Universitat Oberta de Catalunya

uoc.edu

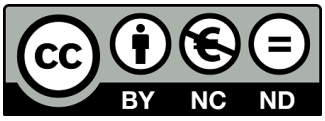
Máster interuniversitario de seguridad de las tecnologías de la información y de las comunicaciones

Trabajo Fin de Máster

"Análisis de familias de ransomware"

Guillermo Cebollero Abadías

Director:
Richard Rivera
Barcelona, 2020



A mis padres, quienes siempre me han apoyado.
A Ester, quien me ha animado en cada bache y me ha dado las fuerzas para continuar.

A todos aquellos que me han ayudado, dado consejo, apoyo y me han ayudado a seguir soñando con la ciberseguridad

Guillermo

Índice general

Dedicatoria **2**

Lista de Tablas **III**

Lista de Figuras **IV**

1. Introducción **3**

1.1. Objetivos 3

1.2. Alcance 5

1.3. Estructura del documento 6

2. Estado del Arte **7**

3. Enfoque **10**

3.1. Arquitectura del entorno 10

3.2. Muestras a analizar 12

3.3. Configuración del anfitrión 13

3.4. Realización de los experimentos 14

3.5. Agregación de los resultados obtenidos 16

4. Resultados **18**

4.1. Análisis general de las muestras 18

4.1.1. Tipología de las muestras 19

4.1.2. Tiempo de análisis 20

4.1.3. Puntuación de las muestras obtenidas 21

4.1.4. VirusTotal scoring 23

4.2. Análisis estático . . . 24
4.2.1. Secciones del ejecutable . . . 25
4.2.2. Importaciones de librerías . . . 26
4.2.3. *Clustering* de muestras mediante `imphash` . . . 28
4.2.4. Otras técnicas de agregación . . . 30
4.3. Análisis dinámico . . . 30
4.3.1. Registro del sistema . . . 32
4.3.2. Árbol de procesos . . . 34
4.3.3. Tráfico de red . . . 38
4.3.4. Análisis del tráfico de red por protocolo . . . 38
4.3.5. Tráfico externo . . . 42
4.4. Firmas . . . 43
4.5. Validación externa . . . 46
4.5.1. imphash . . . 46
4.5.2. Ficheros empaquetados . . . 47
4.5.3. Carga dinámica de librerías . . . 48
4.5.4. Peticiones a *dead hosts* . . . 48
4.5.5. Clasificación de las muestras . . . 49

5. Conclusiones y Trabajo Futuro **51**
5.1. Conclusiones . . . 51
5.2. Trabajo Futuro . . . 54

Bibliografía **55**

A. Apéndice **59**

Índice de tablas

3.1. Búsquedas realizadas en HybridAnalysis 13

4.1. Top 15 categorías de los ficheros analizados 19
4.2. Estudio de la duración de los experimentos 21
4.3. Análisis de las puntuaciones obtenidas con `Cuckoo` 23
4.4. Análisis de las puntuaciones obtenidas en VirusTotal 24
4.5. Secciones de los binarios más comunes 26
4.6. Importaciones de *dll* 27
4.7. Funciones más utilizadas. Todas en *kernel32.dll* 28
4.8. Agregación de muestras que comparten el valor `imphash` 30
4.9. Cadenas *pdb* sospechosas 31
4.10. Total de acciones en el registro detectadas 32
4.11. Cambios de registro maliciosos más detectados 35
4.12. Procesos ejecutados por las muestras 38
4.13. Tráfico `UDP` 41
4.14. Tráfico `TCP` 41
4.15. TOP 10 conexiones HTTP 42
4.16. Firmas 44
4.17. Firmas más detectadas durante los análisis 45

A.1. *Dead host* contactados 59
A.2. *Reverse resolutions* de los *dead hosts* 60

Índice de figuras

1.1. Diagrama Gantt del proyecto 4

3.1. Arquitectura propuesta para el análisis de las muestras. 11

3.2. Arquitectura propuesta para el procesamiento de los resultados. 12

4.1. Distribución en tiempo de análisis 22

4.2. Resultados por puntuación en `Cuckoo` 24

4.3. Resultados por puntuación en VirusTotal 25

4.4. Importaciones de *dll* 27

4.5. Resultado de la agregación por valores `imphash` 29

4.6. Acciones en el registro observadas. Escala logarítmica en ocurrencias. 33

4.7. Procesos ejecutados por las muestras 36

4.8. Representación del tráfico capturado 39

4.9. Distribución de las firmas detectadas 44

4.10. Puntuación promedio en VirusTotal agrupado por `imphash` 46

4.11. Firma `UPX`, distribución en VirusTotal 47

4.12. Firma *PEB Modified*, distribución en VirusTotal 48

4.13. Firma *dead hosts*, distribución en VirusTotal 49

4.14. Distribución de *AVClass* de las muestras 50

Resumen

En los últimos años el número de ataques de *ransomware* así como nuevas variantes de *malware* ha crecido de forma imparable.

Este aumento ha provocado que las técnicas clásicas de detección y clasificación de amenazas se hayan visto obsoletas. El análisis de las muestras debe hacerse de forma automática sobre conjuntos incluyendo las nuevas variantes constantemente.

En este estudio presenta los resultados del análisis automatizado de un conjunto de más de 2000 muestras mediante *Cuckoo Sandboxing* y su posterior agregado y análisis.

Desde el punto de vista del análisis estático, se presentan las principales características y capacidades extraídas de las cabeceras de los ficheros ejecutables. Adicionalmente se presentan dos técnicas básicas de *clustering* de muestras a través de la información extraída previamente.

Posteriormente se detalla la información obtenida a través del análisis dinámico, las técnicas de reconocimiento, explotación, movimiento lateral y comunicaciones con servidores de mando y control observados. Adicionalmente se presentan los resultados obtenidos gracias las firmas modelando patrones de comportamiento y acciones maliciosas conocidas. Para finalizar, se comparan los resultados obtenidos con fuentes de inteligencia y herramientas externas permitiendo la obtención de nuevas conclusiones.

Palabras clave

Cuckoo, Ransomware, Malware, Análisis, Estático, Dinámico, Firmas.

Abstract

In recent years the number of *ransomware* attacks as well as new *malware* variations have grown unstoppable.

This increase has made the classical threat detection and classification techniques obsolete. Analysis of samples must be done automatically on sets including new variants constantly.

This study shows the results of the automated analysis of a set with more than 2000 samples using *Cuckoo Sandboxing* and the subsequent aggregation and analysis.

From the point of view of static analysis, the main characteristics and capabilities extracted from the executable file headers are introduced. Additionally two simple clustering techniques of samples with the information previously extracted are shown.

Later, the information obtained from dynamic analysis is detailed, the reconnaissance techniques, explotation, lateral movement and communications with command and control servers. Additionally the results obtained from signatures modeling behaviour patterns and known malicious actions. At the end, the results are compared with external intelligence sources and tools in order to make new conclusions.

Keywords

Cuckoo, Ransomware, Malware, Analysis, Static, Dynamic, Signatures.

Capítulo 1

Introducción

Cada día emergen alrededor de un millón de nuevas variantes de *malware*. De esta gran cantidad de nuevas muestras, muy pocas son en realidad corresponden a nuevas familias, la mayoría son variantes de *malware* ya conocido [21]. Los desarrolladores de *malware* constantemente están aplicando técnicas de polimorfismo y ofuscación para crear variantes del *malware* en un intento de evitar la detección por los antivirus.

Este TFM consiste en analizar un *dataset* de *ransomware* y *malware* en un sistema de análisis automatizado de malware como *Cuckoo Sandbox* [28] para detectar patrones que permitan realizar su clasificación y detectar vectores de ataque.

1.1. Objetivos

Para la realización de este trabajo se han definido una serie de objetivos a realizar:

- Instalación de *Cuckoo Sandbox* y configuración de sus dependencias.
- Optimización de los experimentos.
- Análisis de muestras de *Ransomware* y *malware*.
- Extracción de características.
- Análisis de los resultados y obtención de patrones de comportamiento.

- Comparación y validación de los resultados obtenidos con herramientas externas.

Estos objetivos se pueden representar mediante un diagrama *Gantt* tal y como muestra la Figura 1.1, permitiendo delimitar los objetivos en una serie de franjas temporales y detectar dependencias entre ellos antes de su realización.

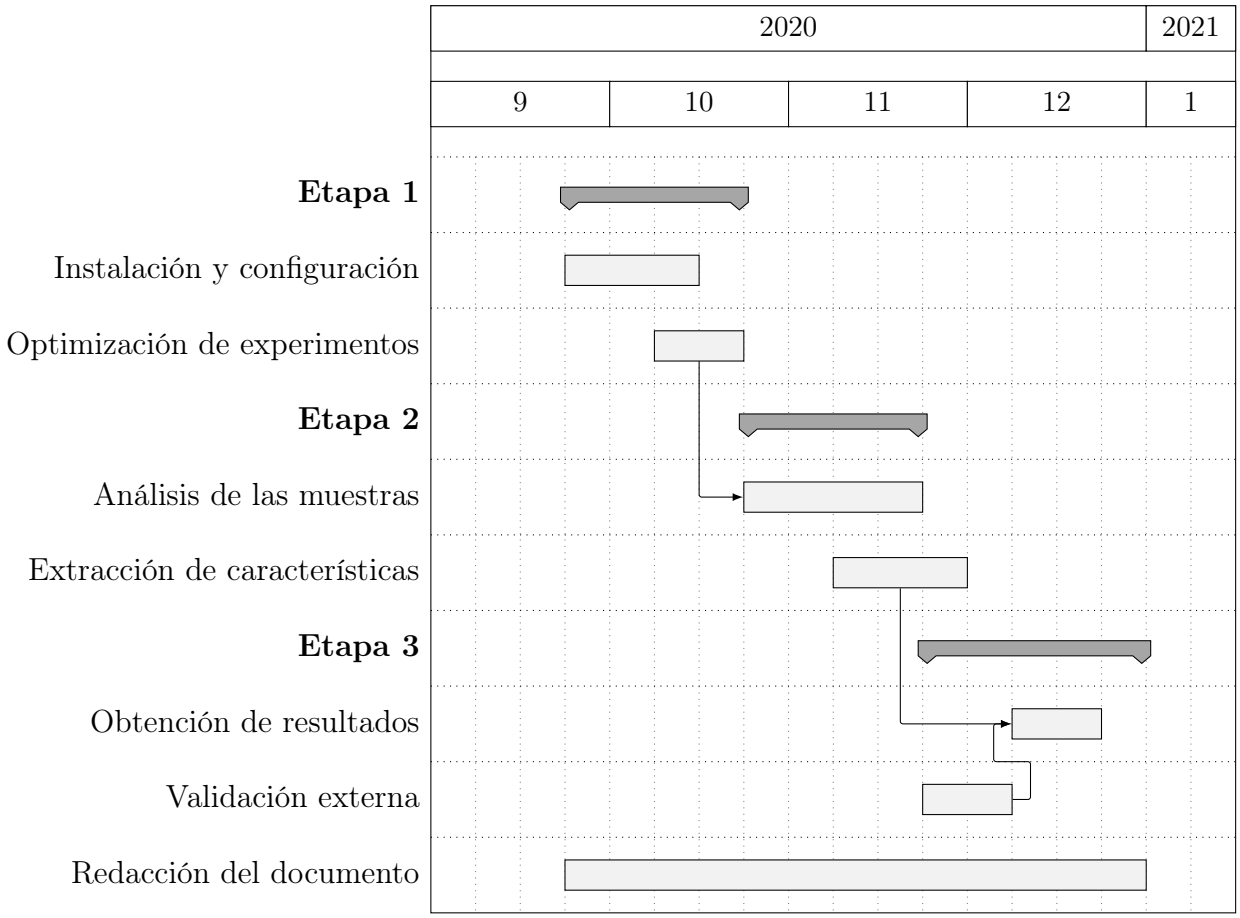


Figura 1.1: *Diagrama Gantt del proyecto*

Como se puede observar, se deberá acabar la optimización de los experimentos antes de comenzar el análisis de las muestras y del mismo modo, existe una dependencia entre la extracción de características, la validación externa y la obtención de resultados.

1.2. Alcance

Para acotar la realización de este trabajo, se han definido una serie de metas temporales definidas a continuación.

Durante la realización de la primera etapa, se procederá a la instalación de *Cuckoo* así como sus dependencias para obtener la máxima información posible de cada análisis. Dicha instalación se realizará en un sistema anfitrión basado en *Ubuntu Server 18.04 LTS* y como sistema virtualizado un Windows 7. Esta etapa también incluye la configuración del entorno virtualizado para favorecer dicho análisis, la cantidad de recursos destinados así como la optimización de la cantidad de experimentos que se realizarán de forma concurrente.

Durante la segunda etapa, se definirá un conjunto inicial de muestras para variar las condiciones del experimento, incluyendo la duración del experimento, simulación de entrada humana, análisis de la memoria del sistema y tipo de paquete a analizar.
Una vez definidas las condiciones iniciales óptimas, se realizará el analisis de todas las muestras obtenidas, agregando la información obtenida y realizando un estudio previo de la información obtenida.
Después de obtener el conjunto de datos agregados se aplicarán técnicas de extracción de características y patrones comunes entre las categorías del malware.

Como fase final, será necesario el estudio de los datos obtenidos contrastandolo con herramientas externas y la obtención de las conclusiones finales.

La redacción de la memoria se realizará de forma trasversal a todas las fases del proyecto.

1.3. Estructura del documento

Una vez definidos los objetivos y el alcance de este trabajo, en el Capítulo 2 se hará una breve revisión del estado del arte actual, introduciendo conceptos y herramientas utilizados en el análisis y clasificación del *malware*.

A continuación, en el Capítulo 3 se introducirá el enfoque y metodología utilizados para el análisis de las muestras y la obtención de resultados. Adicionalmente se matizarán aspectos del entorno de pruebas elegido.

