FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

DESARROLLO DE UNA RED NEURONAL MLP PARA LA DETECCIÓN DE BOTNETS EN EL CONJUNTO DE DATOS UNSW BoT-IoT.

AUTOR

Victor Manuel Totoy Granja

AÑO

2020

FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

DESARROLLO DE UNA RED NEURONAL MLP PARA LA DETECCIÓN DE BOTNETS EN EL CONJUNTO DE DATOS UNSW BoT-IoT.

Trabajo de titulación presentado en conformidad con los requisitos establecidos para optar por el título de Ingeniero en Electrónica y Redes de información.

Profesor Guía

Msg. Iván Patricio Ortiz Garcés

Autor

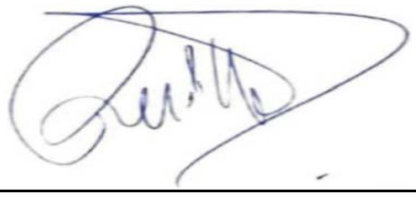
Victor Manuel Totoy Granja

Año

2020

DECLARACIÓN DEL PROFESOR GUÍA

“Declaro haber dirigido el trabajo, Desarrollo de una red neuronal MLP para la detección de botnets en el conjunto de datos UNSW-BoT-IoT, a través de reuniones periódicas con el estudiante Victor Manuel Totoy Granja, en el semestre 202020, orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación”.



Iván Patricio Ortiz Garcés

Magíster en Redes de Comunicaciones

CI: 060235677-6

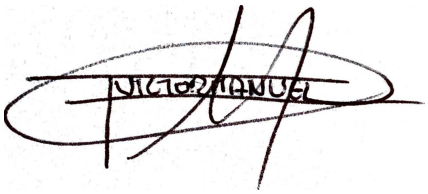
DECLARACIÓN DEL PROFESOR CORRECTOR

“Declaro haber revisado este trabajo, Desarrollo de una red neuronal MLP para la detección de botnets en el conjunto de datos UNSW-BoT-IoT, del estudiante Victor Manuel Totoy Granja, en el semestre 202020, dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación”.

William Eduardo Villegas Chiliquinga
Magister en Redes de Comunicaciones
C.I.: 1715338263

DECLARACIÓN DE AUTORÍA DE LOS ESTUDIANTES

“Declaro que este trabajo es original, de mi autoría, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes”.



Victor Manuel Totoy Granja

CI: 172424431-2

AGRADECIMIENTO

Quiero agradecer infinitamente a mi madre a quien a lo largo de mi vida me a dado apoyo incondicional para poder cumplir mis objetivos y metas, también quiero agradecer a mis hermanos que me han a poyado en este, un triunfo más de mi vida.

DEDICATORIA

Mi Tesis la dedico con todo mi cariño a mis padres Virginia Granja y Marcelo Totoy, por todo el sacrificio y esfuerzo que dieron por mí. A mi hermano Mario Marcelo Totoy quien es como un padre para mí. Y a todas las personas que me ayudaron a lo largo de este camino.

RESUMEN

El presente proyecto trata del desarrollo de una red neuronal MLP que detecta ataques maliciosos dentro de la red; introduciendo las redes neuronales y dando a conocer las características y modelos utilizados, para explicar teóricamente la detección de botnets en el conjunto de datos UNSW-BoT-IoT.

Se tomó el conjunto de datos llamado UNSW-BoT-IoT los cuales contienen ataques botnet entre ellos DDos,Dos, se analizó los datos, posteriormente se utilizó los ataques de BoT para poder procesarlos con ayuda de la aplicación Pycharm y sus herramientas Pandas, NumPy y sklearn, que son herramientas para procesar datos.

Una vez procesados los datos, se realizó la construcción de la red neuronal por medio del modelo perceptrón multicapa (MLP). Se insertaron los datos procesados y se entrenó a esta red, la cual utiliza propagación hacia atrás o también llamado delta para que pueda identificar los ataques botnet y así aprender sola y seguir detectando ataques posteriores botnet.

ABSTRACT

This project deals with the development of an MLP neural network that detects malicious attacks within the network; introducing neural networks and presenting the characteristics and models used to theoretically explain the detection of botnets in the UNSW-BoT-IoT data set.

The data set called UNSW-BoT-IoT was taken, which contains botnet attacks including DDos, Dos, the data was analyzed, later the BoT attacks were used to process them with the help of the Pycharm application and its tools Pandas, NumPy and sklearn, which are tools for processing data.

Once the data had been processed, the construction of the neural network was carried out by means of the multilayer perceptron model (MLP). The processed data was inserted and this network was trained, which uses backward propagation or also called delta so that it can identify botnet attacks and thus learn by itself and continue to detect subsequent botnet attacks.

