Practical Malware Analysis & Triage

Malware Analysis Report

WannaCry Malware Analysis

August 2023 | Dominic Lynch| v1.0

Table of Contents

Executive Summary

SHA256 hash	A6AA84358130078F9455773AF1E9EF2C7710934F72DF8514C9A62ABEB83D2E81

WannaCry was discovered on May 12, 2017, as a zero-day vulnerability. It is a 32-bit C++ program that is capable of functioning on Windows x64. When launched as an administrator, WannaCry creates persistence, seeks to infect adjacent computers, and encrypts and renames files using a second payload that contains multiple components.

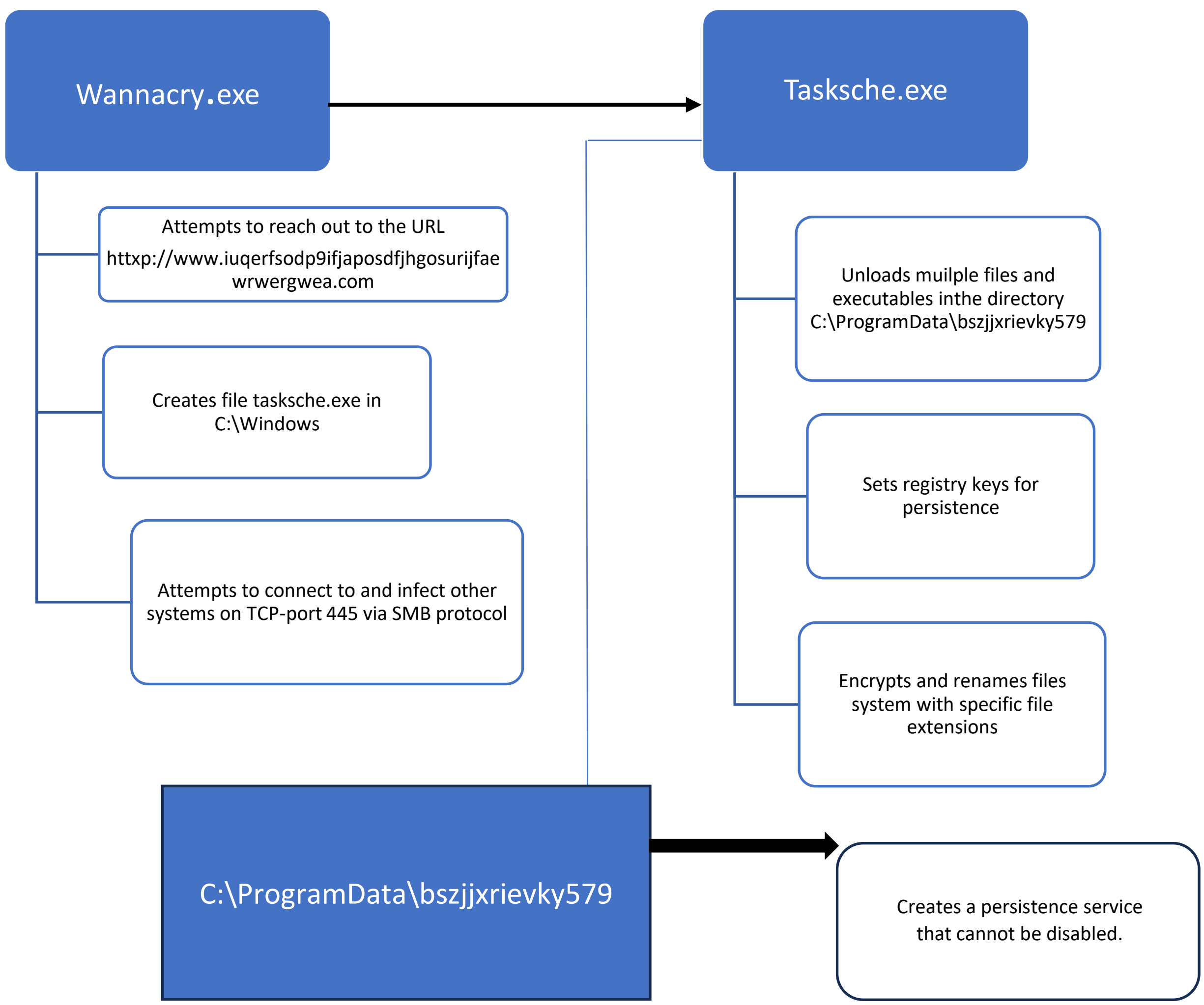
After the detention of the malware cconnection attempts to local systems, a new desktop background stating that files are encrypted, an executable file named "@WanaDecryptor@.exe" on the desktop directory that displays a pop-up message demanding ransom, and files in what appears to be a randomly named folder in the C: Program Data-directory are all signs of infection.

YARA signature rules are included in Appendix A, while screenshots and other specific details on host and network indications are included in Appendix B.

High-Level Technical Summary

WannaCry is made up of two primary components: tasksche.exe, which contains components for persistence, encryption, and file renaming, and WannaCry.exe, which spreads locally throughout the system. The following actions below are taken by the malware:

1. WannaCry.exe when run as an administrator attempts to contact the URL httxp://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com
2. If no connection is performed (and confirm it runs in a simulated environment), the binary creates files in various folders, including tasksche.exe, task.exe and @WanaDecryptor@.exe
3. Persistence is established by setting auto-start registry keys for tasksche.exe in C:\ProgramData\bszjjxrievky579
4. tasksche.exe then encrypts and renames files, selected by their file extensions, on the infected system(s)
5. tasksdl.exe deletes temporary files and taskse.exe starts the recurring popup message demanding ransom.
6. bszjjxrievky579 is created as a persistence service on the system and cannot be disabled by using the task manager program.



Malware Composition

Ransomware.wannacry.exe consists of the following components:

File Name	SHA256 Hash
WannaCry.exe	24D004A104D4D54034DBCFFC2A4B19A11F39008A575AA614EA04703480B1022C
tasksche.exe	ED01EBFBC9EB5BBEA545AF4D01BF5F1071661840480439C6E5BABE8E080E41AA
Taskdl.exe	4A468603FDCB7A2EB5770705898CF9EF37AADE532A7964642ECD705A74794B79
Taskse.exe	2CA2D550E603D74DEDDA03156023135B38DA3630CB014E3D00B1263358C5F00D

Fig 1: All hashes of process created by the binary.

Basic Static Analysis

File Hashes

The following below table shows the Ransomware.wannacry.exe file hashes.

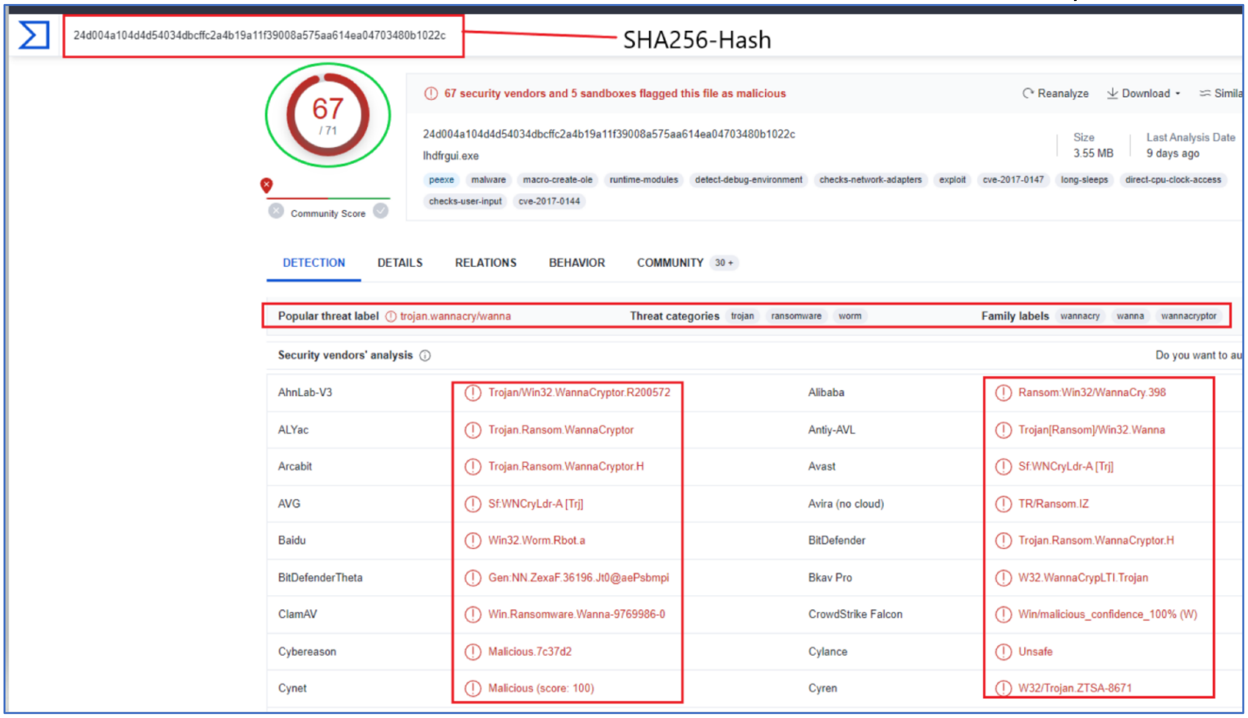
File Name	**Ransomware.wannacry.exe**
SHA256	24d004a104d4d54034dbcffc2a4b19a11f39008a575aa614ea04703480b1022c
SHA1	e889544aff85ffaf8b0d0da705105dee7c97fe26
MD5	db349b97c37d22f5ea1d1841e3c89eb

To obtain each of the file hashes the following commands can be seen below, and in the example screenshot below.

