UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

TRABAJO FIN DE GRADO

Reconocimiento de Patrones Aplicado a la Detección de Intrusiones en Redes de Ordenadores

Ana Chevasco

Tutor: Doroteo Torre Toledano

Mayo 2016

Reconocimiento de Patrones Aplicado a la Detección de Intrusiones en Redes de Ordenadores

AUTOR: Ana Chevasco
TUTOR: Doroteo Torre Toledano

Biometric Recognition Group - ATVS
Dpto. de Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Mayo 2016

Resumen (castellano)

El Trabajo Fin de Grado que se presenta consiste en el análisis de una base de datos de detección de intrusiones en redes Wi-Fi (AWID - Aegean Wireless Intrusion Detection) y en la aplicación de técnicas de reconocimiento de patrones para tratar de detectar las tramas correspondientes a ataques. La base de datos reducida con la que se ha trabajado incluye cerca de 1.8 millones de tramas de entrenamiento y más de medio millón de tramas de test.

El trabajo parte de los resultados de los creadores de la base de datos, y trata de reproducirlos y mejorarlos. Para ello en primer lugar se ha realizado un análisis y selección de atributos manual de la base de datos, pasando de 155 a 82 atributos. En estas pruebas se obtuvieron resultados prácticamente idénticos a los obtenidos por los creadores de la base de datos con todos los atributos. Posteriormente se realizó una selección automática de atributos sobre los 82 atributos restantes, seleccionando automáticamente 18 atributos con los que una vez más se consiguieron resultados similares a los obtenidos por los creadores de la base de datos con una selección manual de 20 atributos.

Finalmente, las últimas pruebas que se realizaron, y las más innovadoras, fueron reorganizar la base de datos en función de las comunicaciones entre dos direcciones MAC e introducir una ampliación del contexto temporal a cada trama. De tal manera que cada trama pase de a tener la información temporal de dos tramas anteriores y dos tramas posteriores a ella misma, dentro de una comunicación entre dos direcciones MAC. Esta expansión de contexto ha conseguido mejorar sustancialmente la detección de un tipo de ataques cuya detección trama a trama resultaba poco efectiva, mejorando así los resultados obtenidos pos los creadores de la base de datos.

Palabras clave (castellano)

Algoritmos de aprendizaje automático, Clasificación, Redes Wi-Fi, Ciberseguridad.

Abstract (English)

This bachelor Thesis here presented, consists in the analysis of a dataset for detection of attacks in wireless networks (AWID – Aegean Wireless Intrusion Detection), and in the application of pattern recognition techniques to detect frames corresponding to an attack, or part of one. The reduced database which we have worked on has over 1.8 million of training frames and over half a million of test frames.

The worked performed in this thesis is based on the results obtained by the creators of this dataset, and tries to reproduce them and even improve them. For that, the first step was to analyze each attribute and perform a manual attribute selection of the dataset. After this process it was possible to go from 155 to 82 attributes. In the tests performed with this new and reduced database, the results obtained were nearly identical as those obtained by the creators of the database. Later, an automatic attribute selection was performed over the 82 final attributes, selecting automatically 18 attributes with which, once again, it was possible to obtain similar results as the ones presented by the creators of the database with a manual selection on 20 attributes.

Finally, the last tests the were performed, and also the most innovative, were to reorganize the dataset according to the communications between two MAC addresses and introduce temporary context expansion to each frame. This way, each frame would have the temporary information of two previous frames, itself and two posterior frames, within a communication among two MAC addresses. This context expansion has improved considerably the detection of a type of attack for which detection on a frame by frame basis was not very effective, upgrading the results obtained by the creators of the dataset.

Keywords

Machine learning algorithms, Classification, Wi-Fi networks, Cybersecurity.

Agradecimientos

A todas las personas que me han ayudado a llegar hasta aquí y han hecho este proyecto posible. En especial dedico este trabajo a Lucas Chevasco.

ÍNDICE DE CONTENIDOS

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

1 Introducción

1.1 Motivación

En la actualidad los dispositivos inteligentes están constantemente conectados a Internet y cada vez más la información reside en la nube. Esto conlleva muchas ventajas, pero debido a la gran vulnerabilidad de los sistemas esto también supone un riesgo. De acuerdo con un informe anual de Cisco del 2014 [1] la empresa detectaba 50.000 intrusiones diarias analizando el tráfico de red. El gran impacto de estas intrusiones sobre las redes de ordenadores obliga a investigadores y profesionales a desarrollar medidas para su detección.

Existen ya sistemas de detección de intrusiones que buscan detectar anomalías que presenten un riesgo potencial a la red. Se basan en analizar todos los paquetes entrantes y salientes de una red y buscar en ellos patrones sospechosos. El problema con estos sistemas ya existentes es que para su correcto funcionamiento hay que actualizarlos constantemente. Por otro lado, las técnicas de reconocimiento de patrones y aprendizaje automático pueden solventar algunos de los problemas que se encuentran en la detección de intrusiones. Este tipo de técnicas ha tenido ya una extensa aplicación en detección de patrones en señales tales como las señales de voz y audio, imágenes, etc., donde actualmente consiguen resultados muy satisfactorios para múltiples aplicaciones [2]. En este sentido en el grupo ATVS dispone de una larga experiencia en el empleo de técnicas de reconocimiento de patrones en señales de voz y audio, y una de las motivaciones del TFG es aplicar en la medida de los posible esa experiencia a un nuevo ámbito, tal y como se ha hecho recientemente al aplicar técnicas de reconocimiento de patrones empleadas en reconocimiento del locutor en ámbitos diferentes como el de procesado de series financieras [2,3].

1.2 Objetivos

Partiendo de este escenario, nuestro objetivo en este proyecto será el de intentar mejorar la detección de intrusiones aplicando técnicas de reconocimiento de patrones y, en la medida de lo posible, técnicas inspiradas en el procesamiento de señales temporales como la voz y el audio. El objetivo final es que al aplicar estas técnicas consigamos aumentar el porcentaje de detecciones correctas y reducir el porcentaje de falsos negativos que proporcionan los sistemas actuales e introducir

mayor flexibilidad en estos sistemas. Aunque estas técnicas resultan suficientemente flexibles para aplicarlas en todo tipo de redes, nos centraremos sobre todo en redes inalámbricas, principalmente por la disponibilidad de bases de datos sobre las que trabajar.

Para llevar a cabo este proyecto, se ha trabajado con una base de datos generada por investigadores de la Universidad de Aegan (Grecia) en el 2015 [4].

Como objetivo inicial tenemos el replicar los resultados proporcionados por este grupo, que también proponía la utilización de técnicas de aprendizaje automático. Para ello en primer lugar nos planteamos entender cada campo de las tramas proporcionadas y realizar una organización y limpieza de la base de datos. Nuestro siguiente objetivo ha sido el de mejorar estos resultados.

Para llevar a cabo estos dos objetivos ha sido necesario alcanzar estos otros sub-objetivos:

- Estudiar la base de datos.
- Analizar los atributos de las tramas.
- Realizar una primera selección de atributos.
- Emplear técnicas de selección de características para reducir más el número de atributos.
- Añadir contexto a cada trama de la base de datos

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Capítulo 1: Introducción**
  Descripción resumida de las motivaciones para llevar a cabo este proyecto, así como los objetivos que se desean conseguir con el fin de obtener un buen desarrollo del trabajo de fin de grado.
- **Capítulo 2: Estado del arte**
  En este apartado hay una introducción a los sistemas de detección de intrusiones actuales, seguido de un resumen de los ataques más comunes de la capa de enlace y finalmente una explicación sobre el funcionamiento de los algoritmos de aprendizaje automático y los distintos algoritmos utilizados.

- **Capítulo 3: Diseño**

  En este apartado se detalla la base de datos que utilizamos y se realiza un breve análisis de los experimentos hechos anteriormente con dicha base de datos. Por otro lado, también se explica el diseño del proyecto y los avances que se van a realizar.
- **Capítulo 4: Desarrollo**

  Se describe el proceso y los pasos hasta los avances finales. Se explica cada etapa del proyecto y los motivos que nos han llevado a tomar una u otra decisión. Inicialmente se habla de los métodos seguidos para realizar el análisis de la base de datos y la eliminación de campos. Por último, se presentan las técnicas utilizadas para obtener los resultados que presentamos en el siguiente capítulo.
- **Capítulo 5: Pruebas y Resultados**

  Se aportan datos, gráficas y tablas sobre las pruebas llevadas a cabo.
- **Capítulo 6: Conclusiones y trabajo futuro**

  Recoge las conclusiones más importantes de los resultados obtenidos. En este capítulo también se explican posibles formas de extender el trabajo realizado

2 Estado del arte

El poder garantizar la seguridad en redes inalámbricas resulta de gran importancia en la actualidad, por este motivo investigadores de todo el mundo se dedican a desarrollar sistemas que ayuden a evitar y detectar a tiempo intrusiones en este tipo de redes.

2.1 Sistemas de detección de intrusiones

El propósito de este tipo de sistemas es monitorear el tráfico de una red y decidir que eventos se corresponden con un ataque y cuáles no. Se consideran de gran importancia ya que las técnicas tradicionales como los cortafuegos y sistemas de encriptación no han resultado suficientes.

Principalmente existen dos tipos de sistemas de detección de intrusiones (Intrusion Detection Systems o IDS): basados en patrones (signature-based system) y basados en anomalías (anomaly based systems). El enfoque de los sistemas basados en anomalías es que el tráfico durante un ataque es distinto al del tráfico normal, por ello construye un modelo del correcto funcionamiento de la red. De esta manera detectan anomalías en la red cuando el tráfico se comporta de modo distinto al del modelo creado. Para este tipo de sistemas lo más complicado es identificar los límites entre trafico normal y anormal.

Por otro lado, los sistemas basados en patrones analizan los paquetes y los comparan con los patrones de ataques conocidos y pre-configurados. Con esta configuración los ataques bien conocidos pueden ser detectados con un porcentaje de falso positivo relativamente bajo, por ello es el sistema más extendido. El problema de este tipo de sistemas es que los ataques no suelen tener un patrón fijo, están constantemente evolucionando y creándose nuevos. Para solucionar estos problemas es necesario actualizar constantemente la base de datos de patrones, lo que resulta poco efectivo y tedioso, o aplicar algoritmos de aprendizaje automático al sistema [5]. En este trabajo nos centraremos en aplicar la segunda técnica de mejora de este tipo de sistemas.

2.2 Ataques comunes de la capa de enlace

Clasificaremos los ataques de acuerdo a su propósito en cuatro grupos distintos. Debido a que nos estamos centrando en las vulnerabilidades del protocolo 802.11 empleado habitualmente en las redes Wi-Fi, esta clasificación se corresponde a ataques en la capa de enlace.

El primer paso para el atacante es encontrar un objetivo. Esto se puede hacer por un sondeo activo o pasivo. En el caso de sondeo activo el atacante envía solicitudes a los puntos de acceso seleccionados con el propósito de obtener su información SSID. Por el contrario, el sondeo pasivo es cuando el atacante escucha todas las conversaciones de una red sin enviar ninguna petición a los puntos de acceso. En el caso de que el tráfico no esté encriptado, la información puede ser obtenida sin ningún esfuerzo extra, si no es el caso se tendrán que aplicar técnicas de desencriptación.

2.2.1 Obtención de clave secreta

Si el tráfico ha sido encriptado se procede a un proceso de obtención de clave secreta. El atacante tiene que monitorear y almacenar varios paquetes, una vez hecho esto el proceso puede seguir sin estar conectado a internet. Para hallar la clave se pueden utilizar distintos paquetes de software tal y como Aircrack o CloudCracker. Cuando la clave haya sido descubierta toda la información estará disponible para el atacante. Hasta este punto, el hacker que escucha el tráfico es ilocalizable, pero normalmente este ataque viene acompañado de una gran cantidad de inyección de paquetes, haciendo al intruso perceptible [4]. La mayoría de estos ataques se basan en la generación y obtención de los vectores de inicialización para que más adelante el intruso pueda aprovecharse de las debilidades del protocolo.

2.2.2 Obtención de *keystreams*

Hay dos posibles formas de realizar este ataque. La primera es falsificando e inyectando paquetes en la red. Esto es posible ya que el estándar 802.11 permite que el emisor del mensaje elija los vectores de inicialización y además permite reutilizarlos. Otra forma de proceder con el ataque es descifrando porciones específicas de los paquetes. De estas dos maneras el atacante puede aprender sobre la topología de la red para en el futuro proceder a ataques más graves o directamente puede construir una base de datos con las parejas de flujos de claves y vectores de inicialización y así ser capaz de descifrando el tráfico [6].

2.2.3 Denegación de servicio

El propósito de este ataque es saturar la red e impedir que sus usuarios puedan hacer uso de ella de manera normal. Para llevar a cabo este ataque el intruso tiene que estar dentro del área de cobertura de dicha red. En este tipo de ataques, se suelen utilizar paquetes de des-autentificación o desconexión para forzar a los usuarios a abandonar la red. En estos casos la dirección destino será la de difusión y así todos los usuarios conectados a la red serán afectados. Resulta relativamente fácil realizar este ataque ya que este tipo de paquetes no son encriptados.

Otra forma de realizar este ataque es desbordando los recursos del punto de acceso. Un intruso puede hacer esto enviando continuamente paquetes de autenticación, y una vez la tabla de direcciones esté saturada no será capaz de conectar a otro cliente.

2.2.4 Ataque por interposición

En este tipo de ataques el objetivo es que el intruso esté en la mitad de la conversación y pueda "escuchar" todo el tráfico. Normalmente el primer paso de este ataque es desbordar al punto de acceso con paquetes de des-autentificación para que el usuario, objetivo del ataque, no pueda conectarse al punto de acceso real la siguiente vez. Una vez el usuario ha sido forzado fuera de la red, el intruso simula ser el punto de acceso con las mismas credenciales, de tal manera que la siguiente vez que el usuario intenta conectarse lo hará al punto de acceso falso creado por el intruso. Tras esto el atacante tendrá acceso a toda la conversación.

2.3 Aprendizaje Automático: Reconocimiento de Patrones

El aprendizaje automático es un conjunto de métodos y procedimientos de computación automática que permiten detectar patrones en distintos datos de entrenamiento para más tarde utilizarlos para predecir sobre nuevos datos o tomar distintas decisiones con ellos [7]. Estos sistemas son de gran utilidad cuando las tareas son complicadas de programar y se requiere de flexibilidad [8].

