

Practical Malware Analysis & Triage

Malware Analysis Report

Wannacry Malware

Oct 2023 | verticalhead04 | v1.0



Table of Contents

Table of Contents	2
Executive Summary	3
High-Level Technical Summary	4
Basic Static Analysis.....	5
Basic Dynamic Analysis	12
Advanced Static Analysis.....	22
Investigation of the high entropy resource section	32
Advanced Dynamic Analysis.....	42
Indicators of Compromise	45
Network Indicators	45
Host-based Indicators	45
Rules & Signatures.....	46
Appendices.....	46
A. Yara Rules	46
B. Capa verbose result.....	47



Executive Summary

SHA-1 hash	e889544aff85ffaf8b0d0da705105dee7c97fe26
------------	--

Wannacry is a Ransomware capable of spreading to vulnerable systems on a network. Indicators of infection include files being encrypted and appended with .wnry extension, wall paper changed containing the instructions to locate the Wannacry decryptor program, and the Wannacry decryptor itself constantly popping-up on the screen containing instructions to connect with the attackers for decryption after payment via Bitcoin.



Figure 1: Sign of wannacry infected computer



High-Level Technical Summary

Wannacry is a sophisticated ransomware known for its advanced defense mechanisms against analysis. It employs techniques such as obfuscating arguments and stack strings, as well as employing conditional execution and time-delay checks to evade detection in sandboxes. Additionally, Wannacry exhibits worm-like behavior, leveraging the Eternalblue exploit to propagate through vulnerable systems on a network. To maintain persistence, it installs itself as a Windows Service. Furthermore, it communicates with a designated kill switch URL, which plays a crucial role in its operation.



Basic Static Analysis

CFF Explorer

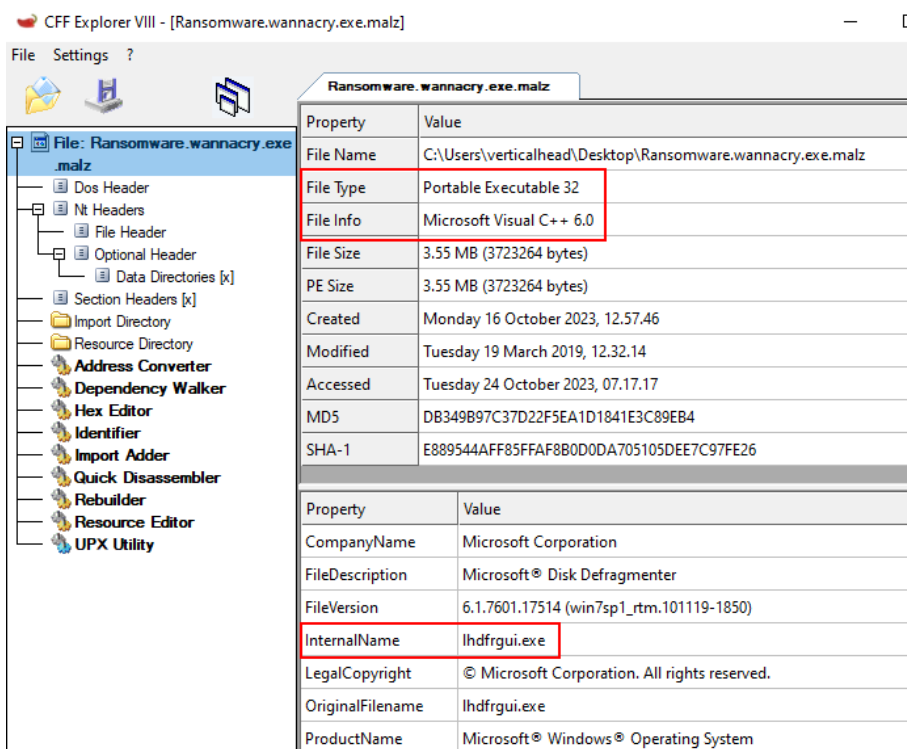


Figure 2: CFF Explorer shows the malware is written in C++ and is 3.55MB in size.

Detect it Easy

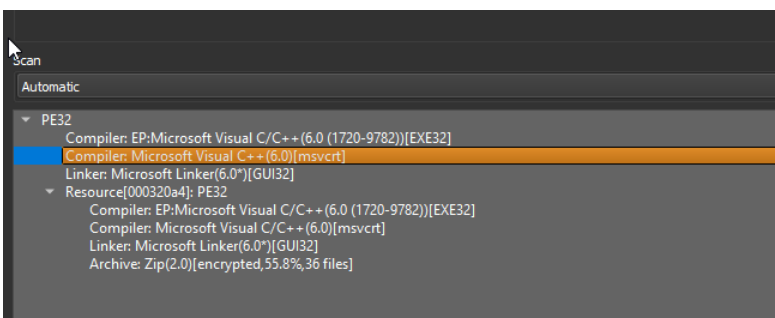


Figure 3: Detect it Easy shows it was compiled using Microsoft Visual C++

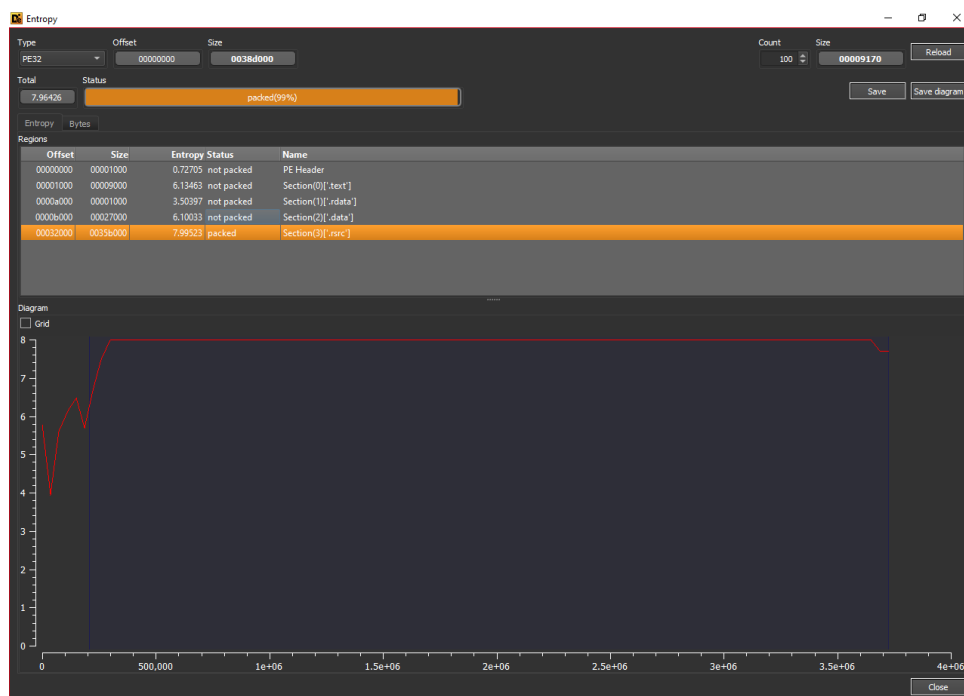


Figure 4: The binary file consists of 5 sections. Notably, the “.rsrc” section displays high entropy which signifies it is likely packed or encrypted.

Yara

```
remnux@remnux:~$ yara -w /usr/local/yara-rules/index.yar Ransomware.wannacry.exe.malz
SEH_Init Ransomware.wannacry.exe.malz
anti_dbg Ransomware.wannacry.exe.malz
win_registry Ransomware.wannacry.exe.malz
win_files_operation Ransomware.wannacry.exe.malz
Str_Win32_Winsock2_Library Ransomware.wannacry.exe.malz
Str_Win32_Wininet_Library Ransomware.wannacry.exe.malz
Str_Win32_Internet_API Ransomware.wannacry.exe.malz
CRC32_poly_Constant Ransomware.wannacry.exe.malz
CRC32_table Ransomware.wannacry.exe.malz
Rijndael_AES Ransomware.wannacry.exe.malz
Rijndael_AES_CHAR Ransomware.wannacry.exe.malz
maldoc_indirect_function_call_3 Ransomware.wannacry.exe.malz
maldoc_getEIP_method_1 Ransomware.wannacry.exe.malz
MS17_010_WannaCry_worm Ransomware.wannacry.exe.malz
WannaDecryptor Ransomware.wannacry.exe.malz
NHS_Strain_Wanna Ransomware.wannacry.exe.malz
Wanna_Cry_Ransomware_Generic Ransomware.wannacry.exe.malz
WannaCry_Ransomware Ransomware.wannacry.exe.malz
WannaCry_Ransomware_Gen Ransomware.wannacry.exe.malz
WannaCry_Ransomware_Dropper Ransomware.wannacry.exe.malz
WannaCry_SMB_Exploit Ransomware.wannacry.exe.malz
wannacry_static_ransom Ransomware.wannacry.exe.malz
worm_ms17_010 Ransomware.wannacry.exe.malz
IsPE32 Ransomware.wannacry.exe.malz
IsWindowsGUI Ransomware.wannacry.exe.malz
IsPacked Ransomware.wannacry.exe.malz
HasRichSignature Ransomware.wannacry.exe.malz
Microsoft_Visual_Cpp_v60 Ransomware.wannacry.exe.malz
Microsoft_Visual_Cpp_v50v60_MFC_additional Ransomware.wannacry.exe.malz
Microsoft_Visual_Cpp_v50 Ransomware.wannacry.exe.malz
Microsoft_Visual_Cpp_v50v60_MFC Ransomware.wannacry.exe.malz
Microsoft_Visual_Cpp Ransomware.wannacry.exe.malz
```

Figure 5: Yara detections

Wannacry Malware
Oct 2023
v1.0



Yara detected functions for anti-debugging, network, registry, and file operations, crypto signatures like CRC32 and the use of Rijndael AES encryption. It also matched the MS17-010 wannacry worm signature. Other signatures indicate it is a packed portable executable file compiled using Microsoft Visual C++.

PESTUDIO

indicator (36)	detail	level
file > embedded	signature: executable, location: .data, offset: 0x0000B020, size: 5263716 (bytes)	1
file > embedded	signature: executable, location: .data, offset: 0x0000F080, size: 5297524 (bytes)	1
file > embedded	signature: executable, location: .rsrc, offset: 0x000320A4, size: 3514368 (bytes)	1
file > extension > count	159	1
libraries > flag > name	Windows Socket Library	1
libraries > flag > name	IP Helper API	1
libraries > flag > name	Internet Extensions for Win32 Library	1
imports > flag > count	28	1
string > size > suspicious	2039 bytes	2
string > size > suspicious	1403 bytes	2
string > size > suspicious	2693 bytes	2
string > size > suspicious	3926 bytes	2
string > size > suspicious	1554 bytes	2
string > size > suspicious	1430 bytes	2
string > size > suspicious	2988 bytes	2
resource > size	R.1831, 3514368 bytes	2
resources > file-ratio	94.41%	2
file > checksum	0x00000000	2
groups > API	synchronization, execution, file, resource, dynamic-library, memory, reconnaissance, services, cryptography, network	2
string > URL	http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrrergwea.com	2
mitre > technique	T1057, T1105, T1124, T1497, T1106, T1569, T1543, T1027, T1082, T1001, T1083, T1055, T1012, T1112, T1485, T1059, T1158	2
file > entropy	7.964	3
file > signature	Microsoft Visual C++ v6.0	3
file > footprint	24D004A104D4D54034DBCFFC2A4B19A11F39008A575AA614EA04703480B1022C	3
file > size	3723264 bytes	3
rich-header > checksum	0xC33D5D11	3
rich-header > offset	0x00000080	3
rich-header > footprint	D4496034DE1F5AF97B361FCDC86EB5D939978830DFF8BF01B6AB3C93961AA425	3
file > tooling	Visual Studio 6.0	3
security > protection	data-execution-prevention (DEP) > OFF	3
security > protection	control-flow-guard (CFG) > OFF	3
security > protection	address-space-layout-randomization (ASLR) > OFF	3
file-name > version	lhdfgrui.exe	3
security > protection	code-integrity (CI) > OFF	3
file > subsystem	GUI	3
imports > ordinal > count	13	3

Figure 6: Indicator section

- Notable string observed
 - string > URL, <http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrrergwea.com>, 2



Imports

pestudio 9.55 - Malware Initial Assessment - www.winitor.com - [c:\users\verticalhead\desktop\ransomware.wannacry.exe.malz\ransomware.wannacry.exe.malz]

file settings about

	imports (91)	flag (28)	first-thunk-original (INT)	first-thunk (AT)	hint	group (16)	technique (8)	type (1)	ordinal (13)	library (7)
StartServiceCtrlDispatcherA	x	0x0000A6F6	0x0000A6F6	0x0000A6F6	586 (0x024A)	services	-	implicit	-	ADVAPI32.dll
ChangeServiceConfig2A	x	0x0000A6C0	0x0000A6C0	0x0000A6C0	52 (0x0034)	services	T1569 System Services	implicit	-	ADVAPI32.dll
CreateServiceA	x	0x0000A688	0x0000A688	0x0000A688	100 (0x0064)	services	T1543 Create or Modify System Proc...	implicit	-	ADVAPI32.dll
QueryPerformanceFrequency	x	0x0000A43A	0x0000A43A	0x0000A43A	676 (0x02A4)	reconnaissance	-	implicit	-	KERNEL32.dll
3 (closesocket)	x	0x80000003	0x80000003	0 (0x0000)	0 (0x0000)	network	-	implicit	x	WS2_32.dll
16 (recv)	x	0x80000010	0x80000010	0 (0x0000)	0 (0x0000)	network	-	implicit	x	WS2_32.dll
19 (send)	x	0x80000013	0x80000013	0 (0x0000)	0 (0x0000)	network	-	implicit	x	WS2_32.dll
8 (htenl)	x	0x80000008	0x80000008	0 (0x0000)	0 (0x0000)	network	-	implicit	x	WS2_32.dll
14 (ntohl)	x	0x8000000E	0x8000000E	0 (0x0000)	0 (0x0000)	network	-	implicit	x	WS2_32.dll
115 (WSAStartup)	x	0x80000073	0x80000073	0 (0x0000)	0 (0x0000)	network	-	implicit	x	WS2_32.dll
12 (inet_ntoa)	x	0x8000000C	0x8000000C	0 (0x0000)	0 (0x0000)	network	-	implicit	x	WS2_32.dll
10 (closesocket)	x	0x8000000A	0x8000000A	0 (0x0000)	0 (0x0000)	network	-	implicit	x	WS2_32.dll
10 (select)	x	0x80000012	0x80000012	0 (0x0000)	0 (0x0000)	network	-	implicit	x	WS2_32.dll
9 (htons)	x	0x80000009	0x80000009	0 (0x0000)	0 (0x0000)	network	-	implicit	x	WS2_32.dll
23 (socket)	x	0x80000017	0x80000017	0 (0x0000)	0 (0x0000)	network	-	implicit	x	WS2_32.dll
4 (connect)	x	0x80000004	0x80000004	0 (0x0000)	0 (0x0000)	network	-	implicit	x	WS2_32.dll
11 (inet_addr)	x	0x8000000B	0x8000000B	0 (0x0000)	0 (0x0000)	network	-	implicit	x	WS2_32.dll
GetAdaptersInfo	x	0x0000A792	0x0000A792	28 (0x001C)	147 (0x0093)	network	-	implicit	-	iphlpapi.dll
InternetOpenA	x	0x0000A7DC	0x0000A7DC	146 (0x0092)	147 (0x0093)	network	-	implicit	-	WININET.dll
InternetOpenUrlA	x	0x0000A7C8	0x0000A7C8	147 (0x0093)	105 (0x0069)	network	-	implicit	-	WININET.dll
InternetCloseHandle	x	0x0000A7B2	0x0000A7B2	105 (0x0069)	623 (0x026F)	file	T1105 Remote File Copy	implicit	-	KERNEL32.dll
MoveFileExA	x	0x0000A576	0x0000A576	623 (0x026F)	326 (0x0146)	execution	T1057 Process Discovery	implicit	-	KERNEL32.dll
GetCurrentThreadId	x	0x0000A524	0x0000A524	326 (0x0146)	325 (0x0145)	execution	-	implicit	-	KERNEL32.dll
GetCurrentThread	x	0x0000A53A	0x0000A53A	325 (0x0145)	150 (0x0096)	cryptography	T1027 Obfuscated Files or Information	implicit	-	ADVAPI32.dll
CryptGenRandom	x	0x0000A650	0x0000A650	150 (0x0096)	133 (0x0085)	cryptography	T1027 Obfuscated Files or Information	implicit	-	ADVAPI32.dll
CryptAcquireContextA	x	0x0000A638	0x0000A638	133 (0x0085)	678 (0x02A6)	cryptography	T1027 Obfuscated Files or Information	implicit	-	MSVCRT.dll
rand	x	0x0000A624	0x0000A624	678 (0x02A6)	692 (0x02B4)	cryptography	T1027 Obfuscated Files or Information	implicit	-	MSVCRT.dll
srand	x	0x0000A632	0x0000A632	692 (0x02B4)	912 (0x0360)	synchronization	-	implicit	-	KERNEL32.dll
WaitForSingleObject	-	0x0000A4F8	0x0000A4F8	912 (0x0360)	556 (0x022C)	synchronization	-	implicit	-	KERNEL32.dll
InterlockedDecrement	-	0x0000A49C	0x0000A49C	556 (0x022C)	552 (0x0228)	synchronization	-	implicit	-	KERNEL32.dll
InterlockedIncrement	-	0x0000A49E	0x0000A49E	552 (0x0228)	152 (0x0098)	synchronization	-	implicit	-	KERNEL32.dll
EnterCriticalSection	-	0x0000A4A6	0x0000A4A6	152 (0x0098)	593 (0x0251)	synchronization	-	implicit	-	KERNEL32.dll
LeaveCriticalSection	-	0x0000A49E	0x0000A49E	593 (0x0251)						

