
# Malware Analysis of WannaCry Ransomware

# Table of Contents:

## Background

In May 2017, a new malware hit the world like never before. This malware was unfortunately called WannaCry by the malware creators themselves and, the malware was so ruthless in its operation that it could make one shed tears. Once in a computer, the malware had different tentacles like an Octopus. On one side it broadcasts itself in the network like a worm. On another, it encrypts all documents leaving them as a ".WNCRY" extension. There were so many parts to this malware. It was concluded that this malware had a financial impact of $4 billion according to Symantec.

This malware has within itself a kill switch designed by the malware authors themselves for whatsoever reason. Security researcher Marcus Hutchins found this kill switch and used it to prevent the spread of the malware. Once inside a computer, the malware tries to contact a particular domain containing random characters and if found stops.

# High – Level Technical Summary

| | |
|---|---|
| Sha256 | 24d004a104d4d54034dbcffc2a4b19a11f39008a575aa614ea0470 3480b1022c |

Once in a computer, the malware behaves like a worm. It calls out to all hosts in the network. This is how the malware propagates. When run, the malware creates a file in C:\Windows and a folder in the C:\ProgramData called ctcoksabd271. This folder contains all the tools Wannacry needs to perform its evil enterprises.

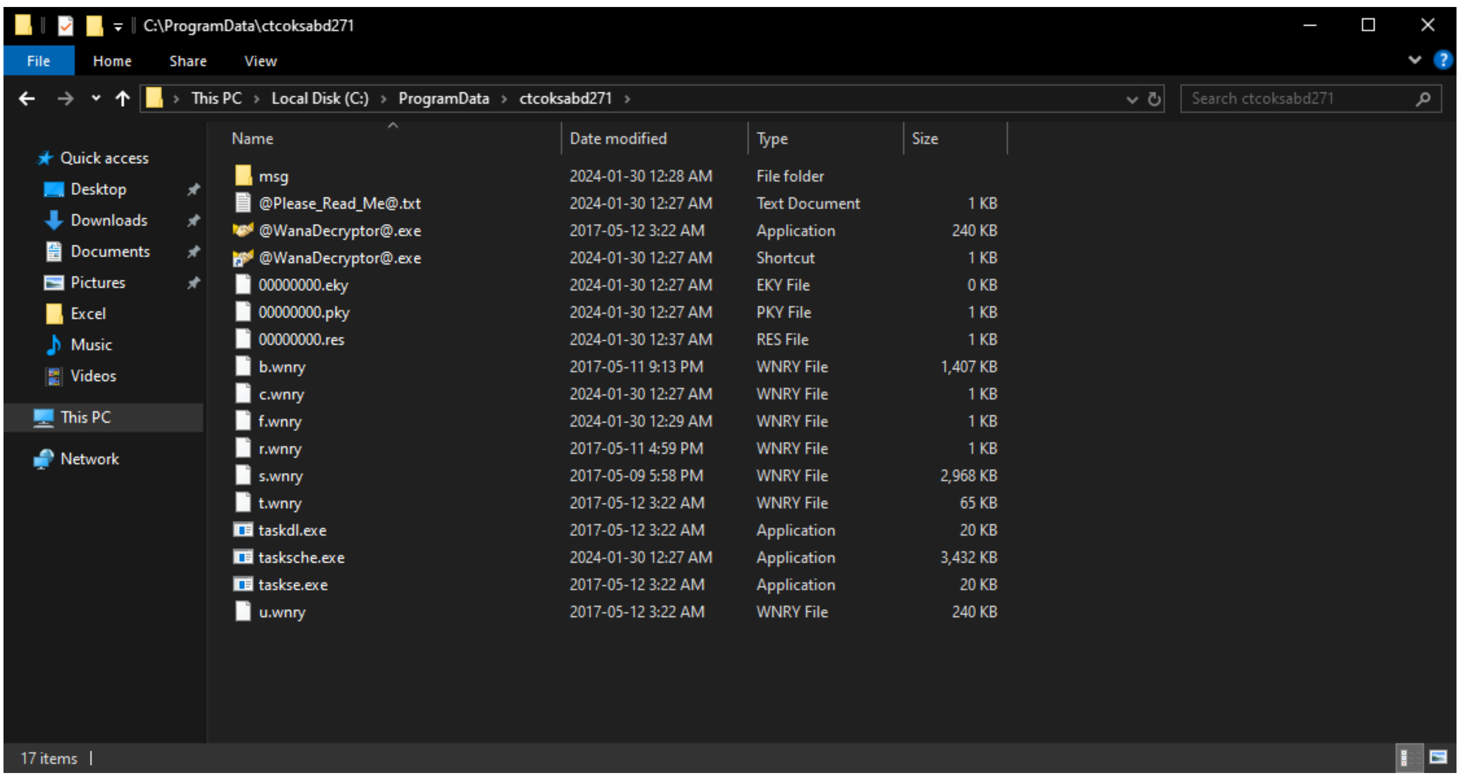



fig 1: hidden file

After detonation, the malware encrypts everything leaving the extensions as ".WNCRY" and eventually changing the background of the computer.


fig 2: wannacry's desktop

## Malware Skeleton

The Wannacry ransomware is dependent on several executables already coded into its software. While performing basic static analysis, some of these files were compiled at runtime as they didn't pop up in my scan.

| | |
|---|---|
| WannaCry | 24d004a104d4d54034dbcffc2a4b19a11f39008a575aa614ea04703480b1022c |
| Tasksche.exe | ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa |
| Taskse.exe | 2ca2d550e603d74dedda03156023135b38da3630cb014e3d00b1263358c5f00d |
| Taskdl.exe | 4a468603fdcb7a2eb5770705898cf9ef37aade532a7964642ecd705a74794b79 |
| Tor.exe | e48673680746fbe027e8982f62a83c298d6fb46ad9243de8e79b7e5a24dcd4eb |
| @WanaDecryptor@.exe | b9c5d4339809e0ad9a00d4d3dd26fdf44a32819a54abf846bb9b560d81391c25 |
| Taskhsvc.exe | e48673680746fbe027e8982f62a83c298d6fb46ad9243de8e79b7e5a24dcd4eb |

fig 3: malware skeleton and sha256 hash

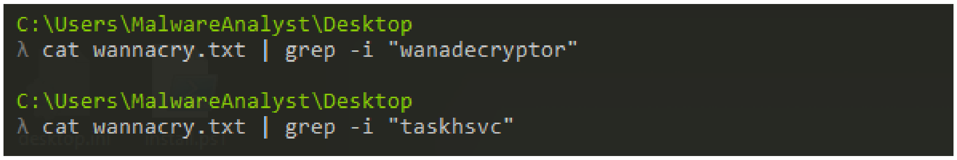



fig 4: compiled at runtime

# Basic Static Analysis

My static analysis begins with running wannacry against floss to grab any strings in the binary. This operation was successful since the wannacry ransomware wasn't a packed executable. So it had a good number of strings in it.

