Universidad de León

Departamento de Ingenierías

Mecánica, Informática y Aeroespacial

Conocimiento experto y minería de datos sobre reportes de *firewall* aplicado a la detección de Amenazas Persistentes Avanzadas

Expert knowledge and data mining over firewall reports to detect Advanced Persistent Threats

Tesis presentada por

Juan Ramón Moya Vasco

para la obtención del Título de Doctor por la Universidad de León

Directores:

Dr. D. Ramón Ángel Fernández Díaz

Dr. D. Miguel Carriegos Vieira

León, julio 2017

... Esta Tesis está dedicada a mi preciosa mujer, Yayi, y mis dos maravillosos hijos, Marcos y Jorge, mi familia, de la que me siento muy orgulloso, quiero y adoro con pasión. Muchas gracias por todo el cariño recibido y el apoyo mostrado a lo largo de toda una vida [...] ...

Agradecimientos

Quisiera expresar mi más profundo agradecimiento a mis directores de Tesis, Ramón Ángel Fernández Díaz y Miguel Carriegos Vieira, por el enorme esfuerzo realizado, por el apoyo incondicional, consejos, revisiones, sugerencias, correcciones e interminables conversaciones telefónicas para darle forma a todo esto.

A Noemí DeCastro, muchas gracias de corazón por todo lo que me has ayudado, tus sabios consejos y aportaciones.

A mi buen amigo Juan Ángel cuyas recomendaciones, sugerencias, ideas, y conjeturas consiguieron mejorar la estructura y el contenido del proyecto de Tesis.

Y en general a todas y cada una de las personas que han participado en ese trabajo, sin los cuales, esta Tesis doctoral jamás hubiera sido acabada.

Juan Ramón Moya Vasco

RESUMEN

En este trabajo se propone una metodología basada en conocimiento experto para construir un sistema inteligente capaz de detectar comportamientos anómalos y clasificar, en su caso, Amenazas Persistentes Avanzadas (*APTs*).

Un experto puede intuir si existe peligro real para una infraestructura *TIC* a partir de la información contenida en los registros de *log* del *firewall* que controla su tráfico de entrada/salida. Este conocimiento experto se puede modelar, y con la ayuda de minería de datos y de sistemas de aprendizaje automático se puede construir una herramienta capaz de identificar tráfico malicioso. Para seleccionar el sistema de aprendizaje automático más adecuado, la información de tráfico real se ha completado con datos sintéticos, a fin de representar diferentes proporciones de actividad anómala en el conjunto de datos de tráfico.

Esta metodología se ha aplicado a un entorno de tráfico real, y el sistema inteligente desarrollado muestra unas tasas de comportamiento aceptables en la detección de ataques por *APT*.

ABSTRACT

This works proposes a methodology based on expert knowledge to develop an intelligent system that detects anomalous performance and, if appropriate, classify Advanced Persistant Threats (APTs).

An expert can suspect real threat for an ICT infrastructure from the information available in its firewall input/output log registers. Such expert knowledge can be modeled, and with the help of data mining and automatic learning systems, used to develop a tool that identifies malicious traffic. To choose the learning system that better fits, real traffic information is completed with synthetic data, so that different proportions of anomalous activity appear in the traffic data set.

The proposed methodology has been used with information from a real environment, and the resulting intelligent system to detect APTs shows acceptable performance rates.

Finally the model has been used in a real traffic environment obtaining an acceptable hit rate in the detection of APT attacks.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Motivación

La sociedad actual, su economía, la prestación de servicios por parte de las administraciones públicas, la actividad de las empresas, profesionales y ciudadanos, así como el acceso a la información, la educación, el comercio o el ocio se sustentan hoy más que nunca en el ciberespacio. Trabajar, jugar, comprar, mantener relaciones sociales, financieras o pagar impuestos *online*, entre otras muchas, constituyen actividades cotidianas que se llevan a cabo usando el amplio abanico de dispositivos electrónicos y redes de comunicaciones que existen actualmente en el mercado [1].

Desde la creación del primer virus informático [2] a comienzos de la década de 1970, la complejidad de los ataques, así como las tecnologías y herramientas utilizadas para llevarlos a cabo han evolucionado drásticamente con el paso del tiempo. Diariamente se descubren nuevas formas de ataque a dispositivos, sistemas informáticos, redes de comunicaciones, o vulnerabilidades a códigos y protocolos [3].

Ningún dispositivo *hardware*, ni *software*, existente en la actualidad, desde el sistema más complejo instalado en una infraestructura estratégica, tal como una central nuclear [4] , hasta la aplicación más simple que tengamos instalada en nuestro teléfono móvil [5], está exento de ser atacada. La amenaza es latente y existe. Es tal su impacto, que existen herramientas para

visualizar en tiempo real los ataques que se producen y detectan en el mundo [6] (figura 1.1).

1.2. Contexto de la investigación

1.2.1. Amenazas a la seguridad de las infraestructuras *TIC*

El objetivo principal de una infraestructura *TIC* consiste en dar servicio a sus usuarios de forma segura. Este objetivo de seguridad implica que los servicios que presta la infraestructura no se vean afectados por potenciales ataques, por un lado; y que la información que gestiona permanezca protegida y custodiada en todo momento, por el otro.

A pesar de los esfuerzos que se realizan en aspectos de seguridad, estas instalaciones están expuestas de manera continua a ataques de muy diversos tipos: virus, gusanos, *spyware, troyanos, exploits, dialers, botnets, APTs*, etc. Las posibilidades de éxito de un ataque aumentan cuando los usuarios no siguen las pautas de comportamiento digital seguro establecidas por cada infraestructura *TIC*.

La seguridad de una infraestructura depende en gran medida del nivel de concienciación en aspectos de seguridad de sus usuarios, y no solo de la cantidad de dispositivos de seguridad (*hardware* o *software*) de que disponga. Los atacantes pueden generar modelos predictivos de comportamiento a partir de la ingente cantidad de información –personal y profesional– que generan los usuarios de una infraestructura, y utilizarlos para atacarla.

Los antivirus, *firewalls, IDS (Intrusion Detection System), IPS (Intrusion Prevention System), NIS (Network Inspection System)*, *SIEM (Security Information and Event Management)*, etc., son sistemas de defensa que alertan y protegen de una manera rápida y precisa de ataques directos contra los activos de una infraestructura, pero no son adecuados para protegerlos frente a ataques indirectos, que utilizan información publicada, conscientemente o no, por los propios usuarios de la infraestructura.

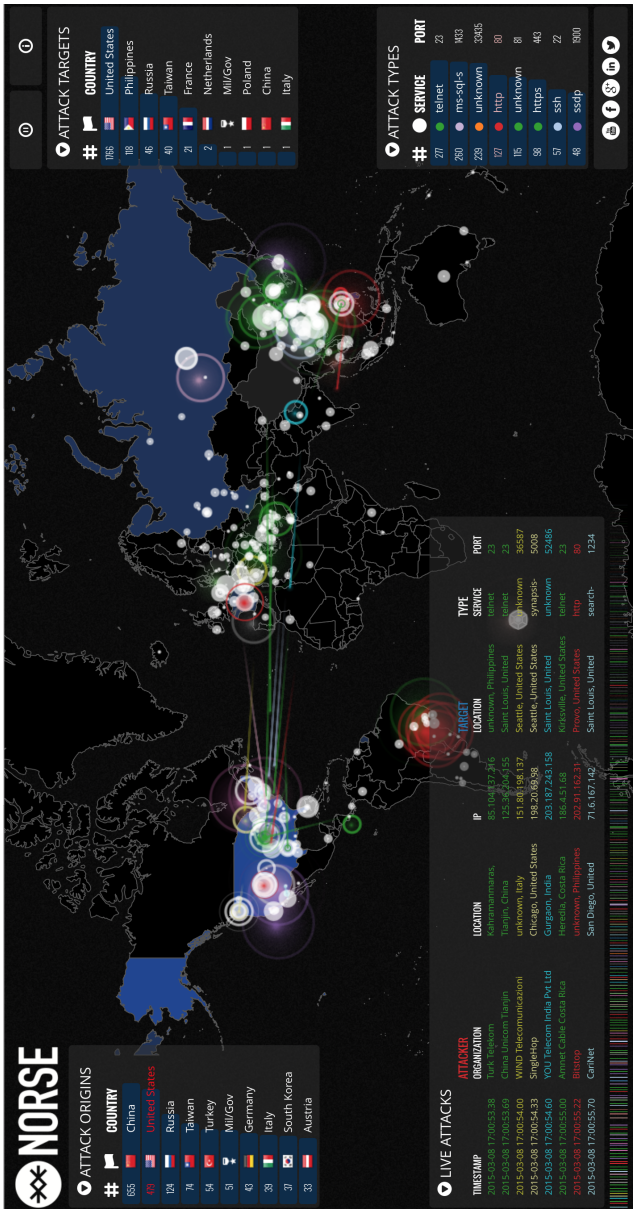


Figura 1.1: Ejemplo de aplicación para visualización de ataques en tiempo real [6]

1.2.2. Amenazas Persistentes Avanzadas (*APTs*)

Las amenazas, que en un principio eran puntuales y muy concretas en cuanto a objetivos, con técnicas muy simples y consecuencias para el usuario más molestas que dañinas, se han transformado en amenazas muy sofisticadas, con objetivos muy heterogéneos y dirigidos a cualquier tipo de organización: empresas del sector de las tecnologías, gobiernos, organizaciones militares, partidos políticos, administraciones de cualquier tipo, etc. Algunas de éstas son las Amenazas Persistentes Avanzadas o *APTs* (*Advanced Persistent Threats),* que utilizan fundamentalmente técnicas de Ingeniería Social basadas en la confianza, y cuyo objetivo es permanecer latentes e indetectables en una infraestructura con el objetivo de robar información durante la mayor cantidad de tiempo posible [7].

1.3. Definición del problema de la investigación

Los ataques por *APT* implican los siguientes aspectos: organización, premeditación, constancia, paciencia, sofisticación, medios económicos, pericia e innovación. Este tipo de ataques pueden estar financiados por gobiernos, por ciberdelincuentes, por empresas, etc., con el fin de mantener campañas de ataques durante periodos largo de tiempo. El grado de sofisticación llega a tal nivel que incluso en caso de ser detectados, lo más probable es que los atacantes tengan un plan de contingencia que les permita reorganizarse y atacar de nuevo haciendo muy difícil su detección con métodos tradicionales: la actividad anormal de usuarios autorizados, el acceso a datos fuera de contexto o una secuencia inusual de comportamiento, que tradicionalmente serían tomados como eventos de bajo riesgo, pueden ser las únicas señales de un ataque *APT* [7]. Además suelen atacar a infraestructuras sensibles, a menudo con datos protegidos por ley, por lo que el acceso a los registros de esta información no se permiten.

De 2012 a 2017 las *APTs* han alcanzado niveles muy altos de sofisticación. En los ataques tradicionales el objetivo era recolectar información acerca de vulnerabilidades técnicas que poseía la víctima o la red a atacar. Sin embargo un ataque *APT* empieza con un análisis metódico y un estudio del

componente humano de la organización objetivo del ataque, obteniendo a través de Ingeniería Social todo tipo de información de los empleados que trabajan en ella, que habitualmente son el eslabón más débil en el contexto de la seguridad [8].

La información está en Internet y es sumamente sencillo conseguir información en una red social de algún empleado de una empresa, el nombre de su mascota, aficiones, fecha de nacimiento, nombre de su hijo, en definitiva posibles datos para un usuario y una contraseña. Tarea mucho más sencilla que intentar superar controles de seguridad proporcionados a través de *switches, firewall, proxies, IPS, IDS*, etc.

Por lo anteriormente expuesto, el problema de la investigación es detectar *APTs* utilizando los rastros de actividad en la red (*logs* del *firewall*). Se distingue el tráfico que genera una *APT* del tráfico de red legítimo para así poder identificar *APTs.*

1.4. Objetivos de esta Tesis

Desarrollar una metodología para implementar sistemas inteligentes sobre infraestructuras *TIC* que detecten la actividad de red que genera una *APT*.

El sistema utiliza minería de datos y aplica técnicas de aprendizaje supervisado y no supervisado [9], utilizando los registros de *logs* que proporcionan los *firewalls* encargados de filtrar el tráfico de entrada/salida de la infraestructura.

La parte experimental se ha realizado utilizando conjuntos de datos semisintéticos, que incluyen registros correspondientes a una *APT* real.

Para llevarlo a cabo, los objetivos específicos son los siguientes:

Ob1.- Redefinir el concepto de *APT*.

Ob2.- Extraer los parámetros de comportamiento básico que permitan detectar una *APT*.

Ob3.- Analizar, normalizar y discretizar los parámetros de comportamiento que permitan detectar una *APT*.

Ob4.- Crear un vector normalizado de elementos.

Ob5.- Crear un algoritmo de normalización para el tratamiento de los *logs*.

Ob6.- Establecer las reglas de comportamiento fundamentales de las *APTs*.

Ob7.- Seleccionar una técnica de minería de datos adecuada.

Ob8.- Diseñar un Sistema de Aprendizaje Automático (*SAA*) que permita detectar una *APT* utilizando las reglas anteriores.

Ob9.- Generar un entorno software de generación de *logs* para entrenamiento y validación del *SAA*.

1.5. Estructura de la memoria de Tesis

El capítulo 1 refleja la motivación de este trabajo relacionado con las amenazas a la seguridad de las infraestructuras *TIC*. Se esboza qué es una *APT*, así como el objetivo general y los específicos de la investigación.

En el capítulo 2 se expone una introducción a las amenazas en general, sus orígenes históricos, así como la evolución del malware hasta las *APTs*. A continuación se analizan las definiciones de *APT*, fases de ataques, escenarios posibles, los marcos teóricos que existen, para detección de *APTs* y casos reales hasta 2017. Se puntualiza que es un *firewall* y los archivos de *logs* que genera. Finalmente se describe la minería de datos y las técnicas utilizadas para los sistemas de aprendizaje automático.

En el capítulo 3 se describe la metodología utilizada para conseguir los objetivos de la investigación. Se muestran las fases y tareas de la investigación así como los trabajos a realizar en cada una de ellas. Se presentan los materiales, instrumentos y muestras que se han utilizado para su desarrollo.

En el capítulo 4 se presenta el diseño del experimento con los flujos de trabajo utilizados en los algoritmos de aprendizaje. Se muestran los ejemplos de entrenamiento y test utilizados. Finalmente se describe como se lleva a cabo la realización del experimento.

En el capítulo 5 se detallan los resultados obtenidos de los experimentos llevados a cabo, con las técnicas de aprendizaje automático, lo que permitirá elegir el algoritmo de entrenamiento adecuado.

En el capítulo 6 se analizan los resultados obtenidos en los experimentos,

se valoran y se obtienen las muestras que consiguen los objetivos de la investigación. Con las muestras seleccionadas se realizan pruebas con datos reales evaluando los resultados para establecer conclusiones.

En el capítulo 7 se manifiestan las conclusiones obtenidas del trabajo de investigación así como las aportaciones de la Tesis. Finalmente se exponen las futuras líneas de investigación.

En el Apéndice A se muestran los ataques más representativos sufridos por las infraestructuras *TIC* desde 2003 a 2017.

El Apéndice B detalla los campos que componen el registro de *log* de un *firewall*.

Por último, en el apéndice C se enumeran las publicaciones relacionadas con esta Tesis.

Capítulo 2

Fundamentos de la investigación

2.1. Evolución de las amenazas hacia las infraestructuras *TIC*

2.1.1. Introducción

En 1949 – 1950, en los laboratorios de *Bell Computer*, tres programadores: *Robert Thomas Morris, Douglas Mcllroy y Victor Vyssotsky* crean un juego denominado *CoreWar* [10] basado en la Teoría de *von Neumann* [11]. Es una batalla entre programas escritos en lenguaje ensamblador que combaten entre sí para ocupar toda la memoria de la máquina. Este juego se considera como el precursor de los virus informáticos.

La evolución de la tecnología, las redes de comunicaciones, los protocolos de conexión, etc., han dado lugar a que la complejidad de los ataques hayan ido evolucionando paralelamente, tanto en conocimiento como en fabricación de herramientas *hardware* y *software*.

Actualmente existe *malware* para todas las plataformas más comunes. Cualquier nueva tecnología será aprovechada como modelo de propagación de un ataque.

En un corto espacio de tiempo, 2010–2017, las amenazas y los daños en el campo de la ciberseguridad crecen de manera exponencial; no pasa un día sin que se produzca una noticia relacionada con alguno de estos ataques. Los

gobiernos de todo el mundo y, en especial, el sector empresarial tratan de proteger su información que constituye una de las principales herramienta de poder.

2.1.2. Orígenes históricos

A finales del siglo III a.C., Roma estaba a punto de ser aniquilada por los ejércitos cartagineses al mando de Aníbal, considerado como uno de los mejores estrategas militares de todos los tiempos [12]. Aníbal se alió con los pueblos enemigos de Roma, hábiles en distintas estrategias militares; juntos constituían una fuerza imparable que llegó hasta las puertas de Roma, sitiando la ciudad.

Pocos años antes del estallido del conflicto, nació Publio Conerlio Escipión "El Africano", hijo del Cónsul de Roma y sobrino de Gneo Cornelio Escipión. Heredó la sabiduría de su padre y el valor guerrero de su tío.

En 209 a.C., la capital del Imperio Cartaginés en Iberia era Karth Hadtha, posteriormente llamada Carthago Nova, y actualmente Cartagena. La ciudad conectaba con la península ibérica a través de un pequeño istmo, protegida por una enorme muralla de 6 metros de grosor que rodeaba la ciudad. Era una ciudad fuerte e inexpugnable, un auténtico bastión, protegida a su vez por accidentes geográficos que daban al mar, y que dejaban como único punto de entrada a la ciudad el pequeño istmo de tierra, con un ejército fuertemente armado, con armas y víveres suficientes como para resistir meses. Karth Hadtha en Iberia cayó a manos de Escipión. El asedio duró 6 días y la planificación del ataque, meses.

¿Cómo lo consiguió? Con un ejército dotado de los mejores expertos en escalada, batallas navales, escaramuzas, combates cuerpo a cuerpo, ... pero ante todo buenos estrategas. Chocar contra murallas, por el mar, o asediar muros infranqueables, eran batallas perdidas, los cartagineses estaban siempre alerta, vigilantes. Inició el ataque por los flancos más protegidos con el objeto de que los cartagineses concentraran sus fuerzas en defender lo inexpugnable.

A la par que se desarrollaba la gran batalla, una pequeña guarnición de

soldados, subrepticiamente, atravesaba una laguna y accedía a una muralla más baja y menos protegida, y habiendo sobornado a alguien de confianza y proveedor de los cartagineses – al cual le habían realizado un seguimiento de costumbres, familia, horarios,... – para que los llevase a la parte más débil, sin apenas vigilancia y accesible de la fortaleza. Entraron en la ciudad amurallada y, poco a poco, se fueron haciendo con posiciones más efectivas, primero un puesto, luego otro, y otro, hasta que se hicieron con todo el control y consiguieron que el grueso del ejército romano entrara en la ciudad. Fue el comienzo del fin de Aníbal y del Imperio Cartaginés.

En este momento de la historia, Escipión fue para los cartagineses un enemigo que poseía niveles sofisticados de conocimientos y recursos que le permitieron atacar a sus objetivos mediante el uso de múltiples vectores de ataque. Una vez atacada la infraestructura y establecidos dentro de ésta, se hicieron con el dominio. Una amenaza persistente avanzada frente a una gran fortaleza. En la actualidad las grandes fortalezas no tienen 6 metros de espesor en sus paredes sino una conexión a Internet de alta velocidad con sistemas de protección hardware y software.

El modus operandi es el mismo. De hecho, Escipión sentó cátedra en las estrategias para la tercera guerra púnica, por lo novedoso y técnico de su ataque. Publio Cornelio Escipión fue para los cartagineses una Amenaza Persistente Avanzada en cuanto a planificación y ejecución.

En la actualidad el campo de batalla es el ciberespacio. El modelo de planificación es prácticamente el mismo, cambian los escenarios y los hitos temporales, pero las motivaciones y objetivos perduran en el tiempo.

2.1.3. Evolución del malware. Del *Corewar* a las *APTs*

Fue en 1939 cuando el científico y matemático *John von Neumann* escribió "La Teoría y Organización de Autómatas Complejos" [11]. Manifestaba por primera vez la posibilidad de poder desarrollar pequeños programas replicantes que fueran capaces de tomar el control de otros programas. La aplicación negativa de la Teoría de *von Neumann* son los virus informáticos.

La RAE (Real Academia Española) [13] define virus como: Organismo de

estructura muy sencilla, compuesto de proteínas y ácidos nucleicos, y capaz de reproducirse solamente en el seno de células vivas específicas, utilizando su metabolismo. Programa introducido subrepticiamente en la memoria de un ordenador que, al activarse, destruye total o parcialmente la información almacenada.

Se propone una definición más pragmática : "agente infeccioso microscópico (pequeño programa) que se multiplica (se replica) y desarrolla dentro de las células (memoria, discos duros) de un organismo (ordenador, *tablet*, *smartphone*, etc.)" se reproducen y se propagan por contagio de unos ordenadores a otros, creando una epidemia y aumentando el número de víctimas de forma exponencial.

A continuación expondremos cómo las amenazas han ido evolucionando a lo largo de las distintas generaciones [14, 15] y cómo se empiezan a esbozar los conatos de defensas contra dichas amenazas.

Durante las tres primeras generaciones, el diseño de virus y antivirus estaba supeditado a laboratorios y grandes empresas tecnológicas, y de ahí la escasez de éstos. Sin embargo a partir de la cuarta generación y el despegue de los ordenadores personales, el *malware* experimenta un importante crecimiento.

- Primera Generación (1940 – 1952).

En 1940 aparece la primera calculadora electro-mecánica que podía manejar números complejos, y en 1952 se diseña la primera computadora binaria, llamada EDVAC.

En 1949 – 1950 en los laboratorios de Bell Computer *Robert Thomas Morris, Douglas McIlroy y Victor Vyssotsky* desarrollan el juego denominado *CoreWar* [10], considerado como el precursor de los virus informáticos y basado en la Teoría de *von Neumann*, consistente en un combate entre programas que tratan de ocupar toda la memoria de la máquina

- Segunda Generación (1952 – 1964).

En 1953 IBM fabrica el IBM650 y en 1964 aparece el lenguaje BASIC. En 1959, el matemático Británico *Lionel Penrose* presentó su punto de vista

sobre las máquinas autorreplicantes en un artículo de la revista *Scientific American* "*Self – reproducing machines*' [16].

A diferencia de *von Neumann*, *Penrose* describió un modelo simple de esta estructura de dos dimensiones que podría ser capaz de activarse, reproducirse, mutar y atacar. Después de un tiempo de publicado el artículo de *Penrose*, *Frederick G. Stahl* reprodujo este modelo en un programa para IBM650.

En 1962 *Vyssotsky, McIlroy y Morris* crearon un juego llamado '*Darwin*'. El juego incluía en un "árbitro" en la memoria del ordenador que determinaba las reglas y el orden de la batalla entre los programas creados por los jugadores [17].

- Tercera Generación (1964 – 1971).

En 1964 aparece el IBM360, con circuitos integrados, y en 1971 Intel presenta su 4004, el primer procesador comercial.

En 1970 – 1971, *Morris* creó el *Creeper* [2], capaz de infectar máquinas IBM360 de la red ARPANET (la precedente de Internet). El virus emitía un mensaje por pantalla que decía "Soy una enredadera (*creeper*), atrápame si puedes". Para eliminarlo, se creó el primer "antivirus" llamado *Reaper* (segadora) [18].

- Cuarta Generación (1971 – 1981).

En 1971 se crea el lenguaje C y aparecen los primeros disquetes, y en 1981 IBM lanza el IBM PC con MS-DOS.

En 1974 el virus *Rabbit*, diseñado para el sistema operativo ASP de IBM, se hacía copias de sí mismo y no hacía otra cosa que reproducirse continuamente, de ahí su nombre, colapsando los recursos del sistema [19].

En 1981, *Richard Skrenta* desarrolla *Elk Cloner* [20], concebido para gastar bromas a los compañeros de clase mostrando un poema. Es el código del primer virus de amplia reproducción en ordenadores *Apple II*, debido a la gran difusión de estos ordenadores, y constituye la primera epidemia a gran escala a través de disquetes infectados.

- Quinta Generación (1981 – actualidad).

En el año 1984 *Frederick B. Cohen* [21] define por primera vez qué es un virus informático: "programa que puede infectar a otros programas incluyendo una copia posiblemente evolucionada de sí mismo".

En 1987 el virus *Jerusalem*, *Viernes 13* [22], famoso por detectarse en la Universidad Hebrea de Jerusalén atacaba al sistema operativo MS–DOS infectando archivos con las extensiones .exe, .com, .sys y .bin.

En 1999 surge el gusano [23] *Happy* desarrollado por el francés *Spanska* [24], con la capacidad de propagarse en datos adjuntos a correos electrónicos, se autoreenviaba a toda la lista de contactos del equipo infectado. No era de carácter destructivo, pero provocaba colapsos en equipos y redes.

En 2000 el gusano *I Love You* [25], conocido también como *Love letter*, se propagó de manera global: una gran cantidad de ordenadores se contagiaron por medio del correo electrónico. La infección fue de tal magnitud, que dio lugar a una repercusión mediática sin precedentes. Esta forma de ataque se puede considerar como el inicio del uso de técnicas de obtención de información utilizando herramientas de Ingeniería Social (*Social Engineering* y *Gathering*) [8].

El apéndice A muestra los ataques más representativos, por su impacto o por la virulencia de sus consecuencias, que tuvieron lugar entre los años 2003 y 2017.

Como se ha indicado, los ataques afectan a empresas del sector de las tecnologías, a gobiernos, a organizaciones militares, a partidos políticos, a administraciones de cualquier tipo, a empresas, etc., y consiguen con paciencia y tesón robar todo tipo de información, provocando un riesgo en la infraestructura que hace que sus efectos sean demoledores, son las denominadas Amenazas Persistentes Avanzadas o *APTs*.

En la cuarta revolución industrial, robots y asistentes personales serán los nuevos vectores de ataque para grupos especializados en *APTs* como pueden ser, entre otros, *LongHorn, Fancy Bear, Anonymous, APT28, APT29, APT30, RawnStorm, The Lotus Blossom APT* o *The Backgear APT* [26].

2.1.4. Medidas de seguridad en sistemas *Windows, Linux* y *Macintosh*

Para intentar minimizar los efectos de ataques a las infraestructuras TIC, las empresas han diseñado para sus sistemas operativos distintas medidas de seguridad:

1. SEH (Structured Exception Handling). Es el mecanismo que dispone *Windows* para gestionar las excepciones que ocurren cuando se ejecuta un programa, de tal manera que éste pueda recuperarse en caso de que se produzca alguna excepción de tipo *hardware* o *software*. Se podía aprovechar un desbordamiento de *buffer* para modificar la dirección del *handler* que se va a ejecutar [27]. Un desbordamiento de *buffer* (*buffer overflow*) es un fallo de programación que se produce cuando la cantidad de datos que se escriben en un *buffer* (área reservada en memoria) es superior al tamaño del propio *buffer*. Es una de las armas más comunes y peligrosas utilizadas por los atacantes [28].

2. *SafeSEH.* Es el mecanismo de seguridad en equipos *Windows XP SP2* en arquitecturas *x386*, que evita que se pueda explotar el gestor de excepciones *SEH*. Para ello los programas o módulos del sistema operativo deberán de estar compilados con el flag */SAFESEH*. En *arquitecturas* de 64 bits, que no utilizan *stackframes*, se emplean unas estructuras llamadas *PDATA*, que evitan los desbordamientos de *buffer* [29].

3. *SEHOP.* Es una medida de seguridad enfocada a evitar la explotación del sistema de gestión de excepciones. No se activa en tiempo de compilación, sino que está implementada en el propio sistema operativo. Esta medida está disponible, pero no activada, desde las versiones Windows Vista SP1 y Windows 7; y sí está activada por defecto en Windows 8 y 8.1 [30].

4. *Stack Cookies.* Es una medida de seguridad utilizada para detectar si se ha producido un desbordamiento de *buffer* en la pila. Esta medida se

implementó a partir de *Windows XP* y está presente en todos los sistemas operativos *Windows,* Linux y Mac [31].

5. *DEP.* Es una medida de seguridad que evita la ejecución de código por parte del atacante que haya conseguido entrar en el sistema y explotar alguna vulnerabilidad. Se trata de máquinas que soporten *hardware–enforced DEP*. Esta funcionalidad se utilizó a partir de *Windows XP* y se añadió por defecto a las versiones *Vista*, 7, 8, 8.1, *Server* 2003, 2008 y 2012 de este sistema operativo. Para evadir *DEP*, se utiliza *ROP* (*Return Oriented Programming*), que es una técnica de programación destinada a los sistemas de protección de memoria más comunes [32]. El atacante lo distribuye en pequeñas porciones de código en memoria que se denominan *ROP Gadgets*, y el conjunto de *Gadgets* se denomina *ROP Chain*. En sistemas de 64 bits *DEP* está activado siempre por defecto [33].

6. ASLR (Adress Space Layout Randomization). Es una medida de seguridad implementada en sistemas *Windows, Linux y Mac*, que permite que los módulos y secciones del código binario se carguen en direcciones de memoria aleatorias, evitando que el atacante pueda conocer la distribución de los componentes del binario en memoria. Aunque *Windows Vista* dio soporte para *ASLR*, se podía atacar con técnicas de fuerza bruta o *heap spraying* (escribir una serie de bytes en distintas zonas de memoria asignadas a programas) hacia navegadores web. En *Windows* 8 ya no funcionan estas técnicas de ataque. En *Windows* 8 64 *b*its, todas las técnicas de ataque hacia *ASLR* son infructuosas, por lo que la única manera de atacar estos sistemas es a través de *Information Leaks.* Esto consiste en explotar la vulnerabilidad específica de un programa, mediante la cual es posible obtener información o investigar sobre la disposición de los objetos que conforman el binario en memoria [34].

7. NX (No eXecute bit). Es una protección por *kernel* o por *hardware* en sistemas Linux y Mac, consistente en poner las páginas de memoria donde se almacena el binario, a no ejecutables, y así evitando que puedan inyectar código malicioso y ejecutarlo. Las técnica para saltar el *NX* es a través de *ROP* [35].

8. PIE (Position Independant Executable). Esta medida de seguridad obliga a que el programa utilice direcciones relativas de memoria en vez de direcciones absolutas, permitiendo aplicar *ASLR* al espacio de direcciones del ejecutable en sistemas Linux y Mac [36].

2.1.5. Técnicas de exfiltración de información en una infraestructura *TIC*

Entre las técnicas utilizadas por los hackers para la exfiltración de datos y mando-control *(command-control, C&C),* tenemos las siguientes [37]:

1. *Dominio o dirección IP dedicados.* Es la técnica primigenia de las primeras *APTs*, eran fáciles de detectar y eliminar, ya que se conoce la dirección *IP* del servidor donde se conectaba al exterior. Una vez detectada una APT, se informaba al proveedor de servicios de Internet para que bloquease su IP.

2. Lista de Dominios. Similar a la anterior, contiene una cantidad ingente de dominios para exfiltraciones de datos y comunicaciones con los ordenadores comprometidos, de tal manera que si uno de los dominios es detectado y anulado, muchos otros seguirán activos. Existen herramientas que permiten detectar este tipo de comportamiento en la red, identificando los dominios y eliminándolos de la misma.

3. Peer to Peer. Cada equipo infectado dentro de la red atacada se utiliza como esclavo o maestro al objeto de recibir o enviar información. Deberá de existir una comunicación interna entre las máquinas comprometidas. Sin embargo cuando un equipo es detectado por un comportamiento anómalo, por alguna de las sondas de seguridad que posea la infraestructura atacada, se procede a eliminar y limpiar dicha máquina, y a realizar un seguimiento de conexiones para seguir la traza del ataque y logara que deje de ser efectivo.

4. Dominios en Tor. Algunas *APTs*, utilizan dominios de la red *Tor (The onion router)* para conseguir exfiltrado de información o acciones de *C&C.*

Sin embargo, la red *Tor* resulta lenta y poco fiable para realizar este tipo de ataques.

5. Servicios de Redes Sociales. La técnica consiste en crear una cuenta en una red social (*Facebook* o Twitter, por ejemplo), a la que la máquina comprometida y el servidor $C\&C$ envían y reciben órdenes o información, utilizando mensajes en formatos predefinidos y codificados a través de algoritmos previamente instalados en los ordenadores atacados. Existen técnicas y herramientas que analizan el tráfico y detectan los mensajes "sin sentido" que se envían a estas redes, indicativos de una posible exfiltración de datos.

6. Algoritmos para generar dominios (*DGA,Domain Generation Algorithm*s). Esta técnica está inspirada en las listas de dominios. Su objetivo es desarrollar un algoritmo que genere en tiempo real un número ingente de dominios, de tal manera que cada cierto tiempo la exfiltración de información o el $C\&C$ va cambiando. De esta manera, se hace muy difícil su seguimiento con el objeto de que su seguimiento sea muy difícil. Detectar las amenazas, y eliminarlas, requiere de técnicas de análisis forense, que realicen ingeniería inversa del algoritmo utilizado para la generación de dominios antes de poder cortar la comunicación con el exterior.

7. Fast Flux Domain. Un solo dominio resuelve múltiples direcciones *IP* que van rotando según un modelo *Round–Robin*. Para ello el atacante crea una *botnet,* red de equipos infectados, dispersos por todo el mundo [38], utilizando como *proxy–caché* cada uno de ellos con un TTL (*Time To Live*) muy bajo, en general inferior a 5 minutos. Se pretende que la *IP* que pertenece a ese dominio cambie continuamente, para lo que va cambiando de *proxy*, de tal manera que si una *IP* es identificada como maliciosa por la electrónica de seguridad o por el administrador de la red, otras muchas estarán accesibles y disponibles para ser utilizadas como $C\&C$ o exfiltración de datos. Esta técnica está inspirada en los balanceadores de carga utilizados por sitios webs legítimos como *Google* o *Amazon*. Para combatirla, se utilizan técnicas de detección y ubicación del dominio, aunque resulta una tarea muy complicada.

2.2. *APTs*

2.2.1. Amenazas Persistentes Avanzadas

Una vez vista la evolución de las amenazas, en esta sección se revisan las definiciones de APT y sus fases, y se propone una definición unificada.

2.2.1.1. Evolución de las definiciones de *APT*

De las definiciones aportadas por instituciones, investigadores, directores de seguridad o empresas de prestigio dentro del sector de la ciberseguridad, se pueden destacar las siguientes definiciones:

1. N.I.S.T. (*National Institute of Standards and Tecnology*) define *APT* como [39]: "*Un enemigo que posee niveles sofisticados de conocimientos y recursos que le permitan atacar a sus objetivos mediante el uso de múltiples vectores de ataque.*

Una vez atacada la infraestructura y establecidos dentro de ésta, pivotan dentro de la infraestructura a los efectos de exfiltrar información, menoscabando, anulando o destruyendo aspectos críticos de la organización, o posicionándose para llevar a cabo estos objetivos en el futuro. La amenaza persistente avanzada se caracteriza por que:

- *Persigue sus objetivos repetidamente durante un período prolongado de tiempo.*
- *Se adapta a las contramedidas de la organización para mantenerse dentro de ésta.*
- *Está decidida a mantener el nivel de interacción necesaria para ejecutar sus objetivos."*

2. INCIBE (Instituto Nacional de Ciberseguridad) define *las APTs* de la siguiente manera [7]: *"Son un tipo de ciberataque selectivo y de gran sofisticación cuyos objetivos son el ciberespionaje, principalmente empresarial,*

gubernamental, militar, el robo y manipulación de información muy valiosa y sensible.