ÍNDICE

1. CAPÍTULO I: MARCO TEÓRICO 4

1.1. Machine Learning 4

1.2. Red Neuronal Artificial 4

1.3. Modelo Neuronal Perceptrón 5

1.3.1. Modelo Neuronal Perceptrón Multicapa 6

1.4. Seguridad de la Información 7

1.5. Ataques Informáticos 7

1.5.1. Botnets 8

1.5.1.1. Modelo Cliente-servidor 8

1.5.1.2. Modelo peer-to-peer 9

1.5.2. Ataques de red distribuidos 10

1.6. Internet de las Cosas 10

1.6.1. IoT cloud. 10

1.7. Cloud Computing 11

1.7.1. Cloud Publica 12

1.7.2. Cloud Privada 12

1.7.3. Cloud Hibrida 13

1.7.4. Infraestructura como servicio 13

1.7.5. Plataforma como servicio 14
1.7.6. Software como servicio 14
1.8. Bases de Datos 14
1.8.1. Sistema de gestión de base de datos 15
1.9. Phyton 16
1.9.1. Scapy 16
1.9.2. NumPy 16
1.9.3. TensorFlow 17
2. CAPITULO II. MODELOS DE REDES NEURONALES 17
2.1. Red Neuronal Perceptrón Multicapa 18
2.1.1. Arquitectura Red Neuronal Perceptrón Multicapa 19
2.1.2. Algoritmo de Propagación hacia atrás 20
2.1.3. Aplicación de un perceptrón multicapa 20
2.1.4. Selección de variables, procesamientos en un perceptrón multicapa 20
2.2. Red Neuronal Memoria Asociativa Bidireccional 20
2.2.1. Arquitectura de la red neuronal Memoria Asociativa Bidireccional 21
2.2.2. Memoria Heteroasociativa 22
2.2.3. Aprendizaje de una red neuronal Memoria Asociativa Bidireccional . 22
2.3. Red Neuronal Learning Vector Quantization 22

2.3.1. Arquitectura de una red Neuronal Learning Vector Quantization 23
2.3.2. Aprendizaje de una red neuronal Learning Vector Quantization 24
2.3.2.1. Algoritmo de aprendizaje de una red neuronal Learning Vector Quantization 25
2.4. Red Neuronal Self-Organizings Maps 25
2.4.1. Arquitectura de una red neuronal Self-Organizings Maps 26
2.4.2. Aprendizaje competitivo 27
2.4.3. Algoritmo de una red neuronal Self-Organizings Maps 27
2.5. Red Neuronal Cascada-Correlación 28
2.5.1. Arquitectura de una red neuronal Cascada-Correlación 28
2.5.2. Aprendizaje Constructivo 29
3. CAPÍTULO III: CONJUNTO DE DATOS USNW-BoT-IoT 31
3.1. Redes Internet de las Cosas 31
3.2. Conjunto de datos BoT-IoT 32
3.2.1. Plataformas de red 32
3.2.2. Servicios IoT simulados 33
3.2.3. Escenarios Benignos 35
3.2.4. Escenarios Botnet 36
3.3. Banco de Pruebas 37

3.3.1. Conjunto de datos DARPA 98 39
3.3.2. Conjunto de datos KDD99 39
3.3.3. Conjunto de datos DEFCON-8 39
3.3.4. Conjunto de datos UNIBS 39
3.3.5. Conjunto de datos CAIDA 40
3.3.6. Conjunto de datos LBNL 40
3.3.7. Conjunto de datos UNSW-NB15 40
3.3.8. Conjunto de datos ISCX 41
3.3.9. Conjunto de datos TUIDS 41
4. CAPÍTULO IV: IMPLEMENTACION DE MODELO DE RED NEURONAL PERCEPTRON MULTICAPA 41
4.1. Descarga Conjunto de Datos 41
4.2. Procesamiento de Datos 42
4.2.1. Procesar campo “Category” 42
4.2.2. Procesar campo “sport” 42
4.3. Mapeo del conjunto de datos 42
4.3.1. Mapeo “category” 43
4.3.2. Mapeo “daddr” 44
4.3.3. Mapeo “flgs” 47
4.3.4. Mapeo “saddr” 48

4.3.5. Mapeo “proto” 49

4.3.6. Mapeo “state” 49

4.4. Normalizar conjunto de datos 50

4.5. Creación red neuronal MLP 50

4.6. Muestreo de datos 51

5. CONCLUSIONES Y RECOMENDACIONES 51

5.1. Conclusiones 51

5.2. Recomendaciones 52

6. Referencias 53

Anexo 59

ÌNDICE DE FIGURAS

Figura 1. Estructura perceptrón simple 6

Figura 2. Estructura Perceptrón Multicapa 7

Figura 3. Modelo Cliente-Servidor de un Botnet..... 9

Figura 4. Modelo Peer-to-Peer de un Botnet..... 10

Figura 5. Arquitectura IoT Cloud. 11

Figura 6. Modelos de Cloud Computing. 12

Figura 7. Tipos de Cloud 13

Figura 8. Mapa Conceptual Redes Neuronales..... 18

Figura 9. Arquitectura red Neuronal Perceptrón Multicapa(MLP) 19

Figura 10.Arquitectura red neuronal BAM 21

Figura 11.Arquitectura de red neuronal LVQ..... 24

Figura 12. Arquitectura de red neuronal SOM..... 26

Figura 13.Arquitectura Cascada-Correlación 29

Figura 14. Entorno del banco de pruebas del conjunto de datos BoT-IoT 32

Figura 15. Simulación IoT con Red-Node. 34

Figura 16. 5% del conjunto de datos 59

Figura 17.Conjunto de datos 59

Figura 18. Código para “Category” 60

Figura 19.Código para el campo “sport” 60

Figura 20.Código mapeo “daddr” 60

Figura 21.Código mapeo “flgs” 61

Figura 22.Código mapeo “saddr” 61

Figura 23.Código mapeo “proto” 61

Figura 24.Código mapeo “state” 62

Figura 25.Código de normalización 62

Figura 26.Código creación MLP 62

Figura 27.Matriz de Confusión 63

Figura 28.reporte de conjunto de datos 63

ÍNDICE DE TABLAS

Tabla 1. Clasificación de tipos de bases de datos..... 14

Tabla 2. Comparación redes Neuronales..... 30

Tabla 3.Estadisticas de Trafico del Conjunto de datos..... 35

Tabla 4. Estadística de ataques en el conjunto de datos..... 36

Tabla 5.Comparacion de conjuntos de datos..... 37

Tabla 6.Mapeo campo “Category”..... 43

Tabla 7.Mapeo campo “daddr”..... 44

Tabla 8.Mapeo campo “flgs”..... 47

Tabla 9.Mapeo campo “saddr”..... 48

Tabla 10.Mapeo campo “proto”..... 49

Tabla 11.Mapeo campo “state”..... 49

INTRODUCCIÓN

El conjunto de datos BoT-IoT fue creado mediante el diseño de un entorno de red realista llamado Kaita en el Cyber Range Lab del centro de UNSW Canberra Cyber, este entorno incorpora combinaciones de tráfico normal y de botnet, este conjunto de datos consta de tres componentes: plataformas de red, servicios Internet de las cosas (IoT) simulados, extracción de funciones y análisis forenses.