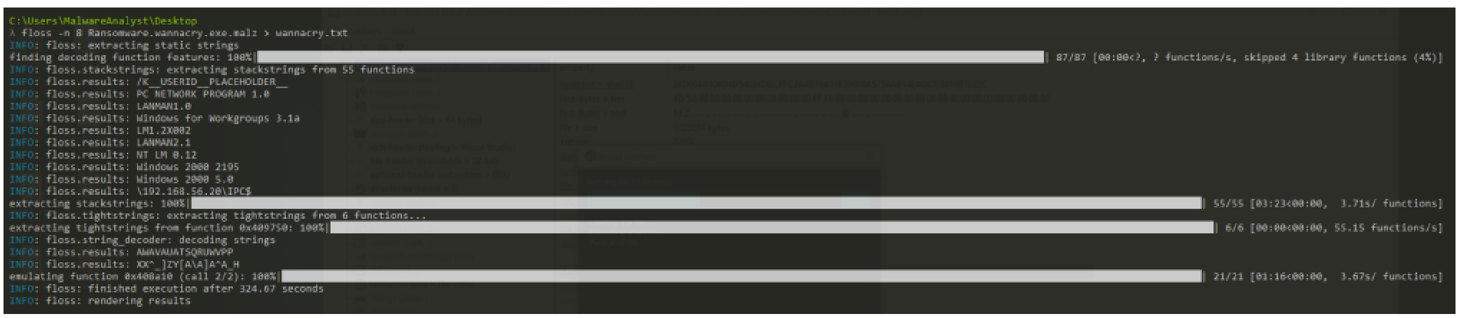



fig 5: running floss


fig 6: finding interesting strings

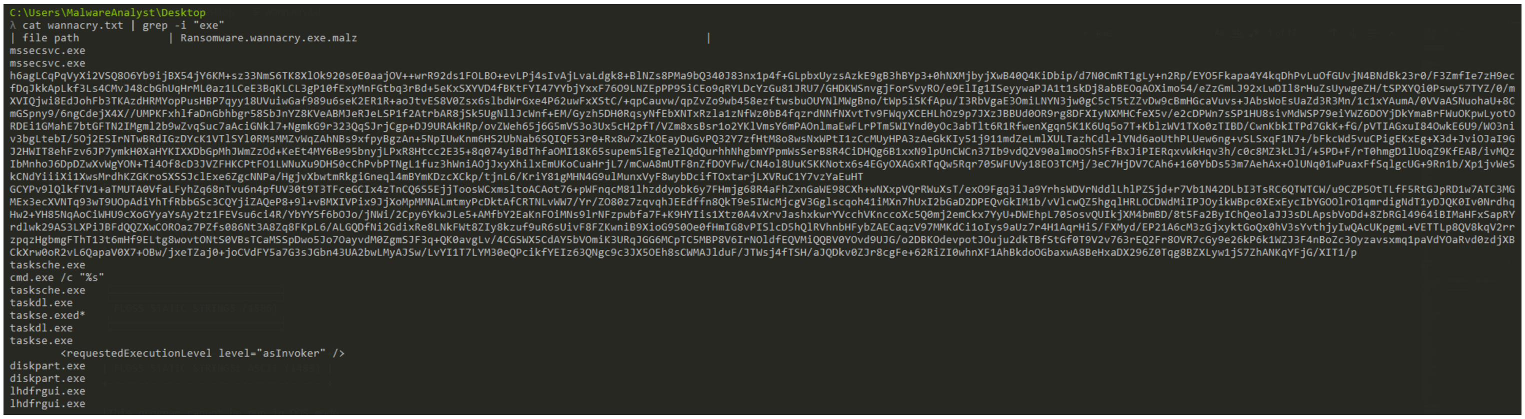



fig 7: finding interesting strings


fig 8: finding interesting strings - 2

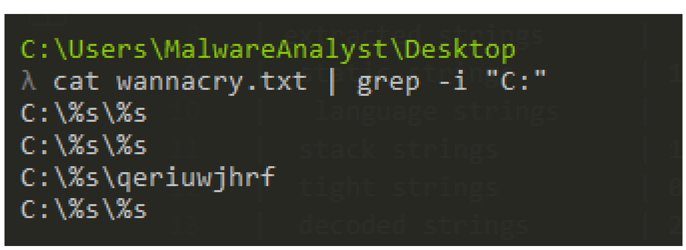



fig 9: finding interesting strings - 3

From floss, I found a URL, exes that would be dropped on the host computer, dlls, and, file system that the malware would work with. We see placeholders in the file path represented with“%s”.

Using Capa, I checked in a glance the malware behaviour, characteristics, and capabilities. Capa also offers the ability to see the malware tactics and techniques with the Mitre Framework.

```
C:\Users\MalwareAnalyst\Desktop
λ capa Ransomware.wannacry.exe.malz
+------------------------+------------------------------------------------------------------------------------+
| md5                    | db349b97c37d22f5ea1d1841e3c89eb4                                                   |
| sha1                   | e889544aff85ffaf8b0d0da705105dee7c97fe26                                           |
| sha256                 | 24d004a104d4d54034dbcffc2a4b19a11f39008a575aa614ea04703480b1022c                   |
| os                     | windows                                                                            |
| format                 | pe                                                                                 |
| arch                   | i386                                                                               |
| path                   | C:/Users/MalwareAnalyst/Desktop/Ransomware.wannacry.exe.malz                       |
+------------------------+------------------------------------------------------------------------------------+

+------------------------+------------------------------------------------------------------------------------+
| ATT&CK Tactic          | ATT&CK Technique                                                                   |
|------------------------+------------------------------------------------------------------------------------|
| DEFENSE EVASION        | Obfuscated Files or Information::Indicator Removal from Tools T1027.005            |
| DISCOVERY              | File and Directory Discovery T1083                                                 |
|                        | System Information Discovery T1082                                                 |
|                        | System Network Configuration Discovery T1016                                       |
| EXECUTION              | Shared Modules T1129                                                               |
|                        | System Services::Service Execution T1569.002                                       |
| PERSISTENCE            | Create or Modify System Process::Windows Service T1543.003                         |
+------------------------+------------------------------------------------------------------------------------+

+-----------------------------+-------------------------------------------------------------------------------+
| MBC Objective               | MBC Behavior                                                                  |
|-----------------------------+-------------------------------------------------------------------------------|
| ANTI-BEHAVIORAL ANALYSIS    | Conditional Execution::Runs as Service [B0025.007]                            |
|                             | Debugger Detection::Timing/Delay Check QueryPerformanceCounter [B0001.033]    |
| ANTI-STATIC ANALYSIS        | Executable Code Obfuscation::Argument Obfuscation [B0032.020]                 |
|                             | Executable Code Obfuscation::Stack Strings [B0032.017]                        |
| COMMAND AND CONTROL         | C2 Communication::Receive Data [B0030.002]                                    |
|                             | C2 Communication::Send Data [B0030.001]                                       |
| COMMUNICATION               | HTTP Communication::Create Request [C0002.012]                                |
|                             | HTTP Communication::Open URL [C0002.004]                                      |
|                             | Socket Communication::Connect Socket [C0001.004]                              |
|                             | Socket Communication::Create TCP Socket [C0001.011]                           |
|                             | Socket Communication::Create UDP Socket [C0001.010]                           |
|                             | Socket Communication::Get Socket Status [C0001.012]                           |
|                             | Socket Communication::Initialize Winsock Library [C0001.009]                  |
|                             | Socket Communication::Receive Data [C0001.006]                                |
|                             | Socket Communication::Send Data [C0001.007]                                   |
|                             | Socket Communication::Set Socket Config [C0001.001]                           |
```

fig 10: capa result

With Pestudio a tool used for analyzing malicious PE files, we see the CPU architecture of the malware and the language it was written in.

