Practical Malware Analysis & Triage

Malware Analysis Report

WannaCry Malware

DEC 2023 | Igor_sec | v1.0

Table of Contents

Executive Summary 4
High-Level Technical Summary 5
Phase 1: Initial Contact and Kill Switch Check 5
Phase 2: System Modification and Propagation 5
Phase 3: Encryption and Ransom Demand 6
Malware Composition 7
Basic Static Analysis 8
Fingerprinting the Malware 8
Linked Libraries and imports. 10
Signature-based Detection 11
String Search- using detect-it-easy(die) 12
Basic Dynamic Analysis 14
Inetsim On 14
Inetsim Off 15
TCPView 15
WireShark 16
ProcessHacker 17
Procmon 19
Procdot 19
Regshot 20
Advanced Static Analysis 24
Advanced Dynamic Analysis 33
With INETSIM ON 33
..... 33
Modifying the ZF flag 39
Indicators of Compromise 41
Network Indicators 41
Host-based Indicators 43
Rules & Signatures 47

Strings 47
Signatures 47
CAPA 47
Appendices 51
Yara Rules 51
Decompiled Code Snippets 52

Executive Summary

MD5 hash	db349b97c37d22f5ea1d1841e3c89eb4
SHA256 hash	24d004a104d4d54034dbcffc2a4b19a11f39008a575aa614ea04703480b1022c

WannaCry is a ransomware-worm hybrid first detected on May 12th, 2017. It is a sophisticated malware strain that targets the Microsoft Windows operating system, exploiting the SMBv1 protocol vulnerability (EternalBlue). WannaCry consists of two main components: an SMB exploit for rapid propagation and a ransomware payload for file encryption. Post-infection symptoms include the encryption of files with specific extensions, displaying a ransom note demanding payment in Bitcoin, and widespread network propagation resulting in significant operational disruptions.

The malware gained notoriety for its global impact, affecting over 200,000 computers across 150 countries, with notable incidents in healthcare, telecommunications, and transportation sectors. Infection indicators are frequent SMB traffic on port 445, unexpected file encryption, and the presence of a "@WanaDecryptor@.exe" file in the system directories.

YARA signature rules for detection and mitigation have been compiled in Appendix A. Details of network indicators, including callback URLs are listed in Appendix B. The malware sample and associated hashes have been extensively analyzed and shared with cybersecurity communities and platforms such as VirusTotal for broader examination and defense coordination.

High-Level Technical Summary

Phase 1: Initial Contact and Kill Switch Check

- **Kill Switch**: The ransomware begins by attempting to connect to hxxp[://]www[.]iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea[.]com (a kill switch URL). The domain's status determines the malware's subsequent behavior.
    - If the domain is active and responsive, WannaCry terminates, halting the infection process.
    - If the domain is unresponsive, indicating the kill switch is not activated, the ransomware proceeds with the attack.
- **Service Creation**: In the absence of a kill switch activation, WannaCry installs itself as a Windows service:
    - Service Name: **mssecsvc2.0**
    - Display Name: **Microsoft Security Center (2.0) Service**
    - Binary Path: **<PATH_TO_WANNACRY>\Ransomware.wannacry.exe -m security**

Phase 2: System Modification and Propagation

- **Task Scheduler Manipulation**: WannaCry checks for the existence of **tasksche.exe** in the Windows directory.
    - If present, it renames the legitimate Task Scheduler executable and replaces it with its copy.
- **Initialisation Process**: A new process is created with the argument **C:\Windows\tasksche.exe** . This step marks the initialization of the ransomware's payload.
- **Directory and Service Setup**: The malware then:
    - Creates a directory **C:\ProgramData**byvtbrpmgfwu791 for persistence.
    - Copies itself to this new directory.
    - Establishes a service pointing to the copied executable to ensure it starts upon system reboot.
- **Payload Dropping**: The service drops various components:
    - Executable files (.exe)
    - Dynamic-link libraries (.dll)
    - Cryptographic key material
    - Bitcoin wallet addresses

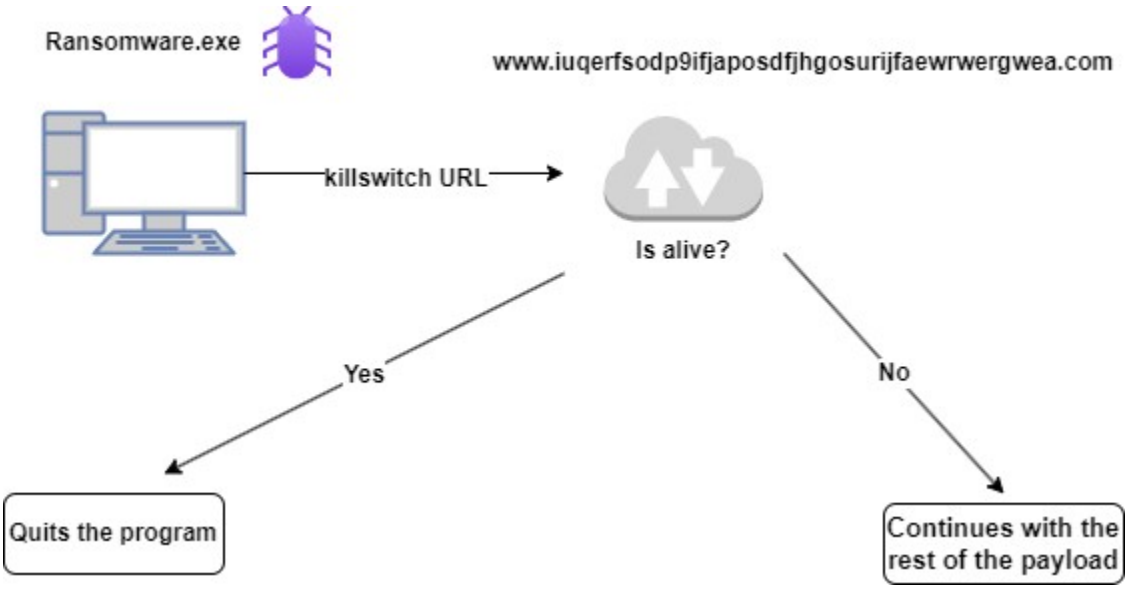
These components are saved in the **C:\ProgramData\<random_string>** directory.

Phase 3: Encryption and Ransom Demand

- **File Encryption**: WannaCry creates multiple threads to encrypt important files on the system, utilizing strong encryption algorithms to lock user data.
- **System Changes**: The ransomware alters the desktop wallpaper and displays instructions on how users can pay the ransom to potentially regain access to their encrypted files.
- **Spread Mechanism**: Concurrently, WannaCry continues to scan and propagate, attempting to infect other vulnerable systems within the network using the EternalBlue exploit.

Throughout these phases, WannaCry demonstrates capabilities typical of advanced ransomware: self-propagation, anti-termination checks, persistent installation, and a robust encryption mechanism, all aimed at maximizing impact and ransom payment probability.

Following is a simple diagram the shows how the malware gets executed in a machine.



Malware Composition

WannaCry consists of the following components:

File Name	SHA256 Hash
Ransomware.wannacry.exe	24d004a104d4d54034dbcffc2a4b19a11f39008a575aa614ea04703480b1022c
tasksche.exe	ED01EBFBC9EB5BBEA545AF4D01BF5F1071661840480439C6E5BABE8E080E41AA
@WanaDecryptor@.exe	B9C5D4339809E0AD9A00D4D3DD26FDF44A32819A54ABF846BB9B560D81391C25
taskdl.exe	4A468603FDCB7A2EB5770705898CF9EF37AADE532A7964642ECD705A74794B79

Ransomware.wannacry.exe

The initial binary file that is executed with admin privileges that checks connection with the killswitch URL.

tasksche.exe

Started through cmd.exe which creates the hidden directory in C:\ProgramData\byvtbrpmgfwu791 that contains the rest of the

@WanaDecryptor@.exe

This file starts the persistent window that provides information on what happened to the computer including the countdown and payment method to recover the files

taskdl.exe

Used for temporarily cleaning file programs.

Basic Static Analysis

This analysis is performed to extract information about the software's purpose, functionality, and potential impact.

Fingerprinting the Malware

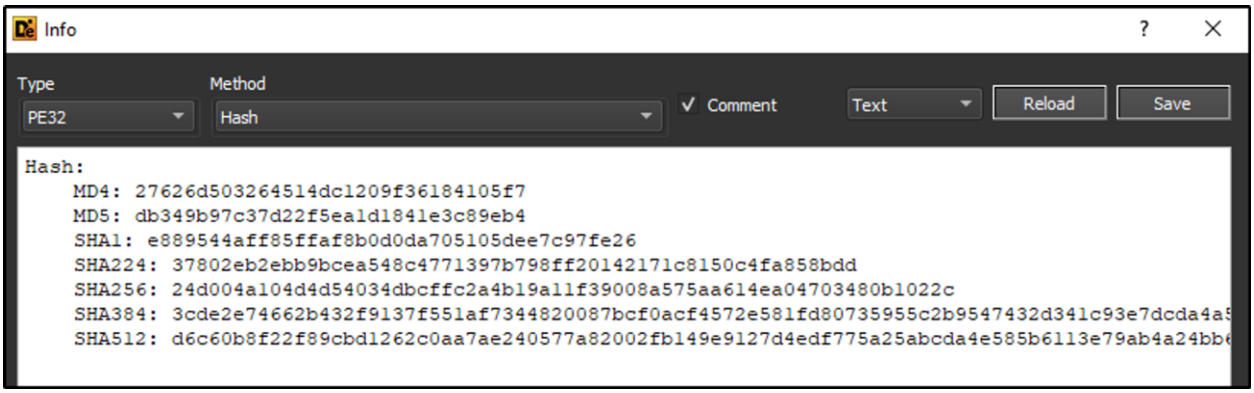


Figure 1 die- file hash

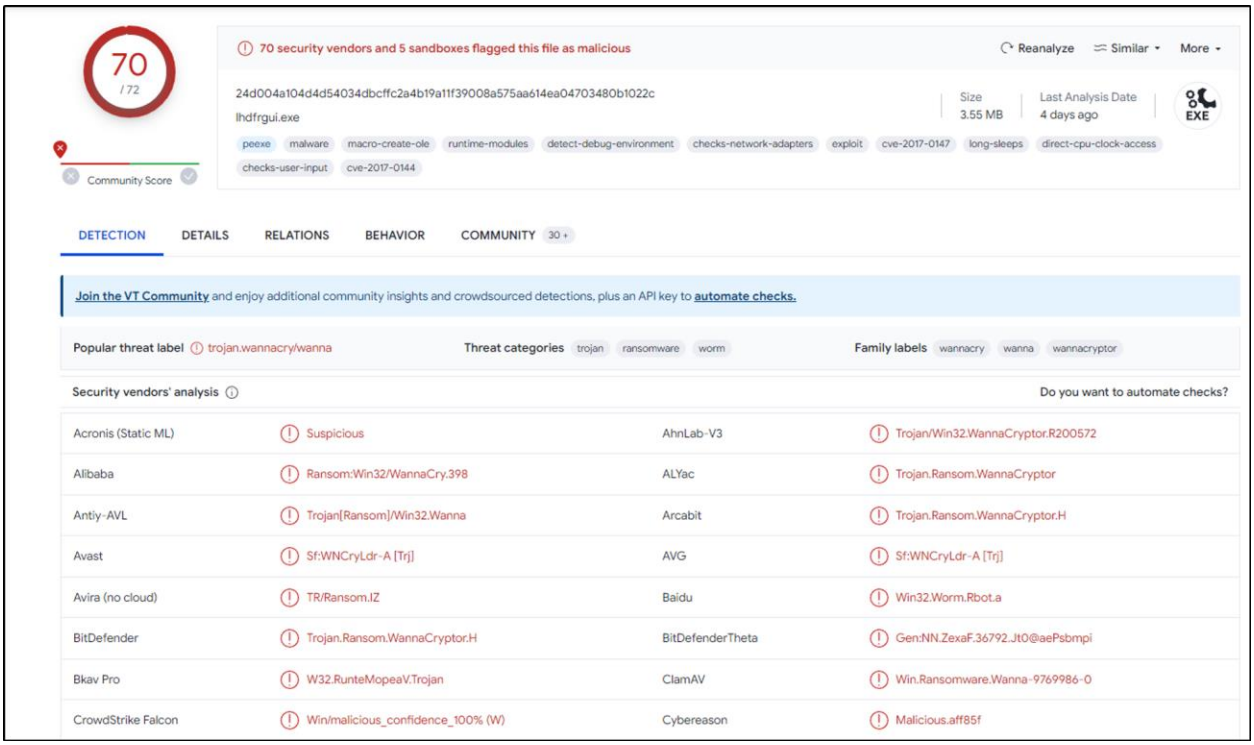


Figure 2 VirusTotal

property	value
footprint > 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 00
first-bytes > text	M Z .. .. @ .. ..
file > size	3723264 bytes
entropy	7.964
signature	Microsoft Visual C++ v6.0
tooling	Visual Studio 6.0
file-type	executable
cpu	32-bit
subsystem	**GUI**
file-version	6.1.7601.17514 (win7sp1_rtm.101119-1850)
description	Microsoft® Disk Defragmenter
stamps	
compiler-stamp	Sat Nov 20 09:03:08 2010 | UTC
debug > stamp	n/a
resource-stamp	n/a
import-stamp	n/a
export-stamp	n/a
file-names	
export	n/a
debug	n/a
version	lhdfrgui.exe
manifest	n/a
.NET > module	n/a

Figure 3 pestudio- basic info

The malware contains files that are embedded within the program.

indicator (36)	detail
file > embedded	signature: executable, location: .data, offset: 0x0000B020, size: 5263716 (bytes)
file > embedded	signature: executable, location: .data, offset: 0x0000F080, size: 5297524 (bytes)
file > embedded	signature: executable, location: .rsrc, offset: 0x000320A4, size: 3514368 (bytes)
file > extension > count	159
libraries > flag > name	Windows Socket Library
libraries > flag > name	IP Helper API
libraries > flag > name	Internet Extensions for Win32 Library
imports > flag > count	28
string > size > suspicious	2039 bytes
string > size > suspicious	1403 bytes
string > size > suspicious	2693 bytes
string > size > suspicious	3926 bytes
string > size > suspicious	1554 bytes
string > size > suspicious	1430 bytes
string > size > suspicious	2988 bytes
resource > size	R.1831, 3514368 bytes
resources > file-ratio	94.41%
file > checksum	0x00000000
groups > API	synchronization, execution, file, resource, dynamic-library, memory, reconnais...
string > URL	http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com
mitre > technique	T1057, T1105, T1124, T1497, T1106, T1569, T1543, T1027, T1082, T1001, T1083, T10...
file > entropy	7.964
file > signature	Microsoft Visual C++ v6.0
file > footprint	24D004A104D4D54034DBCFFC2A4B19A11F39008A575AA614EA04703480B1022C
file > size	3723264 bytes
rich-header > checksum	0xC33D5D11
rich-header > offset	0x00000080
rich-header > footprint	D4496034DE1F5AF97B361FCDC86EB5D939978830DFF8BF01B6AB3C93961AA425
file > tooling	Visual Studio 6.0
security > protection	data-execution-prevention (DEP) > OFF
security > protection	control-flow-guard (CFG) > OFF
security > protection	address-space-layout-randomization (ASLR) > OFF
file-name > version	lhdfrgui.exe
security > protection	code-integrity (CI) > OFF
file > subsystem	GUI
imports > ordinal > count	13

Figure 4 pestudio- indicators

property	value
footprint > sha256	2F3FC51546ADA848DFC8E775554C0DE3689D6FAE7BA4BF3D40E3C8DEC68B277B
location	.rsrc:0x0038C0A4
file-type	executable
language	English-US
code-page	Unicode UTF-16, little endian
CompanyName	Microsoft Corporation
FileDescription	Microsoft® Disk Defragmenter
FileVersion	6.1.7601.17514 (win7sp1_rtm.101119-1850)
InternalName	lhdfrgui.exe
LegalCopyright	© Microsoft Corporation. All rights reserved.
OriginalFilename	**lhdfrgui.exe**
ProductName	Microsoft® Windows® Operating System
ProductVersion	6.1.7601.17514

Figure 5 pestudio- version

Linked Libraries and imports.

library (7)	duplicate (0)	flag (3)	first-thunk-original (INT)	first-thunk (IAT)	type (1)	imports (91)	group	description
KERNEL32.dll	-	-	0x0000A2B0	0x0000A030	implicit	32	-	Windows NT BASE API Client
ADVAPI32.dll	-	-	0x0000A280	0x0000A000	implicit	11	-	Advanced Windows 32 Base API
WS2_32.dll	-	x	0x0000A3C4	0x0000A144	implicit	13	network	Windows Socket Library
MSVCP60.dll	-	-	0x0000A334	0x0000A0B4	implicit	2	-	Windows C Runtime Library
iphlpapi.dll	-	x	0x0000A3FC	0x0000A17C	implicit	2	network	IP Helper API
WININET.dll	-	x	0x0000A3B4	0x0000A134	implicit	3	network	Internet Extensions for Win32 Library
MSVCRT.dll	-	-	0x0000A340	0x0000A0C0	implicit	28	-	Microsoft C Runtime Library

Figure 6 pestudio- libraries

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

Figure 7 pestudio imports

Signature-based Detection

```
C:\Users\igor\Desktop
λ capa.exe Ransomware.wannacry.exe.malz
┌─────────────────────┬──────────────────────
```

Capability	Namespace
reference analysis tools strings	anti-analysis
check for time delay via QueryPerformanceCounter	anti-analysis/anti-debugging/debugger-detection
contain obfuscated stackstrings	anti-analysis/obfuscation/string/stackstring
receive data (5 matches)	communication
send data (5 matches)	communication
connect to URL	communication/http/client
get socket status	communication/socket
initialize Winsock library	communication/socket
set socket configuration	communication/socket
create UDP socket (4 matches)	communication/socket/udp/send
act as TCP client	communication/tcp/client
generate random numbers via WinAPI	data-manipulation/prng
extract resource via kernel32 functions	executable/resource
contain an embedded PE file	executable/subfile/pe
get file size	host-interaction/file-system/meta
move file	host-interaction/file-system/move
read file on Windows	host-interaction/file-system/read
get number of processors	host-interaction/hardware/cpu
terminate process	host-interaction/process/terminate
run as service	host-interaction/service
create service	host-interaction/service/create
modify service	host-interaction/service/modify
start service	host-interaction/service/start
create thread (4 matches)	host-interaction/thread/create
terminate thread	host-interaction/thread/terminate
link function at runtime on Windows	linking/runtime-linking
linked against ZLIB	linking/static/zlib
inspect section memory permissions	load-code/pe
persist via Windows service	persistence/service

Figure 9 capa-2

String Search- using detect-it-easy(die)

The table is a summary of strings that can be used as indictors of compromise.