En el Capítulo 4 se detallarán los resultados obtenidos tras el análisis y agregación de resultados, habiéndose organizado en los resultados obtenidos durante el análisis estático, análisis dinámico o basados en firmas incluyendo además las validaciones realizadas con herramientas y fuentes de conocimiento externas.

Para finalizar, en el Capítulo 5 se detallan brevemente las conclusiones obtenidas así como los trabajos futuros propuestos.

Capítulo 2

Estado del Arte

Los ataques basados en una infección por ransomware están ganando popularidad entre los cibercriminales en los últimos años. Mientras que en 2016 se producía un ataque cada 40 segundos [5] mientras que se estima que durante el 2021 se elevarán a un ataque cada 11 segundos suponiendo un coste total de 20 billones americanos de dólares en daños [30].

En orden de obtener un beneficio económico, los grupos organizados de cibercriminales no cesan de liberar nuevas variantes incluyendo nuevos vectores de infección, cambios en el modelo de comportamiento, nuevas técnicas de evasión y ofuscación. Es el ejemplo de familias como *Conti* [4] han obtenido recientemente capacidad para utilizar hasta 32 hilos de `CPU` que les permiten tener una afectación del sistema mucho más rápida o familias como *Maze* [20] incluyen nuevas técnicas de extorsión incluyendo la filtración de información sensible extraída previamente. A demás, de las variantes de ransomawre enfocadas en dispositivos IoT [19].
Este incremento de aparición de nuevas variantes hace imposible su catalogación, detección y prevención basadas en firmas clásicas o el uso de heurísticas.

Proyectos como *YARA* [33] surgen bajo la necesidad de desarrollar nuevas herramientas que permitan facilitar la investigación y clasificación de las nuevas variantes y familias que se descubren. Mediante el uso de patrones textuales, binarios, expresiones regulares, patro-

nes de comportamiento y uso de condicionales es posible generar reglas que identificarán la variante a la que pertenece dicha muestra.
YARA permite la integración en multitud de herramientas de análisis así como facilitar la distribución de dichas reglas con la comunidad investigadora.

No obstante, esta creación de reglas requiere haber analizado muestras conocidas y haber extraído sus principales características de forma automática o semiautomática.
Una posible forma de crear dichas reglas se basa en el análisis estático de la muestra. Estudios confirman que la creación de patrones y modelos de comprobación es altamente efectiva para la categorización de muestras conocidas o previamente observadas. No obstante, estas reglas son fácilmente evadidas mediantes simples transformaciones del código o mediante herramientas de re-escritura del binario [17].

En un intento de automatizar la obtención de las reglas de las muestras así como incluir características basadas en el comportamiento, surge el análisis dinámico. Esta forma de clasificación de las muestras implica la ejecución de la misma en un entorno controlado y monitorizar las conexiones, llamadas al núcleo del sistema así como archivos creados o modificados. De esta forma, es posible obtener una multitud de características del malware para su posterior clasificación, como se ha demostrado en [23] y [29].

La automatización del proceso de análisis junto la inclusión de las características obtenidas mediante el análisis estático y dinámico rápidamente plantea otro importante reto. La automatización del procesamiento de esas características y la clasificación de las muestras.
Con la cantidad de información disponible, tanto de muestras potencialmente maliciosas así como muestras legítimas, no es de extrañar la inclusión de técnicas de *machine learning* para la obtención de modelos no triviales, teniendo un alto porcentaje de verdaderos positivos con ratios de falsos positivos cercanos a cero. [10][2].

Herramientas *online* como *FalconSandbox*, *Any.Run* o *VMXRay* permiten observar el comportamiento el malware en un entorno virtualizado y obtener las propiedades del fichero, IOCs (*Indicator Of Compromise*) así como la historia de ejecución y grafos que modelan el comportamiento observado y su clasificación.

Debido al alto ratio de detección mediante métodos de aprendizaje automático y su aumento de la popularidad, surgen nuevos riesgos derivados. La utilización de ficheros especialmente diseñados para explotar los modelos predictivos más comunes y la ofuscación de la cadena de acciones o la realización de *Adversarial attacks* [26] se muestra como un factor de riesgo en la clasificación del malware de forma automatizada.

Capítulo 3

Enfoque

En este capítulo se va ha detallar el enfoque seguido durante la realización de este trabajo. En primer lugar, se mostrará la arquitectura propuesta del entorno de pruebas a lo largo de la Sección 3.1, detallando la partes involucradas así como su funcionamiento en sus distintos estados.
En la sección 3.2 se van a comentar brevemente las muestras de malware que van a ser analizadas.
A continuación, en la sección 3.3 se detallarán las especificaciones de las máquinas virtualizadas que servirán como entorno de *sandboxing*.
Después, en la Sección 3.4, se mostrarán las herramientas utilizadas para automatizar el análisis de las muestras así como de las condiciones del experimento.
Para finalizar, se comentarán las técnicas usadas para la agregación y el análisis de los resultados obtenidos.

3.1. Arquitectura del entorno

Para la realización de los análisis, se ha elegido como sistema operativo anfitrión `Ubuntu Linux 18.04 LTS` conjuntamente con un procesador con 8 núcleos a 3.1 Ghz, 16 Gb de memoria `RAM` y 500 Gb de disco duro.
Como entorno de virtualización se ha elegido *VirtualBox*, siendo este el recomendado por Cuckoo. De esta forma, se dispondrá de cinco máquinas invitado, cada una de ellas con 2Gb

de memoria RAM y *Windows 7* como sistema operativo.

De forma adicional, se ha instalado la dependencia opcional *Volatility* [9], un *framework* de extracción de artefactos forenses de volcados de memoria RAM con el objetivo de aumentar la cantidad de información disponible en el informe final de cada muestra.

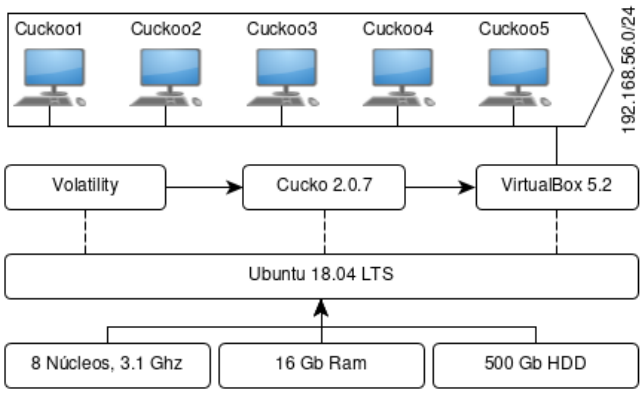


Figura 3.1: *Arquitectura propuesta para el análisis de las muestras.*

Para maximizar el aprovechamiento de las características hardware y limitar posibles afectaciones al rendimiento de los análisis, se han definido dos escenarios de ejecución.

En primer lugar, se define el **escenario de análisis**, en el cual únicamente convivirán las distintas máquinas virtualizadas y la instancia de Cuckoo en ejecución. La necesidad de este estado consiste en otorgar el máximo nivel posible de recursos a dichas máquinas y al posterior volcado de memoria, evitando posibles degradaciones del rendimiento a la hora de monitorizar las acciones generadas por las distintas muestras de malware.

Se puede observar un diagrama de los componentes que interaccionan en este estado en la Figura 3.1.

En segundo lugar, se define el **escenario de procesamiento de los resultados**, en el cual no existirán máquinas virtualizadas en ejecución. En contrapartida, se despliegan siete instancias de procesamiento de los resultados del análisis, volcado de memoria y registros de comportamiento con el objetivo de generar el informe final de cada una de las muestras. Estas tareas se caracterizan por ser extremadamente extensas en consumo de capacidad de procesamiento, operaciones de lectura y prolongadas en el tiempo.

Como la máquina anfitrión cuenta con un total de ocho núcleos, será posible ejecutar hasta siete instancias de procesamiento de forma concurrente; reservando un núcleo para el sistema operativo y el resto de procesos en ejecución.
Se puede observar un diagrama de los componentes que interaccionan en este estado en la Figura 3.2.

3.2. Muestras a analizar

Con el objetivo de obtener el mayor número de muestras para analizar y se han utilizado dos servicios web que permiten el acceso a muestras de malware bajo previo registro.

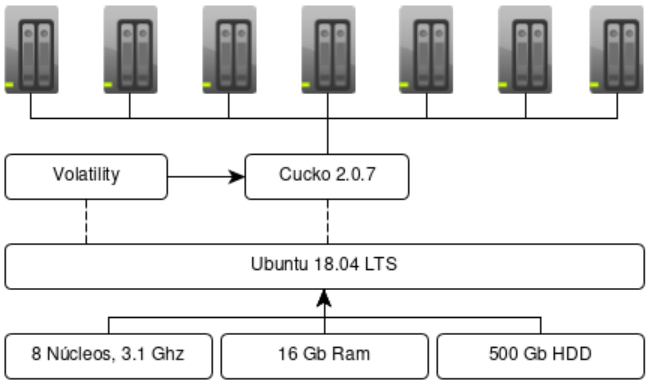


Figura 3.2: *Arquitectura propuesta para el procesamiento de los resultados.*

Del servicio web *VirusShare* [31] se ha obtenido el archivo comprimido *CryptoRansom* que contiene un total de 38.152 muestras incluyendo ejecutables, páginas web, scripts o archivos comprimidos.
Este paquete se generó a mediados del 2016 y por lo tanto se considera que no todas las muestras devolverán resultados relevantes.

Adicionalmente, gracias a la capacidad del servicio web *HybridAnalysis* [12] de generar consultas y descargar las muestras a través de una `API`, se han obtenido 700 muestras adicionales aproximadamente. Se puede consultar el conjunto que consultas realizadas en la Tabla 3.1.

Tipo	Veredicto	Etiqueta	Ratio detección	Clasificación
Ejecutable	Malicioso	`ransomware`	80-100 %	
Ejecutable	Malicioso			`Trojanransom.Generic`
Ejecutable	Malicioso	`wanacrypt0r`		
Ejecutable	Malicioso	`wannacry`		
Ejecutable	Malicioso	`wcry`		
Ejecutable	Malicioso			`CVE-2017-0147`
Ejecutable	Malicioso	`hiddentear`		
Ejecutable	Malicioso			`Trojan.Ransom.`

Tabla 3.1: *Búsquedas realizadas en HybridAnalysis*

3.3. Configuración del anfitrión

Como se ha comentado brevemente en la sección anterior, cada una de las máquinas virtualizadas dispondrá de hasta dos núcleos de procesador, 2 Gb de memoria RAM y *Windows 7* como sistema operativo.

Cada máquina dispondrá de una IP estática dentro de la subred `192.168.56.0/24` que permitirá la correcta comunicación con Cuckoo así como la conectividad con internet.
De forma adicional, dichas máquinas tienen deshabilitadas la opción de recibir actualizaciones del sistema operativo así como el control de cuentas de usuario (UAC) deshabilitado y el inicio de sesión sin contraseña automático de forma que las muestras analizadas no requieran realizar una escalada de privilegios para comprometer el sistema.

Para evitar la detección por parte del malware del sistema virtualizado, dichas máquinas contienen varios documentos depositados en las carpetas de usuario y se han instalado la suite de ofimática *Microsoft Office 2003*, *Adobe Acrobar Reader* y varios navegadores web. Para poblar el historial de navegación, caché y *cookies* de los distintos navegadores se ha utilizado la herramienta *Track This* [27].

Para finalizar, se ha instalado *Python 2.7* y se ha colocado el *script* del agente de Cuckoo

en las carpetas de autoarranque del sistema, de forma que se pueda retomar la comunicación con el entorno de monitorización incluso entre reinicios.

Una vez finalizada la configuración del sistema anfitrión, se ha generado un *snapshot* del estado actual en ejecución que permita recuperar la máquina una vez finalizado el análisis y no sea necesario esperar al arranque del sistema operativo en cada análisis.

3.4. Realización de los experimentos

Una vez terminada la configuración del entorno, de las máquinas invitado y obtenida una cantidad relevante de muestras es necesario definir las condiciones en las que se realizarán los experimentos.
En este aspecto, Cuckoo permite una gran cantidad de opciones entre las que destacan:

- **timeout:** Tiempo de análisis en segundos.
  Por defecto está fijado a 120 segundos pero se ha incrementado su valor a 300 para garantizar que las muestras de *ransomware* tienen tiempo suficiente de encriptar los archivos.

- **package:** Paquete de análisis a utilizar.
  Como se desconoce la extensión exacta de todas las muestras se ha dejado en automático (valor vacío).

- **machine**: Máquina que realizará el análisis.
  Debido a que en esta situación en particular todas las máquinas son idénticas, se ha dejado vacío para que se ejecute en la primera máquina disponible.

- **priority**: Prioridad con la que se realizará el análisis.
  Se ha definido la prioridad 1 (baja) para las nuevas muestras y la prioridad 2 (media) para las muestras analizadas anteriormente en las que se ha producido algún error durante su análisis.

- **unique**: Evita que las muestras ya analizadas vuelvan a ser depositadas para un nuevo análisis.

- **human**: Simula interacción de ratón y teclado.
  Se ha activado para evitar la detección del entorno virtualizado por la falta de interacción del usuario.

- **procmemdump**: Genera el volcado de memoria al finalizar el análisis.
  Se ha activado esta opción para la extracción de artefactos forenses con *Volatility*.

- **procmon**: Obtención de acciones realizadas mediante *procmon* [7].
  Se ha habilitado para incrementar la información relativa al comportamiento de la muestra.

Una vez definidas las opciones de configuración para todos los análisis de las muestras se puede proceder a la realización de los experimentos.

No obstante la necesidad de tener dos escenarios (análisis y procesamiento) conjuntamente con la creación de volcados de memoria implica nuevos retos a la hora de gestionar el entorno.
Por un lado, la creación del volcado de memoria para su posterior extracción de artefactos forenses implica la necesidad de un mayor espacio de almacenamiento. Nótese que cada análisis creará como mínimo un total de 2 Gb de archivos para procesar, debido a los 2 Gb de memoria RAM del que dispone cada máquina virtual.
Por otro lado, debido a las limitaciones del *Global Interpreter Lock* de Python, el procesamiento de los resultados puede verse severamente afectado en su rendimiento si las instancias comparten un mismo intérprete.
Además, en la versión actual de Cuckoo (2.0.7) no existe ninguna característica que permita alternar los escenarios en un mismo nodo.

