

Date: 02/18/2026

Author: <https://github.com/NoxCaelum>

Malware Analysis Report

WannaCry Ransomware

Summary

I. Introduction.....	3
II. Methodology.....	3
III. Malware profile.....	3
A. File information.....	3
B. Public intelligence and pivots.....	3
IV. Static Analysis.....	4
A. Malware behavior.....	4
B. Malware integrated resources.....	4
V. Dynamic Analysis.....	4
A. System modification.....	4
B. Process and Services Creation.....	4
VI. Conclusion.....	4

I. Introduction

This report presents an analysis of the WannaCry Ransomware. The purpose of this analysis is to identify the behavior and the IoCs (Indicators of Compromise) of the ransomware. This study aims to provide insight into ransomware workflows.

II. Methodology

The analysis was conducted in a virtualized Windows 10 operating system, without internet connection and Microsoft Defender. Both static and dynamic analysis as well as automatic triage, were employed to conduct the ransomware analysis. The following tools were used to conduct the static analysis: *Floss*, *DetectItEasy*, *Ghidra*, *ResourceHacker*, *PeBear*, *HxD* and custom scripts. In addition, the tools used to conduct the dynamic analysis were: *Regshot*, *SystemInformer* and *ProcMon* as well as *yara*, *BinaryAlert* and *capa* for automatic triage. Threat intelligence resources like *MITRE ATT&CK*, *Malpedia* and *AnyRun*, were also used to gather additional information such as threat actors and TTPs (Tactics, Techniques and Procedures).

III. Malware profile

WannaCry is a [crypto-ransomware](#) that is known for its rapid propagation in 2017. It targeted the Microsoft Windows operating system and encrypted user's data in order to demand payment of a ransom in cryptocurrency. WannaCry contained a worm component that exploited the [EternalBlue](#) vulnerability in SMBv1 protocol, to remotely spread itself and compromise other hosts in the network. To avoid sandbox detection, the worm component attempted an HTTP connection to a [killswitch domain](#), if the connection was successful, the malware stop itself.

A. File information

Filename:	wncry.exe
SHA-256 Hash:	ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa
File Location/Source:	https://github.com/ytisf/theZoo/blob/master/malware/Binaries/Ransomware.WannaCry/Ransomware.WannaCry.zip
Date Acquired:	02/13/2026
Detection Context:	Malware Analysis

B. Public intelligence and pivots

VirusTotal Link	https://www.virustotal.com/gui/file/ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa
VT Detection Ratio:	67/71
Notable VT Vendor Signatures:	Microsoft Security: Ransom:Win32/WannaCrypt!pz
VT First Submitted:	2017-05-12 07:31:10 UTC
VT Behavioral Summary:	https://www.virustotal.com/gui/file/ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa/behavior
VT Pivots	<p>Vhash: 036046656d1570a8z3631lz1fz</p> <p>Imphash: 68f013d7437aa653a8a98a05807afeb1</p> <p>Rich Pe header Hash: 417a06d07f984f3bce5cd06546c98842</p> <p>TLSH: T173F533F4E221B7ACF2550EF64855C59B6A9724B2EBEF1E26DA8001A70D44F7F8FC0491</p> <p>SSDEEP: 98304:QqPoBhz1aRxcSUDk36SAEdhvxWa9P593R8yAVp2g3x:QqPe1Cxcxk3ZAEUadzR8yc4gB</p> <p>section MD5 hash: .text 920e964050a1a5dd60dd00083fd541a2 .rdata 2c42611802d585e6eed68595876d1a15 .data 83506e37bd8b50cacabd480f8eb3849b .rscr f99ce7dc94308f0a149a19e022e4c316 </p>
Public Sandbox Results:	https://app.any.run/tasks/941e801e-be9e-456a-8994-2d00ae0c94c8
Malpedia Results:	https://malpedia.caad.fkie.fraunhofer.de/details/win.wannacryptor
Known Actor/Campaign Associations (if available):	https://mitre-attack.github.io/attack-navigator/#layerURL=https%3A%2F%2Fattack.mitre.org%2Fsoftware%2FS0366%2FS0366-enterprise-layer.json https://attack.mitre.org/software/S0366/
Other OSINT Results: BTC transactions on the blockchain	Wallet1: https://www.blockchain.com/explorer/addresses/btc/115p7UMMngoj1pMvvpHijcRdfJNXj6LrLn Wallet2: https://www.blockchain.com/explorer/addresses/btc/12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw Wallet3: https://www.blockchain.com/explorer/addresses/btc/13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94

IV. Static Analysis

In this section, we will dive into the static analysis part of the WannaCry sample. It imply all of the analysis techniques performed without executing the sample. This approach is useful for examining the binary structure, code, patterns, and strings, to provide a comprehensive understanding of the malware's behavior.

