Detección de Intrusiones en Redes de Comunicaciones Usando Minería de Datos*

Arturo Calvera[1], Sergio Ilarri[2], and Fernando Tricas[2]

[1] Universidad de Zaragoza, Zaragoza, España
[2] I3A, Universidad de Zaragoza, Zaragoza, España
776303@unizar.es, silarri@unizar.es, ftricas@unizar.es

Resumen Los ciberataques son una importante amenaza a la cual los/as ciudadanos/as están expuestos/as diariamente. En el contexto del proyecto NEAT-AMBIENCE, se aborda el diseño de técnicas de gestión de datos para ayudarles a enfrentarse a problemas modernos y mejorar sus vidas diarias. Por ello, en este trabajo en concreto, consideramos cómo ayudar a usuarios/as de sistemas informáticos a protegerse de ciberataques mediante técnicas de minería de datos. En particular, hemos desarrollado una herramienta de apoyo a la evaluación y dos aplicaciones de detección de intrusiones, habiendo evaluado con una prueba de concepto su potencial utilización en un entorno real. Además, hemos realizado una evaluación experimental comparando el rendimiento de diversas técnicas de minería de datos, que indica que dichas técnicas pueden ayudar a detectar diversos tipos de intrusiones. El trabajo muestra cómo con las técnicas de gestión de datos apropiadas es posible proporcionar asistencia a los usuarios en la difícil tarea de proteger sus sistemas.

Keywords: minería de datos · detección de intrusiones · ayuda a la toma de decisiones

1. Introducción

Los ciberataques son una importante amenaza a la cual las personas están expuestas diariamente. Por ello, en el contexto del proyecto *NEAT-AMBIENCE* (*Next-gEnerATion dAta Management to foster suitable Behaviors and the resilience of cItizens against modErN ChallEnges*) [20], donde se aborda el diseño de técnicas de gestión de datos que ayuden a las personas a tomar mejores decisiones ante los desafíos modernos existentes, resulta de interés explorar el ámbito de la ciberseguridad. Por ejemplo, en el contexto geopolítico actual, la guerra entre Ucrania y Rusia refleja esta situación: las amenazas de ciberseguridad derivadas de este conflicto tienen alcance global, es decir, no afectan únicamente a los países directamente enfrentados en el conflicto, sino también a países occidentales que sancionan a Rusia [12,14].

* Financiado por MCIN/AEI/ 10.13039/501100011033 y el Departamento de Ciencia, Universidad y Sociedad del Conocimiento del Gobierno de Aragón.

Creemos que la aplicación de técnicas de minería de datos puede ayudar a detectar ciberataques. De hecho, existen diversas líneas de investigación que exploran la relación entre ambas líneas, por ejemplo en el área de la defensa activa [18], el campo de la seguridad web [17] y también en el campo de este trabajo, que es concretamente el de los sistemas de detección de intrusiones (*Intrusion Detection Systems* o *IDSs*) [1,10,13]. Existen incluso herramientas comerciales que afirman utilizar este tipo de técnicas (aunque principalmente se centran en los comportamientos extraños, más que en la detección de ataques concretos), como Darktrace [9], o aproximaciones desde el mundo de la investigación, como TIRESIAS [19]. Por ello, en este artículo se considera el desarrollo de herramientas de ayuda a la detección de intrusiones en sistemas de comunicaciones utilizando técnicas de minería de datos. Para ello, se pretende analizar el rendimiento e idoneidad de distintas técnicas de minería de datos aplicadas sobre datos de comunicaciones en redes informáticas. Con estas técnicas se pretende detectar patrones intrínsecos en los datos que permitan identificar aquellas conexiones que puedan estar comprometiendo la seguridad del sistema.

Contar con herramientas de apoyo para protegerse puede ser de gran utilidad. No hay que olvidar que, en determinados contextos, la administración de sistemas puede quedar reducida a equipos mínimos de personas (e incluso a veces esta tarea se aborda en solitario) y que no tienen por qué tener acceso a los recursos de los que disponen organizaciones más grandes o Centros de Operaciones de Seguridad (*Security Operation Center, SOC*) donde los comportamientos sospechosos pueden gestionarse de manera más atendida. En cualquier caso, la ciberseguridad actual parece dirigirse a proporcionar la máxima ayuda a las personas implicadas por ser intensiva en cantidad de intervenciones requeridas, pero también en tiempo, puesto que los ataques pueden producirse en cualquier momento, con aproximaciones muy diversas y con mucha variabilidad de intensidad, desde intentos casuales, exploratorios, hasta ataques de denegación de servicio distribuidos (*Distributed Denial of Service, DDOS*).

Además de realizar una comparación experimental de técnicas, que muestra que incluso el uso de técnicas sencillas podría permitir obtener resultados aceptables, como parte de este trabajo también hemos desarrollado una herramienta de apoyo y dos aplicaciones. Por una lado, hemos obtenido una herramienta que permite comparar la eficiencia de distintas técnicas de minería de datos aplicadas a la detección de intrusiones, que hemos utilizado en nuestra propia evaluación experimental. Por otro lado, hemos implementado una aplicación que permite detectar potenciales intrusiones a partir de un conjunto de datos de comunicaciones de entrada proporcionado y otra aplicación que es un prototipo de detección de intrusiones basado en una de dichas técnicas. La estructura del resto del artículo es como sigue. En primer lugar, en la Sección 2, se presentan la herramienta y las aplicaciones desarrolladas en este trabajo. Después, en la Sección 3 se presenta una evaluación experimental cuyo foco es la evaluación y comparación de diversas técnicas de minería de datos para detección de intrusiones. Finalmente, en la Sección 4, se incluyen las conclusiones y algunas líneas de trabajo futuro que pueden abordarse.

