
Detección de Amenazas de Ransomware en Redes Empresariales EDR

Ransomware Threat Detection on EDR Enterprise Networks



TRABAJO FIN DE GRADO GRADO EN INGENIERÍA INFORMÁTICA, INGENIERÍA DEL SOFTWARE E INGENIERÍA DE COMPUTADORES CURSO 2020–2021

Gonzalo Fernández Megía

Gonzalo Figueroa del Val

Marina López Osorio

Directores

Luis Javier García Villalba

Esteban Alejandro Armas Vega

Departamento de Ingeniería del Software e Inteligencia Artificial

Facultad de Informática

Universidad Complutense de Madrid

Madrid, Junio de 2021

Agradecimientos

Queremos agradecer a Luis Javier García Villalba y a Esteban Alejandro Armas Vega, los Directores del Trabajo de Fin de Grado por el apoyo y disponibilidad ofrecidas durante toda la realización del proyecto. También agradecer a Ana Lucila Sandoval Orozco por su ayuda durante la realización de la memoria.

Asimismo, agradecemos al Grupo GASS (Grupo de Análisis, Seguridad y Sistemas, <http://gass.ucm.es>), en especial a Javier José Pecete García por las facilidades ofrecidas en la recogida masiva de muestras de malware.

Por último, nuestro más sincero agradecimiento a nuestro círculo cercano de familiares y amigos por la confianza brindada y el apoyo a lo largo de todos los años de carrera.

Índice General

Índice de Figuras	IX
Índice de Tablas	XIII
Lista de Acrónimos	XV
Resumen	XXI
Abstract	XXIII
1. Introducción	1
1.1. Introducción y Contexto Actual	1
1.2. Motivación	3
1.3. Objetivos	4
1.4. Plan de Trabajo	4
1.5. Estructura del Trabajo	5
2. Ransomware	7
2.1. Malware	7
2.1.1. Tipos de Malware	7
2.1.2. Ingeniería Social Aplicada al Malware	9
2.2. Definición de Ransomware	10
2.3. Historia del Ransomware	11
2.4. Tipos de Ransomware	16
2.5. Familias de Ransomware	17
2.6. Métodos de Propagación	20
2.7. Modo de Operación	22
2.7.1. Métodos de Cifrado	23
2.8. Detección	25
2.9. Prevención	26
2.9.1. Defensa Individualizada	26
2.9.2. Estrategias Empresariales	27
2.10. Período Post-ataque	28

3. Análisis e Identificación de Malware	31
3.1. Introducción	31
3.2. Análisis Estático	32
3.2.1. Técnicas	32
3.2.2. Herramientas	35
3.2.3. Limitaciones	35
3.3. Análisis Dinámico	35
3.3.1. Técnicas de Análisis	36
3.3.2. Herramientas	38
3.3.3. Limitaciones	38
3.4. Análisis Híbrido	39
3.5. Aprendizaje Automático Aplicado al Análisis de Malware	40
3.5.1. Aprendizaje Supervisado	42
3.5.2. Aplicación del Aprendizaje Automático al Análisis de Malware	48
4. Estado del Arte	49
4.1. Trabajos sobre Análisis de Malware	49
4.2. Trabajos sobre Análisis de Ransomware	52
4.2.1. Trabajos que Utilizan Características Heterogéneas	52
4.2.2. Trabajos que Utilizan Llamadas a la API de Windows	56
5. Técnica de Detección de Ransomware Propuesta	69
5.1. Descripción del Sistema Propuesto	69
5.2. Entorno de Obtención de Datos	71
5.2.1. Laboratorio de análisis	73
5.3. Conformación del <i>Dataset</i>	77
5.3.1. Estructura de reportes	78
5.3.2. Limpieza y Extracción de Características	78
5.4. Modelo y Algoritmos	80
6. Experimentos y Resultados	83
6.1. Métricas Utilizadas	83
6.2. Experimentos	85
6.2.1. Ajustar Parámetros	85
6.2.2. Experimentos con el <i>Dataset binario</i>	86
6.2.3. Experimentos con el <i>Dataset suma</i>	88
6.3. Resultados Finales	90
7. Conclusiones y Trabajo Futuro	93
7.1. Conclusiones	93
7.2. Trabajo Futuro	94

8. Contribuciones Individuales	95
8.1. Gonzalo Fernández Megía	95
8.2. Gonzalo Figueroa del Val	97
8.3. Marina López Osorio	98
9. Introduction	101
9.1. Motivation	103
9.2. Objectives	104
9.3. Work Plan	104
9.4. Work Structure	105
10. Conclusions and Future Work	107
10.1. Conclusions	107
10.2. Future Work	108
A. Análisis de una Muestra con Cuckoo Sandbox	109
B. Lista Extendida de las Familias de Ransomware	113
Bibliografía	119

Índice de Figuras

1.1.	Predicción del coste del ransomware en billones de dólares	1
1.2.	Sectores más atacados por ransomware en el tercer trimestre de 2020	2
1.3.	Ataques ransomware a dispositivos móviles	3
1.4.	Predicción de la cantidad de dispositivos IoT móviles en miles de millones	3
1.5.	Diagrama de Gantt de las tareas desempeñadas.	5
2.1.	Flujo de acción de un troyano	8
2.2.	Infección de equipos por gusano	9
2.3.	Familias de ransomware más observadas entre 2014 y 2015.	13
2.4.	Familias de ransomware más observadas desde enero hasta septiembre de 2016.	13
2.5.	Media de la cantidad de dinero que los <i>hackers</i> pedían por los rescates en los Estados Unidos (2018-2019).	14
2.6.	Volumen de menciones del COVID-19 en correos no deseados de todo el mundo en el mes de Marzo.	15
2.7.	Cronología de la evolución del ransomware.	16
2.8.	Aviso de extorsión del ransomware Petya	18
2.9.	Esquema de red de un ataque con Cryptowall	19
2.10.	Diagrama de acción de un botnet	21
2.11.	Patrón de ataque de un ransomware.	23
2.12.	El algoritmo 3DES aplica 3 veces el algoritmo DES con claves distintas	24
2.13.	Ventajas de la segmentación de la red	27
2.14.	Funcionamiento de honeypot	28
3.1.	Diagrama de flujo de un análisis estático.	34
3.2.	Diagrama de flujo de un análisis dinámico.	38
3.3.	Modelo de aprendizaje supervisado	41
3.4.	Modelo de aprendizaje no supervisado	41
3.5.	Modelo de aprendizaje por refuerzo	41
3.6.	Gráfico de la regresión logística.	43
3.7.	Representación gráfica del algoritmo KNN.	44
3.8.	Representación gráfica de un árbol de decisión.	45
3.9.	Representación gráfica del algoritmo Bosque Aleatorio.	45
3.10.	Representación gráfica de SVM lineal de dos dimensiones.	47

3.11. Taxonomía de las técnicas de ML en el análisis de malware [UAB19a]	48
4.1. Descripción esquemática del marco de análisis de Konrad Rieck <i>et al.</i>	50
4.2. Metodología general del sistema usado en el trabajo de Mamoun Alazab <i>et al.</i>	50
4.3. Arquitectura del sistema de detección de malware propuesto por Chandrasekar Ravi <i>et al.</i>	51
4.4. Patrones del ransomware para cifrar archivos: 1- El atacante sobrescribe el fichero del usuario con la versión cifrada. 2- El atacante lee un fichero, lo cifra en otro distinto y elimina el original. 3- El atacante lee un fichero, crea una nueva versión cifrada, sobrescribiendo el fichero original	53
4.5. Diagrama de secuencia del proyecto	55
4.6. Representación del sistema EldeRan [SMGML16]	57
4.7. Representación del sistema dinámico de detección de ransomware [CKYK17]	57
4.8. Diagrama de flujo del sistema propuesto [BLI19]	59
4.9. Modelo del sistema propuesto por [KAJS19]	60
4.10. Representación del flujo de datos del sistema PEDA propuesto por S.H. Kok <i>et al.</i> [KAJ20]	62
4.11. Flujo del proceso de entrenamiento y prueba.	63
5.1. Extracción de características y creación del modelo predictivo.	70
5.2. Clasificación de un archivo con un modelo predictivo.	70
5.3. Diagrama de flujo del sistema propuesto.	71
5.4. Arquitectura de Cuckoo Sandbox	72
5.5. Funcionamiento de la validación cruzada K-Fold	81
5.6. Codificación one hot	82
6.1. Matriz de confusión.	84
6.2. Comparación de la precisión con distintos métodos de escalado usando el <i>dataset suma</i>	85
6.3. Comparación de la precisión con distintos valores de K en validación cruzada K-fold usando en los dos <i>datasets</i>	86
6.4. Comparación de la precisión con distintas distribuciones de los dos <i>datasets</i>	86
6.5. Comparación de la precisión con y sin validación k-fold de todos los algoritmos en <i>dataset binario</i>	87
6.6. Comparación de la exactitud con y sin validación k-fold de todos los algoritmos en <i>dataset binario</i>	87
6.7. Comparación de la TPR con y sin validación k-fold de todos los algoritmos en <i>dataset binario</i>	88
6.8. Comparación de la precisión con y sin validación k-fold de todos los algoritmos en <i>dataset suma</i>	89
6.9. Comparación de la exactitud con y sin validación k-fold de todos los algoritmos en <i>dataset suma</i>	89

6.10. Comparación de la TPR con y sin validación k-fold de todos los algoritmos en <i>dataset suma</i>	90
9.1. Ransomware cost prediction in billions of dollars	101
9.2. Most attacked sector by ransomware in the third quarter of 2020	102
9.3. Ransomware attacks on mobile devices	103
9.4. Prediction of the number of mobile IoT devices in billions	103
9.5. Gantt chart of the performed tasks.	105
A.1. Pantalla principal Cuckoo Sandbox	110
A.2. Configuración del análisis	110
A.3. Aviso de malware	111
A.4. Pantalla de pago del rescate	111
A.5. Resultados de análisis 1	112
A.6. Resultados de análisis 2	112

Índice de Tablas

2.1. Herramientas para la recuperación de datos	29
3.1. Herramientas de análisis estático	35
3.2. Herramientas de análisis dinámico.	39
3.3. Diferencias entre análisis dinámico y estático.	39
4.1. Mapeo de las bases nitrogenadas en código binario	56
4.2. Tabla comparativa de los trabajos que usan aprendizaje automático para detectar malware mediante la extracción de APIs.	64
5.1. <i>Datasets</i> finales.	80
6.1. Resultados con el <i>dataset binario</i>	87
6.2. Resultados con el <i>dataset suma</i> escalando los datos con <i>StandardScaler</i> . . .	88
6.3. Comparación de los resultados obtenidos con otros trabajos usando el algoritmo RF	90
6.4. Comparación de los <i>datasets</i> construidos con los del estado del arte	91
B.1. Familias de ransomware.	113

Lista de Acrónimos

3DES *Triple DES*

AES *Advanced Encryption Standard*

API *Application Programming Interface*

ASCII *American Standard Code for Information Interchange*

AUC *Area under the curve*

C&C *Command and Control*

CD-ROM *Compact Disc Read-Only Memory*

CF-NCF *Class Frequency - Non-Class Frequency*

CFG *Calls Flow Graphs*

COVID-19 *Corona Virus Disease*

CPU *Central Processing Unit*

CSV *Comma-separated values*

DDoS *Distributed Denial of Service*

DES *Data Encryption Standard*

DGA *Domain Generation Algorithm*

DH *Diffie-Hellman*

DLL *Dynamic Link Library*

DNA *Deoxyribonucleic Acid*

DNS *Domain Name Server*

DSA	<i>Digital Signature Algorithm</i>
DSM	<i>DiskStation Manager</i>
DT	<i>Decision Tree</i>
DVD	<i>Digital Versatile Disc</i>
ELF	<i>Executable and Linkable Format</i>
FDE	<i>Full Disk Encryption</i>
FLOSS	<i>FireEye Labs Obfuscated String Solver</i>
FN	<i>False Negatives</i>
FNR	<i>False Negative Rate</i>
FP	<i>False Positives</i>
FPR	<i>False Positive Rate</i>
FTP	<i>File Transfer Protocol</i>
GP	<i>Gauss Process</i>
HPC	<i>Hardware Performance Counter</i>
HTML	<i>Hypertext Markup Language</i>
HTTP	<i>Hypertext Transfer Protocol</i>
HTTPS	<i>Hypertext Transfer Protocol Secure</i>
ID3	<i>Iterative Dichotomiser 3</i>
IDA	<i>Interactive Disassembler Professional</i>
IoC	<i>Indicators of compromise</i>
IoT	<i>Internet of things</i>
IP	<i>Internet Protocol</i>
IRP	<i>I/O request packets</i>

KNN *K-Nearest Neighbor*

LAN *Local Area Network*

LR *Logistic Regression*

LTS *Long Term Support*

MBR *Master Boot Record*

MD5 *Message-Digest Algorithm*

MFT *Master File Table*

ML *Machine Learning*

MLP *Multilayer perceptron*

NAS *Network Attached Storage*

NB *Näive Bayes*

NLA *Network Level Authentication*

NN *Neural Networks*

NSA *National Security Agency*

OMS La Organización Mundial de la Salud

PE *Portable Executable*

PEDA *Pre-Encryption Detection Algorithm*

PID *Process ID*

QDA *Qualitative Data Analysis*

RaaS *Ransomware as a Service*

RAT	<i>Remote Access Trojan</i>
RATS	<i>Remote Access Terminal Server</i>
RBF	<i>Radial Basis Function</i>
RC4	<i>Rivest Cipher 4</i>
RDP	<i>Remote Desktop Protocol</i>
RF	<i>Random Forest</i>
RISS	<i>Resilient Information Systems Security</i>
ROC	<i>Receiver Operating Characteristic</i>
RSA	<i>Rivest–Shamir–Adleman</i>
SEPE	Servicio Público de Empleo Estatal
SFX	<i>Self-extracting Archive</i>
SGD	<i>Stochastic Gradient Descent</i>
SHA	<i>Secure Hashing Algorithm</i>
SME	<i>Small and Medium Enterprises</i>
SMOTE	<i>Synthetic Minority Oversampling Technique</i>
SMS	<i>Short Message Service</i>
SQL	<i>Structured Query Language</i>
SVM	<i>Support Vector Machine</i>
TF-IDF	<i>Term Frecuency - Inverse Term Frecuency</i>
TLS	<i>Transport Layer Security</i>
TN	<i>True Negatives</i>
TNR	<i>True Negative Rate</i>
TOR	<i>The Onion Router</i>
TP	<i>True Positives</i>
TPR	<i>True Positive Rate</i>

UDP	<i>User Datagram Protocol</i>
URL	<i>Uniform Resource Locator</i>
USB	<i>Universal Serial Bus</i>
VMI	<i>Virtual Machine Introspection</i>
VPN	<i>Virtual Private Network</i>
XSS	<i>Cross-site scripting</i>

Resumen

Con el aumento del uso de las tecnologías en todos los aspectos del día a día, sucede a la vez un aumento de las amenazas hacia estas y, entre ellas, una de las principales amenazas es el ransomware, un tipo de malware que bloquea equipos o cifra información impidiendo el acceso a ella y pide un rescate económico. El daño que provocan estos ataques es cada vez mayor, con objetivos como las *Small and Medium Enterprises (SME)*, que tienen medios más limitados para protegerse o luchar contra estas amenazas. Es por ello que este estudio arranca con la idea de desarrollar una herramienta de fácil acceso e implantación que sirva de solución contra ataques ransomware. En este trabajo se presentan tres partes. La primera, la construcción de un laboratorio de análisis de malware para la ejecución y análisis de muestras de ransomware en un entorno seguro y de fácil despliegue. La segunda, a partir de información de 15352 muestras de ransomware y de programas benignos, el desarrollo de dos *dataset* de una extensión considerablemente mayor a los encontrados en otros trabajos relacionados con la detección de malware. Para construir estos *datasets* se ha hecho uso de las llamadas a la *Application Programming Interface (API)* de Windows de las muestras analizadas, que reflejan el comportamiento del software y su interacción con el sistema. Después de la limpieza de los datos y la selección de características, el primer *dataset* consta de 714 muestras y el segundo de 6630. La tercera y última parte, el desarrollo de un modelo de inteligencia artificial utilizando diferentes algoritmos de aprendizaje automático capaz de identificar muestras de ransomware, alimentado por los *datasets* mencionados previamente. Los resultados varían dependiendo del algoritmo utilizado y el *dataset* en el que se aplica, llegando a conseguir una precisión del 98% y una exactitud del 97% haciendo uso del algoritmo *Random Forest (RF)* sobre el *dataset* de mayor extensión.

Palabras clave: Análisis de Malware, Aprendizaje Automático, Ciberseguridad, Robo de Datos, Sandboxing, Selección de Características, Ransomware.

Abstract

Due to the increasing use of technologies in everyday life, there is an increase of threats against them and one of the main threats is ransomware, a malware that blocks the device or encrypts its information, preventing the accesss to it and then asks for an economic ransom. The damage caused by these attacks is increasingly growing, with targets such as [SME](#), which have limited resources to deal with these threats. Consequently, this study starts with the idea of developing a simple tool with easy access to detect ransomware attacks. This work is threefold. The first part involves the construction of a malware analysis laboratory to execute and analyse ransomware samples in a safe and simple to deploy environment. The second part, based on information of 15352 ransomware and benign software samples, deals with the development of two datasets of a considerably larger extension than those found in other studies related to malware detection. To build both datasets we use Windows [API](#) calls made by the analysed samples, which reflect software behaviour and system interaction. After the cleanup of data and feature selection, the first dataset is made up of 714 samples and the second one 6630. The third and last part involves the development of an artificial intelligence model, feeding the datasets to it and using various machine learning algorithms that are able to identify ransomware among the samples. The results are different depending on the machine learning algorithm and the datasets used. The best results obtained are 98 % accuracy and 97 % precision using the [RF](#) algorithm on the largest dataset.

Keywords: Cybersecurity, Data Theft, Feature Selection, Machine Learning, Malware Analysis, Sandboxing, Windows [API](#), Ransomware.

Capítulo 1

Introducción

1.1. Introducción y Contexto Actual

Hoy en día, prácticamente toda la información se almacena digitalmente y puede ser accedida en cualquier momento de manera sencilla. La mayoría de las personas están interconectados en todo momento con múltiples dispositivos mediante Internet, una plataforma cada vez más segura debido al trabajo de los especialistas, pero en la cual los delincuentes cibernéticos siguen encontrando vulnerabilidades y atacando todo tipo de entidades para conseguir beneficios [SM17]. Hay muchos programas maliciosos como los virus, los gusanos informáticos y los troyanos que pueden dañar los sistemas digitales. Una de las principales amenazas cibernéticas en la actualidad, según un reporte de la agencia Europol (*European Union Agency for Law Enforcement Cooperation*) [Lab19], es el ransomware (rescate, en inglés “ransom”, y “ware”, acortamiento de software). Se trata de un tipo de malware que bloquea equipos o impide el acceso a los datos usando el cifrado de clave privada hasta que el usuario de dicho equipo paga un rescate, normalmente en *bitcoin*. El robo de datos como extorsión lleva siendo utilizado desde alrededor de 2005, pero debido al auge del ransomware y el *bitcoin*, se han incrementado enormemente el número de casos [Zet15], con casi el 40 % de empresas habiendo sufrido al menos uno según un estudio realizado por Malwarebytes [Gua16], lo que evidencia que estos programas maliciosos son una amenaza importante que hay que detectar y frenar cuanto antes.

Los delincuentes cibernéticos o ciberdelincuentes son cada vez más creativos e innovadores y, así mismo, el daño que causan va en aumento. En la Figura 1.1 se muestra la predicción del coste de mitigación de daños frente a ataques ransomware para el año 2021, pudiendo apreciarse un gigantesco aumento respecto a la década pasada. Estos costes se deben tanto a la inversión para defender los sistemas, como a las pérdidas que se producen al no poder utilizar los equipos y al pago de los rescates.

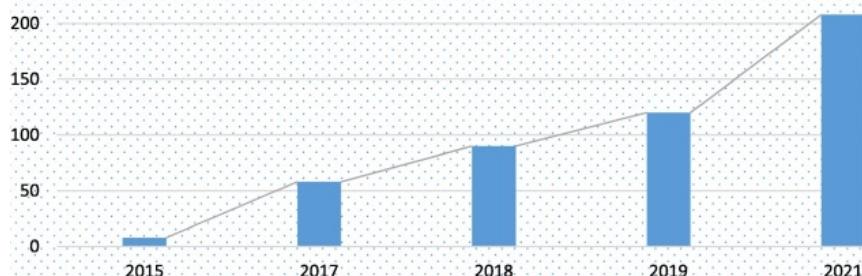


Figura 1.1: Predicción del coste del ransomware en billones de dólares

En la Figura 1.2 se observa los resultados obtenidos de un estudio de Coveware [Cov20] sobre las industrias más afectadas por los ataques ransomware en el último trimestre del año 2020. Se puede apreciar que los servicios profesionales, el sector público y sanitario son los más afectados, ya que son los más rentables para los delincuentes. Esto se debe a su necesidad de estar en constante funcionamiento y por su posesión de datos delicados, por lo que suelen pagar los rescates para que se les devuelva sus datos y el acceso a sus sistemas lo más rápido posible.

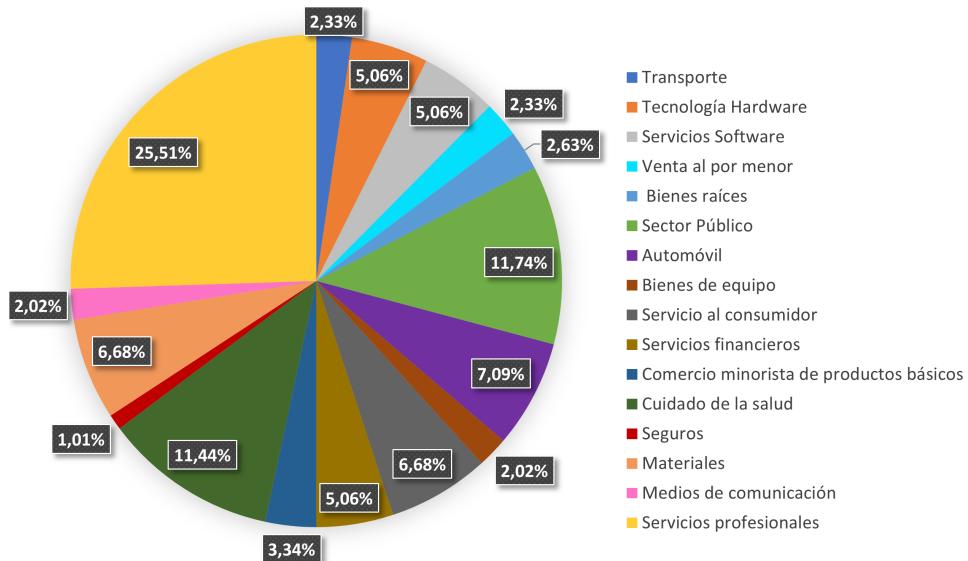


Figura 1.2: Sectores más atacados por ransomware en el tercer trimestre de 2020

El riesgo de un ciberataque es el mismo para una gran empresa que para una SME. Estas compañías también utilizan, producen y almacenan grandes cantidades de datos. La diferencia es que las empresas más pequeñas han de lidiar con estos problemas con recursos mucho más limitados. Por lo tanto, deben hacer un mayor esfuerzo económico para trazar estrategias de seguridad ante estos ataques informáticos [Kur15].

Al escenario planteado se le añade también la situación sanitaria actual, una la pandemia ocasionada por la enfermedad *Corona Virus Disease (COVID-19)*. Mientras esta se propagaba por el mundo, crecía una amenaza tecnológica secundaria basada en ciberataques indiscriminados y cibercampañas centradas en autoridades públicas y organizaciones [LSN+21], así como en los individuos basándose en su desinformación y temor, como la falsa distribución de libros con información sobre el COVID-19 [TIT20], la oferta fraudulenta de medicamentos vía correo electrónico [fFN20], etc. Otro factor que ha favorecido el aumento de ciberataques, y en concreto, de ataques ransomware, es el establecimiento del teletrabajo. En España, estos tipos de ataques aumentaron un 160% en el segundo semestre del año 2020, encabezando el *ranking* europeo, debido a que las empresas que centraron sus esfuerzos en establecer entornos de trabajo a distancia, no aplicaron la seguridad necesaria para ello [pE20]. En España, las únicas empresas afectadas no han sido solo las privadas, ya que, por ejemplo, instituciones públicas recibieron ataques ransomware. El *Servicio Público de Empleo Estatal (SEPE)* tuvo que paralizar la totalidad de sus servicios durante 5 días, añadiendo otras 20 jornadas sin permitir el teletrabajo por un ataque en marzo de 2021 por el ransomware Ryuk, provocando una grave situación, pues la actividad del SEPE se había disparado por el impacto de la pandemia en el desempleo [JP21].

El desarrollo de la tecnología proporciona beneficios y facilidades en todos los ámbitos y tareas, hasta en los más cotidianos. Ejemplo de ello es el Internet de las Cosas, o *Internet of things (IoT)*, que se define como la integración de lugares y objetos de la vida real con Internet, mediante la interconexión de dispositivos, sensores, actuadores, software, etc. mediante una red, que almacena e intercambia información [ADC18]. Sin embargo, la existencia de esta tecnología ha permitido que la cantidad de ataques ransomware crezcan con el tiempo, especialmente tras el incremento del uso de dispositivos IoT inteligentes, particularmente los móviles. En la Figura 1.3 se puede observar la cantidad de ataques ransomware a dispositivos móviles por trimestre en 2018. La Figura 1.4 muestra una predicción del incremento del uso de este tipo de dispositivos debido al auge del IoT. Con estos datos se puede predecir que la cantidad de ataques continuará en alza y se demuestra que existe una necesidad tanto de proteger a los individuos y las organizaciones de esta clase de ataques como de ofrecer la información existente a los usuarios y trabajadores para actuar de una forma responsable y segura para prevenirlas [HJAP21].

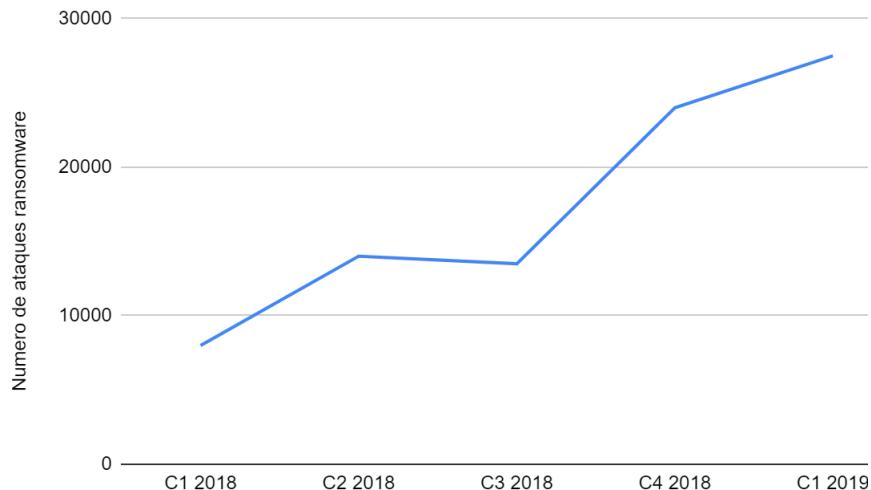


Figura 1.3: Ataques ransomware a dispositivos móviles

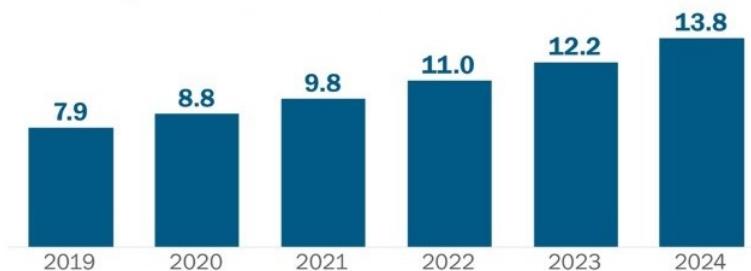


Figura 1.4: Predicción de la cantidad de dispositivos IoT móviles en miles de millones

1.2. Motivación

Como se ha visto en la Sección 1.1, es evidente el impacto de los ataques ransomware en las empresas en los últimos años, provocando grandes inversiones económicas para mejorar la seguridad. Las SME tienen mayor dificultad para afrontar dichos gastos y por

lo tanto están más expuestas, debido a que no tienen sistemas de prevención, especialistas enfocados en la ciberseguridad e incluso información relativa a este sector.

En la Unión Europea, las [SME](#) constituyen un pilar fundamental de la economía, representando en septiembre de 2020 un sorprendente 99,8% de todas las empresas empleadoras, el 65% del empleo del sector privado y el 54% del sector privado de producción bruta. Éstas han sido además muy vulnerables a la pandemia ocasionada por el [COVID-19](#), provocando grandes pérdidas e incluso gran cantidad de cierres [[GKÖPS20](#)]. Debido a todos estos factores, la motivación principal de este estudio es el desarrollo de un modelo de detección de ransomware que cumpla con los siguientes requisitos: Precisión, fiabilidad, escalabilidad, facilidad de uso, rapidez y software libre.

De esta manera se puede llevar a cabo una estrategia de prevención contra ataques ransomware y reducir el impacto de los daños ocasionados por ellos a través de una herramienta accesible a las [SME](#), reduciendo de manera drástica los costes de inversión.

1.3. Objetivos

Al inicio del trabajo, se establecen una serie de objetivos para asegurar su calidad y fijar unas pautas para el correcto desarrollo:

- Utilizar las competencias adquiridas durante el grado.
- Desarrollar un modelo de [Machine Learning \(ML\)](#) capaz de identificar muestras de ransomware con gran precisión.
- Construir un extenso *dataset* para garantizar la fiabilidad en los resultados obtenidos.
- Desarrollar y desplegar un laboratorio para el análisis de muestras de malware.
- Demostrar la efectividad de las llamadas a las [API](#) de Windows de un ejecutable como mecanismo de detección de ransomware.

1.4. Plan de Trabajo

El desarrollo de este trabajo se ha realizado en las siguientes fases:

1. **Investigación:** Esta fase se llevó a cabo durante los primeros cuatro meses, donde se adquirió la información necesaria para el desarrollo del trabajo. Al comienzo de la fase, se realizaron varias reuniones donde se explicaron los objetivos del proyecto y los conocimientos necesarios, siendo estos los conceptos fundamentales del aprendizaje automático. Una herramienta primordial para la investigación fue *Google Scholar*, ya que permite buscar artículos científicos y citarlos con facilidad. Se buscaron artículos y publicaciones sobre detección tanto de ransomware como de malware, además de libros sobre análisis malware para adquirir conocimientos más profundos del tema y entender el contexto del trabajo.
2. **Desarrollo:** Una vez obtenidos los conocimientos necesarios, se comenzó el desarrollo del modelo. No se abandonó por completo la investigación, pero pasó a un segundo plano. Durante esta fase se obtuvieron los datos necesarios y se construyeron los dos *datasets* para entrenar y evaluar el modelo propuesto. Posteriormente se pasó a la familiarización con el lenguaje de programación Python y con la librería *scikit-learn*, usada para la codificación del modelo y la implementación de los algoritmos de [ML](#).

3. **Experimentación y resultados:** En esta fase experimentó con el modelo desarrollado y los datos extraídos. Al obtener los resultados era necesario evaluarlos y demostrar la validez del modelo, por lo que se usó la validación cruzada K-Fold. La experimentación fue vital para la fase de desarrollo, ya que se tuvo que modificar el modelo para mejorar los resultados, ajustando los parámetros, obteniendo más datos...
4. **Documentación:** La fase de documentación se llevó a cabo de manera paralela al resto de fases. Empezó en los últimos meses de la fase de investigación con los conocimientos asentados, y continuó durante el desarrollo y la experimentación. Todos los miembros del equipo participaron en la escritura de la memoria, trabajando en conjunto en todas las secciones. Cualquier actualización debía de ser aceptada y revisada por los otros miembros del equipo.

El modelo desarrollado en este trabajo está subido en el repositorio de Gitlab [TFG Ransomware 20-21](#).

La Figura 9.5 es un diagrama de Gantt donde se pueden apreciar las actividades desempeñadas por el equipo a lo largo de los meses y su duración.

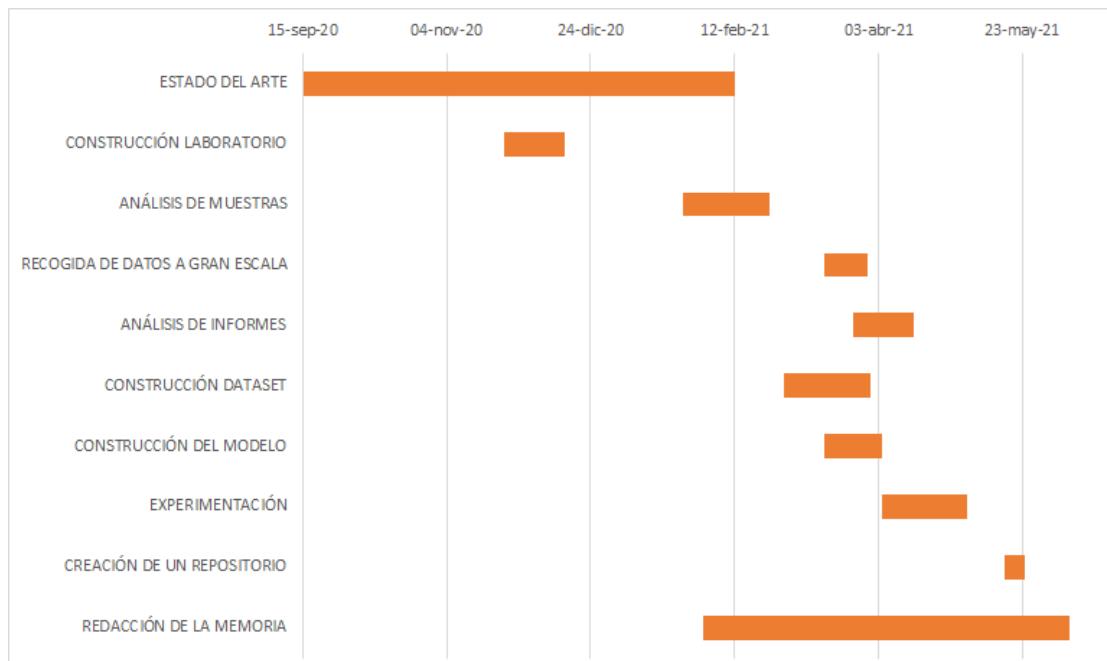


Figura 1.5: Diagrama de Gantt de las tareas desempeñadas.

1.5. Estructura del Trabajo

El resto del trabajo está organizado en 10 capítulos, siguiendo esta estructura:

El Capítulo 2 introduce el ransomware, explicando los conceptos fundamentales sobre el malware y los diferentes tipos, el modelo de negocio que siguen los cibercriminales para sacar beneficio de los ataques y la evolución histórica del ransomware, detallando cada año los diferentes ataques que han habido en el mundo. También se exponen los diferentes tipos de ransomware, las familias, los métodos de propagación más comunes y

posteriormente el modo de operación para infectar el sistema de la víctima. El capítulo termina explicando algunas técnicas usadas por programas antivirus para reconocer la presencia del ransomware y unas estrategias y acciones de prevención que toda empresa debería seguir para proteger sus sistemas de estos ataques, y en caso de sufrirlos, qué pasos deberían seguir para recuperar sus datos.

El Capítulo 3 aborda los tipos de análisis de malware que existen, las herramientas que se usan, las limitaciones de cada uno y las diferentes técnicas de identificación. Posteriormente se habla del aprendizaje automático aplicado al análisis malware, exponiendo cómo se construye un modelo de [ML](#) usando Python como lenguaje de programación y los diferentes algoritmos que existen, indagando en los denominados algoritmos de aprendizaje supervisado, ya que son los que más se utilizan en análisis y detección de malware y ransomware.

El Capítulo 4 se muestran trabajos relacionados con el tema de análisis y detección de malware y ransomware usando modelos de [ML](#) y algoritmos de aprendizaje automático. Se distinguen entre trabajos que solo analizan malware de los que analizan ransomware, y dentro de los trabajos de ransomware se diferencian los que usan las llamadas a la [API](#) de Windows para construir sus modelos de los que no.

El Capítulo 5 expone las tecnologías usadas para desarrollar el modelo de [ML](#), el entorno de trabajo y la obtención de los datos. Se muestra el diagrama de flujo del sistema propuesto y se habla sobre el desarrollo del laboratorio de análisis para el estudio de las muestras de ransomware y goodware, de las cuales se extraen las llamadas a la [API](#) de Windows para la construcción de los *datasets*. Posteriormente se hace una limpieza de los datos en los *datasets* para descartar aquellos que no son relevantes para el estudio y se explica cómo se creará el modelo.

El Capítulo 6 aborda los experimentos realizados y las métricas utilizadas para demostrar la efectividad del modelo para detectar ransomware. Después de mostrar los experimentos sobre los dos *datasets* y los parámetros utilizados para su desarrollo, se obtienen los resultados finales y se validan con la validación cruzada K-fold. Posteriormente se elige el algoritmo que mejores resultados ha obtenido y se comparan con los resultados de los trabajos relacionados expuestos en el Capítulo 4.

El Capítulo 7 muestra las conclusiones del trabajo tras el análisis de los resultados y expone las líneas de investigación futuras.

El Capítulo 8 relata las aportaciones individuales de cada miembro del equipo al desarrollo del trabajo.

Los Capítulos 9 y 10 son las traducciones al inglés de los Capítulos 1 y 7.

Capítulo 2

Ransomware

En este capítulo se realiza un estudio detallado sobre ransomware. Los conceptos fundamentales sobre el malware, sus tipos y su modelo de negocio se presentan en la Sección 2.1. Los tipos de malware con mayor impacto en la actualidad. En la Sección 2.2 se define el ransomware y se enumeran las principales diferencias con respecto otros tipos de malware. Su evolución histórica, los tipos y las familias más importantes se detallan en la Secciones 2.3 a 2.5. En la Sección 2.6 se exponen los diferentes vectores de ataque y los métodos de operación analizan en la Sección 2.7. Las técnicas de detección y prevención de los ataques de ransomware así como las recomendaciones para el post-ataque se explican en las Secciones 2.8 a 2.10.

2.1. Malware

Malware es una abreviación de software malicioso. Se trata de programas o fragmentos de código que ganan acceso o provocan daño en un ordenador, sin necesidad de que su propietario lo sepa. El malware utiliza canales de comunicación populares como el correo electrónico o descargas desde páginas web para expandirse. Su finalidad es explotar vulnerabilidades del sistema. Este software ajeno se instala en el equipo y realiza tareas no deseadas, como la aparición de continua y molesta publicidad en la pantalla, pero incluso puede ser utilizado para obtener beneficios económicos a terceros, mediante el robo de información o infección de redes [PAT17]. El malware se clasifica dependiendo de su modo de operación, que pueden ser los siguientes [Ayc06]:

1. **Auto-replicación:** Intenta propagarse mediante las copias o instancias de él mismo. De modo análogo, otros tipos de malware se copian de manera pasiva, por error de algún usuario, por ejemplo.
2. **Expansión de la infección:** Describe la capacidad de crecimiento en el número de instancias creadas. Aquellos que se reproducen mediante auto-replicación crecen de manera mucho más rápida.
3. **Parasitario:** Requiere otro código ejecutable para existir.

2.1.1. Tipos de Malware

Los tipos de malware más importantes son:

- **Virus:** Un virus es un tipo de malware parasitario y con crecimiento exponencial que, cuando se ejecuta, intenta replicarse en otro código ejecutable. Cuando lo consigue, se

dice que dicho código ha sido infectado. El código infectado, cuando se ejecuta, puede a su vez infectar nuevo código. Este modo de auto-replicarse en código existente en un computador es la característica definitoria de un virus. Tradicionalmente, los virus se pueden propagar en un solo ordenador, o migrar de unos a otros, transportados a través de un humano, vía *Universal Serial Bus (USB)*, *Compact Disc Read-Only Memory (CD-ROM)* o *Digital Versatile Disc (DVD)*, pero nunca a través de ninguna red [Ayc06].

- **Spyware:** Una clase de código malicioso que se instala de manera encubierta en la máquina de la víctima, y una vez activo, silenciosamente monitoriza el comportamiento de los usuarios, graba sus hábitos al navegar en la red y sustrae sus archivos sensibles y contraseñas. También permite realizar capturas de pantalla, obtener información de la cámara y el micrófono, evitar ser desinstalado e incluso instalar software [Wer13]. Normalmente, la información recogida es enviada al distribuidor de *spyware*, y posteriormente utilizada para realizar estudios de publicidad o marketing, e incluso vendida a terceros. Una de las técnicas más importantes es explotar las vulnerabilidades de los navegadores [EKK⁺07].
- **Adware:** Es considerada una forma menos dañina de *spyware*. Mientras que el *spyware* trabaja de manera encubierta, el *adware* es evidente. Algunas de sus formas de actuar pueden ser cambiar la página de inicio de un navegador, modificar el contenido de páginas web para insertar anuncios, realizar un seguimiento del comportamiento del usuario y transmitir dicha información, que puede ser manteniendo o no la privacidad del usuario, para obtener algún beneficio [Wer13].
- **Trojans:** Denominados así en referencia al caballo de madera utilizado por los griegos en el asedio a Troya, debido a que el código malicioso entra en el equipo camuflado, normalmente mediante algún archivo descargado al abrir un correo electrónico extraño. Este tipo de malware parasitario permite a su creador ejecutar comandos sin autorización en un equipo infectado. Permite a los *hackers* modificar ajustes de los archivos, robar información o contraseñas, realizar daños en el equipo, apagarlo o incluso deshabilitar antivirus u otros mecanismos de seguridad. Actúan según la Figura 2.1 [Erb05].

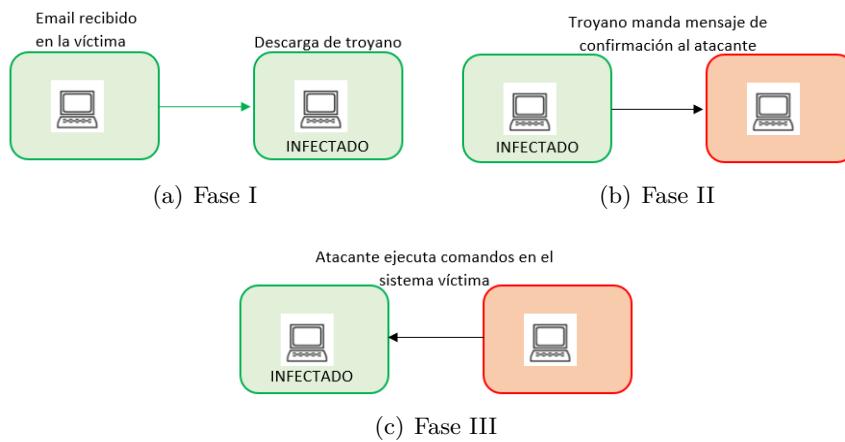


Figura 2.1: Flujo de acción de un troyano

- **Worms:** Es un código malicioso que se origina en un único ordenador y busca otros equipos conectados a un área local *Local Area Network (LAN)* o mediante una conexión a Internet. Cuando el gusano encuentra otro ordenador, se replica en dicho equipo y continúa buscando más ordenadores. Un gusano sigue intentando replicarse indefinidamente o hasta que expira un temporizador, tal como muestra la Figura 2.2 [Erb05].

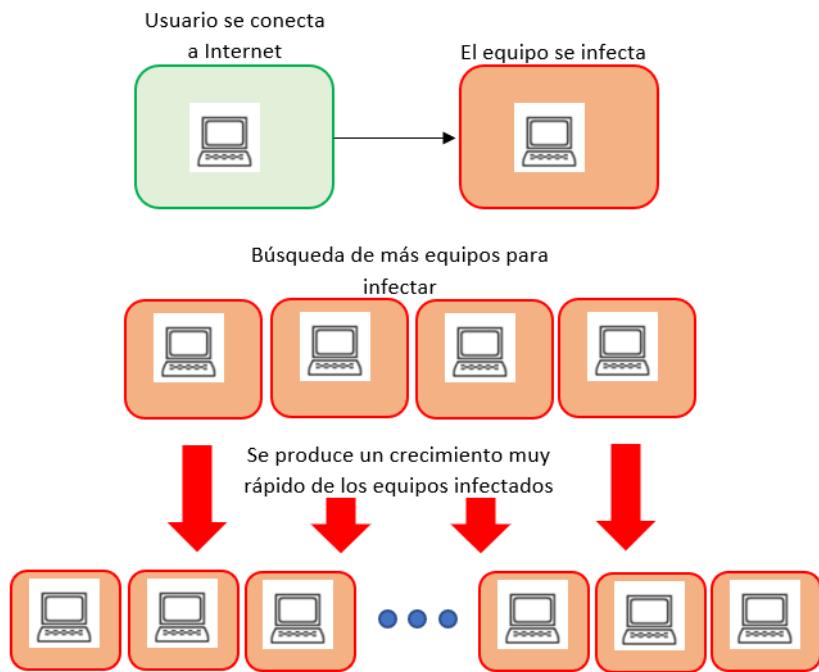


Figura 2.2: Infección de equipos por gusano

- **Logic Bomb.** Código malicioso que consiste en una “carga explosiva”, que puede ser cualquier acción maligna para el equipo víctima, y un disparador: una condición booleana que controla cuando se va a ejecutar la “carga explosiva” [Ayc06].

Algoritmo 2.1: Ejemplo de bomba lógica sencilla

```

1 // Codigo legitimo
2 if fecha is Viernes 13 then
3     apagar_ordenador()
4 // Codigo legitimo

```

2.1.2. Ingeniería Social Aplicada al Malware

La ingeniería social consiste en una serie de técnicas psicológicas y habilidades sociales que permiten persuadir a las personas para obtener de ellas información, ya sea personal o profesional, o para conseguir acceso a su sistema o red. Se basa en una metodología que sigue estas fases [GSBTLR⁺17]:

1. **Recogida de información:** En esta fase se obtiene información sobre la víctima para establecer posibles vectores de ataque.

2. **Obtención de confianza:** Las personas tienden a revelar información o secretos con aquellos en quien confían. Debido a esto, el atacante trata de obtener una relación de confianza con la víctima.
3. **Fase de aprovechamiento:** La relación de confianza establecida da lugar a la petición de información o la acción requerida.
4. **Fase final:** Los resultados obtenidos en la fase anterior son usados para realizar el ataque planeado previamente.

Los tipos fundamentales de ingeniería social utilizados para realizar ataques malware son los siguientes [GSBTLR⁺¹⁷]:

- **Ataques técnicos:** No existe interacción física con la víctima. Se usan herramientas como el correo electrónico o las descargas en la web. Su principal amenaza es que intentan parecer entidades reconocidas y fiables, utilizando logos comerciales, platillas de mensajes, etc.
- **Ataques de ego:** Existe un contacto directo entre el atacante y la víctima. Los criminales ofrecen ayuda demostrando su inteligencia, de forma que las víctimas caen en un tipo de manipulación transparente, y proporcionan la información con facilidad.
- **Ataques de simpatía:** Se crea una relación de confianza con la víctima, la cual siente empatía por el atacante. Se basa en el uso de técnicas conversacionales para generar una comunicación espontánea, de manera que la víctima se siente segura y expone sus vulnerabilidades.
- **Ataques de acoso:** Se basan en la intimidación, amenazas y coacción para obtener información.

2.2. Definición de Ransomware

Ransomware, o secuestro de datos, es un tipo de malware, con características típicas de malware, pero a su vez con sus propias peculiaridades. Se basa en la inyección de procesos en programas destino, que provocan la extracción de datos del usuario y permite la comunicación segura con el servidor de mando y control *Command and Control (C&C)*. Su principal objetivo es cifrar los archivos privados del sistema víctima o inutilizar el equipo y forzar al pago por su rescate. El código ransomware es relativamente fácil de encontrar por la web, escribir y modificar. Esta simplicidad del código combinada con su naturaleza lucrativa, ha resultado en la generación explosiva de gran cantidad de variantes [HKLK20a]. El ransomware se diferencia de otros tipos de malware por las siguientes características [Has19a]:

- La finalidad del ransomware es extorsionar dinero a las víctimas, sin dañar el sistema operativo o los datos de manera irreversible, mientras que otros tipos de malware únicamente buscan convertir en inoperante el equipo.
- No requiere privilegios de administrador para cifrar los archivos.
- La habilidad tanto de cifrar archivos como de modificar su nombre, hace inconsciente a la víctima de la cantidad de archivos cifrados.

- Utilizan diferentes métodos para evadir mecanismos de seguridad como *firewalls* o antivirus.
- Algunos tipos conectan el equipo víctima a *botnets* para otros ciberataques posteriores.
- Los ataques son independientes de la localización geográfica y del lenguaje del área.

2.3. Historia del Ransomware

El ransomware ha sido una amenaza importante para todo tipo de empresas desde mediados de la década del 2000, pero eso no significa que no haya habido ataques anteriores. Su historia se puede estructurar en dos fases según el tipo de acciones malignas que el malware realiza en el equipo [Col18]:

1. **Bloqueo de equipo:** En esta etapa se restringía el acceso al sistema operativo, obligando a la víctima a pagar un rescate para poder utilizar su equipo. Este tipo de ataques comenzó a acarrear gran riesgo por parte de los atacantes debido a que los expertos informáticos aprendieron a luchar contra ellos atacando directamente a los pagos electrónicos.
2. **Cifrado de información:** Con el nacimiento de las cripto-monedas y habiendo eliminado a los intermediarios en los pagos, como los bancos y al ser imposible su rastreo, surge este tipo de ataque, por el cual se pide un rescate a usuarios y empresas tras cifrar archivos con claves robustas, ofreciéndoles mayor cantidad de beneficios de una manera más segura para los atacantes.

El primer ransomware conocido fue el AIDS Trojan, también llamado PC Cyborg Trojan. Fue creado en 1989 por Joseph Popp, un biólogo estadounidense, y distribuido a través de disquetes en una conferencia de la OMS y enviados a través de los servicios postales. Estos disquetes venían en sobres con el remitente “PC Cybor Corporation” y decían tener información sobre el SIDA, por lo que fueron enviados a investigadores alrededor del mundo. Se distribuyeron alrededor de 20.000 disquetes en más de 90 países. Usaba criptografía simétrica simple para cifrar nombres de archivos del sistema y hacía que las impresoras conectadas imprimieran una nota de rescate pidiendo al usuario enviar \$189 dólares a un buzón en Panamá para que pudieran recibir el software que desbloquearía su ordenador. Este ransomware causó daños severos al campo científico, haciendo que compañías perdieran todo su trabajo de investigación sobre el tratamiento del SIDA [Sim15], y fue la base para la evolución del ransomware hacia los ataques más sofisticados que se llevan a cabo hoy en día.

