Date: 02/18/2026

Author: https://github.com/NoxCaellum

Malware Analysis Report

WannaCry Ransomware

Summary

I. Introduction

This report present an analysis of the WannaCry Ransomware. The purpose of this analysis is to indentify the behavior and the IoCs (Indice of Compromission) of the ransomware. This study aims to provide insight of ransomwares 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[1] 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[2] 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[3], if the connection was successful, the malware stop itself.

WannaCry is structured in separate modules that perform distinct functions, including encryption and self-protection. It can operate on different language versions of Windows, increasing its global reach. Certain routines allow it to execute automatically after system restart, ensuring continued activity on infected machines.

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	**Vhash:** 036046656d1570a8z3631lz1fz **Imphash:** 68f013d7437aa653a8a98a05807afeb1 **Rich Pe header Hash:** 417a06d07f984f3bce5cd06546c98842 **TLSH:** T173F533F4E221B7ACF2550EF64855C59B6A9724B2EBEF1E26DA8001A70D 44F7F8FC0491 **SSDEEP:** 98304:QqPoBhz1aRxcSUDk36SAEdhvxWa9P593R8yAVp2g3x:QqPe1Cxcxk3Z AEUadzR8yc4gB **section MD5 hash:** .text 920e964050a1a5dd60dd00083fd541a2 .rdata 2c42611802d585e6eed68595876d1a15 .data 83506e37bd8b50cacabd480f8eb3849b .rscr f99ce7dc94308f0a149a19e022e4c316
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%2FS 0366-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/ 115p7UMMngoj1pMvkpHijcRdfJNXj6LrLn Wallet2:https://www.blockchain.com/explorer/addresses/btc/12t9YDPgwueZ9 NyMgw519p7AA8isjr6SMw Wallet3:https://www.blockchain.com/explorer/addresses/btc/13AM4VW2dhxY gXeQepoHkHSQuy6NgaEb94

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@2ol7
0xf440:$b3: 115p7UMMngoj1pMvkpHijcRdfJNXj6LrLn
0xf464:$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:$x5: WNcry@2ol7
0xf4fc:$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 also identified several strings that appear to correspond to 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 analyses 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 .xltm .xltx .xlc .xlm .xlt .xlw .xlsb .xlsm .xlsx .xls .dotx .dotm .dot .docm .docb .docx .doc

(...)

115p7UMMngoj1pMvkpHijcRdfJNXj6LrLn
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.

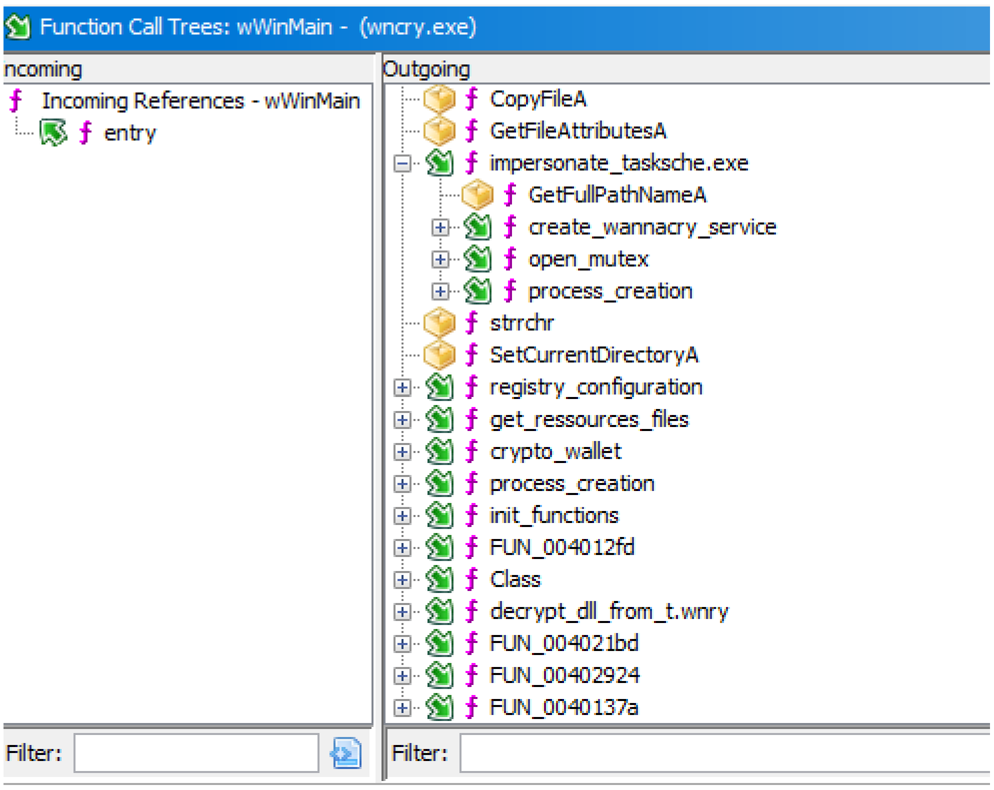
It is noteworthy to examine the system commands executed by the malware. The command *“cmd.exe /c "%s"” indicates that the malware executes a command and immediately close the windows command prompt. The command “icacls . /grant Everyone:F /T /C /Q” silently grants full control permissions to everyone 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 it's 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" throught "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.

