Practical Malware Analysis & Triage

Malware Analysis Report

Wannacry Crypto Ransomware

August 2022 | Nigel Dryden | v1.0

PRACTICAL MALWARE ANALYSIS & TRIAGE

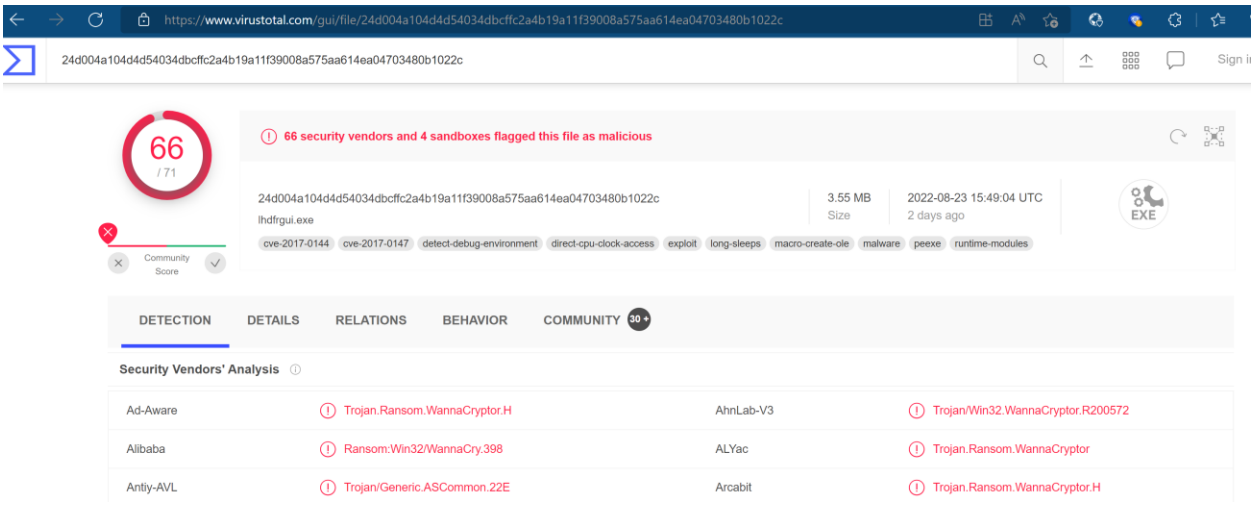
Table of Contents

Executive Summary

MD5 Hash	DB349B97C37D22F5EA1D1841E3C89EB4
SHA256 Hash	24D004A104D4D54034DBCFFC2A4B19A11F39008A575AA614EA04703480B1022C

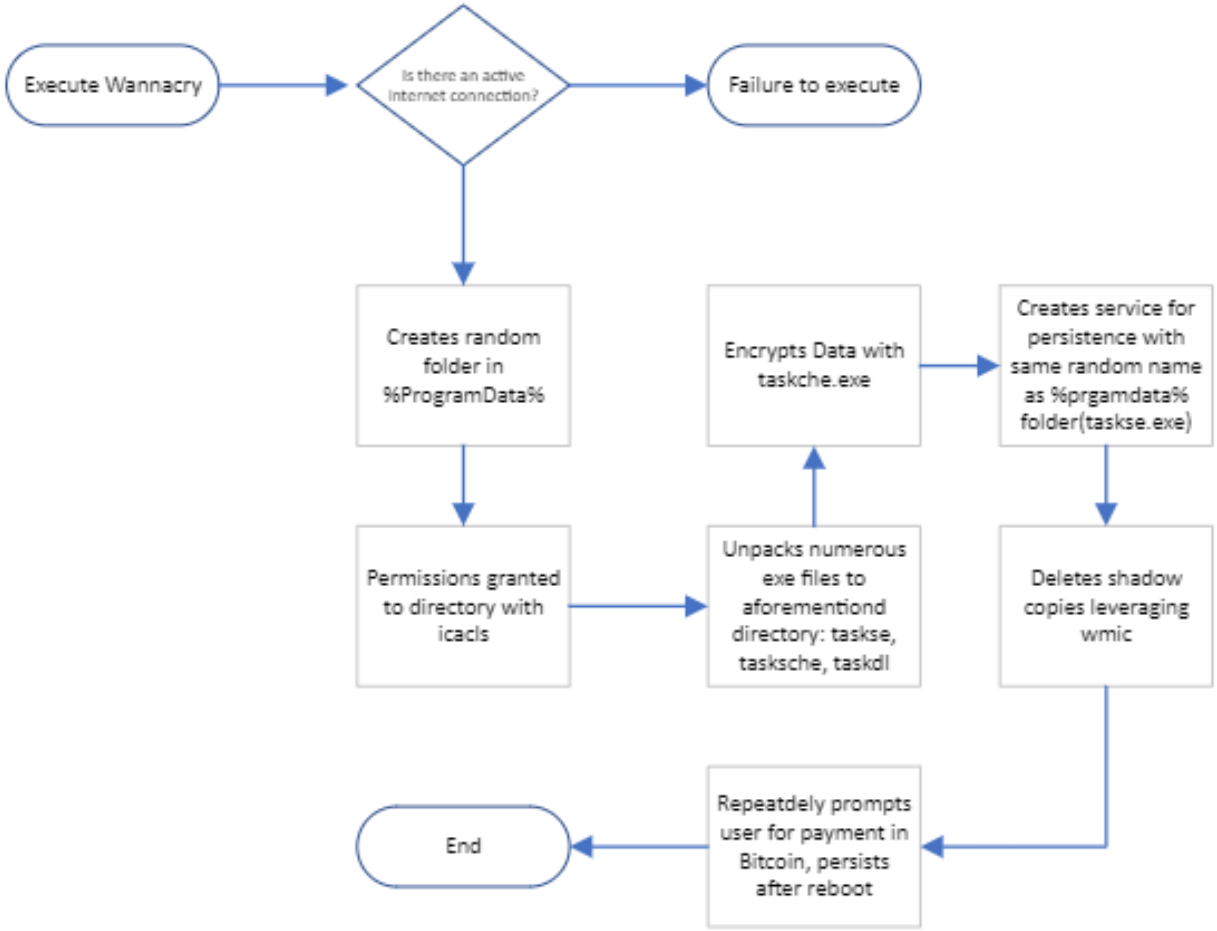
Wannacry is a ransomware crypto malware sample first identified in May 2017. It is a portable executable which runs on Windows operating systems regardless of bitness and comprises three payloads that are executed in succession. These encrypt data, programs, and impact program functionality. Symptoms of infection include frequent pop-ups requesting bitcoin payment with a threatening countdown to create urgency, unpacked files in the %ProgramData% directory and users' desktop, and the background is changed to a prominent warning message.