Se diseñó el banco de pruebas en el laboratorio de investigación de Cyber Gama de UNSW Canberra, en la cual se aplicaron varios escenarios para poder probar el conjunto de datos, una estación meteorológica (Tema: / Smarthome / WeatherStation), que genera información sobre la presión del aire, humedad y temperatura, un frigorífico inteligente (/ Smarthome / nevera), que mide la temperatura del frigorífico, movimiento activado luces (/ Smarthome / motionLights), las cuales se activan o apagan basado en una señal generada aleatoriamente, una puerta de garaje activada a distancia (/ Smarthome / cochera), que se abre o se cierra, basado en una entrada probabilística, un termostato inteligente (/ Smarthome / termostato), que regula la temperatura, activando el sistema de aire acondicionado.

Este conjunto de datos se lo implemento en las plataformas de red que incluyen máquinas normales y virtuales (VM) con dispositivos de red adicionales, tales como un firewall y taps, se lo implemento con servicios Internet de las cosas (IoT) simulados, que contienen algunos servicios de Internet de las cosas como una estación meteorológica, estas fueron simuladas a través de la herramienta de Nodo-rojo. También se lo implemento, en extracción de funciones y análisis forenses, donde se utilizó la herramienta Argus con el fin de extracción de datos, técnicas de Machine Learning con el fin de evaluar los vectores de características para discriminar los casos normales y anormales.

Por medio de esta se pudo constatar el funcionamiento de este conjunto de datos, se lo realizo ya que en los últimos tiempos se ha aumentado la exposición de los datos tanto personales como empresariales, ya que estos datos son objetivos para

piratas informáticos, la UNSW creo este conjunto de datos para demostrar las técnicas de detección de botnets en las redes.

Machine Learning es una disciplina científica del ámbito de la inteligencia artificial que tiene como objetivo crear sistemas que aprenden automáticamente, también en este contexto implica que estos sistemas se mejoran de forma autónoma con el tiempo, sin intervención humana.

EL perceptrón multicapa (MLP) se basa en otra red más simple llamada perceptrón simple solo que el número de capas ocultas puede ser mayor o igual que una, el perceptrón multicapa también es una red unidireccional.

AIcance

El alcance de este proyecto de titulación es Desarrollar una red neuronal MLP para la detección de botnets en el conjunto de datos UNSW-BoT-IoT.

Esta solución será desarrollada para analizar el tráfico dentro de una red para así detectar posibles ataques maliciosos dentro de la red mediante el conjunto de datos en una red neuronal MLP (Perceptrón multicapa). En cuanto a la programación será realizada mediante el lenguaje de programación Python3.0 y la librería Keras en el entorno de desarrollo integrado pycharm 2018, lo cual basaremos la búsqueda a ataques botnets. así determinar el desempeño de dicha red neuronal en el conjunto de datos USNW-BoT-IoT.

Justificación

En la actualidad es fácil observar el crecimiento de la internet, gracias a los avances tecnológicos dentro del área de telecomunicación y las redes de datos, los usuarios de internet cada vez mueven más aspectos de sus vidas hacia la red global, confiando que sus datos estén protegidos, por lo cual esta investigación se la realiza con el propósito de aportar con el conocimiento existente sobre el uso de modelos de Matchine Learning para la detección de intrusos en una red de datos, mediante el estudio de la red neuronal MLP y así en base a los resultados de esta red ayudar a futuros investigadores en el desarrollo de mejores e innovadoras herramientas que protejan la seguridad de los sistemas informáticos.

Objetivo General

Desarrollar una red neuronal MLP para la detección de botnets en el conjunto de datos UNSW-BoT-IoT

Objetivos Específicos:

1. Revisar modelos de redes neuronales de implementación para seleccionar el mejor modelo viable para la detección de botnets en el conjunto de datos UNSW-BoT-IoT
2. Procesar el conjunto de datos USNW-BoT-IoT mediante el lenguaje de programación python y la herramienta scapy.
3. Implementar el modelo MLP mediante el lenguaje de programación python, para su posterior entrenamiento.

1. CAPÍTULO I: MARCO TEÓRICO

En este capítulo, se detallará conceptos básicos para explicar brevemente tanto las herramientas y aplicaciones que se van a utilizar para el desarrollo de la tesis nombrando características esenciales para su uso.

1.1. Machine Learning

El machine learning es un aprendizaje automático que se encuentra en las ciencias informáticas la cual va de la mano con la inteligencia artificial, su único objetivo es crear o desarrollar sistemas que tenga la capacidad de aprender por sí mismos, esta tecnología se dio para automatizar varias operaciones para así reducir la interacción del ser humano. (APD, 2019)

Lo que se refiere con aprendizaje es a identificar series de patrones propuestos por grandes cantidades de parámetros, es decir que la maquina no aprende por si sola si no por medios de algoritmos de programación que llegan a modificarse con la persistente entrada de datos, de esa manera puede predecir acciones futuras o tomar decisiones de manera autónoma.