2. Herramientas y Aplicaciones Desarrolladas

En esta sección se describen las herramientas y aplicaciones desarrolladas como parte del trabajo presentado en este artículo, cuyo propósito se resume en la Figura 1. En la Sección 2.1 se presenta la herramienta *ERTMD*, en la Sección 2.2 se presenta *IDS-NET* y en la Sección 2.3 *IDS-NET-DAEMON*. La herramienta *ERTMD* se ha utilizado como soporte para la evaluación experimental mostrada en la Sección 3 y las otras dos aplicaciones se han desarrollado para ayudar a distintos tipos de usuarios a tomar decisiones de seguridad apropiadas.

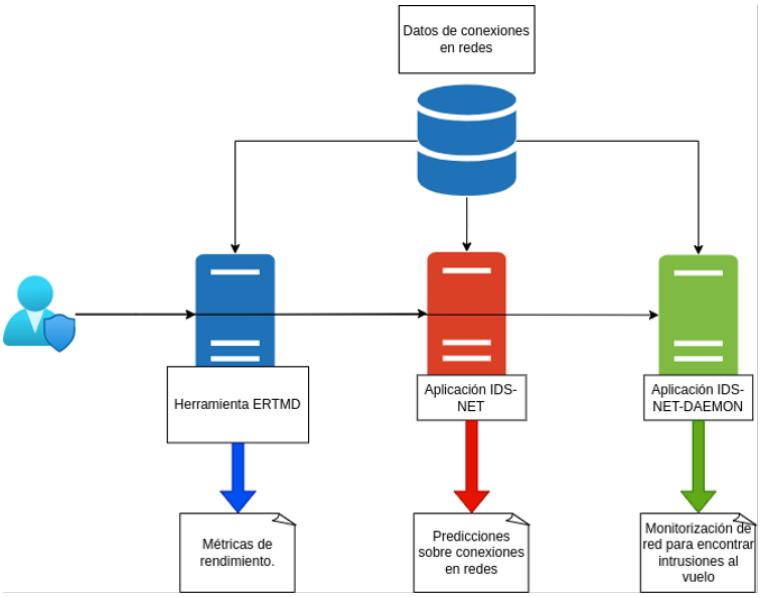

Figura 1. Diagrama resumen de la herramienta y aplicaciones implementadas

2.1. Herramienta ERTMD

En esta sección se presenta la herramienta de apoyo para *Evaluación de Rendimiento de Técnicas de Minería de Datos* (*ERTMD*) desarrollada. El objetivo de la herramienta es facilitar el trabajo de evaluación y comparación de rendimiento de distintas aproximaciones de detección de intrusiones basadas en minería de datos. El usuario objetivo de la aplicación es un usuario experto en la materia, es decir, que entiende las métricas que se van a generar, comprende las consideraciones a tener en cuenta a la hora de proporcionar un conjunto de datos de entrenamiento y prueba, conoce las técnicas de minería de datos y su funcionamiento. En definitiva, la herramienta ERTMD está pensada para personas que están muy familiarizadas con el campo de la minería de datos. Se

trata de una aplicación de línea de instrucciones desarrollada en Python, en la que es posible cargar conjuntos de datos indicados por el usuario para comparar cómo se comportan las distintas técnicas de minería de datos sobre ellos. La herramienta utiliza la biblioteca *scikit-learn* [7] de Python, que da soporte para minería de datos en Python. Se ha seguido una aproximación orientada a objetos con tres módulos diferenciados:

- *data_handler.py*: implementa la clase *DATA_HANDLER*, cuyas instancias están diseñadas para almacenar los datos con los cuales se van a entrenar y evaluar los distintos modelos. Consta de diversos métodos que se encargan de comprobar la correcta estructura de la carpeta de datos indicada, cargar los datos y pre-procesarlos de tal manera que sean adecuados para el entrenamiento. En la etapa de pre-procesamiento se lleva a cabo una normalización de los datos, se traducen las variables cualitativas a variables cuantitativas utilizando la técnica de codificación *one-hot* (*One-Hot Encoding*), que incrementa el número de características que describen cada muestra, y por último se eliminan valores que no tengan sentido.
- *data_miners.py*: implementa una jerarquía de clases diseñada para ofrecer las funcionalidades necesarias para la aplicación de todas las técnicas de minería de datos evaluadas. La clase padre *DATA_MINER* es la base común para todas las técnicas de minería de datos e implementa métodos compartidos por todas las técnicas como, por ejemplo, entrenar el modelo, aplicar PCA (*Principal Component Analysis*), aplicar validación cruzada, etc. Las clases hijas *LOGREG* (regresión logística), *KNN* (*K-Nearest Neighbor*), *DTREE* (árboles de decisión), *GNB* (*Gaussian Naive Bayes*) y *MLPC* (*Multi-Layer perceptron classifier*) implementan las particularidades de cada una de las técnicas consideradas. Como se ha comentado previamente, se utiliza la implementación de scikit-learn. Esta jerarquía ha sido implementada de tal manera que para añadir una nueva técnica de minería de datos únicamente sea necesario desarrollar una subclase que gestione las particularidades que haya que considerar para dicha técnica.
- *ERTMD.py*: implementa la interfaz de usuario. Utiliza los archivos anteriormente mencionados para construir objetos que almacenen los datos, entrenar modelos y realizar las pruebas. Es el archivo que el usuario ejecuta.