- sha256sum.exe C:\Users\husky\Desktop\Ransomware.wannacry.exe.mal
- sha1sum.exe C:\Users\husky\Desktop\Ransomware.wannacry.exe.malz
- md5sum.exe C:\Users\husky\Desktop\Ransomware.wannacry.exe.malz

Virus Total Analysis

Once the file hash was obtained the hashes were submitted to Virus Total for analysis



FLOSS Output

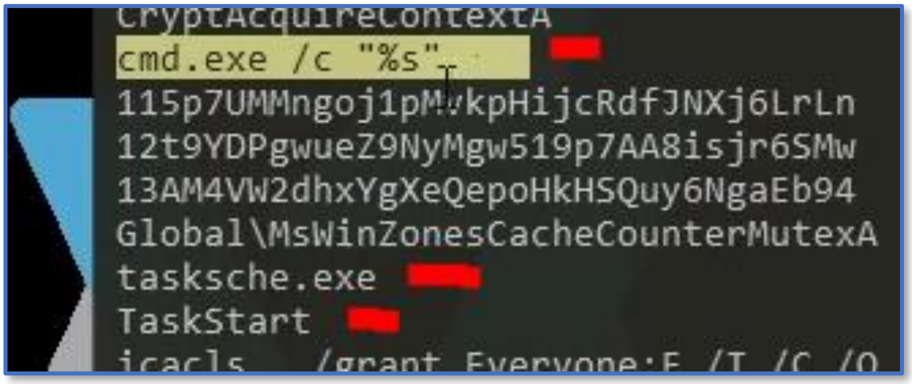
In the investigation of running FLOSS on the binary we detected the following API calls that the binary utilises.

```
| FLOSS STATIC STRINGS (1586) |
-----------------------------

-----------------------------
| FLOSS ASCII STRINGS (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
GetModuleHandleW
ExitProcess
GetModuleFileNameA
LocalFree
LocalAlloc
KERNEL32.dll
CryptAcquireContextA
CryptGenRandom
StartServiceA
CloseServiceHandle
CreateServiceA
OpenSCManagerA
SetServiceStatus
ChangeServiceConfig2A
RegisterServiceCtrlHandlerA
StartServiceCtrlDispatcherA
OpenServiceA
ADVAPI32.dll
WS2_32.dll
??1_Lockit@std@@QAE@XZ
??0_Lockit@std@@QAE@XZ
MSVCP60.dll
GetPerAdapterInfo
GetAdaptersInfo
iphlpapi.dll
InternetCloseHandle
InternetOpenUrlA
InternetOpenA
WININET.dll
_endthreadex
_beginthreadex
__CxxFrameHandler
__p___argc
:
```

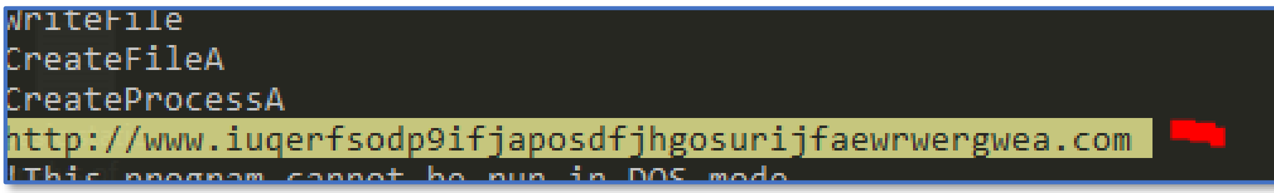
Upon investigation of the FLOSS, output resulted in file paths were detected. As can be seen, the program tasksche.exe can be found in the below screenshot.

```
Microsoft Security Center (2.0) Service
%s -m security
C:\%s\qeriuwjhrf
C:\%s\%s
tasksche.exe
```

Along with a call to cmd.exe with an associated file path.



Along with a URL that was identified.



Import address table.

Below shows the address table, as can be seen, we can observe further information about the binary, for example, the time stamp of when the binary was created, along with the language of the binary is written in C++.

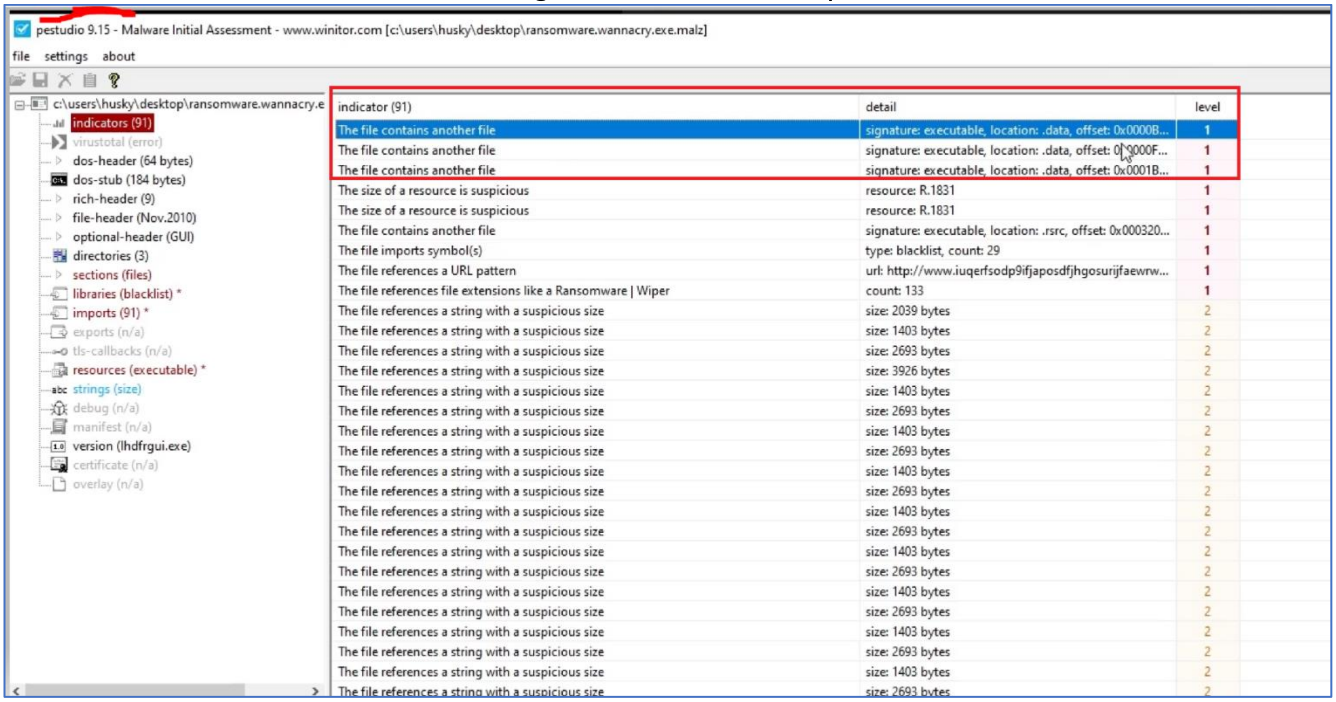
property	value
md5	DB349B97C37D22F5EA1D1841E3C89EB4
sha1	E889544AFF85FFAF8B0D0DA705105DEE7C97FE26
sha256	24D004A104D4D54034DBCFFC2A4B19A11F39008A575AA614EA04703480B1022C
first-bytes-hex	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00
first-bytes-text	M Z .. .. @ .. ..
file-size	3723264 bytes
entropy	7.964
imphash	n/a
signature	Microsoft Visual C++ v6.0
tooling	wait...
entry-point	55 8B EC 6A FF 68 A0 A1 40 00 68 A2 9B 40 00 64 A1 00 00 00 00 50 64 89 25 00 00 00 00 83 EC 68 53
file-version	6.1.7601.17514 (win7sp1_rtm.101119-1850)
description	Microsoft® Disk Defragmenter
file-type	executable
cpu	32-bit
subsystem	GUI
stamps	
compiler-stamp	Sat Nov 20 09:03:08 2010
debugger-stamp	n/a
resource-stamp	n/a
import-stamp	n/a
export-stamp	n/a

We discovered that the binary calls out to several interesting API Calls.

