



Practical Malware Analysis

WannaCry Ransomware

August 2023 | Xpinux | v1.0

Table of Contents

| | |
|--|-----------|
| Executive Summary | 3 |
| High-Level Technical Summary | 4 |
| Basic Static Analysis | 5 |
| File Type | 5 |
| CAPA Analysis..... | 5 |
| String Analysis..... | 7 |
| PE Studio..... | 8 |
| Basic Dynamic Analysis | 10 |
| Network Analysis:..... | 11 |
| PROCmon..... | 11 |
| RegShot..... | 13 |
| Advanced Static Analysis | 15 |
| Advanced Dynamic Analysis | 22 |
| Indicators of Compromise | 25 |
| Network Indicators..... | 25 |
| Host-based Indicators..... | 25 |
| Cuckoo Analysis & Yara RULE | 28 |
| Appendices | 32 |
| A. Yara Rules | 32 |
| B. Figures..... | 32 |

Executive Summary

| | |
|-------------|---|
| MD5 hash | db349b97c37d22f5ea1d1841e3c89eb4 |
| SHA1 Hash | e889544aff85ffaf8b0d0da705105dee7c97fe26 |
| SHA256 hash | 24d004a104d4d54034dbcffc2a4b19a11f39008a575aa614ea04703480b1022c |
| Format | PE |
| IOC | C:\%s\qeriuwjhrf |
| IOC | WANACRY! |
| IOC | http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com |

Wannacry.Ransomware, a highly sophisticated and notorious malware, was analyzed, revealing a two-stage structure that underscored its devious capabilities. The first stage boasted a cunning killswitch mechanism, designed to avoid detonation if a specific URL was accessible. In this manner, the malware ensured self-preservation and stealthy behavior.

However, when the URL proved unattainable, the ransomware swiftly transitioned to its second stage - a perilous propagation attempt within the network. This propagation stage raised the stakes significantly, intensifying the threat landscape for organizations.

During analysis, we discovered the ransomware's reliance on `tasksche.exe`, skillfully employed to unpack files into a mysterious directory nestled within `ProgramData`. This intelligent maneuver enabled the malware to establish persistence, complicating detection and removal efforts.

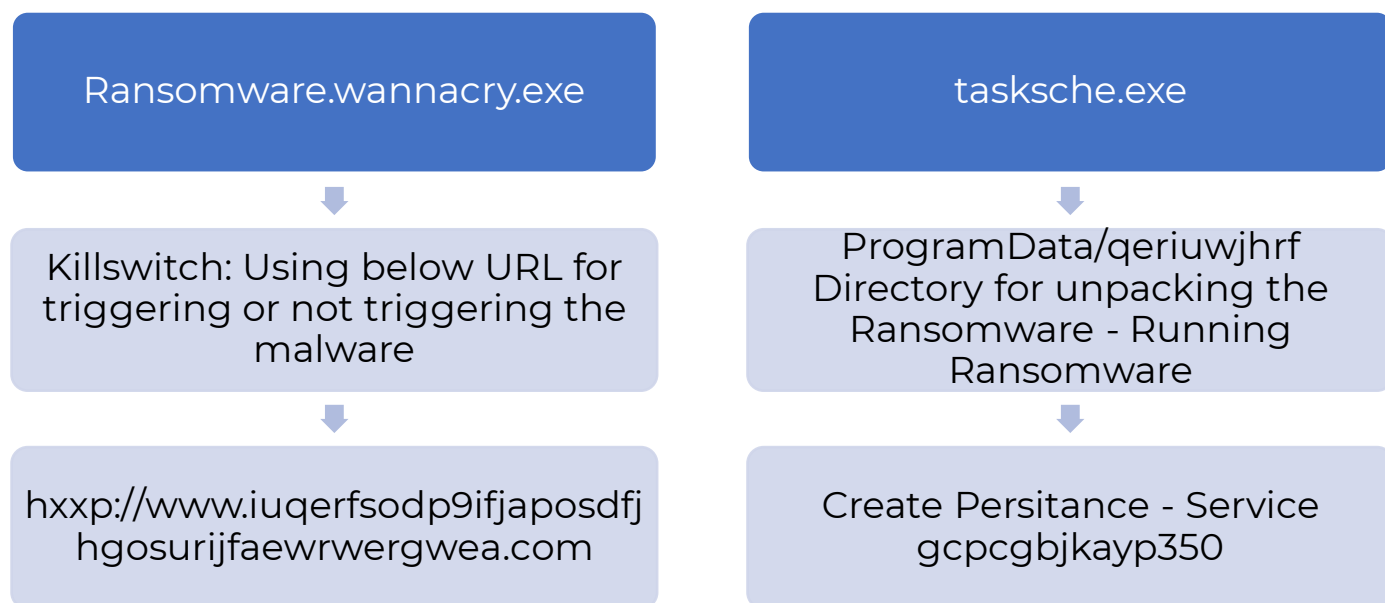
Once the ransomware was in full motion, it executed a relentless encryption process, rendering critical data inaccessible to its victims. To exacerbate matters, it brazenly presented a disconcerting popup, demanding a ransom for the coveted decryption key.

In response to this ominous threat and its potential impact on businesses, we emphasize the urgency of enhancing cybersecurity defenses and fortifying employee awareness. Proactive measures and continuous monitoring are paramount to safeguarding against Wannacry.Ransomware and similar malicious adversaries. By adopting a robust cybersecurity posture, organizations can better protect their digital assets and ensure uninterrupted operations amidst the evolving cyber landscape.

YARA signature rule is attached in Appendix A. Malware sample and hashes have been submitted to VirusTotal with a **Score of 68/71** Detections.

High-Level Technical Summary

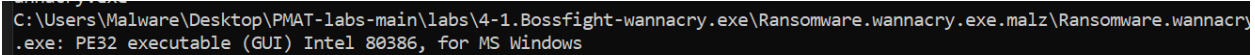
Wannacry.Ransomware is a multi-stage malware comprising a killswitch mechanism and a propagation stage. The killswitch checks the reachability of a URL, preventing detonation if successful. However, failure to reach the URL initiates the propagation process within the network. In the second stage, the malware creates a tasksche.exe process to unpack ransomware files into a peculiar directory within ProgramData. Additionally, it establishes a persistent strange service. Subsequently, the ransomware encrypts data and presents a popup demanding ransom for decryption. This sophisticated ransomware poses a significant threat, necessitating robust security measures and vigilant network monitoring to counter its potential impact.



Basic Static Analysis

File Type

Using File Type, we identify that the Malware Sample is a PE32 Executable (32 Bit) Application.

A screenshot of a Windows command prompt window showing the output of the 'file' command. The path is C:\Users\Malware\Desktop\PMAT-labs-main\labs\4-1.Bossfight-wannacry.exe\Ransomware.wannacry.exe.malz\Ransomware.wannacry.exe. The output is: PE32 executable (GUI) Intel 80386, for MS Windows.

```
C:\Users\Malware\Desktop\PMAT-labs-main\labs\4-1.Bossfight-wannacry.exe\Ransomware.wannacry.exe.malz\Ransomware.wannacry.exe: PE32 executable (GUI) Intel 80386, for MS Windows
```

Figure 1: File Type

CAPA Analysis

Using CAPA without any arguments we can gain a first insight of some of the capabilities of the Malware sample. CAPA detects capabilities in executable files. You run it against a PE, ELF, .NET module, or shellcode file and it tells you what it thinks the program can do. For example, it might suggest that the file is a backdoor, is capable of installing services, or relies on HTTP to communicate.

The CAPA output indicates that the malware sample uses ATT&CK tactics, and by analyzing them, we can gain a preliminary understanding of the malware's capabilities.

➤ Defense Evasion: Obfuscated Files or Information (T1027.005)

The malware uses obfuscation techniques to make its files or information harder to detect and analyze. Obfuscation is a common tactic used by malware authors to hide the true intent of their code and avoid detection by security solutions.

➤ Discovery: a. File and Directory Discovery (T1083)

The malware attempts to gather information about files and directories on the infected system. This information can be used to understand the system's structure and locate potential targets for further exploitation or data exfiltration.

b. System Information Discovery (T1082)

The malware conducts actions to collect information about the infected system. This could include details about the operating system, hardware, software, and other relevant system information.

c. System Network Configuration Discovery (T1016)

The malware tries to gather details about the network configuration of the infected system. This information helps the malware to identify available network resources, potential targets, and ways to propagate across the network.

➤ Execution:

a. Shared Modules (T1129)

The malware utilizes shared modules or dynamic link libraries (DLLs) to execute its malicious code. By using shared modules, the malware can avoid raising suspicions since these files are commonly used by legitimate software.

b. System Services::Service Execution (T1569.002)

The malware leverages system services to execute its code. It may interact with legitimate services or create its own service to achieve persistence and maintain a presence on the infected system.

➤ Persistence: Create or Modify System Process (T1543.003)

The malware employs a technique to establish persistence by creating or modifying system processes. This allows the malware to automatically start each time the system boots or certain events occur, ensuring its continued presence and operation.

- Further Below we can check the Detailed Capabilities of the malware sample:

Notable examples are that it uses Conditional Execution as Service, C2 Communication to send and receive data and the Cryptography API Call.

| MBC objective | MBC Behavior |
|--------------------------|---|
| ANTI-BEHAVIORAL ANALYSIS | Conditional Execution::Runs as Service [B0025.007] |
| ANTI-STATIC ANALYSIS | Debugger Detection::Timing/Delay Check QueryPerformanceCounter [B0001.03] |
| COMMAND AND CONTROL | Executable Code obfuscation::Argument obfuscation [B0032.020] |
| COMMUNICATION | Executable Code obfuscation::Stack Strings [B0032.017] |
| | C2 Communication::Receive Data [B0030.002] |
| | C2 Communication::Send Data [B0030.001] |
| | HTTP Communication::Create Request [C0002.012] |
| | HTTP Communication::Open URL [C0002.004] |
| | Socket Communication::Connect Socket [C0001.004] |
| | Socket Communication::Create TCP socket [C0001.011] |
| | Socket Communication::Create UDP Socket [C0001.010] |
| | Socket Communication::Get Socket Status [C0001.012] |
| | Socket Communication::Initialize Winsock Library [C0001.009] |
| | Socket Communication::Receive Data [C0001.006] |
| | Socket Communication::Send Data [C0001.007] |
| | Socket Communication::Set Socket Config [C0001.001] |
| | Socket Communication::TCP Client [C0001.008] |
| CRYPTOGRAPHY | Generate Pseudo-random Sequence::Use API [C0021.003] |
| DATA | Compression Library [C0060] |
| DISCOVERY | Code Discovery::Inspect Section Memory Permissions [B0046.002] |
| EXECUTION | File and Directory Discovery [E1083] |
| FILE SYSTEM | Install Additional Program [B0023] |
| | Move File [C0063] |
| | Read File [C0051] |
| PROCESS | Create Thread [C0038] |
| | Terminate Process [C0018] |
| | Terminate Thread [C0039] |

Figure 2:MBC Objectives

String Analysis

String analysis in malware analysis involves extracting human-readable text (strings) from malware code to reveal C2 communication, encryption keys, file paths, function names, and IOCs. It helps researchers understand malware behavior and develop mitigation strategies.

By utilizing Floss with the "-n 8" argument and directing the output to a text file, we can analyze the strings within the malware sample. The initial observations reveal Win API Calls employed by the malware, with notable detections such as CryptAcquireContextA, CryptGenRandom, CryptGenKey, CryptDecrypt, CryptEncrypt, CryptDestroyKey, CryptImportKey, and CryptAcquireContextA.

Additionally, we notice the recurrent presence of the string "!This program cannot be run in DOS mode." *This suggests that the executable may contain packed other programs.*

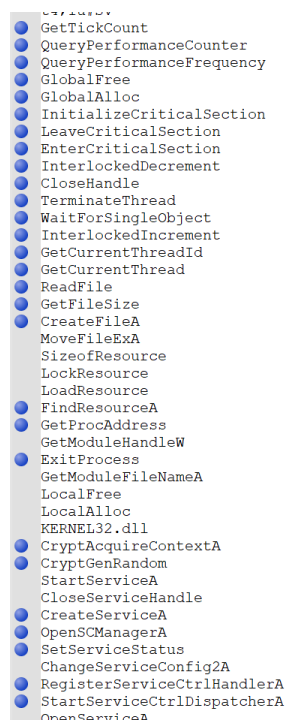


Figure 3: Sample of APIs Used

During the string analysis process, we have discovered intriguing strings, including a notable IOC - a URL: **<http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com>**. This suggests that the malware sample attempts to connect to this URL. Moreover, we've come across an unusual directory indicated by the "%s" string: **C:%s\qeriuwjhrf**. The usage of **tasksche.exe** is also observed.

Furthermore, we've identified the usage of command lines with the "%s" string, indicating potential command-line arguments being passed: **cmd.exe /c "%s"**. Encoded strings have been detected as well, along with the usage of **"icacls . /grant Everyone:F /T /C /Q"**, a command that modifies permissions on directories and files. Lastly, we've encountered the string **"WANACRY!"**.

This string analysis process has provided valuable insights into the behavior and characteristics of the malware sample.

```

518  advapi32.dll
519  ● WANACRY!
520  CloseHandle
521  DeleteFileW
522  MoveFileExW
523  MoveFileW
524  ReadFile
525  WriteFile
526  CreateFileW
527  kernel32.dll
528  2/O-_.X8w.+
529  Microsoft Enhanced RSA and AES Cryptographic Provider
530  ● CryptGenKey
531  ● CryptDecrypt
532  ● CryptEncrypt
533  ● CryptDestroyKey
534  ● CryptImportKey
535  ● CryptAcquireContextA
536  ● cmd.exe /c "%s"
537  ● 115p7UMMngoj1pMvbkpHijcRdfJNXj6LrLn
538  ● 12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw
539  ● 13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94
540  ● Global\MsWinZonesCacheCounterMutexA
541  ● tasksche.exe
542  TaskStart
543  ● icaccls . /grant Everyone:F /T /C /Q
544  attrib +h .
545  ● WNCry@2017
546  GetNativeSystemInfo
547  .?AVexception@@

```

Figure 4: IOC from Strings

PE Studio

Using PE Studio we can get detailed information about this malware Sample

| property | value |
|------------------|--|
| md5 | DB349B97C37D22F5EA1D1841E3C89EB4 |
| sha1 | E889544AFF85FFAF8B0D0DA705105DEE7C97FE26 |
| sha256 | 24D004A104D4D54034DBCFFC2A4B19A11F39008A575AA614EA04703480B1022C |
| first-bytes-hex | 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 |
| first-bytes-text | M Z @ |
| file-size | 3723264 bytes |
| entropy | 7.964 |
| imphash | n/a |
| signature | Microsoft Visual C++ v6.0 |
| tooling | Visual Studio 6.0 |
| entry-point | 55 8B EC 6A FF 68 A0 A1 40 00 68 A2 9B 40 00 64 A1 00 00 00 00 50 64 89 25 00 00 00 00 83 EC 68 53 |
| file-version | 6.1.7601.17514 (win7sp1_rtm.101119-1850) |
| description | Microsoft® Disk Defragmenter |
| file-type | executable |
| cpu | 32-bit |
| subsystem | GUI |
| compiler-stamp | Sat Nov 20 09:03:08 2010 UTC |
| debugger-stamp | n/a |
| resources-stamp | 0x00000000 |
| import-stamp | 0x00000000 |
| exports-stamp | n/a |

Figure 5: PEstudio Info

Information such as hashes, file size, the first bytes, and the CPU architecture can provide valuable insights into the design of this malware. Additionally, examining indicators allows us to gain immediate insight into the suspicious components of the malware sample. Most importantly, we can cross-reference flagged suspicious libraries from PE studio with the API calls detected during string analysis.

c:\users\malware\desktop\pmat-labs-main\labs\

indicators (file > embedded) *

virusotal (error)

dos-header (64 bytes)

dos-stub (184 bytes)

rich-header (Visual Studio)

file-header (Intel-386)

optional-header (GUI)

directories (3)

sections (files) *

libraries (flag)

imports (flag)

exports (n/a)

tls-callback (n/a)

.NET (n/a)

resources (size > file-ratio)

strings (size) *

debug (n/a)

manifest (n/a)

version (lhdfrgui.exe)

certificate (n/a)

overlay (n/a)

imports (91)

flag (28)

StartServiceCtrlDispatcherA

x

ChangeServiceConfig2A

x

CreateServiceA

x

QueryPerformanceFrequency

x

3 (closesocket)

x

16 (recv)

x

19 (send)

x

8 (htonl)

x

14 (ntohl)

x

115 (WSAStartup)

x

12 (inet_ntoa)

x

10 (ioctlsocket)

x

18 (select)

x

9 (htons)

x

23 (socket)

x

4 (connect)

x

11 (inet_addr)

x

GetAdaptersInfo

x

InternetOpenA

x

InternetOpenUrlA

x

InternetCloseHandle

x

MoveFileExA

x

GetCurrentThreadId

x

GetCurrentThread

x

CryptGenRandom

x

CryptAcquireContextA

x

rand

x

srand

x

Figure 6: PEstudio Imports Indicators

Basic Dynamic Analysis

Setting the environment for Dynamic Analysis:

We will configure ProcMon, starting with a process name filter for the malware sample. Additionally, we will open TCPView and Procexp. Finally, we will take an initial registry snapshot using RegShot.

