



Practical Malware Analysis and Triage Malware Analysis Report

Sample: Ransomware.wannacry.exe
Family: WannaCry

November 25, 2023

Muhfat Alam

Table of Contents

| Contents | Page |
|---|-------------|
| Summary | 03 |
| Technical Summary | 04 |
| Static Analysis | 06 |
| Get the File Hash | 06 |
| OSINT Tool | 06 |
| VirusTotal Verdict | 06 |
| MetaDefender | 07 |
| String/Floss Analysis of the Executable | 08 |
| PESstudio Report: | 11 |
| Dynamic Analysis | 14 |
| TCPView Report | 16 |
| Host-Based Indicator (procmon.exe) Report | 18 |
| VirusTotal Verdict for packed executables | 23 |
| Advanced Static Analysis | 24 |
| Advanced Dynamic Analysis | 28 |

Summary

MD5 and SHA256 hash value for **Ransomware.wannacry.exe**.

MD5: db349b97c37d22f5ea1d1841e3c89eb4

SHA256:

24d004a104d4d54034dbcffc2a4b19a11f39008a575aa614ea04703480b1022c

URL: hxxp[:]//www[.]jiuqerfsodp9ifjaposdfjhgosurijfaewrwergwea[.]com

Unpacked Executables: There are three executables were unpacked from the original file Ransomware.wannacry.exe.

1. tasksche.exe /
ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa
2. taskdl.exe /
4a468603fdb7a2eb5770705898cf9ef37aade532a7964642ecd705a74794b79
3. taskse.exe /
2ca2d550e603d74dedda03156023135b38da3630cb014e3d00b1263358c5f00d

Command Lines: C:WINDOWS tasksche.exe /i

File Created: wulicfidttvazj524

File Path: C:\ProgramData

Ransomware.wannacry.exe is a Trojan Virus, part of the **WannaCry** family which encrypts all of files until pay for decryption with Bitcoins. **Besides infecting our local machine, it also has worm capabilities that speed through the local network.**

Technical Summary

The **WannaCry** Trojan, identified as "**Ransomware.wannacry.exe**," is a notorious piece of malware known for its widespread impact on computer systems. This technical summary provides an overview of its functionality, including its behavior when checking for an internet connection, interaction with a specific URL, payload execution, and propagation methods.

Internet Connection Check:

WannaCry employs a **two-stage execution** mechanism. First, it checks for the presence of an internet connection. If an active internet connection is detected, it refrains from executing its payload. This initial check is likely a safeguard to avoid detection and analysis by security researchers. The absence of an internet connection is a trigger for further malicious actions.

Interaction with a Specific URL:

Upon confirming the absence of an internet connection, **WannaCry** attempts to reach out to the URL "**hxxp://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com**." If successful, this communication may have served as a kill switch to halt the malware's execution. The use of a kill switch domain was a significant discovery during the initial **WannaCry** outbreak, as it allowed for a temporary halt in its spread.

Payload Execution:

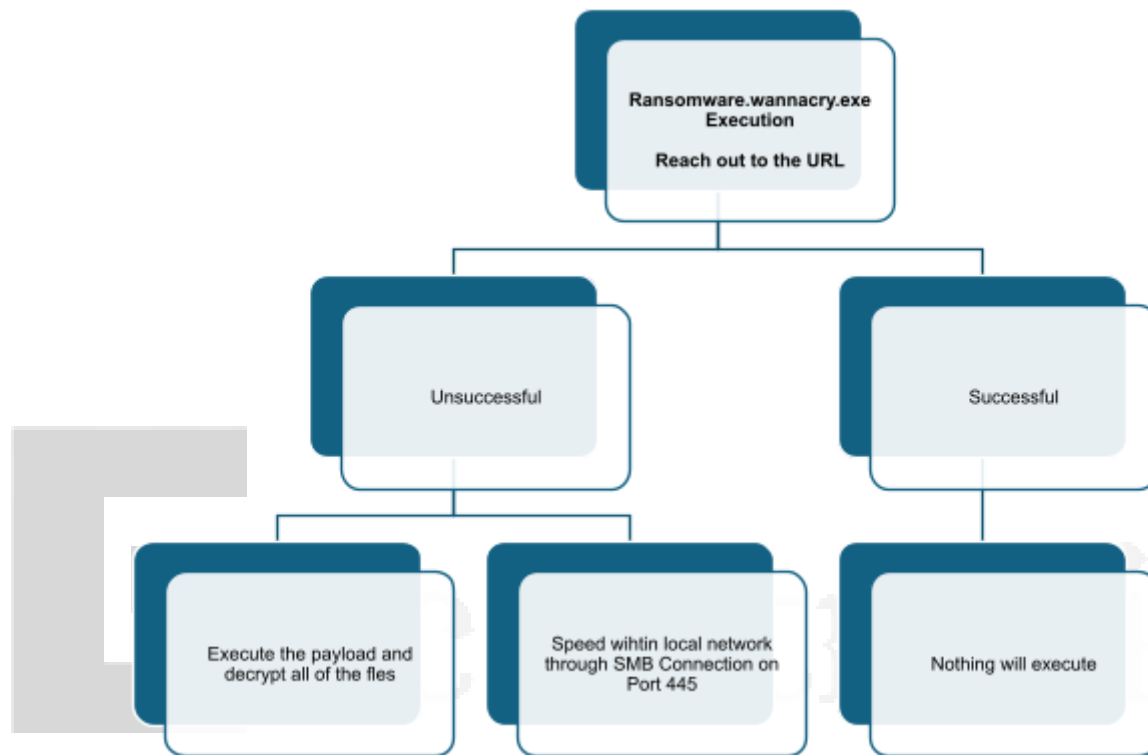
In the absence of an internet connection or if it cannot establish communication with the specified URL, **WannaCry** proceeds with its malicious activities. It executes its payload, which primarily involves encrypting files on the infected system. This encryption is often carried out using strong encryption algorithms, rendering the victim's files inaccessible without a decryption key.

Propagation through SMB Connection:

One of the most notable aspects of **WannaCry** is its ability to create an **SMB (Server Message Block)** connection with local networks. It leverages known vulnerabilities in the Windows operating system, particularly the EternalBlue exploit, to propagate itself across networked systems. This capability allowed **WannaCry** to rapidly spread within organizations and across the internet during its initial outbreak.

The **WannaCry** Trojan, encapsulated in the "**Ransomware.wannacry.exe**" executable, is a highly destructive piece of malware that demonstrates a two-stage execution process. Its behavior is contingent on the presence of an internet connection and its ability to interact with a specific URL. When executed, **WannaCry** employs file encryption, packs additional binary files, and utilizes SMB connections to propagate within local networks. The malware's impact has been felt globally, making it a significant case study in the field of malware analysis and cybersecurity.

Graph:



Static Analysis

Get the File Hash

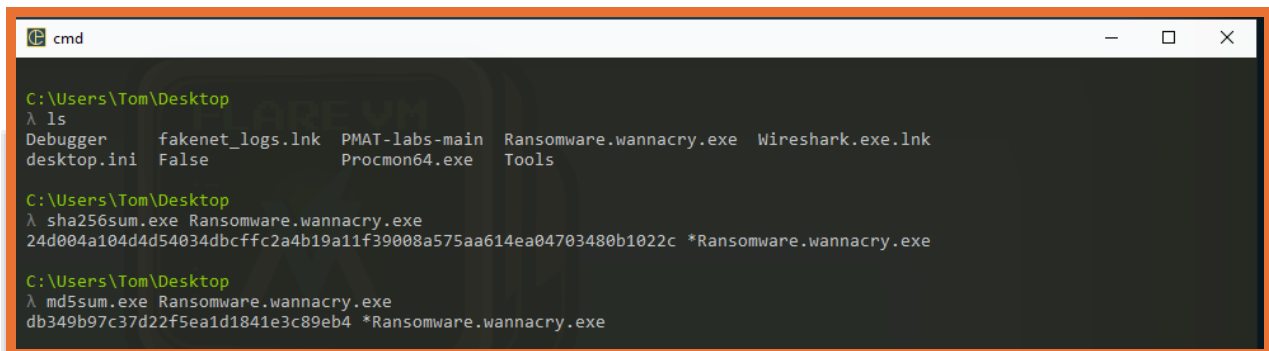
Get the hash value for the executable **Ransomware.wannacry.exe** on the cmd by running the following command (Figure 01),

...

```
sha256sum.exe Ransomware.wannacry.exe //in sha256 hash
```

```
md5sum.exe Ransomware.wannacry.exe //in md5 hash
```

...



```
cmd

C:\Users\Tom\Desktop
λ ls
Debugger      fakenet_logs.lnk  PMAT-labs-main  Ransomware.wannacry.exe  Wireshark.exe.lnk
desktop.ini    False             Procmon64.exe   Tools

C:\Users\Tom\Desktop
λ sha256sum.exe Ransomware.wannacry.exe
24d004a104d4d54034dbcffc2a4b19a11f39008a575aa614ea04703480b1022c *Ransomware.wannacry.exe

C:\Users\Tom\Desktop
λ md5sum.exe Ransomware.wannacry.exe
db349b97c37d22f5ea1d1841e3c89eb4 *Ransomware.wannacry.exe
```

Figure 01

OSINT Tool: As we have the hash value from the command line, we can use some OSINT Tools, like VirusTotal and MetaDefender for any flagged by security vendors.

VirusTotal Verdict: There are **65/67** security vendors flagged this Ransomware.wannacry.exe / 24d004a104d4d54034dbcffc2a4b19a11f39008a575aa614ea04703480b1022c file as malicious and **identify as a trojan, ransomware, and worm with WannaCry family.**

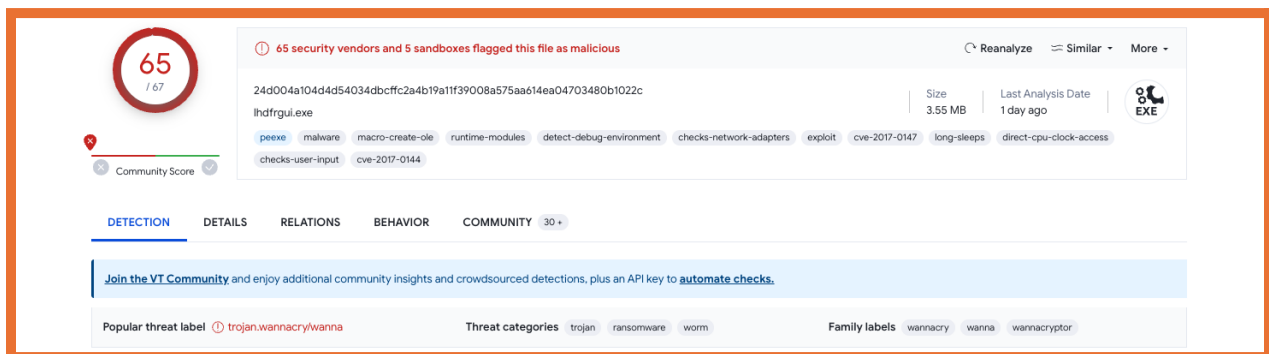


Figure 02

MetaDefender Result: The threat is identified as WannaCry and has 29 threats out of 32. (Figure 03)

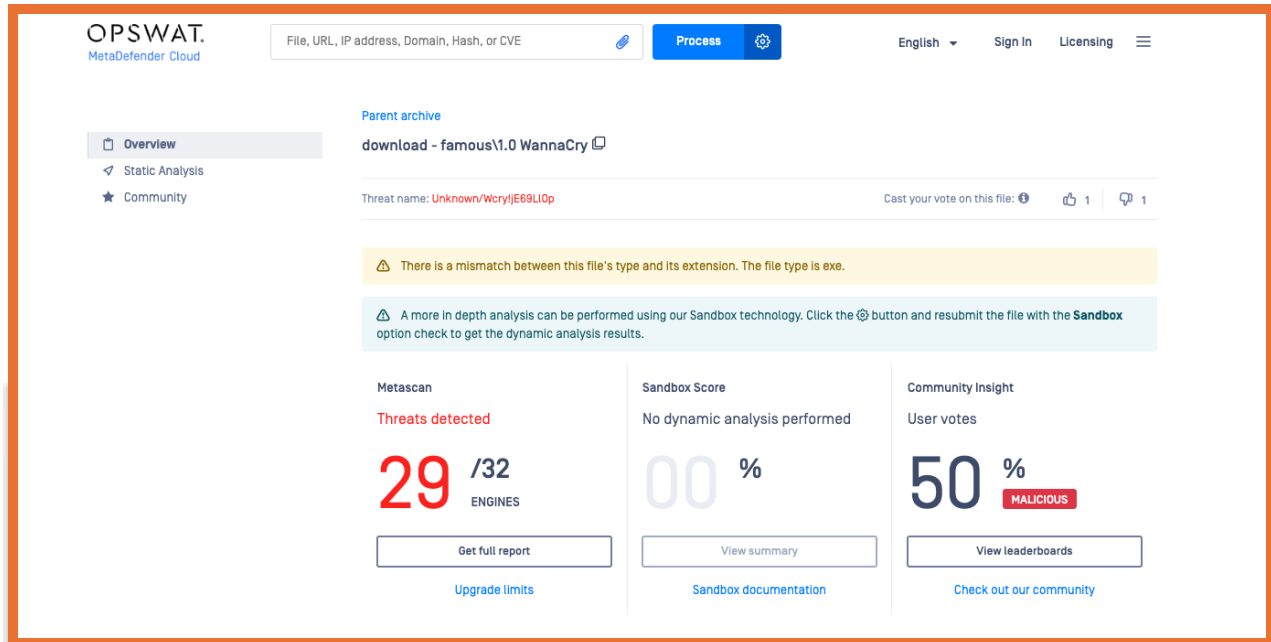


Figure 03

String/Floss Analysis of the Executable:

