



WANNACRY RANSOMWARE / CRYPTOWORM

# PRACTICAL MALWARE ANALYSIS & TRIAGE REPORT

PMAT Report / 2022-10-22 / straysheep-dev / v1.0

---

---

## Table of Contents

<b>Executive Summary</b>	<b>3</b>
EFFECTIVE RISK	3
SCOPE OF INFECTION	3
<b>Technical Summary</b>	<b>4</b>
<b>Dynamic Analysis</b>	<b>5</b>
INITIAL INFECTION	5
POST INFECTION	7
<b>Static Analysis</b>	<b>10</b>
BINARY COMPOSITION	11
advapi32.dll	11
iphlpapi.dll	11
kernel32.dll	12
msvcrt.dll	12
wininet.dll	12
ws2_32.dll	13
DECOMPILED MAIN FUNCTION	14
BINWALK	16
DEBUGGING	17
<b>IOC &amp; YARA Rules</b>	<b>18</b>
<b>Appendices</b>	<b>20</b>

## Executive Summary

SHA256SUM=24D004A104D4D54034DBCFFC2A4B19A11F39008A575AA614EA04703480B1022C

WannaCry is a 32-bit [Windows executable](#) that encrypts all user data on the machine when executed. It then attempts to [infect other machines on both the local network and public internet](#), finally displaying a prompt demanding payment in the form of bitcoin to recover from the infection.

---

### EFFECTIVE RISK

*At the time of writing it's well known that the domain contacted as part of the killswitch routine has been registered as a sinkhole in an effort to slow the spread of infection, and that MS17-010, Eternal Blue, is used to spread the infection.*

- If the killswitch domain serves a valid HTTP response, infection is mitigated.
- If Windows Defender is enabled and updated on the endpoint, infection is mitigated.
- If the endpoint lacks privileges to write system files & services, infection is mitigated.

### SCOPE OF INFECTION

[Administrative accounts and endpoints with higher privileges](#) are the point of infection.

Once infection occurs, non-administrative accounts and other un-patched Windows machines on the network can become infected.

---

## Technical Summary

1. An initial **DNS query** + **HTTP request** is made to:

```
[hxxp://] www [dot] iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea [dot] com /
```

If the request succeeds with a valid response, it terminates and the host is not infected.

If the HTTP request fails, it continues.

2. The *first* **system service** is created for persistence and propagation:

DisplayName: **Microsoft Security Center (2.0) Service**

Path: **HKLM\System\CurrentControlSet\Services\mssecsvc2.0**

ImagePath: **C:\Path\To\wannacry.exe -m security**

If the **mssecsvc2.0** service fails to be created, it exits.

If **mssecsvc2.0** is created, it continues.

3. **C:\Windows\tasksche.exe** is created and executed with **/i**.
4. **tasksche.exe** unpacks utilities to **C:\ProgramData\<random-string>**

5. The *second* **system service** is created to kick off the infection:

DisplayName: **<random-string>**

Path: **HKLM\System\CurrentControlSet\Services\<random-string>**

ImagePath: **cmd.exe /C "C:\ProgramData\<random-string>\tasksche.exe"**

6. From within **C:\ProgramData\<random-string>** the following is executed:
  - a. **attrib +h .** to hide the directory
  - b. **icaccls.exe . /grant Everyone:F /T /C /Q**

- c. `taskdl.exe` is executed
    - i. Note: `taskdl.exe`'s OriginalFilename: `cliconfg.exe`
  - d. `C:\Windows\System32\cmd.exe /c 269581666413478.bat`
  - e. Then `269581666413478.bat` executes:
    - i. `cscript.exe //nologo m.vbs`
  - f. `@WanaDecryptor@.exe` co then executes:
    - i. `.\Taskdata\Tor\taskhsvc.exe`
    - ii. Note: `@WanaDecryptor@.exe`'s OriginalFilename: `LODCTR.EXE`
  - g. `Cmd.exe /c start /b @WanaDecryptor@.exe` vs is executed
7. Next, `cmd.exe /c vssadmin delete shadows /all /quiet & wmic shadowcopy delete & bcdedit /set {default} bootstatuspolicy ignoreallfailures & bcdedit /set {default} recoveryenabled no & wbadmin delete catalog -quiet` is executed
8. Lastly `cmd.exe /c` is once again used to execute `reg add HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v "<random-string>" /t REG_SZ /d "\"C:\ProgramData\<random-string>\tasksche.exe\""` /f

From this point on, the following two commands can be seen running in a loop:

1. `taskdl.exe`
  2. `taskse.exe C:\ProgramData\<random-string>\@WanaDecryptor@.exe`
- `Autoruns64.exe` also shows `<random-string>` exists under `HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Run`
  - The main binary attempts to spread over TCP port 445 using `MS17-010`, `Eternal Blue`
  - Files are encrypted and renamed with an extension of `.WNCRY`
  - A prompt is repeatedly displayed as the foreground window, demanding payment

## Dynamic Analysis

WannaCry's behavior can be broken into two parts, **initial infection**, and **post infection**.

### INITIAL INFECTION

Attempts an HTTP GET request to:

```
[hxxp://] www [dot] iuqerfsodp9ifjaposdfjhgosurijfaewrwergrwa [dot] com /
```

TCP	54 80 → 49826	[ACK] Seq=249 Ack=114 Win=64128 Len=0
DNS	109	Standard query 0xe71b A www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergrwa.com
DNS	125	Standard query response 0xe71b A www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergrwa.com
TCP	66 49827 → 80	[SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
TCP	66 80 → 49827	[SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
TCP	60 49827 → 80	[ACK] Seq=1 Ack=1 Win=262144 Len=0
HTTP	154	GET / HTTP/1.1
TCP	54 80 → 49827	[ACK] Seq=1 Ack=101 Win=64256 Len=0

If a response is returned, it stops and exits. If the **request fails**, it continues execution.

A system service named Microsoft Security Center (2.0) Service (**mssecsvc2.0**) is created.

