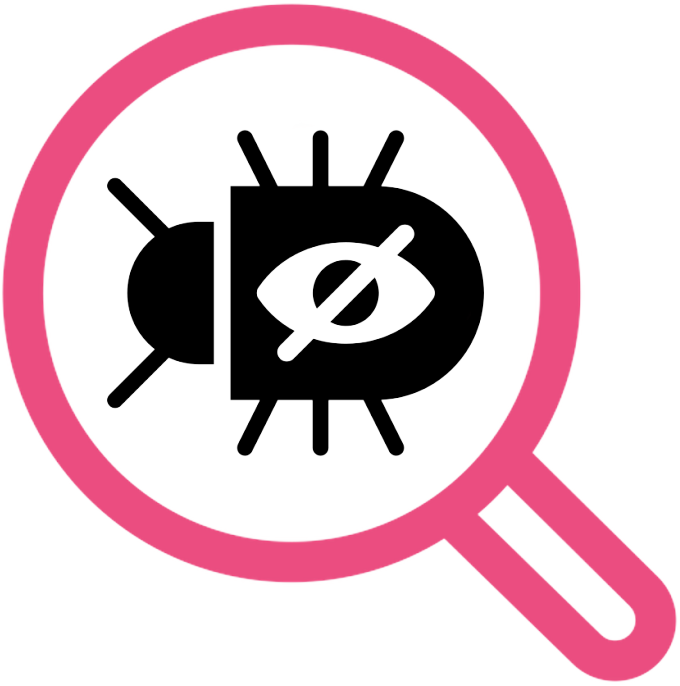

Malware Analysis Report

Wannacry Ransomware

Thomas MacKinnon
February 2024
Version 1.0

Contents

List of Figures

1 Executive Summary

File name	sha256sum
wannacry.exe	24d004a104d4d54034dbcffc2a4b19a11f39008a575aa614ea04703480b1022c

Wannacry.exe is a devastating piece of Ransomware, with Worm like capabilities through network traversal features to infect other hosts through SMB ports, targetting x32 Windows systems. The binary encrypts all user files with the `.WNCRY` extension and prompts the victim to pay the ransom through the "Wana Decrypt0r" application which is found with other related files on the Desktop. The "Wana decrypt0r" application runs continuously, gaining persistence, and the files cannot be recovered unless the ransom is paid, with a deadline before the files are lost forever.

Symptoms of infection are obvious intentionally. Yara rules have been written and can be found in Appendix A. Additionally, a kill switch for the binary was discovered, preventing the payload from detonating.

2 High-Level Technical Summary

Wannacry consists of one large payload after anti-analysis checks, which results in encryption of user files, and the unpacking of associated software from a staging folder named located at `C:\ProgramData\kgxhvkydfwf152`. After encryption completes, several files and the "Wana Decrypt0r" application are copied to the Desktop, which runs continuously until the victim pays. Figure 1 shows the flow of Wannacry.

