



Malware Critical Analysis

Critical Analysis of ransomware to provide detection and mitigation techniques

Alex McNaughton

CMP320: Advanced Ethical Hacking

2022/23

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

Abstract

Ransomware in the past few years has become more and more of a potential threat to vulnerable machines and networks such as those used by medical professionals in hospitals or those used by government bodies in power plants. Due to this rise in popularity, greater focus should be put on analysis of ransomware payloads to better understand how they attack affected machines, and how they propagate through computer networks. To achieve a better understanding of ransomware and its functionality, a sample of ransomware will be identified and analyzed in order to gain a better understanding of these types of attacks.

The malware sample was put through a multitude of analysis tools within a closed virtual environment using both static and dynamic analysis methods which identified the type of malware the sample was, as well the functionality of the sample.

Analysis identified the sample as a branch of the ransomware WannaCry. Which encrypted files based on an internal list of common file types, demanding that a ransom be paid to one of three different bitcoin addresses in order to decrypt all important files. The malware made minimal attempts to hide itself or its processes, and provided several threat indicators that can be used to easily identify future attacks by this specific form of ransomware.

Contents

1	Introduction	1
1.1	Background	1
1.2	Aim	2
2	Procedure.....	3
2.1	Methodology.....	3
2.2	Identification.....	3
2.2.1	Static: Hashing.....	3
2.2.2	Dynamic: Sandboxing.....	4
2.3	Functionality	6
2.3.1	Static: String Searching	6
2.3.2	Static: Packer Detection	7
2.3.3	Static: Library and function Identification	8
2.3.4	Static: PE Analysis.....	8
2.3.5	Static: Decompiling	9
2.3.6	Dynamic: Registry Changes	11
2.3.7	Dynamic: Process Recording.....	11
2.3.8	Dynamic: Network Analysis.....	13
3	Discussion.....	14
3.1	General Discussion.....	14
3.1.1	Identification	14
3.1.2	Functionality	14
3.2	Countermeasures.....	15
3.3	Future Work	16
4	References	17
	Appendices.....	18
	Appendix A – List of File Extensions.....	18
	Appendix B – Contents of XIA.zip.....	19
	Directory Tree	19
	B.wnry 20	
	c.wnry 20	
	r.wnry 21	

s.wnry	22
t.wnry	22
u.wnry	23
taskdl.exe	24
taskche.exe	24
/msg	24
Appendix C – List of imported functions.....	25
KERNEL32.dll	25
USER32.dll	27
ADVAPI.dll	27
MSVCRT.dll	27

1 INTRODUCTION

1.1 BACKGROUND

In recent years, the prevalence of ransomware has drastically increased as a method of extorting victims of unsecure devices by preventing victims from accessing their files unless a sum of money is sent to the perpetrator. Preventing access to a users files is, in most cases, done by encrypting a victim's files with a strong enough encryption method that it would be impossible for them to gain access to their files unless they paid the ransom.

Its rise in popularity has been constant in the past few years, increasing by 63% in 2020, then another 105% in 2021 (Sonicwall, 2023). This rise in use can partly be attributed to the rise in value and use of popular cryptocurrencies like Bitcoin, as bitcoin's decentralized and anonymous nature makes it easier for hackers to collect a victims ransom without revealing their identity.

Ransomware can be incredibly dangerous when it comes into contact with the most critical of infrastructure, including schools, hospitals, water supply controls, power plant systems and more. In the United States alone, up 2,323 hospitals, schools, and government bodies were affected in some way by the use of ransomware (Homeland Security & Govermental Affairs Comittee, 2022). These systems are most at risk due to the lack of funding given to cybersecurity in these fields, as shown by an average of only 5% of a hospital's budget given to the security of healthcare systems in 2017 (Kruse, 2017). This lack in funding leaves a network open to more vulnerabilities, and therefore a higher chance of exploitation by cyber criminals.

Arguably the most popular piece of ransomware in the public consciousness is WannaCry, which , in its initial variant, affected over 300,000 machines over 150 countries in just under 8 hours (The White House, 2017) and could have continued in its rampage if not for an internal kill switch found by security professionals. This single attack costed the global market an unknowable amount, but it is estimated that it caused the main healthcare provider of the UK, the NHS, around 92 million pounds (National Health Executive, 2018) and lead to an increased risk of patient safety as healthcare workers were unable to access systems (S. Ghafur, 2019).

The rise and danger of encryption based ransomware to critical systems raises the need to provide some kind of preventative measures to systems that are potentially vulnerable to ransomware attacks. Due to the nature of cryptographic technology, well made ransomware will make it close to impossible to recover the files it has encrypted, so ransomware mitigation should be focused on pre-emptive measures that prevent the spread of the ransomware, as well as focusing on identifying the signatures of specific malware to make it easy to identify whether a machine is being infected with ransomware.

1.2 AIM

A sample of ransomware has been provided for analysis on a virtual machine. Through analysis of this sample, this paper aims to:

- Identify the given malware and the type of attack it is performing
- Understand how the given sample performs its attack
- Provide insight into identifying characteristics of the sample, with the intention of supplying threat indicators for easy identification of infection

2 PROCEDURE

2.1 METHODOLOGY

A strict analysis methodology will be needed in order to fully achieve the current aims:

- Identification of malware: this can be done through multiple different static and dynamic methods, however the easiest to start with would be hashing the file and searching for it in a virus database. Then, using a virtual environment to run the virus would likely provide some insight into what the virus is.
- Exploration of malware functionality: malware functionality is a broad term that can describe anything from how the actual code of the malware works to higher level functionality like registry changes or imported libraries. A wide range of analysis methods will be used in order to visualize all aspects of the malware.
- Discovery of threat indicators: finding indicators of malware running or existing on the machine will be found during analysis as more and more of the malware's behavior is understood. Once analysis is complete, the threat indicators will be collated and presented to make their status as a threat indicator clearer.

2.2 IDENTIFICATION

2.2.1 Static: Hashing

Using the hashing software HashMyFiles, the executables hashes could be enumerated. Once the hashes were obtained, it was possible to copy out the hash and paste it into the hash lookup service on the VirusTotal website, doing this provided some insight into the nature of the executable:

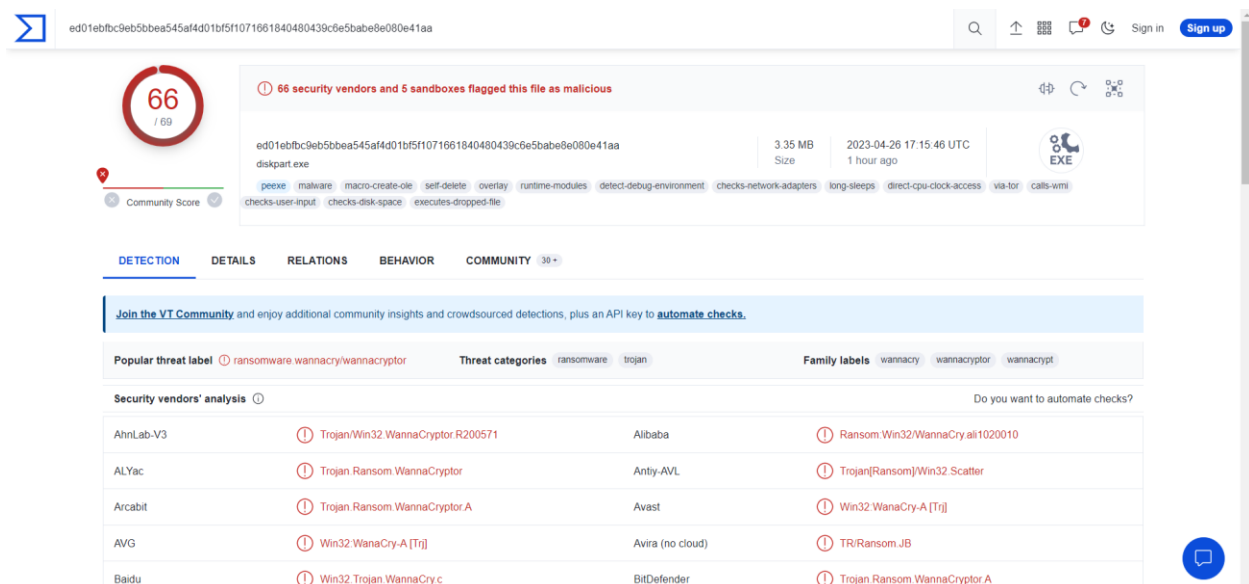


Fig 1: VirusTotal Results

2.2.2 Dynamic: Sandboxing

Simply running the malware in a closed off windows environment revealed some information regarding what the malware is and what it does. When the executable is run, several files are unzipped from the archive found earlier into the directory the executable is in, along with some other files that were previously not found

msg	4/27/2023 9:12 PM	File folder	
TaskData	4/27/2023 9:12 PM	File folder	
@Please_Read_Me@.txt	4/27/2023 9:12 PM	Text Document	1 KB
@WanaDecryptor@.exe	5/12/2017 2:22 AM	Application	240 KB
@WanaDecryptor@.exe	4/27/2023 9:12 PM	Shortcut	1 KB
00000000.eky	4/27/2023 9:12 PM	EKY File	0 KB
00000000.pky	4/27/2023 9:12 PM	PKY File	1 KB
00000000.res	4/27/2023 9:16 PM	RES File	1 KB
1.zip.WNCRY	3/3/2023 12:56 AM	WNCRY File	3,398 KB
89821682626334.bat.WNCRY	4/27/2023 9:12 PM	WNCRY File	1 KB
b.wnry	5/11/2017 8:13 PM	WNCRY File	1,407 KB
c.wnry	4/27/2023 9:12 PM	WNCRY File	1 KB
ed01ebfbc9eb5bbea545af4d01bf5f107166...	5/14/2017 3:29 PM	Application	3,432 KB
f.wnry	4/27/2023 9:12 PM	WNCRY File	1 KB
m.vbs.WNCRY	4/27/2023 9:12 PM	WNCRY File	1 KB
r.wnry	5/11/2017 3:59 PM	WNCRY File	1 KB
s.wnry	5/9/2017 4:58 PM	WNCRY File	2,968 KB
t.wnry	5/12/2017 2:22 AM	WNCRY File	65 KB
taskdl.exe	5/12/2017 2:22 AM	Application	20 KB
taskse.exe	5/12/2017 2:22 AM	Application	20 KB
u.wnry	5/12/2017 2:22 AM	WNCRY File	240 KB

Fig 2: Screenshot of sample directory after executing the sample

After this, the desktop background is changed.

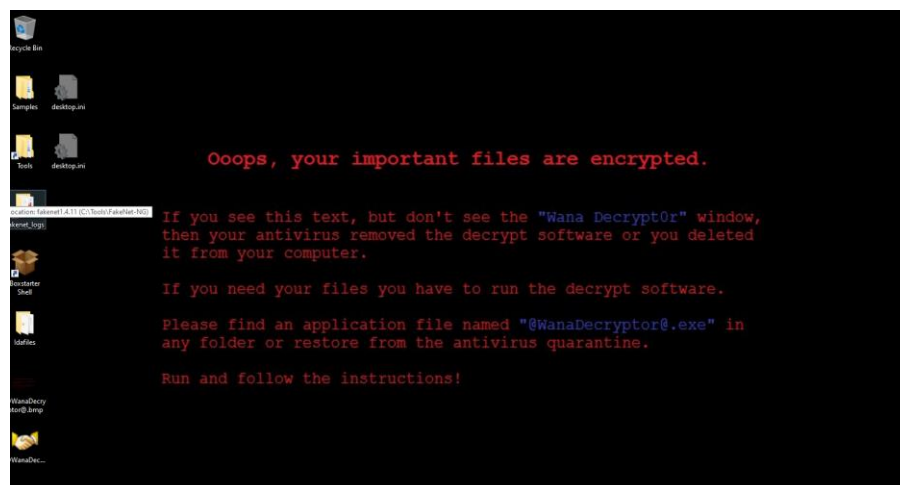


Fig 3: Screenshot of desktop wallpaper after malware execution

Then, a window appears with a ransom message:



Fig 4: Screenshot of main malware window

The claims of encryption are confirmed through viewing a txt file before and after the malware has run:



Fig 5: The contents of text.txt

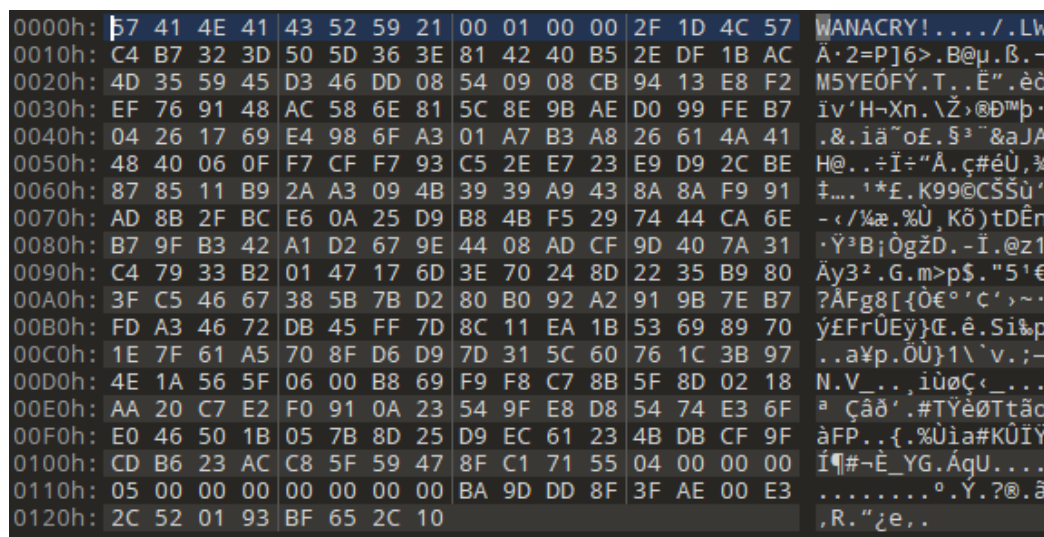


Fig 6: The content of hello.WNCRYPT after the malware has run

2.3 FUNCTIONALITY

2.3.1 Static: String Searching

Using a string searching script called strings64, it was possible to search through the executable for any plain text strings that might be hidden within the file, doing so produced an extremely long output, with a considerable amount of obfuscated data. However, there were still some sections of note that could attest to the evidence gathered during the hashing method.

