UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Creación de un entorno controlado de análisis de Malware en Windows. Análisis y catalogación de un malware

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Alcañiz Bautista, Iván Carlo

Tutor/a: Escobar Román, Santiago

Cotutor/a: Ferri Ramírez, César

CURSO ACADÉMICO: 2021/2022

Dedicado a mis padres por apoyarme incondicionalmente toda la vida.

Me gustaría agradecer a mis compañeros de batalla Cristóbal, José Tomás, Sergio, Aitor y Patxi por su colaboración en este TFG.

A Santiago por guiarme en su elaboración y tutorizarlo.

Y a Javi por creer en que podía hacerlo.

Resumen

Con el incremento de amenazas y ataques cibernéticos en estos últimos años, y cobrando más importancia con el aumento del teletrabajo y la educación a distancia, el malware ha obtenido más presencia que nunca. Se presume que diferentes grupos APT (Amenazas Persistentes Avanzadas) cuentan con una buena cantidad de recursos y material en forma de programas informáticos que ellos mismos crean, coordinando a grupos de programadores que pueden crear malware como un servicio (Malware as a Service - MaaS), alquilando su uso, o incluso por encargo y vendiéndolo a terceros, como si se tratara de una fábrica. En este TFG se expondrá la creación de un entorno controlado de análisis de malware y un análisis sobre un fichero que expondrá su funcionamiento interno así como las Tácticas, Técnicas y Procedimientos (TTP) usadas en su ejecución.

Palabras clave: APT, amenaza, malware, análisis, entorno, TTP.

Abstract

With the increase in threats and cyber attacks in recent years, and becoming more important with the increase in teleworking and distance education, malware has gained more presence than ever. It is presumed that different APT groups (Advanced Persistent Threats) have a good amount of resources and material in the form of computer programs that they create themselves, coordinating groups of programmers that can create malware as a service (Malware as a Service - MaaS) , renting its use, or even commissioned and selling it to third parties, as if it were a factory. In this TFG, the creation of a controlled malware analysis environment and an analysis of a file that will expose its internal operation as well as the Tactics, Techniques and Procedures (TTP) used in its execution will be exposed.

Keywords : APT, threat, malware, environment, analysis, TTP.

Tabla de contenidos

Contenido

Tabla de figuras

1. Introducción

En este capítulo introductorio se describen las características que han dado lugar a la escritura de este TFG tales como la motivación y el objetivo. Además del impacto esperado una vez se publique. También se expondrán su estructura y convenciones usadas en su escritura.

1.1. Motivación

En estos últimos tiempos, se han detectado un aumento de ciberataques a particulares, empresas e infraestructuras críticas como empresas del sector energético [1], sanitario [2], tecnológico… Agravados por el aumento de la teleformación y el teletrabajo debido a la COVID-19 [3], y más recientemente al conflicto ruso-ucraniano que comenzó el 24 de Febrero de 2022 donde Ucrania, Rusia y países afines a los mencionados son participantes, revelando así nuevas amenazas como Hermeticwiper [4], Isaacwiper y Hermeticwizard [5] y como nuevos grupos APT como Chamelgang [6], afín a Ucrania y un aumento de la actividad de algunos grupos como Sandworm [7], atribuido a la Unidad Militar 74455 del GRU Ruso.

Además, el malware evoluciona, cada día que pasa se encuentran nuevos agujeros de seguridad, y sus desarrolladores crean nuevas formas de pasar desapercibido en la ejecución en un sistema, debido a esto, se espera que en un futuro dichos ciberataques evolucionen, tanto en cantidad como en ejecución.

Por ende es necesario que personas que tienen un futuro en ciberseguridad, requieran conocer el funcionamiento de estas amenazas y para ello se tiene que aprender las Tácticas, Técnicas y Procedimientos (TTP) usadas en su ejecución y a partir del resultado de su análisis, crear programas y reglas que puedan remediar los diversos problemas que puedan causar estas amenazas persistentes avanzadas por medio de su malware.

La evolución del malware, cada vez más en auge y sofisticado [8], unido a la poca fama de este área de análisis entre el propio mundo de la ciberseguridad [9], dada su complejidad inicial, en comparación con otras áreas como lo pueden ser “red team” o “blue team”, más conocidas y amigables, nos deja un escenario en el que es difícil ganar.

Para paliar un poco esta desventaja, los analistas de malware suelen incorporar análisis automáticos a sus análisis manuales, que aunque no son totalmente fiables, ya que dos muestras aunque pertenecientes a la misma familia pueden diferir en su comportamiento, permiten al analista hacerse una idea de que es lo que está mostrando su pantalla. Dichos análisis automáticos no sustituyen al análisis manual que puede hacer un analista, se trata meramente de un apoyo al proceso de análisis.

En este TFG, haremos uso de las dos formas de análisis, manual y automática, diferenciando si estamos analizando el binario de forma estática (sin ejecución de código) o dinámica (con ejecución de código), además de proponer un entorno

controlado de análisis que sirva como introducción, algo más amigable a las técnicas del análisis de malware.

1.2. Objetivo

El objetivo principal de este proyecto es facilitar el estudio del proceso de análisis de malware, montando un entorno de análisis y detonación de malware para posteriormente estudiar sus características y poder generar un informe, verificando así si se trata de un malware o, en cambio, de un ejecutable/binario normal.

Al finalizar el montaje del entorno se tendrá un equipo virtual en el que poder analizar binarios y poder hacer análisis estáticos y dinámicos de los mismos.

Al completar el análisis podremos crear un informe con las Tácticas, Técnicas y Procedimientos (TTP) utilizadas y los Indicadores de Compromiso (IOC) encontrados y así poder catalogar el binario analizado.

Además del análisis de la muestra que utilizaremos para hacer el informe final, se proporcionará información sobre análisis parciales de otras muestras para poder comparar características entre ellas.

1.3. Impacto Esperado

Se espera que este TFG ayude a iniciar más fácilmente a personas que quieran dedicarse al trabajo de analista de malware, dado que normalmente al iniciarse para estudiar este tipo de trabajo, se comienza por estudiar ingeniería inversa de binarios.

La ingeniería inversa (reversing), consiste en el proceso de aplicar en un producto final un proceso para intentar extraer sus componentes base, en el caso de un programa, los componentes base serían las instrucciones en ensamblador que lo conforman. Para conseguir un nivel decente en esta forma de análisis, se necesita tener un nivel de control alto sobre lenguaje ensamblador y muchos años de estudio y experiencias relacionadas con la forma en que los programas funcionan a bajo nivel.

Dada esta dificultad, no es fácil encontrar a gente que quiera estudiar esta rama de la seguridad, ni empresas a las que el ratio entre coste y retorno de inversión les resulte positivo para formar a una persona en este área.

1.4. Estructura

A continuación se presenta la estructura de la memoria:

El capítulo 1 contiene la introducción de la memoria en la que se encuentran la motivación de este trabajo, el objetivo que se persigue, el impacto esperado, la estructura de dicha memoria y las convenciones utilizadas.

El capítulo 2 se expondrá el estado del arte relacionado con la creación de entornos de análisis y con los propios análisis de malware y se realiza una crítica compuesta por el enfoque que puede dar el acercamiento de este TFG al análisis de malware con respecto a la situación actual.

El capítulo 3 contiene los tipos de análisis a los que se verá sometido el binario a analizar, se hace una pequeña introducción al análisis de malware y se comparan las diferentes técnicas.

En el capítulo 4 se exponen las diferentes clasificaciones que se le pueden dar a un malware según sus características generales y se explican en detalle.

El capítulo 5 nos mostrará cómo es la estructura básica de un fichero, porque se divide de esa manera y de que nos sirve esa información.

En el capítulo 6 se presentan las posibles protecciones que nos podremos encontrar en un análisis de un fichero y de cómo repercuten en nuestro contexto.

En el capítulo 7 se muestra una breve introducción de las herramientas que compondrán nuestro entorno de análisis de malware, explicando su función.

En el capítulo 8 se abordará la instalación y configuración de un entorno controlado de análisis de malware, integrando las herramientas abordadas en el capítulo anterior para tener un producto final en el que podamos comenzar a trabajar.

El capítulo 9 completará la configuración del entorno virtual, modificándolo para que sea lo más parecido posible a un entorno físico real, intentando simular su comportamiento para poder pasar desapercibido frente a algunos tipos de detección que poseen algunos malware.

El capítulo 10 mostrará cómo hacer un análisis preliminar del fichero, en este punto se tendrá una idea inicial sobre el fichero que hemos comenzado a analizar.

El capítulo 11 comprende la ejecución de un análisis estático (básico y avanzado) sobre el fichero en cuestión en el que se estudiará que tenemos ante nosotros.

El capítulo 12 comprende la ejecución de un análisis dinámico (básico y avanzado) sobre el fichero en cuestión que usaremos para poder analizar su ejecución en un contexto interno.

En el capítulo 13 se muestra un resumen de las características del fichero analizado, obtenidas mediante los análisis realizados en los capítulos anteriores.

El capítulo 14 da una conclusión a este análisis y permite catalogar el fichero, exponiendo sus métodos de ejecución.

El capítulo 15 ánima a seguir investigando más allá del propósito de este trabajo aportando algunos análisis parciales de diferentes muestras.

El capítulo 16 aporta un posible uso práctico del entorno de análisis a gran escala, en combinación con otras herramientas.

El capítulo 17 muestra una serie de mejoras que podrían implementarse en un futuro para mejorar el entorno de análisis.

El capítulo 18 comprende la bibliografía de este trabajo.

Anexo: Funciones de Windows

Anexo: Lista de programas de uso específico

Anexo: Información Extra

Anexo: Estructura PE

Anexo: Objetivos de Desarrollo Sostenible

1.5. Convenciones

- Los comandos utilizados se mostrarán en tipografía *Calibri cursiva 11pts*. Solo se empleará esta tipografía para este contenido.

- Las clasificaciones Malware, Goodware, Programa/Aplicación potencialmente no deseado/a, Troyano… Se mostrarán con sus acrónimos. En el capítulo 4 se encuentra una taxonomía referente a estos acrónimos.

2. Estado del Arte

En este capítulo se expone la situación actual con respecto al análisis de malware y la creación de entornos de análisis, así mismo se expone la solución que pretende dar este TFG al problema dado.

2.1. Estudio estratégico y Crítica al estado del arte

El análisis de malware está fuertemente relacionado con la ingeniería inversa y el lenguaje ensamblador, pero solo es necesario en los análisis estático y dinámico avanzados. Libros de análisis de Malware como “Practical Malware Analysis” y cursos gratuitos en inglés en ingeniería inversa [10] [11] y español [12] se enfocan en la parte más técnica del proceso, la ingeniería inversa, que consiste en leer y modificar las instrucciones en ensamblador una por una para poder elegir el camino de ejecución que sigue el programa. Aunque los libros si tienen capítulos referentes a otras cosas que no son la ingeniería inversa, como lo son las redes y el forense, el reversing tiene mayor repercusión.

Este punto de vista hace creer que se tiene que empezar obligatoriamente por hacer reversing a los ejecutables, aprenderse todo lo referente al lenguaje ensamblador y saber cuándo una rutina cifra o descifra datos en memoria para poder ejecutarse y, si bien es un punto que tarde o temprano tendremos que aprender e incluirlo en nuestro proceso de análisis y aunque no hay que descuidarlo, no necesariamente tiene que ser el punto de partida, sino que puede ser incluido y mejorado posteriormente.

```
00401441 > 8A0433      MOV AL,BYTE PTR DS:[EBX+ESI]
00401444 . 34 CC       XOR AL,0CC
00401446 . 88043B      MOV BYTE PTR DS:[EBX+EDI],AL
00401449 . 43          INC EBX
0040144A .^ E2 F5      LOOPD SHORT virus.00401441
```

Figura 1: Rutina (loop) de cifrado XOR de un programa en ASM

Comenzar a aprender a analizar malware por medio de aplicaciones de monitorización de procesos, editores hexadecimales y en definitiva, diversas herramientas que muestren de forma legible a un ser humano las características de un binario puede dar lugar a una comprensión más asequible y estimule más su interés a una persona que comienza a iniciarse en este área de la seguridad informática.

Por otro lado, si bien hay una amplia documentación sobre creación de entornos para análisis de malware, esta documentación se basa mayormente en Cuckoo Sandbox, que ofrece un análisis automatizado o REMnux, una distribución Linux orientada a la ingeniería inversa. Lo cual no es objetivo de este TFG, en el que se intenta crear un entorno de análisis bajo Windows, que permita el mejor análisis manual posible adecuado a nuestras circunstancias y amigable con los que se intentan iniciar en esta ciencia.

Otra parte diferenciable es que en la mayoría de artículos y trabajos en internet sobre la creación de entornos, aunque es cierto que se usan VM hay poca información sobre

la ocultación de sus características para intentar pasar la VM por una máquina real, algo que aunque no es vital para el análisis de malware, si es recomendable dado que algunos intentan detectar si se están ejecutando en una VM, y si están en esa casuística, no se ejecutan o bien ejecutan un programa como se ejecutaría normalmente sin llamar a las rutinas referentes a la ejecución del MW, necesitando un hardware real para poder analizar este tipo de MW. En nuestro caso, intentaremos hacer pasar la VM por una máquina física, modificando las huellas digitales que el software VM deja en el sistema virtualizado.

2.2. Propuesta

La solución para el caso de estudio que se planteará más adelante pasará por crear un entorno controlado de análisis utilizando Virtualbox y MV Windows que nos permitirá tener un punto de partida con el cuál iniciar un análisis de malware, además de un procedimiento que explicará cómo poder analizar un binario Windows sin hacer un uso excesivo del lenguaje ensamblador, analizando sus características forenses y su ejecución en el sistema.

3. Tipos de análisis

En este capítulo se expondrán los cuatro tipos de análisis existentes, divididos en dos categorías: Básico/Avanzado y Estático/Dinámico. Los cuales, interactúan entre sí, de manera que no podemos encontrar una definición de una categoría sin las dos definiciones correspondientes de la otra y viceversa.

3.1. Básico Estático

Consiste en examinar el archivo ejecutable sin ver las instrucciones reales que lo componen. Solo sirve para confirmar si un archivo es efectivamente malicioso o proporcionar algo de información sobre sus funciones. Es rápido y sencillo pero puede ser ineficaz contra MW sofisticado y se pueden pasar por alto comportamientos relevantes para el análisis. Programas como DIE, CFF Explorer, Resource Hacker… se encargan de esta parte del análisis.

3.2. Básico Dinámico

Implica ejecutar (detonar) el MW y observar su comportamiento en el sistema. Para ello se necesita configurar un entorno que permita ejecutar el malware de forma segura sin dañar el sistema principal o la red en la que se encuentre. Según con qué tipo de binario nos encontremos, éste se podrá ejecutar libremente o en cambio necesitara de un cargador como rundll32 o regsvr32, de los cuáles hablaremos más adelante. Total Uninstaller y Process Monitor son los programas que usaremos en esta fase.

3.3. Avanzado Estático

Este tipo de análisis consiste en hacer ingeniería inversa del MW, cargando el ejecutable/binario en un desensamblador y estudiar sus instrucciones para ver que hace el programa. Dichas instrucciones son ejecutadas por la CPU, así que este tipo de análisis muestra lo que hace realmente el programa, en cambio, requiere de conocimientos especializados que el análisis básico y se tarda más en aprender. IDA, Ghidra, dnSpy e IDR son los programas que se podrán usar en esta fase.

3.4. Avanzado Dinámico

En este análisis se usa un depurador (debugger) para examinar la ejecución de un MW paso por paso. El entorno ejecutará el MW

instrucción por instrucción y se podrá ver los cambios en tiempo real que ocurren en el sistema a nivel interno. X64dbg, x32dbg, OllyDbg y los debuggers de IDA y Ghidra se pueden usar en esta fase.

4. Taxonomía

En este capítulo se comentan las clasificaciones que pueden aplicar a un programa según sus características.

4.1. GW: Goodware.

Se denominará de esta forma a los programas que no presentan comportamiento malicioso en el sistema.

4.2. PUP: Potential Unwanted Program

Programas potencialmente no deseados.

Si bien no es un MW en como tal, si que puede presentar comportamientos cuestionables en su ejecución en un sistema, como descargar ficheros externos a su código (Downloader/Dropper) o mostrar anuncios cada cierto tiempo en pantalla (Adware). Puede hacerse pasar por un programa legítimo, que si bien puede hacer cambios en el sistema, aunque posteriormente reversibles, las consecuencias de su instalación pueden ser molestas.

4.3. MW: Malware

Programas maliciosos, este tipo de programas buscan dañar el equipo o la información contenida en él.

- Virus (Virus): Denominación genérica. Cualquier programa que altera el funcionamiento normal de un componente informático con fines maliciosos.
- Gusano (Worm): Programa que se replica. Puede colapsar sistemas o redes.
- Troyano (Trj): Programa en apariencia inofensivo que permite abrir una brecha de seguridad en el sistema otorgando un acceso remoto.
- Keylogger: Crea un log con las pulsaciones de teclas que teclea el usuario.
- Stealer: Se encarga de robar credenciales o datos comprometidos, enviándolos a un servidor externo.
- Banker: Roba datos relacionados con banca online, como números de cuenta, números de tarjeta de crédito/débito, CVVs…
- Ransomware (Ransom): Se encarga de cifrar los datos de un ordenador, haciéndolos inaccesibles para el usuario, y solicitando un rescate normalmente en criptomonedas como Bitcoin o Monero, como pago por la clave para descifrarlos.

Un MW puede encajar en una o más categorías, dado que puede que no tenga comportamientos propios de un solo tipo. Se podría encontrar un MW que se encargase de guardar las pulsaciones de teclas (Keylogger), detecte los datos relacionados con banca online como usuario, contraseña, número de

tarjeta/cuenta bancaria y CVV (Banker) y envíe esos datos a un servidor externo (Stealer).

5. Estructura de un binario

En este apartado se muestra como es la estructura básica de un fichero PE. Con ella tendremos un breve acercamiento a como es un binario internamente.

5.1. Estructura PE

- Cabecera
  - Cabecera DOS (DOS Header): Es el encabezado de los .exe de DOS (16 bits), se fue adaptando a la evolución del sistema y se agregó información para poder usarla en 32 bits, posteriormente se le añadió más información para los sistemas de 64 bits.
  - Cabecera PE (PE Header): Esta es la primera cabecera que encontramos en 32 bits, la cual tiene un tamaño de 18h bytes.
  - Cabecera opcional (Optional Header): Esta cabecera se llama 'opcional', pero realmente solo es opcional para algunos archivos como los "objetos" (no ejecutables), para el resto de los ejecutables tiene información muy importante.
  - Tabla de secciones (Sections Table): Detalle de las secciones que componen nuestro programa. Esta tabla está compuesta por 'entradas', las cuales se encuentran una a continuación de la otra y cada una ocupa 28h bytes. Estas entradas contienen el detalle de cada sección.
- Secciones
  - Código (Code): Código del programa contenido en una sección
  - Importaciones (Imports): Dependencias, funciones que solicita a librerías externas (propias o del sistema)
  - Datos (Data): Información necesaria para la ejecución del programa

A su vez las cabeceras, directorios, tabla de secciones… tienen sus propios apartados [13], a los cuáles no vamos a entrar mucho en detalle, pero es recomendable tener una idea general. Dichas estructuras están explicadas más en detalle en el Anexo "Estructura PE".

5.2. Otras características

Existen otras características que forman su propia estructura. No entra en disonancia con la estructura PE expuesta anteriormente, sino que es otra forma de estructurar el ejecutable y también debe de sernos familiar, ya que es posible que la veamos durante el proceso de análisis.

- Compiler: Compilador utilizado para compilar el código de un programa.
- Linker: Enlazador usado para combinar los archivos base y crear el ejecutable
- Packer/Protector/Crypter: Diversas protecciones que puede presentar un ejecutable

6. Estructura de protecciones de un binario

En este capítulo se presentan las posibles protecciones que nos podremos encontrar en un análisis de un fichero y cómo éstas repercuten en nuestro contexto.

6.1. Packer

Aunque comúnmente se les llama a todos “packers”, realmente un packer/compresor es un software que comprime el código de un programa y se desempaqueta en memoria cuando se ejecuta el “archivo empaquetado/comprimido”. Normalmente sirven para que un programa ocupe menos espacio (como Winrar o 7zip), además de permitir que el usuario no tenga que descomprimir previamente el ejecutable, a esto se le llama “compresión ejecutable”. Originalmente no nació con el propósito de ocultar MW, pero hoy en día, con las velocidades en internet y la gran capacidad de almacenamiento, cuando se ve algún uso de un packer, probablemente se trate de un archivo malicioso.

Hay muchos packers de uso habitual entre el malware, entre ellos:

- UPX
- ASPack
- PECompact
- WinUpack
- FSG
- BobSoft (Delphi)
- MPress
- ASPack

6.2. Crypter

Un crypter se encarga de cifrar el contenido de un programa, dificultando así su comprensión y análisis por parte de los motores de antivirus. Dado que los AV solo escanean memoria física y no programas en ejecución, el Crypter se aprovecha de este comportamiento para intentar pasar un malware por un programa benévolo.

Un crypter se divide en dos partes, un Builder, que es el programa con GUI que se usa para el proceso y el Stub, que es un trozo de código que se une al archivo original con las instrucciones necesarias para descifrarlo en tiempo de ejecución.

6.3. Protector

El protector se centra en dificultar el análisis del fichero con técnicas antidebug y antivirtualización. Suelen ser de pago, usados generalmente para proteger GW de posibles modificaciones, aunque también pueden ser usados por MW. Themida, VMProtect, Enigma Protector y ASProtect son ejemplos de protectores.

Para analizar los binarios que poseen protector normalmente se opta por hacer un análisis estático y dinámico básicos, ya que para poder hacer los análisis avanzados requeriría desprotegerlo y es una tarea si no imposible, bastante costosa tanto en cantidad de tiempo como de recursos y no sería efectivo.

Algunos ejemplos son:

- Themida
- VMProtect
- Enigma Protector
- ASProtect

7. Configuración del Laboratorio

En este capítulo se muestra una breve introducción de las herramientas que compondrán nuestro entorno de análisis de malware, explicando su función brevemente.

7.1. Software a utilizar

- Virtualbox [14]: Software de virtualización
- Máquina virtual con Windows 7 o Windows 10
- FlareVM [15]: Suite de seguridad informática sobre Windows
- IdaFree [16]/Educational/Home/Pro: Desensamblador
- dnSpy [17]: Depurador .NET y editor ensamblador
- Total Uninstaller [18]: Instalador/Desinstalador de programas
- Resource Hacker [19]: Editor de recursos de aplicaciones Windows
- Detect-It-Easy (Die) [20]: Programa para determinar tipos de archivos
- Process Monitor [21]: Aplicación de monitorización de procesos
- CAPA [22]: Detecta capacidades en archivos PE
- FLOSS [23]: Descifra cadenas (strings) de un binario cifrado

7.2. Alternativas

- Qemu: Software de virtualización https://www.qemu.org/download/#windows
- Indetectables Toolkit: Kit de herramientas de seguridad orientada a análisis de malware https://github.com/indetectables-net/toolkit
- Wine: Reimplementación de la API de Windows para Linux https://www.winehq.org/
- Ghidra: Desensamblador desarrollado por la NSA https://ghidra-sre.org/

7.3. Recursos externos

- IpVoid: Web que permite geolocalizar y medir el nivel de reputación de una IP https://www.ipvoid.com/
- UrlVoid: Web que permite geolocalizar y medir el nivel de reputación de una URL https://www.urlvoid.com
- VirusTotal: Portal que proporciona análisis de archivos a través de antivirus. https://www.virustotal.com Se ha solicitado a VirusTotal una licencia educativa, dicha licencia es válida durante 6 meses y añade datos adicionales a la consulta de muestras que no son accesibles para un usuario no registrado.

8. Creación del Laboratorio

En este capítulo se aborda la instalación y configuración del entorno de análisis, integrando las herramientas expuestas en el capítulo anterior.

8.1. Requisitos mínimos del Host

- S.O Host: Windows 7/10
- HDD: 1TB
- RAM: 8GB

8.2. Requisitos máquina virtual

- S.O: Windows 7
- Almacenamiento: 120 GB
- RAM: 2GB

8.3. Configuración de las máquinas virtuales

Usaremos las MV con licencia temporal que Microsoft permite utilizar para desarrollo disponibles en sus páginas oficiales [24], en caso de necesitar alguna otra configuración, podemos recurrir a Adguard [25] que permitirá descargar diversas máquinas virtuales desde los servidores oficiales de Microsoft.

En nuestro caso partiremos de la configuración “IE11 on Win 7” que actualmente se encuentra en las versiones de desarrollo de Microsoft Edge, para la plataforma Virtualbox.