Constituyen un riesgo de ciberseguridad de mayor gravedad que los habituales ya que, a priori, poseen características que hacen que sus efectos sean mucho más silenciosos y, por ende, más dañinos.

Los principales rasgos que definen a las APTs son: ser capaces de perdurar en el tiempo (infectando por ejemplo una máquina o un servidor sustrayendo información poco a poco), poder aprovecharse de vulnerabilidades desconocidas oficialmente (lo que las hace pasar desapercibidas) y, sobre todo, tratarse de amenazas dirigidas contra un objetivo muy específico (habitualmente los recursos de una compañía).”

3. Colin Tankard, director de seguridad de Digital Pathways, en su artículo de 2011 en *Network Security* [40], define *el concepto de APT* : *“Amenaza persistente avanzada (APT) es un término acuñado en el último par de años para que una nueva generación de amenazas malintencionadas que utilizan diferentes vectores de ataque, de manera sigilosa para evitar ser detectados, con lo cual los hackers pueden tener el control de los sistemas durante largos períodos de tiempo.”*

4. Colin Tankard, posteriormente, redefine su concepto de *APT* [41]: *“Amenaza persistente avanzada, término que se usa para describir nuevos tipos de ataques que se caracterizan por el uso de múltiples métodos para acceder a los sistemas informáticos –incluyendo la Ingeniería Social, malware de día cero, troyanos– y comprometerlos, realizándolo de una manera sofisticada para evitar que sean detectados y permanecer durante largos períodos de tiempo en los sistemas comprometidos, no sólo con el objetivo de ganancias financieras, sino con objetivos múltiples.”*

5. Ivo Friedberg, Florian Skopik, Giuseppe Settanni y Roman Fiedler, en su artículo de *Computers & Security*, aportan la siguiente definición [42]:

“Amenaza persistente avanzada (también conocida como APT) es un ciberataque paulatino y silencioso cuyo objetivo es comprometer sistemas de información interconectados sin revelarse, sin ser detectados. A menudo, las

APTs utilizan una variedad de métodos de ataque al objeto de obtener acceso a un equipo y posteriormente extenderse por toda la red.

En contraste con los ataques tradicionales, no se utilizan para interrumpir los servicios, su objetivo es robar la propiedad intelectual, información sensible, documentos internos y otros datos.

Si un ataque tiene éxito, la detección oportuna es de importancia primordial para mitigar su impacto y evitar la propagación por toda la red.

Sin embargo, los incidentes de seguridad recientes, como la Operación ShadyRat, Operación Octubre Rojo o el descubrimiento de MiniDuke –por nombrar solo unos pocos– han demostrado que los mecanismos de seguridad actuales son en su mayoría insuficientes para evitar los ataques dirigidos y personalizados."

6. Paul Giura y Wei Wang, del *AT&T Security Research Center* de Nueva York, definen *APT* como [43]: *"Ataques que duran largos períodos de tiempo, dándose prioridad a permanecer latentes en los sistemas, realizados por atacantes bien entrenados y organizados, que utilizan un amplio espectro de tecnologías de intrusión en redes, cuya intención es infligir daño y provocar caídas de servicios o robar datos."*

7. Para la empresa de seguridad ZinkSecurity "thinking solutions", empresa de servicios y soluciones de Ciberinteligencia, una APT es [37]: *"Un adversario que posee niveles sofisticados de conocimientos y recursos significativos que le permitan crear oportunidades para lograr sus objetivos mediante el uso de múltiples vectores de ataque.*

Estos objetivos suelen incluir el establecimiento y ampliación de puntos de apoyo dentro de las infraestructuras de tecnologías de la información de las organizaciones, orientadas a los efectos de exfiltración de información, menoscabar o anular los aspectos críticos de una infraestructura, posicionándose para llevar a cabo estos objetivos en el futuro."

2.2.1.2. Fases de una *APT*

Fundamentándonos en lo expuesto en la sección 2.2.1.1, en los enfoques aportados por algunos autores [43, 44] y para entender mejor en qué consiste una APT, se describen a continuación las distintas fases por las que pasa una APT.

Fase 1. Recopilación de información, reconocimiento (*Information Gathering*). Los atacantes indagan acerca de los recursos del objetivo a atacar –empleados, empresas, relaciones con otras entidades– todo aquello que pueda ser aprovechado para recabar información de muy diversa índole creando perfiles por cada posible objetivo; aficiones, intereses, ocio, deportes, familiares, amigos, opiniones, tendencias políticas, sociales, económicas, contactos, emails, etc. Es lo que se conoce como "Ingeniería Social" [8].

Para la identificación y búsqueda de objetivos utilizan redes sociales (*LinkedIn, Facebook, Twitter, WhatsApp, Instagram*, etc.) o fuentes *OSINT* (*Open Source INTeligence*) [45]. Cuanta más información se obtiene, con mayor precisión se diseña el ataque. Esta fase es una de las partes más importante del proceso de intrusión, pues obtiene una visión holística, clara y concisa de su objetivo y cuáles son sus posibles vectores de ataque. La recopilación de información será recurrente a lo largo de todas las fases restantes.

Fase 2. Análisis de vulnerabilidades, punto de entrada. El atacante analiza, clasifica, desecha y organiza todos los resultados obtenidos en la fase anterior con la finalidad de encontrar fallos de seguridad y, a partir de éstos, elaborar el mejor método de acción y ataque.

Se suele simular en laboratorio la infraestructura a atacar con el propósito de evaluar posibilidades y discernir qué método es el más eficaz para explotar las debilidades de la víctima. El uso de escáneres para el análisis de puertos suele ser el procedimiento más habitual en esta etapa [46]. La herramienta por excelencia para esta técnica es *Nmap* o diseño de *scripts*, utilizando técnicas e ingenierías de la programación en lenguajes sofisticados de alto, medio y bajo nivel, entre otros: *LUA, Python, Perl, Ruby, C, C++* o Ensamblador.

Algunos autores destacan cómo el atacante prepara la irrupción contra

el objetivo a través de la herramienta más popular y utilizada, el correo electrónico. Un email infectado (*spearphishing email*) que pueda contener p.e. una invitación a un evento, unas entradas para un partido de futbol, o realizar una suplantación de un sitio web, utilizar *pendrives* infectados, puntos de accesos inalámbricos falsos, imágenes de códigos *QR* (evolución del código de barras) o esteganografía [47] –técnicas que permiten ocultar mensajes dentro de otros para que la comunicación pase inadvertida. El método elegido dependerá de la información recolectada en la fase 1.

El medio de ataque varía a medida que se van incorporando nuevos aspectos tecnológicos. El ataque suele desencadenarlo un malware de día cero (*zeroday*) [48] enviado a través de Ingeniería Social utilizando cualquiera de las técnicas anteriormente descritas.

Un ataque de día cero aprovecha vulnerabilidades en aplicaciones o sistemas que son desconocidas por los desarrolladores y/o fabricantes, que hasta ese momento y no han sido corregidas y que permiten la ejecución de un código malicioso en un sistema, lo que lo convierte en una de las armas más peligrosas en los ataques a las infraestructuras.

Una vez que la víctima abra el correo electrónico, conecte el pendrive, active el código *QR* con el móvil, etc., se creará un *backdoor* –puerta de entrada en la red– que sirve de punto de infiltración. A partir de esta puerta trasera, se generará un *RAT (Remote Access Toolkit)* [40]. Los *RAT* son conjuntos de herramientas cuya finalidad consiste en permitir ejecutar y llevar a cabo numerosas acciones para controlar de forma remota el equipo infectado. Robar credenciales y poder realizar innumerables acciones dentro de la red atacada son los grandes objetivos de estas herramientas de *hackeo*.

A partir de este punto, el atacante está dentro del sistema y procede a reiterar la recopilación de información –de nuevo la fase 1, de recopilación de la información– de todos los activos, al objeto de evaluar y decidir la manera de expandirse por toda la red.

Fase 3. Explotación, mando y control ($C\&C$). Una vez que el atacante está dentro del sistema, se procede a establecer una comunicación entre el atacante y la víctima, denominada *"Command and Control"* ($C\&C$). Esta comunicación va a permitir de dirigir y controlar forma remota el *malware*

utilizado, solapándose con procesos activos del sistema y provocar el menor ruido posible para no ser detectado.

Una vez que el atacante asegura una conexión *C&C*, recopila información acerca de la configuración de seguridad de la máquina vulnerada y la relacionada con el sistema, activa *keyloggers* (capturas de teclado) para capturar nombres de usuarios y contraseñas de acceso a recursos de la infraestructura, extrae la lista de contactos de los correos electrónicos y de los usuarios con acceso a esa máquina como objetivos de futuros ataques, accede a las carpetas compartidas dentro de la red, a la estructura de directorios del disco duro, a las aplicaciones instaladas, a la arquitectura de la red, etc. Todo ello de cara a preparar la siguiente fase.

Fase 4. Pivoteo o movimiento lateral. El objetivo de esta fase es desplazarse por la red e ir aumentando el número de equipos comprometidos, hasta llegar a conseguir el control total de toda la infraestructura *TIC* y, así, poder robar la información deseada (exfiltrar) [40].

A partir de aquí, el atacante salta de un equipo a otro de la red para recopilar credenciales de acceso, obtener privilegios y mantener el control de la infraestructura informática de una manera persistente, fiscalizando así todos los equipos que conforman la red de la víctima, realizando de una manera recurrente y silenciosa el mismo procedimiento de recolección de información que en la fase 3 – explotación, comunicación, mando y control.

Fase 5. Detección de activos. Una vez controlada la red y obtenidos credenciales de distintos niveles, el atacante procede a identificar y comprometer los activos que almacenan los datos de mayor interés. Se redunda dicha información en uno o más de los equipos comprometidos de la red que tienen acceso o comunicaciones con equipos de otras redes externas. Así, una vez que la información es segmentada, comprimida y encriptada será capaz de exfiltrarla en pequeños paquetes de información, evitando que las distintas medidas de seguridad de la infraestructura (antivirus, IDS, IPS, *firewalls*, etc.) lo detecten.

Fase 6. Exfiltración de datos. La información valiosa y dispersada por los distintos equipos bajo el control del atacante es transferida hacia servidores *ad* hoc ubicados fuera de la infraestructura atacada. El canal $C\&C$ se redunda como plan de contingencia, para que nunca se corte la comunicación con el exterior y que las transferencias de información no puedan ser detectadas por ningún dispositivo interno de la red que detecte mucho "ruido" y genere alertas al administrador.

2.2.1.3. Definición unificada de *APT*

Tomando como referencia las diferentes definiciones de APT, y teniendo en cuenta que un ataque debe cumplir con las fases descritas anteriormente, utilizar técnicas novedosas para ganar acceso a los objetivos y evitar todos los medios *hardware* y *software* que hayan sido implantados en la infraestructura, se propone la siguietne definición de APT:

Una APT es una modalidad de ciberataque cuyo principal objetivo es el robo de información. Es llevada a cabo por grupos bien organizados y financiados que rediseñan o desarrollan *exploits* (segmentos de código que utiliza el atacante [49]) para vulnerabilidades aún no descubiertas. Atacan las infraestructuras desde una red de ordenadores zombis –*botnets*– centrando su atención en los usuarios de la infraestructura, que constituyen su eslabón más débil. Consiguen información de sus objetivos a través de Ingeniería Social (redes sociales o fuentes *OSINT*), obteniendo una visión holística del objetivo que les permite recrearla en laboratorio. Acceden a la infraestructura a través de múltiples vectores de ataque evitando todas las medidas de seguridad. Una vez dentro, instalan herramientas de acceso remoto que les permitirá crear y redundar canales de comunicación con el exterior para robar la información, contando con planes de contingencia para dar continuidad al ataque en el caso de ser detectados.

Atendiendo a su acrónimo y continuando con su definición, se puede decir lo siguiente:

Amenaza. La Real Academia Española la define como "*Acción de amenazar. Dicho o hecho con que se amenaza. Delito consistente en intimidar a alguien*

con el anuncio de la provocación de un mal grave para él o su familia" [13].

Partimos de la premisa que la amenaza y, por ende, el ataque, no van a ejecutarse de una manera convencional. Los atacantes son profesionales con perfiles muy dispares, muy especializados y cualificados, organizados, bien financiados, que trabajan de una manera coadyuvada, que tienen una visión clara, concisa y holística del objetivo, así como lo que quieren conseguir, obtienen toda la información que necesitan y diseñan su estrategia de intrusión a los sistemas, creando el código necesario para conseguir "amenazar" a objetivos de cierta envergadura, y llevar a cabo, entre otras, actividades de espionaje industrial, acceder a cuentas bancarias, afectar a infraestructuras civiles o militares, a gobiernos y a administraciones.

Persistente. La Real Academia Española define persistente como *"Que persiste, mantenerse firme o constante en algo, durar por un largo período de tiempo"* [13].

El hecho de que un ataque se prolongue en el tiempo no debería ser un factor tan determinante. Existen sitios web que sufren intentos de denegación de servicio durante días o semanas y también ataques que se han llevado con precisión milimétrica, bien porque se conocía de antemano la infraestructura de la empresa, o bien porque se tenía localizada una vulnerabilidad que permitía una rápida explotación, sin tener que llevar a cabo posteriores intentos para acceder a la información u objetivo del ataque.

En cualquiera de los dos casos, se requiere un trabajo previo. Por tanto, la persistencia no radica en la duración del ataque sino en el tiempo empleado para su ejecución. Publio Cornelio tardó 6 días en conquistar Cartago Nova utilizando distintos vectores de ataque y meses en su planificación.

Ese tiempo "persistente" es una presencia prolongada dentro de la organización, de una manera meliflua, sutil y delicada, y se mide en la variedad de vectores de ataques que puedan utilizar tanto seres humanos como sistemas automatizados, pudiendo esperar de una manera latente durante largos períodos de tiempo, incluso meses, hasta que se den las condiciones óptimas para realizar el ataque.

Avanzada. La Real Academia Española define avanzada como *"Que se distingue por su audacia o novedad en las artes, la literatura, el pensamiento, la política, etc. Que aparece en primera línea, bien en cosas que están en primer término. Aquello que se adelanta o anticipa"* [13].

Por lo tanto, para considerarlo como tal, el ataque debe aportar alguna novedad en el campo de la seguridad e incluso ser específico, estar diseñado para el objetivo atacado y poseer, por definición, alguna primicia.

Partiendo de la premisa de que todas las vulnerabilidades son *bugs*, pero no todos los *bugs* son vulnerabilidades, la sofisticación del ataque deberá ser capaz de detectar las vulnerabilidades o *bugs* a niveles web, de red, de sistemas y de *software*.

En un concepto de ataque avanzado no tiene cabida la obsolescencia de los medio *hardware* y/o *software*, por lo que no podemos considerar que se trate de una *APT* cuando el ataque se realiza a equipos con *Windows XP* o servidores con *Windows* 2000 *Server* a través de técnicas de desbordamiento de *buffer*, que son técnicas utilizadas en los años 90. El ataque debería de ser capaz de evitar las medidas de seguridad de los sistemas y utilizar las técnicas de exfiltración de información expuestas anteriormente.

Deberá tener la capacidad de adaptación en el caso de ser detectado contando con planes de contingencia al objeto de dar continuidad al procedimiento, según la fase del ataque en la que se encuentre, y no ser repudiado por cualquier sonda o detector ubicado en la red para tal fin.

2.2.2. Escenarios posibles de ataque por *APT*

Los escenarios que se pueden dar en una *APT* son muy diversos, pero el procedimiento de ataque y la forma de proceder es coincidente en muchos de los esquemas que se pueden encontrar en la red.

Un esquema resumido, genérico y desde el cual se puede tener una visión de una *APT* corresponde a un informe de Trendmicro [50], empresa especializada en seguridad (figura 2.1).

Ampliando el trabajo de otros autores [43] del modo de proceder del ataque por *APT*, este esquema resumido se puede dividir en 3 escenarios.

- Escenario 1

Figura 2.1: Esquema resumido de posibles escenarios de *APT*

El objetivo consiste en robar ficheros del ordenador de un empleado que posee información de cierto valor.

El atacante ha realizado un trabajo previo de Ingeniería Social con el objeto de conocer la máxima cantidad de información posible acerca del objetivo. Con esa información, se apoya en algún *exploit* disponible o diseña uno que "*explote*" una o varias vulnerabilidades del ordenador de la víctima. Con el *exploit* embebido en una imagen, en un archivo con formato *pdf* o *doc*, etc. Utiliza un ordenador zombi –ordenador que va a ser utilizado sin que su usuario tenga conocimiento– para que en el caso de ser detectado le permita evitar descubrir quién es la persona física que está realizando el ataque. Desde ese ordenador zombi envía un *email* con un correo llamativo que despierte la atención de la víctima. El correo llevará un fichero adjunto que contiene el *exploit*. En cuanto la víctima abre el correo y el fichero adjunto, el *exploit* se ejecuta y el atacante levantará un *C&C* que le permitirá tomar el control de la máquina y robar la información deseada. La figura 2.2 muestra el escenario primero de ataque por *APT*.

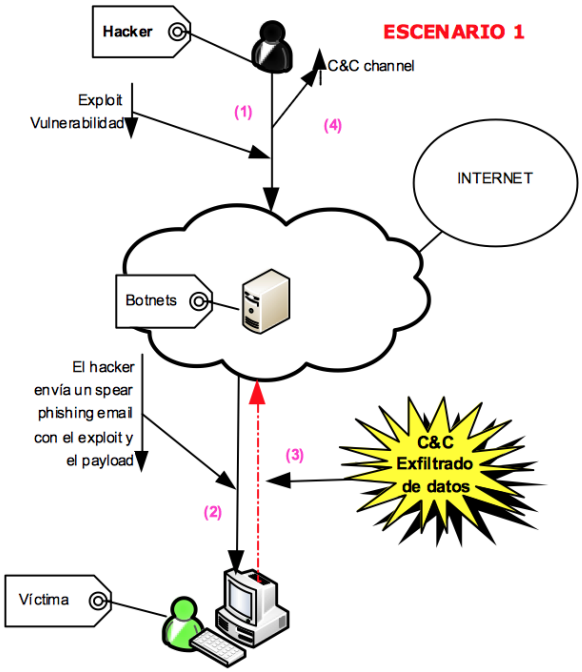


Figura 2.2: Escenario primero de ataque por *APT*

- Escenario 2

  El objetivo consiste en robar información ubicada en un servidor de base de datos a través del administrador de la base de datos.

  Con el mismo método presentado en el escenario primero y con el *exploit* cargado en el ordenador del administrador de la base de datos, el atacante puede perfectamente levantar una sesión de *meterpreter*, código ejecutado en la máquina remota que permite una línea de comandos para conseguir un control total del equipo atacado [51], pues gracias al canal $C\&C$ controla de manera remota todos los equipos comprometidos, lo que va a permitir, entre otras cosas, lanzar un *keylogger* para capturar las secuencias de entrada del teclado. Esta información es visualizada en tiempo real, por lo que cuando el administrador de la base de datos proceda a establecer la conexión, el atacante podrá ver las credenciales necesarias para el acceso. Con esas credenciales robadas, y con el ordenador del administrador comprometido, el atacante se co-

necta a la base de datos y procede a exfiltrar la información utilizando alguna de las técnicas expuestas anteriormente. La figura 2.3 muestra el escenario segundo de ataque por *APT*.

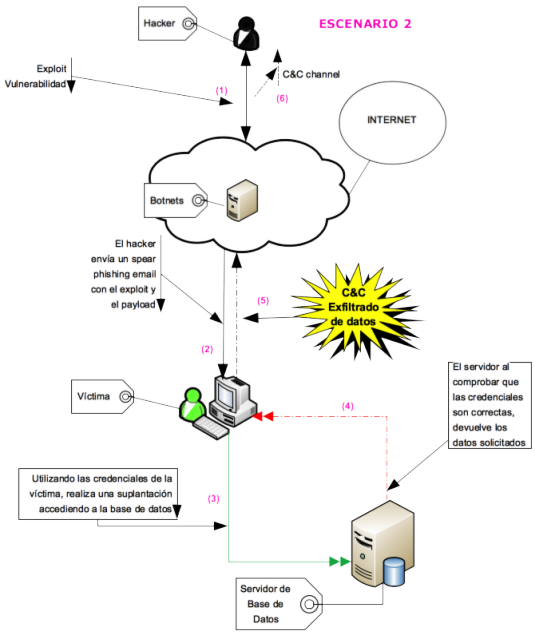


Figura 2.3: Escenario segundo de ataque por *APT*

- Escenario 3

  Es el más complejo. Un primer empleado controlado es utilizado como pivote para alcanzar a una segunda víctima, que suele ser el administrador de la red.

  Para atacar a la segunda víctima se puede utilizar otro *SpearPhishing email*, un mensaje *SMS*, o *IM* (*Instant Messaging*), etc., para así comprometer la segunda víctima, obtener sus credenciales, acceder a los datos y proceder a su exfiltración.

  La máquina comprometida permitirá, al igual que en el escenario segundo, montar una sesión de *meterpreter* y obtener control absoluto.

El atacante, gracias a esto, podrá realizar una suplantación del administrador, y el servidor de base de datos comprobará firmas digitales, usuarios, contraseñas, etc. Serán todas correctas, pues el atacante las hace en remoto utilizando el ordenador del administrador, con lo que el servidor y la electrónica de seguridad no detectarán nada anómalo. La figura 2.4 muestra el escenario tercero de ataque por *APT*.

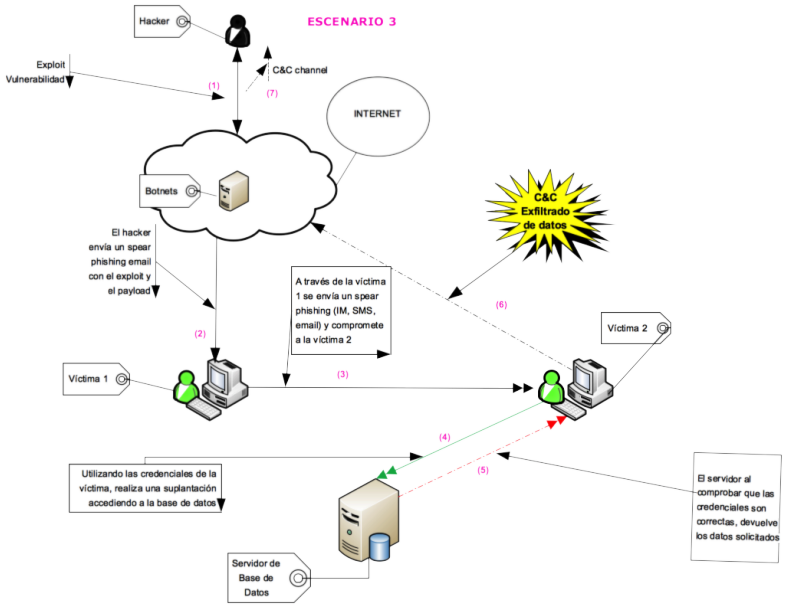


Figura 2.4: Escenario tercero de ataque por *APT*

2.2.3. Sistemas de detección de *APTs*

Existen varios marcos teóricos que proporcionan soluciones o predicen ataques de *APT* [42, 44, 52–61]. Todos ellos comparten una metodología común para diseñar un modelo e incluyen una etapa intermedia antes de realizar la clasificación. Existe una excepción [52] que no necesita ninguna etapa intermedia para conseguir la detección temprana de una *APT* utilizando el algoritmo de inferencia aproximada, *Belief propagation,* conjuntamente con técnicas de minería de datos. Este sistema utiliza conjuntos de datos de una

red privada corporativa para modelar las comunicaciones entre dispositivos internos a la red interna (*hosts*), y entre *hosts* y dominios externos con la ayuda de un gráfico bipartito cuyas aristas enlazan los *hosts* y dominios que están conectados al menos una vez durante el período de observación. Aplicando técnicas de reducción de dimensionalidad, el sistema genera una lista de dominios sospechosos.

Algunos modelos que incluyen etapas intermedias son los siguientes:

El sistema de *Attack Pyramid* [53] es un modelo inspirado en el concepto de árbol de ataque [54,55]. Modela un ataque como una pirámide cuya cúspide representa el objetivo de la *APT*, mientras que sus caras representan los caminos y barreras que ha de superar la amenaza para llegar a la cúspide. Los autores definen un contexto de ataque en una red corporativa privada y generan una alerta al sistema.

Otros autores proponen utilizar la correlación de eventos de red para detectar ataques [42] a partir de conjuntos de datos semisintéticos [56]. Para ello almacenan dos conjuntos de datos que no tienen ningún ataque con el objetivo de desarrollar el modelo y un tercer conjunto con anomalías artificiales para entrenar y evaluar la eficacia del modelo creado. Este enfoque concluye que el modelo es eficaz combinando conjuntos de datos generados dentro de una organización, sin conocimiento previo de sus estructuras. La clasificación es una etapa intermedia, pero el objetivo final es tener un modelo adaptado a cada infraestructura y actualizarlo automáticamente basándose en las entradas de datos.

Hay trabajos que combinan sistemas automáticos creados a partir de procesamiento *Big Data,* aprendizaje máquina y conocimiento experto [57]. Además, hay realimentación continua para que el modelo se actualice en base a nuevos casos anómalos. Los autores utilizan conjuntos de datos generados por servidores web para detectar ataques a los servicios de Internet y conjuntos de datos de *firewalls* para el análisis de posibles exfiltraciones de datos. La parte automática combina tres modelos que funcionan en paralelo (*Matrix Decomposition* [58], *Replicator Neural Networks* y *Density based outlier analysis* [59]). Esta propuesta concluye que la inclusión del conocimiento humano logra mejorar mejorar la detección en un factor superior a 3, mientras

que el número de falsos positivos es 5 veces menor.

A continuación se relacionan algunos de los trabajos más relevantes enfocados a detectar *APTs* o comportamientos que puedan ser catalogados como tales:

Detection of Early-Stage Enterprise Infection by Mining Large-Scale Log Data [52]. En el trabajo, los autores proponen un sistema basado en la teoría de grafos que entrena un modelo de regresión lineal puntuando cada dominio al que se accede. Dirige su atención hacia el tráfico destinado a nuevos dominios o dominios singulares que son accedidos por un pequeño número de equipos "posiblemente infectados". Establece un umbral para los dominios, y los que superan dicho umbral se consideran potencialmente como dominios *C&C*. Aplican el algoritmo a *logs* del *proxy* e identifican cientos de dominios maliciosos pasados por alto por otros productos relacionados con la seguridad y demuestran la efectividad en dos diferentes *datasets*, uno contiene registros *DNS* y otro *web proxy logs*.

Por cada iteración del algoritmo, los dominios son puntuados acorde a distintas características y similitudes con dominios detectados en iteraciones anteriores. El algoritmo puede ser aplicado, ya sea con "*hints*" (semilleros de dominios de sospechosa reputación), o con dominios conocidos obteniéndose *blacklists*, listas negras, comerciales que contienen los indicadores de reputación de los dominios a los cuales la empresa ha accedido.

El sistema es elástico, con pequeñas cantidades de dominios *random* utilizados por los atacantes, y especifica que analizar el tráfico requiere asumir otros métodos y modelos de detección. Utiliza *logs* sintéticos generados por *Los Alamos National Lab (LANL)*.

Ciberseguridad Inteligente [60]. El autor muestra un modelo teórico denominado *MSOR* (Modelo de Sistema Ofensivo de Referencia), involucrado en escenarios de conflictos de 3 niveles: estratégico, operacional y táctico. El modelo se basa en 4 procesos: surgimiento del conflicto, planificación, estrategia, ejecución de operaciones y evaluación de los resultados. La evaluación de este modelo teórico establece como objetivo principal obligar al atacante a ajustar permanentemente sus objetivos y recursos, lo que implicaría para el

atacante un aumento de costes e incremento de tiempo a la hora de lograr sus objetivos, buscando inferir en el bucle de decisión del adversario, indicando que la defensa debe ser más rápida que el ataque. Este hecho se enfatiza en escenarios con amenazas de tipo *APT*.

New rules for combating new threats [61]. En este trabajo, el autor realiza una descripción general de una *APT*, dando importancia al comportamiento proactivo para prevenir el ataque, entre otros: evaluar los datos y clasificarlos en grupos de riesgo, utilizar algoritmos de encriptación para la información sensible, separar la información en distintos servidores, encriptar el tráfico hacia los servidores, auditar el tráfico de entrada/salida hacia esos servidores, controlar usuarios y aplicaciones que acceden a los servidores, usar doble *token* de autenticación en el momento del acceso en lugar de utilizar solamente usuario y *password,* y actualizar a la última versión el *software* todos los dispositivos electrónicos de la red.

Advanced persistent threats: Minimising the damage [44]. El autor hace referencia a la complejidad de los ataques y lo que pueden perdurar en el tiempo sin que la víctima se percate de la situación, enfatizando que las herramientas de seguridad actuales no son suficientes ni capaces de detectar y/o bloquear una *APT,* y que solamente es posible analizar los rastros que van dejando en los *logs,* que las pistas están en las huellas digitales que va dejando a su paso, siendo posible detectarla a través de estos *logs* y en función del comportamiento que va teniendo a lo largo de su incursión dentro de la red. Para ello es importante establecer parámetros de comportamientos en la red con el objeto de examinar cualquier desviación de esos comportamientos.

Combating advanced persistent threats: From network event correlation to incident detection [42]. En este trabajo, sus autores aportan un método para detección de *APTs* y anomalías realizando pruebas con conjuntos de datos semisintéticos. El método propuesto está diseñado como una extensión de la seguridad, especializado en los "*packet-level*" de los *IDS* como mejora de sus resultados, no pretendiendo sustitución alguna de los mecanismos tradicionales de detección. Actúa como adición a los existentes respecto a

la monitorización de las infraestructuras. En el método, cada registro de *log* es atomizado –dividido en partes sumamente pequeñas– creando una huella dactilar de ceros y unos, obteniendo resultados de 3 % de falsos positivos en los *logs* sintéticos. Llega a la conclusión de que los mecanismos de prevención de seguridad basados en firmas son insuficientes para la detección de amenazas, ya que la tasa de detección de diferentes tipos de anomalías varía dependiendo de la complejidad del conjunto de datos, así como del tipo de anomalías generadas diariamente. En las redes complejas, una anomalía que ocurre de nuevo podría no ser detectada por una sola regla. Sólo mediante la combinación de varias reglas en el modelo, el enfoque es capaz de detectar la anomalía y evaluarla de forma fiable.

Using Large Scale Distributed Computing to Unveil Advanced Persistent Threats [53]. Los autores plantean una representación de una *APT* como un ataque piramidal, colocando el ataque en la punta de la pirámide, y en los planos laterales los entornos por los cuales evoluciona. El método busca detectar comportamientos maliciosos a través de la correlación de los eventos que se produzcan y queden registrados en los *logs* de los *IDS*, *IPS*, *DNS*, *email*, *HTTP*, de red, etc. Para el proceso de correlación utiliza técnicas de computación distribuida, como *MapReduce*. La capacidad de procesamiento variará en función de los *workers* (equipos que realizan una tarea determinada) utilizados en el *cluster* (conjunto de equipos distribuidos que funcionan como si fueran uno solo).

En su enfoque para la correlación y detección, utiliza distintos planos o contextos: físico, de usuario, de red, de aplicación y otros. Por cada plano y, basado en reglas de asignación, identifica tres tipos de eventos: candidatos, sospechosos y ataques. Se definen los perfiles de los posibles objetivos de *APT* (*ID* de empleados, *IP* de los servidores de base de datos, puertos de servicios, etc.), estos perfiles se utilizan para agrupar eventos, a su vez agrupados en contextos y, en base a una serie de reglas, se les asignan valores de: confianza, riesgo y umbral de alarma.

Cada vez que se registra un evento en la infraestructura, se realiza una consulta a una base de datos histórica, y todos los eventos similares se devuelven como el resultado de la consulta. Un módulo especial de alerta aplica

las reglas de detección y correlación para cada evento dentro del contexto e informa de la posible actividad maliciosa.

Concluye que el volumen de los datos utilizados como evidencia de ataques está creciendo en volumen, cada vez a mayor velocidad y con más diversidad, por lo que es cada vez más difícil detectar *APTs.*

2.2.4. Casos reales de ataques por *APT*

A continuación se exponen los ataques más significativos desde 2004 hasta 2017, reflejándose la fecha, el método de ataque utilizado y las empresas o países afectados.

- Informe Mandiant [62] sobre $APT1$ (2004 – 2006), la Unidad 61398 y la ciberguerra entre países.

  APT1 es la Unidad 61398 del Ejército Popular Chino. *APT1* robó centenares de Terabytes de información de al menos 141 organizaciones. Para ello utilizaron técnicas de *spearphishing* dirigidas contra personas concretas de la organización, de las que previamente habían obtenido datos mediante técnicas *OSINT*.

El trabajo realizado por Sanz [63] describe casos muy significativos de *APTs* muy famosas entre las que están las siguientes:

- Operación *Ghost Net* (2008).

  Detectada en junio de 2008, esta operación tenía como objetivos ministerios de asuntos exteriores de diferentes países, entidades bancarias y medios de comunicación. No obstante el principal objetivo era la organización tibetana del Dalai Lama.

  El método de ataque fue a través de un *spearphishing*. Infectaron un total de 1.295 ordenadores en 103 países. Los atacantes enviaron un correo malicioso con un enlace de descarga de temas, principalmente aficiones, en los cuales estaban interesados las víctimas que habían sido averiguados con herramientas de Ingeniería Social.

Cuando la víctima accedía al enlace suministrado, se ejecutaba un *exploit* que instalaba un conjunto de herramientas de *hacking* denominadas *RAT* (*Remote Access Toolkit*). Estas herrmientas se encargaban, entre otras cosas, de establecer uno o varios canales de comunicación con el exterior (*C&C*) para robar la información.

Se pudo determinar que el 70 % de los servidores *C&C* estaban en China.

- Operación *Night Dragon* (2009).

  Esta operación tenía como objetivos compañías del sector energético (petroleras, empresas de energías renovable, gas, petroquímicas, etc.). Utilizaba *SQL Injection* (técnica de ataque sobre base de datos) como método principal de ataque y *spearphishing* como método secundario. Robaron gran cantidad de ficheros e información de gran interés de forma cifrada a sus servidores *C&C* externos.

- Operación Aurora (2010).

  Tenía como objetivo compañías comerciales potentes y de prestigio, con solvencia e influencia en todos los niveles, entre las que destacan: *Google, Adobe, Yahoo, Morgan Stanley.*

  En el caso de *Google* el ataque fue contra *Gmail*, donde comprometieron una gran cantidad de cuentas de correo de disidentes chinos, ciudadanos americanos y europeos *VIP*, personas de relevancia relacionadas con la lucha por los derechos humanos.

  El método aprovechaba una vulnerabilidad *zeroday* de *Internet Explorer* acompañado de un *spearphishing* que invitaba al usuario a visitar una página en la cual se activaba la vulnerabilidad y se descargaba el troyano. Una vez instalado el troyano y cargado el *RAT,* se creaba la conexión cifrada *C&C.*

  Como consecuencia de esto, *Google* retiró sus servidores de China y los alojó en Hong–Kong.

- Operación *Shadows in the Cloud* (2010).

A mediados de 2010, sus objetivos eran sistemas de información gubernamentales, académicos, corporativos y militares de la India, la *ONU* y el Tibet.

En esta ocasión utilizaron la técnica de *spearphishing*. Enviaron un correo masivo a multitud de direcciones de correo en la India con un archivo malicioso adjunto que activaba el *RAT* y el *C&C*. Los atacantes consiguieron robar documentos *"top secret"* enviándolos a una dirección de *Yahoo*, que posteriormente era descargada por el *C&C* empleando la red *TOR*.

- Ataque a *RSA* (2011).