%s -m security	cmd.exe /c "%s"	icacls . /grant Everyone:F /T /C /Q
tasksche.exe	lhdfrgui.exe	diskpart.exe
http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com	\\192.168.56.20\IPC$	mssecsvc2.0
Microsoft Security Center (2.0) Service	%s -m security	C:\%s\qeriuwjhrf
tasksche.exe	t.wnry	\\192.168.56.20\IPC$
WNcry@2ol7	InternetCloseHandle	InternetOpenUrlA
InternetOpenA		

```
0002e616    11  U   Windows 2000 2195
0002e63a    10  U   Windows 2000 5.0
0002e68c    14  U   \\192.168.56.20\IPC$
```

Figure 10 strings -1

```
0000a7b4    13  A   InternetCloseHandle
0000a7ca    10  A   InternetOpenUrlA
0000a7de    0d  A   InternetOpenA
```

Figure 11 strings -2

```
00041130    35  A   Microsoft Enhanced RSA and AES Cryptographic Provider
00041168    0b  A   CryptGenKey
00041174    0c  A   CryptDecrypt
00041184    0c  A   CryptEncrypt
00041194    0f  A   CryptDestroyKey
000411a4    0e  A   CryptImportKey
000411b4    14  A   CryptAcquireContextA
0004149c    08  U   %s\Intel
000414b0    0e  U   %s\ProgramData
000414d0    0f  A   cmd.exe /c "%s"
000414e4    22  A   115p7UMMngoj1pMvkpHijcRdfJNXj6LrLn
00041508    22  A   12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw
0004152c    22  A   13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94
00041558    23  A   Global\MsWinZonesCacheCounterMutexA
0004157c    0c  A   tasksche.exe
0004158c    09  A   TaskStart
00041598    06  A   t.wnry
000415a0    23  A   icacls . /grant Everyone:F /T /C /Q
000415c4    0b  A   attrib +h .
000415d0    0a  A   WNcry@2ol7
00041600    13  A   GetNativeSystemInfo
0004162c    0f  A   .?AVexception@@
0004163c    14  A   incompatible version
00041654    0c  A   buffer error
00041664    13  A   insufficient memory
```

Figure 12 strings -3

```
000312b8    09  A   \\%s\IPC$
000312c4    2a  A   Microsoft Base Cryptographic Provider v1.0
000312f0    0b  A   %d.%d.%d.%d
000312fc    0b  A   mssecsvc2.0
00031308    27  A   Microsoft Security Center (2.0) Service
00031330    0e  A   %s -m security
00031344    10  A   C:\%s\qeriuwjhrf
00031358    08  A   C:\%s\%s
00031364    07  A   WINDOWS
0003136c    0c  A   tasksche.exe
00031380    0b  A   CloseHandle
0003138c    09  A   WriteFile
00031398    0b  A   CreateFileA
000313a4    0e  A   CreateProcessA
000313b4    0c  U   kernel32.dll
000313d0    38  A   http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com
000320f1    28  A   !This program cannot be run in DOS mode.
```

Figure 13 strings- 4

Basic Dynamic Analysis

Basic dynamic analysis involves observing and analyzing the behavior of a suspected malware sample while it is running in a controlled environment. This type of analysis is used to gather information about what the malware does when executed, which might

Inetsim On

```
remnux@remnux:~$ inetsim
INetSim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg
Using log directory:      /var/log/inetsim/
Using data directory:     /var/lib/inetsim/
Using report directory:   /var/log/inetsim/report/
Using configuration file: /etc/inetsim/inetsim.conf
Parsing configuration file.
Configuration file parsed successfully.
=== INetSim main process started (PID 1440) ===
Session ID:     1440
Listening on:   10.0.0.3
Real Date/Time: 2023-11-20 22:11:44
Fake Date/Time: 2023-11-20 22:11:44 (Delta: 0 seconds)
 Forking services...
  * dns_53_tcp_udp - started (PID 1450)
  * smtp_25_tcp - started (PID 1453)
  * pop3_110_tcp - started (PID 1455)
  * https_443_tcp - started (PID 1452)
  * pop3s_995_tcp - started (PID 1456)
  * smtps_465_tcp - started (PID 1454)
  * http_80_tcp - started (PID 1451)
  * ftps_990_tcp - started (PID 1458)
  * ftp_21_tcp - started (PID 1457)
 done.
Simulation running.
```

Figure 14 inetsim- on

The malware made a DNS request and connected to the domain. It then ended the connection

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
8	27.199145432	10.0.0.1	239.192.152.143	LSD	178	
9	31.203439405	10.0.0.1	239.192.152.143	LSD	178	
10	31.203977273	10.0.0.1	239.192.152.143	LSD	178	
11	53.452384936	10.0.0.5	10.0.0.3	DNS	109	Standard query 0xc25e A www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com
12	53.461597942	10.0.0.3	10.0.0.5	DNS	125	Standard query response 0xc25e A www.iuqerfsodp9ifjaposdfjhgosurijfaewrwe…
13	53.474812427	10.0.0.5	10.0.0.3	TCP	66	1104 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1
14	53.474834625	10.0.0.3	10.0.0.5	TCP	66	80 → 1104 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=…
15	53.475291716	10.0.0.5	10.0.0.3	TCP	60	1104 → 80 [ACK] Seq=1 Ack=1 Win=262144 Len=0
16	53.480658393	10.0.0.5	10.0.0.3	HTTP	154	GET / HTTP/1.1
17	53.480694585	10.0.0.3	10.0.0.5	TCP	54	80 → 1104 [ACK] Seq=1 Ack=101 Win=64256 Len=0
18	53.495211141	10.0.0.3	10.0.0.5	TCP	204	80 → 1104 [PSH, ACK] Seq=1 Ack=101 Win=64256 Len=150 [TCP segment of a re…
19	53.495557704	10.0.0.5	10.0.0.3	TCP	60	1104 → 80 [ACK] Seq=101 Ack=151 Win=261888 Len=0
20	53.495572926	10.0.0.3	10.0.0.5	HTTP	312	HTTP/1.1 200 OK (text/html)
21	53.495831813	10.0.0.5	10.0.0.3	TCP	60	1104 → 80 [ACK] Seq=101 Ack=409 Win=261632 Len=0
22	53.495831951	10.0.0.5	10.0.0.3	TCP	60	1104 → 80 [FIN, ACK] Seq=101 Ack=409 Win=261632 Len=0
23	53.495941257	10.0.0.5	10.0.0.3	TCP	60	1104 → 80 [RST, ACK] Seq=102 Ack=409 Win=0 Len=0

Figure 15 wireshark- dns and http

```
11 53.452384936  10.0.0.5  10.0.0.3  DNS  109 Standard query 0xc25e A www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com
12 53.461597942  10.0.0.3  10.0.0.5  DNS  125 Standard query response 0xc25e A www.iuqerfsodp9ifjaposdfjhgosurijfaewrwe…

▸ Frame 11: 109 bytes on wire (872 bits), 109 bytes captured (872 bits) on interface enp0s3, id 0
▸ Ethernet II, Src: PcsCompu_5e:7d:87 (08:00:27:5e:7d:87), Dst: PcsCompu_cd:2e:c6 (08:00:27:cd:2e:c6)
▸ Internet Protocol Version 4, Src: 10.0.0.5, Dst: 10.0.0.3
▸ User Datagram Protocol, Src Port: 58558, Dst Port: 53
▾ Domain Name System (query)
    Transaction ID: 0xc25e
  ▸ Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ▾ Queries
    ▸ www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com: type A, class IN
    [Response In: 12]
```

Figure 16 wireshark- domain queried

Inetsim Off

Execute the malware with administrative privileges.

TCPView

An http connection was initiated as soon as the malware was executed to check for connectivity.

TCPView - Sysinternals: www.sysinternals.com

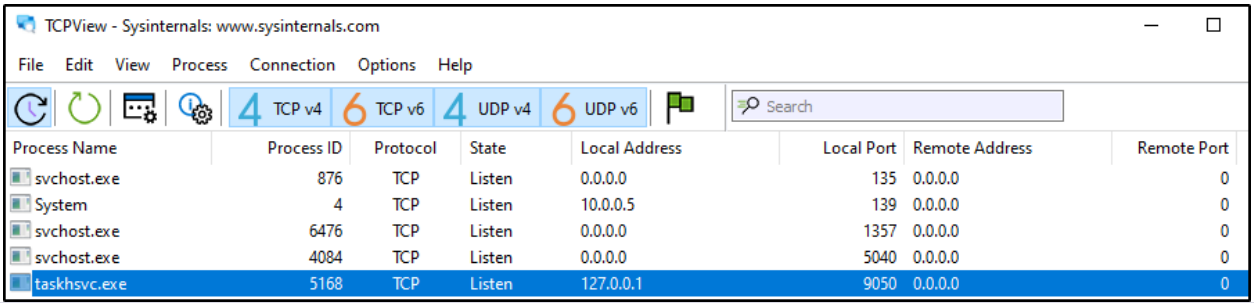
File Edit View Process Connection Options Help

4 TCP v4 6 TCP v6 4 UDP v4 6 UDP v6 Search

Process Name	Process ID	Protocol	State	Local Address	Local Port	Remote Address	Remote Port
svchost.exe	876	TCP	Listen	0.0.0.0	135	0.0.0.0	0
System	4	TCP	Listen	10.0.0.5	139	0.0.0.0	0
svchost.exe	6476	TCP	Listen	0.0.0.0	1357	0.0.0.0	0
svchost.exe	4084	TCP	Listen	0.0.0.0	5040	0.0.0.0	0
Ransomware.wannacr...	5860	TCP	Syn Sent	10.0.0.5	35472	10.0.0.3	80
lsass.exe	644	TCP	Listen	0.0.0.0	49664	0.0.0.0	0

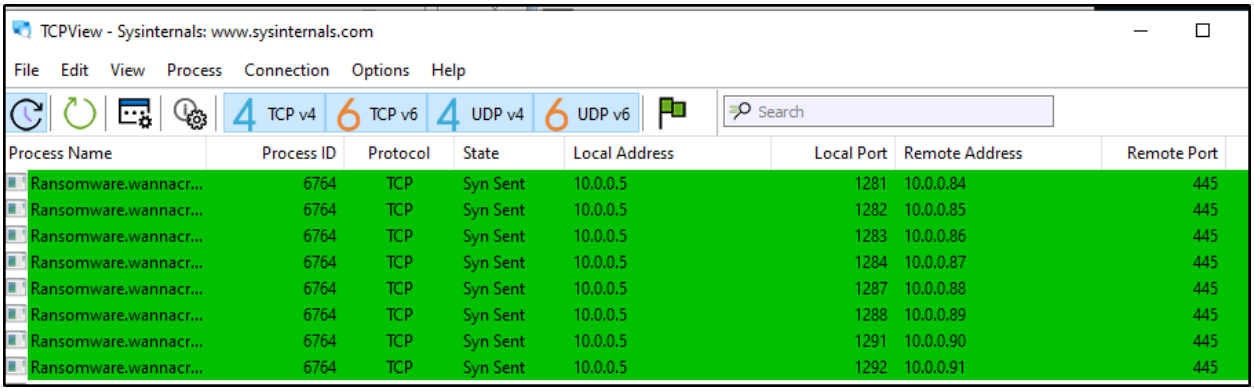
Figure 17 TCPView- Initial call-out to the domain

Once checked that the domain



Process Name	Process ID	Protocol	State	Local Address	Local Port	Remote Address	Remote Port
svchost.exe	876	TCP	Listen	0.0.0.0	135	0.0.0.0	0
System	4	TCP	Listen	10.0.0.5	139	0.0.0.0	0
svchost.exe	6476	TCP	Listen	0.0.0.0	1357	0.0.0.0	0
svchost.exe	4084	TCP	Listen	0.0.0.0	5040	0.0.0.0	0
taskhsvc.exe	5168	TCP	Listen	127.0.0.1	9050	0.0.0.0	0

Figure 18 TCPView- local port on listening mode

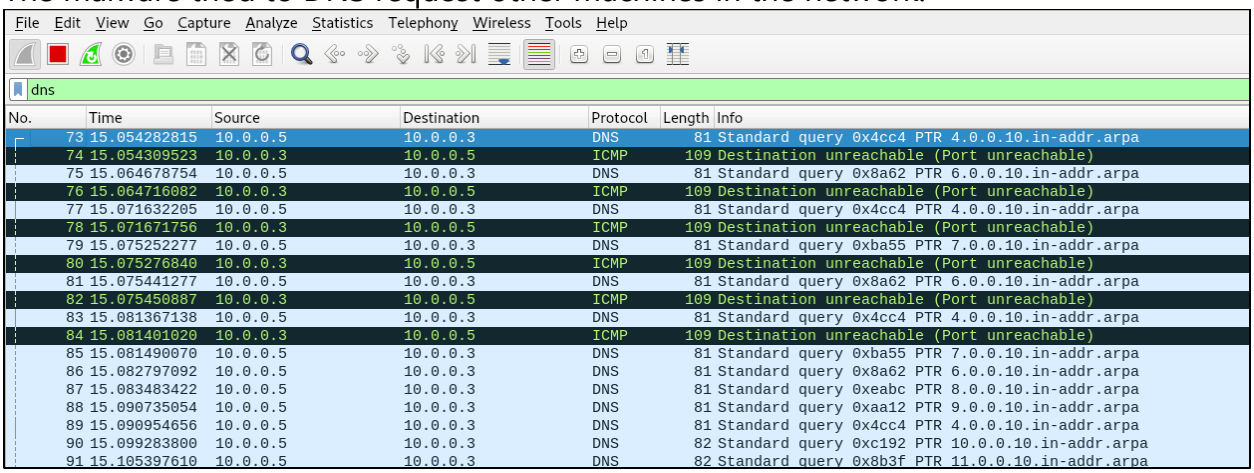


Process Name	Process ID	Protocol	State	Local Address	Local Port	Remote Address	Remote Port
Ransomware.wannacr...	6764	TCP	Syn Sent	10.0.0.5	1281	10.0.0.84	445
Ransomware.wannacr...	6764	TCP	Syn Sent	10.0.0.5	1282	10.0.0.85	445
Ransomware.wannacr...	6764	TCP	Syn Sent	10.0.0.5	1283	10.0.0.86	445
Ransomware.wannacr...	6764	TCP	Syn Sent	10.0.0.5	1284	10.0.0.87	445
Ransomware.wannacr...	6764	TCP	Syn Sent	10.0.0.5	1287	10.0.0.88	445
Ransomware.wannacr...	6764	TCP	Syn Sent	10.0.0.5	1288	10.0.0.89	445
Ransomware.wannacr...	6764	TCP	Syn Sent	10.0.0.5	1291	10.0.0.90	445
Ransomware.wannacr...	6764	TCP	Syn Sent	10.0.0.5	1292	10.0.0.91	445

Figure 19 TCPView- SMB requests to other hosts in the network

WireShark

The malware tried to DNS request other machines in the network.



No.	Time	Source	Destination	Protocol	Length	Info
73	15.054282815	10.0.0.5	10.0.0.3	DNS	81	Standard query 0x4cc4 PTR 4.0.0.10.in-addr.arpa
74	15.054309523	10.0.0.3	10.0.0.5	ICMP	109	Destination unreachable (Port unreachable)
75	15.064678754	10.0.0.5	10.0.0.3	DNS	81	Standard query 0x8a62 PTR 6.0.0.10.in-addr.arpa
76	15.064716082	10.0.0.3	10.0.0.5	ICMP	109	Destination unreachable (Port unreachable)
77	15.071632205	10.0.0.5	10.0.0.3	DNS	81	Standard query 0x4cc4 PTR 4.0.0.10.in-addr.arpa
78	15.071671756	10.0.0.3	10.0.0.5	ICMP	109	Destination unreachable (Port unreachable)
79	15.075252277	10.0.0.5	10.0.0.3	DNS	81	Standard query 0xba55 PTR 7.0.0.10.in-addr.arpa
80	15.075276840	10.0.0.3	10.0.0.5	ICMP	109	Destination unreachable (Port unreachable)
81	15.075441277	10.0.0.5	10.0.0.3	DNS	81	Standard query 0x8a62 PTR 6.0.0.10.in-addr.arpa
82	15.075450887	10.0.0.3	10.0.0.5	ICMP	109	Destination unreachable (Port unreachable)
83	15.081367138	10.0.0.5	10.0.0.3	DNS	81	Standard query 0x4cc4 PTR 4.0.0.10.in-addr.arpa
84	15.081401020	10.0.0.3	10.0.0.5	ICMP	109	Destination unreachable (Port unreachable)
85	15.081490070	10.0.0.5	10.0.0.3	DNS	81	Standard query 0xba55 PTR 7.0.0.10.in-addr.arpa
86	15.082797092	10.0.0.5	10.0.0.3	DNS	81	Standard query 0x8a62 PTR 6.0.0.10.in-addr.arpa
87	15.083483422	10.0.0.5	10.0.0.3	DNS	81	Standard query 0xeabc PTR 8.0.0.10.in-addr.arpa
88	15.090735054	10.0.0.5	10.0.0.3	DNS	81	Standard query 0xaa12 PTR 9.0.0.10.in-addr.arpa
89	15.090954656	10.0.0.5	10.0.0.3	DNS	81	Standard query 0x4cc4 PTR 4.0.0.10.in-addr.arpa
90	15.099283800	10.0.0.5	10.0.0.3	DNS	82	Standard query 0xc192 PTR 10.0.0.10.in-addr.arpa
91	15.105397610	10.0.0.5	10.0.0.3	DNS	82	Standard query 0x8b3f PTR 11.0.0.10.in-addr.arpa

Figure 20wireshark- dns traffic

ProcessHacker

A new process was created upon execution of the malware.

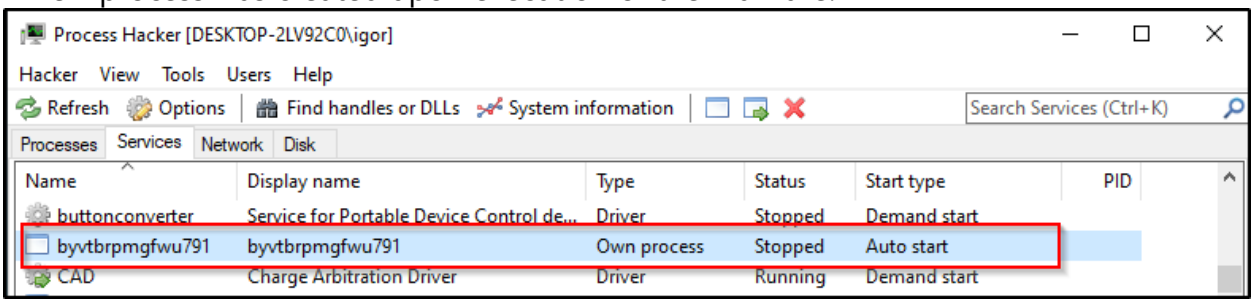


Figure 21 Processhacker- New process created.