- **Network Detonation:** After the initial detonation of the malware with internet capabilities using inetsim, it appears that the payload is not triggered or activated.

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|-------------|----------|-------------|----------|--------|--|
| 1 | 0.000000000 | 10.0.0.3 | 10.0.0.1 | TCP | 74 | 49348 → 53 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1413818008 TSecr=0 WS=128 |
| 2 | 1.030586964 | 10.0.0.3 | 10.0.0.1 | TCP | 74 | [TCP Retransmission] [TCP Port numbers reused] 49348 → 53 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1413818008 TSecr=0 WS=128 |
| 3 | 2.916025487 | 10.0.0.2 | 10.0.0.3 | DNS | 169 | Standard query 0x6a45 A www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com |
| 4 | 2.919769921 | 10.0.0.3 | 10.0.0.2 | DNS | 125 | Standard query response 0x6a45 A www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com A 10.0.0.3 |
| 5 | 2.927282599 | 10.0.0.2 | 10.0.0.3 | TCP | 66 | 50706 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 6 | 2.927217028 | 10.0.0.3 | 10.0.0.2 | TCP | 66 | 80 → 50706 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128 |
| 7 | 2.927356879 | 10.0.0.2 | 10.0.0.3 | TCP | 66 | 50706 → 80 [ACK] Seq=1 Ack=1 Win=262144 Len=0 |
| 8 | 2.928071422 | 10.0.0.2 | 10.0.0.3 | HTTP | 154 | GET / HTTP/1.1 |
| 9 | 2.928077812 | 10.0.0.3 | 10.0.0.2 | TCP | 54 | 80 → 50706 [ACK] Seq=1 Ack=101 Win=64256 Len=0 |
| 10 | 2.938554513 | 10.0.0.3 | 10.0.0.2 | TCP | 204 | 80 → 50706 [PSH, ACK] Seq=1 Ack=101 Win=64256 Len=150 [TCP segment of a reassembled PDU] |
| 11 | 2.938759211 | 10.0.0.2 | 10.0.0.3 | TCP | 60 | 50706 → 80 [ACK] Seq=101 Ack=151 Win=261888 Len=0 |
| 12 | 2.938765960 | 10.0.0.3 | 10.0.0.2 | HTTP | 312 | HTTP/1.1 200 OK (text/html) |
| 13 | 2.938878903 | 10.0.0.2 | 10.0.0.3 | TCP | 60 | 50706 → 80 [ACK] Seq=101 Ack=409 Win=261632 Len=0 |
| 14 | 2.938974807 | 10.0.0.2 | 10.0.0.3 | TCP | 60 | 50706 → 80 [FIN, ACK] Seq=101 Ack=409 Win=261632 Len=0 |

Figure 7: Network Detonation - Enigmatic URL

The malware sample attempts to communicate with the unusual URL, acting as a killswitch; if reached, the malware will not detonate. Interestingly, even with administrative privileges, the malware fails to trigger.

- **Without Network Simulation:**

The malware sample successfully triggers and encrypts our data. Additionally, we encounter the infamous picture associated with WannaCry ransomware, indicating a potential ransomware infection.

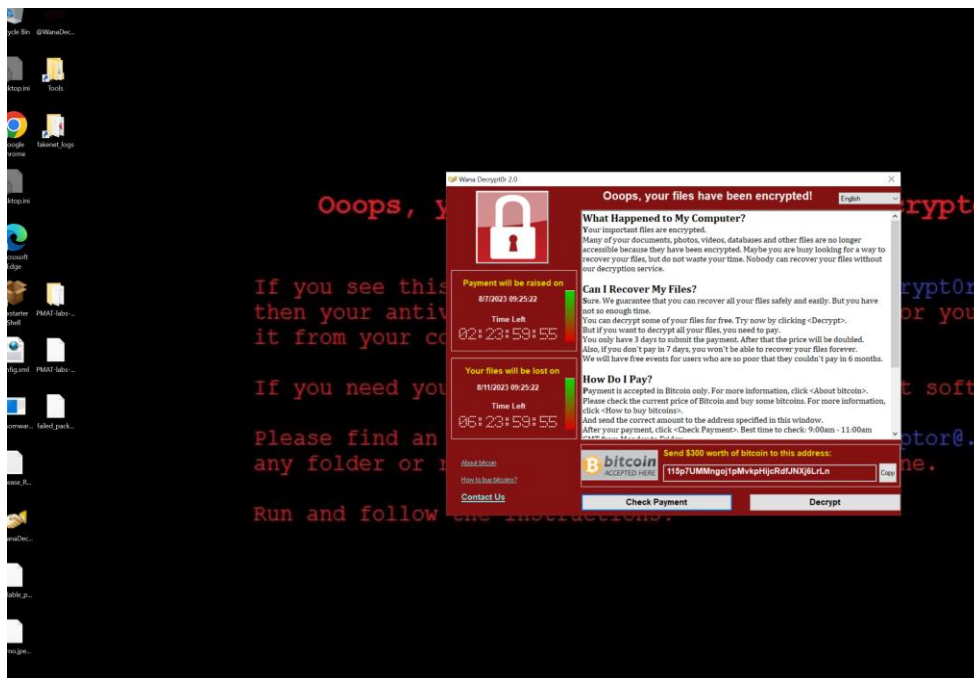


Figure 8: Wannacry Ransomware

Network Analysis:

During the initial activation of the WannaCry ransomware, we can clearly observe the process attempting to communicate with other systems in our network using SMB, aiming to propagate itself and function as a network worm.

Following the ransomware's activation, we observe the WannaCry_Decryptor@exe establishing a connection to remote port 9050.

| | | | | | | | | | |
|--------------------|------|-----|-------------|-----------|-------|-----------|-------|---------------------|--------------------|
| tasksvcs.exe | 1684 | TCP | Established | 127.0.0.1 | 61495 | 127.0.0.1 | 61496 | 04/08/2023 09:24:57 | tasksvcs.exe |
| tasksvcs.exe | 1684 | TCP | Established | 127.0.0.1 | 61495 | 127.0.0.1 | 61495 | 04/08/2023 09:24:57 | tasksvcs.exe |
| tasksvcs.exe | 1684 | TCP | Established | 127.0.0.1 | 9050 | 127.0.0.1 | 21709 | 04/08/2023 09:28:37 | tasksvcs.exe |
| @WanaDecryptor@exe | 2716 | TCP | Established | 127.0.0.1 | 21709 | 127.0.0.1 | 9050 | 04/08/2023 09:28:37 | @WanaDecryptor@exe |
| System | 4 | TCP | Listen | 10.0.0.2 | 139 | 0.0.0.0 | 0 | 31/07/2023 13:13:39 | System |

Figure 9: Immediately after running TCPview

Procmon

Procmon (Process Monitor) is a Windows tool used for malware analysis. It monitors and logs system activities, providing insights into file system, registry, and process behavior. Analysts use Procmon to understand malware actions, track changes, and identify potential malicious activities. Its real-time monitoring aids in detecting and analyzing malware behavior efficiently.

With Procmon, we filtered out the sample using the executable file's process name. After detonating the malware, we can view its behavior in parts. The first part involves process and thread creation.

From the analysis, we detect that the malware created a new process called "taskshe.exe" with PID 5692.

Furthermore, we can see the usage of other dll files like bcrypt for its ransomware purpose.

| | | | | | | |
|-----|-----------------|------|---------------|---|---------|----------------------|
| 2.. | Ransomware.w... | 7840 | Load Image | C:\Windows\SysWOW64\urlmon.dll | SUCCESS | Image Base: 0x728... |
| 2.. | Ransomware.w... | 7840 | Load Image | C:\Windows\SysWOW64\netutils.dll | SUCCESS | Image Base: 0x74c... |
| 2.. | Ransomware.w... | 7840 | Load Image | C:\Windows\SysWOW64\svchost.dll | SUCCESS | Image Base: 0x73a... |
| 2.. | Ransomware.w... | 7840 | Load Image | C:\Windows\SysWOW64\oleaut32.dll | SUCCESS | Image Base: 0x770... |
| 2.. | Ransomware.w... | 7840 | Load Image | C:\Windows\SysWOW64\dnsapi.dll | SUCCESS | Image Base: 0x73f... |
| 2.. | Ransomware.w... | 7840 | Load Image | C:\Windows\SysWOW64\rsadpapi.dll | SUCCESS | Image Base: 0x73f... |
| 2.. | Ransomware.w... | 7840 | Thread Create | | SUCCESS | Thread ID: 8792 |
| 2.. | Ransomware.w... | 7840 | Load Image | C:\Windows\SysWOW64\cryptsp.dll | SUCCESS | Image Base: 0x74a... |
| 2.. | Ransomware.w... | 7840 | Load Image | C:\Windows\SysWOW64\rsaenh.dll | SUCCESS | Image Base: 0x73e... |
| 2.. | Ransomware.w... | 7840 | Load Image | C:\Windows\SysWOW64\bcrypt.dll | SUCCESS | Image Base: 0x75f... |
| 2.. | Ransomware.w... | 7840 | Load Image | C:\Windows\SysWOW64\cryptbase.dll | SUCCESS | Image Base: 0x745... |
| 2.. | Ransomware.w... | 7840 | Load Image | C:\Windows\SysWOW64\cryptprimitives.dll | SUCCESS | Image Base: 0x76c... |
| 2.. | Ransomware.w... | 7840 | Thread Create | | SUCCESS | Thread ID: 8456 |

Figure 10: DLL Used

| | | | | | | |
|-------|-----------------|------|----------------|------------------------|---------|-----------------------|
| 212.. | Ransomware.w... | 7840 | Thread Create | | SUCCESS | Thread ID: 7564 |
| 212.. | Ransomware.w... | 7840 | Thread Create | | SUCCESS | Thread ID: 4936 |
| 212.. | Ransomware.w... | 5692 | Process Create | C:\WINDOWS\taskshe.exe | SUCCESS | PID: 5772, Comma... |
| 212.. | Ransomware.w... | 7840 | Thread Create | | SUCCESS | Thread ID: 512 |
| 212.. | Ransomware.w... | 5692 | Thread Exit | | SUCCESS | Thread ID: 6024, U... |

Figure 11: Creation of taskshe.exe

We can view the process tree of the spawned processes from the WannaCry ransomware.

| | | | | | | | | |
|-----------------------------|--|--|--|----------------------|----------------------|--|------------------------|-------------------|
| Ransomware.wannacry.exe (7) | | | | Microsoft® Disk D... | C:\Users\Malware\... | | Microsoft Corporati... | NT AUTHORITY\S... |
| cmd.exe (6536) | | | | Windows Comm... | C:\Windows\syste... | | Microsoft Corporati... | NT AUTHORITY\S... |
| tasksche.exe (1128) | | | | DiskPart | C:\ProgramData\g... | | Microsoft Corporati... | NT AUTHORITY\S... |
| attrib.exe (9176) | | | | Attribute Utility | C:\Windows\SysW... | | Microsoft Corporati... | NT AUTHORITY\S... |
| Conhost.exe (6968) | | | | Console Window H... | C:\Windows\Syste... | | Microsoft Corporati... | NT AUTHORITY\S... |

Figure 12: Process Tree

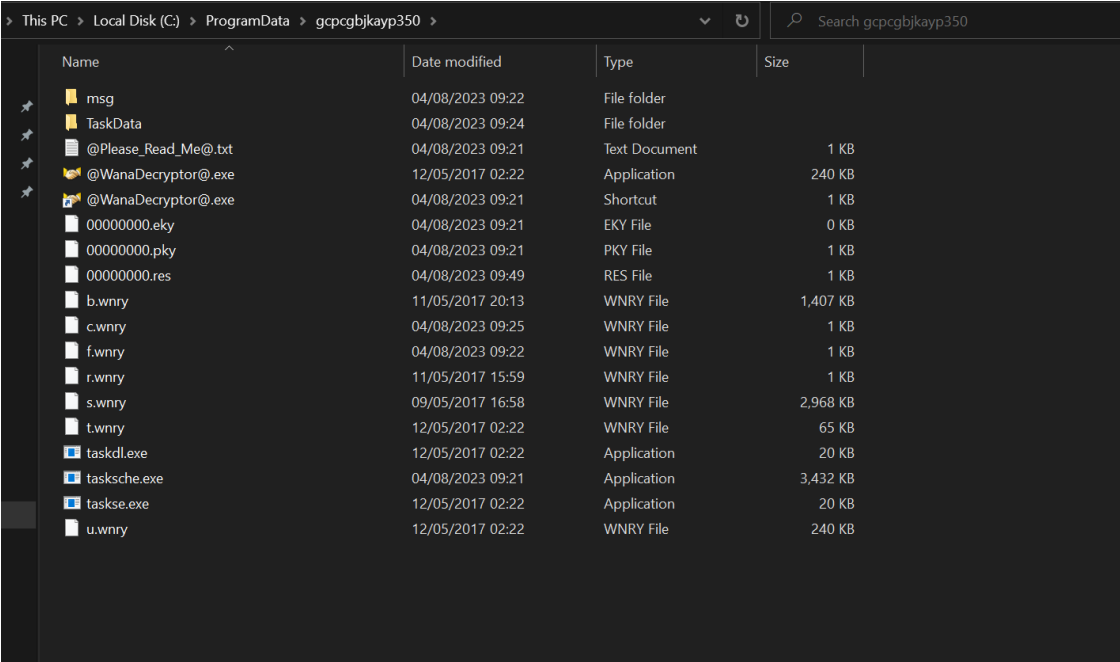
As we observe, it opens a cmd and the tasksche process, which we previously noticed. Now, we will filter Procmon with the parent PID of taskshe.exe to uncover additional evidence of the malware detonation.