Figure 7: Imports section

- Cryptography - T1027 | Obfuscated Files or Information
- Execution - T1057 | Process Discovery
- File - T1105 | Remote File Copy
- services T1569 | System Services - T1543 | Create or Modify System Process
- network using the dlls below
 - WS2_32.dll
 - iphlpapi.dll
 - WININET.dll



Version

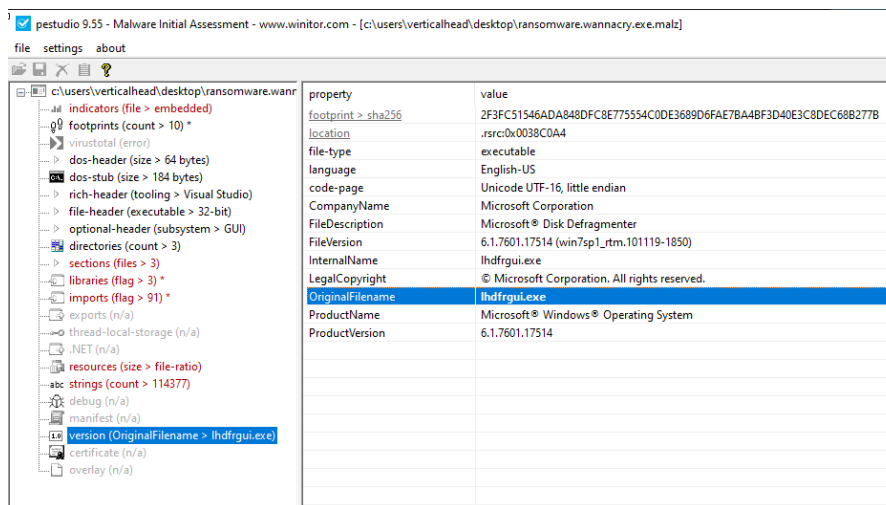


Figure 8: Masquerade as Microsoft Disk Defragmenter

- OriginalFilename: lhdfmgrui.exe
- Showing as Microsoft Disk Defragmenter
- SHA256:
2F3FC51546ADA848DFC8E775554C0DE3689D6FAE7BA4BF3D40E3C8DEC68B277B
 - No matches found in Virustotal

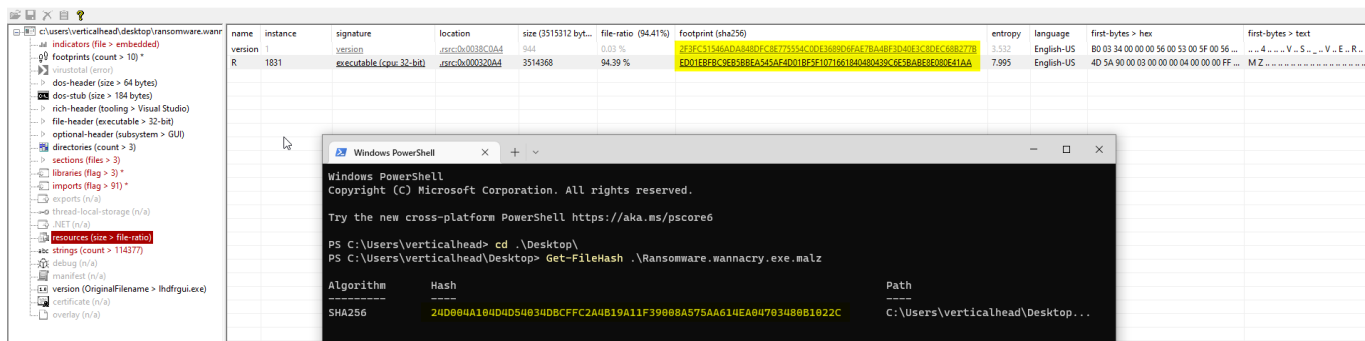


Figure 9: Resources section

- 2F3FC51546ADA848DFC8E775554C0DE3689D6FAE7BA4BF3D40E3C8DEC68B277B
 - Signature: version.
 - Same hash we observed above from OriginalFilename: lhdfmgrui.exe
 - Possibly used to evade or hide the executable's identity.

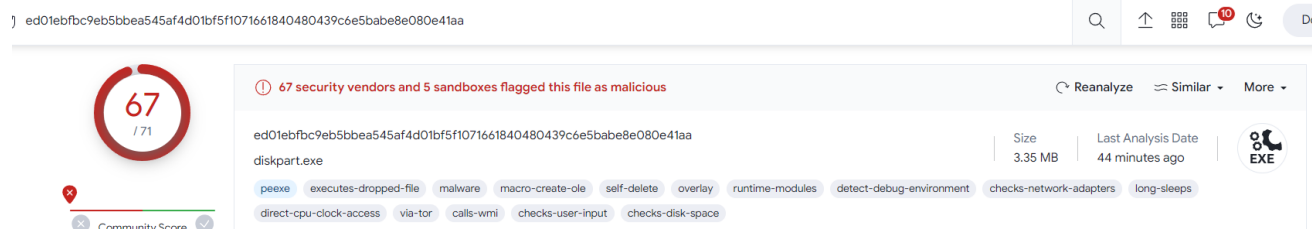


Figure 10: Virustotal result

Security vendors' analysis		Do you want to automate checks?	
AhnLab-V3	Trojan:Win32/WannaCryptor.R200571	Alibaba	Ransom:Win32/WannaCry.a1020010
ALYac	Trojan.Ransom.WannaCryptor	Antiy-AVL	Trojan[Ransom]/Win32.Scatter
Arcabit	Trojan.Ransom.WannaCryptor.A	Avast	Win32:WanaCry-A [Trj]
AVG	Win32:WanaCry-A [Trj]	Avira (no cloud)	TR/Ransom.JB
Baidu	Win32:Trojan.WannaCry.c	BitDefender	Trojan.Ransom.WannaCryptor.A
BitDefenderTheta	Gen:NN.Zexaf.36738.w10@eGEm53di	Bkav Pro	W32.WanaCryptBTIC.Worm
ClamAV	Win.Ransomware.Wannacryptor-994018...	CrowdStrike Falcon	Win/malicious_confidence_100% (W)
Cybereason	Malicious.faa6cb	Cylance	Unsafe
Cynet	Malicious (score: 100)	DeepInstinct	MALICIOUS
DrWeb	Trojan.Encoder.11432	Elastic	Malicious (high Confidence)
Emsisoft	Trojan.Ransom.WannaCryptor.A (B)	eScan	Trojan.Ransom.WannaCryptor.A
ESET-NOD32	Win32/Filecoder.WannaCryptor.D	F-Secure	Trojan.TR/Ransom.JB
Fortinet	W32/WannaCryptor.6F871tr.ransom	GData	Win32:Trojan-Ransom.WannaCry.A
Google	Detected	Gridinsoft (no cloud)	Ransom.Win32.Filecoder.dd
Ikarus	Trojan-Ransom.WannaCry	Jiangmin	Trojan.Wanna.cio

Figure 11: AV vendor signatures from Virustotal

- ED01EBFBC9EB5BBEA545AF4D01BF5F1071661840480439C6E5BABE8E080E41AA
 - According to VT, it matched wannacry signatures from almost all security vendors.
 - Name of this file is R.
 - Signature: executable (cpu-32-bit)
 - Entropy: 7.995 which is a high entropy value. Indicates that this could be a compressed file.
 - It also has a high file-ratio which is 94.39%.
 - After some research on how to approach this, I came across this [link](#).
 - According to the wannacry sample in this article, it contained the same high entropy value with an executable signature just like what we have on this wannacry sample.
 - The article used Resource Hacker to save the resource as a separate file.
 - Continuation for analysis on this high entropy resource section is included in the Advanced Static Analysis section.



Strings

flag (58)	label (543)	group (16)	technique (17)	value (114377)
x	-	execution	T1106 Execution through API	CreateProcess
x	-	execution	T1106 Execution through API	CreateProcess
x	import	services	T1543 Create or Modify System Process	CreateService
x	import	services	T1543 Create or Modify System Process	CreateService
x	import	cryptography	T1027 Obfuscated Files or Information	CryptAcquireContext
x	import	cryptography	T1027 Obfuscated Files or Information	CryptAcquireContext
x	-	cryptography	T1027 Obfuscated Files or Information	CryptDecrypt
x	-	cryptography	T1027 Obfuscated Files or Information	CryptDestroyKey
x	-	cryptography	T1027 Obfuscated Files or Information	CryptEncrypt
x	-	cryptography	T1027 Obfuscated Files or Information	CryptGenKey
x	import	cryptography	T1027 Obfuscated Files or Information	CryptGenRandom
x	-	cryptography	T1027 Obfuscated Files or Information	CryptImportKey
x	-	cryptography	T1027 Obfuscated Files or Information	CryptReleaseContext
x	-	file	T1485 Data Destruction	DeleteFile
x	import	network	-	GetAdaptersInfo
x	-	execution	T1057 Process Discovery	GetCurrentProcess
x	-	reconnaissance	T1057 Process Discovery	GetCurrentProcessId
x	import	execution	-	GetCurrentThread
x	import	execution	T1057 Process Discovery	GetCurrentThreadId
x	import	execution	T1057 Process Discovery	GetCurrentThreadId
x	-	execution	-	GetEnvironmentStrings
x	-	execution	-	GetExitCodeProcess
x	-	reconnaissance	-	GetNativeSystemInfo
x	-	desktop	-	GetProcessWindowStation
x	-	desktop	-	GetObjectInformation
x	import	network	-	InternetCloseHandle
x	import	network	-	InternetOpen
x	import	network	-	InternetOpenUrl
x	-	file	T1105 Remote File Copy	MoveFile
x	import	file	T1105 Remote File Copy	MoveFileEx
x	import	file	T1105 Remote File Copy	MoveFileEx
x	import	reconnaissance	-	QueryPerformanceFrequency
x	-	registry	T1112 Modify Registry	RegCreateKey
x	-	registry	T1112 Modify Registry	RegSetValueEx
x	-	execution	-	RtlLookupFunctionEntry
x	-	-	-	SetCurrentDirectory
x	-	-	-	SetCurrentDirectory
x	-	file	-	SetFileAttributes
x	import	services	-	StartServiceCtrlDispatcher
x	-	execution	-	TerminateProcess
x	-	execution	-	TerminateProcess
x	-	memory	T1055 Process Injection	VirtualAlloc
x	-	memory	T1055 Process Injection	VirtualProtect

Figure 12: MITRE Techniques

- These are the MITRE Techniques observed from strings output in pestudio.
 - T1027 | Obfuscated Files or Information
 - T1055 | Process Injection
 - T1057 | Process Discovery
 - T1105 | Remote File Copy
 - T1106 | Execution through API
 - T1112 | Modify Registry
 - T1485 | Data Destruction
 - T1543 | Create or Modify System Process



Basic Dynamic Analysis

Summary of replication method and results

- I. Double click ransomware sample
 - a. No execution
- II. Run as Administrator (inetsim disabled)
 - a. Ransomware executed.
- III. Run as Administrator (inetsim/fake-net enabled)
 - a. Ransomware did not execute.
- IV. Run as Administrator (inetsim enabled) | Remote Server setup
 - a. To possibly replicate T1105 | Remote File Copy
 - b. To check what's up with \\<Private IP>\IPC\$
 - c. No remote file copy was observed

I. Double click ransomware sample

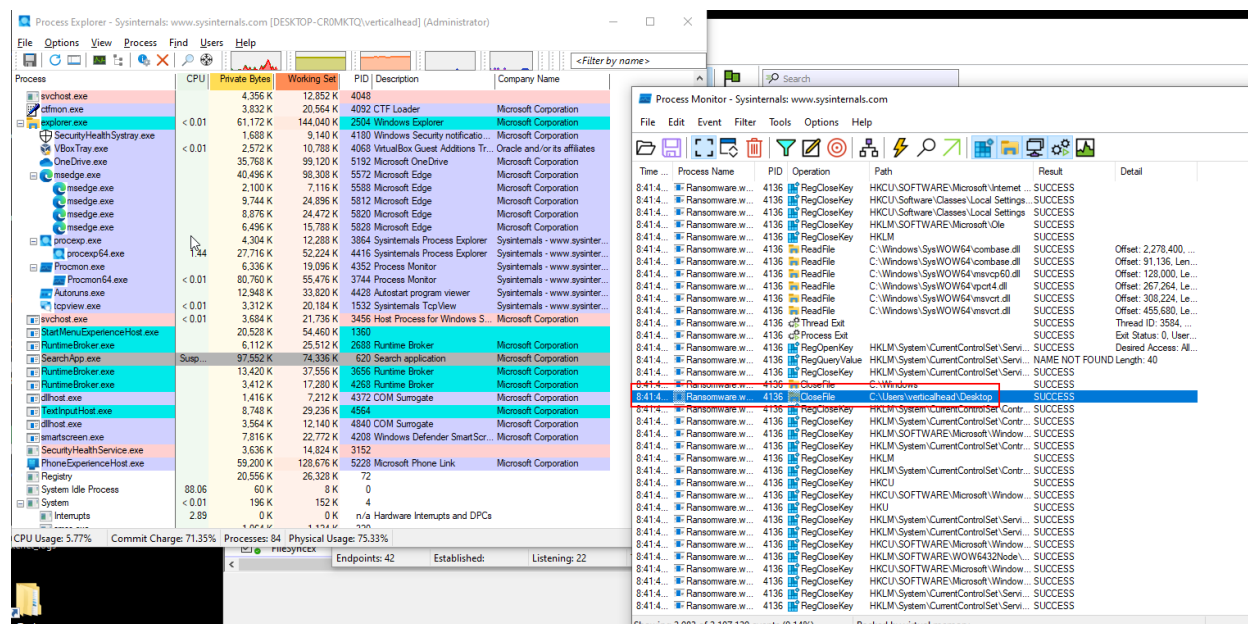


Figure 13: The sample will not proceed with its routine and close itself if not run as Administrator.



II. Run as Administrator (inetsim disabled)

Process Name	Private Bytes	Working Set	Virtual Bytes	Session ID	Description	Company Name
Ransomware.wannacry.exe	< 0.01	24,836 K	21,056 K	3696	Microsoft® Disk Defragmenter	Microsoft Corporation
tasksche.exe	< 0.01	18,488 K	25,580 K	3604	DiskPart	Microsoft Corporation
@WanaDecryptor					Mon Counters	Microsoft Corporation
taskshvc.exe					Window Host	Microsoft Corporation
conhost.exe					Volume Shadow ...	Microsoft Corporation
VSSVC.exe					Mon Counters	Microsoft Corporation
@WanaDecryptor						
Registry						

Command Line:
C:\Users\verticalhead\Desktop\Ransomware.wannacry.exe -m security

Path:
C:\Users\verticalhead\Desktop\Ransomware.wannacry.exe

Services:
Microsoft Security Center (2.0) Service [mssecsvc2.0]

Figure 14: Command line and the process description masquerading as Microsoft Disk Defragmenter

Command Line:

C:\Users\verticalhead\Desktop\Ransomware.wannacry.exe -m security

Services:

mssecsvc2.0 Microsoft Security Center (2.0) Service

Process Name	Private Bytes	Working Set	Virtual Bytes	Session ID	Description	Company Name
wininit.exe	< 0.01	1,836 K	5,216 K	484		
services.exe		1,368 K	6,376 K	576		
svchost.exe	1.54	3,772 K	7,768 K	712	Host Process for Windows S...	Microsoft Corporation
ShellExperienceHost...	< 0.01	10,252 K	15,768 K	2668	Windows Shell Experience H...	Microsoft Corporation
RuntimeBroker.exe	Susp...	10,864 K	0 K	4256	Runtime Broker	Microsoft Corporation
ApplicationFrameHost...		3,976 K	3,120 K	5548	Application Frame Host	Microsoft Corporation
svchost.exe		4,532 K	3,848 K	804	Host Process for Windows S...	Microsoft Corporation
svchost.exe	< 0.01	6,960 K	15,116 K	976	Host Process for Windows S...	Microsoft Corporation
MicrosoftEdgeUpdate...	< 0.01	34,312 K	31,376 K	4992	Microsoft Edge Update	Microsoft Corporation
taskhostw.exe		2,764 K	44 K	6904	Host Process for Windows T...	Microsoft Corporation
svchost.exe		3,356 K	8,524 K	984	Host Process for Windows S...	Microsoft Corporation
svchost.exe	< 0.01	13,632 K	9,720 K	1012	Host Process for Windows S...	Microsoft Corporation
svchost.exe	< 0.01	16,472 K	24,036 K	352	Host Process for Windows S...	Microsoft Corporation
svchost.exe	3.08	69,748 K	63,552 K	356	Host Process for Windows S...	Microsoft Corporation
svchost.exe	< 0.01	12,608 K	13,336 K	1120	Host Process for Windows S...	Microsoft Corporation
VBxService.exe		14,752 K	22,800 K	1308	VirtualBox Guest Additions S...	Oracle and/or its affiliates
svchost.exe		2,292 K	6,280 K	6036	Host Process for Windows S...	Microsoft Corporation
Ransomware.wannacry.e...		1,612 K	672 K	6556	Microsoft® Disk Defragmenter	Microsoft Corporation
svchost.exe	< 0.01	24,872 K	10,656 K	3400	Host Process for Windows S...	Microsoft Corporation
lsass.exe		1,116 K	5,016 K	584	Local Security Authority Proc...	Microsoft Corporation
fontdrvhost.exe	3.08	1,720 K	6,760 K	696	Usemode Font Driver Host	Microsoft Corporation
csrss.exe	< 0.01	6,136 K	16,412 K	492		
winlogon.exe		1,272 K	2,676 K	552	Windows Logon Application	Microsoft Corporation
fontdrvhost.exe	< 0.01	2,008 K	5,428 K	688	Usemode Font Driver Host	Microsoft Corporation

Figure 15: Runs as a service.



