The sample used in analysis was downloaded from https://app.any.run/tasks/7b7d33d9-15f9-4d45-8ae4-291c8cde7ef8/.

Basic Static Analysis

First the sample was submitted to virustotal to see if it is identified as malicious https://www.virustotal.com/gui/file/e5262db186c97bbe533f0a674b08ecdafa3798ea7bc17c705df526419c168b60/detection.

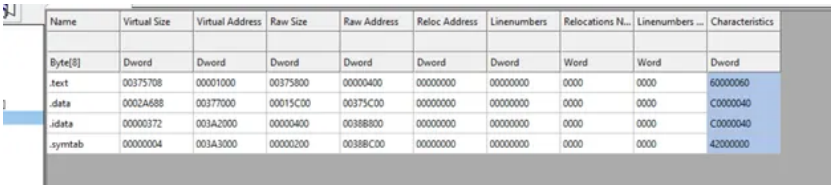
Press enter or click to view image in full size



virustotal

We can see that 60 out of 72 AV engines detect this sample as malicious.

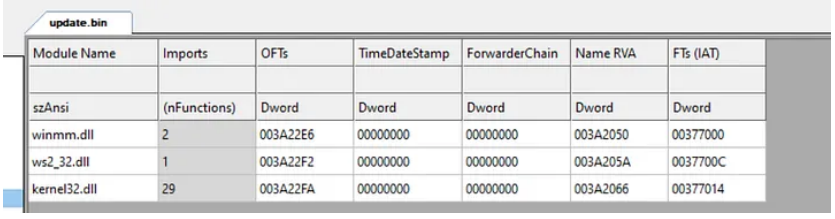
Using coff explorer tool, we observe that there are very few number of imports, and there is a presence of .symtab section. These both are signs of a binary written and compiled using the GO programming language.



Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.text	00375708	00001000	00375800	00000400	00000000	00000000	0000	0000	60000060
.data	0002A688	00377000	00015C00	00375C00	00000000	00000000	0000	0000	C0000040
.idata	00000372	003A2000	00000400	0038B800	00000000	00000000	0000	0000	C0000040
.symtab	00000004	003A3000	00000200	0038BC00	00000000	00000000	0000	0000	42000000

sections

Press enter or click to view image in full size



update.bin

Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
winmm.dll	2	003A22E6	00000000	00000000	003A2050	00377000
ws2_32.dll	1	003A22F2	00000000	00000000	003A205A	0037700C
kernel32.dll	29	003A22FA	00000000	00000000	003A2066	00377014

imports

After string analysis, it is confirmed that this is a go binary.

```
Sysinternals - www.sysinternals.com

!This program cannot be run in DOS mode.
.text
`.data
.idata
.symtab
 Go build ID: "X6lNEpDhc_qgQl56x4du/fgVJOqLlPCCIekQhFnHL/rkxe6tXCg56Ez88otHrz/Y-lXW-OhiIbzg3-ioGRz"
D$$
T$(
```

