



# Practical Malware Analysis & Triage

## Malware Analysis Report

### WannaHusky\_RansomWare

May 2024 | AnikshaShetty | v1.0

# Table of Contents

<b>Table of Contents</b>	<b>2</b>
<b>Executive Summary</b>	<b>3</b>
<b>High-Level Technical Summary</b>	<b>4</b>
Execution Flow	4
<b>Malware Composition</b>	<b>5</b>
WannaHusky.png	5
ps1.ps1	5
<b>Basic Static Analysis</b>	<b>6</b>
Floss - String Analysis	6
PEStudio	9
<b>Basic Dynamic Analysis</b>	<b>10</b>
Initial Detonation	10
ProcMon Analysis	11
<b>Advanced Static Analysis</b>	<b>14</b>
Encryption routine call	16
<b>Advanced Dynamic Analysis</b>	<b>17</b>
ReadFile	18
Encryption Routine	19
Key Generation	22
Encryption routine conclusion	23
Encryption validation	25
Encode function	26
Change Background	28
Dropped files	29
Command Execution	30
Decryption Routine	31
<b>Indicators of Compromise</b>	<b>32</b>
Network Indicators	32
Host-based Indicators	32
<b>Rules &amp; Signatures</b>	<b>33</b>
Yara Rules	33

# Executive Summary

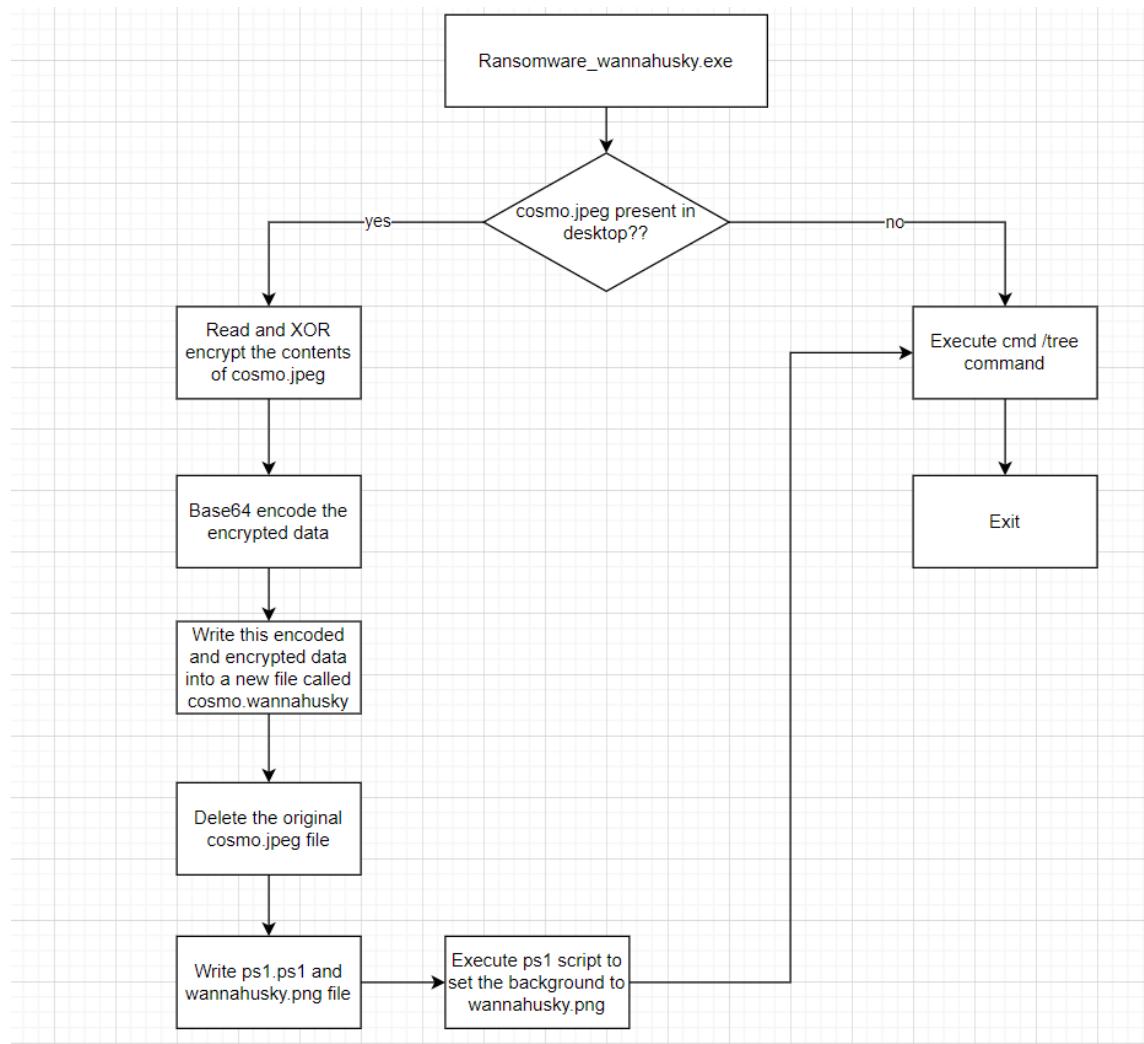
sha256sum	3D35CEBCF40705C23124FDC4656A7F400A316B8E96F1F9E0C187 E82A9D17DCA3
-----------	--

WannaHusky is a nim compiled ransomware sample that encrypts and deletes a particular file(cosmo.jpeg) on the victim's desktop and demands for a ransom payment in order to recover the file. It consists of two parts : encryption routine and the creation of ps1 script to change the background/desktop wallpaper. Symptoms of infection include deletion of cosmo.jpeg file, appearance of cosmo.wannahusky file (base64 encoded encrypted cosmo.jpeg file), wannahusky.png(desktop wallpaper/background image) and a command window with tree view of all directories/sub directories within the system.

# High-Level Technical Summary

Firstly, the binary checks for the presence of cosmo.jpeg file on desktop, if it is not present then exits the program. If it is present, then calls the encryption routine which generates a 16 byte xor key and performs a bit wise xor operation on data in cosmo.jpeg file. The encrypted data is then base64 encoded and written into a new file called cosmo.wannahusky and the original cosmo.jpeg file is deleted off the system. It then creates a new file "Wannahusky.png" and ps1.ps1. The powershell script aims at changing the background/desktop wallpaper to Wannahusky.png. Powershell is executed and lastly the "cmd /tree c:" command is run to list all the directories within the C: drive.

## Execution Flow



# Malware Composition

MalwareName consists of the following components:

FileName	Sha256
ps1.ps1	d6317f374f879cd4e67fb4e9ddc0d283926489f4c0d6cf07d912a247e5cfde99
wannahusky.png	07b3e2937388ac6394a08d35f3a66a80dde38c63b3c459729e2471022961f562

## WannaHusky.png

Background image that contains the ransom note.

## ps1.ps1

Powershell script that is used to set WannaHusky.png as the new background

WannaHusky\_RansomWare

May 2024

v1.0

# Basic Static Analysis

{Screenshots and description about basic static artifacts and methods}

## Floss - String Analysis

Interesting finds : ps1 script being created in desktop, wallpaper is being set to wannahusky.png, cosmo.jpeg is likely renamed as cosmo.wannahusky and some cryptographic functions.

ascii	13	section:.rdata	-	-	-	writeDataImpl
ascii	9	section:.rdata	-	-	-	flushImpl
ascii	4	section:.rdata	-	-	-	data
ascii	3	section:.rdata	-	-	-	pos
ascii	23	section:.rdata	-	-	-	@cannot write to stream
ascii	9	section:.rdata	-	-	-	@tree C:\
ascii	12	section:.rdata	-	-	-	@powershell
ascii	37	section:.rdata	-	-	-	using System.Runtime.InteropServices;
ascii	16	section:.rdata	-	-	-	namespace Win32{
ascii	23	section:.rdata	-	-	-	public class Wallpaper{
ascii	47	section:.rdata	-	-	-	[DllImport("user32.dll", CharSet=CharSet.Auto)]
ascii	101	section:.rdata	-	-	-	static extern int SystemParametersInfo (int uAction , int uParam , string lpszParam , int fuWinIni);
ascii	48	section:.rdata	-	-	-	public static void SetWallpaper(string thePath){
ascii	37	section:.rdata	-	-	-	SystemParametersInfo(20,0,thePath,3);
ascii	14	section:.rdata	-	-	-	add-type \$code
ascii	23	section:.rdata	-	-	-	\$currDir = Get-Location
ascii	31	section:.rdata	-	-	-	\$wallpaper = ".\WANNAHUSKY.PNG"
ascii	58	section:.rdata	-	-	-	\$fullpath = Join-Path -path \$currDir -ChildPath \$wallpaper
ascii	42	section:.rdata	-	-	-	[Win32.Wallpaper]::SetWallpaper(\$fullpath)
ascii	3	section:.rdata	-	-	-	PNG
..						"\$PSN

```
11 oserr.nim
12 raiseOSError
13 @USERPROFILE
14 @unknown OS error
15 @Additional info:
16 ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789-_ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
17 @bcmode.nim(456, 9) ` 
18 ctx.sizeKey() <= len(key)` 
19 @bcmode.nim(455, 9) ` 
20 ctx.sizeBlock() <= len(iv)` 
21 bCryptGenRandom
22 queryProcessCycleTime
23 queryUnbiasedInterruptTime
24 queryIdleProcessorCycleTime
25 coresCount
26 hIntel
27 IOError
28 streams.nim
```

```

-----+
readLineImpl
readDataImpl
peekDataImpl
writeDataImpl
flushImpl
data
@cannot write to stream
@tree C:\ 
@Desktop\ps1.ps1
@powershell
@Desktop\ps1.ps1
@$code = @'
using System.Runtime.InteropServices;
namespace Win32{

    public class Wallpaper{
        [DllImport("user32.dll", CharSet=CharSet.Auto)]
        static extern int SystemParametersInfo (int uAction , int uParam , string lpvParam , int fuWinIni) ;
        public static void SetWallpaper(string thePath){
            SystemParametersInfo(20,0,thePath,3);
        }
    }
}
add-type $code
$currDir = Get-Location
$wallpaper = ".\WANNAHUSKY.PNG"
$fullpath = Join-Path -path $currDir -ChildPath $wallpaper
[Win32.Wallpaper]::SetWallpaper($fullpath)
@Desktop\WANNAHUSKY.png
IHDR
SRGB
gAMA
\tpHYS
.xTPNAmv^

4 ?5SZ
5 %OCZIE
6 I|w7
7 z0Gz
8 1RgOB
9 @Z-B
0 IEND
1 @Desktop\cosmo.WANNAHUSKY
2 @bcmode.nim(503, 9) `len(input) <= len(output)`
3 @COSMO
4 @Desktop\target\cosmo.WANNAHUSKY
5 @Desktop\cosmo.jpeg
6 Unknown error
7 _matherr(): %s in %s(%g, %g)  (retval=%g)
8 Argument domain error (DOMAIN)
9 Argument singularity (SIGN)
0 Overflow range error (OVERFLOW)
1 The result is too small to be represented (UNDERFLOW)
2 Total loss of significance (TLOSS)
3 Partial loss of significance (PLOSS)
4 Mingw-w64 runtime failure:
5 Address %p has no image-section
6     VirtualQuery failed for %d bytes at address %p
7     VirtualProtect failed with code 0x%x
8     Unknown pseudo relocation protocol version %d.
9     Unknown pseudo relocation bit size %d.

