



Practical Malware Analysis & Triage

Malware Analysis Report

WannaCry Malware

Dec 2021 | Tuhin Chowdhury | v1.0



Table of Contents

Table of Contents	2
Executive Summary	3
High-Level Technical Summary	4
Malware Composition	5
WannaCry.exe	5
tasksche.exe	5
Basic Static Analysis	6
Basic Dynamic Analysis	7
Advanced Static Analysis	8
Advanced Dynamic Analysis	9
Indicators of Compromise	10
Network Indicators	10
Host-based Indicators	11
Rules & Signatures	13
Appendices	14
A. Yara Rules	14
B. Callback URLs	14
C. Decompiled Code Snippets	15



Executive Summary

SHA256 hash	24D004A104D4D54034DBCFFC2A4B19A11F39008A575AA614EA04703480 B1022C
-------------	--

WannaCry is a cryptor-dropper malware sample first identified on May 17th, 2017. It is a Microsoft Visual C++ 6.0 malware that runs on the x64 Windows operating system. It consists of two payloads that are executed in succession following a successful spear phishing attempt. Symptoms of infection include encrypting every file Appendix B, static warning screen popups on the endpoint, and an executable named “taskse.exe” appearing in the C:\ProgramData\hytvxvlucrksbt351 directory.

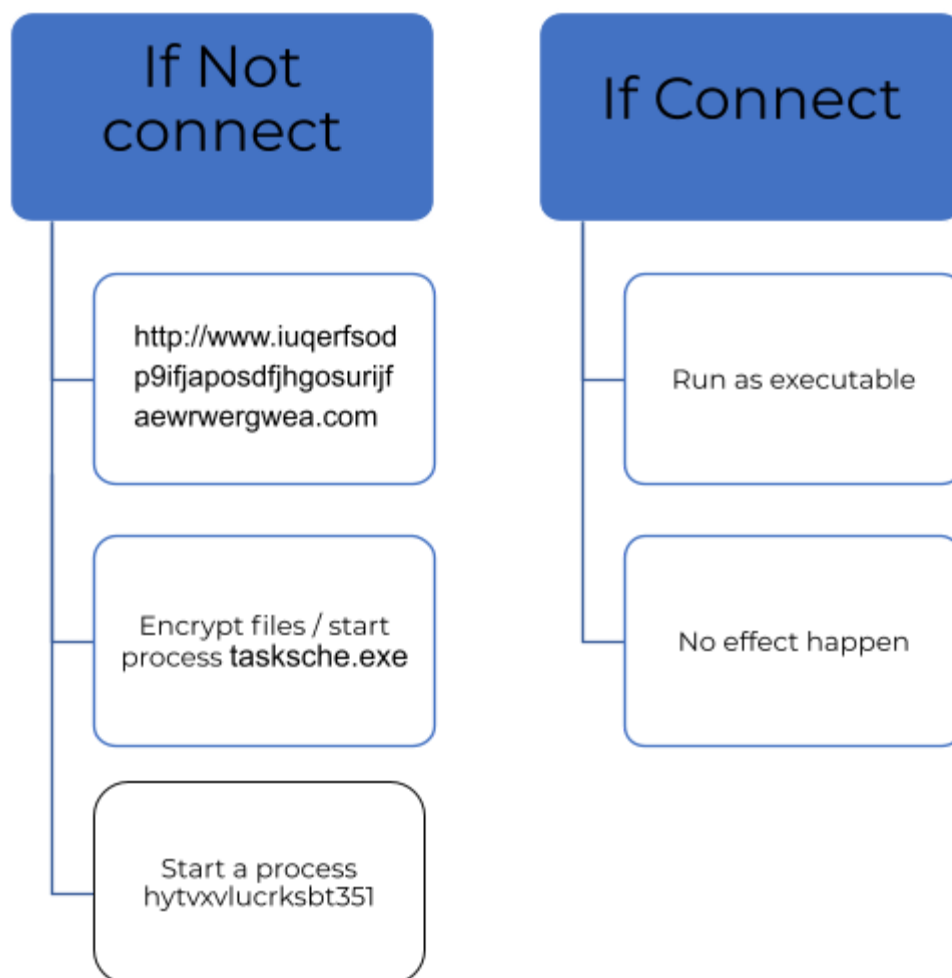
YARA signature rules are attached in Appendix A. Malware sample and hashes have been submitted to VirusTotal for further examination.



