E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

# Algoritmo de detección de ransomwares mediante tráfico SMB en redes con directorios compartidos (REDFISH)

Máster Universitario en
Ingeniería de Telecomunicación

Trabajo Fin de Máster

Eduardo Berrueta Irigoyen

Daniel Morató Osés

Pamplona, 18 de junio de 2018

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

# *Índice*

# *Índice de figuras*

# Índice de tablas

**Resumen**

Este trabajo presenta una solución para detectar una infección por ransomware en un equipo de una red local con directorios compartidos por SMB en uno o varios servidores. Se basa en el análisis de tráfico de las versiones 1 y 2 de este protocolo, en la cantidad de bytes leídos y escritos y en las eliminaciones que haga el usuario en ficheros del servidor. Deben establecerse tres parámetros que caracterizarán al algoritmo ($N$, $T$ y $V_{umbral}$), y que determinarán la cantidad de ficheros que encriptará el ransomware antes de su detección. Aunque esos $N$ ficheros van a ser encriptados en todos los casos en que se detecte el ransomware, se ha desarrollado una herramienta de recuperación para conseguir recuperar estos ficheros, de forma que podemos considerar la herramienta como *sin pérdidas*.

Los resultados son del 100 % de detección de ransomware con una probabilidad de falso positivo menor del 1 % en la mayoría de los días testeados. Las pruebas se han realizado con un total de 53 muestras distintas de ransomware de 18 familias diferentes, corriendo en un entorno virtualizado. Los falsos positivos se han evaluado con muestras de tráfico de usuario de 6 días laborables en la red local de la UPNA y 1 día completo en otra red empresarial.

**Abstract**

This work presents a novel solution for detecting a ransomware infection of a computer in a LAN network with one or more shared directories by SMB protocol. It is based on the SMB traffic analysis, counting the read and written bytes by the user or by the server and the deletion of shared files. Three main parameters must be set in order to the algorithm to work properly ($N$, $T$ and $V_{umbral}$), and they will determine how many files will be encrypt by the ransomware until its detection. Although these $N$ files must be encrypted in all detection cases, a file-recovery tool has been developed in order to make the algorithm zero-losses.

It has been achieved 100% of detection with plenty low false positive rate (lower than 1% in most of the days tested). The tests have been performed with 53 different ransomware samples of 18 families, running in a virtualized environment. The false positive rate has been evaluated using user traffic samples of 6 days in the local network of the UPNA, and 1 complete day of other enterprise network.

***Keywords***— ransomware, análisis de tráfico, seguridad, malware.

# 1. Introducción

Un ransomware es un tipo de malware que está cobrando mucha importancia en los últimos años a nivel mundial (entre 2015 y 2016 el número de ataques se multiplicó por cinco) [34]. Se basa en impedir el acceso a los archivos y pedir una recompensa para recuperarlo. En 2016 la cantidad media pedida por cada ransomware fue de $679 [9] y una vez pagada, lo habitual es que el acceso se restaure, aunque nada lo garantiza. A pesar de esto, no es recomendable su pago, ya que incentiva la creación de nuevas versiones.

Hay dos tipos de ransomware, en función de cómo imposibilitan el acceso a los archivos de la máquina. El primero de ellos, llamado locker-ransomware, bloquea el equipo mostrando un mensaje con las instrucciones a seguir para pagar la recompensa y recuperar el acceso. Es la infección menos dañina, pues se pueden recuperar los archivos del disco duro, ya que no han sido dañados. El segundo tipo, denominado crypto-ransomware o cryptoware, es aquel que no bloquea el equipo, sino que encripta los archivos personales de los usuarios a través de distintos algoritmos de encriptación. Muchos de ellos son avanzados (por ejemplo RSA-2048) y no es factible intentar desencriptarlos. Además, en muchas ocasiones la clave no se guarda en la propia máquina, sino que se almacena en el servidor de control (C&C), lo que hace prácticamente imposible acceder a ella. En algunos casos, el ransomware no solo encripta los ficheros que están alojados en la máquina del usuario, sino que hace lo mismo con los ficheros almacenados en un volumen en red, al que se accede a través de un protocolo concreto. Esto hace que este malware tenga un impacto muy importante en empresas como por ejemplo hospitales, departamentos de policía o empresas financieras. Una mala gestión de los permisos de los usuarios, puede provocar que un único usuario infecte todo el volumen compartido. Si se hace una gestión correcta, solo podrá encriptar aquellos archivos a los que tenga acceso. La única solución actualmente una vez que los archivos de una empresa han sido encriptados, es la recuperación de las máquinas por medio de backups [52]. La frecuencia a la que se hagan, dependerá de la gestión de la propia empresa, siendo habitual que se hagan diariamente (normalmente a la noche).

Debido al impacto y al coste que tiene para las empresas [6] la EUROPOL declaró en un informe de 2016 que el cryptoransomware se ha vuelto la amenaza más destacable para particulares y empresas [37].

Se estima que se han extorsionado cientos de millones de dólares de usuarios año a año [9]. Esto hace que el cryptoware sea el malware más rentable de la historia [31], afectando tanto a usuarios particulares como a todo tipo de empresas. En relación a estas últimas, una encuesta llevada a cabo en Estados Unidos, Alemania, Canadá y Reino Unido, ha concluido que el 50 % han sido víctimas de la extorsión de un cryptoware en el último año [11]. Aunque pagar la recompensa que piden no asegura la recuperación de los archivos y las autoridades recomiendan no pagarla para no favorecer a los delincuentes, alrededor del 40 % de las empresas entrevistadas confesaban haber pagado lo que pedían.

Además de encriptar tanto archivos en local como archivos compartidos en volúmenes en red, algunos cryptowares incluyen un comportamiento de tipo gusano, que se expande por la red infectando el resto de los equipos de la red local. Incluso algunas variantes recientes como el "WannaCry" intentan expandirse hacia la internet a través del ataque a IPs aleatorias [62, 49].

Ya se ve que en estos casos, un solo usuario infectado puede provocar la encriptación de todo un volumen compartido, impidiendo trabajar al resto de usuarios de la empresa, con el impacto económico que eso supone [65]. La política habitual de backups en las empresas es hacerlos cada noche, y se convierte en la principal forma de recuperarse cuando se produce una infección [11]. A pesar de que de esta forma se recuperan todos los datos comprometidos por el ransomware, se pierden las horas de trabajo que se hayan realizado desde el último backup. Esto supone además de

un impacto económico, un impacto a la imagen de la empresa de cara a los clientes y a la opinión pública. También es habitual tener que indeminzar a los clientes afectados ya que sus datos han sido expuestos a un malware.

Las empresas que se ven más afectadas cuando sufren estos ataques (y quizá por eso son las que más habitualmente los sufren), son las empresas financieras y relacionadas con la salud. Además son las que más beneficios producen a los delincuentes, ya que muchas veces éstas se ven obligadas a pagar la recompensa para seguir con su actividad de la cual. En el caso de los hospitales, por ejemplo, la encriptación de los archivos puede suponer una mala administración de los medicamentos o algunos problemas en el diagnóstico de los pacientes. [31, 49].

Los atacantes aprovechan muchas veces el retraso en la renovación o la actualización del software, como en el caso del "WannaCry", que infectó una gran cantidad de empresas en Mayo de 2017 [62]. También ocurrió de forma similar en el caso del ransomware SamSam, infectando los servidores de JBoss [31]. Estos retrasos muchas veces son inevitables, ya que aunque se trate de pequeños cambios en el software, deben ser validados para comprobar su compatibilidad con el software actual de la empresa.

El impacto no es tanto por la infección de los equipos de usuario, sino por la infección de los archivos comunes a todos ellos, que se alojan en un servidor de contenidos. En este trabajo se va a presentar un algoritmo de detección y bloqueo del ransomware cuando intenta encriptar los ficheros compartidos en estos volúmenes en red. Este escenario de ficheros compartidos por todos los usuarios en la misma LAN es muy habitual, y aún así no hay soluciones para frenar las posibles infecciones. A pesar de que en los hosts puede haber instalados antivirus actualizados, el continuo avance en las versiones de ransomwares y la aparición de nuevas familias o variantes con pequeñas mutaciones en el código, hace que este tipo de malware sea indetectable con los métodos tradicionales. Estos antivirus muchas veces utilizan algoritmos de detección *signature-based*, que no son eficaces contra los cryptowares [57]. Para poder detectar los ransomwares es necesario realizar un análisis basado en el comportamiento de los usuarios, con la dificultad que esto supone, pues hay que conseguir diferenciar entre comportamientos maliciosos y comportamientos de usuario.

Este trabajo está centrado en volúmenes de red compartidos usando el protocolo Server Message Block (SMB) porque es el más utilizado en entornos empresariales y de oficinas. Sin embargo, la idea y el procedimiento del algoritmo puede exportarse a otro tipo de protocolos de compartición de archivos. Mientras las soluciones que hay actualmente para la detección de ransomwares son en su mayoría soluciones que deben ser instaladas localmente en los hosts [33, 43, 61], la idea presentada en este documento es un software que no requiere de esta instalación, sino que está situado en un punto de la red desde el que recibe el tráfico de los usuarios. De esta forma nos evitamos que el propio malware desinstale el software y también que éste consuma recursos del ordenador del usuario.

Se explicará cómo el software es capaz de analizar el tráfico de los usuarios y detectar el ransomware analizando su comportamiento (*behaviour-based algorithm*). Además, el software podrá actualizarse más fácil que si estuviera instalado en cada uno de los equipos de la red, ya que es una sola máquina la que debe ser actualizada. Un capturador de tráfico recibirá una copia de los paquetes de los usuarios gracias a un *port-mirror* (ver figura 3). Con este tráfico de usuario, va a analizar las peticiones de archivos, además de acciones como lecturas, escrituras o eliminaciones de datos, pudiendo tomar medidas si detecta algún comportamiento sospechoso (bloqueo de un determinado host). Todo esto sin introducir retardos en la conexión de los usuarios y sin consumir recursos de las máquinas de los mismos.

Los experimentos se han realizado con 53 muestras de ransomware de 18 familias distintas. Todas las muestras han sido detectadas en menos de 20 segundos y en más del 99 % de los casos se han perdido menos de 10 ficheros antes de saltar la alarma, e incluso esos 10 ficheros se podrán recuperar gracias a un software de recuperación desarrollado que se explicará en el último apartado de este

documento.

Este trabajo tiene una serie de contribuciones principales:

- Es el primer algoritmo presentado capaz de detectar un ransomware en un entorno de red con volúmenes compartidos.
- El algoritmo se ha validado con 19 familias de ransomware y tráfico de miles de usuarios sacados de entornos empresariales (2 entornos distintos para no ver sesgadas las estadísticas). Los falsos positivos se evitan configurando los parámetros del algoritmo en función del comportamiento de los usuarios que se observa en las trazas de entrenamiento.
- Se proporciona un modelo analítico del porcentaje de detección y de error del algoritmo.
- Describe una herramienta de análisis de tráfico que permite detectar la infección de equipos por ransomwares sin instalar ningún software en local y sin introducir ningún tipo de retardo al tráfico de usuario. La disposición de los equipos permite también desarrollar una herramienta de recuperación de archivos que reconstruye los ficheros encriptados por el ransomware antes de su detección (típicamente serán 10) a partir del tráfico visto en la red.

El trabajo se estructura de la siguiente manera: En la sección 2 se explican brevemente las herramientas existentes y algunas de las familias de ransomware más importantes. A continuación (sección 3) se desarrolla el escenario en el que se van a realizar los experimentos, tanto las trazas no infectadas como las infectadas. En la sección 4 se compara el comportamiento típico de usuario y el comportamiento de los ransomwares, para analizar en qué aspectos se pueden diferenciar para desarrollar el algoritmo de detección. La sección 5 explica el propio algoritmo, y analiza los parámetros para seleccionar aquellos que reducen los falsos positivos y maximizan el índice de detección. En la siguiente sección (sección 6) se exponen los resultados del algoritmo por familia de ransomware y de forma global. En la sección 7 se estudia brevemente el rendimiento, los posibles problemas de seguridad y la posibilidad de recuperar los archivos que se han perdido antes de la detección del ransomware. Se termina con una serie de conclusiones en la sección 8 extraídas de la realización del trabajo.

# 2. Estado del arte

En esta sección se va a hacer un repaso cronológico de las familias de ransomware más importantes que ha habido, explicando brevemente algunas características importantes y algunos datos de impacto cuando sea posible. A continuación se resumirán las herramientas de detección (comerciales y no comerciales) más importantes que se han desarrollado, clasificándolas en función de sus características y analizando los problemas que puede presentar cada una de ellas.

## 2.1. Ransomwares a lo largo de la historia

Aunque hasta hace cuatro o cinco años el ransomware era un malware bastante desconocido en general, el primero de ellos data de 1989 y se llamaba AIDS Trojan. Aunque había muchas versiones del mismo, ninguna de ellas fue muy efectiva porque la encriptación utilizada era muy débil [42].

Dado el poco éxito de este primer ransomware, en los años siguientes no hay ninguno importante y hay que irse hasta el año 2013 para encontrar los primeros con un impacto bastante elevado y que funcionaban de forma muy similar a los que se conocen hoy en día. El primero que apareció en este año es el CryptoLocker, muy importante porque comenzó a utilizar cifrado asimétrico y su comportamiento no se diferenciaba mucho del comportamiento que tienen ahora los más avanzados [35]. El impacto fue bastante elevado, seguramente por tratarse del primero, y se estima que recolectó más de 1 millón de dólares [36].

En 2014 aparecen nuevas familias de ransomware y se actualizan algunas versiones de CryptoLocker, aunque ninguna tiene demasiada importancia. Avanzando hasta el 2015, el ransomware más importante ese año es TeslaCrypt, que sin ninguna característica muy distinta a CryptoLocker, consigue ganar más de 70.000 dólares en dos meses [32].

Durante el año siguiente (2016) los ataques se multiplican por cinco [34], y la razón más importante es el desarrollo del ransomware Locky en Febrero de ese mismo año. Locky genera los dominios de sus servidores de control mediante DGA (Domain Generation Algorithm) [25], lo que provoca que algunas herramientas no sean capaces de detectarlo. Entre Abril y Mayo de ese mismo año, Locky ya representaba el 34 % de las infecciones totales por ransomware [66], llegando a infectar 4000 dispositivos por hora [20].

A mediados de 2016 aparece otra nueva familia muy importante: Cerber. En [38] la ponen al nivel de Locky en cuanto a importancia, ya que tenía ganancias de cerca de 7 millones de dólares [29].

El número de variantes aparecidas en 2017 multiplica por 3 las aparecidas el año anterior [39]. Este año también destaca porque se producen tres ataques con los que los ransomwares se hicieron conocidos a escala mundial. El primero de ellos se produce en Mayo y se trató del ransomware WannaCry. Hay empresas que llegan a perder 41 millones de dólares ó incluso 117 millones, como se reporta en [67]. En el mismo artículo mencionan que las ganancias de los responsables rondan los 140 mil dólares.

Poco después se produce el ataque del ransomware Petya. Aunque su primera aparición data de 2016, su ataque más importante se produce en 2017. Afectó sobre todo a países del este de Europa, aunque también llegó a empresas de España y Gran Bretaña [17]. Aunque las consecuencias económicas para las empresas en general no fueron tan altas como con el ransomware anterior, la empresa danesa Maersk fue una excepción, ya que valoraban las pérdidas en 200 o 300 millones de dólares [58].

El último ataque importante se produce poco tiempo después a cargo de NotPetya. Un ransomware con con algún parecido a Petya pero que actuó de forma distinta. El impacto sobre las empresas está al nivel del WannaCry, con costes de 300 millones de dólares para FedEx, por ejemplo [16].

Durante el año 2018 ha aparecido una familia importante: GandCrab, que desde principios de año ya se cree que los desarrolladores han ganado 600.000 dólares [19].

La previsión a futuro, según indica Kaspersky en [13], es que el objetivo de los desarrolladores sean países emergentes donde no se tenga conocimiento de anteriores ataques o donde las herramientas de detección todavía no estén del todo implantadas en las empresas.

Viendo la gran cantidad de variantes y de familias existentes, se hace patente la necesidad del desarrollo de herramientas de detección efectivas y rápidas. En la siguientes sub-sección se enumerarán las que existen actualmente con los problemas que presentan, para justificar la necesidad del algoritmo desarrollado en este artículo.

## 2.2. Herramientas de detección

Durante estos últimos tres años aproximadamente, se han desarrollado herramientas buscando tanto la detección como la recuperación tras infecciones de ransomwares. Además de algunas comerciales desarrolladas por empresas de antivirus y de backup, hay muchos proyectos y trabajos de investigación publicados en revistas y conferencias que buscan la mejor manera de detectar este malware. Dado que este trabajo trata sobre un algoritmo de detección, se explicarán las herramientas que tienen este mismo propósito para analizar las posibles semejanzas y diferencias de éstas con el algoritmo presentado.

Se van a dividir las herramientas de detección entre comerciales y no comerciales. Hay que tener en cuenta que en ocasiones no se dispone de muchos detalles de funcionamiento para las primeras.

### 2.2.1. Herramientas no comerciales

En todas las herramientas se buscan una serie de características con las que diferenciar un ransomware de un comportamiento normal de usuario. En función de estas características se detectarán mejor o peor las familias de ransomware y se tendrá una tasa de falsos positivos (casos para los que la herramienta ha detectado la presencia de un ransomware cuando en realidad no lo había). Normalmente hay un compromiso entre el porcentaje de familias que es capaz de detectar y el número de falsos positivos que produce la herramienta. De ahí que haya que seleccionar muy bien las características con las que detectarlos y los umbrales con los que va a saltar la alarma.

En función de las características seleccionadas por cada una de las herramientas, se van a dividir en cuatro grupos, pudiendo haber alguna que tenga características en más de uno de ellos. En la tabla 1 se ve dicha clasificación con una breve descripción de cada grupo.

Al primer grupo de características se le ha denominado análisis local dinámico. Esto engloba todo el análisis de funciones del sistema, actividad en directorios, lecturas, escrituras y eliminaciones de archivos, primitivas criptográficas utilizadas por las funciones de encriptación y todo lo relacionado con los ficheros modificados por las mismas (extensión, tamaño, entropía entre datos escritos y leídos, etc.). La gran mayoría de las herramientas utilizan características de este grupo, para lo cual se debe monitorizar el equipo del usuario. Deben estar instaladas en el propio equipo, pudiendo ser desinstaladas por el malware si este es capaz de escalar privilegios y consumiendo recursos (procesador, memoria) de la máquina del usuario.

Como se ve en la tabla 1, las diez primeras herramientas utilizan solo este conjunto de características, aunque dentro de él hay varios tipos. Amin Kharraz et al. por ejemplo, utiliza *canary files* para no tener que monitorizar las llamadas a todas las funciones del sistema. Distribuye estos ficheros por los directorios del usuario y analiza cualquier modificación en ellos, de forma que puede detectar una encriptación de los mismos. PayBreak solo analiza la llamada a una función criptográfica del sistema para ir almacenando las claves y en caso de infección poder desencriptar los archivos. En

este caso no se trataría de una herramienta de detección sino de recuperación de los ficheros.

En cuanto a los resultados, los mejores los consigue ShieldFS con un 100 % de detección de muestras (de un total de 383 de 11 familias distintas) y solo un 0.038 % de falsos positivos entre el total de 2245 aplicaciones de 11 usuarios distintos. Además, analiza la sobrecarga que tiene la herramienta para el equipo (runtime overhead de 0.26). Es capaz de recuperar los archivos que encripta el ransomware antes de su detección, copiando los mismos a un sector de memoria protegido contra escritura cada vez que un proceso abre algún fichero con permisos de escritura y/o eliminación.

| Herramientas | Local dinámico | Local estático | Machine Learning | Red |
|---|---|---|---|---|
| **CryptoLock** [61] | ✓ | ✗ | ✗ | ✗ |
| **Amin Kharraz et al.** [44] | ✓ | ✗ | ✗ | ✗ |
| **Faustin Mbol et al.** [53] | ✓ | ✗ | ✗ | ✗ |
| **UNVEIL** [46] | ✓ | ✗ | ✗ | ✗ |
| **PayBreak** [47] | ✓ | ✗ | ✗ | ✗ |
| **H. Kim et al.** [45] | ✓ | ✗ | ✗ | ✗ |
| **ShieldFS** [33] | ✓ | ✗ | ✗ | ✗ |
| **FileTracker** [64] | ✓ | ✗ | ✗ | ✗ |
| **Redemption** [43] | ✓ | ✗ | ✗ | ✗ |
| **Yun Feng et al.** [70] | ✓ | ✗ | ✗ | ✗ |
| **Vinayakumar et al.** [68] | ✓ | ✗ | ✓ | ✗ |
| **Rapper** [26] | ✓ | ✗ | ✓ | ✗ |
| **2entFox** [24] | ✓ | ✗ | ✓ | ✗ |
| **Mohammad Mehdi et al.** [25] | ✓ | ✗ | ✗ | ✓ |
| **Krzysztof Cabaj et al.** [30] | ✗ | ✗ | ✗ | ✓ |
| **Raptor** [60] | ✗ | ✗ | ✗ | ✓ |
| **T. Lu et al.** [50] | ✓ | ✗ | ✓ | ✓ |
| **EldeRan** [63] | ✓ | ✓ | ✓ | ✗ |
| **RansHunt** [40] | ✓ | ✓ | ✓ | ✓ |

**Tabla 1:** Clasificación de las herramientas no comerciales

Otras soluciones como Redemption también tienen el 100 % de efectividad en la detección, pero su índice de falsos positivos es algo mayor (0.8 %). Pierde de media 5 ficheros, que luego recupera con la misma técnica que ShieldFS. Los experimentos se realizan con 677 muestras de 29 familias de ransomware y el comportamiento de usuario consiste en el acceso a 50200 archivos. El resto tienen peores resultados o no los mencionan, pero estas diez primeras herramientas comparten que están instaladas en local y que tienen que monitorizar las llamadas del usuario a distintas funciones del sistema. Estas herramientas, en cierto modo se parecen al algoritmo que se presentará en este trabajo porque éste va a analizar las lecturas y escrituras de los usuarios a ficheros del servidor NAS, lo que es equivalente al análisis local dinámico mencionado pero a través de la red, así se evita el problema de la desinstalación y del consumo de recursos en local.

Las tres herramientas siguientes en la tabla, además de realizar el análisis local dinámico, introducen estas características en un motor de machine learning, lo que permite un aprendizaje automático y la posibilidad de analizar un número mucho más alto de características que en los casos anteriores. Por otro lado, en cierto modo se pierde el control sobre la propia herramienta porque se confía en el autoaprendizaje de la misma. Aunque se utilizan distintas redes neuronales, la idea es similar en los tres casos: seleccionar una serie de características como las de las diez anteriores e insertarlas en el algoritmo como entradas, de forma que gracias al aprendizaje con comportamientos de usuarios, pueda deducir si se trata de un equipo infectado o no. Cabría esperar que los resultados fueran mejores que con las herramientas anteriores, pero no es así. Vinayakumar et al. y la herramienta 2entFox consiguen detecciones del 98 % con falsos positivos menores del 1 % en el primer caso. Con

Rapper se detectan el 100 % de las muestras, pero se han hecho las pruebas con una sola muestra de WannaCry, por lo que no es muy representativo.

Las tres herramientas siguientes en la tabla (M. Mehdi et al. K. Cabaj et al. y Raptor) analizan el comportamiento en red de cada máquina. En primer lugar, conviene aclarar que estas herramientas también deben estar instaladas en el equipo del usuario, por lo que los problemas mencionados antes persisten. M. Mehdi et al. combina el análisis local dinámico con el análisis en red buscando una mejor diferenciación del comportamiento del ransomware. El análisis en red consiste en capturar el tráfico y observar las peticiones DNS, intercambio de claves con servidores, etc. El caso de K. Cabaj et al. es algo distinto porque establece un entorno de SDN (Software Define Networking) donde las peticiones DNS que pasan por un switch son enviadas al controlador para ver si el dominio está en la lista negra del mismo. En caso de ser así, éste instala una regla en el switch para no permitir el tráfico de ningún equipo de la red con ese dominio. Las herramientas de análisis en red trabajan habitualmente con estas listas negras de dominios, lo cual tiene el problema de la aparición de nuevas muestras con nuevos servidores de control que todavía no se han listado y por tanto no pueden ser bloqueados. Los resultados de M. Mehdi son los mejores, con un 100 % de detección y 0 % de falsos positivos. Los experimentos se realizan con 20 muestras de ransomware pero no se especifica con qué muestras se calculan los falsos positivos.

T Lu et al. desarrollan una herramienta, apoyada en el *machine learning*, que combina el análisis en local con el análisis en red. Ambos conceptos se han explicado ya con anterioridad y en este caso la inteligencia artificial le facilita el hecho de extraer una mayor cantidad de características y combinarlas para conseguir mejores resultados. A pesar de esto, consigue un 95 % de éxito en la detección con entre 7 y 8 % de falsos positivos, analizados con 1000 muestras de ransomware y 1000 de aplicaciones de usuario.

Las dos últimas son bastante similares. Ambas introducen un nuevo tipo de análisis, el análisis local estático, que consiste en la inspección de los binarios de los ejecutables para ver si tienen alguna característica sospechosa. Esta técnica no es novedosa en la detección de malware en general, pero no se había encontrado en herramientas de detección de ransomware. Buscan llamadas a funciones criptográficas, strings característicos de los ransomwares, etc. RansHunt analiza también el tráfico del equipo en la red, lo que le puede aportar más información, y ambos emplean algoritmos de inteligencia artificial para el análisis de las características extraídas. Los resultados son bastante similares: EldeRan tiene un porcentaje de detección de 93 % con 582 muestras de ransomware, mientras que RansHunt tiene un 97 % con 360 muestras. En cuanto a los falsos positivos, un 1.6 % y un 3 % respectivamente.