```

WannaHusky\_RansomWare  
May 2024  
v1.0

```

writeDataImpl
@tree C:\ 
@Desktop\ps1.ps1
@powershell
@Desktop\ps1.ps1
@$code = @'
using System.Runtime.InteropServices;
namespace Win32{
    public class Wallpaper{
        [DllImport("user32.dll", CharSet=CharSet.Auto)]
        static extern int SystemParametersInfo (int uAction , int uParam , string
lpvParam , int fuWinIni) ;
        public static void SetWallpaper(string thePath){
            SystemParametersInfo(20,0,thePath,3);
        }
    }
}
add-type $code
$currDir = Get-Location
$wallpaper = ".\WANNAHUSKY.PNG"
$fullpath = Join-Path -path $currDir -ChildPath $wallpaper
[Win32.Wallpaper]::SetWallpaper($fullpath)
@Desktop\WANNAHUSKY.png
@Desktop\cosmo.WANNAHUSKY
@bcmode.nim(503, 9)`len(input) <= len(output)`
@COSMO
@Desktop\target\cosmo.WANNAHUSKY
@Desktop\cosmo.jpeg
@USERPROFILE
@Additional info:
ABCDEFHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789-
_ABCDEFHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
@bcmode.nim(456, 9)` 
ctx.sizeKey() <= len(key)` 
@bcmode.nim(455, 9)` 
ctx.sizeBlock() <= len(iv)` 
bCryptGenRandom
queryProcessCycleTime
queryUnbiasedInterruptTime
queryIdleProcessorCycleTime

```

## PEStudio

This is a 32 bit binary, compiled on oct 12 2021

property	value
footprint > sha256	3D35CEBCF40705C23124FDC4656A7F400A316B8E96F1F9E0C187E82A9D17DCA3
first-bytes-hex	4D 5A 00 00 03 00 00 04 00 00 FF FF 00 00 08 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
first-bytes-text	M Z .. @ ..
file > size	412905 bytes
entropy	6.455
signature	n/a
tooling	wait...
file-type	executable
cpu	32-bit
subsystem	console
file-version	n/a
description	n/a
stamps	
compiler-stamp	Tue Oct 12 20:08:25 2021   UTC
debug-stamp	n/a
resource-stamp	n/a
import-stamp	n/a
export-stamp	n/a
names	
file	c:\users\blue\Desktop\ransomware.wannahusky.exe
debug	n/a
export	n/a
version	n/a
manifest	n/a
.NET > module	n/a
certificate > program-name	n/a

Not packed, virtual size and raw size does not have significant difference

property	value	value	value	value	value	value
section	section[0]	section[1]	section[2]	section[3]	section[4]	section[5]
name	.text	.data	.rdata	.bss	.idata	.CRT
footprint > sha256	6299B97342DD96C74485E5C...	076B2D201A61086430CF80C...	923AB031DB80B7FD6EC092...	n/a	288AAE041F27B62B7321813...	FE2509EA26A02F275E6
entropy	6.403	1.477	7.810	n/a	4.936	0.279
file-ratio (84.82%)	14.26 %	0.12 %	9.42 %	n/a	0.50 %	0.12 %
raw-address (begin)	0x00000400	0x0000EA00	0x0000EC00	0x00000000	0x00018400	0x00018C00
raw-address (end)	0x0000EA00	0x0000EC00	0x00018400	0x00000000	0x00018C00	0x00018E00
raw-size (350208 bytes)	0x0000E600 (58880 bytes)	0x00000200 (512 bytes)	0x000009800 (38912 bytes)	0x00000000 (0 bytes)	0x00000800 (2048 bytes)	0x00000200 (512 bytes)
virtual-address	0x00001000	0x00010000	0x00011000	0x0001B000	0x00025000	0x00026000
virtual-size (385182 bytes)	0x0000E404 (58372 bytes)	0x0000009C (156 bytes)	0x0000096D0 (38608 bytes)	0x00000987C (39036 bytes)	0x000007C4 (1988 bytes)	0x00000034 (52 bytes)
characteristics	0x60500060	0xC0600040	0x40600040	0xC0600080	0xC0300040	0xC0300040
write	-	x	-	x	x	x
execute	x	-	-	-	-	-
share	-	-	-	-	-	-
self-modifying	-	-	-	-	-	-
virtual	-	-	-	x	-	-
items						
directory > import	-	-	-	-	0x00025000	-
directory > thread-local-storage	-	-	0x0001A10C	-	-	-
directory > import-address	-	-	-	-	0x00025168	-
base-of-code	0x00001000	-	-	-	-	-
base-of-data	-	0x00010000	-	-	-	-
entry-point	0x000014C0	-	-	-	-	-
thread-local-storage	0x0000E330	-	-	-	-	-
thread-local-storage	0x0000E2E0	-	-	-	-	-

WannaHusky\_RansomWare  
May 2024  
v1.0

# Basic Dynamic Analysis

{Screenshots and description about basic dynamic artifacts and methods}

## Initial Detonation

Running the binary without inet simulation, pops up a black terminal window(cmd??) with some operation happening.

On the desktop new files appear - cosmo.wannahusky, wannahusky.png and ps1 script. After some time, this window and the ps1 script along with the cosmo.jpeg file disappears.

```
File Machine View Input Devices Help
C:\Users\blue\Desktop\Ransomware.wannahusky.exe
└── and64_microsoft-windows-media-streaming-dll_31bf3856ad364e35_10.0.19041.1566_none_c315bd50bc4b72da
    └── f
        ├── and64_microsoft-windows-media-streaming-ps_31bf3856ad364e35_10.0.19041.1_none_fa562f83ace75b88
        ├── and64_microsoft-windows-mediafoundation-msfsrv_31bf3856ad364e35_10.0.19041.1865_none_33422b307865eeba
        └── f
            ├── and64_microsoft-windows-mediafoundation_31bf3856ad364e35_10.0.19041.1865_none_10e43a01c2238eb1
            ├── and64_microsoft-windows-mediacomponent-autoplay_31bf3856ad364e35_10.0.19041.1266_none_8fc08423f52c1606
            ├── and64_microsoft-windows-mediacomponent-core_31bf3856ad364e35_10.0.19041.1566_none_800f26e104636391
            └── f
                ├── and64_microsoft-windows-mediacomponent-logagent_31bf3856ad364e35_10.0.19041.746_none_c939d7042d81ce4
                ├── and64_microsoft-windows-mediacomponent-mls_31bf3856ad364e35_10.0.19041.746_none_4eda1c6d21dd9881
                ├── and64_microsoft-windows-mediacomponent-setup_31bf3856ad364e35_10.0.19041.1266_none_22b99d078bbc3016
                └── f
                    ├── and64_microsoft-windows-mediacomponent-shortcut_31bf3856ad364e35_10.0.19041.1_none_64c27fc7ed12e401
                    ├── and64_microsoft-windows-mediacomponent-skins_31bf3856ad364e35_10.0.19041.2006_none_1e3695b18e994db8
                    └── f
                        ├── and64_microsoft-windows-mediacomponent-vis_31bf3856ad364e35_10.0.19041.1266_none_e5afec1878374111
                        ├── and64_microsoft-windows-mediacomponent-wmasf_31bf3856ad364e35_10.0.19041.1_none_5da6fe23dfc593c7
                        ├── and64_microsoft-windows-mediacomponent-wmerror_31bf3856ad364e35_10.0.19041.1_none_ed4600715653f40d
                        └── and64_microsoft-windows-mediacomponent-wmnetmgr_31bf3856ad364e35_10.0.19041.746_none_253f40e31a7124d4
                    └── f
                        ├── and64_microsoft-windows-mediacomponent-wmpdxdn_31bf3856ad364e35_10.0.19041.1266_none_18fdf6dfdfcdcf40
                        ├── and64_microsoft-windows-mediacomponent-wmpeffects_31bf3856ad364e35_10.0.19041.1266_none_6e13bacd44a30951
                        └── f
                            ├── and64_microsoft-windows-mediacomponent-wmpgs_31bf3856ad364e35_10.0.19041.1_none_642b5d5bdh5acd48
                            ├── and64_microsoft-windows-mediacomponent-wmpshell_31bf3856ad364e35_10.0.19041.1266_none_80fd2d3dfab8926dd
                            └── f
                                ├── and64_microsoft-windows-mediacomponent-wmvcore_31bf3856ad364e35_10.0.19041.1806_none_7bb1760f8b3ee704
                                ├── and64_microsoft-windows-mediacomponent-wwsdk_31bf3856ad364e35_10.0.19041.1_none_5c1c54e5c5f9fc6
                                ├── and64_microsoft-windows-messaging-adm_31bf3856ad364e35_10.0.19041.1_none_ff14ea0301d1ea
                                ├── and64_microsoft-windows-nfaacenc_31bf3856ad364e35_10.0.19041.1_none_8e7de3090639848
                                ├── and64_microsoft-windows-nfasfsrscsnk_31bf3856ad364e35_10.0.19041.1_none_289214b917ff7
                                └── and64_microsoft-windows-nfaucn_31bf3856ad364e35_10.0.19041.1746_none_e77e4c60d41c1b03
                            └── f
                                ├── and64_microsoft-windows-mfc42x.resources_31bf3856ad364e35_10.0.19041.1_en-us_7992c6682e6258c8
                                ├── and64_microsoft-windows-mfc42x_31bf3856ad364e35_10.0.19041.546_none_d10e1541b45e0391
                                └── f
                                    ├── and64_microsoft-windows-nfcore_31bf3856ad364e35_10.0.19041.2006_none_65ae19bd0c581bd6
                                    └── f
                                        ├── and64_microsoft-windows-mfds_31bf3856ad364e35_10.0.19041.1645_none_1a270410d2d089fe
                                        ├── and64_microsoft-windows-mfdudec_31bf3856ad364e35_10.0.19041.1_none_bea308bfe7297b48
                                        ├── and64_microsoft-windows-mfppl_31bf3856ad364e35_10.0.19041.1_none_0f53ce8281c14495
                                        ├── and64_microsoft-windows-nf263enc_31bf3856ad364e35_10.0.19041.1_none_4052cf3d3a53273
                                        └── and64_microsoft-windows-nf264enc_31bf3856ad364e35_10.0.19041.1806_none_fee09de9eaf2b5e
                                    └── f
                                        ├── and64_microsoft-windows-mfmjpegdec_31bf3856ad364e35_10.0.19041.1348_none_8e16cd37892acb5a
                                        └── and64_microsoft-windows-mfmkvsrscsnk_31bf3856ad364e35_10.0.19041.1566_none_bb95e3ad477b3e4d
                                    └── f
                                        ├── and64_microsoft-windows-mfmkvsrscsnk_31bf3856ad364e35_10.0.19041.1566_none_bb95e3ad477b3e4d
                                        └── f
                                            ├── and64_microsoft-windows-mfmkvsrscsnk_31bf3856ad364e35_10.0.19041.1566_none_bb95e3ad477b3e4d
                                            └── f
                                                ├── and64_microsoft-windows-mfmkvsrscsnk_31bf3856ad364e35_10.0.19041.1566_none_bb95e3ad477b3e4d
                                                └── f
                                                    ├── and64_microsoft-windows-mfmkvsrscsnk_31bf3856ad364e35_10.0.19041.1566_none_bb95e3ad477b3e4d
                                                    └── f
                                                        ├── and64_microsoft-windows-mfmkvsrscsnk_31bf3856ad364e35_10.0.19041.1566_none_bb95e3ad477b3e4d
                                                        └── f
                                                            ├── and64_microsoft-windows-mfmkvsrscsnk_31bf3856ad364e35_10.0.19041.1566_none_bb95e3ad477b3e4d
                                                            └── f
                                                                ├── and64_microsoft-windows-mfmkvsrscsnk_31bf3856ad364e35_10.0.19041.1566_none_bb95e3ad477b3e4d
                                                                └── f
                                                                    ├── and64_microsoft-windows-mfmkvsrscsnk_31bf3856ad364e35_10.0.19041.1566_none_bb95e3ad477b3e4d
                                                                    └── f
                                                                        ├── and64_microsoft-windows-mfmkvsrscsnk_31bf3856ad364e35_10.0.19041.1566_none_bb95e3ad477b3e4d
                                                                        └── f
                                                                            ├── and64_microsoft-windows-mfmkvsrscsnk_31bf3856ad364e35_10.0.19041.1566_none_bb95e3ad477b3e4d
                                                                            └── f
                                                                                ├── and64_microsoft-windows-mfmkvsrscsnk_31bf3856ad364e35_10.0.19041.1566_none_bb95e3ad477b3e4d
                                                                                └── f
                                                                                    ├── and64_microsoft-windows-mfmkvsrscsnk_31bf3856ad364e35_10.0.19041.1566_none_bb95e3ad477b3e4d
                                                                                    └── f
                                                                                        ├── and64_microsoft-windows-mfmkvsrscsnk_31bf3856ad364e35_10.0.19041.1566_none_bb95e3ad477b3e4d
                                                                                        └── f
                                                                                            ├── and64_microsoft-windows-mfmkvsrscsnk_31bf3856ad364e35_10.0.19041.1566_none_bb95e3ad477b3e4d
                                                                                            └── f
                                                                                                ├── and64_microsoft-windows-mfmkvsrscsnk_31bf3856ad364e35_10.0.19041.1566_none_bb95e3ad477b3e4d
                                                                                                └── f
                                                                                                    ├── and64_microsoft-windows-mfmkvsrscsnk_31bf3856ad364e35_10.0.19041.1566_none_bb95e3ad477b3e4d
                                                                                                    └── f
                                                                                                        ├── and64_microsoft-windows-mfmkvsrscsnk_31bf3856ad364e35_10.0.19041.1566_none_bb95e3ad477b3e4d
                                                                                                        └── f
                                                                                                            ├── and64_microsoft-windows-mfmkvsrscsnk_31bf3856ad364e35_10.0.19041.1566_none_bb95e3ad477b3e4d
                                                                                                            └── f
                                                                                                                ├── and64_microsoft-windows-mfmkvsrscsnk_31bf3856ad364e35_10.0.19041.1566_none_bb95e3ad477b3e4d
                                                                                                                └── f
                                                                                                                    ├── and64_microsoft-windows-mfmkvsrscsnk_31bf3856ad364e35_10.0.19041.1566_none_bb95e3ad477b3e4d
                                                                                                                    └── f
................................................................
```

