Malware Analysis: “wannacry.exe”

Contents

Tools:

1. FlareVM
2. Pestudio
3. Floss
4. inetsim

Static Analysis

I ran floss.exe against Wannacry to retrieve any string that can be parsed from the binary.

```
GetModuleHandleA
GetStartupInfoA
_stricmp
!This program cannot be run in DOS mode.
CloseHandle
WriteFile
CreateFileA
SizeofResource
LockResource
LoadResource
FindResourceA
CreateProcessA
KERNEL32.dll
MSVCRT.dll
_initterm
_adjust_fdiv
launcher.dll
PlayGame
C:\%s\%s
mssecsvc.exe
!This program cannot be run in DOS mode.
/4%D/4%D/4%D4
D,4%D/4$D
D.4%DRich/4%D
UVWATAUAVAWH
D$HD9T$\
t$pD+d$HD+
A_A^A]A\_^]
WATAUAVAWH
A_A^A]A\_
```

Figure 1Multiple DOS STUB

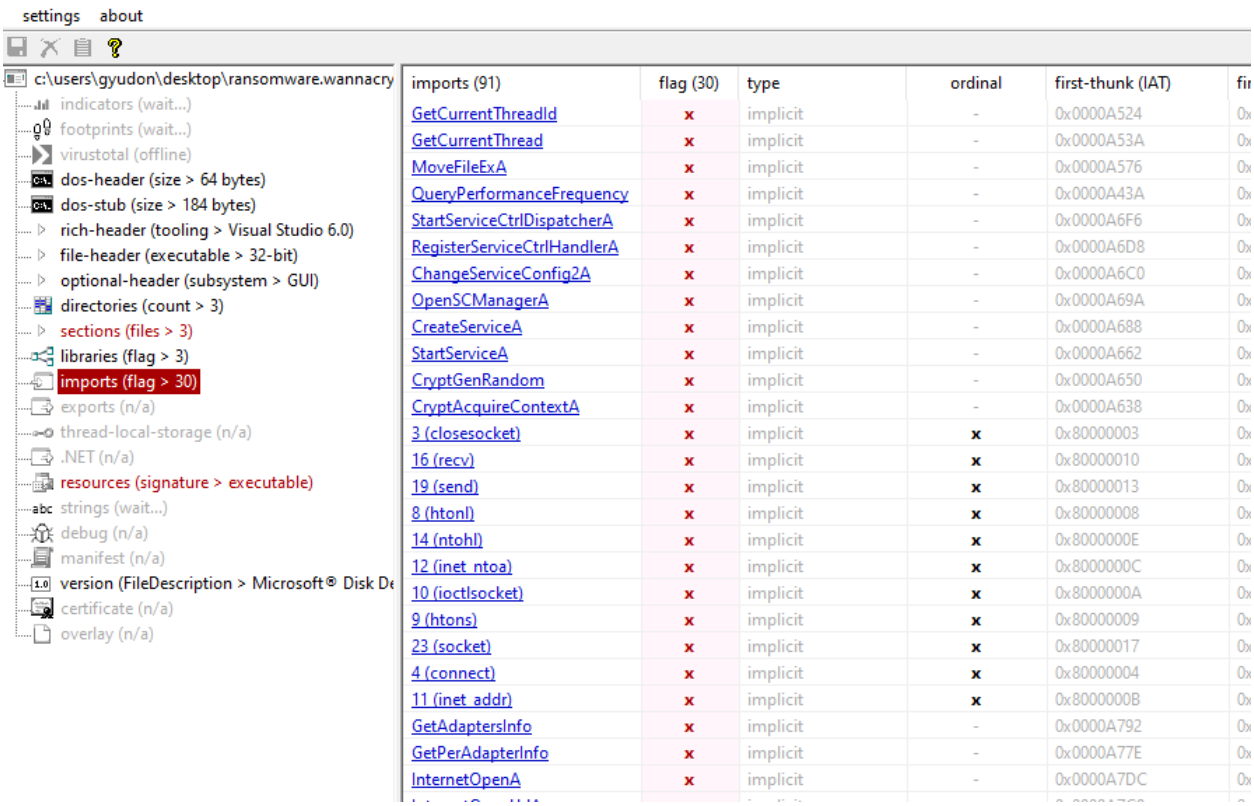
Having multiple DOS STUB indicates a sign of packed binary. There may be more to this binary that may be unpacked at runtime.

```
CloseHandle
WriteFile
CreateFileA
CreateProcessA
http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com
!This program cannot be run in DOS mode.
=j&&LZ66lA??~
f""D~**T
V22dN::t
o%%Jr..\$
```