Las técnicas de aprendizaje automático se dividen en dos grandes grupos, pueden ser supervisadas o no. Cuando hablamos de aprendizaje no supervisado es cuando durante el entrenamiento no le proporcionamos al sistema los resultados correctos para cada entrada. En estos casos se utiliza el aprendizaje automático para agrupar los datos según sus propiedades solamente. Por el contrario, en los sistemas de aprendizaje supervisado durante el entrenamiento sí se proporciona los resultados deseables para cada entrada.

Por otro lado, dependiendo del tipo de problema que se presente o los datos disponibles hay una gran variedad de algoritmos que se pueden utilizar y funcionaran mejor o peor dependiendo de cada caso. A continuación, mencionaremos algunos ejemplos de algoritmos de aprendizaje supervisado.

2.3.1 Naïve Bayes

El clasificador Naïve Bayes se fundamenta en el teorema de Bayes y funciona relativamente bien cuando la dimensión de las entradas es alta. Este clasificador asume que todos los atributos son independientes dada una clase. Normalmente esta suposición es falsa, pero el modelo resultante suele encajar correctamente y funciona sorprendentemente bien para ser tan simple [9].

$$p(y \mid x) = \frac{p(x,y)}{p(x)} = \frac{p(x \mid y)p(y)}{\sum_{y'=1}^{C} p(x \mid y')p(y')}$$

En este ejemplo mostrado en la Ilustración 1 los objetos pueden ser clasificados como verdes o rojos y la tarea del algoritmo será la de clasificar en uno de estos dos grupos nuevos objetos. Debido a que hay el mayor cantidad de objetos verdes que el de rojos, hay mayor probabilidad de que un objeto nuevo sea clasificado como verde. Esta observación se conoce como la probabilidad a priori, que son aquellas basadas en experiencia previa. Para proceder con la clasificación se toma un conjunto de muestras alrededor del nuevo objeto y se calcula la probabilidad de que pertenezca a una u otra clase haciendo uso tanto de la probabilidad a priori como de la probabilidad de pertenecer a una clase dado un vecindario [10].

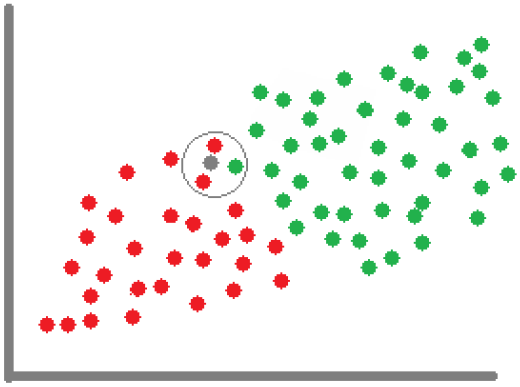

Ilustración 1. Ejemplo Naïve Bayes

Probabilidad de que el nuevo objeto sea verde: *Probabilidad a priori que sea verde* x *Probabilidad de que vea verde dentro de ese vecindario*
Probabilidad de que el nuevo objeto sea rojo: *Probabilidad a priori que sea rojo* x *Probabilidad de que vea rojo dentro de ese vecindario*

2.3.2 Hyperpipes

Hyperpipes es un algoritmo de clasificación muy simple, que es rápido y trabaja bien con grandes cantidades de atributos. La idea básica es crear una única "tubería" para cada clase de la base de datos y durante el entrenamiento la tubería de cada clase realiza un seguimiento de los valores de los atributos que se ha encontrado sin llevar una cuenta de cuantas veces se los ha encontrado. El hecho de que el algoritmo no lleva la cuenta de las veces que se encuentran los atributos tiene la gran desventaja que valores de atributos que han aparecido mil veces tendrán la misma importancia que aquellos que han aparecido solo una vez, por otro lado, esta característica del algoritmo es lo que permite que funcione a gran velocidad. Durante la etapa de test, cada vector de datos será clasificado en una tubería que más se asemeje a los valores del vector de entrada. En la práctica se ha

observado que muchas veces la base de datos de test completa es asignada a una sola clase, la que tenga los atributos más variados [11].

2.3.3 AdaBoost

Adaboost es la abreviación de "Adaptative Boosting" y fue el primer algoritmo basado en la técnica del "boosting" que es básicamente crear un clasificador de alta precisión combinando varios clasificadores relativamente inexactos. Para ello se asume que cada uno de estos clasificadores tendrá una precisión mayor que una clasificación al azar. Esto es lo que se conoce como condición de aprendizaje débil y es el fundamento del algoritmo. A continuación, se explica la idea del algoritmo [12].

- Entrada: Datos de entrenamiento $Z = \{(X_n, Y_n)\}_{n=1}^{N}$
- Para cada t=1,2 …, T,
    - Aprender una regla sencilla h_t de los datos de entrenamiento
    - Hallar la probabilidad de error con dicha regla
    - Enfatizar en los datos de entrenamiento que no sigan dicha regla
- Salida: Combinar la función $H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$ [13]

2.3.4 One Rule

El propósito del algoritmo más comúnmente conocido como OneR, es encontrar una regla que predice la clase de una entrada a base de sus atributos. La regla se establecerá con un solo atributo, el más efectivo y discriminante. El algoritmo asume que los atributos son discretos y en el caso de que no lo sean deben ser discretizados. Por otro lado, los valores desconocidos son tratados a parte como otro posible valor del atributo, de tal manera que el hecho de que un atributo esté o no presente supone de información útil para la predicción. A continuación, se explica la idea del algoritmo [14].

- Para cada atributo ***a*** generamos una regla
    - Para cada valor posible de ***v***:
        - Seleccionamos el conjunto de ejemplos donde el atributo ***a*** tiene el valor de ***v.***
        - Determinamos c, que será la clase más frecuente dentro del conjunto de datos
        - Si ***a*** tiene el valor de ***v*** y la clase es ***c***:
            - Se calcula la precisión de la clasificación con esta regla
- Finalmente se utiliza la regla con la precisión de clasificación más alta.

2.3.5 Zero Rule

El algoritmo ZeroR es el más simple de todos ya que basa su decisión solo en las probabilidades a priori de las clases y no en los atributos. Sencillamente clasifica

todas las entradas como la clase más común de la base de entrenamiento. Este tipo de sistemas son útiles para determinar una base del rendimiento del sistema y así poder compararlo con otros sistemas de clasificación. El algoritmo construye una tabla de frecuencia de las clases posibles y selecciona la clase más frecuente [15].

2.3.6 Árboles de decisión

El método de aprendizaje basado en árboles de decisión es de los más utilizados en algoritmos de inferencia inductiva y son aplicados en una gran variedad de campos. Los árboles de decisión clasifican cada entrada ordenándolos desde la raíz hasta una de sus hojas, la cual sería la clasificación de dicha entrada. Cada nodo del árbol especifica una regla para un atributo de la entrada y cada rama se corresponde con uno de los posibles valores de ese atributo. Para clasificar una entrada, se empieza desde la raíz del árbol. En ese punto se clasifica según el atributo específico de ese nodo, y se sigue por la rama por la cual la entrada cumple la condición o regla [16]. Una de sus principales ventajas de los árboles de decisión es que pueden trabajar con mezcla de todo tipo de datos (numéricos, literales, binarios, textuales, etc.), mientras que muchos otros algoritmos de aprendizaje automático están restringidos a valores numéricos.

Aquí vemos un ejemplo en el que el objetivo es saber si se puede o no jugar al tenis dependiendo de las condiciones atmosféricas. Para tomar una decisión es necesario fijarse en tres características: pronóstico, humedad y ventosidad. Cada una de estas características puede tomar distintos valores y dependiendo de cuales sean, se podrá jugar o no. Los datos mostrados en la Tabla 1 son las posibles combinaciones y sus resultados correspondientes, con esa información se creará un árbol de decisión. El nodo raíz se corresponde con uno de los posibles atributos, y puede variar dependiendo del tipo de árbol de decisión que se utilice, en este caso se ha elegido el pronóstico. En la Ilustración 2 se muestra el árbol de decisión resultante, de tal manera que para clasificar una nueva entrada habrá que seguir el árbol desde su raíz hasta una de sus hojas.

Pronóstico	Humedad	Ventosidad	Jugar al Golf
Lluvioso	Alta	No	No
Lluvioso	Alta	Si	No
Nublado	Alta	No	Si
Soleado	Alta	No	Si
Soleado	Media	No	Si
Soleado	Media	Si	No
Nublado	Media	Si	Si
Lluvioso	Alta	No	No
Lluvioso	Media	Si	Si
Soleado	Media	Si	Si
Lluvioso	Media	Si	Si
Nublado	Alta	Si	Si
Nublado	Media	Si	Si
Soleado	Alta	No	No

Tabla 1. Ejemplo de posibles combinaciones

Ilustración 2.Árbol de Decisión

2.3.6.1 Random Forest

En el algoritmo de Random Forest, el árbol de decisión que se construye es a base de varios árboles. Para clasificar una nueva entrada, el vector de entrada pasa por cada árbol de decisión del *bosque*. Cada árbol da su clasificación y "vota" por una clase. Luego el *bosque* en conjunto elige la clasificación que tenga más "votos". La tasa de error dependerá de dos factores: la correlación entre los árboles del bosque y la precisión de cada árbol individualmente [17].

2.3.6.2 Random Tree

En el algoritmo de Random Tree se construye el árbol de decisión de manera aleatoria. A medida que se va construyendo el árbol el algoritmo elige un atributo

de los que aún no ha utilizado de manera aleatoria, sin tomar en cuenta ningún tipo de condición. Un atributo categórico como por ejemplo el género solo se elige una vez para construir el árbol, por el contrario, un atributo continuo se puede elegir varias veces y cada vez que se determina un umbral aleatorio. El árbol deja de crecer si una de las siguientes condiciones se alcanza [18]:

- El tamaño de árbol va a exceder unos límites.
- Ningún nodo puede dividirse en más nodos.

2.3.6.3 J48

Este tipo de algoritmo es de los mejor valorados en el ámbito del aprendizaje automático [19]. En este, dado un conjunto de S casos, el árbol J48 empieza a crecer utilizando el algoritmo divide y vencerás.

- Si todos los casos del conjunto S pertenecen a la misma clase o es un conjunto pequeño, la hoja del árbol será etiquetada con la clase más frecuente.
- Cuando el conjunto S contiene casos que pertenecen a varias clases, el objetivo será refinar el conjunto S en subconjuntos en los que eventualmente todos los casos serán de una clase. Se realiza un test basado en un solo atributo que tiene varios resultados excluyentes. De esta manera el conjunto S se divide en los subconjuntos *S1*, *S2*... correspondientes. Este proceso se realiza recursivamente hasta que en cada subconjunto se dé el primer caso.

La hipótesis o umbral que se realiza para tomar las decisiones e ir construyendo el árbol puede variar. En este caso, el J48 utiliza una técnica llamada proporción de ganancias (Gain Ratio). Esta es una medida de información que considera diferentes probabilidades de los resultados de cada test [20].

$$I(P) = -\sum_{i=1}^{k} p_i \times \log(p_i) \qquad Info(T) = I(P)$$

$$Info(X,T) = \sum_{i}^{n} \frac{|T_i|}{|T|} Info(T_i) \qquad Gain(X,T) = Info(T) - Info(X,T)$$

$$SplitInfo(X,T) = -\sum_{i}^{n} \frac{|T_i|}{|T|} \log_2 \frac{|T_i|}{|T|} \qquad GainRatio(X,T) = \frac{Gain(X,T)}{SplitInfo(X,T)}$$

3 Diseño

La investigación y los experimentos que se recogen en este trabajo han sido realizados siguiendo el trabajo hecho por C. Kolias et al. [4], y en concreto sobre la base de datos que hicieron pública. Junto con la base de datos proporcionaron resultados utilizando algoritmos de aprendizaje automático para clasificar tramas y poder detectar atraques en redes inalámbricas.

3.1 Base de Datos

3.1.1 Descripción de la base de datos

El conjunto de bases de datos está dividido en dos grupos según cómo han sido etiquetadas las tramas, por un lado, hay cuatro bases de datos clasificadas según el propósito del ataque y por otro lado hay otras cuatro bases de datos clasificadas según el nombre del ataque en concreto. Para este trabajo se ha utilizado dos de las ocho bases de datos que el equipo de C. Kolias hizo pública, las correspondientes al grupo clasificadas según el propósito del ataque.

Dentro de ese grupo, dos de los conjuntos de datos son de entrenamiento. El más grande contiene 96 horas de tráfico y el más pequeño contiene solo uno hora de tráfico. Cada uno de estos conjuntos de datos tiene su correspondiente base de datos de test. Para la base de datos grande el conjunto de test es de 12 horas de tráfico y para el pequeño de 20 minutos.

Inicialmente se decidió realizar las primeras pruebas con la base de datos reducida y una vez se encontrase un método para mejorar el sistema de detección de intrusiones, se exploraría la posibilidad de ampliar la investigación a la base de datos más grande.

Por este motivo empezamos las pruebas con la base de datos AWID-ATK-R-Trn y AWID-ATK-R-Tst, las cuales clasifican las tramas como normales o en tres tipos de ataques (inundación, inyección o interposición). El conjunto de datos de entrenamiento contiene la información de 1.795.575 tramas de las cuales 162.385 pertenecen a ataques. A su vez, el conjunto de test contiene información de 575.643 tramas de las cuales 44.858 se corresponden con ataques.

Cada paquete es representado por 155 atributos, de los cuales la mayoría pertenecen a la capa de MAC, pero también hay información de Radiotap que contiene datos relevantes a la inyección y recepción de tramas en el estándar 802.11.

3.1.2 Captura de la base de datos

Como se ha mencionado anteriormente, la base de datos fue creada por un grupo de investigadores de la Universidad de Aegan, Grecia. Su propósito era el de simular una red SOHO (Small office/ Home Office) típica, por ese motivo en el laboratorio desde donde se realizaron las pruebas se colocaron 8 dispositivos móviles y 2 dispositivos fijos. En todos estos dispositivos se realizaban tareas básicas y comunes tal y como ver vídeos en directo, navegar por internet o enviar y recibir archivos.