El siguiente gran paso en la evolución del ransomware fue en 1996, cuando dos investigadores presentaron un artículo en la Conferencia de Seguridad y Privacidad de IEEE llamado *Cryptovirology: extortion-based security threats and countermeasures* [YY96]. Este documento expuso un programa que usaba cifrado de la clave pública para crear código malicioso que afectara al ordenador infectado y extorsionara a las víctimas para que pagaran una cuantía de dinero por el rescate de su sistema. El documento sugirió los términos *cryptoviral extortion* y *cryptovirology* para nombrar este tipo de ciberataque, que es lo que hoy en día se conoce como ransomware [Has19b].

Antes del 2005 el ransomware no era muy popular entre los delincuentes y no se produjeron ataques importantes. Sin embargo, esto cambió drásticamente en 2005, cuando los desarrolladores de malware comenzaron a utilizar el cifrado en su código

malicioso, creando ransomware de tipo Filecoder. Conocido como el primer ransomware moderno, Trojan Gpcoder o GP Code, fue el más notable ya que utilizaba cifrado *Rivest–Shamir–Adleman (RSA)* de 1024 bits, un cifrado fuerte en ese momento, lo que dificultaba la recuperación de los archivos afectados y se propagaba adjunto a correos electrónicos, simulando ser una aplicación de trabajo. Después de GPCoder, surgieron otras familias como Krotten y Cryzip [Kas16].

En 2007 apareció el primer ransomware de tipo Lockscreen llamado WinLock. Este ransomware se hacía con el control de la pantalla de la víctima y mostraba imágenes inapropiadas junto con un mensaje. Este mensaje informaba al usuario que tenía que pagar un rescate a través de *Short Message Service (SMS)* para desbloquear su sistema [Mal18].

En 2009 surgió el primer ataque de un ransomware de tipo Scareware llamado Vundo, y esto supuso el salto a la monetización intensiva de este tipo de crímenes, ayudados por la proliferación de plataformas anónimas de pago en línea. Los delincuentes comprendieron que se podía ganar dinero con el ransomware y este hecho fue el detonante del rápido crecimiento del número de amenazas cada vez más complejas. En 2011, estos tipos de ransomware estaban en auge, detectándose 60.000 nuevos ataques, y en 2012, esa cifra había aumentado hasta los 200.000. Se estimó que, a finales del 2012, el mercado negro de ransomware tenía un valor aproximado de \$5 millones. El ransomware más notable que utilizó tales tácticas para hacerse pasar por agencias de la ley fueron Reveton y Kovter. Reveton cobró los rescates utilizando *bitcoins* y tarjetas anónimas de pago como *MoneyPak* [Gro19].

CryptoLocker apareció en 2013 y usaba criptografía de clave pública y privada (*Advanced Encryption Standard (AES)* y *RSA* de 2048 bits) para cifrar y descifrar los archivos de las víctimas. Fue distribuido por un correo electrónico que simulaba proceder de UPS o FedEx y la versión original de CryptoLocker cifraba alrededor de 67 tipos diferentes de archivos y daba a las víctimas 3 días para pagar. El precio del rescate rondaba entorno a 2 *bitcoins* o 100 dólares. En diciembre de aquel año, se descubrió que 250.000 máquinas fueron infectadas y en torno a 42000 *bitcoins* de rescate habían sido pagados. Esto provocó un crecimiento explosivo en los pagos de rescate, que alcanzaron más de \$325 millones a finales de 2015 [Has19b]. CryptoLocker fue el primer caso de propagación de ransomware a través páginas web infectadas mediante de la *botnet* llamada Gameover Zeus. No obstante, CryptoLocker también se difundía mediante *spear-phishing* o en correos electrónicos no deseados como un archivo adjunto [Gro19]. La desaparición de CryptoLocker por el cierre de sus servidores [Cat19], condujo a la aparición de varias imitaciones como CryptoWall y TorrentLocker.

En 2015, CryptoWall superó a CryptoLocker como la versión líder de ransomware, que aprovechaba una vulnerabilidad de Java y se distribuía a través de anuncios maliciosos. En este mismo año aparecieron *Ransomware as a Service (RaaS)*, un modelo por el que los atacantes distribuían su ransomware mediante la web *The Onion Router (TOR)*, repartiéndose los beneficios entre los autores y los grupos que realizaban el ataque. CryptoWall fue una de las familias más utilizadas entre abril de 2014 y principios de 2016, habiendo ganado más de \$18 millones [Gro20]. En la Figura 2.3, realizada con los datos de un estudio de Kaspersky [Kas16], se puede ver la distribución de todas las variantes de ransomware que surgieron entre los años 2014 y 2015, siendo CryptoWall el más popular después de la caída de CryptoLocker.

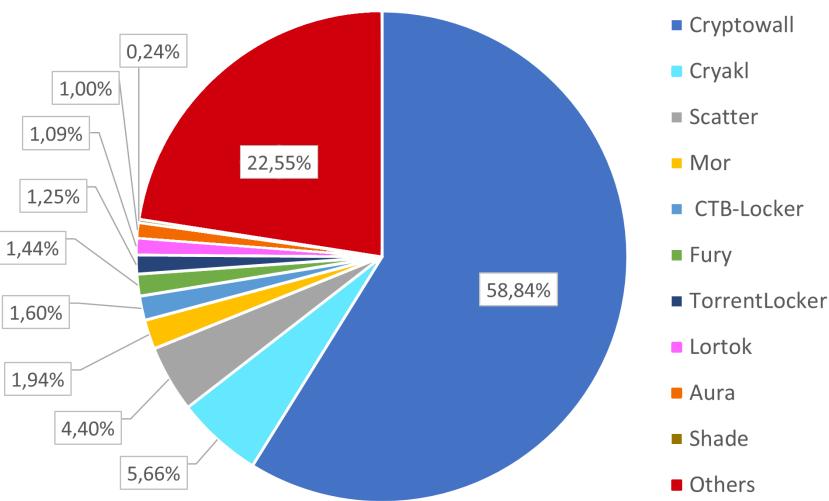


Figura 2.3: Familias de ransomware más observadas entre 2014 y 2015.

En mayo de 2016, ZDNet publicó un artículo [Pal16] donde informaba que las tres principales familias de ransomware más usadas a principios del año fueron Teslacrypt (58,4 %), CTB-Locker (23,5 %) y Cryptowall (3,4 %). A mediados de 2016 surgió Locky y se hizo con el primer puesto, superando a CryptoWall y a Teslacrypt a principios de febrero y solo siendo adelantado por Cerber en junio. El Gráfico 2.4 se ha realizado con los datos de una investigación realizada por PhishMe [Phi16] que estudia todos los hechos mencionados anteriormente.

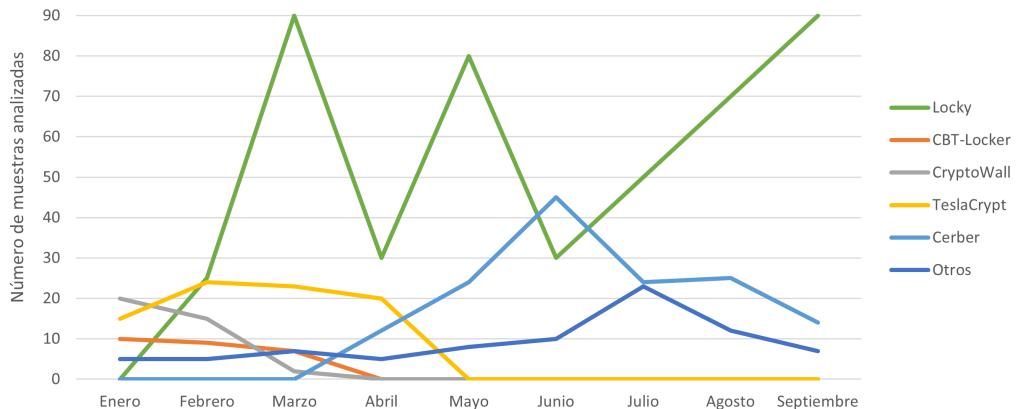


Figura 2.4: Familias de ransomware más observadas desde enero hasta septiembre de 2016.

El año 2017 se consideró por muchos expertos como el “año de oro del ransomware”, ya que el ataque del ransomware WannaCry fue una epidemia global que sucedió en mayo de ese año. Este ransomware cifraba los archivos de los usuarios, pidiendo \$300 en *bitcoins* por el rescate, y posteriormente esta cifra aumentó hasta los \$600. El 12 de mayo de 2017, WannaCry atacó a sus primeras víctimas en España, en concreto la empresa Telefónica. Días después, el número de usuarios infectados fueron alrededor de 230.000 en más de 150 países, convirtiendo a WannaCry en el mayor ataque de ransomware de la historia. Su rápida propagación es debido a que es gusano criptográfico, capaz de replicarse y propagarse automáticamente, aprovechándose de una debilidad en el sistema operativo Microsoft Windows. Este ransomware causó serios daños a bancos, al transporte público,

a universidades y a los servicios nacionales de salud, siendo los de Reino Unido los más afectados con costes de alrededor de 92 millones de libras y más de 19.000 citas canceladas [Hol20]. Además, WannaCry fue el primer ransomware que atacó a los dispositivos IoT, en específico a 55 cámaras de tráfico [Zho17]. En definitiva, WannaCry tuvo un impacto importante y se estima que causó pérdidas de alrededor de \$4 billones en todo el mundo [Kas21b].

En 2018 hubo una disminución en los ataques de ransomware. Según los informes publicados por Kaspersky [Kas19] y Malwarebytes [Mal19a], la infección por ransomware cayó un 30 % en todo el mundo, aunque aumentó el número de ataques dirigidos a organizaciones específicas y a dispositivos IoT. Estos ataques fueron perpetrados por nuevas variantes de ransomware más sofisticadas, como Ryuk y SamSam. Ryuk atacó a empresas que no se podían permitir períodos de inactividad, como periódicos diarios [Smi19] y una empresa de agua de Carolina del Norte que luchaba contra las secuelas del huracán Florence [Smi18]. SamSam atacó a organizaciones de salud, de educación y de gobierno, explotando el protocolo *Remote Desktop Protocol (RDP)* mediante ataques de fuerza bruta y cifrando todos los archivos del sistema. Se estima que el creador de este ransomware ganó alrededor de \$6 millones, ya que atacó a organizaciones tan grandes como el gobierno de Atlanta, que terminó pagando el rescate [Don18].

En 2019 se mantuvo la tendencia de lanzar ataques contra organizaciones con gran poder económico y Ryuk siguió siendo un ransomware muy popular entre los criminales junto con Sodinokibi. Sodinokibi o REvil, una variante de GandCrab, fue responsable del cierre de más de 22 pequeñas ciudades de Texas [Mic19] y del ataque a una compañía de cambio de divisas llamada Travelex en la víspera de año nuevo. Los delincuentes pidieron \$6 millones por el rescate, pero la empresa no pagó [Abr20].

La Figura 2.5, creada con datos sacados de un estudio realizado por Crypsis [Cry20], muestra la media de los rescates pedidos por los delincuentes en los Estados Unidos. Se aprecia un incremento del 140 % entre los dos años, 2018 y 2019 y esto evidencia la situación de que cada vez el ransomware es más sofisticado y ataca a objetivos más grandes con el objetivo de obtener mayores cantidades de dinero. El aumento de la cuantía de los rescates también es debido a que las compañías pagaban a los criminales, lo que les incentivaba a atacar más. En 2018 el 38 % de las empresas afectadas pagaron la suma exigida y en 2019 el 45 %.

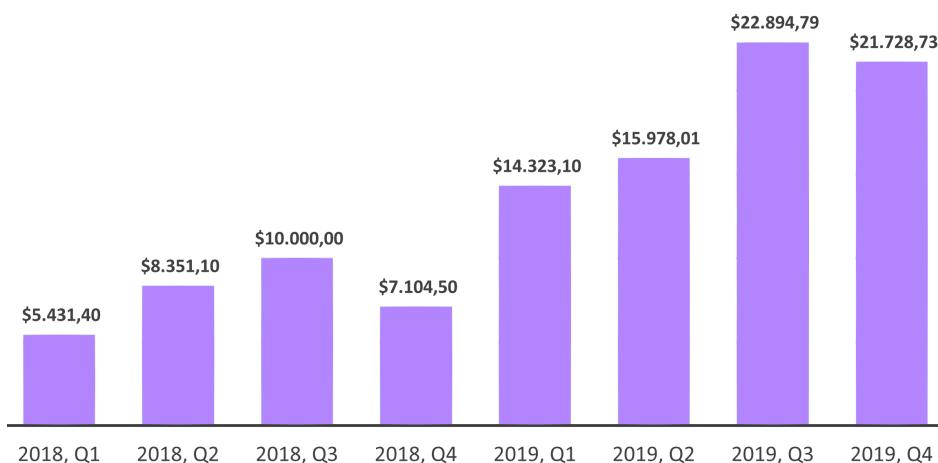


Figura 2.5: Media de la cantidad de dinero que los *hackers* pedían por los rescates en los Estados Unidos (2018-2019).

En 2020 el ransomware continúa la tendencia de los ataques dirigidos a organizaciones, pero empleando el método de la doble extorsión. No solo cifran y roban los datos, también amenazan con publicarlos si no se pagaba el rescate. Esto supuso una seria amenaza especialmente para las organizaciones más grandes que tienen información altamente confidencial en sus sistemas, y dichos ataques les podrían costar mucho dinero y dañar su reputación e imagen [LMMT21]. Maze es un ejemplo de una familia de ransomware que utilizó esta táctica, distribuyéndose de diferentes maneras. La principal fue a través de campañas de *spear-phishing* que instalaban un malware llamado *Cobalt Strike*, pero también infectaba a sistemas con credenciales débiles de *RDP* mediante la explotación de servicios vulnerables de internet para penetrar las redes [FS20]. Un estudio realizado por Sophos [SRRSA20] muestra datos con los que se ha realizado la Figura 2.6, que indica el volumen de menciones sobre el **COVID-19** en correos no deseados después de haber sido declarado como pandemia global por la [La Organización Mundial de la Salud \(OMS\)](#).

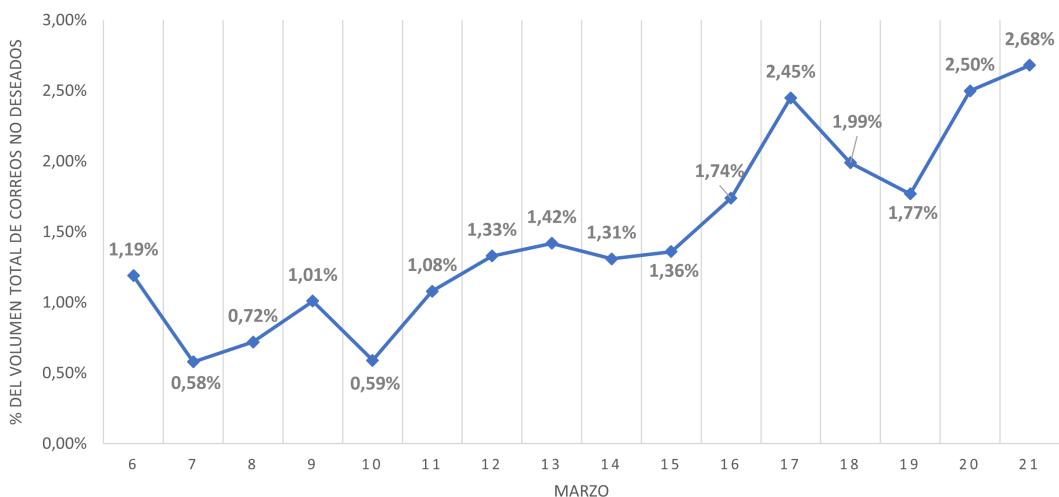


Figura 2.6: Volumen de menciones del COVID-19 en correos no deseados de todo el mundo en el mes de Marzo.

El primer ransomware del 2021 es Babuk, apareciendo por primera vez en enero y afectando especialmente a empresas de España y Chile. Este ransomware usa su propio esquema de cifrado llamado *ChaCha8*, una variante de Sodinokibi, usando criptografía de curva elíptica. Los precios del rescate oscilan entre \$65.000 y \$85.000 [Nie21].

En vista de la constante evolución del ransomware, es evidente que las organizaciones son las que están en peligro constante debido a su alto poder económico y a su necesidad de estar operativas todo el tiempo. No se pueden permitir que un ransomware bloquee sus sistemas y tienden a pagar los rescates, sin darse cuenta de que eso solo incentiva a los criminales y no hay certeza de que lleguen a recuperar todos sus archivos y datos afectados. Los criminales pueden intentar negociar y sacar más dinero al ver que la empresa está dispuesta a pagar, por lo que la prevención es clave y la única solución factible.

La Figura 2.7 muestra la cronología de la evolución del ransomware explicada anteriormente, con los ransomware más destacados y los años en los que surgían.

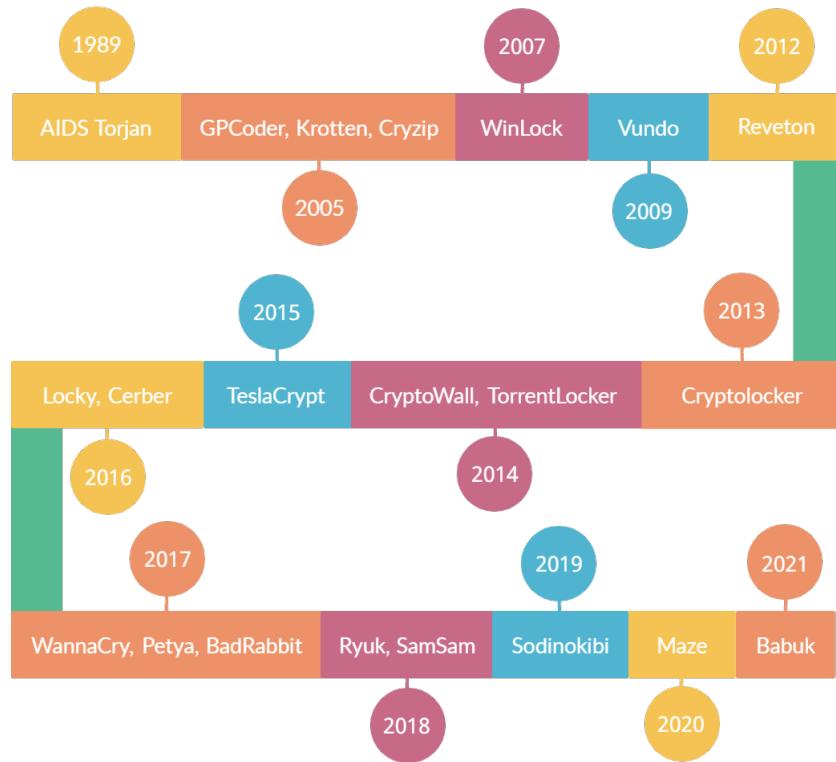


Figura 2.7: Cronología de la evolución del ransomware.

2.4. Tipos de Ransomware

Se identifican dos tipos de ransomware dependiendo de la técnica que utilicen para atacar los sistemas [Has19a]:

1. **Lockscreen:** Se caracterizan por que bloquean el acceso al sistema a través de la pantalla. En ella se muestra un mensaje indicando que el equipo ha sido infectado y el rescate que deben pagar e imposibilitan al usuario a salir de dicha pantalla. Este tipo de ransomware solamente bloquea el acceso a la interfaz del ordenador pero no afecta los archivos. Por lo tanto, existe la posibilidad de eliminar el ransomware y mantener todos los archivos intactos [Can14].
2. **Filecoder:** Estos ataques cifran archivos del usuario impidiendo el acceso de los usuarios a su información. Para obtener la clave de descifrado, los usuarios deberán pagar un rescate. Criptolock es el más famoso [Sec13].

En los últimos años además de estos dos tipos han surgido otras variedades que, aunque similares, tienen sus peculiaridades y es importante diferenciarlos para poder comprender su comportamiento y clasificarlos de mejor manera [Sec19]:

- **Virus de la policía:** Este ransomware es similar al *Lockscreen*, ya que bloquea el acceso al equipo. La diferencia es el mensaje que muestra al arrancar el sistema, siendo este un mensaje falso de la policía, informando al usuario que se ha detectado accesos a páginas ilegales y que tiene que pagar una multa.

- **Wiper:** Paredico al Filecoder pero más destructivo, ya que bloquea definitivamente el acceso a los archivos y posteriormente los elimina.
- **Scareware:** Se hace pasar por un detector de amenazas utilizando ventanas emergentes para informar al usuario sobre supuestos problemas encontrados en su equipo. En lugar de exigir dinero, este ransomware presiona al usuario para que compre un software antivirus falso para resolver todos los problemas instantáneamente. Si el usuario paga e instala el supuesto antivirus, este actuará como malware y recopilará información del equipo y datos personales del usuario [Kas21d].
- **Hoax ransomware:** Este ransomware es similar a un Filecoder, pero no cifra ningún archivo ni bloquea el equipo. Simplemente utiliza técnicas de ingeniería social para asustar al usuario y hacerle pagar un rescate falso.
- **Leakware:** También llamado *doxware*, este ransomware amenaza al usuario con la publicación de su información personal, como datos confidenciales y conversaciones almacenados si no paga un rescate. Este ransomware es peligroso para las empresas, ya que puede revelar información comprometedora [MEZ21].
- **Ataque oculto:** Sucede cuando un usuario simplemente navega a través una página web vulnerada por un *hacker* [Kas21c].
- **Dropper:** Infecta al sistema al instalar un programa que en realidad es malware disfrazado, y esto se denomina instalador (*dropper*) de malware. Bad Rabbit, un ransomware que surgió en 2017, se propagó haciendo uso de una solicitud falsa para instalar Adobe Flash, que en realidad era un instalador de malware [Inc20].

Hay otros tipos de ransomware que no tienen un nombre en concreto, pero es conveniente mencionarlas. La reescritura del *Master Boot Record* (MBR) antes de cifrar los datos es una estrategia inusualmente agresiva que usa el ransomware CoViper, y el *Full Disk Encryption* (FDE) lo realiza el ransomware Mamba. Petya hace algo similar, pero solo cifra la *Master File Table* (MFT), por lo que los datos en sí no se ven afectados [Kas21c].

Finalmente, cabe mencionar otras técnicas de extorsionar a las víctimas [Has19a]:

1. **Scareware:** Este tipo de malware funciona convenciendo a las víctimas para comprar artículos que no necesitan mediante alertas de seguridad falsas.
2. **Extorsión DDoS:** Puede ser considerado *blackmail* cuando los atacantes solicitan dinero por no realizar un ataque distribuido de denegación de servicio.
3. **Extorsión por datos comprometidos:** El cibercriminal amenaza a la víctima con exponer datos sensibles públicamente a no ser que éste pague un rescate.

2.5. Familias de Ransomware

A continuación se listan algunas de las familias más peligrosas y con mas impacto [Col18] [Tor16]:

- **Petya:** Tiene dos variantes, una de 2016 y otra, llamada NotPetya, que apareció en 2017. Una vez ejecutado, Petya sobrescribe el sistema de arranque, posteriormente

cifra la tabla de ficheros maestros, provocando que Windows sea incapaz de localizar los ficheros almacenados. Posteriormente se reinicia el ordenador, y Petya evita que Windows arranque con normalidad y muestra un mensaje de extorsión como el que se puede observar en la Figura 2.8, procedente del análisis de Cuckoo Sandbox de una muestra realizada en este estudio. NotPetya es muy peligroso puesto que puede ser propagada con intervención humana y las claves de cifrado son generadas aleatoriamente y destruidas, lo que hacen que la recuperación de los datos sea imposible [Has19a].

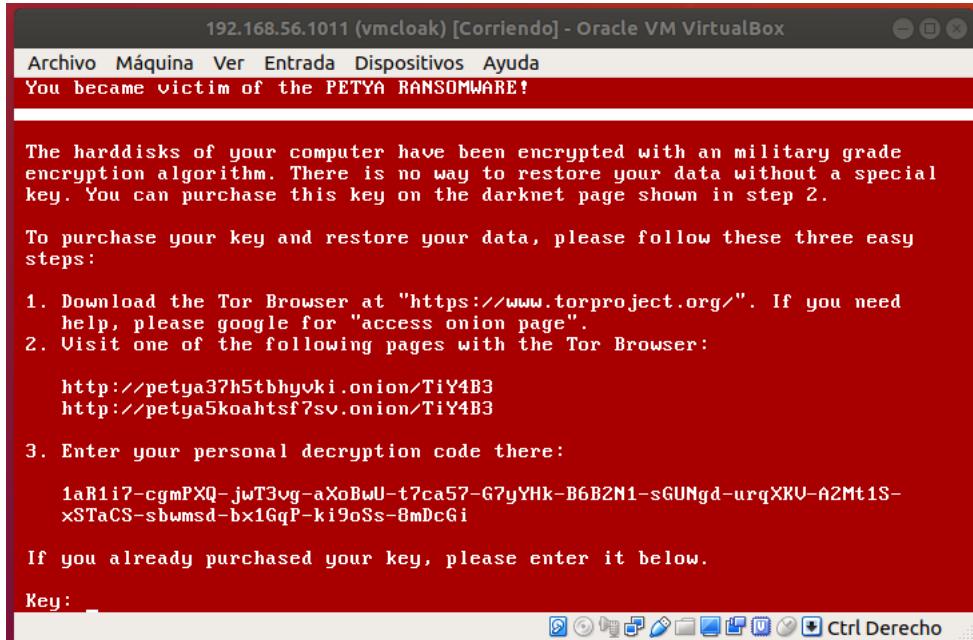


Figura 2.8: Aviso de extorsión del ransomware Petya

- **Locky:** Los usuarios se infectan debido a un correo electrónico. Utiliza macros y archivos de *script* para cifrar archivos cuando se abre un documento. Al ejecutarse el *script*, los archivos cifrados se renombran y Locky muestra al usuario una nota de rescate, configurándola como fondo de pantalla. Locky utiliza un algoritmo de cifrado RSA-2048+AES-128 para cifrar archivos, discos duros e incluso dispositivos extraíbles. Locky ataca a máquinas con base Windows, eliminando el sistema Windows *Shadow Volume Copy* para prevenir cualquier posible recuperación de archivos por parte del usuario.
- **SamSam:** Es usado para atacar a una organización, después de haber realizado un estudio y reconocimiento de sus sistemas IT, para reconocer alguna vulnerabilidad en la red o equipos. Afecta a una máquina y se propaga por red a todos las conectadas. Se diferencia a otras familias porque no utiliza técnicas de ingeniería social como correos electrónicos de *spam* o *phising*, sino que los atacantes realizan ataques de fuerza bruta contra las contraseñas débiles para ganar acceso a la red de la organización tras el estudio previamente comentado [Has19a].
- **CryptoWall:** Se trata del sucesor de CryptoLocker. Su objetivo son máquinas de base Windows, y se propaga mediante e-mails maliciosos y kits de *exploits* como Nuclear o Angler. Una vez ejecutado en el equipo, escribe su propio registro de

auto-arranque, de modo que es resistente a los reinicios. Elimina los sistemas de restauración y procede a cifrar los archivos mediante el algoritmo RSA-2048. Su versión más reciente, CryptoWall 4.0 usa un esquema de comunicación con la red TOR a través de un servidor *proxy* externo [Has19a], como se puede observar en la Figura 2.9.

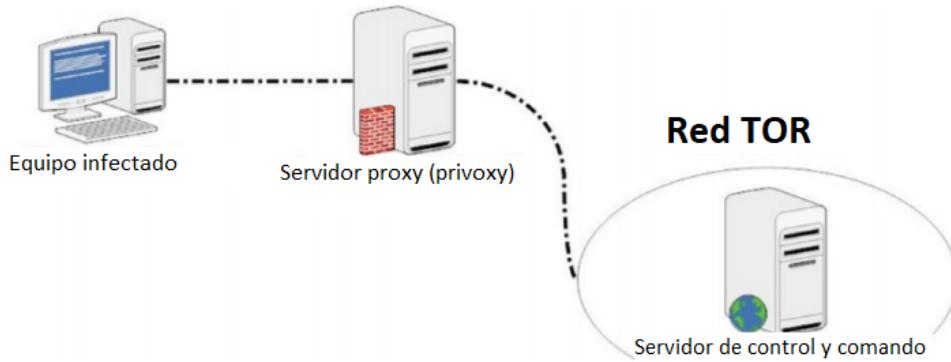


Figura 2.9: Esquema de red de un ataque con Cryptowall

- **CryptXXX:** Familia de ransomware de cifrado que ataca a sistemas operativos Windows. No cifra únicamente los archivos, sino que también ejecuta otro malware de robo de información, como por ejemplo credenciales de VPN, *cookies* o incluso *bitcoins* del ordenador de la víctima y es enviado de nuevo al servidor de control y comando. Utiliza el algoritmo de cifrado RSA-4096 y añade la extensión “.crypt” al nombre del fichero. Posee un sistema anti-sandboxing y anti-analisis [Has19a].
- **Cerber:** Se diferencia al resto porque es capaz de emitir por los altavoces del dispositivo mensajes de alerta de infección del malware. Incluye ataques a plataformas *Cloud* y *Windows Scripting* y ataques *Distributed Denial of Service (DDoS)*. No solo ataca a máquinas de usuarios individuales, sino que también ha llegado a cifrar bases de datos enteras de empresas.
- **Jigsaw:** Debe su nombre a que utiliza como imagen de bloqueo de pantalla al protagonista de la película *Saw*. Infecta archivos de manera exponencial cuanto más tiempo pase sin que la víctima pague el rescate. Posee una característica peculiar que habilita un “chat” que ayuda al usuario a pagar para recuperar sus datos.
- **CryptoLocker:** Es el primero en utilizar *bitcoin* como moneda de rescate. Accede al sistema utilizando técnicas de ingeniería social, mediante correos electrónicos de empresas de paquetería. Se trata de un ransomware que combina las técnicas de Lockscreen y CriptoLocker.
- **WannaCry:** Se diferencia por su éxito de infección, más de 300.000 equipos en más de 150 países fueron afectados por este ransomware debido a una vulnerabilidad de Windows que permite a un atacante remoto ejecutar código arbitrario [NC16]. La dificultad de protegerse contra WannaCry reside en que usa un componente de gusano, que provoca que el ataque sea mucho más efectivo y se necesiten mecanismos de defensa que puedan reaccionar en tiempo real [AVL19].

- **HDDCryptor:** Bloquea la máquina además de afectar a recursos compartidos en la red como impresoras, puertos y archivos a través de *Server Message Block*. Ataca tanto a usuarios individuales como a empresas, y un ejemplo conocido es Mamba.
- **Crysis:** También conocido como Dharma, es un ransomware que utiliza ataques de fuerza bruta para propagarse. En 2017, Dharma consiguió que se duplicaran este tipo de ataques por todo el mundo. Esta familia se ha encontrado en 123 países, concentrándose el 60 % de los ataques en 10 países, entre los cuales España ocupa la segunda posición [Ccn18].

En el Anexo B se presenta una lista extendida de las familias de ransomware más conocidas [Mic20a].

2.6. Métodos de Propagación

En esta sección se exponen los diferentes vectores de ataque usados por los creadores de ransomware para invadir los sistemas víctimas [Has19a].

- **Ataque de fuerza bruta:** Este ataque sucede cuando los criminales emplean el RDP y técnicas para probar una gran variedad de combinaciones de contraseñas, con el objetivo de revelar las credenciales del usuario y así lograr acceso a su sistema [Kas21a].
- **Auto-propagación:** Es la propagación del ransomware de un sistema infectado a otro, difundiendo archivos infectados a todos los contactos disponibles en esa máquina. Antes de iniciar el proceso de cifrado, se expande rápidamente para aumentar sus posibilidades de infectar otros ordenadores [AAI17].
- **Dispositivos USB:** Aunque se trate de un método de propagación primitivo, los cibercriminales siguen utilizándolo para realizar ataques a sistemas. Requiere el acceso físico a algún puerto donde poder conectar el dispositivo ejecutor, aunque existe alguna técnica como esparcir memorias USB por diversos lugares con la expectativa de que algún usuario los conecte.
- **Drive-by-Downloads:** Los atacantes comprometen sitios web legítimos e incrustan código malicioso en sus páginas, como código *JavaScript*. Este código malicioso realiza publicidad maliciosa, redirectiones engañosas, ataques *Cross-site scripting (XSS)* u otros ataques en secreto, sin que el usuario sepa que su sistema está siendo infectado [DK17] [AMK16].
- **E-mail:** Es la manera más utilizada, con un 59 %, según un estudio de IBM de 2017 [Vad]. Cuando los descuidados usuarios descargan y abren el contenido malicioso, el sistema se infecta de inmediato. También pueden contener *Uniform Resource Locator (URL)* a páginas web maliciosas que infectan el computador. Suelen realizarse mediante campañas de *spam*, e-mails no solicitados enviados a grandes grupos de personas, con el objetivo de realizar promociones. Se trata de algo convincente puesto que, los mensajes de *spam* se contabilizaban como el 53.5 % del tráfico mundial en 2018 [Sta]. La técnica más utilizada es el *phising*. Se basa en realizar campañas de *spam* usando el mismo formato que alguna compañía legítima en sus comunicaciones para recaudar información de la víctima.

- **Botnets troyanos:** El ransomware también puede llegar a un sistema a través de otro malware. Este es el caso de los *Botnets*, que son una red de robots que se ejecutan de manera automática y autónoma. El término *Botnet* surge con una combinación de dos palabras, “bot” de robot y “net” de red en inglés. Estos robots controlan a los equipos que infectan para que realicen varias tareas, tales como: atacar a otros ordenadores, enviar correos electrónicos no deseados a los contactos del usuario, entregar ransomware, instalar *spyware* y otras actividades maliciosas similares [UKA13]. La Figura 2.10 muestra un diagrama de acción de una *Botnet*.

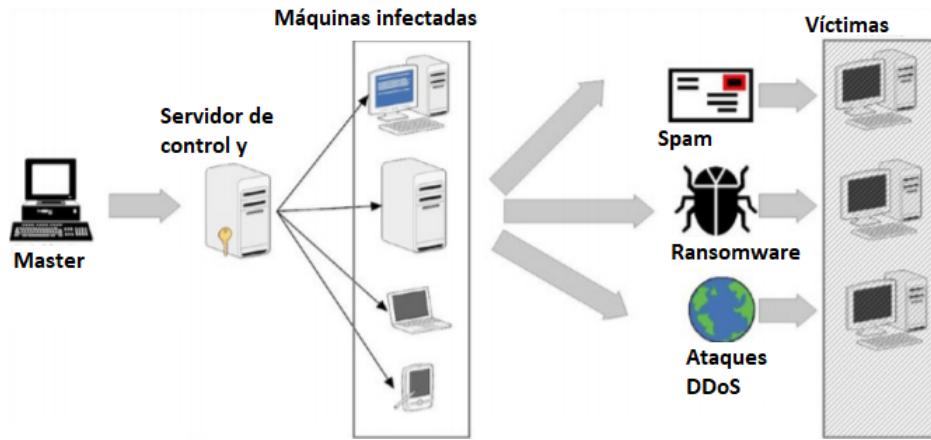


Figura 2.10: Diagrama de acción de un botnet

- **Explotar vulnerabilidades de servidores:** Los criminales explotan las vulnerabilidades del software que se ejecuta en servidores para obtener acceso a la red de una organización y difundir su malware a través de ella [AAII17].
- **Kits de Exploit:** Se definen como plataformas web que permiten ejecutar *exploits* de manera semi-automática mediante la detección de vulnerabilidades en el sistema operativo o la aplicación víctima y comparándolas con las almacenadas en el kit. Se componen de cuatro partes:
 1. **Página de destino:** Contiene el código responsable para escanear las vulnerabilidades a explotar.
 2. **Puerta:** fragmento de código que recibe la información de la página de destino y decide si continuar con el ataque o no, según los criterios predefinidos.
 3. **Exploit:** Explota las vulnerabilidades previamente detectadas ejecutando algún tipo de código malicioso en la máquina víctima para poder acceder a ella.
 4. **Carga:** Tras haber accedido mediante las vulnerabilidades, el kit de *exploit* envía la carga, algún tipo de malware o algún mecanismo que lo descargue, para realizar el daño real.
- **Malvertising:** Anuncios maliciosos distribuidos a través de páginas web confiables por atacantes que compran espacios publicitarios. Los usuarios que acceden a esas páginas ni siquiera tiene que hacer clic en el anuncio en algunos casos, ya que simplemente cargar la página web que aloja la publicidad maliciosa provocará una infección [OPW16]. Esto era muy común cuando los sitios web usaban Adobe Flash para reproducir anuncios animados [Don14].

- **Macros de Microsoft Office:** Son una serie de instrucciones y comandos para automatizar tareas en programas de Microsoft Office como Word o Excel. Los cibercriminales distribuyen por la red ficheros de estos programas con macros maliciosas en ellos y, cuando el usuario los abre, la macro ejecuta funciones malignas para la máquina víctima.
- **Phishing y técnicas de ingeniería social:** El *phishing* es un ciberataque que usa correos electrónicos como arma, siendo el método más efectivo para distribuir ransomware [Bra17]. Su objetivo es recopilar información personal del usuario mediante correos electrónicos o sitios web maliciosos, engañándole para que acceda a un enlace o que descargue un archivo adjunto. Lo que distingue al *phishing* es la forma del mensaje: los atacantes se disfrazan como una entidad confiable, a menudo una empresa o persona real con la que la víctima podría hacer negocios [Fru20]. Otra técnica de ingeniería social usada comúnmente por los delincuentes es engañar al usuario para que descargue un antivirus falso, y al escanear el equipo con este programa, será infectado por el ransomware.
- **Ransomware as a Service:** RaaS es un servicio de ventas de paquetes software de ransomware establecido en mercados negros o foros usando un modelo de suscripción. Se trata de una plataforma emergente muy dañina que permite adquirir software malicioso de manera sencilla. Intervienen tres actores: el autor del código, un proveedor de servicios y los agentes que los adquieren, los futuros atacantes.

2.7. Modo de Operación

Los ataques ransomware siguen las siguientes fases desde que se instalan en el equipo víctima hasta que el usuario puede advertir su presencia [Has19a]:

1. **Infección:** El ransomware es introducido en el sistema mediante las técnicas detalladas en la Sección 2.6.
2. **Ejecución:** Si la infección fue satisfactoria, el ransomware se instala en el equipo víctima y se conecta con el C&C mediante el protocolo criptográfico *Transport Layer Security (TLS)* y le empezará a enviar información de la víctima de forma segura [ZL⁺16].
3. **Destrucción del respaldo:** El ransomware realiza una búsqueda de los ficheros de respaldo y los elimina, de manera que el usuario sea incapaz de recuperar sus archivos. En paralelo, el ransomware también elimina las copias de seguridad del sistema y dependiendo del tipo, también podrá eliminar las *shadow volume copies* (instantáneas) del sistema o cambiar los nombres de los archivos, agregando una extensión relacionada con el nombre de la familia del ransomware [AAII17].
4. **Cifrado:** Se establece una conexión segura con el servidor C&C para que le proporcione al ransomware una clave simétrica generada aleatoriamente y este empezará a cifrar los archivos del usuario, usando el algoritmo de cifrado asimétrico RSA. Este algoritmo usa dos claves diferentes, una pública para el cifrado de los datos y otra privada para el descifrado [BTPS16]. También pueden usar el algoritmo AES, siendo AES-256 la forma más común [MD16].
5. **Rescate:** El ransomware reproduce un mensaje de extorsión en la pantalla del ordenador de la víctima, estableciendo un coste y un período de tiempo.

Los atacantes pueden intentar minimizar la detección del ransomware mediante el uso de certificados digitales legítimos, comprados o robados, para intentar convencer a los software antivirus de que el código es fiable y no necesita ser analizado [Lom19]. Aún así, este método podría fallar, ya que este certificado puede ser revocado en cualquier momento por las autoridades pertinentes, por lo que el ransomware debe actuar rápidamente. Para conseguir rapidez y efectividad, el ransomware dispone de un mecanismo que le ayuda a diferenciar qué archivos son los más importantes para robarlos y cifrarlos primero. La importancia de estos archivos se mide viendo lo relevantes que son para el usuario, comparando cuando se accedieron por última vez y cuando se crearon [KRB⁺15], dónde están ubicados (el directorio “Documentos” suele contener archivos importantes) y su entropía [KAM⁺16a]. Si la entropía es demasiado alta o baja, es porque el archivo contiene información aleatoria o simplemente relleno, por lo que el ransomware interpretará que el archivo fue generado automáticamente por el sistema y lo descartará [HJA19]. Otro método más comúnmente utilizado para evitar ser detectado es el aumento de los privilegios mediante el uso de *exploits*, permitiendo a los atacantes instalar programas como *Remote Access Terminal Server (RATS)* (herramientas de administración en remoto) para cambiar o borrar datos, crear nuevas cuentas con todos los derechos de usuario y lo más importante, desactivar el software de seguridad [Lom19].

Todo lo explicado anteriormente es teniendo en cuenta que el ransomware es de tipo Filecoder. Si es de tipo Lockscreen, sigue el mismo patrón excepto por el cifrado de los datos. Después de haber obtenido los privilegios de administrador y de haber enviado la información robada al servidor C&C, bloquea el sistema y muestra una alerta al usuario con los pasos a seguir para recuperar el acceso [ZL⁺16].

En la Figura 2.11 se representa el patrón de ataque de un ransomware explicado anteriormente, desde que infecta al equipo hasta que demanda un pago por el rescate.



Figura 2.11: Patrón de ataque de un ransomware.

2.7.1. Métodos de Cifrado

El ransomware moderno, como WannaCry, Petya o Locky, utiliza esquemas de cifrado híbrido, dificultando aún más la labor de recuperar la información cifrada en el ataque [Mar18]. A continuación se exponen los métodos de cifrado de archivos más importantes [Sta17]:

1. **Cifrado simétrico:** También denominado cifrado de clave secreta. El texto plano se cifra y se descifra con la misma clave. La seguridad del cifrado no depende del algoritmo, si no de la privacidad de la clave. Permite que el ransomware cifre los datos a gran velocidad, y posteriormente almacena en disco las claves para almacenar cada archivo. Existen dos tipos de algoritmos simétricos:

- **Cifrado de bloque:** Procesa la entrada del texto en bloques de tamaño fijo y produce un texto cifrado de igual tamaño. Los más importantes son *Data Encryption Standard (DES)*, *Triple DES (3DES)*, vistos en la Figura 2.12 y *AES*.

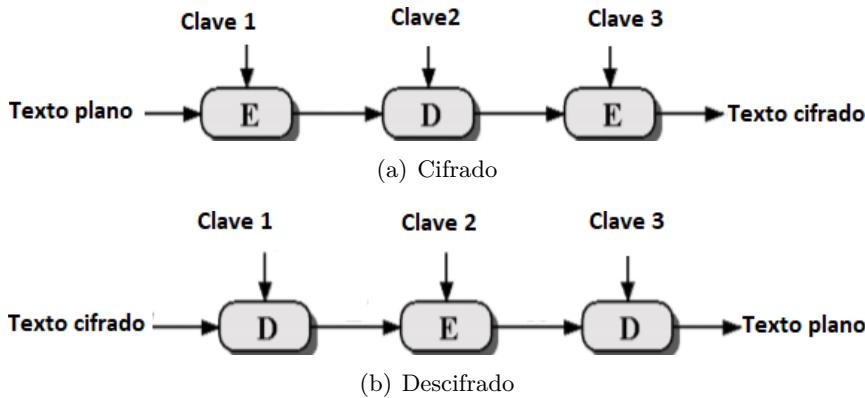


Figura 2.12: El algoritmo 3DES aplica 3 veces el algoritmo DES con claves distintas

- **Cifrado de flujo:** Procesa elementos de entrada y produce un elemento de salida. El más importante es *Rivest Cipher 4 (RC4)*.
2. **Cifrado asimétrico:** También denominado cifrado de clave pública. Está basado en funciones matemáticas en lugar de usar operaciones binarias sobre patrones de bits, por lo tanto este cifrado es más lento. Usa dos claves distintas: una pública y otra privada. El ransomware utiliza la clave pública para cifrar los archivos, y la clave privada es enviada al servidor de mando y control para ser almacenada, por lo que se necesita una conexión a internet. Los algoritmos más utilizados son *Digital Signature Algorithm (DSA)*, *RSA* y *Diffie-Hellman (DH)*. Se diferencian las siguientes aplicaciones:
 - **Cifrado con clave pública:** El emisor cifra con la clave pública del receptor y el receptor descifra con la clave privada.
 - **Intercambio de clave:** Dos partes cooperan para establecer una clave secreta compartida.
 - **Firma digital:** técnica utilizada para asegurar la autenticación y la integridad de los datos.
 - **Certificado digital:** Se utiliza para proporcionar autenticación de las partes y no repudio
 3. **Cifrado híbrido:** Usa cifrado simétrico y asimétrico. No necesita conexión a internet en el cifrado, pero sí en el descifrado. El ransomware y el servidor de mando y control generan un par de claves. Se cifrará la clave privada del cliente utilizando la clave pública del servidor. Al finalizar, todas las claves serán cifradas con la clave pública del cliente. Un ejemplo es el algoritmo utilizado por Locky: *RSA-2048+AES-128* [Mar18].

2.8. Detección

Las técnicas usadas por los software de antivirus para reconocer la presencia de ransomware en el equipo son las siguientes [Has19a]:

- **Detección basada en firma digital:** Cuando la compañía del antivirus descubre el malware crea una firma única para el tipo en concreto, y actualiza la base de datos del antivirus para poder comparar y capturar cuando el malware intente acceder a dicho equipo. No puede detectar avanzados tipos de malware polimórficos o que utilizan técnicas de cifrado para evadir dicha detección.
- **Análisis del comportamiento:** Esta técnica trata de identificar el malware mediante los cambios que produce en el sistema, como puede ser la eliminación de ficheros, el aumento de la entropía (para cifrar los ficheros), las conexiones a redes extrañas, etc. Para recopilar estos datos, una técnica es registrar las llamadas a las APIs, los recursos del sistema que se están utilizando, los archivos que abre cada proceso, e introducirlas en una solución de ML que sea capaz de detectar si esa serie de llamadas pueda corresponder a la ejecución de un ransomware [ADPC20a].
- **Detección heurística:** Se basa en reglas y algoritmos para analizar si el código es malicioso. Puede realizarse un análisis estático, mediante un estudio de los indicadores sin ejecución como anomalías estructurales del código, el desmontaje del programa o el estudio de los elementos de una secuencia dada (n-gramas). Análogamente, el análisis dinámico, realiza un estudio de los indicadores del código en ejecución mediante técnicas de *sandboxing* o virtualización [DRP⁺12].
- **Detección usando la nube:** El archivo sospechoso se envía a un antivirus en la infraestructura de la nube del vendedor, aprovechando la capacidad computacional de la plataforma para analizarla de manera más rápida y eficiente.

Además de las técnicas usadas por los antivirus y debido a que un ransomware puede cifrar archivos en cuestión de minutos, es importante implementar una detección a tiempo real tanto en el ordenador del usuario como en servidores web. Para implementar esta detección es importante seguir estos pasos [MM20]:

- **Implementar escáneres de vulnerabilidad:** Se necesita saber en todo momento la información del sistema y de sus archivos tanto en local como en la nube.
- **Detección de intrusiones:** El ransomware es difícil de detectar antes de que ataque, pero un buen sistema de detección de intrusiones puede ser lo suficientemente rápido como para frenarlo antes de que cifre algún archivo. Un sistema de detección de intrusiones puede detectar comportamientos característicos de un ataque de ransomware en sus etapas iniciales [MOB17]. Algunos ejemplos de estos comportamientos son: deshabilitar el cortafuegos o el software antivirus, iniciar escáneres de red no autorizados, enviar datos a través de un canal encubierto y actualizar la política de auditoría.
- **Activar la monitorización de la integridad de archivos:** Una parte principal del malware, incluyendo el ransomware, es ejecutar procesos para obtener permisos que le permitan acceder a los archivos del sistema. Estos procesos no se ejecutan normalmente, por lo que la monitorización de la integridad de los archivos puede generar una alerta cada vez que los detecta. Esto no va a prevenir el inicio del cifrado de archivos, pero puede evitar futuros ataques y poder aislar y poner en cuarentena el sistema infectado [TG].

- **Implementar la automatización de la seguridad:** Una respuesta rápida es un factor crítico de éxito en cualquier tipo de emergencia. Las innovaciones en el sector de la automatización y gestión de la seguridad han mejorado significativamente la respuesta ante amenazas, asegurando el funcionamiento conjunto de varias herramientas de seguridad. Este enfoque puede proporcionar una desconexión inmediata de la red y el aislamiento de los sistemas cuando se detecten las actividades del ransomware [ÁR].
- **Gestionar y analizar los registros:** Los objetivos del atacante se encuentran en los registros del sistema, de las aplicaciones, de la actividad... Detectar el acceso a los registros es complicado debido a la gran cantidad de ellos, por lo que es necesario automatizar su gestión y análisis.
- **Combinar inteligencia sobre amenazas con la monitorización del sistema:** Los desarrolladores de ransomware más experimentados están en constantemente evolucionando sus métodos y código. Los analistas de seguridad deben investigar cuidadosamente su desarrollo, innovación e infraestructura para desarrollar métodos de gestión pro-activos (por ejemplo, software para la correlación de eventos) y herramientas para detectar los últimos ataques de ransomware.

2.9. Prevención

Se diferencian las estrategias de defensa que siguen las empresas contra los ataques ransomware y las acciones que han de seguir los usuarios finales para no estar expuestos [Has19a].

2.9.1. Defensa Individualizada

La seguridad del equipo del usuario final es clave para proteger la red entera de una organización, puesto que basta con infectar un equipo para que algunos tipos de ransomware puedan propagarse por la totalidad de ellos. Al contrario de lo que piensa mucha gente, no basta solo con instalar un antivirus, sino que hay que cumplir una serie de acciones para alcanzar un grado de seguridad superior [Has19a]:

- **Mantener respaldo de los datos:** Si los datos cifrados estaban replicados de manera consistente en otro dispositivo o en la nube, se pueden recuperar sin problema sin tener que pagar ningún rescate.
- **Mantener el sistema operativo y las aplicaciones actualizadas:** Sirven para arreglar las vulnerabilidades detectadas. El sistema operativo debe estar configurado para actualizarse tan pronto como la actualización esté disponible. Esto permitirá que el sistema sea más seguro frente a las vulnerabilidades, los kits de *exploit*, etc.
- **Uso de la virtualización:** El uso de máquinas virtuales permite ejecutar programas, descargarlos de internet y visitar páginas web comprometidas debido a que se ejecuta en un entorno seguro aislado del sistema operativo raíz. Aun así se debe mantener la precaución de que no el ransomware no se propague por la red en la que se encontraba la máquina virtual.
- **Bloquear la redirección web:** Esto evita páginas web no deseadas y enlaces maliciosos. Los navegadores actuales permiten esta configuración. También es buena práctica añadir extensiones al navegador para bloquear anuncios y *pop ups*.

maliciosos, incluso algunas como **NoScript**, que únicamente permite que se ejecuten los *plugins* en los sitios web establecidos por el usuario.

