Practical Malware Analysis & Triage

Malware Analysis Report

Bossfight-Wannacry.exe

Oct 2022 | maT | v1.0

Table of Contents

Bossfight-Wannacry.exe
Oct 2022
v1.0

Executive Summary

Filename	Ransomware.wannacry.exe
SHA256 hash	24d004a104d4d54034dbcffc2a4b19a11f39008a575aa614ea04703480b1022c
VT detection	68/71 engines

The artifact collected is a WannaCry ransomware specimen. It is an example of ransomware cryptoworm used by cybercriminals to encrypt data and extort money.

This malware is composed of multiple components, such as:

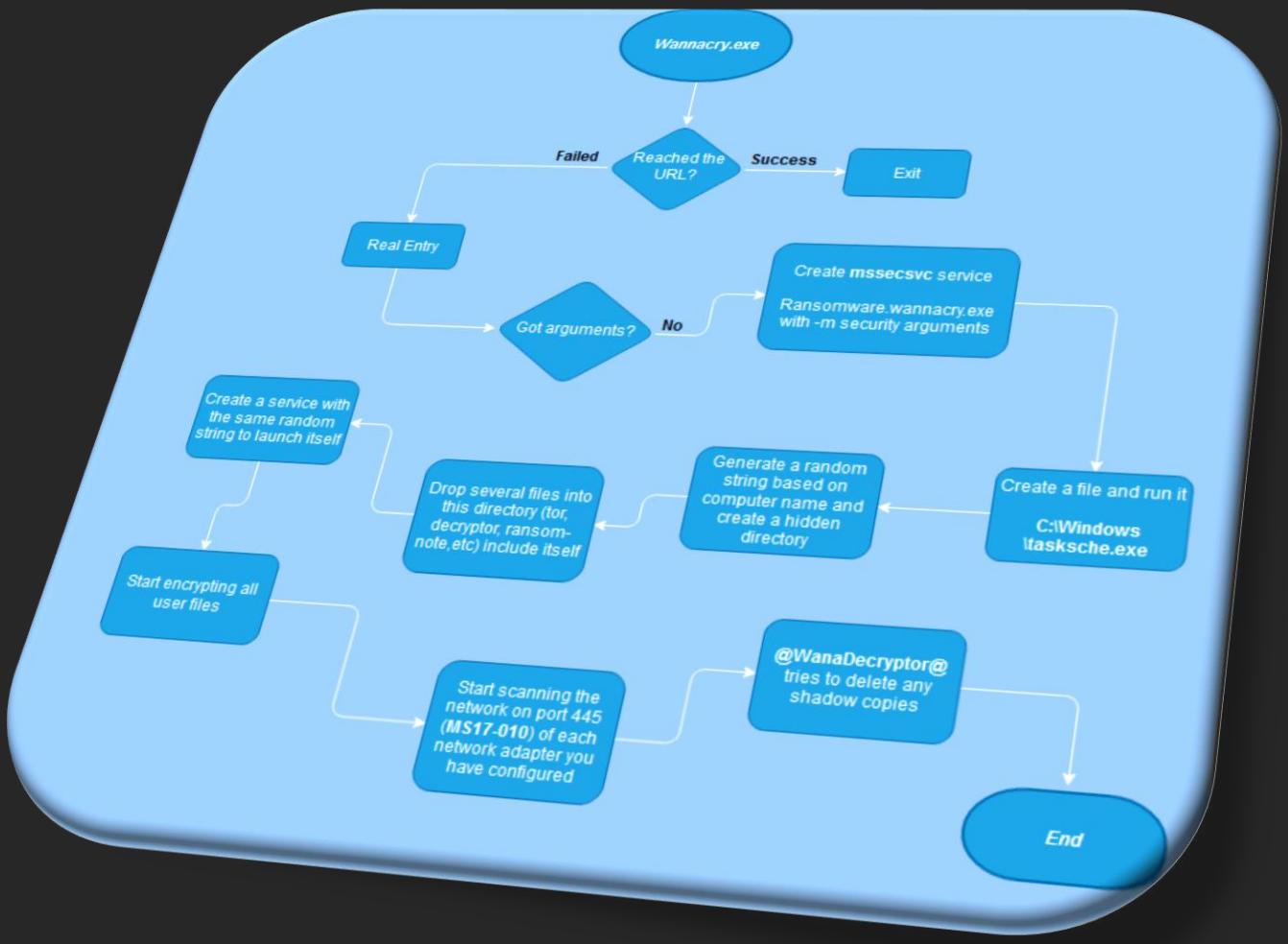
a) An initial dropper contains the encryptor as an embedded resource;
b) The encryptor piece contains a decryption application, a zip file (protected by password) consisting of a Tor browser folder, various files with configuration details and encryption keys.

Once executed, and meeting all the requirements described in the **Dynamic Analysis** section, all the user files are encrypted, a persistence mechanism is set, a scan over the network on port 445 is performed (in order to propagate itself abusing a known vulnerability in the SMB protocol – MS17-010) and the decryptor component tries to delete any shadow copies.

YARA signature has been created and the malware sample and hashes have been submitted to VirusTotal for further examination.