WannaHusky\_RansomWare  
May 2024  
v1.0

## ProcMon Analysis

Looking at the procmon logs, we can see cosmo.jpeg file is first read and then written into cosmo.wannahusky and then there is a powershell script that gets created in desktop which is then executed using cmd and finally cmd /tree is executed.

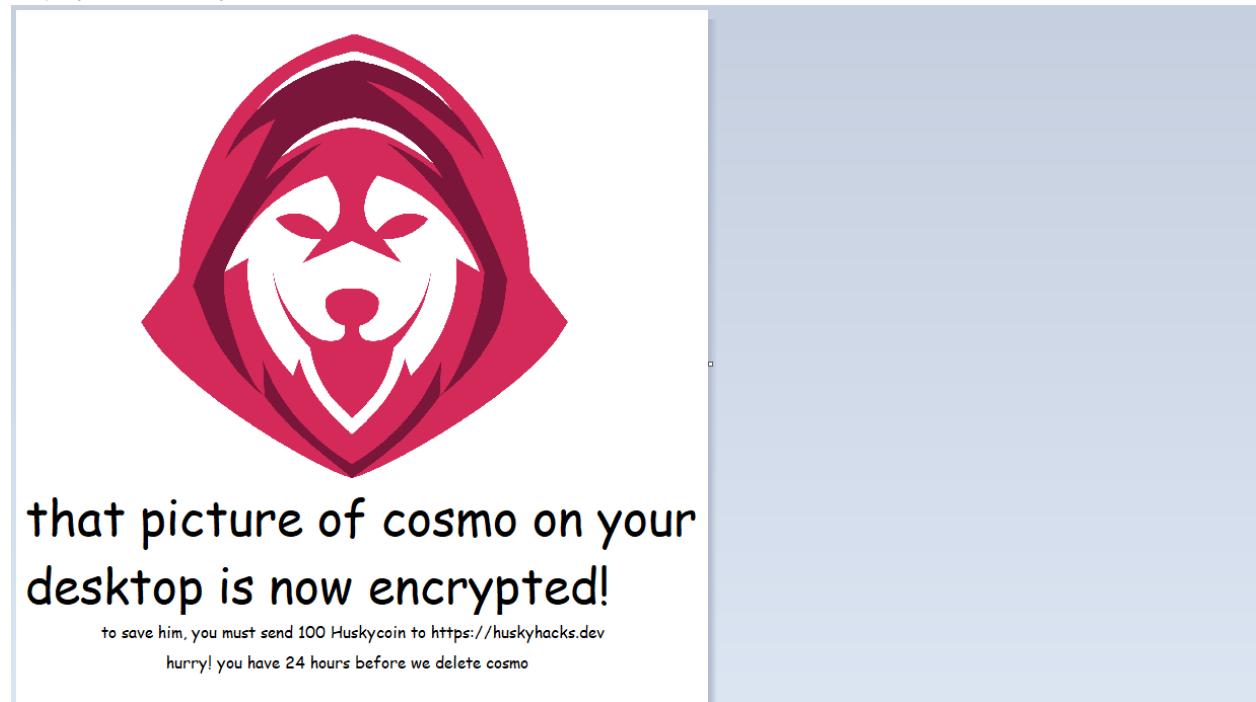
### New file creation logs :

Time o...	Process Name	PID	Operation	Path	Result
6:40:28....	Ransomware.w...	5412	CreateFile	C:\Users\blue\Desktop\cosmo.jpeg	SUCCESS
6:40:28....	Ransomware.w...	5412	QueryStandard...	C:\Users\blue\Desktop\cosmo.jpeg	SUCCESS
6:40:28....	Ransomware.w...	5412	ReadFile	C:\Users\blue\Desktop\cosmo.jpeg	SUCCESS
6:40:28....	Ransomware.w...	5412	ReadFile	C:\Users\blue\Desktop\cosmo.jpeg	SUCCESS
6:40:28....	Ransomware.w...	5412	ReadFile	C:\Users\blue\Desktop\cosmo.jpeg	END OF FIL
6:40:28....	Ransomware.w...	5412	ReadFile	C:\Windows\SysWOW64\msvcrt.dll	SUCCESS
6:40:28....	Ransomware.w...	5412	CloseFile	C:\Users\blue\Desktop\cosmo.jpeg	SUCCESS
6:40:28....	Ransomware.w...	5412	CreateFile	C:\Users\blue\Desktop\cosmo.WANNAHUSKY	SUCCESS
6:40:28....	Ransomware.w...	5412	WriteFile	C:\Users\blue\Desktop\cosmo.WANNAHUSKY	SUCCESS
6:40:28....	Ransomware.w...	5412	WriteFile	C:\Users\blue\Desktop\cosmo.WANNAHUSKY	SUCCESS
6:40:28....	Ransomware.w...	5412	WriteFile	C:\Users\blue\Desktop\cosmo.WANNAHUSKY	SUCCESS
6:40:28....	Ransomware.w...	5412	WriteFile	C:\Users\blue\Desktop\cosmo.WANNAHUSKY	SUCCESS
6:40:28....	Ransomware.w...	5412	WriteFile	C:\Users\blue\Desktop\cosmo.WANNAHUSKY	SUCCESS
6:40:28....	Ransomware.w...	5412	WriteFile	C:\Users\blue\Desktop\cosmo.WANNAHUSKY	SUCCESS
6:40:28....	Ransomware.w...	5412	WriteFile	C:\Users\blue\Desktop\cosmo.WANNAHUSKY	SUCCESS
6:40:28....	Ransomware.w...	5412	WriteFile	C:\Users\blue\Desktop\cosmo.WANNAHUSKY	SUCCESS
6:40:28....	Ransomware.w...	5412	WriteFile	C:\Users\blue\Desktop\cosmo.WANNAHUSKY	SUCCESS
6:40:28....	Ransomware.w...	5412	WriteFile	C:\Users\blue\Desktop\cosmo.WANNAHUSKY	SUCCESS
6:40:28....	Ransomware.w...	5412	WriteFile	C:\Users\blue\Desktop\cosmo.WANNAHUSKY	SUCCESS
6:40:28....	Ransomware.w...	5412	WriteFile	C:\Users\blue\Desktop\cosmo.WANNAHUSKY	SUCCESS
6:40:28....	Ransomware.w...	5412	WriteFile	C:\Users\blue\Desktop\cosmo.WANNAHUSKY	SUCCESS
6:40:28....	Ransomware.w...	5412	WriteFile	C:\Users\blue\Desktop\cosmo.WANNAHUSKY	SUCCESS
6:40:28....	Ransomware.w...	5412	WriteFile	C:\Users\blue\Desktop\cosmo.WANNAHUSKY	SUCCESS
6:40:28....	Ransomware.w...	5412	WriteFile	C:\Users\blue\Desktop\cosmo.WANNAHUSKY	SUCCESS
6:40:28....	Ransomware.w...	5412	CloseFile	C:\Users\blue\Desktop\cosmo.WANNAHUSKY	SUCCESS
6:40:28....	Ransomware.w...	5412	ReadFile	C:\Users\blue\Desktop\Ransomware.wannahusky.exe	SUCCESS
	Ransomware.w...	5412	WriteFile	C:\Users\blue\Desktop\cosmo.WANNAHUSKY	SUCCESS
	Ransomware.w...	5412	WriteFile	C:\Users\blue\Desktop\cosmo.WANNAHUSKY	SUCCESS
	Ransomware.w...	5412	CloseFile	C:\Users\blue\Desktop\cosmo.WANNAHUSKY	SUCCESS
	Ransomware.w...	5412	ReadFile	C:\Users\blue\Desktop\Ransomware.wannahusky.exe	SUCCESS
	Ransomware.w...	5412	CreateFile	C:\Users\blue\Desktop\WANNAHUSKY.png	SUCCESS
	Ransomware.w...	5412	WriteFile	C:\Users\blue\Desktop\WANNAHUSKY.png	SUCCESS
	Ransomware.w...	5412	WriteFile	C:\Users\blue\Desktop\WANNAHUSKY.png	SUCCESS
	Ransomware.w...	5412	CloseFile	C:\Users\blue\Desktop\WANNAHUSKY.png	SUCCESS
	Ransomware.w...	5412	CreateFile	C:\Users\blue\Desktop\cosmo.jpeg	SUCCESS
	Ransomware.w...	5412	QueryAttributeT...	C:\Users\blue\Desktop\cosmo.jpeg	SUCCESS
	Ransomware.w...	5412	CloseFile	C:\Users\blue\Desktop\cosmo.jpeg	SUCCESS
	Ransomware.w...	5412	CreateFile	C:\Users\blue\Desktop\ps1.ps1	SUCCESS
	Ransomware.w...	5412	WriteFile	C:\Users\blue\Desktop\ps1.ps1	SUCCESS
	Ransomware.w...	5412	CloseFile	C:\Users\blue\Desktop\ps1.ps1	SUCCESS
	Ransomware.w...	5412	ReadFile	C:\Windows\SysWOW64\msvcrt.dll	SUCCESS
	Ransomware.w...	5412	CreateFile	C:\Windows\SysWOW64\cmd.exe	SUCCESS
	Ransomware.w...	5412	QueryBasicInfor...	C:\Windows\SysWOW64\cmd.exe	SUCCESS
	Ransomware.w...	5412	CloseFile	C:\Windows\SysWOW64\cmd.exe	SUCCESS

## Process logs showing powershell and cmd execution :

Time o...	Process Name	PID	Operation	Path	Result	Detail
6:40:28...	Ransomware.w...	5412	Process Start	C:\Windows\System32\Conhost.exe	SUCCESS	Parent PID: 3164, Command line: "C:\Users\blue\Desktop\Ransomware.wannahusky.exe", Current directo
6:40:28...	Ransomware.w...	5412	Process Create	C:\Windows\SysWOW64\cmd.exe	SUCCESS	PID: 5132, Command line: '??(C:\Windows\system32)\conhost.exe 0xffffffff -ForceV1
6:40:28...	Ransomware.w...	5412	Process Create	C:\Windows\SysWOW64\cmd.exe	SUCCESS	PID: 5392, Command line: C:\Windows\system32\cmd.exe /c powershell C:\Users\blue\Desktop\ps1.ps1
6:40:30...	Ransomware.w...	5412	Process Create	C:\Windows\SysWOW64\cmd.exe	SUCCESS	PID: 1180, Command line: C:\Windows\system32\cmd.exe /c tree C:\
6:40:58...	Ransomware.w...	5412	Process Exit		SUCCESS	Exit Status: 0, User Time: 0.1875000 seconds, Kernel Time: 0.0000000 seconds, Private Bytes: 10,846,208, R