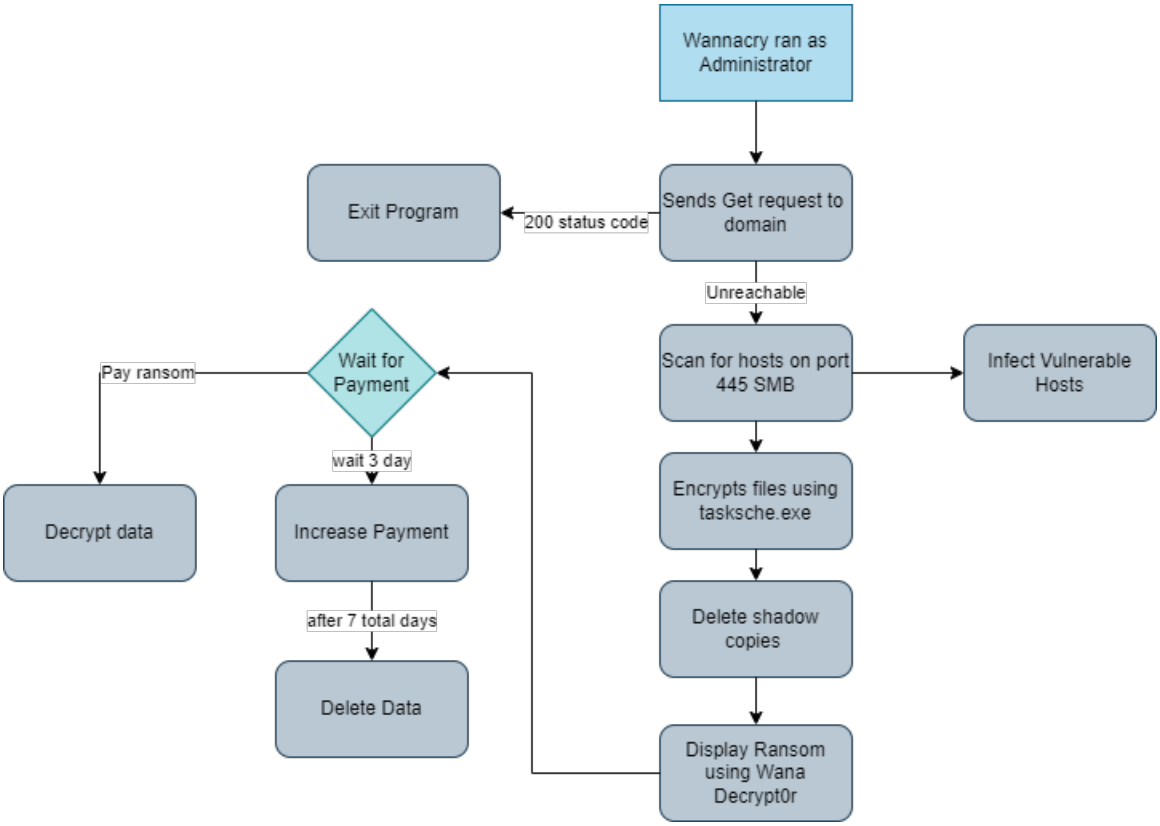

Figure 1: Flowchart of Wannacry

3 Malware Composition

File name	sha256sum	VirusTotal Result
wannacry.exe	24d004a104d4d54034dbcffc2a4b19a11f39008a575aa614ea04703480b1022c	69/72
@WanaDecrypt0r@.exe	b9c5d4339809e0ad9a00d4d3dd26fdf44a32819a54abf846bb9b560d81391c25	
tasksche.exe		

Table 1: Sha256 and VirusTotal results for Malware components

3.1 Wannacry.exe

Malware written in C++ for x32 Windows systems, that contains mainly suspicious strings, imports, and libraries. Encrypts all user files and creates the tasksche.exe process after anti-analysis.

3.2 @WanaDecrypt0r@.exe

The Decryptor tool, which appears in every directory of the victim's machine, constantly reopens and presses payment with a deadline before file deletion. The tool is also responsible for encrypting files, including any new files made post-detonation.

3.3 tasksche.exe

A process that creates a staging folder in `C:\ProgramData\kgxhvkydfwf152`, that creates many of the tools and files used in the later stages of the payload. This also sets the language based on the victim's IP.

4 Basic Static Analysis

The binaries sha256 sum was retrieved and submitted to VirusTotal, which stated it was a very dangerous piece of ransomware named Wannacry.

Next, the strings of Wannacry were extracted using Floss and sent to a text file. There were a lot of suspicious calls to command prompt, changing the security settings, and notably a full URL "hxxtp://www.iugerfsodp9ifjaposdfjhgosurijfaewrwergwea.com" to a very strange domain, as seen in Figure 3

```
__TREEPATH_REPLACE__
\\%s\IPC$
Microsoft Base Cryptographic Provider v1.0
%d.%d.%d.%d
mssecsvc2.0
Microsoft Security Center (2.0) Service
%s -m security
C:\%s\qeriuwjhrf
C:\%s\%s
WINDOWS
tasksche.exe
CloseHandle
WriteFile
CreateFileA
CreateProcessA
http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwerqwea.com
!This program cannot be run in DOS mode.
```

Figure 2: Floss output for wannacry.exe

PE studio revealed the imports and libraries used, with suspicious use of Microsoft encryption and cryptographic libraries. The tool also highlighted the use of Internet connection imports likely to be used with the URL found in strings. The `Attrib +h` was also uncovered, meaning the binary is executing with the hidden attribute as to not appear in directory listings.