2.2. Aplicación IDS-NET

IDS-NET representa un prototipo inicial de IDS dotado de una simple interfaz de línea de instrucciones, que permite a un usuario cargar datos y realizar predicciones sobre los mismos para detectar intrusiones. Esta herramienta, desarrollada en Python, realiza predicciones y recopila las conexiones que considera intrusiones. De esta forma, el usuario podrá evaluar si su sistema está en riesgo y tomar decisiones de seguridad, como podrían ser definir reglas en el cortafuegos o restringir el acceso a determinados puertos donde se ofrecen servicios abiertos al público. Los usuarios objetivo contemplados son usuarios no necesariamente expertos en el campo de minería de datos, pero que comprenden conceptos clave

como el significado de las métricas que se van a presentar, así como el significado de conjunto de entrenamiento y de prueba. El modo de uso de la aplicación puede resumirse en tres pasos: 1) el usuario carga los conjuntos de datos que se utilizan para entrenamiento y prueba; 2) el usuario indica los parámetros de control; y 3) el sistema indica al usuario las conexiones consideradas intrusiones.

Para el diseño de IDS-NET, nuevamente se ha seguido una aproximación orientada a objetos con Python. Además de los módulos *data_handler.py* y *data_miners.py* mencionados anteriormente (véase la Sección 2.1), se han definido dos nuevos módulos:

- *intrusion_detector.py*: implementa la clase *intrusion_detector*. Se trata de una clase que implementa un IDS. Utiliza los módulos *data_handler.py* y *data_miners.py* para cargar datos y realizar predicciones. Proporciona herramientas para las distintas funcionalidades de la aplicación.
- *IDS_NET.py*: implementa la interfaz de usuario. Pide al usuario los datos y parámetros necesarios y crea una instancia de la clase *intrusion_detector* para realizar predicciones. Es el archivo que el usuario ejecuta.

En la detección de intrusiones, se permite al usuario definir un *umbral de decisión* para realizar las predicciones, de forma que si la probabilidad estimada de intrusión está por debajo de dicho umbral se ignorará la predicción. Por otro lado, si no se ha definido ningún umbral entonces no se descartará la estimación de intrusión aunque se trate de una predicción con baja confianza. Por tanto, el umbral de decisión permite balancear la inconveniencia de falsos positivos frente al riesgo de falsos negativos (compromiso entre *precision* y *recall* [3]).

Además, es posible configurar la aplicación para que utilice varios algoritmos de minería de datos simultáneamente, es decir, varios modelos de aprendizaje para la detección de intrusiones. Cuando se configura IDS_NET para que use varios algoritmos, se puede indicar si se desean realizar *predicciones agregadas* o *predicciones desagregadas*. Si la predicción es agregada y no se ha definido un umbral de decisión, entonces la estimación sobre si hay una intrusión o no se determinará teniendo en cuenta la mayoría de las estimaciones realizadas por los modelos utilizados (es decir, globalmente se estima que hay una intrusión si con la mayoría de modelos utilizados se estima que hay una intrusión). Sin embargo, si la predicción es agregada y sí que se ha definido un umbral de decisión entonces la decisión final se hará comparando la media de las probabilidades de intrusión estimadas por los distintos modelos con el umbral de decisión. En caso de predicciones desagregadas se aplican los algoritmos de forma independiente.

2.3. Aplicación IDS-NET-DAEMON

IDS-NET-DAEMON es un prototipo inicial de IDS de tipo demonio de Linux (*Linux daemon*, controlado por *systemd*), que analiza el tráfico de red del sistema y detecta posibles intrusiones. Dado que se instala como un demonio, trabaja de forma transparente (sin molestar a las personas que utilizan y/o administran el sistema) mientras monitoriza las conexiones de red y detecta posibles

amenazas a la seguridad del sistema. Los usuarios objetivo contemplados son administradores de sistemas, más específicamente administradores de servidores con servicios abiertos al público y por tanto sujetos a múltiples conexiones y susceptibles de recibir ataques. Se entiende, por tanto, que los usuarios objetivo son expertos tanto en redes como en seguridad, aunque pueden no ser expertos en minería de datos. El modo de uso de esta aplicación es simple: el administrador debe configurar ciertos parámetros de control, arrancar el demonio y esperar las notificaciones de la aplicación. Una vez se detecten posibles amenazas, el sistema avisará al administrador del sistema para que pueda tomar las precauciones de seguridad necesarias.