YARA signature rules are attached in Appendix A. The malware sample and hashes have been submitted to VirusTotal for further examination which yielded positive identification for the main executable and all subsequent packed executables.



High-Level Technical Summary

Wannacry consists of two parts: a portable exe which checks for an internet connection and depending on the value returned prevents execution, or, unpacks three additional executables which are unpacked to a random folder within %ProgramData% and encrypt the user's files and prompt for payment in Bitcoin.



Malware Composition

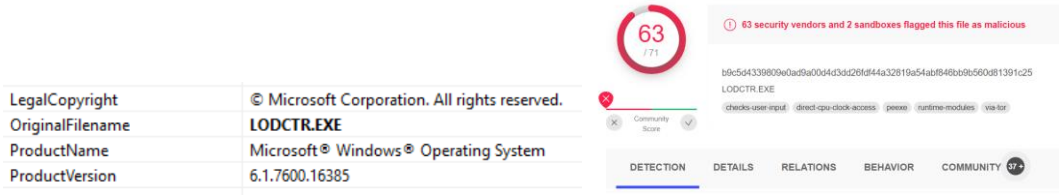
Wannacry Ransomware consists of the following components:

File Name	SHA256 Hash
@wannadecryptor@.exe	B9C5D4339809E0AD9A00D4D3DD26FDF44A32819A54ABF846BB9B560D81391C25
Tasksche.exe	ED01EBFBC9EB5BBEA545AF4D01BF5F1071661840480439C6E5BABE8E080E41AA
Taskdl.exe	4A468603FDCB7A2EB5770705898CF9EF37AADE532A7964642ECD705A74794B79
Taskse.exe	2CA2D550E603D74DEDDA03156023135B38DA3630CB014E3D00B1263358C5F00D

Key Files:

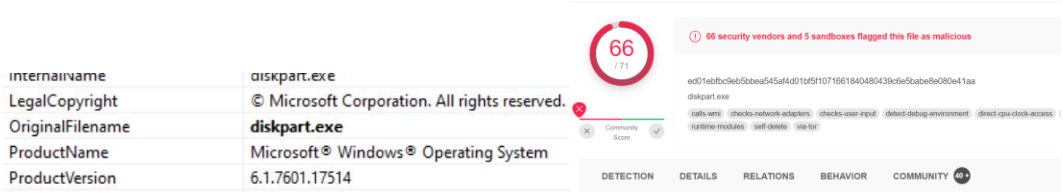
@wannadecryptor@.exe

Portable executable denoted by MZ as the first byte. Triggers other processes, displays ransom message and ensures shadow copies are deleted. Re-engineered LODCTR.exe from Microsoft and defined by VirusTotal as malicious.



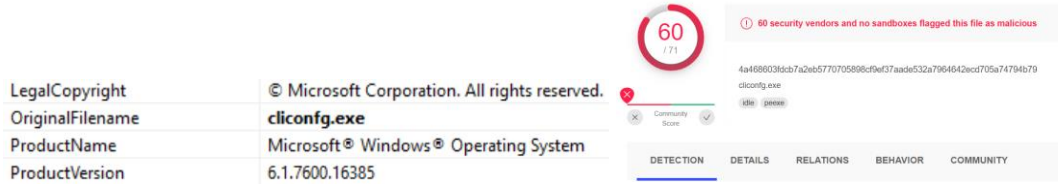
Tasksche.exe

Portable executable denoted by MZ as the first byte. Disguised as diskpart.exe by Microsoft to create perceived legitimacy and detected by VirusTotal as malicious and an integral part of WannaCry RansomWare.



Taskdl.exe

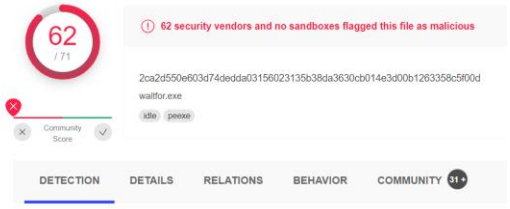
Portable executable denoted by MZ as the first byte. Disguised as cliconfig.exe by Microsoft to create perceived legitimacy and detected by VirusTotal as malicious and an integral part of WannaCry RansomWare.



Taskse.exe

A Base64 encoded CRT file containing the second stage payload. Loren ipsum...

LegalCopyright	© Microsoft Corporation. All rights reserved.
OriginalFilename	**waitfor.exe**
ProductName	Microsoft® Windows® Operating System
ProductVersion	6.1.7600.16385