Process Name	Private Bytes	Working Set	Virtual Bytes	Session ID	Company Name
Ransomware.wannacry.exe	< 0.01	23,812 K	21,000 K	3696	Microsoft Corporation
tasksche.exe	0.31	18,488 K	25,880 K	3604	DiskPart
@WanaDecryptor@.exe	1,764 K	8,940 K	3008	3196	Load PerfMon Counters
taskhsvc.exe			80 K	3976	Console Window Host
conhost.exe			20 K	4276	Microsoft Volume Shadow ...
VSSVC.exe			64 K	3232	Load PerfMon Counters
@WanaDecryptor@.exe				72	
Registry		20,916 K	26,872 K		

Command Line: C:\ProgramData\ugitxayznch118\tasksche.exe
Path: C:\ProgramData\ugitxayznch118\tasksche.exe

Figure 16: Additional executables dropped in C:\ProgramData folder

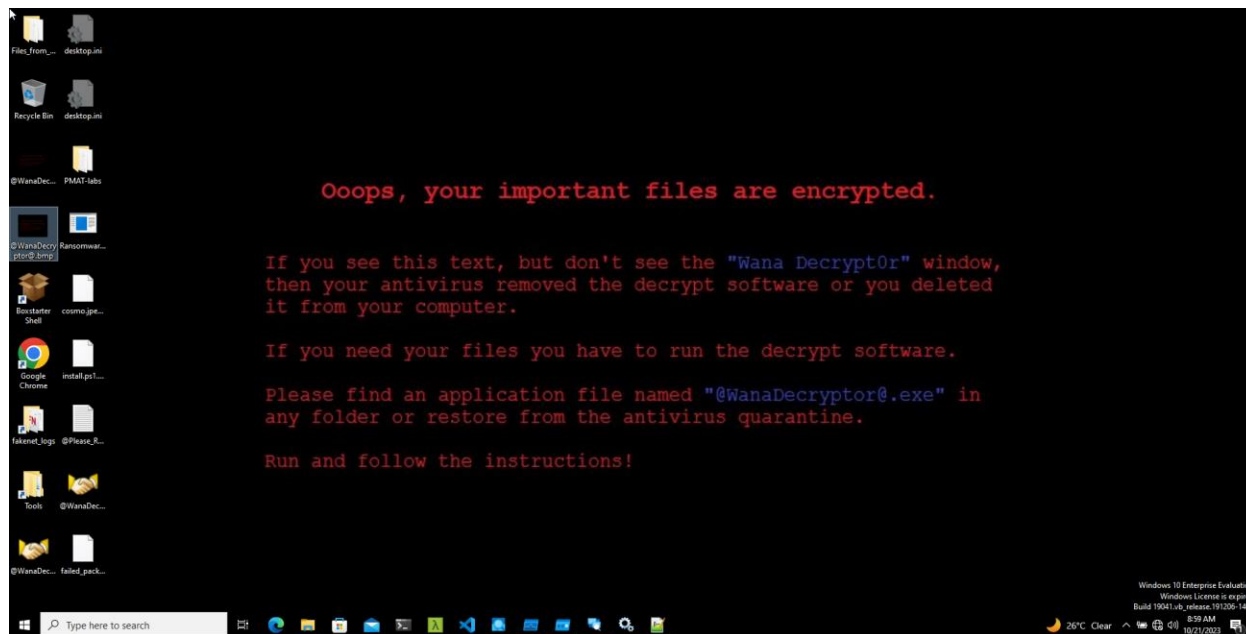


Figure 17: Desktop wallpaper changed, and all files encrypted.



The folder “C:\ProgramData\ugjtxayznch118” contains an executable called “@WanaDecryptor@.exe” which will execute every time and show the pop-up below.

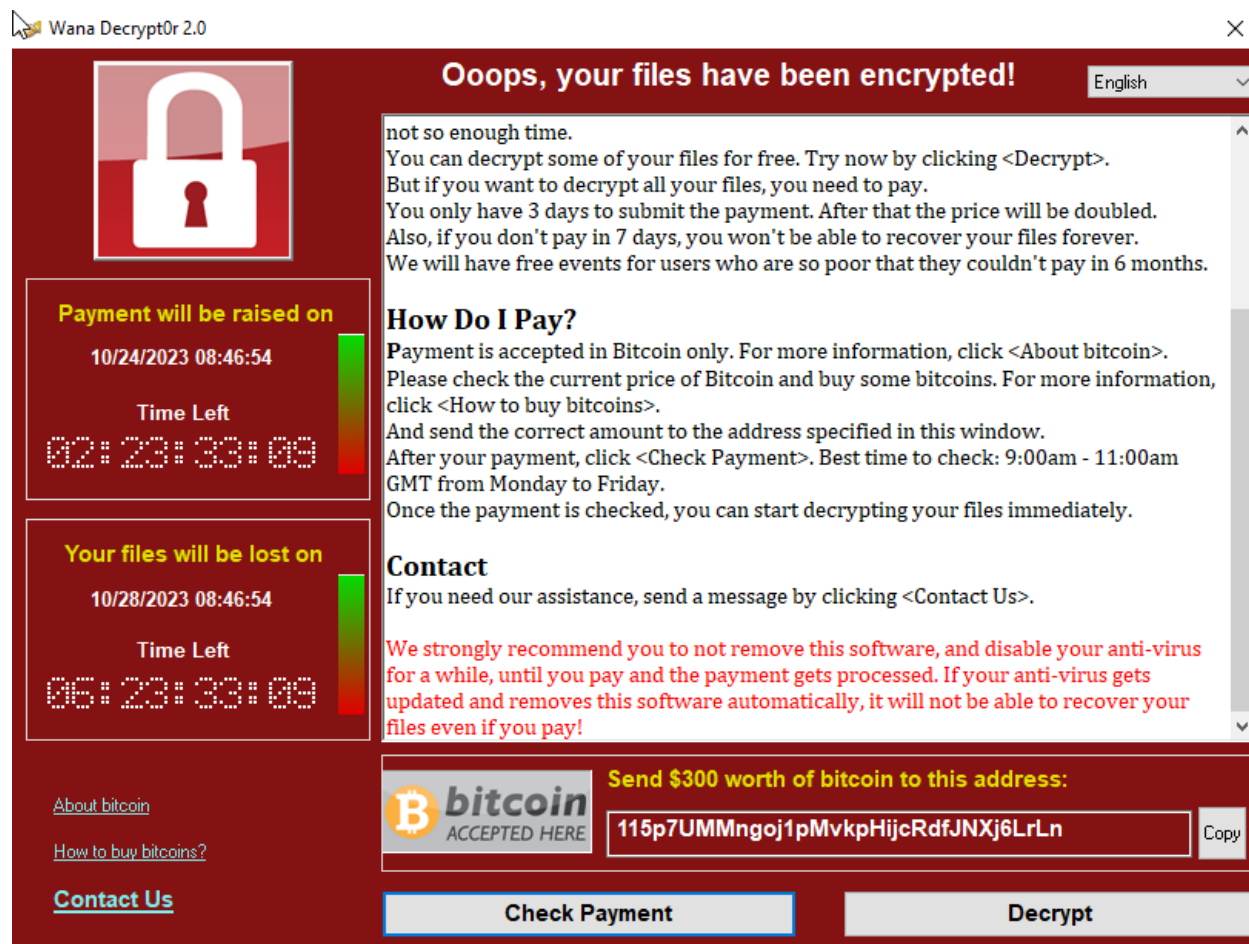


Figure 18: Wannacry Decryptor

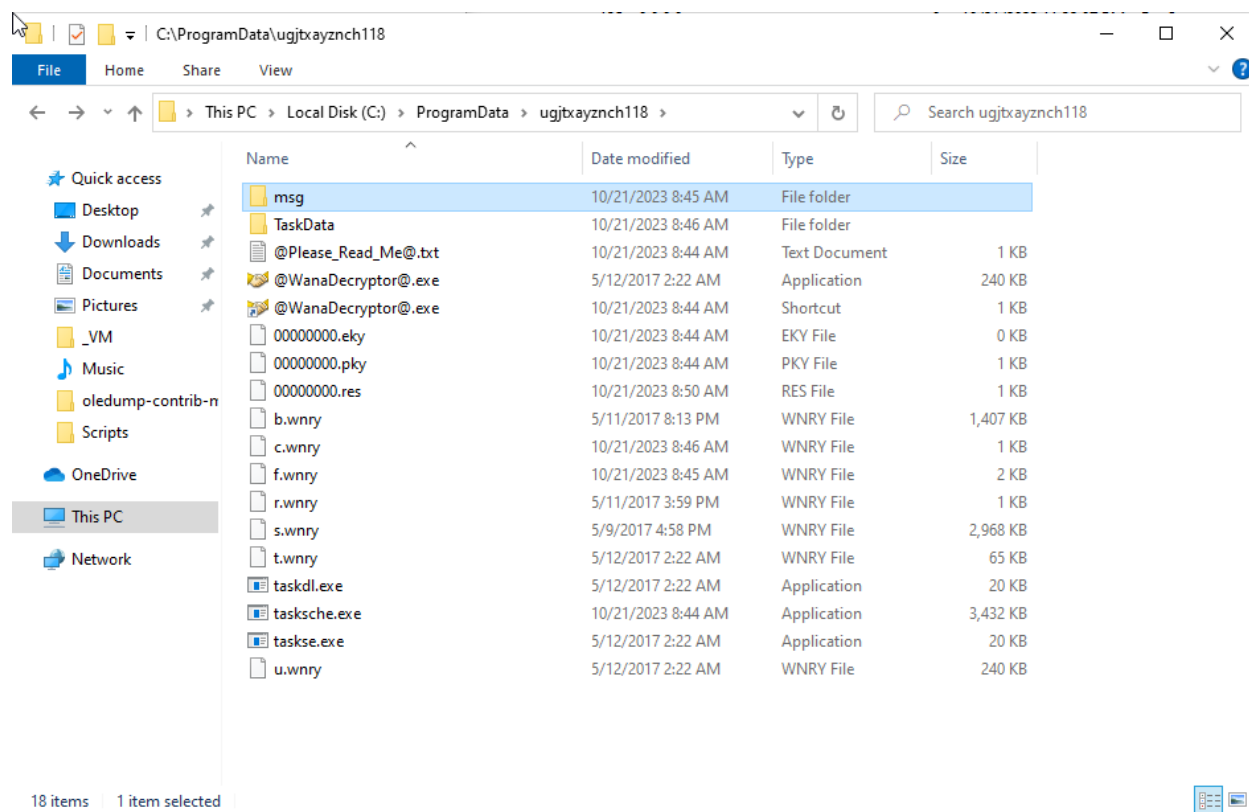


Figure 19: Contents of the hidden folder created under C:\ProgramData

Contains two directories named “msg” and “TaskData”. Other files inside this folder.

- @Please_Read_Me@.txt - a read me text file presented to the victim which contains instructions on how to decrypt their files.
- @WanaDecryptor@.exe - Decryptor of the files.
- .eky
- .pky
- .res
- b.wnry
- c.wnry



File Explorer window showing the folder `C:\ProgramData\ugitxayznch118`. The file `c.wnry` is selected.

Search results for `c.wnry`:

```
File: c.wnry
MD5: 1ed2ead8f8f969a0daf2686d6cb73fd4
Size: 780

Ascii Strings:
-----
000000B2 115p7UMMngoj1pMvKpHijcRdfJNXj6LrLn
000000E4
gx7ekbenv2riucmf.onion;57g7spgrzlojinas.onion;xxlvbrloxvriy2c5.onion;76jdd2ir2embyv47.onion;cwwnhwhlz52maqm7.onion;
000001DE https://dist.torproject.org/torbrowser/6.5.1/tor-win32-0.2.9.10.zip

Unicode Strings:
-----
```

Figure 20: Onion links found inside c.wnry

File: c.wnry
MD5: 1ed2ead8f8f969a0daf2686d6cb73fd4
Size: 780

Ascii Strings:

```
-----
000000B2 115p7UMMngoj1pMvKpHijcRdfJNXj6LrLn
000000E4
gx7ekbenv2riucmf.onion;57g7spgrzlojinas.onion;xxlvbrloxvriy2c5.onion;76jdd2ir2embyv47.onion;cwwnhwhlz52maqm7.onion;
000001DE https://dist.torproject.org/torbrowser/6.5.1/tor-win32-0.2.9.10.zip
Unicode Strings:
-----
```

- f.wnry
- r.wnry
- s.wnry



- t.wnry
- taskdl.exe
- tasksche.exe
- taskse.exe
- u.wnry

```
1963 matches found... C:\ProgramData\ugjbxayznch118\wnry
Find All Save As Min Size Rescan Scan All Differs C:\ugjbxayznch118\wnry File Results More
00020C88 CryptAcquireContextA
00020FC8 %s %s
00020F00 cmd.exe
00020F08 /c vssadmin delete shadows /all /quiet & wmic shadowcopy delete & bcdedit /set {default} bootstatuspolicy ignoreallfailures & bcdedit /set {default} recoveryenabled no & wadmin delete catalog -quiet
000210AC 13AM4VW2dhwgKeQepcRkH9uyGNgkEb94
000210D0 English
000210D8 m_ha.wnry
000210E4 msg\
000210EC <https://
000210F8 <https://
00021104 %d/%d/%d %02d:%02d:%02d
0002111C 00:00:00:00
00021128 https://www.btcfrog.com/qr/bitcoinPNG.php?address=%s
0002115C mailto:%s
00021168 https://www.google.com/search?q=howto+buy+bitcoin
0002119C https://en.wikipedia.org/wiki/Bitcoin
000211C4 Send %i if BTC to this address:
000211E4 %i if BTC
000211F0 Send %d worth of bitcoin to this address:
00021220 %02d:%02d:%02d:%02d
0002123C b.wnry
00021255 %i64
00021260 Failed to send your message!
0002127D Please make sure that your computer is connected to the Internet and
000212C3 your Internet Service Provider (ISP) does not block connections to the TOR Network!
00021318 Your message has been sent successfully!
00021344 You are sending too many mails! Please try again %d minutes later.
00021388 Too short message!
0002139C %d%%
-----
```

Figure 21: Deletion of volume shadow copies.

`cmd.exe /c vssadmin delete shadows /all /quiet & wmic shadowcopy delete & bcdedit /set {default} bootstatuspolicy ignoreallfailures & bcdedit /set {default} recoveryenabled no & wadmin delete catalog -quiet`

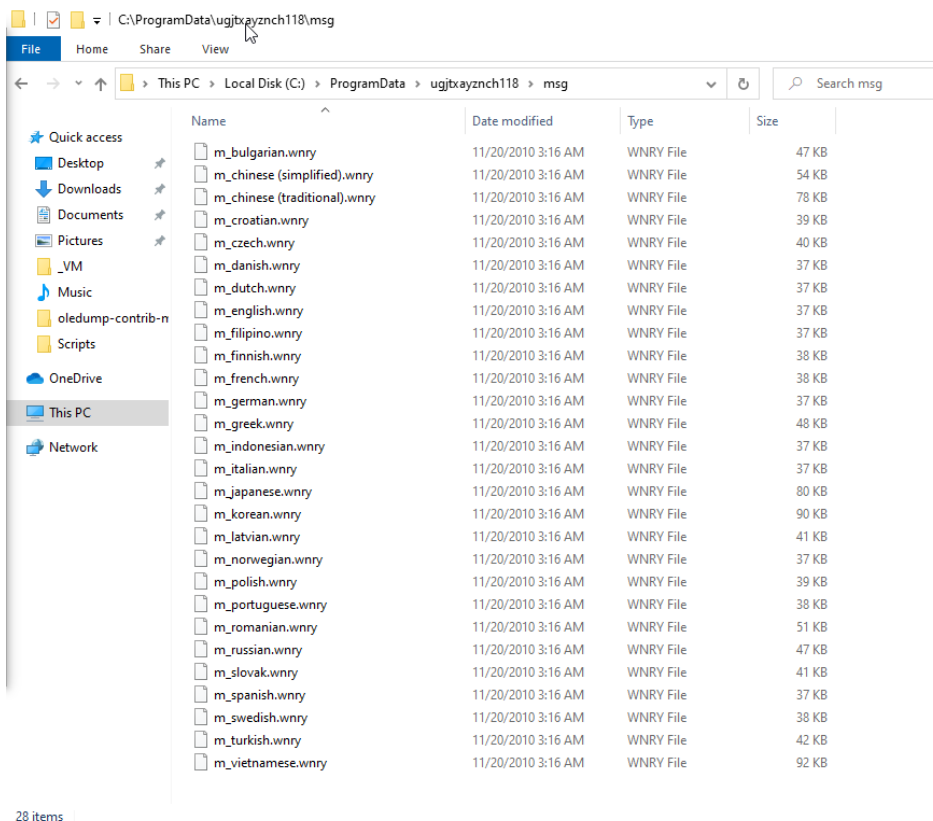


Figure 22: “msg” folder contains all other supported languages possibly used by the malware.

