



Practical Malware Analysis & Triage

Malware Analysis Report

WannaHusky_RansomWare

May 2024 | AnikshaShetty | v1.0

Table of Contents

Table of Contents	2
Executive Summary	3
High-Level Technical Summary	4
Malware Composition	5
WannaHusky.png	5
ps1.ps1	5
Basic Static Analysis	6
Floss - String Analysis	6
PEStudio	
This is a 32 bit binary, compiled on oct 12 2021	9
Basic Dynamic Analysis	10
Initial Detonation	10
ProcMon Analysis	11
Advanced Dynamic Analysis	17
Indicators of Compromise	31
Network Indicators	32
Host-based Indicators	32
Rules & Signatures	33
Appendices	33
A. Yara Rules	33
B. Callback URLs	33
C. Decompiled Code Snippets	33

Executive Summary

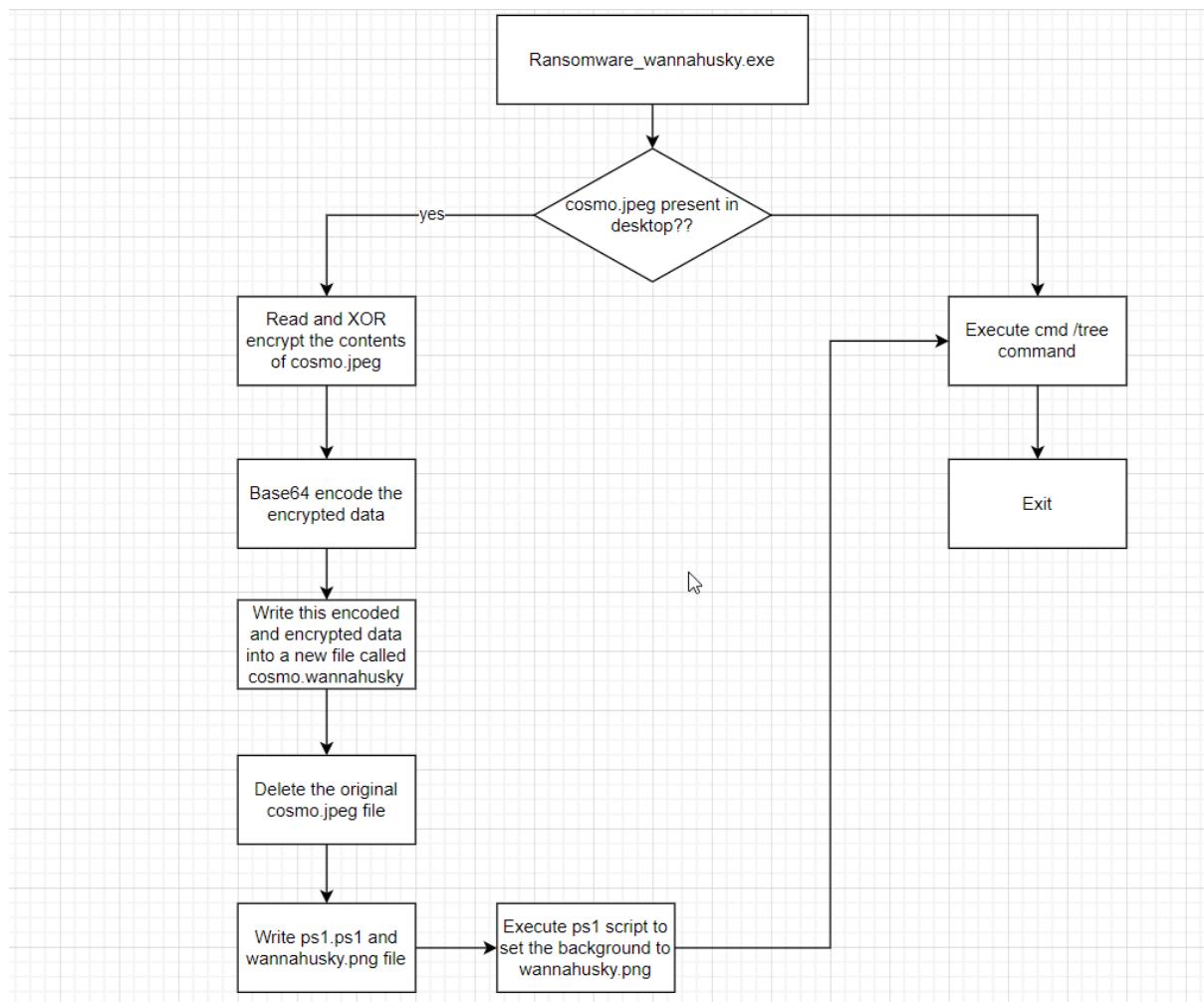
sha256sum	3D35CEBCF40705C23124FDC4656A7F400A316B8E96F1F9E0C187 E82A9D17DCA3
-----------	--

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
cosmo.wannahusky	
ps1.ps1	
wannahusky.png	

WannaHusky.png

Background image that contains the ransom note.