GO build id

Go compiler adds the project used by the malware author

```
C:/Go/src/bufio/scan.go
C:/Go/src/bufio/bufio.go
C:/Users/WIN1/go/src/jobnhbgnnifpodhhpkmf/nfdlhophkeijadgfddim/nfdlhophkeijadgfddim/pdmnjmklbkflhigdnplp/oleutil.go
C:/Users/WIN1/go/src/jobnhbgnnifpodhhpkmf/nfdlhophkeijadgfddim/nfdlhophkeijadgfddim/winrt.go
C:/Users/WIN1/go/src/jobnhbgnnifpodhhpkmf/nfdlhophkeijadgfddim/nfdlhophkeijadgfddim/variables.go
C:/Users/WIN1/go/src/jobnhbgnnifpodhhpkmf/nfdlhophkeijadgfddim/nfdlhophkeijadgfddim/vt_string.go
C:/Users/WIN1/go/src/jobnhbgnnifpodhhpkmf/nfdlhophkeijadgfddim/nfdlhophkeijadgfddim/variant.go
C:/Users/WIN1/go/src/jobnhbgnnifpodhhpkmf/nfdlhophkeijadgfddim/nfdlhophkeijadgfddim/safearrayslices.go
C:/Users/WIN1/go/src/jobnhbgnnifpodhhpkmf/nfdlhophkeijadgfddim/nfdlhophkeijadgfddim/utility.go
C:/Users/WIN1/go/src/jobnhbgnnifpodhhpkmf/nfdlhophkeijadgfddim/nfdlhophkeijadgfddim/safearray_windows.go
C:/Users/WIN1/go/src/jobnhbgnnifpodhhpkmf/nfdlhophkeijadgfddim/nfdlhophkeijadgfddim/ole.go
C:/Users/WIN1/go/src/jobnhbgnnifpodhhpkmf/nfdlhophkeijadgfddim/nfdlhophkeijadgfddim/iunknown_windows.go
C:/Users/WIN1/go/src/jobnhbgnnifpodhhpkmf/nfdlhophkeijadgfddim/nfdlhophkeijadgfddim/iunknown.go
C:/Users/WIN1/go/src/jobnhbgnnifpodhhpkmf/nfdlhophkeijadgfddim/nfdlhophkeijadgfddim/idispatch_windows.go
C:/Users/WIN1/go/src/jobnhbgnnifpodhhpkmf/nfdlhophkeijadgfddim/nfdlhophkeijadgfddim/idispatch.go
C:/Users/WIN1/go/src/jobnhbgnnifpodhhpkmf/nfdlhophkeijadgfddim/nfdlhophkeijadgfddim/guid.go
C:/Users/WIN1/go/src/jobnhbgnnifpodhhpkmf/nfdlhophkeijadgfddim/nfdlhophkeijadgfddim/error_windows.go
C:/Users/WIN1/go/src/jobnhbgnnifpodhhpkmf/nfdlhophkeijadgfddim/nfdlhophkeijadgfddim/error.go
C:/Users/WIN1/go/src/jobnhbgnnifpodhhpkmf/nfdlhophkeijadgfddim/nfdlhophkeijadgfddim/com.go
C:/Users/WIN1/go/src/eicppanmgiigbeblppmd/heapjnpececmcgncmcfb/lkhlnakiocc1jlbjoloc/bdalegiamdedigadphck/types_windows.go
C:/Users/WIN1/go/src/eicppanmgiigbeblppmd/heapjnpececmcgncmcfb/lkhlnakiocc1jlbjoloc/bdalegiamdedigadphck/zsyscall_windows.go
C:/Users/WIN1/go/src/eicppanmgiigbeblppmd/heapjnpececmcgncmcfb/lkhlnakiocc1jlbjoloc/bdalegiamdedigadphck/syscall_windows.go
C:/Users/WIN1/go/src/eicppanmgiigbeblppmd/heapjnpececmcgncmcfb/lkhlnakiocc1jlbjoloc/bdalegiamdedigadphck/syscall.go
C:/Users/WIN1/go/src/eicppanmgiigbeblppmd/heapjnpececmcgncmcfb/lkhlnakiocc1jlbjoloc/bdalegiamdedigadphck/str.go
C:/Users/WIN1/go/src/eicppanmgiigbeblppmd/heapjnpececmcgncmcfb/lkhlnakiocc1jlbjoloc/bdalegiamdedigadphck/security_windows.go
C:/Users/WIN1/go/src/eicppanmgiigbeblppmd/heapjnpececmcgncmcfb/lkhlnakiocc1jlbjoloc/bdalegiamdedigadphck/exec_windows.go
C:/Users/WIN1/go/src/eicppanmgiigbeblppmd/heapjnpececmcgncmcfb/lkhlnakiocc1jlbjoloc/bdalegiamdedigadphck/dll_windows.go
C:/Go/src/log/log.go
```