El laboratorio tenía cuatro habitaciones y solo un punto de acceso el cual proveía de protección con encriptación WEP. Además, fuera del laboratorio se encontrada el atacante que generó todo el tráfico malicioso. Durante los ataques el intruso cambiaba constantemente su dirección MAC e hizo uso de distintos softwares para descifrando partes de tramas o las propias claves secretas.

Para capturar el tráfico se utilizó otro ordenador el cual también se hallaba dentro del laboratorio y la herramienta TShark que es la versión de WireShark para la terminal.

3.1.3 Experimentos anteriores

El grupo de investigadores de la universidad de Aegan realizaron pruebas con una gran variedad de algoritmos con el objetivo de encontrar el que mejor funcionaba con la base de datos AWID. Para el entrenamiento de los algoritmos utilizaron la base de datos AWID-CLS-R-Trn y para las pruebas utilizaron AWID-CLS-R-Tst. Las pruebas las realizaron en un ordenador de 8 núcleos, en una máquina virtual de 56 GB de RAM, localizada en el servicio en la nube de Azure.

En sus pruebas observaron que el algoritmo J48 fue el que mejor funcionó y obtuvo mejores tasas de *"true positives"* (TP), que son el porcentaje de elementos etiquetados correctamente y pertenecientes a la clase positiva, y *"false positives"* (FP), que hacen referencia al porcentaje de elementos clasificados incorrectamente y pertenecientes a la clase negativa, pero fue el que más tiempo de cálculo requirió. Los siguientes algoritmos que mejores resultados obtuvieron fueron Random Forest y OneR.

Predicción / ***Modelo*** / ***Etiqueta Real***	***Clase Positiva***	***Clase Negativa***
Clase Positiva	True Positive (TP)	False Negative (FN)
Clase Negativa	False Positive (FP)	True Negative (TN)

Tabla 2.Tasas de Error en Detección

Al contrario, los algoritmos ZeroR, AdaBoost y Hyperpipes clasificaban incorrectamente todas las tramas de ataque, esto se puede observar mejor en las matrices de confusión de cada algoritmo. A primera vista parece que no funcionan tan mal debido a que su tasa de TP es de 0.922, pero esto se debe al pequeño porcentaje de tramas de ataques que hay en la base de datos, con lo que simplemente clasificando todas las tramas como normales se consigue este resultado. Por otro lado, independientemente del algoritmo utilizado el ataque de interposición fue el más complicado de detectar. El algoritmo de Random Tree fue el que mejor pudo clasificar las tramas de intrusión de ataques de interposición con un porcentaje de acierto de sólo el 7.5% [4]. En el capítulo de 5 de resultados se presentas los resultados que C. Kolias y su equipo obtuvieron y un análisis sobre los mismos.

3.2 Planteamiento y experimentos del proyecto

Como se ha mencionado en el capítulo 1, el objetivo inicial es replicar los resultados de C. Kolias y su equipo. Para ello, el primer paso es entender la base de datos y sus atributos. Por lo que se va a realizar un análisis de cada atributo, como su significado, los valores posibles que toman y su importancia dentro de la base de datos. Al ser bases de datos grandes se decidió que la mejor forma de trabajar con estas sería utilizando el sistema operativo Linux y manipularlas a través de la terminal.

Una vez hecho esto, el siguiente paso fue realizar un primer filtrado de los atributos. Así eliminando todos aquellos que apenas aportan información a los algoritmos de aprendizaje automático y podrían incluso contribuir a tasas de error más altas.

Al igual que el trabajo realizado en [4], en este proyecto también se utilizará la herramienta Weka que es una colección de algoritmos de aprendizaje automático. Este software se puede utilizar a través de su interfaz gráfica o sus funciones se pueden llamar desde código escrito en java y contiene herramientas para el pre-procesado, clasificación, *clustering* y visualización entre otras [21]. Por este motivo, para el siguiente paso es necesario que la base de datos este en el formato correspondiente.

Tras las pruebas con los mismos algoritmos y la eliminación de atributos que no aportan ninguna información se esperará tener unos resultados similares a los presentados en [4]. Llegado a este punto se habrá alcanzado el objetivo inicial.

Para continuar, y con el objetivo de seguir mejorando los resultados obtenidos se procederá a hacer un análisis más exhaustivo de los atributos. Para ello se utilizarán las herramientas de Weka y se harán las pruebas necesarias hasta obtener la selección de atributos que mejor funcione. El propósito es reducir más del 80% el número de atributos sin que afecte significativamente a los resultados en detección de intrusiones.

Hasta este punto se habrá trabajado siempre trama a trama. De tal manera que los algoritmos utilizados van a decidir sobre una trama basándose solo en la información esa trama.

Se observó que muchos de los ataques se basan en el envío de muchas tramas seguidas de unas características concretas, por ese motivo pensamos que tener información del contexto de una trama puede resultar muy beneficioso a la hora de decidir si es intrusiva o no. Por ello, el siguiente paso de este proyecto es realizar una ampliación de contexto en cada trama. Este paso se realizará con un script escrito en Python.

Tras cada uno de estos pasos, para cuyo entrenamiento siempre se utiliza la base de datos de AWID-CLS-R-Trn, se realizará una evaluación sobre la base de datos de AWID-CLS-R-Tst para evaluar la mejora en la detección de intrusiones.

4 Desarrollo

4.1 Entorno de trabajo

Para la realización de este proyecto se ha utilizado un ordenador de 1.9 GB de memoria y procesador Dual-Core CPU E6700 3.2 GHz x 2 y un servidor con 48 GB de memoria y 24 núcleos. Ambos con sistema operativo Ubuntu 12.04. Inicialmente el trabajo se empezó en el propio ordenador, pero debido a su poca memoria y a la gran cantidad de datos que era necesario manejar resultó considerablemente lento. Por esto fue necesario pasar las bases de datos al servidor y continuar el trabajo ahí. Otra de las razones por las que fue necesario utilizar el servidor es porque la herramienta Weka necesita bastante memoria para funcionar ya que se tiene que cargar la base de datos de entrenamiento, de test y aparte la información complementaria de cada algoritmo.

Como se ha mencionado anteriormente se utilizó Weka probar los distintos algoritmos y fue necesario descargar e instalar el Weka tanto en el ordenador como en el servidor. Al momento de realizar este trabajo la última versión de Weka era weka-3-6-14.

4.2 Herramientas empleadas para la manipulación de la base de datos

Para la manipulación de la base de datos se han utilizado diversas herramientas software, cada una según su utilidad y necesidad a lo largo del proyecto. Las herramientas y lenguajes que se han utilizado han sido *awk*, *shell scripting (bash)* y *Python*. Este último fue utilizado solo para la realización de la ampliación de contexto.

Awk es un comando de Linux diseñado exclusivamente para el procesamiento de datos, especialmente archivos estructurados y patrones de texto. Se trata de un comando que admite programas que pueden resultar bastante complejos y que incluyen las estructuras de programación habituales en todos los lenguajes de programación. Resulta de gran utilidad para modificar, buscar o generar informes de bases de datos, todas acciones necesarias para trabajar con la base de datos. Una de las particularidades más interesantes de esta herramienta es que dispone de comandos para descomponer líneas de entrada en campos [22]. Para este proyecto eso resulta de gran utilidad ya que en la base de datos que utilizamos cada línea contiene la información correspondiente a una trama y cada atributo está separado por comas.

La *shell* de Unix es el intérprete de comandos a la vez que permite realizar programas complejos mediante comandos de la Shell y las estructuras de control

habituales de los lenguajes de programación. Como intérprete de comandos sirve como interfaz para el usuario, permitiendo un fácil uso de las utilidades de Linux, y como lenguaje de programación permite la combinación de las utilidades disponibles de Linux. Los comandos *Shell* pueden ser utilizados de manera interactiva, si el usuario teclea la entrada, o no interactiva. También provee de algunos comandos más avanzados como por ejemplo cd, break o pwd. Como el resto de lenguajes de programación de alto nivel *Shell* tiene variables, funciones, control de proceso etc. [23]. En nuestro caso particular la Shell empleada ha sido bash.

En concreto el comando *sed* ha sido de gran utilidad para este proyecto, este es un editor de flujos y de ficheros de texto de forma no interactiva. Permite modificar el contenido de las líneas de un fichero a base de comandos o un fichero de comandos.

El siguiente y último lenguaje utilizado es Python. Es un lenguaje de alto nivel orientado a objetos con semántica dinámica. Tiene una sintaxis fácil y simple lo que ha permitido que tenga un gran éxito. Otra de sus características es que los scripts de Python no tienen que ser compilados explícitamente, los creadores hicieron el lenguaje así con el objetivo de hacer el proceso de ejecución de un programa más rápido [24]. Para generar el script de ampliación de contexto utilizado en este trabajo con conocimientos de *C* y una base de Python fue suficiente.

4.3 Análisis de la base de datos

Antes de aplicar los algoritmos de aprendizaje automático se decidió examinar cada atributo uno por uno con el propósito de mejorar la representación de cada atributo y revisar si todos los atributos aportaban información relevante.

Cabe recalcar que este primer análisis se realizó manualmente, generando un fichero por atributo con la información relevante de cada uno a lo largo de la base de datos. Los datos de cada uno de estos ficheros era los valores que dicho atributo tomaba y el número de veces que tomaba ese valor a lo largo de toda la base de datos de entrenamiento. Así pudimos analizar cada atributo por separado y entender mejor su posible influencia en los algoritmos de aprendizaje automático. En el Anexo I se presenta una lista de todos los atributos y una corta anotación sobre el contenido de cada uno.

4.3.1 Primera reducción de atributos

Después de analizar cada atributo de la base de datos de entrenamiento se descubrió que 73 atributos de los 155 parecían no aportar información. Estos atributos tenían un valor constante para todas o la gran mayoría de las tramas. El

criterio que se eligió para eliminar atributos fue el de descartar todos aquellos que eran constantes con la excepción de 100 tramas o menos con diferentes valores. De un total de 1.8 millones de tramas estos atributos probablemente no aportan información relevante al algoritmo de aprendizaje automático, y por ese motivo se decidió eliminarlos. Al finalizar este proceso la lista de atributos se redujo a tan solo 82 atributos, reduciendo así más de un 45% la lista original. En el Anexo II se puede encontrar una lista de los atributos tras este proceso.

4.4 Adaptación de la base de datos al formato compatible con Weka

El tipo de ficheros con los que trabaja Weka son de tipo Attribute-Relation File Format, más conocidos como ARFF. Son documentos que describen una lista de datos que comparten una serie de atributos. Estos archivos tienen dos secciones bien distinguidas. La primera es la cabecera que contiene el nombre de la relación, la lista de atributos y sus tipos. La segunda sección contiene los datos [25].

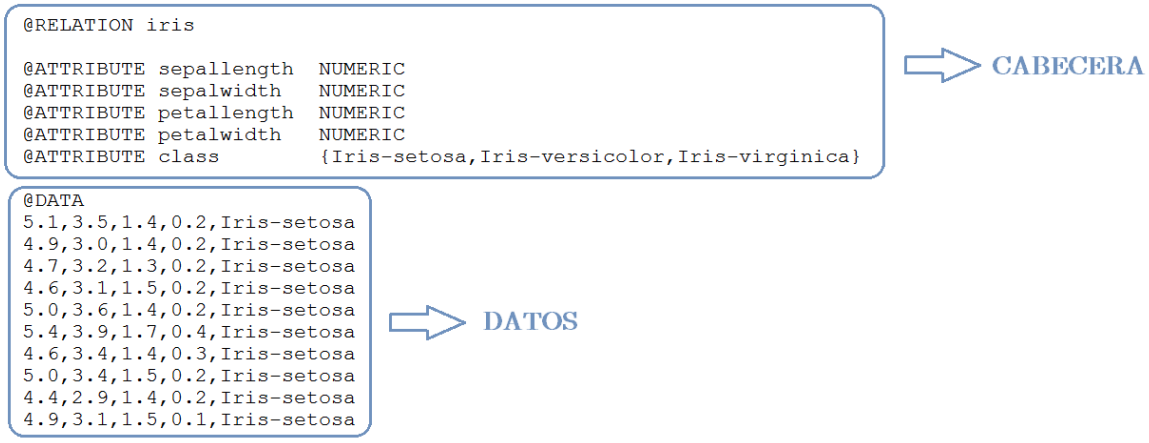

Ilustración 3. Ejemplo de fichero ARFF

Los datos pueden ser de cuatro tipos distintos: numéricos, nominales, cadenas de texto o fechas. Excepto los datos de tipo nominales, el resto no diferencian entre letras mayúsculas o minúsculas.

Para este proyecto en concreto los 82 atributos seleccionados en el primer paso se modelaron de tipo numérico, excepto las direcciones MAC del receptor final de la trama (wlan.da), fuente original de la trama (wlan.sa), receptor intermedio de la trama (wlan.ra), emisor intermedio de la trama (wlan.ta), el identificador BSS (wlan.bssid) y las etiquetas del tipo de ataque.

Las etiquetas del ataque son de tipo nominal y concuerdan con el formato exigido por Weka por lo que no será necesario realizar ningún tipo de trasformación al atributo de la clase.

Por otro lado, la dirección MAC es un identificador de 48 bits el cual es representado en el siguiente formato:

XX:XX:XX:XX:XX:XX

Como se ha explicado anteriormente este formato no encaja con ninguno de los cuatro tipos de atributos, por eso es necesario transformar esos valores a enteros.

Una vez hecho esto la base de datos finalmente tiene 82 atributos, todos numéricos, que aportan información (en mayor o menor medida, como veremos más adelante) y podrá ser utilizada por la herramienta Weka.