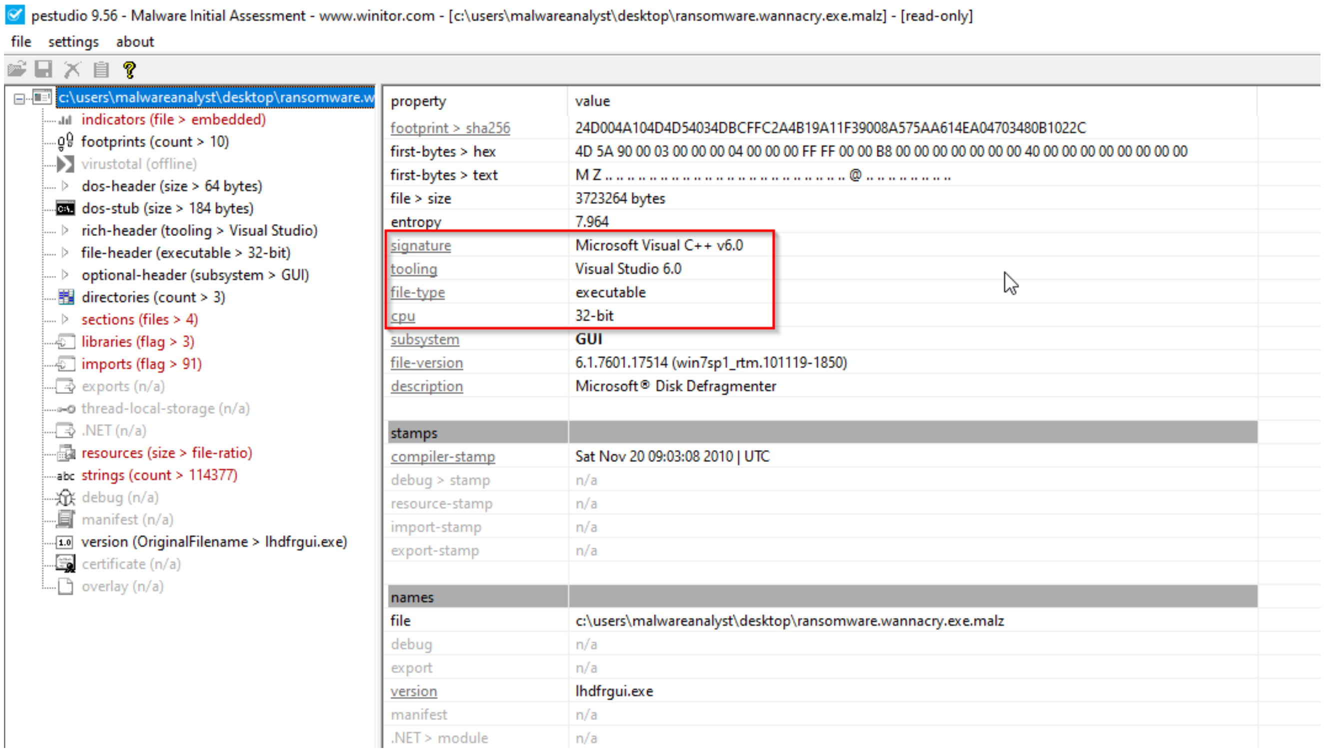



fig 11: pestudio file analysis

# Advanced Static Analysis

Advanced static analysis brings you closer to the source code written by the malware authors themselves. This allows in understanding the code better, the conditions, and the sequence. With cutter, this can be easily done.

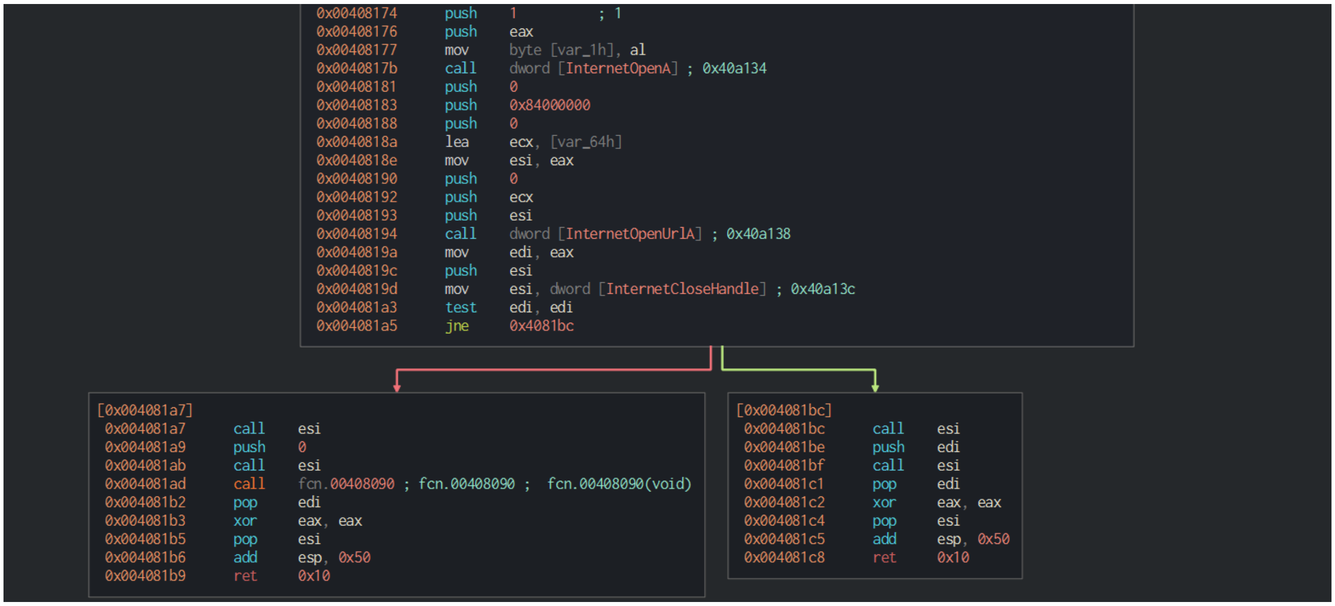



fig 12: wannacry’s main section

Decompiled:

```
int32_t var_64h;
int32_t var_50h;
int32_t var_17h;
int32_t var_13h;
int32_t var_fh;
int32_t var_bh;
int32_t var_7h;
int32_t var_3h;
int32_t var_1h;
ecx = 0xe;
esi = "hxxp[:]//www[.]iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea[,]com";
edi = &var_50h;
eax = 0;
do {
    *(es:edi) = *(esi);
    ecx--;
    esi += 4;
```

```
    es:edi += 4;
  } while (ecx != 0);
  *(es:edi) = *(esi);
  esi++;
  es:edi++;
  eax = InternetOpenA (eax, 1, eax, eax, eax, eax, eax, eax, ax, al);
  ecx = &var_64h;
  esi = eax;
  eax = InternetOpenUrlA (esi, ecx, 0, 0, 0x84000000, 0);
  edi = eax;
  esi = imp.InternetCloseHandle;
  if (edi == 0) {
    void (*esi)() ();
    void (*esi)(uint32_t) (0);
    eax = fcn_00408090 ();
    eax = 0;
    return eax;
  }
  void (*esi)() ();
  eax = void (*esi)(uint32_t) (edi);
  eax = 0;
  return eax;
}
```

Wannacry checks for a URL before proceeding to infect the host. If the domain turns out to be true, the malware is denied execution.