Project Path

Running yara rule in the binary with the rules found in this public repository https://github.com/Yara-Rules/rules we get the following results.

```
DebuggerException__SetConsoleCtrl .\changed.bin
0x38b97c:$: SetConsoleCtrlHandler
SEH__vectored .\changed.bin
0x38baca:$: AddVectoredExceptionHandler
network_udp_sock .\changed.bin
0x1f2c66:$f1: ws2_32.dll
0x38b85a:$f1: ws2_32.dll
0x1f200e:$c0: WSAStartup
0x2d85d4:$c0: WSAStartup
0x1f12b8:$c3: WSASendTo
0x1f3454:$c4: WSARecvFrom
network_tcp_listen .\changed.bin
0x1f2c66:$f1: ws2_32.dll
0x38b85a:$f1: ws2_32.dll
0x1f4403:$f2: mswsock.dll
0x1efb19:$c1: bind
0x1b2507:$c2: accept
0x1b7597:$c2: accept
0x1b81ce:$c2: accept
0x1b89ae:$c2: accept
0x1fc9e2:$c3: GetAcceptExSockaddrs
0x1f08cf:$c4: AcceptEx
0x1fc9e5:$c4: AcceptEx
0x1f200e:$c5: WSAStartup
0x2d85d4:$c5: WSAStartup
0x1f2004:$c7: WSASocket
0x1f013e:$c10: listen
network_tcp_socket .\changed.bin
0x1f2c66:$f1: ws2_32.dll
0x38b85a:$f1: ws2_32.dll
0x1f2004:$c1: WSASocket
0x1f017a:$c2: socket
0x1f427c:$c2: socket
0x2003f9:$c2: socket
0x202729:$c2: socket
0x2027c5:$c2: socket
0x2027dd:$c2: socket
0x2d8711:$c2: socket
0x1aeb70:$c3: send
0x1b0643:$c3: send
0x1b064b:$c3: send
0x1b1f96:$c3: send
0x1b1fa1:$c3: send
0x1b36fb:$c3: send
0x1b5e90:$c3: send
0x1b7e02:$c3: send
```

network connection

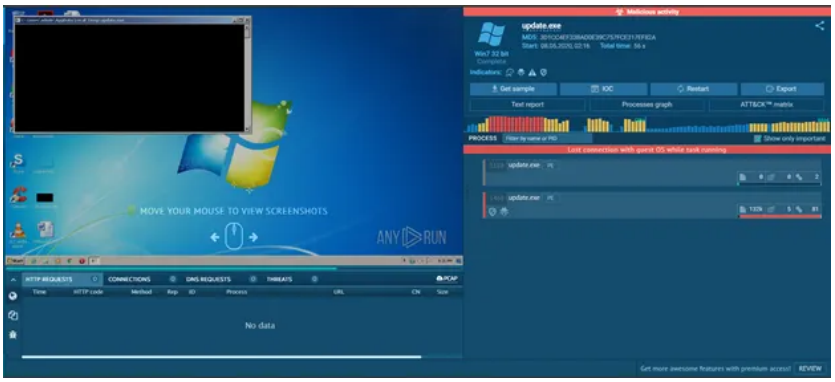
```
0xc65cb:$c6: AB D9 83 1F
0xbdc44:$c7: 19 CD E0 5B
0xc65d2:$c7: 19 CD E0 5B
RijnDael_AES .\changed.bin
0x380b40:$c0: A5 63 63 C6 84 7C 7C F8
RijnDael_AES_CHAR .\changed.bin
0x3798e0:$c0: 63 7C 77 7B F2 6B 6F C5 30 01 67 2B FE D7 AB 76 CA 82 C9 7D FA 59 47 F0 AD D4 A2 AF 9C A4 72 C0
RijnDael_AES_LONG .\changed.bin
0x3798e0:$c0: 63 7C 77 7B F2 6B 6F C5 30 01 67 2B FE D7 AB 76 CA 82 C9 7D FA 59 47 F0 AD D4 A2 AF 9C A4 72 C0
BASE64_table .\changed.bin
0x2073d8:$c0: 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54 55 56 57 58 59 5A 61 62 63 64 65 66 ...
IsPE32 .\changed.bin
IsConsole .\changed.bin
```

AES encryption data

We can see that the malware has possible CnC traffic, and that malware is capable of performing AES encryption.

