

# Practical Malware Analysis & Triage

## Malware Analysis Report

WannaCry - RansomWare

May 2024 | AnikshaShetty | v1.0



# Table of Contents

Table of Contents .....	2
Executive Summary.....	4
High-Level Technical Summary .....	5
Execution flow of tasksche.exe.....	7
Malware Composition.....	8
Ransomware.wannacry.exe .....	8
*tasksche.exe.....	8
*taskdl.exe.....	8
*taskse.exe:.....	8
*t.wnry.....	8
*b.wnry.....	9
*c.wnry .....	9
*r.wnry.....	9
*s.wnry .....	9
*u.wnry.....	9
Basic Static Analysis.....	10
Floss – String Analysis.....	10
PEStudio .....	12
Basic Dynamic Analysis.....	14
Initial Detonation with InetSim .....	14
Detonation without inetsim.....	14
Procmon & TCPView analysis.....	15
Advanced Static Analysis.....	19
Initial Kill Switch.....	19
Real Entry/ Main execution call.....	20
Create_service_drop_next_stage_function call .....	20
Creation of next stage payload .....	22
Advanced Dynamic Analysis.....	23
Debugging Wannacry.exe – Initial payload.....	23
Kill Switch functionality .....	23
Service Creation:.....	23
Evidence of service being created.....	24
Write/Rename Next stage payload .....	24
Debugging tasksche.exe – next stage payload .....	26
Stage Folder Creation.....	26
Persistence – Creation of service.....	27



High level overview of the main functions of tasksche.exe .....	29
Detailed view of extracted files from 2058 resource in tasksche.exe.....	29
List of extracted files .....	30
Command Execution.....	34
Unpack the t.wnry file .....	34
<b>Indicators of Compromise .....</b>	<b>38</b>
Network Indicators .....	38
Host-based Indicators .....	38
<b>Rules &amp; Signatures.....</b>	<b>40</b>
A. Yara Rules .....	40
B. Callback URLs .....	40



## Executive Summary

SHA256 hash	24d004a104d4d54034dbcffc2a4b19a11f39008a575aa614ea04703480b1022c
-------------	--

WannaCry is a Ransomware cryptoworm sample first identified on May 2017. It is a 32 bit C++ compiled program that runs on both x32 and x64 bit windows operating system. It consists of multiple payloads that encrypts files, and exploits the EternalBlue vulnerability to spread across networks. Symptoms of infection include file encryption with “.wncry” extensions, a changed desktop background indicating encryption, and prompts for Bitcoin ransom payment via a decryptor program.

YARA signature rules are attached in Appendix A. Malware sample and hashes have been submitted to VirusTotal for further examination.



# High-Level Technical Summary

WannaCry.exe incorporates a unique safeguard: a killswitch mechanism. When executed, the program attempts to connect to a non-existent domain. If the connection succeeds, the program terminates without further action. However, if the connection fails, it continues with its intended functions. This ingenious tactic is believed to thwart sandbox environments and halt execution when under analysis.

Persistence is achieved by creating a service called Microsoft security center 2.0 (mssecsvc) which launches the original wannacry.exe on reboot. And then writes the next stage payload tasksche.exe which drops further resources for enumeration and encryption routines.

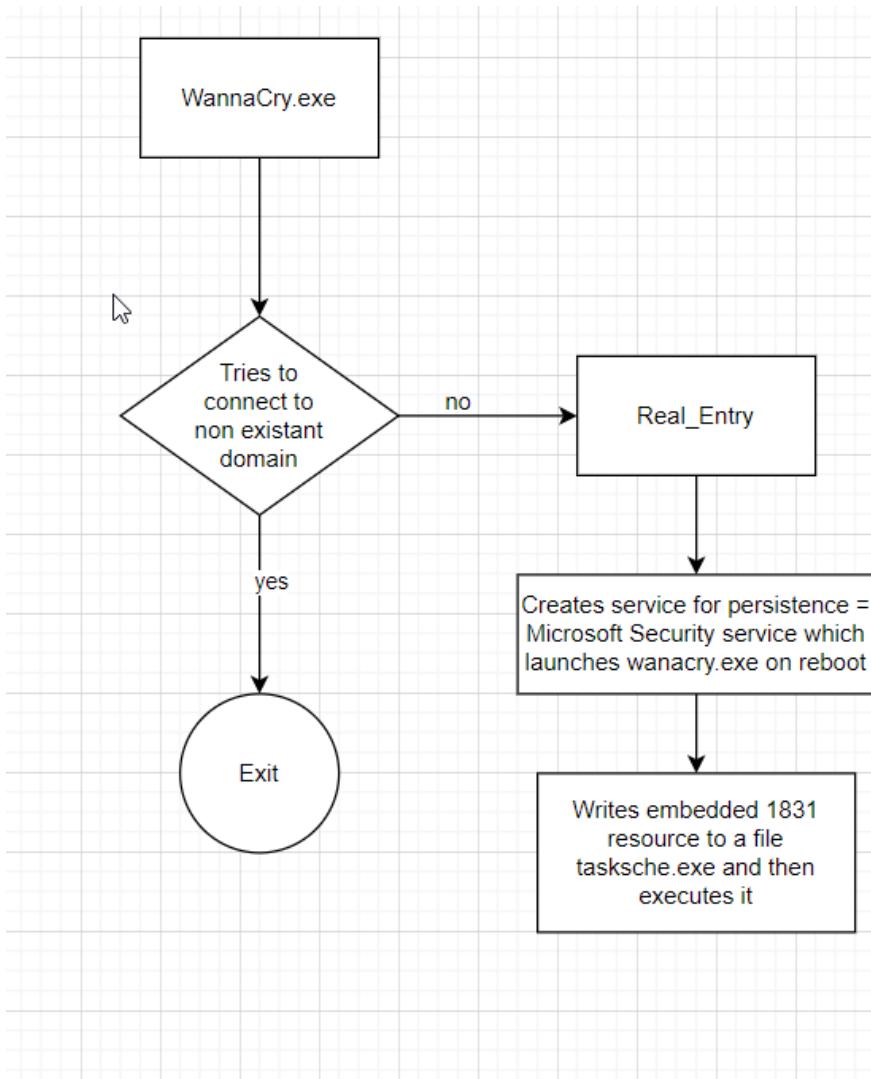


Figure 1 Initial payload wannacry execution flow



Tasksche.exe initially creates a random alpha numeric string based on computer name which is later used as a staging folder and then checks for any arguments passed to the program. If there is an argument, then creates hidden staged folder in program data directory with the random name obtained from previous function and then copies itself to this directory and creates a service with the same name to run tasksche.exe and then starts this service.

If no argument is passed then tasksche.exe, creates a reg key in HKLM/HKCU Software hive with wd value set to current working directory. And then loads and unzips password protected zip file from resource 2058 consisting of config files, encryption routine, decryption routine, background images and the text files containing the msg to be displayed in different languages. After which it writes a bitcoin address to the config file – c.wnry created in previous step. Then executes two commands to make the directory hidden and grant all access to everyone on all subdirectories.

Then uses an embedded rsa and aes key to decrypt the packed t.wnry sample and then executes it. T.wnry was found to be a dll file constituting the encryption routine for the malware. It searches for all the files of interest and encrypts it with RSA AES algorithm either using an embedded key or key file(0000000.pky / 0000000.eky).

There are additional support programs that are dropped which provides different functionality like – deletion of programs, enumeration and propagation through network, config files containing the c2 and decryptor program to show the users that the files are decryptable. See fig2. for execution flow of tasksche.exe

All the details containing the files dropped and their functionality has been detailed in the next section.



### Execution flow of tasksche.exe

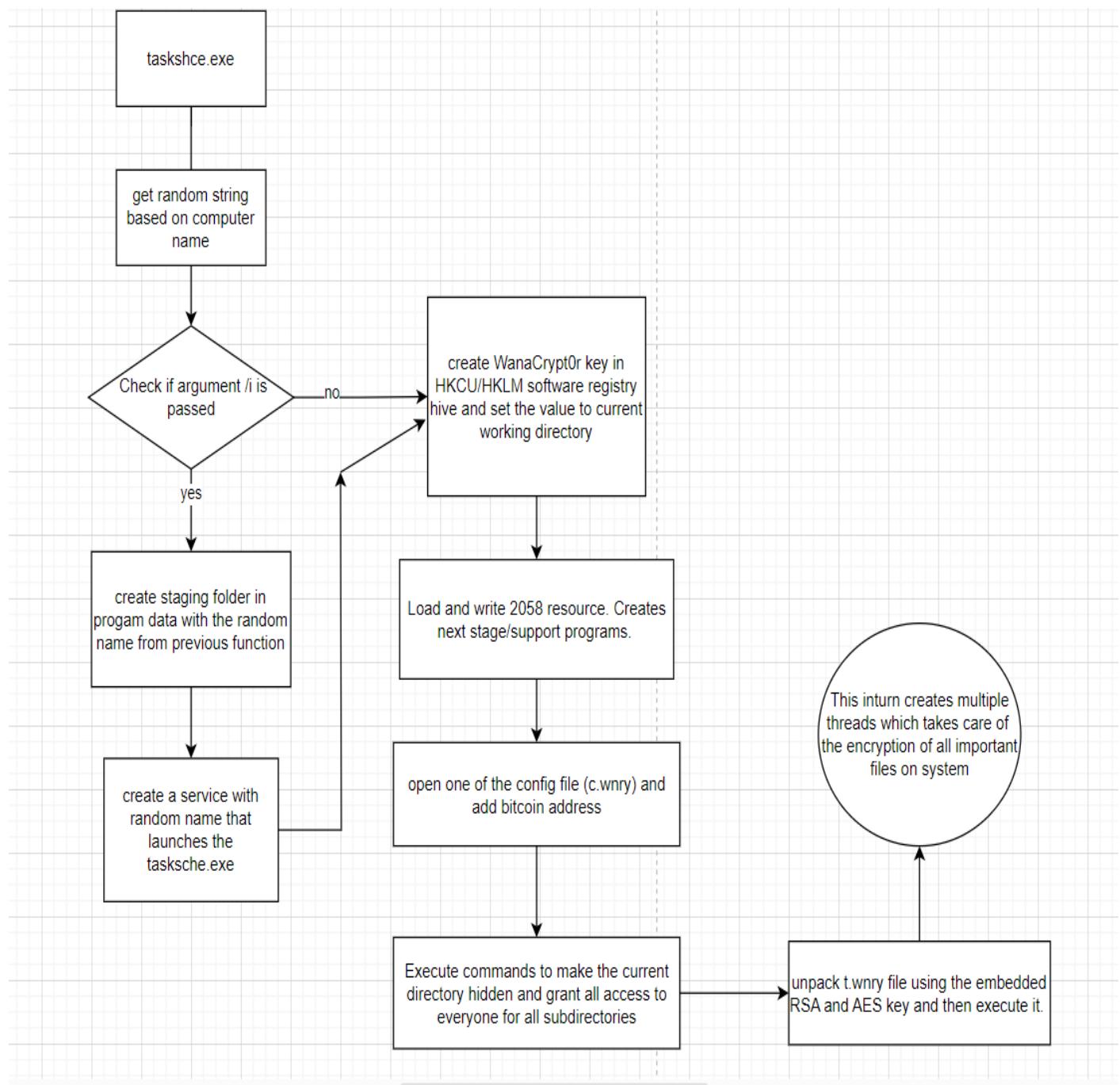


Figure 2 Second stage tasksche.exe execution flow



# Malware Composition

Ransomware consists of the following components :

File Name	FileType	SHA256 Hash
<b>Ransomware.wannacry.exe</b>	PE32 Windows	24d004a104d4d54034dbcffc2a4b19a11f39008a575aa614ea04703480b1022c
<b>*tasksche.exe</b>	PE32 Windows	ed01ebfb9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa
<b>*taskdl.exe</b>	PE32 Windows	4a468603fdcb7a2eb5770705898cf9ef37aade532a7964642ecd705a74794b79
<b>*taskse.exe</b>	PE32 Windows	2ca2d550e603d74dedda03156023135b38da3630cb014e3d00b1263358c5f00d
<b>*t.wnry</b>	Packed dll file	97ebce49b14c46beb9ec2448d00e1e397123b256e2be9eba5140688e7bc0ae6
<b>*b.wnry</b>	Bit map image	d5e0e8694ddc0548d8e6b87c83d50f4ab85c1debadb106d6a6a794c3e746f4fa
<b>*c.wnry</b>	Config file	055c7760512c98c8d51e4427227fe2a7ea3b34ee63178fe78631fa8aa6d15622
<b>*r.wnry</b>	Ransom note	402751fa49e0cb68fe052cb3db87b05e71c1d950984d339940cf6b29409f2a7c
<b>*s.wnry</b>	Tor software zip archive	e18fdd912dfe5b45776e68d578c3af3547886cf1353d7086c8bee037436dff4b
<b>*u.wnry</b>	PE32 Windows	b9c5d4339809e0ad9a00d4d3dd26fdf44a32819a54abf846bb9b560d81391c25
<b>*t.wnry_unpacked</b>	Decrypted t.wnry file – dll	1be0b96d502c268cb40da97a16952d89674a9329cb60bac81a96e01cf7356830
<b>*msg/</b>	Directory containing msg file in different languages	

## Ransomware.wannacry.exe

The initial executable that checks the killswitch and then drops the next stage payload and creates a service for persistence.

### \*tasksche.exe

Second stage payload with an embedded resource containing a password protected archive consisting of next stage support programs that helps in encryptor program, background image, decryptor program, config files containing the command and control servers, bitcoin addresses, msg files and the ransom note.

### \*taskdl.exe

Support program to delete all the temporary files.

### \*taskse.exe:

Copy of tor.exe

### \*t.wnry

Encrypted dll containing the encryption routine



[\*\*\\*b.wnry\*\*](#)

Bitmap image containing the desktop background that displays the ransom note.

[\*\*\\*c.wnry\*\*](#)

Config file consisting the c2 servers (tor/.onion domains) and the bitcoin address

[\*\*\\*r.wnry\*\*](#)

Ransom note

[\*\*\\*s.wnry\*\*](#)

Zip file containing the Tor software and its dependency files

[\*\*\\*u.wnry\*\*](#)

Unknown

[\*\*\\*msg/\*\*](#)

Directory containing msg files displaying the instructions to recover the files in different languages.

Detailed view of these files can be found in [extracted files](#) section in this document.