Microsoft Security Center (2.0) Service	Running	Automatic	Local System
-----------------------------------------	---------	-----------	--------------

Path: **HKLM\System\CurrentControlSet\Services\mssecsvc2.0**

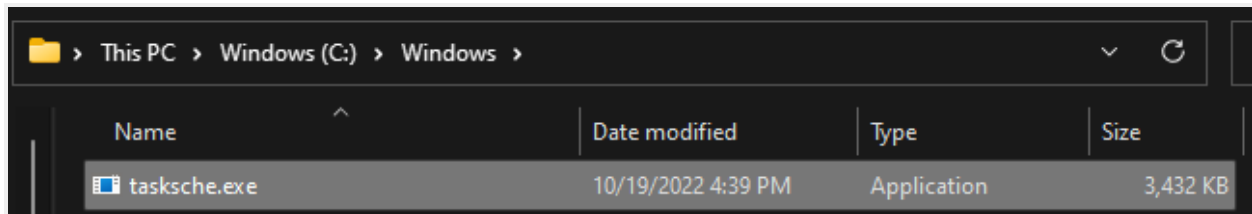
➔ ImagePath: **C:\Path\To\wannacry.exe -m security**

If the service cannot be created, it **exits**.

Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\mssecsvc2.0		
MsBridge	Name	Type
MSDTC	(Default)	REG_SZ
MSDTC Bridge 4.0.0.0	DisplayName	REG_SZ
Msfs	ErrorControl	REG_DWORD
msgpiowin32	FailureActions	REG_BINARY
mshidkmdf	ImagePath	REG_EXPAND_SZ
mshidumdf	ObjectName	REG_SZ
msisadrv	Start	REG_DWORD
MSISCSI	Type	REG_DWORD
msiserver	WOW64	REG_DWORD
MsKeyboardFilter		
MSKSSRV		
	Data	
	(value not set)	
	Microsoft Security Center (2.0) Service	
	0x00000001 (1)	
	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	
	C:\Users\User\Documents\PMAT-labs-main\PMAT-labs-main\labs\4-1.Bossfight-wannacry.exe\Ransomware.wannacry.exe -m security	
	LocalSystem	
	0x00000002 (2)	
	0x00000010 (16)	
	0x0000014c (332)	

*mssecsvc2.0 will run the main wannacry binary with the **-m security** argument*

C:\Windows\tasksche.exe is created.

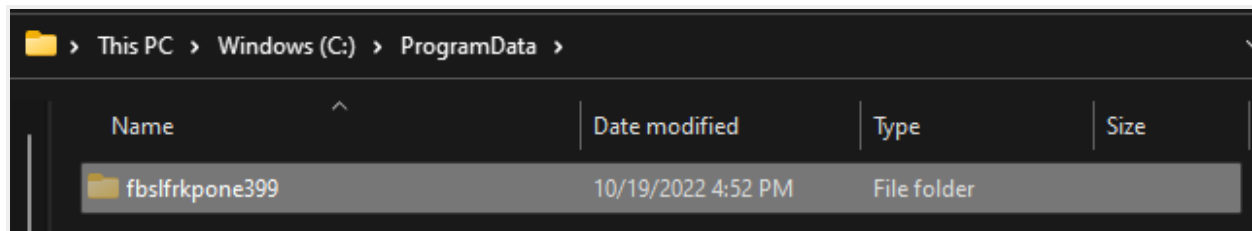


A screenshot of a Windows Explorer window showing the file 'tasksche.exe' in the 'C:\Windows' directory. The file is an application, 3,432 KB in size, and was last modified on 10/19/2022 at 4:39 PM.

Name	Date modified	Type	Size
tasksche.exe	10/19/2022 4:39 PM	Application	3,432 KB

C:\Windows\tasksche.exe is executed and unpacks utilities to a hidden directory:

➤ Path: C:\ProgramData\<random-string>\



A screenshot of a Windows Explorer window showing a hidden folder named 'fbslfrkpone399' in the 'C:\ProgramData' directory. The folder is a file folder, created on 10/19/2022 at 4:52 PM.

Name	Date modified	Type	Size
fbslfrkpone399	10/19/2022 4:52 PM	File folder	

An additional service named after the <random-string> is created.

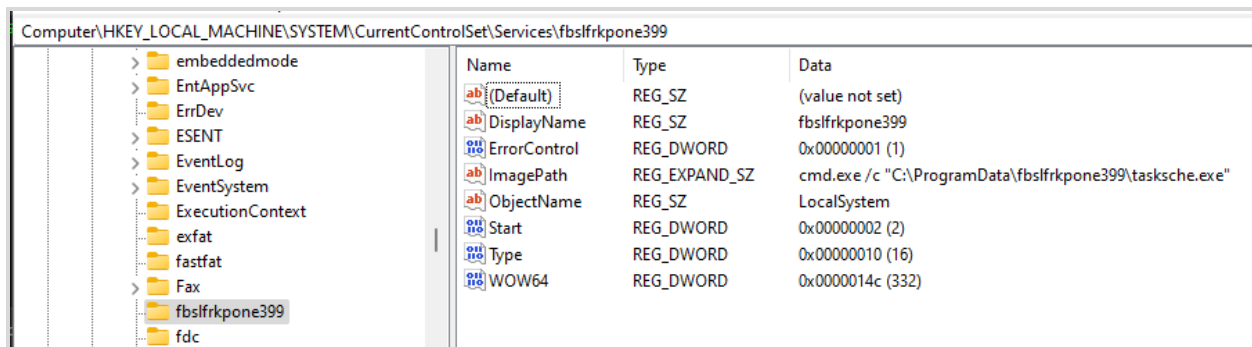


A screenshot of the Windows Services console showing a service named 'fbslfrkpone399'. The service is set to 'Starting' type, 'Automatic' startup, and runs as 'Local System'.

Name	Status	Startup Type	Log On As
fbslfrkpone399	Starting	Automatic	Local System

Path: HKLM\System\CurrentControlSet\Services\<random-string>

➤ ImagePath: cmd.exe /c "C:\ProgramData\<random-string>\tasksche.exe"



A screenshot of the Windows Registry Editor showing the configuration for the service 'fbslfrkpone399' under the path 'Computer\HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\fbslfrkpone399'.