Las computadoras se programan así mismas por medio del machine learning, llegando hasta cierto punto al usar los algoritmos, ya obtienen propios cálculos según los datos que se han recopilado en el sistema y mientras más datos se recopilen aumenta la precisión de las acciones resultantes. (APD, 2019)

1.2. Red Neuronal Artificial

Una red neuronal artificial es un esquema de computación distribuida inspirada en la estructura del sistema nervioso de los seres humanos. La forma más básica de una red neuronal es conectando varios procesadores fundamentales, realizando así un sistema adaptativo que por medio de un algoritmo puede ajustar varios parámetros libres (pesos sinápticos), para poder alcanzar los requerimientos de rendimiento de un problema que se ha presentado. La red neuronal puede ajustar

los parámetros libres por medio del entrenamiento o aprendizaje, y la técnica para llegar a esto se lo conoce como algoritmo de aprendizaje o algoritmo de entrenamiento. (Salas, 2004)

Las redes neuronales artificiales tienen la capacidad de aprender comenzado por un conjunto de datos o patrones, lo que significa que tiene la posibilidad de encontrar un modelo que se asemeje a los datos. Este proceso de entrenamiento puede ser supervisado o no supervisado. (Salas, 2004)

El aprendizaje supervisado se basa en entrenar la red con un conjunto de datos o patrones supervisados, a lo que se refiere a un conjunto de datos de entrada con sus respectivas salidas así la red neuronal ajustara los parámetros de la red neuronal artificial consiguiendo que la red neuronal se ajuste a los datos de salida generada para que sean similares a los datos de salida de una entrada. Se lo llama supervisada ya que se conoce tanto los datos de entrada como los de salida.

El aprendizaje no supervisado, este da a la red neuronal artificial solo un conjunto de datos de entrada, siendo así que la red neuronal artificial tiene que ajustar varias veces los pesos de la red hasta que encuentre alguna forma de los datos. (Salas, 2004)

1.3. Modelo Neuronal Perceptrón

Un perceptrón es un algoritmo simple, el cual ayuda a dividir un conjunto de datos de entrada en dos diferentes partes si, y no, se desarrolló según la neurona humana, su característica principal es que tiene la capacidad de aprender para resolver problemas complejos. (missinglink, 2020)

Esta se puede combinar con varios perceptrones para así poder formar una nueva red neuronal artificial y así poder dar respuesta a cualquier pregunta dependiendo de la información de entrenamiento del perceptrón. En la Figura1 se puede observar la estructura de una capa de entrada y una de salida de un perceptrón simple.

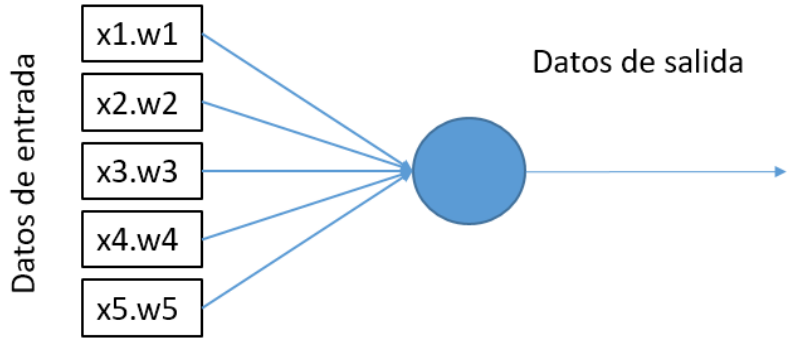


FIGURA 1. Estructura perceptrón simple

Tomado de (missinglink, 2020)

1.3.1. Modelo Neuronal Perceptrón Multicapa

El perceptrón multicapa (MLP) es un modelo de red neuronal que se alimenta directamente, este se caracteriza por tener una capa de entrada, varias capas intermedias también llamadas capas ocultas y una capa de salida, estas están conectadas directamente entre capa y capa.

Utiliza la propagación hacia a tras también llamada delta, así llega a entrenarse el perceptrón multicapa, cabe recalcar que estas estructuras de las capas nos ayudan que no existan ciclos dentro del modelo ya que los datos tienen que tener semejanzas de la capa anterior por lo cual todos los nodos en una capa tienen datos para todos los nodos en la capa anterior. (Torfi, 2020). La capa de entrada es la que conecta con la red exterior, cada neurona se une con cada una de las variables que tiene la red externa, la capa oculta es formada por las neuronas cuyas entradas vienen de la capa de entrada y pasan a las neuronas de la capa siguiente, la capa de salida es la encargada de conectar las redes neuronales ocultas con la capa de salida. En la Figura2 se puede observar la estructura con una capa de entrada, una capa oculta y una de salida de un perceptrón multicapa.

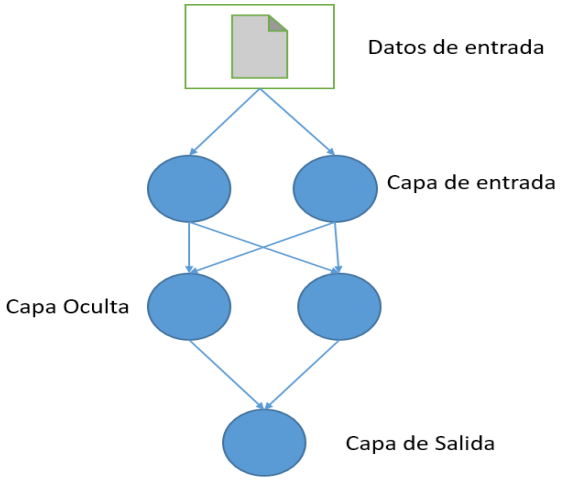


Figura 2. Estructura Perceptrón Multicapa

Tomado de (missinglink, 2020)

