Practical Malware Analysis & Triage
Malware Analysis Report

Wannacry - Ransomware

Ian 2023 | cosmin-stan | v1.0

Table of Contents

Executive Summary

SHA256 hash	24D004A104D4D54034DBCFFC2A4B19A11F39008A575AA614EA04703480B1022C

WannaCry ransomware is a crypto-ransomware worm, compiled in C++, that targets hosts with Windows OS. WannaCry was first identified in May 2017 and spread panic across corporate networks worldwide as it quickly infected more than 200,000 computers in 150 countries.
WannaCry takes advantage of the Eternal Blue vulnerability to spread throughout the network, it is still one of the most commonly attempted exploits against SMB, accounting for over 91.88% of the attacks on port 445 (the SMB port).

#YARA signature rules are attached in Appendix 1

High-Level Technical Summary

WannaCry consists of a main payload that attempts to reach to a URL. If the connection to the URL is successfully, it unpacks its next stage files into *C:\ProgramData\[RANDOM STRING].*

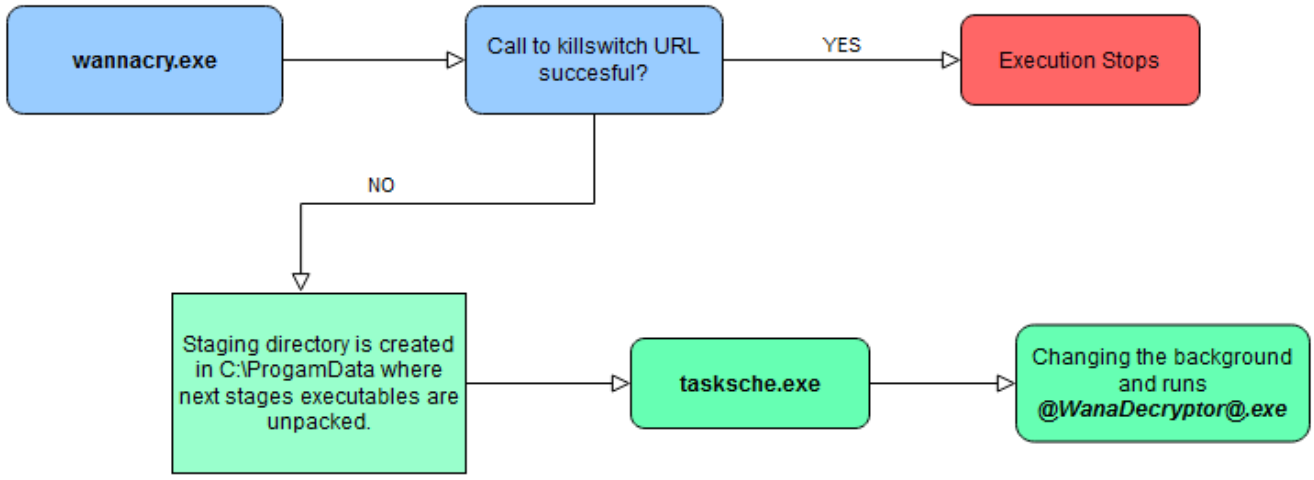


Fig 1: Execution of WannaCry - Flowchart

Malware Composition

WannaCry consists of the following components:

File Name	SHA256 Hash
ransomware.wannacry.exe	24D004A104D4D54034DBCFFC2A4B19A11F39008A575AA614EA04703480B1022C
tasksche.exe	ED01EBFBC9EB5BBEA545AF4D01BF5F1071661840480439C6E5BABE8E080E41AA

ransomware.wannacry.exe.exe

The initial executable succeeds if the call domain is not reachable.

tasksche.exe

The second stage executable is unpacked from the initial WannaCry executable and conducts most of the malicious operations on the host, encrypting the files.

Additional files:

:

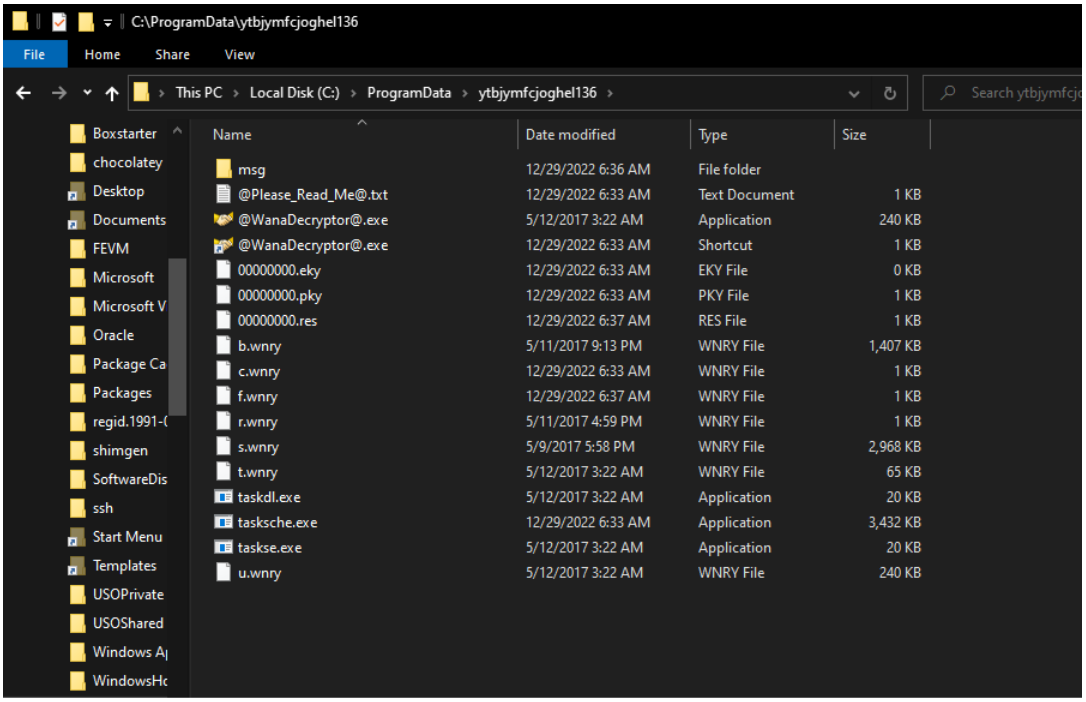


Fig 2: Staging directory for the WannaCry

Basic Static Analysis

{Screenshots and description about basic static artifacts and methods}

We used FLOSS to extract the PE strings and dump them into a txt file:

```
floss.exe -n 8 Ransomware.wannacry.exe.malz > wannacry_strings.txt
```

```
C:\Users\fl-vm\Desktop
λ FLOSS.exe -n 8 Ransomware.wannacry.exe.malz > floss.wannacry.txt
```

Suspicious strings:

Encryption imports:

```
Microsoft Enhanced RSA and AES Cryptographic Provider
CryptGenKey
CryptDecrypt
CryptEncrypt
CryptDestroyKey
CryptImportKey
CryptAcquireContextA
```

Suspicious file path and possible second-stage executable

```
Microsoft Base Cryptographic Provider v1.0
%d.%d.%d.%d
mssecsvc2.0
Microsoft Security Center (2.0) Service
%s -m security
C:\%s\qeriuwjhrf
C:\%s\%s
tasksche.exe
```

Call back domain

```
CreateProcessA
http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com
!This program cannot be run in DOS mode.
```

Suspicious message box

```
MessageBoxW
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
```

Suspicious string and dll

```
advapi32.dll
WANACRY!
```

Advapi32. dll is a part of the advanced API services library. It provides access to advanced functionality that comes in addition to the kernel. It is responsible for things like the Windows registry, restarting and shutting down the system, starting/stopping and creating Windows services, and managing user accounts!

Suspicious Windows utility tool

```
icacls . /grant Everyone:F /T /C /Q
attrib +h .
WNcry@2ol7
```

icacls is a Windows command-line utility that IT admins can use to change access control lists on files and folders.

I also used PEStudio to highlight some IOCs:

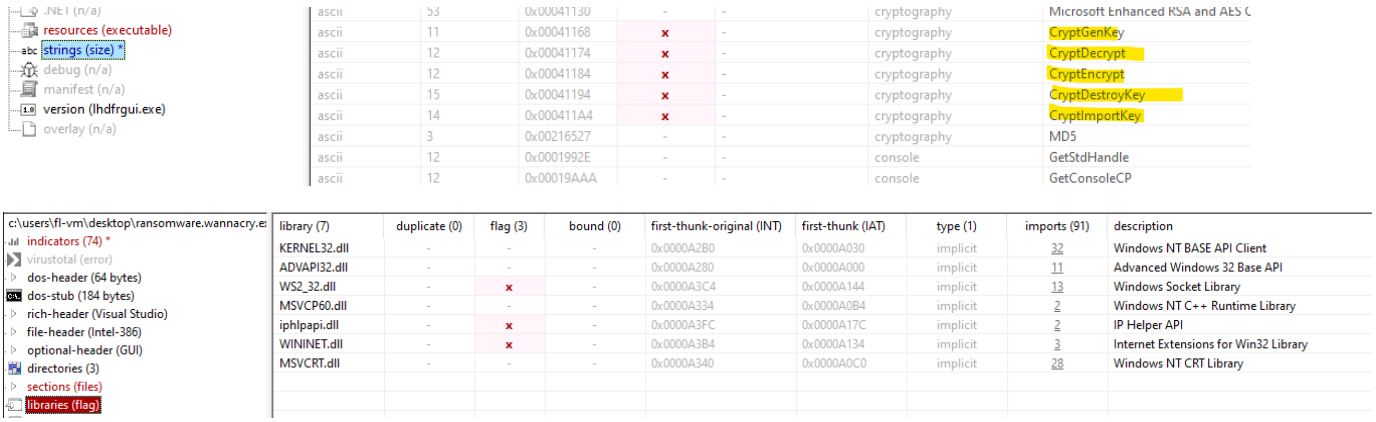


Fig 3: PEView

Basic Dynamic Analysis

{Screenshots and description about basic dynamic artifacts and methods}

I used Procmon to identify host indicators:

Procmon Filters:

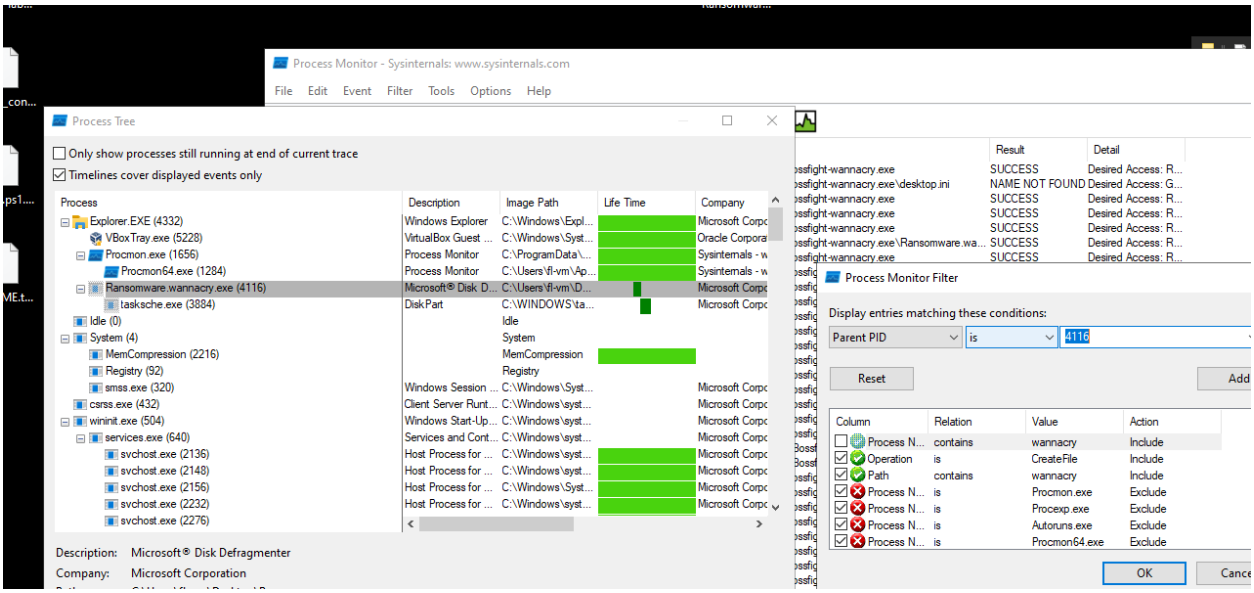


Fig 4: Procmon Filters

We executed our sample (Ransomware.wannacry.exe.malz) on the FLARE-VM.

The malware starts by attempting to connect to the following domain with InternetOpenUrl:

```
www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com
```

NOTE: If this succeeds, the malware immediately exits.

If the connection fails, the malware checks the number of arguments passed to the program. If zero, the malware continues with installation; otherwise it enters service mode.

Wireshark:

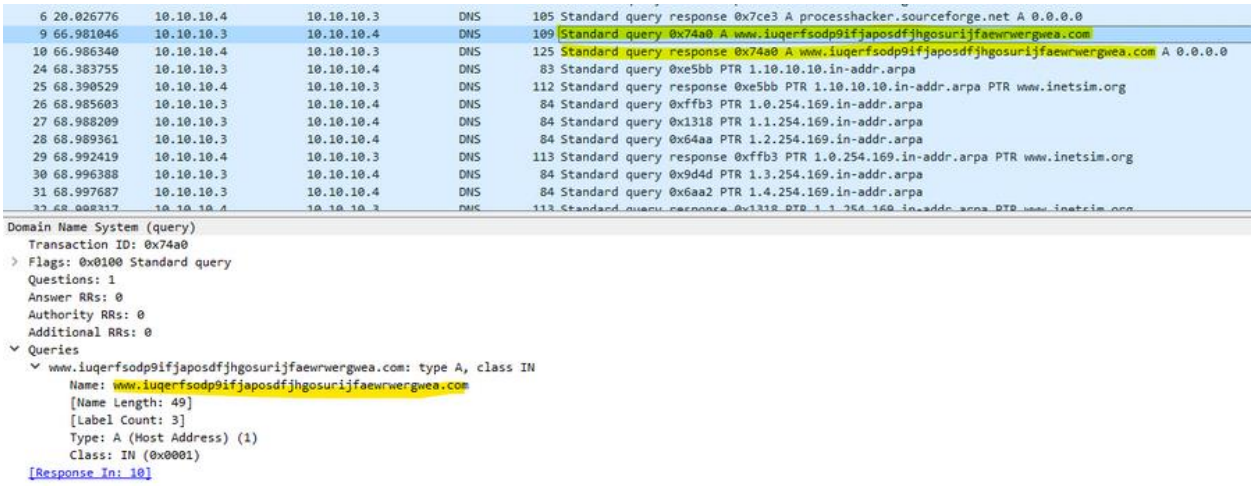


Fig 5: Wireshark logs for the Call Domain

Fig 6: Desktop view after the initial detonation

The malware continues by creating a service named mssecsvc2.0.

Fig 7: Procmon Results

Once created, the malware starts the service.

