UNIVERSIDAD NACIONAL DE CHIMBORAZO
FACULTAD DE INGENIERIA
CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

"Modelo de redes neuronales para la detección y mitigación de ataques de amplificación de DNS en la Unach"

Trabajo de Titulación para optar al título de

Ingeniero en Tecnologías de la Información

Autor:
Cujilema Guananga, Bryan David

Tutor:
Mgs. José Luis Jinez Tapia

Riobamba, Ecuador. 2024

DECLARATORIA DE AUTORÍA

Yo, **Cujilema Guananga Bryan David**, con cédula de ciudadanía **0604752980**, autor del trabajo de investigación titulado: **"MODELO DE REDES NEURONALES PARA LA DETECCIÓN Y MITIGACIÓN DE ATAQUES DE AMPLIFICACIÓN DE DNS EN LA UNACH"**, certifico que la producción, ideas, opiniones, criterios, contenidos y conclusiones expuestas son de mí exclusiva responsabilidad.

Asimismo, cedo a la Universidad Nacional de Chimborazo, en forma no exclusiva, los derechos para su uso, comunicación pública, distribución, divulgación y/o reproducción total o parcial, por medio físico o digital; en esta cesión se entiende que el cesionario no podrá obtener beneficios económicos. La posible reclamación de terceros respecto de los derechos de autor (a) de la obra referida, será de mi entera responsabilidad; librando a la Universidad Nacional de Chimborazo de posibles obligaciones.

En Riobamba, 18 de diciembre del 2024.

Bryan David Cujilema Guananga

C.I: 0604752980

ACTA FAVORABLE - INFORME FINAL DEL TRABAJO DE INVESTIGACIÓN

En la ciudad de Riobamba, a los 08 días del mes de octubre de 2024, luego de haber revisado el informe final del trabajo de investigación presentado por el estudiante **BRYAN DAVID CUJILEMA GUANANGA** con C.C.: **0604752980**, de la carrera de **TECNOLOGÍAS DE LA INFORMACIÓN**, y dando cumplimiento a los criterios metodológicos exigidos, se emite el **ACTA FAVORABLE DEL INFORME FINAL DEL TRABAJO DE INVESTIGACIÓN** titulado "**MODELO DE REDES NEURONALES PARA LA DETECCIÓN Y MITIGACIÓN DE ATAQUES DE AMPLIFICACIÓN DE DNS EN LA UNACH**". Por lo tanto, se autoriza la presentación del mismo para los trámites pertinentes.

Firmado electrónicamente por:
JOSE LUIS JINEZ TAPIA

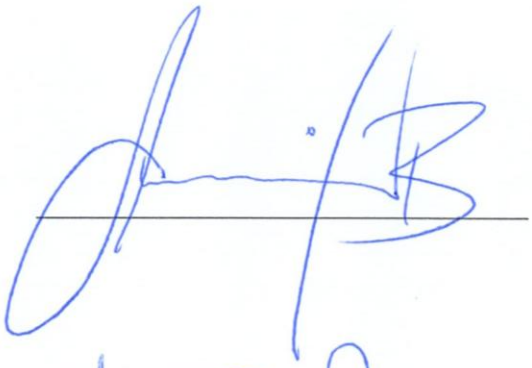
Ing. José Luis Jinez Tapia
TUTOR

CERTIFICADO DE LOS MIEMBROS DEL TRIBUNAL

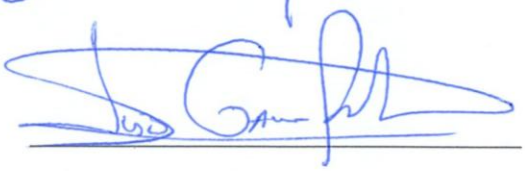
Quienes suscribimos, catedráticos designados Miembros del Tribunal de Grado para la evaluación del trabajo de investigación "MODELO DE REDES NEURONALES PARA LA DETECCIÓN Y MITIGACIÓN DE ATAQUES DE AMPLIFICACIÓN DE DNS EN LA UNACH", presentado por Cujilema Guananga Bryan David, con cédula de identidad número 0604752980, bajo la tutoría de Mgs. José Luis Jinez Tapia; certificamos que recomendamos la APROBACIÓN de este con fines de titulación. Previamente se ha evaluado el trabajo de investigación y escuchada la sustentación por parte de su autor; no teniendo más nada que observar.

De conformidad a la normativa aplicable firmamos, en Riobamba 18 de diciembre del 2024.

Santiago Cisneros, Mgs.
PRESIDENTE DEL TRIBUNAL DE GRADO



Gonzalo Allauca, Mgs.
MIEMBRO DEL TRIBUNAL DE GRADO



Estela Narváez, PhD.
MIEMBRO DEL TRIBUNAL DE GRADO

C E R T I F I C A C I Ó N

Que, **CUJILEMA GUANANGA BRYAN DAVID** con CC: **0604752980**, estudiante de la Carrera **Tecnologías de la Información**, Facultad de **Ingeniería**; ha trabajado bajo mi tutoría el trabajo de investigación titulado **"MODELO DE REDES NEURONALES PARA LA DETECCIÓN Y MITIGACIÓN DE ATAQUES DE AMPLIFICACIÓN DE DNS EN LA UNACH"**, cumple con el **0%**, de acuerdo al reporte del sistema Anti plagio **Turnitin**, porcentaje aceptado de acuerdo a la reglamentación institucional, por consiguiente autorizo continuar con el proceso.

Riobamba, 24 de noviembre de 2024

Firmado electrónicamente por:
JOSE LUIS JINEZ TAPIA

Ing. José Luis Jinez Tapia
TUTOR

DEDICATORIA

A mis queridos padres, cuya dedicación, amor y sacrificio me han brindado la fuerza y el entusiasmo necesarios para lograr mis objetivos. Gracias por ser mi mayor fuente de inspiración incondicional en cada etapa de mi vida.

A mis maestros, quienes con su paciencia, conocimiento y sabiduría me han guiado y motivado a lo largo de este arduo pero gratificante camino. Agradezco profundamente sus consejos y enseñanzas, que han sido esenciales para mi formación académica y profesional.

A mis amigos, por su compañía, amistad y por estar siempre presentes en momentos de alegría y dificultad. Me gustaría expresar mi gratitud por compartir esta experiencia conmigo y por ser un pilar importante en mi vida.

Finalmente, a todas las personas que de alguna manera han contribuido a la realización de este trabajo. Sus aportes y apoyo han sido invaluables.

Bryan David Cujilema Guananga

AGRADECIMIENTO

Agradezco primero a Dios, cuya guía y bendiciones me han dado la fortaleza y la fe necesarias para superar cada obstáculo y lograr esta meta. Su presencia ha sido siempre una fuente de inspiración y esperanza.

A mis padres, quienes con su amor incondicional, sacrificios y constante apoyo han sido el pilar fundamental de mi vida. Gracias por confiar en mí a pesar de mis incertidumbres y por ser mi principal fuente de motivación.

A mis maestros y tutor, quienes con su paciencia, dedicación y sabiduría me han brindado las herramientas y conocimientos necesarios para mi crecimiento académico y profesional. Su ayuda y orientación han sido esenciales para mi desarrollo y aprendizaje.

A mis amigos, por su constante compañía, amistad y por compartir esta travesía conmigo. Gracias por estar siempre presentes, por las risas y por los momentos de apoyo en tiempos difíciles.

Y finalmente, a mí mismo, por la perseverancia, el esfuerzo y la dedicación que he puesto en cada paso de este camino. Reconozco mi propio trabajo y dedicación, que han sido esenciales para alcanzar este logro.

Bryan David Cujilema Guananga

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El presente trabajo de investigación se centra en el desarrollo de un modelo de redes neuronales para la detección y mitigación de ataques de amplificación de DNS, un tipo de ciberataque que puede comprometer la seguridad de las redes informáticas. A través de una investigación experimental, se diseñó y entrenó un modelo de aprendizaje automático que permite identificar el tráfico malicioso en tiempo real.

La investigación se llevó a cabo en un entorno simulado utilizando GNS3, donde se implementaron diversas técnicas de recolección de datos y se definieron las variables clave para el análisis. Las variables identificadas fueron el protocolo, la longitud y la información del paquete, las cuales fueron recolectadas mediante la herramienta Wireshark para su posterior análisis. Los resultados obtenidos demostraron que el modelo tiene una alta capacidad de generalización, con una precisión cercana al 99.8% en la clasificación de tráfico normal y de ataque, indicando su efectividad en la detección de amenazas.

Este estudio no solo contribuye al campo de la ciberseguridad, sino que también proporciona una base sólida para futuras investigaciones en la mitigación de ataques informáticos, destacando la importancia de utilizar tecnologías avanzadas como las redes neuronales en la protección de infraestructuras críticas.

Palabras clave: Ciberseguridad, Red Neuronal, Detección, Mitigación, Simulación, Tráfico, GNS3, Wireshark.

ABSTRACT

This research focuses on the development of a neural network model for detecting and mitigating DNS amplification attacks, a type of cyberattack that can compromise the security of computer networks. Through experimental research, a machine learning model was designed and trained to identify malicious traffic in real time.

The study was conducted in a simulated environment using GNS3, where various techniques were implemented to collect relevant data. The key variables considered for the model's development were the protocol, packet length, and packet information. These data were obtained using the Wireshark tool and subsequently processed for analysis. The results showed that the model has a high generalization capability, achieving 99.8% accuracy in distinguishing between normal and malicious traffic, demonstrating its effectiveness in threat detection.

This study not only significantly contributes to the field of cybersecurity but also provides a solid foundation for future research on mitigating cyberattacks. Furthermore, it highlights the importance of utilizing advanced technologies, such as neural networks, in protecting critical infrastructures and ensuring the security of computer networks.

Keywords: cybersecurity, neural network, detection, mitigation, simulation, traffic, GNS3, Wireshark.

Reviewed by:
Mg. Lourdes del Rocío Quinata Encarnación
ENGLISH PROFESSOR
C.C 1803476215

CAPÍTULO I. INTRODUCCIÓN

En el panorama actual de la tecnología y la conectividad, la seguridad en línea se ha convertido en una preocupación cada vez más relevante. Entre las amenazas cibernéticas más comunes y devastadoras se encuentran los ataques de Denegación de Servicio (DoS) y los ataques de Denegación de Servicio Distribuido (DDoS). Estos ataques, aunque distintos en su ejecución, comparten un objetivo en común de interrumpir o inutilizar los servicios en línea de una organización o individuo [1].

Los ataques DoS/DDoS (Denegation of Service/Denegación de Servicio) son ataques dirigidos por múltiples computadoras comprometidas, llamadas bots o zombies, que se enfocan en un solo sistema. Su motivación es hacer que el sistema objetivo o la red agote sus recursos mediante la inundación de peticiones, con el objetivo de que el servicio se vea obstaculizado o detenido, llevando a la indisponibilidad del servicio [1].

Todos los ataques de amplificación aprovechan una disparidad en el consumo de ancho de banda entre un atacante y el recurso web objetivo. Cuando la disparidad en el costo se multiplica a través de muchas solicitudes, el volumen de tráfico resultante puede perturbar la infraestructura de red. Enviar consultas breves que derivan en extensas respuestas permite al atacante sobrecargar su objetivo con menos esfuerzo. Si se multiplica este aumento haciendo que cada bot en una botnet (red de bots) realice solicitudes similares, el atacante evita ser descubierto y se beneficia plenamente de un aumento significativo del tráfico de ataques [2].

En este contexto, las redes neuronales ofrecen un enfoque prometedor para detectar y mitigar ataques de amplificación de DNS en ataques DDoS. Las redes neuronales son algoritmos de aprendizaje automático que pueden analizar y aprender patrones complejos en los datos de tráfico, particularmente adecuadas para detectar anomalías y comportamientos maliciosos.

El objetivo de esta investigación es utilizar redes neuronales para detectar y mitigar ataques de amplificación de DNS en el contexto de ataques DDoS. Se explorará el desarrollo de modelos de redes neuronales capaces de analizar el tráfico e identificar patrones asociados con ataques mejorados. Además, en función de los resultados de la detección de ataques, se explorarán estrategias de mitigación para mejorar la seguridad de la red y proteger contra dichas amenazas cibernéticas.

1.1 Planteamiento del problema

La creciente sofisticación y frecuencia de los ciberataques plantean grandes desafíos para la seguridad de las redes. Los ataques de denegación de servicio (DoS) y los ataques DDoS (denegación de servicio distribuido) son solo una parte de las amenazas que enfrentan las organizaciones y los individuos en el entorno digital actual. El propósito de estos ataques es interrumpir los servicios en línea saturando los recursos del servidor de destino o sobrecargando la red, resultando en la indisponibilidad del servicio y una experiencia de usuario degradada.

La variante más insidiosa de estos ataques son los ataques de amplificación de DNS, que explotan las vulnerabilidades de los servidores DNS para aumentar el tráfico dirigido a un objetivo deseado. Estos ataques son motivo de especial preocupación debido a su capacidad para sobrecargar la infraestructura de red de la víctima con tráfico excesivo y desestabilizador. Al utilizar técnicas como la explotación de servidores DNS abiertos y la manipulación de protocolos de Internet como NTP, SSDP y Memcached, los atacantes pueden amplificar el impacto de sus acciones, llevando a un ataque de denegación de servicio eficaz.

A pesar de los esfuerzos continuos de la comunidad de la ciberseguridad para mitigar estos ataques, las soluciones existentes aún presentan limitaciones importantes. A medida que evolucionan las tácticas de los atacantes, los métodos tradicionales de detección y mitigación pueden resultar ineficaces, dejando a las organizaciones vulnerables a interrupciones y pérdidas financieras.

En este contexto, la necesidad de desarrollar métodos innovadores y eficaces para detectar y mitigar los ataques de amplificación de DNS es cada vez más urgente. La investigación en esta área es esencial para mejorar la ciberresiliencia y proteger la integridad de los servicios en línea. Es fundamental explorar nuevas tecnologías y métodos, como el uso de redes neuronales, que puedan proporcionar una detección más precisa y una respuesta más rápida a estas amenazas emergentes.

1.2 Justificación

La investigación propuesta se centrará en el diseño e implementación de un modelo de detección basado en redes neuronales, aprovechando su capacidad para analizar y aprender patrones complejos en los datos de tráfico. Este enfoque permitirá una detección eficaz de los ataques de amplificación de DNS en tiempo real, lo que constituye una contribución significativa en la lucha contra estas amenazas cibernéticas.

Al evaluar la efectividad del modelo de detección en diferentes escenarios de ataque y utilizando métricas relevantes de evaluación como Precision, F1-Score, Exhaustividad, Escpecificidad y Tasa de falsos positivos. Además, al llevar a cabo la investigación dentro

de la Universidad Nacional de Chimborazo (UNACH), se realizará un análisis exhaustivo de los tipos de ataques de amplificación de DNS y de las técnicas utilizadas por los atacantes, así como una revisión de las soluciones actuales para la detección y mitigación de estos ataques.

En última instancia, se espera que esta investigación contribuya al desarrollo de nuevas técnicas y herramientas para mejorar la seguridad de las redes contra los ataques de amplificación de DNS, beneficiando no solo a la comunidad académica, sino también a la sociedad en general, al proteger los servicios en línea y garantizar una experiencia de usuario segura y confiable.

1.3 Formulación de la pregunta

¿Cómo la aplicación del modelo de redes neuronales incide en la efectividad para detectar y mitigar los ataques de amplificación de DNS?

1.4 Objetivos

Objetivo general

Desarrollar un modelo de redes neuronales para la detección y mitigación de ataques de amplificación de DNS.

Objetivos específicos

- Investigar un modelo de redes neuronales para la detección y mitigación de patrones asociados con ataques de amplificación de DNS.
- Implementar el modelo en un entorno simulado.
- Validar la efectividad del modelo de redes neuronales para la detección y mitigación de ataques de amplificación de DNS con base a la herramienta Apache Spark.

CAPÍTULO II. MARCO TEÓRICO

2.1 Redes Neuronales

Es un sistema informático diseñado para simular la capacidad de procesamiento de información del cerebro humano. Estas redes neuronales artificiales son parte de una familia de herramientas de inteligencia artificial (IA) que pueden resolver problemas que los humanos no pueden resolver. Estos sistemas, sumados al aprendizaje automático, pueden mejorar la calidad de los resultados obtenidos al extraer la información a procesar [3].

2.1.1 Tipos de redes neuronales

Las redes neuronales artificiales se dividen en diferentes tipos según su arquitectura y tienen aplicaciones diversas. La red neuronal más antigua fue creada por Frank Rosenblatt en 1957. Las redes neuronales de avance, también conocidas como perceptrones multicapa (MLP), están compuestas por una capa de entrada, una capa de salida y una o más capas ocultas, y sus neuronas tienen una forma característica en "S". Las redes neuronales recurrentes (RNN), por otro lado, se reconocen por sus circuitos de retroalimentación y se utilizan en tareas secuenciales o de corta duración, como las predicciones de bolsa o las previsiones de ventas de cadenas de tiendas. Finalmente, las redes neuronales convolucionales (ConvNet o CNN) están formadas por capas convolucionales, capas de fusión y capas completamente conectadas (FC) y se basan en principios del álgebra lineal [4], como se visualiza en la Figura 1.

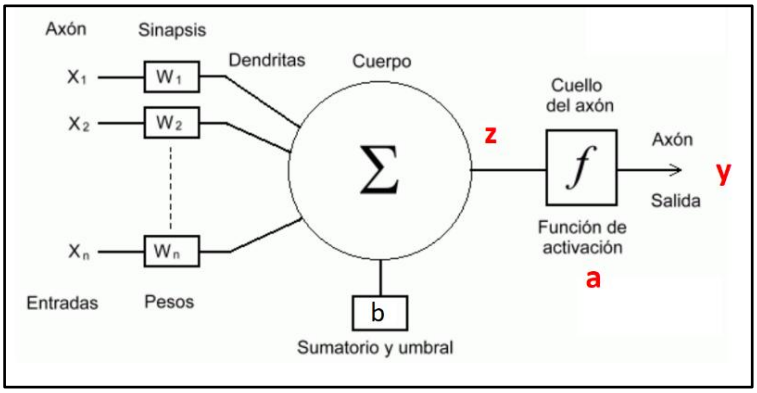


Figura 1. Diagrama de neurona de McCulloch
Fuente: [4]

2.1.2 Arquitecturas comunes de redes neuronales

Una arquitectura de red neuronal convolucional típica utiliza capas de agrupación para reducir el tamaño de los mapas de activación; de lo contrario, no se pueden ejecutar en la GPU. Además, ayuda a reducir el gasto excesivo. Los dos tipos de pooling de la arquitectura de red más comunes son max-pooling que calcula el máximo de los elementos y el averag-pooling que calcula la media de los elementos [5]. A continuación, como se aprecia en la Figura 2 el diagrama de la arquitectura de una red neuronal.