Upon filtering with this parent PID, we obtain the information discovered earlier during string analysis, indicating a peculiar directory in the system.

| | | | | |
|------------|--------------|--------|--------------------|--|
| 99:21:2... | tasksche.exe | 5 / 72 | QuerySecurityFile | C:\Windows\tasksche.exe |
| 99:21:2... | tasksche.exe | 5772 | SetEndOfFileIn... | C:\ProgramData\gpcpgbjkayp350\tasksche.exe |
| 99:21:2... | tasksche.exe | 5772 | ReadFile | C:\Windows\tasksche.exe |
| 99:21:2... | tasksche.exe | 5772 | WriteFile | C:\ProgramData\gpcpgbjkayp350\tasksche.exe |
| 99:21:2... | tasksche.exe | 5772 | ReadFile | C:\Windows\tasksche.exe |
| 99:21:2... | tasksche.exe | 5772 | WriteFile | C:\ProgramData\gpcpgbjkayp350\tasksche.exe |
| 99:21:2... | tasksche.exe | 5772 | ReadFile | C:\Windows\tasksche.exe |
| 99:21:2... | tasksche.exe | 5772 | WriteFile | C:\ProgramData\gpcpgbjkayp350\tasksche.exe |
| 99:21:2... | tasksche.exe | 5772 | ReadFile | C:\Windows\tasksche.exe |
| 99:21:2... | tasksche.exe | 5772 | WriteFile | C:\ProgramData\gpcpgbjkayp350\tasksche.exe |
| 99:21:2... | tasksche.exe | 5772 | ReadFile | C:\Windows\tasksche.exe |
| 99:21:2... | tasksche.exe | 5772 | WriteFile | C:\ProgramData\gpcpgbjkayp350\tasksche.exe |
| 99:21:2... | tasksche.exe | 5772 | ReadFile | C:\Windows\tasksche.exe |
| 99:21:2... | tasksche.exe | 5772 | WriteFile | C:\ProgramData\gpcpgbjkayp350\tasksche.exe |
| 99:21:2... | tasksche.exe | 5772 | SetBasicInfor... | C:\ProgramData\gpcpgbjkayp350\tasksche.exe |
| 99:21:2... | tasksche.exe | 5772 | QueryRemotePr... | C:\ProgramData\gpcpgbjkayp350\tasksche.exe |
| 99:21:2... | tasksche.exe | 5772 | CloseFile | C:\ProgramData\gpcpgbjkayp350\tasksche.exe |
| 99:21:2... | tasksche.exe | 5772 | CloseFile | C:\Windows\tasksche.exe |
| 99:21:2... | tasksche.exe | 5772 | CreateFile | C:\ProgramData\gpcpgbjkayp350\tasksche.exe |
| 99:21:2... | tasksche.exe | 5772 | QueryBasicInfor... | C:\ProgramData\gpcpgbjkayp350\tasksche.exe |
| 99:21:2... | tasksche.exe | 5772 | CloseFile | C:\ProgramData\gpcpgbjkayp350\tasksche.exe |
| 99:21:5... | tasksche.exe | 5772 | CloseFile | C:\Windows |
| 99:21:5... | tasksche.exe | 5772 | CloseFile | C:\ProgramData\gpcpgbjkayp350 |

Figure 13: Parent PID Analysis

After inspecting that strange directory in ProgramData, we have come to realize that it is the location where the ransomware unpacked itself and executed the notorious application.



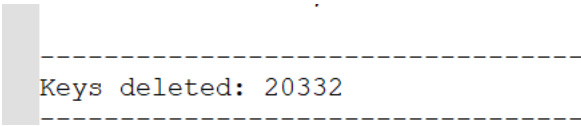
| Name | Date modified | Type | Size |
|----------------------|------------------|---------------|----------|
| msg | 04/08/2023 09:22 | File folder | |
| TaskData | 04/08/2023 09:24 | File folder | |
| @Please_Read_Me@.txt | 04/08/2023 09:21 | Text Document | 1 KB |
| @WanaDecryptor@.exe | 12/05/2017 02:22 | Application | 240 KB |
| @WanaDecryptor@.exe | 04/08/2023 09:21 | Shortcut | 1 KB |
| 00000000.eky | 04/08/2023 09:21 | EKY File | 0 KB |
| 00000000.pky | 04/08/2023 09:21 | PKY File | 1 KB |
| 00000000.res | 04/08/2023 09:49 | RES File | 1 KB |
| b.wnry | 11/05/2017 20:13 | WNRy File | 1,407 KB |
| c.wnry | 04/08/2023 09:25 | WNRy File | 1 KB |
| f.wnry | 04/08/2023 09:22 | WNRy File | 1 KB |
| r.wnry | 11/05/2017 15:59 | WNRy File | 1 KB |
| s.wnry | 09/05/2017 16:58 | WNRy File | 2,968 KB |
| t.wnry | 12/05/2017 02:22 | WNRy File | 65 KB |
| taskdl.exe | 12/05/2017 02:22 | Application | 20 KB |
| tasksche.exe | 04/08/2023 09:21 | Application | 3,432 KB |
| taskse.exe | 12/05/2017 02:22 | Application | 20 KB |
| u.wnry | 12/05/2017 02:22 | WNRy File | 240 KB |

Figure 14: Unpacking Directory

RegShot

Regshot is a utility used in malware analysis to capture and compare system registry snapshots, aiding in identifying changes made by malware to the Windows registry. During our initial static analysis, we noticed that the malware can modify services. By using Regshot, we can discern the specific changes, deletions, and additions made to the registry, comparing a clean snapshot to the one taken after the malware was triggered.

From the comparison, we observed that the ransomware malware deleted 20332 keys.



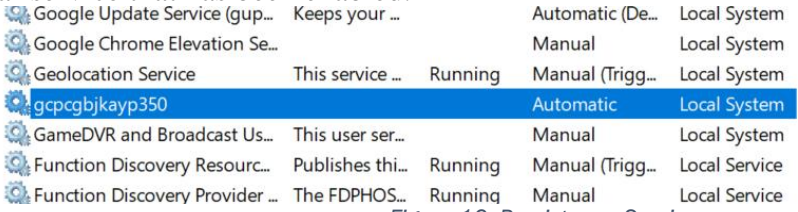
```

-----
Keys deleted: 20332
-----

```

Figure 15: Keys Deleted from Ransomware

We can also observe that it created some keys, and one of them points to the creation of a new service. Furthermore, upon inspecting the Windows services, we can identify the presence of a peculiar service that has been enabled.



| Service Name | Description | Status | Startup Type | Path |
|---------------------------------|------------------|---------|------------------|---------------|
| Google Update Service (gup...) | Keeps your ... | | Automatic (De... | Local System |
| Google Chrome Elevation Se... | | | Manual | Local System |
| Geolocation Service | This service ... | Running | Manual (Trigg... | Local System |
| gcpcgjbkayp350 | | | Automatic | Local System |
| GameDVR and Broadcast Us... | This user ser... | | Manual | Local System |
| Function Discovery Resourc... | Publishes thi... | Running | Manual (Trigg... | Local Service |
| Function Discovery Provider ... | The FDPHOS... | Running | Manual | Local Service |

Figure 16: Persistence Service

```
-----
Keys added: 49
-----
```

```
HKLM\SOFTWARE\WOW6432Node\WanaCrypt0r
HKLM\SYSTEM\ControlSet001\Services\gcpcgjbjkayp350
HKLM\SYSTEM\ControlSet001\Services\mssecsvc2.0
HKLM\SYSTEM\CurrentControlSet\Services\gcpcgjbjkayp350
HKLM\SYSTEM\CurrentControlSet\Services\mssecsvc2.0
HKU\DEFAULT\Software\Microsoft\Windows Script Host
-----
Figure 17: Keys Added
```

```
-----
Values added: 214
-----
HKLM\SOFTWARE\WOW6432Node\WanaCrypt0r\wd: "C:\ProgramData\gcpcgjbjkayp350"
-----
Figure 18: Values Added
```

It also added values to some keys. One of these values is the new strange directory that unpacks the ransomware, as we analyzed earlier.

Other values include the ones below, indicating that the ransomware is running "cmd" as "taskshe.exe." The service and the values added below serve as the running and persistence mechanism of the ransomware.

```
-----
69 00 6E 00 73 00 2E 00 74 00 78 00 74 00 00 00 00 00 00 00
HKLM\SYSTEM\CurrentControlSet\Services\gcpcgjbjkayp350\Type: 0x00000010
HKLM\SYSTEM\CurrentControlSet\Services\gcpcgjbjkayp350\Start: 0x00000002
HKLM\SYSTEM\CurrentControlSet\Services\gcpcgjbjkayp350\ErrorControl: 0x00000001
HKLM\SYSTEM\CurrentControlSet\Services\gcpcgjbjkayp350\ImagePath: "cmd.exe /c "C:\ProgramData\gcpcgjbjkayp350\taskshe.exe""
HKLM\SYSTEM\CurrentControlSet\Services\gcpcgjbjkayp350\DisplayName: "gcpcgjbjkayp350"
HKLM\SYSTEM\CurrentControlSet\Services\gcpcgjbjkayp350\WOW64: 0x0000014C
HKLM\SYSTEM\CurrentControlSet\Services\gcpcgjbjkayp350\ObjectName: "LocalSystem"
HKLM\SYSTEM\CurrentControlSet\Services\mssecsvc2.0\Type: 0x00000010
HKLM\SYSTEM\CurrentControlSet\Services\mssecsvc2.0\Start: 0x00000002
HKLM\SYSTEM\CurrentControlSet\Services\mssecsvc2.0\ErrorControl: 0x00000001
HKLM\SYSTEM\CurrentControlSet\Services\mssecsvc2.0\ImagePath: "C:\Users\Malware\Desktop\Ransomware.wannacry.exe -m security"
HKLM\SYSTEM\CurrentControlSet\Services\mssecsvc2.0\DisplayName: "Microsoft Security Center (2.0) Service"
HKLM\SYSTEM\CurrentControlSet\Services\mssecsvc2.0\WOW64: 0x0000014C
HKLM\SYSTEM\CurrentControlSet\Services\mssecsvc2.0\ObjectName: "LocalSystem"
HKLM\SYSTEM\CurrentControlSet\Services\mssecsvc2.0\FailureActions: 00 00 00 00 01 00 00 00 01 00 00 00 01 00 00 00 14 00 00 00 01 00 00 00 60 EA 00 00
HKU\S-1-5-21-497346990-3591733918-2170752703-1001\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\FeatureUsage\AppSwitched\C:\Users\Malware\AppData\
-----
Figure 19: Values Added
```


Advanced Static Analysis

Let's proceed with the advanced malware analysis using Cutter, a powerful tool that safely disassembles and decompiles executables. With Cutter's capabilities, we can gain detailed insights into the malware's execution, helping us understand its behavior and uncover how it operates.

The first screen we encounter is the dashboard, providing an overview of our malware sample. Here, we gather essential information about its format, class, and type. Additionally, we gain insights into its hashes and receive brief analysis details. This valuable information sets the stage for our in-depth malware analysis.

The screenshot displays the Cutter dashboard interface. At the top, the 'OVERVIEW' section is active, showing a grid of metadata for a file located at 'C:\Users\Malware\Desktop\Ransom'. The file is a 3.55 MB PE32 executable (EXEC) in x86 architecture, compiled on November 20, 2010. Below this, there are tabs for 'Certificates' and 'Version info'. The 'Hashes' section lists MD5, SHA1, SHA256, CRC32, and Entropy values. The 'Libraries' section lists loaded DLLs including kernel32.dll, advapi32.dll, ws2_32.dll, msvcrt.dll, iphlapi.dll, wininet.dll, and msvcrt.dll. The 'Analysis info' section at the bottom provides statistics on functions (83), X-refs (1831), calls (1707), strings (57007), symbols (91), imports (91), analysis coverage (33979 bytes), code size (36864 bytes), and coverage percent (92%).

| OVERVIEW | | | |
|-----------|---------------------------------|---------------|------------------------------|
| Info | | | |
| File: | C:\Users\Malware\Desktop\Ransom | FD: | 3 |
| Format: | pe | Base addr: | 0x00400000 |
| Bits: | 32 | Virtual addr: | True |
| Class: | PE32 | Canary: | False |
| Mode: | r-x | Crypto: | False |
| Size: | 3.55 MB | NX bit: | False |
| Type: | EXEC (Executable file) | PIC: | False |
| Language: | msvc | Static: | False |
| | | Relro: | N/A |
| | | Architecture: | x86 |
| | | Machine: | i386 |
| | | OS: | windows |
| | | Subsystem: | Windows GUI |
| | | Stripped: | False |
| | | Relocs: | True |
| | | Endianness: | LE |
| | | Compiled: | Sat Nov 20 09:03:08 2010 UTC |
| | | Compiler: | N/A |

| Hashes | |
|----------|--|
| MD5: | db349b97c37d22f5ea1d1841e3c89eb4 |
| SHA1: | e889544aff85ffa8b0d0da705105dee7c97fe26 |
| SHA256: | 24d004a104d4d54034dbcffc2a4b19a11f39008a575aa614ea04703480b1022c |
| CRC32: | 9fbb1227 |
| ENTROPY: | 7.964259 |

| Libraries | |
|--------------|--|
| kernel32.dll | |
| advapi32.dll | |
| ws2_32.dll | |
| msvcrt.dll | |
| iphlpapi.dll | |
| wininet.dll | |
| msvcrt.dll | |

| Analysis info | |
|--------------------|-------------|
| Functions: | 83 |
| X-Refs: | 1831 |
| Calls: | 1707 |
| Strings: | 57007 |
| Symbols: | 91 |
| Imports: | 91 |
| Analysis coverage: | 33979 bytes |
| Code size: | 36864 bytes |
| Coverage percent: | 92% |

Figure 20: Cutter Dashboard Screen

Upon analyzing the main function of the malware in assembly, we observe the manipulation of the strange URL, moved to the ESI register. Let's note that URL for later use in Advanced Dynamic Analysis. Subsequently, the program invokes the Windows APIs `InternetOpenA` and `InternetOpenUrlA`, utilizing the URL in the ESI register as one of the arguments. Should the URL be successfully reached it returns a bool value and proceeds to test EDI against itself and proceeds to the end of the program if the jump is not equal with the zero flag. This behavior suggests that the malware has a kill switch mechanism. If the URL is accessed, it terminates the program, preventing the execution of the ransomware, cleaning the stack and going to `ret 0x10` which finished the program. If the URL is reached but is nothing there, it then calls the function `fcn.00408090`.