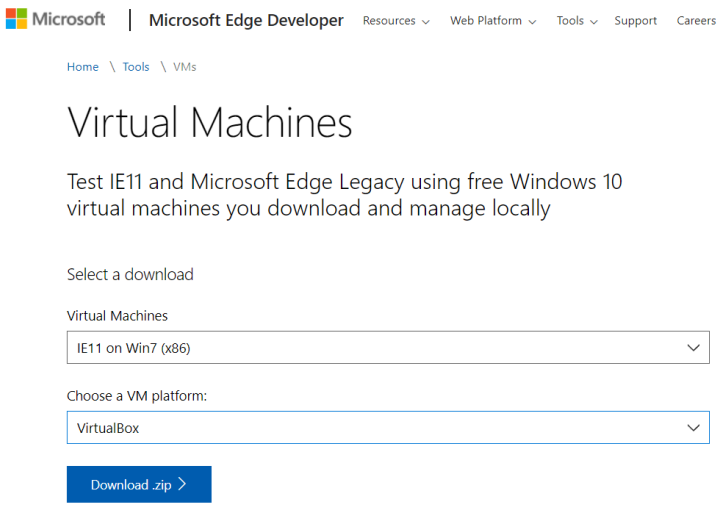


Figura 2: Descarga de VM desde el sitio oficial de Microsoft

Además de la MV de Windows 7, es recomendable contar con una MV de Windows 10, ésta se puede encontrar en la página de descarga de MVs para desarrollo que proporciona Microsoft, con el nombre de "MSEdge on Win 10". Tener dos MVs con sistemas operativos distintos, con la misma configuración, permite poder ejecutar software que en la otra máquina no se puede ejecutar, bien por incompatibilidad o porque el software comprueba en que Sistema Operativo se está ejecutando (posible MW dirigido).

Una vez descargada, haremos una copia y la importaremos en Virtualbox. Para poder utilizarla, habrá que tener siempre una copia de la máquina e ir generando "Snapshots" para poder volver a estados anteriores a la detonación de un MW y así recuperar un entorno seguro de análisis.

Figura 3: Máquinas virtuales importadas en Virtualbox

Las VM de Microsoft tienen un periodo de expiración de 90 días, después de eso, no tendrán soporte oficial y saldrá un mensaje en pantalla de "licencia no original" para extender ese tiempo de prueba haremos uso del comando "*slmgr –rearm*" el cual nos hará un reset de la licencia, se puede usar hasta 3 veces en Windows 7. En Windows 10 no se puede utilizar.

Las VM están configuradas con 4GB de memoria RAM, si queremos utilizar más de una a la vez con fluidez con nuestra configuración actual, hay que bajarle la asignación de memoria a 2GB.

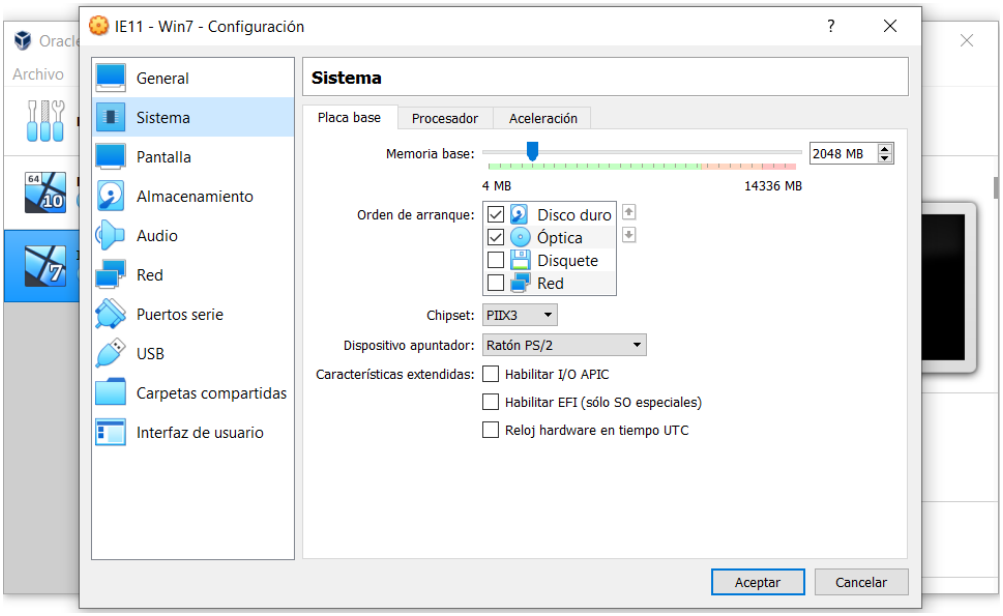


Figura 4: Asignación de memoria RAM

Dado que las VM también están configuradas con 40GB de espacio en disco necesitaremos aumentarlo porque para poder instalar FlareVM necesitaremos al menos 60GB, para evitar problemas de espacio y evitar análisis de espacio que a veces pueden hacer los MW para detectar una máquina virtual, aumentaremos el espacio a 120GB con la herramienta de administración de Virtualbox (VBoxManage).

Hay que remarcar que este comando solo funciona si tenemos reservado el espacio del disco en “dinámico” (por defecto los discos se crean de esta forma en VirtualBox), si son asignados en formato “fijo” este comando no funciona [26].

La sintaxis es: *VBoxManage.exe modifyhd <Archivo.vdi> --resize <Tamaño en MB>*
En nuestro caso: “*C:\Program Files\Oracle\VirtualBox\VBoxManage.exe” modifyhd “IE11 – Win7-disk001.vdi”--resize 122880*

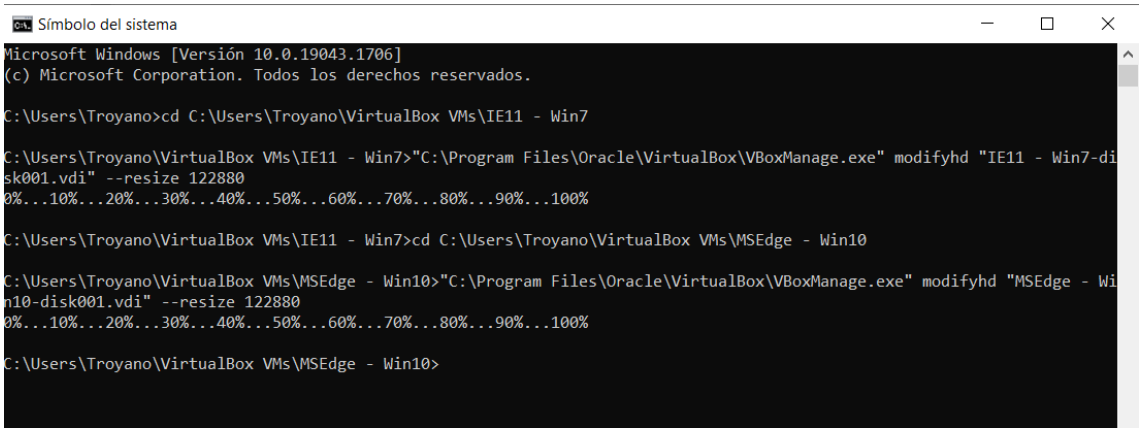


Figura 5: Redimensión de disco virtual

Aunque ya hemos redimensionado los discos, posteriormente a arrancar las VM de Microsoft hay que redimensionar las particiones en el propio sistema operativo puesto que se configuraron con 40GB, para ello usaremos el administrador de discos del sistema.

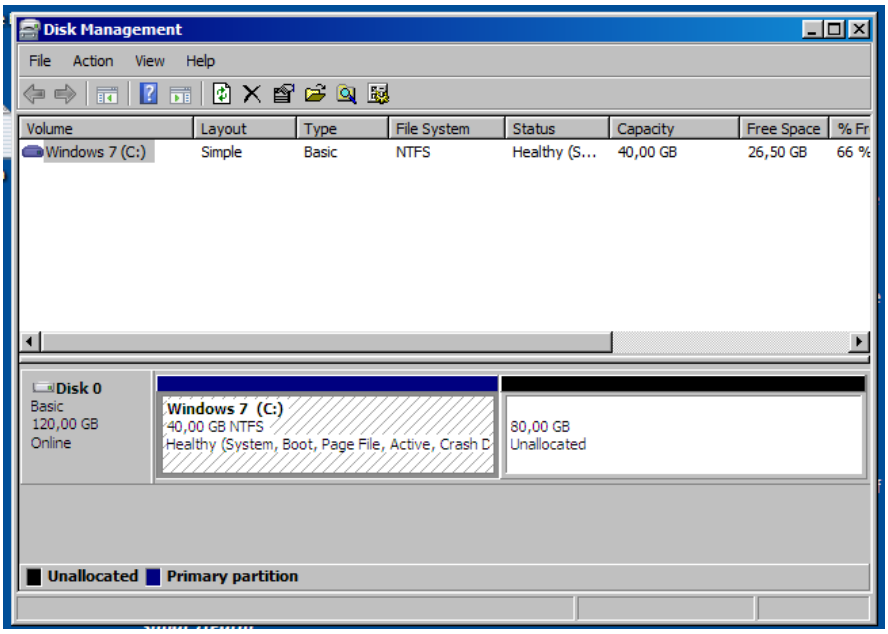


Figura 6: Administrador de discos

Haremos clic con el botón derecho, y en el menú contextual elegiremos “Extend Volume”

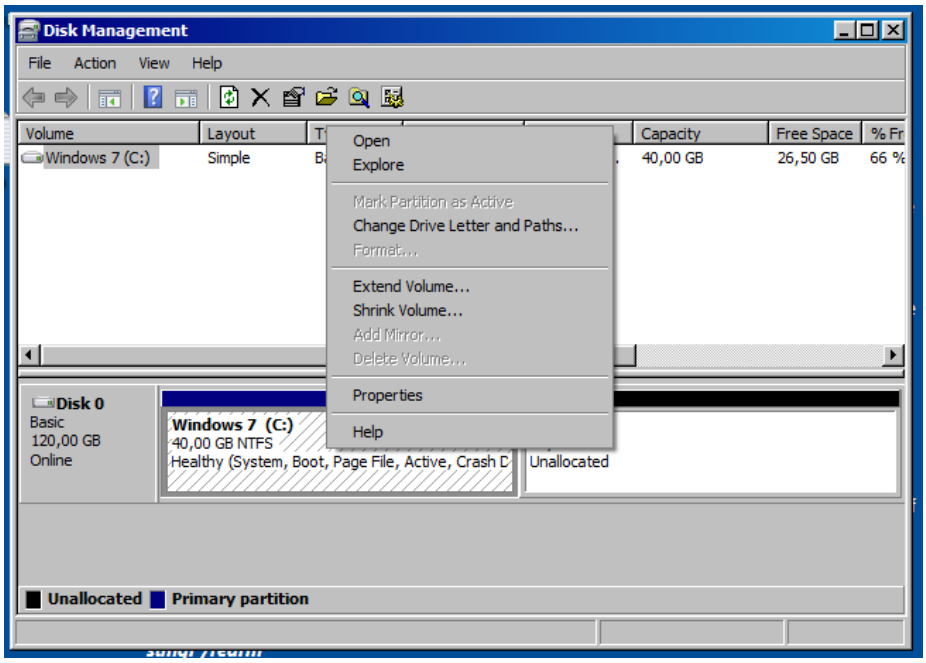


Figura 7: Opciones del menú contextual del administrador

Nos aparecerá un asistente que nos guiará por el proceso de extensión del disco para poder aprovechar todo el espacio extra.

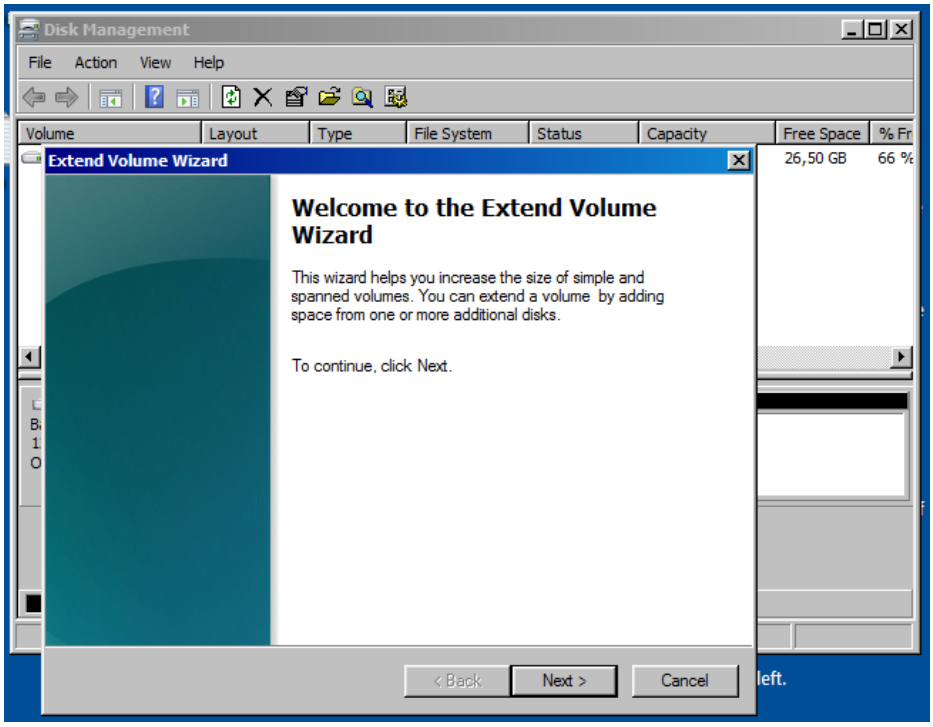


Figura 8: Asistente de extensión de volumen

En el paso siguiente nos mostraran los discos en los que podremos extender el espacio disponible, en nuestro caso solo es uno, Disk 0, con espacio vacío de 81919MB, seleccionado por defecto.

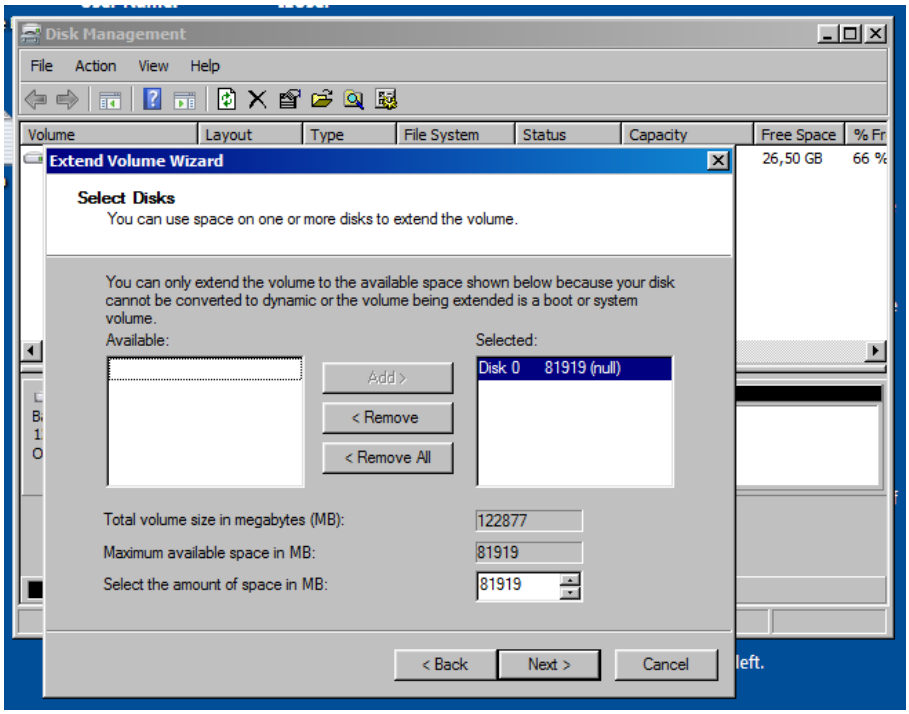


Figura 9: Selección de partición vacía

Al hacer clic en "Next" se extenderá el espacio de la partición y finalizará el asistente.

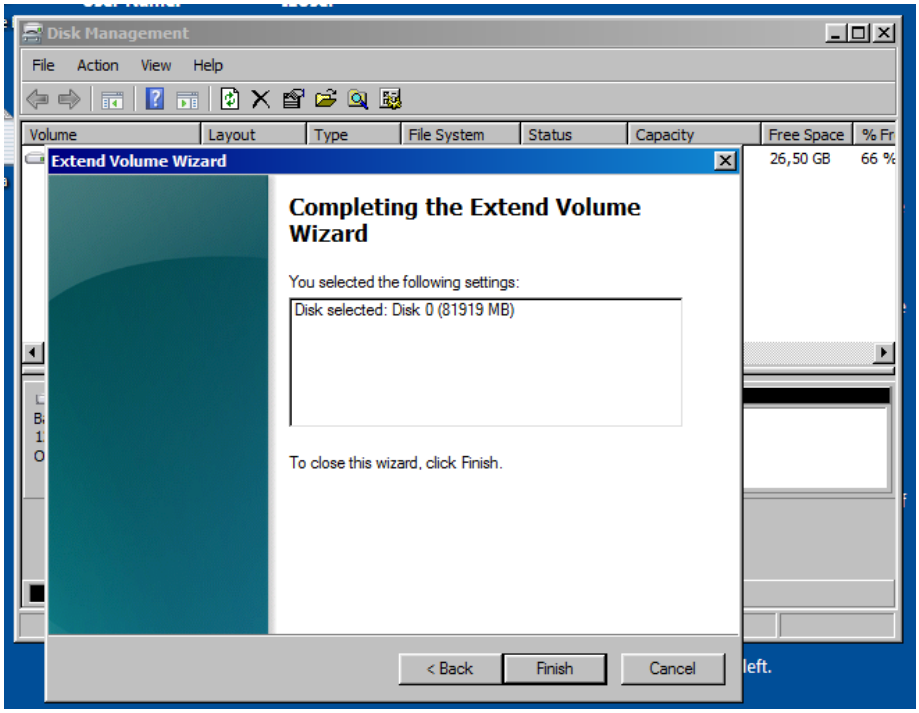


Figura 10: Finalización del proceso

Al finalizar el proceso, estaremos en el mismo paso que en la Figura 7, con la diferencia de que ahora tendremos una partición completa de 120 GB en vez de dos distintas de 40 GB y 80 GB.

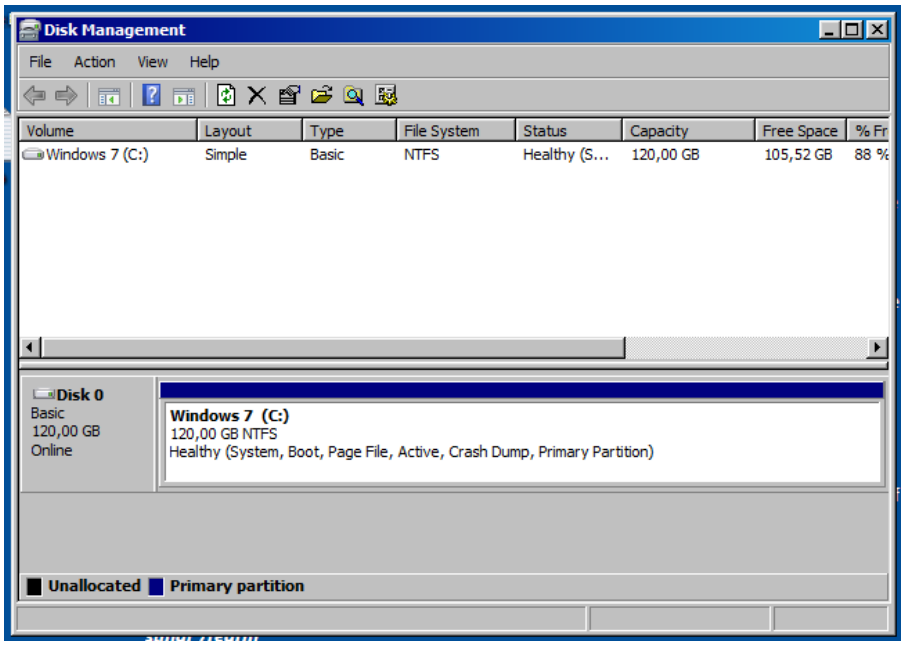


Figura 11: Disco extendido

Después de redimensionar la partición podremos utilizar todo el espacio disponible y proceder a la instalación de la capa de personalización FlareVM [27] que nos servirá para montar el entorno de análisis de malware en Windows.

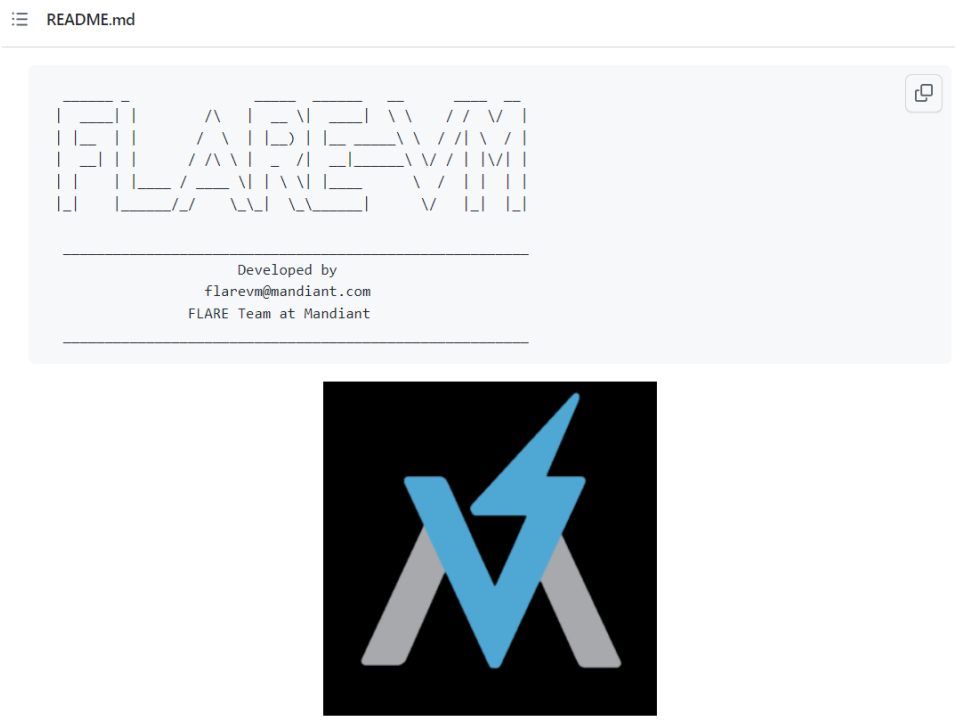


Figura 12: Github de FlareVM

Como necesitamos que la VM no tenga ningún tipo de protección para que el antivirus no nos borre las muestras, se deshabilitará Defender de Microsoft.

Para deshabilitarlo bastará con ejecutar un programa [28] que nos desactive Microsoft Defender, aunque también puede desactivarse de forma manual desde las herramientas de configuración. Se ha preferido usar la herramienta automática por facilidad y rapidez del despliegue, sobre todo en caso de que se tenga que hacer en múltiples MV o equipos.

Descargamos Defender.remover.exe, hacemos clic con el botón derecho y en el menú contextual elegiremos “Run as Administrator”

Al ejecutarlo se nos mostrarán posibles opciones a elegir, introduciremos “*Y*” y pulsaremos “*Enter*” para eliminar Windows Defender del sistema.

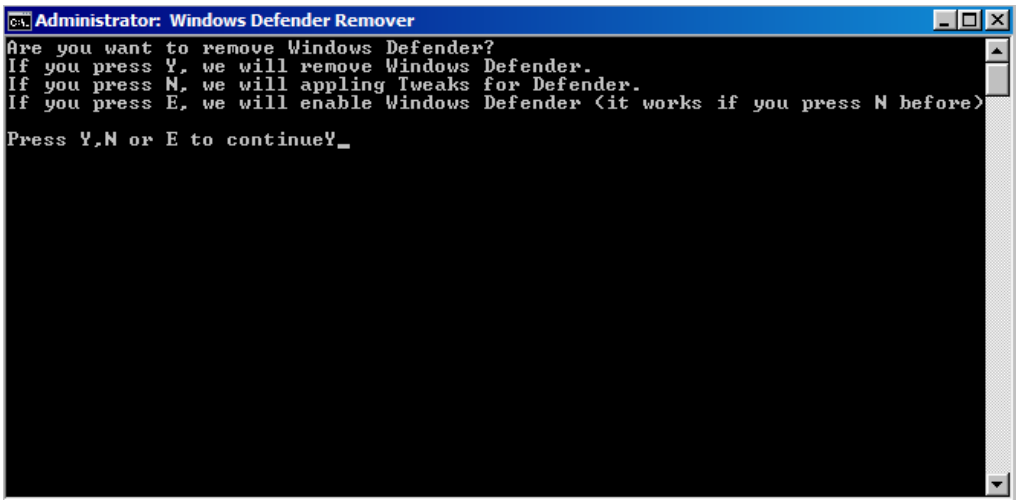


Figura 13: Selección de opción

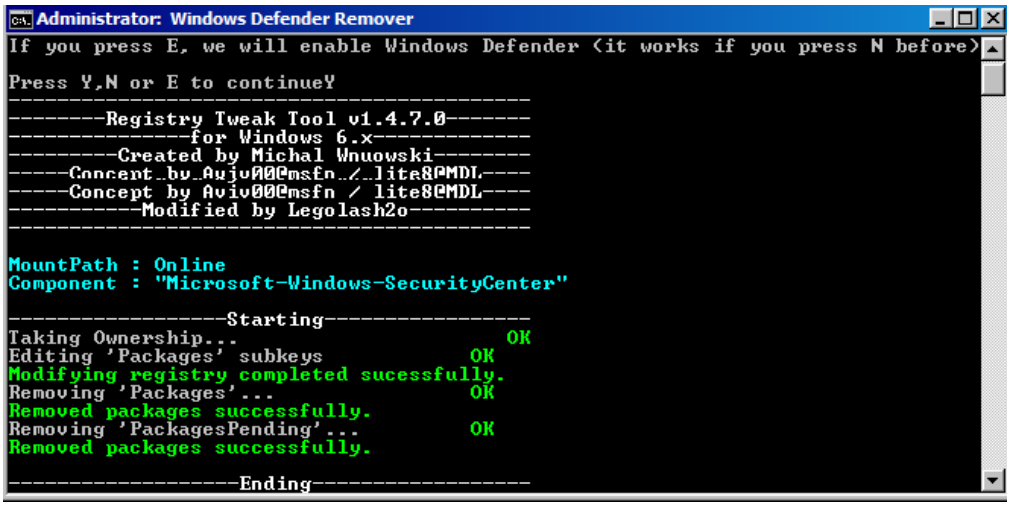


Figura 14: Defender Remover en ejecución

Al final del proceso, nuestra VM se reiniciará, sin ningún rastro de Defender de Microsoft.

