Malicious Software Analysis

Basic Analysis

Yousif Abudieh 20210762 (Leader)

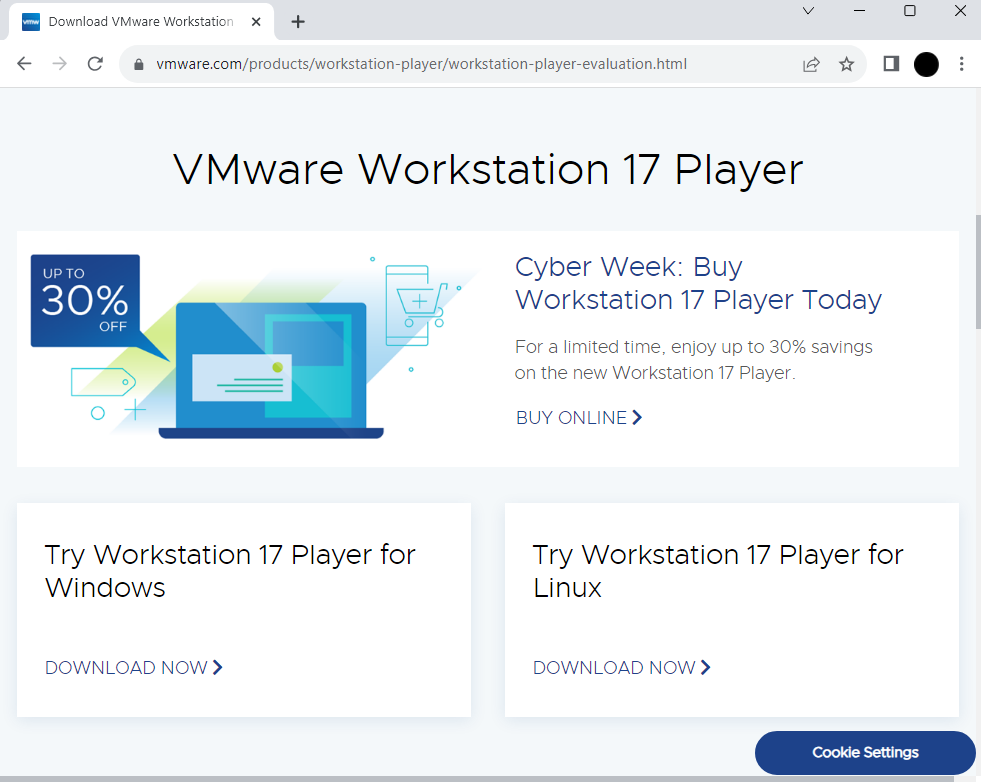
Abdallah Allahham 20210146

Abdulrahman Dweik 20210102

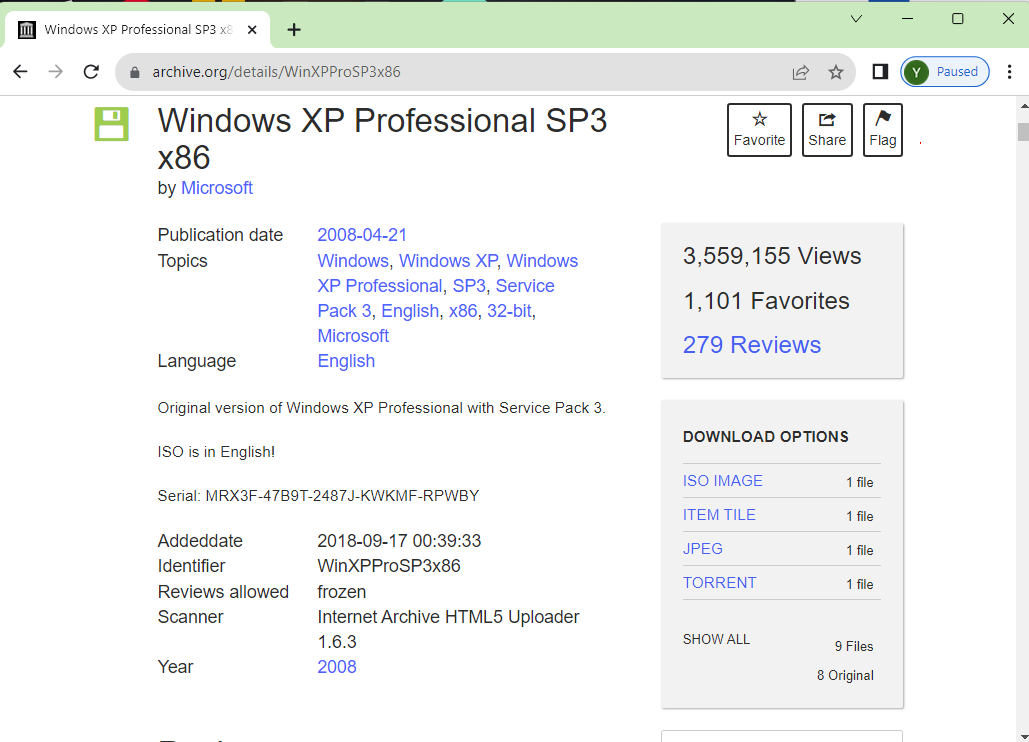
**1- Virtualization phase :**

Since we are downloading malware, we need an air gapped real machine or a virtual machine, we are using a virtual machine in order to see full functionality of the malware.

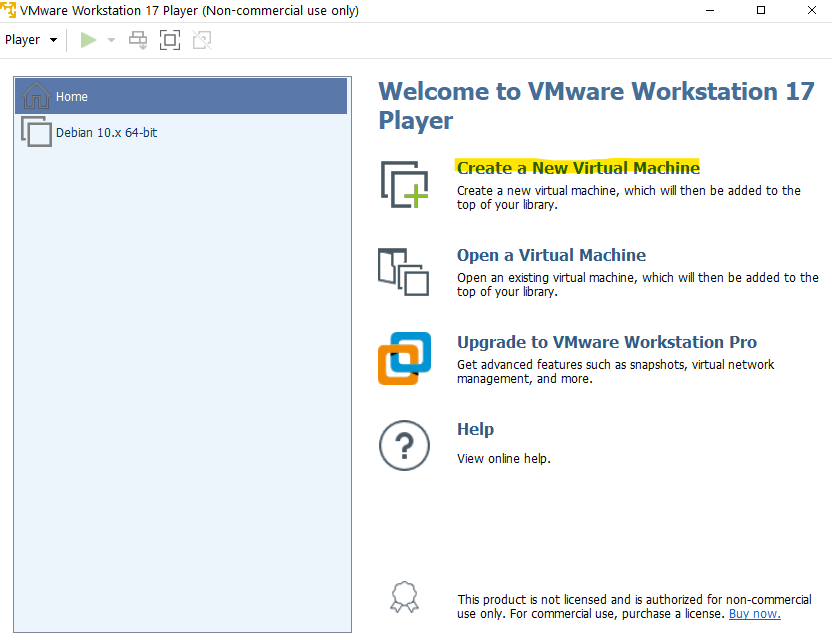
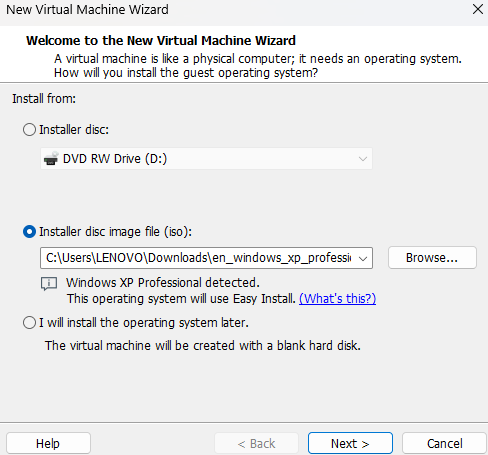
We downloaded VMware from <https://www.vmware.com/products/workstation-player/workstation-player-evaluation.html>

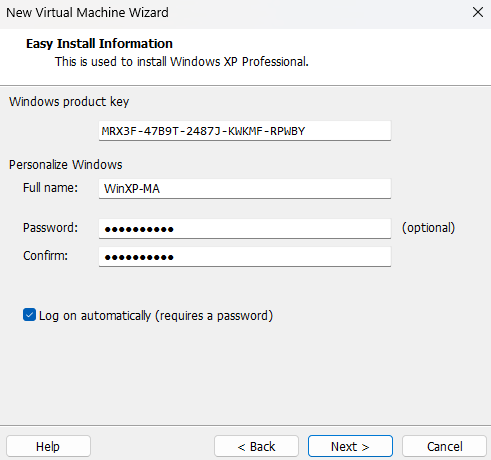


Then we downloaded WinXP iso image from <https://archive.org/details/WinXPProSP3x86>



Creating the virtual machine:

 A screenshot of a computer

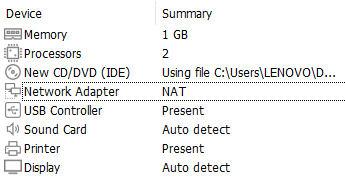
Description automatically generated

A screenshot of a computer

Description automatically generated A screenshot of a computer program

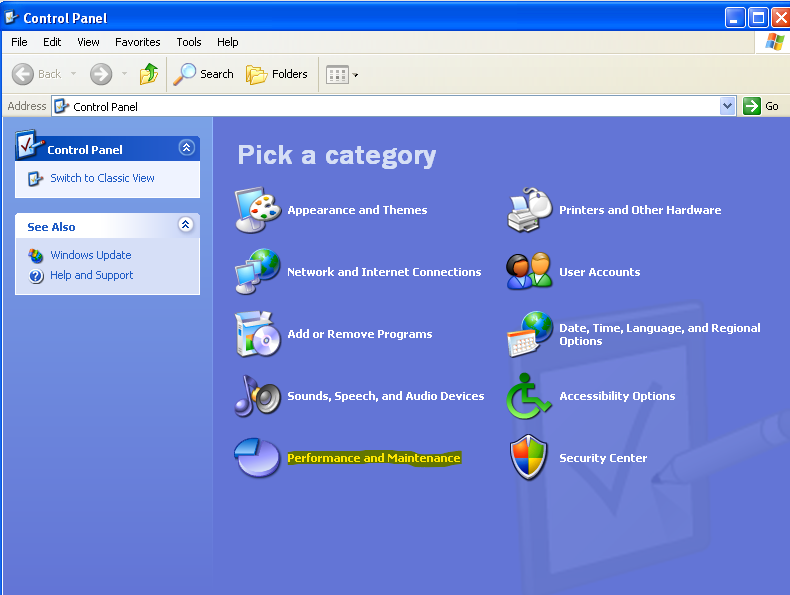
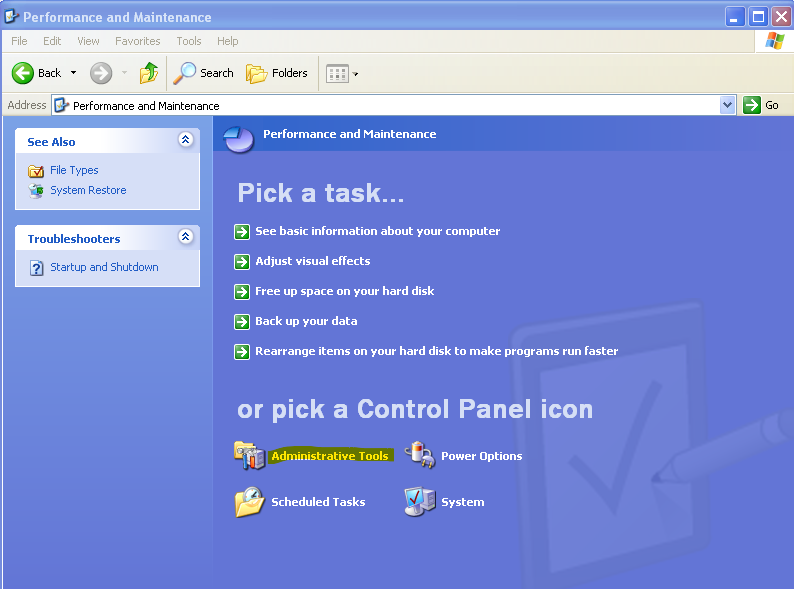
Description automatically generated

A screenshot of a computer

Description automatically generated 

Hardware customization:

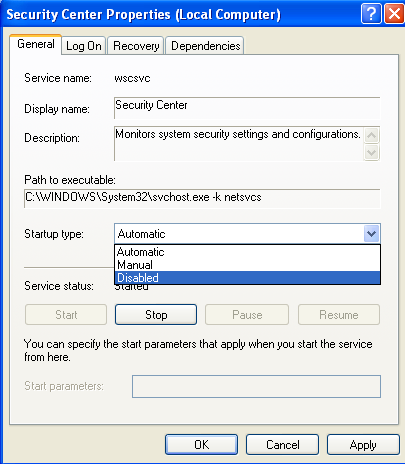
Disabling Security:

A screenshot of a computer

Description automatically generated A computer screen shot of a computer

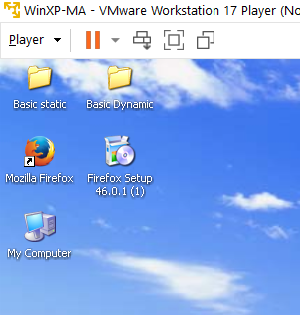
Description automatically generated

 A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

Then we downloaded firefox and the tools needed for the analysis.

 A screenshot of a computer

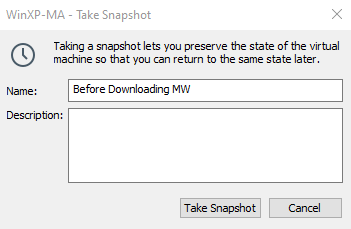
Description automatically generated

A screenshot of a computer

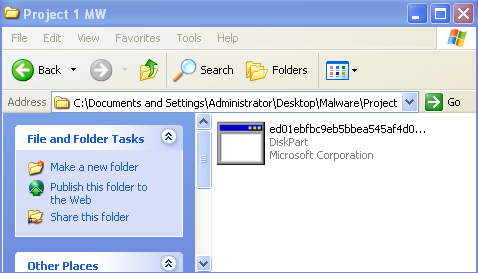
Description automatically generated

**2- Malware Collection Phase:**

Of course, we need to take a snapshot before downloading the file so we could go back to it after we are done analyzing, or if something goes wrong.



This is the file we have

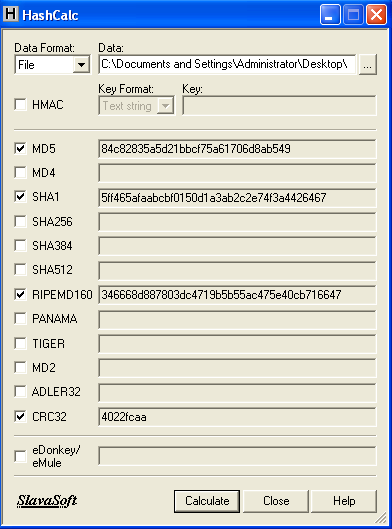


**3- Static analysis phase:**

In this phase, we are going to use various tools that give some information about the malware without actually running it.

1. HashCalc, md5deep, Winmd5: first, we want to get the “fingerprint” of the file, we can use HashCalc or md5deep to get the hash of the file.

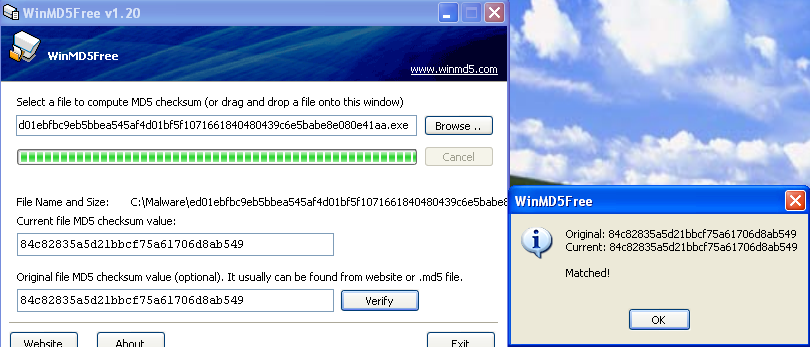
Using HashCalc tool, we got these hash values from the file, each hash value is calculated using a different algorithm.



Using md5deep tool to get the md5 hash just to be sure



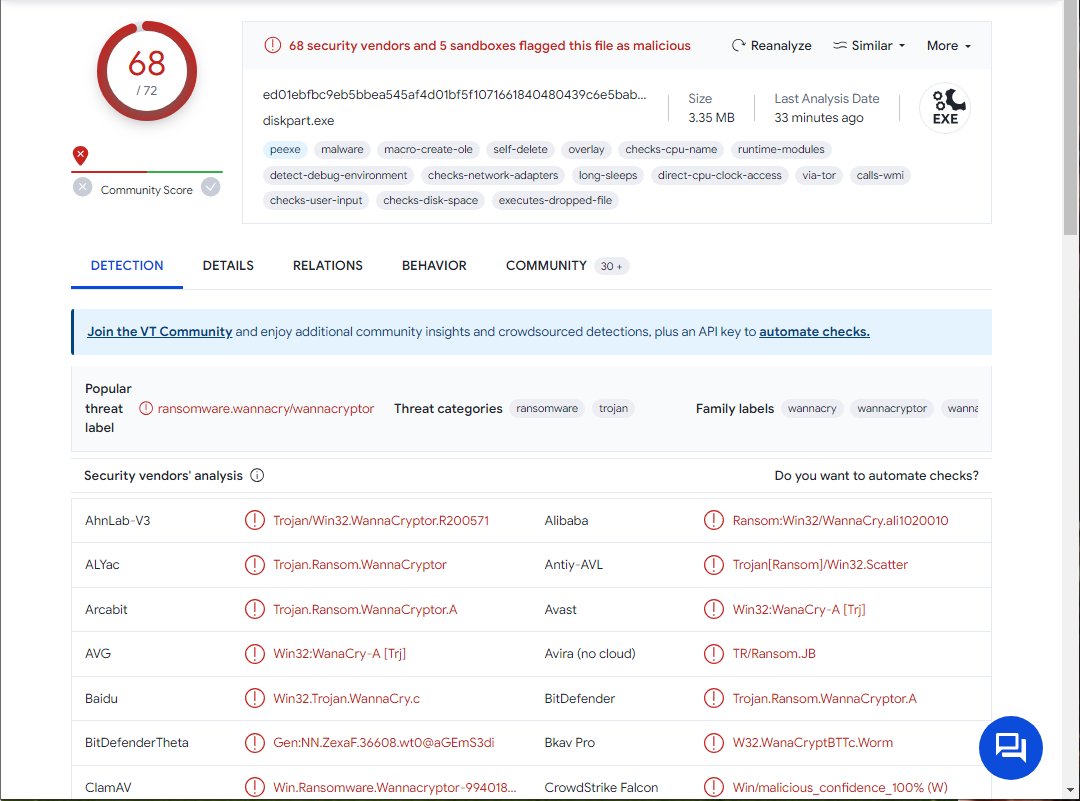
Using Winmd5 and checking if the previous hash matches.