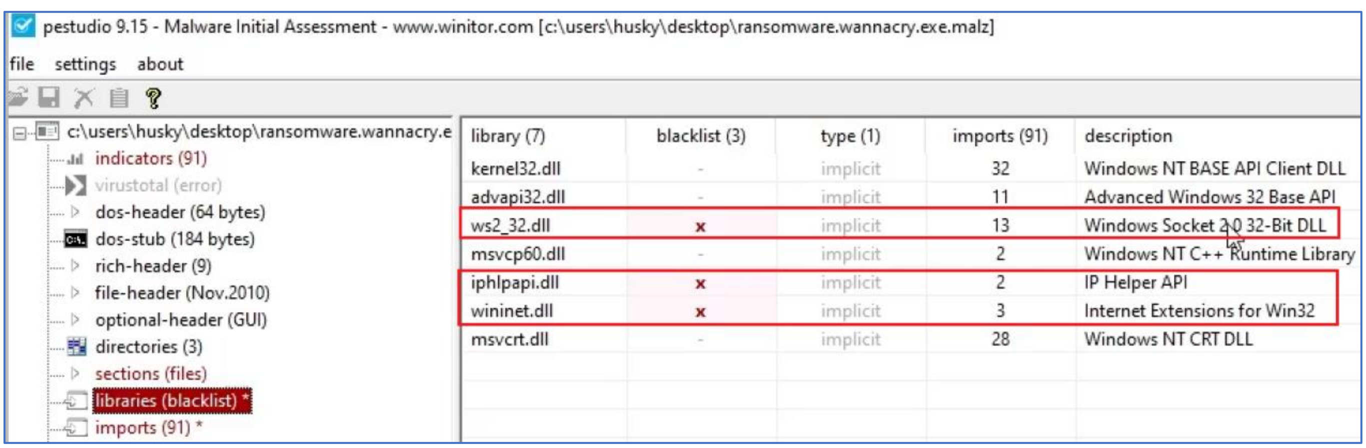
pFile	Data	Description	Value
0000A000	0000A6F6	Hint/Name RVA	024A StartServiceCtrlDispatcherA
0000A004	0000A6D8	Hint/Name RVA	020C RegisterServiceCtrlHandlerA
0000A008	0000A6C0	Hint/Name RVA	0034 ChangeServiceConfig2A
0000A00C	0000A6AC	Hint/Name RVA	0244 SetServiceStatus
0000A010	0000A69A	Hint/Name RVA	01AD OpenSCManagerA
0000A014	0000A688	Hint/Name RVA	0064 CreateServiceA
0000A018	0000A672	Hint/Name RVA	003E CloseServiceHandle
0000A01C	0000A662	Hint/Name RVA	0249 StartServiceA
0000A020	0000A650	Hint/Name RVA	0096 CryptGenRandom
0000A024	0000A638	Hint/Name RVA	0085 CryptAcquireContextA
0000A028	0000A714	Hint/Name RVA	01AF OpenServiceA
0000A02C	00000000	End of Imports	ADVAPI32.dll
0000A030	0000A4F6	Hint/Name RVA	0390 WaitForSingleObject
0000A034	0000A50C	Hint/Name RVA	022C InterlockedIncrement
0000A038	0000A524	Hint/Name RVA	0146 GetCurrentThreadId
0000A03C	0000A53A	Hint/Name RVA	0145 GetCurrentThread
0000A040	0000A54E	Hint/Name RVA	02B5 ReadFile
0000A044	0000A55A	Hint/Name RVA	0163 GetFileSize
0000A048	0000A568	Hint/Name RVA	0053 CreateFileA
0000A04C	0000A576	Hint/Name RVA	026F MoveFileExA
0000A050	0000A584	Hint/Name RVA	0355 SizeofResource
0000A054	0000A4E4	Hint/Name RVA	035F TerminateThread
0000A058	0000A5A6	Hint/Name RVA	0257 LoadResource
0000A05C	0000A5B6	Hint/Name RVA	00E3 FindResourceA
0000A060	0000A5C6	Hint/Name RVA	01A0 GetProcAddress
0000A064	0000A5D8	Hint/Name RVA	0182 GetModuleHandleW
0000A068	0000A5EC	Hint/Name RVA	00B9 ExitProcess
0000A06C	0000A5FA	Hint/Name RVA	017D GetModuleFileNameA
0000A070	0000A610	Hint/Name RVA	025C LocalFree
0000A074	0000A61C	Hint/Name RVA	0258 LocalAlloc
0000A078	0000A4D6	Hint/Name RVA	0034 CloseHandle
0000A07C	0000A4BE	Hint/Name RVA	0228 InterlockedDecrement
0000A080	0000A4A6	Hint/Name RVA	0098 EnterCriticalSection
0000A084	0000A48E	Hint/Name RVA	0251 LeaveCriticalSection
0000A088	0000A472	Hint/Name RVA	0223 InitializeCriticalSection
0000A08C	0000A464	Hint/Name RVA	01F8 GlobalAlloc
0000A090	0000A456	Hint/Name RVA	01FF GlobalFree
0000A094	0000A43A	Hint/Name RVA	02A4 QueryPerformanceFrequency
0000A098	0000A420	Hint/Name RVA	02A3 QueryPerformanceCounter
0000A09C	0000A410	Hint/Name RVA	01DF GetTickCount
0000A0A0	0000A596	Hint/Name RVA	0265 LockResource
0000A0A4	0000A408	Hint/Name RVA	0356 Sleep
0000A0A8	0000A97A	Hint/Name RVA	01B7 GetStartupInfoA
0000A0AC	0000A966	Hint/Name RVA	017F GetModuleHandleA
0000A0B0	00000000	End of Imports	KERNEL32.dll
0000A0B4	0000A73E	Hint/Name RVA	010B ??1_Lockit@std@@QAE@XZ
0000A0B8	0000A758	Hint/Name RVA	00A2 ??0_Lockit@std@@QAE@XZ
0000A0BC	00000000	End of Imports	MSVCP60.dll
0000A0C0	0000A932	Hint/Name RVA	0081 __set_app_type
0000A0C4	0000A98C	Hint/Name RVA	01C1 _stricmp
0000A0C8	0000A924	Hint/Name RVA	006F __p__fmode
0000A0CC	0000A914	Hint/Name RVA	006A __p__commode
0000A0D0	0000A944	Hint/Name RVA	00CA _except_handler3
0000A0D4	0000A8F0	Hint/Name RVA	0083 __setusermatherr
0000A0D8	0000A8E4	Hint/Name RVA	010F _initterm
0000A0DC	0000A8D4	Hint/Name RVA	0058 __getmainargs
0000A0E0	0000A8CA	Hint/Name RVA	008F _acmdln
0000A0E4	0000A904	Hint/Name RVA	009D _adjust_fdiv
0000A0E8	0000A958	Hint/Name RVA	00B7 _controlfp
0000A0EC	0000A8C2	Hint/Name RVA	0249 exit
0000A0F0	0000A8B4	Hint/Name RVA	0048 _XcptFilter
0000A0F4	0000A8AC	Hint/Name RVA	00D3 _exit
0000A0F8	0000A896	Hint/Name RVA	0186 _onexit
0000A0FC	0000A888	Hint/Name RVA	0055 __dllonexit
0000A100	0000A880	Hint/Name RVA	025E free
0000A104	0000A870	Hint/Name RVA	000E ??2@YAPAXI@Z

Basic Dynamic Analysis

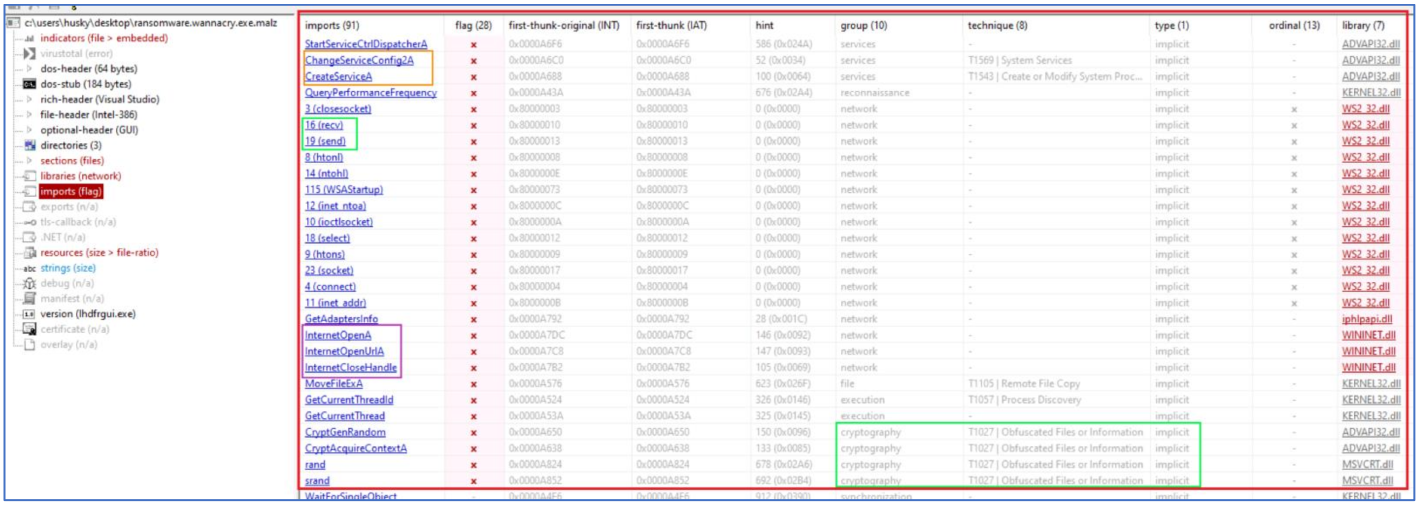
This section demonstrates the functionality of the WannaCry.exe binary upon execution, this will be conducted in a sand-boxed environment by utilising the Flare-VM. It was identified that the binary has various steps of execution to analyse each segment of execution a range of tools was utilised. Below shows PEStedio being used for further analysis.



Below shows a number of libraries that the binary is calling out to, it was observed that there is a Windows socket being used along with the IP helper API, and the Internet extensions dll.



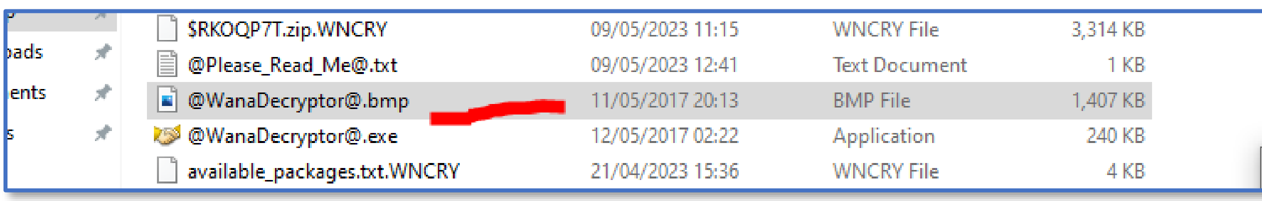
It was also observed that the binary is making use of some dangerous API calls as there is a receive and send API call being utilised. In conjunction with three internet API calls, these internet calls are as follows InterentOpenA, InterentOpenUrlA, and InterentCloseHandle.



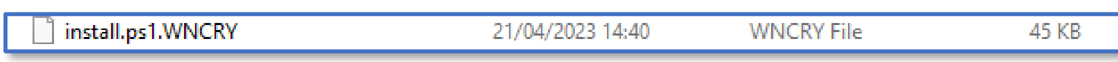
imports (91)	flag (28)	first-thunk-original (INT)	first-thunk (IAT)	hint	group (10)	technique (8)	type (1)	ordinal (13)	library (7)
StartServiceCtrlDispatcherA	x	0x0000A6F6	0x0000A6F6	586 (0x024A)	services	-	implicit	-	ADVAPI32.dll
ChangeServiceConfig2A	x	0x0000A6C0	0x0000A6C0	52 (0x0034)	services	T1569 | System Services	implicit	-	ADVAPI32.dll
CreateServiceA	x	0x0000A688	0x0000A688	100 (0x0064)	services	T1543 | Create or Modify System Proc...	implicit	-	ADVAPI32.dll
QueryPerformanceFrequency	x	0x0000A43A	0x0000A43A	676 (0x02A4)	reconnaissance	-	implicit	-	KERNEL32.dll
3 (closesocket)	x	0x80000003	0x80000003	0 (0x0000)	network	-	implicit	x	WS2_32.dll
16 (recv)	x	0x80000010	0x80000010	0 (0x0000)	network	-	implicit	x	WS2_32.dll
19 (send)	x	0x80000013	0x80000013	0 (0x0000)	network	-	implicit	x	WS2_32.dll
8 (htonl)	x	0x80000008	0x80000008	0 (0x0000)	network	-	implicit	x	WS2_32.dll
14 (ntohl)	x	0x8000000E	0x8000000E	0 (0x0000)	network	-	implicit	x	WS2_32.dll
115 (WSAStartup)	x	0x80000073	0x80000073	0 (0x0000)	network	-	implicit	x	WS2_32.dll
12 (inet_ntoa)	x	0x8000000C	0x8000000C	0 (0x0000)	network	-	implicit	x	WS2_32.dll
10 (ioctlsocket)	x	0x8000000A	0x8000000A	0 (0x0000)	network	-	implicit	x	WS2_32.dll
18 (select)	x	0x80000012	0x80000012	0 (0x0000)	network	-	implicit	x	WS2_32.dll
9 (htons)	x	0x80000009	0x80000009	0 (0x0000)	network	-	implicit	x	WS2_32.dll
23 (socket)	x	0x80000017	0x80000017	0 (0x0000)	network	-	implicit	x	WS2_32.dll
4 (connect)	x	0x80000004	0x80000004	0 (0x0000)	network	-	implicit	x	WS2_32.dll
11 (inet_addr)	x	0x8000000B	0x8000000B	0 (0x0000)	network	-	implicit	x	WS2_32.dll
GetAdaptersInfo	x	0x0000A792	0x0000A792	28 (0x001C)	network	-	implicit	-	iphlpapi.dll
InternetOpenA	x	0x0000A7DC	0x0000A7DC	146 (0x0092)	network	-	implicit	-	WININET.dll
InternetOpenUrlA	x	0x0000A7C8	0x0000A7C8	147 (0x0093)	network	-	implicit	-	WININET.dll
InternetCloseHandle	x	0x0000A7B2	0x0000A7B2	105 (0x0069)	network	-	implicit	-	WININET.dll
MoveFileExA	x	0x0000A576	0x0000A576	623 (0x026F)	file	T1105 | Remote File Copy	implicit	-	KERNEL32.dll
GetCurrentThreadId	x	0x0000A524	0x0000A524	326 (0x0146)	execution	T1057 | Process Discovery	implicit	-	KERNEL32.dll
GetCurrentThread	x	0x0000A53A	0x0000A53A	325 (0x0145)	execution	-	implicit	-	KERNEL32.dll
CryptGenRandom	x	0x0000A650	0x0000A650	150 (0x0096)	cryptography	T1027 | Obfuscated Files or Information	implicit	-	ADVAPI32.dll
CryptAcquireContextA	x	0x0000A638	0x0000A638	133 (0x0085)	cryptography	T1027 | Obfuscated Files or Information	implicit	-	ADVAPI32.dll
rand	x	0x0000A824	0x0000A824	678 (0x02A6)	cryptography	T1027 | Obfuscated Files or Information	implicit	-	MSVCRT.dll
srand	x	0x0000A852	0x0000A852	692 (0x02B4)	cryptography	T1027 | Obfuscated Files or Information	implicit	-	MSVCRT.dll
WaitForSingleObject	-	0x0000A4F6	0x0000A4F6	912 (0x0390)	synchronization	-	implicit	-	KERNEL32.dll