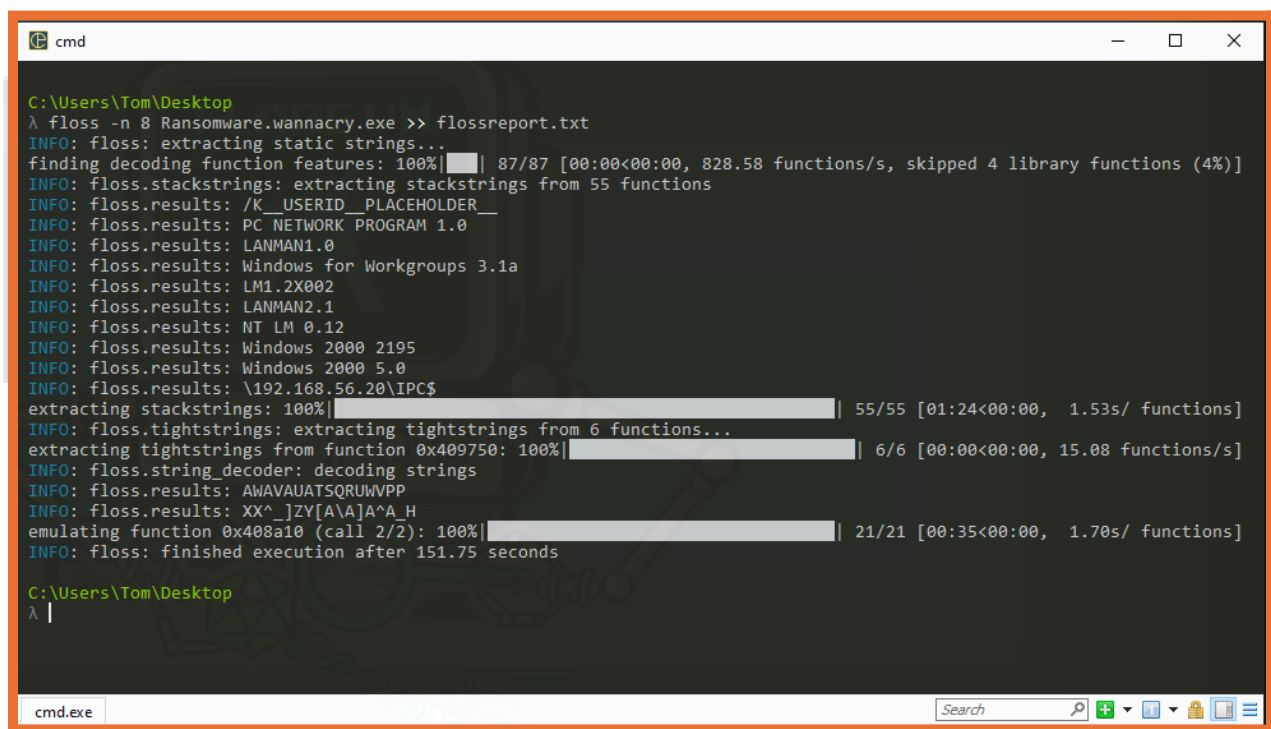
To pull the strings out of the binary of the **Ransomware.wannacry.exe** file, we use **floss** in the command line, and as it has other unnecessary strings, we can pull out more than 8 characters by mentioning “-n 8” in the command line.

As this a very famous and has more data inside of this executable, I’m going to save this string file inside a text document. (Figure 04) Follow the command to get the more than 8 characters string out of this binary file:

...

```
floss -n 8 Ransomware.wannacry.exe >> flossreport.txt
```

...



```
cmd
C:\Users\Tom\Desktop
λ floss -n 8 Ransomware.wannacry.exe >> flossreport.txt
INFO: floss: extracting static strings...
finding decoding function features: 100%|██████████| 87/87 [00:00<00:00, 828.58 functions/s, skipped 4 library functions (4%)]
INFO: floss.stackstrings: extracting stackstrings from 55 functions
INFO: floss.results: /K_USERID__PLACEHOLDER__
INFO: floss.results: PC NETWORK PROGRAM 1.0
INFO: floss.results: LANMAN1.0
INFO: floss.results: Windows for Workgroups 3.1a
INFO: floss.results: LM1.2X002
INFO: floss.results: LANMAN2.1
INFO: floss.results: NT LM 0.12
INFO: floss.results: Windows 2000 2195
INFO: floss.results: Windows 2000 5.0
INFO: floss.results: \192.168.56.20\IPC$
extracting stackstrings: 100%|██████████| 55/55 [01:24<00:00, 1.53s/ functions]
INFO: floss.tightstrings: extracting tightstrings from 6 functions...
extracting tightstrings from function 0x409750: 100%|██████████| 6/6 [00:00<00:00, 15.08 functions/s]
INFO: floss.string_decoder: decoding strings
INFO: floss.results: AWAVAUATSQRUMVPP
INFO: floss.results: XX^_]ZY[A^A]A^A_H
emulating function 0x408a10 (call 2/2): 100%|██████████| 21/21 [00:35<00:00, 1.70s/ functions]
INFO: floss: finished execution after 151.75 seconds

C:\Users\Tom\Desktop
λ |
```

Figure 04

As we open up the string value from the **Ransomware.wannacry.exe**, there are several things that need to be noted. Like, it is an executable by seeing “**!This program cannot be run in DOS mode.**” However, we see this several times, which seems like we have packed executables, and other packed executables inside of this first stage executable. (Figure 05)

There are some API calls imported as well as the **.dll** that are imported for those calls. (Figure 05)

Based on this we can make a hypothesis that there are either a few other executables or .dll that are loaded in and perhaps these APIs underneath this DOS header might be the APIs that are imported from these other executables.

After that, we see there are several base64 encoded characters, which might be helpful later for our analysis. (Figure 07)

We also found a URL

(hxxp[://]www[.]iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea[.]com) that might try to get communicate from our local machine. (Figure 08)

There is a string name “attrib +h .” which creates a hidden directory. (Figure 09)

```
FLARE FLOSS RESULTS (version v2.3.0-0-g037fc4b)

+-----+-----+
| file path           | Ransomware.wannacry.exe |
+-----+-----+
| extracted strings   |                           |
| static strings      | 1586                      |
| stack strings       | 10                        |
| tight strings       | 0                         |
| decoded strings     | 2                         |
+-----+-----+

FLOSS STATIC STRINGS

+-----+-----+
| FLOSS STATIC STRINGS: ASCII (1483) |
+-----+-----+

!This program cannot be run in DOS mode.
t4;1u#SV
GetTickCount
QueryPerformanceCounter
QueryPerformanceFrequency
GlobalFree
GlobalAlloc
InitializeCriticalSection
LeaveCriticalSection
EnterCriticalSection
InterlockedDecrement
CloseHandle
TerminateThread
WaitForSingleObject
InterlockedIncrement
GetCurrentThreadId
GetCurrentThread
ReadFile
GetFileSize
CreateFileA
MoveFileExA
SizeofResource
LockResource
LoadResource
FindResourceA
GetProcAddress
```

Figure 05

```
adjust.iov
launcher.dll
PlayGame
C:\%s\%s
mssecsvc.exe
!This program cannot be run in DOS mode.
/4%D/4%D/4%D4
```

Figure 06

```

0/WXK9d/UQIHT742mCLfI/SkPkd+STSCSdLdwZ7vHDF4tXLS5Pa2W3ArpVhLhQRoU/mNqVivObXCm6Q2atx6H7k9hHfYxX2btGB04e9Wyz+KrieA6wX0SAIE
RL1S0eEq0ScCwzoUrrLcYVYi+9wdiVuPzY6wN28tmhIY07Q+UFFfP/bcJ4FTPy5qnpEnX6v+n88x+c1TSRKczH29osGZF3WtNaDyBIrDNbC0tpV0TCLMao
Z1SG67W1BrYdFwMKBsYUJRWwZkhWE+eFof/Ewh9W8LWe7JEzAvvJYt07N5Sw+c9EShesxkpLngceIndRanvCLX7wXm3fafa4vmNwdsNkYmogq2NlyqNwI
oat18rcwQ6B69YaTzBG61wXqRaLa7LUDf0Ircv1081m+qjz0r4bdkHkLa/udu0625Y10e4sfnCUHLpDtHW050CUj7HyIV1C0r2a
TREEID_PLACEHOLDER
USERID_PLACEHOLDER_@
jmrDxLSLx+Xh5q8F0FE2cThv0tIqd6S1Y4eiHN6d+BFx56Y25KpW03XiXsV9dM0uK9Cnykc833bluEUu+UndX/LZ0idix/C1/kT5iPaQodLnCNRXWSpG
isagFUQ1kPTDE5PaEv7DHH7+cDobnaPw0ZNYGJISUR/kQ1zLE67rBN2haI17MRXoEdLSJmrF179xGu+5mt8gtVP7CYsoceDmfKJymPyZ0d8+N7IXdF3ji
7woekJazvE+qBve/a8t90k2E/BhmKMp003bDuts/AM0L97Chw0v0u33qZfKAX0Pzz643jrfILWY/NXekL+Pur/XwPUDR0LlnRvCy7EqxE3XYWj3YfcP
D00I1pIuE9SLMhQZF/gTrLGXBH0SmaV+lpMUCnn7DqZ/g04ExwCCy8RJR0HtEruT10FYVto187X1faqQceYawLd081EeNIT/LQe3Fg+4H40Mu5d0XR4hk
ig90ZHIkbw5k3D1YS24tEbLEGZQmJCU9px4pPQFVn61r3p22o0PIgEjZ655VMPwyX18a02f5AgNNIBC7t7pnSpTjYwas3U9gTo5BDmderdeAh1bdqarM61K
CBRfdQ1RVGSazoC/zZZXcEcL06MoI9Z6gE5duA00aXrByRwnuu0TV/77KHepF134nHeW6Z5b/TIRRHQB8u06E1mmWtUsjID+LHKrGxRgFbS0y2937EHPiU
2WTF12sg/iZr95EKgP3mmUPBNA068Fwi8C/4n+ycc0d2o70YH76a75h9ofch0u50b29p0nQVSN/KwyJtkqMNUKf8XaYvWbHxob1RWYrk/IqHPRx7+hcGYA
iIzZknS9ChvNkiy9C2ph6AC5TPqHC4i0enEQW9b5kaebv6+2Pua6DMKCjSPNb982W01I+2v048/eaDhIX1KIIsquM+mk0e4TF9RLaUopAoCFC3TCiwiFMR
NkKpwYSna0wJvArW0IBqVJDafo+/Mk0eMkYLSYhdkaED+4pyjvBzju4hu70PUK06iNuBAsae10QybI22YaffbkMtZyUSe9za40a2s6cfx0tp+MUTd+WH
Lbm+nH0xX8Wd2vWfULRmX0CFWt0XqNhxPxY1F9rIpEvYg6MVepvqn8QmJo+LHMHDZj7MZpVXuLrgX81PIrpvrU7v1Cf4T/wvEZNvVWYLS2UWNe93cLPu
U9S00csNULFH5evrsj3lVX0MIePVzVECa6ugpv9qcnq0tbHAMxTbcB14jvyDLL7yPTQ0PFCW1TkpOrYhAcch11HuTyS3NdX1Q+1UyWFOutUxi9NzaCqsRc
l6J79h2y39JwpxZyUdZKFS5P7gAbUqWFnXe/0168Tb2LdoHagxK6D20YFK0I7r6tHhckA6RJGfmQxv9vU1tqxuFZaJlausy9JcgA1LlU
tzfxH4tRAMAPThYnmQ3AWhtZJpXyp4JycPGMEDTbGswLmCyzX09dx04MxqerN0u5Lvg8ubw/zw1+7MwQKpDKrA60B0E4KT6+YXaP1ZBp19m6Wtd8cA
fCtcrADQ5PZI20dtI4ZgfcK6KwCq0jsX1mGx19VoJRTUCLUjZBEI3dupfHnWSpHbMEck0F0D1+Sd1cJ12NpkaTmNaISSbLkqoiXI1XMPt+2E2JVgS0yx
iTG8oP54gX83sN0/C0+qcbkRF12+ShXxq5MWQj7VcZ9nYe1xQP6DuCbm7XWUnsAGTfchn0NUZu3zAnzDb99VSLMKdS8F1t5WNX1kAFFHrgmQBthVR2pTyc
rqnaN2drtI0jmn9b8U8DdI46voDUaCf1Ccw0IHPRT4DNbp80uTo4MhB0M82icievXpYw4CsVa6Uxw4AqVX3yS6vJSW4rQUKnK4wwYe7L0r3aAFsQRFB4X
srlQRpashzdbZmGRm+RF0fduem93+S3fk2Wu/s/0Vr8jnnIbrrhLk0Cdu2RTNuJdxCS6JEDsNyHjSkXPoQi09uUAH6xrPodzLKK1VfwjMaaI2p5D19aNsJ
qDwditvy/cjXonVsSipgV1YjdrWYCKW4GhKhYrPETzxIg1PrPzbpxt4h7uih34du4K4VutQeyVugNQWcsYY2C4ByfHxGoFMDfKfrizAgyVRB40i5aHv9N
c0TjJcdMhwCX0NeUNLlTsAeLPjUtVBobANNBFBKIVBKSobjeIH/XPK1PoRwUibAYAut951w6x0h3D435cqi1GvHxm3dhkqcsH3PBajZivY9e41JM0eg3
Mkv+MuNI2Iax0u25YreU/xmJ6urZ86cqKDanxu8VrfUxRScc01LVmtUkggM0cPkq7k6KEW7ued/BLAUvkbc8L5g5HkS6TStIYpkVM4KM149iygRM841Xx
9as4RZsesIE6ViI4NYWLYtM/bbZ205kh6XwtgeTN7/eYA7tD0Hra9um6Y07MI2WqzViMpy1MdKiBVBsfi10KiFSGBxBEB5IUGcwj7NWJmEDvmR13hAVcJT
uYsXonn7/xxVnKtGSTZEOZebfk9PHHJ5v1VQrbAvsLNUMNQ5fzBFW2C7RorfiBgCB5M/8UC0JXmc+qyN2wff0BuvGZiHYqLpZ/UVCNWatHUhj1zvwYIMb
iYzsANsKYtmsd6vAmGwvYt0I0ex2IVlczdWcVf7TRTbb69p/s1PkJRoyFwaZfnLx575TI/QUAWGS2Ncyafj80NRQdXRIxwLdXYX0mHB3fKqY0dxfJtoIC
CBZr7oIAZu0m2mp4FtkLKsmi0P1f1B2eudK9mfuypt/ds1WA/IX051CH20TPUfMy0+BBTRs2v5+X8U+B/D6IogB5c110Q/81PsTAh7/92s0cLARterruT
JBZKpDS2Pyj6wG1VJNcPI+qgJBM5voWnDnwkyxBX1I/64dEpHuTT9U18qG3rRJNqo8SIEIJRq45TaKwge+3YYXd9XxMNFyVo/IqzSSTBPFrsBPVV4q+Nd7
iH1lecNRIf2fj39QAYqH61Qis099aK4nIgLW9Z0kKq17SZ+qPq2E6h5pM1PbcmyYny+TeAddFeVIS0UU2ln4x9ooZQM1e4K65oTdJwD+7/hffhGwY5c0
WbBY1jx42a0ypDNIEIFT+qlrHtrEx7cYG401C8G+hHgBy2hfJeph1K2jV1bUyWbKAV/hLn0JZFv3GE7+jxL4ZH3IFGh4nzYopsJUNYB816bjYmbrLB7iT
nycV9wMgV9xDRFYK4970/NP2atXeYdQnW26ZD/k4PrxP/h+7s9fD1yb2KxS+LcYIp1AX76nRqL6U8a9z6gqCQ1QtU70SUB1iFu/2NYLd0zbWyc3HGns
N0aK6fsY06mFU7siltIfdgZsV40DfrFynl0a+nwiSv12XWrcGekfCsyL7P1DlBc5hq1TQ7DBLberorm9rqvrv2IkepTh2rtJf0k6Zios9d1eq6ZiFemrw
RmtvpJd4PEc04jccpu0FAjHMGisyHLrqKV9rdP4nzEE079xX4R0ZIpmXNMWxe3k0hYzb8TwY1IguFkVqbp4R0IHxwMMvmgzYX0EHGuXgHtYvGtpyFE
CLiQaUAYEjmy/VV1/AMfkoANRPIMiaHpgP1VCMQTrxW+19f0e1rda6HDD09H0Jz07dbU/WKfHxV4FGWbP1tFY9WQX18W9pC1r0ZqKNZe9dtk044Cp7QqU
T8L1iUW70SH7Yn5bPxiLrCS/X+F8EHZ1QLPc99XW+1Y0umLZK5ABm931z517JagI7ov1UAmephhhtXpw7ZXIGcEzU2GF2fjoAZFgJyQ23IQK8Lldhk
qhejz1jWue/q2qFvywv34VNU02uQRSsPH9b4QnALC3QH0Su0ZyWol+pwn0I5pPo9IM8vYrTlUyTNBQZiYQottLq7zy86iMatDKntTpsZQ3LYH4E8Zg40N
y7j77wRVD+5yCZtVfPs+0upEvBMT9Zd5Cd9A14R5iZHYxFOcnkiFZLRApA9xUlFviGTcg4xvsh/e9CY0zdh66hrcV1Nim0mKB1V0nZcnvhpvpYQKRCIH
ZKEeknzRzy1J3Nm+MhwDZ5K5RRxs89Zf7rSBh34VOMt23PFERMrUdkRMMX1ymqfapqmtHqLQ13H5N98o89BZfofLVR/aLaQrW