A. Automatic detection

To begin with automatic triage, we used Yara and BinaryAlert with the “yara-rules-full.yar” rules to detect if the sample possess known pattern or signature: *yara64.exe -w -s yara-rules-full.yar wncry.exe*.

Output:

```
BINARYALERT_Ransomware_Windows_Wannacry ..\Malware analyse\wncry.exe
0x15e49:$a1: msg/m_chinese
0x189aa:$a1: msg/m_chinese
0x358a7b:$a1: msg/m_chinese
0x358aec:$a1: msg/m_chinese
(...)
0xf520:$a3: attrib +h
0xf52c:$b1: WNCry@2o17
0xf440:$b3: 115p7UMMngo1pMvkpHijcRdfJNXj6LrLn
0xf464:$b4: 12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw
0xf488:$b5: 13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94
SIGNATURE_BASE_Wannacry_Ransomware ..\Malware analyse\wncry.exe
0xf4fc:$x1: icacls . /grant Everyone:F /T /C /Q
0x342d41:$x2: taskdl.exe
0x35962d:$x2: taskdl.exe
0xf4d8:$x3: tasksche.exe
0xf4b4:$x4: Global\MsWinZonesCacheCounterMutexA
0xf52c:$x5: WNCry@2o17
0xf4fc:$x9: icacls . /grant Everyone:F /T /C /Q
0x359d91:$s2: <!-- Windows 10 -->
0xf42c:$s3: cmd.exe /c "%s"
0x41980:$s4: msg/m_portuguese.wnry
0x3591ff:$s4: msg/m_portuguese.wnry
0x2a02:$op4: 09 FF 76 30 50 FF 56 2C 59 59 47 3B 7E 0C 7C
0x26dc:$op5: C1 EA 1D C1 EE 1E 83 E2 01 83 E6 01 8D 14 56
0x22c8:$op6: 8D 48 FF F7 D1 8D 44 10 FF 23 F1 23 C1
```

The sample is recognized as “*Ransomware Windows Wannacry*” by it’s signature “*SIGNATURE BASE Wannacry Ransomware*”. We can also identified some strings that looks like system commands, binaries, cryptocurrency wallets and language packages.

In addition, we used capa to retrieve some binary functions:

```
capa -r .\capa-rules-9.3.1\ -f pe wncry.exe
```

Output:

```
encrypt data using AES (5 matches)
namespace  data-manipulation/encryption/aes
scope      function
(...)

encrypt data using RC4 KSA (3 matches)
namespace  data-manipulation/encryption/rc4
scope      function
(...)

get common file path (3 matches)
namespace  host-interaction/file-system
scope      function
(...)

create directory (2 matches)
namespace  host-interaction/file-system/create
scope      function
(...)

read file on Windows (3 matches)
namespace  host-interaction/file-system/read
scope      function
(...)

write file on Windows (2 matches)
namespace  host-interaction/file-system/write
scope      function
(...)

create process on Windows
namespace  host-interaction/process/create
scope      basic block
(...)

query or enumerate registry value
namespace  host-interaction/registry
scope      function
(...)

set registry value
namespace  host-interaction/registry/create
scope      function
(...)
```

```
create service
namespace host-interaction/service/create
scope      function

start service
namespace host-interaction/service/start
scope      function
(...)

persist via Windows service
namespace persistence/service
scope      function
(...)
```

The results indicate that the binary contains some functions that encrypt data, retrieve common file paths, create processes, enumerate and modify a registry keys and create and start services. It can therefore be assumed that the binary create a malicious service that encrypts user's data, and establishes persistence by modifying the registry and adding registries values that reference this malicious service.