  Empresa reconocida por sus *tokens SecurID* (clave de seguridad adicional para garantizar el acceso, que puede ser *hardware* o *software*), métodos y medios criptográficos utilizados por numerosas empresas norteamericanas y agencias gubernamentales.

  El ataque se produjo a través de un *spearphishing* a empleados de bajo perfil de la infraestructura. Explotaban un *zeroday* de *Adobe Flash Player* instalando el *RAT* y el *C&C*. Los atacantes entraron en el sistema robando el código fuente de *SecurID*, lo que obligó a la empresa *RSA* a cambiar 40 millones de *tokens*.

- Ataque a *Lockheed Martin* (2011).

  Lockheed Martin es una multinacional de la industria aeroespacial con grandes recursos en tecnología avanzada y guerra global. Es el contratista militar más grande del mundo por volumen de ingresos. Tan solo tres meses después del ataque a *RSA*, *Loockheed Martin* sufrió un severo ataque a través de su red privada virtual (*VPN*), que estaba basada en *tokens* de *SecurID*.

- Operación *ShadyRat* (2011).

  Esta operación fue detectada a mediados del año 2011, pero llevaba en funcionamiento desde 2006, afectando entre otros a organizaciones gubernamentales y no gubernamentales, al Comité Olímpico Internacional y a la Agencia Mundial Antidopaje.

El ataque se hacía a través de un *spearphishing,* aprovechando un fallo de seguridad de *Microsoft Excel*, descargando el *RAT* y creando el *C&C*. Se estima que en los cinco años ha podido descargar alrededor de *1PB* de datos de los equipos infectados.

- Operación *Byzantine Hades* (2011).

  Activa desde 2005, tenía como objetivos a sistemas gubernamentales estadounidenses, franceses y alemanes.

  El ataque era el mismo que en los anteriores; un *spearphishing* en función de la vulnerabilidad a explotar e instalación en la víctima del *RAT* y el *C&C*.

- Duqu (2012).

  Utiliza la Ingeniería Social para ciberespionaje a países como Irán, Sudán, Francia y Hungría. Sus objetivos están relacionados con sistemas de control de la producción. Se desconoce su autoría y se inició en el año 2011 [64].

- Madi (2013).

  Atacan a gobiernos y entidades financieras de EE.UU., Israel y Pakistán a través de técnicas de Ingeniería Social con múltiples canales *C&C*.

- *Animal Farm* (2014).

  Se atribuye a países de lengua francesa, atacando a gobiernos, activistas, medios de comunicación, organizaciones humanitarias y militares utilizando Ingeniería Social. Sus ataques iban dirigidos, entre otros, a Siria, Irán, China, USA, Alemania [64].

- Machete (2014).

  Relacionado con temas políticos y militares, ataca a países de habla hispana, Rusia, Malasia y China. Utilizando técnicas de Ingeniería Social propagan el ataque a través de dispositivos USB infectados [64].

- *Equation* (2015).

Se propaga por USB, CD y por replicación. Infecta el *firmware* del disco duro. Sus principales objetivos eran empresas relacionadas con nanotecnología, energía nuclear, militar, aeroespacial, etc., ubicadas en países como Rusia, Afganistán, Pakistán o China [65].

- Regin (2016).

  Es la primera plataforma de ataque cibernético conocida por penetrar y monitorizar redes GSM, además de realizar otras tareas "estándar" de espionaje. Ataca a gobiernos e instituciones financieras de países como Afganistán, Siria, Rusia o Alemania. Se desconoce su origen y el método utilizado para su propagación, y continúa realizando ataques en la actualidad [64].

- Partido Demócrata Americano (2016).

  Dos grupos de espionaje ruso, identificados como *APT28* y *APT29,* consiguen comprometer las redes y activos asociados a las elecciones en Estados Unidos. Acceden a la infraestructura *TIC* del Partido Demócrata, a sus correos internos e incluso de miembros de la campaña de Hillary Clinton. La información es robada y difundida en Internet a través de *Wikileaks* o en páginas web. En el ataque, por un lado, insertaron un código malicioso en la infraestructura y, por otro, a través de un *spearphishing,* enviaron correos maliciosos con un enlace a una suplantación de una página web que solicitaba un cambio de contraseña de sus cuentas de correos electrónicos [66].

- Poseidón (2017).

  Se atribuye a países de habla portuguesa. Ataca a gobiernos, instituciones financieras o televisiones de países como Francia, Rusia, India o Emiratos Árabes. Utilizan *exploits* e Ingeniería Social [64].

La figura 2.5 muestra las *APTs* del año 2010 al 2015, indicando el momento en que empezaron, así como los años durante los cuales estuvieron activas [64].

La mayoría de las soluciones que se plantean en la actualidad para combatir las *APTs* son comerciales, y están directamente relacionadas con la

Figura 2.5: *APTs* activas entre los años 2010 y 2015 [64]

adquisición de sistemas de monitorización de red para intentar detectar una intrusión y hacerlo en su estadio más temprano posible. Estas soluciones son caras, de código cerrado, sus algoritmos no son públicos.

2.3. *Firewalls*

Dentro de esta subsección se describen los sistemas de cortafuegos o *firewalls*, la información que registran de la red así como la información que aportan sus registros (*logs*).

2.3.1. Definición de *firewall*

Un *firewall* es un dispositivo de seguridad de red que protege la infraestructura frente a intrusiones, monitorizando el tráfico entrada/salida, salida/entrada, permitiéndolo o bloqueándolo de acuerdo con un conjunto de reglas de seguridad definidas por el administrador del sistema, y que almacena toda la información en archivos de *log*.

Existen distintos tipos de *firewall*: por *hardware* (en *routers*), por *software* (embebidos en el sistema operativo), por *software* comercial (el que está integrado en las *suites* de los antivirus –conjunto de aplicaciones de seguridad en un solo producto) o en *appliance* (solución combinada de seguridad integrada en *hardware*, que incluye su propio sistema operativo y todo el *software* necesario para su funcionamiento).

Atendiendo al modelo *OSI* (*Open System Interconnection*), existen *firewalls* de nivel 3 (capa de red), de nivel 4 (capa de transporte) y de nivel 7 (capa de aplicación).

La mayoría de las infraestructuras *TIC* utilizan mecanismos de seguridad para detectar actividad maliciosa con el objeto de proteger sus activos y la información que poseen. Estos mecanismos de seguridad generan ficheros con registros de la actividad de la red, orientados a la seguridad de la infraestructura. A continuación se relacionan los registros de *logs* que generan los principales mecanismos de seguridad:

- Antivirus. Genera registros de *logs* en los cuales se almacenan todas

las instancias de *malware* detectadas, archivos en cuarentena, los intentos de desinfección del sistema, análisis realizados, actualizaciones del *software*, ficheros de firmas, etc.

- *IDS/IPS*. Sistemas de detección de intrusos y sistemas de prevención de intrusiones. Proporcionan un registro de información detallada sobre posibles comportamientos sospechosos, sobre posibles ataques, así como de cualquier intrusión o acción maliciosa contra el sistema.

- *Software* de acceso remoto. El acceso remoto se concede a menudo y se fija a través de una red privada virtual (*VPN*). Los registros de *logs* de *VPN* contienen información de los accesos que han tenido lugar, así como de los intentos de conexión, sean exitosos o fallidos, indicando por cada usuario las fechas y horas de conexión, las cantidades de datos enviados y de datos recibidos, el uso de recursos, etc.

- *Proxy–cache* . Son dispositivos intermedios entre el usuario y el sitio *web* al que acceden. Los *logs* de estos dispositivos permiten conocer todas las *URLs* a las que acceden los usuarios, así como los accesos que son permitidos y/o denegados.

- *Software* de gestión de vulnerabilidades. Este *software* permite analizar los activos, equipos y dispositivos implementados en la infraestructura, evaluando sus vulnerabilidades. Generan un informe de *log* de gran tamaño que proporciona detalles del tipo de vulnerabilidad, posibles vectores de ataques, configuraciones, etc.

- Servidores de autenticación. También conocidos como servidores de inicio de sesión único, en sus *logs* guardan información de cada intento de establecimiento de sesión, de cada autenticación, su origen, el nombre de usuario, si la conexión ha tenido éxito o ha fracasado, la fecha y la hora.

- *Routers*. Entre otras cosas, los *routers* pueden ser configurados para permitir o para bloquear ciertos tipos de tráfico de red basándose en algún tipo de política y son útiles para identificar intrusiones. Es posible

detectar y bloquear los ataques más comunes, como por ejemplo los basados en Denegación de Servicio *(DoS), IP spoofing, Death Ping, Land Attack, IP de longitud cero, Smurf Attack, UDP port loopback, Snork Attack, TCP null scan y TCP SYN flooding*. En sus *logs* encontramos información del tráfico que va pasando por ellos.

- *Firewall*. Al igual que los *routers*, los *firewalls* pueden controlar de manera muy exhaustiva el tráfico de la red sobre la base de políticas de acceso. Al ser el propio dispositivo el que enruta las *VLans* de una infraestructura, también controla todo el tráfico de entrada/salida *(inbound/outbound)*. Generan un registro de *logs* muy detallado y completo.

2.3.2. Archivos de log

Según el *NIST (National Institute of Standards and Tecnology)* [67], un *log* es un registro de los eventos que ocurren dentro de los sistemas y redes de una organización. Cada entrada contiene información relacionada con un evento específico que se ha producido dentro de un sistema o red y que puede proceder de muchas fuentes, entre otras: *software* de seguridad, *software* de antivirus, cortafuegos, sistemas de detección de intrusos, sistemas de prevención, sistemas operativos en servidores, estaciones de trabajo, equipos de red, aplicaciones etc.

El número, volumen y variedad de *logs* enfocados y reorientados a la seguridad ha aumentado considerablemente, derivado de la necesidad de disponer de *logs* de gestión, de incidentes, de transmisión, de almacenamiento, de análisis de eliminación de los datos de registro, etc.

Los registros de *log* contienen información relacionada con diferentes tipos de eventos que ocurren dentro de redes y sistemas. Su análisis ayuda a la identificación de los incidentes de seguridad, violaciones de política, actividad fraudulenta, problemas operacionales, etc. También son útiles para realizar auditorías, análisis forense, *pentesting* (ataque a un sistema con el objeto de encontrar debilidades en su sistema de seguridad), etc. Así, permiten al administrador de la infraestructura establecer criterios, identificar anomalías,

tendencias operacionales y problemas a largo plazo.

El *firewall* permite obtener *logs* de:

- Tráfico de la red (*IN/OUT*). Registra el flujo de la información dentro de la infraestructura, cualquier tipo de petición y de posible respuesta.

- Eventos. Registra todas las actividades administrativas como son los *downloads* o *backups* de la configuración del *firewall*, y eventos de sistema, enrutamiento, *VPN* (*Virtual Private Network*), usuarios, alta disponibilidad, *suite* de seguridad, redes inalámbricas, protocolos o demonios del sistema (programas que se ejecutan en segundo plano).

- Seguridad. Registra ataques de virus o intentos de acceso al sistema, que pueden proceder de: Antivirus, control de aplicaciones, *DLP* (*Data Leak Prevention*), *IPS* (*Intrusion Prevention*), filtrado Web, filtrado *email*.

Los *logs* generados por el *firewall* utilizado en los experimentos no utilizan ningún formato estándar de facto como puedan ser *WELF* (*WebTrends Enhanced Log File Format*), *ELFF* (*Extended Log File Format*), *ELF* (*Extended Log Format*), *ArcSight* (correlador de *HP*). Su contenido y diseño obedece a la experiencia recopilada por la empresa propietaria del *firewall* a lo largo de los años en el uso de las *TIC*.

En el apéndice B se describen los campos que componen el registro de *log*.

Cada registro de *log* incluye un valor que estima el nivel de alerta asociado a cada evento. Esta estimación está basada en la programación interna del *firewall*, que bien puede no coincidir con los criterios del administrador de la red. La tabla 2.1 muestra los diferentes niveles de alerta posibles, junto con su descripción.

Nivel	Tipo	Descripción
0	Emergencia	El sistema está inaccesible o no responde
1	Alerta	Se requiere una acción inmediata
2	Crítica	La funcionalidad está afectada
3	Error	Se ha detectado un error y la funcionalidad puede verse afectada
4	Aviso	La funcionalidad podría verse afectada
5	Notificación	Comportamiento inocuo
6	Información	Información general a cerca del sistema

Tabla 2.1: Niveles de estados de alerta preestablecidos en el Firewall

De todos los tipos de *logs* de alerta que ofrece el *firewall*, nos centraremos en los *logs* de tráfico y en los niveles 5 y 6, ya que los demás disparan alertas por sí solos, y no es necesario tratarlos, siendo labor del administrador del sistema tomar las precauciones o medidas necesarias para solventarlos.

2.4. Minería de datos

Las herramientas de minería de datos (*data mining*) extraen información válida, previamente desconocida, comprensible y útil, de bases de datos de gran tamaño, y utilizan dicha información para tomar decisiones [68].

La tabla 2.2 muestra algunas operaciones de minería de datos [69].

Operaciones	Técnica de Minería de Datos
Modelado predictivo	Clasificación
	Predicción de valores
Segmentación de la base de datos	Agrupaciones demográficas
	Agrupaciones neuronales
Análisis de enlaces	Descubrimiento de asociaciones
	Descubrimiento de patrones secuenciales
	Descubrimiento de similitudes en secuencias temporales
Detección de desviaciones	Estadística
	Visualización

Tabla 2.2: Operaciones y técnicas utilizadas en minería de datos

El modelado predictivo se desarrolla utilizando una técnica de aprendizaje supervisado, que tiene dos fases: entrenamiento y test. El entrenamiento permite construir un modelo utilizando un conjunto de entrenamiento. Una vez

entrenado el modelo, se comprueba su precisión y características utilizando datos nuevos o reales [69].

La segmentación de base de datos consiste en dividir un volumen importante de información en una cantidad desconocida de *clusters* o grupos que contienen registros similares. Para ello se utilizan métodos de aprendizaje no supervisado [69].

El análisis de enlaces se fundamenta en la búsqueda de nexos o asociaciones entre datos individuales o conjuntos de ellos utilizando reglas de asociación [69].

La detección de desviaciones es una técnica relativamente nueva que busca las excepciones a una norma conocida o a una cierta expectativa [69].

Para llevar a cabo cualquiera de las operaciones o técnicas descritas anteriormente es necesario utilizar herramientas que permitan [69]:

1. Preparar, limpiar, transformar y muestrear los datos.

2. Seleccionar las operaciones de minería de datos, que irán en función del algoritmo elegido, de las variables predictoras, de los resultados, de la rapidez de la fase de entrenamiento y de la fase de test.

3. Escalabilidad, para aplicar procesamiento paralelo a volúmenes crecientes de información.

4. Comprender los resultados, facilitando medidas que describan, entre otros: la precisión del modelo, matrices de confusión, estadísticas o sensibilidad de los resultados.

La figura 2.6 resume las técnicas de minería de datos basadas en aprendizaje automático, mostrando alguno de sus algoritmos más representativos [9].

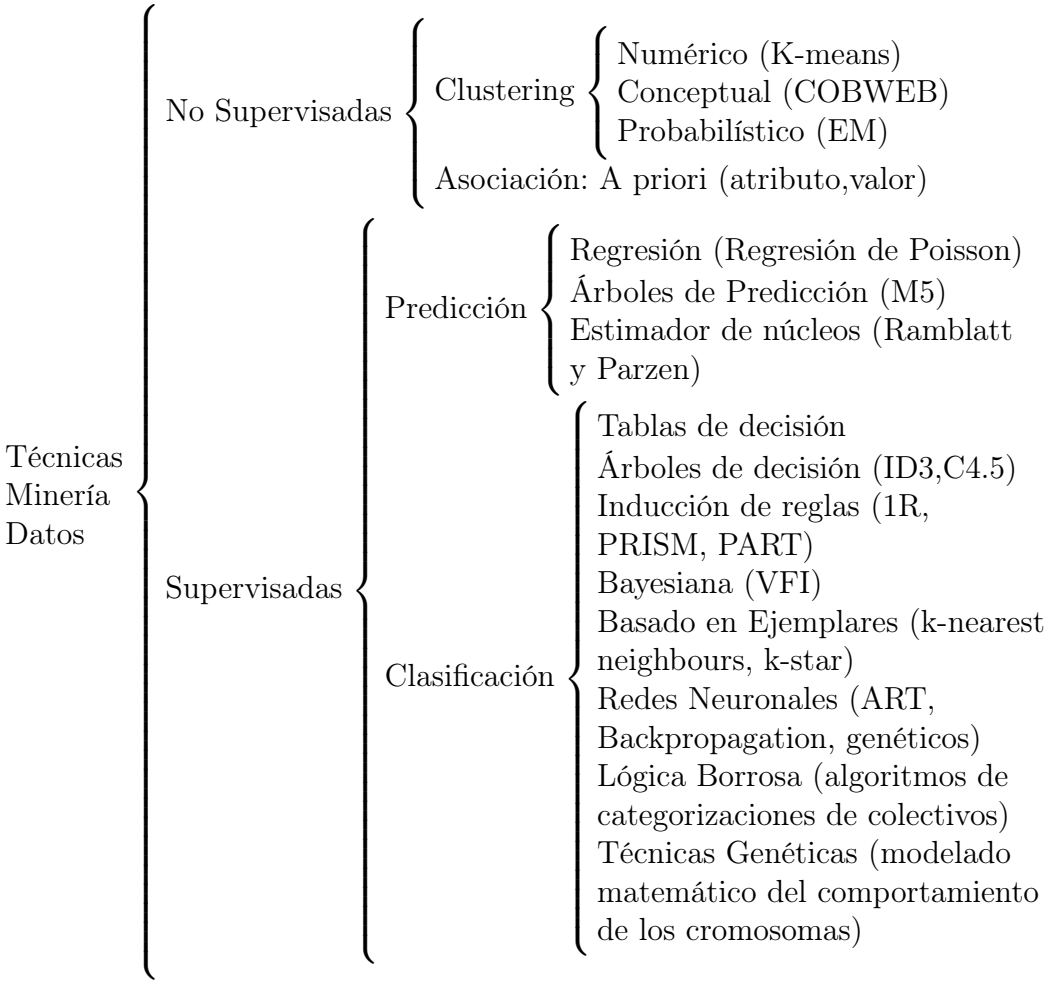


Figura 2.6: Resumen de las técnicas de minería de datos basadas en aprendizaje automático

2.5. Sistemas de aprendizaje automático

La teoría computacional del aprendizaje (*Computational Learning Theory*), aprendizaje automático o aprendizaje máquina (*machine learning*) [70–75] es una rama de la inteligencia artificial que trata de buscar patrones a partir de la información suministrada a un sistema de aprendizaje.

Existen dos tipos de aprendizaje: supervisado y no supervisado. El aprendizaje supervisado se caracteriza por trabajar con datos etiquetados, es decir, ejemplos de los que se conoce su clasificación correcta, que cumplen de mane-

ra satisfactoria el objetivo del aprendizaje. En el aprendizaje no supervisado los datos no están etiquetados, por lo que el sistema no tiene criterio para "saber" si dichos datos cumplen el objetivo del aprendizaje. Se busca que el sistema sea capaz de reconocer y catalogar las entradas.

Las técnicas de Aprendizaje Automático (AA) permiten realizar análisis de grandes cantidades de datos –minería de datos– con el objetivo de poder extraer reglas, regularidades, patrones, etc. Esto va a permitir que a partir de una hipótesis que explique una problemática se puedan realizar experimentos para contrastar dicha hipótesis.

Los algoritmos con mayor éxito inicial suponían que las entradas al sistema eran conjuntos de observaciones previamente clasificadas por alguna fuente externa (aprendizaje supervisado). Posteriormente se adaptaron estas técnicas para afrontar otros problemas en los que no existían entradas clasificadas (aprendizaje no supervisado).

Los objetivos que se buscan con el aprendizaje son los siguientes:

- Adquirir nuevos conceptos: a partir de ejemplos de un concepto, se puede generar un modelo computacional que permita resolver una tarea.
- Reducir los recursos necesarios.

Las acciones necesarias para diseñar un sistema de AA son las siguientes [75]:

- Determinar la función objetivo.
- Recopilar un número razonable de casos o experiencias. Si el número de hipótesis es muy alto, también deberá serlo el número de casos que se han de recopilar, siendo necesario encontrar un equilibrio entre el número de casos, de hipótesis y la calidad de los resultados.
- Elegir los atributos predictivos. Se deben seleccionar "a priori" las variables, los atributos que vayan a influir en la realización efectiva del objetivo.
- Establecer los posibles valores de los atributos predictivos seleccionados:

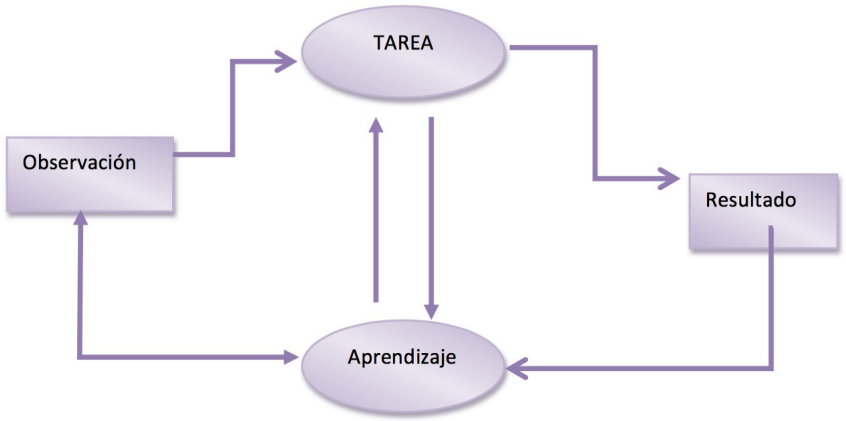


Figura 2.7: Esquema genérico de un sistema de aprendizaje automático

  - Discretos: Conjunto reducido de valores.
  - Agrupados: Cuando los valores son muchos pero finitos y se discretizan englobándolos en pocas categorías.
  - Continuos: Tienen infinitos valores posibles. Algunas técnicas de *AA* no pueden manejar este tipo de valores o generan muchos errores.

- Seleccionar las instancias de entrenamiento. Se deben seleccionar los conjuntos adecuados de muestras con la mayor calidad y la mínima cantidad de errores posibles, utilizando una cantidad porcentual de las muestras para entrenamiento y otra para validación.

La figura 2.7 muestra una descripción somera de un sistema de aprendizaje automático.

Cuando no se conoce de antemano la solución de un problema de clasificación, se puede plantear como un problema de aprendizaje. Los sistemas de *AA* mejoran su comportamiento a partir de la experiencia acumulada, etiquetando de una manera binaria o numérica las observaciones. Para ello es necesario preprocesar los datos con el objeto de adecuar las entradas de datos a las estrategias que se vayan a aplicar. Un caso típico está presente en la minería de datos, donde aparecen atributos, elementos o campos con un gran número de valores diferentes. Muchos algoritmos consideran de forma

aislada cada uno de los valores posibles y los resultados son erróneos. Para solventar esto, se recurre a seleccionar atributos relevantes, se establecen rangos de valores siempre que estos permanezcan dentro del objetivo marcado. Otros casos consideran establecer umbrales que discriminen de una manera más selectiva el volumen de datos.

Para el diseño del sistema de aprendizaje automático se establecen las fases mostradas en la tabla 2.3.

	Aprendizaje automático	**Minería de datos**
AP1	Determinar la función objetivo	Obtener los datos de la fuente de entrada
AP2	Elegir los atributos predictivos	Tratamiento para la extracción de los datos de las entradas
AP3	Codificar los valores de los atributos	Cómo son generadas las fuentes de entrada de datos
AP4	Recopilar un número razonable de ejemplos de entrenamiento y aplicar los distintos *SAA*	Preprocesamiento de las entradas
AP5	Seleccionar las instancias de entrenamiento	Ruido en entradas defectuosas o con incertidumbre
AP6	Estudiar la técnica más adecuada	Representación, normalización de los campos seleccionados y valoración, representación de resultados

Tabla 2.3: Fases para el diseño del sistema de aprendizaje automático

En este trabajo se han utilizado técnicas inductivas supervisadas como los árboles de decisión *ID3–C4.5* y clasificadores *Bayesianos*, técnicas de aprendizaje por agrupación no supervisada como *k–means*, técnicas de aprendizaje puramente no simbólico como redes neuronales artificiales y técnicas de aprendizaje supervisado como máquinas de soporte vectorial (*SVM*).

2.5.1. Árboles de decisión *ID3–C4.5*

Los árboles de decisión son técnicas de AA utilizadas en minería de datos aplicadas a: catalogaciones de clientes en banca, compañías de seguros, robótica, predicción de enfermedades, etc. Buscan predecir comportamientos futuros a partir de los comportamientos pasados.

Gráficamente, un árbol de decisión está formado por un conjunto de nodos (atributos) que contienen arcos o ramas (posibles valores de los atributos) y hojas (resultado binario de la decisión del árbol).

Las reglas en un árbol de decisión son de la forma IF-THEN-ELSE anidadas sobre sus atributos y sus valores.

Algunos autores [75–79] exponen que se utilizan con mucha frecuencia por sus destacables ventajas: aprenden funciones de valores discretos, admiten ejemplos con ruido y obtienen un conjunto de expresiones fácilmente

transformable en un conjunto de reglas.

El *árbol C4.5* [77, 78] y su antecesor, el *ID*3 [76], forman parte de la familia de los *TDIDT (Top Down Induction Decision Trees)*. La estructura principal de ambos métodos es la misma, ya que *C4.5* es una extensión de *ID3.*

Los estudios y comparativas realizado por algunos autores [79] muestran que C4.5 tiene una buena proporcionalidad de tasa de error y velocidad, así como un comportamiento altamente eficiente. Construye un árbol de decisión mediante el algoritmo "divide y vencerás", dividiendo recursivamente árbol y subárbol, basado en la ganancia de información, que permita la elección del atributo que genere subárboles que contengan grupos de ejemplos que pertenezcan a una sola clase. Con esto se busca el mayor grado de homogeneidad posible dentro de las distintas clases con el objetivo de que cada partición contenga ejemplos que pertenezcan a una sola clase o bien que no se puedan realizar más particiones. Evalúa la información utilizando criterios de Entropía, Ganancia o Proporción de Ganancia, que se describen más adelante.

Este método es capaz de trabajar con conjuntos de datos con mucho ruido, lo que permite aplicarlo a situaciones en las cuales puedan aparecer ejemplos mal clasificados, atributos o clases erróneos o sin valor. La figura 2.8 muestra un ejemplo de atributos (nodos), clases (posibles valores del atributo objetivo), valores (ramas).

Un aspecto vital para los algoritmos de árboles de decisión es establecer el mejor criterio para la división de los datos. Para realizar esta selección se utiliza el concepto de Entropía de Shannon [80], que permite calcular el grado de incertidumbre que tiene una muestra. Valores de entropía menores se corresponderán con predicciones mejores.

$$Entropía(E) = -\sum_{i=1}^{C} p_i log_2(p_i) \tag{2.1}$$

Donde:

E es el conjunto de muestras.

p es la proporción de ejemplos que hay de la clasificación i en la muestra.

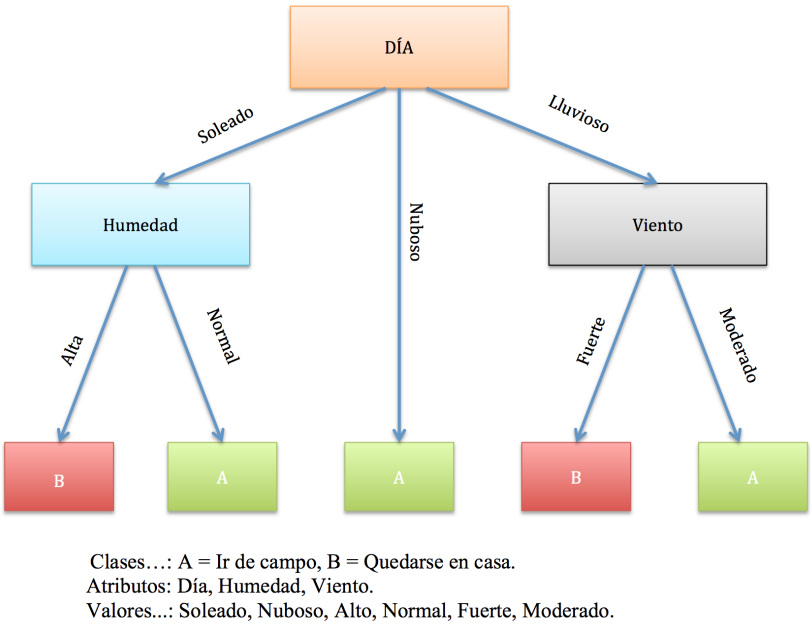


Figura 2.8: Ejemplo de clase, atributo y valor

C es el número de clases distintas.

Si consideramos E el conjunto de ejemplos donde el atributo A_i tiene el valor v_j, podemos calcular su entropía de la siguiente manera [75]:

$$Entropía(E, A_i, v_j) = -\sum_{k=1}^{C} \frac{n_{ijk}}{n_{ij}} log_2(\frac{n_{ijk}}{n_{ij}}) \tag{2.2}$$

Donde:

n_{ijk} es el número de ejemplos con valor v_j del atributo A_i que pertenecen a la clase C_k.

n_{ij} es el número de ejemplos con valor v_j del atributo A_i.

C es el número de clases.

La ganancia de información es la reducción de la entropía causada por la división del conjunto de datos en los valores de un determinado atributo. Su cálculo permite seleccionar el atributo que crea ramas más homogéneas y sea seleccionado como nodo de decisión [78].

$$Ganancia(S, A) = E_t(S) - \sum_{v \in Valores(A)} \frac{\mid S_v \mid}{\mid S \mid} E_t(S_v) \tag{2.3}$$

Donde:

$Valores(A)$ es el conjunto de posibles valores del atributo A.

$\mid S_v \mid$ es el número de ejemplos en S etiquetados con v.

$\mid S \mid$ es el número total de ejemplos.

$E_t(S_v)$ es la entropía de ejemplos etiquetados con v.

Se observa que tomando como referencia la medida de entropía, se calcula la entropía de cada rama del conjunto de datos divididos en función de los atributos. Se suman proporcionalmente las entropías calculadas de cada rama. La diferencia entre las entropías es la Ganancia de Información. El objetivo es alcanzar un valor cero de entropía para conseguir ejemplos con igual clasificación dentro de la rama.

La figura 2.9 muestra el pseudocódigo genérico del algoritmo [81].

1. Comprobar los casos base
2. Para cada atributo *"A"*
   encontrar la ganancia de información normalizada de la división de *"A"*
3. "A_mejor" es el atributo con la ganancia de información normalizada más alta
4. Crear un nodo de decisión que divida *"A_ mejor"*
5. Repetir en las sublistas obtenidas por división de *"A_ mejor"*, y agregar estos nodos como nodos hijos.

Figura 2.9: Pseudocódigo genérico del algoritmo *Decision Tree*

El algoritmo aplica recursividad sobre todas las ramas del árbol hasta que el conjunto de ejemplos pertenezcan a la misma clase. Pero la casuística permite que se agoten los atributos y existan ejemplos con distintos valores de clase, o que se elija un atributo para un nodo de decisión, pero que no exista ningún ejemplo para la rama generada por dicho atributo. En estos casos se etiqueta el nodo de la hoja con la clase mayoritaria y se selecciona un atributo de acuerdo a la heurística anteriormente definida.

2.5.2. Clasificadores *Bayesianos*

El Teorema de Bayes tiene múltiples aplicaciones. Entre ellas, sea h la mejor hipótesis, H el espacio de hipótesis y E el conjunto de ejemplos, utilizando el Teorema de Bayes sobre E podríamos calcular la mejor $h_i \in H$.

Utilizando el desarrollo matemático de los autores [75, 82, 83], se obtiene el siguiente resultado:

$$p(h_i \mid E) = \frac{p(E \mid h_i)p(h_i)}{p(E)} \tag{2.4}$$

la mejor hipótesis h cumplirá que:

$$h = argmax_{h_i \in H} \frac{p(E \mid h_i)p(h_i)}{p(E)} \tag{2.5}$$

Como $p(E)$ es un valor constante, entonces:

$$h = argmax_{h_i \in H}\ p(E \mid h_i)p(h_i) \tag{2.6}$$

Obtención del clasificador bayesiano [75, 82, 83]:

$$p(C_k \mid X_i) = \frac{p(X_i \mid C_k)p(C_k)}{p(X_i)} \tag{2.7}$$

Donde:

$p(A \mid B)$ es la probabilidad que un ejemplo B pertenezca a una clase A.

$p(C_k)$ es la probabilidad de que un ejemplo cualquiera pertenezca a la clase C_k.

$p(X_i)$ es la probabilidad de que ocurra el ejemplo X_i.

Elegiremos la clase C_k que maximice dicha probabilidad. $p(X_i)$ no afecta al cálculo del máximo, ya que es igual para cada clase, por lo que la clase c se calcula como:

$$c = argmax_{C_k \in C}\ p(C_k \mid X_i) = argmax\ p(X_i \mid C_k)\ p(C_k) \tag{2.8}$$

Siendo C el conjunto de clases. El aprendizaje consistiría en calcular, a partir de los ejemplos, los valores $p(X_i \mid C_k)p(C_k)$.

El paradigma *Naïve Bayes* se basa en dos premisas establecidas sobre las

variables predictoras y las variables a predecir [84]:

1.- Las variables a predecir son excluyentes. Es decir, una variable a predecir tomará uno de sus posibles valores.

2.- Las variables predictoras son condicionalmente independientes dada la variable a predecir. Si se conoce el valor de la variable a predecir, el conocimiento del valor de cualquiera de las variables predictoras es irrelevante para el resto de las variables predictoras. Matemáticamente se expresa de la siguiente manera:

$$p(X_i \mid C_k) = \prod_{j=1}^{a} p(A_j = V_{jv_j} \mid C_k) \tag{2.9}$$

Donde $p(A_j = V_{jv_j} \mid C_k)$ es [75]:

- Si el atributo A_j es discreto, su resultado es el número de ejemplos de la clase C_k que poseen el valor V_{jv_j} para el atributo A_j dividido entre el número de ejemplos de la entrada que son de la clase C_k.

- Si el atributo A_j es continuo, se asume que los valores de A_j en las instancias de la clase siguen una distribución normal, y la probabilidad se calcula:

$$p(V_{jv_jk}) = \frac{1}{\sqrt{2\pi\sigma_{jk}^2}} e^{-1/2\frac{(v_j-\mu_{jk})^2}{\sigma_{jk}^2}} \tag{2.10}$$

La utilización del teorema de Bayes necesita que las probabilidades condicionales sean mayores que 0. Se soluciona fijando un umbral mínimo $\epsilon > 0$ o utilizando la estimación–m [75] para estimar las probabilidades condicionales $p(A_j = V_i \mid C_k)$.

$$p(A_j = V_i \mid C_k) = \frac{n_{ijk} + m \times p}{n_k + m} \tag{2.11}$$

Donde:

n_{ijk} es el número de ejemplos que tienen el valor V_i del atributo A_j y pertenecen a la clase C_k.

m es el parámetro corrector denominado tamaño de la muestra equivalente.

n_k es el número de ejemplos de la tabla que pertenecen a la clase C_k.

p es la estimación previa de la probabilidad que se desea estimar. Si p carece de valor o es nulo, la distribución es uniforme.

Si r es el número de valores posibles del atributo A_j, $p = \frac{1}{r}$.

La figura 2.10 muestra el pseudocódigo del algoritmo para clasificadores *Bayesianos* [75].