Wannahusky.png ---> Ransom note stating the file cosmo.jpg has been encrypted and we need to pay 100 huskycoins to recover the file



WannaHusky\_RansomWare  
May 2024  
v1.0



# Advanced Static Analysis

{Screenshots and description about findings during advanced static analysis}

Loading the binary in cutter :

Main program of the binary starts here :

```
[0x0040e052]
@NimMainModule@0();
0x0040e052    push    ebp
0x0040e053    mov     ecx, @TM__njFKfyRiYvomtvTKocFwDw_2@0 ; 0x40d8a1
0x0040e058    mov     ebp, esp
0x0040e05a    sub     esp, 8
0x0040e05d    call    @nimRegisterGlobalMarker@4 ; sym._nimRegisterGlobalMarker_4
0x0040e062    mov     ecx, @TM__njFKfyRiYvomtvTKocFwDw_3@0 ; 0x40d894
0x0040e067    call    @nimRegisterGlobalMarker@4 ; sym._nimRegisterGlobalMarker_4
0x0040e06c    call    @nosgetCurrentDir@0 ; sym._nosgetCurrentDir_0
0x0040e071    mov     edx, eax
0x0040e073    mov     eax, 0x424860 ; 'HB'
0x0040e078    call    _asgnRef ; sym._asgnRef_0x40d830
0x0040e07d    call    @nosgetHomeDir@0 ; sym._nosgetHomeDir_0
0x0040e082    mov     edx, eax
0x0040e084    mov     eax, 0x424870 ; 'pHB'
0x0040e089    call    _asgnRef ; sym._asgnRef_0x40d830
0x0040e08e    call    @wannaHusky__4@0 ; sym._wannaHusky__4JhDTDCSrwYIQ19bJbLaL2w_0
0x0040e093    call    @changeBackground__4_2@0 ; sym._changeBackground__4JhDTDCSrwYIQ19bJbLaL2w...
0x0040e098    mov     ecx, data.00411e40 ; 0x411e40
0x0040e09d    leave
0x0040e09e    jmp     @nosexecShellCmd@4 ; sym._nosexecShellCmd_4
```

Interesting function call here is the wannaHusky --> this function contains the functionality to find and read cosmo.jpeg and the encryption routine.

Looking into the wannacry function in depth, it initially constructs the path for cosmo.jpeg and then passes this path to the readfile function which reads the contents of cosmo.jpeg file.

```

Graph(sym._wannaHusky_4jhDTDCSrvYI0Q19bjbLaL2w_0)
void sym._wannaHusky_4jhDTDCSrvYI0Q19bjbLaL2w_0();
    . . .
0x0040da52    test    eax, eax
0x0040da54    je      0x40da5b
    . . .
[0x0040da56]
0x0040da56    mov     ecx, dword [eax]
0x0040da58    add     ecx, 0x1f ; 31
    . . .
[0x0040da5b]
0x0040da5b    call    @rawNewString@4 ; sym._rawNewString_4
0x0040da60    mov     edx, dword [0x424870]
0x0040da66    call    _appendString ; sym._appendString_0x40d8cf
0x0040da68    mov     edx, data.0041a0a0 ; 0x41a0a0
0x0040da70    call    _appendString.part.0 ; sym._appendString.part.0_0x40d8ae
0x0040da75    mov     ecx, dword [var_518h]
0x0040da7b    call    @readFile_4@4 ; sym._readFile_4@4 ; sym._readFile_4@4
0x0040da80    mov     ecx, data.0041a078, 0x41a078
0x0040da85    mov     esi, eax
0x0040da87    call    @copyString@4 ; sym._copyString_4
0x0040da8c    lea     edx, [var_46ch]
0x0040da92    mov     ecx, 0x8a ; 138
0x0040da97    mov     edi, edx
0x0040da99    mov     edx, dword [var_51ch], eax
0x0040da9f    lea     edx, [var_244h]
0x0040daa5    xor     eax, eax
0x0040daa7    rep    stosd dword es:[edi], eax
0x0040daa9    mov     edi, edx
0x0040daab    mov     ecx, 0x8a ; 138
0x0040dab0    lea     edx, [var_4f4h]
0x0040dab6    rep    stosd dword es:[edi], eax
0x0040dab8    mov     edi, edx
0x0040daba    mov     ecx, 8
0x0040dabf    lea     edx, [var_504h]
0x0040dac5    rep    stosd dword es:[edi], eax
0x0040dac7    mov     ecx, 4
0x0040dacc    mov     edi, edx
    . . .

```

Reads the cosmo.jpeg file

Then there is an encryption routine call, after which the burn memory call is made to likely clear the encryption key stored in memory.

```

void sym._wannaHusky_4jhDTDCSrvYI0Q19bjbLaL2w_0();
    . . .
0x0040dcf2    cmp    dword [var_514h], 0
0x0040dcf9    mov    ecx, ebx
0x0040dcfb    lea    esi, [edi + 8]
0x0040dcfe    je     0x40dd08
    . . .
[0x0040dd00]
0x0040dd00    mov    edi, dword [var_514h]
0x0040dd06    mov    ecx, dword [edi]
    . . .
[0x0040dd08]
0x0040dd08    mov    dword [var_568h], eax ; int32_t arg_4h
0x0040dd0c    mov    edx, dword [var_524h]
0x0040dd12    lea    eax, [var_4fch]
0x0040dd18    mov    dword [esp], esi ; int32_t arg_8h
0x0040dd1b    call   _encrypt_dcoBdmUaaCC9cnR23efFxSLAbcmode ; sym._encrypt_dcoBdmUaaCC9cnR23...
0x0040dd20    mov    edx, 0x200, 052
0x0040dd25    lea    ecx, [var_4fch]
0x0040dd2b    call   @burnMem_4F@8 ; sym._burnMem_4FZHyz34TgxTmMy6XY9c0Sg_8
    . . .
0x0040dd30    lea    eax, [var_564h]
0x0040dd36    mov    dword [var_564h], 0x10 ; 16 ; int32_t arg_8h
0x0040dd3e    lea    edx, [var_4f4h]
0x0040dd44    mov    dword [var_568h], eax ; int32_t arg_4h
0x0040dd48    lea    ecx, [var_244h]
0x0040dd4e    mov    dword [esp], 0x20 ; 32 ; int32_t arg_ch
0x0040dd55    call   @init_QeKCvRTxwnkv4EgDHkgXYA@20 ; sym._init_QeKCvRTxwnkv4EgDHkgXYA_20
0x0040dd5a    mov    edx, ebx
0x0040dd5c    sub    esp, 0xc
0x0040dd5f    cmp    dword [var_520h], 0
0x0040dd66    je     0x40dd70
    . . .
[0x0040dd68]
0x0040dd68    mov    eax, dword [var_520h]
0x0040dd6e    mov    edx, dword [eax]
    . . .

```

This is the function call that handles the encryption routine.

Likely burns the key value in memory

## Encryption routine call

Looks like it performs a bit wise xor operation on each byte of data. Lets get into debugging now.

```
uint32_t encrypt_dcoBdmUaaCC9cnR23eFxSLAbcmode (int32_t arg_4h, uint32_t arg_8h) {
    int32_t var_38h;
    int32_t var_34h;
    int32_t var_28h;
    int32_t var_24h;
    uint32_t var_20h;
    int32_t var_10h;
    esi = eax;
    var_24h = edx;
    var_20h = ecx;
    if (ecx > arg_8h) {
        ecx = data_0041a040;
        @failedAssertImpl_W9cjVocn1tjhW7p7xohJj6A@4 ();
    }
    ebx = *((esi + 0x224));
    edi = 0;
    do {
        if (edi >= var_20h) {
            *((esi + 0x224)) = ebx;
            esp = &var_10h;
            return;
        }
        if (ebx == 0) {
            edx = esi + 0x204;
            eax = esi + 0x1e4;
            ecx = esi;
            edx = eax;
            @encrypt_py6wg79aBw8iTzUm11Z7J0A@20 (0x20, 0x20, edx);
            edx = 0x20;
            ecx = esi + 0x1e4;
            @inc128_vRz5m42fv3XKwSYgATX55Q@8 ();
        }
        if (edi >= arg_8h) {
            eax = arg_8h;
            edx = eax - 1;
            _raiseIndexError2 (edi, edx);
        }
        if (edi >= var_20h) {
            eax = var_20h;
            edx = eax - 1;
            _raiseIndexError2 (edi, edx);
        }
        if (ebx > 0x1f) {
            _raiseIndexError2 (ebx, 0x1f);
        }
        eax = var_24h;
        dl = *((eax + edi));
        eax = arg_4h;
        dl ^= *((esi + ebx + 0x204));
        *((eax + edi)) = dl;
        eax = edi + 1;
        edi ^= eax;
        if (ebx < 0x1f) {
            if (eax >= 0) {
                goto label_0;
            }
            _raiseOverflow (eax);
            eax = var_28h;
        }
    }
label_0:
    ebx++;
    edi = eax;
    ebx &= 0xf;
} while (1);
}
```

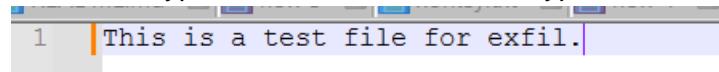
unknown encrypt function that does some shift right and left and xor operation

Bit wise xor operation on each byte of data. esi + ebx + 0x204 should be the xor key.  
resulted encrypted data is stored in eax + edi

# Advanced Dynamic Analysis

{Screenshots and description about advanced dynamic artifacts and methods}

Created a small text file "This is a test file for exfil" and named it as cosmo.jpeg to reduce the time of encryption and see if we can decrypt the data.



Main entry

The screenshot shows a debugger interface with two main windows. The top window displays assembly code for the 'main entry' function, with the instruction at address 0040E052 highlighted in red and labeled 'main entry'. The assembly code includes various calls to 'ransomware.wannahusky - copy' functions. The bottom window shows a memory dump with columns for Address, Hex, and ASCII. The ASCII column shows the raw bytes of the memory dump, which appear to be encrypted data.

Stepping in through the wannacry function and going over each instruction.

WannaHusky\_RansomWare  
May 2024  
v1.0

## ReadFile

Initially there is a call to fileread which reads and stores the content of the data in memory.

The screenshot shows the assembly code and memory dump of the ransomware's process. The assembly code at address 0040DA80 contains a call instruction to the function `copy.read_cosmo_jpeg`. The memory dump window shows the file content "This is a test file for exfil.....".

```

0040DA33 E8 97FFFF call <ransomware.wannahusky - copy.append_String>
0040DA38 8885 ECFAFFFF mov eax,dword ptr ss:[ebp-514]
0040DA3E BA C8A04100 mov edx,ransomware.wannahusky - copy.41A0C8
0040DA43 E8 66FEFFFF call <ransomware.wannahusky - copy.encrypt_!>
0040DA48 A1 70484200 mov eax,dword ptr ds:[424870]
0040DA4D B9 1F000000 mov ecx,1F
0040DA52 85C0 test eax,eax
0040DA54 74 05 je ransomware.wannahusky - copy.40DA5B
0040DA56 8B08 mov ecx,dword ptr ds:[eax]
0040DA58 83C1 1F add ecx,1F
0040DA5B E8 0375FFFF call <ransomware.wannahusky - copy.raw_new_string>
0040DA60 8B15 70484200 mov edx,dword ptr ds:[424870]
0040DA66 E8 64FEFFFF call <ransomware.wannahusky - copy.append_String>
0040DA68 BA A0A04100 mov edx,ransomware.wannahusky - copy.41A0A0
0040DA70 E8 39FEFFFF call <ransomware.wannahusky - copy.encrypt_!>
0040DA75 88D8 ECFAFFFF mov eax,dword ptr ss:[ebp-514]
0040DA76 E8 8048FFFF call <ransomware.wannahusky - copy.read_cosmo_jpeg>
0040DA78 B9 78A04100 mov ecx,ransomware.wannahusky - copy.41A078
0040DA85 89C6 mov esi,eax
0040DA87 E8 2185FFFF call ransomware.wannahusky - copy.405FAD
0040DA8C 8D95 98FBFFFF lea edx,dword ptr ss:[ebp-468]
0040DA92 B9 8A000000 mov ecx,8A
0040DA97 89D7 mov edi,edx
0040DA99 8985 E8FAFFFF mov dword ptr ss:[ebp-518],eax
0040DA9F 8D95 C0FDFFFF lea edx,dword ptr ss:[ebp-240]
0040DAA5 31C0 xor eax,eax
0040DAA7 C3 .ENDP

ecx=EC229A8A
ransomware.wannahusky - copy.exe:$DA80 #CE80
.text:0040DA80 ransomware.wannahusky - copy.exe:$DA80 #CE80

```

Registers:

- EAX: 009DB448
- EBX: 00000000
- ECX: EC229A8A
- DX: 00000000
- BP: 0066F8B8
- ESP: 0066F930
- EST: 00000037
- EDI: 00A900F4
- EIP: 0040DA80

Stack Dump:

009DB448	LE 00 00 00	LE 00 00 00	54 68 69 73	20 69 73 20	... This is a test file for exfil.....
009DB449	61 20 74 65	73 74 20 66	69 66 65 73	20 66 67 72	...
009DB44A	65 78 66 69	6C 2E 00 00	00 00 00 00	00 00 00 00	...
009DB44B	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	...
009DB44C	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	...
009DB44D	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	...
009DB44E	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	...
009DB44F	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	...
009DB450	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	...