PRACTICAL MALWARE ANALYSIS & TRIAGE

Execution Flow



Bossfight-Wannacry.exe
Oct 2022
v1.0

Malware Composition

The WannaCry ransomware consists of several files, but the main ones are:

File Name	SHA256 Hash
Ransomware.wannacry.exe	24D004A104D4D54034DBCFFC2A4B19A11F39008A575AA614EA04703480B1022C
tasksche.exe	ED01EBFBC9EB5BBEA545AF4D01BF5F1071661840480439C6E5BABE8E080E41AA
@WanaDecryptor@.exe	B9C5D4339809E0AD9A00D4D3DD26FDF44A32819A54ABF846BB9B560D81391C25

Ransomware.wannacry.exe

The initial dropper which start all the infection chain and perform the scanning over the internal network

tasksche.exe

The encryptor and main artifact to achieve the malware goal (sequestrate the user files)

@WanaDecryptor@.exe

The decryptor, responsible to scare the user with the ransom-note and to seek & destroy any possibility of file recovery

Static Analysis

Using the **FLOSS** tool, we were able to extract some interesting strings from the sample:

```
C:\Users\root\Desktop
λ FLOSS.exe Ransomware.wannacry.exe -n 8 > strings_wanna.txt
```

Fig 1: FLOSS searching for strings with the length >= 8 chars

- mssecsvc.exe
- tasksche.exe
- taskdl.exe
- cmd.exe /c "%s"
- taskse.exe
- diskpart.exe
- lhdfrgui.exe
- icacls . /grant Everyone:F /T /C /Q
- WNcry@2ol7
- %s -m security
- C:\%s\qeriuwjhrf
- hxxp://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea[.]com

The **PEstudio** was able to show us even more sneaky stuff, increasing our suspicion of this sample

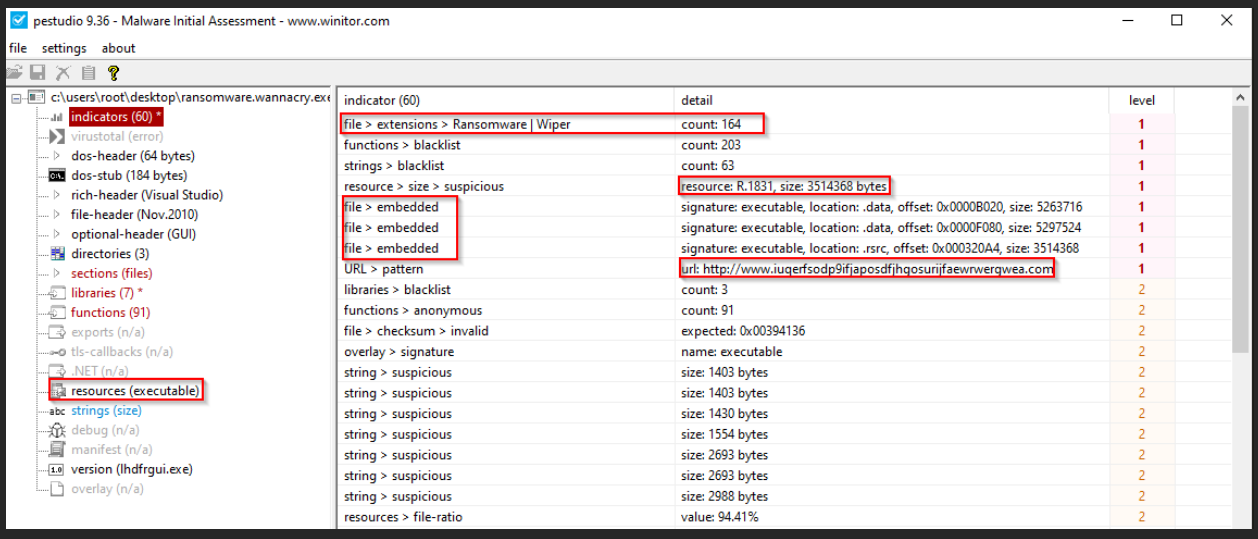


Fig 2: PEstudio showing the main suspicious indicator of the file

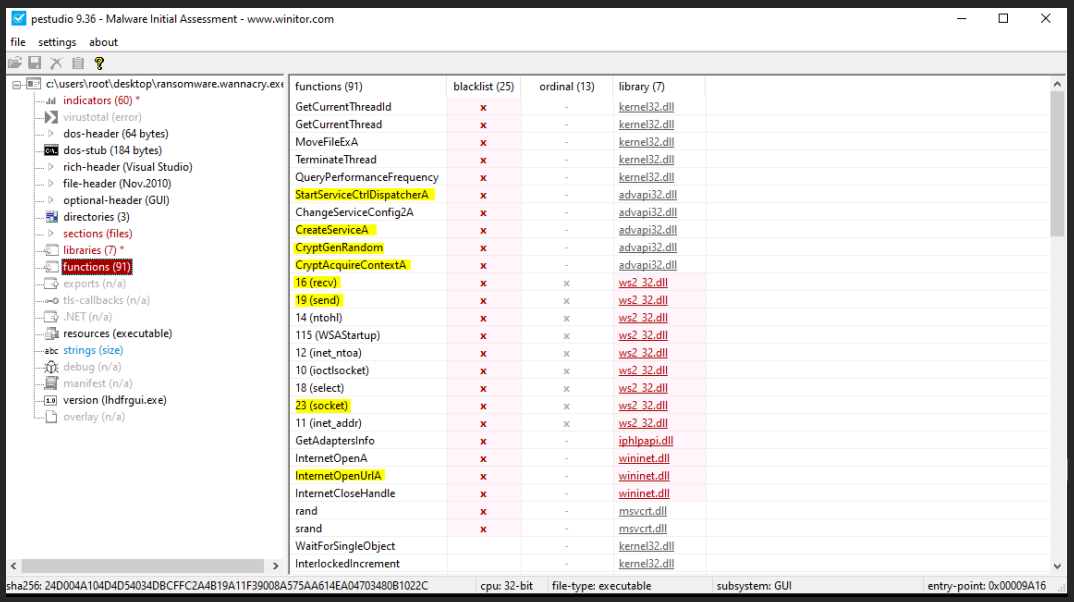


Fig 3: PEstudio showing some suspicious DLL functions used by this binary

In order to take a close look at this suspicious resource embedded into our sample, the tool **ResourcesExtract** (Nimsoft) was pretty useful

