Icon

Description automatically generated

Practical Malware Analysis & Triage

Malware Analysis Report

Ransomware.wannacry

By Abdulah Ibrahim Alharas

Ransomware.wannacry

|  |  |
| --- | --- |
| SHA256 hash | B1BEDE289FBAC6E83A038C18DB7D7B2AA392590D0A9C5550B3FF6EF1062D0AC7 |

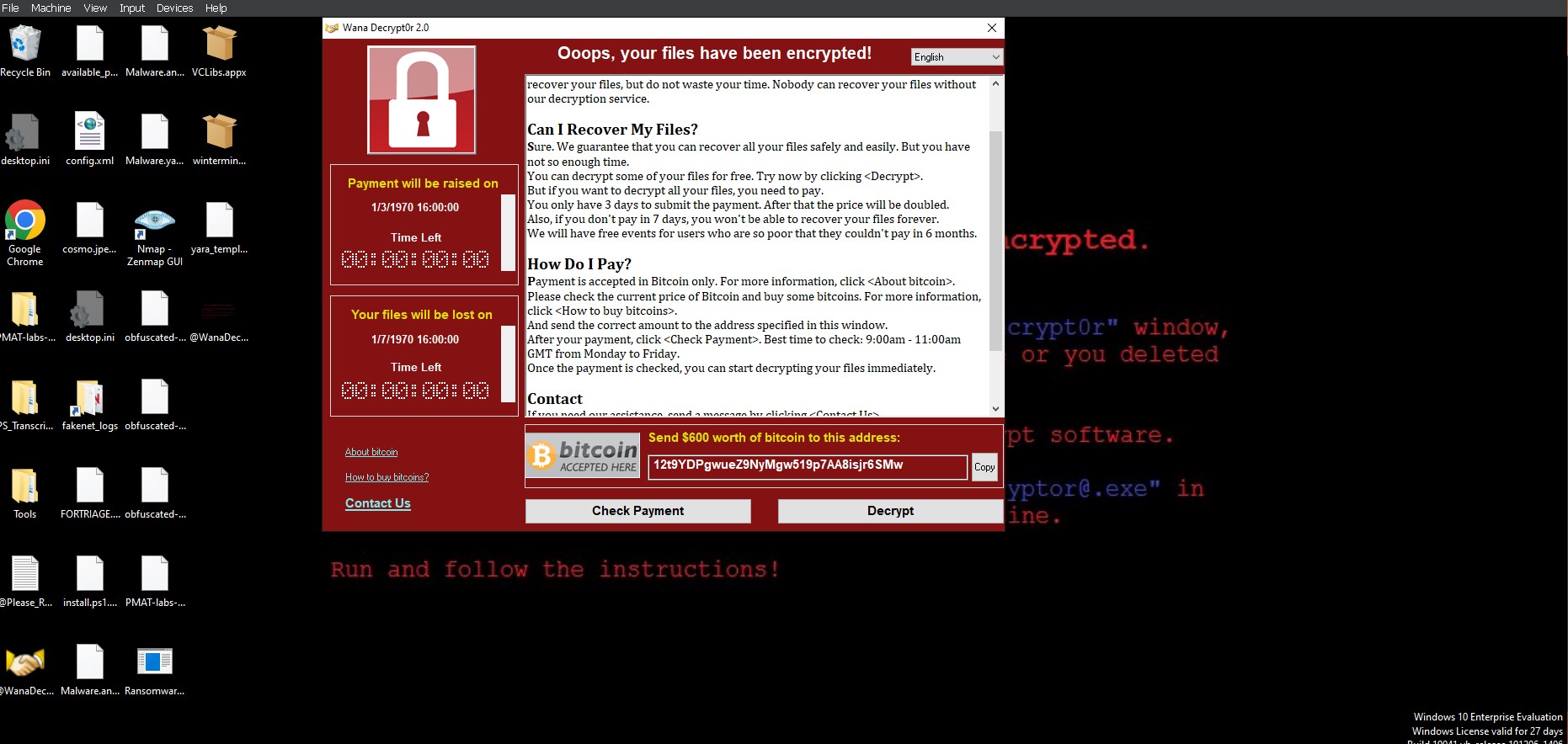
In 2017, the well-known ransomware WannaCry was made available. Through the SMB protocol, it encrypts the files on the device and attempts to spread around the network to encrypt other devices connected to the same network.  
  
The complete analysis of the ransomware and the findings from the various tools and approaches are provided below.  
  
Appendix A has the YARA signature regulations attached. Hashes and a malware sample have been submitted to VirusTotal for additional analysis.

High level technical summary

Wannacry is a ransomware which targets windows to encrypt the user’s files, which include important files, to ask the users for ransom payments in bitcoin to decrypt the files, it also has the ability to propagate through the network and related devices to encrypt as many devices as possible.

Initial detonation:

First I wanted to know how or what the malware do on the victim device, so I ran the ransomware on a virtual machine without a network connection, and here are the findings:



As show new icons and shortcuts has appeared on the desktop, with files like cosmo.jpeg has been corrupted , or decrypted by the ransomware. Only after a few seconds of detonation a scary-looking background and a message appears ask the user for payments. At this point files and tools are inaccessible and encrypted.

Static analysis:

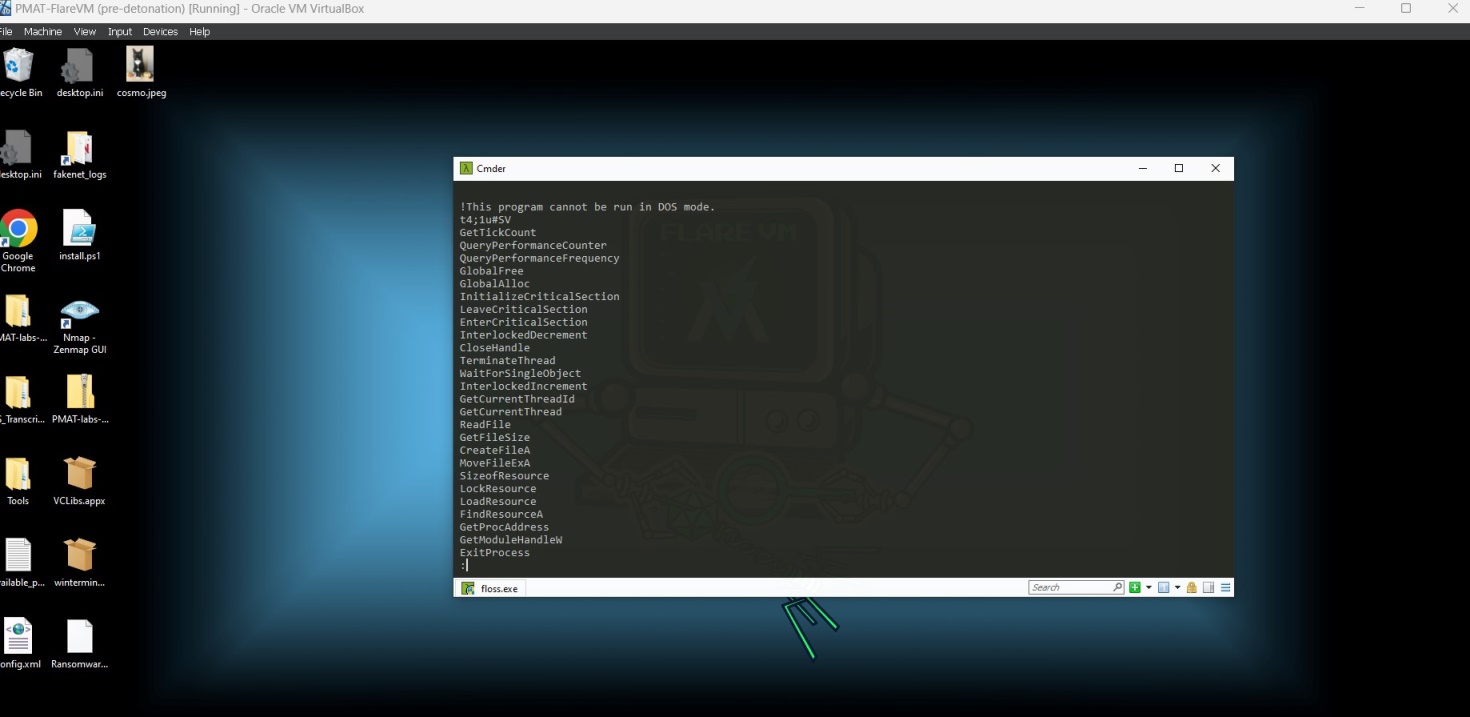
Filename:

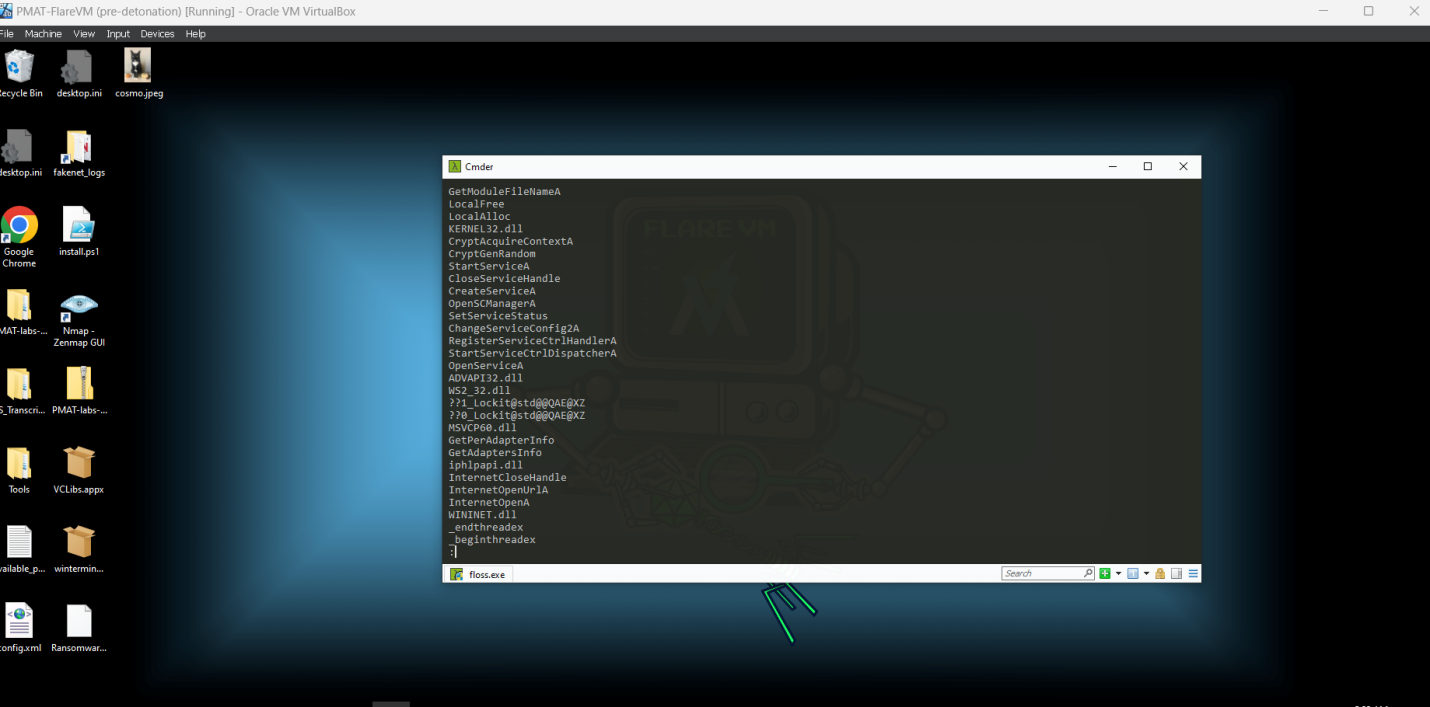
Ransomware.wannacry.exe

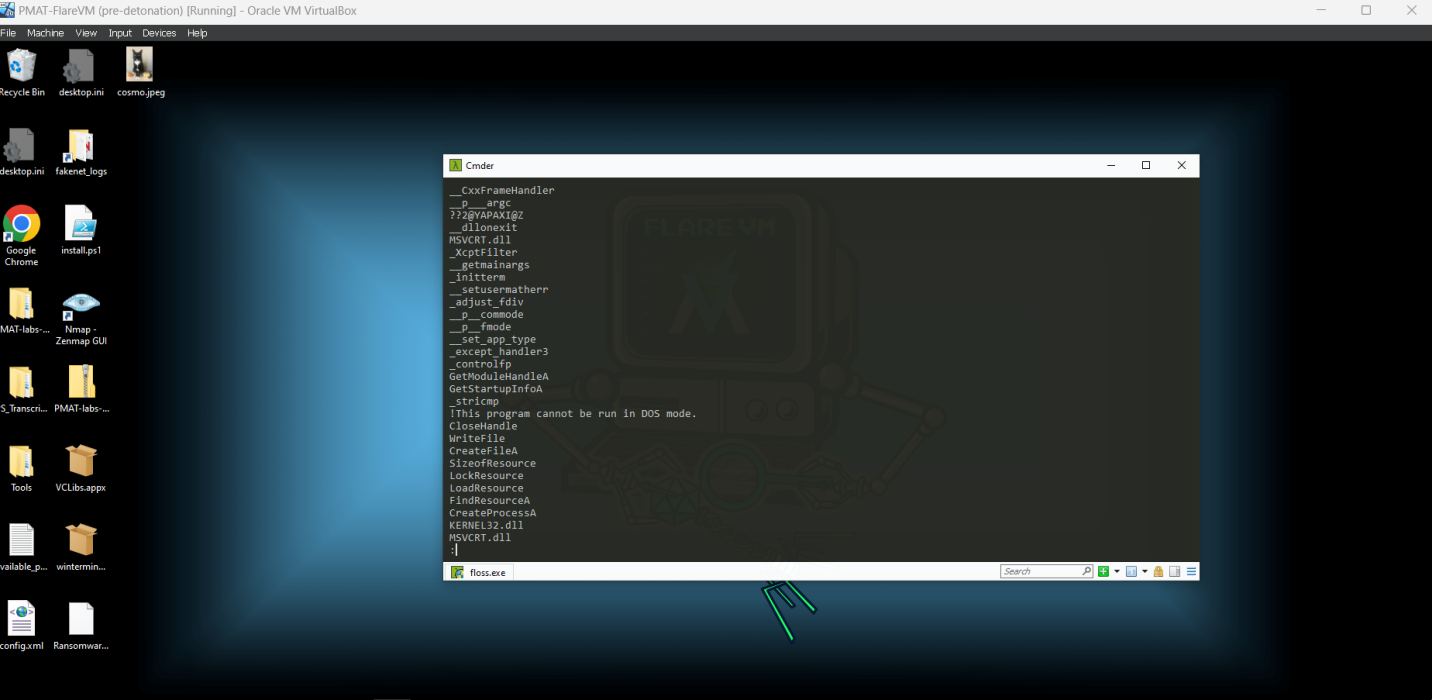
SHA256 hash :

B1BEDE289FBAC6E83A038C18DB7D7B2AA392590D0A9C5550B3FF6EF1062D0AC7

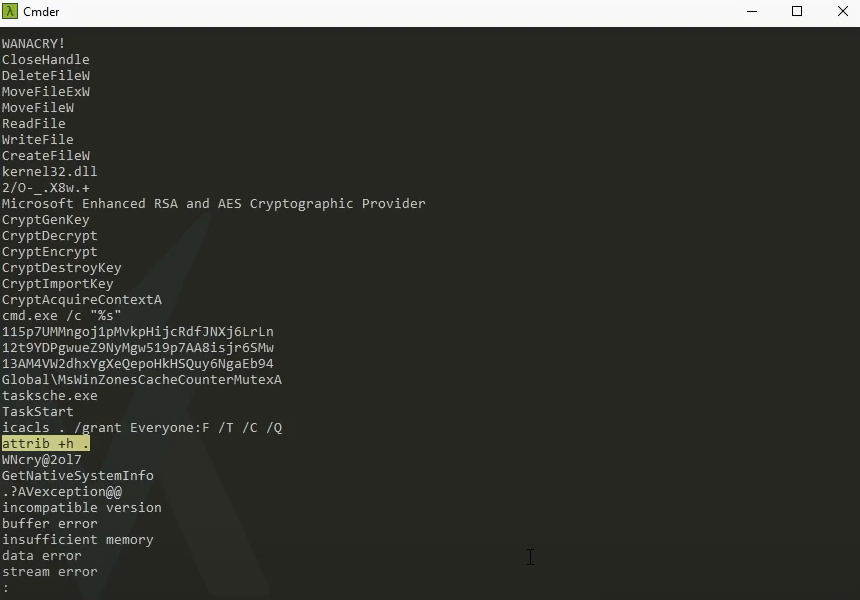
Suspicious string include dlls, APIs, and commands are shown below:



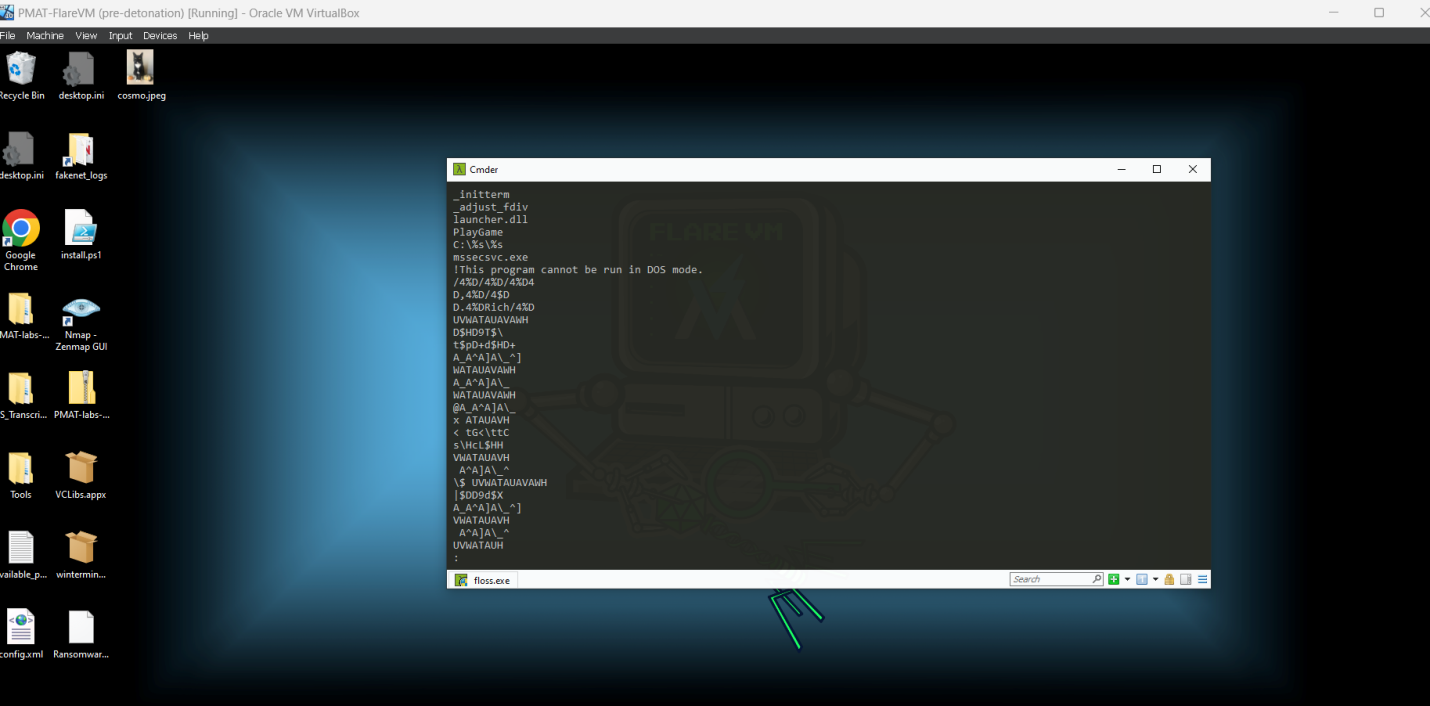
Dlls, libraries, and APIs used for the executable :