Figure 2 Wierd URL found also

I also found a URL which I bookmarked as it may be an indication of callback domain.

I used PEstudio to inspect wannacry next.



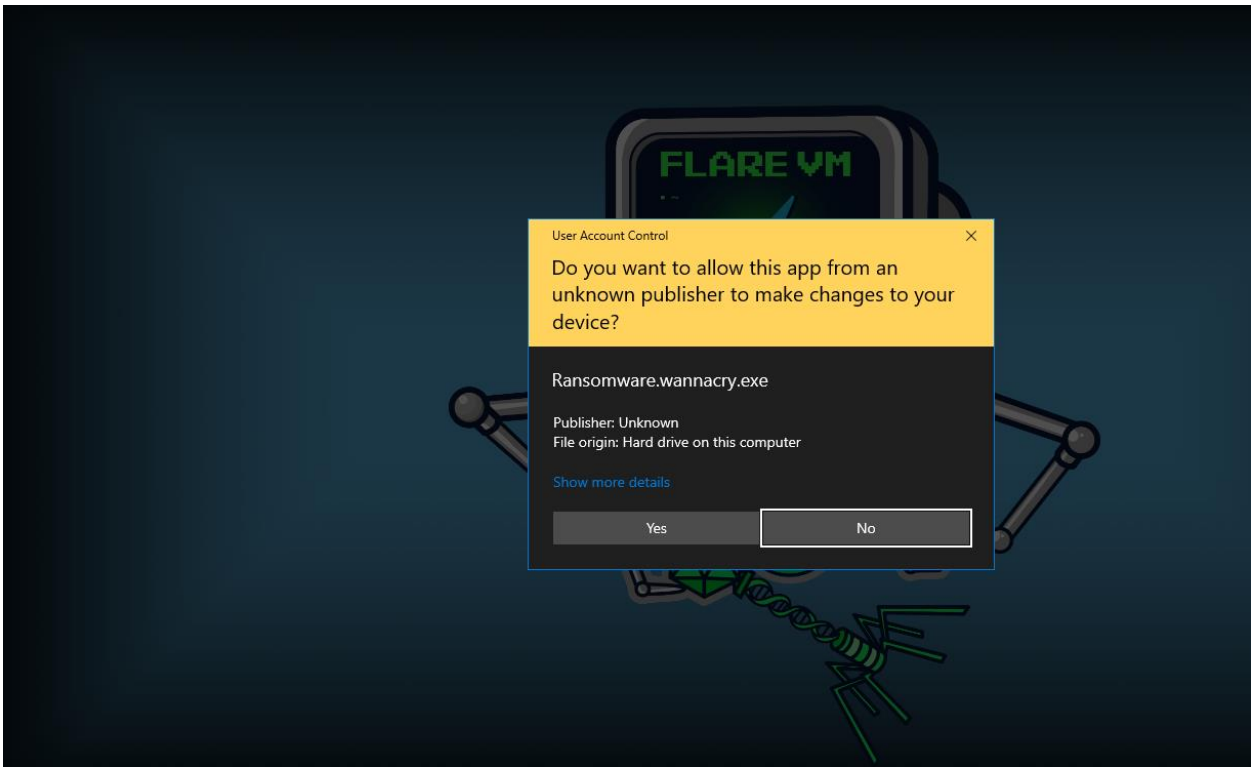
imports (91)	flag (30)	type	ordinal	first-thunk (IAT)
GetCurrentThreadId	x	implicit	-	0x0000A524
GetCurrentThread	x	implicit	-	0x0000A53A
MoveFileExA	x	implicit	-	0x0000A576
QueryPerformanceFrequency	x	implicit	-	0x0000A43A
StartServiceCtrlDispatcherA	x	implicit	-	0x0000A6F6
RegisterServiceCtrlHandlerA	x	implicit	-	0x0000A6D8
ChangeServiceConfig2A	x	implicit	-	0x0000A6C0
OpenSCManagerA	x	implicit	-	0x0000A69A
CreateServiceA	x	implicit	-	0x0000A688
StartServiceA	x	implicit	-	0x0000A662
CryptGenRandom	x	implicit	-	0x0000A650
CryptAcquireContextA	x	implicit	-	0x0000A638
3 (closesocket)	x	implicit	x	0x80000003
16 (recv)	x	implicit	x	0x80000010
19 (send)	x	implicit	x	0x80000013
8 (htonl)	x	implicit	x	0x80000008
14 (ntohl)	x	implicit	x	0x8000000E
12 (inet_ntoa)	x	implicit	x	0x8000000C
10 (ioctlsocket)	x	implicit	x	0x8000000A
9 (htons)	x	implicit	x	0x80000009
23 (socket)	x	implicit	x	0x80000017
4 (connect)	x	implicit	x	0x80000004
11 (inet_addr)	x	implicit	x	0x8000000B
GetAdaptersInfo	x	implicit	-	0x0000A792
GetPerAdapterInfo	x	implicit	-	0x0000A77E
InternetOpenA	x	implicit	-	0x0000A7DC