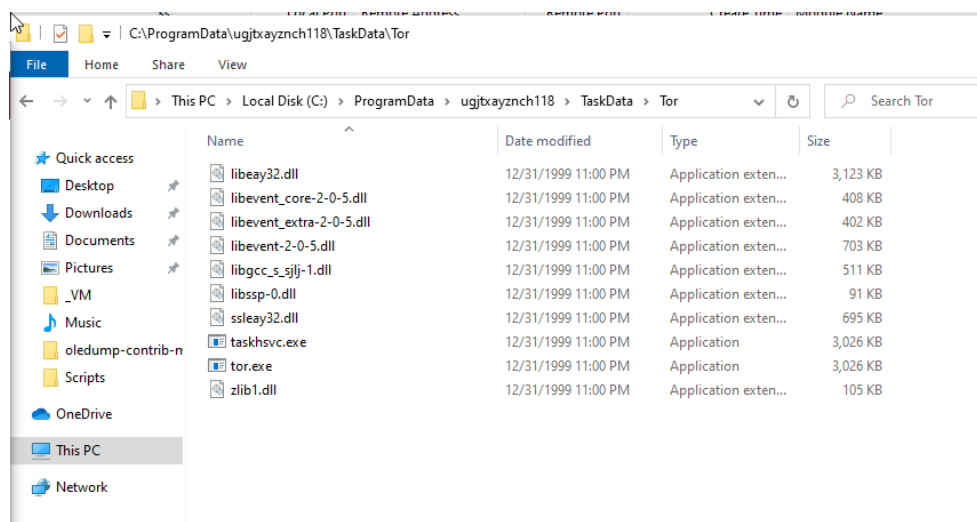


Figure 23: TaskData folder contains Tor folder. It contains libraries possibly required by the Tor browser and taskhsvc.exe.

Process Name	Process ID	Protocol	State	Local Address	Local Port	Remote Address	Remote Port	Create Time	Module Name
svchost.exe	804	TCP	Listen	0.0.0.0	135	0.0.0.0	0	10/21/2023 11:32:07 PM	RpcSs
System	4	TCP	Listen	10.0.0.3	139	0.0.0.0	0	10/24/2023 12:35:39 PM	System
svchost.exe	356	TCP	Listen	0.0.0.0	5040	0.0.0.0	0	10/21/2023 8:34:13 AM	CDPSvc
lsass.exe	584	TCP	Listen	0.0.0.0	49664	0.0.0.0	0	10/21/2023 11:32:07 PM	lsass.exe
wininit.exe	484	TCP	Listen	0.0.0.0	49665	0.0.0.0	0	10/21/2023 11:32:07 PM	wininit.exe
svchost.exe	1012	TCP	Listen	0.0.0.0	49666	0.0.0.0	0	10/21/2023 11:32:08 PM	EventLog
svchost.exe	976	TCP	Listen	0.0.0.0	49667	0.0.0.0	0	10/21/2023 11:32:08 PM	Schedule
spoolsv.exe	1888	TCP	Listen	0.0.0.0	49668	0.0.0.0	0	10/21/2023 8:32:12 AM	Spooler
services.exe	576	TCP	Listen	0.0.0.0	49669	0.0.0.0	0	10/21/2023 8:32:13 AM	services.exe
svchost.exe	1200	TCP	Listen	0.0.0.0	49670	0.0.0.0	0	10/21/2023 8:32:15 AM	PolicyAgent
Ransomware.wannacr...	5844	TCP	Syn Sent	10.0.0.3	49811	10.0.0.98	445	10/24/2023 12:36:30 PM	mssecsv2.0
Ransomware.wannacr...	5844	TCP	Syn Sent	10.0.0.3	49812	10.0.0.99	445	10/24/2023 12:36:30 PM	mssecsv2.0
Ransomware.wannacr...	5844	TCP	Syn Sent	10.0.0.3	49814	10.0.0.100	445	10/24/2023 12:36:30 PM	mssecsv2.0
Ransomware.wannacr...	5844	TCP	Syn Sent	10.0.0.3	49815	10.0.0.101	445	10/24/2023 12:36:30 PM	mssecsv2.0
Ransomware.wannacr...	5844	TCP	Syn Sent	10.0.0.3	49816	10.0.0.102	445	10/24/2023 12:36:30 PM	mssecsv2.0
Ransomware.wannacr...	5844	TCP	Syn Sent	10.0.0.3	49818	10.0.0.103	445	10/24/2023 12:36:30 PM	mssecsv2.0
Ransomware.wannacr...	5844	TCP	Syn Sent	10.0.0.3	49820	10.0.0.104	445	10/24/2023 12:36:30 PM	mssecsv2.0
Ransomware.wannacr...	5844	TCP	Syn Sent	10.0.0.3	49821	10.0.0.105	445	10/24/2023 12:36:31 PM	mssecsv2.0
Ransomware.wannacr...	5844	TCP	Syn Sent	10.0.0.3	49824	10.0.0.106	445	10/24/2023 12:36:31 PM	mssecsv2.0
System	4	TCP	Listen	0.0.0.0	445	0.0.0.0	0	10/21/2023 8:32:12 AM	System
svchost.exe	1752	TCP	Listen	0.0.0.0	7680	0.0.0.0	0	10/21/2023 8:32:12 AM	DoSvc
svchost.exe	804	TCPv6	Listen	::	135	::	0	10/21/2023 11:32:07 PM	RpcSs
System	4	TCPv6	Listen	::	445	::	0	10/21/2023 8:32:12 AM	System
svchost.exe	1752	TCPv6	Listen	::	7680	::	0	10/21/2023 8:32:12 AM	DoSvc

Figure 24: SMB port scan on the network which is caused by Eternablu exploit.

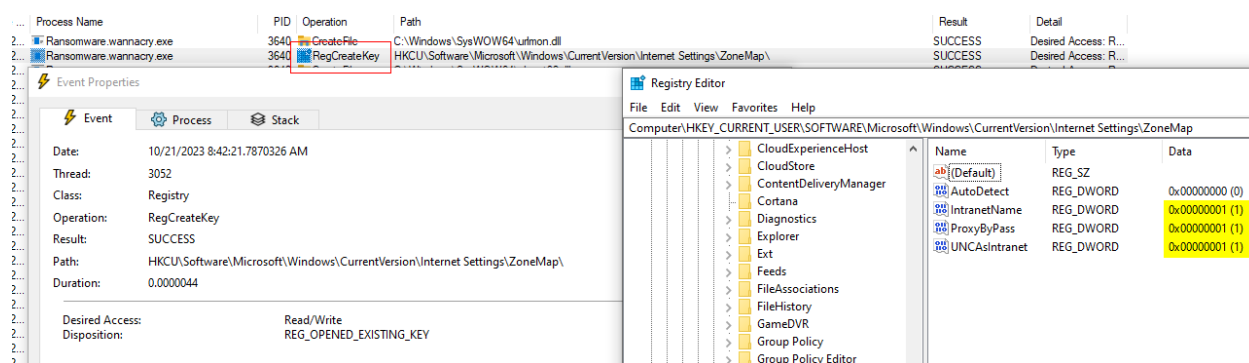


Figure 25: Registry key creation and modification in ZoneMap.

Setting the registry values ProxyBypass, IntranetName, and UNCAsIntranet to 1 in the ZoneMap key (HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\)) can have significant implications for how Internet Explorer handles various types of network traffic. Here's what each of these settings typically means:

ProxyBypass:

When set to 1, this may indicate that the malware is configuring Internet Explorer to bypass any configured proxy servers for certain addresses or domains. This means that network traffic to those specific addresses or domains will not go through the proxy server but will instead be sent directly.

IntranetName:

When set to 1, this likely implies that the malware is designating a particular domain or address as belonging to the local intranet. This can impact how Internet Explorer treats content from this domain, potentially granting it more permissive security settings associated with local intranet content.

UNCAsIntranet:

This setting indicates whether Universal Naming Conventions (UNC) paths should be treated as if they belong to the local intranet. Setting this to 1 suggests that UNC paths will be treated as if they belong to the intranet zone.



Run as Administrator (inetsim/fake-net enabled)

Ransomware did not execute. We are suspecting this was due to the kill switch URL.

Run as Administrator (inetsim enabled) | Remote Server setup

Process Name	Process ID	Protocol	State	Local Address	Local Port	Remote Address	Remote Port	Create Time	Module Name
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1980	221.49.152.225	445	10/28/2023 8:37:37 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1880	216.56.46.7	445	10/28/2023 8:37:32 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1973	215.50.107.119	445	10/28/2023 8:37:37 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1868	212.239.61.226	445	10/28/2023 8:37:32 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1968	212.197.234.103	445	10/28/2023 8:37:36 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1876	202.149.27.121	445	10/28/2023 8:37:32 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1867	199.234.176.172	445	10/28/2023 8:37:32 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1978	198.83.147.117	445	10/28/2023 8:37:37 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1877	188.211.192.244	445	10/28/2023 8:37:32 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1887	185.32.24.93	445	10/28/2023 8:37:32 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1982	183.4.146.17	445	10/28/2023 8:37:37 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1883	181.59.178.103	445	10/28/2023 8:37:32 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1886	173.206.252.112	445	10/28/2023 8:37:32 PM	mssecsv2.0
System	4	TCP	Established	172.16.99.2	1113	172.16.99.5	445	10/28/2023 8:32:54 PM	System
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1970	161.165.102.214	445	10/28/2023 8:37:36 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1872	146.68.81.219	445	10/28/2023 8:37:32 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1879	143.166.90.74	445	10/28/2023 8:37:32 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1977	140.136.147.124	445	10/28/2023 8:37:37 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1986	140.66.57.0	445	10/28/2023 8:37:37 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1888	125.133.199.171	445	10/28/2023 8:37:32 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1988	116.47.59.247	445	10/28/2023 8:37:37 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1967	114.67.142.127	445	10/28/2023 8:37:36 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1976	113.241.233.150	445	10/28/2023 8:37:37 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1987	113.168.101.100	445	10/28/2023 8:37:37 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1885	99.238.162.25	445	10/28/2023 8:37:32 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1985	98.203.26.211	445	10/28/2023 8:37:37 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1969	98.45.23.81	445	10/28/2023 8:37:36 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1878	98.15.206.238	445	10/28/2023 8:37:32 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1869	92.236.20.112	445	10/28/2023 8:37:32 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1873	92.224.203.54	445	10/28/2023 8:37:32 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1966	87.221.39.158	445	10/28/2023 8:37:36 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1875	80.84.172.226	445	10/28/2023 8:37:32 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1881	80.1.154.129	445	10/28/2023 8:37:32 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1974	78.140.173.232	445	10/28/2023 8:37:37 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1984	51.8.190.187	445	10/28/2023 8:37:37 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1975	45.174.178.177	445	10/28/2023 8:37:37 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1971	44.3.48.234	445	10/28/2023 8:37:36 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1889	37.183.94.95	445	10/28/2023 8:37:32 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1871	31.56.173.206	445	10/28/2023 8:37:32 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1981	30.54.210.17	445	10/28/2023 8:37:37 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1979	27.237.187.60	445	10/28/2023 8:37:37 PM	mssecsv2.0
Ransomware.wannacr...	6556	TCP	Syn Sent	172.16.99.2	1884	27.99.187.90	445	10/28/2023 8:37:32 PM	mssecsv2.0

Figure 26: Established TCP connection to the remote testing server IP 172.16.99.5 and a TCP Scan on the network.

- Tried out some of the UNC paths acquired from strings and setup IPC. Configured the static IP 172.16.99.5 for a remote server.
- Client can establish connection to the remote server IP we've setup.
- However, the files on the remote server were not encrypted.
- Most likely, this is because the remote server is a Windows Server 2022 machine, and no downgrade of its security protection was performed.
- This was not tested further, and we're satisfied to just point out that this sample connects to port 445.



Advanced Static Analysis

FLOSS

```
45360 -----
45361 | FLOSS STACK STRINGS (17) |
45362 -----
45363 SMBu
45364 /K__USERID__PLACEHOLDER__
45365 __TREEPATH_REPLACE__
45366 PIPE
45367 SMBr
45368 PC NETWORK PROGRAM 1.0
45369 LANMAN1.0
45370 Windows for Workgroups 3.1a
45371 LM1.2X002
45372 LANMAN2.1
45373 NT LM 0.12
45374 SMBs
45375 SMB2
45376 Windows 2000 2195
45377 Windows 2000 5.0
45378 \\192.168.56.20\IPC$
45379 http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com
45380
45381 -----
45382 | FLOSS TIGHT STRINGS (0) |
45383 -----
45384
45385 -----
45386 | FLOSS DECODED STRINGS (7) |
45387 -----
45388 SMBu
45389 __TREEPATH_REPLACE__
45390 AWAVAUATSQRUWVPP
45391 QQjh
45392 t.M1
45393 XX^_]ZY[A\A]A^A_H
45394 SVQRH
45395
```

Figure 27: Acquired strings from FLOSS

- Observed SMB, LANMAN, NTLM protocols.
- We can see a private IP connection to IPC\$ which is a UNC (Universal Naming Convention) path used in networking on Windows systems. It's not a file or a folder, but rather a special administrative share used for Inter-Process Communication (IPC).
 - IPC\$: This is a hidden administrative share on a Windows machine that is used for communication between processes on a network. "IPC" stands for Inter-Process Communication. This share allows administrative tasks, such as remotely connecting to a computer's registry or performing management tasks.



- We can see the similar URL
hxxp[:]//www[.]iuqerfsodp9ifjaposdfjhgosurijfaewrrergwea[.]com observed from
pestudio.

STRINGS

```
λ strings Ransomware.wannacry.exe.malz | grep IPC
\\172.16.99.5\IPC$
\\192.168.56.20\IPC$
\\%s\IPC$
```

Figure 28: Additional IPC related strings using strings tool.

CAPA

capa Ransomware.wannacry.exe		
md5	db349b97c37d22f5ea1d1841e3c89eb4	
sha1	e889544aff85ffaf8b0d0da705105dee7c97fe26	
sha256	24d004a104d4d54034dbcffc2a4b19a11f39008a575aa614ea04703480b1022c	
os	windows	anti-analysis
format	pe	anti-analysis/anti-debug
arch	i386	anti-analysis/obfuscation
path	C:/Users/verticalhead/Desktop/Ransomware.wannacry.exe	
ATT&CK Tactic	ATT&CK Technique	
DEFENSE EVASION	Obfuscated Files or Information::Indicator Removal from Tools T1027.005	
DISCOVERY	File and Directory Discovery T1083 System Information Discovery T1082 System Network Configuration Discovery T1016	
EXECUTION	Shared Modules T1129 System Services::Service Execution T1569.002	
PERSISTENCE	Create or Modify System Process::Windows Service T1543.003	
MBC Objective	MBC Behavior	
ANTI-BEHAVIORAL ANALYSIS	Conditional Execution::Runs as Service [B0025.007] Debugger Detection::Timing/Delay Check QueryPerformanceCounter [B0001.033]	
ANTI-STATIC ANALYSIS	Executable Code Obfuscation::Argument Obfuscation [B0032.020] Executable Code Obfuscation::Stack Strings [B0032.017]	
COMMAND AND CONTROL	C2 Communication::Receive Data [B0030.002] C2 Communication::Send Data [B0030.001]	
COMMUNICATION	HTTP Communication::Create Request [C0002.012] HTTP Communication::Open URL [C0002.004] Socket Communication::Connect Socket [C0001.004] Socket Communication::Create TCP Socket [C0001.011] Socket Communication::Create UDP Socket [C0001.010] Socket Communication::Get Socket Status [C0001.012] Socket Communication::Initialize Winsock Library [C0001.009] Socket Communication::Receive Data [C0001.006] Socket Communication::Send Data [C0001.007] Socket Communication::Set Socket Config [C0001.001] Socket Communication::TCP Client [C0001.008]	

Figure 29: Summary of MITRE ATT&CK Tactics and Techniques and MBC (Malware Behavior Catalog) Objectives

Capa shows that this binary uses techniques such as defense evasion, discovery, execution and persistence. Malware Behavior Catalog (MBC) identifier showed the specific behaviors



observed. Some notable behaviors which may require further dynamic testing include the C2 send and receive data, HTTP and socket communication and the move file behavior.