# Basic Static Analysis

{Screenshots and description about basic static artifacts and methods}

## Floss – String Analysis

- There are multiple DOS headers. A domain that it connects to.
- Tries to access a network IPC share
- Multiple blocks of base 64 encoded /encrypted strings
- Message to be displayed in different languages
- Looks for various types of file extension

EXE's/cmd of note :

```
tasksche.exe
cmd.exe /c "%s"
tasksche.exe
TaskStart
t.wnry
icacls . /grant Everyone:F /T /C /Q
attrib +h .
WNcry@2017
```

```
taskdl.exe
taskse.exe
lhdfrgui.exe
diskpart.exe
mssecsvc.exe
```

IPC share/network details :

```
\192.168.56.20\IPC$  
\172.16.99.5\IPC$  
hxxp://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwegwae.com
```

```
\%s\IPC$  
Microsoft Base Cryptographic Provider v1.0  
%d.%d.%d.%d
```

```
mssecsvc2.0
Microsoft Security Center (2.0) Service
%s -m security
C:\%s\queriuwjhrf
C:\%s\%s
WINDOWS
tasksche.exe
```

```
CloseHandle
WriteFile
CreateFileA
```

```
CreateProcessA
http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwegwae.com
!This program cannot be run in DOS mode.
```

```
Rich
.text
`.rdata
@.data
.rsrc
```

Figure 3 Floss Output - interesting strings capturing embedded resources and commands and killswitch domain



.der .pfx .key .crt .csr .p12 .pem .odt .ott .sxw .stw .uot .3ds .max .3dm .ods .ots .sxc .stc .dif .slk .wb2 .odp .otp .sxd .std .uop .odg .otg .sxm .mml .lay .lay6 .asc .sqlite3 .sqlitedb .sql .accdb .mdb .dbf .odb .frm .myd .myi .ibd .mdf .ldf .sln .suo .cpp .pas .asm .cmd .bat .ps1 .vbs .dip .dch .sch .brd .jsp .php .asp .java .jar .class .mp3 .wav .swf .fla .wmv .mpg .vob .mpeg .ASF .avi .mov .mp4 .3gp .mkv .3g2 .flv .wma .mid .m3u .m4u .djvu .svg .psd .nef .tiff .tif .cgm .raw .gif .png .bmp .jpg .jpeg .vcd .iso .backup .zip .rar .tgz .tar .bak .tbk .bz2 .PAQ .ARC .aes .gpg .vmx .vmdk .vdi .sldm .sldx .sti .sxi .602 .hwp .snt .onetoc2 .dwg .pdf .wk1 .wks .123 .rtf .csv .txt .vsdx .vsd .edb .eml .msg .ost .pst .potm .potx .ppam .ppsx .ppsm .pps .pot .pptm .pptx .ppt .xltx .xlt .xlc .xlm .xlt .xlw .xlsb .xltm .xlsx .xls .dotx .dotm .dot .docm .docb .docx .doc

Figure 4 File Extensions

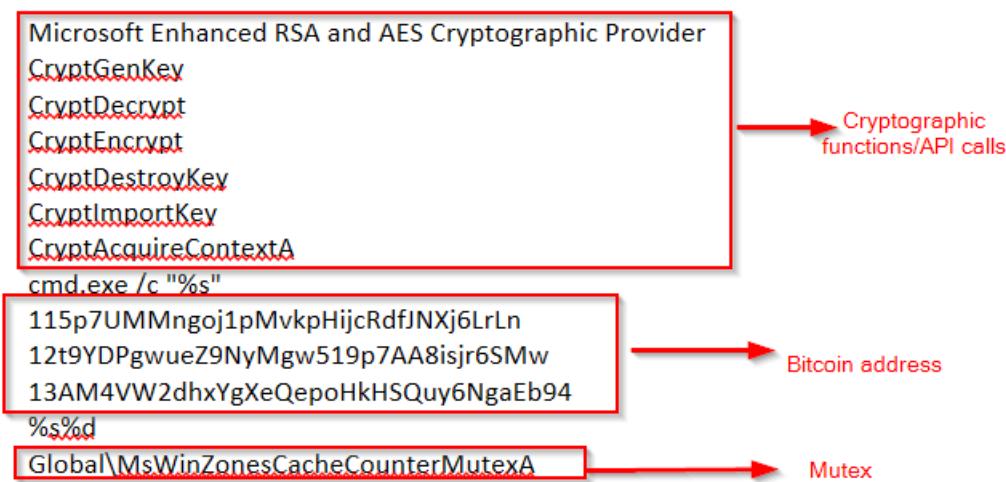


Figure 5 Floss Output interesting strings capturing bitcoin address, cryptographic calls and mutex



## PEStudio

It is a 32 bit program compiled on 20<sup>th</sup> Nov 2010 written in c++ language.

The screenshot shows the PEStudio interface with the following details:

- File Path:** c:\users\blue\Desktop\ransomware.wannacry.exe
- Properties Table:**
  - signature:** Microsoft Visual C++ v6.0
  - tooling:** Visual Studio 6.0
  - file-type:** executable
  - cpu:** 32-bit
- Stamps Table:**
  - compiler-stamp:** Sat Nov 20 09:03:08 2010 | UTC
  - debug-stamp, resource-stamp, import-stamp, export-stamp: n/a
- Names Table:**
  - file:** c:\users\blue\Desktop\ransomware.wannacry.exe.malz
  - debug, export: n/a
- Hash:** sha256: 24D004A104D4D54034DBCFFC2A4B19A11F39008A575AA614EA04703480B1022C
- Bottom Status:** cpu: 32-bit | file-type: executable | subsystem: GUI

Figure 6 PE Studio Output

Not much difference between raw size and virtual size of the binary ---> not packed.

The screenshot shows the TPE Studio interface with the following details:

- File Path:** c:\users\blue\Desktop\ransomware.wannacry.exe
- Sections Table:**

section	value	value	value	value
	section[0]	section[1]	section[2]	section[3]
name	.text	.rdata	.data	.rsrc
footprint > sha256	7609ECC798A357DD1A2F01...	532E9419F23EAF5EB0E882B...	6F93FB1B241A990ECC281F9...	1EFE677209C1284
entropy	6.135	3.504	6.100	7.995
file-ratio (99.89%)	0.99 %	0.11 %	4.29 %	94.50 %
raw-address (begin)	0x00001000	0x0000A000	0x0000B000	0x00032000
raw-address (end)	0x0000A000	0x0000B000	0x00032000	0x003B0000
raw-size (3719168 bytes)	0x00009000 (36864 bytes)	0x00001000 (4096 bytes)	0x00027000 (159744 bytes)	0x00358000 (3518
virtual-address	0x00001000	0x0000A000	0x0000B000	0x00310000
virtual-size (6718034 bytes)	0x00008BCA (35786 bytes)	0x00000998 (2456 bytes)	0x0030489C (3164316 bytes)	0x0035A454 (3515

Figure 7 TPE Studio Output showing Virtual size and raw size



Interesting API calls showing the binary has embedded resources and writes new files into the system and also encryption functionality.

imports (91)	flag (28)	first-thunk-original (INT)	first-thunk (IAT)	hint	group (10)	technique (8)
StartServiceCtrlDispatcherA	x	0x00000A6F6	0x00000A6F6	586 (0x024A)	services	-
ChangeServiceConfig2A	x	0x00000A6F0	0x00000A6F0	52 (0x0034)	services	T1569   System Services
CreateServiceA	x	0x00000A6E8	0x00000A6E8	100 (0x0064)	services	T1543   Create or Modify System Service
QueryPerformanceFrequency	x	0x00000A43A	0x00000A43A	676 (0x02A4)	reconnaissance	-
3 (closesocket)	x	0x80000003	0x80000003	0 (0x0000)	network	-
16 (recv)	x	0x80000010	0x80000010	0 (0x0000)	network	-
19 (send)	x	0x80000013	0x80000013	0 (0x0000)	network	-
8 (htonl)	x	0x80000008	0x80000008	0 (0x0000)	network	-
14 (ntohl)	x	0x8000000E	0x8000000E	0 (0x0000)	network	-
115 (WSAStartup)	x	0x80000073	0x80000073	0 (0x0000)	network	-
12 (inet_ntoa)	x	0x800000C	0x800000C	0 (0x0000)	network	-
10 (ioctlsocket)	x	0x800000A	0x800000A	0 (0x0000)	network	-
18 (select)	x	0x80000009	0x80000009	0 (0x0000)	network	-
9 (htons)	x	0x80000017	0x80000017	0 (0x0000)	network	-
23 (socket)	x	0x80000004	0x80000004	0 (0x0000)	network	-
4 (connect)	x	0x8000000B	0x8000000B	0 (0x0000)	network	-
11 (inet_addr)	x	0x80000009	0x80000009	0 (0x0000)	network	-
GetAdaptersInfo						
InternetOpenA	x	0x00000A7DC	0x00000A7DC	146 (0x0092)	network	-
InternetOpenUrlA	x	0x00000A7C8	0x00000A7C8	147 (0x0093)	network	-
InternetCloseHandle	x	0x00000A7B2	0x00000A7B2	105 (0x0069)	network	-
MoveFileExA	x	0x00000A576	0x00000A576	623 (0x026F)	file	T1105   Remote File Copy
GetCurrentThreadId	x	0x00000A524	0x00000A524	326 (0x0146)	execution	T1057   Process Discovery
GetCurrentThread	x	0x00000A53A	0x00000A53A	325 (0x0145)	execution	-
CryptGenRandom	x	0x00000A650	0x00000A650	150 (0x0096)	cryptography	T1027   Obfuscated Files or Info
CryptAcquireContextA	x	0x00000A638	0x00000A638	133 (0x0085)	cryptography	T1027   Obfuscated Files or Info
rand	x	0x00000A824	0x00000A824	678 (0x02A6)	cryptography	T1027   Obfuscated Files or Info
srand	x	0x00000A852	0x00000A852	692 (0x02B4)	cryptography	T1027   Obfuscated Files or Info
WaitForSingleObject	-	0x00000A4F6	0x00000A4F6	912 (0x0390)	synchronization	-
InterlockedIncrement	-	0x00000A50C	0x00000A50C	556 (0x022C)	synchronization	-
InterlockedDecrement	-	0x00000A4BE	0x00000A4BE	552 (0x0228)	synchronization	-
EnterCriticalSection	-	0x00000A4A6	0x00000A4A6	152 (0x0098)	synchronization	-
LeaveCriticalSection	-	0x00000A48E	0x00000A48E	593 (0x0251)	synchronization	-
InitializeCriticalSection	-	0x00000A472	0x00000A472	547 (0x0223)	synchronization	-
RegisterServiceCtrlHandlerA	-	0x00000A6D8	0x00000A6D8	524 (0x020C)	services	T1106   Execution through API
SetServiceStatus	-	0x00000A6AC	0x00000A6AC	580 (0x0244)	services	T1543   Create or Modify System Service
WaitForSingleObject	-	0x00000A4F6	0x00000A4F6	912 (0x0390)	synchronization	-
InterlockedIncrement	-	0x00000A50C	0x00000A50C	556 (0x022C)	synchronization	-
InterlockedDecrement	-	0x00000A4BE	0x00000A4BE	552 (0x0228)	synchronization	-
EnterCriticalSection	-	0x00000A4A6	0x00000A4A6	152 (0x0098)	synchronization	-
LeaveCriticalSection	-	0x00000A48E	0x00000A48E	593 (0x0251)	synchronization	-
InitializeCriticalSection	-	0x00000A472	0x00000A472	547 (0x0223)	synchronization	-
RegisterServiceCtrlHandlerA	-	0x00000A6D8	0x00000A6D8	524 (0x020C)	services	T1106   Execution through API
SetServiceStatus	-	0x00000A6AC	0x00000A6AC	580 (0x0244)	services	T1543   Create or Modify System Service
OpenSCManagerA	-	0x00000A69A	0x00000A69A	429 (0x01AD)	services	T1569   System Services
CloseServiceHandle	-	0x00000A672	0x00000A672	62 (0x003E)	services	T1569   System Services
StartServiceA	-	0x00000A662	0x00000A662	585 (0x0249)	services	T1569   System Services
OpenServiceA	-	0x00000A714	0x00000A714	431 (0x01AF)	services	T1543   Create or Modify System Service
SizeofResource	-	0x00000A584	0x00000A584	853 (0x0355)	resource	-
LoadResource	-	0x00000A5A6	0x00000A5A6	599 (0x0257)	resource	-
FindResourceA	-	0x00000A5B6	0x00000A5B6	227 (0x00E3)	resource	-
LockResource	-	0x00000A420	0x00000A420	613 (0x0265)	resource	-
QueryPerformanceCounter	-	0x00000A420	0x00000A420	675 (0x02A3)	reconnaissance	-
GetTickCount	-	0x00000A410	0x00000A410	479 (0x01DF)	reconnaissance	T1124   System Time Disc

Figure 8 PE Studio - Interesting API calls



# Basic Dynamic Analysis

{Screenshots and description about basic dynamic artifacts and methods}

## Initial Detonation with InetSim

Does nothing, the process exits as soon as it started.

A screenshot of the Process Monitor application from Sysinternals. The interface shows various monitoring tools like Task Manager, Network Monitor, and File Monitor. A table at the bottom lists two events for a process named 'Ransomware.w32' with PID 1620. The first event is 'Process Start' at 11:59:44, and the second is 'Process Exit' at 11:59:56. Both events show a 'SUCCESS' result and 'Parent PID: 3140'. The 'Detail' column indicates 'Exit Status: 0, User ...'.

Figure 9 Initial detonation with InetSim running

## Detonation without inetsim

Detonating the malware with procmon running in the background, captures many interesting details.

Firstly all the important documents/files in our system is encrypted and the desktop wallpaper is changed and decryptor window pops up on the screen. The decryptor window states that our files are encrypted and demands to pay a ransom amount to a bitcoin address in order to recover the files. It also starts a timer to make the payment.

A screenshot of a Windows desktop. The desktop background has been changed to a dark red color with a large white padlock icon. Overlaid on the desktop is a ransomware dialog box titled 'Wana Decryptor 2.0'. The dialog contains several messages in pink text:

- 'Ooops, your important files have been encrypted.'
- 'If you see this text, but don't know what it means, then your antivirus removed it from your computer.'
- 'If you need your files you can find them in the following folder.'
- 'Please find an application to restore your files from any folder or restore from a backup.'
- 'Run and follow the instructions.'

The dialog also displays a timer: 'Payment will be raised on 5/22/2024 05:56:52' and 'Time Left 02:23:58:22'. Below the timer, it says 'Your files will be lost on 5/26/2024 05:56:52' and 'Time Left 06:23:58:22'. At the bottom right of the dialog, there are buttons for 'About Bitcoin', 'How to buy bitcoins?', 'Contact Us', 'Check Payment', and 'Decrypt'. To the left of the dialog, the desktop icons are visible, including 'Recycle Bin', 'VCLib.appx', 'Tools', 'wintermin...', 'PMAT-labs...', 'install.ps1...', '@Please\_R... available.p...', 'Ransomwar...', '@WanaDec...', 'config.xml', 'desktop.ini', 'desktop.ini', 'Ransomwar...', '@WanaDec...', and 'fakenet\_logs'. The taskbar at the bottom shows the date as 'Tuesday, May 21, 2024'.

Figure 10 Desktop wallpaper change and decryptor window



Clicking on the decrypt button, decrypts some of the sample encrypted files to show that these are recoverable and asks to pay the ransom in order to recover all the files

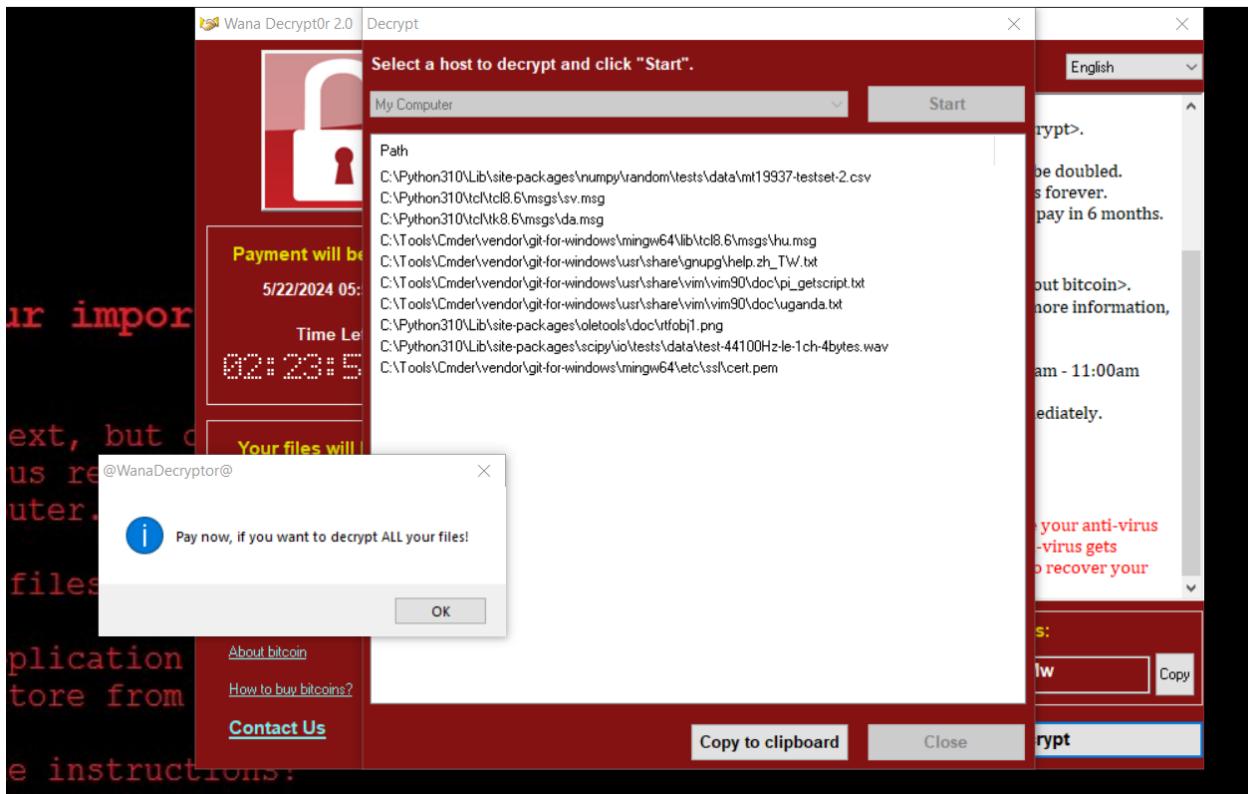


Figure 11 Decryptor program performing decryption operation on certain files in system

## Procmon & TCPView analysis

Now looking at the procmon logs – we can see the program tries attempts to connect over different IP's. Likely scanning the local network to spread through smb (port 445).

