

A Comprehensive Analysis of WannaCry variant

*Conducting both static and dynamic analysis on a WannaCry
variant to discover its characteristics, components,
functionality, and impact*

Adam Board

CMP320: Advanced Ethical Hacking

2022/23

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

Abstract

Malware has been infecting systems since the early days of computing and can cause substantial damage to both individuals and organisations. A few of the impacts of malware attacks are data loss, reputational damage, and financial loss. These impacts can be the cause of an organisation being required to shut down or an individual's way of living could be destroyed. Knowing the potential impact malware presents, a solution was needed to prevent or mitigate future attacks; Malware analysis is used to determine how the malware functions and the damage it can cause, the information obtained is then used to create countermeasures like anti-virus software, to mitigate the damage and prevent the malware's use in future attacks. In this case, a variant of the malware WannaCry is being analysed to confirm its identity, then determine and understand its characteristics, components, functionality, and impact.

This report incorporates an in-depth malware analysis for a variant sample of WannaCry, using the NATO-standard Malware Reverse Engineering Handbook Methodology. The aforementioned methodology was in use to ensure a thorough analysis was conducted using both static and dynamic analysis techniques and tools to discover the different components that provides the malware its functionality to cause damage to a user's machine.

During the analysis, the malware was confirmed to be a variant of WannaCry which is a type of ransomware that encrypts the host's files by acting as a dropper for an encrypted ZIP file embedded within the executable. The ZIP file would then use a multitude of methods to allow itself to persist on the host's machine even through device restarts, and would begin the encrypting process for the numerous files types and change their extension to ".wnry" on the host's machine. Additionally, the malware would prompt the host to pay into a bitcoin wallet to decrypt the files. Afterwards, the malicious executable attempts to connect to different Tor services through onion addresses as way to communicate with a Command-and-Control server. Using the host-based and network-based indicators determined throughout the report, countermeasures are suggested to mitigate and prevent future attacks using this WannaCry Variant such as, using the indicators of compromise to create Yara rules, or updating the Operating system to avoid security issues with older versions.

Contents

| | | |
|-------|--------------------------------------|----|
| 1 | Introduction | 1 |
| 1.1 | Background | 1 |
| 1.2 | Aim | 2 |
| 2 | Procedure..... | 3 |
| 2.1 | Procedure and Results | 3 |
| 2.1.1 | Methodology..... | 3 |
| 2.2 | Setting up the lab environment | 4 |
| 2.3 | Static Analysis..... | 6 |
| 2.3.1 | VirusTotal | 6 |
| 2.3.2 | String Analysis | 7 |
| 2.3.3 | PEiD | 8 |
| 2.3.4 | CFF Explorer | 9 |
| 2.3.5 | Resource Hacker | 10 |
| 2.3.6 | PeStudio | 11 |
| 2.4 | Disassembly..... | 13 |
| 2.4.1 | IDA..... | 14 |
| 2.5 | Dynamic Analysis | 19 |
| 2.5.1 | Process Monitor | 20 |
| 2.5.2 | Process Explorer..... | 21 |
| 2.5.3 | Regshot | 22 |
| 2.5.4 | Inetsim | 23 |
| 2.5.5 | Network Traffic Analysis | 23 |
| 2.5.6 | Debuggers | 25 |
| 3 | Discussion..... | 26 |
| 3.1 | General Discussion..... | 26 |
| 3.1.1 | VirusTotal | 26 |
| 3.1.2 | String Analysis | 26 |
| 3.1.3 | PEiD | 26 |
| 3.1.4 | CFF Explorer | 26 |
| 3.1.5 | Resource Hacker | 26 |
| 3.1.6 | PeStudio | 26 |

| | | |
|---|--------------------------------|----|
| 3.1.7 | IDA..... | 27 |
| 3.1.8 | Process Monitor | 27 |
| 3.1.9 | Process Explorer..... | 27 |
| 3.1.10 | Regshot | 27 |
| 3.1.11 | Network Traffic Analysis | 28 |
| 3.1.12 | Debuggers | 28 |
| 3.2 | Conclusion..... | 28 |
| 3.2.1 | Conclusion..... | 28 |
| 3.2.2 | Countermeasures..... | 28 |
| 3.3 | Future Work..... | 29 |
| 4 | References | 30 |
| Appendices..... | | 33 |
| Appendix A – Omitted Section of Methodology..... | | 33 |
| Appendix B – Tool Output..... | | 33 |
| 4.1.1 | Flare Floss..... | 33 |
| Appendix C – Files found from Malware Analysis..... | | 54 |

1 INTRODUCTION

1.1 BACKGROUND

Malware, short for “Malicious Software”, is a type of software created with the intent to cause harm to computer systems, networks, and users by either stealing information or sabotage; Malware comes in a multitude of different forms, such as viruses, worms, and trojans (Cisco, n.d.). The first known malware, the “Creeper Virus” was created in 1971. The “Creeper Virus” was not malicious and was an experiment to show the capabilities of self-replicating software that could spread to other computers by displaying a short message daring the user to try and capture “the creeper” (Johanns, 2020).

Since then, malware has evolved and has become increasingly sophisticated; In the early 2000s, the malicious actors began using aggressive social engineering techniques in a malware known as the “ILOVEYOU” worm which was one of the most damaging infections of its time, spreading to a final estimate of 10% of internet-connected computers and causing around \$10 billion worth of damage (Root, 2022). A more recent case of malware’s evolution is the wiper malware, “NotPetya” which encrypted a victim’s machine and then began indiscriminately wiping user data to destroy the machine (Shepherd, 2019). This attack cost the organisation, Maersk around \$300 million in damage, as well as the cost of downtime for the organisation when sales could not be made (Lord, 2020).

As a result of these dangerous malware attacks, organisations have had to develop countermeasures to prevent attacks using malware such as firewalls, antivirus software and employee training. However, to be able to create all these countermeasures, the malware needed to be understood and identifiable to the countermeasures.

Malware analysts are cyber-security professionals who specialise in analysing malware to understand how it works and what can be done to mitigate or neutralise malware. There are two main methods for analysing malware: static analysis, which involves analysing the code without running the malware, using tools such as “PEiD” and “UPX”, and dynamic analysis, which entails executing the malware in a controlled environment to observe its behaviour, using tools such as disassemblers or debuggers (Cardona, 2021).

In the field of malware analysis, the first and foremost step is to obtain a malware sample to be analysed. There are various methods of obtaining a malware sample, such as honeypots, downloading manually off a malicious website, or a user handing over a sample of malware. In this case, the sample being analysed within this report was obtained from a collection of malware that was given to the analyst to choose from. The chosen sample of malware to be analysed is “WannaCry”, a piece of ransomware which caused massive damage to the NHS in May of 2017 and cost an estimated £92 million (Allegretti, 2018).

1.2 AIM

This report aims to confirm the identity and analyse a sample of malware to understand its characteristics, components, functionality, and impact. To achieve this, the following sub-aims will be addressed:

- Confirm the identity of the malware sample and establish that it is WannaCry.
- Determine the type of malware that the sample is categorised as (spyware, ransomware, etc.)
- Investigate the characteristics and components of the malware and determine its method of spreading and infecting machines.
- Examine the malware sample using static analysis tools to understand its functionality and determine host-based identifiers.
- Examine the code and its behaviour when it is executed within a controlled environment to further understand the functionality of the malware, confirm any findings from the static analysis and determine network-based identifiers.
- Evaluate the possible impact of the malware, this includes the attack scope and the extent of damage caused by it.

By following a methodology with these aims in mind, the report seeks to deliver an in-depth analysis of the malware and deliver an addition to the understanding of malware identification and analysis techniques.

2 PROCEDURE

2.1 PROCEDURE AND RESULTS

2.1.1 Methodology

For this report, the methodology used by the malware analyst was from a reputable and established organisation, the “NATO Cooperative Cyber Defence Centre of Excellence” (CCDCOE). Their methodology, the “reverse engineering handbook”, released in 2020, encompasses the initial setup of a lab environment, how to conduct static analysis, dynamic analysis, and reverse engineering. The analyst chose this methodology over other available options because the CCDCOE specialises in cyber-security and is a world-renowned organisation (The NATO Cooperative Cyber Defence Centre of Excellence, n.d.). The handbook is based on NATO standards, which ensures the quality of the methodology. Finally, the document is an open-source resource, which makes it accessible and a cost-effective option for individuals looking to follow a standard methodology.

The methodology consists of the following main steps, each with its subheadings:

1. How to set up a lab environment
2. Static malware analysis
 - 2.1. Static analysis techniques & tools
3. Disassembly (IDA)
 - 3.1. IDA free
4. Dynamic Analysis
 - 4.1. Behaviour analysis tools
 - 4.2. Debuggers

This handbook has steps that did not apply to the analysis of the malware and was included in Appendix A, this includes sections that are informational to malware analysis as a field and are not techniques for analysing malware itself or were not in scope of the analysis. Another subsection was not included as it uses multiple tools that are used for the same purpose and only one of the tools is required to gather the results.

For analysing the malware, the analyst used a virtual machine that was provided by the organisation they are a part of. The provided virtual machine was a Windows 10 machine that had the FlareVm script run on it to install a library of tools for malware analysis, such as *IDA*, *PEid*, and *PeStudio*. The virtual machine also had the required malware sample pre-downloaded on the system in a password-protected ZIP file. This can be seen in Figure 1. Finally, the analyst had a second virtual machine called *remnux* which was utilised as a spoof internet connection for catching any requests sent by the malware.

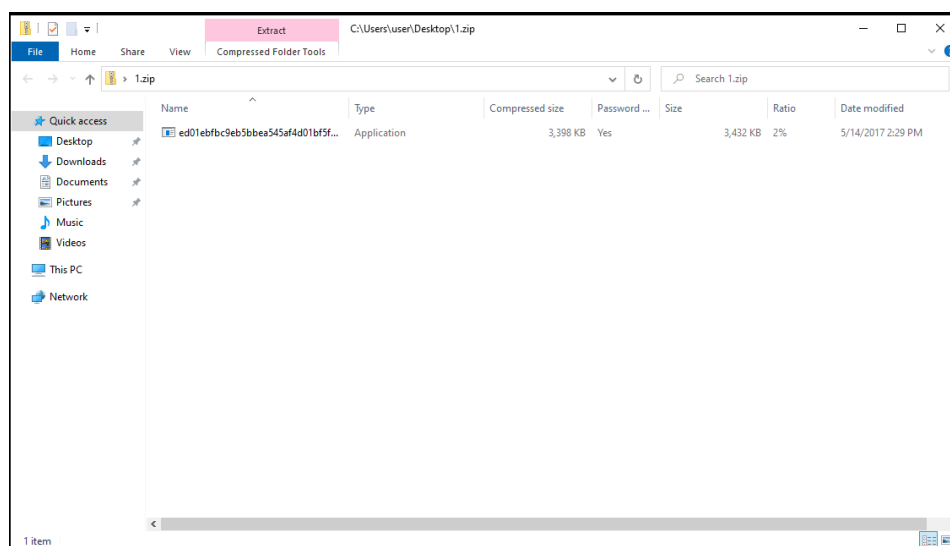


Figure 1 Password-Protected ZIP file containing the malware sample

2.2 SETTING UP THE LAB ENVIRONMENT

Creating a controlled lab environment was crucial for conducting the malware analysis without risking contamination of the analyst's personal computer. However, numerous types of malware are designed to halt their process if they cannot detect an internet connection. To circumvent the problem, the analyst utilised a second virtual machine named **Remnux**, which uses a pre-installed tool named **inetsim** to simulate network services and protocols without connecting to the internet. However, the analyst had to make changes to the configuration file, “/etc/inetsim/inetsim.conf” which can be seen in Figures 2 and 3. **inetsim** was executed to begin the simulated services, and to confirm that it was working, the analyst browsed to a random web address on the Windows virtual machine which gave them the generic reply that is standard for the tool. This can be seen in Figures 4 and 5.

The **Remnux** virtual machine also had **Wireshark** running to capture traffic for analysis purposes. Additionally, to ensure that each part of the analysis was done with no changes to the controlled lab environment, the analyst had taken a snapshot of the clean state before any malware had been executed to have a safe revert point to return to after analysis.


```

start_service dns
start_service http
start_service https
start_service smtp
start_service smtps
start_service pop3
start_service pop3s
start_service ftp
start_service ftps
#start_service tftp
#start_service irc
#start_service ntp
#start_service ringer
#start_service ident
#start_service syslog
#start_service time_tcp
#start_service time_udp
#start_service daytime_tcp
#start_service daytime_udp
#start_service echo_tcp
#start_service echo_udp
#start_service discard_tcp
#start_service discard_udp
#start_service quotd_tcp
#start_service quotd_udp
#start_service chargen_tcp
#start_service chargen_udp
#start_service dummy_tcp
#start_service dummy_udp

#####
# service_bind_address
#
# IP address to bind services to
#
# Syntax: service_bind_address <IP address>
#
# Default: 127.0.0.1
#
service_bind_address 0.0.0.0

```

Figure 2 Configuration file of inetsim – binding service address to 0.0.0.0

```

#####
# dns_default_ip
#
# Default IP address to return with DNS replies
#
# Syntax: dns_default_ip <IP address>
#
# Default: 127.0.0.1
#
dns_default_ip 10.0.0.4

```

Figure 3 Configuration file of inetsim - binding dns default IP

```

remnux@remnux:~$ inetsim
INetSim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg
Using log directory: /var/log/inetsim/
Using data directory: /var/lib/inetsim/
Using report directory: /var/log/inetsim/report/
Using configuration file: /etc/inetsim/inetsim.conf
Parsing configuration file.
Configuration file parsed successfully.
=== INetSim main process started (PID 1855) ===
Session ID: 1855
Listening on: 10.0.0.4
Real Date/Time: 2023-04-20 20:25:48
Fake Date/Time: 2023-04-20 20:25:48 (Delta: 0 seconds)
Forking services...
* dns 53 tcp udp - started (PID 1859)
* ftps 990 tcp - started (PID 1867)
* ftp 21 tcp - started (PID 1866)
* smtp 25 tcp - started (PID 1862)
* https 443 tcp - started (PID 1861)
* http 80 tcp - started (PID 1860)
* pop3 110 tcp - started (PID 1864)
* smtps 465 tcp - started (PID 1863)
* pop3s 995 tcp - started (PID 1865)
done.
Simulation running.