Name	Type	Data
(Default)	REG_SZ	(value not set)
DisplayName	REG_SZ	fbslfrkpone399
ErrorControl	REG_DWORD	0x00000001 (1)
ImagePath	REG_EXPAND_SZ	cmd.exe /c "C:\ProgramData\fbslfrkpone399\tasksche.exe"
ObjectName	REG_SZ	LocalSystem
Start	REG_DWORD	0x00000002 (2)
Type	REG_DWORD	0x00000010 (16)
WOW64	REG_DWORD	0x0000014c (332)

```

TimeCreated : 10/18/2022 3:04:24 PM
ProviderName : Microsoft-Windows-Sysmon
Id : 11
Message : File created:
          RuleName: EXE
          UtcTime: 2022-10-18 22:04:24.438
          ProcessGuid: {f75684c1-22e8-634f-2b01-000000001600}
          ProcessId: 2072
          Image: C:\WINDOWS\tasksche.exe
          TargetFilename: C:\ProgramData\fbslfrkpone399\tasksche.exe
          CreationUtcTime: 2022-10-18 22:04:24.438
          User: WINDEV2209EVAL\User

TimeCreated : 10/18/2022 3:04:24 PM
ProviderName : Microsoft-Windows-Sysmon
Id : 13
Message : Registry value set:
          RuleName: T1031,T1050
          EventType: SetValue
          UtcTime: 2022-10-18 22:04:24.454
          ProcessGuid: {f75684c1-627f-634e-0c00-000000001600}
          ProcessId: 732
          Image: C:\Windows\system32\services.exe
          TargetObject: HKLM\System\CurrentControlSet\Services\fbslfrkpone399\Start
          Details: DWORD (0x00000002)
          User: NT AUTHORITY\SYSTEM

TimeCreated : 10/18/2022 3:04:24 PM
ProviderName : Microsoft-Windows-Sysmon
Id : 13
Message : Registry value set:
          RuleName: T1031,T1050
          EventType: SetValue
          UtcTime: 2022-10-18 22:04:24.454
          ProcessGuid: {f75684c1-627f-634e-0c00-000000001600}
          ProcessId: 732
          Image: C:\Windows\system32\services.exe
          TargetObject: HKLM\System\CurrentControlSet\Services\fbslfrkpone399\ImagePath
          Details: cmd.exe /c "C:\ProgramData\fbslfrkpone399\tasksche.exe"
          User: NT AUTHORITY\SYSTEM

```

Sysmon followed the `<random-string>` service binary being unpacked and starting.

This PC > Windows (C:) > ProgramData > fbslfrkpone399				
Name	Date modified	Type	Size	
msg	10/18/2022 2:54 PM	File folder		
TaskData	10/18/2022 2:55 PM	File folder		
@Please_Read_Me@.txt	10/18/2022 2:54 PM	Text Document	1 KB	
@WanaDecryptor@.exe	5/12/2017 2:22 AM	Application	240 KB	
@WanaDecryptor@.exe	10/18/2022 2:54 PM	Shortcut	1 KB	
00000000.eky	10/18/2022 2:54 PM	EKY File	0 KB	
00000000.pky	10/18/2022 2:54 PM	PKY File	1 KB	
00000000.res	10/18/2022 2:57 PM	Compiled Resourc...	1 KB	
b.wnry	5/11/2017 8:13 PM	WNRY File	1,407 KB	
c.wnry	10/18/2022 2:55 PM	WNRY File	1 KB	
f.wnry	10/18/2022 2:54 PM	WNRY File	1 KB	

Screenshot of the utilities unpacked to the ProgramData directory.



SMB traffic will flood the network, attempting to reach numerous private and public IP's over TCP port 445. If a connection can be established, the `Eternal Blue` exploit, `MS17-010` is sent as a payload to spread the infection.

tcp.port==445									
No.	Time	Source	Destination	Protocol	Length	Info			
8191	192.328353689	10.45.45.6	117.34.169.151	TCP	66	54887 → 445	[SYN]	Seq=0	Win=64240
8197	192.396625755	10.45.45.6	158.25.109.7	TCP	66	54888 → 445	[SYN]	Seq=0	Win=64240
8198	192.428198446	10.45.45.6	170.200.30.244	TCP	66	54889 → 445	[SYN]	Seq=0	Win=64240
8199	192.432574567	10.45.45.6	19.192.229.216	TCP	66	54890 → 445	[SYN]	Seq=0	Win=64240
8200	192.437703695	10.45.45.6	29.228.142.219	TCP	66	54891 → 445	[SYN]	Seq=0	Win=64240
8210	192.531161961	10.45.45.6	93.40.215.179	TCP	66	[TCP Retransmission] [TCP Port numbers reused] 54874 → 445 [SYN] Seq=0			
8211	192.596531383	10.45.45.6	175.215.72.187	TCP	66	54894 → 445	[SYN]	Seq=0	Win=64240
8212	192.596531834	10.45.45.6	49.208.202.65	TCP	66	54892 → 445	[SYN]	Seq=0	Win=64240
8213	192.596592107	10.45.45.6	188.10.134.95	TCP	66	54893 → 445	[SYN]	Seq=0	Win=64240
8214	192.641309092	10.45.45.6	17.84.38.41	TCP	66	54895 → 445	[SYN]	Seq=0	Win=64240
8226	192.688211876	10.45.45.6	70.101.220.46	TCP	66	54896 → 445	[SYN]	Seq=0	Win=64240
8239	192.812828470	10.45.45.6	150.115.184.124	TCP	66	54897 → 445	[SYN]	Seq=0	Win=64240
8240	192.812828961	10.45.45.6	147.24.19.188	TCP	66	54898 → 445	[SYN]	Seq=0	Win=64240
8241	192.859804762	10.45.45.6	79.24.22.19	TCP	66	[TCP Retransmission] [TCP Port numbers reused] 54880 → 445 [SYN] Seq=0			
8242	192.859805192	10.45.45.6	91.29.170.79	TCP	66	[TCP Retransmission] [TCP Port numbers reused] 54881 → 445 [SYN] Seq=0			
8243	192.875889504	10.45.45.6	154.68.22.120	TCP	66	54899 → 445	[SYN]	Seq=0	Win=64240
8244	192.984418511	10.45.45.6	10.28.161.217	TCP	66	54901 → 445	[SYN]	Seq=0	Win=64240
8245	192.984419122	10.45.45.6	93.97.187.234	TCP	66	54900 → 445	[SYN]	Seq=0	Win=64240
8246	192.984478654	10.45.45.6	61.252.79.137	TCP	66	54902 → 445	[SYN]	Seq=0	Win=64240
8247	193.093811869	10.45.45.6	221.96.219.189	TCP	66	54903 → 445	[SYN]	Seq=0	Win=64240
8248	193.141086890	10.45.45.6	90.78.246.220	TCP	66	[TCP Retransmission] [TCP Port numbers reused] 54884 → 445 [SYN] Seq=0			
8291	193.251555915	10.45.45.6	62.127.92.137	TCP	66	54904 → 445	[SYN]	Seq=0	Win=64240
8292	193.251556757	10.45.45.6	78.130.208.98	TCP	66	54905 → 445	[SYN]	Seq=0	Win=64240
8293	193.296930563	10.45.45.6	39.209.239.202	TCP	66	54906 → 445	[SYN]	Seq=0	Win=64240
8294	193.296930974	10.45.45.6	161.47.182.13	TCP	66	54907 → 445	[SYN]	Seq=0	Win=64240
8296	193.454905330	10.45.45.6	175.252.194.84	TCP	66	54908 → 445	[SYN]	Seq=0	Win=64240
Frame 597: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface eth0, id 0 Ethernet II, Src: VMware_b8:1f:3d (00:0c:29:b8:1f:3d), Dst: VMware_96:29:9e (00:0c:29:96:29:9e) Internet Protocol Version 4, Src: 10.45.45.6, Dst: 219.11.195.79 Transmission Control Protocol, Src Port: 54239, Dst Port: 445, Seq: 0, Len: 0									