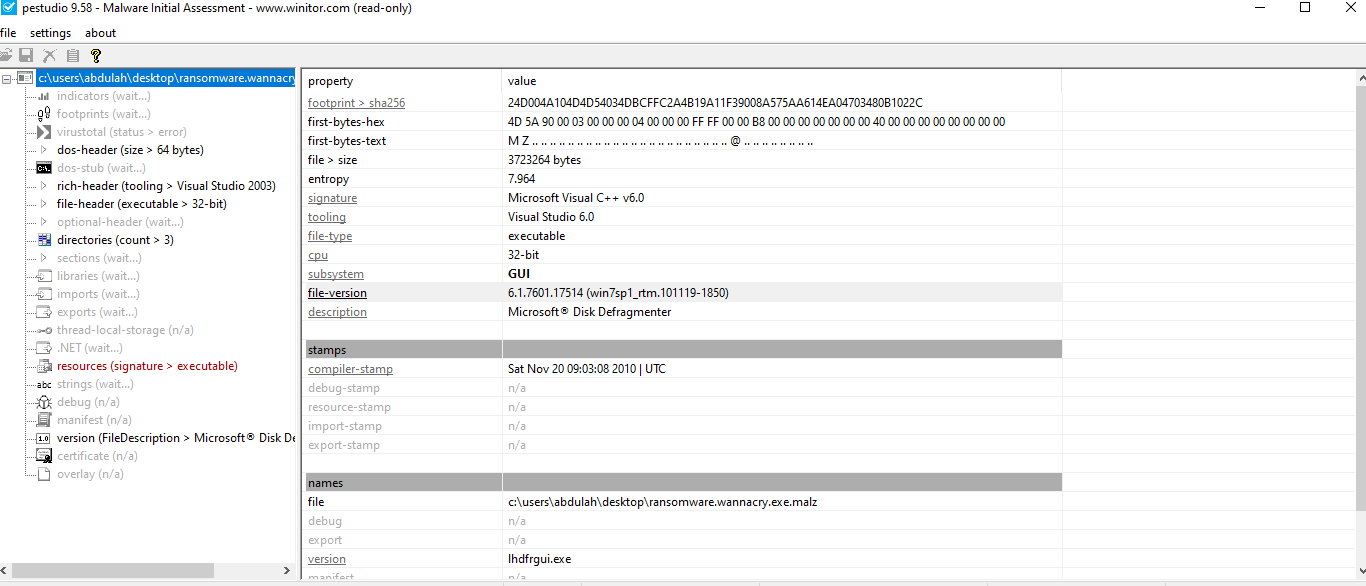
In the following screenshot the executable perform and inject a command using base64 with “taskche.exe” act as a second payload for the ransomware, and the string “attrib +h” to hide a file in a directory.



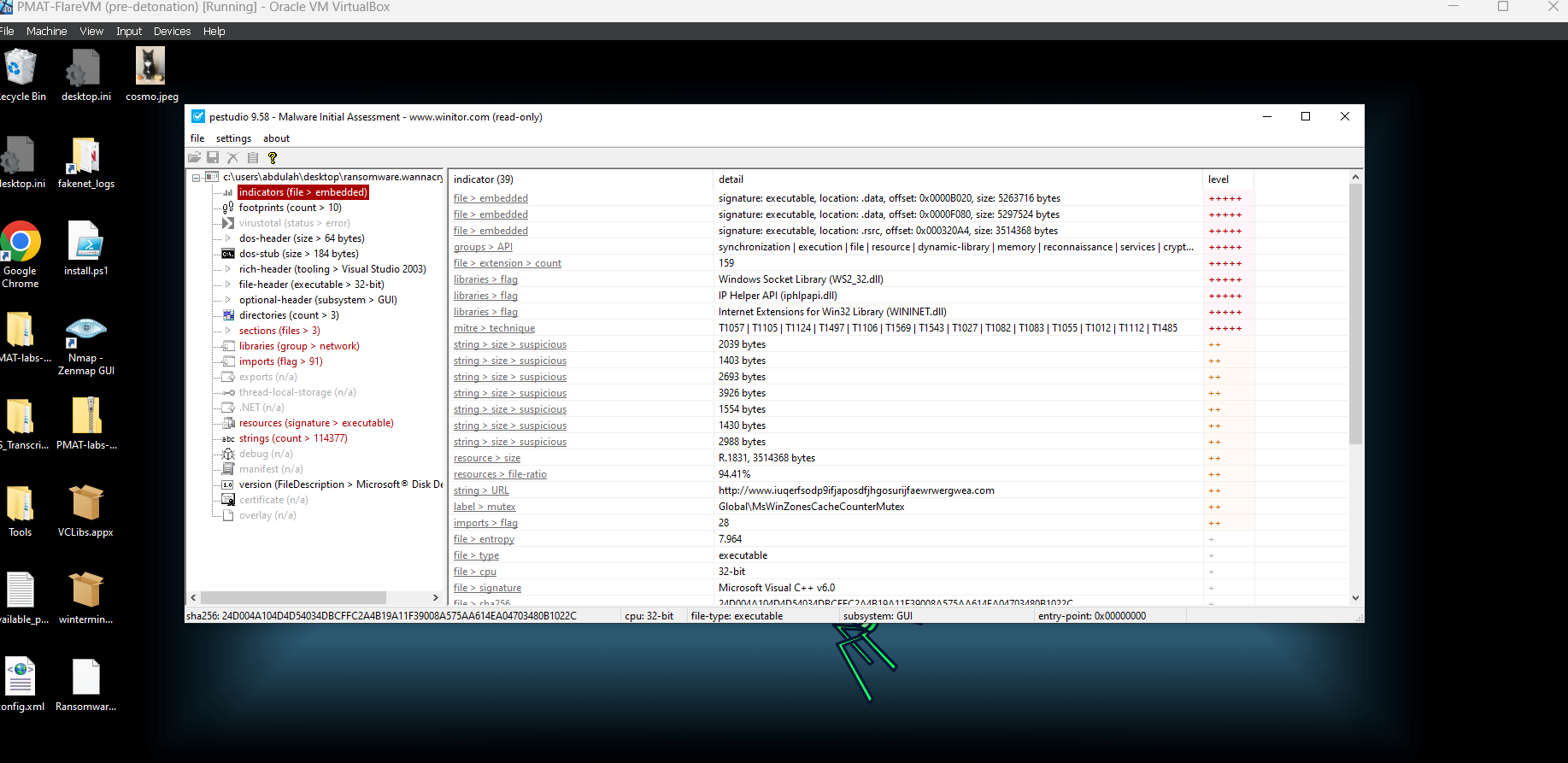
The ransomware delivered with another embedded executables in it act as second stage payloads.