Moreover, we used the FLOSS tools to output and analyse the binary strings: *FLOSS wncry.exe*

Output:

```
inflate 1.1.3 Copyright 1995-1998 Mark Adler
unzip 0.15 Copyright 1998 Gilles Vollant
Microsoft Enhanced RSA and AES Cryptographic Provider
(...)
CreateProcessA
TerminateProcess
GetExitCodeProcess

(...)
LoadLibraryA
GetProcAddress
GetModuleHandleA

(...)
CreateServiceA
OpenServiceA
StartServiceA
OpenSCManagerA

(...)
RegCreateKeyW
RegSetValueExA
RegQueryValueExA
RegCloseKey

(...)
CryptGenKey
```

```
CryptDecrypt
CryptEncrypt
CryptDestroyKey
CryptImportKey
CryptAcquireContextA
(...)
cmd.exe /c "%s"
tasksche.exe
TaskStart
icacls . /grant Everyone:F /T /C /Q
attrib +h .
(...)
.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 .s
qlitedb .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 .bac
kup .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 .xltn .xltx .xlc .xlm .xlt .xlw .xlsb .xlsm .xlsx .xls .dotx .dotm .dot .docm .docb .docx .doc
(...)
115p7UMMngoj1pMvvpHijcRdfJNXj6LrLn
12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw
13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94
```

The extracted strings reveal several noteworthy elements. The presence of “*inflate 1.1.3 Copyright 1995–1998 Mark Adler*” and “*unzip 0.15 Copyright 1998 Gilles Vollant*” suggests the inclusion of a ZIP compression component. Similarly, references to “*Microsoft Enhanced RSA and AES Cryptographic Provider*” and multiple *Crypt** functions indicate the implementation of cryptographic mechanisms.

Functions such as *CreateProcessA*, *CreateServiceA*, and *RegCreateKeyW/RegSetValueExA* confirm the creation of a service and registry modifications. The presence of *LoadLibraryA*, *GetProcAddress*, and *GetModuleHandleA* further indicates that the malware implements dynamic function resolution.

The strings also reference file extensions that may be the files impacted by the cryptographic mechanism, and cryptocurrency wallets.

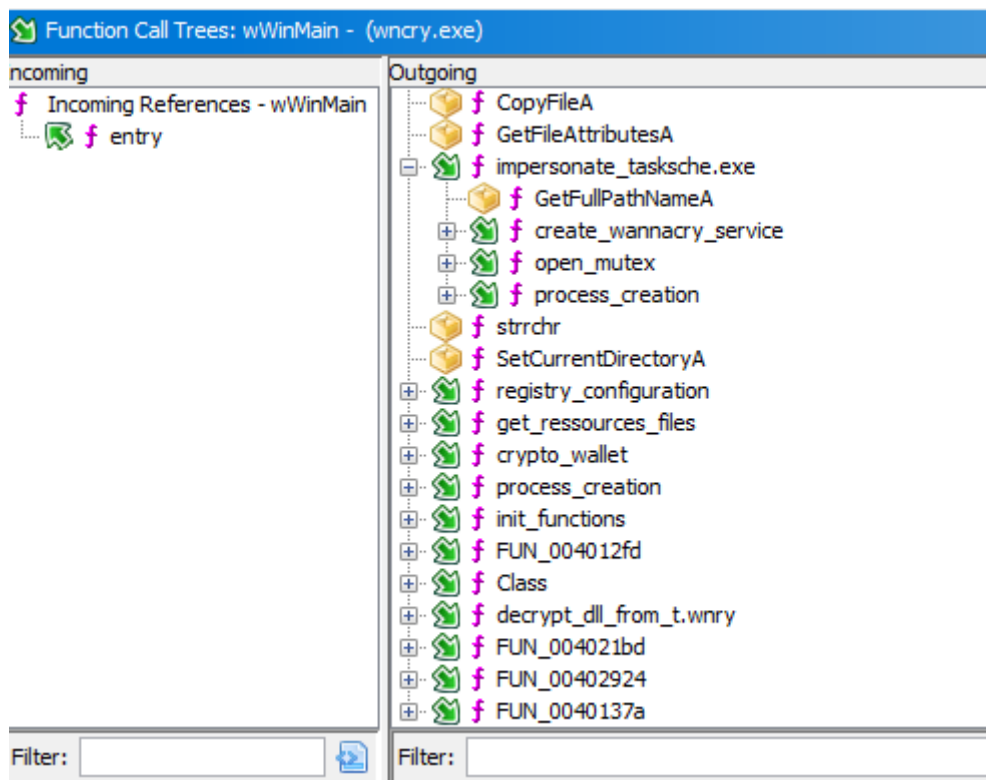
It is noteworthy to examine the system commands executed by the malware. The command “*cmd.exe /c "%s"*” indicates that the malware executes a command and immediately close the windows command prompt. The command “*icacls . /grant Everyone:F /T /C /Q*” silently grants full control permissions to everyone on all the files within the directory and sub-directories and finally, “*attrib +h*” may assign the hidden attribute to some files.