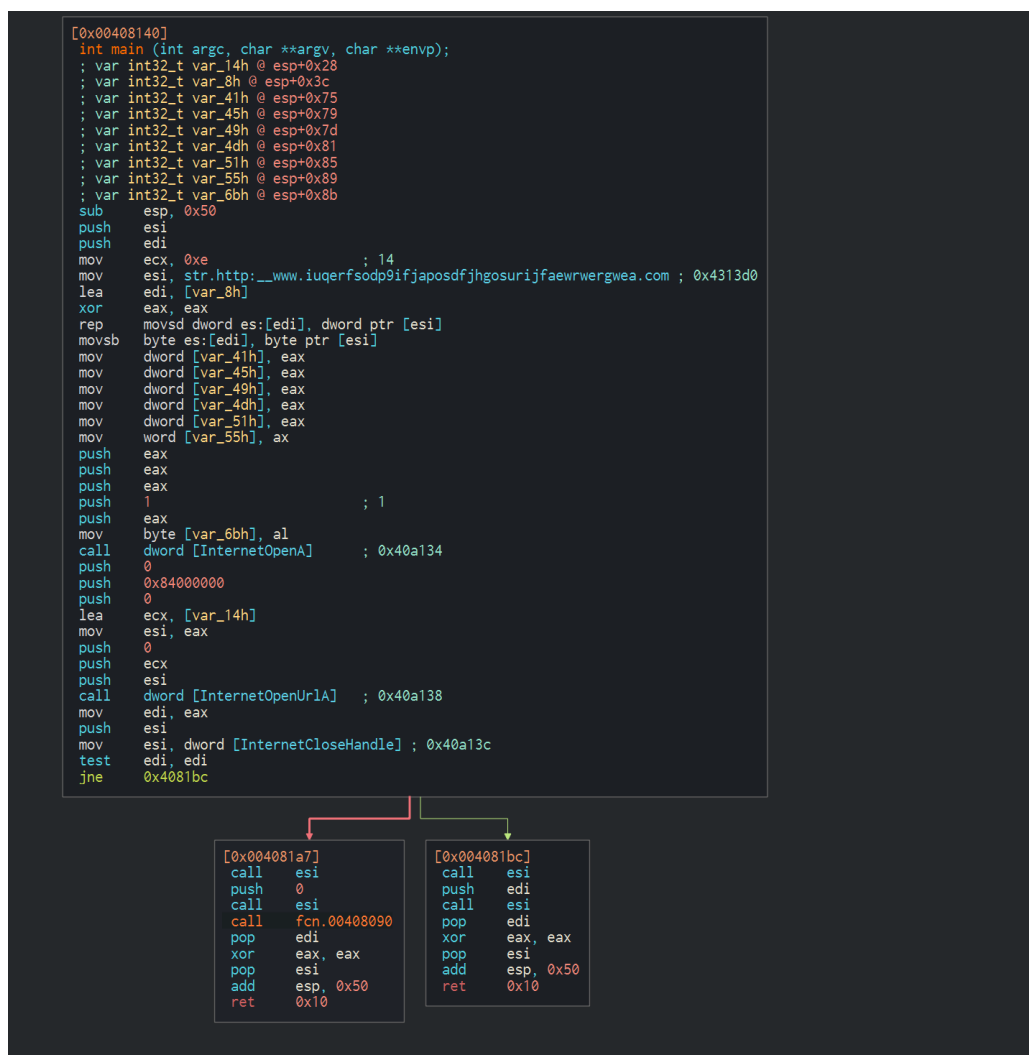


Figure 21: Main of ransomware

Once the malware is successfully executed, it initiates the crucial function call. In a nutshell, this call involves the opening of `SCManager` and `OpenServiceA`. Furthermore, a function call to `fcn.00407f20` is observed, following a conditional jump (`jge`). This behavior indicates the

malware's attempt to gain control and execute its payload, warranting a closer examination of the involved functions.



Figure 22: Call fcn.00408090

The call to fcn.00407f20 leads us to a function that invokes two other functions.

```
[0x00407f20]
fcn.00407f20 ();
call     fcn.00407c40
call     fcn.00407ce0
xor      eax, eax
ret
```

Figure 23: 0040f020 Fuction

The initial function call, fcn.004078c40, is responsible for creating a service with specific characteristics and subsequently starting the service.



Figure 24: First Call

The second call, fcn.00407ce0, represents the ransomware's core payload. This critical function is responsible for orchestrating multiple API calls, including LoadingResources and moveFileExA. These operations suggest that the malware engages in resource loading and file manipulation, which are characteristic behaviors of encryption routines. It's highly likely that this function encrypts files on the system, rendering them inaccessible without the decryption key.

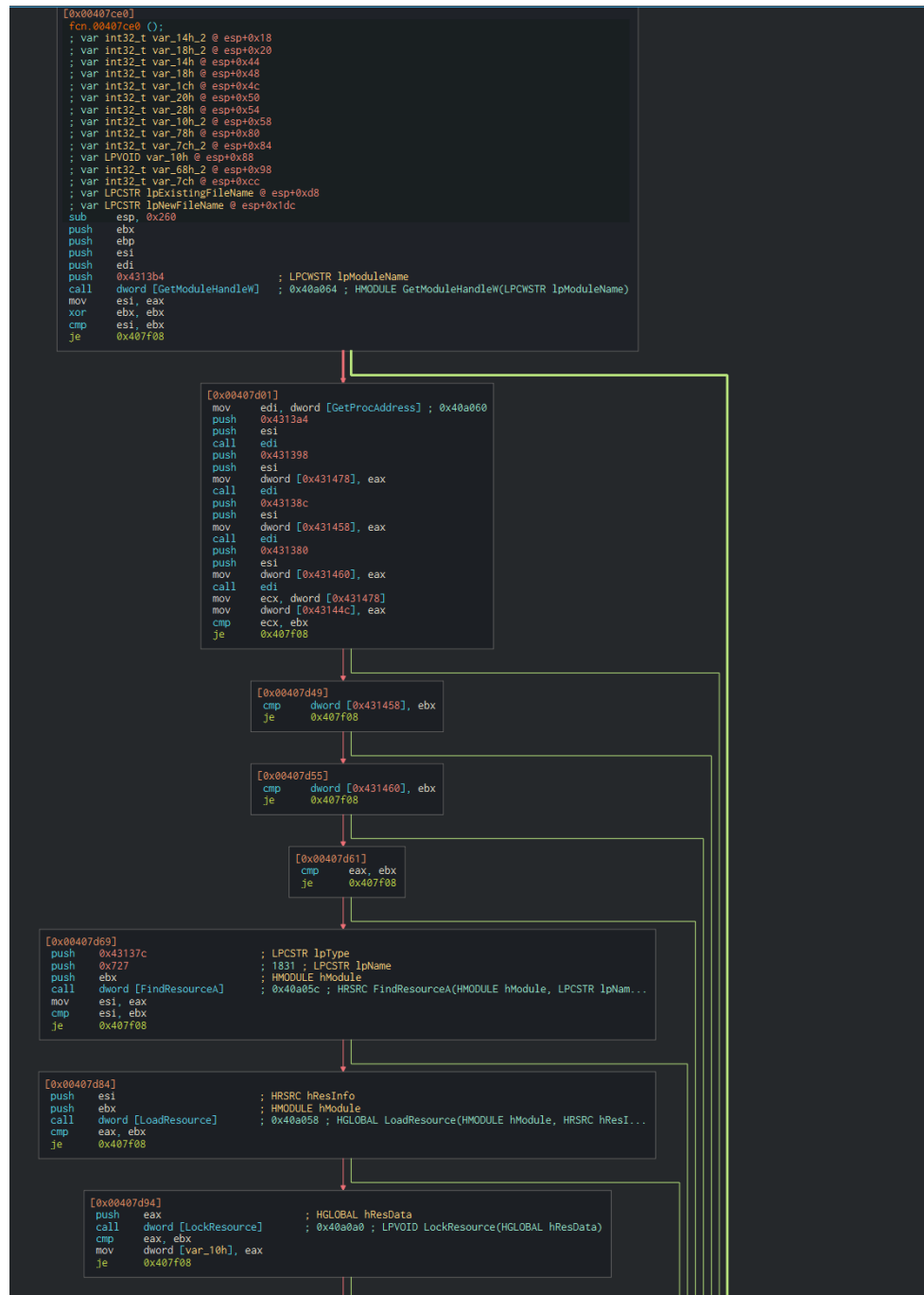


Figure 25: Payload

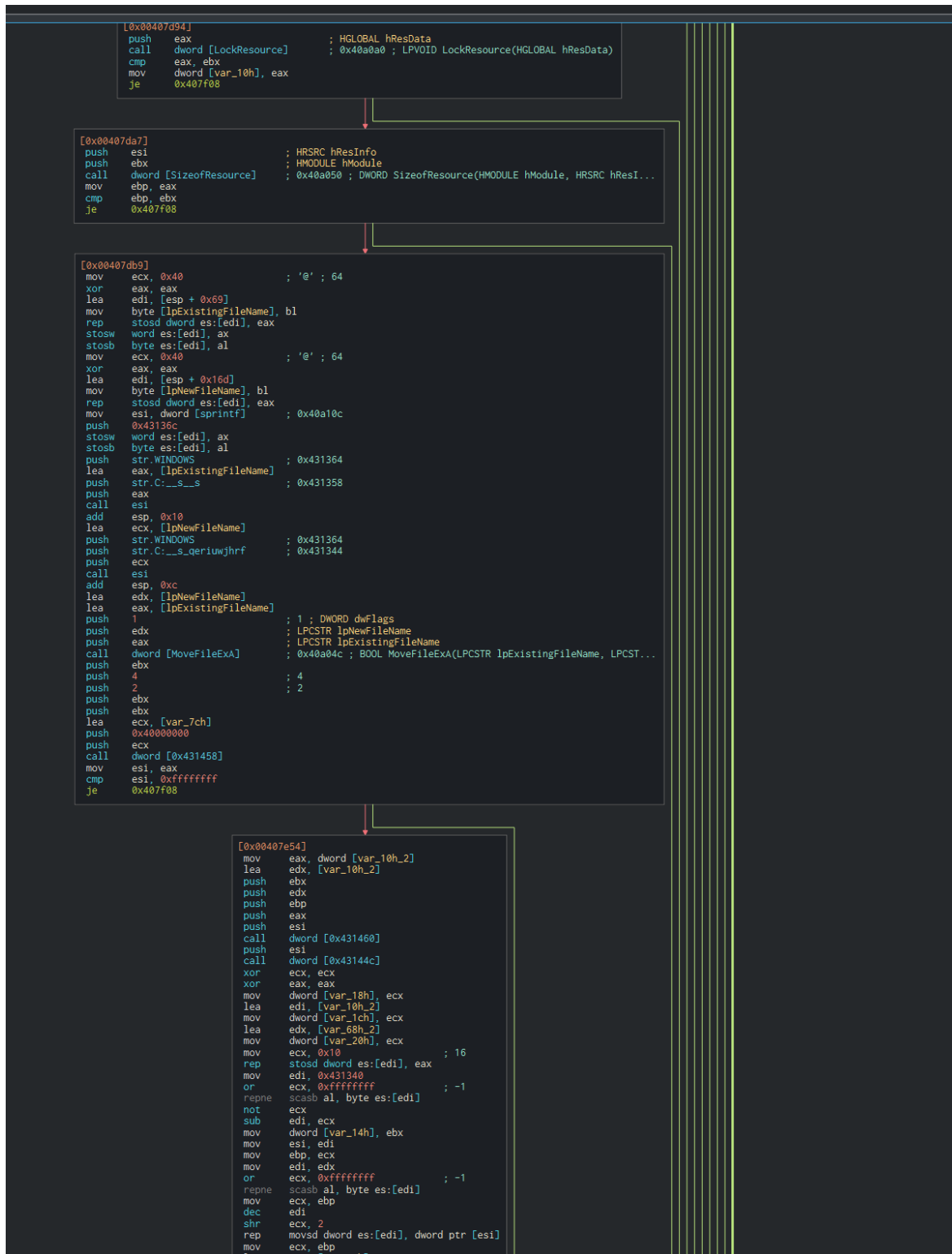


Figure 26:Payload

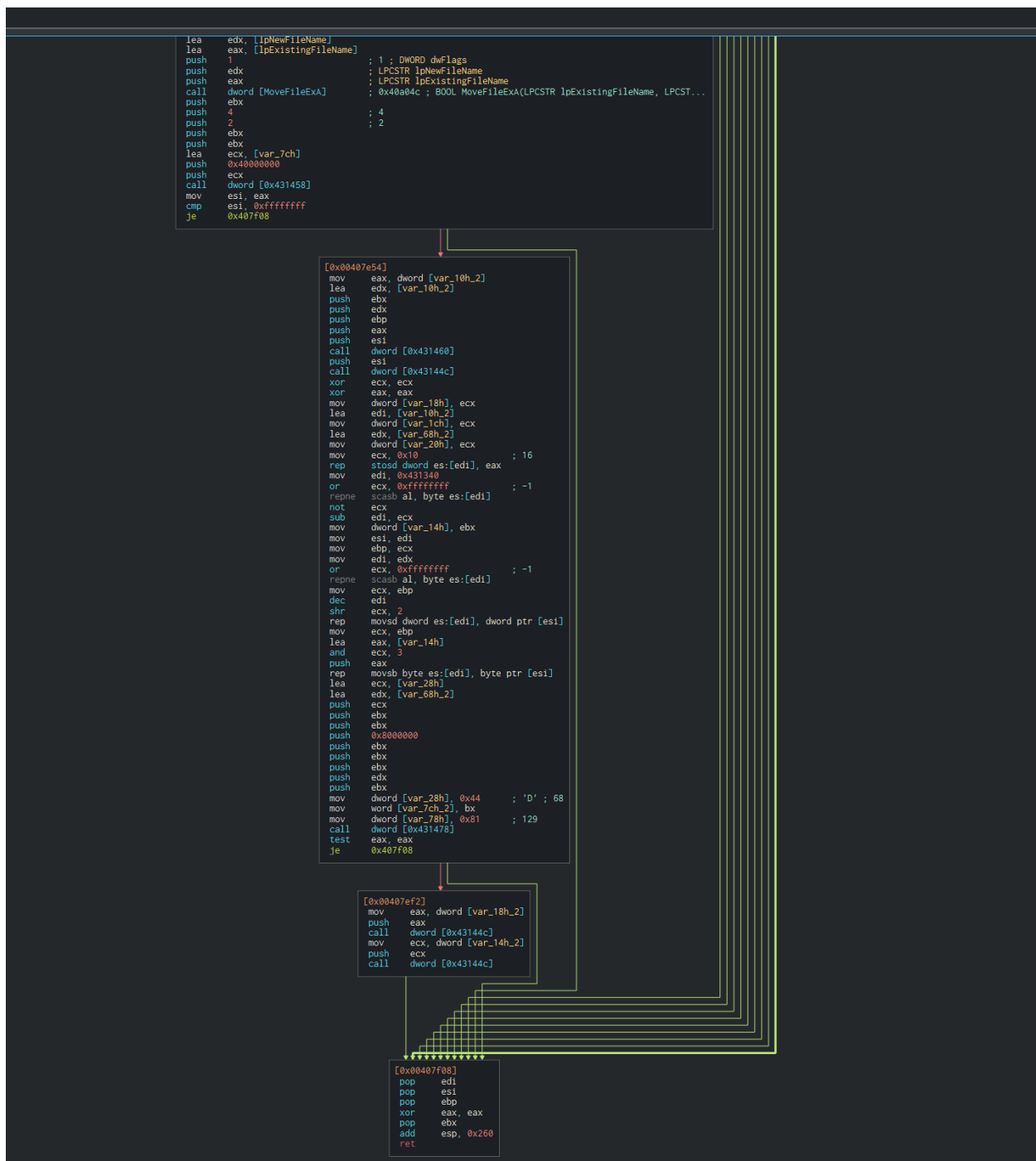


Figure 27: Payload

Advanced Dynamic Analysis

Proceeding with advanced dynamic analysis using debuggers requires utmost caution, as it involves running the program directly on the CPU within the system. This method offers real-time insights into the malware's behavior and interactions with the environment. However, due to its direct execution, there is a risk of unintended consequences and potential system impact. Engaging in controlled environments and employing virtualization is crucial to mitigate risks and maintain a safe testing environment during dynamic analysis.