ps1.ps1

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

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

```
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
\tpHVs
. . . . .
```



```

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:
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789-_ABC
DEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
@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

<table border="1"> <thead> <tr> <th>property</th><th>value</th></tr> </thead> <tbody> <tr> <td><code>footprint > sha256</code></td><td>3D35CEBCF40705C23124FDC4656A7F400A316B8E96F1F9E0C187E82A9D17DCA3</td></tr> <tr> <td><code>first-bytes-hex</code></td><td>4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 40 00 00 00 00 00 00 00</td></tr> <tr> <td><code>first-bytes-text</code></td><td>MZ .. @ ..</td></tr> <tr> <td><code>file > size</code></td><td>412905 bytes</td></tr> <tr> <td><code>entropy</code></td><td>6.455</td></tr> <tr> <td><code>signature</code></td><td>n/a</td></tr> <tr> <td><code>tooling</code></td><td>wait...</td></tr> <tr> <td><code>file-type</code></td><td>executable</td></tr> <tr> <td><code>cpu</code></td><td>32-bit</td></tr> <tr> <td><code>subsystem</code></td><td>console</td></tr> <tr> <td><code>file-version</code></td><td>n/a</td></tr> <tr> <td><code>description</code></td><td>n/a</td></tr> <tr> <td><code>stamps</code></td><td></td></tr> <tr> <td><code>compiler-stamp</code></td><td>Tue Oct 12 20:08:25 2021 UTC</td></tr> <tr> <td><code>debug-stamp</code></td><td>n/a</td></tr> <tr> <td><code>resource-stamp</code></td><td>n/a</td></tr> <tr> <td><code>import-stamp</code></td><td>n/a</td></tr> <tr> <td><code>export-stamp</code></td><td>n/a</td></tr> <tr> <td><code>names</code></td><td></td></tr> <tr> <td><code>file</code></td><td>c:\users\blue\Desktop\ransomware.wannahusky.exe</td></tr> <tr> <td><code>debug</code></td><td>n/a</td></tr> <tr> <td><code>export</code></td><td>n/a</td></tr> <tr> <td><code>version</code></td><td>n/a</td></tr> <tr> <td><code>manifest</code></td><td>n/a</td></tr> <tr> <td><code>.NET > module</code></td><td>n/a</td></tr> <tr> <td><code>certificate > program-name</code></td><td>n/a</td></tr> </tbody> </table>	property	value	<code>footprint > sha256</code>	3D35CEBCF40705C23124FDC4656A7F400A316B8E96F1F9E0C187E82A9D17DCA3	<code>first-bytes-hex</code>	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 40 00 00 00 00 00 00 00	<code>first-bytes-text</code>	MZ .. @ ..	<code>file > size</code>	412905 bytes	<code>entropy</code>	6.455	<code>signature</code>	n/a	<code>tooling</code>	wait...	<code>file-type</code>	executable	<code>cpu</code>	32-bit	<code>subsystem</code>	console	<code>file-version</code>	n/a	<code>description</code>	n/a	<code>stamps</code>		<code>compiler-stamp</code>	Tue Oct 12 20:08:25 2021 UTC	<code>debug-stamp</code>	n/a	<code>resource-stamp</code>	n/a	<code>import-stamp</code>	n/a	<code>export-stamp</code>	n/a	<code>names</code>		<code>file</code>	c:\users\blue\Desktop\ransomware.wannahusky.exe	<code>debug</code>	n/a	<code>export</code>	n/a	<code>version</code>	n/a	<code>manifest</code>	n/a	<code>.NET > module</code>	n/a	<code>certificate > program-name</code>	n/a	
property	value																																																						
<code>footprint > sha256</code>	3D35CEBCF40705C23124FDC4656A7F400A316B8E96F1F9E0C187E82A9D17DCA3																																																						
<code>first-bytes-hex</code>	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 40 00 00 00 00 00 00 00																																																						
<code>first-bytes-text</code>	MZ .. @ ..																																																						
<code>file > size</code>	412905 bytes																																																						
<code>entropy</code>	6.455																																																						
<code>signature</code>	n/a																																																						
<code>tooling</code>	wait...																																																						
<code>file-type</code>	executable																																																						
<code>cpu</code>	32-bit																																																						
<code>subsystem</code>	console																																																						
<code>file-version</code>	n/a																																																						
<code>description</code>	n/a																																																						
<code>stamps</code>																																																							
<code>compiler-stamp</code>	Tue Oct 12 20:08:25 2021 UTC																																																						
<code>debug-stamp</code>	n/a																																																						
<code>resource-stamp</code>	n/a																																																						
<code>import-stamp</code>	n/a																																																						
<code>export-stamp</code>	n/a																																																						
<code>names</code>																																																							
<code>file</code>	c:\users\blue\Desktop\ransomware.wannahusky.exe																																																						
<code>debug</code>	n/a																																																						
<code>export</code>	n/a																																																						
<code>version</code>	n/a																																																						
<code>manifest</code>	n/a																																																						
<code>.NET > module</code>	n/a																																																						
<code>certificate > program-name</code>	n/a																																																						

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

property	value	value	value	value	value	value	value
section	section[0]	section[1]	section[2]	section[3]	section[4]	section[5]	
<code>name</code>	.text	.data	.rdata	.bss	.idata	.CRT	
<code>footprint > sha256</code>	6299B97342DD96C74485E5C...	076B2D201A61086430CF80C...	923AB031DB80B7FD6EC092...	n/a	28BAE041F27B62B7321813...	FE2509EA26A02F275E6...	
<code>entropy</code>	6.403	1.477	7.810	n/a	4.936	0.279	
<code>file-ratio (84.82%)</code>	14.26 %	0.12 %	9.42 %	n/a	0.50 %	0.12 %	
<code>raw-address (begin)</code>	0x00000400	0x0000EA00	0x0000EC00	0x00000000	0x00018400	0x00018C00	
<code>raw-address (end)</code>	0x0000EA00	0x0000EC00	0x00018400	0x00000000	0x00018C00	0x00018E00	
<code>raw-size (350208 bytes)</code>	0x0000E600 (58880 bytes)	0x00000200 (512 bytes)	0x00009800 (38912 bytes)	0x00000000 (0 bytes)	0x00000800 (2048 bytes)	0x00000200 (512 bytes)	
<code>virtual-address</code>	0x00001000	0x00010000	0x00011000	0x0001B000	0x00025000	0x00026000	
<code>virtual-size</code>	0x0000E404 (58372 bytes)	0x0000009C (156 bytes)	0x000096D0 (38608 bytes)	0x0000987C (39036 bytes)	0x000007C4 (1988 bytes)	0x00000034 (52 bytes)	
<code>characteristics</code>	0x05000060	0xC0600040	0x40600040	0xC0600080	0xC0300040	0xC0300040	
<code>write</code>	-	x	-	x	x	x	
<code>execute</code>	x	-	-	-	-	-	
<code>share</code>	-	-	-	-	-	-	
<code>self-modifying</code>	-	-	-	-	-	-	
<code>virtual</code>	-	-	-	x	-	-	
<code>items</code>							
<code>directory > import</code>	-	-	-	-	0x00025000	-	
<code>directory > thread-local-storage</code>	-	-	0x0001A10C	-	-	-	
<code>directory > import-address</code>	-	-	-	-	0x00025168	-	
<code>base-of-code</code>	0x00001000	-	-	-	-	-	
<code>base-of-data</code>	-	0x00010000	-	-	-	-	
<code>entry-point</code>	0x000014C0	-	-	-	-	-	
<code>thread-local-storage</code>	0x0000E330	-	-	-	-	-	
<code>thread-local-storage</code>	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_c315hd50bc4b72da
    └── f
        └── r
            └── and64_microsoft-windows-media-streaming-ps_31bf3856ad364e35_10.0.19041.1_none_fa562f83ace75b88
                └── f
                    └── r
                        └── and64_microsoft-windows-mediafoundation-msfsrv_31bf3856ad364e35_10.0.19041.1865_none_33422b307065eeba
                            └── f
                                └── r
                                    └── and64_microsoft-windows-medialayer-autoplay_31bf3856ad364e35_10.0.19041.1865_none_10e43a01c2238eb1
                                        └── f
                                            └── r
                                                └── and64_microsoft-windows-medialayer-core_31bf3856ad364e35_10.0.19041.1566_none_800f26e104636391
                                                    └── f
                                                        └── r
                                                            └── and64_microsoft-windows-medialayer-logagent_31bf3856ad364e35_10.0.19041.746_none_c93d70420d81ce4
                                                                └── f
                                                                    └── r
                                                                        └── and64_microsoft-windows-medialayer-mls_31bf3856ad364e35_10.0.19041.746_none_4eda1c6d21dd9881
                                                                            └── f
                                                                                └── r
                                                                                    └── and64_microsoft-windows-medialayer-setup_31bf3856ad364e35_10.0.19041.1266_none_22b99d078bbc3016
                                                                                        └── f
                                                                                            └── r
                                                                                                └── and64_microsoft-windows-medialayer-shortcut_31bf3856ad364e35_10.0.19041.1_none_64c27fc7ed12a491
                                                                                             and64_microsoft-windows-medialayer-skins_31bf3856ad364e35_10.0.19041.2006_none_1e3695h18e994dh8
                                                                                                 └── f
                                                                                                     └── r
                                         and64_microsoft-windows-medialayer-vis_31bf3856ad364e35_10.0.19041.1266_none_e5afec1878374111
                                             └── f
                                                 └── r
                                                     └── and64_microsoft-windows-medialayer-wmasf_31bf3856ad364e35_10.0.19041.1_none_5da6fe23dfc593c7
                                                         and64_microsoft-windows-medialayer-wmerror_31bf3856ad364e35_10.0.19041.1_none_ef4600f715653f49d
                                                         and64_microsoft-windows-medialayer-wmnetngr_31bf3856ad364e35_10.0.19041.746_none_253f40e31d7124d4
                                                             └── f
                                                               └── r
                                                               and64_microsoft-windows-medialayer-wmpdgm_31bf3856ad364e35_10.0.19041.1266_none_18fdf6dfdcfd40
                                                               └── f
                                                               └── r
                                                               and64_microsoft-windows-medialayer-wmpeffects_31bf3856ad364e35_10.0.19041.1266_none_6e13bacd44a30951
                                                               └── f
                                                               └── r
                                                               and64_microsoft-windows-medialayer-wmpps_31bf3856ad364e35_10.0.19041.1_none_647h5d5hd15acd49
                                                               and64_microsoft-windows-medialayer-wmpshell_31bf3856ad364e35_10.0.19041.1266_none_38802b3df8a8026dd
                                                               └── f
                                                               └── r
                                                               and64_microsoft-windows-medialayer-wmvcore_31bf3856ad364e35_10.0.19041.1806_none_7bb1760f8b3ee704
                                                               └── f
                                                               └── r
                                                               and64_microsoft-windows-medialayer-wmwdk_31bf3856ad364e35_10.0.19041.1_none_5c1c54e5c5fe9fc0
                                                               and64_microsoft-windows-nfaudioadm_31bf3856ad364e35_10.0.19041.1_none_fff14e03001d1e1ea
                                                               and64_microsoft-windows-nfvc_31bf3856ad364e35_10.0.19041.1_none_8e7de309963398b48
                                                               and64_microsoft-windows-nfaacenc_31bf3856ad364e35_10.0.19041.1_none_289h214h917?ff?
                                                               and64_microsoft-windows-nfasfsrcsnk_31bf3856ad364e35_10.0.19041.1865_none_3502fc1af1f86813
                                                               └── f
                                                               └── r
                                                               and64_microsoft-windows-nfaudionv_31bf3856ad364e35_10.0.19041.746_none_e77e4c60d41c1b03
                                                               └── f
                                                               └── r
                                                               and64_microsoft-windows-nfc42x.resources_31bf3856ad364e35_10.0.19041.1_en-us_7892c6682e6258c8
                                                               and64_microsoft-windows-nfc42x_31bf3856ad364e35_10.0.19041.1456_none_d10e1541b45e0391
                                                               └── f
                                                               └── r
                                                               and64_microsoft-windows-nfcore.resources_31bf3856ad364e35_10.0.19041.1_en-us_57921162b99398b6
                                                               and64_microsoft-windows-nfcore_31bf3856ad364e35_10.0.19041.2086_none_65ae19bd0c581hd6
                                                               └── f
                                                               └── r
                                                               and64_microsoft-windows-nfds_31bf3856ad364e35_10.0.19041.1645_none_1a270410d2d089fe
                                                               └── f
                                                               └── r
                                                               and64_microsoft-windows-nfddcer_31bf3856ad364e35_10.0.19041.1_none_bp380hf72297h48
                                                               and64_microsoft-windows-nfsql_31bf3856ad364e35_10.0.19041.1_none_BF52e28281e1449f
                                                               and64_microsoft-windows-nf1263enc_31bf3856ad364e35_10.0.19041.1_none_4052cf3d3a53273
                                                               and64_microsoft-windows-nf1264enc_31bf3856ad364e35_10.0.19041.1866_none_fee09de9e9af2b5e
                                                               └── f
                                                               └── r
                                                               and64_microsoft-windows-nfmjpegdec_31bf3856ad364e35_10.0.19041.1348_none_8e16cd37092ach5a
                                                               └── f
                                                               └── r
                                                               and64_microsoft-windows-nfmkvsrcsnk_31bf3856ad364e35_10.0.19041.1566_none_jb95e3ad4277b3e4d
```