High-Level Technical Summary

WannaCry consists of two parts: an encrypted stage 0 dropper and an unpacked and decoded stage 2 command execution program. It first attempts to contact its callback URL

(hXXtp://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com)





Malware Composition

WannaCry consists of the following components:

File Name	SHA256 Hash
WannaCry	24D004A104D4D54034DBCFFC2A4B19A11F39008A575AA614EA04703480B1022C

WannaCry.exe

The initial executable that runs if the program don't make the connection with the callback url

A directory created in C:\ProgramData\hytvxvlucrksbt351

The screenshot displays the Windows Task Manager and File Explorer. The Task Manager window shows the 'Processes' tab with a list of running services. The 'hytvxvlucrksbt351' process is highlighted in red. The File Explorer window shows the 'This PC' view with the 'Local Disk (C:)' selected. The 'ProgramData' folder is expanded, and the 'hytvxvlucrksbt351' folder is highlighted in red. The 'Process Monitor' window in the background shows a list of events, including the creation of the 'hytvxvlucrksbt351' directory.

WannaCry Malware
Dec 23
v1.0



Basic Static Analysis

{Screenshots and description about basic static artifacts and methods}

String / Floss Output	http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com KERNEL32.dll mscoree.dll MM/dd/yy CryptEncrypt CryptDecrypt CryptDestroyKey CryptGenRandom
-----------------------------	---

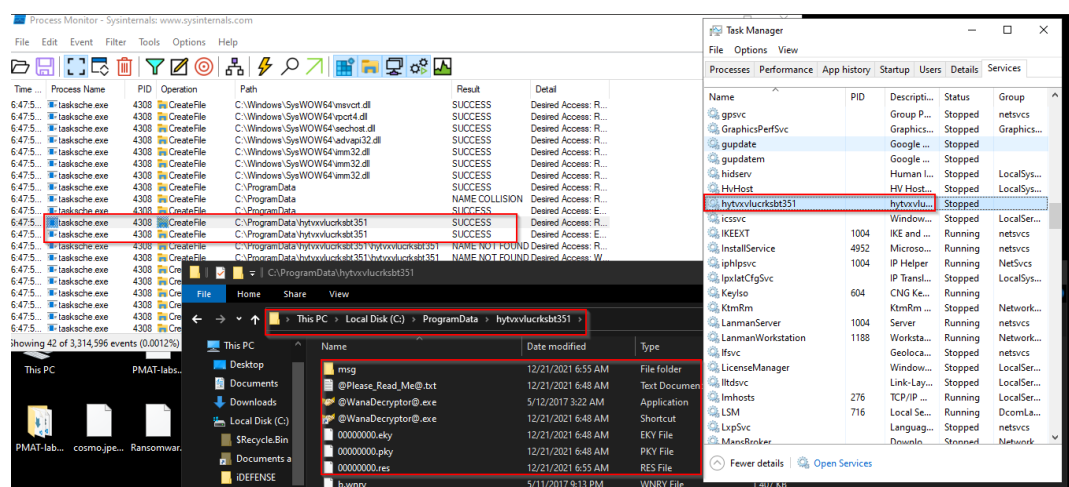
We can find the url and some windows API related to cryptography, So as far our basic analysis we conclude that some function of the malware work with crypto. So it can be ransomware.