- **Deshabilitar macros de Microsoft Office:** Como se ha explicado en la Sección 2.6 esta característica puede ser explotada para introducir ransomware en el equipo víctima. Además, es aconsejable abrir los archivos de Office recibidos vía correo electrónico en otra plataforma como Google Docs.
- **Dispositivos USB:** Está totalmente desaconsejado conectar dispositivos desconocidos, poco fiables o encontrados en el sistema, puesto que puede contener ransomware que se instale al conectarlo. En algún caso, se debe configurar el antivirus para que escanee la unidad antes de montarla.
- **Modificar la extensión de los archivos importantes:** Algunos tipos de ransomware como WannaCry, cifran los ficheros basándose en una búsqueda por la extensión.
- **Remote Desktop Protocol:** Este protocolo permite el acceso remoto a un ordenador, de manera que si un atacante se conecta, podría infectar el equipo. Para restringir el acceso se recomienda utilizar una contraseña consistente, utilizar sesiones **Virtual Private Network (VPN)** fiables y asegurarse de permitir conexiones autenticadas a nivel de red (**Network Level Authentication (NLA)**).

2.9.2. Estrategias Empresariales

De manera complementaria a las medidas de seguridad que deben tomar los usuarios finales vistas en el apartado anterior, las organizaciones, empresas y gobiernos siguen una serie de estrategias de prevención en contra del auge de los ataques ransomware [Has19a].

- **Seguridad física:** La seguridad de los equipos digitales (servidores, almacenamiento del respaldo, ordenadores, dispositivos de red) es tan importante como la de los datos almacenados en ellos. Para asegurarlos, se deben cumplir una serie de reglas como la restricción del acceso con medidas biométricas, monitorizar y restringir los espacios, no dejar equipos desatendidos, desconectar de la red sistemas sin utilizar, etc.
- **Segmentación de red:** Es una medida para luchar contra el ransomware que limita la posibilidad del atacante de acceder al segmento donde se encuentra la información sensible. En la Figura 2.13 de pueden ver las ventajas de esta segmentación.

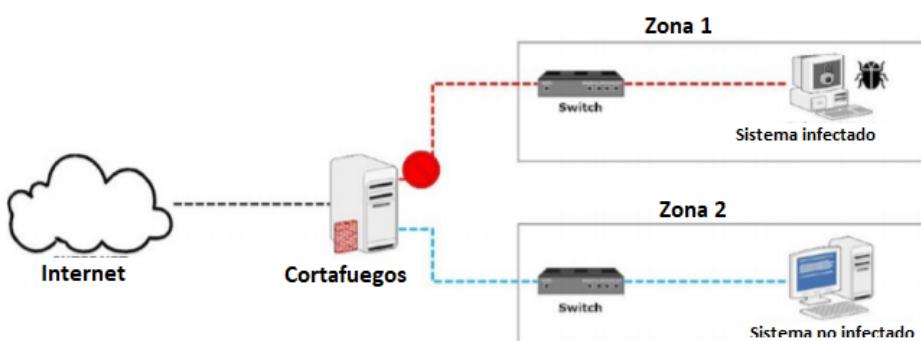


Figura 2.13: Ventajas de la segmentación de la red

- **Cortafuegos modernos:** Cualquier plan de seguridad comienza instalando un cortafuegos en el perímetro de la red, limitando las conexiones y permitiendo solo las permitidas. Los de nueva generación añaden otras herramientas de seguridad como antivirus, antimalware y el soporte para las redes privadas virtuales ([VPN](#)).
- **Seguridad de correo electrónico:** Puesto que es el acceso más utilizado por el ransomware se deben desarrollar sistemas de filtrado avanzado de *spam* mediante filtrado de [Internet Protocol \(IP\)](#) y contenido y deshabilitar el código [Hypertext Markup Language \(HTML\)](#) en los e-mails recibidos por texto plano.
- **Formación de los trabajadores:** Para la creación de un entorno seguro, los trabajadores de la organización deben ser conscientes del riesgo existente y poner a su disposición la formación necesaria y los hábitos que deben seguir, como utilizar contraseñas seguras, no acceder a enlaces ni descargas de e-mails desconocidos, no conectar dispositivos extraños, etc [RN17].
- **Honeypots:** Se tratan de señuelos con información falsa que se anuncian online como objetivos de valor para cibercriminales y que permiten la detección de los ataques. En la Figura 2.14 se aprecia el funcionamiento de honeypot.

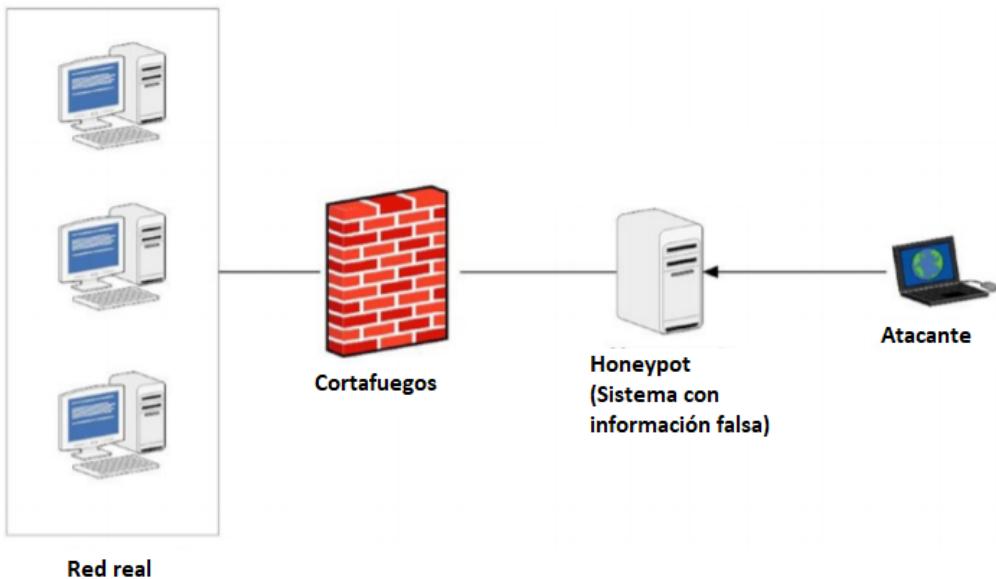


Figura 2.14: Funcionamiento de honeypot

2.10. Período Post-ataque

Una vez el sistema ha sido infectado, se plantea una cuestión fundamental: ¿se debe pagar el rescate para recuperar el equipo infectado? La respuesta de los expertos es clara: no. Por un lado, te convertirías en un mayor objetivo, puesto que esta información es compartida entre atacantes, y conociendo que ya has pagado una vez, podrás hacerlo en otras ocasiones. Además, aunque cada vez más los criminales están cumpliendo con devolver los archivos (un ejemplo de ello es Cryptolocker, que permite descifrar 5 archivos elegidos de manera gratuita) debido a que creando esta confianza, más gente pagará por el rescate, no se puede

establecer confianza en los cibercriminales y en que el equipo sea descifrado. Por último, pagar el rescate provoca que los cibercriminales tengan más motivos para seguir con su labor [RN17].

Si se ha decidido no pagar el rescate por los datos cifrados todavía se puede albergar alguna esperanza. Se debe realizar una valoración para conocer si se pueden recuperar los archivos cifrados siguiendo unos pasos [Ccn18]:

1. **Respaldo:** Si se dispone de una copia de seguridad, se procede a la desinfección del sistema y a la restauración de la copia de seguridad.
2. **Herramientas de descifrado:** Aunque existen pocas variantes de ransomware que se puedan descifrar, por la intervención del C&C y se han obtenido las claves, o porque existen vulnerabilidades en el código malicioso, hay disponibles herramientas públicas para descifrar archivos atacados por ransomware concreto, que se listan en la Tabla 2.1.
3. **Shadow Volume Copy:** Si el ransomware no ha deshabilitado esta función como se ha expuesto previamente, se podrían restaurar los datos que Windows duplica automáticamente estableciendo una copia de seguridad.
4. **Software forense:** En algunos casos es factible la recuperación de algunos ficheros originales mediante programas de análisis forense.

Tabla 2.1: Herramientas para la recuperación de datos

Ransomware	Recuperación de ficheros
Cerber	En las versiones 1 y 2 herramienta Check Point. Versiones posteriores mediante copia de seguridad
Cryptolocker	Comprobar en la web http://www.decryptcryptolocker.com/ por si existe solución para su caso
Locky	Comprobar mediante la herramienta Autolocky
Petya	Primera versión: herramienta Bleepingcomputer. Segunda versión: publicada la clave privada.
Wannacry	Solo mediante copia de seguridad.
Crysis	Algunas variantes mediante RakhniDecryptor
NotPetya	Solo mediante copia de seguridad
Cryptowall	Tercera versión mediante herramientas de recuperación de ficheros. Las demás versiones mediante copia de seguridad.
Cryptodefense	Herramienta Emsisoft.

Capítulo 3

Análisis e Identificación de Malware

En este capítulo se aborda el análisis de malware, en qué consiste, su importancia y los tres tipos de análisis que existen: el estático [3.2](#), el dinámico [3.3](#) y el híbrido [3.4](#). El capítulo termina explicando el aprendizaje automático aplicado al análisis de malware [3.5](#) y los algoritmos más usados para el desarrollo de modelos que detecten automáticamente las amenazas.

3.1. Introducción

El análisis de malware es el estudio del comportamiento del malware, y su objetivo es comprender el funcionamiento del malware, sus capacidades, cómo detectarlo, contenerlo y/o eliminarlo [\[ESKK08\]](#). Esto implica analizar el programa sospechoso en un entorno seguro para identificar sus características y así poder construir mejores defensas para defender la red de una empresa. Además, analizar malware sirve para encontrar vulnerabilidades en nuestros sistemas, nuevos métodos de infección y la evolución de los atacantes y sus propósitos [\[A18\]](#). Las razones por las que es importante realizar un análisis de malware son [\[Mon18\]](#):

- Determinar la naturaleza y finalidad del malware. Es importante saber si el malware es un ransomware, un *spyware* [\[KKB⁺06\]](#), un *rootkit* [\[CL07\]](#), un *Remote Access Trojan (RAT)* [\[KA19\]](#), etc.
- Entender cómo se vio comprometido el sistema y el impacto que tuvo el malware sobre él [\[RHW⁺08\]](#).
- Encontrar los indicadores de red del malware mediante la monitorización de la red. Estos indicadores se utilizan para detectar infecciones parecidas. Por ejemplo, si durante el análisis se concluye que un malware contacta con una dirección **IP** en particular, se puede monitorizar el tráfico de la red para identificar todos los equipos que contactan con esa dirección.
- Extraer conclusiones como claves de registro y nombres de archivo basándose en el equipo. También se utilizan para detectar una infección parecida en otro equipo. Por ejemplo, si un malware crea una clave de registro, se puede utilizar esa misma clave como indicador para identificar todos los equipos que tienen la misma clave de registro.

- Determinar el motivo del atacante, su intención y su identidad si es posible para poder clasificar el malware correctamente.

Para comprender el funcionamiento y las características del malware y evaluar su impacto en los sistemas que infecta, se utilizan tres tipos de análisis: estático, dinámico e híbrido.

3.2. Análisis Estático

El análisis estático de una muestra de malware consiste en analizar un binario en busca de signos de intenciones maliciosas sin tener que ejecutarlo [DDTV⁺17a], es decir, poder determinar si se trata de una muestra maliciosa antes de que comience a actuar, incluso es posible encontrar más información acerca del tipo y familia de malware y el comportamiento que tendrá [MS20]. Es más fácil de realizar y permite extraer los metadatos asociados con el programa sospechoso. Este tipo de análisis se puede interpretar como una “disección” del binario sin el riesgo de que el malware actúe en nuestro equipo [Che14].

A la hora de realizar análisis estático de una muestra, se utiliza principalmente un desensamblador para realizar la “disección” anteriormente mencionada y decodificar el código máquina del malware [Ras20].

Puede que el análisis estático no revele toda la información necesaria, pero puede proporcionar información interesante que ayude a determinar el enfoque de los análisis posteriores. Para determinar si un archivo es malicioso, se usan indicadores técnicos denominados *Indicators of compromise (IoC)* como nombres de archivo, direcciones IP, dominios y datos en el encabezado del archivo [Lor20], y herramientas como analizadores de red para observar el malware sin ejecutarlo para recopilar información sobre cómo funciona [ESKK08].

3.2.1. Técnicas

La elección de la técnica para revelar información de un archivo potencialmente malicioso depende del objetivo y del contexto que rodea al archivo sospechoso [Bak20]. Algunas de las técnicas o fases de análisis estático son:

- **Determinar el tipo de fichero y arquitectura:** La extensión del binario ayuda a averiguar cual es la máquina objetivo del malware (Windows, MacOS, etc.) y la arquitectura del mismo. Por ejemplo, si el tipo del archivo es *Portable Executable (PE)*, que es el formato de archivo para los ejecutables de Windows (.exe, .dll, .sys, .drv, .com, .ocx, etc) [Mic20b], entonces se puede deducir por la extensión del archivo que el malware está diseñado para atacar el sistema operativo Windows. Aun así, esto no es del todo fiable ya que los atacantes utilizan diferentes trucos para ocultar el archivo, modificar la extensión real del fichero y cambiar su apariencia con el fin de engañar a los usuarios para que lo ejecuten. Para encontrar la extensión real de un fichero en Windows, es posible fijarse en la firma del mismo (*file signature*). Una firma de archivo es una secuencia única de bytes que se escribe en su encabezado [SS04]. los primeros bits indicarán si se trata de algún tipo de ejecutable (*PE files*). Esto se puede determinar utilizando métodos manuales o herramientas específicas [A18]. También es posible determinar si se está falseando la extensión o se trata de un archivo sospechoso prestando atención a sus propiedades (información o detalles de la versión, copyright de la aplicación, etc.) o con una aplicación específica, como

CFF Explorer, para observar si la miniatura del archivo ha sido modificada para simular ser un fichero legítimo [MS20].

- **Huellas digital del malware:** Esta técnica consiste en la generación del *hash* criptográfico para los binarios basándose en el contenido del fichero [A18]. Los algoritmos más comunes son MD5, SHA1 y SHA256 y a la hora de realizar un análisis es bueno generar los tres tipos de *hashes* para la misma muestra. El motivo de generar el hash para una muestra es que podemos compartir esta información con plataformas de análisis u otros usuarios para identificar el malware sin tener que compartir el mismo, evitando infectar un equipo ajeno o transmitir información personal del usuario [MS20]. La generación de estos valores *hash* se realiza mediante la ejecución los comandos *md5sum*, *sha256sum* y *sha1sum* en Linux o mediante el uso del módulo *hashlib* en Python.
- **Inspección de la información del encabezado PE:** La cabecera de un archivo, además de contener su tipo, incluye información sobre dónde se debe cargar el ejecutable en la memoria, la dirección donde comienza la ejecución, la lista de bibliotecas y funciones en las que se basa la aplicación y los recursos utilizados [Mar20]. Para realizar tales interacciones, el malware depende de las funciones del sistema operativo. Windows exporta estas funciones llamadas **API** en archivos *Dynamic Link Library (DLL)* (archivos pertenecientes a bibliotecas de enlaces dinámicos) [Fre19]. Los ejecutables importan y llaman a estas funciones normalmente desde varios archivos **DLL** que proporcionan diferentes funcionalidades. La inspección de los archivos **DLL** en las que se basa un malware y las funciones **API** que importa pueden dar una idea sobre la funcionalidad y capacidad del malware. Toda esta información de las importaciones se almacena en tablas en el encabezado **PE** [Mon18].
- **Escaneo del virus con un programa específico como un antivirus:** Realizar esta acción ayuda a saber más acerca de la posible muestra maliciosa y mejorar nuestro análisis [A18]. Para ello, se suele utilizar algún antivirus específico que instalado en el equipo o en páginas web como VirusTotal o Jotti Virus Scan, que ofrecen un servicio de escaneo de malware con varias fuentes con el fin de determinar la peligrosidad de una muestra y almacenan toda la información obtenida a la cual se puede acceder y se comparte con empresas u organizaciones interesadas. Además, existen otras plataformas web como “file2scan” o “scan4you” las cuales son de pago y no comparten la información con nadie, siendo el escaneo anónimo, lo que permite a cibercriminales comprobar la peligrosidad de sus archivos maliciosos [Che14] [A18]. El riesgo que existe en esta técnica es que el escaneo no detecte el fin malicioso de la muestra, pero aún así, siga tratándose de un malware.
- **Detectar protección o empaquetadores:** Los creadores de malware en su mayoría buscan ocultar el contenido y funcionamiento del malware para que en el análisis del mismo sea difícil o imposible encontrar esta información. Para realizar esto, se utilizan programas específicos conocidos como empaquetadores o *packers* o *cryptors* [Che14]. Los empaquetadores o packers ocultan el ejecutable en un nuevo archivo que cuando se activa, “descomprime” el binario original para realizar el ataque. El *cryptor* es parecido, pero utiliza cifrado en lugar de compresión [A18]. Para detectar este tipo de empaquetado o cifrado, se pueden utilizar herramientas como Exeinfo PE, PEiD o RDG Packer Detector.

- **Extracción de cadenas, funciones y metadatos asociados con el binario:** Las cadenas son secuencias de caracteres en *American Standard Code for Information Interchange (ASCII)* y *Unicode* contenidas en un archivo [Lá18]. La extracción de estas cadenas puede dar pistas sobre la funcionalidad del programa, ya que contienen referencias a nombres de archivo, varias [URL](#), nombres de dominio, direcciones IP, comandos de ataque, claves de registro, etc. Aunque las cadenas no dan una imagen clara del propósito y la capacidad de un malware, puede ayudar a encontrar información sobre la acción del fichero malicioso o sobre el atacante, como por ejemplo el tipo de malware, direcciones IP, comandos de ataque, claves, objetivos del malware, etc. [A18]. Para extraer estas cadenas, existe un comando en Linux llamado *strings*, pero esto sólo es posible si la información no ha sido ocultada mediante *packers* o *cryptors* o si estos han empleado una protección vulnerable. Para estos casos se puede usar la herramienta *pestudio* para extraer cadenas de un binario sospechoso o a *FireEye Labs Obfuscated String Solver (FLOSS)* para detectar y extraer información oculta [Fir17].
- **Clasificación de las muestras:** Con el fin de conocer características específicas de un malware o su pertenencia, se compara el binario con otras muestras analizadas previamente o almacenadas en un repositorio que puede ser privado o público. El *hash* criptográfico visto anteriormente es una gran técnica para detectar muestras idénticas, pero no ayuda a identificar muestras similares ya que a menudo los delincuentes cambian ciertos parámetros del malware que modifican el *hash* por completo y dificulta esta tarea. Para poder clasificar las muestras de malware se pueden utilizar técnicas como *Fuzzy Hashing* (mediante la herramienta *ssdeep*) [WeL14], *Import Hash* (que mira las funciones API llamadas por los dos archivos) [Man14] o YARA (una herramienta para crear reglas que consisten en expresiones booleanas y cadenas basadas en patrones de información del binario para poder identificarlo cuya complejidad puede ser muy alta) [Vir20] [Li20].

En la Figura 3.1 se muestra el diagrama de flujo del proceso de un análisis estático, mencionando todas las técnicas explicadas anteriormente.

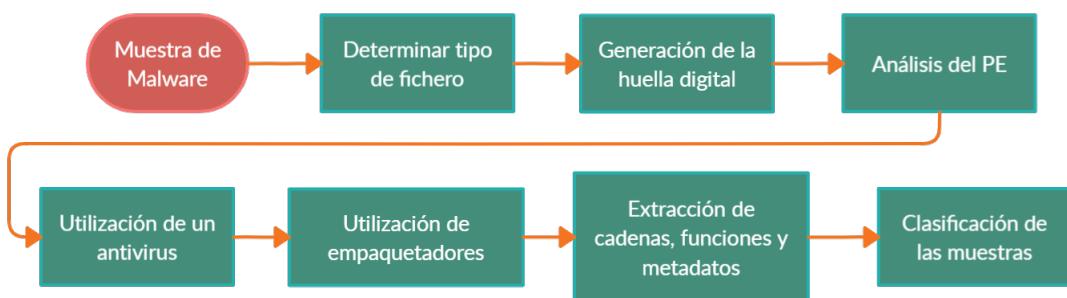


Figura 3.1: Diagrama de flujo de un análisis estático.

El problema del análisis estático es que, al no ejecutar el código, un malware más sofisticado puede incluir comportamientos maliciosos en tiempo de ejecución que pueden pasar desapercibidos. Por ejemplo, si descarga un archivo malicioso basado en una cadena dinámica, el análisis estático no lo detectará [MKK07]. Es por esto por lo que las empresas recurren al análisis dinámico para una comprensión más completa del comportamiento del malware.

3.2.2. Herramientas

Existen diferentes herramientas de análisis estático según la técnica que apliquen, algunas de las más conocidas y usadas se muestran en la Tabla 3.1.

Tabla 3.1: Herramientas de análisis estático

Herramienta	Descripción
WinMD5Free	Una herramienta gratuita con interfaz gráfica para calcular el <i>hash</i> de un fichero utilizando el algoritmo <i>Message-Digest Algorithm (MD5)</i> para comprobar comparando el <i>hash</i> generado si el fichero ha sido alterado en algún momento [Che14].
CFF Explorer	Un conjunto de herramientas gratuitas destinado a la monitorización de procesos y la edición de ficheros PE que permite la modificación de campos especiales, incluye desensamblador y gestión de firmas entre otras muchas funcionalidades [Che14].
Ollydbg	Un simple depurador sólo para sistemas Windows de 32 bits enfocado principalmente al análisis de código binario. Las diferentes características de este programa permiten usar <i>breakpoints</i> y cambiar instrucciones y condiciones en el ejecutable, además, es libre y no necesita instalación. Destaca también por tener una gran variedad de <i>plugins</i> [VDL17].
Ghidra	Herramienta desarrollada en Java analiza tanto ficheros PE como <i>Executable and Linkable Format (ELF)</i> y puede utilizarse en Windows, Linux y MacOS. Se trata de un proyecto desarrollado por la <i>National Security Agency (NSA)</i> de Estados Unidos que, además de desensamblar binarios, contiene un descompilador para sacar un pseudocódigo en lenguaje C del fichero analizado [Ras20].
IDA Pro	Es un proyecto de Hex-Rays utilizado para ingeniería inversa del cual se puede obtener una versión básica con pocas funcionalidades gratuitamente. Básicamente, <i>Interactive Disassembler Professional (IDA)</i> Pro es un sofisticado desensamblador con múltiples funcionalidades en su versión completa que busca hacer el código complejo más legible para el usuario [Eag11].
Radare2	Un proyecto libre el cual se puede obtener de un repositorio en GitHub y está basado en un conjunto de comandos que se pueden usar juntos o aislados. Cuenta con diferentes programas como rabin2, rasm2, rahash2... siendo el principal de todos radare2, que permite “diseccionar” un binario, analizar los datos, desensamblar el código, etc. Además, admite diferentes lenguajes, entre ellos Python, JavaScript o Perl [rad21].
PEiD	Una de las herramientas más usada en todo el mundo para identificar <i>packers</i> ya que es la que mejores resultados ha proporcionado. Una de las claves de su éxito es que contiene un alto número de firmas de empaquetadores para identificar a estos. Además, es gratuita [KDM15].
PEStudio	Una sencilla aplicación libre usada para estudiar cualquier tipo de ejecutable extrayendo cadenas y firmas. Los resultados son similares a los que ofrece VirusTotal, lo que permite complementar la herramienta [AKRR20].

3.2.3. Limitaciones

Este tipo de análisis muchas veces no sirve para encontrar una resolución clara y es necesario cambiar de técnica si se quiere llegar a algún tipo de resultado. En definitiva el análisis estático sirve para encontrar información sobre la muestra y que esta ayude en una siguiente fase [MS20], además, este análisis puede ser muy complicado debido a que el código ensamblador sea ilegible, ya que algunas funciones las codifican y para poder leerlo sea necesario ejecutar el binario, por lo que ya se hablaría de análisis dinámico [Ras20].

3.3. Análisis Dinámico

El análisis dinámico de una muestra se realiza mientras que esta está siendo ejecutada y es más importante que nunca hacerlo en un entorno controlado montando un laboratorio de análisis [DDTV⁺17a]. Esta técnica de ejecutar el programa malicioso en un entorno seguro se conoce como *sandboxing* [Tec16], o aislamiento de procesos en español. En este tipo de análisis se presta atención a las llamadas a API, el tráfico de red o la actividad

en procesos, en archivos o en el registro del sistema [GBS14]. Para realizar el análisis dinámico se pueden utilizar tanto herramientas instaladas en un equipo como plataformas web que ahorran el trabajo de montar un laboratorio de análisis seguro. Normalmente, este análisis se realiza cuando el análisis estático ha llegado a un final ya sea porque es imposible analizarlo o porque los resultados no son satisfactorios.

El malware interactúa con el sistema de varias formas cuando es ejecutado, creando claves de registro y valores para su persistencia, descargando archivos, creando nuevos o comunicándose con un servidor C&C [Mic]. La monitorización de estas interacciones con el sistema y la red ayudará a comprender el propósito y la naturaleza del malware.

3.3.1. Técnicas de Análisis

Al hablar de las técnicas de análisis dinámico, es preciso centrarse principalmente en la monitorización, sin dejar de prestar a atención al resto. Existen diferentes técnicas utilizadas en el análisis dinámico de malware; sin embargo, para conseguir los mejores resultados, será necesario combinarlas:

- **Sandboxing:** *Sandboxing* es una técnica que consiste en aislar la ejecución de un programa no fiable o malicioso llevándola a un entorno seguro que no comprometa el sistema, de forma que sea posible analizar las actividades del malware sin preocupación por los cambios que realice el proceso maligno. Existen múltiples herramientas que utilicen *sandboxing* y permitan construir un laboratorio de análisis de malware, entre ellas, Buster Sandbox Analyzer, Zero Wine, Malheur o Cuckoo Sandbox. Normalmente estas herramientas simulan servicios de red para que el malware actúe con normalidad. Utilizar *sandboxing* también tiene inconvenientes, por ejemplo, si el malware necesita un comando de activación o este actúa pasado un tiempo superior a la duración del análisis este no llegará a activarse y por lo tanto el análisis será fallido. Además, muchos malwares pueden detectar que se encuentran en una máquina virtual y debido a ello interrumpen su ejecución [SH12].
- **Creación de instantáneas:** Cuando para el análisis se cuente con un laboratorio seguro, las muestras serán ejecutadas en máquinas virtuales con las herramientas de análisis que se vayan a utilizar instaladas en ellas. Al ejecutar una muestra, esta puede realizar cambios en el sistema de la máquina virtual, dejar trazas o mantener el sistema infectado después de haber analizado el malware. En ese caso, si se reutiliza el entorno para analizar una nueva muestra, las acciones de análisis previos podrían interferir en el nuevo análisis. Para solucionar esto, se guarda el estado de la máquina virtual puesta a punto para su primer análisis mediante instantáneas (*snapshots*) de forma que, tras analizar una muestra y obtener los resultados, se retroceda al estado inicial de la máquina para garantizar que ninguna ejecución de malware anterior interfiere en el análisis [MS20].
- **Monitorización:** Un malware puede interactuar tanto con el sistema como con la red, descargando algún componente malicioso, modificando el sistema de ficheros (crear nuevos archivos, borrar otros, etc.) hacerse con el control del equipo o comunicándose con un servidor C&C entre otras cosas [Mic]. Para estudiar el comportamiento del malware, se recurre a la monitorización de procesos, del sistema de ficheros, de registros y de red [A18]. Existen tipos de malware, como es el caso del ransomware, cuyo comportamiento es muy explícito y se puede notar a simple vista y otros que, si no se recurre a la monitorización, pueden pasar fácilmente por

alto [MS20]. Los diferentes tipos de monitorización que se podrán realizar durante el análisis dinámico son [Mon18]:

- **Monitorización de procesos, del registro y del sistema de archivos:** Implica monitorizar las actividades de los procesos y examinar sus propiedades, ver si interactúan con el sistema de archivos y si acceden o modifican claves de registro durante la ejecución del malware. Process Hacker [Sou] es una herramienta de código abierto para examinar los procesos que se ejecutan en el sistema e inspeccionar sus atributos. También se puede utilizar para explorar servicios, conexiones de red, actividad del disco, etc. Una vez que se ejecuta la muestra de malware, esta herramienta puede ayudar identificar dicha muestra y examinar sus atributos o detener la ejecución. Process Monitor [Rus21] es otra herramienta enfocada en la supervisión que muestra la interacción en tiempo real de los procesos con el sistema de archivos, el registro y la actividad de procesos y subprocesos. Para solo centrarse en las actividades resultantes de la actividad del malware, hay opciones de filtrado que permiten filtrar por atributos específicos. Pero esto puede resultar muy tedioso, por lo que existe un programa en Python llamado Noriben [Sol18] que viene con filtros predefinidos.
- **Monitorización de la red:** Es el estudio del tráfico de la red durante la ejecución de malware en el sistema, necesario para encontrar el canal de comunicación utilizado dicho malware y para determinar los indicadores basados en la red. Como ya sabemos, la mayoría del malware al ejecutarse se comunica con un servidor C&C. Para realizar un análisis completo, hay que emular este servidor para proporcionar al malware todos los servicios requeridos para que pueda continuar con su funcionamiento. INetSim [HE20] es un software gratuito basado en Linux para simular servicios de Internet, como un servidor ficticio, que es justo lo que necesitamos. INetSim puede registrar comunicaciones y también puede responder a solicitudes *Hypertext Transfer Protocol* (HTTP) o *Hypertext Transfer Protocol Secure* (HTTPS) y devolver archivos. Por ejemplo, si el malware solicita un archivo ejecutable, INetSim se lo puede proporcionar. De esa manera se podrá saber qué hace el malware con el archivo después de descargarlo del servidor C&C.
- **Simulación de una red:** Muchos malwares establecen conexión con un servidor externo en la red y si el entorno de análisis no permite conexión a Internet, estas muestras nunca mostrarán su comportamiento al completo. Para solucionar esto, se simula una red y así se obtiene información como *Domain Name Server* (DNS) o direcciones IP. En muchas ocasiones, si se utiliza una herramienta de *sandboxing*, se tiene la opción de simular una red [SH12].
- **Análisis de bibliotecas de vínculos dinámicos (DLL):** Una DLL es un módulo con código y datos que utilizan los distintos programas para realizar cierta función de la DLL, ayudando a promover la reutilización de código y el uso eficaz de la memoria [Lia20]. Las funciones exportadas por los archivos DLL se llaman API (interfaces de programación de aplicaciones en español) [Mon18]. Los sistemas operativos proporcionan a las aplicaciones las API para que puedan realizar tareas comunes, como interactuar con el sistema de archivos, el registro del sistema, la red y la interfaz gráfica de usuario [ESKK08]. Los últimos sistemas operativos de Windows proporcionan tanto una API nativa (llamadas al sistema) como una API de Windows [AHSF09a]. Muchas veces los autores de malware implementan el código

como DLLs ya que esto permite al atacante cargar la DLL en cualquier proceso, proporcionando la persistencia del malware en el sistema y ocultando la actividad maliciosa [A18]. Analizar la interacción que realiza el malware mediante DLLs es un factor muy importante en el análisis dinámico.

En la Figura 3.2 se muestra el diagrama de flujo del proceso de un análisis dinámico, en el que se puede ver todo lo explicado anteriormente.

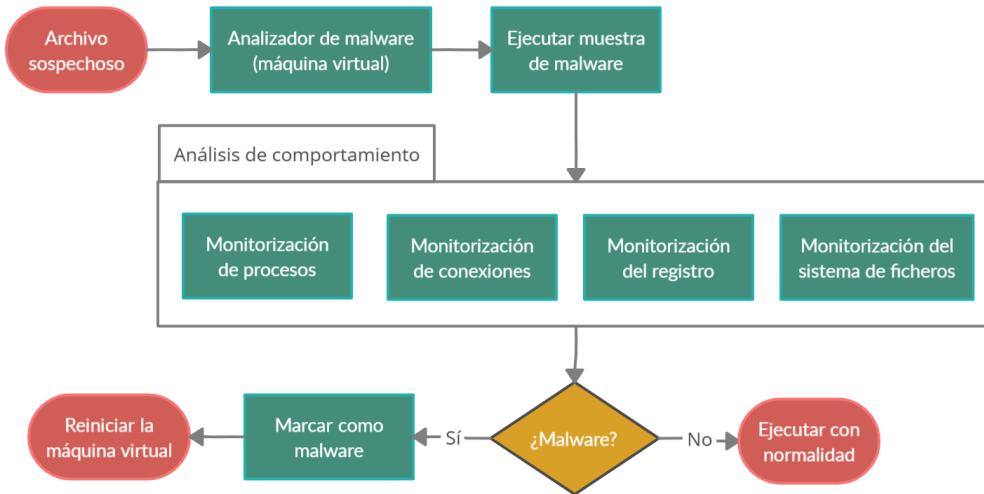


Figura 3.2: Diagrama de flujo de un análisis dinámico.

3.3.2. Herramientas

Cuando se habla de las herramientas de análisis dinámico, se puede diferenciar entre análisis dinámico automatizado y análisis dinámico manual. Por lo general, los programas o plataformas web basadas en *sandboxing* pertenecen al análisis automatizado e incluyen monitorización, en cambio, las herramientas de análisis manual implican controlar la información que se quiere obtener de ella y analizarla después. En la Tabla 3.2 se muestran las herramientas más utilizadas en el análisis dinámico de malware que no hayan sido explicadas anteriormente.

3.3.3. Limitaciones

El análisis dinámico es más efectivo que el análisis estático descrito anteriormente y no requiere desensamblar el ejecutable. El punto negativo del análisis dinámico es que el consumo de recursos es mayor y lleva más tiempo, aumentando los problemas de escalabilidad. Por otro lado, a veces el malware está destinado a un cierto sistema o condiciones del mismo (tener un motor de búsqueda concreto, determinadas DLLs, etc.), o quizás tenga protección contra máquinas virtuales, lo que podría provocar que no se pueda detectar su comportamiento malicioso en el laboratorio de análisis seguro que se haya montado [GBS14]. En definitiva, lo más efectivo viene a ser combinar tanto el análisis estático como el análisis dinámico para conseguir el mejor resultado posible. La Tabla 3.3 se muestra de manera clara las diferencias fundamentales entre el análisis el estático y el dinámico, indicando los métodos de análisis que usan cada uno [UAB19b].

Tabla 3.2: Herramientas de análisis dinámico.

Herramienta	Descripción
Cuckoo Sandbox	Herramienta de análisis automatizado de malware de código abierto en Windows, macOS, Linux y Android que utiliza la tecnología de sandboxing. Cuckoo Sandbox tiene un diseño modular, lo que permite personalizar cualquier aspecto del procesamiento del análisis de los resultados, de la generación de informes y del entorno de análisis.
Regshot	Programa que realiza dos instantáneas (una antes de ejecutar el malware y otra después) y las compara encontrando los valores modificados durante la actuación del código malicioso [SH12].
Wireshark	Herramienta más usada en todo el mundo para analizar el tráfico de red con gran detalle. Soporta gran variedad de plataformas y protocolos y dispone de múltiples filtros, formatos y formas de exportar la información obtenida [RSL21] [Wir].
ApateDNS	Herramienta gratuita que se puede utilizar para encontrar peticiones DNS realizadas por un malware. Esto lo hace suplantando respuestas DNS a una dirección IP específica escuchando en el puerto 53 <i>User Datagram Protocol (UDP)</i> en la máquina local [SH12].
APIMiner	Herramienta que registra las llamadas a API realizadas durante la actuación de una muestra. El resultado que ofrece es un registro de las llamadas a API junto a sus argumentos ordenadas según el momento en el que fueron invocadas [MS20].

Tabla 3.3: Diferencias entre análisis dinámico y estático.

Métodos de análisis	Estático	Dinámico
Extracción de cadenas	Sí	No
Secuencias de bytes	Sí	No
Códigos de operación	Sí	No
Llamadas a las API	Sí	Sí
Sistema de ficheros	No	Sí
Registros de <i>Central Processing Unit (CPU)</i>	Sí	Sí
Características del PE	Sí	No
Monitorización de la red	No	Sí
Acceso a memoria	No	Sí
Sandboxing	No	Sí
Excepciones	No	Sí

3.4. Análisis Híbrido

El análisis híbrido es una combinación de las técnicas de análisis estáticas y dinámicas, proporcionando al sistema una mayor seguridad usando ambos enfoques [DDTV⁺17a]. Este análisis puede detectar malware que está tratando de ocultarse y extraer los indicadores técnicos mediante el análisis estático. Aplica el análisis estático a datos generados por el análisis dinámico en tiempo de ejecución, por lo que, si el malware es cambiante, en análisis dinámico detectaría esto y enviaría los nuevos datos al análisis estático, repitiéndose el ciclo. De esta manera, se generarán indicadores IoC de manera más rápida y eficiente [Bak20]. Este enfoque a primera vista parece ser el mejor, ya que combina el análisis estático con el dinámico, pero este no es siempre el caso. Un estudio [DDTV⁺17b] revela que es poco probable que un enfoque híbrido sea superior al análisis completamente dinámico o estático. Bien es cierto que el análisis híbrido ofrece beneficios en algunos casos, pero en ese trabajo se demuestra que tales afirmaciones deberían estar sujetas a un escrutinio cuidadoso. Además, se debe determinar si estos beneficios existen para el análisis de una amplia gama de muestras de malware o si solo son relevantes para un conjunto reducido de muestras.

3.5. Aprendizaje Automático Aplicado al Análisis de Malware

El aprendizaje automático, en inglés [ML](#), es una rama de la inteligencia artificial que permite crear sistemas con la capacidad de identificar patrones de comportamiento y de hacer predicciones de forma autónoma [[Com19](#)]. Esto es posible gracias a algoritmos que analizan una gran cantidad de datos y reconocen patrones para realizar predicciones con nuevos datos sin estar programados para producir un resultado en particular. El objetivo final es que el modelo construido aprenda solo de forma que reduzca la implicación del usuario y este realice su tarea de forma independiente [[Kar18](#)]. El proceso de construir un modelo sigue ciertos pasos [[TVE20](#)]:

- **Recolección de datos:** En esta fase se recogen todos los datos que vayan a ser usados para formar el *dataset* del modelo y se dividen en datos de entrenamiento y datos de prueba. Los primeros serán los utilizados para construir el modelo y los últimos para evaluar el mismo.
- **Preparación de los datos:** Es importante procesar los datos antes de introducirlos al modelo, y para ello, será necesario aplicar una serie de operaciones de transformación y limpieza sobre ellos. Se emplean operaciones como la normalización, la estandarización o la aplicación de expresiones no lineales.
- **Selección del modelo:** Dependiendo del tipo de datos se vayan a utilizar y la finalidad del modelo, se elegirá un modelo de [ML](#) u otro, ya que existen diversos tipos y cada uno es más adecuado a un tipo de datos o forma de trabajo. Por ejemplo, los modelos de regresión están más destinados a tratar variables continuas.
- **Extracción de características:** El *dataset* utilizado puede tener ciertas variables o características que no sean importantes o determinantes para el modelo, en ese caso, se seleccionaran las necesarias para implementar el modelo de forma que no haya variables redundantes que perjudiquen la precisión del modelo.
- **Entrenamiento del modelo:** Antes de probar e implementar el modelo, será necesario entrenarlo con los datos previamente seleccionados para esta tarea.
- **Prueba del modelo:** En esta fase, se prueba el modelo entrenado con los datos seleccionados para ello.
- **Implementación del modelo:** Por último, se implementa el modelo para su funcionamiento final en tiempo real.

En definitiva, el [ML](#) se utiliza para que los ordenadores, entrenados, tomen sus propias decisiones basándose en un algoritmo, imitando el comportamiento humano. Los algoritmos de [ML](#) se clasifican en cuatro categorías, que son las siguientes [[TVE20](#)]:

- **Aprendizaje supervisado:** Cuando se habla de aprendizaje supervisado, la persona que construye el modelo conoce tanto las variables de entrada como las de salida [[Che18](#)]. La variable de salida es también conocida como la señal de supervisión. Algunos algoritmos usados para este tipo de aprendizaje son los árboles de decisión, el algoritmo *Näive Bayes (NB)* o las máquinas de soporte vectorial (*Support Vector Machine (SVM)*) [[TVE20](#)]. En la Figura 3.3 se puede ver un gráfico que muestra el comportamiento descrito.

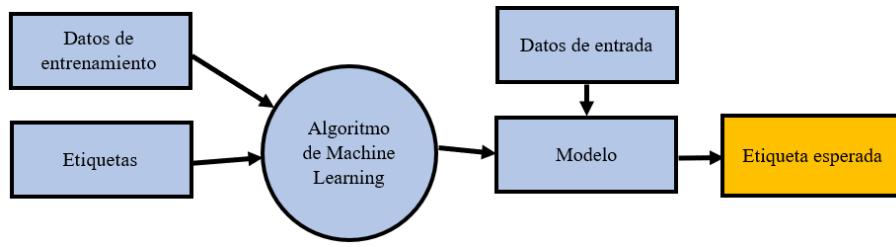


Figura 3.3: Modelo de aprendizaje supervisado

- **Aprendizaje no supervisado:** En este tipo de aprendizaje, no se tiene claro cuál será la salida de los modelos [Che18] y únicamente se conocen los datos de entrada. De esta forma, estos algoritmos intentan encontrar en los datos patrones y clases que puedan ser útiles. Algoritmos comúnmente usados en aprendizaje no supervisado son las redes neuronales o diferentes tipos de agrupamientos (*clustering*) como el algoritmo k-means, normalmente indicado para datos numéricos [TVE20]. En la Figura 3.4 se puede ver un gráfico que muestra el comportamiento descrito.

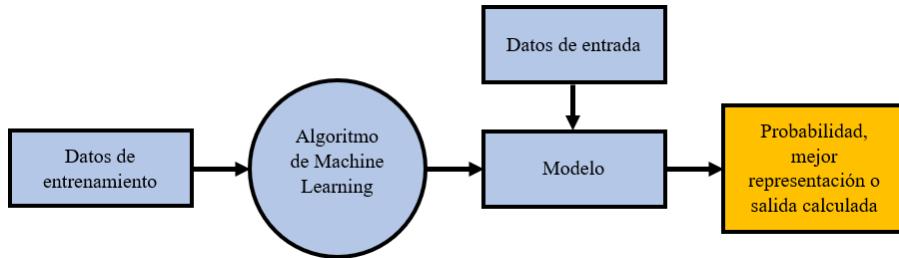


Figura 3.4: Modelo de aprendizaje no supervisado

- **Aprendizaje por refuerzo:** El modelo aprende a base de prueba y error. El modelo (o agente) interactúa con el entorno (contiene las reglas y limitaciones) y va mejorando. El agente, según el estado del entorno, realiza una acción y el entorno responde, entonces, el agente utiliza la información recibida como recompensa.^o “castigo” para orientar su comportamiento de forma que si la respuesta del entorno es positiva (recompensa), el agente reforzará ese comportamiento y si, por el contrario, esta respuesta es negativa (castigo), el agente actuará de forma distinta llegado de nuevo el mismo estado u otro similar [Che18] [TVE20]. En la Figura 3.5 se puede ver un gráfico que muestra el comportamiento descrito.

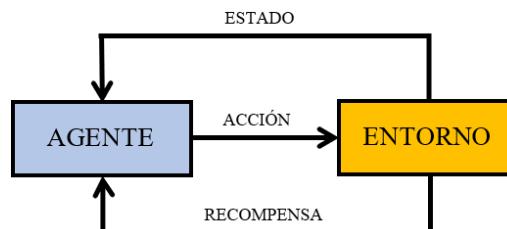


Figura 3.5: Modelo de aprendizaje por refuerzo

- **Aprendizaje semi-supervisado:** El aprendizaje semi-supervisado se aplica cuando una parte de los datos están etiquetados (se conoce la entrada y la salida) y la otra no. Esto puede ocurrir cuando generar datos es un proceso muy costoso [Che18]. En estos casos, el problema se encuentra entre el aprendizaje supervisado y el no supervisado y se aplican algoritmos como modelos generativos (*generative models*) o métodos basados en gráficos (*graph-based methods*) o máquinas de soporte vectorial semi-supervisadas [TVE20].

La utilización de estos tipos depende de las necesidades y del problema que se quiera resolver; en el caso del análisis de malware, el aprendizaje supervisado es el más usado [AVWA11].

3.5.1. Aprendizaje Supervisado

La característica que define al aprendizaje supervisado es la disponibilidad de datos de entrenamiento etiquetados. El nombre indica una idea de un “supervisor” que instruye al sistema con etiquetas para que pueda asociarlas con los datos de entrada y que posteriormente sea capaz de clasificar otros datos no etiquetados [CC08].

El aprendizaje supervisado se puede dividir en dos tipos: clasificación y regresión. Los modelos de regresión se basan en el análisis de relaciones entre tendencias y variables para poder realizar predicciones sobre variables continuas, y los de clasificación asignan etiquetas discretas a observaciones particulares [Ras14]. A continuación, se describirán de forma general los algoritmos de clasificación de aprendizaje supervisado más utilizados en análisis de malware:

- **Regresión logística (*Logistic Regression (LR)*):** Es comúnmente empleado para los sistemas de puntuación en aplicaciones y para el modelado de riesgo crediticio [SS18]. En primer lugar, se realiza una regresión lineal sobre la relación entre variables para obtener el modelo. Después, la función logística se aplicará a la regresión para obtener la probabilidad de que la muestra pertenezca a cada una de las clases y se clasificará en función de la probabilidad más alta. La función logística, también llamada función *sigmoidea*, es una curva en forma de “S” que toma cualquier número lo asigna a un valor entre 0 y 1, pero nunca esos valores exactamente [Bro16]. Los trabajos que usan este algoritmo para sus análisis son: [SMGML16], [MMB18a] y [Jet19].

Para realizar las predicciones, hay que tener en cuenta que la probabilidad debe de transformarse en valores binarios (0 o 1), y para ello se introduce en la ecuación de regresión logística junto con los coeficientes (valores beta) del algoritmo, estimados mediante el método de máxima verosimilitud [BSVB14]. Los mejores coeficientes darán como resultado un modelo que prediga un valor muy cercano a 1 para una clase y 0 para la otra.

Para obtener la función de regresión logística se usa la Fórmula 3.1 [Bro16]

$$y = \frac{e^{(\beta_0 + \beta_1 x)}}{1 + e^{(\beta_0 + \beta_1 x)}} \quad (3.1)$$

donde, y el valor de salida previsto, β_0 el coeficiente de sesgo o intersección y β_1 el coeficiente para cada valor de entrada x

La Figura 3.6 representa gráficamente todo lo explicado anteriormente.

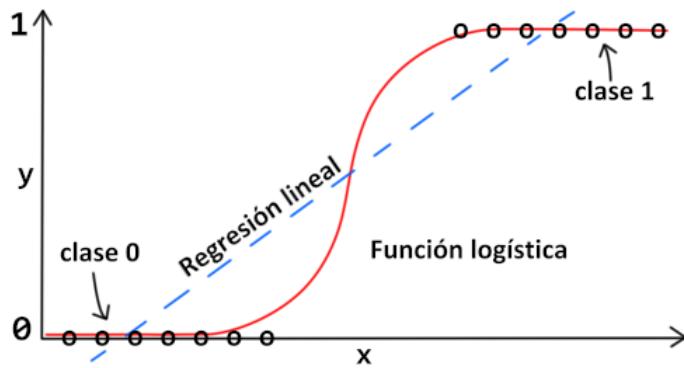


Figura 3.6: Gráfico de la regresión logística.

- **Bayesiano ingenuo (NB):** Es un clasificador basado en el teorema de Bayes. Almacena la probabilidad previa de cada clase y la probabilidad condicional de cada valor de los atributos. Estima estas cantidades contando la frecuencia de ocurrencia de las clases y de los atributos en los datos de entrenamiento. Después, asumiendo la independencia condicional de los atributos, usa la regla de Bayes para calcular la probabilidad posterior de cada clase dados nuevos datos sin etiquetar [KM06]. Los trabajos que usan este algoritmo son: [SMGML16], [CKYK17], [VSVG17], [MMB18a], [BLI19] y [KAJS19].

Para calcular la probabilidad posterior de la clase se usa la Fórmula 3.2 [SS18]

$$P(\text{clase}|\text{datos}) = \frac{P(\text{datos}|\text{clase}) * P(\text{clase})}{P(\text{datos})} \quad (3.2)$$

donde, $P(\text{clase})$ es la probabilidad previa de una etiqueta, $P(\text{datos}|\text{clase})$ es la probabilidad previa de que un conjunto de características dado se clasifique como etiqueta y $P(\text{datos})$ es la probabilidad previa de que apareciera un conjunto de características.

- **K-vecinos más cercanos (K-Nearest Neighbor (KNN)):** Este algoritmo de aprendizaje perezoso clasifica los datos en función de su proximidad con otros mediante funciones de distancia [Pet09]. Ser un algoritmo perezoso significa que no se necesita generalizar los datos de entrenamiento para la generación del modelo [Aha13]. En KNN, la "K" es el número de vecinos más cercanos y es conveniente que sea impar si hay un número par de clases. Para seleccionar la "K" adecuada, se ejecuta el algoritmo KNN varias veces con diferentes valores de "K" y se elige el valor que reduzca la cantidad de errores y mantenga la capacidad del algoritmo para hacer predicciones con precisión con datos nuevos. La principal desventaja ese algoritmo es que se vuelve más lento a medida que aumenta el número de datos [Har18]. Los trabajos que usan este algoritmo para construir sus modelos de ML son: [VSVG17] y [BLI19].

La distancia entre los k vecinos se puede calcular con la Fórmula de distancia euclidiana 3.3 [Pan18]

$$d(p, q) = \sqrt{\sum_{i=1}^n (q_i - p_i)^2} \quad (3.3)$$

donde, p y q son la posición de dos vecinos cualquiera, d la distancia que queremos calcular y n el número de dimensiones o características.

En la Figura 3.7 se puede ver una representación gráfica del algoritmo.