```

Figure 4 Running inetsim tool

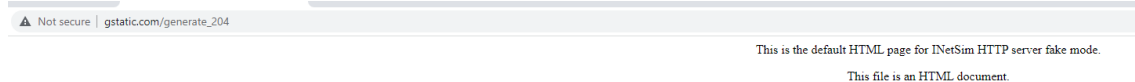


Figure 5 Evidence of inetsim working

2.3 STATIC ANALYSIS

Static analysis refers to the analysis of the Portable Executable files (PE files) without running them; The analyst used basic static analysis techniques and tools to confirm the malicious intent of the file by analysing information such as its functionality, imports, and compilation date (Ahmet, et al., 2020). Additionally, these tools were used by the analyst to confirm the type of malware, the identity of the malware, the different components which give the malware its functionality, and the damage the malware causes.

2.3.1 VirusTotal

To begin the analysis, the malware's md5 hash had been calculated using the tool, **HashMyFiles** on version 2.43, which would be uploaded to the website VirusTotal for cross-referencing against numerous antivirus programs and their detection of malware. This was done to determine whether it is a known sample of malicious malware. This can be seen in Figure 6.

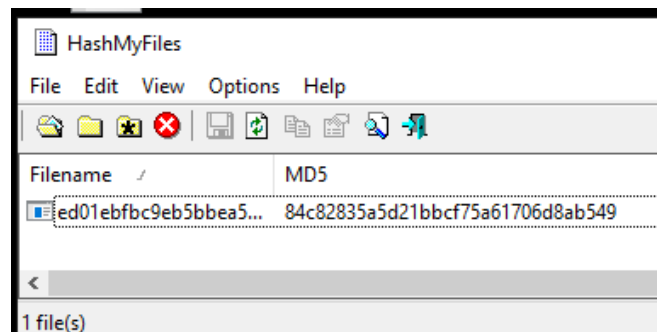


Figure 6 Calculating the hash of malware sample

Upon uploading the hash, "84c82835a5d21bbcf75a61706d8ab549" to VirusTotal, it indicated that the malware identity was "wannacryptor" which is commonly identified as "WannaCry". This can be seen in Figure 7.

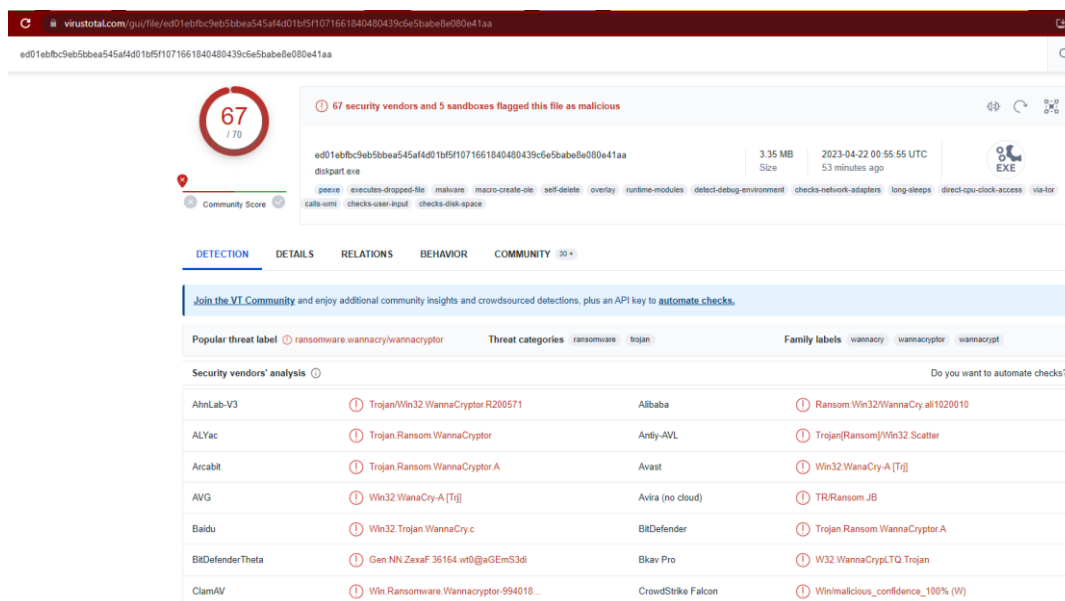


Figure 7 Uploaded hash to VirusTotal

2.3.2 String Analysis

After cross-referencing the malware sample the analyst chose to perform string analysis; string analysis is the process of extracting Ascii and Unicode characters from the binary of the file to be readable. Using the tool **Flare Floss** (mandiant, 2023), the analyst attempted to extract the readable Ascii and Unicode within the malware sample using the command “floss” along with the name of the malware sample. A screenshot of the results can be seen in Figures 8, 9 and 10. The full output of the command can be seen in Appendix B.

```
lAgg
b.wnry
c.wnry
lAgg
msg/m_bulgarian.wnry
"t="
msg/m_chinese (simplified).wnry
"t="
Vbq-
msg/m_chinese (traditional).wnry
```

Figure 8 Output results - wnry file extension

```

| FLOSS UTF-16LE STRINGS (194) |
WannaCrypt0r
Software\
.der
.pfx
.key
.crt
.csr
.p12
.pem
.odt
.otf
.sxw
.stw
.uot
.3ds
.max
.3dm
.ods
.ots
.sxc
.stc
.tif
.slk
.wb2
.odp
.otp
.sxd
.std
.uop
.odg
.otg
.sxm
.mml
.lay
.lay6
.asc
.sqlite3

```

Figure 9 Output results – file extensions

```

OriginalFilename
diskpart.exe
ProductName

```

Figure 10 Output results – diskpart executable

The output from the string analysis revealed four main points of interest for the analyst. A set of files with the extension “.wnry” was located within Figure 8, which is a file extension used by WannaCry either once a file had been affected by it or the file had been originally packaged with the malware (fileinfo, 2019). Secondly, a string within the malware sample indicated the malware to be “WannaCrypt0r” as seen in Figure 9. Additionally, within Figure 9, the string analysis revealed a list of different file extensions held within the malware sample which indicates filetypes that are targeted for encryption (Team, 2017). Lastly, a filename appeared from the string analysis named “diskpart.exe” which can be seen in Figure 10; diskpart.exe is a command interpreter that assists in managing a user’s drives, such as disks, partitions, or volumes (Microsoft, 2023) and could be used to hide the malware as a legitimate file.

2.3.3 PEiD

The next stage for the analysis was using the tool **PEiD**; The tool **PEiD** is used for investigating the PE header to gather more information about the cryptors, packers, and compilers within the malware sample. This information gathering is done by checking the static signatures that are within the sample malware. The analyst uploaded the executable into **PEiD** which can be seen in Figure 11 and 12.

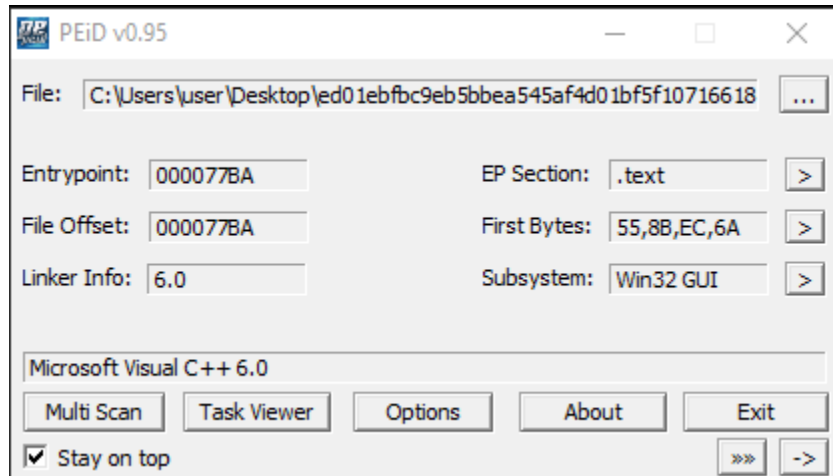


Figure 11 PEiD output - uploaded malware sample

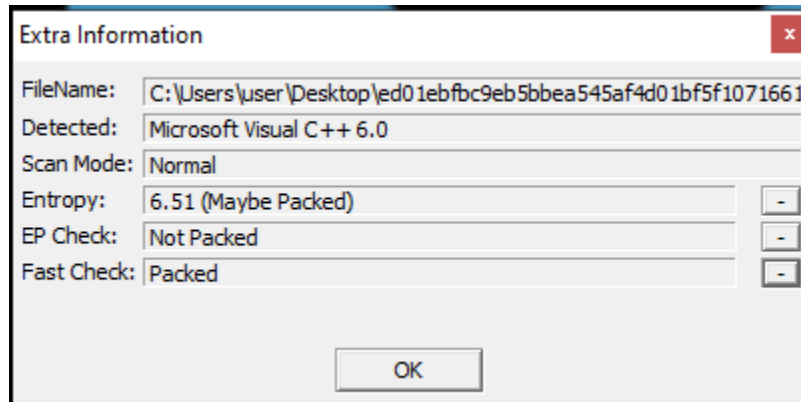


Figure 12 PEiD output - Checking entropy

From Figure 12, the tool indicates that the malware could possibly be packed due to its Entropy value but would need further analysis using packing detectors to confirm whether the malware is packed.

2.3.4 CFF Explorer

Typically, **CFF Explorer** is a tool used to make modifications inside the PE, however, the analyst does not want to edit the malware sample, instead, it can be used to extract information such as the compilation date(displayed in Epoch Unix Time)and architecture type. The analyst inserted the PE into **CFF Explorer**, which returned that the compilation date value was “4CE78F41”. This can be seen in figure 13. Putting this value through an Epoch Unix Time converter reveals it to be compiled on “Saturday, 20 November 2010 09:05:05”.

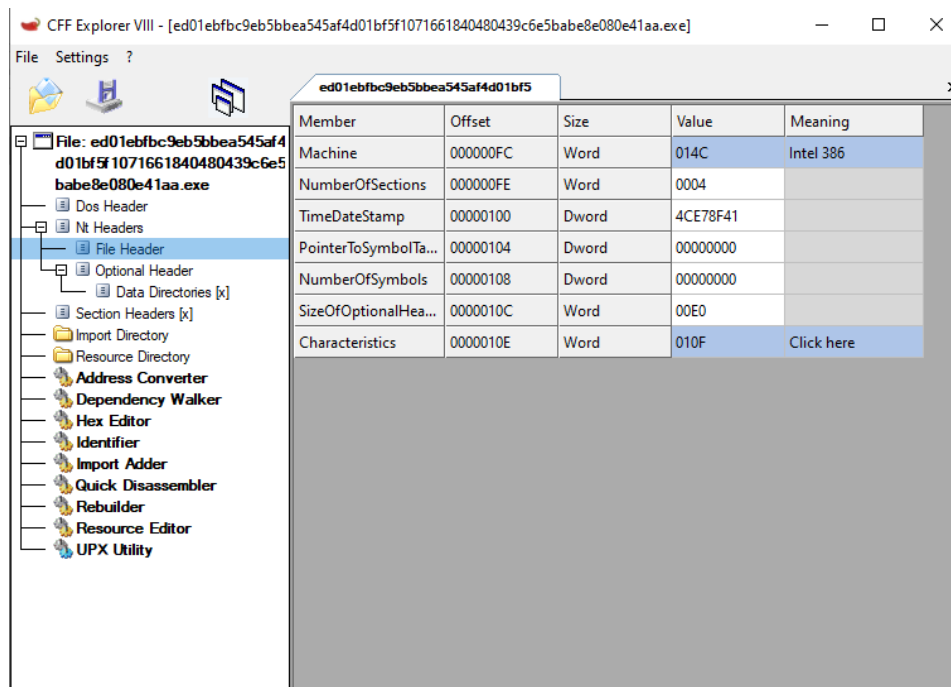


Figure 13 CFF Explorer - File Header

A secondary use of **CFF Explorer** is to validate if section Headers are altered or if there are unusual headers (UPX, etc..) which could indicate the usage of a packing software. The unaltered section headers are “.text”, “.rdata”, “.data”, and “.rsrc” (Ahmet, et al., 2020). Once the analyst compared the headers of the malware sample against headers that are usually present, it appeared to not be affected by packing software as the headings indicate no change from external software. This can be seen in Figure 14.