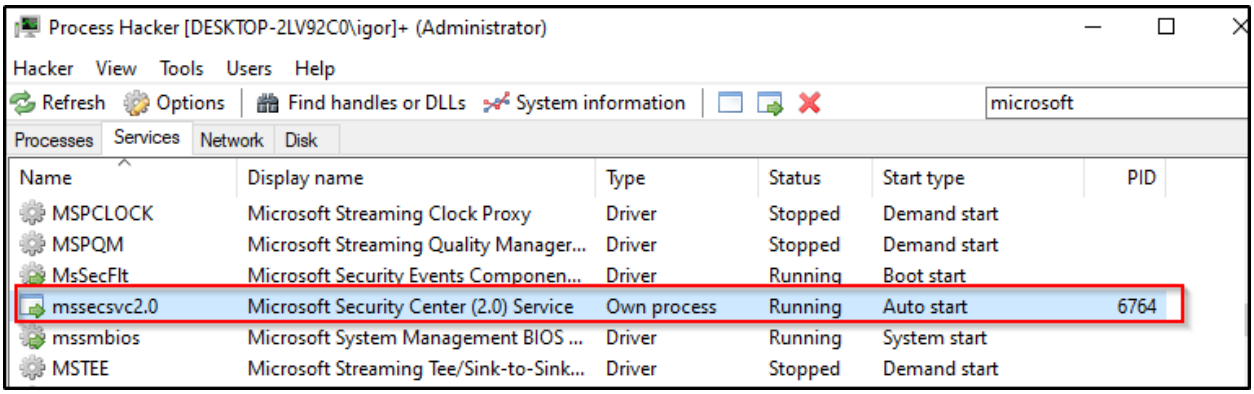
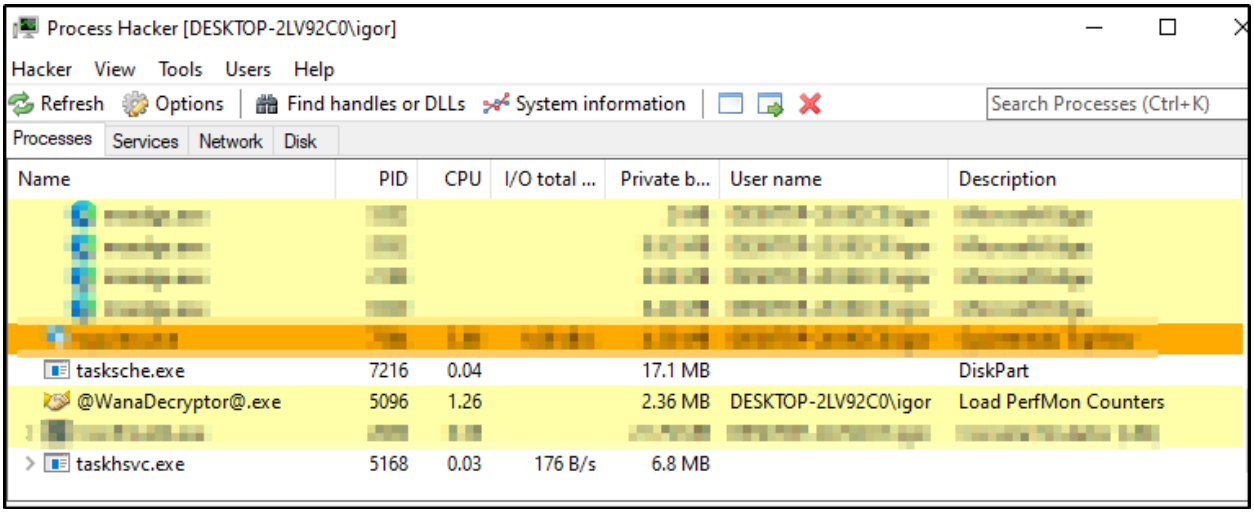


Figure 22 Proceshacker- mssecsvc2.0 started



Staging directory of the malware: C:\ProgramData\byvtbrpmgfwu791

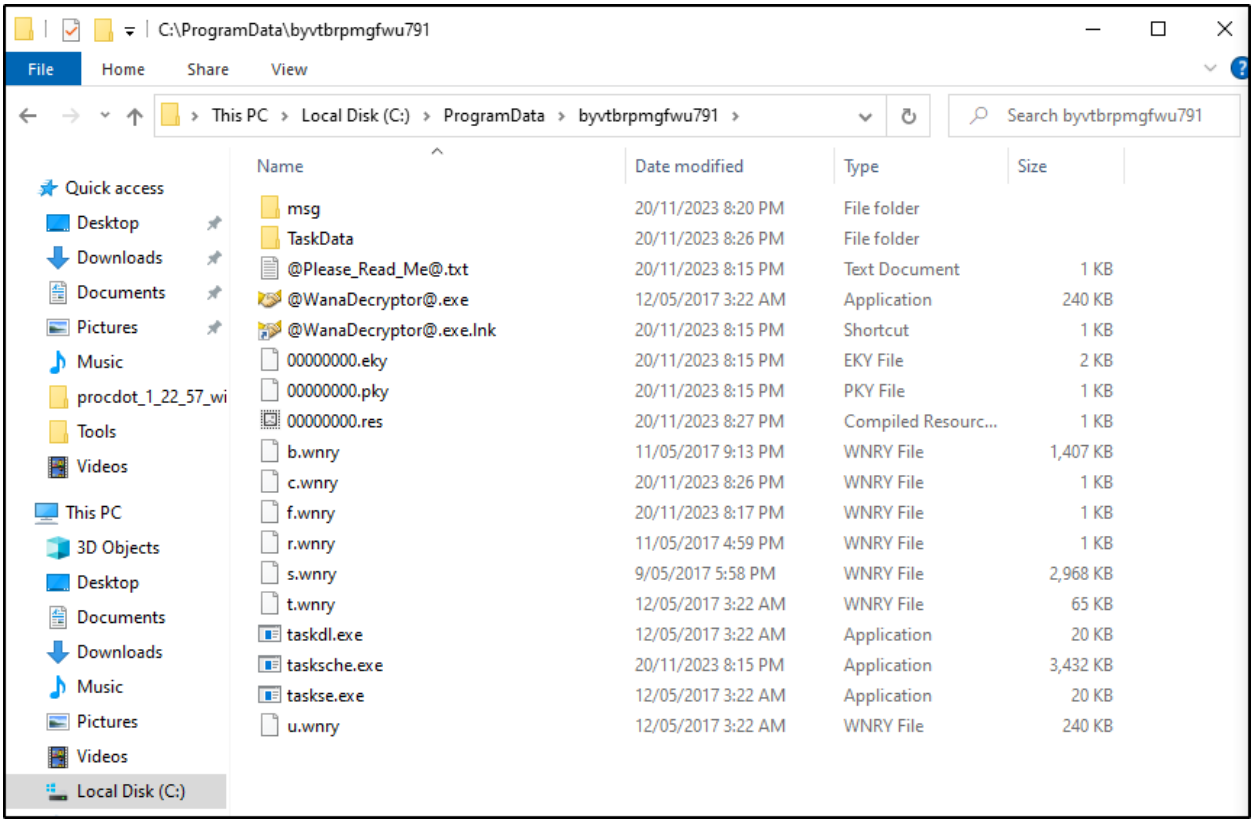


Figure 23 malware resources

```
PS C:\ProgramData\byvtbrpmgfwu791> Get-FileHash  *.exe

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
SHA256          B9C5D4339809E0AD9A00D4D3DD26FDF44A32819A54ABF846BB9B560D81391C25       C:\ProgramData\byvtbrpmgfwu791\@WanaDecryptor@.exe
SHA256          4A468603FDCB7A2EB5770705898CF9EF37AADE532A7964642ECD705A74794B79       C:\ProgramData\byvtbrpmgfwu791\taskdl.exe
SHA256          ED01EBFBC9EB5BBEA545AF4D01BF5F10716618404804 39C6E5BABE8E080E41AA      C:\ProgramData\byvtbrpmgfwu791\tasksche.exe
SHA256          2CA2D550E603D74DEDDA03156023135B38DA3630CB014E3D00B1263358C5F00D       C:\ProgramData\byvtbrpmgfwu791\taskse.exe
```

Figure 24 hash of the embedded binaries

Procmon

Process Tree

☐ Only show processes still running at end of current trace
☑ Timelines cover displayed events only

Process	Description	Image Path	Life Time	Company	Owner	Command
svchost.exe (7868)	Host Process for ...	C:\Windows\syst...		Microsoft Corporat...	NT AUTHORITY\...	C:\Windows\system32\svchost.exe -k netsvcs -p -s.
GoogleUpdate.exe (6520)	Google Installer	C:\Program Files (...		Google LLC	NT AUTHORITY\...	"C:\Program Files (x86)\Google\Update\GoogleUp..
svchost.exe (1776)	Host Process for ...	C:\Windows\syst...		Microsoft Corporat...	NT AUTHORITY\...	C:\Windows\system32\svchost.exe -k LocalServic..
svchost.exe (7848)	Host Process for ...	C:\Windows\syst...		Microsoft Corporat...	NT AUTHORITY\...	C:\Windows\system32\svchost.exe -k netsvcs -p -s.
Ransomware.wannacry.exe	Microsoft® Disk D...	C:\Users\igor\De...		Microsoft Corporat...	NT AUTHORITY\...	C:\Users\igor\Desktop\Ransomware.wannacry.exe.
cmd.exe (5260)	Windows Comma...	C:\Windows\syst...		Microsoft Corporat...	NT AUTHORITY\...	cmd.exe /c "C:\ProgramData\byvtbrpmgfwu791\ta..
tasksche.exe (7216)	DiskPart	C:\ProgramData\...		Microsoft Corporat...	NT AUTHORITY\...	C:\ProgramData\byvtbrpmgfwu791\tasksche.exe
attrib.exe (8120)	Attribute Utility	C:\Windows\Sys...		Microsoft Corporat...	NT AUTHORITY\...	attrib +h .
Conhost.exe (666	Console Window ...	C:\Windows\Syst...		Microsoft Corporat...	NT AUTHORITY\...	\??\C:\Windows\system32\conhost.exe 0xffffffff -F..
icacls.exe (4692)		C:\Windows\Sys...		Microsoft Corporat...	NT AUTHORITY\...	icacls . /grant Everyone:F /T /C /Q
Conhost.exe (605	Console Window ...	C:\Windows\Syst...		Microsoft Corporat...	NT AUTHORITY\...	\??\C:\Windows\system32\conhost.exe 0xffffffff -F..
taskdl.exe (5688)	SQL Client Config...	C:\ProgramData\...		Microsoft Corporat...	NT AUTHORITY\...	taskdl.exe
cmd.exe (8116)	Windows Comma...	C:\Windows\Sys...		Microsoft Corporat...	NT AUTHORITY\...	C:\Windows\system32\cmd.exe /c 526417005401..
Conhost.exe (527	Console Window ...	C:\Windows\Syst...		Microsoft Corporat...	NT AUTHORITY\...	\??\C:\Windows\system32\conhost.exe 0xffffffff -F..
cscript.exe (4796)	Microsoft ® Conso...	C:\Windows\Sys...		Microsoft Corporat...	NT AUTHORITY\...	cscript.exe //nologo m.vbs
taskdl.exe (7052)	SQL Client Config...	C:\ProgramData\...		Microsoft Corporat...	NT AUTHORITY\...	taskdl.exe
taskdl.exe (5360)	SQL Client Config...	C:\ProgramData\...		Microsoft Corporat...	NT AUTHORITY\...	taskdl.exe
taskdl.exe (4828)	SQL Client Config...	C:\ProgramData\...		Microsoft Corporat...	NT AUTHORITY\...	taskdl.exe

Figure 25 processtree- processes created by the malware

Process Tree

☐ Only show processes still running at end of current trace
☑ Timelines cover displayed events only

Process	Description	Image Path	Life Time	Company	Owner	Command
@WanaDecryptor@.exe (4828)	Load PerfMon Co...	C:\ProgramData\...		Microsoft Corporat...	NT AUTHORITY\...	@WanaDecryptor@.exe co
taskhsvc.exe (5168)		C:\ProgramData\...			NT AUTHORITY\...	TaskData\Tor\taskhsvc.exe
Conhost.exe (5124)	Console Window ...	C:\Windows\Syst...		Microsoft Corporat...	NT AUTHORITY\...	\??\C:\Windows\system32\conhost.exe 0xffffffff -F...
cmd.exe (3128)	Windows Comma...	C:\Windows\Sys...		Microsoft Corporat...	NT AUTHORITY\...	cmd.exe /c start /b @WanaDecryptor@.exe vs
Conhost.exe (3164)	Console Window ...	C:\Windows\Syst...		Microsoft Corporat...	NT AUTHORITY\...	\??\C:\Windows\system32\conhost.exe 0xffffffff -F...
@WanaDecryptor@.exe (4256)	Load PerfMon Co...	C:\ProgramData\...		Microsoft Corporat...	NT AUTHORITY\...	@WanaDecryptor@.exe vs
cmd.exe (4160)	Windows Comma...	C:\Windows\Sys...		Microsoft Corporat...	NT AUTHORITY\...	cmd.exe /c vssadmin delete shadows /all /quiet & ...
Conhost.exe (7652)	Console Window ...	C:\Windows\Syst...		Microsoft Corporat...	NT AUTHORITY\...	\??\C:\Windows\system32\conhost.exe 0xffffffff -F...
WMIC.exe (5352)	WMI Commandlin...	C:\Windows\Sys...		Microsoft Corporat...	NT AUTHORITY\...	wmic shadowcopy delete
taskdl.exe (7328)	SQL Client Config...	C:\ProgramData\...		Microsoft Corporat...	NT AUTHORITY\...	taskdl.exe
taskse.exe (3656)	waitfor - wait/send...	C:\ProgramData\...		Microsoft Corporat...	NT AUTHORITY\...	taskse.exe C:\ProgramData\byvtbrpmgfwu791\@...
@WanaDecryptor@.exe (5096)	Load PerfMon Co...	C:\ProgramData\...		Microsoft Corporat...	DESKTOP-2LV92...	"C:\ProgramData\byvtbrpmgfwu791\@WanaDecry...
cmd.exe (7120)	Windows Comma...	C:\Windows\Sys...		Microsoft Corporat...	NT AUTHORITY\...	cmd.exe /c reg add HKLM\SOFTWARE\Microsoft\...
Conhost.exe (5484)	Console Window ...	C:\Windows\Syst...		Microsoft Corporat...	NT AUTHORITY\...	\??\C:\Windows\system32\conhost.exe 0xffffffff -F...
reg.exe (4744)	Registry Console ...	C:\Windows\Sys...		Microsoft Corporat...	NT AUTHORITY\...	reg add HKLM\SOFTWARE\Microsoft\Windows\...
taskdl.exe (7820)	SQL Client Config...	C:\ProgramData\...		Microsoft Corporat...	NT AUTHORITY\...	taskdl.exe
taskse.exe (3816)	waitfor - wait/send...	C:\ProgramData\...		Microsoft Corporat...	NT AUTHORITY\...	taskse.exe C:\ProgramData\byvtbrpmgfwu791\@...
@WanaDecryptor@.exe (1916)	Load PerfMon Co...	C:\ProgramData\...		Microsoft Corporat...	DESKTOP-2LV92...	"C:\ProgramData\byvtbrpmgfwu791\@WanaDecry...

Figure 26 processtree- processes created by the malware

Procdot

The output from procdot shows a graphical process and network flow diagram for the renaming of tasksche.exe.