In this Phase we will use x32dbg.

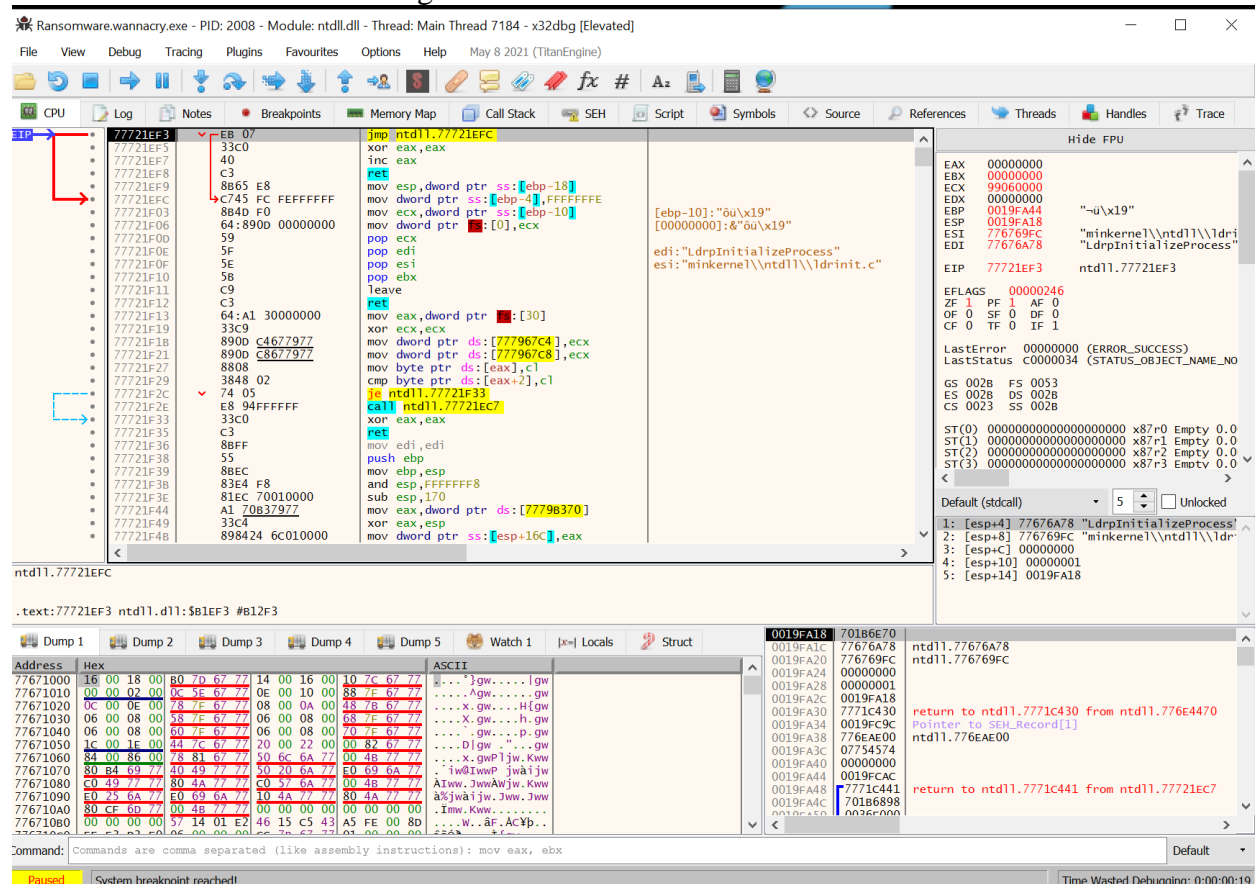


Figure 28: x32dbg

We will press F9 once to jump to the entry point. Previously, we identified the strange URL, and we are aware that there's a comparison (test) near it, determining the jump for the malware's kill switch. To locate this comparison, we will conduct a thorough search in all modules for the string reference of the strange URL. Once found, we will strategically place a breakpoint to

examine the malware's behavior at that critical point in the code.

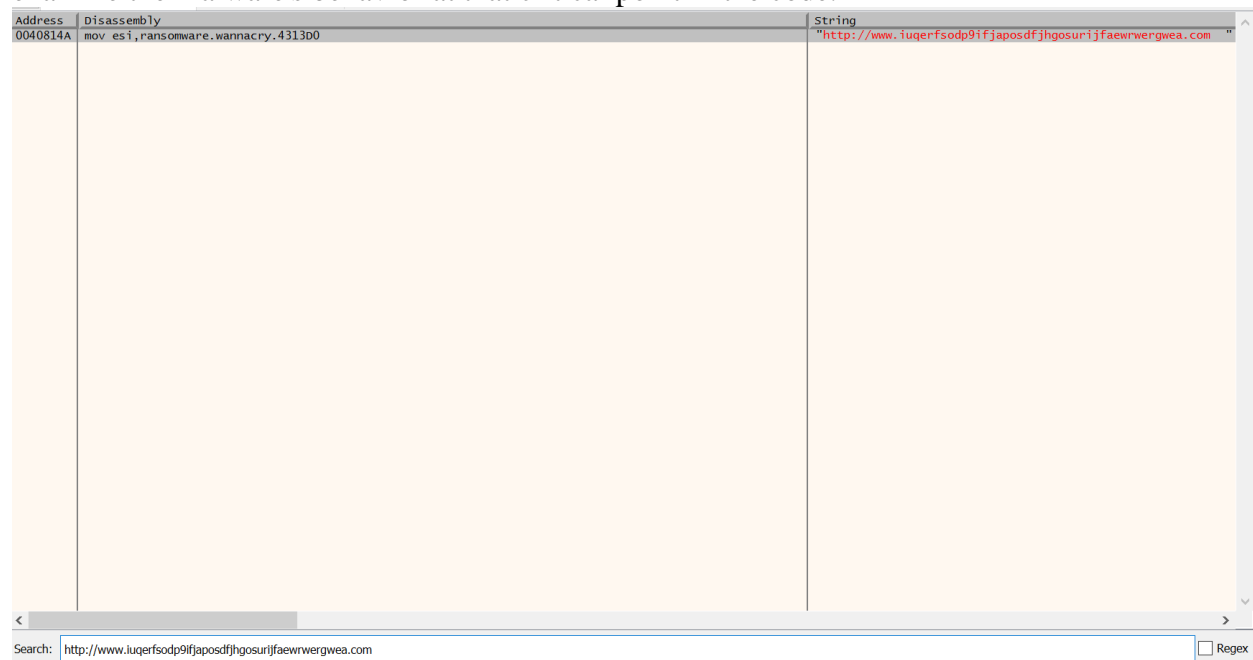


Figure 29: Breaking Point for Enigmatic URL

Upon pressing F9 to execute the program, we will reach the breakpoint we previously set. This strategic breakpoint allows us to pause the execution at a critical moment, enabling us to inspect the malware's behavior and gather valuable insights for further analysis.

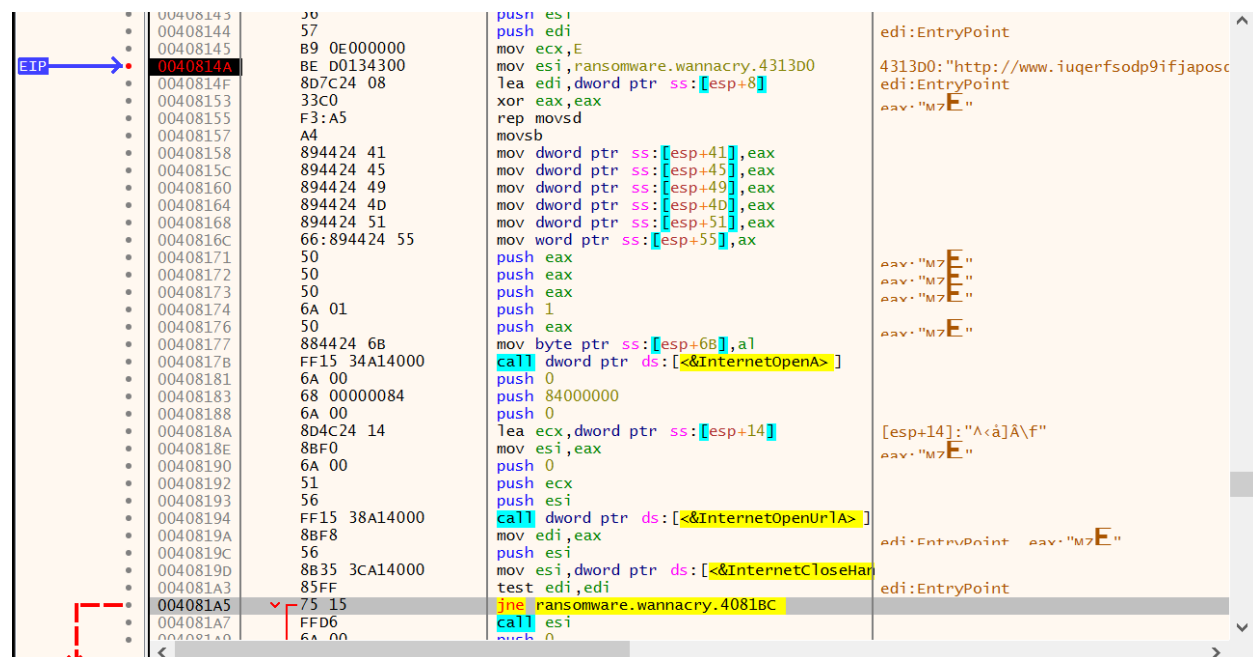


Figure 30: Finding the sweet spot

In the above string, we recognize a familiar sequence of code that we previously encountered in the disassembler. This section of the program employs the InternetOpenA API and

InternetOpenUrlA. Now, we will run the program until we reach the "test edi,edi" assembly instruction.

Upon inspection, we observe that the Zero Flag (ZF) is set to 1, indicating that the malware did not reach the strange URL, and it is prepared to detonate. If we were to change the ZF to 0, the malware would not execute, as the killswitch mechanism would activate, preventing its further progression.

```

EIP 004081A5 ransomware.wanr
EFLAGS 00000344
ZF 1 PF 1 AF 0
OF 0 SF 0 DF 0
CF 0 TF 1 IF 1

```

Figure 31: If network is not Enabled - ZF 1

```

EFLAGS 00000304
ZF 0 PF 1 AF 0
OF 0 SF 0 DF 0
CF 0 TF 1 IF 1

```

Figure 32: If network is enabled and reaches out to the URL ZF 0

| | | | | |
|-------|----------|-------------|---------------------------------|----------------|
| • | 004081A7 | FFD6 | call esi | |
| • | 004081A9 | 6A 00 | push 0 | |
| • | 004081AB | FFD6 | call esi | |
| • | 004081AD | E8 DEFEFFFF | call ransomware.wannacry.408090 | |
| • | 004081B2 | 5F | pop edi | edi:EntryPoint |
| • | 004081B3 | 33C0 | xor eax,eax | |
| • | 004081B5 | 5E | pop esi | |
| • | 004081B6 | 83C4 50 | add esp,50 | |
| • | 004081B9 | C2 1000 | ret 10 | |
| • | 004081BC | FFD6 | call esi | |
| • | 004081BE | 57 | push edi | edi:EntryPoint |
| • | 004081BF | FFD6 | call esi | |
| • | 004081C1 | 5F | pop edi | edi:EntryPoint |
| • | 004081C2 | 33C0 | xor eax,eax | |
| • | 004081C4 | 5E | pop esi | |
| • | 004081C5 | 83C4 50 | add esp,50 | |
| EIP → | 004081C8 | C2 1000 | ret 10 | |
| • | 004081CB | 90 | nop | |
| • | 004081CC | 90 | nop | |
| • | 004081CD | 90 | nop | |
| • | 004081CE | 90 | nop | |

Figure 33: With ZF 0 that we changed earlier, the program finishes

Indicators of Compromise

The full list of IOCs can be found in the Appendices.

Network Indicators

- Detonation with Network:

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|-------------|----------|-------------|----------|--------|--|
| 1 | 0.009999999 | 10.0.0.3 | 10.0.0.1 | TCP | 74 | 32388 → 53 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1413818908 TSecr=0 WS=128 |
| 2 | 0.039586964 | 10.0.0.3 | 10.0.0.1 | TCP | 74 | [TCP Retransmission] [TCP Port numbers reused] 42348 → 53 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1413818908 TSecr=0 WS=128 |
| 3 | 2.916025487 | 10.0.0.2 | 10.0.0.3 | DNS | 109 | Standard query 0x6a45 A www.iugersfodp91fjaposdfjhqosurijfaewrwerwgea.com |
| 4 | 2.919769921 | 10.0.0.3 | 10.0.0.2 | DNS | 125 | Standard query response 0x6a45 A www.iugersfodp91fjaposdfjhqosurijfaewrwerwgea.com A 10.0.0.3 |
| 5 | 2.927202599 | 10.0.0.2 | 10.0.0.3 | TCP | 66 | 50706 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 6 | 2.927217028 | 10.0.0.3 | 10.0.0.2 | TCP | 66 | 80 → 50706 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128 |
| 7 | 2.927356876 | 10.0.0.2 | 10.0.0.3 | TCP | 66 | 50706 → 80 [ACK] Seq=1 Ack=1 Win=262144 Len=0 |
| 8 | 2.928071422 | 10.0.0.2 | 10.0.0.3 | HTTP | 154 | GET / HTTP/1.1 |
| 9 | 2.928077812 | 10.0.0.3 | 10.0.0.2 | TCP | 54 | 80 → 50706 [ACK] Seq=1 Ack=101 Win=64256 Len=0 |
| 10 | 2.938554513 | 10.0.0.3 | 10.0.0.2 | TCP | 204 | 80 → 50706 [PSH, ACK] Seq=1 Ack=101 Win=64256 Len=150 [TCP segment of a reassembled PDU] |
| 11 | 2.938759211 | 10.0.0.2 | 10.0.0.3 | TCP | 66 | 50706 → 80 [ACK] Seq=101 Ack=151 Win=261888 Len=0 |
| 12 | 2.938765960 | 10.0.0.3 | 10.0.0.2 | HTTP | 312 | HTTP/1.1 200 OK (text/html) |
| 13 | 2.938878903 | 10.0.0.2 | 10.0.0.3 | TCP | 66 | 50706 → 80 [ACK] Seq=101 Ack=409 Win=261632 Len=0 |
| 14 | 2.938974807 | 10.0.0.2 | 10.0.0.3 | TCP | 66 | 50706 → 80 [FIN, ACK] Seq=101 Ack=409 Win=261632 Len=0 |

Figure 34: Enigmatic URL 200 OK from inetsim

| Time o... | Process Name | PID | Operation | Path | Result | Detail |
|------------|-----------------|------|----------------|-------------------------------------|---------|------------------------|
| 08:34:0... | Ransomware.w... | 7828 | TCP Connect | DESKTOP-UCICOFs.localdomain:5070... | SUCCESS | Length: 0, mss: 14... |
| 08:34:0... | Ransomware.w... | 7828 | TCP Send | DESKTOP-UCICOFs.localdomain:5070... | SUCCESS | Length: 100, starti... |
| 08:34:0... | Ransomware.w... | 7828 | TCP Receive | DESKTOP-UCICOFs.localdomain:5070... | SUCCESS | Length: 150, sequ... |
| 08:34:0... | Ransomware.w... | 7828 | TCP Receive | DESKTOP-UCICOFs.localdomain:5070... | SUCCESS | Length: 258, sequ... |
| 08:34:0... | Ransomware.w... | 7828 | TCP Disconnect | DESKTOP-UCICOFs.localdomain:5070... | SUCCESS | Length: 0, sequ... |