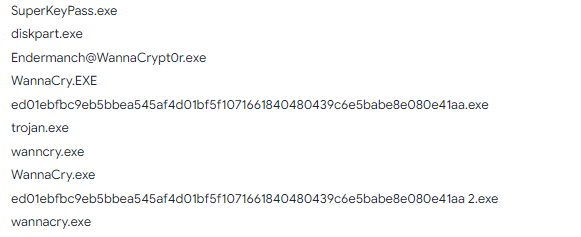
Now we have the md5 Hash of the file: 84c82835a5d21bbcf75a61706d8ab549

2. VirusTotal:

We used the hash to search in VirusTotal, and this is what we got



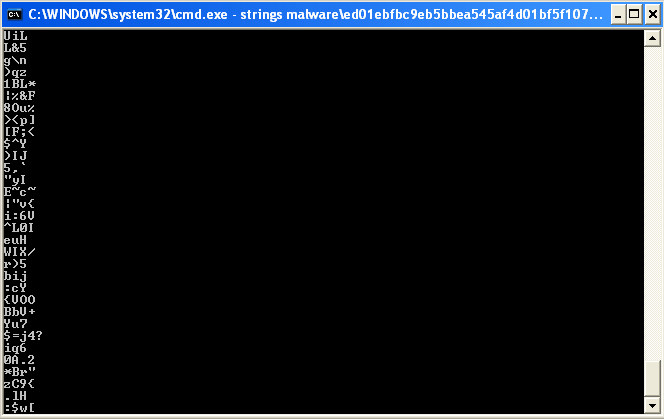
Out of 72 antivirus programs / Security vendors, 68 flagged the file as malicious, which means that it is definitely malicious.

These are other names of the file:

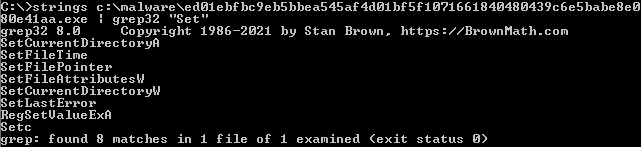
3. Strings, Grep, Bintext:

We used these tools to find strings in the program

First, we used strings tool:



We got thousands of strings, most of which were meaningless, so we downloaded grep tool to help us look for some keywords.



We got these interesting results, let’s see what they mean from <https://learn.microsoft.com/en-us/windows/win32/api/winbase/#functions> :

SetCurrentDirectory: Changes the directory for the current process.

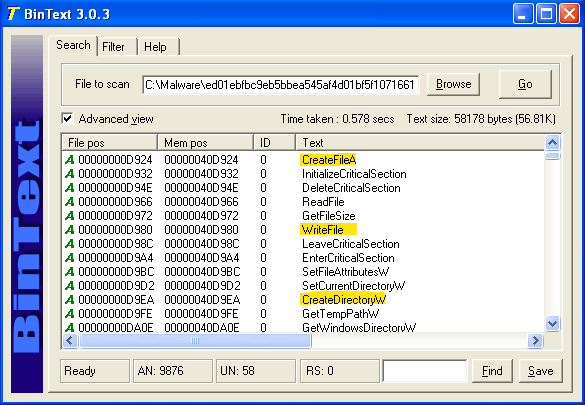
SetFileTime: Changes the timestamp of when a specified file was created, last accessed, and last modified, this can be used if an attacker wants to modify a file, without showing that they did, after modifying the file, they can use this function to set the timestamp to the previous time.

SetFilePointer: Moves file pointer in the file.

SetFileAttributesW: Sets attributes of a specified file.

RegSetValueExA: Sets the data and type of a specified value under a registry key.

Using BinText, I excluded a few characters to narrow down the number of strings shown and I found this.

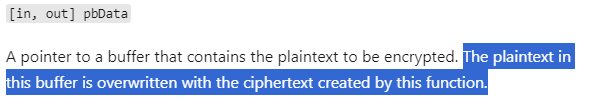


The malware creates and modifies files and directories.

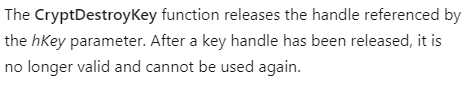


Another interesting finding, it looks like the program generates keys using RSA and encrypts victim’s data using AES. So I decided to read more about these functions.

I found this while reading about the parameters of CryptEncrypt Function.



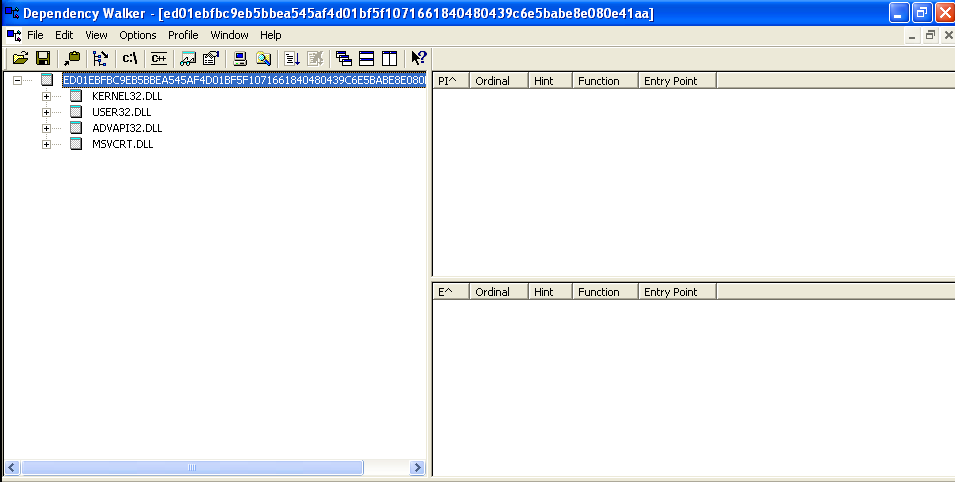
CryptDestroyKey function:



4- Dependency Walker:

This tool shows imported and exported DLLs of a file.

These are the DLLs used by the Malware:



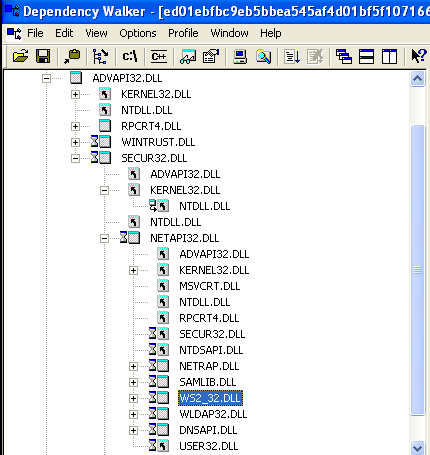
Kernel32.DLL : Contains access and manipulation of memory, files, and hardware.

User32.DLL : This implies that the malware has an interface, because this DLL contains all user-interface components.

AdvAPI32.DLL : Provides access to advanced components like Service manager and Registry.