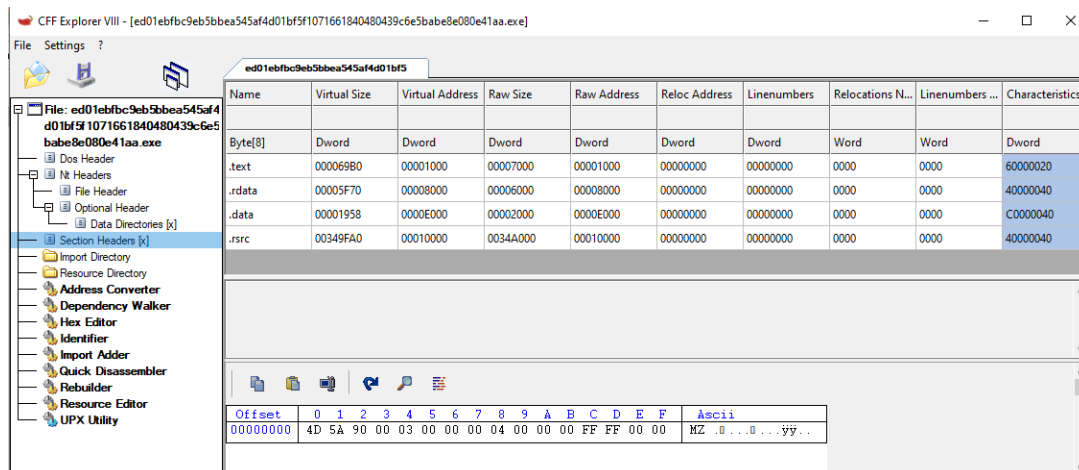


Figure 14 CFF Explorer - Section Headers

2.3.5 Resource Hacker

Resource Hacker is a tool used to extract, modify, and add resources from Windows binaries. In this instance, the analyst used the tool to investigate additional PE files within the malware sample’s resources. two points of interest discovered from the output of the tool was an indicator of a Windows PE named “diskpart.exe” being embedded within the malware sample and ZIP file with the extension

type “XIA” that has encrypted data within it. Both could potentially be a host-based indicator. This can be seen in Figure 15.

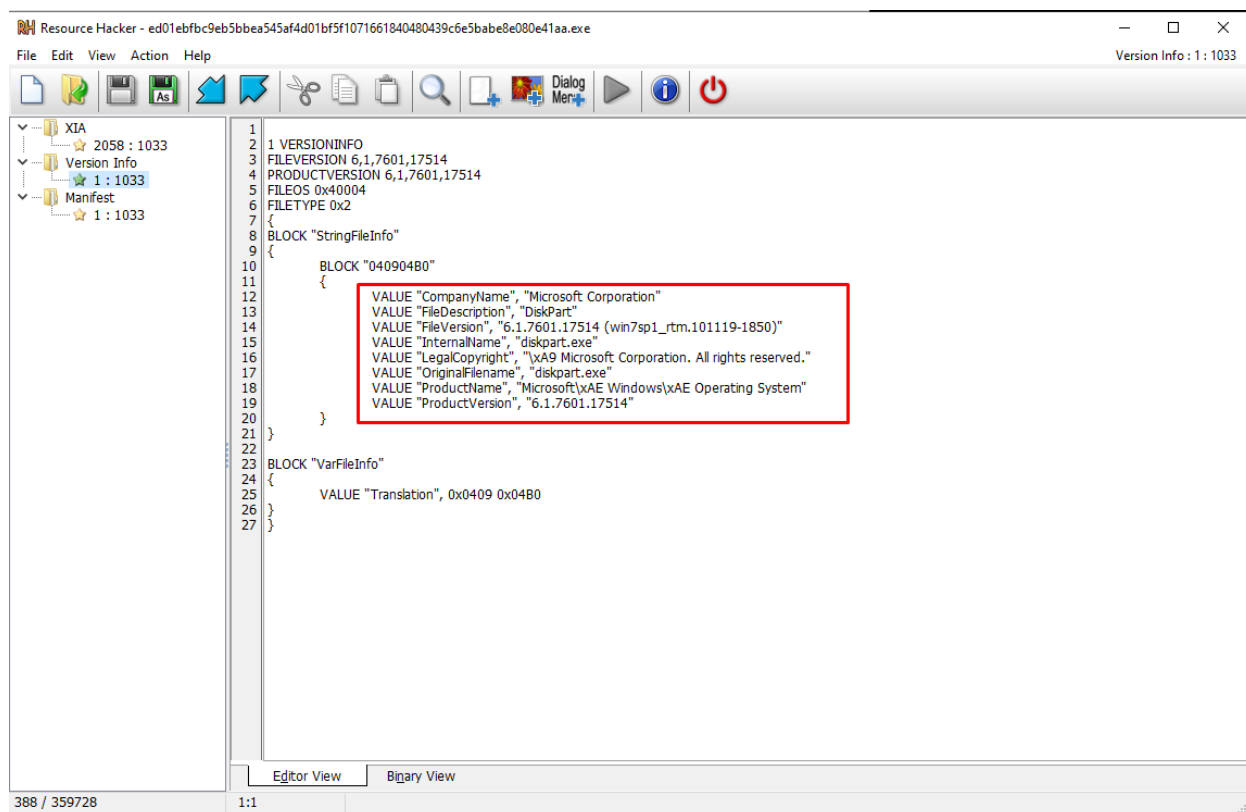


Figure 15 Resource Hacker - diskpart.exe

2.3.6 PeStudio

After finding indicators of a portable executable within the resources of the malware sample, the analyst used **PeStudio** version 9.47 to find suspicious artefacts that may be within the executable files. **PeStudio** has the capability to identify malicious functionalities which are commonly used by malware developers and displays the findings in a multitude of tabs. Opening the malware sample within **PeStudio**, the analyst inspects the “indicators” tab to examine the common indicators of the executable file being malicious. This can be seen in Figure 16.

peStudio 9.47 - Malware Initial Assessment - www.winitor.com - [c:\users\user\desktop\ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe]

file settings about

| 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 16 PeStudio - Indicator tab

Within the “indicators” tab, a list of all indications towards the file being malicious is shown, with the details of what makes it malicious and confidence level, with 1 being a confident malicious indicator (Fox, 2023). From Figure 16, there was four level 1 confidence indicators, two level 2 confidence indicators, and the majority being level 3 confidence indicators. The top level 1 indicator appears to show a resource named “XIA” which is an encrypted file (filext, n.d.) that is “3,446,325” bytes, while the entirety of the executable file size is “3,514,368” bytes. This could potentially indicate that the executable is a dropper sample which installs the malware.

The second level 1 indicator further hints at the executable being a dropper, as an embedded pkzip signature was located within the “.rsrc” section header; “PKWARE” introduced the commonly used ZIP file format in 1989 as apart of their file archiving program, “PKZIP” (PKWARE, n.d.). The existence of a ZIP file within the PE file, suggested the contents of the ZIP file are malicious and the rest of the PE file is a dropper to install the content.

The next tabs of interest to the analyst were the “imports” and “strings” tabs; “imports” contains the imported function names and “strings” contains all strings from the executable that could be parsed from the executable. More information about each imported function can be found by reading the “learn.microsoft.com” webpages for each imported function. The “imports” and “strings” tabs can be seen in Figures 17 and 18.

| imports (114) | flag (14) | first-thunk-original (INT) | first-thunk (IAT) | hint | group (10) | technique (11) | type (1) | ordinal (0) | library (4) |
|----------------------|-----------|----------------------------|-------------------|-------------|--------------|--------------------------|----------|-------------|--------------|
| VirtualProtect | x | 0x0000B36 | 0x0000B36 | 902 (0x386) | memory | Process Injection | implicit | - | KERNEL32.dll |
| CryptReleaseContext | x | 0x0000C14 | 0x0000C14 | 160 (0x0A0) | cryptography | Obfuscated Files or L... | implicit | - | ADVAPI32.dll |
| rand | x | 0x0000CE6 | 0x0000CE6 | 678 (0x2A6) | cryptography | Obfuscated Files or L... | implicit | - | MSVCRT.dll |
| rand | x | 0x0000CEE | 0x0000CEE | 692 (0x2B8) | cryptography | Obfuscated Files or L... | implicit | - | MSVCRT.dll |
| RegCreateKeyW | x | 0x0000C04 | 0x0000C04 | 467 (0x1D3) | registry | Modify Registry | implicit | - | ADVAPI32.dll |
| RegSetValueExA | x | 0x0000BF2 | 0x0000BF2 | 516 (0x204) | registry | Modify Registry | implicit | - | ADVAPI32.dll |
| CreateServiceA | x | 0x0000B32 | 0x0000B32 | 102 (0x066) | execution | Execution through A... | implicit | - | KERNEL32.dll |
| CreateServiceA | x | 0x0000C2A | 0x0000C2A | 100 (0x064) | services | Create or Modify Sys... | implicit | - | ADVAPI32.dll |
| WriteFile | x | 0x0000D97E | 0x0000D97E | 932 (0x3A4) | file | - | implicit | - | KERNEL32.dll |
| SetFileAttributesW | x | 0x000098A | 0x000098A | 794 (0x31A) | file | - | implicit | - | KERNEL32.dll |
| TerminateProcess | x | 0x0000808 | 0x0000808 | 862 (0x35E) | execution | - | implicit | - | KERNEL32.dll |
| GetExitCodeProcess | x | 0x00007F2 | 0x00007F2 | 346 (0x15A) | execution | - | implicit | - | KERNEL32.dll |
| GetCurrentDirectoryW | x | 0x00009D0 | 0x00009D0 | 779 (0x30B) | - | - | implicit | - | KERNEL32.dll |
| GetCurrentDirectoryA | x | 0x0000882 | 0x0000882 | 778 (0x30A) | - | - | implicit | - | KERNEL32.dll |

Figure 17 PeStudio - Imports tab

| encoding (2) | size (bytes) | location | flag (33) | label (424) | group (11) | technique (16) | value (111594) |
|--------------|--------------|------------|-----------|-------------|----------------|---------------------------------|---------------------|
| ascii | 13 | 0x0000C2C | x | import | services | Create or Modify System Process | CreateService |
| ascii | 13 | 0x0000BF4 | x | import | registry | Modify Registry | RegSetValueEx |
| ascii | 12 | 0x0000C06 | x | import | registry | Modify Registry | RegCreateKey |
| ascii | 19 | 0x0000F5C | x | - | reconnaissance | - | GetNativeSystemInfo |
| ascii | 9 | 0x0000B38 | x | import | memory | Process Injection | VirtualProtect |
| ascii | 14 | 0x0000D98 | x | import | file | - | WriteFile |
| ascii | 17 | 0x0000D9C | x | import | file | - | SetFileAttributes |
| ascii | 9 | 0x0000E0D | x | import | file | - | WriteFile |
| ascii | 10 | 0x0000EBA0 | x | - | file | Data Destruction | DeleteFile |
| ascii | 10 | 0x0000EBA0 | x | - | file | Remote File Copy | MoveFileEx |
| ascii | 8 | 0x0000EB8 | x | - | file | Remote File Copy | MoveFile |
| ascii | 18 | 0x00007F4 | x | import | execution | - | GetExitCodeProcess |
| ascii | 16 | 0x0000D80A | x | import | execution | - | TerminateProcess |
| ascii | 13 | 0x0000D834 | x | import | execution | Execution through API | CreateProcess |
| ascii | 19 | 0x0000C16 | x | import | cryptography | Obfuscated Files or Information | CryptReleaseContext |
| ascii | 4 | 0x0000CE8 | x | - | cryptography | Obfuscated Files or Information | rand |
| ascii | 5 | 0x0000CF0 | x | - | cryptography | Obfuscated Files or Information | rand |
| ascii | 11 | 0x0000FDC4 | x | - | cryptography | Obfuscated Files or Information | CryptGenKey |
| ascii | 12 | 0x0000FDD0 | x | - | cryptography | Obfuscated Files or Information | CryptDecrypt |
| ascii | 12 | 0x0000FDE0 | x | - | cryptography | Obfuscated Files or Information | CryptEncrypt |
| ascii | 15 | 0x0000FDF0 | x | - | cryptography | Obfuscated Files or Information | CryptDestroyKey |
| ascii | 14 | 0x0000FE00 | x | - | cryptography | Obfuscated Files or Information | CryptImportKey |
| ascii | 19 | 0x0000F110 | x | - | cryptography | Obfuscated Files or Information | CryptAcquireContext |
| ascii | 19 | 0x0000D884 | x | import | - | - | SetCurrentDirectory |
| ascii | 19 | 0x0000D9D2 | x | import | - | - | SetCurrentDirectory |
| ascii | 3 | 0x0000FE82 | x | - | - | - | xmR |
| ascii | 3 | 0x00072ABF | x | - | - | - | 614 |
| ascii | 3 | 0x000BCAF1 | x | - | - | - | 430 |
| ascii | 3 | 0x000DF261 | x | - | - | - | 83A |
| ascii | 3 | 0x0012A649 | x | - | - | - | xMR |
| ascii | 3 | 0x0017647F | x | - | - | - | 83W |
| ascii | 3 | 0x001D944A | x | - | - | - | 82W |
| ascii | 3 | 0x0020FC3A | x | - | - | - | xMr |

Figure 18 PeStudio - Strings tab organised by flag

The analyst found an imported function name which could indicate the functionality of the malware sample, this can be seen in Figure 17. The function name “CryptReleaseContext”, is a part of the “Cryptography” group and is used to release the handle of a cryptographic service provider and a key container and is typically used for Ransomware (mrd0x, n.d.).

Within the “strings” tab, further hints towards the malware being ransomware are the values within the “Cryptography” group that are “CryptGenKey”, “CryptEncrypt”, “CryptDecrypt”, and “CryptDestroyKey”. These values indicate a resource being encrypted and the key for the encryption being destroyed as to remove the method to Decrypt the resources.

2.4 DISASSEMBLY

The purpose of disassembling a PE file is to give a general understanding of the functionality of the file. Executable files contain machine code in the form of binary; Disassemblers such as “IDA” and “Ghidra” translate machine code into more convenient assembly language. The analyst made use of the disassembler IDA due to it being beginner friendly compared to Ghidra because IDA is a professional tool with commercial development for reverse engineering (Ahmet, et al., 2020).