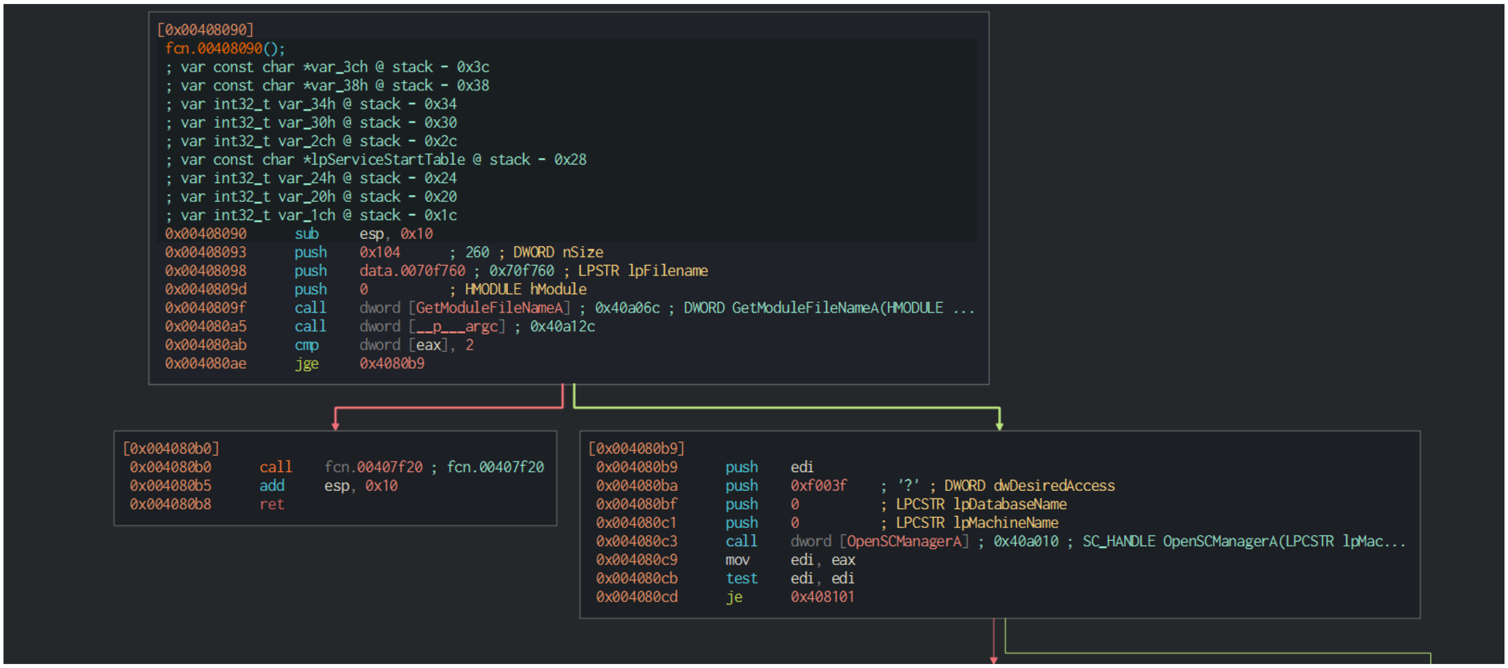



fig 13: wannacry's main section - 2

If the malware doesn't receive a response from the domain, it goes on to run the malware by calling fcn_00408090.

```
uint32_t fcn_00408090 (void) {
    const char * var_3ch;
    const char * var_38h;
    int32_t var_34h;
    int32_t var_30h;
    int32_t var_2ch;
    const char * lpServiceStartTable;
    int32_t var_24h;
    int32_t var_20h;
    int32_t var_1ch;
    GetModuleFileNameA (0, data.0070f760, 0x104);
    eax = p_argc ();
    if (*(eax) < 2) {
        fcn_00407f20 ();
        return eax;
    }
    eax = OpenSCManagerA (0, 0, 0xf003f, edi);
    edi = eax;
    if (edi != 0) {
        eax = OpenServiceA (edi, "mssecsvc2.0", 0xf01ff, esi, ebx);
        ebx = imp.CloseServiceHandle;
        esi = eax;
        if (esi != 0) {
            fcn_00407fa0 (esi, 0x3c);
            void (*ebx)(uint32_t) (esi);
        }
        void (*ebx)(uint32_t) (edi);
    }
    eax = &lpServiceStartTable;
    StartServiceCtrlDispatcherA (eax, "mssecsvc2.0", data.00408000, 0, 0);
    return eax;
}
```

fig 14: wannacry’s main section decompiled

## Dynamic Analysis:

Just like the result from basic analysis, I ran the sample with an internet connection using the fake internet provided by Remnux, the malware didn’t infect the host computer but it did leave behind network artifacts of contacting the domain This can be seen in wireshark

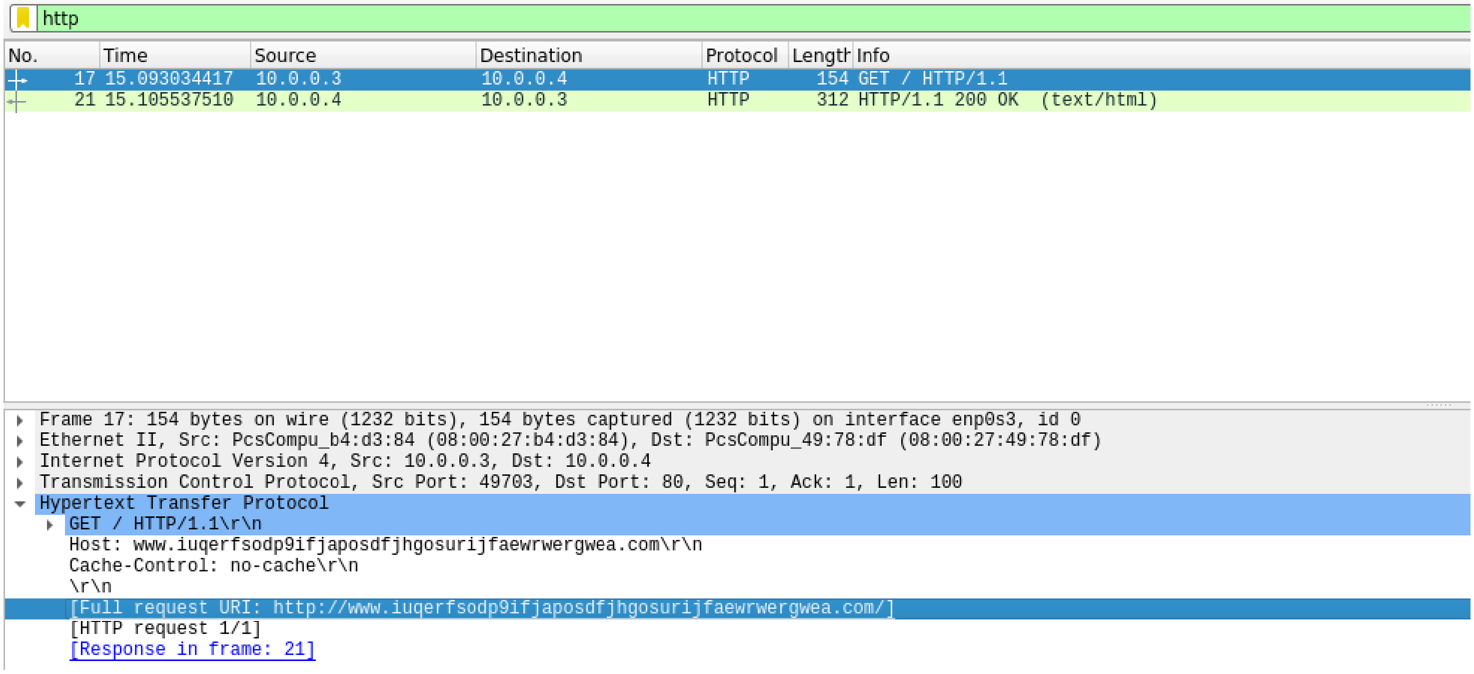



fig 15: wannacry's traffic analysis

Turning off the internet connection and running the sample again, we see it going to work.


fig 16: wannacry's infection

# Indicators of Compromise

## Host–Based Indicators:

Wannacry depends on many files. These files are dropped into the host system at infection after the domain check is performed by the malware. They are responsible for encryption, payment, network propagation, and persistence.