Basic Dynamic Analysis

{Screenshots and description about basic dynamic artifacts and methods}

Create file
(It create a directory in
%ProgramData% name
hytvxvlucrksbt351
and file taskse.exe)



BD-Figure-1

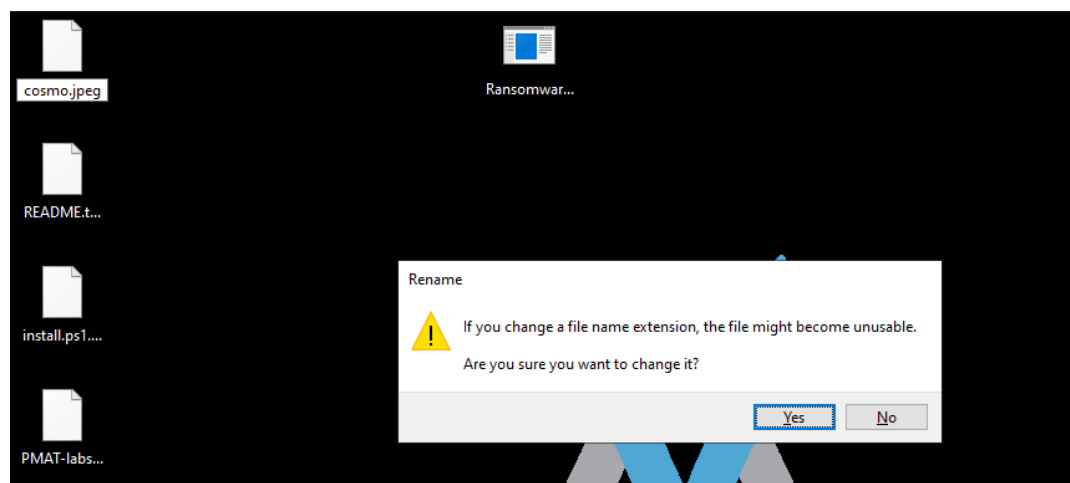
taskshvc.exe connection
status

wininit.exe	500	TCP	Listen	0.0.0.0	49665	0.0.0.0	0	11/22/2021 11:03:...
taskshvc.exe	4776	TCP	Established	127.0.0.1	9050	127.0.0.1	62417	12/23/2021 1:07:...
taskshvc.exe	4776	TCP	Listen	127.0.0.1	9050	0.0.0.0	0	12/23/2021 1:01:...
taskshvc.exe	4776	TCP	Established	127.0.0.1	21409	127.0.0.1	21411	12/23/2021 1:01:...
taskshvc.exe	4776	TCP	Established	127.0.0.1	21411	127.0.0.1	21409	12/23/2021 1:01:...
System	4	UDP		10.0.0.3	138	*		12/23/2021 12:28:...