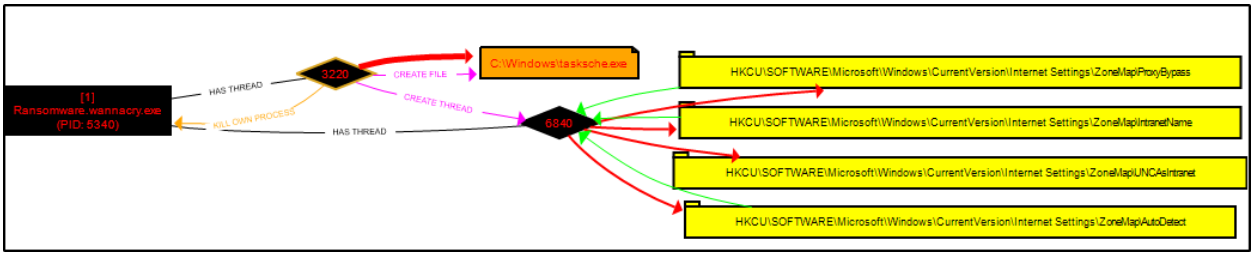


Figure 27 procdot- tasksche.exe creation

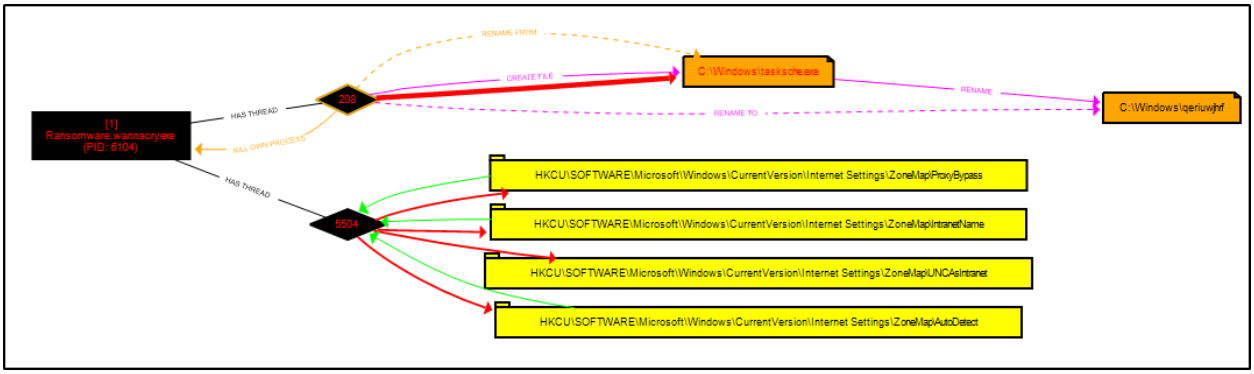


Figure 28 procdot- renamed as a system file

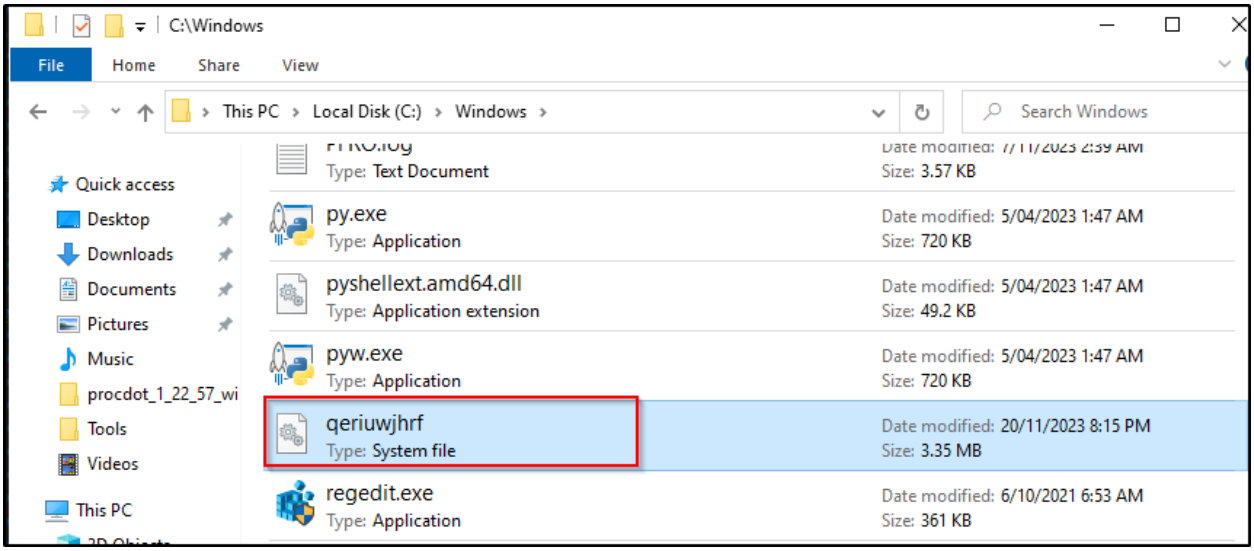


Figure 29 file location

Regshot

This tool will compare two snapshots of the machine, before and after the execution of the malware. Below shows keys and values that were added and modified to the Registry.

```
HKLM\SOFTWARE\WOW6432Node\WanaCrypt0r
HKLM\SYSTEM\ControlSet001\Services\byvtbrpmgfwu791
HKLM\SYSTEM\ControlSet001\Services\mssecsvc2.0
HKLM\SYSTEM\CurrentControlSet\Services\byvtbrpmgfwu791
HKLM\SYSTEM\CurrentControlSet\Services\mssecsvc2.0
```

Figure 30 regshot- registry keys added

```
\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Wallpapers\BackgroundHistoryPath0: "C:\ProgramData\_VM\background.png"
\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Wallpapers\BackgroundHistoryPath0: "C:\Users\igor\Desktop\@WanaDecryptor@.bmp"
```

Figure 31 regshot- registry values added

```
HKLM\SOFTWARE\WOW6432Node\Microsoft\EdgeUpdate\RetryAfter: 0x655CBB80
HKLM\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Run\byvtbrpmgfwu791: ""C:\ProgramData\byvtbrpmgfwu791\tasksche.exe""
HKLM\SOFTWARE\WOW6432Node\WanaCrypt0r\wd: "C:\ProgramData\byvtbrpmgfwu791"
HKLM\SYSTEM\ControlSet001\Services\bam\State\UserSettings\S-1-5-21-330533299-3180603657-300593454-1001\\Device\HarddiskVolume2\ProgramData\byvtbrpmgfwu791\@WanaDecryptor@.exe: 29
HKLM\SYSTEM\ControlSet001\Services\byvtbrpmgfwu791\Type: 0x00000010
HKLM\SYSTEM\ControlSet001\Services\byvtbrpmgfwu791\Start: 0x00000002
HKLM\SYSTEM\ControlSet001\Services\byvtbrpmgfwu791\ErrorControl: 0x00000001
HKLM\SYSTEM\ControlSet001\Services\byvtbrpmgfwu791\ImagePath: "cmd.exe /c "C:\ProgramData\byvtbrpmgfwu791\tasksche.exe""
HKLM\SYSTEM\ControlSet001\Services\byvtbrpmgfwu791\DisplayName: "byvtbrpmgfwu791"
HKLM\SYSTEM\ControlSet001\Services\byvtbrpmgfwu791\WOW64: 0x0000014C
HKLM\SYSTEM\ControlSet001\Services\byvtbrpmgfwu791\ObjectName: "LocalSystem"
HKLM\SYSTEM\ControlSet001\Services\mssecsvc2.0\Type: 0x00000010
HKLM\SYSTEM\ControlSet001\Services\mssecsvc2.0\Start: 0x00000002
HKLM\SYSTEM\ControlSet001\Services\mssecsvc2.0\ErrorControl: 0x00000001
HKLM\SYSTEM\ControlSet001\Services\mssecsvc2.0\ImagePath: "C:\Users\igor\Desktop\Ransomware.wannacry.exe -m security"
HKLM\SYSTEM\ControlSet001\Services\mssecsvc2.0\DisplayName: "Microsoft Security Center (2.0) Service"
HKLM\SYSTEM\ControlSet001\Services\mssecsvc2.0\WOW64: 0x0000014C
HKLM\SYSTEM\ControlSet001\Services\mssecsvc2.0\ObjectName: "LocalSystem"
HKLM\SYSTEM\ControlSet001\Services\mssecsvc2.0\FailureActions: 00 00 00 00 01 00 00 00 01 00 00 00 01 00 00 00 14 00 00 00 01 00 00 00 60 EA 00 00
HKLM\SYSTEM\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-330533299-3180603657-300593454-1001\\Device\HarddiskVolume2\ProgramData\byvtbrpmgfwu791\@WanaDecryptor@.exe:
HKLM\SYSTEM\CurrentControlSet\Services\byvtbrpmgfwu791\Type: 0x00000010
HKLM\SYSTEM\CurrentControlSet\Services\byvtbrpmgfwu791\Start: 0x00000002
HKLM\SYSTEM\CurrentControlSet\Services\byvtbrpmgfwu791\ErrorControl: 0x00000001
HKLM\SYSTEM\CurrentControlSet\Services\byvtbrpmgfwu791\ImagePath: "cmd.exe /c "C:\ProgramData\byvtbrpmgfwu791\tasksche.exe""
HKLM\SYSTEM\CurrentControlSet\Services\byvtbrpmgfwu791\DisplayName: "byvtbrpmgfwu791"
```

Figure 32 regshot- registry values added

A persisting decryptor will continuously be popping-up with instructions how to recover encrypted files.

Figure 33 persistent decryptor

Background image of the machine is changed.

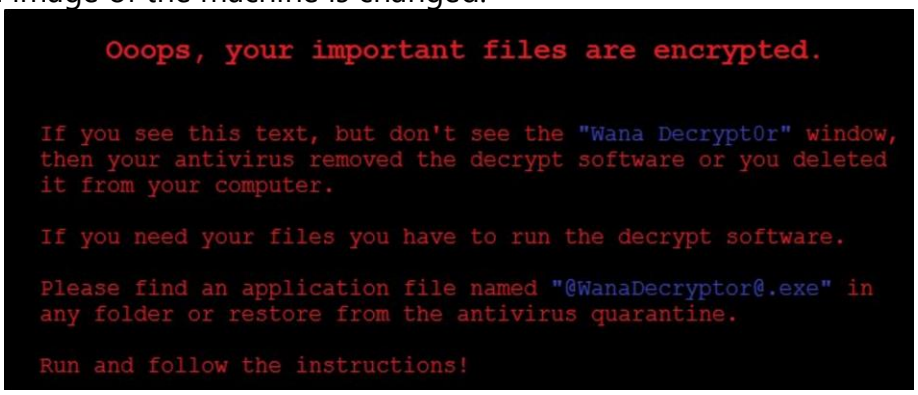


Figure 34 background changed

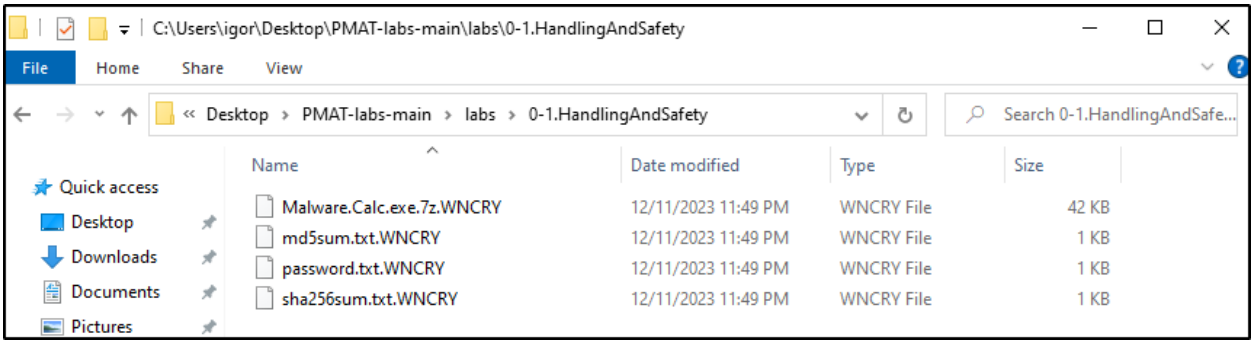


Figure 35 encrypted files sample.

Advanced Static Analysis

Advanced static analysis examines the software's code without executing it, using disassembly, and decompiling tools to translate binary into assembly or high-level code. Below is the main function of the malware.

```
[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
```

```
[0x004081a7]
 0x004081a7      call    esi
 0x004081a9      push    0
 0x004081ab      call    esi
 0x004081ad      call    fcn.00408090 ; fcn.00408090
 0x004081b2      pop     edi
 0x004081b3      xor     eax, eax
 0x004081b5      pop     esi
 0x004081b6      add     esp, 0x50
 0x004081b9      ret     0x10
```

```
[0x004081bc]
 0x004081bc      call    esi
 0x004081be      push    edi
 0x004081bf      call    esi
 0x004081c1      pop     edi
 0x004081c2      xor     eax, eax
 0x004081c4      pop     esi
 0x004081c5      add     esp, 0x50
 0x004081c8      ret     0x10
```

Figure 36 cutter- graph of the main function

1. **Stack Variable Declarations:**
    - The function declares several local variables (**var_64h**, **var_50h**, **var_17h**, **var_13h**, **var_fh**, **var_bh**, **var_7h**, **var_3h**, **var_1h**) at different stack offsets. These offsets indicate that the function uses a fair amount of stack space, likely for local data storage.
2. **Stack Frame Allocation:**
    - **sub esp, 0x50** adjusts the stack pointer to allocate 0x50 (80) bytes for the function's local variables and stack frame.
3. **Register Preparation and String Handling:**
    - Registers **esi** and **edi** are used for string operations, as seen in the **mov**, **rep movsd**, and **movsb** instructions. This indicates the use of string copying or manipulation.
    - The string "http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com" is moved into the local stack space. This is the killswitch URL of the malware.
4. **Zeroing Local Variables:**
    - Multiple **mov** instructions set local variables to zero.
5. **Windows API Calls:**
    - The **InternetOpenA** function is called, which is used to initialize an application's use of WinINet functions. The parameters pushed onto the stack before this call are setting up options for this function.
    - **InternetOpenUrlA** is called with the previously stored URL. This indicates an attempt to open a URL.
    - The result of **InternetOpenUrlA** is checked (**test edi, edi** and **jne**), which is a common error handling pattern to see if the function succeeded.
6. **Clean-up and Exit:**
    - The **InternetCloseHandle** function is referenced, which suggests that there is an attempt to properly close the handles opened by **InternetOpenA** and **InternetOpenUrlA**
7. **test edi, edi Instruction:**
    - The **test** instruction performs a bitwise AND operation between the two operands (**edi** and **edi** in this case) and sets the flags in the FLAGS register based on the result, but does not store the result of the AND operation.
    - In this specific case, **test edi, edi** is effectively checking whether the value in the **edi** register is zero. This is because the AND operation of any value with itself will be zero if and only if the value itself is zero.

8. **`jne 0x4081bc` Instruction:**
    - **`jne`** stands for "Jump if Not Equal". It is a conditional jump that takes place if the Zero Flag (ZF) in the FLAGS register is not set.
    - The Zero Flag is set by the previous **`test`** instruction if the result of the bitwise AND is zero. In this context, it means if **`edi`** is zero, the Zero Flag will be set.
9. **Implication in the Context:**
    - In the preceding lines of code, **`edi`** is assigned the return value of the **`InternetOpenUrlA`** function. In the Windows API, many functions return a non-zero value (often a handle or a pointer) to indicate success, and a zero value to indicate failure.
    - Therefore, the **`test edi, edi`** followed by **`jne 0x4081bc`** is checking whether the **`InternetOpenUrlA`** call was successful. If **`edi`** is non-zero (indicating success), the Zero Flag will not be set, and the **`jne`** instruction will cause a jump to the address **`0x4081bc`**.
    - If **`edi`** is zero (indicating failure), the Zero Flag will be set, and the **`jne`** instruction will not jump, but continue to the next sequential instruction which would be the left branch with the red arrow path.

Right Branch (Green Arrow Path) 0x004081bc:

1. **Function Call**: There's a **`call esi`** indicating a dynamic or indirect function call.
2. **Push, Call, and Pop Sequence**: There's a sequence where **`edi`** is pushed onto the stack, a function is called, and then **`edi`** is popped back. This suggests **`edi`** may hold a value or address that is needed after the function call.
3. **Register Operations**: The **`xor eax, eax`** clears the **`EAX`** register. This is followed by a series of **`pop`** instructions to clean up the stack, likely reversing the pushes done before the function calls.
4. **Stack and Function Return**: The final instructions clean up the stack with **`add esp, 0x50`** and return with **`ret 0x10`**, similar to the left branch.
5. This essentially quits the program

Left Branch (Red Arrow Path):

1. Almost the same with the right branch, except there is the function call **'fcn.00408090'** which contains the rest of the encryption payload.

fcn.00408090:

1. **File Path Retrieval:**
   - `GetModuleFileNameA` is called to retrieve the filename of the executable module.
   - The function uses `data.0070f760` as a buffer to store the file name and specifies `0x104` (260) as the size, which is the maximum path length in Windows.
2. **Argument Check:**
   - The program checks if at least two command-line arguments are provided. If not, it executes a different function (`fcn.00407f20`) and then terminates early, if fewer than two arguments are provided (`cmp dword [eax], 2` and `jge 0x4080b9`). If yes, it continues to execute the rest of the payload as denoted in the right-arrow path.

Figure 37 graph of fcn.00408090

Right Branch (Green Arrow Path)

1. **Opening an Existing Service**: The code uses `OpenServiceA` to open an existing service named "mssecsvc2.0" with significant access rights.
2. **Service Control Dispatcher Setup**: The program calls `StartServiceCtrlDispatcherA` with a service table pointer (`lpServiceStartTable`). This

function is used in service applications to handle service control requests (like start, stop, pause, etc.) from the Service Control Manager (SCM).

3. **Custom Function Invocation**: After successfully opening the service, it calls a custom function (`fcn.00407fa0`).

Left Branch (Red Arrow Path) fcn.00407fa0:

`fcn.00407fa0` is a function designed to change specific configuration settings of the Windows service.

```
[0x00407fa0]
fcn.00407fa0(SC_HANDLE hService, int32_t arg_8h);
; var int32_t var_1ch @ stack - 0x1c
; var int32_t var_18h @ stack - 0x18
; var LPVOID lpInfo @ stack - 0x14
; var int32_t var_10h @ stack - 0x10
; var int32_t var_ch @ stack - 0xc
; var int32_t var_8h @ stack - 0x8
; var int32_t var_4h @ stack - 0x4
; arg SC_HANDLE hService @ stack + 0x4
; arg int32_t arg_8h @ stack + 0x8
0x00407fa0      sub     esp, 0x1c
0x00407fa3      mov     eax, dword [arg_8h]
0x00407fa7      xor     edx, edx
0x00407fa9      mov     dword [esp], 1
0x00407fb1      mov     dword [lpInfo], 0
0x00407fb9      lea     ecx, [eax + eax*4]
0x00407fbc      lea     ecx, [ecx + ecx*4]
0x00407fbf      lea     ecx, [ecx + ecx*4]
0x00407fc2      shl     ecx, 3
0x00407fc5      cmp     eax, 0xffffffff
0x00407fc8      mov     dword [var_18h], ecx
0x00407fcc      setne   dl
0x00407fcf      mov     eax, data.0070f87c ; 0x70f87c
0x00407fd4      lea     ecx, [lpInfo]
0x00407fd8      mov     dword [var_8h], edx
0x00407fdc      mov     edx, dword [hService]
0x00407fe0      mov     dword [var_ch], eax
0x00407fe4      mov     dword [var_10h], eax
0x00407fe8      push    ecx         ; LPVOID lpInfo
0x00407fe9      lea     eax, [var_1ch]
0x00407fed      push    2           ; 2 ; DWORD dwInfoLevel
0x00407fef      push    edx         ; SC_HANDLE hService
0x00407ff0      mov     dword [var_4h], eax
0x00407ff4      call    dword [ChangeServiceConfig2A] ; 0x40a008 ; BOOL ChangeServiceConfig2A(SC_...
0x00407ffa      add     esp, 0x1c
0x00407ffd      ret
```