Función NAIVEBAYES (*E, A, V, C*): M_c, M_{ad}, M_{ac}
E: Conjunto de ejemplos
A: Conjunto de atributos con sus posibles valores
$V_{[a]}$: matriz de valores del atributo discreto $a \in A$
C: conjunto de clases
$M_c\,[c]$: matriz de probabilidades de las clases $c \in C$
inicialmente, $\forall c \in C$, $M_c\,[c] = 0$
$M_{ad}\,[a, v, c]$: matriz de probabilidades de los valores $v \in V_{[a]}$
de los atributos discretos $a \in A$ para cada clase $c \in C$
$M_{ac}\,[a, 2, c]$: matriz de medias (posiciones 1) y varianzas (posiciones2)
de los atributos continuos $a \in A$ para cada clase $c \in C$
Para cada $e \in E$
c = ejemplo.clase(e);
$M_c\,[c] = M_c\,[c] + 1$;
Para cada $a \in A$
Si a es discreto THEN
Para cada $v \in V_{[a]}$
Para cada $c \in C$
n = NúmeroEjemplosClase(a, v, c);
$M_{ad}\,[a, v, c] = mín\left\{\frac{n}{M_c[c]}, \epsilon\right\}$
ELSE (a es continuo),
Para cada $c \in C$
$M_{ac}\,[a, 1, c] = Media\,(a, c, E)$
$M_{ac}\,[a, 2, c] = Varianza\,(a, c, E)$
Para cada $c \epsilon C$
$M_c\,[c] = \frac{M_c[c]}{|E|}$;
Return M_c, M_{ad}, M_{ac}

Figura 2.10: Pseudocódigo del algoritmo de los clasificadores *Bayesianos* [75]

El algoritmo guarda en la matriz $M_c[c]$ el número de ejemplos de cada clase $c \in C$ que serán utilizados posteriormente. El segundo bucle calcula por cada atributo sus valores correspondientes a las matrices de probabilidades condicionales. Si el atributo es discreto, calcula la probabilidad de pertenecer a una clase en función de su valor. La función *NúmeroEjemplosClase* calcula el número de ejemplos en los que el atributo a tiene el valor v para la clase c. Si el atributo es continuo, calcula la media y la varianza de cada uno de sus valores.

Finalmente termina de calcular la probabilidad de ocurrencia de una clase, dividiendo el número de ejemplos de la clase entre el número de ejemplos total.

2.5.3. *K–means clustering*

Las técnicas de aprendizaje no supervisadas por agrupamiento [75, 85, 86] resuelven el problema del etiquetado previo con metodologías que agrupan los datos en las clases (particiones) más probables. El objetivo primario de estos métodos consiste en encontrar las clases más probables y con mayor sentido en las que se puedan dividir.

El algoritmo de *clustering k–means* utiliza técnicas de optimización iterativas para abordar este problema [86].

Dado un conjunto de puntos $X = \{x_i\}, i = 1, \ldots, n$ con el objeto de ser agrupados en un conjunto de K *clusters* $C = \{c_k, k = 1, \ldots, K\}$, el algoritmo *K-means* encuentra una partición tal, que el error cuadratico entre la media empírica de un *cluster* y los puntos en el *cluster* se minimiza.

Sea μ_k la media del cluster c_k. El error cuadrático entre los puntos de un *cluster* se define como

$$J(c_k) = \sum_{x_i \in c_k} \| x_i - \mu_k \|^2 \tag{2.12}$$

El objetivo de *K-means* es minimizar la suma de errores cuadráticos sobre todos los K *clusters*,

$$J\left(c_k\right) = \sum_{k=1}^{K} \sum_{x_i \in c_k} \parallel x_i - \mu_k \parallel^2 \tag{2.13}$$

Este algoritmo inicialmente selecciona aleatoriamente un elemento como prototipo inicial de la clase. Una vez elegido el elemento, empieza a iterar repetidamente y en cada iteración agrupa el elemento a tratar en la clase que más se le parece hasta que converge la clase, y recalcula el elemento prototipo de cada clase como el punto medio del centroide de todos los elementos que pertenecen a la clase. La figura 2.11 muestra el pseudocódigo [85].

Introducir los datos del estudio D y el número de *clusters*, k
Paso 1. Inicializar k centroides aleatoriamente.
Paso 2. Asociar cada valor de D con el centroide más próximo. Esto dividirá D en k *clusters*.
Paso 3. Recalcular la posición de los centroides.
Repetir los pasos 2 y 3 hasta que no se produzcan más cambios en los grupos creados en D.

Figura 2.11: Pseudocódigo del algoritmo *k–means*

La función termina cuando el algoritmo converge. Un criterio de convergencia, podría consistir, por ejemplo, en comprobar si durante dos iteraciones no cambian las posiciones de los centroides. Otro criterio de convergencia podría considerar que la distancia de cualquier elemento a su centroide sea menor que un determinado umbral.

Si en una clase se tienen m ejemplos, cada ejemplo tiene a duplas (atributo, valor), el centroide c se calcula como [75]:

$$c = \left[\frac{\sum_{i=1}^{m} v_{i1}}{m}, \frac{\sum_{i=1}^{m} v_{i2}}{m}, \ldots, \frac{\sum_{i=1}^{m} v_{ia}}{m} \right] \tag{2.14}$$

Donde:

v_{ij} es el valor del atributo j para el ejemplo i.

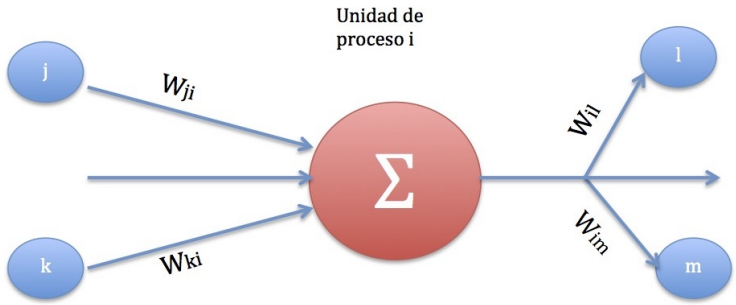


Figura 2.12: Esquema general de una unidad de proceso de una Red Neuronal Artificial

2.5.4. Redes Neuronales

Las Redes Neuronales Artificiales están inspiradas en el funcionamiento del sistema nervioso y hormonal de humanos y animales [75] [87, 88].

El sistema de comunicación neuronal, cuya unidad más esencial es la neurona, se compone de tres partes:

1. Los receptores, encargados de recoger la información en forma de estímulos.

2. El sistema nervioso, que recibe la información, la almacena y, una vez elaborada, la reenvía a los efectores y zonas del sistema nervioso.

3. Órganos efectores, se encargan de ejecutar las órdenes que reciben. Las respuestas pueden ser motoras o glandulares.

Existen modelos muy diversos de redes neuronales en función de las filosofías de diseño, de las reglas de aprendizaje y de las funciones de construcción de las respuestas. Entre las más conocidas están las redes supervisadas y las auto–organizativas; sin embargo, para todas ellas la estructura básica es la misma (figura 2.12 [75]).

Las Redes Neuronales Artificiales son sistemas computacionales formados por un conjunto de unidades de proceso llamadas neuronas interconectadas entre sí, capacitadas para determinar la función implícita que mejor caracterice a todos los elementos de una muestra de la manera más exacta posible.

Las entradas pueden provenir de las salidas de otras células o de las entradas de la propia red de neuronas, estableciéndose una conexión. Cada conexión tiene asociado un peso representado por:

$$W_{1i}, W_{2i}, \ldots, W_{ni} \tag{2.15}$$

La célula genera una única salida que puede ser la entrada a otra célula o una de las salidas de la propia red. El valor de cada entrada a una célula es igual a la suma de los valores de salida de las células con las que está conectada ponderados por sus respectivas conexiones. El resultado final es la salida que generará la célula, que se calcula de la siguiente manera [75]:

$$S_j = F\left(\sum_{i=0}^{n} S_j W_{ij}\right) \tag{2.16}$$

Donde:

S_j es la salida de la célula j-ésima.

W_{ij} es el peso de la conexión entre la célula i y la j.

$F(x)$ es una función no lineal de tipo umbral.

Una muestra i vendrá definida por un vector de entrada $\overrightarrow{e_i}$ y uno de salida $\overrightarrow{s_i}$. Definiremos $S(\overrightarrow{x})$ como la salida que genera la red neuronal al introducirse el vector $\overrightarrow{x}$ en la entrada. Para que la muestra esté caracterizada por la red neuronal se deberá de producir una salida s_i cuando se introduce una entrada e_i:

$$S\left(\overrightarrow{e_i}\right) = \overrightarrow{s_i}$$

Donde:

$S\left(\overrightarrow{x}\right)$ es la salida que produce la red neuronal al introducirse como entrada el vector $\overrightarrow{x}$.

Teniendo en cuenta que tanto los vectores de entrada como los de salida son numéricos, se puede calcular la diferencia entre el valor obtenido por la red neuronal de $S(\overrightarrow{x})$ y el asociado a su entrada $\overrightarrow{x}$.

Se dice que la red neuronal ha conseguido completar su tarea de aprendizaje cuando el error calculado sobre el conjunto de muestras de aprendizaje está por debajo de un cierto umbral [75].

$$\frac{1}{n}\sum_{i=0}^{n}(S(e_i)-S_i)^2 < \theta \tag{2.17}$$

Donde:

$S(e_i)$ es el valor obtenido como salida por la red neuronal.

S_i es el valor asociado a la entrada.

Las entradas son numéricas en forma de matriz siguiendo el esquema atributo, valor.

Para el banco de pruebas se ha utilizado *Probabilistic Neural Networks (PNN)* [89]. Su algoritmo genera reglas basadas en datos numéricos. Cada regla se define y almacena como una función de *Gauss*. La distribución de probabilidad de cada clase la modela a través de una combinación o mezcla de las funciones de *Gauss* almacenadas. El algoritmo contiene tantas neuronas como reglas, por lo que no es viable para grandes volúmenes de datos, ya que el tamaño de la red neuronal sería enorme. Para solventar esto se utiliza el algoritmo *Dynamic Decay Adjustment* (*DDA*), que permite la construcción automática de la red neuronal incluso para conjuntos de datos muy grandes. Esto lo consigue reduciendo el tamaño de la red neuronal construyéndola y optimizándola dinámicamente.

Para ello durante el entrenamiento el algoritmo establece un umbral mínimo θ^- y otro máximo θ^+. El umbral máximo establece la probabilidad de que la clasificación sea correcta. El umbral mínimo establece que la probabilidad de una clasificación incorrecta sea menor o igual al umbral.

Las clases se consideran correctamente clasificadas cuando están por encima del umbral máximo; erróneas o con probabilidad de error en la clasificación, cuando están por debajo del umbral mínimo; e inciertas cuando se ubican en la zona de conflicto.

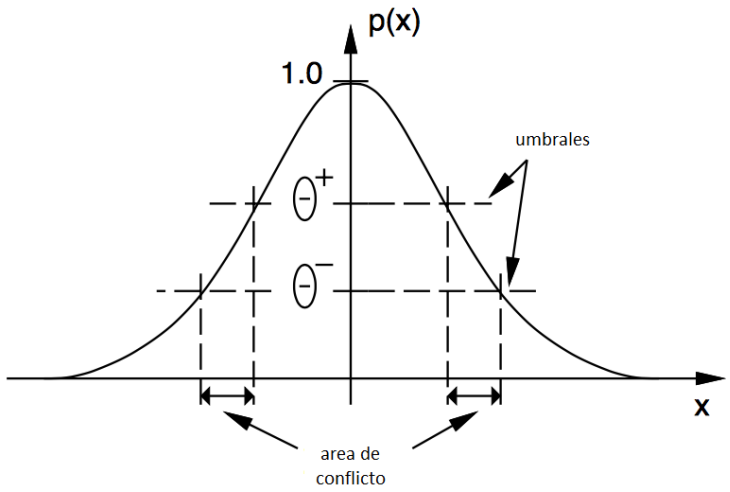


Figura 2.13: Umbrales utilizados en el algoritmo *DDA*

La figura 2.13 muestra la función de *Gauss* con los umbrales del algoritmo *DDA* y la zona de conflicto [89].

2.5.5. Máquinas de soporte vectorial (*SVM*)

Las máquinas de soporte vectorial (*SVM*, *Support Vector Machine*) tienen su origen en los trabajos sobre la teoría del aprendizaje estadístico, y fueron introducidas en los años 90 por *Vladimir Vapnik y Corinna Cortes* [90].

En la génesis de la investigación de las *SVM*, éstas fueron ponderadas para la resolución de problemas de clasificación binaria. Su continua evolución ha permitido extrapolar el modelo para resolución de problemas de diversa índole como: regresión, agrupamiento, multiclasificación, etc. [90–92].

Sus fundamentos teóricos gozan de gran robustez y han facultado su utilización en otros campos de lo más dispar: visión artificial y reconocimiento de caracteres, categorización de texto e hipertexto, clasificación de proteínas, procesamiento de lenguaje natural, análisis de series temporales, etc.

En el campo del aprendizaje estadístico es importante la construcción de hipótesis teniendo en cuenta los siguientes aspectos:

1. Minimización del riesgo empírico, para construir hipótesis cuyo margen

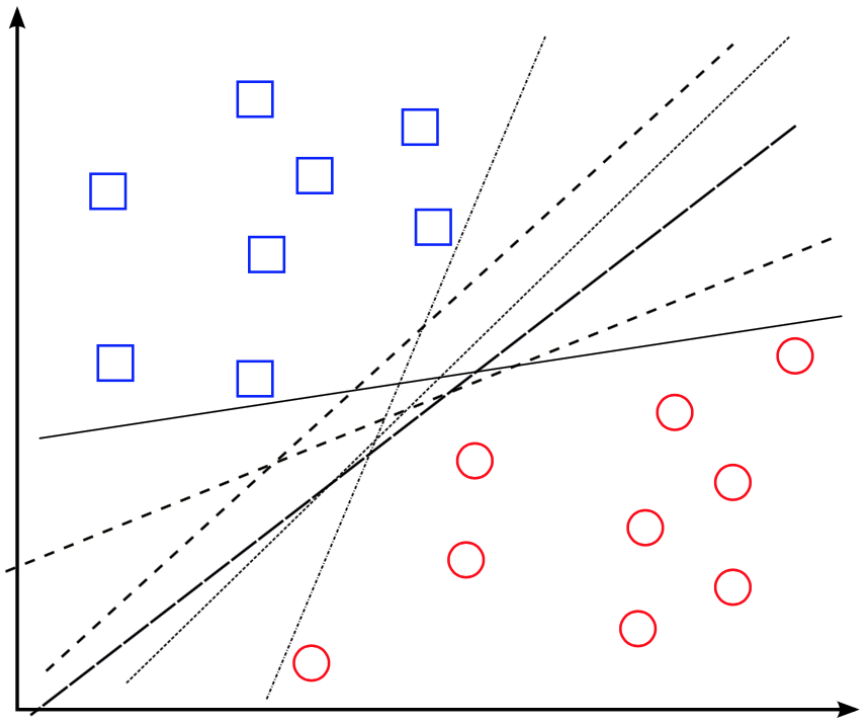


Figura 2.14: SVM: Ejemplos de hiperplanos de separación

de error sea mínimo o nulo, esto quiere decir que, teniendo una serie de muestras de entrenamiento, la solución pasa por encontrar la recta que permita una clasificación correcta de las muestras, p.e. puntos rojos, puntos azules. Ceñidos a este ejemplo, el objetivo sería encontrar el hiperplano óptimo que coloque correctamente agrupados dichos puntos por encima o debajo del hiperplano, minimice el error de clasificación y sean asignados correctamente a una clase.

2. Minimización del riesgo estructural, cuyo objetivo es construir hipótesis que minimicen el riesgo de cometer errores en clasificaciones futuras. Es decir, cuando se analizen con nuevas muestras, tendrá que asignarlas correctamente a la clase que le corresponda: por encima o por debajo del hiperplano.

La figura 2.14 muestra algunos de los infinitos hiperplanos posibles que permiten dividir el conjunto de ejemplos en dos clases.

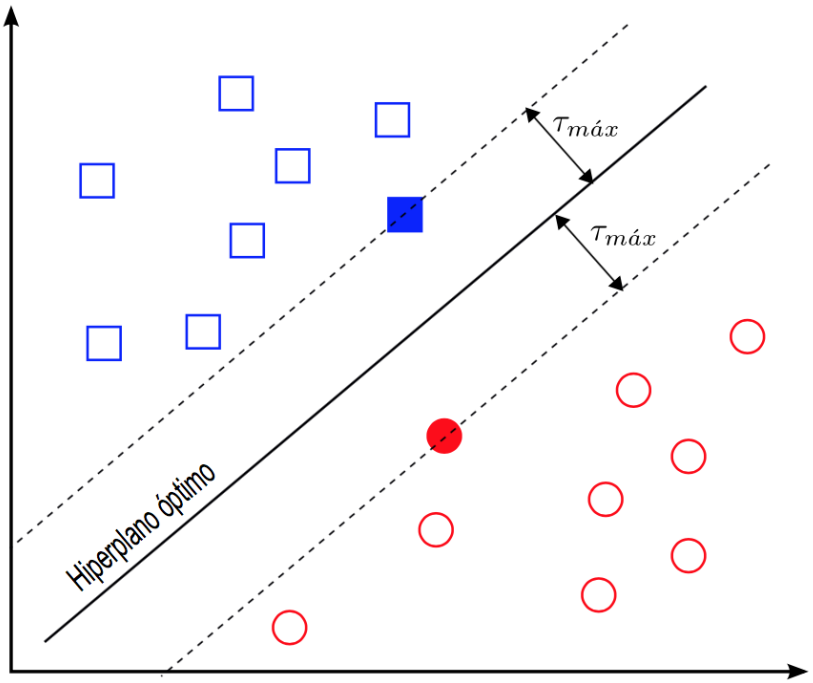


Figura 2.15: SVM: Hiperplano de separación óptimo

La figura 2.15 muestra el hiperplano de separación óptimo e ideal como una función lineal que es capaz de separar dicho conjunto sin error. Un hiperplano óptimo se denominará así si el margen de separación entre ambos planos es de tamaño máximo. El concepto de margen máximo está relacionado directamente con la capacidad de generalización del hiperplano de separación, de tal forma que, a mayor margen, la equidistancia entre los ejemplos de la clase será mayor.

La mayoría de los métodos de aprendizaje se ajustan para minimizar el riesgo empírico. En las *SVM* radica en la minimización del riesgo estructural. *Vladimir Vapnik* en su trabajo "*The Nature of Statistical Learning Theory*" [93], desarrolló toda una teoría que permite la construcción de hiperplanos que solventan y minimizan el riesgo estructural.

Dentro de la tarea de clasificación, las *SVM* pertenecen a la categoría de los clasificadores lineales, puesto que inducen separadores lineales, también llamados hiperplanos, ya sea en el espacio original de los ejemplos de entrada, si éstos son linealmente separables o cuasi separables (ruido), o en un espacio

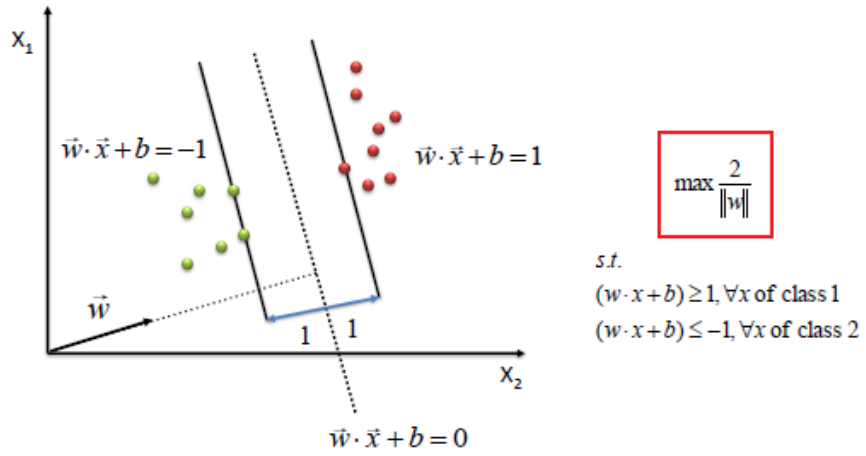


Figura 2.16: SVM: Minimización del error cuadrático

transformado (espacio de características), si los ejemplos no son linealmente separables en el espacio original. Como se verá más adelante, la búsqueda del hiperplano de separación en estos espacios transformados, normalmente de muy alta dimensión, se hará de forma implícita utilizando las denominadas funciones *kernel*.

Además, a la hora de definir el hiperplano, solo se consideran los ejemplos de entrenamiento de cada clase que caen justo en la frontera de dichos márgenes. Estos ejemplos reciben el nombre de vectores soporte. Desde un punto de vista práctico, el hiperplano separador de margen máximo ha demostrado tener una buena capacidad de generalización, evitando en gran medida el problema de sobreajuste a los ejemplos de entrenamiento.

En los modelos de clasificación lineal, en el más básico, el objetivo consiste en encontrar de entre las infinitas rectas separadoras, hiperplanos, aquella recta óptima que quede lo más equidistante posible de los vectores de soporte, consiguiendo que deje la máxima distancia posible a cada lado del hiperplano, lo que permita en el futuro poder catalogar de manera precisa y correcta nuevas muestras, y evitar el problema de sobreajustes a los ejemplos de entrenamiento. Así, el problema se reduce a encontrar el vector normal a esa recta.

En la figura 2.16 se observa que el hiperplano ideal sería $w \cdot x + b = 0$, donde

la incógnita es el vector normal, W. Los hiperplanos paralelos $w \cdot x + b = 1$ y $w \cdot x + b = -1$ son los que pasan más cerca de las clases. La solución eficiente por mínimos cuadrados sería:

$$argmin \frac{1}{2} \parallel W \parallel^2, w_1, w_2, \ldots, w_d \tag{2.18}$$

condicionado a:

$$y_i(w \cdot x_i + b) \geq 1 \quad \forall i = 1, \ldots, n \tag{2.19}$$

Cuando haya ruido en los datos –muestras que no se correspondan con la distribución lineal de los datos– provocará alteraciones en el hiperplano separador.

Existe un problema aún mayor cuando los datos están solapados, por lo que no son linealmente separables, con lo que la minimización del error cuadrático se convierte en irresoluble. Para ello nos apoyamos en la búsqueda de hiperplanos tolerantes y en el uso de *kernels*.

Un hiperplano tolerante es aquél que admite un cierto número de errores. Para su cálculo se introduce una variable de tolerancia C (permisividad ξ_i para x_i) a cada uno de los datos, lo que permitirá no tener en cuenta las muestras ruidosas o que difieran de una distribución normal de los datos.

La figura 2.17 muestra gráficamente el concepto de tolerancia, y se aclara que el objetivo es maximizar el valor $\frac{2}{\|W\|}$.

La solución para al problema de cálculo de hiperplanos tolerantes sería:

$$argmin \frac{1}{2} \parallel W \parallel^2 + C \sum_{i=1}^{n} \xi_i \tag{2.20}$$

condicionado a:

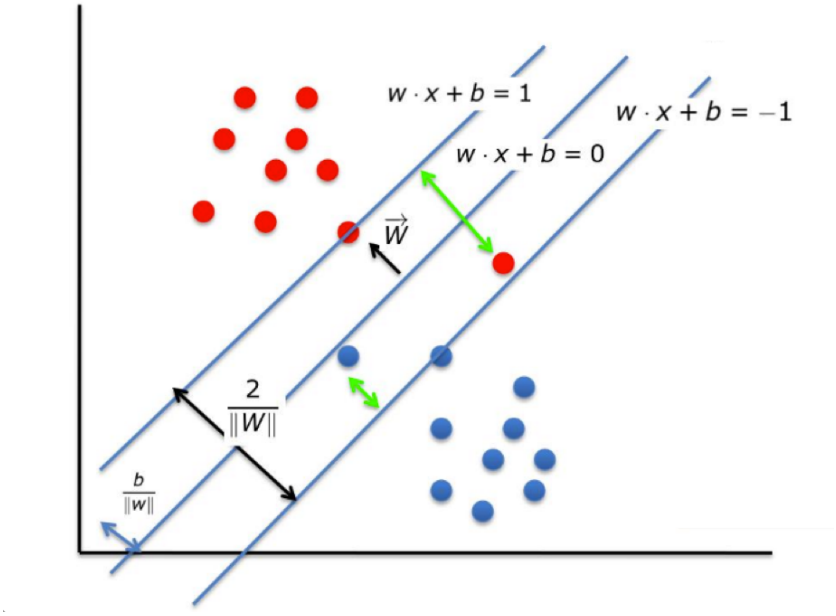


Figura 2.17: Tolerancia. Permisividad ξ_i para cada x_i [92]

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i, \forall i = 1, , , , n \tag{2.21}$$

Cuando los datos tienen una distribución curvilínea, para ellos se proyectan los datos a un espacio de dimensión superior (*kernel trick* [94]) en el que los datos son linealmente separables. Si conocemos la distribución de nuestros datos, podemos utilizar un *kernel* o función de proyección que se adecúe a nuestra distribución. Existen multitud de *kernels*, entre los más conocidos están los siguientes:

- Lineal.

$$K_l(x, x') =< x, x' > \tag{2.22}$$

- Polinómico de grado–p.

$$K_p(x, x') = [\Upsilon < x, x' > +\tau]^p, \Upsilon > 0 \qquad (2.23)$$

- Gaussiano RBF o *Radial Basic Function.*

$$K_g(x, x') = exp\left(-\frac{\|x - x'\|^2}{2\Upsilon^2}\right) \qquad (2.24)$$

- Sigmoidal.

$$K_s(x, x') = tanh(\Upsilon < x, x' > +\tau) \qquad (2.25)$$

- Hipertangente.

$$tanh(\Upsilon(x \cdot x') + \theta),\ \Upsilon > 0\ y\ \theta \in R \qquad (2.26)$$

Donde Υ, τ y p son los parámetros del *kernel.*
Se puede concluir que *SVM* [92]:

1. Es el más potente de los clasificadores en la optimización del error estructural.
2. Encuentra la frontera de decisión lineal óptima.
3. Es sensible al ruido en los puntos de soporte. Si tenemos ruido en las fronteras de decisión, se alterará el hiperplano separador.
4. Tiene limitaciones cuando trabaja con datos que no son linealmente separables, pero se puede trabajar utilizando *kernels* y tolerancia.
5. Intenta minimizar el error cuando los datos tienen cierto grado de solapamiento entre las clases.

Capítulo 3

Diseño de la investigación

3.1. Metodología

En este capítulo se describe la metodología utilizada para definir el sistema inteligente que detecte comportamientos anómalos que puedan clasificarse como *APT*.

Se ha utilizado una metodología de investigación mixta, con una vertiente cualitativa y otra cuantitativa. La parte cualitativa incluye estudios exploratorios de documentos, búsqueda y revisión bibliográfica, y toma de notas de campo para definir las *APTs* (Fase1). Para definir el concepto de *APT* se ha hecho una revisión de la literatura, a fin de detectar conceptos claves e ideas sobre los datos y los análisis. El proceso llevado a cabo es inductivo e interpretativo a partir de la información bibliográfica y del conocimiento de personas expertas en la materia. La otra metodología adoptada es de carácter cuantitativo, y a través de ella se va a desarrollar un sistema de aprendizaje automático. Se comparan diferentes métodos de aprendizaje basados en minería de datos, se estudian y analizan los sistemas de aprendizaje desarrollando experimentos que permitan comparar los distintos sistemas y seleccionar el que presente un mejor comportamiento.

La tabla 3.1 muestra los objetivos (descritos en la sección 1.4) asignados a cada tipo de investigación.

Tipo de Investigación	Objetivos de la investigación
Cualitativa	Ob1
Cuantitativa	Ob2, Ob3, Ob4, Ob5, Ob6, Ob7, Ob8, Ob9

Tabla 3.1: Tipos de investigación y objetivos asignados

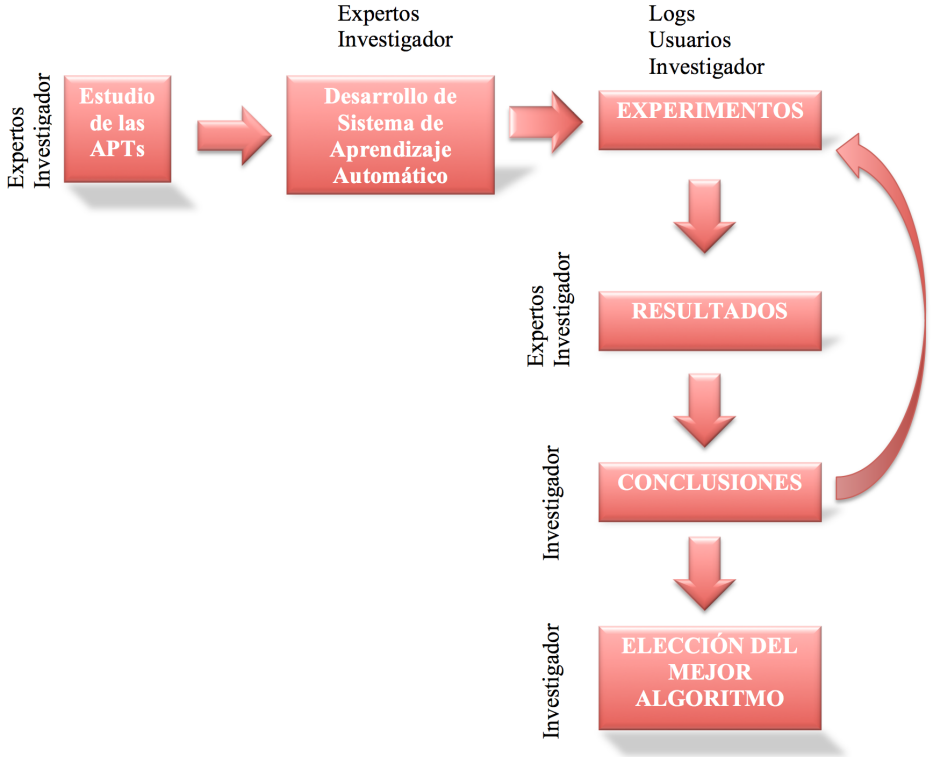


Figura 3.1: Esquema general y actores que intervienen

La figura 3.1 muestra de forma general el conjunto de etapas que intervienen en la investigación, así como los actores que forman parte en cada una de ellas.

3.2. Fases y tareas de la investigación

La tabla 3.2 muestra las distintas tareas que componen cada fase, quiénes han intervenido en su realización y las herramientas utilizadas para llevarlas a cabo.

Fase	**Tarea**	**Herramienta**
F1 Estudio de las *APTs*	F1T1. Revisión de la literatura. (Expertos, investigador)	Informes
	F1T2. Consulta a expertos. (Expertos, investigador)	Informes
	F1T3. Redefinición del concepto de *APT*. (Investigador)	Informes
F2 Desarrollo de un *SAA*	F2T1. Obtención del tráfico de la red (*logs*) de la infraestructura *TIC*. (Expertos, investigador)	Informes, *raw logs*
	F2T2. Normalización de los registros de *logs*. (Expertos, investigador)	Informes, descriptor, *raw logs*, vector entrenamiento
	F2T3. Discretización de los registros de *logs*. (Expertos, investigador)	Informes, vector discretizado
	F2T4. Generación de valores (tráfico normal y sintético). (Investigador)	Programas *Python*, vector discretizado
	F2T5. Uso de *SAA* para adquirir capacidades predictivas. (Investigador)	Instancias, muestras, *Knime*
	F2T6. Selección de las variables de evaluación. (Expertos, investigador)	Informes, *Knime*
	F2T7. Propuesta de modelo del *SAA*. (Investigador)	Informes, *Knime*

Sigue en la página siguiente.

Fase	**Tarea**	**Herramienta**
F3 Experimentos	F3T1. Definición de las actividades a realizar. (Investigador)	Informes, *Knime*
	F3T2. Descripción de la infraestructura objeto de estudio. (Investigador)	Informes
	F3T3. Selección de los conjuntos de entrenamiento. (Investigador)	Instancias, programa *Python*
	F3T4. Ejecución de experimentos. (Investigador)	Informes, *Knime*, hoja de cálculo
F4 Resultados	F4T1. Obtención de resultados. (Expertos, investigador)	Informes, *Knime*
	F4T2. Elección de los mejores métodos predictivos. (Investigador)	Informes, *Knime*
	F4T3. Pruebas reales. (Investigador)	*Knime*, Instancias, muestras
	F4T4. Selección del mejor SAA. (Expertos, investigador)	Informes
F5 Conclusiones	F5T1. Extracción de conclusiones. (Investigador)	Informes

Tabla 3.2: Tareas realizadas por etapas, y herramientas utilizadas

Para llevar a cabo las tareas presentadas en la tabla 3.2 se han realizado una serie de trabajos que se exponen a continuación.

F1. Estudio de las *APTs*

F1T1 Se han realizado estudios de informes de expertos, revisión de bases de datos bibliográficas, así como entrevistas relacionadas con:

1. Medidas de seguridad por *hardware* y *software* que poseen las infraestructuras *TIC* para hacer frente a los ataques.
2. Técnicas de robo de información.
3. Evolución de la definición de *APT*.

F1T2 Se han realizado estudios de informes de expertos, revisión de bases de datos bibliográficas, así como entrevistas relacionadas con:

1. Estudio de las fases por las que pasa una *APT* cuando realiza un ataque a una infraestructura *TIC*.
2. Estudio de los escenarios que se pueden dar en el ataque por *APT*.
3. Estudio de los casos reales más relevantes de *APTs*.
4. Sistemas existentes para la detección de *APTs*.

F1T3 Se ha analizado toda la documentación e informes obtenidos en las tareas anteriores para emitir una definición de *APT*.

1. Definición unificada de *APT*.

F2. Desarrollo de un *SAA*

Para llevar a cabo las tareas de la fase 2 se han desarrollado trabajos basados en el análisis de *raw logs*, a fin de desarrollar descriptores que permitan obtener conjuntos de vectores de entrenamiento, normalizados y discretizados, adecuados para ser utilizados por herramientas de minería de datos y de aprendizaje automático.

La figura 3.2 muestra un esquema general de la fase 2.

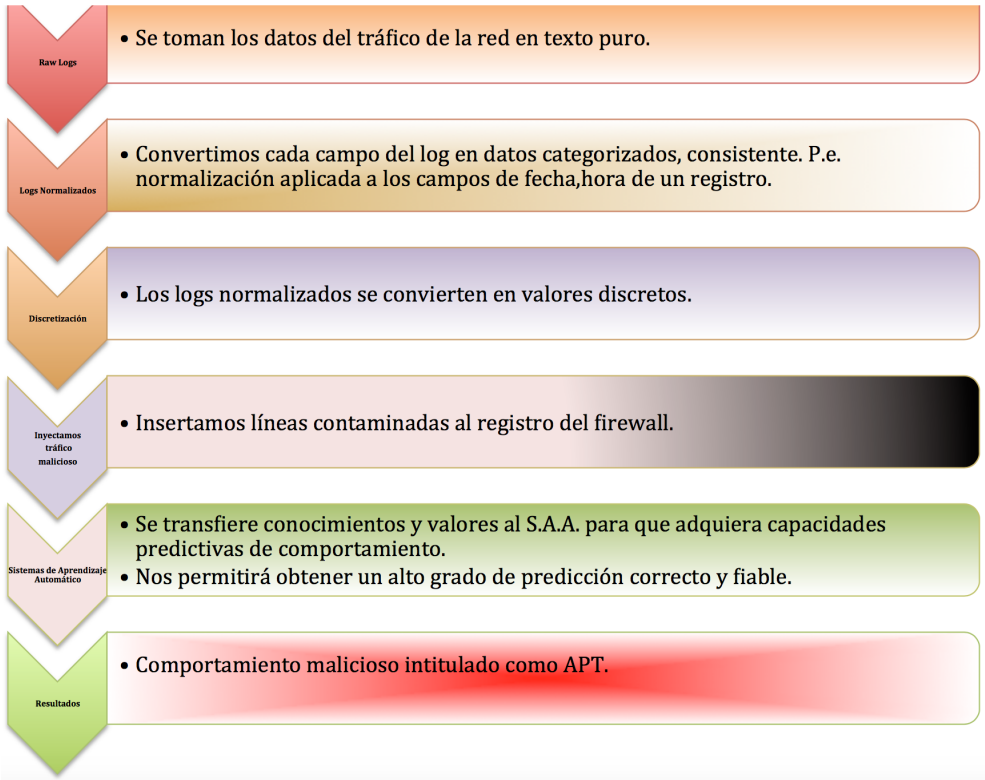


Figura 3.2: Esquema general de la fase 2 (desarrollo SAA)

F2T1 Trabajos realizados en la tarea 1:

1. Analizar la información que aportan los registros de *logs* en general.
2. Analizar la información que aportan los *logs* de los *firewalls* de una infraestructura *TIC* en producción.
3. Establecer los niveles de alerta que utiliza el *firewall* sobre los que actuará una *APT*.
4. Obtener la información del tráfico de entrada/salida en *raw* de la infraestructura *TIC*.
5. Catalogar el tráfico de la infraestructura en una ventana de tiempo que se toma como referencia.