La aplicación IDS-NET-DAEMON ha sido diseñada para funcionar a partir de un directorio propio que almacene tanto el código de la aplicación como los archivos de datos generados y proporcionados. Dicho directorio es */etc/ids* y deberá ser creado por la persona que administre el sistema. Los módulos que conforman la herramienta son los siguientes:

- *ids-daemon.service*: unidad de control (*unit file*) de servicio para *systemctl*, con objeto de controlar el demonio. Esta unidad especifica qué archivo de código Python debe ejecutarse, qué usuario debe ejecutarlo, de qué unidades depende y cierta información adicional para identificar el servicio. Dado que la aplicación funciona en segundo plano, se debe notificar al usuario de forma asíncrona; para ello, se decide utilizar la aplicación *mail*, habitualmente disponible en sistemas Linux, y que puede ser fácilmente utilizada desde el código de la aplicación. El usuario tendrá a su disposición el archivo */etc/ids/ids_intrusions.csv*, que contendrá las conexiones catalogadas como intrusiones.
- *Herramienta CICFlowMeter modificada*: se encarga de proporcionar la base para la monitorización y descripción de las conexiones de red. La monitorización de paquetes es un aspecto clave para el correcto funcionamiento de la aplicación. Para realizar la monitorización se ha hecho uso de la herramienta *CICFlowMeter* [6]. Sin embargo, el código abierto de la herramienta está diseñado para ser utilizado como aplicación por línea de instrucciones, no para formar parte de otro código. Por tanto, se ha tenido que realizar una pequeña modificación al código para satisfacer las necesidades de la aplicación IDS-NET-DAEMON. Las conexiones que se obtengan de esta monitorización deben tener el mismo formato que el de las conexiones descritas en los datos de entrenamiento, es decir, las conexiones deben estar representadas por las mismas características. Como CICFlowMeter fue la herramienta utilizada para generar el conjunto de datos SSH_FTP_ISCX (como se comenta posteriormente en la Sección 3.1), se pueden utilizar dichos datos como conjunto de entrenamiento. Esto implica que la herramienta en su concepción original detecta ataques de fuerza bruta a servicios SSH y FTP. Si el usuario quisiera usar conjuntos de datos de entrenamiento con un formato distinto, debería entonces modificar el subproceso de monitorización de la red.
- *ids_1.0.py*: archivo con el código de la aplicación. Se ha decidido reunir todo el código en un único archivo para evitar problemas por dependencias entre

archivos, mejorar la legibilidad y facilitar el estudio/modificación del código por parte de las personas que vayan a utilizar IDS-NET-DAEMON, en caso de que fuese necesario. De entre todas las técnicas de minería de datos consideradas, la aplicación IDS-NET-DAEMON utiliza únicamente el algoritmo de árboles de decisión, aunque podría modificarse para utilizar una técnica diferente. Se ha decidido utilizar una única técnica de clasificación con el objetivo de reducir el consumo de memoria RAM; en caso contrario, dado que la aplicación está diseñada para funcionar en modo continuo en segundo plano, el rendimiento en sistemas con poca memoria podría verse afectado. El funcionamiento básico de la aplicación se puede resumir en los siguientes pasos: 1) entrenamiento del modelo con los datos de entrenamiento; 2) captura de paquetes de red utilizando CICFlowMeter durante 60 segundos; 3) predicción de intrusiones sobre los datos capturados y notificación al usuario; 4) repetir indefinidamente desde el paso 2. Además, cuando funciona en modo de depuración, realiza el paso 1 y genera métricas de rendimiento.

Existen diversos parámetros que pueden utilizarse para configurar IDS-NET-DAEMON. Dos parámetros obligatorios son *INTERFACE* y *CONFIDENCE_THRESHOLD*, que representan el nombre de la interfaz de red sobre la cual se realiza la monitorización y el umbral de decisión para las predicciones, respectivamente. Además, existen parámetros optativos: *TRAIN_DATA_FILE* es el *path* al archivo .csv con los datos de entrenamiento, *SIM_INPUT_FILE* el *path* al archivo .csv con los datos etiquetados usados para realizar comprobaciones de rendimiento en modo de uso en depuración (que se comenta a continuación) y *FOUND_INTRUSIONS_FILE* es el *path* al archivo .csv donde se almacenarán las conexiones consideradas intrusiones por la aplicación.

Como funcionalidad secundaria de IDS-NET-DAEMON, se puede mencionar un *modo de uso en depuración*, que permite probar el correcto funcionamiento de la aplicación generando métricas de rendimiento a partir de un conjunto de datos de entrenamiento y de evaluación. En este caso el demonio no captura datos reales de la red de comunicaciones, sino que usa datos predefinidos y previamente etiquetados como ayuda a la depuración de su funcionamiento interno. Esto podría ayudar, por ejemplo, a detectar la necesidad de actualizar el modelo de datos o el algoritmo utilizado si el rendimiento observado no resultara adecuado.

3. Evaluación Experimental

En esta sección, evaluamos experimentalmente diversas técnicas de minería de datos para determinar cuáles pueden resultar apropiadas para la detección de potenciales intrusiones en redes de comunicaciones y presentamos también un escenario de prueba para la herramienta IDS-NET-DAEMON. En primer lugar, en la Sección 3.1, describimos los conjuntos de datos que hemos utilizado para la comparación de técnicas de minería de datos. Después, en la Sección 3.2, presentamos los resultados experimentales obtenidos en esa comparación. Final-

mente, en la Sección 3.3, se describe un escenario que hemos reproducido para comprobar el correcto funcionamiento de IDS-NET-DAEMON.