TCPView - Sysinternals: www.sysinternals.com									
Process Name	Process ID	Protocol	State	Local Address	Local Port	Remote Address	Remote Port	Create Time	Module Name
Ransomware.wannacry...	1636	TCP	Syn Sent	10.0.0.4	50393	154.137.19.35	445	5/19/2024 6:38:51 AM	mssecsvc2.0
Ransomware.wannacry...	1636	TCP	Syn Sent	10.0.0.4	50394	7.231.225.120	445	5/19/2024 6:38:51 AM	mssecsvc2.0
Ransomware.wannacry...	1636	TCP	Syn Sent	10.0.0.4	50395	81.213.11.201	445	5/19/2024 6:38:51 AM	mssecsvc2.0
Ransomware.wannacry...	1636	TCP	Syn Sent	10.0.0.4	50396	187.218.235.217	445	5/19/2024 6:38:51 AM	mssecsvc2.0
Ransomware.wannacry...	1636	TCP	Syn Sent	10.0.0.4	50397	192.169.241.220	445	5/19/2024 6:38:51 AM	mssecsvc2.0
Ransomware.wannacry...	1636	TCP	Syn Sent	10.0.0.4	50398	99.212.13.70	445	5/19/2024 6:38:51 AM	mssecsvc2.0
Ransomware.wannacry...	1636	TCP	Syn Sent	10.0.0.4	50399	213.234.141.141	445	5/19/2024 6:38:51 AM	mssecsvc2.0
Ransomware.wannacry...	1636	TCP	Syn Sent	10.0.0.4	50400	176.61.204.230	445	5/19/2024 6:38:51 AM	mssecsvc2.0
Ransomware.wannacry...	1636	TCP	Syn Sent	10.0.0.4	50401	150.231.127.134	445	5/19/2024 6:38:51 AM	mssecsvc2.0
Ransomware.wannacry...	1636	TCP	Syn Sent	10.0.0.4	50402	25.136.29.142	445	5/19/2024 6:38:51 AM	mssecsvc2.0
Ransomware.wannacry...	1636	TCP	Syn Sent	10.0.0.4	50403	215.121.168.81	445	5/19/2024 6:38:51 AM	mssecsvc2.0
Ransomware.wannacry...	1636	TCP	Syn Sent	10.0.0.4	50404	205.38.140.108	445	5/19/2024 6:38:51 AM	mssecsvc2.0
Ransomware.wannacry...	1636	TCP	Syn Sent	10.0.0.4	50405	199.228.196.131	445	5/19/2024 6:38:51 AM	mssecsvc2.0
Ransomware.wannacry...	1636	TCP	Syn Sent	10.0.0.4	50406	75.214.76.154	445	5/19/2024 6:38:51 AM	mssecsvc2.0
Ransomware.wannacry...	1636	TCP	Syn Sent	10.0.0.4	50407	223.78.245.230	445	5/19/2024 6:38:51 AM	mssecsvc2.0
Ransomware.wannacry...	1636	TCP	Syn Sent	10.0.0.4	50408	171.115.250.235	445	5/19/2024 6:38:51 AM	mssecsvc2.0
Ransomware.wannacry...	1636	TCP	Syn Sent	10.0.0.4	50409	182.97.229.64	445	5/19/2024 6:38:51 AM	mssecsvc2.0
Ransomware.wannacry...	1636	TCP	Syn Sent	10.0.0.4	50410	80.162.53.194	445	5/19/2024 6:38:51 AM	mssecsvc2.0
Ransomware.wannacry...	1636	TCP	Syn Sent	10.0.0.4	50411	82.151.230.188	445	5/19/2024 6:38:52 AM	mssecsvc2.0
Ransomware.wannacry...	1636	TCP	Syn Sent	10.0.0.4	50364	157.71.119.29	445	5/19/2024 6:38:50 AM	mssecsvc2.0
Ransomware.wannacry...	1636	TCP	Syn Sent	10.0.0.4	50359	150.150.164.236	445	5/19/2024 6:38:50 AM	mssecsvc2.0
Ransomware.wannacry...	1636	TCP	Syn Sent	10.0.0.4	50354	166.205.123.124	445	5/19/2024 6:38:50 AM	mssecsvc2.0

Figure 12 TCP view of malware scanning the network to exploit SMB vuln



5:53:56...	Ransomware.w...	2196	TCP Reconnect	DESKTOP-TGRDRIT\50643 -> 164.222.216.138.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:56...	Ransomware.w...	2196	TCP Disconnect	DESKTOP-TGRDRIT\50642 -> 200.91.223.177.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:56...	Ransomware.w...	2196	TCP Disconnect	DESKTOP-TGRDRIT\50643 -> 164.222.216.138.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:56...	Ransomware.w...	2196	TCP Disconnect	DESKTOP-TGRDRIT\50640 -> 130.160.128.217.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:56...	Ransomware.w...	2196	TCP Disconnect	DESKTOP-TGRDRIT\50638 -> 192.4.25.254.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:56...	Ransomware.w...	2196	TCP Disconnect	DESKTOP-TGRDRIT\50637 -> 72.25.119.66.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:56...	Ransomware.w...	2196	TCP Reconnect	DESKTOP-TGRDRIT\50644 -> 72.115.210.49.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:56...	Ransomware.w...	2196	TCP Reconnect	DESKTOP-TGRDRIT\50645 -> 114.54.123.83.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:56...	Ransomware.w...	2196	TCP Disconnect	DESKTOP-TGRDRIT\50646 -> 33.10.68.44.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:56...	Ransomware.w...	2196	TCP Disconnect	DESKTOP-TGRDRIT\50648 -> 87.133.240.176.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:56...	Ransomware.w...	2196	TCP Disconnect	DESKTOP-TGRDRIT\50647 -> 207.154.139.65.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:56...	Ransomware.w...	2196	TCP Disconnect	DESKTOP-TGRDRIT\50646 -> 72.25.119.66.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:56...	Ransomware.w...	2196	TCP Disconnect	DESKTOP-TGRDRIT\50645 -> 114.54.123.83.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:56...	Ransomware.w...	2196	TCP Reconnect	DESKTOP-TGRDRIT\50644 -> 72.115.210.49.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:56...	Ransomware.w...	2196	TCP Disconnect	DESKTOP-TGRDRIT\50649 -> 34.51.73.103.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:56...	Ransomware.w...	2196	TCP Reconnect	DESKTOP-TGRDRIT\50649 -> 34.51.73.103.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:56...	Ransomware.w...	2196	TCP Disconnect	DESKTOP-TGRDRIT\50650 -> 133.192.210.189.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:56...	Ransomware.w...	2196	TCP Reconnect	DESKTOP-TGRDRIT\50652 -> 132.203.126.250.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:56...	Ransomware.w...	2196	TCP Disconnect	DESKTOP-TGRDRIT\50651 -> 52.98.100.230.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:56...	Ransomware.w...	2196	TCP Disconnect	DESKTOP-TGRDRIT\50650 -> 133.192.210.189.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:56...	Ransomware.w...	2196	TCP Disconnect	DESKTOP-TGRDRIT\50651 -> 52.98.100.230.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:56...	Ransomware.w...	2196	TCP Disconnect	DESKTOP-TGRDRIT\50652 -> 132.203.126.250.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:56...	Ransomware.w...	2196	TCP Reconnect	DESKTOP-TGRDRIT\50653 -> 108.215.124.238.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:56...	Ransomware.w...	2196	TCP Reconnect	DESKTOP-TGRDRIT\50654 -> 15.14.223.175.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:56...	Ransomware.w...	2196	TCP Disconnect	DESKTOP-TGRDRIT\50653 -> 108.215.124.238.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:56...	Ransomware.w...	2196	TCP Disconnect	DESKTOP-TGRDRIT\50654 -> 15.14.223.175.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:57...	Ransomware.w...	2196	TCP Reconnect	DESKTOP-TGRDRIT\50655 -> 43.124.52.107.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:57...	Ransomware.w...	2196	TCP Disconnect	DESKTOP-TGRDRIT\50655 -> 43.124.52.107.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:57...	Ransomware.w...	2196	Thread Create		SUCCESS	Thread ID: 6492
5:53:57...	Ransomware.w...	2196	TCP Disconnect	DESKTOP-TGRDRIT\50656 -> 189.89.67.251.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:57...	Ransomware.w...	2196	TCP Reconnect	DESKTOP-TGRDRIT\50657 -> 38.128.209.14.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:57...	Ransomware.w...	2196	TCP Disconnect	DESKTOP-TGRDRIT\50657 -> 115.240.25.96.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:57...	Ransomware.w...	2196	TCP Reconnect	DESKTOP-TGRDRIT\50661 -> 214.202.191.8.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:57...	Ransomware.w...	2196	TCP Reconnect	DESKTOP-TGRDRIT\50658 -> 181.40.208.126.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:57...	Ransomware.w...	2196	TCP Reconnect	DESKTOP-TGRDRIT\50660 -> 25.94.2.148.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:57...	Ransomware.w...	2196	TCP Reconnect	DESKTOP-TGRDRIT\50662 -> 177.234.52.196.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:57...	Ransomware.w...	2196	TCP Disconnect	DESKTOP-TGRDRIT\50659 -> 115.240.25.96.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:57...	Ransomware.w...	2196	TCP Disconnect	DESKTOP-TGRDRIT\50658 -> 181.40.208.126.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:57...	Ransomware.w...	2196	TCP Disconnect	DESKTOP-TGRDRIT\50662 -> 177.234.52.196.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:57...	Ransomware.w...	2196	TCP Disconnect	DESKTOP-TGRDRIT\50661 -> 214.202.191.8.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:57...	Ransomware.w...	2196	TCP Disconnect	DESKTOP-TGRDRIT\50660 -> 25.94.2.148.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:57...	Ransomware.w...	2196	TCP Reconnect	DESKTOP-TGRDRIT\50664 -> 148.252.54.221.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:57...	Ransomware.w...	2196	TCP Reconnect	DESKTOP-TGRDRIT\50663 -> 137.130.244.216.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:57...	Ransomware.w...	2196	TCP Disconnect	DESKTOP-TGRDRIT\50664 -> 148.252.54.221.microsoft-ds	SUCCESS	Length: 0. seqnum: ...
5:53:57...	Ransomware.w...	2196	TCP Disconnect	DESKTOP-TGRDRIT\50663 -> 137.130.244.216.microsoft-ds	SUCCESS	Length: 0. seqnum: ...

Figure 13 Procmon logs showing TCP connection over local network

Filtering for process creation events, we can see the initial binary spawns tasksche.exe from windows directory

Process Monitor - Sysinternals: www.sysinternals.com						
File	Edit	Event	Filter	Tools	Options	Help
Time o...	Process Name	PID	Operation	Path	Result	Detail
5:37:21...	Ransomware.w...	1580	Process Start		SUCCESS	Parent PID: 536, Co...
5:37:34...	Ransomware.w...	1636	Process Start		SUCCESS	Parent PID: 628, Co...
5:37:34...	Ransomware.w...	1580	Process Create	C:\WINDOWS\tasksche.exe	SUCCESS	PID: 2424, Comma...
5:37:34...	Ransomware.w...	1580	Process Exit		SUCCESS	Exit Status: 0, User ...

Figure 14 Child process tasksche.exe being created

Reviewing windows directory we can see tasksche.exe has been modified.

File	Home	Share	View	Application Tools	
←	→	↑	↓	This PC	Local Disk (C) > Windows
Name	Date modified	Type	Size		
bootstat.dat	5/19/2024 5:59 AM	DAT File	66 KB		
tasksche.exe	5/19/2024 5:54 AM	Application	3,432 KB		
WindowsUpdate.log	5/19/2024 5:52 AM	Text Document	1 KB		
DtcInstall.log	2/13/2024 6:13 PM	Text Document	2 KB		
Isasetup.log	2/13/2024 6:12 PM	Text Document	2 KB		
PFRO.log	2/13/2024 7:51 AM	Text Document	4 KB		
SMSS-PFRO080e.tmp	1/14/2024 9:36 PM	TMP File	399 KB		
FindDll.exe	6/2/2023 5:28 PM	Application	50 KB		



Filtering for tasksche.exe, we can see this too creates multiple next stage files in randomly named folder programdata directory. This appears to be staging area for the program.