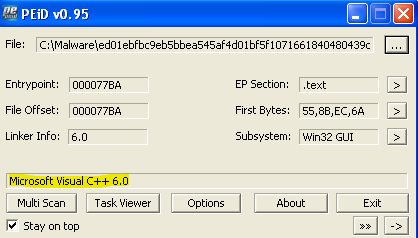
MSVCRT.DLL : Part of MS Visual Studio 6.0

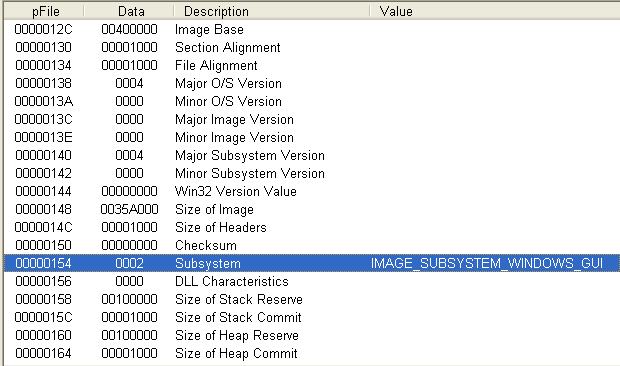
I also found that ADVAPI.DLL uses SECUR32.DLL, which uses NETAPI32.DLL, which uses WS2\_32.DLL, which means that the program likely performs some network related tasks:



5- PEid, PEview:

Shows information gained from PE Header, PEid also detects possible packers, and compiler used, for this malware, Microsoft Visual C++ 6.0 compiler was used





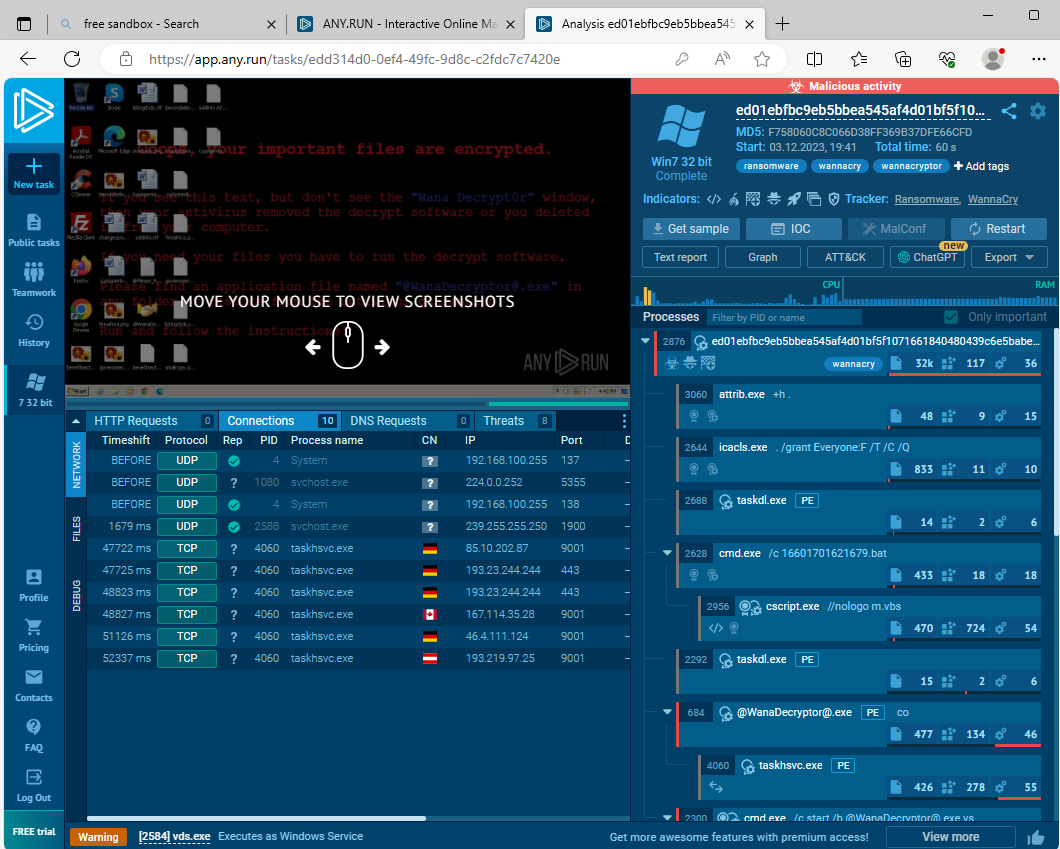
The highlighted part of the previous screenshot means that the malware is a GUI program that runs within the windows system rather than the command window.

**4- Dynamic Analysis Phase:**

It is now time to run the malware, and see what it does live.

1- SandBox:

I used a free online sandbox <https://any.run>



On the right, we can see the processes that are created by the malware, these are some of them:



attrib.exe: this is a command line process, and the option +h sets the hidden file attribute to a file or directory.



Icacls.exe: modifies access control on specified files.

/grant: grants permissions.

/T : performs operation on all files in current directory and all of its subdirectories.

/C : continues operation despite errors.

/Q : suppresses success messages.

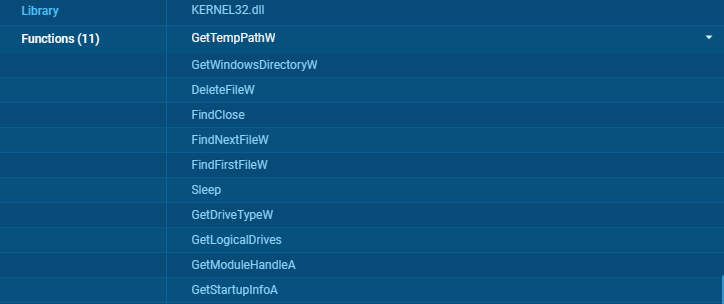


cscript.exe: runs a specified script

/nologo: Specifies that the windows script host banner is not displayed



This is not a windows command or process, so I clicked on PE to see more information

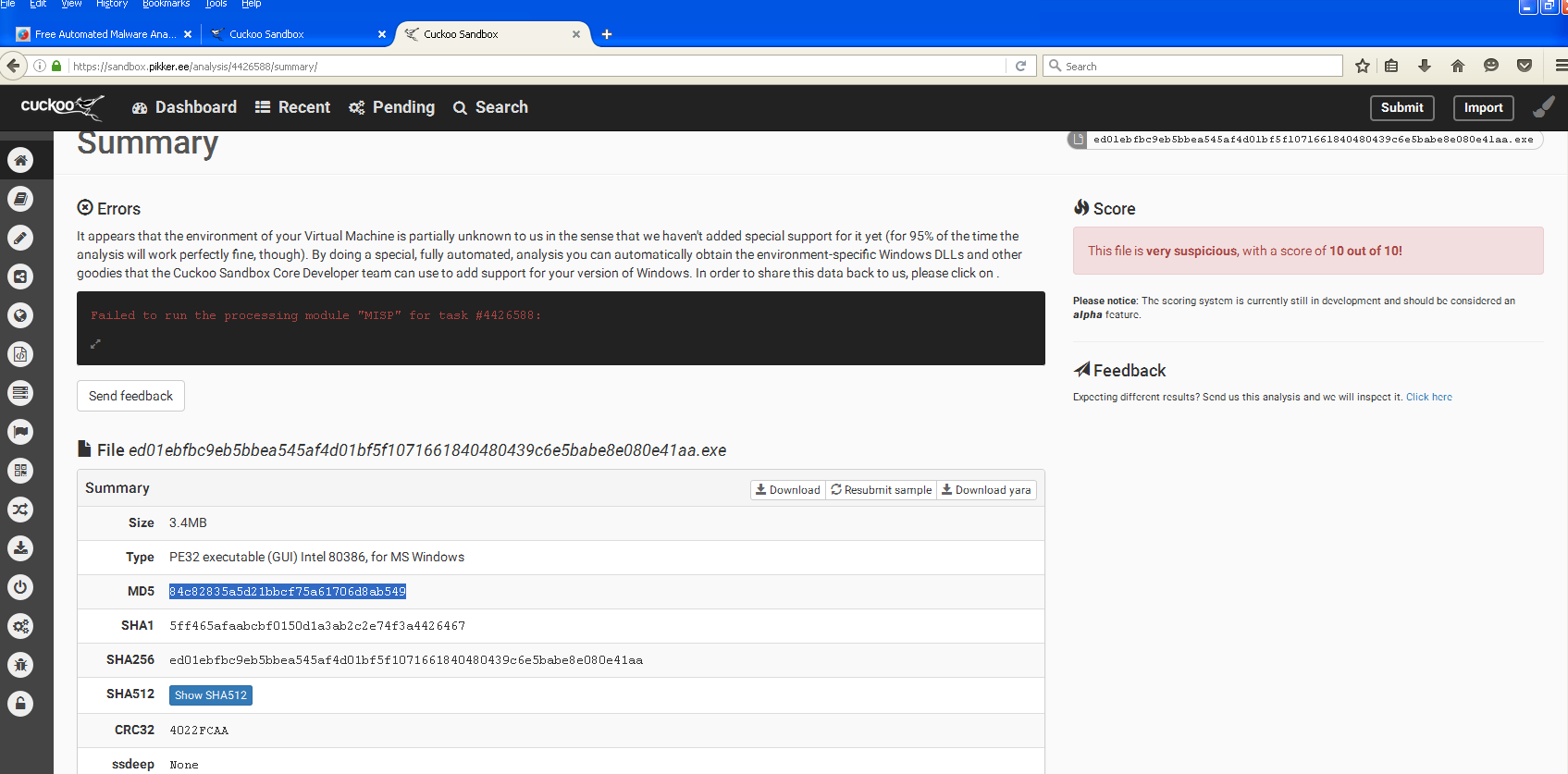


These are some functions that are used by the process.