```

Figure 07

```

%$ -m security
C:\%$%geriuiwhrf
C:\%$%$
tasksche.exe
CloseHandle
WriteFile
CreateFileA
CreateProcessA
http://www.iuqerfsodp9ifajaposdfjhgosurijfaewrgwea.com
!This program cannot be run in DOS mode.
=j&6LZ66LA??~
f""D~*~T
V22dN:::t
o%$Jr..%$
&6Lj66LZ??~A
""Df*~T~
;22dV:::tN
%$Jo..%$
&Lj&6LZ6?~A?
"Df""*~T~*

```

Figure 08

```

tasksche.exe
TaskStart
icacLS . /grant Everyone:F /T /C /Q
attrib +h .
WNCry2017
GetNativeSystemInfo
.7AVexception@@
incompatible version
buffer error

```

Figure 09

PEStudio Report:

We can bring this executable **Ransomware.wannacry.exe** file inside the pestudio.

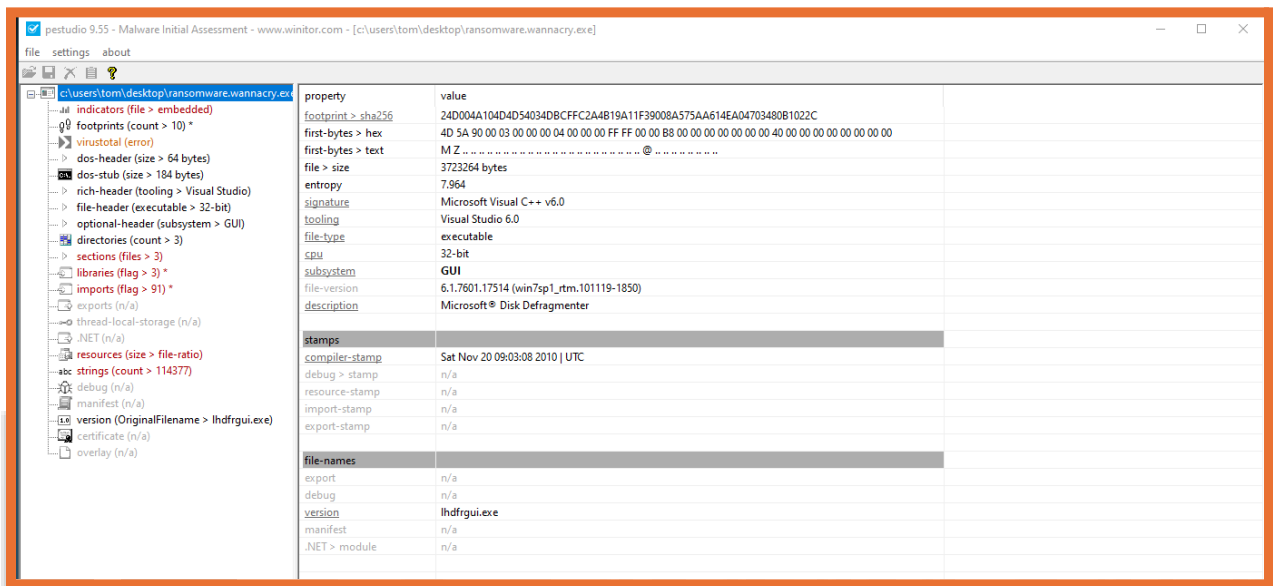


Figure 10

From the above screenshot (Figure 10), we see there are some overviews of the file, which shows this is a 32-bit CPU architecture, file size, original file name, and so on.

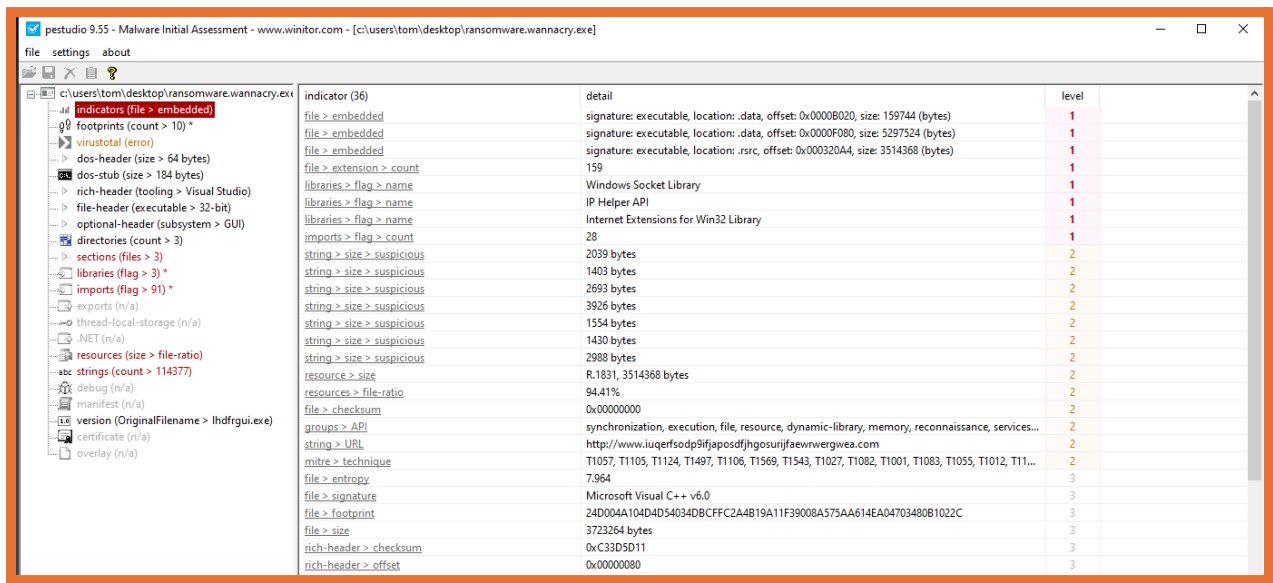


Figure 11

In the indicators section (Figure 11), we can see any significant indicator, and it turns out that pestudio identifies that there are apparently three others packed inside of this

first stage executable. That also gives us the byte offset of these locations. It also identifies the URL string that we found in my floss report.

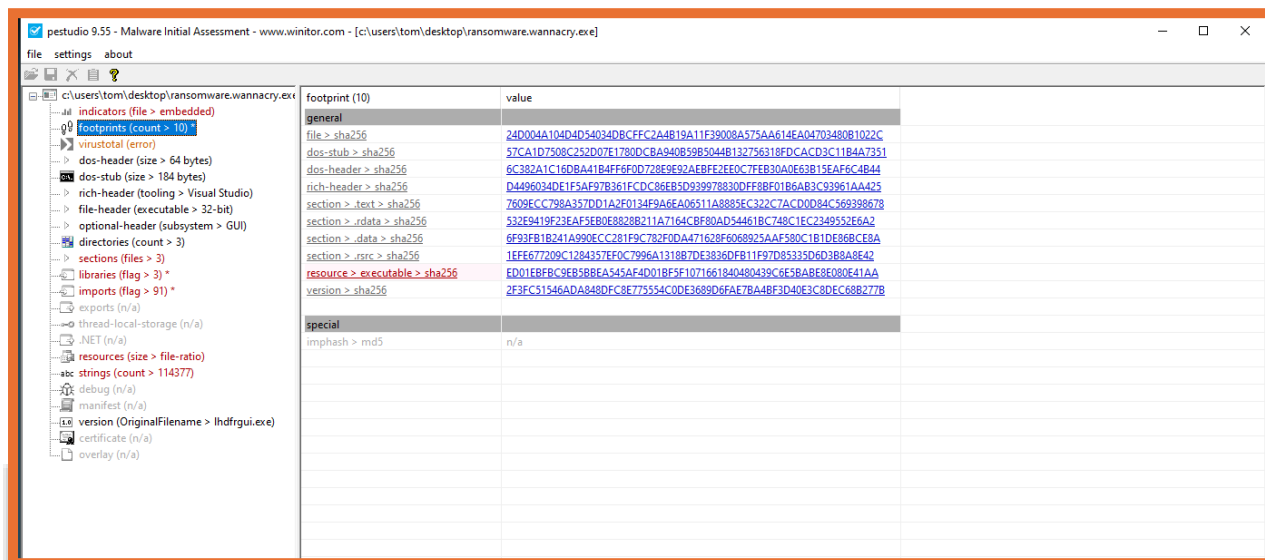


Figure 12

In the **footprints** section (Figure 12), we see **sha256** hash value for inside packed executables with the main file.