Una vez hecho esto, hay que instalar .NET 4.5 [29] y WMF 5.1 [30] desde los servidores oficiales de Microsoft, una vez hecho podremos instalar la capa de personalización FlareVM que tiene Mandiant en su Github [31], siguiendo las instrucciones de su instalador (en inglés)

Al usar las MV de Microsoft, .NET 4.5 ya se encuentra instalado, así que solo hay que instalar Windows Management Framework 5.1.

Ejecutaremos la actualización Win7-KB3191566-x86.msu para instalar esta característica.

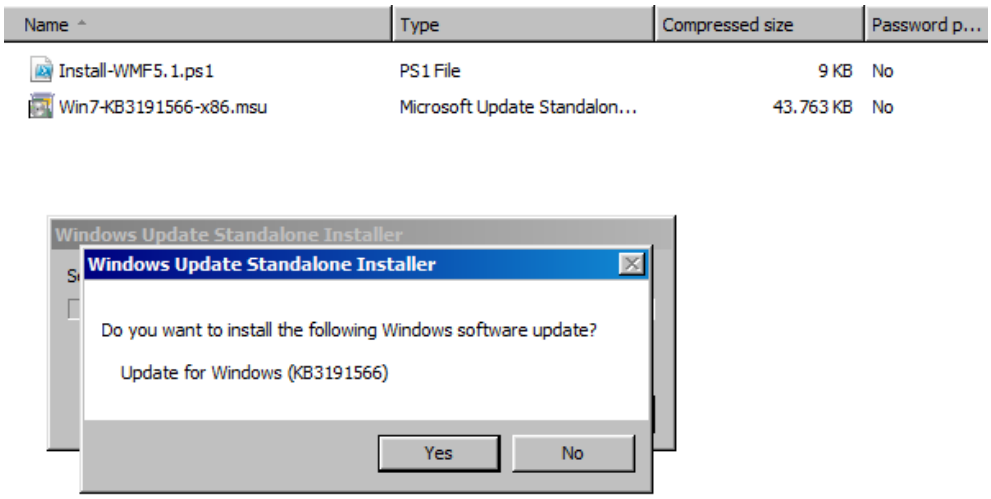


Figura 15: Ejecución de la actualización Win7-KB3191566-x86.msu

Reiniciamos la máquina y procedemos a instalar FlareVM, descargaremos su "install.ps1" y abriremos una ventana de Powershell como administrador y habilitaremos la política para ejecutar scripts sin firmar con "*Set-ExecutionPolicy Unrestricted*" con opción "A" para iniciar el proceso.

Ejecutamos el script de instalación con "*.\install.ps1*" y cuando nos pregunte si queremos instalarlo, usamos la opción "*R*".

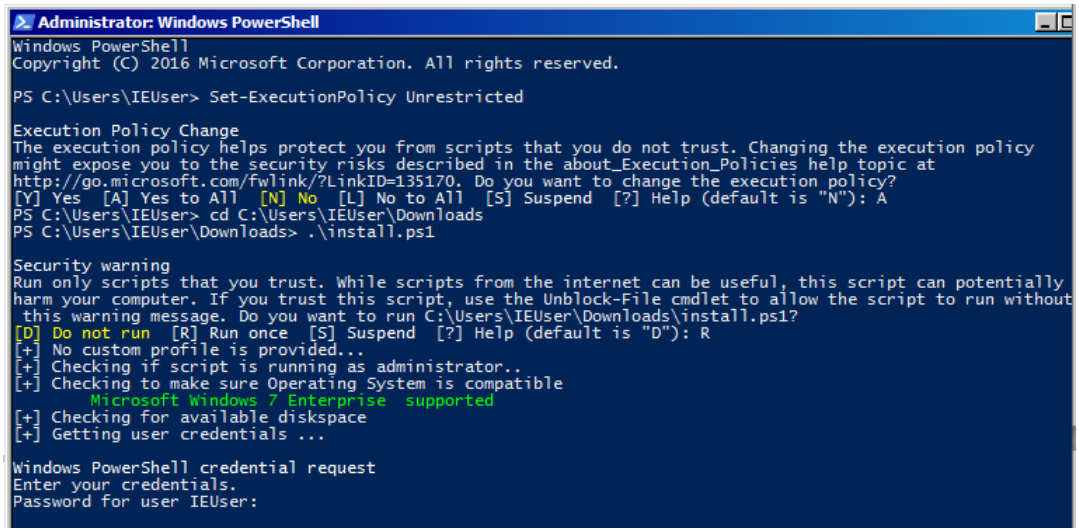


Figura 16: Ejecución de powershell, habilitación de política de ejecución, Opción "*R*" y solicitud de credenciales

Una vez introducidas las credenciales necesarias, la descarga e instalación de programas continuará. Comenzará un largo proceso de instalación en que la VM podría reiniciarse varias veces.

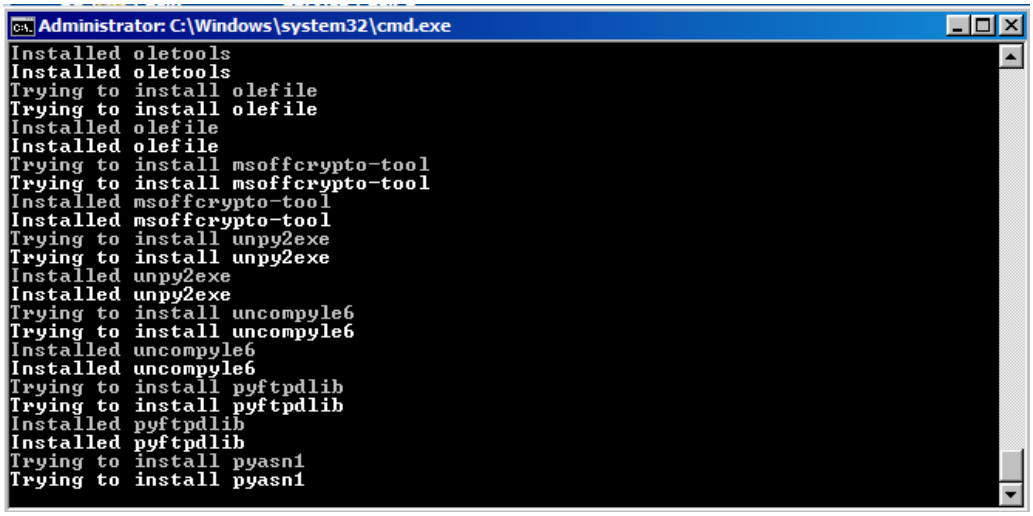


Figura 17: Instalando programas de la suite

De todos los programas que instalará, los que más usaremos serán Detect It Easy (DIE), dnSpy, Total Uninstaller, Process Monitor (procmon) y Resource Hacker, dado que a IDA Free y debuggers solo llegaremos en caso de que no tengamos un veredicto final, en busca de algún rastro más que nos pueda servir para determinar la naturaleza del binario.

FlareVM instala herramientas extra que quedan fuera del alcance de este trabajo, dado que son para usos más específicos como:

- Autopsy, herramienta de análisis digital forense
- Oledump, dedicada a analizar ficheros OLE (Object Linking and Embedding - enlazado e incrustación de objetos), los cuales se encuentran insertados en ficheros de Microsoft Office
- Exe2Aut que se encarga de extraer de un fichero .exe el código AutoIt que tuviese en caso de estar programado con éste
- Uniextract, que permite desempaquetar ejecutables detectando el packer con el que están empaquetados, permitiendo así su posterior análisis
- FLOSS, que intenta desofuscar las cadenas de texto encontradas en los ejecutables
- Wireshark, con el que se puede extraer información de comunicaciones de red, por mencionar algunos ejemplos.

A esas aplicaciones con un uso más específico se les dedicará un anexo.

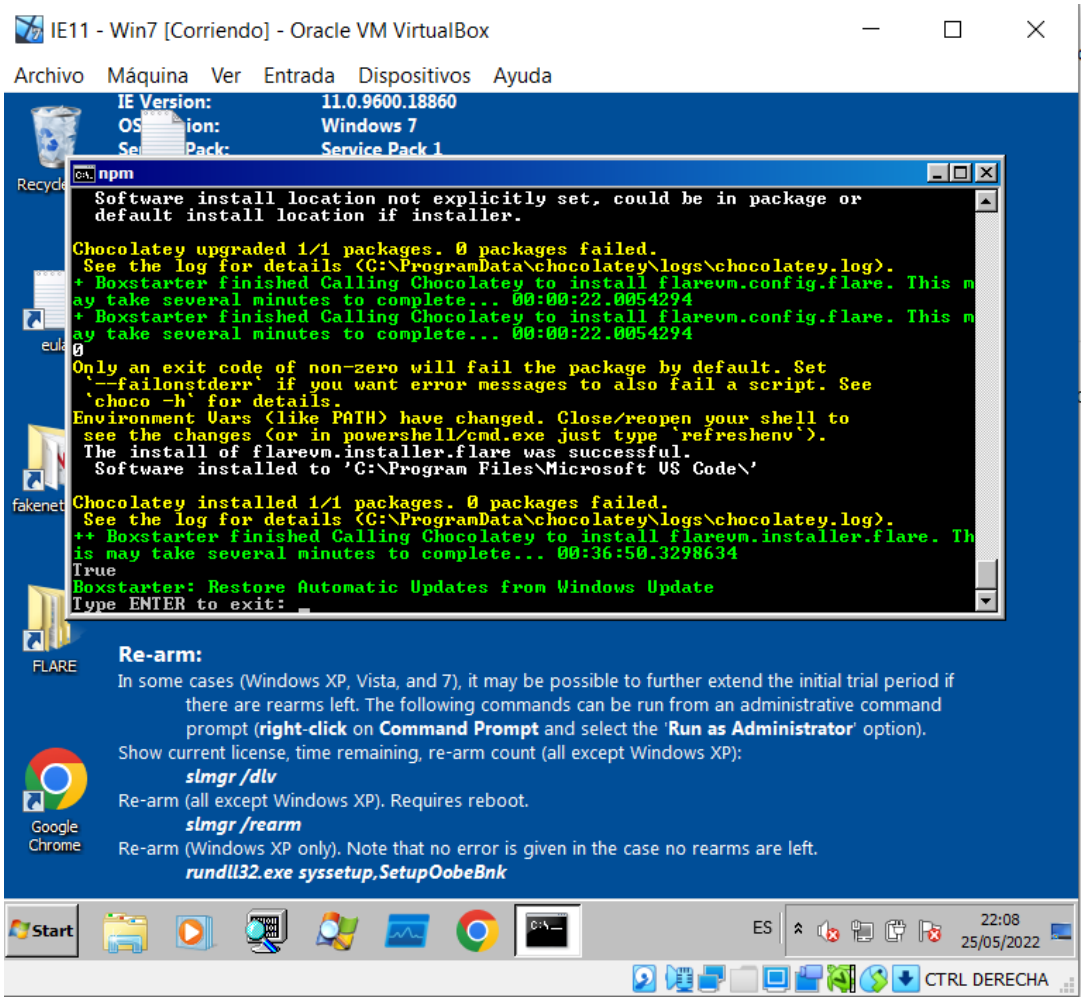


Figura 18: Finalizando la instalación

Al finalizar, pulsaremos "*Enter*" y finalizaremos la ejecución del script de instalación de FlareVM.

En el Github de FlareVM existe una lista completa de los programas que instala [32]

Para instalar un programa nuevo basta con abrir una consola de comandos y ejecutar la orden "*cinst <programa>*"

Ejemplo: *cinst capa*

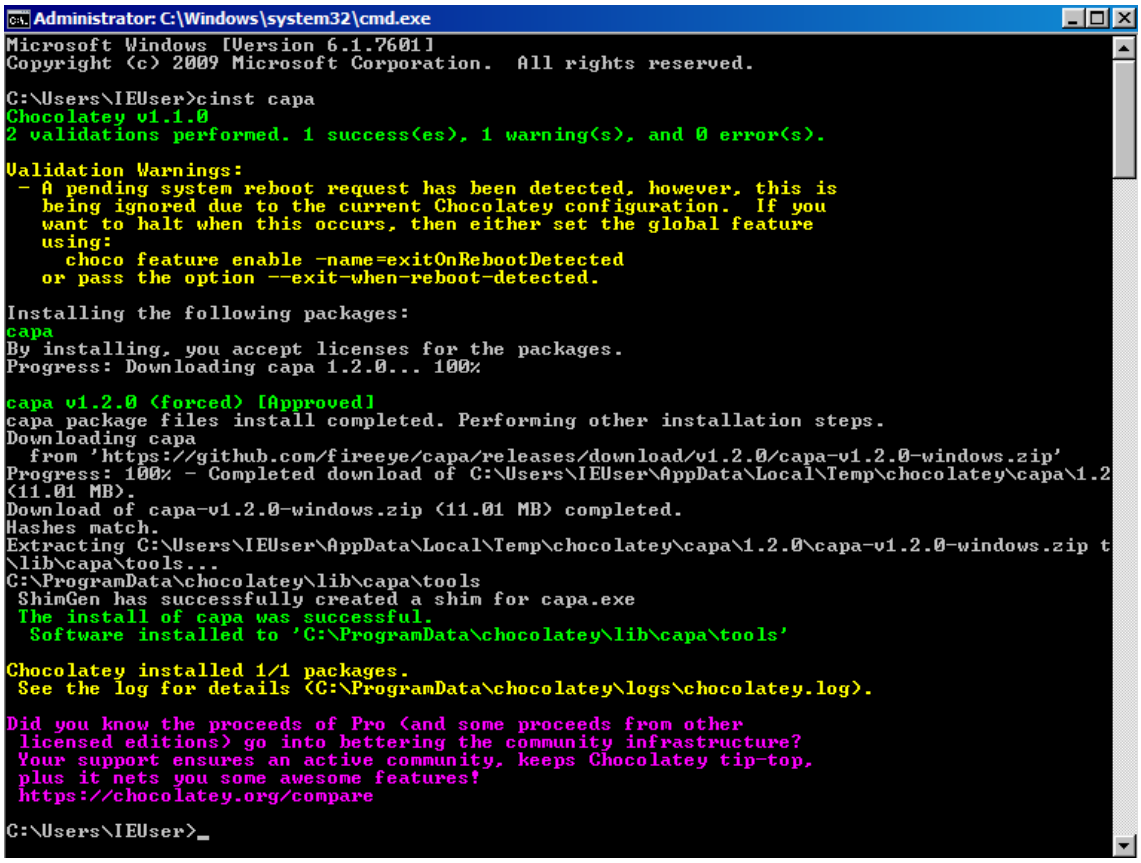


Figura 19: Instalación de la aplicación CAPA mediante el comando cinst

Se recomienda que después de la instalación de FlareVM se cambie la configuración de la tarjeta de red a "Adaptador solo-anfitrión" (Host Only), para aislar la VM de la red de área local a la que pertenece el sistema host, y así prevenir que alguna muestra se escape y dañe los equipos conectados a la red principal.

En este punto es obligatorio hacer una snapshot del entorno que acabamos de instalar, aunque no hallamos borrado las huellas digitales del entorno, ya que en caso de algún problema podríamos volver a este punto y solo tendríamos que ocultar las huellas de la máquina en caso de que queramos hacerlo, ya que hay malware que no hace uso de técnicas anti-virtualización, y con la configuración actual podríamos analizarlo.

Además de FlareVM, podemos hacer uso de "Indetectables Toolkit" [33], un pack de herramientas, que si bien hay muchas de uso general que se repiten, complementa

las herramientas de uso en casos específicos. Podemos usarlo alternativamente a FlareVM si creemos que se adapta mejor a nuestras necesidades.

9. Ocultación de la VM (Stealth VM)

En este capítulo se completará la configuración del entorno virtual, suprimiendo en la medida de lo posible su huella digital para que sea lo más parecido posible a un entorno físico real, con el fin de eludir comprobaciones de entornos que pueden hacer algunos malware.

9.1. Huella digital

Es inevitable dejar algún tipo de rastro usando un sistema operativo, ya que son intrínsecas al comportamiento de éste (configuraciones, ficheros del sistema…). Todo este ecosistema genera una huella digital similar entre los sistemas operativos que los programadores de malware aprovechan para discernir si su malware se está ejecutando en una máquina virtual o en una máquina física.

Nos centraremos en las huellas digitales que genera Virtualbox con Windows, ya que esta es la configuración que trataremos en este trabajo. Los rastros que Virtualbox genera sobre sistemas Windows constan de:

- Comprobación de existencia de hipervisor y fabricante del mismo
- Comprobación de espacio en disco
- Comprobaciones de cadenas en entradas de registro relacionadas con el sistema de virtualización, en nuestro caso “Virtualbox” y “VBOX”
- Comprobación de dirección MAC
- Comprobación de cadenas referentes a Virtualbox en los dispositivos del sistema
- Comprobación de cadenas referentes a Virtualbox en el nombre de la BIOS
- Entre otros

En la figura 20 se muestra cómo aparecen algunos de estos rastros en el editor de registro de Windows. Las cadenas referentes a la BIOS, dado que es una BIOS virtual, el “fabricante” es el propio Virtualbox y le hace referencia con la cadena “VBOX”.

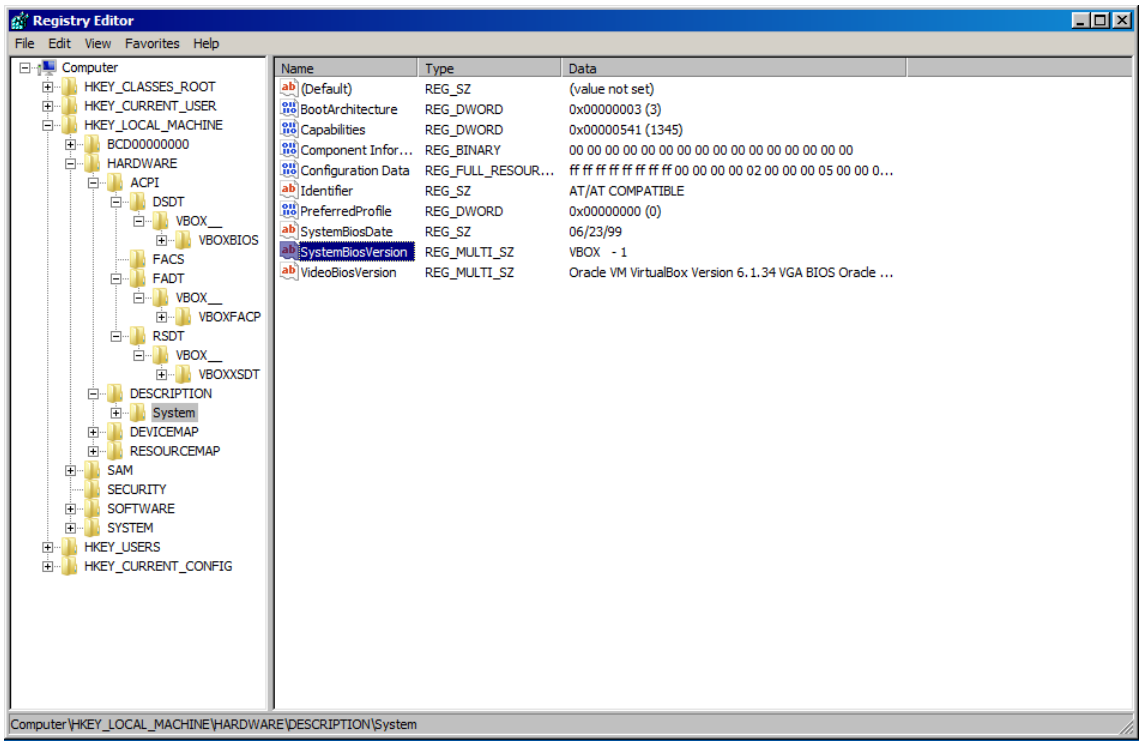


Figura 20: Registros con la cadena “VBOX” propia de Virtualbox

Con la utilidad Pafish [34] podemos detectar que es lo que delata a nuestro entorno virtual, así que la descargamos y la ejecutamos. Se nos mostrarán en pantalla las comprobaciones.

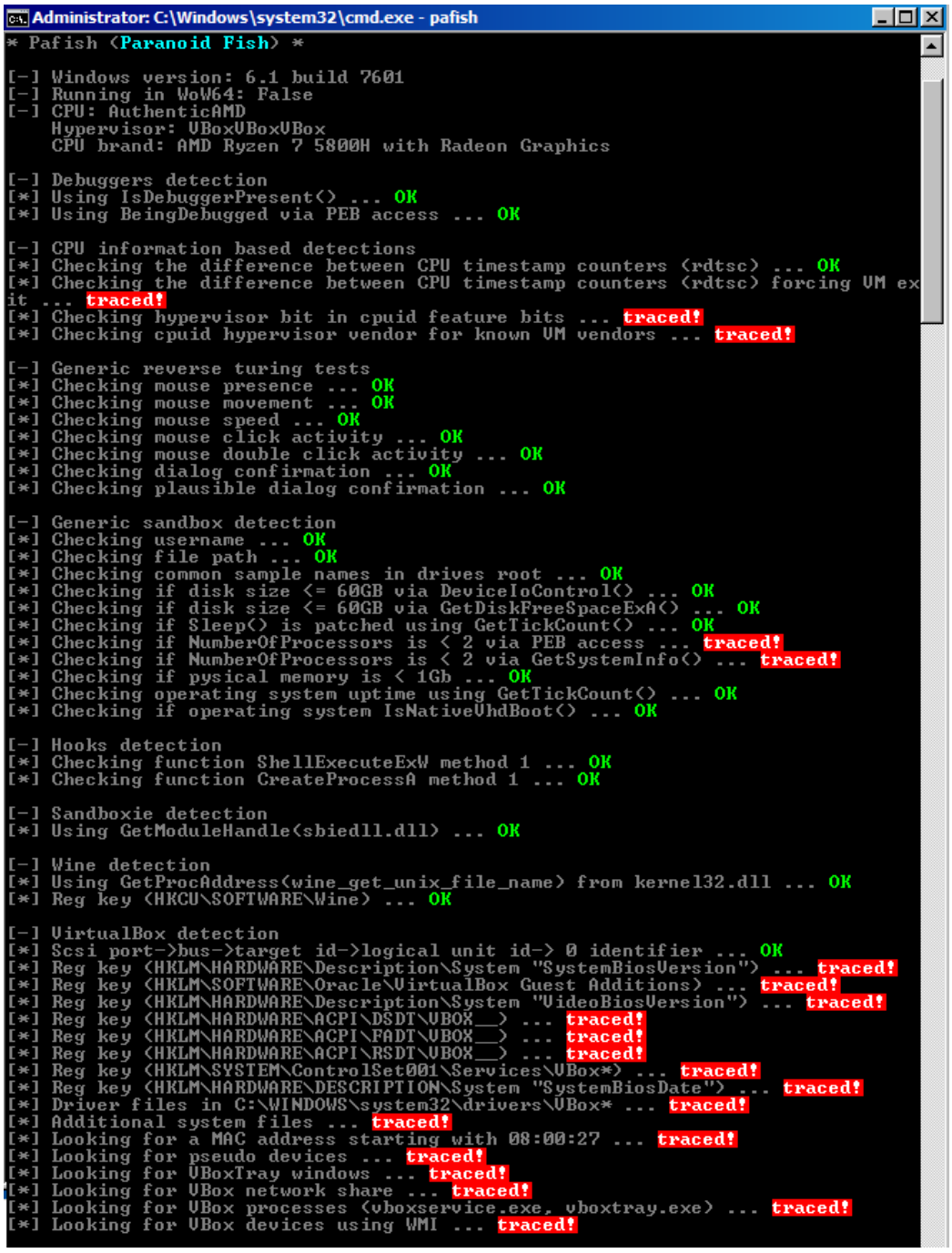


Figura 21: Primera comprobación de Pafish

Como se puede observar, hay varios tests que ya superamos, como la comprobación de espacio en disco <=60GB, dado que lo ampliamos en el punto 8.3 y la memoria física < 1GB, dado que hemos configurado nuestra máquina con 2GB.

No superamos el test de procesadores porque hemos configurado nuestra máquina con 1, si tuviéramos la posibilidad de ampliarle la configuración de

los núcleos este test se habría superado. Depende de si el equipo host nos permite esta configuración.

Para eliminar los rastros que delatan nuestro Hipervisor, haremos uso de la utilidad “vBoxSysInfoMod-v6” [35], la cual se tiene que ejecutar desde el sistema host en el que tengamos las máquinas virtuales, ya que modifica parámetros intrínsecos a las mismas y no se puede hacer de forma interna.