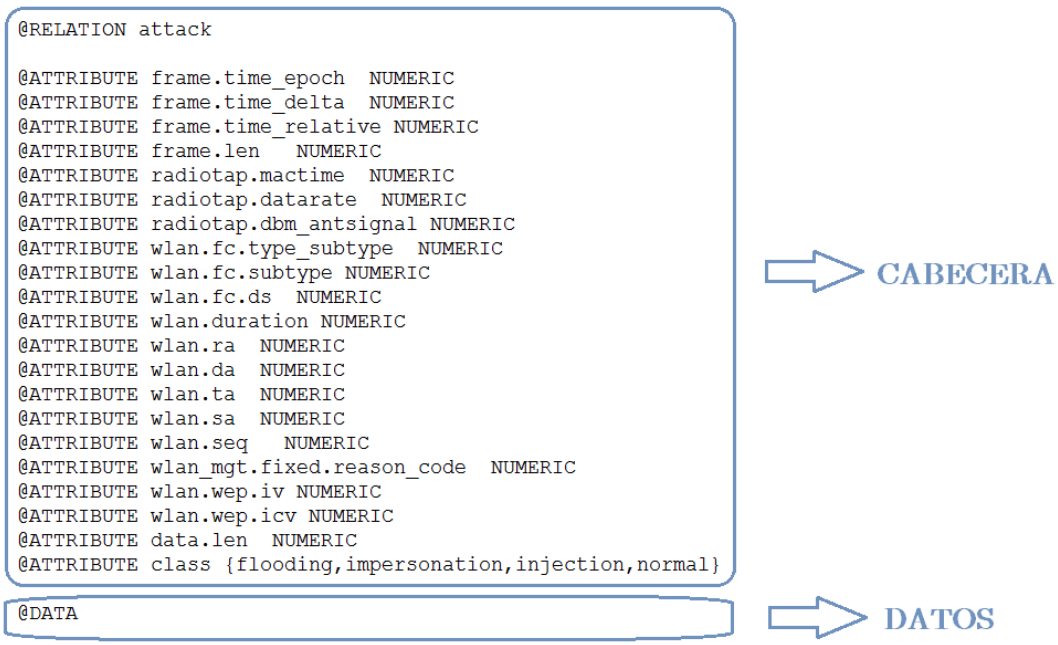

Ilustración 4. Ejemplo de cabecera de la base de datos

4.5 Selección de atributos

A pesar de haber reducido la base de datos en más de un 45% con la reducción del número de atributos se observó que era posible reducir aún más sin afectar de manera considerable los resultados. De hecho, con una selección de atributos adecuada se podría hasta mejorar los resultados. Para realizar esta reducción de atributos utilizamos las herramientas de selección de atributos proporcionadas por Weka.

Tras realizar varias pruebas, obtuvimos los mejores resultados utilizando el evaluador de atributos "*InfoGainAttributeEval*", que evalúa el valor de un atributo en función de su ganancia con respecto a la clase, y el método de búsqueda "*Ranker*" que clasifica los atributos según su evaluación individual [25]. Después de

esta evaluación de los atributos se seleccionaron 18 de los 82 atributos como los que aportaban la información más relevante.

La lista final de atributos seleccionados fue (en este orden):

1. wlan.wep.iv	2. frame.len	3. wlan.sa
4. wlan.ta	5. radiotap.mactime	6. frame.time_epoch
7. wlan.fc.type_subtype	8. wlan.fc.subtype	9. wlan.da
10. wlan.seq	11. wlan.wep.icv	12. wlan.ra
13. data.len	14. wlan.duration	15. wlan.fc.ds
16. radiotap.dbm_antisignal	17. radiotap_datarate	18. wlan_mgt.fixed.reason_code

Tabla 3. Lista de 18 atributos

Se puede observar que entre los atributos más relevantes se encuentra el emisor original de la trama (wlan.sa) o el receptor final (wlan.da). Esto nos llevó a pensar que estos dos atributos pueden tener una alta correlación con el ataque y entre ellos.

4.6 Expansión de contexto

Con la intención de continuar mejorando los resultados se decidió incluir contexto en cada trama. La hipótesis es que de esta manera el clasificador tendrá información adicional para cada trama y podrá tomar una mejor decisión. Esta hipótesis se fundamente en que varios ataques se basan en el envío continuo de tramas de un tipo concreto por ejemplo de autenticación o des-autenticación con el objetivo de saturar la red o conocer la topología de la misma. Así que con la ampliación de contexto el clasificador sabrá cuál es la trama o tramas enviadas anteriormente y cuál es la trama o tramas enviadas posteriormente a la trama que desea clasificar.

Para realizar la expansión de contexto lo primero era decidir el criterio a seguir para la expansión. Como en la base de datos se tiene información de las direcciones MAC de origen y destino y en la selección de atributos se observó que sus valores estaban muy relacionados con el ataque se decido expandir el contexto de acuerdo a la “conversación” entre dos direcciones MAC.

Tal y como se comentó en el apartado **“Obtención de la base de datos”** los datos fueron capturados por un ordenador aparte de la red y con la herramienta TShark. Por lo que la base de datos está en orden en que llegan los paquetes a la red, independientemente del destinatario o emisor, y por esto para continuar con la

expansión de contexto fue necesario ordenar las bases de datos en orden de llegada y en parejas de emisor y destino.

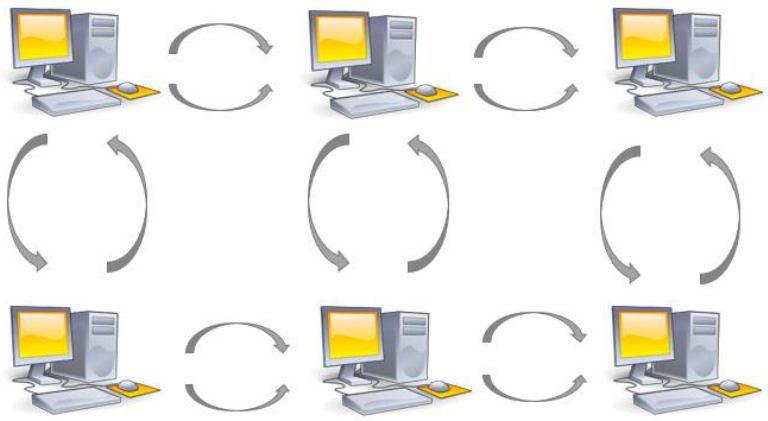

Ilustración 5. Distintas conversaciones en las que hemos dividido el tráfico de la red

Para la primera prueba de expansión de contexto de decidió incluir la información de dos tramas a cada lado. Esto quiere decir que cada nueva trama vendrá dada por las dos tramas anteriores, la trama en si a clasificar y dos tramas posteriores. Cabe recalcar que esto es posible debido a la gran reducción de atributos que se ha hecho anteriormente, de lo contrario tendría un coste computacional mucho mayor y probablemente no se podría trabajar con herramientas como Weka por restricciones de memoria.

Tramas	Ataque
Trama-1	Normal
Trama-2	Ataque
Trama-3	Normal
...	...
Trama-N	Normal

Tramas			Ataque
Trama-0	**Trama-1**	Trama-2	Normal
Trama-1	**Trama-2**	Trama-3	Ataque
Trama-2	**Trama-3**	Trama-4	Normal
	...		...
Trama-(N-1)	**Trama-N**	Trama-0	Normal

Ilustración 6. Ejemplo de expansión de contexto temporal con una trama posterior y una anterior.

Para llevar a cabo la introducción de contexto en cada trama de la base de datos se utilizó un script de Python. Para crear la base de datos con tramas extendidas había varios casos a tomar en cuenta

- Las tramas iniciales ya que no tenían tramas anteriores con las que concatenar.
    - Solución propuesta: Como los algoritmos utilizados permiten la utilización de atributos indefinidos creamos tramas auxiliares con

todos los atributos con valor indefinido y con estas concatenamos la trama inicial.

- Los cambios de "conversación". Una vez la conversación de dos dispositivos que estaban intercambiando información ha terminado empieza la siguiente conversación.
    - Solución propuesta: En estos casos era importante no mezclar estas tramas en la expansión de contexto. Por lo que se implementó un mecanismo para detectar estos casos y ampliar el contexto de la última trama de la conversación con la trama auxiliar cuyos atributos están todos con el valor indefinido de Weka ('?'). De la misma manera se trata la primera trama de la siguiente conversación. Se amplía su contexto por el lado de las tramas anteriores con la trama auxiliar.
- Conversaciones compuestas por una sola trama.
    - Solución propuesta: Para estas tramas se incluía la información de contexto haciendo uso de la trama auxiliar, tanto para la expansión de las tramas posteriores como anteriores.
- Última trama de toda la base de datos.
    - Solución propuesta: Esta última trama se trata de la misma manera que cuando termina una conversación. Se añade la información de tramas posteriores haciendo uno de la trama auxiliar.

5 Pruebas y resultados

5.1 Evaluación y comparación de resultados

En este apartado se presentan los resultados obtenidos de la realización de este trabajo y se comparan con los presentados por C. Kolias et al. en [4]. En las tablas encontramos una evaluación de varios algoritmos de clasificación, con el objetivo de encontrar cuál produce mejores resultados con las distintas bases de datos. Como se verá a continuación, los algoritmos que mejor funcionaron tanto en las pruebas realizadas en [4] como en las bases de datos con listas de atributos reducidas que se construyeron a lo largo este proyecto, fueron el J48 y Random Forest.

5.1.1 Resultados obtenidos con 82 atributos

Aquí se exponen los resultados del apartado 4.3.1 de este trabajo, correspondientes la utilización de una base de datos con la primera reducción de atributos que se realizó en este proyecto. Se puede observar que el algoritmo que mejor funciona es el J48 clasificando sobre un 96% de las tramas correctamente. Un parámetro que resulta importante analizar es el área bajo la curva ROC. Para el algoritmo J48 se obtiene un valor de 0,743 que indica que la clasificación es buena, aunque no excelente, además hay que tomar en cuenta que la base de datos está desbalanceada ya que el porcentaje de tramas de ataque es mucho menor al de tramas normales. El otro algoritmo que funciona también muy bien es Random Forest y como se observa más adelante en las matrices de confusión, clasifica mejor que el J48 para los ataques de tipo inundación e interposición y las tramas normales. Para este algoritmo, obtenemos un valor de 0,988 para el área debajo de la curva ROC, indicando una clasificación correcta, aunque igual cabe recalcar que la base de datos está desbalanceada y por ese motivo se obtienen resultados aparentemente óptimos. Por ello para un análisis más profundo a continuación se presenta un análisis de las matrices de confusión de los algoritmos que mejor funcionan y en el Anexo III se presentan resultados más detallados para el resto de algoritmos.

Algoritmo	Clasificado Correctamente	Precisión	Recall	F-Measure	Área ROC
AdaBoost	91,8495	0,909	0,918	0,91	0,946
Hyperpipes	92,2167	0,882	0,922	0,886	0,95
J48	**96,0568**	**0,962**	**0,961**	**0,946**	**0,743**
Naive Bayes	83,7173	0,888	0,837	0,843	0,583
OneR	**94,5758**	**0,9**	**0,946**	**0,922**	**0,652**
Random Forest	**95,7187**	**0,959**	**0,957**	**0,943**	**0,988**
Random Tree	92,6687	0,932	0,927	0,921	0,863
ZeroR	92,2073	0,85	0,922	0,885	0,5

Tabla 4. Evaluación de varios algoritmos de clasificación con base de datos de 82 atributos

En las siguientes tablas presentamos las matrices de confusión de los algoritmos con mejores resultados para hacer un análisis más detallado de los mismos. Se observa que los algoritmos apenas tienen falsos positivos (tramas normales clasificados como ataques): 0,0147% en el caso del algoritmo J48 y 0,00018% en el caso del algoritmo Random Forest.

En el contexto de la detección de ataques, lo que nos interesa es ser capaces de detectar ataques, por ello definimos la tasa de Falsos Positivos como el porcentaje de tramas normales que se detectan como ataque (independientemente de cual sea), y la tasa de Falsos Negativos como el porcentaje de tramas de ataque que se detectan como normales. Analizando las matrices de confusión se observa que la tasa de falsos negativos es muy elevada (>90%) para los ataques de interposición (impersonation) y para los ataques de inundación (flooding) (~40%). Los únicos ataques que presentan una tasa de falsos negativos aceptable son los ataques de inyección (injection) (<0.1% en el caso del J48). El motivo por el cual estos algoritmos de aprendizaje son capaces de clasificar correctamente la mayoría de los ataques de inyección se debe a que las tramas suelen ser pequeñas y los vectores de inicialización son siempre iguales, lo que es estadísticamente poco probable en periodos de tiempo cortos. Estas características permiten se resulte relativamente sencillo detectar este tipo de ataques.

Classified →	flooding	impersonation	injection	normal
flooding	4183	0	0	3914
impersonation	0	1374	0	18705
injection	0	0	16680	2
normal	62	16	0	530707

Tabla 5. Matriz de confusión del algoritmo J48 con base de datos de 82 atributos

Classified →	flooding	impersonation	injection	normal
flooding	4977	0	0	3120
impersonation	0	1426	0	18653
injection	0	0	13811	2871
normal	0	1	0	530784

Tabla 6. Matriz de confusión del algoritmo Random Forest con base de datos de 82 atributos

5.1.2 Resultados presentados por C. Kolias et al. con 155 atributos [4]

En la siguiente tabla se presentan los resultados obtenidos por los creadores de la base de datos. Se puede ver que los resultados obtenidos por este grupo de investigadores con una lista de 155 atributos son muy similares a los obtenidos en este trabajo con 82 atributos. Los dos algoritmos que mejor clasifican la base de datos, al igual que los resultados presentados en la tabla 3, son el J48 y el Random Forest. En concreto, para estos dos algoritmos se presentan pocas diferencias significativas. Con 155 atributos el algoritmo J48 clasifica un 0.14% mejor que con tan solo 82 atributos, esta mejor clasificación también se puede observar en el valor para el área debajo de la curva ROC que aumenta un poco. Por otro lado, utilizando 155 características el algoritmo Random Forest funciona ligeramente peor que con la lista reducida de 82 atributos.

Algoritmo	Clasificado Correctamente	Precisión	Recall	F-Measure	Área ROC
AdaBoost	92.2073	0,85	0,922	0,885	0,5
Hyperpipes	92.2073	0,85	0,922	0,885	0,5
J48	**96.1982**	**0.954**	**0.962**	**0.948**	**0.759**
Naive Bayes	89.4323	0.891	0.894	0.877	0.594
OneR	**94.5758**	**0.9**	**0.946**	**0.922**	**0.652**
Random Forest	**95.5891**	**0.958**	**0.956**	**0.941**	**0.955**
Random Tree	91.4379	0.914	0.914	0.91	0.733
ZeroR	92.2073	0,85	0,922	0,885	0,5

Tabla 7. Evaluación de varios algoritmos de clasificación con 155 atributos presentados en [4]

5.1.3 Diferencia entre ambos

Con objeto de observar con mayor facilidad la diferencia en el desempeño de los algoritmos se presenta la Tabla 7. Como se puede ver a continuación el porcentaje de clasificación correcta de las tramas disminuye considerablemente con el algoritmo de Naive Bayes al utilizar la base de datos con la lista de atributos reducida a 82. También se aprecia este mismo efecto, pero en menor medida con el algoritmo AdaBoost y J48. En cambio, para el resto de algoritmos, esta reducción de atributos supone una tasa de clasificación correcta igual o mejor que la obtenida

con 155 atributos y presentada en [4]. Estos resultados muestran que con esta primera reducción de atributos se pueden obtener resultados similares y en algunos casos mejores que los presentados por los creadores de la base de datos.