Como vemos, todas las herramientas comparten los problemas de la instalación en local y el consumo de recursos. Algunas de ellas recuperan los archivos gracias a backups de los mismos previos a las escrituras por parte del ransomware, y otros analizan el tráfico en red. Dentro de estos cinco últimos, ninguno analiza el tráfico de la forma que se va a realizar en este trabajo (cliente y servidor en red NAS).

Vemos pues que se trata de una solución original y novedosa, que aborda un problema que no está resuelto y que, como hemos visto en la sección 2.1, provoca grandes pérdidas en las empresas.

### 2.2.2. Herramientas comerciales

Las empresas de antivirus y algunas de gestión de redes también han desarrollado sus herramientas para solventar los problemas asociados con los ransomwares. En esta sección se analizarán las más importantes y se dividirán entre aquellas que buscan la detección del malware y las que tratan de lograr la recuperación de los equipos tras la infección.

En relación al segundo grupo, la más importante es la desarrollada por la empresa Quorum

[14]. Se trata de una solución que consiste en guardar imágenes de los servidores de una red de forma que estos puedan recuperarse rápidamente en caso de infección. El problema de la solución es que siempre se van a perder los datos que hayan sido añadidos ó modificados en el servidor entre la última imagen guardada y la infección del ransomware. Aunque se pueden tomar imágenes del servidor frecuentemente para que la pérdida sea menor en caso de ataque, siempre va a haber datos que no se encuentren en la imagen guardada.

Pasando ya a las herramientas de detección, la más parecida a la solución propuesta en este trabajo es la desarrollada por ExtraHop, llamada Ransomware Bundle [18]. En ella se analiza el tráfico SMB entre cliente y servidor, pero no las operaciones sobre los archivos, sino las extensiones de éstos. La herramienta debe tener una lista negra de extensiones conocidas de los ransomwares para ver si en algún momento se escribe un archivo con alguna de ellas, en cuyo caso se tratará de un usuario infectado. El problema es que hay que tener esta lista actualizada en todo momento y ya se ha visto en la sección 2.1 que aparecen nuevas muestras de ransomware continuamente.

Soluciones como CryptoStopper [12] de WatchPoint colocan ficheros diseminados por los directorios de la máquina del usuario que son monitorizados para detectar la encriptación. En este caso es imprescindible que el ransomware comience encriptando estos archivos para que pueda ser detectado en poco tiempo y antes de cifrar los ficheros del usuario.

Varonis y NetApp desarrollaron una herramienta de detección basada en la caracterización del comportamiento de los usuarios que tienen instaladas otras herramientas de las empresas. También analiza las extensiones de los ficheros en los equipos de usuario para buscar aquellas conocidas por ser características de algún ransomware.

Por último, Cisco ha desarrollado una herramienta llamada Cisco Umbrella [10] que consiste en una lista negra con los dominios DNS pertenecientes a los ransomwares. Las empresas deben redireccionar las peticiones DNS para que pasen por el equipo que tenga instalado el software y que este las analice y determine si son o no maliciosas, de forma que pueda bloquearlas. El problema es similar a la lista negra con las extensiones, hay que tener la lista de dominios actualizada, y ya es sabido que hay algunos ransomwares que generan dominios por DGA (Ver sección 2.1) por lo que esta tarea es complicada.

# 3. Escenario

Para la realización de este trabajo se ha establecido un escenario de red LAN en el que un servidor de contenidos se encuentra accesible a través de una red IP (escenario NAS). Este escenario es habitual en empresas donde se comparten archivos entre varios usuarios o departamentos. Estos archivos se alojan en el servidor de contenidos, desde donde los usuarios van a tener acceso a ellos para leerlos, escribirlos y/o eliminarlos, dependiendo de los permisos que tengan sobre dichos ficheros.

En algunos casos, el servidor puede no limitarse al almacenamiento de ficheros, sino que podría contener también software al que los usuarios accederían también a través de la red IP proporcionada por la empresa. Este segundo caso no se va a contemplar en este trabajo, ya que los ransomwares tienen una serie de extensiones *objetivo* que van a intentar encriptar, y no suelen incluir extensiones propias de programas (Ver sección 4). Además, la encriptación de software no es tan crítica como puede serlo la de documentos de usuario, ya que el software se puede reinstalar, y además no se modifica desde la versión almacenada en el backup, con lo que se puede recuperar en la versión previa al ataque. En el Anexo B se muestran las extensiones *objetivo* más habituales de los ransomwares.

En la Figura 1 podemos ver un ejemplo de un escenario para el que está diseñado el algoritmo.

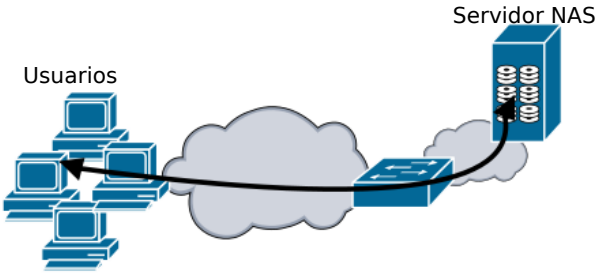



**Figura 1:** Escenario de red LAN

A lo largo de esta sección, se estudiarán los protocolos y características propias de estos escenarios, además de analizar el comportamiento habitual de los usuarios en cuanto al tráfico en la red, gracias a una serie de capturas en un entorno conocido que permitirá modelar este comportamiento "habitual". También se analizará cómo es el tráfico en uno de estos escenarios cuando un equipo en la red ha sido infectado.

## 3.1. Redes de almacenamiento

La red que se ha mostrado en la Figura 1 puede utilizarse para el almacenamiento de archivos en el servidor NAS (NAS filer). Para este intercambio (el almacenamiento supone envío y recepción de datos), los protocolos más habituales se encapsulan sobre TCP/IP. El algoritmo desarrollado en este trabajo se ha realizado para las versiones 1 y 2 del protocolo SMB debido a que es el protocolo que viene por defecto en el sistema operativo Windows y este, a su vez, es el sistema operativo más utilizado hoy en día en portátiles y ordenadores de sobremesa [15]. Aún así, es un algoritmo que podría llegar a adaptarse a otros protocolos, debido a que no se utiliza ninguna funcionalidad exclusiva de SMB para la detección del ransomware.

El protocolo SMB2 comienza a utilizarse en Windows Vista, y se mantiene como protocolo por defecto para el almacenamiento de archivos en red hasta Windows 8, que pasa a utilizar por defecto la versión 3 [27]. Se trata de un protocolo con estado y autentificación en cada conexión. Al comenzar

la comunicación de un equipo con el servidor se negocia la versión del protocolo a utilizar durante el resto de la comunicación. La versión 1 del protocolo se encuentra ya en desuso porque ha quedado relegada a Windows XP y anteriores, y la versión 3 todavía no está del todo implantada [27]. Aún así, todavía hay empresas que emplean la primera versión, y es por esto que nuestro algoritmo será capaz de procesar y analizar tanto la versión 2 como la 1.

En el caso de las dos versiones SMB tratadas en este trabajo, el servidor acepta peticiones TCP entrantes en el puerto 445 y/o 139. Cada equipo mantiene una conexión TCP con el servidor, aunque en caso de que este último utilice varias direcciones IP, cada host puede mantener abiertas varias conexiones TCP con él.

La conexión comienza con mensajes de negociación donde van a establecer la versión del protocolo a utilizar (NEGOTIATE REQUEST/RESPONSE). A continuación se negocia la encriptación y se asigna un ID de sesión y un ID de cliente (SESSION SETUP REQUEST/RESPONSE). Para acceder a un directorio determinado, se pide el acceso a través del comando TREE CONNECT. El servidor genera un Tree ID, que es un valor numérico que identifica la conexión del usuario con el árbol de directorios. En el momento que el usuario pida un archivo a través de un mensaje CREATE REQUEST, el servidor generará un identificador de este archivo y le asignará un ID de archivo (GUID) que se utilizará en el resto de peticiones del archivo en cuestión. A partir de aquí y a lo largo de todo el documento, al conjunto de peticiones y de mensajes SMB asociados a cada fileID se les denominará mensajes asociados a un fichero. Cada conjunto vendrá delimitado por un CREATE al inicio y un CLOSE al final. Un ejemplo de esta secuencia de paquetes para la versión 2 de SMB se puede ver en la figura 2. Para la primera versión del protocolo el nombre de los mensajes es algo diferente (READX en vez de READ) y el SETINFO no existe como tal, sino que hay mensajes específicos para el renombrado o la eliminación.

En este trabajo se presentan resultados de conexiones TCP individuales de cada usuario. El algoritmo se aplica por conexión y puede desplegarse para múltiples usuarios concurrentes empleando paralelismo o análisis secuencial del tráfico de los mismos.

El protocolo, en su primera versión (CIFS/SMB1), contaba con 75 comandos distintos [6] cada uno de ellos con su request y response. Esto hacía el protocolo bastante ineficiente por la cantidad de mensajes y comandos que había que enviar en la comunicación con el servidor. Así pues, se desarrolló la segunda versión del algoritmo con 19 comandos distintos [7]. Las posibilidades y las acciones que se pueden realizar sobre los archivos son similares a la primera versión, la principal diferencia es que se agrupan muchos de estos 75 comandos en uno más genérico (SET_INFO). Algunos de estos comandos son los siguientes:

- SMB2 CREATE [Request | Response]: El cliente envía el request al servidor para crear un archivo o para acceder a él en el caso de que ya exista en el directorio. La respuesta (response) contiene el FID (que identificará los mensajes asociados a este fichero), el tamaño del archivo en cuestión, y otros datos del archivo como fechas de modificación, permisos, etc.

- SMB2 READ [Request | Response]: El cliente envía el request sobre un archivo (identificándolo con el FileID) indicando la cantidad de datos que desea leer y el byte desde el que quiere comenzar (offset). El servidor responde con los datos del fichero que se le han pedido.

- SMB2 WRITE [Request | Response]: Similar al SMB2 READ pero utilizado para escribir en lugar de para leer. El cliente envía los datos a escribir, la cantidad y el offset. El servidor responde con la cantidad de bytes que ha escrito.

- SMB2 SET INFO [Request | Response]: Tiene varias opciones para lograr diversos objetivos. En concreto en este trabajo se analizarán las opciones de eliminación y de renombrado. En el

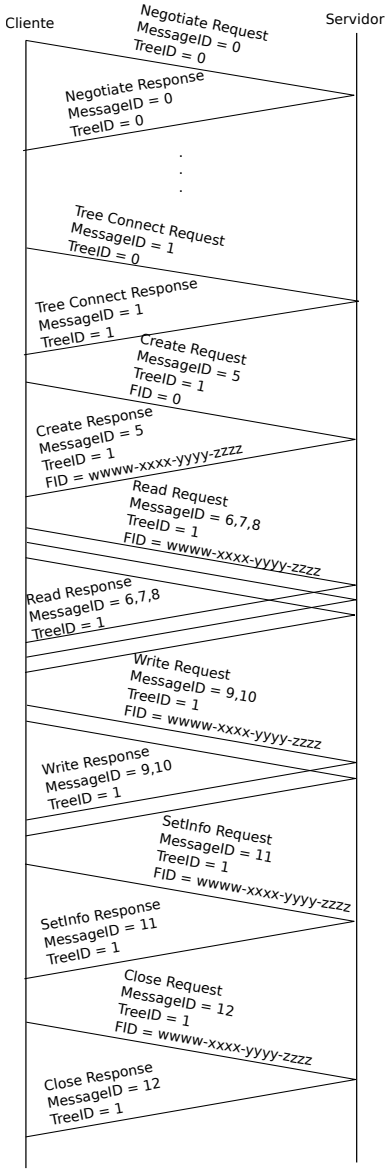



**Figura 2:** Secuencia típica de paquetes SMB2 para la apertura y trabajo con un fichero

caso de la eliminación, el fichero no será eliminado hasta que se cierren todas las instancias abiertas al mismo [55].

- SMB2 CLOSE [Request | Response]: Cierra la instancia del fichero que se ha abierto en el CREATE.

En cada Request el cliente identifica el fichero sobre el que quiere actuar gracias al FileID, y el propio mensaje con el MessageID. El response no tiene FileID, sino que indica el request al que hace referencia con el MessageID.

En la figura 2 se puede ver una secuencia de paquetes típica entre cliente y servidor para abrir un fichero y leer y/o escribir sobre él. Los valores de messageID podrían no coincidir con los de una conexión real, puesto que entre los mensajes de negociación y el TREE_CONNECT podría haber mensajes de inicio de sesión (SESSION_SETUP) que consumirían números en la secuencia de

messageIDs. Se trata de un ejemplo de apertura de fichero, en el que se leen y escriben una serie de datos y a continuación se modifica la información del fichero a través del SET_INFO (renombrándolo por ejemplo).

Para capturar los paquetes de los usuarios y poder efectuar un análisis correcto de los mismos es necesario tener una sonda en la red que pueda capturar los paquetes de todos los usuarios y analizarlos en paralelo. Además, es una condición imprescindible para la implantación en un entorno real que no introduzca retardo en la red. Es por esto que se ha decidido replicar los paquetes en un conmutador y reenviarlos por otro de sus puertos al equipo corriendo el algoritmo que se describe en las siguientes secciones. Esto es lo que se llama un *port mirroring*, y es una práctica bastante utilizada en el análisis de tráfico en red (Ver Figura 3).

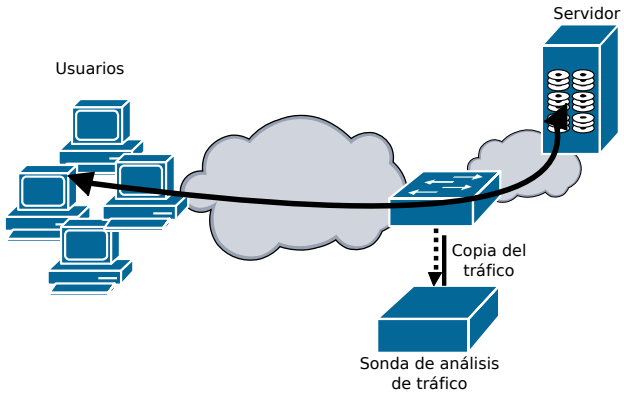



**Figura 3:** Escenario de red LAN con sniffer

## 3.2. Escenario no infectado

Para poder detectar un comportamiento anómalo en el tráfico de la red, es necesario analizar primero el comportamiento habitual de los usuarios en un entorno sin la presencia del malware. Para esto se han analizado trazas de tráfico de la Universidad Pública de Navarra (UPNA) cuyas características pueden consultarse en el Cuadro 2. Es un escenario muy similar al explicado anteriormente (sección 3), donde los usuarios tienen acceso a un servidor central en el que guardan los ficheros. Es un escenario con miles de usuarios, sobre todo trabajadores de oficina, cientos de ellos accediendo simultáneamente a los volúmenes compartidos. El tráfico de protocolos distintos a SMB se eliminó de las trazas.

Para el proceso de establecimiento de los parámetros adecuados del algoritmo (podría entenderse como un entrenamiento) se utilizaron 8 horas de tráfico capturado durante un día de trabajo. A esta traza se le llamará traza *1dUPNA* 2. A continuación, se validaron los resultados obtenidos con 6 trazas, cada una de 24 horas de duración, obtenidas de 6 días consecutivos y laborables en el mismo entorno (trazas *1sUPNA-01* a *1sUPNA-06*). Para validar estos resultados en un entorno distinto se realizó una validación con tráfico de una empresa privada (1 día laborable durante 24h), a cuya traza llamaremos *1dPrivada*. Los resultados de la validación para cada una de las trazas se podrán ver en la sección 6

Podemos ver algunas estadísticas de todas estas trazas en el Cuadro 2. Se han eliminado las conexiones que no contenían comandos READ y WRITE, ya que no hay manipulación de archivos, y en ningún caso puede provocar un falso positivo (ver sección 5).

| | 1dUPNA | 1sUPNA-01 | 1sUPNA-02 | 1sUPNA-03 | 1sUPNA-04 | 1sUPNA-05 | 1sUPNA-06 | 1dPrivada |
|---|---|---|---|---|---|---|---|---|
| **Lugar** | UPNA | UPNA | UPNA | UPNA | UPNA | UPNA | UPNA | Compañía Financiera |
| **Fecha** | 16-01-2017 | 22-02-2017 | 23-02-2017 | 24-02-2017 | 27-02-2017 | 28-02-2017 | 1-03-2017 | 24-04-2015 |
| **Duración** | 8h40min | 24h | 24h | 24h | 24h | 24h | 24h | 24h |
| Tamaño (Gbytes) | 212 | 469 | 352 | 333 | 383 | 355 | 714 | 1100 |
| **Total conexiones SMB** | 46609 | 45414 | 43716 | 34864 | 42593 | 42162 | 43543 | 2717091 |
| Datos para conexiones más largas de 5 minutos que tienen operaciones de lectura y escritura | | | | | | | | |
| **Conexiones SMB** | 401 | 424 | 391 | 375 | 362 | 369 | 386 | 21764 |
| **Client hosts** | 330 | 327 | 320 | 306 | 302 | 306 | 313 | 4882 |
| **Server hosts** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1677 |
| **Duración de conexión (horas)** | 5.93 | 5.43 | 5.70 | 5.60 | 5.97 | 5.86 | 5.86 | 3.5 |
| **Ficheros abiertos por conexión** | 7640 | 9437 | 8589 | 6115 | 12224 | 7717 | 7870 | 1863 |
| **Mbytes leídos por conexión** | 91.4 | 68.6 | 75.6 | 65.3 | 94.9 | 69.8 | 82.5 | 18.4 |
| **Mbytes escritos por conexión** | 294.19 | 295.6 | 288.97 | 237.9 | 315.1 | 307.5 | 306.1 | 5.2 |

**Tabla 2:** Trazas de tráfico para escenarios no infectados

## 3.3. Escenario infectado

Se han obtenido muestras operativas de ransomwares desde el verano de 2016 y se han lanzado en un entorno controlado con varias máquinas virtuales simulando el escenario presentado en secciones anteriores. Las muestras se han descargado desde [4] y [3]. Las muestras del malware las suben usuarios registrados en la página y son analizadas por una serie de antivirus para determinar si se trata de un ransomware o de algún malware de otro tipo. Si se determina que es una muestra de ransomware se etiqueta como tal. Es gracias a esa etiqueta como se han conseguido las muestras para los experimentos, y se han clasificado en diferentes familias en función de los resultados que ofrece la propia página.

Nuestro entorno de pruebas se ha construido en una sola máquina con un Intel Core 2 Duo CPU E6750 con un procesador a 2.66GHz. Se han instalado dos máquinas virtuales (VM) corriendo Windows 7, una de ellas haciendo de cliente y otra haciendo de servidor. El servidor va a montar un directorio compartido, que por defecto utilizará SMB, con una población de archivos que se explicará a continuación. La muestra de ransomware se ejecuta desde el cliente e infectará tanto el propio cliente como los archivos compartidos en el directorio entre cliente y servidor. El tráfico entre las máquinas virtuales se captura en un fichero pcap durante la infección, como si se tratara del elemento capturador de tráfico visto en 3.

En la figura 4 se puede ver esta configuración de las dos máquinas virtuales cliente y servidor conectadas a través de un switch virtual de virtualBox.

El directorio compartido por las dos máquinas es un factor importante a analizar, ya que va a ser su encriptación la que va a generar el tráfico desde el que se extraerá el comportamiento del ransomware. Hay que tener en cuenta y ser conscientes de la necesidad de mantener una copia de dicho directorio en una ubicación que no se vea comprometida por el ransomware, ya que va a ser necesario recuperar el mismo después de cada infección (en nuestro caso el directorio se ha guardado en el host anfitrión). Para las muestras de ransomware antiguas se utilizó un directorio de 1916 archivos, con un tamaño total de 1.88GB y un tamaño de archivo máximo de 25.6MB. Para las muestras más recientes (desde Marzo de 2017) se ha utilizado un directorio mayor, en el

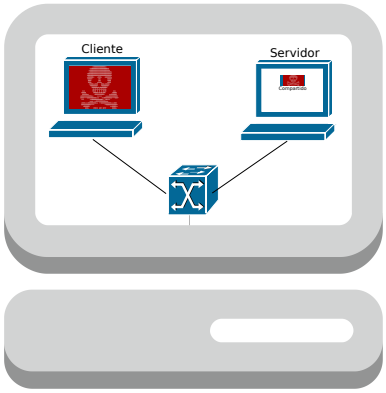



**Figura 4:** Virtualización del escenario en una red compartida (NAS)

que los tamaños de los archivos se han obtenido siguiendo una distribución lognormal con una cola de Pareto. Este modelo de tamaño de fichero y estructura de directorio, está descrito en [23], y se ha obtenido a partir de datos de más de 60000 equipos corriendo Windows. Se han utilizado los parámetros recomendados en ese artículo. La estructura del directorio es aleatoria, imitando un sistema de ficheros real. El resultado del directorio contiene 5138 archivos con un tamaño total de 5.3GB, estos ficheros tienen un tamaño máximo de 838MB y una profundidad de 8 directorios.

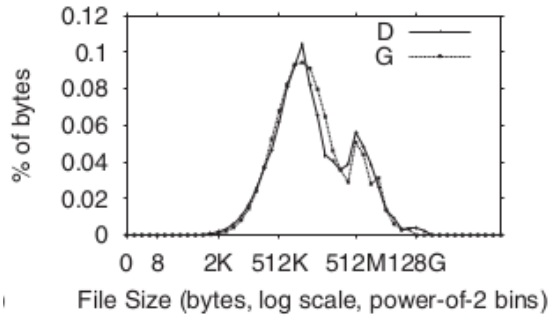



**Figura 5:** Porcentaje de bytes que corresponden a ficheros de un tamaño determinado. Figura 2(d) de [23]

En la Figura 5 se pueden observar las distribuciones lognormales con cola de Pareto que se han utilizado para la generación del directorio. La línea D es la distribución teórica que se desea generar, mientras que la G es la distribución que se consigue recrear. Se debe advertir que el eje vertical es el porcentaje de bytes que pertenecen a archivos del tamaño correspondiente al eje X. Es decir, un archivo de 1GB tiene la misma cantidad de bytes que 1024 archivos de 1MB, consideración que se debe tener en cuenta para entender la gráfica mostrada.

Las 53 muestras recogidas a lo largo de todo este tiempo, se han clasificado en 18 familias distintas (ver Cuadro 3). Cada muestra representa un fichero binario distinto que encripta los archivos de la población expuesta. A pesar de que algunos trabajos como [43, 61] realizan sus experimentos con cientos de muestras, no todas las muestras funcionan correctamente (algunas de las etiquetadas como ransomware no lo son) y no todas las familias encriptan los ficheros a través de la red. Esta es la razón por la que algunas familias famosas como TorrentLocker o Matrix no se encuentran presentes en este trabajo. En relación con [2], tenemos muestras de las familias más importantes y por tanto se considera que esta población de muestras es representativa. Otras familias o muestras presentadas

en otros trabajos más antiguos han quedado obsoletas porque sus servidores C&C han quedado bloqueados y por lo tanto ya no se pueden ejecutar.

| Familia | Versiones | Fecha de aparición | Comportamiento Tipo | Número de muestras |
|---|---|---|---|---|
| VirLock | VirLock | Diciembre 2014 | I | 1 |
| CTBLocker | CTBLocker v4.0 | Enero 2015 | I | 3 |
| Teslacrypt | TeslaCrypt v3.0 | Febrero 2015 | III | 1 |
| TorrentLocker | CryptoFortress | Marzo 2015 | III | 1 |
| DMALocker | DMALocker | Enero 2016 | III | 1 |
| Locky | Locky v1.0, Aesir, Odin, Osiris, Diablo6 | Febrero 2016 | III | 10 |
| Cerber | Cerber v2.0, v4.0, v4.1.6, v5.0, v4.1, Red Cerber | Marzo 2016 | III | 15 |
| CryptXXX | CryptMIC v5.001 | Abril 2016 | II | 1 |
| Bart | Bart v2.0 | Junio 2016 | I | 1 |
| CryptoMix | CryptFile2 | Junio 2016 | I | 10 |
| Crysis | Crysis, Dharma | Noviembre 2016 | III | 1 |
| Sage | Sage v2.0 | Diciembre 2016 | III | 1 |
| MRCR | MRCR1 | Diciembre 2016 | III | 1 |
| Spora | Spora | Enero 2017 | II | 1 |
| WannaCry | WannaCry v2.0 | Febrero 2017 | I | 2 |
| BTCWare | Aleta | Marzo 2017 | III | 1 |
| Jaff | Jaff | Junio 2017 | III | 1 |
| Globe | GlobeImposter v2.0 | Junio 2017 | III | 1 |

**Tabla 3:** Muestras de ransomware

Para el análisis del tráfico entre el host infectado y el servidor, estas más de 50 muestras se han clasificado en 3 grupos en función de su comportamiento. Esta diferenciación es importante, ya que se trata de un detector *behaviour-based*.