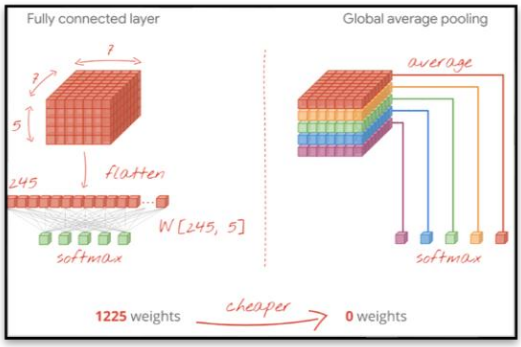


Figura 2. Arquitectura de una red neuronal
Fuente: [5]

2.2 Fundamentos de los ataques de amplificación de DNS

2.2.1 Descripción de los ataques de amplificación de DNS

Los ataques de amplificación ocurren cuando un atacante utiliza una vulnerabilidad en un servidor DNS para amplificar el tráfico malicioso enviado a un objetivo utilizando direcciones IP falsificadas o IP de origen falsificadas. En un escenario típico, un atacante envía una serie de solicitudes a diferentes servidores de nombres utilizando la dirección IP de destino como dirección de "retorno". Estos servidores envían respuestas a la dirección IP de forma secuencial. Dado que la respuesta es mucho mayor que la solicitud original, genera más tráfico, haciendo que el objetivo se vea abrumado con un tráfico que puede ser 50 veces mayor que la solicitud original [6].

2.2.2 Ataque NXDOMAIN

Este es un tipo de ataque de inundación de DNS en el que un atacante envía solicitudes a un servidor DNS para obtener registros que no existen, provocando un ataque de denegación de servicio en el tráfico legítimo. Esto se puede lograr utilizando sofisticadas herramientas de ataque que generan automáticamente subdominios únicos para cada solicitud. Los ataques NXDOMAIN también pueden apuntar a analizadores recursivos para llenar el caché del analizador con solicitudes de spam [7].

2.2.3 Impacto en la seguridad de la red

La ciberseguridad es un área de la seguridad de la red que se centra en proteger las redes informáticas de las amenazas en línea. La ciberseguridad tiene tres objetivos principales: evitar el acceso no autorizado a los recursos de la red, detectar y bloquear ciberataques y violaciones de seguridad en curso, y garantizar que los usuarios autorizados puedan acceder de forma segura [8].

2.3 Métodos convencionales de detección y mitigación

2.3.1 Mitigación de ataques de amplificación de DNS

Los recursos legales están limitados para individuos o empresas que tienen sitios web o servicios en línea. La razón es que, aunque los servidores personales pueden ser objetivos de ataques, no son el efecto principal de los ataques de volumen. Debido a la gran cantidad de tráfico generado, la infraestructura circundante al servidor será la más afectada. Es posible que su proveedor de servicios de Internet (ISP) u otro proveedor de infraestructura no pueda manejar el tráfico entrante sin verse abrumado. Además de los servicios de protección remota como Cloudflare DDoS Protection, las estrategias de mitigación son principalmente soluciones preventivas de infraestructura de red [9].

2.3.2 Cómo funciona un ataque de amplificación de DNS

Los ataques de amplificación de DNS aprovechan la diferencia en el ancho de banda entre el atacante y el objetivo, enviando múltiples solicitudes pequeñas que generan respuestas extensas, saturando así la infraestructura de red del objetivo y provocando un ataque de denegación de servicio [10], como se visualiza en la Figura 3.

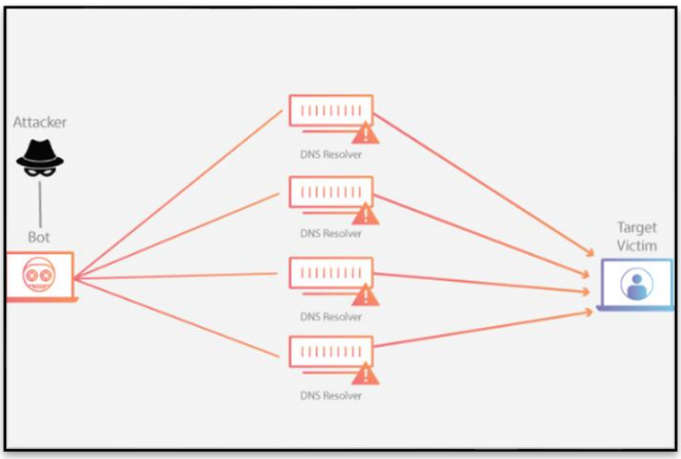


Figura 3. Ataque de amplificación de DNS
Fuente: [10]

2.4 Aplicaciones de Redes Neuronales en Ciberseguridad

2.4.1 Uso de redes neuronales en la detección de intrusiones

Un sistema de detección de intrusos (IDS) es una herramienta que refuerza la seguridad al examinar el tráfico de red para detectar actividad no deseada y generar alertas para sus sistemas o administradores de red, siguiendo una política segura de confidencialidad, integridad y disponibilidad. Estos sistemas combinan medidas de autenticación de nodos de red con controles de acceso que autorizan las conexiones que cada nodo intenta realizar [11].

2.4.2 Aplicaciones exitosas de redes neuronales en ciberseguridad

La seguridad inteligente combina varios aspectos del aprendizaje automático, la inteligencia artificial y su aplicación a la seguridad tradicional es una tendencia innovadora en los últimos años. Como dice Panda Security, estas herramientas pueden adaptarse mejor a las nuevas amenazas y proteger nuevos tipos de aplicaciones. Como ventaja, aprende sobre la marcha y permite desarrollar nuevos criterios de clasificación sin intervención humana. Por ejemplo, se utiliza para combatir el malware y el fraude en línea, ya que los ciberdelincuentes evolucionan rápidamente y crean amenazas que se adaptan a la seguridad del sistema. Por lo tanto, el aprendizaje profundo es capaz de detectar y clasificar estas amenazas y brindar soluciones de manera eficiente y rápida [12].

2.5 Aplicaciones de redes neuronales en ciberseguridad

2.5.1 Implementaciones de redes neuronales en sistemas de detección de intrusiones

Los sistemas NID se ocupan del flujo de información entre servidores y clientes. Estos dispositivos, a menudo llamados rastreadores de paquetes, interceptan paquetes enviados a través del medio de comunicaciones y transmiten los datos encapsulados en varios protocolos como Frame Relay o ATM (Modo de transferencia asíncrono). Algunos dispositivos NID comparan paquetes con una firma de ataque conocida y una base de datos de huellas digitales de paquetes maliciosos, mientras que otros analizan el comportamiento de los paquetes en busca de comportamientos inusuales que puedan ser maliciosos. En ambos casos, los dispositivos IDS deben considerarse principalmente como protección del perímetro de la red [13].

2.5.2 Uso de redes neuronales en la identificación de vulnerabilidades de seguridad

Las redes neuronales tienen una amplia gama de aplicaciones prácticas, incluida la detección de intrusiones, la ciencia forense digital y el reconocimiento de patrones sospechosos. En particular, las redes neuronales se pueden utilizar para detectar y prevenir ciberataques. Una de las aplicaciones más populares de las redes neuronales en ciberseguridad es el análisis de comportamiento. Las redes neuronales pueden analizar grandes cantidades de datos, como registros de auditoría y tráfico de red, para identificar patrones que indiquen ataques. Por ejemplo, las redes neuronales pueden detectar patrones de tráfico de red inusuales que no corresponden al comportamiento típico de los usuarios o de la organización. El malware puede ser difícil de detectar porque a menudo está diseñado para interferir con el software antivirus tradicional y evitar la detección [14]. Como se indica en la Figura 4 la red neuronal simple.

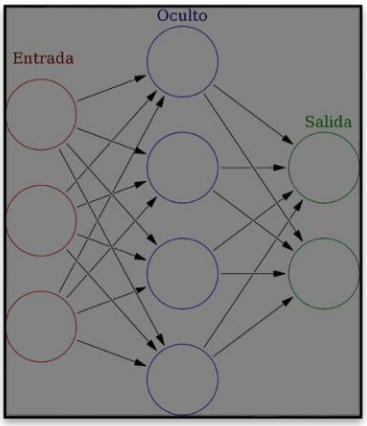


Figura 4. Diagrama de una red neuronal multicapa
Fuente: [14]

2.5.3 Escaneo de Vulnerabilidades

Esta tecnología de seguridad de red permite identificar evaluar servicios y aplicaciones de infraestructura para detectar posibles debilidades en los sistemas de red. El proceso consiste en probar redes, aplicaciones y sistemas informáticos en busca de fallos de seguridad que puedan ser explotados por ciberatacantes. Las debilidades detectadas incluyen software desactualizado, configuraciones incorrectas o la falta de medidas de seguridad adecuadas. Al realizar un análisis de este tipo, una organización puede obtener una comprensión detallada del estado de seguridad de la infraestructura de red en un momento dado. Por ello, se recomienda realizar estos análisis de manera periódica para detectar nuevas amenazas y reducir los riesgos. El objetivo final es promover una estrategia de ciberseguridad más sólida y flexible, lo cual permite una mejor comprensión de los riesgos presentes en la red y la implementación de medidas preventivas antes de que tengan un impacto negativo [15].

2.6 Modelo de redes neuronales para detección de amenazas

2.6.1 Diseño y arquitectura del modelo de redes neuronales

Una red neuronal es un modelo simplificado de cómo el cerebro humano procesa la información, combinando una gran cantidad de unidades de procesamiento interconectadas que actúan como versiones abstractas de neuronas. Estas unidades están organizadas jerárquicamente y se dividen en tres partes principales una capa de entrada, una o más capas ocultas y una capa de salida.

En la capa de entrada, las unidades representan las variables de los datos de entrada, estas unidades están conectadas a las neuronas de las capas ocultas, donde ocurre el procesamiento más complejo. Finalmente, la capa de salida contiene las unidades que producen el resultado o la predicción final. Las conexiones entre unidades tienen diferentes fuerzas o "pesos" que regulan la importancia de cada conexión en el proceso.

Los datos de entrada se presentan en la capa de entrada y se transmiten de una neurona a otra a través de las capas ocultas, hasta llegar a la capa de salida. La red aprende examinando

registros individuales, generando predicciones para cada uno y ajustando los pesos cuando las predicciones son incorrectas. Este proceso se repite múltiples veces, y la red continúa mejorando sus predicciones hasta que se cumplen ciertos criterios de parada. La información de comparación se retroalimenta a través de la red, ajustando gradualmente los pesos.

A medida que avanza el entrenamiento, la red se vuelve cada vez más precisa en la replicación de resultados conocidos. Una vez entrenada, la red se puede utilizar en situaciones donde se desconocen los resultados futuros [16].

2.6.2 Selección de datos y características relevantes para el entrenamiento

El primer paso es cargar los datos en un formato adecuado para su uso en Scikit-learn. Los conjuntos de datos estaban disponibles en varios formatos, como CSV, Excel o JSON. En este caso, se utilizó la función read_csv de Pandas para cargar los datos desde un archivo CSV. Tras la carga de los datos, fue importante realizar un análisis exploratorio para comprender mejor sus características y detectar posibles problemas, como datos faltantes o valores atípicos. Para ello, se emplearon diversas herramientas de visualización y estadística descriptiva. El último paso en la preparación de los datos consistió en preprocesarlos para que pudieran ser utilizados en el entrenamiento del modelo, lo cual incluyó la selección de características apropiadas, estandarización de los datos y eliminación de valores faltantes [17].

2.6.3 Metodología de evaluación del rendimiento del modelo

En el mundo de la inteligencia artificial y el aprendizaje automático, la matriz de confusión es una herramienta que permite visualizar el desempeño de un algoritmo de aprendizaje supervisado. Cada columna de la matriz representa el número de predicciones para cada categoría, mientras que cada fila representa eventos en la categoría real [18].

2.6.4 GNS3

GNS3 (Graphic Network Simulation o Simulación Gráfica de Redes) es un simulador gráfico de red que permite diseñar topologías de red complejas y poner en marcha simulaciones sobre ellos. Con GNS3 los usuarios tienen la posibilidad de poder escoger cada uno de los elementos que llegarán a formar parte de una red informática [19].

2.6.5 Apache Spark

Apache Spark es un motor de procesamiento de datos de código abierto ultrarrápido para aplicaciones de aprendizaje automático e inteligencia artificial, maneja fácilmente conjuntos de datos a gran escala y es un sistema de agrupación en clústeres rápido y de uso general muy adecuado para PySpark. Está diseñado para ofrecer la velocidad computacional [20].

6.5.6 Proceso KDD

El Descubrimiento de conocimiento en bases de datos (kdd, del inglés Knowledge Discovery in Databases) es básicamente un proceso automático en el que se combinan descubrimiento y análisis. El proceso consiste en extraer patrones en forma de reglas o funciones, a partir de los datos, para que el usuario los analice. Esta tarea implica generalmente preprocesar los datos, hacer minería de datos (data mining) y presentar resultados. El proceso KDD que se indica en la Figura 5 es interactivo e iterativo, involucra pasos con la intervención del usuario en la toma de muchas decisiones [21].

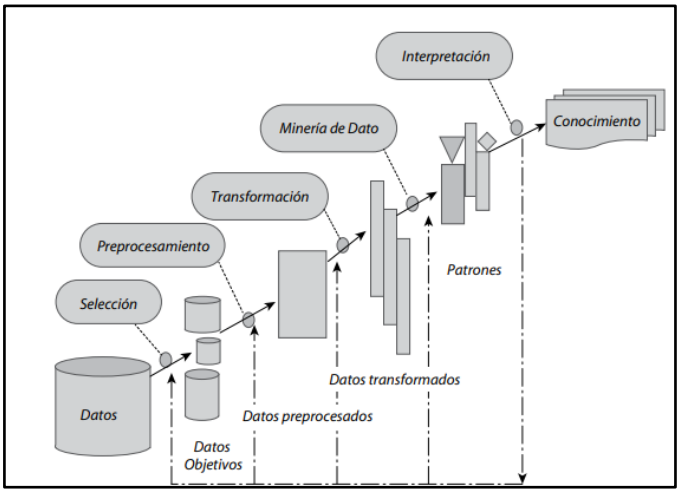


Figura 5. Etapas del proceso KDD
Fuente: [21]

6.5.7 Colab

Google Colab, o Colaboratory, es una herramienta gratuita basada en la nube que permite a los usuarios escribir, ejecutar y compartir código Python.
Utiliza notebooks, documentos interactivos que pueden contener texto, código, imágenes y otras opciones, permitiendo facilitar el proceso de desarrollo y documentación de proyectos. Además, como está alojado en la nube, Colab elimina la necesidad de instalar software adicional en sus computadoras proporcionando acceso a recursos de hardware como GPU y TPU sin coste adicional. Esta plataforma, por tanto, resulta especialmente útil para tareas de aprendizaje automático, análisis de datos y cualquier otro proyecto que requiera de un procesamiento intensivo de datos [22].

6.5.8 Docker

Es una plataforma de código abierto que permite a los desarrolladores crear, implementar, ejecutar, actualizar y gestionar aplicaciones en contenedores.
Los contenedores son componentes estandarizados y ejecutables que combinan el código fuente de la aplicación con las bibliotecas del sistema operativo (SO) y las dependencias necesarias para ejecutar ese código en cualquier entorno [23].

CAPÍTULO III. METODOLOGÍA

3.1 Metodología de investigación

Este proyecto de investigación adoptó un enfoque experimental para desarrollar e implementar un modelo de redes neuronales en un entorno simulado en GNS3, utilizando el proceso KDD (Knowledge Discovery in Databases). El objetivo fue detectar y mitigar de manera efectiva los ataques de amplificación de DNS. Al aplicar técnicas de aprendizaje automático y minería de datos sobre conjuntos de datos de tráfico de red, se identificaron patrones anómalos que caracterizaban dichos ataques.

3.2 Tipo de investigación

El tipo de investigación es cuantitativa puesto que mediante el uso de la herramienta Apache Spark se medirá parámetros de efectividad del modelo de redes neuronales en un ambiente simulado, mediante la manipulación de variables, como la configuración del modelo de redes neuronales y los parámetros de entrenamiento, se buscó establecer una relación causal entre el modelo y su capacidad para detectar y mitigar los ataques. Los experimentos se llevaron a cabo en un entorno controlado, utilizando tanto datos reales de la red de la Universidad Nacional de Chimborazo como datos simulados generados en GNS3.

3.3 Diseño de la investigación

El diseño de la investigación siguió un proceso iterativo basado en la metodología KDD. Se inició con la selección y el preprocesamiento de los datos, seguido de la aplicación de técnicas de minería de datos para descubrir patrones relevantes. Posteriormente, se diseñó y entrenó un modelo de redes neuronales, que fue evaluado en un conjunto de datos de prueba. Finalmente, se interpretaron los resultados y se extrajeron conclusiones sobre la efectividad del modelo propuesto.

3.3.1 Procedimiento de la investigación

- Recolección del tráfico real del servidor DNS de la UNACH.
- Implementación del escenario de red simulado utilizando GNS3.
- Recolección del tráfico normal y de ataque al servidor DNS sobre el escenario simulado.
- Investigación de un modelo de redes neuronales para la detección y mitigación de patrones asociados con ataques de amplificación de DNS.
- Ampliación del modelo de redes neuronales definido sobre el escenario simulado.
- Validación de la efectividad del modelo de redes neuronales para la detección y mitigación de ataques de amplificación de DNS con base a la herramienta Apache Spark.
    - Generación de matriz de confusión para el modelo de redes neuronales investigado y definido.
    - Generación de matriz de confusión para el modelo SVM.
    - Análisis comparativo de la efectividad de ambos modelos.

3.4 Población y muestra

Población

La población de este estudio estuvo compuesta por los datos del tráfico de red de la Universidad Nacional de Chimborazo y de un entorno simulado creado en GNS3.

Muestra

La muestra es un subconjunto de datos de la población elegidos de manera aleatoria para incluir tanto tráfico normal como tráfico malicioso asociado a ataques de amplificación de DNS.

3.5 Técnicas de recolección de datos

Para la recolección de datos, se empleó diversas técnicas, incluyendo:

- Monitoreo en tiempo real de la red: Se utilizó herramientas de monitoreo de red como Wireshark para registrar el tráfico entrante y saliente, capturando así datos relevantes sobre los ataques de amplificación de DNS y el tráfico normal.
- Herramientas de simulación: Se utilizó herramientas de simulación como GNS3 y máquinas virtuales para generar tráfico controlado y evaluar el rendimiento del modelo de redes neuronales en condiciones controladas.
- Herramienta de evaluación: Se utilizó Apache Spark para evaluar la efectividad del modelo, aplicando las métricas de evaluación en el aprendizaje automático, como precisión, F1-Score, exhaustividad, especificidad y tasa de falsos positivos.

3.6 Identificación de variables

3.6.1 Variable dependiente

Efectividad del modelo de redes neuronales para la detección y mitigación de ataques de DNS

3.6.2 Variable independiente

Modelo de redes neuronales

3.7 Operacionalización de variables

En la Tabla 1 se puede visualizar la operacionalización de variable.