Con todas esas limitaciones en consideración, para facilitar la gestión de los experimentos y automatizar la transición entre los distintos escenarios, se ha desarrollado la herramienta *Cuckoommander* [1].Esta herramienta desarrollada en Python, permite conectarse a una máquina remota a través de `SSH`, desplegar instancias de la `API` de Cuckoo , instancias de análisis o de procesamiento de resultados. De esta forma, es posible la monitorización remota del entorno automatizando el cambio de escenarios maximizando el rendimiento de los experimentos.

3.5. Agregación de los resultados obtenidos

Una vez que finaliza el análisis de de una muestra y se procesan los resultados con las evidencias de la ejecución, Cuckoo permite elegir el formato de la salida del informe, siendo posible `html`, `pdf` o `json`.
Adicionalmente, Cuckoo también guardará una copia del mismo en la instancia de *MongoDB* si se han instalado las dependencias para su interfaz web.

Conforme se empiezan a generar los primeros informes, surge el problema de cómo agregar los resultados de los mismos para la extracción de las características comunes entre ellos. Debido a la gran cantidad de contenido y variación de la estructura del mismo en función de las evidencias obtenidas, la agregación de los resultados en bases de datos relacionales supone un reto de pre-procesamiento de los datos y obligando a hacerse una vez finalizados todos los experimentos.
Por otro lado, la utilización de bases de datos no relacionales, como podría ser *MongoDB*, también supone un reto de discernir que datos pueden ser descartados ya que en muchas ocasiones, el tamaño del informe supera el máximo del documento permitido por el sistema gestor de base de datos.

Para mitigar los problemas de pre-procesamiento y agregación de los datos se ha utili-

zado como entorno de análisis *Jupyter Notebook* [13] que mediante las librerías de análisis de datos previamente disponibles en Python permite la agregación y análisis de los resultados obtenidos en los ficheros `json` mientras se siguen realizando expermientos de forma concurrente.

Capítulo 4

Resultados

En este capítulo, se presentan la evaluación del sistema propuesto. En la Sección 4.1 se detallarán aspectos relacionados sobre el análisis de las muestras y realización de los experimentos desde un punto de vista general. A continuación, en la Sección 4.2 se detallarán las características obtenidas mediante la realización del análisis estático. Por otro lado, en la Sección 4.3 se mostrarán los resultados que se han realizado mediante el análisis dinámico. Para finalizar el capítulo, en la Sección 4.4 se detallan las firmas más observadas y que tipo de *malware* están asociadas comúnmente.

4.1. Análisis general de las muestras

Antes de comenzar a describir los resultados obtenidos, es necesario realizar un estudio de los ficheros analizados, sus características generales, así como en análisis preliminar obtenido de ellos.

En la subsección 4.1.1 se detallará la tipología de las muestras obtenidas y sus implicaciones básicas durante la etapa de análisis. A continuación, en la subsección 4.1.2 se analizarán las diferencias obtenidas entre el tiempo asignado para la realización de cada experimento y la duración real. Para finalizar, en la subsección 4.1.3 se comenta brevemente las puntuaciones obtenidas.

4.1.1. Tipología de las muestras

Debido al origen heterogéneo de las muestras, así como agregados previa la adquisición final para su análisis, la mayoría han perdido su nombre y formato original, habiéndose sustituido por su valor *hash*, siendo imposible su clasificación previa al análisis.
A la hora de enviar una muestra a la instancia de `Cuckoo`, es posible determinar a que tipo de fichero pertenece la muestra para así mejorar en análisis posterior de ella. Debido a las condiciones de las muestras obtenidas y carecer del tipo de fichero a la que pertenecen, se ha optado por permitir que la instancia de `Cuckoo` determine automáticamente el tipo de fichero a la que pertenecen.

En la Tabla 4.1, se puede observar los 15 tipos de ficheros más comunes encontrados durante el análisis de las muestras. Se evidencia así la heterogeneidad de las muestras que

Tipo de fichero	**Ocurrencia**
PE32 exe (GUI) Intel 80386, for MS Windows	1378
PE32 exe (DLL) (GUI) Intel 80386, for MS Windows	583
PE32 exe (GUI) Intel 80386, for MS Windows, Nullsoft Installer self-extracting archive	169
PE32 exe (console) Intel 80386 (stripped to external PDB), for MS Windows	142
PE32 exe (GUI) Intel 80386, for MS Windows, UPX compressed	65
PE32 exe (GUI) Intel 80386 (stripped to external PDB), for MS Windows	47
PE32 exe (GUI) Intel 80386 Mono/.Net assembly, for MS Windows	38
PE32 exe (DLL) (GUI) Intel 80386, for MS Windows, UPX compressed	38
PE32 exe (DLL) (console) Intel 80386, for MS Windows	34
PE32 exe (console) Intel 80386, for MS Windows	25
Google Chrome extension, version 2	23
ASCII text	21
data	18
MS-DOS exe, MZ for MS-DOS	17
HTML document, ASCII text, with CRLF line terminators	16

Tabla 4.1: *Top 15 categorías de los ficheros analizados*

puede llegar a afectar negativamente a los resultados obtenidos.
Por ejemplo, los ficheros que pertenezcan al tipo "*HTML document*", "*ASCII text*" o "Google

Chrome extension" no podrán ser ejecutados directamente, necesitando cargar dependencias adicionales como editores de texto o navegadores disponibles en los entornos de *sandboxing*. Esta carga de dependencias, genera un gran volumen de información que deberá ser correctamente filtrada si se desea evaluar correctamente la muestra.

Además, tipos de ficheros como las librerías dinámicas (*DLL*), pueden generar análisis donde no se encuentre actividad maliciosa aunque dichas librerías sean mal intencionadas. La principal diferencia entre un fichero ejecutable y una librería dinámica es que estas últimas necesitan tener especificado el *entrypoint* o punto de entrada para su ejecución, dificultando enormemente su análisis.

4.1.2. Tiempo de análisis

Dentro de las opciones de configuración de `Cuckoo` para realizar el análisis de las muestras, un parámetro clave es el tiempo asignado a la realización de cada experimento. Si un análisis ha superado el tiempo máximo permitido, el sistema procederá a cerrar todos los procesos activos, obtención de la información generada así como proceder a la creación del volcado de la memoria para su posterior análisis.

El tiempo por el cual se permite la ejecución de una muestra es de vital importancia para los resultados obtenidos. Un tiempo de experimento demasiado corto puede provocar que no se obtenga toda la información de la muestra a analizar. Además, también es posible que no se obtenga información alguna debido a que es posible que se requiera periodo de tiempo inicial antes de la ejecución de las acciones por parte del *malware* siendo este un método de ofuscación y evasión habitual.
En contrapartida, un tiempo de análisis demasiado elevado puede afectar negativamente a la cantidad de muestras analizadas y no aportar información adicional de las ya analizadas.

A la hora de enviar una muestra para su análisis, se ha fijado un valor de *timeout* de 300 segundos de análisis. Este valor se ha fijado basándose en la información obtenida de la simulación manual de varias muestras de *ransomware*, habiendo permitido la encriptación total de la máquina en dicho tiempo.

Los resultados de la duración del análisis se pueden observar en la Tabla 4.2 y en la Figura 4.1.
Como se puede observar, únicamente el 25 % de los análisis ha terminado en un tiempo inferior a los 617 segundos, siendo casi el doble del tiempo fijado.
Estos resultados se deben a que el tiempo de generación del volcado de la memoria no está contabilizado en el *timeout* definido, por lo que la media de la duración en segundos de los experimentos se fija en aproximadamente 760 segundos. Este valor es más del doble del valor definido inicialmente por lo que afecta muy negativamente a la cantidad de experimentos que se pueden llegar a realizar y se deberá de tener en cuenta para futuros experimentos.

Adicionalmente, también se observan valores atípicos por debajo de los 50 segundos y por encima de los 1400 segundos que deberán ser evaluados en trabajos futuros.

	N. exp.	**Media**	**Std**	**Min**	**25 %**	**50 %**	**75 %**	**Max**
Duración (s)	2740	759.60	331.55	2.0	617.0	717.5	850.25	5320.0

Tabla 4.2: *Estudio de la duración de los experimentos*

4.1.3. Puntuación de las muestras obtenidas

Es evidente que la puntuación asignada a cada una de las muestras es algo subjetivo y depende enormemente de la heurística empleada. La correlación de diferentes heurísticas no es un tema trivial y no corresponde al ámbito de este estudio.

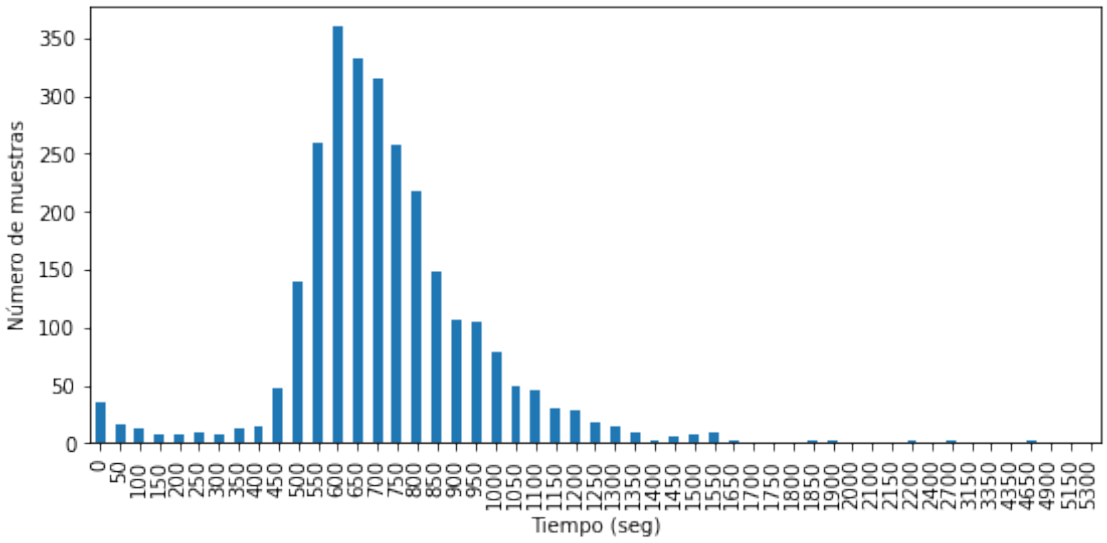


Figura 4.1: *Distribución en tiempo de análisis*

Por este motivo, únicamente se van a mostrar los dos sistemas de *scoring* empleados y las conclusiones obtenidas de cada uno de ellos.

Cuckoo scoring

La funcionalidad por parte de `Cuckoo` para generar una puntuación de la muestra de *malware* está en fase *alpha* por parte de sus desarrolladores, por lo que los resultados obtenidos pueden variar considerablemente en futuras versiones.

Además, dicha puntuación no está limitado por un valor máximo o limitada a base-10, siendo esta únicamente la suma de las puntuaciones parciales asignadas a cada acción maliciosa detectada por `Cuckoo`. De esta forma, una muestra que genera multitud de acciones maliciosas con puntuaciones parciales bajas podría llegar a tener una puntuación final superior a una muestra que realiza muy pocas acciones con puntuaciones parciales altas.
Además, al tratarse de una simple suma de acciones detectadas, es posible que la puntuación varíe entre distintos análisis de una misma muestra.

En la Tabla 4.3 se puede observar como la puntuación máxima asignada corresponde al valor 18,5, evidenciando que la funcionalidad está en *alpha* y que no corresponde a una escala en base-10.

Además, se evidencia que el 75 % de las muestras están por debajo de la puntuación 5,0. Esta conclusión se observa gráficamente en la Figura 4.2, donde la mayoría de resultados están entre los valores 3,0 y 5,5.

Aunque este sistema de *scoring* pueda arrojar valores atípicos puede ser de especial utilidad

	Media	**Std**	**Min**	**25 %**	**50 %**	**75 %**	**Max**
Puntuación	4.29	1.38	0.0	4.0	4.0	5.0	18.5

Tabla 4.3: *Análisis de las puntuaciones obtenidas con* `Cuckoo`

si se hace una correlación entre los análisis con una puntuación muy baja y las muestras. De esta forma, se pueden buscar acciones de ofuscación o evasión que puedan romper la cadena de eventos y que `Cuckoo` no sea capaz de detectar.

Dicha correlación y análisis está fuera del ámbito de este estudio y se propone como trabajo futuro.

4.1.4. VirusTotal scoring

El servicio web VirusTotal [32] ofrece la posibilidad de analizar una muestra en distintas soluciones de antivirus, tanto gratuitas como comerciales y ofrecer el agregado de las detecciones obtenidas.