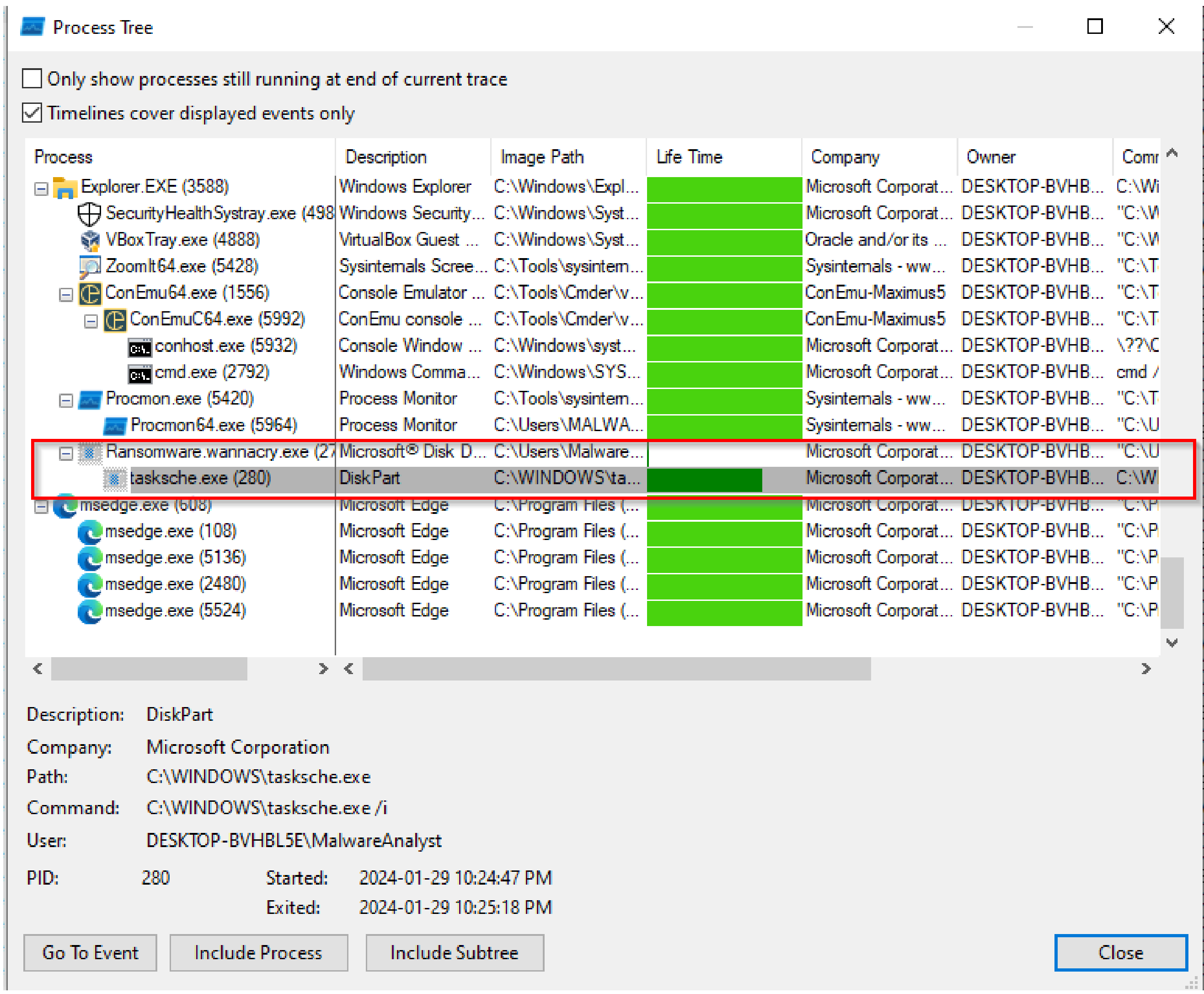



fig 17: wannacry's process activity

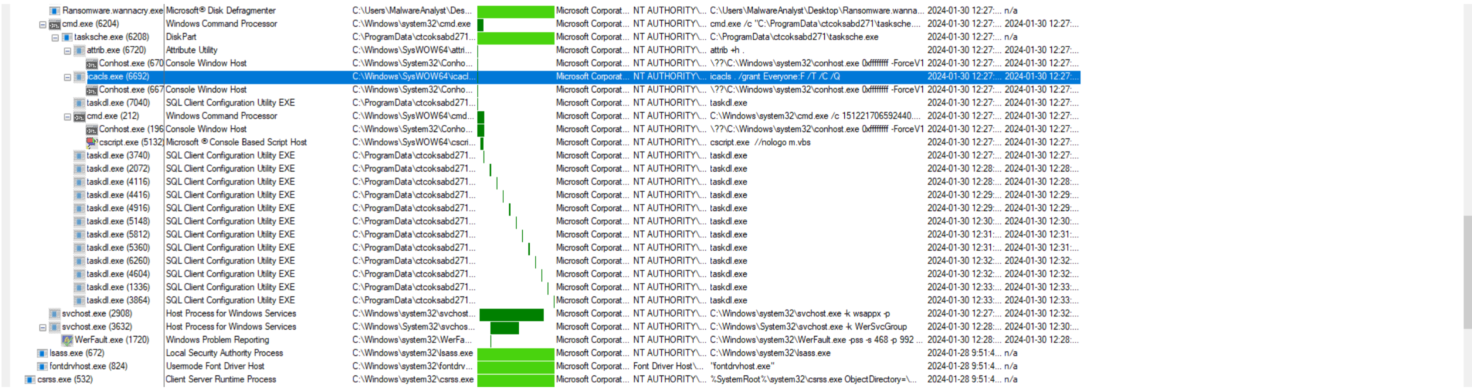


fig 18: wannacry's process activity – 2

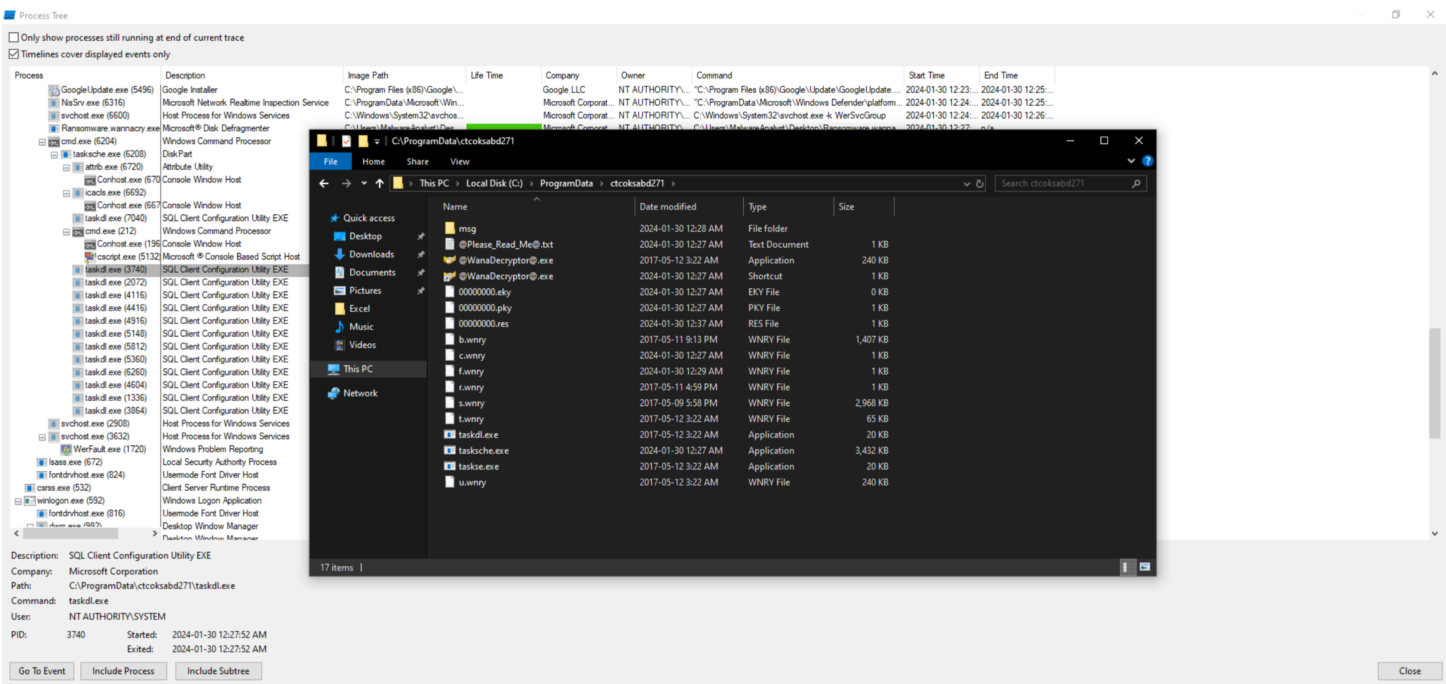


fig 19: wannacry's folder

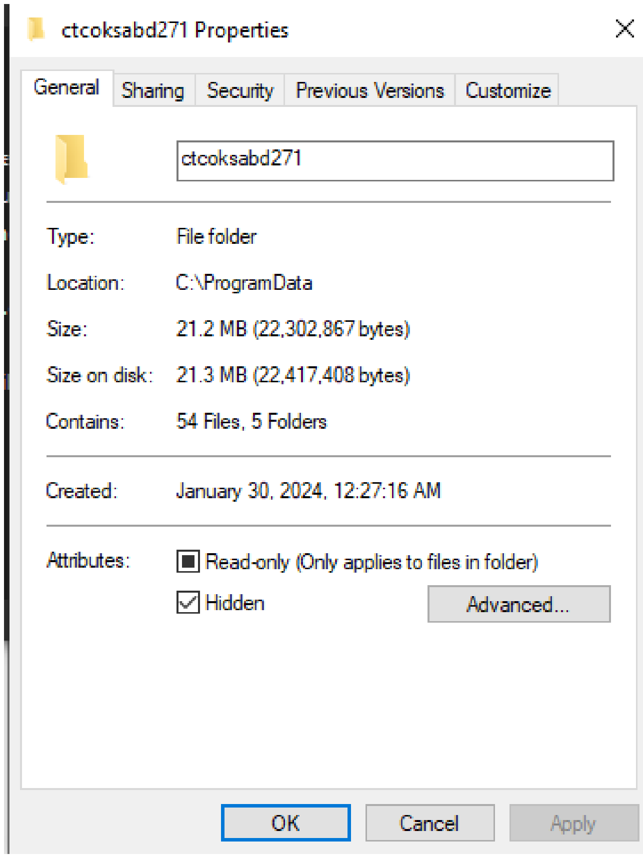



fig 20: folder properties

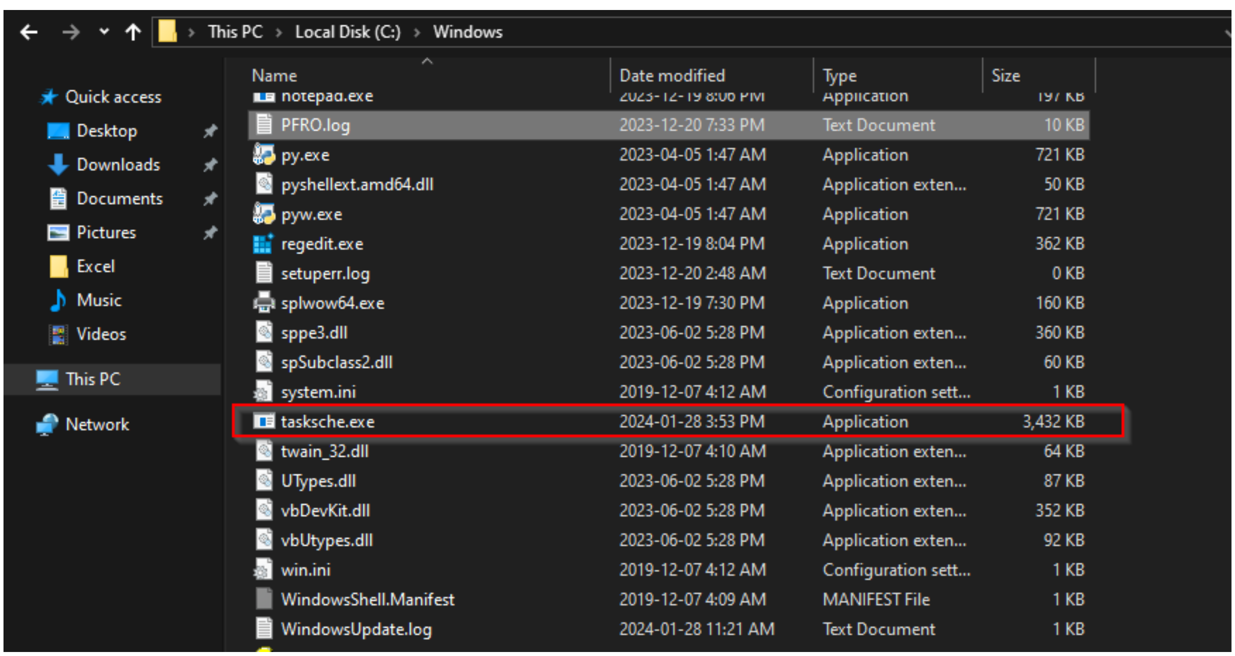
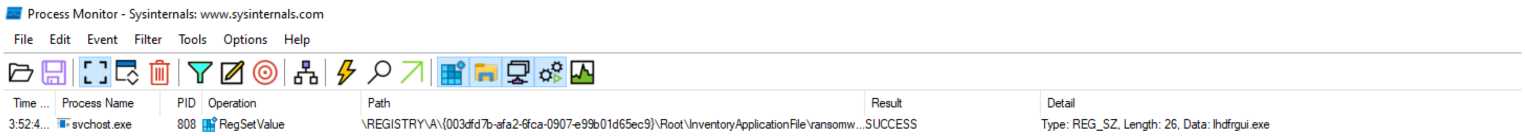



fig 21: dropped file

# During the infection, registry keys were modified



Process Monitor - Sysinternals: www.sysinternals.com

| Time ... | Process Name | PID | Operation | Path | Result | Detail |
|---|---|---|---|---|---|---|
| 3:52:4... | svchost.exe | 808 | RegSetValue | \REGISTRY\A\{003dfd7b-afa2-6fca-0907-e99b01d65ec9}\Root\InventoryApplicationFile\ransomw... | SUCCESS | Type: REG_SZ, Length: 26, Data: lhdfrgui.exe |