Tabla 1. Operacionalización de variables

PROBLEMA	TEMA	OBJETIVOS	VARIABLES	CONCEPTUALIZACIÓN	DIMENSIÓN	INDICADORES
¿Cómo la aplicación del modelo de redes neuronales incide en la efectividad para detectar y mitiga los ataques de amplificación de DNS?	Modelo de redes neuronales para la detección y mitigación de ataques de amplificación de DNS en la UNACH.	GENERAL Desarrollar un modelo de redes neuronales para la detección y mitigación de ataques de amplificación de DNS.	INDEPENDIENTE Modelo de redes neuronales.	Un modelo de redes neuronales es una estructura computacional inspirada en el funcionamiento del cerebro humano que se utiliza para aprender a partir de datos y realizar tareas específicas, como clasificación, regresión, reconocimiento de patrones, entre otros.	Redes neuronales	Independiente: • Tamaño de red neuronal • Tiempo de desarrollo • Tipo de datos
		ESPECIFICOS • Investigar un modelo de redes neuronales para la detección y mitigación de patrones asociados con ataques de amplificación de DNS. • Implementar el modelo en un entorno simulado. • Validar la efectividad del modelo de redes neuronales para la detección y mitigación de ataques de amplificación de DNS con base a la herramienta Apache Spark.	DEPENDIENTE Validar la Efectividad del modelo de redes neuronales para la detección y mitigación de ataques de DNS en la Universidad Nacional de Chimborazo.	Efectividad se refiere a evaluar en qué medida cumple con los requisitos establecidos.	Calidad de los modelos redes neuronales	Dependiente Efectividad: ✓ Precisión (Accuracy) ✓ Precisión (Precision) ✓ F1-Score ✓ Exhaustividad ✓ Especificidad ✓ Tasa de falsos positivos

3.8 Recolección de datos y diseño de topología de red en GNS3

Para la recolección de datos en la Universidad Nacional de Chimborazo se utilizó la herramienta Wireshark y se empleó una máquina virtual con Ubuntu para realizar el ataque al servidor DNS. Esta misma metodología se aplicó en la topología creada en GNS3 con máquinas virtuales.

Esquema de recolección de tráfico real del DNS en la UNACH

Se llevó a cabo la captura de tráfico de red utilizando la herramienta Wireshark. Esta actividad se realizó en la Universidad Nacional de Chimborazo, donde se dispone de una infraestructura compuesta por tres servidores DNS como se muestra en la Figura 6. Uno de estos servidores funciona como principal, encargado de gestionar y resolver las solicitudes DNS, mientras que los otros dos servidores operan como réplicas, sincronizando y replicando la información procesada por el servidor principal. Este diseño asegura redundancia y disponibilidad del servicio DNS. La herramienta Wireshark permitió capturar y analizar el tráfico de red entre los clientes y los servidores DNS, obteniendo información clave sobre los protocolos, la longitud y los detalles de los paquetes para el estudio del tráfico y la detección de patrones asociados a posibles ataques de amplificación.

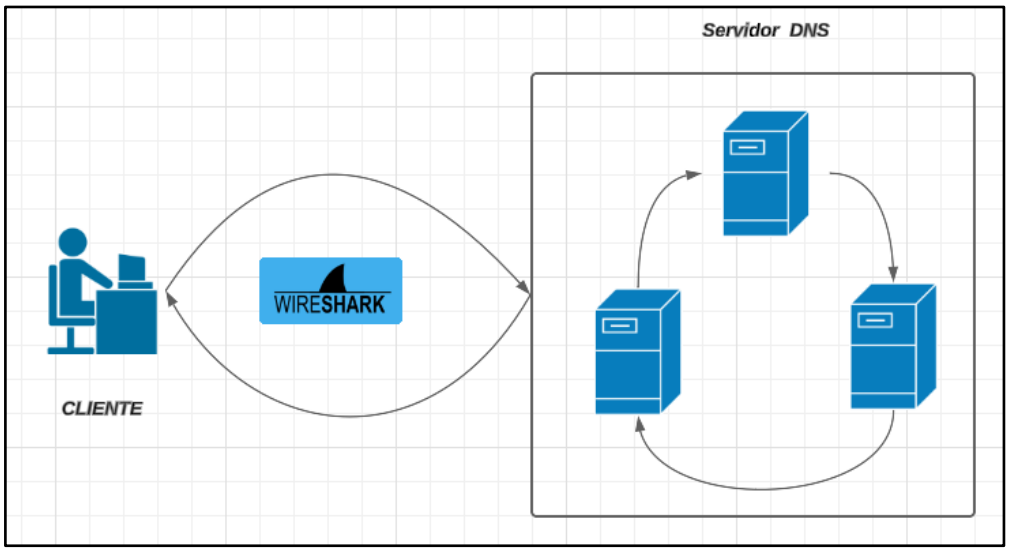


Figura 6. Esquema de captura de tráfico DNS en la UNACH

Fase de recolección de datos

En esta etapa, se procedió a capturar el tráfico de red del DNS en la Universidad Nacional de Chimborazo utilizando la herramienta de Wireshark, tanto en momentos de ataques de amplificación de DNS como en periodos de tráfico normal y se procedió a guardar en archivos (.csv). Como se indica en las Figuras 7, 8 y 9.

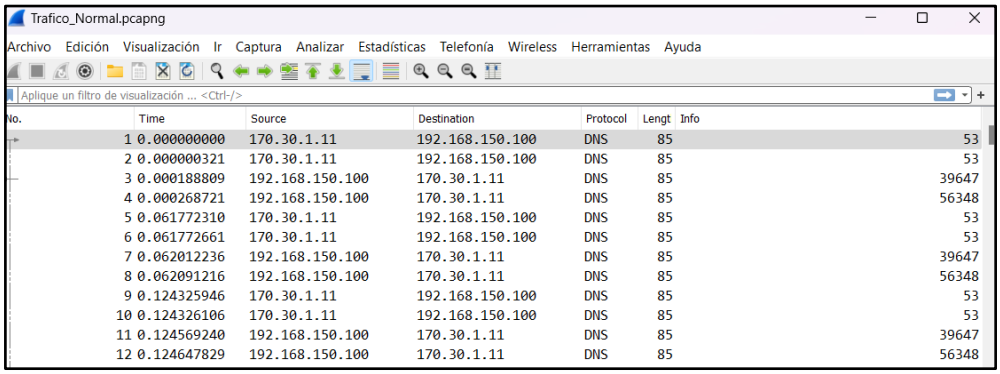


Figura 7. Captura de tráfico de red con la herramienta Wireshark

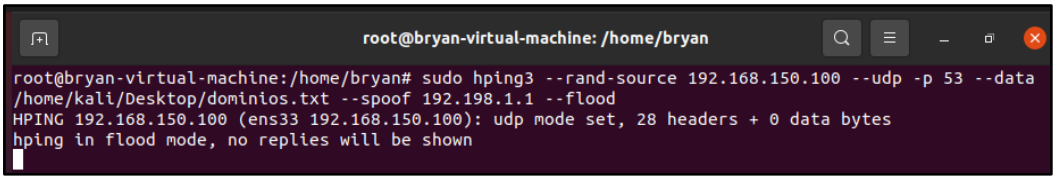


Figura 8. Comando estructurado para realizar un ataque al servidor DNS

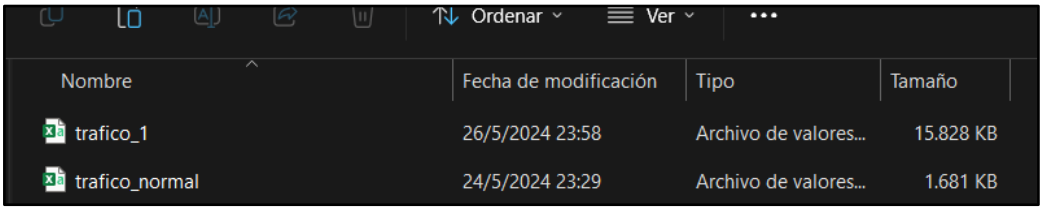


Figura 9. Archivos guardados de tráfico de red capturado (.csv)

3.9 Implementación del escenario de red simulado utilizando GNS3

Se diseñó una topología de red en GNS3, un simulador de redes que permitió crear entornos virtuales. En este entorno, se configuraron varias máquinas virtuales, todas basadas en el sistema operativo Ubuntu como se indica en la Figura 10.

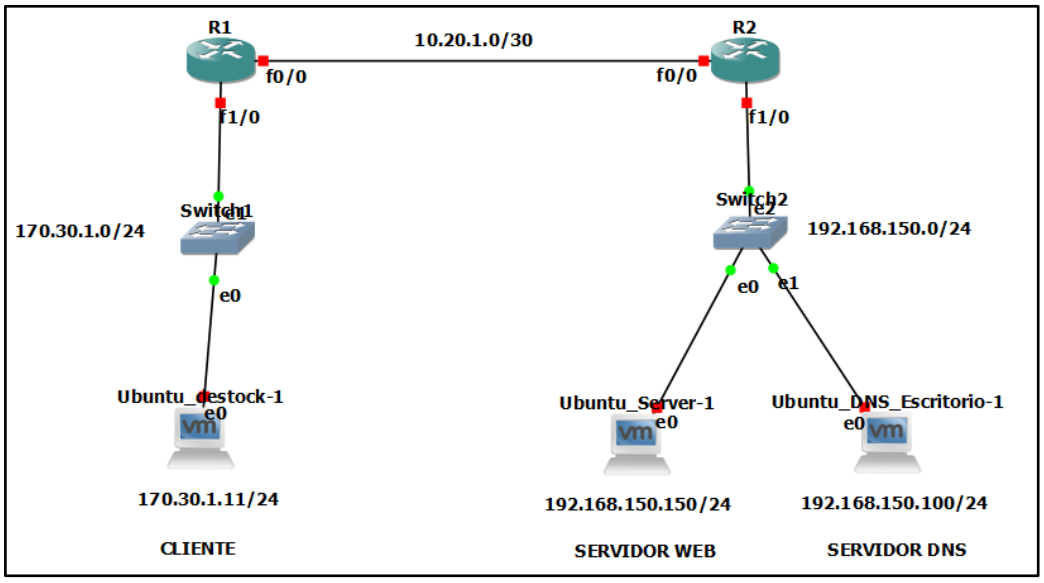


Figura 10. Escenario implementado en GNS3 con máquinas virtuales Ubuntu

Se implementaron dos instancias de Ubuntu Desktop 20.04.6. La primera máquina Ubuntu se configuró como cliente, en la cual solo se instaló el software necesario para realizar solicitudes de resolución de DNS. La segunda maquina Ubuntu se configuró como servidor DNS, en la que se instaló y configuró BIND9 para proporcionar servicios de resolución de nombres de dominio.

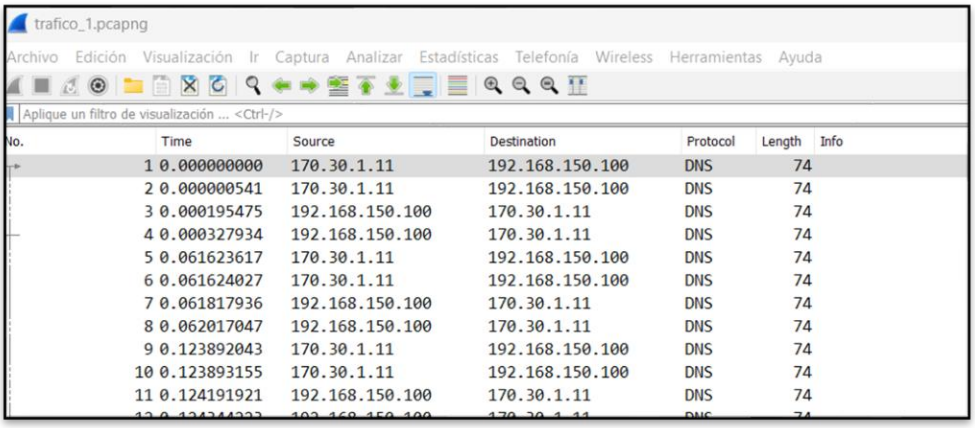
Además, se utilizó una máquina virtual con Ubuntu 22.04.4-live-server, que desempeñó el rol de servidor web. En esta máquina se instaló y configuró Apache2 para proporcionar servicios web, permitiendo que el cliente accediera a recursos web utilizando nombres de dominio resueltos a través del servidor DNS. Esta configuración permitió simular interacciones típicas de una red, donde el cliente realizó solicitudes al servidor DNS para resolver nombres de dominio y, a su vez, acceder a recursos en el servidor web.

3.10 Recolección del tráfico normal y de ataque al servidor DNS sobre el escenario simulado

Para generar y analizar tanto el tráfico normal como el de ataque, se utilizó la herramienta Wireshark 4.03. Esta herramienta permitió capturar y registrar paquetes de datos en tiempo real, facilitando el estudio del comportamiento del tráfico de red en condiciones normales y durante ataques de amplificación de DNS.

El ataque generado fue un ataque de tipo NXDOMAIN, que se caracteriza por enviar solicitudes DNS a servidores con nombres de dominio inexistentes, obligando al servidor a responder con errores y consumiendo recursos. Este ataque fue simulado utilizando el siguiente comando en la herramienta hping3 (sudo hping3 --rand-source 192.168.150.100 --udp -p 53 --spoof 8.8.8.8 –flood).

La capacidad de monitoreo de Wireshark 4.03 fue esencial para analizar este tráfico y evaluar la eficacia de las medidas de seguridad implementadas. Además, permitió identificar patrones en el tráfico de red asociados con actividades maliciosas, cómo se indica en las Figura 11.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	170.30.1.11	192.168.150.100	DNS	74	
2	0.000000541	170.30.1.11	192.168.150.100	DNS	74	
3	0.000195475	192.168.150.100	170.30.1.11	DNS	74	
4	0.000327934	192.168.150.100	170.30.1.11	DNS	74	
5	0.061623617	170.30.1.11	192.168.150.100	DNS	74	
6	0.061624027	170.30.1.11	192.168.150.100	DNS	74	
7	0.061817936	192.168.150.100	170.30.1.11	DNS	74	
8	0.062017047	192.168.150.100	170.30.1.11	DNS	74	
9	0.123892043	170.30.1.11	192.168.150.100	DNS	74	
10	0.123893155	170.30.1.11	192.168.150.100	DNS	74	
11	0.124191921	192.168.150.100	170.30.1.11	DNS	74	

Figura 11. Captura de tráfico de red con la herramienta Wireshark

Fase de identificación de variables del tráfico de red

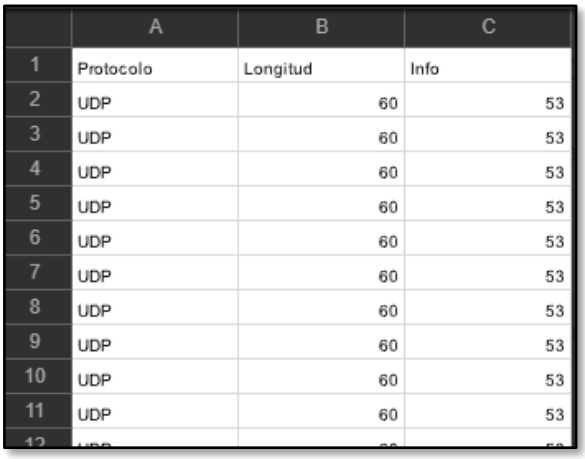
Se identificaron los siguientes indicadores para detectar ataques de amplificación de DNS, los cuales presentan características distintivas que los diferencian del tráfico normal.

- En la columna protocolo, la presencia de la etiqueta UDP sugiere que los atacantes están enviando tráfico directo y en un tráfico normal existe la etiqueta DNS.

- La longitud de los paquetes durante los ataques fue de 60 bytes, mientras que en el tráfico normal superó los 80 bytes. Esta diferencia indica el uso de paquetes UDP, que son más pequeños y específicos, diseñados para explotar vulnerabilidades en los servidores DNS

- Además, el valor "53" en la columna "Info" durante los ataques correspondió al puerto UDP 53, que se utiliza por defecto para las comunicaciones DNS, lo cual ayuda a identificar paquetes relacionados con el ataque.

Estos indicadores resultaron fundamentales para la detección y mitigación de ataques de amplificación de DNS como se muestra en la Figura 12.



	A	B	C
1	Protocolo	Longitud	Info
2	UDP	60	53
3	UDP	60	53
4	UDP	60	53
5	UDP	60	53
6	UDP	60	53
7	UDP	60	53
8	UDP	60	53
9	UDP	60	53
10	UDP	60	53
11	UDP	60	53
12			

Figura 12. Identificación de variables de tráfico DNS

Desarrollo de un modelo de redes neuronales

Para el desarrollo del modelo de la red neuronal perceptrón se utilizó Colab para ejecutar el proceso KDD que consta de las siguientes fases.

Fase 1: Comprensión del dominio

Se cargaron y prepararon los datos de tráfico de red capturados tanto en escenarios de tráfico normal como durante ataques de amplificación de DNS. Los datos fueron obtenidos de archivos .CSV, uno correspondiente al tráfico normal y otro al tráfico malicioso, con el objetivo de realizar un análisis exhaustivo en el modelo.

```
# Cargar datos
data_ataque = pd.read_csv("/content/drive/MyDrive/tesis_pruebas/trafico_ataque_1.csv", sep=';')
data_normal = pd.read_csv("/content/drive/MyDrive/tesis_pruebas/trafico_normal.csv", sep=';')
```

Fase 2: Selección de datos

Para identificar el tráfico de ataques y tráfico normal, se añadió una columna de etiqueta a cada conjunto de datos. Al tráfico identificado como ataque se le asignó una etiqueta de 1, mientras que al tráfico normal se le asignó una etiqueta de 0. Posteriormente, ambos conjuntos de datos fueron combinados en un único DataFrame, facilitando el procesamiento y análisis posterior en el modelo.

```
# Añadir columna de etiqueta
data_ataque['Etiqueta'] = 1
data_normal['Etiqueta'] = 0
# Combinar DataFrames
data = pd.concat([data_ataque, data_normal])
```

Fase 3: Procesamiento de datos

Para preparar los datos para el entrenamiento del modelo, se aplicó una técnica de preprocesamiento de datos conocida como codificación de etiquetas (Label Encoding) para convertir los valores categóricos de la columna 'Protocolo' en valores numéricos. Posteriormente, se realizó una verificación para identificar y eliminar valores nulos o infinitos dentro del conjunto de datos, utilizando la función de reemplazo y la eliminación de valores faltantes, con el fin de evitar errores en las fases de procesamiento y entrenamiento posteriores.

```
# Convertir la columna 'Protocolo' a valores numéricos usando LabelEncoder
le = LabelEncoder()
data['Protocolo'] = le.fit_transform(data['Protocolo'])
# Comprobar y eliminar valores nulos o infinitos
data.replace([np.inf, -np.inf], np.nan, inplace=True)
data.dropna(inplace=True)
```

Fase 4: Reducción de dimensionalidad

En esta fase, las características del conjunto de datos fueron normalizadas utilizando MinMaxScaler para garantizar que todos los valores estuvieran en la misma escala, facilitando el entrenamiento del modelo de redes neuronales.

```
# Normalizar características
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
```

Fase 5: Modelado

Se construyó el modelo de red neuronal Perceptrón utilizando la librería Sequential de Keras. La arquitectura del modelo incluyó tres capas densas (completamente conectadas), empleando la función de activación relu en las dos primeras capas y sigmoid en la capa final para realizar una clasificación binaria.

```
# Construir el modelo
model = Sequential()
model.add(Dense(64, input_dim=X_train.shape[1], activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

Fase 6: Entrenamiento

El modelo se compiló con el optimizador Adam y una tasa de aprendizaje de 0.0001, utilizando la función de pérdida binary_crossentropy, adecuada para realizar una clasificación binaria. Luego, se entrenó durante 50 épocas, es decir, ciclos completos en los que el modelo recorrió todo el conjunto de datos de entrenamiento. Se empleó un tamaño de lote de 32 muestras y un conjunto de validación equivalente al 20% del total de datos de entrenamiento para evaluar el rendimiento del modelo durante el proceso.

```
# Compilar el modelo
model.compile(optimizer=Adam(learning_rate=0.0001), loss='binary_crossentropy',
metrics=['accuracy'])
# Entrenar el modelo
history = model.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.2)
```

Fase 7: Evaluación

Una vez finalizado el entrenamiento, el modelo fue evaluado en el conjunto de datos de prueba para calcular la pérdida y la precisión. Estos valores permitieron evaluar el rendimiento del modelo para la tarea de clasificación entre tráfico normal y ataques de amplificación de DNS.