Figure 35: ProcMon Connections

- Detonation Without Network:

| Process Name | PID | Operation | Path | Result | Detail |
|------------------------|------|-----------|----------|----------|-----------------|
| Ransomware.wannacry... | 7840 | TCP | Syn Sent | 10.0.0.2 | 50844 10.0.0.80 |
| Ransomware.wannacry... | 7840 | TCP | Syn Sent | 10.0.0.2 | 50846 10.0.0.87 |
| Ransomware.wannacry... | 7840 | TCP | Syn Sent | 10.0.0.2 | 50848 10.0.0.88 |
| Ransomware.wannacry... | 7840 | TCP | Syn Sent | 10.0.0.2 | 50849 10.0.0.89 |
| Ransomware.wannacry... | 7840 | TCP | Syn Sent | 10.0.0.2 | 50851 10.0.0.90 |
| Ransomware.wannacry... | 7840 | TCP | Syn Sent | 10.0.0.2 | 50852 10.0.0.91 |
| Ransomware.wannacry... | 7840 | TCP | Syn Sent | 10.0.0.2 | 50854 10.0.0.92 |
| Ransomware.wannacry... | 7840 | TCP | Syn Sent | 10.0.0.2 | 50855 10.0.0.93 |
| Ransomware.wannacry... | 7840 | TCP | Syn Sent | 10.0.0.2 | 50856 10.0.0.94 |
| Ransomware.wannacry... | 7840 | TCP | Syn Sent | 10.0.0.2 | 50857 10.0.0.95 |

Figure 36: Worm Attributes - SMB

| | | | | | | | | |
|---------------------|------|-----|-------------|-----------|-----------------|-------|---------------------|---------------------|
| tasksvcs.exe | 1684 | TCP | Established | 127.0.0.1 | 61495 127.0.0.1 | 61496 | 04/08/2023 09:24:57 | tasksvcs.exe |
| tasksvcs.exe | 1684 | TCP | Established | 127.0.0.1 | 61496 127.0.0.1 | 61495 | 04/08/2023 09:24:57 | tasksvcs.exe |
| tasksvcs.exe | 1684 | TCP | Established | 127.0.0.1 | 9050 127.0.0.1 | 21709 | 04/08/2023 09:28:37 | tasksvcs.exe |
| @WanaDecryptor@.exe | 2716 | TCP | Established | 127.0.0.1 | 21709 127.0.0.1 | 9050 | 04/08/2023 09:28:37 | @WanaDecryptor@.exe |
| System | 4 | TCP | Listen | 10.0.0.2 | 139 0.0.0.0 | 0 | 31/07/2023 13:13:39 | System |

Figure 37: TCPView Wannacry

Host-based Indicators

| | | | | | |
|---------|-----------------|------|----------------|---------|-----------------------|
| 21.2... | Ransomware.w... | 7840 | Thread Create | SUCCESS | Thread ID: 7564 |
| 21.2... | Ransomware.w... | 7840 | Thread Create | SUCCESS | Thread ID: 4936 |
| 21.2... | Ransomware.w... | 5692 | Process Create | SUCCESS | PID: 5772, Comm... |
| 21.2... | Ransomware.w... | 7840 | Thread Create | SUCCESS | Thread ID: 512 |
| 21.2... | Ransomware.w... | 5692 | Thread Exit | SUCCESS | Thread ID: 6024, U... |

Figure 38: Creation of taskshe.exe

| | | | |
|--|----------------------|------------------------|------------------------|
| Ransomware.wannacry.exe (7i Microsoft® Disk D... | C:\Users\Malwarel... | Microsoft Corporati... | NT AUTHORITY\... |
| cmd.exe (6536) | Windows Commant... | C:\Windows\sysste... | Microsoft Corporati... |
| tasksche.exe (1128) | DiskPart | C:\ProgramData\g... | Microsoft Corporati... |
| attrib.exe (9176) | Attribute Utility | C:\Windows\SysW... | Microsoft Corporati... |
| Conhost.exe (6968) | Console Window H... | C:\Windows\Syste... | Microsoft Corporati... |

Figure 39: Process Tree

This PC > Local Disk (C:) > ProgramData > gpcgcbjkayp350

Search: gpcgcbjkayp350

| Name | Date modified | Type | Size |
|----------------------|------------------|---------------|----------|
| msg | 04/08/2023 09:22 | File folder | |
| TaskData | 04/08/2023 09:24 | File folder | |
| @Please_Read_Me@.txt | 04/08/2023 09:21 | Text Document | 1 KB |
| @WanaDecryptor@.exe | 12/05/2017 02:22 | Application | 240 KB |
| @WanaDecryptor@.exe | 04/08/2023 09:21 | Shortcut | 1 KB |
| 00000000.eky | 04/08/2023 09:21 | EKY File | 0 KB |
| 00000000.pky | 04/08/2023 09:21 | PKY File | 1 KB |
| 00000000.res | 04/08/2023 09:49 | RES File | 1 KB |
| b.wnry | 11/05/2017 20:13 | WNRY File | 1,407 KB |
| c.wnry | 04/08/2023 09:25 | WNRY File | 1 KB |
| f.wnry | 04/08/2023 09:22 | WNRY File | 1 KB |
| r.wnry | 11/05/2017 15:59 | WNRY File | 1 KB |
| s.wnry | 09/05/2017 16:58 | WNRY File | 2,968 KB |
| t.wnry | 12/05/2017 02:22 | WNRY File | 65 KB |
| taskdl.exe | 12/05/2017 02:22 | Application | 20 KB |
| tasksche.exe | 04/08/2023 09:21 | Application | 3,432 KB |
| taskse.exe | 12/05/2017 02:22 | Application | 20 KB |
| u.wnry | 12/05/2017 02:22 | WNRY File | 240 KB |

Figure 40: Directory of Ransomware

| | | | | | |
|-------------------------|--------|-----------|-------------|------|--|
| System Idle Process | 99.62 | 60 K | 8 K | 0 | |
| procxp64.exe | 0.57 | 34,844 K | 38,068 K | 7092 | Sysinternals Process Explorer Sysinternals - www.sysinter... |
| Memory Compression | 0.38 | 4,088 K | 1,861,924 K | 2000 | |
| System | < 0.01 | 196 K | 32 K | 4 | |
| Ransomware.wannacry.exe | < 0.01 | 24,948 K | 13,552 K | 7840 | Microsoft® Disk Defragmenter Microsoft Corporation |
| explorer.exe | < 0.01 | 90,352 K | 98,560 K | 3032 | Windows Explorer Microsoft Corporation |
| @WanaDecryptor@.exe | < 0.01 | 2,440 K | 12,772 K | 3132 | Load PerfMon Counters Microsoft Corporation |
| Interrupts | < 0.01 | 0 K | 0 K | n/a | Hardware Interrupts and DPCs |
| dwm.exe | < 0.01 | 216,776 K | 48,160 K | 500 | Desktop Window Manager Microsoft Corporation |
| Procmon64.exe | < 0.01 | 137,264 K | 122,816 K | 1264 | Process Monitor Sysinternals - www.sysinter... |
| conhost.exe | < 0.01 | 2,940 K | 7,424 K | 6224 | Console Window Host Microsoft Corporation |
| ConEmuC64.exe | < 0.01 | 5,308 K | 6,264 K | 3564 | ConEmu console extender (x... ConEmu-Maximus5 |
| ConEmu64.exe | < 0.01 | 37,068 K | 18,288 K | 812 | Console Emulator (x64) ConEmu-Maximus5 |
| csrss.exe | < 0.01 | 23,440 K | 33,652 K | 584 | |
| vmtoolsd.exe | < 0.01 | 33,124 K | 38,088 K | 6944 | VMware Tools Core Service VMware, Inc. |
| svchost.exe | < 0.01 | 9,276 K | 12,432 K | 988 | Host Process for Windows S... Microsoft Corporation |
| taskhsvc.exe | < 0.01 | 6,988 K | 12,216 K | 1684 | |
| vmtoolsd.exe | < 0.01 | 7,380 K | 10,068 K | 2960 | VMware Tools Core Service VMware, Inc. |
| svchost.exe | < 0.01 | 8,276 K | 19,544 K | 1616 | Host Process for Windows S... Microsoft Corporation |
| svchost.exe | < 0.01 | 11,132 K | 17,560 K | 864 | Host Process for Windows S... Microsoft Corporation |
| tasksche.exe | < 0.01 | 7,572 K | 13,480 K | 1128 | DiskPart Microsoft Corporation |
| svchost.exe | < 0.01 | 2,620 K | 4,536 K | 3580 | Host Process for Windows S... Microsoft Corporation |
| SearchIndexer.exe | < 0.01 | 32,464 K | 29,648 K | 5904 | Microsoft Windows Search In... Microsoft Corporation |
| RuntimeBroker.exe | < 0.01 | 40,968 K | 44,706 K | 4460 | Runtime Broker Microsoft Corporation |

Figure 41: ProcExp of wannacry and taskshe.exe








| | | | | | |
|---|---------------------------------|------------------|---------|------------------|---------------|
|  | Google Update Service (gup... | Keeps your ... | | Automatic (De... | Local System |
|  | Google Chrome Elevation Se... | | | Manual | Local System |
|  | Geolocation Service | This service ... | Running | Manual (Trigg... | Local System |
|  | gpcgcbjkayp350 | | | Automatic | Local System |
|  | GameDVR and Broadcast Us... | This user ser... | | Manual | Local System |
|  | Function Discovery Resourc... | Publishes thi... | Running | Manual (Trigg... | Local Service |
|  | Function Discovery Provider ... | The FDPHOS... | Running | Manual | Local Service |

Figure 42: Service for persistence

Cuckoo Analysis & Yara RULE

Cuckoo Sandbox is an open-source automated malware analysis system. It allows security researchers to execute and analyze suspicious files in a controlled environment. Cuckoo Sandbox provides valuable insights into malware behavior, helping identify potential threats and enhance cybersecurity defenses.

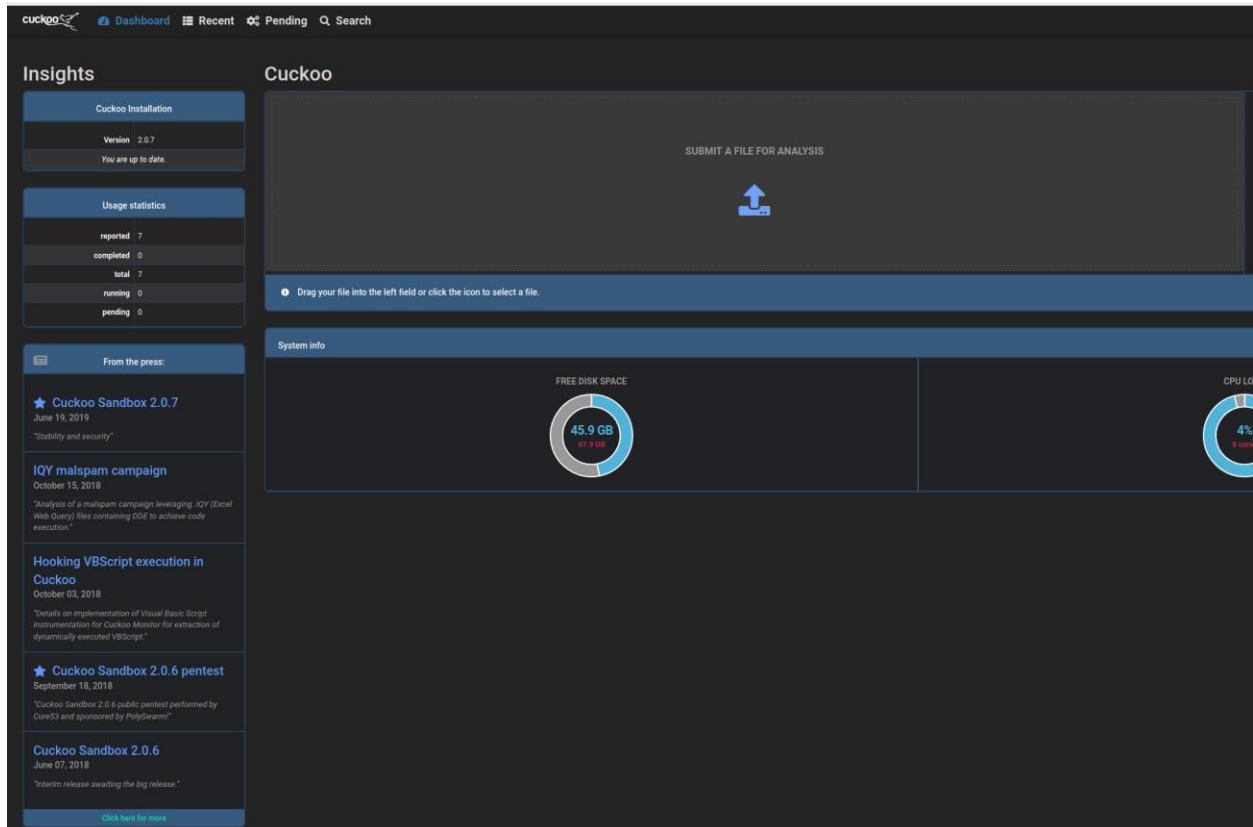


Figure 43: Home Screen Of Cuckoo Sandbox

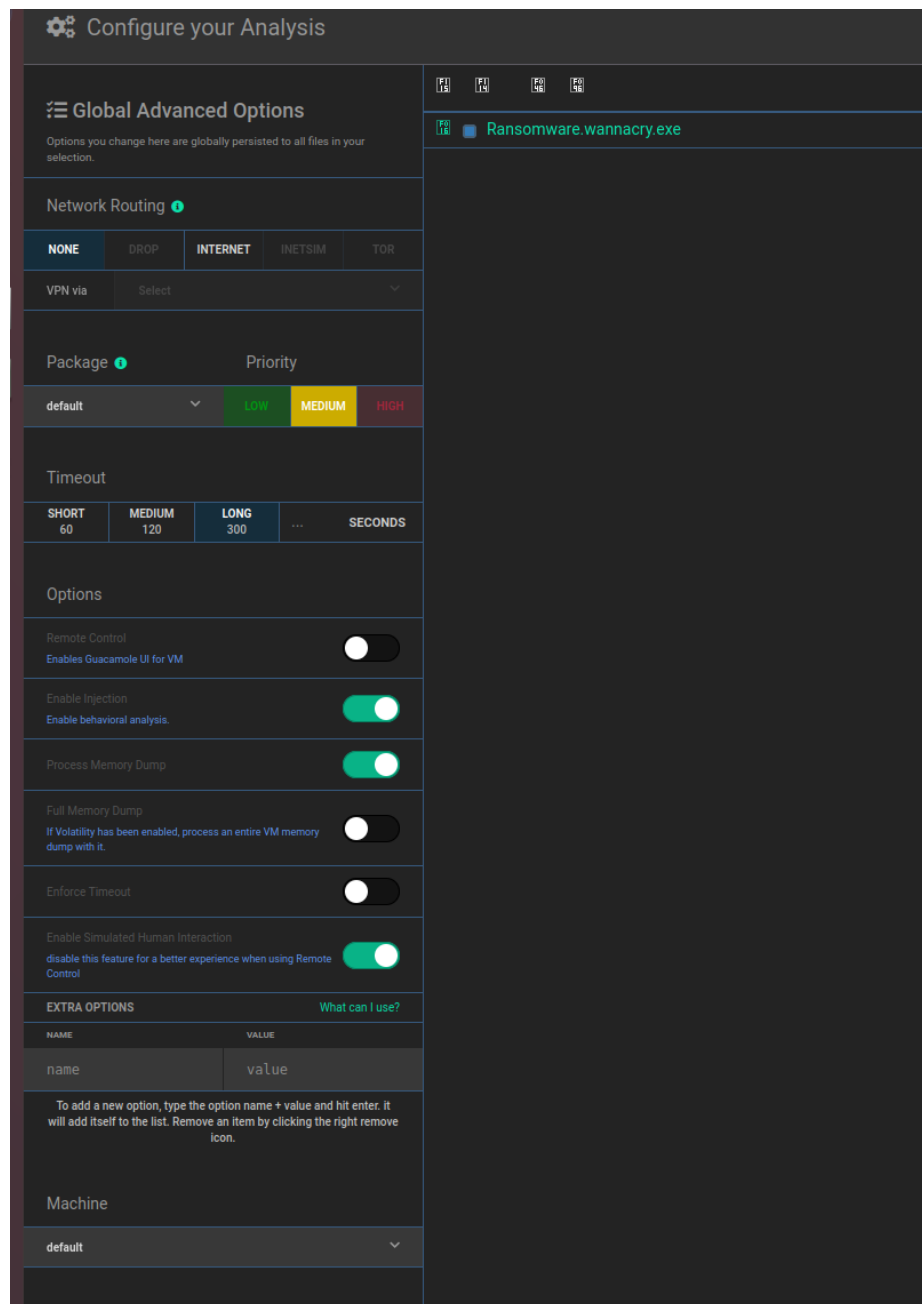


Figure 44: Submitting Wannacry without internet to Cuckoo

Putting our malware into Cuckoo Sandbox involves submitting the suspicious file to the platform for automated analysis. Cuckoo will execute the malware in a controlled environment, monitor its behavior, and generate detailed reports on its actions. This process helps us gain valuable insights into the malware's capabilities and aids in devising effective countermeasures to protect against similar threats.