After Detonation

The below-highlighted image shows the picture that is used for the desktop background once the binary is executed. Along with other files that are created upon detonation of the Wannacry.exe binary.

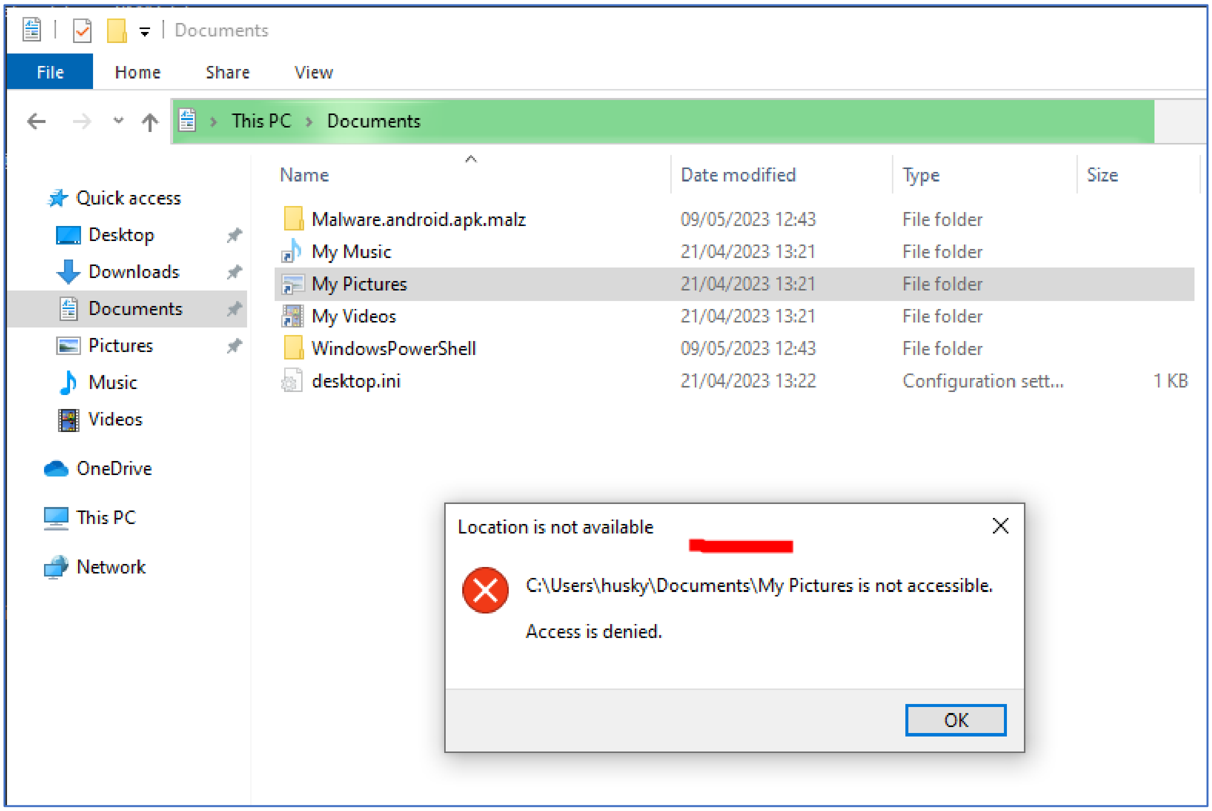


$RKOQP7T.zip.WNCRY	09/05/2023 11:15	WNCRY File	3,314 KB
@Please_Read_Me@.txt	09/05/2023 12:41	Text Document	1 KB
@WanaDecryptor@.bmp	11/05/2017 20:13	BMP File	1,407 KB
@WanaDecryptor@.exe	12/05/2017 02:22	Application	240 KB
available_packages.txt.WNCRY	21/04/2023 15:36	WNCRY File	4 KB



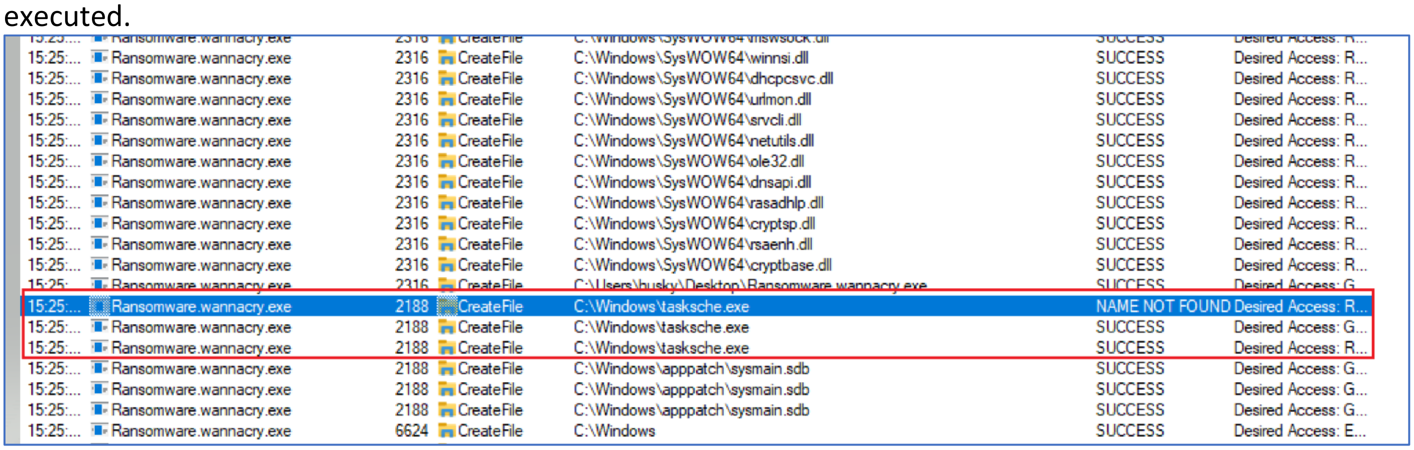
install.ps1.WNCRY	21/04/2023 14:40	WNCRY File	45 KB

When attempting to access files after detonation, files are encrypted and can be accessed as can be seen in the below example.



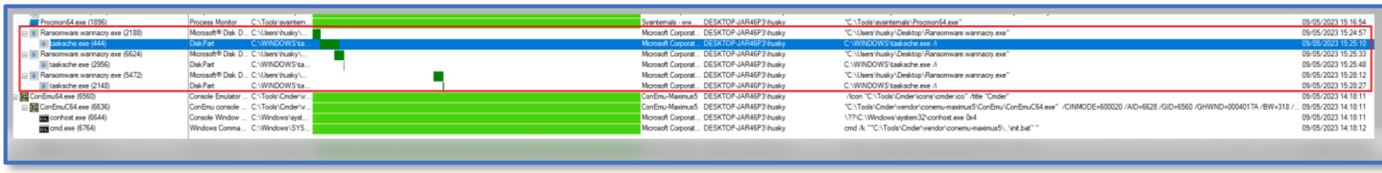
Advanced Static Analysis

Using the security tool Procmon it was determined that there is a file creation upon execution of the binary, the below screenshot shows the details that Procmon has identified. As can be seen, the file creation shows the location of the tasksche.exe program once the binary is executed.