*eky/pky

Encryption key and private key for handling encryption

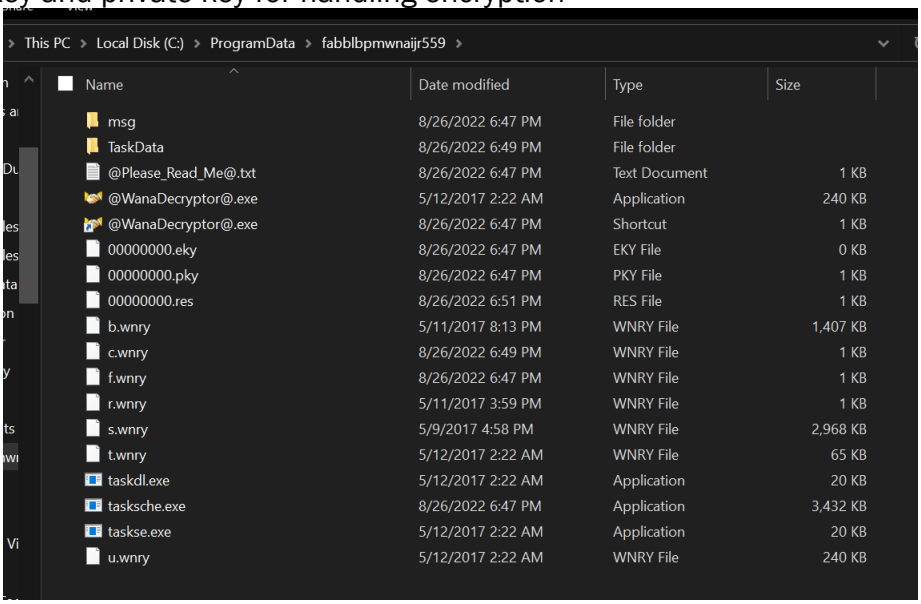


Fig 1: Files extracted upon successful execution

```
C:\Users\drydni\Desktop\fabblbpmwnaijr559
λ file *.*
00000000.pky:            b.out pure object file 86 186 286 286 386 Large Text
00000000.res:            data
@Please_Read_Me@.txt:    ASCII text, with CRLF line terminators
@WanaDecryptor@.exe:     PE32 executable (GUI) Intel 80386, for MS Windows
@WanaDecryptor@.exe.lnk: MS Windows shortcut, Item id list present, Points to a file or directory, Has Relative path,
t Aug 27 10:42:11 2022, mtime=Sat Aug 27 10:42:11 2022, atime=Fri May 12 01:22:56 2017, length=245760, window=hide
b.wnry:                  PC bitmap, Windows 3.x format, 800 x 600 x 24, image size 1440000, resolution 3779 x 3779 px
, bits offset 54
c.wnry:                  data
f.wnry:                  ASCII text, with CRLF line terminators
r.wnry:                  ASCII text, with CRLF line terminators
s.wnry:                  Zip archive data, at least v1.0 to extract, compression method=store
t.wnry:                  data
taskdl.exe:              PE32 executable (GUI) Intel 80386, for MS Windows
tasksche.exe:            PE32 executable (GUI) Intel 80386, for MS Windows
taskse.exe:              PE32 executable (GUI) Intel 80386, for MS Windows
u.wnry:                  PE32 executable (GUI) Intel 80386, for MS Windows
```

Fig 2: File details for those extracted during execution.

Basic Static Analysis

Basic string analysis conducted with floss, PE Studio, and PE View. Floss focused on strings of 7 or more characters. Potential indicators of malicious intent discovered and detailed below:

File is a portable executable as denoted by the first byte:

```
first-bytes-text          M Z .
```

Numerous examples of obfuscation indicating malicious intent:

```
C:\%s\%s
```

```
Microsoft Security Center (2.0) Service
%s -m security
C:\%s\qeriuwjhrf
C:\%s\%s
```

```
cmd.exe /c "%s"
115p7UMMngoj1pMvkpHijcRdfJNXj6LrLn
12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw
13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94
```

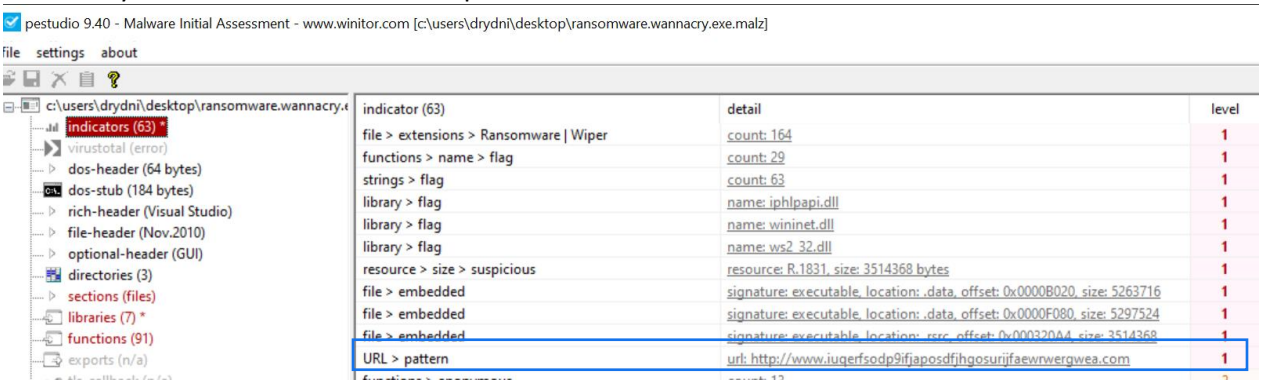
Connection(s) to obscure or unknown URL:

```
http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com
```

Further reinforced by modules imported and marked as potentially malicious:

InternetOpenA	x	-	wininet.dll
InternetOpenUrlA	x	-	wininet.dll
InternetCloseHandle	x	-	wininet.dll

Defined by PEStudio as an indicator of possible malicious intent:



SMB share connection(s) initiated:

```
\\172.16.99.5\IPC$
Windows 2000 2195
Windows 2000 5.0
\\192.168.56.20\IPC$
```

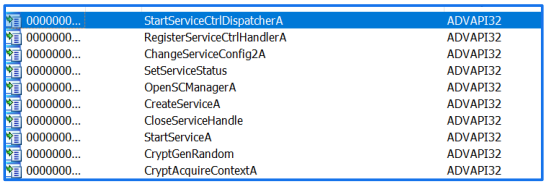
File creation and file name identification:

```
tasksche.exe
CloseHandle
WriteFile
CreateFileA
CreateProcessA
```

Secondary analysis of .exe files grepping for .exe found files of similar name:

```
cmd.exe /c "%s"
tasksche.exe
taskdl.exe
taskse.exed*
taskdl.exe
```

Evidence of service creation:



0000000...	StartServiceCtrlDispatcherA	ADVAPI32
0000000...	RegisterServiceCtrlHandlerA	ADVAPI32
0000000...	ChangeServiceConfig2A	ADVAPI32
0000000...	SetServiceStatus	ADVAPI32
0000000...	OpenSCManagerA	ADVAPI32
0000000...	CreateServiceA	ADVAPI32
0000000...	CloseServiceHandle	ADVAPI32
0000000...	StartServiceA	ADVAPI32
0000000...	CryptGenRandom	ADVAPI32
0000000...	CryptAcquireContextA	ADVAPI32

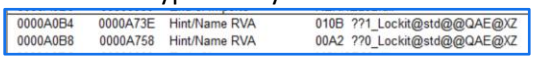
Evidence of encryption:

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

Further reinforced by secondary analysis:

0000000...	CryptGenRandom	ADVAPI32

Possible encryption key evidence:



0000A0B4	0000A73E	Hint/Name RVA	010B	??1_Lockit@std@@QAE@XZ
0000A0B8	0000A758	Hint/Name RVA	00A2	??0_Lockit@std@@QAE@XZ

Permission alterations and attempted obfuscation using hidden attribute:

```
icacls . /grant Everyone:F /T /C /Q
attrib +h .
```

Indicator of Ransomware variant as confirmed by VirusTotal

```
WNcry@2ol7
```

Language agnostic and variant reconfirmed by file type. This can be leveraged by Yara as seen in Appendix S

```
c.wnry%
msg/m_bulgarian.wnry
```