1.4. Seguridad de la Información

La seguridad de la información es proteger tanto la información como los sistemas de información a las cuales se tengan acceso, previniendo así su uso, divulgación, alteración, modificaciones, eliminaciones no autorizadas. (Soriano, 2014)

Este es el nombre básico que se le otorga a las diferentes herramientas creadas con el fin de proteger los datos en un equipo y así evitar ataques informáticos. (Universidad de Valencia, 2018)

1.5. Ataques Informáticos

Un ataque informático es aprovechar las debilidades o vulnerabilidades, ya estos se encuentren dentro del software, hardware o también del personal que forma parte de los que manejan el campo informático. Su objetivo es obtener beneficios generalmente económicos así rompiendo las seguridades de los sistemas que luego se ven afectados a los bienes de las organizaciones. (Mieres, 2009)

1.5.1. Botnets

Un botnet es la agrupación de varios host que se encuentran conectados a la internet, interactuando entre ellos para así cumplir una tarea distribuida, este término botnet se lo utiliza más para actos ilícitos ya que al igual que un conjunto de host pueden estar agrupados para un trabajo, estos sistemas están compuestas por los host que se encuentran infectados controlados sin el permiso de sus dueños, estos host infectados se los conoce comúnmente como drones o zombies, estos aplican un software malicioso que se lo conoce como bot.

Estos botnets pueden realizar diferentes ataques como es el CLick Fraud el cual consiste que el software del bot puede visitar páginas web dando clic automáticamente a anuncios, también se los utiliza comúnmente para el keylogging que esta es la mayor amenaza en la privacidad individual, sabiendo así la actividad que realiza el usuario en el teclado teniendo por medio de esto acceso a la información personal del usuario. (Agncia de Regularizacion y Control de las Telecomunicaciones , sf). Principalmente estos se los utiliza para poder enviar spam o virus, robar información personal o realizar ataques de denegación de servicios DDoS. (avast, 2020)

1.5.1.1. Modelo Cliente-servidor

El modelo cliente-servidor es una manera antigua en que los bots o zombies podían recibir instrucciones desde una sola ubicación, que casi siempre solía ser una página web o un servidor compartido, la eliminación de un botnet por medio de este modelo tiene una forma sencilla de hacerlo ya que, al eliminar el sitio web o el servidor, el sistema se desmorona. (AVG Internet Security, 2019), En la Figura3 se puede observar la forma de infección que realiza el modelo cliente-servidor de un botnet, se puede observar que los equipos de color naranja se encuentran infectados por el botmaster al enviar y recibir información.

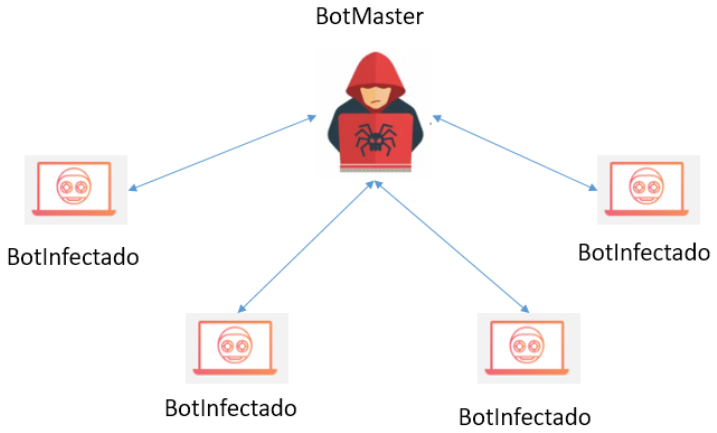


Figura 3. Modelo Cliente-Servidor de un Botnet.

Tomado de (AVG Internet Security, 2019)

1.5.1.2. Modelo peer-to-peer

El modelo peer-to-peer cada equipo que se encuentra infectado se comunica directamente con otro equipo en la red y este se encuentra conectado a otros equipos, y este se encuentra conectado a otros equipos realizando así una red de computadoras infectadas, hasta que todo el sistema se encuentra unido, al eliminar cualquier equipo aún no se eliminara el botnet ya que los demás siguen conectados e infectando a otros más. (AVG Internet Security, 2019). En la Figura4 se puede observar la forma de infección que realiza el modelo Peer to peer de un botnet, siendo los equipos de color naranja los equipos infectados.

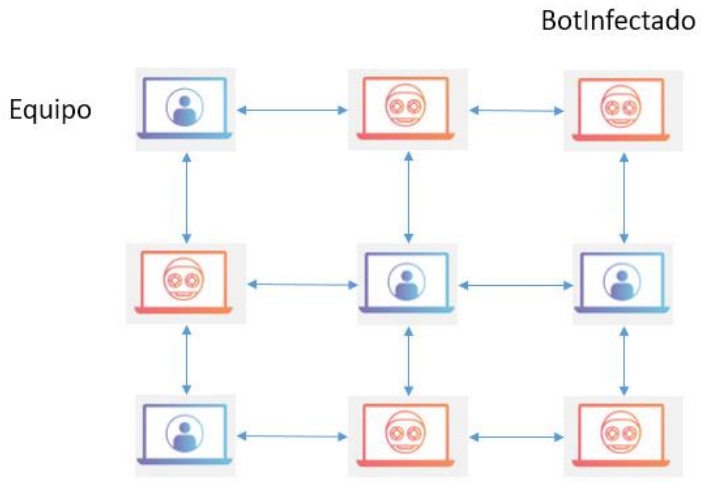


Figura 4. Modelo Peer-to-Peer de un Botnet.