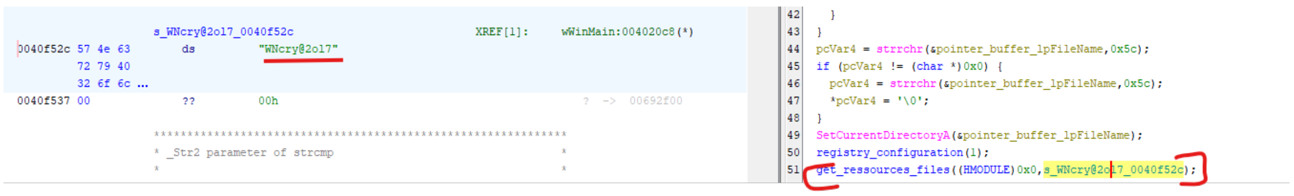
Wncry.exe: Ghidra - Function Call Tree



```
Function Call Trees: wWinMain -  (wncry.exe)

ncoming
ƒ Incoming References - wWinMain
  ƒ entry

Outgoing
ƒ CopyFileA
ƒ GetFileAttributesA
ƒ impersonate_tasksche.exe
    ƒ GetFullPathNameA
    ƒ create_wannacry_service
    ƒ open_mutex
    ƒ process_creation
ƒ strrchr
ƒ SetCurrentDirectoryA
ƒ registry_configuration
ƒ get_ressources_files
ƒ crypto_wallet
ƒ process_creation
ƒ init_functions
ƒ FUN_004012fd
ƒ Class
ƒ decrypt_dll_from_t.wnry
ƒ FUN_004021bd
ƒ FUN_00402924
ƒ FUN_0040137a

Filter:                    Filter:
```

The "get_ressources_file" functions take as an argument an hardcoded password: "*WNcry@2ol7*" and look for a file named *"2058"* with the *FindResourceA()* function. The malware use this password to unzip it's compressed file.

Wncry.exe: Ghidra – get_resources_file function

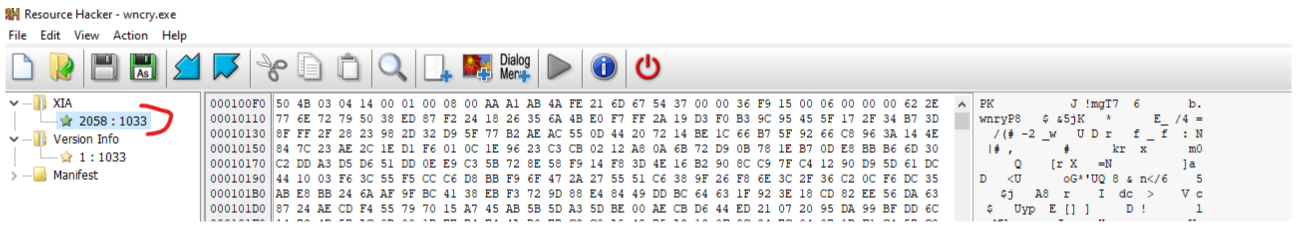


```
                      s_WNcry@2ol7_0040f52c                   XREF[1]:     wWinMain:004020c8(*)
0040f52c 57 4e 63        ds          "WNcry@2ol7"
         72 79 40
         32 6f 6c ...
0040f537 00              ??          00h                                     ?  ->  00692f00

                      ***************************************************************
                      * _Str2 parameter of strcmp                                   *
                      *                                                             *
```

```
    }
  }
  pcVar4 = strrchr(&pointer_buffer_lpFileName,0x5c);
  if (pcVar4 != (char *)0x0) {
    pcVar4 = strrchr(&pointer_buffer_lpFileName,0x5c);
    *pcVar4 = '\0';
  }
  SetCurrentDirectoryA(&pointer_buffer_lpFileName);
  registry_configuration(1);
  get_ressources_files((HMODULE)0x0,s_WNcry@2ol7_0040f52c);
```