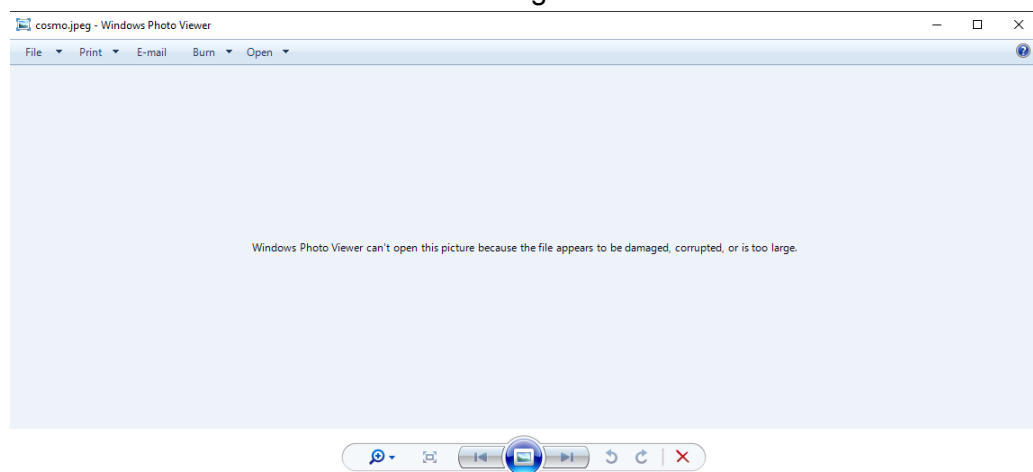
BD-Figure-2



Try to open after deleting
extension



BD-Figure-3

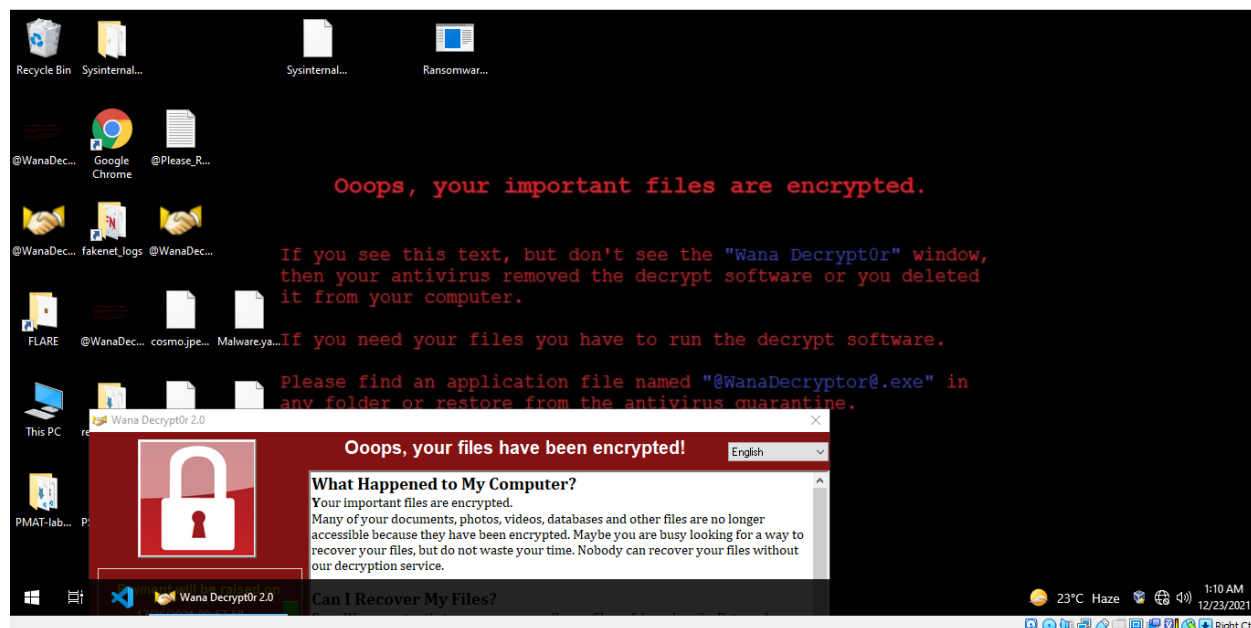


BD-Figure-4

In the BD-Figure-1 we can see a directory and a file created in %ProgramData% folder and start a process. A Program name taskhsvc.exe starts listening on 9050 port (BD-Figure-2). When we try to open any file after removing the extension (BD-Figure-3), We got an output like BD-Figure-4



d



After execution the program the wallpaper changed and a decrypter program start with a message and also files got encrypted