Time o...	Process Name	PID	Operation	Path	Result	Detail
5:53:44	tasksche.exe	2224	CreateFile	C:\ProgramData\kovbiruwfvvqwh488\00000000.dky	NAME NOT FOUND Desired Access R.	
5:53:49	tasksche.exe	2224	CreateFile	C:\ProgramData\kovbiruwfvvqwh488\askdl.exe	SUCCESS	Desired Access R..
5:53:49	tasksche.exe	2224	CreateFile	C:\ProgramData\kovbiruwfvvqwh488\askdl.exe	SUCCESS	Desired Access R..
5:53:49	tasksche.exe	2224	CreateFile	C:\ProgramData\kovbiruwfvvqwh488\askdl.exe	SUCCESS	Desired Access R..
5:53:49	tasksche.exe	2224	CreateFileMap	C:\ProgramData\kovbiruwfvvqwh488\askdl.exe	FILE LOCKED WIT...	SyncType SyncTy...
5:53:49	tasksche.exe	2224	CreateFileMap	C:\ProgramData\kovbiruwfvvqwh488\askdl.exe	SUCCESS	SyncType SyncTy...
5:53:49	tasksche.exe	2224	CreateFile	C:\Windows\apppatch\sysmain.sdb	SUCCESS	Desired Access G..
5:53:49	tasksche.exe	2224	CreateFile	C:\ProgramData\kovbiruwfvvqwh488\askdl.exe	SUCCESS	Desired Access G..
5:53:50	tasksche.exe	2224	CreateFile	C:\ProgramData\kovbiruwfvvqwh488\00000000.dky	NAME NOT FOUND Desired Access R.	
5:54:04	tasksche.exe	2224	CreateFile	C:\ProgramData\kovbiruwfvvqwh488\00000000.res	NAME NOT FOUND Desired Access R.	
5:54:05	tasksche.exe	2224	CreateFile	C:\ProgramData\kovbiruwfvvqwh488\rest	SUCCESS	Desired Access G..
5:54:09	tasksche.exe	2224	CreateFile	C:\ProgramData\kovbiruwfvvqwh488\00000000.dky	NAME NOT FOUND Desired Access R.	
5:54:14	tasksche.exe	2224	CreateFile	C:\ProgramData\kovbiruwfvvqwh488\00000000.dky	NAME NOT FOUND Desired Access R.	
5:54:19	tasksche.exe	2224	CreateFile	C:\ProgramData\kovbiruwfvvqwh488\00000000.dky	NAME NOT FOUND Desired Access R.	
5:54:20	tasksche.exe	2224	CreateFile	C:\ProgramData\kovbiruwfvvqwh488\askdl.exe	SUCCESS	Desired Access R..
5:54:20	tasksche.exe	2224	CreateFile	C:\ProgramData\kovbiruwfvvqwh488\askdl.exe	SUCCESS	Desired Access R..
5:54:20	tasksche.exe	2224	CreateFile	C:\ProgramData\kovbiruwfvvqwh488\askdl.exe	FILE LOCKED WIT...	SyncType SyncTy...
5:54:20	tasksche.exe	2224	CreateFile	C:\ProgramData\kovbiruwfvvqwh488\askdl.exe	SUCCESS	SyncType SyncTy...
5:54:20	tasksche.exe	2224	CreateFile	C:\Windows\apppatch\sysmain.sdb	SUCCESS	Desired Access G..
5:54:24	tasksche.exe	2224	CreateFile	C:\ProgramData\kovbiruwfvvqwh488\00000000.dky	NAME NOT FOUND Desired Access R.	
5:54:29	tasksche.exe	2224	CreateFile	C:\ProgramData\kovbiruwfvvqwh488\00000000.dky	NAME NOT FOUND Desired Access R.	
5:54:29	tasksche.exe	2224	CreateFile	C:\ProgramData\kovbiruwfvvqwh488\wnry	SUCCESS	Desired Access G..
5:54:29	tasksche.exe	2224	CreateFile	C:\ProgramData\kovbiruwfvvqwh488\wnry	FILE LOCKED WIT...	SyncType SyncTy...
5:54:29	tasksche.exe	2224	CreateFile	C:\ProgramData\kovbiruwfvvqwh488\wnry	SUCCESS	SyncType SyncTy...
5:54:29	tasksche.exe	2224	CreateFile	C:\ProgramData\kovbiruwfvvqwh488\@Please_Read_Me@.txt	NAME NOT FOUND Desired Access R.	
5:54:29	tasksche.exe	2224	CreateFile	C:\Windows\SysWOW64\windows storage.dll	SUCCESS	Desired Access R..
5:54:29	tasksche.exe	2224	CreateFile	C:\Windows\SysWOW64\windows storage.dll	SUCCESS	Desired Access R..
5:54:29	tasksche.exe	2224	CreateFileMap	C:\Windows\SysWOW64\windows storage.dll	FILE LOCKED WIT...	SyncType SyncTy...
5:54:29	tasksche.exe	2224	CreateFileMap	C:\Windows\SysWOW64\windows storage.dll	SUCCESS	SyncType SyncTy...
5:54:29	tasksche.exe	2224	CreateFile	C:\Windows\Globalization\Sorting\SortDefault.nls	NAME NOT FOUND Desired Access R.	
5:54:29	tasksche.exe	2224	CreateFileMap	C:\Windows\Globalization\Sorting\SortDefault.nls	FILE LOCKED WIT...	SyncType SyncTy...
5:54:29	tasksche.exe	2224	CreateFileMap	C:\Windows\Globalization\Sorting\SortDefault.nls	SUCCESS	SyncType SyncTy...
5:54:29	tasksche.exe	2224	CreateFile	C:\Windows\SysWOW64\Config\systemprofile\Desktop	NAME NOT FOUND Desired Access R.	
5:54:29	tasksche.exe	2224	CreateFile	C:\Windows\SysWOW64\Config\systemprofile\Documents	NAME NOT FOUND Desired Access R.	
5:54:29	tasksche.exe	2224	CreateFile	C:\Users\Public\Desktop	SUCCESS	Desired Access R..
5:54:29	tasksche.exe	2224	CreateFile	C:\Users	SUCCESS	Desired Access R..
5:54:29	tasksche.exe	2224	CreateFile	C:\ProgramData\kovbiruwfvvqwh488\sspiCli.dll	NAME NOT FOUND Desired Access R.	
5:54:29	tasksche.exe	2224	CreateFile	C:\Windows\SysWOW64\sspiCli.dll	SUCCESS	Desired Access R..
5:54:29	tasksche.exe	2224	CreateFile	C:\Windows\SysWOW64\sspiCli.dll	SUCCESS	Desired Access R..

Figure 15 Procmon logs showing creation of next stage files

#### Programdata directory with the next stage files

This PC > Local Disk (C:) > ProgramData > kovbiruwfvvqwh488						
	Name	Date modified	Type	Size		
cess	msg	5/19/2024 5:55 AM	File folder			
o	TaskData	5/19/2024 5:55 AM	File folder			
ents	@Please_Read_Me@.txt	5/19/2024 5:54 AM	Text Document	1 KB		
ects	@WanaDecryptor@.exe	5/12/2017 2:22 AM	Application	240 KB		
ads	00000000.eky	5/19/2024 5:54 AM	EKY File	2 KB		
isk (C:)	00000000.pky	5/19/2024 5:52 AM	PKY File	1 KB		
e (D): Virtua	00000000.res	5/19/2024 6:01 AM	RES File	1 KB		
	b.wnry	5/11/2017 8:13 PM	WNRY File	1,407 KB		
	c.wnry	5/19/2024 5:56 AM	WNRY File	1 KB		
	f.wnry	5/19/2024 5:55 AM	WNRY File	1 KB		
	r.wnry	5/11/2017 3:59 PM	WNRY File	1 KB		
	s.wnry	5/9/2017 4:58 PM	WNRY File	2,968 KB		
	t.wnry	5/12/2017 2:22 AM	WNRY File	65 KB		
	taskdl.exe	5/12/2017 2:22 AM	Application	20 KB		
	tasksche.exe	5/19/2024 5:54 AM	Application	3,432 KB		
	taskse.exe	5/12/2017 2:22 AM	Application	20 KB		
	u.wnry	5/12/2017 2:22 AM	WNRY File	240 KB		



Taksche.exe also executes multiple files that were created in previous step

Time o...	Process Name	PID	Operation	Path
5:53:49....	tasksche.exe	2224	Process Create	C:\ProgramData\kovbiruwfvvqwh488\taskdl.exe
5:54:20....	tasksche.exe	2224	Process Create	C:\ProgramData\kovbiruwfvvqwh488\taskdl.exe
5:54:50....	tasksche.exe	2224	Process Create	C:\ProgramData\kovbiruwfvvqwh488\taskdl.exe
5:55:20....	tasksche.exe	2224	Process Create	C:\ProgramData\kovbiruwfvvqwh488\taskdl.exe
5:55:39....	tasksche.exe	2224	Process Create	C:\ProgramData\kovbiruwfvvqwh488@\WanaDecryptor@.exe
5:55:39....	tasksche.exe	2224	Process Create	C:\Windows\SysWOW64\cmd.exe
5:55:49....	tasksche.exe	2224	Process Create	C:\ProgramData\kovbiruwfvvqwh488\taskse.exe
5:55:49....	tasksche.exe	2224	Process Create	C:\Windows\SysWOW64\cmd.exe
5:55:51....	tasksche.exe	2224	Process Create	C:\ProgramData\kovbiruwfvvqwh488\taskdl.exe
5:56:19....	tasksche.exe	2224	Process Create	C:\ProgramData\kovbiruwfvvqwh488\taskse.exe
5:56:21....	tasksche.exe	2224	Process Create	C:\ProgramData\kovbiruwfvvqwh488\taskdl.exe
5:56:49....	tasksche.exe	2224	Process Create	C:\ProgramData\kovbiruwfvvqwh488\taskse.exe
5:56:52....	tasksche.exe	2224	Process Create	C:\ProgramData\kovbiruwfvvqwh488\taskdl.exe
5:57:19....	tasksche.exe	2224	Process Create	C:\ProgramData\kovbiruwfvvqwh488\taskse.exe
5:57:22....	tasksche.exe	2224	Process Create	C:\ProgramData\kovbiruwfvvqwh488\taskdl.exe
5:57:50....	tasksche.exe	2224	Process Create	C:\ProgramData\kovbiruwfvvqwh488\taskse.exe
5:57:52....	tasksche.exe	2224	Process Create	C:\ProgramData\kovbiruwfvvqwh488\taskdl.exe
5:58:20....	tasksche.exe	2224	Process Create	C:\ProgramData\kovbiruwfvvqwh488\taskse.exe
5:58:23....	tasksche.exe	2224	Process Create	C:\ProgramData\kovbiruwfvvqwh488\taskdl.exe
5:58:50....	tasksche.exe	2224	Process Create	C:\ProgramData\kovbiruwfvvqwh488\taskse.exe
5:58:53....	tasksche.exe	2224	Process Create	C:\ProgramData\kovbiruwfvvqwh488\taskdl.exe
5:59:20....	tasksche.exe	2224	Process Create	C:\ProgramData\kovbiruwfvvqwh488\taskse.exe
5:59:23....	tasksche.exe	2224	Process Create	C:\ProgramData\kovbiruwfvvqwh488\taskdl.exe
5:59:50....	tasksche.exe	2224	Process Create	C:\ProgramData\kovbiruwfvvqwh488\taskse.exe
5:59:53....	tasksche.exe	2224	Process Create	C:\ProgramData\kovbiruwfvvqwh488\taskdl.exe
6:00:20....	tasksche.exe	2224	Process Create	C:\ProgramData\kovbiruwfvvqwh488\taskse.exe
6:00:23....	tasksche.exe	2224	Process Create	C:\ProgramData\kovbiruwfvvqwh488\taskdl.exe
6:00:50....	tasksche.exe	2224	Process Create	C:\ProgramData\kovbiruwfvvqwh488\taskse.exe
6:00:53....	tasksche.exe	2224	Process Create	C:\ProgramData\kovbiruwfvvqwh488\taskdl.exe
6:01:20....	tasksche.exe	2224	Process Create	C:\ProgramData\kovbiruwfvvqwh488\taskse.exe
6:01:23....	tasksche.exe	2224	Process Create	C:\ProgramData\kovbiruwfvvqwh488\taskdl.exe
6:01:50....	tasksche.exe	2224	Process Create	C:\ProgramData\kovbiruwfvvqwh488\taskse.exe
6:01:53....	tasksche.exe	2224	Process Create	C:\ProgramData\kovbiruwfvvqwh488\taskdl.exe



# Advanced Static Analysis

{Screenshots and description about findings during advanced static analysis}

## Initial Kill Switch

Attempts to reach out to “www.iuqerfsodp9ifjaposdfjhgosurijfaewrwegwea[.]com”.

If connection is successful :

- ⇒ Do nothing and exit the program

If connection fails :

- ⇒ Enter the real main/ execute the next stage.

The screenshot shows assembly code from address 0x00408140 to 0x004081a5. A red box highlights the connection attempt at 0x0040814a. A callout notes "KillSwitch/Non-existent domain". Another red box highlights the connection attempt at 0x00408193, with a callout noting "Makes a request to the above domain, if no response received then proceeds to the next part of the program. If connection is successful then exits". A third red box highlights the "REAL\_ENTRY" call at 0x004081a1, with a callout "Execute the main functionality of program". A fourth red box highlights the exit path at 0x004081c8, with a callout "Exit".

```
[0x00408140]
int main(int argc, char **argv, char **envp);
; var int32_t var_64h @ stack - 0x64
; var int32_t var_50h @ stack - 0x50
; var int32_t var_17h @ stack - 0x17
; var int32_t var_13h @ stack - 0x13
; var int32_t var_fh @ stack - 0xf
; var int32_t var_bh @ stack - 0xb
; var int32_t var_7h @ stack - 0x7
; var int32_t var_3h @ stack - 0x3
; var int32_t var_1h @ stack - 0x1
0x00408140 sub esp, 0x50
0x00408143 push esi
0x00408144 push edi
0x00408145 mov ecx, 0xe ; 14
0x0040814a mov esi, str.http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwegwea.com ; 0x4313d0
0x0040814f lea edi, [var_64h]
0x00408153 xor eax, eax
0x00408155 rep movsd dword es:[edi], dword ptr [esi]
0x00408157 movsb byte es:[edi], byte ptr [esi]
0x00408158 mov dword [var_17h], eax
0x0040815c mov dword [var_13h], eax
0x00408160 mov dword [var_fh], eax
0x00408164 mov dword [var_bh], eax
0x00408168 mov dword [var_7h], eax
0x0040816c mov word [var_3h], ax
0x00408171 push eax
0x00408172 push eax
0x00408173 push eax
0x00408174 push eax
0x00408176 push eax
0x00408177 mov byte [var_1h], al
0x0040817b call dword [InternetOpenA] ; 0x40a134
0x00408181 push 0
0x00408183 push 0x84000000
0x00408188 push 0
0x0040818a lea ecx, [var_64h]
0x0040818e mov esi, eax
0x00408190 push 0
0x00408192 push ecx
0x00408193 push esi
0x00408194 call dword [InternetOpenUrlA] ; 0x40a138
0x0040819a mov edi, eax
0x0040819c push esi
0x0040819d mov esi, dword [InternetCloseHandle] ; 0x40a13c
0x004081a3 test edi, edi
0x004081a5 jne 0x4081bc ; KILL SWITCH ---> if the connection is successful, exit out o...
```

[0x004081a7]
0x004081a7 call esi
0x004081a9 push 0
0x004081ab call ss:REAL\_ENTRY ; REAL\_ENTRY
0x004081ad pop edi
0x004081ae xor eax, eax
0x004081bf pop esi
0x004081c0 add esp, 0x50
0x004081c6 ret 0x10

[0x004081bc]
0x004081bc call esi
0x004081be push edi
0x004081bf call esi
0x004081c1 pop edi
0x004081c2 xor eax, eax
0x004081c4 pop esi
0x004081c5 add esp, 0x50
0x004081c8 ret 0x10



## Real Entry/ Main execution call

Initially checks if there are any arguments :

If not then :

- ⇒ Creates service and drops the next stage

If yes then :

- ⇒ Makes some changes to the config of existing service mssecsv2.0 and then starts it

```
void real_entry();

[0x00408090]
real_entry();
; var const char *var_3ch @ stack - 0x3c
; var const char *var_38h @ stack - 0x38
; var int32_t var_34h @ stack - 0x34
; var int32_t var_30h @ stack - 0x30
; var int32_t var_2ch @ stack - 0x2c
; var const char *lpServiceStartTable @ stack - 0x28
; var int32_t var_24h @ stack - 0x24
; var int32_t var_20h @ stack - 0x20
; var int32_t var_1ch @ stack - 0x1c
0x00408090    sub    esp, 0x10
0x00408093    push   0x104      ; 260 ; DWORD nSize
0x00408098    push   data.0070f760 ; 0x70f760 ; LPSTR lpFilename
0x0040809d    push   0          ; HMODULE hModule
call    dword [GetModuleFileNameA] ; 0x40a06c ; DWORD GetModuleFileNameA(HMODULE ...
0x004080ab    cmp    dword [eax], 2
0x004080ae    jge    0x4080b9

Checks if any arguments were passed.
```

If no arg is passed then creates service and drops next stage.

```
[0x004080b0]
0x004080b0    call   create_service_drop_next_stage ; create_service_drop_next_stage
0x004080b5    add    esp, 0x10
0x004080b8    ret

[0x004080b9]
0x004080b9    push   edi
0x004080ba    push   0xf003f   ; '?' : DWORD dwDesiredAccess
0x004080bf    push   0          ; LPCSTR lpDatabaseName
0x004080c1    push   0          ; LPCSTR lpMachineName
0x004080c3    call   dword [OpenSCManagerA] ; 0x40a010 ; SC_HANDLE OpenSCManagerA(LPCSTR lpMac...
0x004080c9    mov    edi, eax
0x004080cd    test   edi, edi
0x004080cd    je     0x408101

If yes, just starts this service mssecsv2.0 which launches the wanacry binary.
```

```
[0x004080cf]
0x004080cf    push   ebx
0x004080d0    push   esi
0x004080d1    push   0xa1ff    ; DWORD dwDesiredAccess
0x004080d6    push   str.mssecsvc2.0 ; 0x4312fc ; LPCSTR lpServiceName
0x004080db    push   edi, eax
0x004080dc    call   dword [OpenServiceA] ; 0x40a028 ; SC_HANDLE OpenServiceA(SC_HANDLE hSCMan...
0x004080e2    mov    eax, dword [CloseServiceHandle] ; 0x40a018
0x004080e8    mov    esi, eax
0x004080ea    test   esi, esi
0x004080ec    je     0x4080fc
```