CRYPTOGRAPHY	Generate Pseudo-random Sequence::Use API [C0021.003]	anti-analysis/obfuscation/communication
DATA	Compression Library [C0060]	communication
DISCOVERY	Analysis Tool Discovery::Process detection [B0013.001] Code Discovery::Inspect Section Memory Permissions [B0046.002] File and Directory Discovery [E1083]	communication/socket communication/socket communication/socket
EXECUTION	Install Additional Program [B0023]	communication/socket/ communication/tcp/client
FILE SYSTEM	Move File [C0063] via WinAPI Read File [C0051] via kernel32 functions	data-manipulation/prng executable/resource
PROCESS	Create Thread [C0038] Terminate Process [C0018] Terminate Thread [C0039]	host-interaction/file host-interaction/file host-interaction/file

Figure 30: MBC Objective

Capability	Namespace
reference analysis tools strings	anti-analysis
check for time delay via QueryPerformanceCounter	anti-analysis/anti-debugging/debugger-detection
contain obfuscated stackstrings	anti-analysis/obfuscation/string/stackstring
receive data (5 matches)	communication
send data (5 matches)	communication
connect to URL	communication/http/client
get socket status	communication/socket
initialize Winsock library	communication/socket
set socket configuration	communication/socket
create UDP socket (4 matches)	communication/socket/udp/send
act as TCP client	communication/tcp/client
generate random numbers via WinAPI	data-manipulation/prng
extract resource via kernel32 functions	executable/resource
contain an embedded PE file	executable/subfile/pe
get file size	host-interaction/file-system/meta
move file	host-interaction/file-system/move
read file on Windows	host-interaction/file-system/read-code/pe
get number of processors	host-interaction/hardware/cpu
terminate process	host-interaction/process/terminate
run as service	host-interaction/service
create service	host-interaction/service/create
modify service	host-interaction/service/modify
start service	host-interaction/service/start
create thread (4 matches)	host-interaction/thread/create
terminate thread	host-interaction/thread/terminate
link function at runtime on Windows	linking/runtime-linking
linked against ZLIB	linking/static/zlib
inspect section memory permissions	load-code/pe
persist via Windows service	persistence/service

Figure 31: Wannacry summarized capabilities

Capa in verbose mode showed a comprehensive list which included the addresses for easier cross reference when debugging. See the appendix section for the complete information.



Cutter



Figure 32: Graph of the main function located in 0x00408140



- We can see a call to API InternetOpenA using the string “str.http:___www.iuqerfsodp9ifjaposdfjhgosurijfaewrrergwea.com” as one of the passed arguments.
- If a connection is not made on the URL above, it will call the function “fcn.00408090”.
- Otherwise, it appears to clean up the stack and do a return instruction.
- We are suspecting that this URL is the kill switch. More on the advanced dynamic analysis section.

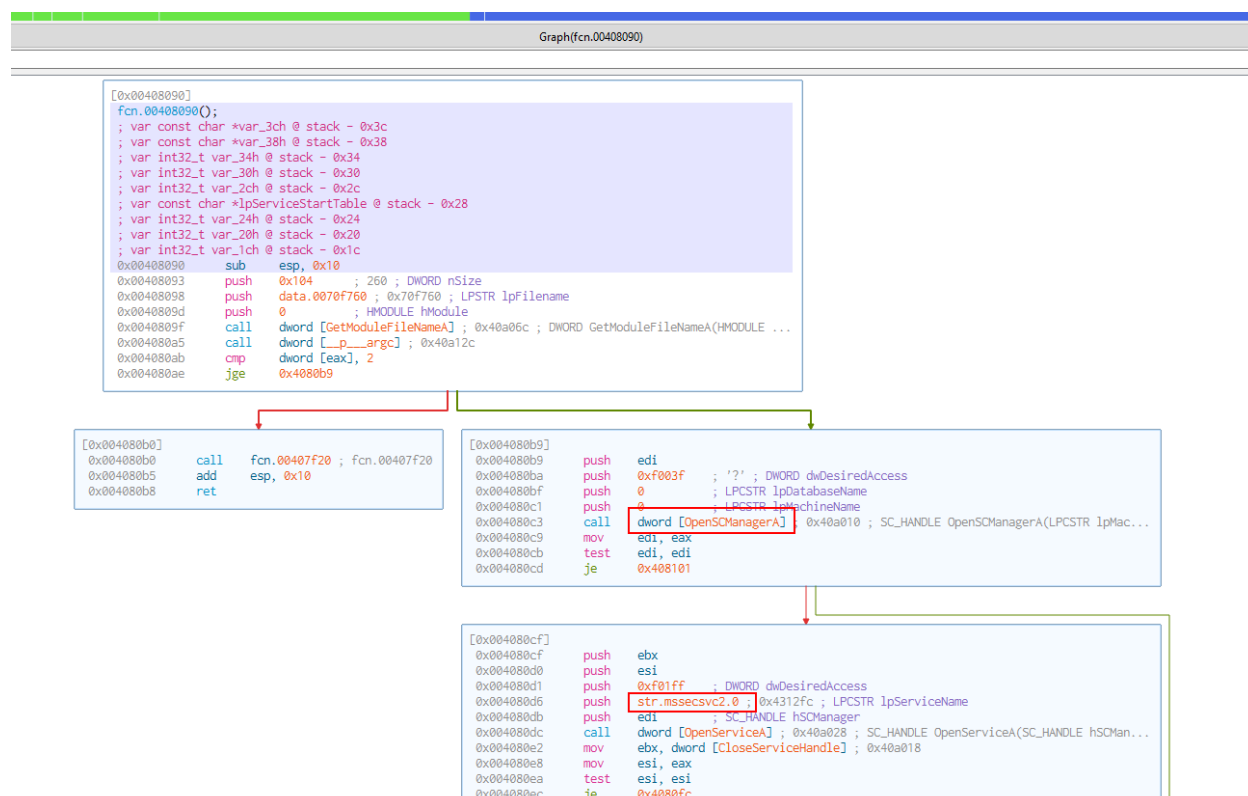


Figure 33: Graph of the “fcn.00408090” function which opens the service with name “mssecsvc2.0”.

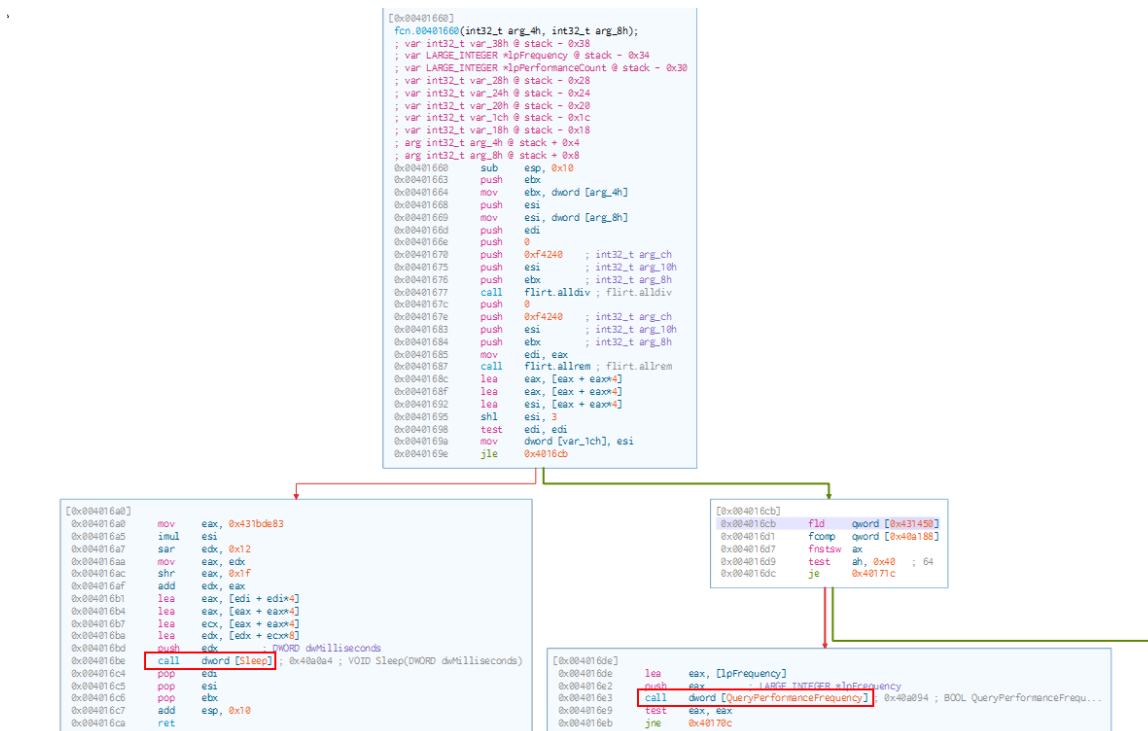


Figure 34: Possible time delay via QueryPerformanceCounter function located in 0x401660

```

        ...
lpPerformanceCount = (LARGE_INTEGER *)0x40167c;
arg_14h = arg_4h;
iVar1 = Flirt.alldiv(arg_4h, arg_8h, 1000000, 0, unaff_ESI, unaff_ESI);
lpPerformanceCount = (LARGE_INTEGER *)0x0;
iVar2 = Flirt.allrem(arg_4h, arg_8h, 1000000, 0, arg_14h);
iVar2 = iVar2 * 1000;
var_1ch = iVar2;
if (0 < iVar1) {
    (*KERNEL32.dll_Sleep)(iVar2 / 1000000 + iVar1 * 1000);
    return;
}
if ((*double *)0x431450 == 0.0) {
    iVar1 = (*KERNEL32.dll_QueryPerformanceFrequency)(&var_28h);
    if (iVar1 == 0) {
        (*KERNEL32.dll_Sleep)(iVar2 / 1000000);
        return;
    }
    *double *)0x431450 = (double)CONCAT44(var_24h, var_28h) * 1e-09;
}
uVar3 = sub.MSVCRT.dll_ftol();
var_18h = (int32_t)((uint64_t)uVar3 >> 0x20);
iVar1 = iVar2 / 1000000 + -10;
(*KERNEL32.dll_QueryPerformanceCounter)(&lpPerformanceCount);
iVar2 = (int32_t)lpPerformanceCount + (uint32_t)(0xffff0dbf < (uint32_t)uVar3) + var_1ch;
if (0 < iVar1) {
    (*KERNEL32.dll_Sleep)(iVar1);
}
(*KERNEL32.dll_QueryPerformanceCounter)(&var_20h);
if (var_20h <= iVar2) {
    if (var_20h < iVar2) goto code_r0x0040178b;
    do {
        if ((uint32_t)uVar3 + 1000000 <= (uint32_t)var_24h) {
            return;
        }
    } while (var_20h <= iVar2);
} while (var_20h <= iVar2);
}
return;
code_r0x0040178b:
do {
    (*KERNEL32.dll_QueryPerformanceCounter)(&var_24h);
} while (var_20h <= iVar2);
} while (var_20h <= iVar2);
}
return;
}
```

These seem to be custom functions related to integer division and remainder calculations.

It will use kernel32.dll to sleep for a calculated duration.

Figure 35: It implements a time delay or sleep function based on the system's performance counter.



```
act as TCP client
namespace communication/tcp/client
scope function
matches 0x407480
```

According to capa result above, a TCP client appears to be being set up in memory address 0x407480.

Graph(fcn.00407480)



Figure 36: Setting up socket connection.



Checking in cutter, this function appears to be involved in setting up a socket, performing an ioctlsocket operation, connecting to a destination, using select for monitoring events on the socket, and closing the socket.

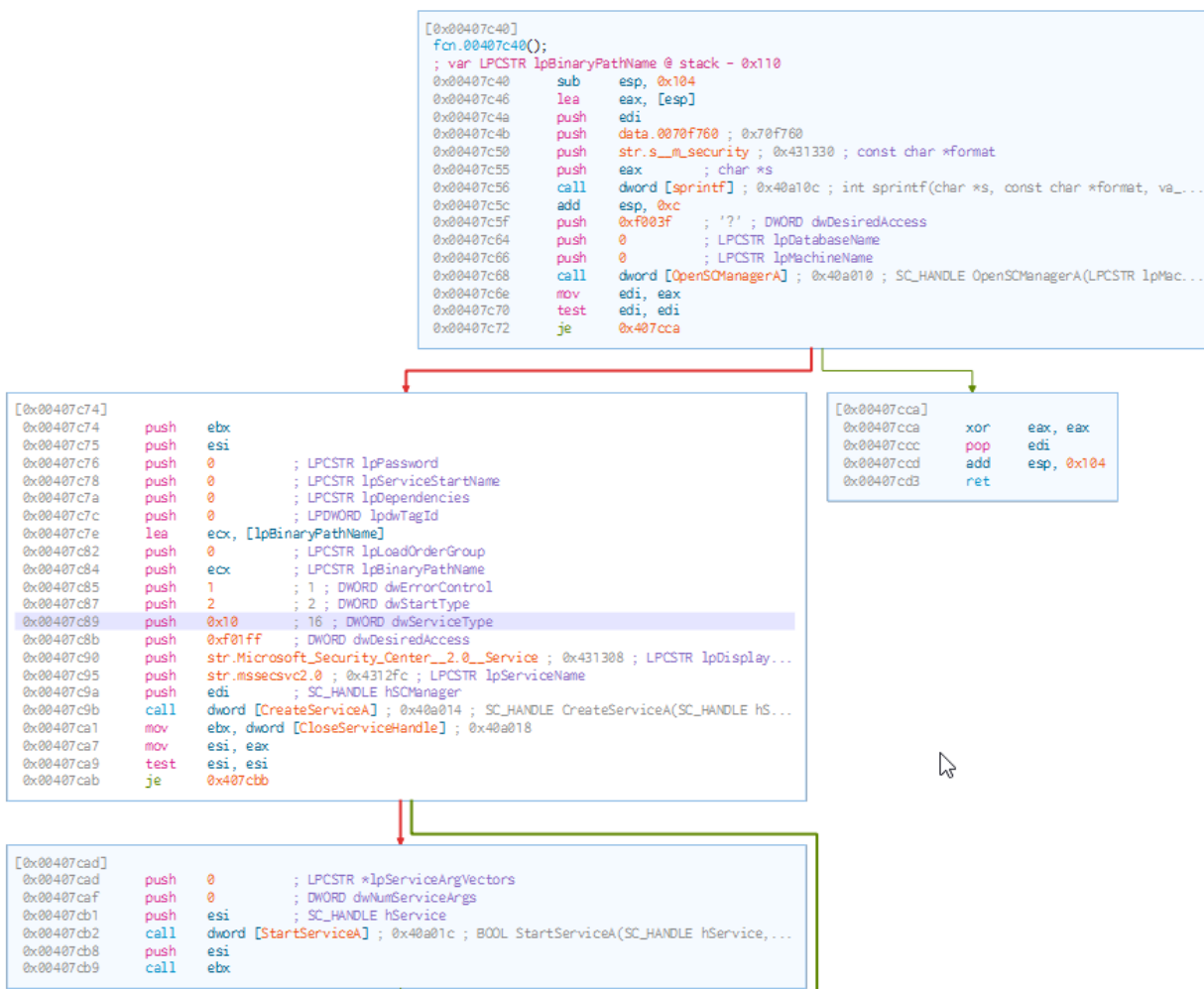


Figure 37: Service creation and service start for persistence.

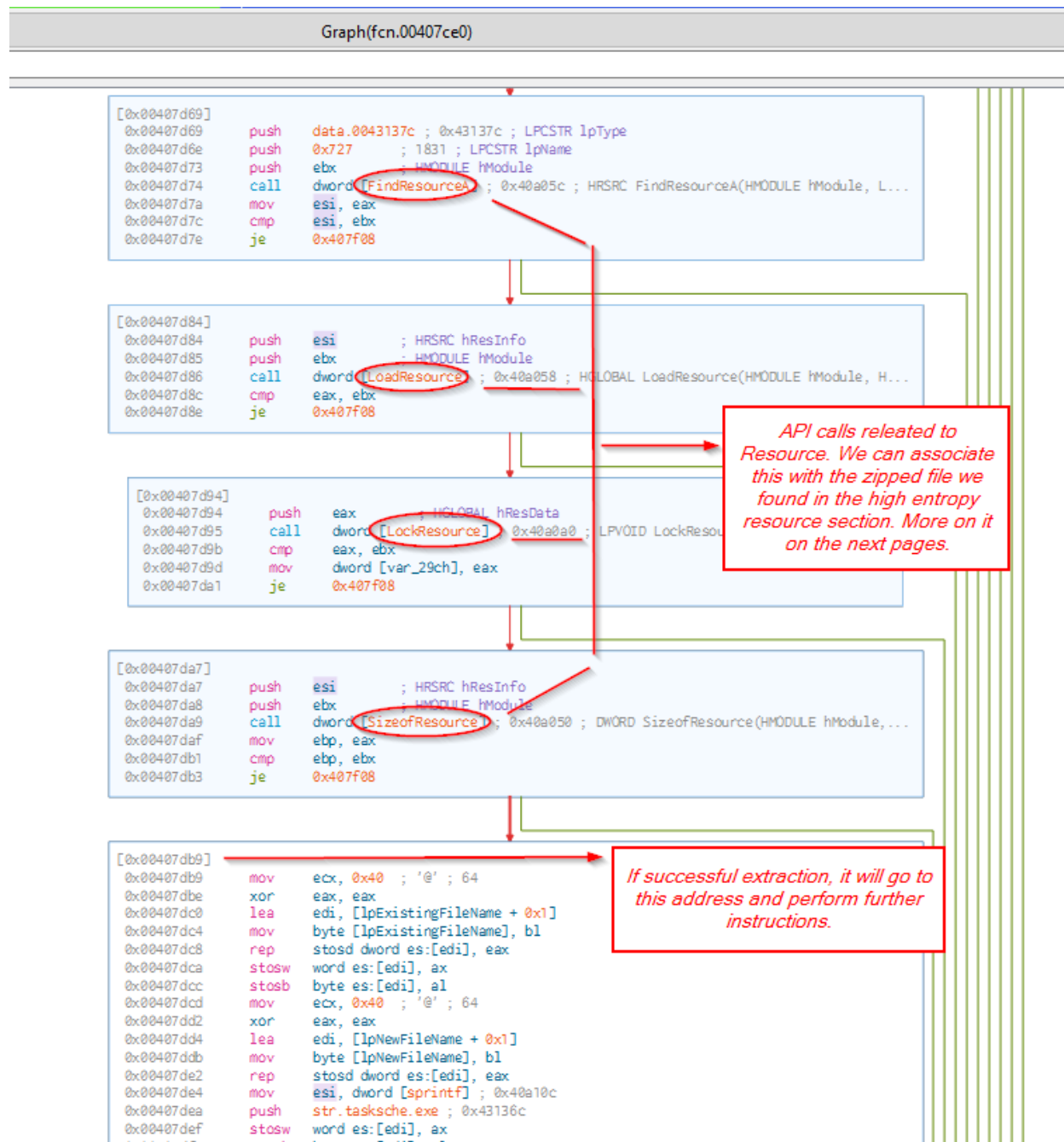


Figure 38: API calls referencing "resource". To be discussed on the next pages.