Another sandbox is Cuckoo Sandbox tool:

We used Cuckoo Sandbox to analyze the malware, providing us with a comprehensive summary of its activities and behavior:

1-As we can see it gave a high suspicion rating of 10/10 and showed us the type of malware and its hashes.



2- Yara rules detected WannaCry malware and now the malware is trying just to explore the environment that it wants to infect by:

a- Finding Hostname:

Figures out the name of the Windows VM, probably to learn more about the target computer.

b- Checking Command Prompt:

Looks at the command prompt to understand the computer setup or to do more actions.

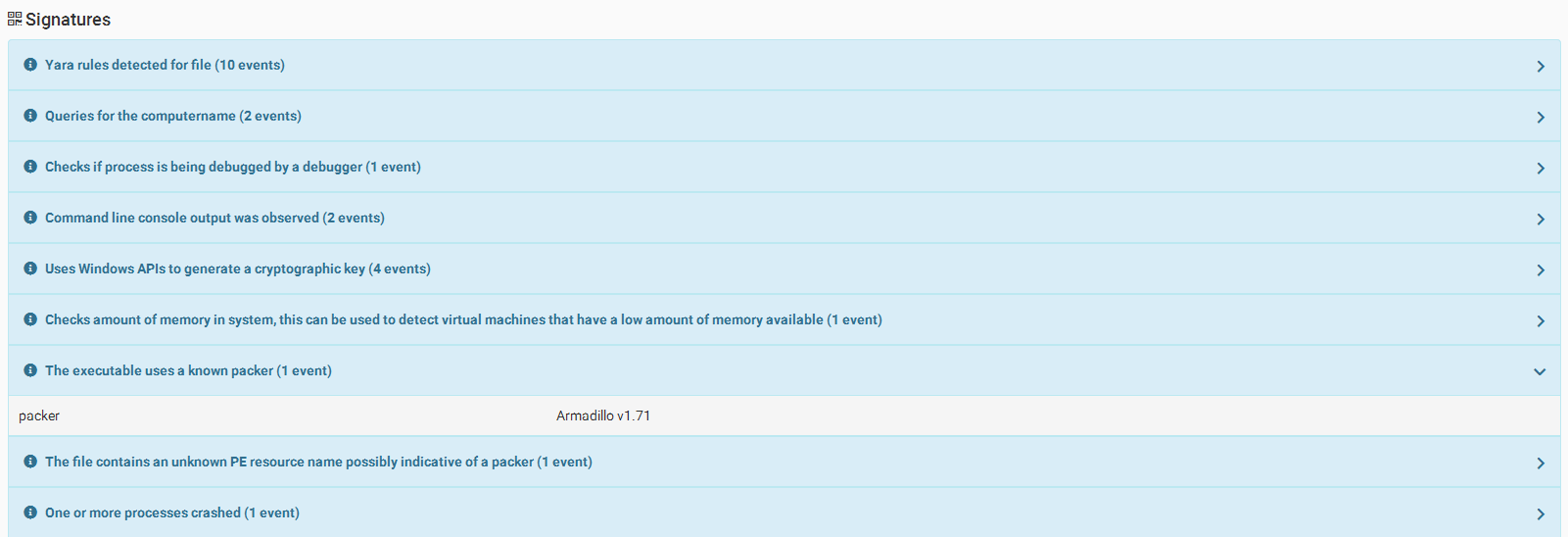
c-Trying to generate a cryptographic key to encrypt the data with later.

d- RAM Check:

-Checks how much memory (RAM) the computer has, which helps the malware work better or understand the computer's abilities.

WannaCry is basically exploring and learning about the infected computer before launching a bigger attack to make it more effective.

All these actions are shown and detected by the sandbox as you can see in the picture:



3- WannaCry is a smart malware. It tries to figure out if it's on a real computer or under analysis in a controlled environment like a sandbox:

1. To slow down analysis, it intentionally introduces delays, making it harder for researchers to study and stop it. (As shown in the picture it delayed analysis time by 284 seconds)
2. The malware also checks how much memory the system has, and tricks like examining disk space (if it is too small for example), to figure out if it's being analyzed. (As shown in the picture it produces API calls to know the disk size).

Although malware analysts and sandboxes can trick malwares by:

\*Using enough memory and disk space

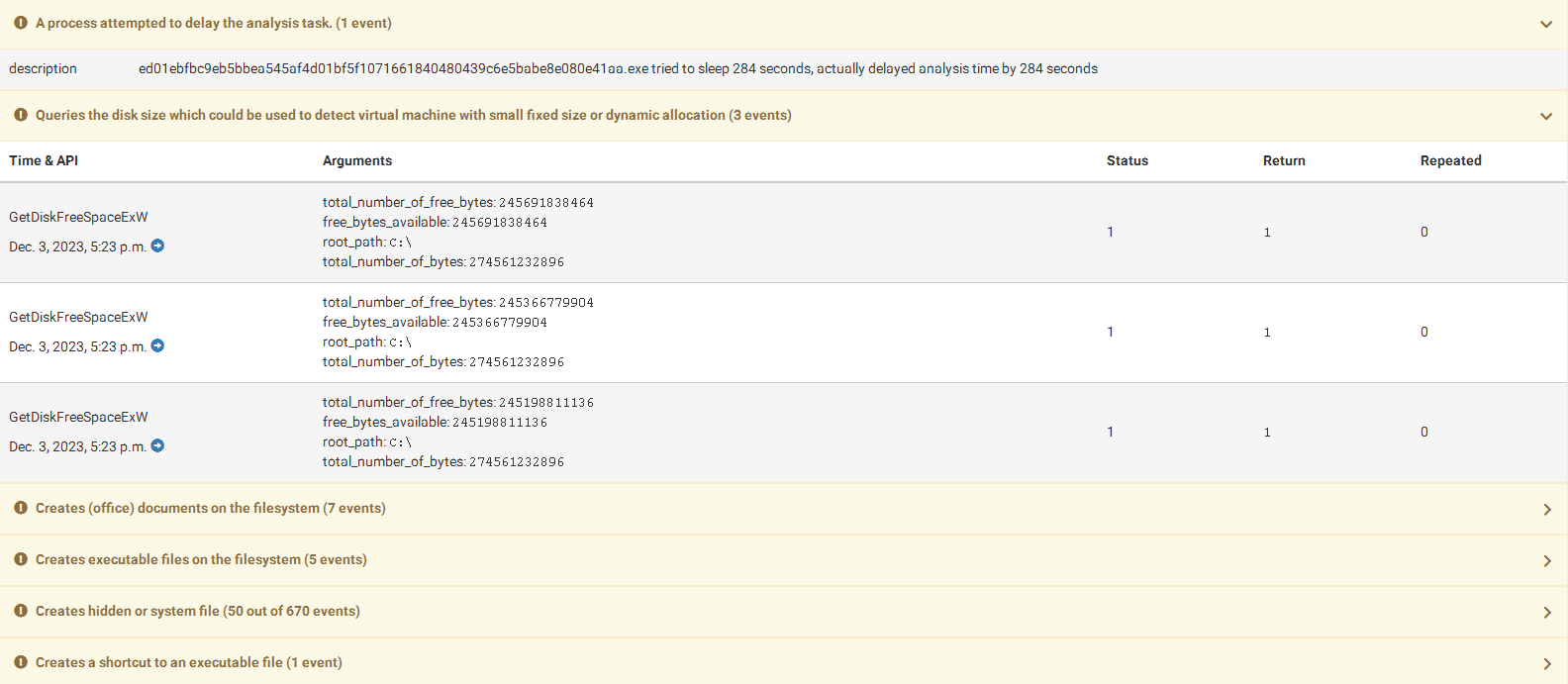
\* Simulating network behavior with tools like FakeNet etc.

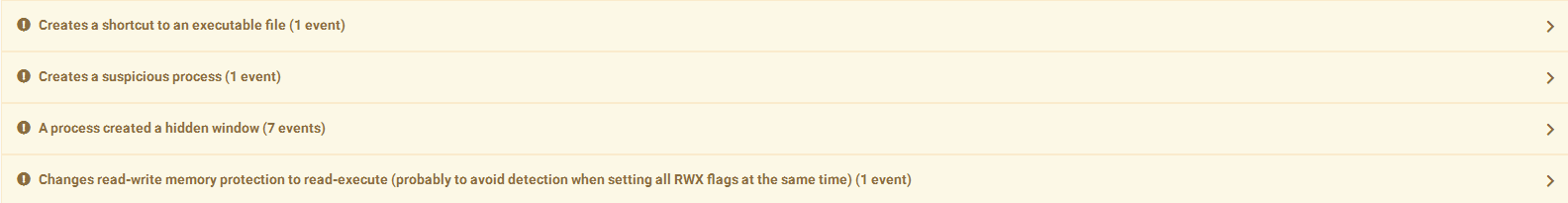
c- Execution of malicious actions:

WannaCry ensures its lasting by creating hidden files and executables on the infected system. This helps the malware remain persistent even after system events or reboots.

d- changing memory permissions from read-write to read-execute:

to avoid being detected and run its code as executable, allowing it to execute its malicious actions more effectively.