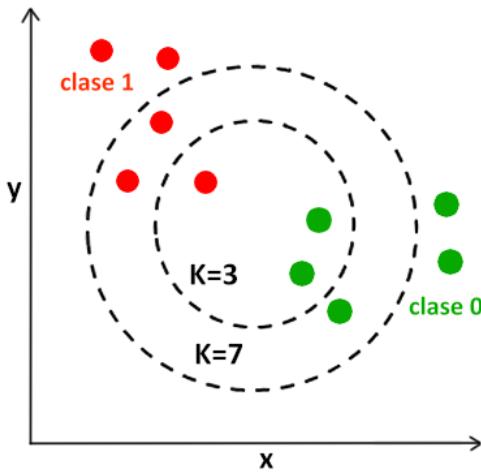


Figura 3.7: Representación gráfica del algoritmo KNN.

- **Árboles de decisión (Decision Tree (DT)):** Proporciona una partición jerárquica de los datos de entrenamiento, donde las 'hojas' o nodos del árbol representan etiquetas y las ramas representan las características o atributos de los datos [Qui86]. El conjunto de datos es dividido en subconjuntos cada vez más pequeños, y esto se repite de manera recursiva hasta que ya no haya más datos por analizar o se cumplan ciertas condiciones, como una profundidad límite. Sigue la estructura del algoritmo *Iterative Dichotomiser 3 (ID3)* para determinar la división de los datos [Sak20]. Los trabajos que usan este algoritmo para sus análisis son: [VSVG17], [MMB18a] y [BLI19].

El algoritmo *ID3* selecciona la mejor característica mientras construye el árbol de decisión. Para esta elección, se usa una métrica llamada mérito o *information gain*, que calcula la reducción de la entropía [Bro19]. La característica con el mayor mérito será la mejor. El mérito de una característica se calcula con la Fórmula 3.4

$$IG(S, a) = E(S) - E(S|a) \quad (3.4)$$

donde, $IG(S, a)$ es el mérito para el conjunto de datos S para la característica de la columna a , $E(S)$ es la entropía del conjunto de datos antes de cualquier cambio y $E(S|a)$ es la entropía condicionada por a .

Para obtener la entropía se usa la Fórmula 3.5 [Sak20]

$$E(S) = \sum_{i=1}^n -p_i \log_2(p_i) \quad (3.5)$$

donde, E es la entropía que queremos calcular del es el conjunto de datos S , n el número total de clases en la columna de destino y p_i la probabilidad de la clase i o la relación entre el número de filas con clase i en la columna de destino y el número total de filas del conjunto de datos.

En la Figura 3.8 se puede ver un ejemplo de un árbol de decisión sobre préstamos.



Figura 3.8: Representación gráfica de un árbol de decisión.

- **Bosque aleatorio (RF)**: Como sugiere el nombre, un bosque aleatorio es la fusión de valores aleatorios provenientes de un conjunto de árboles de decisión [Bre01]. Estos árboles de decisión están generalmente entrenados con el método de “*bagging*”, que consiste en la combinación de modelos de aprendizaje para aumentar la precisión del resultado final [CCS12]. Una ventaja de este algoritmo es que se puede utilizar para problemas de clasificación y también para problemas de regresión. Los trabajos que hacen uso de este algoritmo son: [CKYK17], [ABD18], [CMK18], [MMB18a], [BLI19] y [KAJ20]. Para elegir los valores aleatorios, se buscará la mejor característica en un conjunto aleatorio de características. Se puede incrementar la aleatoriedad aún más mediante el uso de umbrales para cada característica en lugar de buscar las mejores [Don19].

La Figura 3.9 es una representación gráfica del algoritmo con dos árboles de decisión.

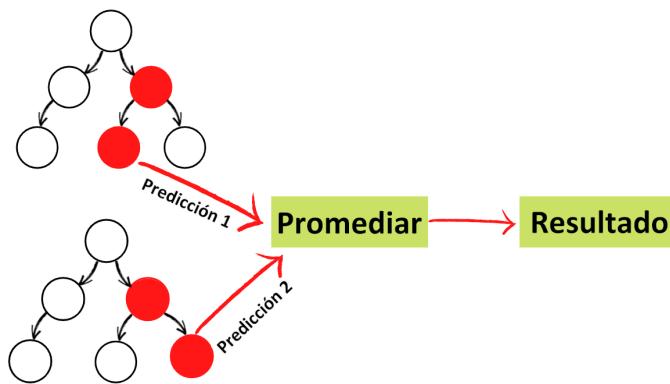


Figura 3.9: Representación gráfica del algoritmo Bosque Aleatorio.

Para promediar los resultados, se utiliza la Fórmula 3.6 [Ron18]

$$RFfi_i = \frac{\sum_{j \in n} normfi_{ij}}{T} \quad (3.6)$$

donde, T es el número de árboles de decisión, $RFfi_i$ la importancia de la característica i calculada a partir de todos los árboles de decisión n y $normfi_{ij}$ la importancia de la característica i normalizada en el árbol de decisión j .

- **Máquina de soporte vectorial (SVM):** Una máquina de soporte vectorial clasifica los datos mediante la construcción de un plano de decisión (hiperplano) de N dimensiones que separa los datos en dos categorías, dependiendo de sus características [SS18]. Los modelos SVM están relacionados con las redes neuronales, ya que un modelo que utilice una función sigmoide es equivalente a una red neuronal de perceptrón de dos capas [Ayo10]. Un conjunto de características junto con sus etiquetas, que representan las categorías, se llama vector. El objetivo del SVM es encontrar el hiperplano óptimo que separe a los vectores según sus categorías. Los trabajos que usan este algoritmo para sus análisis son: [SMGML16], [CKYK17], [VSVG17], [BLI19] y [Jet19].

Los vectores cercanos al hiperplano se llaman vectores de apoyo, y la distancia entre estos vectores de apoyo se llama margen o *maximal margin* [SHG20]. Es difícil separar datos perfectamente en dos categorías, por lo que se puede permitir cierta flexibilidad en la clasificación con los denominados márgenes blandos o *soft-margins*, introduciendo un nuevo parámetro c en la máquina.

SVM utiliza un conjunto de funciones matemáticas que se definen como el núcleo o kernel, el cual procesa los datos de entrada y devuelve el producto interno entre dos puntos en un espacio de características de cualquier dimensión [Dat17]. Si los datos se pueden separar con un hiperplano lineal, usamos SVM Lineal con kernel lineal. Si ese no es el caso, debemos recurrir a SVM No Lineal y usar kernels para transformar espacios no lineales en espacios lineales [Bur98].

Los diferentes algoritmos de SVM utilizan diferentes tipos de kernel. Los más comúnmente usados son los siguientes con sus respectivas ecuaciones:

- **Lineal:** El más simple de todos, calculado con la Fórmula 3.7 [Sou10]

$$k(x_i, x_j) = (x_i x_j + c) \quad (3.7)$$

donde, (x_i, x_j) son los dos puntos en el espacio de características y c es el parámetro de flexibilidad.

- **Polinómico:** Usado en el procesamiento de imágenes y calculado con la Fórmula 3.8 [Yad18]

$$k(x_i, x_j) = (\alpha x_i x_j + c)^d \quad (3.8)$$

donde, (x_i, x_j) son los dos puntos en el espacio de características, d es el grado del polinomio, α la pendiente y c un parámetro que intercambia la influencia de los términos de orden superior frente a los de orden inferior en el polinomio. Si $c = 0$, el kernel se denomina homogéneo.

- **Función de base radial Gaussiana:** Utilizado cuando no hay conocimiento previo sobre los datos y calculado con la Fórmula 3.9 [Dat17]

$$k(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2) \quad (3.9)$$

donde, (x_i, x_j) son los dos puntos en el espacio de características y γ es la función *Radial Basis Function (RBF)*.

- **Sigmoide:** Usada para las redes neuronales y calculada con la Fórmula 3.10 [Sou10]

$$k(x_i, x_j) = \tanh(\alpha x_i x_j + c) \quad (3.10)$$

donde, (x_i, x_j) son los dos puntos en el espacio de características y α es la pendiente.

La Figura 3.10 muestra una representación gráfica de un ejemplo de SVM lineal de dos dimensiones:

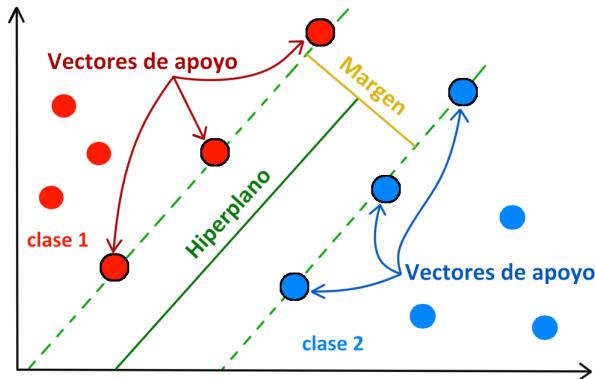


Figura 3.10: Representación gráfica de SVM lineal de dos dimensiones.

A continuación, se listan las librerías de Python más utilizadas para desarrollar un modelo de ML:

- **Numpy:** Se trata de una de las librerías numéricas más utilizadas para llevar a cabo tanto operaciones matemáticas y lógicas, como ordenado, selección, transformaciones y operaciones estadísticas sobre *arrays*. Proporciona cantidad de funcionalidades de álgebra lineal, muy útiles en ML. Se trata de una librería cuyas operaciones se realizan con gran rapidez, debido a la vectorización: no existe ninguna clase de bucle, indexado, etc. en el código, sino que estas partes más costosas se encuentran por debajo en código C pre-compilado [com20].
- **TensorFlow:** Es una interfaz para expresar algoritmos de ML, y una implementación para ejecutarlos. Es un sistema flexible con capacidad para la utilización de una gran cantidad de algoritmos, así como para sistemas heterogéneos y diferentes dispositivos, incluidos móviles, desarrollado por Google [AAB⁺15].
- **Keras:** Esta librería es muy utilizada para implementar modelos de *deep learning*. Está construida encima de TensorFlow, lo que hace que sea sencilla de utilizar [Che18].
- **Pandas:** Pandas es un herramienta para explorar, limpiar y procesar datos tabulares, como bases de datos, convirtiéndolos en un tipo de datos concreto, mucho más fácil de manejar, denominado *DataFrame*. Es ampliamente utilizada en modelos de ML para obtener una entrada de datos eficiente y eficaz [MtPDT21].
- **Matplotlib:** Es una librería utilizada en el campo de la ciencia de datos para visualizar los datos gráficamente. Este aspecto es muy importante para obtener conocimiento de los datos y para razonar los resultados que ofrece el modelo de ML [Che18].
- **Scikit-learn:** Se trata de una librería que proporciona herramientas de ML con un entorno para usuarios que no sean expertos. Está enfocada para un uso de propósito general mediante lenguaje de alto nivel. Destaca por su eficiencia, accesibilidad y reusabilidad en varios contextos [BLB⁺13].

3.5.2. Aplicación del Aprendizaje Automático al Análisis de Malware

Cuando se habla del aprendizaje automático en el análisis de malware, es preciso saber que existen diferentes objetivos posibles a conseguir, distintas características a utilizar y cómo extraerlas y el tipo de algoritmo de aprendizaje automático que se utiliza.

Primero, es necesario saber cuál es el objetivo del análisis, que puede ser: la detección de malware (el principal y predominante) para determinar si una muestra es maliciosa; el análisis de similitudes, para estudiar la evolución o variación de las muestras o detectar la pertenencia de una muestra desconocida a una familia; o la detección de categorías de malware, para conocer el objetivo del malware.

Dependiendo del objetivo y del conjunto de datos que se tenga, se hará uso de un tipo concreto de algoritmo de ML [UAB19a]. Esto se puede ver reflejado en el esquema que se muestra en la Figura 3.11.

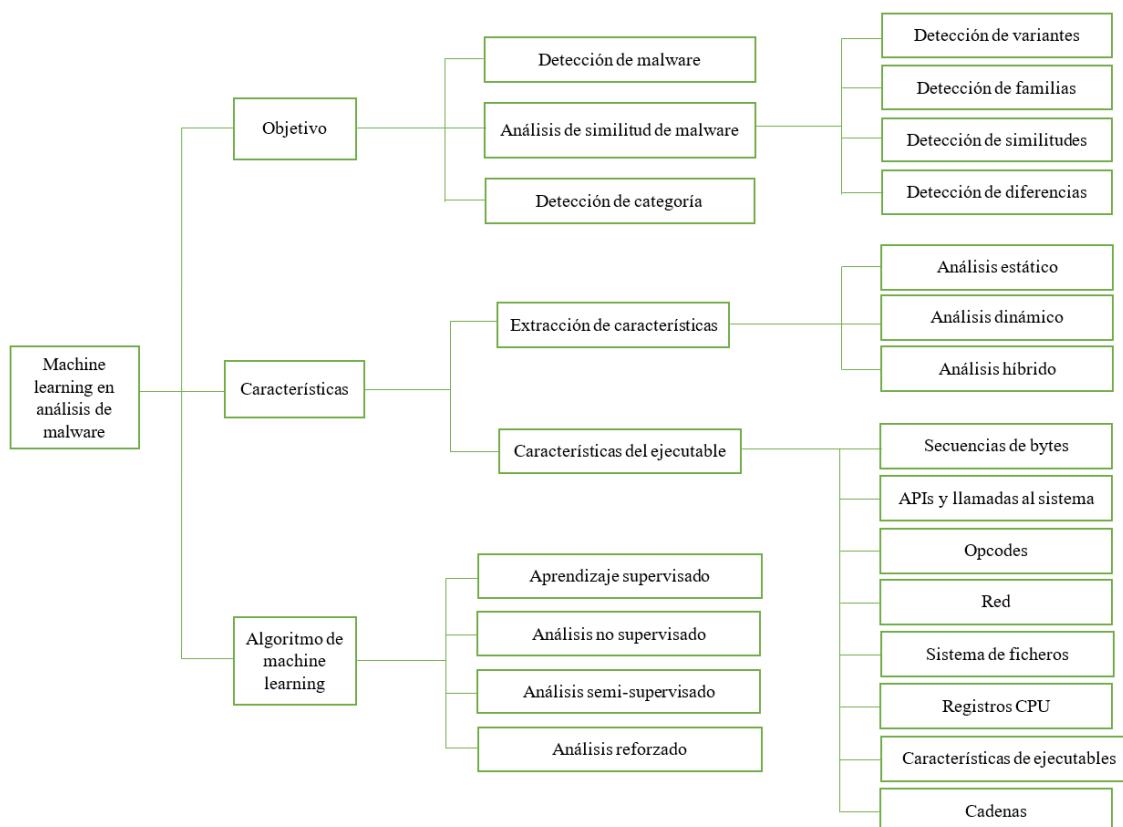


Figura 3.11: Taxonomía de las técnicas de ML en el análisis de malware [UAB19a]

Es importante también conocer las características del ejecutable que se van a utilizar para el análisis, como pueden ser secuencias de bytes, llamadas al sistema y APIs, actividad de red o cadenas. Todas estas características se extraerán haciendo uso del análisis estático (Sección 3.2), del análisis dinámico (Sección 3.3) o de la combinación de ambos.

En el caso del análisis de malware y de este trabajo, que está enfocado en detectar programas ransomware, usar un modelo de clasificación es lo más conveniente.

Capítulo 4

Estado del Arte

En este capítulo se abordan los trabajos relacionados con el tema de análisis y detección de amenazas de malware y ransomware mediante el uso de algoritmos de aprendizaje automático y otras técnicas mencionadas en el Capítulo 3. Se describirá cómo han creado sus modelos de [ML](#) y qué resultados han obtenido. Se realiza una distinción entre los estudios cuya finalidad es la detección de malware [4.1](#) y aquellos para el análisis de ransomware [4.2](#). Al final del capítulo se mostrará una tabla con los trabajos relacionados y los resultados detallados.

4.1. Trabajos sobre Análisis de Malware

Konrad Rieck *et al.* (2011) [[RTWH11](#)] proponen un *framework* (entorno o marco de trabajo) para analizar mediante el aprendizaje automático el comportamiento de malware. Este marco permite identificar nuevas clases de malware con un comportamiento parecido (*clustering* o agrupamiento) y asignarlas en una estas clases (clasificación). En la Figura [4.1](#) se muestra una descripción esquemática de este marco de análisis. Exponen un enfoque para el análisis basado en el comportamiento que puede procesar miles de archivos de malware a diario en un entorno seguro usando CWSandbox. Proponen un mapeo del comportamiento en un espacio vectorial, de modo que los patrones de comportamiento sean accesibles de manera eficiente por las técnicas de aprendizaje automático. El comportamiento que coincide con el de malware conocido se identifica mediante vectores prototipo de clústeres previamente descubiertos, y posteriormente los informes que contienen el comportamiento no identificado se agrupan para descubrir nuevos tipos de malware. Usan dos conjuntos de datos para los experimentos, uno que contiene 3.133 muestras de malware conocido sacado de la página web de CWSandbox, y otro con 33.698 muestras desconocidas obtenidas de Sunbelt Software. Limitan las muestras de cada tipo de malware a 300, las analizan usando CWSandbox, y esto da como resultado 3.133 informes de comportamiento. Su *framework* reduce los requisitos de memoria en un 94 % con un factor de aceleración de 4, lo que permite procesar 33.000 informes de comportamiento de malware en menos de 25 minutos, algo que superaba a los métodos de análisis más avanzados en su día. En los experimentos con uno de los prototipos obtienen una exactitud (*precision*) del 99,6 % y un valor-F (*F-score*) del 95 %.

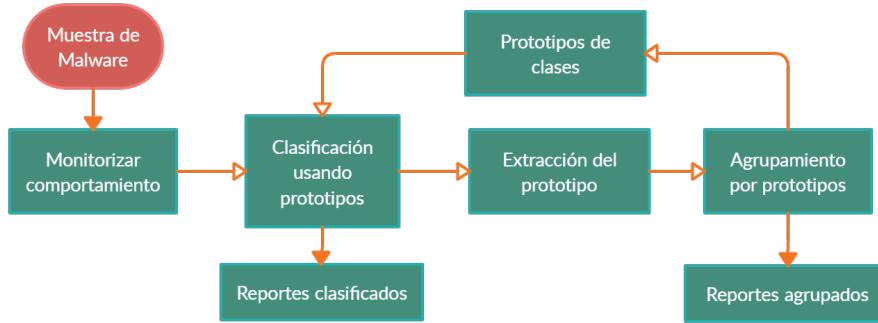


Figura 4.1: Descripción esquemática del marco de análisis de Konrad Rieck *et al.*

Mamoun Alazab *et al.* (2011) [AVWA11] proponen un sistema que emplea varias técnicas de minería de datos para detectar y clasificar malware en función de la frecuencia de las llamadas a la API de Windows. La metodología general que siguen es la mostrada en la Figura 4.2, donde primero desempaquetan el malware, extraen las API y analizan su comportamiento, comparando los resultados con una base de datos de firmas. Después de haber extraído las mejores características, crean el modelo de aprendizaje supervisado para clasificar la muestra (malware o benigno), utilizando algoritmos como NB, KNN, DT y SVM con 4 diferentes núcleos. Utilizan la validación cruzada de k iteraciones (*K-fold cross-validation*) para evaluar los resultados del análisis con $k = 10$. El conjunto de datos consta de 51.223 muestras de malware sacadas de VX Heaven y Honeynet y 15.480 muestras de software benigno. El sistema consigue una tasa de verdaderos positivos (*True Positive Rate (TPR)*) de más del 98,5 % y una tasa de falsos positivos (*False Positive Rate (FPR)*) de menos del 2,5 %, algo que no se había logrado antes.

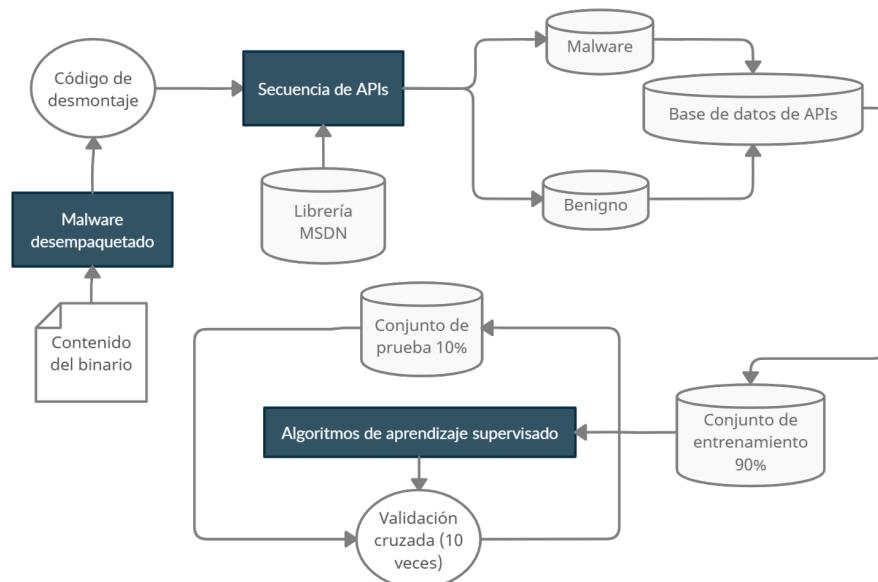


Figura 4.2: Metodología general del sistema usado en el trabajo de Mamoun Alazab *et al.*

Chandrasekar Ravi *et al.* (2012) [RM12] proponen un sistema de detección de malware que modela la secuencia de llamadas de la API de Windows con una cadena de Markov de tercer orden. La Figura 4.3 es una representación de la arquitectura de este sistema. Usan algoritmos de clasificación basados en asociación y monitorizan un subconjunto mínimo de categorías de API para asegurar la eficiencia del sistema. La novedad que introducen es el proceso de aprendizaje iterativo combinado con la monitorización del comportamiento del programa sospechoso en tiempo de ejecución, creando un sistema dinámico de detección de malware que consta de 3 fases de aprendizaje: offline, online e iterativo. La fase offline está constituida por los siguientes componentes: un conjunto de datos (*dataset*), un rastreador de llamadas de las API, una base de datos con los índices de las API, una base de datos de firmas, un generador de reglas y una base de datos de reglas. El conjunto de datos contiene 179 muestras de varios tipos de malware y 94 programas benignos. La fase online consta de: el proceso sospechoso que se va a analizar, un rastreador de llamadas de las API para ese proceso en específico, una base de datos con los índices de las API y el clasificador. El clasificador asigna una etiqueta (benigno o malware) al proceso, utilizando la secuencia rastreada de llamadas a las API y las reglas generadas en la fase offline. En la fase iterativa, después de cada clasificación, la etiqueta asignada y la secuencia de las API del proceso analizado se agregan iterativamente a la base de datos de firmas. Esto mejorará el modelo de entrenamiento al tener más información para comparar con otros procesos en el futuro. El rendimiento de este sistema supera a los sistemas de detección de malware existentes de la época, con una precisión (*accuracy*) del 90 %.

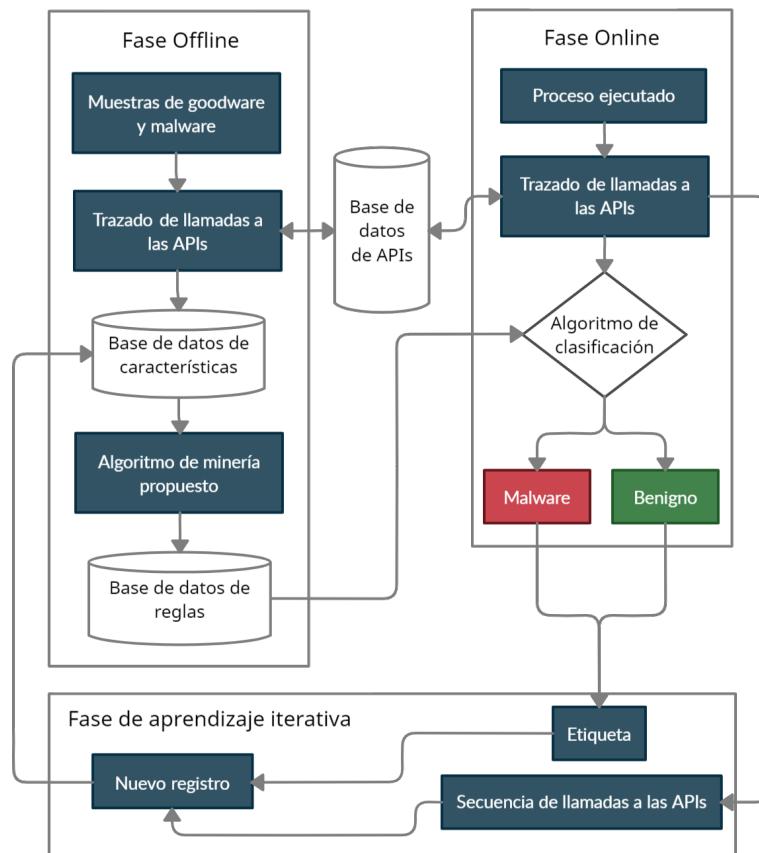


Figura 4.3: Arquitectura del sistema de detección de malware propuesto por Chandrasekar Ravi *et al.*

4.2. Trabajos sobre Análisis de Ransomware

4.2.1. Trabajos que Utilizan Características Heterogéneas

En esta sección se agrupan diversos trabajos con métodos heterogéneos para la detección de ransomware distintos a los que se van a utilizar en este estudio, pero de los cuales se puede obtener información relevante.

Amin Kharaz *et al.* en [KAM⁺16b] (2016) realizan un estudio analizando 148.223 muestras de malware reciente, mostrando que el sistema es capaz de detectar correctamente 13.367 muestras de ransomware de diferentes familias, 7.572 de ellas eran desconocidas previamente y no habían sido detectadas por ningún antivirus.

Se trata de una aplicación que se acopla al sistema al driver del sistema de ficheros, permitiendo su monitorización. Además, se combina con la creación de un entorno artificial, que detecta cambios en el escritorio para la detección de ransomware Lockscreen. Este ha de ser lo más realista posible: los datos tienen que ser reales, válidos y no deterministas para no ser reconocidos por los atacantes. Para cada extensión de archivos presente en el sistema de usuario, se crean un número aleatorio de archivos con cabeceras válidas y contenido usando bibliotecas estándar y *queries* de palabras en inglés obteniendo 100.000 frases distintas. Al igual se tiene precaución a la hora de crear los directorios y su estructura de forma realista y no determinista. Se lleva a cabo un algoritmo estadístico para establecer las fechas de creación, modificación y acceso. En cuanto a la monitorización de sistema de archivos se observa que existen patrones repetidos, usan diferentes estrategias para evitar el acceso a los archivos, hay tres patrones diferenciados que aparecen en la Figura 4.4. Se comprueba la entropía de los archivos en las operaciones de entrada y salida, en lugar de las llamadas a la API de Windows, para saber si están siendo cifrados. Se realiza mediante el *framework* Windows Filesystem Minifilter Driver, del kernel de Windows. Esto permite que Unveil este localizado en la capa más cercana posible al *Filesystem*.

Por otro lado, la detección de Lockscreens se realiza mediante capturas de pantalla de antes y después de ejecutar la muestra y se realizan métodos de análisis de imágenes para comprobar si una gran parte de la pantalla ha cambiado repentinamente, como intensidad de los *pixels*, contraste, dependencias de los píxeles cercanos, etc. También se realiza el análisis del texto de la pantalla en búsqueda de palabras claves de un mensaje de extorsión.

Amin Kharraz *et al.* (2017) en [KK17] proporcionan una defensa ante ataques ransomware, garantizando cero pérdidas de datos. Es una solución *end-point* sin limitación ya que está diseñado para proteger los archivos originales intactos ofreciendo recuperación total de datos si ocurre un ataque. Además, no depende de ninguna técnica para identificar funciones de cifrado. En la investigación de este proyecto se asume que el ransomware puede emplear cualquier técnica para atacar máquinas, es decir, puede emplear estrategias para evadir la detección, comprometer máquinas vulnerables y atacar ficheros. También se asume que el proceso puede emplear cualquier sistema de cifrado personalizado o estándar, así como longitudes de clave arbitrarias. Sumado a esto, se plantea que el usuario puede instalar programas de fuentes no fiables y así el código malicioso puede ejecutarse con privilegios del usuario.

Redemption media el acceso al sistema de ficheros y redirige cada petición de escritura a un área protegida para ejecutar la acción en un archivo reflejado sin cambiar el estado del archivo original. De esta forma, cuando se sabe que no supone un riesgo de ransomware, se aplican los cambios al archivo original. Redemption sabe que la acción no es maliciosa basándose en una puntuación que establece al proceso (usando como métricas la entropía, la operación de borrado y la reescritura de contenido) y que compara con un umbral establecido.

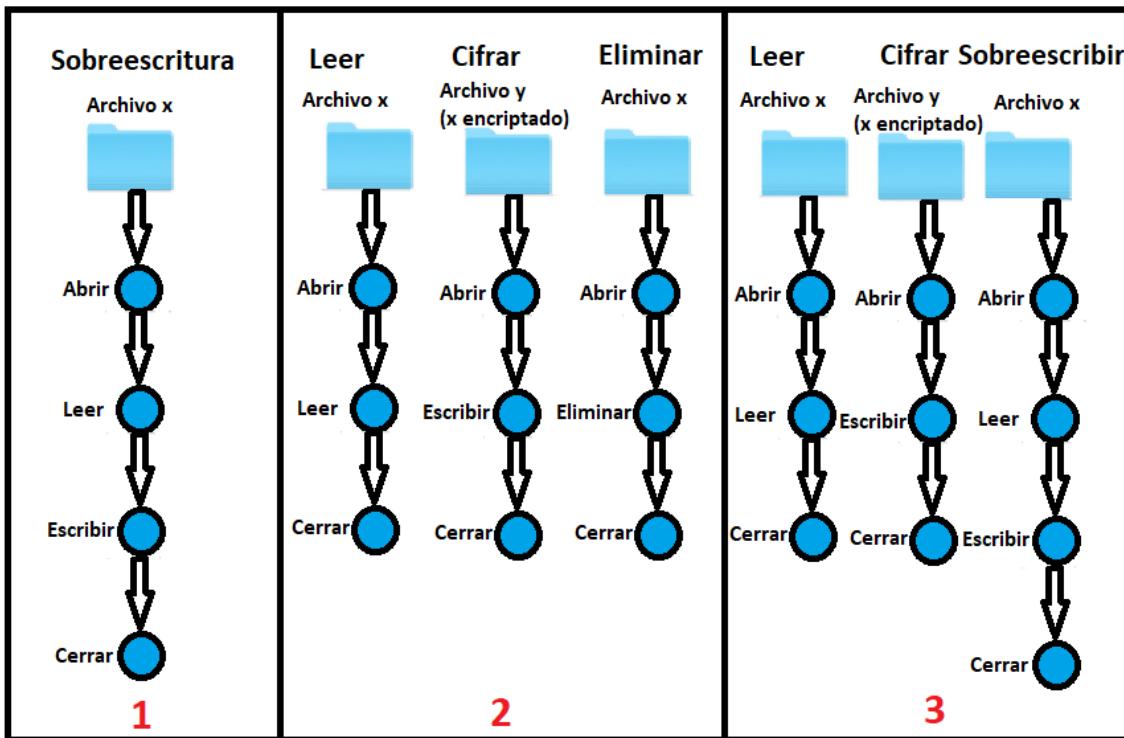


Figura 4.4: Patrones del ransomware para cifrar archivos: 1- El atacante sobrescribe el fichero del usuario con la versión cifrada. 2- El atacante lee un fichero, lo cifra en otro distinto y elimina el original. 3- El atacante lee un fichero, crea una nueva versión cifrada, sobre escribiendo el fichero original

Este sistema hace mucho énfasis en garantizar la consistencia de los datos, por ejemplo cuando el sistema se cuelga, si los metadatos no están actualizados, el sistema trata los cambios como incompletos y descarta los cambios realizados retrocediendo y vuelve a repetir la acción.

En cuanto a los resultados, Redemption detecta ransomware en todas las familias de ransomware utilizadas para las pruebas, con una media de observación de 5 ficheros, no encontrando problemas si el sistema no ha sido entrenado con ciertas familias desconocidas. Además, se obtiene que las aplicaciones tan solo tardan un 2,6 % más en completar sus tareas por utilizar Redemption, demostrando su eficiencia y el ligero impacto al rendimiento.

Shagufta Mehnaz *et al.* (2018) en [MMB18b] exponen que la mayoría de herramientas de detección de ransomware no son capaces de reconocer el agente maligno en tiempo real, lo que conlleva el cifrado de los archivos que luego y las técnicas de descifrado actuales están limitadas, y tienen una tasa de falsos negativos elevada: no son capaces de distinguir el malware de grandes operaciones con archivos como el cifrado benigno o la compresión. RWGuard es una herramienta que cumple los siguientes puntos:

1. Emplea técnicas de señuelo. Se colocan archivos falsos en nuestro sistema, que el usuario nunca va a escribir. Por lo tanto, si se detecta alguna operación con ellos se deduce al instante que estamos siendo atacados.
2. Monitoriza los procesos en ejecución, así como el sistema de ficheros.

3. Omite los cambios benignos del usuario en los archivos (*file change monitoring*): mediante un mecanismo que monitoriza las propiedades del archivo modificado. Además se utiliza CryptoAPI para que los usuarios y las aplicaciones modifiquen los archivos, así si se producen llamadas a ésta, sabremos que es un cifrado benigno.

El tiempo medio del sistema para la detección de todos los procesos malignos del ransomware es de 8.87 segundos. Se puede descifrar los datos de ataques realizados por las familias Locky, CryptoWall, y CryptoLocker, ya que se pueden recuperar los parámetros del cifrado (como la clave de descifrado) de las llamadas a las funciones criptográficas debido a la monitorización que se realiza.

Las limitaciones del sistema residen en que está basado en el registro de llamadas *I/O request packets (IRP)* y actividad de archivos. El lapso de tiempo entre el registro de estas actividades y su análisis en busca de anomalías proporciona una pequeña ventana para que el ransomware pueda realizar sus actividades malignas.

Fei Tang *et al.* (2020) en [TML⁺20] proponen un sistema cuya finalidad es detectar ransomware de cifrado, basándose en la técnica de introspección de máquina virtual *Virtual Machine Introspection (VMI)*. Esta técnica permite, monitorizar cualquier operación que se realiza tanto en el sistema de ficheros como en la comunicación de red, mediante la utilización de máquinas virtuales, en este caso con una Windows 7, de forma que el sistema que queremos defender no se infecta por el malware. El sistema se ha de establecer fuera del sistema operativo. En otro caso, el programa malicioso podría escalar privilegios en el sistema y podría ejecutar órdenes o eliminar/modificar archivos. En este trabajo se explica cómo se realiza el escalado de privilegios y cómo poder detectarlo.

La técnica que emplean para la detección del ransomware se basa en calcular la entropía de los datos escritos en los ficheros, puesto que los datos cifrados tienen un alto valor, y se comparan con un umbral definido para conocer si el sistema de ficheros está siendo atacado por ransomware de cifrado.

También cabe destacar la monitorización de las redes, a través de la cual se puede rastrear la forma en que los atacantes obtienen de servidores remotos las claves de cifrado y posteriormente pueden enviarlas.

El sistema predice el 100 % de los patrones de entrada/salida provocados en el sistema de ficheros por el ransomware, y tan solo en un 6,23 % de los ejemplos no se apreció ninguna comunicación por red producida por el ransomware. Sumando la capacidad de detección de estas técnicas, teniendo en cuenta falsos negativos y positivos, se calcula una precisión del 100 % en las muestras utilizadas.

En cuanto a las limitaciones, este estudio solo se centra en una serie de patrones tanto del sistema de ficheros como de la red, por lo tanto, pueden existir ejemplos de ransomware que utilicen diferentes patrones o modos de operación. Por otro lado, tampoco es posible recuperar los datos cifrados porque no se realiza la detección en tiempo real.

Amos Ren *et al.* (2020) en [RLH⁺20] proponen como objetivo aislar los archivos potencialmente maliciosos (fundamentalmente el ransomware de la familia Petya) en máquinas virtuales antes de que dañen el equipo mediante un sistema de seguridad en tres niveles.

El primer nivel se trata de una extensión para navegador web que detecta las páginas que intenta descargar cualquier contenido sin autorización, bloqueando el sitio. Está construido mediante dos capas:

1. La primera capa usa un método híbrido de detección basado en firmas, que permite analizar las propiedades sintácticas y estructurales del programa.

2. La segunda capa utiliza una detección basada en anomalías, que es capaz de detectar nuevos tipos de malware.

El segundo nivel hace uso de las máquinas virtuales para crear un entorno seguro, dentro del que actúa el tercer nivel, que usa soluciones anti-ransomware para escanear los archivos y eliminar las amenazas que hayan pasado los niveles anteriores.

La creación del entorno seguro además provoca que el atacante piense que el ataque ha sido exitoso, evitando así un segundo ataque. La solución se basa en la creación de un entorno virtual individual para cada descarga, para evitar infecciones de entornos seguros por parte de entornos infectados.

La limitación fundamental de este sistema reside en la dificultad de los ordenadores actuales en ejecutar varias máquinas virtuales al mismo tiempo, por lo que solo se permiten 4 descargas simultaneas.

Firoz Khan *et al.* (2020) exponen en [KNR⁺20] una nueva técnica para la detección de ransomware basadas en secuencias de *Deoxyribonucleic Acid (DNA)* digital, rechazando aquellas basadas en firmas.

Este mecanismo primero selecciona las características para generar la secuencia de DNA digital y posteriormente clasificarlas mediante un algoritmo de ML, como se aprecia en la Figura 4.5. El DNA se usa para construir proteínas y otros componentes celulares de

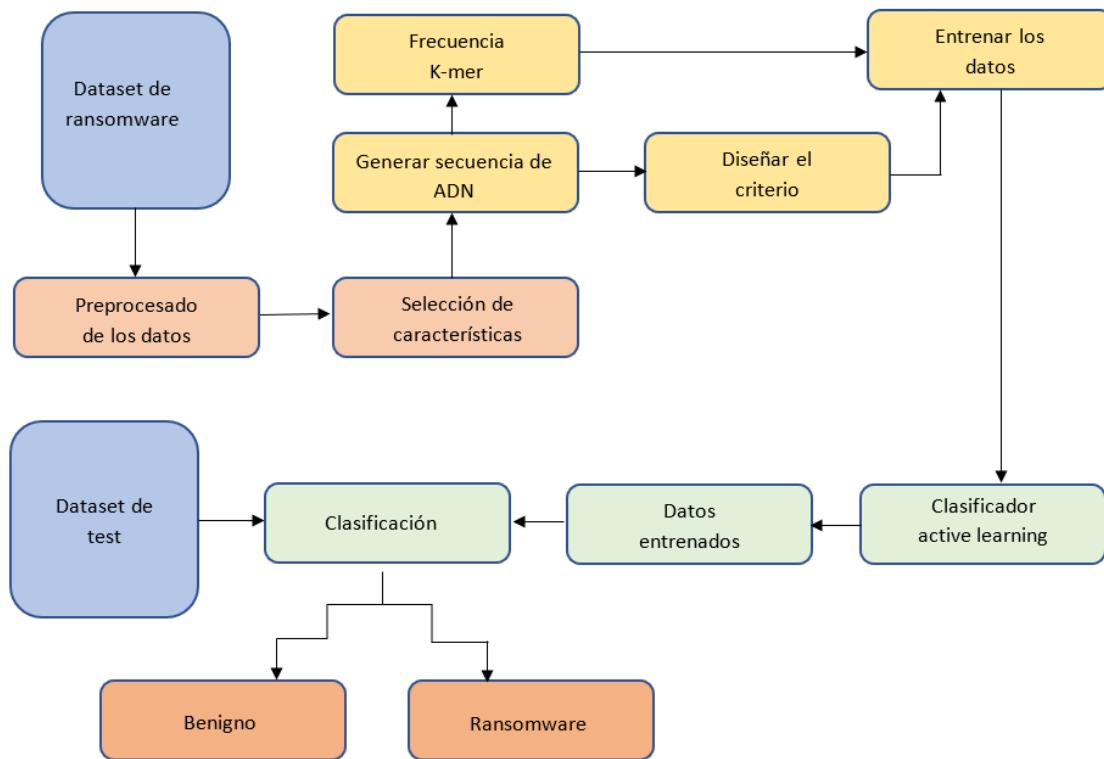


Figura 4.5: Diagrama de secuencia del proyecto

los seres vivos. Está compuesta por largas cadenas conformadas por las bases nitrogenadas: Adenina (A), Guanina (G), Citosina (C) y Timina (T). Para la creación de la secuencia de DNA digital, se realiza una correspondencia binaria para mapear estas moléculas como se observa en la Tabla 4.1.

Tabla 4.1: Mapeo de las bases nitrogenadas en código binario

Codificación	Moléculas
00	A
01	C
10	G
11	T

Posteriormente, se generan secuencias de [DNA](#) digital de manera aleatoria para entrenar un modelo de [ML](#). Se utiliza *active learning*, un método por el que se pueden realizar consultas de manera interactiva para etiquetar nuevos datos con su salida deseada. Finalmente, se usan algoritmos de clasificación ([NB](#), [RF](#) y Optimización Mínima Secuencial) para detectar las familias de ransomware.

El *dataset* utilizado se compone de 1524 muestras, de las cuales 582 pertenecen a ransomware y 942 a goodware, con un total de 30970 características.

4.2.2. Trabajos que Utilizan Llamadas a la API de Windows

En esta sección se agrupan los trabajos que usan las llamadas a la [API](#) de Windows como característica para desarrollar sus modelos, al igual que este trabajo.

Daniele Sgandurra *et al.* (2016) [[SMGML16](#)] presentan EldeRan, un sistema que usa un algoritmo de aprendizaje automático para clasificar y analizar ransomware dinámicamente. EldeRan se basa en la observación de que las muestras de ransomware suelen realizar acciones que son únicas o significativas con respecto a las realizadas por un programa benigno. Por lo tanto, EldeRan monitoriza los programas sospechosos en sus primeras fases de instalación en un entorno controlado con Cuckoo Sandbox, comprobando si sus acciones coinciden con las que haría un ransomware. La Figura 4.6 representa este sistema. Analizan un conjunto de datos descargados de VirusShare que contiene 582 muestras de ransomware pertenecientes a 11 familias y 942 aplicaciones benignas. A partir de estos datos, se analizan las siguientes características: llamadas a la [API](#) de Windows, operaciones de clave de registro y del sistema de archivos (lectura, apertura, escritura y eliminación), el conjunto de operaciones realizadas con cierta extensión, operaciones de directorio, archivos eliminados (es decir, el conjunto de archivos que se eliminan por una aplicación durante su instalación) y las cadenas de caracteres incrustadas en el archivo. Todas las funciones, excepto las cadenas, se recopilan mientras se analiza dinámicamente el ransomware. Después de esta fase de análisis, se aplica un algoritmo de selección de características para seleccionar las más relevantes. Por último, las matrices que contienen estas características se utilizan en un clasificador [LR](#) regularizado que devuelve si es ransomware o un archivo benigno. EldeRan no se basa en firmas, por lo que no necesita tener conocimientos previos de ransomware y puede detectar nuevas familias. Comparan EldeRan con los algoritmos [SVM](#) y [NB](#) y demuestran que el rendimiento máximo de estos algoritmos se alcanza con 400 características. Por lo tanto, agregar más características no mejora la precisión, por lo que la selección de características es importante para reducir la complejidad de los algoritmos de aprendizaje automático sin afectar el rendimiento. EldeRan alcanza un área bajo la curva *Receiver Operating Characteristic (ROC)* del 99,5 % y tiene una tasa de fallos de 2,4 %, inferior a la de [SVM](#) (4,2 %) y [NB](#) (8,0 %).

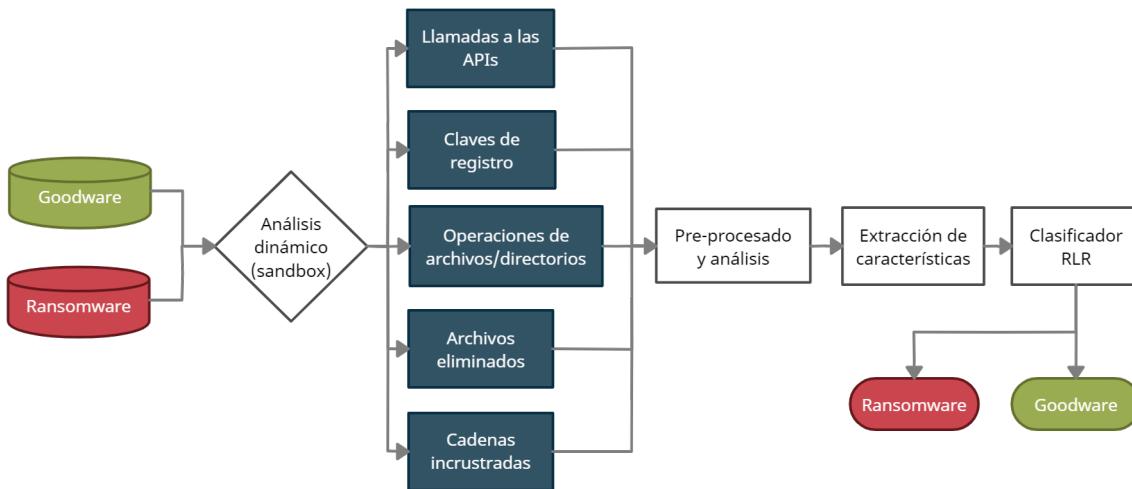


Figura 4.6: Representación del sistema EldeRan [SMGML16]

Zhi-Guo Chen *et al.* (2017) [CKYK17] proponen un sistema dinámico representado en la Figura 4.7 que utiliza gráficos de flujo de llamadas API (*Calls Flow Graphs* (CFG)) y algoritmos como RF, SVM, NB y LR para detectar programas ransomware conocidos y desconocidos. Este sistema utiliza la herramienta *API Monitor* para recopilar los datos de las llamadas a las API que realiza el programa sospechoso cuando se ejecuta en una máquina virtual. Después, el sistema genera los CFG para representar el comportamiento del programa y convierte estos gráficos en vectores de características etiquetados. Posteriormente selecciona el mínimo número de características para reducir el tiempo de la fase de aprendizaje y mejorar el rendimiento. Por último, se usan los algoritmos anteriormente mencionados para construir el modelo de detección que prediga si un programa es ransomware o no, entrenándolo con los vectores de características generados.

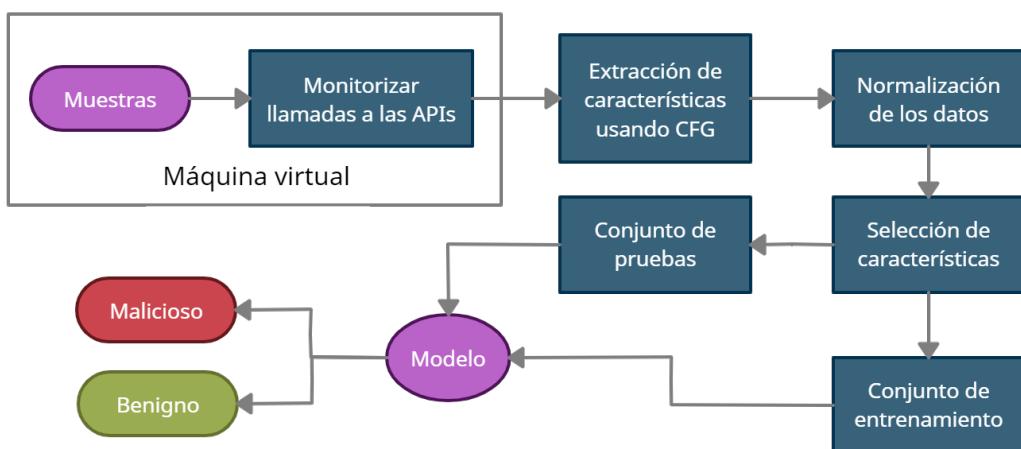


Figura 4.7: Representación del sistema dinámico de detección de ransomware [CKYK17]

Hacen uso de la validación cruzada de k iteraciones (*K-fold cross-validation*) para entrenar y probar el modelo. Esta validación consiste en dividir aleatoriamente el conjunto de datos original en k subconjuntos de igual tamaño, usando los k-1 conjuntos para el

entrenamiento y el conjunto restante para las pruebas. Este proceso se repite k veces para que todos los subconjuntos se usen para las pruebas una vez. En este trabajo, se realiza la validación con $k = 10$. Tras varios experimentos, los mejores resultados se obtienen cuando se normalizan los datos y se aplican criterios de correlación y ganancia, obteniendo un 98,2% en precisión (*accuracy*) y un 97,6% en **TPR** con el algoritmo **LR**. En cuanto al conjunto de datos usado, recopilan 83 muestras de ransomware diferentes y 85 muestras de software benigno.

Vinayakumar R *et al.* (2017) [VSVG17] proponen un sistema con una red neuronal profunda, en específico un perceptrón multicapa (*Multilayer perceptron (MLP)*), para la detección y clasificación de ransomware, usando las llamadas a las **API**, algo que usan todos los trabajos. Analizan 7 familias de ransomware diferentes que han sacado de diversas fuentes, como Open Malware, Contagio Malware Dump, Malwr, theZoo, VirusTotal y VirusShare, recopilando un total de 755 muestras de ransomware y 219 de software benigno. Consideran alrededor de 131 llamadas **API** sacadas de los registros de Cuckoo Sandbox tras analizar las muestras con dicha herramienta. La red **MLP** es una función parametrizada, lo que significa que la **TPR** de ransomware depende de los parámetros óptimos, la cantidad de capas ocultas, número de unidades, la tasa de aprendizaje, etc. Todos los experimentos con **MLP** se realizan en tres capas con 1000, 500 y 250 unidades y se ejecutan con 500 ciclos de aprendizaje (llamados *epochs*), con una tasa de aprendizaje entre 0,01 y 0,5. Este sistema alcanza una precisión (*accuracy*) de 1,0 para detectar si un programa es ransomware o no.

Yuki Takeuchi *et al.* (2018) [TSF18] proponen un sistema de detección de ransomware utilizando máquinas de soporte vectorial (**SVM**) y transformando las llamadas **API** que realiza en ransomware en un conjunto de datos vectoriales. A diferencia de las soluciones existentes, este sistema propuesto analiza en profundidad el historial de llamadas a las **API** con más detalle, demostrando una mejora en la **TPR** correcta de ransomware. Usan **SVM** porque generalmente tiene una generalización más alta que los otros algoritmos. La generalización es la capacidad de predecir datos desconocidos, y en este caso es muy útil para detectar ransomware desconocido, ya que no siempre se tienen muestras de todos los tipos posibles de ransomware. Ejecutan 276 muestras de ransomware y 312 muestras de programas benignos o *goodware* en un entorno controlado y seguro con Cuckoo Sandbox, como se ha visto en anteriores trabajos. Después de la ejecución, Cuckoo Sandbox genera reportes que no solo contienen las llamadas a la **API** de Windows, sino también los accesos a los archivos, los árboles de procesos, etc. A partir de estos reportes, solo se extraen las llamadas a la **API** y se guardan como vectores que serán usados en **SVM**. Para evaluar el sistema, usan la métrica de precisión (*accuracy*) y obtienen un 97,48%. También calculan la tasa de fallos, que es el número de falsos negativos (*False Negatives (FN)*) entre el número total de muestras, obteniendo un 1,64%.