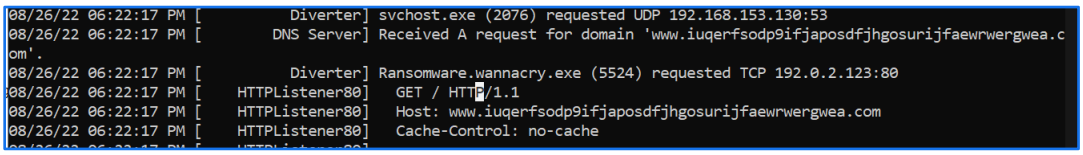
```
msg/m_chinese (simplified).wnryR9
```

Basic Dynamic Analysis

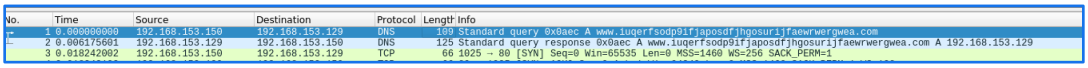
As witnessed in static analysis, connection to the following URL was expected:
Possible connection(s) to obscure or unknown URL:

`http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com`

FakeNet-NG executed locally on the host confirmed the attempts to locate the URL seen within the basic static analysis



When connected to a secondary Remnux workstation acting as a DNS server, Wireshark also captured the same URL in both HTTP and DNS:



However, when an internet connection was present, the malware failed to detonate. This is further explained in the Advanced Static Analysis and Advanced Dynamic Analysis sections below.

Advanced Static Analysis

Cutter and Ida were utilised to dissect and analyse the intricacies and behaviour of the malware. Findings shown in Basic analysis are confirmed through advanced static and dynamic analysis.

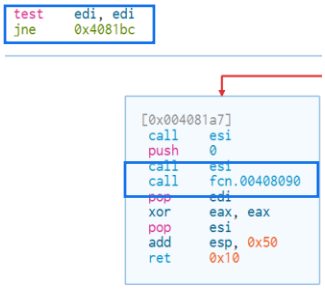
Cutter identified the **main** function within the executable. The URL which had been identified in the basic dynamic analysis section was immediately obvious.

```
mov     ecx, 0xe                    ; 14
mov     esi, str.http:__www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com ; 0x4313d0
lea     edi, [var_8h]
```

Debugging the code proved that the URL is accessed by the program as an initial step. This value is parsed in to the memory and leveraged by the InternetOpenUrlA function to confirm connectivity. Depending upon the result determines whether the Ransomware will complete.

```
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
```

The failure to launch (killswitch) is covered in the Advanced dynamic analysis section. However, should internet access be unavailable or the connection fail, the program will execute.



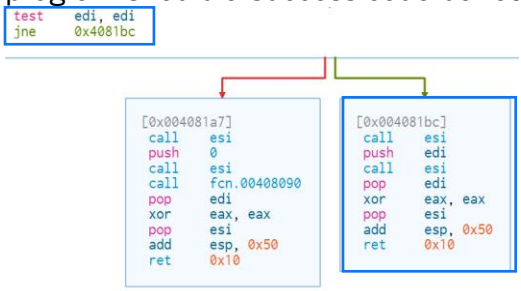
The initial function which contains the crux of the malware is **Fcn.00408090** and proceeds to initiate the malware before routing to numerous other functions to create the services, begin encryption, and create the files and user prompts for payment (shown below)

Fig 3: Ransomware function called post internet connectivity test

Whilst further static analysis has been conducted, as the malware had to be activated, extracted and each component part analysed, these are all detailed (where relevant) in the following Advanced Dynamic Analysis section.

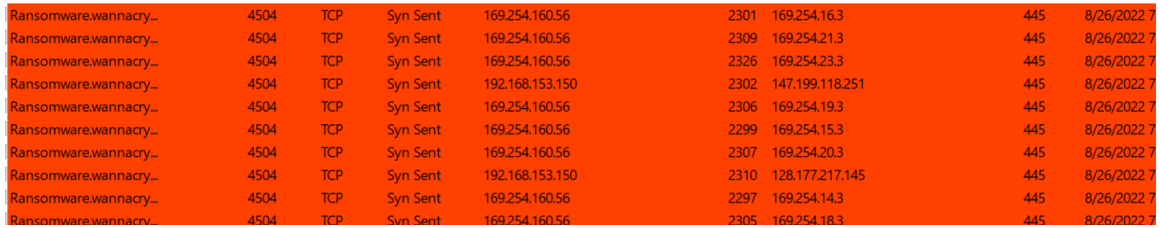
Advanced Dynamic Analysis

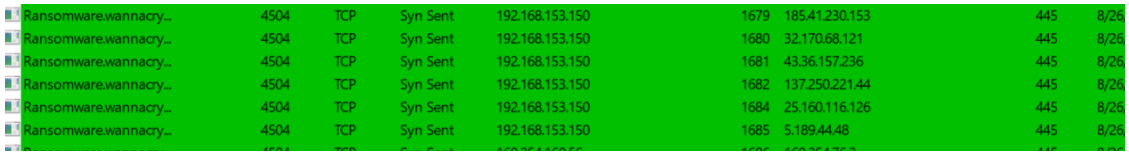
As stated prior, conversely to expectations, the ransomware only executes when a successful connection is **not** established. Using FakeNet and INetSim resulted in a failure to launch by the application and this is proven by the jne jumping out of the program should a success code be returned:



However, if TCP View is launched whilst these connections are not active, the application attempts to connect to numerous locations on port 445.

TCP View:





Furthermore, once executed, a plethora of subfunctions are executed, and whilst analysis by cutter or IDA can provide the exact functions and calls, by simply monitoring in procmon, the processes can be scrutinised and digested more easily.

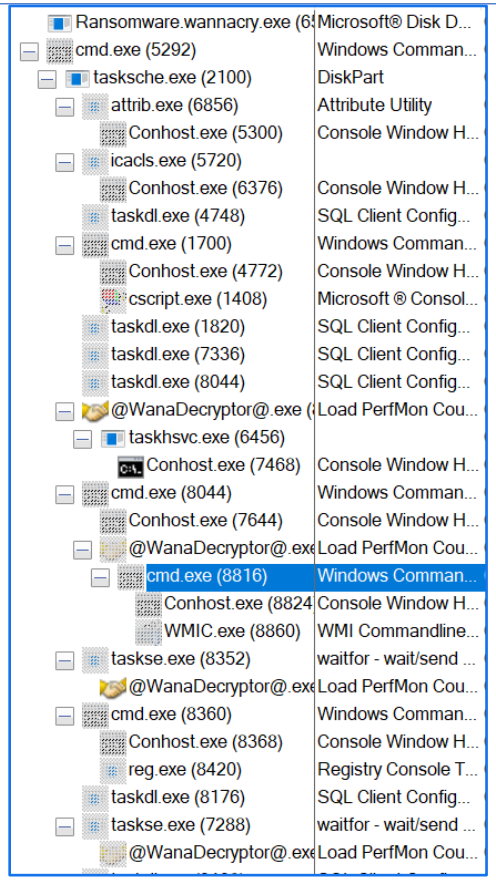
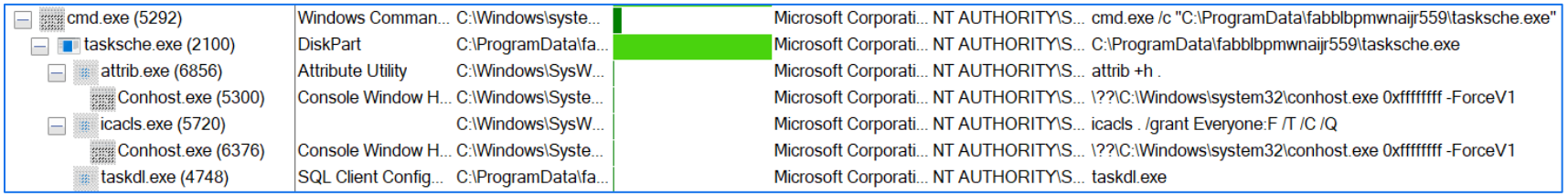