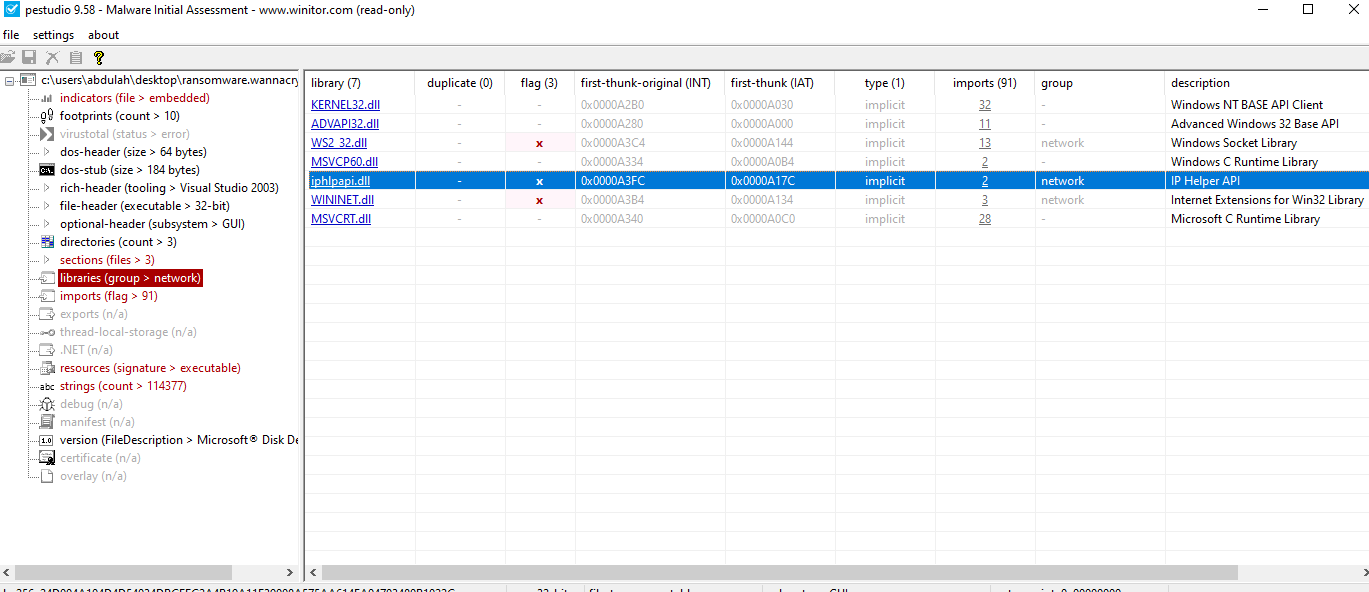


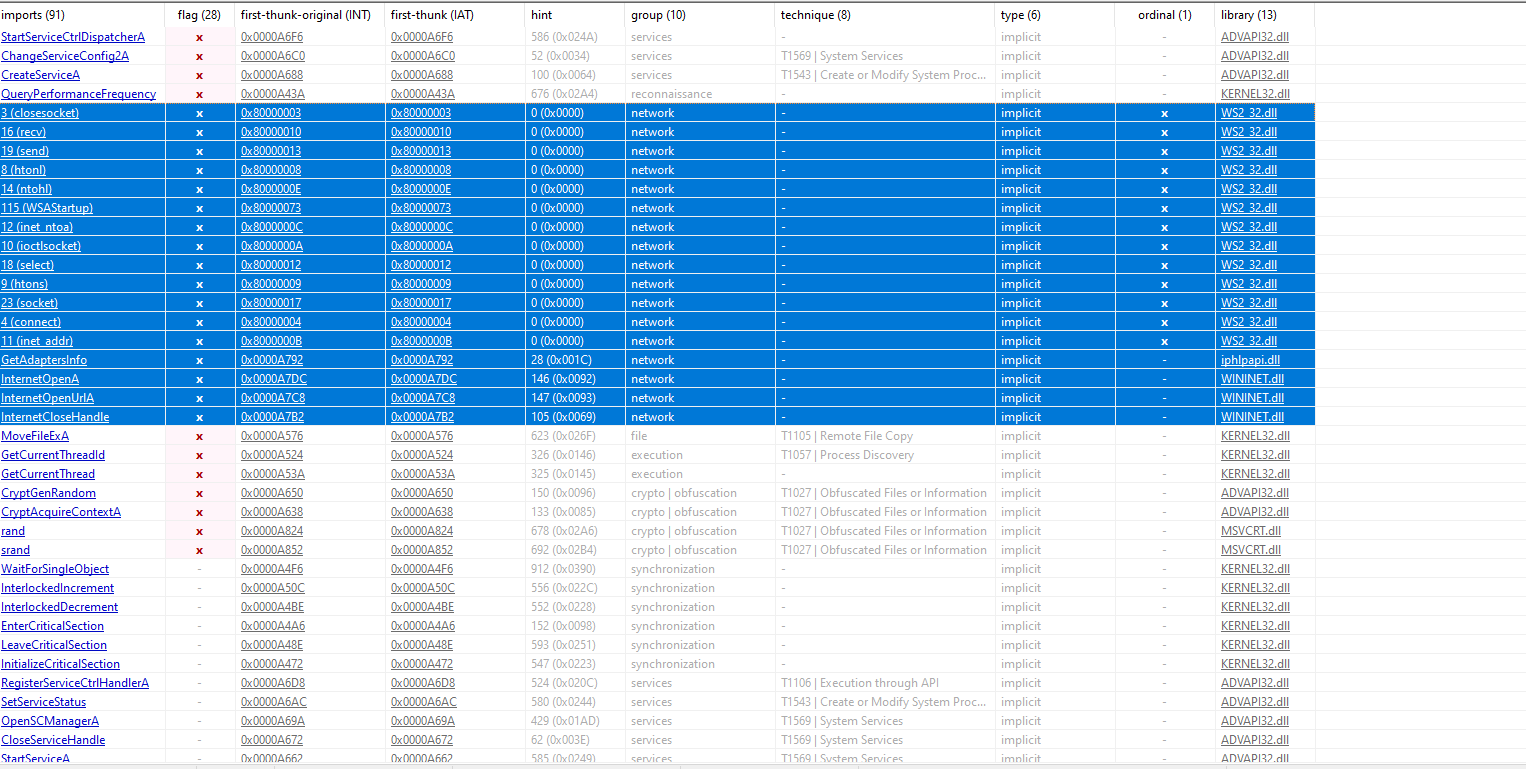
Next I started analyzing the ransomware using PEStudio with the results shown below :



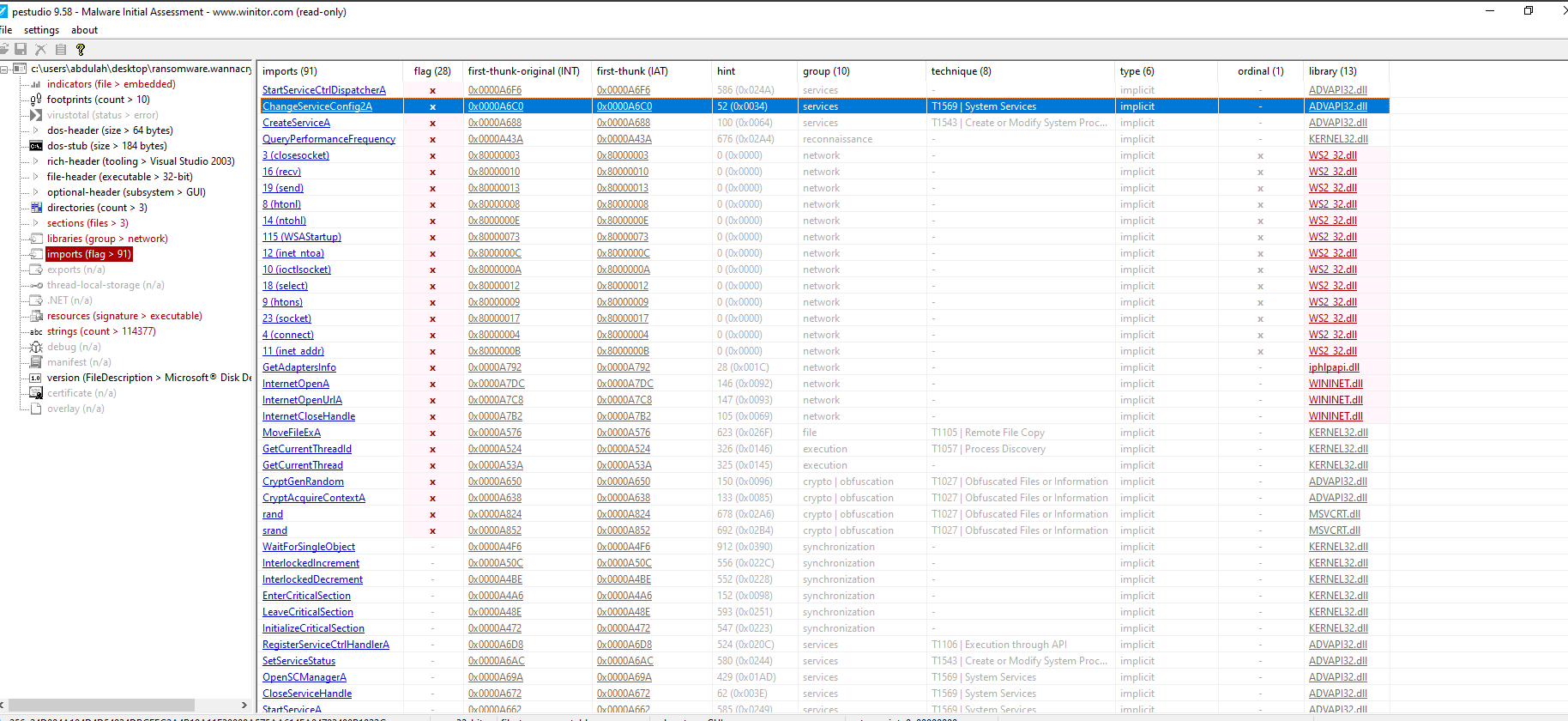
As show above and proven below, the ransomware has another payloads embedded within the binary of the executable as a second payload. The payload also has the ability to create internet sockets and threads using libraries like WS2\_32.dll, WININET.dll, and iphlpapi.dll:

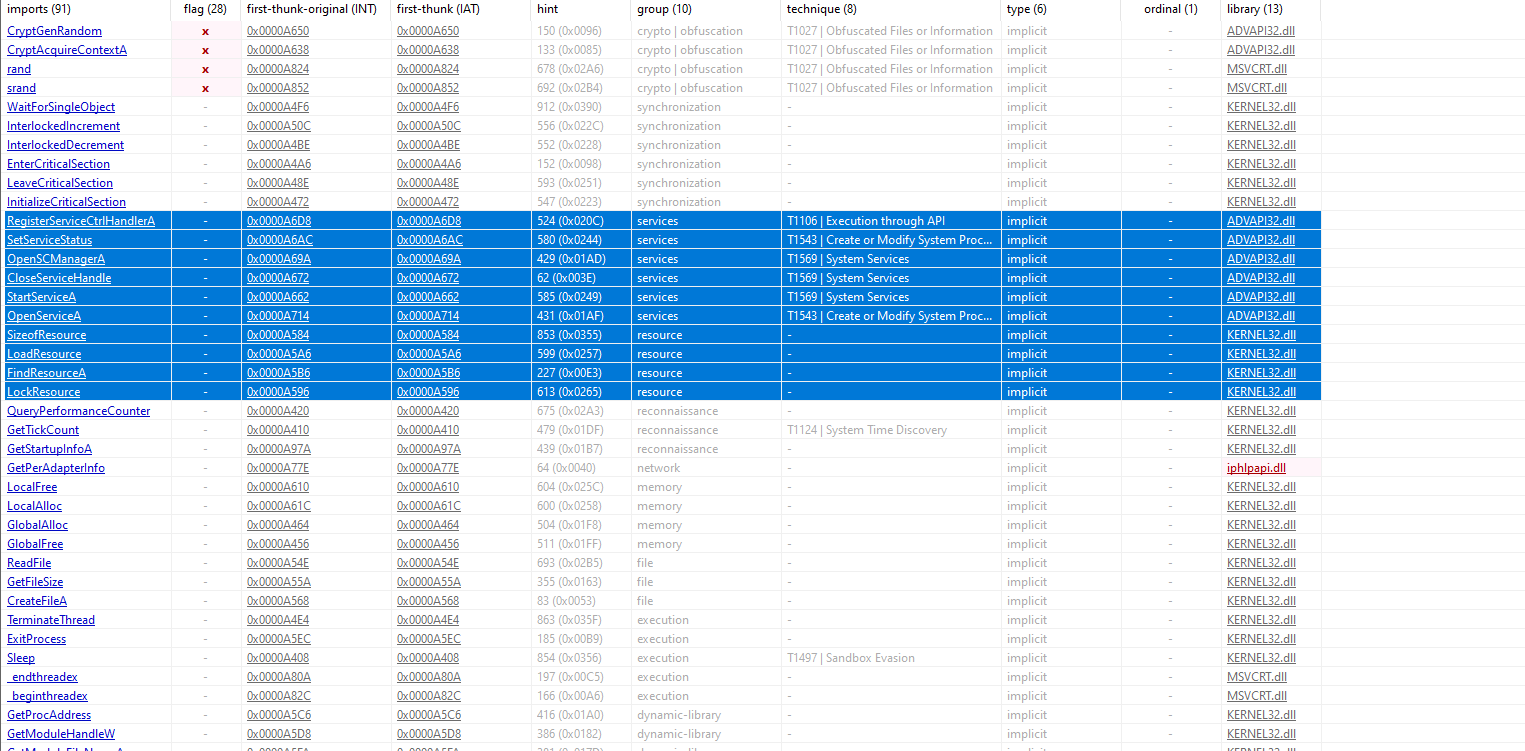






By importing “ChangeServiceConfig2A” the ransomware use it as a persistence method to set foot on the device and reinitiate after shutdowns:

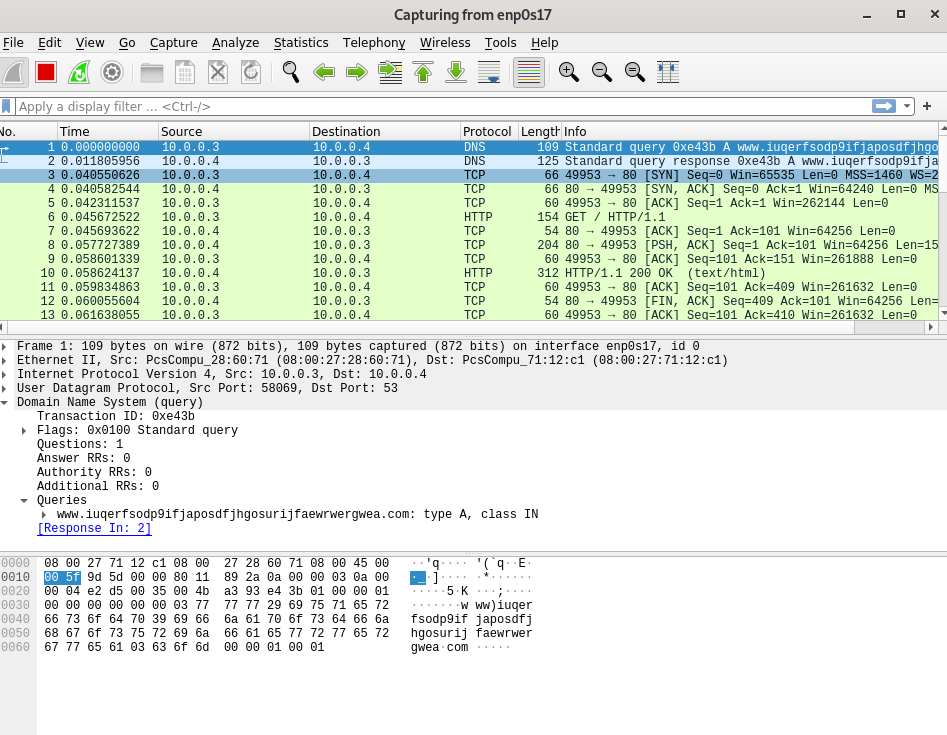


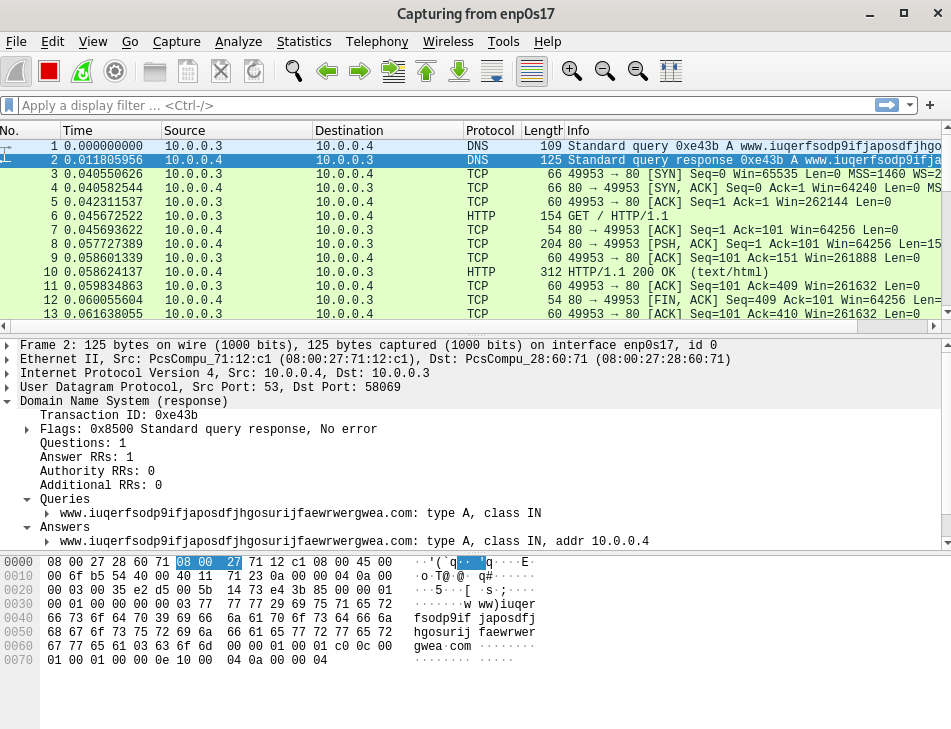


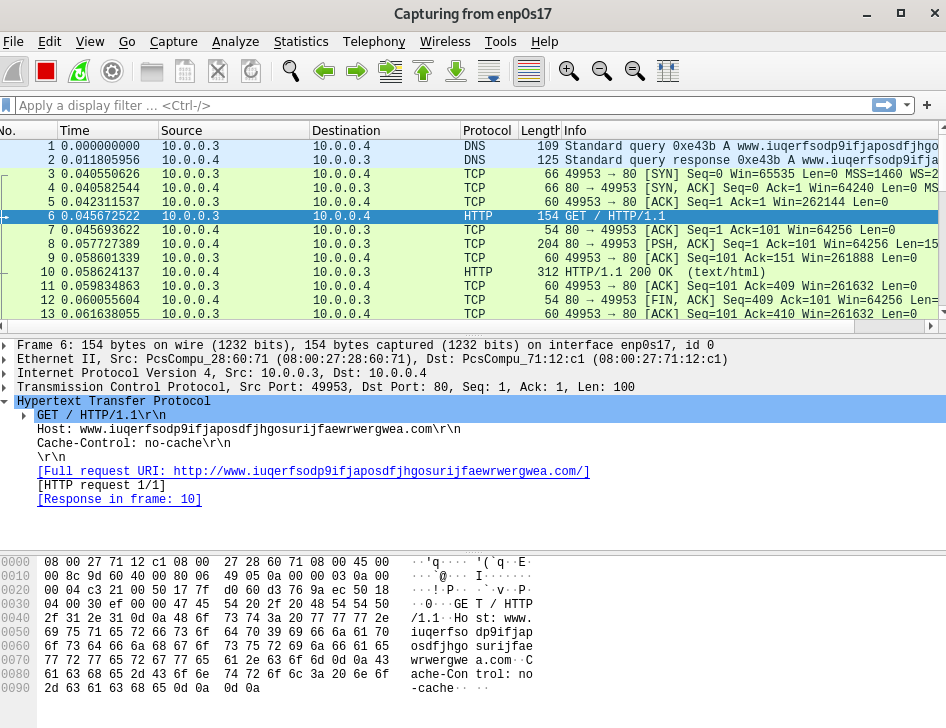
Dynamic Analysis:

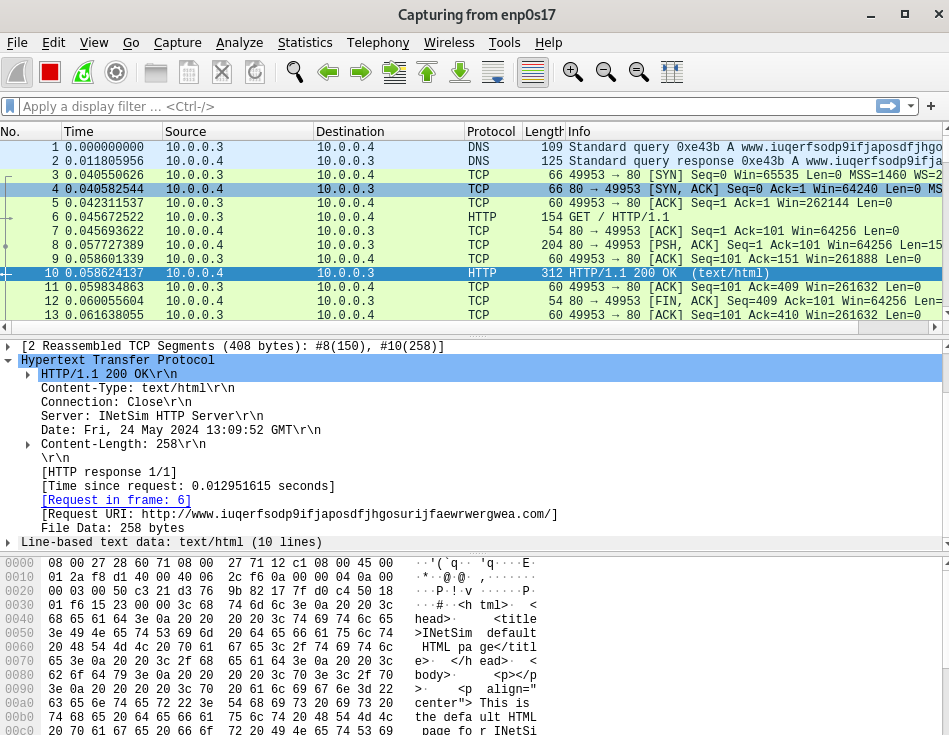
Using wireshark I found the following :

The ransomware tries to reach out to the URL in the screenshots :





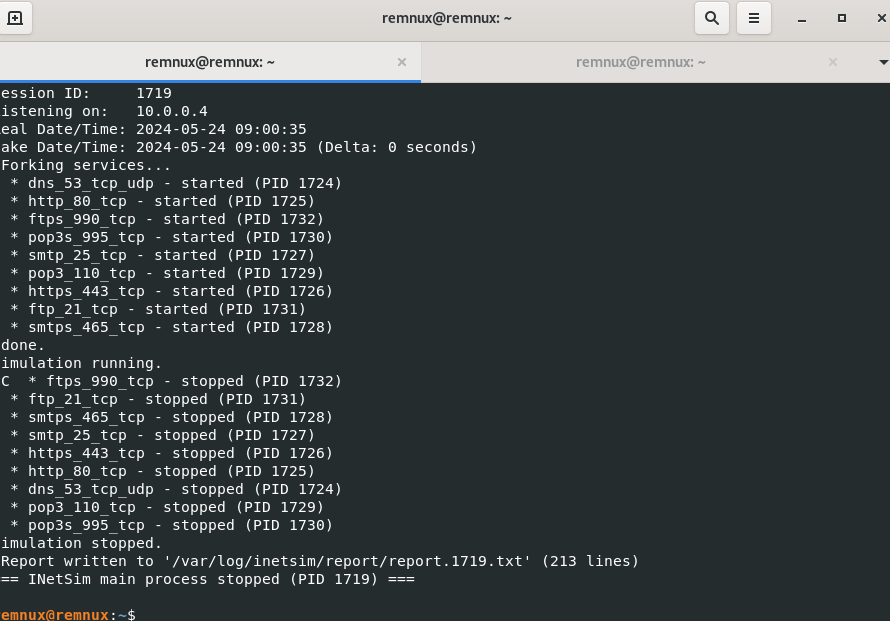




When the payload succeeds in connecting to the URL it dosen’t run on the device and so the file are not encrypted:



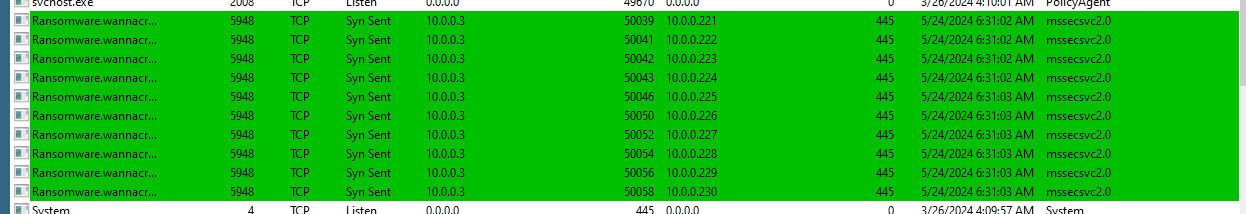
Inetsim act as an internet server:



When the device is not connected to the internet it encrypt the files on the system :

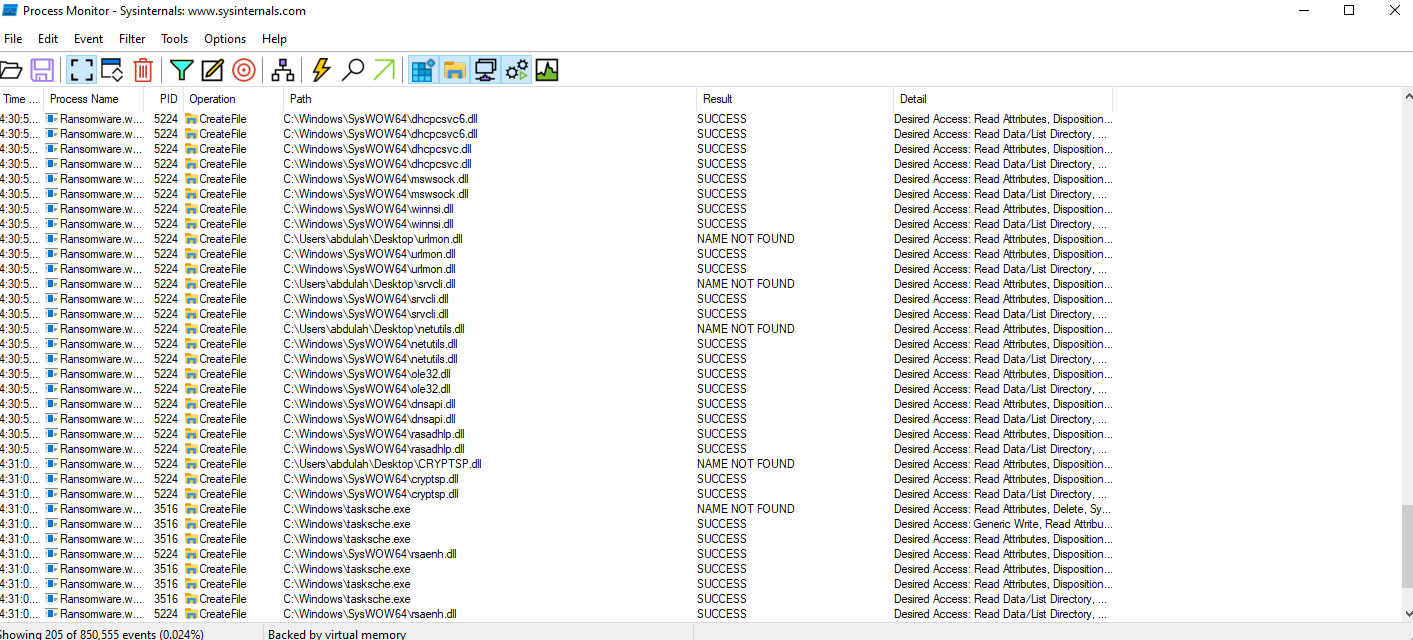


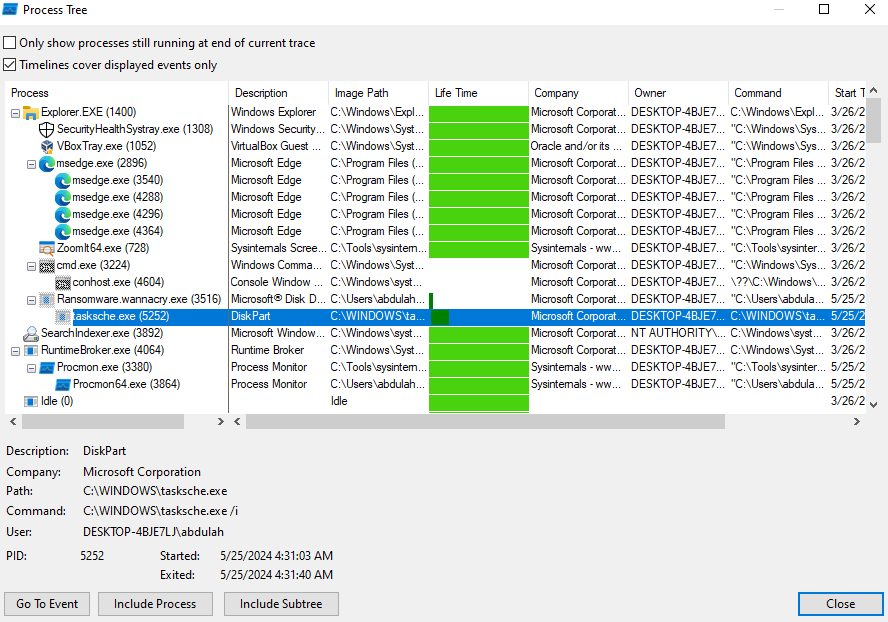
TCP connections and ports by the ransomware using the TCPView :



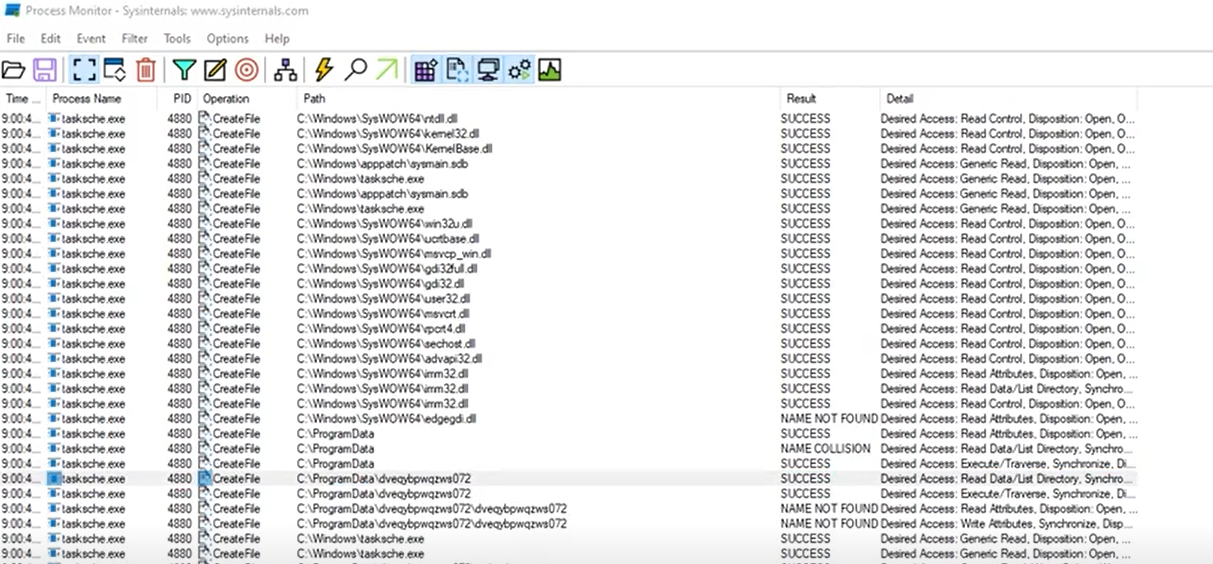
Moving to the next phase of analysis, I used procmon and found the following :

The ransomware spawns a new executable payload and write to the directory “c:\Windows\tasksche.exe” as show below :

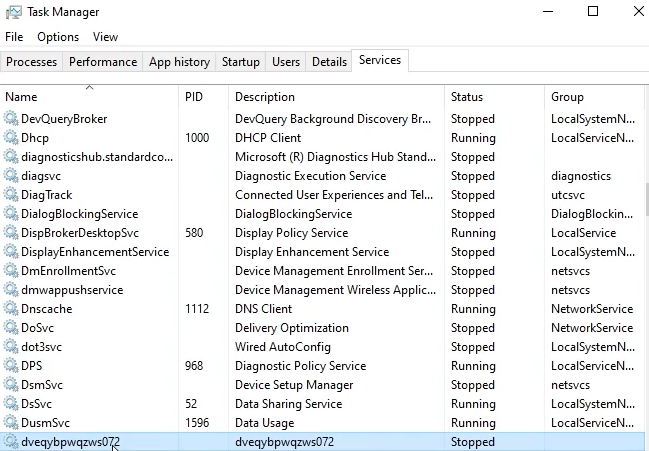




The ransomware created a staging area in a strange file name hidden in the directory “c:\program\” as show below :