Es necesario anotar que las soluciones de antivirus disponibles en la página (actualmente más de 70) pueden no corresponder exactamente a las soluciones comerciales y que su detección es orientativa. Además, no ofrece un sistema de *scoring* pero si el número total de motores que han realizado una detección positiva respecto al total de los analizados. Generando un sistema de *scoring* consistente y robusto basado en la experiencia y conocimiento adquiridos por las soluciones de antivirus actuales. Por estos motivos, el sistema de *scoring*

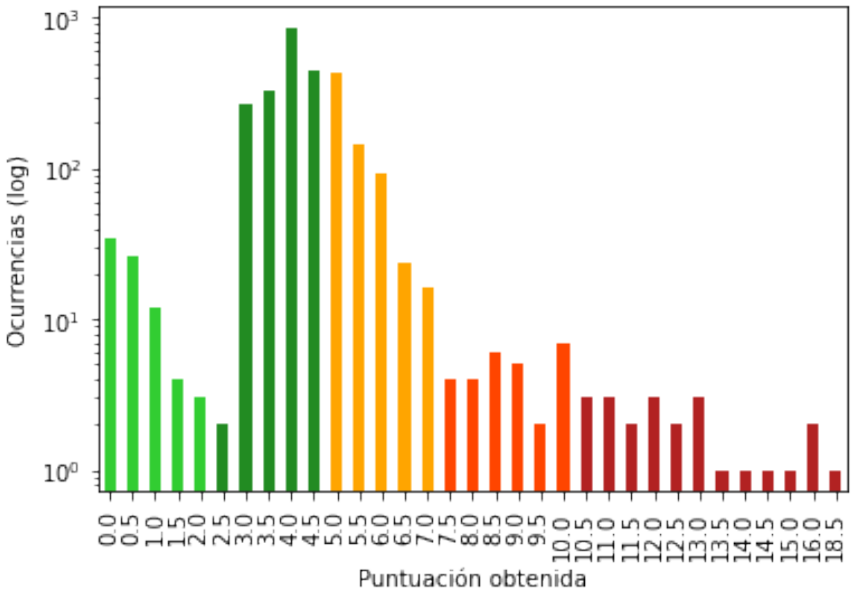


Figura 4.2: *Resultados por puntuación en* `Cuckoo`

propuesto corresponde a la Ecuación 4.1.

$$Score = \frac{Numero\ de\ detecciones\ positivas}{Numero\ de\ motores\ AV\ disponibles} * 10 \tag{4.1}$$

De esta forma, se puede obtener una vista general sobre las muestras analizadas. En la Tabla 4.4 se puede apreciar como la media de la puntuación asignada está en 8.26 puntos y que la mayoría de las muestras ha obtenido una puntuación superior a 8.

En la Figura 4.3 se puede observar la distribución de las muestras por su puntuación. Es necesario matizar que la escala de las ocurrencias está en base logarítmica.

	Muestras	**Media**	**Std**	**Min**	**25 %**	**50 %**	**75 %**	**Max**
Score	2740	8.26	1.18	0.0	8.03	8.48	8.97	9.71

Tabla 4.4: *Análisis de las puntuaciones obtenidas en VirusTotal*

4.2. Análisis estático

En esta sección se van a detallar los resultados obtenidos del análisis estático de las muestras.

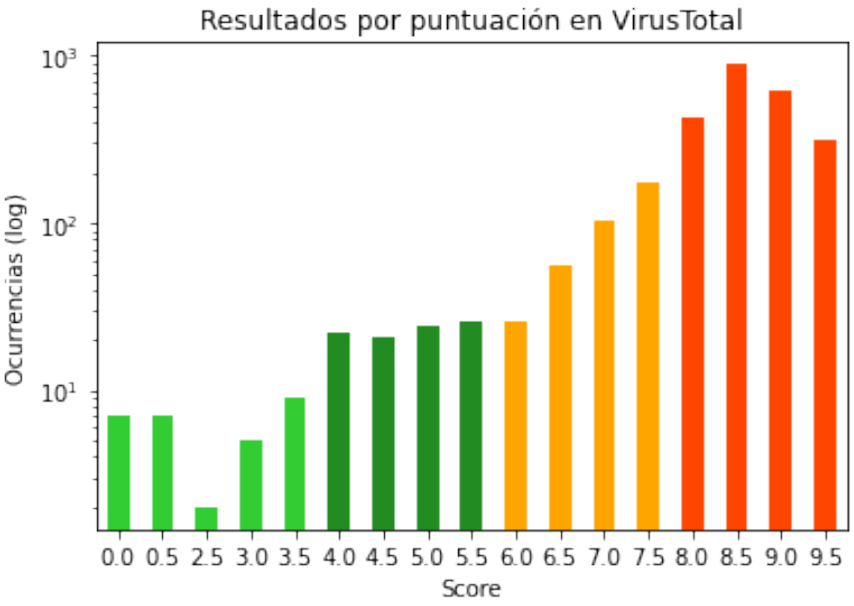


Figura 4.3: *Resultados por puntuación en VirusTotal*

El análisis estático de las muestras permite obtener una gran cantidad de información del fichero sin la necesidad de ejecutar el archivo potencialmente malicioso. Esta información puede permitir obtener una idea general de las capacidades del fichero.
Sin embargo, es habitual el uso de técnicas para enmascarar dichas capacidades así como incluir información genérica que pueda llevar a alterar el resultado del análisis.

4.2.1. Secciones del ejecutable

Las secciones del ejecutable es la primera información disponible sobre la organización interna del ejecutable, permitiendo esbozar ligeramente las funcionalidades del archivo.
En la Tabla 4.5 se puede observar el agregado de las secciones localizadas en las muestras analizadas. Mientras que las secciones más comunes (como `.text`, `.data` o `.rsrc`) son utilizadas por la mayoría de los ejecutables legítimos, el uso de secciones empaquetadas (por ejemplo `UPX0` o `UPX1`) pueden ser utilizadas en un intento de ofuscar las capacidades del fichero y por lo tanto considerarse como sospechosos. Adicionalmente, es posible detectar

Sección	.text	.data	.rsrc	.rdata	.reloc	.idata	.ndata	.tls	UPX0	UPX1
Num.	1927	1871	1853	1358	994	199	169	118	101	101
%	17.59	17.08	16.92	12.40	9.08	1.82	1.54	1.08	0.92	0.92

Tabla 4.5: *Secciones de los binarios más comunes*

secciones sospechosas en las secciones de las muestras a partir del valor de entropía calculado. Secciones con un valor de entropía muy alto son más probables de ser secciones empaquetadas o encriptadas en un intento de ofuscación y evasión. Por el contrario, secciones con un valor de entropía muy bajo pueden indicar la inclusión de secciones señuelo.

En la versión actual de `Cuckoo` no permite el cálculo del valor de la entropía de las muestras, por lo que no es posible realizar este estudio.

4.2.2. Importaciones de librerías

Una forma de obtener más información sobre las posibles capacidades de las que es capaz de realizar la muestra, es en función de que librerías y funciones importadas.

Uno de los métodos más simples de clasificación de las muestras de *malware* por sus capacidades es a partir del conteo de las librerías importadas. En la Tabla 4.6 se puede observar como la media de librerías importadas en las muestras analizadas está en 4 *dlls*, por lo que aquellas muestras que tengan un número más elevado de importaciones pueden resultar de especial interés.
Incluso si se eleva el número de importaciones a 10 (por encima del percentil 75 de las muestras analizadas), se pueden observar un gran número de muestras que cumplen dicha condición como se puede observar en la Figura 4.4. Estas muestras que realizan una elevada importación de librerías, podrían relacionarse con tareas de obtención de información del sistema infectado o funcionalidades avanzadas.

	Media	Std	Min	25 %	50 %	75 %	Max
Importaciones	4.45	3.63	1	2	3	7	29

Tabla 4.6: *Importaciones de dll*

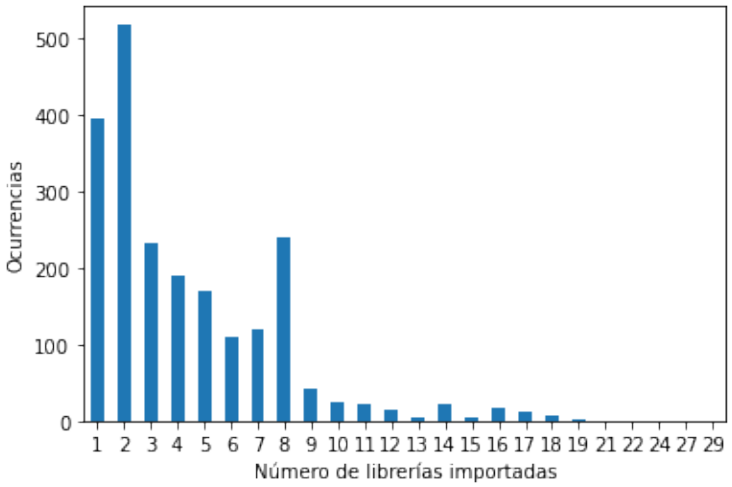


Figura 4.4: *Importaciones de dll*

También es posible obtener información relativa a las capacidades de las muestras en función de las funciones importadas. En la Tabla 4.7 se recogen las funciones más utilizadas en las muestras analizadas. No es extraño observar que todas pertenecen a la librería *kernel32.dll*, puesto que esta recoge multitud de funciones que permiten el acceso al sistema anfitrión y sus recursos.

El uso de funciones como `Sleep` o `GetTickCount` pueden evidenciar capacidades sobre el control del tiempo transcurrido y pausar temporalmente la ejecución. Por lo tanto, muestras donde se observen dichas funciones durante su análisis estático podrían estar latentes en el sistema anfitrión durante un periodo de tiempo determinado, por lo que sería conveniente aumentar el tiempo máximo de análisis para dichas muestras.
Del mismo modo, muestras con funciones como `CreateProcessA`, `TerminateProcess`, `GetProcAddress` o `LoadLibraryA` entre otras funciones, pueden evidenciar capacidades de creación de procesos e inyección en intentos de evitar ser detectadas las acciones maliciosas

rompiendo la cadena de eventos.

Otro método de evasión típico por parte de muestras de *malware* corresponde en lanzar excepciones sin capturar intentando determinar así si se encuentran en un entorno de *debugging*. En caso de que dichas excepciones sean capturadas por una herramienta externa, el *malware* puede cancelar su ejecución evitando así realizar acciones maliciosas que fuesen detectadas.

Dicha capacidad es posible si el programa importa funciones como `GetLastError` entre otras.

Función	Ocurrencias	Función	Ocurrencias
CloseHandle	969	WriteFile	954
CreateFileA	793	GetProcAddress	779
GetLastError	743	ExitProcess	716
GetModuleHandleA	702	GetCurrentProcess	688
GetTickCount	654	LoadLibraryA	646
Sleep	645	GetModuleFileNameA	622
MultiByteToWideChar	621	GetCommandLineA	528
CreateProcessA	523	GetCurrentThreadId	489
ReadFile	487	HeapAlloc	485
TerminateProcess	477	SetFilePointer	469

Tabla 4.7: *Funciones más utilizadas. Todas en kernel32.dll*

4.2.3. *Clustering* de muestras mediante `imphash`

Como se ha detallado en la subsección anterior, el análisis estático puede generar una gran cantidad de información sobre los ficheros y sus capacidades.

Una vez obtenida la información del análisis estático, el siguiente reto a considerar es la agrupación de diferentes muestras que puedan estar relacionadas, aún cuando el contenido del ejecutable portable (PE) pueda ser diferente.

Una de las técnicas más extendidas para realizar esta agrupación y correlación es la ge-

neración del `imphash` del fichero [22]. Esta técnica genera la suma de verificación *md5* de todos los símbolos importados por el ejecutable. De esta forma, dada una muestra de *malware* conocido, es posible recuperar otros ejecutables similares no categorizadas previamente aunque se hayan variado partes del fichero para evitar su detección.

La premisa sobre la cual trabaja `imphash` consiste en que para alterar el valor generado, debería ser necesaria la modificación del código, recompilación y/o modificar la fase de enlazamiento durante la compilación, siendo estas técnicas relativamente costosas y por ende, poco utilizadas. Aunque existen estudios que han intentado alterar los resultados generados [3], su efectividad está demostrada en numerosos entornos.

En la Tabla 4.8 se pueden observar los 10 valores `imphash` más repetidos de las muestras analizadas. La aplicación de esta técnica permite la agrupación de las 2740 muestras diferentes en únicamente 987 grupos, habiendo agrupado el 63.28 % de las muestras (Figura 4.5).

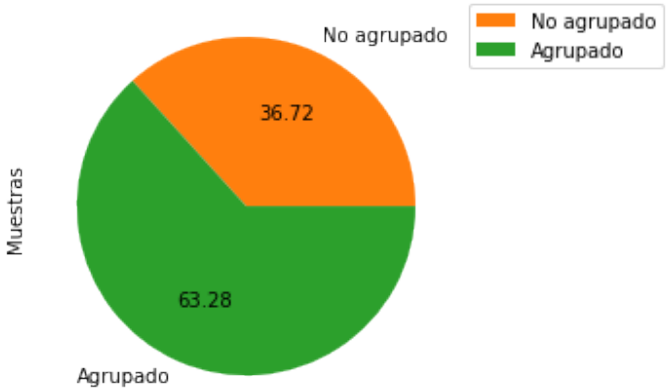


Figura 4.5: *Resultado de la agregación por valores `imphash`*

La utilización de esta técnica de agrupación puede ser de especial interés cuando se correlacionan las muestras agregadas con fuentes de información externa tal y como se detallará en la sección 4.5.1.

imphash	Total	imphash	Total
2e5708ae5fed0403e8117c645fb23e5b	299	7fa974366048f9c551ef45714595665e	155
4afc596e677872a5ef6d27ddd953788c	141	c15737f8c755f9044ac853318e8d6e15	53
f34d5f2d4577ed6d9ceec516c1f5a744	38	bfbf457d52153d2191e67bb6c9212334	31
c3c2b4ccfef5b46f908b991e2cd862b7	29	f86dec4a80961955a89e7ed62046cc0e	18
94400fe3e62cd2376124312fe435b8e4	17	87bed5a7cba00c7e1f4015f1bdae2183	16

Tabla 4.8: *Agregación de muestras que comparten el valor* `imphash`

4.2.4. Otras técnicas de agregación

Es normal que las muestras de *malware* recolectadas carezcan de los símbolos necesarios para realizar las tareas de *debug* en un claro intento de ofuscar las tareas de *reverse engineering*, generación de reglas Yara o generación de patrones de comportamiento que permitan una detección más eficaz.

Es por este motivo que habitualmente los ficheros maliciosos carezcan del fichero `pdb` (*Program DataBase*). No obstante, se ha observado en distintas familias de *ransomware* la sustitución de la ruta del fichero `pdb` por cadenas de texto en un intento de mandar un mensaje [18].

Estas rutas alteradas o mensajes pueden ser utilizadas como método de identificación de familias de *malware* o detectar archivos sospechosos.