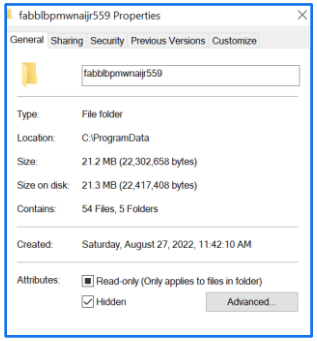
Fig 4: Tasks and associated calls to portable executables

The initial call to **tasksche.exe is by Ransomware Wannacry.exe**.

Tasksche.exe attempts to hide the folder created by wannacry and grants full access permissions to this folder and all subfolders, and suppresses the success messages. Carries out the encryption work due to the modules contained within the application as detailed below:



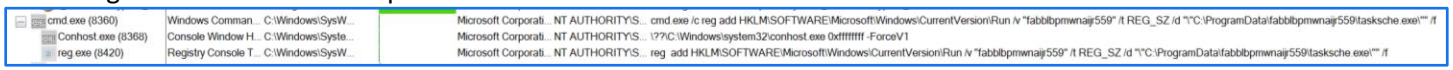
cmd.exe (5292)	Windows Comman...	C:\Windows\syste...	Microsoft Corporati...	NT AUTHORITY\S...	cmd.exe /c "C:\ProgramData\fabblbpmwnaijr559\tasksche.exe"
tasksche.exe (2100)	DiskPart	C:\ProgramData\fa...	Microsoft Corporati...	NT AUTHORITY\S...	C:\ProgramData\fabblbpmwnaijr559\tasksche.exe
attrib.exe (6856)	Attribute Utility	C:\Windows\SysW...	Microsoft Corporati...	NT AUTHORITY\S...	attrib +h .
Conhost.exe (5300)	Console Window H...	C:\Windows\Syste...	Microsoft Corporati...	NT AUTHORITY\S...	\??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
icacls.exe (5720)		C:\Windows\SysW...	Microsoft Corporati...	NT AUTHORITY\S...	icacls . /grant Everyone:F /T /C /Q
Conhost.exe (6376)	Console Window H...	C:\Windows\Syste...	Microsoft Corporati...	NT AUTHORITY\S...	\??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1
taskdl.exe (4748)	SQL Client Config...	C:\ProgramData\fa...	Microsoft Corporati...	NT AUTHORITY\S...	taskdl.exe



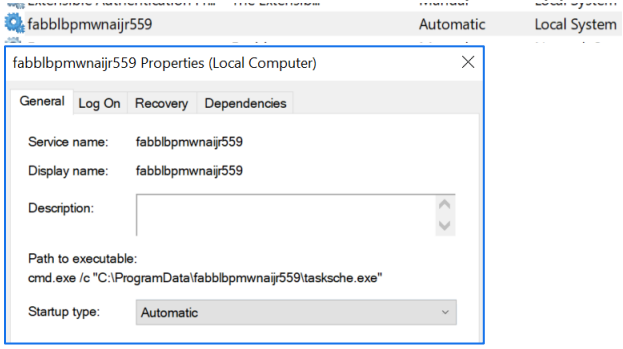
indicator (46)	detail	level
functions > name > flag	count: 14	1
file > extensions > Ransomware | Wiper	count: 163	1
strings > flag	count: 34	1
resource > size > suspicious	resource: XIA,2058, size: 3446325 bytes	1
file > embedded	signature: PKZIP, location: .rsrc, offset: 0x000100F0, size: 3446325	1
file > checksum > invalid	expected: 0x00363012	2
overlay > signature	name: PKZIP	2

encoding (2)	size (bytes)	location	flag (34)	hint (424)	value (111594)
ascii	1430	0x0035A000	-	size	<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">\r\n <tr...
ascii	53	0x0000F08C	-	-	Microsoft Enhanced RSA and AES Cryptographic Provider
ascii	45	0x0000CE3C	-	-	inflate 1.1.3 Copyright 1995-1998 Mark Adler
ascii	43	0x0000D453	-	-	- unzip 0.15 Copyright 1998 Gilles Vollant
unicode	43	0x0B0D991E	-	-	Microsoft Corporation. All rights reserved.
ascii	40	0x0000004D	-	dos-message	!This program cannot be run in DOS mode.
unicode	40	0x0B0D9868	-	-	6.1.7601.17514 (win7sp1_rtm.101119-1850)
ascii	39	0x0000F6E4	-	-	oversubscribed dynamic bit lengths tree
ascii	35	0x0000F4FC	-	utility	icacls . /grant Everyone:F /T /C /Q
ascii	35	0x0000F668	-	-	too many length or distance symbols

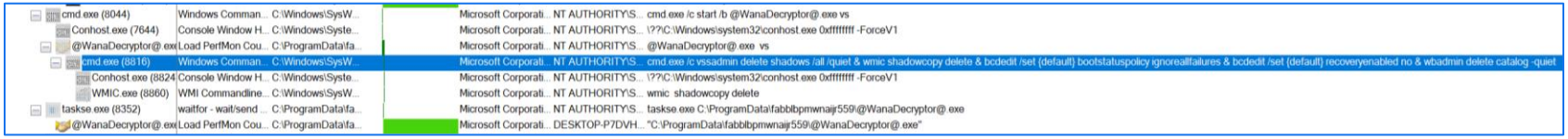
The application creates persistence mechanisms by adding registry keys to the HKLM/…/Run which call to the tasksche.exe on startup so rebooting will not eradicate the problem.