Dashboard | Strings | Imports | Search | Callgraph | Disassembly | Graph(real\_entry) | Hexdump | Decompiler (entry0)

## Create\_service\_drop\_next\_stage\_function call

As the name suggests, makes a function call to create a service for persistence and then writes the next stage payload to tasksche.exe

```
[0x00407f20]
create_service_drop_next_stage();
0x00407f20    call   create_service ; create_service
0x00407f25    call   write_tasksche.exe ; write_tasksche.exe
0x00407f2a    xor    eax, eax
0x00407f2c    ret
```



## Creation of service

Name of the service – msseccsvc2 → based on [tcpview](#) logs seen in dynamic analysis this is responsible for scanning/spreading over local network by exploiting smb vulnerability

```
void create_service();
```

The assembly code is as follows:

```

[0x00407c40]
create_service();
; var LPCSTR lpBinaryPathName @ stack - 0x110
0x00407c40    sub    esp, 0x104
0x00407c46    lea    eax, [esp]
0x00407c4a    push   edi
0x00407c4b    push   data.0070f760 ; 0x70f760
0x00407c50    push   str.s_m_security ; 0x431330 ; const char *format
0x00407c55    push   eax ; char *s
0x00407c56    call   dword [sprintf] ; 0x40a10c ; int sprintf(char *s, const char *format, va...
0x00407c5c    add    esp, 0xc
0x00407c5f    push   0x0000 ; 0x0000 dwDesiredAccess
0x00407c64    push   0 ; LPCSTR lpDatabaseName
0x00407c66    push   0 ; LPCSTR lpMachineName
0x00407c68    call   dword [OpenSCManagerA] ; 0x40a010 ; SC_HANDLE OpenSCManagerA(LPCSTR lpMac...
0x00407c6e    mov    edi, eax
0x00407c70    test   edi, edi
0x00407c72    je    0x407cca

[0x00407c74]
0x00407c74    push   ebx
0x00407c75    push   esi
0x00407c76    push   0 ; LPCSTR lpPassword
0x00407c78    push   0 ; LPCSTR lpServiceName
0x00407c7a    push   0 ; LPCSTR lpDependencies
0x00407c7c    push   0 ; LPVOID lpWaitHint
0x00407c7e    lea    ecx, [lpBinaryPathName]
0x00407c82    push   0 ; LPCSTR lpLoadOrderGroup
0x00407c84    push   ecx ; LPCSTR lpBinaryPathName
0x00407c85    push   1 ; 1 ; DWORD dwErrorControl
0x00407c87    push   2 ; 2 ; DWORD dwStartType
0x00407c89    push   0x10 ; 16 ; DWORD dwServiceType
0x00407c8b    push   0xf0ff ; DWORD dwDesiredAccess
0x00407c90    push   str.Microsoft_Security_Center_2_0_Service ; 0x431308 ; LPCSTR lpDisplay...
0x00407c95    push   str.msseccsvc2.0 ; 0x4312fc ; LPCSTR lpServiceName
0x00407c9a    push   edi ; SC_HANDLE hSCManager
0x00407c9b    call   dword [CreateServiceA] ; 0x40a014 ; SC_HANDLE CreateServiceA(SC_HANDLE hS...
0x00407ca1    mov    ebx, dword [!CloseServiceHandle] ; 0x40a018
0x00407ca7    mov    esi, eax
0x00407ca9    test   esi, esi
0x00407cab    je    0x407ccb

[0x00407cad]
0x00407cad    push   0 ; LPCSTR *lpServiceArgVectors
0x00407caf    push   0 ; DWORD dwNumServiceArgs

[0x00407cca]
0x00407cca    xor    eax, eax
0x00407ccc    pop    edi
0x00407ccd    add    esp, 0x104
0x00407cd3    ret

[0x00407cad]
0x00407cad    push   0 ; LPCSTR *lpServiceArgVectors
0x00407caf    push   0 ; DWORD dwNumServiceArgs

```

Annotations in the code:

- Opens a handle to Service control manager**: Points to the call to OpenSCManagerA.
- If previous call is successful, then creates a service by the name msseccsvc2.0 that launches initial wannacry binary with "/m security" args.**: Points to the CreateServiceA call.

Then just starts this newly created service.

The assembly code is as follows:

```

[0x00407cad]
0x00407cad    push   0 ; LPCSTR *lpServiceArgVectors
0x00407caf    push   0 ; DWORD dwNumServiceArgs
0x00407cb1    push   esi ; SC_HANDLE hService
0x00407cb2    call   dword [StartServiceA] ; 0x40a01c ; BOOL StartServiceA(SC_HANDLE hService, ...
0x00407cb5    push   esi
0x00407cb9    call   ebx

[0x00407ccb]
0x00407ccb    push   edi
0x00407cbc    call   ebx
0x00407cbe    pop    esi
0x00407cbf    pop    ebx
0x00407cc0    xor    eax, eax
0x00407cc2    pop    edi
0x00407cc3    add    esp, 0x104
0x00407cc9    ret

```

Annotation in the code:

- Starts the service**: Points to the call to StartServiceA.



## Creation of next stage payload

Finds and loads embedded 1831 resource.

The screenshot shows four consecutive memory dump sections (0x00407d69 to 0x00407da7) with assembly code. Red boxes highlight specific instructions in each section:

- [0x00407d69]: push data\_0042127c ; 0x42127c ; LPCSTR lpType  
push 0x727 ; 1831 ; LPCSTR lpName  
push ebx ; HMODULE hModule  
call dword [FindResourceA] ; 0x40a05c ; HRSRC FindResourceA(HMODULE hModule, L...
- [0x00407d84]: push esi ; HRSRC hResInfo  
push ebx ; HMODULE hModule  
call dword [LoadResource] ; 0x40a058 ; HGLOBAL LoadResource(HMODULE hModule, H...
- [0x00407d94]: push eax ; HGLOBAL hResData  
call dword [LockResource] ; 0x40a0a0 ; LPVOID LockResource(HGLOBAL hResData)
- [0x00407da7]: push esi ; HRSRC hResInfo  
push ebx ; HMODULE hModule  
call dword [SizeofResource] ; 0x40a050 ; DWORD SizeofResource(HMODULE hModule,...

A red callout box points to the first section with the text "Finds and loads 1831 resource into the memory".

Next part is not clear in disassembled output, but we can see the translated function calls in debugger. Basically it creates tasksche.exe in C:/Windows directory and writes the loaded 1831 resource into it.

The screenshot shows assembly code with several red boxes highlighting specific instructions:

- 0x00407db9: mov ecx, 0x40 ; 'e' ; 64
- 0x00407dbe: xor eax, eax
- 0x00407dc0: lea edi, [lpExistingFileName + 0x1]
- 0x00407dc4: mov byte [lpExistingFileName], bl
- 0x00407dc8: rep stosd dword es:[edi], eax
- 0x00407dca: stosw word es:[edi], ax
- 0x00407dcc: stosb byte es:[edi], al
- 0x00407ddc: mov ecx, 0x40 ; 'e' ; 64
- 0x00407dd2: xor eax, eax
- 0x00407dd4: lea edi, [lpNewFileName + 0x1]
- 0x00407ddb: mov byte [lpNewFileName], bl
- 0x00407de2: rep stosd dword es:[edi], eax
- 0x00407de4: mov esi, dword [sprintf] ; 0x40a10c
- 0x00407dea: push str.tasksche.exe ; 0x43136c
- 0x00407def: stosw word es:[edi], ax
- 0x00407df1: stosb byte es:[edi], al
- 0x00407df2: push str.WINDOWS ; 0x431364
- 0x00407df7: lea eax, [lpExistingFileName]
- 0x00407dfb: push str.C:\_s\_s ; 0x431358
- 0x00407e00: push eax
- 0x00407e01: call esi
- 0x00407e03: add esp, 0x10
- 0x00407e06: lea ecx, [lpNewFileName]
- 0x00407e0d: push str.WINDOWS ; 0x431364
- 0x00407e12: push str.C:\_s\_qeriuwjhrf ; 0x431344
- 0x00407e17: push ecx
- 0x00407e18: call esi
- 0x00407e1a: add esp, 0xc
- 0x00407e1d: lea edx, [lpNewFileName]
- 0x00407e24: lea eax, [lpExistingFileName]
- 0x00407e28: push 1 ; 1 ; DWORD dwFlags
- 0x00407e2a: push edx ; LPCSTR lpNewFileName
- 0x00407e2b: push eax ; LPCSTR lpExistingFileName
- 0x00407e2c: call dword [MoveFileExA] ; 0x40a04c ; BOOL MoveFileExA(LPCSTR lpExistingFileName,...
- 0x00407e32: push ebx
- 0x00407e33: push 4 ; 4
- 0x00407e35: push 2 ; 2
- 0x00407e37: push ebx
- 0x00407e38: push ebx
- 0x00407e39: lea ecx, [var\_258h]
- 0x00407e3d: push 0x00000000
- 0x00407e42: push ecx
- 0x00407e43: call dword [data.00431458] ; 0x431458
- 0x00407e49: mov esi, eax
- 0x00407e4b: cmp esi, 0xffffffff
- 0x00407e4e: je 0x407f08

Red callouts point to the file move section (0x00407e2b) with the text "Rename of file, existing to new" and to the file creation section (0x00407e43) with the text "Unknown call --- likely file create??".



# Advanced Dynamic Analysis

{Screenshots and description about advanced dynamic artifacts and methods}

## Debugging Wannacry.exe – Initial payload

## Kill Switch functionality

Domain is loaded into esi

Makes a request to this domain and checks the response.  
EAX=1, if successful connection...

Domain is loaded into esi

Zero flag is set since we did not receive a response

## Service Creation:

```
00407C68 FF15 100A0000 call dword ptr ds:[<openSCManagerA>]
00407C70 8BFF mov edi,edi
00407C74 74 0E je ransomware.wannacry.407CCA
00407C75 53 push ebx
00407C76 56 push esi
00407C77 6A 00 push 0
00407C78 6A 00 push 0
00407C7A 6A 00 push 0
00407C7C 6A 00 push 0
00407C7E 804C24 1C lea ecx,dword ptr ss:[esp+1C]
00407C84 51 push ecx
00407C85 6A 02 push 2
00407C86 6A 02 push 2
00407C89 6A 10 push 10
00407C8A 68 FF010F00 push F0FF
00407C90 68 08134300 push ransomware.wannacry.431308
00407C91 FC124300 push ransomware.wannacry.4312FC
00407C9A 57 push edi
00407C9B FF15 100A0000 call dword ptr ds:[<createserviceA>]
00407C9C 8BFF mov edi,edi
00407C9D 6A 00 push 0
00407C9E 6A 00 push 0
00407CAF 56 push ebx
00407CB0 56 push esi
00407CB1 56 push ebx
00407CB2 FF15 1CA04000 call dword ptr ds:[<startserviceA>]
00407CB3 56 push ebx
00407CB4 56 push esi
00407CB5 FD31 call ebx
00407CB9 57 push edi
00407CBF 57 push edi

00407D40 00407D40 <win32k!win32k!CloseHandle>
00407D41 ESI 747300CAE0 <wininet!InternetCloseHandle>
00407D42 EDI 0069CAE8
00407D43 EIP 00407C9A ransomware.wannacry.00407C9A

EFLAGS 00000000206
ZF 0 PF 0 AF 0
OF 0 SF 0 DF 0
CF 0 TF 0 IF 1

LastError 00000000 (ERROR_SUCCESS)
LastStatus 00000000 (STATUS_NO_TOKEN)

GS 0028 FS 0053
ES 0028 DS 0028
CS 0028 SS 0028

ST(0) 00000000000000000000000000000000 x87r0 Empty 0.00000000000000000000000000000000
ST(1) 00000000000000000000000000000000 x87r1 Empty 0.00000000000000000000000000000000
ST(2) 00000000000000000000000000000000 x87r2 Empty 0.00000000000000000000000000000000
ST(3) 00000000000000000000000000000000 x87r3 Empty 0.00000000000000000000000000000000
ST(4) 00000000000000000000000000000000 x87r4 Empty 0.00000000000000000000000000000000
ST(5) 00000000000000000000000000000000 x87r5 Empty 0.00000000000000000000000000000000

Default (stdcall) ▾ 5 □ Unlocked
1: [esp] 00431308 ransomware.wannacry.00431308 "Micros
2: [esp+4] 00000000000000000000000000000000
3: [esp+8] 00000000000000000000000000000000
4: [esp+12] 00000000000000000000000000000000
5: [esp+14] 00000000000000000000000000000000
```

The screenshot shows the Immunity Debugger interface with the assembly dump tab selected. The assembly dump window displays memory starting at address 0019FD24, which contains the instruction `CALL [rax]`. The registers window shows the CPU register state. A red box highlights the stack setup code at the beginning of the dump.

Address	Instruction	Description
0019FD24	CALL [rax]	ransomware.wannacry."mssevcsvc2.0"
0019FD28	CALL [rax]	ransomware.wannacry."Microsoft Security Center (2.0) Service"
0019FD2C	MOV rax, 0	Stack setup before the service call
0019FD30	MOV rax, 0	"C:\Users\blue\Desktop\Ransomware.wannacry.exe -m security"
0019FD34	MOV rax, 0	00000000
0019FD38	MOV rax, 0	00000000
0019FD3C	MOV rax, 0	00000000
0019FD40	MOV rax, 0	00000000
0019FD44	MOV rax, 0	00000000
0019FD48	MOV rax, 0	00000000
0019FD4C	MOV rax, 0	00000000
0019FD50	MOV rax, 0	00000000
0019FD54	74730EFD	wininet.internetCloseHandle
0019FD58	00000000	
0019FD5C	00000000	"l'aigh"
0019FD60	553C3443	
0019FD64	73726573	
0019FD68	756C625C	windows.storage.ShellExecuteExW+86EC



## Evidence of service being created

Microsoft Security Center (2.0) Service Properties (Local Computer)

Name	Description	Status	Startup Type	Log On As
Microsoft Account Sign-in Assistant	Enables user...	Manual (Trigg...	Local System	
Microsoft App-V Client	Manages Ap...	Disabled	Local System	
Microsoft Cloud Identity Service				
Microsoft Defender Antivirus Network I...				
Microsoft Defender Antivirus Service				
Microsoft Edge Elevation Service (Micro...				
Microsoft Edge Update Service (edgeup...				
Microsoft iSCSI Initiator Service				
Microsoft Keyboard Filter				
Microsoft Passport				
Microsoft Passport Container				
Microsoft Security Center (2.0) Service				
Microsoft Software Shadow Copy Provider				
Microsoft Storage Spaces SMP				
Microsoft Store Install Service				
Microsoft Windows SMS Router Service				
Natural Authentication				
NetTcp Port Sharing Service				
Netlogon				
Network Connected Devices Auto-Setup				
Network Connection Broker				
Network Connections				
Network Connectivity Assistant				
Network List Service				
Network Location Awareness				
Network Setup Service				
Network Store Interface Service				
Offline Files				
OpenSSH Authentication Agent				
Optimize drives				

## Write/Rename Next stage payload

Checks if tasksche.exe exists, if yes then renames the file into a weird random name: qeruwjhrf.

```

00407DE4 8835 0CA14000 mov esi,dword ptr ds:[<sprintf>]
00407DEA 68 6C134300 push ransomware.wannacry,43136C
00407DEF 66:A8 stosw
00407DF1 AA stosb
00407DF2 68 64134300 push ransomware.wannacry,431364
00407DF7 804424 70 lea eax,dword ptr ss:[esp+70]
00407DFB 68 58134300 push ransomware.wannacry,431358
00407E00 50 push eax
00407E01 FF06 call esi
00407E05 83C4 10 add esp,10
00407E06 808C24 6C010000 lea ecx,dword ptr ss:[esp+10C]
00407E0D 68 44134300 push ransomware.wannacry,431344
00407E12 68 44134300 push ransomware.wannacry,431344
00407E17 51 push ecx
00407E18 FF06 call esi
00407E1A 83C4 0L add esp,L
00407E1D 809424 6C010000 lea edx,dword ptr ss:[esp+10C]
00407E24 804424 68 lea eax,dword ptr ss:[esp+68]
00407E28 6A 01 push 1
00407E2A S2 push edx
00407E2B 50 push eax
00407E2C FF15 4CA04000 call dword ptr ds:[_MoveFileExA]
00407E2E 52 push edx
00407E31 6A 04 push 4
00407E33 6A 04 push 4

Prints C:\Windows\tasksche.exe
Prints C:\Windows\qeruwjhrf
copies/renames tasksche.exe in windows
directory to random name "qeruwjhrf" if it exists

```

Event Properties

Date:	5/7/2024 6:41:15.9485063 AM
Thread:	3264
Class:	File System
Operation:	SetRenameInformationFile
Result:	SUCCESS
Path:	C:\Windows\tasksche.exe
Duration:	0.0000645
ReplaceIfExists:	True
FileName:	C:\Windows\qeruwjhrf

File System View

Name	Date modified	Type	Size
bootstat.dat	5/7/2024 5:41 AM	DAT File	66 KB
WindowsUpdate.log	5/7/2024 5:39 AM	Text Document	1 KB
<b>qeruwjhrf</b>	4/26/2024 8:42 AM	System file	3,432 KB
DtcInstall.log	2/13/2024 6:13 PM	Text Document	2 KB
Isasetup.log	2/13/2024 6:12 PM	Text Document	2 KB
PFR0.log	2/13/2024 7:51 AM	Text Document	4 KB
SMSS-PFR0080e.tmp	1/14/2024 9:36 PM	TMP File	399 KB
FindDll.exe	6/2/2023 5:28 PM	Application	50 KB
qdiprocs.exe	6/2/2023 5:28 PM	Application	173 KB



Else, creates tasksche.exe in windows directory and writes the 1831 resource.