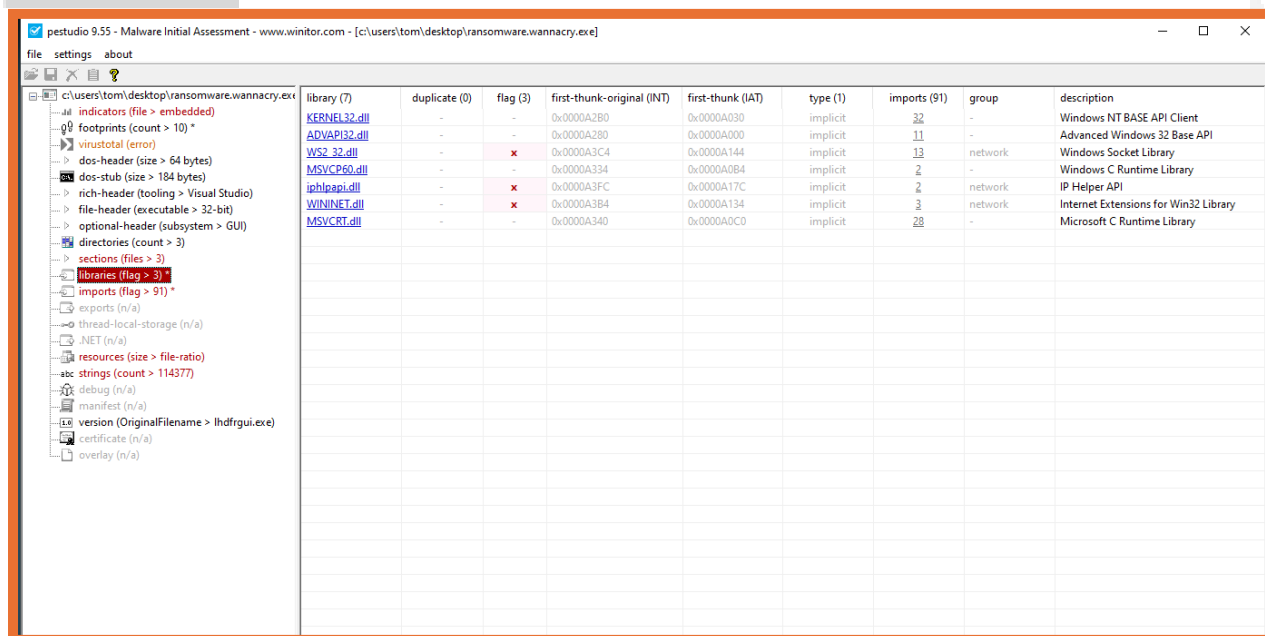


Figure 13

In the **libraries** section (Figure 13), there are Windows Socket, IP Helper, and Internet Extensions for Win32 marked as blacklisted.

| imports (91) | flag (28) | first-thunk-original (INT) | first-thunk (IAT) | hint | group (16) | technique (8) | type (1) |
|-----------------------------|-----------|----------------------------|-------------------|--------------|-----------------|---|----------|
| StartServiceCtrlDispatcherA | x | 0x0000A6F6 | 0x0000A6F6 | 586 (0x024A) | services | - | implicit |
| ChangeServiceConfig2A | x | 0x0000A6C0 | 0x0000A6C0 | 52 (0x0034) | services | T1569 System Services | implicit |
| CreateServiceA | x | 0x0000A688 | 0x0000A688 | 100 (0x0064) | services | T1543 Create or Modify System Proc... | implicit |
| QueryPerformanceFrequency | x | 0x0000A43A | 0x0000A43A | 676 (0x02A4) | reconnaissance | - | implicit |
| 3 (closesocket) | x | 0x80000003 | 0x80000003 | 0 (0x0000) | network | - | implicit |
| 16 (recv) | x | 0x80000010 | 0x80000010 | 0 (0x0000) | network | - | implicit |
| 19 (send) | x | 0x80000013 | 0x80000013 | 0 (0x0000) | network | - | implicit |
| 8 (htonl) | x | 0x80000008 | 0x80000008 | 0 (0x0000) | network | - | implicit |
| 14 (ntohl) | x | 0x8000000E | 0x8000000E | 0 (0x0000) | network | - | implicit |
| 115 (WSAStartup) | x | 0x80000073 | 0x80000073 | 0 (0x0000) | network | - | implicit |
| 12 (inet_ntoa) | x | 0x8000000C | 0x8000000C | 0 (0x0000) | network | - | implicit |
| 10 (ioctlsocket) | x | 0x8000000A | 0x8000000A | 0 (0x0000) | network | - | implicit |
| 18 (select) | x | 0x80000012 | 0x80000012 | 0 (0x0000) | network | - | implicit |
| 9 (htons) | x | 0x80000009 | 0x80000009 | 0 (0x0000) | network | - | implicit |
| 23 (socket) | x | 0x80000017 | 0x80000017 | 0 (0x0000) | network | - | implicit |
| 4 (connect) | x | 0x80000004 | 0x80000004 | 0 (0x0000) | network | - | implicit |
| 11 (inet_addr) | x | 0x8000000B | 0x8000000B | 0 (0x0000) | network | - | implicit |
| GetAdaptersInfo | x | 0x0000A792 | 0x0000A792 | 28 (0x001C) | network | - | implicit |
| InternetOpenA | x | 0x0000A7DC | 0x0000A7DC | 146 (0x0092) | network | - | implicit |
| InternetOpenUrlA | x | 0x0000A7C8 | 0x0000A7C8 | 147 (0x0093) | network | - | implicit |
| InternetCloseHandle | x | 0x0000A7B2 | 0x0000A7B2 | 105 (0x0069) | network | - | implicit |
| MoveFileExA | x | 0x0000A576 | 0x0000A576 | 623 (0x026F) | file | T1105 Remote File Copy | implicit |
| GetCurrentThreadId | x | 0x0000A524 | 0x0000A524 | 326 (0x0146) | execution | T1057 Process Discovery | implicit |
| GetCurrentThread | x | 0x0000A53A | 0x0000A53A | 325 (0x0145) | execution | - | implicit |
| CryptGenRandom | x | 0x0000A650 | 0x0000A650 | 150 (0x0096) | cryptography | T1027 Obfuscated Files or Information | implicit |
| CryptAcquireContextA | x | 0x0000A638 | 0x0000A638 | 133 (0x0085) | cryptography | T1027 Obfuscated Files or Information | implicit |
| rand | x | 0x0000A824 | 0x0000A824 | 678 (0x02A6) | cryptography | T1027 Obfuscated Files or Information | implicit |
| srand | x | 0x0000A852 | 0x0000A852 | 692 (0x02B4) | cryptography | T1027 Obfuscated Files or Information | implicit |
| WaitForSingleObject | - | 0x0000A4F6 | 0x0000A4F6 | 912 (0x0390) | synchronization | - | implicit |
| InterlockedIncrement | - | 0x0000A50C | 0x0000A50C | 556 (0x022C) | synchronization | - | implicit |
| InterlockedDecrement | - | 0x0000A4BE | 0x0000A4BE | 552 (0x0228) | synchronization | - | implicit |
| EnterCriticalSection | - | 0x0000A4A6 | 0x0000A4A6 | 152 (0x0098) | synchronization | - | implicit |
| LeaveCriticalSection | - | 0x0000A4BE | 0x0000A4BE | 593 (0x0251) | synchronization | - | implicit |
| InitializeCriticalSection | - | 0x0000A472 | 0x0000A472 | 547 (0x0223) | synchronization | - | implicit |
| RegisterServiceCtrlHandlerA | - | 0x0000A6D8 | 0x0000A6D8 | 524 (0x020C) | services | T1106 Execution through API | implicit |
| SetServiceStatus | - | 0x0000A6AC | 0x0000A6AC | 580 (0x0244) | services | T1543 Create or Modify System Proc... | implicit |

Figure 14

In the **imports** section (Figure 14), we are sorted by **flag**, we saw some cryptography groups there which we anticipate that this is a ransomware binary that calls the APIs that are imported into the binary are going to deal with crypto. So this is likely used for the encryption routine itself.

Then we saw some open, close, send socket imported which seems to be opening ports and maybe attempting remote connections to different ports.

There are **InternetOpenA**, and **InternetOpenUrlA** imported, which are API calls to download some files or create a network connection.

Based on two interesting files imported, which are **CreateServiceA**, **ChangeServiceConfig2A**, we can anticipate that there are some kind of service creation with this binary, and perhaps that's a persistence mechanism.

Dynamic Analysis

Running the **Ransomware.wannacry.exe** in two different stages

1. While Network is Connected
2. No Network Connection

Normally, when we see some weird web URL inside the string output, we assume the executable will download some file and run the malicious payload from this URL. However, for this sample **Ransomware.wannacry.exe** file behavior is different.

So **Ransomware.wannacry.exe** will attempt to connect a URL (**hxxp[://]www[.]iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea[.]com**) and if it does connect this URL, it does not trigger the payload. On the other hand, if there is no internet connection, it will trigger the payload and encrypt all of our files.

To test this, we can open up the fake internet in our **Remnux** machine and monitor the traffic from **Wireshark**. Then run the **Ransomware.wannacry.exe** as an administrator. Running this malicious file, we see a TCP handshake, then an HTTP packet that makes a full request URL to **hxxp[://]www[.]iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea[.]com**, our fake internet response with 200 OK for this (Figure 15). Now if we go back to our Windows machine, we can see nothing executed.

Based on our analysis, WannaCry (Ransomware.wannacry.exe) does not execute of the 200 OK to the callback URL it is attempting to.

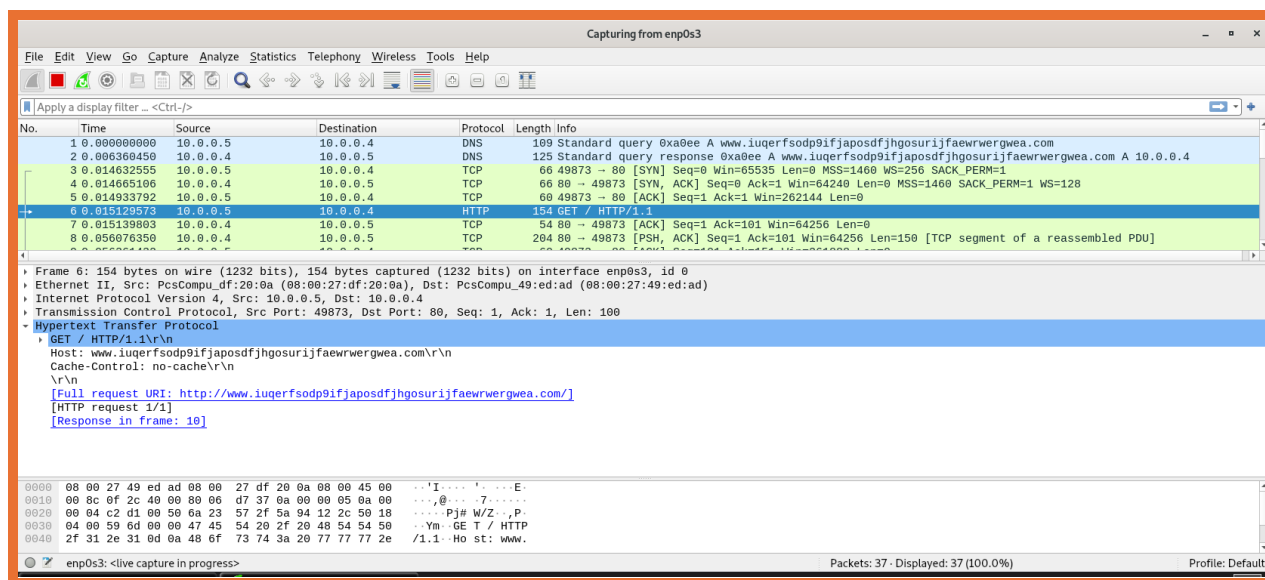


Figure 15

Now, if we stop the internet connection, and run the **Ransomware.wannacry.exe** file one more time, WannaCry is attempting to contact that URL, it fails and then the rest of the binary is executed. (Figure 16)

So the condition necessary to get the sample to detonate we actually need to stop running the internet.



Figure 16

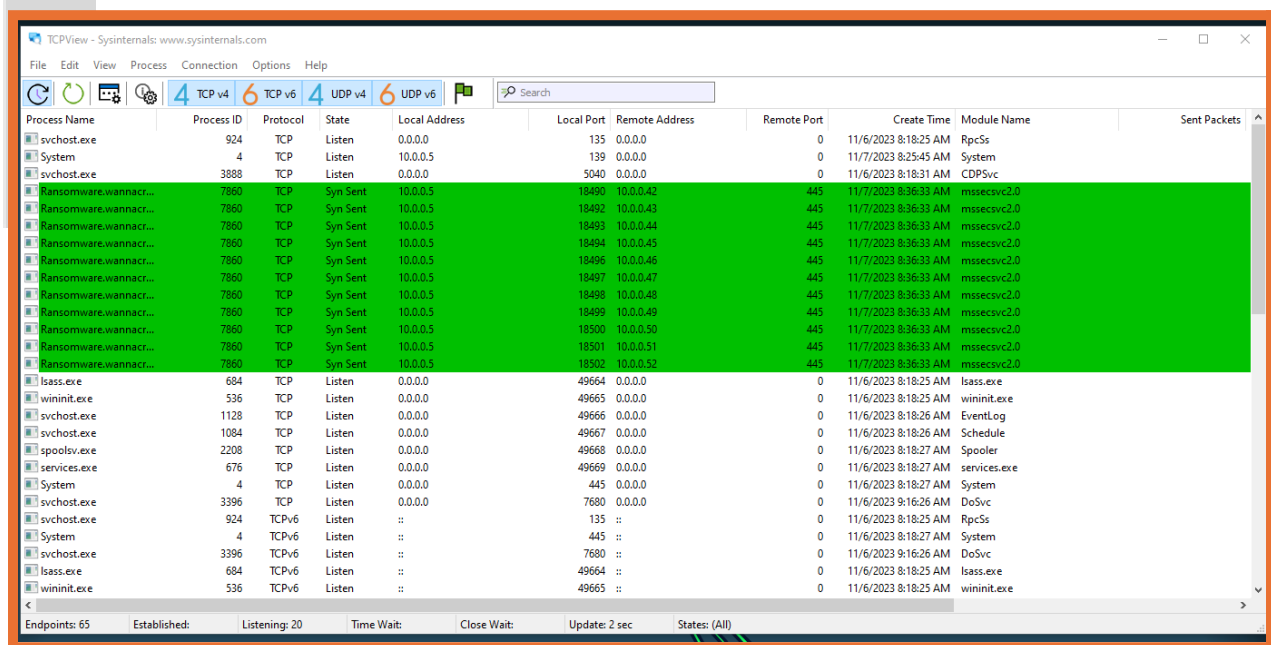
TCPView Report:

For our analysis, we will use a tool called TCPView inside the endpoint itself to gather those network artifacts.

After opening up the TCPView, we run the **Ransomware.wannacry.exe**. Now as we run this file, we notice there are several traffic going out to different remote addresses starting with **10.0.0.X** on port **445** which means that there is no real address connectivity present to be able to make a connection to that, and seems like an auto-assigned IP address. (Figure 17)

Point is noticing all the network connections going out on port **445**, apparently, those IPs are subnet networks, and it will be the different hosts around our network. We discovered how **WannaCry** is trying to propagate itself through the network.