- Tipo I - El ransomware lee el fichero original, crea un nuevo fichero con diferente nombre y/o extensión y escribe el contenido encriptado en él. A continuación elimina el fichero original y comienza con el siguiente archivo.
- Tipo II - Muy similar al tipo I pero los datos encriptados son escritos sobre el fichero original, es decir, sobreescribe el fichero antiguo. No le cambia de nombre ni le añade extensión.
- Tipo III - Similar al II pero después de sobreescribir el contenido del fichero original renombra el fichero y le añade una extensión específica de la familia y versión del ransomware.

Esta clasificación es bastante parecida a la que se realiza en [61], aunque se tiene otro punto de vista del comportamiento, puesto que se analiza desde una visión de red, observando los paquetes intercambiados entre cliente y servidor, en lugar de analizar las funciones de acceso al disco.

Después de analizar el tráfico generado por cada una de estas muestras del malware podemos extraer estas características comunes a todos ellos, y que además son inevitables para que el ransomware realice la función para la que está diseñado.

A Debe leer los archivos. Es una característica obligatoria para que pueda encriptar los datos.

B Debe escribir datos. Es imprescindible que escriba en un fichero los datos encriptados. Normalmente escribirá un número de bytes similar al número de bytes leídos.

C Debe eliminar datos. Ya sea sobreescribiendo los originales o eliminando directamente del disco. Esta acción también es imprescindible y es lo que va a provocar que pida una recompensa.

D Las acciones de lectura y escritura de datos estarán cercanas en el tiempo. Incluso pueden llegar a producirse en paralelo. Si el fichero es grande las dos acciones pueden verse algo más distanciadas en tiempo, pero siempre limitado por la cantidad de memoria RAM disponible en el equipo.

E Intentará realizar las operaciones de lectura, escritura y eliminación rápido. Es necesario para que el usuario no se de cuenta de lo que ocurre hasta que ya sea demasiado tarde, y además de esta forma es menos probable que el usuario apague el equipo o entre en funcionamiento un proceso de copia de seguridad.

Estas cinco características se han observado en todas las muestras que se han recogido, y ninguna de ellas nos parece evitable por parte del Crypto-ransomware, siempre que siga teniendo las mismas características que hasta ahora (encriptando y pidiendo una recompensa por los datos), puesto que en caso contrario se trataría de otro tipo de malware y serían necesarias otro tipo de herramientas para su detección.

# 4. Comparación de comportamientos

Hasta ahora se ha visto cómo se han obtenido las muestras de trazas en escenarios infectados y no infectados, así como las distintas familias de ransomware con las que se van a realizar los experimentos, su clasificación en función de su comportamiento y una serie de características ineludibles para un funcionamiento propio del malware. Dicho esto, es importante comparar los funcionamientos habituales de usuarios en un escenario sin presencia del ransomware, con el comportamiento provocado por la presencia de un ransomware en la red.

En primer lugar, vamos a comprobar que los ransomware tienen las cinco características comentadas en la sección 3.3. Después analizaremos esas mismas características en las trazas *limpias* y por último propondremos una serie de aspectos en los que centrarnos para poder diferenciar los dos comportamientos.

## 4.1. Características de los ransomwares

Las cinco características mencionadas anteriormente pueden verse en la Figura 6. Dicha figura representa los bytes leídos (morado), los bytes escritos (verde), y las eliminaciones (azul) frente al tiempo. Los bytes leídos son ligeramente inferiores a la cantidad de bytes escritos, debido sobre todo a que además de los archivos encriptados, el ransomware va a escribir algunos archivos más en el directorio para indicar instrucciones que debe seguir el usuario para desencriptar los archivos [54]. Podemos apreciar cómo aumenta de forma rápida la cantidad de bytes leídos y escritos, y la cantidad de eliminaciones que se producen en un corto intervalo de tiempo.

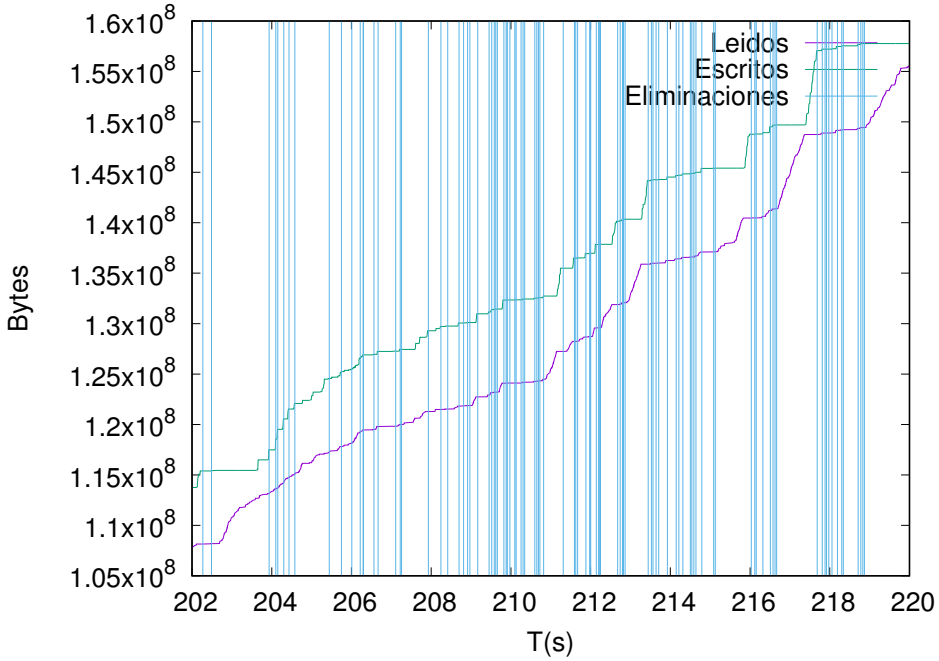



**Figura 6:** Comportamiento de ransomwares

En la Figura 6 se pueden ver tan solo 18 segundos del proceso de encriptación, que duró varios minutos. Se ha escogido un periodo breve de forma que se pudieran apreciar bien las lecturas y escrituras, así como las eliminaciones de los ficheros. Aunque se ha representado un tipo concreto de ransomware, todos ellos se comportan igual y no se aprecian grandes diferencias en estas gráficas

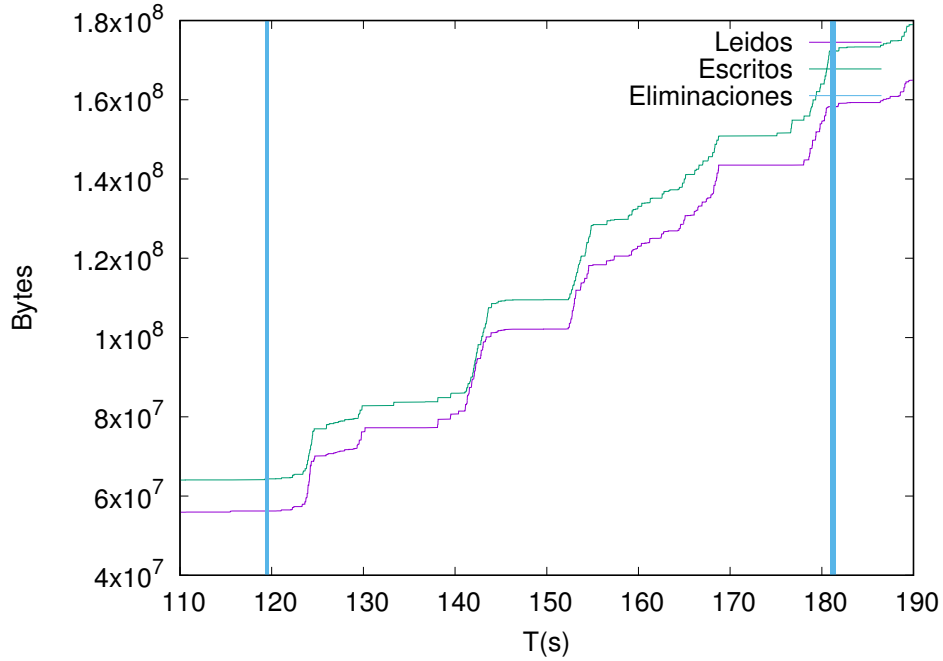



**Figura 7:** Comportamiento WannaCry

para cada uno de ellos, salvo la excepción del "WannaCry" que se comentará a continuación. Sí que se pueden apreciar diferencias en la cantidad de bytes escritos (algunos de ellos encriptan solo partes del archivo) o en la velocidad a la que los encriptan (depende del algoritmo que empleen para ello), pero en general es el mismo método para todos.

"WannaCry" es el único que presenta algunas diferencias significativas en su comportamiento, leyendo y escribiendo una gran cantidad de archivos durante aproximadamente un minuto, para luego eliminar los archivos originales sin producirse lecturas y escrituras entre los borrados. Puede verse esto en la Figura 7. Con este comportamiento el ransomware ya no tiene lecturas y escrituras entre las eliminaciones, algo que era característico de las otras muestras. El comportamiento parece más similar al de un usuario que trabaja en un directorio (leyendo y escribiendo) y luego realiza múltiples borrados de archivos que ya no le son útiles. A pesar de esto y como se verá en la siguiente sección, la cantidad de bytes que lee un ransomware y la cantidad de ficheros que elimina va a determinar con una alta tasa de acierto si se trata o no de un comportamiento maligno.

Cada una de las "ráfagas" de eliminaciones, se produce en un intervalo muy corto de tiempo (en el caso del "WannaCry), que en la figura se aprecia como una única línea más gruesa. Se ve también que los bytes leídos y escritos son más similares que en el caso anterior. De cualquier forma, las cinco características mencionadas en la Sección 3.3 se siguen cumpliendo.

## 4.2. Características del comportamiento de usuario

Visto el tráfico habitual de un usuario infectado por un ransomware, es necesario establecer una serie de parámetros que permitan diferenciar este tráfico del de un usuario normal no infectado. Para ello, se dispone de una serie de muestras de comportamientos de usuarios en un entorno real (ver Cuadro 2) que van a permitir comparar estos dos tipos de tráfico.

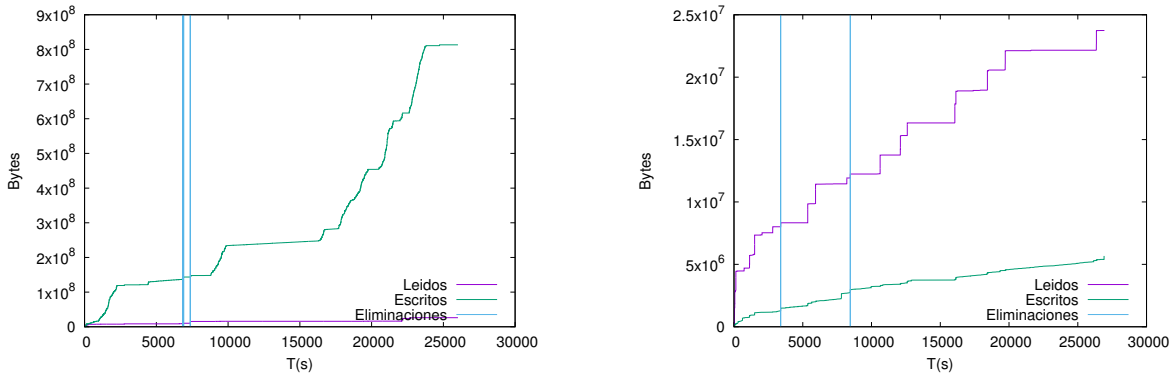
Para esta comparación, se escogen dos trazas de usuarios de un tiempo significativo (entre 7 y 8 horas) que se considera una duración habitual en una jornada laboral. En el Cuadro 4 se pueden

ver las características de las trazas de estos dos usuarios.

| | Usuario 1 | Usuario 2 |
|---|---|---|
| **Duración** | 7.23 horas | 7.47 horas |
| **Ficheros abiertos** | 6594 | 15659 |
| **MBytes leídos** | 30.5 | 29.42 |
| **MBytes Escritos** | 891,7 | 6.01 |
| **Ficheros Eliminados** | 6 | 4 |

**Tabla 4:** Características de las trazas de usuarios

A continuación se presenta una gráfica similar a la presentada para el comportamiento del ransomware (Figura 8), en este caso podemos ver que se producen muchas menos eliminaciones, además de apreciarse la diferencia entre los bytes leídos y escritos, que es mayor que en el caso de infección.

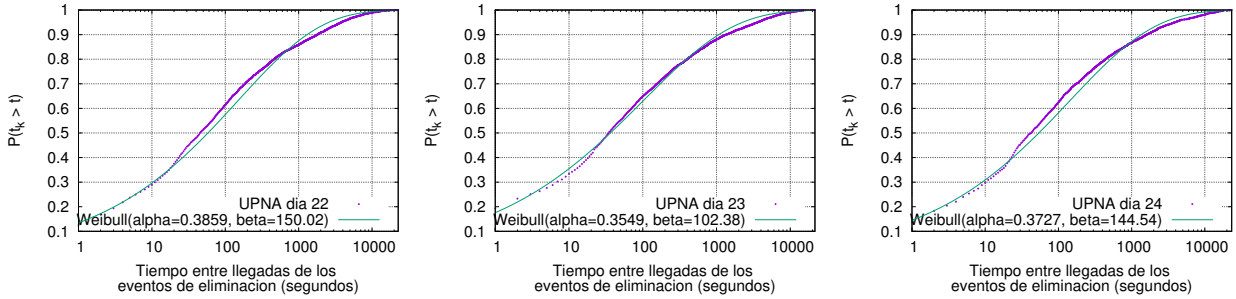



(a) Usuario 1 (b) Usuario 2

**Figura 8:** Bytes leídos, escritos y ficheros eliminados en función del tiempo para usuarios no infectados

En el primer caso (Figura 8a), se eliminan 6 ficheros en las 8 horas de duración de la traza, mientras que en el segundo (Figura 8b) se eliminan 4 ficheros, agrupados de dos en dos y separados más de una hora. En el usuario 1 las eliminaciones se encuentran en dos grupos, separadas varios minutos. La diferencia entre los bytes que leen y que escriben los dos usuarios es bastante grande, sobre todo en el caso del primero, aunque en ambos es mayor que en el caso de los ransomwares.

Aunque se trata de dos ejemplos concretos de usuarios, podemos ver algunas diferencias claras. Para ofrecer una visión macroscópica con todos los usuarios de la traza se ha optado por estimar la distribución de probabilidad del tiempo entre eliminaciones de ficheros. Es decir, se ha buscado un modelo para la variable aleatoria de tiempo entre dos eventos de eliminación, suponiendo estos tiempos independientes e idénticamente distribuidos. Se comparará así de forma más objetiva las diferencias en el comportamiento entre el ransomware y el usuario habitual. Se han cogido las trazas de 6 días presentadas anteriormente (ver Cuadro 2) y se han dibujado las funciones de probabilidad como se ve en la figura 9



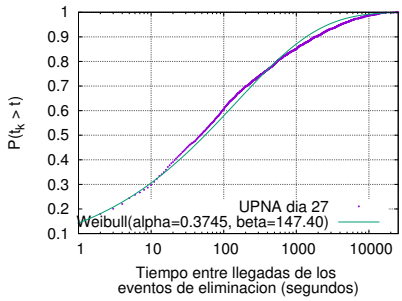

(a) Día22



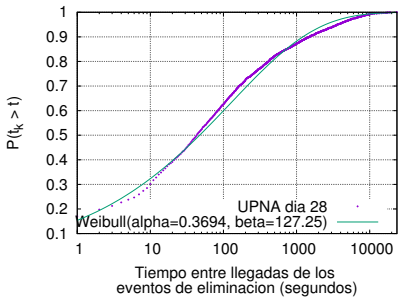

(b) Día 23



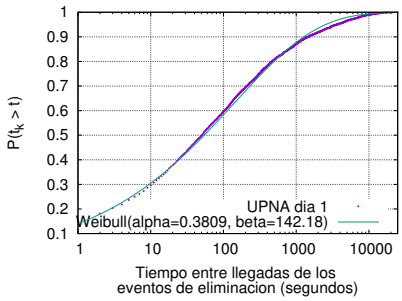

(c) Día 24




(d) Día 27




(e) Día 28




(f) Día 1

**Figura 9:** Tiempos entre eliminaciones para trazas de usuarios no infectados

Vemos la gran similitud entre los distintos días en los que se recogieron las muestras de tráfico. Menos del 20 % de los eventos de eliminación se producen menos de un segundo después del anterior en todos los días estudiados. La distribución se aproxima a una distribución Weibull de probabilidad con los parámetros que se indican en la figura para cada uno de los días. La aproximación de los datos a la distribución mencionada, se ha realizado en varios pasos.

En primer lugar se linealiza la expresión matemática de la función de probabilidad acumulada. Quedando de la siguiente manera:

$$\ln\left(-\ln\left(1-y\right)\right) = \alpha \ln x - \alpha \ln \beta \tag{1}$$

A continuación, se puede observar que se dispone de un menor número de muestras para valores pequeños del eje de tiempos (eje X) (ver Figura 10). Para una correcta aproximación a lo largo de todo el eje X, se ha procedido a muestrear los datos experimentales, obteniendo puntos equiespaciados en el eje para hacer una aproximación con precisión uniforme en todo el eje X. En la Figura 11 se pueden ver representadas las muestras experimentales linealizadas y las muestras escogidas para la aproximación tras del muestreo.

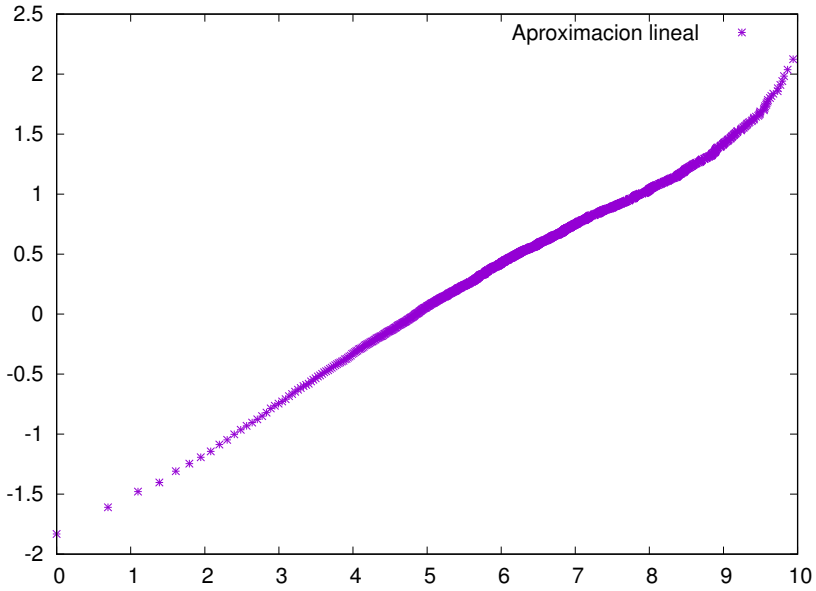



**Figura 10:** Muestras experimentales linealizadas

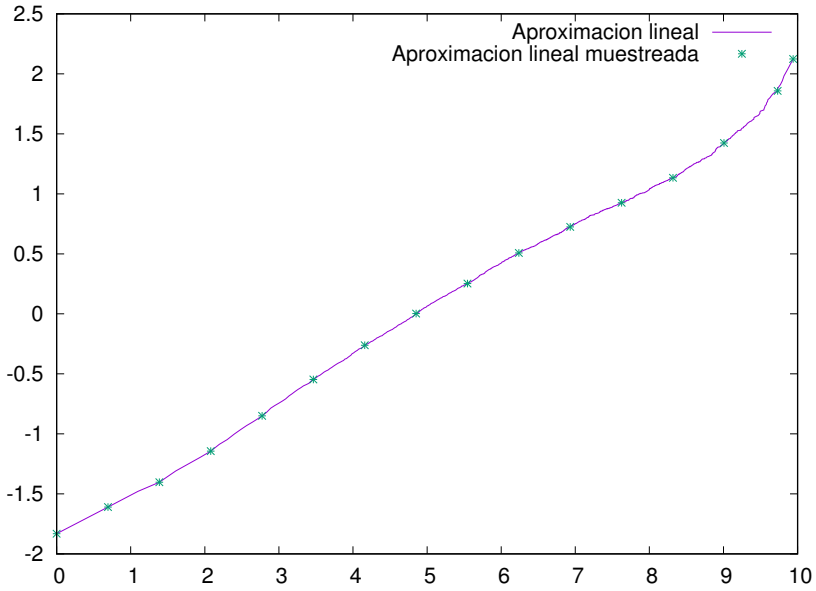



**Figura 11:** Comparación entre las muestras linealizadas y las mismas muestreadas

A continuación ya podemos aproximar esta recta por mínimos cuadrados siguiendo las siguientes ecuaciones:

$$m = \frac{NS_{xy} - S_x S_y}{NS_{xx} - S_x S_x} \tag{2}$$

$$n = \frac{S_{xx} S_y - S_x S_{xy}}{NS_{xx} - S_x S_x} \tag{3}$$

$$S_x = \sum_{i=1}^{N} x_i \tag{4}$$

$$S_y = \sum_{i=1}^{N} y_i \tag{5}$$

$$S_{xx} = \sum_{i=1}^{N} x_i^2 \tag{6}$$

$$S_{xy} = \sum_{i=1}^{N} x_i y_i \tag{7}$$

Con ellas se calculan la pendiente y la ordenada aproximadas de la recta representada por las muestras experimentales. Partiendo de la ecuación 1 y comparando con la función característica de una recta, se obtiene la relación entre los parámetros m y n de la recta con los de la distribución Weibull que buscamos.

$$\alpha = m \tag{8}$$

$$\beta = e^{n/-m} \tag{9}$$

Así se obtienen los parámetros característicos de una distribución Weibull. Repitiendo el proceso para cada uno de los días se obtienen las gráficas de las aproximaciones de la Figura 9.

Esta aproximación no se cumple para el caso del ransomware, cuya función de probabilidad es totalmente diferente, como se aprecia en la figura 12. En este caso, el ransomware elimina los ficheros a mayor velocidad (como se había visto en el análisis del comportamiento (ver sección 4.1). Los eventos de borrado se separan entre ellos menos de un segundo en el 97 % de los casos, frente al 20 % en el caso de los usuarios. El 100 % de las eliminaciones se demoran entre ellas como máximo 100 segundos, aunque el 99.9 % de las veces el tiempo entre borrados es inferior a 10 segundos.Hay que tener en cuenta el comportamiento del ransomware "WannaCry", que hace las eliminaciones a ráfagas, habiendo veces que entre dos de ellas hay alrededor de 60 segundos, lo cual explicaría ese 0.1 % de veces sobre el total de las eliminaciones que se producen en tiempos mayores de 10 segundos.

Con estos datos podemos considerar que el tiempo entre eliminaciones (o lo que es lo mismo, la velocidad de eliminación de ficheros) es una característica que nos va a permitir discernir entre comportamientos normales e infectados con una eficacia considerable. Aún así, habrá determinados comportamientos normales en los que las eliminaciones también se produzcan a gran velocidad. Es por esto por lo que es necesario añadir más características que nos permitan diferenciar entre esos casos de eliminaciones rápidas en los usuarios.

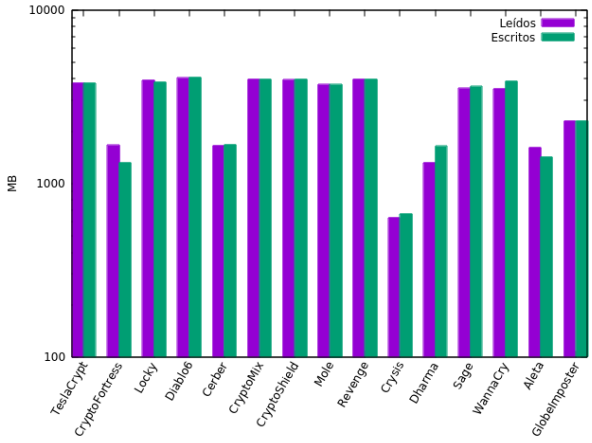



**Figura 13:** Bytes leídos y escritos por algunos de los ransomwares

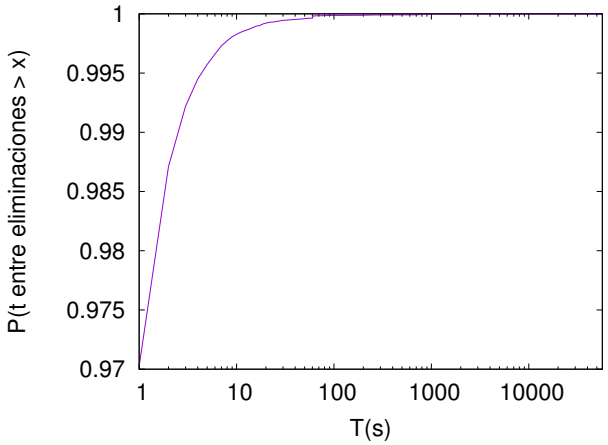



**Figura 12:** Distribución de probabilidad del tiempo entre borrados para el caso del ransomware

Otra característica representativa de los ransomwares es la similitud entre la cantidad de bytes que se leen y escriben. El comportamientos del usuario no infectado difiere, puesto que muchas aplicaciones leen gran cantidad de bytes pero escriben pocos, o viceversa. En la figura 13 podemos ver algunas de las clases de ransomware con la cantidad de bytes que han sido leídos y escritos en nuestro directorio de prueba. Como se ha mencionado en la sección 3.3, el directorio que se ha encriptado es de 5.3GB, sin embargo, en la figura 13 podemos ver que algunas familias de ransomware (Crysis por ejemplo) no leen y escriben 5GB. Esto es porque algunos ransomwares no encriptan todo el archivo, sino que lo hacen solo en la primera parte de los archivos, de forma que consiguen mayor velocidad y un impacto similar, porque el archivo queda inutilizable. De cara a nuestro algoritmo, que como se verá en la sección siguiente se basa en la velocidad de las lecturas y las escrituras, esto no afecta de forma significativa. Aunque hay menos cantidad de bytes para calcular la velocidad, el tiempo es menor, y por lo tanto el resultado no se ve muy alterado respecto a otros tipos de ransomware. Como se verá en la sección 6.1, los resultados para esta familia en concreto son igual de buenos que para otras, por lo que podemos concluir que no nos vemos afectados por este comportamiento particular.

