UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Estudio y detección de ciberataques en red mediante redes neuronales

Andrei Constantin
Tutor: Luis Fernando Lago Fernández

Julio 2020

Estudio y detección de ciberataques en red mediante redes neuronales

AUTOR: Andrei Constantin
TUTOR: Luis Fernando Lago Fernández

Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Julio de 2020

Resumen

Internet se ha convertido en la pieza clave de la sociedad digital en la que vivimos. Cada vez son más los sectores económicos que incorporan dispositivos conectados a la red para mejorar su producción de bienes o servicios. Debido a esto, las infraestructuras críticas se ven expuestas al peligro que acecha detrás de internet, lo cual, unido al gran volumen de tráfico que manejan las redes modernas, provoca que sea más complicado analizar todo lo que pasa a través de estas. El uso de la inteligencia artificial junto con otras medidas de seguridad, pueden ser la solución para proteger de forma más eficaz todos los dispositivos conectados a cualquier red.

En este Trabajo de Fin de Grado se presenta un análisis de posibles soluciones para la detección de ciberataques en red mediante aprendizaje profundo. En concreto, se analiza la eficacia de detección de tres tipos de redes de aprendizaje profundo, las redes neuronales profundas como el perceptrón multicapa, los autoencoders y las redes recurrentes. Se han elegido estos modelos, pues durante el trabajo se va a realizar un estudio de la temporalidad de los datos, para comprobar si la aportación de contexto a la hora de realizar clasificaciones del tráfico en la red mejora los resultados.

Para el entrenamiento de estas redes se hace uso del dataset público CSE-CIC-IDS 2018, que incluye numerosos tipos de ciberataques muy comunes en la actualidad, como denegación de servicio, fuerza bruta o ataques web. Los resultados de la clasificación de estos ataques varían según el modelo, consiguiendo entre todos detectar al menos un 85% de las ocurrencias de cada ataque. El mejor modelo realizado en este trabajo presenta un índice de detección cercano al 99%.

Palabras clave

Aprendizaje profundo, ciberseguridad, redes neuronales recurrentes, autoencoders.

Abstract

Internet has become a key piece in the digital society we live in. More and more economic sectors are incorporating devices connected to the network to improve their production of goods or services. Due to this, the critical infrastructures are exposed to the danger that lurks behind the internet, which, together with the large volume of traffic handled by modern networks, is increasing the dificulty of analyzing everything that passes through the networks. The use of artificial intelligence, along with other security measures, can be the solution to more effectively protect all the devices connected to any network.

This Bachelor Thesis presents an analysis of possible solutions for detecting network cyber attacks through deep learning. Specifically, it analyzes the effectiveness of detection of three types of deep learning neural network, such as the multilayer perceptron, autoencoders, and recurrent neural networks. These models have been chosen, since during the thesis a study of the temporality of the data is carried out, to verify whether the contribution of context when making traffic classifications in the network improves the results.

For the training of these networks, the public data set CSE-CIC-IDS 2018 is used, which includes several types of cyber attacks very common today, such as denial of service, brute force or web attacks. The results of the classification of these attacks vary according to the model, managing together to detect at least 85% of the occurrences of each attack. The best model carried out in this work has a detection rate close to 99%.

Keywords

Deep learning, cybersecurity, recurrent neural network, autoencoders.

Agradecimientos

A Luis y Paco por haberme guiado en esta divertida aventura, por vuestro apoyo y por el esfuerzo que habéis realizado para ayudarme a sacar este trabajo adelante.

A mi familia, por haberme acompañado durante este viaje y por haberme ayudado a convertirme en la persona que soy hoy en día. Especialmente a mi madre, gracias por el sacrificio que has realizado durante estos últimos 15 años, gran parte de lo que soy se lo debo a tu esfuerzo.

A mis compañeros de universidad, gracias por todos los momentos que me habéis regalado, sin vuestra compañía en las largas horas en los laboratorios y la diversión de los momentos en el césped, estos cuatro años no hubiesen valido tanto la pena.

A mis compañeros de trabajo, especialmente a Fran, Roberto, Enrique, María, Carmen y Juan Carlos. Sin los cafés de cada mañana, y sin vuestros consejos y enseñanzas, no hubiese conseguido dar el salto al mundo profesional con tanta ilusión.

A Rubén, Ángeles y Andrea, sin vuestra compañía durante los viajes de cada mañana no hubiese encontrado motivación para haberme levantado de la cama. Gracias por haberme ayudado a dar mis primeros pasos.

A mis amigos de siempre, Andrés, Víctor y María. Gracias por mantenerme en pie cuando más lo necesitaba, y gracias por hacerme sentir que siempre tengo un hombro donde apoyarme.

Y por último y especialmente, quiero darle las gracias a Marta, gracias por haberme acompañado en este bonito viaje, por haberme aguantado durante estos últimos dos años y medio, gracias por haber estado ahí en todo momento, lo único de lo que me arrepiento es de no haberte conocido antes.

INDICE DE CONTENIDOS

1 INTRODUCCIÓN 1

1.1 MOTIVACIÓN 1
1.2 OBJETIVOS 1
1.3 HERRAMIENTAS Y RECURSOS 1
1.4 ORGANIZACIÓN DE LA MEMORIA 2

2 ESTADO DEL ARTE 3

2.1 INTRODUCCIÓN 3
2.2 DATASETS Y ESTUDIOS PREVIOS REALIZADOS 3
2.3 TIPOS DE APRENDIZAJE 4
2.3.1 Modelos supervisados 4
2.3.2 Modelos semi supervisados 4
2.3.3 Modelos no supervisados 5
2.4 ARQUITECTURAS DE DEEP LEARNING 5
2.4.1 Deep Neural Networks 5
2.4.2 Redes recurrentes 5
2.4.3 Redes convolucionales 5
2.4.4 Autoencoders 5
2.4.5 Deep Belief Networks 6
2.5 TIPOS DE CIBERATAQUES 6

3 DISEÑO Y DESARROLLO 9

3.1 INTRODUCCIÓN 9
3.2 DATASET UTILIZADO 9
3.3 EXPLORACIÓN DE LOS DATOS 11
3.3.1 Exploración temporal 12
3.3.2 Atributos más relevantes 14
3.4 PROCESAMIENTO DE LOS DATOS 15
3.4.1 Procesos comunes a todos los modelos 16
3.4.2 Procesos en común de los modelos supervisados 17
3.4.3 Procesos exclusivos de los modelos no supervisados 17
3.4.4 Procesos exclusivos de las redes recurrentes 17
3.5 MODELOS GENERADOS 18
3.5.1 Perceptrón multicapa 18
3.5.2 Autoencoder 19
3.5.3 Redes recurrentes 19

4 PRUEBAS Y RESULTADOS 21

4.1 DESCRIPCIÓN DE LAS PRUEBAS REALIZADAS 21
4.2 RESULTADOS OBTENIDOS CON CLASIFICADORES 21
4.2.1 Resultados perceptrón multicapa 22
4.2.2 Resultados redes recurrentes 23
4.3 RESULTADOS OBTENIDOS CON EL AUTOENCODER 25
4.4 ANÁLISIS DE LOS RESULTADOS 28

5 CONCLUSIONES Y TRABAJO FUTURO 31

5.1 CONCLUSIONES 31
5.2 TRABAJO FUTURO 31

REFERENCIAS 33

GLOSARIO ..... - 1 -

ANEXOS ..... **- 2 -**

A ATRIBUTOS DEL DATASET CIC-IDS-2018 ..... - 2 -
B EXPLORACIONES TEMPORALES DATASET CIC-IDS-2018. ..... - 4 -
C RESULTADOS ..... - 11 -

INDICE DE FIGURAS

FIGURA 1: ENTORNO EN LA NUBE DE AMAZON WEB SERVICES OBTENIDA DE LA REFERENCIA [23]. 11

FIGURA 2: HISTOGRAMA DEL TRÁFICO. 13

FIGURA 3: DISTRIBUCIÓN DE LAS CANTIDADES DE FLUJO A LO LARGO DE LOS MINUTOS DE LA HORA. 13

FIGURA 4: HISTOGRAMA DE LOS SEGUNDOS DE CADA FLUJO. 14

FIGURA 5: IMPORTANCIAS DE LOS ATRIBUTOS. 15

FIGURA 6: VARIACIÓN DE LAS MÉTRICAS RESPECTO AL UMBRAL. 26

FIGURA 7: HISTOGRAMA DE ERRORES DE LOS FLUJOS PREDICHOS POR EL AUTOENCODER 27

FIGURA 8: HISTOGRAMA DE LOS DIFERENTES FLUJOS EN EL DATASET DURANTE LAS 13 Y LAS 15 HORAS. ..... - 4 -

FIGURA 9: HISTOGRAMA DE LOS FLUJOS DEL ATAQUE DOS-HULK ..... - 4 -

FIGURA 10: HISTOGRAMA DE LOS FLUJOS DEL ATAQUE DOS-SLOWHTTPTEST ..... - 5 -

FIGURA 11: HISTOGRAMA DE LOS FLUJOS DEL ATAQUE DOS-SLOWLORIS ..... - 5 -

FIGURA 12: HISTOGRAMA DE LOS FLUJOS DEL ATAQUE FTP-BRUTEFORCE ..... - 6 -

FIGURA 13: HISTOGRAMA DE LOS FLUJOS DEL ATAQUE INFILTERATION ..... - 6 -

FIGURA 14: HISTOGRAMA DE LOS FLUJOS DEL ATAQUE SQL INJECTION ..... - 7 -

FIGURA 15: HISTOGRAMA DE LOS FLUJOS DEL ATAQUE SSH-BRUTEFORCE ..... - 7 -

FIGURA 16: HISTOGRAMA DE LOS FLUJOS DEL ATAQUE BOT ..... - 8 -

FIGURA 17: HISTOGRAMA DE LOS FLUJOS DEL ATAQUE BRUTE FORCE -WEB ..... - 8 -

FIGURA 18: HISTOGRAMA DE LOS FLUJOS DEL ATAQUE BRUTE FORCE -XSS ..... - 9 -

FIGURA 19: HISTOGRAMA DE LOS FLUJOS DEL ATAQUE DDOS-HOIC ..... - 9 -

FIGURA 20: HISTOGRAMA DE LOS FLUJOS DEL ATAQUE DDOS-LOIC UDP ..... - 10 -

FIGURA 21: HISTOGRAMA DE LOS FLUJOS DEL ATAQUE DOS-GOLDENEYE ..... - 10 -

FIGURA 22: MATRIZ DE CONFUSIÓN DEL PERCEPTRÓN MULTICAPA. ..... - 11 -

FIGURA 23: EVOLUCIÓN DE LA FUNCIÓN OBJETIVO RESPECTO A LAS ÉPOCAS DE ENTRENAMIENTO DEL PERCEPTRÓN MULTICAPA. ..... - 11 -

FIGURA 24: MATRIZ DE CONFUSIÓN DEL AUTOENCODER. ..... - 12 -

FIGURA 25: EVOLUCIÓN DE LA FUNCIÓN OBJETIVO RESPECTO A LAS ÉPOCAS DE ENTRENAMIENTO DEL AUTOENCODER. ..... - 12 -

FIGURA 26: MATRIZ DE CONFUSIÓN DE LA RED LSTM *STATELESS*. ..... - 13 -

FIGURA 27: EVOLUCIÓN DE LA FUNCIÓN OBJETIVO RESPECTO A LAS ÉPOCAS DE ENTRENAMIENTO DE LA RED LSTM *STATELESS*. ..... - 13 -

FIGURA 28: MATRIZ DE CONFUSIÓN DE LA RED LSTM *STATEFUL*. ..... - 14 -

FIGURA 29: EVOLUCIÓN DE LA FUNCIÓN OBJETIVO RESPECTO A LAS ÉPOCAS DE ENTRENAMIENTO DE LA RED LSTM *STATEFUL*. ..... - 14 -

INDICE DE TABLAS

TABLA 1: TÉCNICAS DE APRENDIZAJE PROFUNDO PARA LA DETECCIÓN DE INTRUSIONES RED, ESTUDIO EN EL QUE SE UTILIZAN Y EL DATASET UTILIZADO. NOTA: TABLA ADAPTADA DEL ARTÍCULO [14]. 4

TABLA 2: CANTIDAD DE FLUJOS ORGANIZADOS POR CLASE EN CADA FICHERO DEL DATASET. 10

TABLA 3: CLASES DE LOS FLUJOS PRESENTES EN EL DATASET Y CANTIDAD PRESENTE DE CADA UNO. 12

TABLA 4: TIPOS DE PREPROCESAMIENTO REALIZADOS, MOSTRADOS POR ORDEN DE UTILIZACIÓN.15

TABLA 5: CANTIDAD DE FLUJOS TOTALES, CON VALORES NAN O INFINITO Y CANTIDAD TOTAL TRAS EL FILTRADO. 16

TABLA 6: CANTIDAD DE SERIES GENERADAS SEGÚN EL TAMAÑO DE LA SERIE ELEGIDO. 18

TABLA 7: REPORTE DE CLASIFICACIÓN DEL PERCEPTRÓN MULTICAPA. 23

TABLA 8: REPORTE DE CLASIFICACIÓN DE LA RED LSTM *STATELESS*. 24

TABLA 9: REPORTE DE CLASIFICACIÓN DE LA RED LSTM *STATEFUL*. 25

TABLA 10: REPORTE DE CLASIFICACIÓN DEL AUTOENCODER. 27

TABLA 11: RESUMEN DE LOS RESULTADOS OBTENIDOS PARA LA PRECISIÓN. NOTA: LOS RESULTADOS MARCADOS CON * REPRESENTAN DATOS CON SESGO. 28

TABLA 12: RESUMEN DE LOS RESULTADOS OBTENIDOS PARA EL RECALL. NOTA: LOS RESULTADOS MARCADOS CON * REPRESENTAN DATOS CON SESGO..... 28

1 Introducción

1.1 Motivación

Con el paso de los años, el incremento de la rentabilidad del cibercrimen ha aumentado, posicionándose como el "negocio" más rentable para los criminales [1] superando al narcotráfico o la trata de blancas. Esto supone que la popularidad del cibercrimen aumente, por lo tanto, es de alto interés la introducción de tecnologías avanzadas como el Deep learning, que pueden llegar a producir una revolución dentro del sector y de esta forma ayudar a mejorar la situación actual.

Los avances en las nuevas ciber amenazas son constantes, destacan la aparición de nuevos vectores de ataque que rompen todos los esquemas tradicionales, como puede ser el caso del malware mutante que es capaz de evitar los antivirus tradicionales basados en firmas [2]; o la aparición de amenazas que son capaces de comprometer las redes "seguras" de gran parte de las corporaciones del mundo como fue el caso de WannaCry [3] en 2017.

Resulta evidente que los avances producidos en las amenazas están generando una respuesta en los mecanismos detección, que están sufriendo una transición desde los modelos tradicionales, basados en firmas y reglas manuales, hacia mecanismos más complejos, como pueden ser los basados en comportamiento. Muchas soluciones del mercado están desarrollando modelos basados en comportamiento para la detección de estos nuevos ataques, como puede ser el caso de Cylance [4] o WatchGuard [5] en la detección de malware y DarkTrace [6] o ExtraHop [7] en la detección de ataques en red.

1.2 Objetivos

El objetivo de este TFG consiste en realizar un estudio de técnicas basadas en redes neuronales para detectar ciberataques en la red, utilizando como dato una forma condensada del tráfico conocida como *flow*. Estos *flows* o flujos son etiquetados diferenciando entre flujos benignos y distintos tipos de ciberataques. Se analizarán diferentes arquitecturas de redes neuronales para comprobar cuál es la más efectiva para clasificar los flujos, y se estudiará si la aportación de contexto a la red, en forma de series temporales, ofrece algún beneficio frente a clasificaciones puntuales.

Estos objetivos se pueden dividir en subobjetivos, todos serán enumerados a continuación.

- Estudiar el estado del arte de la detección de ciberataques en red.
- Estudiar los datasets más frecuentes y modelos utilizados previamente.
- Tratar de identificar qué modelo de Deep learning obtiene mejores resultados.
- Estudiar si el tiempo puede ayudar a la detección de ciberataques, y en caso afirmativo realizar una primera exploración con la construcción de un modelo que se aproveche de la temporalidad.

1.3 Herramientas y recursos

El trabajo ha sido desarrollado en el entorno Google Colab [8], que permite ejecutar y programar en Python haciendo uso de Jupyter Notebooks [9]. Este entorno ha sido conectado

a Google Drive [10], pues aquí han sido almacenados los datos y los modelos de las redes generadas.

Para facilitar la creación de los modelos, el procesamiento de los datos y de los resultados, y la generación de las gráficas, se ha hecho uso de software de terceros, en concreto:

- Pandas para el procesamiento de los datos.
- Scikit-Learn para el procesamiento de los datos y resultados.
- Keras para la creación de los modelos.
- Seaborn para la elaboración de las gráficas.

1.4 Organización de la memoria

La memoria consta de los siguientes capítulos:

- Estado del arte, donde se exponen los estudios y trabajos realizados previamente, junto con la actualidad y las tendencias referentes a las distintas disciplinas que componen este trabajo.

- Diseño y desarrollo, donde se expone el proceso técnico desarrollado para resolver el problema objetivo.

- Pruebas y resultados, donde se muestran las métricas más relevantes de los modelos desarrollados

- Conclusiones y trabajo futuro, donde se resumen los avances realizados, se contrastan los objetivos propuestos con los resultados obtenidos, y, además, se exponen los siguientes pasos que podrían realizarse a partir del estudio y trabajo realizados.

2 Estado del arte

2.1 Introducción

Debido a la variedad de ramas y disciplinas exploradas durante la realización de este trabajo, resulta interesante realizar un estudio del estado del arte de las relaciones de cada rama con el mundo de la ciberseguridad. Por lo tanto, el estado del arte quedará dividido en:

- Datasets más utilizados y estudios previos realizados.
- Tipos de aprendizaje
- Arquitecturas de Deep learning.
- Tipos de ciberataques

2.2 Datasets y estudios previos realizados

El tráfico de una red está formado por una gran cantidad de paquetes que navegan a través de ella. Estos paquetes contienen grandes cantidades de información, que se agrupan en siete capas diferentes, conocidas como el modelo OSI [11]. Dentro de este modelo, las capas superiores son específicas para las distintas aplicaciones y conexiones a alto nivel, por otro lado, las capas inferiores incluyen los estándares más extendidos y utilizados por la mayor parte de paquetes, pues son necesarios para el enrutamiento y transporte de los paquetes a través de las redes, como por ejemplo los protocolos TCP/IP [12] o UDP [13].

Para generar modelos que puedan garantizar la detección de ciberataques en red, analizar los datos comunes a todos los paquetes parece, en primera instancia, la mejor idea. Esta idea se ve reforzada por la extensión del tráfico cifrado en internet, pues estos datos no pueden ser extraídos ni analizados pues la información transmitida está oculta. Además, en 2018 el tráfico cifrado ya suponía el 50% del tráfico de internet [14] aumentando año tras año.