The malware writes the file C:\Windows\tasksche.exe. The malware executes and then writes a new hidden folder:

C:\ProgramData\ytbjymfcjoghel136\tasksche.exe

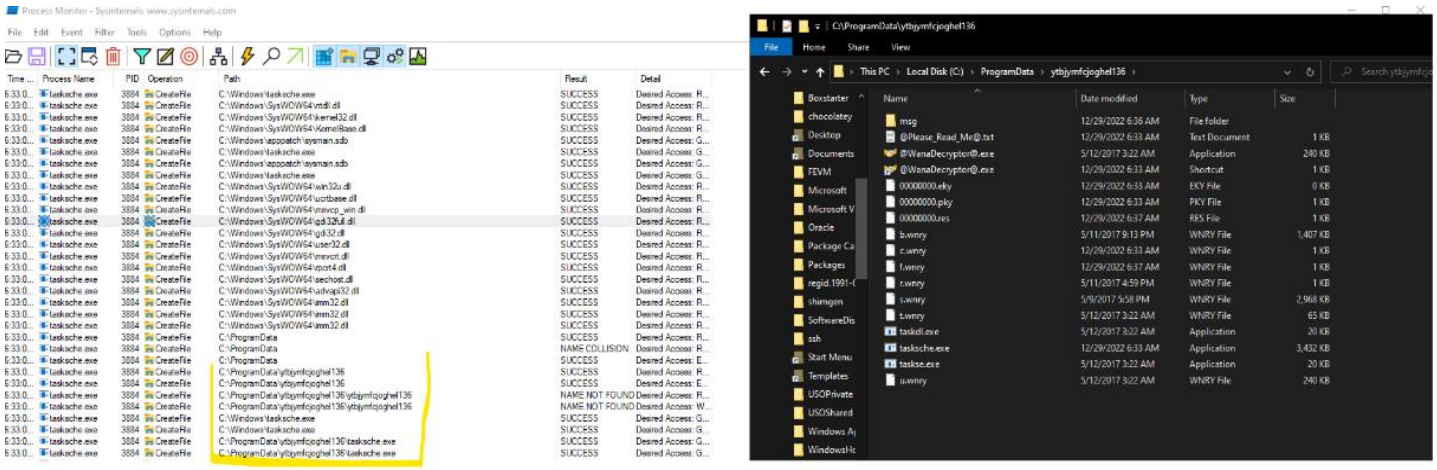


Fig 8: Staging directory

The folder acts as a staging folder for the ransomware malware. It includes all the files needed for the ransomware request, including the message for the users.

Checking Services from Task Manager, I noticed that WannaCry establish its persistence by installing a service in case of restart.

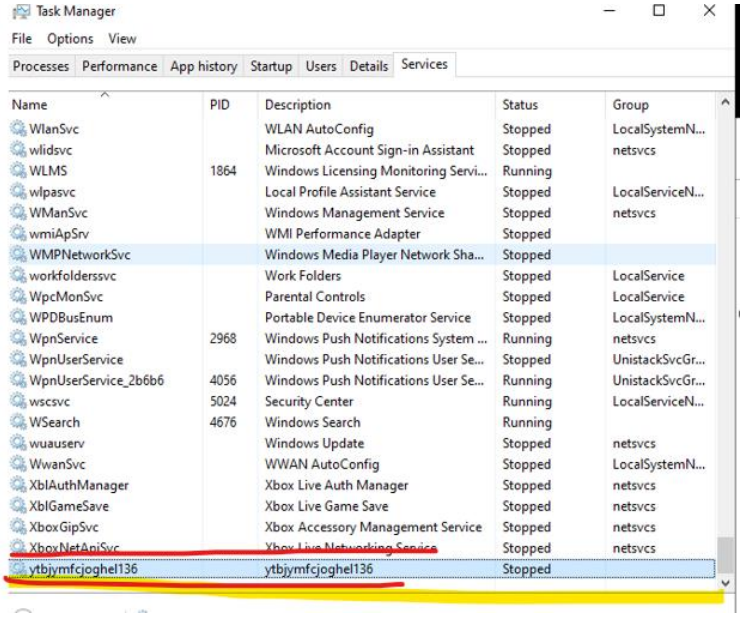


Fig 9: Task Manager

Advanced Static Analysis

{Screenshots and description about findings during advanced static analysis}

Part of WannaCry's infection routine involves sending a DNS request that checks for a live URL/domain. If its request returns showing that the URL is alive or online, it will activate the kill switch, prompting WannaCry to exit the system and no longer proceed with its propagation and encryption routines. Thus, even if the infected machine restarts, the kill switch will prevent WannaCry from performing its routines on it.