The screenshot shows the Immunity Debugger interface with assembly code and memory dump tabs at the bottom. The assembly pane displays several calls to `call dword ptr ds:[<...>]`, which are highlighted with red boxes. One such call at address 0040E424 is annotated with a red box and the text "Creates tasksche.exe and writes the loaded 1831 resource". The assembly code involves pushing arguments onto the stack and calling these functions. The memory dump tabs at the bottom show the raw hex and ASCII data for the memory dump.

Executes the newly created file tasksche.exe with /i argument

The screenshot illustrates the process of executing a newly created binary. The assembly code shows the preparation of arguments for a function call, specifically `createProcessA`. The memory dump pane shows the command `C:\Windows\tasksche.exe /i` being loaded into memory. A red box highlights the `call` instruction, which corresponds to the creation of the new process. Another red box highlights the memory dump entry, and a third red box contains the text "Executes the newly created binary with /i argument".

Once this is executed, all important files in our computer gets encrypted and a decryptor program is launched stating the instruction on how to recover the files.

We can place a breakpoint at the create process call, and stop the execution of the binary at this point and then look at the second stage – tasksche.exe and dissect it further.



## Debugging tasksche.exe – next stage payload

### Stage Folder Creation

Initially calls a function which uses the computer name to create a unique randomized string. This string is later used to name the staging folder and service created by the program.

The screenshot shows the assembly code for tasksche.exe. Several red boxes highlight specific parts of the code:

- Call to get\_random\_string\_based\_on\_computer\_name:** A call instruction to address 00401B8C is highlighted, with a note: "This call returns a unique random string based on computer name".
- Check for arguments:** A cmp instruction comparing the argument with '\$' is highlighted, with a note: "Checks if any arguments is passed, if yes, compares the argument with '\$'. If yes then creates a staging directory named with unique string from previous call and sets it as current working directory".
- Get computer name:** A mov instruction moving the computer name to a register is highlighted, with a note: "Gets computer name of the victim and then does manipulation on this to generate a random string".
- Call to create\_directory:** A call instruction to address 00401B1E is highlighted, with a note: "Creates randomly named directory in programdata and sets it as current working directory".
- Memory dump:** A dump window shows the ASCII representation of the generated random string, starting with 'L"\kovbiruufvvqhd488"'.
- Registers and Stack:** The Registers and Stack panes show the current state of the CPU registers and stack.

Creation of staged folder using the randomly generated string in previous function:

The screenshot shows the assembly code for tasksche.exe. Several red boxes highlight specific parts of the code:

- Call to create\_directory:** A call instruction to address 00401B1E is highlighted, with a note: "Creates randomly named directory in programdata and sets it as current working directory".
- Call to setcurrentdirectory:** A call instruction to address 00401B20 is highlighted, with a note: "edi:setCurrentDirectoryW".
- Call to create\_directory again:** Another call instruction to address 00401B1E is highlighted, with a note: "Creates randomly named directory in programdata and sets it as current working directory".
- Memory dump:** A dump window shows the ASCII representation of the generated random string, starting with 'L"\kovbiruufvvqhd488"'.
- Registers and Stack:** The Registers and Stack panes show the current state of the CPU registers and stack.



## Persistence – Creation of service

### Creation of tasksche.exe service for persistence

The screenshot shows assembly code from address 00401014 to 0040109E. A red box highlights the instruction `call dword ptr ds:[<closeServiceHandle>]`. A callout box points to this instruction with the text "Creates a service with generated random name that runs tasksche exe via cmd". Another callout box points to the same instruction with the text "[ebp+8]:\"C:\ProgramData\kovbiruwfvvqwh488\tasksche.exe\"". The assembly code includes various pushes, moves, and comparisons related to service creation.

```

00401014 BB FF010F00 mov ebx,F01FF
00401015 B0 ACF84000 mov eax,tasksc...
00401016 53 push ebx
00401017 56 push esi
00401018 50 push eax
00401021 FF15 04804000 call dword ptr ds:[<op...
00401027 3B87 mov eax,edi
00401029 8945 F4 mov dword ptr ss:[ebp-C],eax
0040102A 2A 17 jae tasksche.401D45
0040102E 57 push edi
0040102F 57 push edi
00401030 50 push eax
00401031 FF75 08 08804000 call dword ptr ds:[<start...
00401037 FF75 F4 0C804000 push dword ptr ss:[ebp-C]
0040103A FF15 0C804000 call dword ptr ds:[<closeServiceHandle>]
00401040 60 01 push 1
00401042 5E pop esi
00401043 EB 56 jmp tasksche.401D9B
00401045 FF75 08 push dword ptr ss:[ebp+8]
00401046 48 8B 00FBFFF... mov eax,dword ptr ss:[ebp-40]
0040104E 68 2C440000 push tasksche.40F42c
00401053 50 push eax
00401054 FF15 1C814000 add esp,4
00401055 50 0C push 0
0040105D 8035 F4FBFFFF lea eax,dword ptr ss:[ebp-40]
00401063 57 push edi
00401064 57 push edi
00401065 57 push edi
00401066 57 push edi
00401067 57 push edi
00401068 50 push eax
00401069 6A 01 push 1
0040106B 6A 02 push 2
0040106D 6A 10 push 10
00401070 53 push 1
00401071 56 push esi
00401072 FF75 FC 08804000 call dword ptr ds:[<createService>]
00401075 FF15 08804000 mov esi,eax
00401078 8BF0 cmp edi,esi
0040107D 3B87 jae tasksche.401D9B
00401081 2A 17 push edi
00401083 57 push edi
00401085 56 push esi
0040108A FF15 08804000 call dword ptr ds:[<startService>]
0040108B FF15 0C804000 mov eax,dword ptr ss:[ebp-4]
0040108C C0 00 01000000 mov esp,dword ptr ss:[ebp-8]
00401088 8B75 08 push dword ptr ss:[ebp-4]
0040108B FF75 FC call dword ptr ds:[<closeServiceHandle>]

```

Registers and Stack Dump:

```

EAX 0000003A :'
EBX 000000FF :'
ECX 000000F2 :'
EDX 0000003A :'
EBP 0019F5C4 :'
ESP 0019F1AC &"tasksche.exe"
ESI 0040F8AC "kovbiruwfvvqwh488"
EDI 00000000
EIP 0040105D tasksche.0040105D

EFLAGS 000000206
ZF 0 PE 1 AF 0
OF 0 SF 0 DF 0
CF 0 TF 0 IF 1

LastError 000000424 (ERROR_SERVICE_DOES_NO...
LastStatus 00000034 (STATUS_OBJECT_NAME_NO...

GS 0028 FS 0053
DS 0028 DS 0028
CS 0023 SS 0028

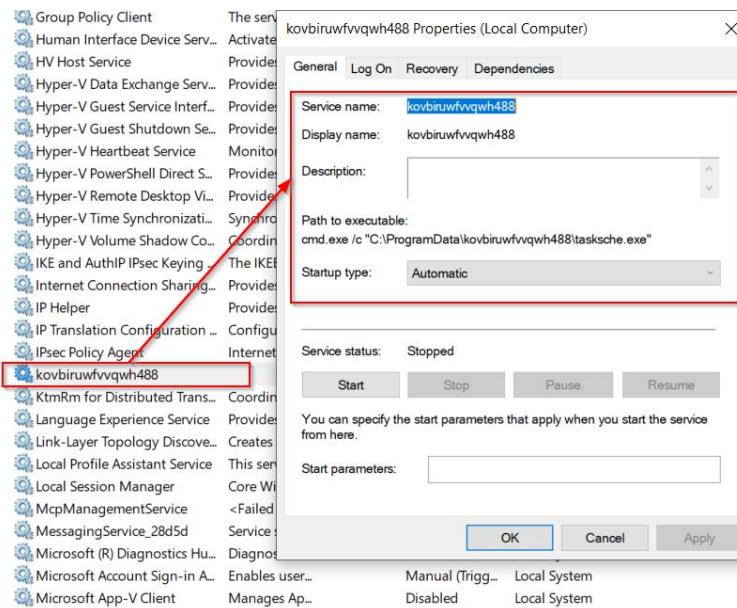
ST(0) 00000000000000000000000000000000 x87r0 Empty 0.0
ST(1) 00000000000000000000000000000000 x87r1 Empty 0.0
ST(2) 00000000000000000000000000000000 x87r2 Empty 0.0
ST(3) 00000000000000000000000000000000 x87r3 Empty 0.0
ST(4) 00000000000000000000000000000000 x87r4 Empty 0.0
ST(5) 00000000000000000000000000000000 x87r5 Empty 0.0
ST(6) 00000000000000000000000000000000 x87r6 Empty 0.0
ST(7) 00000000000000000000000000000000 x87r7 Empty 0.0

x87Tagword FFFF
x87Tw_0 3 (Empty) x87Tw_1 3 (Empty)
x87Tw_2 3 (Empty) x87Tw_3 (Empty)
x87Tw_4 3 (Empty) x87Tw_5 3 (Empty)
x87Tw_6 3 (Empty) x87Tw_7 3 (Empty)

Default (stdcall)
1: [esp-4] 00000000 00000000
2: [esp-8] 0019F7DC 0019F7DC
3: [esp+C] 2E646063 2E646063
4: [esp+10] 20557865 20557865
5: [esp+14] 2220632F 2220632F

```

### Evidence of service being created





Creates a reg key for wanacrypt0r in (HKLM|HKCU)/SOFTWARE/ hive

```

void wannacrypt0r_Regkey_creation(uint32_t arg_4h);

0x0040113c    stcs   word es:[edi], bx
0x0040113e    stcsb  byte es:[edi], al
0x0040113f    inc    es:[edi]
0x00401145    push   str.WanaCrypt0r ; 0x40e034 ; wchar_t *s2
0x00401146    call   dword [wccat] ; 0x408134 ; wchar_t *wcscat(wchar_t *s1, wchar_t *s2)
0x00401151    pop    ecx
0x00401155    pop    edx
0x00401156    pop    ecx
0x00401157    mov    edi, data.0040e030 ; 0x40e030

[0x00401152]    lea    eax, [hKey]
0x00401154    xor    esi, esi
0x00401155    cmp    dword [var_ch], esi
0x00401164    push   eax
0x00401165    lea    eax, [s1]
0x0040116b    push   eax
0x0040116c    jne    0x401175

[0x0040116e]    push   0x80000002
0x00401173    jmp    0x401175
0x00401175    push   0x80000001

[VX0040117a]
0x0040117a    call   dword [RegCreateKeyW] ; 0x408014 ; LSTATUS RegCreateKeyW(HKEY hKey, LPCTSTR
0x00401180    cmp    dword [hKey], esi
0x00401183    je     0x4011c0

[0x00401189]
0x00401189    cmp    je    dword [arg_4h], esi
0x0040118c    cmp    je    0x4011cc

[0x0040118e]
0x0040118e    lea    eax, [lpPathName]
0x00401194    push   eax
0x00401195    push   0x207 ; 519 ; NtOpenFileLength
0x00401195    push   0x4011cc

[0x004011cc]
0x004011cc    lea    eax, [lpchData]
0x004011cf    mov    dword [lpchData], 0x207 ; 519
0x004011d6    push   eax
0x004011d6    push   0x4011cc

```

And the registry key value is set to current working directory

	Name	Type	Data
(Default)	REG_SZ	(value not set)	
wd	REG_SZ	C:\Users\blue\Desktop	

Time	User	Action	Target	Status
9:11:45...	SYSTEM	RegQueryKey	HKLM	SUCCESS
9:11:45...	tasksche.exe	RegQueryKey	HKLM	SUCCESS
9:11:45...	tasksche.exe	RegCreateKey	HKLM\Software\WOW6432Node\WanaCrypt0r	SUCCESS
9:11:45...	tasksche.exe	RegSetInfoKey	HKLM\Software\WOW6432Node\WanaCrypt0r	SUCCESS
9:11:45...	tasksche.exe	RegQueryKey	HKLM\Software\WOW6432Node\WanaCrypt0r	SUCCESS
9:11:45...	tasksche.exe	RegSetValue	HKLM\Software\WOW6432Node\WanaCrypt0r\wd	SUCCESS
9:11:45...	tasksche.exe	RegCloseKey	HKLM\Software\WOW6432Node\WanaCrypt0r	SUCCESS



## High level overview of the main functions of tasksche.exe

After creating/setting the registry key, it unpacks an embedded resource and creates new files. Once the files have been extracted, it modifies the config file (c.wnry) and adds a random bitcoin address. Executes commands to make the directory hidden and then grants all access to everyone to all the subdirectories.

In the end, decrypts the t.wnry sample using an embedded rsa and aes key and then finally executes this program.

Detailed view of extracted files from 2058 resource in tasksche.exe

Initially loads the 2058 resource :

The screenshot shows a debugger interface with several windows. The assembly window displays x86 assembly code with various memory addresses highlighted in red. The memory dump window below shows the raw hex and ASCII data for the memory regions. The registers window shows CPU register values. The registers include EIP, ECX, EDX, EBX, ECSP, EDI, ECST, and ECDS. The stack pointer (ESP) is at address 00401E00. The instruction pointer (EIP) is at address 00401DC3. The memory dump shows data starting at address 00401000, including strings like "40E43C:XIA" and "Resource ID - 80A - 2058". The calculator window on the right shows the value 80A in various formats (HEX, DEC, OCT, BIN). The status bar at the bottom indicates the current memory location is at 00401000.