2.4.1 IDA

The tool, **IDA** is a standard tool used by reverse engineers and malware researchers. The analyst used the free version of **IDA** for the investigation. The initial step after opening the malware sample with **IDA** was to check the “view” tab in the menu, which contains a list of Strings, Imports, Exports, Names, and Functions under the same location. Examining the Exports reveals only one function, the start function, which is the main entry into the executable once it has been executed. The first particularly interesting call was to “sub_401FE7” at location address “004078E9”, this is because there was referencing to an executable file named “tasksche.exe”. This can be seen in Figure 19.

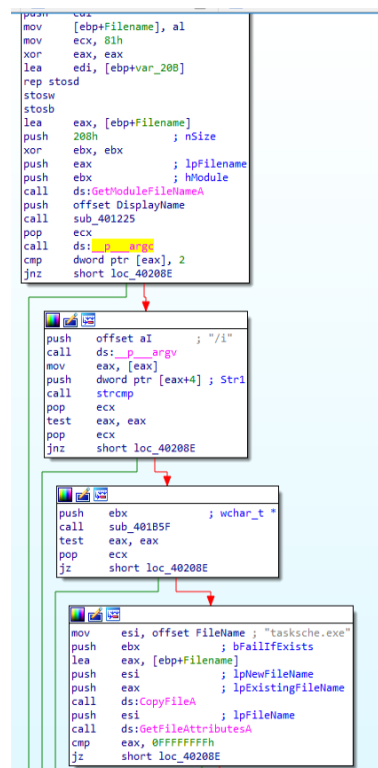


Figure 19 IDA - jumps between logic blocks leading to tasksche.exe

The logic block containing “taskche.exe” checks if the file exists already and if it doesn’t then it would copy the file, additionally if the comparison of the register after calling GetFileAttributesA and subtracting -1 was equal to zero, then it would jump to “loc_40208E”. At the new logic block, the executable would set the memory pointer to the last occurrence of the character “\” and would then move to “loc_4020B4”, which begins by changing the current directory and running a subroutine named “sub_4010FD”. This can be seen in figure 20.

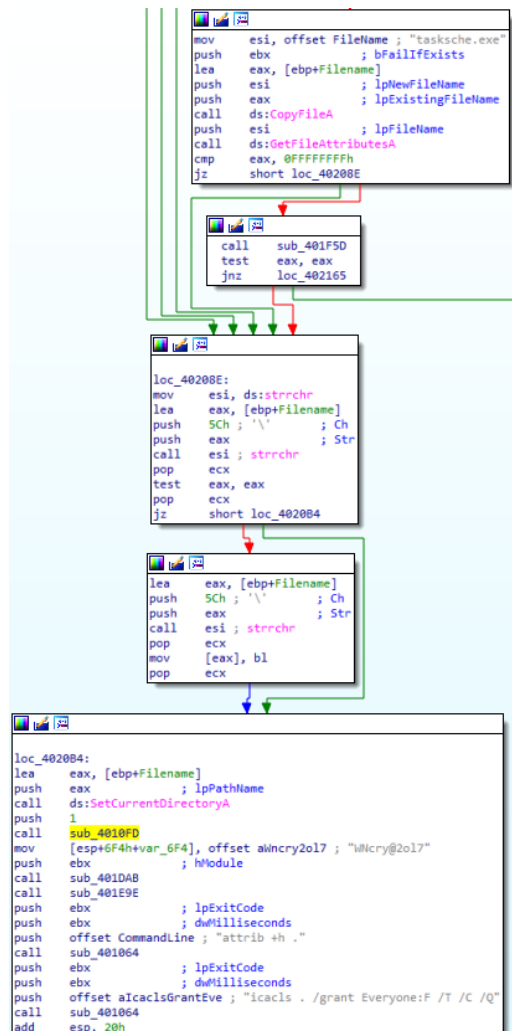


Figure 20 IDA - Logic Blocks after tasksche.exe

Within the subroutine “sub_4010FD”, contains three comments “Software”, “WanaCrypt0r” and “wd”, this further hints towards the malware sample being WannaCry; the instructions around the location addresses of the of three comments indicate the executable was creating a new registry key named “wd” under the Windows Registry, “SOFTWARE”. The subroutine shows further indications of attempts to creating new registry keys and editing them using functions such as “RegCreateKeyW”, “RegQueryValueExA”, and “RegSetValueExA”. This can be seen in Figures 21 and 22.

```

; Attributes: bp-based frame

sub_4010FD proc near

Buffer= byte ptr -20Ch
var_208= byte ptr -208h
Destination= word ptr -804h
var_C0= byte ptr -0C0h
cbData= dword ptr -0Ch
var_8= dword ptr -8
phkResult= dword ptr -4
arg_0= dword ptr 8

push    ebp
mov     ebp, esp
sub     esp, 20Ch
push    esi
push    edi
push    5
mov     esi, offset aSoftware ; "Software\\"
pop     ecx
lea     edi, [ebp+Destination]
rep movsd
push    20h ; '-'
xor     eax, eax
and     [ebp+Buffer], al
pop     ecx
lea     edi, [ebp+var_C0]
and     [ebp+phkResult], 0
rep stosd
mov     ecx, 81h
lea     edi, [ebp+var_208]
rep stosd
stosw
stosb
lea     eax, [ebp+Destination]
push    offset Source ; "WanaCrypt0r"
push    eax ; Destination
call    ds:wscat
and     [ebp+var_8], 0
pop     ecx
pop     ecx
mov     edi, offset ValueName ; "wd"

```

Figure 21 IDA - Comments possibly identifying the malware sample

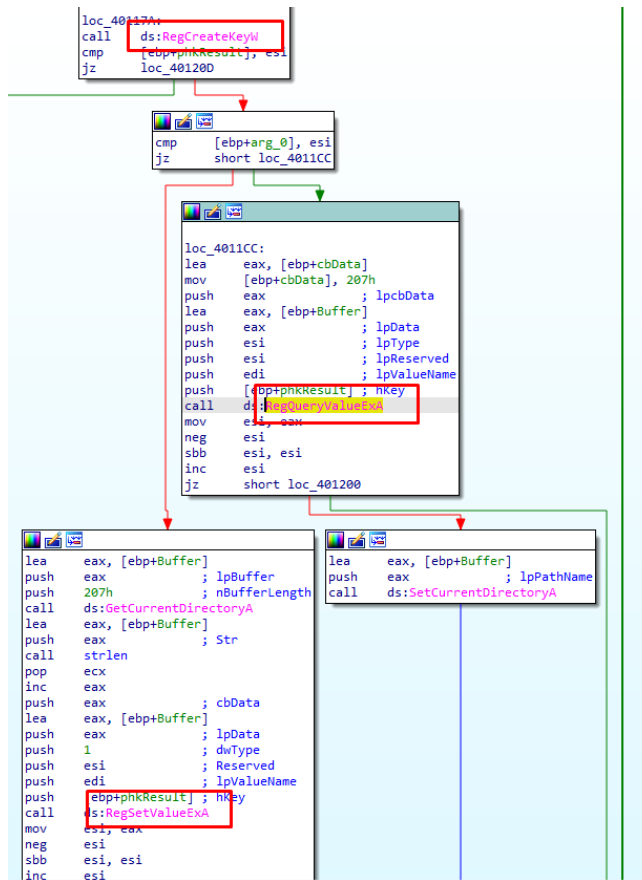


Figure 22 IDA - Attempts to create and edit registry key

Once this subroutine had finished and returned to the logic block at “loc_4020B4”, it would copy the value at the address “WNCry@20l7” and call two subroutines, “sub_401DAB” and “sub_401E9E”. This can be seen in Figure 20. The first subroutine called the function “FindResourceA” to find the file type “XIA” which was first discovered by the analyst in Figure 16. Once it finds the resource the subroutine would load the resource and push the file “c.winry” onto the stack. This can be seen in Figures 23 and 24.

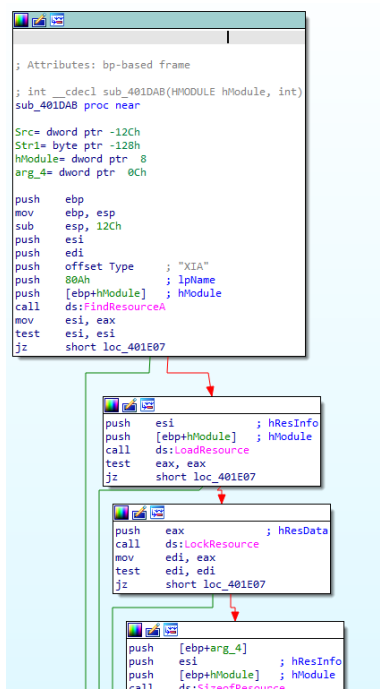


Figure 23 IDA - File type XIA loaded resource

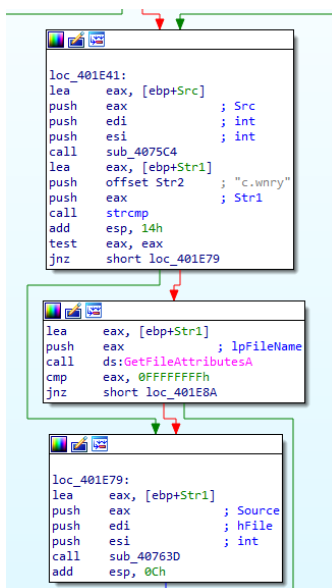
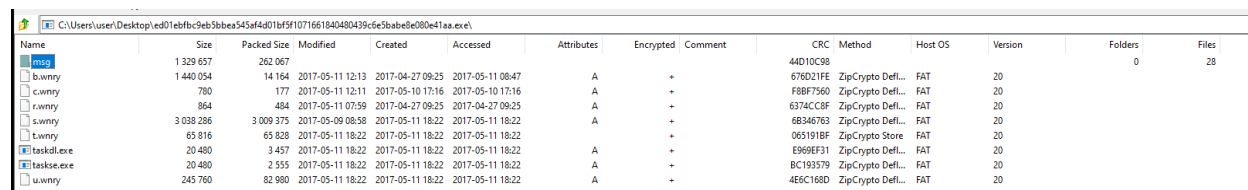


Figure 24 IDA - c.winry file pushed onto stack

Based on the evidence of a ZIP filetype within the malware sample, the analyst attempted to use **7zip** on the malware sample and chose the option “open archive” which reveals multiple files within the executable, with a numerous amount using the “.wnry” extension. This can be seen in Figure 25.



| Name | Size | Packed Size | Modified | Created | Accessed | Attributes | Encrypted | Comment | CRC | Method | Host OS | Version | Folders | Files |
|------------|-----------|-------------|------------------|------------------|------------------|------------|-----------|---------|----------|-------------------|---------|---------|---------|-------|
| mso | 1 329 657 | 262 067 | | | | | | | 44D10C98 | | | | | |
| b.wnry | 1 440 054 | 14 164 | 2017-05-11 12:13 | 2017-04-27 09:25 | 2017-05-11 08:47 | A | + | | 676D21FE | ZipCrypto Defl... | FAT | 20 | 0 | 28 |
| c.wnry | 780 | 177 | 2017-05-11 12:11 | 2017-05-10 17:16 | 2017-05-10 17:16 | A | + | | F8BF7560 | ZipCrypto Defl... | FAT | 20 | | |
| r.wnry | 864 | 484 | 2017-05-11 07:59 | 2017-04-27 09:25 | 2017-04-27 09:25 | A | + | | 6374CC8F | ZipCrypto Defl... | FAT | 20 | | |
| s.wnry | 3 038 286 | 3 009 375 | 2017-05-09 08:58 | 2017-05-11 18:22 | 2017-05-11 18:22 | A | + | | 6B346763 | ZipCrypto Defl... | FAT | 20 | | |
| t.wnry | 65 816 | 65 828 | 2017-05-11 18:22 | 2017-05-11 18:22 | 2017-05-11 18:22 | A | + | | 065191BF | ZipCrypto Store | FAT | 20 | | |
| taskl.exe | 20 480 | 3 457 | 2017-05-11 18:22 | 2017-05-11 18:22 | 2017-05-11 18:22 | A | + | | E969EF31 | ZipCrypto Defl... | FAT | 20 | | |
| taskse.exe | 20 480 | 2 555 | 2017-05-11 18:22 | 2017-05-11 18:22 | 2017-05-11 18:22 | A | + | | BC193579 | ZipCrypto Defl... | FAT | 20 | | |
| u.wnry | 245 760 | 82 980 | 2017-05-11 18:22 | 2017-05-11 18:22 | 2017-05-11 18:22 | A | + | | 4E6C168D | ZipCrypto Defl... | FAT | 20 | | |

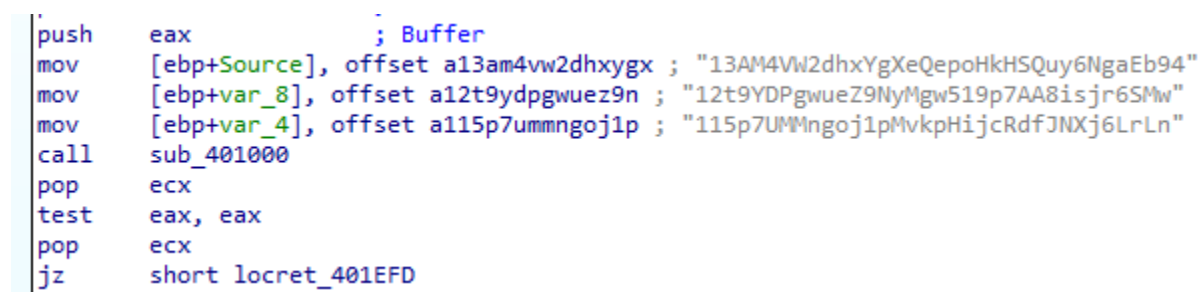
Figure 25 7ZIP - open archive on executable

Attempting to open the file brings up the “enter password” popup for **7Zip**; From the findings of **IDA**, the analyst tried the value “WNcry@2oI7” found within Figure 20. Entering this value works and the analyst had access to the contents of the encrypted ZIP file, which was originally detected using the tool **PeStudio**, as shown in Figure 16. This can be seen in Figure 26.