Figure 38 fcn.00407fa0

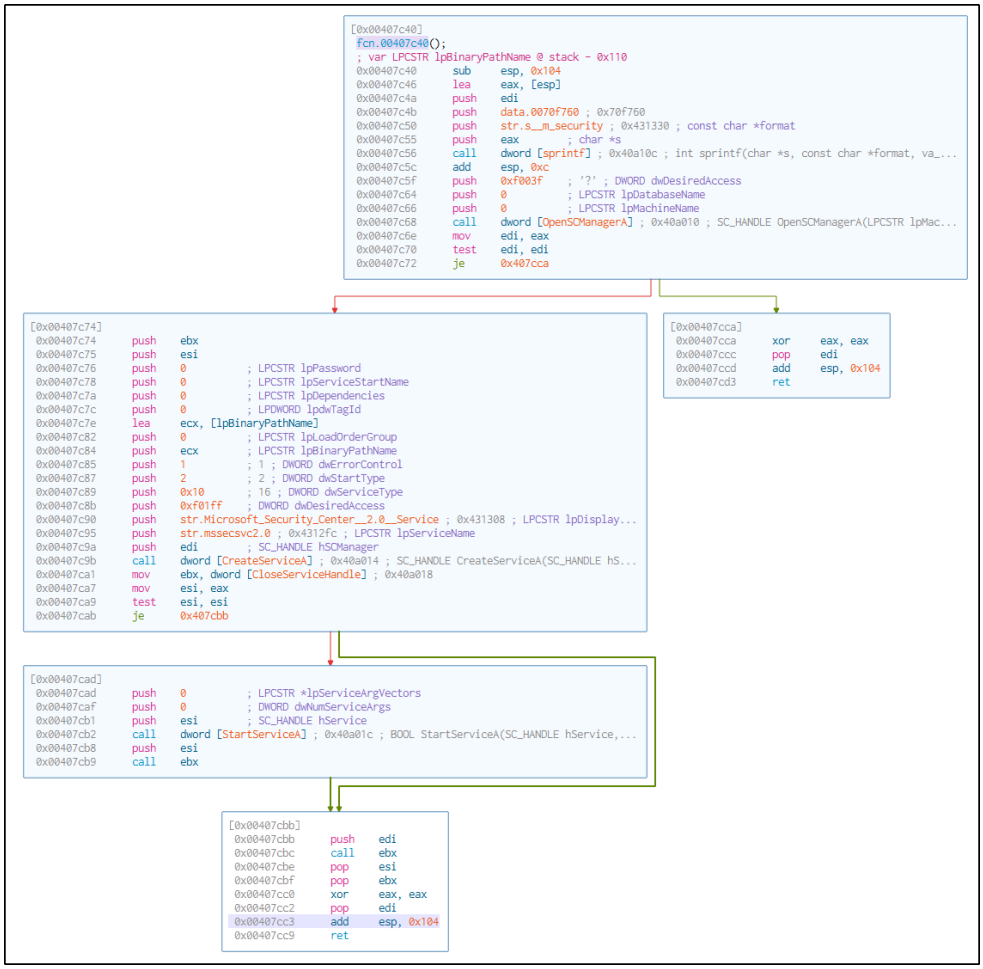
Left Branch (Red Arrow Path) fcn.00407f20:

The function consists of two calls to other functions: `fcn.00407c40` and `fcn.00407ce0`.

```
[0x00407f20]
fcn.00407f20();
0x00407f20      call    fcn.00407c40 ; fcn.00407c40
0x00407f25      call    fcn.00407ce0 ; fcn.00407ce0
0x00407f2a      xor     eax, eax
0x00407f2c      ret
```

fcn.00407c40:

- The function is involved in creating and starting a new Windows service.



fcn.00407ce0:

The function `fcn.00407ce0` appears to be involved in creating or manipulating the file named "tasksche.exe", and “qeriuwjhrf”.

```
[0x00407ce0]
 fcn.00407ce0();
 ; var int32_t var_30ch @ stack - 0x30c
 ; var int32_t var_304h @ stack - 0x304
 ; var int32_t var_2e0h @ stack - 0x2e0
 ; var int32_t var_2dch @ stack - 0x2dc
 ; var int32_t var_2d8h @ stack - 0x2d8
 ; var int32_t var_2d4h @ stack - 0x2d4
 ; var int32_t var_2d0h @ stack - 0x2d0
 ; var int32_t var_2cch @ stack - 0x2cc
 ; var int32_t var_2a4h @ stack - 0x2a4
 ; var int32_t var_2a0h @ stack - 0x2a0
 ; var LPVOID var_29ch @ stack - 0x29c
 ; var int32_t var_28ch @ stack - 0x28c
 ; var int32_t var_258h @ stack - 0x258
 ; var LPCSTR lpExistingFileName @ stack - 0x24c
 ; var LPCSTR lpNewFileName @ stack - 0x148
 0x00407ce0      sub     esp, 0x260
 0x00407ce6      push    ebx
 0x00407ce7      push    ebp
 0x00407ce8      push    esi
 0x00407ce9      push    edi
 0x00407cea      push    str.kernel32.dll ; 0x4313b4 ; LPCWSTR lpModuleName
 0x00407cef      call    dword [GetModuleHandleW] ; 0x40a064 ; HMODULE GetModuleHandleW(LPCWSTR lp...
 0x00407cf5      mov     esi, eax
 0x00407cf7      xor     ebx, ebx
 0x00407cf9      cmp     esi, ebx
 0x00407cfb      je      0x407f08
```

```
[0x00407d01]
 0x00407d01      mov     edi, dword [GetProcAddress] ; 0x40a060
 0x00407d07      push    str.CreateProcessA ; 0x4313a4 ; LPOVERLAPPED lpOverlapped
 0x00407d0c      push    esi         ; LPDWORD lpNumberOfBytesWritten
 0x00407d0d      call    edi
 0x00407d0f      push    str.CreateFileA ; 0x431398 ; DWORD nNumberOfBytesToWrite
 0x00407d14      push    esi         ; LPCVOID lpBuffer
 0x00407d15      mov     dword data.00431478, eax ; 0x431478
 0x00407d1a      call    edi
 0x00407d1c      push    str.WriteFile ; 0x43138c ; HANDLE hFile
 0x00407d21      push    esi
 0x00407d22      mov     dword data.00431458, eax ; 0x431458
 0x00407d27      call    edi
 0x00407d29      push    str.CloseHandle ; 0x431380 ; HANDLE hObject
 0x00407d2e      push    esi
 0x00407d2f      mov     dword data.00431460, eax ; 0x431460
 0x00407d34      call    edi
 0x00407d36      mov     ecx, dword data.00431478 ; 0x431478
 0x00407d3c      mov     dword data.0043144c, eax ; 0x43144c
 0x00407d41      cmp     ecx, ebx
 0x00407d43      je      0x407f08
```

```
[0x00407d49]
 0x00407d49      cmp     dword data.00431458, ebx ; 0x431458
 0x00407d4f      je      0x407f08
```

```
[0x00407d55]
 0x00407d55      cmp     dword data.00431460, ebx ; 0x431460
 0x00407d5b      je      0x407f08
```

```
[0x00407db9]
 0x00407db9      mov     ecx, 0x40  ; '@' ; 64
 0x00407dbe      xor     eax, eax
 0x00407dc0      lea     edi, [lpExistingFileName + 0x1]
 0x00407dc4      mov     byte [lpExistingFileName], bl
 0x00407dc8      rep     stosd dword es:[edi], eax
 0x00407dca      stosw   word es:[edi], ax
 0x00407dcc      stosb   byte es:[edi], al
 0x00407dcd      mov     ecx, 0x40  ; '@' ; 64
 0x00407dd2      xor     eax, eax
 0x00407dd4      lea     edi, [lpNewFileName + 0x1]
 0x00407ddb      mov     byte [lpNewFileName], bl
 0x00407de2      rep     stosd dword es:[edi], eax
 0x00407de4      mov     esi, dword [sprintf] ; 0x40a10c
 0x00407dea      push    str.tasksche.exe ; 0x43136c
 0x00407def      stosw   word es:[edi], ax
 0x00407df1      stosb   byte es:[edi], al
 0x00407df2      push    str.WINDOWS ; 0x431364
 0x00407df7      lea     eax, [lpExistingFileName]
 0x00407dfb      push    str.C:__s__s ; 0x431358
 0x00407e00      push    eax
 0x00407e01      call    esi
 0x00407e03      add     esp, 0x10
 0x00407e06      lea     ecx, [lpNewFileName]
 0x00407e0d      push    str.WINDOWS ; 0x431364
 0x00407e12      push    str.C:__s_qeriuwjhrf ; 0x431344
 0x00407e17      push    ecx
 0x00407e18      call    esi
 0x00407e1a      add     esp, 0xc
 0x00407e1d      lea     edx, [lpNewFileName]
 0x00407e24      lea     eax, [lpExistingFileName]
 0x00407e28      push    1          ; 1 ; DWORD dwFlags
 0x00407e2a      push    edx        ; LPCSTR lpNewFileName
 0x00407e2b      push    eax        ; LPCSTR lpExistingFileName
 0x00407e2c      call    dword [MoveFileExA] ; 0x40a04c ; BOOL MoveFileExA(LPCSTR lpExistingFileNa...
 0x00407e32      push    ebx
 0x00407e33      push    4          ; 4
 0x00407e35      push    2          ; 2
 0x00407e37      push    ebx
 0x00407e38      push    ebx
 0x00407e39      lea     ecx, [var_258h]
 0x00407e3d      push    0x40000000
 0x00407e42      push    ecx
 0x00407e43      call    dword [data.00431458] ; 0x431458
 0x00407e49      mov     esi, eax
 0x00407e4b      cmp     esi, 0xffffffff
 0x00407e4e      je      0x407f08
```

Advanced Dynamic Analysis

Advanced dynamic analysis refers to the process of evaluating and investigating a software program by executing it in a controlled environment and observing its behavior in real-time.

With INETSIM ON

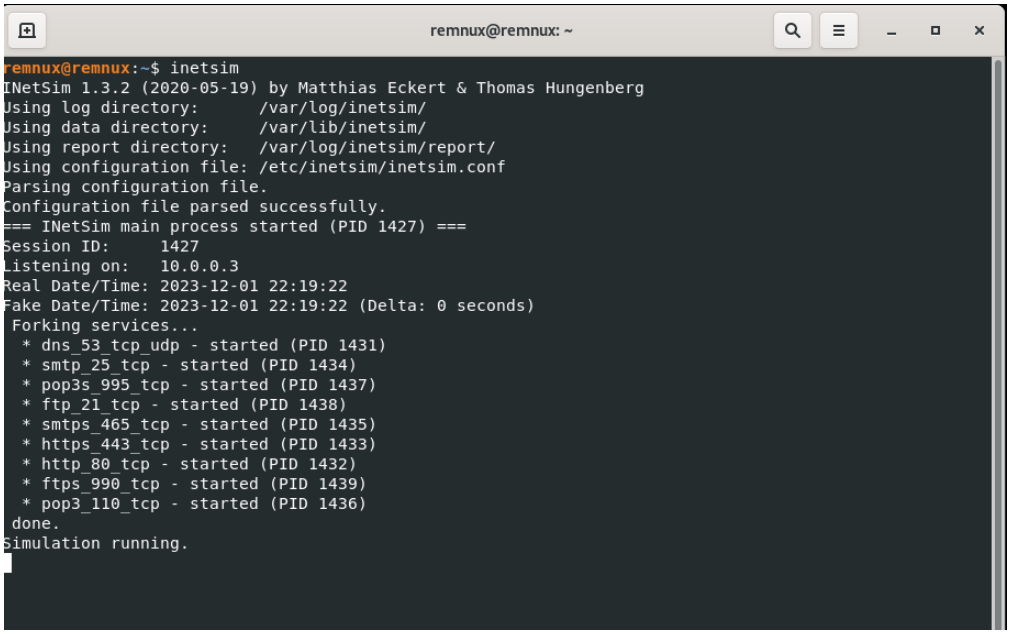


Figure 39 inetsim on

Open the malware in x32dbg. From the entry point, run it once to be directed to the main function.

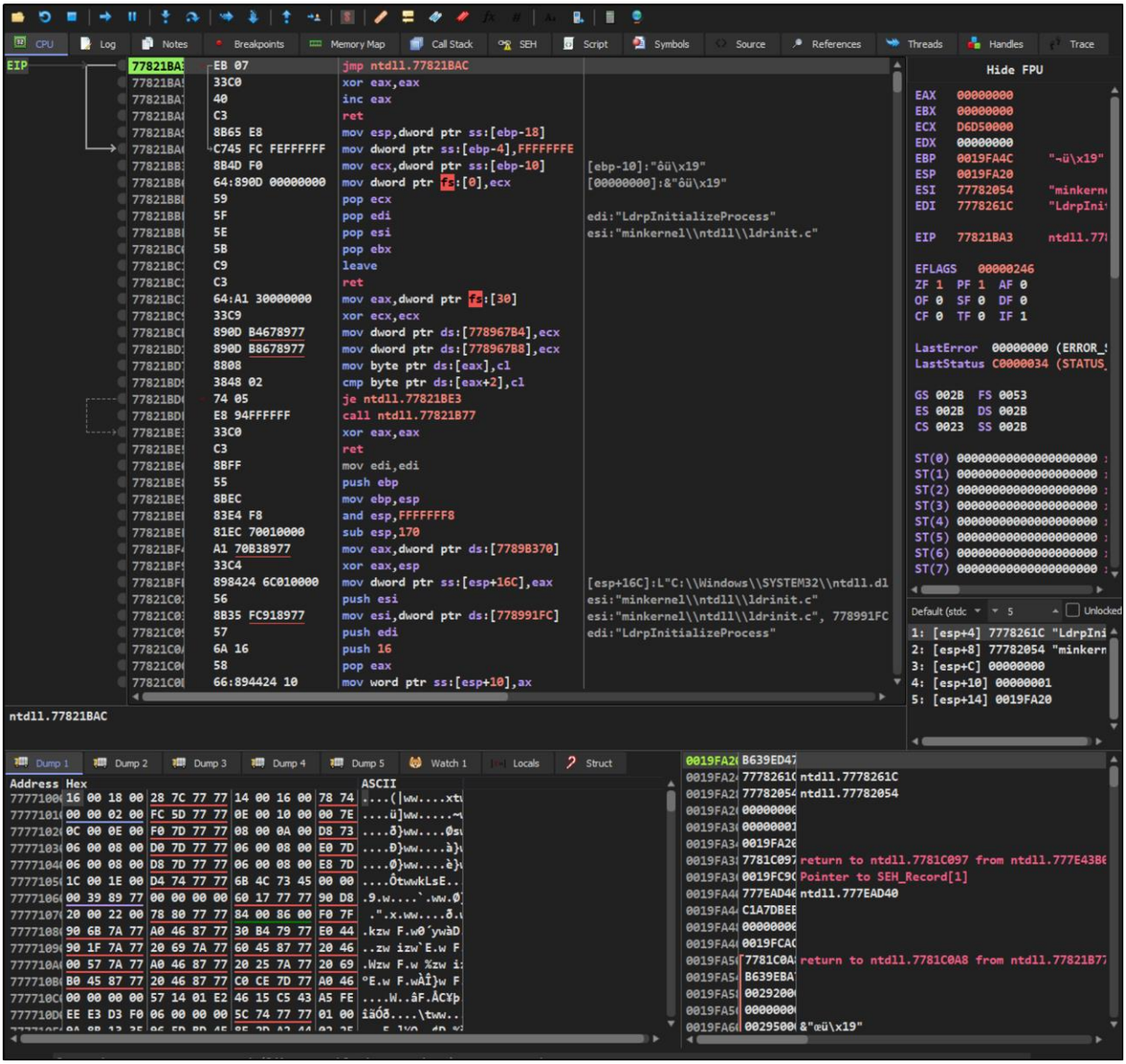


Figure 40 entry point

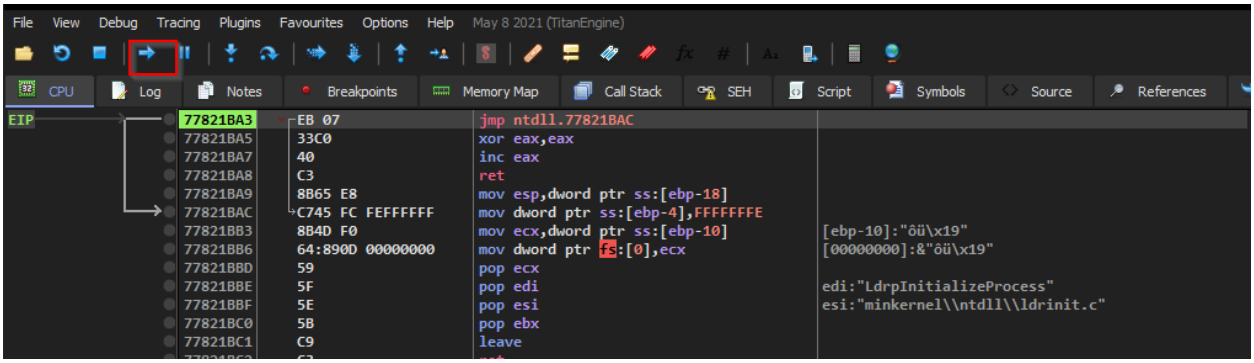


Figure 41 main function

Search for the killswitch URL which will point us to the location when the API will be called which will eventually determine whether to execute the rest of the payload or not.

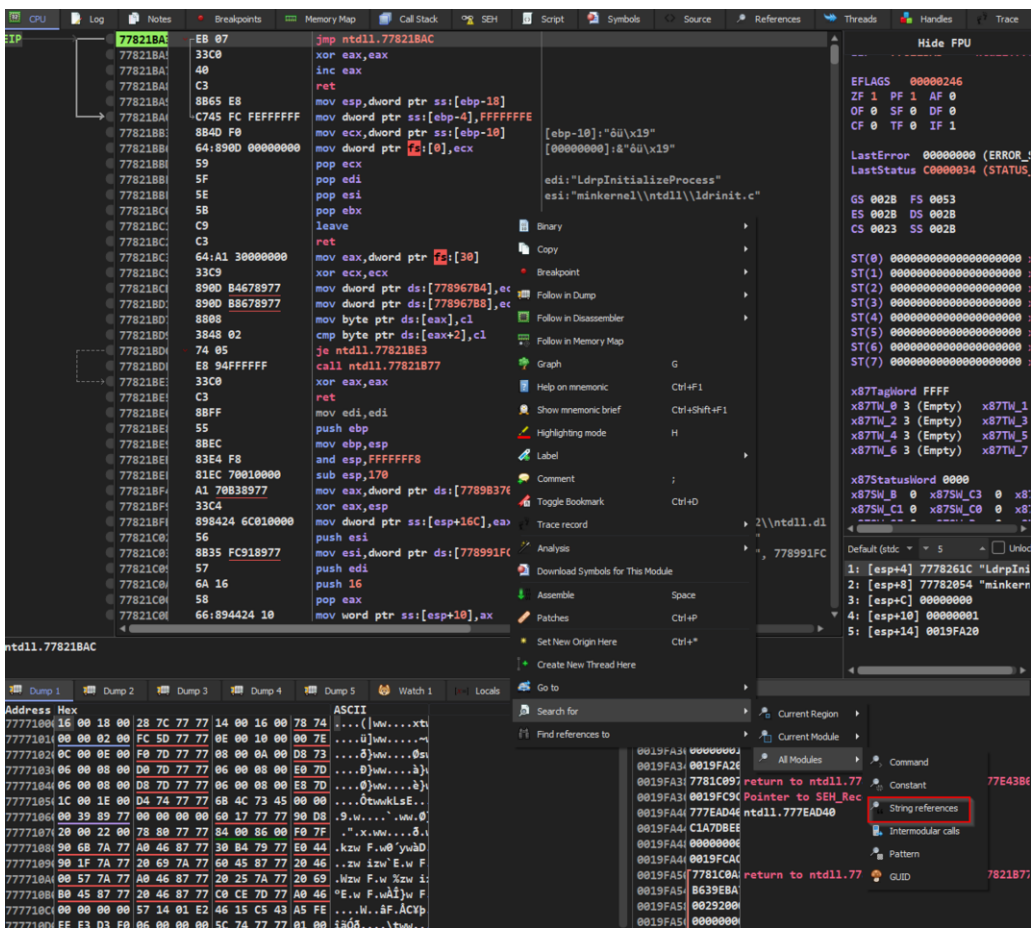


Figure 42 searching for the killswitch URL

Once the address is located, set a break point.