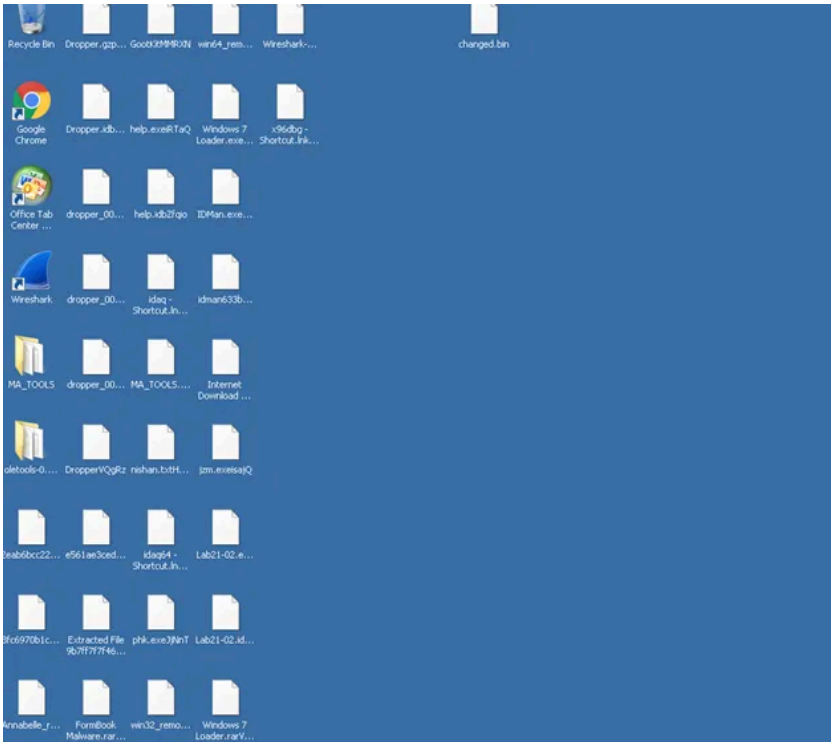
Basic Dynamic Analysis

For basic dynamic analysis, the malware was already run in app.any.run, so i observed the data from there.



Any run sandbox data

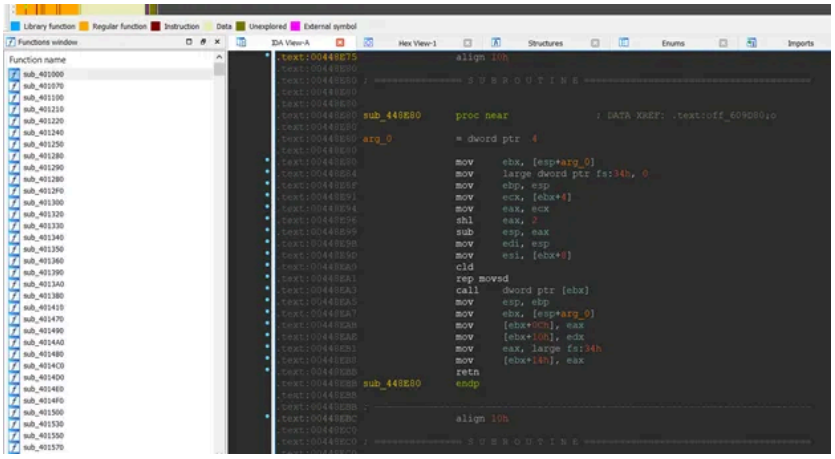
The malware is touted as malicious, but i do not see any network connection. So I ran the malware in a VM, the malware encrypted most of the data, but i could not file any help text, as to how to recover them.



Running in VM.

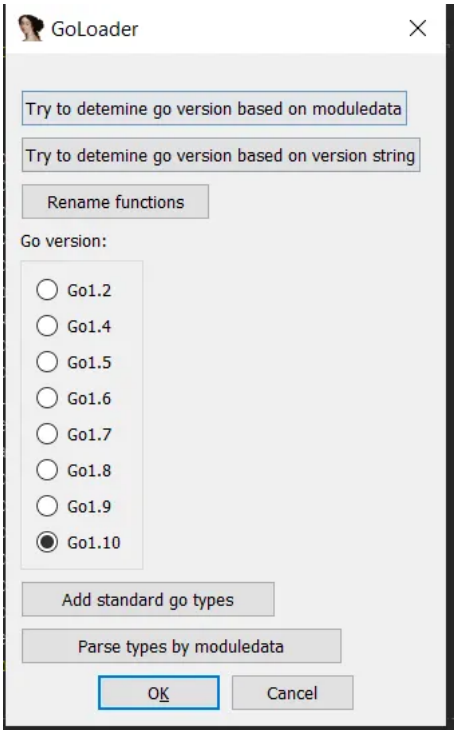
Code Reversing

Unlike regular golang binaries, there were no regular function names. The malware author has thus attempted to obfuscate the various function names.