Resultados / Algoritmos	155 Atributos [4]	82 Atributos	Diferencia de resultados con 155 atributos y 82 atributos
AdaBoost	92,2073	91,8495	-0,3578
Hyperpipes	92,2073	92,2167	**+0,0094**
J48	**96,1982**	**96,0568**	**-0,1414**
Naive Bayes	89,4323	83,7173	-5,715
OneR	94,5758	94,5758	0
Random Forest	95,5891	95,7187	**+0,1296**
Random Tree	91,4379	92,6687	**+1,2308**
ZeroR	92,2073	92,2073	0

Tabla 8. Diferencia entre los resultados en [4] y los resultados con 82 atributos tras una primera reducción de la base de datos

5.1.4 Resultados obtenidos con 18 atributos

A continuación, se presentan los resultados del apartado 4.5 de este proyecto correspondientes las pruebas utilizando una lista de atributos reducida aún más tras utilizar técnicas de selección de atributos automáticas y seleccionar 18 atributos. Al igual que en los resultados obtenidos con la lista de 82 atributos, el algoritmo que mejor funciona es el J48. Con esta reducción el clasificador J48 aumenta su porcentaje de tramas clasificadas correctamente un 0.15% aproximadamente y el área debajo de la curva ROC también aumenta ligeramente indicando una mejora en la clasificación. Con esta nueva lista de atributos aún más reducida el algoritmo Random Tree mejora un 3% su clasificación, superando al algoritmo Random Forest, aun así, su área bajo la curva ROC disminuye. Por otro lado, el algoritmo Random Forest empeora ligeramente su porcentaje de tramas correctamente clasificadas ya que disminuye el número de ataques de inundación e inyección clasificados correctamente. Con respecto a la clasificación de los otros algoritmos se observa que para el algoritmo OneR, ZeroR y Naive Bayes el área debajo de la curva ROC tiene valores bajos, indicando una clasificación poco adecuada.

Algoritmo	Clasificado Correctamente	Precisión	Recall	F-Measure	Área ROC
AdaBoost	91,8495	0,909	0,918	0,910	0,946
Hyperpipes	92,2167	0,882	0,922	0,886	0,950
J48	96,2029	**0,963**	**0,962**	**0,948**	**0,759**
Naive Bayes	75,3031	0,881	0,753	0,794	0,537
OneR	94,5758	0,9	0,946	0,922	0,652
Random Forest	**95,5968**	**0,958**	**0,956**	**0,941**	**0,993**
Random Tree	**95,8471**	**0,961**	**0,958**	**0,944**	**0,794**
ZeroR	92,2073	0,85	0,922	0,885	0,5

Tabla 9. Evaluación de varios algoritmos de clasificación con base de datos de 18 atributos

Al igual que en las tablas 5 y 6 a continuación se presentan las matrices de confusión para los algoritmos con los que se obtuvo mejores resultados. Al reducir aún más el número de atributos se observa que la tasa de falsos positivos es aún menor, de hecho, para el algoritmo de Random Forest no se produce ningún falso positivo. Con respecto a la tasa de falsos negativos para el ataque de inundación el algoritmo J48 funciona mejor con la lista de atributos reducida obteniendo alrededor de una tasa de falsos negativos del 40% aproximadamente. Este valor aumenta a un 50% al utilizar el algoritmo Random Forest y aún más a un 56% al utilizar el algoritmo Random Tree. En cuanto a la detección de ataques de interposición, como podemos observar mejora mínimamente para ambos algoritmos estando la tasa de falsos negativos sobre el 90%. Finalmente, la detección de ataques de inyección se mantiene igual al utilizar el algoritmo J48, falla en muy pocas al utilizar el algoritmo Random Tree y con el algoritmo Random Forest mejora un ligeramente. Como se puede observar los resultados son muy similares a los obtenidos con 82 atributos. Esto resulta muy beneficioso ya que el proceso de detección de ataques tiene un alto coste computacional y reducir el número de atributos ayuda a reducir estos costes, a la vez que simplifica el modelo, mejorando por tanto sus capacidades de generalización.

Classified →	flooding	impersonation	injection	normal
flooding	4871	0	0	3226
impersonation	0	1471	0	18608
injection	0	0	16680	2
normal	6	16	0	530763

Tabla 10. Matriz de confusión del algoritmo J48 con base de datos de 18 atributos

Classified →	flooding	impersonation	injection	normal
flooding	4038	0	0	4059
impersonation	0	1462	0	18617
injection	0	0	14011	2671
normal	0	0	0	530785

Tabla 11. Matriz de confusión de algoritmo Random Forest con base de datos de 18 atributos

Classified →	flooding	impersonation	injection	normal
flooding	3476	24	0	4597
impersonation	0	1410	4763	13906
injection	0	0	16667	15
normal	2	27	572	530184

Tabla 12. Matriz de confusión de algoritmo Random Tree con base de datos de 18 atributos

5.1.5 Resultados presentados por C Kolias et al. con 20 atributos [4]

En la siguiente tabla se presentan los resultados obtenidos por los creadores de la base de datos al reducir la lista de atributos manualmente a 20 atributos. Al igual que en las pruebas realizadas en este trabajo con la base de datos de 18 atributos, los dos algoritmos que mejor funcionan son el J48 y Random Tree. Ambos funcionan bastante bien y tienen valores para el área ROC buenos, corroborando que la clasificación de ambos algoritmos es óptima. Con esta reducción de atributos el grupo de investigadores creadores de la base de datos fueron capaces de mejorar o mantener la tasa de TP para todos los algoritmos. En el anexo III se explica y compara con más detalle estos resultados.

Algoritmo	Clasificado Correctamente	Precisión	Recall	F-Measure	Área ROC
AdaBoost	92,2073	0,85	0,922	0,885	0,5
Hyperpipes	92,2363	0,879	0,922	0,885	0,935
J48	**96,2574**	**0,962**	**0,963**	**0,948**	**0,752**
Naive Bayes	90,5504	0.917	**0,906**	0,909	0,774
OneR	94,5741	0,9	0,946	0,922	0,652
Random Forest	**95,8247**	**0,959**	**0,958**	**0,944**	**0,958**
Random Tree	**96,2258**	**0,959**	**0,962**	**0,948**	**0,762**
ZeroR	92,2073	0,85	0,922	0,885	0,5

Tabla 13. Evaluación de varios algoritmos de clasificación con base de datos de 20 atributos

5.1.6 Diferencia entre ambos

A continuación, se muestra una tabla comparando las tasas de acierto de los resultados obtenidos en este trabajo al aplicar la reducción automática a 18 atributos y los resultados obtenidos con la reducción manual a 20 atributos

realizada por C. Kolias y su equipo. Se puede observar que, para prácticamente todos los algoritmos, excepto para OneR y Naive Bayes, los resultados con 20 atributos presentados en [4] son ligeramente mejores. Para el único algoritmo para el cual se aprecia un empeoramiento significativo es para el algoritmo de Naive Bayes que empeora un 15%. Para el resto de algoritmos se obtienen resultados similares y debido a que la selección de atributos en este proyecto fue realizada automáticamente supone una ventaja sobre el trabajo realizado por los creadores de la base de datos.

Resultados / Algoritmos	20 Atributos [4]	18 Atributos	Diferencia de resultados con 20 atributos y 18 atributos
AdaBoost	92,2073	91,8495	0,3578
Hyperpipes	92,2363	92,2167	0,0196
J48	**96,2574**	**96,2029**	0,0545
Naive Bayes	90,5504	75,3031	15,2473
OneR	94,5741	94,5758	-0,0017
Random Forest	**95,8247**	**95,5968**	**0,2279**
Random Tree	**96.2258**	**95,8471**	**0,3787**
ZeroR	92,2073	92,2073	0

Tabla 14. Diferencia entre los resultados en [4] y los resultados con 18 atributos tras una segunda reducción de la base de datos

5.2 Evaluación de distintos algoritmos de clasificación con extensión de contexto

En este apartado se presentan los resultados del apartado 4.6 de este trabajo, correspondientes las pruebas utilizando una lista de atributos reducida e incluyendo contexto temporal en cada trama con la información de dos tramas posteriores y dos anteriores. Una vez más el algoritmo que mejor funciona es el J48, aunque su porcentaje de clasificación correcta de tramas disminuye ligeramente, más adelante se presenta un análisis de la clasificación realizada por este algoritmo. También, al incluir la expansión de contexto se obtienen buenos resultados para los algoritmos Random Tree y Random Forest con valores del área bajo la curva ROC muy elevados, aunque aún se mantienen un poco por debajo de los obtenidos utilizando la lista de 82 atributos, o 18 atributos. Para ver realmente la influencia de la expansión de contexto en la clasificación de las tramas es necesario analizar las matrices de confusión de los algoritmos.

Algoritmo	Clasificado Correctamente	Precisión	Recall	F-Measure	Área ROC
AdaBoost	93,1775	0,91	0,932	0,918	0,956
Hyperpipes	80,8986	0,906	0,809	0,84	0,867
J48	96.1004	**0,961**	**0,961**	**0,947**	**0,765**
Naive Bayes	69,1734	0,878	0,692	0,756	0,516
OneR	94,5758	0,9	0,946	0,922	0,652
Random Forest	**95,6609**	**0,959**	**0,957**	**0,942**	**0,993**
Random Tree	**95,6176**	**0,956**	**0,956**	**0,941**	**0,951**
ZeroR	92,2073	0,85	0,922	0,885	0,5

Tabla 15. Evaluación de distintos algoritmos de clasificación con base de datos que incluye expansión de contexto temporal con dos tramas posteriores y dos anteriores

A continuación, se muestras las matrices de confusión de los algoritmos que mejores resultados consiguieron utilizando bases de datos que incluyen la expansión de contexto. En estos experimentos se observa que la tasa de falsos positivos prácticamente se mantiene igual que en el resto de pruebas. De la misma manera la tasa de falsos negativos tanto para interposición e inyección se mantiene alrededor de los mismos valores. La gran mejora la podemos observar en la detección de ataques de tipo inundación en los que la tasa de falsos negativos pasa de estar alrededor de un 40% a un 30% aproximadamente. Esto seguramente se debe a que al realizar estos ataques se envía una gran cantidad de paquetes de un tipo en concreto al punto de acceso y al añadir contexto a cada trama el continuo envío de este tipo de paquetes el más fácilmente detectable por el algoritmo de clasificación. Probablemente si se amplía el contexto temporal aún más se podrían obtener mejores resultados en la detección de este ataque.

Classified →	flooding	impersonation	injection	normal
flooding	5508	0	0	2589
impersonation	0	1469	0	18610
injection	0	0	16680	2
normal	1225	5	0	529538

Tabla 16. Matrices de confusión del algoritmo J48 con base de datos de 18 atributos y expansión de contexto

Classified →	flooding	Impersonation	injection	normal
flooding	4876	0	0	3221
impersonation	0	1196	0	18883
injection	0	0	13809	2873
normal	1	0	0	530784

Tabla 17. Matrices de confusión del algoritmo Random Forest con base de datos de 18 atributos y expansión de contexto

Classified →	flooding	impersonation	injection	normal
flooding	4905	0	0	3177
impersonation	11	983	0	19084
injection	2	0	13838	2842
normal	40	48	7	530690

Tabla 18. Matrices de confusión del algoritmo Random Tree con base de datos de 18 atributos y expansión de contexto

5.3 Comparación entre distintas bases de datos con diferentes algoritmos

En este apartado se presenta una tabla en la que se compara el rendimiento de los tres algoritmos que mejor funcionan dependiendo de la lista de atributos que se utilice para construir la base de datos. Los mejores resultados a lo largo de todo este proyecto se obtuvieron con el algoritmo J48 y la base de datos de 18 atributos, aunque para todos los casos el porcentaje de clasificación correcta de las tramas se mantuvo por encima del 96%. Con respecto al algoritmo Random Forest, los mejores resultados se obtuvieron con la base de datos de 82 atributos, pero al igual que para el algoritmo J48, entre las bases de datos los resultados variaron levemente y el porcentaje de tramas clasificadas correctamente siempre se mantuvo por encima del 95.5%. En cuanto al algoritmo Random Tree, los mejores resultados se obtuvieron con la base de datos de 18 atributos. Para este algoritmo sí que se apreciaron mejoras considerables al pasar de una lista de atributos a otra más corta ya que inicialmente con la base de datos creada con la lista de 82 atributos se tenía un porcentaje de clasificación correcta del 92.71%, y al pasar a la base de datos de 18 atributos se mejora alrededor de un 3%.

Algoritmo	Nº de características	82	18	18+expansión
J48	Clasificadas Correctamente %	96.0568	96.2029	96.1004
Random Forest	Clasificadas Correctamente %	95.7187	95.5968	95,6609
Random Tree	Clasificadas Correctamente %	92.6687	95.8471	95.6176

Tabla 19. Comparación entre distintas bases de datos utilizando y los algoritmos que mejor clasifican

5.4 Análisis de tiempo en construir el modelo de cada algoritmo

Los algoritmos que mejor funcionan son el J48, Random Forest y Random Tree, de estos los dos primeros son los que más tardan en construir el modelo de clasificación (junto con AdaBoost). Como se puede observar en la tabla, el

algoritmo Random Forest es el más costoso de todos y con mucha diferencia respecto al resto de algoritmos. Por otro lado, el algoritmo Random Tree obtuvo muy buenos resultados para la base de datos con la lista de 18 atributos y la base de datos que incluye expansión del contexto temporal y a diferencia del J48 y Random Forest es mucho menos costo. En cualquier caso, el coste computacional de la creación del modelo no es muy importante porque es un proceso que no se tiene por qué realizar en tiempo real. El coste computacional realmente importante es el de la detección, que debería realizarse en tiempo real y que en cualquier caso es mucho menor que el de la creación del modelo.