La infraestructura objeto de estudio tiene las siguientes características:

- Volumen: La información almacenada en los logs llega a ser de 5.5 GB por día, lo que constituye unos 7.5 millones de líneas de tráfico diarias. Por otro lado, tal como se aprecia en la Figura 3.3, la mayor parte del tráfico es externo. Además hay que tener en cuenta que en la infraestructura existen otros elementos de seguridad, destinados a proteger la red, y que generan una enorme cantidad de *logs*, del orden de PB.
- Velocidad: Por hora se generan ficheros de *logs* con un tamaño de unos 240 MB, que contienen más de 300.000 líneas por cada fichero.
- Variedad: Los *logs* tienen información de diferente naturaleza (eventos, seguridad, tráfico). Nosotros vamos a considerar solamente los correspondientes al tráfico de entrada/salida del *firewall,* ya que el resto de *logs* producen una alerta ellos mismos sin necesidad de realizar tratamiento alguno.

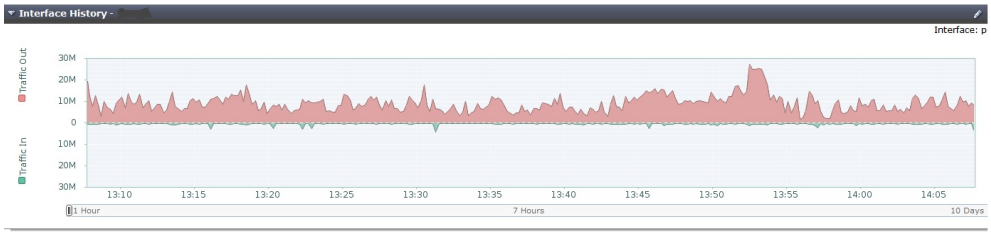


Figura 3.3: Ejemplo de distribución del tráfico en una infraestructura durante una ventana temporal

F2T2 Trabajos realizados en la tarea 2:

1. Definir los parámetros indicadores de comportamiento de los equipos que conforman la infraestructura *TIC*.
2. Diseñar un descriptor inicial a partir del *raw log* y cuyos atributos estén normalizados.
3. Utilizar computación paralela con MPI para normalizar la información.

F2T3 Trabajos realizados en la tarea 3:

1. Diseñar instancias con los atributos predictivos seleccionados.
2. Diseñar reglas de cuantificación para la creación de los atributos predictivos.
3. Diseñar reglas de correlación para asignar el comportamiento de grupo.
4. Crear instancias con los atributos predictivos discretizados y añadiendo el atributo de estado de grupo.

F2T4 Trabajos realizados en la tarea 4:

1. Insertar tráfico sintético.
2. Crear muestras con distintas proporciones de tráfico.

F2T5 Trabajos realizados en la tarea 5:

1. Estudio de técnicas de aprendizaje supervisadas (árboles de decisión y clasificadores bayesianos), por agrupación no supervisada (*k-means*), redes neuronales artificiales y máquinas de soporte vectorial (SVM).

2. Resolver las tareas especificadas en la tabla 2.3.

   Las tareas de aprendizaje automático son las siguientes:

   - AP1. Determinar la función objetivo. El objetivo es la detección de comportamientos anómalos que puedan ser debidos a un ataque por *APT*.
   - AP2. Elegir los atributos predictivos. Tras el preprocesamiento se eligen los elementos que van a formar los vectores de entrenamiento.
   - AP3. Codificar los valores de los atributos. Se les aplica una codificación numérica y de rangos de valores.
   - AP4. Recopilar un número razonable de ejemplos de entrenamiento. Se adquieren del tráfico real tanto inocuo como malicioso. Se mezclan con tráfico sintético.
   - AP5. Seleccionar las instancias de entrenamiento y aplicar los distintos *SAA*.
   - AP6. Seleccionar la técnica más adecuada.
   - AP1. Obtendremos la información y datos necesarios de la fuente de entrada de datos que serán analizados. *Firewall*.
   - AP2. Tratamiento para la extracción de los datos de las entradas. La extracción puede ser incremental (una a una) o en un solo paso. Se opta por la extracción en un solo paso.
   - AP3. Cómo son generadas las fuentes de entrada de datos. Las entradas pueden ser generadas por algún mecanismo externo o por algún otro sistema. Se op-

ta por ambos. Los datos de entrada son relevantes y parecidos entre sí para que puedan formar parte de la descripción asociada al objetivo.

- AP4. Preprocesamiento de las entradas. Las entradas se preprocesan para eliminar información superflua y, así, obtener elementos consistentes.
- AP5. Ruido en entradas defectuosas o con incertidumbre. Se analizan las entradas defectuosas o con incertidumbre para su eliminación.
- AP6. Representación y normalización de los elementos seleccionados de los datos de entradas que aportarán valor para la extracción del conocimiento. La representación puede ser simbólica, numérica, estructurada, etc. Se opta por representación numérica. En los datos sintéticos y en los datos de tráfico real se seleccionan los atributos relevantes de las entradas asociadas al objetivo con ejemplos representativos del concepto y el fin buscado. La representación de los resultados se puede llevar a cabo con formalismos por reglas, árboles, grafos, etc. Se ha optado por utilizar tablas y gráficas.

3. Transferir las instancias al sistema de aprendizaje automático.

F2T6 Se etiqueta el tráfico anómalo de la red que pudiera corresponder a un ataque por *APT* como comportamiento *rojo*. El resto del tráfico se marca como comportamiento *verde*. Los trabajos correspondientes a la tarea 6 son los siguientes:

1. Selección de las variables de evaluación para la elección del modelo de aprendizaje automático. Las variables son las siguientes:
   a) Exactitud del modelo creado con la técnica seleccionada.

b) Mejora versus el modelo trivial.

c) Sensibilidad en comportamientos rojos.

d) Precisión de la resistencia.

e) Mejora de la resistencia versus modelo trivial.

f) Sensibilidad de la resistencia en el comportamiento rojo.

F2T7 Trabajo realizado en la tarea 7:

1. Propuesta de modelo de un sistema inteligente que cree modelos predictivos de comportamiento para detectar actuaciones anómalas que puedan ser intituladas como *APTs*.

F3. Experimentos

Para llevar a cabo las tareas de la fase 3 se han realizado una serie de trabajos apoyados en estudios de informes de expertos, programas *Python*, *Knime*, hojas de cálculo, indagación de las bases de datos bibliográficas, entrevistas relacionadas con los diseños de minería de datos y sistemas de aprendizaje automático.

F3T1 Trabajo realizado en la tarea 1:

1. La figura 3.4 presenta un modelado *UML (Unified Modeling Language)* de las actividades a realizar para el diseño del sistema de aprendizaje automático, que abarca desde la captura del tráfico de la infraestructura *TIC* hasta la alerta por posible *APT*.

F3T2 Trabajos realizados en la tarea 2:

1. La infraestructura objeto de este estudio es una infraestructura en producción, real, de tamaño y dispersión geográfica considerable, tiene repartidos sus recursos en múltiples edificios interconectados por un anillo de fibra óptica, centralizando comunicaciones y recursos en un nodo central. La infraestructura *TIC* obedece a una infraestructura tipo con la siguiente descripción (Figura 3.5):

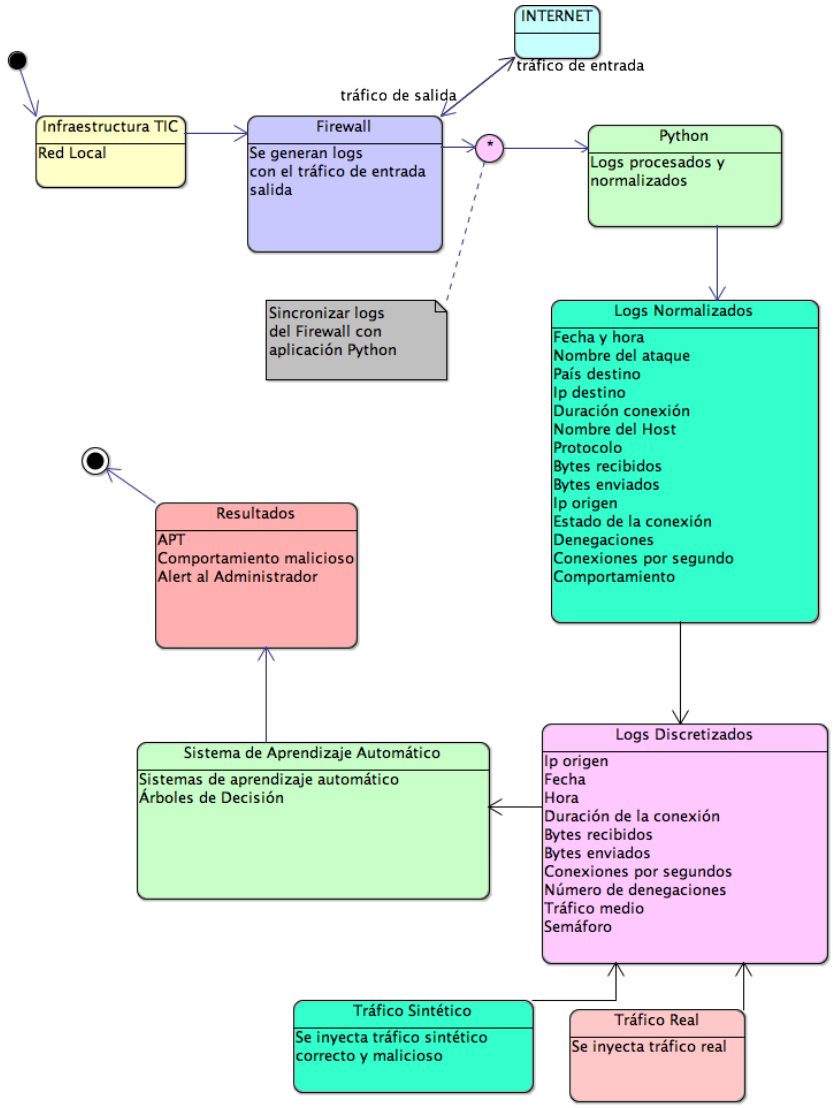


Figura 3.4: Modelo UML del sistema de aprendizaje automático propuesto

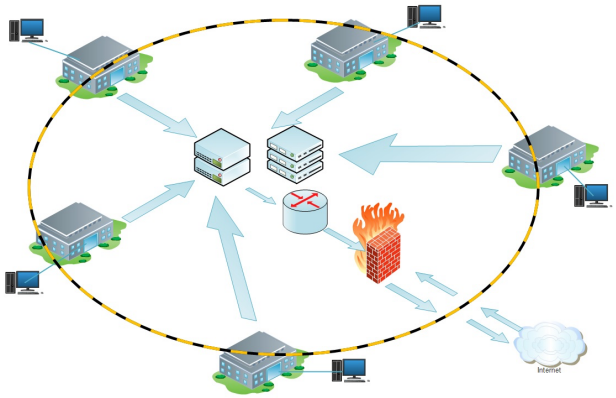


Figura 3.5: Diseño de la Infraestructura

a) Múltiples edificios conectados por un anillo de fibra óptica.

b) Más de quinientos ordenadores conectados en red.

c) Varios dominios de *broadcast*, entre ellas la *DMZ*.

d) Accesos *VPN* por *IPSec* y *SSL*.

e) Ordenadores portátiles, *tablets* y teléfonos móviles conectados a la red.

f) Varios *data centers*, uno principal y el resto secundarios.

g) *Clusters* de servidores físicos y virtualizados.

h) *Network security appliance firewall* en alta disponibilidad.

i) Cabinas de discos con tecnologías *SAN* y *NAS*.

j) *Core* de gestión de la electrónica de red.

k) Gestor de base de datos objeto-relacional.

l) Conexión a Internet.

F3T3 Trabajos realizados en la tarea 3:

1. Crear las particiones de las muestras para entrenamiento y test de acuerdo con el modelado predictivo descrito en la sección 2.4.

2. Crear distintas muestras compuestas por instancias de registros reales y artificiales con diferentes proporciones de tráfico inocuo y malicioso.
3. Diseñar los diagrama de flujo de cada uno de los algoritmos de aprendizaje.
4. Configurar los *learners* y *predictors* de cada uno de los algoritmos de aprendizaje.
5. Entrenar los algoritmos de aprendizaje con las combinaciones de instancias.

F3T4 Trabajos realizados en la tarea 4:

1. Crear los modelos predictivos (*PMML, Predictive Model Markup Languaje*) para cada uno de los algoritmos de aprendizaje.
2. Calcular las matrices de confusión [95, 96] para evaluar el rendimiento de cada uno de los sistemas de apredizaje utilizados.
3. Obtener la resistencia del modelo (diferencia entre la precisión de la muestra modelo y la precisión del modelo trivial, siendo la precisión del modelo trivial el valor más alto del porcentaje de muestra rojo/verde tratado) creado con cada muestra a través de pruebas de validación con ejemplos de diferente naturaleza.
4. Obtener la mejora de cada modelo con respecto al trivial.
5. Obtener la sensibilidad de las pruebas en el comportamiento anómalo.
6. Seleccionar las variables de los resultados de análisis con cada muestra.
7. Cuantificar en cuartiles los resultados de las variables.

F4. Resultados

Para llevar a cabo las tareas de la fase 4, se han realizado una serie de trabajos apoyados en el estudio de informes de expertos, en el uso de la herramienta *Knime* [100] y de hojas de cálculo, en la revisión de bases de datos bibliográficas, y en entrevistas relacionadas con el uso de minería de datos y de sistemas de aprendizaje automático.

F4T1 Trabajos realizados en la tarea 1:

1. Analizar la precisión de cada técnica utilizada, así como los errores obtenidos.
2. Seleccionar la mejor técnica de aprendizaje automático a partir de los cálculos de precisión y de errores.

F4T2 Trabajos realizados en la tarea 2:

1. Obtener la mejora sobre el modelo trivial de la técnica elegida.
2. Analizar los resultados de la matriz de confusión sobre todas las muestras utilizando el mejor modelo.
3. Examinar los resultados de las pruebas de validación sobre el conjunto de muestras para comprobar la resistencia del mejor modelo.
4. Asignar un cuartil a cada muestra en cada variable y calcular sus medias.
5. Presentación gráfica de los resultados.
6. Elegir las mejores muestras del algoritmo de aprendizaje automático en base a los resultados obtenidos.

F4T3 Trabajos realizados en la tarea 3:

1. Someter a los modelos predictivos a pruebas con datos reales.
2. Obtener las matrices de confusión y las estadísticas para ambos modelos.

F4T4 Trabajos realizados en la tarea 4:

1. Analizar los resultados obtenidos en las pruebas con datos reales.
2. Seleccionar la mejor muestra para el algoritmo de aprendizaje automático.

F5. Conclusiones.

F5T1 El investigador emitirá las conclusiones finales y las líneas futuras de investigación.

3.3. Materiales e Instrumentos

Los materiales e instrumentos utilizados para llevar a cabo las fases y tareas descritas en la sección 3.2 son los siguientes:

1. Informes y entrevistas con expertos e investigadores interdisciplinares pertenecientes al *CSIRT-CV* (Centro de Seguridad *TIC* de la Comunidad Valenciana), al Instituto Nacional de Ciberseguridad (*INCIBE*), a empresas del sector de la seguridad y de las comunicaciones, y al Instituto de Ciencias Aplicadas a la Ciberseguridad (RIASC) de la Universidad de León.

2. *Raw logs*. Se obtienen del *firewall* y contienen la información sin procesar del tráfico de entrada/salida y salida/entrada de la infraestructura *TIC*. Se encuentran descritos en el apéndice B.

3. Descriptor del tráfico de red. Consta de 15 parámetros que permiten identificar el comportamiento de un equipo que pertenece a la infraestructura *TIC*.

   El conocimiento experto de administradores de redes permite identificar comportamientos anómalos en el tráfico de red que supera el filtro de

un *firewall* (niveles 5 y 6 de la tabla 2.1) y que podría estar relacionado con ataques por *APT*.

Se propone utilizar un descriptor del tráfico de red de un equipo, identificado por su dirección *IP*, que consta de 15 parámetros:

- Tiempo total que una máquina de la infraestructura permanece conectada tanto con tráfico de entrada como de salida (*in-out*), (*out-in*).
- Máquinas de la propia infraestructura a las que se conecta.
- Días y horas de establecimiento de las conexiones.
- Días en los que la máquina debería de estar conectada y en los que no.
- Horas en las que la máquina debería de conectarse y en las que no.
- Existencia de tráfico de entrada/salida fuera del horario permitido.
- Existencia de intentos de establecer varias conexiones en un segundo a uno o varios servidores externos.
- Conjunto de servidores externos a los que se conecta habitualmente.
- Existencia de solicitudes de resolución *DNS* que no son resueltas.
- Existencia de conexiones a direcciones externas de dudosa reputación.
- Existencia de un número alto de denegaciones de servicio *(deny)* por parte del *firewall*.
- Existencia de varias solicitudes de resoluciones *DNS* en un segundo.
- Existencia de tráfico de salida muy alto.
- Valor medio del tráfico.
- El tráfico medio es mayor que un cierto umbral pre-establecido.

Estos parámetros se obtienen procesando la información disponible en el volcado *raw* del *firewall.*

4. Vector de entrenamiento. Compuesto por los atributos predictivos seleccionados del *raw log* que van a permitir aplicar el descriptor del tráfico de red.

   La tabla 3.3 muestra los atributos predictivos que conforman el vector de entrenamiento –como aparecen reflejados en el *raw log*– y su denominación correspondiente.

Atributo	**Denominación**
Date	Fecha de la conexión
Time	Hora de la conexión
Attackname	Nombre del ataque, si existe
Dstcountry	País destino, al que se realiza la conexión
Dstip	Dirección IP a la que se conecta
Duration	Duración de la conexión
Hostname	Identificación del ordenador que realiza la conexión
Proto	Protocolo utilizado para la conexión
Rcvdbyte	Bytes recibidos en la conexión
Sentbyte	Bytes enviados en la conexión
Srcip	Dirección IP del equipo que realiza la conexión
Status	Estado de la conexión que se ha realizado o se ha intentado realizar

Tabla 3.3: Diseño del vector de entrenamiento

5. *MPI* (*Message Passing Interface*) [97].

   Se ha desarrollado un aplicación paralela para normalizar los *logs* acorde al vector de entrenamiento, basada en paso de mensajes, utilizando librerías de *MPI* para *Python.*

   El algoritmo de normalización de *logs* carece prácticamente de código secuencial, el volumen de datos a procesar es considerable y el número de operaciones a realizar sobre cada dato es alto. Todo esto consigue que

se incida positivamente en el *SpeedUp* –ganancia resultante de emplear un algoritmo paralelo en lugar de uno secuencial.

Utilizamos el modelo maestro/esclavo (1 maestro - "x" esclavos) sobre una arquitectura *MIMD (Multiple Instruction, Multiple Data)*. La cantidad de esclavos vendrá determinada por la potencia de la máquina así como los valores de ganancia que se puedan calcular para ello. El modelo se puede paralelizar en un *cluster* o un supercomputador que soporte las librerías *MPI* [98, 99].

La aplicación se lanza en paralelo sobre N procesadores. Los procesos no avanzan sincronizados, y la sincronización no es implícita, tiene que ser explícita y controlada. Todos los procesos tienen un espacio de memoria completamente separado y tanto el intercambio de información entre los procesos como la sincronización se hacen mediante paso de mensajes. Para las comunicaciones entre los procesos no ha sido necesario crear agrupaciones ni formar comunicadores entre ellos.

En función del comportamiento que vaya teniendo el algoritmo de aprendizaje o el procesamiento de los registros de *logs*, la aplicación está diseñada para que de una manera fácil se pueda gestionar el vector de entrenamiento, incorporando nuevos campos del *raw log,* o eliminando aquellos que no sean necesarios.

Además, las pruebas a realizar para comprobar la viabilidad de las modificaciones son fáciles de llevar a cabo, gracias a la portabilidad de *MPI* así como el código interpretado que ofrece *Python*.

Para la gran cantidad de *logs* de tráfico real que ha generado la infraestructura objeto de estudio, se han utilizado en las pruebas 7.445.736 líneas de registros de *logs* generados en un día, realizándose la paralelización de la aplicación sobre 5 procesadores. El tiempo de ejecución fue de 12 minutos, lo que arroja un ratio de 30 segundos por cada hora de *log* generado por el *firewall*.

Las figuras 3.6 y 3.7 muestran las gráficas del crecimiento del tamaño de los ficheros que almacenan los registros de *log* en función del tiempo y la variación del tiempo de ejecución del programa paralelo en función

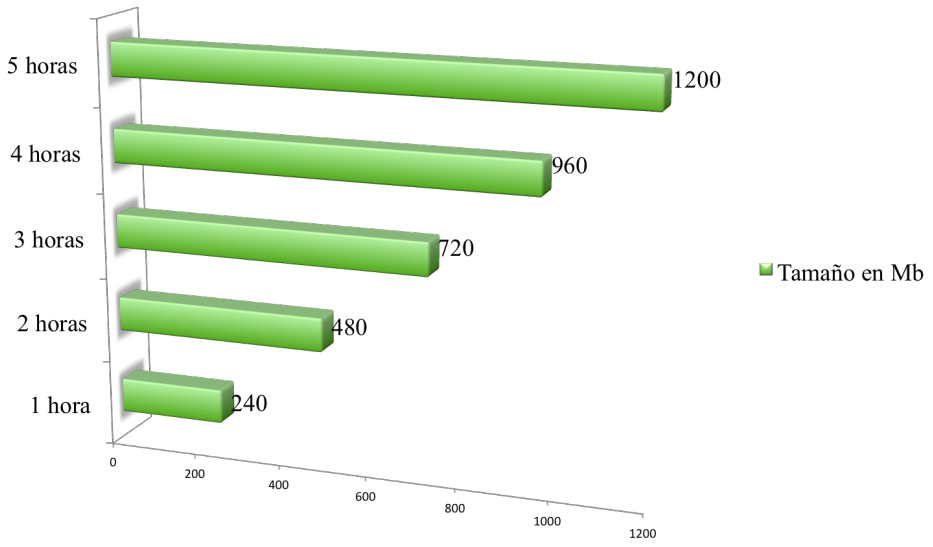


Figura 3.6: Representación del tamaño de los ficheros de *logs* en función del tiempo

del número de procesadores utilizados.

6. Vector normalizado. Es la conversión de todos o de parte de los campos que componen el vector de entrenamiento a un modelo sólido y equilibrado, con la información procesada y depurada.

   Para ello se han realizado las siguientes conversiones:

   a) Las fechas son convertidas a una cadena de caracteres que utiliza cuatro dígitos para el año, dos dígitos para el mes y otros dos para el día (AAAAMMDD).

   b) Las horas son convertidas a una cadena de caracteres que utiliza dos dígitos para la hora, dos dígitos para los minutos y dos dígitos para los segundos (HHMMSS).

   c) Los valores numéricos son enteros.

   d) Las direcciones *IP* se transforman en cadenas de 12 dígitos. Como una dirección *IP* consta de 4 bloques de tres dígitos cada uno, se ajustará cada uno de los bloques con ceros a la izquierda.

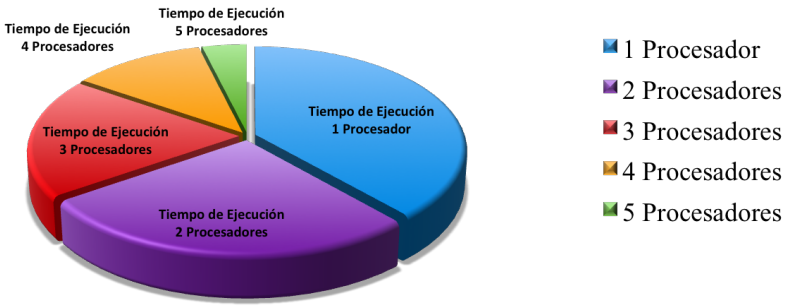


Figura 3.7: Representación del tiempo de ejecución en función del número de procesadores

e) Si algún atributo del vector carece de valor, se le asignará el valor cero.

f) Los sitios web y el estado de la conexión se codifican con caracteres en minúsculas.

7. Vector discretizado. Conjunto de atributos a los cuales se aplican reglas de cuantificación y de correlación.

   Partiendo de los *logs* depurados y normalizados, se han utilizado 9 atributos predictivos relacionados con la *IP* origen. Ocho de ellos son estados que podemos obtener de los registros normalizados así como de cálculos realizados sobre otros campos. Se incorpora un valor de estado de grupo al que denominaremos semáforo, el cual podrá conmutar entre dos valores: rojo (comportamiento peligroso o anómalo) y verde (comportamiento inocuo).

   La figura 3.8 muestra el diseño y los atributos que componen el vector discretizado.

   Cada elemento i del vector se representa mediante una variable x_i, de

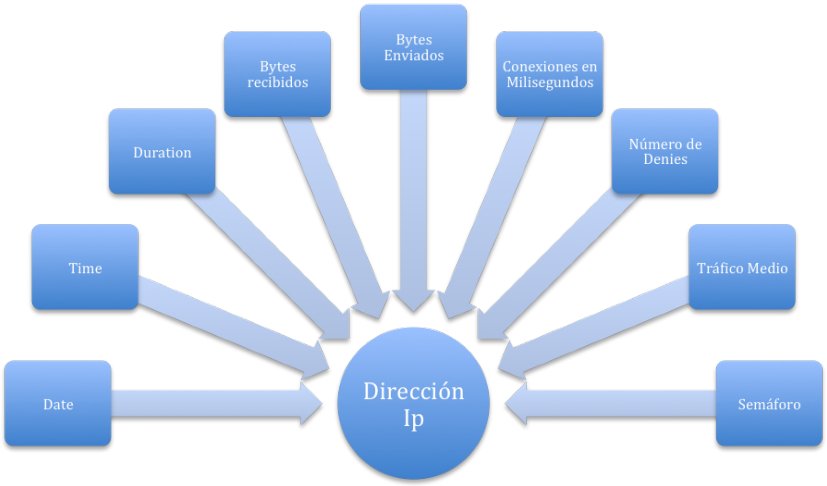


Figura 3.8: Vector discretizado

tal manera que x_1 = fecha, x_2 = hora, x_3 = duración, x_4 = bytes recibidos, x_5 = bytes enviados, x_6 = conexiones, x_7 = denegaciones, x_8 = trafico medio, x_9 = semáforo.

En función del conocimiento experto establecido por diversos autores y especialistas, tomamos en consideración las siguientes premisas para la creación de las reglas de cuantificación y de correlación que se exponen a continuación.

Reglas de cuantificación. Diseñadas para reducir la variabilidad y la complejidad de los datos antes de ser utilizados en los algoritmos de aprendizaje.

La discretización de los atributos predictivos implica distinguir, por una parte, los días laborables y los no laborables, y, por otro, las mañanas, las tardes y las noches (Ecuaciones 3.1 y 3.2).

$$\text{Date} = \begin{cases} 1 & \text{Día Laboral} \\ 2 & \text{Fin de Semana} \end{cases} \tag{3.1}$$

$$\text{Time} = \begin{cases} 1 & \text{Mañana} \\ 2 & \text{Tarde} \\ 3 & \text{Noche} \end{cases} \tag{3.2}$$

El resto de variables se codifican usando sus medias aritméticas (Ecuación 3.3).

$$X = \begin{cases} 1 & \text{if } x \leq \overline{x} \\ 2 & \text{if } \overline{x} < x \leq 1{,}20\overline{x} \\ 3 & \text{if } x > 1{,}20\overline{x} \end{cases} \tag{3.3}$$

Donde x y X son los valores de la misma variable antes y después de la cuantificación, respectivamente.

Reglas de correlación. Diseñadas para asignar al vector un estado de grupo que indicará si el comportamiento es inocuo o malicioso.

La ecuación 3.4 muestra las siguientes reglas de correlación, donde $|w|_a$ representa el número de veces que aparece el valor a dentro de la cadena w:

$$x_9 = \begin{cases} r(rojo) & \text{Si} \begin{cases} |x|_3 > 2 \\ (|x|_2 > 2) \wedge (|x|_3 > 0) \wedge (x_1 = 2) \\ x_2 = 3 \\ (x_1 > 1) \wedge [(x_3 = 3) \vee (x_5 = 3) \\ \vee (x_6 = 3) \vee (x_7 = 3)] \\ (x_2 > 1) \wedge [(x_5 = 3) \vee (x_6 = 3) \vee (x_7 = 3)] \end{cases} \\ v(verde) & \text{En otro caso} \end{cases} \tag{3.4}$$

El resultado de aplicar las reglas de correlación corresponde al comportamiento del grupo, que es el que determina si el tráfico es inocuo (verde) o malicioso (rojo).

8. Instancias. Vectores discretizados con los valores correspondientes después de aplicar las reglas de cuantificación y correlación, y que serán los que emplearemos para crear las muestras que se utilizarán en los experimentos.

9. Programas *Python*.

Con el objetivo de recopilar y crear un número razonable de ejemplos de entrenamiento, mezclando tráfico real y sintético tanto malicioso como inocuo, para aplicarlos sobre distintos Sistemas de Aprendizaje Automático, se han diseñado dos aplicaciones para inyectar tráfico sintético de manera bien individual, bien masiva. También se ha diseñado una aplicación para el cálculo del algoritmo *K-means*. A continuación se describen estas aplicaciones.

Inyección de ataques individuales. El objetivo es dotar de un entorno interactivo desde el cual se puedan dar valores independientes a los atributos predictivos para diseñar los vectores discretizados que necesitemos con un criterio previamente establecido por el investigador. La aplicación establece un mecanismo de control de excepciones respecto a los valores que se solicitan en la entrada de datos.

La aplicación pedirá la cantidad de líneas que se quieren insertar y troceará el fichero en varias partes en función de la cantidad de registros y de las líneas a insertar con el objeto de darle aleatoriedad al contenido del vector. Desde la línea de comandos se indican los ficheros de entrada y de salida. El de salida será el fichero donde se almacenarán los registros del fichero de entrada más los registros sintéticos generados.

La figura 3.9 muestra el pseudocódigo, utilizando el paradigma de programación estructurada, que permitirá inyectar tráfico sintético de una manera individual.

Inyección de ataques masivos. Su objetivo es inyectar de manera masiva registros de ataques, sin intervención del usuario, para mezclar tráfico inocuo con tráfico malicioso, obteniendo por cada vector tantos vectores como atributos predictivos contenga, incrementándose el valor de cada atributo predictivo en una cantidad porcentual con respecto a la media. Con estos valores modificados se genera una cantidad de registros ingente, que permitirá simular ataques a partir de un fichero de entrada en el que todo el tráfico es inocuo.

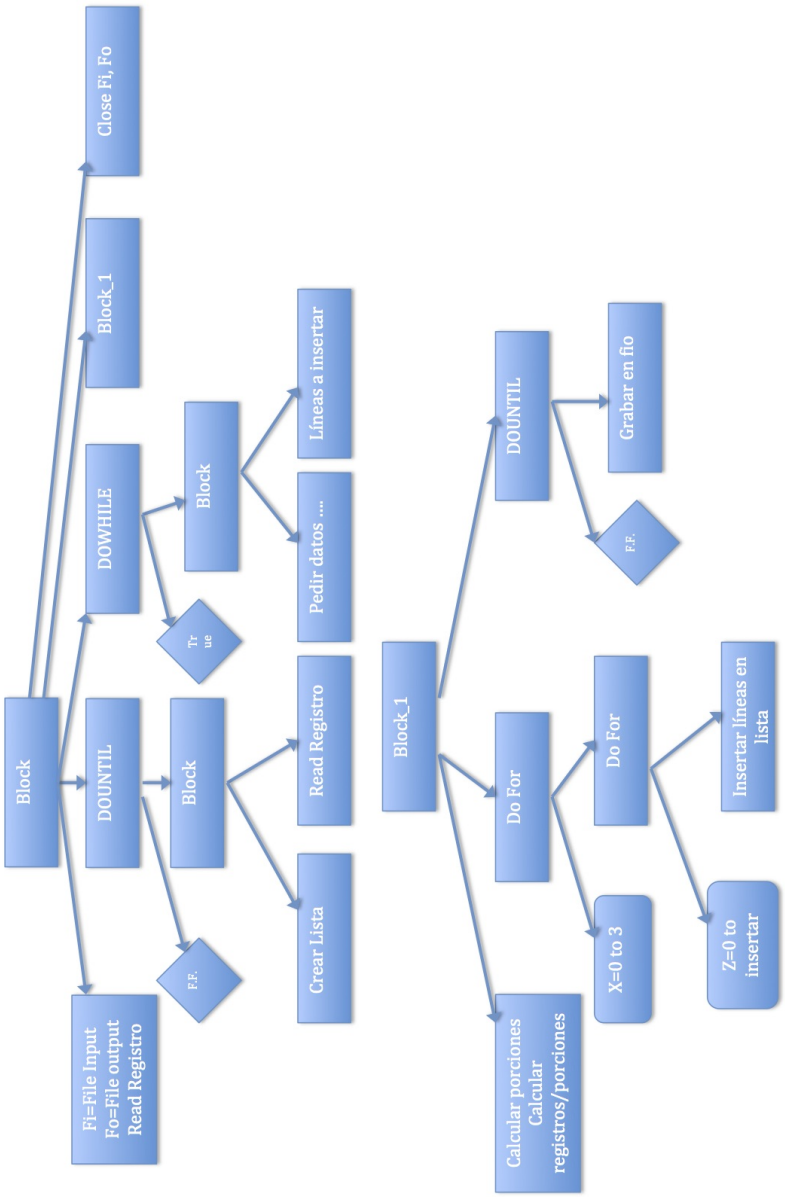


Figura 3.9: Pseudocódigo para inyectar ataques individuales sintéticos

La figura 3.10 muestra el pseudocódigo, bajo el paradigma de programación estructurada, que permitirá inyectar tráfico sintético masivo.

Cálculo de *k-means*. Para comprobar los resultados a través del algoritmo de aprendizaje automático *k-means* se ha desarrollado un programa en el que se toman los registros de *log* normalizados y se les aplica un algoritmo *hash* de conversión numérica. A continuación se pasan los datos, el número de *clusters* y el número máximo de iteraciones a la función *k_ means*.

La figura 3.11 muestra el pseudocódigo secuencial que permite aplicar el algoritmo *k-means*.