Tomado de (AVG Internet Security, 2019)

1.5.2. Ataques de red distribuidos

EL ataque de red distribuido (DDos), es un ataque que aprovecha el límite de los recursos de red, enviando muchas solicitudes a los recursos web atacado así desbordando la capacidad del sitio web siendo así la caída de la página o que esta no funcione de manera correcta. (Kaspersky, 2020)

1.6. Internet de las Cosas

El internet de las cosas(IoT) es la conexión por medio del internet de dispositivos informáticos que se encuentran añadidos en objetos cotidianos, el cual los permite recibir y enviar datos, por lo cual ayuda a controlar dispositivos inteligentes de manera remota. (Peña, 2019)

1.6.1. IoT cloud.

El IoT cloud es un grupo de herramientas que permite conectar, procesar, almacenar y analizar datos de forma local y también por medio de la nube, su

principal característica es que tiene un conjunto de software integrado con funciones de aprendizaje autónomo para los recursos que se encuentran locales o del perímetro. (Google cloud, s.f). En la Figura 5 podemos observar la arquitectura que maneja el IoT Cloud, desde los dispositivos IoT hacia la nube.

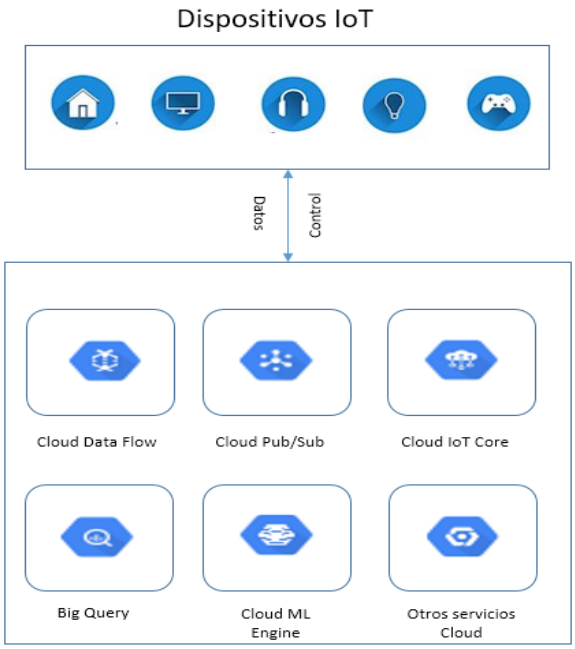


Figura 5. Arquitectura IoT Cloud.

Tomado de (Google cloud, s.f).

1.7. Cloud Computing

El cloud computing es la distribución de tecnologías de la información que se encuentran en demanda por medio del internet, manejando un esquema de pago por uso, esto ahorra al usuario a adquirir, mantener y poseer servidores y centros de datos físicos, teniendo así acceso a servicios tecnológicos, como almacenamiento, bases de datos, utilizándolos basándose en sus necesidades a través de un proveedor de la nube que da estos servicios. (Amazon Web Services,

2020). En la Figura5 podemos observar los diferentes tipos de servicios que ofrece el cloud a los usuarios.

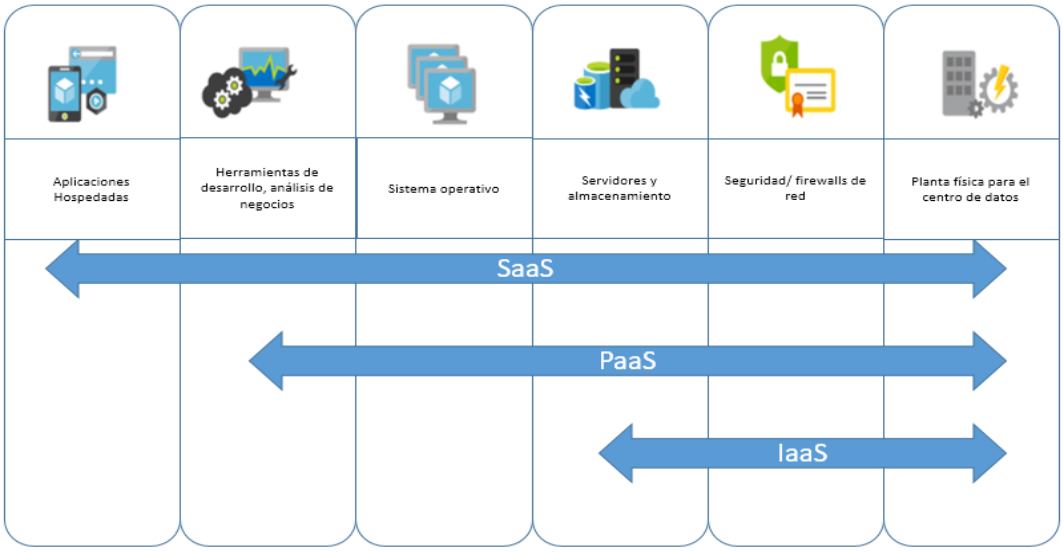


Figura 6. Modelos de Cloud Computing.

Tomado de (Microsoft Azure, 2019)

1.7.1. Cloud Publica

En el cloud público el proveedor de servicio ofrece sus recursos de forma pública a cualquier usuario, ya puede ser una pequeña empresa o una empresa grande, estos recursos pueden ser gratuitos o de pago, si fuera de pago seria que los usuarios lo único que pagarían sería el ancho de banda. (EINATEC, s.f)

1.7.2. Cloud Privada

En el cloud privado el proveedor de servicio ofrece sus recursos a los usuarios que lo pidan, estos servicios se los puede dar por medio de redes privadas o internet, como su nombre lo indica este tipo de servicio no está disponible para todo le público, siendo así que la infraestructura se encuentra de manera local. (EINATEC, s.f)

1.7.3. Cloud Hibrida

El cloud hibrido combina el cloud público y el cloud privado, así permitiendo a las compañías utilizar los servicios de cloud privada basándose en su escalabilidad por medio del cloud público, utilizando un entorno informático combinado para que las aplicaciones y datos puedan compartirse. (EINATEC, s.f)

En la Figura6 se puede observar las diferencias entre los tres tipos de cloud que existen.