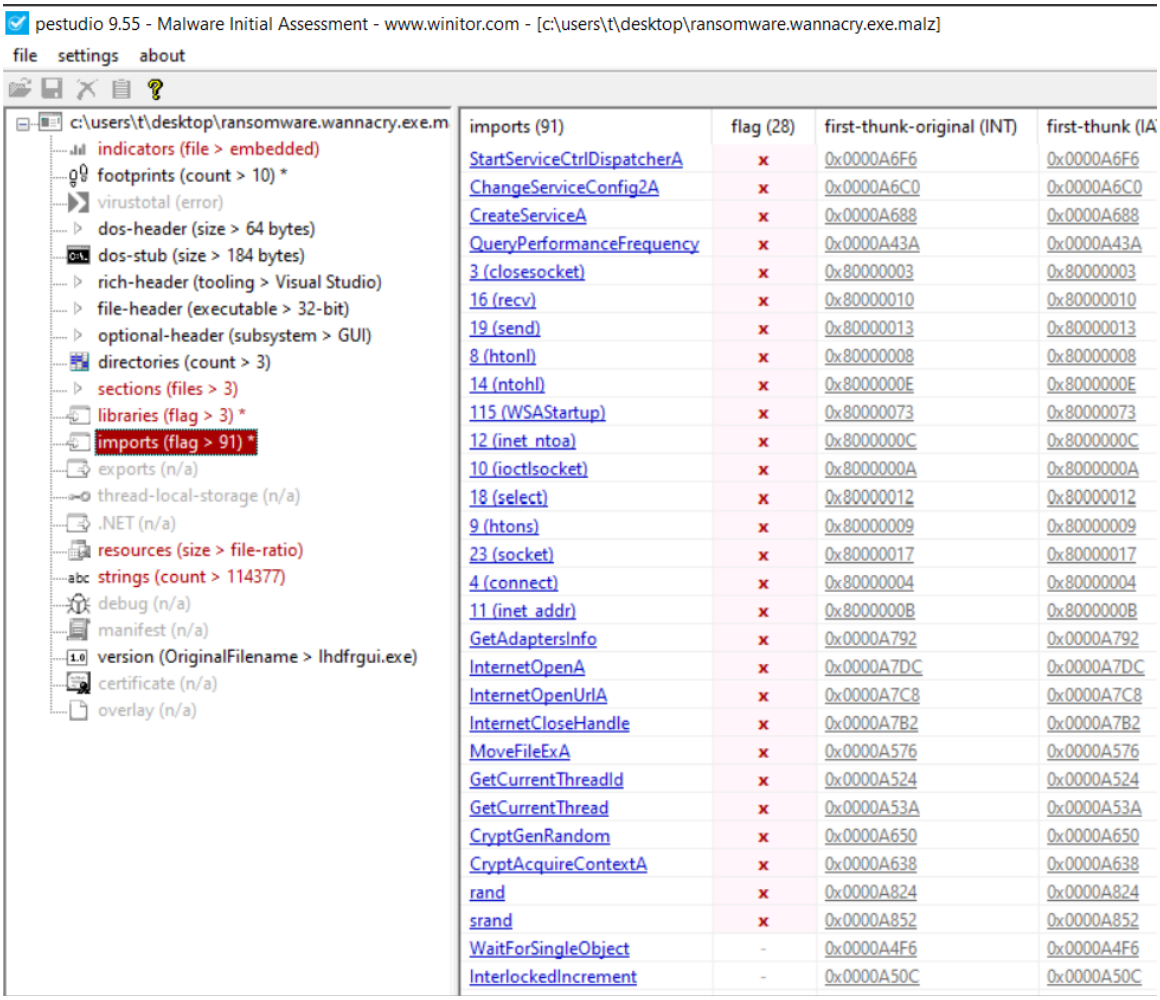

Figure 3: PE Studio findings for Wannacry

PE View was used to check if the binary was packed or not, however, the sizes were very similar indicating that there was no unpacking involved.