As expected, there a couple of API that is relevant to Ransomware such as , CryptGenRandom, CryptAcquireContextA. There were also socket api, such as 3(closesocket), 16(recv) and 19(send). Persistence may also be possible from the CreateServiceA API.

Dynamic Analysis

Using my inetsim, I simulated a fake webserver on my REMNUX to allow Wannacry to do DNS query of any domain if it is available.

```
remnux@remnux: ~
remnux@remnux:~$ sudo 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 1360) ===
Session ID:     1360
Listening on:   0.0.0.0
Real Date/Time: 2025-05-10 12:59:33
Fake Date/Time: 2025-05-10 12:59:33 (Delta: 0 seconds)
 Forking services...
  * dns_53_tcp_udp - started (PID 1364)
  * http_80_tcp - started (PID 1365)
  * https_443_tcp - started (PID 1366)
  * smtps_465_tcp - started (PID 1368)
  * ftps_990_tcp - started (PID 1372)
  * pop3s_995_tcp - started (PID 1370)
  * smtp_25_tcp - started (PID 1367)
  * pop3_110_tcp - started (PID 1369)
  * ftp_21_tcp - started (PID 1371)
 done.
Simulation running.
```



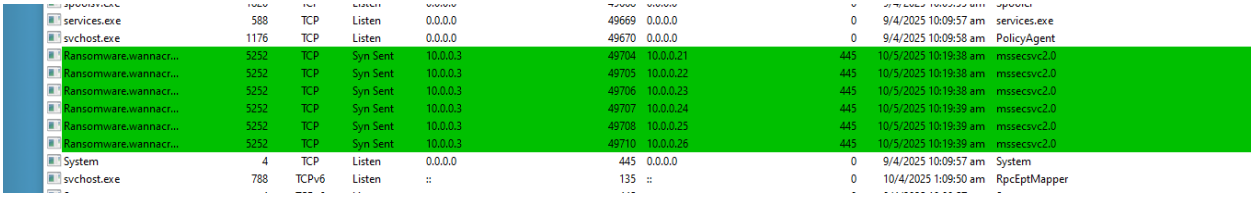
350	59.261733698	10.0.0.1	239.255.255.250	SSDP	167	M-SEARCH * HTTP/1.1
351	61.523048066	10.0.0.3	10.0.0.4	DNS	109	Standard query 0xb305 A www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com
352	61.529829317	10.0.0.4	10.0.0.3	DNS	125	Standard query response 0xb305 A www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com A 10.0.0.4
353	61.543436659	10.0.0.3	10.0.0.4	TCP	66	49700 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
354	61.543457765	10.0.0.4	10.0.0.3	TCP	66	80 → 49700 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
355	61.543739162	10.0.0.3	10.0.0.4	TCP	60	49700 → 80 [ACK] Seq=1 Ack=1 Win=262144 Len=0
356	61.544086561	10.0.0.3	10.0.0.4	HTTP	154	GET / HTTP/1.1
357	61.544093705	10.0.0.4	10.0.0.3	TCP	54	80 → 49700 [ACK] Seq=1 Ack=101 Win=64256 Len=0
358	61.557163271	10.0.0.4	10.0.0.3	TCP	204	80 → 49700 [PSH, ACK] Seq=1 Ack=101 Win=64256 Len=150 [TCP PDU reassembled in 360]