En la Tabla 4.9 se encuentran algunos de las cadenas de texto encontradas en la sección del `pdb`. Aunque la clasificación manual de este tipo de cadenas puede ser tediosa, la aplicación de técnicas de *machine learning* podría permitir categorizar ficheros donde se haya apreciado una alteración.

4.3. Análisis dinámico

Aunque las técnicas descritas en el apartado anterior así como la información obtenida puede ayudar a la clasificación de una muestra como maliciosa o benigna, esta información está incompleta y no nos permite determinar con exactitud las acciones maliciosas realiza-

String PDB	Ocurrencias
C:\Users\Admin\Documents\VisualStudio2015\ProjectsFromRyuk\ ConsoleApplication54\x64\Release\ConsoleApplication54.pdb	9
monkapew.pdb	4
klipopga.pdb	3
j:_txupd_exe\release_txupd.pdb	3
nuktopwe.pdb	2
K:\oIelJTzLo\scxhgfeeUdgq\cekveje.pdb	2
wetertyyyyhjyu5ujy5_x.pdb	2
x:__\obj\Release\Biologyistooloffascism.docx.pdb	1
mspaint.pdb	1
irubmydickonxpncivsh1t.pdb	1
D:\fake.pdb	1
C:\zITS\Applications\RansomWare\Release\jackpot.pdb	1
C:\Wuhan\Lab\coronashit.pdb	1
C:\Users\Godbuntu\source\repos\wyvernlocker\Release\ wyvernlocker.pdb	1
c:\With\An\Configuration\Are\Truth.pdb	1
c:\Whereby\The\Correspond\Of.pdb	1

Tabla 4.9: *Cadenas pdb sospechosas*

das.

Es en este punto donde se introduce el término de análisis dinámico, donde el punto de vista varía hacia la muestra en ejecución, sus acciones realizadas y a los cambios realizados en el sistema [26].

En primer lugar se revisarán las acciones típicas por parte del *malware* en el registro del sistema y su importancia. A continuación se detallarán las cadenas de acciones y procesos involucrados durante las primeras fases de la infección en los análisis. Para finalizar, se analizará el tráfico de red en busca de indicadores adicionales que permitan detectar acciones maliciosas.

4.3.1. Registro del sistema

El registro de Windows es una base de datos jerárquica que permite el almacenamiento a bajo nivel de entradas tipo *clave-valor*, también conocidas como registros.
Estos registros pueden contener información, ajustes, opciones y otro tipo de valores asociados a programas, servicios, controladores, *kernel* o incluso cuentas de usuarios.

Debido a la facilidad de acceso al registro, a su información contenida así como realizar cambios en la configuración del sistema operativo, es habitual encontrar *malware* que abusa de esta funcionalidad tanto como para la fase de reconocimiento inicial como generación de persistencia incluso tras el reinicio del dispositivo infectado. Es por este motivo que la monitorización del registro y sus cambios son utilizados como técnicas para descubrir nuevas muestras de *malware* desconocidas hasta la fecha [15] o agrupar muestras en función del comportamiento realizado [24], quedando en evidencia su utilidad en los análisis forenses de los dispositivos [6].

Durante los momentos iniciales de la ejecución de un programa, es habitual el acceso y lectura a claves de registro en busca de parámetros de configuración o características de la configuración del dispositivo. Es por esto que será habitual que el número de claves leídas sea muy superior al de claves escritas.
Tras analizar los resultados obtenidos, esta hipótesis se confirma tal y como se puede observar en la Tabla 4.10, donde la cantidad de operaciones de lectura es 100 veces mayor que la de escritura.

Operación	**Media**	**Std**	**Min**	**25 %**	**50 %**	**75 %**	**Max**
Escritura	5.46	13.63	0	0.00	0.0	5.0	113
Abrir	101.73	242.51	0	0.00	6.0	58.0	2172
Leer	504.35	1148.71	0	1.25	19.5	156.0	8922

Tabla 4.10: *Total de acciones en el registro detectadas*

Además, la distribución de las lecturas de claves por muestra que el 75 % de las muestras realiza menos de 156 lecturas mientras que la media está en 504. Esto evidencia la existencia de algunas muestras que realizan un número de lecturas muy superior a la media, fijando el máximo de lecturas en 8922 claves.

Debido a la existencia de datos atípicos en la lectura de las claves y la cantidad de información que debería analizarse, no se va a proseguir con el estudio de las claves de registro leídas. Es posible observar dichos datos atípicos en la Figura 4.6.

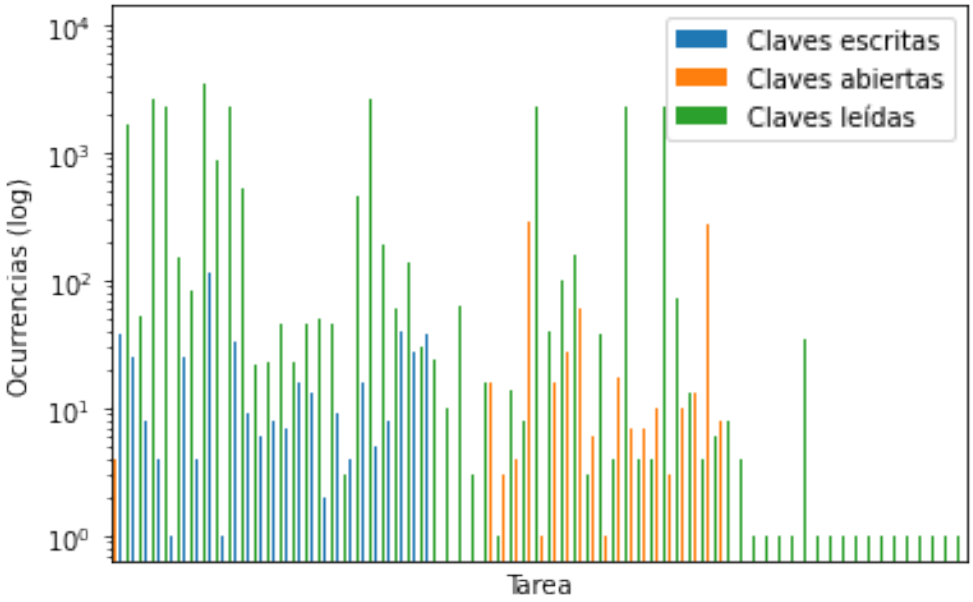


Figura 4.6: *Acciones en el registro observadas. Escala logarítmica en ocurrencias.*

En contrapartida, el número de claves escritas suele ser muy inferior en número y puede provocar un impacto directo en el rendimiento y la salud del sistema. Las claves que tras su modificación pueden generar un mayor impacto en el sistema son conocidas como *Registry Hives*.

Un *hive* es un grupo lógico de claves y sus valores en el registro que son cargados en la memoria del sistema operativo durante su arranque.

Estos *hives* pueden ser genéricos del sistema operativo o propios para cada usuario y

hacen referencia a los ajustes de las aplicaciones del usuario, escritorio, conexiones de red, entorno, impresoras, etc.

La modificación de estas claves es de especial interés por parte de actores maliciosos para poder realizar una o varias de las siguientes acciones entre otras:

1. **Obtención de persistencia:**
   La inclusión de registros en `CurrentVersion\Run` y `CurrentVersion\RunOnce` puede permitir la ejecución de archivos potencialmente peligrosos al arranque o inicio del sistema operativo.

2. **Modificar parámetros de seguridad de navegadores:**
   Los cambios en `\Microsoft\InternetExplorer\MAIN\FeatureControl` pueden permitir una posterior intrusión o exfiltración de datos al no seguir las políticas de seguridad recomendadas.

3. **Monitorización de las acciones en el dispositivo:**
   Registrar un programa malicioso en las claves `\Microsoft\Tracing` podría otorgar capacidades de *debugging* y monitorización no autorizados.

Un ejemplo de las alteraciones del registro catalogadas como potencialmente maliciosas detectadas durante el análisis de las muestras está disponible en la Tabla 4.11.

4.3.2. Árbol de procesos

Otro método de interacción con el sistema por parte de los ficheros potencialmente maliciosos es a través del uso de ejecutables incorporados en el sistema operativo. La utilización de dichos binarios reduce la cantidad de funcionalidades que deben ser desarrolladas para comprometer el dispositivo, son independientes de la arquitectura del sistema y son clasificados como benignos por la mayor parte de soluciones de antivirus.

Clave escrita	Ocurrencias
HKCU\Software\Microsoft\Windows\CurrentVersion\InternetSettings*	224
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts\[EXTENSION]	106
HKLM\SOFTWARE\Wow6432Node\Microsoft\Tracing\[MALICIOUSFILE]\[TRACINGMETHOD]	18
HKCU\Software\Microsoft\Windows\CurrentVersion\Run\[MALICIOUSFILE]	14
HKLM\SOFTWARE\Wow6432Node\Microsoft\InternetExplorer\MAIN\FeatureControl\[FEATURE]\[MALICIOUSFILE]	12
HKCU\Software\Microsoft\InternetExplorer\Main\NoProtectedModeBanner	10
HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\System\DisableTaskMgr	8
HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Run\[MALICIOUSFILE]	7
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\[MALICIOUSFILE]	7
HKLM\SOFTWARE\Wow6432Node\SNOW*	4

Tabla 4.11: *Cambios de registro maliciosos más detectados*

Una vez que el *malware* sea ejecutado en el sistema víctima, realizará una cadena de acciones realizadas tanto por el ejecutable como por otros ejecutables disponibles. Esta sucesión de acciones es también conocida como árbol de procesos. Este árbol permite categorizar las muestras en función de su comportamiento. Es obvio pensar que un *malware* de tipo *ransomware* no realizará las mismas acciones que un *RAT* (*Remote Access Tool*) o que un *infostealer*.

Cuando se dispone de suficientes árboles de procesos de las muestras maliciosas es posible trazar patrones de comportamiento que permitan inferir la taxonomía de las muestras e incluso, la pertenencia a una familia en concreto dentro de la misma clasificación.
Tanto en la Figura 4.7 como en la Tabla 4.12 se muestran los procesos pertenecientes al sistema que más se han detectado durante los análisis. Se han eliminado de la agregación el

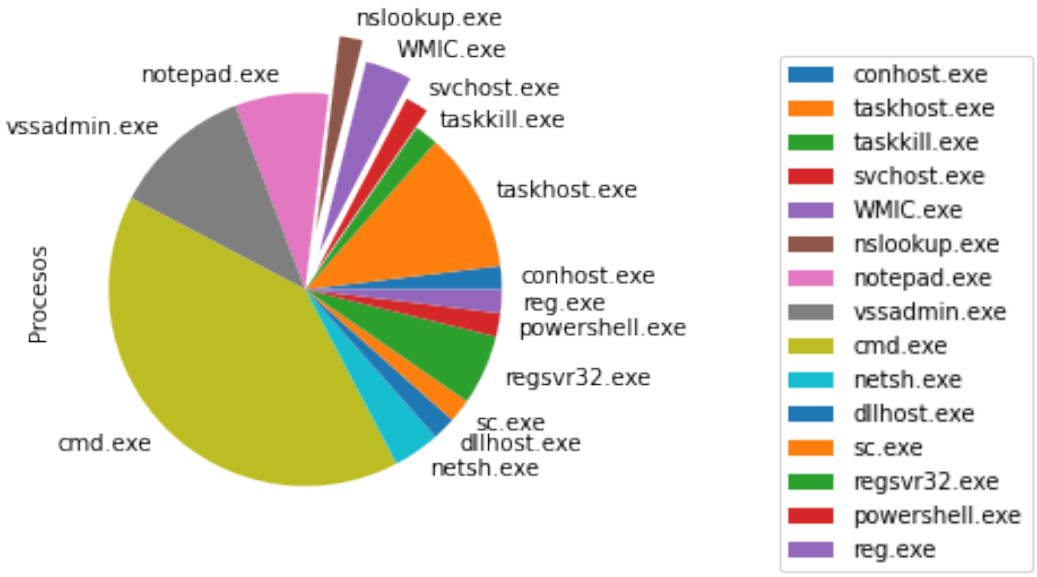


Figura 4.7: *Procesos ejecutados por las muestras*

fichero ejecutable inicial que crea el árbol de procesos. De esta forma, se puede observar de modo general que ejecutables son los más habituales.

Cada ejecutable será utilizado para una o varias tareas en específico pero pueden ser agrupados en las siguientes categorías:

- ***Shell***: Ejecutables como `cmd.exe` o `powershell.exe` permiten generar una instancia de una terminal para la ejecución de otros comandos, realizar tareas de *scripting* y evitar detecciones basadas en modelos de comportamientos. *Windows PowerShell* está disponible desde Windows XP SP2 aunque existen cambios introducidos en sus tres versiones que pueden provocar errores de sintaxis.

- **Registro**: Comandos como `reg.exe` o `regsvr32.exe` permiten la interacción con el registro sin la necesidad de utilizar una interfaz gráfica. Como se ha comentado en la subsección 4.3.1, pueden utilizarse en tareas de reconocimiento o generación de persistencias.

- **Procesos**: Binarios como `taskkill.exe` o `taskhost.exe` para interaccionar con otros

procesos del sistema o crear nuevos procesos.

- **Servicios**: Los programas `svchost.exe` y `sc.exe` permiten la ejecución de servicios, así como detener, iniciar, crear o borrar servicios ya existentes. La ejecución de binarios como servicios suele estar asociada a un nivel mayor de permisos administrativos en el sistema. También es posible su uso para detener servicios críticos para la seguridad del sistema como podrían ser soluciones de antivirus.

- **Inyección**: `conhost.exe` y `dllhost.exe` permiten la carga de procesos y librerías dinámicamente, lo que podría llegar a provocar una ejecución arbitraria de código en procesos del sistema.

- **Red**: Esta categoría agrupa a los ejecutables están directamente relacionados con operaciones de red. Por ejemplo, `nslookup.exe` podría permitir a un atacante la resolución de un nombre de dominio contra un servidor `DNS` en concreto y obtener así la dirección `IP` de un servidor de control. Por el contrario, `netsh.exe` permite la interacción sin necesidad de interfaz gráfica con el `firewall` del sistema pudiendo llegar a admitir conexiones entrantes o salientes bloqueadas anteriormente.