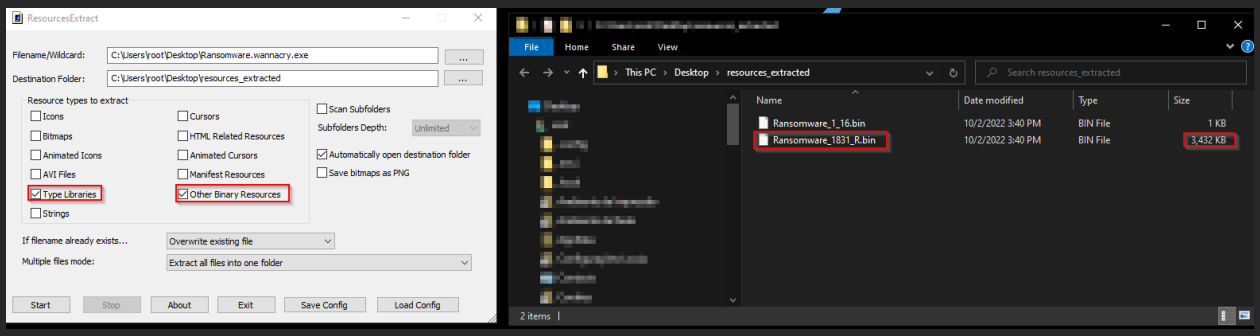


Fig 4: ResourceExtract collecting the suspicious embedded resource

```
C:\Users\root\Desktop\resources_extracted
λ file *
Ransomware_1831_R.bin: PE32 executable (GUI) Intel 80386, for MS Windows
Ransomware_1_16.bin:   data

C:\Users\root\Desktop\resources_extracted
λ sha256sum.exe *
ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa *Ransomware_1831_R.bin
2f3fc51546ada848dfc8e775554c0de3689d6fae7ba4bf3d40e3c8dec68b277b *Ransomware_1_16.bin
```

We've just found out the encryptor tasksche.exe

Fig 5: Quickly assessment to understand what we've collected

So now, it's time to call on our friends **FLOSS** and **PEstudio** again to help us to get to know this new PE better

Strings got with FLOSS:

- cmd.exe /c "%s"
- Global\MsWinZonesCacheCounterMutexA
- icacls . /grant Everyone:F /T /C /Q
- attrib +h .
- WNcry@2ol7
- msg/m_portuguese.wnry (and several others for other languages)
- taskdl.exe
- taskse.exe
- WanaCrypt0r (the name of our decryptor)
- %s\ProgramData
- diskpart.exe

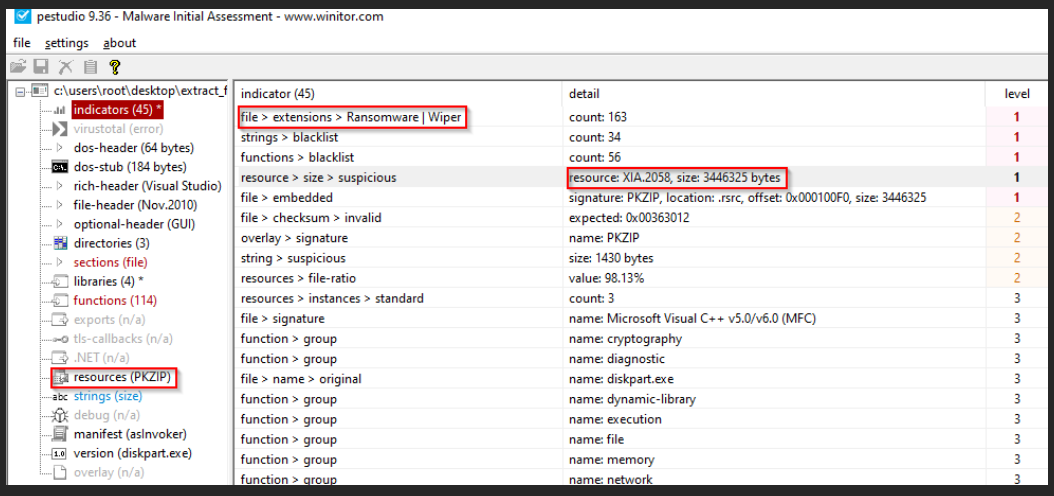


Fig 6: PEstudio analyzing our new PE found (Ransomware_1831_R.bin)

As we can see, this file also contains a suspicious resource, but in this particular case a **ZIP** file (which one we certainly will take a look)

In the image below, there are several other files inside our zip file, but unfortunately it requires a password to unzip them (which we haven't found yet)

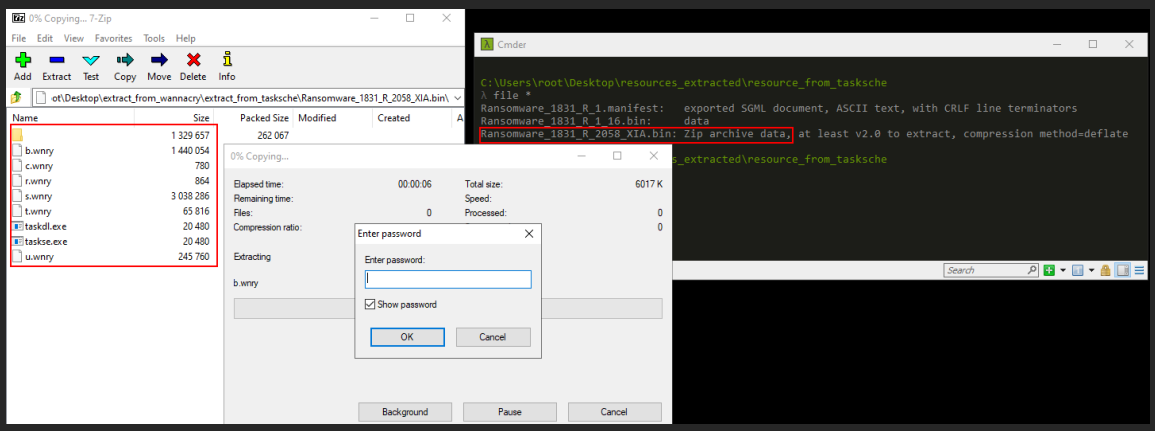


Fig 7: The ZIP file asking for the password

The decryptor (@WanaDecryptor@.exe) has been found only during the dynamic analysis, so after we got it we performed the same assessment procedure already done with the dropper and the encryptor