So we can state this is a ransomware binary, but it also has worm capabilities.



| Process Name | Process ID | Protocol | State | Local Address | Local Port | Remote Address | Remote Port | Create Time | Module Name | Sent Packets |
|-------------------------|------------|----------|----------|---------------|------------|----------------|-------------|----------------------|--------------|--------------|
| svchost.exe | 924 | TCP | Listen | 0.0.0.0 | 135 | 0.0.0.0 | 0 | 11/6/2023 8:18:25 AM | RpcSs | |
| System | 4 | TCP | Listen | 10.0.0.5 | 139 | 0.0.0.0 | 0 | 11/7/2023 8:25:45 AM | System | |
| svchost.exe | 3888 | TCP | Listen | 0.0.0.0 | 5040 | 0.0.0.0 | 0 | 11/6/2023 8:18:31 AM | CDPSvc | |
| Ransomware.wannacry.exe | 7860 | TCP | Syn Sent | 10.0.0.5 | 18490 | 10.0.0.42 | 445 | 11/7/2023 8:36:33 AM | mssecsv2.0 | |
| Ransomware.wannacry.exe | 7860 | TCP | Syn Sent | 10.0.0.5 | 18492 | 10.0.0.43 | 445 | 11/7/2023 8:36:33 AM | mssecsv2.0 | |
| Ransomware.wannacry.exe | 7860 | TCP | Syn Sent | 10.0.0.5 | 18493 | 10.0.0.44 | 445 | 11/7/2023 8:36:33 AM | mssecsv2.0 | |
| Ransomware.wannacry.exe | 7860 | TCP | Syn Sent | 10.0.0.5 | 18494 | 10.0.0.45 | 445 | 11/7/2023 8:36:33 AM | mssecsv2.0 | |
| Ransomware.wannacry.exe | 7860 | TCP | Syn Sent | 10.0.0.5 | 18496 | 10.0.0.46 | 445 | 11/7/2023 8:36:33 AM | mssecsv2.0 | |
| Ransomware.wannacry.exe | 7860 | TCP | Syn Sent | 10.0.0.5 | 18497 | 10.0.0.47 | 445 | 11/7/2023 8:36:33 AM | mssecsv2.0 | |
| Ransomware.wannacry.exe | 7860 | TCP | Syn Sent | 10.0.0.5 | 18498 | 10.0.0.48 | 445 | 11/7/2023 8:36:33 AM | mssecsv2.0 | |
| Ransomware.wannacry.exe | 7860 | TCP | Syn Sent | 10.0.0.5 | 18499 | 10.0.0.49 | 445 | 11/7/2023 8:36:33 AM | mssecsv2.0 | |
| Ransomware.wannacry.exe | 7860 | TCP | Syn Sent | 10.0.0.5 | 18500 | 10.0.0.50 | 445 | 11/7/2023 8:36:33 AM | mssecsv2.0 | |
| Ransomware.wannacry.exe | 7860 | TCP | Syn Sent | 10.0.0.5 | 18501 | 10.0.0.51 | 445 | 11/7/2023 8:36:33 AM | mssecsv2.0 | |
| Ransomware.wannacry.exe | 7860 | TCP | Syn Sent | 10.0.0.5 | 18502 | 10.0.0.52 | 445 | 11/7/2023 8:36:33 AM | mssecsv2.0 | |
| lsass.exe | 684 | TCP | Listen | 0.0.0.0 | 49664 | 0.0.0.0 | 0 | 11/6/2023 8:18:25 AM | lsass.exe | |
| wininit.exe | 536 | TCP | Listen | 0.0.0.0 | 49665 | 0.0.0.0 | 0 | 11/6/2023 8:18:25 AM | wininit.exe | |
| svchost.exe | 1128 | TCP | Listen | 0.0.0.0 | 49666 | 0.0.0.0 | 0 | 11/6/2023 8:18:26 AM | EventLog | |
| svchost.exe | 1084 | TCP | Listen | 0.0.0.0 | 49667 | 0.0.0.0 | 0 | 11/6/2023 8:18:26 AM | Schedule | |
| spoolsv.exe | 2208 | TCP | Listen | 0.0.0.0 | 49668 | 0.0.0.0 | 0 | 11/6/2023 8:18:27 AM | Spooler | |
| services.exe | 676 | TCP | Listen | 0.0.0.0 | 49669 | 0.0.0.0 | 0 | 11/6/2023 8:18:27 AM | services.exe | |
| System | 4 | TCP | Listen | 0.0.0.0 | 445 | 0.0.0.0 | 0 | 11/6/2023 8:18:27 AM | System | |
| svchost.exe | 3396 | TCP | Listen | 0.0.0.0 | 7680 | 0.0.0.0 | 0 | 11/6/2023 9:16:26 AM | DoSvc | |
| svchost.exe | 924 | TCPv6 | Listen | :: | 135 | :: | 0 | 11/6/2023 8:18:25 AM | RpcSs | |
| System | 4 | TCPv6 | Listen | :: | 445 | :: | 0 | 11/6/2023 8:18:27 AM | System | |
| svchost.exe | 3396 | TCPv6 | Listen | :: | 7680 | :: | 0 | 11/6/2023 9:16:26 AM | DoSvc | |
| lsass.exe | 684 | TCPv6 | Listen | :: | 49664 | :: | 0 | 11/6/2023 8:18:25 AM | lsass.exe | |
| wininit.exe | 536 | TCPv6 | Listen | :: | 49665 | :: | 0 | 11/6/2023 8:18:25 AM | wininit.exe | |

Figure 17

After further investigation inside the TCPView, we see that there is a new process that spawned **taskhsvc.exe**, which opens up a listening port on 9050. (Figure 18)

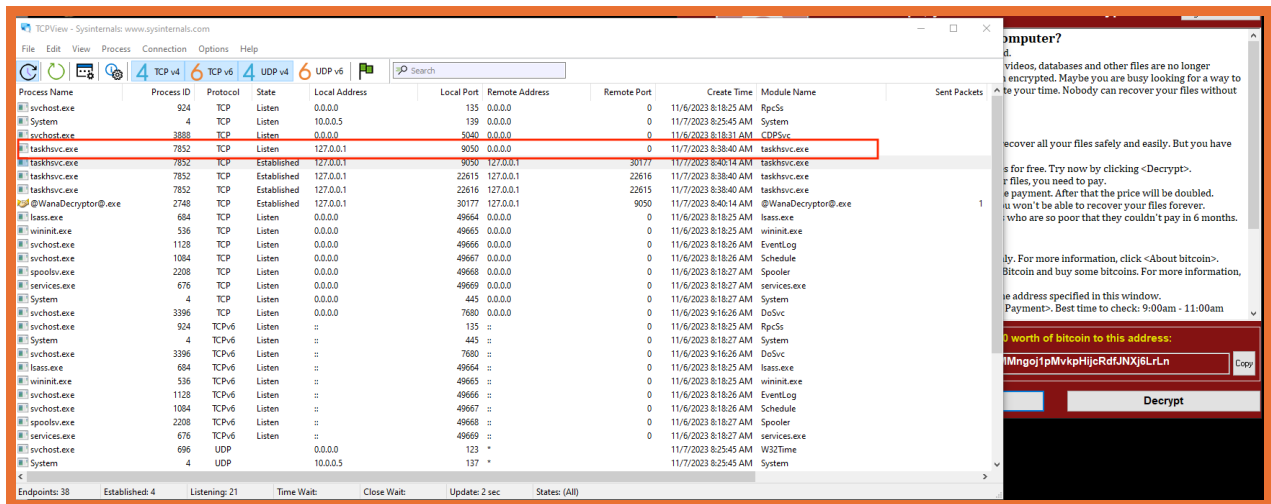


Figure 18

Eccentricity

Host-Based Indicator (procmon.exe) Report:

Inside the **procmon.exe**, we can create two (2) filters to see any **Ransomware.wannacry.exe** related events that were created by running this file as “Run as administrator”.

Filters are (Figure 19)

Process Name is **Ransomware.wannacry.exe**
Operation contains **CreateFile**

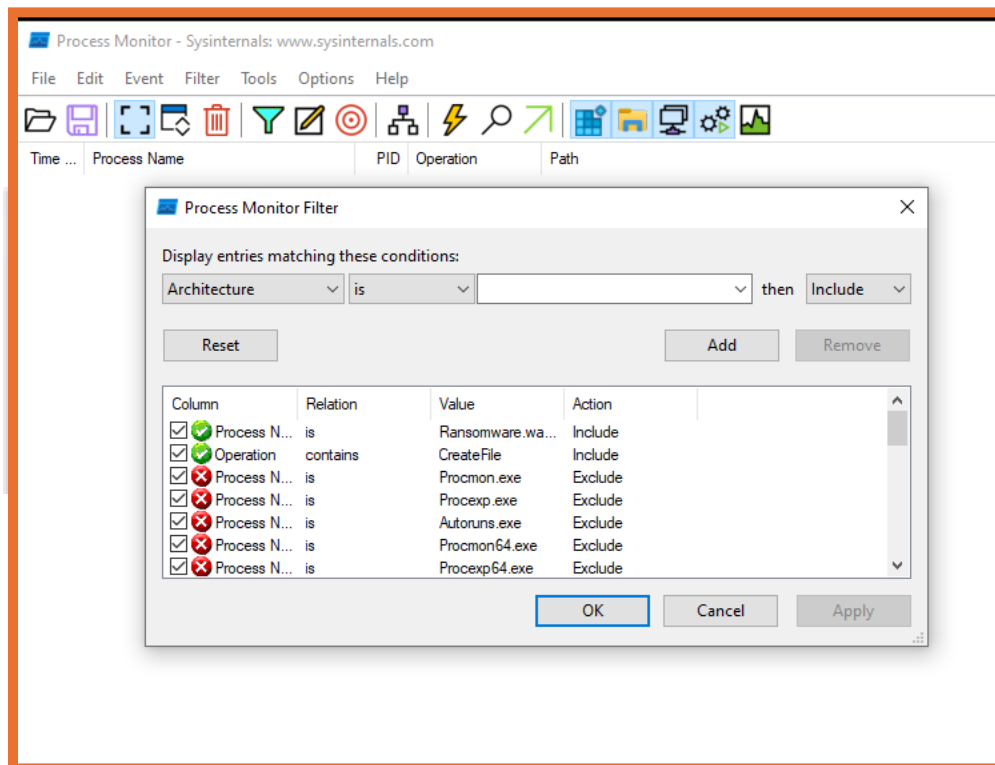


Figure 19

After creating the filter, we ran the file and saw there were several files created. However there are “**tasksche.exe**” file is created in **C:\Windows** directory. If we check the process tree for the “tasksche.exe”, it seems to be spawned from the original **Ransomware.wannacry.exe** binary and run with the argument of “**C:\Windows\tasksche.exe /i**” command. (Figure 20)