Este hecho provoca que los datos de las capas 4 en adelante sean en muchos casos inservibles. Debido a esto, los esfuerzos de este trabajo se centrarán en el análisis de los datos provenientes de la capa 3 y los datos no cifrados de la capa 4. Para obtener datos de estas capas, existen herramientas y protocolos que resumen y agrupan de forma muy eficiente todo el tráfico intercambiado entre las distintas conexiones de una red, entre los que destacan los Intrusion Detection Systems (IDS) [15]. Estos sistemas en muchos casos aportan información añadida, por lo que se centrarán los esfuerzos en estudiar los datasets y trabajos que empleen datos provenientes de estas fuentes.

El número de datasets públicos con datos de red no ha sufrido mucho dinamismo con el paso de los años, gran parte de los estudios publicados siguen utilizando datasets antiguos o con poca cantidad de ataques. Esto se puede observar en la tabla resumen Tabla 1.

Técnica de deep learning utilizada	IDS	Dataset utilizado	No. de veces citado (a día 30/05/2019)
Deep neural network	Tang et al. (2016)	NSL-KDD dataset	110
Deep neural network	Potluri and Diedrich (2016)	NSL-KDD dataset	37
Deep neural network	Kang and Kang (2016)	Vehicular network communication	137
Deep neural network	Zhou et al. (2018)	4 types of attacks (DOS R2L U2R and PROBING)	0
Deep neural network	Feng et al. (2019)	KDD Cup 1999 dataset	1
Deep neural network	Zhang et al. (2019)	KDD Cup 1999 dataset	0
Deep neural network	Roy et al. (2017)	KDD Cup 1999 dataset	23
Feed forward deep neural network	Kasongo and Sun (2019)	NSL-KDD dataset	0
Recurrent neural network	Kim et al. (2016)	KDD Cup 1999 dataset	86
Recurrent neural network	Yin et al. (2017)	NSL-KDD dataset	100
Recurrent neural network	Tang et al. (2018)	NSL-KDD dataset	9
Recurrent neural network	Jiang et al. (2018)	NSL-KDD dataset	22
Convolutional neural network	Basumallik et al. (2019)	IEEE-30 bus and IEEE-118 bus	1
Convolutional neural network	Fu et al. (2016)	Credit card transaction data	47
Convolutional neural network	Zhang et al. (2018)	Commercial bank B2 online transaction data	3
Convolutional neural network	Feng et al. (2019)	KDD Cup 1999 dataset	1
Convolutional autoencoder	Yu et al. (2017)	Contagio-CTU-UNB dataset	17
Restricted Boltzmann machine	Alrawashdeh and Purdy (2016)	KDD Cup 1999 dataset	35
Restricted Boltzmann machine	Aldwairi et al. (2018)	ISCX dataset	4
Restricted Boltzmann machine	Fiore et al. (2013)	KDD Cup 1999 dataset	176
Restricted Boltzmann machine	Salama et al. (2011)	NSL-KDD dataset	96
Restricted Boltzmann machine	Gao et al. (2014)	KDD Cup 1999 dataset	69
Restricted Boltzmann machine	Alom et al. (2015)	NSL-KDD dataset	59
Restricted Boltzmann machine	Yang et al. (2017)	Real online network traffic	16
Restricted Boltzmann machine	Otoum et al. (2019)	KDD Cup 1999 dataset	3
Deep belief network	Zhao et al. (2017)	KDD Cup 1999 dataset	13
Deep auto-encoder	Shone et al. (2018)	NSL-KDD dataset	66
Deep auto-encoder	Khan et al. (2019)	UNSW-NB15 dataset	0
Deep auto-encoder	Papamartzivanos et al. (2019)	NSL-KDD dataset	1
Denoising auto-encoder	Abusitta et al. (2019)	KDD Cup 1999 dataset	1

Tabla 1: Técnicas de aprendizaje profundo para la detección de intrusiones red, estudio en el que se utilizan y el dataset utilizado. Nota: Tabla adaptada del artículo [16].

2.3 Tipos de aprendizaje

Las técnicas de aprendizaje profundo se pueden dividir en tres grandes grupos, los modelos supervisados, los modelos no supervisados y los modelos semi supervisados. Cada uno de estos modelos trabaja con una filosofía diferente, por lo que los datos necesarios y las metodologías de funcionamiento para cada uno poseen unas diferencias sustanciales, que se explicarán a continuación. Esta explicación se apoya en contenidos de las referencias [16], [17] y [18].

2.3.1 Modelos supervisados

Los modelos supervisados incluyen arquitecturas como redes neuronales profundas, redes neuronales convolucionales o redes neuronales recurrentes. Este tipo de redes requieren el etiquetado de los datos de entrada para su correcto funcionamiento, requisito que, en muchos casos, dificulta la utilización de este tipo de modelos, pues en el caso particular de este trabajo, el etiquetado de flujos de red no es un proceso sencillo. Además, estos modelos ven sus resultados influenciados por el desbalance de las proporciones entre las clases del dataset, factor muy típico en datasets con tráfico de red, pues el tráfico benigno suele tener volúmenes mucho mayores dentro de una red. Sin embargo, si estos problemas son solventados, los modelos supervisados ofrecen por norma general mejores resultados frente a modelos no supervisados.

2.3.2 Modelos semi supervisados

Los modelos semi supervisados se encuentran entre el aprendizaje supervisado y el no supervisado. Estos modelos utilizan una pequeña cantidad de datos de entrenamiento

etiquetados junto con una gran cantidad de datos de entrenamiento no etiquetados. Esto ocurre a menudo en situaciones del mundo real en las que los datos etiquetados son muy caros o existe un gran volumen de datos que no pueden ser etiquetados.

2.3.3 Modelos no supervisados

Este tipo de modelos pretenden conseguir el aprendizaje de la composición y distribución inherente de datos, sin usar etiquetas proporcionadas explícitamente, de esta forma, pueden aprender a agrupar los datos según sus atributos en el caso de los algoritmos de clustering, o pueden aprender a reconstruir los datos que le son enseñados, como es el caso de los autoencoders.

2.4 Arquitecturas de Deep learning

Dentro del campo del aprendizaje profundo existen numerosas arquitecturas, a continuación, se describen brevemente sus características principales. Su utilización en diferentes estudios puede ser consultada en la Tabla 1.

2.4.1 Deep Neural Networks

Este tipo de redes combinan múltiples instancias del tipo perceptrón [19], conectando sus capas de forma lineal sin que exista ningún tipo de retroalimentación, por eso también son conocidas como perceptrón multicapa.

2.4.2 Redes recurrentes

Las redes recurrentes extienden los límites de las redes neuronales clásicas, pues operan con secuencias de datos. Para lograr trabajar con secuencias, disponen de bucles de realimentación donde las neuronas reciben como dato la salida del instante anterior además del dato de entrada, permitiendo que la información persista durante varias épocas. Esto supone un gran avance pues permiten a estas redes reconocer la temporalidad, pero presenta varios problemas, pues utiliza el mismo mecanismo de aprendizaje que el perceptrón multicapa, lo que dificulta en muchos casos el aprendizaje de las primeras neuronas de las capas recurrentes, esto se soluciona incorporando capas LSTM [20] o GRU [21], que permiten un nuevo mecanismo de aprendizaje que solventa este problema.

2.4.3 Redes convolucionales

Las redes convolucionales son otra variante del perceptrón multicapa. Son utilizadas principalmente para el procesamiento de imágenes, pues son capaces de modelar variaciones y comportamientos complejos. Estas redes se caracterizan por el modelado consecutivo de pequeñas piezas de información de la entrada de datos a través filtros que aumentan drásticamente la dimensionalidad del problema. Estas operaciones son conocidas como convoluciones, adicionalmente, utilizan otros tipos de capas que submuestrean los datos, o capas densamente conectadas, entre otras.

2.4.4 Autoencoders

Un autoencoder está formado una estructura simple similar a la del perceptrón multicapa, donde la única diferencia es el número de neuronas de la capa de salida, que en caso del autoencoder es el mismo que el número de neuronas en la primera capa. Esta peculiaridad se debe a que el autoencoder no debe identificar ninguna etiqueta, sino que tiene como objetivo reconstruir la entrada que le ha sido introducida, por eso se trata de un modelo no supervisado. Para conseguir esto, el autoencoder dispone de dos subestructuras un encoder

y un decoder. El encoder reduce la dimensionalidad de los datos para aprender una versión comprimida de los mismos, que a continuación es reconstruida por el decoder.

2.4.5 Deep Belief Networks

Las redes Deep Belief están compuestas por múltiples instancias de Boltzmann Machines restringidas [22] o RBM, que son modelos estocásticos usados para aprender la distribución de probabilidad sobre su conjunto de entradas. Las redes Deep Belief conectan múltiples RBM, donde la capa oculta de cada subred actúa como la capa de entrada para la capa adyacente.

2.5 Tipos de ciberataques

En la actualidad existen una gran cantidad de ciberataques y su clasificación puede ser en base a numerosos factores. Para agrupar los ciberataques más comunes se hace uso de una clasificación realizada por Cisco [23] y de los ataques que se encuentran dentro del top 10 de vulnerabilidades de OWASP [24]. Entre otros destacan los siguientes:

Malware, este término es empleado para referirse a cualquier tipo de software malicioso, sean virus, gusanos o ransomware. Para que el malware cause algún daño, debe infiltrarse a los sistemas a través de algún mecanismo, como puede ser a través de una vulnerabilidad en la red, o su instalación voluntaria o involuntaria por parte de un usuario.

Phishing, es un término utilizado para describir los ataques que pretenden suplantar una fuente fiable. Su objetivo es obtener acceso a datos sensibles de las víctimas o instalar malware en el sistema de la víctima.

Man-in-the-middle, este tipo de ataque se da cuando un atacante se sitúa entre la víctima y el otro extremo de la conexión que esta intenta realizar, obteniendo acceso a todos los datos que navegan a través de dicha conexión pudiendo espiar a la víctima. Además, en algunos casos, el atacante puede modificar los datos de respuesta del otro extremo de la conexión tratando de engañar a la víctima para cualquier fin malicioso.

Denial-of-service, este tipo de ataques tienen como objetivo agotar los recursos de una red, sistema o servidor, con el fin de imposibilitar al objetivo en cuestión la atención de solicitudes legítimas. Cuando estos ataques son realizados desde múltiples dispositivos, son conocidos como ataques de denegación de servicio distribuidos, como puede ser el caso de las botnets [25].

Infiltration, los ataques de infiltración agrupan una gran variedad de ataques y no tienen una metodología en concreto, lo que tienen en común es que se realizan al interior de la red, una vez una víctima ha sido comprometida y se ha creado una puerta trasera o *backdoor*.

Brute force, los ataques de fuerza bruta utilizan la capacidad de cómputo de los sistemas modernos para probar todas las combinaciones posibles dentro de un sistema de autenticación con credenciales, con el fin de obtener credenciales válidas y poder robar datos sensibles de la víctima o suplantar su identidad.

SQL injection, este tipo de ataque se produce cuando un atacante envía un tipo de información específica a través de un formulario con la intención de que los procesos internos de la aplicación ejecuten órdenes para las que no estaban diseñados. Esta

información específica suele estar compuesta por código específico de consultas de bases de datos SQL, de esta forma, los atacantes pueden ser capaces de manipular la base de datos, pudiendo obtener información no autorizada o incluso borrar datos de la base de datos.

Cross-site scripting (XSS), estos ataques se producen cuando una aplicación habilita el permiso de ejecución de fuentes externas sin verificación. De esta forma, los atacantes inducen al usuario a ejecutar código en su navegador, pudiendo redirigir al usuario a sitios maliciosos o secuestrar su sesión.

XML External Entities (XXE), estos ataques se producen cuando una aplicación dispone de un analizador de código XML mal configurado o antiguo. Debido a esto, los atacantes pueden enviar archivos XML que, al ser mal analizados, pueden llegar a ejecutar código externo o compartir archivos internos de la víctima.

3 Diseño y desarrollo

3.1 Introducción

Durante este capítulo se va a analizar en detalle el dataset utilizado, realizando varios análisis para conseguir un entendimiento mayor de los datos pues se realizará, por un lado, un análisis temporal, y, por otro lado, una exploración de los atributos más relevantes dentro del dataset. Todo esto es realizado para poder generar unos modelos de mayor calidad pues estarán desarrollados para resolver este problema particular.

3.2 Dataset utilizado

Como ya hemos visto en el estado del arte, existen numerosos datasets utilizados para la detección de ciberataques en red [Tabla 1], pero la mayor parte de estos presentan diferentes problemas. Por un lado, tenemos datasets muy desactualizados, que datan de fechas cercanas al año 2000, y por otro lado tenemos datasets muy específicos, donde los ataques que incluyen no representan una casuística que imite con fidelidad un entorno moderno y real.

Por lo que, para conseguir imitar este entorno real y moderno, el dataset utilizado ha sido el que ha sido considerado más fiel a estas características, pues está compuesto por ataques vanguardistas, incluye tráfico con protocolos recientes y ofrece tanto datos reales como simulados. Este dataset es el denominado CSE-CIC-2018, nombrado a partir de ahora únicamente como dataset, creado conjuntamente por la Communications Security Establishment (CSE) y el Canadian Institute for Cybersecurity (CIC) [26].

El dataset contiene un total de 80 atributos y está fragmentado en diferentes archivos, cada cual correspondiente a los datos obtenidos durante la captura de tráfico de diferentes días entre las fechas 14-02-2018 y 02-03-2018 entre las 8 y las 18 horas de cada día. Los flujos contenidos en cada fichero son etiquetados como benignos o malignos, aunque los malignos se diferencian entre trece clases diferentes.

Estos flujos representan resúmenes de todos los paquetes intercambiados entre dos pares IP-puerto mediante un mismo protocolo. Los flujos comienzan cuando se transmite el primer paquete y finalizan una vez que ha sido enviado el flag FIN del protocolo TCP, o cuando han transcurrido 6 horas desde el envío del último paquete.

Los detalles de los flujos albergados en cada fichero se mencionan a continuación. Un resumen gráfico se puede observar en la Tabla 2.

Wednesday-14-02-2018_TrafficForML_CICFlowMeter.csv
Este fichero contiene flujos benignos y ataques de fuerza bruta.

- FTP-BruteForce
- SSH-Bruteforce

Thursday-15-02-2018_TrafficForML_CICFlowMeter.csv
Este fichero contiene flujos benignos y ataques de denegación de servicio.

- DoS-GoldenEye
- DoS-Slowloris

Friday-16-02-2018_TrafficForML_CICFlowMeter.csv
Este fichero contiene flujos benignos y ataques de denegación de servicio.

- DoS-SlowHTTPTest
- DoS-Hulk

Wednesday-21-02-2018_TrafficForML_CICFlowMeter.csv
Este fichero contiene flujos benignos y ataques de denegación de servicio distribuidos.

- DDOS-LOIC-UDP
- DDOS-HOIC

Thursday-22-02-2018_TrafficForML_CICFlowMeter.csv
Este fichero contiene flujos benignos y ataques web.

- BruteForce -Web
- BruteForce -XSS
- SQL Injection

Friday-23-02-2018_TrafficForML_CICFlowMeter.csv
Este fichero contiene flujos benignos y ataques web.

- BruteForce -Web
- BruteForce -XSS
- SQL Injection

Wednesday-28-02-2018_TrafficForML_CICFlowMeter.csv
Este fichero contiene flujos benignos y dos ataques de infiltración.

- Infiltration 1
- Infiltration 2

Thursday-01-03-2018_TrafficForML_CICFlowMeter.csv
Este fichero contiene flujos benignos y dos ataques de infiltración.

- Infiltration 1
- Infiltration 2

Friday-02-03-2018_TrafficForML_CICFlowMeter.csv
Este fichero contiene flujos benignos y dos ataques de bots.

- Bot 1
- Bot 2

Dia-Mes	14-02	15-02	16-02	21-02	22-02	23-02	28-02	01-03	02-03	Total
Benign	667626	996077	446772	360833	1048213	1048009	544200	238037	762384	6112151
DDOS attack-HOIC	0	0	0	686012	0	0	0	0	0	686012
DoS attacks-Hulk	0	0	461912	0	0	0	0	0	0	461912
Bot	0	0	0	0	0	0	0	0	286191	286191
FTP-BruteForce	193360	0	0	0	0	0	0	0	0	193360
SSH-Bruteforce	187589	0	0	0	0	0	0	0	0	187589
Infilteration	0	0	0	0	0	0	68871	93063	0	161934
DoS attacks-SlowHTTPTest	0	0	139890	0	0	0	0	0	0	139890
DoS attacks-GoldenEye	0	41508	0	0	0	0	0	0	0	41508
DoS attacks-Slowloris	0	10990	0	0	0	0	0	0	0	10990
DDOS attack-LOIC-UDP	0	0	0	1730	0	0	0	0	0	1730
Brute Force -Web	0	0	0	0	249	362	0	0	0	611
Brute Force -XSS	0	0	0	0	79	151	0	0	0	230
SQL Injection	0	0	0	0	34	53	0	0	0	87
Total	1048575	1048575	1048574	1048575	1048575	1048575	613071	331100	1048575	8284195

Tabla 2: Cantidad de flujos organizados por clase en cada fichero del dataset.

Resaltar que el fichero Tuesday-20-02-2018_TrafficForML_CICFlowMeter.csv no ha sido utilizado, pues es un fichero muy voluminoso, causando problemas de espacio y rendimiento, donde solo se incluye un tipo nuevo de ciberataque (el DDoS attacks-LOIC-HTTP). Por estas razones se ha decidido descartar su uso.

Los 80 atributos proceden del procesamiento del tráfico de la red a través de la herramienta CICFlowMeter-V3 [27], para producir *flows* o flujos de datos, que resumen todo el tráfico intercambiado durante la conexión entre dos pares IP-puerto mediante un mismo protocolo. Este mecanismo es similar al utilizado por otros estándares del mercado como Netflow [28], Zeek [29] o IPFIX [30], aunque cada herramienta o estándar proporciona unas características del tráfico diferentes.

Este tráfico capturado es proveniente de un entorno en la nube que pretende simular una arquitectura de red convencional [Figura 1], formado por máquinas con servidores Linux, Windows 8, Windows 10 y Windows Server. El origen del tráfico tiene por un lado fuentes anónimas benignas, y por otro lado fuentes malignas controladas, pues el tráfico maligno es generado únicamente desde una plataforma delimitada compuesta por 50 equipos.

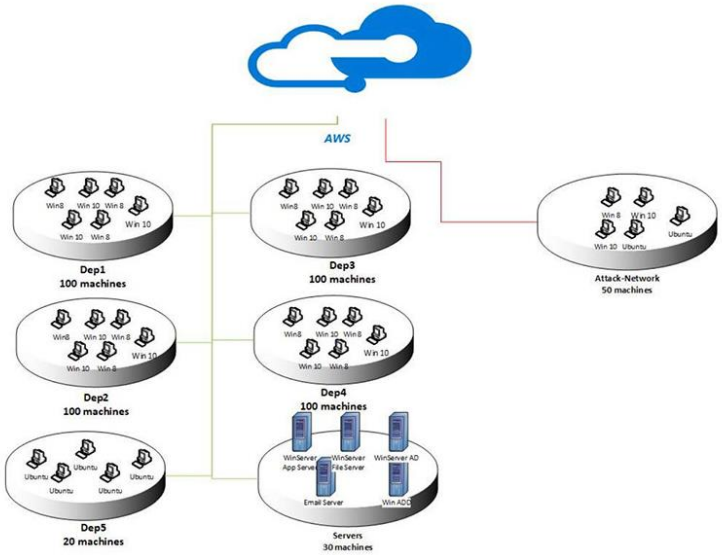

Figura 1: Entorno en la nube de Amazon Web Services obtenida de la referencia [26].