Esta utilidad nos permitirá reescribir el nombre del fabricante “Virtualbox” por otro que queramos nosotros en los diferentes rastros que haya dejado la máquina virtual en el sistema (registro, hardware…). Para limitar la detección de la máquina virtual por parte del malware que tenga características de detección de entornos virtuales.

Para ello, tendremos que desactivar temporalmente el AV de la máquina host y ejecutar “*Unpack and launch vBoxSysInfoMod.bat*”, que nos creará una carpeta con el archivo ejecutable e iniciará el programa, en caso de no iniciarse, puede ejecutarse manualmente con el archivo “*vBoxSysInfoMod v6.exe*”, accediendo a la carpeta recientemente creada.

Ejecutaremos el programa y con el comando “*ModifyVM*” comenzaremos la modificación de las huellas de nuestras máquinas virtuales.

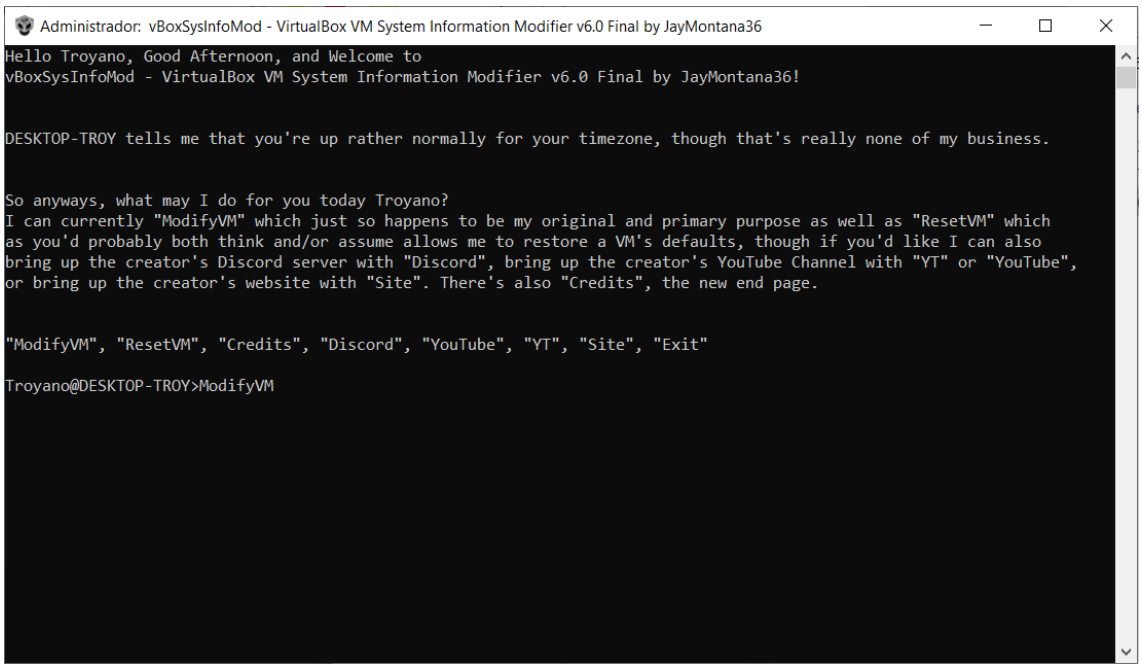


Figura 22: ModifyVM

Nos solicitará el nombre de la máquina que queramos modificar, en nuestro caso IE11 - Win7”, el nombre que identifique al nuevo fabricante, el modelo del sistema y la nueva fecha de la BIOS. Una vez hemos aportado estos datos, pulsaremos “Enter” para continuar con el proceso de sobreescritura. La máquina virtual a modificar deberá estar apagada.

Es importante que no se usen sólo números en el fabricante y modelo, ya que Virtualbox al arrancar lo interpretará como un numero entero en vez de como una cadena y abortará la ejecución debido a un error.

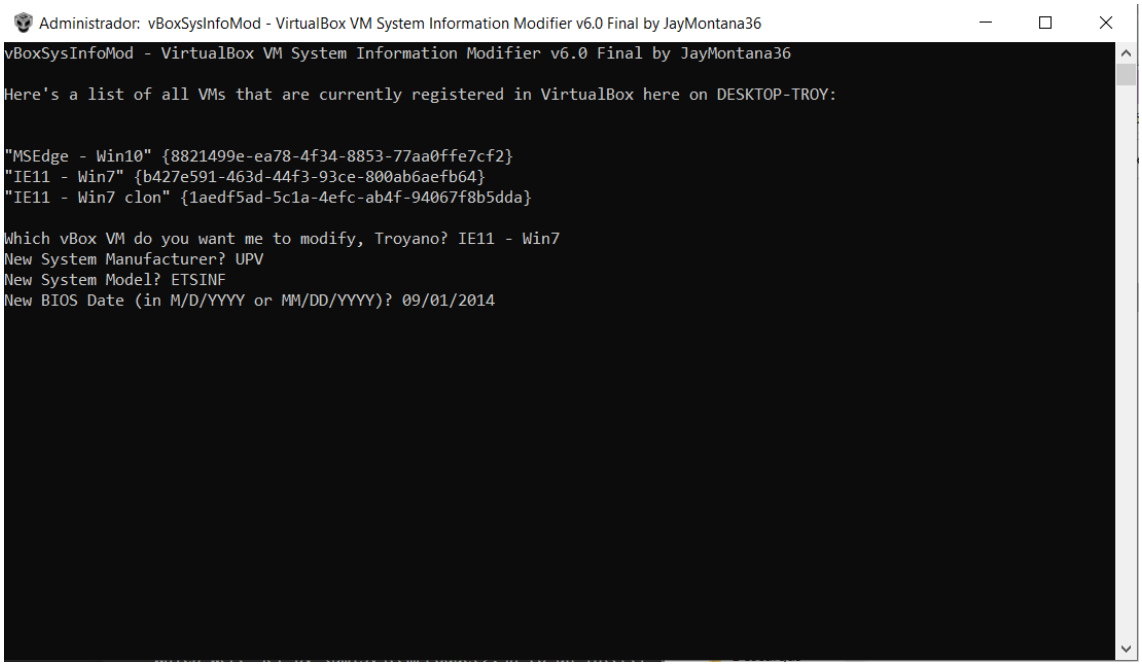


Figura 23: Modificación de datos

Se nos mostrará un pequeño resumen de los cambios a realizar, pulsaremos “Enter” de nuevo para confirmar y continuar. Hecho esto, aplicará los cambios en la máquina virtual.

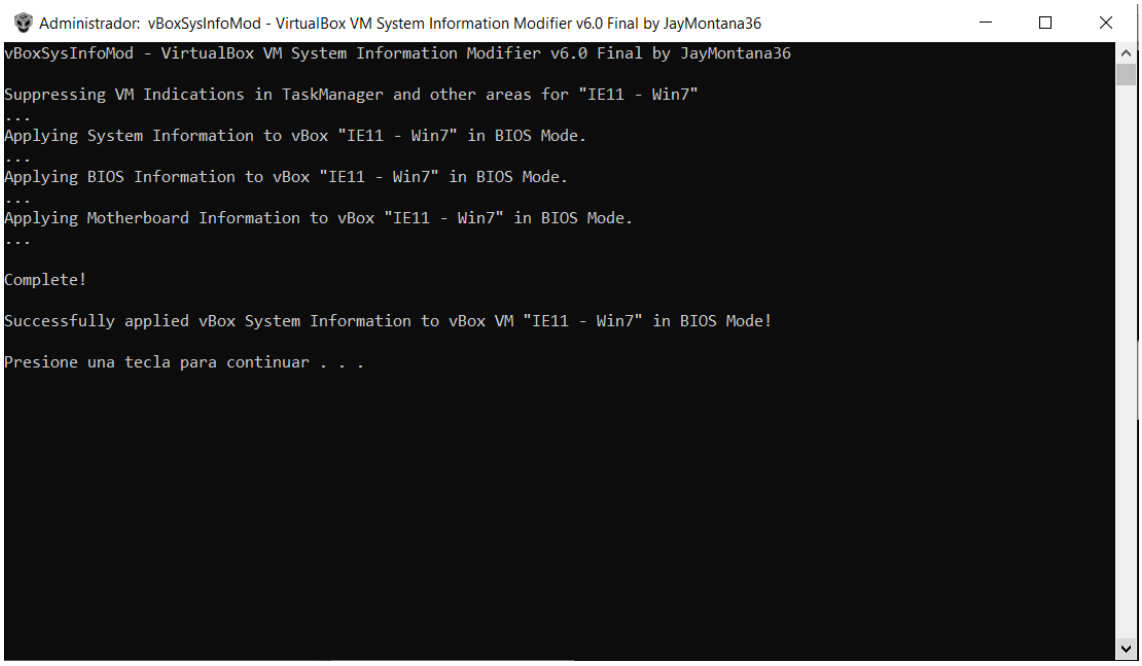


Figura 24: Aplicación de cambios

Realizados estos cambios, volveremos a ejecutar Pafish, para ver cuánto ha mejorado nuestra evasión a la detección.

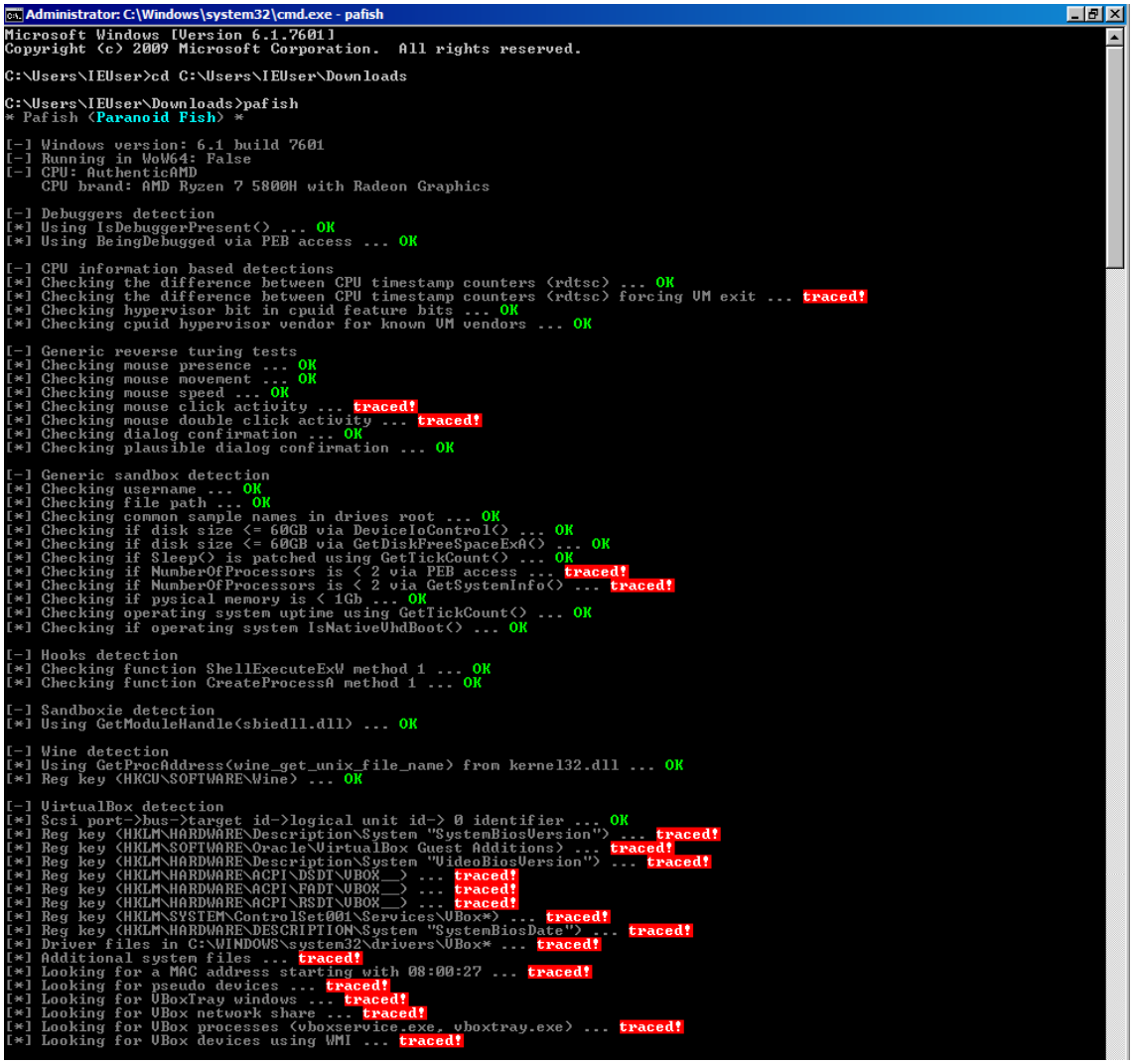


Figura 25: Segunda comprobación de Pafish

Los tests de turing inversos los superaremos si estamos moviendo el ratón en ese momento, ya que es lo que utiliza para detectar si el ordenador está siendo utilizado por una persona o en cambio es un sistema automático.

Como se puede observar, aunque si superamos algunas pruebas genéricas, no superamos las pruebas específicas para Virtualbox.

Utilizaremos la utilidad VBoxCloak [36] que se encarga de modificar entradas de registro y diversos artefactos adicionales para esconder los rastros de Virtualbox en el sistema. Para ello, nos descargamos el código de Github y ejecutamos vía Powershell el fichero “VBoxCloak.ps1”

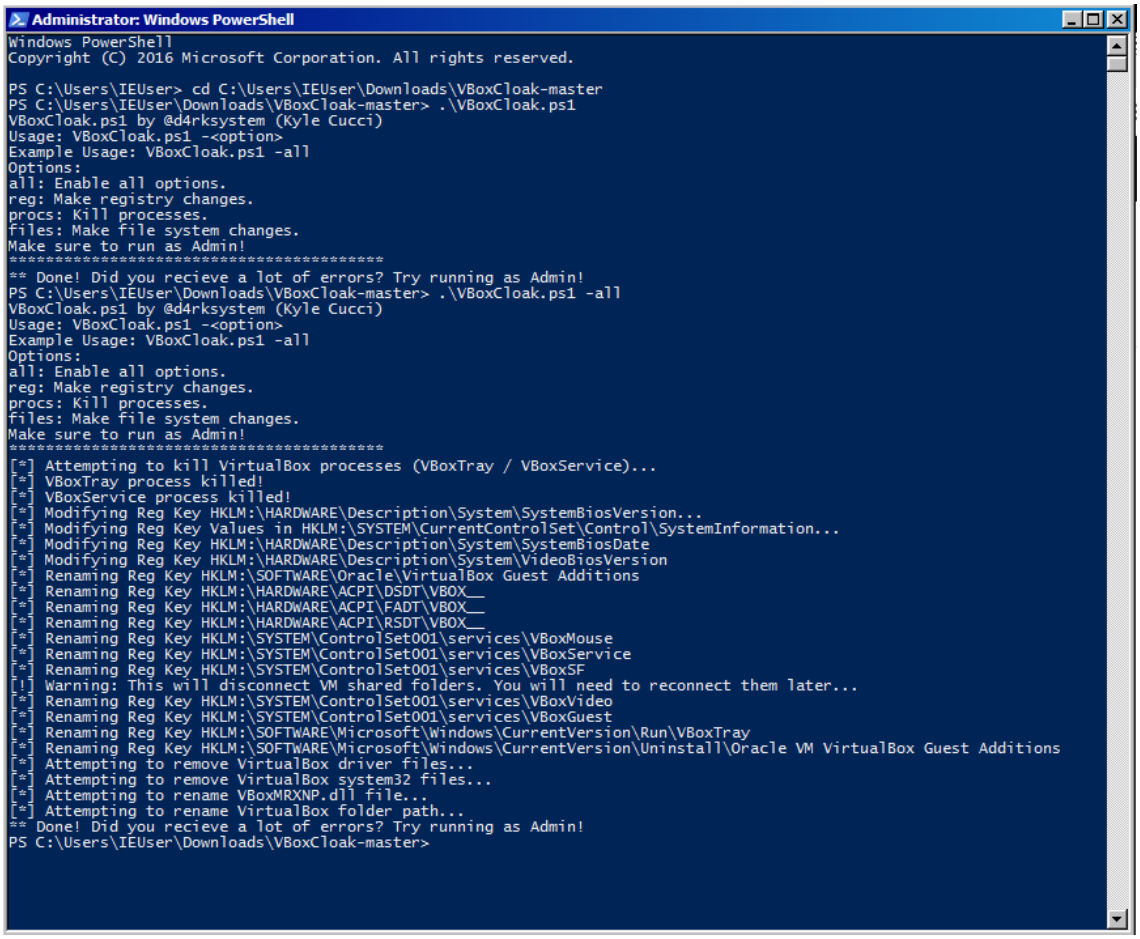


Figura 26: Ejecución de VBoxCloak

Hecho esto, volvemos a ejecutar Pafish para ver si hemos mejorado las propiedades anti detección de la máquina virtual.

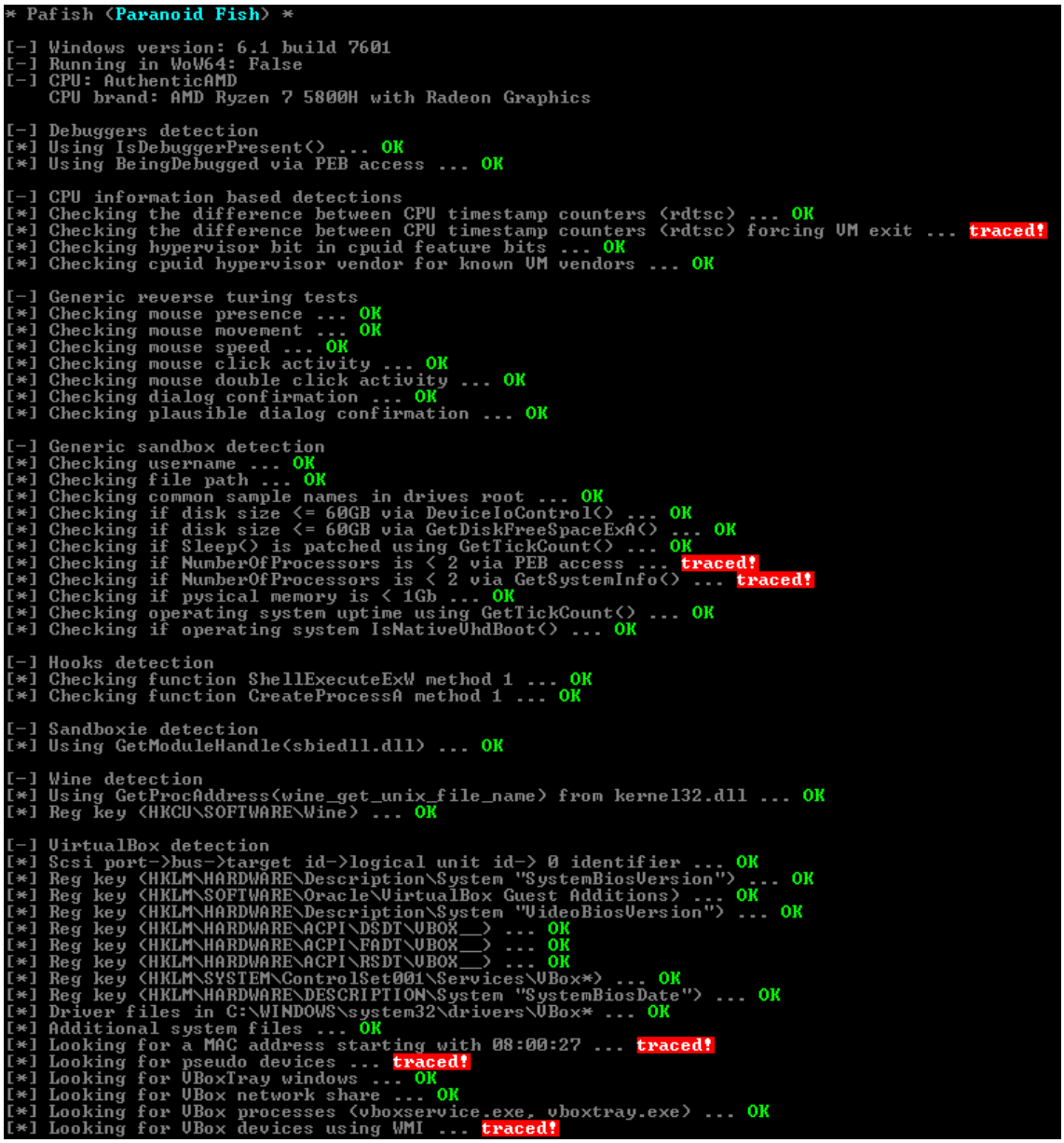


Figura 27: Tercera ejecución de Pafish

Aunque aún quedan rastros de un posible sistema virtualizado, esta configuración servirá para superar los test que haga la gran mayoría del malware actualmente. Uno de los peores casos que nos podríamos encontrar sería tener un MW protegido contra virtualización con Themida, ya que éste detecta los entornos VM por medio de cadenas de texto concretas en claves de registro.

Con nuestra configuración actual, a no ser que existieran algunas comprobaciones extras, probablemente superaríamos esos tests.

Además el propio Windows deja rastros con configuraciones de fábrica que pueden dar lugar a que se detecten como máquinas virtuales como WORKGROUP en el nombre del grupo de trabajo o DESKTOP como parte

del nombre del equipo. Para evitar esto, basta con modificar dichos nombres.

Software comercial también hace uso de este tipo de detección para intentar bloquear su ejecución en VM.

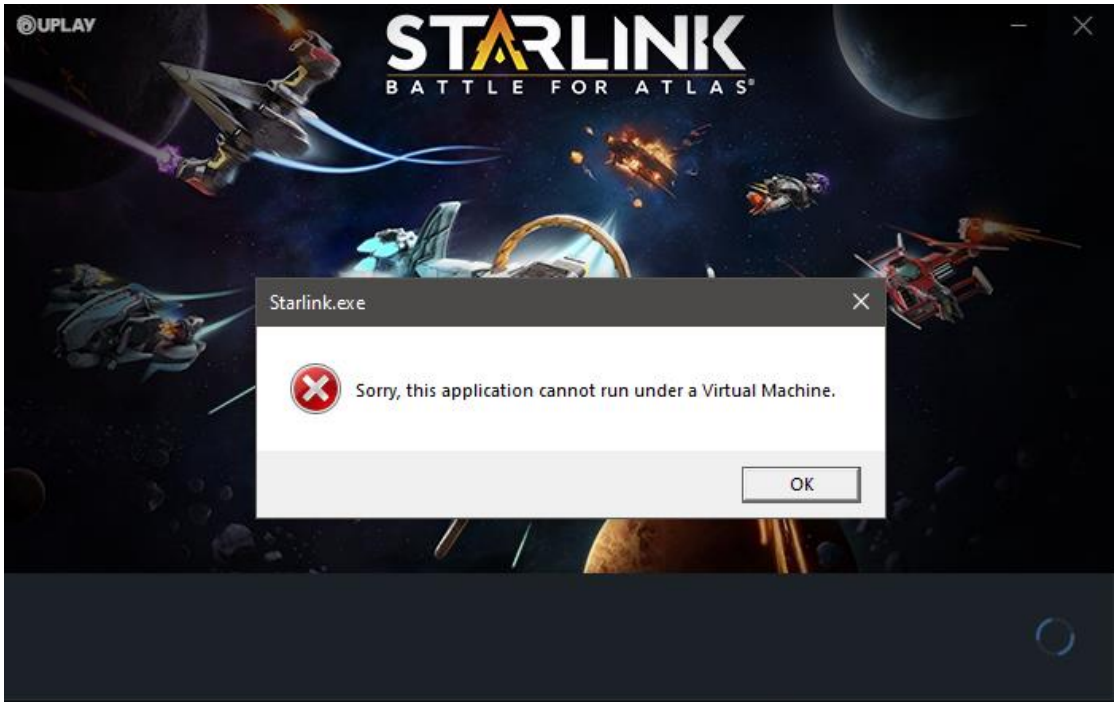


Figura 28: Programa comercial con detección de entorno virtualizado

En este punto, es recomendable crear otra snapshot del entorno, dado que acabamos de borrar los rastros que nos podrían delatar como VM y podremos volver a este punto en caso de que el sistema se infecte.

10. Proceso de Análisis – Análisis preliminar

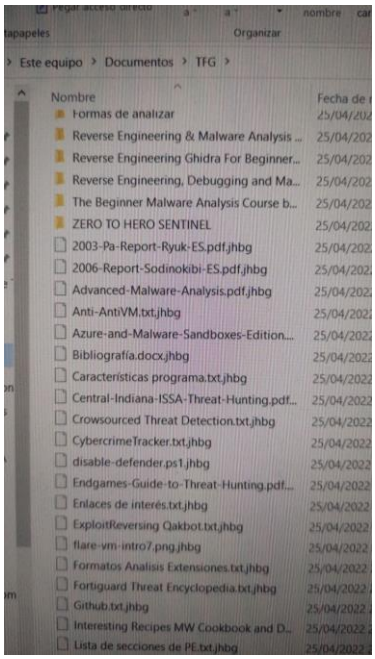
10.1. Escoger una muestra (Sitios de descarga)

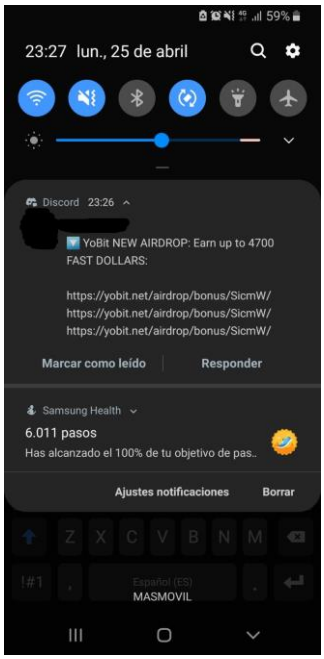
Hay varios sitios donde poder escoger una muestra, como Virustotal (Con licencia Enterprise) o la sección de muestras de VxUnderground [37] o incluso en Twitter [38]. Aunque podemos encontrar cualquier espécimen que nos sirva como muestra con una rápida búsqueda que contenga la palabra “crack” o “keygen”.