Summary

File: Ransomware.wannacry.exe

Size: 3.6MB

Type: PE32 executable (GUI) Intel 80386, for MS Windows

MD5: db349b97c37d22f5ea1d1841e3c99eb4

SHA1: e88954a3ff85ffaf8b8b8d9785185dee7c37fe26

SHA256: 24f09461046f4d548340cfff2a4b19a11f39088d57a0e14e04703480b1022c

SHA512: [Show SHA512](#)

CRC32: 9f8b1227

ssdeep: None

Yara: None matched

Information on Execution

| Category | Started | Completed | Duration | Routing | Logs |
|----------|------------------------|------------------------|------------|----------|--|
| FILE | Aug 4, 2023, 2:12 p.m. | Aug 4, 2023, 2:13 p.m. | 60 seconds | internet | Show Analysis Log Show Cuckoo Log |

Signatures

Figure 45: Cuckoo Summary

After analyzing the malware in Cuckoo, we are presented with a comprehensive summary screen containing essential information such as hashes and other details about the analyzed file. Additionally, Cuckoo assigns a score that provides an initial assessment of the file's threat level. In the tab view, we have a range of options to proceed with further analysis, allowing us to delve deeper into the malware's behavior, network interactions, and other critical aspects, aiding us in crafting effective mitigation strategies.

Static Analysis

Static Analysis | Strings | Antivirus | IDA

PE Compile Time: 2018-11-29 11:40:08

PE Imphash: [Show 1776a6b8f7b0d38d3f1c0d710d7124](#)

Version Info

LegalCopyright: Load Microsoft Corporation. All rights reserved.

InternalName: lndfrgui.exe

FileVersion: 6.1.7601.17514 (win7sp1_rtm.101119-1850)

CompanyName: Microsoft Corporation

ProductName: Microsoft Windows Operating System

ProductVersion: 6.1.7601.17514

FileDescription: Microsoft Windows Disk Defragmenter

OriginalFilename: lndfrgui.exe

Translation: 0x409 0x406

PEID Signatures

Armitage v1.71

Sections

| Name | Virtual Address | Virtual Size | Size of Raw Data |
|--------|-----------------|--------------|------------------|
| .text | 0x0001000 | 0x00000ca | 0x00007000 |
| .rdata | 0x000a000 | 0x0000098 | 0x00001000 |
| .data | 0x000a000 | 0x0000480 | 0x00007000 |
| .rsrc | 0x0010000 | 0x0003a14 | 0x00038000 |

Resources

| Name | Offset | Size | Language | Sub-language | File type |
|------------|-----------|-----------|--------------|--------------------|---|
| R | 0x0010004 | 0x0005a00 | LANG_ENGLISH | SUBLANG_ENGLISH_US | PE32 executable (GUI) Intel 80386, for MS Windows |
| RT_VERSION | 0x005a004 | 0x0000020 | LANG_ENGLISH | SUBLANG_ENGLISH_US | Data |