We can manually extract this resource using *ResourceHacker:*

Wncry.exe: ResourceHacker

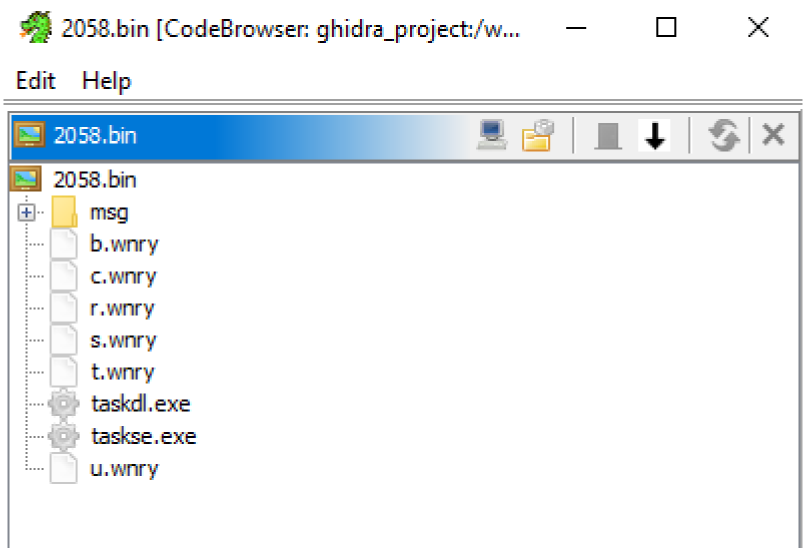


```
000100F0 50 4B 03 04 14 00 01 00 08 00 AA A1 AB 4A FE 21 6D 67 54 37 00 00 36 F9 15 00 06 00 00 00 62 2E   PK           J !mgT7  6       b.
00010110 77 6E 72 79 50 38 ED 87 F2 24 18 26 35 6A 4B E0 F7 FF 2A 19 D3 F0 B3 9C 95 45 5F 17 2F 34 B7 3D   wnryP8  $ &5jK   *      E_ /4 =
00010130 8F FF 2F 28 23 98 2D 32 D9 5F 77 B2 AE AC 55 0D 44 20 72 14 BE 1C 66 B7 5F 92 66 C8 96 3A 14 4E     /(# -2 _w   U D r   f _ f  : N
00010150 84 7C 23 AE 2C 1E D1 F6 01 0C 1E 96 23 C3 CB 02 12 A8 0A 6B 72 D9 0B 78 1E B7 0D E8 BB B6 6D 30    |# ,      #      kr  x      m0
00010170 C2 DD A3 D5 D6 51 DD 0E E9 C3 5B 72 8E 58 F9 14 F8 3D 4E 16 B2 90 8C C9 7F C4 12 90 D9 5D 61 DC        Q    [r X   =N          ]a
00010190 44 10 03 F6 3C 55 F5 CC C6 D8 BB F9 6F 47 2A 27 55 51 C6 38 9F 26 F8 6E 3C 2F 36 C2 0C F6 DC 35   D   <U      oG*'UQ 8 & n</6    5
000101B0 AB E8 BB 24 6A AF 9F BC 41 38 EB F3 72 9D 88 E4 84 49 DD BC 64 63 1F 92 3E 18 CD 82 EE 56 DA 63      $j   A8  r    I  dc  >    V c
000101D0 87 24 AE CD F4 55 79 70 15 A7 45 AB 5B 5D A3 5D BE 00 AE CB D6 44 ED 21 07 20 95 DA 99 BF DD 6C    $   Uyp  E [] ]     D !      l
```