Figure 26 7ZIP - Contents of c.wnry

The second subroutine after “aWncry2oI7” within “loc_4020B4” copies three seemingly random strings of equal length; the three strings were “13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94”, “12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw”, “115p7UMMngo1pMvvpHijcRdfJNXj6LrLn”. These can be seen in Figure 27.



```

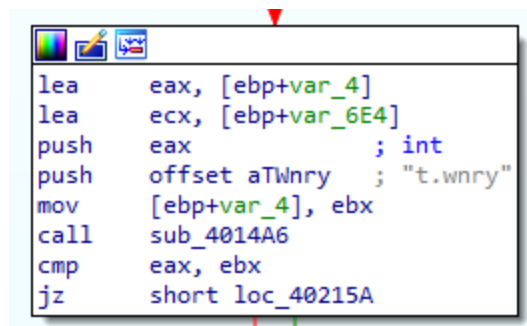
push    eax                ; Buffer
mov     [ebp+Source], offset a13am4vw2dhxygx ; "13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94"
mov     [ebp+var_8], offset a12t9ydpgwueZ9n ; "12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw"
mov     [ebp+var_4], offset a115p7ummngo1p ; "115p7UMMngo1pMvvpHijcRdfJNXj6LrLn"
call    sub_401000
pop     ecx
test    eax, eax
pop     ecx
jz      short locret_401EFD

```

Figure 27 IDA - Seemingly random strings

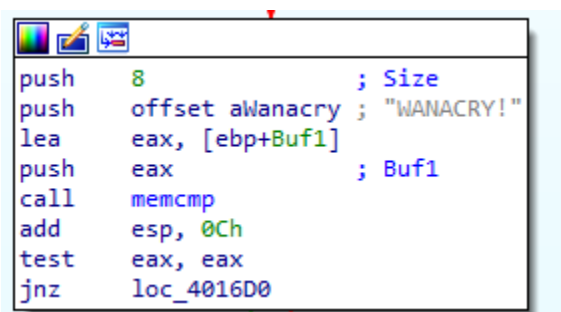
Once both subroutines were finished, two commands were pushed and executed, “attrib +h .”, which makes the file attributes as hidden and not visible to the user, and “icacis . /grant Everyone:F /T /C /Q”, which grants all users full access controls. These can be seen in Figure 20.

At memory location “00402125” it appears that a file named “t.wnry” was being pushed onto the stack and used within the next subroutine; the subroutine “sub_4014A6” calls the function “CreateFileA” to attempt to open a file if it already exists, this is indicated by the option for the function, “dwCreationDisposition” being pushed as value 3. Additionally, the value “8” was pushed onto the stack along with an offset with the value “WANACRY!”. This can be seen in Figures 28 and 29.



```
lea    eax, [ebp+var_4]
lea    ecx, [ebp+var_6E4]
push   eax                ; int
push   offset aTWnry      ; "t.wnry"
mov     [ebp+var_4], ebx
call   sub_4014A6
cmp     eax, ebx
jz      short loc_40215A
```

Figure 28 IDA - t.wnry



```
push    8                ; Size
push    offset aWanacry   ; "WANACRY!"
lea     eax, [ebp+Buf1]
push    eax                ; Buf1
call    memcmp
add     esp, 0Ch
test    eax, eax
jnz     loc_4016D0
```

Figure 29 IDA - WANACRY! pushed after size 8

Figure 29 appeared to be checking the header of the file “t.wnry” to check that it had the heading “WANACRY!” before using it. However, at this stage in the analysis, cannot be confirmed since the file was encrypted.

Nothing else of interest was discovered through either static analysis or **IDA** by the analyst; the next step was to use the information gathered from static analysis and disassembly to begin the dynamic analysis of the malware sample.

2.5 DYNAMIC ANALYSIS

Dynamic analysis is conducted by analysing the code whilst it is running, to investigate the behaviour of the executable; The analyst used dynamic analysis to gather hints towards the functionality and behaviour of the malware while it is running on the system. Before executing the malware, the tools, **Process Monitor**, **Process Explorer**, **Regshot**, **inetsim**, and **Wireshark** were setup to capture requests to different services and discover changes to processes and registry within the Windows virtual machine.

Upon executing the malware sample, the analyst is met with a popup window that is named “wana DecryptOr 2.0” and has a bitcoin wallet address which is used to buy bitcoin, this same address was found first using **IDA** in figure 27. The popup window can be seen in figure 30.



Figure 30 Popup Window - Wana Decrypt0r 2.0

2.5.1 Process Monitor

The first step in dynamic analysis was to set up the different **Process Monitor**; **Process Monitor** is used to observe the creation or termination of a process. The tool utilises non-destructive filters, which provides an efficient method to target a specific process. Before executing the malware sample, the analyst Set the filter to the name of the executable to specifically observe all processes related to the executable.

The popup in Figure 28 had an option to select between different languages; the files for the different languages can be found from **Process Monitor**. Within the tool, there was numerous “.wnry” files being created for different languages and the file “.t.wnry” that was found within **IDA** in Figure 28 was created when the malware was executed, which indicates the file extension is used specifically by the malware sample. This can be seen in Figures 31 and 32.

| | | | | |
|---------|---|------|--------------------|---|
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | CreateFile | C:\Users\user\Desktop\msg |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | CreateFile | C:\Users\user\Desktop\msg\vn_bulgarian.wm |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | WriteFile | C:\Users\user\Desktop\msg\vn_bulgarian.wm |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | WriteFile | C:\Users\user\Desktop\msg\vn_bulgarian.wm |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | WriteFile | C:\Users\user\Desktop\msg\vn_bulgarian.wm |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | SetBasicInfor... | C:\Users\user\Desktop\msg\vn_bulgarian.wm |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | CloseFile | C:\Users\user\Desktop\msg\vn_bulgarian.wm |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | CreateFile | C:\Users\user\Desktop |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | QueryBasicInfor... | C:\Users\user\Desktop |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | CloseFile | C:\Users\user\Desktop |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | CloseFile | C:\Users\user\Desktop |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | CreateFile | C:\Users\user\Desktop\msg |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | QueryBasicInfor... | C:\Users\user\Desktop\msg |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | CloseFile | C:\Users\user\Desktop\msg |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | QueryBasicInfor... | C:\Users\user\Desktop\msg |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | CloseFile | C:\Users\user\Desktop\msg |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | CreateFile | C:\Users\user\Desktop\msg\vn_chinese (simplified).wm |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | WriteFile | C:\Users\user\Desktop\msg\vn_chinese (simplified).wm |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | WriteFile | C:\Users\user\Desktop\msg\vn_chinese (simplified).wm |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | WriteFile | C:\Users\user\Desktop\msg\vn_chinese (simplified).wm |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | SetBasicInfor... | C:\Users\user\Desktop\msg\vn_chinese (simplified).wm |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | CloseFile | C:\Users\user\Desktop\msg\vn_chinese (simplified).wm |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | CreateFile | C:\Users\user\Desktop |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | QueryBasicInfor... | C:\Users\user\Desktop |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | CloseFile | C:\Users\user\Desktop |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | CreateFile | C:\Users\user\Desktop |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | QueryBasicInfor... | C:\Users\user\Desktop\msg |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | CloseFile | C:\Users\user\Desktop\msg |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | CreateFile | C:\Users\user\Desktop\msg |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | QueryBasicInfor... | C:\Users\user\Desktop\msg |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | CloseFile | C:\Users\user\Desktop\msg |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | CreateFile | C:\Users\user\Desktop\msg\vn_chinese (traditional).wm |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | WriteFile | C:\Users\user\Desktop\msg\vn_chinese (traditional).wm |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | WriteFile | C:\Users\user\Desktop\msg\vn_chinese (traditional).wm |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | WriteFile | C:\Users\user\Desktop\msg\vn_chinese (traditional).wm |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | WriteFile | C:\Users\user\Desktop\msg\vn_chinese (traditional).wm |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | SetBasicInfor... | C:\Users\user\Desktop\msg\vn_chinese (traditional).wm |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | CloseFile | C:\Users\user\Desktop\msg\vn_chinese (traditional).wm |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | CreateFile | C:\Users\user\Desktop |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | QueryBasicInfor... | C:\Users\user\Desktop |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | CloseFile | C:\Users\user\Desktop |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | CreateFile | C:\Users\user\Desktop |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | QueryBasicInfor... | C:\Users\user\Desktop |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | CloseFile | C:\Users\user\Desktop |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | CreateFile | C:\Users\user\Desktop\msg |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | QueryBasicInfor... | C:\Users\user\Desktop\msg |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | CloseFile | C:\Users\user\Desktop\msg |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | CreateFile | C:\Users\user\Desktop\msg\vn_croatian.wm |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | WriteFile | C:\Users\user\Desktop\msg\vn_croatian.wm |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | WriteFile | C:\Users\user\Desktop\msg\vn_croatian.wm |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | WriteFile | C:\Users\user\Desktop\msg\vn_croatian.wm |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | SetBasicInfor... | C:\Users\user\Desktop\msg\vn_croatian.wm |
| 1047... | ed01ebfbc9eb5bbea545af4d01f91071661840480433c6e5babe9e080e41aa... | 3636 | CloseFile | C:\Users\user\Desktop\msg\vn_croatian.wm |

Figure 31 Procmon - ".wm" files for languages

| | | | | | |
|----------------------|------|------------------|----------------------------|---------|-------------------------------------|
| 6e5babe9e080e41aa... | 3636 | CreateFile | C:\Users\user\Desktop\t.wm | SUCCESS | Desired Access: Generic Read, D |
| 6e5babe9e080e41aa... | 3636 | QueryStandard... | C:\Users\user\Desktop\t.wm | SUCCESS | AllocationSize: 69,632, EndOfFile |
| 6e5babe9e080e41aa... | 3636 | ReadFile | C:\Users\user\Desktop\t.wm | SUCCESS | Offset: 0, Length: 8, Priority: Nom |
| 6e5babe9e080e41aa... | 3636 | ReadFile | C:\Users\user\Desktop\t.wm | SUCCESS | Offset: 8, Length: 4 |
| 6e5babe9e080e41aa... | 3636 | ReadFile | C:\Users\user\Desktop\t.wm | SUCCESS | Offset: 12, Length: 256 |
| 6e5babe9e080e41aa... | 3636 | ReadFile | C:\Users\user\Desktop\t.wm | SUCCESS | Offset: 268, Length: 4 |
| 6e5babe9e080e41aa... | 3636 | ReadFile | C:\Users\user\Desktop\t.wm | SUCCESS | Offset: 272, Length: 8 |
| 6e5babe9e080e41aa... | 3636 | ReadFile | C:\Users\user\Desktop\t.wm | SUCCESS | Offset: 280, Length: 65,536 |
| 6e5babe9e080e41aa... | 3636 | CloseFile | C:\Users\user\Desktop\t.wm | SUCCESS | |

Figure 32 Procmon - "t.wm" created by malware Sample

2.5.2 Process Explorer

The tool "Process Explorer" is used to provide further information about the running processes and is displayed in a tree structure with child and parent relationships. The analyst used this tool to gather further information about the malware sample and its components. Upon opening the tool and examining the executable, the description, "DiskPart" was of interest to the investigation as it had the same name as the executable found from both Strings analysis and **Resource Hacker** in Figures 10 and 15 which indicates that the malware sample is attempting to disguise itself as DiskPart.exe. This can be seen in Figure 33.

| | | | | | |
|----------------------------|--------|---------|----------|----------------------------|-----------------------|
| ed01ebfbc9eb5bbea545af4... | < 0.01 | 7,600 K | 4,972 K | 3636 DiskPart | Microsoft Corporation |
| @WanaDecryptor@.exe | < 0.01 | 2,872 K | 11,224 K | 6896 Load PerfMon Counters | Microsoft Corporation |

Figure 33 Procexp - Description contains DiskPart

Additionally, clicking “Check Payment” while using the tool’s TCP/IP tab, indicates an attempt at a request to the internet and once it had finished checking, the connection is removed. This can be seen in Figure 34.

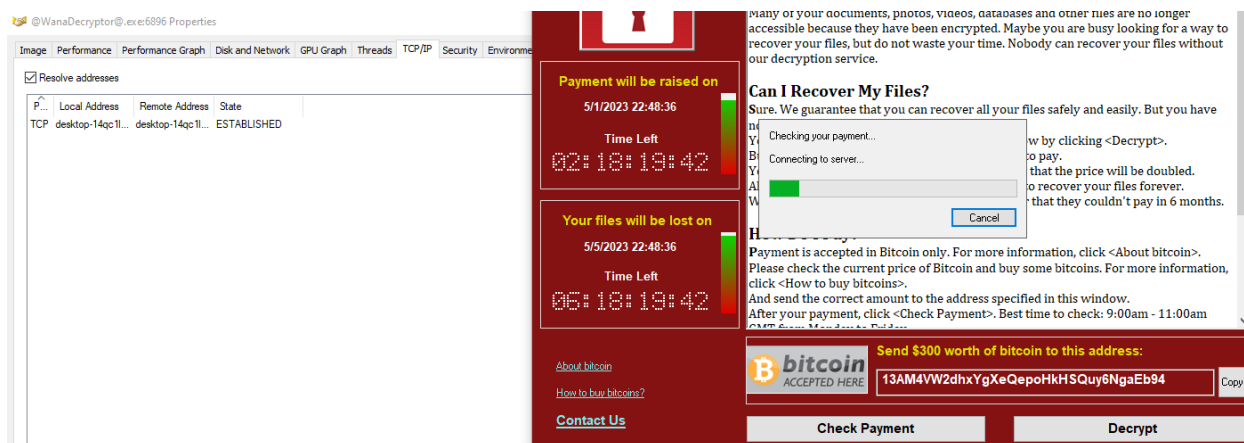


Figure 34 Procexp - Connection established

2.5.3 Regshot

After using **Process Explorer** the analyst used the tool **Regshot**; **Regshot** captures two snapshots of the Windows Registry, one before the execution of the malware and one snapshot after the malware has been executed. The two snapshots are compared to identify changes made within the Windows Registry or if files have been dropped by the malicious executable. Once the analyst had taken the two snapshots, the built-in compare feature was utilised to examine the changes. This can be seen in Figures 35, 36 and 37.