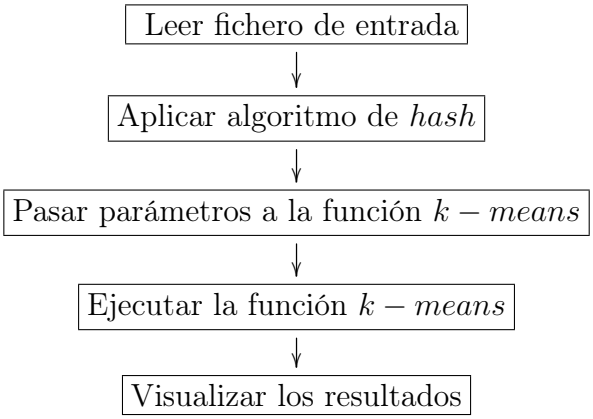


Figura 3.11: Pseudocódigo del algoritmo *k-means*

10. *Knime* versión 3.3.1 es un entorno para el desarrollo y la ejecución de técnicas de minería de datos. Fue desarrollado originalmente en el departamento de Bioinformática y Minería de Datos de la Universidad de Constanza, en Alemania, bajo la supervisión del profesor *Michael Berthold* [100].

11. Hoja de Cálculo. Se ha utilizado *Microsoft Excel* para realizar cálculos y para generar gráficas de los resultados obtenidos.

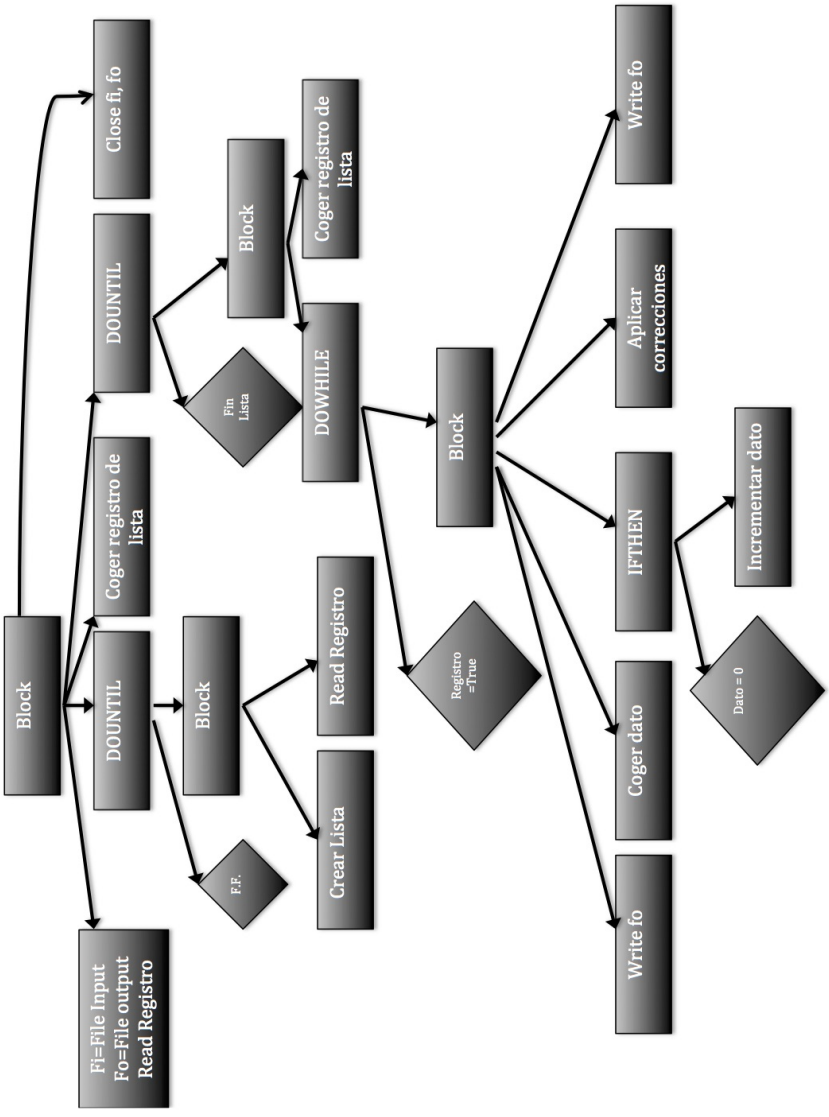


Figura 3.10: Pseudocódigo para inyectar ataques masivos sintéticos

"itime=1455521331","date=2016-02-
15","time=08:31:46","devid= ›xxxxxx","vd=root","type=traffic","su
btype=forward","app=""HTTP.BROWSER_Chrome""","","appcat=""Web.
Others""","appid=34039","applist=""Aplicaciones
Bloqueadas""","attack=""HTTP.BROWSER_Chrome""","","","devname=F
","","dstcountry=""Spain""","dstintf=""port16""","dstip=xxx.xxx.x
xx.xxx","dstport=80","duration=120","hostname=""xxxxxxxxxxxxxxxx""
","level=notice","logid=13","","","","","policyid=63","proto=6","rcvdbyte=
528","rcvdpkt=5","","","sentbyte=949","sentpkt=6","service=xxxxxxxxxx
Usuarios","sessionid=20918574","","srccountry=""Reserved""","srcintf=""
port10""","srcip=xxx.xxx.xxx.xxx","","","srcport=xxxxx","status=close","t
randisp=snat","transip=xxx.xxx.xxx","transport=xxxxx","","","utmaction=
passthrough","utmevent=app-ctrl"\

Figura 3.12: Ejemplo de una entrada de tráfico *log* en formato *raw*

3.4. Muestras

La figura 3.12 presenta una muestra de tráfico de *raw log* del *firewall* que posee la infraestructura *TIC*. En dicha muestra se ha sobrescrito información relevante para garantizar la seguridad y protección de los datos. La descripción de los campos del *raw log* se encuentra en el apéndice B. Por otro lado, la tabla 3.4 presenta ejemplos de los vectores de entrenamiento descritos en la sección anterior.

Las figuras 3.13, 3.14 y 3.15 presentan varios tipos de vectores normalizados con las reglas de normalización aplicadas. En dichas figuras se pueden ver tres tipos de *raw log*, así como los vectores normalizados que se generan a partir de cada uno de ellos. Se ha sobrescrito la información relevante para garantizar la seguridad y protección de los datos.

2016-02-15,08:34:46,0,Spain,185.43.182.65,10,nets.com,6,528,949,192.168.1.10,close
2016-02-15,09:01:46,0,Spain,185.43.182.83,120,www.h.es,6,472,1373,192.168.2.11,close
2016-02-15,09:31:46,0,United States,216.58.210.130,142,pagealesy,6,3853,8449,192.168.3.13,close
2016-02-15,10:40:46,0,Spain,185.43.182.65,120,nets.voc,6,528,904,192.168.4.14,close
2016-02-15,10:41:46,0,United States,13.107.4.50,0,nets.voc,6,0,0,192.168.5.15,deny
2016-02-15,23:31:46,0,United States,13.107.4.50,0,nets.voc,6,0,0,192.168.1.153,deny
2016-02-15,00:51:46,0,Spain,185.43.182.65,0,nets.com,6,0,0,192.168.1.121,deny
2016-02-15,10:13:46,0,United States,216.58.210.130,12,pagealesy,6,3,8,192.168.3.121,close

Tabla 3.4: Ejemplo de vectores de entrenamiento

Vector Normalizado 1

20160215,133321,0,Spain,185043182065,120,nets.vocento.com,6,528,949,192168026006,close

Raw log 1

itime=1455521331,"date=2016-02-15","time=13:33:21","devid=xxxxxx-xxx-xxx-x","vd=root",
"type=traffic","subtype=forward","app=""HTTP.BROWSER_Chrome""","",
"appcat=""Web.Others""","appid=34039","applist=""AplicacionesBloqueadas""",
"attack=""HTTP.BROWSER_Chrome""","","","devname=xxxxxx","","dstcountry=""Spain""",
"dstintf=""port16""","dstip=185.43.182.65","dstport=80","duration=120",
"hostname=""nets.vocento.com""","level=notice","logid=13","","","","","policyid=63",
"proto=6","rcvdbyte=528","rcvdpkt=5","","","sentbyte=949","sentpkt=6","service=xxxxxxxx",
"sessionid=20918574","","srccountry=""Reserved""","srcintf=""portxx""","srcip=192.168.26.6"
,"","","srcport=43815","status=close","trandisp=snat","transip=xxx.xx.xx.xx","transport=43815"
,"","","utmaction=passthrough","utmevent=app-ctrl"

Figura 3.13: Ejemplo 1 de vector normalizado a partir del tráfico *raw* del *log*

Vector Normalizado 2

20160215,083146,0,United States, 207046194014,0,0,6,0,0,192168039011,deny

Raw log 2

itime=1455521331,"date=2016-02-5","time=08:31:46","devid=xxxxxx-xxx-xxx-x","vd=root",
"type=traffic","subtype=forward","","","","","","","craction=131072","crscore=2432696350",
"devname=xxxxxx","devtype=""WindowsPC""","dstcountry=""UnitedStates""",
"dstintf=""port16""","dstip=207.46.194.14","dstport=80","duration=0","","level=notice",
"logid=13","","mastersrcmac=30:37:a6:ca7a:","osname=""Windows""","osversion=""xxxx"""
,"policyid=107","proto=6","rcvdbyte=0","","","","sentbyte=0","","service=HTTP",
"sessionid=20926751","","srccountry=""Reserved""","srcintf=""portxx""",
"srcip=192.168.39.11","srcmac=xx:xx:xx:xx:","srcname=KKKKKKKK","srcport=55087",
"status=deny","trandisp=noop","","","unauthuser=""ws-emimun05$""",
"unauthusersource=""kerberos""","","",""

Figura 3.14: Ejemplo 2 de vector normalizado a partir del tráfico *raw* del *log*

Vector Normalizado 3

20150318,133941,tcp_src_session,0,192168025006,0,0,6,0,0,083056029089,clear_session

Raw log 3

itime=1426682426,"date=2015-03-18","time=13:39:41","devid=xxxxxx-xxx-xxx-x","vd=root"
,"type=utm","subtype=*IPS*","attackid=100663402","attackname=""tcp_src_session""",
"count=1","craction=4096","crscore=3422552114","devname=XXXXX",
"dstip=192.168.25.6","dstport=80","eventtype=anomaly","identidx=N/A","level=alert",
"logid=18432","msg=""anomaly: tcp_src_session, 2 > threshold1""","policyid=N/A","proto=6"
,"ref=""http://www.fortinet.com/*IDS*/xxxxxx""","sensor=""DoSpolicy1"""
,"service=http","sessionid=0","severity=critical","srcintf=""portxx""","srcip=xxx.xxx.xxx.xxx",
"srcport=48451","status=clear_session"

Figura 3.15: Ejemplo 3 de vector normalizado a partir del tráfico *raw* del *log*

Atendiendo a la descripción de las instancias, se muestran algunos ejemplos de éstas.

I1 = (1,1,1,2,1,1,1,2,v)

I2 = (2,1,1,1,1,2,3,1,r)

I3 = (1,3,2,1,3,1,1,2,r)

I4 = (1,1,1,1,1,1,1,1,v)

La figura 3.16 presenta una muestra con varias instancias. Por cada una de ellas se puede apreciar el atributo de estado de grupo (v = verde o tráfico inocuo, r = rojo o tráfico malicioso). Estas muestras serán las que alimentarán al sistema de aprendizaje automático.

Row ID	Col0	Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8
Row0	2	2	1	2	2	3	3	2	r
Row1	2	1	3	3	1	3	1	2	r
Row2	2	3	2	3	3	2	2	3	r
Row3	2	2	2	2	1	2	2	1	v
Row4	1	3	1	3	2	1	2	3	r
Row5	1	3	3	1	3	3	3	2	r
Row6	2	2	2	2	2	3	3	2	r
Row7	1	1	1	3	3	2	1	3	v
Row8	1	2	3	2	2	2	2	2	v
Row9	1	3	1	2	2	1	2	2	r
Row10	1	1	2	3	3	1	1	3	r
Row11	1	3	1	3	1	1	2	2	r
Row12	2	3	1	2	3	1	2	3	r
Row13	1	2	2	3	1	1	3	2	r
Row14	1	3	1	2	3	3	2	3	r
Row15	1	2	3	2	3	3	3	3	r
Row16	1	1	1	1	3	1	3	2	v
Row17	2	2	2	3	1	2	1	2	v
Row18	2	1	1	1	3	3	2	2	r
Row19	2	3	3	1	1	2	2	1	r
Row20	2	2	2	3	3	2	2	3	r
Row21	2	2	1	1	2	2	2	2	v
Row22	1	1	3	1	2	3	1	2	v
Row23	1	3	1	3	3	3	2	3	r
Row24	1	2	1	1	1	2	3	1	r
Row25	1	3	1	3	3	3	3	3	r
Row26	2	1	3	3	2	2	1	3	r
Row27	2	2	3	1	2	2	2	2	r
Row28	2	3	3	3	1	3	3	2	r
Row29	2	2	1	3	3	3	2	3	r

Figura 3.16: Muestra de instancias con atributos predictivos discretizados enviados al sistema de aprendizaje automático

Capítulo 4

Experimentación

4.1. Diseño del experimento

Para el diseño del experimento se han utilizado las muestras descritas en la sección 3.4, llevándose a cabo las tareas 1, 2, 3 y 4 de la fase 3 mostradas en la tabla 3.2.

A continuación se muestran los diseños utilizados en los algoritmos de aprendizaje automático.

Algoritmo *k–means* Para la detección de comportamientos anómalos a través sistemas de aprendizaje no supervisado utilizamos el modelo de *k-means* por ser un algoritmo sencillo y eficiente a la vez. No obstante, no es el adecuado en algunos escenarios, como veremos a continuación.

Como premisa para el método *k-means*, los datos de entrada deben ser numéricos. Para cumplir con este requisito y poder utilizar esta técnica, los atributos predictivos son convertidos a numéricos a través de un programa *Python* que les aplica un algoritmo de *hash*.

A los datos se les aplica el algoritmo *k-means* utilizando el programa expuesto en la sección 3.3.

Algoritmo de *Naïve Bayes* Se utiliza este algoritmo porque es perfectamente adaptable cuando los datos proceden de una única fuente de entrada y no de múltiples sensores que aportan información al sistema de AA,

lo que provocaría redundancia, pues un error se podría detectar de varias formas [75, 82–84].

Las entradas de entrenamiento, validación y test deberán ser numéricas o un conjunto discreto de valores. El ruido en las entradas no le afecta mucho ya que los cálculos estadísticos permiten suavizarlo.

El método *bayesiano* aprende un conjunto de probabilidades a partir de un conjunto de entrenamiento, lo que le permitirá catalogar los atributos de entrada.

La figura 4.1 muestra el diseño del diagrama de flujo para llevar a cabo el experimento.

Algoritmo de árboles de decisión Es uno de los más utilizados en el campo del aprendizaje automático, no es complicado de implementar y es potente en sus predicciones [101].

Para que los ejemplos sean clasificados correctamente y tenga la capacidad de catalogar ejemplos futuros es necesario obtener un árbol de decisión simple y predictivo, apoyado en ejemplos representados en duplas atributo-valor cuyos valores asociados pueden ser discretos o numéricos. Se adapta perfectamente a las necesidades de este trabajo, ya que un árbol de decisión toma como entrada un objeto o situación representados por un conjunto de atributos y devuelve una decisión "verdadero/falso" o, en nuestro caso, "verde/rojo" (tráfico inocuo/tráfico malicioso).

El método permite tratar ejemplos con ruido en los que pueden presentarse ocasionalmente clasificaciones incorrectas de ejemplos, atributos erróneos o atributos que carecen de valor.

La fiabilidad del método es muy alta, ya que si los ejemplos no tienen ruido el método genera el árbol idóneo que engloba a todo el conjunto de ejemplos.

La figura 4.2 muestra el diseño del diagrama de flujo para llevar a cabo el experimento.

Algoritmo SVM Cuando se utiliza y entrena una máquina de soporte vectorial se necesitan requisitos de cálculo considerables, por lo que se recurre a una solución de mínimos cuadrados (QP).

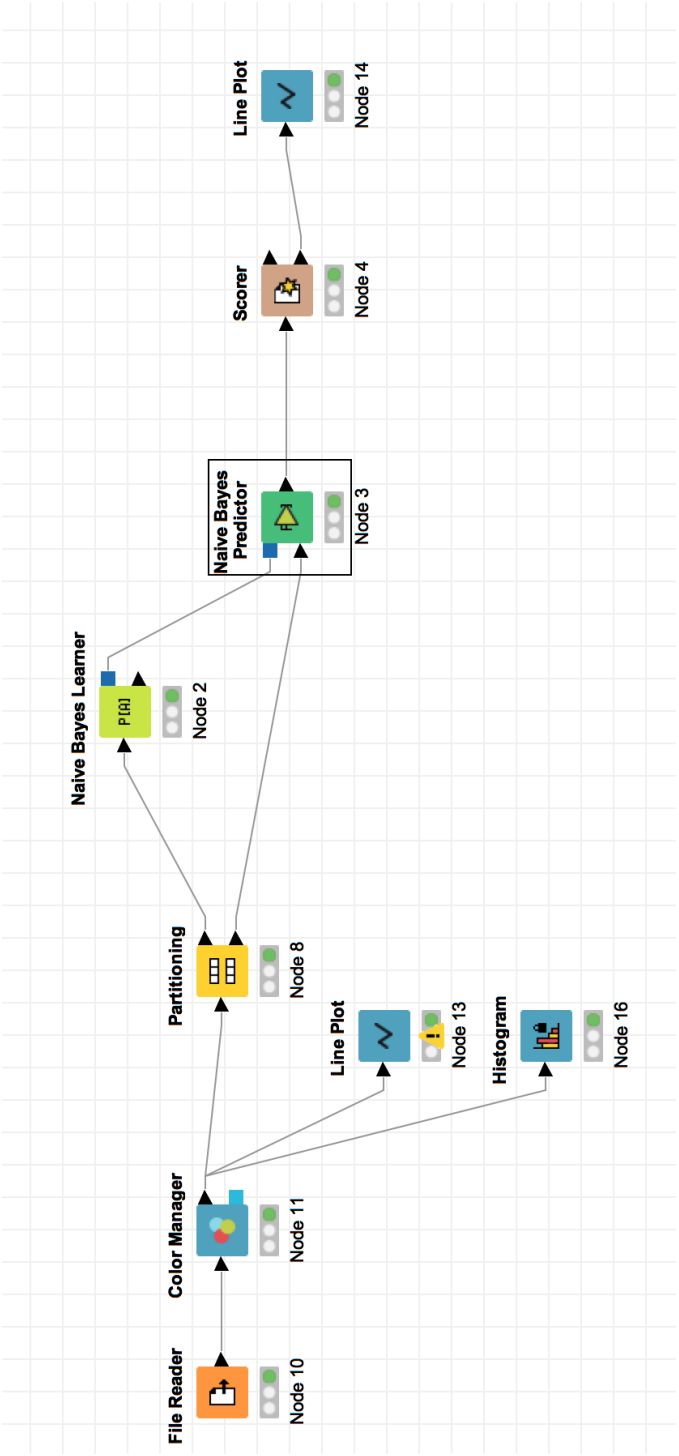


Figura 4.1: *Knime* diagrama de flujo. *Naïve Bayes*

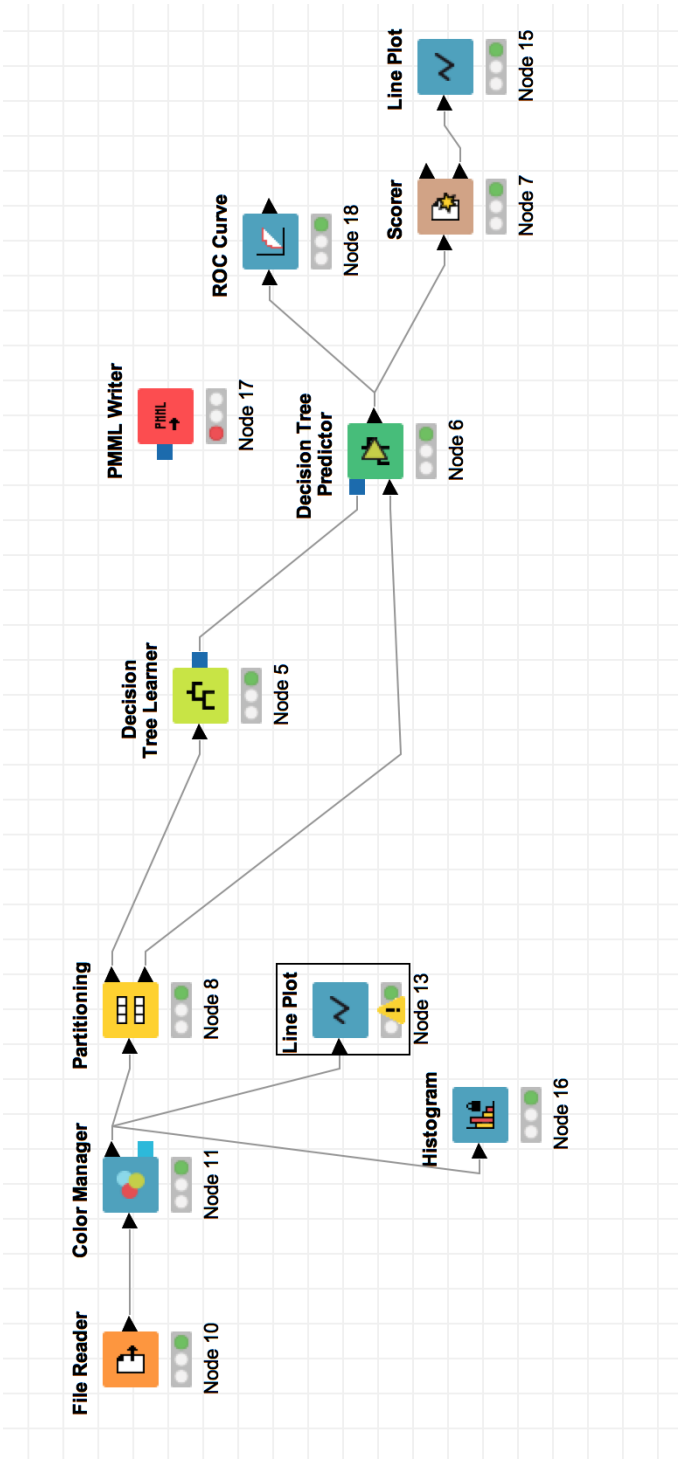


Figura 4.2: *Knime* diagrama de flujo. Árboles de decisión

Apoyados en la herramienta *Knime,* que permite *kernels* de tipo polinómico, hipertangente, Gaussiano RBF, utiliza *SMO* (*Sequential Minimal Optimization*) [102] con el objeto de dividir los problemas de programación de mínimos cuadrados (QP) en otros mucho más pequeños, más manejables, que rebajan el coste computacional de una manera sustancial. La resolución de las porciones de QP se resuelven analíticamente, lo que evita el uso de una optimización QP numérica que computacionalmente implica bucles internos que consumen recursos y tiempo de computación. La cantidad de memoria requerida para utilizar *SMO* es lineal en el tamaño del conjunto de entrenamiento, pero al dividir QP, permitirá el manejo de conjuntos de entrenamiento muy grandes. Para evitar el cálculo de grandes matrices, *SMO* realiza variaciones entre lineal y cuadrático en el tamaño del conjunto de entrenamiento para varios problemas de prueba, mientras que un gradiente de conjugado proyectado estándar (PCG) varía entre el lineal y el cúbico en el tamaño del conjunto de entrenamiento. El tiempo de computación de *SMO* está dominado por la evaluación de *SVM*, por lo tanto *SMO* es más rápido para *SVM* lineales y conjuntos de datos escasos [102].

El bloque de aprendizaje de *SVM* en *Knime* soporta múltiples problemas de clase (calculando el hiperplano entre cada clase y el resto), lo que provocará un aumento en el tiempo de ejecución.

El algoritmo de aprendizaje SVM utilizado en *Knime* aparece descrito en [102] y [103].

La figura 4.3 muestra el diseño del diagrama de flujo para llevar a cabo el experimento.

Redes neuronales El objetivo de utilizar este tipo de aprendizaje es generar una función lo más representativa posible de las muestras que se utilicen, por lo que cuanto más se aproxime la función a la muestra, más fiable será el método.

Las entradas y salidas están constituidas por atributos cuyos valores deberán ser numéricos. Asimismo las entradas a una célula pueden emanar de las salidas de otra. No obstante, el método se podría resumir como [75]:

1. Se introduce un nuevo ejemplo de entrada para el que se conoce su

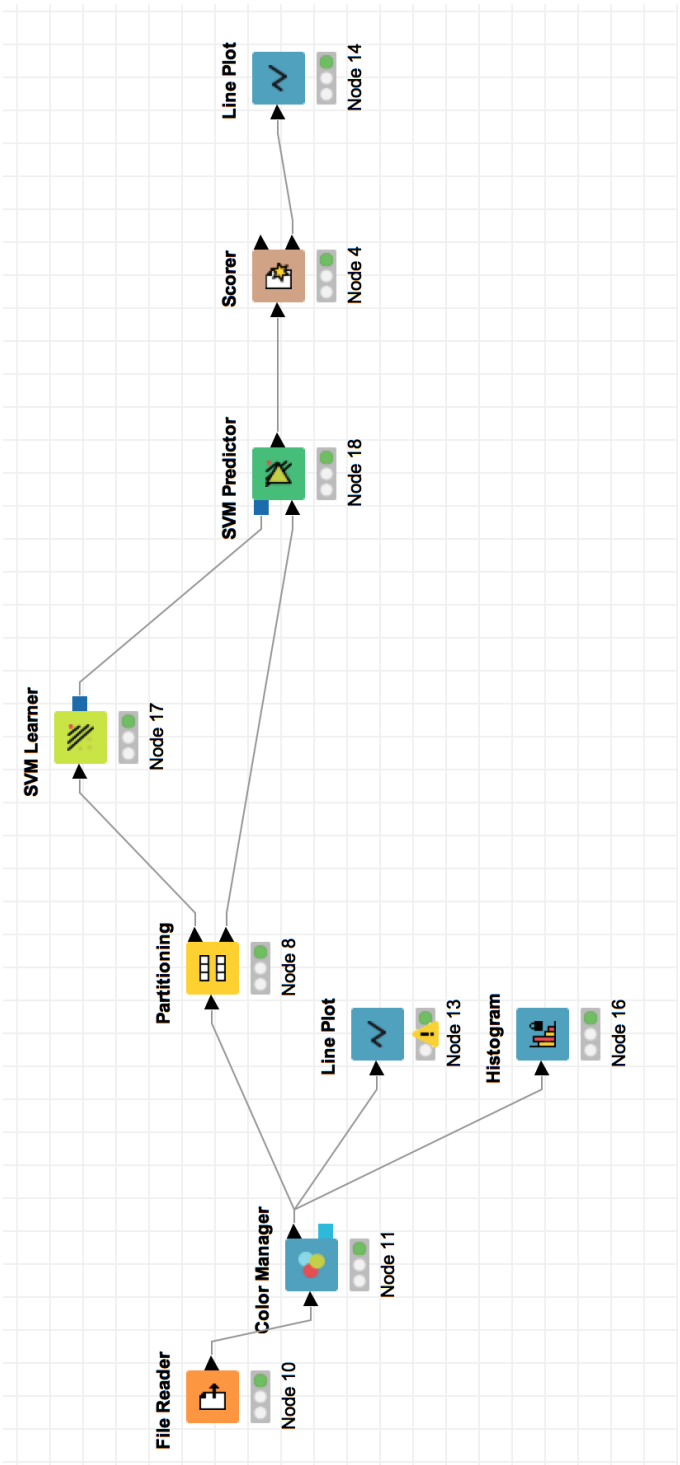


Figura 4.3: *Knime* diagrama de flujo. *SVM*

salida.

2. Para aquellas células cuyas entradas sean las propias entradas de la red, sus salidas se calculan como la suma de los valores de las salidas de las células con las que están conectadas ponderadas por sus respectivas conexiones.

3. Se calculan –como en el apartado anterior- las salidas de aquellas células de las que se dispongan de todos los valores de sus entradas.

4. Se itera 3 hasta que todas las células hayan producido su salida.

5. Se comparan las salidas generadas con las que supuestamente deberían de haberse producido.

6. Según los resultados obtenidos en el paso 5, se procede a reajustar todos los pesos.

7. Se repite todo el proceso mientras existan ejemplos de entrenamiento.

La figura 4.4 muestra el diseño del diagrama de flujo para llevar a cabo el experimento.

4.2. Ejemplos de entrenamiento

Los ataques por *APT* son muy poco frecuentes, por lo que la proporción de sus registros en los archivos de *log* es muy pequeña. Esto significa que nuestros conjuntos de datos implican distribuciones desequilibradas. Varios trabajos proponen el uso de datos sintéticos para mejorar los conjuntos de datos que sufren de distribuciones de clase desequilibrada, incluyendo métodos no heurísticos como submuestreo aleatorio o sobremuestreo [104], y aquellos que utilizan algún tipo de interpolación para sobremuestreo de los conjuntos de entrenamiento [105, 106].

En nuestro caso, los conjuntos de datos desequilibrados fueron mejorados por sobremuestreo aleatorio, de modo que los experimentos utilizaron archivos de registro de *logs* generados por el *firewall* de la infraestructura operativa y real en producción, combinando con registros sintéticos generados a

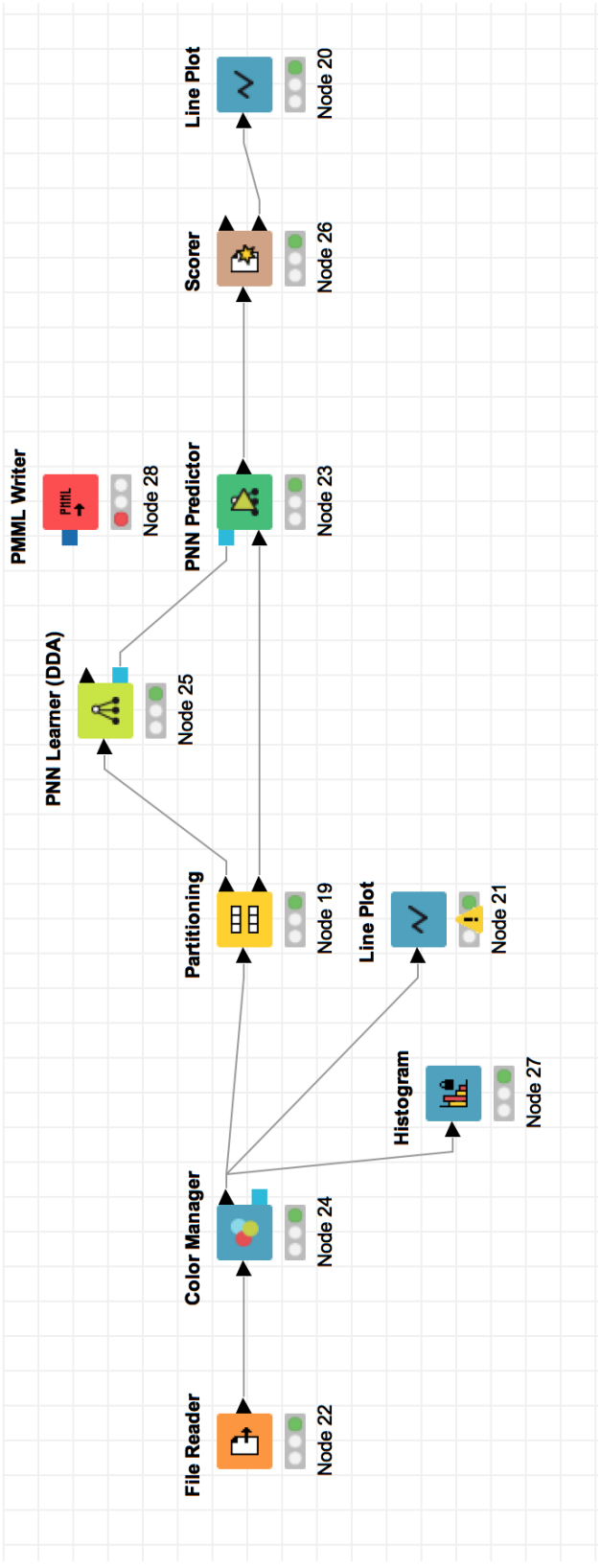


Figura 4.4: *Knime* diagrama de flujo. Red Neuronal

través del conocimiento experto. En particular, hemos creado y analizado 9 muestras (S_i, i = 1, ..., 9) con diferentes proporciones de comportamientos inocuos y anómalos, que permiten decidir cuál es la muestra más eficiente para nuestro sistema.

La muestra S fue elegida en una ventana temporal particular que permite clasificar todos los *logs* de S. En este caso, los registros fueron catalogados como comportamiento inocuo y, por esta razón, para completar y obtener un modelo más realista, fue necesario añadir registros sintéticos que representarán comportamientos anómalos (APT potencial). Obsérvese que la adición de registros sintéticos es una herramienta experimental habitual cuando hay ausencia de datos [107].

Las muestras se componen de un 20 % de datos reales aleatorios y un 80 % de datos sintéticos, con diferentes proporciones de comportamiento verde/rojo.

La tabla 4.1 presenta, por cada muestra S_i, las proporciones de elementos correspondientes a comportamiento inocuo (verde) y a comportamiento anómalo (rojo).

Muestra	**Comportamiento verde (%)**	**Comportamiento rojo (%)**
S_1	23.7	76.3
S_2	69.95	30.05
S_3	52.28	47.72
S_4	48.98	51.02
S_5	50	50
S_6	70.83	29.17
S_7	12	88
S_8	60.48	39.52
S_9	100	0

Tabla 4.1: Muestras de conjuntos de entrenamiento, S_i

Para alimentar al sistema de aprendizaje se han repartido las muestras en un conjunto de entrenamiento (65 %) y en un conjunto de test (35 %). La tabla 4.2 incluye la proporción de comportamientos verdes y rojos (CV y CR) de acuerdo con la división establecida para entrenamiento y test.

	Conjunto entrenamiento		Conjunto test	
Muestra	**CV (%)**	**CR (%)**	**CV (%)**	**CR (%)**
S_1	15.40	49.50	8.30	26.80
S_2	45.45	19.53	24.50	10.52
S_3	33.98	31.02	18.30	16.70
S_4	31.84	33.16	17.14	17.86
S_5	32.50	32.50	17.50	17.50
S_6	46.03	18.97	24.80	10.20
S_7	7.60	57.40	4.40	30.60
S_8	39.28	25.72	21.20	13.80
S_9	65.00	0	35.00	0

Tabla 4.2: Proporciones de comportamiento en entrenamiento y test de las muestras S_i

La tabla 4.3 muestra una visión de conjunto de las muestras utilizadas así como las particiones realizadas para cada ejemplo.

	Comportamiento		Conj. entrenamiento		Conjunto Test	
Muestra	**Verde (%)**	**Rojo (%)**	**CV (%)**	**CR (%)**	**CV (%)**	**CR (%)**
S_1	23.70	76.30	15.40	49.50	8.30	26.80
S_2	69.95	30.05	45.45	19.53	24.50	10.52
S_3	52.28	47.72	33.98	31.02	18.30	16.70
S_4	48.98	51.02	31.84	33.16	17.14	17.86
S_5	50	50	32.50	32.50	17.50	17.50
S_6	70.83	29.17	46.03	18.97	24.80	10.20
S_7	12	88	7.60	57.40	4.40	30.60
S_8	60.48	39.52	39.28	25.72	21.20	13.80
S_9	100	0	65.00	0	35.00	0

Tabla 4.3: Tabla general de conjuntos de entrenamiento y proporciones de comportamiento verde y rojo para entrenamiento y test de las muestras S_i

4.3. Realización del experimento

La figura 4.5 presenta todo el desarrollo del experimento. Abarca desde la obtención de los conjuntos de datos de la infraestructura *TIC* en producción, hasta la generación de una alerta por detección de *APT*. Con la obtención de los *raw logs*, su normalización, discretización e inserción de tráfico sintético,

permite crear las instancias con las cuales obtendremos los conjuntos de datos. A estos conjuntos de datos se les aplican los algoritmos de aprendizaje automático, para obtener los resultados. En vista de los resultados se elige el algoritmo de árboles de decisión, se calculan las variables, se dividen en cuartiles y se obtiene la mejor muestra. Se prueba el modelo con datos reales para comprobar que es válido para ser utilizado en la detección de *APTs* en la infraestructura real.