B. Code Analysis

The code analysis of `wWinMain()` and its core functions confirm several key points inferred during the automatic triage phase. The malware creates a malicious service via the “`impersonate_tasksche.exe`” function, the ransomware copy itself as “`tasksche.exe`” and execute itself via the system command “`cmd.exe /c %s`” through “`create_wannacry_service`”.

It establish persistence by modifying the registry through the “`registry_configuration`” function and retrieves compressed resources using the “`get_resources_files`” function.

Wncry.exe: Ghidra - Function Call Tree



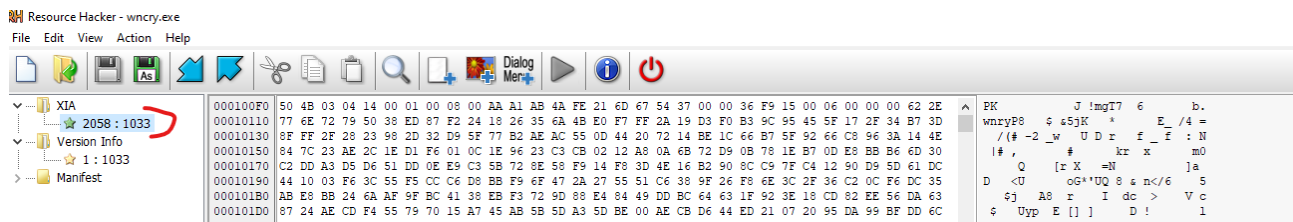
The “`get_ressources_file`” functions take as an argument an hardcoded password: “`WNcry@2017`” and look for a file named “`2058`” with the `FindResourceA()` function. The malware use this password to unzip its compressed file.

Wncry.exe: Ghidra – get_resources_file function

<pre>0040f52c 57 4e 63 ds "WNcry@2017" 72 79 40 32 6f 6c ... 0040f537 00 ?? 00h</pre>	<pre>XREF[1]: wWinMain:004020c8(*)</pre>	<pre>42 } 43 } 44 pcVar4 = strchr((pointer_buffer_lpFileName, 0x5c); 45 if (pcVar4 != (char *)0x0) { 46 pcVar4 = strchr((pointer_buffer_lpFileName, 0x5c); 47 *pcVar4 = '\0'; 48 } 49 SetCurrentDirectoryA((pointer_buffer_lpFileName); 50 registry_configuration(1); 51 get_ressources_files((HMODULE) 0x0, s_WNcry82c17_0040f52c);</pre>
---	---	--

We can manually extract this resource using *ResourceHacker*:

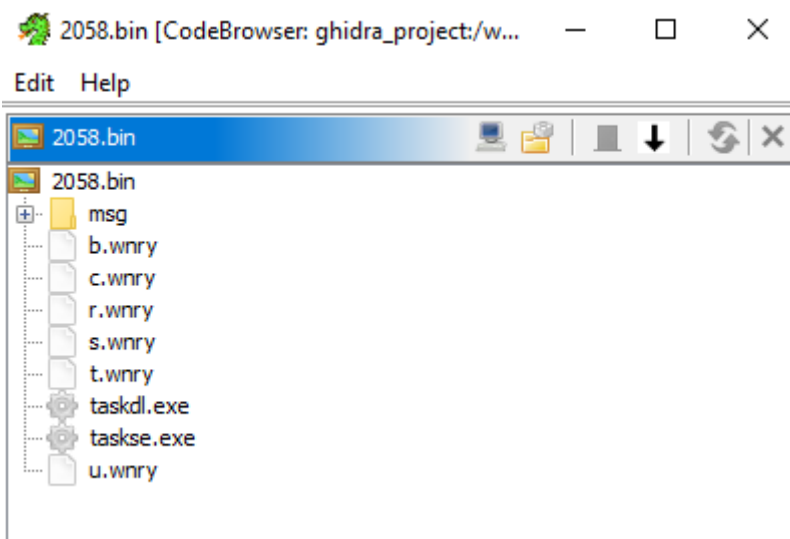
Wncry.exe: ResourceHacker



As a result, it can be observed that this component contains the resources used by wncry.exe such as:

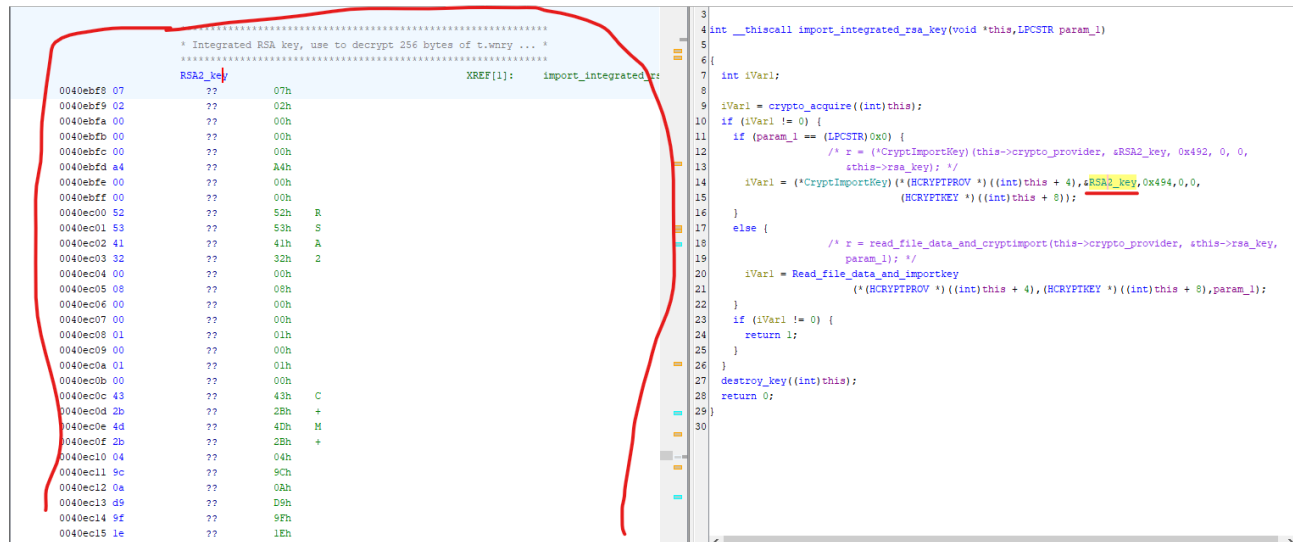
- b.wnry: bitmap image used as a wallpaper with the demand of a ransom.
- c.wnry: tor Command&Control addresses
- r.wnry: ransom message
- s.wnry: tor software
- u.wnry: decryptor module
- t.wnry: Encrypted DLL
- taskdl.exe: temporary file cleanup program
- taskse.exe: display decryptor windows to RDP sessions
- msg: contain the ransom demand in multiple languages

2058.bin: Ghidra



The programme leverages an hardcoded RSA in the decrypt_rsa_key function, to decrypt an encrypted AES key in t.wnry.

Wncry.exe: Ghidra - decrypt_rsa_key



Therefore, with a custom script that can be found at https://github.com/NoxCaelum/WannaCry_reverse_engineering, we can recreate the wncry.exe workflow to decrypt the AES encrypted key and the encrypted DLL in t.wnry.

Decrypted DLL:

```
FLARE-VM Wed 02/18/2026 9:38:19.38
C:\Users\test\Desktop\Malware analyse>file decrypted_twnry.dec
decrypted_twnry.dec: PE32 executable (DLL) (GUI) Intel 80386, for MS Windows
```

This DLL is responsible for the file encryption on the host.

V. Dynamic Analysis

In this section, we will focus on the dynamic analysis of the WannaCry sample. It involves executing the sample in a controlled and isolated environment to observe its real-time behavior. This approach is useful for monitoring file system changes, registry modifications, network communications, and process activities, providing a complete understanding of how the malware operates during execution.