WannaCry Malware
Dec 23
v1.0



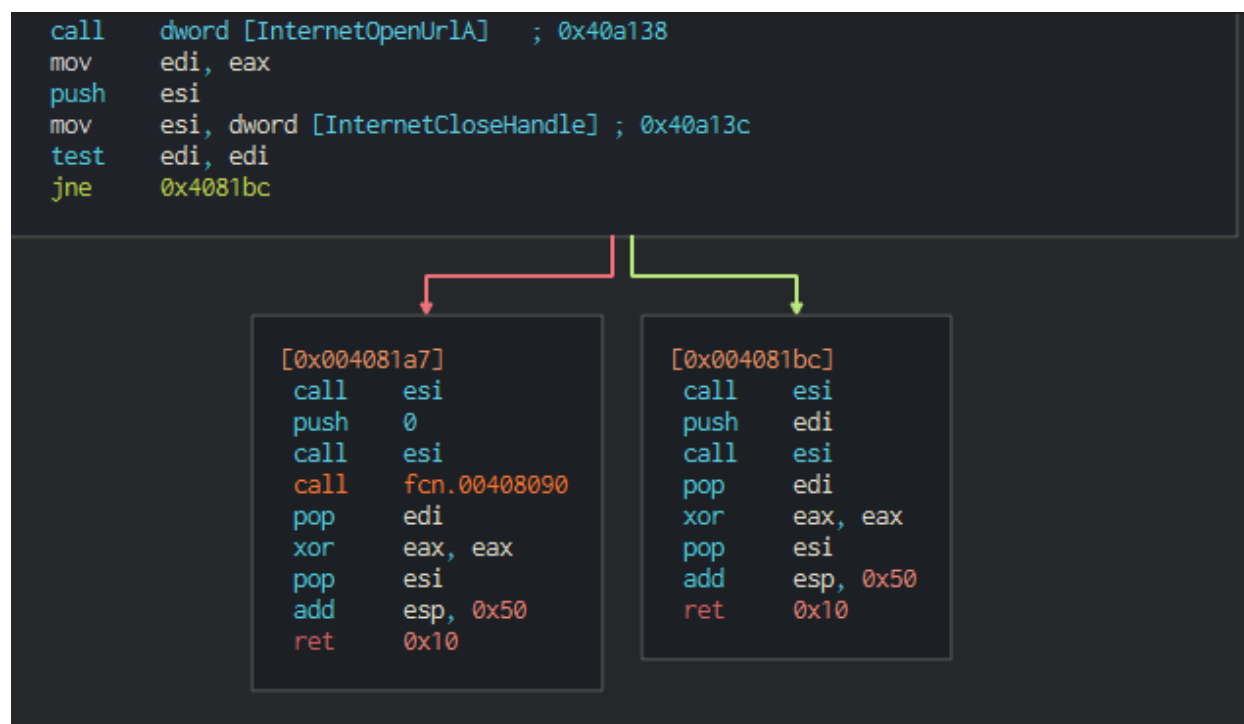
Advanced Static Analysis

{Screenshots and description about findings during advanced static analysis}

```
sub     esp, 0x50
push    esi
push    edi
mov     ecx, 0xe             ; 14
mov     esi, str.http:___www.iuqerfsodp9ifjaposdfjhgosurijfaewrnwergwea.com ; 0x4313d0
lea     edi, [var_8n]
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
```

AS-Figure-1

In the AS-Figure-1 Cutter output we can see the call back url and first InternetOpenA windows API call.



AS-Figure-2

In the AS-figure-2 Cutter output we can see the conditional logic if code execution as per the screenshot in both condition the code approximately run same but in left hand indicator a special function call in 00408090 location. So we can determine this the separation part of the logic flow.



```
int32_t var_45h;  
int32_t var_49h;  
int32_t var_4dh;  
int32_t var_51h;  
int32_t var_55h;  
int32_t var_6bh;  
ecx = 0xe;  
esi = "http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrgwea.com";  
edi = &var_8h;  
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, ax, al);  
ecx = &var_14h;  
esi = eax;  
eax = uint32_t (*InternetOpenUrlA)(void, void, void, void, void, void) (esi, ecx, 0, 0, 0x84000000, 0);  
edi = eax;  
esi = *(InternetCloseHandle);  
if (edi == 0) {  
    void (*esi)() ();  
    void (*esi)(void) (0);  
    eax = fcn_00408090 ();  
}
```

AS-Figure-3

This is the decompile main function code. Here we can actually where the url store and when the API call happens.