3- In this phase the real infection and the disaster starts:

a- WannaCry modifies the filenames of encrypted files by adding a specific file extension to indicate that the files have been encrypted and are subject to ransom.

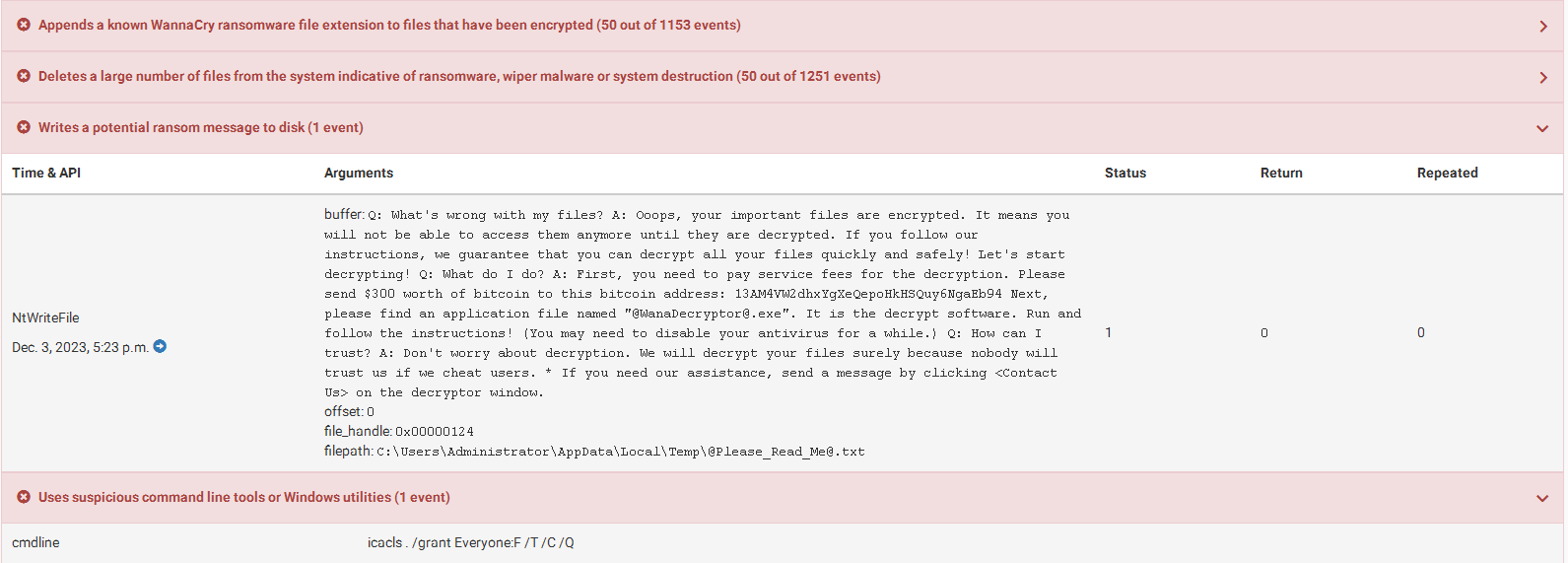
b- deletion of files, a characteristic of ransomware, which aims to destroy or render the system.

c- Writes a potential ransom message to disk (As the message is shown in the picture).

d- The malware uses unusual or suspicious command-line tools or Windows utilities, which is a common behavior of malware attempting to hide its activities or gain control over the system.

e- WannaCry moves files around and probably locking them up in the process. This is a clear sign of ransomware, where files are moved and locked so nobody can use them without paying to unlock them.

f- The malware has been flagged as malicious by a number of antivirus engines.



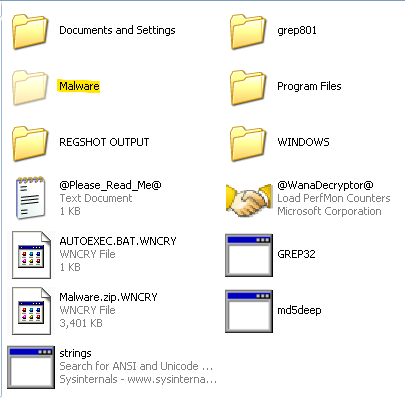


That was a summary about WannaCry malware by Cuckoo sandbox tool, although sandboxes use many tools to check each move the malware makes, so we can have more details, even though sandboxes sometimes miss certain actions of a malware because as we said it is an automated tool and the malware can sometimes choose not to do certain things on purpose if it realizes it's in a sandbox.

2- Procmon:

Now we will run the malware on our own VM, but before doing that, we will run the procmon tool to capture system calls done by the malware.



As I mentioned before, the malware used a command that hides a file/directory, that directory was the one that had the malware.

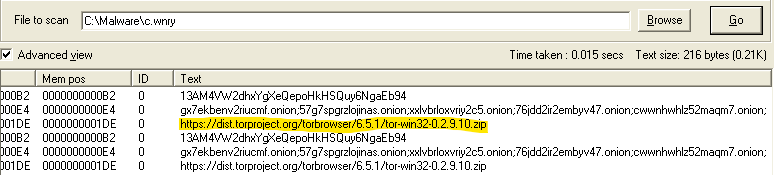
Lets see what files were created by the malware



There are many files, a lot of which are just files that contain the same message but in different languages (The message says that you have to pay the ransom in order to decrypt the files).



This doesn’t give any information about what’s in the file, so I put it in bintext to see what strings does it have.



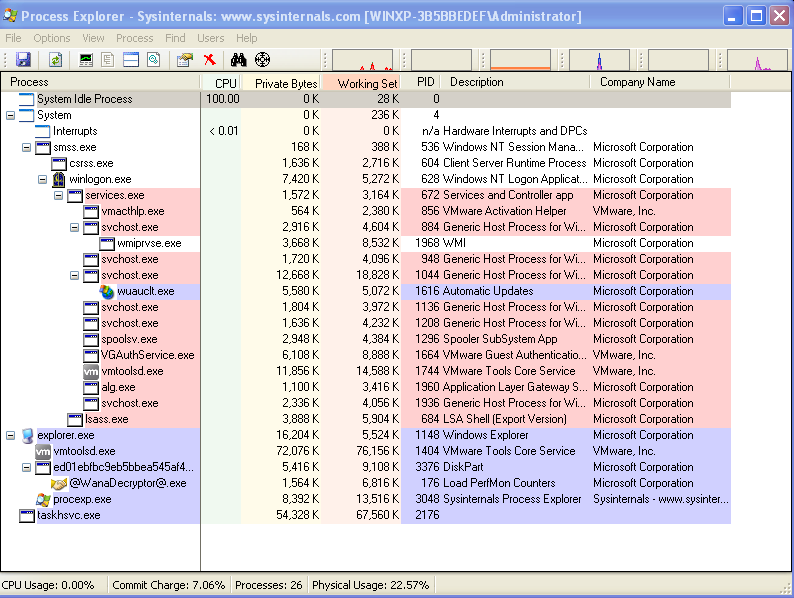
I found this URL.

Most CreateFile operations were just creating encrypted versions of the files which take the original files’ place, and the extension .WNCRY is added.

3- Process Explorer:

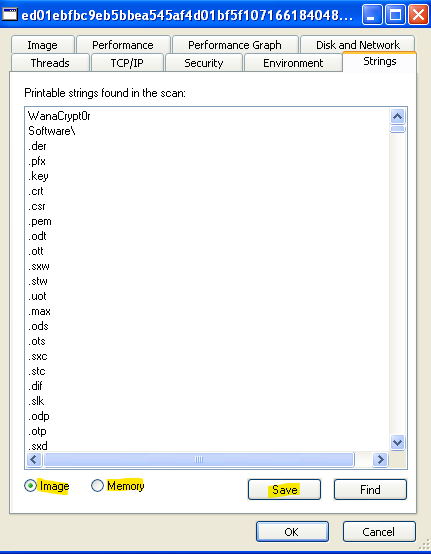
This tool shows current running processes, and a lot of information about them.



I went to see the image strings and memory strings of the original malware file, and to know if the malware is packed, I wanted to compare the strings on disk and in memory.