5 Basic Dynamic Analysis

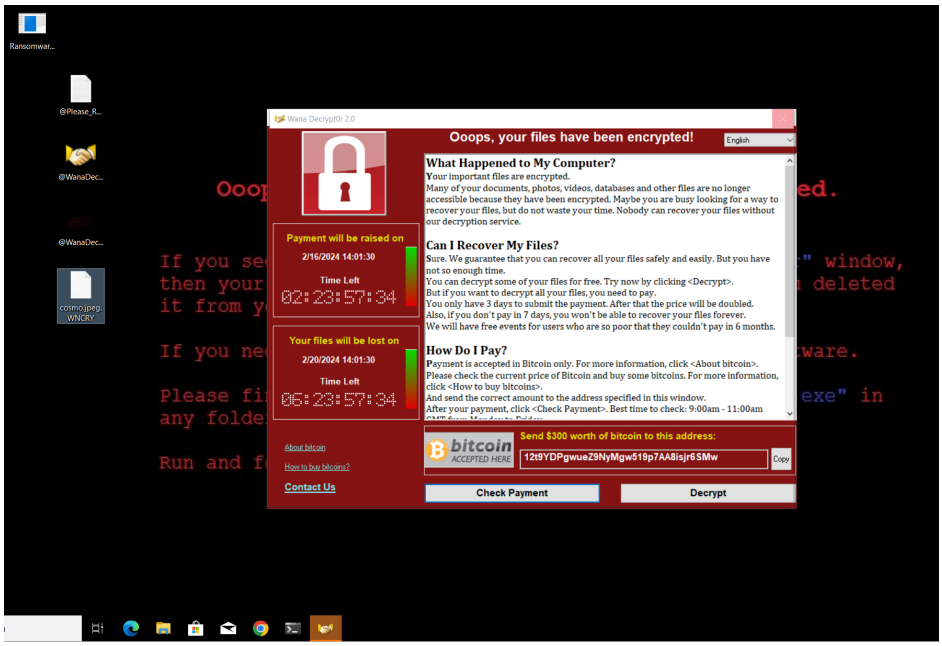

Figure 4: Result after detonation

The binary does nothing upon regular detonation, only triggering when run as an administrator, likely to use the libraries that require elevated permissions. Wannacry has no visible indicators initially, but shortly after every file will become unreadable with the `.WNCRY` extension, the "Wana Decrypt0r 2.0" application will appear and a README text file, and an image will all be found on the desktop. The image will be set as the wallpaper, informing the user that the files on the machine are encrypted, and a ransom must be paid to get them back. The "Wana Decrypt0r" will continuously run and reopen, pressuring the user further with a time limit before the files are lost forever, as well as contact information. Finally, the text file includes similar information to the previous two, just informing and pressuring the victim into paying. All of this can be seen in Figure 4.

Running the binary without an internet connection produces an unreachable response in Wireshark, however, a simulated internet connection (such as INetSim) produces the result in Figure 5. As the suspicious domain is there, this is likely an anti-analysing technique to detect a simulated connection, as the author at the time of writing the binary knew the domain did not exist.