3.3 Exploración de los datos

Para poder generar unos modelos adecuados, es importante entender con detalle los datos con los que se trabaja, sobre todo cuando se trata un dataset con 80 atributos diferentes y cerca de 8,3 millones de datos. La primera característica notable del dataset es el desbalance que presentan las clases, como se puede observar en la Tabla 3. Esta distribución de las clases requerirá el uso de algún proceso de balance, como undersampling u oversampling, si las clases con menor representación no son bien clasificadas por los modelos generados.

Tipo de flujo	Cantidad
Benign	6112151
DDOS attack-HOIC	686012
DoS attacks-Hulk	461912
Bot	286191
FTP-BruteForce	193360
SSH-Bruteforce	187589
Infilteration	161934
DoS attacks-SlowHTTPTest	139890
DoS attacks-GoldenEye	41508
DoS attacks-Slowloris	10990
DDOS attack-LOIC-UDP	1730
Brute Force -Web	611
Brute Force -XSS	230
SQL Injection	87
Total	8284195

Tabla 3: Clases de los flujos presentes en el dataset y cantidad presente de cada uno.

El dataset cuenta con una cantidad notoria de atributos que describen distintas características de los flujos de datos, como pueden ser, cantidad de paquetes enviados y recibidos, cantidad de bytes enviados y recibidos, tiempo mínimo, medio y máximo entre paquetes, tamaño medio de los paquetes etcétera. Para una exploración más detallada consultar el Anexo A. Dentro de la gran cantidad de atributos, existe uno que aporta una información que no está directamente relacionada a los flujos, el timestamp, por lo que a continuación se realizará un análisis en detalle de este atributo con el objetivo de comprobar la distribución de los flujos y descubrir si es posible utilizar redes recurrentes en caso de que los datos estén bien ordenados respecto al tiempo.

Para el resto de los atributos, dada la gran cantidad de ellos, la única exploración a realizar ha sido la existencia de valores atípicos, como infinitos, NaN y la exploración de la importancia mediante el uso de árboles de decisión.

3.3.1 Exploración temporal

El primer análisis realizado tiene como objetivo identificar la distribución de los ataques durante las horas del día, para observar si están repartidos durante todo el espectro temporal o si están sesgados al ser realizados de forma sintética. Realizando el histograma de la cantidad de flujos por minuto representado en la Figura 2, se observa que los flujos están distribuidos de forma relativamente equitativa durante el espectro especificado en la descripción del dataset, desde las 8 hasta las 18 horas. Aunque cabe destacar la existencia de dos zonas con cantidades de flujos muy superiores a la media, situadas al final de las 13 horas, y durante las 14. Como se puede contemplar en la Figura 8 estos altos volúmenes de tráfico se deben a la presencia de dos ataques cuyo volumen es muy alto, pero un dato preocupante es el aumento repentino de flujos benignos, que podría suponer un sesgo en los datos, pues no hay razón aparente para su drástico aumento.

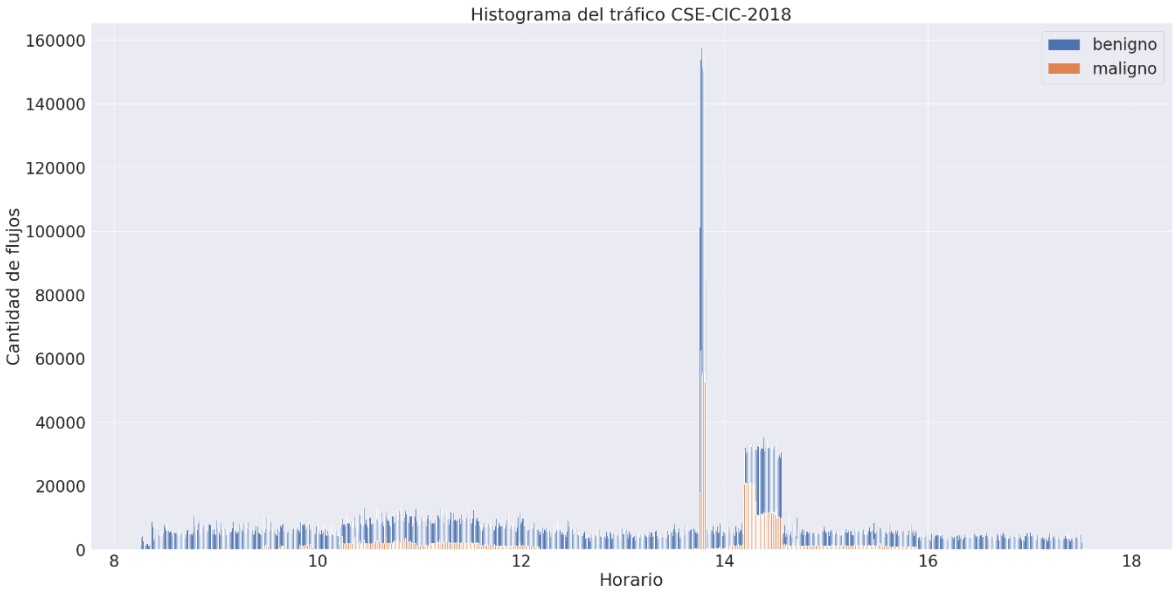

Figura 2: Histograma del tráfico.

El siguiente análisis realizado tiene la intención de comprobar la distribución de los datos durante los minutos y segundos, para esto se ha realizado un histograma del minuto [Figura 3] y del segundo [Figura 4] de la variable *timestamp* de cada flujo. Los segundos están distribuidos de forma uniforme, por lo que a pesar de un dataset proveniente de una simulación, no se encuentran sesgos. Sin embargo, en el histograma de los minutos se pueden diferenciar dos regiones con mayor cantidad de flujos, que se corresponden con los minutos de las regiones con un comportamiento anómalo detectadas en el primer análisis.

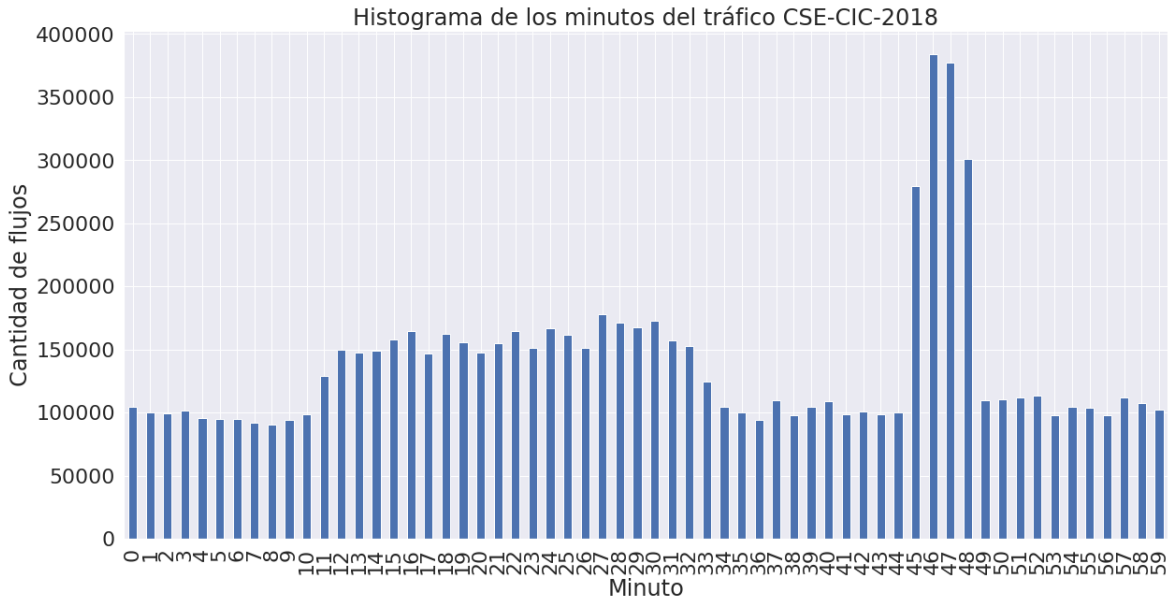

Figura 3: Distribución de las cantidades de flujo a lo largo de los minutos de la hora.

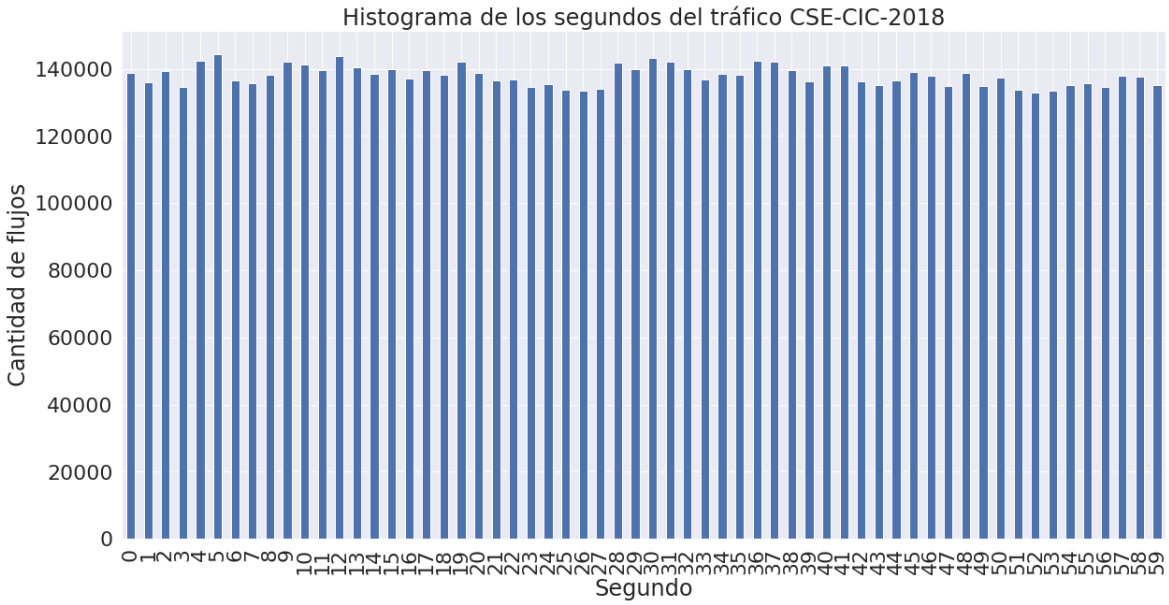

Figura 4: Histograma de los segundos de cada flujo.

Se ha realizado una exploración temporal particular para cada tipo de ataque, estas distribuciones temporales pueden ser observadas en las gráficas comprendidas entre la Figura 9 y la Figura 21 del Anexo B. Tras haber realizado estos análisis, habiendo observado que el tráfico sigue un orden temporal, que los flujos están distribuidos de forma uniforme durante los segundos y que solo existe sesgo en dos clases en momentos puntuales, se considera viable la exploración de los resultados con el uso de redes LSTM, pues estas son capaces de detectar patrones en información secuencial.

3.3.2 Atributos más relevantes

A pesar de que las redes neuronales se pueden considerar clasificadores universales pues son capaces de resolver prácticamente cualquier problema, no destacan por su transparencia, pues es difícil conocer qué atributos pesan más a la hora de decidir dentro de la red. Como parte del estudio, y con intención de conseguir más información al respecto, se ha realizado una clasificación utilizando otro mecanismo de machine learning, los árboles de decisión, que sí son capaces de definir qué atributos aportan más información a la hora de definir cada clase.

En este caso se ha entrenado un ExtraTreesClassifier [31] formado por 500 árboles de decisión y se han obtenido los resultados reflejados en la Figura 5. En esta figura podemos observar que existen dos atributos que poseen un peso considerablemente superior a los demás a la hora de tomar decisiones, en concreto el número de bytes enviados en la ventana inicial y la cantidad mínima de bytes en un segmento. Si analizamos estos dos atributos, se puede deducir que, dada la gran cantidad de muestras de ataques de denegación de servicio y de fuerza bruta y al basarse estos ataques en la repetición continua y envío constante de paquetes, estos dos atributos caracterizan especialmente a estos flujos malignos más abundantes.

Por otro lado, cabe destacar que la mayor parte de los atributos cuya importancia a la hora de tomar decisiones es prácticamente nula tienen en común el hecho de que su valor refleja la media de los valores de la característica del flujo que representan.

En conclusión, salvo los atributos recién comentados, la mayor parte de los atributos del dataset aportan algún tipo de información a la hora de tomar decisiones, y, puesto que la diferencia entre ellos es continua y no existen diferencias muy sustanciales, se ha decidido no eliminar ningún otro atributo salvo los comentados a continuación en el preprocesamiento de los datos.

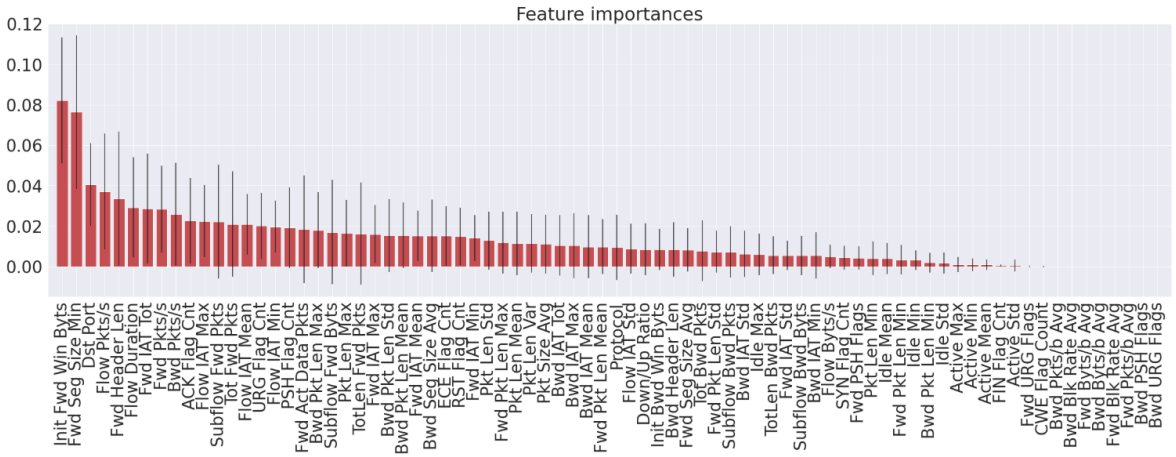

Figura 5: Importancias de los atributos.

3.4 Procesamiento de los datos

Una vez realizada la exploración y entendidos los atributos del dataset, se procede a realizar una limpieza de los datos para la adecuación de su uso en el entrenamiento y predicción de los modelos.

Para facilitar el trabajo con todos los datos, se han unificado todos los datos procedentes de los diferentes ficheros en un único DataFrame.

Cada modelo requiere de un preprocesamiento de los datos específico, por lo que se van a dividir los diferentes procesos realizados diferenciando para cada modelo. Para contemplar el orden en el que han sido realizados cada uno de los preprocesamientos, véase la Tabla 4, donde la 'X' representa que el modelo correspondiente a la columna utiliza el proceso indicado en la fila.

Preprocesamiento	Perceptron	Autoencoder	Redes recurrentes
Eliminación de los valores atípicos	X	X	X
Balanceo de los datos	X		X
Separación entre datos benignos y malignos		X	
Conversión de las clases	X		X
Separación entre entrenamiento y test	X	X	X
Ordenación de los datos respecto a la fecha			X
Eliminación de la variable timestamp	X	X	X
Estandarización de los datos	X	X	X
Agrupación en series			X

Tabla 4: Tipos de preprocesamiento realizados, mostrados por orden de utilización.

3.4.1 Procesos comunes a todos los modelos

3.4.1.1 Eliminación de los valores atípicos.

El primer procedimiento común a todos los datos ha sido la eliminación de los flujos donde alguno de sus atributos estaba compuesto por valores NaN o infinito. Al ser una cantidad muy escasa como se puede observar en Tabla 5, se ha optado por este método, y no por la reconstrucción de los datos con metodologías de proximidad o similares, para así simplificar el problema.

Tipo de flujo	Cantidad	Valores con NaN o infinito	Cantidad sin infinito ni NaN
Benign	6112151	35006	6077145
DDOS attack-HOIC	686012	0	686012
DoS attacks-Hulk	461912	0	461912
Bot	286191	0	286191
FTP-BruteForce	193360	6	193354
SSH-Bruteforce	187589	0	187589
Infilteration	161934	1295	160639
DoS attacks-SlowHTTPTest	139890	0	139890
DoS attacks-GoldenEye	41508	0	41508
DoS attacks-Slowloris	10990	0	10990
DDOS attack-LOIC-UDP	1730	0	1730
Brute Force -Web	611	0	611
Brute Force -XSS	230	0	230
SQL Injection	87	0	87
Total	8284195	36307	8247888

Tabla 5: Cantidad de flujos totales, con valores NaN o infinito y cantidad total tras el filtrado.

3.4.1.2 Separación entre entrenamiento y test.

Para el correcto entrenamiento de los datos y evitar sobreajuste, los datos han sido separados en dos subgrupos, uno de entrenamiento y otro de test, esta separación se realiza de forma aleatoria con unos porcentajes de 75% para entrenamiento y 25% para test.

3.4.1.3 Eliminación de la variable timestamp.

Los análisis de la variable temporal, realizados en el capítulo 3.3.1, han mostrado que los ataques no están repartidos de forma equitativa durante el espectro temporal, sino que muchos se encuentran durante una breve franja horaria, por lo que para evitar que se creen sesgos y que las redes relacionen la hora con un ataque, se elimina el atributo *timestamp* de los conjuntos de entrenamiento y test.

3.4.1.4 Estandarización de los datos.

Los datos de los diferentes atributos presentan diferentes ordenes de magnitud, esto puede causar problemas a la hora de entrenar la red, pues los pesos de las conexiones de la red pueden dispararse a infinito si los datos numéricos de las entradas son muy altos. Para evitar esto se ha realizado una estandarización de los datos utilizando el paquete Standar Scaler [32] de Scikit Learn [33]. Para garantizar que la información de los datos de test no ha sido presentada a la red de ninguna manera, el proceso de estandarización es entrenado únicamente con los datos de entrenamiento, es decir, las medias y las desviaciones típicas necesarias son calculadas únicamente con los datos entrenamiento, y, una vez hecho esto, se procede al cálculo de los nuevos valores estandarizados para entrenamiento y test.

3.4.2 Procesos en común de los modelos supervisados

3.4.2.1 Balanceo de los datos.

Como hemos podido ver en la Tabla 3, la cantidad de flujos de cada clase está muy desproporcionada, por lo tanto se ha realizado un proceso de undersampling para balancear las clases y así aumentar la proporción de los ataques, con el fin de que las redes neuronales supervisadas aprendan mejor la estructura de estos, y se consiga mejorar el índice de detección, pues es uno de los objetivos de este trabajo.