En el caso de la velocidad de lectura y escritura, los ransomwares que se muestran en la figura 8

leen y escriben a una velocidad en torno a 20Mbps (velocidad media) , mientras que los usuarios lo hacen a una velocidad de 244Kbps (el usuario 1 escribiendo). Este parámetro, aunque representativo, no es muy fiable, puesto que se ha calculado la velocidad media a lo largo de toda la conexión, a pesar de que podrían producirse pequeños intervalos temporales donde el usuario escribiera y leyera a más velocidad, pareciéndose más al comportamiento tipo de un ransomware. De no ser así, podríamos establecer un umbral de velocidad como detector de forma muy sencilla. Es complicado mostrar de distinta forma la velocidad de lectura y escritura, puesto que su medición exige establecer ventanas temporales, y en función de estas pueden salir valores más o menos parecidos al de los ransomwares. Es por esto que se ha optado por hacer la media a lo largo de toda la conexión, advirtiendo del hecho de su poca representatividad, aunque no por ello es menos cierto que esta velocidad es una característica muy destacable de este tipo de malware.

Así pues, tenemos varios aspectos en los que podemos diferenciar a un ransomware de un usuario normal. En primer lugar la cantidad de eliminaciones que realiza un ransomware es mucho mayor que las de un usuario y mucho más rápida. Por otro lado la velocidad a la que lee y escribe un ransomware también es considerablemente mayor, y por último tenemos el parecido entre los bytes leídos y escritos por el ransomware, mientras que difieren bastante en el caso de los usuarios. En la siguiente sección veremos cómo implementar el algoritmo para diferenciar estos aspectos minimizando los ficheros que se terminan encriptados y el número de falsos positivos.

# 5. Algoritmo de detección

Para el desarrollo de este algoritmo de detección se ha considerado necesario no depender del tipo de algoritmo criptográfico que utilice un ransomware en cuestión puesto que puede cambiar de una familia a otra y hacer la herramienta inservible para algunas variantes del malware. Tampoco se tiene en cuenta la posibilidad de una entropía elevada entre el contenido del fichero cuando es leído y cuando es escrito por el ransomware, debido principalmente a dos razones: a) si el fichero original está comprimido, la entropía puede ser pequeña entre la versión original y la versión encriptada y b) el procesado de la entropía para todos los ficheros que se ven en el tráfico en red de todos los usuario incrementaría la carga en la CPU del equipo de monitorización, aumentando su requisitos hardware para una misma tasa de tráfico. Esta medición de la entropía entre ficheros es una de las características más utilizada en las soluciones existentes para la detección de ransomwares, aunque en todos los casos se trata de herramientas instaladas en local, que comparan la entropía gracias a los datos extraídos de las llamadas a lecturas y escrituras del propio sistema [33, 43, 63, 64, 46]. Hay que tener en cuenta que en este caso, el mismo dispositivo analizador del tráfico captura y procesa los paquetes de todos los usuarios de la red. La presente propuesta no lleva a cabo cálculos de entropía y no necesita recoger ni procesar el contenido de los ficheros que son transportados por SMB, ahorrando carga computacional en el equipo.

Como se ha visto en la sección 4.1, todos los ransomwares van a leer el contenido de los ficheros y lo van a escribir encriptado en un fichero diferente (antes de borrar el original) o sobre el mismo archivo, provocando la pérdida de información del usuario en cualquiera de los dos casos. Además, en base a la comparativa hecha en la misma sección, se han determinado las características que permiten diferenciar a los ransomwares de los usuarios, y partiendo de esto se ha construído el algoritmo pensando en minimizar el número de falsos positivos y el número de ficheros encriptados por el ransomware antes de su detección.

A partir de ahora y a lo largo de todo el documento, se denominará evento de borrado a cualquiera de las dos acciones mediante las cuales un ransomware puede hacer perder información al usuario: la eliminación de un fichero, la sobreescritura de datos de ficheros y el truncado del contenido de un fichero. Para detectar el ransomware se va a establecer un umbral mínimo de ficheros perdidos (parámetro $N$). Se debe establecer este mínimo de eventos de borrado teniendo en cuenta que cuanto menor sea, menos información perderá el usuario (aunque como se verá en la sección 7.3, los N ficheros perdidos se podrán recuperar una vez el ransomware sea detectado). A su vez, cuanto menor sea $N$ más probable será que un usuario realice este número de eliminaciones en su trabajo habitual y por lo tanto aumentará la probabilidad de que se produzca un falso positivo. Este compromiso entre la detección rápida y la probabilidad de falsos positivos es lo que ocupará gran parte de esta sección. Además, los $N$ eventos de borrado deben producirse en un máximo intervalo de tiempo $T$. Como se explicará a continuación en más profundidad, cuanto mayor sea este tiempo $T$ en el que deben producirse los $N$ eventos, más probabilidades habrá de detectar el ransomware, aunque habrá también mayor probabilidad de que se produzcan falsos positivos.

En las siguientes secciones, se analizará en detalle el funcionamiento y los parámetros del algoritmo REDFISH, y se calcularán $N$ y $T$ para conseguir un 100 % de detección en las muestras de ransomware presentadas en la tabla 3

## 5.1. REDFISH

De forma general, se define $\{\rho_i\}, i \in \mathbb{N}$ como el proceso aleatorio de llegadas de tiempo-discreto para los eventos de lectura de ficheros; $\{\omega_i\}$ es un proceso similar para las operaciones de escritura, y $\{\tau_i\}$ para los eventos de eliminación de un fichero. Todas estas variables hacen referencia al instante

de tiempo en que se producen estos eventos. El evento de eliminación se produce en el instante en el que el fichero se cierra con un CLOSE después de haberse producido un comando de eliminación (SET INFO), un truncado del fichero (opción en el CREATE) o una sobreescritura de datos en el fichero en cuestión (en el WRITE). Dos eventos no pueden producirse en el mismo instante exacto de tiempo desde el punto de vista de la sonda ya que el tráfico de sus mensajes viene serializado por el interfaz de captura.

REDFISH procesa los eventos mencionados de forma secuencial. Para el caso de los eventos de borrado, se define también una variable de tiempo entre llegadas $\{t_k\}$ donde $t_k = \tau_{k+1} - \tau_k k \in \mathbb{N}$. Esta definición se utilizará para el modelo analítico en próximas secciones.

Para cada evento de lectura, escritura o eliminación se asocia un directorio determinado donde se encuentra este fichero. Este directorio será extraído del mensaje que pide la apertura de ese fichero enviado por el cliente hacia el servidor (SMB CREATE Request). Por el momento solo se almacena la ruta hasta el fichero, ignorando su nombre y su extensión, aunque más adelante serán utilizadas para diferenciar posibles comportamientos de usuario de otros comportamientos infectados. Se denomina $p_{\rho i}$ a la ruta del archivo donde se ha producido la operación de lectura en el instante $\rho_i$. De forma similar, se define $p_{\omega i}$ y $p_{\tau i}$. Q se llamará al conjunto de posibles rutas dentro de nuestro sistema de ficheros.

A continuación se establecen $b_{\rho i}$ y $b_{\omega i}$ como la cantidad de bytes que se han leído y escrito respectivamente en cada operación de lectura y escritura. Para cualquier ruta $p$ del sistema se define $P_p(t)$ (ecuación 10) como la cantidad acumulada de bytes leídos en los ficheros contenidos en ese directorio $p$ (pero no en subdirectorios del mismo). De la misma manera se llamará $\Omega_p(t)$ (ecuación 11) a los bytes escritos.

$$P_p(t) = \sum_i b_{\rho_i}, \forall i \in \mathbb{N} \mid \rho_i \leq t \text{ and } p_{\rho_i} = p \qquad P_p(0) = 0, \forall p \in Q \tag{10}$$

$$\Omega_p(t) = \sum_i b_{\omega_i}, \forall i \in \mathbb{N} \mid \omega_i \leq t \text{ and } p_{\omega_i} = p \qquad \Omega_p(0) = 0, \forall p \in Q \tag{11}$$

Como se ha mencionado y comprobado en secciones anteriores (ver sección 4), la cantidad de bytes leídos y escritos por el ransomware no es igual, pero sí es similar. Además, como el malware no cambia la ubicación del archivo encriptado respecto al original (para que el usuario sea consciente de la encriptación de los archivos), la cantidad de bytes leídos y escritos por el ransomware en un mismo directorio debe ser también similar. Para reducir el ruido causado por el comportamiento normal de usuario, como por ejemplo una gran cantidad de bytes leídos por alguna aplicación, se define $m_p(t)$ como el mínimo entre la cantidad de bytes leídos y escritos en un directorio $p$ (ecuación 12). Las aplicaciones benignas que solo lean o escriban en ficheros del directorio, no incrementarán $m_p(t)$, mientras que el ransomware lo incrementará mucho cuando esté encriptando los archivos.

$$m_p(t) = \text{mín}\{P_p(t), \Omega_p(t)\} \tag{12}$$

REDFISH hará saltar la alarma de infección cuando se produzca un evento de borrado y además, se cumplan estas dos condiciones:

- Para el k-ésimo fichero eliminado ($k \geq N$) el intervalo de tiempo que comprende a los últimos N eventos de borrado, $\Delta_N \tau_k$ (ecuación 13), debe ser menor o igual que T ($\Delta_N \tau_k \leq T$).

$$\Delta_N \tau_k = \tau_k - \tau_{k-N+1} = \sum_{i=k-N+1}^{k-1} t_i \tag{13}$$

- Para el k-ésimo fichero eliminado ($k \geq N$), se denomina V[k] (ecuación 14) a la velocidad media de las operaciones de lectura y escritura en los directorios donde esos ficheros $k$ fueron eliminados. El conjunto de esos directorios de esos eventos es $D = \{p_{\tau_i}\}, \forall i \in \mathbb{N} \mid k - N + 1 \leq i \leq k$. La cantidad de bytes para el cálculo de la velocidad se obtiene del incremento de $m_p(t)$ para esos paths. El intervalo de tiempo en el que tuvo lugar esa actividad va desde el primer incremento de $m_p(t)$ en cualquiera de los paths hasta el último evento de eliminación $\tau_k$. Se llamará $\tau^*$ a ese momento donde tuvo lugar ese primer incremento. $\tau^*$ es el valor mínimo entre los valores de tiempo $\eta_i$ cuando el correspondiente $m_{p_{\tau_i}}(t)$ se incrementó, para cualquiera de los paths en el conjunto $D$ (ecuación 15.

  Si $V[k] > V_{thres}$ entonces la alarma saltará en $\tau_k$, por lo que $V_thres$ se convierte en uno más de los parámetros fundamentales del algoritmo, y la condición $V[k] > V_{thres}$ en la segunda condición que debe cumplirse cuando se produce un evento de borrado para que la alarma se produzca.

$$V[k] = \frac{\sum_{i=k-N+1}^{K}(m_{p_{\tau_i}}(\tau_k) - m_{p_{\tau_i}}(\tau^*))}{\tau_k - \tau^*} \tag{14}$$

$$\tau^* = \min_{k-N+1 \leq i \leq k}\{\eta_i\}, k - N + 1 \leq i \leq k \tag{15}$$

Se define $\eta_i$ como se ve en la ecuación 16. Para cada evento de eliminación $\tau_i$, el correspondiente $\eta_i$ es el primer timestamp en el que se incrementó $m_{p_{\tau_i}}(t)$ desde el valor que tenía en la eliminación anterior $\tau_j$ en el mismo path.

$$\begin{gathered} \eta_i \in \mathbb{R} \mid \eta_i < \tau_i \\ p_{\tau_j} = p_{\tau_i} \\ m_{p_{\tau_j}}(\tau_j) < m_{p_{\tau_i}}(\eta_i) \\ m_{p_{\tau_i}}(t) = m_{p_{\tau_i}}(\tau_j), \forall \tau_j \leq t < \eta_i \end{gathered} \tag{16}$$

Con estas dos condiciones se trata de diferenciar por un lado el comportamiento de usuario en el que las eliminaciones se producen en intervalos de tiempo amplios (la alarma no saltará si $\Delta_N \tau_k > T$), y por el otro se subordina la alarma a una lectura y escritura rápida ($V[k] > V_{thres}$) en el mismo directorio (depende esta velocidad del incremento de $m_p(t)$).

Con esto, REDFISH necesita 3 parámetros de entrada que controlarán la tolerancia del algoritmo frente a determinados comportamientos.

- T: Es el intervalo de tiempo que debe contener al menos N eliminaciones para que la alarma pueda saltar. Valores altos de T provocarán agrupar eliminaciones que se han producido en ventanas de tiempo mayores. En el caso de los usuarios, esto puede provocar que se produzcan falsos positivos a pesar de que, como hemos visto en la imagen 8a, las eliminaciones de los usuarios se producen generalmente alejadas en el tiempo. Por el contrario, tendrá efectos positivos en la detección del ransomware, puesto que permitirá detectarlo aunque haya una serie de ficheros de gran tamaño. Un conjunto de archivos pesados provocarían que el ransomware tarde más tiempo en encriptar el contenido, alejando los eventos de borrado de los ficheros en cuestión. En caso de aumentar T, podríamos detectar este tipo de comportamientos.

  Minimizarlo, a su vez, reduce la posibilidad de encontrar falsos positivos, pero también aumenta la de no detectar el ransomware si los ficheros son grandes y las eliminaciones se separan.

- N: Es el mínimo número de eliminaciones que deben tener lugar en T segundos para que pueda saltar la alarma. Esta no saltará nunca antes de haberse eliminado al menos N ficheros. Por esta razón, N se desea lo más pequeño posible. Sin embargo, valores muy bajos de este parámetro provocarían que saltara la alarma si el usuario elimina uno o dos ficheros. A pesar de esta limitación, para que salte la alarma, además de la eliminación de ficheros, el usuario debe leer y escribir datos en el mismo directorio y a una velocidad considerable (marcada por $V_{thres}$). Esto permite bajar el parámetro N y seguir detectando los ransomwares mientras no se producen falsos positivos en el escenario (ver sección 5.4).
- $V_{thres}$ : Es la mínima velocidad media de lecturas y escrituras asociadas a los N eventos de eliminación para saltar la alarma. Este parámetro es necesario porque los usuarios pueden borrar una gran cantidad de ficheros de un determinado directorio, sin ello significar que un ransomware haya infectado el equipo. Este parámetro permite diferenciar una acción de este tipo, requiriendo que además de eliminar ficheros, se produzcan en el directorio una gran cantidad de lecturas y escrituras. Para la selección de este valor, se han realizado una serie de pruebas, y se ha escogido el valor que provocaba el mínimo número de falsos positivos, para una detección del 100 % de las muestras de ransomware presentadas.

## 5.2. Pseudocódigo

En esta sección se presentará el pseudocódigo de REDFISH para explicar a continuación cada uno de los pasos que se siguen para la detección del ransomware (ver algoritmo 1). Lo primero que hay que destacar es que los paquetes van llegando de forma secuencial al equipo que corre esta herramienta, y por lo tanto el algoritmo consistirá en un bucle en el que permanecerá mientras lleguen más paquetes.

Para cada uno de estos paquetes se mirará el tipo de comando SMB y en función de este se realizarán una serie de funciones. Hay que destacar que se tienen dos variables *flujos* y *paths* que irán creciendo conforme aumenten el número de ficheros abiertos simultáneamente y de directorios utilizados respectivamente. La eliminación de un fichero se produce cuando se sobreescriben datos o cuando se produce un comando de eliminación, pero siempre se procesa el evento cuando se recibe el *CLOSE_RESPONSE* asociado al fichero.

**Algorithm 1** Pseudocódigo de REDFISH

```
1: global Map flujos = new Map()
2: global Map paths = new Map()
3: global List listaEliminaciones = new List()
4: global Int N                                          ▷ Numero de eliminaciones
5: global Int T                                          ▷ Tiempo de la ventana (ms)
6: global Int V                                  ▷ Velocidad umbral para saltar la alarma
7: while nuevoPaquete do
8:     switch nuevoPaquete.tipo do
9:         case CREATE_REQUEST
10:            createRequest = new CreateRequest()
11:            flujos.create(createRequest.fileID, createRequest)  ▷ Nueva entrada en la tabla con ese FileID
12:        case CREATE_RESPONSE
13:            createResponse = new CreateResponse()                ▷ Objeto del tipo CREATE_RESPONSE
14:            if !paths.contains(createResponse.path) then  ▷ Compruebo que ese path NO este en la tabla
15:                paths.create(CreateResponse.path)
16:            end if
17:            flujos.add(createResponse.fileID, createResponse)  ▷ Añado el mensaje al flujo con su FileID
18:        case READ_RESPONSE
19:            readResponse = new ReadResponse()  ▷ Objeto del tipo READ_RESPONSE
20:            if flujos.contains(readResponse.fileID) AND readResponse.success then
21:                path = paths.get(readResponse.path)
22:                path.bytesLeidos+= readResponse.length  ▷ Actualizo los bytes leidos y el mínimo
```

23: **if** *path.minimo* == 0 AND *path.bytesEscritos*! = 0 AND *path.bytesLeidos*! = 0 **then**
24: path.tprimerCambioMinimo = readResponse.time ▷ Primer aumento de $m_p(t)$ ($\eta_i$)
25: **end if**
26: path.minimo=min(path.bytesLeidos, path.bytesEscritos)
27: path.tlastAccess = readResponse.time
28: flujos.add(readResponse.fileID, readResponse) ▷ Añado el mensaje al flujo con su FileID
29: **end if**
30: **case** $WRITE\_RESPONSE$
31: writeResponse = new WriteResponse() ▷ Objeto del tipo WRITE_RESPONSE
32: **if** flujos.contains(writeResponse.fileID) AND writeResponse.success **then**
33: path = paths.get(writeResponse.path)
34: path.bytesEscritos+= writeResponse.length ▷ Actualizo los bytes escritos y el mínimo
35: **if** *path.minimo* == 0 AND *path.bytesEscritos*! = 0 AND *path.bytesLeidos*! = 0 **then**
36: path.tprimerCambioMinimo = writeResponse.time ▷ Primer aumento de $m_p(t)$ ($\eta_i$)
37: **end if**
38: createResponse = flujos.get(writeResponse.FileID).get(1)
39: **if** writeResponse.offset < createResponse.endOfFile **then** ▷ Fichero sobreescrito en parte
40: flujos.get(writeResponse.fileID).eliminado = true
41: **end if**
42: path.minimo=min(path.bytesLeidos, path.bytesEscritos)
43: path.tlastAccess = writeResponse.time
44: **end if**
45: flujos.add(writeResponse.fileID, writeResponse) ▷ Añado el mensaje al flujo con su FileID
46: **case** $SETINFO\_RESPONSE$
47: setInfo = new setInfo() ▷ Objeto del tipo SETINFO_RESPONSE
48: path = paths.get(setInfo.path)
49: **if** flujos.contains(setInfo.fileID **then**
50: **if** $setInfo.cmd$ == $eliminacion$ AND $setInfo.success$ AND *path.minimo*! = 0 **then**
51: flujos.get(setInfo.fileID).eliminado = true ▷ Flujo marcado eliminado
52: **end if**
53: **end if**
54: flujos.add(setInfo.fileID, setInfo) ▷ Añado el mensaje al flujo con su FileID

```
55:         case CLOSE_RESPONSE
56:             closeResponse = new closeResponse() ▷ Objeto del tipo CLOSE_RESPONSE
57:             path = paths.get(closeResponse.path)
58:             if flujos.contains(closeResponse.fileID then
59:                 if flujos.get(closeResponse.fileID).eliminado then
60:                     nuevaEliminacion(closeResponse.path)
61:                 end if
62:             end if
63:             flujos.delete(closeResoponse.fileID)  ▷ Añado el mensaje al flujo con su FileID
64: end while
```

A continuación se detalla la implementación de la función *nuevaEliminacion*, que es fundamental para entender el funcionamiento del algoritmo. En ella se construye un nuevo evento de borrado, en el que se van a guardar todas las variables que se han visto en secciones anteriores (ver sección 5.1). La acumulación de estos eventos, guardados en un vector, es lo que va a determinar si se considera o no un comportamiento sospechoso, ya que representan el número de eliminaciones que se han producido en $T$.

**Algorithm 2** Función de nueva eliminación REDFISH

```
 1: function NUEVAELIMINACION(path)
 2:     eliminacion = new Eliminacion()
 3:     eliminacion.path = path
 4:     eliminacion.firstTIncrMinimo = paths.get(path).tprimerCambioMinimo
 5:     eliminacion.min = paths.get(path).minimo
 6:     eliminacion.time = closeResponse.time
 7:     paths.get(path).minimo = 0                    ▷ Reseteo el valor del minimo en el path
 8:     listaEliminaciones.add(eliminacion)
 9:     primerTIncrMinimoGlob = min(listaEliminaciones.get(all).tprimerCambioMinimo)
10:     if eliminacion.time − listaEliminaciones.get(0).time <= T then
11:         if listaEliminaciones.length() == N then
12:             veloc=(sum(listaEliminaciones.minimo)) / (eliminacion.time - firstTGlobalMi-
    nimumIncrease)
13:             if veloc >= Vthres then
14:                 alarma();
15:             end if
16:         end if
17:     else
18:         listaEliminaciones.deslizar()
19:     end if ▷ Se elimina de la lista la eliminacion mas antigua y se mueven las demas una
    posicion
20: end function
```

Es en este método donde se va a implementar también la comprobación del número de eliminaciones que se han producido hasta el momento, el tiempo que ha pasado entre la primera y la última y la velocidad de aumento del mínimo entre ellas. Por lo tanto, el algoritmo determinará que un

comportamiento es sospechoso durante la ejecución de este método que, como se ve en el algoritmo 1, es llamado con la llegada de determinados *CLOSE_RESPONSE*.

Además de lo visto en el pseudocódigo, hay una serie de extensiones que se van a excluir del algoritmo por varias razones. La más importante es que los ransomwares las ignoran ya que no tiene sentido perder tiempo encriptándolas porque no son datos de usuario o son datos temporales. Otra razón es que la propia naturaleza o el propósito de los archivos con estas extensiones provoca comportamientos similares a los de un ransomware. El ejemplo más claro son los ficheros temporales que crea Office cuando guardamos un archivo. El escenario planteado en este trabajo se basa en el almacenamiento de datos de usuario en volúmenes en red, no en el almacenamiento de datos internos de programas, por lo que las extensiones con las que vamos a trabajar son mayoritariamente extensiones de ficheros de usuario (.jpg, .pdf, .docx, etc). Dicho esto, lo primero que se va a mirar cuando se abre un fichero es su extensión, ya que si no está en la lista de extensiones *encriptables* por el ransomware los mensajes asociados a ese fichero no se va a procesar.

En caso de que el ransomware renombre un fichero y cambie su extensión por una fuera de la lista, ese fichero se marcará como eliminado y los bytes del tamaño del fichero se marcarán como leídos y escritos a efectos prácticos. De no hacerlo así, sería una forma muy evidente de evasión de nuestro detector. La lista de extensiones que encriptan los ransomwares puede verse en el anexo B.

Los valores de tipo entero: número de eliminaciones, tiempo de la ventana y velocidad umbral, deben ser introducidas por el usuario. Lógicamente, el valor de estos parámetros debe ser estudiado cuidadosamente, puesto que de ellos dependerá el número de ficheros y el tiempo que pase desde el comienzo de la encriptación hasta la detección. Para encontrar los valores óptimos, se presentará en la sección 5.4 el análisis que se ha hecho para conseguir obtener unos valores con el 100 % de detección del malware y el menor número de falsos positivos posible.

### 5.2.1. Correspondencia entre variables y parámetros

En el apartado 5.1 se han visto los parámetros del algoritmo, expresados a través de las fórmulas, mientras que en el apartado 5.2 hemos visto las variables del programa. Para clarificar el comportamiento del algoritmo, se va a proceder a asignar a cada variable del programa su parámetro del algoritmo.