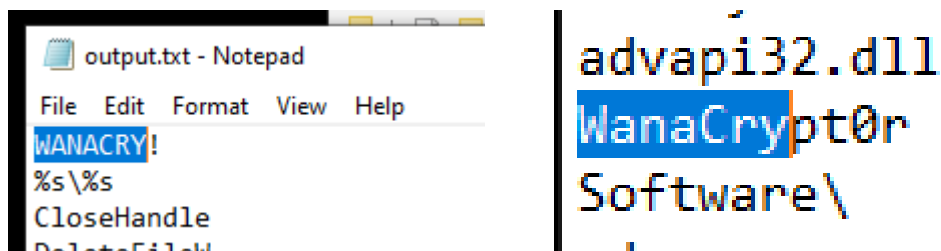


Fig 7: References to WannaCry in string search

The output of the string search also showed some insight into what windows functions were being imported for use with the executable.

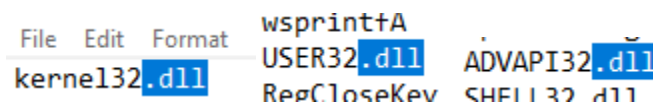


Fig 8: References to imported libraries in string search

However, inclusion in this file does not prove its use in the actual malware and will require further analysis to confirm its use.

Further searching after an initial search revealed more information about the malware, particularly its use of the Microsoft Enhanced RSA and AES Cryptographic Provider.

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

Fig 9: References to encryption provider in string search

Further searching also indicated that the software may have many different translations of itself stored in memory:

```
msg/m_bulgarian.wnry
msg/m_chinese (simplified).wnry
msg/m_chinese (traditional).wnry
msg/m_croatian.wnry
msg/m_czech.wnry
msg/m_danish.wnry
msg/m_dutch.wnry
msg/m_english.wnry
msg/m_filipino.wnry
msg/m_finnish.wnry
msg/m_french.wnry
msg/m_german.wnry
msg/m_greek.wnry
msg/m_indonesian.wnry
msg/m_italian.wnry
msg/m_japanese.wnry
msg/m_korean.wnry
msg/m_latvian.wnry
msg/m_norwegian.wnry
msg/m_polish.wnry
msg/m_portuguese.wnry
msg/m_romanian.wnry
msg/m_russian.wnry
msg/m_slovak.wnry
msg/m_spanish.wnry
msg/m_swedish.wnry
msg/m_turkish.wnry
msg/m_vietnamese.wnry
```

Fig 10: References to multiple languages in string search

A long list of file extensions was also found during string searching which can be found in appendix A

2.3.2 Static: Packer Detection

Using PEiD, it is possible to identify the packing software used for this particular piece of malware.

Results showed that the malware did not use any form of packing software, however it did reveal that the malware was made using Microsoft Visual C++ 6.0.

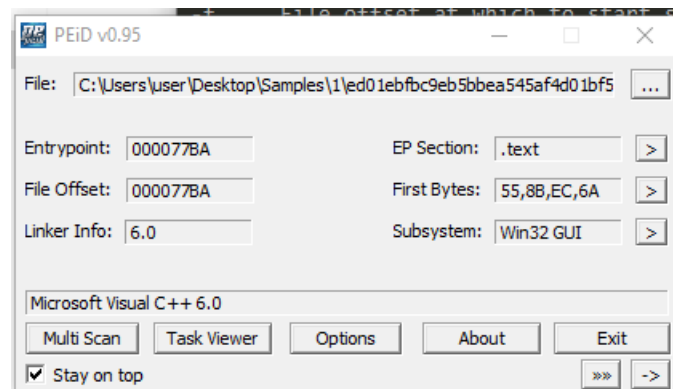


Fig 11: Screenshot of PEiD window after analysis of sample

2.3.3 Static: Library and function Identification

‘Dependency Walker’ is a piece of software that allows for the enumeration of dynamically linked libraries and the functions used from each library by the software when it is executed. Using Dependency Walker on the malware sample shows which libraries are being imported when the malware runs:

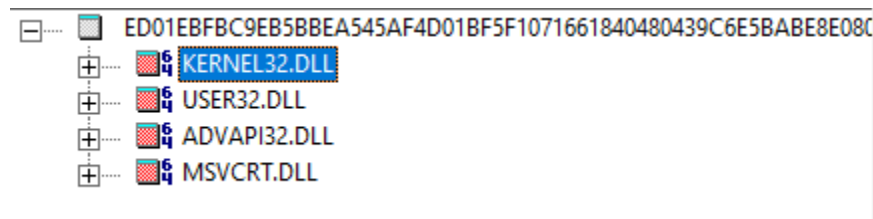


Fig 12: Screenshot of Dependency Walker with exported libraries.

Investigating the functions used by the software reveal that the sample might be doing some potentially dangerous things, like obtaining the startup info of the device the software is running from. Dependency Walker also highlights the use of cryptographic functions being used by the sample, presumably to encrypt files.

(A full list of function imports can be found in Appendix C)

2.3.4 Static: PE Analysis

Looking at the resources of the file with PEStudio revealed a large archive contained within the executable.

name	insta...	signature
version	1	version
manifest	1	manifest
XIA	2058	PKZIP

Fig 13: PEstudio screenshot of the resources found in the executable

This archive was extracted from the sample using PEStudios dump option, allowing for the files inside to be viewed with 7zip:

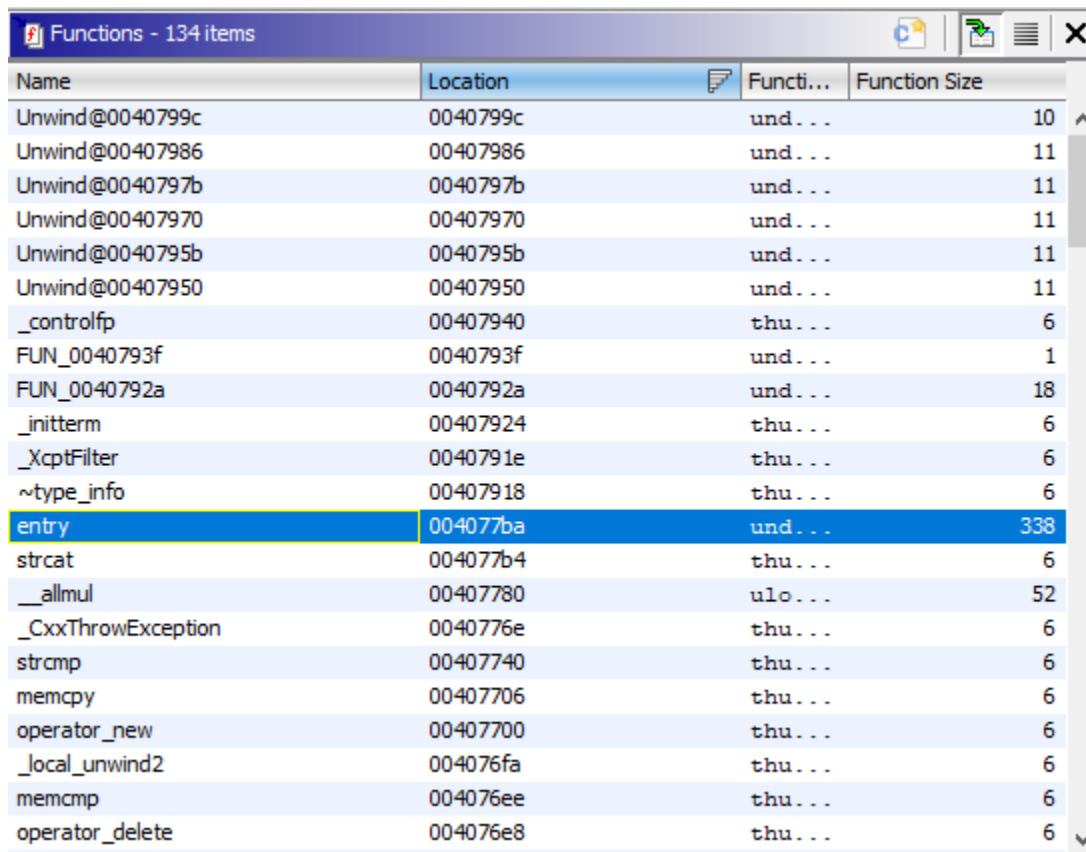
msg	1 329 657
b.wnry	1 440 054
c.wnry	780
r.wnry	864
s.wnry	3 038 286
t.wnry	65 816
taskdl.exe	20 480
taskse.exe	20 480
u.wnry	245 760

Fig 14: screenshot of initial state of internal zip file

The msg folder contains the list of different languages found during the string search method, and 2 new executables can be seen within the executable. However, attempting to extract any of these files requires a password.

2.3.5 Static: Decompiling

After performing all of the previous methods, ghidra was finally used to analyze the code to identify any behaviors or values that were not seen during previous analysis. The executable was imported into ghidra, and the first step was to find the main function, since there was no function explicitly called main, analysis started at the first function called, named 'entry'.



Name	Location	Funci...	Function Size
Unwind@0040799c	0040799c	und...	10
Unwind@00407986	00407986	und...	11
Unwind@0040797b	0040797b	und...	11
Unwind@00407970	00407970	und...	11
Unwind@0040795b	0040795b	und...	11
Unwind@00407950	00407950	und...	11
_controlfp	00407940	thu...	6
FUN_0040793f	0040793f	und...	1
FUN_0040792a	0040792a	und...	18
_initterm	00407924	thu...	6
_XcptFilter	0040791e	thu...	6
~type_info	00407918	thu...	6
entry	004077ba	und...	338
strcat	004077b4	thu...	6
__allmul	00407780	ulo...	52
_CxxThrowException	0040776e	thu...	6
strcmp	00407740	thu...	6
memcpy	00407706	thu...	6
operator_new	00407700	thu...	6
_local_unwind2	004076fa	thu...	6
memcmp	004076ee	thu...	6
operator_delete	004076e8	thu...	6

Fig 15: List of functions ordered by location in the file

Looking through the entry function, the code in this function is very similar to the default code generated for windows executables, with the real main function found at the bottom of the function.

```
local_6c = WinMain(hInstance, hPrevInstance, pCmdLine, nCmdShow);
```

Fig 16: Location of the real main function

Accessing this function show us more detail about how the code actually executes. Some areas of note include:

2.3.5.1 Password for internal resources

In the main section of the sample code, a function is used to load a given internal resource

```
004020c8 c7 04 24      MOV      dword ptr [ESP]=>local_6f8,s_WNcry@2017_0040f52c = "WNcry@2017"
                2c f5 40 00
004020cf 53           PUSH      EBX
004020d0 e8 d6 fc      CALL      FUN_00401dab      undefined4 FUN_00401dab
```

Fig 17: screenshot of the program calling the resource loading function

Looking at the function more closely, the first line of this function tries to find a resource with the XIA filetype with an id of 2058

```
hResInfo = FindResourceA(param_1, (LPCSTR) 2058, &XIA);
```

Fig 18: screenshot of the line of code that finds a given resource

From PE analysis, the resource with the id 2058 and filetype XIA is the internal archive of resources that required a password to access. Knowing this, it was possible that the value of the variable being pushed to the stack before the function ('WNcry@2017') could be the password for the archive. After testing this, it was confirmed that this was the password for the archive (a full description of the archives contents can be found in appendix B).

2.3.5.2 Bitcoin address selector

As soon as the archive is exported, the next thing that the program does is select a bitcoin address from memory and write it to the file c.wnry. the decompiled code to do this has been annotated to be easier to read:

```
void select_bitcoin_address(void)
{
    bool c_buffer;
    undefined3 extraout_var;
    int randomnum;
    char unknown_array [780];
    char *bitcoin_address [3];

    bitcoin_address[0] = s_13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb_0040f488;
    bitcoin_address[1] = s_12t9YDPgwueZ9NyMgw519p7AA8isjr6S_0040f464;
    bitcoin_address[2] = s_115p7UMMngoJlpMvKpHijcRdfJNXj6Lr_0040f440;
    /* read c.wnry */
    c_buffer = read_or_write_c.wnry(unknown_array,1);
    /* if c.wnry exists */
    if (CONCAT31(extraout_var,c_buffer) != 0) {
        randomnum = rand();
        /* set unknown array to a randomly selected option from bitcoin_addresses */
        strcpy(unknown_array + 0xb2,bitcoin_address[randomnum % 3]);
        /* write new value to c.wnry */
        read_or_write_c.wnry(unknown_array,0);
    }
    return;
}
```

Fig 19 :The decompiled bitcoin selector function, with renamed variables and comments

The result of this code can be seen once the malware is run, and displays its ransom window:

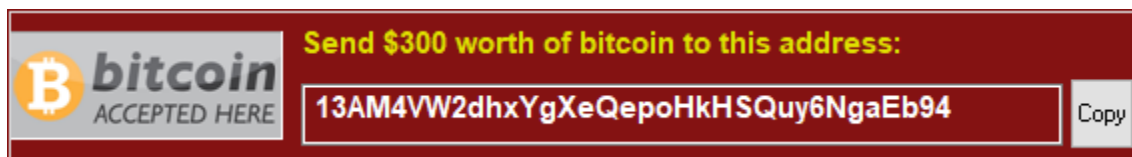


Fig 20: Screenshot of malware window showing the first address in the array was selected

2.3.6 Dynamic: Registry Changes

Using RegShot, The changes made to the machines registry can be captured and used to identify if the malware made changes to the registry. Once the malware is run and the changes are captured, an output file reveals some of the changes made:

```
HKU\S-1-5-21-2169232433-3398496680-935370409-1000\Software\WanaCrypt0r
HKU\S-1-5-21-2169232433-3398496680-935370409-1000\Control Panel\Desktop\WallPaper: "C:\Users\user\Desktop\WanaDecryptor@.bmp"
```

Fig 21: Evidence of registry modification.

2.3.7 Dynamic: Process Recording

From the sandboxing test, it is clear that the sample is making a lot of changes to the machine that are not necessarily visible to the user. To view these hidden processes, a tool called Process Monitor was used to capture the processes that run as the device is being infected with the malware. Doing this type of analysis gives a reasonably high insight into the timeline of events that the sample performs as it runs:

Firstly, the malware performed some basic setup , like calling the 32bit software emulator for windows , and obtaining the current system language.

```
21:00:48... ed01ebfbc9eb5... 3488 Load Image C:\Windows\SysWOW64\ntdll.dll
21:00:48... ed01ebfbc9eb5... 3488 RegOpenKey HKLM\SYSTEM\CurrentControlSet\Control\Nls\CustomLocale
```

Fig 22: Sample of startup procedure

Then, it began to write the files in its internal zip file into the directory of the sample:

```
21:00:48... ed01ebfbc9eb5... 3488 WriteFile C:\Users\user\Desktop\b.wnry
21:00:48... ed01ebfbc9eb5... 3488 WriteFile C:\Users\user\Desktop\c.wnry
21:00:48... ed01ebfbc9eb5... 3488 CreateFile C:\Users\user\Desktop\msg\m_finnish.wnry
21:00:48... ed01ebfbc9eb5... 3488 WriteFile C:\Users\user\Desktop\msg\m_finnish.wnry
```

Fig 23: Sample of resource unpacking

Once this is done, it then imported the main decryptor window from its binary.

```
21:00:49... ed01ebfbc9eb5... 3488 WriteFile C:\Users\user\Desktop\@WanaDecryptor@.exe
```

Fig 24: main file unpacking

After some more setup, the program began querying every file on the system, presumably to check if it has one of the file types in the list of files types in its code. Once it finds a file with a file type that matches the internal file type list, it replaced the file with an encrypted version of the file with the file type WNCRYT.

```
21:00:53... ed01ebfbc9eb5... 3488 CreateFile C:\Python39\Lib\site-packages\prompt_toolkit\formatted_text\~SDAE9F.tmp
21:00:53... ed01ebfbc9eb5... 3488 CloseFile C:\Python39\Lib\site-packages\prompt_toolkit\formatted_text\~SDAE9F.tmp
21:00:53... ed01ebfbc9eb5... 3488 CreateFile C:\Python39\Lib\site-packages\prompt_toolkit\formatted_text\~SDAE9F.tmp
21:00:53... ed01ebfbc9eb5... 3488 CloseFile C:\Python39\Lib\site-packages\prompt_toolkit\formatted_text\~SDAE9F.tmp
21:00:53... ed01ebfbc9eb5... 3488 CreateFile C:\Python39\Lib\site-packages\prompt_toolkit\formatted_text\~SDAE9F.tmp
21:00:53... ed01ebfbc9eb5... 3488 QueryAttributeTagFile C:\Python39\Lib\site-packages\prompt_toolkit\formatted_text\~SDAE9F.tmp
21:00:53... ed01ebfbc9eb5... 3488 SetDispositionInformati... C:\Python39\Lib\site-packages\prompt_toolkit\formatted_text\~SDAE9F.tmp
21:00:53... ed01ebfbc9eb5... 3488 CloseFile C:\Python39\Lib\site-packages\prompt_toolkit\formatted_text\~SDAE9F.tmp
```

Fig 25: the program queries the file type of the file ~SDAE9F.tmp, since .tmp isn't part of the internal file type list, it closes the file and moves on to the next file in the directory.

```
21:00:... ed01ebfbc9e... 3488 QueryStandardInfor... C:\Tools\Cmder\vendor\git-for-windows\mingw64\lib\tk8.6\msgs\cs.msg
21:00:55... ed01ebfbc9eb5... 3488 QueryBasicInformation... C:\Tools\Cmder\vendor\git-for-windows\mingw64\lib\tk8.6\msgs\cs.msg
21:00:55... ed01ebfbc9eb5... 3488 ReadFile C:\Tools\Cmder\vendor\git-for-windows\mingw64\lib\tk8.6\msgs\cs.msg
21:00:... ed01ebfbc9e... 3488 ReadFile C:\Tools\Cmder\vendor\git-for-windows\mingw64\lib\tk8.6\msgs\cs.msg
21:00:55... ed01ebfbc9eb5... 3488 CreateFile C:\Tools\Cmder\vendor\git-for-windows\mingw64\lib\tk8.6\msgs\cs.msg.WNCRYT
21:00:... ed01ebfbc9e... 3488 WriteFile C:\Tools\Cmder\vendor\git-for-windows\mingw64\lib\tk8.6\msgs\cs.msg.WNCRYT
21:00:... ed01ebfbc9e... 3488 WriteFile C:\Tools\Cmder\vendor\git-for-windows\mingw64\lib\tk8.6\msgs\cs.msg.WNCRYT
21:00:... ed01ebfbc9e... 3488 WriteFile C:\Tools\Cmder\vendor\git-for-windows\mingw64\lib\tk8.6\msgs\cs.msg.WNCRYT
21:00:... ed01ebfbc9e... 3488 WriteFile C:\Tools\Cmder\vendor\git-for-windows\mingw64\lib\tk8.6\msgs\cs.msg.WNCRYT
21:00:... ed01ebfbc9e... 3488 WriteFile C:\Tools\Cmder\vendor\git-for-windows\mingw64\lib\tk8.6\msgs\cs.msg.WNCRYT
```

Fig 26: example of a match to the file type list. Since .msg is a file type in the file type list, the program reads the information from the file, encrypts it, and writes the result to a new file

Once the program had finished encrypting every file that matched its extension list, it changed the wallpaper of the desktop, and runs the main decryptor window.

```
21:01:... ed01ebfbc9e... 3488 RegSetValue HKCU\Control Panel\Desktop\Wallpaper
```

Fig 27: evidence of wallpaper tampering

```
21:01:... ed01ebfbc9e... 3488 Process Create C:\Users\user\Desktop\@WanaDecryptor@.exe
```

Fig 28: Creation of the main malware window

2.3.8 Dynamic: Network Analysis

Partial network analysis could be performed on the sample using FakeNet. Once fakenet was running and the malware executed, the running malware process made several requests to a tor proxy.

```
05/01/23 07:15:02 PM [ Diverter] @WanaDecryptor@.exe (3016) requested TCP 127.0.0.1:9050
```

Fig 29: Evidence of main malware request to a set up tor connection

A new process named taskhsvc.exe also appeared to make several suspicious requests through the loopback adapter.

```
05/01/23 07:16:02 PM [ Diverter] taskhsvc.exe (2148) requested TCP 127.0.0.1:49845
```

Fig 30: evidence of suspicious connections

Looking at the log files of FakeNet using WireShark reveals that the sample also made several NetBIOS requests.