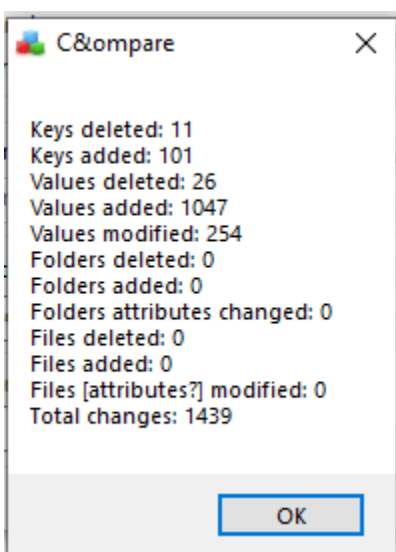


Figure 35 Regshot - Compare overview

```

HKU\S-1-5-21-2169232433-3398496680-935370409-1000\Software\Microsoft\Windows Script Host
HKU\S-1-5-21-2169232433-3398496680-935370409-1000\Software\Microsoft\Windows Script Host\Settings
HKU\S-1-5-21-2169232433-3398496680-935370409-1000\Software\Classes\Local Settings\VirtualCache\XSCProgram Files\XSC\WindowsApps\XSC\Microsoft.Windows.Photos_2022.30120.12007.0_x64__8wekyb3d8bbwe\XSC\Microsoft.system.package.metadata\XSCS-1-5-21-2169232433-3398496680-935370409-1000\Software\Classes\VirtualStore\MACHINE
HKU\S-1-5-21-2169232433-3398496680-935370409-1000\Software\Classes\VirtualStore\MACHINE\SOFTWARE
HKU\S-1-5-21-2169232433-3398496680-935370409-1000\Software\Classes\VirtualStore\MACHINE\SOFTWARE\Wow6432Node
HKU\S-1-5-21-2169232433-3398496680-935370409-1000\Software\Classes\VirtualStore\MACHINE\SOFTWARE\Wow6432Node\WanaCrypt0r
HKU\S-1-5-21-2169232433-3398496680-935370409-1000\Software\Classes\VirtualStore\MACHINE\SOFTWARE\Wow6432Node\WanaCrypt0r
HKU\S-1-5-21-2169232433-3398496680-935370409-1000\Classes\VirtualStore\MACHINE\SOFTWARE\Wow6432Node
HKU\S-1-5-21-2169232433-3398496680-935370409-1000\Classes\VirtualStore\MACHINE\SOFTWARE\Wow6432Node\WanaCrypt0r
HKU\S-1-5-18\Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts\*.bmp\UserChoice
HKU\S-1-5-18\Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts\*.bmp\UserChoice

```

Figure 36 Regshot - Keys added

```

HKU\S-1-5-21-2169232433-3398496680-935370409-1000\Software\Microsoft\Windows\CurrentVersion\Run\osnhnowfratdjot119: ""C:\Users\user\Desktop\Tasksche.exe""
HKU\S-1-5-21-2169232433-3398496680-935370409-1000\Software\Microsoft\Windows\CurrentVersion\Run\osnhnowfratdjot119: ""C:\Users\user\Desktop\Tasksche.exe""

```

Figure 37 Regshot - Key added Tasksche.exe

From the output, it appeared that new keys had been created with the Windows Registry which was originally indicated from **IDA** in Figures 21 and 22. Looking for the registry keys using **Registry Editor** provides further evidence towards the creation of these files. This can be seen in Figure 38.

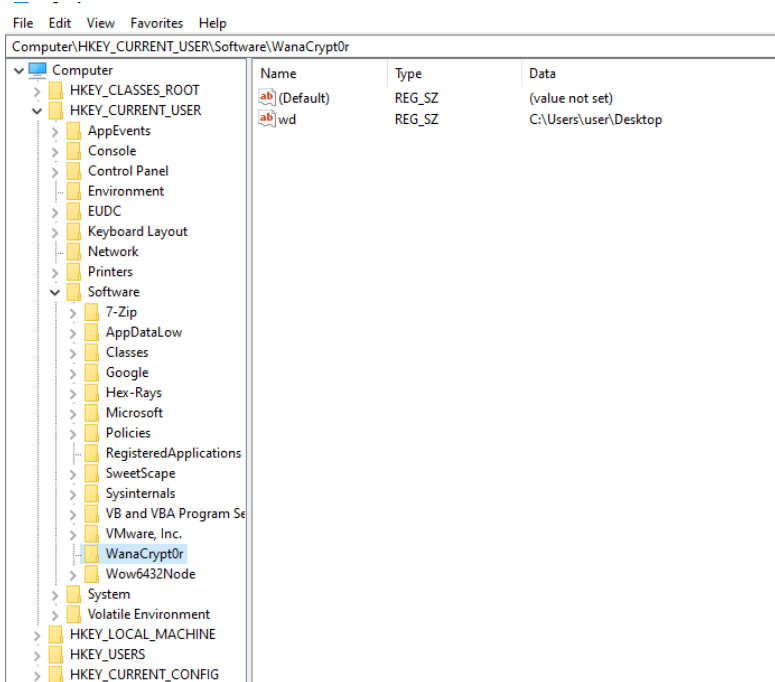


Figure 38 Registry Editor - Evidence of new registry keys

2.5.4 Inetsim

Occasionally Malware will not work without an internet connection because the payload of the malware is not contained within the file and requires an internet connection to request the payload; **INetSim** is used to mimic multiple internet services for investigation, such as DNS, FTP, or HTTP. The tool is constantly run on the second virtual machine that was set up during the initial steps of setting up the lab environment.

2.5.5 Network Traffic Analysis

While **inetsim** is used to simulate multiple internet services, A different tool was used to analyse the traffic; **Wireshark** was used to inspect captured packets from the interface connected between the two

virtual machines to investigate how the malware is communicating with different services. **Wireshark** had captured packets with destinations to various IP addresses. This can be seen in Figure 39.

| | | | | | |
|-----|--------------|-----------------|-----------------|-----|--|
| 99 | 52.956642999 | 10.0.0.3 | 86.59.21.38 | TCP | 66 [TCP Retransmission] [TCP Port numbers reused] 50683 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 100 | 52.056292840 | 10.0.0.3 | 93.180.156.84 | TCP | 66 [TCP Retransmission] [TCP Port numbers reused] 50682 → 9001 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 102 | 52.994241525 | 10.0.0.3 | 78.24.75.53 | TCP | 66 [TCP Retransmission] [TCP Port numbers reused] 50684 → 9001 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 109 | 58.940606341 | 10.0.0.3 | 193.35.52.53 | TCP | 66 50680 → 9001 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 110 | 58.948170832 | 10.0.0.3 | 171.25.193.9 | TCP | 66 50687 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 112 | 59.963188699 | 10.0.0.3 | 193.35.52.53 | TCP | 66 [TCP Retransmission] [TCP Port numbers reused] 50686 → 9001 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 113 | 59.963189039 | 10.0.0.3 | 171.25.193.9 | TCP | 66 [TCP Retransmission] [TCP Port numbers reused] 50687 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 116 | 61.978648568 | 10.0.0.3 | 193.35.52.53 | TCP | 66 [TCP Retransmission] [TCP Port numbers reused] 50686 → 9001 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 117 | 61.978648948 | 10.0.0.3 | 171.25.193.9 | TCP | 66 [TCP Retransmission] [TCP Port numbers reused] 50687 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 122 | 64.681481865 | Vmware 55:f6:05 | Vmware 3f:36:9e | ARP | 60 Who has 10.0.0.4? Tell 10.0.0.3 |
| 123 | 64.681495175 | Vmware 3f:36:9e | Vmware 55:f6:05 | ARP | 42 10.0.0.4 is at 00:0c:29:3f:36:9e |
| 125 | 65.978572254 | 10.0.0.3 | 193.35.52.53 | TCP | 66 [TCP Retransmission] [TCP Port numbers reused] 50688 → 9001 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 126 | 65.978572274 | 10.0.0.3 | 171.25.193.9 | TCP | 66 [TCP Retransmission] [TCP Port numbers reused] 50687 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 127 | 66.197544815 | 10.0.0.3 | 217.79.179.177 | TCP | 66 50688 → 9001 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 129 | 67.205476741 | 10.0.0.3 | 217.79.179.177 | TCP | 66 [TCP Retransmission] [TCP Port numbers reused] 50688 → 9001 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 179 | 69.210724060 | 10.0.0.3 | 217.79.179.177 | TCP | 66 [TCP Retransmission] [TCP Port numbers reused] 50688 → 9001 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 231 | 72.225640468 | 10.0.0.3 | 217.79.179.177 | TCP | 66 [TCP Retransmission] [TCP Port numbers reused] 50688 → 9001 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 233 | 74.007331777 | 10.0.0.3 | 193.35.52.53 | TCP | 66 [TCP Retransmission] [TCP Port numbers reused] 50686 → 9001 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 234 | 74.007332167 | 10.0.0.3 | 171.25.193.9 | TCP | 66 [TCP Retransmission] [TCP Port numbers reused] 50687 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 241 | 78.178888116 | Vmware 55:f6:05 | Vmware 3f:36:9e | ARP | 60 Who has 10.0.0.4? Tell 10.0.0.3 |

Figure 39 Wireshark - TOR IP addresses

Using the tool **host** with the command “host -t PTR <host>” on the different IP addresses indicated that all the addresses are linked with TOR servers, which matches with findings from **IDA** as seen in Figure 26, where a file named “c.wnry” contains links to different “.onion” addresses; Onion addresses are used to access hidden services that are only accessible through the TOR network (torProject, n.d.). This can be seen in Figure 40.

```
adam@PandorasBox:/mnt/c/Windows/system32$ host -t PTR 78.24.75.53
53.75.24.78.in-addr.arpa domain name pointer tor-relay2.rofl.cat.
adam@PandorasBox:/mnt/c/Windows/system32$ host -t PTR 86.59.21.38
38.21.59.86.in-addr.arpa is an alias for 38.32-29.21.59.86.in-addr.arpa.
38.32-29.21.59.86.in-addr.arpa domain name pointer tor.noreply.org.
```

Figure 40 Command Line - Host command detecting TOR servers

When the malware was first executed, it attempted to download updates using the user-agent Microsoft-CryptoAPI/10.0. which can be seen in multiple TCP streams. CryptoAPI/10.0 is used to encrypt and decrypt messages and data (Microsoft, 2023). This can be seen in Figures 41.

```
GET /msdownload/update/v3/static/trustedr/en/authrootst1.cab?a14e0c4b54c6b780 HTTP/1.1
Connection: Keep-Alive
Accept: */*
User-Agent: Microsoft-CryptoAPI/10.0
Host: ctldl.windowsupdate.com

HTTP/1.1 200 OK
Content-Length: 258
Connection: Close
Content-Type: text/html
Date: Sat, 29 Apr 2023 03:34:32 GMT
Server: INetSim HTTP Server

<html>
<head>
<title>INetSim default HTML page</title>
</head>
<body>
<p></p>
<p align="center">This is the default HTML page for INetSim HTTP server fake mode.</p>
<p align="center">This file is an HTML document.</p>
</body>
</html>
```

Figure 41 Wireshark - TCP stream from VM1 to VM2

2.5.6 Debuggers

While disassemblers and debuggers are similar in what they offer such as lists of functions and strings, debuggers allow in-depth monitoring of malicious code execution while observing memory, registers, stacks, and control elements. Another benefit of using a debugger is the opportunity to control the execution of the code during the analysis. For the report, the debugger, **x32dbg** will be used.