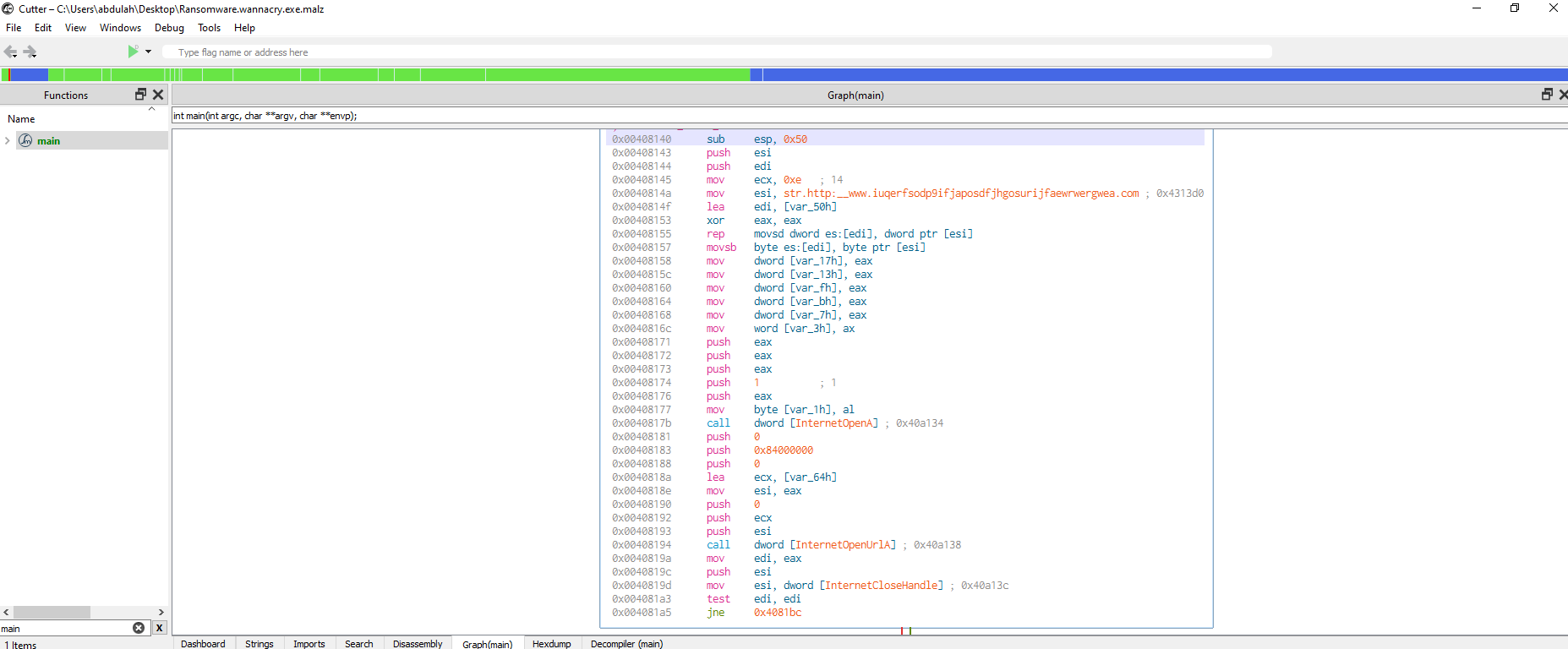
In task manager we can find that a proccess name identical to that of the file found in the directory above, that most likely used as a persistence method for the payload :



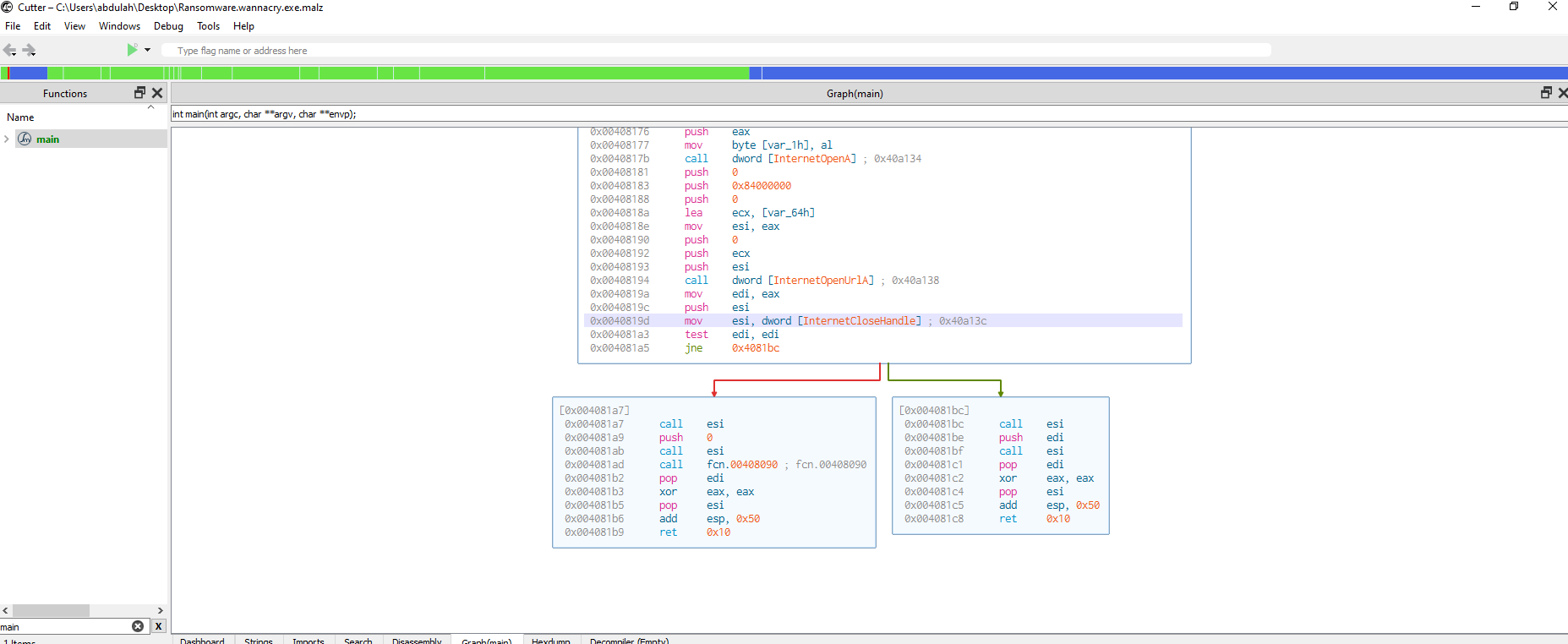
Advanced Static Analysis:

In the next phase, I used cutter for analysis and here are the findings :

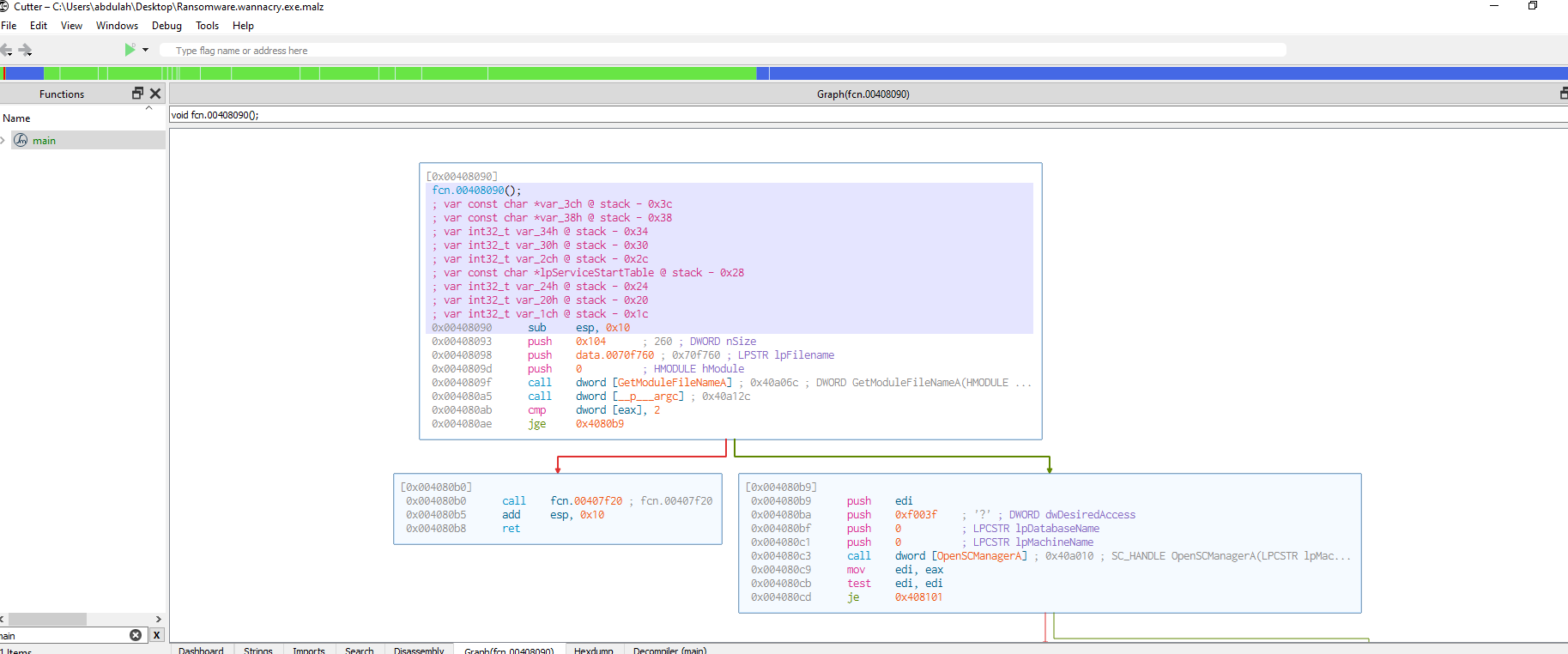
The ransomware used various API calls to try connecting to the URL, found in the dynamic analysis above, follow along in cutter I found that the payload calls The API “InternetOpenUrlA” with the esi register pushed into the stack containing the url right before the call, as the url stored in the esi register used as parameter for the api call.