8	3.189220	169.254.31.215	169.254.255.255	NBNS	92 Name query NB DESKTOP-14QC1L8<1c>
9	3.189725	169.254.31.215	169.254.255.255	NBNS	92 Name query NB DESKTOP-14QC1L8<1c>
11	3.950394	169.254.31.215	169.254.255.255	NBNS	92 Name query NB DESKTOP-14QC1L8<1c>
12	3.950889	169.254.31.215	169.254.255.255	NBNS	92 Name query NB DESKTOP-14QC1L8<1c>
17	4.685410	169.254.31.215	169.254.255.255	NBNS	92 Name query NB DESKTOP-14QC1L8<1c>
18	4.685820	169.254.31.215	169.254.255.255	NBNS	92 Name query NB DESKTOP-14QC1L8<1c>

Fig 31: machine making several NetBIOS requests after the malware is run

3 DISCUSSION

3.1 GENERAL DISCUSSION

3.1.1 Identification

From the information gathered through analysis of the sample, it is clear that the malware being scanned is the encryption software associated with the ransomware virus WannaCry, given the hashing report, the multiple references to WannaCry in the string search, and the behavior of the sample as it is run in a virtual environment.

However, the sample varies in some cases to analysis performed by third parties. Many external samples make references to an action performed by WannaCry, which checks for if a given hardcoded URL is available and does not perform any malicious actions if a connection to this URL is possible (Waleed Alraddadi, 2018). This 'kill switch' did not appear in the given sample during disassembly, and appears to be removed entirely so that the malware will run no matter what. This indicates that the sample provided may have been a later copy of the original malware, with some parts removed to regain functionality and allow it to keep encrypting files.

3.1.2 Functionality

Sandboxing revealed the main purpose of the sample is to encrypt the important files of the computer and demand the user pay a ransom to decrypt these files. Files on the system are chosen for encryption based on an internal list of common file extensions.

While how the given sample propagates through the network is unknown, given its identification as a branch of WannaCry it is likely that the networking functionality makes use of an exploit in the SMB protocol that allows for remote code execution on affected computers (ESET, 2017). However, this is purely speculation and would require further analysis of the networking functionality to confirm. What is known about the network functionality is that the sample attempts to discover other devices on the network through the NetBIOS protocol, presumably to perform this exploit and inject itself into other devices.

The virus authors have made several attempts to anonymize their connections to their victims: Firstly, the payment for decryption must be paid through one of three different bitcoin addresses as chosen by the software on startup, since payments and bitcoin wallet owners can be made completely anonymous, there would be no way to track down the creator of the address through the address name alone. Secondly, the victim can contact the malware owners organization through a message system in the main malware window, this connection is made through a tor connection to one of five different onion links, making it difficult to track down the exact location of the server used for connecting the malware owners and their victims.

Through further analysis of the malware binary, it is reasonable to identify the malware to have been written in C++, using several imported windows libraries such as kernel32.dll for file management, and user32.dll for wallpaper changing. Files are encrypted using the Microsoft Enhanced RSA and AES Cryptographic Provider, a cryptography service provided by Microsoft to encrypt data into multiple encryption formats.

Static analysis of the file revealed that there was no packing of any kind done to prevent outside analysis, nor any attempt made to obfuscate code. The only attempt made to hide an aspect of the software was using a password on the internal XIA zip file, however the password was left hard coded into the code and was easily found through disassembly of the sample.

3.2 COUNTERMEASURES

The main functionality of the sample is the encryption of the majority of the files on the machine, meaning that once these files are encrypted there is no way to decrypt them. The simplest countermeasure against ransomware of this nature is to save backups of crucial files off network so that they can be preserved even if a ransomware attack affects a machine.

Identification of threat indicators unique to the sample would also help as a countermeasure in order to quickly identify if a machine is potentially vulnerable or already infected, and prevent the infected machine from coming into contact with other machines through the use of antivirus software or firewall rules.

There are several indicators of infection with this specific sample:

File/behavior	Justification as threat indicator
Executable file with a SHA256 hash of the file matching the malware sample.	Identical hashes indicate that the file is the malware payload, and could infect the machine if run.
Executable file with the strings 'WANACRY!', or 'WanaCrypt0r'	These specific strings would rarely be included in other safe files, and could indicate an executable is malware.
Any files with the extension .wnry	The .wnry extension is used extensively as an obfuscation method to stop users from accessing files critical to the functionality of the malware sample. Its existence on a machine could indicate that the malware has already run on the machine and has extracted its files to run.
Any files with the extension .WNCRY	The sample uses this extension to hold the data of files it has encrypted. If a large amount of files have this extension, it is likely that the machine has already been infected with the malware, and should be disconnected from any networks to prevent potential infection.
@WanaDecryptor@.exe	The executable that holds the ransom message, its existence on a machine indicates it has already been infected, and the malware has moved to asking for ransom.
@Please_Read_Me@.txt	Ransom message that appears in directories that have encrypted files in them.
Wallpaper is set to b.wnry	The malware changes the wallpaper to a bitmap image called b.wnry once it has finished encryption.

3.3 FUTURE WORK

Further analysis of the network side of the sample would have lead to a greater understanding of the virality of the sample, as it is clear that the network functionality found in the sample would have lead to further exploitation of a network, if one had been provided. For future work, a virtual test network could be used in order to observe how the malware travels through a network, while protecting a physical network from infection.

Further analysis of the decompiled code may have also provided some more insight into the behavior of the malware, as there are still aspects of the malware that are still not fully explained by this paper.

4 REFERENCES

ESET, 2017. [CA6443] *Vulnerability CVE-2017-0144 in SMB exploited by WannaCryptor ransomware to spread over LAN*. [Online]

Available at: https://support.eset.com/en/ca6443-vulnerability-cve-2017-0144-in-smb-exploited-by-wannacryptor-ransomware-to-spread-over-lan?locale=en_US&viewlocale=en_US

[Accessed 1 May 2023].