Nº de características	**J48**	**Random Forest**	**Random Tree**	**OneR**	**Naive Bayes**	**AdaBoost**	**Hyperpipes**	**ZeroR**
82	442,95	2626,26	20,65	35,59	32,21	377,92	0,82	3,65
18	135,53	2252,89	39,92	17,68	11,75	388,68	0,83	0,31
18+expansión	456,97	3294	31.43	41.13	40.13	358,9	1	0,21

Tabla 20. Análisis de tiempo en construir el modelo de detección de intrusiones

6 Conclusiones y trabajo futuro

Los experimentos realizados en este trabajo indican que con tan solo 82 atributos podemos obtener resultados muy similares, para la mayoría de algoritmos, a los obtenidos con 155 atributos en el trabajo de investigación que servía como base para este trabajo [4]. Con estos primeros experimentos se observó que para la detección de intrusiones los otros 73 atributos apenas contribuían.

Por otro lado, fue posible la reducción de la lista de atributos a tan solo 18 atributos realmente relevantes para los algoritmos de clasificación. Debido a que esta selección de atributos se realizó de forma automática esto supone ya una ventaja sobre el trabajo realizado por C. Kolias y su equipo, ya que ellos lo hicieron de forma manual y no publicaron la lista de atributos seleccionada. Con esta selección de atributos no solo se logró obtener resultados muy similares, sino que además se redujo el coste computacional. A lo largo de la realización de este proyecto también se observó que varios de los atributos parecían estar correlacionados y que los ataques varias veces se generaban a base del envío de un gran número de tramas seguidas. Por ello se decidió realizar la expansión de contexto temporal.

Con respecto a los resultados obtenidos con la base de datos de 18 atributos y además expansión de contexto con dos tramas posteriores y dos anteriores se puede observar una gran mejora en la detección de ataques de inundación. Como se comentó en el apartado de resultados, esto se debe a que este ataque se basa en el envío continuo de grandes cantidades de paquetes específicos de tal manera que el punto de acceso se sature, y por ello incluir contexto temporal tiene gran importancia. Seguramente si se amplía aún más el contexto temporal se obtendrían aún mejores resultados. En cuanto al resto de ataques, no se observa una gran mejora. Esto posiblemente se deban a que la expansión de contexto se basa en que los paquetes utilizados para establecer una conversación sean enviados y recibidos a lo largo de toda la conexión por los mismos dispositivos que tendrán siempre la misma dirección MAC. Por el contrario, en la creación de la base de datos, durante un ataque a la red el intruso cambiaba su dirección MAC varias veces, haciendo que la inclusión de contexto a un paquete resulte complicada, al menos si las condiciones para incluir contexto se basan en las direcciones de origen y destino del paquete.

Se puede observar a lo largo del proyecto que el aprendizaje automático tiene un gran potencial en el campo de la detección de intrusiones. Por este motivo, se considera que para continuar mejorando los resultados obtenidos en este trabajo sería importante explorar con mayor profundidad la ampliación de contexto. Esto

supondría experimentar la ampliación de contexto aumentando el número de tramas utilizadas para dicho proceso o modificando la forma en que se ha segregado el tráfico de la red para realizar la expansión del contexto.

Por otro lado, puede resultar interesante combinar distintos detectores de intrusiones. A lo largo del trabajo se ha visto que unos detectores con un tipo particular de características funcionan mejor que otros para detectar los distintos tipos de tramas de ataques o normales, por lo que una adecuada combinación de los mismos podría retener las fortalezas de los distintos algoritmos.

Referencias

[1] Cisco 2014 Annual Security Report, http://www.cisco.com/web/offer/gist_ty2_asset/Cisco_2014_ASR.pdf, (consultado 22/03/2016)

[2] Franco-Pedroso, J., & Gonzalez-Rodriguez, J. (2016). Linguistically-constrained formant-based i-vectors for automatic speaker recognition. *Speech Communication*, *76*, 61-81.

[3] Biometric Recognition Group –ATVS, http://atvs.ii.uam.es/listprojects.do, (consultado 18/05/2016)

[4] Kolias, C., Kambourakis, G., Stavrou, A., & Gritzalis, S. (2015). Intrusion detection in 802.11 networks: empirical evaluation of threats and a public dataset, *IEEE Communications Surveys & Tutorials*, no. 99, 2015.

[5] Wu, S. X., & Banzhaf, W. (2010). The use of computational intelligence in intrusion detection systems: A review. *Applied Soft Computing*, *10*(1), 1-35

[6] C. Low. (2005). "Understanding Wireless attacks & detection," GIAC Security Essentials Certification (GSEC) Practical Assignment Version 1.4c, http://cnscenter.future.co.kr/resource/hot-topic/wlan/1633.pdf

[7] Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

[8] Shalev-Shwartz, S., & Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge University Press., pp. 21-22

[9] Murphy, K. P. (2006). Naive bayes classifiers, *Technical Report,* http://www.cs.ubc.ca/murphyk/Teaching/CS340-Fall06/reading/NB.pdf, (consultado 19/05/2016)

[10] Naïve Bayes Clasifier, Dell, https://documents.software.dell.com/statistics/textbook/naive-bayes-classifier, (consultado 18/05/2016)

[11] Deeb, Z. A., Devine, T., & Geng, Z. (2010). Randomized Decimation HyperPipes. ACM.

[12] Schapire, R. E. (2013). Explaining adaboost. In *Empirical inference* (pp. 37-52). Springer Berlin Heidelberg.

[13] Hsuan- Tien Lin. (2008) Introdcution to Adaptative Boosting. National Taiwan University, https://www.csie.ntu.edu.tw/~htlin/course/ml08fall/doc/adaboost.pdf, (consultado 22/03/2016).

[14] Nevill-Manning, C. G., Holmes, G., & Witten, I. H. (1995, November). The development of Holte's 1R classifier. In *Artificial Neural Networks and Expert Systems, 1995. Proceedings., Second New Zealand International Two-Stream Conference on* (pp. 239-242). IEEE.

[15] Nasa, C. (2012). Evaluation of different classification techniques for web data. *International Journal of Computer Applications*, *52*.

[16] Mitchell, T. M. (1997). Machine learning, chapter 3: Decision Tree Learning. *Machine Learning*, McGraw-Hill, New York.

[17] Breiman, L., Cutler A., Random Forests, Berkeley University, https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#intro, (consultado 19/05/2016).

[18] Fan, W. (2004, July). On the optimality of probability estimation by random decision trees. In *Association for the Advancement of Artificial Intelligence (AAI)*. Vol. 2004, pp. 336-341).

[19] Quinlan, J. R. (2014). *C4. 5: programs for machine learning*. Elsevier.

[20] Frank, V. B. (2003). Classification trees: C4. 5. http://www.applied-mathematics.net/classification/classifier.pdf, (consultado 19/05/2016).

[21] Weka, http://www.cs.waikato.ac.nz/ml/weka/, (consultado 20/05/2016)

[22] Cortés, J. A. V. (2002). El Lenguaje de Programación AWK/GAWK. http://statusnet.fermosit.es/file/elmanytas-20120806T071150-aaysezg.pdf, (consultado 20/05/2016)

[23] Ramey, C., & Fox, B. (2006). Bash reference manual: reference documentation for Bash edition 4.3, for Bash version 4.3. Network Theory Limited.

[24] What is Python? Executive Summary, https://www.python.org/doc/essays/blurb/, (consultado 20/05/2016).

[25] Attribute-Relation File Format, Weka, http://www.cs.waikato.ac.nz/ml/weka/arff.html, (consultado 20/05/2016)

[26] Wireshark, https://www.wireshark.org/docs/dfref/w/wlan.html, (consultado 23/05/2015)

[27] Wireshark, https://www.wireshark.org/docs/dfref/w/wlan_mgt.html, (consultado 23/05/2015)

Anexos

I. Lista y contenido de cada atributo

1. frame.interface_id: Todo a ‘0’
2. frame.dlt: Todo a ‘?’
3. frame.offset_shift: Todo a ‘0’
4. frame.time_epoch: Tiempo de inicio de captura de tramas: 01/03/2014 08:08:22-09:08:22
5. frame.time_delta: Diferencia de tiempos entre un paquete y el anterior.
6. frame.time_delta_displayed: Diferencia de tiempos entre un paquete y el anterior. Igual que el atributo Nº5
7. frame.time_relative: Tiempo relativo desde la primera trama.
8. frame.len: Longitud paquete “on the wire”. Longitud minima:100 bytes; longitud máxima: 999 bytes
9. frame.cap_len: Longitud paquete almacenado en la captura. Igual que el atributo Nº8
10. frame.marked: Todo a ‘0’.
11. frame.ignored: Todo a ‘0’.
12. radiotap.version: Versión de cabecera. Todo a ‘0’.
13. radiotap.pad: Todo a ‘0’.
14. radiotap.length: Longitud de cabecera. 0,9996% valor 26, 0.0004% valor 13.
15. radiotap.present.tsft: Time synchronization function timer. 0,9996% valor 1, 0.0004% valor 0.
16. radiotap.present.flags: Propiedades de la trama. 0.9996% valor 1, 0.0004% valor 0.
17. radiotap.present.rate : Tasa de datos de transmisión y recepción. Todo a ‘1’.
18. radiotap.present.channel: Frecuencia de transmisión y recepción. 0.9996% valor 1, 0.0004% valor 0.
19. radiotap.present.fhss : Todo a ‘0’.
20. radiotap.present.dbm_antsignal: Potencia de señal de radio frecuencia en la antena. 0.9996% valor 1, 0.0004% valor 0.
21. radiotap.present.dbm_antnoise: Ruido de radio frecuencia en la antena. Todo a ‘0’.
22. radiotap.present.lock_quality: Todo a ‘0’.
23. radiotap.present.tx_attenuation: Todo a ‘0’.
24. radiotap.present.db_tx_attenuation: Todo a ‘0’.
25. radiotap.present.dbm_tx_power: Potencia de transmisión. Todo a ‘0’.
26. radiotap.present.antenna: Indicadores de la antena de recepción y transmisión de este paquete. 0.9996% valor 1, 0.0004% valor 0.
27. radiotap.present.db_antsignal: Potencia de señal de radio frecuencia en la antena. Todo a ‘0’.
28. radiotap.present.db_antnoise: Ruido de radio frecuencia en la antena. Todo a ‘0’.
29. radiotap.present.rxflags: Propiedades de las tramas recibidas. 0.9996% valor 1, 0.0004% valor 0.
30. radiotap.present.xchannel: Todo a ‘0’.
31. radiotap.present.mcs: Indica la información conocida. Todo a ‘0’.
32. radiotap.present.ampdu: Todo a ‘0’.
33. radiotap.present.vht: Todo a ‘0’.
34. radiotap.present.reserved: Todo a ‘0’.
35. radiotap.present.rtap_ns: Todo a ‘0’.
36. radiotap.present.vendor_ns: Todo a ‘0’.
37. radiotap.present.ext: Todo a ‘0’.
38. radiotap.mactime: Valor en microsegundos de la Función de sincronización de tiempo de la MAC cuando el primer MPDU (MAC protocol Data Unit) llega a la MAC. 0.0004% con valor ‘?’
39. radiotap.flags.cfp: 0.9996% valor 0, 0.0004% valor ?.
40. radiotap.flags.preamble: emisor y receptor con preámbulo corto. 9996% valor 0, 0.0004% valor ?.
41. radiotap.flags.wep: emisor y receptor con encriptación wep. 9996% valor 0, 0.0004% valor ?.
42. radiotap.flags.frag: emisor y receptor con fragmentación. 9996% valor 0, 0.0004% valor ?.
43. radiotap.flags.fcs: La trama incluye un campo de distribución del sistema (fcds). 0.9996% valor 1, 0.0004% valor ?.

44. radiotap.flags.datapad: Relleno entre la cabecera y el payload de la trama. 0.996% valor 0, 0.0004% valor ?.
45. radiotap.flags.badfcs: Trama recibido con un Frame Check Secuence erróneo. 0.9996% valor 0, 0.0004% valor '?'.
46. radiotap.flags.shortgi: Intervalo de guardia corto. Hay un 0.9996% valor 0, 0.0004% valor '?'.
47. radiotap.datarate: Velocidad a la que la trama fue enviada. 12 valores distintos
48. radiotap.channel.freq: Frecuencia del canal en MHz a la que la trama fue enviada. 15 valores distintos. se podría reducir a 10 valores distintos.
49. radiotap.channel.type.turbo: canal en estado 'Turbo'. 0.9996% valor 0, 0.0004% valor '?'.
50. radiotap.channel.type.cck: Modulación complementaria del código de claves. Tres valores: 0.0004% = '?', 0,5525% = 0, 0,4471% = 1
51. radiotap.channel.type.ofdm: Orthogonal frequency division multiplexing. tres valores: 0.0004% = '?', 0,5525% = 1, 0,4471% = 0.
52. radiotap.channel.type.2ghz: Espectro de 2 GHz. 0.9996% valor 1, 0.0004% valor '?'.
53. radiotap.channel.type.5ghz: Espectro de 5 GHz. 0.9996% valor 0, 0.0004% valor '?'.
54. radiotap.channel.type.passive: 0.9996% valor 0, 0.0004% valor '?'.
55. radiotap.channel.type.dynamic: Modulación dinámica complementaria del código de claves. 0.9996% = 0, 0.0004% = ?.
56. radiotap.channel.type.gfsk: Modulación "gaussian frequency shift keying". 0.9996% = 0, 0.0004% = ?.
57. radiotap.channel.type.gsm: 0.9996% valor 0, 0.0004% valor ?.
58. radiotap.channel.type.sturbo: Estado del turbo. 0.9996% valor 0, 0.000% valor ?.
59. radiotap.channel.type.half: Mitad de la tasa del canal. 0.9996% valor 0, 0.0004% valor ?.
60. radiotap.channel.type.quarter: Un cuarto de la tasa del canal, 0.999% valor 0, 0.0004% valor ?.
61. radiotap.dbm_antsignal: Potencia de la señal de radiofrecuencia en la antena. 67 valores distintos.
62. radiotap.antenna: Número de antena del que se envió esta trama. 0.9996 valor 1, 0.0004 valor ?.
63. radiotap.rxflags.badplcp: Trama con valor erróneo de PLCP (Physical Layer Convergence Procedure). 0.9996% valor 0, 0.0004% valor ?.
64. wlan.fc.type_subtype: Campo de subtipo de la trama dentro del tipo de dicha trama. 21 valores distintos, 15 representativos.
65. wlan.fc.version: Campo de la versión de la trama. todo a 0, no aporta nada
66. wlan.fc.type: Campo del tipo de la trama. 3 valores distintos: 0.2369% = 0, 0.25362% = 1, 0.5095% = 2
67. wlan.fc.subtype: Campo del subtipo de la trama. 12 valores distintos.
68. wlan.fc.ds: Estado del Sistema de distribución. 3 valores distintos. 0.4905% = 0x00, 0.1369% = 0x01, 0.3726% = 0x02.
69. wlan.fc.frag: Indica si hay más fragmentos faltantes. 2 valores, 0.99963%=0, 0.00037=1
70. wlan.fc.retry: Procesar de nuevo la trama. 2 valores, 0.87289%=0, 0.12711%=1.
71. wlan.fc.pwrmgt: Administración de la potencia. 2 valores, 0,97675= 0, 0,02325=1.
72. wlan.fc.moredata: El punto de acceso tiene más tramas almacenadas que enviar. 2 valores, 0,99823=0, 0,00177=1.
73. wlan.fc.protected: Indica si está protegida o no la trama. 2 valores. 0,50671=0, 0,49329=1.
74. wlan.fc.order: Todo a '0'.
75. wlan.duration: Duración del valor de cada uno de los campos. 323 valores distintos.
76. wlan.ra: Dirección MAC del receptor intermedio. 1776 valores distintos.
77. wlan.da: Dirección MAC del receptor final. 171 valores distintos.
78. wlan.ta: Dirección MAC del emisor intermedio: 3467 valores distintos.
79. wlan.sa: Dirección MAC del emisor original de la trama. 3473 valores distintos.
80. wlan.bssid: bss identificador. 3392 valores distintos
81. wlan.frag: Número de fragmentos. 14 valores distintos. 0,74601% = 0, 0,25362% = ? y el resto otros valores no representativos.
82. wlan.seq: Número de secuencia. 4097 valores distintos.
83. wlan.bar.type: Prácticamente todo asignado al valor '?' excepto 23 valores (0,00001).