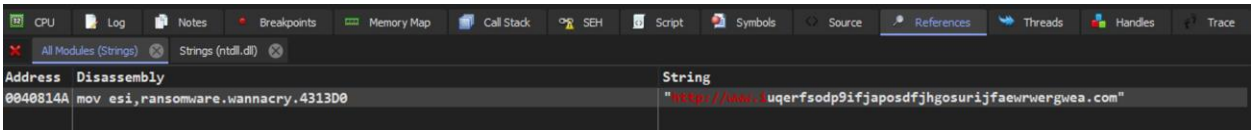


Figure 43 breakpoint created

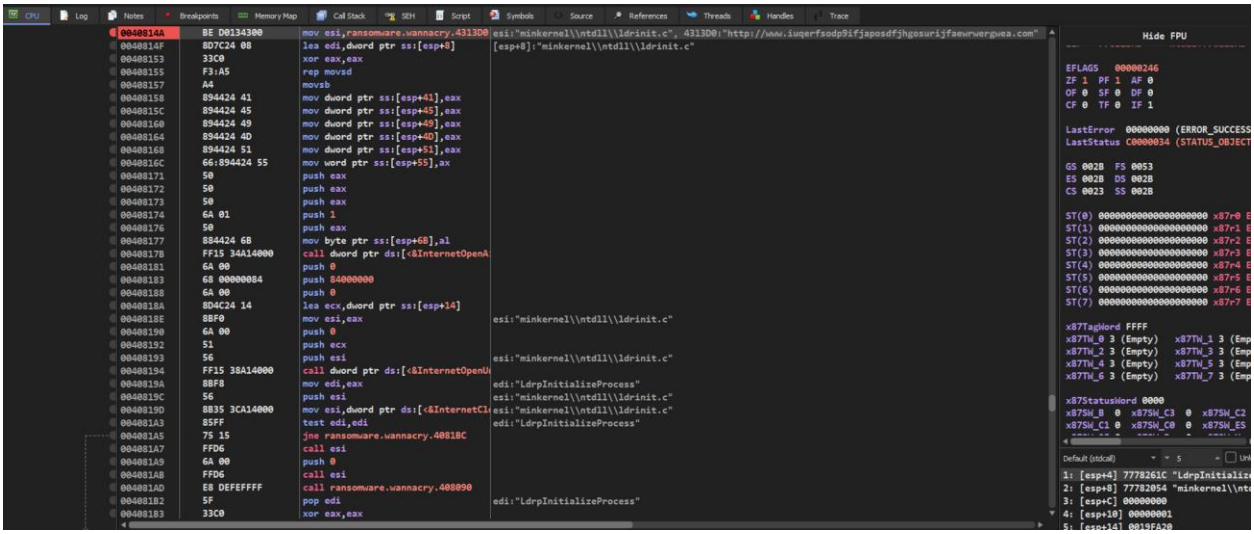


Figure 44 stepping over just before the test instruction

Step over until the zero flag is being evaluated, which is the test instruction.

EDI has non-zero values which indicate a successful connection to the killswitch URL thus the ZF flag has 1 value.

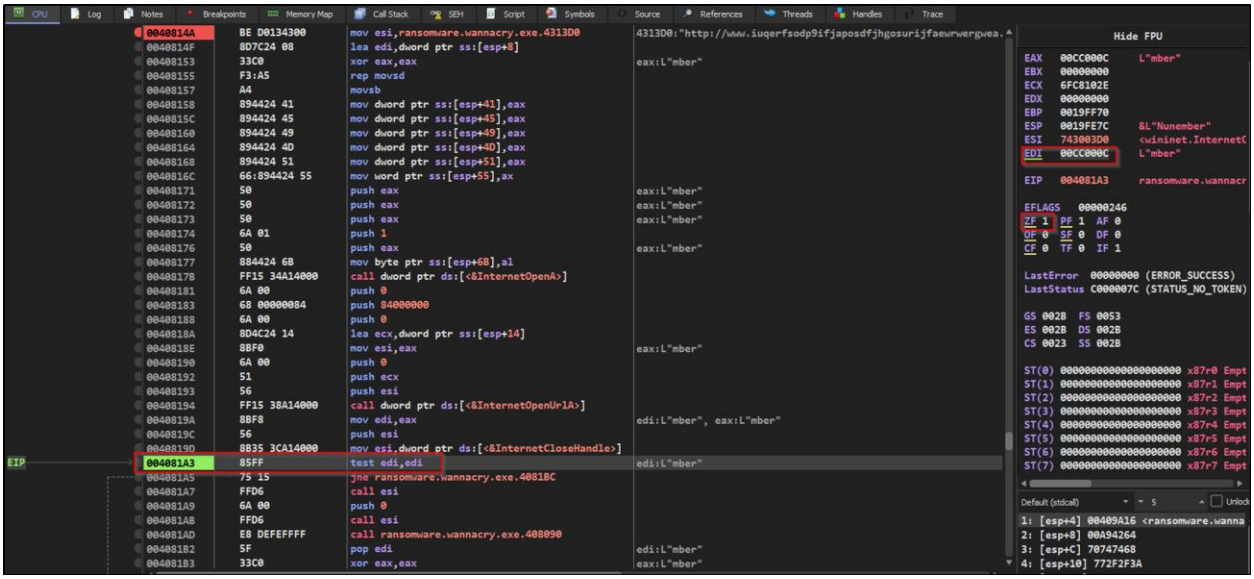


Figure 45 edi tested against itself

Step over to the next instruction which will test EDI against itself. The ZF flag is 0 which means that the EDI is not set. This would mean then that the next set of instructions to be performed will be the right-branch which will end the program.

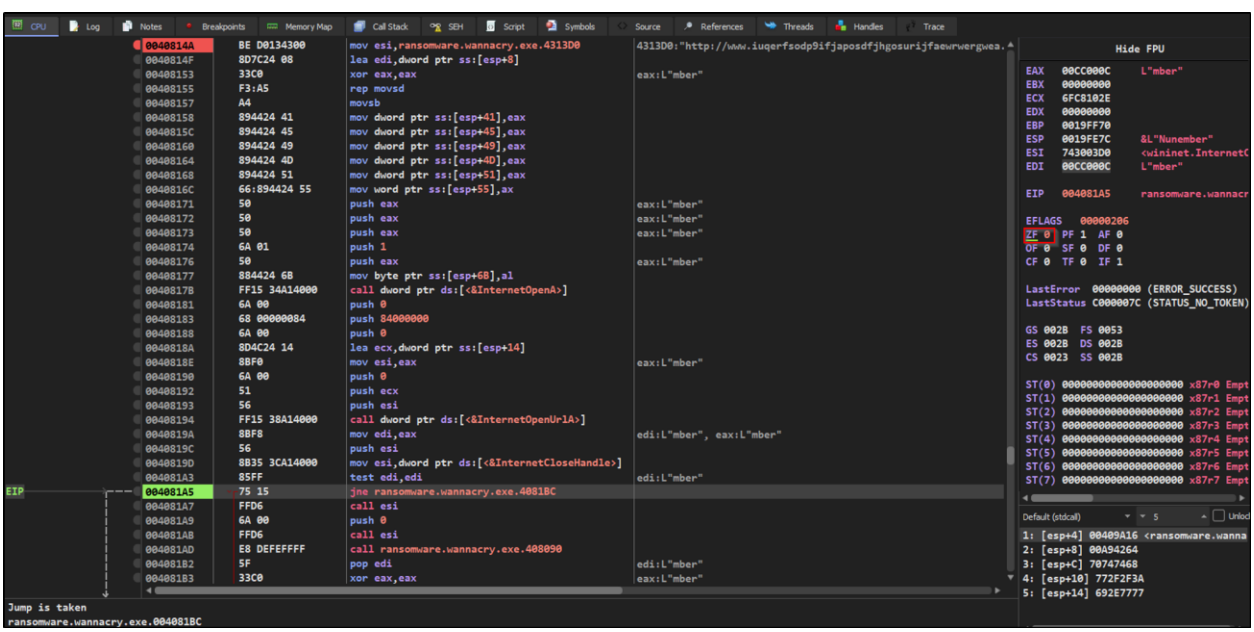


Figure 46 jne instruction

Step over without changing the ZF flag.

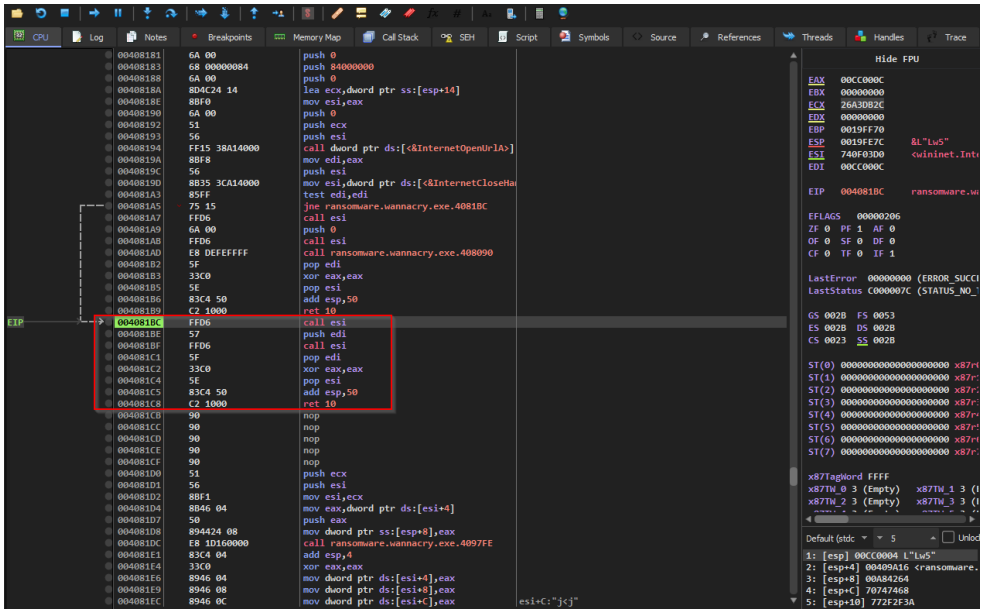


Figure 47 ZF flag not modified

Step over until the return instruction is executed and run the program. The next set of instructions as seen in the static analysis is the termination of the program.

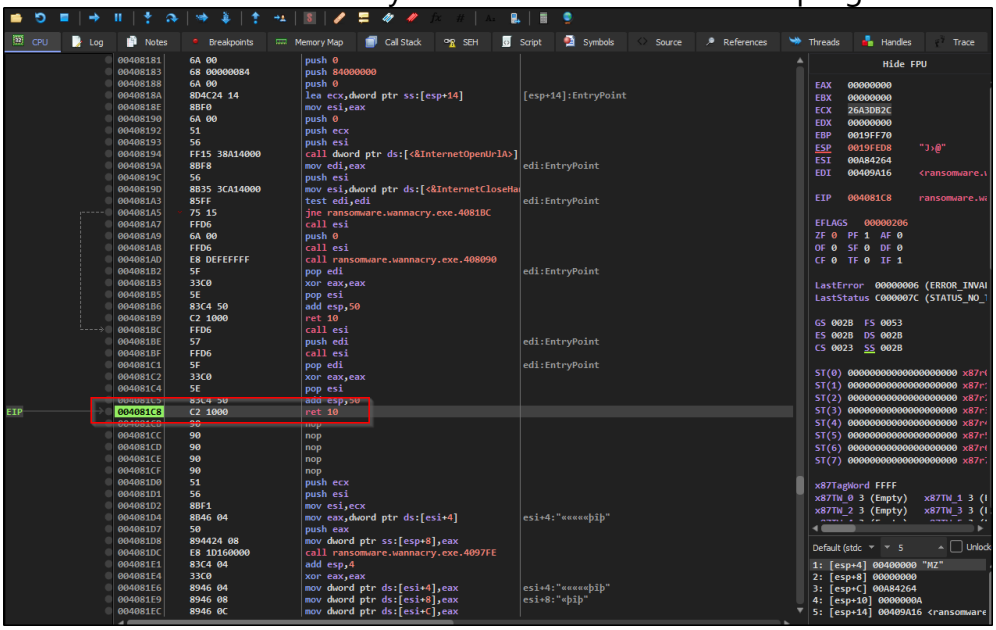


Figure 48 return instruction

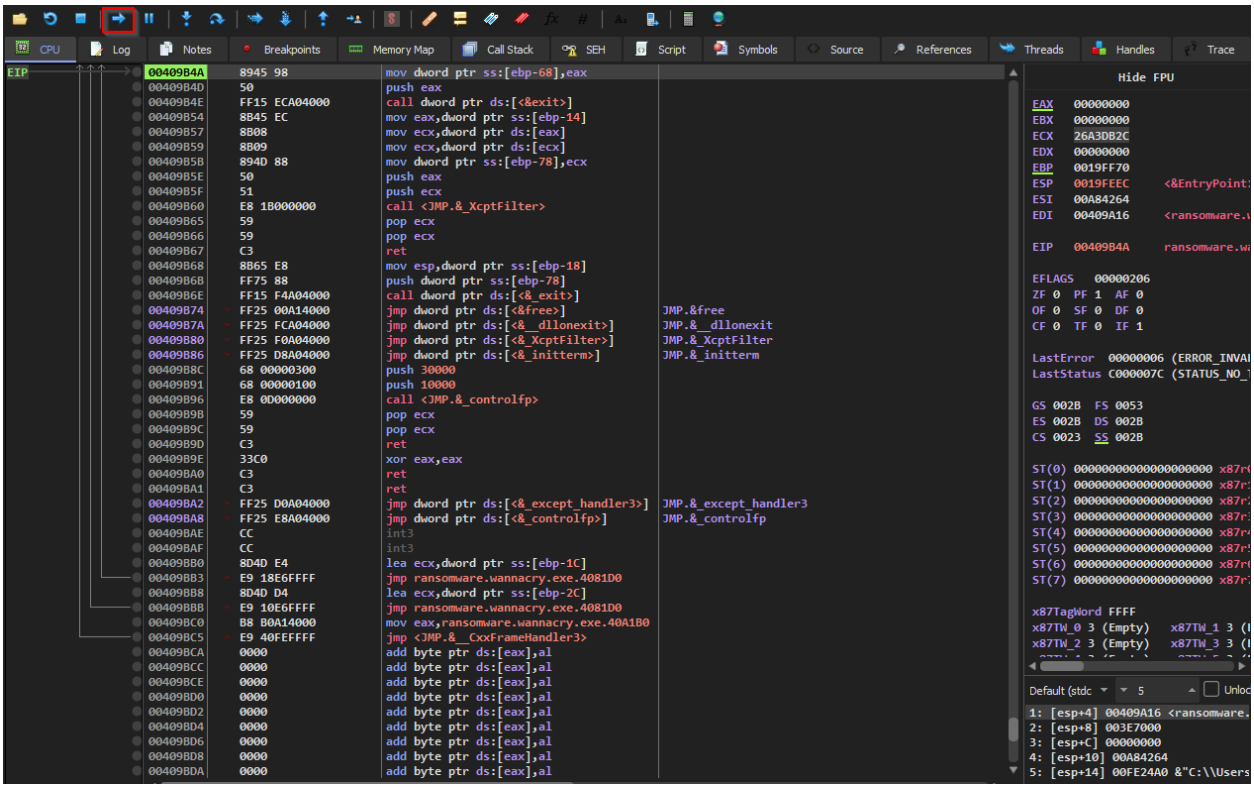


Figure 49 run once which ends the program

Modifying the ZF flag

By modifying the ZF flag, the program will execute the rest of the payload despite INETSIM being on.

Restart the malware and run until the breakpoint where the killswitch URL is moved into the stack. Step over before the conditional jump (jne) is taken.

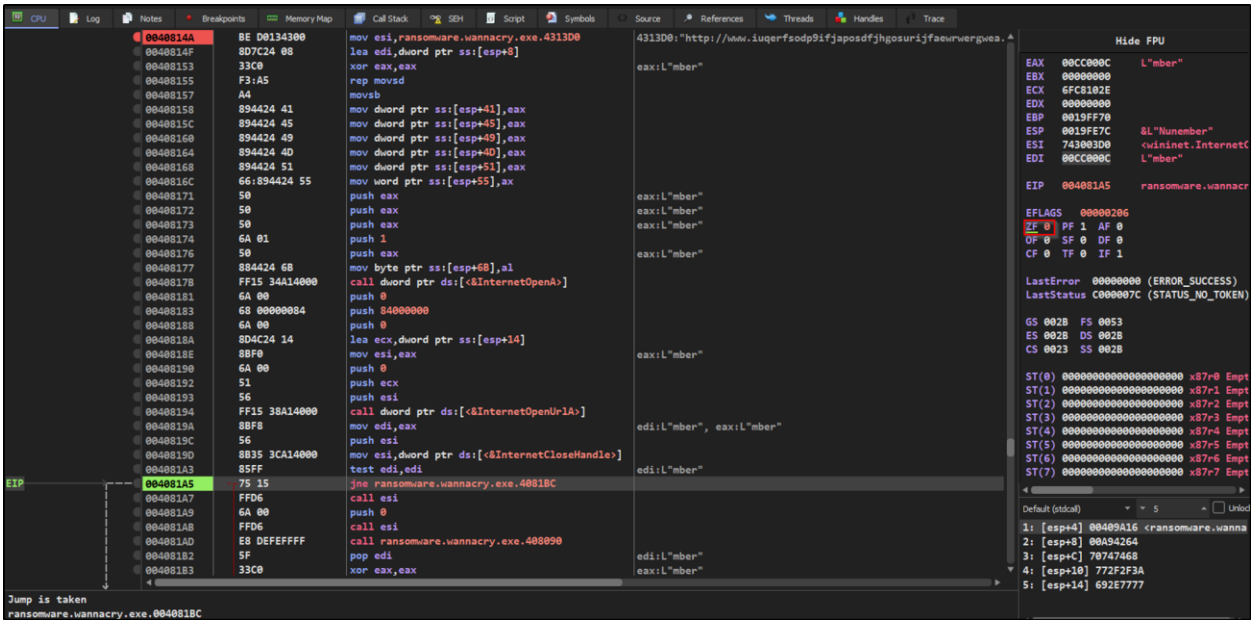


Figure 50 jne instruction where ZF flag is not yet modified

Modify the ZF value to 1 so that the jump will not be taken, and the rest of the payload will be executed.