*Traffic leaving an infected host targeting TCP destination port 445*

At just over 56 minutes the infected machine attempted to reach over 83,500 hosts.

Note that even if Windows Defender is able to stop the file encryption routine on the host, **this propagation mechanism still runs successfully in the background** as a separate process of the main `wannacry.exe` binary with the `-m security` argument.

It's assumed target IP's are generated at random based on the string `%d.%d.%d.%d`.

Tor connections are also made to hosts on port 9001 and 9101:

Ethernet - 1	IPv4 - 3	IPv6	TCP - 3	UDP										
Address A	Port A	Address B	Port B	Packets	Bytes	Stream ID	Total Packets	Percent Filtered	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A		
10.45.45.6	57875	62.210.92.11	9001	5	330 bytes	7863	5	100.00%	5	330 bytes	0	0 bytes	5	0 bytes
10.45.45.6	59883	78.24.75.53	9001	5	330 bytes	9125	5	100.00%	5	330 bytes	0	0 bytes	5	0 bytes
10.45.45.6	62064	128.31.0.39	9101	5	330 bytes	15212	5	100.00%	5	330 bytes	0	0 bytes	5	0 bytes

While a local proxy is started and *listening* on 127.0.0.1:9050:

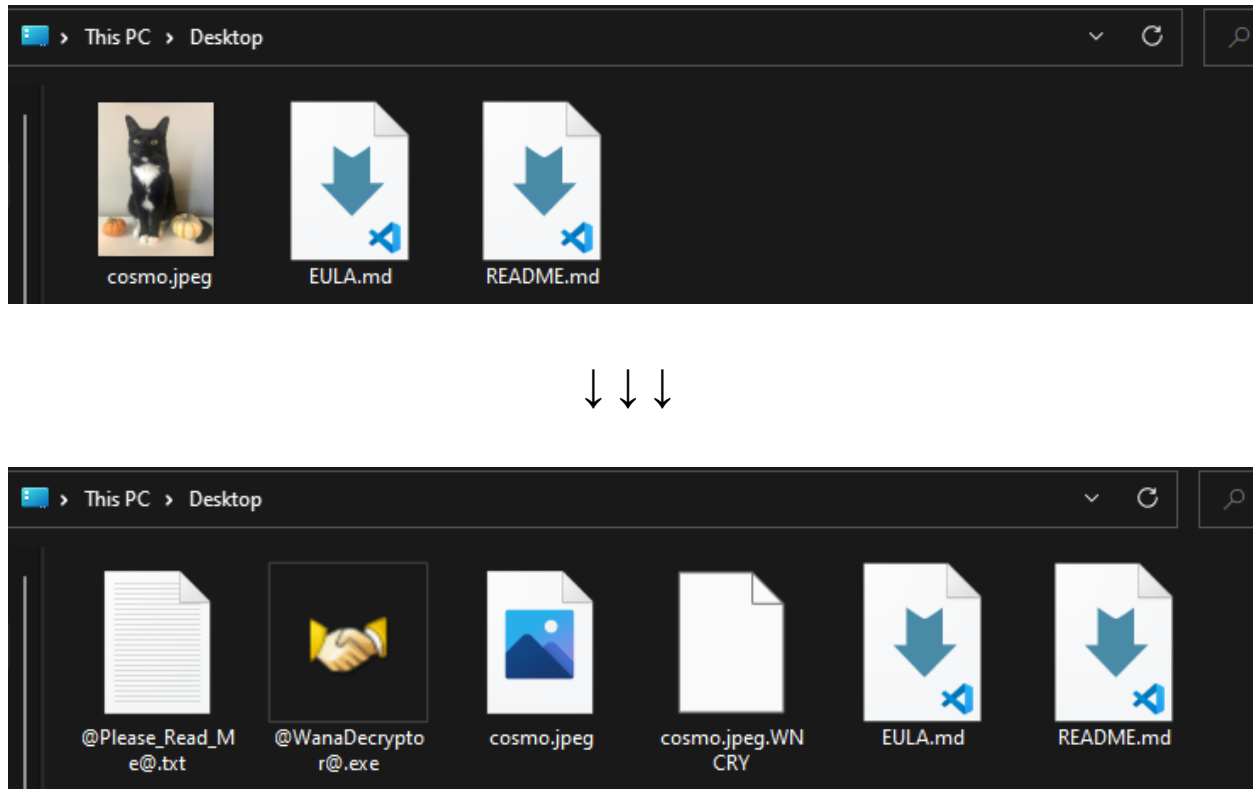
Process Name	Process ID	Protocol	State	Local Address	Local Port	Remote Address	Remote Port	Create Time	Module Name
taskhsvc.exe	9264	TCP	Listen	127.0.0.1	9050	0.0.0.0	0	10/19/2022 5:26:46 PM	taskhsvc.exe
taskhsvc.exe	9264	TCP	Established	127.0.0.1	9050	127.0.0.1	49842	10/19/2022 5:27:20 PM	taskhsvc.exe
[Time Wait]		TCP	Time Wait	127.0.0.1	49841	127.0.0.1	9050		
@WanaDecryptor@.exe	8744	TCP	Established	127.0.0.1	49842	127.0.0.1	9050	10/19/2022 5:27:20 PM	@WanaDecryptor@.exe