En este último caso, lo mejor que nos podría pasar sería encontrar un programa que realmente hiciera lo que se supone que debería hacer, siendo su nivel de peligro relativamente bajo.

En uno de los peores casos podríamos encontrarnos ante un ransomware con una rápida propagación y robo de credenciales (Figuras 29, 30 y 31).

También podemos encontrar diversos sitios de internet donde se hallan listas de sitios web [39] que permiten descargar muestras.





Figuras 29 y 30: Archivos de la investigación del TFG cifrados por un ransomware y distribución de phishing por Discord

Facebook	El correo electrónico principal de Facebook ha cambiado a fkzgscnapexfslptwfei@hotmail.com Hol...	Bandeja de ent...	26/04/2022
Facebook	El correo electrónico principal de Facebook ha cambiado a fkzgscnapexfslptwfei@hotmail.com Hol...	Bandeja de ent...	26/04/2022
Facebook	Cambio de la contraseña de Facebook Hola, Ivan Carlo: Se ha usado el número de teléfono +6283...	Bandeja de ent...	26/04/2022
Facebook	Se ha añadido el número +62 838-4754-7281 a tu cuenta de Facebook Hola, Ivan Carlo: El número...	Bandeja de ent...	26/04/2022

Figura 31: Robo de cuenta de Facebook por malware

10.2. Análisis preliminar - Tipos de ejecutables / Extensiones

Según el tipo del ejecutable tendremos que usar unas herramientas u otras para hacer un análisis correcto, en caso de encontrarse por primera vez con un formato de archivo que no conozcamos, procederemos con un análisis estático general que nos permita extraer algo de información. Para, posteriormente, ver si podemos o no, ejecutar un análisis dinámico.

Antes que nada necesitaremos saber de qué tipo es el binario, para poder usar las herramientas correctas, ya que si no lo hacemos así, es probable que no funcionen apropiadamente.

Para ello, haremos uso de Detect It Easy (DIE) [40], un programa que nos puede aportar bastante información inicial del fichero. Para detectar el tipo del fichero utiliza información puede encontrar en las cabeceras (File Type).

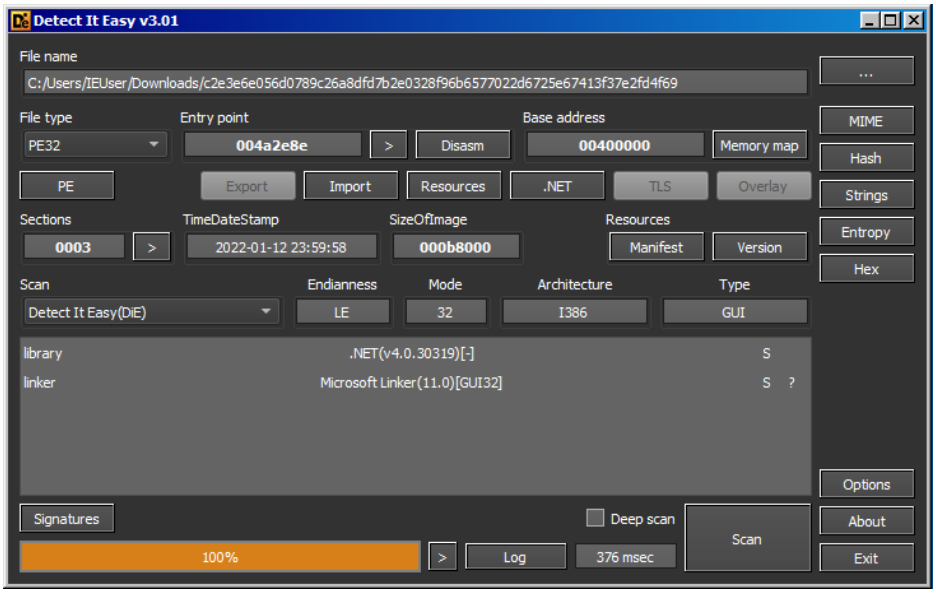


Figura 32: Información general de un ejecutable en .Net

Además, muestra información extra como el Compilador, Linker y Packer utilizados en el programa, las cadenas de texto que puede encontrar y la entropía dividida por secciones. Si tenemos un binario empaquetado con un packer, tendremos secciones con alta entropía.

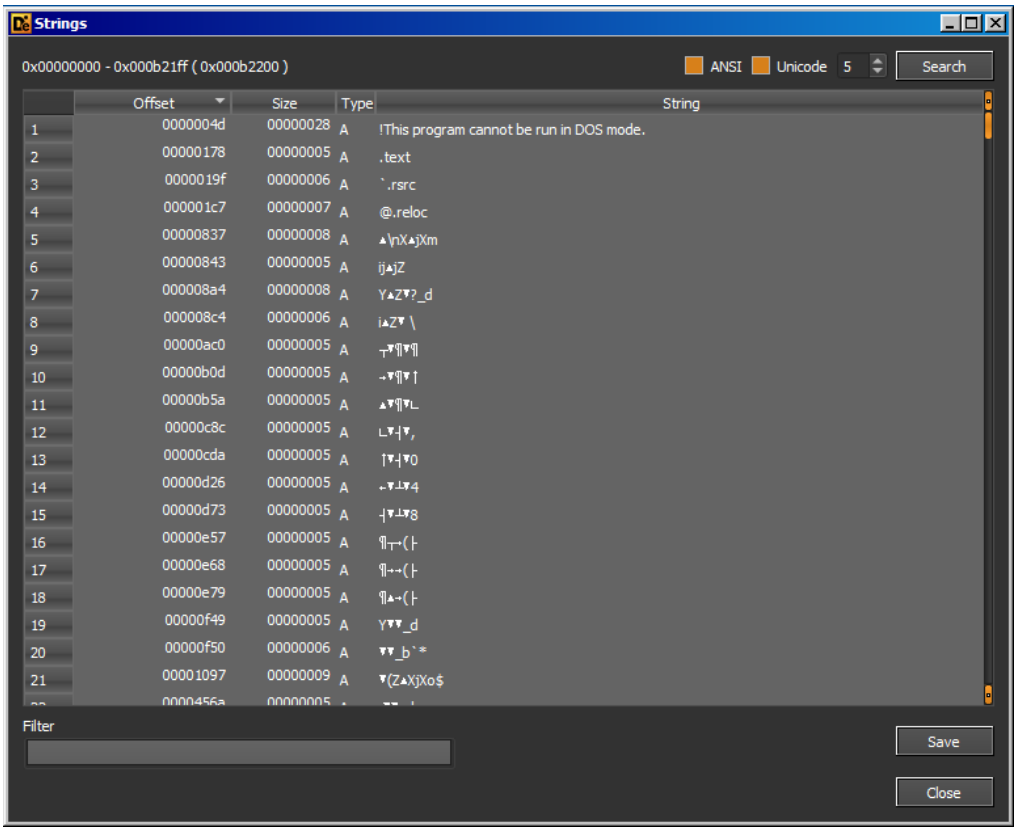


Figura 33: Strings del ejecutable

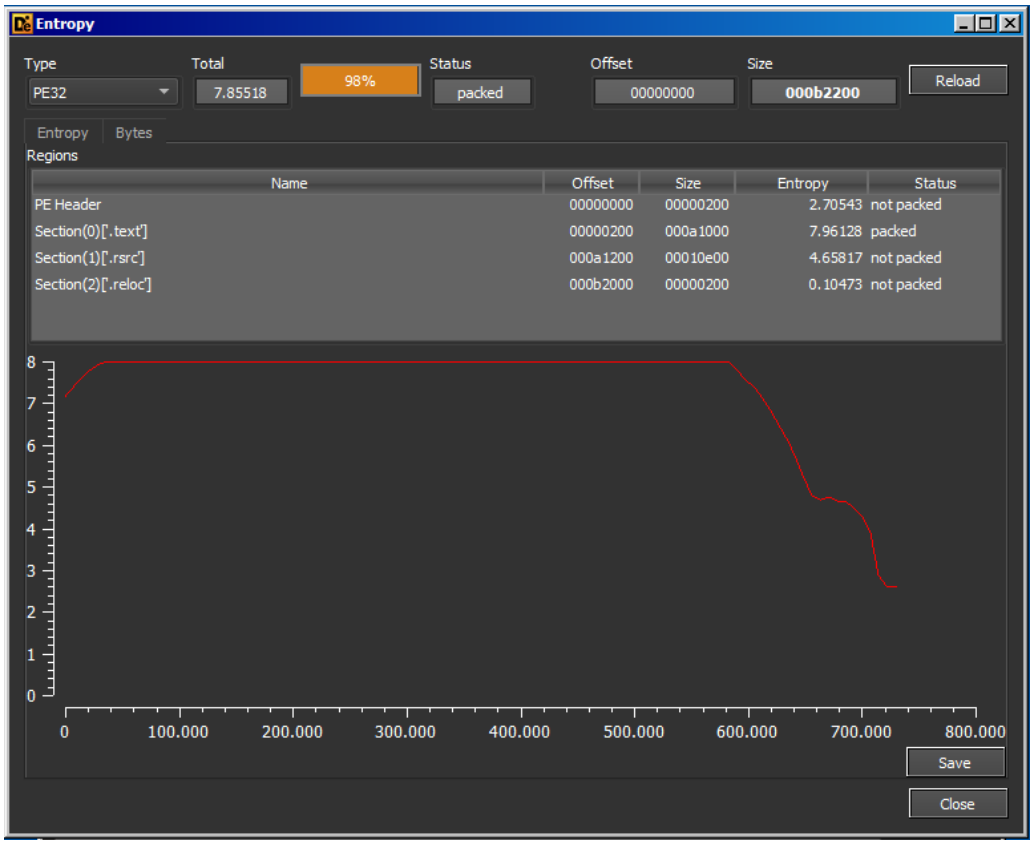


Figura 34: Secciones y entropía

Pueden haber secciones con nombres aleatorios o inexistentes, hay que diferenciar si esas secciones las crea el compilador/packer que se usó para generar el programa o en cambio son generadas por infección de un malware, en este punto se apela al sentido común para diferenciar entre unas y otras. Lo mismo sucede con las cadenas de texto encontradas en el programa, ya que es posible que el creador del malware haya dejado algún rastro, como una “firma” que lo delate.

Detectamos que, en el apartado de entropía, la sección “text” se marca como packed, con una alta entropía ocupando la mayoría del programa. Pero en la Figura 32 en cambio, no hemos detectado ningún packer. Esto podría implicar que el código en esa sección está ofuscado y comprimido (empaquetado) para dificultar su lectura, probablemente un empaquetado a medida, y que DIE no ha podido encontrar una firma de un packer comercial o conocido que pueda ajustarse. Este hecho puede ser un indicio de malware.

Para ficheros hechos con .NET, es preferible usar dnSpy en vez de DIE, ya que podremos ver el código fuente del ejecutable. También podemos usar dnSpy en cualquier ejecutable para detectar si hay alguna sección con permisos de Lectura, Escritura y Ejecución (RWX).

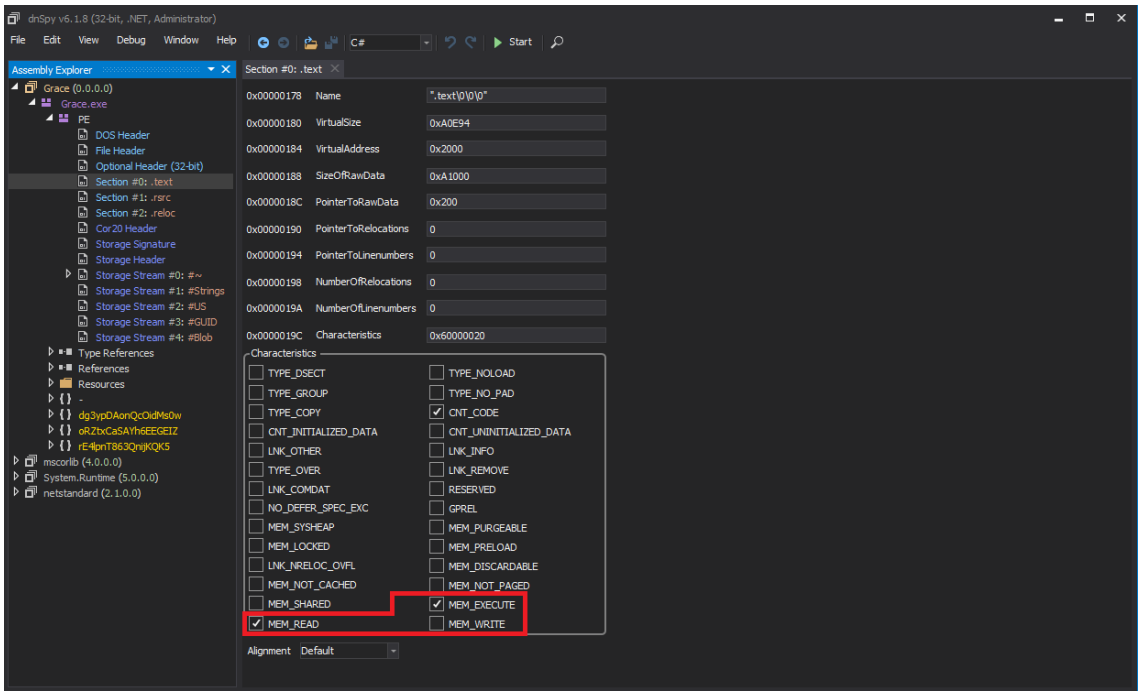


Figura 35: Características de las secciones en dnSpy

Un programa que tenga estos permisos activados, es muy probable que sea un malware, o esté infectado por uno. Ya que al poder ejecutar código y escribir lo que quiera, el propio código puede modificarse a sí mismo.

No es algo que esté prohibido, pero si es algo que no está bien visto si se trata de un programa “legal”, salvo en el caso de que se trate de un “packer”.

Que un programa no tenga secciones a RWX no quiere decir que no sea un malware. Pero en caso de tenerlas, con bastante seguridad podemos afirmar que se trata realmente de uno.

Si el ejecutable está programado en .Net, como es nuestro caso, además de las secciones y sus características, dnSpy mostrará su código que podremos examinar con mayor profundidad y así estaríamos haciendo un análisis estático avanzado del binario.

También podremos comprobar si hay información de la muestra en VirusTotal, ya que podremos añadir información valiosa a nuestro análisis que quizá se nos podría haber pasado o no haber encontrado. En el anexo “Información extra” se encuentra información más completa de esta muestra a día 22/06/2022.

11. Proceso de Análisis – Análisis estático

En este capítulo abordaremos el análisis estático (sin ejecución) sobre un fichero de estudio.

11.1. Estático básico

Como comentamos en el capítulo 3.1, un análisis Estático básico consiste en examinar el archivo ejecutable sin ver las instrucciones reales que lo componen.

Algunos analistas eliminan la extensión de los archivos por seguridad, en caso de no tener extensión, si no se nos ha dado ninguna indicación sobre que puede ser ese archivo, haremos uso de los “Magic Numbers” [41], unos códigos hexadecimales en la cabecera de los ficheros, al comienzo de ella, que se relacionan con un tipo de fichero concreto. Esto también nos puede servir en caso de que queramos investigar un fichero comprimido que no se pueda abrir.

Se pueden comprobar las cabeceras con cualquier editor Hexadecimal, como por ejemplo HxD. IDA Free también puede mostrar el contenido hexadecimal de un binario. Dependerá de las preferencias del analista o de su forma de trabajar el usar uno u otro.

Habiendo utilizado ya DIE, y detectado su tipo de fichero por medio de éste o de las cabeceras, podemos usar Resource Hacker para extraer algo más de información sobre el binario. Nombre del programa, versión, fabricante… o, si las detecta, algunas de sus ventanas de la interfaz gráfica. Esto nos facilita comprobar si hay alguna inconsistencia, por ejemplo, un programa podría mostrar el icono de cualquier antivirus pero no coincidir con su fabricante. Ese dato puede servir como pista de que el programa realmente no es lo que parece que es.

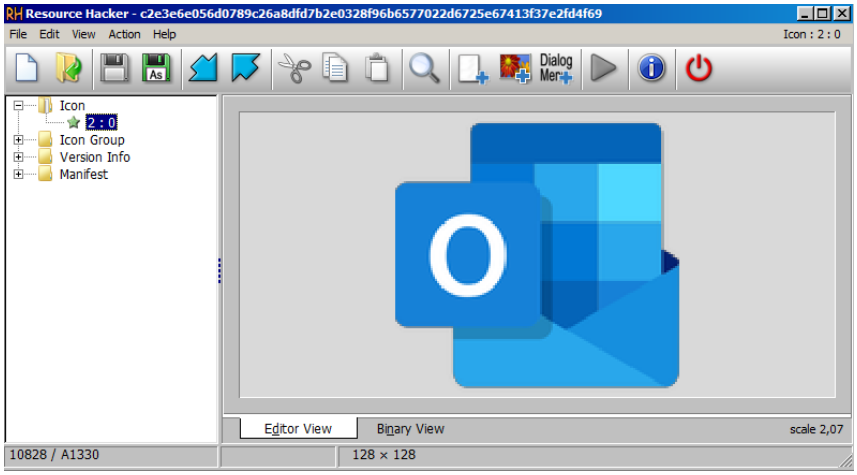


Figura 36: Icono usado por Grace.exe

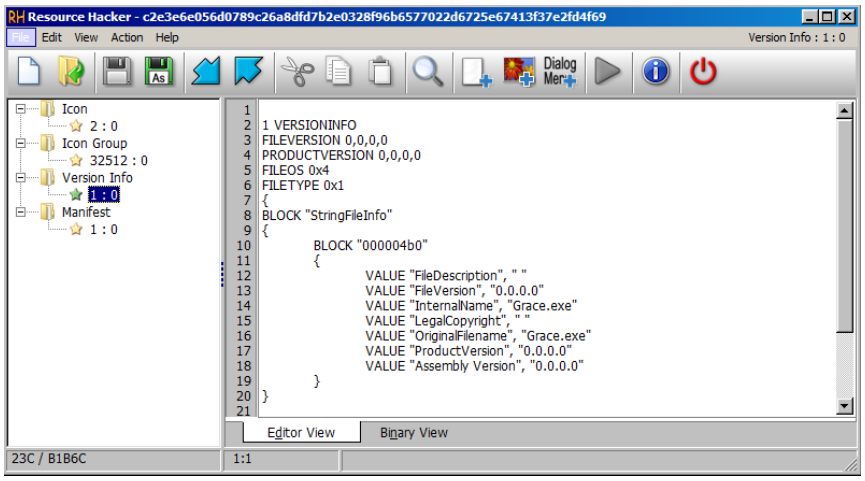


Figura 37: Datos de versión de Grace.exe

Como podemos observar, nuestra muestra posee un icono de Outlook, además de no encontrar ninguna referencia a Microsoft en su código. Estos datos son un posible indicio de malware.

Para contrastar información, compararemos los resultados extraídos con las demás herramientas con los resultados mostrados por CFF Explorer, lo más común es que sean los mismos, y CFF Explorer nos de algo más de información, ya que posee la información más estructurada.

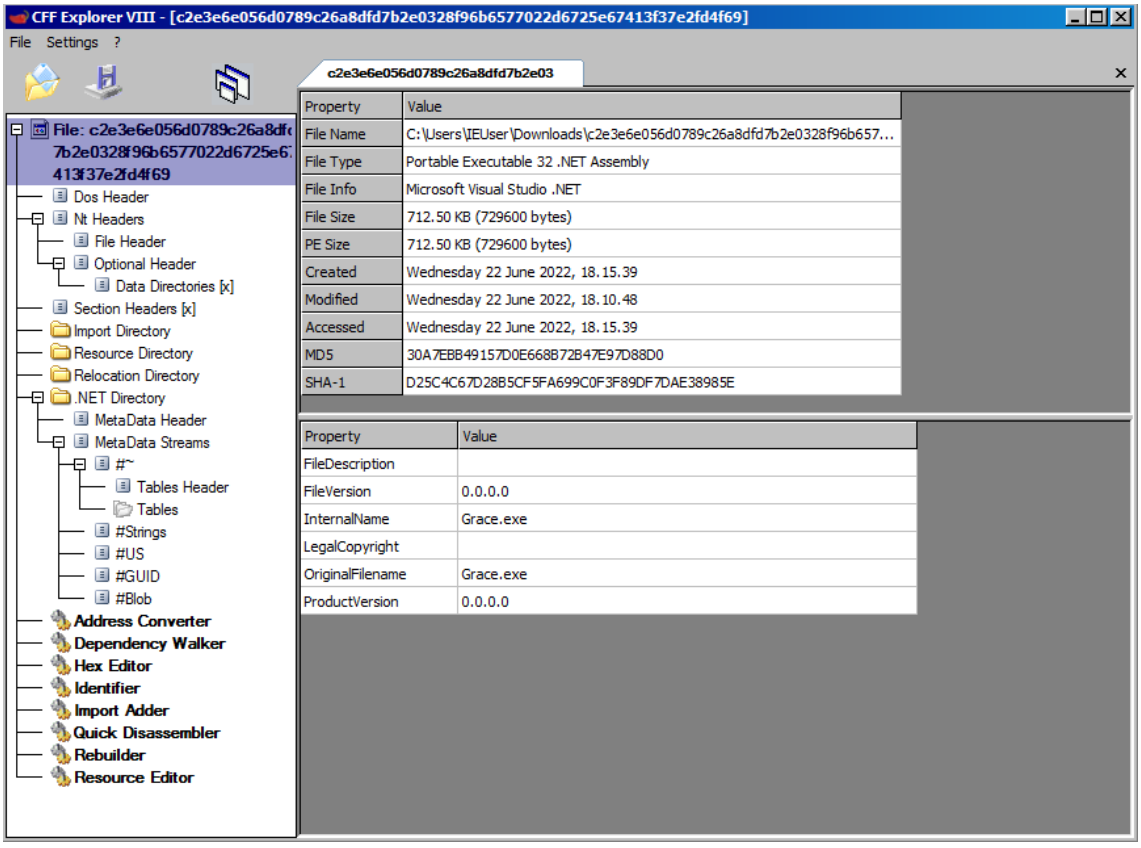


Figura 38: Grace.exe en CFF Explorer

Para acabar con este punto del análisis ejecutaremos CAPA, que nos permitirá conocer que comportamientos del programa coinciden con comportamientos vistos en ejecuciones de malware por MITRE. En este caso el uso de CAPA es infructuoso dado que no es capaz de analizar programas creados en .Net.

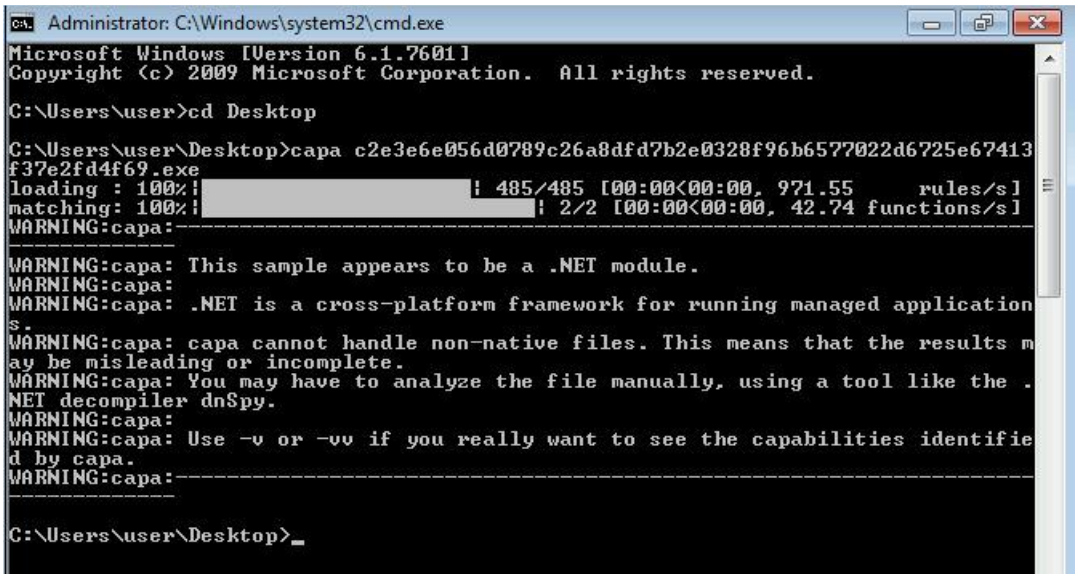


Figura 39: Ejecución de CAPA

11.2. Estático avanzado

El paso lógico, si realmente teníamos un .Net, ya que teníamos en ejecución dnSpy para ver sus permisos, hubiera sido pasar a hacer un análisis estático avanzado del código.