Existen numerosas técnicas de undersampling, desde las más simples que se basan en selección uniforme y aleatoria, a otras basadas en la proximidad. Dada la gran cantidad de datos y las limitaciones del sistema, el cálculo de las proximidades en las clases más abundantes supone una tarea muy costosa debido al coste computacional que supone el cálculo de la distancia entre todos los datos. Por este motivo y para simplificar el problema, se ha descartado el uso de técnicas como NearMiss, Condensed Nearest Neighbour o ClusterCentroids [34] [35] en favor de la selección aleatoria. A pesar de ser una selección aleatoria, la cantidad de flujos elegidos está influenciada por los valores elegidos en el trabajo [16], pues uno de los objetivos de este trabajo es comparar los resultados obtenidos con resultados anteriores.

3.4.2.2 Conversión de las clases.

Durante este paso el objetivo es procesar la clase para que sean entendibles por la red. Se ha realizado un One-Hot Encoder [36], convirtiendo las cadenas de caracteres que definen cada ataque, en un vector binario con longitud igual al número de clases [Tabla 3], donde cada clase es asignada una posición del vector, y, por lo tanto, cada cadena es convertida en un vector de ceros salvo un uno en la posición que corresponda a la clase a convertir.

3.4.3 Procesos exclusivos de los modelos no supervisados

3.4.3.1 Separación entre datos benignos y malignos.

El objetivo para el correcto funcionamiento de este tipo de modelos consiste en garantizar que aprendan a identificar lo mejor posible el tráfico benigno. Es por esto por lo que es necesaria la separación de los datos entre benignos y malignos, pues los primeros serán los únicos utilizados durante el entrenamiento, y los segundos solo serán utilizados para las predicciones.

3.4.4 Procesos exclusivos de las redes recurrentes

3.4.4.1 Ordenación de los datos respecto a la fecha.

Para el caso de las redes recurrentes, el tiempo es un factor importante, debido a esto, el preprocesamiento de los datos para estas redes incluye un paso añadido, que consiste en la ordenación de los datos de cada subgrupo en función del atributo *timestamp*.

3.4.4.2 Agrupación en series.

Este tipo de modelos necesitan que los datos sean presentados en forma de series. Para lograr esto, los flujos son agrupados en series de tamaño fijo. La elección de este parámetro se ha realizado con dos objetivos en mente, por un lado, mantener un número adecuado de datos en entrenamiento y test, y, además, el no elegir un tamaño muy grande para poder observar

las diferencias entre la red *stateless*, que solo dispone de la información de cada serie, y la red *stateful*, que sí utiliza la memoria completa de las redes LSTM.

Por lo tanto, en base a este último argumento y a las cantidades de datos generadas en función del tamaño de la serie [Tabla 6], se ha elegido 24 como tamaño ideal pues satisface ambos objetivos.

Tamaño de la serie	Cantidad de series en train	Cantidad de series en test	Cantidad de series totales
1	47616,8	15872,3	63489,0
4	11904,2	3968,1	15872,3
8	5952,1	1984,0	7936,1
12	3968,1	1322,7	5290,8
24	1984,0	661,3	2645,4
36	1322,7	440,9	1763,6
48	992,0	330,7	1322,7
60	793,6	264,5	1058,2
72	661,3	220,4	881,8
84	566,9	189,0	755,8
96	496,0	165,3	661,3
108	440,9	147,0	587,9
120	396,8	132,3	529,1
132	360,7	120,2	481,0
144	330,7	110,2	440,9
156	305,2	101,7	407,0
168	283,4	94,5	377,9
180	264,5	88,2	352,7
192	248,0	82,7	330,7

Tabla 6: Cantidad de series generadas según el tamaño de la serie elegido.

3.5 Modelos generados

Los modelos generados han sido desarrollados utilizando Keras [37], el api de alto nivel de TensorFlow [38]. Todos los modelos están basados en modelos secuenciales, donde las capas son apiladas y solo se conectan con la siguiente según el orden en el que han sido creadas. Todos los modelos han sido compilados utilizando el optimizador “Adam” [39].

3.5.1 Perceptrón multicapa

El perceptrón multicapa está construido con una arquitectura simple, formada por 4 capas densas completamente conectadas, las tres primeras utilizan como función de activación ReLU [40] y disponen de 78, 45 y 30 nodos en sus capas, respectivamente. La última capa está compuesta por 14 neuronas y utiliza la función de activación Softmax [41].

La elección de las neuronas de la primera y de la última capa no es aleatoria, pues estos tamaños se deben al número de atributos y al número de clases, respectivamente. La elección del número de neuronas de las capas ocultas sigue un modelo des incremental y progresivo, pues tiene como intención favorecer la generalización de los datos; la elección del número de capas ha sido basada en la arquitectura elegida para el autoencoder, pues se ha tratado de mantener la igualdad entre los modelos, para que, de esta forma, sea el método de

funcionamiento de cada modelo el que dicte qué modelo resuelve mejor el problema, no la cantidad de parámetros de los que se disponga.

La función objetivo utilizada durante entrenamiento ha sido *Categorical Crossentropy* [42] al tratarse de un problema de clasificación con múltiples clases.

3.5.2 Autoencoder

El Autoencoder está construido de forma similar al perceptrón multicapa, tanto en número de capas como de neuronas, pues el objetivo era realizar una comparación lo más justa posible. Las 5 capas densas que lo conforman están completamente conectadas, las 4 primeras utilizan la función de activación ReLU mientras que la última no tiene función de activación. Las capas contienen 78, 45, 30, 45 y 78 neuronas respectivamente. Las capas 2 y 3 se corresponden al encoder, y las capas 3 y 4 se corresponden al decoder. La primera y la última capa disponen de 78 neuronas pues se corresponde con el número de atributos.

La función objetivo utilizada durante el entrenamiento ha sido el error cuadrático medio, pues esta función de coste es simétrica, y dado que el objetivo es que el autoencoder aprenda a reconocer el tráfico benigno, es necesario penalizarle de igual forma si predice valores superiores o inferiores a los deseados.

3.5.3 Redes recurrentes

Las redes recurrentes mantienen la misma filosofía que los modelos anteriores en cuanto a la reducción de la cantidad de neuronas por capa para aumentar la generalización, pero las capas utilizadas son diferentes. En concreto las redes recurrentes están formadas por una capa LSTM con 64 unidades de cómputo, una capa de *dropout* con una proporción de 0.2 para regularizar los resultados y evitar sobreajuste, y una capa densamente conectada con 14 neuronas correspondientes a la cantidad de clases.

La función objetivo utilizada durante el entrenamiento ha sido *Categorical Crossentropy* al tratarse de un problema de clasificación con múltiples clases al igual que con el perceptrón multicapa.

Para las redes recurrentes se han realizado dos modelos diferentes, uno con el parámetro *stateful* activado [43] y otro este atributo desactivado (*stateless*). Este parámetro produce que, en el caso de haber sido desactivado, cuando la neurona es expuesta al siguiente lote, los pesos y los valores de las neuronas de las capas ocultas se vuelvan a inicializar, mientras que, si el parámetro *stateful* está activado, los pesos se mantienen entre un lote y otro. La idea detrás de esta decisión es la de comprobar si el hecho de que la red conozca todo lo que le ha sido mostrado previamente, supone alguna ventaja respecto a presentarle únicamente el contexto más próximo de cada flujo proveniente de la serie en la que estos se encuentren.

4 Pruebas y resultados

4.1 Descripción de las pruebas realizadas

Todos los modelos descritos durante el apartado 3 han sido entrenados durante 100 épocas. Para comprobar la eficacia de cada modelo, se han realizado una serie de métricas idénticas en cada caso para garantizar la obtención del mejor método de detección.

Los índices calculados más relevantes para todos los casos han sido:

- La proporción de verdaderos positivos (TP), o detección correcta de ciberataques,
- La proporción de falsos positivos (FP), o clasificación incorrecta de tráfico benigno,
- La proporción de falsos negativos (FN), o clasificación incorrecta de ciberataques
- La proporción de verdaderos negativos (TN), o clasificación correcta del tráfico benigno.

A partir de estos valores se pueden construir métricas más complejas como el recall o la precisión. Estos se calculan de la siguiente forma:

- Precisión = TP / (TP + FP)
- Recall = TP / (TP + FN) = TPR

La precisión indica la tasa de acierto conseguida sobre el total de datos predichos como maliciosos, el recall, sin embargo, representa la tasa de acierto conseguida sobre el total de las muestras del dataset, definición que coincide con el TPR o ratio de verdaderos positivos.

Dado que uno de los objetivos principales de este trabajo es la detección de ciberataques, se tomará especial atención a la métrica del recall, pues como hemos visto, esta métrica indica el TPR, que es la cantidad de elementos relevantes que han sido verdaderamente clasificados como tales, siendo los elementos relevantes los ciberataques en este caso; pero nunca a costa de sacrificar la efectividad de la correcta identificación del tráfico benigno, aunque este pasará a un segundo plano.

También se obtendrá la evolución del loss en cada época de entrenamiento, para verificar si las épocas establecidas han sido suficientes para que el modelo aprenda a diferenciar las clases.

El autoencoder, dado que presenta un hiperparámetro añadido respecto a los métodos supervisados, será expuesto a otra prueba en la que se comprobarán los resultados de las pruebas nombradas anteriormente en función del umbral escogido.

Para conseguir una mejor presentación y un formato único, se ha hecho uso de las librerías Classification Report [44] y Confusion Matrix [45] de Scikit Learn.

4.2 Resultados obtenidos con clasificadores

Los resultados obtenidos por los clasificadores son fáciles de contrastar, por un lado, se ha obtenido durante la fase de preprocesamiento un vector con las clases y por otro, al predecir utilizando el modelo recién entrenado obtenemos el vector de las predicciones de la red. Este vector de predicciones representa el valor que alberga cada neurona de salida para cada uno

de los datos que se le presentan, por lo que para traducir esto a una predicción en sí, se obtiene el índice de la neurona con un valor de activación mayor, este se corresponderá con la codificación One-Hot de alguna clase.

Una vez procesado el vector de predicciones, este será enviado a la librería de creación de reportes junto a los valores reales de las clases.

4.2.1 Resultados perceptrón multicapa

Los resultados del perceptrón multicapa se pueden apreciar en la Tabla 7, estos resultados obtenidos destacan en tres aspectos principales. Por un lado, el perceptrón multicapa es capaz de detectar correctamente prácticamente el 100% de los ataques de denegación de servicio al igual que los flujos correspondientes a los bots y los ataques bruteforce ssh. En segundo lugar, dada la gran cantidad de datos presentes de estas clases mencionadas anteriormente, el porcentaje total de acierto se eleva hasta valores cercanos al 90%, posicionando al perceptrón multicapa como una medida de detección muy eficaz, salvo en la detección de ciertos ataques. Esto nos lleva a destacar el último aspecto interesante, la baja efectividad del perceptrón multicapa a la hora de detectar ataques de infiltración, bruteforce tanto ftp como -XSS.

Explorando más en detalle los resultados de las clases peor clasificadas, si observamos la matriz de confusión del perceptrón multicapa en la Figura 22, podemos deducir el porqué de los malos resultados obtenidos para la clasificación de los flujos de ftp bruteforce e infiltración por un lado y los malos resultados de los ataques web por otro.

Los malos resultados de los primeros ataques mencionados son los más sorprendentes, pues a pesar de ser dos clases relativamente mayoritarias que poseen una gran cantidad de flujos con los que la red puede entrenar, esta no es capaz de detectarlos con facilidad. Observando la matriz de confusión, los flujos de infiltración son confundidos principalmente con flujos benignos, este resultado es razonable, pues este ataque pretende realizar fines maliciosos imitando al tráfico benigno. Por otra parte, los flujos del ataque bruteforce ftp son confundidos principalmente con los ataques de denegación de servicio.

Los flujos de los ataques web son muy minoritarios dentro del dataset, por lo tanto, los resultados obtenidos para la detección de estos flujos son cercanos a los esperados, además, estos ataques se basan en el contenido enviado en la carga de los paquetes, información que no se ve muy bien representada por ningún atributo del dataset.

	precision	recall	f1-score	support
Benign	0.92	0.97	0.94	15952
Bot	1.00	1.00	1.00	3555
Brute Force -Web	0.68	0.87	0.76	130
Brute Force -XSS	1.00	0.52	0.69	67
DDOS attack-HOIC	1.00	1.00	1.00	8578
DDOS attack-LOIC-UDP	1.00	1.00	1.00	415
DoS attacks-GoldenEye	1.00	1.00	1.00	5206
DoS attacks-Hulk	1.00	1.00	1.00	5869
DoS attacks-SlowHTTPTest	0.85	0.98	0.91	17400
DoS attacks-Slowloris	1.00	1.00	1.00	1417
FTP-BruteForce	0.87	0.40	0.54	4806
Infilteration	0.63	0.30	0.40	2022
SQL Injection	0.49	0.87	0.62	23
SSH-Bruteforce	1.00	1.00	1.00	2335
accuracy			0.92	67775
macro avg	0.89	0.85	0.85	67775
weighted avg	0.92	0.92	0.91	67775

Tabla 7: Reporte de clasificación del perceptrón multicapa.

4.2.2 Resultados redes recurrentes

Los resultados de las redes recurrentes han sido divididos en dos subapartados, cada uno correspondiente a cada uno de los modelos generados. Los resultados de estas redes esperan ser mejores que el perceptrón multicapa, pues las tres redes se entrenan empleando el mismo mecanismo, pero las redes recurrentes poseen, por un lado, información de los flujos que la rodean en caso de la red con el parámetro *stateless*, y, por otro lado, información de todos los flujos anteriores en el caso de la red con el parámetro *stateful*.

4.2.2.1 Stateless

La red *stateless* se encuentra en unas condiciones muy similares al perceptrón multicapa, pues la única información extra que posee es la proveniente de los flujos de la serie procesados en las neuronas anteriores, aportándole a la red una memoria, aunque esta sea muy escasa.

Como podemos observar en la Tabla 8, los ataques que han sido bien detectados por el perceptrón multicapa lo son también por esta red recurrente, que, además, gracias a la pequeña memoria que dispone, le aporta una información extra suficiente como para mejorar drásticamente los resultados de los ataques que no eran muy bien clasificados por el perceptrón multicapa. Este hecho se muestra principalmente en la tasa de detección de los ataques de infiltración y bruteforce ftp.

Una posible explicación de estos resultados reside en las figuras Figura 12 y Figura 13. Si observamos estas gráficas, que representan el histograma de la cantidad de flujos del ataque respecto al tiempo, podemos observar que los flujos se encuentran muy seguidos entre sí, por lo que, al disponer de una memoria, es muy probable que las series dispongan de varias ocurrencias de los flujos malignos, ayudando a caracterizar a este tipo de ataques.

	precision	recall	f1-score	support
Benign	0.98	0.98	0.98	15916
Bot	1.00	1.00	1.00	3526
Brute Force -Web	0.88	0.87	0.88	158
Brute Force -XSS	0.91	0.70	0.79	60
DDOS attack-HOIC	1.00	1.00	1.00	8453
DDOS attack-LOIC-UDP	1.00	1.00	1.00	438
DoS attacks-GoldenEye	1.00	1.00	1.00	5242
DoS attacks-Hulk	1.00	1.00	1.00	5850
DoS attacks-SlowHTTPTest	0.97	0.99	0.98	17583
DoS attacks-Slowloris	1.00	1.00	1.00	1412
FTP-BruteForce	0.97	0.88	0.92	4794
Infilteration	0.88	0.86	0.87	1994
SQL Injection	0.62	0.32	0.42	25
SSH-Bruteforce	1.00	1.00	1.00	2301
accuracy			0.98	67752
macro avg	0.94	0.90	0.92	67752
weighted avg	0.98	0.98	0.98	67752

Tabla 8: Reporte de clasificación de la red LSTM *stateless*.

4.2.2.2 Stateful

Sobre el papel, la red entrenada con este parámetro dispone de una ventaja frente a las demás, pues muchos ataques no están basados en un solo flujo, sino que la presencia de varios flujos es lo que lleva a considerar a esta serie de flujos un ataque, como es el caso de la denegación de servicio, los ataques de fuerza bruta o los escaneos de puertos. Además, otros ataques pueden ser mejor caracterizados una vez que se conoce el contexto del tráfico previo.

La red entrenada con la característica *stateful* dispone de todo el contexto, pues toda la información que ha atravesado la red es transmitida a través del estado de las neuronas. La ventaja de la que dispone esta red sobre el papel se ve reflejada en los resultados de la Tabla 9.

Si la red entrenada con el parámetro *stateless* obtenía los mismos resultados en los flujos que ya eran bien identificados por el perceptrón multicapa, y obtenía mejores resultados en los que peor eran identificados, la red recurrente con el parámetro *stateful* consigue mejorar aún más los resultados anteriores, llegando a identificar con éxito el 99% de los flujos presentes en el dataset, logrando más de un 95% de detección en todos los ataques salvo los sql injection.

	precision	recall	f1-score	support
Benign	1.00	0.98	0.99	15916
Bot	1.00	1.00	1.00	3526
Brute Force -Web	0.91	0.94	0.92	158
Brute Force -XSS	0.92	0.95	0.93	60
DDOS attack-HOIC	1.00	1.00	1.00	8453
DDOS attack-LOIC-UDP	1.00	1.00	1.00	438
DoS attacks-GoldenEye	1.00	1.00	1.00	5242
DoS attacks-Hulk	1.00	1.00	1.00	5850
DoS attacks-SlowHTTPTest	0.99	1.00	0.99	17583
DoS attacks-Slowloris	1.00	1.00	1.00	1412
FTP-BruteForce	0.98	0.98	0.98	4794
Infilteration	0.89	0.99	0.93	1994
SQL Injection	0.76	0.52	0.62	25
SSH-Bruteforce	1.00	1.00	1.00	2301
accuracy			0.99	67752
macro avg	0.96	0.95	0.96	67752
weighted avg	0.99	0.99	0.99	67752

Tabla 9: Reporte de clasificación de la red LSTM *stateful*.

4.3 Resultados obtenidos con el autoencoder

Los resultados de los autoencoders, al igual que los resultados de los clasificadores, requieren de un preprocesamiento antes de poder realizar los reportes y la matriz de confusión.

Una vez realizadas las predicciones, los resultados obtenidos son vectores con reconstrucciones del flujo según lo aprendido por el autoencoder, por lo que para conseguir una predicción de una clase, es necesario calcular el error cuadrático medio del flujo introducido en la red respecto al que ha sido producido, de esta forma, y tras comprobar si el error supera el umbral establecido inicialmente, el flujo será clasificado como benigno si es inferior al umbral, o como ataque si supera el umbral de error.

Para conseguir reportes más detallados, los flujos malignos que han sido correctamente clasificados como tales, son convertidos al tipo de ataque exacto que pertenecen, así es posible observar las métricas para cada clase y poder obtener conclusiones más acertadas.

Antes de obtener el reporte de clasificación, es necesario establecer el umbral, su elección no es un proceso trivial, y una ligera variación puede acarrear grandes cambios en las predicciones, debido a esto se ha realizado una prueba inicial para identificar cual es el valor óptimo del umbral.