- $\rho_i$ : variable *readResponse.time*
- $\omega_i$ : variable *writeResponse.time*
- $\tau_i$ : variable *eliminacion.time*
- $p_{\rho i}$ : variable *readResponse.path*
- $p_{\omega i}$ : variable *writeResponse.path*
- $p_{\rho i}$ : variable *eliminacion.path*
- $b_{\rho i}$ : variable *readResponse.length*
- $b_{\omega i}$ : variable *writeResponse.length*
- $P_p(t)$ : variable *paths.get(readResponse.path).bytesLeidos*. Ecuación 10
- $\Omega_p(t)$ : variable *paths.get(readResponse.path).bytesEscritos*. Ecuación 11
- $m_p(t)$ : variable *paths.get(p).minimo*. Ecuación 12

- $\tau^*$ : variable *primerTIncrMinimoGlob*. Ecuación 15
- $V[k]$ : variable *veloc*. Ecuación 14
- $\eta_i$ : variable *eliminacion.firstTIncrMinimo*. Ecuación 16

## 5.3. Ejemplo

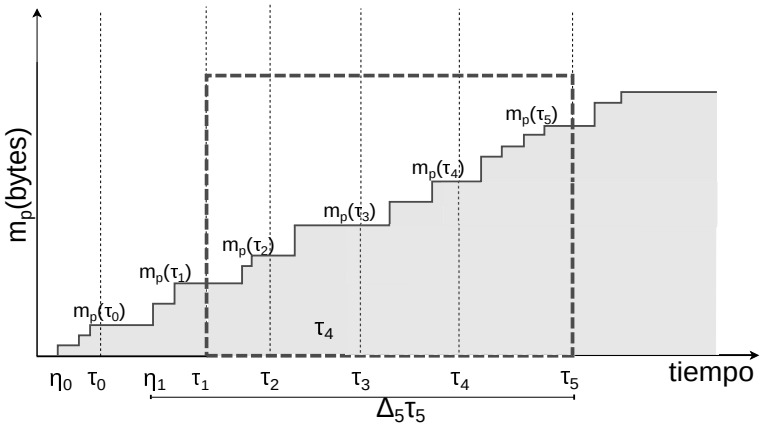



**Figura 14:** Ejemplo de funcionamiento del algoritmo

En la figura 14 se puede ver un ejemplo del funcionamiento del algoritmo, pero por simplicidad se ha supuesto que solo hay un directorio en nuestro sistema de ficheros compartidos. En caso de haber más directorios, los eventos de eliminación que ocurran en ellos se computan de forma similar al ejemplo de la figura. Además, habría que calcular el inicio del intervalo temporal para calcular la velocidad con las ecuaciones 15 y 16.

El ejemplo tiene $N = 5$ y se puede ver cómo va incrementándose el mínimo de los bytes leídos y escritos en el directorio en cada una de las eliminaciones. Estaríamos ante un comportamiento típico de ransomware. El incremento en los bytes tiene una forma similar al visto en la figura 6. La alarma podrá saltar solo si $\Delta_N \tau_k$ es menor que T. En este caso, habría habido una primera ventana desde $\tau_0$ hasta $\tau_4$ (5 eventos de borrado) en la que no habría podido saltar la alarma, bien porque la $\Delta_N \tau_k > T$ o porque la velocidad no superaba $V_{umbral}$. La ventana se desliza hasta la siguiente eliminación, y se vuelven a computar los contadores.

En la figura 14 se puede entender mejor qué es $\eta_i$ y para qué se utiliza.También se puede apreciar cómo afectaría el incremento en T o el incremento en N a la velocidad de detección del ransomware y a la posibilidad de que hubiera falsos positivos, para lo que debemos observar el comportamiento típico de los usuarios (Figuras 8a y 8b). Las eliminaciones se encuentran separadas un tiempo mayor que T (ver figura 9) con una probabilidad bastante alta, y además aún en caso de entrar dentro del tiempo de ventana, el incremento del mínimo debe ser lo suficientemente grande para que la velocidad supere el umbral, lo que decrementa las probabilidades de falsos positivos.

## 5.4. Análisis de parámetros

Una vez que se ha entendido el funcionamiento del algoritmo y se intuyen teóricamente los efectos que puede tener el cambio en los parámetros iniciales del mismo, se va a proceder a variar

estos parámetros de forma que se vea el verdadero efecto que tienen sobre la detección y sobre los falsos positivos.

Como el objetivo de este análisis es encontrar los parámetros $N$, $T$ y $V_{umbral}$ para los que los falsos positivos se minimizan consiguiendo una detección del 100 % de los ransomwares, lo primero que se va a hacer es un barrido de $T$ y $N$ en las trazas de usuario de entrenamiento. Con esto vamos a conseguir el mínimo valor de V que hay que poner para cada pareja de valores $T$ y $N$ de tal forma que en esa traza no se produzcan falsos positivos. Es decir, calcularemos la $V_{umbral}$ como la máxima velocidad entre todos los falsos positivos que se producen para cada uno de los parámetros $T$ y $N$, siendo también la mínima velocidad que habrá que poner en el algoritmo para que no se produzcan estos falsos positivos.

Haciendo este cálculo, obtenemos la figura 15, donde se puede ver la relación que se establece entre los tres parámetros del algoritmo. Conforme crece $N$, la velocidad de los falsos positivos decrece, puesto que es más improbable que un usuario haga una secuencia de eliminaciones muy alta con un incremento del mínimo sustancial, como para que la velocidad sea muy alta.

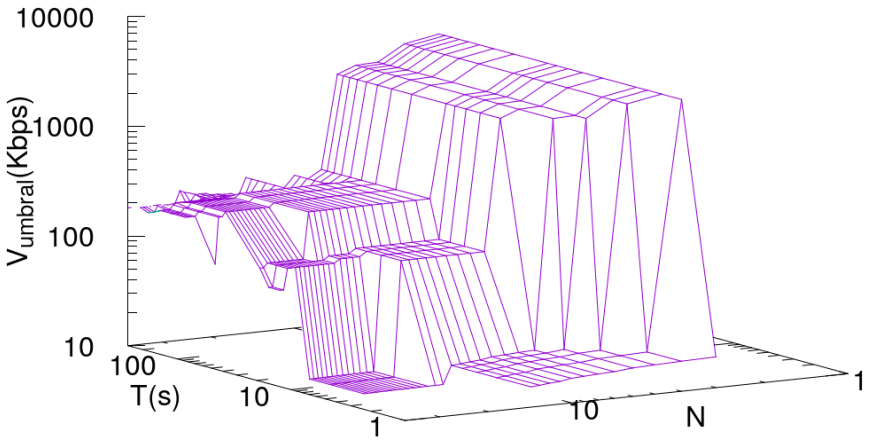
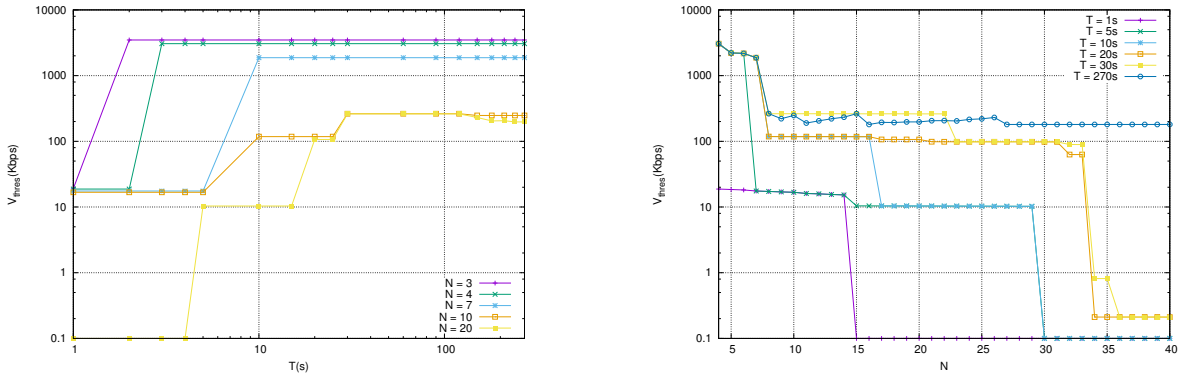



**Figura 15:** $V_{umbral}$ calculada durante el entrenamiento para las parejas de valores $T$ y $N$

En cuanto al parámetro $T$, conforme disminuye, también lo hace la velocidad. Aunque crezca la $N$, si también aumenta la $T$, se mantiene la velocidad en torno a los 300Kbps, velocidades que luego se verá hasta qué punto disminuyen la detección temprana de los ransomwares.

Para $N$ muy pequeña, independientemente de la ventana temporal que se seleccione, se van a producir eliminaciones y la velocidad en este caso se acerca a los 100Mbps, lo cual ya se puede imaginar (aunque luego será comprobado) que disminuye el porcentaje de detección del algoritmo. En la figura 16 se ven dos cortes de la figura 15, barriendo $T$ para distintos valores de $N$ (figura 16a) y barriendo $N$ para distintos valores de $T$ (figura 16b). De forma general se ve lo que se ha comentado anteriormente: la tendencia de la velocidad cuando incrementamos $N$ es decreciente mientras que

el incremento de $T$ tiene el efecto contrario. Se podría pensar ya viendo estos resultados que va a ser complicado detectar el ransomware con eficacia para $N < 7$, porque las velocidades son muy elevadas.




(a) Barriendo $T$ para distintos valores de $N$ (b) Barriendo $N$ para distintos valores de $T$

**Figura 16:** Velocidad umbral para que no haya falsos positivos en la traza *1dUPNA*.

Para concretar más y poder deducir los parámetros apropiados, se tiene que analizar el comportamiento de los mismos con trazas de ransomwares reales. Para eso se ha pasado el algoritmo para cada una de las muestras presentadas en la tabla 3, seleccionando los parámetros que salían de los experimentos con las trazas de usuario.

La figura 17 muestra el porcentaje de veces que se han perdido $N$ ficheros (que es el mínimo número de ficheros que se pierden antes de detectar el ransomware). En un color más oscuro se ven los resultados cuando se detectan todas las familias de ransomwares presentes en la Cuadro 3. Es decir, cuando se consigue un 100 % de índice de detección, aunque no en todos los casos se detecta el ransomware perdiendo $N$ ficheros, en algunos casos se pierden más.

Para conseguir este porcentaje, se ha pasado el algoritmo barriendo los parámetros $N$ y $T$, y poniendo $V_{umbral}$ como salía en los experimentos con usuarios reales. Cuando saltaba la alarma en una de estas trazas infectadas, se reseteaban los valores de la ventana y el algoritmo comenzaba de nuevo como si fuera el inicio de la encriptación por parte del ransomware. Para poder hacerlo de esta manera, se consideran los procesos de eliminación como independientes, de forma que uno no dependa de los anteriores. De esta forma de cada muestra de ransomware recogida se obtiene un gran número de eventos de borrado y de alarma, siendo equivalente a analizar un gran número de muestras de cada familia.

Los resultados rondan el 99 % para una detección de todas las familias (que es la zona que nos interesa). Para $T$ pequeño y $N$ grande, hay familias que no se detectan, ya que no eliminan ficheros a tanta velocidad. Ya sea porque leen y/o escriben más despacio o por el algoritmo de encriptación, no son capaces de encriptar en menos de un segundo tantos ficheros como para provocar un salto de la alarma. En esta zona es donde las estadísticas son más bajas, llegando a ser de un 20 % para los mayores $N$ y menores $T$.

Para $N$ pequeño también hay una pequeña bajada en el porcentaje, que llega a ser de en torno al 90 %. Es debida a que la velocidad umbral calculada en esa zona es mucho mayor y por lo tanto aunque se producen los $N$ borrados en la ventana $T$, algunas de esas veces (en torno al 10 %) no llega a la velocidad de aumento del mínimo de bytes leídos y escritos necesaria para que salte la alarma. En estos casos el ransomware sí es detectado en el 100 % de los casos, pero en el 10 % de estos casos, ha perdido más de $N$ ficheros. Más adelante se verán una serie de gráficas donde se analiza cuántos ficheros más se han perdido en estos casos.

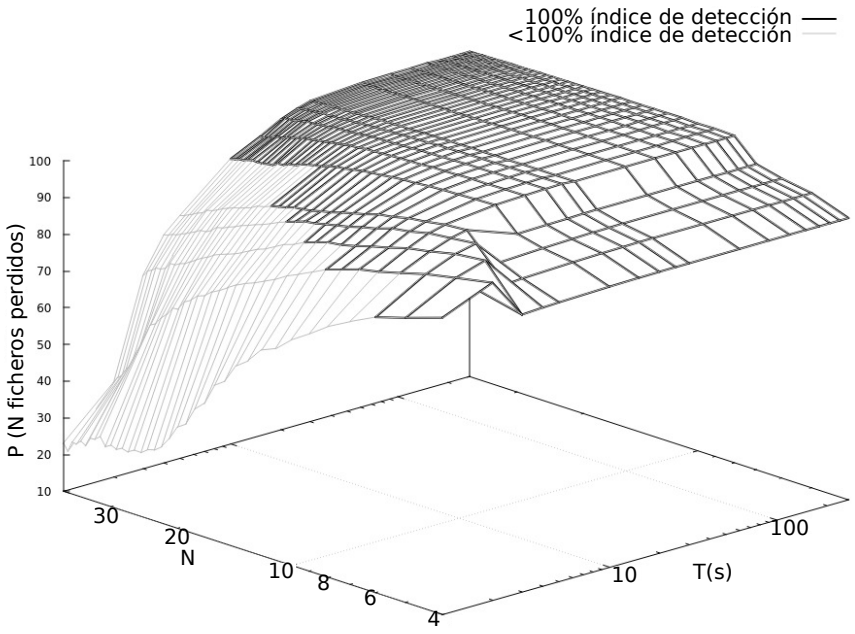



**Figura 17:** Porcentaje de veces que se pierden N ficheros

En la Figura 17, se aprecia que la zona de más interés, por sus probabilidades de detección, es la de $N > 7$ y $T > 8$, ya que es el índice más alto en el que podemos perder $N$ ficheros. Para $N$ muy grande, donde el porcentaje es del 100 %, se estarían perdiendo en todas las alarmas $N$ ficheros (siendo $N$ un número elevado de ficheros). Por otro lado, para $N$ en torno a 10, en un 99 % de las alarmas solo se pierden $N = 10$ ficheros.

Para analizar el caso de ese 1 % de las alarmas en que se pierden más de 10 ficheros, se ha calculado el percentil 99.9, que es la máxima cantidad de ficheros que se pierden en el 99.9 % de las alarmas.

En la figura 18 se ha recortado la imagen para poder apreciar mejor los valores exactos del percentil. El eje de $N$ va desde 4 hasta 15 cuando en figuras anteriores (figura 17) llegaba hasta 40. Los datos para $N > 15$ no aportan demasiada información en este caso. Se observa que para $N$ grande y $T$ pequeño el percentil obtiene su máximo valor: por encima de 80 ficheros, mientras que para $N$ pequeño y $T$ más grande, el incremento de la $V_{umbral}$ provoca que el percentil se sitúe en 50 o 60 ficheros, haciendo de esta zona la segunda con el percentil más alto. Los datos de la Figura 17 concuerdan con los que se ven en ésta, ya que para la zona donde el percentil es de 50 o 60 ficheros, la probabilidad de perder solo $N$ ficheros disminuye (hay más probabilidad de que se pierdan más). Además se observa que para valores de $T$ en torno a 20 segundos y $N = 10$ ficheros el valor del percentil se sitúa en torno a los 20 o 25 ficheros. Esto asegura que solo 1 de cada 1000 veces el algoritmo detectará el ransomware perdiendo más de 25 ficheros y, además, en la Figura 17 se ve que será detectado cualquier tipo de ransomware.

Hasta ahora se ha medido la eficacia de la detección en función de los ficheros que se pierden. También se puede analizar desde el punto de vista del tiempo que tarda en detectarlo. Este tiempo va a estar directamente relacionado con el tamaño de los archivos en el directorio compartido, ya que el ransomware va a tener que leer, encriptar y escribir ese contenido. Se ha construido una función de probabilidad acumulada para mostrar estos resultados.

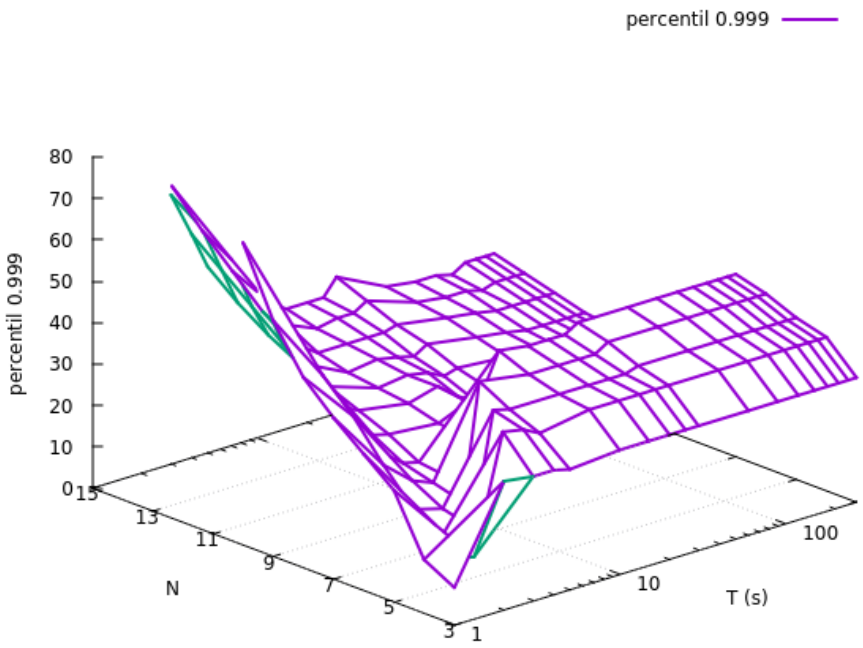



**Figura 18:** Cantidad máxima de ficheros que se pierden el 99.9 % de las veces

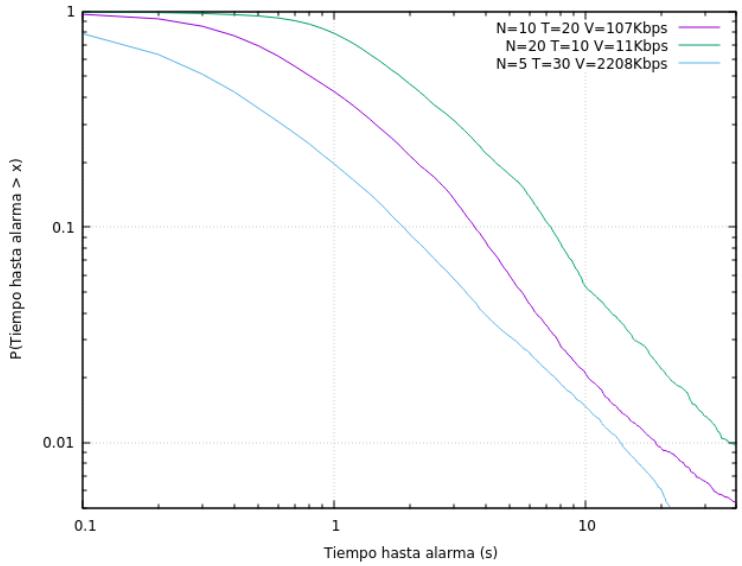



**Figura 19:** Distribución de probabilidad acumulada para el tiempo hasta que el ransomware es detectado

En la figura 19 se puede ver la probabilidad de que REDFISH tarde más de un cierto tiempo en detectar un ransomware. Se muestran los resultados con todos los ransomwares juntos para tres conjuntos de parámetros de configuración del algoritmo. Para $T = 30s$, $N = 5$ y $V_{umbral} = 2208Kbps$ se obtiene una distribución con tiempos menores, pero a su vez, se puede ver gracias a la figura 13

que el algoritmo no es capaz de detectar todos los ransomwares. Por lo tanto, esta opción, aunque tiene una distribución que puede parecer mejor, queda descartada. Para el caso de $T = 20s$, $N = 10$ y $V_{umbral} = 107Kbps$, quedan tiempos por debajo de los 20 segundos para el 99 % de los casos. quedan tiempos por debajo de los 20 segundos para el 99 % de los casos. Solo para el 10 % de los mismos al algoritmo le lleva más de 4 segundos de actividad del ransomware el detectarlo. Para el último conjunto de parámetros mostrado en la figura ($T = 10s$, $N = 20$ y $V_{umbral} = 11Kbps$), los tiempos son algo peores, costando un máximo de 40 segundos detectarlo en el 99 % de los casos, y en el 90 % puede costar como mucho alrededor de 8 segundos detectar el ransomware.

Viendo estas figuras, se ha decidido optar por los parámetros $T = 20s$, $N = 10$ y $V_{umbral} = 107Kbps$ para el resto del documento, por dar los mejores resultados tanto en ficheros perdidos como en tiempo y en porcentaje de detección de las familias de ransomware escogidas.

Aunque se han seleccionado los parámetros para llevar a cabo los siguientes apartados de análisis de los resultados, estos se pueden modificar en función del escenario que se tenga pues, como se ha comentado, el tamaño de los archivos va a afectar al tiempo que necesitan los ransomwares para encriptar los archivos. Para ficheros pequeños es conveniente poner ventanas de tiempo pequeñas, de forma que reduzcamos la probabilidad de falsos positivos. Manteniendo la $N$ en torno a 10, o bajándola hasta 5 se pueden utilizar ventanas pequeñas en cuanto a tiempo ($T$).

# 6. Resultados experimentales

El proceso de selección de los parámetros óptimos para el algoritmo se ha realizado utilizando todas las muestras de ransomwares presentadas en la sección 3. Para $N = 10$, $T = 20$ y $V_{umbral} = 107Kbps$, se detectan el 100 % de las muestras, en el 99 % de los casos con tan solo $N$ ficheros perdidos (ver figura 17). Además, en un 99 % de los casos ha saltado la alarma en menos de 20 segundos desde que empezara la actividad del ransomware (ver figura19).

A pesar de estos resultados, los diferentes comportamientos que tienen las distintas familias de ransomwares pueden provocar que haya distintos resultados en función de cada una de las familias presentadas. Como se explicó en la sección 3.3, se han clasificado los ransomwares en tres tipos en función de este comportamiento. Debido a que algunos realizan más acciones sobre los archivos, o a que son acciones más costosas (en términos de paquetes en la red), determinadas familias van a tardar más que otras en encriptar los archivos. Además, el orden en el que se va a producir la encriptación por el árbol de directorios e incluso dentro del mismo directorio puede afectar a la velocidad en la detección del ransomware. Esto es porque en función del tamaño de los ficheros las eliminaciones van a separarse más entre ellas, provocando que tal vez la ventana $T$ no sea lo suficientemente grande para abarcar $N$ eliminaciones y con ello que se pierdan más de $N$ ficheros antes de detectar la infección.

En cuanto a las trazas no infectadas de usuario, el entrenamiento del algoritmo se ha realizado con la traza *1dUPNA*, de donde se han sacado los valores de la $V_{umbral}$ en la sección 5.4. A partir de esta traza de un día en el campus de la UPNA se determinan los parámetros para que no se produzcan falsos positivos. A pesar de esto, pueden producirse falsos positivos en otro día distinto en el mismo escenario (trazas *1sUPNA-X*) o también en un día en un entorno diferente, como puede ser una empresa privada (traza *1dPrivada*). Las características de las trazas se han expuesto en la sección 3.2.

A lo largo de este capítulo se van a exponer por un lado los resultados del algoritmo para cada familia de ransomware individualmente, analizando las razones por las que los resultados de detección se alejan de la media global. A continuación se evaluará el algoritmo en escenarios sin presencia del malware, observando la aparición de falsos positivos y sus razones, proponiendo luego soluciones para evitarlos o para reducir su impacto en la red.

## 6.1. Resultados para las distintas familias

La figura 20 muestra el porcentaje de veces que ha saltado la alarma en las que se han perdido sólo $N$ ficheros. En este caso se ha configurado $T = 20$ y se ha barrido el parámetro $N$ en torno a $N = 10$, de forma que para cada pareja $T$, $N$ se ha configurado el valor correspondiente de $V_{umbral}$.

Por claridad se han representado en la figura las familias más representativas o aquellas cuyos comportamientos pueden dar algún problema al algoritmo. Se distinguen dos zonas claras, que ya se veían en figuras anteriores (figura 17): una hasta $N = 7$ y otra a partir de ahí. Esto es debido sobre todo a la velocidad umbral, que se veía que era mayor en la primera zona y por lo tanto provoca peores resultados en cuanto a detección. Cuando la velocidad cae, los índices de detección perdiendo N ficheros se disparan hasta casi el 100 % en la mayoría de familias. No así en el caso de "wannaCry" ni en el caso de "CTBLocker".

Para el primero ya se ha comentado con anterioridad (ver sección 3.2) que su comportamiento es algo diferente al del resto. El hecho de que elimine los ficheros a ráfagas provoca que las eliminaciones se concentren en intervalos cortos de tiempo, pero el primer aumento del mínimo ($\eta_i$ en el algoritmo, sección 5.1) se produce alejado en el tiempo, lo que provoca una disminución en la velocidad efectiva del ransomware.

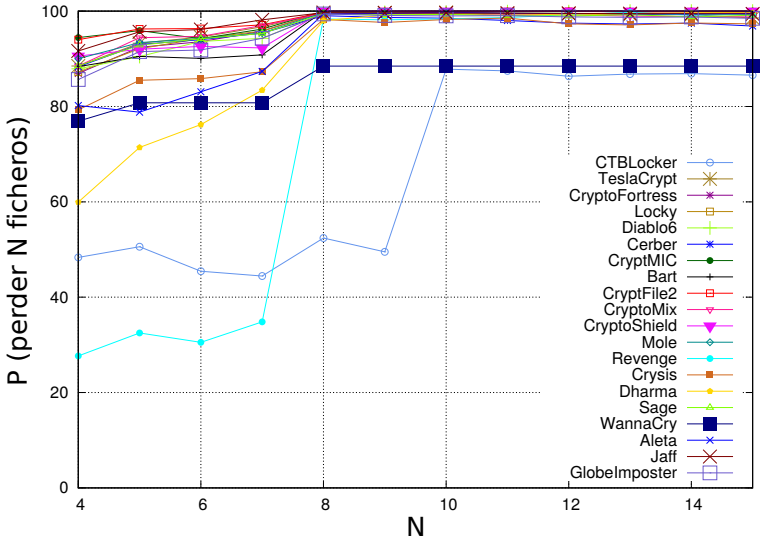



**Figura 20:** Porcentaje de casos donde se pierden solo $N$ ficheros para $T = 20$ segundos

En el caso de "CTBLocker", lo que ocurre es que a pesar de que lee todo el archivo completo y lo encripta, también lo comprime, por lo que escribe menos bytes en el fichero destino. El resultado es que el mínimo no aumenta tan rápido como en otros casos y esto provoca que los resultados sean peores en cuanto a velocidad. En el caso del "Revenge", que es otro ransomware cuyos resultados destacan por ser más bajos que el resto, el problema es simplemente que la velocidad de encriptación de los datos es más baja. Es por esto que en el momento que baja esa velocidad (lo cual se da cuando $N > 7$), los resultados pasan a ser similares a los de otras familias.