I downloaded the strings to my host machine

Then I wrote a python code to give me the number of strings in each of them, and see the difference between the both strings



Code:

image=open("mwimagestrings.txt",'r')

mem=open("mwmemorystrings.txt",'r')

imagestr=image.readlines()

memstr=mem.readlines()

image.close()

mem.close()

notinimagestr=[]

memstrlen=len(memstr)

print(f"Number of memory strings : {len(memstr)}")

print(f"Number of image strings : {len(imagestr)}\n\n-----------------------------------------------\n")

if len(imagestr) < 0.8 \* len(memstr):

    print("IMAGE STRINGS ARE MUCH LESS THAN MEMORY STRINGS")

else:

    counter=0

    i=0

    while i < len(memstr):

        found=False

        j=0

        while j<len(imagestr):

            if memstr[i].strip()==imagestr[j].strip():

                found=True

                counter+=1

                memstr.pop(i)

                imagestr.pop(j)

                break

            else:

                j+=1

        if not found:

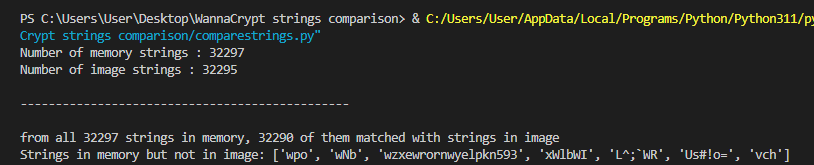
            notinimagestr.append(memstr[i].strip())

            i+=1

print(f"from all {memstrlen} strings in memory, {counter} of them matched with strings in image")

print(f"Strings in memory but not in image: {notinimagestr}")

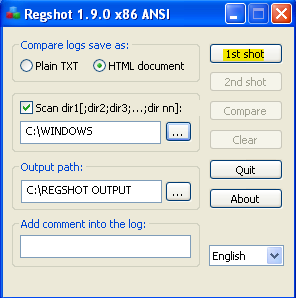
Output:



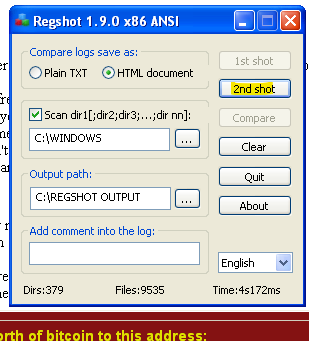
The strings in memory and on disk are almost identical, so the malware is probably not packed.

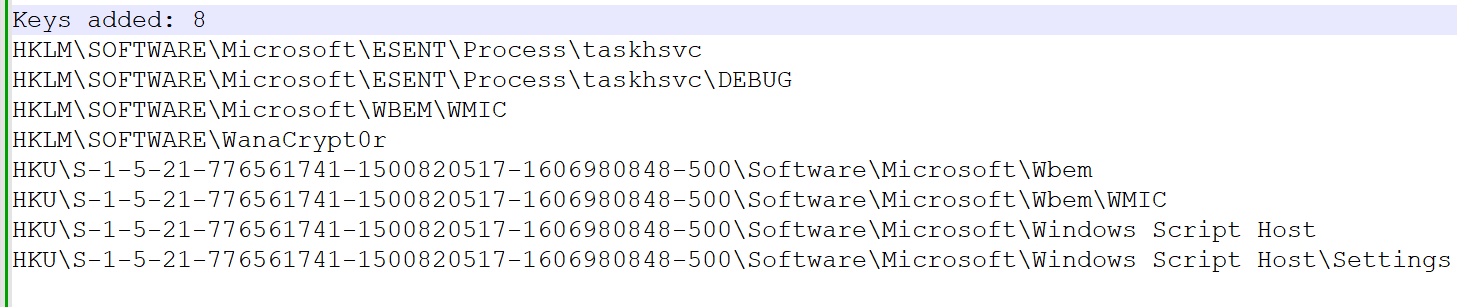
4- Regshot:

This tool shows keys and values added in the registry between two shots, so I reverted back to the snapshot I took earlier, opened regshot, and took the first shot.

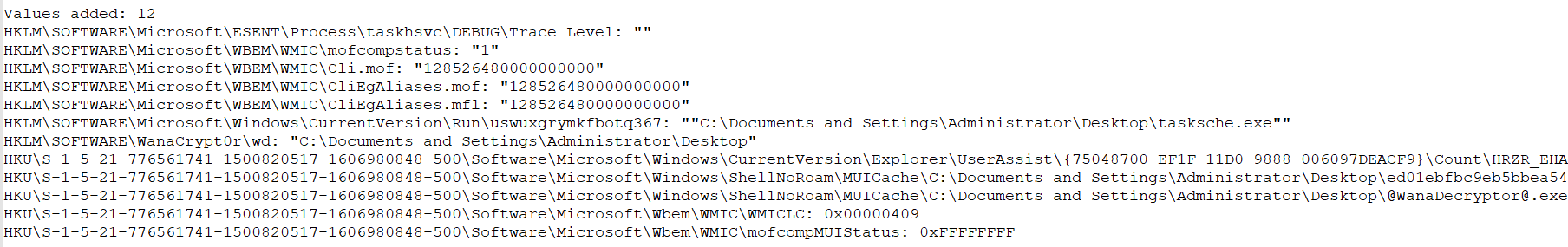


After that I ran the malware then took the second shot.



These are the keys added by the malware:  


Here are values added by the malware:



WannaCrypt creates the following registry keys:

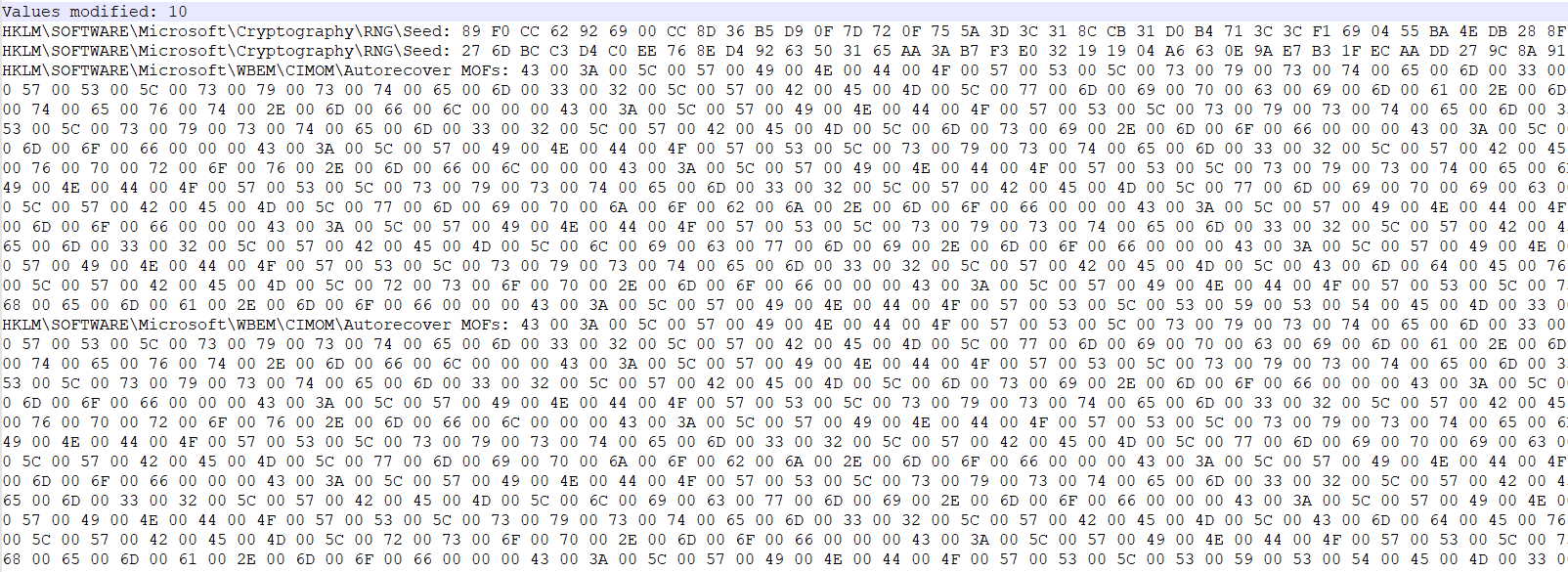
1) HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\\<random string> = “<malware working directory>\tasksche.exe”

For example: HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\uswuxgrymkfbotq367: ""C:\Documents and Settings\Administrator\Desktop\tasksche.exe""

2) HKLM\SOFTWARE\WanaCrypt0r\\wd = “<malware working directory>”

For example: HKLM\SOFTWARE\WanaCrypt0r\wd: "C:\Documents and Settings\Administrator\Desktop"

Here some values modified in the registry.