359	61.557849835	10.0.0.3	10.0.0.4	TCP	60	49700 → 80 [ACK] Seq=101 Ack=151 Win=261888 Len=0
360	61.557863964	10.0.0.4	10.0.0.3	HTTP	312	HTTP/1.1 200 OK (text/html)
361	61.558078204	10.0.0.3	10.0.0.4	TCP	60	49700 → 80 [ACK] Seq=101 Ack=409 Win=261632 Len=0
362	61.559290284	10.0.0.3	10.0.0.4	TCP	60	49700 → 80 [FIN, ACK] Seq=101 Ack=409 Win=261632 Len=0
363	61.559299406	10.0.0.4	10.0.0.3	TCP	60	49700 → 80 [RST, ACK] Seq=102 Ack=409 Win=0 Len=0

One interesting thing which I have found about Wannacry is that it would attempt to contact a weird DNS as shown from my Wireshark output, and if it successfully does so, it will not detonate. If it fails, it detonates, which is a stark contrast from normal malware with self-deleting capabilities.

After disabling inetsim

Ransomware Wannacry is detonated upon disabling my internet simulator! Next up , I open up TCPView to see any possible host based network activities from wannacry.

Using TCPView



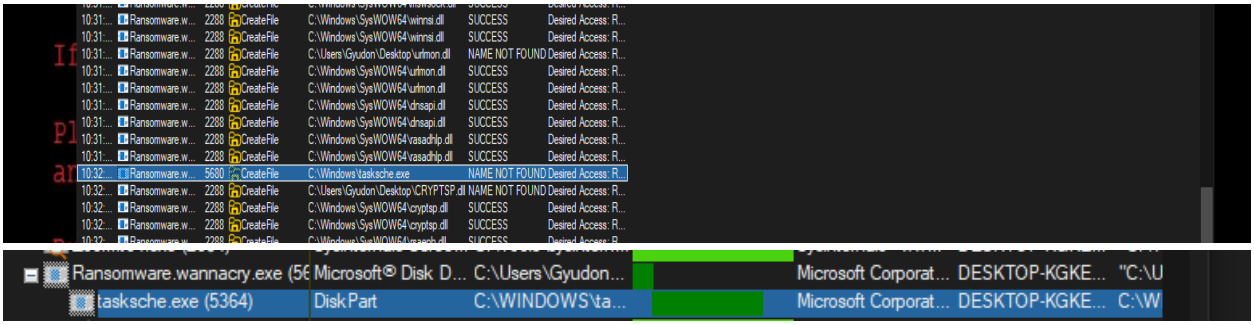
services.exe	588	TCP	Listen	0.0.0.0	49669	0.0.0.0	0	9/4/2025 10:09:57 am	services.exe
svchost.exe	1176	TCP	Listen	0.0.0.0	49670	0.0.0.0	0	9/4/2025 10:09:58 am	PolicyAgent
Ransomware.wannacr...	5252	TCP	Syn Sent	10.0.0.3	49704	10.0.0.21	445	10/5/2025 10:19:38 am	mssecsvc2.0
Ransomware.wannacr...	5252	TCP	Syn Sent	10.0.0.3	49705	10.0.0.22	445	10/5/2025 10:19:38 am	mssecsvc2.0
Ransomware.wannacr...	5252	TCP	Syn Sent	10.0.0.3	49706	10.0.0.23	445	10/5/2025 10:19:38 am	mssecsvc2.0
Ransomware.wannacr...	5252	TCP	Syn Sent	10.0.0.3	49707	10.0.0.24	445	10/5/2025 10:19:39 am	mssecsvc2.0
Ransomware.wannacr...	5252	TCP	Syn Sent	10.0.0.3	49708	10.0.0.25	445	10/5/2025 10:19:39 am	mssecsvc2.0
Ransomware.wannacr...	5252	TCP	Syn Sent	10.0.0.3	49710	10.0.0.26	445	10/5/2025 10:19:39 am	mssecsvc2.0
System	4	TCP	Listen	0.0.0.0	445	0.0.0.0	0	9/4/2025 10:09:57 am	System
svchost.exe	788	TCPv6	Listen	::	135	::	0	10/4/2025 1:09:50 am	RpcEptMapper