Strings got with FLOSS:

- /c vssadmin delete shadows /all /quiet & wmic shadowcopy delete & bcdedit /set {default} bootstatuspolicy ignoreallfailures & bcdedit /set {default} recoveryenabled no & wbadmin delete catalog -quiet
- hxxp://www.btcfrog[.]com/qr/bitcoinPNG.php?address=%s
- Ooops, your files have been encrypted!
- \Program Files (x86)
- \Local Settings\Temp
- \AppData\Local\Temp
- @WanaDecryptor@.exe.lnk
- @WanaDecryptor@.bmp
- Wana Decrypt0r 2.0
- WanaCrypt0r

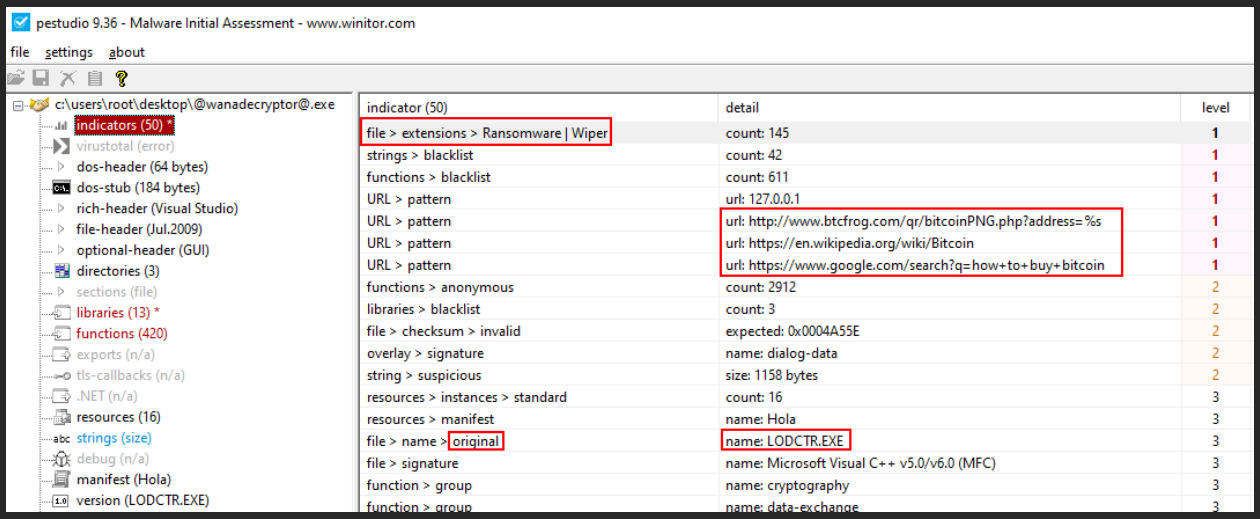


Fig 8: PEstudio assessing the decryptor

Dynamic Analysis

Before detonate our sample, we must be prepared to collect as much relevant information as possible. So, based on that, the following tools were always running previous any detonation:

- Wireshark
- Procmon
- TCPview
- Process Hacker

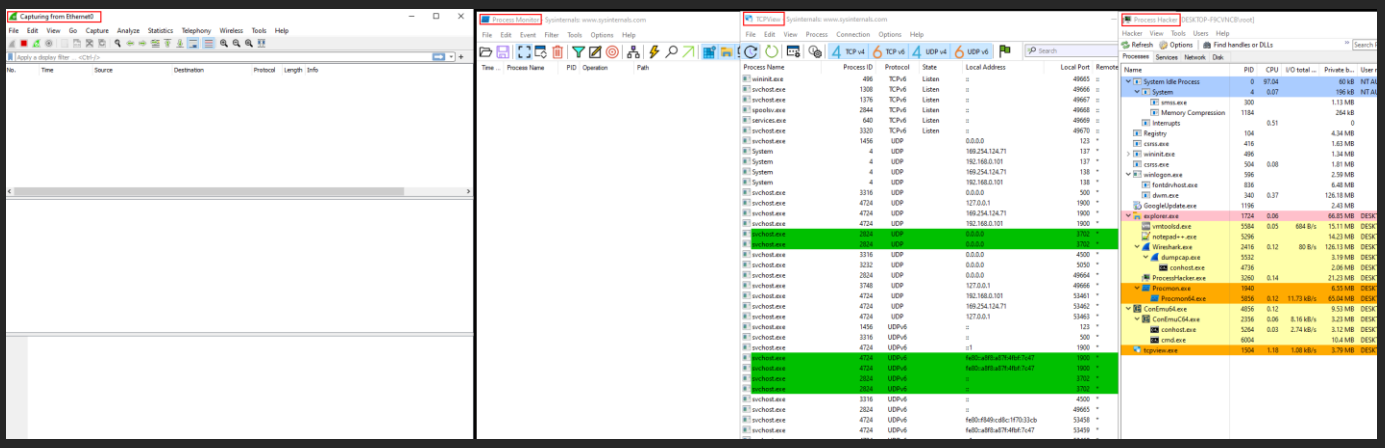


Fig 9: Preparing before detonation

After some sample detonations, we could identify the requirements for the malware executes its payload:

- Run it with **administrator** privileges
- The DNS/HTTP request to **www[.]iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com** must not be answered (killswitch mechanism)

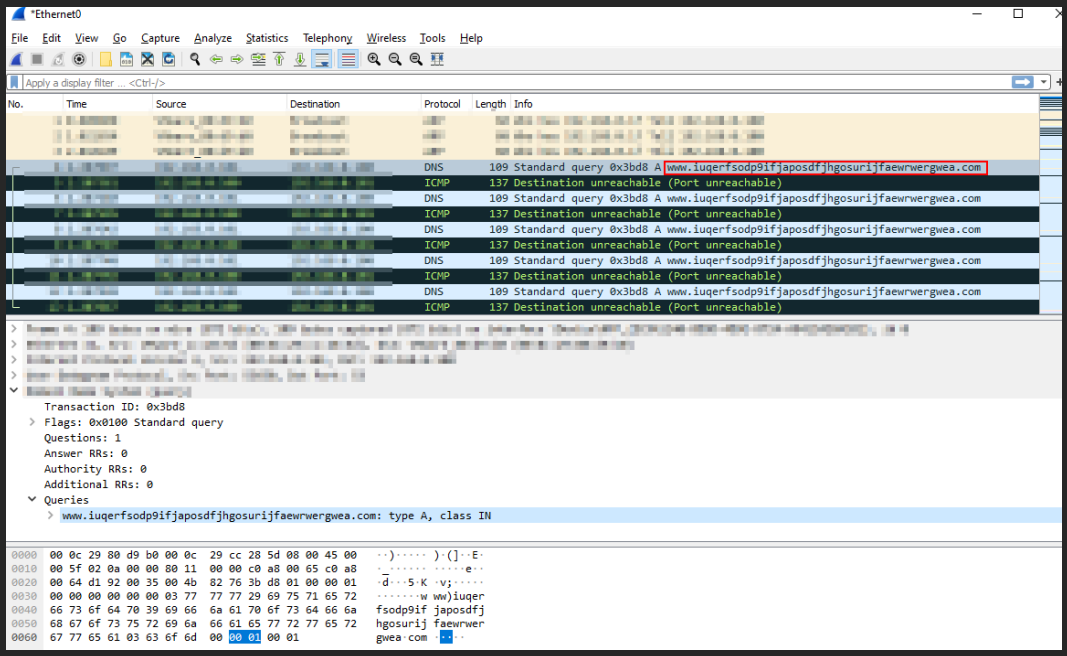


Fig 10: Killswitch mechanism

Once the requirements mentioned above were meet, we can observe the following behavior of our sample

1) The encryptor (**tasksche.exe**) being dropped in 2 location

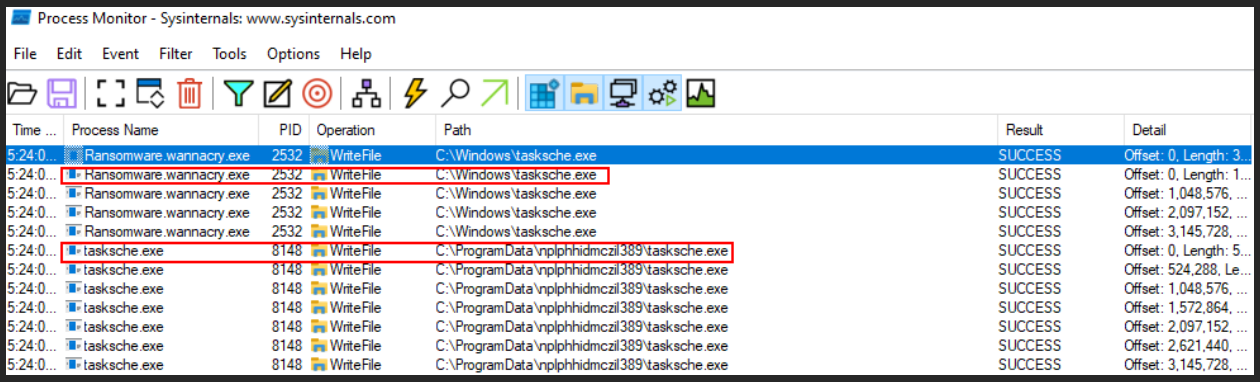


Fig 11: tasksche.exe being born

2) A scanning has been initiated towards the 2 networks related to my network adapters