Decompiler Graph of the Wannacry:

The "killswitch" mechanism can be found in the disassembly of the main() function:

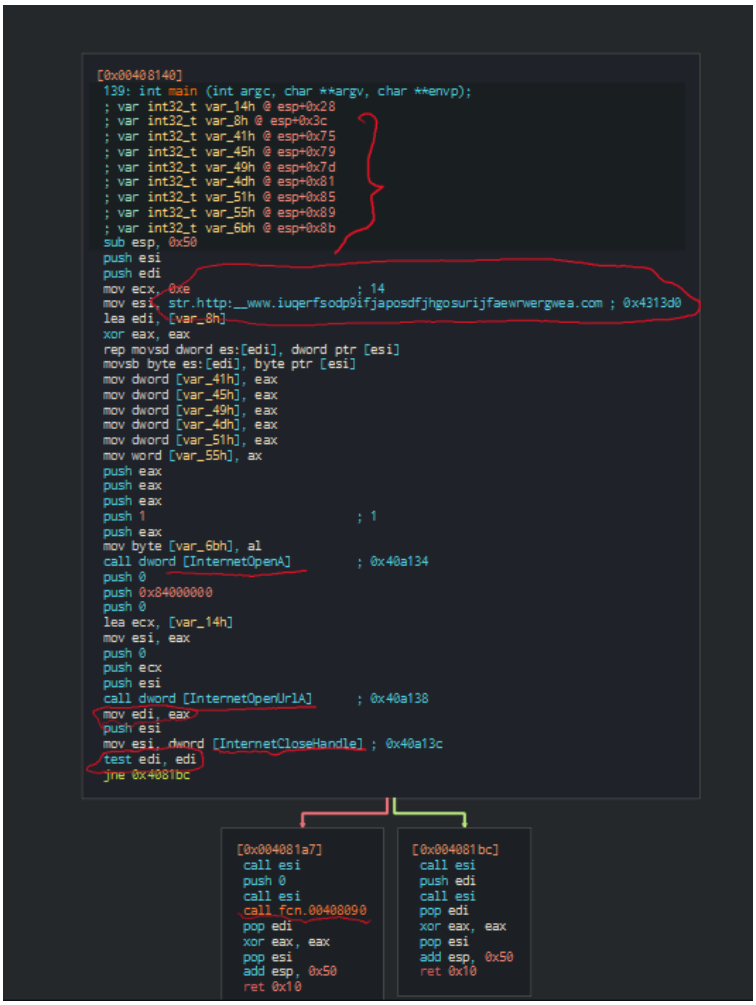


Fig 10: Graph view from the Cutter Decompiler

Noticed from the Decompiler Graph, the URL string containing the killswitch URL.

It moves the string into the esi Register:

```
mov     esi, str.http:__www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com ; 0x4313d0
```

It can also be noticed some important API calls:

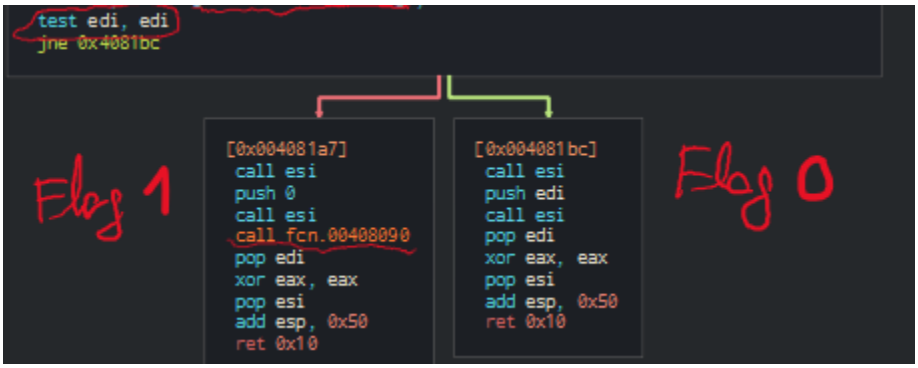
```
call    dword [InternetOpenA]      ; 0x40a134

call    dword [InternetOpenUrlA]   ; 0x40a138

mov     esi, dword [InternetCloseHandle] ; 0x40a13c
```

Everything is moved into edi Register where it performs a test, if it has a flag value of 0, it will continue executing the malware, if it has a flag value of 1, it will return and kill the process. The flag value depends if the call to the killswitch domain succeeds or not.

```
test    edi, edi
```



For Flag 0, the malware manages to connect to the killswitch domain, and the malware immediately exits.

```
add esp, 0x50
ret 0x10
```

For Flag 1, the malware doesn't succeed in connecting to the killswitch Domain and it performs a call to another function:

```
call    fcn.00408090
```

Advanced Dynamic Analysis

{Screenshots and description about advanced dynamic artifacts and methods}

We will start analyzing the WannaCry ransomware from that killswitch URL:

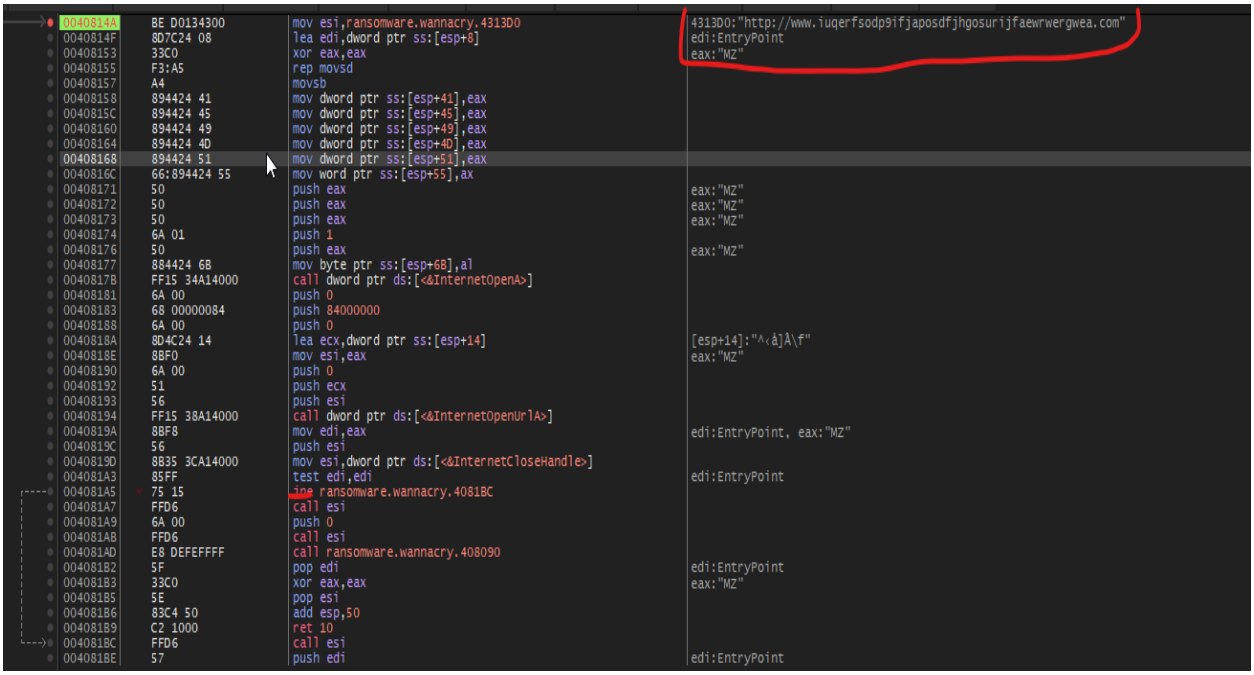


Fig 11: x32dbg Debugger

We will step over until we reach the *test edi,edi* function

The program tests the reachability to the killswitch URL using the API call InternetOpenUrlA.

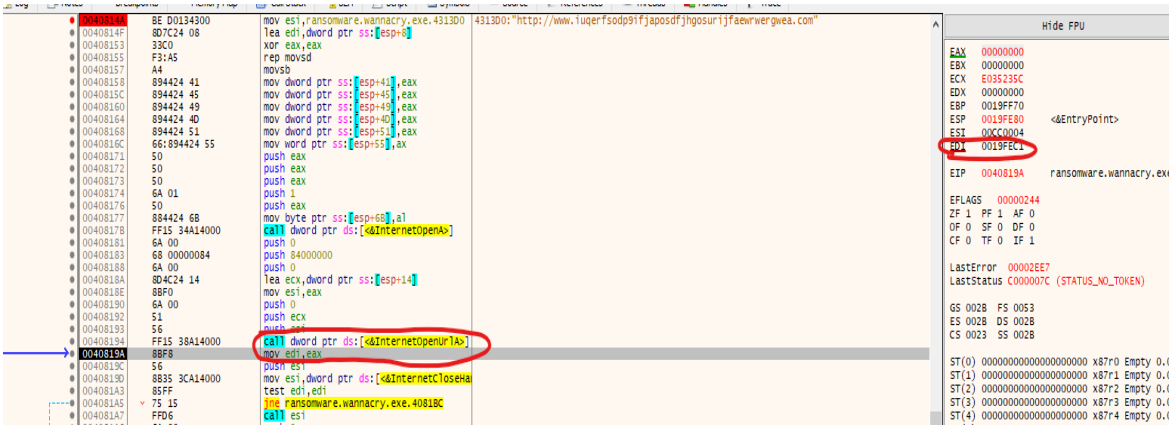


Fig 12: InternetOpenUrlA API call in x32dbg

It can be observed that the value of the EDI is set at this point in the function. The debugger was run without inetsim running, so the EDI it is not 0.

If the program manages to reach the killswitch URL, it will have a flag of 0 and it will jump (jne) and exit the program:

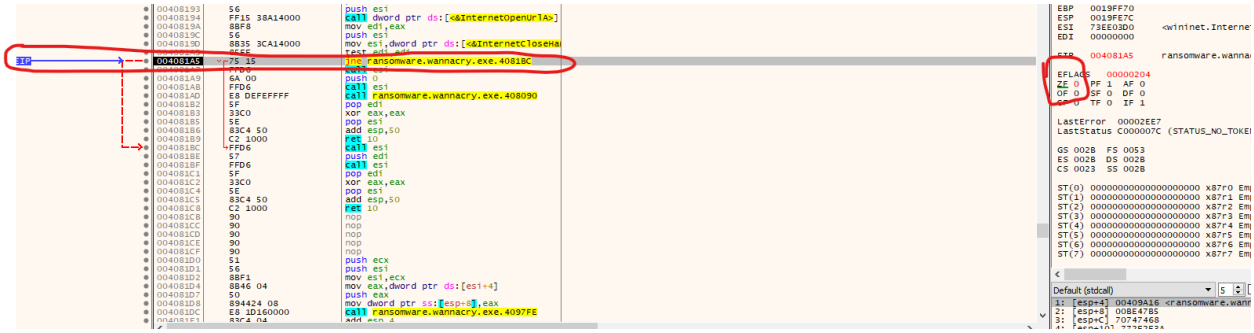


Fig 13: jne instruction (killswitch)

If the flag is set to 1, it will not jump (jne) and it will continue the program, infecting the host.