```
# Evaluar el modelo
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Loss: {loss}, Accuracy: {accuracy}')
```

Fase 8. Interpretación

Finalmente, se realizaron predicciones sobre el conjunto de datos de prueba y se generó un reporte de clasificación para analizar las métricas de precisión, recall y F1-Score, permitiendo identificar posibles áreas de mejora en el rendimiento del modelo. Además, el modelo se guardó en un archivo con extensión .h5 para facilitar su uso en futuras simulaciones y en la detección de tráfico malicioso en tiempo real.

```
# Predicciones y reporte de clasificación
y_pred = (model.predict(X_test) > 0.5).astype(int)
print(classification_report(y_test, y_pred))
# Guardar el modelo entrenado en un archivo .h5
model.save('model.h5') from google.colab import files
# Descargar el archivo model.h5 a tu máquina local
files.download('model.h5')
```

3.11 Aplicación del modelo de redes neuronales definido sobre el escenario simulado

Para analizar y entrenar la red neuronal, se implementó el modelo de red neuronal entrenada en un entorno simulado de GNS3 con máquinas virtuales. Como se indica en la Figura 13.

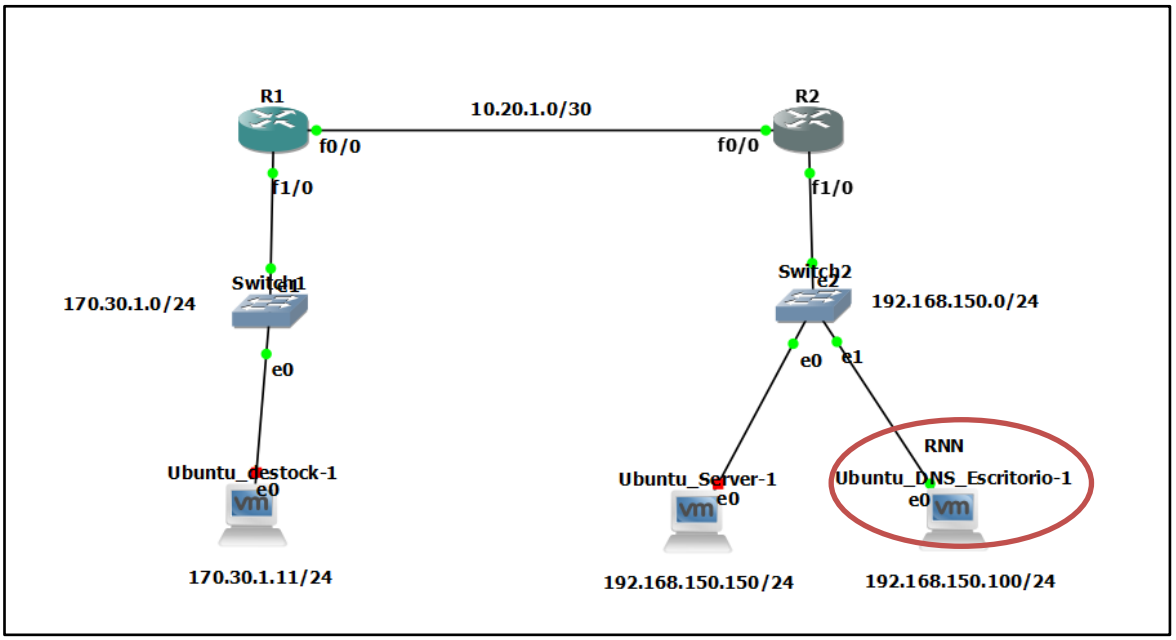


Figura 13. Implementación de la red neuronal en el servidor DNS sobre el entorno simulado de GNS3

Para la implementación del modelo de redes neuronales, se configuró el entorno de una máquina virtual del servidor DNS.

Paso 1: Para la configuración de TensorFlow Serving en una máquina virtual. Se instaló Docker, la imagen de tensorflow serving para el modelo en formato SavedModel y se inicializó TensorFlow Serving como se indica en la Figura 14 y 15.

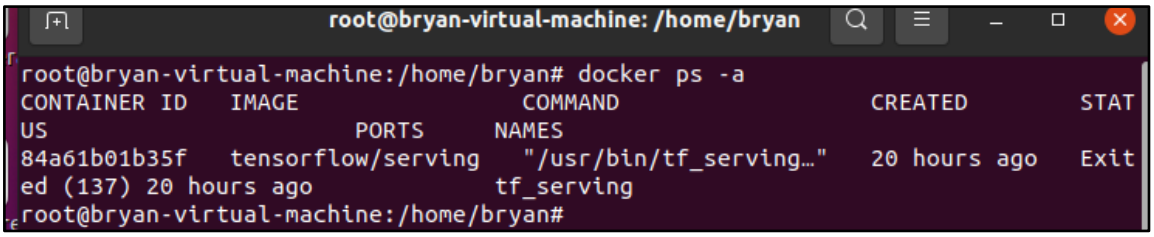


Figura 14. Inicialización de Docker

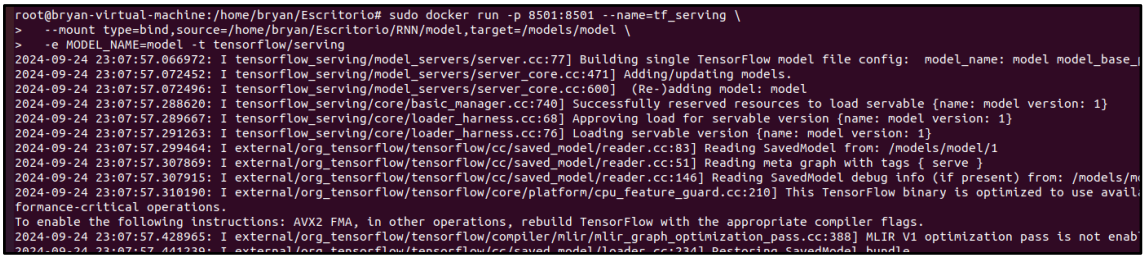


Figura 15. Inicialización de TensorFlow Serving

Paso 2. Para la creación de un script que permitiera la captura de tráfico de red en tiempo real, se utilizó Python para desarrollar las instrucciones necesarias para monitorear y registrar el tráfico. Este proceso de captura fue implementado según lo ilustrado en la Figura 16.

Figura 16. Configuración del script para capturar el tráfico en tiempo real

Paso 3. Se inicializo el script de captura de tráfico en tiempo real y, a continuación, se inició el ataque desde la máquina cliente Ubuntu. Tras verificar que el proceso se completó correctamente, se confirmó que el sistema estaba registrando el tráfico de red en tiempo real y detectando las anomalías asociadas con el ataque, como se ilustra en la Figura 17 y 18.

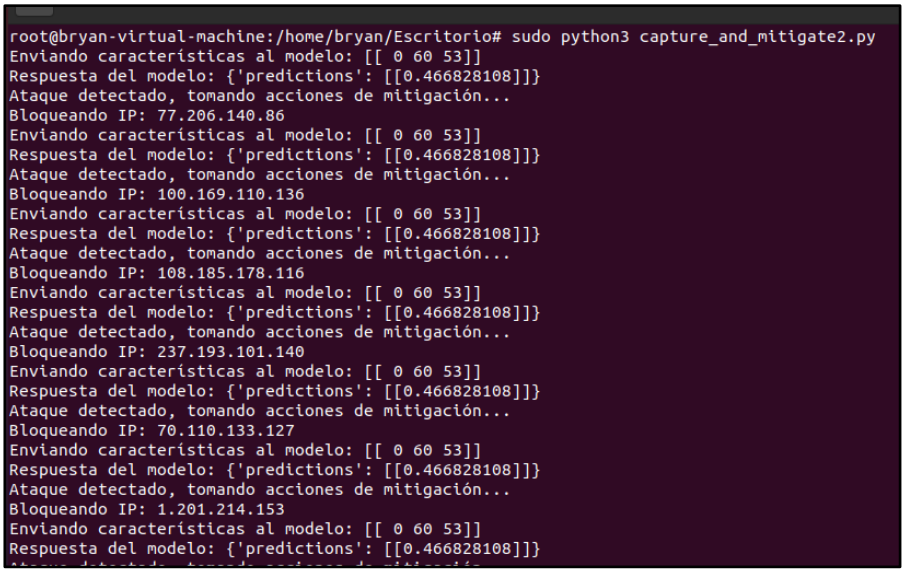


Figura 17. Captura del tráfico en tiempo real con el Script

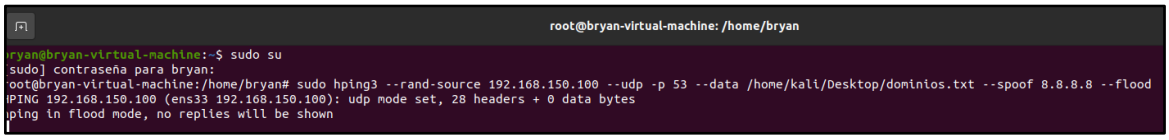


Figura 18. Comando para realizar un ataque de amplificación de DNS

3.12 Validación de la efectividad del modelo

La validación de la efectividad del modelo en términos de Precisión, F1-Score, Exhaustividad, Especificidad y Tasa de falsos positivos es crucial para analizar el comportamiento de la red neuronal a través del tiempo y verificar la detección y mitigación de los ataques de amplificación de DNS.

Fase 1. Se capturó el tráfico normal y el tráfico de ataque de red una vez implementado el modelo de red neuronal en el escenario de GNS3, este tráfico se capturó con la herramienta Wireshark y se procedió a guardar en archivos (.csv) como se muestra en la Figura 19 y 20.

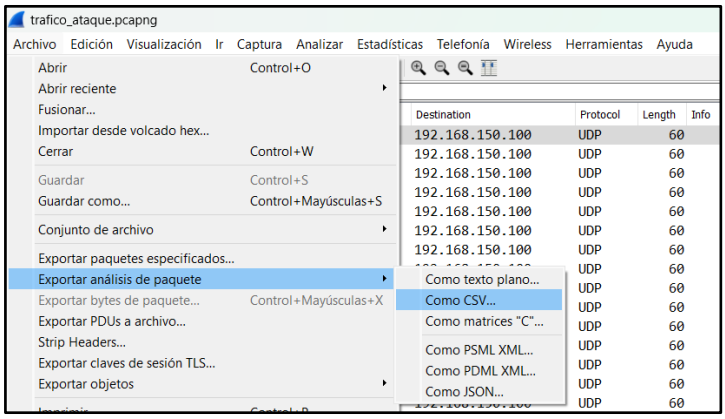


Figura 19. Captura de tráfico de ataque con la red neuronal implementada

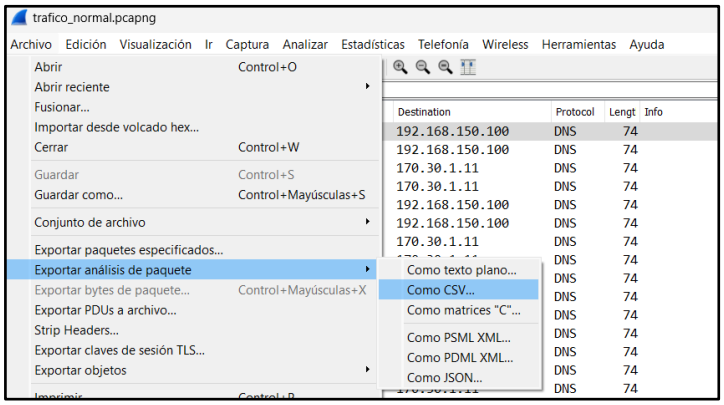


Figura 20. Captura de tráfico normal con la red neuronal implementada

Fase 2. Se utilizaron los archivos (.csv) capturados del tráfico normal y tráfico de ataque y a través de la red neuronal configurada en Colab se ejecuta y analiza el modelo como se muestra en la Figura 21.

```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)

# Cargar datos
data_ataque = pd.read_csv("/content/drive/MyDrive/tesis_preuba2/trafico_ataque_1.csv", sep=';')
data_normal = pd.read_csv("/content/drive/MyDrive/tesis_preuba2/trafico_normal.csv", sep=';')

# Añadir columna de etiqueta
data_ataque['Etiqueta'] = 1
data_normal['Etiqueta'] = 0

# Combinar DataFrames
data = pd.concat([data_ataque, data_normal])

# Convertir la columna 'Protocolo' a valores numéricos usando LabelEncoder
le = LabelEncoder()
data['Protocolo'] = le.fit_transform(data['Protocolo'])

# Comprobar y eliminar valores nulos o infinitos
data.replace([np.inf, -np.inf], np.nan, inplace=True)
data.dropna(inplace=True)

# Separar características y etiquetas
X = data.drop(columns=['Etiqueta'])
y = data['Etiqueta']

# Normalizar características
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)

# Dividir los datos en conjuntos de entrenamiento y de prueba
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42, stratify=y)

# Construir el modelo
model = Sequential()
model.add(Dense(64, input_dim=X_train.shape[1], activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

Figura 21. Configuración de la red neuronal en Colab con el modelo de redes neuronales.

Fase 3. Se generaron gráficas de la pérdida y la precisión de los conjuntos de entrenamiento y validación a lo largo de las épocas, así como la matriz de confusión, para evaluar el comportamiento del modelo de red neuronal durante el entrenamiento. Estas representaciones mostraron cómo se comportó el modelo frente al tráfico de ataque y al tráfico normal.

Gráfico de pérdida a lo largo de las épocas

Al inicio del entrenamiento, la pérdida en los conjuntos de entrenamiento y validación fue alta, dado que el modelo estaba ajustando sus parámetros. Durante las primeras épocas, la pérdida disminuyó rápidamente, lo que indicaba que el modelo estaba aprendiendo de manera eficiente. Con el tiempo, la reducción de la pérdida se volvió más lenta y ambas curvas comenzaron a estabilizarse en valores bajos, lo que señaló que el modelo había aprendido los patrones relevantes de los datos.

La cercanía entre las curvas de pérdida de entrenamiento y validación mostró que no hubo sobreajuste, lo que indicó una buena capacidad de generalización del modelo. Al final del entrenamiento, las dos curvas se estabilizaron alrededor de 0.01, reflejando un buen desempeño en la identificación de tráfico normal y de ataque, como se muestra en la Figura 22.

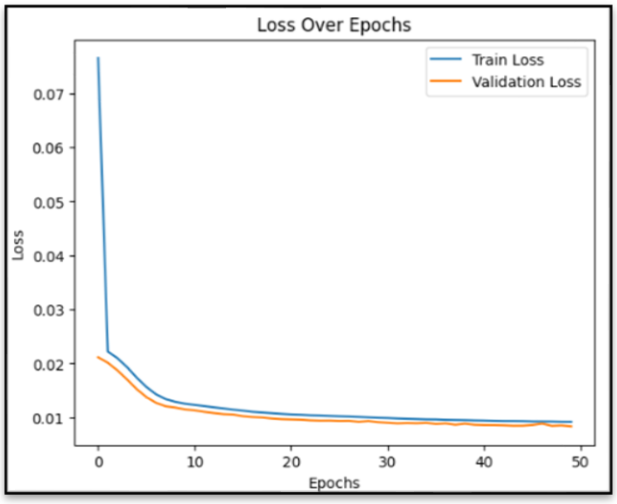


Figura 22. Perdida en el conjunto de entrenamiento a lo largo de las épocas

Gráfico de precisión a lo largo de las épocas

El gráfico mostró la evolución de la precisión del modelo en los conjuntos de entrenamiento y validación durante 50 épocas. Al inicio, la precisión fue baja, debido que el modelo estaba ajustando sus parámetros y aún no había aprendido a predecir correctamente. Sin embargo, en las primeras épocas, la precisión aumentó rápidamente, indicando un aprendizaje efectivo y una mejora significativa en sus predicciones.

Con el tiempo, la mejora en la precisión se volvió más gradual y las curvas de entrenamiento y validación se estabilizaron en valores altos. Ambas curvas se mantuvieron muy cercanas, demostrando que no había sobreajuste y que el modelo tenía una buena capacidad de generalización.

Al final del entrenamiento, las dos curvas alcanzaron una precisión muy alta, cercana al 99.8%, lo que reflejaba un excelente rendimiento en la clasificación de tráfico normal y de ataque, con una mínima tasa de error. Como se muestra en la Figura 23.

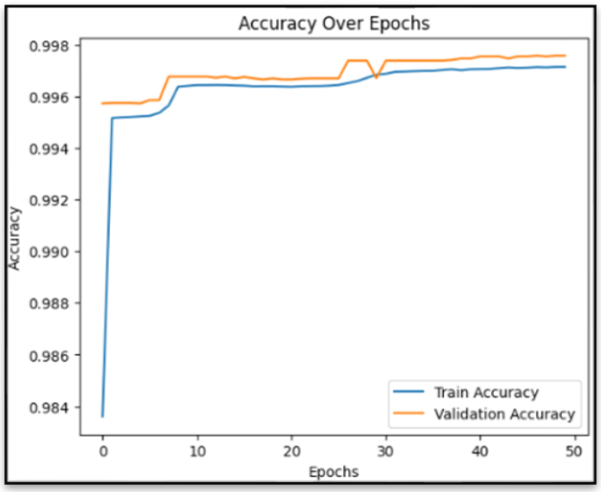


Figura 23. Precisión en el conjunto de entrenamiento a lo largo de las épocas

Generación de matriz de confusión para el modelo SVM

Se configuró el entorno en Colab para entrenar y evaluar el modelo de Máquina de Vectores de Soporte (SVM). Tras entrenar el modelo con los datos de entrenamiento, se realizaron predicciones sobre los datos de prueba. Para evaluar su desempeño, se generó la matriz de confusión, que permitió visualizar la cantidad de predicciones correctas e incorrectas del modelo, clasificadas por las etiquetas normal y ataque, mostrados en la Figura 24.

```
# Entrenar y evaluar el modelo SVM
svm_model = SVC()
svm_model.fit(X_train, y_train)
y_pred_svm = svm_model.predict(X_test)

# Reporte de clasificación para SVM
print("SVM - Clasificación:")
print(classification_report(y_test, y_pred_svm))

# Matriz de confusión para SVM
conf_matrix_svm = confusion_matrix(y_test, y_pred_svm)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix_svm, annot=True, fmt="d", cmap="Blues", xticklabels=['Normal', 'Ataque'], yticklabels=['Normal', 'Ataque'])
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Matriz de Confusión - SVM')
plt.show()
```

Figura 24. Configuración para la matriz de confusión en Colab con el modelo SVM

CAPÍTULO IV. RESULTADOS Y DISCUSIÓN

Resultados del objetivo específico 1

El modelo Perceptrón Multicapa (MLP) fue seleccionado debido a su capacidad para manejar problemas de clasificación binaria, como la detección de ataques de amplificación de DNS, donde se requiere distinguir entre tráfico normal y tráfico malicioso. En la Figura 25 se indica el esquema funcional del modelo defino.