3.1. Conjuntos de Datos Utilizados para la Comparación de Técnicas de Minería de Datos

Hemos considerado dos conjuntos de datos, de los cuales hemos extraído diferentes subconjuntos para la evaluación experimental:

- *KDD99 Intrusion Detection Dataset* [11]. Se trata de un conjunto de datos originalmente diseñado y utilizado en *"The Third International Knowledge Discovery and Data Mining Tools Competition"*, donde se requería a los participantes obtener un modelo capaz de distinguir entre conexiones que representan intrusiones o ataques y conexiones normales. Los datos de este conjunto de datos fueron generados mediante simulaciones de distintos tipos de ataques en el entorno de una red militar estadounidense. Cada conexión, representada con 41 características, es una secuencia de paquetes TCP con distinta duración y se clasifica como normal o como el tipo específico de ataque que simula. Actualmente, este conjunto de datos ha ido perdiendo relevancia en el campo debido al proceso natural de "envejecimiento" [8] (nuevos ataques aparecen diariamente y en el contexto actual han ganado relevancia nuevos vectores de ataque). No obstante, a día de hoy sigue siendo un conjunto de datos relevante para realizar las primeras validaciones de modelos de detección propuestos.
  En nuestra evaluación experimental hemos considerado tres subconjuntos de este conjunto de datos: *KDD99 – 25K* (contiene alrededor de 25000 observaciones disponibles para entrenamiento), *KDD99 – 500K* (alrededor de 500000) y *KDD99 – 4.9M* (en torno a 4,9 millones).
- *Intrusion Detection Evaluation Dataset* (*CIC-IDS2017*) [5]. Se trata de un conjunto de datos diseñado por el Instituto Canadiense de Ciberseguridad (*Canadian Institute for Cybersecurity* o *CIC*). Su objetivo es suplir las carencias de los conjuntos de datos para IDSs existentes hasta la fecha.
  Concretamente, para nuestra evaluación experimental, decidimos trabajar con los datos de CIC-IDS2017 generados el 4 de julio de 2017, en los que se representan ataques de fuerza bruta sobre servidores *FTP* (*File Transfer Protocol*) y *SSH* (*Secure SHell*) realizados con la herramienta *Patator* [15]. Este subconjunto de datos (*SSH_FTP_ISCX*) contiene alrededor de 400000 observaciones disponibles para entrenamiento.

En este trabajo, no pretendemos distinguir entre distintos tipos de ataque, por lo que suprimimos esta información cuando está disponible en el conjunto de datos de entrada; en lugar de eso, simplemente consideramos una etiqueta que indica si la conexión es normal o representa una intrusión.

3.2. Comparación de Técnicas de Minería de Datos

En esta sección, presentamos resultados experimentales que comparan el rendimiento de diversas técnicas de minería de datos aplicadas en el contexto del

problema abordado en este artículo. Cuando nos referimos al rendimiento de las técnicas de minería de datos, a efectos prácticos, estamos evaluando la capacidad de los modelos entrenados para discernir entre conexiones normales e intrusiones. Se está, por tanto, evaluando una tarea de clasificación binaria. Utilizando datos previamente etiquetados, se pueden construir modelos alimentados con datos acerca de conexiones normales y maliciosas. Una conexión está constituida por una secuencia de paquetes (*communication flow*), en la que los paquetes fluyen desde la dirección IP origen a la dirección IP destino usando un determinado protocolo de comunicación y durante un intervalo de tiempo específico. Los modelos de datos entrenados reciben como entrada una única conexión y a partir de ésta, con independencia al resto de conexiones, realizan una predicción. Por tanto, la corrección de la clasificación depende de si el modelo es capaz de predecir de forma adecuada la clase de la conexión.

Para la evaluación, consideramos cuatro métricas de rendimiento habitualmente utilizadas para evaluar modelos de aprendizaje automático y aproximaciones de recuperación de información: *exactitud* (*accuracy*), *precisión* (*precision*), *recuperación* (*recall*) y *medida F1* (*F1-score*). Como método de evaluación, se utiliza la validación cruzada de k vías (*k-fold cross-validation*) [2,16], con k=10, que se utiliza habitualmente para medir el rendimiento de los modelos de aprendizaje automático. Se utiliza una implementación del método que preserva en cada *fold* el porcentaje de muestras positivas y negativas del conjunto original. A efectos de la clasificación binaria, consideramos que una predicción positiva representa una intrusión y una negativa representa una conexión normal.

Las técnicas de minería de datos evaluadas son las soportadas por la herramienta ERTMD que hemos desarrollado (descrita en la Sección 2.1): *LOGREG*, *KNN*, *DTREE*, *GNB* y *MLPC*. Además, consideramos dos técnicas de reducción de la dimensionalidad, que permiten reducir el número de características a considerar en las conexiones antes de entrenar los modelos: análisis de componentes principales (*Principal Component Analysis*, *PCA*) y la eliminación recursiva de características (*Recursive Feature Elimination*, *RFE*). Las pruebas se realizan utilizando las configuraciones por defecto de cada algoritmo en la implementación utilizada (biblioteca *scikit-learn* [7]), ya que los resultados de rendimiento son ya de por sí buenos y realizar un ajuste de hiper-parámetros estaba fuera del alcance del trabajo que pretendíamos realizar; por ejemplo, para la evaluación con la técnica de los k vecinos más cercanos, el valor de k por defecto es 5, por lo que se consideran los 5 vecinos más cercanos para decidir (por mayoría) el tipo de conexión (normal o maliciosa). En cuanto a la topología del clasificador por perceptrón multicapa escogida, la arquitectura propuesta está formada por cinco capas (la de entrada, tres capas internas con 50, 20 y 10 neuronas, respectivamente, y la capa de salida); con los tamaños de las capas internas fijados se busca respetar que el tamaño de cada capa esté entre el tamaño de la capa anterior y el de la siguiente.