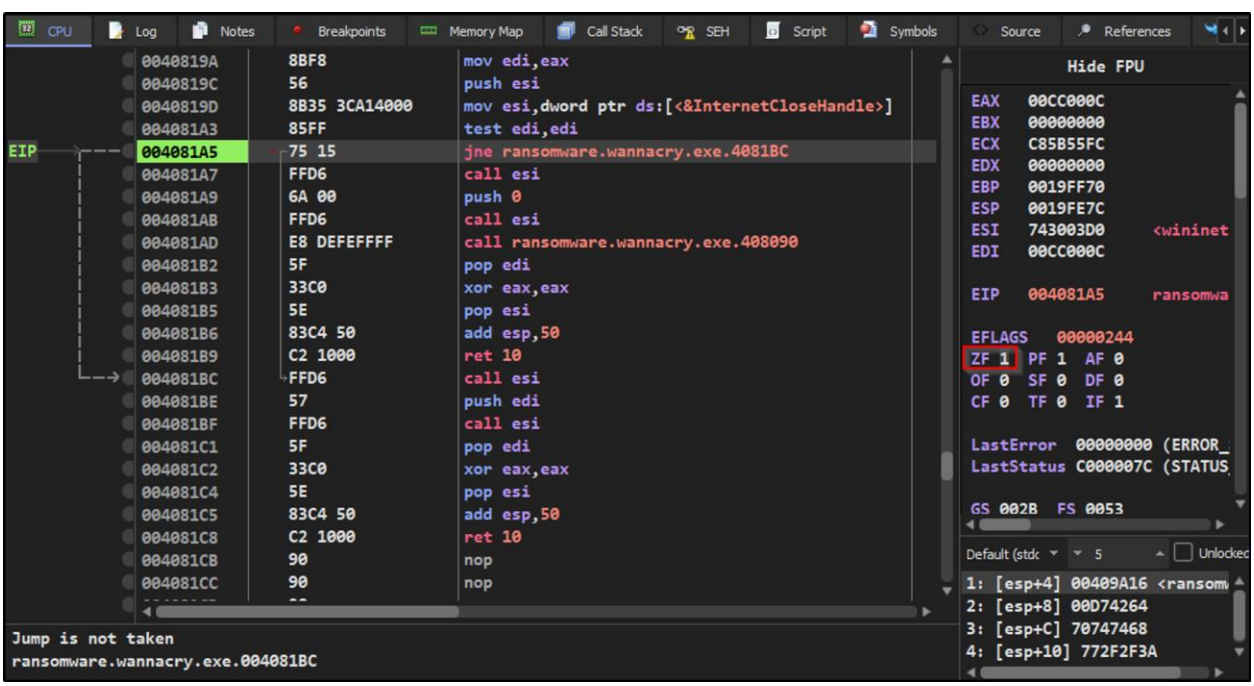


Figure 51 jne instruction where ZF flag was modified

Step over a few of the instructions until the function call which contains the remaining payload of the malware.

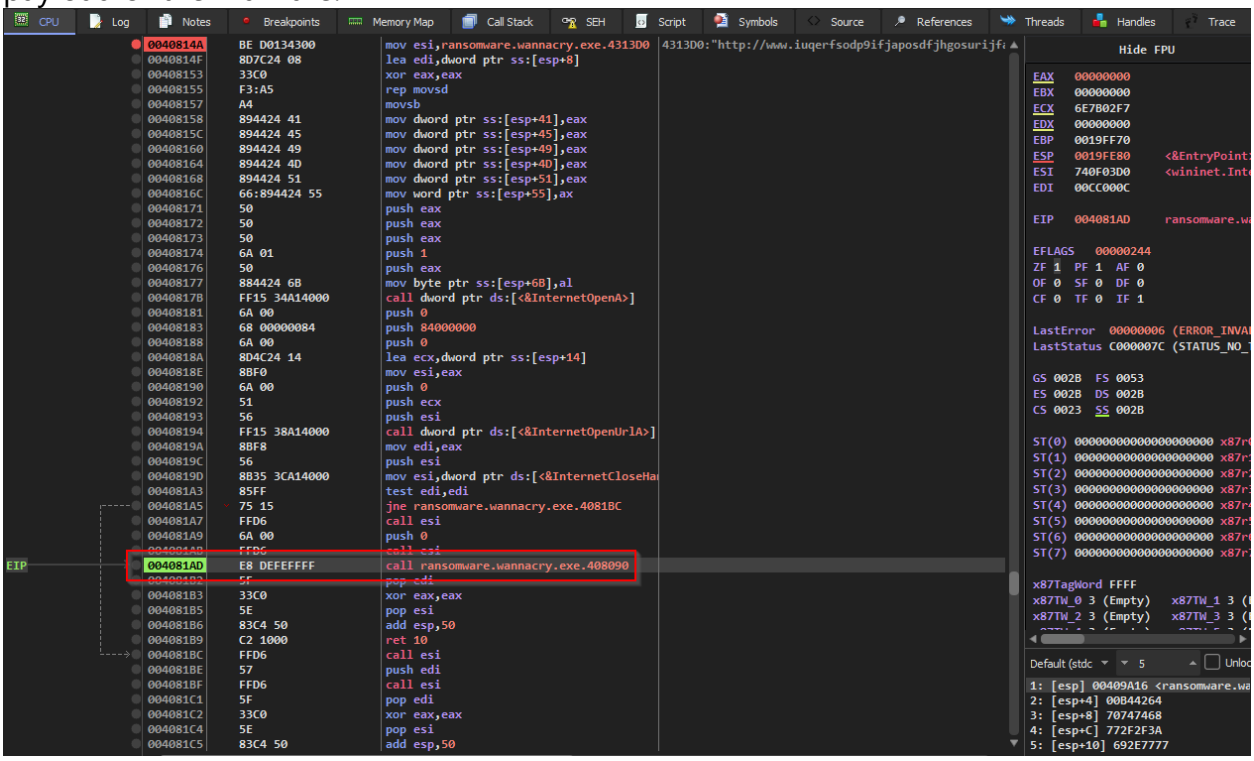


Figure 52 executes the call function which proceeds with the rest of the payload

Once called, it will execute the rest of the payload.

Indicators of Compromise

The full list of IOCs can be found in the Appendices.

Network Indicators

Checks the killswitch URL.

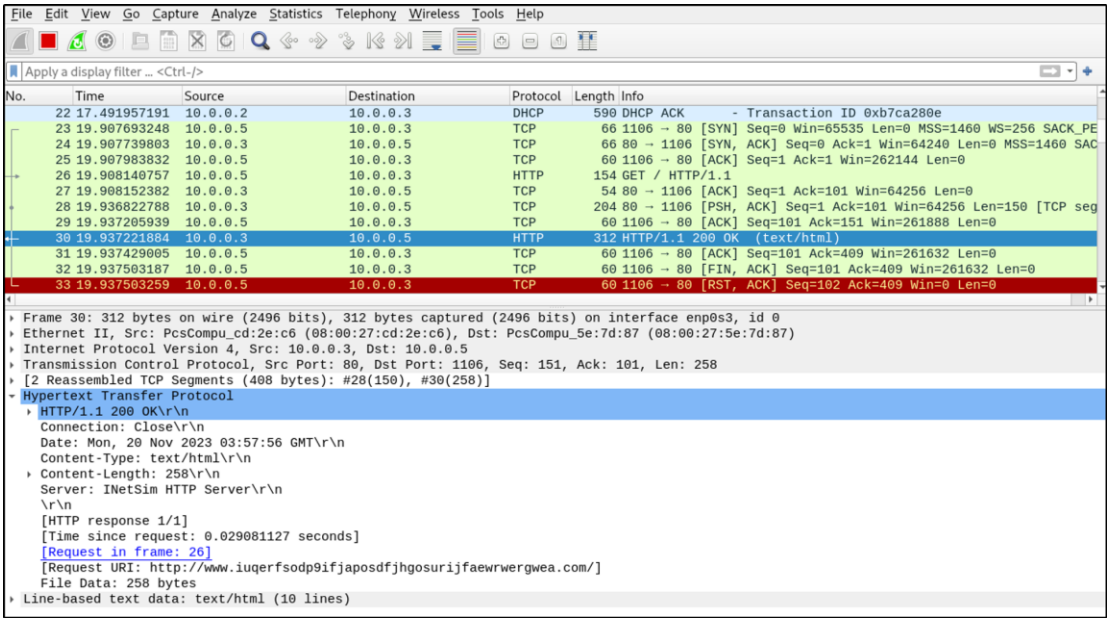


Figure 53WireShark Packet Capture of the Killswitch URL

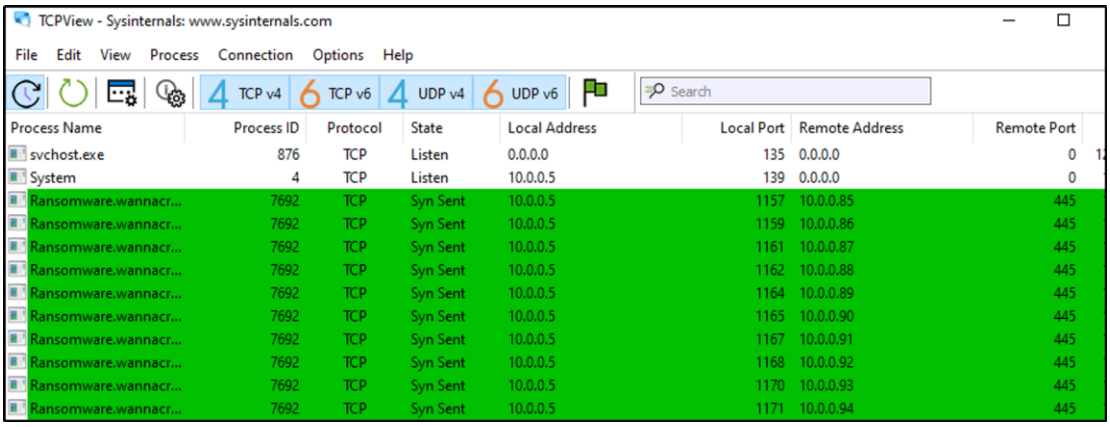


Figure 54 SMB requests in an attempt to infect other hosts in the network.

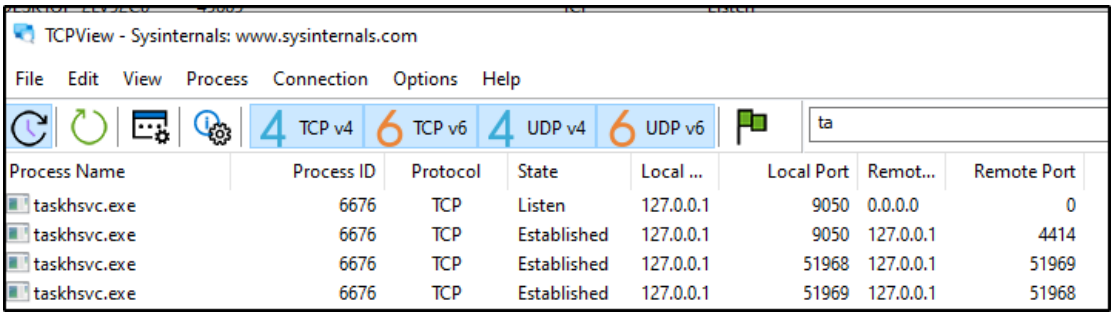


Figure 55 Opened local ports.

Host-based Indicators

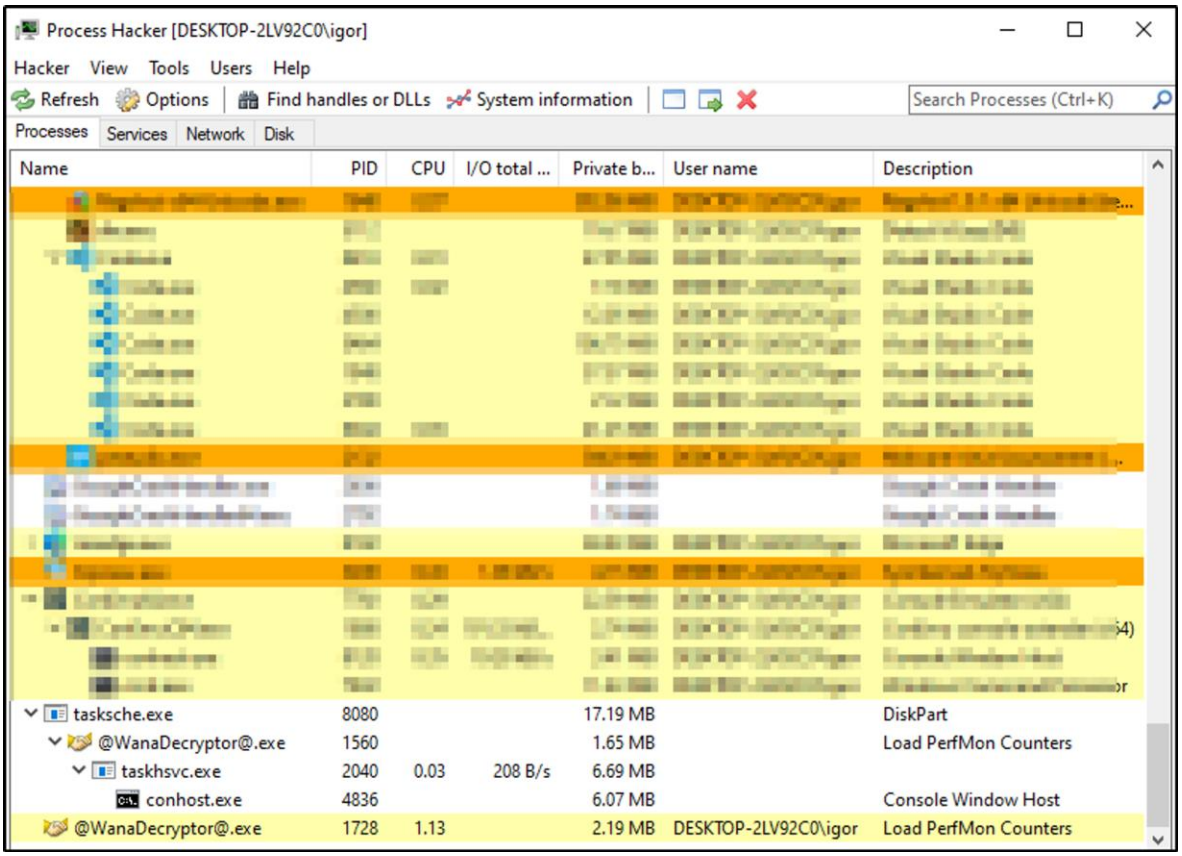


Figure 56 child processes of tasksche.exe

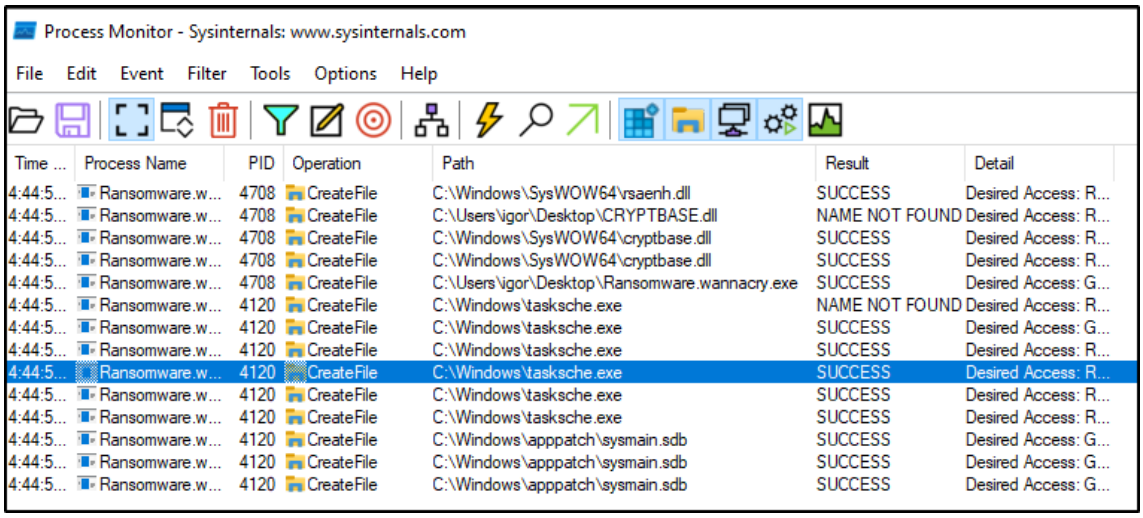


Figure 57 Created file tasksche.exe

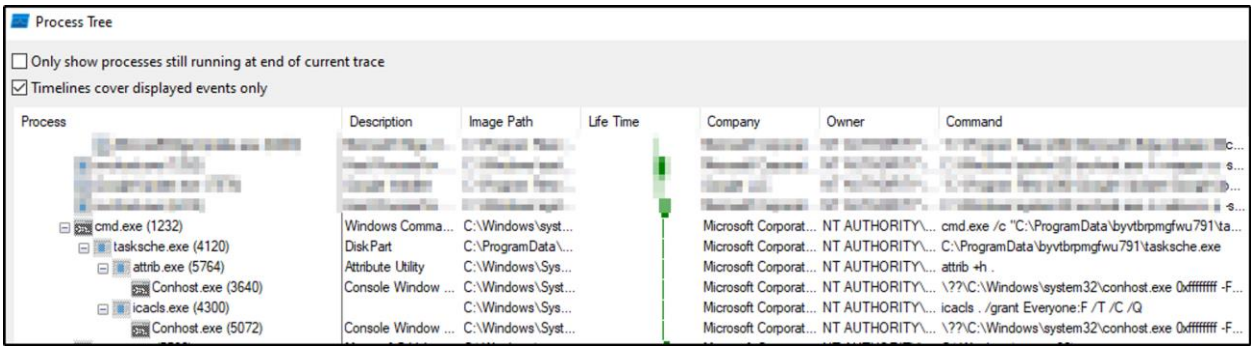


Figure 58 process tree of tasksche.exe

Figure 59 Registry key values added

```
HKLM\SOFTWARE\WOW6432Node\WanaCrypt0r
HKLM\SYSTEM\ControlSet001\Services\byvtbrpmgfwu791
HKLM\SYSTEM\ControlSet001\Services\mssecsvc2.0
HKLM\SYSTEM\CurrentControlSet\Services\byvtbrpmgfwu791
HKLM\SYSTEM\CurrentControlSet\Services\mssecsvc2.0
```
Figure 60 Regisrtry keys added