Call Stack:

- 1: [esp+4] 00000000 00
- 2: [esp+8] 00100000 00
- 3: [esp+C] 00100000 00
- 4: [esp+10] 0010A1E8 00
- 5: [esp+14] 00000000 00

Registers (dumped):

0066F930	0066F9E0
0066F934	00000000
0066F938	00100000
0066F93C	00000000
0066F940	00000000
0066F944	00000008
0066F948	0000007F
0066F94C	00100000
0066F950	00000000
0066F954	0010A730
0066F958	00000000
0066F95C	00000000
0066F960	00000000
0066F964	00000000
0066F968	00000000
0066F96C	00104738

Process Monitor Log:

Time of Day	Process Name	PID	Operation	Path	Result	Detail
5.7484093 AM	Ransomware.w...	804	ReadFile	C:\Windows\SysWOW64\msvcr.dll	SUCCESS	Offset 361,472, Length: 28,672, I/O Flags: Non-cached, R...
1.3781749 AM	Ransomware.w...	804	Thread Create		SUCCESS	Thread ID: 5168
6.2196212 AM	Ransomware.w...	804	ReadFile	C:\Windows\SysWOW64\msvcr.dll	SUCCESS	Offset 484,352, Length: 32,768, I/O Flags: Non-cached, R...
6.2228459 AM	Ransomware.w...	804	ReadFile	C:\Windows\SysWOW64\msvcr.dll	SUCCESS	Offset 517,120, Length: 28,672, I/O Flags: Non-cached, R...
6.2234453 AM	Ransomware.w...	804	ReadFile	C:\Windows\SysWOW64\msvcr.dll	SUCCESS	Offset 308,224, Length: 32,768, I/O Flags: Non-cached, R...
6.2245634 AM	Ransomware.w...	804	CreateFile	C:\Users\blue\Desktop\cosmo.jpeg	SUCCESS	Desired Access: Generic Read, Disposition: Open, Opti...
6.2246335 AM	Ransomware.w...	804	QueryStandardI...	C:\Users\blue\Desktop\cosmo.jpeg	SUCCESS	AllocationSize: 32, EndOfFile: 30, NumberOfLinks: 1, De...
6.2247910 AM	Ransomware.w...	804	ReadFile	C:\Windows\SysWOW64\msvcr.dll	SI.CESS	Offset 283,648, Length: 24,576, I/O Flags: Non-cached, R...
6.2253411 AM	Ransomware.w...	804	ReadFile	C:\Users\blue\Desktop\cosmo.jpeg	SUCCESS	Offset 0, Length: 30, Priority: Normal
6.2254241 AM	Ransomware.w...	804	ReadFile	C:\Users\blue\Desktop\cosmo.jpeg	END OF FILE	Offset 30, Length: 4,096
6.2254633 AM	Ransomware.w...	804	CloseFile	C:\Users\blue\Desktop\cosmo.jpeg	SUCCESS	

## Encryption Routine

Skipping to the memory location of encryption call "0x0040d8d6" and adding a break point and stepping over each instruction within the call.

```

0040D8D5 EB D9 jmp <ransomware.wannahusky - copy.encrypt>
0040D8D5 C3 ret
0040D8D6 55 push ebp
0040D8D7 89E5 mov ebp,esp
0040D8D9 57 push edi
0040D8DA 56 push esi
0040D8DB 89C6 mov esi,eax
0040D8DD 53 push ebx
0040D8DE 83EC 2C sub esp,2C
0040D8E1 8955 E0 mov dword ptr ss:[ebp-20],edx
0040D8E4 894D E4 mov dword ptr ss:[ebp-1C],ecx
0040D8E7 3B4D 0C cmp ecx,dword ptr ss:[ebp+C]
0040D8EA 7E 0A jle ransomware.wannahusky - copy.40D8F6
0040D8EC B9 A04100 mov ecx,ransomware.wannahusky - copy.41A
0040D8F1 E8 BE3CFFFF call ransomware.wannahusky - copy.4015B4
0040D8F6 8B9E 24020000 mov ebx,dword ptr ds:[esi+224]
0040D8FC 31FF xor edi,edi
0040D8FE 3B7D E4 cmp edi,dword ptr ss:[ebp-1C]
0040D901 7C 0E jl ransomware.wannahusky - copy.40D911
0040D903 899E 24020000 mov dword ptr ds:[esi+224],ebx
0040D909 89EE EA lea esp,dword ptr ss:[ebp-C]

```

Beginning of encrypt function

[ebp-20]: "This is a test file for ex...

EBP=0066F928

.text:0040D8D6 ransomware.wannahusky - copy.exe:\$D8D6 #CD6 <encrypt\_\_>

Address	Hex	ASCII
009DB448	1E 00 00 00 1E 00 00 00	.....This is
009DB458	61 20 74 65 73 74 20 66	a test file for
009DB468	65 78 66 69 66 2E 00 00	exfil.....^A
009DB478	1E 00 00 1E 00 00 00 54	.....This is
009DB488	61 20 74 65 73 74 20 66	a test file for
009DB498	65 78 66 69 66 2E 00 00	exfil.....^A.
009DB4A8	1E 00 00 00 1E 00 00 00	.....
009DB4B8	00 00 00 00 00 00 00 00	.....

0066F8F0 00000020  
0066F8F4 0066FC24  
0066F8F8 00000020  
0066F8FC 0040C4B4 return to ransomware.wannahusky - copy.4015B4  
0066F900 00000000  
0066F904 0000000F  
0066F908 009DB480 "This is a test file for ex..."  
0066F90C 0000001E  
0066F910 009DB448  
0066F914 00000000  
0066F918 0066F928  
0066F920 00000000  
0066F924 009DB478  
0066F928 0066F8B8  
0066F932 00000000  
0066F936 009DB480  
0066F940 0000001F

Analyzing the execution flow, firstly there is a call to encrypt\_\_! which does some shift and xor operation. After further review was found to be generating the xor key for encryption.

```

0040D911 85DB test ebx,ebx
0040D913 75 3B jne ransomware.wannahusky - copy.40D950
0040D915 8D96 04020000 lea edx,dword ptr ds:[esi+204]
0040D91B C74424 08 20000 mov dword ptr ss:[esp+8],20
0040D923 8D86 E4010000 lea eax,dword ptr ds:[esi+1E4]
0040D929 89F1 mov ecx,esi
0040D92B 895424 04 mov dword ptr ss:[esp+4],edx
0040D92F 89C2 mov edx,eax
0040D931 C70424 20000000 mov dword ptr ss:[esp],20
0040D933 E8 0DE3FFFF call <ransomware.wannahusky - copy.encrypt__!1>
0040D93D BA 20000000 mov edx,20
0040D942 8D8E E4010000 lea ecx,dword ptr ds:[esi+1E4]
0040D948 83EC 0C sub esp,C
0040D94B E8 71EBFFFF call ransomware.wannahusky - copy.40C4C1
0040D94B 3B7D 0C cmp edi,dword ptr ss:[ebp+C]
0040D950 72 12 jb ransomware.wannahusky - copy.40D967
0040D953 8B45 0C mov eax,dword ptr ss:[ebp+C]
0040D955 89C2 21 mov dword ptr ss:[esp],edi

```

Likely generates an xor key

EIP

.text:0040D938 ransomware.wannahusky - copy.exe:\$D938 #CD38 <encrypt\_\_!1>

Address	Hex	ASCII
009DB448	1E 00 00 00 1E 00 00 00	.....This is
009DB458	61 20 74 65 73 74 20 66	a test file for
009DB468	65 78 66 69 66 2E 00 00	exfil.....^A.
009DB478	1E 00 00 1E 00 00 00 54	.....This is
009DB488	61 20 74 65 73 74 20 66	a test file for
009DB498	65 78 66 69 66 2E 00 00	exfil.....^A.
009DB4A8	1E 00 00 00 1E 00 00 00	.....
009DB4B8	00 00 00 00 00 00 00 00	.....
009DB4C8	00 00 00 00 00 00 00 00	.....
009DB4D8	1E 00 00 00 1E 00 00 00	.....
009DB4E8	00 00 00 00 00 00 00 00	.....
009DB4F8	00 00 00 00 00 00 00 00	.....
009DB508	00 00 00 00 00 00 00 00	.....
009DB518	00 00 00 00 00 00 00 00	.....

0066F8F0 00000020  
0066F8F4 0066FC24  
0066F8F8 00000020  
0066F8FC 0040C4B4 return to ransomware.wannahusky - copy.4015B4  
0066F900 00000000  
0066F904 0000000F  
0066F908 009DB480 "This is a test file for ex..."  
0066F90C 0000001E  
0066F910 009DB448  
0066F914 00000000  
0066F918 0066F928  
0066F920 00000000  
0066F924 009DB478  
0066F928 0066F8B8  
0066F932 00000000  
0066F936 009DB480  
0066F940 0000001F

We can see the encryption is carried out within a loop, by performing a bit wise xor operation on each byte of data with each byte of key at a time.

The screenshot shows the debugger interface with two main panes: Assembly and Dump.

**Assembly View:**

```

0040D983 C74424 04 1F00 mov dword ptr ss:[esp+4],tf
0040D988 891C24 mov dword ptr ss:[esp],ebx
0040D98E E8 4A86FFFF call ransomware.wannahusky - copy.405FDD
0040D993 8B45 E0 mov eax,dword ptr ss:[ebp-20]
0040D996 8A1438 mov dl,byte ptr ds:[eax+edi]
0040D999 8B45 08 mov eax,dword ptr ss:[ebp+8]
0040D99C 32941E 04020000 xor dl,byte ptr ds:[esi+ebx+204]
0040D9A3 881438 mov byte ptr ds:[eax+edi],dl
0040D9A6 8D47 01 lea eax,dword ptr ds:[edi+1]
0040D9A9 31C7 xor edi,eax
0040D9AB 79 0F jns ransomware.wannahusky - copy.40D9BC
0040D9AD 85C0 test eax,eax
0040D9AF 79 0B jns ransomware.wannahusky - copy.40D9BC
0040D9B1 8945 DC mov dword ptr ss:[ebp-24],eax
0040D9B4 E8 F86CF0FF call ransomware.wannahusky - copy.4046B1
0040D9B5 8945 DC mov dword ptr ss:[ebp-24],eax

```

**Dump View:**

- XOR key used for the first 16 bytes of data:** The memory dump shows the XOR key starting at address 0066FC24, which is used for the first 16 bytes of data.
- First byte of data is loaded into DL register which is the letter "T":** The assembly shows the first byte of data being loaded into the DL register, which is identified as the letter "T".
- and operation of ebx with F signifies that whenever ebx goes over a value of 16 its content are cleared to 0:** The assembly shows the ebx register being compared with F, indicating that when ebx exceeds 16, its content is cleared to 0.
- Then there is a jump to location 40D8FE, which is basically a loop for encryption. This continues until all the data of the cosmo.jpeg is encrypted:** The assembly shows a jump to location 40D8FE, which is a loop for encrypting the data.
- Encrypted data is being stored here. 1st 16 bytes of data:** The memory dump shows the encrypted data being stored in memory, specifically the first 16 bytes of data.