A continuación se realiza el mismo experimento pero para diferente valor de $T$ (figura 21. Se realiza con $T = 120$ segundos,y analizando los resultados se ve que para casi todas las familias son similares a los anteriores (Figura 20. El porcentaje está en torno al 100 % para la mayoría de las familias, incluyendo "WannaCry", que mejora sustancialmente respecto a $T = 20$. En el caso de "CTBLocker", en cambio, los resultados caen hasta algo menos del 80 %.

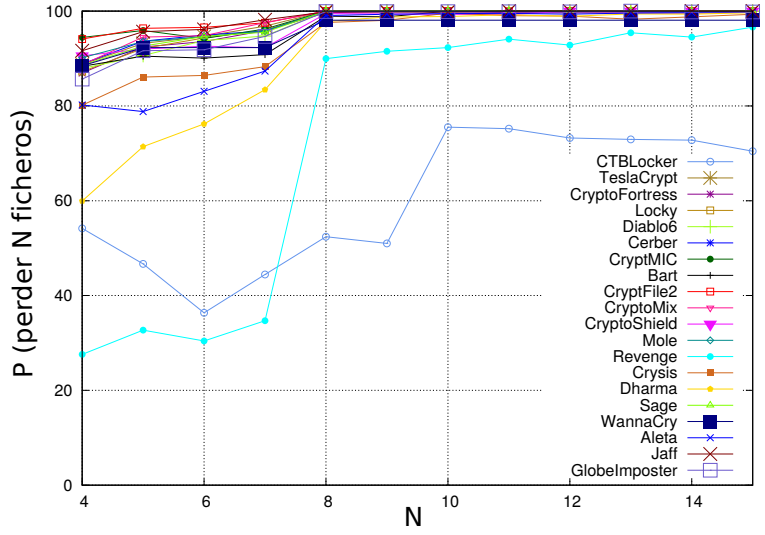



**Figura 21:** Porcentaje de casos donde se pierden solo $N$ ficheros para $T = 120$ segundos

Viendo estos resultados, parece más adecuado tener una ventana temporal de $T = 20$, donde todas las familias son detectadas perdiendo $N$ ficheros cerca del 90 % de las veces. Las figuras 20 y 21 expresan la probabilidad de perder $N$ ficheros pero, al igual que sucedía en el caso de las gráficas conjuntas de todas las familias, es necesario saber, para hacerse una idea de la efectividad del algoritmo, la cantidad máxima de ficheros que se pierden en el 99 % de los casos. Esta medida es el percentil 99, y se muestra en la figura 22.

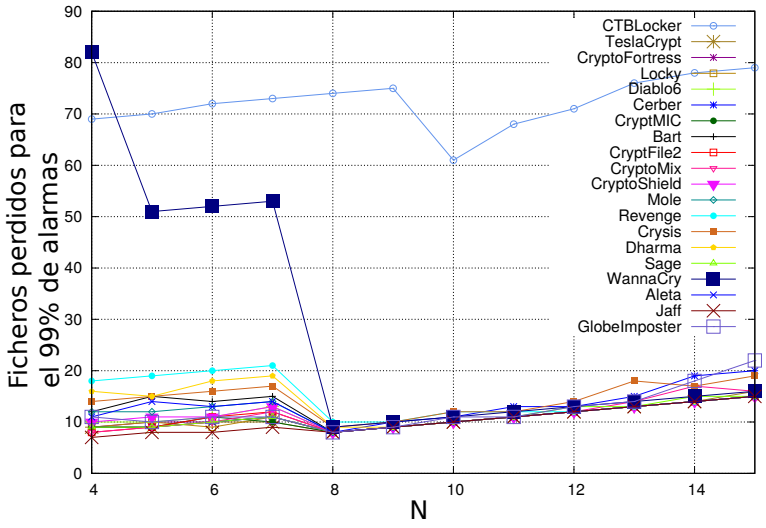



**Figura 22:** Máximo número de ficheros perdidos para el 99 % de las alarmas con $T = 20$s

De nuevo se aprecia la mejora de los resultados para $N > 7$, al igual que en gráficas anteriores. Para $N = 10$, el percentil 99 está por debajo de 12 para todas las familias excepto para "CTBLocker", lo cual significa que en el 99 % de las alarmas, el mayor número de ficheros que se ha perdido es 12. Estos resultados son muy positivos porque en muchos casos la pérdida de 12 ficheros es asumible, e incluso podrían recuperarse como se explicará en la sección 7.3. El caso de "CTBLocker" sigue siendo el que peor resultados proporciona, con un percentil 99 que ronda los 70 ficheros a lo largo de los distintos valores de $N$. Es una cantidad de ficheros considerable, aunque es importante mencionar que la media de ficheros perdidos para el mismo está en torno a 20, aunque esta cifra no se ve en la gráfica.

Se consiguen pues unos muy buenos resultados de todas las familias excepto de "CTBLocker" cuyos resultados no son malos del todo, (90 % de las alarmas saltan con $N$ ficheros perdidos), pero sí son peores que los de otras para el máximo número de ficheros que se pierden en ese 10 % restante de las alarmas.

## 6.2. Validación con trazas reales de usuario

Los parámetros más apropiados se han establecido analizando los resultados en unas trazas de usuarios de un solo día en un entorno determinado. Parece lógico que pueda haber cambios bastante significativos cuando el comportamiento de usuario varía. Para analizar cómo afectan estos cambios, se han analizado trazas de usuarios del mismo entorno pero a lo largo de una semana de trabajo, y un día de usuarios en un entorno diferente.

Una vez se analicen las trazas, se estudiarán los resultados y se determinará si la elección de los parámetros hecha en la sección 5.4 ($T = 20$, $N = 10$ y $V_{umbral} = 107Kbps$), es válida, observando la

cantidad de falsos positivos que podrían saltar en función de la cantidad de usuarios en el escenario. Los resultados se muestran en la tabla 5.

| | | | **Número de falsos positivos** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| T | N | $V_{thres}$ (Kb/s) | 1sUPNA-01 | 1sUPNA-02 | 1sUPNA-03 | 1sUPNA-04 | 1sUPNA-05 | 1sUPNA-06 | 1dPrivada |
| 20 | 10 | 107 | 0 | 1 | 0 | 0 | 0 | 1 | 2 |
| 20 | 12 | 107 | 0 | 1 | 0 | 0 | 0 | 1 | 2 |
| 20 | 14 | 107 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 20 | 16 | 107 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 20 | 21 | 107 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 20 | 61 | 107 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | | | | | | | | |
| 120 | 10 | 263 | 1 | 1 | 0 | 0 | 0 | 0 | 3 |
| 120 | 12 | 263 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 120 | 14 | 263 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 120 | 21 | 263 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 120 | 61 | 263 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Tabla 5:** Falsos positivos para los distintos entornos

Aunque los resultados para los ransomwares, sobre todo vistos individualmente, eran mejores para $T = 20$, se ha decidido hacer la validación con los dos valores de la ventana temporal, ya que en el caso de que hubiera gran diferencia con los falsos positivos la balanza podría decantarse más por una o por otra configuración. En la tabla 5 vemos los resultados conforme se aumenta $N$ para los dos casos de $T$. En el caso de $N = 10$, de los seis días del entorno de la universidad, sólo en dos de ellos se han producido falsos positivos, uno de los cuales desaparecía cuando la $N > 20$ y el otro cuando $N > 60$. Para el entorno de empresa privada, los dos falsos positivos que aparecen en el caso de $N = 10$, desaparecen con $N > 12$ y $N > 15$.

Hay que destacar que en la empresa privada la traza contiene conexiones de 4882 usuarios, accediendo a 1677 servidores SMB diferentes durante un día completo (24 horas). Esto también incluye los posibles procesos que se realicen durante la noche en la empresa en cuestión. Que de esos 4882 usuarios a 2 les haya saltado una falsa alarma, significa que salta falso positivo a un 0.04 % de los usuarios en un día.

También se puede apreciar que los resultados son similares para los dos valores de $T$. Aunque aparece un falso positivo más para $T = 120$ en la empresa privada y en uno de los días de la UPNA, desaparece uno en otro de los días. El hecho de que aparezca algún nuevo falso positivo se debe al aumento del tiempo de la ventana, que da más tiempo a los eventos de eliminación para producirse. En el caso contrario, la desaparición de falsos positivos es provocada por el aumento de la $V_{umbral}$.

Para el caso de la empresa privada (*1dPrivada*), se producen tres falsos positivos en todo el día para $T = 120$ segundos. Analizándolos uno a uno más en detalle, se deben a dos situaciones distintas:

- Modificación de ficheros de bases de datos: Normalmente este tipo de ficheros se sobreescriben cuando se necesita modificarlos. Muchos programas o aplicaciones aplican múltiples cambios en bases de datos en períodos muy cortos de tiempo. En este caso concreto el usuario ha modificado 10 archivos de bases de datos en menos de dos minutos. Estos falsos positivos son complicados de evitar porque su comportamiento es muy similar al de un ransomware. A pesar de esto, podríamos evitarlos si estas bases de datos se encuentran en directorios que no se alojan en el NAS. Recordemos que el escenario es un NAS en el que se alojan ficheros de usuario, no ficheros de programas. Por lo tanto, tal vez estas bases de datos no estén dentro del escenario para el que está pensado REDFISH. También se podrían evitar si se considera que estas bases de datos no son cruciales en el sistema y no se consideran extensiones a monitorizar aquellas como por ejemplo *.sqlite* ó *.db*. Dicho esto, no es una buena práctica alojar una base de

datos importante para la empresa en la carpeta de documentos del usuario, y es recomendable además guardar copias de seguridad de la misma en otro servidor y de forma automática.

- Modificaciones de archivos de Microsoft Excel en un intervalo corto de tiempo: Esto probablemente se debe a que unos archivos están enlazados con otros y cambios manuales en alguno de ellos pueden provocar cambios en cadena en otros muchos. La sobreescritura de muchos ficheros se asemeja a la actividad de un ransomware. Este tipo de falso positivo también es complicado de evitar, puesto que se trata de ficheros creados y modificados por el usuario, con una extensión que es habitual. En las trazas analizadas ha ocurrido dos veces en el mismo día, ambos sucesos provocados por el mismo usuario. Viendo la complejidad que hay para evitarlos se puede optar por la solución de reducir el impacto sobre el usuario. Por seguridad la mejor práctica sería bloquear el acceso del usuario al servidor en cuanto se detecta un comportamiento sospechoso, y lanzar un aviso a usuario y administrador para que analicen esta alarma.

En el caso de la misma traza pero para $T = 20$ segundos, se producen dos falsos positivos, uno por cada una de las razones mencionadas para $T = 120$ segundos. Uno de los tres falsos positivos con $T = 120$s relativos a los archivos de Microsoft Excel desaparece debido a que las sobreescrituras se producen en un intervalo de tiempo mayor que $T$ (20 segundos).

Estas tres alarmas se producen a lo largo de un día donde los usuarios abren en total más de 40 millones de archivos (ver tabla 2). Esto da un índice de falsos positivos de sólo 1 entre 10 millones de archivos abiertos. Además, para el caso de $T = 120$ y $N > 12$, no hay falsos positivos en ningún momento del día.

Pasando ahora a las trazas de la universidad (*1sUPNA-X*), los dos falsos positivos para $T = 20$ segundos que se producen en los días 2 y 6 son por dos razones:

- Borrado de ficheros PDF: El usuario copia un conjunto de ficheros PDF sobre el mismo directorio y a continuación elimina los anteriores. El resultado es un comportamiento muy similar al de un ransomware, ya que al copiar los archivos la aplicación tiene que leer los originales, escribir los nuevos y si a continuación se borran los originales no hay forma de distinguir este comportamiento de uno infectado. Tendremos que optar por la solución explicada anteriormente de bloquear el acceso y lanzar un aviso al usuario, que debe ser consciente de lo que está haciendo para indicar que no se trata de un ransomware. No parece un comportamiento muy habitual por parte de los usuarios, de hecho sólo uno de los usuarios lo hace en una semana de capturas en un entorno de casi 400 usuarios (ver tabla 2).

- Descarga de una página web: Un usuario parece haberse descargado una página web (archivos *.xml, .js*, etc...) y a continuación los ha eliminado de su máquina. El proceso de guardado de páginas web en el equipo y el hecho de que momentos después haya eliminado los archivos es lo que provoca el falsos positivo. Como en el caso anterior, es difícil diferenciarlo de un comportamiento realmente infectado, y por lo tanto habrá que tomar medidas para reducir el impacto sobre el usuario. Al igual que antes, es un comportamiento poco común en los usuarios.

Para el caso de $T = 120$ en las trazas de la universidad, se observan dos falsos positivos, uno de los cuales se produce por el borrado de ficheros PDF, y el otro es debido al borrado de una serie de ficheros, de los que no se puede deducir ni su procedencia ni si el usuario lo ha hecho manualmente o ha sido alguna aplicación. Son ficheros con extensión habitual de usuarios y perfectamente encriptable por un ransomware.

De nuevo destaca el bajo porcentaje de falsos positivos, incluso menor que en la traza de la empresa privada, aunque en este caso se han escogido sólo las horas de trabajo en oficina, pues durante el resto el uso del servidor NAS era ínfimo. En el peor caso, tenemos que poner $N = 60$ para conseguir que no haya falsos positivos en todo el día.

# 7. Seguridad, rendimiento y recuperación de archivos

Para completar el análisis de REDFISH, es necesario explicar tres aspectos importantes a tener en cuenta en toda herramienta o algoritmo de procesado. Como ya se ha comentado en secciones anteriores, el algoritmo no introduce retardo alguno en el tráfico entre el usuario y la cabina gracias a la copia que se hace del mismo en el 'port-mirroring' del switch. Descartando este aspecto, la seguridad y el rendimiento son importantes para evitar la desinstalación del software y la sobrecarga de la del equipo respectivamente. Estos dos aspectos se van a contemplar en esta sección, además de la posibilidad de recuperar los $N$ ficheros que han sido encriptados por el ransomware en el tiempo que ha tardado REDFISH en bloquear el acceso del usuario infectado a la cabina.

## 7.1. Seguridad

En la sección de resultados (sección 6) se ha visto que REDFISH es capaz de detectar el ransomware en pocos segundos, perdiendo tan solo 10 ficheros en el 99 % de los casos. Una vez que el ransomware es detectado, la sonda puede emplear mecanismos de control SDN para configurar el bloqueo del tráfico de ese usuario e impedir que siga encriptando los ficheros del NAS.

Como las probabilidades de falsos positivos son muy bajas (1 entre 10 millones de archivos abiertos), la opción del bloqueo inmediato es la más recomendable. Aún así, para evitar molestias a ese bajo porcentaje de usuarios que van a sufrir un falso positivo, se puede enviar una alerta al administrador de la red o al propio usuario para que sean conscientes de la posible infección. En este caso, el administrador puede tardar unos minutos en comprobar si se trata o no de una infección, por lo que pueden perderse un mayor número de ficheros y causar daños mayores. La aplicación de una u otra opción dependerá del escenario y de la disponibilidad o no de un administrador en la red. La tercera opción es una intermedia entre las dos anteriores. Se trata de una reducción de los privilegios al posible usuario infectado, dejándole permisos sólo para leer archivos del NAS. Es una alternativa menos invasiva y cuyos efectos no son tan graves como las anteriores. El usuario puede que no se dé cuenta de la restricción, mientras que el ransomware va a verse detenido por su imposibilidad de escribir los archivos infectados o de eliminar los originales.

Una vez que ha saltado la alarma, se pueden tomar medidas, eliminar el ransomware del equipo en cuestión y continuar con una actividad normal del usuario. Estas acciones pueden realizarse de manera remota, accediendo desde la red de administración al equipo involucrado.

Un aspecto positivo de esta configuración es que el software no se encuentra instalado en cada uno de los equipos de la red, lo que facilita la instalación y las actualizaciones. Además, evita que el propio ransomware pueda eliminar este software del equipo, comportamiento que se ha detectado en algunos de los ransomwares como se ha comentado en la sección 2. Se debe configurar el equipo de forma que ninguna máquina de la red tenga acceso a él, pues de lo contrario podría infectarlo también. Idealmente solo se debe tener acceso a este equipo desde la red de administración de la empresa.

A todo esto hay que añadir la importancia de una buena administración de los privilegios de cada usuario en el NAS. Se debe tener especial cuidado de a qué ficheros y con qué permisos puede acceder cada usuario, ya que aunque se pierdan pocos ficheros gracias al algoritmo REDFISH, se reduciría el impacto si esos ficheros no son cruciales para la empresa.

## 7.2. Rendimiento

Como se ha visto en la tabla 2, la traza de la empresa privada *1dPrivada* contiene más de 1Tbyte de datos de tráfico, y alcanza más de 400Mbps de tráfico sostenido. Aunque el algoritmo no

introduzca retardo al tráfico del usuario, sí que debe ser capaz de absorber la velocidad del enlace, ya que de esta forma se podrá bloquear el acceso del usuario infectado al disco inmediatamente después de perder los $N$ ficheros. Si tuviera algo de retraso, podrían perderse un número mayor de archivos. Dicho esto, el prototipo es capaz de analizar la traza en un tiempo 32 veces menor que su duración real, usando un único núcleo del CPU core i5-4690 a 3.5GHz. Esto significa que pueden procesarse alrededor de 10Gbps de tráfico en tiempo real. Además, al contrario que otras herramientas como [33, 43, 63, 64, 46], REDFISH no necesita analizar el contenido de los ficheros. No necesita capturar los bytes leídos y escritos dentro de los mensajes SMB que se ven en la red y por lo tanto consigue velocidades mucho mayores. Para escenarios con velocidades mayores de 10Gbps, que cada vez son más comunes en redes empresariales, se pueden utilizar varios núcleos de la CPU o varias CPUs.

Otra ventaja de esta implementación respecto a las herramientas que se instalan en las máquinas de los usuarios es que la CPU no se comparte con el usuario. Toda la potencia y los núcleos del equipo se pueden dedicar al procesado, sin entorpecer ni ralentizar la actividad de los usuarios, que pueden trabajar con total normalidad incluso sin ser conscientes de que el algoritmo está funcionando.

## 7.3. Recuperación de los archivos

Uno de los problemas principales que tiene el algoritmo REDFISH es que al menos se pierden $N$ ficheros antes de que el ransomware sea detectado. La pérdida de estos ficheros no puede evitarse, ya que la detección se logra gracias a la acumulación de estos $N$ eventos de borrado. Soluciones como los backups son útiles sólo hasta cierto punto, ya que los datos modificados tras el último backup no podrían recuperarse. Algunas herramientas vistas en la sección 2 proponen soluciones de recuperación de ficheros que exigen modificar las llamadas a funciones del sistema para que no escriban datos sobre el fichero original, sino que lo hagan sobre un archivo temporal por si se trata de la acción de un ransomware, para tener intacto el fichero con los datos originales. Esto exige un mayor uso de memoria y es poco eficiente ya que por cada operación de escritura del usuario hay que modificar el archivo temporal.

La alternativa que se propone en este trabajo para complementar el algoritmo de detección se basa en la recuperación de los archivos a partir de la traza de tráfico entre el usuario y la cabina. La misma traza que el algoritmo ha utilizado para la detección del ransomware puede utilizarse para reconstruir los ficheros que han sido eliminados por el mismo.

### 7.3.1. Funcionamiento

El objetivo de esta herramienta es recuperar la última versión del fichero antes de que el ransomware entrara en acción. Para conseguir los datos SMB ordenados y poder decodificarlos, es necesario que se reconstruyan los flujos TCP sobre los que va este protocolo. Para esto, existen algunas herramientas como por ejemplo tcpflow [8].

En cuanto a herramientas existentes para la recuperación de archivos, no hay ninguna que se ajuste a estas necesidades. Lo más cercano es un plugin de Wireshark que recupera los ficheros abiertos por SMB, pero crea un fichero por cada CREATE de la traza sin hacer un seguimiento del fichero en cuestión. Esto es un problema porque si el usuario abre el archivo más de una vez para leer partes distintas del fichero no lo recuperará completo, sino en varios ficheros cada uno con una parte del original. Además, al no tener en cuenta el instante de tiempo en que entra en acción el ransomware recuperaría el fichero encriptado, haciendo imposible la recuperación de los datos originales.

La figura 23 representa un ejemplo de lecturas y escrituras que se pueden ver en una traza de tráfico SMB entre un usuario y un servidor antes y durante la infección del mismo por un

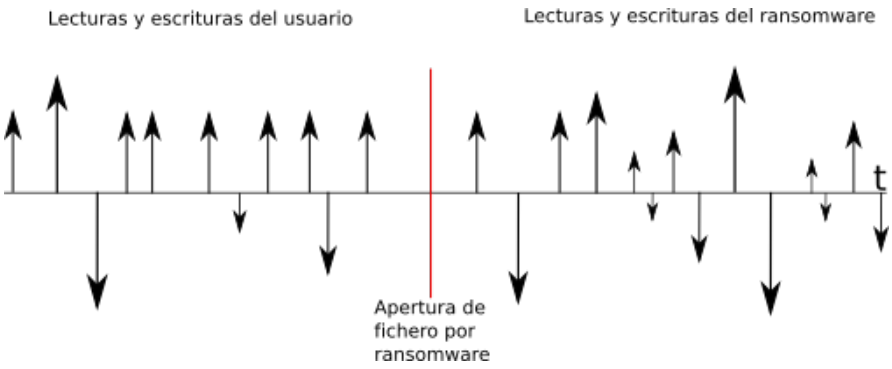



**Figura 23:** Línea temporal de una traza SMB de un ransomware

ransomware. Las flechas verticales representan las lecturas (hacia arriba) y escrituras (hacia abajo) y la línea vertical roja representa el CREATE que realiza el ransomware para encriptar el fichero. Todas las lecturas y escrituras mostradas se realizan sobre el mismo fichero, que es uno de los $N$ ficheros que se van a perder antes de la detección.

Lo ideal sería recuperar el fichero en la versión inmediatamente anterior al CREATE del ransomware (línea vertical). Para esto es necesario reconstruir el fichero a partir de los datos escritos y leídos por parte del usuario antes de ese instante. Hasta ese momento cualquier lectura o escritura en el archivo buscado debe volcarse a un fichero en disco con el offset y el tamaño marcado en el mensaje READ o WRITE de la traza.

Una vez que el ransomware ha abierto el fichero (instante conocido porque REDFISH lo proporciona), los WRITEs con cualquier fileID de cualquier conexión que hagan referencia a ese fichero (mismo path) no deben volcarse a disco, ya que puede tratarse de datos encriptados. Las lecturas que haga el ransomware pueden volcarse a disco siempre que esos datos no hayan sido escritos en instantes posteriores a la apertura del fichero por el ransomware. Por tanto y resumiendo, podemos volcar los datos de un READ a disco si:

- El timestamp del READ en cuestión es menor que el timestamp de apertura del fichero dado por REDFISH a la herramienta de recuperación.
- El timestamp del READ es mayor que el del CREATE y además esos datos no han sido escritos por el ransomware (no hay WRITEs de esos datos posteriores a la apertura del fichero por parte del ransomware).

En cuanto a los WRITEs, solo habrá un caso en el que se vuelquen a fichero. Cuando el instante temporal de la operación sea menor que el instante de apertura del fichero del ransomware (facilitado por REDFISH). En cualquier otro caso, se apuntarán esos bytes del fichero como escritos por el ransomware, de forma que si a partir de ese momento llega un READ a esos datos, no se vuelquen al fichero.

Actualmente esta herramienta está desarrollada únicamente para la versión 2 del protocolo SMB porque como se ha comentado es la más utilizada. Se han realizado pruebas de recuperación con las mismas trazas infectadas con las que se ha trabajado durante el documento y se han recuperado todos los archivos encriptados. El único caso en que la herramienta no recupera los archivos completos es cuando el ransomware no encripta el fichero completo. Esto es debido a que los datos de fichero no están en la captura y por tanto no se pueden recuperar. Por otro lado, si los datos no se ven en la red es porque el ransomware no los ha encriptado (no los lee y por tanto no puede hacerlo). En una instalación real de la herramienta, la captura almacenada debería ir desde el instante en que se

realizó el último backup hasta el momento de la infección, por lo que los datos que no se puedan recuperar a partir de la captura, estarán intactos en el backup. Así, podría recuperarse el fichero original combinando los datos que sí son recuperados por la herramienta con los sectores originales guardados en el backup.

### 7.3.2. Estructura

La herramienta va a recuperar la última versión no encriptada del fichero indicado por el usuario. Para esto debe seguir el flujo TCP, reconstruirlo y así decodificar los mensajes SMB. Cada conexión TCP puede tener varias conexiones SMB, identificadas mediante un TreeID y un SessionID, y cada una de ellas tiene una serie de ficheros abiertos identificados cada uno con un FileID. Así, la estructura interna del programa queda organizada como se ve en la siguiente figura.