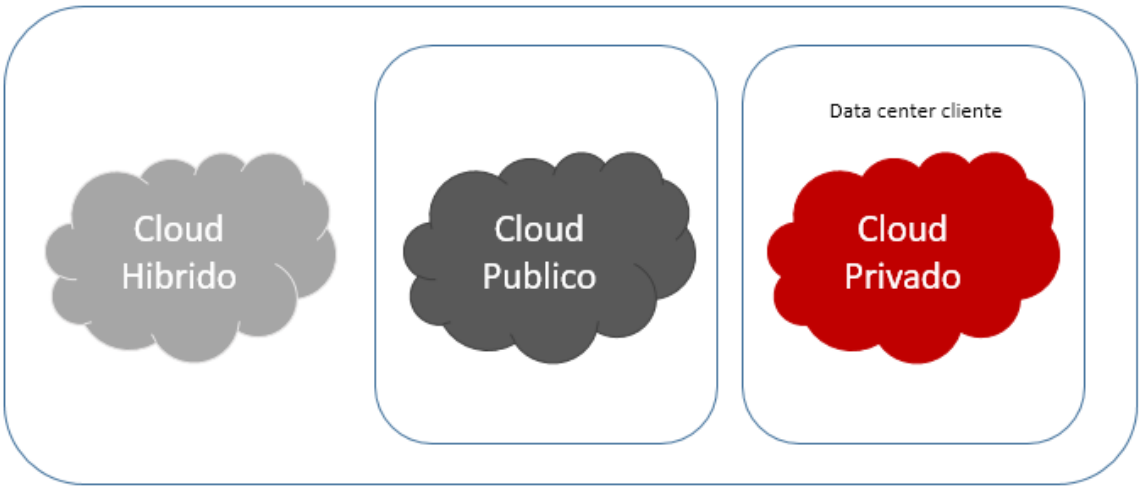


Figura 7. Tipos de Cloud

Tomado de (COPIMAR, s.f)

1.7.4. Infraestructura como servicio

La infraestructura como servicio(Iaas), esta permite acceder a la conexión en red, a los equipos ya sean virtuales o en software dedicado, también permite acceder al espacio de almacenamiento de datos, su característica principal es que ofrece una flexibilidad alta y tiene control de la administración a los recursos de tecnologías de información. (Amazon Web Services, 2020)

1.7.5. Plataforma como servicio

Las Plataformas como servicio(Paas) este servicio permite al usuario eliminar la necesidad de administrar infraestructura como hardware, sistemas operativos, así permitiéndole trabajar arduamente en la implementación y administración de las aplicaciones, contribuyendo con la eficacia del usuario dejando de lado la preocupación de la planificación de capacidad, mantenimiento y varias tareas que lleva la aplicación para ejecutarse. (Amazon Web Services, 2020)

1.7.6. Software como servicio

El software como servicio(SaaS) permite al usuario poder conectarse a aplicaciones basadas en la nube por medio del internet, ofreciendo soluciones de software integrales utilizando un modelo de pago por uso, así ayudando al usuario a tener portabilidad por medio de SaaS. (Microsoft Azure, 2019)

1.8. Bases de Datos

Una base de datos es la que almacena datos conectándola con su unidad lógica junto a los metadatos que se necesiten para poder así realizar su procesamiento, es una utilidad que ayuda al usuario para gestionar gran cantidad de ficheros y poder consultar la información. (IONOS, s.f). En la Tabla No.1 se puede observar la clasificación de los tipos de bases de datos utilizados en los sistemas de gestión de base de datos.

Tabla 1. Clasificación de tipos de bases de datos

Base de datos	Definición	Aplicaciones	Marcas

Jerárquico	Almacenan su información en una estructura jerárquica.	Bancos, Compañías, Sistemas de Bancos	IMS/DB
Relacional	Esta organizadas en forma de tabla.	Facturación, Inventarios	MySQL,Oracle,SQLite
Orientado a Objetos	Almacena en la base de datos los objetos completos tanto estaco como comportamiento.	Inventarios	Db4o

Tomado de (IONOS, s.f)

1.8.1. Sistema de gestión de base de datos

El sistema de gestión de base de datos es el responsable de administrar los datos que se encuentran dentro de una base de datos, determinando la estructura, orden, accesos, dependencias, etc. Existen varios sistemas de gestión como por ejemplo Oracle Database, Microsoft SQl Server, MySQL, los cuales el usuario podrá elegir para administrar su base de datos dependiendo de la necesidad que este tenga, ya

que los diferentes sistemas tienen cada uno sus ventajas y desventajas correspondientemente. (IONOS, s.f)

1.9. Phyton

Python es un lenguaje de programación multidiagrama, ya que soporta la programación orientada a objeto, siendo su principal característica que es de código abierto, se lo puede utilizar en varias plataformas y sistemas operativos, como por ejemplo Windows, Mac OS, Linux, Python se ha desarrollado para la creación de aplicaciones científicas, comunicaciones de red, aplicaciones con una interfaz gráfica para el usuario, para crear juegos y también para aplicaciones web. (Montoro, 2012)