Seong Il Bae *et al.* (2019) [BLI19] proponen un sistema de detección de ransomware en un entorno virtual en Windows 7 utilizando VMware que pueda distinguir entre ransomware y archivos benignos, así como entre ransomware y malware. Afirman que los métodos de detección de malware basados en firmas tienen dificultades para detectar ataques de día cero, por lo que no son adecuados para proteger los archivos de los usuarios. Por lo tanto, plantean un nuevo mecanismo de protección especializado que se centra en operaciones específicas de ransomware para poder diferenciarlos de otros malware y archivos benignos. La Figura 4.8 muestra el diagrama de flujo del sistema propuesto. En este trabajo extraen las secuencias de invocación de la **API** de Windows mediante la herramienta intel PIN tool que realizan 1000 archivos de ransomware, 900 archivos de malware y 300 archivos ejecutables benignos. Si el tamaño del reporte de las llamadas es

inferior a 10 KB, el reporte se descarta, ya que consideran que el archivo no se ejecutó correctamente. Con esa información generan los vectores de características utilizando un conjunto de n-grama, una subsecuencia de n elementos, y valores de *Class Frequency - Non-Class Frequency* (CF-NCF), un indicador para modelos de clasificación basado en *Term Frecuency - Inverse Term Frecuency* (TF-IDF), que es un algoritmo que permite medir la importancia de una palabra en un conjunto de documentos. El rango del valor n de n-grama es entre 1 y 4, ya que comentan que las matrices se vuelven demasiado grandes para ser calculadas cuando n es mayor a cuatro. Posteriormente usan seis algoritmos de aprendizaje automático para crear su modelo de ML con los vectores de características generados. Los seis algoritmos que utilizan son: RF, LR, NB, descenso de gradiente o *Stochastic Gradient Descent* (SGD), KNN y SVM. Se realiza una validación cruzada a los datos de entrenamiento, que constituyen el 80 % del dataset, y a los de test, el 20 % restante. Dichos procesos de entrenamiento y test fueron repetidos 5 veces. Al realizar sus experimentos con todos los algoritmos, consiguen una precisión (*accuracy*) de entre 97 % y 98,65 %.

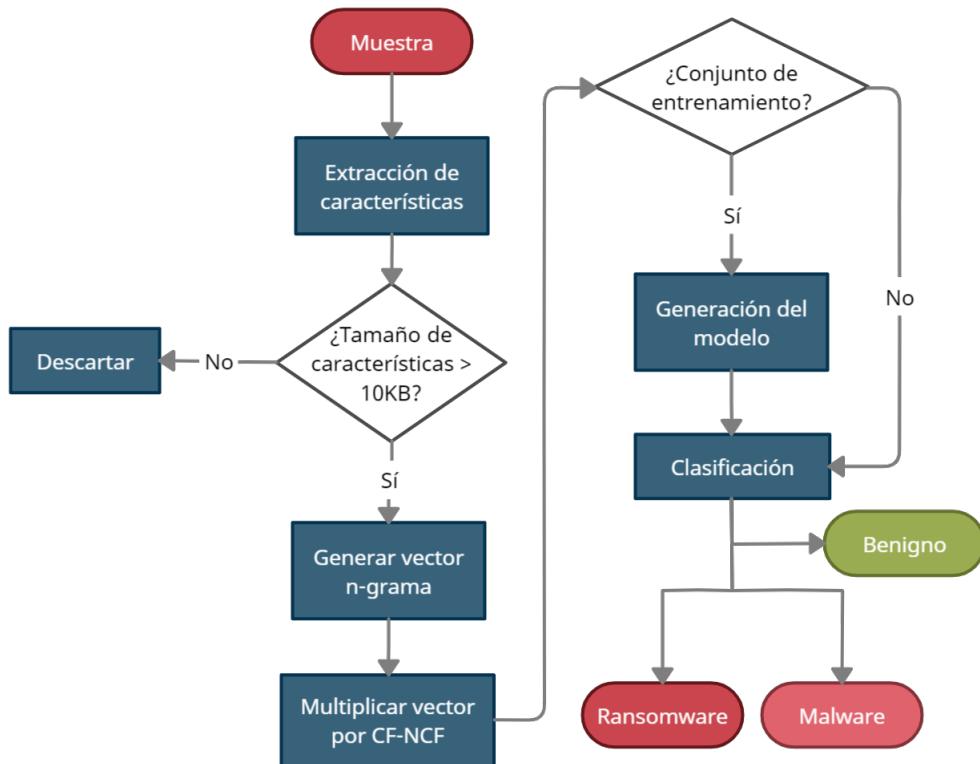


Figura 4.8: Diagrama de flujo del sistema propuesto [BLI19]

Mahadevan Supramaniam *et al.* (2019) [KAJS19] proponen un sistema de dos fases que detecta cripto-ransomware antes de que pueda cifrar los archivos del usuario. Las dos fases son:

- **Algoritmo de aprendizaje:** El archivo de muestra se introduce en Cuckoo Sandbox, el cual extrae las llamadas a las API de Windows de cifrado (un total de 232). Posteriormente se analizan utilizando un algoritmo de aprendizaje que determina si el programa sospechoso es un cripto-ransomware o no. Este algoritmo

selecciona características de forma aleatoria para generar árboles de decisión, ya que hay una gran cantidad de llamadas a las API y esto causa *overfitting* o “sobreajuste”. Si el programa es un cripto-ransomware, el algoritmo genera una firma del programa malicioso y envía una notificación a un repositorio de firmas.

- **Repositorio de firmas:** Una vez que se ha clasificado la muestra, se almacena un la firma de ella utilizando el algoritmo RSA en una base de datos *Structured Query Language (SQL)*. La finalidad es que, cuando se vaya a ejecutar un archivo, primero se compruebe en la base de datos, de forma más rápida y eficiente, mediante consultas SQL. Tener este repositorio permite la detección de las amenazas en una etapa mucho más temprana, aunque solo puede detectar cripto-ransomware conocido.

Las dos fases de este sistema, el cual está representado en la Figura 4.9, forman dos capas de detección temprana para cripto-ransomware, garantizando al usuario que no va a perder ningún archivo. En este trabajo se centran en la primera fase, desarrollando un algoritmo de aprendizaje al que llaman PEDA, obteniendo una TPR del 98,72 %, un *Area under the curve (AUC)* del 99,3 % y una tasa de falsos positivos (FPR) un 1,56 % más baja en comparación con los algoritmos NB, RF, una mezcla de los dos anteriores (método de ensemble) y con el algoritmo EldeRan desarrollado en [SMGML16]. Usan un *dataset* extraído por el *Resilient Information Systems Security (RISS)* en 2016, que contiene datos de API de 10 familias de ransomware y de software benigno, en concreto 582 muestras de ransomware y 942 de software benigno.

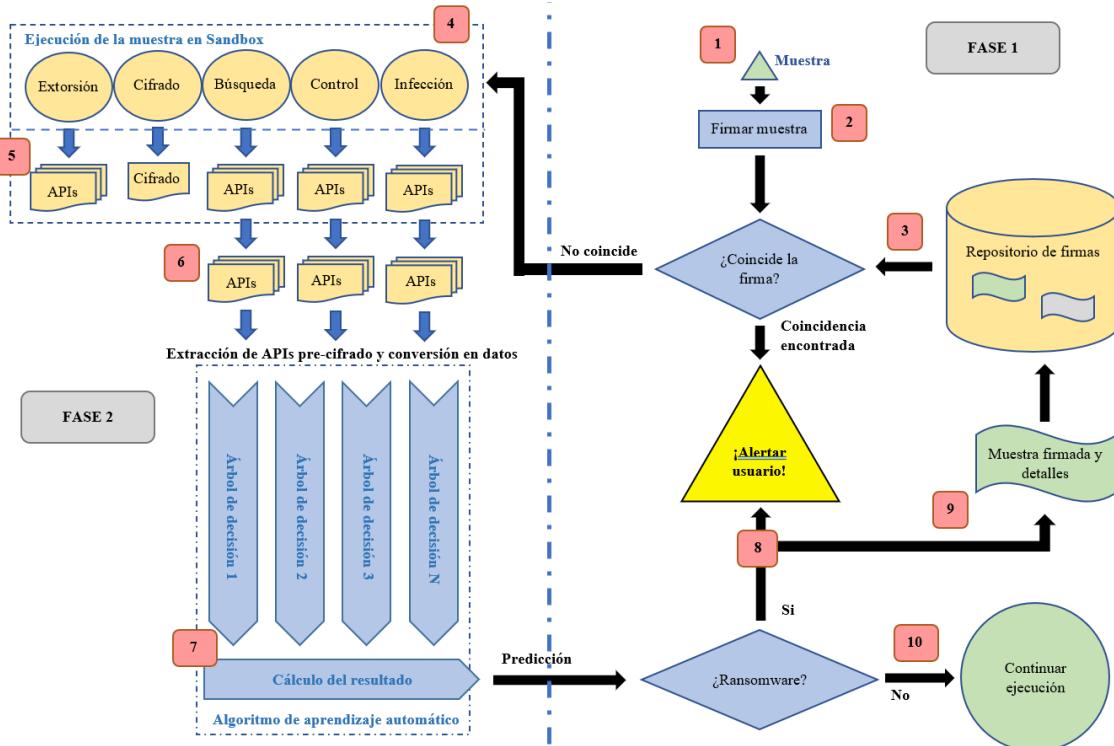


Figura 4.9: Modelo del sistema propuesto por [KAJS19]

Brijesh Jethva *et al.* (2019) [Jet19] propone un sistema de detección de ransomware dinámico mediante la combinación de un modelo de aprendizaje automático con modelos de entropía y firmas de archivos. Con este nuevo enfoque, se observa que el objetivo del ransomware son áreas específicas de claves de registro y su ejecución implica operaciones de alta entropía en extensiones de archivo desconocidas. Las características más seleccionadas son las llamadas a la API y las claves de las operaciones de registro, pero la presencia de otras características como las bibliotecas de enlaces dinámicos (DLL), los directorios accedidos, las líneas de comando, cadenas insertadas, etc. Con todas estas características, se seleccionan las más importantes con la ayuda de métodos de selección de características para evitar el *overfitting* o sobreajuste para posteriormente clasificarlas mediante técnicas de aprendizaje automático, que son SVM, LR y RF. Para realizar el estudio, utilizan un conjunto de datos que consta de 666 muestras de ransomware y 103 archivos benignos. Para balancear estos datos, usan la técnica *Synthetic Minority Oversampling Technique (SMOTE)*, la cual utiliza el algoritmo KNN para generar datos nuevos para las clases con menos muestras. En este caso, se generan datos para ambas clases, obteniendo 1.072 muestras de cada una. En los experimentos, el algoritmo que mejor rendimiento tiene es el de LR, obteniendo una TPR del 100 %, una precisión (*accuracy*) del 98,7 % y una tasa de falsos positivos del 1,41 %.

Jinsoo Hwang *et al.* (2020) [HKLK20b] proponen un modelo de detección de ransomware con dos etapas. La primera etapa es un modelo de Markov constituido por dos cadenas de Markov, una para ransomware y la otra para programas benignos. Una cadena de Markov se crea con un espacio de estados y una matriz de probabilidades de transición. El espacio de estado abarca todas las llamadas a la API de Windows realizadas tanto por las muestras de ransomware como por las de *goodware*. Estas cadenas se utilizan para decidir si la muestra entrante es ransomware o no, comparando las probabilidades de que una muestra pertenezca a cada clase. La segunda etapa consiste en un modelo de aprendizaje automático, usando RF para controlar la tasa de falsos negativos (*False Negative Rate (FNR)*) que devuelve el modelo de Markov, ya que es bastante alta (20 %). Para los experimentos, recopilan 1909 muestras de ransomware de VirusShare y 1139 muestras de *goodware* de Softonic, ejecutándolas dentro de Cuckoo Sandbox para obtener los reportes de donde extraen la secuencia de llamadas a las API. Al final de las dos etapas, obtienen una precisión (*accuracy*) del 97,3 %, una tasa de falsos positivos (*FPR*) del 4,8 % y una tasa de falsos negativos (*FNR*) del 1,5 %.

S.H. Kok *et al.* (2020) [KAJ20] proponen un sistema que detecte cripto-ransomware antes de que entre en la etapa de cifrado de archivos (*Pre-Encryption Detection Algorithm (PEDA)*). PEDa proporciona dos niveles de detección: el primer nivel consiste en una comparación de firmas, generadas con *Secure Hashing Algorithm (SHA)*-256. Se compara la firma del archivo sospechoso con firmas de diferentes tipos de ransomware conocidos almacenadas en un repositorio MySQL. Si coinciden los *hashes*, se alerta al usuario de que el archivo es un ransomware. Si no, PEDa entra en la segunda etapa, que consiste en enviar ese archivo a Cuckoo Sandbox para que lo analice y obtenga el reporte de su comportamiento. De ese reporte se extraen las API que se llaman antes de que empiece el proceso de cifrado, por lo que el algoritmo sacará todas las API hasta que encuentre una que empiece por “crypt”. Se identifican 14 APIs importantes para diferenciar entre ransomware y goodware, con 3 de ellas encontradas en la mayor parte de ransomwares. Posteriormente las API extraídas (232 en total) se guardarán en un archivo de valores separados por comas (*Comma-separated values (CSV)*), el cual se introducirá en el algoritmo de aprendizaje automático RF para que prediga si el archivo sospechoso es un ransomware o no. Si la predicción lo identifica como ransomware, PEDa pondrá en

cuarentena el archivo, alertará al usuario y almacenará su firma en el repositorio MySQL de firmas. La Figura 4.10 muestra el flujo de datos del sistema PEDA. Para entrenar el modelo de ML, utilizan tres *datasets* con un total de 942 muestras de *goodware* y 1691 muestras de ransomware de VirusShare y theZoo. El modelo obtiene una exhaustividad (*recall*) del 99,9 % con validación cruzada de 10 iteraciones en dos de las tres *datasets*, usando un ratio de entrenamiento:prueba 80:20. En cuanto a las limitaciones, PEDA tiene una importante dependencia de uso de las APIs de Windows, lo que supone que no sería capaz de detectar ransomware que utilice su propio código nativo de cifrado, lo que convierte este algoritmo en un suplemento para la detección de ransomware y no un sistema completo para ello.

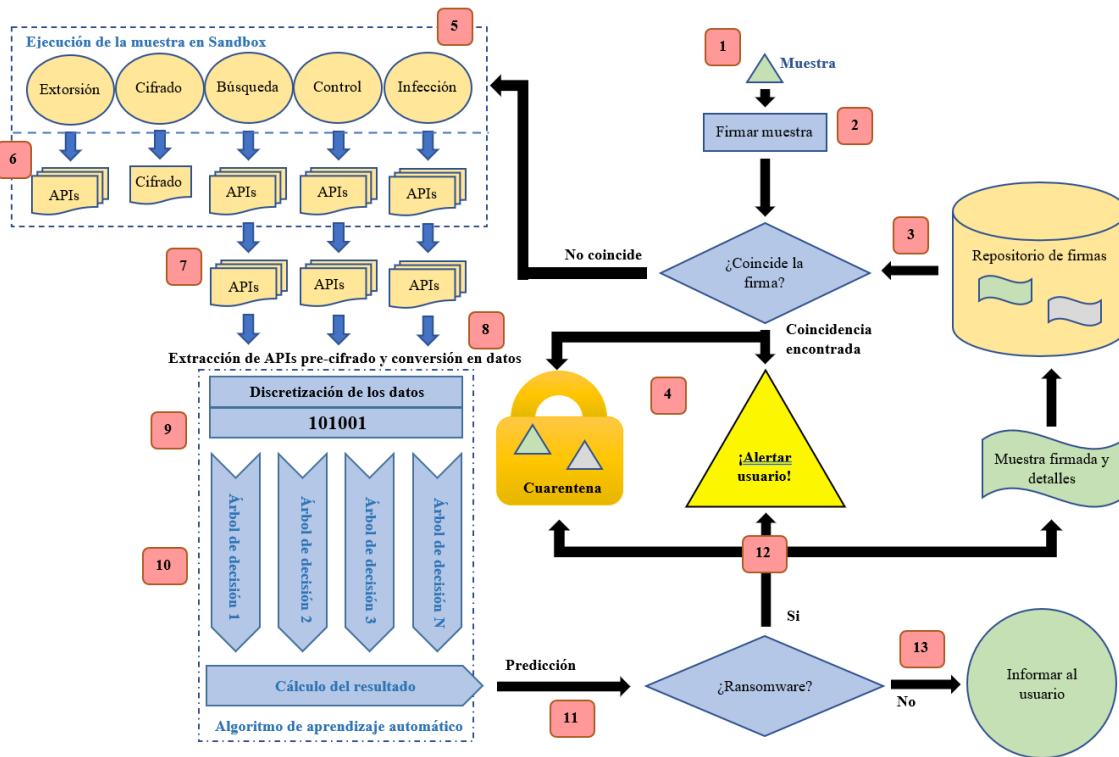


Figura 4.10: Representación del flujo de datos del sistema PEDA propuesto por S.H. Kok *et al.* [KAJ20]

La detección basada en el análisis del comportamiento se basa en identificar cambios relevantes en el sistema. Abdullahi Arabo *et al.* en [ADPC20b] (2020) desarrollaron un programa que recopila del sistema las llamadas a DLL de API, el uso de los recursos del sistema, y los archivos utilizados por los procesos. Dicha información se discretiza y posteriormente toma dos caminos: a una solución de ML, utilizando árboles de decisión, RF y redes neuronales, que será capaz de detectar ransomware después del suficiente entrenamiento, y a un sistema de detección mediante umbrales, debido a que no cualquier tipo de datos, como las cadenas, pueden ser introducidos a una solución de ML.

La detección del ransomware se lleva a cabo mediante un *script* de Python. Se ejecutan ataques ransomware en una máquina virtual Windows 7, que destruirían dicho *script* con la información que ha recopilado, así que, dicha máquina está conectada con otra base Linux mediante un canal seguro para el paso de la información. Se crean dos hilos principales diferenciados porque hay datos que no tardan el mismo tiempo en ser obtenidos: Hilo “analysis”, que recoge datos variables en bucle hasta el final del proceso y el Hilo “info”,

el cual recoge datos constantes, los manda y finaliza. También existen otros tres para leer el nombre de los ficheros, y reconocer su extensión, puesto que los datos cifrados de forma maligna adaptan extensiones reconocibles (p.e.: .wncry), los *tagfiles* y para los datos que son muy lentos de obtener. Se utiliza también el programa Tiny tracer: cuando se ejecuta un programa con él, genera un archivo con todas las llamadas a **DLL** de **API**.

Se crea un sistema basado en media de pesos para reconocer si un dato detecta el ransomware o no. Los datos son procesados por funciones, que tienen un cierto umbral: si lo sobrepasan devolverán 1, por el contrario 0, y se compara con un coeficiente, que depende del umbral y la función. Al final, todos estos valores se utilizan para elaborar una media. Si es por encima de 0,5 es sospechoso de ser malicioso y salta un pop-up al usuario para permitir matar al proceso.

Sana Aurangzeb *et al.* (2021) [ARA+21] exponen un trabajo basado principalmente en el análisis dinámico enfocado a los *Hardware Performance Counter (HPC)*, que son variables propias de la actividad del hardware como por ejemplo ciclos de reloj, accesos a caché, fallos de caché, instrucciones de salto tomadas, etc. En este estudio se utilizan 11 variables **HPC** para la clasificación de ransomware, que se hará entre ransomware y non-ransomware, siendo non-ransomware muestras de malware y no de goodware. Esto se debe ya que se el ransomware está considerado uno de los tipos de malware que mayor daño provocan en términos financieros. A la hora de construir el *dataset*, se obtienen 160 muestras de malware de VirusShare y tras un análisis estático en la misma plataforma se etiquetan como ransomware o non-ransomware. Una vez obtenidas las muestras, se ejecutan en un entorno seguro y se recoge la información relativa a los **HPC**. Para asegurar la precisión y corrección de los datos se ejecuta cada muestra tres veces en una máquina virtual. Los 11 **HPC** recogidos para entrenar el modelo son *task clock*, *context switching*, *CPU utilized*, *CPU migrations*, *page faults*, *CPU cycles*, *cache-misses*, *instruction retires*, *branch taken*, *branch-misses* y *execution time*. Con estas variables se construye una matriz de correlación para seleccionar los atributos concretos a utilizar y así ahorrar costes computacionales. Para el entrenamiento de los datos se utiliza un 80 % del *dataset* y el resto son usados para prueba. Además, se utiliza el método *k-fold cross-validation*. En la Figura 4.11 se puede ver el flujo del proceso de entrenamiento y prueba. Se utilizan cuatro modelos de clasificación, árboles de decisión (ofrece una precisión de 0,94), bosques aleatorios (precisión de 0,97), potenciación de gradiente (precisión de 0,94) y potenciación extrema de gradiente (ofrece una precisión de 0,97). Las conclusiones obtenidas de este estudio es que existe la posibilidad de explotar características hardware para la detección de ransomware. Por otro lado, como no se trabaja con muestras benignas, se desconoce el comportamiento que tendrá el uso de estas características para detectar ransomware, además, la recogida de características es específica de un sistema con una configuración hardware determinada, lo que implica que si el sistema en el que se recogen los datos tiene otra arquitectura, el modelo no será aplicable.

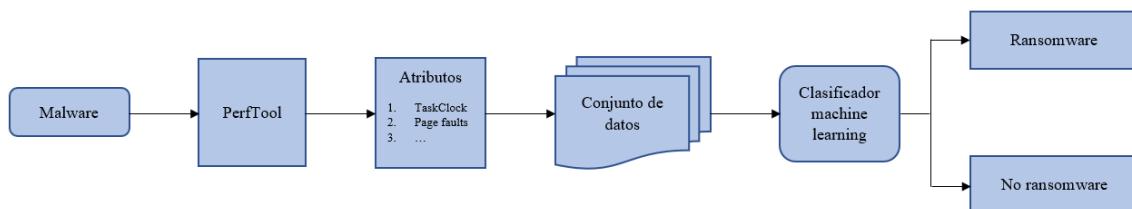


Figura 4.11: Flujo del proceso de entrenamiento y prueba.

A continuación se muestra una Tabla 4.2 comparativa de los trabajos previamente mencionados que usan la extracción de APIs para la detección de malware y ransomware, exponiendo de manera clara qué algoritmos de aprendizaje automático utilizan, qué datos usan y qué resultados obtienen:

Tabla 4.2: Tabla comparativa de los trabajos que usan aprendizaje automático para detectar malware mediante la extracción de APIs.

Trabajo	Algoritmos	Conjuntos de datos	Resultados
[RTWH11] (2011)	- Hierarchical clustering (agrupamiento jerárquico) - Nearest prototype classifier	- 3.133 muestras de clases conocidas de malware de CWSandbox - 33.698 muestras de clases desconocidas de malware de Sunbelt Software	- Precisión: 99,6 % - Valor-F: 95 %.
[AVWA11] (2011)	- NB - KNN - DT - SVM	- 51.223 muestras de malware del proyecto Honeynet y VX Heaven - 15.480 muestras de goodware	- DT: TPR 93 %, FPR 6,8 %, Precisión 93,1 %, Valor-F 93 %, Area ROC 93,1 % - KNN: TPR 94,8 %, FPR 5,1 %, Precisión 94,8 %, Valor-F 94,8 %, Area ROC 96,6 % - NB: TPR 91 %, FPR 9 %, Precisión 91 %, Valor-F 91 %, Area ROC 93,8 % - SVM PolyKernel normalizado: TPR 98,6 %, FPR 2,5 %, Precisión 97,8 %, Valor-F 98,6 %, Area ROC 98,2 % - SVM PolyKernel: TPR 93,4 %, FPR 6,9 %, Precisión 93,6 %, Valor-F 93,4 %, Area ROC 93,2 % - SVM Puk: TPR 94 %, FPR 6,4 %, Precisión 94 %, Valor-F 93,2 %, Area ROC 93,9 % - SVM RBF: TPR 92,9 %, FPR 7,3 %, Precisión 92,9 %, Valor-F 92,9 %, Area ROC 92,8 %
[RM12] (2012)	- NB - SVM - DT - Algoritmo clasificador propio de 4-gramas	- 179 muestras de malware de VXHeavens - 94 muestras de goodware de una instalación de Windows XP	Precisión: - NB: 48,69 % - SVM: 64,43 % - DT: 56,77 % - Algoritmo propuesto: 90 %
[SMGML16] (2016)	- NB - SVM - EldeRan	- 582 muestras de ransomware de 11 familias de VirusShare - 942 muestras de goodware de Software Informer	- EldeRan: AUC 99,49 %, FPR 1,61 %, TPR 96,34 % - SVM: AUC 99,29 %, FPR 1,99 %, TPR 92,19 % - NB: AUC 96,96 %, FPR 9,58 %, TPR 94,53 %

Trabajo	Algoritmos	Conjuntos de datos	Resultados
[VSVG17] (2017)	- MLP - LR - NB - DT - RF - KNN - SVM	- 755 muestras de ransomware de varias fuentes - 219 muestras de goodware	- MLP : Precisión 100 %, Exactitud 100 %, Exhaustividad 100 %, Valor-F 100 % - LR : Precisión 98,8 %, Exactitud 98,5 %, Exhaustividad 100 %, Valor-F 99,3 % - NB : Precisión 97,2 %, Exactitud 96,6 %, Exhaustividad 100 %, Valor-F 98,3 % - DT : Precisión 96,4 %, Exactitud 95,7 %, Exhaustividad 100 %, Valor-F 97,8 % - RF : Precisión 98,4 %, Exactitud 98 %, Exhaustividad 100 %, Valor-F 99 % - KNN : Precisión 96,8 %, Exactitud 96,2 %, Exhaustividad 100 %, Valor-F 99,3 % - SVM : Precisión 98,8 %, Exactitud 98,5 %, Exhaustividad 100 %, Valor-F 99,3 %
[CKYK17] (2017)	- NB - SVM - RF - LR	- 83 muestras de ransomware diferentes de VirusShare - 85 muestras de goodware de Software Informer	- LR : Precisión: 98,2 %, TPR 97,6 %, FPR 1,2 % - SVM : Precisión: 95,8 %, TPR 96,4 %, FPR 4,7 % - RF : Precisión: 95,8 %, TPR 96,4 %, FPR 4,7 % - NB : Precisión: 92,3 %, TPR 94 %, FPR 9,4 %
[TSF18] (2018)	- SVM	- 276 muestras de ransomware - 312 muestras de goodware	- Precisión: 97,48 % - Tasa de error: 1,64 %
[BLI19] (2019)	- RF - LR - NB - SGD - KNN - SVM	- 1000 muestras de ransomware - 900 muestras de malware - 300 muestras de goodware	- RF : Precisión 98,39 %, Exhaustividad 98,55 %, Valor-F 98,56 %, Exactitud 98,58 % - KNN : Precisión 96,13 %, Exhaustividad 96,13 %, Valor-F 96,25 %, Exactitud 95,50 % - NB : Precisión 87,43 %, Exhaustividad 87,92 %, Valor-F 82,72 %, Exactitud 89,38 % - SGD : Precisión 93,96 %, Exhaustividad 90,95 %, Valor-F 90,22 %, Exactitud 91,61 % - LR : Precisión 90,08 %, Exhaustividad 93,99 %, Valor-F 92,98 %, Exactitud 92,07 % - SVM : Precisión 74,81 %, Exhaustividad 78,99 %, Valor-F 75,98 %, Exactitud 79,95 %
[KAJS19] (2019)	- PEDA - RF - NB - Ensamble (RF+NB)	- 582 muestras de ransomware - 942 muestras de goodware Los dos conjuntos de muestras fueron extraídos por el RISS	- PEDA : AUC 99,30 %, FPR 1,56 %, Precisión 97,05 %, TPR 95 % - RF : AUC 95 %, FPR 7 %, Precisión 92 %, TPR 91 % - NB : AUC 81 %, FPR 15 %, Precisión 83 %, TPR 79 % - Ensamble: AUC 82,5 %, FPR 16 %, Precisión 88 %, TPR 98 %

Trabajo	Algoritmos	Conjuntos de datos	Resultados
[Jet19] (2019)	- SVM - RF - LR	Inicialmente: - 666 muestras de ransomware - 103 muestras de goodware Para balancear los datos, usan SMOTE y obtienen 1.072 muestras de cada clase	- SVM : Precisión 98 %, Exhaustividad 97 %, Valor-F 97 % - RF : Precisión 97 %, Exhaustividad 97 %, Valor-F 97 % - LR : Precisión 97 %, Exhaustividad 97 %, Valor-F 97 %
[HKLK20b] (2020)	- RF	- 1909 muestras de ransomware de VirusShare - 1139 muestras de goodware de Softonic	- Precisión: 97,3 % - FPR : 4,8 % - FNR : 1,5 % - Valor-F: 97,0 %
[KAJ20] (2020)	- PEDA (RF discretizado) - RF	Tres datasets: - Dataset PE (con ransomware criptográfico): 942 muestras de goodware 205 muestras de cripto-ransomware - Dataset Old (creada en 2016): 942 muestras de goodware 582 muestras de ransomware - Dataset Full (PE + Old + nuevas muestras de VirusShare y theZoo): 942 muestras de goodware 904 muestras de ransomware Tras obtener dicho dataset y realizar mediante el algoritmo de este estudio una limpieza de los datos para eliminar repetidos, el dataset restante obtenido es de 912 muestras	- Dataset PE: PEDA : TPR 99,8 %, Precisión 99,9 %, Valor-F 99,9 %, ROC 99,9 % RF : TPR 99,6 %, Precisión 99,6 %, Valor-F 99,4 %, ROC 99,5 % - Dataset Old: PEDA : TPR 95,6 %, Precisión 95,8 %, Valor-F 95,6 %, ROC 98,5 % RF : TPR 95,2 %, Precisión 95,2 %, Valor-F 95,2 %, ROC 97,0 % - Dataset Full: PEDA : TPR 99,6 %, Precisión 99,6 %, Valor-F 99,4 %, ROC 99,6 % RF : TPR 99,7 %, Precisión 99,7 %, Valor-F 99,5 %, ROC 99,5 %
[ADPC20b] (2020)	- Neural Networks (NN) - KNN - SVM lineal - SVM RBF - Gauss Process (GP) - DT - RF - AdaBoost - Qualitative Data Analysis (QDA) - NB	- 41 aplicaciones benignas - 34 procesos malware, incluidos 7 muestras de ransomware	Precisión: - NN : 52,85 % - KNN : 57,06 % - SVM lineal: 63,05 % - SVM RBF : 60,50 % - GP : 59,8 % - DT : 70,14 % - RF : 75,01 % - AdaBoost: 61,64 % - QDA : 62,7 % - NB : 58,55 %

Trabajo	Algoritmos	Conjuntos de datos	Resultados
[ARA ⁺ 21] (2021)	<ul style="list-style-type: none"> - DT - RF - Gradient Boosting - Extreme Gradient Boosting 	160 muestras de las cuales pertenecen 80 a ransomware y 80 a otros tipos de malware	<p>Precisión:</p> <ul style="list-style-type: none"> - DT: 94,5 % - RF: 97 % - Gradient Boosting: 94,5 % - Extreme Gradient Boosting: 97 % <p>Valor F:</p> <ul style="list-style-type: none"> - DT: 94 % - RF: 97 % - Gradient Boosting: 94 % - Extreme Gradient Boosting: 97 % <p>TPR:</p> <ul style="list-style-type: none"> - DT: 88 % - RF: 94 % - Gradient Boosting: 88 % - Extreme Gradient Boosting: 94 %

Capítulo 5

Técnica de Detección de Ransomware Propuesta

El capítulo incluye información relacionada con las tecnologías utilizadas en el proyecto, el entorno de trabajo y los datos. En la Sección 5.1 se describe el concepto de [API](#), una de las fuentes de datos en crudo y la estructura del modelo. A continuación, la Sección 5.2 está destinada al montaje del laboratorio de análisis de muestras *ransomware*. La Sección 5.3 incluye el proceso de construcción del conjunto de datos final, incluyendo información de la estructura de los reportes obtenidos en el análisis y las pautas seguidas para la limpieza y la extracción de características. Por último, en la Sección 5.4 se detallan el modelo y los algoritmos utilizados, así como librerías importantes para la construcción del mismo.

5.1. Descripción del Sistema Propuesto

El objetivo del sistema propuesto es la detección de ransomware mediante la extracción de las llamadas a la [API](#) de Windows, utilizando algoritmos supervisados de aprendizaje automático para construir un modelo de clasificación que detecte si un archivo es ransomware o no.

Una [API](#) (interfaz de programación de aplicaciones) proporciona una abstracción para un problema y especifica cómo los clientes deben interactuar con los componentes software para implementar una solución. Suelen ser distribuidas como librerías software, definiendo bloques de desarrollo reusables que permiten añadir funcionalidades en múltiples aplicaciones [Red11].

Microsoft suministra a los desarrolladores de Windows una [API](#) documentada profesionalmente para facilitar el proceso de creación de una aplicación. Ésta provee todas las funcionalidades básicas y permite el despliegue de entornos ricos e interactivos. De manera análoga, una aplicación de Windows puede ser mapeada y explicada por las llamadas a la API que realice, entendiendo así su comportamiento. Por ejemplo, si el proceso llama a la función *WriteFile*, se puede saber que se va a escribir en un fichero del sistema. Si se invoca, por ejemplo a *CryptEncrypt*, la aplicación va a efectuar un cifrado en nuestro sistema de ficheros. Este tipo de llamadas pueden ser muy útiles para la detección de ransomware de cifrado, aunque aplicaciones benignas también lo utilicen. Para este estudio, se han recogido un total de 302 llamadas distintas a la [API](#) de Windows, realizando una distinción en la recogida de datos: se conforma un *dataset* en el que, de manera binaria, un proceso a invocado a la función de sistema correspondiente, y otro que señala la cantidad de llamadas totales para cada función [AHSF09b].

Los datos utilizados en el proyecto se han obtenido de dos formas diferentes. Primero,

el grupo de investigación GASS [Gru21] proporcionó 21 muestras de *ransomware* para su análisis en un entorno controlado, como se explica en la Sección 5.2.1. Por otro lado, el grupo GASS [Gru21] ha desarrollado una herramienta con la que se accede a un número elevado de muestras, extrayendo múltiples informes obtenidos de análisis realizados en la página web de Cuckoo Sandbox por usuarios anónimos. De esta forma se obtiene un fichero en formato .json con información de 5435 muestras de *ransomware* y otro fichero con información de 10896 muestras de *goodware* (software benigno). Combinando ambos métodos, es posible componer un conjunto de datos inicial de gran extensión y mayor variedad. El análisis de las muestras y la extracción de las características relevantes se explica con más detalle en la Sección 5.3. Finalmente, los resultados obtenidos del análisis se usarán para entrenar al modelo de ML. La Figura 5.1 muestra el proceso en términos generales.

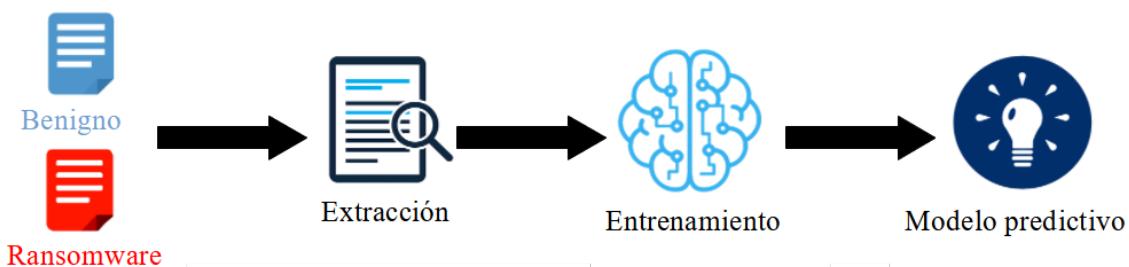


Figura 5.1: Extracción de características y creación del modelo predictivo.

A partir del entrenamiento se creará un modelo predictivo que podrá decir si archivo es ransomware o benigno. En la fase de predicción se hace uso del modelo predictivo para sacar inferencias, y este proceso es similar al de entrenamiento, ya que ambos comienzan extrayendo características. Sin embargo, esta vez en lugar de entrenar el modelo usando esas características, se predecirá la naturaleza del archivo desconocido. El desarrollo del modelo y los algoritmos de aprendizaje automático usados se detallan en la Sección 3.5.1. La Figura 5.2 muestra este proceso.

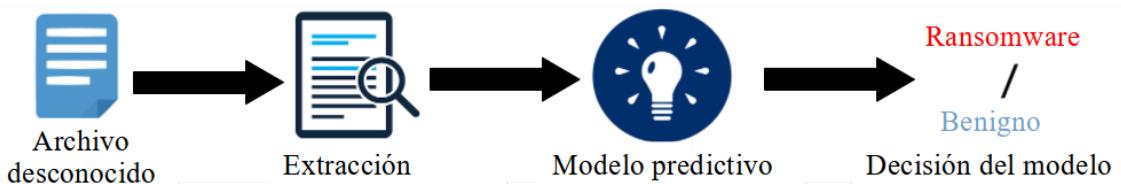


Figura 5.2: Clasificación de un archivo con un modelo predictivo.

La Figura 5.3 representa el diagrama de flujo del sistema propuesto en este trabajo, siguiendo los pasos detallados a continuación:

1. Obtención de muestras ransomware y benignas e información de reportes.
2. Ejecución y análisis de las muestras en un entorno controlado usando Cuckoo Sandbox.
3. Unión y limpieza de los datos, extracción de características y *dataset*.
4. División del *dataset* en datos de entrenamiento y datos de prueba.

5. Entrenamiento del modelo.
6. Utilización del modelo.
7. Salida del modelo y clasificación de la muestra.
8. Análisis, generación de gráficas y conclusiones.

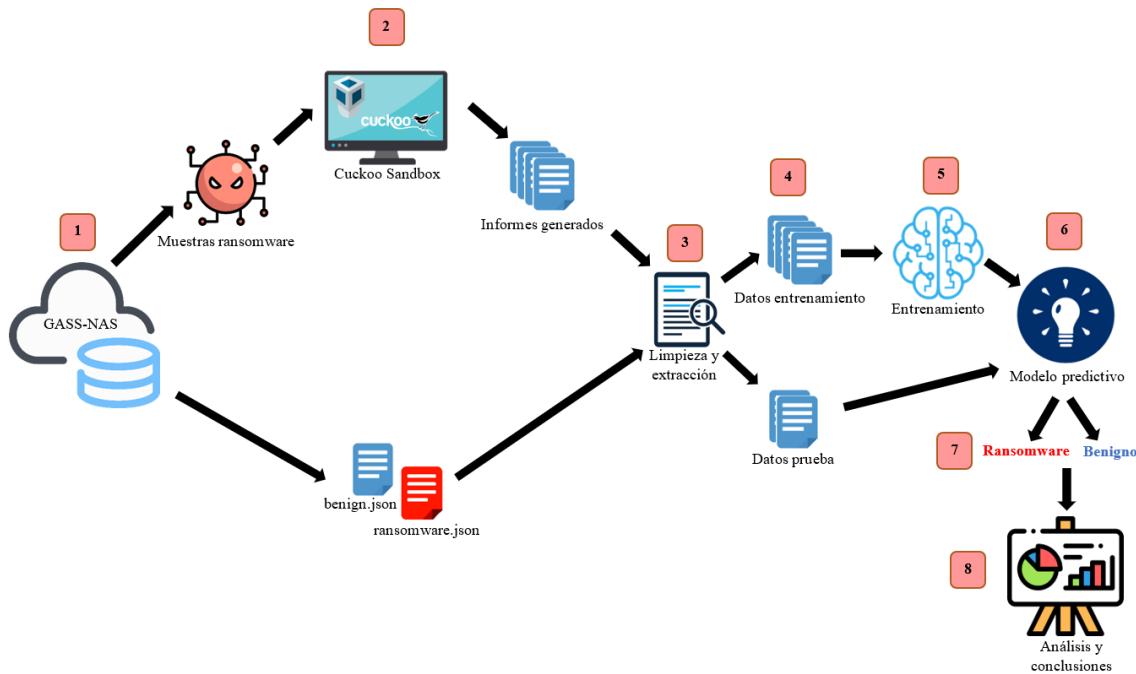


Figura 5.3: Diagrama de flujo del sistema propuesto.

5.2. Entorno de Obtención de Datos

A la hora de ejecutar muestras reales de *ransomware* para su análisis, se puede causar daño al sistema con borrado de archivos, cambios en los registros y robo de información confidencial, entre otras cosas. Para evitar esto, es preciso contar con un entorno seguro donde ejecutar el malware sin poner en riesgo los equipos o la red [DO13].

Cuckoo Sandbox es una herramienta de análisis automatizado de malware de código abierto en Windows, macOS, Linux y Android. Esta herramienta utiliza la tecnología de *sandboxing*, que, en terminología informática, es una técnica que consiste en aislar la ejecución de un programa no fiable o malicioso llevándola a un entorno seguro que no comprometa el sistema, de forma que se puedan analizar las actividades del malware sin preocupación por los cambios que realice el proceso maligno.

Existen múltiples herramientas que utilicen *sandboxing* y permitan construir un laboratorio de análisis de malware, entre ellas, Buster Sandbox Analyzer, Zero Wine o Malheur, pero en este trabajo se recurre a Cuckoo Sandbox debido a que es la más completa y, siendo de código abierto y teniendo un diseño modular, permite personalizar cualquier aspecto del procesamiento del análisis de los resultados, de la generación de informes y del entorno de análisis. Además, tiene una guía detallada de los requisitos para integrar la herramienta con el sistema.

Cuckoo Sandbox, se inició como un proyecto de verano parte del programa Google Summer of Code en 2010, lanzando la primera beta en febrero de 2011. El diseñador y desarrollador de Cuckoo, Claudio Guarnieri, es aún el principal coordinador del proyecto. En marzo de 2012, Cuckoo Sandbox ganó la primera ronda del programa Magnificent7 organizado por Rapid7 (un importante sistema de Gestión de Vulnerabilidades y análisis de *endpoints*, con más de 20 años de trayectoria). Esta herramienta fue mejorando y obteniendo méritos hasta febrero de 2016, cuando la versión 2.0 fue publicada, siendo esta una versión muy completa y adaptada al uso diario.

La arquitectura de Cuckoo se muestra en la Figura 5.4. Consiste en un software central que maneja la ejecución y el análisis de muestras, en el caso de este proyecto, ejecutando cada una en una máquina virtual aislada. De esta forma, los componentes principales de la infraestructura del laboratorio de análisis de malware (y por lo tanto de Cuckoo Sandbox) son una máquina anfitrión que gestiona el proceso de análisis, la cual tendrá acceso a Internet, y un número de máquinas virtuales conectadas a una red virtual donde las muestras de malware se ejecutan de forma segura y aislada para ser analizadas.

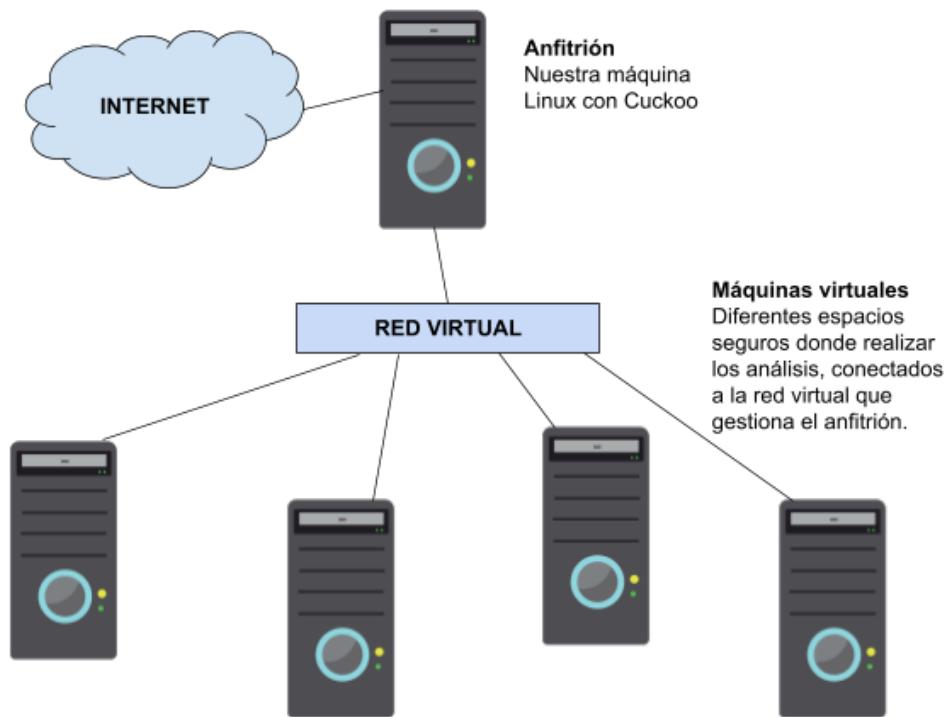


Figura 5.4: Arquitectura de Cuckoo Sandbox

Hay otra manera de utilizar Cuckoo Sandbox sin necesidad de una máquina virtual propia, y es mediante el uso de la interfaz web de la herramienta ([Cuckoo Sandbox Online](#)). La ventaja de este método es que no es necesario construir un laboratorio de análisis con Cuckoo descargado, ya que es accesible mediante Internet. Pero tiene un gran inconveniente, y es la lentitud del análisis, ya que es necesario subir los archivos manualmente uno por uno y si la página web tiene mucha afluencia, el análisis tardará más. Por esta razón se decidió construir un laboratorio de análisis propio en una máquina con el sistema operativo recomendado por los autores de Cuckoo Sandbox, que es Linux con distribución Ubuntu 18.04.5 [Long Term Support \(LTS\)](#).

5.2.1. Laboratorio de análisis

Una vez entendido el concepto de *sandboxing*, y la necesidad de construir un laboratorio de análisis de malware para el trabajo, comienza la preparación del mismo.

La máquina anfitrión será un ordenador con sistema operativo Linux, utilizando la distribución Ubuntu en la versión 18.04.5 LTS.

Se necesitan una serie de requisitos para la instalación y el correcto funcionamiento de Cuckoo.

Será necesario tener la versión adecuada de Python (actualmente soporta Python 2.7) e instalar una serie de bibliotecas:

```
$ sudo apt-get install python python-pip python-dev libffi-dev libssl-dev
$ sudo apt-get install python-virtualenv python-setuptools
$ sudo apt-get install libjpeg-dev zlib1g-dev swig
```

Para poder utilizar la interfaz web Django y PostgreSQL como base de datos, se necesita MongoDB y PostgreSQL:

```
$ sudo apt-get install mongodb
$ sudo apt-get install postgresql libpq-dev
```

Las máquinas virtuales basadas en Windows 7 donde se ejecutarán las muestras de malware serán generadas a través de VirtualBox, un software de virtualización muy completo el cual está soportado por Cuckoo, aunque se puede utilizar otro software de virtualización. En este caso se utiliza la versión 5.2.42 de VirtualBox, la cual se puede descargar e instalar desde la página oficial o utilizando los comandos siguientes:

```
$ echo deb http://download.virtualbox.org/virtualbox/debian xenial contrib | sudo \
tee -a /etc/apt/sources.list.d/virtualbox.list
$ wget -q https://www.virtualbox.org/download/oracle_vbox_2016.asc -O- | sudo \
apt-key add -
$ sudo apt-get update
$ sudo apt-get install virtualbox-5.2
```

Para poder obtener información de la actividad de red del malware se recurre a tcpdump, que rastrearía la red para capturar el tráfico a la hora de analizar una muestra.

```
$ sudo apt-get install tcpdump apparmor-utils
```

Esta herramienta necesita privilegios de superusuario, los cuales les serán otorgados únicamente a ella y no a Cuckoo Sandbox:

```
$ sudo groupadd pcap
$ sudo usermod -a -G pcap (nombre_usuario)
$ sudo chgrp pcap /usr/sbin/tcpdump
$ sudo setcap cap_net_raw,cap_net_admin=eip /usr/sbin/tcpdump
$ sudo aa-disable /usr/sbin/tcpdump
```

Tras esto, se instala M2crypto, un módulo de Python que será necesario para el análisis de muestras:

```
$ sudo pip install m2crypto
```

En este caso, se utiliza VirtualBox y será necesario incluir al usuario con el que se ejecuta Cuckoo en el grupo “vboxusers”, para que Cuckoo pueda identificar las máquinas

virtuales:

```
$ sudo usermod -a -G vboxusers (nombre_usuario)
```

A continuación, se crea un entorno virtual, para añadir otro factor más de seguridad al laboratorio y en este entorno será donde posteriormente se instale Cuckoo Sandbox y se trabaje. Para ello, descargamos un [script bash de GitHub](#) el cual sirve para instalar esta funcionalidad:

```
$ sudo -u <USERNAME> cuckoo-setup-virtualenv.sh
```

Una vez instalada esta herramienta, se crea un entorno virtual, en el caso de este trabajo se le ha dado el nombre de “cuckoo-test” y llegados a este punto, siempre se trabajará dentro de este entorno, por lo que será muy importante asegurarse de que estamos dentro del entorno virtual al utilizar un terminal.

```
$ mkvirtualenv -p python2.7 cuckoo-test
```

Tras la ejecución del comando anterior, se estará trabajando dentro de un entorno virtual donde se realizan el resto de pasos. Primero se requiere la actualización de los módulos pip y setuptools y después, se procede a la instalación de Cuckoo:

```
(cuckoo-test) $ pip install -U pip setuptools
(cuckoo-test) $ pip install -U cuckoo
```

A continuación, se deberá descargar una ISO de Windows 7 y se monta: [\[AP19\]](#)

```
(cuckoo-test) $ wget https://cuckoo.sh/win7ultimate.iso
(cuckoo-test) $ mkdir /mnt/win7
(cuckoo-test) $ sudo mount -o ro,loop win7ultimate.iso /mnt/win7
```

Para gestionar la creación de máquinas virtuales para Cuckoo, el software que usan y la captura de estados, el uso de la herramienta VMCloak facilitará estas tareas ya que está diseñada para crear máquinas virtuales que Cuckoo Sandbox pueda utilizar. [\[Bre15\]](#) Es necesario instalar algunos paquetes previos para la correcta integración con Cuckoo, y después instalar VMCloak: [\[AP19\]](#)

```
(cuckoo-test) $ sudo apt-get -y install build-essential libssl-dev libffi-dev \
python-dev genisoimage
(cuckoo-test) $ sudo apt-get -y install zlib1g-dev libjpeg-dev
(cuckoo-test) $ sudo apt-get -y install python-pip python-virtualenv \
python-setuptools swig
(cuckoo-test) $ pip install -U vmcloak
```

Una vez la herramienta está instalada, se crea una interfaz de red a la que se conectarán las máquinas virtuales, se configura la ISO montada previamente con las características adecuadas (en este caso, serán máquinas virtuales con 2 CPUs y una memoria RAM de 2048MB) utilizando los siguientes comandos: [\[AP19\]](#)

```
(cuckoo-test) $ vmcloak-vboxnet0
(cuckoo-test) $ vmcloak init --verbose --win7x64 win7x64base --cpus 2 --ramsize 2048
```

El siguiente paso es instalar el software necesario, que una vez creados los estados de la imagen, no se podrá modificar, por eso, el primer paso será clonar la máquina en su

versión original vacía, para instalar software ahí y guardar los estados. Entre el software que necesario para las máquinas encontramos java, adobepdf, flash y pillow (este último será el paquete que realice capturas de pantalla durante el análisis del malware): [AP19]

```
(cuckoo-test) $ vmcloak clone win7x64base cuckooVM
(cuckoo-test) $ vmcloak install cuckooVM adobepdf dotnet java flash vcredist \
vcredist.version=2015u3 wallpaper
(cuckoo-test) $ vmcloak install cuckooVM pillow
```

Llegados a este punto, se crean los estados de la máquina virtual. Dadas las características del equipo en el que se está construyendo el laboratorio de análisis de malware, se crean 3 estados, lo que permitirá realizar un máximo de 3 análisis simultáneos. Con el siguiente comando, se crearán 3 máquinas virtuales con IPs 192.168.56.101, 192.168.56.102 y 192.168.56.103: [AP19]

```
(cuckoo-test) $ vmcloak snapshot --count 3 cuckooVM 192.168.56.101
```

Con todo lo realizado hasta ahora, el laboratorio está casi preparado y únicamente quedaría configurar Cuckoo Sandbox para ponerlo a funcionar. Para ello es necesario iniciar cuckoo, que mostrará por defecto el directorio de trabajo “/home/nombre_usuario/.cuckoo”, este directorio se podrá modificar sin problema, aunque en el caso de este proyecto, se trabajará en el definido por defecto.

```
(cuckoo-test) $ cuckoo init
(cuckoo-test) $ cuckoo community
```

También será necesario modificar los ficheros de configuración de Cuckoo, que se encuentran en la carpeta “conf” dentro del directorio principal de trabajo, personalizando los ajustes y configurando las máquinas virtuales.