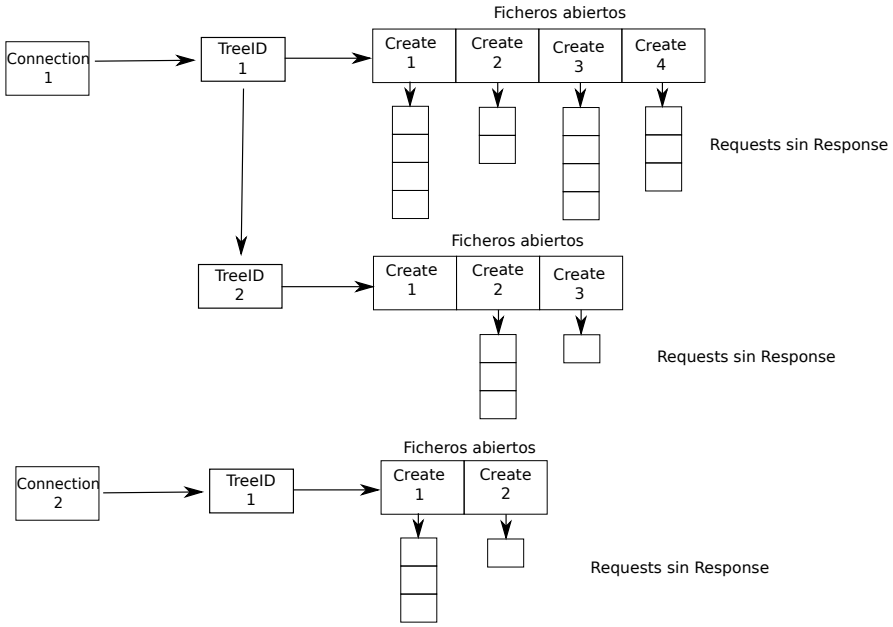



**Figura 24:** Estructuras de la herramienta

Se lleva una lista por cada conexión TCP abierta en la traza. Cada una apuntará a una lista de TreeIDs, identificados por el TreeID y el SessionID. Cada TreeID identifica la conexión de un cliente con un volumen compartido en el servidor. Como en cada uno de estos TreeIDs puede haber varios ficheros abiertos, cada uno apunta a una lista de CREATEs. Cada CREATE tiene que tener a su vez una lista con los mensajes que están pendientes del RESPONSE del servidor. Apuntará pues cada uno a una lista de REQUESTs, ya que cada uno de estos se identifica con el fileID del CREATE.

Los mensajes que se deben tener en cuenta para poder recuperar el archivo son estos cinco:

- CREATE REQUEST|RESPONSE: Para ver el nombre del fichero y ver si es el que se busca. En ese caso se añadirá al TreeID que le corresponda (si existe, si no se añade un nuevo TreeID). Es importante ver si el CREATE trunca el fichero o solo lo abre. Si lo trunca y el timestamp es menor que el del ransomware, el fichero recuperado en disco también debe truncarse. Si el timestamp es mayor que el del ransomware significa que es el propio ransomware el que lo está truncando y en ese caso no debe hacerse en el fichero recuperado.
- CLOSE REQUEST|RESPONSE: Simplemente para eliminar la lista asociada a ese FID (creada en el CREATE) y cerrar el fichero que permanece abierto en el sistema.

- READ REQUEST|RESPONSE: Ya se ha comentado en qué casos se deben volcar los datos del READ a disco. Estos datos se deben volcar al fichero en el momento en que llegue el RESPONSE, ya que hay que comprobar que el resultado ha sido *succeed*. En el momento en que se tienen la pareja REQUEST-RESPONSE, los datos se eliminan, ya se hayan pasado a disco o no.
- WRITE REQUEST|RESPONSE: Similar al caso del READ.
- SETINFO REQUEST|RESPONSE: Solo interesa el caso del renombrado. Si el fichero que se busca es renombrado, a partir de ese momento debemos tener en cuenta los CREATE de ficheros con ese nuevo nombre, por lo que tendremos que guardarlo. Igual que en casos anteriores, se tiene en cuenta cuando se ve en el RESPONSE que el resultado ha sido bueno.

A estos cinco tipos de mensaje hay que añadir el TREE_CONNECT, que aunque no es imprescindible, cuando llega uno nuevo, se crea una nueva estructura TreeID que apuntará a una lista de CREATEs. En el caso de que no se vea en la traza, cuando llegue un mensaje con un TreeID nuevo, se creará el objeto en el momento y ese mensaje se añadirá a la lista de CREATEs.

Es necesario crear una lista de los sectores que han sido escritos en el fichero por el ransomware, cada uno con su tamaño y su offset para que al llegar un READ podamos ver si hay bytes del mismo que ya han sido encriptados (escritos por el malware). Esta lista no puede ir asociada a un FileID en concreto, puesto que puede haber varios CREATEs apuntando al mismo fichero, incluso varios usuarios al mismo tiempo en distintas conexiones TCP modificándolo al mismo tiempo. Hay que crear una estructura global con el nombre del fichero y la lista de sectores asociada al mismo. Para dejar abierta la posibilidad de recuperar varios ficheros en la misma ejecución, se ha creado una lista global con estas estructuras. En principio habrá una sola estructura con el nombre del fichero a recuperar y sus sectores, pero en el caso de añadir más de un fichero para recuperar, esa lista tendrá más elementos.

# 8. Conclusiones

A lo largo de este documento se ha descrito y analizado a fondo el algoritmo REDFISH, que es capaz de detectar la presencia de un usuario infectado por un ransomware en una red con directorios compartidos por SMB. Se basa en el análisis del comportamiento de los usuarios en cuanto a ficheros que leen, escriben o eliminan, y a la frecuencia con que se realizan estas acciones. El tráfico se analiza gracias a un equipo que recibe una copia del tráfico entre cliente y servidor (de esta forma no introduce retardos en la red).

El escenario para el que está pensado el algoritmo son redes de empresas en las que haya uno o varios servidores almacenando archivos a los que los usuarios puedan acceder. El algoritmo analizará una copia del tráfico que parte y se dirige al servidor, sin ningún tipo de efecto sobre los usuarios.

Se ha analizado el algoritmo a fondo, y se han configurado una serie de parámetros para los que el algoritmo tiene unos resultados óptimos. Para esto se disponía de un día de tráfico en una red con alrededor de 300 usuarios, y una serie de muestras de ransomwares de 20 familias distintas.

Una vez que los parámetros son optimizados, tenemos un índice de detección del 100 % de las muestras, y en el 99 % de los casos se pierden solo 10 ficheros. Estos ficheros se pueden recuperar, gracias a la herramienta de recuperación que se ha explicado también en la parte final del documento. Por tanto, se consigue un 100 % de efectividad con 0 ficheros perdidos.

En cuanto a los falsos positivos, son muy poco probables. Ya se ha visto que solo hay uno entre cada 10 millones de archivos que se abren. Para estas mediciones se han utilizado trazas reales de una empresa con más de 4800 usuarios trabajando durante todo el día, accediendo a más de 1500 unidades de red compartidas.

Este algoritmo puede implementarse en una sonda de tráfico de 10Gbps usando un número bastante reducido de núcleos en un procesador. Además, no va a tener impacto alguno en la experiencia del usuario, puesto que no está instalada en su máquina local, y no consume recursos de la misma. Todo nuestro sistema se instalaría en una parte de la red separada de la que utilizan los usuarios, evitando que pueda ser atacado por cualquier otro malware que pueda desactivarlo.

Se trata pues de un algoritmo de detección de ransomware que va a detectar cualquier familia, saltando una alarma en ese momento y permitiendo bloquear el acceso o reducir los permisos de la víctima. Con un índice tan bajo de falsos positivos, la acción a tomar por el administrado de la red cuando se produzca una infección dependerá de la red en cuestión y la política de seguridad de la propia empresa.

Como todas las herramientas y los trabajos orientados a seguridad, el continuo avance en el desarrollo de malware puede provocar que determinadas herramientas se queden obsoletas y no sean capaces de detectar las variantes más actuales. En este caso, a pesar de que el ransomware es un malware especialmente cambiante, REDFISH analiza una serie de características ineludibles para el ransomware, de forma que cualquier familia será detectada perdiendo más o menos ficheros. Se podría pensar que algún ransomware podría evadir la detección bajando la velocidad de la encriptación, pero si bajara tanto la velocidad como para situarse por debajo del umbral de 107kbps, un directorio de por ejemplo 10GB (tamaño bastante conservador para un servidor en un entorno empresarial) tardaría en encriptarlo 217 horas. Este tiempo son varios días de trabajo, entre los cuales pueden realizarse copias de seguridad y analizarse para buscar cualquier tipo de malware.

Por último, se trata de una solución novedosa y original, dado que no existe ninguna herramienta que se base en el tráfico SMB para la detección de este tipo de malware. Los resultados no son peores que los vistos para herramientas instaladas en local (Sección 2) y los efectos sobre los usuarios van a ser claramente inferiores.

# A. Anexo I. Modelo analítico

En este anexo se va a desarrollar una evaluación analítica de la probabilidad de detección del ransomware y de la probabilidad de que se produzca un falso positivo.

A lo largo del trabajo se han visto resultados experimentales de detección y falsos positivos, pero ahora se va va a desarrollar un modelo matemático que permita estimar estos resultados sin necesidad de recurrir a muestras experimentales.

La detección del ransomware se basa en dos únicos eventos, que tienen que producirse simultáneamente:

- Evento $e_T$ : Se producen al menos $N$ eliminaciones en menos de $T$ segundos.
- Evento $e_{rw}$ : La velocidad a la que ha aumentado el mínimo entre los bytes leídos y escritos entre la primera y la última de las $N$ eliminaciones supera $V_{umbral}$.

Si se dan estos dos eventos cuando el tráfico es de un usuario normal, se producirá un falso positivo. En cambio, si estos dos eventos no ocurren cuando tenemos tráfico de ransomware, el algoritmo habrá sido incapaz de detectar la infección (falso negativo). Todavía podrá detectar el ransomware, pero perderá más de $N$ ficheros, que es el objetivo. Se busca minimizar la probabilidad de que se produzca una falsa alarma y maximizar la probabilidad de que el ransomware sea detectado cuando ha encriptado solo los $N$ primeros ficheros.

$\mathbb{P}(e_T)$ es la probabilidad de que ocurra el evento $e_T$ y $\mathbb{P}(e_{rw}/e_T)$ la probabilidad de que la velocidad de lectura y escritura de datos sea superior a $V_{thres}$ cuando se produce $e_T$. La alarma se producirá cuando se produzcan tanto $e_T$ como $e_{rw}$. Esto es igual a $\mathbb{P}(e_T \cap e_{rw})$. Se tomará como aproximación que los dos eventos $e_T$ y $e_{rw}$ son independientes, y se escoge como límite superior (peor caso) que $\mathbb{P}(e_{rw}) = 1$. Supuesto esto, y como se ve en la ecuación 17, se puede aproximar la probabilidad que buscamos como la probabilidad de $e_T$.

$$\mathbb{P}(e_T \cap e_{rw}) = \mathbb{P}(e_T)\mathbb{P}(e_{rw} \mid e_T) = \mathbb{P}(e_T)\mathbb{P}(e_{rw}) \leq \mathbb{P}(e_T) \tag{17}$$

Para que el evento $e_T$ se produzca, al menos tienen que eliminarse $N$ ficheros en menos de $T$ segundos. A continuación vamos a estimar la probabilidad de que el ransomware sea detectado cuando ha encriptado solo $N$ ficheros (el mínimo) y además calcularemos la probabilidad de que se produzca algún falso positivo en una población alta de usuarios.

## A.1. Probabilidad de detección con N ficheros

El comportamiento característico de un ransomware, como se ha visto a lo largo del documento, consiste en leer un fichero, escribir sus datos encriptados y luego borrar el original, aunque el último paso puede no hacerlo si la versión encriptada sobreescribe la original. En cualquier caso, los datos originales de los ficheros siempre se pierden, y es que este es el objetivo principal de los ransomwares, encriptarlos para luego pedir un rescate por su recuperación. Los ficheros son encriptados secuencialmente, por lo que se crea una serie de eventos de eliminación $\{\tau_k\}$, donde se puede definir $\{t_k\}$ como el tiempo entre eliminaciones. Este tiempo es el que tarda el ransomware en leer, encriptar y escribir cada uno de los ficheros.

Esta variable $t_k$ tiene que depender del tamaño de los archivos, porque es el tiempo que tarda en leer, encriptar y escribir, y todas estas acciones dependen de este tamaño. Además, cuanto mayor es un fichero más tiempo va a tardar en transferirse por la red desde el usuario al servidor y vuelta.

Como se ha comentado en la sección 3.3, se ha utilizado una distribución de probabilidad específica para los tamaños de los ficheros en el directorio compartido, tratando de emular el contenido típico

en un disco [23]. Cada fichero $k$ tiene un tamaño $s_k$, donde todas las $s_k$ son variables independientes e idénticamente distribuidas (i.i.d). El ransomware normalmente va a recorrer el árbol de directorios, analizando desde dónde puede comenzar la encriptación [61]. Aunque cada $s_k$ es una combinación de una variable aleatoria lognormal y una cola de Pareto, se va a aproximar simplemente como una variable lognormal, ya que es así como se generan más del 99.99 % de los valores [23]. La ecuación 18 define la función de densidad de probabilidad (PDF) para una distribución lognormal, que está basada en dos parámetros, $\mu$ y $\sigma$.

$$f_{LN}(x) = \frac{1}{\sigma x\sqrt{2\pi}} e^{\frac{-(\ln x-\mu)^2}{2\sigma^2}} \tag{18}$$

Se parte de una la variable aleatoria $t_k$, relacionada con el tamaño del fichero y con la velocidad de transferencia de la red $s$. Las variables aleatorias resultantes de $t_k = sk/s$ son variables aleatorias i.i.d, y siguen una distribución de probabilidad lognormal. El parámetro $s$ depende de la velocidad de la red entre el usuario y el servidor. Está determinado por la capacidad de la red, la velocidad de procesado del usuario, la velocidad de acceso a disco del servidor, el número de usuarios accediendo al mismo simultáneamente y los parámetros de los protocolos TCP y SMB (tamaños de ventanas, algoritmos *congestion avoidance*, etc.). Hoy en día la velocidad de acceso a los ficheros en las redes empresariales es de cientos de megabits por segundo. Velocidades menores dificultarían mucho a los usuarios el trabajo en directorios compartidos; por eso es complicado ver empresas cuyas redes de área local no lleguen a esas velocidades de transferencia. Una empresa típica con aplicaciones o ficheros en volúmenes compartidos en la red local, tiene enlaces de 10Gb/s [1, 59], y a pesar de esto vamos a suponer un caso extremo de red con poca capacidad (rango de 10Mb/s a 100Mb/s por usuario) dado que es el caso peor para el algoritmo propuesto.

En este escenario, $\Delta_N \tau_k$ es la suma de $N$ variables aleatorias lognormales i.i.d y por tanto su PDF es la convolución de $N$ PDFs lognormales. Como no hay una forma conocida para esta convolución, la PDF se puede calcular también como el producto de las $N$ funciones características de las variables aleatorias lognormales. Sin embargo, la función característica de la distribución lognormal tampoco es conocida. La literatura sobre este tema aporta muchas publicaciones con diferentes aproximaciones para calcular la suma de $N$ variables lognormales [71, 28, 56, 48]. Para este trabajo se ha seleccionado el procedimiento ofrecido en [71], basado en la familia de funciones de Pearson.

Partiendo de los parámetros de la lognormal recomendados en [23], la PDF de la distribución seleccionada se puede ver en la ecuación 19, donde los parámetros $u$, $d$, $m$, y $v$ son calculados a partir de las muestras de la lognormal y $k$ es el factor de escala para una probabilidad total acumulada de 1.

$$f_{IV}(x) = k\left(1 + \frac{(x+u)^2}{d^2}\right)^{-m} e^{-v\tan^{-1}(\frac{x+u}{d})} \tag{19}$$

El procedimiento para la estimación comienza con la distribución lognormal de 1 millón de tamaños sacados del algoritmo de generación de directorios visto en [23]. Calculando la media y la desviación típica de estos tamaños, podemos sacar la media y la varianza de la distribución normal asociada a esta lognormal con Matlab como se explica en [5].

A continuación se genera la variable aleatoria como suma de 10 valores de las muestras de los tamaños que tenemos en la distribución, se pasan a logaritmo decimal y se sacan los cuatro primeros momentos como se indica en [71]. Con estos cuatro momentos podemos calcular $\beta_1$, $\beta_2$ y $\kappa$. Ahora utilizando la Pearson IV (ver ecuación 19) debido al valor obtenido para $\kappa$ (0.2099) y a la tabla II de [71]. Como vemos en 19, es necesario calcular los valores de $u$, $d$, $m$ y $v$ para obtener la expresión Pearson IV. Estos valores se calculan a su vez con las ecuaciones 7 y 8 de [71].

Los valores obtenidos aplicando estas ecuaciones son:

- $u = -51.7517$
- $m = 7.32193$
- $d = 16.1771$
- $v = -6.51776$

Por último, para sacar la PDF de la distribución hay que normalizar la ecuación 19. Para esto se observa en un gráfico el rango en el que se encuentra la práctica totalidad de la masa en la PDF y se hace la integral numérica de la misma, obteniendo un valor de renormalización de 50.1152. Ya se ha obtenido la función de distribución del tamaño de la suma de 10 ficheros con tamaño lognormal (Figura 25.

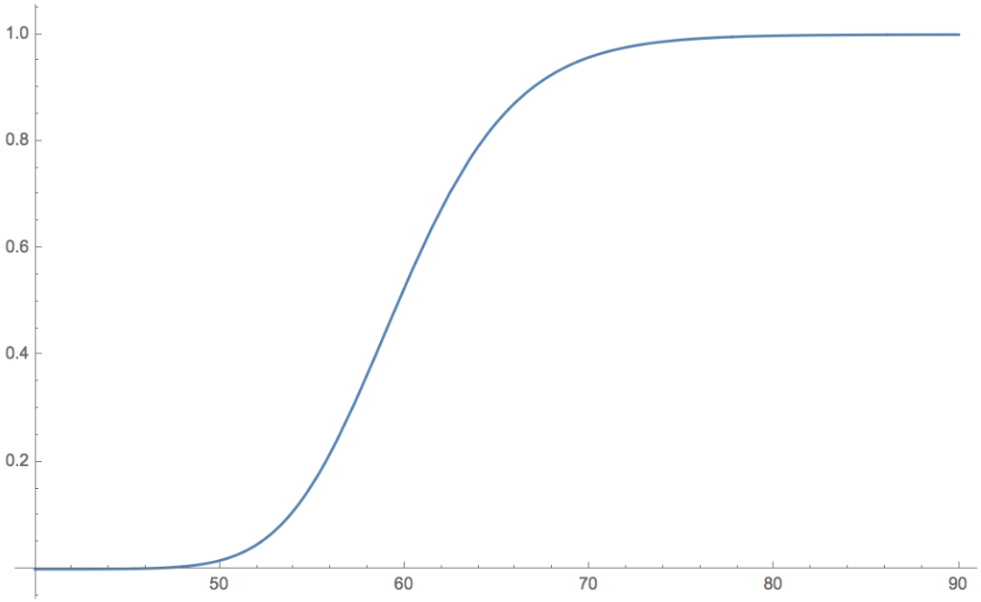



**Figura 25:** Distribución de probabilidad del tamaño de 10 ficheros

En esta figura podemos ver que en el 99 % de las veces, $x < 75$. Hay que tener en cuenta que debido a las transformaciones hechas al inicio de la aproximación, será necesario deshacer el algoritmo decimal en el eje X, quedando el nuevo eje como $X' = 10^{X/10}$. Por tanto, el 99 % de las veces podemos decir que los 10 ficheros no superan los 32Mbytes de tamaño. Por lo tanto con velocidades mayores que 12.65 Mb/s sale un tiempo de transferencia menor que 20 segundos y por lo tanto la alarma saltaría perdiendo solo 10 ficheros.

Para dibujar la Figura 26, se ha calculado la probabilidad para distintas velocidades de red (de 10 a 100Mb/s), es decir, con 10 Mb/s en 20 segundos se pueden transferir 25 Mbytes (lo máximo que pueden sumar los tamaños de los 10 ficheros). Pasándolo a los valores del eje X de la Figura 25, $10 * \log(25M)$ que es 74. Tenemos una probabilidad de 98.6 % de que sean menores y por tanto de que salte la alarma. Con estos datos ya se puede generar la figura 26.

La probabilidad de que se pierdan más de $N$ ficheros es menor de 0.01 para velocidades de transferencia mayores de 12.7 Mb/s. Para una red de 100Mb/s la probabilidad de no detectar el ransomware cuando se pierde el décimo fichero es menor de 0.001 (0.1 %).

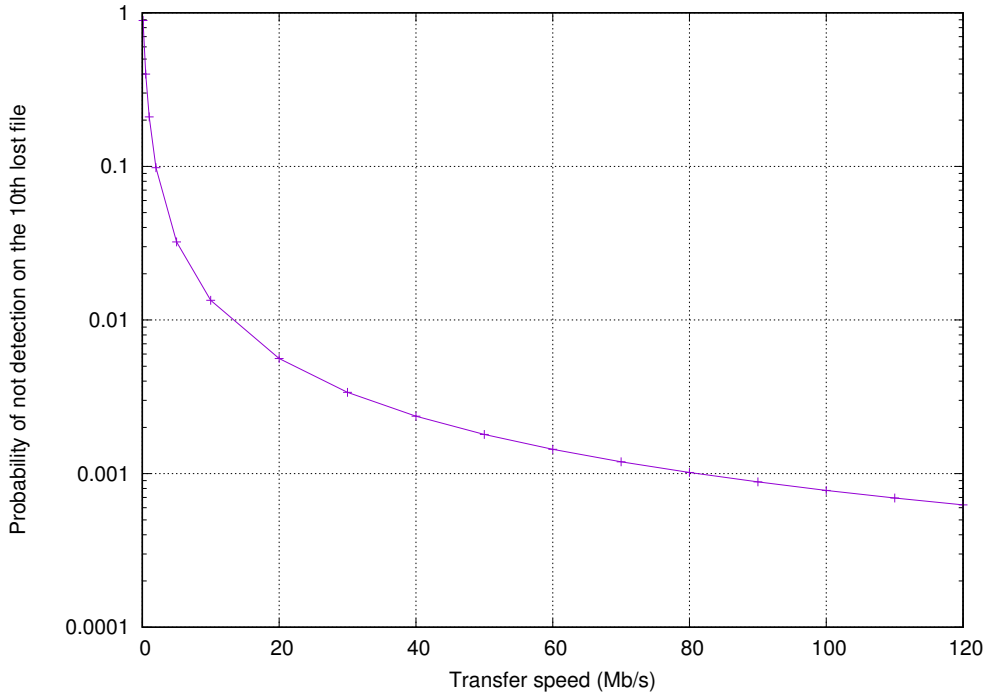



**Figura 26:** Resultados analíticos de la probabilidad de detección de ransomware perdiendo $N$ ficheros

Hay que destacar que este último resultado no significa que el ransomware no sea detectado en el 0.1 % de los casos, sino que no será detectado cuando se pierde el décimo fichero, pudiendo serlo cuando haya encriptado algún fichero más. Dicho esto, como la probabilidad de que un ransomware evada la detección en el décimo evento de borrado es 0.001, entonces la probabilidad de que se detecte cuando lleva 20 ficheros es $1-(0.001)^2 = 0.999999$, o más de "cinco nueves", calidad requerida típicamente en redes de telecomunicación.

Dicho esto, hay que tener en cuenta que la velocidad representada en el eje X se trata de la velocidad efectiva de transferencia de datos sobre TCP/IP con el protocolo SMB. Comprobando la velocidad lograda en los experimentos y probando a limitar la velocidad en la red entre cliente y servidor se observa una diferencia notable entre la velocidad del enlace y la velocidad efectiva de transferencia de datos. Así este cálculo se trata de una aproximación bastante optimista del índice de detección, aunque siempre se puede recurrir a los datos experimentales para observar los resultados de forma empírica.

## A.2. Modelo de probabilidad de falsos positivos

Para que se produzca un falso positivo, un usuario no infectado (sin ningún ransomware corriendo en su equipo) realiza una serie de eventos de eliminación cercanos temporalmente. Además, debe haber una cantidad mínima de bytes leídos y escritos por el mismo, superando el umbral de velocidad ($V_{umbral}$) fijado para REDFISH. Para estimar $\mathbb{P}(e_T)$, es necesario modelar la distribución de probabilidad del tiempo entre eliminaciones de ficheros en el comportamiento de usuario.

En la sección 4.2 se ha modelado este comportamiento, aproximándolo a una distribución de probabilidad Weibull. En la figura 9 se puede ver cada una de estas aproximaciones para cada día, junto con los parámetros de la distribución.