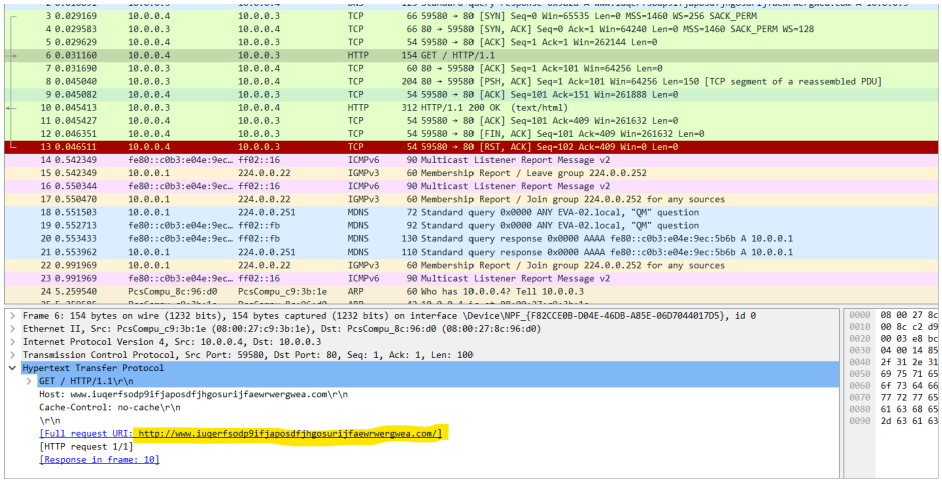

Figure 5: Wireshark picking up the HTTP request to the suspicious domain

Additionally, an array of ARP requests where sent which Wireshark caught, primarily to `10.0.0.3` Remnux machine but also to every possible host on the network. This is typical of a Worm, discovering other hosts to potentially infect, showing the multifaceted nature of this binary.

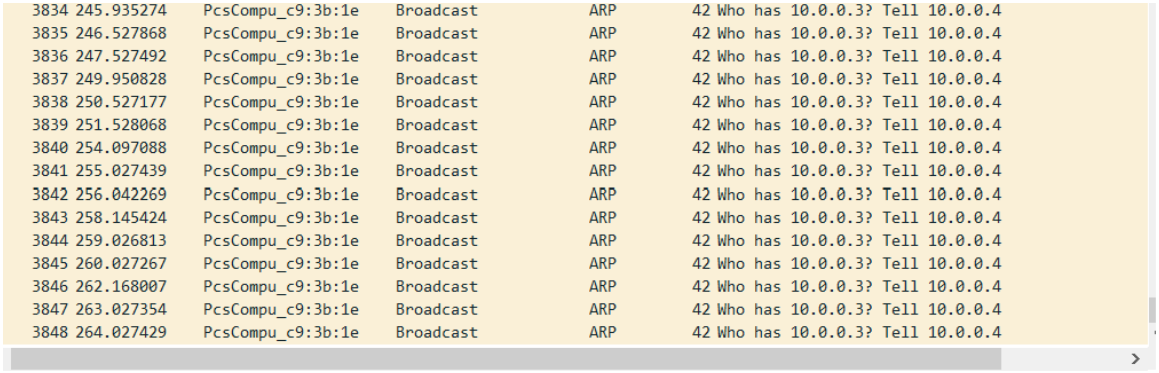

Figure 6: ARP requests from Wannacry

Checking TCP View at detonation shows that these requests are being sent to the Server Message Block (SMB) remote ports of other hosts on the network, as seen in Figure 7. This is likely to share the malicious files and cause further infections, and thus more potential ransoms paid for the threat actor.

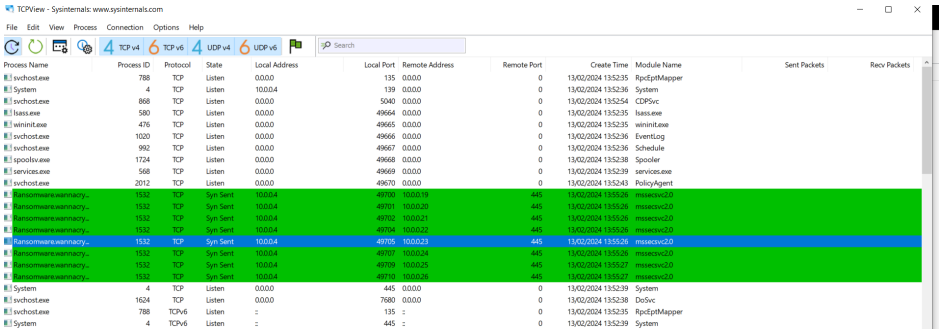

Figure 7: TCP View showing the attempt of spreading the malware

Procmon revealed the creation of the `tasksche.exe` process, which interestingly created a folder within the `ProgramData` of the C drive. Digging further reveals that this is a staging folder for Wannacry, where the various tools are originally stored before being copied to other directories in the victims machine.