Como entorno de ejecución de las pruebas se ha utilizado la máquina "cosmos1" del grupo de investigación COSMOS (`http://cos2mos.unizar.es`), que está dotada de altas prestaciones de RAM (256 GB), necesarias para realizar

las pruebas sobre los conjuntos de datos más grandes, y otras características de interés para el procesamiento (AS-1014S-WTRT SuperMicro AMD 1U H12 ULTRA 1 x AMD ROME 7302P UP 16C/32T 3.0G 128M 155W 4094 HF). El uso de esta máquina permite realizar pruebas grandes con un bajo coste temporal.

En la Figura 2, se resumen los resultados obtenidos para la métrica *F1-score* con los algoritmos basados en la utilización de un árbol de decisión y un perceptrón multicapa, que son los que han obtenido, en general, los mejores resultados. Observando las tres primeras parejas de barras, se aprecia cómo al añadir más datos para entrenar los algoritmos generan mejores resultados, pero la ganancia obtenida no es excesivamente grande. El mayor incremento en la medida *F1-score* se obtuvo al pasar de utilizar MLPC sobre el conjunto de datos *KDD99 – 25K* a utilizar el conjunto de datos *KDD99 – 5M* (incremento de 0,0254). Nótese que se ha iniciado el eje Y en el valor 96 (representando la métrica *F1-score* en tanto por ciento) para que se aprecien mejor las diferencias. Vale la pena señalar que los protocolos seleccionados son bien conocidos, reciben ataques de fuerza bruta con frecuencia y suelen ser bastante llamativos, lo que probablemente influye en los buenos resultados de detección obtenidos.

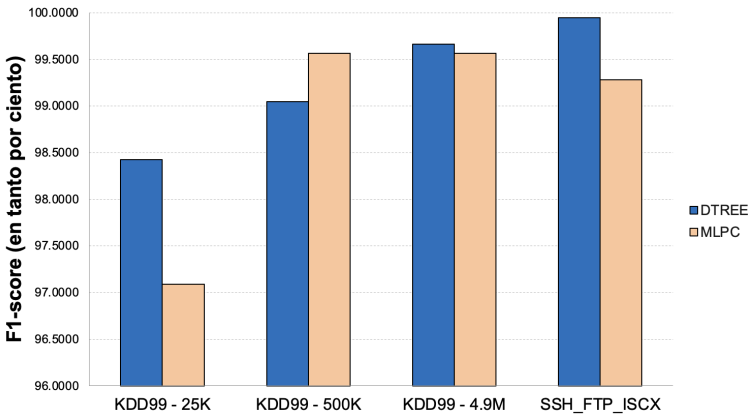

Figura 2. *F1-score* (en tanto por ciento) utilizando DTREE y MLPC

Todas las técnicas salvo la de los k-vecinos más cercanos incurren en un tiempo de predicción muy pequeño, lo que las hace adecuadas para la predicción en tiempo real. En cuanto al tiempo de entrenamiento, todas las técnicas salvo la de los k-vecinos más cercanos sufren un incremento del coste temporal en la etapa de entrenamiento al incrementarse el volumen de observaciones del conjunto de datos; la técnica de los k-vecinos más cercanos es un algoritmo perezoso (*lazy algorithm*) y, por tanto, todo su coste temporal se concentra en la etapa de predicción, en la cual sí se puede apreciar que incrementa el tiempo a medida que lo hace el volumen de observaciones del conjunto de datos. Además, para el conjunto *KDD99 – 4.9M* no se lograron ejecutar pruebas con la técnica de los

k-vecinos más cercanos, debido a una tendencia de crecimiento exponencial de su coste temporal con el tamaño del conjunto de datos; por ello, se consideró que dicha técnica no era apropiada para ese conjunto de datos. Como ejemplo, en la Figura 3 se muestran los tiempos de ejecución para el conjunto de datos *KDD99 – 4.9M* (algunos tiempos son tan pequeños que resultan indistinguibles en la gráfica). GNB ha incurrido con diferencia en el menor tiempo total de ejecución. Dado que el rendimiento de detección que se obtiene en la mayoría de las pruebas con GNB no es malo (aunque sí inferior al del resto de técnicas; por ejemplo, véase la Tabla 1), se podría utilizar la técnica GNB si se desea sacrificar un poco de rendimiento de detección a cambio de un coste temporal muy bajo.

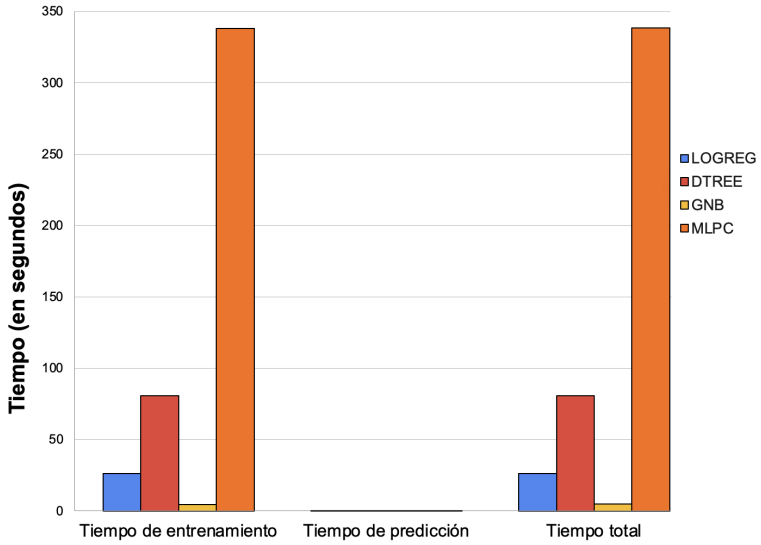

Figura 3. Tiempo medio de entrenamiento y predicción con *KDD99 – 4.9M*

	Accuracy	*Precision*	*Recall*	*F1-score*
LOGREG	99,9960	100,0000	97,5000	98,5714
kNN	99,9841	93,3333	98,3333	95,0909
DTREE	99,9921	100,0000	97,0833	98,4242
GNB	99,9643	100,0000	81,0000	88,3348
MLPC	99,9921	100,0000	95,0000	97,0909