En primer lugar, la edición del fichero “virtualbox.conf” que contiene opciones sobre el software de virtualización, en este caso, se edita el campo “mode” que por defecto tendrá el valor “headless” y pone el valor “gui”. Este cambio permitirá ver la máquina virtual en ejecución cuando se realice un análisis, si por el contrario se quiere ver únicamente el resultado final y no lo que va sucediendo, se mantendrá el valor “headless”. El fichero quedaría de la siguiente manera:

Listing 5.1: Fichero “virtualbox.conf”

```
1 [virtualbox]
2 # Specify which VirtualBox mode you want to run your machines on.
3 # Can be "gui" or "headless". Please refer to VirtualBox's official
4 # documentation to understand the differences.
5 mode = gui
```

Con el siguiente comando, que modificará el fichero anterior, se añaden las máquinas virtuales creadas a Cuckoo con su configuración. [AP19]

```
(cuckoo-test) $ while read -r vm ip; do cuckoo machine --add $vm $ip;
done < <(vmcloak list vms)
```

El resultado del fichero será el siguiente:

Listing 5.2: Fichero “virtualbox.conf”

```

1 [192.168.56.1011]
2 # Specify the label name of the current machine as specified in your
3 # VirtualBox configuration.
4 label = 192.168.56.1011
5
6 # Specify the operating system platform used by current machine
7 # [windows/darwin/linux].
8 platform = windows
9
10 # Specify the IP address of the current virtual machine. Make sure that the
11 # IP address is valid and that the host machine is able to reach it. If not,
12 # the analysis will fail.
13 ip = 192.168.56.101
14
15 # (Optional) Specify the snapshot name to use. If you do not specify a
16 # snapshot name, the VirtualBox MachineManager will use the current snapshot
17 # Example (Snapshot1 is the snapshot name):
18 snapshot =
19
20 # (Optional) Specify the name of the network interface that should be used
21 # when dumping network traffic from this machine with tcpdump. If specified,
22 # overrides the default interface specified in auxiliary.conf
23 # Example (vboxnet0 is the interface name):
24 interface =
25
26 # (Optional) Specify the IP of the Result Server, as your virtual machine
27 # sees it.
28 # The Result Server will always bind to the address and port specified in
29 # cuckoo.conf, however you could set up your virtual network to use NAT/PAT,
30 # so you can specify here the IP address for the Result Server as your
31 # machine sees it. If you don't specify an address here, the machine will
32 # use the default value from cuckoo.conf.
33 # NOTE: if you set this option you have to set result server IP to 0.0.0.0
34 # in cuckoo.conf.
35 # Example:
36 resultserver_ip = 192.168.56.1
37 # (Optional) Specify the port for the Result Server, as your virtual machine
38 # sees it. The Result Server will always bind to the address and port
39 # specified in cuckoo.conf, however you could set up your virtual network
40 # to use NAT/PAT, so you can specify here the port for the Result Server as
41 # your machine sees it. If you don't specify a port here, the machine will
42 # use the default value from cuckoo.conf.
43 # Example:
44 resultserver_port = 0
45 # (Optional) Set your own tags. These are comma separated and help to
46 # identify specific VMs. You can run samples on VMs with tag you require.
47 tags =
48 # Mostly unused for now. Please don't fill it out.
49 options =
50 # (Optional) Specify the OS profile to be used by volatility for this
51 # virtual machine. This will override the guest_profile variable in
52 # memory.conf which solves the problem of having multiple types of VMs
53 # and properly determining which profile to use.
54 osprofile =
55
56 [192.168.56.1012]
57 .
58 .
59 .

```

Hecho esto, se quiere dar acceso a Internet a las máquinas de análisis y para ello, es necesario activar la redirección de paquetes (*forwarding*) en la interfaz creada para VirtualBox y en la interfaz de salida (que en este caso es “wlo1”, pero puede ser “eth0” u otra, dependiendo del sistema).

La conexión a Internet es importante para que las muestras de malware muestren su comportamiento completo en los análisis, ya que que se conecten a la red es algo básico en la mayoría de ellas [AP19]:

```
(cuckoo-test) $ sudo sysctl -w net.ipv4.conf.vboxnet0.forwarding=1
(cuckoo-test) $ sudo sysctl -w net.ipv4.conf.wlo1.forwarding=1
```

Otro aspecto a configurar de Cuckoo, es el módulo “rooter”, que concede a Cuckoo ciertos permisos para trabajar con comandos de red y poder realizar análisis con opciones de encaminamiento y así sacar el máximo partido a análisis:

```
(cuckoo-test) $ cuckoo rooter --sudo --group gonzalo
```

Por último, antes de poder iniciar la interfaz web de Cuckoo Sandbox y empezar a realizar análisis, los últimos ficheros que necesitan edición son el fichero “*routing.conf*”, cambiando el apartado “*internet*” con valor “*none*” y dándole el nuevo valor “*wlo1*” (la interfaz de red) y el fichero “*reporting.conf*” cambiando el valor de “*enabled*” en el apartado “*mongodb*” de “*no*” a “*yes*” [AP19].

Listing 5.3: Fichero “*routing.conf*”

```
1 [routing]
2 # Network interface that allows a VM to connect to the entire internet , the
3 # "dirty line" so to say. Note that , just like with the VPNs, this will allow
4 # malicious traffic through your network. So think twice before enabling it .
5 # (For example , to use eth0 as dirty line: "internet = eth0") .
6 internet = wlo1
```

Listing 5.4: Fichero “*reporting.conf*”

```
1 [mongodb]
2 enabled = yes
3 host = 127.0.0.1
4 port = 27017
5 db = cuckoo
6 store_memdump = yes
7 paginate = 100
8 # MongoDB authentication (optional) .
9 username =
10 password =
```

Con el laboratorio puesto a punto, ya es posible realizar el análisis de las muestras para obtener parte de los datos que serán procesados para alimentar el modelo de ML. El análisis de una muestra usando Cuckoo Sandbox se detalla en el Anexo A.

5.3. Conformación del *Dataset*

En esta sección se explica cómo se ha formado el *dataset* a partir de los datos obtenidos del entorno de laboratorio con Cuckoo Sandbox instalado, descrito en la Sección 5.2.

5.3.1. Estructura de reportes

Cuando se realiza el análisis de muestras de ransomware, los resultados obtenidos y los informes generados se guardan en carpetas nombradas con un identificador generado automáticamente por Cuckoo Sandbox en el directorio “/storage/analyses/” dentro del directorio principal de trabajo de la herramienta. La estructura de la información generada sigue aproximadamente el siguiente esquema:

```
.
|-- analysis.log
|-- binary
|-- dump.pcap
|-- memory.dmp
|-- files
|   |-- 1234567890_dropped.exe
|-- logs
|   |-- 1232.bson
|   |-- 1540.bson
|   '-- 1118.bson
|-- reports
|   |-- report.html
|   '-- report.json
`-- shots
    |-- 0001.jpg
    |-- 0002.jpg
    |-- 0003.jpg
    '-- 0004.jpg
```

De todos los archivos que genera Cuckoo Sandbox, se hace hincapié en el fichero “**report.json**”, que contiene todo lo generado por el análisis dividido por secciones según la información que contengan, tal y como se muestra en el siguiente esquema [RG18]:

```
report
|-- info
|-- signatures
|-- target
|-- shots
|-- static
|-- dropped
|-- behavior
|-- debug
|-- metadata
|-- screenshots
|-- strings
|-- network
`-- virustotal
```

De todas las secciones anteriores, se presta especial atención a “**behavior**”, que muestra todo lo relacionado con los procesos ejecutados por el malware durante su actuación, el orden de los procesos, el comportamiento, las interacciones con el sistema, las llamadas a APIs de Windows, etc., y a “**signatures**”, que contiene las firmas que han dado un resultado positivo en el análisis, y sirven para identificar patrones o comportamientos concretos personalizados ya que Cuckoo Sandbox ofrece la posibilidad de crear firmas u obtener firmas creadas por otros usuarios de la comunidad [RG18] [Fou].

5.3.2. Limpieza y Extracción de Características

En los ficheros proporcionados por el Grupo de Investigación [Gru21] con resultados de análisis de muestras de *ransomware* y muestras de *goodware* se encuentra información variada como la puntuación del análisis, una lista APIs o información de red (*pcaps*),

entre otros. Se realiza una limpieza de los datos, dejando únicamente la lista de APIs y la lista de *signatures*, que será la información necesaria para componer el *dataset* final.

En cuanto a los informes generados por los análisis de muestras realizados en el laboratorio construido en este proyecto, se hará uso del fichero “`report.json`” para obtener tanto las llamadas a APIs de Windows, que se encuentran en la sección “`apistats`” dentro de “`behavior`”, donde para cada proceso de la ejecución, referido por su *Process ID (PID)*, se muestran las APIs llamadas por cada uno y el número de veces que se realizan estas llamadas, como los nombres de las firmas (*signatures*) del análisis, que se encuentran en la sección “`signatures`”.

Mediante un *script* en Python, se analizan todos los informes de una sola vez, tanto los incluidos en el fichero de extensión `.json` en el cual se ha ejecutado la limpieza anteriormente mencionada, como los generados con el análisis de muestras, accediendo a la sección correspondiente para cada uno y recorriendo las llamadas a APIs, obteniendo todas las diferentes y generando un fichero `CSV` con la información.

Algoritmo 5.5: Pseudocódigo de análisis de informes ransomware

```

1      Abrir y leer fichero
2      Inicializar diccionario vacío
3
4      Para cada muestra dentro del fichero json
5          extraer lista de apis
6          para cada api de la lista
7              si la api no se encuentra en el diccionario entonces
8                  incluir api al diccionario
9
10     Crear fichero csv
11     Escribir cabecera
12
13     Para cada muestra dentro del fichero
14         asignar diccionario vacío
15         inicializar lista de llamadas vacía
16         extraer lista de apis
17         para cada api de la lista
18             si la api se encuentra en el diccionario vacío
19                 asignar cantidad de llamadas
20
21     escribir lista en csv

```

En cuanto a la representación del fichero de extensión `CSV` generado en el análisis de los reportes, la primera de las columnas se utiliza para identificar cada informe mediante el listado de las *signatures* del informe generado, la siguiente, determinará si se trata de una muestra ransomware o una muestra benigna y, el resto de columnas se corresponden con las 302 diferentes APIs únicas que se han obtenido en el análisis de los informes recogidos. En cuanto a las filas, cada una de ellas corresponde a una muestra representando de manera binaria en sus celdas, si la API correspondiente ha sido llamada durante el proceso de ejecución o no.

Una vez construido el fichero `CSV`, compuesto por 15352 filas, de las cuales 5456 corresponden a muestras *ransomware* y 10896 a muestras *goodware*, se procede a una limpieza de datos en dos pasos. En el primer paso se eliminan las filas cuyo identificador (compuesto por la lista de firmas (*signatures*)) se encuentre repetido, eliminando las muestras que realicen el mismo patrón de firmas y obteniendo 5416 filas *ransomware* y 9648 filas *goodware*. En el segundo paso, considerando que las muestras que hacen uso de las mismas funciones tienen un comportamiento y objetivos similares, se eliminan las filas idénticas en cuanto a llamadas a APIs, buscando la mayor heterogeneidad posible

del conjunto de datos y evitando el *overfitting* (el sobreentrenamiento de los datos, lo que refuerza comportamientos concretos del modelo). Con esta limpieza, el resultado obtenido es de 1596 filas *ransomware* y 357 filas *goodware*.

Tras la limpieza, el número de filas *ransomware* supera ampliamente al resto, por lo que para equilibrar el *dataset*, se escogen aleatoriamente 357 filas de este tipo de muestras para que no existan sesgos en los datos por la presencia predominante de un tipo de muestras. Realizado este proceso, el *dataset* que se utilizará para alimentar el modelo de [ML](#) estará compuesto de 714 filas, correspondiendo la mitad a muestras *ransomware* y la otra mitad a muestras *goodware* y teniendo la misma representación detallada anteriormente, con la excepción de que existirá una primera columna con un número entero identificador único que es asignado a cada fila durante la limpieza de los datos. Este *dataset* se denomina *dataset binario*.

Debido a la gran reducción de datos ocasionada por la limpieza y a la intención de ampliar la investigación, se decide formar un segundo *dataset* siguiendo las mismas pautas y empleando las mismas técnicas aplicadas anteriormente con la diferencia de que esta vez, además de reflejar las llamadas por cada muestra a las diferentes [APIs](#), se detallará el número de veces que éstas han sido invocadas, por lo que ya no es necesario eliminar las filas idénticas en cuanto a llamadas a [APIs](#).

Siguiendo estos pasos, el segundo *dataset* tendrá un total de 6630 filas, correspondiendo cada mitad (3315) a muestras *ransomware* y *goodware* respectivamente, lo que supone una cantidad considerablemente mayor respecto a la primera limpieza de datos realizada. Este segundo *dataset* se denomina *dataset suma*.

La Tabla 5.1 muestra el nombre, distribución y tamaño de los *datasets* finales que se usarán para alimentar el modelo de [ML](#).

Tabla 5.1: *Datasets* finales.

Nombre	Distribución	Tamaño
<i>Dataset binario</i>	357 muestras de ransomware y 357 muestras de goodware	714 muestras
<i>Dataset suma</i>	3315 muestras de ransomware y 3315 muestras de goodware	6630 muestras

5.4. Modelo y Algoritmos

En esta sección se exponen el modelo de [ML](#) utilizado para la clasificación del ransomware, así como los diversos algoritmos empleados para ello. Para desarrollar un modelo adecuado para el problema es preciso seguir una serie de pasos [Rom19] [TVE20]:

- 1. Definición del problema:** Es necesario conocer si existe una solución para el problema, y si las salidas se podrán predecir con una serie de entradas proporcionadas. Se debe tener claro que el [ML](#) se utiliza para memorizar pautas en los datos, por lo que solo se podrá obtener soluciones a problemas similares al entrenamiento realizado. En el caso de este proyecto, el problema se define como la clasificación de muestras ejecutables en benignas y ransomware.
- 2. Recopilación de los datos:** En esta fase se recogen todos los datos que vayan a ser usados para formar el *dataset* del modelo y se dividen en datos de entrenamiento y datos de prueba. Los primeros serán los utilizados para construir el modelo y los últimos para evaluar el mismo. Se trata de un proceso crítico, puesto que los datos de entrada constituyen el combustible principal para el funcionamiento correcto del modelo. La obtención de los datos se explica con detalle en la Sección 5.2.

3. **Elección de indicadores de éxito:** Las métricas o indicadores son imprescindibles para conocer el funcionamiento y los resultados del modelo, de forma que se puedan introducir cambios para superarlos. En el caso de este estudio se escogen la precisión (*accuracy*), exactitud (*precision*), exhaustividad (*recall* o **TPR**), valor-F (*f1-score*), la tasa de error, **True Negative Rate (TNR)**, **FPR** y **FNR**, explicadas con mayor detalle en la Sección 6.1.
4. **Establecimiento de protocolo de evaluación:** La validación cruzada permite estimar la capacidad de predicción de un modelo, evaluando el rendimiento de los algoritmos de aprendizaje automático que se utilizan [RTWH11]. Se basa en la división de las muestras en un conjunto de entrenamiento para entrenar el modelo y un conjunto de pruebas para evaluarlo. La Figura 5.5 muestra el funcionamiento de la validación cruzada K-Fold [MTSH12].

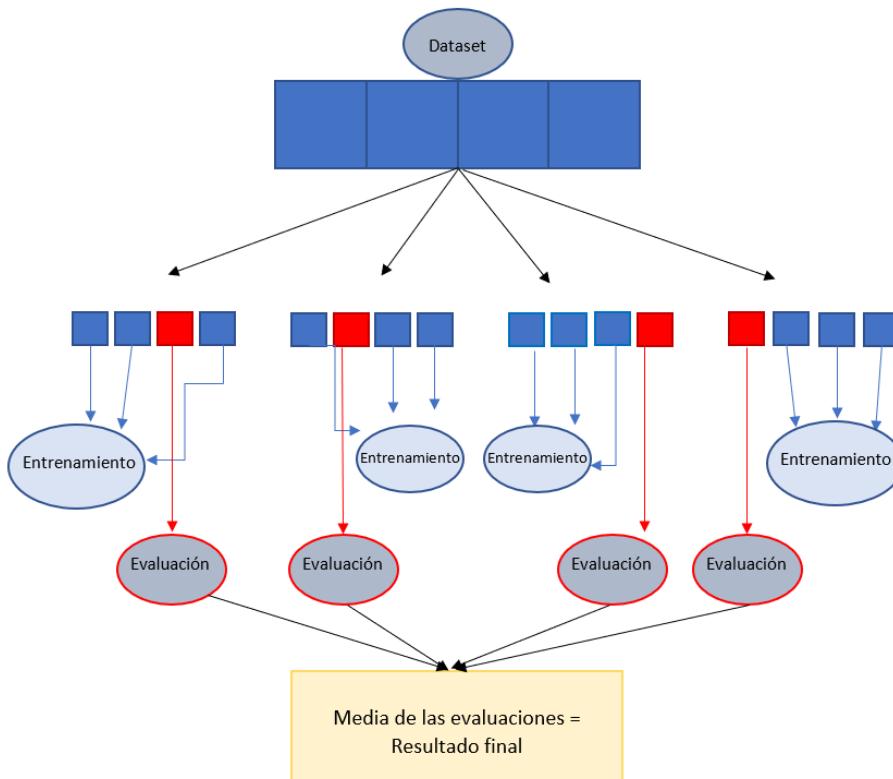


Figura 5.5: Funcionamiento de la validación cruzada K-Fold

En esta validación, el conjunto de muestras se divide aleatoriamente en K subconjuntos de igual tamaño. El entrenamiento se realiza con un subconjunto $k-1$, y el subconjunto restante se emplea para probar los datos. Este proceso se repite k veces para que cada uno de los k subconjuntos se use una vez como datos de prueba. Los resultados de cada iteración se promedian para estimar el rendimiento del modelo [JWHT13]. Este trabajo utiliza el método de validación *k-Fold*, en el cual el *dataset* se divide de manera aleatoria en k subconjuntos. Uno de estos subconjuntos se emplean para el test como datos de validación, mientras que los $k-1$ restantes conforman los datos de entrenamiento. Los k resultados obtenidos son ponderados para proporcionar un único resultado.

5. **Preparación de los datos:** Es necesario procesar los datos antes de introducirlos al modelo, por lo que se debe realizar una limpieza para obtener un formato adecuado: eliminar campos vacíos, despreciar muestras insignificantes, eliminar filas repetidas, etc. Para ello, será necesario aplicar una serie de operaciones de transformación sobre ellos. Se emplean operaciones como la normalización, la estandarización o la aplicación de expresiones no lineales. Es preciso que los datos sean del mismo tipo, y deben ser datos categóricos, transformando los datos a enteros, por ejemplo mediante codificación “one hot” que se aprecia en la Figura 5.6.



Tipo de vehículo	Berlina	Compacto	Utilitario
Berlina	1	0	0
Compacto	0	1	0
Utilitario	0	0	1
Berlina	1	0	0
Compacto	0	1	0

Figura 5.6: Codificación one hot

6. **Extracción de características:** El *dataset* utilizado puede tener ciertas características que no sean importantes para el modelo, por lo que se seleccionaran las necesarias para implementar el modelo de forma que no haya variables redundantes que perjudiquen la precisión del modelo y se obtenga la mayor eficiencia. Se trata de un proceso de vital importancia, que junto a las pautas anteriores para la preparación de los datos, ayuda a reducir el *overfitting*, un mal funcionamiento del modelo debido a la redundancia de los datos. La extracción de características llevada a cabo en el proyecto se detalla en la Sección 5.3.2.
7. **Selección del modelo:** Dependiendo del tipo de datos se vayan a utilizar y la finalidad del modelo, se elegirá un modelo de [ML](#) u otro, ya que existen diversos tipos y cada uno es más adecuado a un tipo de datos o forma de trabajo. Por ejemplo, los modelos de regresión están más destinados a tratar variables continuas.
8. **Desarrollo del modelo:** El objetivo es crear un modelo de comparación, para medir el rendimiento de los diferentes algoritmos para obtener el mejor resultado posible y más ajustado. En este estudio se han utilizado los siguientes algoritmos: [SVM](#) tanto con kernel [RBF](#) como lineal, [NB](#), [KNN](#), [LR](#), [DT](#) y [RF](#), explicados en mayor profundidad en la Sección 3.5.1.
9. **Prueba del modelo:** En esta fase, se prueba el modelo entrenado con los datos seleccionados y se obtienen los resultados correspondientes.

Para el desarrollo del modelo y los experimentos, se utiliza el lenguaje de programación Python y se hace uso principalmente de la librería de [ML](#) “scikit-learn” [PVG+11]. Esta librería *open source* proporciona herramientas simples y eficientes para el análisis predictivo de datos. Su diseño permite la interoperabilidad con la librería matemática NumPy, que proporciona gran capacidad de cómputo numérico en Python. Una vez obtenidos los resultados, se emplea la librería “matplotlib” [Hun07] para mostrar de manera gráfica el comportamiento del modelo, que posibilita la creación de figuras y esquemas de gran precisión totalmente adaptables.

Capítulo 6

Experimentos y Resultados

En este capítulo se exponen los experimentos realizados para demostrar la efectividad del estudio para clasificar ransomware. En la Sección 6.1 se listan las métricas usadas para detectar la efectividad del modelo. La Sección 6.2 expone la elección de los parámetros más relevantes, se detallan los experimentos realizados en los dos *datasets* y se muestran los resultados con diferentes algoritmos. En la Sección 6.3 se elegirá el algoritmo con mejor rendimiento y se compararán los resultados con los de los trabajos mencionados en el Capítulo 4.

6.1. Métricas Utilizadas

Antes de explicar las métricas utilizadas, se deben definir los siguientes conceptos fundamentales:

- **P:** Archivos ransomware.
- **N:** Archivos benignos.
- **True Positives (TP):** Número de archivos ransomware que han sido correctamente etiquetados como ransomware.
- **False Positives (FP):** Número de archivos benignos que han sido incorrectamente etiquetados como ransomware.
- **True Negatives (TN):** Número de archivos benignos que han sido correctamente etiquetados como benignos.
- **FN:** Número de archivos ransomware que han sido incorrectamente etiquetados como benignos.
- **Matriz de confusión:** Matriz mostrada en la Figura 6.1 donde se visualizan los conceptos anteriores para saber el rendimiento del algoritmo.

		Predicción	
		Positivos	Negativos
Observación	Positivos	Verdaderos Positivos (TP)	Falsos Positivos (FP)
	Negativos	Falsos Negativos (FN)	Verdaderos Negativos (TN)

Figura 6.1: Matriz de confusión.

A continuación se explicarán las métricas que se utilizarán en los experimentos:

- **Precisión:** También conocida como *accuracy*, es la proporción de archivos ransomware y benignos que han sido correctamente etiquetados.

$$\text{Precisión} = \frac{TP + TN}{P + N}$$

- **Exactitud:** También conocida como *precision*, es la proporción entre los positivos reales predichos por el algoritmo y todos los casos positivos.

$$\text{Exactitud} = \frac{TP}{TP + FP}$$

- **Tasa de error:** Proporción de archivos ransomware y benignos etiquetados incorrectamente respecto al número total de archivos.

$$ERR = \frac{FP + FN}{P + N}$$

- **TPR:** También conocida como *recall* o exhaustividad, es la proporción de archivos ransomware etiquetados correctamente.

$$TPR = \frac{TP}{TP + FN}$$

- **TNR:** También conocida como *specificity*, es la proporción de archivos benignos etiquetados correctamente.

$$TNR = \frac{TN}{TN + FP}$$

- **FPR:** Proporción de archivos benignos etiquetados incorrectamente.

$$FPR = \frac{FP}{FP + TN}$$

- **FNR:** Proporción de archivos ransomware etiquetados incorrectamente.

$$FNR = \frac{FN}{FN + TP}$$

- **Valor F:** También conocido como *F1-score*, es la media harmónica de la exactitud y la exhaustividad o **TPR**.

$$\text{Valor } F = 2 \cdot \frac{\text{exactitud} \cdot \text{exhaustividad}}{\text{exactitud} + \text{exhaustividad}}$$

6.2. Experimentos

Los experimentos se han realizado sobre los dos *datasets* explicados en la Sección 5.3.2 y se han extraído todas las métricas detalladas en la Sección 6.1. Como bien se ha mencionado en la Sección 5.4, se ha utilizado el lenguaje de programación Python y la librería “scikit-learn” [PVG⁺¹¹] para el desarrollo de los experimentos, y para la visualización de las gráficas se ha usado “matplotlib” [Hun07].

6.2.1. Ajustar Parámetros

Primero se realizan una serie de pruebas sobre los *datasets* para obtener los parámetros que mejor precisión vayan a proporcionar a nuestros experimentos. Los parámetros a estudiar son:

- Método de escalado utilizado: Debido a que el *dataset suma* posee un rango muy amplio de valores, se decide llevar a cabo una estandarización de los datos, de manera que el modelo se comporte de una manera más eficiente y se obtenga un mayor rendimiento. Para ello, en este estudio se proponen dos métodos de la librería de Python “sci-kit learn”. Por un lado, la función *StandardScaler()*, estandariza los valores de X restando la media y luego escalando a la varianza de la unidad, mientras que *MinMaxScaler()*, escala las características a un rango dado, en este caso [0,1]. Mediante la Figura 6.2, es posible observar una mayor precisión en los algoritmos utilizando la función *StandardScaler()*.

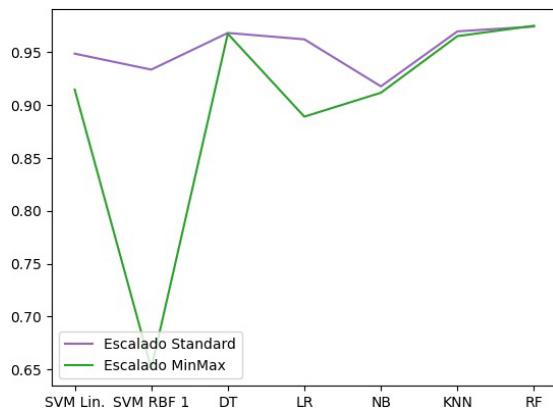


Figura 6.2: Comparación de la precisión con distintos métodos de escalado usando el *dataset suma*

- Valor de K en la validación cruzada: Para la validación mediante la técnica K-Fold se utilizan 3 valores distintos para tratar de obtener el mejor resultado posible. Las Figuras 6.3(a) y 6.3(b) dan constancia de un mejor comportamiento para K=10 en ambos *datasets*, por lo tanto será el valor usado en los experimentos que utilicen validación cruzada.
- Distribución del *dataset*: Se realizan experimentos para utilizar una proporción óptima entre los datos de entrenamiento y los datos de prueba. Como se puede observar en las Figuras 6.4(a) y 6.4(b), se decide utilizar un 70 % de los datos para el entrenamiento del modelo y un 30 % para las pruebas, puesto que reflejan una mayor precisión tanto en el *dataset suma* como en el *dataset binario*.

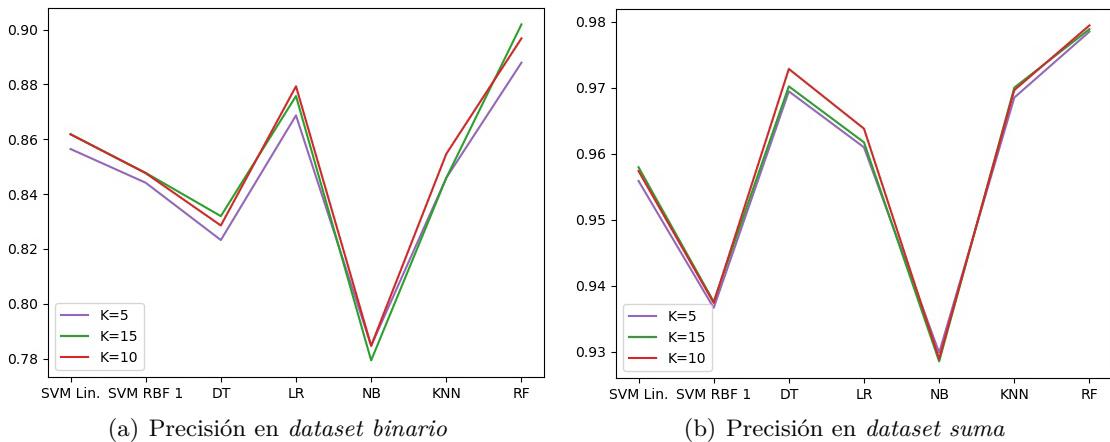


Figura 6.3: Comparación de la precisión con distintos valores de K en validación cruzada K-fold usando en los dos *datasets*

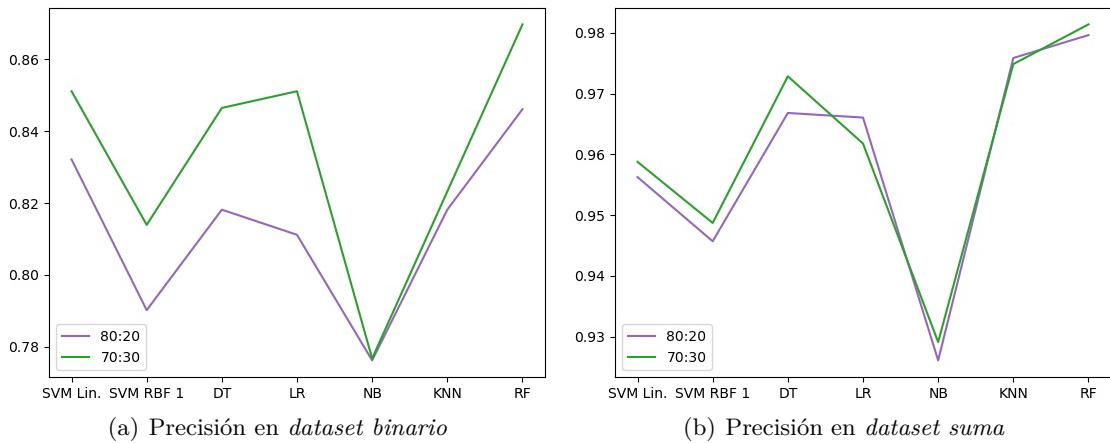


Figura 6.4: Comparación de la precisión con distintas distribuciones de los dos *datasets*

6.2.2. Experimentos con el *Dataset binario*

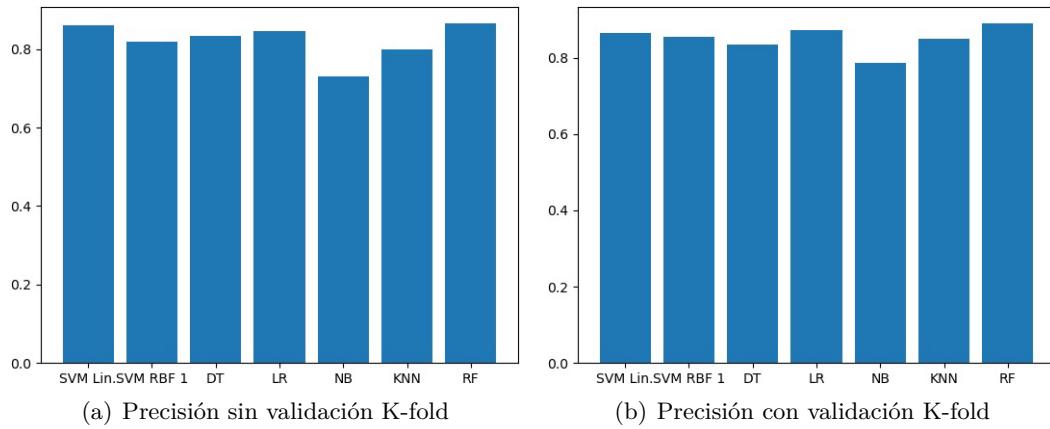
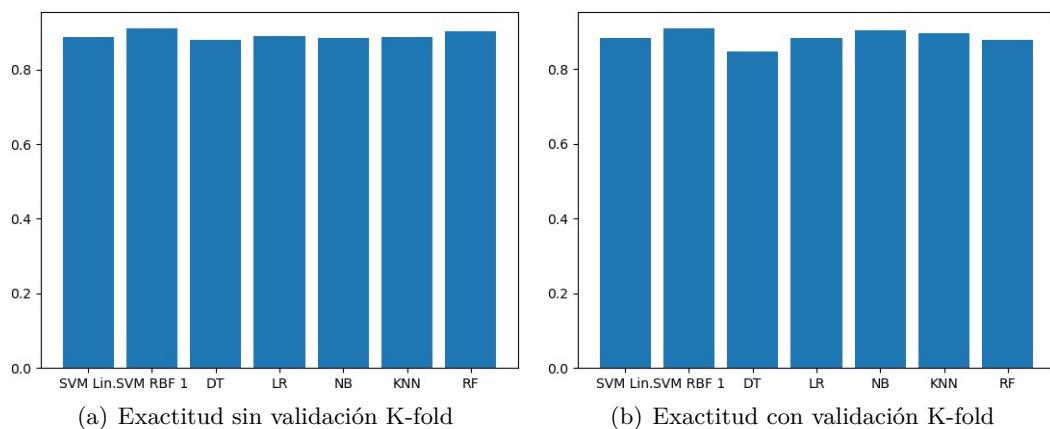
El primer *dataset* es el llamado *dataset binario*, que contiene los archivos y las APIs a las que llama (1 si llama a esa API, 0 si no). Para realizar los experimentos, se utiliza un script de Python para entrenar los datos y realizar las predicciones. Posteriormente, dicho script genera una serie de gráficas para visualizar los resultados y poder decidir qué algoritmo y qué configuración es la deseada para construir el modelo.

Todos los resultados de las métricas analizadas pueden observarse en la Tabla 6.1. Se puede apreciar que el algoritmo con el mejor rendimiento es RF por tener la precisión y el valor f más altos, unas tasas verdaderas (TPR y TNR) altas y tasas falsas bajas (FPR y FNR), lo que significa que el algoritmo ha sido capaz de clasificar las muestras con pocos errores, hecho que se confirma con su tasa de error, que es la más baja de todas (0.041).

Tabla 6.1: Resultados con el *dataset binario*.

Algoritmo	Precisión	Exactitud	Valor F	TPR	TNR	FPR	FNR	Tasa de Error
SVM Lineal	0.86	0.89	0.86	0.84	0.88	0.12	0.16	0.042
SVM RBF	0.82	0.91	0.81	0.72	0.92	0.09	0.25	0.055
DT	0.83	0.88	0.83	0.79	0.88	0.12	0.21	0.050
LR	0.85	0.89	0.85	0.80	0.89	0.11	0.2	0.045
NB	0.73	0.89	0.68	0.55	0.92	0.11	0.34	0.081
KNN	0.8	0.89	0.79	0.71	0.90	0.11	0.26	0.056
RF	0.87	0.90	0.87	0.83	0.90	0.09	0.17	0.041

Las gráficas de barras de las Figuras 6.5, 6.6 y 6.7 permiten comparar la precisión, la exactitud y la exhaustividad (TPR) de los algoritmos del modelo al usar o no la validación K-Fold.

Figura 6.5: Comparación de la precisión con y sin validación k-fold de todos los algoritmos en *dataset binario*Figura 6.6: Comparación de la exactitud con y sin validación k-fold de todos los algoritmos en *dataset binario*

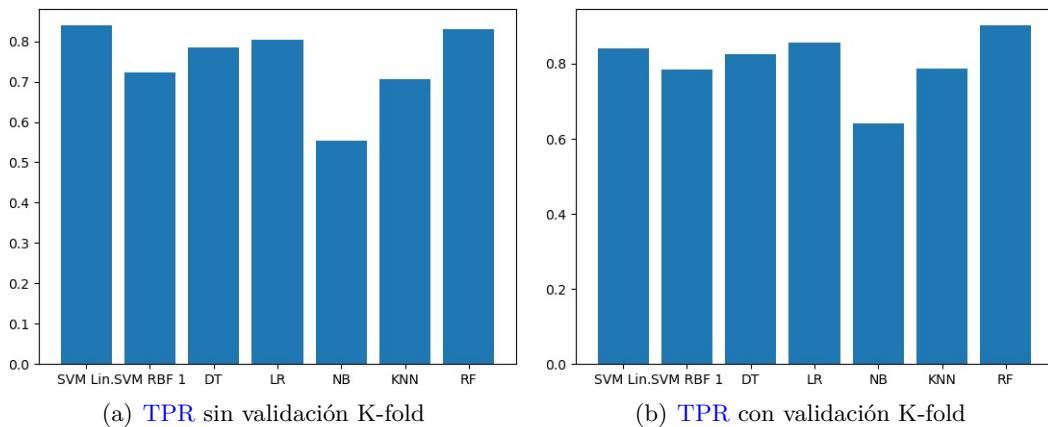


Figura 6.7: Comparación de la TPR con y sin validación k-fold de todos los algoritmos en *dataset binario*

Los resultados no varían demasiado al usar o no la validación K-fold, y hasta algunos algoritmos rinden mejor sin usarla, como es el caso de **RF**, **LR** y **DT** con la precisión y exactitud, y de **SVM Linear** con la **TPR**. Con esto se garantiza que nuestro modelo es válido, ya que los resultados son independientes de la partición de datos de prueba y entrenamiento. Se concluye que los mejores resultados se obtienen con el algoritmo **RF**.

6.2.3. Experimentos con el *Dataset suma*

El segundo *dataset* es el llamado *dataset suma*, que contiene los archivos y el número de veces que llama a las **APIs**. Al igual que para el *dataset* anterior, se utiliza un *script* muy similar, pero se añade la estandarización de las características para reducir el tiempo de cómputo.

En la Tabla 6.2 se muestran todos los datos obtenidos en los experimentos realizados para el *dataset suma*. Al igual que sucedió con el primer *dataset*, el algoritmo que da mejores resultados es **RF**, siendo mejor que los demás en todas las métricas exceptuando la tasa de falsos positivos **FPR** con un valor de 0.026, pero la diferencia con la mejor, que es 0.020 con el algoritmo **SVM RBF**, es mínima.

Tabla 6.2: Resultados con el *dataset suma* escalando los datos con *StandardScaler*.

Algoritmo	Precisión	Exactitud	Valor F	TPR	TNR	FPR	FNR	Tasa de Error
SVM Lineal	0.95	0.96	0.95	0.95	0.96	0.04	0.051	0.014
SVM RBF	0.95	0.93	0.95	0.96	0.93	0.068	0.041	0.021
DT	0.98	0.98	0.97	0.97	0.98	0.020	0.03	0.008
LR	0.96	0.97	0.96	0.95	0.97	0.03	0.05	0.012
NB	0.92	0.97	0.91	0.87	0.97	0.03	0.12	0.024
KNN	0.97	0.98	0.97	0.96	0.98	0.23	0.036	0.009
RF	0.98	0.97	0.98	0.99	0.97	0.026	0.013	0.006

Las Figuras 6.8, 6.9 y 6.10 muestran las diferencias en las métricas de precisión y exactitud tras usar o no la validación K-Fold. Como se ha visto en el primer *dataset*, los resultados al usar o no la validación no son muy diferentes, y en algoritmos como **RF** y **DT** los resultados son mejores sin la validación cruzada. Una vez más se demuestra la validez del modelo, gracias a que los resultados con y sin validación son muy similares.

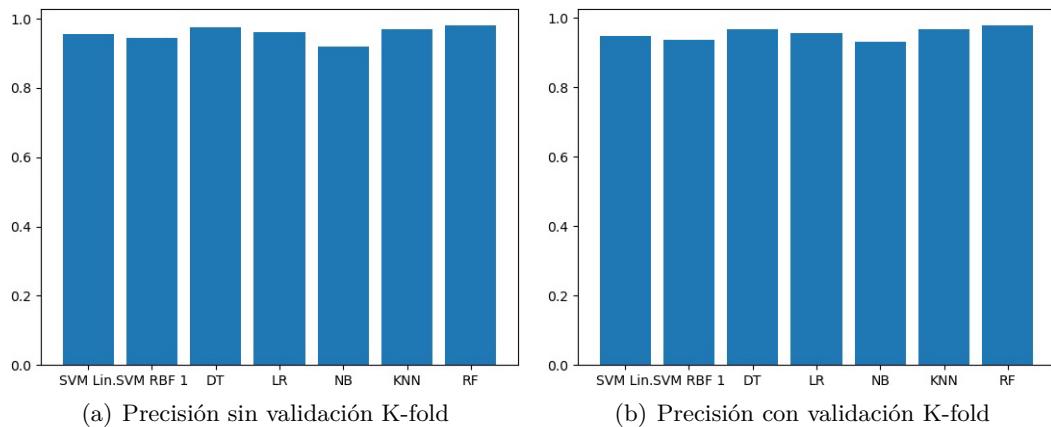


Figura 6.8: Comparación de la precisión con y sin validación k-fold de todos los algoritmos en *dataset suma*

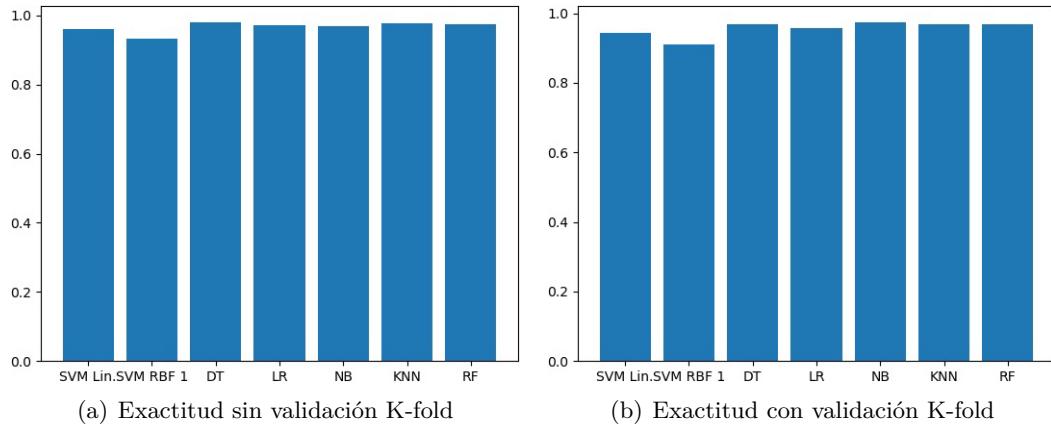


Figura 6.9: Comparación de la exactitud con y sin validación k-fold de todos los algoritmos en *dataset suma*

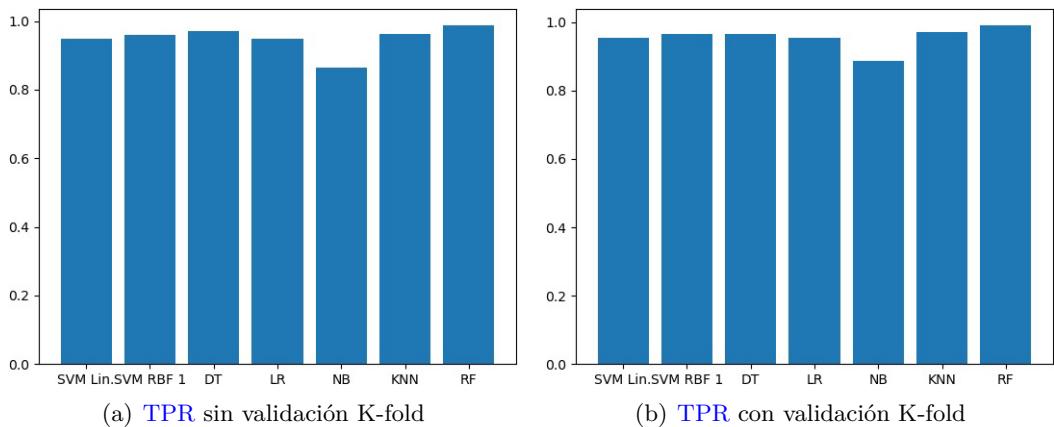


Figura 6.10: Comparación de la TPR con y sin validación k-fold de todos los algoritmos en *dataset suma*

Al igual que con el primer *dataset*, se concluye que el algoritmo más eficiente y con los mejores resultados es **RF**.

6.3. Resultados Finales

Como se ha visto en las Secciones 6.2.2 y 6.2.3, el algoritmo con los mejores resultados es **RF**, usando el *dataset suma* distribuido de manera que el 70 % de los datos se usen para el entrenamiento del modelo y un 30 % para las pruebas. Con todo esto en cuenta, se generará el modelo final de este trabajo.

A continuación se muestran la Tabla 6.3 que compara los resultados del modelo final de este trabajo con los resultados de los otros trabajos que usan el algoritmo **RF**, vistos en la Tabla 4.2 del Capítulo 4. Se puede apreciar que los resultados de este trabajo están entre los mejores, teniendo las menores tasas de falsos positivos y negativos (**FPR** y **FNR**), la segunda mejor tasa de verdaderos positivos (**TPR**) y unos valores de precisión y valor F bastante altos, demostrando la efectividad del modelo propuesto.

Tabla 6.3: Comparación de los resultados obtenidos con otros trabajos usando el algoritmo RF

Trabajo	Precisión	Exactitud	Valor F	TPR	TNR	FPR	FNR	Tasa de Error
[VSVG17] (2017)	0.98	0.98	0.99	1.0	-	-	-	-
[CKYK17] (2017)	0.958	-	-	0.96	-	0.047	-	-
[BLI19] (2019)	0.98	0.98	0.98	0.98	-	-	-	-
[KAJS19] (2019)	0.92	-	-	0.91	-	0.07	-	-
[Jet19] (2019)	0.97	-	0.97	0.97	-	-	-	-
[HKLK20b] (2020)	0.973	-	0.97	-	-	0.048	0.015	-
[KAJ20] (2020)	0.99	-	0.99	0.99	-	-	-	-
[ADPC20b] (2020)	0.75	-	-	-	-	-	-	-
[ARA ⁺ 21] (2021)	0.97	-	0.97	0.94	-	-	-	-
Este trabajo	0.98	0.97	0.98	0.99	0.97	0.026	0.013	0.006

La Tabla 6.4 coteja el *dataset* construido en este trabajo con el de los demás trabajos.

Tabla 6.4: Comparación de los *datasets* construidos con los del estado del arte

Trabajo	Distribución de las muestras	Tamaño total
[VSVG17] (2017)	755 muestras de ransomware y 219 muestras benignas	974
[CKYK17] (2017)	83 muestras de ransomware y 85 muestras benignas	168
[BLI19] (2019)	<i>Dataset</i> conformado por 1000 muestras de ransomware, 900 muestras de malware y 300 muestras benignas	2200
[KAJS19] (2019)	582 muestras de ransomware y 942 muestras benignas	1524
[Jet19] (2019)	1072 muestras de ransomware y 1072 muestras benignas generadas sintéticamente a partir de un <i>dataset</i> real de 663 muestras ransomware y 103 muestras benignas	2144
[HKLK20b] (2020)	1909 muestras de ransomware y 1139 muestras benignas	3048
[KAJ20] (2020)	904 muestras de ransomware y 942 muestras benignas.	Eliminando repetidos, al igual que en este estudio, se obtiene un <i>dataset</i> final de 912 muestras totales.
[ADPC20b] (2020)	34 muestras de malware, de las cuales 7 son ransomware, y 41 muestras benignas	75
[ARA ⁺ 21] (2021)	80 muestras de ransomware y 80 muestras de otros tipos de malware	160
Este trabajo	<i>Dataset</i> binario: mitad ransomware mitad benigno. Representada la aparición de las API en formato binario	714
	<i>Dataset</i> suma: Muestras ransomware y benignas a partes iguales. Representando la aparición de las API por el número de llamadas.	6630

Los datos reflejan la superioridad en tamaño del *dataset* conformado en este trabajo, en concreto el segundo *dataset*, con un número total de 6630 muestras, que es el que se ha usado para obtener los mejores resultados. Además, gracias a la cortesía de S.H. Kok *et al.* en [KAJ20], se han podido hacer pruebas con el *dataset* que utilizan en el trabajo. Tras realizar el proceso de limpieza de su *dataset* con el sistema propuesto, se reducen las muestras casi al 50 %. Se puede asumir por lo tanto, que el *dataset* del modelo propuesto, a parte de ser mayor, es menos propenso al *overfitting*, debido a que no aparecen muestras repetidas en ningún caso.