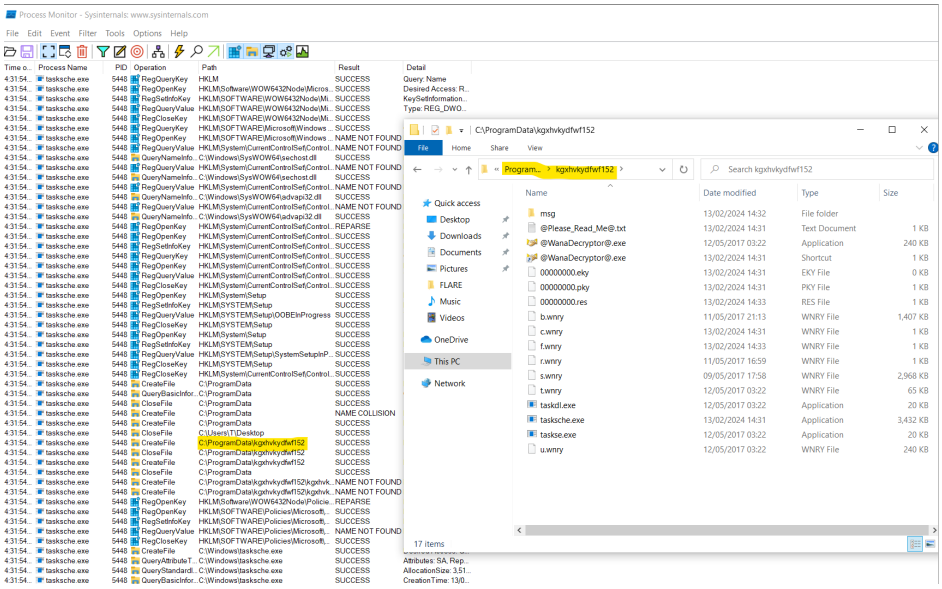

Figure 8: Tasksche.exe creating an unpacking folder

6 Advanced Static Analysis

Wannacry notably contains the request to "hxxtp://www.iugerfsodp9ifjaposdfjhgosurijfaewrwergwea.c and if there is a 200 OK status code will stop before the payload is executed. This technique is used to prevent the analysis of binary in a virtual machine with a simulated internet connection, such as the analysis being conducted for this paper. However, using Cutter's write mode this can be reversed to force Wannacry into executing even after receiving a successful status code. Once the main function of the Assembly code was found, the decision-making statement was easy to identify, essentially jumping to a specific memory address if the condition is met. Selecting this `jne` (jump if not equal) instruction and reversing the jump, as seen in Figure 9, will change it to a `je`(Jump if Equal).