Tor is also included within the set of utilities under ProgramData:

This PC > Windows (C:) > ProgramData > fbslfrkhone399 > TaskData > Tor					
Name	Date modified	Type	Size		
libeay32.dll	12/31/1999 11:00 PM	Application exten...	3,123 KB		
libevent_core-2-0-5.dll	12/31/1999 11:00 PM	Application exten...	408 KB		
libevent_extra-2-0-5.dll	12/31/1999 11:00 PM	Application exten...	402 KB		
libevent-2-0-5.dll	12/31/1999 11:00 PM	Application exten...	703 KB		
libgcc_s_sjlj-1.dll	12/31/1999 11:00 PM	Application exten...	511 KB		
libssp-0.dll	12/31/1999 11:00 PM	Application exten...	91 KB		
ssleay32.dll	12/31/1999 11:00 PM	Application exten...	695 KB		
taskhsvc.exe	12/31/1999 11:00 PM	Application	3,026 KB		
tor.exe	12/31/1999 11:00 PM	Application	3,026 KB		
zlib1.dll	12/31/1999 11:00 PM	Application exten...	105 KB		

*Tor binaries and libraries*

The visual indicators are the most obvious, starting with all user files being encrypted:



This is followed by the wallpaper and payment prompt:



All user files available to the host have the `.WNCRY` extension appended at this point.

## POST INFECTION

WannaCry does not encrypt file extensions it does not know about.

- This was first observed when `.yar` YARA rule files were not encrypted after infection
- This was tested on various valid and invalid file extensions, which proved to survive
- During static analysis, strings were discovered that match common file extensions
- The entire list of target file extensions for encryption is included in the Appendices
- It's assumed this is how WannaCry determines what to encrypt without breaking the operating system

WannaCry may flood (depending on your configuration), but does not encrypt logs.

- A sample of possible hunting queries is included in the IOC & YARA section
- A list of suspicious log entries is available in the Appendices

Review of the logs after infection revealed a few additional noteworthy points.

- WannaCry runs `vssadmin delete shadows /all`, a common ransomware TTP
- `C:\Windows\tasksche.exe /i` is run, noting the `/i` argument
- A file named `C:\ProgramData\<random-string>\203341666389597.bat` is executed
- `C:\ProgramData\<random-string>\m.vbs` is run with `cscript.exe //nologo m.vbs`

## Static Analysis

SHA256SUM=24D004A104D4D54034DBCFFC2A4B19A11F39008A575AA614EA04703480B1022C

[VirusTotal Score \(67/72\)](#)

Md5 db349b97c37d22f5ea1d1841e3c89eb4

Sha1 e889544aff85ffaf8b0d0da705105dee7c97fe26

Sha256 24d004a104d4d54034dbcffc2a4b19a11f39008a575aa614ea04703480b1022c

File-size 3723264 bytes

Entropy 7.964

Cpu 32-bit

Original Filename: lhdfrgui.exe

Interesting strings extracted with `floss.exe`:

```
\\%s\IPC$
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
WINDOWS
tasksche.exe
CloseHandle
WriteFile
CreateFileA
CreateProcessA
http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com
!This program cannot be run in DOS mode.
```

*Specific strings and signatures are provided under IOC & YARA Rules and the Appendices*

## BINARY COMPOSITION

`pestudio.exe` shows multiple executables embedded which we can confirm with `binwalk`:

		DECIMAL	HEXADECIMAL	DESCRIPTION
-	-	-	-	-
-	0x00310000	-	-	-
-	-	0	0x0	Microsoft executable, portable (PE)
-	-	45088	0xB020	Microsoft executable, portable (PE)
-	-	61568	0xF080	Microsoft executable, portable (PE)
0x0000B020 (executable, 5263716 bytes)	0x000320A4 (executable, 35...	204964	0x320A4	Microsoft executable, portable (PE)
0x0000F080 (executable, 5297524 bytes)	-	257775	0x3EEEF	Copyright string: "Copyright 1995-1998 t
0x0002CB74 (executable, 159744 bytes)	-	258296	0x3F0F8	CRC32 polynomial table, little endian
		259332	0x3F504	Copyright string: "Copyright 1998 Gilles
		270740	0x42104	Zip archive data, extracted at least v2

List of interesting imported functions and their use, picked up by `pestudio.exe`:

### advapi32.dll

*11 Total Imported Functions*

[ChangeServiceConfig2A](#) | Changes the optional configuration parameters of a service

[CreateServiceA](#) | Creates a service object, adds it to a specified service control manager database

[CryptAcquireContextA](#) | Used to acquire a handle to a particular key container

[CryptGenRandom](#) | Fills a buffer with cryptographically random bytes

[StartServiceCtrlDispatcherA](#) | Connects the main thread of a service process to the service control manager