Homeland Security & Governmental Affairs Committee, 2022. *Rise of Cryptocurrency in Ransomware Attacks, Available Data, and National Security Concerns*. [Online]

Available at: https://www.hsgac.senate.gov/wp-content/uploads/imo/media/doc/HSGAC%20Majority%20Cryptocurrency%20Ransomware%20Report_Executive%20Summary.pdf

[Accessed 25 April 2023].

Kruse, C. S. S. B. V. H. & N. A., 2017. Security Techniques for the Electronic Health Records.. *Journal of medical systems*, 41(8), p. 127.

National Health Executive, 2018. *WannaCry cyber-attack cost the NHS £92m after 19,000 appointments were cancelled*. [Online]

Available at: <https://www.nationalhealthexecutive.com/articles/wannacry-cyber-attack-cost-nhs-ps92m-after-19000-appointments-were-cancelled>

[Accessed 25 April 2023].

S. Ghafur, S. K. K. H. G. M. A. D. & P. A., 2019. A retrospective impact analysis of the WannaCry cyberattack on the NHS. *npj Digital Medicine*, Volume 2, p. 2.

Sonicwall, 2023. *2023 Cyber Threat Report*. [Online]

Available at: <https://www.sonicwall.com/medialibrary/en/white-paper/2023-cyber-threat-report.pdf>

[Accessed 25 April 2023].

The White House, 2017. *Briefing on the Attribution of the WannaCry Malware Attack to North Korea*. [Online]

Available at: <https://www.youtube.com/watch?v=pfqing4mbVI&t>

[Accessed 25 April 2023].

Waleed Alraddadi, H. S., 2018. *A Comprehensive Analysis of WannaCry: Tecnical Analysis,Reverse Englineering, and Motivation*. [Online]

Available at: https://people-ece.vse.gmu.edu/coursewebpages/ECE/ECE646/F19/project/F18_presentations/Session_III/Session_III_Report_3.pdf

[Accessed 29 April 2023].

APPENDICES

APPENDIX A – LIST OF FILE EXTENSIONS

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

APPENDIX B – CONTENTS OF XIA.ZIP

Directory Tree

Folder PATH listing

Volume serial number is F87E-C97A

C:\USERS\USER\DESKTOP\SAMPLES\1\XIA

| b.wnry

| c.wnry

| r.wnry

| s.wnry

| t.wnry

| taskdl.exe

| taskse.exe

| u.wnry

|

\---msg

 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

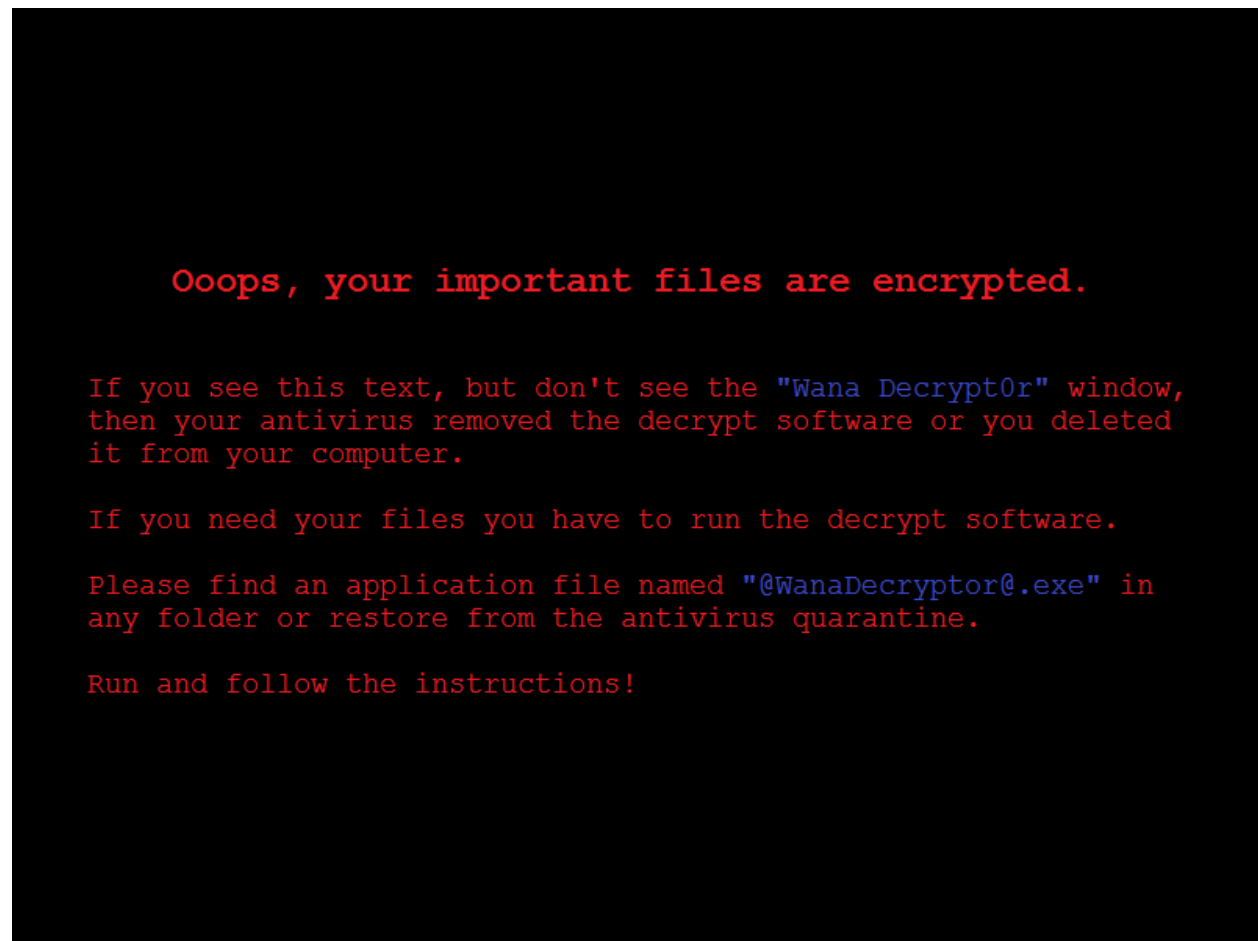
 m_latvian.wnry

 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
```

B.wnry

A bitmap file which is used as the wallpaper of the machine once it is infected



c.wnry

a text file containing some tor links, and a link to the tor browser zip file