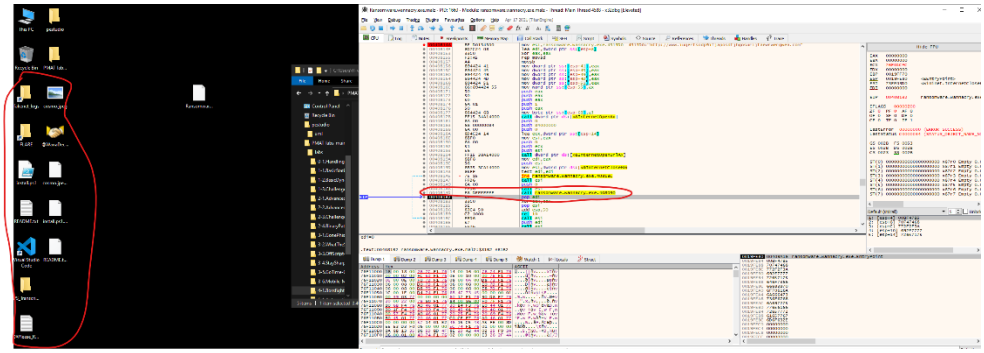


Fig 13: x32dbg execute the WannaCry ransomware from the Debugger

Indicators of Compromise

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

Network Indicators

{Description of network indicators}

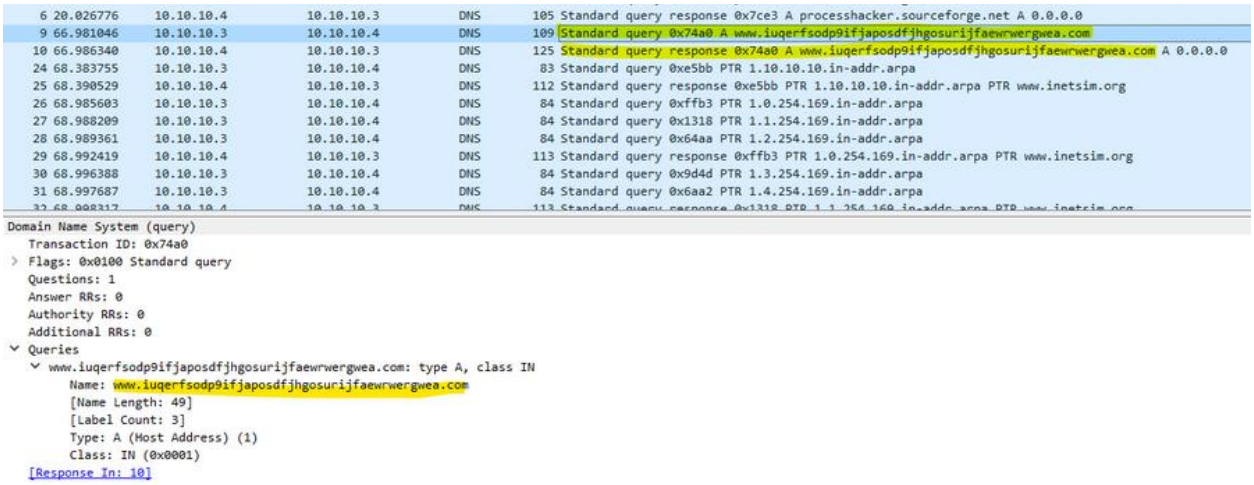


Fig 14: WireShark Packet Capture of the initial call to the killswitch URL

Fig 15: TCPView

The executable makes some TCP calls to an IP on port 445. (that IP from the image above it is not the actual malicious IP, it is generated automatically when the DHCP server is not reachable)

WannaCry takes advantage of the Eternal Blue exploit to spread throughout the network, it is still one of the most commonly attempted exploits against SMB, accounting for over 91.88% of the attacks on port 445 (the most common SMB port).

Host-based Indicators

{Description of host-based indicators}

1. The directory used for the staging area for WannaCry, Fig 1

Note: The directory may be different from host to host.

2. Establish persistence by installing a service in case of restart.

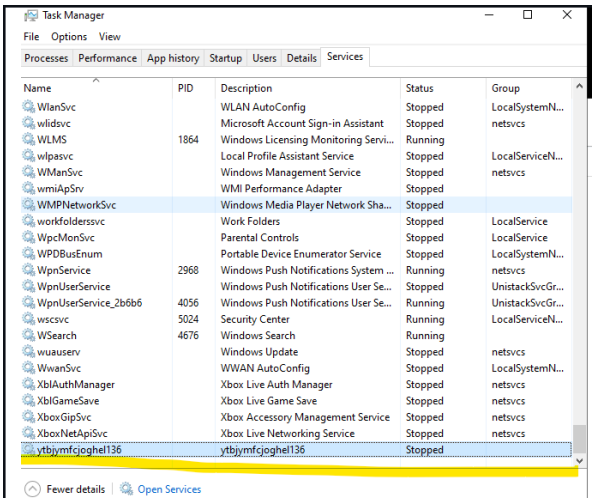
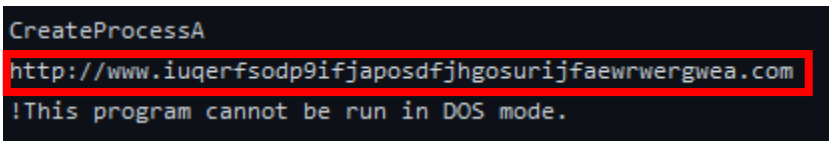


Fig 16: Task Manager – Service for persistence

3. *Call to the malicious URL*



4. The malware changes the Desktop wallpaper and adds the decryptor and other files on the desktop.

Rules & Signatures

A full set of YARA rules is included in Appendix A.