It seems that there is an attempt from wannacry to spread through SMB (port 445) throughout the local network. So wannacry has imbued WORM capabilities too.

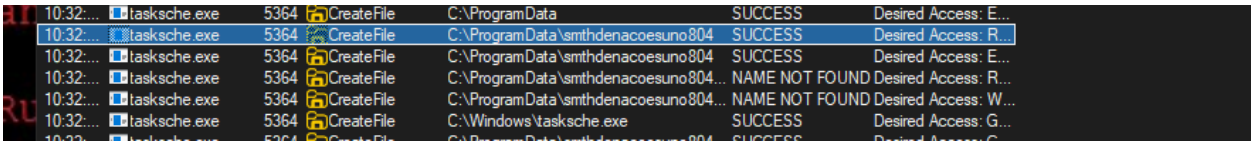
taskhsvc.exe	4480	TCP	Listen	127.0.0.1	9050	0.0.0.0	0	10/5/2025 10:20:25 am	taskhsvc.exe
lsass.exe	596	TCP	Listen	0.0.0.0	49664	0.0.0.0	0	10/4/2025 1:09:50 am	lsass.exe
wininit.exe	476	TCP	Listen	0.0.0.0	49665	0.0.0.0	0	10/4/2025 1:09:50 am	wininit.exe
svchost.exe	332	TCP	Listen	0.0.0.0	49666	0.0.0.0	0	10/4/2025 1:09:51 am	EventLog
svchost.exe	976	TCP	Listen	0.0.0.0	49667	0.0.0.0	0	10/4/2025 1:09:51 am	Schedule
spoolsv.exe	1820	TCP	Listen	0.0.0.0	49668	0.0.0.0	0	9/4/2025 10:09:55 am	Spooler
services.exe	588	TCP	Listen	0.0.0.0	49669	0.0.0.0	0	9/4/2025 10:09:57 am	services.exe
svchost.exe	1176	TCP	Listen	0.0.0.0	49670	0.0.0.0	0	9/4/2025 10:09:58 am	PolicyAgent
taskhsvc.exe	4480	TCP	Established	127.0.0.1	50491	127.0.0.1	50492	10/5/2025 10:20:25 am	taskhsvc.exe
taskhsvc.exe	4480	TCP	Established	127.0.0.1	50492	127.0.0.1	50491	10/5/2025 10:20:25 am	taskhsvc.exe
System	4	TCP	Listen	0.0.0.0	445	0.0.0.0	0	9/4/2025 10:09:57 am	System

At the same time from TCP view, we can see a process, “tasksche” generated upon detonating wannacry too.