### iphlpapi.dll

*2 Total Imported Functions*

[GetAdaptersInfo](#) | retrieves adapter information for the local computer

## kernel32.dll

*32 Total Imported Functions*

[GetCurrentThread](#) | Retrieves a pseudo handle for the calling thread

[GetCurrentThreadId](#) | Retrieves the thread identifier of the calling thread

[MoveFileExA](#) | Moves an existing file or directory, including its children

[QueryPerformanceFrequency](#) | Retrieves the frequency of the performance counter. The frequency of the performance counter is fixed at system boot and is consistent across all processors

## msvcrt.dll

*28 Total Imported Functions*

[rand](#) | Generates a pseudorandom number

[srand](#) | Sets the starting seed value for the pseudorandom number generator used by the rand function

## wininet.dll

*3 Total Imported Functions*

[InternetCloseHandle](#) | Closes a single Internet handle

[InternetOpenA](#) | Initializes an application's use of the WinINet functions

[InternetOpenUrlA](#) | Opens a resource specified by a complete FTP or HTTP URL

## **ws2\_32.dll**

### *13 Total Imported Functions*

- 10 ([ioctlsocket](#)) | controls the I/O mode of a socket
- 11 ([inet\\_addr](#)) | converts a string containing an IPv4 dotted-decimal address into a proper address for the IN\_ADDR structure
- 115 ([WSAStartup](#)) | Initiates use of the Winsock DLL by a process
- 12 ([inet\\_ntoa](#)) | Converts an (IPv4) Internet network address into an ASCII string
- 14 ([ntohl](#)) | Converts a u\_long from TCP/IP network order to host byte order
- 16 ([recv](#)) | Receives data from a connected socket or a bound connectionless socket
- 18 ([select](#)) | Determines the status of one or more sockets to perform synchronous I/O
- 19 ([send](#)) | Sends data on a connected socket
- 23 ([socket](#)) | Creates a socket that is bound to a specific transport service provider
- 3 ([closesocket](#)) | Closes an existing socket
- 4 ([connect](#)) | Establishes a connection to a specified socket
- 8 ([htonl](#)) | Converts a u\_long from host to TCP/IP network byte order
- 9 ([htons](#)) | Converts a u\_short from host to TCP/IP network byte order



## DECOMPILED MAIN FUNCTION

The `main()` function decompiled in Cutter shows the killswitch using `InternetOpenUrlA` + `InternetCloseHandle` to obtain the exit code of its HTTP request.

```
Decompiler (main)
#include <stdint.h>

int32_t main (void) {
    int32_t var_14h;
    int32_t var_8h;
    int32_t var_41h;
    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.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.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, eax, ax, al);
    ecx = &var_14h;
    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;
}
```

Jumping back to the graph view, we can see the `test edi, edi` where it will either terminate and exit (if it receives a valid response from its HTTP request) or continue execution into `fcn.00408090` if the GET request fails.



*Killswitch at the bottom of the main() function*

## BINWALK

`binwalk` reveals multiple Windows executables within the main binary:

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	Microsoft executable, portable (PE)
45088	0xB020	Microsoft executable, portable (PE)
61568	0xF080	Microsoft executable, portable (PE)
204964	0x320A4	Microsoft executable, portable (PE)
257775	0x3EEEF	Copyright string: "Copyright 1995-1998 Mark Adler "
258296	0x3F0F8	CRC32 polynomial table, little endian
259332	0x3F504	Copyright string: "Copyright 1998 Gilles Vollant "
270740	0x42194	Zip archive data, encrypted at least v2.0 to extract, compressed size
284940	0x4590C	Zip archive data, encrypted at least v2.0 to extract, compressed size
285153	0x459E1	Zip archive data, encrypted at least v2.0 to extract, compressed size
294607	0x47ECF	Zip archive data, encrypted at least v2.0 to extract, compressed size
305712	0x4AA30	Zip archive data, encrypted at least v2.0 to extract, compressed size
317407	0x4D7DF	Zip archive data, encrypted at least v2.0 to extract, compressed size

There also are a number of zip archives embedded. Extracting one of them we can see what files it contains, however the contents are password protected:

36973	2010-11-20 04:16	msg/m_english.wnry
37580	2010-11-20 04:16	msg/m_filipino.wnry
38377	2010-11-20 04:16	msg/m_finnish.wnry
38437	2010-11-20 04:16	msg/m_french.wnry
37181	2010-11-20 04:16	msg/m_german.wnry
49044	2010-11-20 04:16	msg/m_greek.wnry
37196	2010-11-20 04:16	msg/m_indonesian.wnry
36883	2010-11-20 04:16	msg/m_italian.wnry
81844	2010-11-20 04:16	msg/m_japanese.wnry
91501	2010-11-20 04:16	msg/m_korean.wnry
41169	2010-11-20 04:16	msg/m_latvian.wnry
37577	2010-11-20 04:16	msg/m_norwegian.wnry
39896	2010-11-20 04:16	msg/m_polish.wnry
37917	2010-11-20 04:16	msg/m_portuguese.wnry
52161	2010-11-20 04:16	msg/m_romanian.wnry
47108	2010-11-20 04:16	msg/m_russian.wnry
41391	2010-11-20 04:16	msg/m_slovak.wnry
37381	2010-11-20 04:16	msg/m_spanish.wnry
38483	2010-11-20 04:16	msg/m_swedish.wnry
42582	2010-11-20 04:16	msg/m_turkish.wnry
93778	2010-11-20 04:16	msg/m_vietnamese.wnry
864	2017-05-11 15:59	r.wnry
3038286	2017-05-09 16:58	s.wnry
65816	2017-05-12 02:22	t.wnry
20480	2017-05-12 02:22	taskdl.exe
20480	2017-05-12 02:22	taskse.exe
245760	2017-05-12 02:22	u.wnry
-----	-----	-----
6162177		36 files

## DEBUGGING

Recursively extracting the embedded binaries doesn't immediately lead to `tasksche.exe`.

The binary at `320A4` contains the main ransomware routine and attempts a tor connection, but has no services, utilities, propagation, or persistence mechanisms.