Este valor óptimo es una métrica subjetiva, pues dependiendo de la relevancia que se le otorgue a la detección de ciberataques, el valor adecuado cambiará. También hay que tener en cuenta que los falsos positivos se traducen en tráfico benigno clasificado como maligno y, por lo tanto, en caso de ser los resultados de la red una métrica para bloquear tráfico, se reduciría la calidad del tráfico de red que se está analizando, pues las conexiones benignas se verían interrumpidas al ser clasificadas como ataques.

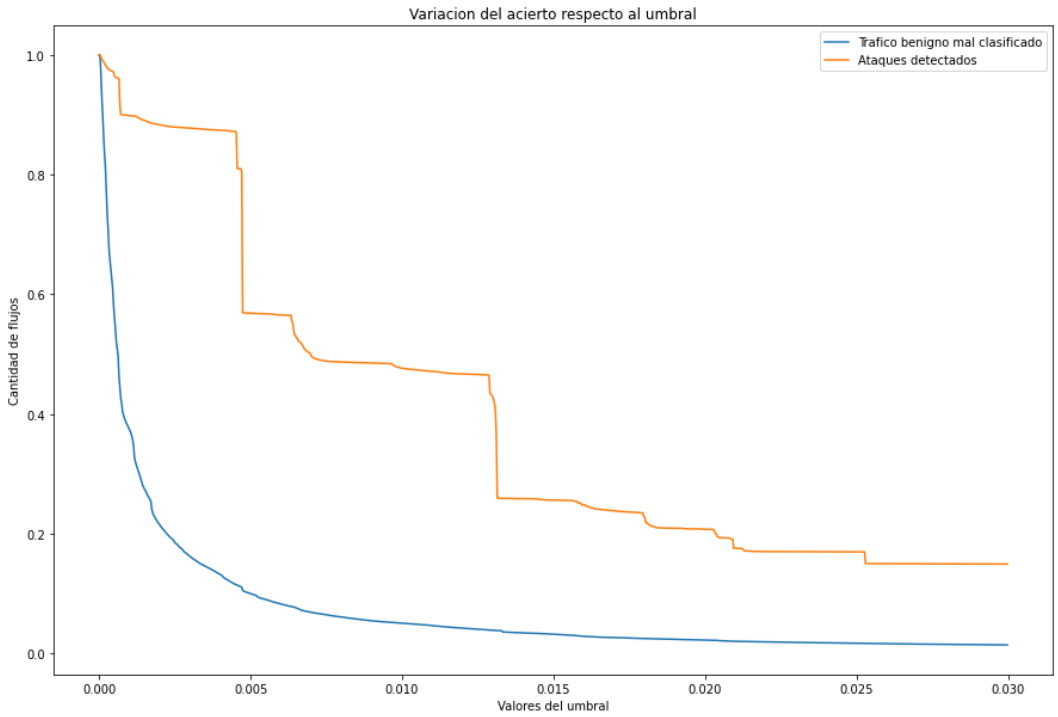

Figura 6: Variación de las métricas respecto al umbral.

Como podemos observar en la Figura 6, para conseguir un alto nivel de detección de ciberataques, es necesario sacrificar una gran cantidad de flujos benignos, a pesar de esto, como el objetivo de este trabajo es principalmente la detección de ciberataques, utilizando una medida ponderada y con disposición de sacrificar un máximo del 15% de los flujos benignos, el umbral elegido para la generación del informe es de 0.0045. Los resultados del reporte están representados en la Tabla 10.

Dado el procesamiento realizado al vector de predicciones para ser convertido al formato necesario para realizar el reporte, es conveniente destacar que el valor del recall de los flujos benignos, y los valores de la precisión de los ataques, son métricas que se encuentran sesgadas, pero esto no impide la posibilidad de realizar un análisis del resto de valores. Este sesgo también es reflejado en la matriz de confusión [Figura 24], pues es esta es construida a partir de las métricas sesgadas.

	precision	recall	f1-score	support
Benign	0.96	1.00	0.98	6077145
Bot	1.00	0.49	0.66	286191
Brute Force -Web	1.00	0.60	0.75	611
Brute Force -XSS	1.00	0.50	0.67	230
DDOS attack-HOIC	1.00	1.00	1.00	686012
DDOS attack-LOIC-UDP	1.00	1.00	1.00	1730
DoS attacks-GoldenEye	1.00	1.00	1.00	41508
DoS attacks-Hulk	1.00	1.00	1.00	461912
DoS attacks-SlowHTTPTest	1.00	1.00	1.00	139890
DoS attacks-Slowloris	1.00	1.00	1.00	10990
FTP-BruteForce	1.00	1.00	1.00	193354
Infilteration	1.00	0.18	0.31	160639
SQL Injection	1.00	0.52	0.68	87
SSH-Bruteforce	1.00	1.00	1.00	187589
accuracy			0.97	8247888
macro avg	1.00	0.81	0.86	8247888
weighted avg	0.97	0.97	0.96	8247888

Tabla 10: Reporte de clasificación del autoencoder.

En primer lugar, destaca la efectividad del autoencoder a la hora de detectar los ataques de denegación de servicio y bruteforce tanto ftp como ssh, pues ha detectado el 100% de todos los flujos. Estos resultados se deben a que este tipo de ataques se basan en la repetición, por lo que los flujos por lo general son muy parecidos, esto provoca que, si un flujo es detectado, el resto lo van a ser también. Este fenómeno se puede observar en la Figura 7, donde se representan el histograma del error cuadrático medio de los flujos, pues existen numerosas regiones donde los ataques se agrupan en grandes cantidades con un mismo error.

En segundo lugar, destaca que el autoencoder ha sido capaz de detectar al menos un flujo de cada uno de los ataques, obteniendo peores resultados en la detección de flujos malignos que imitan el comportamiento humano o poseen flujos menos característicos, como pueden ser los ataques web, los ataques bot y los ataques de infiltración.

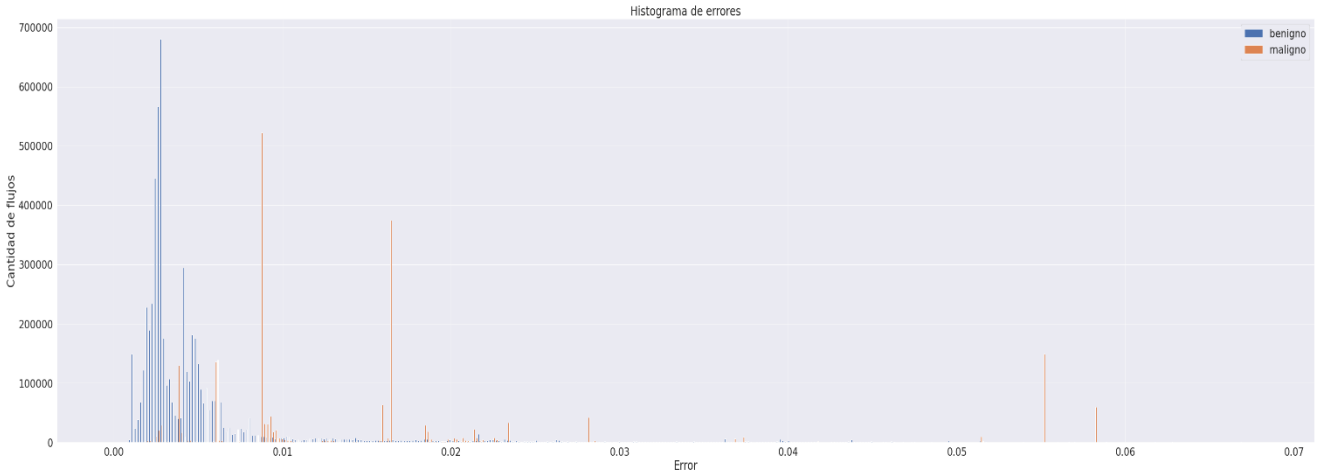

Figura 7: Histograma de errores de los flujos predichos por el autoencoder

4.4 Análisis de los resultados

Se considera que todos los modelos han convergido y han realizado un aprendizaje suficiente, pues la función objetivo de todos ellos se mantiene estable entorno a un valor durante las últimas épocas de entrenamiento, como se puede observar en las figuras Figura 23, Figura 25, Figura 27 y Figura 29. Además, la función objetivo mantiene un decrecimiento constante en todos los casos, salvo el autoencoder, que presenta un decrecimiento, pero este no es tan uniforme.

Con el fin de facilitar la tarea de análisis de los resultados, se elaboran las siguientes tablas resumen Tabla 11 y Tabla 12, dónde se pueden observar los resultados obtenidos por cada modelo para los valores de precisión y recall, respectivamente.

Precision	Autoencoder	Perceptron	LSTM Stateless	LSTM Statefull
Benign	0,96	0,92	0,98	1
Bot	1*	1	1	1
Brute Force -Web	1*	0,68	0,88	0,91
Brute Force -XSS	1*	1	0,91	0,92
DDOS attack-HOIC	1*	1	1	1
DDOS attack-LOIC-UDP	1*	1	1	1
DoS attacks-GoldenEye	1*	1	1	1
DoS attacks-Hulk	1*	1	1	1
DoS attacks-SlowHTTPTest	1*	0,85	0,97	0,99
DoS attacks-Slowloris	1*	1	1	1
FTP-BruteForce	1*	0,87	0,97	0,98
Infilteration	1*	0,63	0,88	0,89
SQL Injection	1*	0,49	0,62	0,76
SSH-Bruteforce	1*	1	1	1

Tabla 11: Resumen de los resultados obtenidos para la precisión. Nota: los resultados marcados con * representan datos con sesgo.

Recall	Autoencoder	Perceptron	LSTM Stateless	LSTM Statefull
Benign	1*	0,97	0,98	0,98
Bot	0,49	1	1	1
Brute Force -Web	0,6	0,87	0,87	0,94
Brute Force -XSS	0,5	0,52	0,7	0,95
DDOS attack-HOIC	1	1	1	1
DDOS attack-LOIC-UDP	1	1	1	1
DoS attacks-GoldenEye	1	1	1	1
DoS attacks-Hulk	1	1	1	1
DoS attacks-SlowHTTPTest	1	0,98	0,99	1
DoS attacks-Slowloris	1	1	1	1
FTP-BruteForce	1	0,4	0,88	0,98
Infilteration	0,18	0,3	0,86	0,99
SQL Injection	0,52	0,87	0,32	0,52
SSH-Bruteforce	1	1	1	1

Tabla 12: Resumen de los resultados obtenidos para el recall. Nota: los resultados marcados con * representan datos con sesgo.

Solo con la información que nos transmite la guía de colores, donde el verde representa muy buenos resultados, el naranja resultados mediocres y el rojo malos resultados, se puede observar que ambas tablas están teñidas principalmente de verde, reflejando los muy buenos resultados obtenidos. Aunque bien es cierto que no en todos los casos los resultados han sido excelentes, sí se ha conseguido un alto grado de detección de cada ataque con al menos un modelo.

De forma general, se han podido observar las carencias que poseen algunos modelos como los autoencoders, donde ataques cuya actividad en la red es similar al tráfico benigno pueden resultar invisibles para esta red, o la carencia que posee el perceptrón multicapa a la hora de detectar ataques que dependen de la sucesión de muchos flujos.

Por otro lado, los modelos que incluyen contexto, al disponer de información añadida, consiguen aprovecharla para obtener mejores resultados donde los demás clasificadores flaquean. También existe el fenómeno contrario, donde el contexto no aporta ninguna información pues los ataques producen flujos aislados, como es el caso de los ataques de sql injection, con los que el perceptrón multicapa produce buenos resultados y las redes LSTM no consiguen buenas clasificaciones.

Los modelos generados son todo un éxito, especialmente la red recurrente LSTM *stateful*, pues es capaz de detectar el 99% de los flujos malignos, consiguiendo más de un 95% de detección en 12 ataques diferentes. Los resultados de esta red, unidos a los del perceptrón multicapa para los ataques que peor clasifica la red recurrente, concluyen los resultados de este trabajo con una tasa de detección perfecta o cercana al 90% en todos los ataques estudiados.

5 Conclusiones y trabajo futuro

5.1 Conclusiones

Tras un estudio de las soluciones existentes para la detección de ciberataques en red, se han analizado diferentes aproximaciones del uso de aprendizaje profundo para la detección de ciberataques. Esto se ha podido llevar a cabo gracias al *dataset* público CIC-IDS-2018, que incluye muestras de tráfico benigno y gran cantidad de ataques actuales.

Se ha estudiado en profundidad los diferentes atributos y clases de este dataset. Se ha hecho hincapié especialmente en la temporalidad de los datos, pues uno de los objetivos de este trabajo es estudiar si la aportación de contexto a la hora de clasificar mejora los resultados. Sin este análisis temporal, no se hubiese podido concluir que el uso de redes LSTM era una opción viable, y sin esta decisión no se hubiesen podido alcanzar los excelentes resultados obtenidos.

Los resultados finales de la clasificación de los ataques varían según el modelo, consiguiendo entre todos detectar al menos un 85% de las ocurrencias de cada ataque. Obteniendo los mejores resultados con la red LSTM *stateful* que presenta un índice total de detección cercano al 99% y consigue detectar un 90% de la mayor parte de los ataques.

La ventaja de haber realizado varios modelos reside en que el estudio, a pesar de buscar el modelo que mejores resultados obtiene, no excluye el uso de múltiples modelos para un caso de uso real, donde se podrían establecer sistemas de confianza o voto para la detección conjunta entre varias redes, con la intención última de bloquear cualquier intento de ataque.

En definitiva, observando la evolución de las detecciones en función de la cantidad de contexto que posee cada modelo, se puede afirmar que la temporalidad es un factor muy importante para la detección de ciberataques en red, pues cuanta más información tienen los modelos acerca de los flujos anteriores, mejores resultados se han obtenido.

5.2 Trabajo futuro

Una posible continuación a este estudio podría introducir nuevos tipos de ataques, para así estudiar la efectividad de los modelos generados actualmente frente a estos nuevos ataques.

En un apartado más técnico, una mejora sustancial al trabajo vendría de la automatización de:

- La obtención en vivo de los datos de red.
- La generación y procesamiento de los datos.
- La clasificación de los datos mediante los contenidos realizados durante este trabajo.

Respecto a la temporalidad, se podría realizar un estudio más en detalle sobre el impacto del tamaño elegido en la agrupación de flujos en series.

Referencias

[1] https://cybersecuritynews.es/el-cibercrimen-ya-mueve-mas-dinero-que-el-narcotrafico/ consultado el día 2 de junio de 2020.

[2] https://es.wikipedia.org/wiki/Polimorfismo_(malware) consultado el día 2 de junio de 2020.

[3] https://es.wikipedia.org/wiki/WannaCry consultado el día 2 de junio de 2020.

[4] https://www.cylance.com/en-us/index.html consultado el día 2 de junio de 2020.

[5] https://www.watchguard.com/es/wgrd-about/press-releases/watchguard-technologies-lanza-un-antivirus-basado-en-inteligencia consultado el día 8 de julio de 2020.

[6] https://www.darktrace.com/es/ consultado el día 3 de junio de 2020.

[7] https://www.extrahop.com/ consultado el día 8 de julio de 2020.

[8] https://colab.research.google.com/notebooks/intro.ipynb consultado el día 3 de junio de 2020.

[9] https://jupyter.org/ consultado el día 3 de junio de 2020.

[10] https://drive.google.com/ consultado el día 3 de junio de 2020.

[11] https://en.wikipedia.org/wiki/OSI_model consultado el día 3 de junio de 2020.

[12] https://es.wikipedia.org/wiki/Modelo_TCP/IP consultado el día 29 de mayo de 2020.

[13] https://es.wikipedia.org/wiki/Protocolo_de_datagramas_de_usuario consultado el día 29 de mayo de 2020.

[14] https://www.sandvine.com/hubfs/downloads/phenomena/2018-phenomena-report.pdf consultado el día 19 de junio de 2020.

[15] https://es.wikipedia.org/wiki/Sistema_de_detecci%C3%B3n_de_intrusos consultado el día 19 de junio de 2020.

[16] Ferrag, M. A., Maglaras, L., Janicke, H., & Smith, R. (2019, September). Deep learning techniques for cyber security intrusion detection: a detailed analysis. In 6th International Symposium for ICS & SCADA Cyber Security Research 2019 6 (pp. 126-136).

[17] Berman, D. S., Buczak, A. L., Chavis, J. S., & Corbett, C. L. (2019). A survey of deep learning methods for cyber security. Information, 10(4), 122.

[18] https://towardsdatascience.com/understanding-the-different-types-of-machine-learning-models-9c47350bb68a consultado el día 25 de marzo de 2020.

[19] https://es.wikipedia.org/wiki/Perceptr%C3%B3n consultado el día 25 de marzo de 2020.

[20] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

[21] Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2015, June). Gated feedback recurrent neural networks. In International conference on machine learning (pp. 2067-2075).

[22] https://en.wikipedia.org/wiki/Restricted_Boltzmann_machine consultado el día 10 de junio de 2020.

[23] https://www.cisco.com/c/en/us/products/security/common-cyberattacks.html#~how-cyber-attacks-work consultado el día 10 de junio de 2020.

[24] https://owasp.org/www-project-top-ten/ consultado el día 10 de junio de 2020.

[25] https://es.wikipedia.org/wiki/Botnet consultado el día 10 de junio de 2020.

[26] https://www.unb.ca/cic/datasets/ids-2018.html consultado el día 1 de abril de 2020.

[27] https://www.unb.ca/cic/research/applications.html#CICFlowMeter consultado el día 1 de abril de 2020.

[28] https://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/ios-netflow/prod_white_paper0900aecd80406232.html consultado el día 10 de junio de 2020.

[29] https://docs.zeek.org/en/current/script-reference/log-files.html consultado el día 10 de junio de 2020.

[30] https://www.iana.org/assignments/ipfix/ipfix.xhtml consultado el día 10 de junio de 2020.

[31] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html el día 5 de abril de 2020.

[32] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html consultado el día 5 de abril de 2020.

[33] https://scikit-learn.org/stable/index.html consultado el día 1 de abril de 2020.

[34] https://towardsdatascience.com/sampling-techniques-for-extremely-imbalanced-data-part-i-under-sampling-a8dbc3d8d6d8 consultado el día 19 de mayo de 2020.

[35] Yen, S. J., & Lee, Y. S. (2006). Under-sampling approaches for improving prediction of the minority class in an imbalanced dataset. In Intelligent Control and Automation (pp. 731-740). Springer, Berlin, Heidelberg.

[36] https://es.wikipedia.org/wiki/One-hot consultado el día 19 de mayo de 2020.

[37] https://keras.io/ consultado el día 1 de abril de 2020.

[38] https://www.tensorflow.org/guide/keras?hl=es consultado el día 1 de abril de 2020.

[39] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980

[40] https://en.wikipedia.org/wiki/Rectifier_(neural_networks) consultado el día 1 de abril de 2020.

[41] https://es.wikipedia.org/wiki/Funci%C3%B3n_SoftMax consultado el día 1 de abril de 2020.

[42] https://www.tensorflow.org/api_docs/python/tf/keras/losses/CategoricalCrossentropy consultado el día 1 de abril de 2020.

[43] https://keras.io/api/layers/recurrent_layers/lstm/ consultado el día 1 de abril de 2020.

[44] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html consultado el día 11 de abril de 2020.

[45] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html consultado el día 11 de abril de 2020.