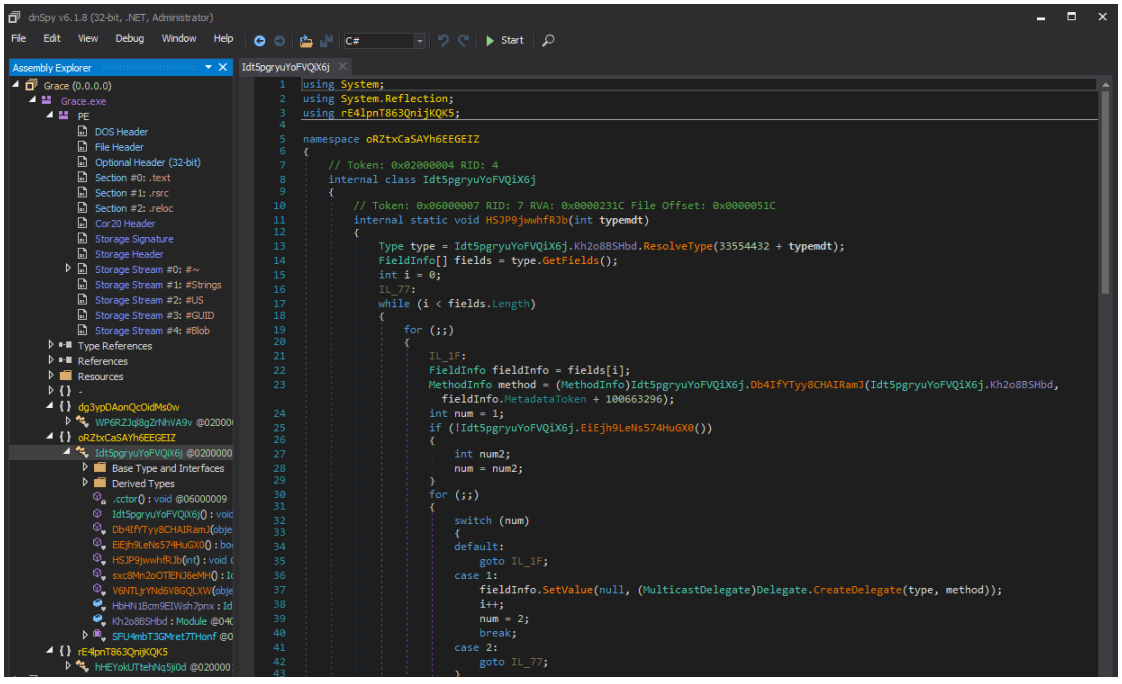


Figura 40: Código semi-ofuscado de Grace.exe

Como podemos observar, si bien el código es legible, algunas importaciones de librerías y uso de variables ofuscadas, lo cual nos reafirma la información recogida en el análisis preliminar (ofuscación) y que realmente se trataba de un .Net, ya que de otra manera no hubiéramos podido acceder a su código con dnSpy.

En otros casos, como por ejemplo Delphi, se usa “Interactive Delphi Reconstructor” (IDR). Según que ejecutable tengamos entre manos variará la forma que tengamos de analizarlo. A veces podremos acceder a su código, otras no y otras no podremos hacer un análisis estático y habrá que intentar recoger información mediante el análisis dinámico, o viceversa, en caso de que tengamos un archivo de datos, no podremos hacer un análisis dinámico, dado que no son archivos ejecutables y tendremos que conformarnos con la información extraída en el análisis estático.

En este momento del análisis se suele usar IDA para examinar las órdenes ensamblador del programa y sus importaciones de funciones externas a librerías aunque no es posible hacerlo con un .Net. Este paso nos puede dar información bastante relevante ya que si abrimos el ejecutable y navegamos hacia la pestaña de “Imports” podríamos encontrar funciones como:

- IsDebuggerPresent(): Detecta si el programa se está ejecutando bajo un debugger, y así comprobar si lo están monitorizando en proceso de ejecución

- CreateFile(): Permite al programa crear un archivo.

- VirtualAllocEx(): Usada por malware para inyectar su propio código en memoria perteneciente a otro proceso.

Como siempre, el uso de estas funciones no implica que un programa sea malware, dependerá de la naturaleza del programa y del contexto en el que se encuentre, ya que si es un programa legítimo, es posible que use IsDebuggerPresent() para protegerse frente al pirateo o si se trata de un procesador de textos puede hacer un uso legítimo de CreateFile() para crear un archivo nuevo.

Dado que la lista de funciones que podrían utilizarse es bastante extensa, se puede recurrir al anexo “Funciones de Windows” para profundizar en éste área.

Si intentamos desofuscar las variables con FLOSS, obtendremos que en este caso no ha podido descifrar nada y nos muestra todas las cadenas de texto que componen el programa. Pero, debido a la ejecución de este programa, se muestra que hay referencias a “Open” “Find” “Virtual” “Alloc” “Write” “Process” “Memory”, las cuales por separado pueden resultar referencias inofensivas, pero en el mismo contexto pueden relacionarse con funciones como “VirtualAllocEx()” o “WriteFile()” lo que pueden indicar una posible inyección de código y re-escritura de archivos, comportamiento habitual en un Ransomware.

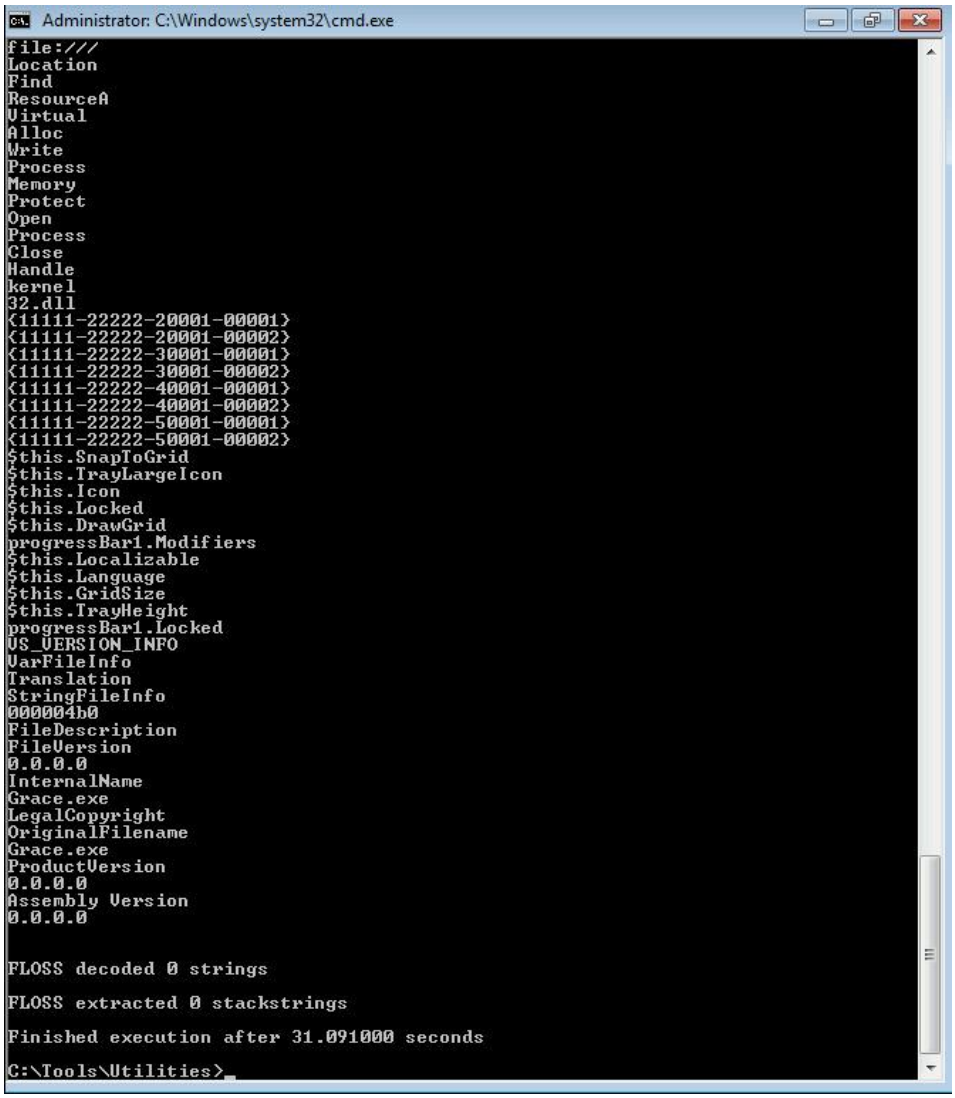


Figura 41: Ejecución de FLOSS para desofuscación de cadenas

12. Proceso de Análisis – Análisis dinámico

En este capítulo abordaremos el análisis dinámico sobre un fichero de estudio, posibilitando ver su ejecución en un sistema.

12.1. Dinámico básico

Usualmente, usaremos Total Uninstaller y Process Monitor como programas principales para hacer un dinámico básico, pero la forma de actuar varía según lo que estemos analizando.

En el caso de archivos .dll, usaremos rundll32.exe ejecutado vía cmd, con la orden "*rundll32.exe Nombre.dll, Almohadilla [Numero función a ejecutar]*" (sin []), es posible que tengamos que hacerlos tantas veces como funciones tenga la librería, dado que cada función se accede por un identificador numérico distinto, y previamente, monitorizando rundll32.exe con Process Monitor. Para ver qué cambios hace en el sistema a tiempo real.

Cuando se trata de archivos .ocx, ejecutaremos con regsvr32 via cmd, con la orden "*regsvr32.exe Nombre.ocx*", pre-monitorizando regsvr32.exe con Process Monitor, siendo el proceso muy parecido a los archivos .dll.

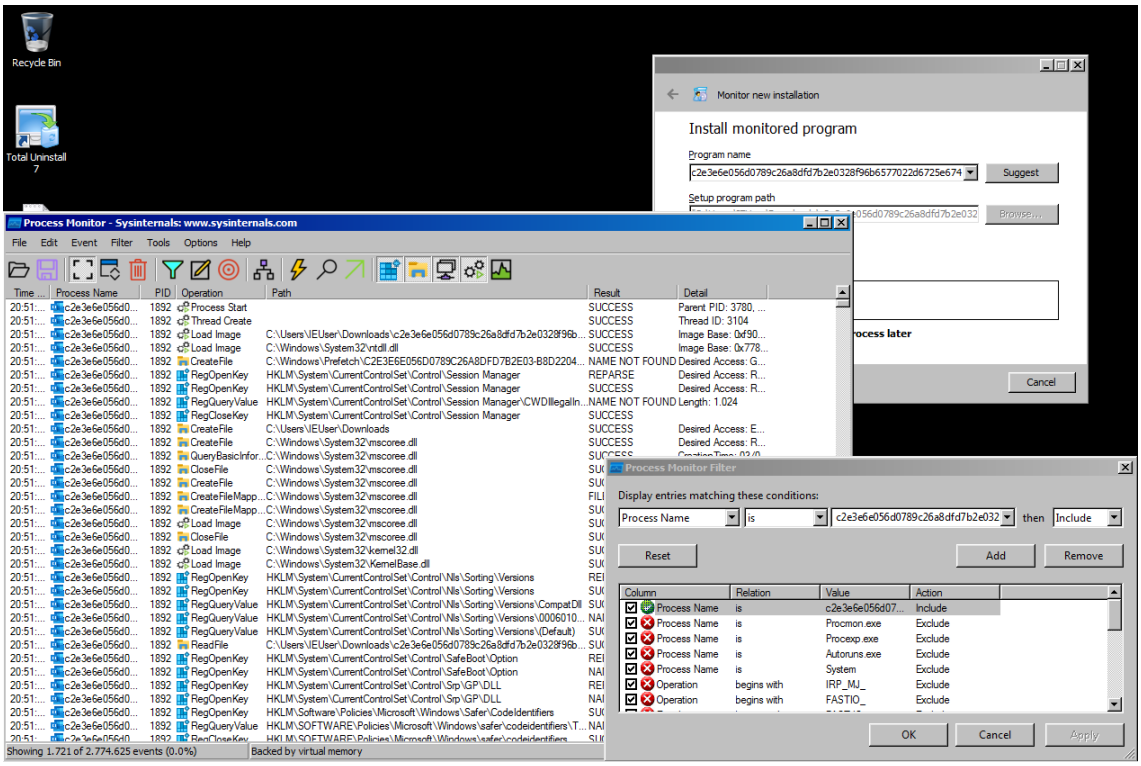


Figura 42: Grace.exe siendo monitorizado con Total Uninstaller y Process Monitor

En el proceso de monitorización de la ejecución, si un programa no muestra ventanas, sería posible que se tratase de un programa que trabaja vía cmd, así que habría que cambiar la forma de ejecución, o en cambio se trate de un programa que actúe en segundo plano con intenciones sospechosas. En el primer caso, cambiaríamos la forma de ejecución y monitorización ejecutando por cmd, si fuera el segundo caso, podría ser un indicio de malware.

12.2. Dinámico avanzado

En esta última instancia, intentaremos analizar el flujo de ejecución del programa y detectar funciones importadas de librerías externas que puedan usarse de forma ilegítima, este tipo de análisis requiere un alto nivel de conocimiento del lenguaje ensamblador y del flujo de ejecución de los programas. Las herramientas que se pueden usar en este proceso son los debuggers X64dbg/X32dbg o los debuggers integrados de IDA y Ghidra. Este debería ser el último paso de un análisis si aún no hemos verificado si la muestra que estamos analizando es un malware.

13. Resumen de características

A continuación se especifica que la muestra con MD5 30a7ebb49157d0e668b72b47e97d88d0:

- Se trata de un .Net
- Empaquetado desconocido o personalizado con alta entropía en la sección .text
- No tiene permisos Lectura(R), Escritura(W) y Ejecución(X) activados a la vez en ninguna de sus secciones
- Comprende ofuscación en variables e importaciones de librerías
- Usa icono de Microsoft Outlook
- No hay referencias a Microsoft ni a Outlook en sus cadenas de texto
- No muestra ventanas en ejecución
- Se encuentra una cadena de texto con nombre “Grace.exe”
- Posible comportamiento ransomware debido a referencias con “VirtualAllocEx()” y “WriteFile()”
- Se encuentra en VirusTotal con 57 detecciones a día 22/06/2022
- Con relaciones a un dominio sospechoso (Anexo “Información Extra”)
- Comentarios en VirusTotal referentes a Lokibot con enlaces a análisis externos

14. Conclusión del análisis

Dadas las incongruencias encontradas en el análisis de la muestra, como el uso de un icono para intentar hacerse pasar por otro programa, ausencia de información del fabricante “legítimo” del programa al que intenta parecerse, la ofuscación encontrada en sus variables e importaciones de librerías, alta entropía en la sección .text y su ejecución en segundo plano, se concluye que este programa es un malware.

La conclusión se ve respaldada por las detecciones de los motores antivirus en la web de VirusTotal, junto con otros datos como relaciones con dominios sospechosos y comentarios de la comunidad que lo catalogan como “Lokibot” [42]

15. Análisis parciales y comparativas

Podríamos comparar el análisis de nuestro ejecutable con otros binarios para poder ver diferencias entre ellos y profundizar en las herramientas que se pueden utilizar, aunque está fuera del alcance de este trabajo dado que no es necesario para la catalogación de un malware, se provee en el anexo “Información extra” diversos análisis parciales de muestras goodware y malware para ver las diferencias entre ellas y poder ganar competencias en este ámbito.

16. Anotaciones finales

Este entorno de análisis puede escalarse y generar un producto/servicio bastante interesante debido a su potencial de generación de inteligencia.

Los entornos de análisis manual, se pueden replicar y gestionar rápidamente de forma remota, lo que puede dar lugar a un PaaS de análisis de malware, ideal para analistas que necesiten un entorno o varios, ya que es fácilmente escalable o para estudiantes que se interesen sobre este área y necesiten un entorno controlado.

Por otro lado las herramientas automáticas, como “Capa” o sandbox completas como Cuckoo, permitirían ofrecer el servicio de generación de informes de inteligencia sobre malware. Al extraer muchos de los IOCs del malware a analizar, al guardar dicha información en bases de datos y con una plataforma como MISP que permite hacer una correlación entre los diferentes sucesos, podría detectarse de forma proactiva si se está ante una posible campaña de distribución de MW tanto general, como dirigida.

17. Mejoras a Futuro

A continuación se muestra una serie de mejoras que podrían implementarse en un futuro para enriquecer el entorno de análisis:

- Uso de Proxmox para gestionar múltiples MV de cara a crear servicios PaaS de entornos de análisis y generación de informes automáticos, dado que sería posible replicar este entorno de análisis y añadir Cuckoo Sandbox junto con CAPA, Malwoverview y Easyhunting.

- Instalación de Docker en las MV, y creación de un contenedor con las herramientas de FlareVM para facilitar los despliegues.

- Mejorar el borrado de huellas digitales de las MV modificando la dirección MAC, el número de procesadores asignados y parcheando la instrucción rdtsc.

- Instalación de MISP en un entorno separado por seguridad, pero, alimentado por los análisis realizados en el entorno FlareVM, creando así una base de datos con la que poder hacer seguimiento a posibles campañas de amenazas.

- Volcado de la información generada por los análisis automáticos en el entorno MISP, para nutrir constantemente a la base de datos de información de nuevas muestras.

- Uso de Kibana y ElasticSearch para visualizar los datos de MISP de forma amigable para el usuario.

18. Bibliografía

[1] https://www.newtral.es/rusia-ucrania-cronologia-de-una-ciberguerra-ciberataques/20220301/ (Último acceso el día 22/06/2022)

[2] https://www.pandasecurity.com/es/mediacenter/src/uploads/2017/11/Hospitals_Cyber-Pandemic-es.pdf (Último acceso el día 22/06/2022)

[3] https://www.interpol.int/es/Delitos/Ciberdelincuencia/Ciberamenazas-relacionadas-con-la-COVID-19 (Último acceso el día 22/06/2022)

[4] https://www.welivesecurity.com/la-es/2022/02/24/hermeticwiper-nuevo-malware-tipo-wiper-ataca-ucrania/ (Último acceso el día 22/06/2022)

[5] https://www.welivesecurity.com/la-es/2022/03/02/isaacwiper-hermeticwizard-nuevo-wiper-y-worm-utilizados-ciberataques-ucrania/ (Último acceso el día 22/06/2022)

[6] https://cybersecuritynews.es/descubren-un-nuevo-grupo-de-amenazas-apt-llamado-chamelgang/ (Último acceso el día 22/06/2022)

[7] https://www.welivesecurity.com/la-es/2022/03/22/sandworm-historia-ciberataques-atribuidos-este-grupo/ (Último acceso el día 22/06/2022)

[8] https://www.youtube.com/watch?v=nRY3QapA6Hg (Último acceso el día 22/06/2022)

[9] https://www.youtube.com/watch?v=2uB_PZlum2w (Último acceso el día 22/06/2022)

[10] https://malwareunicorn.org/workshops/re101.html (Último acceso el día 22/06/2022)

[11] https://malwareunicorn.org/workshops/re102.html (Último acceso el día 22/06/2022)

[12] http://ricardonarvaja.info/ - https://www.youtube.com/watch?v=Af5pvCl0CBE (Último acceso el día 22/06/2022)

[13] https://download.microsoft.com/download/9/C/5/9C5B2167-8017-4BAE-9FDE-D599BAC8184A/pecoff.docx (Versión 11 - 2017) (Último acceso el día 22/06/2022)

[14] https://www.virtualbox.org/ (Último acceso el día 22/06/2022)

[15] https://github.com/mandiant/flare-vm (Último acceso el día 22/06/2022)

[16] https://hex-rays.com/ida-free/ (Último acceso el día 22/06/2022)

[17] https://github.com/dnSpy/dnSpy (Último acceso el día 22/06/2022)

[18] https://www.martau.com/es/uninstaller-download.php (Último acceso el día 22/06/2022)

[19] http://www.angusj.com/resourcehacker/ (Último acceso el día 22/06/2022)

[20] https://github.com/horsicq/Detect-It-Easy (Último acceso el día 22/06/2022)

[21] https://docs.microsoft.com/en-us/sysinternals/downloads/procmon (Último acceso el día 22/06/2022)

[22] https://github.com/mandiant/capa (Último acceso el día 22/06/2022)

[23] https://github.com/mandiant/flare-floss (Último acceso el día 22/06/2022)

[24] https://developer.microsoft.com/en-us/windows/downloads/virtual-machines/ y https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/ (Último acceso el día 22/06/2022)

[25] https://tb.rg-adguard.net/public.php (Último acceso el día 22/06/2022)

[26] Documentación de VirtualBox sobre VBoxManage: https://www.virtualbox.org/manual/ch08.html#vboxmanage-modifymedium (Último acceso el día 22/06/2022)

[27] https://github.com/mandiant/flare-vm (Último acceso el día 22/06/2022)

[28] https://github.com/jbara2002/windows-defender-remover (Último acceso el día 22/06/2022)

[29] https://www.microsoft.com/en-us/download/details.aspx?id=30653 (Último acceso el día 22/06/2022)

[30] https://www.microsoft.com/en-us/download/details.aspx?id=54616 (Último acceso el día 22/06/2022)

[31] https://github.com/mandiant/flare-vm (Último acceso el día 22/06/2022)

[32] https://github.com/mandiant/flare-vm/blob/master/packages.csv (Último acceso el día 22/06/2022)

[33] https://github.com/indetectables-net/toolkit (Último acceso el día 22/06/2022)

[34] https://github.com/a0rtega/pafish (Último acceso el día 22/06/2022)

[35] https://sites.google.com/site/jaymontana36jasen/documentation/software/vBoxSysInfoMod-v6 (Último acceso el día 22/06/2022)

[36] https://github.com/d4rksystem/VBoxCloak (Último acceso el día 22/06/2022)

[37] https://samples.vx-underground.org/ (Último acceso el día 22/06/2022)

[38] https://twitter.com/malwrhunterteam (Último acceso el día 22/06/2022)

[39] https://zeltser.com/malware-sample-sources/ (Último acceso el día 22/06/2022)

[40] https://github.com/horsicq/Detect-It-Easy (Último acceso el día 22/06/2022)

[41] https://www.welivesecurity.com/la-es/2015/10/01/extension-de-un-archivo-cabeceras/ (Último acceso el día 22/06/2022)

[42] https://malwiki.org/index.php?title=LokiBot (Último acceso el día 22/06/2022)

Anexo A – Funciones de Windows

A continuación se provee una breve lista de funciones de Windows a tener en cuenta, aunque hay más funciones que las que se comentan aquí [1], las siguientes son las que se pueden encontrar de forma más usual usadas por malware.

Que sean usadas por el malware no implica que todos los programas que las usen lo sean. Por ejemplo IsDebuggerPresent, también es usada por programas legítimos para detectar si los están monitorizando, como forma de protegerse contra el pirateo de software.

También hay funciones referentes a creación de archivos, descarga de archivos desde la red, conexión a internet… En resumen, son funciones legítimas del sistema operativo, pero podrían usarse en el funcionamiento de un malware. Que un malware sea reconocido como tal dependerá del contexto y de qué forma use dichas funciones.

Funciones

AdjustTokenPrivileges
Se utiliza para activar o desactivar privilegios de acceso específicos. El malware que realiza la inyección de procesos suele llamar a esta función para obtener permisos adicionales.

CreateFile
Crea un nuevo archivo o abre un archivo existente.

CreateProcess
Crea y lanza un nuevo proceso. Si el malware crea un nuevo proceso, tendrá que analizar también el nuevo proceso.

CreateRemoteThread
Se utiliza para iniciar un hilo en un proceso remoto. El malware utiliza CreateRemoteThread para inyectar código en un proceso diferente.

CreateService
Crea un servicio que puede iniciarse en el momento del arranque. El malware suele utilizar CreateService para persistencia u ocultación.

gethostname
Recupera el nombre del ordenador. Los backdoors a veces utilizan gethostname como parte de un estudio de la víctima.

GetProcAddress
Recupera la dirección de una función en una DLL cargada en memoria. Se utiliza para

importar funciones de otras DLL además de las funciones importadas en la cabecera del archivo PE.

GetStartupInfo
Recupera una estructura que contiene detalles sobre cómo se configuró el proceso actual para ejecutarse.

GetVersionEx
Devuelve información sobre qué versión de Windows se está ejecutando actualmente. Esto se puede utilizar como parte de un estudio de víctimas o para seleccionar diferentes flujos de ejecución según sea el tipo del sistema operativo víctima.

InternetOpenUrl
Abre una URL específica para una conexión mediante FTP, HTTP o HTTPS.

InternetReadFile
Lee los datos de una URL previamente abierta.

IsDebuggerPresent
Comprueba si el proceso actual está siendo depurado con un debugger, a menudo como parte de una técnica anti-debug.

IsNTAdmin
Comprueba si el usuario tiene privilegios de administrador.

LoadLibrary
Carga un DLL en un proceso que puede no haber sido cargado cuando el programa se inició. Su uso es bastante común.

LdrLoadDll
Función de bajo nivel para cargar una DLL en un proceso, al igual que LoadLibrary. Los programas normales utilizan LoadLibrary, así que la presencia de esta importación puede indicar un programa que intenta ser sigiloso.