Rather than bruteforcing the zip file passwords and continuing to extract binaries, we turn to running WannaCry in a debugger. Shortly after the killswitch routine fails, it writes `HKLM\System\CurrentControlSet\Services\mssecsvc2.0` and `C:\Windows\tasksche.exe` to disk, then exits.

Here you'll need to prevent `tasksche.exe` from starting its services. Defender can do this.

`binwalk C:\Windows\tasksche.exe` once again reveals similar output, but this time with only one PE file embedded:

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	Microsoft executable, portable (PE)
52811	0xCCE4B	Copyright string: Copyright 1993-1998 Mark Adler "
53332	0xD054	CRC32 polynomial table, little endian
54368	0xD460	Copyright string: "Copyright 1998 Gilles Vollant "
55776	0x100F0	Zip archive data, encrypted at least v2.0 to extract, compressed size: 1
79976	0x13868	Zip archive data, encrypted at least v2.0 to extract, compressed size: 1
80180	0x1202D	Zip archive data, encrypted at least v2.0 to extract, compressed size: 0

Now attaching `tasksche.exe` to a debugger, looking at strings across all modules we see a reference to `CryptGenKey`:

00401A71	push tasksche.40F110	0040F110	"CryptAcquireContextA"
00401A79	push tasksche.40F100	0040F100	"CryptImportKey"
00401A86	push tasksche.40F0F0	0040F0F0	"CryptDestroyKey"
00401A93	push tasksche.40F0E0	0040F0E0	"CryptEncrypt"
00401AA0	push tasksche.40F0D0	0040F0D0	"CryptDecrypt"
00401AAD	push tasksche.40F0C4	0040F0C4	"CryptGenKey"
00401B46	push tasksche.40EB88	0040EB88	L"%s\\%s"

Extracting the decryption key from memory is beyond the scope of this analysis, but placing a breakpoint on `CryptGenKey` may be a good starting point to finding the private key(s).

WannaCry appears to have an anti-debugging mechanism. Writing `C:\Windows\tasksche.exe` to disk and kicking off the `<random-string>` service essentially happen simultaneously, separated by milliseconds. This won't occur unless WannaCry is executing at normal speed, (holding down F7 to step through quickly). Until it detects it's executing "normally", it will continue looping over the same routine.

For these reasons, further analysis will be required to fully unpack the malware's components, as well as potentially recover the decryption key(s) from memory.

## IOC & YARA

The following points are the best **indicators of compromise**, to quickly identify infection:

### NETWORK

1. DNS A record query for:

```
www [dot] iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea [dot] com
```

2. HTTP GET request to:

```
[hxxp://] www [dot] iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea [dot] com /
```

3. TCP port 445 connection attempts to public IP addresses flooding the network:

tcp.port==445							
No.	Time	Source	Destination	Protocol	Length	Info	
8191	192.328353689	10.45.45.6	117.34.160.151	TCP	66	54887 → 445 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM	
8197	192.396625755	10.45.45.6	158.25.100.7	TCP	66	54888 → 445 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM	
8198	192.428190446	10.45.45.6	170.200.30.244	TCP	66	54889 → 445 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM	
8199	192.432574567	10.45.45.6	19.192.229.216	TCP	66	54890 → 445 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM	
8200	192.437703695	10.45.45.6	29.228.142.219	TCP	66	54891 → 445 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM	
8210	192.531161961	10.45.45.6	93.40.215.179	TCP	66	[TCP Retransmission] [TCP Port numbers reused] 54874 → 445 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM	
8211	192.596531383	10.45.45.6	175.215.72.187	TCP	66	54894 → 445 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM	
8212	192.596531834	10.45.45.6	49.208.202.65	TCP	66	54892 → 445 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM	
8213	192.596592107	10.45.45.6	188.10.134.95	TCP	66	54893 → 445 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM	
8214	192.641309092	10.45.45.6	17.84.38.41	TCP	66	54895 → 445 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM	
8226	192.688211876	10.45.45.6	70.101.220.46	TCP	66	54896 → 445 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM	
8239	192.812828470	10.45.45.6	150.115.184.124	TCP	66	54897 → 445 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM	
8240	192.812828961	10.45.45.6	147.24.19.188	TCP	66	54898 → 445 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM	
8241	192.859804762	10.45.45.6	79.24.22.19	TCP	66	[TCP Retransmission] [TCP Port numbers reused] 54880 → 445 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM	
8242	192.859805192	10.45.45.6	91.29.176.79	TCP	66	[TCP Retransmission] [TCP Port numbers reused] 54881 → 445 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM	
8243	192.875889504	10.45.45.6	154.68.22.120	TCP	66	54899 → 445 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM	

### HOST

4. C:\Windows\tasksche.exe was created
5. A system service named Microsoft Security Center (2.0) Service (mssecsvc2.0) exists  
Path: HKLM\System\CurrentControlSet\Services\mssecsvc2.0
6. HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Run\ has an entry for <random-string>
7. A hidden folder C:\ProgramData\<random-string>\ was created
8. CLI logs for: vssadmin delete shadows /all /quiet & wmic shadowcopy delete
9. The visual GUI indicators (wallpaper and payment prompt)

## HUNTING QUERIES

WannaCry may flood the logs, but as mentioned in [post infection](#) it does not encrypt them.

A list of suspicious log entries generated is included in the Appendices.

These queries try to make the assumption that file, service, or argument names may later be obfuscated in a future version, and try to match on unique CLI and LOLBAS strings.