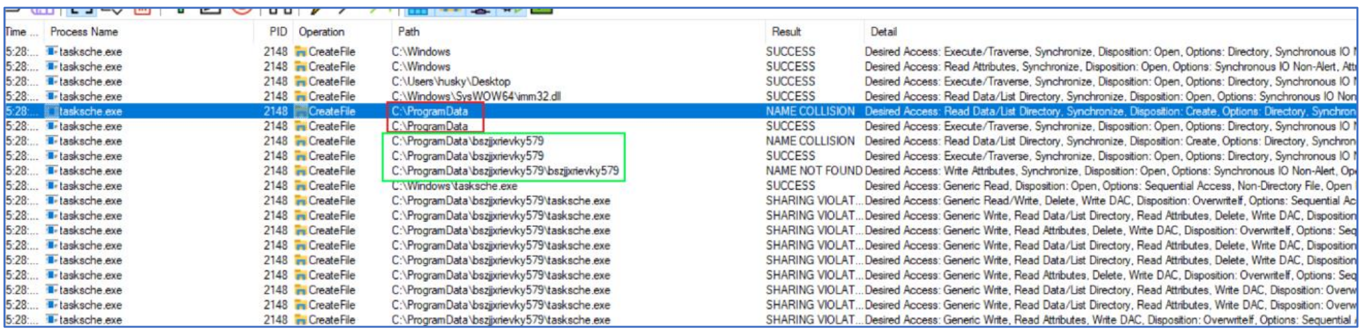


15:25:...	Ransomware.wannacry.exe	2316	CreateFile	C:\Windows\SysWOW64\winnsi.dll	SUCCESS	Desired Access: R...
15:25:...	Ransomware.wannacry.exe	2316	CreateFile	C:\Windows\SysWOW64\dhcpcsvc.dll	SUCCESS	Desired Access: R...
15:25:...	Ransomware.wannacry.exe	2316	CreateFile	C:\Windows\SysWOW64\urlmon.dll	SUCCESS	Desired Access: R...
15:25:...	Ransomware.wannacry.exe	2316	CreateFile	C:\Windows\SysWOW64\srvcli.dll	SUCCESS	Desired Access: R...
15:25:...	Ransomware.wannacry.exe	2316	CreateFile	C:\Windows\SysWOW64\netutils.dll	SUCCESS	Desired Access: R...
15:25:...	Ransomware.wannacry.exe	2316	CreateFile	C:\Windows\SysWOW64\ole32.dll	SUCCESS	Desired Access: R...
15:25:...	Ransomware.wannacry.exe	2316	CreateFile	C:\Windows\SysWOW64\dnsapi.dll	SUCCESS	Desired Access: R...
15:25:...	Ransomware.wannacry.exe	2316	CreateFile	C:\Windows\SysWOW64\rasadhlp.dll	SUCCESS	Desired Access: R...
15:25:...	Ransomware.wannacry.exe	2316	CreateFile	C:\Windows\SysWOW64\cryptsp.dll	SUCCESS	Desired Access: R...
15:25:...	Ransomware.wannacry.exe	2316	CreateFile	C:\Windows\SysWOW64\rsaenh.dll	SUCCESS	Desired Access: R...
15:25:...	Ransomware.wannacry.exe	2316	CreateFile	C:\Windows\SysWOW64\cryptbase.dll	SUCCESS	Desired Access: R...
15:25:...	Ransomware.wannacry.exe	2316	CreateFile	C:\Users\husky\Desktop\Ransomware.wannacry.exe	SUCCESS	Desired Access: G...
15:25:...	Ransomware.wannacry.exe	2188	CreateFile	C:\Windows\tasksche.exe	NAME NOT FOUND	Desired Access: R...
15:25:...	Ransomware.wannacry.exe	2188	CreateFile	C:\Windows\tasksche.exe	SUCCESS	Desired Access: G...
15:25:...	Ransomware.wannacry.exe	2188	CreateFile	C:\Windows\tasksche.exe	SUCCESS	Desired Access: R...
15:25:...	Ransomware.wannacry.exe	2188	CreateFile	C:\Windows\apppatch\sysmain.sdb	SUCCESS	Desired Access: G...
15:25:...	Ransomware.wannacry.exe	2188	CreateFile	C:\Windows\apppatch\sysmain.sdb	SUCCESS	Desired Access: G...
15:25:...	Ransomware.wannacry.exe	2188	CreateFile	C:\Windows\apppatch\sysmain.sdb	SUCCESS	Desired Access: G...
15:25:...	Ransomware.wannacry.exe	6624	CreateFile	C:\Windows	SUCCESS	Desired Access: E...

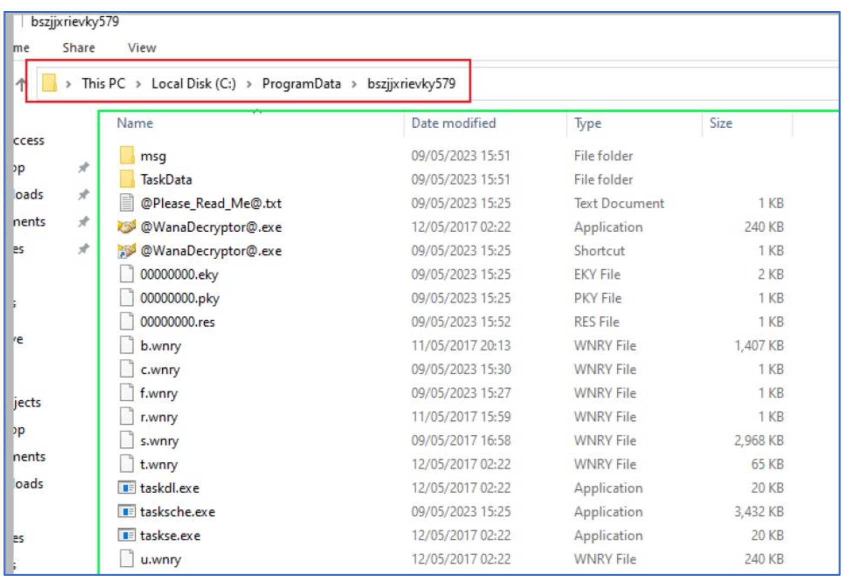
Using the process tree, it was detected that from the original binary that was executed, the tasksche.exe that was created is unpacked and is running as a process with the argument /I. As shown in the below example.



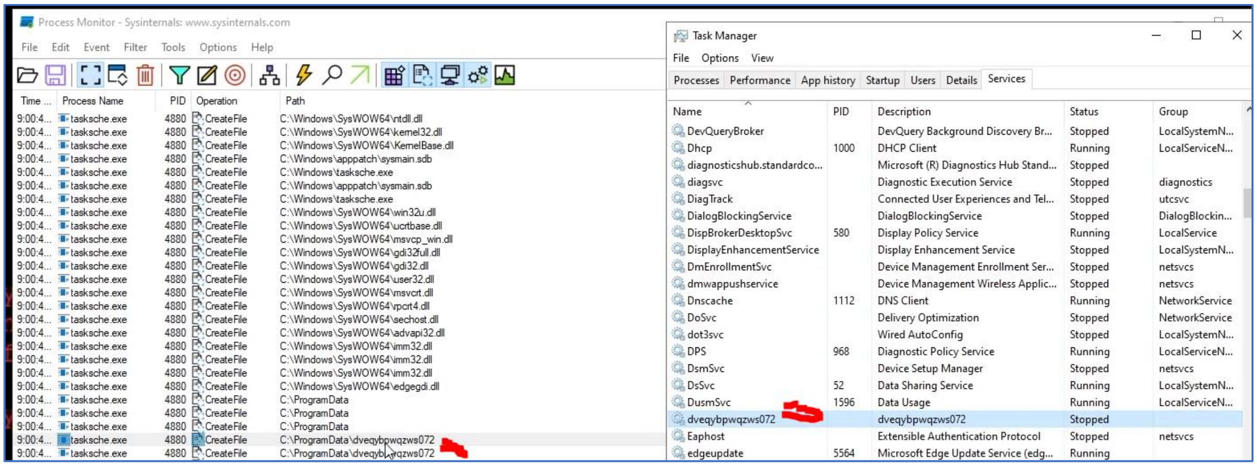
Upon further investigation of the binary, it was discovered that there was another file creation that was performed when the binary was executed, this can be seen in the below example there is a directory created along with several file creations.



Once the file path location was obtained of the other files that were created, it was investigated to gain an understanding of what were the contents of the file and directory that was discovered. As can be seen below the contents of this directory is responsible for encrypting the system files when the WannaCry binary is executed.



The bszjjxrievky579 is also run as a service on the local machine after the execution of the binary, this was identified by using the task manager and checking the running services of the system. It was not possible to terminate this service once the Wanna Cry binary was executed.

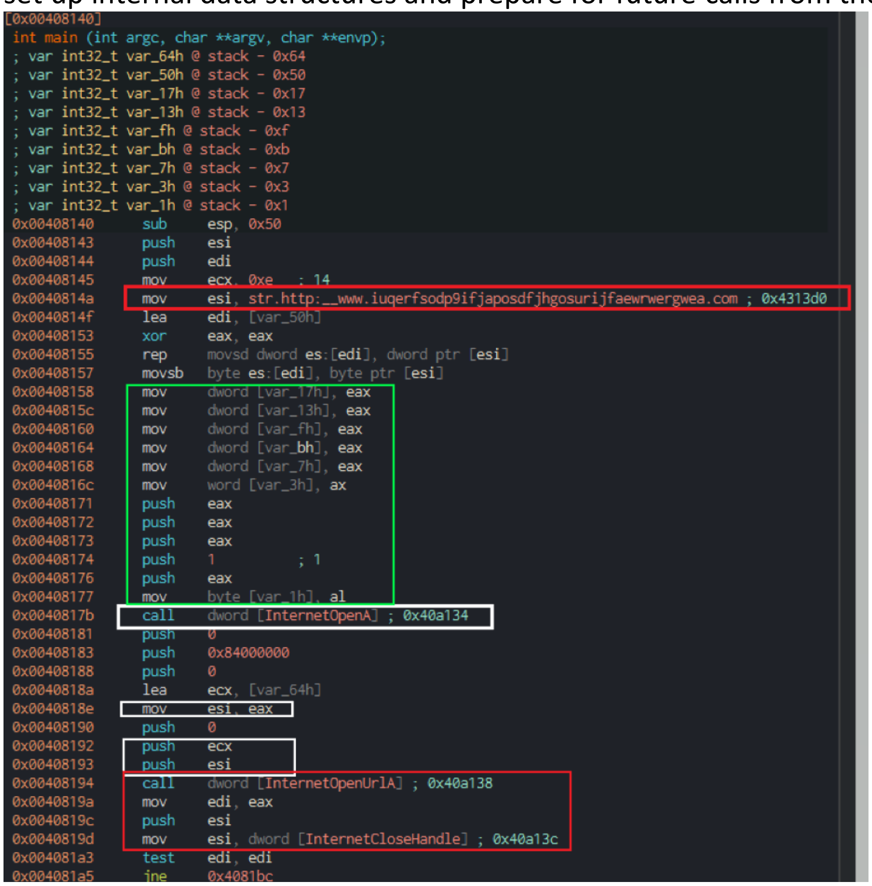


Advanced Dynamic Analysis

This section contains further analysis of the WannaCry binary with a decompiler called Cutter, Cutter is an integrated decompiler that enables malware to be examined at the assembly level to understand what is occurring with a segment of code that is contained within the program.

As demonstrated below the Wanna Cry binary is broken down into assembly language so we can attain how the binary actually functions at each stack position. It was observed that upon execution of the binary, there is an API call being performed to the URL in the first highlighted section in the below example, there are also multiple stack operations that are moving eax further onto the stack.

After multiple mov operations are performed there is another call operation which calls the API call InternetOpenA, this indicates that the Wanna Cry binary is instructing the Internet DLL to set up internal data structures and prepare for future calls from the binary.