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

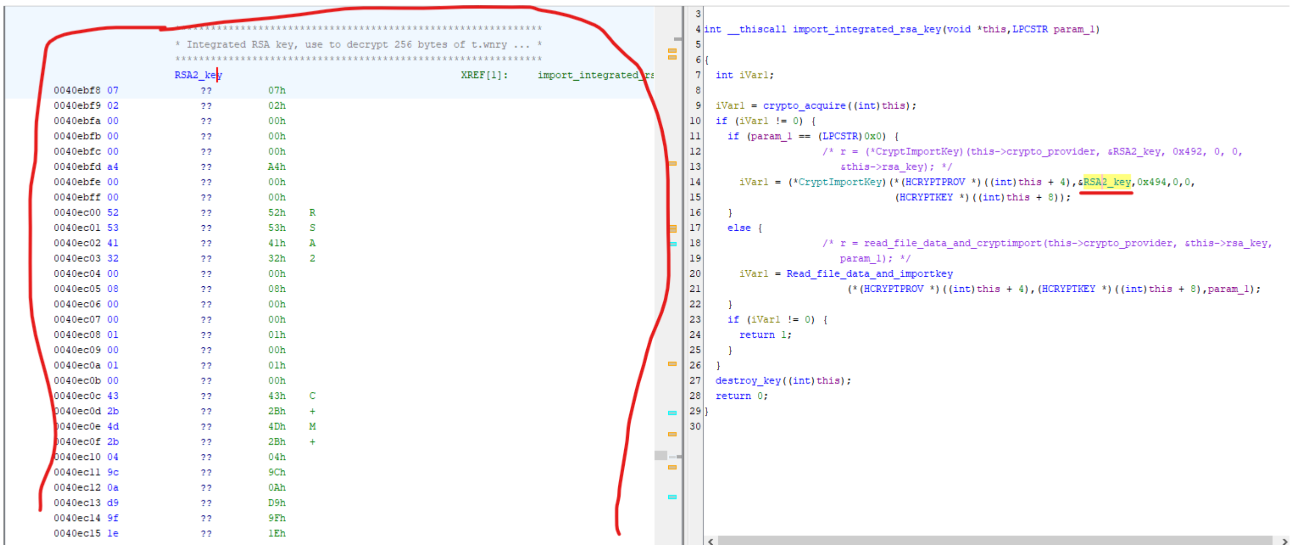
- b.wnry: bitmap image used as a wallpaper with the demand of a ransom.
- c.wnry: tor Command&Control addresses
- r.wnry: ransom message
- s.wnry: tor software
- u.wnry: decryptor module
- t.wnry: Encrypted DLL
- taskdl.exe: temporary file cleanup program
- taskse.exe: display decryptor windows to RDP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