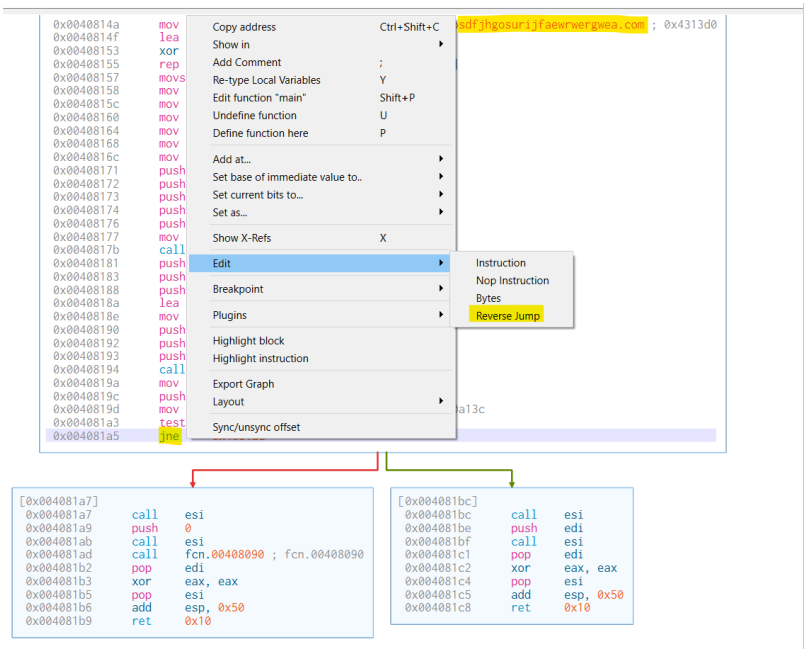

Figure 9: Reversing the Jump in Wannacry Assembly code

This results as intended, with Wannacry executing its full payload with a simulated internet connection, as seen in Figure 10 with INetSim running on the Remnux VM. This is similar to how Wannacry was actually stopped in 2017, with Marcus Hutchins registering the domain that is called in the binary, meaning a successful HTTP status code would be received by any machine with an internet connection, essentially acting as a kill switch for this piece of malware.

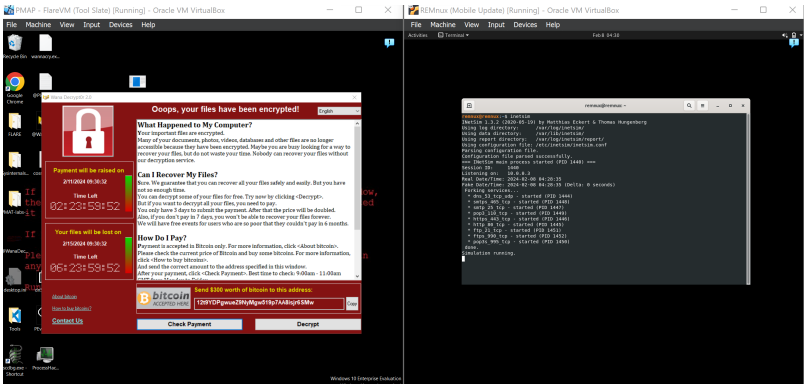

Figure 10: Wannacry infecting machine whilst having a simulated connection

7 Advanced Dynamic Analysis

A debugger can be used to get past the check for simulated internet connection similarly to the Cutter technique, this time changing the register values before the JNE is performed to falsify the result without editing the code. Wannacry.exe was loaded into x32 Debugger, the position was found from the early Cutter analysis, and then the Zero Flag was changed in the JNE from 0 to 1 thus allowing the program to run again even with a simulated internet connection, as seen in Figure 11.

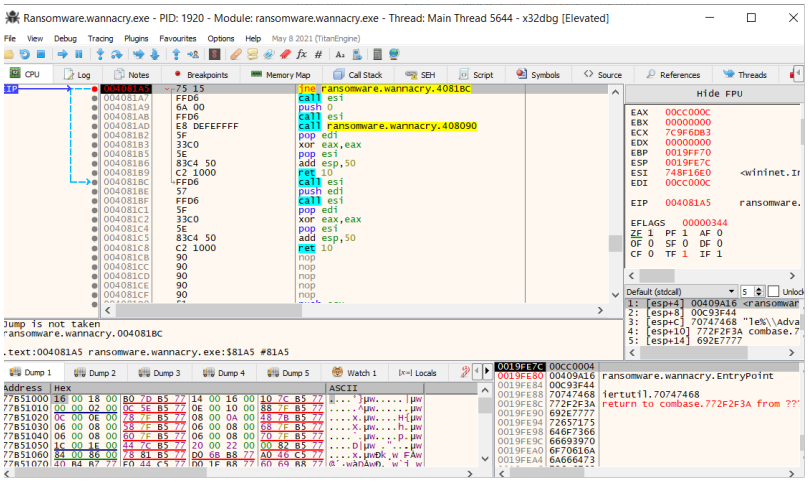

Figure 11: Editing registers to bypass anti-analysis techniques

One of the strings in Wannacry was the "IsDebuggerPresent" import, often used for anti-analysis, however, this could not be found within the binary and did not affect the debugger, suggesting it was a holdover from an older version or simply broken. The same technique could be used on this import, though, so it would not have caused any issues to this investigation.

8 Indicators of Compromise

Like most ransomware, the Indicators of Compromise for Wannacry are rather obvious, as the threat actor responsible is eager for the victim to know and be forced into paying the ransom.