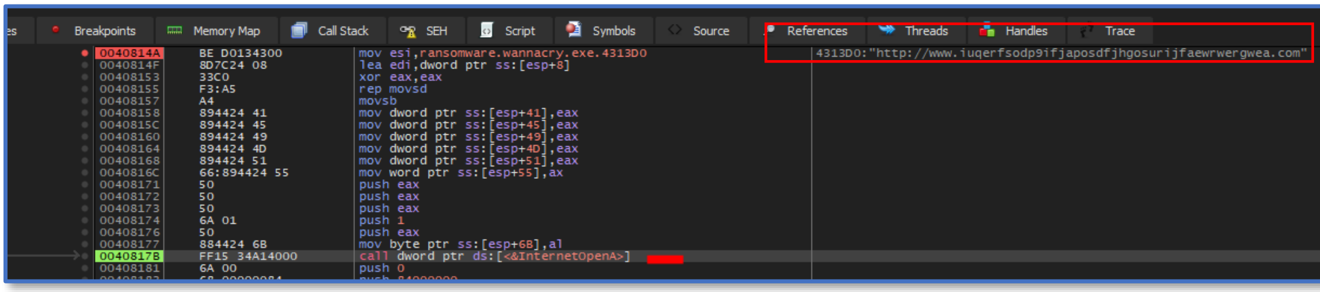
```
[0x00408140]
 int main (int argc, char **argv, char **envp);
 ; var int32_t var_64h @ stack - 0x64
 ; var int32_t var_50h @ stack - 0x50
 ; var int32_t var_17h @ stack - 0x17
 ; var int32_t var_13h @ stack - 0x13
 ; var int32_t var_fh @ stack - 0xf
 ; var int32_t var_bh @ stack - 0xb
 ; var int32_t var_7h @ stack - 0x7
 ; var int32_t var_3h @ stack - 0x3
 ; var int32_t var_1h @ stack - 0x1
 0x00408140      sub     esp, 0x50
 0x00408143      push    esi
 0x00408144      push    edi
 0x00408145      mov     ecx, 0xe    ; 14
 0x0040814a      mov     esi, str.http:__www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com ; 0x4313d0
 0x0040814f      lea     edi, [var_50h]
 0x00408153      xor     eax, eax
 0x00408155      rep     movsd dword es:[edi], dword ptr [esi]
 0x00408157      movsb   byte es:[edi], byte ptr [esi]
 0x00408158      mov     dword [var_17h], eax
 0x0040815c      mov     dword [var_13h], eax
 0x00408160      mov     dword [var_fh], eax
 0x00408164      mov     dword [var_bh], eax
 0x00408168      mov     dword [var_7h], eax
 0x0040816c      mov     word [var_3h], ax
 0x00408171      push    eax
 0x00408172      push    eax
 0x00408173      push    eax
 0x00408174      push    1           ; 1
 0x00408176      push    eax
 0x00408177      mov     byte [var_1h], al
 0x0040817b      call    dword [InternetOpenA] ; 0x40a134
 0x00408181      push    0
 0x00408183      push    0x84000000
 0x00408188      push    0
 0x0040818a      lea     ecx, [var_64h]
 0x0040818e      mov     esi, eax
 0x00408190      push    0
 0x00408192      push    ecx
 0x00408193      push    esi
 0x00408194      call    dword [InternetOpenUrlA] ; 0x40a138
 0x0040819a      mov     edi, eax
 0x0040819c      push    esi
 0x0040819d      mov     esi, dword [InternetCloseHandle] ; 0x40a13c
 0x004081a3      test    edi, edi
 0x004081a5      jne     0x4081bc
```

Once the decompiler recognizes that the outcome of the InternetOpenA URL CALL is loaded into the register of eax register, the eax register contents of each are then loaded into the EDI register in the Cutter output. This can be seen in the below example. It is noted that the contents of esi is the of the URL to which the binary calls out upon execution of the binary.

Decompiler (main)

```
/* jsdec pseudo code output */
/* C:\Users\husky\Desktop\Ransomware.wannacry.exe.malz @ 0x408140 */
#include <stdint.h>

int32_t main (void) {
    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 = "http://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;
}
```

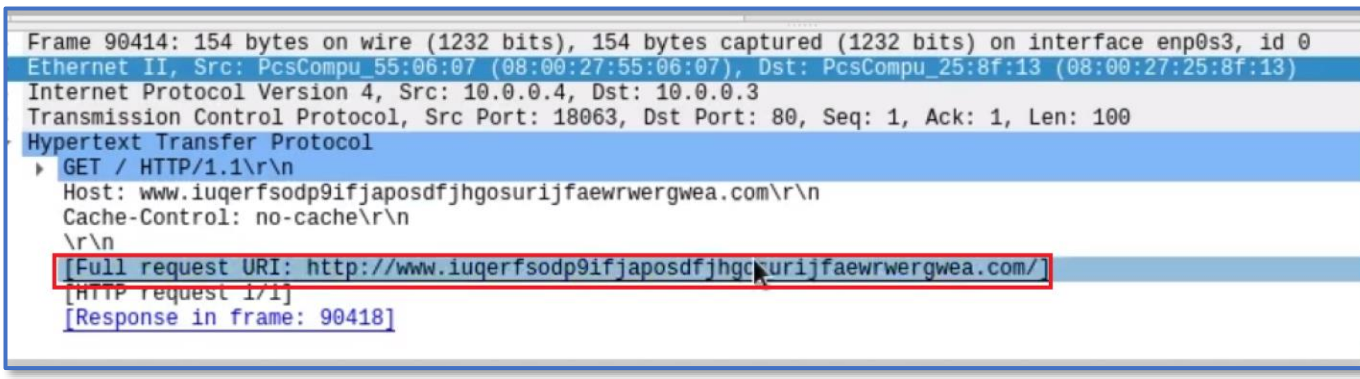
After the esi function is pushed onto the stack the binary will attempt to make a connection to the URL that is shown in the below example then the binary will instruct another API call to occur the InternetOpenA



Indicators of Compromise

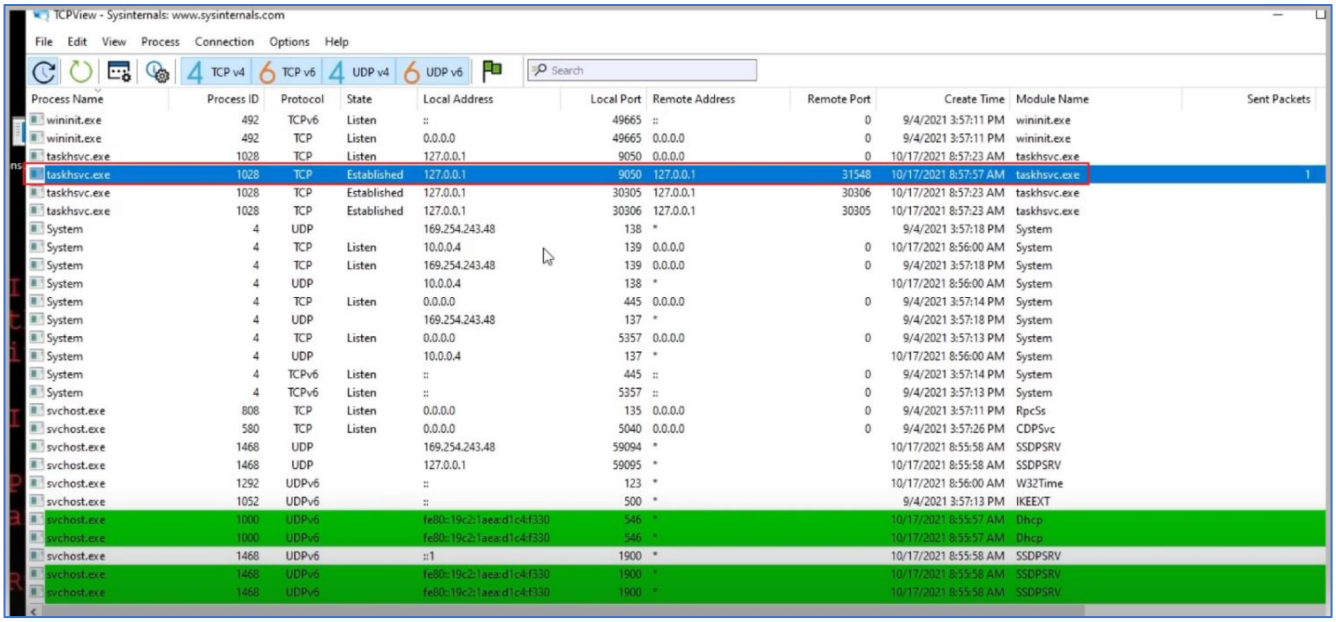
Network Indicators

The below example shows the network traffic that was captured using Wireshark, to be able to generate network traffic for this binary a fake internet simulation had to be utilised so the binary would think that it is trying to obtain a connection to a machine. For this test, the Linux operating system REMnux was utilised, the below screenshot shows the URL that the binary calls out to if an internet connection is present. If there is no internet connection present the binary will not execute.



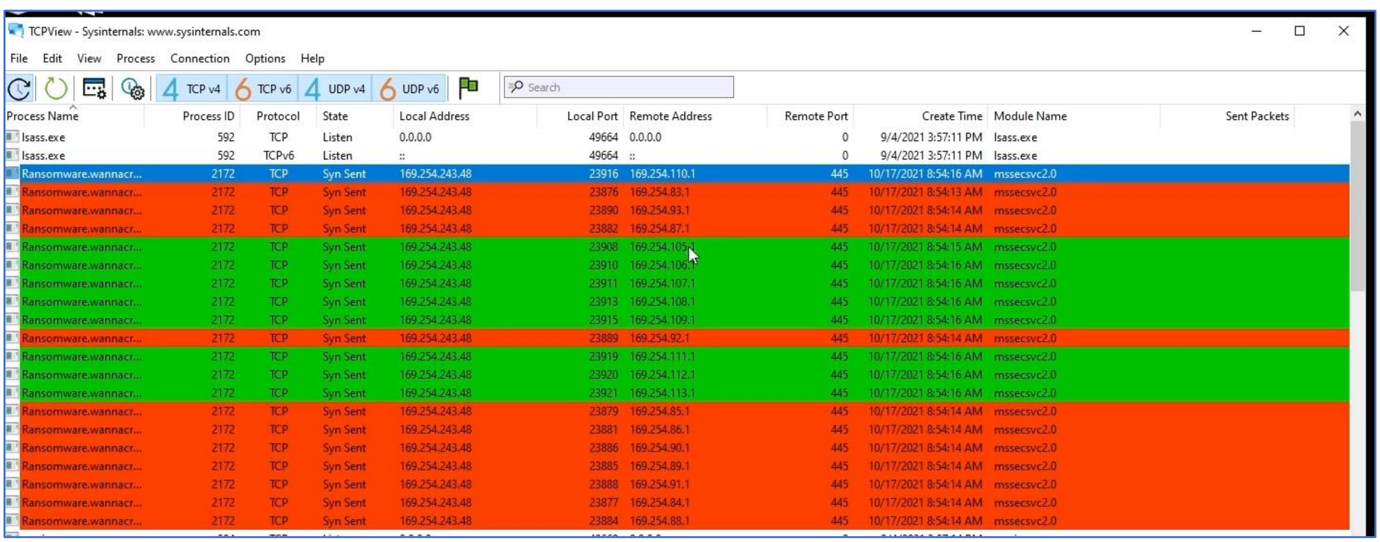
Host-based Indicators

Upon execution of the binary, it was discovered that the binary makes a connection to the remote port of 31548 it can also be observed that the taskhsvc.exe is the connection that the binary is calling out to. The taskhsvc.exe is a program that is created on the machine after the binary has been successfully executed on the local machine.