Una distribución Weibull presenta la función de distribución acumulada que se muestra en la ecuación 20, donde $\alpha$ y $\beta$ son los parámetros de la distribución aproximados en la sección 4.2.

$$\mathbb{P}(t_k > x) = 1 - e^{-(\frac{x}{\beta})^{\alpha}}, x \geq 0 \tag{20}$$

Partiendo de [69] se puede calcular la distribución para $\Delta_N \tau_k$ como la suma de $N$ variables i.i.d Weibull. El resultado se aproxima utilizando la ecuación 21 de [41].

$$\begin{aligned} \mathbb{P}(e_T) = \mathbb{P}(\Delta_N \tau_k < T) = \mathbb{P}\left(\sum_{i=k-N+1}^{k-1} t_i < T\right) \approx 1 - e^{-(\frac{p}{\beta}T)^{\alpha}} \sum_{k=0}^{N-2} \frac{1}{k!}\left(\frac{c}{\beta}T\right)^{\alpha k} = \\ = 1 - \frac{\Gamma(N-1, (\frac{c}{\beta}T)^{\alpha})}{\Gamma(N-1)} \end{aligned} \tag{21}$$

$\Gamma(n)$ es la función gamma, $\Gamma(n, x)$ es la función gamma incompleta y $c$ es calculada usando la ecuación 22.

$$c = \frac{\Gamma(N - 1 + \frac{1}{\alpha})}{(N-1)!\Gamma(1 + \frac{1}{\alpha})} \tag{22}$$

Para cada uno de los días de las trazas de usuario, se ha aproximado una distribución gracias al método de mínimos cuadrados visto en la sección 4.2. Cada uno de los valores para cada día se pueden ver en el cuadro 6

| **Día** | $\alpha$ | $\beta$ | $c$ | $\mathbb{P}(e_T)$ | $p_{day}$ | **tiempo hasta primera alarma** |
|---|---|---|---|---|---|---|
| **Día 22** | 0.3859 | 150.02 | 11.09 | 0.0000037 | 0.0062 (0.62 %) | 161.54 días |
| **Día 23** | 0.3549 | 102.38 | 14.71 | 0.000021 | 0.049 (4.9 %) | 21 días |
| **Día 24** | 0.3727 | 144.54 | 12.45 | 0.0000057 | 0.0053 (0.53 %) | 191 días |
| **Día 27** | 0.3745 | 147.40 | 12.25 | 0.0000051 | 0.0063 (0.63 %) | 160 días |
| **Día 28** | 0.3694 | 127.25 | 12.84 | 0.0000089 | 0.019 (1.9 %) | 53 días |
| **Día 1** | 0.3809 | 142.18 | 11.58 | 0.000005 | 0.0078 (0.78 %) | 128 días |

**Tabla 6:** Resultados de la distribución Weibull para cada día

$\mathbb{P}(e_T) = 0.000032$ es la probabilidad de que se produzcan $N = 10$ eventos de eliminación consecutivos y en un intervalo menor de $T = 20$ segundos. Cada usuario tiene muchas oportunidades de provocar un salto de la alarma durante un único día, porque pueden producirse muchos conjuntos de $N$ eliminaciones, y por cada uno significa una nueva oportunidad. Si cada usuario elimina $M$ ficheros durante un día, se puede estimar que hay un número máximo de oportunidades para que salte la alarma de $opts = \text{mín}\{0, \lfloor M - N + 1 \rfloor\}$. Asumiendo que se trata de eventos independientes, la probabilidad de que haya una o más de una alarmas en un día para un usuario ($p_{user}$) es el resultado de la ecuación 23.

$$p_{user} = \mathbb{P}(\text{alarmas por usuario y día} \geq 1) = 1 - (1 - \mathbb{P}(e_T))^{opts} \tag{23}$$

Se ha estimado M del comportamiento de más de 300 usuarios en las trazas de entrenamiento. El resultado de la probabilidad de que haya una falsa alarma en un día se puede ver en el cuadro 6, así como el tiempo que pasará hasta que se produzca la primera alarma.

# B. Anexo II. Extensiones objetivo de los ransomwares

Las extensiones susceptibles de ser encriptadas son las que se listan a continuación. Se han obtenido de tres fuentes distintas, ya que había extensiones que no estaban en alguna de ellas por tratarse de listas para ransomwares específicos. Las extensiones son las siguientes.

*txt, doc, dot, docx, docm, cotx, dotm, xml, xls, xlt, xlm, xlsx, xlsm, xltx, xltm, xlsb, xla, xlam, xll, xlw, ppt, pot, pptx, pptm, potx, potm, ppam, ppsx, ppsm, sldx, sldm, accdb, accde, accdt, accdr, rtf, csv, odb, odt, ott, oth, odm, sxw, stw, sxg, ods, ots, sxc, stc, odp, odg, otp, sxi, sti, otg, sxd, std, odf, sxm, pdf, djvu, ab4, ac2, acr, bgt, bpw, cdf, cfp, dac, ddd, dgc, ffd, hbk, ibank, mmw, moneywell, bkp, bak, backup, bik, backupdb, accdb, adb, al, bdb, cis, db, db-journal, dbf, db3, erbsql, fdb, ibd, ibz, idx, dbx, kpdx, myd, ns2, ns3, ns4, nsd, nsf, nsg, nsh, nx1, nx2, nyf, psafe3, rdb, s3db, sas7bdat, sav, sdf, sql, sqlite, sqlite3, sqlitedb, mdb, jpg, jpeg, mpg, 3fr, 3pr, arw, ce1, ce2, cib, cmt, cmt, cr2, craw, crw, dc2, dcr, dng, erf, exc, fff, fpx, gray, grey, gry, iiq, kc2, kdc, mdc, mef, mfw, mos, mrw, ndd, nef, nop, nrw, nwb, orf, pcd, pef, ptx, ra2, raf, raw, rw2, rwl, rwz, sd0, sd1, sr2, srf, st4, st5, st6, st7, st8, stx, x3f, ycbcra, agd1, ai, ait, awg, cdr, cdr3, cdr4, cdr5, cdr6, cdrw, cdx, cgm, cpi, csh, csi, dcs, ddoc, ddrw, design, drw, dxb, fh, fhd, fxg, pat, ps, sda, dwg, 3ds, apj, blend, drf, rar, zip, 7z, c, cpp, h, hpp, asm, incpas, php, asp, js, css, lua, ppy, pl, der, cer, crt, pem, pfx, p12, p7b, p7c, psd, wb2, mid, wma, flv, mkv, mov, avi, asf, mpeg, vob, mpg, wmv, fla, swf, wav, qcow2, vdi, vmdk, vmx, gpg, aes, arc, paq, bz2, tbk, bak, tar, tgz, rar, djv, djvu, svg, bmp, png, gif, raw, cgm, tif, tiff, nef, psd, cmd, bat, class, jar, java, asp, brd, sch, dch, dip, vbs, asm, pas, cpp, php, ldf, mdf, ibd, myi, myd, frm, odb, dbf, mdb, asc, lay6, lay, ms11, sldm, sldx, ppam, docb, mml, sxm, otg, odg, uop, pps, sti, sxi, otp, odp, wks, slk, xlw, xlt, xlm, xlc, dif, hwp, max, uot, csr, key, dat, yuv, x11, wpd, tex, srw, rat, qby, qbx, qbw, qbr, qba, plc, plus_muhd, pdd, nxl, nx2, nk2, mny, kpdx, kdc, kdbx, jpe, liq, exf, eml, dxg, des, csl, bay, bank, back, ads, ach, vmxf, vmsd, vhdx, vhd, vbox, stm, rvt, qcow, qed, pif, pdb, pab, ost, ogg, nvram, ndf, m2ts, hdd, groups, flvv, edb, dit, bin, aiff, xlk, wad, tlg, say, qbm, qbb, oil, nd, indd, iif, dtd, adp, act, xlr, wps, tga, r3d, pspimage, pct, m4v, flac, eps, cls, aac, thm, srt, save, safe, rm, pwm, pages, obj, m1b, md, mbx, lit, laccdb, kwm, html, flf, dxf, dds, config, cfg, asx, aspx, aoi, 7zip, 1cd, wab, prf, oab, msg, mapimail, jnt, contact, n64, m4a, m4u, m3u, 3gp, mp4, mov, mp3, wallet, upk, re4, ltx, litesql, litemod, lbf, iwi, forge, das, d3dbsp, bsa, bik, asset, apk, 7z, rb, cs, vb, 011, 010, 009, 008, 007, 006, 005, 004, 003, 002, 001, 123, 602.*

Estas extensiones son una combinación de las que se mencionan en [21, 22, 51]. Además, en [21] se menciona que el ransomware Locky, no encriptará ficheros cuyo path completo incluya alguno de los siguientes strings: *tmp, winnt, Application Data, AppData, Program Files (x86), Program Files, temp, thumbs.db, $Recycle.Bin, System Volume Information, Boot, Windows.* Esto confirma nuestras sospechas de que hay ransomwares que no van a atacar ficheros de programas ya sea porque no encripta determinados directorios o porque no lo hace con algunas extensiones. En cualquier caso y como se ha comentado en secciones del documento, el escenario para el que se ha pensado la herramienta es aquel que guarda ficheros de usuario en el servidor.

# *Referencias*


[1] *Isilon All-Flash Scale-Out NAS Storage. Isilon F800 Specification Sheet.*
Informe técnico., DELL EMC.
`https://www.supernaeyeglass.com/ransomware-defender`.

[2] *Cybercrime tactics and techniques Q1 2017.*
Informe técnico., Malware Bytes.
`https://www.malwarebytes.com/pdf/labs/Cybercrime-Tactics-and-Techniques-Q1-2017.pdf`.

[3] *Hybrid Analysis.*
`https://www.hybrid-analysis.com/`, Último acceso Enero de 2018.

[4] *Malware Traffic Analysis.*
`http://www.malware-traffic-analysis.net`, Último acceso Enero de 2018.

[5] *MathWorks documentation. lognstat.*
`https://es.mathworks.com/help/stats/lognstat.html?s_tid=gn_loc_drop`.

[6] *Microsoft Corporation, Common Internet File System (CIFS) protocol.*
`https://msdn.microsoft.com/en-us/library/ee442092.aspx`, Último acceso Enero de 2018.

[7] *Microsoft Corporation, Server Message Block (SMB) Protocol versions 2 and 3.*
`https://msdn.microsoft.com/en-us/library/cc246482.aspx`.

[8] *tcpflow - Linux man page.*
`https://linux.die.net/man/1/tcpflow`.

[9] *Ransomware and Businesses 2016.*
Informe técnico., Symantec Corporation, 2016.
`http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/ISTR2016_Ransomware_and_Businesses.pdf`.

[10] *Umbrella, the Cisco's software to increase protection against ransomwares.*
Informe técnico., Cisco Systems, 2016.
`https://www.cisco.com/c/dam/m/hr_hr/training-events/2017/cisco-connect/pdf/Introducing_Cisco_Umbrella_for_cloud_based_threat_protection.pdf`.

[11] *Understanding the Depth of the Global Ransomware Problem.*
Informe técnico., Osterman Research, Inc., August 2016.
`https://www.malwarebytes.com/pdf/white-papers/UnderstandingTheDepthOfRansomwareIntheUS.pdf`.

[12] *CryptoStopper. WhatchPoint's solution for ransomware detection*, 2017.
`https://www.watchpointdata.com/cryptostopper/`, Último acceso Enero de 2018.

[13] *KSN Report: Ransomware in 2016-2017*, 2017.
`https://securelist.com/ksn-report-ransomware-in-2016-2017/78824/`, "Último acceso 20 de Mayo de 2018".

[14] *onQ Ransomware Edition*, 2017.
`https://quorum.com/resources/onq-ransomware-edition`, Último acceso Mayo de 2018.

[15] *Operating Systems - Statistics and Facts*, 2017.
`https://www.statista.com/topics/1003/operating-systems/`, Último acceso Mayo de 2018.

[16] *Paying the Price of Destructive Cyber Attacks*, 2017.
https://hi.cybereason.com/hubfs/Content%20PDFs/Paying-the-Price-of-Destructive-Cyber-Attacks.pdf?t=1526482908645, "Último acceso 20 de Mayo de 2018".

[17] *Petya: Nuevo ataque global del ransomware.* , 2017.
https://www.pandasecurity.com/spain/mediacenter/malware/petya-ataque-ransomware/, "Último acceso 20 de Mayo de 2018".

[18] *Ransomware Bundle v1.2.6*, 2017.
https://docs.extrahop.com/current/walkthrough-bundle-ransomware/#appendix, Último acceso Abril de 2018.

[19] *GandCrab, el ransomware que se ha cobrado más de 50.000 víctimas en dos meses*, 2018.
https://bitlifemedia.com/2018/03/gandcrab-ransomware-50-000-victimas/, "Último acceso 20 de Mayo de 2018".

[20] G. A.: *Ransomware and the GDPR.*
Network Security, 3:18–19, 2017.
https://doi.org/10.1016/S1353-4858(17)30030-2.

[21] L. Abrams: *The Locky ransomware Encrypts Local Files and Unmapped Network Shares*, February 2016.
https://www.bleepingcomputer.com/news/security/the-locky-ransomware-encrypts-local-files-and-unmapped-network-shares/, Último acceso Noviembre de 2017.

[22] L. Abrams: *Lawrence Abrams. Locky Ransomware now using the Aesir Extension for Encrypted Files*, November 2017.
https://www.bleepingcomputer.com/news/security/locky-ransomware-now-using-the-aesir-extension-for-encrypted-files/, Último acceso Noviembre de 2017.

[23] N. Agrawal, A. C. Arpaci-Dusseau y R. H. Arpaci-Dusseau: *Generating realistic impressions for file-system benchmarking.*
ACM Transactions on Storage, 5(4):1–30, dec 2009.
https://doi.org/10.1145/1629080.1629086.

[24] M. M. Ahmadian y H. R. Shahriari: *2entFOX: A framework for high survivable ransomwares detection.*
En *2016 13th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC)*, págs. 79–84, Sept 2016.

[25] M. M. Ahmadian, H. R. Shahriari y S. M. Ghaffarian: *Connection-monitor & connection-breaker: A novel approach for prevention and detection of high survivable ransomwares.*
En *2015 12th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC)*, págs. 79–84. IEEE, sep 2015.
https://doi.org/10.1109/iscisc.2015.7387902.

[26] M. Alam, S. Bhattacharya, D. Mukhopadhyay y A. Chattopadhyay: *RAPPER: Ransomware Prevention via Performance Counters.*
CoRR, abs/1802.03909, 2018.
http://arxiv.org/abs/1802.03909.

[27] J. Barreto: *Windows Server 2012 R2: Which version of the SMB protocol (SMB 1.0, SMB 2.0, SMB 2.1, SMB 3.0 or SMB 3.02) are you using?*, 2013.

https://blogs.technet.microsoft.com/josebda/2013/10/02/windows-server-2012-r2-which-version-of-the-smb-protocol-smb-1-0-smb-2-0-smb-2-1-smb-3-0-or-smb-3-02-are-you-using/, Último acceso Mayo de 2018.

[28] N. Beaulieu y F. Rajwani: *Highly Accurate Simple Closed-Form Approximations to Lognormal Sum Distributions and Densities.*
IEEE Communications Letters, 8(12):709–711, dec 2004.
https://doi.org/10.1109/lcomm.2004.837657.

[29] R. Brandom: *Ransomware victims have paid out more than 25 million dolars, Google study finds*, 2017.
https://www.theverge.com/2017/7/25/16023920/ransomware-statistics-locky-cerber-google-research, "Último acceso 10 de Mayo de 2018".

[30] K. Cabaj y W. Mazurczyk: *Using Software-Defined Networking for ransomware mitigation: The case of CryptoWall.*
IEEE Network, 30(6):14–20, November 2016, ISSN 0890-8044.

[31] Cisco Systems: *2016 Midyear Cybersecurity Report.*
Informe técnico., Cisco Systems, 2016.
https://www.cisco.com/c/dam/m/en_ca/never-better/assets/files/midyear-security-report-2016.pdf.

[32] L. Constantin: *TeslaCrypt victims can now decrypt their files for free*, 2016.
https://www.csoonline.com/article/3072461/security/teslacrypt-victims-can-now-decrypt-their-files-for-free.html, "Último acceso 15 de Mayo de 2018".

[33] A. Continella, A. Guagnelli, G. Zingaro, G. D. Pasquale, A. Barenghi, S. Zanero y F. Maggi: *ShieldFS: a self-healing, ransomware-aware filesystem.*
En *Proceedings of the 32nd Annual Conference on Computer Security Applications - ACSAC 16*. ACM Press, 2016.
https://doi.org/10.1145/2991079.2991110.

[34] A. Drozhzhin: *Historias y evolución del ransomware: datos y cifras.*
https://www.kaspersky.es/blog/ransomware-blocker-to-cryptor/8526/, Último acceso Abril de 2018.

[35] P. Ducklin: *Destructive malware "CryptoLocker" on the Loose - here's what to do*, 2013.
https://nakedsecurity.sophos.com/2013/10/12/destructive-malware-cryptolocker-on-the-loose/, "Último acceso 15 de Mayo de 2018".

[36] P. Ducklin: *Destructive malware "CryptoLocker" on the Loose - here's what to do*, 2013.
https://nakedsecurity.sophos.com/2013/10/12/destructive-malware-cryptolocker-on-the-loose/, "Último acceso 15 de Mayo de 2018".

[37] EUROPOL: *Internet Organised Crime Thread Assessment (IOCTA) 2016.*
Informe técnico., Europol - European Police Office, 2016.
https://doi.org/10.2813/275589.

[38] EUROPOL: *Internet Organised Crime Thread Assessment (IOCTA) 2017.*
Informe técnico., Europol - European Police Office, 2017.

[39] A. Gendre: *Ransomware Statistics 2017*, 2017.
https://www.vadesecure.com/en/ransomware-statistics-2017/, "Último acceso 9 de Mayo de 2018".

[40] M. M. Hasan y M. M. Rahman: *RansHunt: A support vector machines based ransomware analysis framework with integrated feature set.*
En *2017 20th International Conference of Computer and Information Technology (ICCIT)*, págs. 1–7, Dec 2017.

[41] L. Johnson: *GMR Reliability Manual.*
Informe técnico., General Motors Research Labs, 1960.

[42] E. Kalaimannan, S. K. John, T. DuBose y A. Pinto: *Influences on ransomware's evolution and predictions for the future challenges.*
Journal of Cyber Security Technology, 1(1):23–31, 2017.
https://doi.org/10.1080/23742917.2016.1252191.

[43] A. Kharraz y E. Kirda: *Redemption: Real-Time Protection Against Ransomware at End-Hosts.*
En *Research in Attacks, Intrusions, and Defenses*, págs. 98–119. Springer International Publishing, 2017.
https://doi.org/10.1007/978-3-319-66332-6_5.

[44] A. Kharraz, W. Robertson, D. Balzarotti, L. Bilge y E. Kirda: *Cutting the Gordian Knot: A Look Under the Hood of Ransomware Attacks.*
En *Detection of Intrusions and Malware, and Vulnerability Assessment*, págs. 3–24. Springer International Publishing, 2015.
https://doi.org/10.1007/978-3-319-20550-2_1.

[45] H. Kim, D. Yoo, J. S. Kang y Y. Yeom: *Dynamic ransomware protection using deterministic random bit generator.*
En *2017 IEEE Conference on Application, Information and Network Security (AINS)*, págs. 64–68, Nov 2017.

[46] E. Kirda: *UNVEIL: A large-scale, automated approach to detecting ransomware.*
En *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, feb 2017.
https://doi.org/10.1109/saner.2017.7884603.

[47] E. Kolodenker, W. Koch, G. Stringhini y M. Egele: *PayBreak.*
En *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security - ASIA CCS - 17*. ACM Press, 2017.
https://doi.org/10.1145/3052973.3053035.

[48] C. L. Lam y T. Le-Ngoc: *Estimation of typical sum of lognormal random variables using log shifted gamma approximation.*
IEEE Communications Letters, 10(4):234–235, apr 2006.
https://doi.org/10.1109/lcomm.2006.1613731.

[49] M. Lee, W. Mercer, P. Rascagneres y C. Williams: *Player 3 Has Entered the Game: Say Hello to 'WannaCry'*, 2017.
http://blog.talosintelligence.com/2017/05/wannacry.html, Último acceso Febrero de 2018.

[50] T. Lu, L. Zhang, S. Wang y Q. Gong: *Ransomware detection based on V-detector negative selection algorithm.*
En *2017 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC)*, págs. 531–536, Dec 2017.

[51] M. E. M. Lvéillé: *TorrentLocker*, December 2014.
https://www.welivesecurity.com/wp-content/uploads/2014/12/torrent_locker.pdf.

[52] J. D. Mathieu Rosemain, Yann le Guernigou: *Renault, Nissan European operations deal with global cyber attack.*
`http://www.autonews.com/article/20170513/OEM01/170519882/renault-nissan-european-operations-deal-with-global-cyber-attack`, "Último acceso Enero de 2018".

[53] F. Mbol, J. M. Robert y A. Sadighian: *An efficient approach to detect torrentlocker ransomware in computer systems.*
En *International Conference on Cryptology and Network Security*, págs. 532–541. Springer, 2016.

[54] T. Micro: *Ransom Notes: Know what ransomware hit you.*
`https://www.trendmicro.com/vinfo/us/security/news/cybercrime-and-digital-threats/ransom-notes-know-what-ransomware-hit-you`, Último acceso Abril de 2018.

[55] Microsoft Corporation: *File System Behavior in the Microsoft Windows Environment.*
Informe técnico., Microsoft Corporation.
`http://download.microsoft.com/download/4/3/8/43889780-8d45-4b2e-9d3a-c696a890309f/File%20System%20Behavior%20Overview.pdf`.

[56] H. Nie y S. Chen: *Lognormal Sum Approximation with Type IV Pearson Distribution.*
IEEE Communications Letters, 11(10):790–792, oct 2007.
`https://doi.org/10.1109/lcomm.2007.070842`.

[57] D. Nieuwenhuizen: *A behavioural-based approach to ransomware detection. MWR Labs Whitepaper.*
Informe técnico., MWR Labs, 2016.
`https://info.varonis.com/hubfs/docs/whitepapers/en/Varonis-Ransomware-Whitepaper-Netapp.pdf`.

[58] D. Palmer: *Petya ransomware: Cyberattack costs could hit 300 million dollars for shipping giant Marsk.* , 2017.
`https://www.zdnet.com/article/petya-ransomware-cyber-attack-costs-could-hit-300m-for-shipping-giant-maersk/`, "Último acceso 14 de Mayo de 2018".

[59] QNAP: *TVS-EC2480U-SAS-RP R2.*
`https://www.qnap.com/en/product/tvs-ec2480u-sas-rp`, Último acceso Enero de 2018.

[60] F. Quinkert, T. Holz, K. S. M. T. Hossain, E. Ferrara y K. Lerman: *RAPTOR: Ransomware Attack PredicTOR.*
CoRR, abs/1803.01598, 2018.
`http://arxiv.org/abs/1803.01598`.

[61] N. Scaife, H. Carter, P. Traynor y K. R. B. Butler: *CryptoLock (and Drop It): Stopping Ransomware Attacks on User Data.*
En *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, págs. 303–312, June 2016.
`https://doi.org/10.1109/ICDCS.2016.46`.

[62] K. Selvaraj, E. Florio, A. Lelli y T. Ganacharya: *WannaCrypt ransomware worm targets out-of-date systems*, 2017.
`https://blogs.technet.microsoft.com/mmpc/2017/05/12/wannacrypt-ransomware-worm-targets-out-of-date-systems/`, Último acceso Febrero de 2018.

[63] D. Sgandurra, L. Muñoz-González, R. Mohsen y E. C. Lupu: *Automated Dynamic Analysis of Ransomware: Benefits, Limitations and use for Detection.*

CoRR, abs/1609.03020, 2016.
http://arxiv.org/abs/1609.03020.

[64] M. Shukla, S. Mondal y S. Lodha: *POSTER: Locally Virtualized Environment for Mitigating Ransomware Threat.*
En *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security - CCS'16*. ACM Press, 2016.
https://doi.org/10.1145/2976749.2989051.

[65] S. Sjouwerman: *New Ransomware CryptoFortress Encrypts Unmapped Network Shares*, 2017.
https://blog.knowbe4.com/new-ransomware-cryptofortress-encrypts-unmapped-network-shares, Último acceso Febrero de 2018.

[66] K. Terashita y R. D. Paz: *Cerber Ransomware Marks Its Presence in the Wild, Catches up with CryptoWall and Locky*, 2016.
https://www.fortinet.com/blog/threat-research/cerber-ransomware-marks-its-presence-in-the-wild-catches-up-with-cryptowall-and-locky.html, "Último acceso 13 de Mayo de 2018".

[67] G. Villalobos: *Gigantes europeos confirmaron pérdidas multimillonarias tras impacto de los virus Petya y WannaCry*, 2017.
https://www.criptonoticias.com/sucesos/gigantes-europeos-confirmaron-perdidas-multimillonarias-tras-impacto-virus-petya-wannacry/, "Último acceso 10 de Mayo de 2018".

[68] R. Vinayakumar, K. P. Soman, K. K. S. Velan y S. Ganorkar: *Evaluating shallow and deep networks for ransomware detection and classification.*
En *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, págs. 259–265, Sept 2017.

[69] F. Yilmaz y M. S. Alouini: *Sum of Weibull variates and performance of diversity systems.*
En *Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing Connecting the World Wirelessly - IWCMC'09*. ACM Press, 2009.
https://doi.org/10.1145/1582379.1582434.

[70] B. L. Yun Feng, Chaoge Liu: *Poster: A new approach to detecting ransomware with Deception.*
En *38th IEEE Symposium on Security and Privacy*, May 2017.
https://www.ieee-security.org/TC/SP2017/poster-abstracts/IEEE-SP17_Posters_paper_26.pdf.

[71] Q. Zhang y S. Song: *A Systematic Procedure for Accurately Approximating Lognormal-Sum Distributions.*
IEEE Transactions on Vehicular Technology, 57(1):663–666, jan 2008.
https://doi.org/10.1109/tvt.2007.905611.