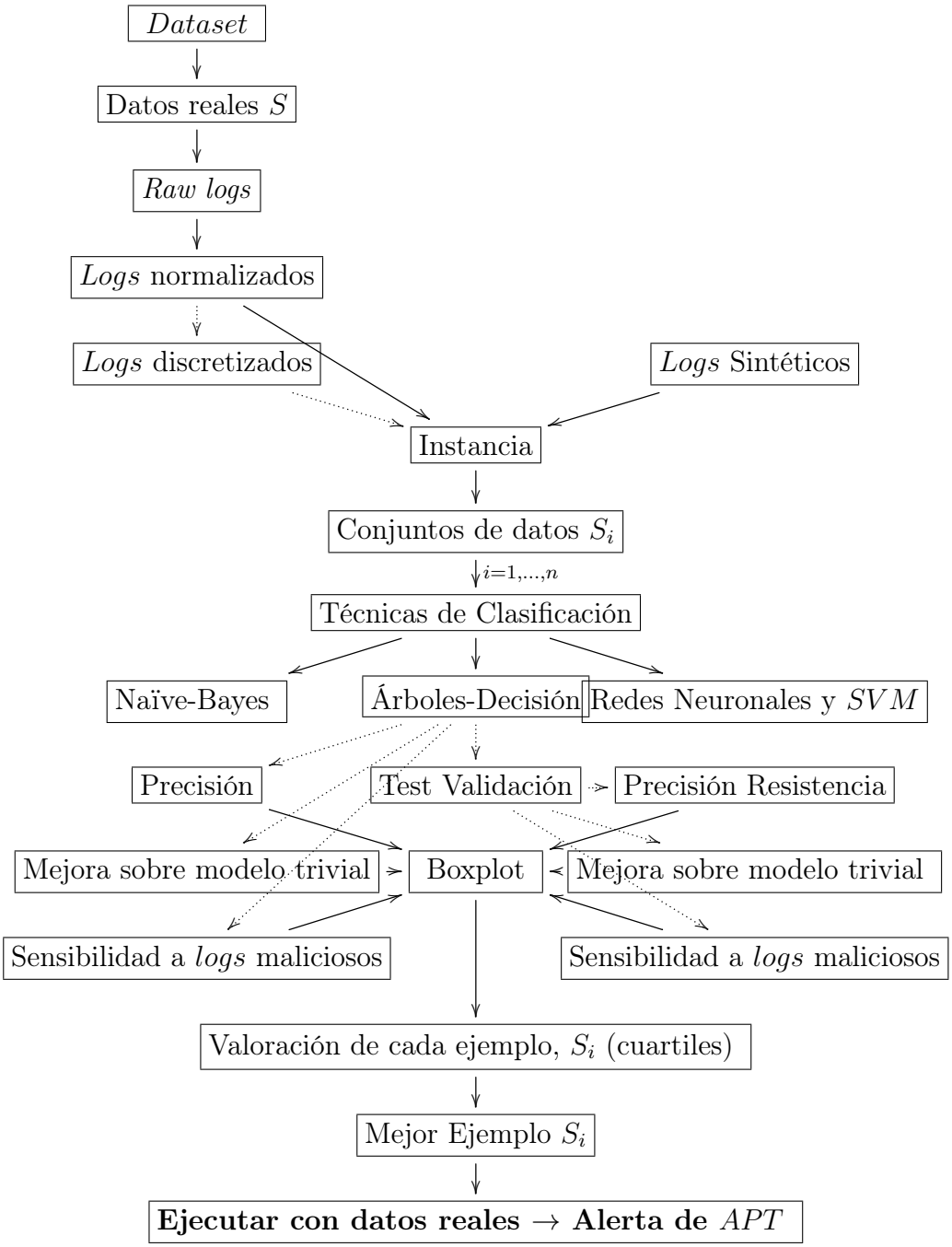


Figura 4.5: Diagrama descriptivo del desarrollo del experimento para detectar *APTs*

Capítulo 5

Resultados

La tabla 5.1 muestra la precisión (*accuracy*) y los errores obtenidos (*errors*) utilizando las técnicas de *Naïve Bayes* (NB), árbol de decisión (DT), redes neuronales artificiales (PNN) y máquinas de soporte vectorial (SVM). Los resultados obtenidos con el algoritmo *k-means* no han sido buenos, por lo que se descarta su uso en este trabajo.

	Naïve Bayes		**ID3-C4.5 DT**		**PNN**		**SVM**	
Ejemplo	**Pre (%)**	**Err (%)**	**Pre (%)**	**Err (%)**	**Pre (%)**	**Err (%)**	**Pre (%)**	**Err (%)**
S_1	90.26	9.74	95.96	4.04	91.12	8.88	87.42	12.58
S_2	85.47	14.53	99.24	0.76	88.20	11.80	89.40	10.60
S_3	89.03	10.97	97.49	2.51	86.39	13.61	86.21	13.79
S_4	89.15	10.85	97.44	2.56	88.69	11.31	86.19	13.81
S_5	86.76	13.24	97.95	2.05	89.87	10.13	86.65	13.35
S_6	85.52	14.48	97.99	2.01	83.18	16.82	87.83	12.17
S_7	89.20	10.80	95.10	4.90	88.57	11.43	87.14	12.86
S_8	85.25	14.75	98.82	1.18	91.64	8.36	91.23	8.77
S_9	83.29	16.71	100	0	100	0	100	0

Tabla 5.1: Precisión y errores usando *Naïve Bayes*, árboles de decisión, redes neuronales y máquinas de soporte vectorial

La tabla 5.2 muestra los porcentajes de mejora sobre el modelo trivial por cada S_i utilizando la técnica de árboles de decisión (E.V.: entrenamiento verde; E.R.: entrenamiento rojo; Pre: Precisión; PMT: precisión modelo trivial; MMT: mejora sobre el modelo trivial).

Muestra	E.V. (%)	E.R. (%)	Test V (%)	Test R (%)	Pre (%)	PMT (%)	MMT (%)
S_1	23.69	76.15	23.71	76.23	95.96	76.23	19.73
S_2	69.92	30.05	70	30.06	99.24	70	29.24
S_3	52.28	47.72	52.29	47.71	97.49	52.29	45.20
S_4	48.98	51.02	48.98	51.03	97.44	51.03	46.42
S_5	50	50	50	50	97.95	50	47.95
S_6	70.82	29.18	70.86	29.14	97.99	70.86	27.13
S_7	11.69	88.31	12.57	87.43	95.10	87.43	7.67
S_8	60.43	39.57	60.57	39.43	98.82	60.57	38.25
S_9	100	0	100	0	100	100	0

Tabla 5.2: Mejora sobre el modelo trivial con las muestras S_i

La tabla 5.3 muestra los resultados de la precisión, la mejora del modelo trivial (*improvement over trivial model*) y la sensibilidad (*sensitivity*) para cada muestra S_i utilizando la técnica de árboles de decisión (Pre: precisión; Err: error; MMT: Mejora sobre el Modelo Trivial; Sen: sensibilidad).

Muestra	**Pre (%)**	**Err (%)**	**MMT (%)**	**Sen (v)**	**Sen (r)**
S_1	95.96	4.05	19.73	1	0.95
S_2	99.24	0.76	29.24	0.98	1
S_3	97.49	2.51	45.20	0.95	1
S_4	97.44	2.56	46.42	1	0.95
S_5	97.95	2.05	47.95	0.96	1
S_6	97.99	2.02	27.13	0.98	0.98
S_7	95.10	4.90	7.67	0.94	1
S_8	98.82	1.18	38.25	0.95	1
S_9	100	0	0	1	-

Tabla 5.3: Resultados de la matriz de confusión sobre S_i con árboles de decisión

La tabla 5.4 presenta los resultados del análisis de la resistencia del modelo incluyendo las muestras que se utilizaron como pruebas de validación y los valores de la resistencia de la precisión (*accuracy/resistance*), la mejora de la resistencia del modelo sobre el modelo trivial y la sensibilidad para cada muestra (Val: validación; Pre: precisión; Res: resistencia; MMT: mejora sobre el modelo trivial; Sen: sensibilidad).

Muestra	**Val**	**Pre/Res (%)**	**MMT (%)**	**Sen (r)**
S_1	S_2	87.37	17.42	0.831
S_2	S_1	99.98	23.68	1
S_3	S_4	100	48.98	1
S_4	S_3	100	47.72	1
S_5	S_1	99.98	23.68	1
S_6	S_1	84.14	7.84	0.79
S_7	S_2	100	30.05	1
S_8	S_1	96.63	20.33	1

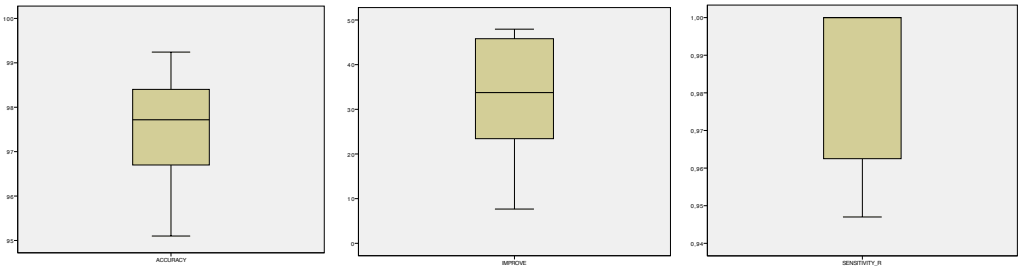
Tabla 5.4: Resultados de la prueba de validación sobre S_i

La tabla 5.5 presenta los valores de las variables para su distribución en cuartiles (Pre: precisión; MMT: mejora sobre el modelo trivial; Sen: sensibilidad; Res: resistencia).

Muestra	**Pre (%)**	**MMT (%)**	**Sen (r)**	**Pre/Res (%)**	**MMT (%)**	**Sen (r)**
S_1	95.96	19.73	0.95	87.37	17.42	0.831
S_2	99.24	29.24	1	99.98	23.68	1
S_3	97.49	45.20	1	100	48.98	1
S_4	97.44	46.42	0.95	100	47.72	1
S_5	97.95	47.95	1	99.98	23.68	1
S_6	97.99	27.13	0.98	84.14	7.84	0.79
S_7	95.10	7.67	1	100	30.05	1
S_8	98.82	38.25	1	96.63	20.33	1

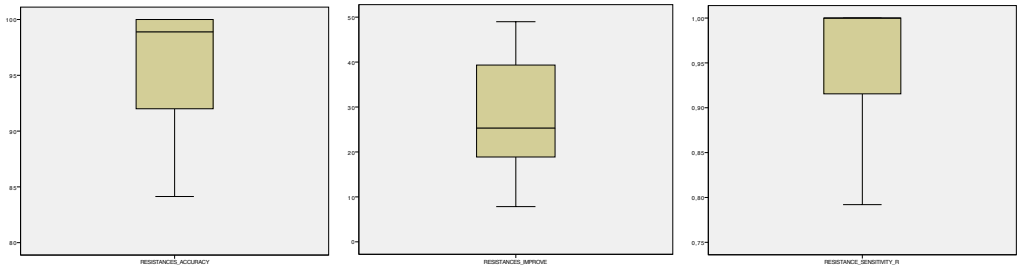
Tabla 5.5: Resultados de las pruebas sobre los conjuntos de entrenamiento S_i para asignaciones de cuartiles

Las Figuras 5.1 y 5.2 muestran los *boxplots* de las variables precisión, mejora sobre el modelo trivial, sensibilidad, precisión de la resistencia , mejora de la resistencia del modelo y sensibilidad de la resistencia con todas las muestras.



(a) Precisión (b) Mejora vs Modelo Trivial (c) Sensibilidad(R)

Figura 5.1: *Boxplots* de precisión, mejora sobre el modelo trivial y sensibilidad



(a) Precisión de la Resistencia (b) Mejora de la Resistencia (c) Sensibilidad de la Resistencia(R)

Figura 5.2: *Boxplots* de precisión de la resistencia, mejora de la resistencia sobre el modelo trivial y sensibilidad de la resistencia

La tabla 5.6 presenta la distribución en cuartiles de las variables así como los valores medios para cada muestra (Pre: precisión; M_vs_MT: mejora sobre el modelo trivial; Sen: sensibilidad; PR: precisión de la resistencia; MR: mejora de la resistencia; SR: sensibilidad de la resistencia). La figura 5.3 muestra un diagrama de barras con dichos valores cuantificados.

Muestra	Pre	M_vs_MT	Sen(r)	PR	MR	SR(r)	MEDIA
S_1	1	1	1	1	1	1	1
S_2	4	2	4	3	2	4	3.17
S_3	2	2	4	4	4	4	3.33
S_4	2	4	1	4	4	4	3.17
S_5	3	4	4	3	2	4	3.33
S_6	3	2	4	1	1	1	2.00
S_7	1	1	4	4	3	4	2.83
S_8	4	3	4	2	2	4	3.17

Tabla 5.6: Valores de cuartiles y medias sobre los conjuntos de entrenamiento S_i

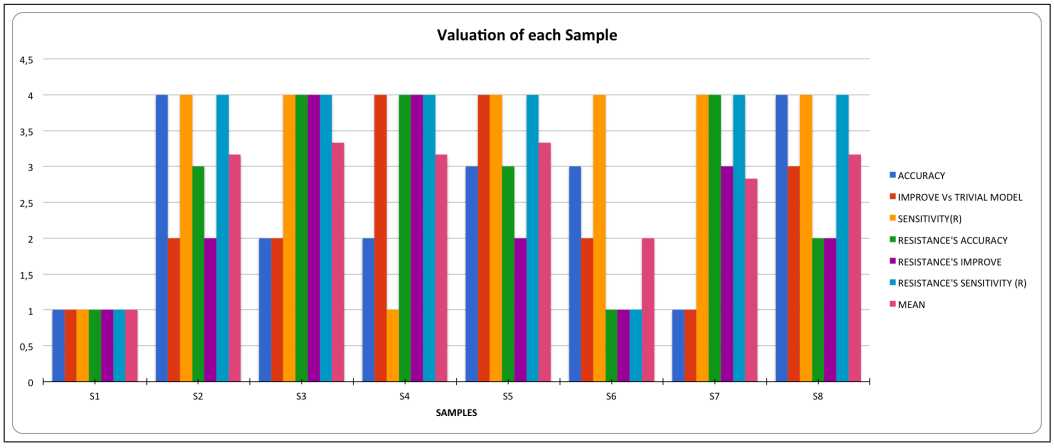


Figura 5.3: Diagrama de barras con los valores cuantificados para cada muestra

Capítulo 6

Análisis de los resultados

6.1. Análisis de los datos

En el capítulo 5 se han presentado los resultados de los experimentos. Para el análisis de los datos se han utilizado las variables definidas en la tarea F2T6 de la tabla 3.2.

Además, se ha tenido en cuenta la sensibilidad de las pruebas en el comportamiento rojo, porque en el contexto de la ciberseguridad interesa que en el proceso de clasificación de comportamientos anómalos existan la menor cantidad posible de casos con falsos negativos.

En todos los análisis se ha medido la mejora de nuestros modelos con respecto al modelo trivial. Este modelo trivial usaría el comportamiento más frecuente en la muestra para etiquetar cada actividad desconocida, es decir, si la mayor parte de la actividad en la muestra es un comportamiento verde, entonces el modelo trivial marcaría todos los elementos como comportamiento verde.

Se ha medido la resistencia del modelo creado con cada muestra S_i a través de pruebas de validación con ejemplos de diferente naturaleza. Así, para ese mismo modelo y para cada muestra, S_i, su resistencia se mide utilizando otra muestra, S_j con diferente proporción de comportamiento verde / rojo. Por ejemplo, S6 valida el modelo construido con S1, mientras que S2 es validado por S1 y así sucesivamente. En todo el análisis se ha medido la mejora de cada modelo con respecto al trivial.

Finalmente, los valores de las variables se cuantificarían considerando el cuartil al que pertenecen (4 para el cuartil superior, 1 para el inferior), y su promedio para cada muestra estimaría su aptitud, correspondiendo la mejor muestra al promedio más alto.

A continuación se indican las aportaciones de cada tabla para establecer los valores de las variables:

La tabla 5.1 busca conocer, de las técnicas utilizadas, cuál de ellas es la que obtiene mayor precisión y menor error con los juegos de muestras que contienen distintas proporciones de tráfico inocuo y malicioso, para poder elegir la técnica a utilizar.

En las matrices de confusión resultantes de las pruebas de análisis descritas en el capítulo 5 sobre cada S_i, se aprecia que el árbol de decisión *ID3–C4.5 (DT)* proporciona mayor precisión y menor error que *Naïve Bayes*, que la red neuronal artificial y que las máquinas de soporte vectorial.

Los mejores resultados se obtienen utilizando la técnica de árboles de decisión, por lo que se opta por esta técnica y se procede a calcular la mejora sobre el modelo trivial.

La tabla 5.2 aporta información acerca de la mejora sobre el modelo trivial después de aplicar el algoritmo seleccionado –árboles de decisión (*ID3–C4.5*)– sobre cada muestra. Para ello se calcula la precisión sobre el modelo trivial y se utilizan los valores de la precisión calculados en la tabla 5.1 del algoritmo *ID3–C4.5.* Con estos datos se obtiene el porcentaje de mejora de cada muestra. Los porcentajes más altos corresponden a los modelos S3, S4 y S5, siendo S5 el que corresponde con el mayor de todos (47.95 % de mejora sobre el modelo trivial).

La tabla 5.3 proporciona información de las muestras que son más sensibles con respecto al comportamiento verde y rojo, lo que permitirá minimizar el número de falsos negativos. A partir de los datos obtenidos de la matriz de confusión resultante de aplicar el algoritmo de árboles de decisión sobre cada muestra: sensibilidad al comportamiento verde y sensibilidad al comportamiento rojo, así como de los valores de la precisión de la muestra, error y la mejora sobre el modelo trivial presentados en la tabla 5.2, comprobamos que los modelos S1, S4 y S9 son los que presentan los mejores resultados

respecto a la sensibilidad al comportamiento verde. Los modelos S2, S3, S5, S7 y S8 son los que presentan mejores valores respecto a la sensibilidad al comportamiento rojo. De entre ellos, los modelos más equilibrados son el S2 y el S5, ambos con sensibilidad 1 respecto al comportamiento rojo, puesto que su sensibilidad al comportamiento verde es 0.98 y 0.96, respectivamente. Estos valores indican que S2 y S5 son las muestras que mejores resultados podrán obtener y menos falsos negativos generarán.

En la tabla 5.4 se visualiza la resistencia de cada muestra. Para ello, cada muestra se valida con la proporcionalmente contraria, calculando la matriz de confusión sobre cada muestra, que aportará la precisión sobre la resistencia del modelo y la sensibilidad al comportamiento rojo. Con el valor de la precisión de la resistencia del modelo y los datos aportados por la tabla 4.1 se calcula la mejora sobre el modelo trivial. Los resultados indican que S3, S4 y S7 obtienen valores de 100 % en la precisión sobre la resistencia del modelo; mientras que S2 y S5 obtienen un 99.98 %. S1 y S6 son las muestras que obtienen un menor valor en la sensibilidad al comportamiento rojo, por lo que generarían más falsos negativos que el resto de los modelos.

No utilizamos la muestra S_9 porque su mejora respecto al modelo trivial es cero.

En la tabla 5.5 se presentan los valores de las variables precisión del modelo, mejora sobre el modelo trivial y sensibilidad al rojo, que corresponden a los obtenidos de la tabla 5.3. Además, incluyen los valores de precisión de la resistencia, mejora sobre el modelo trivial y sensibilidad al rojo 5.4.

Para poder determinar cuáles son los mejores modelos en función de los valores obtenidos y dividirlos en conjuntos de datos ordenados, se utilizan cuartiles y, una vez asignados, se calculan sus medias para obtener valores representativos.

La tabla 5.6 y la figura 5.3 muestran los cuartiles de las variables y sus medias, y se observa que S3 y S5 obtienen las media más alta con un valor de 3.33, por lo que serán las muestras más adecuadas para realizar pruebas con datos reales.

6.2. Realización de pruebas con datos reales

Se ha probado el modelo S5 utilizando el árbol de decisión con *KNIME* 3.3.1 sobre dos conjuntos de datos reales. El primer conjunto de datos representa la actividad normal, es decir, sin registros correspondientes a ataques por *APT*. El segundo conjunto de datos contiene datos relativos a una *APT* que atacó una infraestructura real y que permaneció persistente durante 25 días, hasta que fue detectada por inspección y eliminada. Durante ese tiempo, la *APT* generó 3710 entradas en el registro de *logs* del *firewall*.

Para los experimentos se realizó un muestreo de los datos reales de los registros de *logs*, manteniendo la proporción de rojo/verde.

Las figuras 6.1, 6.2 y 6.3 muestran los valores de las matrices de confusión correspondientes a utilizar el algoritmo de árbol de decisión con tráfico real procedente de la infraestructura *TIC*. Las figuras 6.1 y 6.2 corresponden al uso de los modelos S5 y S3, respectivamente, con tráfico inocuo; mientras que la figura 6.3 es el resultado de utilizar el modelo S5 sobre tráfico real que contiene el ataque por *APT* sufrido por la infraestructura.

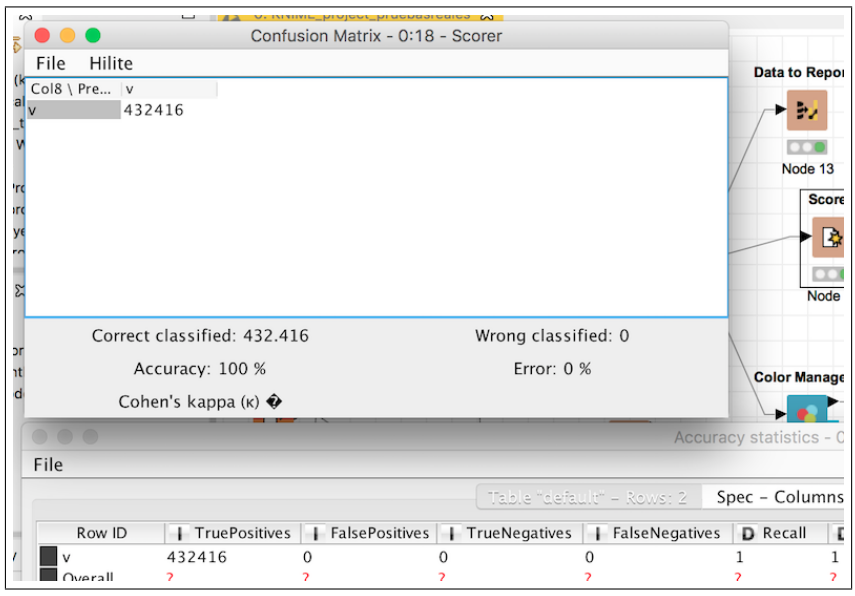


Figura 6.1: Matriz de confusión usando el modelo S5 y el árbol de decisión sobre un conjunto de datos de tráfico inocuo, D_h

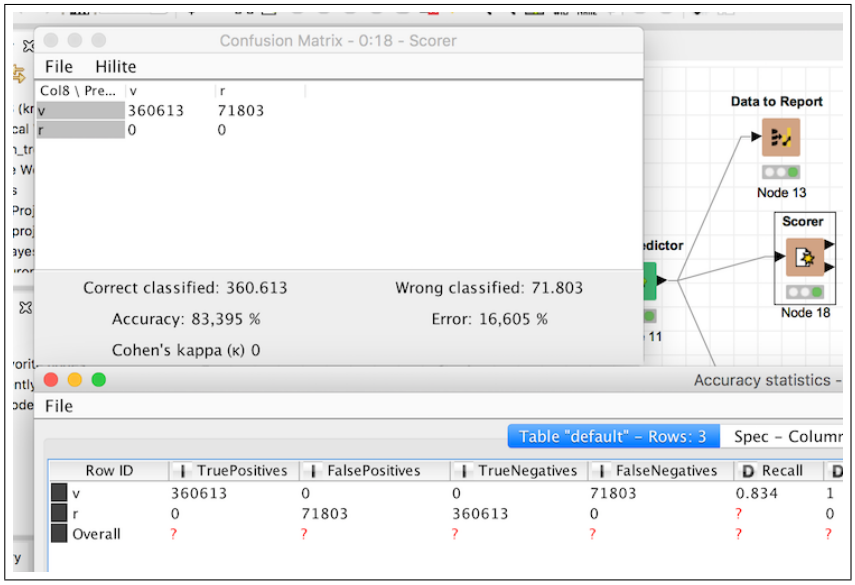


Figura 6.2: Matriz de confusión usando el modelo S3 y el árbol de decisión sobre D_h

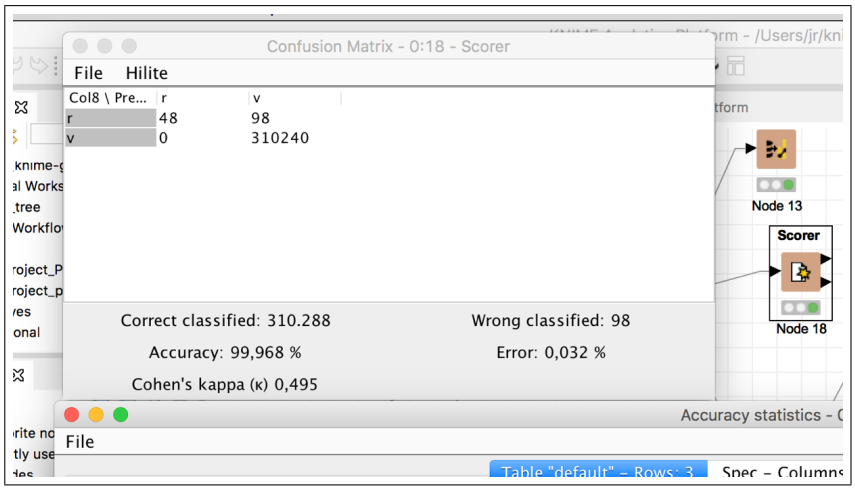


Figura 6.3: Matriz de confusión usando el modelo S5 y el árbol de decisión sobre un conjunto de datos afectados por una *APT* real

6.3. Análisis de los resultados

Las matrices de confusión muestran que el árbol de decisión *ID3–C4.5* proporciona mayor precisión y menor error que Naïve Bayes, la red neuronal

probabilística y las máquinas de soporte vectorial (SVM).

La matriz de confusión resultante de la prueba realizada con el conjunto de datos reales de tráfico inocuo sobre el modelo S5, mostrada en la figura 6.1, permite comprobar que el modelo consigue una precisión del 100 %, lo que indica que sería capaz de detectar correctamente el tráfico catalogado como inocuo.

La precisión del modelo S3 es menor que la del S5, y su tasa de errores es mayor. Su índice de error es del 16.605 %, mientras que su tasa de aciertos es del 83.395 % lo que indica que este modelo considera como malicioso al 16.605 % del tráfico real inocuo.

Los resultados mostrados en la figura 6.3 utilizando el modelo S5 sobre datos reales que contiene una APT real, muestran una precisión del 99.968 % y una tasa de errores del 0.032 %. Aunque genera alguna falsa alerta, el número de aciertos es suficientemente significativo como para generar alertas que permitan eliminar el ataque, ya que todos los registros provienen de la misma fuente.

Capítulo 7

Conclusiones y líneas futuras de investigación

7.1. Conclusiones

Como hemos visto a lo largo de la exposición de este trabajo de investigación, nuestro método se compone principalmente de tres enfoques.

Un primer enfoque relacionado con definir de una manera clara, concisa y actualizada el concepto de *APT*, cómo se comporta, sus escenarios, cómo se protege para no ser detectada, cuales son las medidas de seguridad que poseen los sistemas operativos más comunes (*Windows*, *Linux*, *Mac*) y qué hacen las *APTs* para evadirlas y robar información.

Un segundo enfoque, orientado a la educción del conocimiento experto y la utilización de minería de datos con los registros de *logs* del *firewall* de una infraestructura real para poder detectar una *APT*.

Un tercer enfoque dirigido a la creación de un sistema inteligente cimentado en técnicas de aprendizaje automático capaz de detectar comportamientos maliciosos que puedan intitularse como *APTs.*

La evaluación del sistema se realiza con conjuntos de datos del mundo real, de los registros de *logs* obtenidos de una infraestructura en producción que ha sufrido un ataque por *APT.*

Sobre estas bases de evaluación podemos concluir que:

1. Nuestro modelo ha demostrado ser eficaz, y utiliza tecnologías que no dependen de la arquitectura.

2. El modelo es capaz de detectar anomalías que son consecuencias de un ataque por *APT* en una infraestructura *TIC* concreta y tanto en conjuntos de datos reales, como sintéticos, como semi–sintéticos.

3. Del análisis de los resultados se desprende que el sistema de aprendizaje automático utilizado para la detección de *APT* es capaz de detectar y diferenciar el tráfico normal del tráfico anómalo.

4. El sistema de aprendizaje automático obtiene valores de tasas de acierto aceptables tanto para tráfico real como sintético.

5. El modelo creado funciona para el ataque recibido por la infraestructura y es extrapolable a infraestructuras similares.

6. Los resultados realizados con pruebas reales concluyen que la propuesta es adecuada para el objetivo de la detección temprana de *APT*, es decir, para la seguridad proactiva.

Como conclusión final cabe señalar que el mejor sistema de aprendizaje automático, de entre los analizados, y para el conjunto de datos utilizado, es el árbol de decisión (*ID3–C4.5*) utilizando la muestra S5.

Como aportaciones principales de la Tesis, se consideran las siguientes:

1. Redefinición y unificación del concepto de *APT*.

2. Utilización de tecnología *MPI* para preprocesamiento de los registros de *logs*.

3. Creación de un modelo de *APT* (de acuerdo con los registros de *log* del *firewall*).

4. Definición de un modelo de procesamiento de *log* para identificar peligros potenciales que no detectan ni los antivirus ni los *firewalls* y que está basado en conocimiento experto.

5. Establecimiento de una metodología de detección de *APTs* basada en minería de datos con un modelo de aprendizaje automático, fundamentado en conocimiento experto.

6. Diseño de un sistema de aprendizaje máquina adecuado para detectar la *APT* modelada en una infraestructura *TIC.*

7.2. Líneas futuras de investigación

El sistema propuesto ha de ser actualizado a medida que evolucione la infraestructura *TIC* a la que protege.

La continuación de este trabajo podría incluir las siguientes actividades:

1. Monitorización de los resultados sospechosos de ser *APT.*

2. Utilización de *HPC* para procesar los *logs* en tiempo real con el algoritmo de detección de *APT* seleccionado.

3. Utilización de *HPC* para conjuntos de entrenamientos masivos, que permitan el cálculo con grandes matrices.

4. Utilizar herramientas de *Big Data* para infraestructuras con grandes volúmenes de información y almacenamiento distribuido.

5. Como el modelo está enriquecido por el conocimiento experto basado en distintos escenarios y ataques por *APT*, se podría validar con otros casos de ataques reales.

6. Exportar el modelo a otras infraestructuras.

7. Implementar un lenguaje formal para traducir entre diferentes formatos de *logs*, utilizando estándares como *Archsight CEF* revisión 2.0 (*Common Event Format*).

Bibliografía

[1] CCN–CERT, "CCN–CERT," Centro Criptológico Nacional, España, 2014, p. 123

[2] E. U. de Informática Politécnica de Madrid, "Virus Creeper." [Online]. Available: http://www.eui.upm.es/museo_virtual/4g/rtmorris. [Accessed: 12–Apr–2014].

[3] CERTUY, Centro de Respuesta a Incidentes de Seguridad de Uruguay. [Online]. Available: http://www.cert.uy/inicio/novedades/amenazas_-y_alertas /predicciones+para+2013. [Accessed: 13–Dec–2014].

[4] B. H. Cymerman, "Irán sufre el mayor ataque cibernético de su historia," La Vanguardia, Jerusalén, 28-Sep-2010.

[5] J. L. Micó, "Aumentan los ataques cibernéticos en móviles y redes sociales," La Vanguardia, Barcelona, 28-Apr-2011.

[6] Norsecorp, "Ataques en tiempo real," 2016. [Online]. Available: map.norsecorp.com.

[7] INTECO, "¿Qué son las amenazas persistentes avanzadas (APTs)?", 2013, pp. 1–11.

[8] [21] K. Krombholz, H. Hobel, M. Huber, and E. Weippl, "Advanced social engineering attacks," Elsevier Ltd., 2014.

[9] J.H. Wieken, "Metadata for Data Marts and Data Warehouses. The Data Warehouse Concept.", Gabler, Wiesbaden, Alemania, 1998, 275–315.

[10] F. Corno, E. Sanchez, and G. Squillero, "On the evolution of core-war warriors," in Proceedings of the 2004 Congress on Evolutionary Computation, CEC2004, 2004, vol. 1, pp. 133–138.

[11] J. L. von Neumann, "Theory of Self–reproducing Automata.", 1967, p. 388.

[12] S. Posteguillo, "Africanus. El hijo del Cónsul.", España, 2011, pp 711.

[13] RAE, "RAE," Real Academia de la Lengua Española. [Online]. Available: www.rae.es. [Accessed: 15-Dec-2014].

[14] N. Marroquín, "Tras los pasos de un hacker.", Ecuador, 2010.

[15] A. M. V. Varela, "Introducción a la Informática y al uso y manejo de aplicaciones comerciales.", España, 2006.

[16] L. S. Penrose, "Self–reproducing machines.", Scientific American, 1959, vol. 200, no 6, p. 105–114.

[17] L. Kaspersky, "Securelist threats 1960s. Darwin." [Online]. Available: https://securelist.com/threats/1960s/. [Accessed: 12–Apr–2014].

[18] D. Sullivan, "The Shortcut Guide to Extended Validation *SSL* Certificates.", Reaper. 2007, p. 81.

[19] L. Kaspersky, "Securelist threats 1970s. Rabbit.", [Online]. Available: https://securelist.com/threats/1970s/. [Accessed: 08–Jun–2017].

[20] E. Cloner, "Virus Elk Cloner.", in Malware: Fighting Malicious Code, 2004, p. 647.

[21] F. B. Cohen, "Computer Viruses.", University of Southern California, 1985, p. 114.

[22] C. Rutstein, "Executive Guide to Computer Viruses.", DIANE Publishing, 1992, p. 60.

[23] Support Microsoft, “Cómo impedir y eliminar virus y otro malware.”, 2013. [Online]. Available: http://support.microsoft.com/kb/129972/es-es. [Accessed: 12-Jan-2015].

[24] Panda, “Panda Security.”, [Online]. Available: http://www.pandasecurity.com. [Accessed: 12–Apr–2014].

[25] V. G. Andy Clarke, “Computers & Security.”, Elsevier Ltd., 2000, vol. 19, no. 8, pp. 692–697.

[26] CyberNews “APT Group Launches Spear Phishing Before Zero-days Get Patched.”, 2016. [Online]. Available: https://www.quannsecurity.com/151116-APT-Group.php. [Accessed: 01-Jun-2017].