- **Integridad del sistema**: Aunque el ejecutable `wmic.exe` permite acceso a la *Windows Management Instrumentation* y puede ser utilizado para distintas acciones, su uso por parte del *malware* está asociado a las familias de *ransomware* al poder modificar las políticas de los *Shadow files*. Estos ficheros recogen los cambios incrementales realizados en los ficheros del sistema, permitiendo su restauración posterior. Aunque no sustituye a una solución de *backup*, podría llegar a mitigar la afectación del *ransomware* y recuperar parcial o totalmente los archivos. El ejecutable `vssadmin.exe` permite interaccionar con estas copias incrementales así como su borrado.

	conhost.exe	taskhost.exe	taskkill.exe	svchost.exe	WMIC.exe
%	1.92	11.54	1.92	1.92	3.85
	nslookup.exe	**notepad.exe**	**vssadmin.exe**	**cmd.exe**	**netsh.exe**
%	1.92	7.69	11.54	40.38	3.85
	dllhost.exe	**sc.exe**	**regsvr32.exe**	**powershell.exe**	**reg.exe**
%	1.92	1.92	5.77	1.92	1.92

Tabla 4.12: *Procesos ejecutados por las muestras*

4.3.3. Tráfico de red

Otro aspecto muy importante a monitorizar durante el análisis de *malware* es las conexiones de red que genera durante su ejecución.
La mayoría de muestras requerirán generar una o más conexiones para realizar tareas de reconocimiento, descargar dependencias, recibir comandos desde un servidor de control o enviar la clave privada en el caso de las infecciones por *ransomware*.

En la Figura 4.8 se puede observar de manera general el tráfico de red capturado agregado por el protocolo al que pertenecen las tramas. Por claridad, se han creado las categorías `tls`, `http` y `http_ex` que, aunque la comunicación se realice sobre el protocolo `tcp`, aportan mayor precisión e información al análisis.

4.3.4. Análisis del tráfico de red por protocolo

En el apartado anterior se ha podido observar gráficamente que existen cuatro protocolos que abarcan la totalidad de las comunicaciones observadas. Es por ello que es necesario realizar un análisis más en profundidad de las comunicaciones realizadas en cada protocolo, los servicios asociados a ellas y obtener una idea general de la función que realizan.

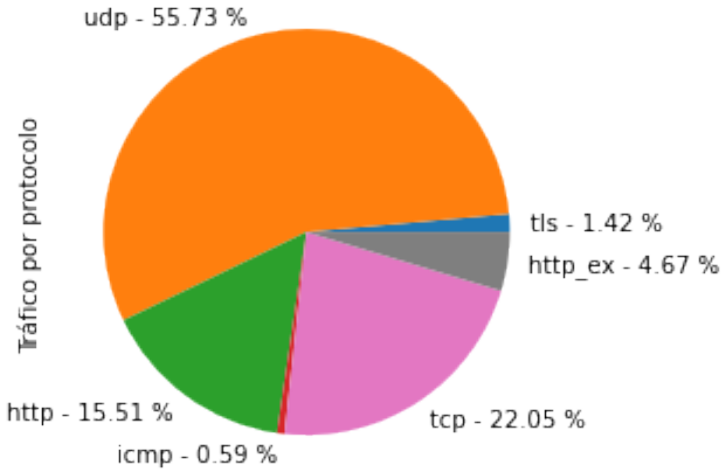


Figura 4.8: *Representación del tráfico capturado*

ICMP

El protocolo `ICMP` o *Internet Control Message Protocol* realiza tareas de soporte en las comunicaciones en la capa de internet. Estos mensajes no suelen usarse para la transferencia de datos arbitrarios, estando orientados en el envío de mensajes sobre el estado de la conexión.
Son usados ampliamente por dispositivos de red o programas para el diagnóstico de problemas de conexión, obtención de información de la arquitectura de red o el descubrimiento de dispositivos conectados en la misma subred.

Aunque en el estudio anterior, las comunicaciones mediante `ICMP` sólo representan el 0.59 % del total, no es extraño su uso por parte del *malware* durante las fases de movimiento lateral permitiendo descubrir otros dispositivos adyacentes en la red para posteriormente intentar comprometer dichos dispositivos.

UDP

El protocolo `UDP` o *User Datagram Protocol* es uno de los componentes más importantes de la *Internet Protocol Suite*. Este protocolo permite el envío de datagramas mediante el uso de puertos garantizando la integridad de los datos mediante sumas de verificación. Por contrapartida, dicho protocolo no dispone de mecanismos para garantizar la comunicación ni diálogos de *handshaking*, por lo que no está orientado a comunicaciones que requieran el uso de sesiones.

Uno de los requisitos para iniciar una comunicación mediante el protocolo es conocer a que puerto se va a enviar el datagrama a la máquina destino. Es por ello que multitud de servicios siempre utilizan el mismo número de puerto para iniciar la comunicación, permitiendo inferir que servicio está detrás de dicha comunicación y por ende, su utilidad.
Relacionando el puerto destino especificado con los servicios conocidos [16] que usan dicho puerto, se han podido identificar exitosamente 7 servicios diferentes reflejados en la Tabla 4.13.

Aunque la mayor parte de las tramas interceptadas no han podido ser relacionadas, se puede observar una gran cantidad de conexiones `DNS` y `NTP`. Los servicios propietarios de Microsoft como *Link-Local MulticastName Resolution*, *Web Services Discover*, *SSDP* o *NetBIOS* evidencian tareas de reconocimiento de dispositivos en la subred en intentos de movimientos laterales.

TCP

El protocolo `TCP` o *Transmission Control Protocol* es otro de los protocolos base utilizados para realizar comunicaciones por internet. A diferencia de `UDP`, este protocolo permite la implementación de sesiones así como control de errores de comunicación. Es por este motivo

Servicio	P.dst	Tramas
Desconocido		56572
DNS	53	21411
Link-Local Multicast Name Resolution	5355	18249
Web Services Discovery	3702	10425
NetBIOS	137	6810
SSDP Discover Service	1900	5279
NTP	123	2099
NetBIOS Datagram Service	138	835

Tabla 4.13: *Tráfico* `UDP`

Servicio	P.dst	Tramas
Microsoft Network Discovery	5357	19419
SSDP Discover Service	2869	18288
HTTP	80	6656
HTTPS	443	3106
Desconocido		689

Tabla 4.14: *Tráfico* `TCP`

su uso está más extendido y se considera más fiable.

Utilizando la misma aproximación que en el apartado anterior, se han podido asociar la mayoría de las comunicaciones a un servicio en concreto. Los resultados están recogidos en la Tabla 4.14.

Se vuelve a evidenciar el uso de protocolos propietarios de Microsoft que pueden permitir el descubrimiento de dispositivos en la red.

HTTP

El protocolo `HTTP` o *HyperText Transfer Protocol* permite el acceso a recursos remotos y compartidos a través de la *Web*. Este protocolo está definido en la capa de aplicación y usa internamente el protocolo `TCP` para realizar el transporte de la información.

Entre las funcionalidades aportadas por este protocolo se encuentran los métodos, permitiendo definir que acción se tomará sobre el recurso externo. Otra funcionalidad corresponde a la definición del *user-agent* que permite la identificación del actor que realiza dichas comunicaciones.

En la Tabla 4.15 se han agregado las comunicaciones más observadas en función del *user-agent* empleado, el puerto de destino y el método utilizado.
Consistente con los resultados obtenidos en el apartado anterior, la mayoría de conexiones corresponden a los protocolos asociados al descubrimiento de dispositivos.
Por otro lado, las comunicaciones utilizando el *user-agent Microsoft BITS* evidencian la obtención de recursos externos sin el uso de los navegadores web instalados. Este protocolo permite la transferencia de recursos en segundo plano de forma totalmente invisible al usuario, permitiendo obtener dependencias del fichero malicioso en ejecución o incluso la descarga de nuevas variantes de *malware*.

Para finalizar, los *user-agents* asociados a los navegadores web previamente instalados en las máquinas anfitrión corresponden a los análisis de los ficheros categorizados como documentos `HTML` en la Tabla 4.1.

user-agent	**Puerto**	**Método**	**Ocurrencia**
WSDAPI	5357	POST	19429
Microsoft-Windows/6.1 UPnP/1.0 Windows-Media-Player-DMS/12.0.7601.17514 DLNADOC/1.50	2869	GET	9614
Windows-Media-Player-DMS/12.0.7601.17514	2869	GET	8877
Microsoft-CryptoAPI/6.1	80	GET	6914
Microsoft BITS/7.5	80	GET	1166
		HEAD	364
FF_INTEGRATED	80	GET	120
Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko	80	GET	57
Google Chrome/1.3.35.452;winhttp;cup-ecdsa	80	POST	49
OUpdatesWebReporter	80	HEAD	35

Tabla 4.15: *TOP 10 conexiones HTTP*

4.3.5. Tráfico externo

Aunque la mayoría del tráfico analizado corresponde a conexiones dentro del área local, las conexiones externas son de especial interés ya que pueden permitir la detección de cone-

xiones a servidores de control remoto por parte del *malware*.

Para evitar la detección de este tipo de conexiones por *firewalls* o *IDS*, los actores maliciosos utilizan los protocolos de comunicación más comunes en un intento de asemejarse a las comunicaciones lícitas. Aunque estudios anteriores [14] han tenido éxito en la detección de conexiones a servidores de control remotos mediante `HTTP`, la falta de un sistema automatizado de análisis de los paquetes de comunicación generados no ha permitido la correcta identificación de ellos.

La mayoría de servicios web requieren de una disponibilidad muy alta para su correcto funcionamiento y calidad del servicio. Es por esto que aquellas comunicaciones hacia servidores externos que no hayan obtenido respuesta pueden revelar servidores de control no operativos actualmente.
La agregación y estudio de dichas conexiones no ha arrojado información relevante y está disponible para su consulta en los Anexos A.1 y A.2.

4.4. Firmas

La información obtenida en las secciones anteriores está relacionada a características, acciones o conexiones concretas que puedan permitir la identificación de acciones potencialmente maliciosas por parte de las muestras analizadas. Aunque la importancia de obtener esta información es crucial, también deja en constancia la dificultad de realizar su interrelación para observar comportamientos maliciosos desde un punto de vista a alto nivel.

Es imposible determinar si una muestra es legítima o por el contrario corresponde a un *malware* por la simple suma de sus acciones. Es por ello que aparece el término de *patrón de comportamiento*, siendo este la definición de una sucesión de eventos o acciones de forma ordenada que permita la relación de dichas acciones a un comportamiento malicioso cono-

cido.

Basado en este concepto, `Cuckoo` dispone de un conjunto de firmas desarrolladas por la comunidad [8] que permiten la identificación automática de los comportamientos potencialmente maliciosos. Aunque estas firmas abarcan multitud de lo casos de uso, es posible desarrollar casos de uso personalizados para acciones concretas o nuevas variantes de *malware*.

Durante la realización de los análisis se han utilizado las últimas firmas de la comunidad disponibles donde se ha podido observar que de media se detectan 211 firmas por muestra (Tabla 4.16). No obstante observando la distribución de las firmas detectadas se observan ciertos datos atípicos, habiendo generado un análisis 7473 firmas. Tal y como se observa en la Figura 4.9, la mayoría de los análisis han generado cerca de 100 firmas.

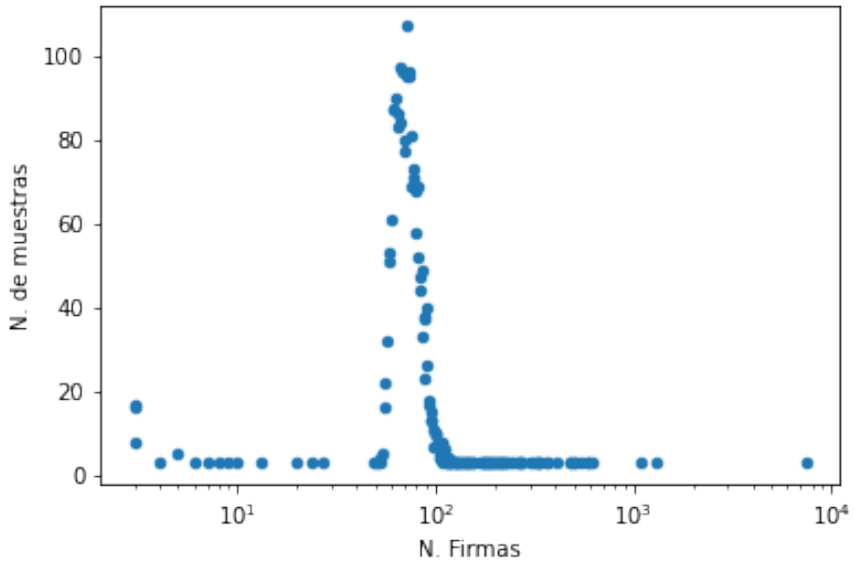


Figura 4.9: *Distribución de las firmas detectadas*

	Media	Std	Min	25 %	50 %	75 %	Max
N. Firmas	211.58	623.93	3	72	109	193	7473

Tabla 4.16: *Firmas*

La utilización de estas firmas no únicamente permite la detección de comportamientos maliciosos, sino que es posible la clasificación del *malware*. Por ejemplo, se han marcado en negrita aquellas firmas de la Tabla 4.17 que permiten la clasificación de las muestras como *ransomware*.

Estas firmas han detectado acciones unitarias como el borrado, movimiento o renombrado de un fichero, que al repetirse de una cantidad y formas definidas, permiten detectar el patrón de comportamiento propio de las infecciones por *ransomware*.