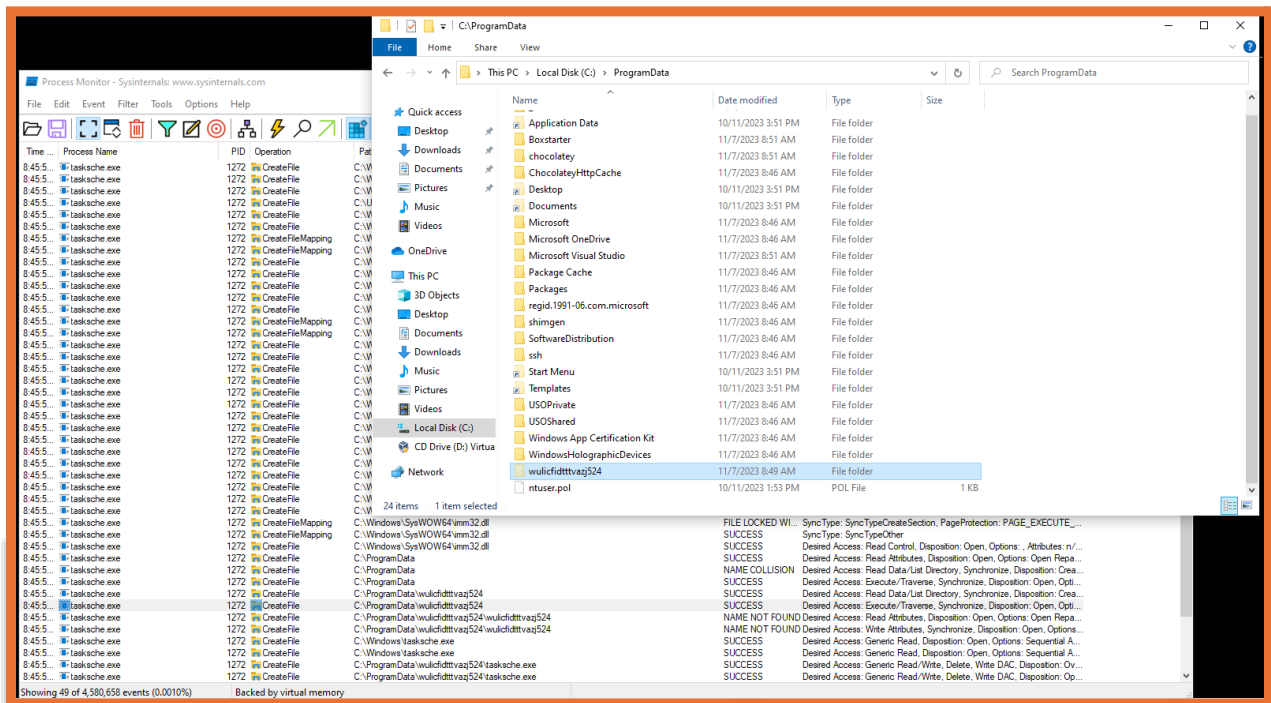


Figure 23

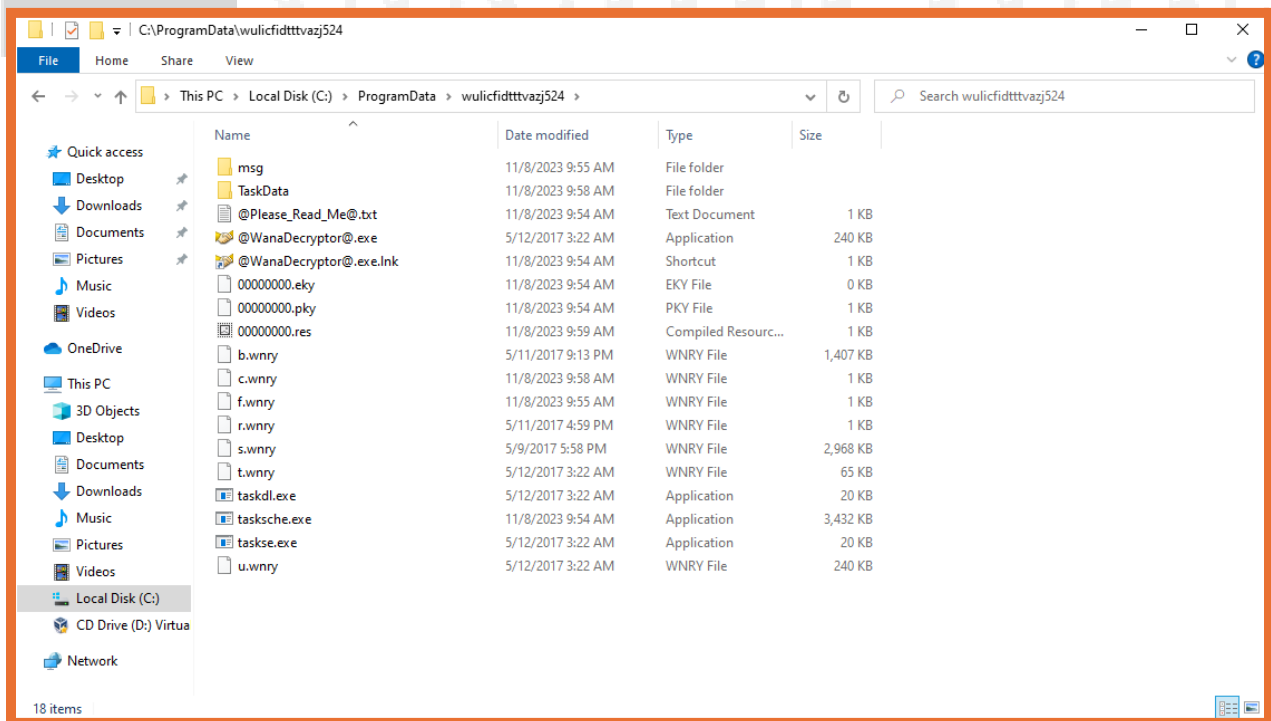


Figure 24

We can also find the **wulicfidttvazj524** services inside the Task Manager (Figure 25). So this is the service that makes it so if you restart your computer, WannaCry kicks back

on and will re-encrypt anything that's added to the host whether that's a USB flash drive or SD card.

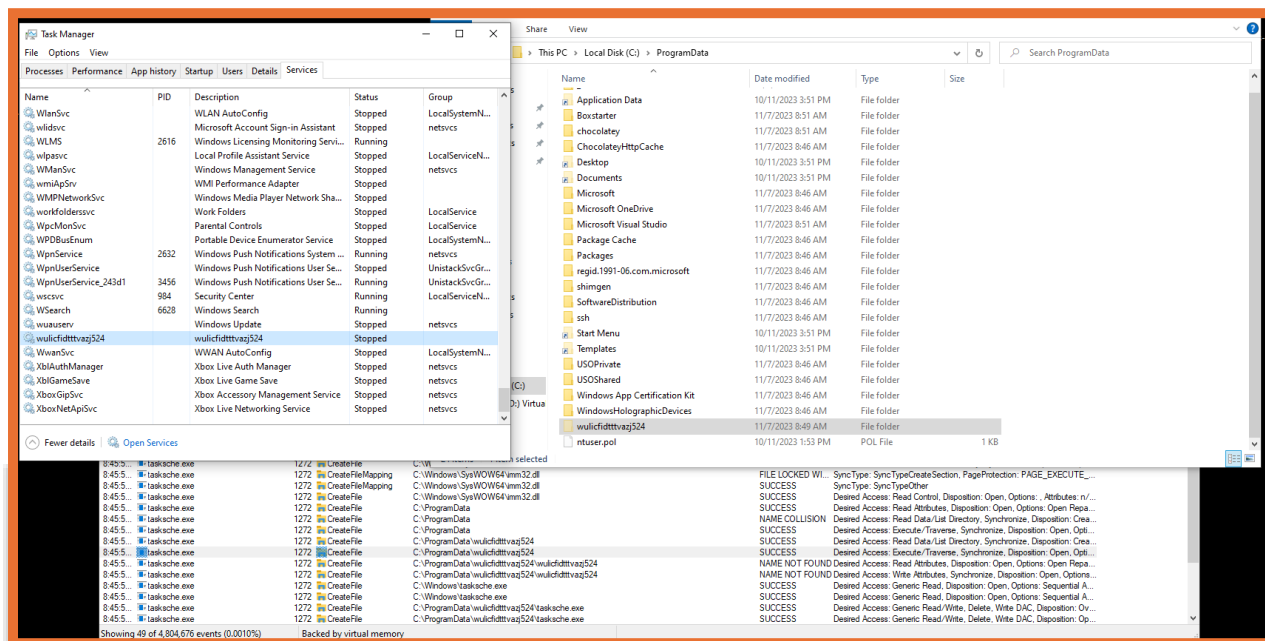


Figure 25

Inside of the **C:\ProgramData\wulicfidttvazj524**, we can identify thee (3) packed executable which was spawned from the original file **Ransomware.wannacry.exe**. Running the PowerShell command, we can have the hash value for those executables. (Figure 26)

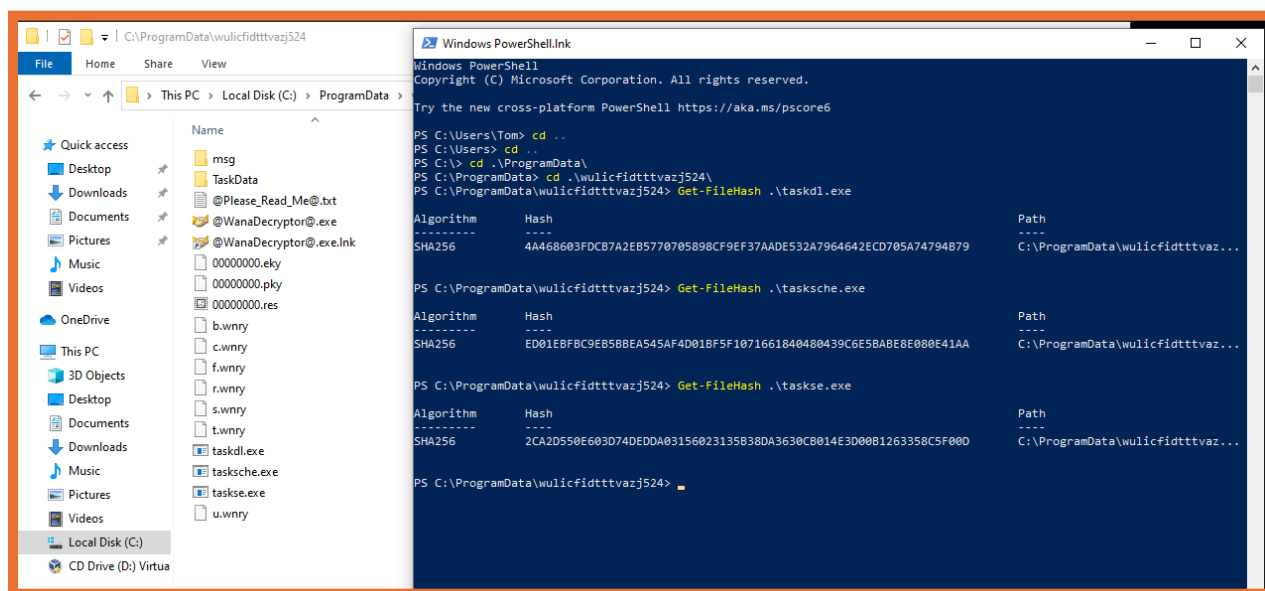


Figure 26

VirusTotal Verdict for packed executables:

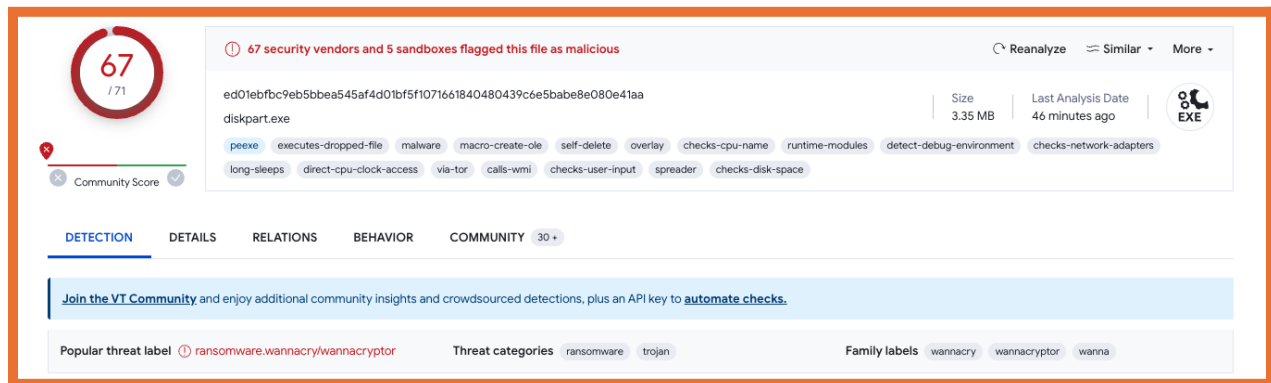


Figure 27: FileName: `tasksche.exe`

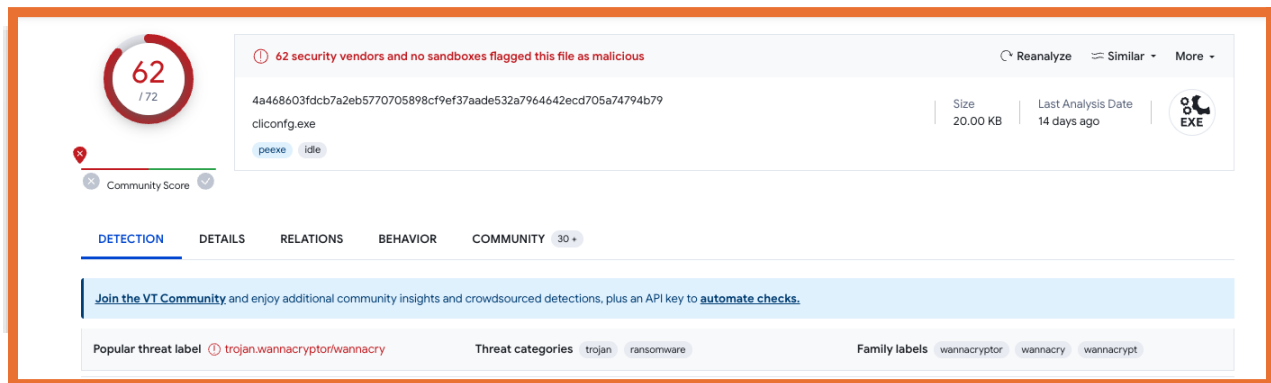


Figure 28: FileName: `taskdl.exe`

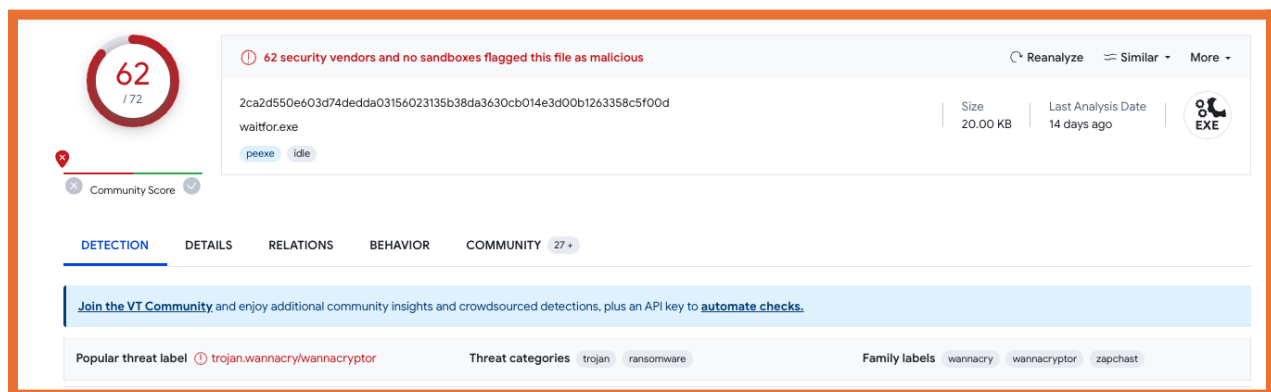


Figure 29: FileName: `taskse.exe`

Advanced Static Analysis

For further analysis, we are going to look at the kill switch function inside the “**Cutter**” for the binary of **Ransomware.wannacry.exe**.