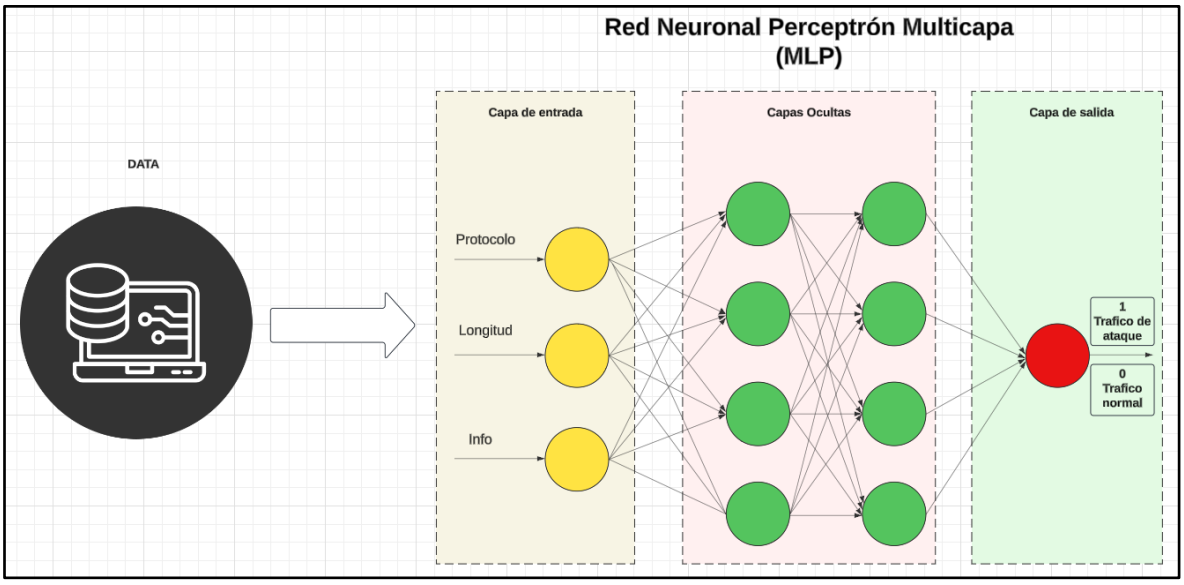


Figura 25. Modelo de red neuronal Perceptrón multicapa para la clasificación binaria

La red neuronal Perceptrón multicapa (MLP) sobresale en la identificación de relaciones no lineales en los datos, algo crucial en el análisis de tráfico de red. Su arquitectura permite procesar múltiples características simultáneamente, como el protocolo, la longitud y la información del paquete, que son determinantes para clasificar el tráfico.

Capa de entrada

Los datos de entrada correspondieron a tres variables principales extraídas del tráfico de red el protocolo, longitud e info del paquete. Estas características fueron seleccionadas porque demostraron ser determinantes para distinguir entre tráfico legítimo y tráfico malicioso. Cada una de estas variables fue representada como un nodo en la capa de entrada del modelo.

Capas ocultas

La red cuenta con dos capas ocultas, la primera con 32 neuronas y la segunda con 64 neuronas, ambas con la función de activación ReLU (Rectified Linear Unit). Estas capas permiten que el modelo aprenda patrones complejos ajustando los pesos y sesgos durante el entrenamiento mediante retropropagación y el optimizador Adam.

- **Primera capa oculta (32 neuronas).** Se encarga de identificar patrones iniciales en los datos, proporcionando una representación básica de las relaciones entre las variables.
- **Segunda capa oculta (64 neuronas).** Refina los patrones detectados en la primera capa, maximizando la capacidad del modelo para generalizar y distinguir entre tráfico normal y de ataque.

Capa de salida

Está formada por una sola neurona con la función de activación sigmoide, adecuada para tareas de clasificación binaria. El resultado final fue un valor entre 0 y 1 que indicó la probabilidad de que el tráfico analizado fuese malicioso, para clasificar el tráfico.

- **0** tráfico normal.
- **1** tráfico malicioso.

La capa de salida convierte los patrones detectados en una clasificación binaria que indica si el tráfico analizado es malicioso o no. Este resultado puede ser interpretado directamente para tomar decisiones, como bloquear el tráfico de ataque y permitir el tráfico legítimo.

Resultados del objetivo específico 2

Como resultado de la implementación en un entorno simulado utilizando GNS3 con máquinas virtuales, se configuró un escenario utilizando diferentes herramientas como se muestra en la Figura 26.

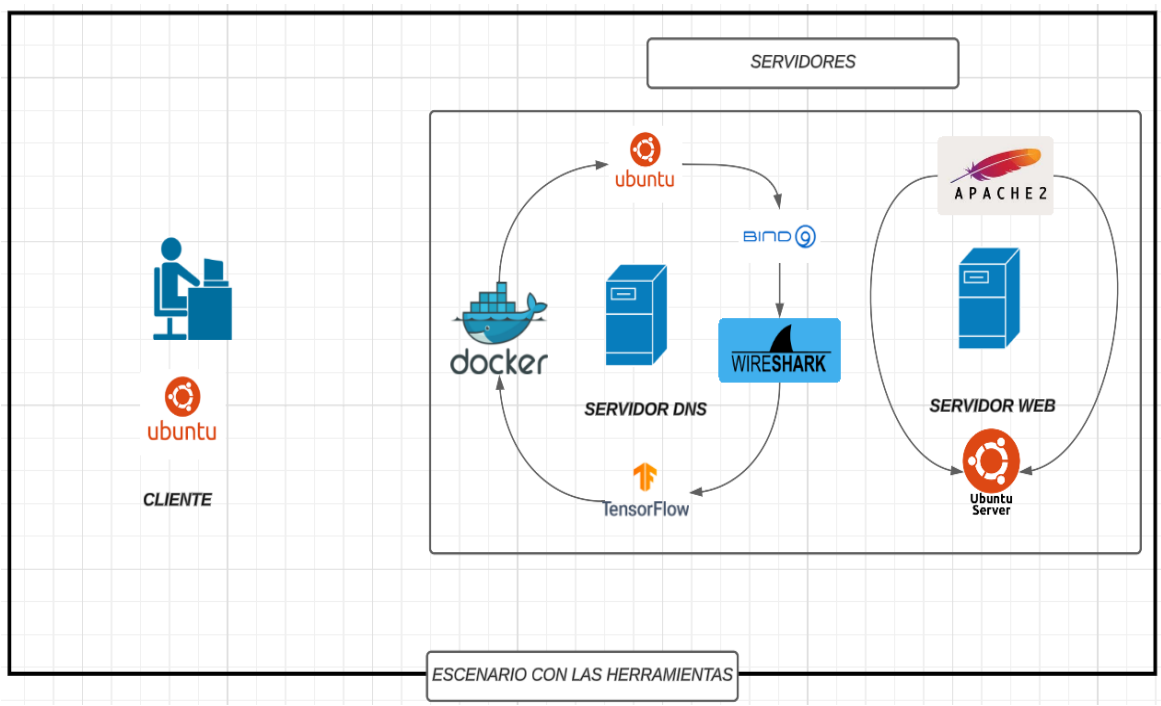


Figura 26. Escenario representativo con las herramientas utilizadas

Cliente. Se implementó un cliente basado en Ubuntu, configurado para realizar consultas al servidor web.

Servidor web. Se configuró un servidor web con el sistema operativo Ubuntu Server, utilizando Apache 2 como servidor HTTP. Este servidor permitió alojar un sitio web creado específicamente para las pruebas.

Servidor DNS. Se configuró un servidor DNS basado en Ubuntu Desktop, en el cual se instaló y configuró BIND9. En este servidor, se definieron las zona directa e inversa para resolver el dominio asociado al sitio web alojado en el servidor web.

TensorFlow y Docker. Se utilizó TensorFlow para desplegar el modelo de aprendizaje automático, permitiendo que este pudiera interactuar con un script en Python para procesar el tráfico de red. Docker se empleó para simplificar y optimizar los procesos de implementación del modelo, asegurando un entorno controlado y eficiente.

Captura de tráfico en tiempo real. Se desarrolló un script en Python, configurado para capturar el tráfico de red en tiempo real. Este script permitió que, al integrarse con el modelo de redes neuronales desplegado, el sistema pudiera detectar tráfico normal y tráfico de ataque. Además, el modelo mitigó el tráfico de ataque al bloquear las solicitudes maliciosas mientras permitía el paso del tráfico legítimo.

Resultados del objetivo del especifico 3

Evaluar la efectividad del modelo de redes neuronales para la detección y mitigación de ataques de amplificación de DNS con base a la herramienta Apache Spark. Como resultado de la implementación del modelo de redes neuronales sobre el entrono simulado con GNS3 y luego del entrenamiento del modelo utilizando trafico normal y trafico de ataque se genera la matriz de confusión para el análisis de los parámetros establecidos para evaluar la efectividad del modelo con redes neuronales investigado y definido.

Matriz de confusión con el modelo de redes neuronales

En la Figura 27 se muestra una matriz de confusión para evaluar la efectividad del modelo de redes neuronales utilizado en la detección y mitigación de ataques de amplificación de DNS. Esta matriz proporciona una representación visual de las predicciones realizadas por el modelo, comparándolas con las etiquetas reales de los datos. En este gráfico, se presentan cuatro valores clave que ayudan a entender cómo se desempeña el modelo en la clasificación de tráfico normal y tráfico de ataque, para evaluar su efectividad.

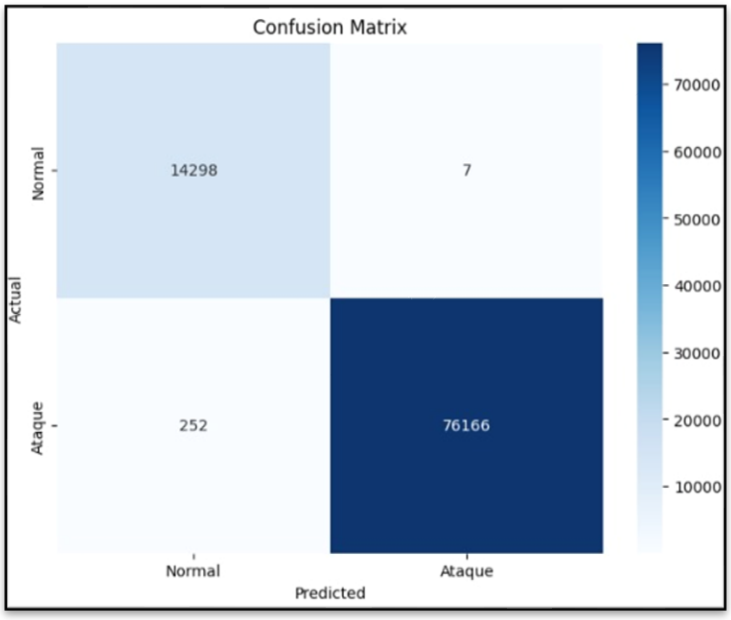


Figura 27. Matriz de Confusión de evaluación del modelo de redes neuronales

Componentes de la Matriz

En la Tabla 2 se muestra los datos proporcionados de la matriz de confusión.

Tabla 2. Interpretación de datos

Componente	Valor	Interpretación
Verdaderos Positivos (TP):	76166	Este número indica la cantidad de veces que el modelo predijo correctamente un ataque. Un alto valor de verdaderos positivos muestra que el modelo es muy eficaz en la detección de ataques de amplificación de DNS.
Verdaderos Negativos (TN):	14298	Esto representa la cantidad de veces que el modelo predijo correctamente que el tráfico era normal. La alta cantidad de verdaderos negativos indica que el modelo también es muy bueno en identificar tráfico normal sin etiquetarlo erróneamente como un ataque.
Falsos Positivos (FP):	7	Este número muestra las veces que el modelo predijo incorrectamente un ataque cuando en realidad era tráfico normal. Un bajo número de falsos positivos es crucial, ya que significa que el modelo no genera muchas alertas falsas.
Falsos Negativos (FN):	252	Este valor indica la cantidad de veces que el modelo no detectó un ataque, clasificando incorrectamente como tráfico normal. Aunque este número es mayor que el de falsos positivos, sigue siendo

		relativamente bajo, indicando que el modelo tiene una buena capacidad para detectar ataques, aunque hay margen para mejora.

Para evaluar la efectividad del modelo de redes neuronales para la detección y mitigación de ataques de amplificación de DNS, en términos de Precisión, Exhaustividad, F1-Score, Especificidad, Tasa de falsos positivos. Además, se realizó una comparación con otro modelo, el SVM (Máquinas de Vectores de Soporte), para evaluar y contrastar su desempeño en términos de efectividad.

Con los datos obtenidos de la matriz de confusión, se realizaron los cálculos en base a las métricas de evaluación de los modelos en el aprendizaje automático.

- Verdaderos positivos (TP): 76166
- Verdaderos negativos (TN): 14298
- Falsos positivos (FP): 7
- Falsos negativos (FN): 252

Precisión (Accuracy)

$$Precisión = \frac{TP + TN}{TP + TN + FP + FN} = \frac{76166 + 14298}{76166 + 14298 + 7 + 252} \approx 0.9972 \ (99.72\%)$$

Una precisión del 99.72% indica que el modelo clasifica correctamente la gran mayoría de las instancias, tanto positivas como negativas, siendo muy efectivo el modelo.

Precisión (Precision)

$$Precisión = \frac{TP}{TP + FP} = \frac{76166}{76166 + 7} \approx 0.9999 \ (99.99\%)$$

La precisión del 99.99% indica que casi todas las instancias clasificadas como ataques eran, en efecto, ataques, lo que mostró que el modelo cometió muy pocos errores en este aspecto.

Exhaustividad (Recall o Sensibilidad)

$$Exhaustividad = \frac{TP}{TP + FN} = \frac{76166}{76166 + 252} \approx \ 0.9967 \ (99.67\%)$$

Una exhaustividad del 99.67% indicó que el modelo detecta casi todos los ataques mostrando una alta efectividad en indicar instancias positivas.

Puntuación F1 (F1 Score)

$$Puntuacion\ F1 = 2 * \frac{Precisión * Exhaustividad}{Precisión + Exhaustividad} = 2 * \frac{0.9999 * 0.9967}{0.9999 + 0.9967} \approx 0.9982 \ (99.82\%)$$

Una puntuación F1 del 99.82% indicó que el modelo tiene un excelente equilibrio entre precisión y exhaustividad, siendo altamente efectivo en la clasificación de instancias positivas y negativas.

Especificidad (Specificity)

$$Especificidad = \frac{TN}{TN + FP} = \frac{14298}{14298 + 7} \approx 0.9995\ (99.95\%)$$

Una especificidad del 99.95% indicó que el modelo es muy eficaz para identificar correctamente el tráfico normal, cometiendo muy pocos errores al clasificar tráfico no malicioso.

Tasa de falsos positivos

$$Tasa\ de\ falsos\ positivos = \frac{FP}{FP + TN} = \frac{7}{7 + 14298} \approx 0.0005\ (0.05\%)$$

Una tasa de falsos positivos del 0.05% indicó que el modelo rara vez clasifica incorrectamente el tráfico normal como ataques, mostrando una alta fiabilidad en la identificación de tráfico no malicioso.

En la Tabla 3 se muestran las características principales del modelo de red neuronal para la detección y mitigación de ataques de amplificación de DNS.

Tabla 3. Detalles del modelo entrenado

Métrica	Valor
Tamaño de la red neuronal	3 capas: una capa densa con 64 neuronas, una capa densa con 32 neuronas, una capa densa con 1 neurona.
Tiempo de desarrollo	4 semanas
Tipo de datos	Se categorizo en tres datos: protocolo, longitud e info.
Datos de ataques	76166
Datos de tráfico normal	14298

Matriz de confusión con el modelo SVM

En la Figura 28 se muestra una matriz de confusión para evaluar la efectividad del modelo SVM utilizado en la detección y mitigación de ataques de amplificación de DNS. Esta matriz proporciona una representación visual de las predicciones realizadas por el modelo, comparándolas con las etiquetas reales de los datos. En este gráfico, se presentan cuatro valores clave que ayudan a entender cómo se desempeña el modelo en la clasificación de tráfico normal y tráfico de ataque, para evaluar su efectividad.

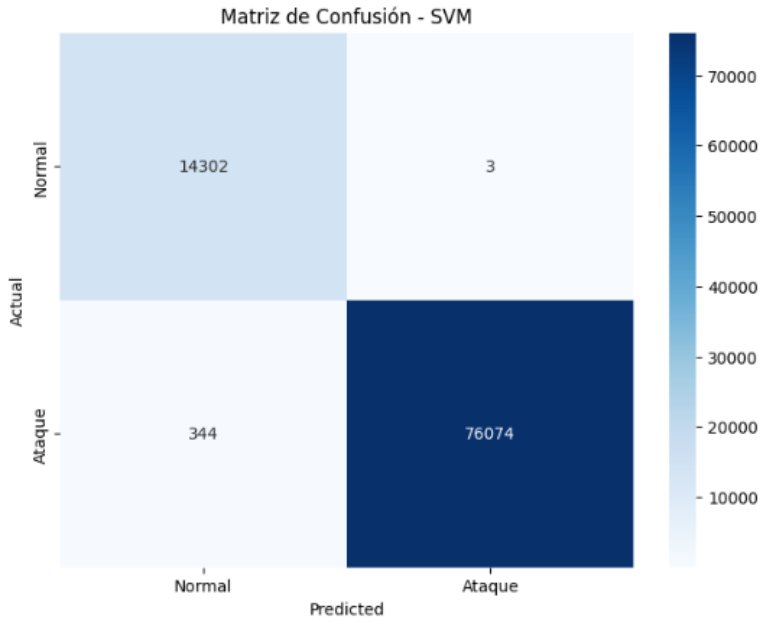


Figura 28. Matriz de Confusión de evaluación del modelo SVM

Con los datos obtenidos de la matriz de confusión del modelo SVM, se realizaron los cálculos en base a las métricas de evaluación de los modelos en el aprendizaje automático:

- Verdaderos positivos (TP): 76074
- Verdaderos negativos (TN): 14302
- Falsos positivos (FP): 3
- Falsos negativos (FN): 344

Precisión (Accuracy)

$$Precisión = \frac{TP + TN}{TP + TN + FP + FN} = \frac{76074 + 14302}{76074 + 14302 + 3 + 344} \approx 0.9961\ (99.61\%)$$

Precisión (Precision)

$$Precisión = \frac{TP}{TP + FP} = \frac{76074}{76074 + 3} \approx 0.9999\ (99.99\%)$$

Exhaustividad (Recall o Sensibilidad)

$$Exhaustividad = \frac{TP}{TP + FN} = \frac{76074}{76074 + 344} \approx\ 0.9954\ (99.54\%)$$

Puntuación F1 (F1 Score)

$$Puntuacion\ F1 = 2 * \frac{Precisión * Exhaustividad}{Precisión + Exhaustividad} = 2 * \frac{0.9999 * 0.9954}{0.9999 + 0.9954} \approx 0.9976\ (99.76\%)$$

Especificidad (Specificity)

$$Especificidad = \frac{TN}{TN + FP} = \frac{14302}{14302 + 3} \approx 0.9997\ (99.97\%)$$

Tasa de falsos positivos

$$Tasa\ de\ falsos\ positivos = \frac{FP}{FP + TN} = \frac{3}{3 + 14302} \approx 0.0002\ (0.02\%)$$

4.1 Discusión

Para evaluar la efectividad de ambos modelos se analizaron los resultados obtenidos en las pruebas de rendimiento como se muestra en la Tabla 4.