Comportamiento	**Num.**
PEB modified to hide loaded modules. Dll very likely not loaded by LoadLibrary	166785
Performs some HTTP requests	7615
Deletes a large number of files from the system indicative of ransomware, wiper malware or system destruction	4129
Foreign language identified in PE resource	3537
Repeatedly searches for a not-found process, you may want to run a web browser during analysis	2968
One or more thread handles in other processes	2661
Malfind detects one or more injected processes	2660
Stopped Application Layer Gateway service	2658
Stopped Firewall service	2657
The binary likely contains encrypted or compressed data indicative of a packer	2424
Steals private information from local Internet browsers	2213
The executable contains unknown PE section names indicative of a packer (could be a false positive)	2072
Communicates with host for which no DNS query was performed	1988
Allocates read-write-execute memory (usually to unpack itself)	1832
Searches running processes potentially to identify processes for sandbox evasion, code injection or memory dumping	1582
The file contains an unknown PE resource name possibly indicative of a packer	797
Generates some ICMP traffic	763
Installs itself for autorun at Windows startup	690
Performs file moves indicative of a ransomware file encryption process	594
Appends a new file extension or content to files indicative of a ransomware file encryption process	594

Tabla 4.17: *Firmas más detectadas durante los análisis*

4.5. Validación externa

En esta sección se van a recoger algunas de las ideas, conceptos o razonamientos realizados en los apartados anteriores validando las conclusiones a través de herramientas o servicios externos.

4.5.1. imphash

En la Sección 4.2.3 se había conseguido agrupar conjuntos de muestras que compartían el mismo valor `imphash` bajo la hipótesis de que la firma generada con los símbolos importados permite agrupar ficheros con unas capacidades similares.

Para comprobar la efectividad de esta técnica, se ha calculado la media ponderada del *scoring* generado con la Ecuación 4.1 y el servicio web *VirusTotal*. De esta forma, los ficheros maliciosos similares deberían tener un *score* similar. La Figura 4.10 permite observar la

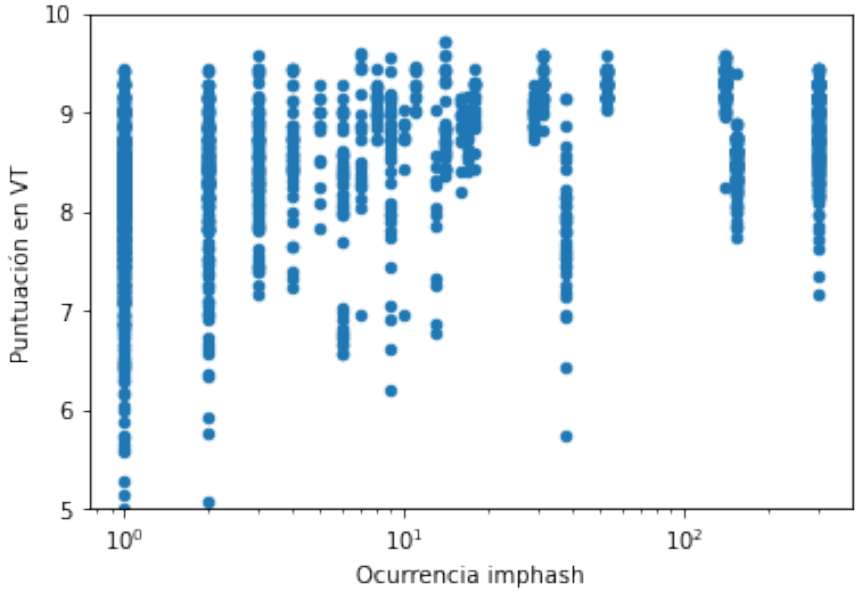


Figura 4.10: *Puntuación promedio en VirusTotal agrupado por* `imphash`

puntuación promedio como los ficheros con el mismo valor `imphash`. Aquellas muestras más

próximas a la esquina superior derecha serán mutaciones de un mismo *malware* conocido, pudiendo centrar el análisis en dichas muestras en trabajos futuros.

4.5.2. Ficheros empaquetados

En la Sección 4.2.1, durante el análisis de las secciones de los ficheros ejecutables, se detectaron ficheros con secciones empaquetadas.

Aunque la utilización de los *empaquetadores* también puede ser utilizado por programas benignos puesto que reduce el tamaño final del ejecutable, estas técnicas son asociadas con intentos de ofuscar las capacidades de los ficheros maliciosos.

Mediante la utilización de las firmas de `Cuckoo`, es posible observar que ficheros utilizan

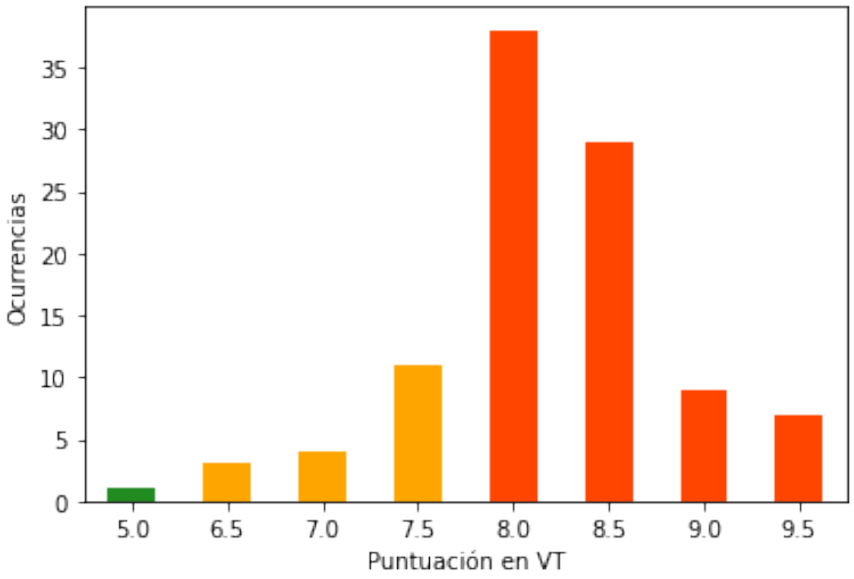


Figura 4.11: *Firma `UPX`, distribución en VirusTotal*

algún método de empaquetado y consultar sus detecciones en VirusTotal. En la Figura 4.11 se puede comprobar como los ficheros empaquetados tienden a tener un ratio de detección mayor por los sistemas de antivirus.

4.5.3. Carga dinámica de librerías

Durante la revisión de los resultados obtenidos en la fase del análisis dinámico se han encontrado ficheros que realizaban cargas dinámicas de librerías en un intento de evasión durante el análisis estático de la muestra.
En la Sección 4.3.2 se detallaban las herramientas del sistema que podrían utilizarse para realizar esta carga de librerías.

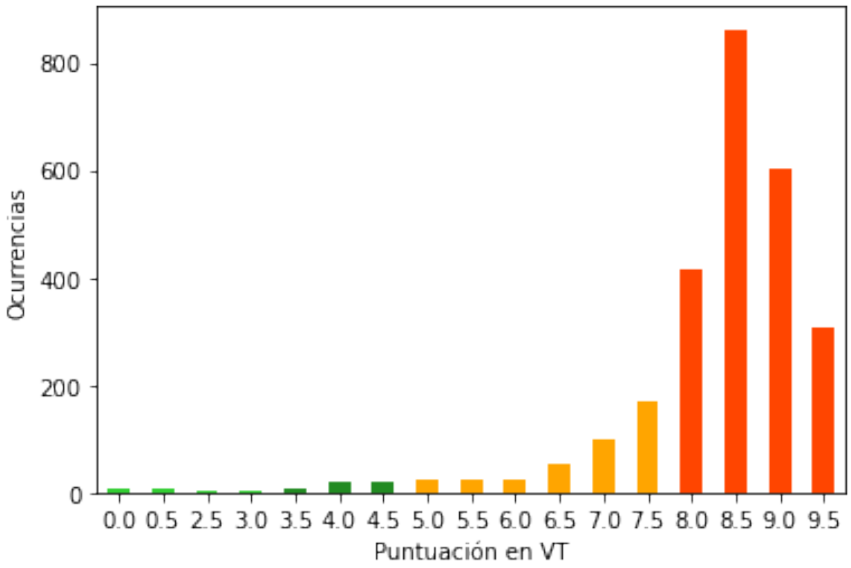


Figura 4.12: *Firma PEB Modified, distribución en VirusTotal*

Utilizando la firma de modificación del *PEB*, podemos localizar aquellos procesos que realizan la carga de módulos sin utilizar la función `LoadLibrary`. En la Figura 4.12 se puede comprobar que aunque existen muestras con puntuaciones muy bajas, la mayoría de las muestras que utilizan esta técnica han recibido puntuaciones por encima del 8.

4.5.4. Peticiones a *dead hosts*

Durante el análisis del tráfico web externo en la Sección 4.3.5, se evidenció la dificultad de detectar si los servidores externos que no generaban respuesta a las peticiones de las muestras habían realizado tareas de *Command & Control* en el pasado.

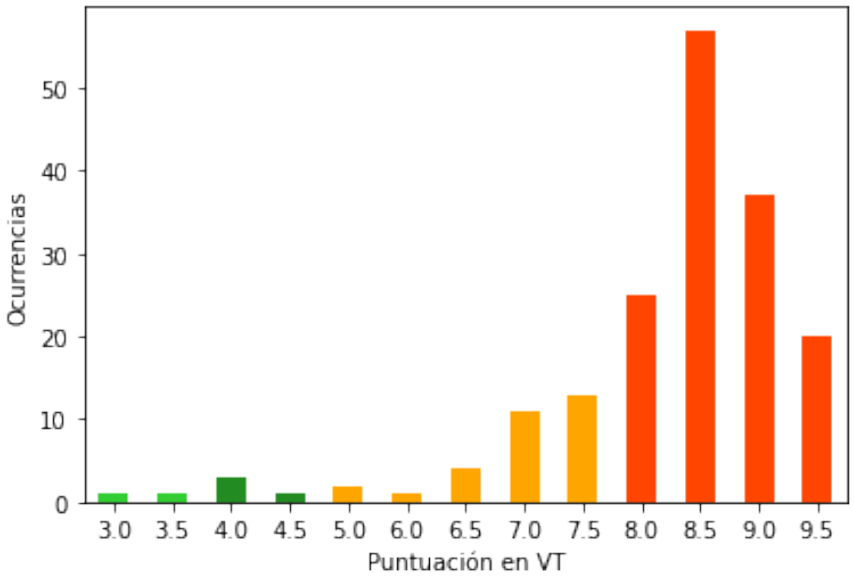


Figura 4.13: *Firma dead hosts, distribución en VirusTotal*

Debido a que posiblemente las soluciones de antivirus han podido realizar el análisis de las muestras mientras que el servidor de control seguía operativo y analizar la comunicación es posible utilizar las soluciones de antivirus como referencia. En la Figura 4.13 se puede comprobar que existen ejecutables con puntuaciones muy bajas, la mayoría de las muestras obtienen puntuaciones altas.

Aunque este comportamiento pueda ser realizado por ficheros maliciosos y benignos por igual, es seguro afirmar que se trata de un comportamiento sospechoso que podría relacionarse con *malware* antiguo.

4.5.5. Clasificación de las muestras

Aún con toda la información obtenida durante las diferentes partes del análisis, en ocasiones la clasificación de las muestras a un tipo de familia en concreto no es una tarea trivial. La enorme cantidad de técnicas utilizadas por los actores maliciosos, el constante número

de muestras así como la falta de un estándar claro provoca que existan discrepancias en el etiquetado de las muestras por parte de la comunidad investigadora.

Herramientas como `AVClass` [25, 11] permiten un etiquetado automático a través de los resultados obtenidos por distintos servicios.
Utilizando los resultados de los análisis en las diferentes soluciones de antivirus mediante VirusTotal, se han agregado todas las etiquetas asignadas a las diferentes muestras.

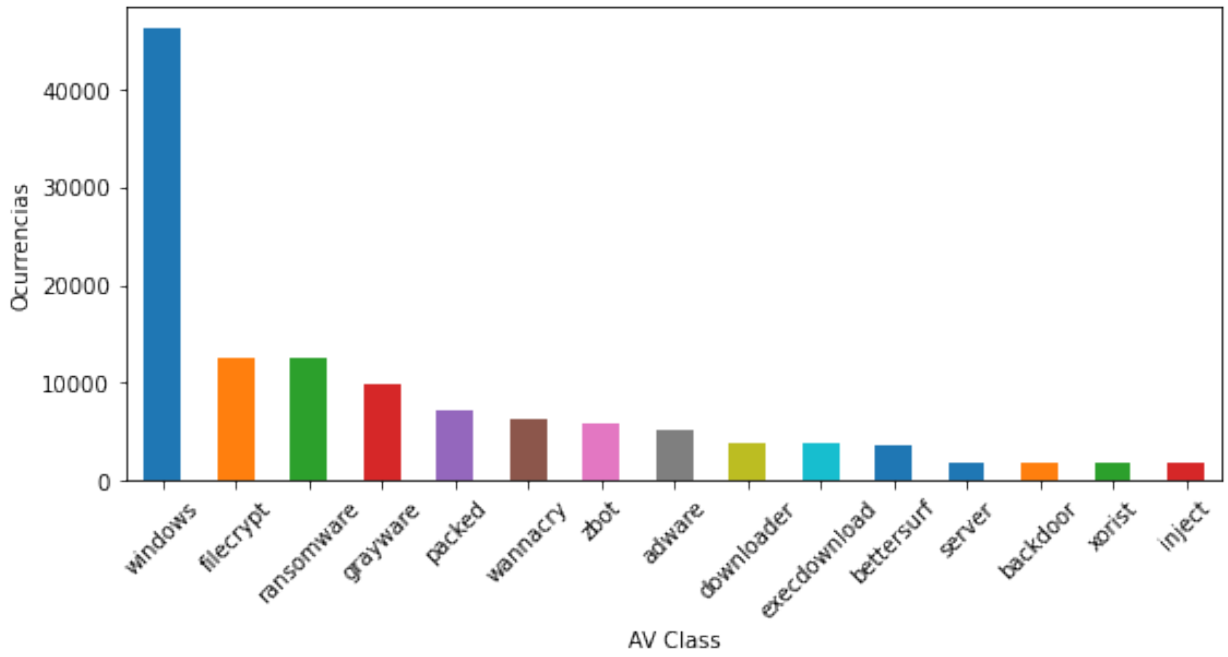


Figura 4.14: *Distribución de AVClass de las muestras*

La Figura 4.14 evidencia las diferencias del etiquetado entre los distintos antivirus. Etiquetas como *filecrypt* o *wannacry* son utilizadas para categorizar las muestras de *ransomware* de la que ya existe una etiqueta más genérica previa.
También queda constancia de etiquetas que, aunque puedan aportar información sobre la muestra, son demasiado genéricas como podría ser el caso de las etiquetas *windows* o *inject*.