A. System modification

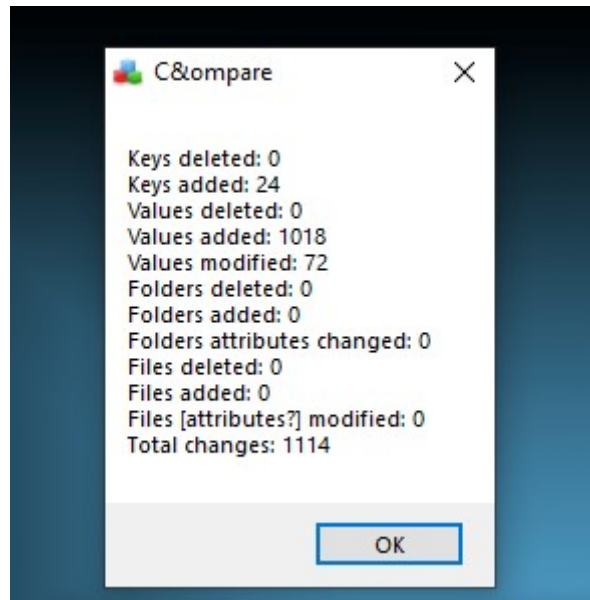
Dynamic analysis confirmed the prior static analysis. When the wncry.exe is executed, it encrypts the user's file, changes the desktop wallpaper using the bitmap image contained in c.wnry and displays a window presenting the ransom demands along with a cryptocurrency wallet.

Wncry.exe: WanaDecrypt0r@.exe



A comparison of the system before and after the execution of wncry.exe, using the *Regshot* tools, highlights key actions performed by the malware:

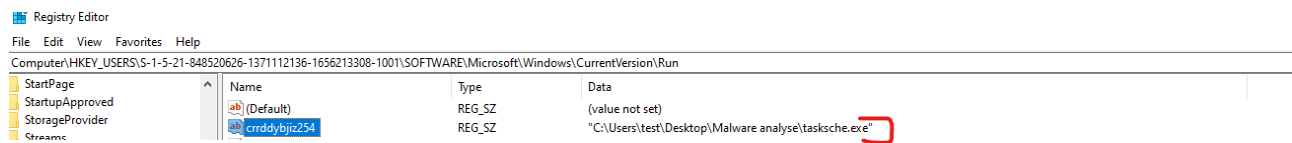
Regshot output:



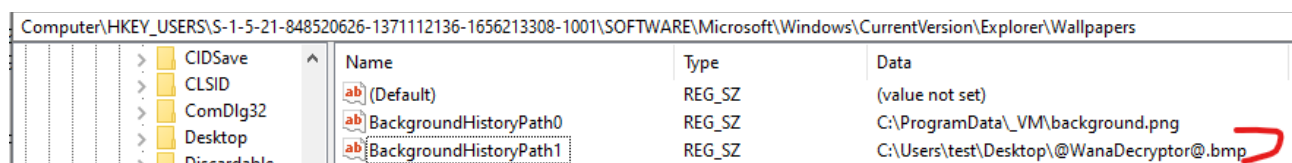
Registry keys added:

```
HKU\S-1-5-21-848520626-1371112136-1656213308-1001\SOFTWARE\Classes\VirtualStore\MACHINE\SOFTWARE\WOW6432Node\WanaCrypt0r
HKU\S-1-5-21-848520626-1371112136-1656213308-1001\SOFTWARE\WanaCrypt0r
HKU\S-1-5-21-848520626-1371112136-1656213308-1001_Classes\VirtualStore\MACHINE\SOFTWARE\WOW6432Node\WanaCrypt0r
HKU\S-1-5-21-848520626-1371112136-1656213308-1001\SOFTWARE\WanaCrypt0r\wd
HKLM\SYSTEM\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-848520626-1371112136-1656213308-1001\Device\HarddiskVolume3\Users\test\Desktop\Malware
analyse\@WanaDecryptor@.exe
HKU\S-1-5-21-848520626-1371112136-1656213308-1001\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\crrddybjz254 "C:\Users\test\Desktop\Malware
analyse\tasksche.exe"
HKU\S-1-5-21-848520626-1371112136-1656213308-1001\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Wallpapers\BackgroundHistoryPath1 "C:
\Users\test\Desktop\@WanaDecryptor@.bmp"
HKU\S-1-5-21-848520626-1371112136-1656213308-1001\SOFTWARE\Classes\VirtualStore\MACHINE\SOFTWARE\WOW6432Node\WanaCrypt0r
```

It can be noteworthy that the registry key: “HKU\S-1-5-21-848520626-1371112136-1656213308-1001\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\crrddybjiz254” establish persistence by executing “C:\Users\test\Desktop\Malware analyse\tasksche.exe” causing tasksche.exe, the cryptographic component of wncry.exe, to automatically start at each sessions startup.