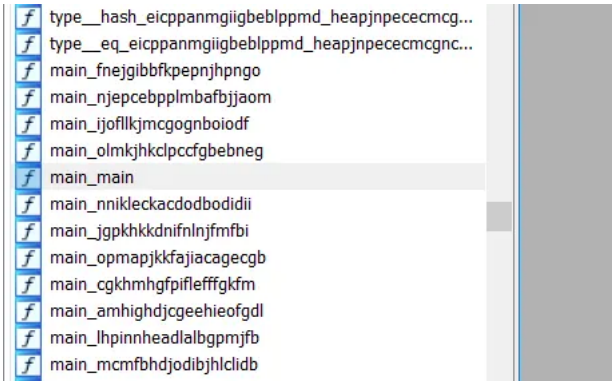
Code Obfuscation

So in order to de-obfuscate the function names a tool called idagolang helper was used https://github.com/sibears/IDAGolangHelper.



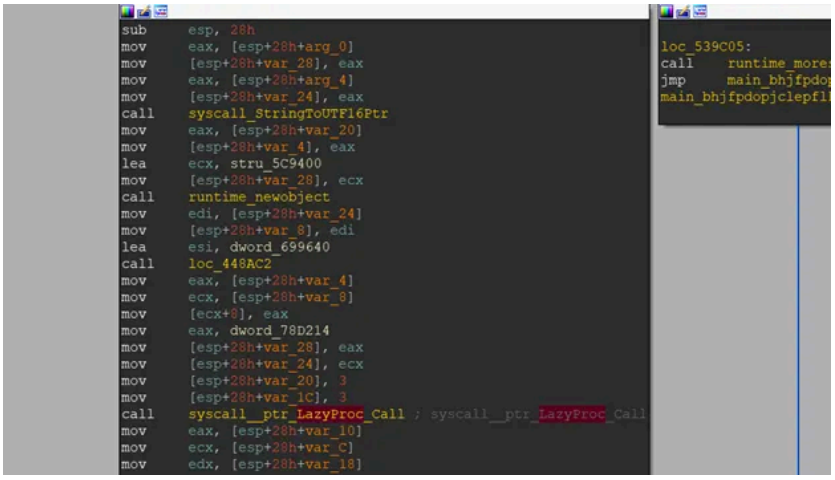
Using IDA Golang Helper tool

Now we getting somewhere, there are a lot of golang builtin functions that we can see. But looking at function name, I was trying to get a hint of the purpose the various main functions , but the function names itself is weird and was most likely obfuscated/randomized to slow down reversing process.



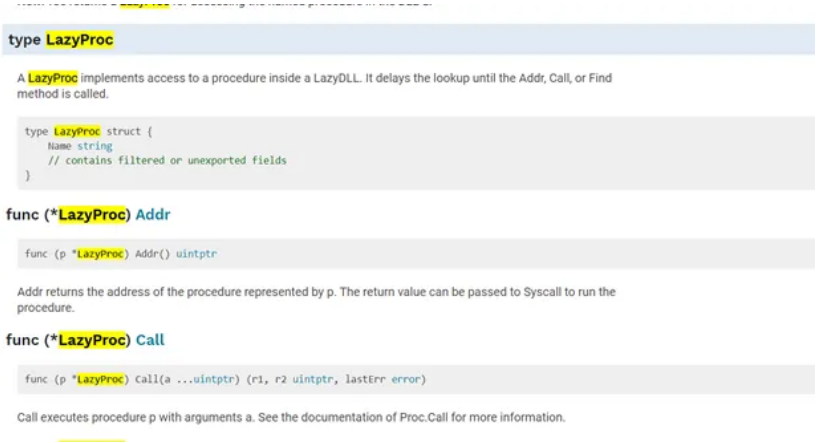
Golang built in functions renamed, but main names are garbage

So I started analyzing the main functions, eventually it led me to the syscall function call from the golangs built in function.



syscall

So after referring to the golangs documentationat https://golang.org/pkg/syscall/?GOOS=windows, it seemed that it was used to call a function from a windows dll file. Here's a snippet.



Golang documentation

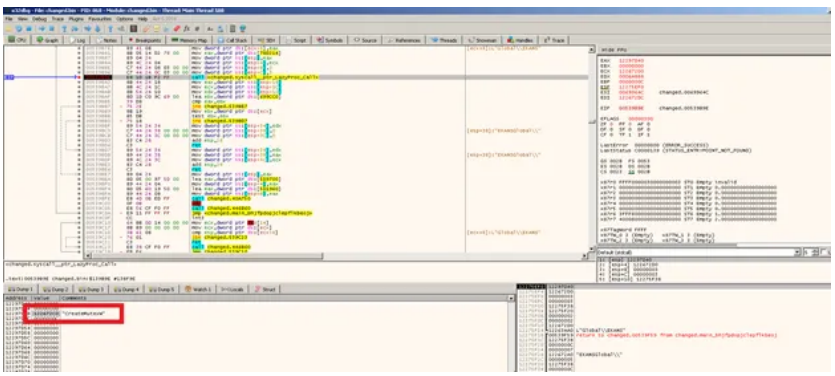
I could not determine the exact function being called via static analysis, so had to run the sample inside a debugger.