Capítulo 5

Conclusiones y Trabajo Futuro

En este capítulo se recogen aquellas conclusiones obtenidas durante todo el desarrollo del estudio y fases del análisis realizadas. Adicionalmente, se recogen todas aquellos aspectos en los que se ha detectado que podrían otorgar conclusiones interesantes pero que no han podido desarrollarse en profundidad en este trabajo.

5.1. Conclusiones

En este trabajo, se ha propuesto una arquitectura del entorno de *sandboxing* `Cuckoo` con dos escenarios diferencias. La primer escenario trata de maximizar el número de instancias de análisis concurrentes. Por otro lado, el segundo escenario está centrado de aprovechar todos los recursos *hardware* para el análisis de los volcados de memoria y la obtención de artefactos forenses.

Posteriormente se ha obtenido un conjunto de muestras iniciales a partir de la agregación de distintas fuentes y repositorios públicos de *malware* con el objetivo de establecer una muestra lo más heterogénea con familias de *ransomware* conocidas.

Tras el análisis del máximo número de muestras posibles del conjunto inicial, se han agregado los informes generados por `Cuckoo` para poder consultar toda la información generada de una forma sencilla y consistente.

Una vez obtenidos los primeros resultados, se ha comprobado como el sistema de *scoring* ofrecido por `Cuckoo` no dispone de la robustez y consistencia necesaria para poder determi-

nar si las muestras analizadas son *malware*. Por este motivo se ha propuesto un sistema de puntuación simple basado en diferentes motores de antivirus y el servicio de VirusTotal.

En la Sección 4.2 se han introducido el concepto de análisis estático y que información es posible extraer de las muestras antes de su ejecución. Esto permite obtener una idea general de las capacidades del fichero en función de las librerías importadas y obtener indicios sobre alteraciones de la cabecera del fichero que son usadas habitualmente como técnicas de ofuscación.
Además se han analizado dos técnicas para realizar el *clustering* de las muestras de forma rápida a partir de muestras maliciosas ya conocidas.

En la Sección 4.3 se ha procedido a revisar la información obtenida mediante la ejecución de las muestras. Se ha remarcado la importancia del registro del sistema operativo y las alteraciones típicas que permitan alterar el sistema, adquirir información del dispositivo y persistencia en el dispositivo. A continuación se han revisado los diferentes mecanismos utilizados para interaccionar con el sistema y su posterior compromiso. Habiendo sido detectados algunas de las funcionalidades del sistema que son usadas para infectar el dispositivo queda en evidencia la criticidad de monitorización de dichos procesos. Para finalizar con esta sección, se ha analizado el tráfico de red realizado por las muestras y los protocolos de red utilizados. El análisis del tráfico interno de la red es de vital importancia para la detección de técnicas de movimiento lateral que puedan comprometer a otros dispositivos de la red. Por otro lado, el análisis del tráfico externo puede permitir detectar comunicaciones contra servidores *Command & Control* activos o inactivos, siendo de especial dificultad la detección de estas conexiones sin un sistema de monitorización de red y un sistema de detección de intrusiones adecuado.

En la Sección 4.4 se han introducido el concepto de patrón de comportamiento y su vital

importancia para realizar la correlación de eventos unitarios que permita identificar cadenas de acciones maliciosas. Este tipo de detección permite la variación del punto de vista clásico sobre la detección del *malware*. No es necesario identificar la muestra como potencialmente maliciosa si es posible detectar y bloquear las acciones que vayan a impactar en la salud del sistema operativo.

Por último, en la Sección 4.5 se han realizado una serie de correlaciones entre la información generada por `Cuckoo` con fuentes externas, permitiendo obtener nuevas conclusiones y comprobar algunas de las hipótesis planteadas.

En conclusión, a pesar de que el número de muestras de *malware* siga aumentando constantemente así como la incorporación de nuevas técnicas de ofuscación y evasión por actores maliciosos, la cantidad de información que se puede extraer del análisis puede permitir su detección. Siendo necesario realizar un análisis en profundidad de todas las interacciones con el sistema y la agregación de la información obtenida a través del análisis estático, dinámico, las firmas de comportamiento disponibles así como de la información disponible de distintas fuentes de inteligencia externa.

Aunque la cantidad de muestras analizadas corresponde a una fracción muy pequeña de la diversidad del *malware* disponible, la cantidad de información generada durante el análisis supone un enorme reto para realizar una clasificación de las muestras. Aunque el análisis automatizado permite la obtención de estas características de una forma relativamente sencilla no permite la clasificación automática de forma fidedigna.
Para el procesamiento de la información y la clasificación de las muestras, es posible la utilización de técnicas de *machine learning* e inteligencia artificial para automatizar el proceso.

5.2. Trabajo Futuro

Un aspecto crítico a la hora de analizar una muestra es el sistema de*scoring* que marcará la confianza sobre cada muestra. El desarrollo de un sistema de *scoring* que permita la incorporación de la información obtenida mediante `Cuckoo` y fuentes de inteligencia externas se ha propuesto como aspecto crítico a ser evaluado en un futuro.

Por otro lado, durante la realización del análisis estático se ha comprobado la falta indicadores que puedan confirmar una alteración de las secciones de la cabecera de los ficheros ejecutables. Se propone el cálculo de la entropía que permita la detección de secciones empaquetadas, cifradas o creadas como señuelo.

En relación con el análisis del tráfico de red generado, se ha propuesto la inclusión de un sistema de detección de intrusiones que permita identificar comunicaciones sospechosas o anómalas de una forma simple y eficaz.

Bibliografía

[1] G. C. Abadías. Cuckoommander, 2020. "https://github.com/gcebollero/Cuckoommander".

[2] B. Alsulami and S. Mancoridis. Behavioral malware classification using convolutional recurrent neural networks. *CoRR*, abs/1811.07842, 2018.

[3] C. Balles and A. Sharfuddin. Breaking imphash. *CoRR*, abs/1909.07630, 2019.

[4] C. T. d. Brian Baskin. Conti ransomware, 2020. "https://www.carbonblack.com/blog/tau-threat-discovery-conti-ransomware/".

[5] K. S. Bulletin. The ransomware revolution, 2016. "https://securelist.com/kaspersky-security-bulletin-2016-story-of-the-year/76757/".

[6] H. Carvey. The windows registry as a forensic resource. *Digital Investigation*, 2(3):201–205, 2005.

[7] M. Corporation. Process monitor, 2020. "https://docs.microsoft.com/en-us/sysinternals/downloads/procmon".

[8] Cuckoo. repository of modules and signatures contributed by the community, 2020. https://github.com/cuckoosandbox/community.

[9] T. V. Foundation. Volatility framework, 2020. "https://www.volatilityfoundation.org/".

[10] D. Gavriluţ, M. Cimpoeşu, D. Anton, and L. Ciortuz. Malware detection using machine learning. In *2009 International Multiconference on Computer Science and Information Technology*, pages 735–741, 2009.

[11] R. P. R. Guevara. *Tools for the detection and analysis of potentially unwanted programs*. PhD thesis, 2018.

[12] HybridAnalysis. Free automated malware analysis service - powered by falcon sandbox, 2020. "https://www.hybrid-analysis.com/".

[13] P. Jupyter. Jupyter's next-generation notebook interface, 2020. "https://jupyter.org/".

[14] N. Kheir, G. Blanc, H. Debar, J. Garcia-Alfaro, and D. Yang. Automated classification of c&c connections through malware url clustering. *ICT Systems Security and Privacy Protection*, pages 252–266, 2015.

[15] K. Kono, S. Phomkeona, and K. Okamura. An unknown malware detection using execution registry access. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, volume 02, pages 487–491, 2018.

[16] List of tcp and udp port numbers, 2020. https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers.

[17] A. Moser, C. Kruegel, and E. Kirda. Limits of static analysis for malware detection. In *Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007)*, pages 421–430, 2007.

[18] A. Mundo. Ransomware maze | mcafee blogs, 2020.

[19] L. Pazmiño, F. Flores, L. Ponce, J. Zaldumbide, V. Parraga, B. Loarte, G. Cevallos, I. Maldonado, and R. Rivera. Challenges and opportunities of iot deployment in ecuador. In *2019 International Conference on Information Systems and Software Technologies (ICI2ST)*, pages 108–115. IEEE, 2019.

[20] M. Pieter Arntz. Maze: the ransomware that introduced an extra twist, 2020. "https://blog.malwarebytes.com/threat-spotlight/2020/05/maze-the-ransomware-that-introduced-an-extra-twist/".

[21] K. Platon, M. Srdjan, R. Richard, and C. Juan. Certified pup: Abuse in authenticode code signing. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, page 465–478, New York, NY, USA, 2015. Association for Computing Machinery.

[22] R. Rivera. Análisis de características estáticas de ficheros ejecutables para la clasificación de malware. Master's thesis, Universidad Politécnica de Madrid, 2014.

[23] R. P. Rivera-Guevara. Deteccion y clasificacion de malware con el sistema de análisis de malware cuckoo. Master's thesis, 2018.

[24] N. A. Rosli, W. Yassin, F. M.A, and S. Rahayu. Clustering analysis for malware behavior detection using registry data. *International Journal of Advanced Computer Science and Applications*, 10(12), 2019.

[25] M. Sebastián, R. Rivera, P. Kotzias, and J. Caballero. Avclass: A tool for massive malware labeling. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 230–253. Springer, 2016.

[26] J. W. Stokes, D. Wang, M. Marinescu, M. Marino, and B. Bussone. Attack and defense of dynamic analysis-based, adversarial neural malware classification models. *CoRR*, abs/1712.05919, 2017.

[27] T. This. Mozilla fundation, 2020. "https://www.trackthis.link/".

[28] C. G. A. T. J. B. M. S. K. H. R. van Zutphen y Ben de Graaff. Cuckoo sandbox, 2012. "https://cuckoosandbox.org".

[29] S. Vemparala. Malware detection using dynamic analysis. Master's thesis, San Jose State University, 2015.

[30] C. Ventures. Global ransomware damage costs predicted to reach $20 billion (usd) by 2021, 2019. "https://cybersecurityventures.com/global-ransomware-damage-costs-predicted-to-reach-20-billion-usd-by-2021/".

[31] VirusShare. Virusshare.com - because sharing is caring, 2020. "https://virusshare.com".

[32] Virustotal. "https://www.virustotal.com".

[33] VirusTotal. Yara. the pattern matching swiss knife for malware researchers. "https://virustotal.github.io/yara/".

Apéndice A

Apéndice

Port	Host	ASN	Tramas
80	137.74.163.43	OVH SAS	1
	2.17.152.87	Akamai Technologies, Inc.	1
	23.50.56.123	Akamai Technologies, Inc.	1
	216.58.211.238	Google LLC	2
	172.217.168.174	Google LLC	1
	87.248.100.215	Yahoo! UK Services Limited	1
	216.58.211.46	Amazon.com, Inc.	2
	212.166.133.204	VODAFONE ESPANA S.A.U.	4
	13.224.106.115	Amazon.com, Inc.	1
	13.32.90.38	Amazon.com, Inc.	1
	13.33.234.144	Amazon.com, Inc.	1
443	193.235.207.60	Cogent Communications	1
	23.10.77.13	Akamai Technologies, Inc.	1
	23.216.122.214	Akamai Technologies, Inc.	96
	44.231.216.202	Amazon.com, Inc.	1
	216.58.211.227	Google LLC	1
	172.217.17.3	Google LLC	2
	216.58.211.35	Google LLC	4
	204.11.56.48	Confluence Networks Inc	1
8080	50.7.243.114	Cogent Communications	1
13076	195.154.242.226	ONLINE S.A.S.	1

Tabla A.1: *Dead host contactados*

Host	Reverse resolution	Dominios
137.74.163.43	43.ip-137-74-163.eu	5
2.17.152.87	a2-17-152-87.deploy.static.akamaitechnologies.com	5
23.50.56.123	a23-50-56-123.deploy.static.akamaitechnologies...	152
216.58.211.238	mad01s24-in-f14.1e100.net	44
172.217.168.174	mad07s10-in-f14.1e100.net	293
87.248.100.215	media-router-fp73.prod.media.vip.ir2.yahoo.com	39
216.58.211.46	muc03s14-in-f46.1e100.net	111
212.166.133.204	r1.sn-8vq54voxn25po-cjol.gvt1.com	3
13.224.106.115	server-13-224-106-115.mad50.r.cloudfront.net	117
13.32.90.38	server-13-32-90-38.mad51.r.cloudfront.net	8
13.33.234.144	server-13-33-234-144.mad51.r.cloudfront.net	87
193.235.207.60	60.207.235.193.in-addr.arpa	2
23.10.77.13	a23-10-77-13.deploy.static.akamaitechnologies.com	3
23.216.122.214	a23-216-122-214.deploy.static.akamaitechnologi...	2
44.231.216.202	ec2-44-231-216-202.us-west-2.compute.amazonaws...	3
216.58.211.227	mad01s24-in-f3.1e100.net	123
172.217.17.3	mad07s09-in-f3.1e100.net	142
216.58.211.35	muc03s14-in-f35.1e100.net	126
204.11.56.48	ns1.verification-hold.suspended-domain.com.	178
50.7.243.114	No resolutions available	0
195.154.242.226	server02.quebecorp.com	1

Tabla A.2: *Reverse resolutions de los dead hosts*