Tabla 1. Rendimiento con *KDD99 – 25K* y sin reducción de dimensionalidad

En cuanto al uso de técnicas de reducción de dimensionalidad, en los experimentos hemos visto que resultan ser en general efectivas, incluso generando

mejores resultados en algunos escenarios. Sin embargo, usadas en conjunción con GNB se observaron resultados desfavorables en ciertos casos, observándose un incremento del *recall* a costa de la *precision*, llevando a un *F1-score* bajo.

Por limitaciones de espacio, no es posible recoger aquí todos los resultados experimentales obtenidos, pero se ofrecen más detalles en la página web del proyecto [4]. A. partir de los distintos experimentos realizados, se concluye que las técnicas de minería de datos consideradas son bastante efectivas para detectar intrusiones y se ha comprobado además que la herramienta ERTMD (descrita en la Sección 2.1) es útil para realizar este tipo de comparativas.

3.3. Escenario de Prueba

Para probar la aplicación IDS-NET-DAEMON (descrita en la Sección 2.3) se ha diseñado un entorno de pruebas que simula una secuencia de ataque de intento de inicio de sesión para un servicio SSH por fuerza bruta, que se resume en la Figura 4. La arquitectura del entorno está constituida por dos máquinas en la misma red: la máquina atacante con un sistema operativo Kali Linux y la máquina atacada con un sistema operativo Ubuntu. La máquina atacante utiliza la herramienta *patator* [15], que permite realizar distintos tipos de ataque por fuerza bruta. La máquina atacada tiene un servicio SSH disponible y además está ejecutando la aplicación IDS-NET-DAEMON.

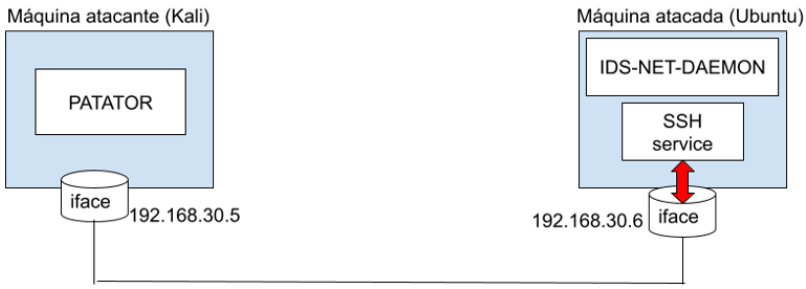

Figura 4. Diagrama del entorno de pruebas para la aplicación IDS-NET-DAEMON

Así pues, la máquina atacante realiza un ataque de fuerza bruta utilizando la herramienta *patator* y la prueba se considera superada si la aplicación notifica mediante la herramienta *mail* al usuario administrador (*root*) de un intento de ataque. Este escenario, que representa un entorno de producción simulado, nos ha servido como prueba de concepto para verificar que potencialmente podría instalarse IDS-NET-DAEMON en un entorno real.

4. Conclusiones y Trabajo Futuro

En este trabajo hemos abordado el problema de detección de potenciales intrusiones en redes de comunicaciones. Para ello, hemos comparado experi-

mentalmente diversas técnicas de minería de datos que pueden utilizarse en la detección. Además, hemos desarrollado diversas herramientas y aplicaciones de apoyo que pueden utilizarse, tanto para comparar el rendimiento de diversas técnicas de minería de datos como para realizar la propia detección de intrusiones. Si bien el trabajo que hemos desarrollado es de suponer que está lejos de las funcionalidades de las aplicaciones comerciales existentes, prueba que incluso el uso de técnicas no especialmente sofisticadas puede servir de ayuda para la detección de ataques, y por tanto de apoyo a la toma de decisiones, en diversos escenarios. En la página web del proyecto [4] está disponible todo el código desarrollado, así como diversas capturas de pantalla, vídeos y detalles adicionales de la evaluación experimental realizada.

Como trabajo futuro, podría considerarse extender el trabajo aquí realizado para identificar no sólo las conexiones maliciosas sino también el tipo de ataque o intrusión, lo que requeriría realizar una clasificación multiclase. Por otro lado, podría abordarse el desarrollo de interfaces gráficas para todas las herramientas desarrolladas, con objeto de mejorar la experiencia de usuario. Otras posibles mejoras incluyen la consideración de otros algoritmos de aprendizaje automático [1,13] e incorporar una capacidad de aprendizaje continuo que permita ir mejorando los modelos conforme se disponen de más datos de entrenamiento.