After encrypting the initial 16 bytes of data, there's again a call to keygenerator function. And this new key is used to encrypt the next 16 bytes of data.

The screenshot shows the assembly code and memory dump of the ransomware. The assembly code at address 0040D938 contains a call instruction to the `ransomware.wannahusky - copy.encrypt_!1` function. The memory dump shows the first 16 bytes of encrypted data stored at address 009B8480. A red box highlights the first 16 bytes of encrypted data in the memory dump.

```

0040D92F 89C2 mov edx,eax
0040D931 C70424 20000000 mov dword ptr ss:[esp],20
0040D935 E8 0DE3FFFF call <ransomware.wannahusky - copy.encrypt_!1>
0040D93D BA 20000000 mov edx,20
0040D942 8D8E E4010000 lea ecx,dword ptr ds:[esi+1E4]
0040D948 83EC 0C sub esp,c
0040D94B E8 71BF0FFF call ransomware.wannahusky - copy.40C4C1
0040D950 3B7D 0C cmp edi,dword ptr ss:[ebp+c]
0040D953 72 12 jb ransomware.wannahusky - copy.40D967
0040D955 8845 0C mov eax,dword ptr ss:[ebp+c]
0040D958 893C24 mov dword ptr ss:[esp],edi
0040D95B 8D50 FF lea edx,dword ptr ds:[eax-1]
0040D95E 895424 04 mov dword ptr ss:[esp+4],edx
0040D962 E8 7686FFFF call ransomware.wannahusky - copy.405FDD
0040D967 3B7D E4 cmp edi,dword ptr ss:[ebp-1C]
0040D96A 72 12 jb ransomware.wannahusky - copy.40D97E
0040D96C 8845 E4 mov eax,dword ptr ss:[ebp-1C]
0040D96F 893C24 mov dword ptr ss:[esp],edi
0040D972 8D50 FF lea edx,dword ptr ds:[eax-1]
0040D975 895424 04 mov dword ptr ss:[esp+4],edx
0040D979 E8 5F86FFFF call ransomware.wannahusky - copy.405FDD
0040D97E 83FB 1F cmp ebx,1F
0040D981 76 10 jbe ransomware.wannahusky - copy.40D993
0040D983 C74424 04 1F000 mov dword ptr ss:[esp+4],1F
0040D988 891C24 mov dword ptr ss:[esp],ebx
0040D98E E8 4A86FFFF call ransomware.wannahusky - copy.405FDD
0040D993 8845 E0 mov eax,dword ptr ss:[ebp-20]
0040D996 8A1438 mov dl,byte ptr ds:[eax+edi]
0040D998 8845 08 mov eax,dword ptr ss:[ebp+8]
0040D99C 32941E 04020000 xor dl,byte ptr ds:[esi+ebx+204]
0040D9A3 881438 mov byte ptr ds:[eax+edi],dl
0040D9A6 8047 01 lea eax,dword ptr ds:[edi+1]
0040D9A9 31C7 xor edi,eax
0040D9B0 72 05 jne 0040D931

[ebp-20]: "This is a test file for exfil"

Default (stdcall)
1: [esp] 00000020 00000020
2: [esp+4] 0066FC24 0066FC24
3: [esp+8] 00000020 00000020
4: [esp+C] 0040C4B4 ransom
5: [esp+10] 00000000 00000000

0066FBF0 00000020
0066FBF4 0066FC24
0066FBF8 00000020
0066FBFC 0040C4B4 return to ransomware.wannahusky - copy.0040C4B4
0066FBF00 00000000
0066FB904 0000000F
0066FB908 009B8480 "this is a test file for exfil."
0066FB90C 0000001E
0066FB910 009B8484
0066FB914 00000000
0066FB918 0066F928
0066FB91C 00000000
0066FB920 009B8480
0066FB924 009B8478

Encrypted data is stored in this location.
First 16 bytes of encrypted data.

```

## Key Generation

First xor key generation :

```

EIP: 0040D938
BA 20000000
8D8E E4010000
83EC 0C
3B7D 0C
8845 0C
893C24
8D50 FF
895424 04
E8 7686FFFF
3B7D E4
8845 E4
893C24
8D50 FF
895424 04
E8 5F86FFFF
83FB 1F
76 10
C74424 04 1F00
891C24
E8 4A86FFFF
8B45 E0
8A1438
881438
8D47 01
31C7
79 0F
85C0
79 0B
32941E 04020000
xor dl,byte ptr ds:[esi+ebx+204]
mov byte ptr ds:[eax+edi],dl
lea eax,dword ptr ds:[edi+1]
xor edi,eax
jns ransomware.wannahusky - copy.40D9BC
test eax,eax
jns ransomware.wannahusky - copy.40D9BC

```

Address Hex ASCII  
0066FBF4 00 CF FC 30 FC F3 FC 30 C0 3F 00 FC 0E 00 00 00 .I0u0u0A?.U...  
0066FC04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 ...  
0066FC14 DC 95 C0 78 A2 40 89 89 AD 48 A2 14 92 84 20 87 U.Ax@...Hc...  
0066FC34 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...  
0066FC54 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...  
0066FC64 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...

[ebp-20]:"This is a test file for exfi

Second xor key generation :

```

EIP: 0040D938
BA 20000000
8D8E E4010000
83EC 0C
3B7D 0C
8845 0C
893C24
8D50 FF
895424 04
E8 7686FFFF
3B7D E4
8845 E4
893C24
8D50 FF
895424 04
E8 5F86FFFF
83FB 1F
76 10
C74424 04 1F00
891C24
E8 4A86FFFF
8B45 E0
8A1438
881438
8D47 01
31C7
79 0F
85C0
79 0B
32941E 04020000
xor dl,byte ptr ds:[esi+ebx+204]
mov byte ptr ds:[eax+edi],dl
lea eax,dword ptr ds:[edi+1]
xor edi,eax
jns ransomware.wannahusky - copy.40D9BC
test eax,eax
jns ransomware.wannahusky - copy.40D9BC

```

Address Hex ASCII  
0066FBF4 00 CF FC 30 FC F3 FC 30 C0 3F 00 FC 0E 00 00 00 .I0u0u0A?.U...  
0066FC04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 ...  
0066FC14 53 0F 8A FB C7 45 36 B9 A9 63 B4 F1 C4 CB 73 8B S..0CE6%8C%AES...  
0066FC34 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...  
0066FC54 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...  
0066FC64 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...

[ebp-20]:"This is a test file for exfi

0066FBF4 0040C4B4 return to ransomware.wan

WannaHusky\_RansomWare  
May 2024  
v1.0

## Encryption routine conclusion

So to conclude, the program generates a random 16 byte key each round of encryption where 16 bytes data gets encrypted. Implying every 16 byte of data is encrypted with randomly generated 16 byte xor key.

The screenshot shows a debugger interface with two main panes. The top pane displays assembly code for the encryption routine. The bottom pane shows a memory dump with hex, ASCII, and plain text representations of data. A red box highlights the assembly instruction 'ret' at address 0040D910, with a note 'Returns after completing the encryption routine.' A red box also highlights the plain text data 'plain text data from cosmo.jpeg' in the memory dump. Another red box highlights the encrypted data 'Encrypted data' in the memory dump.

Assembly code (partial):

```
0040D8FE 3B7D E4 cmp edi,dword ptr ss:[ebp-1C]
0040D901 7C 0E j1 ransomware.wannahusky - copy.40D911
0040D903 899E 24020000 mov dword ptr ds:[esi+224],ebx
0040D909 8D65 F4 lea esp,dword ptr ss:[ebp-C]
0040D90C 5B pop ebx
0040D90E 5E pop esi
0040D90F 5F pop edi
0040D910 C3 ret Returns after completing the encryption routine.
0040D911 85DB test ebx,ebx
0040D913 75 3B jne ransomware.wannahusky - copy.40D950
0040D915 8D96 04020000 lea edx,dword ptr ds:[esi+204]
0040D918 C74424 08 20000 mov dword ptr ss:[esp+8],20
0040D923 8D86 E4010000 lea eax,dword ptr ds:[esi+1E4]
0040D929 89F1 mov ecx,esi
0040D92B 89C2 mov dword ptr ss:[esp+4],edx
0040D92F 895424 04 mov edx,eax
0040D931 C70424 20000000 mov dword ptr ss:[esp],20
0040D938 E8 0DE3FFFF call <ransomware.wannahusky - copy.encrypt__!1>
0040D93D BA 20000000 mov edx,20
0040D942 8D8E E4010000 lea ecx,dword ptr ds:[esi+1E4]
0040D948 83EC 0C sub esp,C
0040D950 E8 71EBFFFF call ransomware.wannahusky - copy.40C4C1
0040D953 3B7D 0C cmp edi,dword ptr ss:[ebp+C]
0040D955 72 12 jb ransomware.wannahusky - copy.40D967
0040D958 8B45 0C mov eax,dword ptr ss:[ebp+C]
0040D95B 893C24 mov dword ptr ss:[esp],edi
0040D95E 8D50 FF lea edx,dword ptr ds:[eax-1]
0040D962 895424 04 mov dword ptr ss:[esp+4],edx
0040D967 E8 7686FFFF call ransomware.wannahusky - copy.405FDD
0040D96A 3B7D E4 cmp edi,dword ptr ss:[ebp-1C]
0040D96C 72 12 jb ransomware.wannahusky - copy.40D97E
0040D96F 8B45 E4 mov eax,dword ptr ss:[ebp-1C]
0040D972 893C24 mov dword ptr ss:[esp],edi
```

Memory dump:

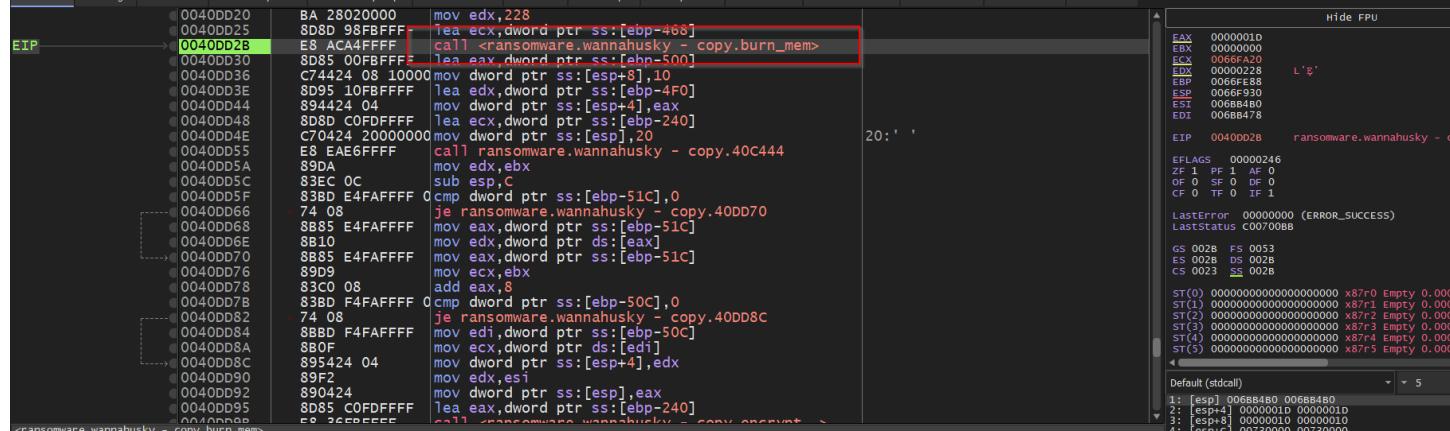
Address	Hex	ASCII
009DB460	69 6C 65 20 66 6F 72 20 65 78 66 69 6C 2E 00 00	file for exfil...
009DB470	04 00 00 00 E0 BB 41 00 15 09 00 15 09 00 00 00 00	This is a test f
009DB480	54 68 69 73 20 69 32 20 61 20 74 65 72 74 20 66	ile for exfil...
009DB490	69 6C 65 20 66 6F 72 20 65 78 66 69 6C 2E 00 00	This is a test f
009DB4A0	04 00 00 00 E0 BB 41 00 1E 00 00 1E 00 00 00 00 00	ile for exfil...
009DB4B0	88 FD A9 08 82 29 FA A9 CC 68 D6 71 E1 F0 00 E1	.ye...)ééhñqäð á...
009DB4C0	3A 63 EF DB A1 2A 44 99 CC 1B D2 98 A8 E5 00 00	:c10*D.I.O..
009DB4D0	04 00 00 00 E0 BB 41 00 1E 00 00 00 1E 00 00 00 00	.....
009DB4E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
009DB500	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
009DB510	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....