Tabla 4. Redes neuronales Vs SVM

Métrico	Redes Neuronales	SVM
Verdaderos Positivos (TP)	76166	76074
Verdaderos Negativos (TN)	14298	14302
Falsos positivos (FP)	7	3
Falsos negativos (FN)	252	344
Precisión (Accuracy)	99,72%	99,61%
Precisión (Precision)	99.99%	99,99%
Exhaustividad	99,67%	99,54%
Puntuación F1	99,82%	99,76%
Especificidad	99,95%	99,97%
Tasa de falsos positivos	0.05%	0.02%

Los dos modelos, redes neuronales y SVM, demostraron una alta efectividad en la detección de ataques de amplificación de DNS. Sin embargo, las redes neuronales superaron al SVM en varios aspectos clave. Su mayor precisión y capacidad para adaptarse a patrones complejos las convierten en una herramienta más robusta para la detección de este tipo de ataques y presentando varias ventajas significativas.

Mayor Precisión: El modelo de redes neuronales alcanzó una precisión del 99.72%, superando al SVM que obtuvo un 99.61%. Esto indica que las redes neuronales clasifican correctamente una mayor cantidad de instancias, tanto de tráfico normal como de ataque.

Mejor Exhaustividad (Recall): La exhaustividad del modelo de redes neuronales fue del 99.67%, en comparación con el 99.54% del SVM. Esto muestra que el modelo de redes neuronales es más efectivo al detectar la mayoría de los ataques, minimizando los falsos negativos.

Puntuación F1: El modelo de redes neuronales obtuvo una puntuación F1 de 99.82%, frente al 99.76% del SVM. Esta métrica equilibra precisión y exhaustividad, indicando que las redes neuronales mantienen un mejor balance en la clasificación de ataques y tráfico normal.

Menor Tasa de Falsos Positivos: Aunque ambos modelos tienen una tasa de falsos positivos muy baja, el modelo de redes neuronales logró una especificidad de 99.95%, mientras que el SVM alcanzó un 99.98%. Esto refleja que, aunque el SVM es ligeramente mejor en evitar falsos positivos, la diferencia es mínima.

Adaptabilidad a Patrones Complejos: Las redes neuronales tienen una mayor capacidad de generalización, lo que les permite adaptarse mejor a patrones complejos y no lineales presentes en los datos de tráfico de red. Esto es especialmente útil en el contexto de ataques de amplificación de DNS, donde el comportamiento malicioso puede variar ampliamente.

CAPÍTULO V. CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

Se logró investigar y desarrollar un modelo de redes neuronales capaz de identificar patrones asociados con ataques de amplificación de DNS. Este proceso incluyó el análisis exhaustivo de los datos de tráfico de red y la identificación de características clave para la detección de estos ataques. La aplicación de la metodología KDD permitió garantizar un enfoque sistemático en la preparación y selección de los datos, contribuyendo significativamente a la efectividad del modelo.

El modelo fue implementado con éxito en un entorno controlado, utilizando GNS3 para simular redes reales y TensorFlow Serving para desplegar el modelo en tiempo real. Esto permitió evaluar el rendimiento del sistema en condiciones similares a un entorno de producción, demostrando su capacidad para clasificar de manera eficiente el tráfico legítimo y el tráfico malicioso.

La validación realizada con Apache Spark confirmó la alta efectividad del modelo desarrollado. Los resultados obtenidos, con métricas como una precisión del 99.72%, un recall del 99.67%, y una tasa de falsos positivos del 0.05%, reflejan un desempeño sobresaliente en la detección de patrones asociados con ataques de amplificación de DNS. Esto demuestra que el modelo no solo cumple con los objetivos planteados, sino que también establece un estándar para futuras investigaciones en el ámbito de la ciberseguridad.

5.2 Recomendaciones

A pesar de los excelentes resultados, recomiendo que el modelo sea actualizado para asegurar su escalabilidad. Esto podría incluir el uso de técnicas de ensamblaje, la exploración de diferentes arquitecturas de redes neuronales o el ajuste de hiperparámetros adicionales.

Debido a la naturaleza cambiante de los ataques de amplificación de DNS, considero crucial monitorear continuamente el rendimiento del modelo y actualizarlo con nuevos datos y patrones de ataque. Esto garantizaría que el modelo permanezca resistente a nuevas formas de ataques.

Se sugiere continuar con la investigación y el desarrollo de modelos adicionales utilizando tecnologías emergentes, como redes neuronales profundas (aprendizaje profundo) o técnicas de aprendizaje automático no supervisado, para mejorar la detección y la mitigación de ataques.

BIBLIOGRÁFIA

[1] V. Jairo, "UNIVERSIDAD NACIONAL DE CHIMBORAZO FACULTAD DE INGENIERÍA," *29*, 2021.

[2] cloudflare, "Ataque DDoS por amplificación de DNS | Cloudflare." Accessed: Mar. 10, 2024. [Online]. Available: https://www.cloudflare.com/es-es/learning/ddos/dns-amplification-ddos-attack/

[3] O. Tirosh, "Redes neuronales: definición, tipos, beneficios y casos de éxito." Accessed: Apr. 27, 2024. [Online]. Available: https://es.tomedes.com/blog-de-traduccion/redes-neuronales-definicion-tipos-beneficios-casos-de-exito

[4] Y. Musienki, "Pros y Contras de la Arquitectura de Redes Neuronales - Merehead." Accessed: Apr. 27, 2024. [Online]. Available: https://merehead.com/es/blog/pros-contras-arquitectura-redes-neuronales/

[5] T. School, "Arquitectura típica de una red neuronal convolucional." Accessed: Apr. 27, 2024. [Online]. Available: https://keepcoding.io/blog/arquitectura-tipica-red-neuronal-convolucional/

[6] A. Aguilar, "Ataque de Amplificación DNS: Entendiendo y Mitigando el Riesgo." Accessed: Apr. 28, 2024. [Online]. Available: https://blog.tecnetone.com/ataque-de-amplificaci%C3%B3n-dns-qu%C3%A9-son-y-como-funcionan

[7] CLOUDFLARE, "¿Qué es una inundación de DNS? | Ataque DDoS de inundación de consultas DNS | Cloudflare." Accessed: Apr. 28, 2024. [Online]. Available: https://www.cloudflare.com/es-es/learning/ddos/dns-flood-ddos-attack/

[8] IBM, "¿Qué es la seguridad de red? | IBM," IBM. Accessed: Apr. 28, 2024. [Online]. Available: https://www.ibm.com/mx-es/topics/network-security

[9] CLOUDFLARE, "Ataque DDoS por amplificación de DNS | Cloudflare." Accessed: Apr. 28, 2024. [Online]. Available: https://www.cloudflare.com/es-es/learning/ddos/dns-amplification-ddos-attack/

[10] CLOUDFLARE, "Ataque DDoS por amplificación de DNS | Cloudflare." Accessed: Apr. 28, 2024. [Online]. Available: https://www.cloudflare.com/es-es/learning/ddos/dns-amplification-ddos-attack/

[11] T. Cubillo, D. Tutor, B. Navarro, V. Juan Cotutor, and P. Cámara, "UNIVERSITAT POLITÈCNICA DE VALÈNCIA Escuela Técnica Superior de Ingeniería Informática," 2020.

[12] D. I. Quirumbay Yagual, C. Castillo Yagual, and I. Coronel Suárez, “Una revisión del Aprendizaje profundo aplicado a la ciberseguridad,” *Revista Científica y Tecnológica UPSE*, vol. 9, no. 1, pp. 57–65, Jun. 2022, doi: 10.26423/RCTU.V9I1.671.

[13] B. J. Daniel, A. Silvia, and V. Laura, “Detección de Intrusiones mediante el uso de Redes Neuronales,” 2023.

[14] WordPres, “6. Redes neuronales y su aplicación en la seguridad informática - Inteligencia Artificial.” Accessed: Apr. 28, 2024. [Online]. Available: https://inteligenciaartificial.science/6-redes-neuronales-y-su-aplicacion-en-la-seguridad-informatica/

[15] P. Lozano, “Escaneo de vulnerabilidades: Herramientas y técnicas | OpenWebinars.” Accessed: Apr. 28, 2024. [Online]. Available: https://openwebinars.net/blog/escaneo-de-vulnerabilidades/

[16] IBM, “El modelo de redes neuronales - Documentación de IBM.” Accessed: Apr. 28, 2024. [Online]. Available: https://www.ibm.com/docs/es/spss-modeler/saas?topic=networks-neural-model

[17] P. Huet, “Entrenar un modelo de Machine Learning con Scikit-learn | OpenWebinars.” Accessed: Apr. 28, 2024. [Online]. Available: https://openwebinars.net/blog/como-entrenar-un-modelo-de-machine-learning-con-scikit-learn/

[18] J. Barrios, “La matriz de confusión y sus métricas – Inteligencia Artificial –.” Accessed: Apr. 28, 2024. [Online]. Available: https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/

[19] Universidad Complutense Madrid, “UCM-Proyecto de Innovación Software libre para ciencias e ingenierías.” Accessed: Nov. 10, 2024. [Online]. Available: https://www.ucm.es/pimcd2014-free-software/gns3

[20] IBM, “¿Qué es Apache Spark? | IBM.” Accessed: Nov. 10, 2024. [Online]. Available: https://www.ibm.com/mx-es/topics/apache-spark

[21] S. R. , H.-A. I. , C.-Z. S. J. , H.-T. A. y A. J. C. Timarán-Pereira, *El proceso de descubrimiento de conocimiento en bases de datos*, Descubrimiento., vol. 2. 2020.

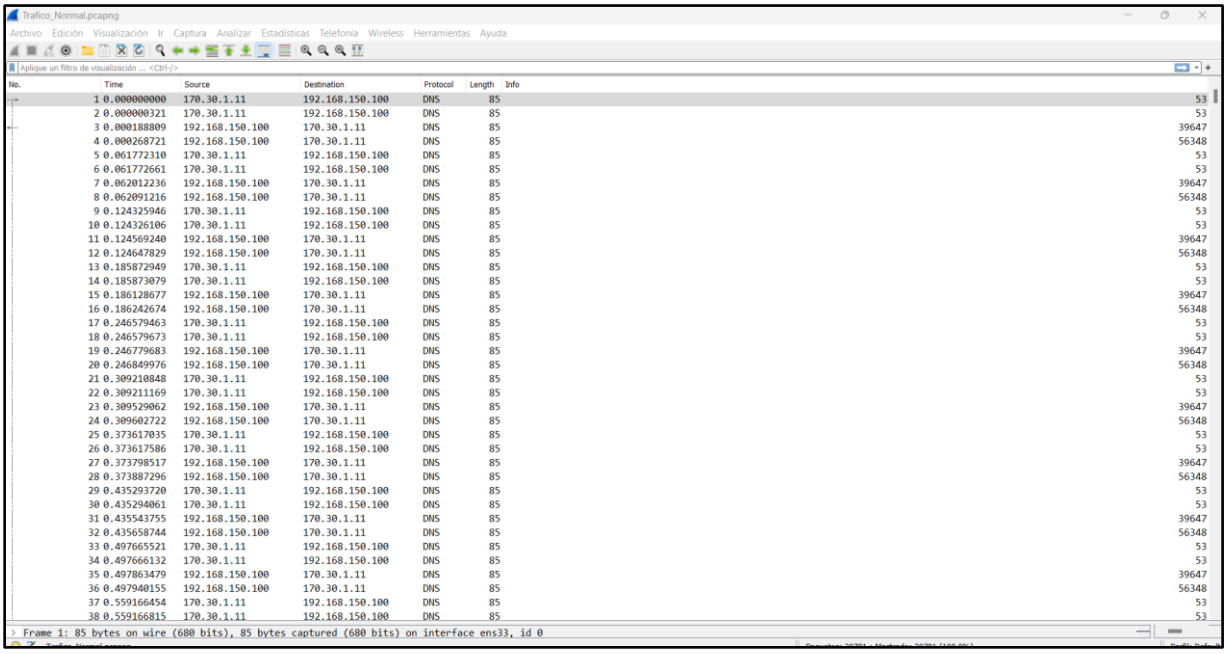
[22] Fernando Paul Lara Galacia, “UCM-Proyecto de Innovación Software libre para ciencias e ingenierías.” Accessed: Nov. 10, 2024. [Online]. Available: https://www.ucm.es/pimcd2014-free-software/gns3

[23] I. S. Stephanie Susnjara, “¿Qué es Docker? | IBM.” Accessed: Nov. 12, 2024. [Online]. Available: https://www.ibm.com/es-es/topics/docker

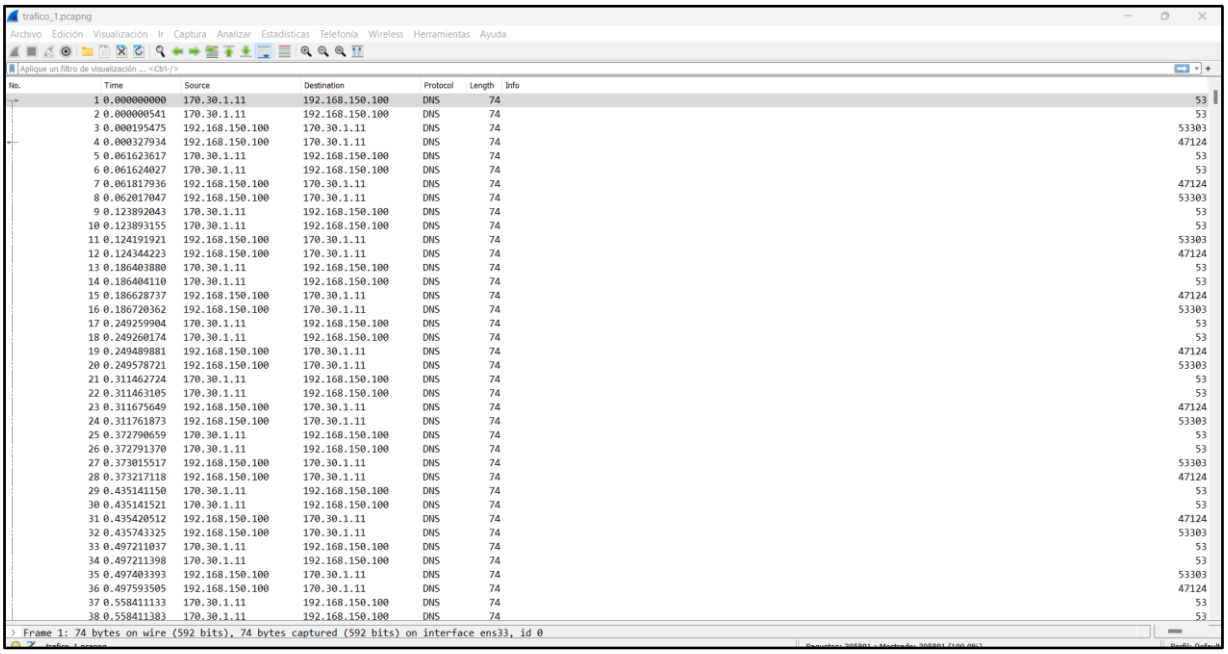
ANEXOS

Recolección de datos en la Universidad Nacional de Chimborazo

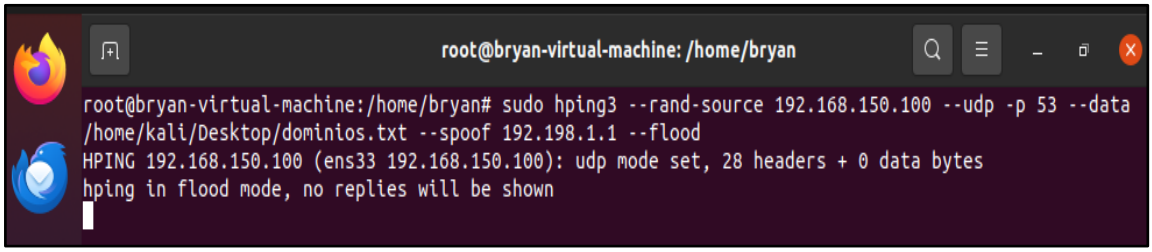
Trafico normal



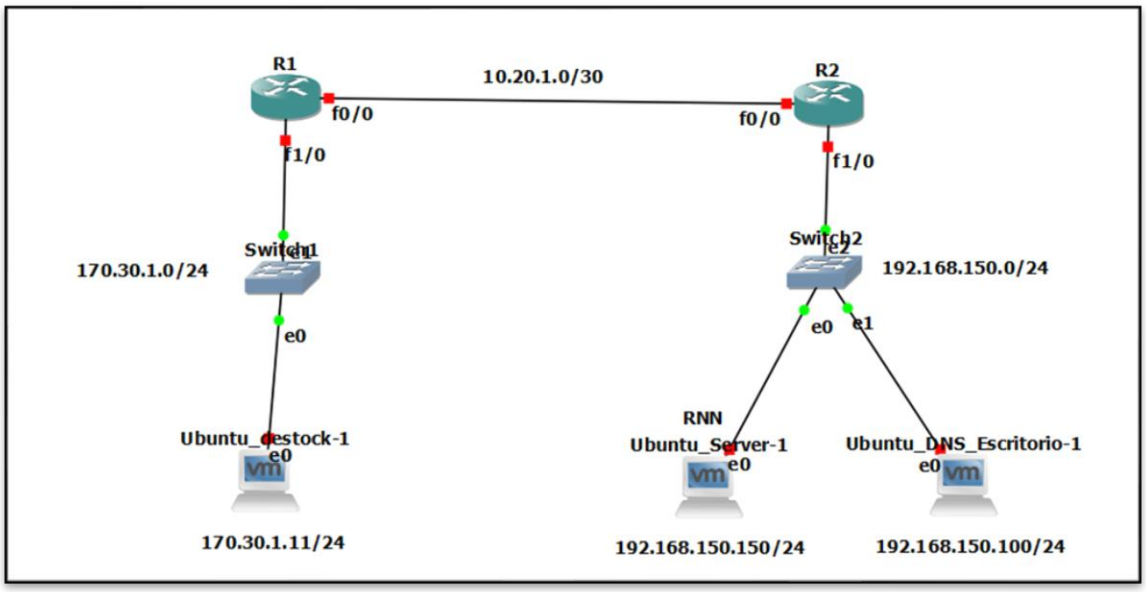
Tráfico de ataques



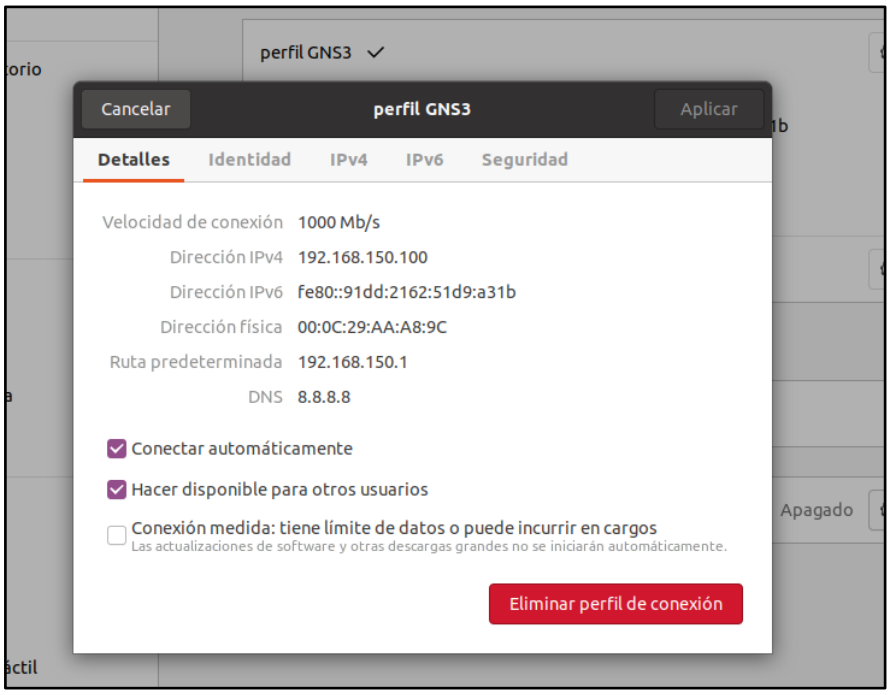
Comando para realizar un ataque de aplificacion de DNS