Por otro lado, al utilizar dos *datasets* diferentes, se proporcionan dos enfoques distintos, realizando así un estudio más completo. Con toda esta información, y debido a unos resultados muy favorables vistos en la Tabla 6.3, se puede afirmar que se obtiene una mayor precisión en la identificación de ransomware al utilizar una mayor cantidad de muestras, con un equilibrio entre las muestras benignas y de ransomware y recogiendo la cantidad de llamadas a las API de Windows en lugar de utilizar su aparición.

Capítulo 7

Conclusiones y Trabajo Futuro

7.1. Conclusiones

Los ataques ransomware son cada vez más comunes y, en consecuencia, son más las empresas y organizaciones que los sufren, en concreto las SME. Estas constituyen un pilar fundamental en la economía europea y sus recursos destinados a la prevención de ciber-ataques son más limitados. Por lo tanto, existe la necesidad de elaborar un modelo ML *open source* de fácil uso, capaz de proteger a este tipo de empresas de manera eficaz. Los datos de entrada del modelo consisten en las llamadas a la API de Windows que realizan las muestras ejecutables. Tras la realización de este trabajo se han obtenido las siguientes conclusiones:

- En el campo de la detección de malware, el uso de ML es fundamental para obtener resultados óptimos, por lo que se opta por crear un modelo de ML.
- En trabajos sobre análisis y detección tanto de malware como de ransomware, los algoritmos utilizados en los modelos de ML son algoritmos de aprendizaje supervisado.
- Las llamadas a la API de Windows son un atributo fiable para analizar y detectar ransomware, ya que una gran cantidad de proyectos las emplean como características principales para la construcción de sus modelos.
- Usar el número de llamadas a cada una de las APIs para construir un *dataset* proporciona mejores resultados que considerar únicamente la aparición de estas con una representación binaria. Esto se evidencia en los experimentos, donde se usan dos *datasets*, *dataset binario* y *dataset suma*, siendo el *dataset suma* el que obtiene una mejora de alrededor de un 10 % en los resultados.
- El *sandboxing* es una técnica muy fiable para la ejecución y análisis de muestras en un entorno seguro. Por lo tanto, se construye un laboratorio de análisis de malware para examinar este tipo de muestras, puesto que suponen un gran riesgo para los equipos. Para analizar las muestras, se utiliza una herramienta llamada Cuckoo Sandbox, que ofrece multitud de configuraciones para recoger características específicas de las muestras.
- La mayoría de los estudios relacionados con la detección de ransomware utilizan *datasets* relativamente pequeños, en comparación con la cantidad de muestras que se utilizan para el ML en general. Por lo tanto, en este trabajo se ha utilizado

un primer *dataset* de extensión similar a los de los otros trabajos, obteniendo una precisión del 87 %, y un segundo *dataset* considerablemente mayor, obteniendo una precisión del 98 %. Con esto se concluye que el sistema propuesto obtiene mejores resultados con un *dataset* más extenso que los otros trabajos usando menos datos.

- En la experimentación se han utilizado 7 algoritmos de [ML](#) y [RF](#) ha sido el que mejor resultados ofrece a la hora de detectar muestras de ransomware, siendo mejor que otros algoritmos como [DT](#), [SVM](#) o [NB](#).

7.2. Trabajo Futuro

Las posibles líneas de investigación y desarrollo en un futuro son las siguientes:

- Implementación de un modelo cliente-servidor incluyendo una [API](#) de consumo o una interfaz gráfica para facilitar el uso del modelo. De esta manera, apoyándose en el modelo de [ML](#) construido, sería posible analizar archivos para detectar la posible presencia de ransomware.
- Incluir un sistema basado en firmas para la detección de ransomware, almacenando dichas firmas en una base de datos accedida para comparar las firmas de las nuevas muestras analizadas.
- Implementar detección temprana de ransomware, encontrando la presencia del mismo antes de su ejecución.
- Tras la preparación y limpieza del *dataset*, se ha reducido el número de muestras finales, por lo es conveniente continuar ampliando el conjunto de datos con muestras de otras fuentes, aumentando así la fiabilidad del sistema.
- Considerar otras perspectivas para mejorar la detección de ransomware. Entre las posibles se encuentran analizar las conexiones a la red efectuadas por la muestra, la entropía del sistema, la monitorización del sistema de archivos o la observación de los [HPC](#), tal y como se ha visto en otros trabajos relacionados en el Capítulo 4.
- Tener en cuenta muestras de malware y no solamente de ransomware, creando un modelo con más posibilidades de uso.

Capítulo 8

Contribuciones Individuales

En este capítulo se explica la implicación de cada miembro del equipo y las tareas que han realizado para la elaboración de este trabajo. Cabe destacar que los tres miembros estaban en constante contacto y se ayudaban mutuamente, tanto en la fase de investigación como en la de desarrollo, experimentación y escritura de la memoria. Trabajar en equipo para realizar las tareas era indispensable y se ha llevado a cabo con éxito, ya que todos los miembros han adquirido conocimientos importantes y han aportado en todas las áreas del trabajo.

8.1. Gonzalo Fernández Megía

Al inicio, se realizó una búsqueda exhaustiva de información relacionada con el tema, en concreto con proyectos ya realizados por otros investigadores. Leímos gran cantidad de artículos, de los que se realizaban resúmenes, extrayendo conclusiones, plasmándolas en tablas. Se trató de un proceso que nos llevó gran cantidad de horas, puesto que la intención era tener una base sólida de conocimientos para poder después desarrollar nuestro trabajo propio, de manera que estuviera bien fundamentado y tuviera la calidad necesaria. Toda esta información ha sido utilizada a lo largo del proyecto para replicar otros trabajos. En mi caso, además hice hincapié en la investigación de los conjuntos de datos utilizados por estudios similares, así como en otros trabajos exclusivamente de [ML](#), información que nos sería útil para la construcción de nuestro *dataset*, y que luego nos serviría para hacer la comparación con nuestro *dataset*, superior en tamaño.

Una vez elegido el camino a seguir y los métodos a utilizar para la detección de ransomware, se realiza otra recogida de información, esta vez para contextualizar el trabajo y proporcionar conocimientos sobre las materias claves del estudio: el malware, el ransomware y los métodos más importantes para analizarlos. Al empezar a escribir la memoria del trabajo, realicé una búsqueda de información sobre la situación social y económica en relación a los ataques ransomware para enmarcar nuestro trabajo en la actualidad y poder preparar una introducción adecuada al momento. Es destacable también el aprendizaje de lenguaje Latex para el desarrollo y escritura de la memoria, haciendo uso de manuales, que nos proporcionó un mecanismo muy potente a la hora de plasmar toda la información con apoyo visual como tablas, figuras o esquemas. Posteriormente se continúa estudiando para obtener material, en este caso, centrándonos más en los aspectos tecnológicos y científicos para poder aplicarlos en nuestro trabajo de manera independiente. Tras toda esta investigación, se decide realizar un modelo de [ML](#) para la identificación de ransomware, puesto que es el mecanismo más usado por la mayoría de los investigadores y el que mejor resultados ofrece. Como características

elegimos las llamadas a la [API](#) de Windows, puesto que son relativamente fáciles de extraer y constituyen un artefacto sólido para la detección de ransomware debido a que trazan el comportamiento de este en el sistema víctima.

A continuación, debíamos comenzar a construir un *dataset* para alimentar el modelo y comenzar a hacer pruebas. Tras la investigación relativa a los conjuntos de datos, pudimos solicitar a aquellos autores cuyos *dataset* se consideraban de mayor calidad para realizar las primeras aproximaciones y establecer una referencia que posteriormente nos facilitaría la tarea. Resultó de ayuda para mis compañeros en el desarrollo de los *scripts* necesarios para la extracción y limpieza de los datos para la creación de nuestro conjunto de datos propio, proceso fundamental en el desarrollo del proyecto, puesto que era de vital importancia obtener un conjunto de datos adecuado y lo suficientemente grande con la finalidad de alimentar al modelo de [ML](#) con datos reales y consistentes, obtener óptimos resultados y evitar el *overfitting*. Primero se construye el *dataset binario*, en el que solo se cuentan las apariciones de las llamadas a la [API](#) de Windows. Tras aplicar el algoritmo de limpieza de datos se ve reducido de manera brusca. Debido a este factor, y al de ampliar la investigación para obtener otro enfoque, se construye un segundo conjunto de datos, el *dataset binario*. Los datos que contiene se forman sumando la cantidad de llamadas a la [API](#) de Windows que realizan los procesos pertenecientes a las muestras obtenidas. Obtenemos así un *dataset* de mayor tamaño tras la limpieza y unos mejores resultados posteriores.

De manera paralela, se empieza a desarrollar el modelo. Lo primero que hice fue aprender a construir un modelo desde cero mediante el uso de la librería de Python *sci-kit Learn*, para lo cual, tuve que leer la documentación ofrecida por los desarrolladores en su manual de uso. En los inicios tuve que llevar a cabo gran cantidad de pruebas con distintos conjuntos de datos y diversos algoritmos para entender el funcionamiento del sistema y obtener la configuración y parámetros idóneos para nuestro proyecto. Una vez construido un modelo mínimamente funcional, comienzo a desarrollar los experimentos en mi máquina virtual Kali Linux 2020.3, y a registrar los primeros resultados. Decidimos llevar a cabo la extracción de gran variedad de métricas para analizar mejor dichos resultados y poder mejorar la efectividad del modelo. Asimismo, se utilizan un total de 7 algoritmos de [ML](#) distintos, consiguiendo una gran variedad en los experimentos y unos resultados contrastados. Cuando el modelo alcanza la efectividad que esperábamos, se lleva a cabo una recogida de los resultados, los introducimos en tablas y realizo la labor de plasmarlos en gráficas para su mejor entendimiento.

También contribuyo a la creación de un [repositorio online](#) para el almacenamiento de los ficheros relacionados con el *dataset* y el modelo. Además, junto a mi compañero Gonzalo Figueroa, hemos desarrollado un manual para su mejor entendimiento y uso de manera más sencilla para todo tipo de usuario.

Es necesaria la mención al desarrollo de un diagrama de Gantt, que muestra las actividades realizadas por el grupo de trabajo a lo largo del tiempo. Este facilita el entendimiento de la pauta de trabajo que hemos seguido y cuáles son los ciclos de tiempo de realizar un estudio como el nuestro.

Finalmente, aunando resultados obtenidos e información recogida, somos capaces de redactar una serie de conclusiones finales para afianzar nuestro estudio, que traduzco al inglés junto con mi compañero Gonzalo Figueroa.

8.2. Gonzalo Figueroa del Val

A la hora de iniciar el proyecto, lo primero fue realizar una amplia investigación acerca del tema a tratar para construir una base de conocimiento de la que partir. Para esta investigación, hemos leído y analizado gran cantidad de trabajos y libros sobre análisis y detección de malware, haciendo hincapié en el detección de ransomware, aprendiendo sobre diferentes técnicas y enfoques para ello. La documentación leída la hemos obtenido de fuentes verificadas, como la biblioteca online de la Universidad Complutense de Madrid o editoriales como *SpringerLink* o *IEEE Xplore*, entre otras, realizando las búsquedas a través de *Google Scholar*. Además, recogimos las referencias correspondientes en formato *BibTeX* para incorporarlas a nuestra memoria.

El siguiente paso fue montar un laboratorio de análisis de malware para extraer características de las muestras que teníamos para analizar. El laboratorio lo construí en mi equipo con sistema operativo Linux con distribución Ubuntu LTS 18.0.5 e incorporando la herramienta Cuckoo Sandbox, la cual ha sido esencial para el desarrollo del trabajo. El proceso de montar el laboratorio fue bastante tedioso puesto que el trabajo con este entorno era nuevo para mí y aparecían continuamente errores. Finalmente, consultando el manual de Cuckoo Sandbox y otras fuentes externas, el laboratorio de análisis de malware quedó construido y listo para trabajar, desarrollando mi propio manual de montaje y configuración que se puede contemplar en la Sección 5.2.1 del Capítulo 5.

Mientras no dejábamos de empaparnos de información, se llegó a la conclusión de que la característica principal que alimentaría nuestro trabajo y por lo tanto también el modelo de [ML](#) que desarrollaríamos serían las llamadas a la [API](#) de Windows. Esta característica se puede obtener fácilmente de los informes generados por Cuckoo Sandbox al analizar muestras ya que se concentran en una sección concreta llamada '*apistats*'. Con esto, empezamos a trabajar en la extracción de las llamadas a la [API](#) de Windows por cada proceso de ejecución del malware dando lugar a la creación de un *script Python* para procesar la información que acabaría construyendo nuestro *dataset*.

Con nuestros informes y los proporcionados por el grupo de investigación dediqué bastante tiempo para, con la ayuda de mis compañeros, escribir los *scripts* definitivos que procesarían correctamente todos los informes y limpiarían los datos adecuadamente para así construir los dos *datasets* finales que se utilizarían para los experimentos. El primero de ellos, nombrado como *dataset binario* representa de manera binaria la llamada o no de cada [API](#) durante la actuación de cada muestra y se construyó basándonos en otros trabajos que utilizaban también esta representación binaria. La creación del segundo *dataset* nombrado como *dataset suma* cuenta para cada llamada [API](#) el número de veces que ha sido realizada por la muestra, diferenciándonos así de otros trabajos utilizando este *dataset*. Para realizar todo este proceso se tuvo en cuenta las buenas prácticas para construir un *dataset* adecuado, teniendo en cuenta las fases de extracción, transformación, limpieza y carga.

A la hora de trabajar con [ML](#), se han tenido en cuenta todas las fases correspondientes al proceso de desarrollar un modelo de aprendizaje automático, desde la definición del problema hasta la prueba del modelo. Probamos multitud de algoritmos y configuraciones para los dos *datasets*, probando diferentes técnicas de escalado, repartos de datos de entrenamiento y prueba o validaciones cruzadas con varios valores. Una vez recopilados los resultados ofrecidos por el modelo, los plasmamos en la memoria y realizamos una comparación entre los algoritmos utilizados y las distintas configuraciones probadas. Además, comparamos los resultados con los trabajos del estado del arte más relevantes. Por último, se redactaron las conclusiones finales para terminar la memoria del trabajo.

Cabe mencionar que también hemos creado un [repositorio online](#) donde se almacenan los datos y ficheros relacionados con la construcción del *dataset* y el modelo y donde junto a mi compañero Gonzalo Fernández se ha escrito un manual de uso del sistema propuesto, incluyendo la instalación de requisitos previa al desarrollo del *dataset* y del modelo, la construcción del laboratorio y del *dataset* y la ejecución del modelo de [ML](#) desarrollado con sus diferentes configuraciones.

Durante todo el proceso de investigación y experimentación he realizado consultas a los manuales de las diferentes herramientas o módulos de Python de los que he hecho uso, así como realizado consultas concretas en foros muy reputados como *StackOverflow*. En cuanto al equipo hemos trabajado con una comunicación constante estructurando tareas y colaborando entre todos contribuyendo al proyecto de forma equilibrada y apoyándonos en las dificultades encontradas.

8.3. Marina López Osorio

Al comenzar este trabajo, el primer paso fue buscar información, artículos y todo tipo de estudios sobre el malware, su análisis y detección para luego centrarse en el ransomware. Para encontrar toda esta información, mis compañeros y yo usamos la herramienta llamada *Google Scholar* que permitía buscar con facilidad artículos científicos y leer los que eran más citados por otros estudios, lo que indicaba que era un artículo útil y con información veraz. Leyendo estos artículos aprendí los conceptos fundamentales del malware y ransomware, como se distribuían e infectaban a los sistemas, cómo se podían analizar y detectar, los diferentes tipos y familias, etc. También buscamos información sobre el aprendizaje automático, ya que muchos trabajos lo utilizaban para la detección de amenazas, y una asignatura que me ayudó a comprender los conceptos y me enseñó algunos algoritmos de aprendizaje automático fue Ingeniería del Conocimiento. Además de eso, realicé algunos cursos de Coursera sobre aprendizaje automático para afianzar lo aprendido. Con todos estos conocimientos ya nos pudimos centrar en cómo implementar un modelo de [ML](#) para detectar ransomware.

El método más común que utilizan los trabajos para detectar amenazas tanto de ransomware como de malware es la extracción de las llamadas a la [API](#) de Windows que realizan al infectar el sistema de la víctima. Estas llamadas son las que transforman en vectores de características para alimentar al modelo de [ML](#) para detectar ransomware. Para la extracción de características, la mayoría de trabajos utilizaban la herramienta llamada Cuckoo Sandbox, por lo que buscamos información sobre ella y empecé a hacer pruebas con muestras en la versión online de la herramienta. Cuckoo Sandbox devolvía unos reportes en formato json, por lo que tuve que escribir un *script* en Python que trajese las llamadas a la [API](#). Había programado antes en Python pero muy pocas veces en comparación con otros lenguajes como C++ o Java, por lo que tuve que refrescarme la memoria y familiarizarme con la librería de *scikit-learn* para implementar los algoritmos de aprendizaje automático, y otras librerías como *Pandas* para la lectura y análisis de las muestras. Este *script* que extraía las [APIs](#) de los reportes de Cuckoo Sandbox al final fue reemplazado por otro, ya que los datos de las [APIs](#) los terminamos cogiendo directamente de otra fuente con los reportes ya analizados por Cuckoo, pero sirvió para familiarizarme con la herramienta y saber qué se sacaba exactamente de las muestras.

Al tener el vector de las características, era hora de construir el modelo. Busqué tutoriales de todo tipo y empecé a escribir el código, utilizando simplemente un algoritmo de aprendizaje supervisado, que era [SVM](#). Después de muchas pruebas con diferentes datos, tanto de nuestras muestras como las de otros trabajos, junté mi *script* con el de

mis compañeros y así pudimos comparar posibles errores y tener más algoritmos para obtener resultados con todos ellos y ver cual era el que mejor rendía. Para obtener estos resultados, se utilizaron las métricas del módulo `sklearn.metrics` de la librería `scikit-learn`. Busqué información de todas las métricas que se podían sacar y elegí las que me parecían más relevantes y las que usaban otros trabajos relacionados. Para la validación de los resultados, decidimos usar la validación cruzada K-Fold, por lo que busqué información de cómo implementarla y lo añadimos al código.

El modelo sufrió varios cambios debido a que nuestro *dataset* también los tuvo, y después obtener los dos *datasets* finales (*dataset binario* y *dataset suma*), se pudo experimentar más a fondo con los algoritmos y obtener los resultados definitivos. Al ver que el *dataset suma* tenía valores no binarios, era necesario escalar los datos, ya que el tiempo de ejecución era demasiado alto y algunos algoritmos no funcionaban correctamente. Investigué y encontré dos escaladores de la librería `scikit-learn` que podríamos utilizar, el *MinMaxScaler* y el *StandardScaler*. Experimenté con los dos escaladores sobre el *dataset suma* y el *StandardScaler* obtuvo mejores resultados, y esto se confirmó cuando mi compañero realizó las gráficas que lo mostraban de manera clara. Posteriormente pasé a experimentar con el *dataset binario* y a obtener los resultados. De todos los algoritmos usados, vi que el **RF** era el que mejores resultados obtenía, por lo que ese fue el que se usó para construir el modelo final de este trabajo.

Durante las etapas de investigación y desarrollo del sistema propuesto, se fue realizando la memoria en *LaTeX*, una herramienta para redactar artículos y documentos en el ámbito de la investigación. Nunca la había usado antes, pero gracias a mis tutores y al manual que se puede encontrar en la web, no fue difícil adaptarme a la herramienta y me pareció muy útil, debido a que facilitó enormemente la realización de las citas. Escribí todos los conocimientos que iba adquiriendo y el desarrollo del trabajo a medida que avanzábamos, coordinándome con mis compañeros para no repetir información. También me encargué de la traducción al inglés de varias partes de la introducción.

Contribuí a la creación del [repositorio online](#) y público en Gitlab para el almacenamiento de todos los ficheros relacionados con el desarrollo del modelo.

A lo largo del desarrollo de este trabajo he adquirido conocimientos tanto técnicos como de trabajo en equipo, ya que tenía que estar en constante contacto con mis compañeros y asistir a reuniones semanales con mis tutores, siendo una experiencia importante para el futuro en una empresa. También adquirí la habilidad de extraer información de fuentes científicas, distinguiendo cuáles son las fiables de las que no. Este trabajo ha sido una buena experiencia que me ha permitido aprender mucho sobre la ciberseguridad y el aprendizaje automático, dos áreas que actualmente están muy presentes en el mundo laboral.

Capítulo 9

Introduction

Nowadays, information is stored digitally and can be easily accessed at any time. Most people are connected continuously with numerous devices through Internet, an increasingly safe platform due to the work of many specialists, but in which cybercriminals still find vulnerabilities and attack all kinds of entities to acquire profits [SM17]. There are many malicious programs such as viruses, worms or trojans that can damage systems. One of the most common cyberattacks according to a report published by the Europol (European Union Agency for Law Enforcement Cooperation) [Zet15] is ransomware. Ransomware is a malware type that blocks devices or prevents data access using private key encryption until the victims pays an economic ransom, usually in bitcoin. Data theft extortion has been used since 2005, but due to the rise of ransomware and bitcoin, the number of cases has increased hugely and almost 40 % of companies have suffered at least one ransomware attack according to a study by Malwarebytes [Gua16], which shows the evidence that these malicious programs are an important threat to detect and mitigate as soon as possible.

Cybercriminals are becoming more creative and innovative and the damage caused by them is getting worse. Figure 9.1 shows the prediction of mitigating damage cost against ransomware attacks for the year 2021, making it clear that there has been a huge increase in comparison to the past decade. These costs are both to the investment in defending systems and the losses resulting from not being able to use the devices and the ransom payments.

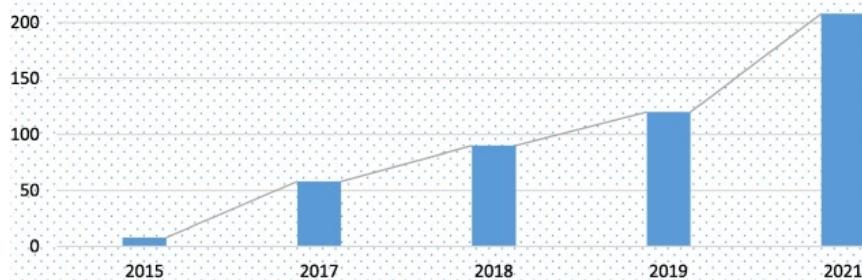


Figura 9.1: Ransomware cost prediction in billions of dollars

Figure 9.2 shows the results obtained from a Coveware study [Cov20] about most attacked industries by ransomware attacks in the last quarter of 2020. Professional services, public sector and health sector are the most affected, considering that they are the most profitable for criminals. This is caused by the need of being in constant operation and because they store sensitive data, so they usually pay the ransom to recover their data

and the access to the system as soon as possible.

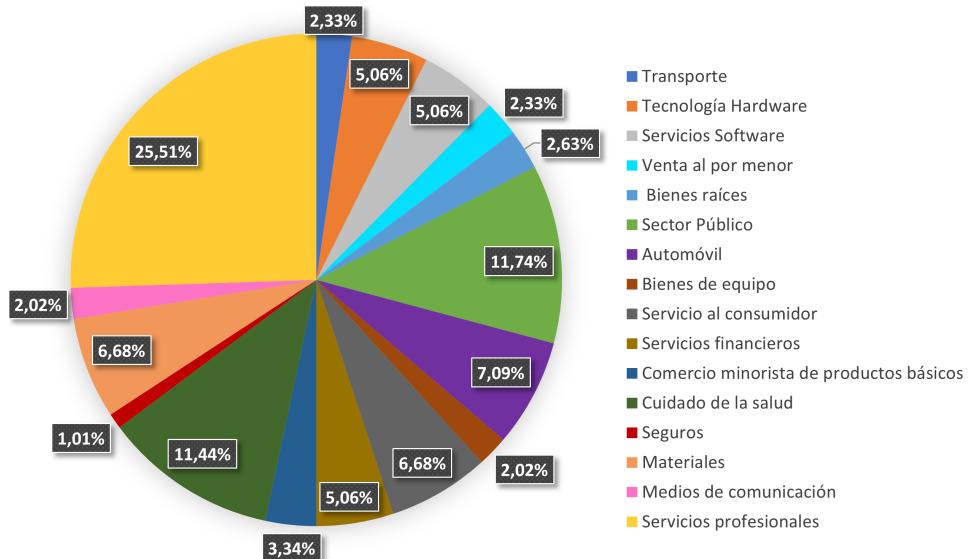


Figura 9.2: Most attacked sector by ransomware in the third quarter of 2020

The risk of this is just as significant for large enterprises as for SME. These companies also use, produce and store large amounts of data. The difference is that smaller companies have to deal with these problems using limited resources. Therefore, they need an economic effort to draw up security strategies against these cyberattacks [Kur15].

The current health situation is also added to the suggested scene, a pandemic caused by the COVID-19 disease. While this disease was spreading around the world, a technological threat was growing based on indiscriminate cyberattacks and cybercampaigns focused on public authorities and organisations [LSN⁺21], as well as on people based on their misinformation and concern, such as the false distribution of COVID-19 books [TIT20], the fraudulent medicine offered via email [FN20], etc. Another factor that supported the increase of cyberattacks, especially ransomware attacks, is the establishment of remote work. In Spain, these attacks increased by 160% in the second half of 2020, leading the European ranking, due to the fact that companies that focused their efforts on establishing remote work environments, did not apply the necessary security to do so [pE20]. In Spain, the affected companies have not been only private ones, public institutions have as well received ransomware attacks. The SEPE had to paralyse its services for 5 days, adding 20 without remote work due to an attack in March 2021 by Ryuk ransomware, causing a serious situation, so the SEPE activity shot up due to the pandemic impact on unemployment [JP21].

The development of technology provides benefits and facilities in all fields and tasks, even in the more quotidian. An example of this is IoT, that is defined as the integration of real life places and objects that use Internet through an interconnected network of devices, sensors, software, etc. that store and exchange data [ADC18]. However, the existence of this technology allows for the number of ransomware attacks to grow over time, especially after the use of intelligent IoT devices, in particular smartphones. In Figure 9.3 we can observe the amount of ransomware attacks intended to smartphones in each quarter of 2018. Figure 9.4 shows a prediction of how the use of these devices will rise due to the growth of IoT. Based on this data we can predict that the amount of attacks will continue rising and it demonstrates the need both for protecting users and organisations from this

type of attacks and for providing users and workers the existing information to act in a responsible and safe way to prevent them [HJAP21].

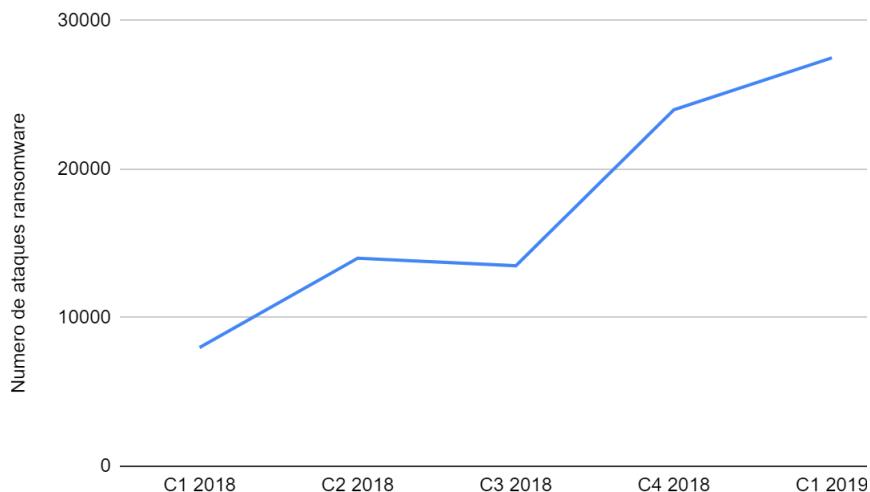


Figura 9.3: Ransomware attacks on mobile devices

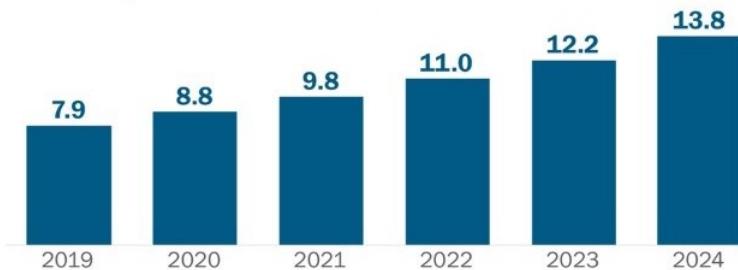


Figura 9.4: Prediction of the number of mobile IoT devices in billions

9.1. Motivation

As seen in Section 1.1, the impact of the ransomware attacks on companies in recent years is evident, causing large financial investments from those companies to improve their security. SMEs have greater difficulty in meeting those expenses due to the lack of prevention systems, specialists focused on cybersecurity and information related to this sector, therefore they are more exposed to the attacks.

In the European Union, SME are a fundamental pillar of the economy, representing in September 2020 an astonishing 99,8% of all employing companies, 65% of private sector employment and 54% of the private sector. They have also suffered large economic losses and an even larger number of closures [GKÖPS20] due to the COVID-19 pandemic. Due to all of those factors, the main motivation for this study is the development of a ransomware detection model that meets the following requirements: High precision, reliability, scalability, easy to use, speed and open source software.

With this approach it is possible to carry out a prevention strategy against ransomware attacks and mitigate the impact of the damage caused by them, drastically reducing the expenses of the SME.

9.2. Objectives

At the beginning of this work, a series of objectives and guidelines were established to ensure its quality and correct development:

- Use the acquired skills during the academic degree.
- Develop a [ML](#) model capable of identifying ransomware samples with great precision.
- Create a large enough dataset to guarantee the reliability of the obtained results.
- Develop and deploy a laboratory to analyse the malware samples.
- Prove the effectiveness of the Windows [API](#) calls done by an executable as a ransomware detection mechanism.

9.3. Work Plan

The development of this work has been carried out in the following phases:

1. **Research:** This phase was carried out during the first four months to acquire the necessary information for the development of the work. At the beginning of this phase, several meetings were held where the objectives of the project and the necessary knowledge were explained, which was the fundamental concepts of machine learning. The main research tool used was *Google Scholar*, since it allows to search for scientific articles with ease and facilitates the citing. Articles and papers on detection of both ransomware and malware were searched, as well as books on malware analysis to gain a deeper understanding of the topic and the context of the study.
2. **Development:** The development of the model began once the necessary knowledge has been obtained. The investigation was not completely abandoned, but it was second priority. During this phase, the necessary data was obtained and the two *datasets* were built to train and evaluate the proposed model. For coding the model and implementing the [ML](#) algorithms, the programming language Python and the *scikit-learn* library were used.
3. **Experimentation and results:** In this phase the developed model with the extracted data was experimented on. It was necessary to evaluate the results obtained and to demonstrate the validity of the model, so the K-fold cross validation was used. The experimentation was vital for the development phase, since the model had to be modified to improve the results. The parameters were adjusted, more data was obtained...
4. **Documentation:** This phase was carried out in parallel to the rest of the phases. It started in the final months of the research phase and continued throughout the development and experimentation phases. All the members of the team participated in the writing of the documentation, working together in all of the sections. Any update had to be accepted and reviewed by the other team members.

The model developed in this work was uploaded to the Gitlab repository [TFG Ransomware 20-21](#)

Figure 9.5 is a Gantt chart where the different activities performed by the team throughout the months and their duration is shown.

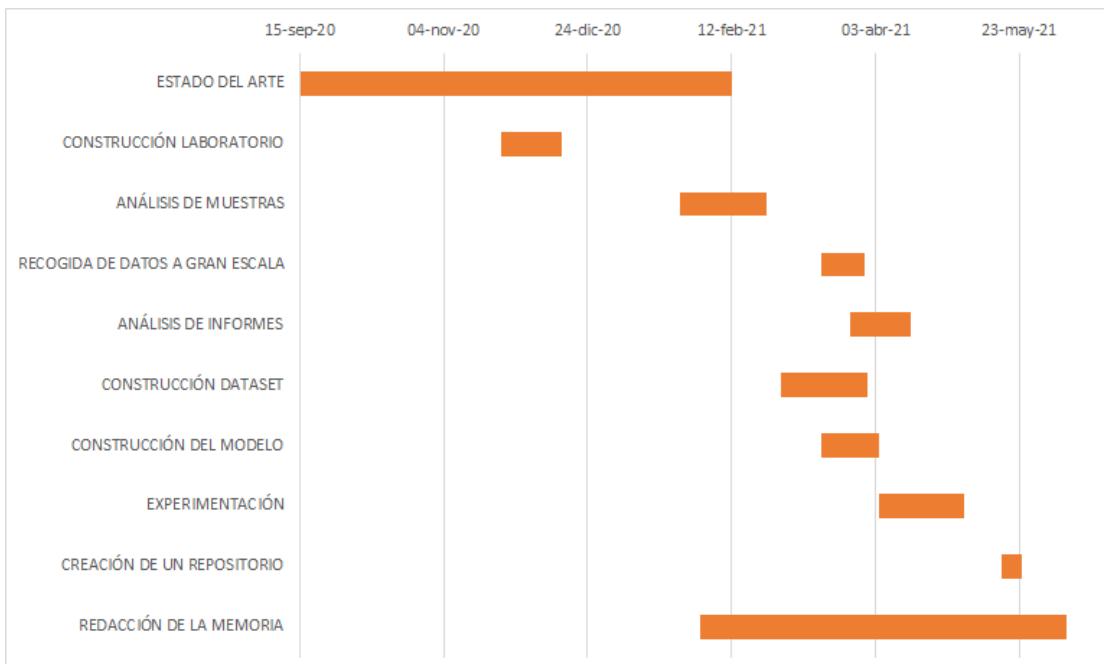


Figura 9.5: Gantt chart of the performed tasks.

9.4. Work Structure

The rest of the work is organised in 10 chapters, following this structure:

Chapter 2 introduces ransomware, explaining the fundamental concepts about malware and the different types, the business model that cybercriminals follow to profit from the attacks and the historical evolution of ransomware, detailing the different attacks that happened in the world each year. The different types of ransomware, the families, the most common spreading methods and subsequently the mode of operation to infect the victim's system are also explained. The chapter ends with the explanation of techniques used by antivirus programs to recognise the presence of ransomware, and with some prevention strategies and actions that every company should follow to protect their systems from these attacks, and if they are a victim of them, what steps they should follow to recover their data.

Chapter 3 addresses the types of malware analysis that exist and the limitations of each one, the tools used for the analysis and the different identification techniques. Machine learning applied to malware analysis is also discussed, exposing how a model is built using Python as a programming language and the different algorithms that exist. The supervised learning algorithms are the main focus since they are most commonly used in malware and ransomware detection and analysis.

Chapter 4 shows related works to the topic of malware and ransomware analysis and detection using ML models and algorithms. A distinction is made between papers that only analyse malware from those that analyse ransomware, and within the ransomware works, those that use Windows API calls to build their models are distinguished from those that do not.

Chapter 5 explains the technologies used to develop the proposed ML model, the sandbox environment and the process of obtaining the data. The flow diagram of the proposed system is shown and the development of the analysis laboratory for the study

of ransomware and goodware samples is discussed, from which the calls to the Windows API are extracted for the construction of the datasets. The data is then cleaned to discard those elements that are not relevant for the study. Finally, it is explained how the model will be created.

Chapter 6 discusses the experiments performed and the metrics used to demonstrate the effectiveness of the model to detect ransomware. After doing the experiments on the two datasets and showing the parameters used for their development, the final results are obtained and validated with K-fold cross validation. The algorithm that performs better than others is chosen and the results obtained with it are compared with the results of the related works presented in Chapter 4.

Chapter 7 shows the conclusions of the work after analysing the results, and exposes future lines of research.

Chapter 8 explains the individual contributions of each team member to the development of the work.

Chapters 9 and 10 are the English translations of Chapters 1 and 7.

Capítulo 10

Conclusions and Future Work

10.1. Conclusions

Ransomware attacks are becoming more and more common everyday and, consequently, more companies and organizations suffer them, in particular SME. These constitute a key pillar for the European economy and their resources destined to the prevention of cyber-attacks are more limited. Therefore, there is a need to develop an easy to use open source ML model able to protect these kinds of companies effectively. The model input data consists of the Windows API calls made by the executable samples. After carrying out this work, the following conclusions have been obtained:

- In malware detection field, the use of ML is essential to obtain optimal results, so we decided to create a ML model.
- In related work on malware and ransomware analysis and detection, the algorithms used in ML models are supervised learning algorithms.
- Windows API calls are a reliable attribute to analyze and detect ransomware, as a large number of projects use them as main features to build their models.
- Using the number of calls to each of the APIs to build a dataset provides better results than considering only their appearance with a binary representation. This is evidenced on the experiments, where two datasets are used, *dataset suma* and *dataset binario*, being *dataset suma* the one that obtains an improvement of around 10 % in the results.
- Sandboxing is a very reliable technique to run and analyze samples in a safe environment. Therefore, a malware analysis lab is built to examine these sample types, since they suppose a great risk to computers. To analyze the samples, we use a tool called Cuckoo Sandbox, which offers a multitude of settings to collect specific sample characteristics.
- Most related works on ransomware detection use relatively small datasets, compared to the number of samples used for ML at large. Therefore, in this work we have used a first dataset with a similar extension to those of the other works, obtaining a precision of 87 %, and a second dataset, considerably bigger, which obtains a precision of 98 %. This concludes that the proposed system obtains better results with a larger dataset than the other works using less data.

- In the experimentation, 7 [ML](#) algorithms have been used and [RF](#) has been the one that offers the best results when detecting ransomware samples, being better than other algorithms such as [DT](#), [SVM](#) or [NB](#).

10.2. Future Work

The possible research and development lines in the future are the following:

- Implementation of a client-server model including a consumer [API](#) or a graphical interface to make the use of the model easier. In this way, based on the [ML](#) model built, it would be possible to analyze files to detect the potential presence of ransomware.
- Include a signature-based system for ransomware detection, storing these signatures in a database accessed to compare the new samples analyzed signatures.
- Implement early detection of ransomware, finding its presence ahead of its execution.
- After the dataset preparation and cleaning, the number of final samples has been reduced, so it is convenient to continue expanding the dataset with samples from other sources, increasing the system's reliability.
- Consider other perspectives to improve ransomware detection. Among the possible options are analyzing network connections made by the sample, entropy of the system, monitoring the file system or [HPC](#) observation, as seen in other related works in Chapter [4](#)
- Consider malware samples and not just ransomware ones, creating a model with more application possibilities.

Apéndice A

Análisis de una Muestra con Cuckoo Sandbox

Configurada la herramienta, se podrán realizar análisis en el laboratorio construido utilizando Cuckoo Sandbox. Para ello, es necesario ejecutar una serie de comandos para iniciar correctamente el entorno cada vez que el sistema sea reiniciado:

1. En cada terminal que se abra para trabajar, iniciar el entorno virtual “cuckoo-test”:

```
$ workon cuckoo-test
```

2. Activar forwarding para la interfaz de red:

```
(cuckoo-test) $ sudo sysctl -w net.ipv4.conf.wlo1.forwarding=1
```

3. Crear interfaz con VMCloak:

```
(cuckoo-test) $ vmcloak-vboxnet0
```

4. Ejecutar Cuckoo Rooter:

```
(cuckoo-test) $ cuckoo rooter --sudo --group gonzalo
```

5. Ejecutar Cuckoo:

```
(cuckoo-test) $ cuckoo
```

6. Ejecutar Cuckoo Web:

```
(cuckoo-test) $ cuckoo web --host 127.0.0.1 --port 8080
```

Una vez ejecutados estos comandos, si se accede en el navegador a la dirección 127.0.0.1:8080 se encontrará la página principal de la interfaz web de Cuckoo Sandbox, como muestra la Figura A.1, donde aparece información de la versión de la herramienta,

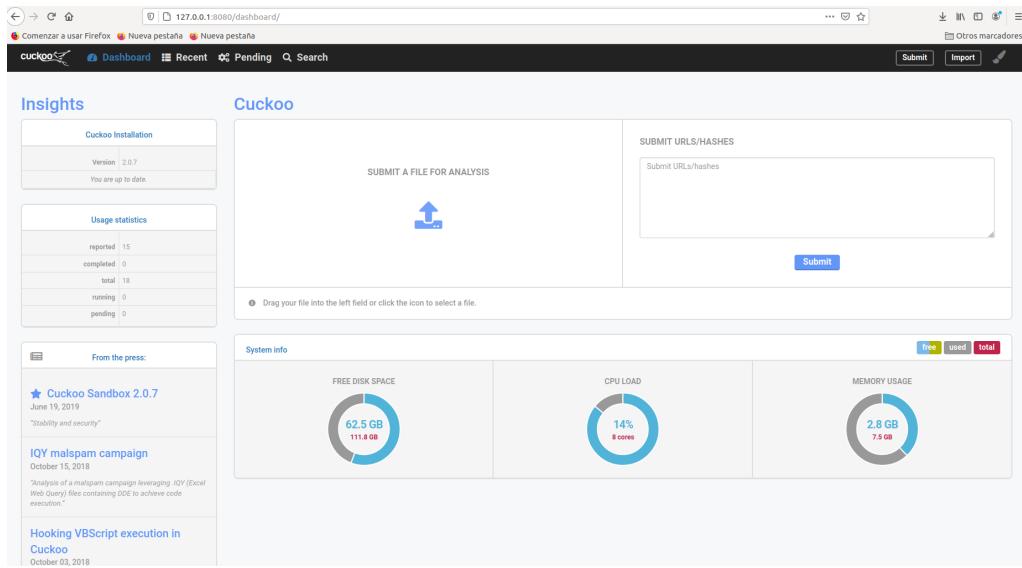


Figura A.1: Pantalla principal Cuckoo Sandbox

detalles sobre el uso de memoria y otros. Además, desde aquí será posible subir una muestra para su análisis.

Al subir un archivo de malware para analizar, es posible configurar las opciones del análisis para todas las muestras o personalizar cada una. Entre las opciones, la que más interesa es la de "Internet" dentro de "Network Routing", para permitir al proceso acceder a Internet durante la ejecución. En la Figura A.2 se pueden observar distintas opciones de configuración.

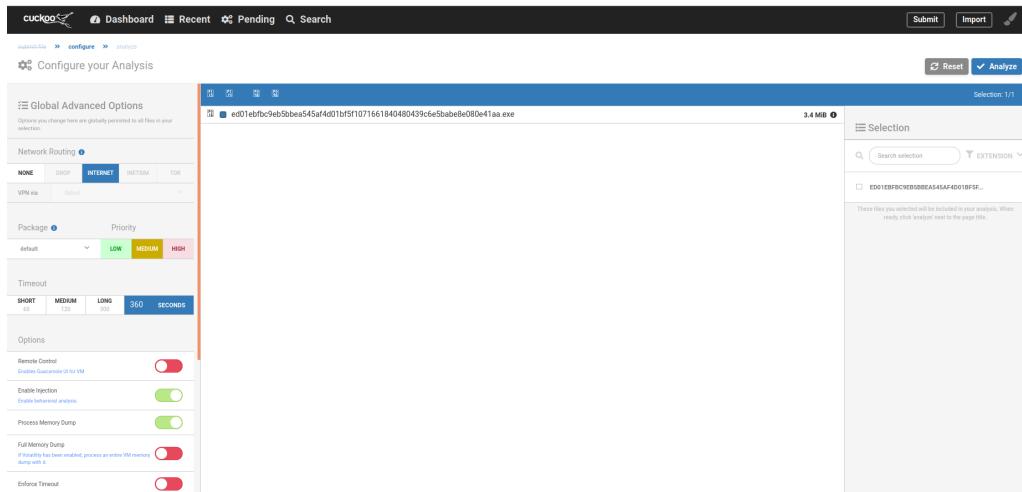


Figura A.2: Configuración del análisis

Después de configurar las opciones, pulsando en **Analyze** comenzará el análisis y se podrá observar cómo se abre la máquina virtual de Windows 7 y comienza la ejecución, viendo en todo momento qué va ocurriendo. Si se decide cambiar la configuración de Cuckoo para que no aparezcan las máquinas virtuales, no se podrá observer el comportamiento en directo de la muestra pero al haber instalado la herramienta `"pillow.en`

las máquinas virtuales, se realizarán capturas de pantalla de lo que está ocurriendo.

En las Figuras A.3 y A.4, se pueden observar capturas de pantalla realizadas durante el proceso de análisis que muestran un aviso de equipo infectado y una pantalla de instrucciones para el pago del rescate por la encriptación.

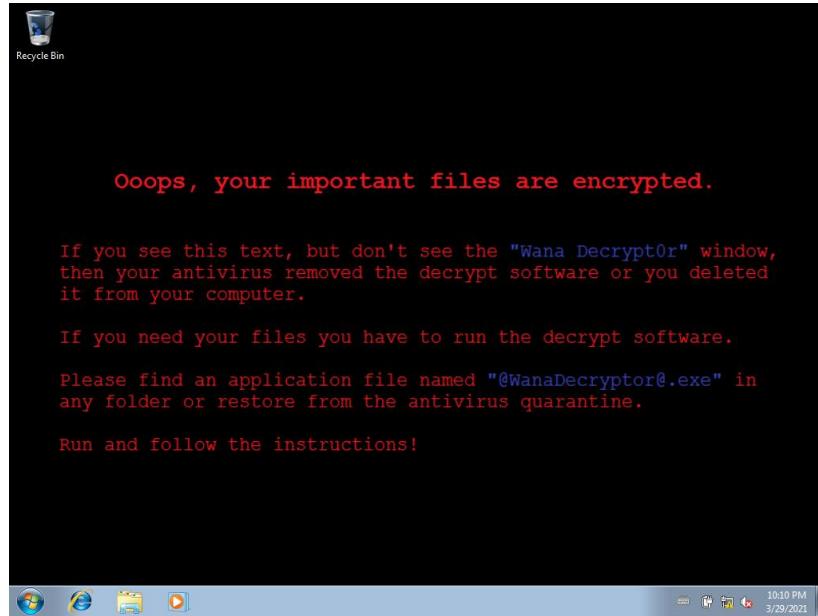


Figura A.3: Aviso de malware



Figura A.4: Pantalla de pago del rescate

Al terminar, la máquina virtual se cerrará y en el navegador se podrán visualizar los resultados del análisis, con la puntuación obtenida, detalles técnicos sobre la muestra, capturas de pantalla, información sobre los eventos y su peligrosidad y un menú lateral

para acceder a distintas características, entre otras cosas, tal y como se muestra en las Figuras A.5 y A.6.

The screenshot shows the Cuckoo Sandbox interface with the following details:

- Summary:** File ed01ebfb9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe
- Score:** 20.2 out of 10!
- Information on Execution:**
 - Category: FILE
 - Started: March 29, 2021, 9:10 p.m.
 - Completed: March 29, 2021, 9:17 p.m.
 - Duration: 445 seconds
 - Routing: Internet
 - Logs: Show Analyzer Log, Show Cuckoo Log
- Signatures:**
 - Queries for the computername (10 events)
 - Checks if process is being debugged by a debugger (5 events)
 - Command line console output was observed (11 events)

Figura A.5: Resultados de análisis 1

The screenshot shows the Cuckoo Sandbox interface with the following details:

- Suspicious Activities:**
 - Resumed a suspended thread in a remote process potentially indicative of process injection (2 events)
 - Uses suspicious command line tools or Windows utilities (3 events)
 - Installs Tor on the machine (4 events)
 - Performs 2148 file moves indicative of a ransomware file encryption process (50 out of 2148 events)
 - Appends a new file extension or content to 2148 files indicative of a ransomware file encryption process (50 out of 2148 events)
 - Connects to IP addresses that are no longer responding to requests (legitimate services will remain up-and-running usually) (13 events)
- Screenshots:** A grid of 16 screenshots showing various stages of the malware's activity on the victim machine.
- Table:** A table listing network connections and their status.

Name	Response	Post-Analysis Lookup	IP Address	Status	Action
www.msfnncai.com		2.21.26.96	131.188.40.189	Active	Moloch
dns.msfnncai.com		131.107.255.255	149.56.45.200	Active	Moloch
teredo.ipv6.microsoft.com			151.80.42.103	Active	Moloch
			154.35.175.225	Active	Moloch
			171.25.193.9	Active	Moloch

Figura A.6: Resultados de análisis 2

Toda esta información que se encuentra en la interfaz web, estará también en la carpeta “storage” dentro del directorio principal de trabajo de Cuckoo, organizada por carpetas para cada muestra, y repartida en sus correspondientes archivos de extensión “.log”, “.json”, “.pcap”...

Apéndice B

Lista Extendida de las Familias de Ransomware

A continuación, se expone una tabla B.1 con muchas de las familias de ransomware más conocidas [Mic20a]. No se listan todas, ya que además de haber una gran cantidad de ellas, hay infinidad de variantes que simplemente son copias de otras familias o son muy similares y no merece la pena nombrarlas.

Tabla B.1: Familias de ransomware.

Familia	Alias	Descripción
ACCDFISA	Anti Cyber Crime Department of Federal Internet Security Agency	Cifra archivos y bloquea la pantalla, después los atacantes solicitan el pago a través de Moneypak, Paysafe o Ukash. Está empaquetado como un archivo autoextraíble (<i>Self-extracting Archive (SFX)</i>) y puede venir incluido con aplicaciones de terceros como WinRAR [Sar12].
ANDROIDOS_LOCKER	Androidos_Locker	Fue el primer ransomware para móviles detectado. Utiliza TOR , un servicio que permite conexiones a servidor anónimas.
CRIBIT	BitCrypt	Cifra los archivos de la víctima mediante el cifrado RSA -AES. Existen dos tipos, el 1 y el 2. El tipo 1 usa cifrado RSA -426 y el tipo 2 RSA -1024. Agrega la cadena bitcrypt1 o bitcrypt2, dependiendo del tipo, al nombre de la extensión de los archivos que cifra.
CRILOCK	CryptoLocker	Cifra ciertos tipos de archivos usando una clave pública RSA y guarda la clave privada en los servidores C&C . Para conectarse con el servidor C&C , emplea el <i>Domain Generation Algorithm (DGA)</i> . En octubre de 2013 se descubrió que este ransomware es descargado por paquetes de malware troyanos como ZBOT que son enviados a través de correos electrónicos. [Sec15]
WANNACRYPT	WannaCry	Gusano criptográfico que explotaba la vulnerabilidad de Windows conocida como MS17-010. Se propagó a través de la red, infectando todo tipo de dispositivos y convirtiéndose en el ransomware más agresivo de 2017 [Kas21b].