DNS requests (Sysmon Event Id='22', Property=4):

```
$query = 'iuqerfsodp9ifjaposdfjhgosurijfaewrwegwea'
```

```
$query = '\w{20,}'
```

Suspicious command line arguments (Sysmon Event Id='1', Property=10):

```
$query = 'attrib(.exe)?\s+\h\s+\.'
```

```
$query = 'icaccls(.exe)?\s+\.\s+/grant\s+Everyone:F\s+/T\s+/C\s+/Q'
```

```
$query = 'vssadmin\s+delete\s+shadows'
```

```
$query = 'wmic\s+shadowcopy\s+delete'
```

```
$query = '-\w\s+security'
```

Suspicious File Creation (Sysmon Event Id='11', Property=5):

```
$query = '\w.vbs'
```

```
$query = '\d+.bat'
```

---

## YARA RULE

The following YARA rule file was developed in efforts to precisely detect this version of the sample. Ideally you can run this like a threat hunting query recursively on a file system and only match on this sample. It's designed to match on **either** *the sha256sum*, or a number of *specific strings this sample contains*. A link to the rule file is included in the appendices below.

```
import "hash"
import "pe"

rule wannacry_ransomware {
    meta:
        last_updated = "2022-10-20"
        author = "straysheep-dev"
        description = "WannaCry ransomware 2017-05-12"
    strings:
        $string1 = "iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea" ascii
        $string2 = "C:\\\\s\\qeriuwjhrf" ascii
        $string3 = "%s -m security" ascii
        $string4 = "lhdfrgui.exe" wide
        $string5 = "115p7UMMngo1pMvkhHijcRdfJNXj6LrLn" ascii
        $string6 = "12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw" ascii
        $string7 = "13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94" ascii
    condition:
        hash.sha256(0, filesize) ==
        "24d004a104d4d54034dbcfcc2a4b19a11f39008a575aa614ea04703480b1022c" or
        pe.is_pe and
        $string1 and
        $string2 and
        $string3 and
        $string4 and
        $string5 and
        $string6 and
        $string7
}
```



## Appendices

### LINKS

Public link to the YARA rule:

<https://github.com/straysheep-dev/malware-analysis/blob/main/wannacry-20170512.yar>

VirusTotal submission for this sample:

<https://www.virustotal.com/gui/file/24d004a104d4d54034dbcffc2a4b19a11f39008a575aa614ea04703480b1022c>

Microsoft Security Bulletin MS17-010 - Critical:

<https://learn.microsoft.com/en-us/security-updates/SecurityBulletins/2017/ms17-010>

Relevant CVE's:

<https://www.cve.org/CVERecord?id=CVE-2017-0143>

<https://www.cve.org/CVERecord?id=CVE-2017-0144>

<https://www.cve.org/CVERecord?id=CVE-2017-0145>

<https://www.cve.org/CVERecord?id=CVE-2017-0146>

<https://www.cve.org/CVERecord?id=CVE-2017-0147>

<https://www.cve.org/CVERecord?id=CVE-2017-0148>

## TARGET FILE EXTENSIONS

The list of file extensions targeted for encryption, this was extracted from the main binary.

.123	.3dm	.3ds	.3g2	.3gp	.602	.accdb	.aes	.ARC	.asc
.asf	.asm	.asp	.avi	.backup	.bak	.bat	.bmp	.brd	.bz2
.cgm	.class	.cmd	.cpp	.crt	.csr	.csv	.dbf	.dch	.der
.dif	.dip	.djvu	.doc	.docb	.docm	.docx	.dot	.dotm	.dotx
.dwg	.edb	.eml	.fla	.flv	.frm	.gif	.gpg	.hwp	.ibd
.iso	.jar	.java	.jpeg	.jpg	.jsp	.key	.lay	.lay6	.ldf
.m3u	.m4u	.max	.mdb	.mdf	.mid	.mkv	.mml	.mov	.mp3
.mp4	.mpeg	.mpg	.msg	.myd	.myi	.nef	.odb	.odg	.odp
.ods	.odt	.onetoc2	.ost	.otg	.otp	.ots	.ott	.p12	.PAQ
.pas	.pdf	.pem	.pfx	.php	.png	.pot	.potm	.potx	.ppam
.pps	.ppsm	.ppsx	.ppt	.pptm	.pptx	.ps1	.psd	.pst	.rar
.raw	.rtf	.sch	.sldm	.sldx	.slk	.sln	.snt	.sql	.sqlite3
.sqlitedb	.stc	.std	.sti	.stw	.suo	.svg	.swf	.sxc	.sxd
.sxi	.sxm	.sxw	.tar	.tbk	.tgz	.tif	.tiff	.txt	.uop
.uot	.vbs	.vcd	.vdi	.vmdk	.vmx	.vob	.vsd	.vsdx	.wav
.wb2	.wk1	.wks	.wma	.wmv	.xlc	.xlm	.xls	.xlsb	.xlsm
.xlsx	.xlt	.xltn	.xltx	.xlw	.zip				

## SUSPICIOUS LOG ENTRIES

```
"C:\Users\User\Documents\Ransomware.wannacry.exe"
C:\Users\User\Documents\Ransomware.wannacry.exe -m security
C:\WINDOWS\tasksche.exe /i
cmd.exe /c "C:\ProgramData\fbslfrkpone399\tasksche.exe"
C:\ProgramData\fbslfrkpone399\tasksche.exe
attrib +h .
icacls . /grant Everyone:F /T /C /Q
taskdl.exe
C:\Windows\system32\cmd.exe /c 269581666413478.bat
cscript.exe //nologo m.vbs
taskdl.exe
@WanaDecryptor@.exe co
cmd.exe /c start /b @WanaDecryptor@.exe vs
@WanaDecryptor@.exe vs
TaskData\Tor\taskhsvc.exe
cmd.exe /c vssadmin delete shadows /all /quiet & wmic shadowcopy delete &
bcdedit /set {default} bootstatuspolicy ignoreallfailures & bcdedit /set
{default} recoveryenabled no & wbadm delete catalog -quiet
taskse.exe C:\ProgramData\fbslfrkpone399\@WanaDecryptor@.exe
cmd.exe /c reg add HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v
"fbslfrkpone399" /t REG_SZ /d "\"C:\ProgramData\fbslfrkpone399
\tasksche.exe\"" /f
taskdl.exe
taskse.exe C:\ProgramData\fbslfrkpone399\@WanaDecryptor@.exe
"C:\ProgramData\fbslfrkpone399\@WanaDecryptor@.exe"
```

CLI captured by Sysmon, then sorted manually. Displayed chronologically, earliest entry at the top.