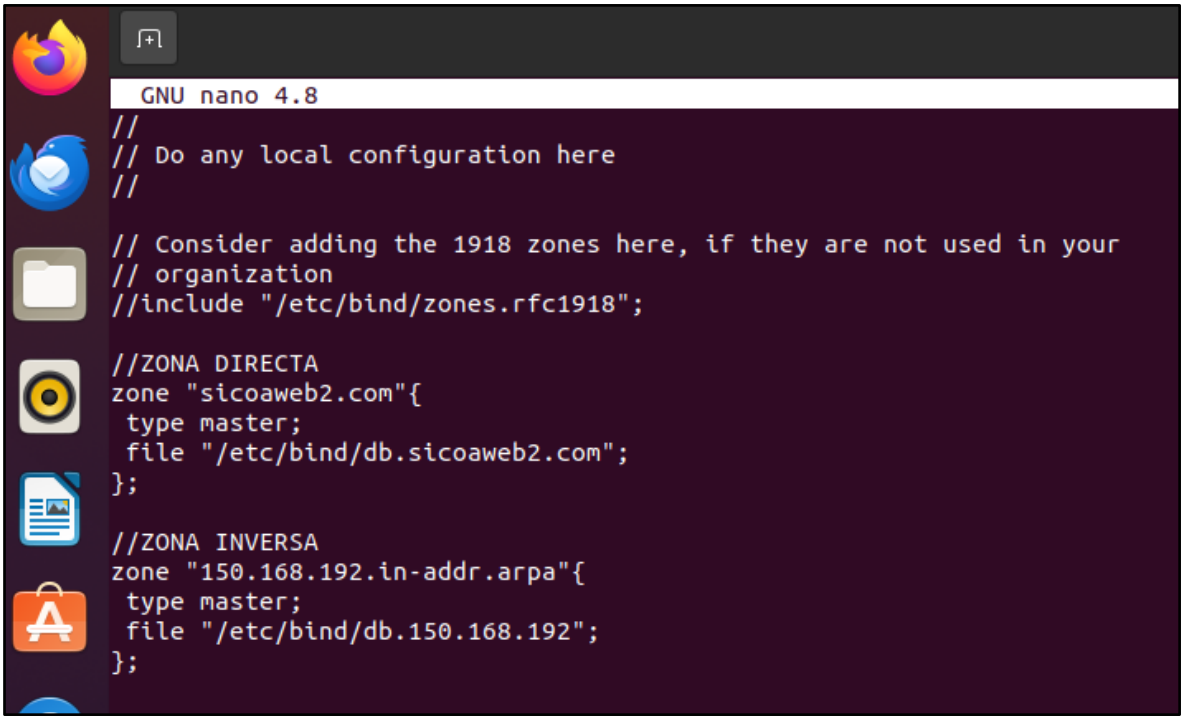
Topología de red en GNS3 con máquinas virtuales



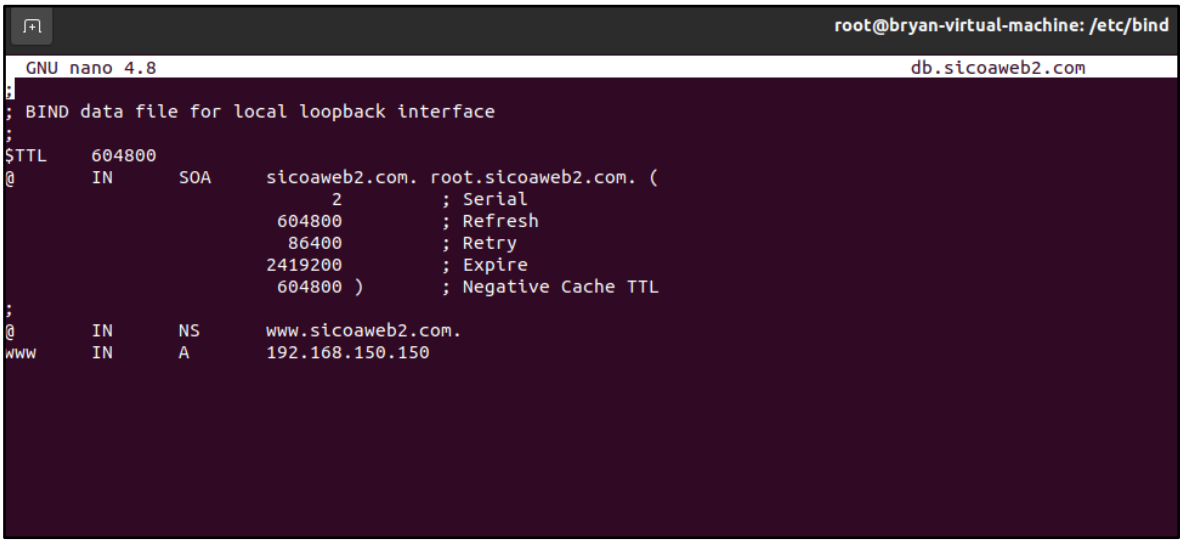
Ubuntu DNS Escritorio



Configuracion de zonas directa e inversas

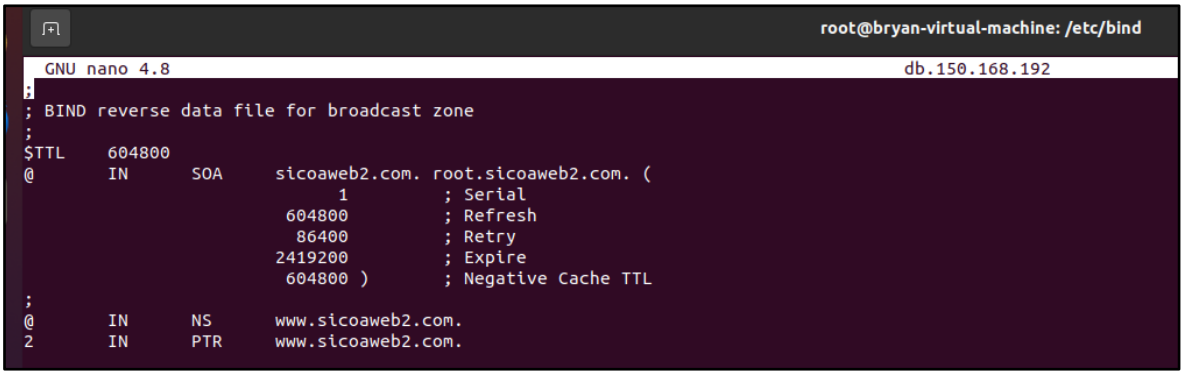


```
 GNU nano 4.8
//
// Do any local configuration here
//

// Consider adding the 1918 zones here, if they are not used in your
// organization
//include "/etc/bind/zones.rfc1918";

//ZONA DIRECTA
zone "sicoaweb2.com"{
 type master;
 file "/etc/bind/db.sicoaweb2.com";
};

//ZONA INVERSA
zone "150.168.192.in-addr.arpa"{
 type master;
 file "/etc/bind/db.150.168.192";
};
```



root@bryan-virtual-machine: /etc/bind

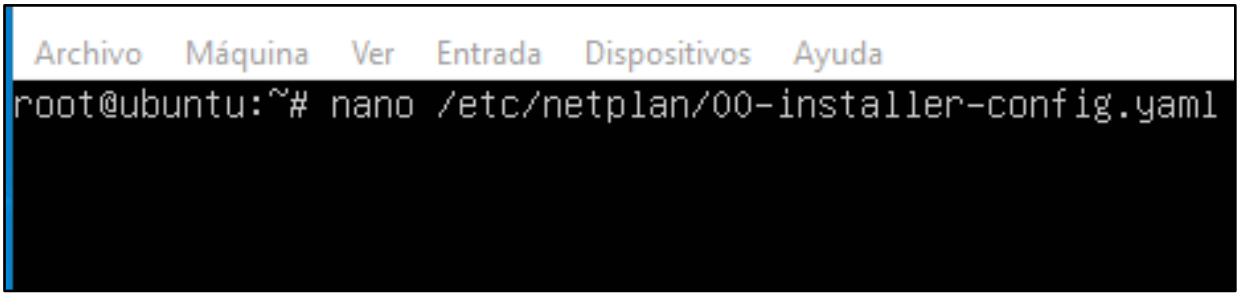
```
 GNU nano 4.8                                                db.sicoaweb2.com
;
; BIND data file for local loopback interface
;
$TTL    604800
@       IN      SOA     sicoaweb2.com. root.sicoaweb2.com. (
                              2         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      www.sicoaweb2.com.
www     IN      A       192.168.150.150
```

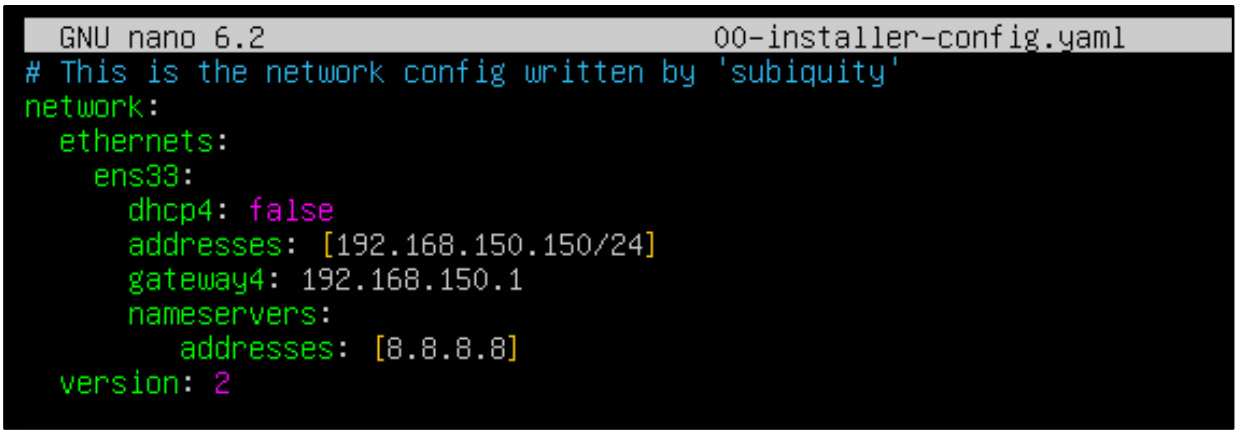


root@bryan-virtual-machine: /etc/bind

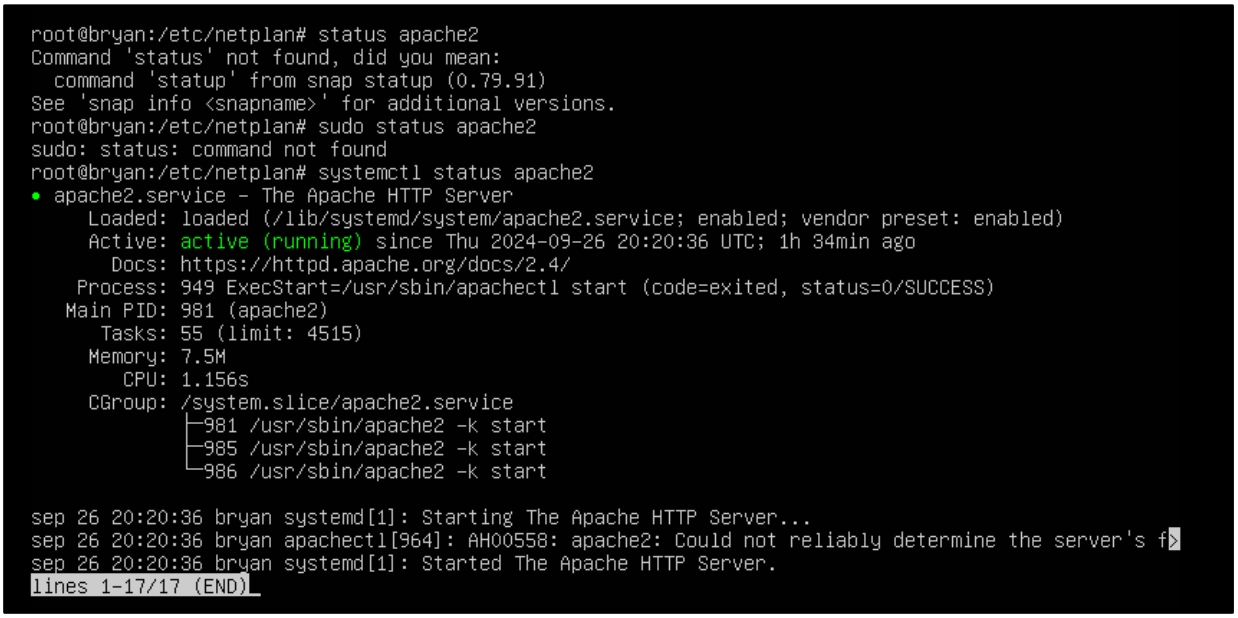
```
 GNU nano 4.8                                                db.150.168.192
;
; BIND reverse data file for broadcast zone
;
$TTL    604800
@       IN      SOA     sicoaweb2.com. root.sicoaweb2.com. (
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      www.sicoaweb2.com.
2       IN      PTR     www.sicoaweb2.com.
```