Graph(fcn.00407ce0)

```
[0x00407db9]
0x00407db9  mov     ecx, 0x40 ; '@' ; 64
0x00407dbe  xor     eax, eax
0x00407dc0  lea     edi, [lpExistingFileName + 0x1]
0x00407dc4  mov     byte [lpExistingFileName], bl
0x00407dc8  rep     stosd dword es:[edi], eax
0x00407dca  stosw   word es:[edi], ax
0x00407dcc  stosb   byte es:[edi], al
0x00407dcd  mov     ecx, 0x40 ; '@' ; 64
0x00407dd2  xor     eax, eax
0x00407dd4  lea     edi, [lpNewFileName + 0x1]
0x00407ddb  mov     byte [lpNewFileName], bl
0x00407de2  rep     stosd dword es:[edi], eax
0x00407de4  mov     esi, dword [sprintf], 0x40a10c
0x00407dea  push    str.tasksche.exe ; 0x43136c
0x00407def  stosw   word es:[edi], ax
0x00407df1  stosb   byte es:[edi], al
0x00407df2  push    str.WINDOWS ; 0x431364
0x00407df7  lea     eax, [lpExistingFileName]
0x00407dfb  push    str.C:___s ; 0x431358
0x00407e00  push    eax
0x00407e01  call    esi
0x00407e03  add     esp, 0x10
0x00407e06  lea     ecx, [lpNewFileName]
0x00407e0d  push    str.WINDOWS ; 0x431364
0x00407e12  push    str.C:___s_qeriuwjhrf ; 0x431344
0x00407e17  push    ecx
0x00407e18  call    esi
0x00407e1a  add     esp, 0xc
0x00407e1d  lea     edx, [lpNewFileName]
0x00407e24  lea     eax, [lpExistingFileName]
0x00407e28  push    1 ; 1 ; DWORD dwFlags
0x00407e2a  push    edx ; LPCSTR lpNewFileName
0x00407e2b  push    eax ; LPCSTR lpExistingFileName
0x00407e2c  call    dword [MoveFileExA] ; 0x40a04c ; BOOL MoveFileExA(LPCSTR lpExistingFileNa...
0x00407e32  push    ebx
0x00407e33  push    4 ; 4
0x00407e35  push    2 ; 2
0x00407e37  push    ebx
0x00407e38  push    ebx
0x00407e39  lea     ecx, [var_258h]
0x00407e3d  push    0x40000000
0x00407e42  push    ecx
0x00407e43  call    dword [data.00431458] ; 0x431458
0x00407e49  mov     esi, eax
0x00407e4b  cmp     esi, 0xffffffff
0x00407e4e  je      0x407f08
```

Figure 39: Possible result is "C:\WINDOWS\qeriuwjhrf".

The string "C:%s\qeriuwjhrf" is most likely used as a format string in a call to the sprintf function.



Investigation of the high entropy resource section

R.bin extracted using Resource hacker.

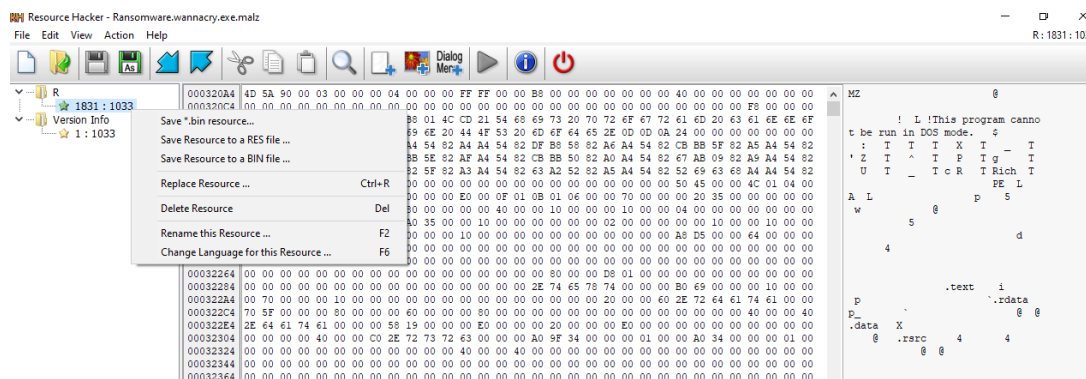


Figure 40: Extracting R using Resource hacker tool and saving it as R.bin.

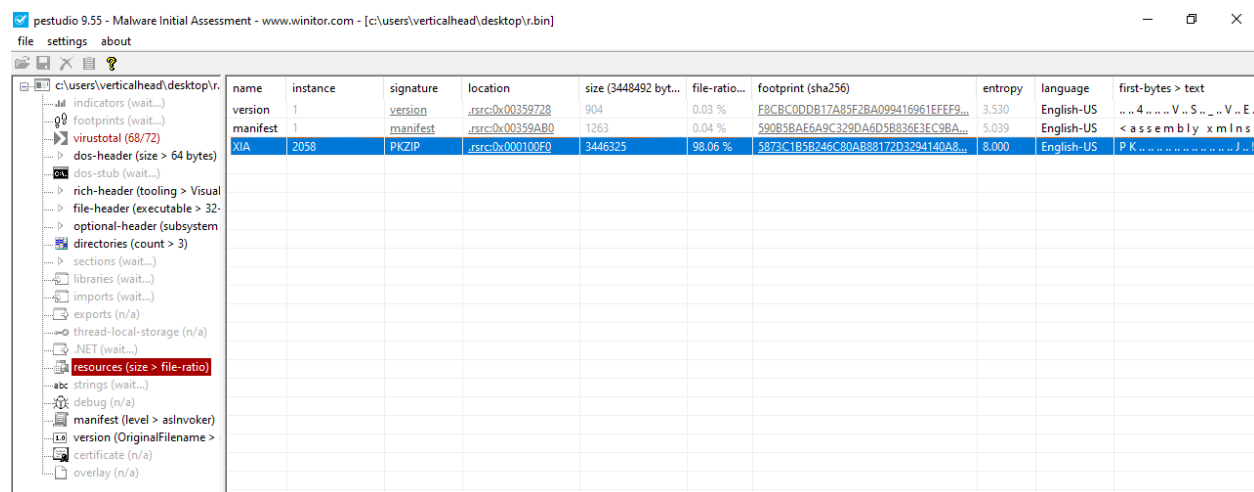


Figure 41: Another resource named "XIA" with PK header which indicates this is a zipped file.

Opening r.bin in pestudio shows another resource named.

Extract XIA using resource hacker and named it as XIA2058.zip.

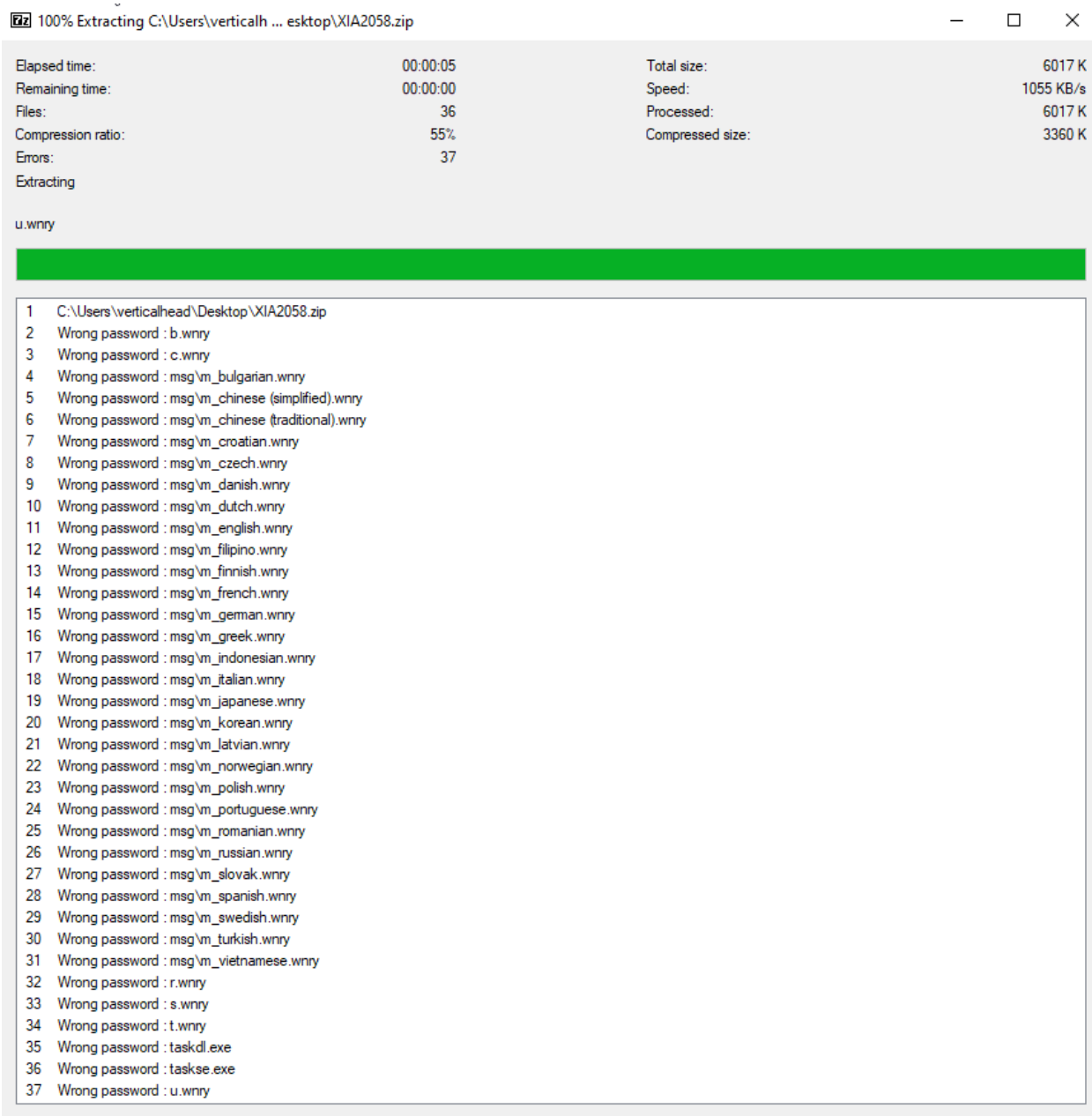


Figure 42: XIA password protected

Tried to extract using random password. We can see the contents but unable to extract since it requires a password.

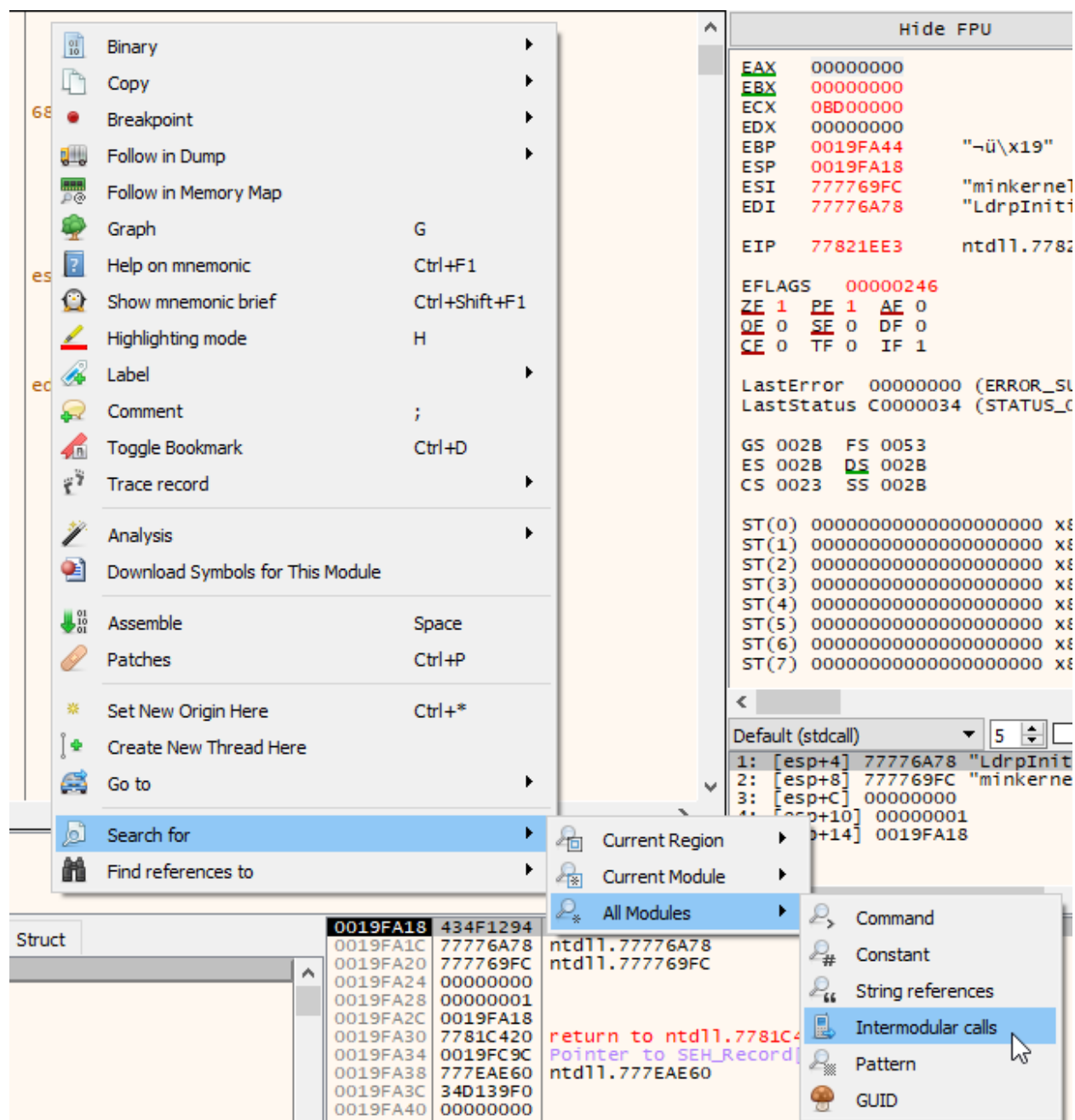


Figure 43: Search for Intermodular calls in r.bin using x32dbg.

Open R.bin in x32dbg.
Search for Intermodular calls.
We'll see it listed on the screenshot below.



CPU Log Notes Breakpoints Memory Map				
All Modules (Calls)				
Address	Disassembly			
00401DC3	call dword ptr ds:[<&FindResourceA>]			
00401DD3	call dword ptr ds:[<&LoadResource>]			
00401DDE	call dword ptr ds:[<&LockResource>]			
00401DF1	call dword ptr ds:[<&SizeofResource>]			
73D94AC9	call dword ptr ds:[<&FindResourceW>]			
73D94AE0	call dword ptr ds:[<&LoadResource>]			
73D94B06	call dword ptr ds:[<&SizeofResource>]			

Figure 44: API calls referencing resource

We know there is a “resource” inside this r.bin binary that is zipped and needs to be unzipped, so we look for any API calls referencing “resource”.

Under References tab > All Modules (Calls) we can find string “Resource” after searching for Intermodular calls.

D9E	FF15 0C804000	call dword ptr ds:[<&CloseServiceHan	esi:"minkernel\\ntdll\\ldrinit.c"
DA4	8BC6	mov eax,esi	esi:"minkernel\\ntdll\\ldrinit.c"
DA6	5E	pop esi	
DA7	5B	pop ebx	
DA8	5F	pop edi	edi:"LdrpInitializeProcess"
DA9	C9	leave	
DAA	C3	ret	
DAB	55	push ebp	
DAC	8BEC	mov ebp,esp	
DAE	81EC 2C010000	sub esp,12C	
DB4	56	push esi	esi:"minkernel\\ntdll\\ldrinit.c"
DB5	57	push edi	edi:"LdrpInitializeProcess"
DB6	68 3CF44000	push r.40F43C	40F43C:"XIA"
DB8	68 0A080000	push 80A	
DC0	FF75 08	push dword ptr ss:[ebp+8]	
DC3	FF15 00814000	call dword ptr ds:[<&FindResourceA>]	
DC9	8BF0	mov esi,eax	esi:"minkernel\\ntdll\\ldrinit.c"
DCB	85F6	test esi,esi	esi:"minkernel\\ntdll\\ldrinit.c"
DCD	74 38	je r.401E07	
DCF	56	push esi	esi:"minkernel\\ntdll\\ldrinit.c"
DD0	FF75 08	push dword ptr ss:[ebp+8]	
DD3	FF15 74804000	call dword ptr ds:[<&LoadResource>]	
DD9	85C0	test eax,eax	
DDB	74 2A	je r.401E07	
DDD	50	push eax	
DDE	FF15 70804000	call dword ptr ds:[<&LockResource>]	
DE4	8BF8	mov edi,eax	edi:"LdrpInitializeProcess"
DE6	85FF	test edi,edi	edi:"LdrpInitializeProcess"
DE8	74 1D	je r.401E07	
DEA	FF75 0C	push dword ptr ss:[ebp+c]	
DED	56	push esi	esi:"minkernel\\ntdll\\ldrinit.c"
DEE	FF75 08	push dword ptr ss:[ebp+8]	
DF1	FF15 6C804000	call dword ptr ds:[<&SizeofResource>]	
DF7	50	push eax	

Figure 45: String XIA located before the function call FindResourceA.