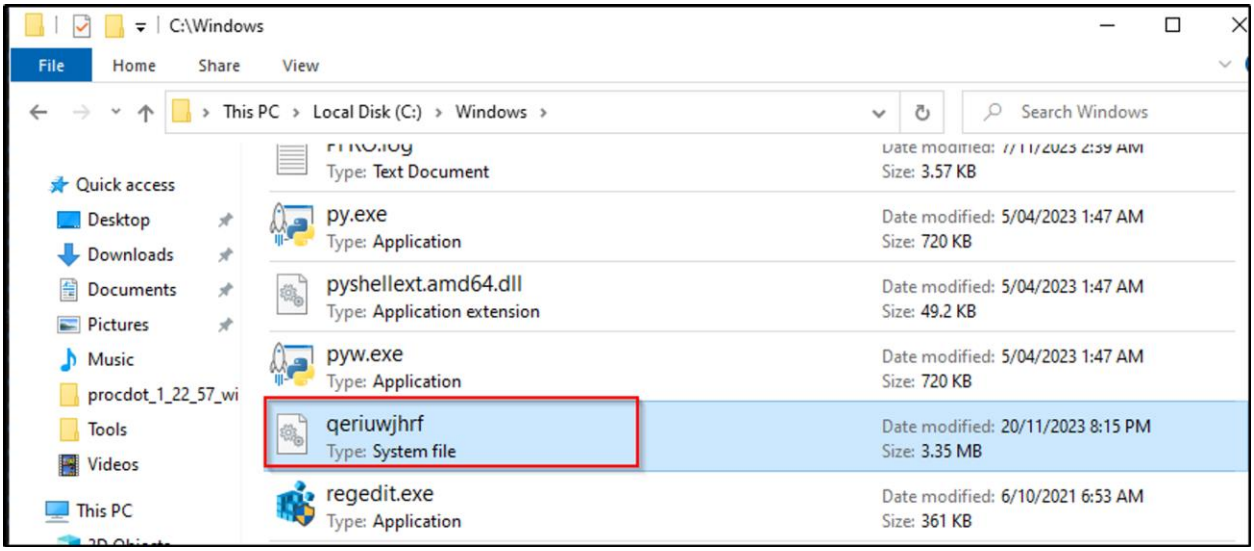


Figure 61 The random string file created and saved in "C:\Windows" directory

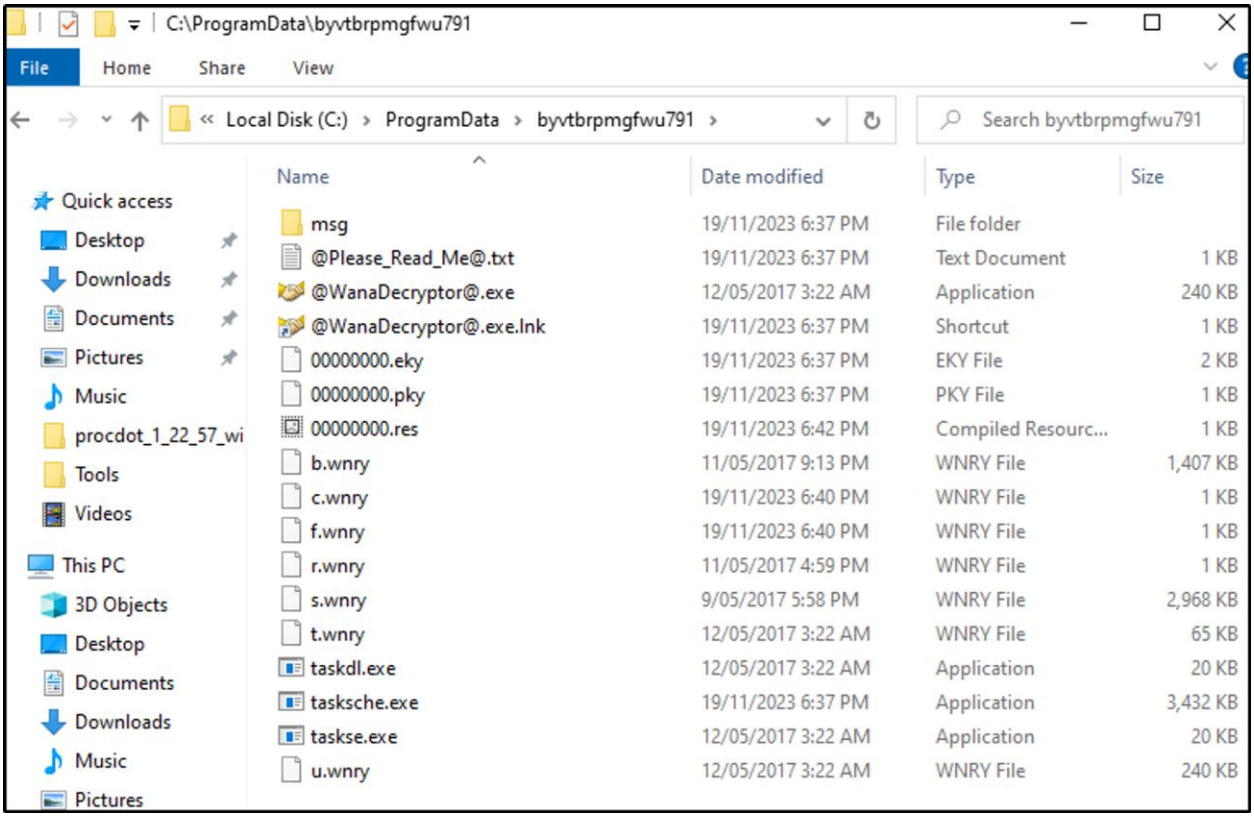


Figure 62Malware directory for persistence

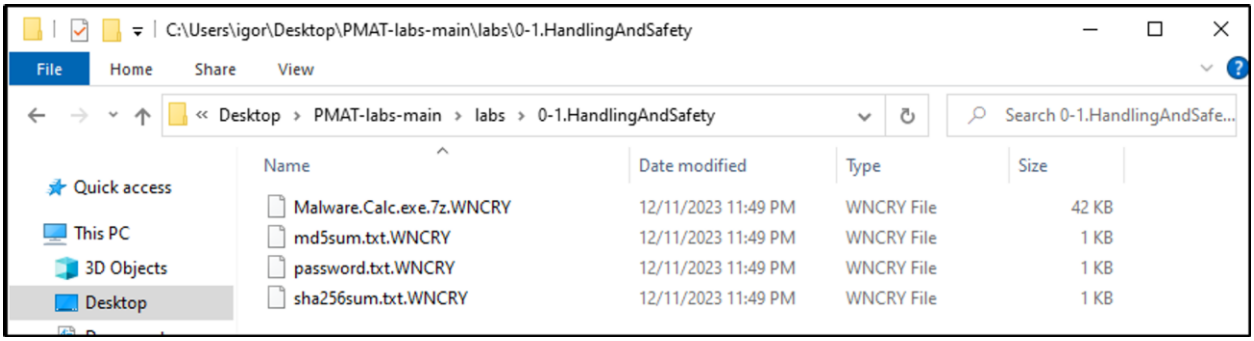


Figure 63 Sample of encrypted files

Rules & Signatures

Strings

- %s -m security
- cmd.exe /c "%s"
- tasksche.exe
- lhdfrgui.exe
- http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com
- C:\%s\qeriuwjhrf
- tasksche.exe
- WNcry@2ol7

Signatures

CAPA

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] Socket Communication::TCP Client [C0001.008]
CRYPTOGRAPHY	Generate Pseudo-random Sequence::Use API [C0021.003]
DATA	Compression Library [C0060]
DISCOVERY	Analysis Tool Discovery::Process detection [B0013.001] Code Discovery::Inspect Section Memory Permissions [B0046.002] File and Directory Discovery [E1083]

EXECUTION	Install Additional Program [B0023]
FILE SYSTEM	Move File [C0063] Read File [C0051]
PROCESS	Create Thread [C0038] Terminate Process [C0018] Terminate Thread [C0039]

Capability	**Namespace**
reference analysis tools strings	anti-analysis
check for time delay via QueryPerformanceCounter	anti-analysis/anti-debugging/debugger-detection
contain obfuscated stackstrings	anti-analysis/obfuscation/string/stackstring
receive data (5 matches)	communication
send data (5 matches)	communication
connect to URL	communication/http/client
get socket status	communication/socket
initialize Winsock library	communication/socket
set socket configuration	communication/socket
create UDP socket (4 matches)	communication/socket/udp/send
act as TCP client	communication/tcp/client
generate random numbers via WinAPI	data-manipulation/prng
extract resource via kernel32 functions	executable/resource
contain an embedded PE file	executable/subfile/pe
get file size	host-interaction/file-system/meta
move file	host-interaction/file-system/move
read file on Windows	

get number of processors	host-interaction/file-system/read
terminate process	host-interaction/hardware/cpu
run as service	host-interaction/process/terminate
create service	host-interaction/service
modify service	host-interaction/service/create
start service	host-interaction/service/modify
create thread (4 matches)	host-interaction/service/start
terminate thread	host-interaction/thread/create
link function at runtime on Windows	host-interaction/thread/terminate
linked against ZLIB	linking/runtime-linking
inspect section memory permissions	linking/static/zlib
persist via Windows service	load-code/pe
	persistence/service

Appendices

Yara Rules

```
rule Ransomware_wannacry_exe {

   meta:
      last_updated = "2023-12-04"
      author = "igor_sec"
      description = "yara rule for detecting WannaCry Malware"
      hash1 = "24d004a104d4d54034dbcffc2a4b19a11f39008a575aa614ea04703480b1022c"

   strings:
      $x1 = "cmd.exe /c \"%s\"" fullword ascii
      $s2 = "msvcrtd.dll" fullword ascii
      $s3 = "launcher.dll" fullword ascii
      $s4 = "taskdl.exe" fullword ascii
      $s5 = "mssecsvc.exe" fullword ascii
      $s6 = "taskse.exe" fullword ascii
      $s7 = "tasksche.exe" fullword ascii
      $s8 = "http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com" fullword
ascii
      $s9 = "C:\\%s\\qeriuwjhrf" fullword ascii
      $10 = “WNcry@2ol7” fullword ascii

   condition:
      uint16(0) == 0x5a4d and filesize < 11000KB and
      1 of ($x*) and 4 of them
}
```

Decompiled Code Snippets

Decompiler (main)

```
/* jsdec pseudo code output */
/* C:\Users\igor\Desktop\Ransomware.wannacry.exe @ 0x408188 */
#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;
}
```

Figure 64 main function in Cutter

Decompiler (fcn.00408090)

```
/* jsdec pseudo code output */
/* C:\Users\igor\Desktop\Ransomware.wannacry.exe.malz @ 0x408090 */
#include <stdint.h>

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;
}
```

Figure 65 fcn.00408090 mssecsv2.0 service being opened in Cutter

Decompiler (fcn.00407f20)

```
/* jsdec pseudo code output */
/* C:\Users\igor\Desktop\Ransomware.wannacry.exe.malz @ 0x407f20 */
#include <stdint.h>

uint32_t fcn_00407f20 (void) {
    fcn_00407c40 ();
    eax = fcn_00407ce0 ();
    eax = 0;
    return eax;
}
```

Figure 66 calls for two functions with different purpose

Decompiler (fcn.00407c40)

```
/* jsdec pseudo code output */
/* C:\Users\igor\Desktop\Ransomware.wannacry.exe.malz @ 0x407c40 */
#include <stdint.h>

uint32_t fcn_00407c40 (void) {
    LPCSTR lpBinaryPathName;
    eax = esp;
    sprintf (eax, "%s -m security", data.0070f760, edi);
    eax = OpenSCManagerA (0, 0, 0xf003f);
    edi = eax;
    if (edi != 0) {
        ecx = &lpBinaryPathName;
        eax = CreateServiceA (edi, "mssecsvc2.0", "Microsoft Security Center (2.0) Service", 0xf01ff, 0x10, 2, 1, ecx, 0, 0, 0, 0, 0, esi, ebx);
        ebx = imp.CloseServiceHandle;
        esi = eax;
        if (esi != 0) {
            StartServiceA (esi, 0, 0);
            void (*ebx)(uint32_t) (esi);
        }
        eax = void (*ebx)(uint32_t) (edi);
        eax = 0;
        return eax;
    }
    eax = 0;
    return eax;
}
```

Figure 67 fcn.00407c40 is involved in creating and starting the Windows service mssecsvc2.0

Decompiler (fcn.00407ce0)

```
/* jsdec pseudo code output */
/* C:\Users\igor\Desktop\Ransomware.wannacry.exe.malz @ 0x407ce0 */
#include <stdint.h>

uint32_t fcn_00407ce0 (void) {
    int32_t var_30ch;
    int32_t var_304h;
    int32_t var_2e0h;
    int32_t var_2dch;
    int32_t var_2d8h;
    int32_t var_2d4h;
    int32_t var_2d0h;
    int32_t var_2cch;
    int32_t var_2a4h;
    int32_t var_2a0h;
    LPVOID var_29ch;
    int32_t var_28ch;
    int32_t var_258h;
    LPCSTR lpExistingFileName;
    LPCSTR lpNewFileName;
    eax = GetModuleHandleW ("kernel32.dll", edi, esi, ebp);
    esi = eax;
    ebx = 0;
    if (esi == ebx) {
        goto label_0;
    }
    edi = imp.GetProcAddress;
    eax = void (*edi)(uint32_t, char*) (esi, "CreateProcessA");
    *(data.00431478) = eax;
    eax = void (*edi)(uint32_t, char*) (esi, "CreateFileA");
    *(data.00431458) = eax;
    eax = void (*edi)(uint32_t, char*) (esi, "WriteFile");
    *(data.00431460) = eax;
    eax = void (*edi)(uint32_t, char*) (esi, "CloseHandle");
    ecx = *(data.00431478);
    *(data.0043144c) = eax;
    if (ecx == ebx) {
        goto label_0;
    }
    if (*(data.00431458) == ebx) {
        goto label_0;
    }
    if (*(data.00431460) == ebx) {
        goto label_0;
    }
    if (eax == ebx) {
        goto label_0;
    }
    eax = FindResourceA (ebx, 0x727, data.0043137c);
    esi = eax;
    if (esi == ebx) {
        goto label_0;
    }
    eax = LoadResource (ebx, esi);
    if (eax == ebx) {
        goto label_0;
    }
    eax = LockResource (eax);
    var_29ch = eax;
    if (eax == ebx) {
        goto label_0;
    }
    eax = SizeofResource (ebx, esi);
    if (ebp == ebx) {
        goto label_0;
    }
    ecx = 0x40;
    eax = 0;
    edi = lpExistingFileName + 0x1;
    memset (edi, eax, ecx);
    *(es:edi) = ax;
    es:edi += 2;
    *(es:edi) = al;
    es:edi++;
    ecx = 0x40;
    eax = 0;
    edi = lpNewFileName + 0x1;
    memset (edi, eax, ecx);
    esi = imp.sprintf;
    *(es:edi) = ax;
    es:edi += 2;
    *(es:edi) = al;
    es:edi++;
    eax = &lpExistingFileName;
```

```
Decompiler (fcn.00407ce0)
eax = &lpExistingFileName;
void (*esi)(uint32_t, uint32_t, uint32_t, char*) (eax, bl, bl, "tasksche.exe");
ecx = &lpNewFileName;
void (*esi)(uint32_t, char*, char*) (ecx, "C:\%s\qeriuwjhrf", "WINDOWS");
edx = &lpNewFileName;
eax = &lpExistingFileName;
MoveFileExA (eax, edx, 1);
ecx = &var_258h;
eax = uint32_t (*data.00431458)(uint32_t, uint32_t, uint32_t, uint32_t, uint32_t, uint32_t, uint32_t) (ecx, 0x40000000, ebx, eb
esi = eax;
if (esi == -1) {
    goto label_0;
}
eax = var_2cch;
edx = &var_2cch;
uint32_t (*data.00431460)(uint32_t, uint32_t, uint32_t, uint32_t) (esi, eax, ebp, edx);
eax = uint32_t (*data.0043144c)(uint32_t) (esi);
ecx = 0;
eax = 0;
var_2dch = ecx;
edi = &var_2cch;
var_2d8h = ecx;
edx = &var_28ch;
var_2d4h = ecx;
ecx = 0x10;
memset (edi, eax, ecx);
edi = data.00431340;
ecx |= 0xffffffff;
__asm ("repne scasb al, byte es:[edi]");
ecx = ~ecx;
edi -= ecx;
var_2e0h = ebx;
esi = edi;
edi = edx;
ecx |= 0xffffffff;
__asm ("repne scasb al, byte es:[edi]");
ecx = ebp;
edi--;
ecx >>= 2;
do {
    *(es:edi) = *(esi);
    ecx--;
    esi += 4;
    es:edi += 4;
} while (ecx != 0);
ecx = ebp;
eax = &var_2e0h;
ecx &= 3;
*(es:edi) = *(esi);
ecx--;
esi++;
es:edi++;
ecx = &var_2d0h;
edx = &var_28ch;
eax = uint32_t (*data.00431478)(uint32_t, uint32_t, uint32_t, uint32_t, uint32_t, uint32_t, uint32_t, uint32_t, uint32_t, uint3
if (eax != 0) {
    eax = var_304h;
    uint32_t (*data.0043144c)(uint32_t) (eax);
    ecx = var_30ch;
    eax = uint32_t (*data.0043144c)(uint32_t) (ecx);
}
```

Figure 68 The function fcn.00407ce0 creating or manipulating the file named "tasksche.exe", and "qeriuwjhrf",