One thing about xor operation is that, if you have the xor key and the encrypted data. Performing another round of xor operation on the encrypted data with the same key will give you the original plain text data.

So I saved both the keys from the memory in a file on my desktop so as to decrypt the encrypted data found in the wannahusky file.

After this burnmem function is called which clears the content of key stored in memory.

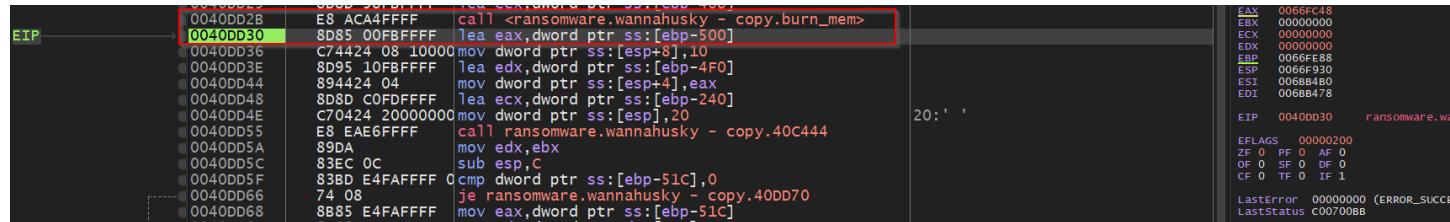
Before Call :



The screenshot shows the assembly code for the ransomware. The EIP register points to the instruction at address 0x040DD2B, which is a call to the `<ransomware.wannahusky - copy.burn_mem>` function. This function is responsible for clearing memory. The assembly code includes various memory operations like mov, lea, and cmp instructions, and function calls to `ransomware.wannahusky - copy.40C444`. The registers and stack status are also visible on the right side of the debugger.

Key in memory - before the call to burn mem.

After call :



The screenshot shows the assembly code after the `burnmem` function has been called. The EIP register now points to the instruction at address 0x040DD30, which is another call to the `<ransomware.wannahusky - copy.burn_mem>` function. The assembly code is identical to the previous state. The registers and stack status are also visible on the right side of the debugger.

Key in memory - after call -- all content is cleared

eax=0066FC48  
dword ptr ss:[ebp-500]=[0066F988]-0

.text:0040D30 ransomware.wannahusky - copy.exe:\$D30 #0130

WannaHusky\_RansomWare  
May 2024  
v1.0

## Encryption validation

The program calls the encryption routine again, likely to validate the key is correct?? Generates the same set of keys and performs an xor operation on the previously encrypted data which results in the original plain text contents.

The screenshot shows two windows from a debugger. The left window displays assembly code for the ransomware's encryption routine. A red box highlights a section of the assembly code where the program performs an XOR operation on previously encrypted data using the same key, resulting in the original plain text. The right window shows a memory dump of the file being encrypted, with a red box highlighting the encrypted data. A callout box states "Same encryption routine, but this time the data passed to eax/edx contains the encrypted text from previous call." Another callout box in the middle right states "Second round of encryption/decryption test". A third callout box at the bottom right states "Encrypted...".

The assembly code in the left window includes instructions like:

```

    mov eax,dword ptr ss:[esp+4]
    call ransomware.wannahusky - copy.405FDD
    xor eax,dword ptr ss:[ebp-20]
    mov dl,byte ptr ds:[eax+edi]
    mov eax,dword ptr ss:[ebp+8]
    xor dl,byte ptr ds:[esi+edi+204]
    lea eax,dword ptr ds:[edi+1]
    xor edi, eax
    jns ransomware.wannahusky - copy.40D9BC
    test eax, eax
    jns ransomware.wannahusky - copy.40D9BC
    xor eax,dword ptr ss:[ebp-24],eax
    call ransomware.wannahusky - copy.4046B1
    mov eax,dword ptr ss:[ebp-24]
    inc ebx
    mov edi, eax
    and ebx, F
    jmp ransomware.wannahusky - copy.40D8FE
    push ebp
    push esp
    push edi
    push esi
    push ebx
    sub esp, 14C

```

The memory dump in the right window shows the file being encrypted, with a red box highlighting the encrypted data. A callout box states "Encrypted...".

The assembly code in the bottom window shows the continuation of the encryption process, with a red box highlighting the XOR operation on previously encrypted data. A callout box states "[ebp+08]:'this is a test file'".

The memory dump in the bottom window shows the plain text message being generated by performing an XOR operation on encrypted data with the same key. A callout box states "We can see plain text msg being generated by performing xor operation on encrypted data with same key".

## Encode function

After this call, the encrypted data is passed to another function which converts it into a base64 encoded string.

The screenshot shows two windows from a debugger, likely Immunity Debugger, illustrating the execution flow between two functions of the ransomware.

**Top Window (Function 1):**

- EIP:** 0040DD88
- Call Site:** Call to `<ransomware.wannahusky - copy.encrypt_>` at address E8 36FBFFFF.
- Registers:**
  - EAX = 0018E4A8 (Addr of encrypted data)
  - ECX = 0018E4B0
  - EDX = 0000001E (Length of data)
  - ESP = 0066F920
  - ESTI = 0018E4B0
  - EDI = 0018E4A8
- Stack:** LastError = 00000000 (ERROR\_SUCCESS), LastStatus = C007008B.

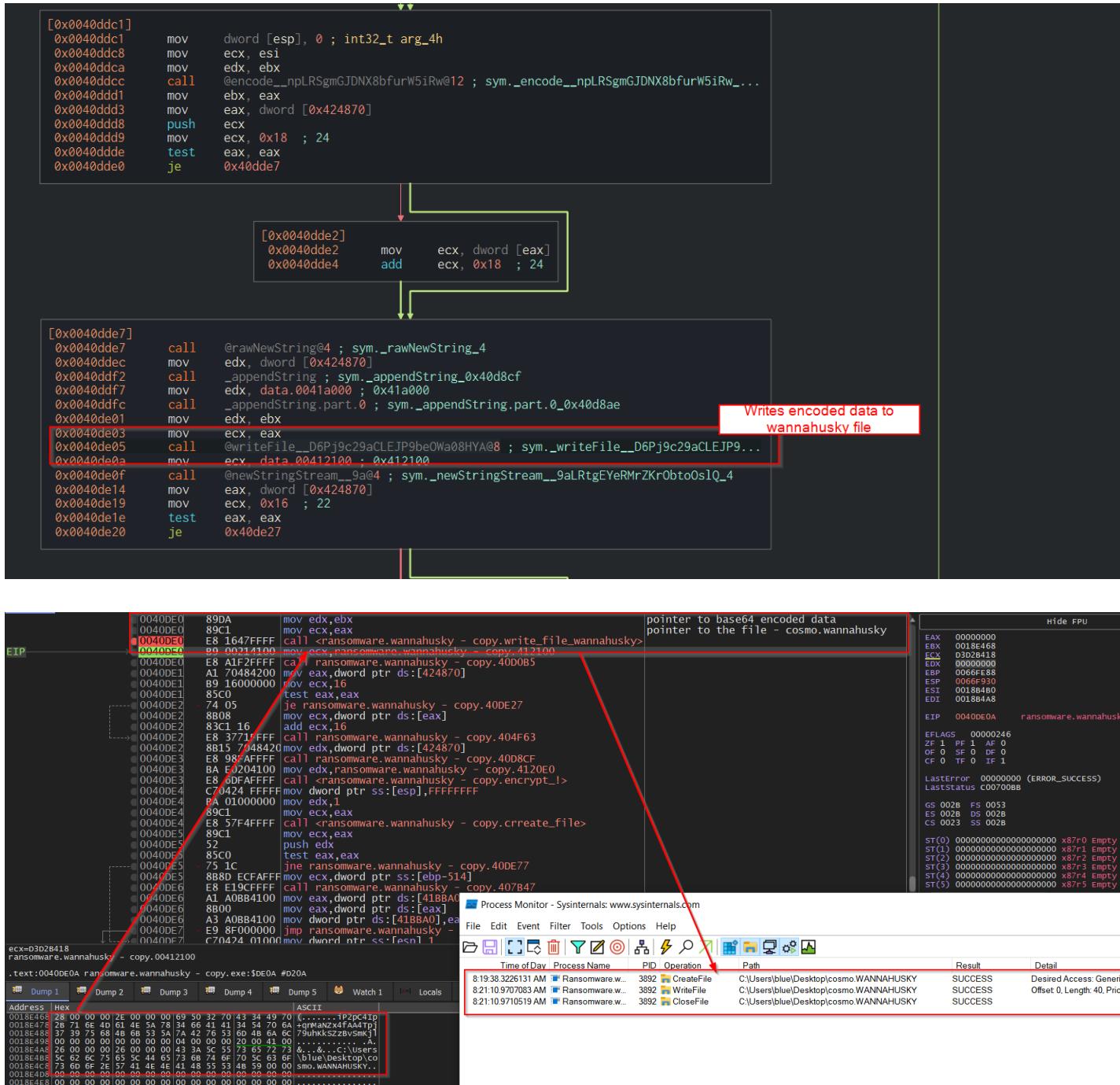
**Bottom Window (Function 2):**

- EIP:** 0040DDC0
- Call Site:** Call to `<ransomware.wannahusky - copy.encode_b64>` at address E8 3B9EFFFF.
- Registers:**
  - EAX = 00000000 (ecx - points to memory containing encrypted data)
  - ECX = 00000000 (edx = length of the data to be encoded)
  - ESP = 0066F920
  - ESTI = 0018E4B0
  - EDI = 0018E4A8
- Stack:** GS 0028 FS 0053, ES 0028 DS 0028, CS 0023 SS 0028.

**Call Flow:**

The call from the first function (E8 36FBFFFF) to the second function (E8 3B9EFFFF) is highlighted with a red box. A red arrow points from the "Encrypted data" label in the first window to the "Encoded data after the call" label in the second window, indicating the data flow between the two stages of processing.

Then writes this data into cosmo.wannahusky file. The write file function takes two arguments, 1 pointer to encoded data and the other file path of cosmo.wannahusky



## Change Background

Then there is a call to changebackground function which writes the wannahusky.png and ps1.ps1 file on disk and then executes the newly written powershell script.

This powershell script tries to change the background/desktop wallpaper to wannahusky.png.