8.1 Host Based Indicators

- **Encrypted file -** All user files on the machine will be encrypted with the `.wcry` file extension.
- **Wannacry software -** The Wana Decrypt0r application will continue to open, attempting to extort the victim to send money to the Bitcoin wallet to unlock their files. This program can be found in every directory of the victim's machine
- **Wannacry wallpaper -** The Desktop wallpaper will change to a warning message exclaiming the machine's files have been encrypted and to use the Wana Decrypt0r software. This file will also appear on the desktop, and can be seen in Figure 12.
- **Wannacry text file -** found on the desktop informing on how to pay.

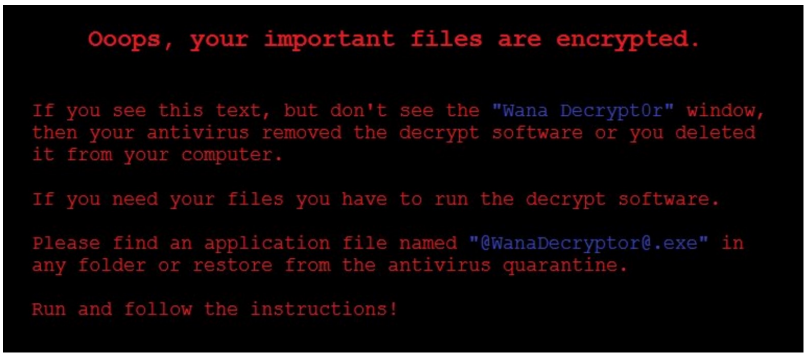

Figure 12: Wannacry wallpaper

8.2 Network Based Indicators

- **Call to suspicious domain -** The binary will call to "hxxtp://www.iugerfsodp9ifjaposdfjhgosurijfaewrwergwea.com", and use the http status code to either continue or abort the payload.
- **Network discovery -** ARP requests to discover other hosts on the network for further infection.

9 Rules and Signatures

Yara rules were written using the indicators identified, focusing on finding wannacry before it can execute or gathering all locations of malicious files after encryption. The full rules can be found in the Appendix, and Figure 13 shows the rules in action flagging the different locations before detonation.

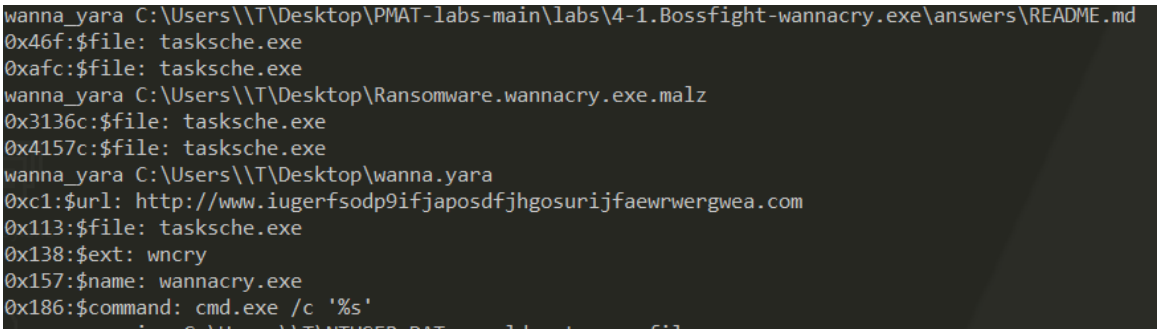

Figure 13: Yara rules for Wannacry

Appendix

Yara Rules

```
rule wanna_yara {

    meta:
        last_updated = "2024-13-02"
        author = "Thomas MacKinnon"
        description = "Yara Rules for Wannacry."

    strings:
        $url = "http://www.iugerfsodp9ifjaposdfjhgosurijfaewrwergwea.com"
        $file = "tasksche.exe" ascii
        $ext = "wncry" ascii
        $name = "wannacry.exe" ascii
        $command = "cmd.exe /c '%s'" ascii



    condition:

        $url or $file or $ext or $name or $command

}
```