Glosario

API	Application Programming Interface
IDS	Intrusion detection system
Flow	Flujo
Overfit	Sobreajuste
Deep learning	Aprendizaje profundo

Anexos

A Atributos del dataset CIC-IDS-2018.

En este anexo se describen de forma detallada todos los atributos del dataset CIC-IDS-2018.

fl_dur	Duración del flujo
tot_fw_pk	Paquetes totales enviados
tot_bw_pk	Paquetes totales recibidos
tot_l_fw_pkt	Tamaño total de los paquetes enviados
fw_pkt_l_max	Tamaño del paquete enviado más grande
fw_pkt_l_min	Tamaño del paquete enviado más pequeño
fw_pkt_l_avg	Tamaño promedio de los paquetes enviados
fw_pkt_l_std	Desviación estándar de los tamaños de los paquetes enviados
Bw_pkt_l_max	Tamaño del paquete recibido más grande
Bw_pkt_l_min	Tamaño del paquete recibido más pequeño
Bw_pkt_l_avg	Tamaño medio de los paquetes recibidos
Bw_pkt_l_std	Desviación estándar de los tamaños de los paquetes recibidos
fl_byt_s	Cantidad de bytes transferidos por segundo
fl_pkt_s	Cantidad de paquetes transferidos por segundo
fl_iat_avg	Tiempo promedio entre dos flujos consecutivos
fl_iat_std	Desviación estándar del tiempo entre dos flujos consecutivos
fl_iat_max	Tiempo máximo entre dos flujos consecutivos
fl_iat_min	Tiempo mínimo entre dos flujos consecutivos
fw_iat_tot	Tiempo total entre paquetes consecutivos enviados
fw_iat_avg	Tiempo medio entre paquetes consecutivos enviados
fw_iat_std	Desviación estándar del tiempo entre dos paquetes consecutivos enviados
fw_iat_max	Tiempo máximo entre dos paquetes consecutivos enviados
fw_iat_min	Tiempo mínimo entre dos paquetes consecutivos enviados
bw_iat_tot	Tiempo total entre paquetes consecutivos recibidos
bw_iat_avg	Tiempo medio entre paquetes consecutivos recibidos
bw_iat_std	Desviación estándar del tiempo entre dos paquetes consecutivos recibidos
bw_iat_max	Tiempo máximo entre dos paquetes consecutivos recibidos
bw_iat_min	Tiempo mínimo entre dos paquetes consecutivos recibidos
fw_psh_flag	Número de veces que el indicador PSH se configuró en enviados (0 para UDP)
bw_psh_flag	Número de veces que el indicador PSH se configuró en paquetes recibidos (0 para UDP)
fw_urg_flag	Número de veces que se estableció el indicador URG en paquetes enviados (0 para UDP)
bw_urg_flag	Número de veces que se estableció el indicador URG en paquetes recibidos (0 para UDP)
fw_hdr_len	Total de bytes utilizados para encabezados de paquetes enviados
bw_hdr_len	Total de bytes utilizados para encabezados de paquetes recibidos
fw_pkt_s	Número de paquetes enviados por segundo
bw_pkt_s	Número de paquetes recibidos por segundo
pkt_len_min	Longitud mínima de un flujo
pkt_len_max	Longitud máxima de un flujo

pkt_len_avg	Longitud media de un flujo
pkt_len_std	Desviación estándar de la longitud del flujo
pkt_len_va	Tiempo mínimo de llegada entre paquetes
fin_cnt	Número de paquetes con FIN
syn_cnt	Número de paquetes con SYN
rst_cnt	Número de paquetes con RST
pst_cnt	Número de paquetes con PUSH
ack_cnt	Número de paquetes con ACK
urg_cnt	Número de paquetes con URG
cwe_cnt	Número de paquetes con CWE
ece_cnt	Número de paquetes con ECE
down_up_ratio	Ratio de descarga y envío
pkt_size_avg	Tamaño promedio de los paquetes
fw_seg_avg	Tamaño promedio observado de los paquetes enviados
bw_seg_avg	Tamaño promedio observado de los paquetes recibidos
fw_byt_blk_avg	Velocidad promedio en envío de bytes por lotes
fw_pkt_blk_avg	Velocidad promedio en envío de paquetes por lotes
fw_blk_rate_avg	Media de la cantidad de lotes enviados
bw_byt_blk_avg	Media de la cantidad de lotes recibidos
bw_pkt_blk_avg	Media de la cantidad de paquetes por lotes recibidos
bw_blk_rate_avg	Media de la cantidad de paquetes por lotes enviados
subfl_fw_pk	El número promedio de paquetes enviados en un subflujo
subfl_fw_byt	El número promedio de bytes enviados en un subflujo
subfl_bw_pkt	El número promedio de paquetes recibidos en un subflujo
subfl_bw_byt	El número promedio de bytes recibidos en un subflujo
fw_win_byt	Número de bytes enviados en la ventana inicial
bw_win_byt	Número de bytes recibidos en la ventana inicial
Fw_act_pkt	Número de paquetes con al menos 1 byte de datos TCP enviados
fw_seg_min	Tamaño mínimo de segmento enviado que se ha observado
atv_avg	Tiempo medio que un flujo estuvo activo antes de volverse inactivo
atv_std	Desviación estándar del tiempo que un flujo estuvo activo antes de volverse inactivo
atv_max	Tiempo máximo que un flujo estuvo activo antes de volverse inactivo
atv_min	Tiempo mínimo que un flujo estuvo activo antes de volverse inactivo
idl_avg	Tiempo medio que un flujo estuvo inactivo antes de activarse
idl_std	Desviación estándar del tiempo que un flujo estuvo inactivo antes de activarse
idl_max	Tiempo máximo que un flujo estuvo inactivo antes de activarse
idl_min	Tiempo mínimo que un flujo estuvo inactivo antes de activarse

B Exploraciones temporales dataset CIC-IDS-2018.

Figuras correspondientes al histograma del tráfico presente entre las 13 y las 15 horas, donde cada barra representa a la cantidad de flujos en un intervalo de 60 segundos.

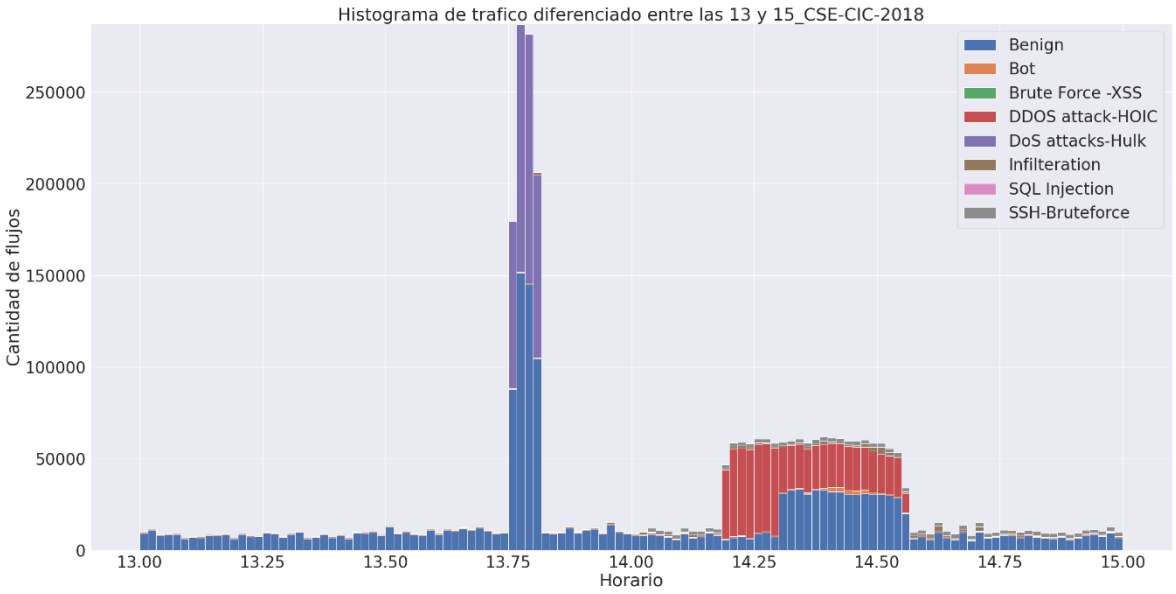

Figura 8: Histograma de los diferentes flujos en el dataset durante las 13 y las 15 horas.

Figuras correspondientes al histograma de cada ataque presente en el dataset, donde cada barra representa a la cantidad de flujos en un intervalo de 60 segundos.