Advanced Dynamic Analysis

{Screenshots and description about advanced dynamic artifacts and methods}

```
004081A5 75 15 jne_ransomware.wannacry.4081A5
004081A7 FFD6 call esi
004081A9 6A 00 push 0
004081AB FFD6 call esi
004081AD E8 DEFEFFFF call_ransomware.wannacry.408090
004081B2 5F pop edi
004081B3 33C0 xor eax,eax
004081B5 5E pop esi
004081B6 83C4 50 add esp,50
004081B9 C2 1000 ret 10
004081BC FFD6 call esi
004081BE 57 push edi
004081BF FFD6 call esi
004081C1 5F pop edi
004081C2 33C0 xor eax,eax
```

EIP 004081A5 ransomware.wannacry.004081A5

EFLAGS 00000244
ZF 1 PF 1 AF 0
OF 0 SF 0 DF 0
CF 0 TF 0 IF 1

LastError: 00000000 (ERROR_SUCCESS)
LastStatus: C000007C (STATUS_NO_TOKEN)

AD-Figure-1

In the x32 debugger we change the flag 0 to 1 manually to see the result. In that condition the program connect with the callback url in 1 flag but as we changed this into 0 so the program code manipulated and don't get the connection.

```
004081A5 75 15 jne_ransomware.wannacry.4081A5
004081A7 FFD6 call esi
004081A9 6A 00 push 0
004081AB FFD6 call esi
004081AD E8 DEFEFFFF call_ransomware.wannacry.408090
004081B2 5F pop edi
004081B3 33C0 xor eax,eax
004081B5 5E pop esi
004081B6 83C4 50 add esp,50
004081B9 C2 1000 ret 10
004081BC FFD6 call esi
```

EIP 004081A5 ransomware.wannacry.004081A5

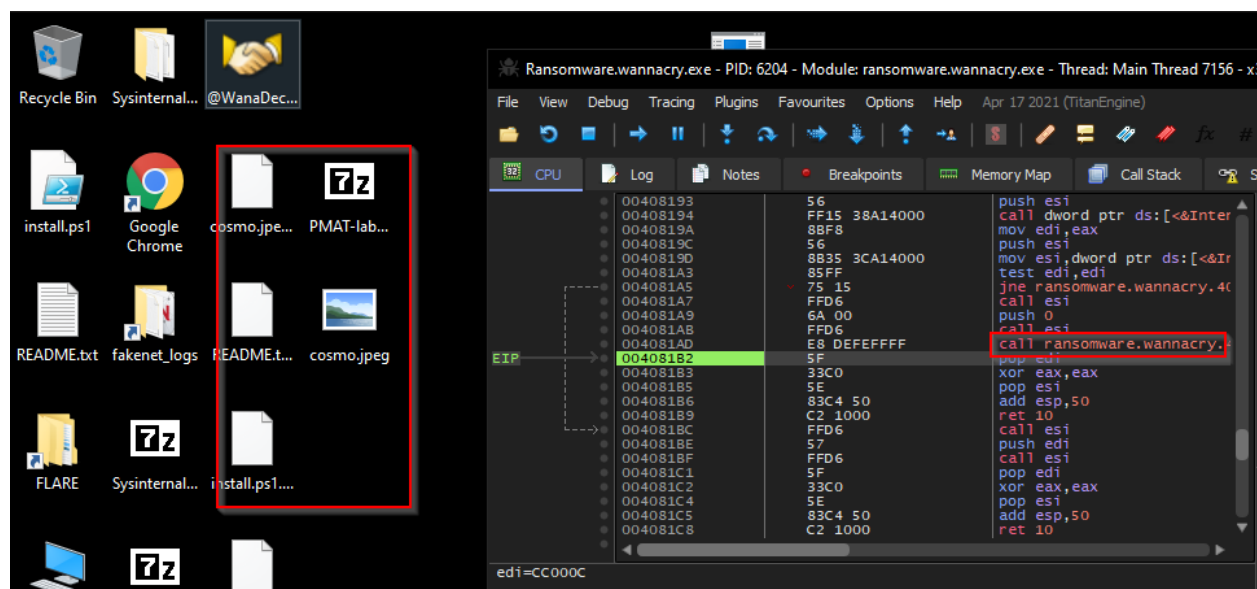
EFLAGS 00000244
ZF 1 PF 1 AF 0
OF 0 SF 0 DF 0
CF 0 TF 0 IF 1