Services are also created and started by the application further reinforcing the persistence mechanisms:



Shadow copies are also rendered useless by WannaDecryptor as it leverages WMIC to destroy any backups, thus adding urgency to the need for decryption.



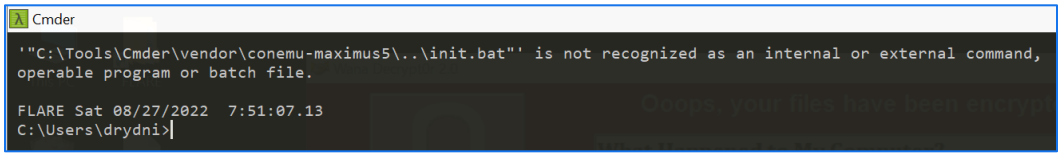
@WanaDecryptor@.exe can be run individually and whilst it appears similar in result, the encryption is not conducted. The encryption warning, and background are applied, but attempting to open the cosmo.jpeg on desktop results in the unencrypted image displayed as standard. This is merely the message delivery system, whereas the other parts perform the heavy lifting and actual damage.

Selecting the “contact us” option from within the display window does show connections initiated by the application on ports 9101 and the standard HTTP port 443.

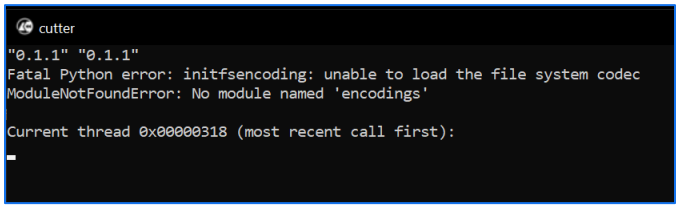
taskhsvc.exe	4608	TCP	Syn Sent	192.168.153.150	11829	178.62.60.37	443
taskhsvc.exe	4608	TCP	Listen	127.0.0.1	9050	0.0.0.0	0
taskhsvc.exe	4608	TCP	Established	127.0.0.1	11822	127.0.0.1	11823
taskhsvc.exe	4608	TCP	Established	127.0.0.1	11823	127.0.0.1	11822
taskhsvc.exe	4608	TCP	Syn Sent	192.168.153.150	11827	185.13.39.197	443
taskhsvc.exe	4608	TCP	Syn Sent	192.168.153.150	11828	128.31.0.39	9101

Whilst the main purpose of the Ransomware is the encryption of files and deletion of backups , this same mechanism also hinders the function of other applications:

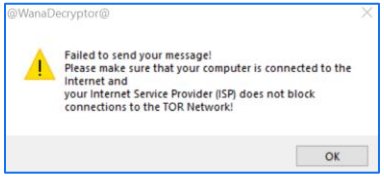
CMDER affected by the encryption clearly impacting critical files.



Furthermore, Cutter was unable to be launched post execution and IDA had to be leveraged instead.



Finally, Tor is leveraged to finalise the transactions as it is contained within the Taskdata folder (see IOC: Fig 9) and attempts at connection to process payment result in the following



Indicators of Compromise

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

Network Indicators

Wireshark capture from Remnux DNS server showing initial connection. However, if the connection is returned as successful, the ransomware fails to detonate as detailed in the Advanced Static Analysis