After opening up the file, we see the overview of the binary (**Ransomware.wannacry.exe**), like the file size, hashes, libraries imported, and so on. (Figure 30)

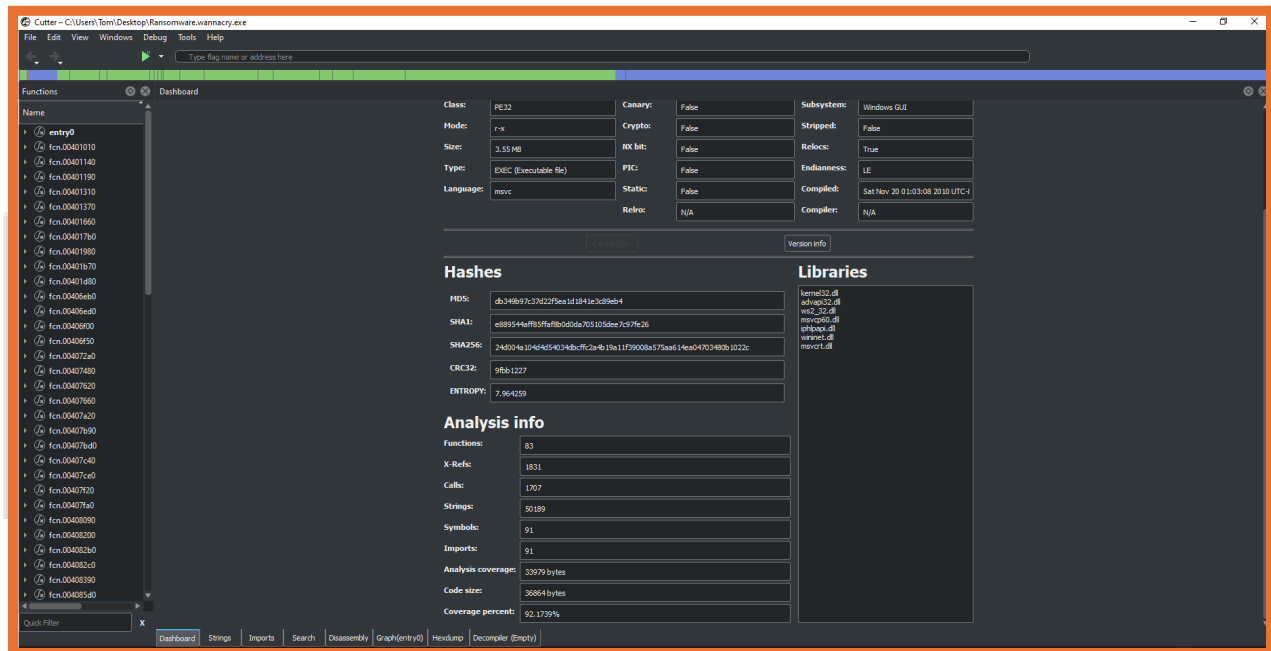


Figure 30

Then if we look into the **main function** in the graph tab, we can see the string reference to the URL (**hxxp[:]//www[.]iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea[.]com**) is loaded into **esi** right at the beginning of the program. So, as the string reference is loaded into the **esi**, there are several arguments are invoked to make an API call. The first API call that we make is “**InternetOpenA**” which is going to prep to open up a handle to a given web resource. In addition, the argument for “**InternetOpenUrlA**” is invoked and pushed to the stack, and among those one of them is **esi**. (Figure 31)

So the URL string (**hxxp[:]//www[.]iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea[.]com**) is pushed onto the stack in order to be used with the “**InternetOpenUrlA**” API call.

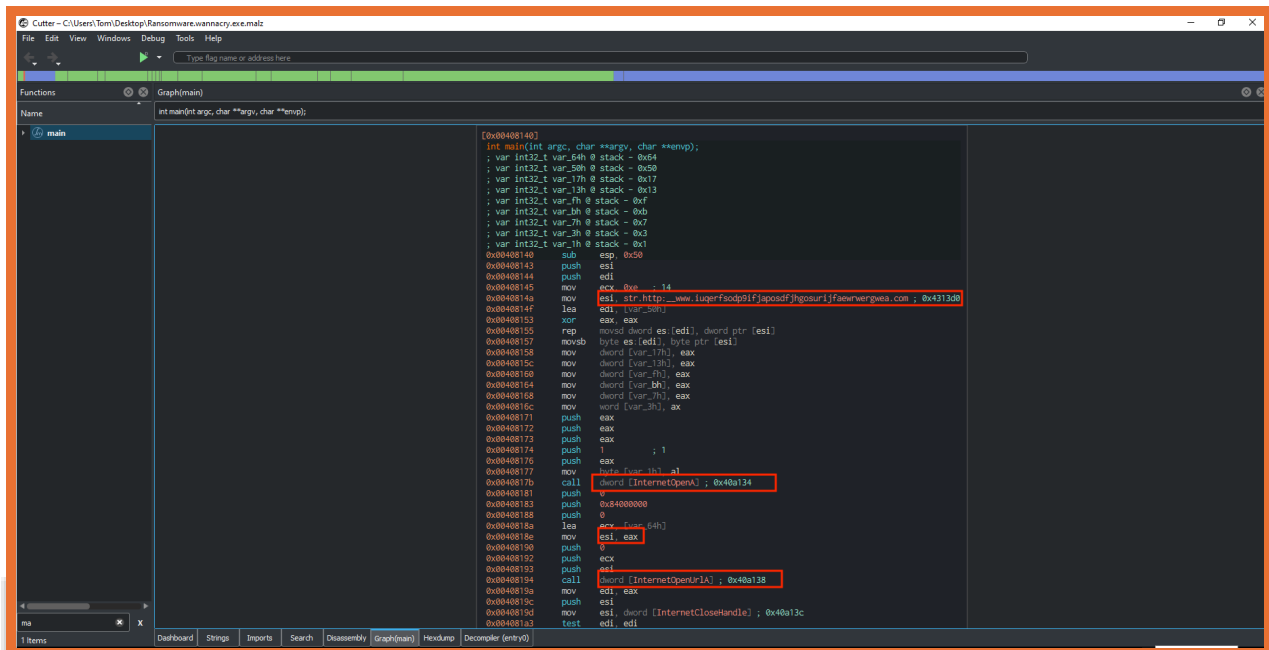


Figure 31

Now if we jump to the Decompiler tab, the decompiler recognizes that the outcome of the “**InternetOpenUrlA**” call is loaded into the **eax** register and then the content of **eax** register will be loaded into the **edi** register. **InternetOpenUrlA** API holds a Boolean value of 1 or 0. (Figure 32)

So, basically **InternetOpenUrlA** will load the URL and make a request to this API, then whatever it returns, either yes or no, it will load into the **eax** register. So either one or zero. Then loaded to edi.

Then if the “InternetOpenUrlA” cannot reach to the internet, then the edi will be 0 and jump to the different location and invoke a function “**fcn_00408090**” (Figure 33). And if it reaches the internet then return as 1 which will return nothing. (F.g 7 (3))

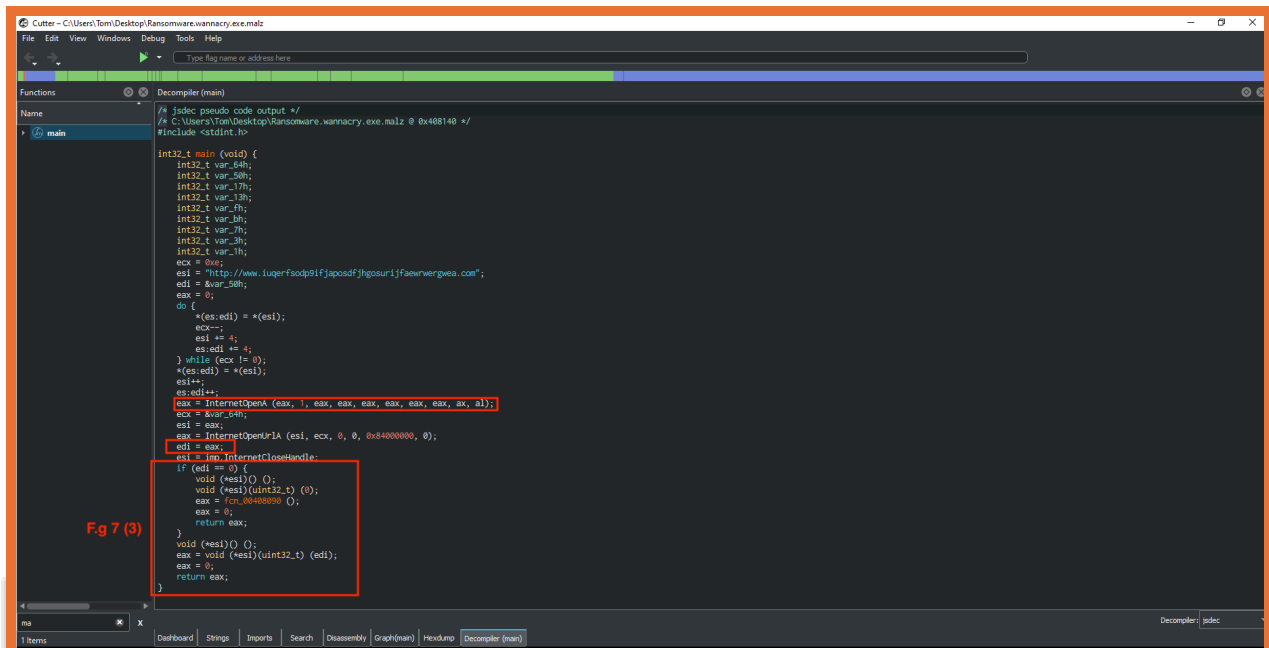
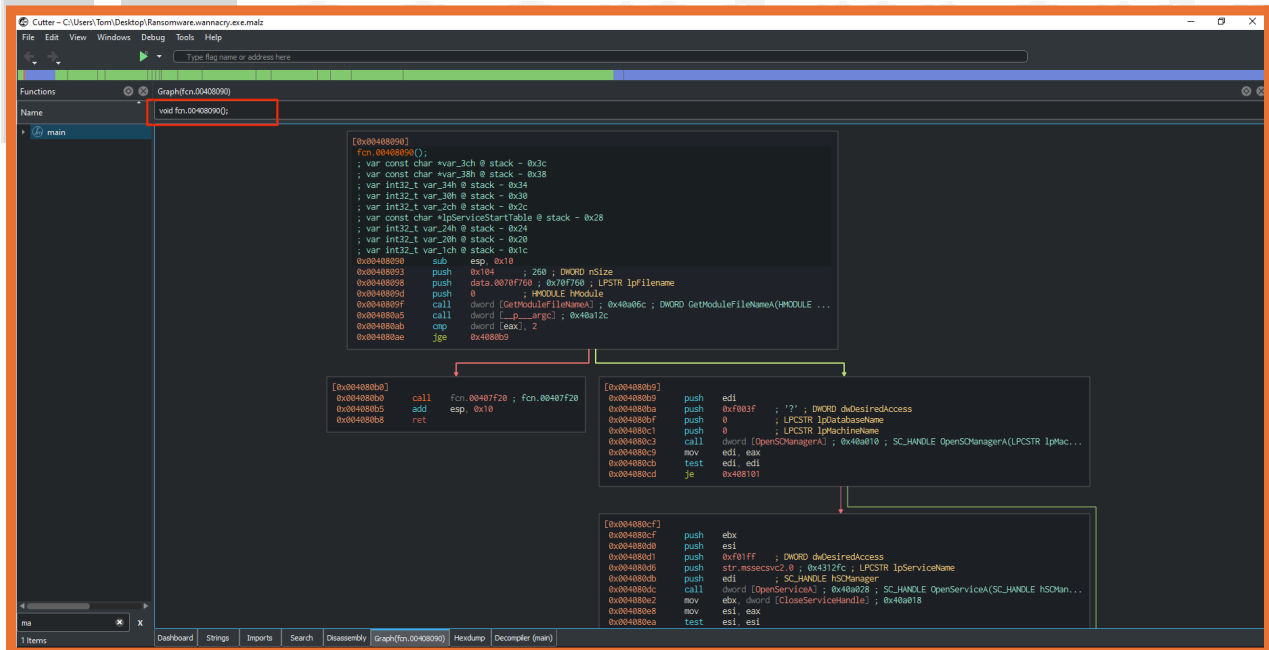


Figure 32



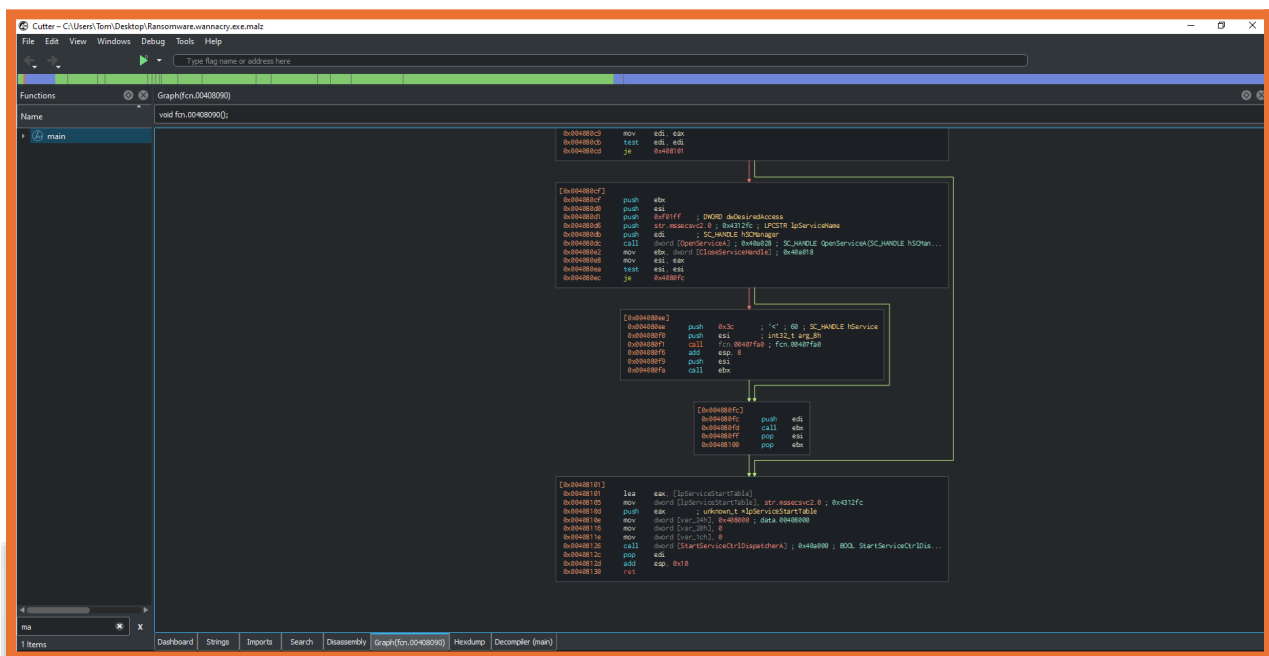


Figure 33: Function “`fcn_00408090`” will invoke, if it cannot reach to the internet.