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 :

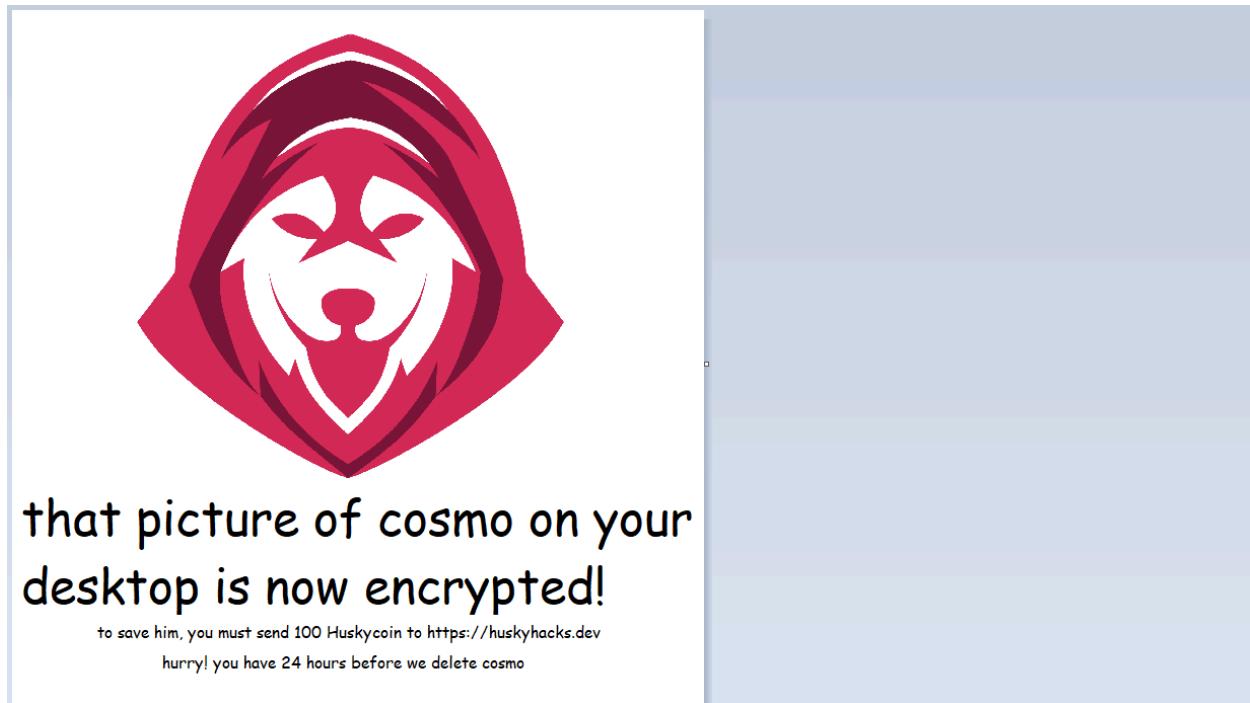
Time	Process	File Operation	File Path	Status
6:40:28...	Ransomware.w...	5412 WriteFile	C:\Users\blue\Desktop\cosmo.WANNAHUSKY	SUCCESS
6:40:28...	Ransomware.w...	5412 WriteFile	C:\Users\blue\Desktop\cosmo.WANNAHUSKY	SUCCESS
6:40:28...	Ransomware.w...	5412 CloseFile	C:\Users\blue\Desktop\cosmo.WANNAHUSKY	SUCCESS
6:40:28...	Ransomware.w...	5412 ReadFile	C:\Users\blue\Desktop\Ransomware.wannahusky.exe	SUCCESS
6:40:28...	Ransomware.w...	5412 CreateFile	C:\Users\blue\Desktop\WANNAHUSKY.png	SUCCESS
6:40:28...	Ransomware.w...	5412 WriteFile	C:\Users\blue\Desktop\WANNAHUSKY.png	SUCCESS
6:40:28...	Ransomware.w...	5412 WriteFile	C:\Users\blue\Desktop\WANNAHUSKY.png	SUCCESS
6:40:28...	Ransomware.w...	5412 CloseFile	C:\Users\blue\Desktop\WANNAHUSKY.png	SUCCESS
6:40:28...	Ransomware.w...	5412 CreateFile	C:\Users\blue\Desktop\cosmo.jpeg	SUCCESS
6:40:28...	Ransomware.w...	5412 QueryAttributeT...	C:\Users\blue\Desktop\cosmo.jpeg	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\ps1.ps1	SUCCESS
6:40:28...	Ransomware.w...	5412 WriteFile	C:\Users\blue\Desktop\ps1.ps1	SUCCESS
6:40:28...	Ransomware.w...	5412 CloseFile	C:\Users\blue\Desktop\ps1.ps1	SUCCESS
6:40:28...	Ransomware.w...	5412 ReadFile	C:\Windows\SysWOW64\msvcr.dll	SUCCESS
6:40:28...	Ransomware.w...	5412 CreateFile	C:\Windows\SysWOW64\cmd.exe	SUCCESS
6:40:28...	Ransomware.w...	5412 QueryBasicInfor...	C:\Windows\SysWOW64\cmd.exe	SUCCESS
6:40:28...	Ransomware.w...	5412 CloseFile	C:\Windows\SysWOW64\cmd.exe	SUCCESS

Process logs showing powershell and cmd execution :

WannaHusky_RansomWare
May 2024
v1.0

Time o...	Process Name	PID	Operation	Path	Result	Detail
6:40:28...	Ransomware.w...	5412	Process Start		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\System32\Conhost.exe	SUCCESS	PID: 5132, Command line: '\?\C:\Windows\system32\conhost.exe 0xffffffff-ForceVI
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,

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

Cosmo.wannahusky ---> contains base64 encoded encrypted data.

In 1, Col 1

100% Windows

Decoding b64 gives encrypted data :