The result of the API call is then stored in eax register and then moved to edi register, so the register can be tested against itself to store the result in zero flag and decide to which set of instructions the program will go to using the instruction “jne” :

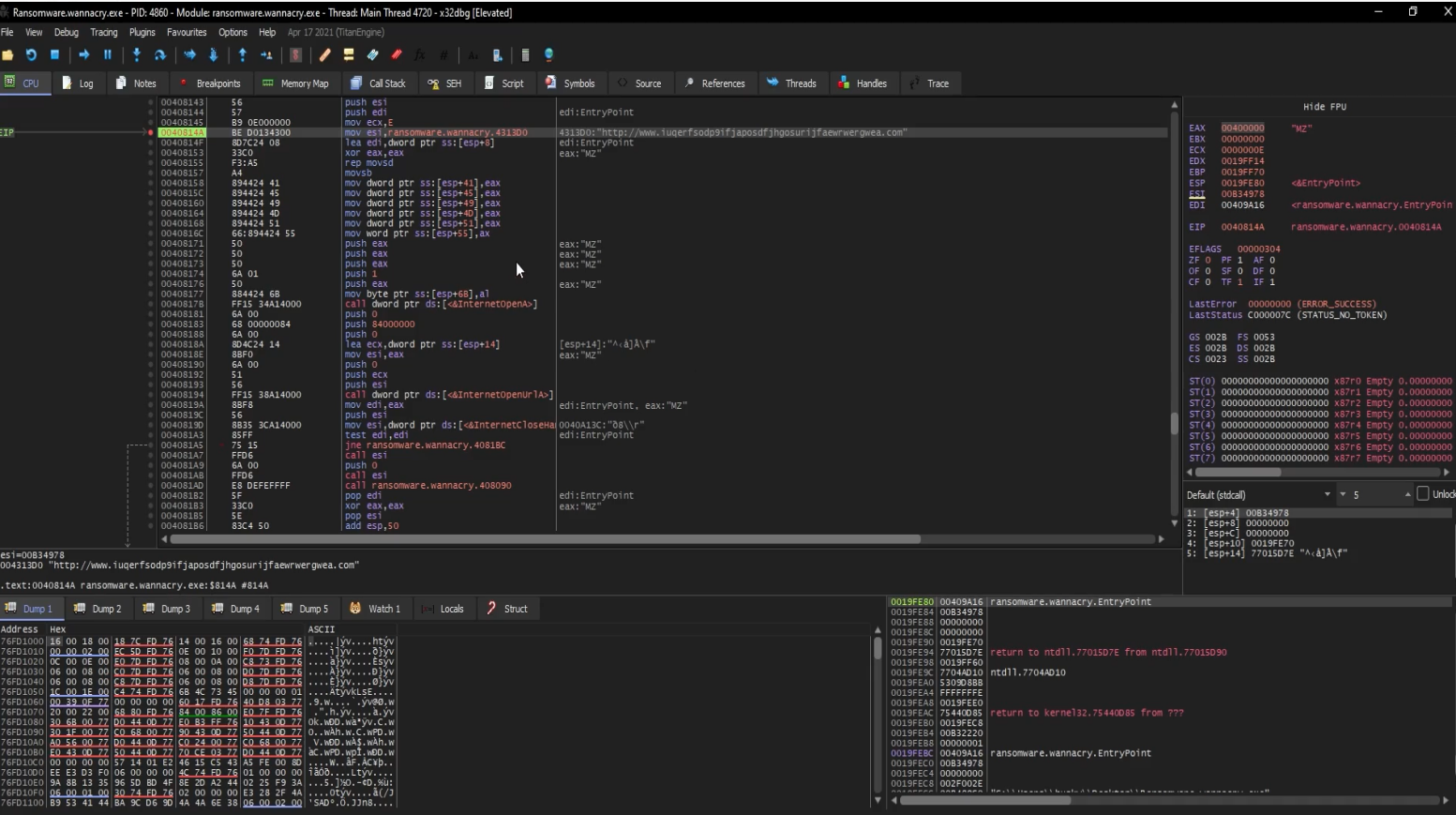


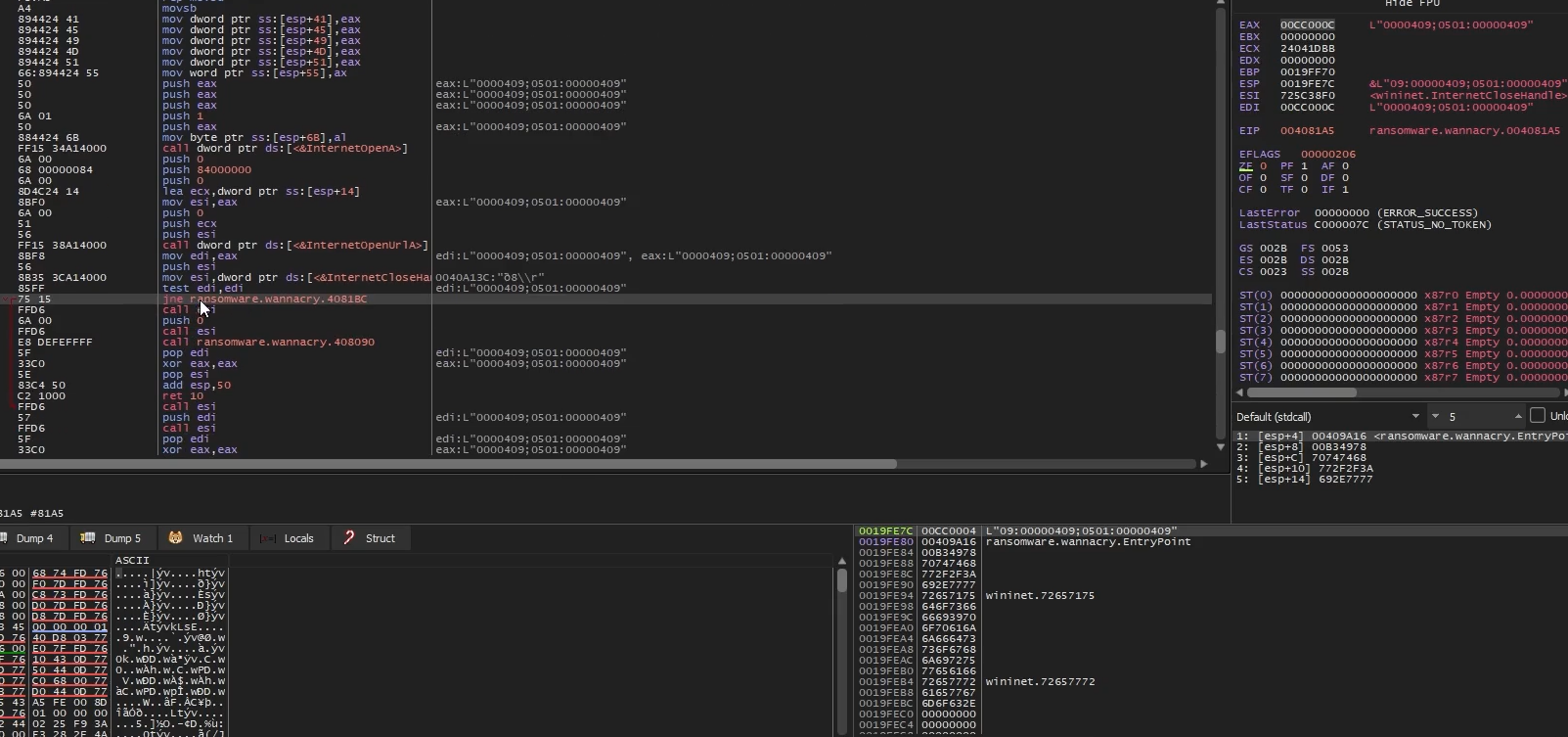
If the program connected to the malicious domain the zero flag will be 1 and so the jump go the address specified in the instruction and will not encrypt the the files on the system, otherwise it go the left hand side set of instructions to begin its encryption process, calling the function “fcn.00408090” :



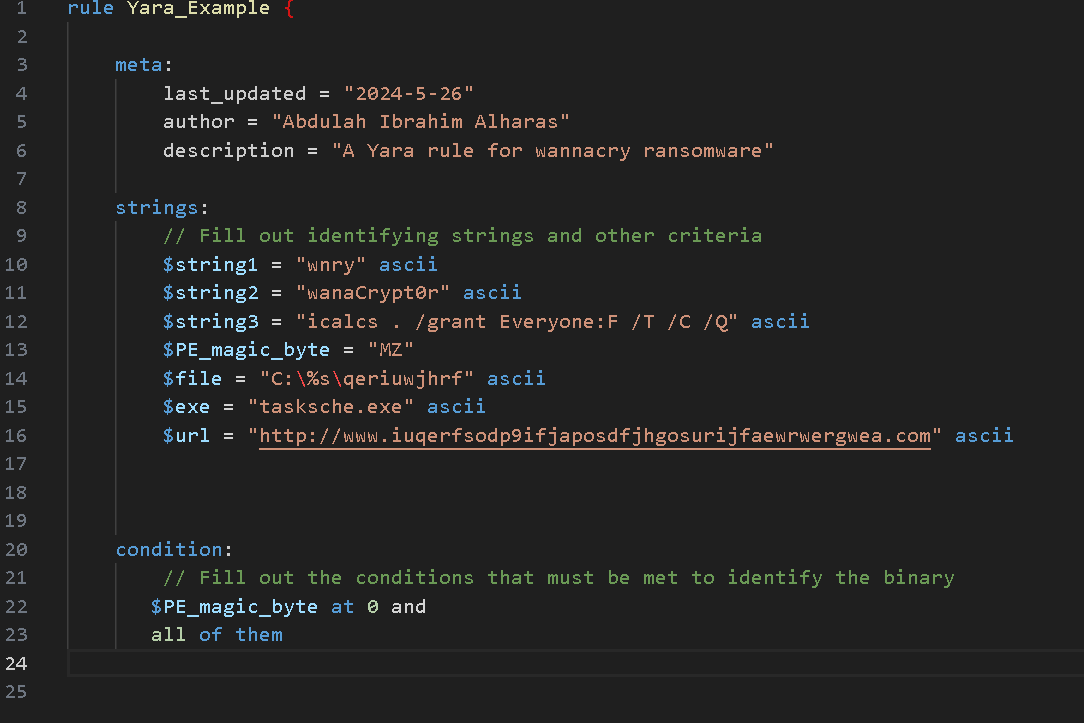
Advanced Dynamic Analysis :

By using debugger we can see that the same process take place by with the ability to change the content of the registers and flags on the right hand side so we can initiate our internet server , inetsim , and change the zf , zero flag , to zero let the ransomware initiate its process of encryption:





**Appendices:**

****