Now, we can see the if condition, inside the bottom of the graph. (Figure 34)

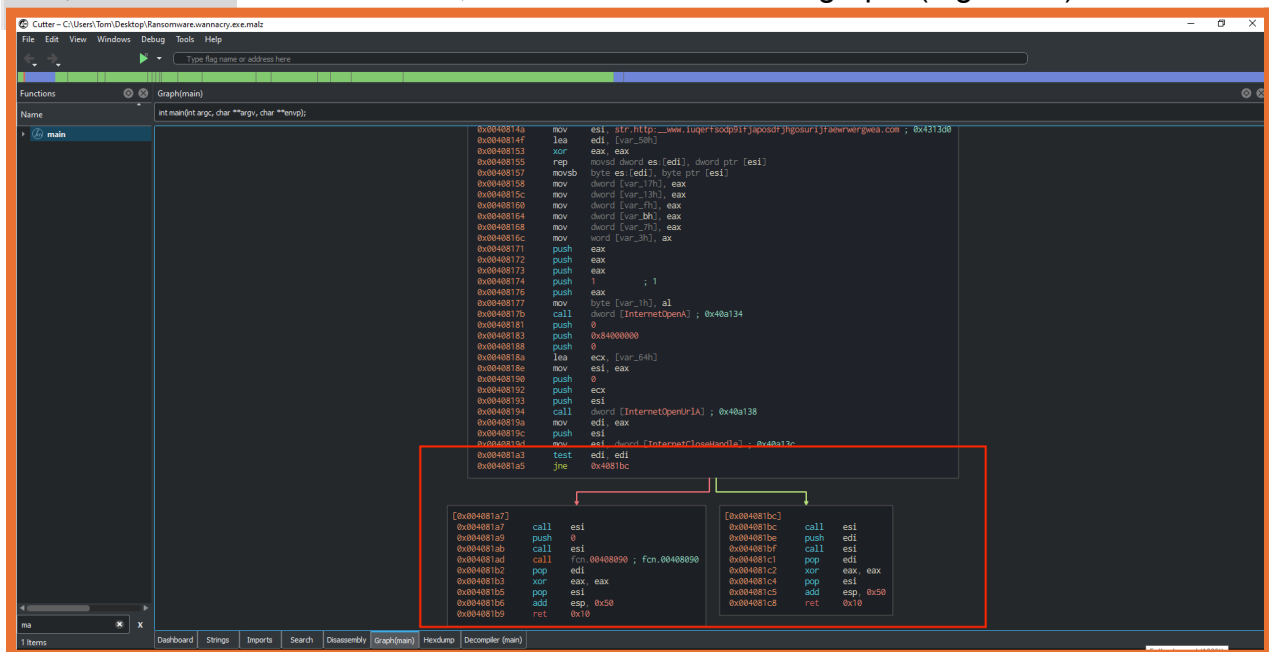


Figure 34

Advanced Dynamic Analysis

For advanced dynamic analysis, we will use the **debugger** application (x32) and try to manipulate the executable (**Ransomware.wannacry.exe**) process even if it gets a result for that URL (**hxxp[://]www[.]iuqerfsodp9ifjaposdfjhgosurijfaewrrergwea[.]com**) that is calling out to.

We can run the fake internet from the Remnux machine and open up the Wireshark program to see the network activity. Then load the file into the debugger. (Figure 35)

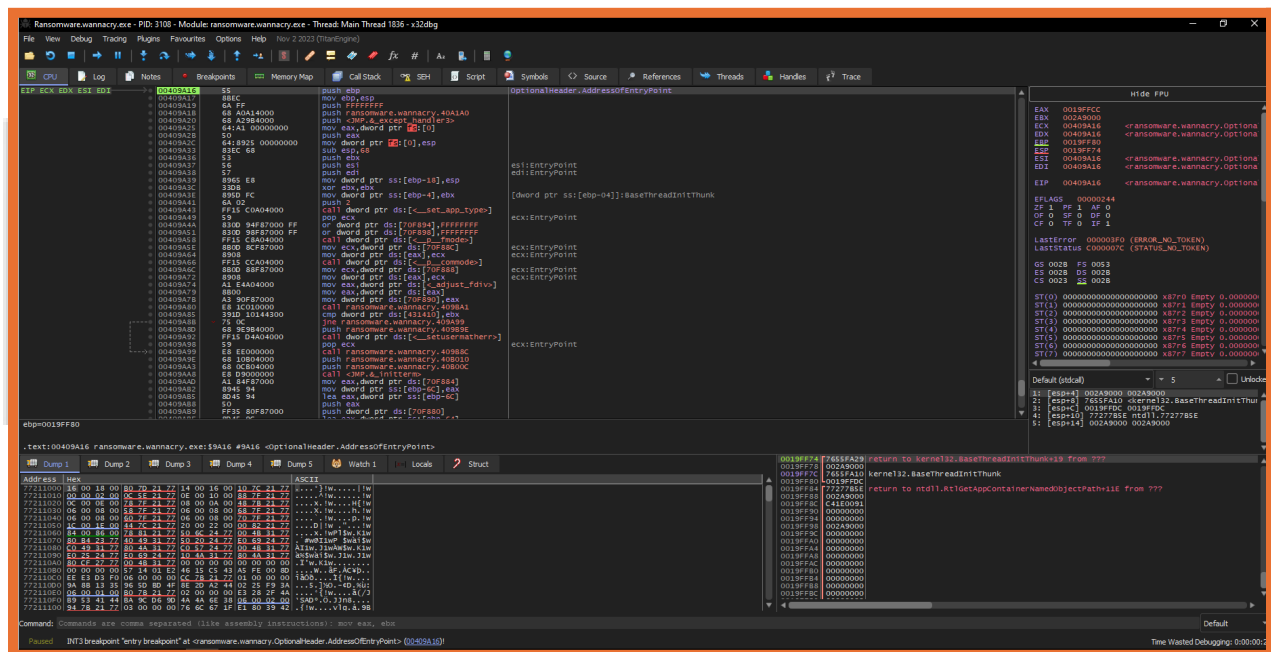


Figure 35

After loading the executable inside the debugger, we are currently at the entry point function, so we want to find the **main function**. To find the main function, press **F9** from the keyboard. Then we can search the URL string and create a break point for this process. (Figure 36).

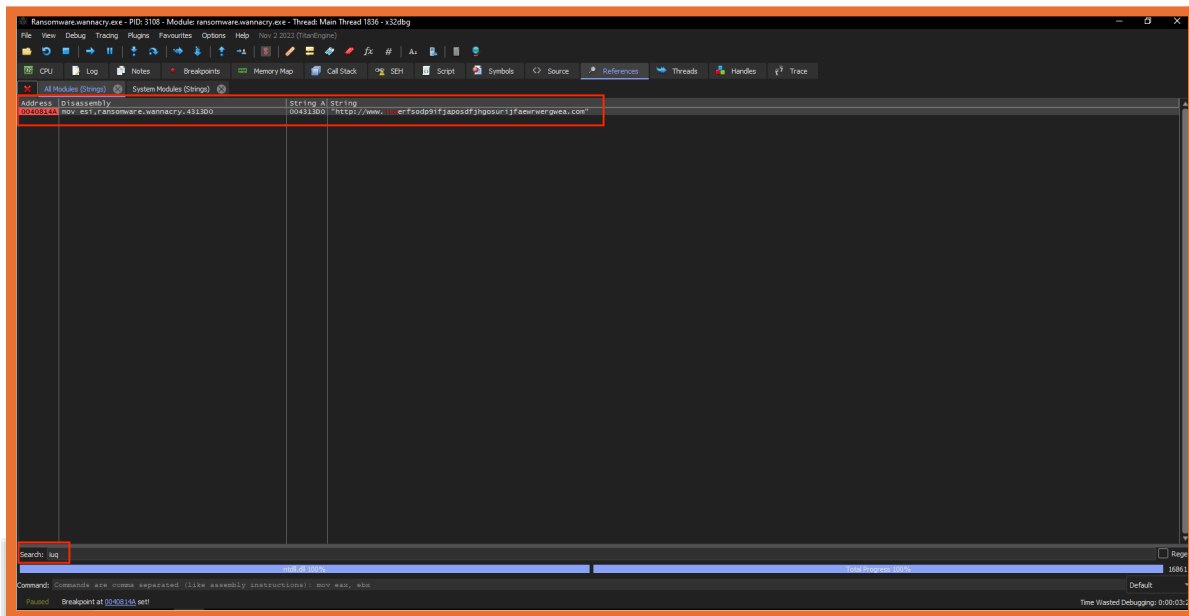


Figure 36

After that, we can we back to the main CPU tab, and press F9 one more time, it will jump to the process where it calls the URL string. Now the URL has been pushed to the stack, we can press **F8** several times to step over these instructions until we get to the point where the zero flags is evaluated, and it says "**jne ransomware.wannacry.4081BC**". (Figure 37)

We can notice that **EFLAG ZE** section is currently 0, cause as the internet is connected to our machine. Now if we double-click click top of the 0, it will change to the 1 and it will think that there is no network connection.

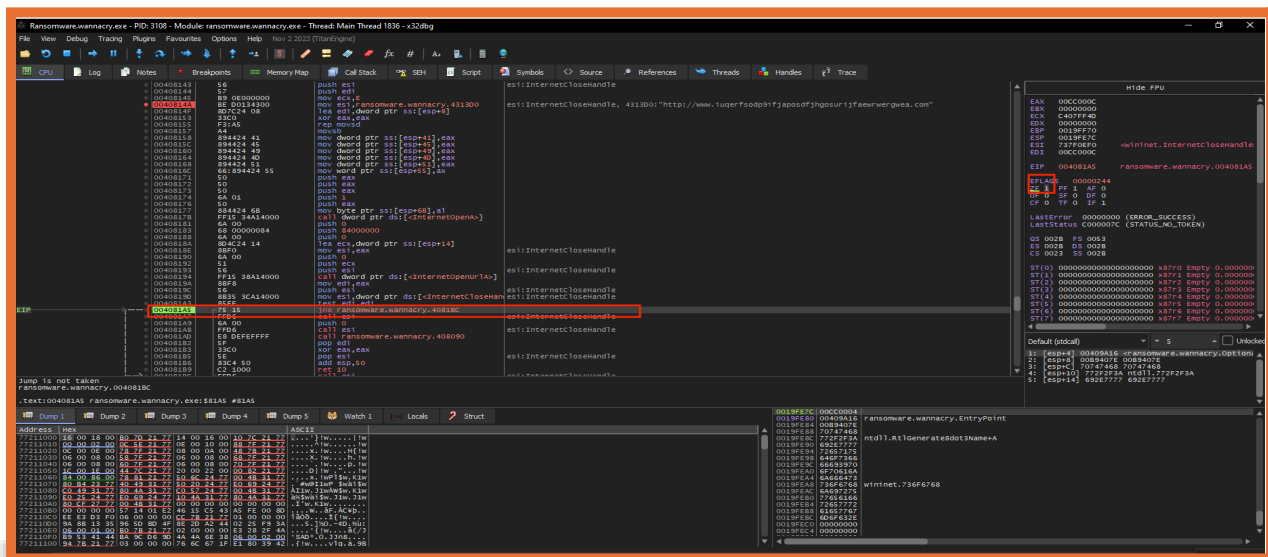


Figure 37

We can see in our host machine, WannaCry is running, and all of our files get encrypted, even if it reach out the URL (hxxp[://]www[.]iuerfsodp9ifjaposdfjhgosurijfaewrwergwea[.]com) with HTTP 200 OK request, which we can get confirm to see inside the Wireshark into our Remnux Machine. (Figure 38)

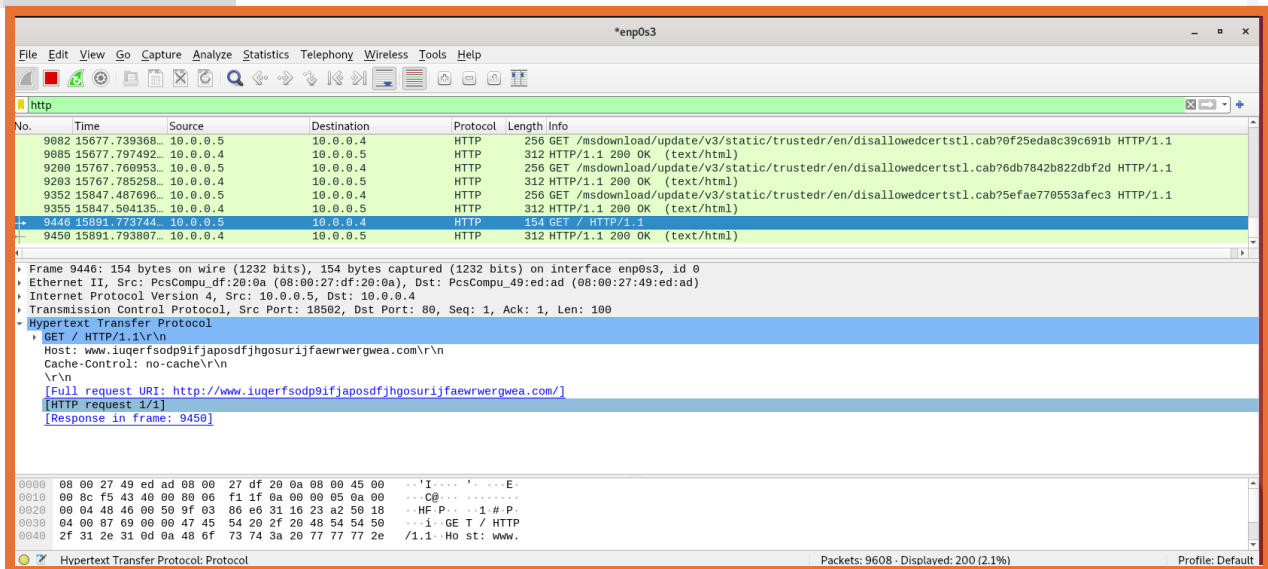


Figure 38