[27] J. Yao, J. Pang, Y. Zhang, Y. Yu and Lu, “A method and implementation of control flow obfuscation using SEH.”, 2012, pp. 336–339.

[28] J. C. Foster, V. Osipov, N. Bhalla, N. Heinen, D. Aitel, J. C. Foster, V. Osipov, N. Bhalla, N. Heinen, and D. Aitel, “Buffer Overflow Attacks.”, 2005.

[29] Q. Wei, T. Wei, and J. Wang, “Evolution of exploitation and exploit mitigation.”, 2011, vol. 51, no. 10, pp. 1274–1280.

[30] Support Microsoft, “How to enable Structured Exception Handling Overwrite Protection (SEHOP) in Windows operating systems.”, 2011.

[31] D. Dang, T.H.Y. P. Maniatis and Wagner, “The performance cost of shadow stacks and stack canaries.”, 2015, pp. 555–556.

[32] M. Prandini, M. Ramilli. "Return-oriented programming." IEEE Security & Privacy, 2012, 10.6 84-87.

[33] R. . Qiao, R.b , Zhang, M.a , Sekar, “A principled approach for ROP defense,” in 31st Annual Computer Security Applications Conference, ACSAC 2015; Los Angeles; United States; 7 December 2015 through 11 December 2015; Code 118804, pp. 101–110.

[34] R. E. Aditya K Sodd, "Targeted Cyber Attacks Multi–staged Attacks Driven by Exploits and Malware.", 2014.

[35] B. L. Sharp, G. D. Peterson, and K. Y. Lok, "Extending hardware based mandatory access controls for memory to multicore architectures.", 2008.

[36] A. M. Jai Prakash Jyotiyana, "Secure Authentication: Eliminating Possible Backdoors in Client–Server Endorsement.", Int. Conf. Comput. Model. Secur., 2016, pp. 606–615.

[37] Zinksecurity thinking solutions, "Advance Persistent Threats (APTs)." Curso formación, España, 2015.

[38] Gu, Guofei; Zhang, Junjie; Lee, Wenke. BotSniffer, "Detecting botnet command and control channels in network traffic.", 2008.

[39] NIST, "National Institute of Standards and Technology.", USA, 2015.

[40] C. Tankard, "Advanced Persistent threats and how to monitor and deter them.", Netw. Secur., vol. 2011, 2011, no. 8, pp. 16–19.

[41] C. Tankard, "New rules for combating new threats.", Comput. Fraud Secur., vol. 2014, 2014, no. 4, pp. 14–16, Apr.

[42] I. Friedberg, F., G. Settanni, and R. Fiedler, "Combating advanced persistent threats: From network event correlation to incident detection.", Comput. Secur., vol. 48, pp. 35–57, Feb. 2015. http://www.sciencedirect.com/science/article/pii/S0167404814001461.

[43] P. Giura and W. Wang, "Using Large Scale Distributed Computing to Unveil Advanced Persistent Threats.", ASE, 2012, pp. 1–13.

[44] R. Brewer, "Advanced persistent threats: Minimising the damage.". Elsevier BV, 2014, vol. 2014, no. 4, pp. 5–9.

[45] OSINT, "OSINT.", Open Source INTeligence. [Online]. Available: http://www.intelpage.info/open-source-intelligence-osint.html. [Accessed: 12–May–2014].

[46] A. Gómez. "Tipos de ataques e intrusos en las redes informáticas.", Ponencia, Escuela de Negocios Caixanova, 2013.

[47] J. Sammons, "The Basics of Digital Forensics.", Boston, Syngress, 2012, pp. 177.

[48] M. H. S. and R. S. Rob Shein aka Rogue Shoten, "Acknowledgments BT - Zero-Day Exploit.", Cyber-Fiction. Syngress, Rockland, 2004, p. 339.

[49] M. B. and J. C. Foster, "Hacking the Code.", Burlington: Syngress, 2004.

[50] Trendmicro, "Comprender un ataque *APT*.", 2013. [Online]. Available: http://www.trendmicro.es/grandes-empresas/ataques-dirigidos-avanzados/#comprender-un-ataque.

[51] J. O'GORMAN, D. KEARNS and AHARONI, Mati. Metasploit: the penetration tester's guide. No Starch Press, 2011.

[52] Oprea, A., Li, Z., Yen, T. F., Chin, S. H. and Alrwais, S.,"Detection of early–stage Enterprise infection by mining large–scale log data.", In 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2015, pp. 45–56.

[53] P. Giura and W. Wang, "Using Large Scale Distributed Computing to Unveil Advanced Persistent Threats.", Science Journal, 1(3), 2012, pp. 1–13 92–105.

[54] Amoroso, E.G., " Fundamentals of computer security technology.", Upper Saddle River, NJ, USA: Prentice – Hall, Inc., 1994.

[55] Schneier, B., " Attack Trees – Modeling Security Threats.", Dr. Dobb's Journal, December 1999.

[56] Skopik, F., Settanni, G., Fiedler, R. and Friedberg, I., " Semi–synthetic data set generation for security software evaluation.", In: 12th Annual Conference on Privacy, Security and Trust. IEEE, 2014, pp. 156–163.

[57] Veeramachaneni, K. and Arnaldo, I., " AI^2: training a big data machine to defend.", In: International Conference on Big Data Security. IEEE, New York, 2016.

[58] Shyu, M. L., Chen, S. C., Sarinnapakorn, K. and Chang, L., " A novel anomaly detection scheme based on principal component classifier.", In in Proceedings of the IEEE Foundations and New Directions of Data Mining Workshop, in conjunction with the Third IEEE International Conference on Data Mining (ICDM'03), 2003, pp. 172–179.

[59] Hawkins, Simon, et al. "Outlier detection using replicator neural networks.", International Conference on Data Warehousing and Knowledge Discovery. Springer Berlin Heidelberg, 2002.

[60] J. M. R. Mosso, "Ciberseguridad Inteligente.", 2015, p. 21, http://arxiv.org/abs/1506.03830. ArXiv.ID 1506.03830

[61] C. Tankard, "New rules for combating new threats.", Comput. Fraud Secur., 2014, vol. 2014, no. 4, pp. 14–16.

[62] I. Mandiant, "Informe Mandiant.", 2013. [Online]. Available: http://www.elladodelmal.com/2013/02/informe-mandiant-sobre-APT1-unit-61398.html. [Accessed: 12–May–2014].

[63] C. de APT, "Casos de APT´s.", 2012. [Online]. Available: http://www.slideshare.net/ansanz/capacidades-de-china-para-la-ciberguerra. [Accessed: 12–May–2014].

[64] L. Kaspersky, "Targeted Cyberattack Logbook.", [Online]. Available: https://apt.securelist.com/#secondPage.

[65] KasperskyLab, "EQUATION.", 2015. [Online]. Available: http://www.securitybydefault.com/2015/02/equation-el-apt-mas-sofisticado-hasta.html. [Accessed: 05-Mar-2015].

[66] C. F. Pereda, "Así se produjo el ciberataque ruso en la campaña electoral de EE UU, según el FBI.", El PAIS, Washington, 30-Dec-2016.

[67] National Institute of Standards and Technology, "CSRC AND NIST CYBERSECURITY for USA.", 2015, [Online]. Available: http://csrc.nist.gov/. [Accessed: 18–Jul–2016].

[68] R.J.Brachman, T. Khabaza, W. Kloesgen, G. Piatetsky–Shapiro and E. Simoudis, "Industrial Applications of Data Mining and Knowledge Discovery.", 1996, in Proc. Of the Second International Conference on Knowledge Discovery and Data Mining (KDD–96), Menlo Park, Calif.: AAAI Press.

[69] T.M. Connolly, and C.E. Begg, "Database systems: a practical approach to design, implementation, and management. Pearson Education.", 2005.

[70] K. Hammond, Case–based planning: "Viewing planning as a memory task.", Elsevier, 2012.

[71] Dietterich, G. Thomas, and S. Ryszard Michalski. "A comparative review of selected methods for learning from examples.", Machine Learning. Springer Berlin Heidelberg, 1983, 41–81.

[72] Fisher, H. Douglas, "Knowledge acquisition via incremental conceptual clustering.", Machine learning 2.2, 1987, 139–172.

[73] J.G. Carbonell, "Derivational analogy: A theory of reconstructive problem solving and expertise acquisition.", 1985.

[74] Langley, Pat. "Areas of application for machine learning.", In Proceedings of the Fifth International Symposium on Knowledge Engineering, Rank Xerox Española, S.A., 1992, pages 51–58, Madrid, Spain.

[75] D. Borrajo, J. González and P. Isasi, "Aprendizaje Automático.", Editorial Sanz y Torres, S.L., España, pp. 563.

[76] J. Ross Quinlan. "Induction of decision trees. Machine Learning.", 1986, 1(1):81–106.

[77] J. Ross Quinlan. "Learning logical definitions from relations. Machine Learning.", 5(3):239–266, August 1990.

[78] J. Ross Quinlan. “C4. 5: programs for machine learning.”, Elsevier, 2014.

[79] TS Lim, WY Loh and YS Shih. “A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-three Old and New Classification Algorithms.”, 1999

[80] Shannon, Claude Elwood. "A mathematical theory of communication.", ACM SIGMOBILE Mobile Computing and Communications Review 5.1, 2001, 3-55.

[81] Kotsiantis, Sotiris B., I. Zaharakis, and P. Pintelas. "Supervised machine learning: A review of classification techniques.", 2007, 3-24.

[82] Langley, Pat, W. Iba, and K. Thompson. “An analysis of Bayesian classifiers.”, Aaai. Vol. 90. 1992.

[83] Xu, Lei, A. Krzyzak, and Ching Y. Suen. “Methods of combining multiple classifiers and their applications to handwriting recognition.”, IEEE Transactions on systems, man, and cybernetics, 1992, 22.3: 418–435.

[84] P. Larrañaga, I. Inza, and A. Moujahid, “Tema 6. Clasificadores Bayesianos.”, Departamento de Ciencias de la Computación e Inteligencia Artificial–Universidad del País Vasco-Euskal Herriko Unibertsitatea, 1997.

[85] D. Singh, C.K. Reddy, “A survey on platforms for big data analytics Journal of Big Data.”, 2015, 2 (1), art. no. 8.

[86] A. K. Jain, “Data clustering: 50 years beyond K-means.”, Pattern recognition letters 31.8, 2010, 651-666.

[87] D. E. Rummelhart, and J. L. McClelland, “Parallel Distributed Processing, Vol. 1: Foundations.”, 1986.

[88] J. L. Mc Clelland, and D. E. Rumelhart. “Parallel distributed processing, Vol. 2. Psychological andbiological models.”, 1986.

[89] M. R. Berthold, and J. Diamond, "Constructive training of probabilistic neural networks.", Neurocomputing, 1998, 19(1), 167–183.

[90] V. Vapnik and C. Cortes, "Support–vector networks. Machine Learning.", Kluwer Acad. Publ. Bost., 1995, p. 273 297.

[91] E. J. Carmona Suárez, "Máquinas de Vectores Soporte (SVM).", Dpto. Intel. Arti cial, ETS Ing. Informática (UNED), 2016, pp. 1–27.

[92] D. Masip, "SVM Tolerancia datos no lineales.", Universidad Oberta de Cataluña.

[93] V. Vapnik, "The nature of statistical learning theory.", Springer science & business media, 2013.

[94] B. Schölkopf, "The kernel trick for distances.", Advances in neural information processing systems. 2001.

[95] R. Kohavi and F. Provost. "Confusion matrix.", Machine learning, 1998, 30.2-3 271-274.

[96] S. Visa, B. Ramsay, A.L. Ralescu and E. Van Der Knaap, "Confusion Matrix-based Feature Selection.", In MAICS, 2011, pp. 120-127.

[97] "Livermore Computing Center (LC) is home to a first–class computational infrastructure that supports the computing requirements of the Laboratory's research scientists.", [Online]. Available: https://computing.llnl.gov/tutorials/mpi/

[98] Fundación Computación y Tecnologías Avanzadas de Extremadura (COMPUTAEX), "Supercomputador Lusitania.", 2017. [Online]. Available: http://www.cenits.es/cenits/supercomputacion-en-extremadura.

[99] La Fundación Centro de Supercomputación de Castilla y León (FCSCL), "Supercomputador Caléndula.", 2017. [Online]. Available: https://www.fcsc.es/index.php/servicios.

[100] KNIME GmbH, "Open for innovation Knime.", [Online]. Available: www.knime.org. [Accessed: 01-Jun-2016].

[101] M. Shafer, J. Agrawal and R. Mehta, "A scalable parallel classifier for data mining.", En Proc. 1996 Int. Conf. Very Large Data Bases, 1996, pp. 544–555

[102] J. Platt, "Fast Training of Support Vector Machines Using Sequential Minimal Optimization.", Microsoft Res., 1998.

[103] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and M. K.R.K., "Improvements to Platt´s SMO Algorithm form SVM Classifier Design.", Control Div. Dept. Mech. Prod. Eng. Natl. Univ. Singapore Tech. Rep. CD–99–14.

[104] V. López, A. Fernández, S. García, V. Palade and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics.", Information Sciences, 2013, 250, 113–141.

[105] H. Han, W.Y. Wang and B.H. Mao, Borderline–SMOTE: "A new over–sampling method in imbalanced data sets learning.", In: Proceedings of the 2005 International Conference on Intelligent Computing (ICIC'05), Lecture Notes in Computer Science, 2005, 3644, 878–887.

[106] H. He, Y. Bai, E.A. Garcia and S. Li, ADASYN: "Adaptive synthetic sampling approach for imbalanced learning.", In: Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IJCNN'08), 2008, 1322–1328.

[107] M.M. López–Cabeceira, H. Diez–Machío, M.T. Trobajo and M.V. Carriegos, "Spectra analysis in detection of traces of explosives.", Int. J. Modern Phys. B, 2012, 26(25), 1246013.

[108] D. Forte, "Slammer — the Return of the Network Nightmare.", Netw. Secur., 2003, vol. 2003, no. 2, pp. 17–18.

[109] "Detectan el primer virus para Mac OS X," El Mundo, España, 20-Feb-2006.

[110] "Viruslist," 2015. [Online]. Available: http://www.viruslist.com/sp/. [Accessed: 13–Feb–2015].

[111] Expertos, "Introduction to Cyber–Warfare.", Elsevier, 2013, p. 336.

[112] Symantec, "W32.DUQU.", white Pap. symantec, 2011, p. 46.

[113] PandaLabs, "Informe Seguridad 2011.", 2017. [Online]. Available: http://www.pandasecurity.com/spain/mediacenter/src/uploads/2014/07/Informe-Anual-PandaLabs-2011.pdf. [Accessed: 14–May–2017].

[114] PandaLabs, "Informe Seguridad 2012.", 2017. [Online]. Available: http://www.pandasecurity.com/spain/mediacenter/src/uploads/2014/07/Informe-Anual-PandaLabs-2012.pdf. [Accessed: 14–May–2017].

[115] D. Lee, "Flame: Massive cyber–attack discovered, researchers say.", BBC News, BBC News Technology, 2012.

[116] PandaLabs, "Informe Seguridad 2013.", 2017. [Online]. Available: http://www.pandasecurity.com/spain/mediacenter/src/uploads/2014/07/Informe-Anual-PandaLabs-20132.pdf. [Accessed: 14–May–2017].

[117] PandaLabs, "Informe Seguridad 2014.", 2017. [Online]. Available: http://www.pandasecurity.com/spain/mediacenter/src/uploads/2015/02/Pandalabs2014-es.pdf. [Accessed: 14–May–2017].

[118] PandaLabs, "Informe Seguridad 2015.", 2017. [Online]. Available: http://www.pandasecurity.com/spain/mediacenter/src/uploads/2014/07/Pandalabs-2015-anual-ES.pdf. [Accessed: 14–May–2017].

[119] P. G. M. Ballesteros, "Maniobras de ciberguerra a orillas del Atlántico.", El PAIS, Londres / Washington / Madrid, 17-Enero-2015.

[120] PandaLabs, "Informe Seguridad 2016.", 2017. [Online]. Available: http://www.pandasecurity.com/spain/mediacenter/src/uploads/2016/12/Pandalabs-2017-predicciones-es.pdf. [Accessed: 14–May–2017].

[121] J. Albors, "Informe Seguridad 2016-2017 EsetNod32.", 2017. [Online]. Available: http://noticias.eset.es/aggregator/sources/1. [Accessed: 15–May–2017].

[122] INCIBE, J. R. Moya, "APTs: la amenaza oculta.", https://www.certsi.es/blog/apt-amenaza-oculta. 2015

[123] Centro de Excelencia, J.R. Moya, "¿Tenemos la seguridad de que nuestros ordenadores no forman parte de una APT?.", http://www.eoi.es/fdi/extremadura/innovación-s %C3 %AD-pero-con-seguridad, 18-Junio-2015

Apéndice A

Ataques representativos de 2003 a 2017

Virus/año Nombre Técnico	**Afecta a:**	**Técnica Utilizada**	**Ataque**	**Síntomas/Referencia**
Slammer/2003 W32/SQL Slammer	*SQL sin Service Pack* 3	Desbordamiento de buffer	Denegación de Servicio (*DoS*)	Ralentiza el servidor, aumenta el tráfico por el protocolo *UDP*, puerto 1434 [108]
Cabir/2004 Cabir.A.worm	Teléfonos móviles con *Symbian* que tengan *bluetooth*	Se replica por bluetooth a todos los móviles que estén conectados en ese instante	Crear los ficheros caribe.app, .rsc, .hld y .sys	Muestra el mensaje "CARIBE" en la pantalla del móvil cada vez que se enciende [24]
Leap/2006 Leap.A	*Mac OS X*	Se propaga por la red a través de una vulnerabilidad del protocolo Bonjour	A través de mensajería instantánea ichat, camuflado en una imagen *jpg*	Marca con el texto "oompa" todos los archivos, editándolos y dejándolos inservibles [109]

Zhelatin/2007 *Worm.win32.Zhelatin*	*Windows 95/98/ME/XP/NT/2000 y 2003*	Vulnerabilidades en sistemas *Windows*. Se caracteriza porque plasma todos los avances técnicos hasta este momento, reflejados en este virus, de hecho se inyecta en procesos del sistema en ejecución	Email con ficheros adjuntos, redes P2P, Servidores Webs	Ralentiza el sistema, desactiva los procesos del antivirus, pérdida de productividad en la red, degradación del ancho de banda [24]

				Vulnerabilidad MS08—067, que	
Koobface/2008	*Windows*	Desbordamiento de		permite la ejecución	Desactiva los servicios de:
W32/koobface.GQ.worm	*XP/Vista/2000/server*	*buffer* por		de código remoto en el	*Windows Update*, *Security*
	2003/server 2008	vulnerabilidades en		servidor	*Center*, *Defender*, *Error*
		entornos *Windows*			*Reporting*. Se adhiere a
					servicios como *svchost.exe*,
					explorer.exe, *services.exe*.
					Bloquea las cuentas de
					usuario, los antivirus no
					pueden actualizarse,
					congestiona la red [24]

Conficker/2008 Conficker	*Windows XP/Vista/2000/server 2003/server 2008*	Desbordamiento de *buffer* por vulnerabilidades en entornos *Windows*	Vulnerabilidad MS08–067, que permite la ejecución de código remoto en el servidor	Desactiva los servicios de: *Windows Update, Security Center, Defender, Error Reporting.* Se adhiere a servicios como *svchost.exe, explorer.exe, services.exe.* Bloquea las cuentas de usuario, los antivirus no pueden actualizarse, congestiona la red [24]

			La víctima para acceder a la página porno de manera gratuita, descarga el	
FakePlayer/2010 Trojan–SMS.AndroidOS.FakePlayer Android	*Smartphones* con	Accesos a búsqueda de material pornográfico	fichero *pornoplayer.apk*, se instala el troyano y envía 4 sms a números cortos de pago	Envíos de *SMS* masivos a números cortos ubicados en Rusia [110]

Stuxnet/2010 Stuxnet	Sistemas SCADA (*Supervisory Control And Data Acquisition*), sistemas usados en infraestructuras de servicios como gas, electricidad, agua, etc) con plataformas *Windows XP/Vista/2003/7/2008* que utilicen *Wincc* de *Siemens*	Vulnerabilidad MS10–046 de *Windows* que afecta a los accesos directos y que permiten ejecutar código remoto. La infección se produce a través de dispositivos extraíbles como llaves USB	Ejecuta código remoto e instala *rootkits* (acceso de privilegio continuo oculto a los administradores). Instalan archivos con firmas digitales legítimas robadas a otras empresas, haciéndose pasar por archivos legítimos	No existen síntomas. Encontramos diferentes opiniones si se puede considerar una *APT* [24] [111]

				La víctima recibe un documento *Word* que al abrirlo queda infectado	
Duqu/2011 W32.Duqu	Es similar a *Stuxnet*, de hecho el binario es prácticamente el mismo, no obstante *Stuxnet* destruye y *Duqu* recopila información de la infraestructura	Ingeniería Social y vulnerabilidades de *Microsoft Word*			No se detecta síntoma alguno. Al igual que *Stuxnet*, algunos lo consideran una *APT* [112]
2011	Redes Sociales	Ingeniería Social y vulnerabilidades de sistemas	*Facebook, Twitter, Google+*		Hackearon cuentas de *Fox News*, *Mark Zuckerberg* y relacionadas con la muerte de *Steve Job* [113]
2011	Información sensible	Ingeniería Social y vulnerabilidades de sistemas	*The Pentagon Federal Credit Union*		Obliga a la Comisión Europea a suspender el sistema de comercio de derechos de emisión de CO2 [113]

2011	Sistemas de pagos *on line*	Ingeniería Social y vulnerabilidades de sistemas	*Webmoney, MoneyBookers,* el Fondo Monetario Internacional	Robos de información de cuentas y tarjetas [113]
2011	Información sensible	Ingeniería Social y vulnerabilidades de sistemas	*PlayStation Network, Sega Pass, Sony Online*	Robos de información de más de 100 millones de usuarios [113]
2011	Ciberguerra	Ingeniería Social y vulnerabilidades de sistemas	Compañías Energéticas de *EE.UU.*, Departamento de Defensa de los Estados Unidos, *Mitsubishi Heavy Industries*	Información muy sensible, en especial de misiles guiados, equipamiento para centrales nucleares y motores para cohetes. Aviones no tripulados (*UAV*) utilizados por Estados Unidos [113]

2011	Ciberactivismo. *Anonymous*, *LulzSec*	Ingeniería Social y vulnerabilidades de sistema	*Fox*, *CBS*, *CIA*, *OTAN*	En defensa de *Wikileaks*, roban 90.000 cuentas de correo de militares, documentación, tarjetas de crédito. Caen la web de la CIA [114]
2012/Virus de la policía	Teléfonos móviles, ordenadores	Ingeniería Social y vulnerabilidades de sistema	Países de la Unión Europea	Suplantan a la policía del país en cuestión solicitando un pago o bloqueo del equipo [114]
2012	Redes sociales	Ingeniería Social y vulnerabilidades de sistema	*Facebook*, *Twitter*, *Linkedin*	Ataques más virulentos y sofisticados robando 6.5 millones de contraseñas [114]
2012	Ordenadores	Ingeniería Social y vulnerabilidades de sistema	*Ransomware*	Piden un rescate por desencriptar los ficheros [114]
2012	encriptación de ficheros *Anonymous*, *LulzSec*	Ingeniería Social y vulnerabilidades de sistema	Ministerio de Justicia americano, Universal Music, Vaticano	Caída de servicios y robo de bases de datos [114]

2012	Ordenadores, teléfonos móviles, *tablets*, etc	Ingeniería Social y vulnerabilidades de sistema	*Youporn, Dropbox, Sony music, Wikipedia, Adobe, Blizzard* o la agencia de noticias *Reuters*	Robos de información y publicación de datos sensibles [114]
2012	*Ciberguerra*	Ingeniería Social y vulnerabilidades de sistema	China, EE.UU., Japón, *Al Qaeda*, Irán, Corea del Norte	Ataques y contraataques de todo tipo. Se desarrollan las primeras ciberarmas para identificar, localizar y desactivar ciberataques [114]

			Graba audio, captura pantallas, pulsaciones de teclado, tráfico de red, graba conversaciones de *Skype*, controla el *bluetooth* y exfiltra toda la información	
Flame/2012 Flame	Sistemas *Windows* con objeto de ciberespionaje	*USB* utilizando vulnerabilidades de los sistemas *Windows* existentes en esos momentos. Es el más sofisticado y refinado hasta la fecha		No se detectan síntomas. Se cree que no fue un trabajo de cibercriminales independientes, más bien, que detrás había un gobierno, ya que afectó a países como Irán, Israel, Siria, Líbano, Arabia Saudí y Egipto [115]
2013	Ordenadores encriptación de ficheros	Ingeniería Social y vulnerabilidades de sistema	*Ransomware*	Fuerte resurgimiento de estos ataques más sofisticado con *CryptoLocker* con cifrado asimétrico [116]

2013	Ordenadores, teléfonos móviles, *tablets*, etc	Ingeniería Social y vulnerabilidades *Java*	*Twitter, Facebook, Apple, Evernote, Adobe, Microsoft* y los juegos de rol on line como *World of Warcraft Armory*	Robos de información sensible, cuentas de correo, contraseñas, etc [116]
2013	*Ciberguerra* grupo *Syrian Electronic Army, Syrian Liberation Army*	Ingeniería Social y vulnerabilidades de sistema	*New York Time, Twitter* trabajadores de la Casa Blanca, *NASA*, la Reserva Federal Estadounidense, cuentas de *Twitter* de *Barack Obama*	Sufren ataques, caídas de servicios y robos de información [116]
2013	*Ciberataques* desde China	Ingeniería Social y vulnerabilidades de sistema	*Associated Press, New York Times*, *Wall Street Journal, The Washington Post*	Sufren ataques, caídas de servicios y robos de información. El *Dow Jones* cae 145 puntos [116]

			EADS (Eruropean Aeronautic Defence and Space Company)	
2013	*Ciberataques* desde China	Ingeniería Social y vulnerabilidades de sistema	fabricante del *Eurofighter* y dueña de *Airbus*	Acceden a información sensible de misiles *Patriot* o de aviones caza en desarrollo como el *F—35* [116]
			Target Corp, Korea Credit Bureau (KCB), Corea del Sur, La Oficina Federal para la Seguridad de la Información alemana	Robos de cuentas de correo electrónico, cuentas de clientes e información sensible [117]
2014	*Ciberataques*	Ingeniería Social y vulnerabilidades de sistema por fin de ciclo del *Windows XP*, fallos en las librerías *OpenSSL*		

2014	Ordenadores, teléfonos móviles, *tablets*, etc.	Ingeniería Social y vulnerabilidades de sistema por fin de ciclo del *Windows XP,* fallos en las librerías *OpenSSL,* el agujero de seguridad en *Bash* de *Linux* y *Mac*	Y*ahoo, Orange, Forbes, eBay, Spotify, Domino´s Pizza, iCloud, Sony, Xbox Live, PlayStation Network*	Robos de credenciales y ataques de diversa índole [117]
2014	*Ciberdelicuencia*, *Syrian Electronic Army*	Utilizaron las noticias del vuelo de *Malasia Airlines MH 370*		Utilizaron las noticias de *Facebook.com y Twitter* como lanzadera de sus ataques [117]

				Usuarios y mandatarios de distintos gobiernos capturando imágenes mediante sus *webcams* entre los que se encontraba la canciller *Ángela Merkel*	*Eduward Snowden*, filtra nuevos documentos relacionados con labores de espionaje de la *NSA* y el *GHCQ* [117]
2014	*Ciberespionaje*	Ingeniería Social y vulnerabilidades de sistema por fin de ciclo del *Windows XP*, fallos en las librerías *OpenSSL*, el agujero de seguridad en *Bash* de *Linux* y *Mac*	Ordenadores, teléfonos móviles, *tablets*, etc.		
2015	Ordenadores, teléfonos móviles, *tablets* e Internet de las Cosas (*IoT*)	Ingeniería Social y vulnerabilidades de sistema, fallos de seguridad en *Adobe Flash*		*Ryanair*, *AdultFriendFinder* y *Ashley Madison*	Robos de información, publicación y chantaje [118]

2015	Ordenadores, teléfonos móviles, *tablets* e Internet de las Cosas (*IoT*)	Ingeniería Social y vulnerabilidades de sistema, fallos de seguridad en *Adobe Flash*	Cadenas hoteleras como *El Hard Rock Hotel & Casino de las Vegas,* cadena *Hilton*, *Trump Hotels*, *Starwood*, etc.	Robos de información sensible, tarjetas de créditos [118]
2015	Ordenadores, teléfonos móviles, *tablets* e Internet de las Cosas (*IoT*)	Ingeniería Social y vulnerabilidades de sistema, fallos de seguridad en *Adobe Flash*	*JPMorgan, T–Mobile, Hello Kitty, Candy Crush*	Robos de información sensible, datos bancarios y tarjetas de crédito [118]
2015	Internet de las Cosas (*IoT*)		J*eep Cherokee, Land Rover, Tesla, Toyota*, etc.	Hackeos y la toma de control de algunos elementos de vehículos [118]

2015	*Ciberguerra, Anonymous*	Ingeniería Social y vulnerabilidades de sistema, redes sociales		*Anonymous* lanza una campaña contra *ISIS* publicando sitios webs, cuentas y redes sociales de sus miembros [118]
2015	*Ciberguerra*	Ingeniería Social y vulnerabilidades de sistema, redes sociales		*Edward Snowden* filtra nuevos datos a la prensa y se confirma el robo de Terabytes de datos sobre el caza *F—35* [118]
2015	*Ciberguerra*		Estados Unidos e Inglaterra	Se crean cibercélulas de ciberagentes con miembros del *MI5* y *FBI*, realizando simulaciones de ataques al Banco de Inglaterra y a *Wall Street* [119]
2016—2017	Ordenadores, teléfonos móviles, *tablets*, ficheros	Ingeniería Social y vulnerabilidades de sistema	*Ransomware*	Los *Ransomware* que pasan de cifrar documentos a cifrar el *MBR (Master Boot Record)* [120]

2016–2017	*IoT (Internet of Things)*	Vehículos, viviendas, etc.		Atacando desde termostatos de viviendas, sistemas de frenado, aceleradores o volantes con el vehículo en marcha a secuestros de las *Smart TV* [120]
2016–2017	Ordenadores, teléfonos móviles, *tablets*, *TPVs*	Ingeniería Social y vulnerabilidades de sistema, redes sociales	*DropBox, Yahoo, MySpace, Linkedin*, Banco Central de Bangladesh, restaurantes y lugares de comidas rápidas	Robos de información sensible, robo de datos de usuarios, 900 millones de cuentas de correo comprometidas [120]
2016–2017	Ordenadores, teléfonos móviles, *tablets*	Ingeniería Social y vulnerabilidades de sistema, redes sociales	*Netflix, Twitter, Amazon o The New York Times*	Caídas de servicio que provocaría pérdidas millonarias [120]
2016–2017	*Ciberguerra*, *Daesh*	Ingeniería Social y vulnerabilidades de sistema, redes sociales		Estados Unidos se lanzan ataques contra *Daesh* [120]

2016–2017	*Ciberguerra*, Corea del Norte	Ingeniería Social y vulnerabilidades de sistema, redes sociales		Corea del Norte atacando sedes gubernamentales de Corea del Sur [120]
2016–2017	*Ciberguerra*	Ingeniería Social y vulnerabilidades de sistema, redes sociales	La Agencia Nacional de Seguridad de Estados Unidos (*NSA*)	Hackearon la *NSA* y robaron información de ciberarmas poniéndose a la venta en el mercado negro. Los Cibersoldados alcanzarán su máximo auge [120]
Mayo 2017	Ordenadores, teléfonos móviles, *tablets*, encriptación de ficheros	Ingeniería Social y vulnerabilidades de sistema	*Ransomware*. Telefónica, Servicio Público de Salud en Reino Unido, el Ministerio del Interior Ruso o la empresa de paquetería *Fedex* entre otras muchas del mundo	El mundo recibe el mayor ataque de *Ransomware* perpetrado hasta ahora, son objeto del ataque de *WannaCryptor*, también conocida como *Wanna Cry [121]*

Apéndice B

Campos que componen el registro de log

Campo	Descripción	Ejemplo
itime	Itime en el cual el evento fué recibido por el firewall	itime=1455521331
date	Día, mes y año cuando el registro de log fué creado.	date=2016-07-04
time	Hora cuando el registro es grabado	time=14:26:59
devid	Número de serie del dispositivo	devid=FGxxxxxxxx
vdom(vm)	Domínio virtual en el cual el log fué grabado	vd=vdom1
logid	Codificación del tipo, subtipo, evento y mensaje	logid=0001000014
msg	Identificador del mensaje	msg=000100000012
type	Tipo de mensaje	type=traffic
subtype	Subtipo de mensaje	subtype=forward
app	Nombre de la aplicación	app=http. browser_Chrome
appcat	Categoría de la aplicación	appcat=Web.Others
appid	Identificador de la aplicación	appid=34039
applist	Nombre del profile para el control de aplicaciones	applist=Aplicaciones Bloqueadas

Sigue en la página siguiente.

Campo	Descripción	Ejemplo
attack	Nombre del posible ataque	attack=http. browser_Chrome
devname	Nombre del dispositivo	devname=MiFirewall
dstcountry	Nombre del país de la IP destino	dstcountry=Spain
dstintf	El interface de salida hacia el destino	dstintf=port16
dstip	IP de destino	dstip=10.10.20.10
dstport	Puerto de destino	dstport=80
duration	Duración de la conexión	duration=1
hostname	Nombre del host o url	host=www.ppppp.com
Level	Nivel de prioridad del registro de log	level=warning
policyid	Id de la política que genera el log	policyid=63
Protocol	Código del protocolo usado	proto=6
rcvdbyte	Número de bytes recibidos	rcvdbyte=528
rcvdpkt	Número de paquetes recibidos	rcvdpkt=5
sentbyte	Número de bytes enviados	sentbyte=949
sentpkt	Número de paquetes enviados	sentpkt=6
service	Nombre del servicio ejecutado	service=ppppp service
sessionid	Identificador de la sesión	sessionid=20918574
srccountry	Nombre del país de la IP origen. Reserved=red interna	srccountr=Reserved
srcintf	Interface de entrada desde el origen	srcintf=port10
srcip	Dirección IP que inició la conexión	srcip=10.6.30.220
srcport	Puerto del tráfico origen	srcport=54705
status	Estado de la sesión: close, deny, start, timeout, etc.	status=close
trandisp	Tipo de traslación NAT:dnat, noop, snat, snat+dnat	trandisp=snat
transip	IP destino del NAT (IP del router)	transip=xxx.xxx.xxx.xxx
transport	Puerto utilizado para el NAT	transport=43816
utmaction	Acción de seguridad realizada por UTM	utmaction= passthrough

Sigue en la página siguiente.

Campo	**Descripción**	**Ejemplo**
utmevent	Evento que genera el control utm	utmevent=app-crtl

Tabla B.1: Descripción de los campos del registro de *log* del tráfico en la red

Apéndice C

Publicaciones

1. Artículo en el Blog del Instituto Nacional de Ciberseguridad de España con el título "*APT*s, la amenaza oculta" [122].

2. Ponencia sobre *"Amenazas Persistentes Avanzadas APT"* en la Universidad de Extremadura (Mérida). Centro de Excelencia en Gestión de la Innovación promovido por la Escuela de Organización Industrial (EOI) y el Gobierno de Extremadura [123].

3. J.R. Moya, N. DeCastro, R.A. Fernández, "Expert knowledge and data analysis for detecting advanced persistant threats", Open Mathematics (2017) (Ref.- OPENMATH-D-16-00253R1, aceptado 04-05-2017). Indicios de calidad: Open Mathematics está catalogada en los índices de impacto JCR de 2016 en Q2.