LoadResource
Carga un recurso de un archivo PE en la memoria. El malware a veces utiliza los recursos para almacenar cadenas de texto, información de configuración u otros archivos maliciosos.

OpenProcess
Abre un manejador (handler) a otro proceso que se ejecuta en el sistema. Este manejador se puede utilizar para leer y escribir en memoria de otro proceso.

recv
Recibe datos de una máquina remota. El malware suele utilizar esta función para recibir datos de un servidor command-and-control.

send
Envía datos a una máquina remota. El malware suele utilizar esta función para enviar datos a un servidor command-and-control.

RegisterHotKey
Se utiliza para registrar un manejador que sea notificado cada vez que un usuario introduzca una combinación de teclas determinada (como CTRL-ALT-J), independientemente de la ventana que esté activa cuando el usuario pulse la combinación de teclas. Esta función es utilizada a veces por programas espía que permanecen ocultos al usuario hasta que se pulsa la combinación de teclas.

RegOpenKey
Abre un manejador a una clave de registro para su lectura y edición. Las claves del registro se escriben a veces como una forma de que el software logre persistencia. El registro también contiene bastante información de configuración del sistema operativo y de las aplicaciones.

SetWindowsHookEx
Establece una función gancho (hook) para que sea llamada cada vez que se llame a un determinado evento. Comúnmente utilizado con keyloggers y spyware.

ShellExecute
Se utiliza para ejecutar otro programa. Si el malware crea un nuevo proceso, se tendrá que analizar también el nuevo proceso.

URLDownloadToFile
Una llamada de alto nivel para descargar un archivo de un servidor web y guardarlo en el disco. Esta función es popular entre los “downloaders” porque implementa toda la funcionalidad de un descargador en una sola llamada a la función.

VirtualAllocEx
Una rutina de asignación de memoria que puede asignar memoria en un proceso remoto. El malware a veces utiliza VirtualAllocEx como parte de la inyección de código en otros procesos.

VirtualProtectEx
Cambia la protección de una región de memoria. El malware puede utilizar esta función para cambiar una sección de memoria de sólo lectura por un ejecutable.

Referencias

[1] Lista de funciones de la API de Windows: https://docs.microsoft.com/es-es/windows/win32/apiindex/windows-api-list (Último acceso el día 28/06/2022)

Anexo B – Lista de programas de uso específico

En este anexo se encuentran las herramientas que no se han incluido en el trabajo principal al ser más específicas, solo usarse en determinados casos o avanzadas. Por ejemplo, que un software este escrito en un lenguaje determinado (Como Delphi o AutoIt) o herramientas que extraigan información de bajo nivel, como por ejemplo, las posiciones de memoria de las secciones.

Herramientas

Interactive Delphi Reconstructor - IDR [1]: Herramienta que intenta reconstruir el código fuente desde el código Delphi compilado de un programa

ScyllaHide [2]: Biblioteca Anti-Anti-Depuración, es soportada por medio de plugins en OllyDbg, x64dbg e IDA

DbgChild [3]: DbgChild es una herramienta independiente para depurar procesos hijo.

PEStudio [4]: Detecta artefactos de archivos ejecutables para facilitar y acelerar la evaluación inicial de malware.

PE-Bear [5]: Herramienta gratuita de reversing de archivos PE.

Exe2Aut/AutoIt3 Decompiler [6]: Herramienta que permite extraer el código AutoIt de un ejecutable.

Hollows Hunter [7]: Analiza los procesos en ejecución y detecta una variedad de “implantes” potencialmente maliciosos como PEs reemplazados o implantados, shellcodes, hooks, parches en memoria…

Malwoverview [8]: Comprueba un ejecutable contra los principales sitios de análisis, para usar la mayoría es necesaria una API que se solicita en el sitio web de cada sitio. Genera un informe con toda la información encontrada.

Easyhunting [9]: Extrae las reglas Yara, Sigma y Suricata de la muestra que estemos analizando y tiene la capacidad de buscar muestras similares, lo que nos permite comprobar si estamos ante una nueva versión de algo que ya existía, identificar una campaña... etc.

Autopsy [10], Plataforma de análisis forense digital e interfaz gráfica para The Sleuth Kit (TSK) y otras herramientas de análisis forense digital.

Oledump [11], dedicada a analizar ficheros OLE (Object Linking and Embedding - enlazado e incrustación de objetos), los cuales pueden encontrarse insertados en ficheros de Microsoft Office, por ejemplo. Oledump permite analizar estos datos.

Uniextract [12], herramienta diseñada para extraer archivos de cualquier tipo de archivo extraíble. No se limita a archivos estándar como .zip y .rar. También puede manejar instaladores de aplicaciones o imágenes de disco

Wireshark [13], analizador de protocolos de red ampliamente utilizado.

Referencias

[1] https://github.com/crypto2011/IDR (Último acceso el día 30/06/2022)

[2] https://github.com/x64dbg/ScyllaHide (Último acceso el día 30/06/2022)

[3] https://github.com/therealdreg/DbgChild (Último acceso el día 30/06/2022)

[4] https://www.winitor.com/ (Último acceso el día 30/06/2022)

[5] https://hshrzd.wordpress.com/pe-bear/ (Último acceso el día 30/06/2022)

[6] http://domoticx.com/autoit3-decompiler-exe2aut/ (Último acceso el día 30/06/2022)

[7] https://github.com/hasherezade/hollows_hunter (Último acceso el día 30/06/2022)

[8] https://github.com/alexandreborges/malwoverview (Último acceso el día 30/06/2022)

[9] https://github.com/ppt0/easyhunting (Último acceso el día 30/06/2022)

[10] https://www.autopsy.com/ (Último acceso el día 30/06/2022)

[11] https://blog.didierstevens.com/programs/oledump-py/ (Último acceso el día 30/06/2022)

[12] https://github.com/Bioruebe/UniExtract2 (Último acceso el día 30/06/2022)

[13] https://www.wireshark.org/ (Último acceso el día 30/06/2022)

Anexo C – Información Extra

En este anexo se mostrarán algunos detalles de otras muestras distintas con el fin de compararlas entre sí y con la muestra “30a7ebb49157d0e668b72b47e97d88d0” vista en el TFG.

Si recordamos las características de esta última, vistas en el capítulo 13, tenemos que entre otras características destacan:

- Tiene un empaquetado desconocido o personalizado con alta entropía en la sección .text
- Comprende ofuscación en variables e importaciones de librerías
- Usa icono de Microsoft Outlook para intentar hacerse pasar por dicha herramienta
- Comentarios en VirusTotal referentes a Lokibot con enlaces a análisis externos.

Se detecta a 57/62 antivirus en Virustotal a día 22/06/2022 y está relacionada con un dominio potencialmente malicioso

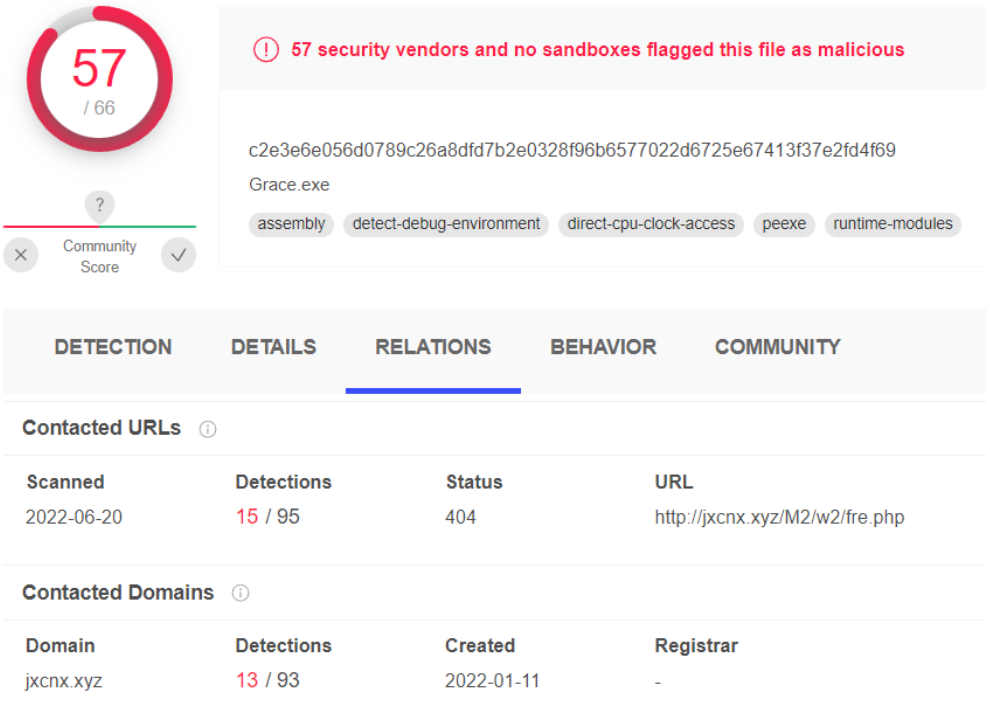


Figura 43. Relaciones de Grace.exe

Y la comunidad de VirusTotal lo relaciona con la amenaza Lokibot.

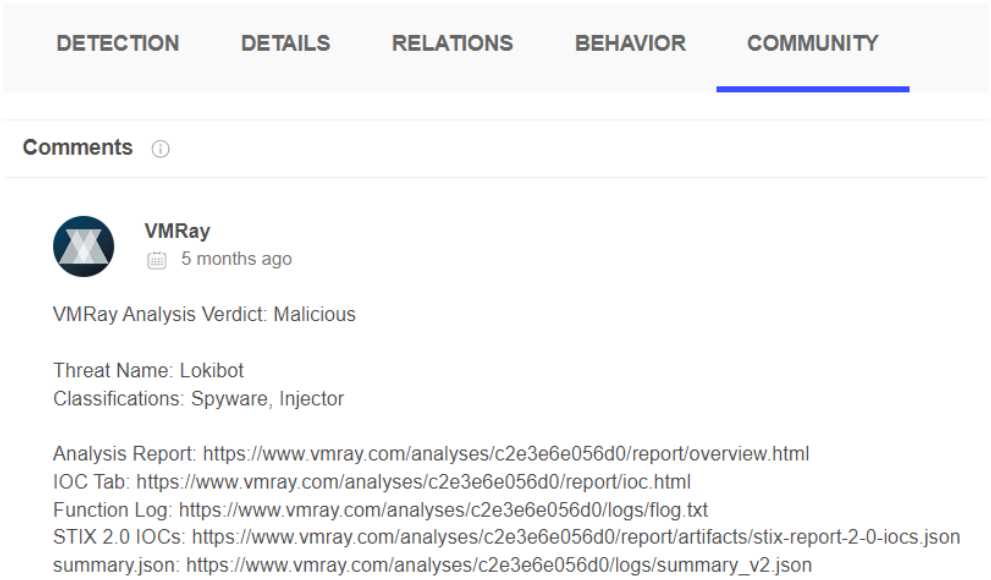


Figura 44. Comentario de la comunidad de VT sobre Grace.exe

A continuación se proveen de varios MD5 con el que comparar algunas características:

- MD5: “98c3385d313ae6d4cf1f192830f6b555”
  Se encuentra que a día 29/06/2022 tiene 48 detecciones y se relaciona como proceso hijo de ejecutables ampliamente detectados.

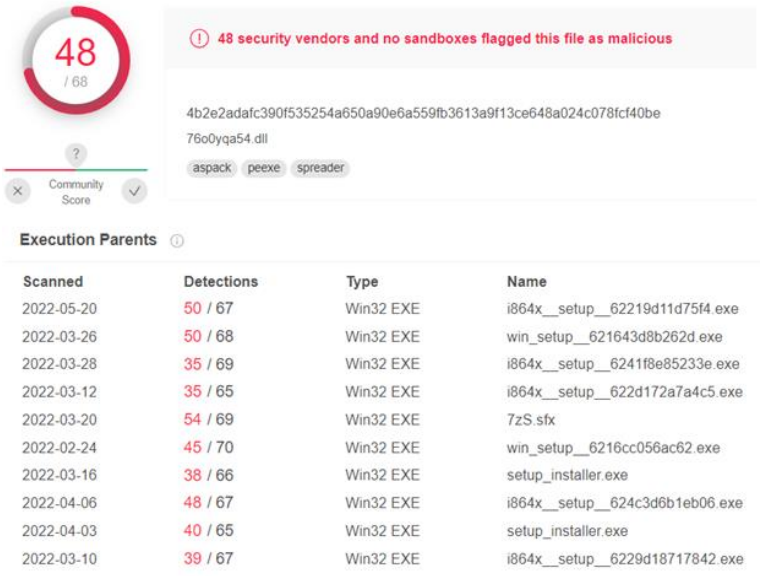


Figura 45. Detecciones VT y relaciones 98c3385d313ae6d4cf1f192830f6b555

Como podemos observar en dnSpy, esta muestra tiene .text con los permisos RWX activados, lo que implicaría la presencia de un malware. Si en vez de .text fuera otra sección propia de un packer, este comportamiento sería más aceptable.

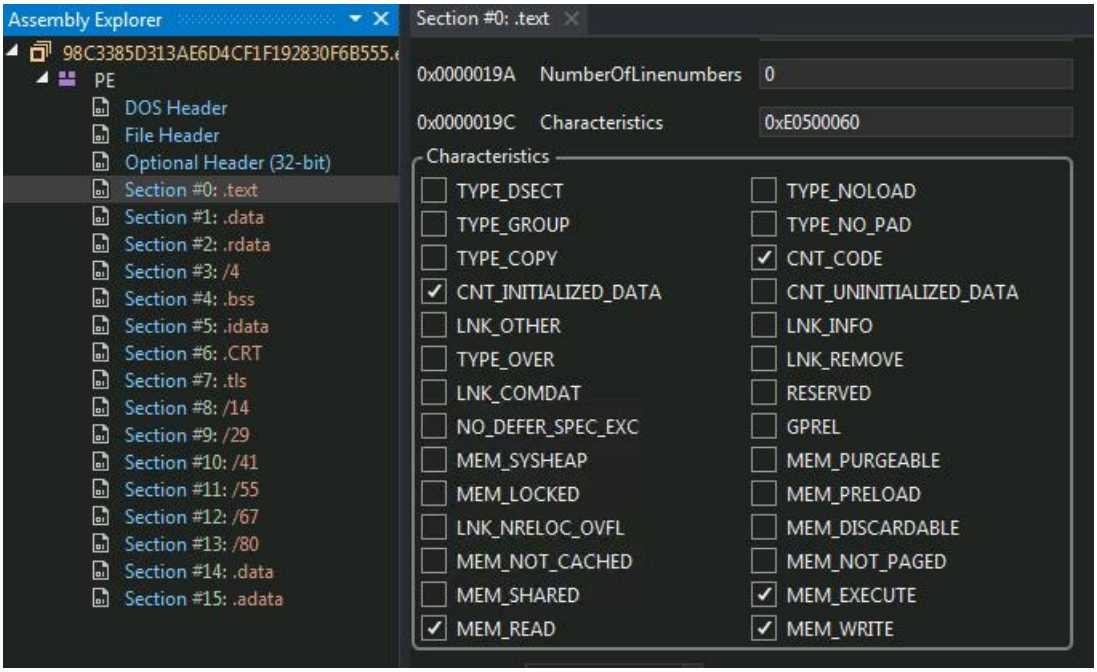


Figura 46. Die RWX

Las secciones con nombre extraño “slash-numero” en este caso probablementen sean creadas por el compilador.

- MD5: “9e380d087330997a4814e8377bbb242a”
  Se encuentra que a día 29/06/2022 tiene 46 detecciones y se relaciona con archivos posiblemente maliciosos que deja en el equipo.

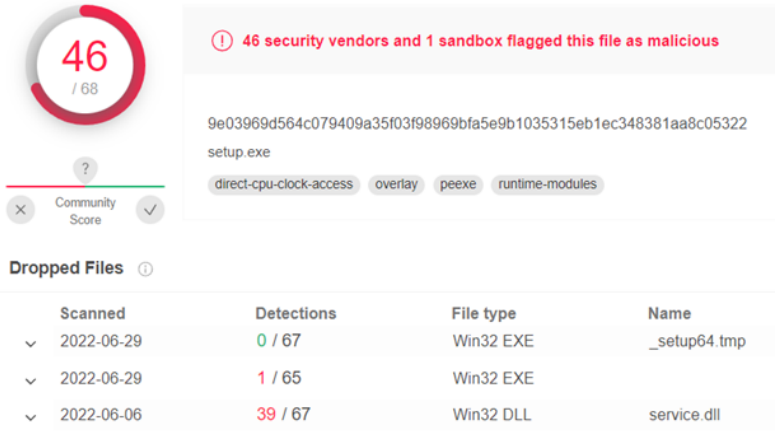


Figura 47: Detecciones VT y relaciones 9e380d087330997a4814e8377bbb242a

Como podemos observar en Resource Hacker, tiene cadenas de texto referentes a SetupLdr y Winsock, lo que podría implicar que descarga y ejecuta algún tipo de binario. También encontraremos referencias a un icono que parece relacionado con descargar algo, por lo que parece que estamos frente a un “downloader”.

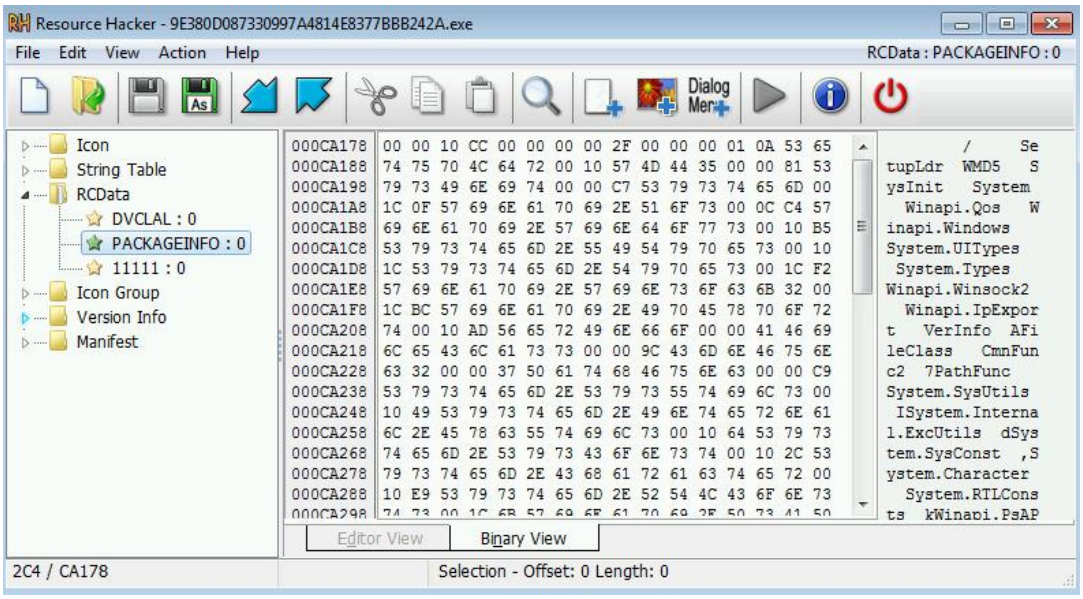


Figura 48. Cadenas de 9e380d087330997a4814e8377bbb242a

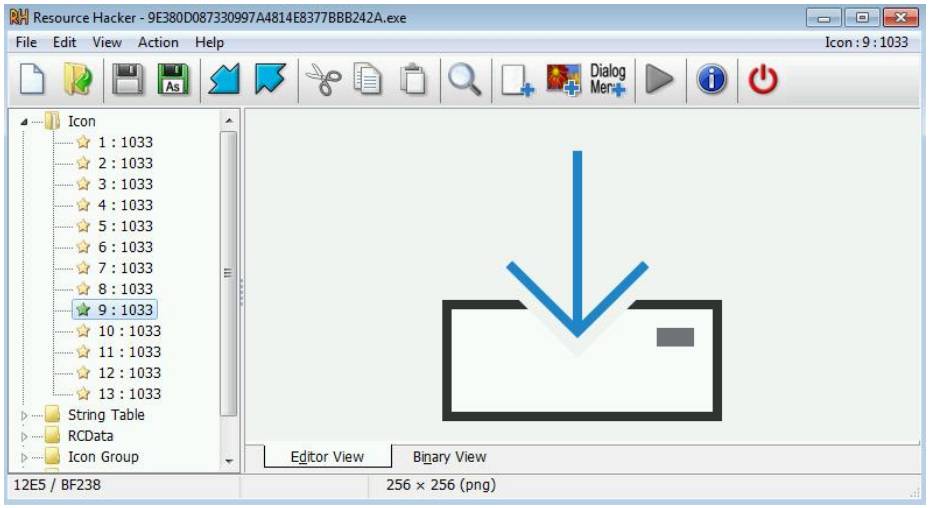


Figura 49. Icono de 9e380d087330997a4814e8377bbb242a

- MD5: “c128ca8bf368842af7b5c13e2f42cc12”
  Se encuentra que a día 29/06/2022 tiene 45 detecciones, además se ha encontrado con otros nombres sospechosos y se relaciona con otros archivos catalogados con anterioridad.

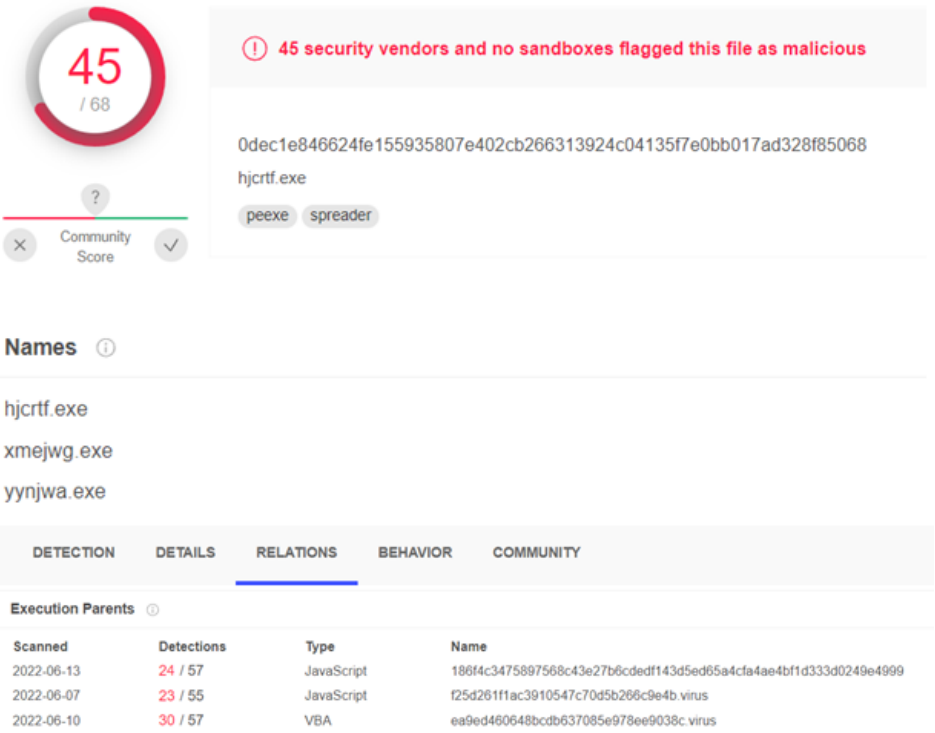


Figura 50. VT de c128ca8bf368842af7b5c13e2f42cc12

Con Die podemos comprobar como esta muestra no tiene packer conocido a pesar de mostrar una alta entropía y una posible ofuscación de sus cadenas de texto. Implicando así la presencia de un malware.