fig 22: registry key activity of lhdfrgui.exe

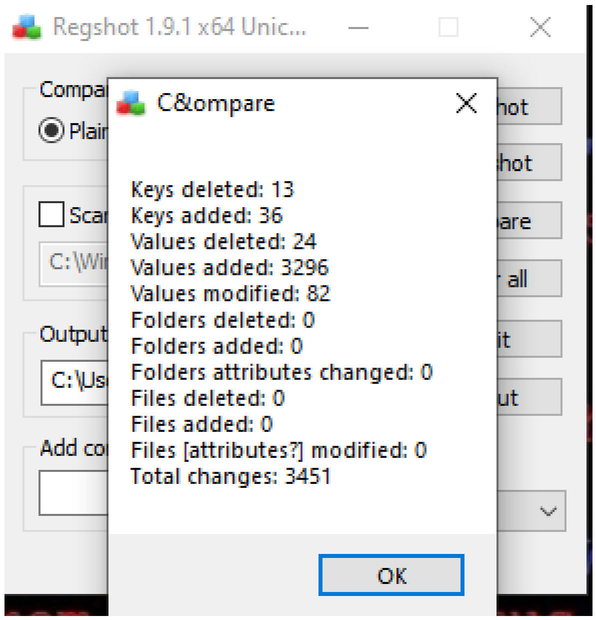
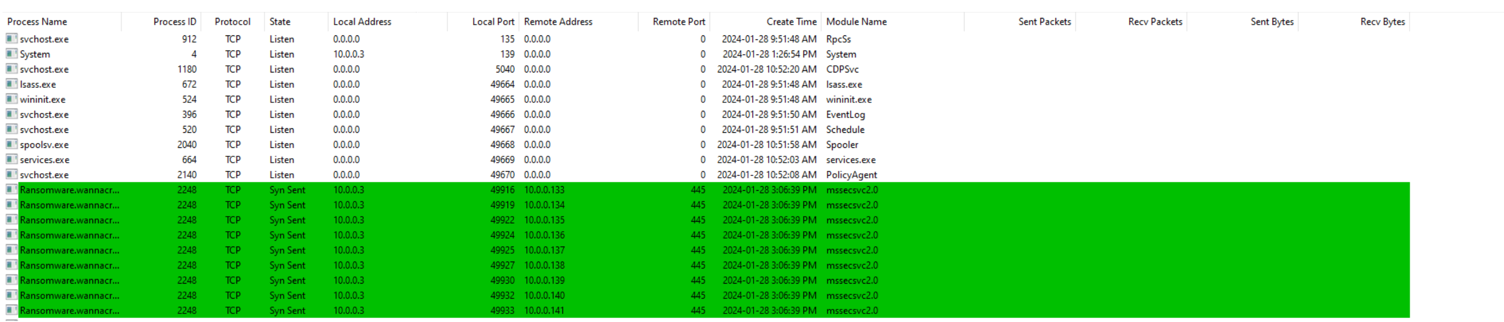



fig 23: registry key data

## Network–Based Indicators:

Over the network, wannnacry broadcasts itself by calling out to every host on the network.



| Process Name | Process ID | Protocol | State | Local Address | Local Port | Remote Address | Remote Port | Create Time | Module Name | Sent Packets | Recv Packets | Sent Bytes | Recv Bytes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| svchost.exe | 912 | TCP | Listen | 0.0.0.0 | 135 | 0.0.0.0 | 0 | 2024-01-28 9:51:48 AM | RpcSs | | | | |
| System | 4 | TCP | Listen | 10.0.0.3 | 139 | 0.0.0.0 | 0 | 2024-01-28 1:26:54 PM | System | | | | |
| svchost.exe | 1180 | TCP | Listen | 0.0.0.0 | 5040 | 0.0.0.0 | 0 | 2024-01-28 10:52:20 AM | CDPSvc | | | | |
| lsass.exe | 672 | TCP | Listen | 0.0.0.0 | 49664 | 0.0.0.0 | 0 | 2024-01-28 9:51:48 AM | lsass.exe | | | | |
| wininit.exe | 524 | TCP | Listen | 0.0.0.0 | 49665 | 0.0.0.0 | 0 | 2024-01-28 9:51:48 AM | wininit.exe | | | | |
| svchost.exe | 396 | TCP | Listen | 0.0.0.0 | 49666 | 0.0.0.0 | 0 | 2024-01-28 9:51:50 AM | EventLog | | | | |
| svchost.exe | 520 | TCP | Listen | 0.0.0.0 | 49667 | 0.0.0.0 | 0 | 2024-01-28 9:51:51 AM | Schedule | | | | |
| spoolsv.exe | 2040 | TCP | Listen | 0.0.0.0 | 49668 | 0.0.0.0 | 0 | 2024-01-28 10:51:58 AM | Spooler | | | | |
| services.exe | 664 | TCP | Listen | 0.0.0.0 | 49669 | 0.0.0.0 | 0 | 2024-01-28 10:52:03 AM | services.exe | | | | |
| svchost.exe | 2140 | TCP | Listen | 0.0.0.0 | 49670 | 0.0.0.0 | 0 | 2024-01-28 10:52:08 AM | PolicyAgent | | | | |
| Ransomware.wannacr... | 2248 | TCP | Syn Sent | 10.0.0.3 | 49916 | 10.0.0.133 | 445 | 2024-01-28 3:06:39 PM | mssecsvc2.0 | | | | |
| Ransomware.wannacr... | 2248 | TCP | Syn Sent | 10.0.0.3 | 49919 | 10.0.0.134 | 445 | 2024-01-28 3:06:39 PM | mssecsvc2.0 | | | | |
| Ransomware.wannacr... | 2248 | TCP | Syn Sent | 10.0.0.3 | 49922 | 10.0.0.135 | 445 | 2024-01-28 3:06:39 PM | mssecsvc2.0 | | | | |
| Ransomware.wannacr... | 2248 | TCP | Syn Sent | 10.0.0.3 | 49924 | 10.0.0.136 | 445 | 2024-01-28 3:06:39 PM | mssecsvc2.0 | | | | |
| Ransomware.wannacr... | 2248 | TCP | Syn Sent | 10.0.0.3 | 49925 | 10.0.0.137 | 445 | 2024-01-28 3:06:39 PM | mssecsvc2.0 | | | | |
| Ransomware.wannacr... | 2248 | TCP | Syn Sent | 10.0.0.3 | 49927 | 10.0.0.138 | 445 | 2024-01-28 3:06:39 PM | mssecsvc2.0 | | | | |
| Ransomware.wannacr... | 2248 | TCP | Syn Sent | 10.0.0.3 | 49930 | 10.0.0.139 | 445 | 2024-01-28 3:06:39 PM | mssecsvc2.0 | | | | |
| Ransomware.wannacr... | 2248 | TCP | Syn Sent | 10.0.0.3 | 49932 | 10.0.0.140 | 445 | 2024-01-28 3:06:39 PM | mssecsvc2.0 | | | | |
| Ransomware.wannacr... | 2248 | TCP | Syn Sent | 10.0.0.3 | 49933 | 10.0.0.141 | 445 | 2024-01-28 3:06:39 PM | mssecsvc2.0 | | | | |

fig 24: wannacry’s host network activity

Here we see wannacry sending out SYN – packets which is the first stage of three three–way handshake. It sends these packets to port 445 which is used by SMB. Wannacry was built to exploit SMB using eternal blue.