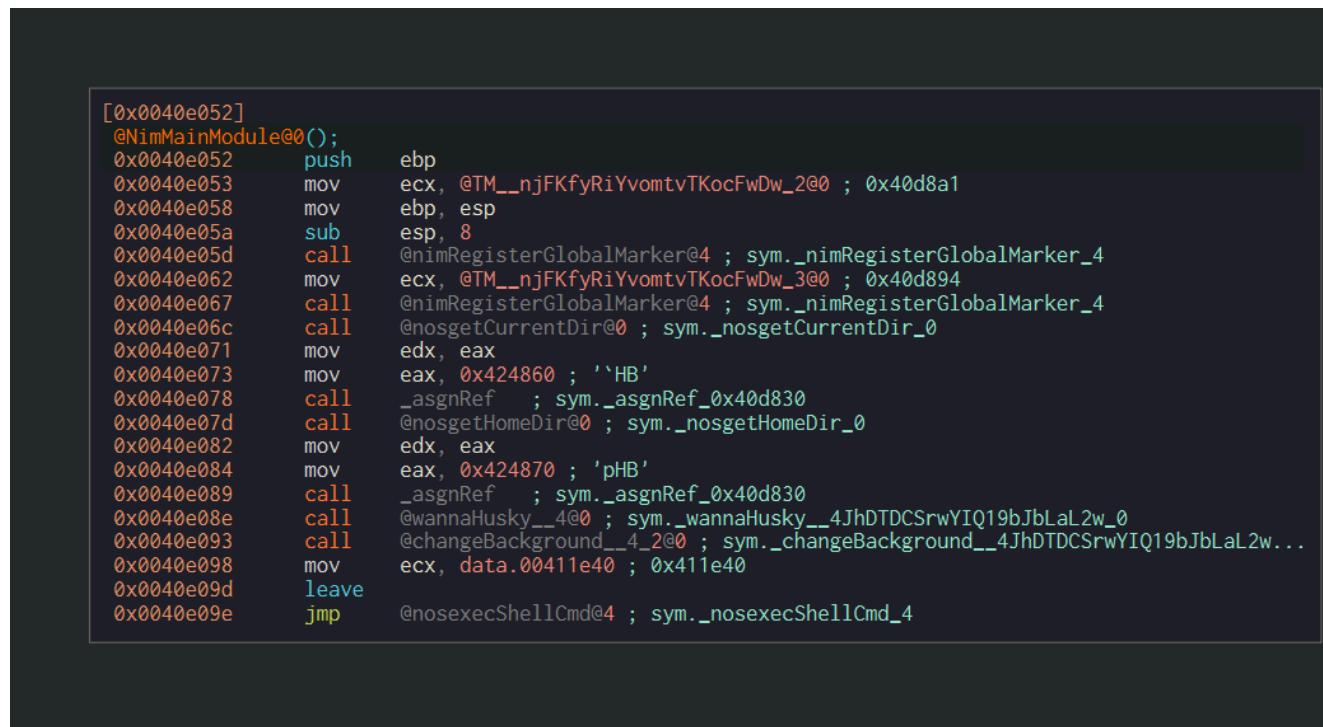
Advanced Static Analysis

WannaHusky_RansomWare
May 2024
v1.0

{Screenshots and description about findings during advanced static analysis}

Loading the binary in cutter :

Main program of the binary starts here :



The screenshot shows the assembly code for the main program entry point. The assembly code is as follows:

```
[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_4jhDTDCSrwYIq19bJbLaL2w_0)

void sym._wannaHusky_4jhDTDCSrwYIq19bJbLaL2w_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
0x0040da5c    mov     edx, dword [0x424870]
0x0040da60    mov     esi, eax
0x0040da66    call   _appendString ; sym._appendString_0x40d8cf
0x0040da6b    mov     edx, 0x041a0a0 ; 0x41a0a0
0x0040da70    call   _appendString.part.0 ; sym._appendString.part.0_0x40d8ae
0x0040da75    mov     ecx, dword [var_51ch]
0x0040da7b    call   @readFile_4@4 ; sym._readFile_4PGnM9bWmsH0Nu7dnr3XzgA_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     eax, edx
0x0040da97    xor     eax, 0x8a ; 138
0x0040da99    mov     edi, edx
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
0x0040daccc  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_4jhDTDCSrwYIq19bJbLaL2w_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_46ch]
0x0040dd18    mov    dword [esp], esi ; int32_t arg_8h
0x0040dd1b    call   _encrypt_dcoBdmUaaCC9cnR23eFxSLAbmode ; sym._encrypt_dcoBdmUaaCC9cnR23...
0x0040dd20    mov    edx, 0x20 ; 32
0x0040dd25    lea    eax, [var_46ch]
0x0040dd2b    call   @_burnMem_4F7Hz34TgxTmMy6XY9c0Sg_8
0x0040dd30    lea    eax, [var_504h]
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_W9cjVocn1tjhW7p7xohIj6A@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_py6wg79abw8iTzUm11Z7JOA@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 the assembly view of a debugger. The assembly code is as follows:

```
EIP: 0040E000 55 push ebp
      0040E001 B9 A1D84000 mov ecx, ransomware.wannahusky - copy.40D8A1
      0040E002 89E5 mov ebp, esp
      0040E003 83EC 08 sub esp, 8
      0040E004 E8 177EFFFF call <ransomware.wannahusky - copy.reg_global_marker>
      0040E005 B9 94D84000 mov ecx, ransomware.wannahusky - copy.40D894
      0040E006 E8 0D7EFFFF call <ransomware.wannahusky - copy.reg_global_marker>
      0040E007 E8 2497FFFF call <ransomware.wannahusky - copy.get_current_dir>
      0040E008 89C2 mov edx, eax
      0040E009 B8 60484200 mov eax, ransomware.wannahusky - copy.424860
      0040E00A E8 B3F7FFFF call <ransomware.wannahusky - copy.unknown>
      0040E00B E8 199AFFFF call <ransomware.wannahusky - copy.get_home_dir>
      0040E00C 89C2 mov edx, eax
      0040E00D B8 70484200 mov eax, ransomware.wannahusky - copy.424870
      0040E00E E8 A2F7FFFF call <ransomware.wannahusky - copy.unknown>
      0040E00F E8 34F9FFFF call <ransomware.wannahusky - copy.read_write??>
      0040E010 E8 84FEFFFF call <ransomware.wannahusky - copy.change_background>
      0040E011 B9 401E4100 mov ecx, ransomware.wannahusky - copy.411E40
      0040E012 C9 leave
      0040E013 E9 D09AFFFF jmp <ransomware.wannahusky - copy.execute_shell_cmd>
      0040E014 EB AD jmp ransomware.wannahusky - copy.40E052
      0040E015 90 nop
      0040E016 90 nop
      0040E017 90 nop
      0040E018 66:90 nop
      0040E019 66:90 nop
      0040E01A 66:90 nop
```

Registers and memory dump tabs are visible on the right side of the debugger interface.

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 from a debugger. The assembly code includes calls to `copy.append_String`, `copy.encrypt_!`, `copy.raw_new_string`, `copy.append_String`, `copy.encrypt_!`, `copy.read_cosmo_jpeg`, and `copy.41A078`. The memory dump shows the file content: "...This is a test file for exfil....". The file operations in Process Monitor show multiple ReadFile operations on the file `C:\Users\blue\Desktop\cosmo.jpeg`.

Time of Day	Process Name	PID	Operation	Path	Result	Detail
5.7484093 AM	Ransomware.w...	804	ReadFile	C:\Windows\SysWOW64\msvcrt.dll	SUCCESS	Offset: 361,472, Length: 28,672, I/O Flags: Non-cached,
1.3781749 AM	Ransomware.w...	804	Thread Create		SUCCESS	Thread ID: 5168
6.2196212 AM	Ransomware.w...	804	ReadFile	C:\Windows\SysWOW64\msvcrt.dll	SUCCESS	Offset 484,352, Length: 32,768, I/O Flags: Non-cached,
6.2228459 AM	Ransomware.w...	804	ReadFile	C:\Windows\SysWOW64\msvcrt.dll	SUCCESS	Offset 517,120, Length: 28,672, I/O Flags: Non-cached,
6.2234453 AM	Ransomware.w...	804	ReadFile	C:\Windows\SysWOW64\msvcrt.dll	SUCCESS	Offset 308,224, Length: 32,768, I/O Flags: Non-cached,
6.2245634 AM	Ransomware.w...	804	CreateFile	C:\Users\blue\Desktop\cosmo.jpeg	SUCCESS	Desired Access: Generic Read, Disposition: Open, Op
6.2246335 AM	Ransomware.w...	804	QueryStandard...	C:\Users\blue\Desktop\cosmo.jpeg	SUCCESS	AllocationSize: 32, EndOfFile: 30, NumberOfLinks: 1, De
6.2247910 AM	Ransomware.w...	804	ReadFile	C:\Windows\SysWOW64\msvcrt.dll	SUCCESS	Offset 283,648, Length: 24,576 I/O Flags: Non-cached,
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.