LastError: 00000000 (ERROR_SUCCESS)
LastStatus: C000007C (STATUS_NO_TOKEN)

Before taking the call (Files are not encrypted)

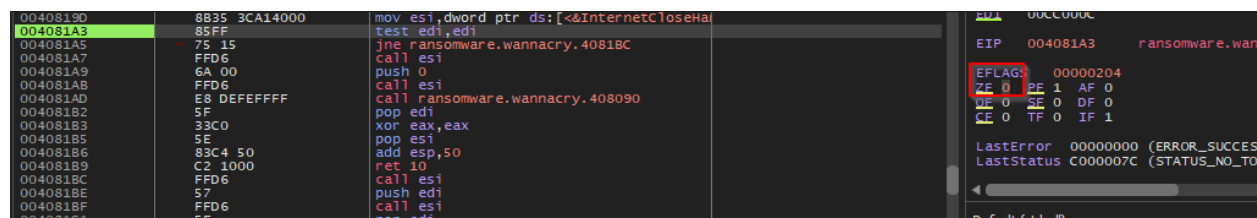


d



After taking the call (Files got encrypted)

Before taking the call:



In this condition we let the flag 0

After Taking the call:



d

Address	Disassembly	Comment
0040819A	8BF8	mov edi, eax
0040819C	56	push esi
0040819D	8B35 3CA14000	mov esi, dword ptr ds:[<&InternetCloseHa
004081A3	85FF	test edi, edi
004081A5	75 15	jne ransomware.wannacry.4081BC
004081A7	FFD6	call esi
004081A9	6A 00	push 0
004081AB	FFD6	call esi
004081AD	E8 DEFEFFFF	call ransomware.wannacry.408090
004081B2	5F	pop edi
004081B3	33C0	xor eax, eax
004081B5	5E	pop esi
004081B6	83C4 50	add esp, 50
004081B9	C2 1000	ret 10
004081BC	FFD6	call esi
004081BD	57	push edi
004081BF	FFD6	call esi
004081C1	5F	pop edi
004081C2	33C0	xor eax, eax
004081C4	5E	pop esi
004081C5	83C4 50	add esp, 50

EIP 004081BC

ESI 72C203D0

EDI 00CC000C

EFLAGS 00000206

ZF 0 PF 1 AF 0

OF 0 SF 0 DF 0

CF 0 TF 0 IF 1

LastError 00000000

LastStatus C000007C

Default (stdcall)

1: [esp] 00CC0004

2: [esp+4] 00409A16

3: [esp+8] 00C64908

4: [esp+C] 70747468

5: [esp+10] 77353533

Return:

Address	Disassembly	Comment
004081C2	33C0	xor eax, eax
004081C4	5E	pop esi
004081C5	83C4 50	add esp, 50
004081C8	C2 1000	ret 10

In the above scenario as you can see if the program successfully connects with the callback url then it jumps from 004081A3 location to 004081BC. But if it doesn't connect it go into 004081A3 to and continues to 004081A5 to 004081B9. This is the main execution portion.