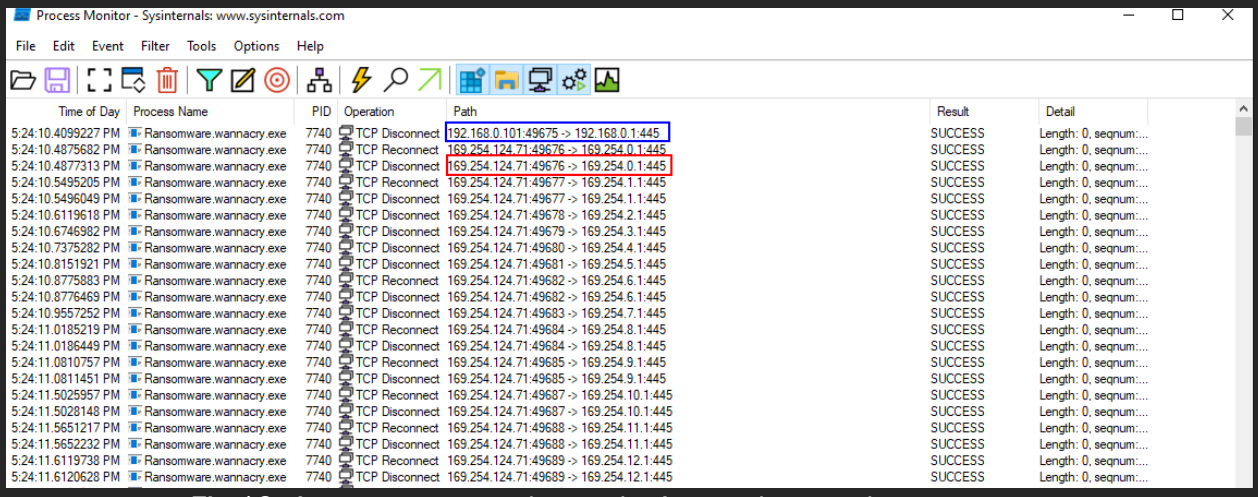


Fig 12: Attempt to spread over the internal network

3) Suspicious folder/files created by **tasksche.exe**

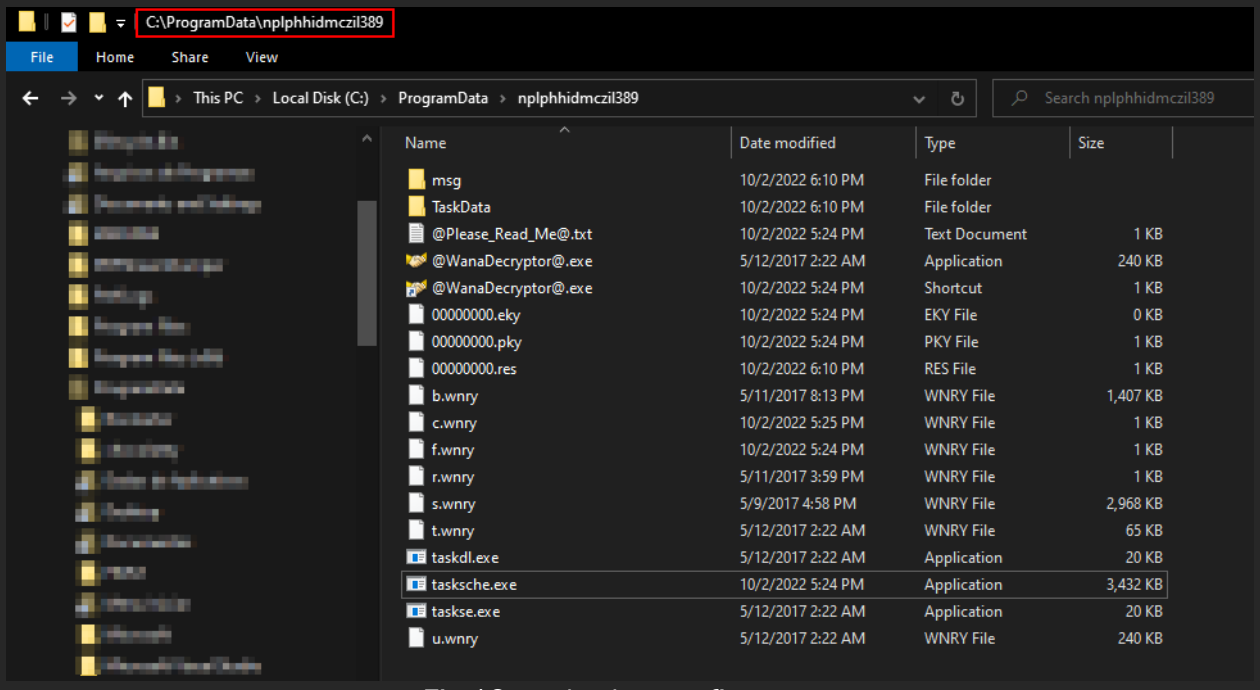


Fig 13: tasksche.exe first steps

4) The persistence mechanism being set

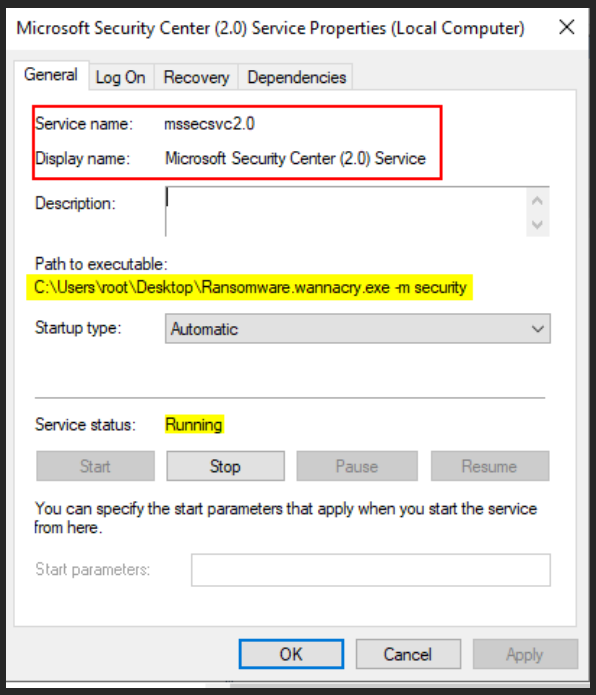


Fig 14: Service mssecsvc2.0 being created & running

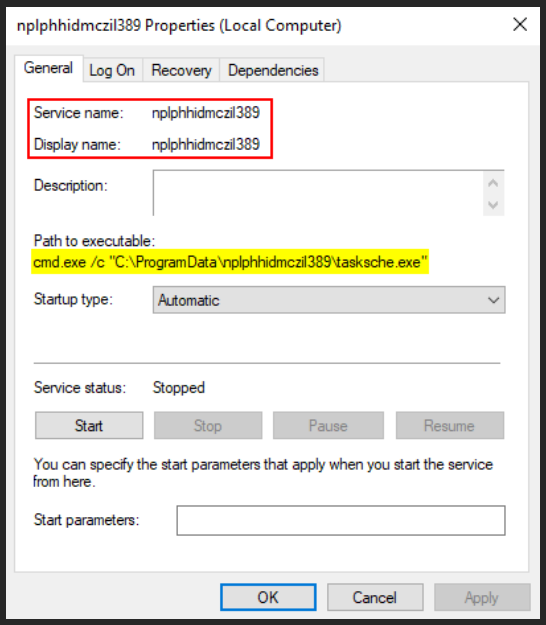


Fig 15: Service named with the same name as the previously mentioned folder

5) Then the files start to be encrypted

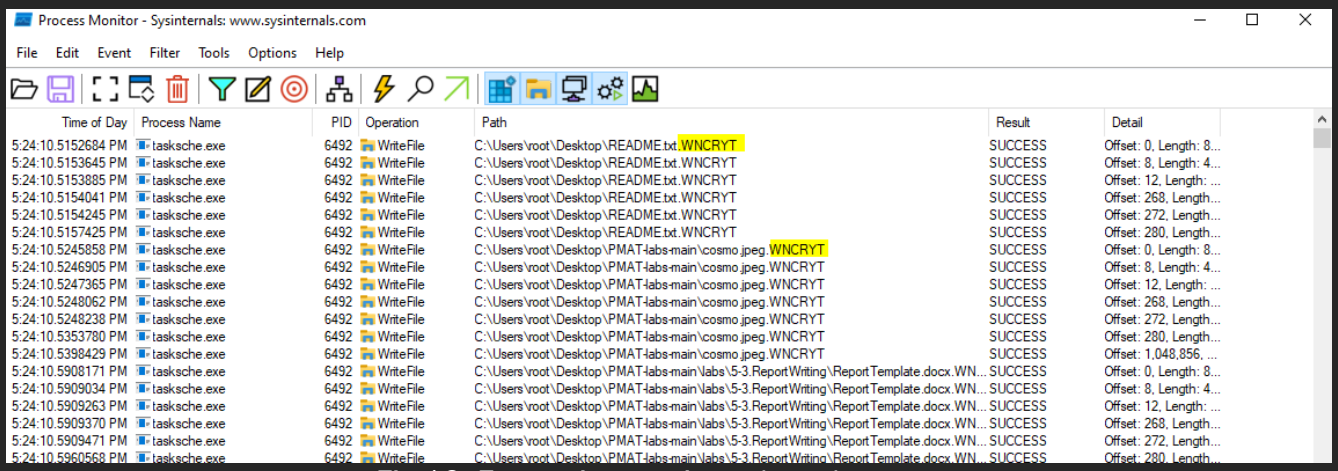


Fig 16: Encryption routine takes place

6) After the encryption procedure is completed, the decryptor (@WanaDecryptor@.exe) pop up to the user with the ransom-note and attempts to delete any Windows shadow copies using the following command (found with PEstudio during the static analysis step)

```
vssadmin delete shadows /all /quiet & wmic shadowcopy delete & bcdedit /set