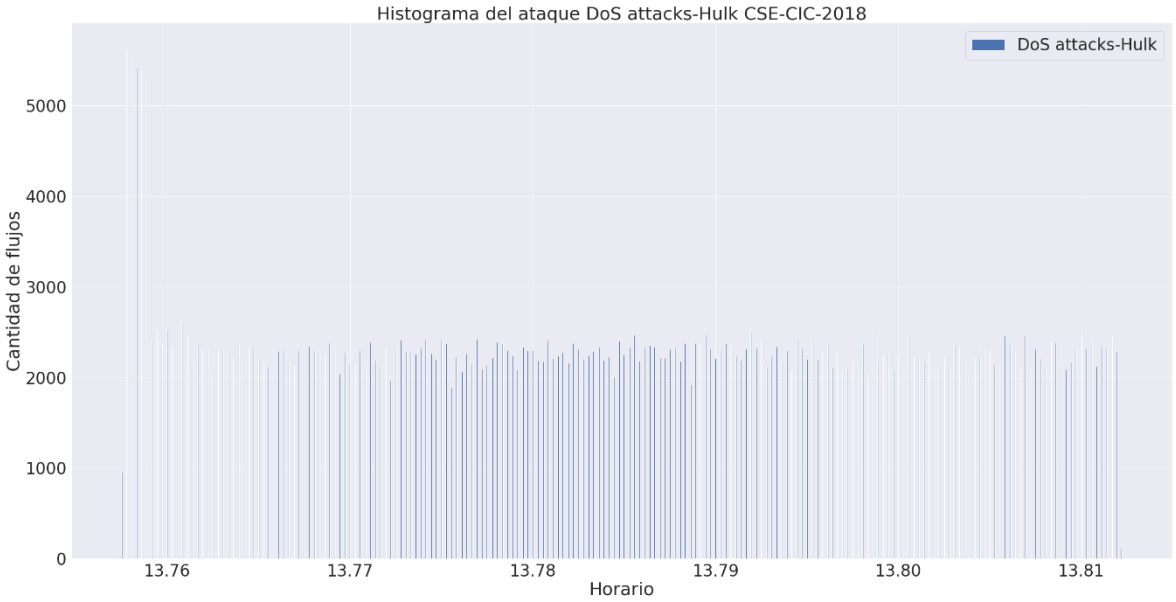

Figura 9: Histograma de los flujos del ataque DoS-Hulk

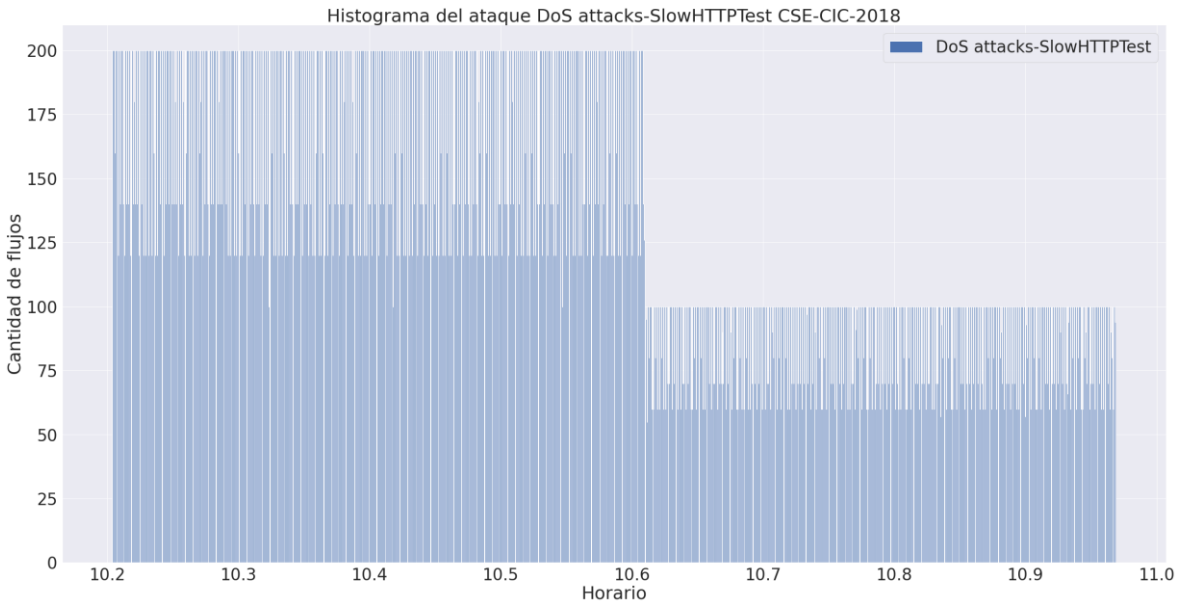

Figura 10: Histograma de los flujos del ataque DoS-SlowHTTPTest

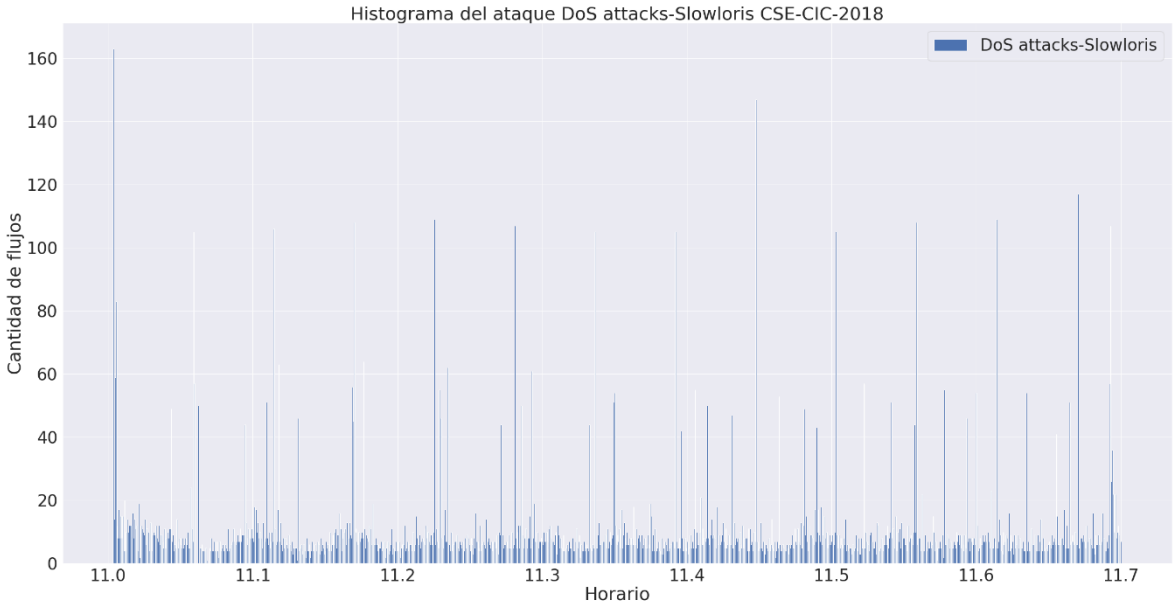

Figura 11: Histograma de los flujos del ataque DoS-Slowloris

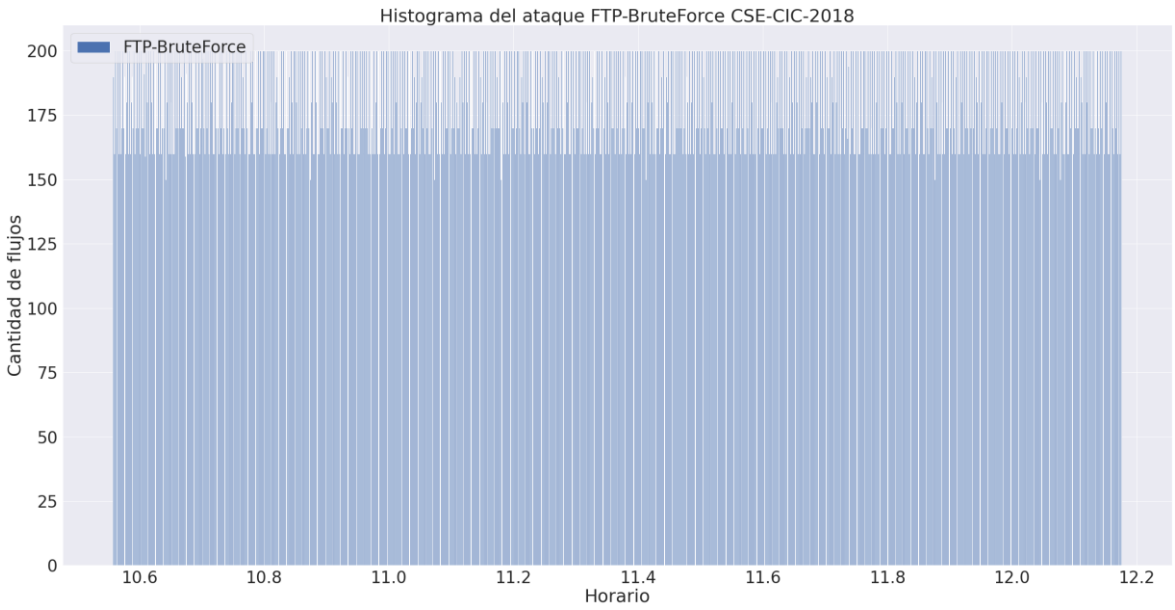

Figura 12: Histograma de los flujos del ataque FTP-BruteForce

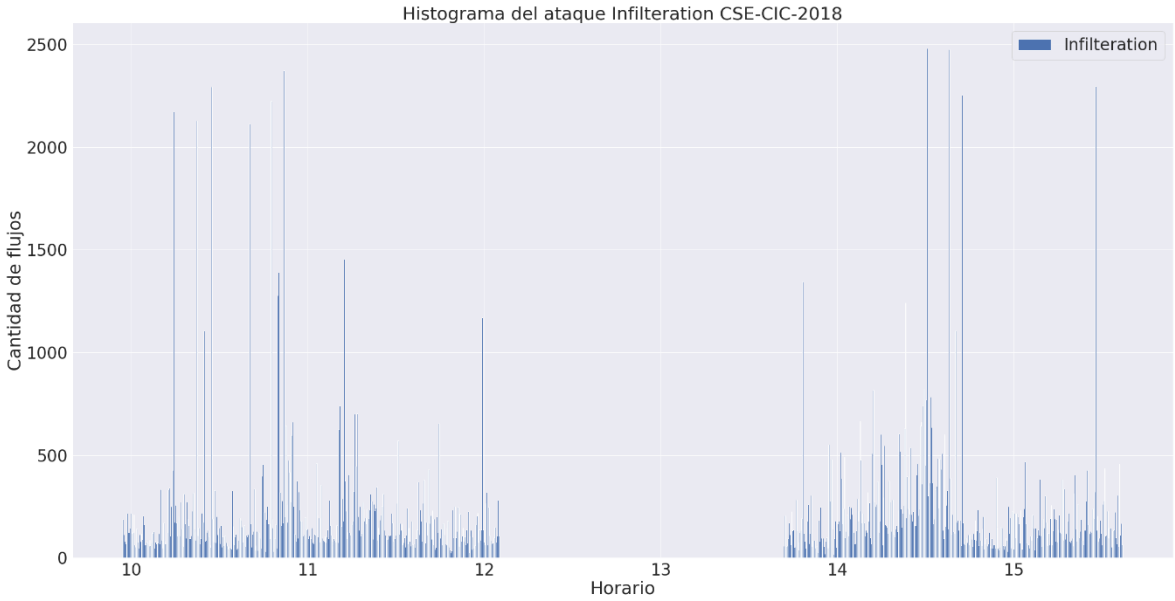

Figura 13: Histograma de los flujos del ataque Infilteration

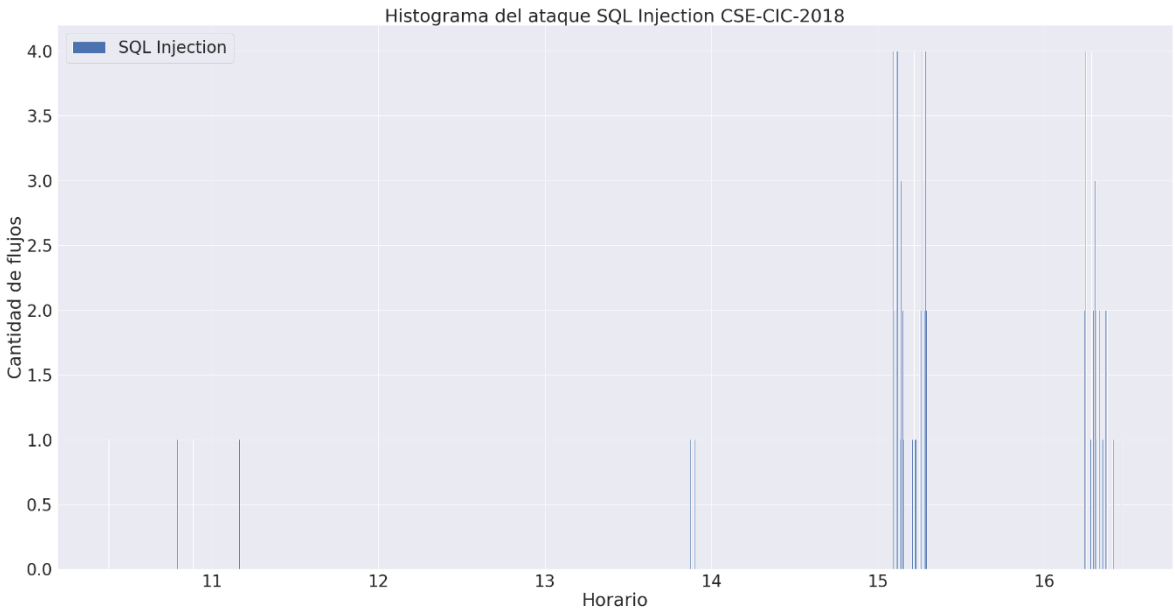

Figura 14: Histograma de los flujos del ataque SQL Injection

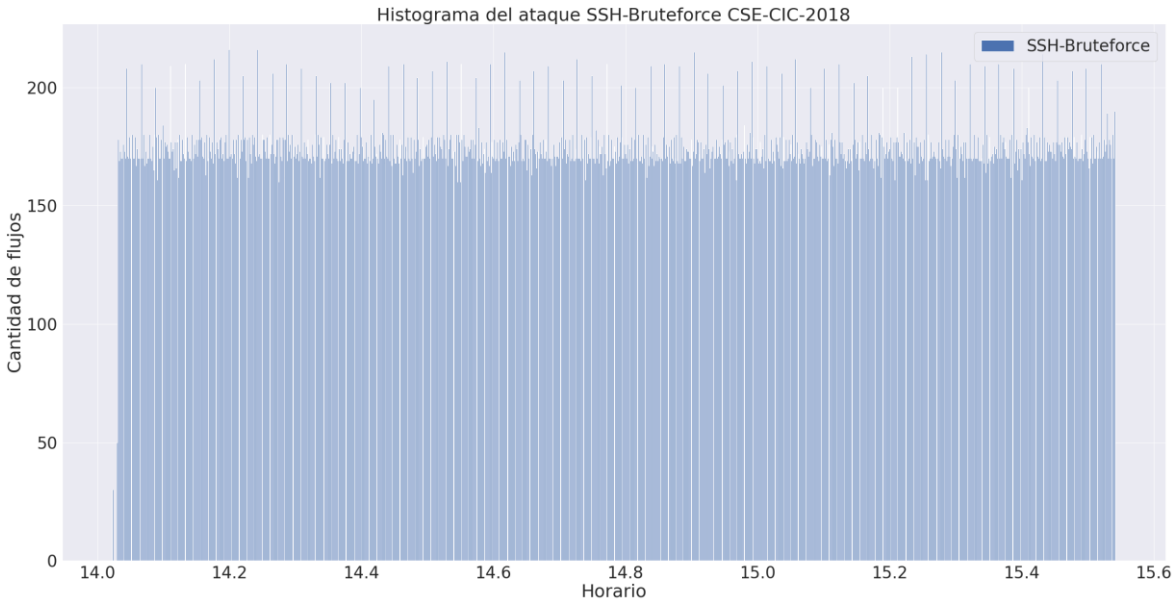

Figura 15: Histograma de los flujos del ataque SSH-Bruteforce

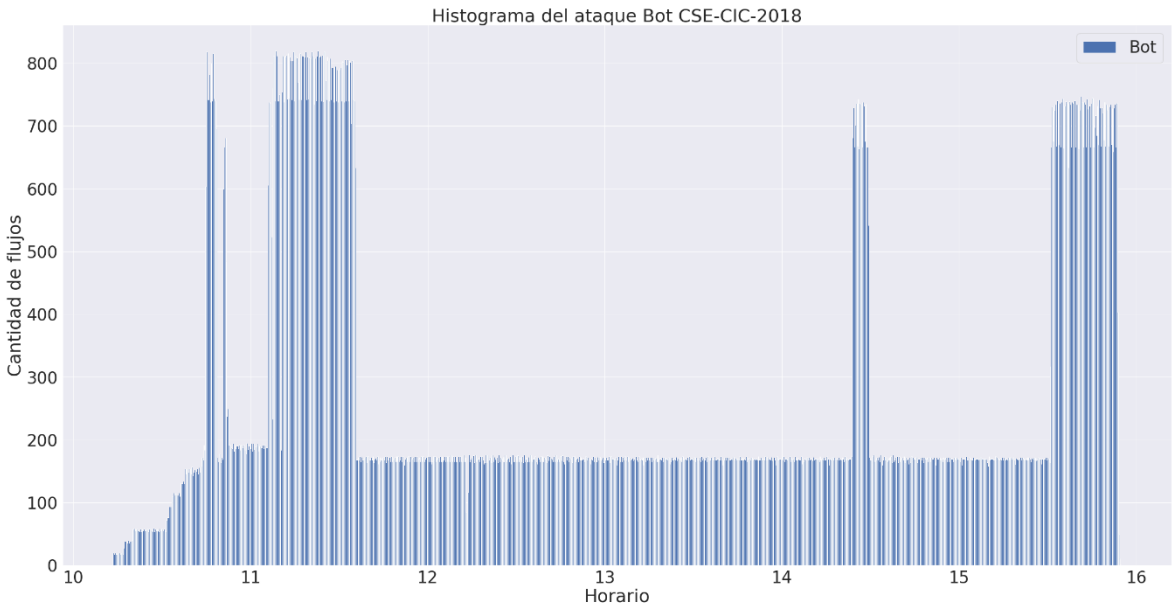

Figura 16: Histograma de los flujos del ataque Bot

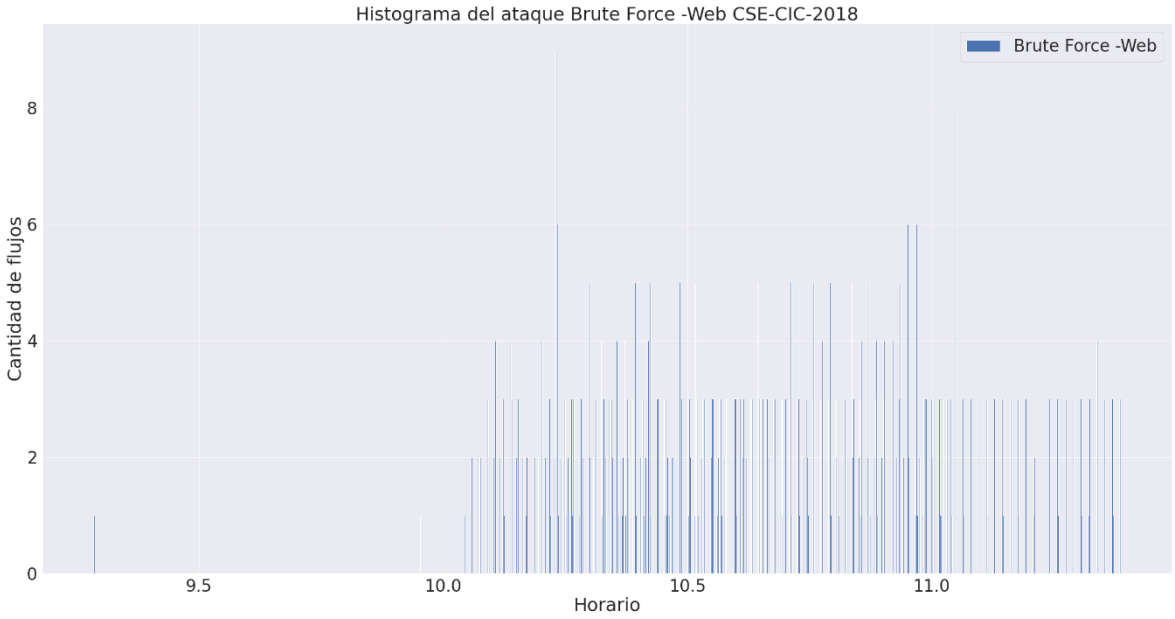

Figura 17: Histograma de los flujos del ataque Brute Force -Web

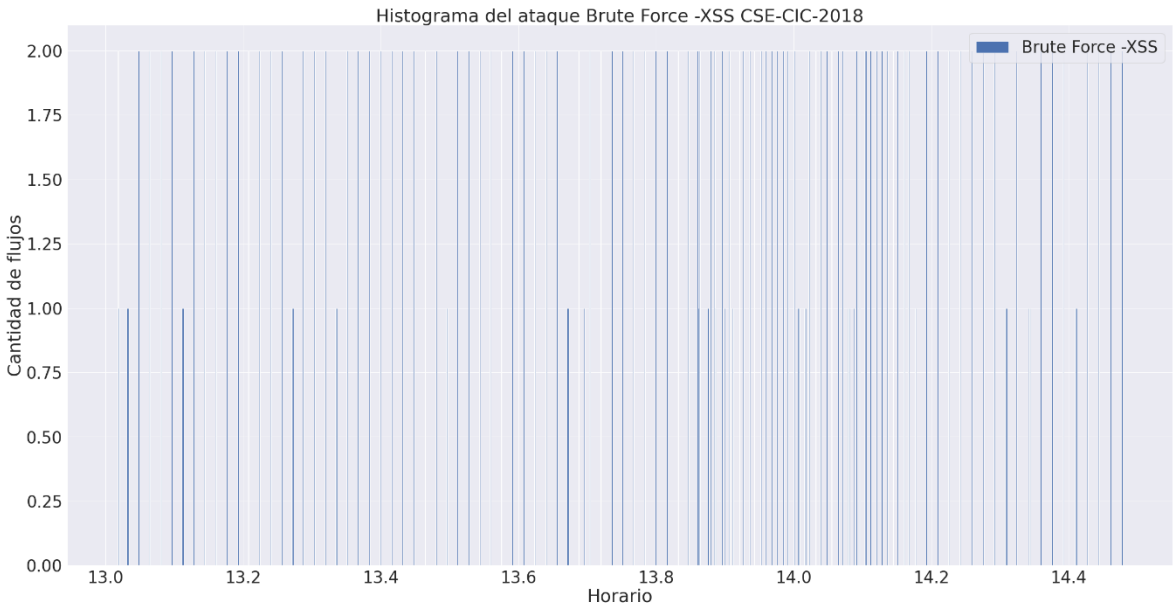

Figura 18: Histograma de los flujos del ataque Brute Force -XSS

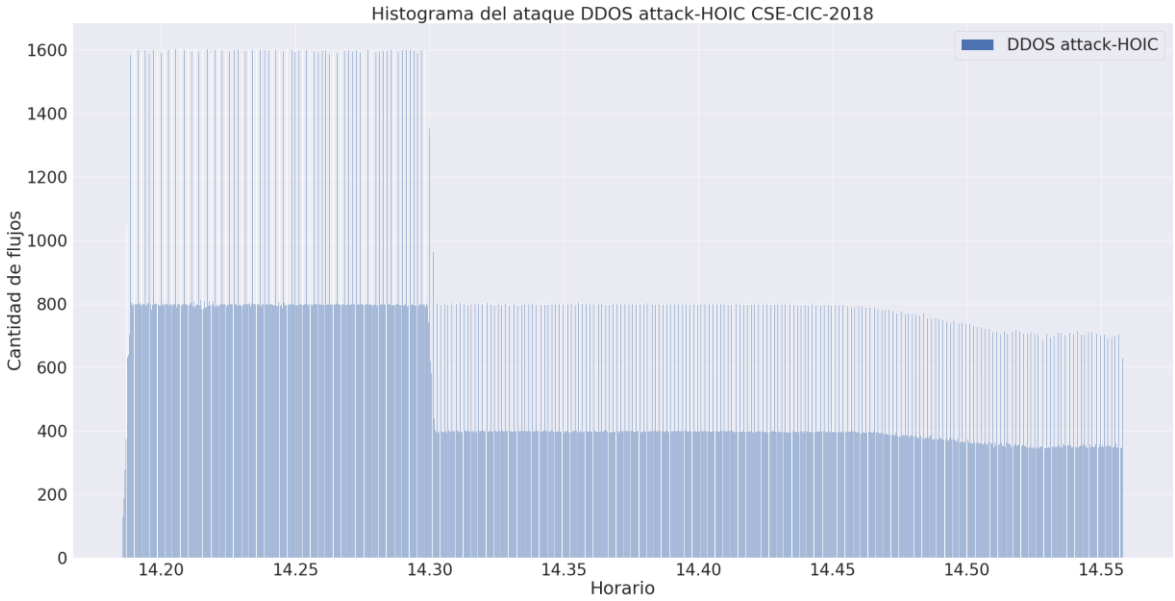

Figura 19: Histograma de los flujos del ataque DDoS-HOIC