Process Name	Process ID	Protocol	State	Local Address	Local Port	Remote Address	Remote Port	Create Time	Module Name	Sent Packets
wininit.exe	492	TCPv6	Listen	::	49665	::	0	9/4/2021 3:57:11 PM	wininit.exe	
wininit.exe	492	TCP	Listen	0.0.0.0	49665	0.0.0.0	0	9/4/2021 3:57:11 PM	wininit.exe	
taskhsvc.exe	1028	TCP	Listen	127.0.0.1	9050	0.0.0.0	0	10/17/2021 8:57:23 AM	taskhsvc.exe	
taskhsvc.exe	1028	TCP	Established	127.0.0.1	9050	127.0.0.1	31548	10/17/2021 8:57:57 AM	taskhsvc.exe	1
taskhsvc.exe	1028	TCP	Established	127.0.0.1	30305	127.0.0.1	30306	10/17/2021 8:57:23 AM	taskhsvc.exe	
taskhsvc.exe	1028	TCP	Established	127.0.0.1	30306	127.0.0.1	30305	10/17/2021 8:57:23 AM	taskhsvc.exe	
System	4	UDP		169.254.243.48	138	*		9/4/2021 3:57:18 PM	System	
System	4	TCP	Listen	10.0.0.4	139	0.0.0.0	0	10/17/2021 8:56:00 AM	System	
System	4	TCP	Listen	169.254.243.48	139	0.0.0.0	0	9/4/2021 3:57:18 PM	System	
System	4	UDP		10.0.0.4	138	*		10/17/2021 8:56:00 AM	System	
System	4	TCP	Listen	0.0.0.0	445	0.0.0.0	0	9/4/2021 3:57:14 PM	System	
System	4	UDP		169.254.243.48	137	*		9/4/2021 3:57:18 PM	System	
System	4	TCP	Listen	0.0.0.0	5357	0.0.0.0	0	9/4/2021 3:57:13 PM	System	
System	4	UDP		10.0.0.4	137	*		10/17/2021 8:56:00 AM	System	
System	4	TCPv6	Listen	::	445	::	0	9/4/2021 3:57:14 PM	System	
System	4	TCPv6	Listen	::	5357	::	0	9/4/2021 3:57:13 PM	System	
svchost.exe	808	TCP	Listen	0.0.0.0	135	0.0.0.0	0	9/4/2021 3:57:11 PM	RpcSs	
svchost.exe	580	TCP	Listen	0.0.0.0	5040	0.0.0.0	0	9/4/2021 3:57:26 PM	CDPSvc	
svchost.exe	1468	UDP		169.254.243.48	59094	*		10/17/2021 8:55:58 AM	SSDPSRV	
svchost.exe	1468	UDP		127.0.0.1	59095	*		10/17/2021 8:55:58 AM	SSDPSRV	
svchost.exe	1292	UDPv6		::	123	*		10/17/2021 8:56:00 AM	W32Time	
svchost.exe	1052	UDPv6		::	500	*		9/4/2021 3:57:13 PM	IKEEXT	
svchost.exe	1000	UDPv6		fe80::19c2:1aea:d1c4:f330	546	*		10/17/2021 8:55:57 AM	Dhcp	
svchost.exe	1000	UDPv6		fe80::19c2:1aea:d1c4:f330	546	*		10/17/2021 8:55:57 AM	Dhcp	
svchost.exe	1468	UDPv6		::1	1900	*		10/17/2021 8:55:58 AM	SSDPSRV	
svchost.exe	1468	UDPv6		fe80::19c2:1aea:d1c4:f330	1900	*		10/17/2021 8:55:58 AM	SSDPSRV	
svchost.exe	1468	UDPv6		fe80::19c2:1aea:d1c4:f330	1900	*		10/17/2021 8:55:58 AM	SSDPSRV	

However, once the program is executed, we observed there is a lot of traffic that is making a remote connection using the SMB protocol on TCP-port 445 through the service mssecsvc2.0. This is shown in the below example. This was achieved by simulating a fake internet connection. The example used to achieve this was the Linux operating system REMnux in conjunction with Inetsim.



Process Name	Process ID	Protocol	State	Local Address	Local Port	Remote Address	Remote Port	Create Time	Module Name	Sent Packets
lsass.exe	592	TCP	Listen	0.0.0.0	49664	0.0.0.0	0	9/4/2021 3:57:11 PM	lsass.exe	
lsass.exe	592	TCPv6	Listen	::	49664	::	0	9/4/2021 3:57:11 PM	lsass.exe	
Ransomware.wannacr...	2172	TCP	Syn Sent	169.254.243.48	23916	169.254.110.1	445	10/17/2021 8:54:16 AM	mssecsvc2.0	
Ransomware.wannacr...	2172	TCP	Syn Sent	169.254.243.48	23876	169.254.83.1	445	10/17/2021 8:54:13 AM	mssecsvc2.0	
Ransomware.wannacr...	2172	TCP	Syn Sent	169.254.243.48	23890	169.254.93.1	445	10/17/2021 8:54:14 AM	mssecsvc2.0	
Ransomware.wannacr...	2172	TCP	Syn Sent	169.254.243.48	23882	169.254.87.1	445	10/17/2021 8:54:14 AM	mssecsvc2.0	
Ransomware.wannacr...	2172	TCP	Syn Sent	169.254.243.48	23908	169.254.105.1	445	10/17/2021 8:54:15 AM	mssecsvc2.0	
Ransomware.wannacr...	2172	TCP	Syn Sent	169.254.243.48	23910	169.254.106.1	445	10/17/2021 8:54:16 AM	mssecsvc2.0	
Ransomware.wannacr...	2172	TCP	Syn Sent	169.254.243.48	23911	169.254.107.1	445	10/17/2021 8:54:16 AM	mssecsvc2.0	
Ransomware.wannacr...	2172	TCP	Syn Sent	169.254.243.48	23913	169.254.108.1	445	10/17/2021 8:54:16 AM	mssecsvc2.0	
Ransomware.wannacr...	2172	TCP	Syn Sent	169.254.243.48	23915	169.254.109.1	445	10/17/2021 8:54:16 AM	mssecsvc2.0	
Ransomware.wannacr...	2172	TCP	Syn Sent	169.254.243.48	23889	169.254.92.1	445	10/17/2021 8:54:14 AM	mssecsvc2.0	
Ransomware.wannacr...	2172	TCP	Syn Sent	169.254.243.48	23919	169.254.111.1	445	10/17/2021 8:54:16 AM	mssecsvc2.0	
Ransomware.wannacr...	2172	TCP	Syn Sent	169.254.243.48	23920	169.254.112.1	445	10/17/2021 8:54:16 AM	mssecsvc2.0	
Ransomware.wannacr...	2172	TCP	Syn Sent	169.254.243.48	23921	169.254.113.1	445	10/17/2021 8:54:16 AM	mssecsvc2.0	
Ransomware.wannacr...	2172	TCP	Syn Sent	169.254.243.48	23879	169.254.85.1	445	10/17/2021 8:54:14 AM	mssecsvc2.0	
Ransomware.wannacr...	2172	TCP	Syn Sent	169.254.243.48	23881	169.254.86.1	445	10/17/2021 8:54:14 AM	mssecsvc2.0	
Ransomware.wannacr...	2172	TCP	Syn Sent	169.254.243.48	23886	169.254.90.1	445	10/17/2021 8:54:14 AM	mssecsvc2.0	
Ransomware.wannacr...	2172	TCP	Syn Sent	169.254.243.48	23885	169.254.89.1	445	10/17/2021 8:54:14 AM	mssecsvc2.0	
Ransomware.wannacr...	2172	TCP	Syn Sent	169.254.243.48	23888	169.254.91.1	445	10/17/2021 8:54:14 AM	mssecsvc2.0	
Ransomware.wannacr...	2172	TCP	Syn Sent	169.254.243.48	23877	169.254.84.1	445	10/17/2021 8:54:14 AM	mssecsvc2.0	
Ransomware.wannacr...	2172	TCP	Syn Sent	169.254.243.48	23884	169.254.88.1	445	10/17/2021 8:54:14 AM	mssecsvc2.0	

Rules & Signatures

A full set of YARA rules is included in Appendix A. Shown below shows a list of strings that are considered to be malicious that are contained within the binary.

Strings for the infected binary

- cmd.exe /c "%s"
- tasksche.exe
- icacls . /grant Everyone:F /T /C /Q
- WNcry@2ol7
- taskdl.exe
- diskpart.exe
- lhdfrgui.exe

Signatures (Hashes)

WannaCry.exe

- 24d004a104d4d54034dbcffc2a4b19a11f39008a575aa614ea04703480b1022c

taskse.exe

- 2CA2D550E603D74DEDDA03156023135B38DA3630CB014E3D00B1263358C5F00D

taskdl.exe

- 4A468603FDCB7A2EB5770705898CF9EF37AADE532A7964642ECD705A74794B79

tasksche.exe

- ED01EBFBC9EB5BBEA545AF4D01BF5F1071661840480439C6E5BABE8E080E41AA

Appendices

A. Yara Rules

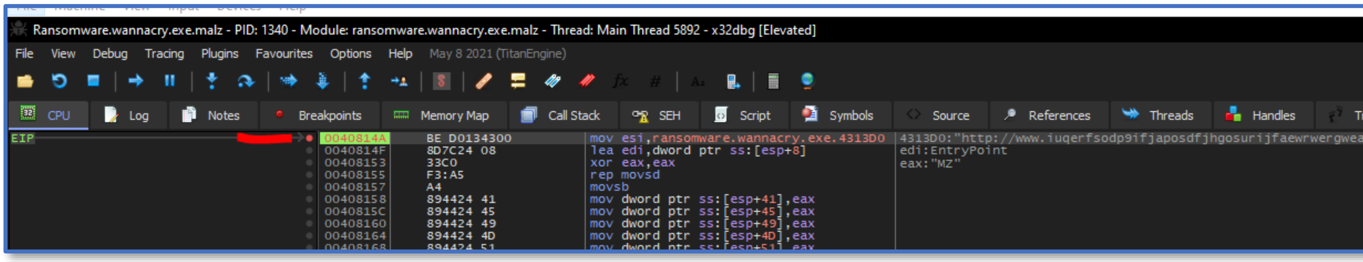
```
rule Yara_Example_WannaCry {

    meta:
        last_updated = "07-08-2023"
        author = "Dominic Lynch"
        description = "Self-Learned Yara rules for WannaCry-sample"

    strings:
        // Fill out identifying strings and other criteria
        $PE_magic_byte = "MZ"
        $string1 = "iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea"
        $string2 = ".msg/m_vietnamese.wnry"
        $string3 = "WANACRY"
        $string4 = "SMB"
        $string5 = "tasksche.exe"
        $string6 = "XX^_]ZY[A\A]A^A_H"
    condition:
        // Fill out the conditions that must be met to identify the binary
        $PE_magic_byte at 0
        and $string1
        and $string2
        and $string3
        and $string4
        and $string5
        and $string6
```