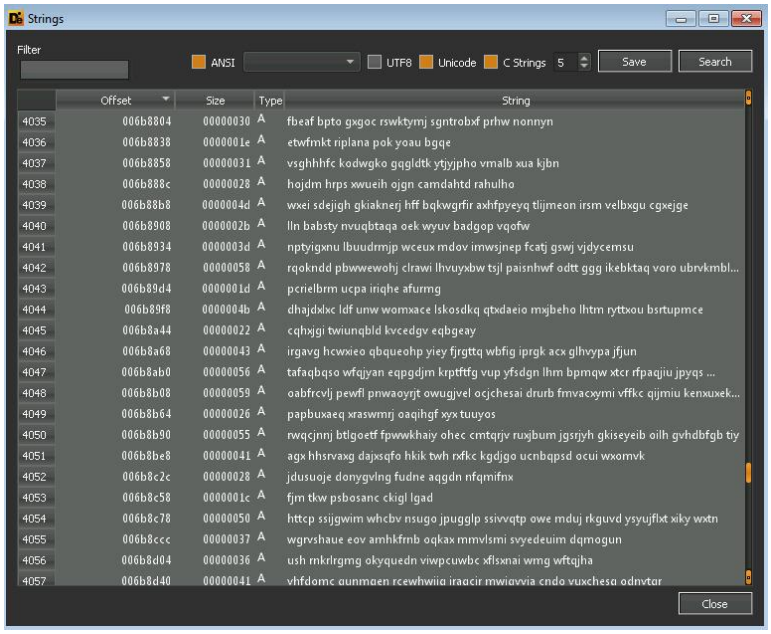


Figura 51. Cadenas con posible ofuscación

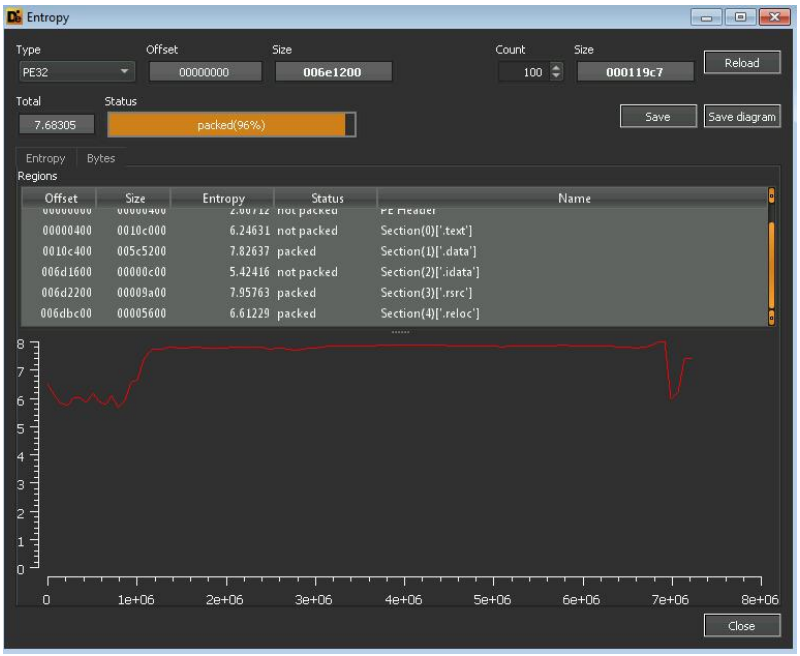


Figura 52. Entropía de c128ca8bf368842af7b5c13e2f42cc12

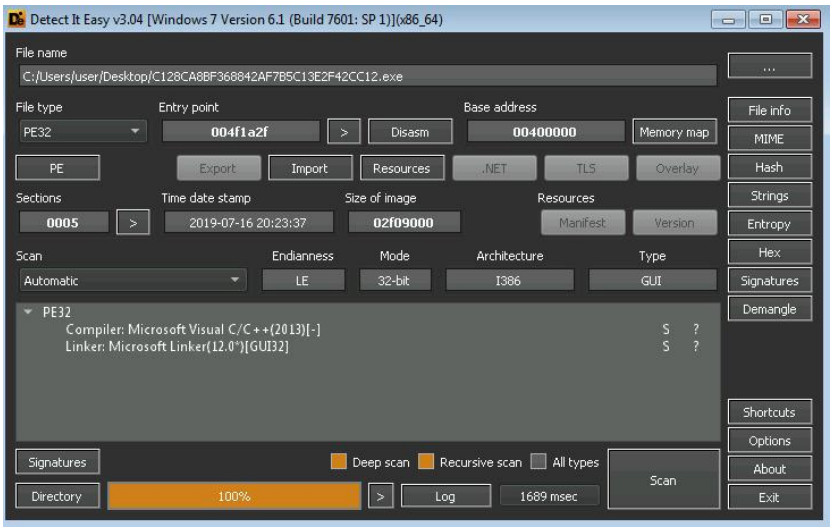


Figura 53. No detección de packer

- MD5: “0e343550d5c215c0c00ed0ae061f78c0”
  Se encuentra que a día 29/06/2022 tiene 60 detecciones, además se ha encontrado con comentarios de la comunidad que lo relacionan con una regla Yara existente en THOR APT Scanner

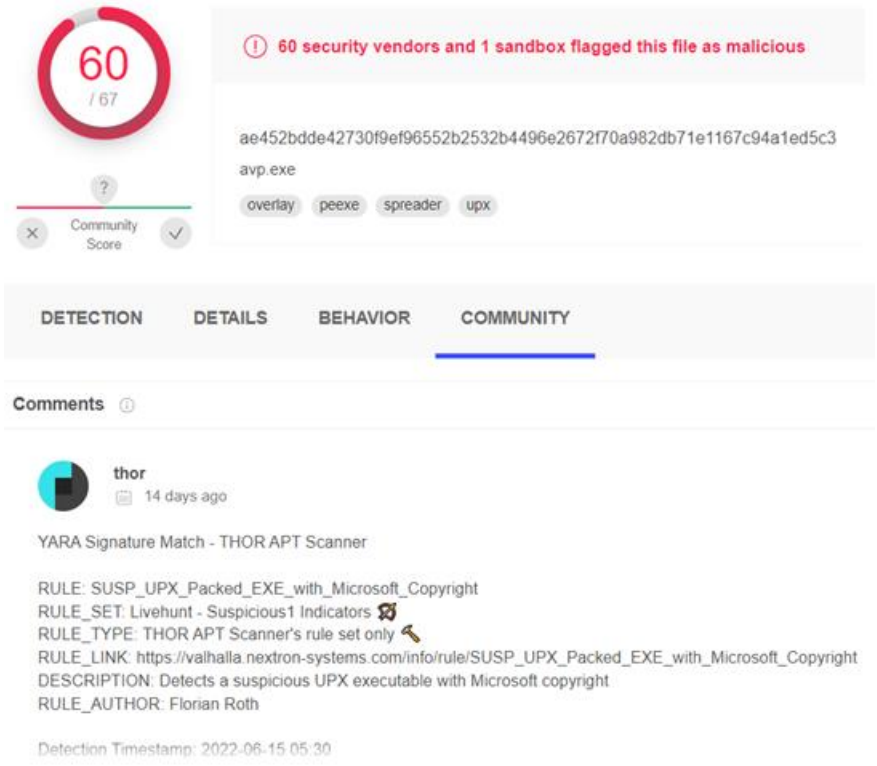


Figura 54. VT de 0e343550d5c215c0c00ed0ae061f78c0

En el análisis de esta muestra podemos observar que no tiene un icono usual, lo cual nos indica que posiblemente no es un programa legítimo. Si la detonamos, descubrimos que borra su propio ejecutable y replica su icono en algunas herramientas. Lejos de ser una broma, si hemos seguido de cerca sus pasos descubriremos que se ha asegurado persistencia en el registro, ha dropeado un fichero llamado CCTV.exe y tiene cadenas relativas al navegador Mozilla Firefox y a una librería llamada “Shellink.dll”.

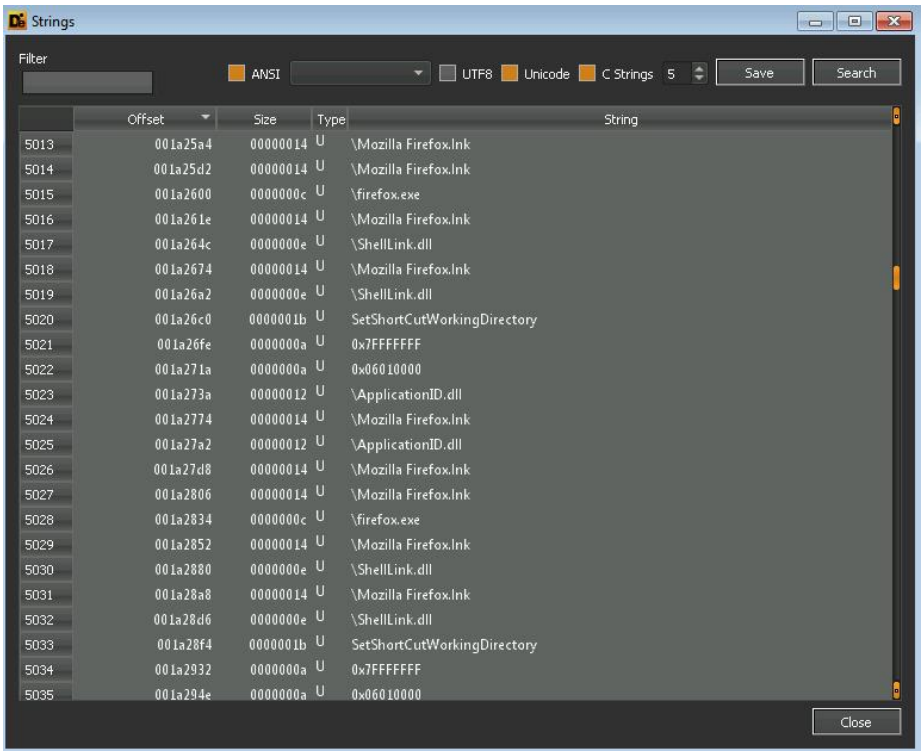


Figura 55. Strings de 0e343550d5c215c0c00ed0ae061f78c0

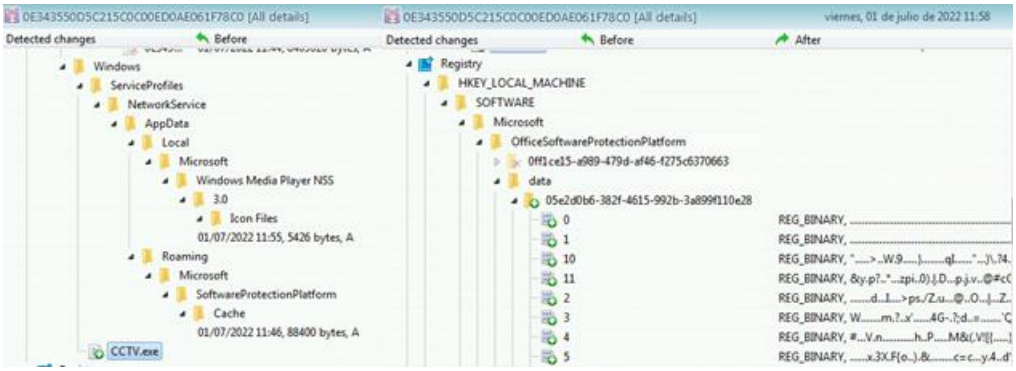


Figura 56. Dropeo de CCTV.exe y persistencia en registro

Figura 57. Modificación de iconos de las herramientas en el escritorio

Anexo D – Estructura PE

En este anexo se estudiará más detalladamente la estructura de un fichero PE. Este anexo sirve para profundizar en la estructura vista en el capítulo 5.

Para facilitar su lectura se provee de un pequeño índice con los campos en los que se divide la estructura.

Se ha extraído la información de la página de Aldeid [1]

Índice

Cabecera DOS (DOS Header)

Offset	Tamaño	Etiqueta	Descripción
0x00	WORD	emagic	Número mágico DOS firma MZ (0x4d 0x5A)
0x02	WORD	e_cblp	Bytes de la última página del fichero
0x04	WORD	e_cp	Páginas del fichero
0x06	WORD	e_crlc	Reubicaciones
0x08	WORD	e_cparhdr	Tamaño de la cabecera en párrafos
0x0A	WORD	e_minalloc	Mínimo de párrafos adicionales necesarios
0x0C	WORD	e_maxalloc	Máximo de párrafos adicionales necesarios
0x0E	WORD	e_ss	Valor inicial (relativo) de SS (registro)
0x10	WORD	e_sp	Valor inicial del SP (registro)
0x12	WORD	e_csum	"Checksum" / Suma de comprobación
0x14	WORD	e_ip	Valor inicial de IP
0x16	WORD	e_cs	Valor inicial (relativo) del CS (registro)
0x18	WORD	e_lfarlc	Dirección del archivo de la tabla de reubicación
0x1A	WORD	e_ovno	Número de "Overlay" (Superposición)
0x1C	WORD	e_res[4]	Espacio reservado (4 WORDs)
0x24	WORD	e_oemid	Identificador OEM
0x26	WORD	e_oeminfo	Información OEM
0x28	WORD	e_res2[10]	Espacio reservado (10 WORDs)
0x3c	DWORD	e_lfanew	Desplazamiento al inicio de la cabecera PE

Cabecera PE (PE Header)

Offset	Tamaño	Etiqueta	Descripción
Cabecera de fichero (Image File Header) (0x00)			
0x00	DWORD	Signature	Valor/Número mágico PE
0x04	WORD	Machine	Valor Id referente a un tipo de máquina (diferentes fabricantes/arquitecturas)
0x06	WORD	NumberOfSections	Número de secciones
0x08	DWORD	TimeDateStamp	Valor que indica la fecha de creación del fichero
0x0C	DWORD	PointerToSymbolTable	El desplazamiento del archivo de la tabla de símbolos COFF, o cero si no hay tabla de símbolos COFF.
0x10	DWORD	NumberOfSymbols	El número de entradas en la tabla de símbolos. Este dato puede utilizarse para localizar la tabla de cadenas, que sigue inmediatamente a la tabla de símbolos.
0x14	WORD	SizeOfOptionalHeader	El tamaño de la cabecera opcional, que es necesaria para los archivos ejecutables pero no para los archivos objeto.
0x16	WORD	Characteristics	Las banderas "flags" que indican los atributos del archivo.

Cabecera opcional (Optional Header)

Offset	Tamaño	Etiqueta	Descripción
0x18	WORD	Magic	El entero sin signo que identifica el tipo del archivo. El número más común es 0x10B, que lo identifica como un archivo ejecutable normal (PE32). 0x107 lo identifica como una imagen ROM, y 0x20B lo identifica como un ejecutable PE32+.
0x1A	BYTE	MajorLinkerVersion	El número máximo de versión del enlazador (linker).
0x1B	BYTE	MinorLinkerVersion	El número mínimo de versión del enlazador (linker).
0x1C	DWORD	SizeOfCode	El tamaño de la sección de código (.text), o la suma de todas las secciones de código si hay varias secciones.
0x20	DWORD	SizeOfInitializedData	El tamaño de la sección de datos inicializada (.data), o la suma de todas esas secciones si hay varias secciones de datos (idata, edata...)
0x24	DWORD	SizeOfUninitializedData	El tamaño de la sección de datos no inicializados (BSS), o la suma de todas esas secciones si hay varias secciones BSS.
0x28	DWORD	AddressOfEntryPoint	La dirección del punto de entrada relativa a la base de la imagen cuando el archivo ejecutable se carga en la memoria. Para las imágenes de programa, es la dirección de inicio. Para los controladores de dispositivos, es la dirección de la función de inicialización.
0x2C	DWORD	BaseOfCode	La dirección relativa a la dirección base de la sección de inicio de código cuando se carga en la memoria.
0x30	DWORD	BaseOfData	Este campo no aparece en PE32+. La dirección que es relativa a la base de la sección de inicio de datos cuando se carga en la memoria.
0x34	DWORD	ImageBase	La dirección predeterminada del primer byte de la imagen cuando se carga en la memoria; debe ser un múltiplo de 64K. El valor predeterminado para las DLL es 0x10000000. El valor predeterminado para los EXE es 0x00400000.
0x38	DWORD	SectionAlignment	La alineación (en bytes) de las secciones cuando se cargan en la memoria. Debe ser mayor o igual que “FileAlignment”. El valor por defecto es el tamaño de la página para la arquitectura.
0x3C	DWORD	FileAlignment	El factor de alineación (en bytes) que se utiliza para alinear los datos brutos de las secciones del archivo de imagen. El valor debe ser una potencia de 2 entre 512 y 64K, ambos inclusive. El valor por defecto es 512. Si “SectionAlignment” es menor que el tamaño de página de la arquitectura, entonces “FileAlignment” debe coincidir

			con “SectionAlignment”.
0x40	WORD	MajorOperatingSystemVersion	El número de versión máximo requerido del sistema operativo, para determinar si se ejecuta o no.
0x42	WORD	MinorOperatingSystemVersion	El número de versión mínimo del sistema operativo, para determinar si se ejecuta o no.
0x44	WORD	MajorImageVersion	El número de versión máximo de la imagen.
0x46	WORD	MinorImageVersion	El número de versión mínimo de la imagen.
0x48	WORD	MajorSubsystemVersion	El número de versión máximo del subsistema.
0x4A	WORD	MinorSubsystemVersion	El número de versión mínimo del subsistema.
0x4C	DWORD	Win32VersionValue	Reservado, debe de ser cero.
0x50	DWORD	SizeOfImage	El tamaño (en bytes) de la imagen, incluyendo todas las cabeceras, cuando la imagen se carga en la memoria. Debe ser un múltiplo de “SectionAlignment”.
0x54	DWORD	SizeOfHeaders	El tamaño combinado de un “stub” de MS DOS, la cabecera PE y las cabeceras de sección redondeado a un múltiplo de “FileAlignment”.
0x58	DWORD	CheckSum	La suma de comprobación del archivo de imagen. (Valor)
0x5C	WORD	Subsystem	El subsistema que se requiere para ejecutar esta imagen. (Valor)
0x5E	WORD	DllCharacteristics	Características de la DLL. (Valor)
0x60	DWORD	SizeOfStackReserve	El tamaño de la pila a reservar. Sólo se reserva “SizeOfStackCommit”; el resto se pone a disposición una página cada vez hasta alcanzar el tamaño máximo de reserva.
0x64	DWORD	SizeOfStackCommit	El tamaño de la pila.
0x68	DWORD	SizeOfHeapReserve	El tamaño del espacio de la pila local a reservar. Sólo se compromete “SizeOfHeapCommit”; el resto se pone a disposición una página cada vez hasta alcanzar el tamaño de reserva.
0x6C	DWORD	SizeOfHeapCommit	El tamaño del espacio de la pila local.
0x70	DWORD	LoaderFlags	Reservado, debe ser cero.
0x74	DWORD	NumberOfRvaAndSizes	El número de entradas del directorio de datos en el resto de la cabecera opcional. Cada una describe una ubicación y un tamaño. **Nota:** El malware puede establecer un valor inválido para esta etiqueta para bloquear el depurador.

Directorios de datos (Data Directories)

Offset	Tamaño	Etiqueta	Descripción
0x78	DWORD	Export Table	Dirección Virtual Relativa del Directorio de Exportación
0x7C	DWORD		Tamaño del directorio de exportación
0x80	DWORD	Import Table	Dirección Virtual Relativa del Directorio de Importación (array de identificadores)
0x84	DWORD		Tamaño del directorio de importación (array de identificadores)
0x88	DWORD	Resource Table	Dirección Virtual Relativa del directorio de recursos
0x8C	DWORD		Tamaño del directorio de recursos
0x90	DWORD	Exception Table	Dirección Virtual Relativa del directorio de excepciones
0x94	DWORD		Tamaño del directorio de excepciones
0x98	DWORD	Certificate Table	Desplazamiento en bruto del directorio de seguridad
0x9C	DWORD		Tamaño del directorio de seguridad
0xA0	DWORD	Base Relocation Table	Dirección Virtual Relativa del Directorio de Reubicación Base
0xA4	DWORD		Tamaño del Directorio de Reubicación Base
0xA8	DWORD	Debug	Dirección Virtual Relativa del Directorio de Depuración
0xAC	DWORD		Tamaño del Directorio de Depuración
0xB0	DWORD	Architecture	Dirección Virtual Relativa de la nota de derechos de autor
0xB4	DWORD		Tamaño de la nota de derechos de autor
0xB8	DWORD	Global Ptr	Dirección Virtual Relativa a utilizar como puntero global (sólo IA-64)
0xBC	DWORD		No se utiliza
0xC0	DWORD	TLS Table	Dirección Virtual Relativa del TLS (Thread Local Storage)
0xC4	DWORD		Tamaño del TLS (Thread Local Storage)
0xC8	DWORD	Load Config Table	Dirección Virtual Relativa del directorio de configuración
0xCC	DWORD		Tamaño del directorio de configuración
0xD0	DWORD	Bound Import	Dirección Virtual Relativa de Directorio de Importación enlazado
0xD4	DWORD		Tamaño de Directorio de Importación enlazado
0xD8	DWORD	IAT	Dirección Virtual Relativa de la primera tabla de direcciones de importación
0xDC	DWORD		Tamaño total de todas las tablas de direcciones de importación
0xE0	DWORD	Delay Import Descriptor	Dirección Virtual Relativa del Directorio de Importación retrasada
0xE4	DWORD		Tamaño del directorio de importación retrasada
0xE8	DWORD	CLR Runtime Header	Dirección Virtual Relativa de la cabecera COM (información de nivel superior y metadatos...)
0xEC	DWORD		tamaño de la cabecera COM (en los ejecutables .NET)
0xF0	DWORD	ZERO (Reservado)	Reservado
0xF4	DWORD	ZERO (Reservado)	Reservado

Tabla de secciones (Sections Table) (a partir de 0xF8)

Offset	Tamaño	Etiqueta	Descripción
0x00	DWORD	Name	Una cadena codificada en UTF-8 de 8 bytes con relleno “nulo”.
0x08	DWORD	VirtualSize	El tamaño total de la sección cuando se carga en memoria. Si este valor es mayor que “SizeOfRawData”, la sección se rellena con cero.
0x0C	DWORD	VirtualAddress	Para imágenes ejecutables, la dirección del primer byte de la sección relativa a la base de la imagen cuando la sección se carga en la memoria.
0x10	DWORD	SizeOfRawData	Tamaño de los datos inicializados en el disco. **Nota:** El malware puede establecer un valor inválido para esta etiqueta para bloquear el depurador.
0x14	DWORD	PointerToRawData	El puntero del archivo a la primera página de la sección dentro del archivo COFF. Para las imágenes ejecutables, debe ser un múltiplo de “FileAlignment” de la cabecera opcional.
0x18	DWORD	PointerToRelocations	El puntero al comienzo de las entradas de reubicación para la sección.
0x1C	DWORD	PointerToLinenumbers	El puntero al comienzo de las entradas de números de línea para la sección.
0x20	WORD	NumberOfRelocations	El número de entradas de reubicación para la sección.
0x22	WORD	NumberOfLineNumbers	El número de entradas de número de línea para la sección.
0x24	DWORD	Characteristics	Valor que describe las características de la sección.
(Repetir la estructura para el resto de secciones)			
0x00	8 Bytes	Name	Nombre de la segunda/tercera/cuarta... sección
...	...	...	...

Referencias

[1] https://www.aldeid.com/wiki/PE-Portable-executable (Último acceso el día 30/06/2022)

Anexo E – Objetivos de Desarrollo Sostenible

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. **Fin de la pobreza.**				X
ODS 2. **Hambre cero.**				X
ODS 3. **Salud y bienestar.**				X
ODS 4. **Educación de calidad.**	X			
ODS 5. **Igualdad de género.**				X
ODS 6. **Agua limpia y saneamiento.**				X
ODS 7. **Energía asequible y no contaminante.**				X
ODS 8. **Trabajo decente y crecimiento económico.**	X			
ODS 9. **Industria, innovación e infraestructuras.**				X
ODS 10. **Reducción de las desigualdades**.				X
ODS 11. **Ciudades y comunidades sostenibles**.				X
ODS 12. **Producción y consumo responsables.**				X
ODS 13. **Acción por el clima.**				X
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**				X
ODS 16. **Paz, justicia e instituciones sólidas.**				X
ODS 17. **Alianzas para lograr objetivos.**				X

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

De los Objetivos de Desarrollo Sostenible aprobados por la Organización de las Naciones Unidas en la Agenda 2030, este TFG se relacionaría con:

- **Educación de Calidad**: Personalmente creo que la idea presentada proporciona un contenido al que una a persona le puede aportar valor y como tal, se espera que dicha persona se plantee pagar por dicho contenido. Especialmente si este contenido se encuentra encapsulado en un temario de un curso, dado que este TFG trata sobre creación de un entorno de análisis, sienta las bases para futuros análisis de malware más complejos y dada la variedad de conocimientos que podrían requerirse (Informática forense, redes, sistemas…) en el momento del análisis, estaríamos frente al caldo de cultivo perfecto para la creación de un curso o asignatura referente al análisis de malware. La inclusión de los anexos también indica que hay mucha más información que puede integrarse en un temario de una asignatura reglada, tanto por profundidad, a la que no se ha llegado en el TFG por considerarse más avanzada que lo que se quería exponer, como por amplitud, dado que de apartados como el "Análisis Dinámico" se podría generar información que daría para otros TFG, TFM, asignaturas o cursos debido a las formas que el malware posee y desarrolla para esconderse, y técnicas y programas que se crean cada día para poder detectar las nuevas capacidades. Actualmente, existen cursos relacionados con análisis de malware, tanto básicos como avanzados, que se venden. Sirva como ejemplo la plataforma de educación [1] que posee Kaspersky, con cursos como "*Advanced Malware Analysis Techniques*" o "*Targeted Malware Reverse Engineering*", relacionados directamente con el análisis de malware.

- **Trabajo Creciente y Crecimiento Económico**: Como se ha comentado en el apartado "Motivación" de este TFG, esta ciencia no es muy popular. Tanto por la proliferación de otros perfiles técnicos relacionados, como lo pueden ser perfiles "Blue Team" de SOC (Security Operations Center) o "Red Team" como los "Pentesters" (Seguridad Ofensiva) con más fama, como por la alta dificultad y el largo tiempo de aprendizaje que requiere el área del reversing (ingeniería inversa), bastante relacionado con el análisis de malware. Este TFG, al iniciar al futuro analista en otras áreas como lo pueden ser informática forense, sistemas, redes… antes que en el área de reversing, permite comenzar a trabajar con dichos conocimientos y al mismo tiempo ir ganando experiencia en el campo de la ingeniería inversa, por lo que no hace falta esperar a controlar el área del reversing para poder trabajar de analista de malware. A su vez, el trabajo de analista de malware, dado que está dentro del área de TI y además hay poca gente que lo ejerza en comparación a otros trabajos relacionados, son profesionales más buscados y cotizados por algunas compañías del sector, llegando a ofrecer salarios altos para integrar a estos profesionales en su plantilla.

Referencias

[1] https://xtraining.kaspersky.com/ (Último acceso el día 30/06/2022)