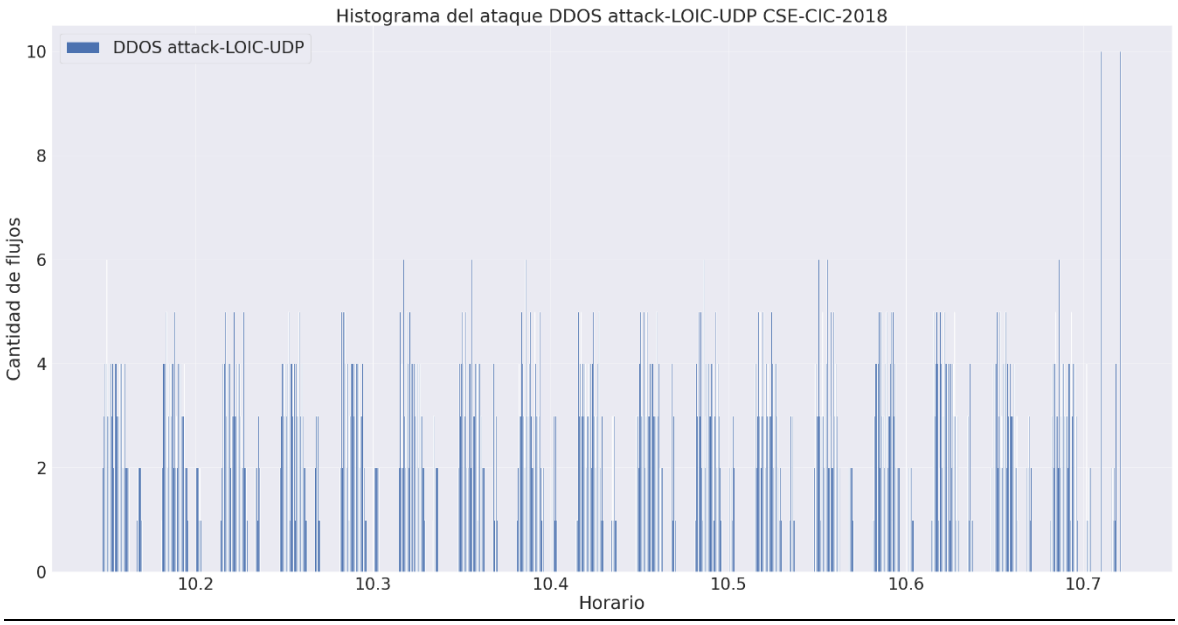

Figura 20: Histograma de los flujos del ataque DDoS-LOIC UDP

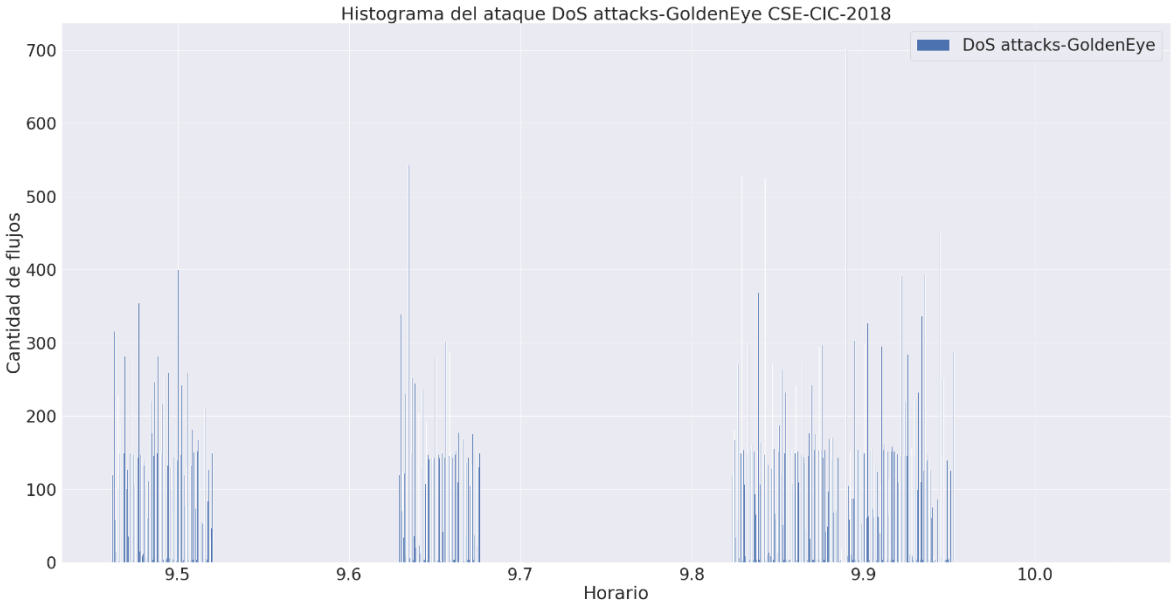

Figura 21: Histograma de los flujos del ataque DoS-GoldenEye

C Resultados

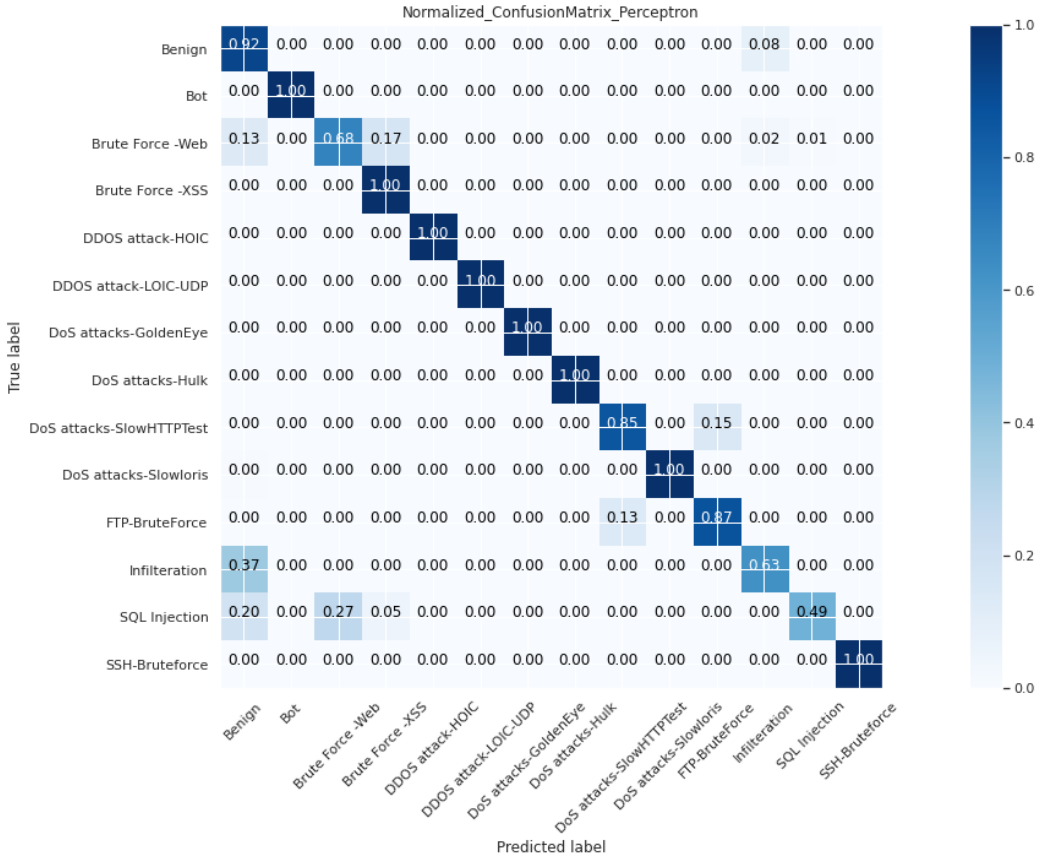

Figura 22: Matriz de confusión del perceptrón multicapa.

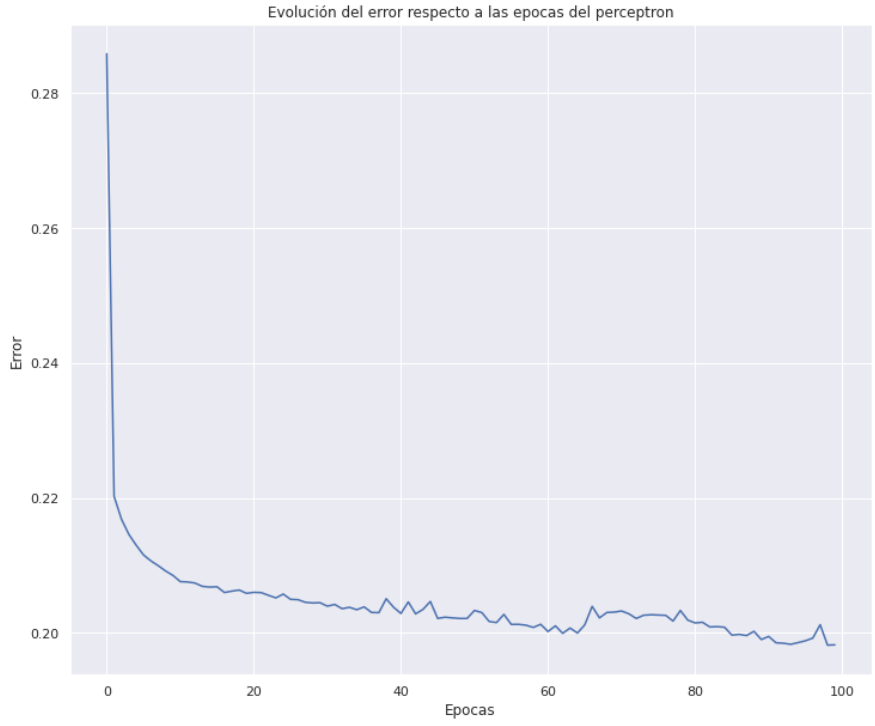

Figura 23: Evolución de la función objetivo respecto a las épocas de entrenamiento del perceptrón multicapa.

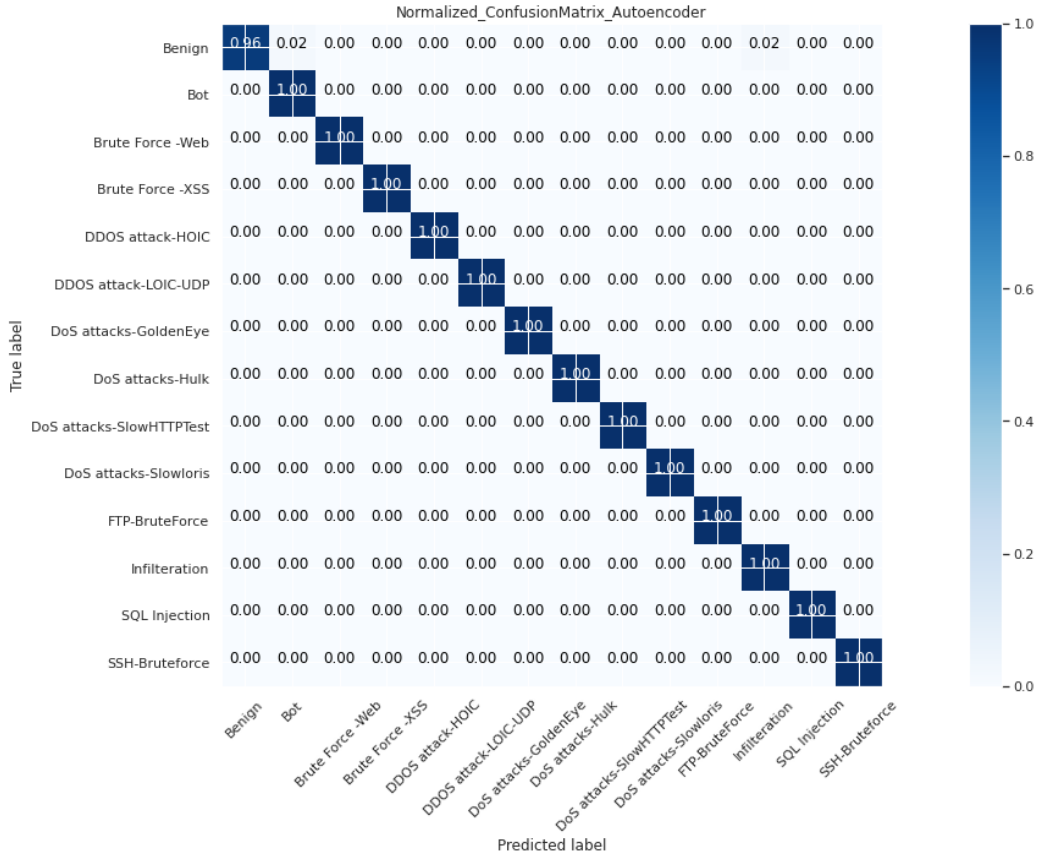

Figura 24: Matriz de confusión del autoencoder.

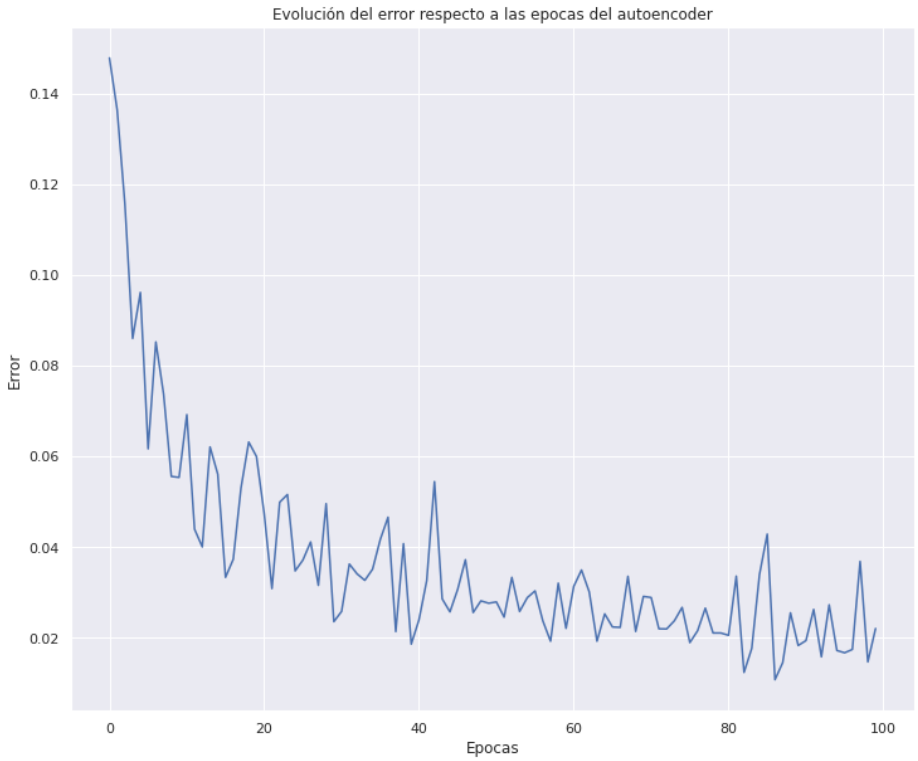

Figura 25: Evolución de la función objetivo respecto a las épocas de entrenamiento del autoencoder.

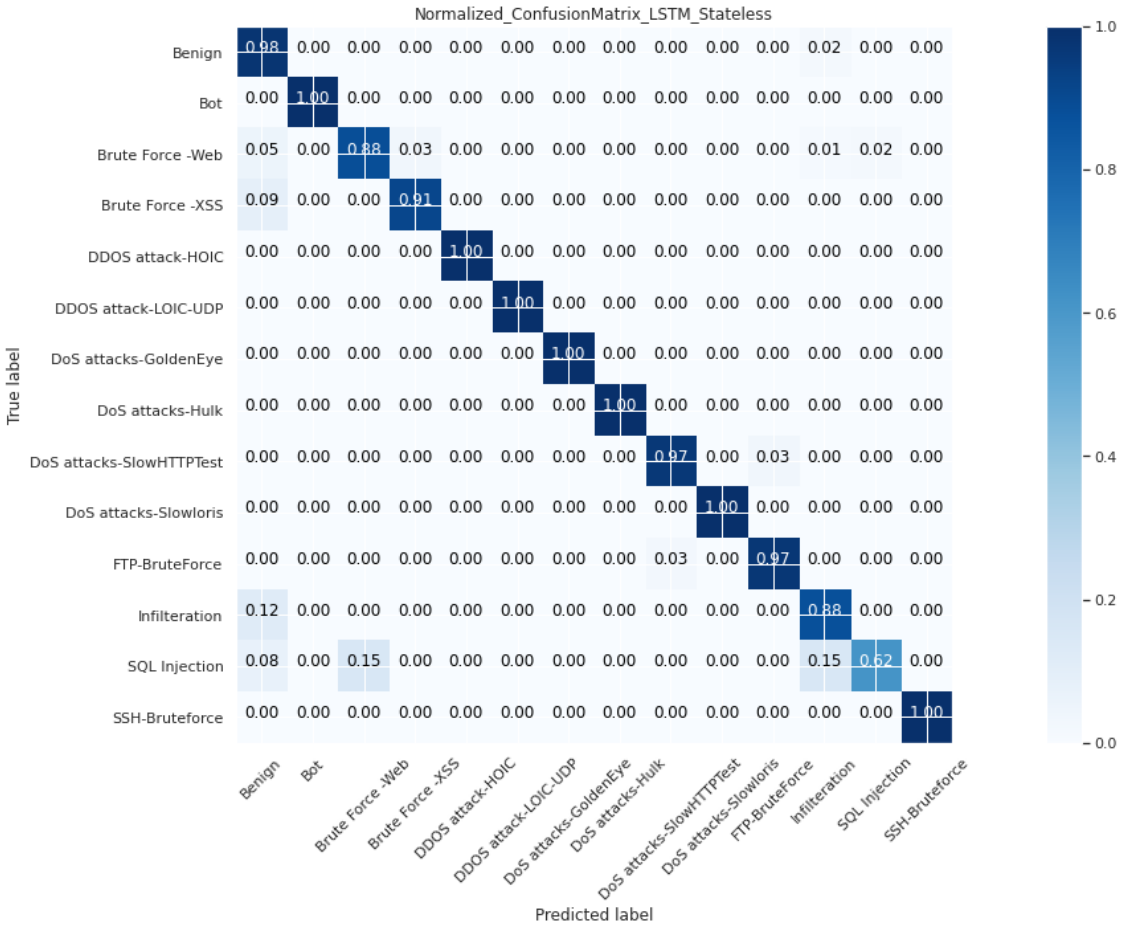

Figura 26: Matriz de confusión de la red LSTM *stateless*.

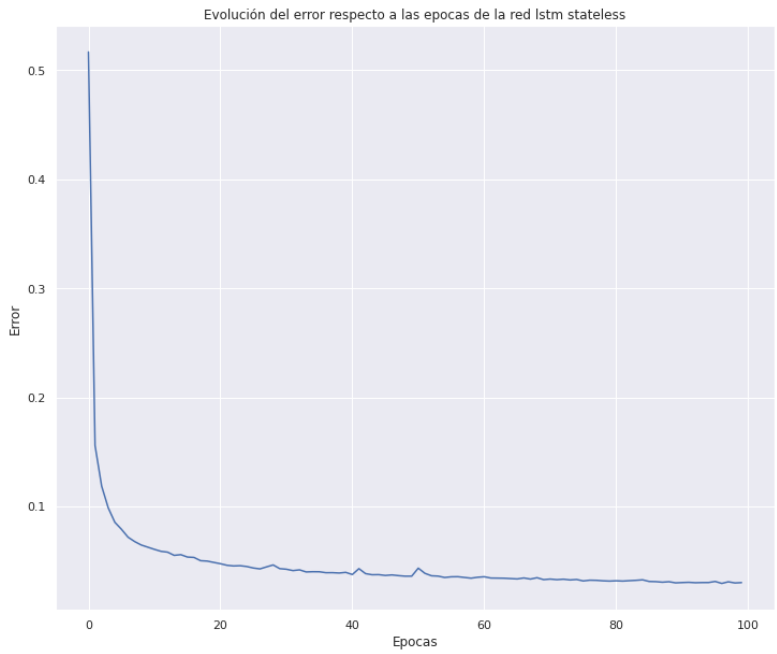

Figura 27: Evolución de la función objetivo respecto a las épocas de entrenamiento de la red LSTM *stateless*.

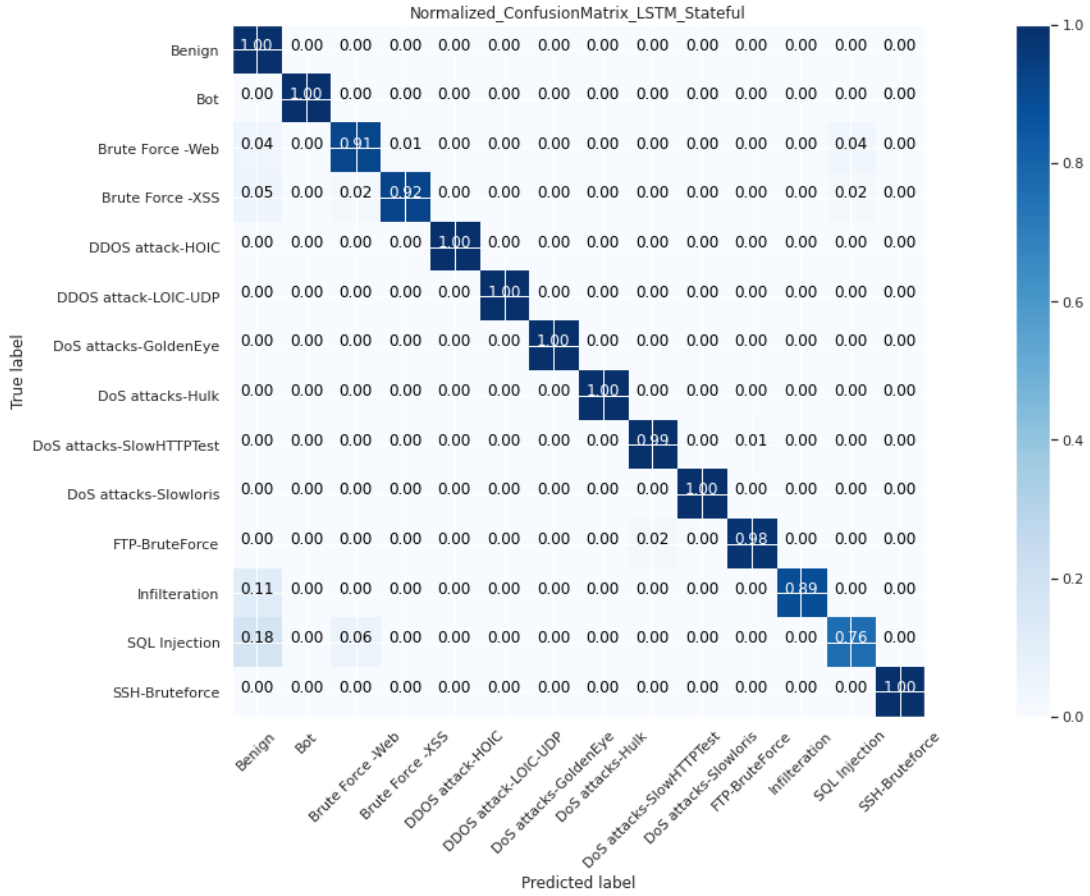

Figura 28: Matriz de confusión de la red LSTM *stateful*.

Figura 29: Evolución de la función objetivo respecto a las épocas de entrenamiento de la red LSTM *stateful*.