```

00E0h: 00 00 00 00 67 78 37 65 6B 62 65 6E 76 32 72 69 ....gx7ekbenv2ri
00F0h: 75 63 6D 66 2E 6F 6E 69 6F 6E 3B 35 37 67 37 73 ucmf.onion;57g7s
0100h: 70 67 72 7A 6C 6F 6A 69 6E 61 73 2E 6F 6E 69 6F pgrzlojinas.onio
0110h: 6E 3B 78 78 6C 76 62 72 6C 6F 78 76 72 69 79 32 n;xxlvbrloxvriy2
0120h: 63 35 2E 6F 6E 69 6F 6E 3B 37 36 6A 64 64 32 69 c5.onion;76jdd2i
0130h: 72 32 65 6D 62 79 76 34 37 2E 6F 6E 69 6F 6E 3B r2embyv47.onion;
0140h: 63 77 77 6E 68 77 68 6C 7A 35 32 6D 61 71 6D 37 cwwnhwhlz52maqm7
0150h: 2E 6F 6E 69 6F 6E 3B 00 00 00 00 00 00 00 00 .onion;.....

```

gx7ekbenv2riucmf.onion

57g7spgrzlojinas.onion

xxlvbrloxvriy2c5.onion

76jdd2ir2embyv47.onion

cwwnhwhlz52maqm7.onion

```

01D0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 68 74 .....ht
01E0h: 74 70 73 3A 2F 2F 64 69 73 74 2E 74 6F 72 70 72 tps://dist.torpr
01F0h: 6F 6A 65 63 74 2E 6F 72 67 2F 74 6F 72 62 72 6F oject.org/torbro
0200h: 77 73 65 72 2F 36 2E 35 2E 31 2F 74 6F 72 2D 77 wser/6.5.1/tor-w
0210h: 69 6E 33 32 2D 30 2E 32 2E 39 2E 31 30 2E 7A 69 in32-0.2.9.10.zi
0220h: 70 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 p.....
0230h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

r.wnry

plaintext file containing the text that appears in the read me file that generates when the executable is run.

Q: What's wrong with my files?

A: Ooops, your important files are encrypted. It means you will not be able to access them anymore until they are decrypted.

If you follow our instructions, we guarantee that you can decrypt all your files quickly and safely!

Let's start decrypting!

Q: What do I do?

A: First, you need to pay service fees for the decryption.

Please send %s to this bitcoin address: %s

Next, please find an application file named "%s". It is the decrypt software.

Run and follow the instructions! (You may need to disable your antivirus for a while.)

Q: How can I trust?

A: Don't worry about decryption.

We will decrypt your files surely because nobody will trust us if we cheat users.

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

s.wnry

zip archive of Tor 0.2.9.0, with all associated libraries required

Folder PATH listing

Volume serial number is F87E-C97A

C:\USERS\USER\DESKTOP\SAMPLES\1\XIA\S

+---Data

| \---Tor

\---Tor

libeay32.dll

libevent-2-0-5.dll

libevent_core-2-0-5.dll

libevent_extra-2-0-5.dll

libgcc_s_sjlj-1.dll

libssp-0.dll

ssleay32.dll

tor.exe

zlib1.dll

t.wnry

currently unknown file as the file is encrypted.

u.wnry

The decryptor executable that holds the ransom message once files have been decrypted.



Above: running the window on its own doesn't start the timer or import the bitcoin wallet address, as these are done when the main sample is run, however it does change the wallpaper on startup.

The exe also contains modified properties to masquerade as a windows server tool:

Property	Value
Description	
File description	Load PerfMon Counters
Type	Application
File version	6.1.7600.16385
Product name	Microsoft® Windows® Operating System
Product version	6.1.7600.16385
Copyright	© Microsoft Corporation. All rights reserv ...
Size	240 KB
Date modified	5/11/2017 6:22 PM
Language	English (United States)
Original filename	LODCTR.EXE

taskdl.exe

executable that searches for files with the .wncrypt extension and deletes them. Files with this extension are created as temporary files when the sample is executed.

taskche.exe

will run a given program in multiple user sessions, used to spread the malware across multiple users on the same device

/msg

Directory containing multiple translations of the text that appears in the decryptor executable. Saved as rich text format files.

m_english.wnry

What Happened to My Computer?

Your important files are encrypted.

Many of your documents, photos, videos, databases and other files are no longer accessible because they have been encrypted. Maybe you are busy looking for a way to recover your files, but do not waste your time. Nobody can recover your files without our decryption service.

Can I Recover My Files?

Sure. We guarantee that you can recover all your files safely and easily. But you have not so enough time.

You can decrypt some of your files for free. Try now by clicking <Decrypt>.

But if you want to decrypt all your files, you need to pay.

You only have 3 days to submit the payment. After that the price will be doubled.

Also, if you don't pay in 7 days, you won't be able to recover your files forever.

We will have free events for users who are so poor that they couldn't pay in 6 months.

How Do I Pay?

Payment is accepted in Bitcoin only. For more information, click <About bitcoin>.

Please check the current price of Bitcoin and buy some bitcoins. For more information, click <How to buy bitcoins>.

And send the correct amount to the address specified in this window.

After your payment, click <Check Payment>. Best time to check: 9:00am - 11:00am GMT from Monday to Friday.

Once the payment is checked, you can start decrypting your files immediately.

Contact

If you need our assistance, send a message by clicking <Contact Us>.

We strongly recommend you to not remove this software, and disable your anti-virus for a while, until you pay and the payment gets processed. If your anti-virus gets updated and removes this software automatically, it will not be able to recover your files even if you pay!

APPENDIX C – LIST OF IMPORTED FUNCTIONS

KERNEL32.dll

CloseHandle

CopyFileA

CreateDirectoryA

CreateDirectoryW

CreateFileA

CreateProcessA

DeleteCriticalSection

EnterCriticalSection

FindResourceA

FreeLibrary

GetComputerNameW

GetCurrentDirectoryA

GetExitCodeProcess

GetFileAttributesA

GetFileAttributesW

GetFileSize

GetFileSizeEx

GetFullPathNameA

GetModuleFileNameA

GetModuleHandleA

GetProcAddress
GetProcessHeap
GetStartupInfoA
GetTempPathW
GetWindowsDirectoryW
GlobalAlloc
GlobalFree
HeapAlloc
HeapFree
InitializeCriticalSection
IsBadReadPtr
LeaveCriticalSection
LoadLibraryA
LoadResource
LocalFileTimeToFileTime
LockResource
MultiByteToWideChar
OpenMutexA
ReadFile
SetCurrentDirectoryA
SetCurrentDirectoryW
SetFileAttributesW
SetFilePointer
SetFileTime
SetLastError
SizeofResource
Sleep
SystemTimeToFileTime
TerminateProcess
VirtualAlloc

VirtualFree
VirtualProtect
WaitForSingleObject
WriteFile

USER32.dll

wsprintfA

ADVAPI.dll

CloseServiceHandle
CreateServiceA
CryptReleaseContext
OpenSCManagerA
OpenServiceA
RegCloseKey
RegCreateKeyW
RegQueryValueExA
RegSetValueExA
StartServiceA

MSVCRT.dll

calloc
exit
fclose
fopen
fread
free
fwrite
malloc
memcmp
memcpy
memset
rand

realloc
sprintf
srand
strcat
strcmp
strcpy
strlen
strchr
swprintf
wscat
wcslen
wcsrchr