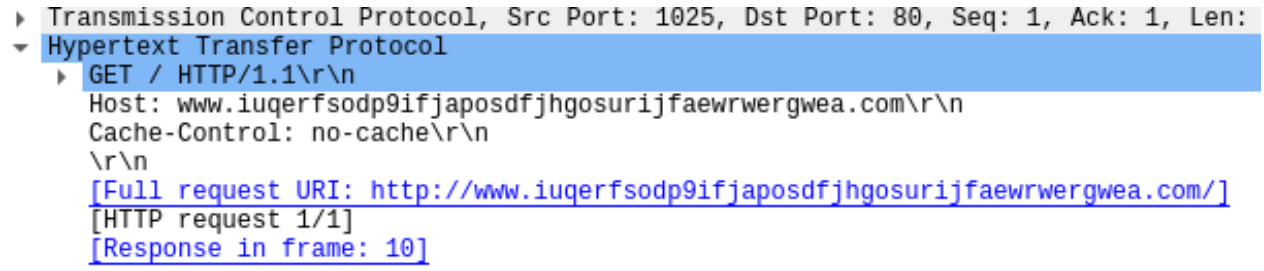


Fig 5: WireShark Packet Capture of initial beacon check-in

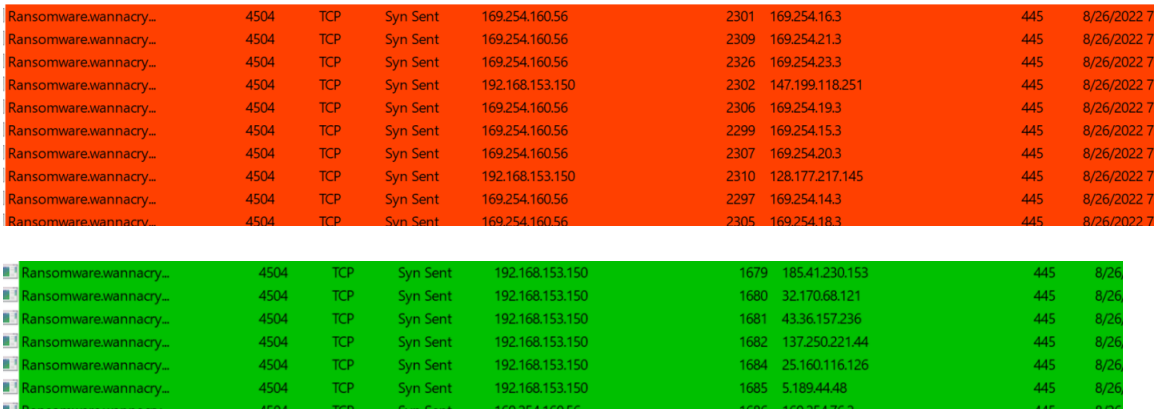


Fig 6: TCP View Captures of TCP traffic on Port 445 by Wannacry

Host-based Indicators

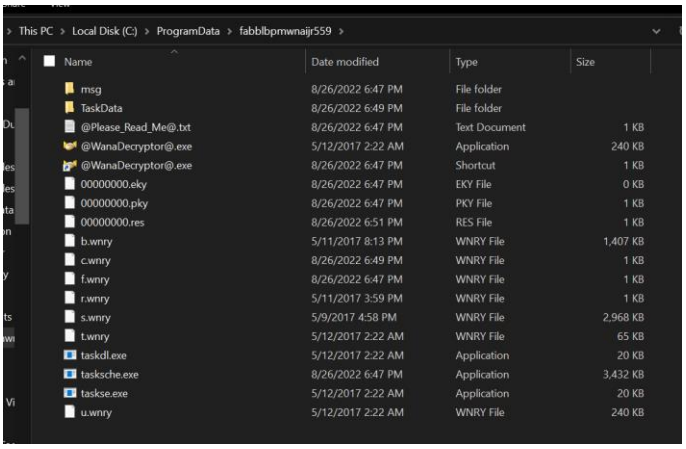


Fig 7: Initial file extraction from core Ransomware Executable

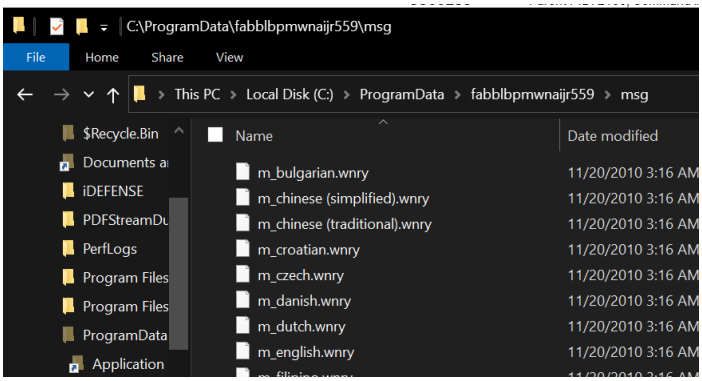


Fig 8: Language files to ensure widest impact

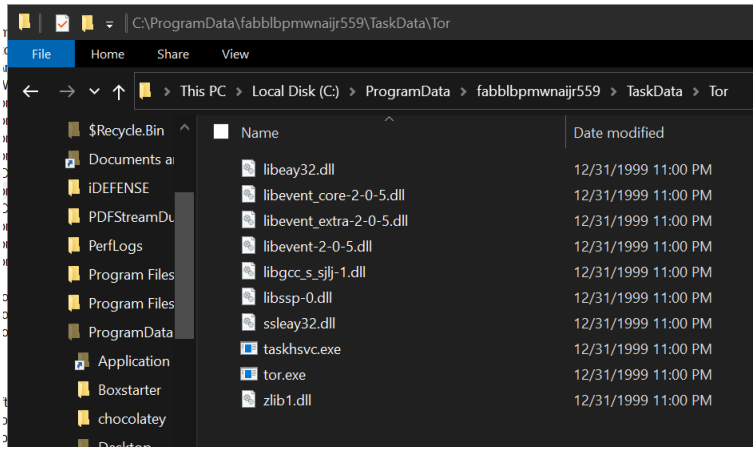


Fig 9: Tasks folder containing Tor.exe likely for extraction connecting obfuscation

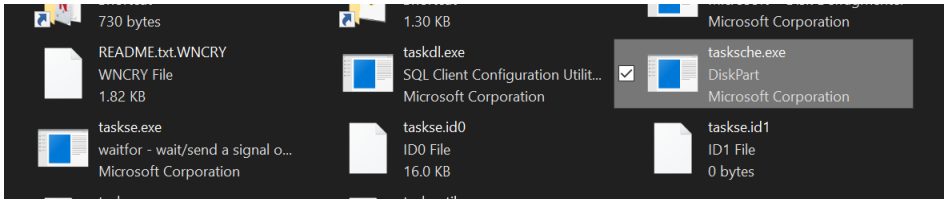


Fig 10: Executables masquerading as normal MS files

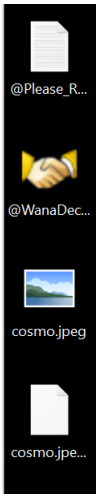


Fig 11: Files added to Desktop

Fig 12: Previously viewable files now inaccessible

*@Please_Read_Me@.txt - Notepad

File Edit Format View Help

```
sQ:  What's wrong with my files?

A:  Ooops, your important files are encrypted. It means you will not be able to access them anymore until they are decrypted.
    If you follow our instructions, we guarantee that you can decrypt all your files quickly and safely!
    Let's start decrypting!

Q:  What do I do?

A:  First, you need to pay service fees for the decryption.
    Please send $300 worth of bitcoin to this bitcoin address: 12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw

    Next, please find an application file named "@WanaDecryptor@.exe". It is the decrypt software.
    Run and follow the instructions! (You may need to disable your antivirus for a while.)

Q:  How can I trust?

A:  Don't worry about decryption.
    We will decrypt your files surely because nobody will trust us if we cheat users.


*   If you need our assistance, send a message by clicking <Contact Us> on the decryptor window.
```

Fig 13: Readme detailing requirements for unecryption

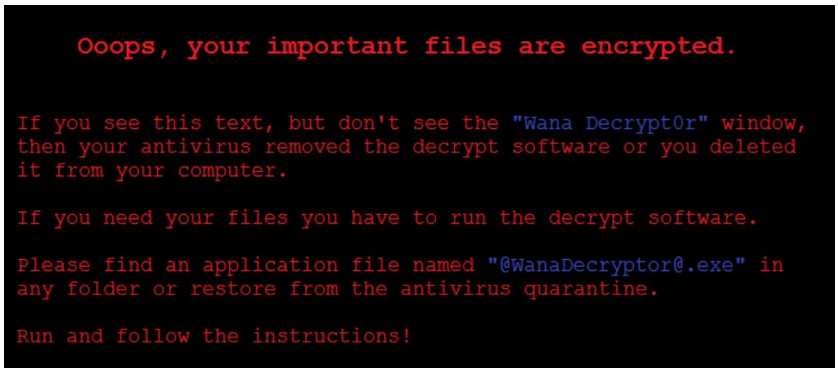


Fig 14: Background bitmap file which is applied to the Desktop

Fig 15: Repeating message informing of attack with timer

Rules & Signatures

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

As detailed previously, numerous key factors allow for detection rules to be created for the detection and prevention of WannaCry Ransomware.

The URL: http://www[.]iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea[.]com

The file extensions: wnry

The executables: @WanaDecryptor@.exe, tasksche.exe, taskdl.exe, and taskse.exe

The first byte denoting the portable executable: “MZ”

Other factors such as text referring to bitcoin, Wikipedia, encryption, shadowcopy, and other references could also be stated to narrow down the rules to with a greater degree of specificity but by focusing on the higher level aspects, it may also be possible to capture variants using the same files and extensions.