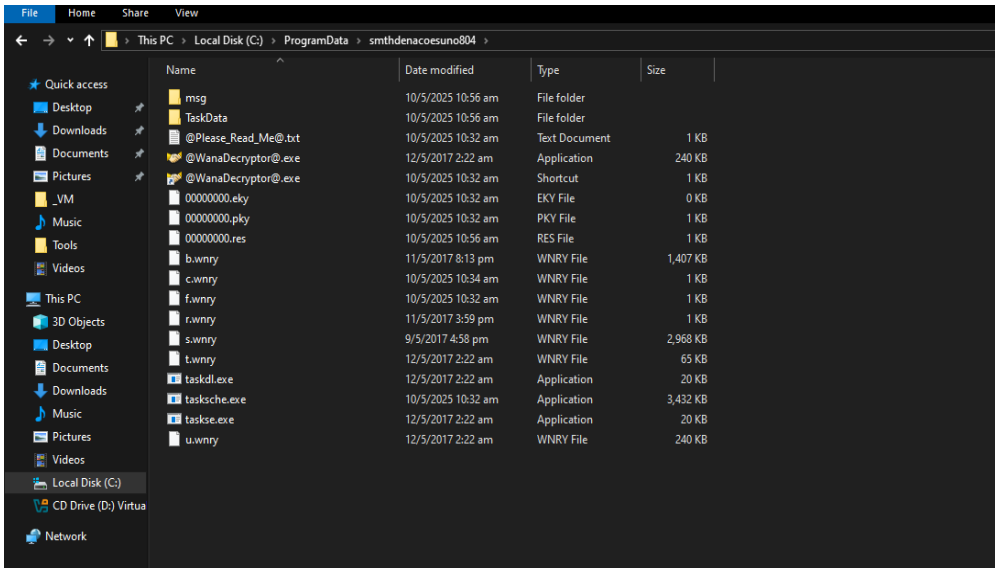
Using procmon

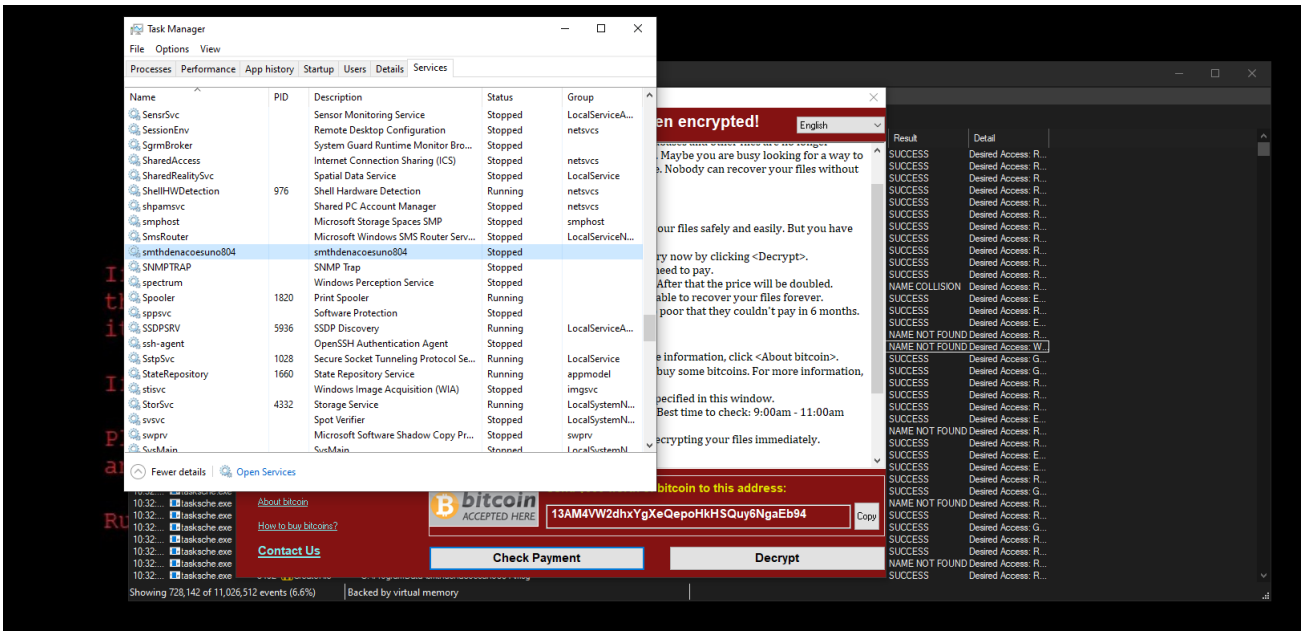


Using procmon, I was able to further confirm that the process tasksche.exe was unpacked by wannacry.