Familia	Alias	Descripción
CRYPAURA	PayCrypt	Cifra archivos y agrega la dirección de correo electrónico de contacto para el descifrado como extensión de los archivos. Cada usuario tiene un correo distinto, siguiendo este formato: nombre del archivo.id-ID de víctima-paycrypt@aol.com. ⁷
CRYPCTB	Critroni, CTB Locker, Curve-Tor-Bitcoin Locker	Cifra los archivos y se asegura de que no haya recuperación posible de ellos al eliminar sus instantáneas (en inglés <i>shadow copies</i>). El usuario recibe un correo no deseado con un archivo adjunto, que en realidad es un programa que descarga de este ransomware. Utiliza TOR para ocultar sus comunicaciones con el servidor C&C .
CRYPWEB	PHP ransomware	Ataca a servidores web, cifrando sus bases de datos. Utiliza HTTPS para comunicarse con el servidor C&C .
CRYPTFILE	Cryptfile	Utiliza una clave pública única generada (RSA -2048) para el cifrado de archivos. Pide a los usuarios que paguen 1 <i>bitcoin</i> para obtener una clave privada que se usa para descifrar los archivos.
CRYPWALL	CryptoWall, CryptWall, CryptoWall 3.0, Cryptowall 4.0	Es la versión actualizada de CryptoDefense. Utiliza la red TOR con fines de anonimato y llega por correo no deseado, siguiendo la cadena de infección UPATRE-ZBOT-RANSOM. CryptoWall 3.0 contiene un software espía FAREIT. Cryptowall 4.0 también cifra el nombre de archivo y llega al usuario a través de spam como un archivo adjunto de JavaScript, y puede ser descargado por variantes de TROJ_KASIDET. [Gra14]
CRYPTOR	Batch file ransomware	Ransomware de archivos por lotes (<i>batch file</i>) capaz de cifrar los archivos del usuario mediante la aplicación GNU Privacy Guard.
CRYPTOSHIELD	CryptoShield	Cifra los archivos mediante el cifrado RSA -2048 y muestra un error falso de la aplicación Explorer.exe. Se distribuye a través del <i>exploit kit</i> RIG. [McA17]
PGPCODER	Pgpcoder	Fue descubierto en 2005. Crea dos claves de registro, una para asegurarse de que el ransomware continua su ejecución aun habiendo reiniciado el ordenador y otra para contar el número de archivos infectados. Cuando termina de cifrar todos los archivos que encuentra, le facilita al usuario un correo para que pague \$100–200 por el rescate.
CRYYSIS	Crysis	Ransomware de tipo Filecoder que utiliza los cifrados RSA y AES con llaves de cifrado largas. Esto hace prácticamente imposible la recuperación de los archivos cifrados. [Per16]
KOVTER	Kovter	Este ransomware se activa mediante una macro de un documento Word enviado al usuario a través de un correo electrónico. Si se habilitan las macros en el documento, se descargará un archivo que crea un comando de <i>PowerShell</i> y lo almacena en el registro para que sea persistente. Posteriormente, este archivo se elimina para que no quede rastro. [Spa20]

Familia	Alias	Descripción
MATSNU	Matsnu	Es un <i>backdoor</i> (un tipo de troyano que permite el acceso al sistema infectado y su control remoto) que puede bloquear el ordenador y pedir un rescate para su desbloqueo. [Mic14]
REVETON	Police Ransom	Ransomware de tipo Virus de la policía, explicado en la sección anterior. Bloquea el sistema y se hace pasar por una advertencia de la policía, exigiendo que el usuario pague una multa por un delito falso.
VBUZKY	Vbuzky	Utiliza la inyección Shell_TrayWnd y habilita la opción TESTSIGNING de Windows 7.
CRYPTLOCK	TorrentLocker	Se hace pasar por CryptoLocker, mostrándose al usuario como 'crypt0l0cker'. Utiliza un cifrado AES para cifrar los archivos y un cifrado asimétrico RSA para cifrar la clave AES. [Lab21b]
CRYPDIRT	Dirty Decrypt	Bloquea el ordenador del usuario y crea una clave en el registro del sistema llamada 'DirtyDecrypt' con el valor 'DirtyDecrypt.exe'. Pide \$100 para desbloquear el ordenador.
CRYPTESLA	TeslaCrypt	La pantalla que se muestra al usuario es similar a la de CryptoLocker. Cifra archivos relacionados con juegos: perfiles, partidas guardadas, etc. Este ransomware no cifra archivos con un tamaño superior a 268 MB. Las versiones 2.1 y 2.2 añaden a los archivos cifrados las extensiones .vvv y .ccc. La versión 3.0 tiene un algoritmo de cifrado mejorado y agrega .xxx, .ttt y .mp3 a los archivos que cifra. [Lab21a]
CRYPVAULT	VaultCrypt	Utiliza la herramienta de cifrado GNU Privacy Guard. Descarga de herramientas para robar credenciales almacenadas en navegadores web y utiliza sDelete 16 veces para prevenir la recuperación de archivos.
CRYPSHED	Troldesh	Fue visto por primera vez en Rusia. Además de agregar la extensión .xtbl a los archivos cifrados, también codifica sus nombres, lo que hace que los usuarios no sepan que archivos han sido afectados.
SYNOLOCK	SynoLocker	Aprovecha el sistema operativo de los dispositivos Synology <i>Network Attached Storage (NAS)</i> (<i>DiskStation Manager (DSM)</i> 4.3-3810 o anterior) para cifrar sus archivos.
KRYPTOVOR	CryptInfinite, DecryptorMax	Utiliza una biblioteca Delphi llamada LockBox 3 para cifrar archivos.
CRYPFIRAGO	Crypfirago	Cifra archivos y les agrega las extensiones .1999 o .bleep. Utiliza Bitmessage para comunicarse con los delincuentes.
CRYPRADAM	Radamant	Agrega .rpm a los archivos que cifra. Se difunde mediante correos no deseados al abrir los documentos PDF o Word adjuntos a estos. [Kig15]
CRYPTRITU	Ransom32	Fue el primer ransomware desarrollado en JavaScript y afecta a todos los sistemas operativos por igual. [Med16]

Familia	Alias	Descripción
CRYPBOSS	CrypBoss	Llega al usuario como correo no deseado. Añade .crypt a los archivos cifrados y usa el cifrado AES . También elimina las <i>Shadow Volume Copies</i> , conocidas como instantáneas, que sirven para restaurar el sistema a un punto anterior. [McA16a]
LOCKY	Locky	Cambia el nombre de los archivos cifrados a valores hexadecimales y les agrega la extensión .locky. Se distribuye a través de correo no deseado con un documento adjunto integrado en una macro.
CRYPHYDRA	HydraCrypt	Distribuido mediante el <i>Angler exploit kit</i> , un programa que se aprovecha de las vulnerabilidades en el lado del cliente (Java, Flash, PDFs, etc). Amenaza al usuario con vender sus archivos en la Dark Web si no paga el rescate. [McA16b]
CERBER	Cerber	Cifra los archivo, cambiándoles el nombre y añadiéndoles la extensión .cerber. Se distribuye mediante correos no deseados o mediante la descarga de archivos infectados en páginas web. Contiene archivos de audio para hablar con la víctima, además de instrucciones para pagar el rescate. [Bel20]
JIGSAW	Jigsaw	Ransomware de tipo Wiper que elimina los archivos del ordenador infectado y aumenta la cantidad a pagar por el rescate cada hora. Algunas variantes tienen soporte de chat en vivo para hablar sus víctimas.
PETYA	Petya	Cifra archivos y puede bloquea el disco duro entero, impidiendo que el equipo arranque y provocando una pantalla azul. Al reiniciar el sistema, se muestra un mensaje informando al usuario de la situación y del rescate. [Bel19]
WALTRIX	CRYPTXXX, WALTRIX, Exxroute	Como HydraCrypt, es distribuido mediante el <i>Angler exploit kit</i> como un fichero DLL . Bloquea el sistema y cifra todos los archivos, añadiéndoles la extensión .crypt.
JSRAA	RAA	Escrito en JScript y diseñado para el motor <i>Windows Scripting Host</i> en Internet Explorer. No se ejecuta en el navegador Microsoft Edge.
RYUK	Ryuk	Este ransomware está especializado en atacar empresas, y es especialmente peligroso ya que puede distribuirse por una red LAN privada sin importar que los ordenadores estén apagados. Esto lo consigue enviando paquetes WoL (Wake-on-LAN). Cifra los archivos del sistema y usa una clave pública RSA , además de eliminar las copias de seguridad del sistema. En la nota que deja el ransomware, no viene la cantidad a pagar sino una dirección de correo electrónico y una billetera de <i>bitcoin</i> . La víctima tiene que ponerse en contacto con los atacantes para negociar el rescate y este varía dependiendo de la empresa afectada. [Gon20]

Familia	Alias	Descripción
MAZE	Maze	Al igual que Ryuk, es un ransomware que se dedica a atacar empresas. Cifra los archivos y amenaza con publicarlos y venderlos si la empresa no paga el rescate. Este ransomware se distribuye mediante correos no deseados con archivos adjuntos infectados o a través de kits de exploits (Fallout EK y Spelevo EK). [FS20]
SAMSAM	SamSam	Los ataques de este ransomware son manuales, por lo que los objetivos suelen ser empresas para ganar más dinero. Estos ataques se ejecutan en remoto, explotando vulnerabilidades en RDP, servidores <i>File Transfer Protocol (FTP)</i> y servidores Java para obtener acceso a la red de la víctima. Cifra los archivos del sistema usando RSA-2048 y pide un rescate en bitcoins. [Boy18]
BADRABBIT	Bad Rabbit	Es un ejemplo de ransomware de tipo 'dropper' que se mencionó anteriormente. Se disfraza como una instalación de Adobe Flash y se propaga a través de descargas automáticas en sitios web vulnerados. Bloque el ordenador y pide un rescate de \$280 en bitcoins con un plazo de 40 horas. [Pro19]
GANDCRAB	GandCrab	Distribuido por la <i>DarkNet</i> siguiendo el modelo <i>RaaS</i> , este ransomware cifra los archivos del sistema y pide un rescate en <i>Dash</i> (una criptomoneda como <i>bitcoin</i>). [Mal19b]

Bibliografía

- [A18] Monnappa K A. *Learning Malware Analysis*. Packt Publishing Ltd., Livery Place, 35 Livery Street, Birmingham, UK, 1 edition, 6 2018.
- [AAB⁺15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems, 2015.
- [AAII17] Sana Aurangzeb, Muhammad Aleem, Muhammad Iqbal, and Arshad Islam. Ransomware: A survey and trends. *Journal of Information Assurance and Security (ESCI - Thomson Reuters Indexed)*, ISSN: 1554-101, 12:48–58, June 2017.
- [ABD18] Omar MK Alhawi, James Baldwin, and Ali Dehghantanha. Leveraging machine learning techniques for windows ransomware network traffic detection. In *Cyber Threat Intelligence*, pages 93–106. Springer, 2018.
- [Abr20] Lawrence Abrams. Sodinokibi ransomware says travelex will pay, one way or another. <https://www.bleepingcomputer.com/news/security/sodinokibi-ransomware-says-travelex-will-pay-one-way-or-another/>, January 2020.
- [ADC18] A. Azmoodeh, A. Dehghantanha, and M. Conti. Detecting crypto-ransomware in iot networks based on energy consumption footprint. *Journal of Ambient Intelligence and Humanized Computing*, 9, 2018.
- [ADPC20a] Abdullahi Arabo, Remi Dijoux, Timothee Poulain, and Gregoire Chevalier. Detecting ransomware using process behavior analysis. *Procedia Computer Science*, 168:289–296, 2020.
- [ADPC20b] Abdullahi Arabo, Remi Dijoux, Timothee Poulain, and Gregoire Chevalier. Detecting ransomware using process behavior analysis. *Procedia Computer Science*, 168:289–296, 2020. “Complex Adaptive Systems”Malvern, PennsylvaniaNovember 13-15, 2019.
- [Aha13] David W Aha. *Lazy learning*. Springer Science & Business Media, 2013.
- [AHSF09a] Faraz Ahmed, Haider Hameed, M Zubair Shafiq, and Muddassar Farooq. Using spatio-temporal information in api calls with machine learning algorithms for malware detection. In *Proceedings of the 2nd ACM Workshop on Security and Artificial Intelligence*, pages 55–62, 2009.
- [AHSF09b] Faraz Ahmed, Haider Hameed, M. Zubair Shafiq, and Muddassar Farooq. Using spatio-temporal information in api calls with machine learning algorithms for malware detection. In *Proceedings of the 2nd ACM Workshop on Security*

- and Artificial Intelligence*, AISeC '09, page 55–62, New York, NY, USA, 2009. Association for Computing Machinery.
- [AKRR20] Amiruddin Amiruddin, Candra Kurniawan, Eka Hero Ramadhani, and Julio Rinaldi. Learning the basic strcuture of several ransomwares using static analysis tecgnique. *IOP Conference Series: Materials Science and Engineering*, 1007:012072, December 2020.
- [AMK16] Azad Ali, Raj Murthy, and Frederick Kohun. Recovering from the nightmare of ransomware-how savvy users get hit with viruses and malware: A personal case study. *Issues in Information Systems*, 17(4), 2016.
- [AP19] Ricardo van Zutphen Alwin Peppels. Cuckoo Sandbox Setup for People in a Hurry. <https://hatching.io/blog/cuckoo-sandbox-setup/>, Jul 2019.
- [ÁR] Sergio Álvarez Rubio. Despliegue de la herramienta”zeekz su posterior explotación para el análisis de actividades sospechosas en la red.
- [ARA⁺21] Sana Aurangzeb, Rao Naveed Bin Rais, Muhammad Aleem, Muhammad Arshad Islam, and Muhammad Azhar Iqbal. On the classification of microsoft-windows ransomware using hardware profile. *PeerJ Computer Science*, 7:e361, February 2021.
- [AVL19] Maxat Akbanov, Vassilios G. Vassilakis, and Michael D. Logothetis. WannaCry ransomware: Analysis of infection, persistence, recovery prevention and propagation mechanisms. *Journal of Telecommunications and Information Technology*, 1:113–124, March 2019.
- [AVWA11] Mamoun Alazab, Sitalakshmi Venkatraman, Paul Watters, and Moutaz Alazab. Zero-day malware detection based on supervised learning algorithms of api call signatures. In *Proceedings of the Ninth Australasian Data Mining Conference - Volume 121*, AusDM '11, page 171–182, AUS, 2011. Australian Computer Society, Inc.
- [Ayc06] John Daniel Aycock. *Computer viruses and malware*. Springer, 2006.
- [Ayo10] Taiwo Oladipupo Ayodele. Types of machine learning algorithms. *New advances in machine learning*, 3:19–48, 2010.
- [Bak20] Kurt Baker. Malware analysis. <https://www.crowdstrike.com/cybersecurity-101/malware/malware-analysis/>, March 2020.
- [Bel19] Ivan Belcic. Ransomware petya: Cómo funciona y cómo protegerse. <https://www.avast.com/es-es/c-petya>, 2019.
- [Bel20] Ivan Belcic. Ransomware cerber: Todo lo que necesita saber. <https://www.avast.com/es-es/c-cerber>, 2020.
- [BLB⁺13] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.
- [BLI19] Seong Il Bae, Gyu Bin Lee, and Eul Gyu Im. Ransomware detection using machine learning algorithms. *Concurrency and Computation: Practice and Experience*, 32(18):e5422, 2019.
- [Boy18] Christopher Boyd. Samsam ransomware: what you need to know. <https://blog.malwarebytes.com/cybercrime/2018/05/samsam-ransomware-need-know/>, May 2018.
- [Bra17] Patrick O Branche. *Ransomware: An Analysis of the Current and Future Threat Ransomware Presents*. Utica College, 2017.

- [Bre01] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
- [Bre15] Jurriaan Bremer. VMCloak Documentation. <https://vmcloak.readthedocs.io/en/latest/>, Jan 2015.
- [Bro16] Jason Brownlee. Logistic regression for machine learning. <https://machinelearningmastery.com/logistic-regression-for-machine-learning/>, April 2016.
- [Bro19] Jason Brownlee. Information gain and mutual information for machine learning. <https://machinelearningmastery.com/information-gain-and-mutual-information/>, October 2019.
- [BSTPS16] Mihai Barbulescu, Adrian Stratulat, Vlad Traista-Popescu, and Emil Simion. Rsa weak public keys available on the internet. In *Innovative Security Solutions for Information Technology and Communications: 9th International Conference, SECITC 2016, Bucharest, Romania, June 9-10, 2016, Revised Selected Papers*, volume 10006, page 92. Springer, 2016.
- [BSVB14] Vanesa Berlanga Silvente and Ruth Vilà Baños. Cómo obtener un modelo de regresión logística binaria con spss. *REIRE. Revista d'Innovació i Recerca en Educació, 2014, vol. 7, num. 2*, 2014.
- [Bur98] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.
- [Can14] David Cantón. Ransomware iii: Variante lock screen. <https://www.incibe-cert.es/blog/ransomware-lock-screen>, Enero 2014.
- [Cat19] Simone Catania. Los 10 ransomware más peligrosos de los últimos años. <https://www.noticias.ltda/seguridad-informatica/10-ransomware-mas-peligrosos/>, 2019.
- [CC08] Matthieu Cord and Pádraig Cunningham. *Machine learning techniques for multimedia: case studies on organization and retrieval*. Springer Science & Business Media, 2008.
- [Ccn18] Medidas de seguridad contra ransomware. Technical report, Centro Criptológico Nacional, CCN-CERT, May 2018.
- [CCS12] Adele Cutler, D Richard Cutler, and John R Stevens. Random forests. In *Ensemble machine learning*, pages 157–175. Springer, 2012.
- [Che14] Galoget Jontze Latorre Cheng. Análisis estático y dinámico de una muestra de malware en sistemas microsoft windows xp para determinar qué efectos produce sobre un sistema infectado. Master's thesis, Escuela Politécnica Nacional, Av. Ladrón de Guevara 253, Quito 170517, Ecuador, Octubre 2014.
- [Che18] Chiheb Chebbi. *Mastering Machine Learning for Penetration Testing*. Packt Publishing Ltd., 1 edition, 6 2018.
- [CKYK17] Zhi-Guo Chen, Ho-Seok Kang, Shang-Nan Yin, and Sung-Ryul Kim. Automatic ransomware detection and analysis based on dynamic api calls flow graph. In *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*, pages 196–201, 2017.
- [CL07] Ken Chiang and Levi Lloyd. A case study of the rustock rootkit and spam bot. *HotBots*, 7(10-10):7, 2007.
- [CMK18] Greg Cusack, Oliver Michel, and Eric Keller. Machine learning-based detection of ransomware using sdn. In *Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, pages 1–6, 2018.

- [Col18] Carlos Estrada Cola. Estudio sobre el malware Ransomware. Master's thesis, Universidad Abierta de Cataluña, 2018.
- [Com19] BBVA Communications. Machine learning: What is it and how does it work? <https://www.bbva.com/en/machine-learning-what-is-it-and-how-does-it-work/>, November 2019.
- [com20] The SciPy community. *NumPy v1.20 Manual*. SciPy, 2020.
- [Cov20] Coveware. Ransomware demands continue to rise as data exfiltration becomes common, and maze subdues. <https://www.coveware.com/blog/q3-2020-ransomware-marketplace-report>, November 2020.
- [Cry20] Crypsis. 2020 incident response & data breach report. Technical report, Crypsis, June 2020.
- [Dat17] DataFlair. Kernel functions-introduction to svm kernel & examples. <https://data-flair.training/blogs/svm-kernel-functions/>, August 2017.
- [DDTV⁺17a] Anusha Damodaran, Fabio Di Troia, Corrado Aaron Visaggio, Thomas H Austin, and Mark Stamp. A comparison of static, dynamic, and hybrid analysis for malware detection. *Journal of Computer Virology and Hacking Techniques*, 13(1):1–12, 2017.
- [DDTV⁺17b] Anusha Damodaran, Fabio Di Troia, Corrado Aaron Visaggio, Thomas H Austin, and Mark Stamp. A comparison of static, dynamic, and hybrid analysis for malware detection. *Journal of Computer Virology and Hacking Techniques*, 13(1):1–12, 2017.
- [DK17] Catherine Dwyer and Ameet Kanguri. Malvertising-a rising threat to the online ecosystem. *Journal of Information Systems Applied Research*, 10(3):29, 2017.
- [DO13] Iqbal Muhardianto Digit Oktavianto. *Cuckoo Malware Analysis*. Packt Publishing Ltd., October 2013.
- [Don14] Brian Donohue. ¿qué es el “malvertising”? <https://www.kaspersky.es/blog/que-es-el-malvertising/4214/>, Septiembre 2014.
- [Don18] Fred Donovan. Samsam ransomware attacks net creator \$6m so far. <https://healthitsecurity.com/news/samsam-ransomware-attacks-net-creator-6m-so-far>, August 2018.
- [Don19] Niklas Donges. A complete guide to the random forest algorithm. <https://builtin.com/data-science/random-forest-algorithm>, June 2019.
- [DRP⁺12] T. Dube, R. Raines, G. Peterson, K. Bauer, M. Grimalia, and S. Rogers. Malware target recognition via static heuristics. *Computers & Security*, 31(1):137–147, 2012.
- [Eag11] Chris Eagle. *The IDA Pro Book: The Unofficial Guide to the World’s Most Popular Disassembler*. Penguin Random House LLC., 2 edition, 7 2011.
- [EKK⁺07] Manuel Egele, C. Krügel, E. Kirda, Heng Yin, and D. Song. Dynamic spyware analysis. In *USENIX Annual Technical Conference*, 2007.
- [Erb05] Michael Erbschloe. *TROJANS, WORMS and SPYWARE, A computer security professional’s guide to malicious code*. Elsevier Inc, 2005.
- [ESKK08] Manuel Egele, Theodoor Scholte, Engin Kirda, and Christopher Kruegel. A survey on automated dynamic malware-analysis techniques and tools. *ACM Comput. Surv.*, 44(2), March 2008.
- [Fir17] FireEye. Floss. <https://www.fireeye.com/services/freeware/floss.html>, May 2017.
- [fN20] Steve Symanovich for NortonLifeLock. Coronavirus phishing emails: How to protect against COVID-19 scams. <https://us.norton.com/internetsecurity-online-scams-coronavirus-phishing-scams.html>, Marzo 2020.

- [Fou] Cuckoo Foundation. Cuckoo Sandbox Book. Technical Report, Cuckoo Foundation.
- [Fre19] Jonathan Freeman. What is an api? application programming interfaces explained. <https://www.infoworld.com/article/3269878/what-is-an-api-application-programming-interfaces-explained.html>, August 2019.
- [Fru20] Josh Fruhlinger. What is phishing? how this cyber attack works and how to prevent it. <https://www.cscoonline.com/article/2117843/what-is-phishing-how-this-cyber-attack-works-and-how-to-prevent-it.html>, September 2020.
- [FS20] Nikita Galimov Fedor Sinitsyn, Vladimir Kuskov. El ransomware maze. <https://securelist.lat/maze-ransomware/91660/>, October 2020.
- [GBS14] Ekta Gandotra, Divya Bansal, and Sanjeev Sofat. Malware analysis and classification: A survey. *Journal of Information Security*, 05(02):56–64, 2014.
- [GKÖPS20] Pierre-Olivier Gourinchas, Sebnem Kalemli-Özcan, Veronika Penciakova, and Nick Sander. Covid-19 and sme failures. Working Paper 27877, National Bureau of Economic Research, September 2020.
- [Gon20] Yolanda González. Ryuk, el ransomware más temido por las empresas. <https://protecciondatos-1opd.com/empresas/ryuk-ransomware>, Julio 2020.
- [Gra14] Grafix. Cryptowall, un nuevo virus que circula per internet que encripta todos los archivos de tu ordenado. <https://grafix.es/virus-cryptowall/>, Julio 2014.
- [Gro19] Velorcios Group. ¿sabías que el ransomware tiene sus inicios en 1989? <https://velorciosgroup.com/ransomware-tiene-sus-inicios-en-1989/>, 2019.
- [Gro20] Juliana De Groot. A history of ransomware attacks: The biggest and worst ransomware attacks of all time. <https://digitalguardian.com/blog/history-ransomware-attacks-biggest-and-worst-ransomware-attacks-all-time>, December 2020.
- [Gru21] Grupo de Análisis, Seguridad y Sistemas. XXXXX. Technical Report, Universidad Complutense de Madrid, May 2021.
- [GSBTLR⁺17] Pablo L. Gallegos-Segovia, Jack F. Bravo-Torres, Victor M. Larios-Rosillo, Paul E. Vintimilla-Tapia, Ivan F. Yuquilima-Albarado, and Juan D. Jara-Saltos. Social engineering as an attack vector for ransomware. In *2017 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*. IEEE, October 2017.
- [Gua16] The Guardian. Ransomware threat on the rise as 'almost 40 % of businesses attacked'. <https://www.theguardian.com/technology/2016/aug/03/ransomware-threat-on-the-rise-as-40-of-businesses-attacked>, Agosto 2016.
- [Har18] Onel Harrison. Machine learning basics with the k-nearest neighbors algorithm. <https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761>, September 2018.
- [Has19a] Nihad A. Hassan. *Ransomware revealed : a beginner's guide to protecting and recovering from ransomware attacks*. Apress, 2019.
- [Has19b] Nihad A. Hassan. *Ransomware Revealed: A Beginner's Guide To Protecting And Recovering From Ransomware Attacks*. Apress, Berkeley, CA, New York, 2019.
- [HE20] Thomas Hungenberg and Matthias Eckert. Inetsim: Internet services simulation suite. <https://www.inetsim.org/about.html>, May 2020.

- [HJA19] Gavin Hull, Henna John, and Budi Arief. Ransomware deployment methods and analysis: views from a predictive model and human responses. *Crime Science*, 8(1):1–22, 2019.
- [HJAP21] Mamoon Humayun, NZ Jhanjhi, Ahmed Alsayat, and Vasaki Ponnusamy. Internet of things and ransomware: Evolution, mitigation and prevention. *Egyptian Informatics Journal*, 22(1):105–117, 2021.
- [HKLK20a] Jinsoo Hwang, Jeankyung Kim, Seunghwan Lee, and Kichang Kim. Two-stage ransomware detection using dynamic analysis and machine learning techniques. *Wireless Personal Communications*, 112, 2020.
- [HKLK20b] Jinsoo Hwang, Jeankyung Kim, Seunghwan Lee, and Kichang Kim. Two-stage ransomware detection using dynamic analysis and machine learning techniques. *Wireless Personal Communications*, 112(4):2597–2609, 2020.
- [Hol20] Keiron Holyome. The landscape of cyber security in the nhs – and how this has changed since the wannacry attack in 2017. <https://www.ns-healthcare.com/analysis/wannacry-ransomware-nhs/>, March 2020.
- [Hun07] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.
- [Inc20] Incibe. Dropper, la amenaza silenciosa. <https://www.incibe.es/protege-tu-empresa/blog/dropper-amenaza-silenciosa>, Agosto 2020.
- [Jet19] Brijesh Jethva. *A new ransomware detection scheme based on tracking file signature and file entropy*. PhD thesis, 2019.
- [JP21] Xataka Javier Pastor. Así es Ryuk, el ransomware que ha dejado tumbado al SEPE (y que antes tumbó a otros muchos). <https://www.xataka.com/seuridad/asi-ryuk-ransomware-que-ha-dejado-tumbado-al-sepe-que-antes-tumbo-a-otros-muchos>, Marzo 2021.
- [JWHT13] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.
- [KA19] İlker Kara and Murat Aydos. The ghost in the system: technical analysis of remote access trojan. *International Journal on Information Technologies & Security*, 11(1):73–84, 2019.
- [KAJ20] S.H. Kok, Azween Abdullah, and NZ Jhanjhi. Early detection of crypto-ransomware using pre-encryption detection algorithm. *Journal of King Saud University - Computer and Information Sciences*, July 2020.
- [KAJS19] S. Kok, Azween Abdullah, NZ Jhanjhi, and Mahadevan Supramaniam. Prevention of crypto-ransomware using a pre-encryption detection algorithm. *Computers*, 8(4):79, November 2019.
- [KAM⁺16a] Amin Kharaz, Sajjad Arshad, Collin Mulliner, William Robertson, and Engin Kirda. Unveil: A large-scale, automated approach to detecting ransomware. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 757–772, 2016.
- [KAM⁺16b] Amin Kharaz, Sajjad Arshad, Collin Mulliner, William Robertson, and Engin Kirda. UNVEIL: A large-scale, automated approach to detecting ransomware. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 757–772, Austin, TX, August 2016. USENIX Association.
- [Kar18] Md. Rezaul Karim. *Scala Machine Learning Projects*. Packt Publishing Ltd., 1 edition, 1 2018.
- [Kas16] Kaspersky. Ksn report: Pc ransomware in 2014-2016. <https://securelist.com/pc-ransomware-in-2014-2016/75145/>, June 2016.

- [Kas19] Kaspersky. Ksn report: Ransomware and malicious cryptominers 2016-2018. Technical report, Kaspersky, February 2019.
- [Kas21a] Kaspersky. Brute force attack: Definition and examples. <https://www.kaspersky.com/resource-center/definitions/brute-force-attack>, January 2021.
- [Kas21b] Kaspersky. What is wannacry ransomware? <https://www.kaspersky.com/resource-center/threats/ransomware-wannacry>, January 2021.
- [Kas21c] Kaspersky. ¿cuáles son los diferentes tipos de ransomware? <https://www.kaspersky.es/resource-center/threats/ransomware-examples>, Enero 2021.
- [Kas21d] Kaspersky. ¿qué es scareware?— definición de filtro web. <https://latam.kaspersky.com/resource-center/definitions/scareware>, Enero 2021.
- [KDM15] Kesav Kancherla, John Donahue, and Srinivas Mukkamala. Packer identification using byte plot and markov plot. *Journal of Computer Virology and Hacking Techniques*, 12(2):101–111, September 2015.
- [Kig15] Ugnius Kiguolis. Eliminar el virus radamant (instrucciones de eliminación). <https://losvirus.es/radamant-ransomware/>, Diciembre 2015.
- [KK17] Amin Kharraz and Engin Kirda. Redemption: Real-time protection against ransomware at end-hosts. In Marc Dacier, Michael Bailey, Michalis Polychronakis, and Manos Antonakakis, editors, *Research in Attacks, Intrusions, and Defenses*, pages 98–119, Cham, 2017. Springer International Publishing.
- [KKB⁺06] Engin Kirda, Christopher Kruegel, Greg Banks, Giovanni Vigna, and Richard Kemmerer. Behavior-based spyware detection. In *Usenix Security Symposium*, page 694, 2006.
- [KM06] J Zico Kolter and Marcus A Maloof. Learning to detect and classify malicious executables in the wild. *Journal of Machine Learning Research*, 7(12), 2006.
- [KNR⁺20] Firoz Khan, Cornelius Ncube, Lakshmana Kumar Ramasamy, Seifedine Kadry, and Yunyoung Nam. A digital dna sequencing engine for ransomware detection using machine learning. *IEEE Access*, 8:119710–119719, 2020.
- [KRB⁺15] Amin Kharraz, William Robertson, Davide Balzarotti, Leyla Bilge, and Engin Kirda. Cutting the gordian knot: A look under the hood of ransomware attacks. In Magnus Almgren, Vincenzo Gulisano, and Federico Maggi, editors, *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 3–24, Cham, 2015. Springer International Publishing.
- [Kur15] Thorsten Kurpjuhn. The sme security challenge. *Computer Fraud & Security*, 2015(3):5–7, 2015.
- [Lab19] Malwarebytes Labs. Europol: Ransomware remains top threat in iocta report. <https://blog.malwarebytes.com/awareness/2019/10/europol-ransomware-remains-top-threat-in-iocta-report/>, Octubre 2019.
- [Lab21a] Kaspersky Lab. Ataques de ransomware teslacrypt. <https://www.kaspersky.es/resource-center/threats/teslacrypt>, 2021.
- [Lab21b] Kaspersky Lab. Ransomware torrentlocker. <https://latam.kaspersky.com/resource-center/threats/torrentlocker-malware>, 2021.
- [Li20] Vickie Li. Intro to malware detection using yara. <https://infosecwriteups.com/intro-to-malware-detection-using-yara-eacab8373cf4>, February 2020.
- [Lia20] Han Liang. Qué es una dll. <https://docs.microsoft.com/es-es/troubleshoot/windows-client/deployment/dynamic-link-library>, 9 2020.

- [LMMT21] Magno Logan, Erika Mendoza, Ryan Maglaque, and Nikko Tamaña. The state of ransomware 2020's catch-22. <https://www.trendmicro.com/vinfo/in/security/news/cybercrime-and-digital-threats/the-state-of-ransomware-2020-s-catch-22>, February 2021.
- [Lom19] Mark Loman. How ransomware attacks. Technical report, SophosLabs, November 2019.
- [Lor20] Nate Lord. What are indicators of compromise? <https://digitalguardian.com/blog/what-are-indicators-compromise>, December 2020.
- [LSN⁺21] Harjinder Singh Lallie, Lynsay A. Shepherd, Jason R.C. Nurse, Arnau Erola, Gregory Epiphaniou, Carsten Maple, and Xavier Bellekens. Cyber security in the age of covid-19: A timeline and analysis of cyber-crime and cyber-attacks during the pandemic. *Computers & Security*, 105:102248, 2021.
- [Lá18] Diego Lázaro. Codificación de caracteres en programación. <https://diego.com.es/codificacion-de-caracteres-en-programacion>, 2018.
- [Mal18] Malwarebytes. Ransomware. <https://es.malwarebytes.com/ransomware/>, November 2018.
- [Mal19a] Malwarebytes. Cybercrime tactics and techniques: Q2 2018. Technical report, Malwarebytes, February 2019.
- [Mal19b] Malwarebytes. Gandcrab. <https://www.malwarebytes.com/gandcrab/>, 2019.
- [Man14] Mandiant. Tracking malware with import hashing. <https://www.fireeye.com/blog/threat-research/2014/01/tracking-malware-import-hashing.html>, January 2014.
- [Mar18] Tarcisio Marinho. Ransomware encryption techniques. <https://medium.com/@tarcisioma/ransomware-encryption-techniques-696531d07bb9>, August 2018.
- [Mar20] Graciela Marker. Archivos dll: Qué son? para qué sirven? <https://www.tecnologia-informatica.com/archivos-dll-que-son-sirven/>, Enero 2020.
- [McA16a] McAfee. Crybboss - ransomware. <https://www.mcafee.com/enterprise/es-mx/threat-center/threat-landscape-dashboard/ransomware-details.crybboss-ransomware.html>, 2016.
- [McA16b] McAfee. Hydracrypt variant of ransomware distributed by angler exploit kit. <https://www.mcafee.com/blogs/other-blogs/mcafee-labs/hydracrypt-variant-of-ransomware-distributed-by-angler-exploit-kit/>, February 2016.
- [McA17] McAfee. Cryptoshield - ransomware. <https://www.mcafee.com/enterprise/es-mx/threat-center/threat-landscape-dashboard/ransomware-details.cryptoshield-ransomware.html>, 2017.
- [MD16] Steve Mansfield-Devine. Ransomware: taking businesses hostage. *Network Security*, 2016(10):8–17, 2016.
- [Med16] Eduardo Medina. Ransom32, un ransomware que afecta a windows, mac y linux. <https://www.muycomputer.com/2016/01/04/ransom32-ransomware-windows-mac-linux/>, Enero 2016.
- [MEZ21] TY MEZQUITA. Leakware. <https://cyberhoot.com/cybrary/leakware/>, March 2021.
- [Mic] Trend Micro. Command and control [c&c] server. <https://www.trendmicro.com/vinfo/us/security/definition/command-and-control-server>.
- [Mic14] Trend Micro. Matsnu. <https://www.trendmicro.com/vinfo/us/threat-encyclopedia/malware/matsnu>, July 2014.

- [Mic19] Trend Micro. Texas municipalities hit by revil/sodinokibi paid no ransom, over half resume operations. <https://www.trendmicro.com/vinfo/us/security/news/cyber-attacks/texas-municipalities-hit-by-revil-sodinokibi-paid-no-ransom-over-half-resume-operat>, September 2019.
- [Mic20a] Trend Micro. Ransomware. <https://www.trendmicro.com/vinfo/us/security/definition/ransomware>, 2020.
- [Mic20b] Microsoft. Pe format. <https://docs.microsoft.com/en-us/windows/win32/debug/pe-format>, November 2020.
- [MKK07] Andreas Moser, Christopher Kruegel, and Engin Kirda. Limits of static analysis for malware detection. In *Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007)*, pages 421–430. IEEE, 2007.
- [MM20] G Margarov and E Mitrofanova. Management of ransomware detection and prevention in multilevel environmental monitoring information system. In *Functional Nanostructures and Sensors for CBRN Defence and Environmental Safety and Security*, pages 125–131. Springer, 2020.
- [MMB18a] Shagufta Mehnaz, Anand Mudgerikar, and Elisa Bertino. Rwgard: A real-time detection system against cryptographic ransomware. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 114–136. Springer, 2018.
- [MMB18b] Shagufta Mehnaz, Anand Mudgerikar, and Elisa Bertino. RWGuard: A real-time detection system against cryptographic ransomware. In *Research in Attacks, Intrusions, and Defenses*, pages 114–136. Springer International Publishing, 2018.
- [MOB17] Guillermo Roberto Solarte Martinez, Carlos Alberto Ocampo, and Yanci Viviana Castro Bermúdez. Sistema de detección de intrusos en redes corporativas. *Scientia et technica*, 22(1):60–68, 2017.
- [Mon18] K A Monnappa. *Learning malware analysis: explore the concepts, tools, and techniques to analyze and investigate Windows malware*. Packt Publishing Ltd, 2018.
- [MS20] Abhijit Mohanta and Anoop Saldanha. *Malware Analysis and Detection Engineering*. Apress, 2020.
- [MtPDT21] Wes McKinney and the Pandas Development Team. *pandas: powerful Python data analysis toolkit*. pandas, 2021.
- [MTSH12] Jose García Moreno-Torres, José A. Saez, and Francisco Herrera. Study on the impact of partition-induced dataset shift on k -fold cross-validation. *IEEE Transactions on Neural Networks and Learning Systems*, 23(8):1304–1312, 2012.
- [NC16] NCIBE-CERT. CVE-2017-0144. Technical report, INCIBE-CERT, March 2016.
- [Nie21] Ana Nieto. Progresión de babuk, el primer ransomware del año. <https://unaaldia.hispasec.com/2021/02/progresion-de-babuk-el-primer-ransomware-del-ano.html>, Febrero 2021.
- [OPW16] Dick O'Brien, John-Paul Power, and Scott Wallace. An istr special report: Ransomware and businesses. Technical report, Symantec, July 2016.
- [Pal16] Danny Palmer. Ransomware is now the biggest cybersecurity threat. <https://www.zdnet.com/article/ransomware-is-now-the-top-cybersecurity-threat-warns-kaspersky/>, March 2016.
- [Pan18] Shubham Panchal. k nearest neighbor classifier (knn)-machine learning algorithms. <https://equipintelligence.medium.com/k-nearest-neighbor-classifier-knn-machine-learning-algorithms-ed62feb86582>, March 2018.

- [PAT17] Dr.P.B. PATHAK. Malware a growing cybercrime threat: Understanding and combating malvertising attacks. *International Journal of Advanced Research in Computer Science*, 7, 2017.
- [pE20] Juan Diego Godoy para ElPais. Los ataques por ‘ransomware’ se disparan en España por el teletrabajo. <https://elpais.com/tecnologia/2020-10-09/los-ataques-por-ransomware-se-disparan-en-espana-por-el-teletrabajo.html>, Octubre 2020.
- [Per16] Diego Perez. Nueva herramienta de descifrado para el ransomware crysis. <https://www.welivesecurity.com/la-es/2016/11/24/herramienta-descifrado-ransomware-crysis/>, Noviembre 2016.
- [Pet09] Leif E Peterson. K-nearest neighbor. *Scholarpedia*, 4(2):1883, 2009.
- [Phi16] PhishMe. Enterprise phishing susceptibility report. Technical report, Cofense, October 2016.
- [Pro19] Proofpoint. Bad rabbit. <https://www.proofpoint.com/es/glossary/bad-rabbit>, 2019.
- [PVG⁺11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [Qui86] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.
- [rad21] radare org. *The Official Radare2 Book*, 3 2021. This book is an updated version (started by majin) of the original radare1 book (written by pancake). Which is actively maintained and updated by many contributors over the Internet.
- [Ras14] Sebastian Raschka. Predictive modeling, supervised machine learning, and pattern classification. https://sebastianraschka.com/Articles/2014_intro_supervised_learning.html, August 2014.
- [Ras20] Paul Rascagneres. *Seguridad informática y malwares: análisis de amenazas e implementación de contramedidas*. ENI, 2 edition, 6 2020.
- [Red11] Martin Reddy. *API design for C*. Morgan Kaufmann, Boston, 2011.
- [RG18] Richard Rivera Guevara. Detección y clasificación de malware con el sistema de análisis de malware cuckoo. Master’s thesis, Universidad Internacional de La Rioja, 9 2018. Director del trabajo: Bermejo Higuera, Javier.
- [RHW⁺08] Konrad Rieck, Thorsten Holz, Carsten Willems, Patrick Düssel, and Pavel Laskov. Learning and classification of malware behavior. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 108–125. Springer, 2008.
- [RLH⁺20] Amos Ren, Chong Liang, Im Hyug, Sarfraz Broh, and NZ Jhanjhi. A three-level ransomware detection and prevention mechanism. *EAI Endorsed Transactions on Energy Web*, 0(0):162691, January 2020.
- [RM12] Chandrasekar Ravi and R Manoharan. Malware detection using windows api sequence and machine learning. *International Journal of Computer Applications*, 43(17):12–16, 2012.
- [RN17] Ronny Richardson and Max M North. Ransomware: Evolution, Mitigation and Prevention. PhD Thesis, Faculty Publications, January 2017.
- [Rom19] Victor Román. Machine learning: Cómo desarrollar un modelo desde cero. <https://medium.com/datos-y-ciencia/machine-learning-c%C3%B3mo-desarrollar-un-modelo-desde-cero-cc17654f0d48>, 2019.

- [Ron18] Stacey Ronaghan. The mathematics of decision trees, random forest and feature importance in scikit-learn and spark. <https://towardsdatascience.com/the-mathematics-of-decision-trees-random-forest-and-feature-importance-in-scikit-learn-5a2a0d93e0f>, May 2018.
- [RSL21] Ed Warnicke Richard Sharpe and Ulf Lampert. *Wireshark's documentation*. Wireshark, 2021. YARA is a tool aimed at (but not limited to) helping malware researchers to identify and classify malware samples.
- [RTWH11] Konrad Rieck, Philipp Trinius, Carsten Willems, and Thorsten Holz. Automatic analysis of malware behavior using machine learning. *Journal of Computer Security*, 19(4):639–668, 2011.
- [Rus21] Mark Russinovich. Process monitor v3.61. <https://docs.microsoft.com/en-us/sysinternals/downloads/procmon>, January 2021.
- [Sak20] Yaser Sakkaf. Decision trees: Id3 algorithm explained. <https://towardsdatascience.com/decision-trees-for-classification-id3-algorithm-explained-89df76e72df1>, March 2020.
- [Sar12] Sarah. The accdfisa malware family – ransomware targeting windows servers. <https://blog.emsisoft.com/en/818/the-accdfisa-malware-family-ransomware-targetting-windows-servers/>, April 2012.
- [Sec13] We Live Security. Filecoder: dinero a cambio de información secuestrada. <https://www.welivesecurity.com/la-es/2013/09/24/filecoder-dinero-cambio-informacion-secuestrada/>, September 2013.
- [Sec15] Panda Security. Cryptolocker: Qué es y cómo evitarlo. <https://www.pandasecurity.com/es/mediacenter/malware/cryptolocker/>, Mayo 2015.
- [Sec19] Globb Security. Estos son los cinco tipos de ransomware que más afectan a los usuarios. <https://globbsecurity.com/estos-son-los-cinco-tipos-de-ransomware-que-mas-afectan-a-los-usuarios-44183/>, Enero 2019.
- [SH12] Michael Sikorski and Andrew Honig. *Practical Malware Analysis: A Hands-On Guide to Dissecting Malicious Software*. No Starch Press, 1 edition, 2 2012.
- [SHG20] Pratap Chandra Sen, Mahimarnab Hajra, and Mitadru Ghosh. Supervised classification algorithms in machine learning: A survey and review. In *Emerging technology in modelling and graphics*, pages 99–111. Springer, 2020.
- [Sim15] Alina Simone. The strange history of ransomware. <https://medium.com/un-hackable/the-bizarre-pre-internet-history-of-ransomware-bb480a652b4b>, August 2015.
- [SM17] Manisha Patil Savita Mohurle. A brief study of wannacry threat: Ransomware attack 2017. *INTERNATIONAL JOURNAL OF ADVANCED RESEARCH IN COMPUTER SCIENCE*, 8, 2017.
- [SMGML16] Daniele Sgandurra, Luis Muñoz-González, Rabih Mohsen, and Emil C Lupu. Automated dynamic analysis of ransomware: Benefits, limitations and use for detection. *arXiv preprint arXiv:1609.03020*, 2016.
- [Smi18] Ms. Smith. Ransomware attack hits north carolina water utility following hurricane. <https://www.csoonline.com/article/3314557/ransomware-attack-hits-north-carolina-water-utility-following-hurricane.html>, October 2018.

- [Smi19] Ms. Smith. Major us newspapers crippled by ryuk ransomware attack. <https://www.csoonline.com/article/3330645/major-us-newspapers-crippled-by-ryuk-ransomware-attack.html>, January 2019.
- [Sol18] Indian Cyber Security Solutions. Noriben: Portable, simple, malware analysis sandbox. <https://indiancybersecuritysolutions.com/noriben-malware-analysis-sandbox/>, April 2018.
- [Sou] Sourceforge. Process hacker. <https://processhacker.sourceforge.io/>.
- [Sou10] César Souza. Kernel functions for machine learning applications. <http://crsouza.com/2010/03/17/kernel-functions-for-machine-learning-applications/#linear>, March 2010.
- [Spa20] Spambrella. What is kovter malware? <https://www.spambrella.com/what-is-kovter-malware/>, 2020.
- [SRRSA20] SophosLabs, Sophos Managed Threat Response, Sophos Rapid Response, and Cloud Security Sophos AI. Sophos 2021 threat report navigating cybersecurity in an uncertain world. Technical report, SophosLabs, November 2020.
- [SS04] Jordi Serra Serra. La firma electrónica y el archivo digital. 2004.
- [SS18] R Saravanan and Pothula Sujatha. A state of art techniques on machine learning algorithms: a perspective of supervised learning approaches in data classification. In *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 945–949. IEEE, 2018.
- [Sta] Statista. Number of sent and received e-mails per day worldwide from 2017 to 2022 (in billions).
- [Sta17] William Stallings. *Network security essentials : applications and standards*. Pearson Education Limited, 2017.
- [Tec16] TechTerms. Sandboxing. <https://techterms.com/definition/sandboxing>, July 2016.
- [TG] Jorge Tomás Guerra. Monitorización de seguridad con wazuh.
- [TIT20] Malware Labs Threat Intelligence Team. Cybercriminals impersonate World Health Organization to distribute fake coronavirus e-book. <https://blog.malwarebytes.com/social-engineering/2020/03/cybercriminals-impersonate-world-health-organization-to-distribute-fake-coronavirus-ebook/>, Marzo 2020.
- [TML⁺20] Fei Tang, Boyang Ma, Jinku Li, Fengwei Zhang, Jipeng Su, and Jianfeng Ma. Ransomspector: An introspection-based approach to detect crypto ransomware. *Computers & Security*, 97:101997, 2020.
- [Tor16] Miguel Torres. 7 TERRORÍFICAS FAMILIAS DE RANSOMWARE. BlogSmartekh, 2016.
- [TSF18] Yuki Takeuchi, Kazuya Sakai, and Satoshi Fukumoto. Detecting ransomware using support vector machines. In *Proceedings of the 47th International Conference on Parallel Processing Companion*, pages 1–6, 2018.
- [TVE20] Tony Thomas, Athira P. Vijayaraghavan, and Sabu Emmanuel. *Machine Learning Approaches in Cyber Security Analytics*. Springer Singapore, 2020.
- [UAB19a] Daniele Ucci, Leonardo Aniello, and Roberto Baldoni. Survey of machine learning techniques for malware analysis. *Computers & Security*, 81:123–147, March 2019.
- [UAB19b] Daniele Ucci, Leonardo Aniello, and Roberto Baldoni. Survey of machine learning techniques for malware analysis. *Computers & Security*, 81:123–147, 2019.

- [UKA13] Ihsan Ullah, Naveed Khan, and Hatim A Aboalsamh. Survey on botnet: Its architecture, detection, prevention and mitigation. In *2013 10th IEEE International Conference on Networking, Sensing and Control (ICNSC)*, pages 660–665. IEEE, 2013.
- [Vad] Vadesecure. Ransomware Statistics 2017.
- [VDL17] Liberios Vokorokos, Zuzana Dankovičová, and L’ubor Leščišin. Using of the forensic analyzing tools, code obfuscation. In *2017 IEEE 15th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*. IEEE, January 2017.
- [Vir20] VirusTotal. *YARA’s documentation*, 2020. YARA is a tool aimed at (but not limited to) helping malware researchers to identify and classify malware samples.
- [VSVG17] R Vinayakumar, KP Soman, KK Senthil Velan, and Shaunak Ganorkar. Evaluating shallow and deep networks for ransomware detection and classification. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 259–265. IEEE, 2017.
- [WeL14] WeLiveSecurity. Lo que probablemente no sabías sobre fuzzy hashing. <https://www.welivesecurity.com/la-es/2014/02/21/probablemente-no-sabias-fuzzy-hashing/>, Febrero 2014.
- [Wer13] Dave Werden. Review of spyware and adware by john ayccock. *ACM SIGACT News*, 44:17–19, 03 2013.
- [Wir] Wireshark. About wireshark. <https://www.wireshark.org/>.
- [Yad18] Ajay Yadav. Support vector machines(svm). <https://towardsdatascience.com/support-vector-machines-svm-c9ef22815589>, October 2018.
- [YY96] A. Young and Moti Yung. Cryptovirology: extortion-based security threats and countermeasures. In *Proceedings 1996 IEEE Symposium on Security and Privacy*, pages 129–140. IEEE, 1996.
- [Zet15] K Zetter. Hacker lexicon: A guide to ransomware, the scary hack that’s on the rise. <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>, 2015.
- [Zho17] Naaman Zhou. Traffic cameras in victoria infected by wannacry ransomware. <https://www.theguardian.com/australia-news/2017/jun/22/traffic-cameras-in-victoria-infected-by-wannacry-ransomware>, June 2017.
- [ZL⁺16] Pavol Zavarsky, Dale Lindskog, et al. Experimental analysis of ransomware on windows and android platforms: Evolution and characterization. *Procedia Computer Science*, 94:465–472, 2016.