Get Nishan Maharjan's stories in your inbox

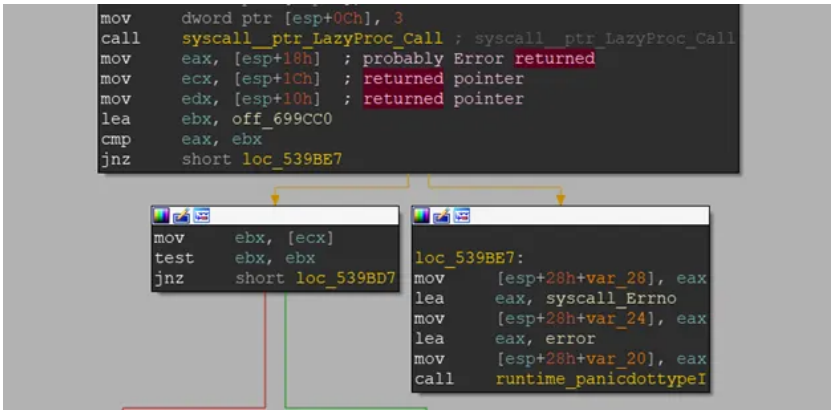
Join Medium for free to get updates from this writer.

Remember me for faster sign in

It seems the function that was being called is CreateMutexW.

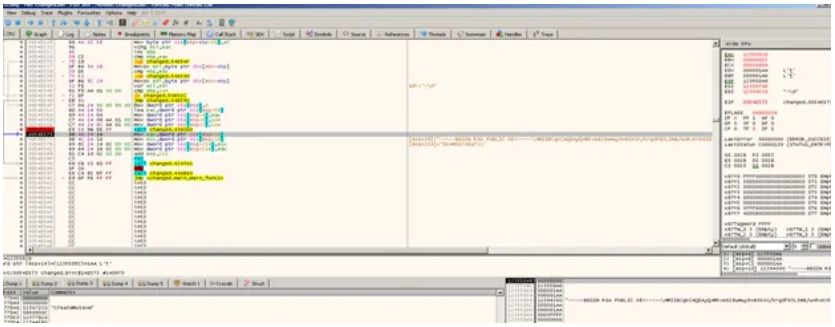


CreateMutexW



By observing that the value esp+0x18 being used in error condition check we can assume that it is the lastErr value and the rest is the returned pointer. Seeing this we can safely assume, that the function main_bhjfpdopjclepflkbeoj is responsible for calling the windows function CreateMutexW and returning its value. Renaming the function to sycall_function. Renamed main_ienocpmdijcihngapkhe to calls_function_Creating_global_and_sycall_create_mutex.

Continuing further analysis, the function at 0054E480 was responsible for key extraction.