Indicators of Compromise

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

Network Indicators

{Description of network indicators}

1	0.00000000	10.0.0.3	10.0.0.4	TCP	66 49673 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1
2	0.000030310	10.0.0.4	10.0.0.3	TCP	66 80 → 49673 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1
3	0.002363366	10.0.0.3	10.0.0.4	TCP	66 49673 → 80 [ACK] Seq=1 Ack=1 Win=262144 Len=0
4	0.002363601	10.0.0.3	10.0.0.4	HTTP	154 GET / HTTP/1.1
5	0.002418644	10.0.0.4	10.0.0.3	TCP	54 80 → 49673 [ACK] Seq=1 Ack=101 Win=64256 Len=0
6	0.040955639	10.0.0.4	10.0.0.3	TCP	204 80 → 49673 [PSH, ACK] Seq=1 Ack=101 Win=64256 Len=150 [TCP segment o...]
7	0.041991328	10.0.0.3	10.0.0.4	TCP	60 49673 → 80 [ACK] Seq=101 Ack=151 Win=261888 Len=0

▶ Frame 4: 154 bytes on wire (1232 bits), 154 bytes captured (1232 bits) on interface enp0s17, id 0
▶ Ethernet II, Src: PcsCompu_f5:09:32 (08:00:27:f5:09:32), Dst: PcsCompu_d7:3a:25 (08:00:27:d7:3a:25)
▶ Internet Protocol Version 4, Src: 10.0.0.3, Dst: 10.0.0.4
▶ Transmission Control Protocol, Src Port: 49673, Dst Port: 80, Seq: 1, Ack: 1, Len: 100
▶ Hypertext Transfer Protocol
 ▶ GET / HTTP/1.1\r\n
 Host: www.iuqerfsodp9ifjaposdfjhgosurijfaewrgwea.com\r\n
 Cache-Control: no-cache\r\n
 \r\n
 [Full request URI: http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrgwea.com/]
 [HTTP request 1/1]
 [Response in frame: 9]

Fig 3: WireShark Packet Capture of initial beacon check-in

Rules & Signatures

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

{Information on specific signatures, i.e. strings, URLs, etc}

WannaCry Malware

Dec 23

v1.0



Appendices

A. Yara Rules

```
rule Yara_Example {  
  
    meta:  
        last_updated = "2021-12-23"  
        author = "Tuhin"  
        description = "rule for wanna cry"  
  
    strings:  
        // simple rules  
        $string1 = "Oops, your files have been encrypted!" wide ascii nocase  
        $string2 = "Wanna Decryptor" wide ascii nocase  
        $PE_magic_byte = "MZ"  
        $sus_hex_string = { FF E4 ?? 00 FF }  
  
    condition:  
        // Basic condition  
        ($PE_magic_byte at 0 and  
        ($string1 and $string2) or  
        $sus_hex_string  
}
```

B. Callback URLs

Domain	Port
http://www.iuqerfsodp9ifjaposdfjhgosurijf aewrwergrwea.com	80



C. Decompiled Code Snippets

```
int32_t var_45h;  
int32_t var_49h;  
int32_t var_4dh;  
int32_t var_51h;  
int32_t var_55h;  
int32_t var_6bh;  
ecx = 0xe;  
esi = "http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrgwea.com";  
edi = &var_8h;  
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, ax, al);  
ecx = &var_14h;  
esi = eax;  
eax = uint32_t (*InternetOpenUrlA)(void, void, void, void, void, void) (esi, ecx, 0, 0, 0x84000000, 0);  
edi = eax;  
esi = *(InternetCloseHandle);  
if (edi == 0) {  
    void (*esi)() ();  
    void (*esi)(void) (0);  
    eax = fcn_00408090 ();
```

Fig 5: Process Injection Routine in Cutter

Program Execution flow

- If url connected successfully
 - Malware not executed
 - The host seems ok with no errors.
- If url cannot connect
 - Program execute
 - Encrypt every file
 - Start a process called tasksche.exe
 - Create a file in Program Data
 - Change the background