Just before the “call dword ptr ds:[<&FindResourceA>]”, we can see the familiar resource name of XIA which is the zipped file inside r.bin.

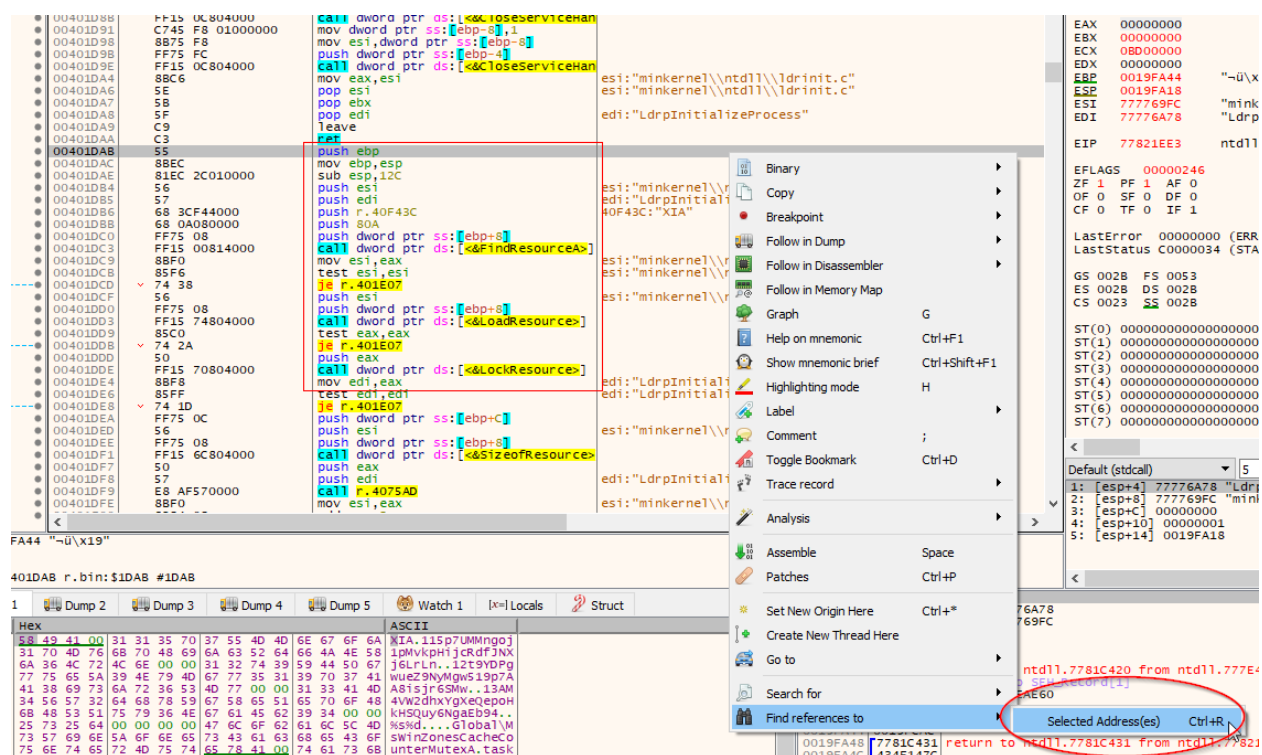


Figure 46: Back-tracing the function which called these "resource" referenced API calls.

Navigate back to the first instruction of this function, because x32Dbg can back-trace function calls to see what code is called in this function.

To do this, right click on the first instruction which is push ebp then click Find References to > Selected Address.

We know this is the first instruction since just above it is the ret instruction.

Constant: 00401DAB (Region r.bin) X		Constant: 004	
Address	Disassembly		
004020D0	call r.401DAB		

Figure 47: Function call which used the "resource" related API calls.

Under References, we can only see one function call. Double click on it.



```
0040208A 50      push    eax
0040208B FF15 D8804000  call    dword ptr ds:[<&SetCurrentDirectoryA>]
004020C1 6A 01     push    1
004020C3 E8 35F0FFFF  call    r.4010FD
004020C8 C70424 2CF54000  mov     dword ptr ss:[esp],r.40F52C  40F52C:"WNcry@2o17"
004020CF 53      push    ebx
004020D0 E8 D6FCFFFF  call    r.401DAB
004020D5 E8 C4FDFFFF  call    r.401E9E
004020DA 53      push    ebx
004020DB 53      push    r.00401DAB
004020DC 68 20F54000  push    r.40F54000
004020E1 E8 7EEFFFFF  call    r.401E9E
004020E6 53      push    ebx
004020E7 53      push    ebx
004020E8 68 FCF44000  push    r.40FCF44000
004020ED E8 72EFFFFF  call    r.401E9E
004020F2 83C4 20     add     esp,20
004020F5 E8 10F6FFFF  call    r.401E9E
004020FA 85C0       test    eax,ebx
004020FC 74 67     je      r.401E9E
004020FE 8D8D 1CF9FFFF  lea     ecx, dword ptr ss:[ebp+8]
00402104 E8 F4F1FFFF  call    r.401E07
00402109 53      push    esi
0040210A 53      push    ebx
0040210B 53      push    dword ptr ss:[ebp+8]
0040210C 8D8D 1CF9FFFF  lea     ecx, dword ptr ss:[<&LoadResource>]
00402112 E8 20F3FFFF  call    r.401E07
00402117 85C0       test    eax,ebx
00402119 74 3F     je      r.401E9E
0040211B 8D45 FC     lea     eax, dword ptr ds:[<&LockResource>]
0040211E 8D8D 1CF9FFFF  lea     ecx, dword ptr ss:[ebp-6E4]
00402124 50      push    eax
00402125 68 F4F44000  push    r.40F4F44000
0040212A 895D FC     mov     dword ptr ss:[ebp-4],ebx
0040212D E8 74F3FFFF  call    r.4014A6
00402132 3BC3       cmp     eax,ebx
00402134 74 24     je      r.40215A
00402136 FF75 FC     push    dword ptr ss:[ebp-4]
00402139 50      push    eax
0040213A E8 7E000000  call    r.4021BD
```

Figure 48: "WNcry@2o17". Notable string before the instruction "call r.401DAB".

Hovering over the function "call r.401DAB", we can see the same exact same pattern we previously observed which is the combination of FindResourceA, LoadResource, LockResource.

There is a notable string which looks like the password.
40F52C:"WNcry@2o17"

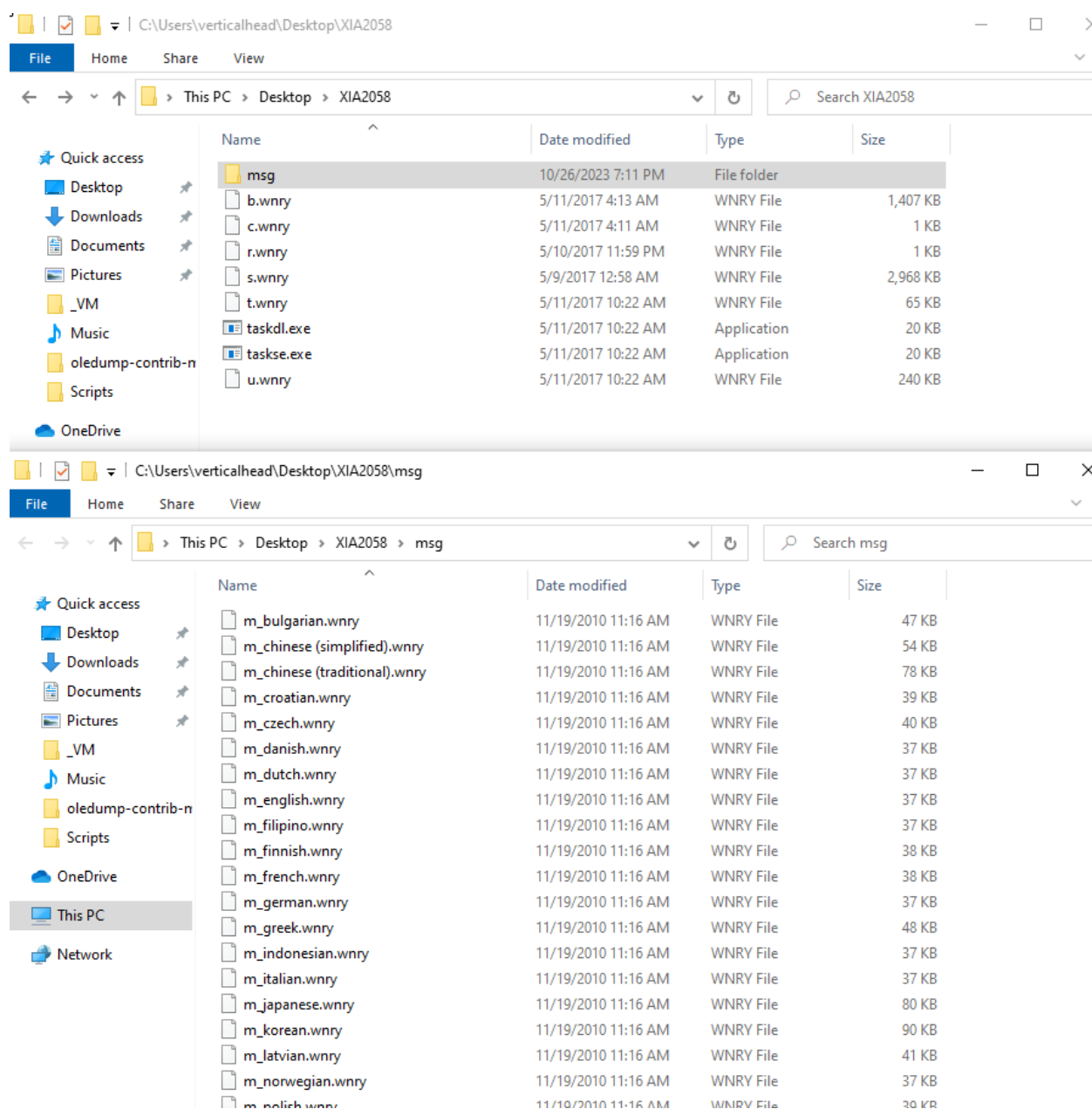


Figure 49: Extracted XIA2058.zip

Now we're able to extract the zip file. The screenshot above contains the contents of XIA2058.zip.



CPU	Log	Notes	Breakpoints	Memory Map	Call Stack	SEH	Script	Symbols	Source	Refer
Base	Module	Address	Type	Ordinal	Symbol					
00400000	r.bin	0040778A	Export	0	OptionalHeader.AddressOfEntryPoint					
73D70000	apphelp.dll	0040802C	Import		kernel32.GetFileAttributesW					
75630000	sechost.dll	00408030	Import		kernel32.GetFileSizeEx					
75760000	msvcrt4.dll	00408034	Import		kernel32.CreateFileA					
75DA0000	kernelbase.dll	00408038	Import		ntdll.InitializeCriticalSection					
76100000	rpcrt4.dll	0040803C	Import		ntdll.DeleteCriticalSection					
76C70000	gd132fu11.dll	00408040	Import		kernel32.ReadFile					
76D50000	user32.dll	00408044	Import		kernel32.GetFileSize					
76F80000	win32u.dll	00408048	Import		kernel32.WriteFile					
77200000	advapi32.dll	0040804C	Import		ntdll.LeaveCriticalSection					
77280000	gd132.dll	00408050	Import		ntdll.EnterCriticalSection					
772B0000	ucrtbase.dll	00408054	Import		kernel32.SetFileAttributesW					
77460000	kernel32.dll	00408058	Import		kernel32.SetCurrentDirectoryW					
77610000	msvcrt.dll	0040805C	Import		kernel32.CreateDirectoryW					
77770000	ntdll.dll	00408060	Import		kernel32.GetTempPathW					
		00408064	Import		kernel32.GetWindowsDirectoryW					
		00408068	Import		kernel32.GetFileAttributesA					
		0040806C	Import		kernel32.SizeofResource					
		00408070	Import		kernel32.LockResource					
		00408074	Import		kernel32.LoadResource					
		00408078	Import		kernel32.MultiByteToWideChar					
		0040807C	Import		kernel32.Sleep					
		00408080	Import		kernel32.OpenMutexA					
		00408084	Import		kernel32.GetFullPathNameA					
		00408088	Import		kernel32.CopyFileA					
		0040808C	Import		kernel32.GetModuleFileNameA					
		00408090	Import		kernel32.VirtualAlloc					
		00408094	Import		kernel32.VirtualFree					
		00408098	Import		kernel32.FreeLibrary					
		0040809C	Import		ntdll.HeapAlloc					
		004080A0	Import		kernel32.GetProcessHeap					
		004080A4	Import		kernel32.GetModuleHandleA					
		004080A8	Import		kernel32.SetLastError					
		004080AC	Import		kernel32.VirtualProtect					
		004080B0	Import		kernel32.IsBadReadPtr					
		004080B4	Import		kernel32.HeapFree					

Figure 50: Another approach using Symbols tab.

Tried another approach using the Symbols tab in x32dbg just like what was instructed in purpl3f0xsecr1ty's blog.

Double click on r.bin.



The screenshot shows the Immunity Debugger interface. The CPU tab is active, displaying assembly code. A right-click context menu is open over the CPU tab, with 'Search for' selected. The 'Search for' submenu is open, showing 'Find references to' and 'String references' (highlighted). The 'String references' list shows several results, including 'minkernel\\ntdll\\ldrinit.c' and 'minkernel\\ntdll\\ldrinit.c'.

Figure 51: Search for string references

Right click anywhere in CPU tab and search for string references.



Wannacry Malware
Oct 2023
v1.0



Advanced Dynamic Analysis

The goal is to evade the kill switch URL even if we have fake-net or inetsim enabled.

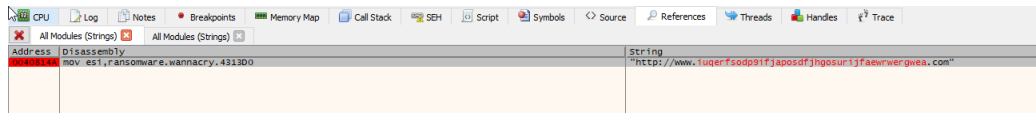


Figure 53: Search for string reference using the kill switch URL

Double click on the address and we'll be redirected to CPU tab showing the URL string passed as an argument to the InternetOpenUrlA function.

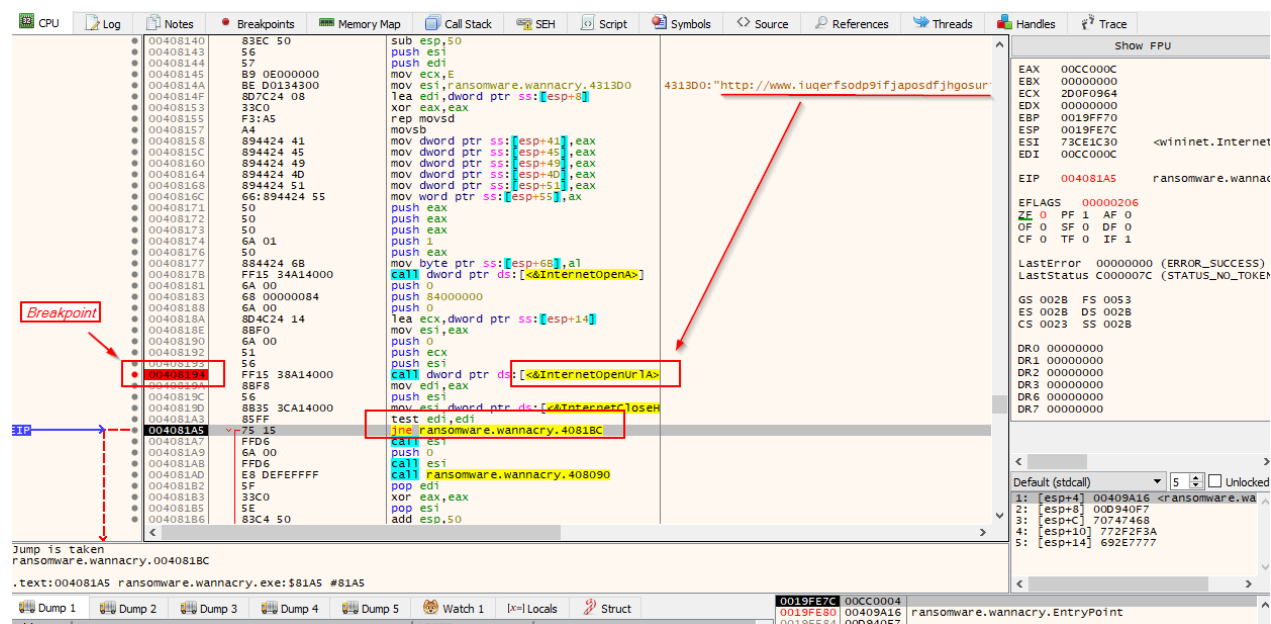


Figure 54: Kill Switch URL passed as an argument to the InternetOpenUrlA function.