{default} bootstatuspolicy ignoreallfailures & bcdedit /set {default}
recoveryenabled no & wbadmin delete catalog -quiet
```

Fig 17: Decryptor interface

Fig 18: Command line inside the @WanaDecryptor@.exe

A bit of Reverse Engineering

Leveraging the power of the **Cutter** tool, we are able to understand various steps seen during the dynamic analysis, such as

The kill switch mechanism (via Graph view)

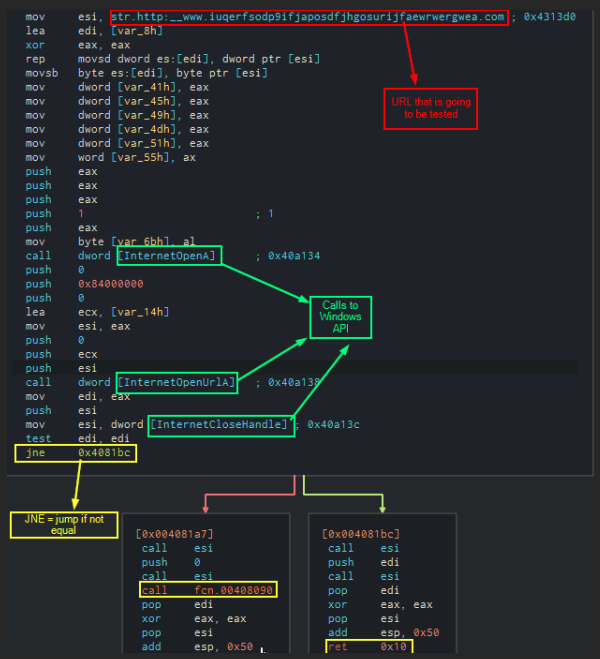


Fig 19: First lines of the main function

In a nutshell, the killswitch mechanism works with the following logic

if ZF* == 0 (happens when the HTTP request is performed with success and there is a value set on the register EDI) then **exit** the program

if ZF* == 1 (happens when the EDI = 0, because the result of bitwise AND operation between 0 AND 0 = 0 and then the ZF is set to 1) so the program continues and run the "real entry" located within the function **fcn.00408090**