Imports

Library KERNEL32.dll

- 0x400010: WaitForSingleObject
- 0x400014: InterlockedIncrement
- 0x400018: SetCurrentThread
- 0x40001c: SetCurrentThread
- 0x400020: HeapSize
- 0x400024: GetFileTime
- 0x400028: HeapSize
- 0x40002c: HeapSize
- 0x400030: HeapSize
- 0x400034: HeapSize
- 0x400038: HeapSize
- 0x40003c: HeapSize
- 0x400040: HeapSize
- 0x400044: HeapSize
- 0x400048: HeapSize
- 0x40004c: HeapSize
- 0x400050: HeapSize
- 0x400054: HeapSize
- 0x400058: HeapSize
- 0x40005c: HeapSize
- 0x400060: HeapSize
- 0x400064: HeapSize
- 0x400068: HeapSize
- 0x40006c: HeapSize
- 0x400070: HeapSize
- 0x400074: HeapSize
- 0x400078: HeapSize
- 0x40007c: HeapSize
- 0x400080: HeapSize
- 0x400084: HeapSize
- 0x400088: HeapSize
- 0x40008c: HeapSize
- 0x400090: HeapSize
- 0x400094: HeapSize
- 0x400098: HeapSize
- 0x40009c: HeapSize
- 0x4000a0: HeapSize
- 0x4000a4: HeapSize
- 0x4000a8: HeapSize
- 0x4000ac: HeapSize
- 0x4000b0: HeapSize
- 0x4000b4: HeapSize
- 0x4000b8: HeapSize
- 0x4000bc: HeapSize
- 0x4000c0: HeapSize
- 0x4000c4: HeapSize
- 0x4000c8: HeapSize
- 0x4000cc: HeapSize
- 0x4000d0: HeapSize
- 0x4000d4: HeapSize
- 0x4000d8: HeapSize
- 0x4000dc: HeapSize
- 0x4000e0: HeapSize
- 0x4000e4: HeapSize
- 0x4000e8: HeapSize
- 0x4000ec: HeapSize
- 0x4000f0: HeapSize
- 0x4000f4: HeapSize
- 0x4000f8: HeapSize
- 0x4000fc: HeapSize
- 0x400100: HeapSize
- 0x400104: HeapSize
- 0x400108: HeapSize
- 0x40010c: HeapSize
- 0x400110: HeapSize
- 0x400114: HeapSize
- 0x400118: HeapSize
- 0x40011c: HeapSize
- 0x400120: HeapSize
- 0x400124: HeapSize
- 0x400128: HeapSize
- 0x40012c: HeapSize
- 0x400130: HeapSize
- 0x400134: HeapSize
- 0x400138: HeapSize
- 0x40013c: HeapSize
- 0x400140: HeapSize
- 0x400144: HeapSize
- 0x400148: HeapSize
- 0x40014c: HeapSize
- 0x400150: HeapSize
- 0x400154: HeapSize
- 0x400158: HeapSize
- 0x40015c: HeapSize
- 0x400160: HeapSize
- 0x400164: HeapSize
- 0x400168: HeapSize
- 0x40016c: HeapSize
- 0x400170: HeapSize
- 0x400174: HeapSize
- 0x400178: HeapSize
- 0x40017c: HeapSize
- 0x400180: HeapSize
- 0x400184: HeapSize
- 0x400188: HeapSize
- 0x40018c: HeapSize
- 0x400190: HeapSize
- 0x400194: HeapSize
- 0x400198: HeapSize
- 0x40019c: HeapSize
- 0x4001a0: HeapSize
- 0x4001a4: HeapSize
- 0x4001a8: HeapSize
- 0x4001ac: HeapSize
- 0x4001b0: HeapSize
- 0x4001b4: HeapSize
- 0x4001b8: HeapSize
- 0x4001bc: HeapSize
- 0x4001c0: HeapSize
- 0x4001c4: HeapSize
- 0x4001c8: HeapSize
- 0x4001cc: HeapSize
- 0x4001d0: HeapSize
- 0x4001d4: HeapSize
- 0x4001d8: HeapSize
- 0x4001dc: HeapSize
- 0x4001e0: HeapSize
- 0x4001e4: HeapSize
- 0x4001e8: HeapSize
- 0x4001ec: HeapSize
- 0x4001f0: HeapSize
- 0x4001f4: HeapSize
- 0x4001f8: HeapSize
- 0x4001fc: HeapSize
- 0x400200: HeapSize
- 0x400204: HeapSize
- 0x400208: HeapSize
- 0x40020c: HeapSize
- 0x400210: HeapSize
- 0x400214: HeapSize
- 0x400218: HeapSize
- 0x40021c: HeapSize
- 0x400220: HeapSize
- 0x400224: HeapSize
- 0x400228: HeapSize
- 0x40022c: HeapSize
- 0x400230: HeapSize
- 0x400234: HeapSize
- 0x400238: HeapSize
- 0x40023c: HeapSize
- 0x400240: HeapSize
- 0x400244: HeapSize
- 0x400248: HeapSize
- 0x40024c: HeapSize
- 0x400250: HeapSize
- 0x400254: HeapSize
- 0x400258: HeapSize
- 0x40025c: HeapSize
- 0x400260: HeapSize
- 0x400264: HeapSize
- 0x400268: HeapSize
- 0x40026c: HeapSize
- 0x400270: HeapSize
- 0x400274: HeapSize
- 0x400278: HeapSize
- 0x40027c: HeapSize
- 0x400280: HeapSize
- 0x400284: HeapSize
- 0x400288: HeapSize
- 0x40028c: HeapSize
- 0x400290: HeapSize
- 0x400294: HeapSize
- 0x400298: HeapSize
- 0x40029c: HeapSize
- 0x4002a0: HeapSize
- 0x4002a4: HeapSize
- 0x4002a8: HeapSize
- 0x4002ac: HeapSize
- 0x4002b0: HeapSize
- 0x4002b4: HeapSize
- 0x4002b8: HeapSize
- 0x4002bc: HeapSize
- 0x4002c0: HeapSize
- 0x4002c4: HeapSize
- 0x4002c8: HeapSize
- 0x4002cc: HeapSize
- 0x4002d0: HeapSize
- 0x4002d4: HeapSize
- 0x4002d8: HeapSize
- 0x4002dc: HeapSize
- 0x4002e0: HeapSize
- 0x4002e4: HeapSize
- 0x4002e8: HeapSize
- 0x4002ec: HeapSize
- 0x4002f0: HeapSize
- 0x4002f4: HeapSize
- 0x4002f8: HeapSize
- 0x4002fc: HeapSize
- 0x400300: HeapSize
- 0x400304: HeapSize
- 0x400308: HeapSize
- 0x40030c: HeapSize
- 0x400310: HeapSize
- 0x400314: HeapSize
- 0x400318: HeapSize
- 0x40031c: HeapSize
- 0x400320: HeapSize
- 0x400324: HeapSize
- 0x400328: HeapSize
- 0x40032c: HeapSize
- 0x400330: HeapSize
- 0x400334: HeapSize
- 0x400338: HeapSize
- 0x40033c: HeapSize
- 0x400340: HeapSize
- 0x400344: HeapSize
- 0x400348: HeapSize
- 0x40034c: HeapSize
- 0x400350: HeapSize
- 0x400354: HeapSize
- 0x400358: HeapSize
- 0x40035c: HeapSize
- 0x400360: HeapSize
- 0x400364: HeapSize
- 0x400368: HeapSize
- 0x40036c: HeapSize
- 0x400370: HeapSize
- 0x400374: HeapSize
- 0x400378: HeapSize
- 0x40037c: HeapSize
- 0x400380: HeapSize
- 0x400384: HeapSize
- 0x400388: HeapSize
- 0x40038c: HeapSize
- 0x400390: HeapSize
- 0x400394: HeapSize
- 0x400398: HeapSize
- 0x40039c: HeapSize
- 0x4003a0: HeapSize
- 0x4003a4: HeapSize
- 0x4003a8: HeapSize
- 0x4003ac: HeapSize
- 0x4003b0: HeapSize
- 0x4003b4: HeapSize
- 0x4003b8: HeapSize
- 0x4003bc: HeapSize
- 0x4003c0: HeapSize
- 0x4003c4: HeapSize
- 0x4003c8: HeapSize
- 0x4003cc: HeapSize
- 0x4003d0: HeapSize
- 0x4003d4: HeapSize
- 0x4003d8: HeapSize
- 0x4003dc: HeapSize
- 0x4003e0: HeapSize
- 0x4003e4: HeapSize
- 0x4003e8: HeapSize
- 0x4003ec: HeapSize
- 0x4003f0: HeapSize
- 0x4003f4: HeapSize
- 0x4003f8: HeapSize
- 0x4003fc: HeapSize
- 0x400400: HeapSize
- 0x400404: HeapSize
- 0x400408: HeapSize
- 0x40040c: HeapSize
- 0x400410: HeapSize
- 0x400414: HeapSize
- 0x400418: HeapSize
- 0x40041c: HeapSize
- 0x400420: HeapSize
- 0x400424: HeapSize
- 0x400428: HeapSize
- 0x40042c: HeapSize
- 0x400430: HeapSize
- 0x400434: HeapSize
- 0x400438: HeapSize
- 0x40043c: HeapSize
- 0x400440: HeapSize
- 0x400444: HeapSize
- 0x400448: HeapSize
- 0x40044c: HeapSize
- 0x400450: HeapSize
- 0x400454: HeapSize
- 0x400458: HeapSize
- 0x40045c: HeapSize
- 0x400460: HeapSize
- 0x400464: HeapSize
- 0x400468: HeapSize
- 0x40046c: HeapSize
- 0x400470: HeapSize
- 0x400474: HeapSize
- 0x400478: HeapSize
- 0x40047c: HeapSize
- 0x400480: HeapSize
- 0x400484: HeapSize
- 0x400488: HeapSize
- 0x40048c: HeapSize
- 0x400490: HeapSize
- 0x400494: HeapSize
- 0x400498: HeapSize
- 0x40049c: HeapSize
- 0x4004a0: HeapSize
- 0x4004a4: HeapSize
- 0x4004a8: HeapSize
- 0x4004ac: HeapSize
- 0x4004b0: HeapSize
- 0x4004b4: HeapSize
- 0x4004b8: HeapSize
- 0x4004bc: HeapSize
- 0x4004c0: HeapSize
- 0x4004c4: HeapSize
- 0x4004c8: HeapSize
- 0x4004cc: HeapSize
- 0x4004d0: HeapSize
- 0x4004d4: HeapSize
- 0x4004d8: HeapSize
- 0x4004dc: HeapSize
- 0x4004e0: HeapSize
- 0x4004e4: HeapSize
- 0x4004e8: HeapSize
- 0x4004ec: HeapSize
- 0x4004f0: HeapSize
- 0x4004f4: HeapSize
- 0x4004f8: HeapSize
- 0x4004fc: HeapSize
- 0x400500: HeapSize
- 0x400504: HeapSize
- 0x400508: HeapSize
- 0x40050c: HeapSize
- 0x400510: HeapSize
- 0x400514: HeapSize
- 0x400518: HeapSize
- 0x40051c: HeapSize
- 0x400520: HeapSize
- 0x400524: HeapSize
- 0x400528: HeapSize
- 0x40052c: HeapSize
- 0x400530: HeapSize
- 0x400534: HeapSize
- 0x400538: HeapSize
- 0x40053c: HeapSize
- 0x400540: HeapSize
- 0x400544: HeapSize
- 0x400548: HeapSize
- 0x40054c: HeapSize
- 0x400550: HeapSize
- 0x400554: HeapSize
- 0x400558: HeapSize
- 0x40055c: HeapSize
- 0x400560: HeapSize
- 0x400564: HeapSize
- 0x400568: HeapSize
- 0x40056c: HeapSize
- 0x400570: HeapSize
- 0x400574: HeapSize
- 0x400578: HeapSize
- 0x40057c: HeapSize
- 0x400580: HeapSize
- 0x400584: HeapSize
- 0x400588: HeapSize
- 0x40058c: HeapSize
- 0x400590: HeapSize
- 0x400594: HeapSize
- 0x400598: HeapSize
- 0x40059c: HeapSize
- 0x4005a0: HeapSize
- 0x4005a4: HeapSize
- 0x4005a8: HeapSize
- 0x4005ac: HeapSize
- 0x4005b0: HeapSize
- 0x4005b4: HeapSize
- 0x4005b8: HeapSize
- 0x4005bc: HeapSize
- 0x4005c0: HeapSize
- 0x4005c4: HeapSize
- 0x4005c8: HeapSize
- 0x4005cc: HeapSize
- 0x4005d0: HeapSize
- 0x4005d4: HeapSize
- 0x4005d8: HeapSize
- 0x4005dc: HeapSize
- 0x4005e0: HeapSize
- 0x4005e4: HeapSize
- 0x4005e8: HeapSize
- 0x4005ec: HeapSize
- 0x4005f0: HeapSize
- 0x4005f4: HeapSize
- 0x4005f8: HeapSize
- 0x4005fc: HeapSize
- 0x400600: HeapSize
- 0x400604: HeapSize
- 0x400608: HeapSize
- 0x40060c: HeapSize
- 0x400610: HeapSize
- 0x400614: HeapSize
- 0x400618: HeapSize
- 0x40061c: HeapSize
- 0x400620: HeapSize
- 0x400624: HeapSize
- 0x400628: HeapSize
- 0x40062c: HeapSize
- 0x400630: HeapSize
- 0x400634: HeapSize
- 0x400638: HeapSize
- 0x40063c: HeapSize
- 0x400640: HeapSize
- 0x400644: HeapSize
- 0x400648: HeapSize
- 0x40064c: HeapSize
- 0x400650: HeapSize
- 0x400654: HeapSize
- 0x400658: HeapSize
- 0x40065c: HeapSize
- 0x400660: HeapSize
- 0x400664: HeapSize
- 0x400668: HeapSize
- 0x40066c: HeapSize
- 0x400670: HeapSize
- 0x400674: HeapSize
- 0x400678: HeapSize
- 0x40067c: HeapSize
- 0x400680: HeapSize
- 0x400684: HeapSize
- 0x400688: HeapSize
- 0x40068c: HeapSize
- 0x400690: HeapSize
- 0x400694: HeapSize
- 0x400698: HeapSize
- 0x40069c: HeapSize
- 0x4006a0: HeapSize
- 0x4006a4: HeapSize
- 0x4006a8: HeapSize
- 0x4006ac: HeapSize
- 0x4006b0: HeapSize
- 0x4006b4: HeapSize
- 0x4006b8: HeapSize
- 0x4006bc: HeapSize
- 0x4006c0: HeapSize
- 0x4006c4: HeapSize
- 0x4006c8: HeapSize
- 0x4006cc: HeapSize
- 0x4006d0: HeapSize
- 0x4006d4: HeapSize
- 0x4006d8: HeapSize
- 0x4006dc: HeapSize
- 0x4006e0: HeapSize
- 0x4006e4: HeapSize
- 0x4006e8: HeapSize
- 0x4006ec: HeapSize
- 0x4006f0: HeapSize
- 0x4006f4: HeapSize
- 0x4006f8: HeapSize
- 0x4006fc: HeapSize
- 0x400700: HeapSize
- 0x400704: HeapSize
- 0x400708: HeapSize
- 0x40070c: HeapSize
- 0x400710: HeapSize
- 0x400714: HeapSize
- 0x400718: HeapSize
- 0x40071c: HeapSize
- 0x400720: HeapSize
- 0x400724: HeapSize
- 0x400728: HeapSize
- 0x40072c: HeapSize
- 0x400730: HeapSize
- 0x400734: HeapSize
- 0x400738: HeapSize
- 0x40073c: HeapSize
- 0x400740: HeapSize
- 0x400744: HeapSize
- 0x400748: HeapSize
- 0x40074c: HeapSize
- 0x400750: HeapSize
- 0x400754: HeapSize
- 0x400758: HeapSize
- 0x40075c: HeapSize
- 0x400760: HeapSize
- 0x400764: HeapSize
- 0x400768: HeapSize
- 0x40076c: HeapSize
- 0x400770: HeapSize
- 0x400774: HeapSize
- 0x400778: HeapSize
- 0x40077c: HeapSize
- 0x400780: HeapSize
- 0x400784: HeapSize
- 0x400788: HeapSize
- 0x40078c: HeapSize
- 0x400790: HeapSize
- 0x400794: HeapSize
- 0x400798: HeapSize
- 0x40079c: HeapSize
- 0x4007a0: HeapSize
- 0x4007a4: HeapSize
- 0x4007a8: HeapSize
- 0x4007ac: HeapSize
- 0x4007b0: HeapSize
- 0x4007b4: HeapSize
- 0x4007b8: HeapSize
- 0x4007bc: HeapSize
- 0x4007c0: HeapSize
- 0x4007c4: HeapSize
- 0x4007c8: HeapSize
- 0x4007cc: HeapSize
- 0x4007d0: HeapSize
- 0x4007d4: HeapSize
- 0x4007d8: HeapSize
- 0x4007dc: HeapSize
- 0x4007e0: HeapSize
- 0x4007e4: HeapSize
- 0x4007e8: HeapSize
- 0x4007ec: HeapSize
- 0x4007f0: HeapSize
- 0x4007f4: HeapSize
- 0x4007f8: HeapSize
- 0x4007fc: HeapSize
- 0x400800: HeapSize
- 0x400804: HeapSize
- 0x400808: HeapSize
- 0x40080c: HeapSize
- 0x400810: HeapSize
- 0x400814: HeapSize
- 0x400818: HeapSize
- 0x40081c: HeapSize
- 0x400820: HeapSize
- 0x400824: HeapSize
- 0x400828: HeapSize
- 0x40082c: HeapSize
- 0x400830: HeapSize
- 0x400834: HeapSize
- 0x400838: HeapSize
- 0x40083c: HeapSize
- 0x400840: HeapSize
- 0x400844: HeapSize
- 0x400848: HeapSize
- 0x40084c: HeapSize
- 0x400850: HeapSize
- 0x400854: HeapSize
- 0x400858: HeapSize
- 0x40085c: HeapSize
- 0x400860: HeapSize
- 0x400864: HeapSize
- 0x400868: HeapSize
- 0x40086c: HeapSize
- 0x400870: HeapSize
- 0x400874: HeapSize
- 0x400878: HeapSize
- 0x40087c: HeapSize
- 0x400880: HeapSize
- 0x400884: HeapSize
- 0x400888: HeapSize
- 0x40088c: HeapSize
- 0x400890: HeapSize
- 0x400894: HeapSize
- 0x400898: HeapSize
- 0x40089c: HeapSize
- 0x4008a0: HeapSize
- 0x4008a4: HeapSize
- 0x4008a8: HeapSize
- 0x4008ac: HeapSize
- 0x4008b0: HeapSize
- 0x4008b4: HeapSize
- 0x4008b8: HeapSize
- 0x4008bc: HeapSize
- 0x4008c0: HeapSize
- 0x4008c4: HeapSize
- 0x4008c8: HeapSize
- 0x4008cc: HeapSize
- 0x4008d0: HeapSize
- 0x4008d4: HeapSize
- 0x4008d8: HeapSize
- 0x4008dc: HeapSize
- 0x4008e0: HeapSize
- 0x4008e4: HeapSize
- 0x4008e8: HeapSize
- 0x4008ec: HeapSize
- 0x4008f0: HeapSize
- 0x4008f4: HeapSize
- 0x4008f8: HeapSize
- 0x4008fc: HeapSize
- 0x400900: HeapSize
- 0x400904: HeapSize
- 0x400908: HeapSize
- 0x40090c: HeapSize
- 0x400910: HeapSize
- 0x400914: HeapSize
- 0x400918: HeapSize
- 0x40091c: HeapSize
- 0x400920: HeapSize
- 0x400924: HeapSize
- 0x400928: HeapSize
- 0x40092c: HeapSize
- 0x400930: HeapSize
- 0x400934: HeapSize
- 0x400938: HeapSize
- 0x40093c: HeapSize
- 0x400940: HeapSize
- 0x400944: HeapSize
- 0x400948: HeapSize
- 0x40094c: HeapSize
- 0x400950: HeapSize
- 0x400954: HeapSize
- 0x400958: HeapSize
- 0x40095c: HeapSize
- 0x400960: HeapSize
- 0x400964: HeapSize
- 0x400968: HeapSize
- 0x40096c: HeapSize
- 0x400970: HeapSize
- 0x400974: HeapSize
- 0x400978: HeapSize
- 0x40097c: HeapSize
- 0x400980: HeapSize
- 0x400984: HeapSize
- 0x400988: HeapSize
- 0x40098c: HeapSize
- 0x400990: HeapSize
- 0x400994: HeapSize
- 0x400998: HeapSize
- 0x40099c: HeapSize
- 0x4009a0: HeapSize
- 0x4009a4: HeapSize
- 0x4009a8: HeapSize
- 0x4009ac: HeapSize
- 0x4009b0: HeapSize
- 0x4009b4: HeapSize
- 0x4009b8: HeapSize
- 0x4009bc: HeapSize
- 0x4009c0: HeapSize
- 0x4009c4: HeapSize
- 0x4009c8: HeapSize
- 0x4009cc: HeapSize
- 0x4009d0: HeapSize
- 0x4009d4: HeapSize
- 0x4009d8: HeapSize
- 0x4009dc: HeapSize
- 0x4009e0: HeapSize
- 0x4009e4: HeapSize
- 0x4009e8: HeapSize
- 0x4009ec: HeapSize
-

In Cuckoo Sandbox, during static analysis, we can view the imported APIs to understand the malware's capabilities and interactions with the system. Additionally, we can perform string analysis, extracting and examining strings embedded within the file, which can reveal valuable information about the malware's intent and potential behavior. These insights obtained from static analysis are crucial in assessing the threat and designing appropriate defense mechanisms.

During network analysis in Cuckoo, we observed the malware's attempt to access the enigmatic URL. This activity is a critical indicator of potential command and control.

| Name | Response |
|---|-------------------|
| www.iuqerfsodp9ifajoposdfhg0surjfaewrwergrwea.com | A → 104.16.173.80 |
| | A → 104.17.244.81 |

Figure 47: Enigmatic URL

Appendices

A. Yara Rules

```
rule wannacry {

  meta:
    date = "2023-08-04"
    author = "xpinux"
    description = "YARA rule to detect strings associated with WannaCry ransomware"

  strings:
    // Fill out identifying strings and other criteria
    $PE_Magic_Byte = "MZ"
    $str1 = "icaccls . /grant Everyone:F /T /C /Q"
    $str2 = "cmd.exe /c \"%s\""
    $str3 = "I15p7UMMngoJlpMvKpHjCrdfJNXj6LrLn"
    $str4 = "I2t9YDPgwueZ9NyMgw519p7AA8isjr6SMw"
    $str5 = "I3AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94"
    $str6 = "C:\\%s\\qeriuwjhrf"
    $str7 = "http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com" ascii

  condition:
    // Fill out the conditions that must be met to identify the binary
    $PE_Magic_Byte at 0 and (( $str7 and $str1 ) or ( $str2 and $str6 ) or ( $str3 and $str4 and $str5 ))
}
```

B. Figures

| | |
|--|----|
| Figure 1: File Type | 5 |
| Figure 2: MBC Objectives | 6 |
| Figure 3: Sample of APIs Used | 7 |
| Figure 4: IOC from Strings | 8 |
| Figure 5: PEstudio Info | 8 |
| Figure 6: PEstudio Imports Indicators | 9 |
| Figure 7: Network Detonation - Enigmatic URL | 10 |
| Figure 8: Wannacry Ransomware | 11 |
| Figure 9: Immediately after running TCPview | 11 |
| Figure 10: DLL Used | 12 |
| Figure 11: Creation of taskshe.exe | 12 |
| Figure 12: Process Tree | 12 |
| Figure 13: Parent PID Analysis | 12 |
| Figure 14: Unpacking Directory | 13 |
| Figure 15: Keys Deleted from Ransomware | 13 |
| Figure 16: Persistence Service | 13 |
| Figure 17: Keys Added | 14 |

| | |
|--|----|
| Figure 18: Values Added | 14 |
| Figure 19: Values Added | 14 |
| Figure 20: Cutter Dashboard Screen | 15 |
| Figure 21: Main of ransomware..... | 16 |
| Figure 22:Call fcn.00408090 | 17 |
| Figure 23: 0040f020 Fuction | 18 |
| Figure 24: First Call..... | 18 |
| Figure 25: Payload | 19 |
| Figure 26:Payload | 20 |
| Figure 27: Payload | 21 |
| Figure 28: x32dbg..... | 22 |
| Figure 29: Breaking Point for Enigmatic URL | 23 |
| Figure 30: Finding the sweet spot | 23 |
| Figure 31: If network is not Enabled - ZF 1..... | 24 |
| Figure 32: If network is enabled and reaches out to the URL ZF 0..... | 24 |
| Figure 33: With ZF 0 that we changed earlier, the program finishes | 24 |
| Figure 34: Enigmatic URL 200 OK from inetsim..... | 25 |
| Figure 35: ProcMon Connections..... | 25 |
| Figure 36: Worm Attributes - SMB | 25 |
| Figure 37: TCPView Wannacry | 25 |
| Figure 38:Creation of taskshe.exe | 25 |
| Figure 39: Process Tree | 25 |
| Figure 40: Directory of Ransomware | 26 |
| Figure 41: ProcExp of wannacry and taskshe.exe | 26 |
| Figure 42: Service for persistence | 27 |
| Figure 43: Home Screen Of Cuckoo Sandbox..... | 28 |
| Figure 44: Submitting Wannacry without intenret to Cuckoo..... | 29 |
| Figure 45: Cuckoo Summary..... | 30 |
| Figure 46: Cuckoo Static Analysis..... | 30 |
| Figure 47: Enigmatic URL | 31 |