The WannaCry ransomware malware has very apparent signature,

- various obvious strings can be found in the binary: “wnry”, “WANACRY!”, “WNcry@2017”
- callback to that well-known URL used for WannaCry Ransomware

Appendices

A. Yara Rules

Full Yara repository located at:
https://github.com/cosmin-stan/Malware-Analysis/tree/main/Malware/Ransomware/Wannacry

```
rule Ransomware_WannaCry {

    meta:
        last_updated = "2021-01-08"
        author = "cosmin_stan"
        description = "A sample Yara rule to detect WannaCry Ransomware"

    strings:
        $PE_magic_byte = "MZ"
        $string1 = "mssecsvc2.0" ascii
        $string2 = "tasksche.exe" ascii
        $string3 = "http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com"
ascii
        $string4 = "WANACRY!" ascii
        $string5 = "WNcry@2ol7" ascii
        $string6 = "WanaCrypt0r" ascii
    condition:
        $PE_magic_byte at 0 and
        $string1 or $string2 or $string3 or $string4 or $string5 or $string6

}
```

```
C:\Users\fl-vm\Desktop
λ yara32 yara_template.yara -r C:\Users\fl-vm\Desktop -w -s -p 32
Ransomware_WannaCry C:\Users\fl-vm\Desktop\PMAT-labs-main\labs\4-1.Bossfight-wannacry.exe\answers\README.md
0x46f:$string2: tasksche.exe
0xafc:$string2: tasksche.exe
0x323:$string3: http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com
0x641:$string3: http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com
0x4a0:$string5: WNcry@2ol7
Ransomware_WannaCry C:\Users\fl-vm\Desktop\yara_template.yara
0xf7:$string1: mssecsvc2.0
0x11e:$string2: tasksche.exe
0x146:$string3: http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com
0x19a:$string4: WANACRY!
0x1be:$string5: WNcry@2ol7
0x1e4:$string6: WanaCrypt0r
Ransomware_WannaCry C:\Users\fl-vm\Desktop\Ransomware.wannacry.exe
0x0:$PE_magic_byte: MZ
0x312fc:$string1: mssecsvc2.0
0x3136c:$string2: tasksche.exe
0x4157c:$string2: tasksche.exe
0x313d0:$string3: http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com
0x40c20:$string4: WANACRY!
0x415d0:$string5: WNcry@2ol7
```

Fig 17: Testing the Yara rule

B. Callback URL

Domain	Port
hxxp://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com	443

C. Decompiled Code Snippets

```
[0x00408140]
139: int main (int argc, char **argv, char **envp);
; var int32_t var_14h @ esp+0x28
; var int32_t var_8h @ esp+0x3c
; var int32_t var_41h @ esp+0x75
; var int32_t var_45h @ esp+0x79
; var int32_t var_49h @ esp+0x7d
; var int32_t var_4dh @ esp+0x81
; var int32_t var_51h @ esp+0x85
; var int32_t var_55h @ esp+0x89
; var int32_t var_6bh @ esp+0x8b
sub     esp, 0x50
push    esi
push    edi
mov     ecx, 0xe                    ; 14
mov     esi, str.http:__www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com ; 0x4313d0
lea     edi, [var_8h]
xor     eax, eax
rep     movsd dword es:[edi], dword ptr [esi]
movsb   byte es:[edi], byte ptr [esi]
mov     dword [var_41h], eax
mov     dword [var_45h], eax
mov     dword [var_49h], eax
mov     dword [var_4dh], eax
mov     dword [var_51h], eax
mov     word [var_55h], ax
push    eax
push    eax
push    eax
push    1                           ; 1
push    eax
mov     byte [var_6bh], al
call    dword [InternetOpenA]       ; 0x40a134
push    0
push    0x84000000
push    0
lea     ecx, [var_14h]
mov     esi, eax
push    0
push    ecx
push    esi
call    dword [InternetOpenUrlA]    ; 0x40a138
mov     edi, eax
push    esi
mov     esi, dword [InternetCloseHandle] ; 0x40a13c
test    edi, edi
jne     0x4081bc
```

```
[0x004081a7]
call    esi
push    0
call    esi
call    fcn.00408090
pop     edi
xor     eax, eax
pop     esi
add     esp, 0x50
ret     0x10
```

```
[0x004081bc]
call    esi
push    edi
call    esi
pop     edi
xor     eax, eax
pop     esi
add     esp, 0x50
ret     0x10
```

Fig 17: Process Injection Routine in Cutter