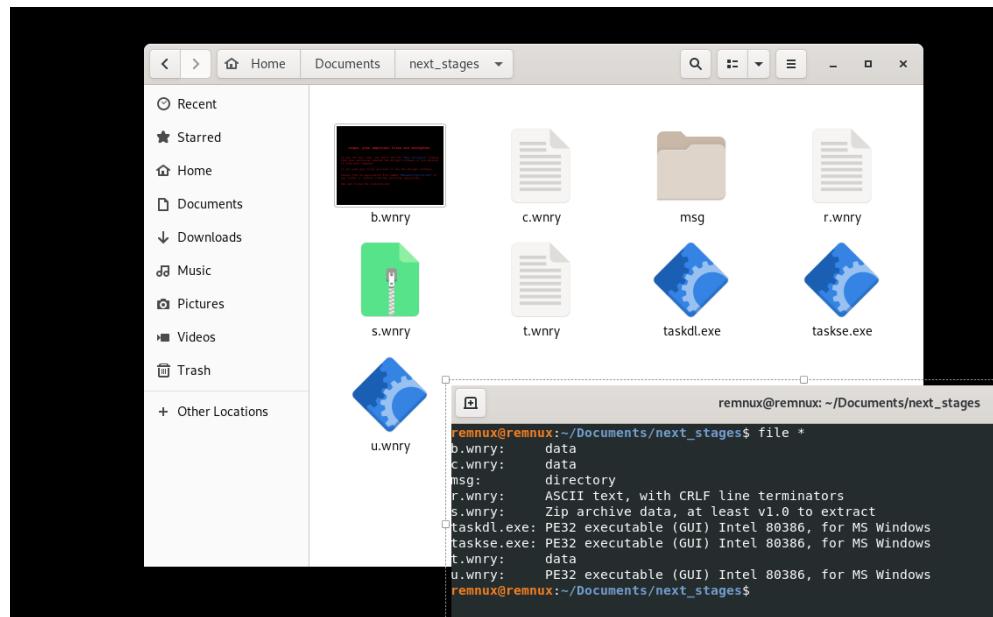


And then extracts all the files from this loaded resource. Over 36 files are created in this process.

The screenshot shows the Immunity Debugger interface with assembly code and registers. A red box highlights a section of assembly code that writes payload to a file named 'c.wnry'. Another red box highlights a CMP instruction comparing EBX with 0x24, which leads to a jump back to the creation routine if EBX is low. Below the debugger, Process Monitor shows numerous file operations performed by the 'tasksche.exe' process, listing actions like CloseFile, ReadFile, CreateFile, WriteFile, SetBasicInfor..., and CreateFile, many of which result in success.

### List of extracted files

Below is the list of files and directories that gets created. These files have been already detailed in the malware composition section.





1. b.wnry is a bitmap image that appears as a desktop wallpaper after infection :



2. c.wnry → is a config file containing the tor server(.onion domains) which are likely the command and control servers used by the threat actor.

## IOC domains :

gx7ekbenv2riucmf[.]jonion  
57g7spgrzlojinas[.]jonion  
xxlvbrlovxriy2c5[.]jonion  
76jdd2ir2embyv47[.]jonion  
cwnnhwhlz52maqm7[.]jonion



3. msg -> directory containing ransom note in different languages

```
remnux@remnux:~/Documents/next_stages$ ls msg
m_bulgarian.wnry          m_english.wnry        m_italian.wnry      m_romanian.wnry
'm_chinese (simplified).wnry'  m_filipino.wnry    m_japanese.wnry   m_russian.wnry
'm_chinese (traditional).wnry' m_finnish.wnry    m_korean.wnry     m_slovak.wnry
m_croatian.wnry           m_french.wnry       m_latvian.wnry    m_spanish.wnry
m_czech.wnry                m_german.wnry      m_norwegian.wnry  m_swedish.wnry
m_danish.wnry              m_greek.wnry       m_polish.wnry     m_turkish.wnry
m_dutch.wnry               m_indonesian.wnry  m_portuguese.wnry m_vietnamese.wnry
remnux@remnux:~/Documents/next_stages$
```

4. r.wnry => text file containing the instructions on how to recover the files

```
remnux@remnux:~/Documents/next_stages$ cat r.wnry
Q: What's wrong with my files?
A: Ooops, your important files are encrypted. It means you will not be able to access them anymore until they are decrypted.
If you follow our instructions, we guarantee that you can decrypt all your files quickly and safely!
Let's start decrypting!
Q: What do I do?
A: First, you need to pay service fees for the decryption.
Please send %s to this bitcoin address: %

Next, please find an application file named "%s". It is the decrypt software.
Run and follow the instructions! (You may need to disable your antivirus for a while.)
Q: How can I trust?
A: Don't worry about decryption.
We will decrypt your files surely because nobody will trust us if we cheat users.

* If you need our assistance, send a message by clicking <Contact Us> on the decryptor window.
remnux@remnux:~/Documents/next_stages$
```

5. s.wnry -> zip file containing the tor software and its dependency files

```
remnux@remnux:~/Documents/next_stages$ file s.wnry
s.wnry: Zip archive data, at least v1.0 to extract
remnux@remnux:~/Documents/next_stages$ 
remnux@remnux:~/Documents/next_stages$ 
remnux@remnux:~/Documents/next_stages$ unzip s.wnry
Archive: s.wnry
replace Tor/libeay32.dll? [y]es, [n]o, [A]ll, [N]one, [r]ename: A
  inflating: Tor/libeay32.dll
  inflating: Tor/libevent-2-0-5.dll
  inflating: Tor/libevent_core-2-0-5.dll
  inflating: Tor/libevent_extra-2-0-5.dll
  inflating: Tor/libgcc_s_sjlj-1.dll
  inflating: Tor/libssp-0.dll
  inflating: Tor/ssleay32.dll
  inflating: Tor/tor.exe
  inflating: Tor/zlib1.dll
remnux@remnux:~/Documents/next_stages$
```

6. taskdl.exe => support program to delete files from system

```
remnux@remnux:~/Documents/next_stages$ file taskdl.exe
taskdl.exe: PE32 executable (GUI) Intel 80386, for MS Windows
remnux@remnux:~/Documents/next_stages$
```



7. t.wnry → encrypted dll file which consists the encryption routine for the malware

```
remnux@remnux:~/Documents/next_stages$ file t.wnry
t.wnry: data
remnux@remnux:~/Documents/next_stages$ xxd t.wnry | head -n 5
00000000: 5741 4e41 4352 5921 0001 0000 1e38 2227  WANACRY!....8"
00000010: fde6 7f0c 5de7 7e3e 28a7 affd 2a50 6449  ....].~>(...*PdI
00000020: 66c6 b627 176d 3ed2 ff1c 32cb 8c30 8860  f...'.m>....2..0.
00000030: 70f6 eae9 9981 5e15 fe03 2349 7cbb ce3c  p.....^....#I|..<
00000040: ee57 e042 dc3d afa8 82b8 4d01 057a 7846  .W.B.=....M..zxF
remnux@remnux:~/Documents/next_stages$
```

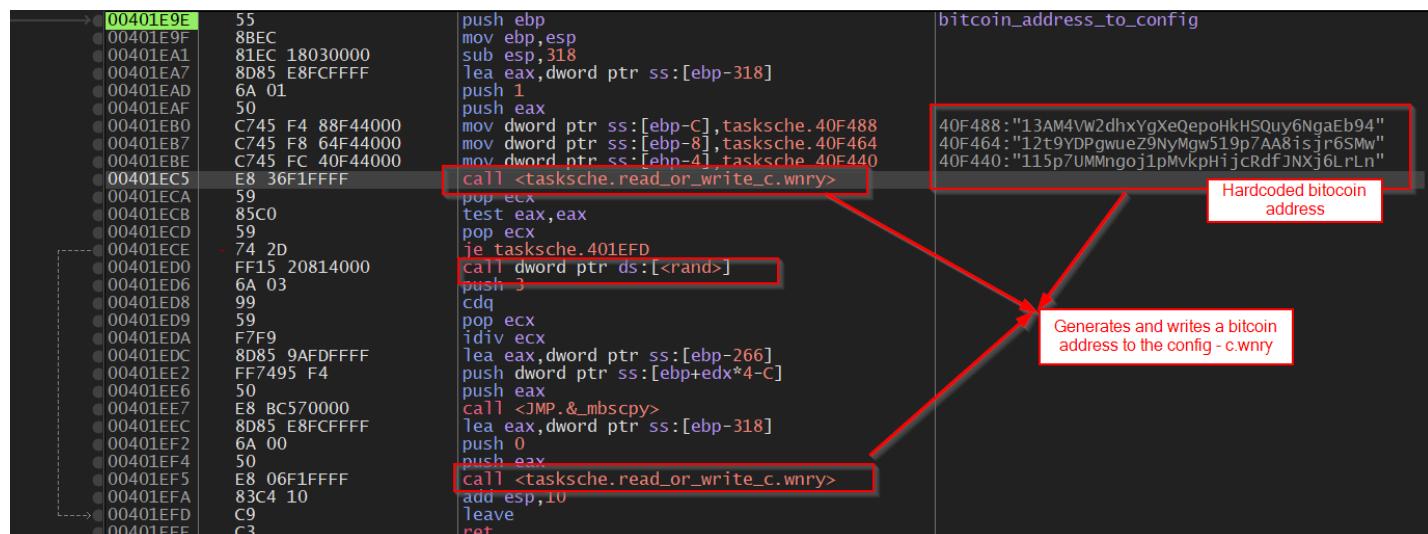
- ## 8. taskse.exe → PE file – unknown functionality

```
remnux@remnux:~/Documents/next_stages$ file taskse.exe  
taskse.exe: PE32 executable (GUI) Intel 80386, for MS Windows  
remnux@remnux:~/Documents/next_stages$
```

- ## 9. u.wnry → PE file --- unknown functionality

```
remnux@remnux:~/Documents/next_stages$ file u.wnry
u.wnry: PE32 executable (GUI) Intel 80386, for MS Windows
remnux@remnux:~/Documents/next_stages$
```

Next modifies c.wnry file and adds a random bitcoin address



After modification, we can see that c.wnry has a bitcoin address appended at the beginning.



## Command Execution

The program executes commands like “attrib +h” & “icacls . /grant Everyone:F /T /C /Q” to make the directory hidden and grant all access respectively.

## Unpack the t.wnry file

Initially reads the t.wnry sample, iteratively reading initial 8 bytes, next 4 and then 256 and then again 4 bytes.

The screenshot shows the Immunity Debugger interface with assembly code and a linked Process Monitor log.

**Assembly Code:**

```

00401572 E8 77610000 call    <JMP.&memcpy>
00401573 48C1 F0 add    eax, 0C
0040157A 85C0 test   eax, eax
0040157C .- OF85 4E010000 jne    tasksche.401600
00401582 53 push   ebx
00401583 8D45 E4 lea    eax, dword ptr ss:[ebp-1C]
00401584 50 push   eax
00401587 0A 04 push   4
00401589 8D85 BCFDFFFFF push  eax
0040158F 50 push   eax
00401590 57 push   edi
00401591 FF15 80F84000 call    dword ptr ds:[&<ReadFile>]
00401597 C3 C0 ret
00401599 .- OF84 31010000 je     tasksche.401600
0040159F B8 00010000 mov    eax, 100
004015A4 3985 BCFDFFFFF cmp    dword ptr ss:[ebp-244], eax
004015A5 0005 20010000 jns    tasksche.401600
004015A8 53 push   ebx
004015B1 8D4D E4 lea    ecx,dword ptr ss:[ebp-1C]
004015B4 51 push   ecx
004015B5 50 push   eax
004015B6 FF15 C8040000 call    dword ptr ds:[esi+4C8]
004015B8 57 push   edi
004015C3 FF15 80F84000 call    dword ptr ds:[&<ReadFile>]
004015C9 85C0 test   eax, eax
004015C9 JE     tasksche.401600
004015CC 8D45 E4 lea    eax,dword ptr ss:[ebp-1C]
004015CF 50 push   eax
004015D0 60 04 push   4
004015D2 8D85 BCFDFFFFF push  eax
004015D4 50 push   eax
004015D5 57 push   edi
004015D6 FF15 80F84000 call    dword ptr ds:[&<ReadFile>]
004015D8 85C0 test   eax, eax
004015D9 OF84 05010000 je     tasksche.401600
004015E0 53 push   ebx
004015E2 57 push   edi
004015E4 FF15 80F84000 call    dword ptr ds:[&<ReadFile>]
004015E6 85C0 test   eax, eax
004015E7 OF84 E8000000 je     tasksche.401600
004015E9 53 push   ebx
004015EA 57 push   edi

```

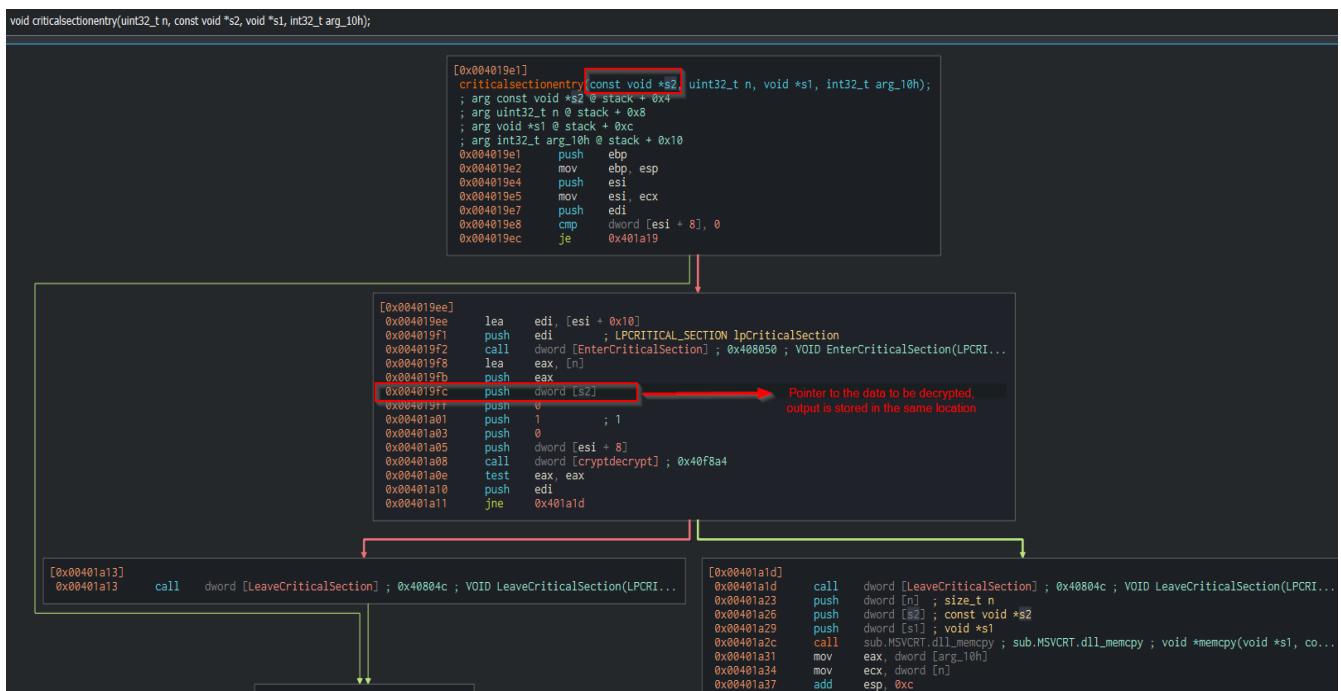
**Process Monitor Log:**

Time	Process Name	PID	Operation	Path	Result	Detail
10:25:51.000	tasksche.exe	1736	QueryStandardI	C:\Users\blue\Desktop\twnry	SUCCESS	AllocationSize: 69,632, EndOfFile: 65,816, NumberOfLinks: 1, DeletePending: False, Directory: False
10:26:13.000	tasksche.exe	1736	ReadFile	C:\Users\blue\Desktop\twnry	SUCCESS	Offset 0, Length: 8, Priority: Normal
10:27:44.000	tasksche.exe	1736	ReadFile	C:\Users\blue\Desktop\twnry	SUCCESS	Offset 8, Length: 4
10:28:04.000	tasksche.exe	1736	ReadFile	C:\Users\blue\Desktop\twnry	SUCCESS	Offset 12, Length: 256
10:28:07.000	tasksche.exe	1736	ReadFile	C:\Users\blue\Desktop\twnry	SUCCESS	Offset 268, Length: 4

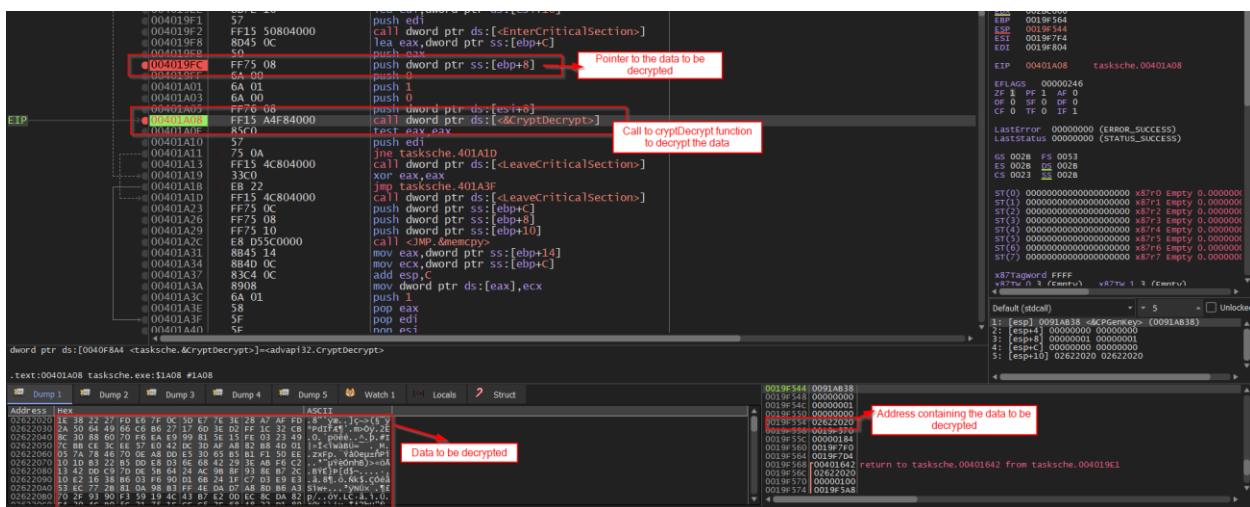
A red box highlights the first memory read operation (offset 0, length 8) in both the assembly and the Process Monitor log. A callout box points to the assembly code for this read, with the text: "Reads the twnry file iteratively, first 8 bytes, next 4, then 256 and then again 4 bytes."