Wncry.exe: Ghidra - decrypt_rasa_key

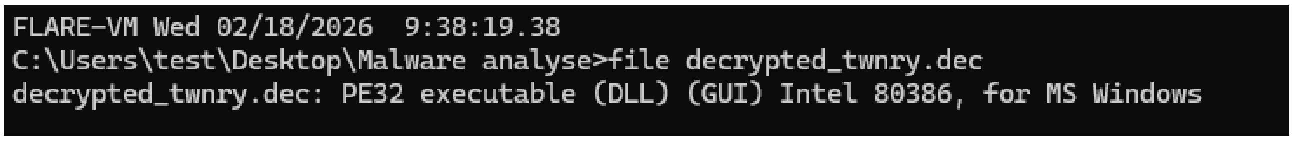


```
                     ****************************************************************
                     * Integrated RSA key, use to decrypt 256 bytes of t.wnry ... *
                     ****************************************************************
                     RSA2_key                                  XREF[1]:     import_integrated_rs
0040ebf8 07              ??          07h
0040ebf9 02              ??          02h
0040ebfa 00              ??          00h
0040ebfb 00              ??          00h
0040ebfc 00              ??          00h
0040ebfd a4              ??          A4h
0040ebfe 00              ??          00h
0040ebff 00              ??          00h
0040ec00 52              ??          52h    R
0040ec01 53              ??          53h    S
0040ec02 41              ??          41h    A
0040ec03 32              ??          32h    2
0040ec04 00              ??          00h
0040ec05 08              ??          08h
0040ec06 00              ??          00h
0040ec07 00              ??          00h
0040ec08 01              ??          01h
0040ec09 00              ??          00h
0040ec0a 01              ??          01h
0040ec0b 00              ??          00h
0040ec0c 43              ??          43h    C
0040ec0d 2b              ??          2Bh    +
0040ec0e 4d              ??          4Dh    M
0040ec0f 2b              ??          2Bh    +
0040ec10 04              ??          04h
0040ec11 9c              ??          9Ch
0040ec12 0a              ??          0Ah
0040ec13 d9              ??          D9h
0040ec14 9f              ??          9Fh
0040ec15 1e              ??          1Eh
```

```c

int __thiscall import_integrated_rsa_key(void *this,LPCSTR param_1)

{
  int iVar1;

  iVar1 = crypto_acquire((int)this);
  if (iVar1 != 0) {
    if (param_1 == (LPCSTR)0x0) {
                    /* r = (*CryptImportKey)(this->crypto_provider, &RSA2_key, 0x492, 0, 0,
                       &this->rsa_key); */
      iVar1 = (*CryptImportKey)(*(HCRYPTPROV *)((int)this + 4),&RSA2_key,0x494,0,0,
                                (HCRYPTKEY *)((int)this + 8));
    }
    else {
                    /* r = read_file_data_and_cryptimport(this->crypto_provider, &this->rsa_key,
                       param_1); */
      iVar1 = Read_file_data_and_importkey
                        (*(HCRYPTPROV *)((int)this + 4),(HCRYPTKEY *)((int)this + 8),param_1);
    }
    if (iVar1 != 0) {
      return 1;
    }
  }
  destroy_key((int)this);
  return 0;
}
```

Therefore, with a custom script that can be found at https://github.com/NoxCaellum/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.