** ZF means Zero Flag*

Once inside the function **fcn.00408090**, the number of arguments is checked and then we are led to the function **fcn.00407c40,** which is responsible for creating & running the service called **mssecsvc2.0** (Decompiler Ghidra view)

```
undefined4 fcn.00407c40(void)
{
    code *pcVar1;
    int32_t iVar2;
    int32_t iVar3;
    char *pcStack272;
    undefined4 uStack268;
    undefined auStack260 [260];

    uStack268 = 0x70f760;
    pcStack272 = "%s -m security";
    (*_sprintf)(auStack260);
    uStack268 = 0xf003f;
    pcStack272 = (char *)0x0;
    iVar2 = (*_OpenSCManagerA)(0);
    if (iVar2 != 0) {
        iVar3 = (*_CreateServiceA)(iVar2, "mssecsvc2.0", "Microsoft Security Center (2.0) Service", 0xf01ff, 0x10, 2, 1
                                   , &pcStack272, 0, 0, 0, 0, 0);
        pcVar1 = _CloseServiceHandle;
        if (iVar3 != 0) {
            (*_StartServiceA)(iVar3, 0, 0);
            (*pcVar1)(iVar3);
        }
        (*pcVar1)(iVar2);
        return 0;
    }
    return 0;
}
```

Fig 20: The persistence being set

And then along with the **fcn.00407ce0** the encryptor is prepared and dropped on the victim machine (Decompiler jsdec)

```
eax = &lpExistingFileName;
void (*esi)(void, void, void, char*) (eax, bl, bl, "tasksche.exe");
ecx = &lpNewFileName;
void (*esi)(void, char*, char*) (ecx, 'C:\\%s\\qeriuwjhrf", "WINDOWS");
edx = &lpNewFileName;
eax = &lpExistingFileName;
uint32_t (*MoveFileExA)(void, void, void) (eax, edx, 1);
ecx = &var_7ch;
eax = uint32_t (*0x431458)(void, void, void, void, void, void, void) (ecx, 0x40000000, ebx, ebx, 2, 4, ebx);
esi = eax;
```

Fig 21: The encryptor being cooked & delivered

Indicators of Compromise

The list of some IOCs can be found below.

Network Indicators

IOC	Description
hxxp://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea[.]com	Killswitch URL

Host-based Indicators

IOC	Description
24D004A104D4D54034DBCFFC2A4B19A11F39008A575AA614EA04703480B1022C	Ransomware.wannacry.exe
ED01EBFBC9EB5BBEA545AF4D01BF5F1071661840480439C6E5BABE8E080E41AA	tasksche.exe
B9C5D4339809E0AD9A00D4D3DD26FDF44A32819A54ABF846BB9B560D81391C25	@WanaDecryptor@.exe
E8406294C82D351EAFE539E881F5608E92AA99347CF47142A448AE69BF016D14	00000000.eky
BB025FE52E2C82E02BDD9CBE2A458CBEFF0D955911C566CB6744525FDE8C93BB	00000000.pky
E829A8FB718A2FC0CC6B0A484458B8DE8C1E0C838DB0323FB2025DBA8691F85B	00000000.res
4A25D98C121BB3BD5B54E0B6A5348F7B09966BFFEEC30776E5A731813F05D49E	@Please_Read_Me@.txt
C725AEB013F116C1477D8D767F3BAEF0CA8C71CA2DE3AC956AB5C7805FE1AC45	@WanaDecryptor@.exe.lnk
D5E0E8694DDC0548D8E6B87C83D50F4AB85C1DEBADB106D6A6A794C3E746F4FA	b.wnry
818CD840D99B8EC263FCCF7CC43798A207020A1F9A5F7F5711086B31A0F950C7	c.wnry
61AE893E866452CC9ED9FC12F9693D2732B21CA61748A4381157E93465F4FC54	f.wnry
402751FA49E0CB68FE052CB3DB87B05E71C1D950984D339940CF6B29409F2A7C	r.wnry
E18FDD912DFE5B45776E68D578C3AF3547886CF1353D7086C8BEE037436DFF4B	s.wnry
97EBCE49B14C46BEBC9EC2448D00E1E397123B256E2BE9EBA5140688E7BC0AE6	t.wnry
4A468603FDCB7A2EB5770705898CF9EF37AADE532A7964642ECD705A74794B79	taskdl.exe
2CA2D550E603D74DEDDA03156023135B38DA3630CB014E3D00B1263358C5F00D	taskse.exe
mssecsvc2.0	service name (persistence)
HKCU\SOFTWARE\WanaCrypt0r\wd	registry key

Rules & Signatures

YARA rule

```
rule WannaCry_Ransomware {

    meta:
        author = "maT"
        description = "Detects the WannaCry dropper + encryptor + decryptor"
        filetype = "PE"
        version = "1.0"
        reference = "https://academy.tcm-sec.com/courses/enrolled/1547503"
        hash_dropper =
"24D004A104D4D54034DBCFFC2A4B19A11F39008A575AA614EA04703480B1022C"
        hash_encryptor =
"ED01EBFBC9EB5BBEA545AF4D01BF5F1071661840480439C6E5BABE8E080E41AA"
        hash_decryptor =
"B9C5D4339809E0AD9A00D4D3DD26FDF44A32819A54ABF846BB9B560D81391C25"

    strings:
        $s1 = "icacls . /grant Everyone:F /T /C /Q" fullword ascii
        $s2 = "www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com" fullword ascii
        $s3 = "WNcry@2ol7" fullword ascii
        $s4 = "taskse.exe" fullword ascii
        $s5 = "tasksche.exe" fullword ascii
        $s6 = "taskdl.exe" fullword ascii
        $s7 = "msg/m_portuguese.wnry" fullword ascii
        $s8 = "Global\\MsWinZonesCacheCounterMutexA" fullword ascii
        $s9 = "vssadmin delete shadows" fullword ascii

        $s10 = "WanaCrypt0r" fullword wide
        $s11 = "Ooops, your files have been encrypted!" fullword wide
        $s12 = "@WanaDecryptor@.bmp" fullword wide

    condition:
        uint16(0) == 0x5A4D // MZ
        and 4 of ($s*)
        and filesize < 4MB
}
```