Most important changes:

**HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed:**

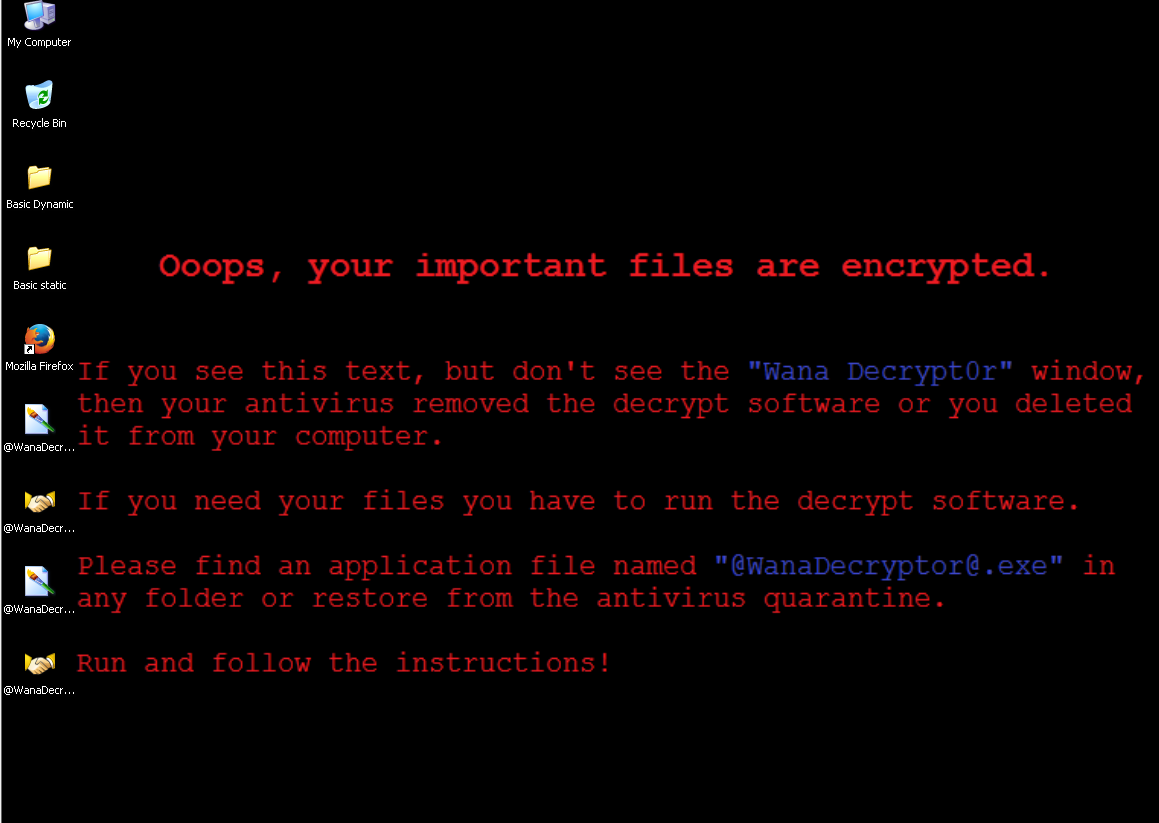
* This registry key stores seed values for the Cryptographic Random Number Generator, which is used to generate cryptographic keys and other security-related functions.

**HKU\S-1-5-21-776561741-1500820517-1606980848-500\Control Panel\Desktop\Wallpaper: "C:\WINDOWS\web\wallpaper\Bliss.bmp"**

* This is the original windows wallpaper

**HKU\S-1-5-21-776561741-1500820517-1606980848-500\Control Panel\Desktop\Wallpaper: "C:\Documents and Settings\Administrator\Desktop\@WanaDecryptor@.bmp"**

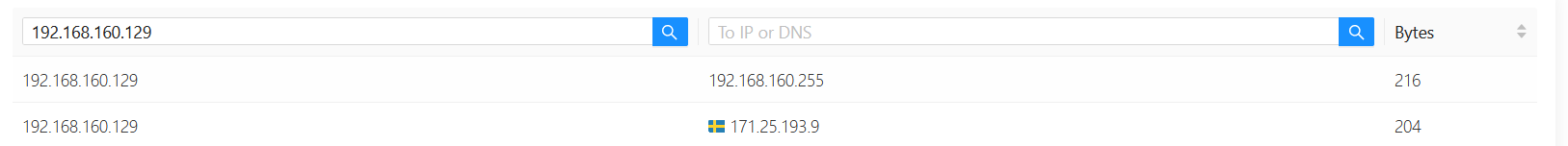
* This is the new wallpaper after modifying the registry key:



It creates the following files in the malware’s working directory:

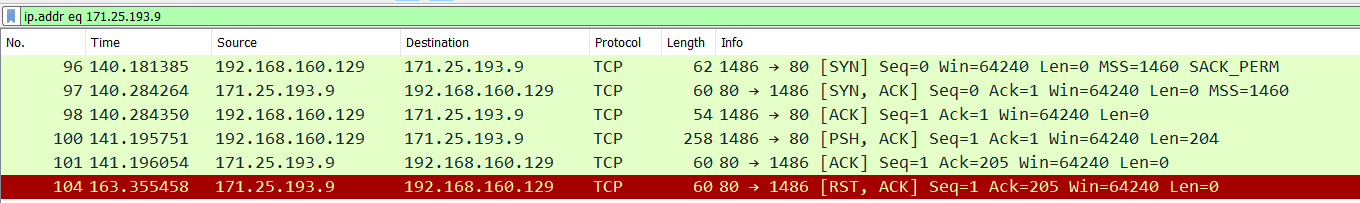
* 00000000.eky
* *00000000.pky*
* *00000000.res*
* *274901494632976.bat*
* *@Please\_Read\_Me@.txt*
* *@WanaDecryptor@.bmp*
* *@WanaDecryptor@.exe*
* *b.wnry*
* *c.wnry*
* *f.wnry*
* *r.wnry*
* *s.wnry*
* *t.wnry*
* *taskdl.exe*
* *taskse.exe*
* *u.wnry*

5- Wireshark:

While conducting network analysis on the running malware, I observed that it connects to the IP address 171.25.193.9 and downloads files onto my device using <https://apackets.com/>. 

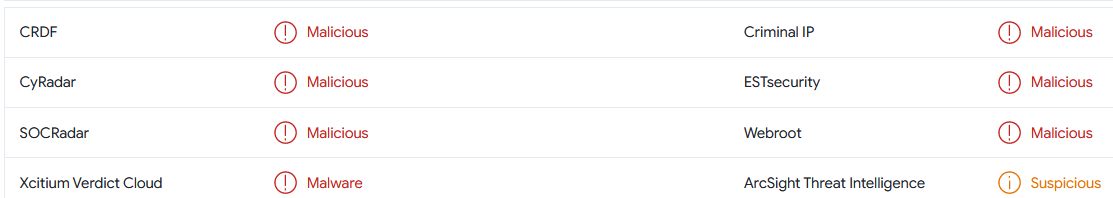
In packet 96, 97, and 98, there is a 3-way handshake between my device and the 171.25.193.9 address.

Packet 100: The second host sends a PSH, ACK packet to the first host. This packet contains actual data (204 bytes) and acknowledges the first host's ACK packet.  
  
Packet 104: The client disconnects [RST, ACK] (terminate) after sending data [PSH, ACK].



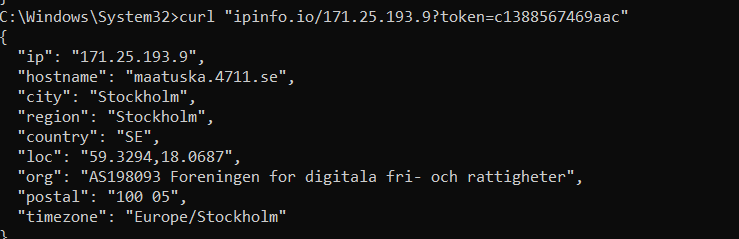
This raised suspicions about the IP address, prompting further investigation. To validate my concerns, I visited VirusTotal.





A user on VirusTotal reported that this IP is associated with the WannaCry malware.

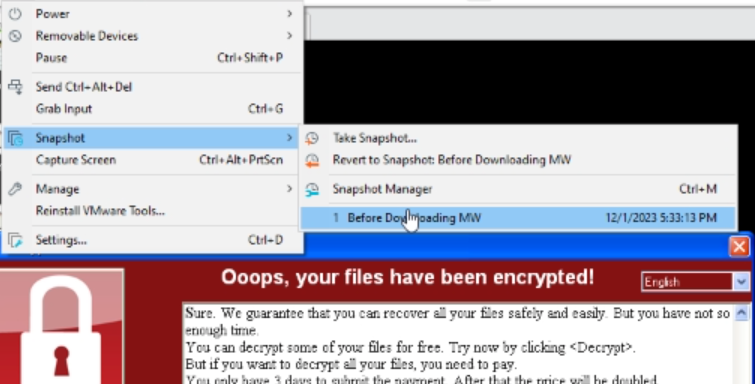
I proceeded to conduct additional investigations to gather more information about this IP address.



Now, I have confirmed that the IP is from Sweden with the following details:

* **IP Address:** 171.25.193.9
* **Hostname:** maatuska.4711.se
* **City:** Stockholm
* **Region:** Stockholm
* **Country:** Sweden (SE)
* **Location:** 59.3294, 18.0687
* **Organization:** AS198093 Foreningen for digitala fri- och rattigheter
* **Postal Code:** 100 05
* **Timezone:** Europe/Stockholm

Finally, we reverted the machine back to the snapshot we took earlier.



Conclusion:

The malware is a ransomware, it hides the working directory of the malware and adds files to it. It encrypts victim’s files, and displays a message demanding a ransom (300$ in bitcoin) in exchange for the original files back. It also alters the timestamp when files were created, modified, and last accessed.

The victim has 3 days to pay the ransom, then it is doubled, and if the ransom is not payed within 7 days, the files can’t be restored.