Agradecimientos Este trabajo se ha desarrollado como parte del proyecto de I+D+i PID2020-113037RB-I00, financiado por MCIN/AEI/ 10.13039/50110001-1033 y PID2019-104358RB-I00 financiado por MCINN. Además de esos proyectos (proyecto NEAT-AMBIENCE, proyecto DL-AGEING), se agradece también el apoyo del Departamento de Ciencia, Universidad y Sociedad del Conocimiento del Gobierno de Aragón (última referencia de grupo: T64_23R, grupo COSMOS). La investigación presentada en este artículo se realizó en el marco del trabajo final de grado en Ingeniería Informática del primer autor, dirigido por los otros dos autores del artículo, en colaboración con el proyecto NEAT-AMBIENCE.

Referencias

1. Abdallah, E.E., Eleisah, W., Otoom, A.F.: Intrusion detection systems using supervised machine learning techniques: A survey. Procedia Computer Science **201**, 205–212 (2022). https://doi.org/10.1016/j.procs.2022.03.029
2. Arlot, S., Celisse, A.: A survey of cross-validation procedures for model selection. Statistics Surveys **4**(none) (January 2010). https://doi.org/10.1214/09-ss054
3. Buckland, M., Gey, F.: The relationship between recall and precision. Journal of the American Society for Information Science **45**(1), 12–19 (January 1994). https://doi.org/10.1002/(sici)1097-4571(199401)45:1<12::aid-asi2>3.0.co;2-l
4. Calvera, A., Ilarri, S., Tricas, F.: IDSdm: Intrusion Detection Systems based on Data Mining. http://webdiis.unizar.es/~silarri/prot/IDSdm (2023), last access: June 20, 2023.
5. Canadian Institute of Cybersecurity: Intrusion Detection Evaluation Dataset (CIC-IDS2017). https://www.unb.ca/cic/datasets/ids-2017.html (2017), last access: June 20, 2023.

6. Canadian Institute of Cybersecurity: CICFlowMeter. https://github.com/CanadianInstituteForCybersecurity/CICFlowMeter (2018–2020), last access: June 20, 2023.
7. Cournapeau, D., Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V.: scikit-learn. https://scikit-learn.org (2016–2023), last access: June 20, 2023.
8. Creech, G., Hu, J.: Generation of a new IDS test dataset: Time to retire the KDD collection. In: IEEE Wireless Communications and Networking Conference (WCNC 2013). IEEE (April 2013). https://doi.org/10.1109/wcnc.2013.6555301
9. Darktrace: Darktrace. https://darktrace.com, last access: June 20, 2023.
10. Hu, Y., Panda, B.: A data mining approach for database intrusion detection. In: ACM Symposium on Applied Computing (SAC 2004). pp. 711–716. ACM, New York, NY, USA (2004). https://doi.org/10.1145/967900.968048
11. International Conference on Knowledge Discovery and Data Mining-99: KDD99 Intrusion Detection Dataset. https://www.kaggle.com/datasets/galaxyh/kdd-cup-1999-data (1999), last access: June 20, 2023.
12. Kolbe, P.R., Morrow, M.R., Zabierek, L.: The cybersecurity risks of an escalating Russia-Ukraine conflict. https://hbr.org/2022/02/the-cybersecurity-risks-of-an-escalating-russia-ukraine-conflict (February 2022), Harvard Business Review. Last access: June 20, 2023.
13. Liu, H., Lang, B.: Machine learning and deep learning methods for intrusion detection systems: A survey. Applied Sciences **9**(20), 4396 (October 2019). https://doi.org/10.3390/app9204396
14. Lohrmann, D.: Global cybersecurity ramifications from the war in Ukraine. https://www.govtech.com/blogs/lohrmann-on-cybersecurity/global-cybersecurity-ramifications-from-the-war-in-ukraine (March 2022), government Technology. Last access: June 20, 2023.
15. Macke, S.: Patator. https://github.com/lanjelot/patator (2011–2022), last access: June 20, 2023.
16. Refaeilzadeh, P., Tang, L., Liu, H.: Cross-validation. In: Encyclopedia of Database Systems, pp. 677–684. Springer (2018). https://doi.org/10.1007/978-1-4614-8265-9_565
17. Sharif, M., Urakawa, J., Christin, N., Kubota, A., Yamada, A.: Predicting impending exposure to malicious content from user behavior. In: ACM SIGSAC Conference on Computer and Communications Security (CCS 2018). pp. 1487–1501. ACM, New York, NY, USA (2018). https://doi.org/10.1145/3243734.3243779
18. Shen, Y., Mariconti, E., Vervier, P.A., Stringhini, G.: Tiresias: Predicting security events through deep learning. In: ACM SIGSAC Conference on Computer and Communications Security (CCS 2018). pp. 592–605. ACM, New York, NY, USA (2018). https://doi.org/10.1145/3243734.3243811
19. Shen, Y., Mariconti, E., Vervier, P.A., Stringhini, G.: Tiresias: Predicting Security Events Through Deep Learning. In: ACM SIGSAC Conference on Computer and Communications Security (CCS). pp. 592–605. ACM (2018). https://doi.org/10.1145/3243734.3243811
20. Universidad de Zaragoza: Next-gEnerATion dAta Management to foster suitable Behaviors and the resilience of cItizens against modErN ChallEnges (NEAT-AMBIENCE). http://webdiis.unizar.es/~silarri/NEAT-AMBIENCE (2021–2025), project funded by the Agencia Estatal de Investigación (PID2020-113037RB-I00 / AEI / 10.13039/501100011033). Last access: June 20, 2023.