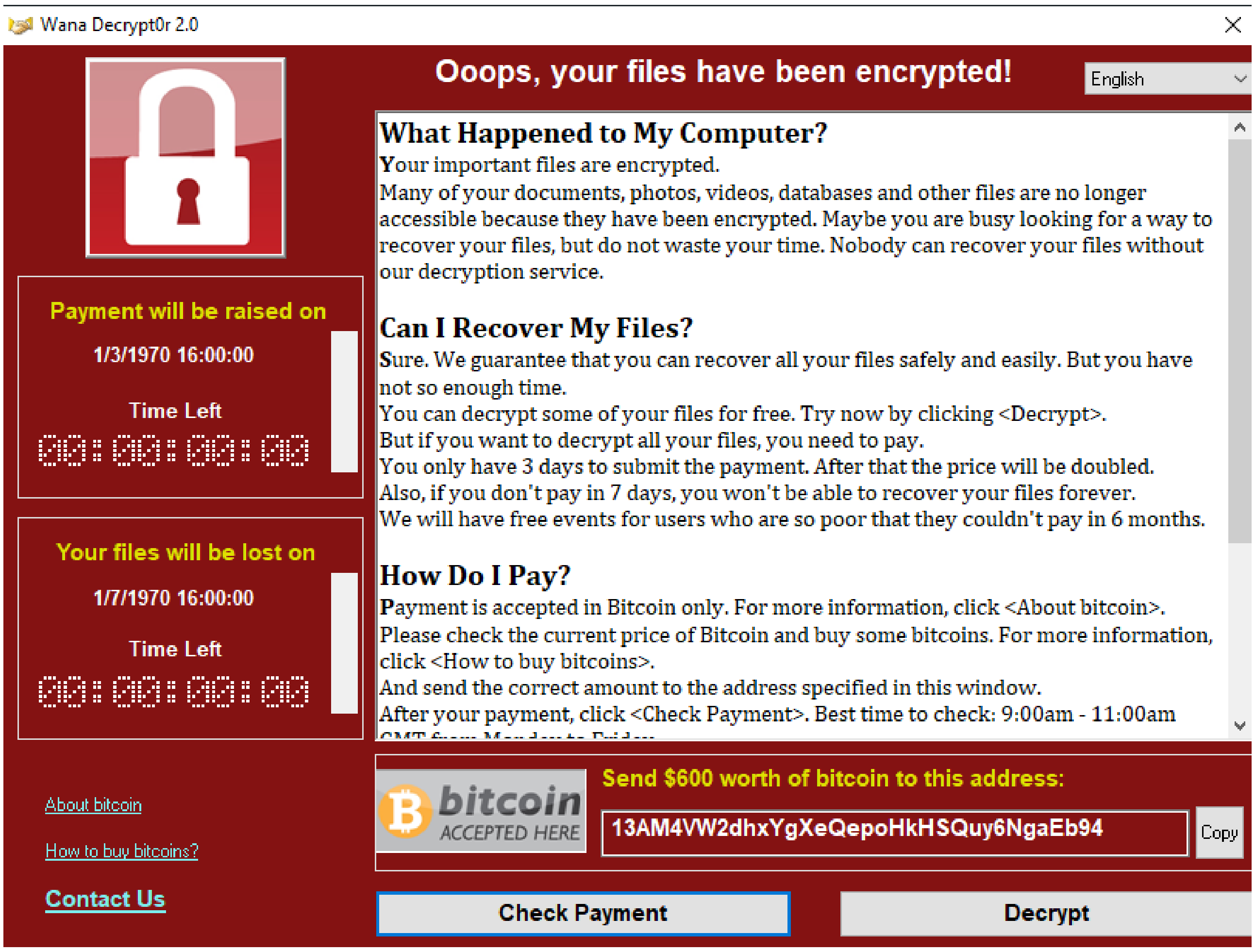
V. Dynamic Analysis

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



Ooops, your files have been encrypted!

Payment will be raised on

1/3/1970 16:00:00

Time Left

00:00:00:00

Your files will be lost on

1/7/1970 16:00:00

Time Left

00:00:00:00

What Happened to My Computer?

Your important files are encrypted.

Many of your documents, photos, videos, databases and other files are no longer accessible because they have been encrypted. Maybe you are busy looking for a way to recover your files, but do not waste your time. Nobody can recover your files without our decryption service.

Can I Recover My Files?

Sure. We guarantee that you can recover all your files safely and easily. But you have not so enough time.

You can decrypt some of your files for free. Try now by clicking <Decrypt>.

But if you want to decrypt all your files, you need to pay.

You only have 3 days to submit the payment. After that the price will be doubled.

Also, if you don't pay in 7 days, you won't be able to recover your files forever.

We will have free events for users who are so poor that they couldn't pay in 6 months.

How Do I Pay?

Payment is accepted in Bitcoin only. For more information, click <About bitcoin>.

Please check the current price of Bitcoin and buy some bitcoins. For more information, click <How to buy bitcoins>.

And send the correct amount to the address specified in this window.

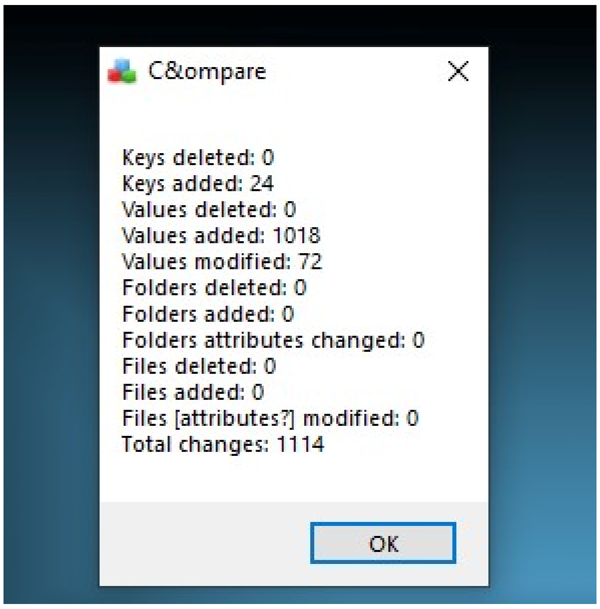
After your payment, click <Check Payment>. Best time to check: 9:00am - 11:00am

Send $600 worth of bitcoin to this address:

13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94

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:

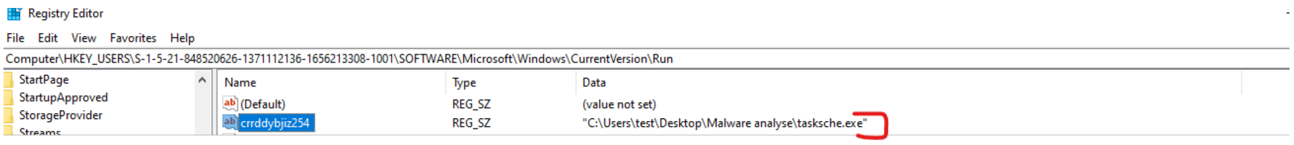


```
Keys deleted: 0
Keys added: 24
Values deleted: 0
Values added: 1018
Values modified: 72
Folders deleted: 0
Folders added: 0
Folders attributes changed: 0
Files deleted: 0
Files added: 0
Files [attributes?] modified: 0
Total changes: 1114
```

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\crrddybjiz254 "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.


Registry Editor

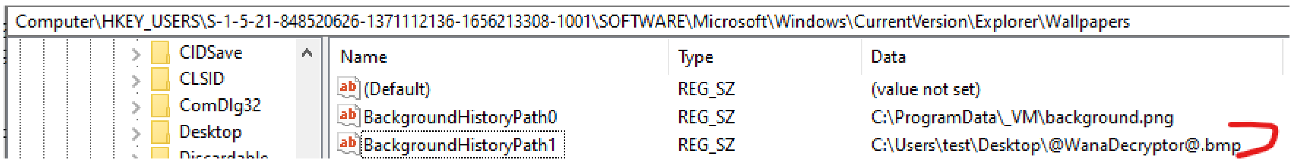
File Edit View Favorites Help

Computer\HKEY_USERS\S-1-5-21-848520626-1371112136-1656213308-1001\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

StartPage
StartupApproved
StorageProvider

Name	Type	Data
(Default)	REG_SZ	(value not set)
crrddybjiz254	REG_SZ	"C:\Users\test\Desktop\Malware analyse\tasksche.exe"

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.


Computer\HKEY_USERS\S-1-5-21-848520626-1371112136-1656213308-1001\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Wallpapers