84. wlan.ba.control.ackpolicy: Prácticamente todo asignado al valor '?'.
85. wlan.ba.control.multitid: Identificador de tráfico múltiple. Prácticamente todo asignado al valore '?' (0.99994%) y resto a '0'.
86. wlan.ba.control.cbitmap: Compressed bit map. Prácticamente todo a '?' (0.99994%) y resto a 1. no aporta nada.
87. wlan.bar.compressed.tidinfo: Información de indentificador del multitráfico. Prácticamente todo a '?' (0.99999%).
88. wlan.ba.bm: Prácticamente todo a '?' (0.99996%)
89. wlan.fcs_good: Confirmación del número de secuencia. dos valores, 0.9996% valor 1, 0.0004% valor '?'.
90. wlan_mgt.fixed.capabilities.ess: Capacidades del set de servicios extendidos. 3 valores 0,89934% = '?', 0,00044% = 0, 0,10022% = 1.
91. wlan_mgt.fixed.capabilities.ibss: Estado del ibss (independent basic service set). 3 valores 0,89934% = '?', 0,10023% = 0, 0,00043% = 1.
92. wlan_mgt.fixed.capabilities.cfpoll.ap: Capacidades del CFP (Contention Free Period). 9 valores distintos, solo uno realmente representativo el 0x0000
93. wlan_mgt.fixed.capabilities.privacy: Capacidades de privacidad. 3 valores. 0.89933% = '?', 0,03250% = 0, 0,06817% = 1
94. wlan_mgt.fixed.capabilities.preamble: Preámbulo. 3 valores distintos 0.89933% = '?', 0.04539% = 0, 0.05528% = 1.
95. wlan_mgt.fixed.capabilities.pbcc: Capacidades de la modulación PBCC. 3 valores distintos, 0.89933% = '?', 0.10023% = 0, 0,00043% = 1.
96. wlan_mgt.fixed.capabilities.agility: Capacidades de agilidad del canal. 3 valores distintos, 0.89933% = '?', 0.10022% = 0, 0.00044% = 1.
97. wlan_mgt.fixed.capabilities.spec_man: Capacidades de administración del espectro. 3 valores distintos, 0.89933% = '?', 0.10021=0, 0.00045=1.
98. wlan_mgt.fixed.capabilities.short_slot_time: Utilizacicón de short slot time que permite incrementar el rendimiento. 0.89933% = '?', 0.00379% = 0, 0.09687% = 1.
99. wlan_mgt.fixed.capabilities.apsd: Utilizacicón de automatic power save delivery. 3 valores distintos 0.89933% = '?', 0.10021% = 0, 0.00045% = 1.
100. wlan_mgt.fixed.capabilities.radio_measurement: Medida de radio. 3 valores distintos, 0.89933% = ?, 0.10021% = 0, 0,00045% = 1.
101. wlan_mgt.fixed.capabilities.dsss_ofdm: Indica el tipo de multiplexación utilizada. 3 valores distintos, 0.89933% = '?', 0.10023% = 0, 0.00043% = 1.
102. wlan_mgt.fixed.capabilities.del_blk_ack: Utilización del esquema opcional Block ACK para mejorar la eficiencia de la MAC. 3 valores, 0.89933% = '?', 0.10023% = 0, 0.00043% = 1.
103. wlan_mgt.fixed.capabilities.imm_blk_ack: Bloque ACK inmediato. 3 valores, 0.89933% = '?', 0.10023% = 0, 0.00043% = 1.
104. wlan_mgt.fixed.listen_ival: Intervalo de escucha. 6 valores distintos, solo 2 representativos y un 0.99133% con valor '?'.
105. wlan_mgt.fixed.current_ap: Current access point. 5 valores distintos , un 0.99995% a '?'.
106. wlan_mgt.fixed.status_code: Código de estado. 4 valores distintos, un 0.98388% a '?' y un 0.01478% = 0x0000.
107. wlan_mgt.fixed.timestamp: Timestamp. 137911 valores distintos un 0.91530% a '?' y un 0.0039% = 0x00
108. wlan_mgt.fixed.beacon: Intervale de tramas baliza. 1546 valores distintos, un 0.91530% a '?' y un 0.08383% = 100
109. wlan_mgt.fixed.aid: Identificador de asociación. 7 valores distintos, un 0.99271% a '?'.
110. wlan_mgt.fixed.reason_code: Reason code que indica el motivo de una notificación de administración no solicitada. 26 valores distintos, un 0.87651% = '?' y tres valores realmente representativos
111. wlan_mgt.fixed.auth.alg: Algoritmo de autentificación. 3 valores 0,99117% = '?', 0.00858% = 0, 0.00025= 1.
112. wlan_mgt.fixed.auth_seq: Autentificación del número de secuencia. 3 valores distintos, un 0.99116%

= ‘?’.

113. wlan_mgt.fixed.category_code: Código de la categoría. 2 valores distintos, un 0.99998% = ‘?’. Apenas aporta información.
114. wlan_mgt.fixed.htact: Atributos de “high throughput” action. un 0.99999% = ‘?’ y resto a 0.
115. wlan_mgt.fixed.chanwidth: Ancho del canal soportado. 3 valores distintos, un 0.99999% = ‘?’.
116. wlan_mgt.fixed.fragment: Fragmento de la respuesta de la consulta GAS (Genreic Advertisement Service).Un 0.99994% = ‘?’ y resto a 0.
117. wlan_mgt.fixed.sequence: Número de secuencia inicial. Hay 43 valores distintos pero con un 0.99994% = ‘?’
118. wlan_mgt.tagged.all: Parámetros etiquetados. Hay un 0.89390% = ‘?’ y un 0.10610% = 1
119. wlan_mgt.ssid: Service set identifier sid, sirve para identificar y nombrar la red WAN
120. wlan_mgt.ds.current_channel:14 valores distintos con un 0.91265% = ‘?’ y un 0.08526% = 6.
121. wlan_mgt.tim.dtim_count Cuenta del DTIM (Delivery traffic indication map or message). Hay 26 valores de los cuales un 0.92766% = ‘?’.
122. wlan_mgt.tim.dtim_period: Perido de DTIM (Delivery traffic indication map or message). Hay 27 valores distintos, pero solo hay 2 representativos, un 0.92766% = ‘?’.
123. wlan_mgt.tim.bmapctl.multicast: Indica si el tráfico almacenado en el punto de acceso es de difusión o multicast. Hay un 0.92766% = ?, un 0.07135% = 0, 0.00100%= 1.
124. wlan_mgt.tim.bmapctl.offset: Bitmap offset. Hay 22 valores distintos, un 0.92766% = ‘?’, un 0.07232% = 0x00.
125. wlan_mgt.country_info.environment: Entorno. Hay 21 valores, un 0.99724% = ? y solo 2 valores son realmente representativos
126. wlan_mgt.rsn.version: Versión de los RNs (Relay Nodes). Hay 22 valores distintos, un 0.95715% = ?, un 0.04284% = 1.
127. wlan_mgt.rsn.gcs.type: Conjuntos de cifrado. Hay 21 valores distintos, un 0.95715% = ‘?’, un 0.04279% = 2.
128. wlan_mgt.rsn.pcs.count: Conteo de pares de conjuntos de cifrado. Hay 23 valores distintos, un 0.95715% = ‘?’, un 0.04014% = 2 y un 0.00270% = 1.
129. wlan_mgt.rsn.akms.count: Conteo de la clave de autenticación. Hay 20 valores distintos con 0.95715% = ‘?’ y un 0.04284% = 1.
130. wlan_mgt.rsn.akms.type: Tipo de administración de la clave de autenticación. Hay un 0.95716% = ‘?’ y 0.04284% = 2.
131. wlan_mgt.rsn.capabilities.preauth: Capacidades de pre-autenticación del RNs. Hay 0.95715% = ‘?’, 0.04282% = 0, 0.00003% = 1.
132. wlan_mgt.rsn.capabilities.no_pairwise: Capacidad de no emparejar los RNs. Hay un 0.95715% = ‘?’, 0.04285% = 0, 0.00001% = 1.
133. wlan_mgt.rsn.capabilities.ptksa_replay_counter: Conteo de capacidades de los RNs ptksa. Hay un 0.95714% = ‘?’.
134. wlan_mgt.rsn.capabilities.gtksa_replay_counter: Conteo de capacidades de los RNs gtksa. Hay un 0.95714% = ‘?’.
135. wlan_mgt.rsn.capabilities.mfpr: Protección requerida para la trama. Un 0.95715% = ‘?’, 0.04285% = 0, 0.00001 = 1.
136. wlan_mgt.rsn.capabilities.mfpc: Capacidad de protección de la trama. Un 0.95715% = ‘?’, 0,04285 = 0, 0.0000 = 1.
137. wlan_mgt.rsn.capabilities.peerkey: Habilitación de claves de pares. Un 0.95715% = ‘?’, 0.04285% = 0, 0.00001%=1.
138. wlan_mgt.tcprep.trsmt_pow: Potencia de transmisión. Un 0.99998% = ‘?’.
139. wlan_mgt.tcprep.link_mrg: Margen de enlace. Un 0.99998% = ‘?’.
140. wlan.wep.iv: Vector de inicialización. Hay 705130 valores distintos, un 0.52619% = ‘?’.
141. wlan.wep.key: Índice de la clave. Hay 4 valores distintos (0,1,2,3), un 0.50671% = ‘?’.
142. wlan.wep.icv: Valor de confirmación de integridad de WEP. Hay 709750 valores distintos.
143. wlan.tkip.extiv: Vector de inicialización tkip (temporal Key Integrity Protocol). Hay 31258 valores distintos, de los cuales un 0.98222% = ‘?’.

144. wlan.ccmp.extiv: CCMP del vector de inicialización. Hay 968 valores de los cuales un 0.99829% = ‘?’.
145. wlan.qos.tid: Calidad de servicio de TID (Traffic identificación). Hay 4 valores distintos (0,1,2,6), un 0.08559% = 0 y 0.28321% = 1, un 0.63112% = ‘?’.
146. wlan.qos.priority: Prioridad de servicio de calidad. Hay 4 valores distintos (0,1,2,6), un 0.08559% = 0 y 0.28321% = 1, un 0.63112% = ‘?’.
147. wlan.qos.eosp: End of service period. Hay 0.71279% = ‘?’ , un 0.28721% = 0.
148. wlan.qos.ack: Calidad de servicio del ack. Hay dos valores un 0.368872% = 0x0000, un 0.631126% = ‘?’.
149. wlan.qos.amsdupresent: Tipo de carga del paquete (payload). Hay un 0.63168% = ‘?’ y un 0.36832% = 0.
150. wlan.qos.buf_state_indicated: Todo a ‘?’ .
151. wlan.qos.bit4: Calidad de servicio del bit 4. Hay un 0.91833% = ‘?’ y un 0.08167% = 0.
152. wlan.qos.txop_dur_req: Duración de la solicitud TXOP (transmition opportunity). Hay un 0.91833% = ‘?’ y un 0.08167% = 0.
153. wlan.qos.buf_state_indicated: Estado del buffer. Hay un 0.71279% = ‘?’ y un 0.28721% = 0.
154. data.len: Tamaño del paquete, 1309 valores distintos, un 0.50291% = ‘?’. [26, 27]

II. Lista de atributos de cada experimento

1.Lista de 82 atributos

1. frame.time_epoch
2. frame.time_delta
3. frame.time_relative
4. frame.len
5. radiotap.length
6. radiotap.mactime
7. radiotap.datarate
8. radiotap.channel.freq
9. radiotap.channel.type.cck
10. radiotap.channel.type.ofdm
11. radiotap.dbm_antsignal
12. wlan.fc.type_subtype
13. wlan.fc.type
14. wlan.fc.subtype
15. wlan.fc.ds
16. wlan.fc.frag
17. wlan.fc.retry
18. wlan.fc.pwrmgt
19. wlan.fc.moredata
20. wlan.fc.protected
21. wlan.duration
22. wlan.ra
23. wlan.da
24. wlan.ta
25. wlan.sa
26. wlan.bssid
27. wlan.frag
28. wlan.seq
29. wlan.ba.control.ackpolicy
30. Wlan.ba.control.multitid
31. wlan_mgt.fixed.capabilities.ess
32. wlan_mgt.fixed.capabilities.ibss
33. wlan_mgt.fixed.capabilities.cfpoll.ap
34. wlan_mgt.fixed.capabilities.privacy
35. wlan_mgt.fixed.capabilities.preamble
36. wlan_mgt.fixed.capabilities.pbcc
37. wlan_mgt.fixed.capabilities.agility
38. wlan_mgt.fixed.capabilities.spec_man
39. wlan_mgt.fixed.capabilities.short_slot_time
40. wlan_mgt.fixed.capabilities.apsd
41. wlan_mgt.fixed.capabilities.radio_measurement
42. wlan_mgt.fixed.capabilities.dsss_ofdm
43. wlan_mgt.fixed.capabilities.del_blk_ack
44. wlan_mgt.fixed.capabilities.imm_blk_ack
45. wlan_mgt.fixed.listen_ival
46. wlan_mgt.fixed.status_code