Once the malicious file was uploaded into **x32dbg**, the analyst used the “search for” option to find all string references that is contained within the malware sample. This can be seen in Figure 42.

| | | |
|----------|---|---|
| 00401016 | push ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40E010 | "C:\wrry" |
| 00401016 | mov esi,ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40E04C | "Software\l" |
| 00401145 | push ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40E034 | "L\wanacryptor" |
| 00401265 | mov esi,dword ptr ds:[<dwcs1en>] | "e\l\w" |
| 004012A7 | mov ebx,dword ptr ds:[<arand>] | "&?c" |
| 004012F0 | mov eax,ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40F966 | "000" |
| 00401348 | mov dword ptr ds:[esi],ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.4081D8 | "&v\Re\X14" |
| 00401388 | mov dword ptr ds:[esi],ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.4081D8 | "&v\Re\X14" |
| 00401566 | push ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40E07C | "WANACRY!" |
| 00401727 | push ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40E0E3 | "kernel32.dll" |
| 00401743 | push ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40E0DC | "CreateFileW" |
| 00401748 | push ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40E0D0 | "WriteFile" |
| 00401758 | push ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40E0C4 | "ReadFile" |
| 00401765 | push ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40E0B8 | "MoveFileW" |
| 00401772 | push ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40E0AC | "MoveFileExW" |
| 00401777 | push ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40E0A0 | "DeleteFileW" |
| 0040178C | push ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40E094 | "CloseHandle" |
| 004017E7 | mov dword ptr ds:[esi],ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.4081EC | "&v\Re\X14" |
| 00401818 | mov dword ptr ds:[ecx],ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.4081EC | "Microsoft Enhanced RSA and AES Cryptographic Provider" |
| 00401840 | and eax,ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40F08C | "advapi32.dll" |
| 00401A55 | push ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40E020 | "CryptAcquireContextA" |
| 00401A73 | push ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40F100 | "CryptImportKey" |
| 00401A86 | push ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40F0F0 | "CryptDestroyKey" |
| 00401A93 | push ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40F0E0 | "CryptEncrypt" |
| 00401AA0 | push ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40F0D0 | "CryptDecrypt" |
| 00401A0D | push ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40F0C4 | "CryptGenKey" |
| 00401B46 | push ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40E0B8 | "L\%s\%s" |
| 00401BF6 | push ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40F40C | "L\%s\ProgramData" |
| 00401C40 | push ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40F3F8 | "L\%s\Intel" |
| 00401C4E | push ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40F42C | "cmd.exe /c \"%s\"" |
| 00401D86 | push ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40F43C | "XIA" |
| 00401E55 | push ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40E010 | "C:\wrry" |
| 00401E80 | mov dword ptr ss:[ebp-C],ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40F488 | "1344W2dzhxygxeqepohKHS0uy6NqaEb94" |
| 00401EB7 | mov dword ptr ss:[ebp-8],ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40F464 | "12t9vDPwue29NyMgw519p7AA813jr69Mw" |
| 00401EBE | mov dword ptr ss:[ebp-4],ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40F440 | "115p7Uwmg01j0hVKhP1jCRdF7N0J6LrL" |
| 00401F08 | push ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40F484 | "Global\WanZoneCacheCounterMutexA" |
| 00401F30 | push ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40F4AC | "nsld" |
| 00402061 | mov esi,ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40F4D8 | "tasksche.exe" |
| 004020C8 | mov dword ptr ss:[esp],ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40F52C | "Wncry2017" |
| 004020D6 | push ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40F520 | "attrib +h ." |
| 004020E8 | push ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40F4FC | "icacls . /grant Everyone:F /T /C /Q" |
| 00402125 | push ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c65babe8e080e41aa.40F4F4 | "t.wrry" |

Figure 42 x32dbg - List of string references

From the string references, there is indications of cryptography functions being used within the executable, such as “CryptImportKey” and “CryptEncryptKey”. The string “Microsoft Enhanced RSA and AES Cryptographic Provider” indicates that the malware uses Microsoft’s own RSA and AES cryptography functions.

The next step the analyst took was setting breakpoints throughout the code at important functions to monitor their state and to confirm what was found from the static analysis and disassembly. From analysing the different subroutines and functions, “t.wrry” was found to have the list of extensions and header “WANACRY!” within it, indicating that this file was used to as the method for encrypting all files within a host machine. This can be seen in Figure 43.

```
.x.l.t...x
l.w...x.l.s
b...x.l.s.m.
.x.l.s.x...x
l.s...d.o.t
x...d.o.t.m.
.d.o.t...d
o.c.m...d.o.c
b...d.o.c.x
.d.o.c...WANA
CRY!...%s.\%.
s...CloseHandle.
DeleteFileW.Move
FileExW.MoveFile
W...ReadFile...
WriteFile...Crea
teFileW.kernel32
.dll...R.
RSA2...C+M+
...U..U_i2@iA.P
ô.cq{i'v.X'ô.É¶
w[a.-.Öjy;p.?
&c-3ü.-c.c.c.c.p
```

Figure 43 x32dbg - contents of "t.wrry"

3 DISCUSSION

3.1 GENERAL DISCUSSION

3.1.1 VirusTotal

VirusTotal was able to identify the malware sample as Wannacry immediately based on the signature that was cross-referenced with numerous anti-virus software. Since VirusTotal uses multiple different anti-viruses to detect and flag the signature of the file as malware, most hosts would be safe from this malware sample and variants made from the Wannacry malware.

3.1.2 String Analysis

The readable ASCII and Unicode strings found using *Flare Floss* revealed the file extension, “.wnry” that the malware was using for its files which is a clear host-based indicator that a host has been affected by this malware. It also revealed a list of file extensions it was targeting for encryption which shows the attack scope of the malware and the level of damage it can cause if it is executed on a host. Additionally, the strings contain an executable named “DiskPart.exe” as well as the string “WannaCrypt0r” which could both be host-based indicators for the malicious file.

3.1.3 PEID

This tool does not explicitly give indications towards the functionality or impact of the malware, it does however indicate that the malware was possibly packed based on its entropy level which was not particularly helpful as it gave no definitive answer to the malware being packed or obfuscated.

3.1.4 CFF Explorer

CFF Explorer was used to find the compilation date of “Saturday, 20 November 2010 09:05:05”, which can be beneficial when creating a timeline of an attack campaign. However, an advanced threat could alter the compilation date to prevent the analyst from knowing the actual compilation date of the malware (O'Reilly, n.d.). Additionally, no signs of alterations to the four section headers from external software were present, which typically means that the malware itself is not packed.

3.1.5 Resource Hacker

Upon inspecting the resources within the PE file, it appeared to have two files embedded within it, an encrypted ZIP file and a Windows PE named “diskpart.exe”. However, after further analysis from other steps, it was clear the Windows PE was not an embedded file but was the disguise the malware was using to hide itself once executed. The encrypted ZIP file embedded within the resources typically indicates that the malware functions as a dropper that unpacks and drops the payload onto the machine once it has been run.

3.1.6 PeStudio

PeStudio was able to gather various host-based indicators that the PE file was malicious and rated the indicators on a scale of confidence from one to three, one being the most confident. The size of the resources being 98.13% of the size of the PE file was the most notable indicator the malware was a functioning as a dropper. Additionally, the 158 extensions within the PE file suggested to PeStudio that

the malware was a type of ransomware or wiper. The import function names that were grouped within “cryptography” on PeStudio was another potential host-based indicator of ransomware.

3.1.7 IDA

Using IDA, enabled an in-depth analysis by opening the file in assembly code to read and understand the functionality of the malware sample. The first potential host-based indicator found was the executable “tasksche.exe” which was created by the malware as a method of persistence.

The next host-based indicator that was located was the creation of the run registry key which was attached to the PE file “tasksche.exe”. The executable file found was used as a persistence function to allow the GUI popup menu to persist when the computer was restarted. Additionally, the registry key “WanaCrypt0r/wd” within the “SOFTWARE” registry is another indicator that can be used to detect the malware on a host.

The subroutine “sub_401DAB” finds the “XIA” resource which is an encrypted ZIP file within the executable and loaded it with the password “WNcry@2017” to access the ZIP file. With the ZIP file password entered, the malware sample acts as a dropper to drop the various files within it, such as “c.wnry” and “t.wnry”. Within the “c.wnry” file, was multiple onion addresses which are used for accessing TOR network services. The t.wnry file contents were encrypted but the file itself can be used as an indicator that the malware is on a host.

Additionally, two commands were run by the malware which enabled it to hide file attributes from all users and grant all users full access controls on the machine, which can be used to grant the malware full access to all files within the machine and assists the malware in persisting on the system.

Though the malware was just as malicious, this malware sample is not the original WannaCry ransomware as it does not have any worm functionality or “killswitch” to stop the malware from executing.

3.1.8 Process Monitor

Through the use of Process Monitor, there was indications of the PE file writing files with the extension “.wnry” to allow different languages within the popup window which would be another host-based indicator that the machine had been infected. Another indicator was the executable reading from the “t.wnry” file that was used to implement file-encryption functionality (Team, 2017).

3.1.9 Process Explorer

The tool, Process Explorer was able to confirm that the executable was attempting to disguise itself as “DiskPart.exe”, the Windows PE, which is a way the malware would evade detection from malware analysts. While clicking “Check Payment” gives a request attempt towards the internet, it doesn’t particularly work as a network-based indicator due to the malware having to be already activated and running to find it.

3.1.10 Regshot

Snapshots of the device before and after the execution of the malware was used to compare the Windows Registry to see what had been created or deleted, since the PE file was not executed with elevated privileges the registry key “wd”, was created inside the HKCU hive, which is a potential host-

based indicator to check for. If it was run with elevated privileges to begin with, it should have been created inside of the HKLM hive (Team, 2017).

The “Tasksche.exe” executable was found to enable persistence functionality by creating a run registry key that links to the executable file to open the GUI popup whenever the computer is restarted.

3.1.11 Network Traffic Analysis

Using Wireshark to capture packets going through the interface between the two virtual machines in the lab environment enabled the ability to see all network traffic; The captured packets and the “c.wnry” file, indicated the use of TOR network services which could be used as a possible method for communicating with a Command and Control centre and is a potential network-based indicator for detecting the malware.

3.1.12 Debuggers

From searching for anything related to cryptography within the debugger, the analyst discovered Microsoft’s RSA and AES cryptography was being utilised for encryption which is a potential indicator of ransomware within the system. Using the debugger the functionality for the file “t.wnry” was revealed to be the encryptor that was packaged within the encrypted ZIP file.

3.2 CONCLUSION

3.2.1 Conclusion

The conclusive results of the malware analysis on the malware sample provided by the organisation the analyst is under is that the malware can be confirmed as a variant of the ransomware, WannaCry; There was no evidence that the variant could worm its way through a network and it does not have a “killswitch” to stop it from executing, the malware once executed runs two commands to give itself full access control to the host and hide itself from users. The malware then creates its registry keys for persistence on the system and drops the ZIP file onto the host to begin the unpacking and encrypting process. Finally, once all the files that don’t affect the machine running have been encrypted using “t.wnry”, the malware will prompt the user to pay bitcoin into one of three bitcoin wallet addresses and sends traffic to one of the 5 TOR onion addresses which were found within “c.wnry”. The full list of files found from the malware analysis can be found within Appendix C.

3.2.2 Countermeasures

There are numerous countermeasures that can be used to mitigate the damage caused by the WannaCry variant; Updating the Microsoft security patch is an effective and cost-efficient method of protecting against all types of malware, especially malware that exploits security issues in older versions of software like how WannaCry and its variants typically use EternalBlue.

Another countermeasure is using the found host-based and network-based indicators to create Yara rules which can detect the malware variant in future attacks and can be uploaded onto a github so other users and analysts can use it in their system or investigation.

Additionally, having an offsite backup and incident response plan is an effective method of mitigating the impact and damage should the malware sample be executed on a host, this is so a user or organisation can rapidly recover from the attack without losses of data.

3.3 FUTURE WORK

Should a more in-depth and advanced analysis be carried out, the analyst would employ the use of sandboxing, evaluate already existing yara rules, and create custom yara rules for this malware sample. Additionally, the analyst would attempt to do further analysis with a debugger than was possible with the current timeframe given to the analyst.

Sandboxing limits the spread of infection and offers options to dump the process memory to better evaluate what is occurring in RAM. The analyst would use Cuckoo sandbox as it is the leading open-source automated malware analysis system and can be deployed locally. This sandbox has beneficial features such as, being able to take screenshots of the execution of the malware, intercept deleted and downloaded files, run concurrent analyses on multiple machines, generate and dump traffic, and acquire full memory dumps of the virtual machines (Ahmet, et al., 2020).

Yara rules are a method of spotting malware and other files by creating rules based on characteristics of the malware, which can be used to identify specific variants or families of malware (Arntz, 2017). The analyst would use premade Yara rules against the malware sample to detect the malware sample and if it does not work then the malware analyst would attempt to create their own set of Yara rules as a countermeasure for detecting the malware.

4 REFERENCES

Ahmet, B., Dan, U. & Jaromir, V., 2020. *Malware Reverse Engineering Handbook*. 1st ed. Tallinn: The NATO Cooperative Cyber Defence Centre of Excellence.

Allegretti, A., 2018. *sky news*. [Online]

Available at: <https://news.sky.com/story/cost-of-wannacry-cyber-attack-to-the-nhs-revealed-11523784>
[Accessed 19 April 2023].

Arntz, P., 2017. *malwarebytes*. [Online]

Available at: <https://www.malwarebytes.com/blog/news/2017/09/explained-yara-rules>
[Accessed 1 May 2023].

Cardona, M., 2021. *mimecast*. [Online]

Available at: <https://www.mimecast.com/blog/malware-analysis/>
[Accessed 19 April 2023].

Cisco, n.d. *cisco*. [Online]

Available at: <https://www.cisco.com/site/us/en/products/security/what-is-malware.html>
[Accessed 19 April 2023].

fileinfo, 2019. *fileinfo*. [Online]

Available at:
<https://fileinfo.com/extension/wnry#:~:text=A%20file%20with%20a%20WNRY,DOCX%2C%20>
[Accessed 22 April 2023].

filext, n.d. *filext*. [Online]

Available at: <https://filext.com/file-extension/XIA>
[Accessed 25 April 2023].

Fox, N., 2023. *Varonis*. [Online]

Available at: <https://www.varonis.com/blog/pestudio>
[Accessed 24 April 2023].

Johanns, K., 2020. *smartermsp*. [Online]

Available at: <https://smartermsp.com/the-creeper-and-the-reaper-make-cybersecurity-history/>
[Accessed 19 April 2023].

Lord, N., 2020. *digital Guardian*. [Online]

Available at: <https://www.digitalguardian.com/blog/cost-malware-infection-maersk-300-million>
[Accessed 19 April 2023].

mandiant, 2023. *github*. [Online]

Available at: <https://github.com/mandiant/flare-floss>
[Accessed 21 April 2023].