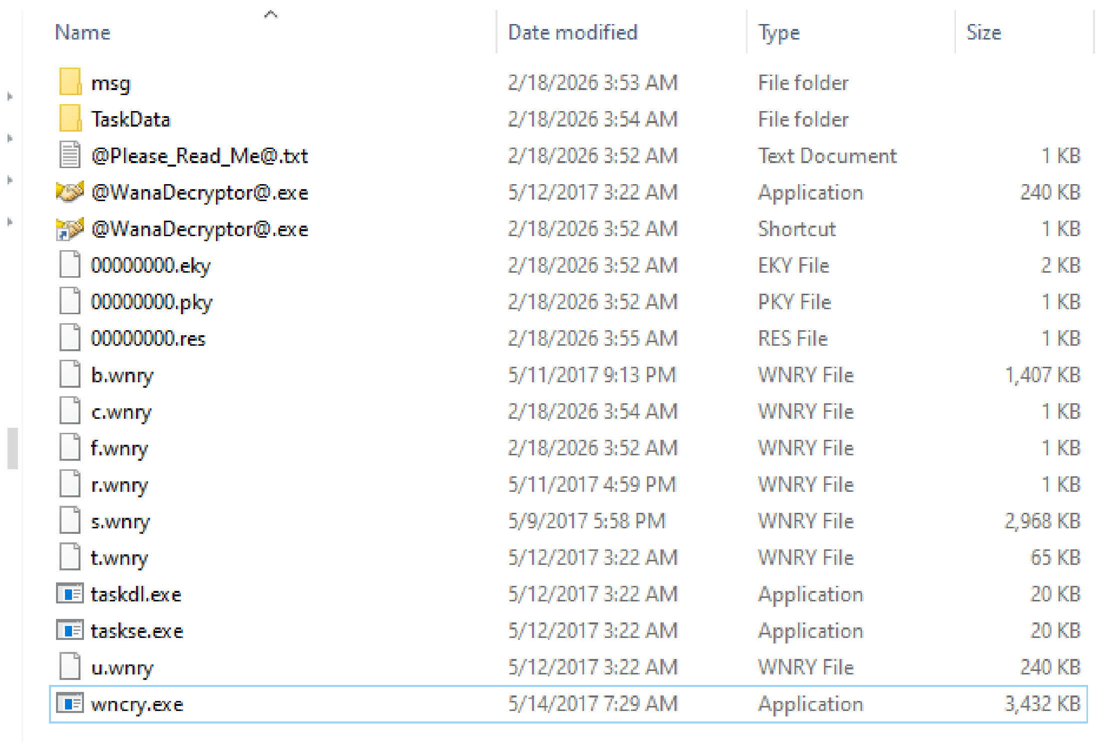
CIDSave
CLSID
ComDlg32
Desktop

Name	Type	Data
(Default)	REG_SZ	(value not set)
BackgroundHistoryPath0	REG_SZ	C:\ProgramData_VM\background.png
BackgroundHistoryPath1	REG_SZ	C:\Users\test\Desktop\@WanaDecryptor@.bmp

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.pky: 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 modyle of the ransomware decryptor (u.wnry)
- @Please_Read_Me@txt: Ransom demand text (r.wnry) [4]

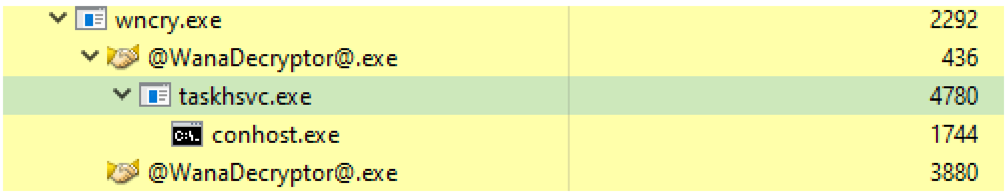
Wncry.exe: Dropped files after execution



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" wich 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
- cwwnhwhlz52maqm7.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.

Bibliography

1: Kasperky, What is WannaCry ransomware?,
2: Avast, What Is EternalBlue and Why Is the MS17-010 Exploit Still Relevant?, 2020
3: Cisco Talos, Player 3 Has Entered the Game: Say Hello to 'WannaCry', 2017
4: Counter Threat Unit Research Team, WCry Ransomware Analysis, 2017

Glossary

AES – Advanced Encryption Standard, a symmetric encryption algorithm used for encrypting files or data.

API functions – Application Programming Interface functions provided by the operating system or libraries, used by malware to interact with system resources.

CAPA – Capabilities for Automated Malware Analysis, a tool used in static analysis to identify functions and behaviors in a binary.

Cryptocurrency wallet – A digital storage of private keys used to hold or transfer cryptocurrency, sometimes targeted or referenced by ransomware.

DetectItEasy – A tool for identifying packers, compilers, and other metadata in executable files to assist in malware analysis.

DLL – Dynamic Link Library, a file containing executable code that can be loaded by Windows applications.

Dynamic Analysis – Observation of malware while it is executed in a controlled environment to understand its runtime behavior, including system changes, network activity, and process execution.

Ghidra – A reverse engineering tool and disassembler used to analyze binary code and understand malware functionality.

IoC – Indicator of Compromise, a piece of information (e.g., hash, IP address, file path) that can be used to detect malware infection or malicious activity.

Malware – Malicious software designed to disrupt, damage, or gain unauthorized access to computer systems.

MITRE ATT&CK – A knowledge base of adversary tactics, techniques, and procedures (TTPs), used to map and understand malware behavior.

PE – Portable Executable, the standard file format for executables, DLLs, and system files in Windows.

Ransomware – Malware that encrypts files on a victim's system and demands payment in exchange for decryption.

Registry – The Windows system database used to store configuration settings, which can be read or modified by malware.

Static Analysis – Examination of a binary or file without executing it, to study code structure, strings, patterns, and embedded resources.

SysInformer – A monitoring tool used to observe running processes, services, and system activity during dynamic analysis.

WannaCry – A ransomware that targets Windows systems, encrypts files, and spreads via SMB exploits.

Onion domain – A type of domain used on the Tor network, often referenced by ransomware for communication or payment sites.

YARA – A tool used to create rules for identifying malware samples based on patterns, strings, or binary structures.