| Process Name | Process ID | Protocol | State | Local Address | Local Port | Remote Address | Remote Port | Create Time | Module Name | Sent Packets | Recv Packets | Sent Bytes | Recv Bytes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| svchost.exe | 912 | TCP | Listen | 0.0.0.0 | 135 | 0.0.0.0 | 0 | 2024-01-28 9:51:48 AM | RpcSs | | | | |
| System | 4 | TCP | Listen | 10.0.0.3 | 139 | 0.0.0.0 | 0 | 2024-01-28 1:26:54 PM | System | | | | |
| svchost.exe | 1180 | TCP | Listen | 0.0.0.0 | 5040 | 0.0.0.0 | 0 | 2024-01-28 10:52:20 AM | CDPSvc | | | | |
| taskhsvc.exe | 5012 | TCP | Listen | 127.0.0.1 | 9050 | 0.0.0.0 | 0 | 2024-01-28 3:14:23 PM | taskhsvc.exe | | | | |
| taskhsvc.exe | 5012 | TCP | Established | 127.0.0.1 | 9050 | 127.0.0.1 | 27392 | 2024-01-28 3:15:00 PM | taskhsvc.exe | 1 | 2 | 2 | 33 |
| taskhsvc.exe | 5012 | TCP | Established | 127.0.0.1 | 24587 | 127.0.0.1 | 24588 | 2024-01-28 3:14:23 PM | taskhsvc.exe | | | | |
| taskhsvc.exe | 5012 | TCP | Established | 127.0.0.1 | 24588 | 127.0.0.1 | 24587 | 2024-01-28 3:14:23 PM | taskhsvc.exe | | | | |
| @WanaDecryptor@.exe | 4788 | TCP | Established | 127.0.0.1 | 27392 | 127.0.0.1 | 9050 | 2024-01-28 3:15:00 PM | @WanaDecryptor@.exe | 2 | 1 | 33 | 2 |
| lsass.exe | 672 | TCP | Listen | 0.0.0.0 | 49664 | 0.0.0.0 | 0 | 2024-01-28 9:51:48 AM | lsass.exe | | | | |
| wininit.exe | 524 | TCP | Listen | 0.0.0.0 | 49665 | 0.0.0.0 | 0 | 2024-01-28 9:51:48 AM | wininit.exe | | | | |
| svchost.exe | 396 | TCP | Listen | 0.0.0.0 | 49666 | 0.0.0.0 | 0 | 2024-01-28 9:51:50 AM | EventLog | | | | |
| svchost.exe | 520 | TCP | Listen | 0.0.0.0 | 49667 | 0.0.0.0 | 0 | 2024-01-28 9:51:51 AM | Schedule | | | | |
| spoolsv.exe | 2040 | TCP | Listen | 0.0.0.0 | 49668 | 0.0.0.0 | 0 | 2024-01-28 10:51:58 AM | Spooler | | | | |

fig 25: wannacry's network process

On the host network, wannacry deploys different executables on the network for doing different things.

## Threat Intelligence

Looking up the Bitcoin address, I did find it and the transactions it had taken place in over the years.

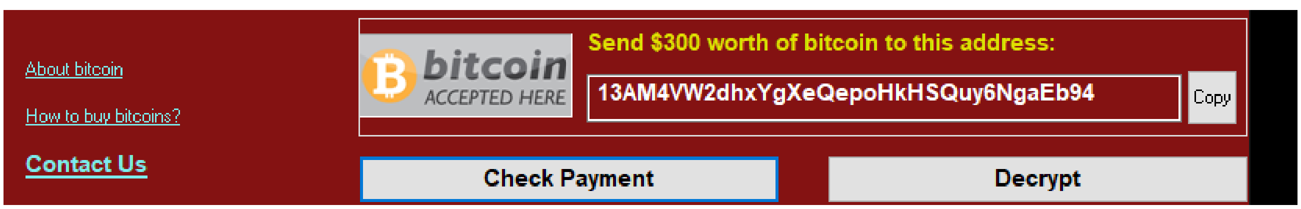



fig 26: wannacry bitcoin address

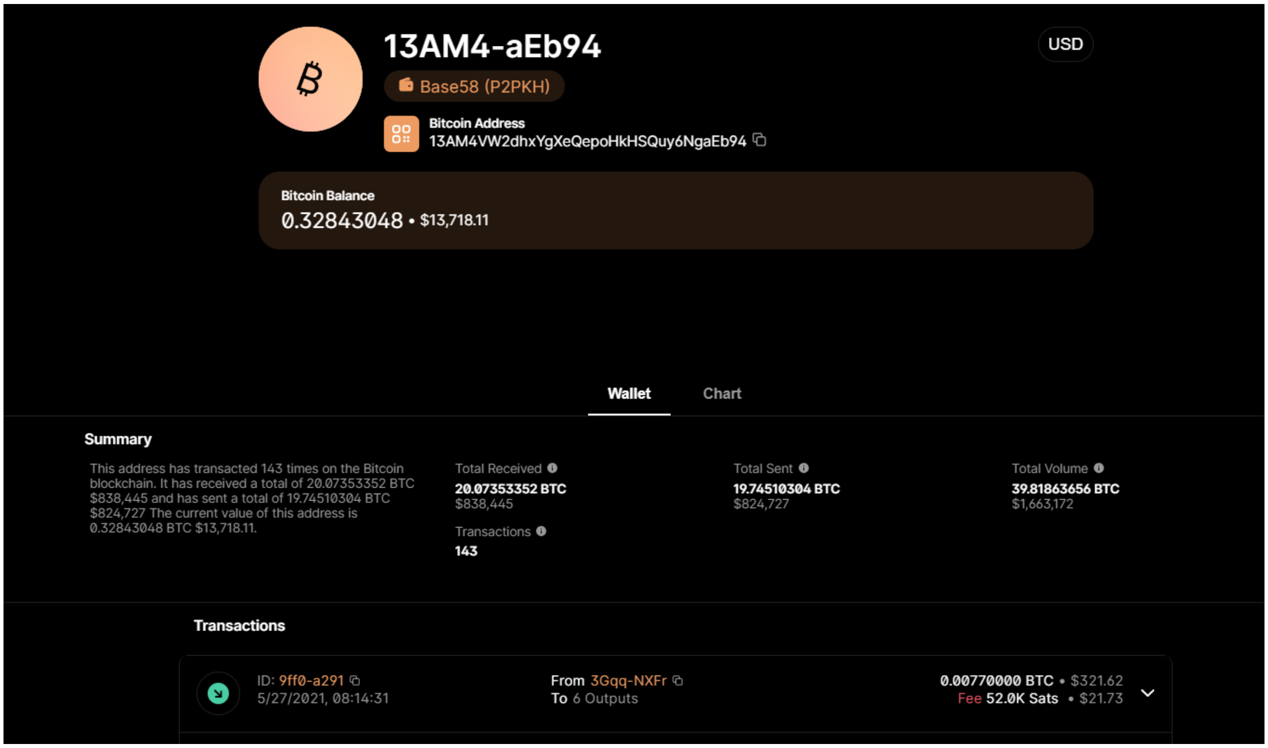



fig 27: wannacry's bitcoin wallet

# Rule set

To identify the presence of wannacry, I built a rule using Yara to do this.

```
rule WannaCry {

    meta:
        author = "Ab_Sec"
        description = "Yara rule for WannaCry ransomware"
        last_updated = "2024-01-30"

    strings:
        // wannacry won't run properly without some of these files
        $dropped_files = "tasksche.exe"
        $dropped_files1 = "tasksdl.exe"
        $dropped_files2 = "taskse.exe"
        $dropped_files3 = "@WannaDecryptor@.exe"
        $dropped_files4 = "mssecsvc.exe"
        $dropped_files5 = "lhdfrgui.exe"
        $dropped_files6 = "diskpart.exe"
        $malware_note = "MZ"
        $malware_note1 = ".WNCRY"
        $malware_note2 = ".wnry"
        $malware_note3 = "PADDINGXXPADDING"
        $malware_note4 = "icacls . /grant Everyone:F /T /C /Q"
        $malware_check = "http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com"

    condition:
        $malware_note at 0 and $dropped_files and $dropped_files1 and $dropped_files2 and $dropped_files3 and $dropped_files4 and
        $dropped_files5 and $dropped_files6 and $malware_note1 and $malware_note2 and $malware_note4 or $malware_check or $malware_note3

}
```

fig 28: yara rule for detecting wannacry