Then makes a call to a function that decrypts the data that was read from t.wnry using an embedded RSA key in the program.



### *Decryption routine/ unpacking call*



*Figure 16 Before Decryption call*



## Data after decryption

Doing some OSINT on the t.wnry file, I found that the first 16 bytes this decrypted data contains an AES key which can be used to decrypt the rest of the encrypted data in t.wnry sample.

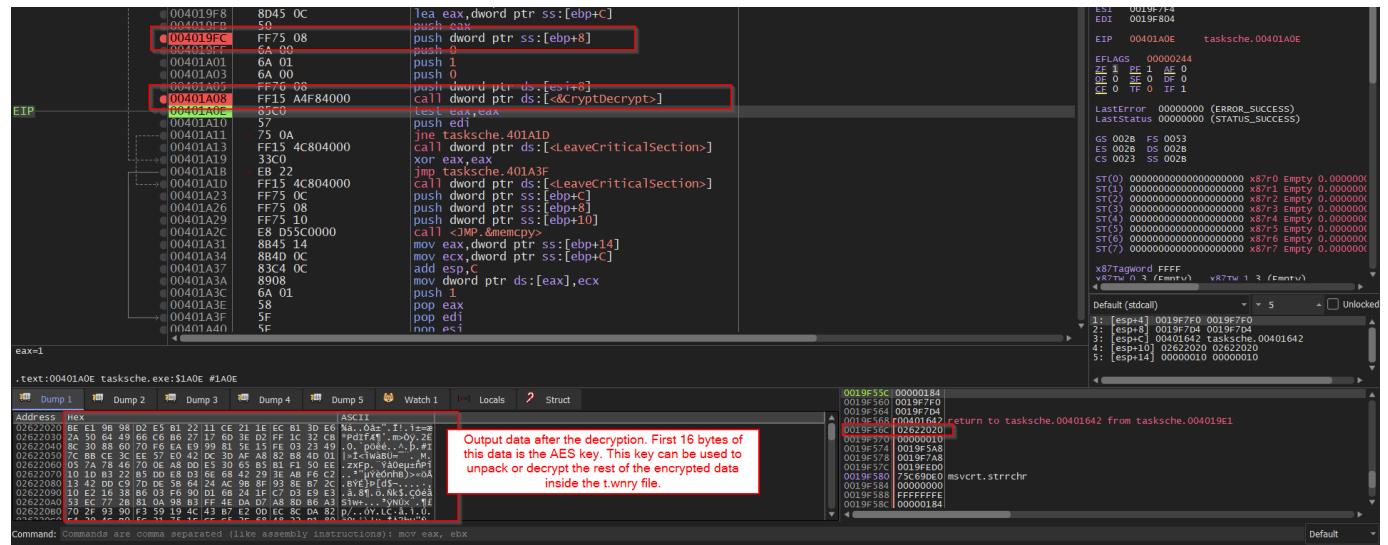


Figure 17 After the decryption call

A simple python script can be used to decrypt or unpack this sample.

Used this script from ghidraninja to decrypt the rest of the data in t.wnry file

[https://github.com/ghidraninja/ReversingWannacry/blob/master/decrypt\\_large\\_chunk.py](https://github.com/ghidraninja/ReversingWannacry/blob/master/decrypt_large_chunk.py)

And found this to be a windows dll file.

```
remnux@remnux: ~/Documents/next_stages
$ ./large_chunk.dec
remnux@remnux: ~/Documents/next_stages$
```

The figure shows two terminal windows. The left window shows the execution of a python script named `decrypt.py` to decrypt the data. The right window shows the resulting decrypted DLL file being executed.

```
remnux nano 4.8
remnux@remnux: ~/Documents/next_stages
#!/usr/bin/env python3

from Crypto.Cipher import AES

# The bytes we just decrypted
key = bytes.fromhex("bee19b98d2e5b12211ce211eebc13de6")

# Try an empty IV
iv = 16 * b'\x00'

# Initialize cipher
cipher = AES.new(key, AES.MODE_CBC, iv=iv)

# Read in large_chunk.bin
f = open("large_chunk.bin", "rb").read()

# Decrypt it
d = cipher.decrypt(f)

# Write it to large_chunk.dec
fout = open("large_chunk.dec", "wb")
fout.write(d)
```



After this, it executes this decrypted dll file by calling the taskstart function and then prepares the program for exit by clearing all the pointers and memory.

Address	OpCode	Description
0040212D	E8 74F3FFFF	call <tasksche.unpack_t.wnry>
00402132	3BC3	cmp eax,ebx
00402134	74 24	je tasksche.40215A
00402136	FF75 FC	push dword ptr ss:[ebp-4]
00402139	50	push eax
0040213A	E8 7E000000	call tasksche.4021BD
0040213F	59	pop ecx
00402140	3BC3	cmp eax,ebx
00402142	59	pop ecx
00402143	74 15	je tasksche.40215A
00402145	68 F8F44000	push tasksche.40F4E8
0040214A	50	push eax
0040214B	E8 D4070000	call <tasksche.execute_dll_main_func_taskstart>
00402150	59	pop ecx
00402151	3BC3	cmp eax,ebx
00402153	59	pop ecx
00402154	74 04	je tasksche.40215A
00402156	53	push ebx
00402157	53	push ebx
00402158	FFD0	call eax
0040215A	8D8D 1CF9FFFF	lea ecx,dword ptr ss:[ebp-6E4]
00402160	E8 15F2FFFF	call <tasksche.prepare_exit>
00402165	5F	pop edi
00402166	5E	pop esi
00402167	33C0	xor eax,eax
00402169	5B	pop ebx
0040216A	C9	leave

Annotations:

- 40F4E8: "TaskStart" - Points to the call instruction at 0040214B.
- Executes / runs the dll file by calling the Taskstart function in dll - Description for the TaskStart call.
- Prepares for exit - Points to the call instruction at 00402160.

I am going to stop the advanced analysis at this point. The execution of dll leads to the encryption of all our files on computer and the popup for decryptor appears on the desktop.

It looks like the dll likely has the encryption routine functionality.



# Indicators of Compromise

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

## Network Indicators

Initial killswitch domain:

www.iuquerfsodp9ifjaposdfjhgosurijfaewrwerwgea[.]com

Command and control servers from the config file (c.wnry):

gx7ekbenv2riucmf[.]onion  
57g7spgrzlojinash[.]onion  
xxlvbrloxvriy2c5[.]onion  
76jdd2ir2embyv47[.]onion  
cwwnhwhlz52maqm7[.]onion

## Host-based Indicators

See the [malware composition](#) & list of [extracted files](#) section for host-based indicators

File Home Share View Application Tools				
This PC > Local Disk (C) > Windows				
	Name	Date modified	Type	Size
Quick access	bootstat.dat	5/19/2024 5:59 AM	DAT File	66 KB
Desktop	taskscche.exe	5/19/2024 5:54 AM	Application	3,432 KB
Downloads	WindowsUpdate.log	5/19/2024 5:52 AM	Text Document	1 KB
Documents	DtcInstall.log	2/13/2024 6:13 PM	Text Document	2 KB
Pictures	Isasetup.log	2/13/2024 6:12 PM	Text Document	2 KB
Music	PFRO.log	2/13/2024 7:51 AM	Text Document	4 KB
Videos	SMSS-PFRO080e.tmp	1/14/2024 9:36 PM	TMP File	399 KB
	FindDll.exe	6/2/2023 5:28 PM	Application	50 KB

Figure 18 Main payload of wannacry dropped by initial detonation

Share View				
This PC > Local Disk (C) > ProgramData > kovbiruwfvqwh488				
	Name	Date modified	Type	Size
test	msg	5/19/2024 5:55 AM	File folder	
ads	TaskData	5/19/2024 5:55 AM	File folder	
ents	@Please_Read_Me@.txt	5/19/2024 5:54 AM	Text Document	1 KB
ects	@WanaDecryptor@.exe	5/12/2017 2:22 AM	Application	240 KB
ents	@WanaDecryptor@.exe	5/19/2024 5:54 AM	Shortcut	1 KB
ads	00000000.eky	5/19/2024 5:54 AM	EKY File	2 KB
isks (C):	00000000.pky	5/19/2024 5:52 AM	PKY File	1 KB
e (D): Virtua	00000000.res	5/19/2024 6:01 AM	RES File	1 KB
	b.wnry	5/11/2017 8:13 PM	WNRY File	1,407 KB
	c.wnry	5/19/2024 5:56 AM	WNRY File	1 KB
	f.wnry	5/19/2024 5:55 AM	WNRY File	1 KB
	r.wnry	5/11/2017 3:59 PM	WNRY File	1 KB
	s.wnry	5/9/2017 4:58 PM	WNRY File	2,968 KB
	t.wnry	5/12/2017 2:22 AM	WNRY File	65 KB
	taskdl.exe	5/12/2017 2:22 AM	Application	20 KB
	taskscche.exe	5/19/2024 5:54 AM	Application	3,432 KB
	taskse.exe	5/12/2017 2:22 AM	Application	20 KB
	u.wnry	5/12/2017 2:22 AM	WNRY File	240 KB

Figure 19 Staging folder and support programs for wannacry

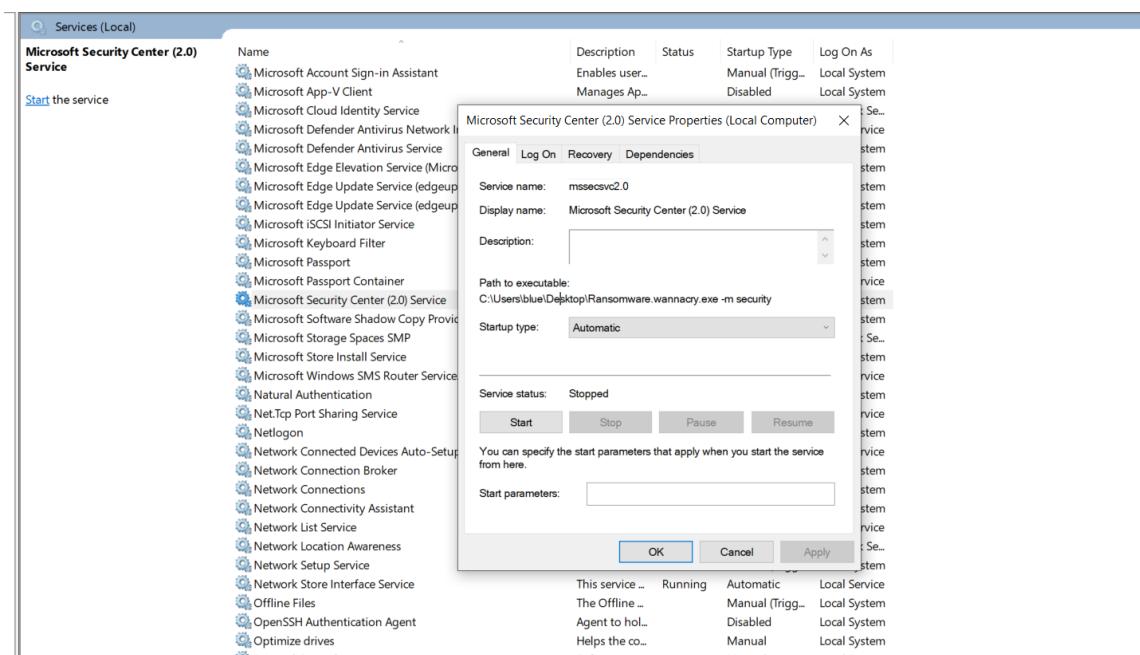


Figure 20 mscsvc service that launches wannacry payload to spread through smb

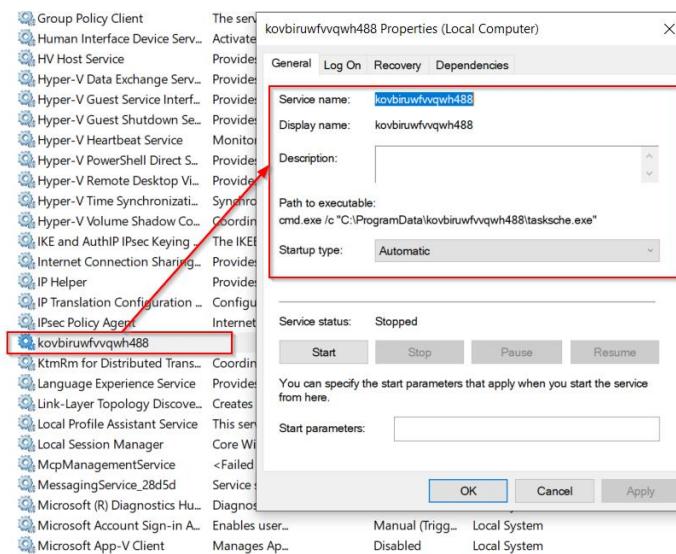


Figure 21 Tasksche service



# Rules & Signatures

## A. Yara Rules

```
rule ransomware_wannacry {
    meta:
        author = "Aniksha Shetty"
        description = "Detect wanancryransomware file"
        created = "04-15-2024"

    strings:
        $weird_file_name = "C:\\\\%s\\\\qeriuwjhrf" ascii
        $kill_switch_url = "iuqerfsodp9ifjaposdfjhgosurijfaewrwegwea.com" ascii
        $bitcoin_addr1 = "115p7UMMngoj1pMvkpHijcRdfJNXj6LrLn" ascii
        $bitcoin_addr2 = "12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw" ascii
        $bitcoin_addr3 = "13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94" ascii
        $passkey = "WNcry@2o17" ascii
        $ipc_share = "\\\\%s\\\\IPC$" ascii
        $file_extension = ".wnry"
        $pe_magic_byte = "MZ"

    condition:
        $pe_magic_byte at 0 and
        3 of them
}
```

## B. Callback URLs

Domain	Type
www.iuqerfsodp9ifjaposdfjhgosurijfaewrwegwea[.]com	Kill Switch domain
76jdd2ir2embyv47[.]onion	C2
cwwnhwhlz52maqm7[.]onion	C2
xxlvbrloxvriy2c5[.]onion	C2
gx7ekbenv2riucmf[.]onion	C2
57g7spgrzlojinaz[.]onion	C2