- We set a breakpoint on InternetOpenUrlA.
- Step over until we arrive at the jne instruction.
- As we can see, the ZF value is 0.
- If ZF is set to 0, it means that the result of test edi,edi is not 0.
- If the result of test edi,edi is not 0, it means InternetOpenUrlA is able to connect to the kill switch URL.
- This is expected since we have inetsim enabled.



- Since ZF is set to 0, it will perform the jump to 'call esi' instruction.

```
00408145 89 0E000000 mov ecx,E
0040814A BE D0134300 mov esi,ransomware.wannacry.4313D0
0040814F 8D7C24 08 lea edi,dword ptr ss:[esp+8]
00408153 33C0 xor eax,eax
00408155 F3:A5 rep movsd
00408157 A4 movsb
00408158 894424 41 mov dword ptr ss:[esp+41],eax
0040815C 894424 45 mov dword ptr ss:[esp+45],eax
00408160 894424 49 mov dword ptr ss:[esp+49],eax
00408164 894424 4D mov dword ptr ss:[esp+4D],eax
00408168 894424 51 mov dword ptr ss:[esp+51],eax
0040816C 66:894424 55 mov word ptr ss:[esp+55],ax
00408171 50 push eax
00408172 50 push eax
00408173 50 push eax
00408174 6A 01 push 1
00408176 50 push eax
00408177 884424 68 mov byte ptr ss:[esp+68],al
00408178 FF15 34A14000 call dword ptr ds:[<&InternetOpenA>]
00408181 6A 00 push 0
00408183 68 00000084 push 84000000
00408188 6A 00 push 0
0040818E 8D4C24 14 lea ecx,dword ptr ss:[esp+14]
00408190 8BF0 mov esi,eax
00408190 6A 00 push 0
00408192 51 push ecx
00408193 56 push esi
00408194 FF15 38A14000 call dword ptr ds:[<&InternetOpenUrlA>]
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 58 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
```

Figure 55: Jump to call esi instruction

Clicking “(F7) Step into” the ‘call esi’ instruction and following the code, we’ll arrive at this exit instruction. This validates that the jump to the “call esi” instruction will eventually arrive at this exit instruction.

```
00409B44 50 push eax
00409B45 E8 F6E5FFFF call ransomware.wannacry.408140
00409B4A 8945 98 mov dword ptr ss:[ebp-68],eax
00409B4D 50 push eax
00409B4E FF15 ECA04000 call dword ptr ds:[<&exit>]
00409B54 8845 EC mov eax,dword ptr ss:[ebp-64]
00409B57 8B08 mov ecx,dword ptr ds:[eax]
00409B59 8B09 mov ecx,dword ptr ds:[ecx]
00409B5B 894D 88 mov dword ptr ss:[ebp-78],ecx
00409B5E 50 push eax
00409B5F 51 push ecx
00409B60 E8 18000000 call <JMP.&_XcptFilter>
00409B65 59 pop ecx
00409B66 59 pop ecx
00409B67 C3 ret
00409B68 8B65 E8 mov esp,dword ptr ss:[ebp-18]
00409B6B FF75 88 push dword ptr ss:[ebp-78]
00409B6E FF15 F4A04000 call dword ptr ds:[<&_exit>]
00409B74 FF25 00A14000 jmp dword ptr ds:[<&free>]
00409B7A FF25 FCA04000 jmp dword ptr ds:[<&_dllexport>]
00409B80 FF25 F0A04000 jmp dword ptr ds:[<&_XcptFilter>]
00409B86 FF25 D8A04000 jmp dword ptr ds:[<&_initterm>]
00409B8C 68 00000300 push 30000
00409B91 68 00000100 push 10000
00409B96 E8 0D000000 call <JMP.&_controlfp>
00409B98 59 pop ecx
00409B9C 59 pop ecx
00409B9D C3 ret
00409B9E 33C0 xor eax,eax
00409BA0 C3 ret
00409BA1 C3 ret
00409BA2 FF25 D0A04000 jmp dword ptr ds:[<&_except_handler3>]
00409BA8 FF25 E8A04000 jmp dword ptr ds:[<&_controlfp>]
00409BAE CC int3
00409BAF CC int3
00409BB0 8D4D E4 lea ecx,dword ptr ss:[ebp-1C]
00409BB3 E9 18E6FFFF jmp ransomware.wannacry.4081D0
00409BB8 8D4D D4 lea ecx,dword ptr ss:[ebp-2C]
00409BBB E9 10E6FFFF jmp ransomware.wannacry.4081D0
00409BC0 B8 80A14000 mov eax,ransomware.wannacry.40A180
00409BC5 E9 40FEFFFF jmp <JMP.&_CxxFrameHandler3>
00409BCA 0000 add byte ptr ds:[eax],al
```

Figure 56: Call to exit instruction.



Clicking F8 (Step over) after the exit instruction above will eventually lead to the debugging being stopped. Now, we can confirm that this is the kill switch URL.

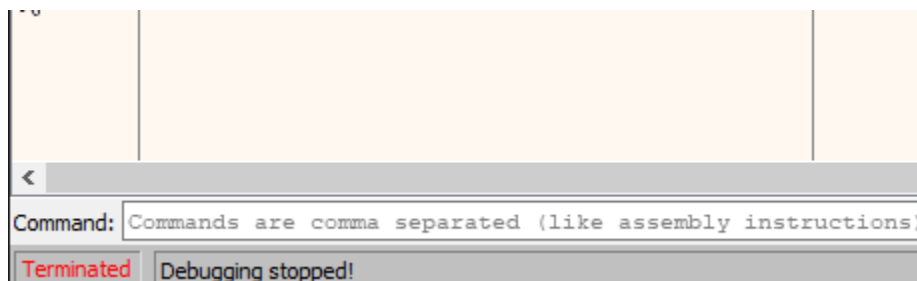


Figure 57: Terminated debugger

Now, let's test what will happen if we set ZF to 1 before executing the “jne ransomware.wannacry.4081BC” instruction.

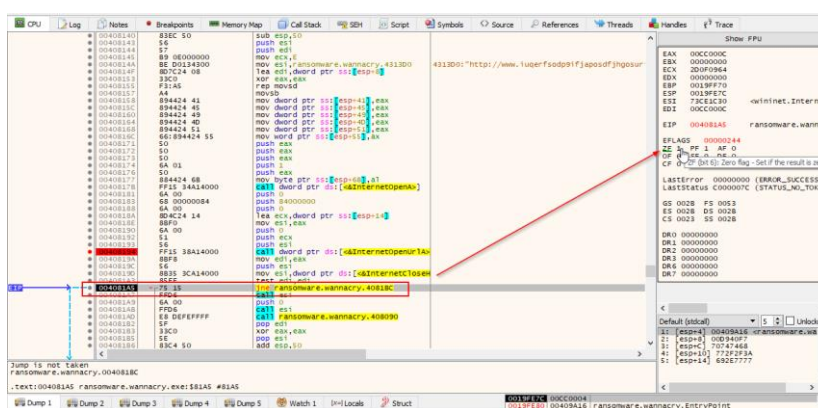


Figure 58: ZF is changed from 0 to 1.

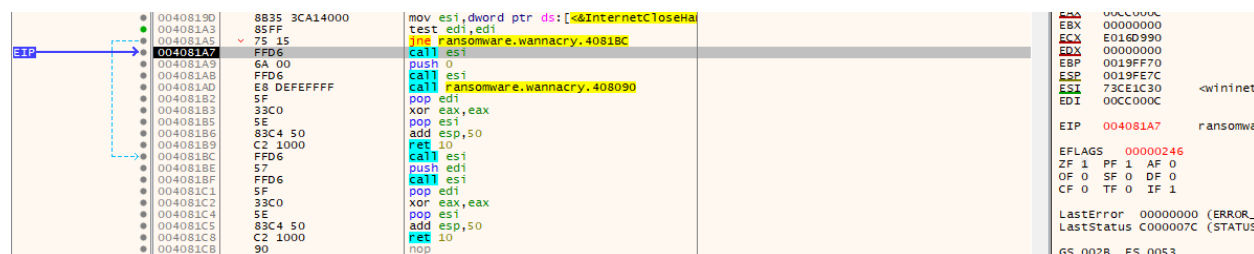


Figure 59: Skipped the jump and proceed to the next instruction below it.

It will eventually call the “fcn.00408090” function which is what we found from the graph of the main function in advanced static analysis section. This function contains the service installed by the wannacry malware for persistence.

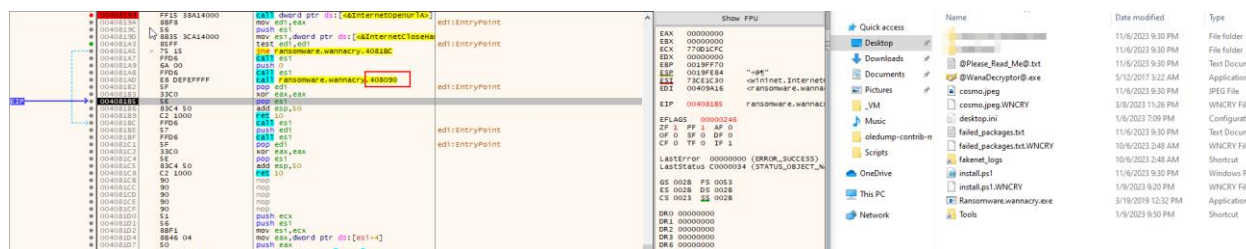


Figure 60: Bypassed the kill switch URL

Ransomware proceeds with its intended routine which is to encrypt the files. At this point, we're able to execute the ransomware even if it can connect to the kill switch URL.

Indicators of Compromise

Network Indicators

URL	Description
gx7ekbenv2riucmf[.]onion	Onion Links
57g7spgrzlojinas[.]onion	
xxlvbrloxvriy2c5[.]onion	
76jdd2ir2embyv47[.]onion	
cwwnhwhlz52maqm7[.]onion	
http[:]//www[.]iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea[.]com	Kill switch URL

Host-based Indicators

File Name	MD5	SHA-1
Ransomware.wannacry.exe	db349b97c37d22f5ea1d1841e3c89eb4	e889544aff85ffaf8b0d0da705105dee7c97fe26
@WanaDecryptor@.exe	7bf2b57f2a205768755c07f238fb32cc	45356a9dd616ed7161a3b9192e2f318d0ab5ad10
c.wnry	383a85eab6ecda319bfddd82416fc6c2	2a9324e1d02c3e41582bf5370043d8afeb02ba6f
f.wnry	b55553edf56f1a836cdde37b1aa3ba00	a9f1fe156145332fdb5a38c4734d2b9f53c09de
r.wnry	3e0020fc529b1c2a061016dd2469ba96	c3a91c22b63f6fe709e7c29cafb29a2ee83e6ade
s.wnry	ad4c9de7c8c40813f200ba1c2fa33083	d1af27518d455d432b62d73c6a1497d032f6120e

Wannacry Malware
Oct 2023
v1.0



t.wnry	5dcaac857e695a65f5c3ef1441a73a8f	7b10aeeee05e7a1efb43d9f837e9356ad55c07dd
taskdl.exe	4fef5e34143e646dbf9907c4374276f5	47a9ad4125b6bd7c55e4e7da251e23f089407b8f
tasksche.exe	84c82835a5d21bbcf75a61706d8ab549	5ff465afaabcbf0150d1a3ab2c2e74f3a4426467
taskse.exe	8495400f199ac77853c53b5a3f278f3e	be5d6279874da315e3080b06083757aad9b32c23
u.wnry	7bf2b57f2a205768755c07f238fb32cc	45356a9dd616ed7161a3b9192e2f318d0ab5ad10
00000000.eky	6770da192604180fad0633ac6e3853c9	5ec78b06ea4aebb96ee24b1eb374a0097d3b24de
00000000.pky	376deea3ed97b37ba39571ce27a96fc5	de41e5da1e80dc202173eab51b765a5166faeef2
00000000.res	2046fef12d822680369dc75d4d9583d9	58e16fc2928b23b0cac15859adf574adc395b7c4
b.wnry	c17170262312f3be7027bc2ca825bf0c	f19eceda82973239a1fdc5826bce7691e5dcb4fb

Commands:

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

Rules & Signatures

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

Appendices

A. Yara Rules

```
rule WannaCry {  
  
  meta:  
    last_updated = "30/10/2023"  
    author = "verticalhead"  
    description = "wannacry rule"  
  
  strings:  
    $PE_magic_byte = "MZ"  
    $wnry = { 2E 77 6E 72 79 } // .wnry  
    $domain = { 69 75 71 65 72 66 73 6F 64 70 39 69 66 6A 61 70 6F 73 64 66 6A 68 67 6F 73 75 72 69 6A 66 61 65  
77 72 77 65 72 67 77 65 61 2E 63 6F 6D } // iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com  
    $smb = { 53 4D 42 } // SMB  
    $ipc = { 49 50 43 } // IPC  
    $security = { 2D 6D 20 73 65 63 75 72 69 74 79 } // -m security
```

Wannacry Malware
Oct 2023
v1.0



```
$qeriuwjhrf = { 71 65 72 69 75 77 6A 68 72 66 } // qeriuwjhrf
condition:
  all of them
}
```

B. Capa verbose result

```
md5      db349b97c37d22f5ea1d1841e3c89eb4
sha1     e889544aff85ffa8b0d0da705105dee7c97fe26
sha256   24d004a104d4d54034dbcffc2a4b19a11f39008a575aa614ea04703480b1022c
path     C:/Users/verticalhead/Desktop/Ransomware.wannacry.exe
timestamp 2023-10-24 08:34:42.465471
capa version 6.1.0
os        windows
format    pe
arch      i386
extractor VivisectFeatureExtractor
base address 0x400000
rules     C:/Users/verticalhead/AppData/Local/Temp/_ME150962/rules
function count 82
library function count 5
total feature count 52376
```

```
reference analysis tools strings
namespace anti-analysis
scope file
```

```
check for time delay via QueryPerformanceCounter
namespace anti-analysis/anti-debugging/debugger-detection
scope function
matches 0x401660
```

```
contain obfuscated stackstrings
namespace anti-analysis/obfuscation/string/stackstring
scope basic block
matches 0x401D80
```

```
receive data (5 matches)
namespace communication
description all known techniques for receiving data from a potential C2 server
scope function
matches 0x401370
        0x401980
        0x401B70
        0x406F50
        0x4072A0
```

```
send data (5 matches)
namespace communication
description all known techniques for sending data to a potential C2 server
scope function
matches 0x401370
        0x401980
        0x401B70
        0x406F50
        0x4072A0
```

```
connect to URL
namespace communication/http/client
scope function
matches 0x408140
```

```
create HTTP request
namespace communication/http/client
scope function
matches 0x408140
```

```
get socket status
namespace communication/socket
scope function
```

Wannacry Malware
Oct 2023
v1.0



```
matches 0x407480

initialize Winsock library
namespace communication/socket
scope function
matches 0x407B90

set socket configuration
namespace communication/socket
scope function
matches 0x407480

receive data on socket (5 matches)
namespace communication/socket/receive
scope function
matches 0x401370
0x401980
0x401B70
0x406F50
0x4072A0

send data on socket (5 matches)
namespace communication/socket/send
scope function
matches 0x401370
0x401980
0x401B70
0x406F50
0x4072A0

connect TCP socket
namespace communication/socket/tcp
scope function
matches 0x407480

create TCP socket
namespace communication/socket/tcp
scope basic block
matches 0x407480

create UDP socket (4 matches)
namespace communication/socket/udp/send
scope basic block
matches 0x401980
0x401B70
0x4072A0
0x407480

act as TCP client
namespace communication/tcp/client
scope function
matches 0x407480

generate random numbers via WinAPI
namespace data-manipulation/prng
scope function
matches 0x407660

extract resource via kernel32 functions
namespace executable/resource
scope function
matches 0x407CE0

contain an embedded PE file
namespace executable/subfile/pe
scope file

get file size
namespace host-interaction/file-system/meta
scope function
matches 0x407A20

move file
namespace host-interaction/file-system/move
scope function
matches 0x407CE0

read file on Windows
namespace host-interaction/file-system/read
scope function
matches 0x407A20
```




```
get number of processors
namespace host-interaction/hardware/cpu
scope function
matches 0x407720

terminate process
namespace host-interaction/process/terminate
scope function
matches 0x408000

run as service
namespace host-interaction/service
scope file

create service
namespace host-interaction/service/create
scope function
matches 0x407C40

modify service
namespace host-interaction/service/modify
scope function
matches 0x407FA0

start service
namespace host-interaction/service/start
scope function
matches 0x407C40

create thread (4 matches)
namespace host-interaction/thread/create
scope basic block
matches 0x4076C2
    0x4077C3
    0x407BDA
    0x407C0D

terminate thread
namespace host-interaction/thread/terminate
scope basic block
matches 0x4076F2

link function at runtime on Windows
namespace linking/runtime-linking
scope function
matches 0x407CE0

linked against ZLIB
namespace linking/static/zlib
scope file

inspect section memory permissions
namespace load-code/pe
description translate section memory permissions (specified in the 'Characteristics' field of the image section header) into page protection constants
scope function
matches 0x401D80

persist via Windows service
namespace persistence/service
scope function
matches 0x407C40
```