47. wlan_mgt.fixed.timestamp
48. wlan_mgt.fixed.beacon
49. wlan_mgt.fixed.aid
50. wlan_mgt.fixed.reason_code
51. wlan_mgt.fixed.auth.alg
52. wlan_mgt.fixed.auth_seq
53. wlan_mgt.fixed.fragment
54. wlan_mgt.tagged.all
55. wlan_mgt.ds.current_channel
56. wlan_mgt.tim.dtim_count
57. wlan_mgt.tim.dtim_period
58. wlan_mgt.tim.bmapctl.multicast
59. wlan_mgt.tim.bmapctl.offset
60. wlan_mgt.country_info.environment
61. wlan_mgt.rsn.version
62. wlan_mgt.rsn.gcs.type
63. wlan_mgt.rsn.pcs.count
64. wlan_mgt.rsn.akms.count
65. wlan_mgt.rsn.akms.type
66. wlan_mgt.rsn.capabilities.preauth
67. wlan_mgt.rsn.capabilities.no_pairwise
68. wlan_mgt.rsn.capabilities.ptksa_replay_counter
69. wlan_mgt.rsn.capabilities.gtksa_replay_counter
70. wlan_mgt.rsn.capabilities.mfpc
71. wlan.wep.iv
72. wlan.wep.key
73. wlan.wep.icv
74. wlan.tkip.extiv
75. wlan.qos.tid
76. wlan.qos.eosp
77. wlan.qos.ack
78. wlan.qos.amsdupresent
79. wlan.qos.bit4
80. wlan.qos.txop_dur_req
81. wlan.qos.buf_state_indicated
82. data.len

2.Lista de los 18 atributos

1. frame.time_epoch	1. wlan.ra
2. frame.len	2. wlan.da
3. radiotap.mactime	3. wlan.ta
4. radiotap.datarate	4. wlan.sa
5. radiotap.dbm_antsignal	5. wlan.seq
6. wlan.fc.type_subtype	6. wlan_mgt.fixed.reason_code
7. wlan.fc.subtype	7. wlan.wep.iv
8. wlan.fc.ds	8. wlan.wep.icv
9. wlan.duration	9. data.len

III. Resultados detallados

1. Matrices de confusión y comparación de resultados obtenidos con la base de datos de 82 atributos

A continuación, se muestran las matrices de confusión de todos los algoritmos utilizados. En rojo están marcado el algoritmo que mejor funciona para detectar un tipo de ataque en concreto. Con el objetivo de comparar los resultados obtenidos una lista de atributos reducida a 82 atributos y obtenidos por los creadores de la base de datos, se presentan en la tabla 20 y 21 las matrices de confusión de los dos algoritmos que mejor funcionaron.

Como se puede observar, con la lista de 82 atributos el algoritmo de Random Forest es capaz de detectar un mayor número de ataques de inundación y de tramas normales, que con la lista de 155 atributos que utilizaron los creadores de la base de datos. Por el contrario, los ataques de interposición se detectan de mejor manera utilizando la base de datos de 155 atributos. A pesar de estas pequeñas diferencias y mejoras, no se aprecian grandes mejoras en cuanto a clasificación.

Classified→	flooding	impersonation	injection	normal
flooding	**4183**	0	0	3914
impersonation	0	1374	0	18705
injection	0	0	16680	2
normal	62	16	0	530707

Tabla 21. Resultados con el algoritmo J48 y base de datos de 82 atributos

Classified →	flooding	impersonation	injection	normal
flooding	**4977**	0	0	3120
impersonation	0	1426	0	18653
injection	0	0	13811	2871
normal	0	1	0	530784

Tabla 22. Resultados con el algoritmo Random Forest y base de datos de 82 atributos

Classified→	flooding	impersonation	injection	normal
flooding	4857	599	0	2641
impersonation	0	1450	0	18629
injection	0	0	16680	2
normal	8	6	0	530771

Tabla 23. Resultados presentados en [4] con el algoritmo J48 y base de datos de 155 atributos

Classified →	flooding	impersonation	injection	normal
flooding	4020	0	0	4077
impersonation	0	1291	28	18760
injection	0	0	14212	2470
normal	1	1	54	530729

Tabla 24. Resultados presentados en [4] con el algoritmo Random Forest y base de datos de 155 atributos

A continuación, se presentan el resto de matrices de confusión obtenidas en la evaluación de los distintos algoritmos. De este apartado cabe destacar la gran clasificación del algoritmo de Naive Bayes. La tasa de falsos negativos para el ataque de inundación es <0.1%, los mejores resultados obtenidos para la detección de este ataque. Por el contrario, también en este caso obtenemos la tasa de falsos positivos del 0.1%, mucho más alta que para el resto de algoritmos.

Otra observación a recalcar de este apartado, es que ninguno de estos algoritmos es capaz de detectar ni siquiera una trama del ataque de interposición, siendo este claramente el ataque más difícil de detectar.

Classified →	flooding	impersonation	injection	normal
flooding	**0**	0	0	8097
impersonation	**0**	0	0	20079
injection	**0**	0	13644	3038
normal	**0**	3	7	530775

Tabla 25. Resultados con el algoritmo OneR y base de datos de 82 atributos

Classified →	flooding	impersonation	injection	normal
flooding	**432**	0	0	7665
impersonation	0	0	18606	1473
injection	0	0	16193	489
normal	3	32831	683	516816

Tabla 26. Resultados con el algoritmo Random Tree y base de datos de 82 atributos

Classified →	flooding	impersonation	injection	normal
flooding	83	0	0	8014
impersonation	371	0	0	19708
injection	0	0	167	16515
normal	196	0	0	530589

Tabla 27. Resultados con el algoritmo Hyperpipes y base de datos de 82 atributos

Classified →	flooding	impersonation	injection	normal
flooding	0	0	0	8097
impersonation	0	0	18606	1473
injection	0	0	13954	2728
normal	0	3	16014	514771

Tabla 28. Resultados con el algoritmo AdaBoost y base de datos de 82 atributos

Classified →	flooding	impersonation	injection	normal
flooding	7851	0	0	246
impersonation	1346	0	0	18733
injection	0	0	33	16649
normal	56756	0	0	474029

Tabla 29. Resultados con el algoritmo Naive Bayes y base de datos de 82 atributos

Classified →	flooding	impersonation	injection	normal
flooding	0	0	0	8097
impersonation	0	0	0	20079
injection	0	0	0	16682
normal	0	0	0	530785

Tabla 30. Resultados con el algoritmo ZeroR y base de datos de 82 atributos

2. Matrices de confusión y comparación de resultados obtenidos con la base de datos de 18 atributos

Aquí igualmente se presentan las matrices de confusión de los algoritmos que mejor funcionaron con la lista de 18 atributos y las matrices de confusión de los algoritmos con los que los creadores de la base de datos obtuvieron mejores resultados. Se puede observar que las matrices de confusión varían muy poco y las tasas de falsos negativos son muy similares. Se puede observar una mejora en la detección del ataque de tipo inundación utilizando el algoritmo J48 en los resultados presentados en [4] con respecto a los obtenidos en este trabajo. Por otro lado, en los resultados presentados en [4], la tasa de falsos positivos aumenta, aunque un porcentaje mínimo.

Classified →	flooding	impersonation	injection	normal
flooding	4871	0	0	3226
impersonation	0	1471	0	18608
injection	0	0	16680	2
normal	6	16	0	530763

Tabla 31. Resultados con el algoritmo J48 y base de datos de 18 atributos

Classified →	flooding	impersonation	injection	normal
flooding	**4038**	0	0	4059
impersonation	**0**	1462	0	18617
injection	**0**	0	14011	2671
normal	**0**	0	0	530785

Tabla 32. Resultados con el algoritmo Random Forest y base de datos de 18 atributos

Classified→	flooding	impersonation	injection	normal
flooding	5544	0	0	2553
impersonatio n	148	1287	0	18644
injection	0	0	16680	2
normal	116	75	6	530588

Tabla 33. Resultados presentados en [4] con el algoritmo J48 y base de datos de 20 atributos

Classified →	flooding	impersonation	injection	normal
flooding	5494	0	161	2442
impersonation	0	1470	0	18609
injection	0	156	16253	273
normal	3	82	1	530700

Tabla 34. Resultados presentados en [4] con el algoritmo Random Tree y base de datos de 20 atributos

A continuación, se presentan el resto de matrices de confusión obtenidas en la evaluación de los distintos algoritmos. De este apartado cabe destacar, al igual que en el anterior, la óptima clasificación de ataques de inundación realizada por el algoritmo de Naive Bayes. Al igual que con la base de datos con 82 atributos, en este caso se obtiene una tasa de falsos negativos <0.1%. Por el contrario, la tasa de falsos positivos aumenta considerablemente a un 20%. Por otro lado, una vez más se ve la dificultad de detectar los algoritmos de interposición, aunque con esta nueva lista de atributos el algoritmo de Random Tree realiza una mejor clasificación de las tramas, pero mejora su tasa de falsos negativos, pasando de un 7% a un 70%.

Classified →	flooding	impersonation	injection	normal
flooding	**0**	0	0	8097
impersonation	**0**	0	0	20079
injection	**0**	0	13644	3038
normal	**0**	3	7	530775

Tabla 35. Resultados con el algoritmo OneR y base de datos de 18 atributos

Classified →	flooding	impersonation	injection	normal
flooding	3476	24	0	4597
impersonation	0	1410	4763	13906
injection	0	0	16667	15
normal	2	27	572	530184

Tabla 36. Resultados con el algoritmo Random Tree y base de datos de 18 atributos

Classified →	flooding	impersonation	injection	normal
flooding	5927	0	0	2170
impersonation	1473	0	0	18606
injection	0	13407	404	2871
normal	59612	13976	0	457197

Tabla 37. Resultados con el algoritmo Hyperpipes y base de datos de 18 atributos

Classified →	flooding	impersonation	injection	normal
flooding	0	0	0	8097
impersonation	0	0	18606	1473
injection	0	0	13954	2728
normal	0	3	16014	514771

Tabla 38. Resultados con el algoritmo AdaBoost y base de datos de 18 atributos

Classified →	flooding	impersonation	injection	normal
flooding	7823	0	0	274
impersonation	1421	0	0	18658
injection	0	0	263	16419
normal	105394	0	0	425391

Tabla 39. Resultados con el algoritmo Naive Bayes y base de datos de 18 atributos

Classified →	flooding	impersonation	injection	normal
flooding	0	0	0	8097
impersonation	0	0	0	20079
injection	0	0	0	16682
normal	0	0	0	530785

Tabla 40. Resultados con el algoritmo ZeroR y base de datos de 18 atributos

3. Matrices de confusión y comparación de resultados obtenidos con la base de datos de 18 atributos que incluye ampliación de contexto

Como se observa en las siguientes tablas y se comenta en el apartado de resultados, al incluir la ampliación de contexto el algoritmo J48 mejora considerablemente su detección de los ataques de tipo inundación. Por otro lado, al algoritmo de Random Forest obtiene una tasa de falsos positivos de prácticamente el 0%, clasificando todas menos una trama normal correctamente. En cuanto a la detección de los ataques de tipo interposición no se aprecia ninguna mejora considerable. Otro aspecto importante a resalta en este apartado es el aumento de clasificación como ataque a las tramas normales, en concreto un aumento en la clasificación de tipo inundación a tramas normales.

Classified →	flooding	impersonation	injection	normal
flooding	5508	0	0	2589
impersonation	0	1469	0	18610
injection	0	0	16680	2
normal	1225	22	0	529538

Tabla 41. Resultados con el algoritmo J48, base de datos de 18 atributos y ampliación de contexto

Classified →	flooding	impersonation	injection	normal
flooding	4876	0	0	3221
impersonation	0	1196	0	18883
injection	0	0	13809	2873
normal	1	0	0	530784

Tabla 42. Resultados] con el algoritmo Random Forest, base de datos de 18 atributos y ampliación de contexto

Classified →	flooding	impersonation	injection	normal
flooding	**0**	0	0	8097
impersonation	**0**	0	0	20079
injection	**0**	0	13644	3038
normal	**0**	3	7	530775

Tabla 43. Resultados con el algoritmo OneR, base de datos de 18 atributos y ampliación de contexto

Classified →	flooding	impersonation	injection	normal
flooding	4905	0	0	3177
impersonation	11	983	0	19084
injection	2	0	13838	2842
normal	40	48	7	530690

Tabla 44. Resultados con el algoritmo Random Tree, base de datos de 18 atributos y ampliación de contexto

Classified →	flooding	impersonation	injection	normal
flooding	5590	0	0	2507
impersonation	1472	1	0	18606
injection	0	12900	907	2875
normal	57845	13751	0	459189

Tabla 45. Resultados con el algoritmo Hyperpipes, base de datos de 18 atributos y ampliación de contexto

Classified →	flooding	impersonation	injection	normal
flooding	0	0	0	8097
impersonation	0	0	18563	1516
injection	0	0	13794	2888
normal	0	3	8209	522576

Tabla 46. Resultados con el algoritmo AdaBoost, base de datos de 18 atributos y ampliación de contexto

Classified →	flooding	impersonation	injection	normal
flooding	7944	0	0	153
impersonation	2511	0	0	17568
injection	3	0	553	16126
normal	141090	0	0	389695

Tabla 47. Resultados con el algoritmo Naive Bayes, base de datos de 18 atributos y ampliación de contexto

Classified →	flooding	impersonation	injection	normal
flooding	0	0	0	8097
impersonation	0	0	0	20079
injection	0	0	0	16682
normal	0	0	0	530785

Tabla 48. Resultados con el algoritmo ZeroR, base de datos de 18 atributos y ampliación de contexto