```

0040D8D3 EB D9 jmp <ransomware.wannahusky - copy.encrypt>
0040D8D5 C3 ret
0040D8D6 <rando> 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 OA jle ransomware.wannahusky - copy.40D8F6
0040D8EC B9 40A04100 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 j1 ransomware.wannahusky - copy.40D911
0040D903 899E 24020000 mov dword ptr ds:[esi+224],ebx
0040D906 89E5 E1 lea esp,dword ptr ss:[ebp+C]

```

Beginning of encrypt function

ebp=0066F928

.text:0040D8D6 ransomware.wannahusky - copy.exe:\$D8D6 #CCD6 <encrypt__>

Address	Hex	ASCII	Comments
009DB448	1E 00 00 00 1E 00 00 00 54 68 69 73 20 69 73 20This is	
009DB458	61 20 74 65 73 74 20 66 69 6C 65 20 66 6F 72 20	a test file for	
009DB468	65 78 66 69 6C 2E 00 04 00 00 00 E0 BB 41 00	exfil.....à»A	
009DB478	1E 00 00 00 1E 00 00 00 54 68 69 73 20 69 73 20This is	
009DB488	61 20 74 65 73 74 20 66 69 6C 65 20 66 6F 72 20	a test file for	
009DB48A	65 78 66 69 6C 2E 00 04 00 00 00 E0 BB 41 00	exfil.....à»A	
009DB48B	1E 00 00 00 1E 00 00 00 00 00 00 00 00 00 00 00	
009DB4C8	00 00 00 00 00 00 00 00 04 00 00 00 E0 BB 41 00à»A	

Address	Hex	ASCII	Comments
0066F8E0	00000020		return to ransomware.wanna
0066F8F4	0066FC24		
0066F8F8	00000020		
0066F8FC	0040C4B4		
0066F900	00000000		
0066F904	0000000F		
0066F908	009DB480	"This is a test file for	
0066F90C	0000001E		
0066F910	009DB448		
0066F914	00000000		
0066F918	0066F928		
0066F91C	00000000		
0066F920	009DB480		

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]
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 895424 04 mov dword ptr ss:[esp+4],edx
0040D92F 89C2 mov edx,eax
0040D931 C70424 20000000 mov dword ptr ss:[esp],20
0040D932 E8 0DE3FFFF call <ransomware.wannahusky - copy.encrypt__!1>
0040D933 BA 20000000 mov edx,20
0040D934 8D8E E4010000 lea ecx,dword ptr ds:[esi+1E4]
0040D935 83EC 0C sub esp,C
0040D936 E8 71EBFFFF call ransomware.wannahusky - copy.40C4C1
0040D937 3B7D 0C cmp edi,dword ptr ss:[ebp+C]
0040D938 72 12 jb ransomware.wannahusky - copy.40D967
0040D939 8B45 0C mov eax,dword ptr ss:[ebp+C]
0040D93A 89C2 01 mov dword ptr ss:[ebp+C],edi

```

Likely generates an xor key.

EIP: 0040D938

.text:0040D938 ransomware.wannahusky - copy.exe:\$D938 #CD38

Address	Hex	ASCII	Comments
009DB448	1E 00 00 00 1E 00 00 00 54 68 69 73 20 69 73 20This is	
009DB458	61 20 74 65 73 74 20 66 69 6C 65 20 66 6F 72 20	a test file for	
009DB468	65 78 66 69 6C 2E 00 04 00 00 00 E0 BB 41 00	exfil.....à»A	
009DB478	1E 00 00 00 1E 00 00 00 54 68 69 73 20 69 73 20This is	
009DB488	61 20 74 65 73 74 20 66 69 6C 65 20 66 6F 72 20	a test file for	
009DB48A	65 78 66 69 6C 2E 00 04 00 00 00 E0 BB 41 00	exfil.....à»A	
009DB48B	1E 00 00 00 1E 00 00 00 00 00 00 00 00 00 00 00	
009DB4C8	00 00 00 00 00 00 00 00 04 00 00 00 E0 BB 41 00à»A	
009DB4D8	1E 00 00 00 1E 00 00 00 00 00 00 00 00 00 00 00	
009DB4E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
009DB4F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
009DB508	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
009DB518	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	

Address	Hex	ASCII	Comments
0066F8E0	00000020		return to r
0066F8F4	0066FC24		
0066F8F8	00000020		
0066F8FC	0040C4B4		
0066F900	00000000		
0066F904	0000000F		
0066F908	009DB480	"This is a	
0066F90C	0000001E		
0066F910	009DB448		
0066F914	00000000		
0066F918	0066F928		
0066F91C	00000000		
0066F920	009DB480		
0066F924	009DB478		
0066F928	0066F928		
0066F92C	0040D920	return to r	
0066F930	009DB4B0		
0066F934	0000001E		

WannaHusky_RansomWare

May 2024

v1.0

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 1F0000 mov dword ptr ss:[esp+4], IF
0040D98B 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 F86CFFFF call ransomware.wannahusky - copy.4046B1
0040D9B9 8945 DC mov eax,dword ptr ss:[ebp-24]

```

A red box highlights the instruction `xor dl,byte ptr ds:[esi+ebx+204]`. A callout box points to it with the text "esi+ebx+204 contains the xor key". Another callout box points to the `xor` instruction with the text "XOR encryption routine".

Memory Dump View:

The dump pane shows memory starting at address 0066FBF4. A red box highlights the first 16 bytes of data: `00 CF FC 30 FC F3 CO 3F 00 FC 0E 00 00 00 .I00000A?.`. A callout box points to it with the text "XOR key used for the first 16 bytes of data".

The EIP register is at 0040D99C. A red box highlights the instruction `xor dl,byte ptr ds:[esi+ebx+204]`. A callout box points to it with the text "First byte of data is loaded into DL register which is the letter 'T'".

The dump pane continues to show memory starting at address 009DB480. A red box highlights the instruction `xor dl,byte ptr ds:[esi+ebx+204]`. A callout box points to it with the text "First byte of data is xor'd with byte value stored in esi+ebx+204 which is the 1st byte of xor key".

Further down, a red box highlights the instruction `jmp ransomware.wannahusky - copy.40D8FE`. A callout box points to it with the text "and operation of ebx with F signifies that whenever ebx goes over a value of 16 its content are cleared to 0.". Another callout box points to the `jmp` instruction with the text "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."

Callouts:

- esi+ebx+204 contains the xor key
- XOR encryption routine
- XOR key used for the first 16 bytes of data
- First byte of data is loaded into DL register which is the letter "T"
- First byte of data is xor'd with byte value stored in esi+ebx+204 which is the 1st byte of xor key
- and operation of ebx with F signifies that whenever ebx goes over a value of 16 its content are 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.

XOR'ed data is stored in the location pointed by eax+edi

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

Assembly View:

```

0040D983 C74424 04 1F0000 mov dword ptr ss:[esp+4], IF
0040D98B 891C24 mov dword ptr ss:[esp], ebx
0040D98E 8A1438 mov dl,byte ptr ds:[eax+edi]
0040D993 8845 08 mov eax,dword ptr ss:[ebp-20]
0040D996 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 F86CFFFF call ransomware.wannahusky - copy.4046B1
0040D9B9 8945 DC mov eax,dword ptr ss:[ebp-24]

```

A red box highlights the instruction `xor dl,byte ptr ds:[esi+ebx+204]`. A callout box points to it with the text "Encrypted data is being stored here. 1st 16 bytes of data."

Memory Dump View:

The dump pane shows memory starting at address 009DB480. A red box highlights the instruction `xor dl,byte ptr ds:[esi+ebx+204]`. A callout box points to it with the text "Encrypted data is being stored here. 1st 16 bytes of data."

The dump pane continues to show memory starting at address 009DB480. A red box highlights the instruction `xor dl,byte ptr ds:[esi+ebx+204]`. A callout box points to it with the text "Encrypted data is being stored here. 1st 16 bytes of data."

Callouts:

- Encrypted data is being stored here. 1st 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.

Call to encrypt_!1 function to generate the next key for encrypting the next 16 bytes.

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

Key Generation

First xor key generation :

Key generator function call

First xor key for the first 16 bytes

Second xor key generation :

Key generator function

Second xor key for the next 16 bytes

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 the OllyDbg debugger interface. The assembly pane displays the encryption routine, with the instruction at address 0040D910 highlighted in yellow. A tooltip for this instruction states: "Returns after completing the encryption routine." The memory dump pane below shows two blocks of data: plain text from cosmo.jpeg and encrypted data. The plain text is labeled "plain text data from cosmo.jpeg" and the encrypted data is labeled "Encrypted data".

Assembly code (partial):