Ubuntu server web- apache

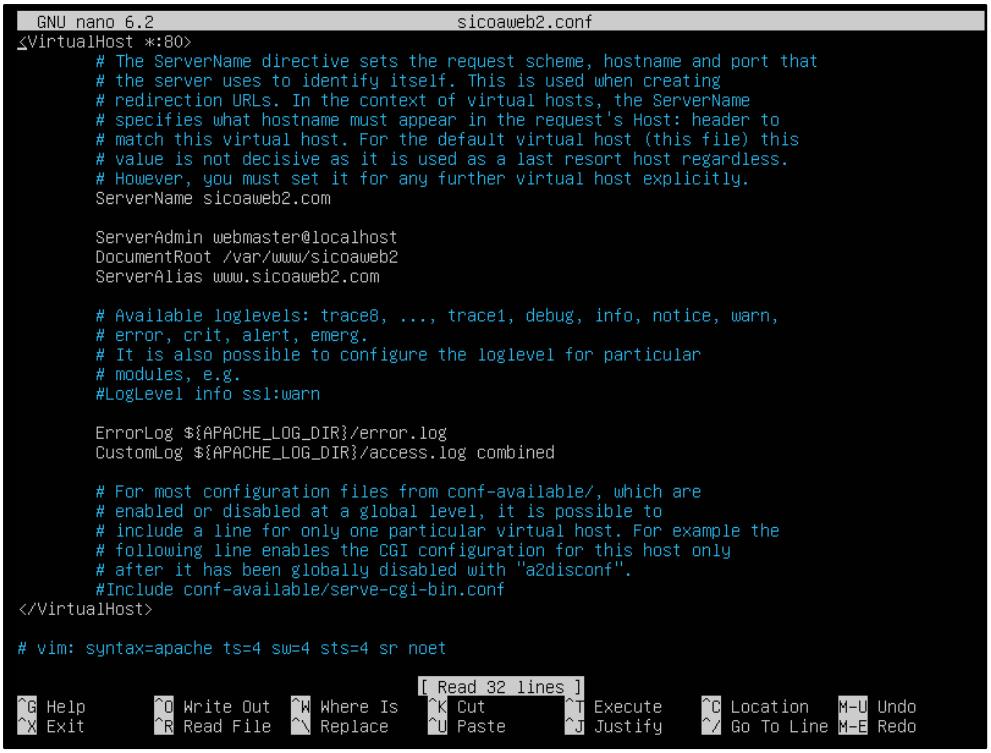


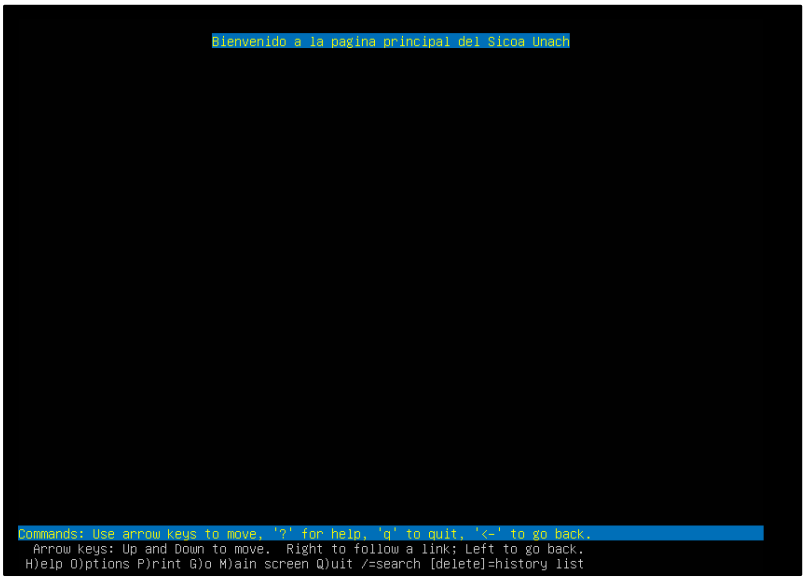


Instalación de apache 2



Configuracion de los host virtuales





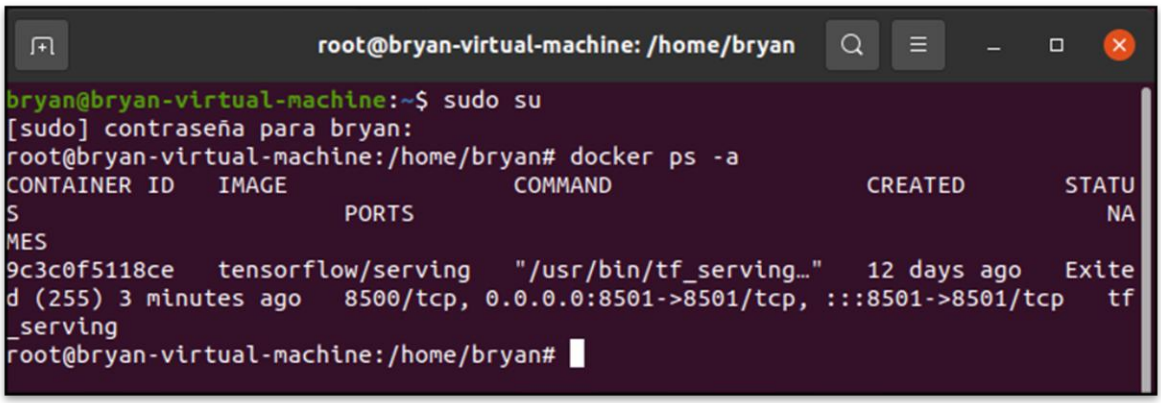
Ubuntu cliente

Procesamiento de los datos y elaboración de la red neuronal perceptrón multicapa

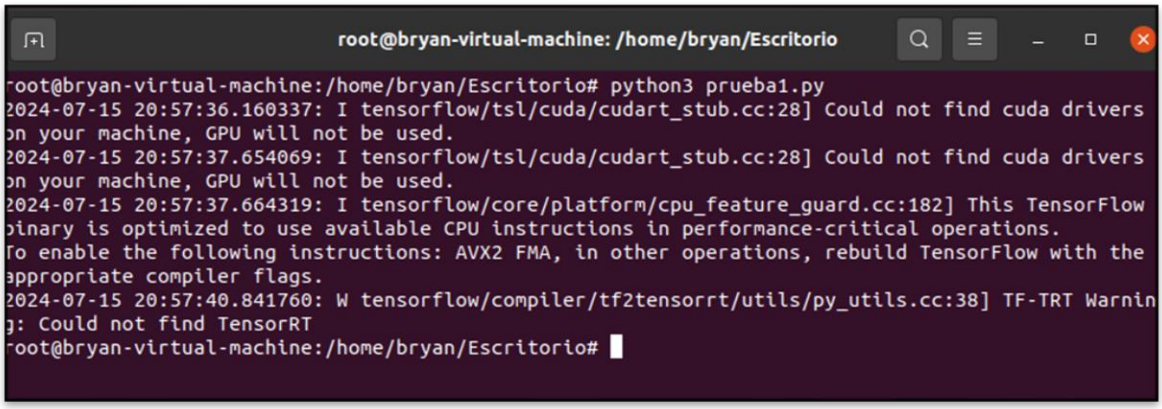
```
# Cargar datos
data_ataque = pd.read_csv("/content/drive/MyDrive/tesis_pruebas/trafico_ataque_1.csv", sep=';')
data_normal = pd.read_csv("/content/drive/MyDrive/tesis_pruebas/trafico_normal.csv", sep=';')
# Añadir columna de etiqueta
data_ataque['Etiqueta'] = 1
data_normal['Etiqueta'] = 0
# Combinar DataFrames
data = pd.concat([data_ataque, data_normal])
# Convertir la columna 'Protocolo' a valores numéricos usando LabelEncoder
le = LabelEncoder()
data['Protocolo'] = le.fit_transform(data['Protocolo'])
# Comprobar y eliminar valores nulos o infinitos
data.replace([np.inf, -np.inf], np.nan, inplace=True)
data.dropna(inplace=True)
# Normalizar características
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
# Construir el modelo
model = Sequential()
model.add(Dense(64, input_dim=X_train.shape[1], activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
# Compilar el modelo
model.compile(optimizer=Adam(learning_rate=0.0001), loss='binary_crossentropy',
metrics=['accuracy'])
# Entrenar el modelo
history = model.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.2)
# Evaluar el modelo
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Loss: {loss}, Accuracy: {accuracy}')
# Predicciones y reporte de clasificación
y_pred = (model.predict(X_test) > 0.5).astype(int)
print(classification_report(y_test, y_pred))
# Guardar el modelo entrenado en un archivo .h5
model.save('model.h5') from google.colab import files
# Descargar el archivo model.h5 a tu máquina local
files.download('model.h5')
```

Implementación de la red neuronal en el servidor DNS

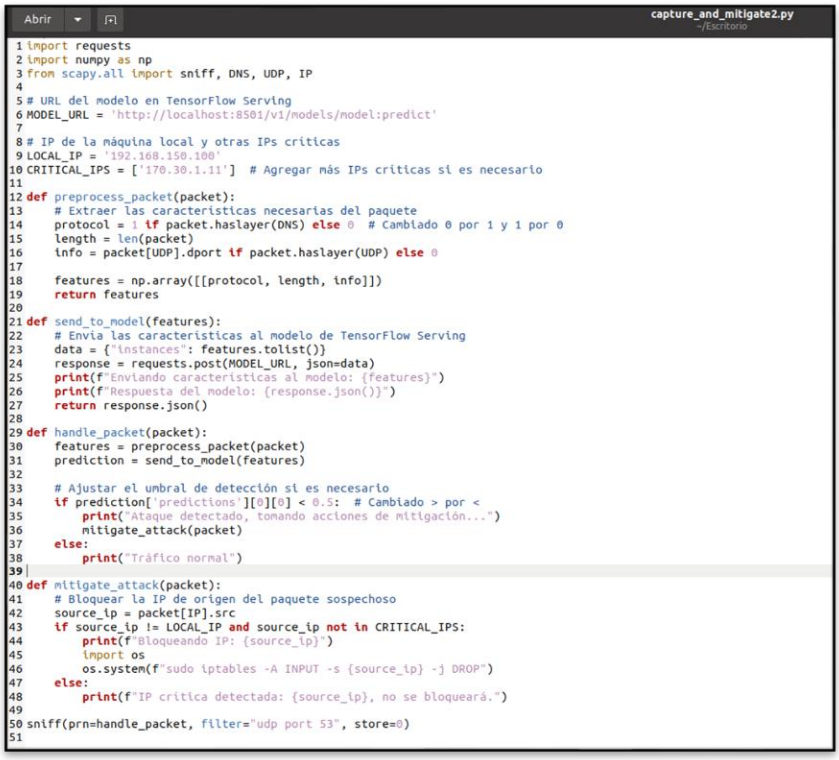
Configuración de Docker



Inicialización de TensorFlow

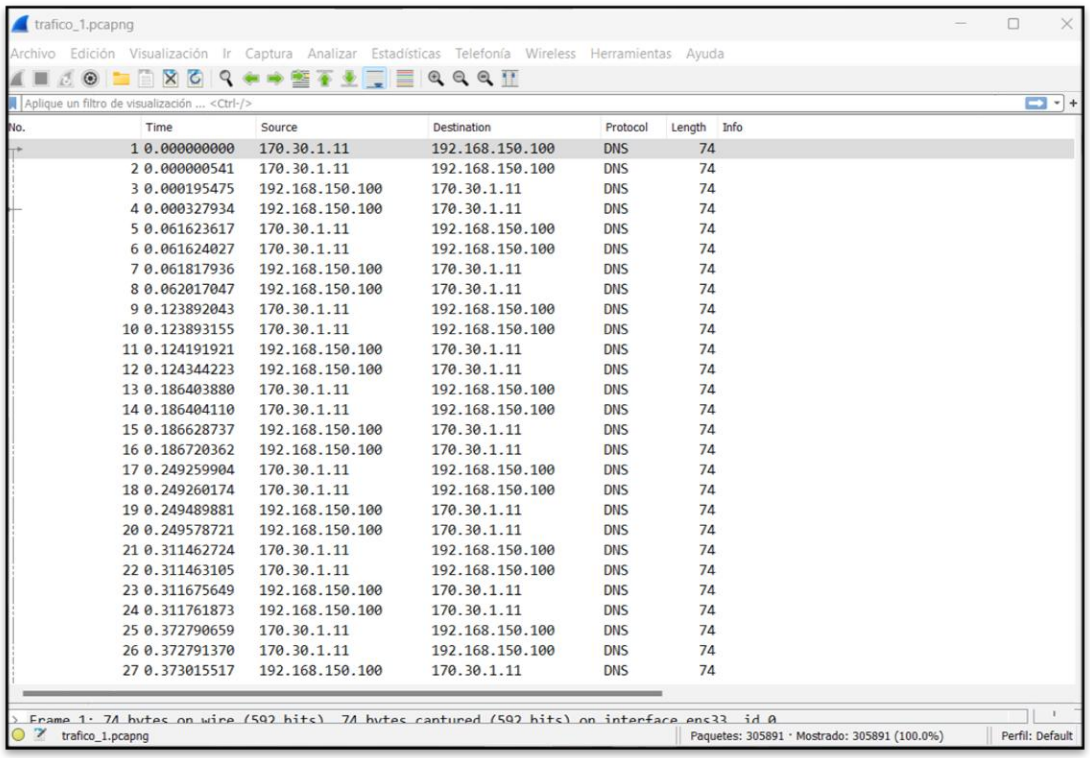


Script para capturar el tráfico de red en tiempo real

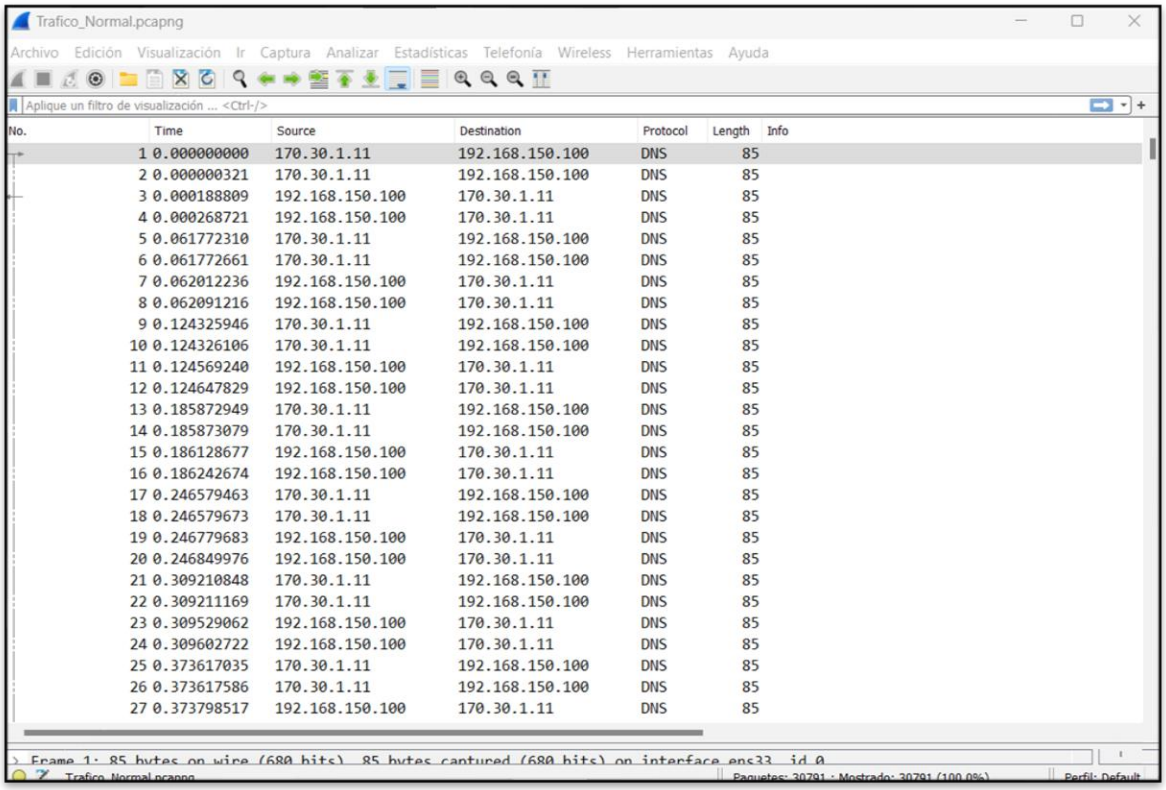


Recolección de datos de tráfico de la simulación

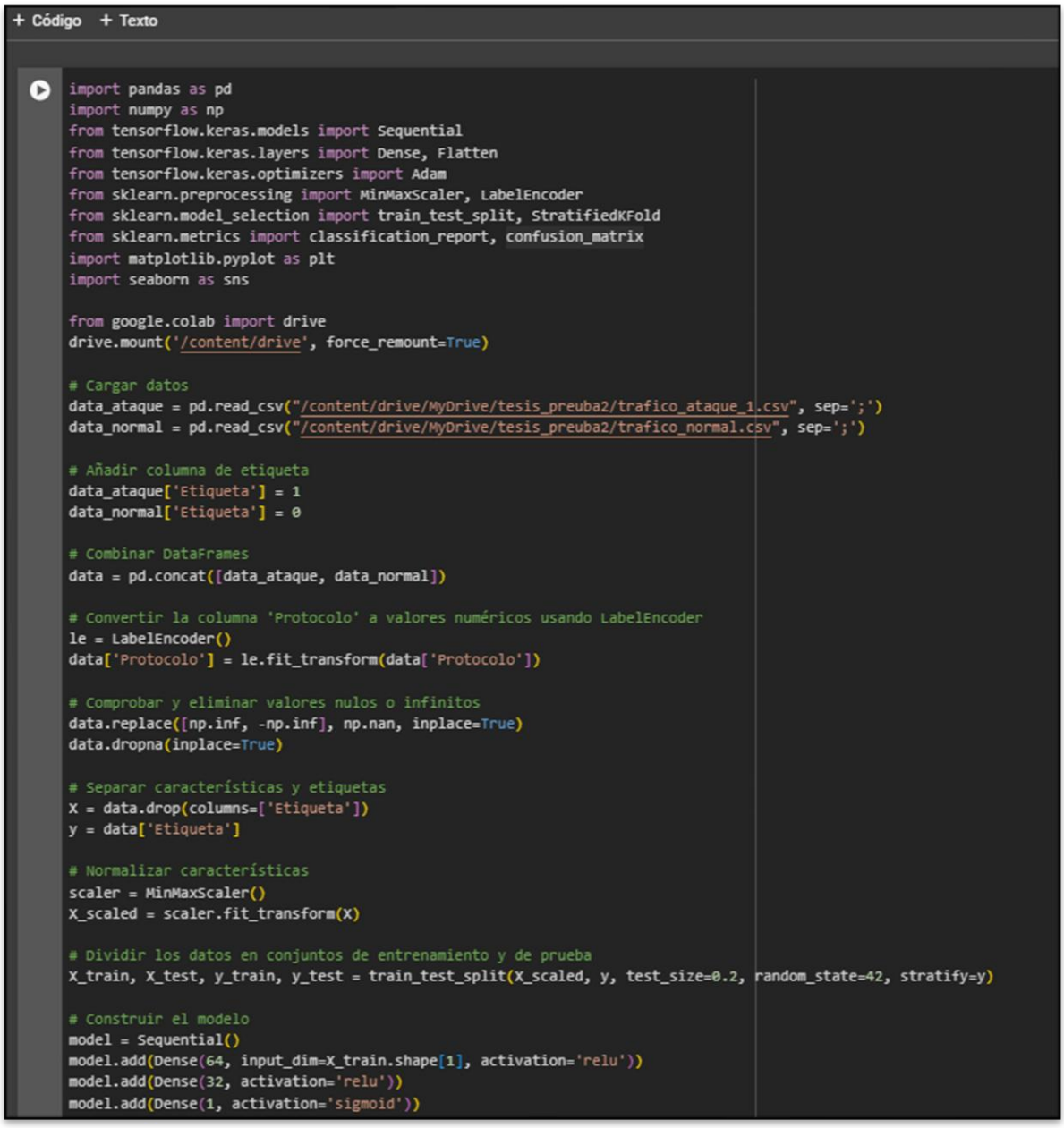
Tráfico normal



Tráfico de ataque



Modelo de redes neuronales para la evaluación



+ Código + Texto

```
import pandas as pd
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.optimizers import Adam
from sklearn.preprocessing import MinMaxScaler, LabelEncoder
from sklearn.model_selection import train_test_split, StratifiedKFold
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

from google.colab import drive
drive.mount('/content/drive', force_remount=True)

# Cargar datos
data_ataque = pd.read_csv("/content/drive/MyDrive/tesis_preuba2/trafico_ataque_1.csv", sep=';')
data_normal = pd.read_csv("/content/drive/MyDrive/tesis_preuba2/trafico_normal.csv", sep=';')

# Añadir columna de etiqueta
data_ataque['Etiqueta'] = 1
data_normal['Etiqueta'] = 0

# Combinar DataFrames
data = pd.concat([data_ataque, data_normal])

# Convertir la columna 'Protocolo' a valores numéricos usando LabelEncoder
le = LabelEncoder()
data['Protocolo'] = le.fit_transform(data['Protocolo'])

# Comprobar y eliminar valores nulos o infinitos
data.replace([np.inf, -np.inf], np.nan, inplace=True)
data.dropna(inplace=True)

# Separar características y etiquetas
X = data.drop(columns=['Etiqueta'])
y = data['Etiqueta']

# Normalizar características
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)

# Dividir los datos en conjuntos de entrenamiento y de prueba
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42, stratify=y)

# Construir el modelo
model = Sequential()
model.add(Dense(64, input_dim=X_train.shape[1], activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

Reporte para la evaluación

+ Código + Texto

```
# Compilar el modelo
model.compile(optimizer=Adam(learning_rate=0.0001), loss='binary_crossentropy', metrics=['accuracy'])

# Entrenar el modelo
history = model.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.2)

# Evaluar el modelo
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Loss: {loss}, Accuracy: {accuracy}')

# Predicciones y reporte de clasificación
y_pred = (model.predict(X_test) > 0.5).astype(int)
print(classification_report(y_test, y_pred))

# Graficar la pérdida y precisión durante el entrenamiento
plt.figure(figsize=(14, 5))

# Pérdida
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.title('Loss Over Epochs')

# Precisión
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Accuracy Over Epochs')

plt.show()

# Matriz de confusión
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=['Normal', 'Ataque'], yticklabels=['Normal', 'Ataque'])
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion Matrix')
plt.show()
```

Resultados de la red neuronal entrenada

```
Epoch 41/50
9073/9073 [==============================] - 30s 3ms/step - loss: 0.0094 - accuracy: 0.9971 - val_loss: 0.0086 - val_accuracy: 0.9975
Epoch 42/50
9073/9073 [==============================] - 27s 3ms/step - loss: 0.0094 - accuracy: 0.9971 - val_loss: 0.0086 - val_accuracy: 0.9975
Epoch 43/50
9073/9073 [==============================] - 28s 3ms/step - loss: 0.0093 - accuracy: 0.9971 - val_loss: 0.0085 - val_accuracy: 0.9975
Epoch 44/50
9073/9073 [==============================] - 27s 3ms/step - loss: 0.0093 - accuracy: 0.9971 - val_loss: 0.0085 - val_accuracy: 0.9975
Epoch 45/50
9073/9073 [==============================] - 27s 3ms/step - loss: 0.0093 - accuracy: 0.9971 - val_loss: 0.0085 - val_accuracy: 0.9975
Epoch 46/50
9073/9073 [==============================] - 27s 3ms/step - loss: 0.0093 - accuracy: 0.9971 - val_loss: 0.0086 - val_accuracy: 0.9975
Epoch 47/50
9073/9073 [==============================] - 26s 3ms/step - loss: 0.0093 - accuracy: 0.9971 - val_loss: 0.0089 - val_accuracy: 0.9976
Epoch 48/50
9073/9073 [==============================] - 27s 3ms/step - loss: 0.0092 - accuracy: 0.9971 - val_loss: 0.0084 - val_accuracy: 0.9975
Epoch 49/50
9073/9073 [==============================] - 27s 3ms/step - loss: 0.0092 - accuracy: 0.9971 - val_loss: 0.0085 - val_accuracy: 0.9976
Epoch 50/50
9073/9073 [==============================] - 26s 3ms/step - loss: 0.0092 - accuracy: 0.9971 - val_loss: 0.0083 - val_accuracy: 0.9976
2836/2836 [==============================] - 5s 2ms/step - loss: 0.0093 - accuracy: 0.9971
Loss: 0.009323048405349255, Accuracy: 0.9971451759338379
2836/2836 [==============================] - 5s 2ms/step
              precision    recall  f1-score   support

           0       0.98      1.00      0.99     14305
           1       1.00      1.00      1.00     76418

    accuracy                           1.00     90723
   macro avg       0.99      1.00      0.99     90723
weighted avg       1.00      1.00      1.00     90723
```

Graficas para la evaluación del modelo de redes neuronales