B. Callback URLs

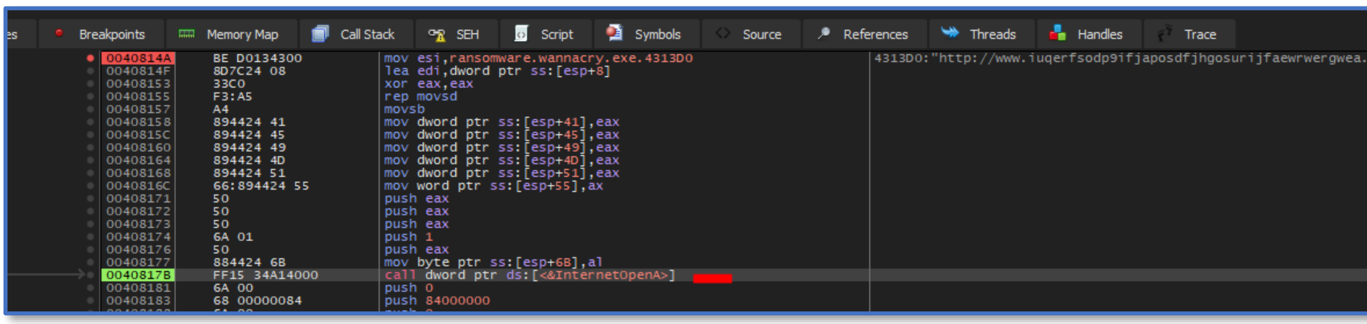
Domain	Port
httxp://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com	80

C. Decompiled Code Snippets

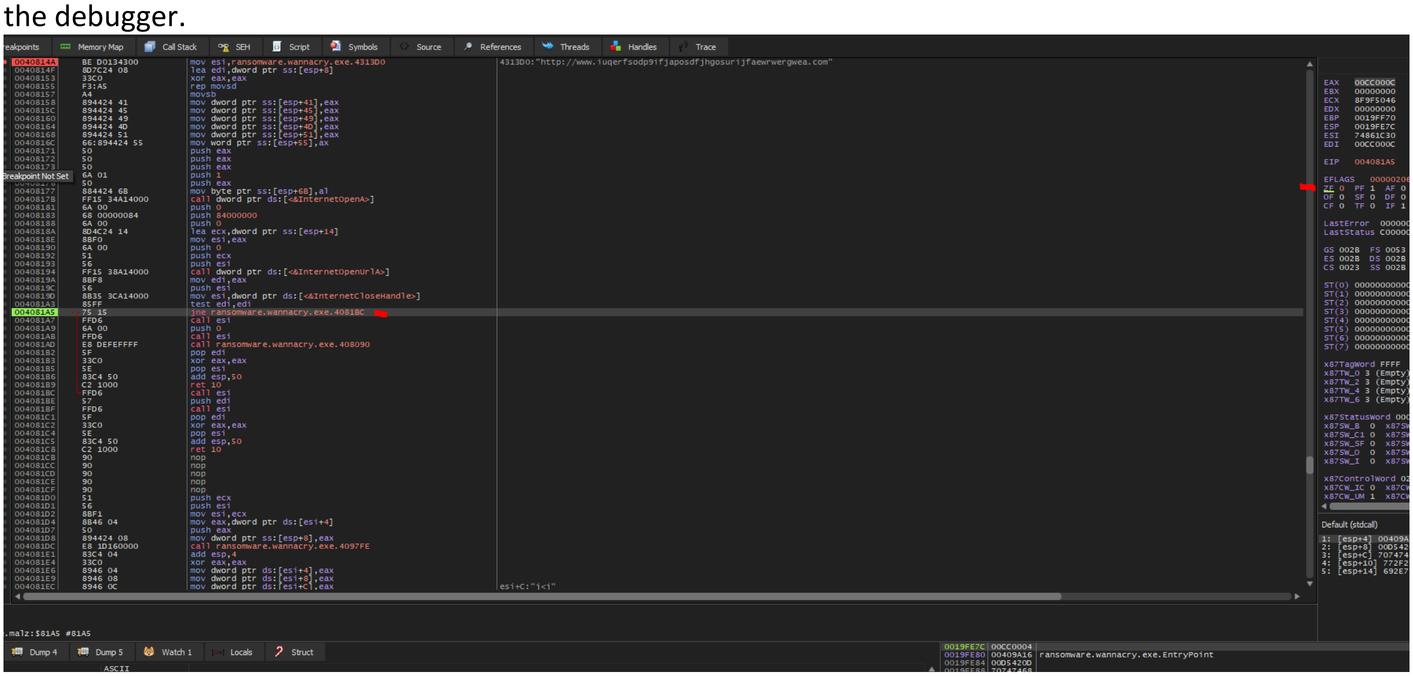
Below shows the assembly code of the binary, the section of the code below the ransomware.wannacry.exe is performing a mov operation to make the binary become prepared to call out to the URL shown below.



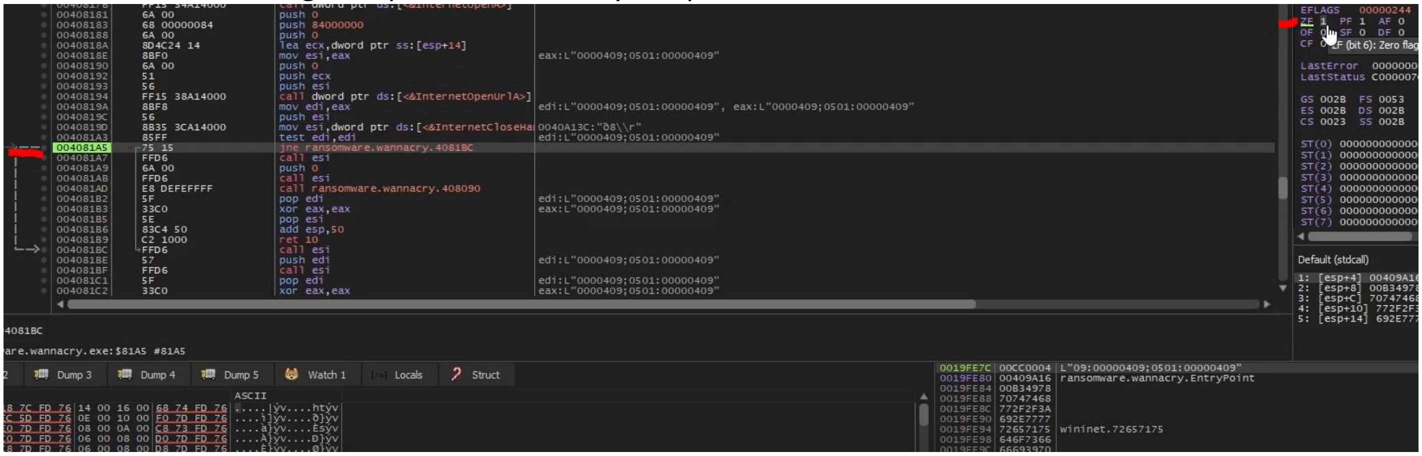
After the mov operation, the binary is then making a call operation with the API call of InterentOpenA being called onto the stack.



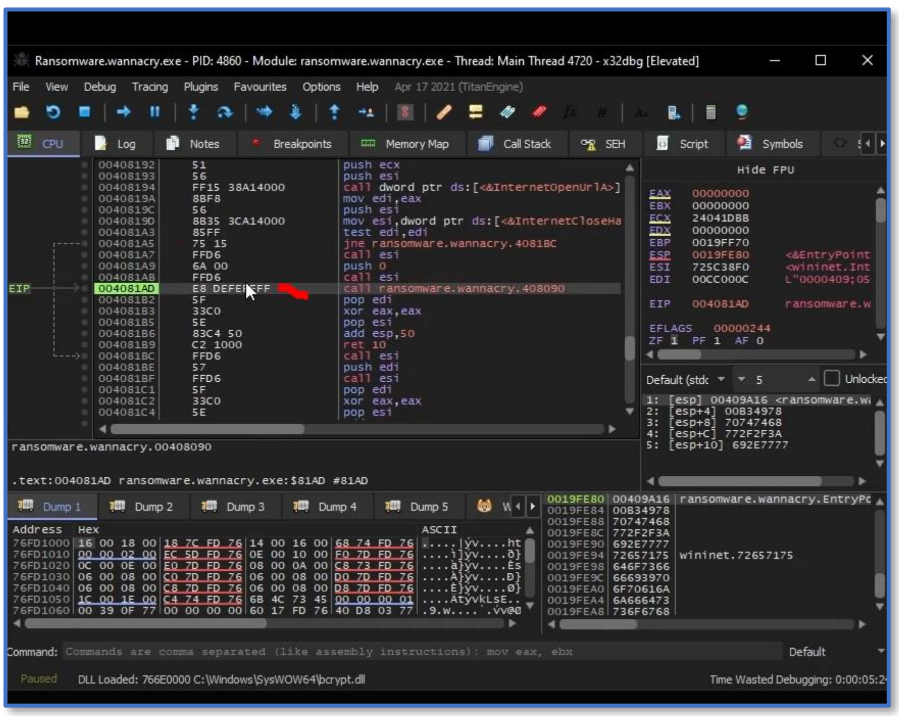
The below screenshot shows the operation (Jump If Not Equal) JNE, with the zero-flag set to zero. The zero indicates there is no internet connection the JNE operation will jump over the remaining API calls if the value stays at zero. Therefore, the binary will not execute from within the debugger.



However, if the ZF flag is modified to 1 this means there is an internet connection, and it can be seen what the following next process the binary will perform.



When the flag was changed to 1 the remaining parts of the program will be executed, this will execute the full binary from within the debugger, The execution of the binary took place because the JNE value was set to 1 and there was a fake internet simulation set up for testing this binary.



Below shows that the binary was executed after execution from within the debugger program. It now can be observed when the execution point exists and what conditions have to be met for the binary to be executed.