Microsoft, 2023. *learn microsoft*. [Online]

Available at: <https://learn.microsoft.com/en-us/windows-server/administration/windows->

commands/diskpart

[Accessed 22 April 2023].

Microsoft, 2023. *learn microsoft*. [Online]

Available at: <https://learn.microsoft.com/en-us/windows/win32/seccrypto/cryptoapi-system-architecture>

[Accessed 30 April 2023].

mrd0x, n.d. *malapi*. [Online]

Available at: <https://malapi.io/winapi/CryptReleaseContext>

[Accessed 24 April 2023].

O'Reily, n.d. *oreilly*. [Online]

Available at: <https://www.oreilly.com/library/view/learning-malware-analysis/9781788392501/c96fdc6c-92a7-4a28-8d1f-0715b8fcdd55.xhtml#:~:text=The%20PE%20header%20contains%20information,timeline%20of%20the%20attack%20campaign.>

[Accessed 30 April 2023].

PKWARE, n.d. *PKWARE*. [Online]

Available at: <https://www.pkware.com/company/history>

[Accessed 24 April 2023].

Root, E., 2022. *kaspersky*. [Online]

Available at: <https://www.cisco.com/site/us/en/products/security/what-is-malware.html>

[Accessed 19 April 2023].

Shepherd, A., 2019. *itpro*. [Online]

Available at: <https://www.itpro.com/malware/34381/what-is-notpetya>

[Accessed 19 April 2023].

Team, C. T. U. R., 2017. *Secureworks*. [Online]

Available at: <https://www.secureworks.com/research/wcry-ransomware-analysis>

[Accessed 30 April 2023].

The NATO Cooperative Cyber Defence Centre of Excellence, n.d. *ccdcoe*. [Online]

Available at: <https://ccdcoe.org/about-us/>

[Accessed 20 April 2023].

torProject, n.d. *tb-manual.torproject*. [Online]

Available at: <https://tb-manual.torproject.org/onion-services/>

[Accessed 30 April 2023].

APPENDICES

APPENDIX A – OMITTED SECTION OF METHODOLOGY

This appendix contains sections of the methodology “Malware Reverse Engineering Handbook” which have been omitted from the analysis. This is due to multiple reasons such as, not being a step to analyse malware, out of scope for analysing the malware, or similar tool used in the same section. The omitted sections can be found below.

- Abstract
- Why perform Malware analysis?
- Disassembly
 - Ghidra
- Dynamic analysis
 - Sandboxing
- Packed Executable/Unpacking
- Incident response collaboration (Misp & Yara)
- references

APPENDIX B – TOOL OUTPUT

4.1.1 Flare Floss

```
07B)V>
}q>*
k)g{
Ta@5D
NHZ6v
1c&h
+{ebM
'B2S
%kUp
Y>pj"PBQ
i4u\[X@
Y^--c
zrfF}g
/B.{
abQeyj
5fJ2
<qcn
IFt:z
[J-t0
TJfS
U`<I
```

1KmN
)+dO
\$zuIS
_uCE
fh"X
l7+)
&}N~
g'ht
sdO(
\^RGM
4baP
k^|=Q
s<iY
!^}HY
.+agD
]xZQ
76JD
nz=vk
,O^/
76nhV
""O-
wKk]
ArKlj
ZF|9
1<?z2
\vO
aO~2
|4yx
2m~+*b
y8vv
zGiv
*7/EAh
YwQX
Q!Nbu
2!|`
qSLU
)`e5
rTv{
nkDm
a,8i4T
R;Vq
5EzL9&
"CUu
T`3?
*Qhjev
+c{B
(a*>
UGR'.|

kd7j@
{0 g
\# ,
LNo_
d^g>
+4?\n
]rtI7Ik3j
w%hK
NL3I
k=vEJ
g>qm
@gmb
F|
pF8V/)
=wdY
{GFT
9RIBKS
=X_`
[N\OQ
*e){
+1\o
gSpuf
tM!XP#rx
*IEE
-b&t
[z6~
By 8
t;X9xVP
O}09H_
61g_
79UI
}`FaE~
"9x0
ak#O
"pX]W!g_
QRJX
#{YG
P#H=
93?m}
He8O
Fa#M
\$zs[=1
g.V}
6AdSPJM
cgr;R
Vn[:U
'?=W
ab[#

G,s]
Tudz
_le~
%aJ6
~7D)}4
X@q9w
!^iW
[BT,.
eMT{
bjM|
x\$f
J1aU~
o)f7G~
@\\^`Y
C"U-
jE-Nn
D5m~
pO4F
() a
G? [=pg
fKbK
6,^%F
At 1
1-Fx
Kdy%8ik
,y;f3
pUzW
X@uz!
#|(U
Jr?&
YU((
&q4w
bqe0
S7IVJ
!RLJ4
K7M^
2!`>
~GL=
<: G
r&E|
OT>s
uGi8
!W"_
MkFTL
;qatx
'e{9u-:d
B-8(
rs9C

BNG-
8R0/
p+0`Np
oB@U
lmgk
{5UP
YYPz
~K8t:
13o};
1<mo
a*7J
<1/2
}|su
R:'
"``w
l=Ga
li_!
bgMwF
f Gs)
v(VX
%3s=
j@Y1A
DaIP
|% 5ug
'WF|
Rblf
rkX.1
32mJ
+EB<1
ynd_
8>p3/
7J%c
e=+{Alb
jU%1
{ e#
%Q!4
qFvu Xq3
k*EZ
El3uNaOu
}+YC
`Fql
rCoW
f?vV
}L\${
P?uR9
J^;hvg
g|(Vr=S

8DbV
ur{wI7R
`kz|
;B0\<
{6t@O
nA-K
;L6u
t0[t
vduN
C) um
EE\T
8H)m
}uP{
{%BVfcU2
*XxV
6[II
lv{%M
]Qss
B4(v
ggVUj
D(0a
Tht~A
:wK=
tp^9
u{nIR
tZfD
-W0{
ML(Y
};R{
")][w
t 9jg
5>(~
2\P&Z,
CgzY
B(5B
'p[i
[Ccx0
\.l"f
Fvc'
FG-l
lcqF
%T*
|Pe3
:kzj
:' %
?Da[
`q_~iG

trXP
5x~ho
p-yo
gkDK\$T
]<(,|
wz`
);"~4
jK*H
4V|<
[c<{
x @Z
a.S5
xo8r!
+ c7Y%
RZwUXK
|Rk>W5d
b2P3.Ct
vm,{R
L(0r3f
4WV|z
L ({v
]k,w
| eUX
~O,B
YnTQ
yWfOv
>r<J
n#2._
c4Nd
)(_QJ
#}Q.S
\$&X"
Ufk49y-
FQ@D]w
:>)u
Ijd>J
Ad+*
q~v`
vcy'
W^d2"/
b!XOnU
puGB
EU30
y}@*
zitsH
g;=j
K4#|
G-R@

HX:w
ics
d?h<
F9Sm{
w>lj
 V0vQ
Ltu2:6
r>Zz
"qN"
y{"
a7c<*
Mw +
^:zp
9h]C
%8z_X
 |PWJn
xWX7`{
+h',
;-Y\$G
\$xLd
}3Q)
tx`*
h&}IJ,
bZX}
 ;Ar
`Ub
Xo^;jW
N\$?.
<*o
CTRDa``
+Fx:
PEq:
Ur(1CZ
KWJ7
R\j~
^3b/
T/uVk
YZ,X~(j
efqL
HPD5
/at2
t+[W
WFcP
:kv0
Q\IU
uQmV
H8y8
a6O9

XtaWE
/{zx
W^T
!+%2
tg<}@ C5
_*g^
)^5++2Z
KNUV&x
2x95
jE"\
\wkZ
K}LU
RR}&j
ZobB
S*fe
;bZl
.\$...Te
1%<D
R:-_
]Jc<
G_"%
F_5^T
~W|j
HoKWi
[{s/
>%b~
~SJ2(T
pq4F
"|5O
/Pq\$.
i[U+
t#EYi
*!^!
S^Pq
:U7dp
`n>4
aBQ(
M{%~CJP@X
ay<J
<s|?V"n;
OUW._
@kW]
]_[y
/m:Rab
k*qo
J4&p
PD;r
k*7`

z3]_
F|:NI{
_h^&
nTe/
.+*4&
~./t
l(7m
|:=2
<'3M
{}\$E
+R/iY
VL&q
_bT|
n<E{
?)Y
-z-<
S(T
'F@2n
EJ*4\$>v
CFLW0
c 9R
'+' (
TJf\$
D|n"@
[<yv
HXh{
G>!M
f;'Z!ey
\$m3v
/SP}
lt<h
\m'&jd
X`cl
I5CF!(
vKLcE
o{F
M+4J
1LTFz
pT^n`
YbY`S
OS9m
I5i:
I5G6f
)%?
:*_&B
IWJ0
g h
mPXY

ZTW#h
<+3R
=/\o
S36.
p6Ha}
{-<j
w.DJ
&-;H
r6[g
[fg3
\b,^
O`f%
m'\$5
h:nR
f=bWI
%LNU
m^2`
:w\
wjK\$2;
?Q!6
<\$sHF^
_?\e
IV'43RC
c2'6
sZQ95yz3{U
v&Pb
^cAz
-<_mP
Q~Yy[
V~wq
{k=GyIs]
|ED/
|^xtU
"L,RQx
eLfu
}BAI
|ypw
,_%{
oFWG
M;)K
yrw5
9mlw
;{%c
}Uwu
qDtm<
HcMQ
Y}0%(
}F9t0MoQ

=Mf0
%jS]
M119]
Gb]mt
3P!Ss
yl?v
a1!
h]s%
eki{
YfBs
Y2,/kL
B_e#
^Ern
}N*Eg
pg{Xk
y\Qd
#LKi9
v7JY
%mCvm2%
W''}]
l2y&
L\g-
<gU%
l-M:R
tLmV[1x n
wl<`
MQ6r
TT_~
ehPy
0&'h
hlcx
1(Ve
A=2@
J%]vY
HUQ5
lm]bEW1t
#M]2~
RfA0
;r*&u
\$VQ;E
`KY4
V1q-
(HC)
n!*l+
C;qW
7^e8
g}>`
&5&f

P!hx8)4s
'u;+qP
p}A/
:\$4HXH;&d
]f(?
dH:P
BaeT
OG+b
>Ux>
/K?H
B(zIOF
ASi#
&KQF[3
~ 5T
rRt/
Z}0Oj
7o%_
'u9a
,w[_
N+R~
,K2X-
o:hG
IV!k
a}Jb
AV4ZO
}~r:
|# \
ySHc2H)
G+s7
8A 9
i4\$B(
W&~+
XO{[
fYaCe Z57
([|
-a@*9
\$gz0
Qbge
OqW^m
T(gS
'*)Rh
h}}aW
2(#It
6@}?P
Psk=
\;-s&
drC"
hMxt

d15c
R"\PM
)5R{
F`vf
[kp*B
U;tx1
F{hVV
3F"
{W4^Q
H2B7.L
_`=4
>%&{
9ia
KQBg
SeWY]
w("`Y2
Qu,T
m\$6H|
`8^Z\o
s`@
#/ X
X+8H;6
srsY
Sl%Q
"CQ+
 &f*
/{G0
\$B6T[r
8mJK
Bh/_
 .@]k
T4#R
ML,I
uz1gS
]-8k^~
@j,R
oSND~\
t?U\p
q:o0
k86<
pd2a0
kPJz
U]CdG
O:m}
]'7]~
].F\${
%-4h
~Kn/~

!N;Rg
Y]j!
c"YP1
,KO%s
ru})0z
axQe
\7/8
t^aaS
u}9"
=WX8
w<P*Z
lhw"
n+1E
U:y1
pz\$'svQ
*6pQ
y#jK
_S5
3\Lt
?e_L
:7Tp
:P%Q
P]Fu
/tZ'
r2rg
;s#8#
cW2)`Ag
\)[U
otf~
]Fzi
z=jt
%1?E)!o
kvA"
l<B.
QjFW
/Ld)
n6[J
\YQt
^UQ_0
43:g7}
9X{dX
i_rT
9\$RO
6oV<
!t5d
`r%0
*(Nqg]
`#!N

6%|^
2{lck
iP4,
!mgT7
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
*{$:{
taskdl.exe
taskse.exe
IN$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>
PPADDINGXXPPADDINGPPADDINGXXPPADDINGPPADDINGXXPPADDINGPPADDINGXXPPADDI
NGPPADDINGXXPPADDING

```

| 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) |

APPENDIX C – FILES FOUND FROM MALWARE ANALYSIS

List of files from malware analysis

- b.wnry
- c.wnry
- r.wnry
- s.wnry
- t.wnry
- taskdl.exe
- taskse.exe
- u.wnry
- m_bulgarian.wnry
- m_chinese (simplified).wnry
- m_chinese (traditional).wnry
- m_croatian.wnry
- m_czech.wnry
- m_danish.wnry
- m_dutch.wnry
- m_english.wnry
- m_filipino.wnry
- m_finnish.wnry
- m_french.wnry
- m_german.wnry
- m_greek.wnry

- m_indonesian.wnry
- m_italian.wnry
- m_japanese.wnry
- m_korean.wnry
- m_latvian.wnry
- m_norwegian.wnry
- m_polish.wnry
- m_portuguese.wnry
- m_romanian.wnry
- m_russian.wnry
- m_slovak.wnry
- m_spanish.wnry
- m_swedish.wnry
- m_turkish.wnry
- m_vietnamese.wnry
- @Please_Read_Me@.txt
- @WanaDecryptor@.bmp
- @WanaDecryptor@.exe
- 00000000.eky
- 00000000.pky
- 00000000.res
- DiskPart.exe
- TaskScheduler.exe
- f.wnry