The registry key: “HKU\S-1-5-21-848520626-1371112136-1656213308-1001\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Wallpapers\BackgroundHistoryPath1” refer to the “C:\Users\test\Desktop\@WanaDecryptor@.bmp” that contain the bitmap used as a desktop wallpaper.



Moreover, the malware dropped new files such as:

- [b.wnry: bitmap image used as a wallpaper with the demand of a ransom.](#)
- [c.wnry: tor Command&Control addresses](#)
- [r.wnry: ransom message](#)
- [s.wnry: tor software](#)
- [u.wnry: decryptor module](#)
- [t.wnry: Encrypted DLL](#)
- [taskdl.exe: temporary file cleanup program](#)
- [taskse.exe: display decryptor windows to RDP sessions](#)
- [msg: contain the ransom demand in multiple languages](#)
- [00000000.pk: PUBLICKEYBLOB containing the RSA public key](#)
- [00000000.res: data for C2 communication](#)
- [00000000.dky: Decrypted RSA private key transmitted to victim after ransom payment](#)
- [f.wnry: randomly encrypted file with embedded RSA private key](#)
- [@WannaDecryptor@exe: Main module of the ransomware decryptor \(u.wnry\)](#)
- [@Please Read Me@txt: Ransom demand text \(r.wnry\)](#)

Wncry.exe: Dropped files after execution

Name	Date modified	Type	Size
msg	2/18/2026 3:53 AM	File folder	
TaskData	2/18/2026 3:54 AM	File folder	
@Please_Read_Me@.txt	2/18/2026 3:52 AM	Text Document	1 KB
@WanaDecryptor@.exe	5/12/2017 3:22 AM	Application	240 KB
@WanaDecryptor@.exe	2/18/2026 3:52 AM	Shortcut	1 KB
00000000.eky	2/18/2026 3:52 AM	EKY File	2 KB
00000000.pky	2/18/2026 3:52 AM	PKY File	1 KB
00000000.res	2/18/2026 3:55 AM	RES File	1 KB
b.wnry	5/11/2017 9:13 PM	WNRy File	1,407 KB
c.wnry	2/18/2026 3:54 AM	WNRy File	1 KB
f.wnry	2/18/2026 3:52 AM	WNRy File	1 KB
r.wnry	5/11/2017 4:59 PM	WNRy File	1 KB
s.wnry	5/9/2017 5:58 PM	WNRy File	2,968 KB
t.wnry	5/12/2017 3:22 AM	WNRy File	65 KB
taskdl.exe	5/12/2017 3:22 AM	Application	20 KB
taskse.exe	5/12/2017 3:22 AM	Application	20 KB
u.wnry	5/12/2017 3:22 AM	WNRy File	240 KB
wncry.exe	5/14/2017 7:29 AM	Application	3,432 KB

Throughout the analysis using the SysInformer tool, it can be observed that “taskhsvc.exe” which is a local tor server for the Command&Control communication, is executed. This server establishes connections to the following domains:

- gx7ekbenv2riucmf.onion
- 57g7spgrzlojinas.onion
- xxlvbrloxvriy2c5.onion
- 76jdd2ir2embyv47.onion
- cwwnhwhlz52maq7.onion

▼ wncry.exe	2292
▼ @WanaDecryptor@.exe	436
▼ taskhsvc.exe	4780
conhost.exe	1744
@WanaDecryptor@.exe	3880

VI. Conclusion

The analysis of the WannaCry cryptographic module demonstrates its core functionality in file encryption and persistence. The malware creates a local service “*tasksche.exe*” to execute cryptographic operations, establishes persistence through registry modifications, and dynamically resolves functions to perform encryption routines. Embedded resources, including encrypted DLLs and configuration files, are retrieved and decrypted using a hardcoded RSA key on t.wnry.

Dynamic analysis confirmed that the module encrypts user files, modifies the desktop wallpaper with ransom information, and displays a ransom message alongside cryptocurrency wallets. System commands and registry changes observed during execution further ensure the malware maintains control and survives system reboots.

Overall, the cryptographic module exhibits a structured workflow, combining resource extraction, key management, encryption, and persistence mechanisms of the WannaCry file-encryption component.