```
0040D8FE 3B7D E4 cmp edi,dword ptr ss:[ebp-1C]
0040D901 7C 0E j1 ransomware.wannahusky - copy.40D911
0040D903 89E 24020000 mov dword ptr ds:[esi+224],ebx
0040D905 80D5 F4 lea esp,dword ptr ss:[ebp-C]
0040D907 5B pop ebx
0040D90D 5E pop esi
0040D90E 5F pop edi
0040D90F 5D pop ebp
0040D910 C3 ret Returns after completing the encryption routine.
0040D911 85DB test ebx,ebx
0040D913 75 3B jne ransomware.wannahusky - copy.40D950
0040D915 89D6 04020000 lea edx,dword ptr ds:[esi+204]
0040D91B C74424 08 20000 mov dword ptr ss:[esp+8],20
0040D923 8086 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
0040D938 E8 0DE3FFFF call <ransomware.wannahusky - copy.encrypt__!1>
0040D93D BA 20000000 mov edx,20
0040D942 808E E4010000 lea ecx,dword ptr ds:[esi+1E4]
0040D948 83EC 0C sub esp,C
0040D94B E8 71EBFFFF 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
```

Memory dump (partial):

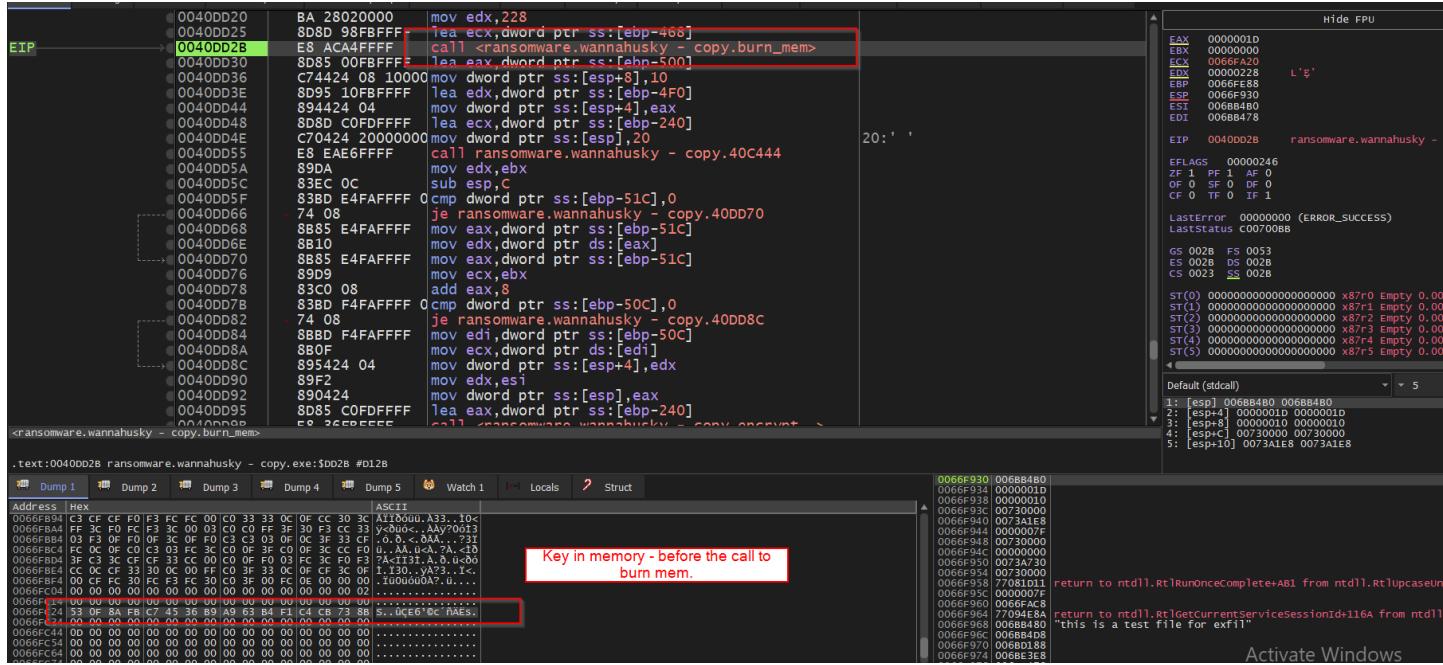
Address	Hex	ASCII
009DB460	69 6C 65 20 66 6F 72 20 65 78 66 69 6C 2E 00 00	file for exfil...
009DB470	04 00 00 00 E0 BB 41 00 1E 00 00 00 1E 00 00 00	This is a test f
009DB480	54 68 69 73 20 69 73 20 61 20 74 65 73 74 20 66	ile for exfil... This is a test f
009DB490	69 6C 65 20 66 6F 72 20 65 78 66 69 6C 2E 00 00	ile for exfil...
009DB4A0	04 00 00 00 E0 BB 41 00 1E 00 00 00 1E 00 00 00	This is a test f
009DB4B0	88 FD A9 08 82 29 FA A9 CC 68 D6 71 E1 F0 00 E1	ile for exfil... .y.)ü@ihqad.ä
009DB4C0	3A 63 EF DB A1 2A 44 99 CC 1B D2 98 A8 E5 00 00	:ci0;^D.i.o. ä..
009DB4D0	00 00 00 00 E0 BB 41 00 1E 00 00 00 1E 00 00 00a/A.....
009DB4E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
009DB4F0	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.

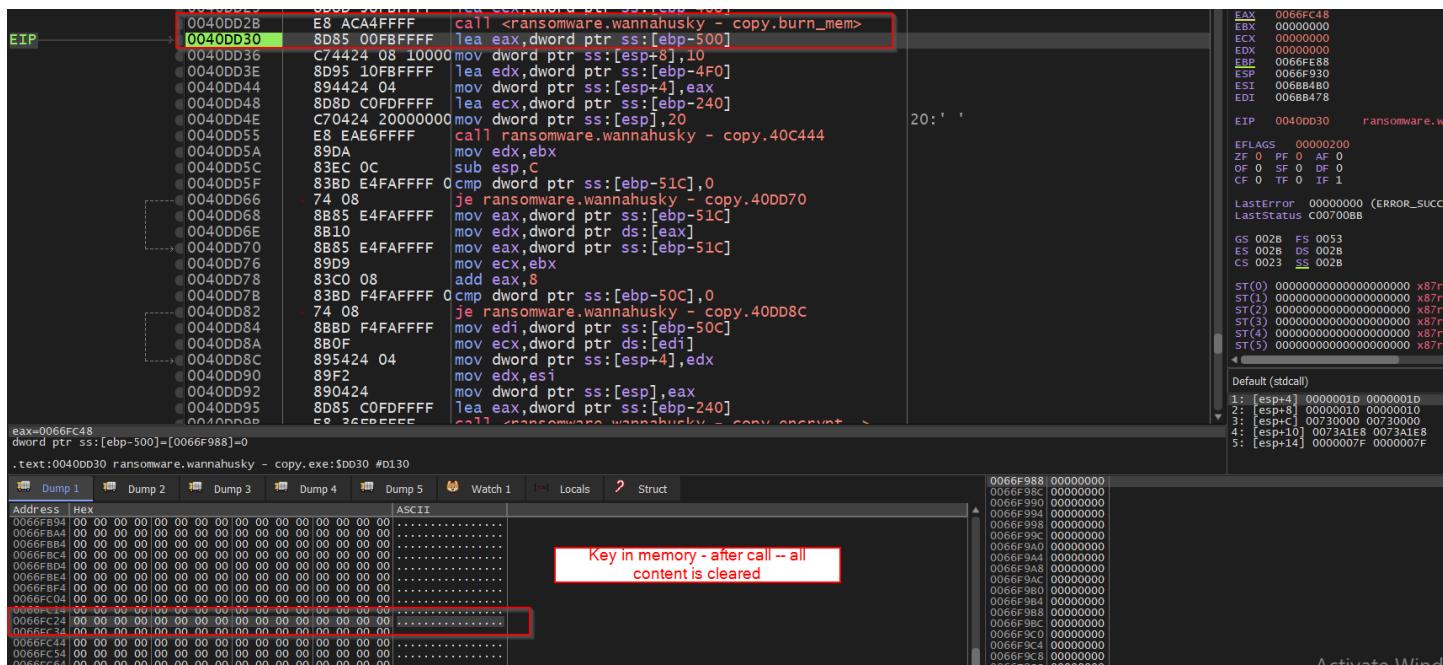
So I saved both the keys from the memory in a file on my desktop so as to decrypt the encrypted data found in the wannahusky file.

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

Before Call :



After call :



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.

```

    mov dword ptr ss:[esp],ebx
    call ransomware.wannahusky - copy.405FDD
    mov 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+eax+204]
    mov byte ptr ds:[eax+edi],dl
    lea eax,dword ptr ds:[evi+1]
    xor edi,eax
    jns ransomware.wannahusky - copy.40D9BC
    test eax,eax
    jns ransomware.wannahusky - copy.40D9BC
    mov 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
    mov esp,ebp
    push edi
    push esi
    push ebx
    sub esp,1C
    xor eax,dword ptr ss:[ebp-1C]
    jb ransomware.wannahusky - copy.40D97E
    mov eax,dword ptr ss:[ebp-1C]
    lea edx,dword ptr ds:[eax-1]
    mov dword ptr ss:[esp+4],edx
    call ransomware.wannahusky - copy.405FDD
    cmp ebx,1F
    jbe ransomware.wannahusky - copy.40D993
    mov dword ptr ss:[esp-4],1F
    mov dword ptr ss:[esp],ebx
    call ransomware.wannahusky - copy.405FDD
    mov 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+ebx+204]
    xor byte ptr ds:[eax+edi],dl
    Test eax,dword ptr ds:[edi+1]
    xor edi,eax
    jns ransomware.wannahusky - copy.40D9BC
    test eax,eax
    jns ransomware.wannahusky - copy.40D9BC
    mov 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

```

Same encryption routine, but this time the data passed to eax/edx contains the encrypted text from previous call.

read_write??

Second round of encryption/decryption test

Encrypted...

[ebp+8]:"this is a test file"

```

    cmp edi,dword ptr ss:[ebp-1C]
    jb ransomware.wannahusky - copy.40D97E
    mov eax,dword ptr ss:[ebp-1C]
    lea edx,dword ptr ds:[eax-1]
    mov dword ptr ss:[esp+4],edx
    call ransomware.wannahusky - copy.405FDD
    cmp ebx,1F
    jbe ransomware.wannahusky - copy.40D993
    mov dword ptr ss:[esp-4],1F
    mov dword ptr ss:[esp],ebx
    call ransomware.wannahusky - copy.405FDD
    mov 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+ebx+204]
    xor byte ptr ds:[eax+edi],dl
    Test eax,dword ptr ds:[edi+1]
    xor edi,eax
    jns ransomware.wannahusky - copy.40D9BC
    test eax,eax
    jns ransomware.wannahusky - copy.40D9BC
    mov 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

```

[ebp+8]:"this is a test file"

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.