The screenshot shows a debugger interface with assembly code and memory dump sections. Red annotations highlight several key points:

- EIP:** Points to the assembly instruction `call <ransomware.wannahusky - copy.execute_shell_cmd>`. A red box around this instruction is labeled "Executes the powershell script which tries to change the background to wannahusky.png".
- ECX value here points to the powershell command to be executed:** A red box highlights the ECX register value in the assembly code, which is annotated with a red arrow pointing to the "powershell command to be executed" in the EIP annotation.
- Write file function that writes Wannahusky.png and ps1.ps1:** A red box highlights the assembly instruction `call <ransomware.wannahusky - copy.write_file>`, which is annotated with a red arrow pointing to the "Write file function that writes Wannahusky.png and ps1.ps1" in the EIP annotation.
- Process Monitor Log:** A separate window shows a log of file operations. Red boxes highlight several entries:
  - 8:19:30.226131 AM: Ransomware.w... CreateFile C:\Users\blue\Desktop\cosmo.WANNAHUSKY SUCCESS
  - 8:21:10.9707083 AM: Ransomware.w... WriteFile C:\Users\blue\Desktop\cosmo.WANNAHUSKY SUCCESS
  - 8:21:10.9710519 AM: Ransomware.w... CloseFile C:\Users\blue\Desktop\cosmo.WANNAHUSKY SUCCESS
  - 8:25:38.7176614 AM: Ransomware.w... CreateFile C:\Users\blue\Desktop\WANNAHUSKY.png SUCCESS
  - 8:26:07.0731269 AM: Ransomware.w... WriteFile C:\Users\blue\Desktop\WANNAHUSKY.png SUCCESS
  - 8:26:15.812058 AM: Ransomware.w... CloseFile C:\Users\blue\Desktop\WANNAHUSKY.png SUCCESS
  - 8:26:15.8121941 AM: Ransomware.w... CreateFile C:\Users\blue\Desktop\cosmo.jpeg SUCCESS
  - 8:26:33.6837381 AM: Ransomware.w... WriteFile C:\Users\blue\Desktop\cosmo.jpeg SUCCESS
  - 8:26:33.6838227 AM: Ransomware.w... SetDisposition... C:\Users\blue\Desktop\cosmo.jpeg SUCCESS
  - 8:26:33.6838592 AM: Ransomware.w... CloseFile C:\Users\blue\Desktop\cosmo.jpeg SUCCESS
  - 8:26:33.683962 AM: Ransomware.w... CreateFile C:\Users\blue\Desktop\ps1.ps1 SUCCESS
  - 8:27:28.0984758 AM: Ransomware.w... WriteFile C:\Users\blue\Desktop\ps1.ps1 SUCCESS
  - 8:27:58.0100141 AM: Ransomware.w... CloseFile C:\Users\blue\Desktop\ps1.ps1 SUCCESS
  - 8:27:58.0114391 AM: Ransomware.w... CreateFile C:\Users\blue\Desktop\ps1.ps1 SUCCESS
- Assembly Dump:** The bottom section shows assembly dumps for two different memory locations. Red boxes highlight specific assembly instructions and memory dump sections, with arrows pointing to the corresponding entries in the Process Monitor log.

## Dropped files

Powershell script : Ps1.ps1

```
1 $code = @'
2     using System.Runtime.InteropServices;
3
4     namespace Win32{
5
6         public class Wallpaper{
7
8             [DllImport("user32.dll", CharSet=CharSet.Auto)]
9             static extern int SystemParametersInfo (int uAction , int uParam , string lpvParam , int fuWinIni) ;
10
11            public static void SetWallpaper(string thePath){
12                | SystemParametersInfo(20,0,thePath,3);
13            }
14        }
15    }
16    '@
17 add-type $code
18
19 $currDir = Get-Location
20 $wallpaper = ".\WANNAHUSKY.PNG"
21 $fullpath = Join-Path -path $currDir -ChildPath $wallpaper
22
23 [Win32.Wallpaper]::SetWallpaper($fullpath)
24
```

WannaHusky.PNG :



that picture of cosmo on your  
desktop is now encrypted!

to save him, you must send 100 Huskycoin to <https://huskyhacks.dev>

hurry! you have 24 hours before we delete cosmo

WannaHusky\_RansomWare  
May 2024  
v1.0



## Decryption Routine

I saved the keys used for encryption on my desktop.

Writing a simple python script to decrypt the above contents using the saved keys :

Running the script, we can get the encrypted data back :

The image shows a terminal window titled 'Cmder' with a light blue background. It displays the command line and the output of a Python script. The command line shows the user navigating to their desktop and running the script. The output shows the decrypted data, which is a test file for exfil.

```
import base64

def xor_decrypt(data, key):
    decrypted = ''
    for i in range(len(data)):
        decrypted += chr(data[i] ^ key[i % len(key)])
    return decrypted

def main():
    # Base64 encoded xor encrypted data
    encoded_data = b'iP2pC4Ip+qnMaNZx4fAA4Tpj79uhKksZzBvSmKj1'

    # XOR key
    with open("key1.bin", "rb") as keyfile:
        xor_key_1 = keyfile.read()

    with open("key2.bin", "rb") as keyfile2:
        xor_key_2 = keyfile2.read()

    # Decode the Base64 encoded data
    decoded_data = base64.b64decode(encoded_data)

    # Decrypt the decoded data using XOR
    decrypted_data = xor_decrypt(decoded_data[:16], xor_key_1)
    decrypted_data += xor_decrypt(decoded_data[16:], xor_key_2)

    print("Decrypted Data:", decrypted_data.encode())

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

```
C:\Users\blue
λ cd Desktop\
C:\Users\blue\Desktop
λ python decrypt.py
Decrypted Data: b'This is a test file for exfil.'
C:\Users\blue\Desktop
λ |
```

# Indicators of Compromise

The full list of IOCs can be found in the Appendices.

## Network Indicators

{Description of network indicators}

N/A - no network indicators found for this sample.

## Host-based Indicators

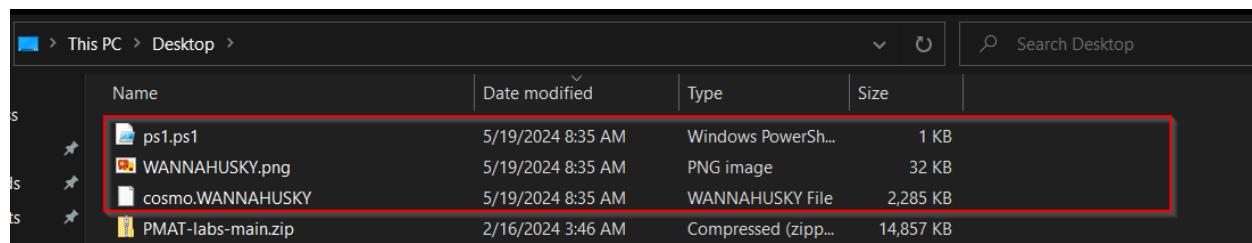
{Description of host-based indicators}

Powershell script - ps1.ps1 ===> changes the background/desktop wallpaper to wannahusky.png

WannaHusky.png ===> image file containing the ransom note

Cosmo.wannahusky ===> encrypted cosmo.jpeg file

Execution of cmd /tree C:/ command ===> likely to hide the powershell window



Name	Date modified	Type	Size
ps1.ps1	5/19/2024 8:35 AM	Windows PowerSh...	1 KB
WANNAHUSKY.png	5/19/2024 8:35 AM	PNG image	32 KB
cosmo.WANNAHUSKY	5/19/2024 8:35 AM	WANNAHUSKY File	2,285 KB
PMAT-labs-main.zip	2/16/2024 3:46 AM	Compressed (zipp...	14,857 KB

```
x86_netfx4-system_enterpriseservices_t1b_b03f5f7f11d50a3a_4_0_15805_0_none_6b8061d4f0a842e1
x86_netfx4-system_ni_b03f5f7f11d50a3a_4_0_15805_0_none_3230b5d6b21f5a4
x86_netfx4-system_t1b_b03f5f7f11d50a3a_4_0_15805_0_none_568dd2c96cc4ee5
x86_netfx4-system_web_t1b_b03f5f7f11d50a3a_4_0_15805_0_none_d768e52477f719da
x86_netfx4-uninstall_windataview_t1b_b03f5f7f11d50a3a_4_0_15805_0_none_e555f21bb95819
x86_netfx4-ucrbase_t1b_b03f5f7f11d50a3a_4_0_15805_0_none_4664a855d446c9
x86_netfx4-ucrbase_clr_dll_b03f5f7f11d50a3a_4_0_15805_0_none_355abc99c3e69081
x86_netfx4-uninstallsqlpersiststate_t1b_b03f5f7f11d50a3a_4_0_15805_0_none_794b02dbe92715b
x86_netfx4-uninstallsqlstatetemplate_sgl_b03f5f7f11d50a3a_4_0_15805_0_none_231dfc33015c6db
x86_netfx4-uninstallsqlservice_t1b_b03f5f7f11d50a3a_4_0_15805_0_none_e77f71a6dad084b1
x86_netfx4-ubc7ui.dll_b03f5f7f11d50a3a_4_0_15805_0_none_d08ea519_a0b41
x86_netfx4-ubc7ui.dll_b03f5f7f11d50a3a_4_0_15805_0_none_c05ce4e7243b7fee
x86_netfx4-vbe_exe_b03f5f7f11d50a3a_4_0_15805_0_none_d9b06e519e58ab6f
x86_netfx4-webengine4.dll_b03f5f7f11d50a3a_4_0_15805_0_non_2efdf3d759de32d3
x86_netfx4-webengine4.dll_b03f5f7f11d50a3a_4_0_15805_0_none_21a607a7dfb96fe0
x86_netfx4-webengine4.dll_b03f5f7f11d50a3a_4_0_15805_0_none_e7000000000000000000000000000000
x86_netfx4-webengine4.dll_b03f5f7f11d50a3a_4_0_15805_0_none_1439be0d98050ba0
x86_netfx4-web_config_b03f5f7f11d50a3a_4_0_15805_0_non_16ee165232eca15
x86_netfx4-web_hightrust_config_b03f5f7f11d50a3a_4_0_15805_0_none_e7000000000000000000000000000000
x86_netfx4-web_lowtrust_config_b03f5f7f11d50a3a_4_0_15805_0_none_40404040404040404040404040404040
x86_netfx4-web_lowtrust_config_default_b03f5f7f11d50a3a_4_0_15805_0_none_64c27b3126a2f1b3
x86_netfx4-web_mediumtrust_config_b03f5f7f11d50a3a_4_0_15805_0_none_1033hd44dfdf917d6
x86_netfx4-web_minimaltrust_config_b03f5f7f11d50a3a_4_0_15805_0_none_90e8556ad9960374
x86_netfx4-workflow_minimaltrust_config_b03f5f7f11d50a3a_4_0_15805_0_none_2a0b6ab1d229bd70
x86_netfx4-mininet_util.dll_b03f5f7f11d50a3a_4_0_15805_0_none_13ac38a1checch00
x86_netfx4-workflowlowerv..ormancecounters.dll_b03f5f7f11d50a3a_4_0_15805_0_none_ff9984c8589b22fd
x86_netfx4-workflowlowerv..ormancecounters.dll_b03f5f7f11d50a3a_4_0_15805_101_none_f765d0a3a0bd0
x86_netfx4-workflowlowerv..ormancecounters_man_b03f5f7f11d50a3a_4_0_15805_0_none_ff82830658ac60db
x86_netfx4-workflow_low_targets_files_b03f5f7f11d50a3a_4_0_15805_0_none_448273f6a759d065
x86_netfx4-upfetman_b03f5f7f11d50a3a_4_0_15805_0_none_clef3be816e5688c
```

WannaHusky\_RansomWare

May 2024

v1.0

# Rules & Signatures

## Yara Rules

```
rule RANSOMWARE_WANNAHUSKY {
    meta:

        author = "Aniksha Shetty"
        description = "Detect malicious ransomware sample, from wannahusky family"
        created = "05-19-2024"
        strings:
            $target_file = "@Desktop\\cosmo.jpeg" ascii
            $background_file = "\\WANNAHUSKY.PNG" ascii
            $powershell = "@Desktop\\ps1.ps1" ascii
            $encrypted_file = "@Desktop\\cosmo.WANNAHUSKY"
            $base_64 =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789-
_ABCDEFHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/" ascii
            $compile_lang = "@nim" ascii
            $crypt_api = "bCryptGenRandom" ascii
            $pe_magic_byte = "MZ"

        condition:
            $pe_magic_byte at 0 and $compile_lang and $crypt_api and
            ($background_file or $powershell) or
            ($target_file and $base_64 and $encrypted_file)

}
```