Upon filtering for tasksche.exe from procmon , I was able to detect the creation of a folder in my C drive by tasksche.exe. The content of it is most likely the unpacked files for the ransomware by wannacry.





Upon inspecting, Task Manager under Services, a service of the same name was also found. I can further deduce this to be some sort of persistence mechanism.

Assembly analysis using cutter

```
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]
```

The URL callback domain used by wannacry was moved into the esi register for use later.

```
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
```

We can see a bunch of dword variable being moved into eax and push into the stack to be used for the “InternetOpenA” API call.

```
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] ;
```

Here , we can see the weird URL that is stored in esi, is pushed into the stack to be used with “InternetOpenUrlA” api

```
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
```

The return value of “InternetOpenUrlA” is then stored in edi. If it is 1 , the zero flag will be set to 0, then the program would jump to the location in memory in the right branch in the picture, which cleans up the argument in the stack, and returns out of the program. If the zero flag is evaluated to be 0, then it would jump to the location memory in the left, which have a function call before returning out of the program.

When the zero flag is set to 0, the program will not jump to the specified memory address, it would then jump to the specified memory address on the left side of the image.

```
[0x00408090]
fcn.00408090();
; var const char *var_3ch @ stack - 0x3c
; var const char *var_38h @ stack - 0x38
; var int32_t var_34h @ stack - 0x34
; var int32_t var_30h @ stack - 0x30
; var int32_t var_2ch @ stack - 0x2c
; var const char *lpServiceStartTable @ stack - 0x28
; var int32_t var_24h @ stack - 0x24
; var int32_t var_20h @ stack - 0x20
; var int32_t var_1ch @ stack - 0x1c
0x00408090      sub     esp, 0x10
0x00408093      push    0x104       ; 260 ; DWORD nSize
0x00408098      push    data.0070f760 ; 0x70f760 ; LPSTR lpFilename
0x0040809d      push    0           ; HMODULE hModule
0x0040809f      call    dword [GetModuleFileNameA] ; 0x40a06c ; DWORD GetModuleFileNameA(HMODULE ...
0x004080a5      call    dword [__p___argc] ; 0x40a12c
0x004080ab      cmp     dword [eax], 2
0x004080ae      jge     0x4080b9
```

```
[0x004080b0]
0x004080b0      call    fcn.00407f20 ; fcn.00407f20
0x004080b5      add     esp, 0x10
0x004080b8      ret
```

```
[0x004080b9]
0x004080b9      push    edi
0x004080ba      push    0xf003f     ; '?' ; DWORD dwDesiredAccess
0x004080bf      push    0           ; LPCSTR lpDatabaseName
0x004080c1      push    0           ; LPCSTR lpMachineName
0x004080c3      call    dword [OpenSCManagerA] ; 0x40a010 ; SC_HANDLE OpenSCManagerA(LPCSTR lpMac...
0x004080c9      mov     edi, eax
0x004080cb      test    edi, edi
0x004080cd      je      0x408101
```

```
[0x004080cf]
0x004080cf      push    ebx
0x004080d0      push    esi
0x004080d1      push    0xf01ff     ; DWORD dwDesiredAccess
0x004080d6      push    str.mssecsvc2.0 ; 0x4312fc ; LPCSTR lpServiceName
0x004080db      push    edi         ; SC_HANDLE hSCManager
0x004080dc      call    dword [OpenServiceA] ; 0x40a028 ; SC_HANDLE OpenServiceA(SC_HANDLE hSCMan...
0x004080e2      mov     ebx, dword [CloseServiceHandle] ; 0x40a018
0x004080e8      mov     esi, eax
0x004080ea      test    esi, esi
```

And the program will execute this function call, which leads to all the encryption of the files in the filesystem.