Appendices

A. Yara Rules

Full Yara repository located at: http://github.com/HuskyHacks/PMAT-lab

```
rule Wannacry_Ransomware_Rules {

    meta:
        last_updated = "2022-08-28"
        author = "Nigel Dryden"
        description = "Yara rules for the detection and prevention of WannaCry
Ransomware Executions"

    strings:
        // Fill out identifying strings and other criteria
        $URL_string = "http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com"
ascii
        $Filename_string = "wnry"
        $PE_Magic_Byte = "MZ"
        $UnpackedFile1 = "tasksche.exe"
        $UnpackedFile2 = "taskdl.exe"
        $UnpackedFile3 = "taskse.exe"
        $UnpackedFile4 = "@WanaDecryptor@.exe"
        $UnpackedFile5 = "tasksche.exe"


    condition:
        // Fill out the conditions that must be met to identify the binary
        $PE_Magic_Byte at 0 and
        ($Filename_string and $URL_string) and
        ($UnpackedFile1 or $UnpackedFile2 or $UnpackedFile3 or $UnpackedFile4 or
$UnpackedFile5)

}
```

B. Decompiled Code Snippets

```
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
```

Fig 16: Initial call to test internet connectivity

```
Imports from WININET.dll

HINTERNET __stdcall InternetOpenA(LPCSTR lpszAgent, DWORD dwAccessType, LPCSTR lpszProxy, LPCSTR lpszProxyBypass, DWORD dwFlags)
          extrn InternetOpenA:dword
                                        ; CODE XREF: sub_408140+3B↑p
                                        ; DATA XREF: sub_408140+3B↑r ...
HINTERNET __stdcall InternetOpenUrlA(HINTERNET hInternet, LPCSTR lpszUrl, LPCSTR lpszHeaders, DWORD dwHeadersLength, DWORD dwFlags, DWORD_PTR dwContext)
          extrn InternetOpenUrlA:dword
                                        ; CODE XREF: sub_408140+54↑p
                                        ; DATA XREF: sub_408140+54↑r
BOOL __stdcall InternetCloseHandle(HINTERNET hInternet)
          extrn InternetCloseHandle:dword
                                        ; CODE XREF: sub_408140+67↑p
                                        ; sub_408140+6B↑p ...
```

Fig 17: WININET.DLL leveraged for URL call

```
idata:004156BC ; Imports from urlmon.dll
idata:004156BC ;
idata:004156BC ; HRESULT __stdcall URLDownloadToFileA(LPUNKNOWN, LPCSTR, LPCSTR, DWORD, LPBINDSTATUSCALLBACK)
idata:004156BC                 extrn __imp_URLDownloadToFileA:dword
idata:004156BC                                         ; DATA XREF: URLDownloadToFileA↑r
idata:004156BC                                         ; .rdata:0041CC5C↓o
idata:004156C0
idata:004156C0
rdata:004156C4 ; ===========================================================================
rdata:004156C4
rdata:004156C4 ; Segment type: Pure data
rdata:004156C4 ; Segment permissions: Read
rdata:004156C4 _rdata          segment para public 'DATA' use32
rdata:004156C4                 assume cs:_rdata
rdata:004156C4                 ;org 4156C4h
rdata:004156C4                 align 8
rdata:004156C8 off_4156C8      dd offset sub_404C30    ; DATA XREF: sub_401130↑o
rdata:004156CC                 dd offset unk_4156D0
rdata:004156D0 unk_4156D0      db  13h                 ; DATA XREF: .rdata:004156CC↑o
rdata:004156D1                 db    1
rdata:004156D2                 db    0
```

Fig 18: URLMON.DLL leveraged for Downloads

```
.rdata:0040A664                 db 'StartServiceA',0
.rdata:0040A672 word_40A672     dw 3Eh                  ; DATA XREF: .rdata:0040A298↑o
.rdata:0040A674                 db 'CloseServiceHandle',0
.rdata:0040A687                 align 4
.rdata:0040A688 word_40A688     dw 64h                  ; DATA XREF: .rdata:0040A294↑o
.rdata:0040A68A                 db 'CreateServiceA',0
.rdata:0040A699                 align 2
.rdata:0040A69A word_40A69A     dw 1ADh                 ; DATA XREF: .rdata:0040A290↑o
.rdata:0040A69C                 db 'OpenSCManagerA',0
.rdata:0040A6AB                 align 4
.rdata:0040A6AC word_40A6AC     dw 244h                 ; DATA XREF: .rdata:0040A28C↑o
.rdata:0040A6AE                 db 'SetServiceStatus',0
.rdata:0040A6BF                 align 10h
.rdata:0040A6C0 word_40A6C0     dw 34h                  ; DATA XREF: .rdata:0040A288↑o
.rdata:0040A6C2                 db 'ChangeServiceConfig2A',0
.rdata:0040A6D8 word_40A6D8     dw 20Ch                 ; DATA XREF: .rdata:0040A284↑o
.rdata:0040A6DA                 db 'RegisterServiceCtrlHandlerA',0
.rdata:0040A6F6 word_40A6F6     dw 24Ah                 ; DATA XREF: .rdata:off_40A280↑o
.rdata:0040A6F8                 db 'StartServiceCtrlDispatcherA',0
.rdata:0040A714 word_40A714     dw 1AFh                 ; DATA XREF: .rdata:0040A2A8↑o
.rdata:0040A716                 db 'OpenServiceA',0
.rdata:0040A723                 align 4
.rdata:0040A724 aAdvapi32Dll    db 'ADVAPI32.dll',0     ; DATA XREF: .rdata:0040A200↑o
.rdata:0040A731                 align 2
.rdata:0040A732 aWs232Dll       db 'WS2_32.dll',0       ; DATA XREF: .rdata:0040A214↑o
.rdata:0040A73D                 align 2
.rdata:0040A73E word_40A73E     dw 10Bh                 ; DATA XREF: .rdata:off_40A334↑o
.rdata:0040A740                 db '??1_Lockit@std@@QAE@XZ',0
```

Fig 19: Service creation for persistence

```
.rdata:0040A62A aKernel32Dll    db 'KERNEL32.dll',0     ; DATA XREF: .rdata:0040A1EC↑o
.rdata:0040A637                 align 4
.rdata:0040A638 word_40A638     dw 85h                  ; DATA XREF: .rdata:0040A2A4↑o
.rdata:0040A63A                 db 'CryptAcquireContextA',0
.rdata:0040A64F                 align 10h
.rdata:0040A650 word_40A650     dw 96h                  ; DATA XREF: .rdata:0040A2A0↑o
.rdata:0040A652                 db 'CryptGenRandom',0
```

Fig 20: Encryption key generation and creation