```

00400D8 880F           mov ecx,dword ptr ds:[edi]
00400D8 895424 04      mov dword ptr ss:[esp+4],edx
00400D9 89F2           mov edx,esi
00400D9 890424          mov dword ptr ss:[esp],eax
00400D9 8085 C0FDFFFF  lea eax,dword ptr ss:[ebp-240]
00400D9 E8 36FBFFFF    call <ransomware.wannahusky - copy.encrypt_>
00400DA BA 28020000    mov edx,228
00400DA 88D8 C0FDFFFF  lea ecx,dword ptr ss:[ebp-240]
00400DA E8 2CA4FFFF    call ransomware.wannahusky - copy.4081DC
00400DB 83BD F4FAFFFF  cmp dword ptr ss:[ebp-50C],0
00400DB 74 08           je ransomware.wannahusky - copy.40DDC1
00400DB 8885 F4FAFFFF  mov eax,dword ptr ss:[ebp-50C]
00400DB 8B18           mov ebx,dword ptr ds:[eax]
00400DB C70424 00000000 mov dword ptr ss:[esp],0
00400DC 89F1           mov ecx,esi
00400DC 89DA           mov edx,ebx
00400DC E8 389EFFFF    call <ransomware.wannahusky - copy.encode_b64>
00400DE 89C3           mov ebx,ecx
00400DE A1 70484200    mov eax,dword ptr ds:[424870]
00400DE 51              push ecx
00400DE B9 18000000    mov ecx,18
00400DE 85C0           test eax,eax
00400DE 74 05           je ransomware.wannahusky - copy.40DDE7
00400DE 8B08           mov ecx,dword ptr ds:[eax]
00400DE 83C1 18         add ecx,18
00400DE E8 7771FFFF    call ransomware.wannahusky - copy.404F63
00400DE 8B15 70484200    mov edx,dword ptr ds:[424870]
00400DF E8 D8FAFFFF    call ransomware.wannahusky - copy.40D8CF
00400DF BA 00A04100    mov edx,ransomware.wannahusky - copy.41A000
00400DF E8 ADFAFFFF    call <ransomware.wannahusky - copy.encrypt_!>
00400DF 89DA           mov edx,ebx

```

Registers:

- EAX 001884A8
- EBX 0000001E
- ECX 001884B0
- EDX 00000010
- EBP 0068F835
- ESP 0066F930
- ESI 00188480
- EDI 001884A8
- EIP 00400DCC ransomware.wannahusky - copy.00400DCC

Flags:

- EFLAGS 000000202
- ZF 0 PF 0 AF 0
- OF 0 SF 0 DF 0
- CF 0 TF 0 IF 1

LastError: 00000000 (ERROR_SUCCESS)

LastStatus: C007008B

Stack:

- GS 002B FS 0053
- ES 002B DS 002B
- CS 0023 SS 002B

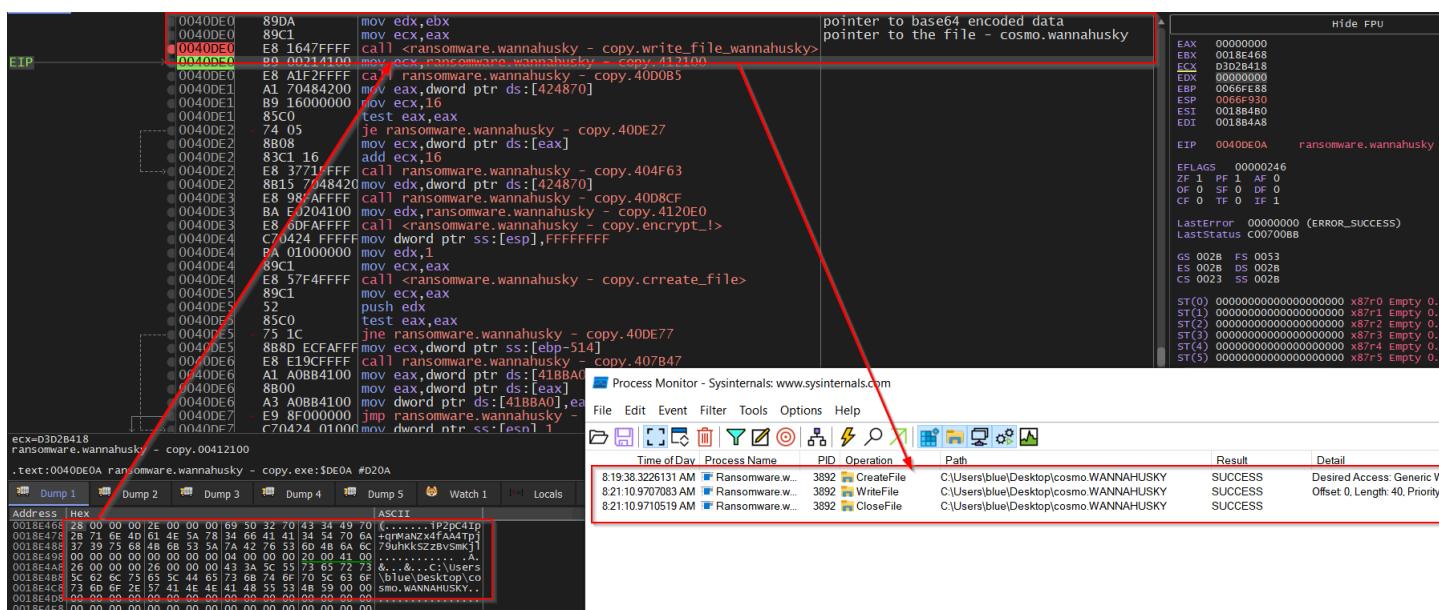
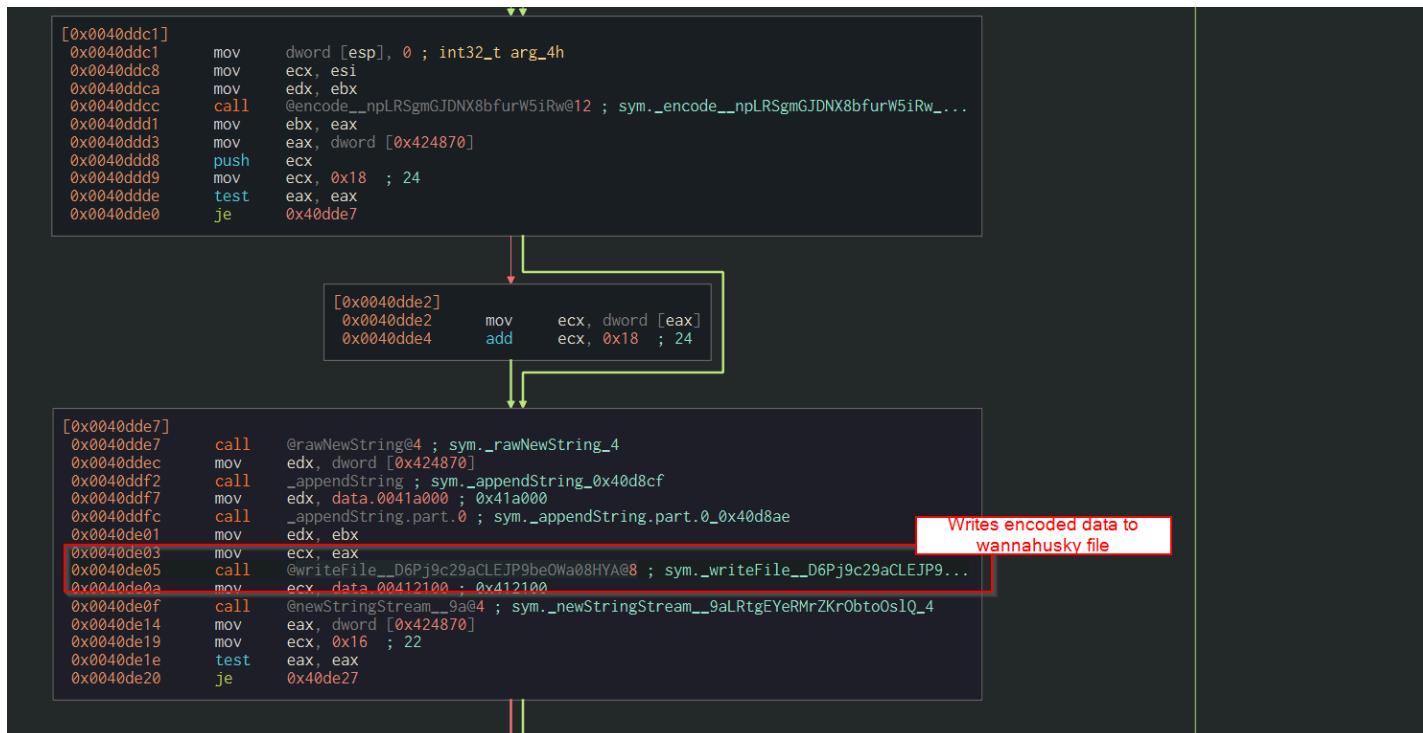
Default (stdcall):

- 1: [esp] 00000000 00000000
- 2: [esp+4] 0000001E 0000001E
- 3: [esp+8] 00000010 00000010
- 4: [esp+C] 007B0000 007B0000
- 5: [esp+10] 007BA1E8 007BA1E8

Encrypted data: 0066F97C 001884A8

Encoded data after the call: 0066F934 0000001E

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`



WannaHusky_RansomWare
May 2024
v1.0

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 specific assembly instructions and memory dump sections, which are then linked to entries in a Process Monitor log.

Assembly Code Annotations:

- A red box highlights the instruction `call <ransomware.wannahusky - copy.write_file>`. A callout points to the Process Monitor log entry for `Ransomware.w...` at PID 3892, showing a `CreateFile` operation on `C:\Users\blue\Desktop\cosmo WANNAHUSKY`.
- A red box highlights the instruction `call <ransomware.wannahusky - copy.execute_shell_cmd>`. A callout points to the Process Monitor log entry for `Ransomware.w...` at PID 3892, showing a `WriteFile` operation on `C:\Users\blue\Desktop\WANNAHUSKY.png`.
- A red box highlights the instruction `call <ransomware.wannahusky - copy.append_str>`. A callout points to the Process Monitor log entry for `Ransomware.w...` at PID 3892, showing a `WriteFile` operation on `C:\Users\blue\Desktop\WANNAHUSKY.png`.
- A red box highlights the instruction `call <ransomware.wannahusky - copy.append_str>`. A callout points to the Process Monitor log entry for `Ransomware.w...` at PID 3892, showing a `CloseFile` operation on `C:\Users\blue\Desktop\WANNAHUSKY.png`.
- A red box highlights the instruction `call <ransomware.wannahusky - copy.append_str>`. A callout points to the Process Monitor log entry for `Ransomware.w...` at PID 3892, showing a `CreateFile` operation on `C:\Users\blue\Desktop\cosmo.jpeg`.