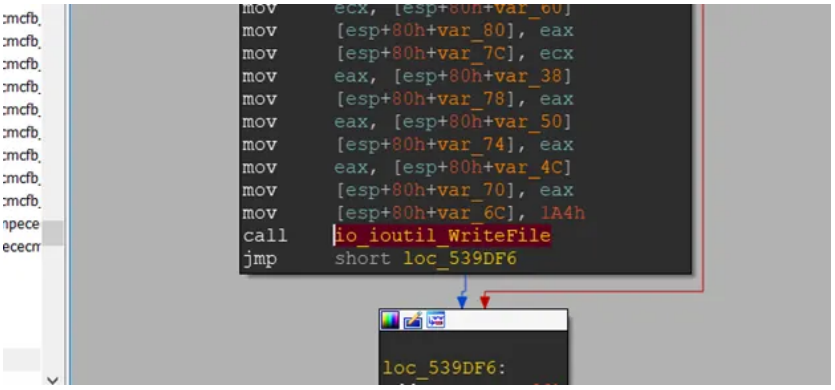


Key extraction

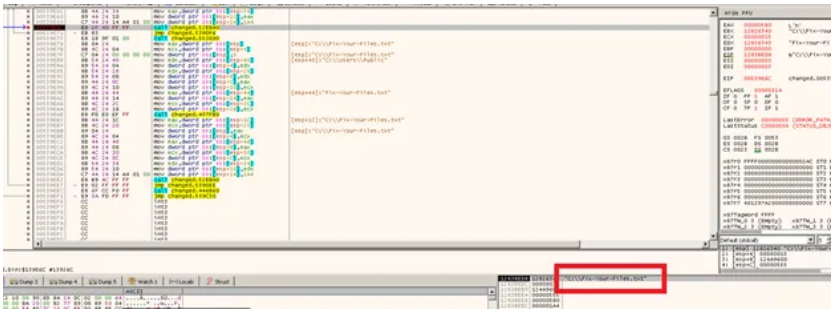
I did not see network communication for getting key, and after running the malware multiple times, the key seems to be the same ***“ — — -BEGIN RSA PUBLIC KEY — — -\nMIIBCgKCAQEAyQ+M5ve829umuy9+BSsUX/krgdF83L3m8/uxRvKX5EZbSh1+buON\nZYr5MjfhrdiOGnrbB1j0Fy31U/uzvWcy7VvK/zcsO/5aAhu — — -END RSA PUBLIC KEY — — -\n”.***

After indepth analysis, it seems function with name pattern main_ffhlhdmhalodcojcaeok_*** seems to be junk function, because the function just converts slice of byte to string or vice versa, and the value is not useful at all.

Digging further, function sub_0053A060 is responsible for looping through each files and encrypting them and finally the function main_amgmhoeegchpnnaljano 00539C30 eventually calls io_ioutil_WriteFile. I thought it could be possibly help file, so i debugged it and trying to find what it actually wrote.



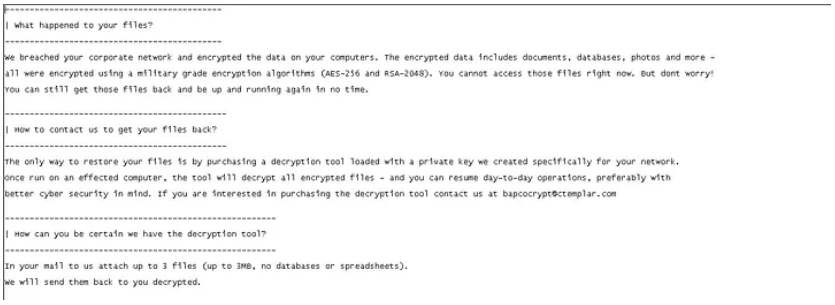
Static Analysis



Debugging

The file is at C:\\Fix-Your-Files.txt. but I could not find the file. The ransomware seems to have failed to write the file. So i thought may be it needed admin privileges to write the help file. I ran the sample again with admin privileges, and now it has written the file.

Press enter or click to view image in full size



```
-----------------------------------------------
| What happened to your files?
-----------------------------------------------
We breached your corporate network and encrypted the data on your computers. The encrypted data includes documents, databases, photos and more -
all were encrypted using a military grade encryption algorithms (AES-256 and RSA-2048). You cannot access those files right now. But dont worry!
You can still get those files back and be up and running again in no time.

-----------------------------------------------
| How to contact us to get your files back?
-----------------------------------------------
The only way to restore your files is by purchasing a decryption tool loaded with a private key we created specifically for your network.
Once run on an effected computer, the tool will decrypt all encrypted files - and you can resume day-to-day operations, preferably with
better cyber security in mind. If you are interested in purchasing the decryption tool contact us at bapcocrypt@ctemplar.com

-------------------------------------------------------------
| How can you be certain we have the decryption tool?
-------------------------------------------------------------
In your mail to us attach up to 3 files (up to 3MB, no databases or spreadsheets).
We will send them back to you decrypted.
```

Help Message

This just seems extremely odd. I do not understand why the author chose to write the help file in such a specific location where the user had to search for it. Other ransomwares seem to have changed the wallpaper itself, or drop the help file in each file. It just seems like a bad choice to keep the help file in a specific location where to write admin privileges was required.

Yara rule for this ransomware:

```
rule SnakeRansomware{meta:Author = "Nishan Maharjan"Description = "A yara rule to catch snake ransomware"Data
```

Source: https://medium.com/@nishanmaharjan17/malware-analysis-snake-ransomware-a0e66f487017