- A red box highlights the instruction `call <ransomware.wannahusky - copy.append_str>`. A callout points to the Process Monitor log entry for `Ransomware.w...` at PID 3892, showing a `QueryAttribute...` operation on `C:\Users\blue\Desktop\cosmo.jpeg`.
- A red box highlights the instruction `call <ransomware.wannahusky - copy.append_str>`. A callout points to the Process Monitor log entry for `Ransomware.w...` at PID 3892, showing a `SetDisposition...` operation on `C:\Users\blue\Desktop\cosmo.jpeg`.
- A red box highlights the instruction `call <ransomware.wannahusky - copy.append_str>`. A callout points to the Process Monitor log entry for `Ransomware.w...` at PID 3892, showing a `CloseFile` operation on `C:\Users\blue\Desktop\cosmo.jpeg`.
- A red box highlights the instruction `call <ransomware.wannahusky - copy.append_str>`. A callout points to the Process Monitor log entry for `Ransomware.w...` at PID 3892, showing a `CreateFile` operation on `C:\Users\blue\Desktop\ps1.ps1`.
- A red box highlights the instruction `call <ransomware.wannahusky - copy.append_str>`. A callout points to the Process Monitor log entry for `Ransomware.w...` at PID 3892, showing a `WriteFile` operation on `C:\Users\blue\Desktop\ps1.ps1`.
- A red box highlights the instruction `call <ransomware.wannahusky - copy.append_str>`. A callout points to the Process Monitor log entry for `Ransomware.w...` at PID 3892, showing a `CloseFile` operation on `C:\Users\blue\Desktop\ps1.ps1`.

Memory Dump Annotations:

- A red box highlights the memory dump section containing the PowerShell command `...owershe...wannahusky.ps1`. This corresponds to the `WriteFile` operation on `C:\Users\blue\Desktop\ps1.ps1` in the Process Monitor log.

Process Monitor Log (Selected Entries):

Time of Day	Process Name	PID	Operation	Path	Result	Detail
8:19:46 3226131 AM	Ransomware.w...	3892	CreateFile	C:\Users\blue\Desktop\cosmo WANNAHUSKY	SUCCESS	
8:21:09 7/07/083 AM	Ransomware.w...	3892	WriteFile	C:\Users\blue\Desktop\WANNAHUSKY.png	SUCCESS	
8:25:38 7/17/6614 AM	Ransomware.w...	3892	WriteFile	C:\Users\blue\Desktop\WANNAHUSKY.png	SUCCESS	
8:26:15 8/12/0580 AM	Ransomware.w...	3892	CloseFile	C:\Users\blue\Desktop\WANNAHUSKY.png	SUCCESS	
8:26:33 6/8/37381 AM	Ransomware.w...	3892	CreateFile	C:\Users\blue\Desktop\cosmo.jpeg	SUCCESS	
8:26:33 6/8/38227 AM	Ransomware.w...	3892	QueryAttribute...	C:\Users\blue\Desktop\cosmo.jpeg	SUCCESS	Attrib...
8:26:33 6/8/38592 AM	Ransomware.w...	3892	SetDisposition...	C:\Users\blue\Desktop\cosmo.jpeg	SUCCESS	Flag...
8:26:33 6/8/38962 AM	Ransomware.w...	3892	CloseFile	C:\Users\blue\Desktop\cosmo.jpeg	SUCCESS	
8:27:28 09/84758 AM	Ransomware.w...	3892	CreateFile	C:\Users\blue\Desktop\ps1.ps1	SUCCESS	
8:27:58 0/100141 AM	Ransomware.w...	3892	WriteFile	C:\Users\blue\Desktop\ps1.ps1	SUCCESS	
8:27:58 0/14/391 AM	Ransomware.w...	3892	CloseFile	C:\Users\blue\Desktop\ps1.ps1	SUCCESS	

Time of Day	Process Name	PID	Operation	Path	Result	Detail
8:38:37 4437566 AM	Ransomware.w...	3892	Process Create	C:\Windows\SysWOW64\cmd.exe	SUCCESS	PID: 4428, Command line: C:\Windows\system32\cmd.exe /c powershell C:\Users\blue\Desktop\ps1.ps1

WannaHusky_RansomWare
May 2024
v1.0

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>

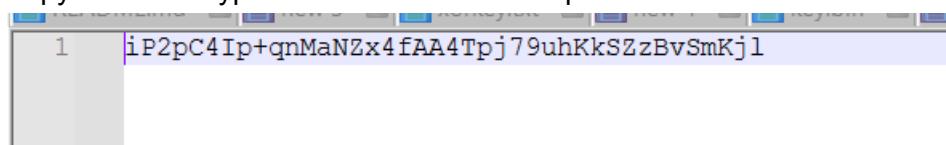
hurry! you have 24 hours before we delete cosmo

WannaHusky_RansomWare
May 2024
v1.0

Command Execution

And then finally executes cmd /tree C: command that lists all the file system paths for c drive in a tree format -- which is the window that we see during initial detonation.

Copy of the encrypted cosmo file on desktop :



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 :

```
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+qnMaNZx4fAA4Tpj79uhKkS2zBvSmKj1'

    # 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.'  
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

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)

}
```