1.9.1. Scapy

Es una herramienta escrita en el lenguaje de programación Python que su funcionamiento es crear o manipular paquetes, escanear, puede hasta crear graficas en diferentes formatos desde 2d hasta pdf, esta herramienta puede ser incluida como librería en los proyectos utilizados en Python, esta herramienta está disponible para varias plataformas como para Linux, Mac OS, cabe recalcar que para utilizar esta herramienta se tiene que tener permisos de administrador o de superusuario ya que para utilizar esta herramienta se tiene que tener el control de las interfaces de red, esta herramienta nos permite testear las seguridades de las aplicaciones que se desarrollan, de los protocolos utilizados. (Campos, 2012)

1.9.2. NumPy

NumPy es una herramienta que se utiliza principalmente en el machine learning, ya que esta herramienta proporciona una estructura de datos de tipo matriz, esta es una herramienta de Python que nos permite realizar matrices simples o multidimensionales, esta estructura garantiza la realización de cálculos con un alto nivel de eficiencia. (González, 2018)

1.9.3. TensorFlow

TensorFlow es una herramienta que se ha dirigido para el machine learning, esta herramienta fue creada por google para así poder construir y entrenar redes neuronales, esta herramienta actualmente se la está utilizando para la producción de productos google, esta herramienta también está disponible para diferentes plataformas como Linux, MAC OS. (Delgado, 2017)

TensorFlow utiliza una unidad de procesamiento del tensor llamada TPU, este es un acelerador de inteligencia artificial que se lo puede programar y está orientado para correr modelos de redes neuronales para así poder entrenarlos. (Delgado, 2017)

2. CAPITULO II. MODELOS DE REDES NEURONALES

En este capítulo, se detallará diferentes modelos de redes neuronales, con sus respectivos métodos de aprendizaje, con el fin de realizar una comparativa entre los modelos y el modelo MLP verificando sus características. En el mapa conceptual detallado en la Figura 8 se explica las redes neuronales, sus divisiones, sus características.

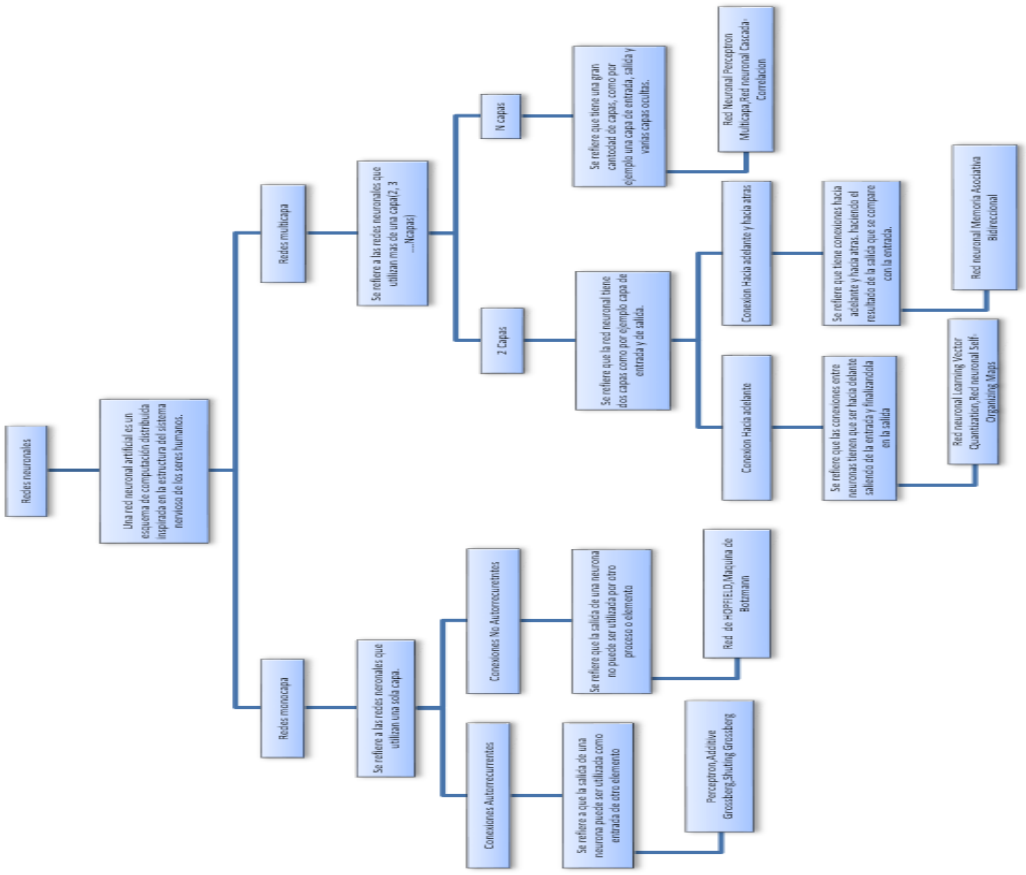


Figura 8. Mapa Conceptual Redes Neuronales

2.1. Red Neuronal Perceptrón Multicapa

El perceptrón multicapa (MLP) es un modelo de red neuronal que se alimenta directamente, este se caracteriza por tener una capa de entrada, varias capas intermedias también llamadas capas ocultas y una capa de salida, estas están conectadas directamente entre capa y capa. (Torfi, 2020)

Esta funciona con un método de aprendizaje back-propagation por lo que así tiene la capacidad de contener por lo menos una capa con unidades no lineales para que así pueda aproximar funciones o relaciones continuamente en los datos de entrada

y salida, por lo cual esta forma de aprendizaje convierte a este modelo en una herramienta flexible y no lineal.

2.1.1. Arquitectura Red Neuronal Perceptrón Multicapa

La red Neuronal Perceptrón Multicapa está formado por una capa de entrada, una capa de salida, y una o más capas ocultas, como se puede observar en la Figura 7, indica la arquitectura del modelo formado por una capa de entrada, una capa oculta y una de salida.

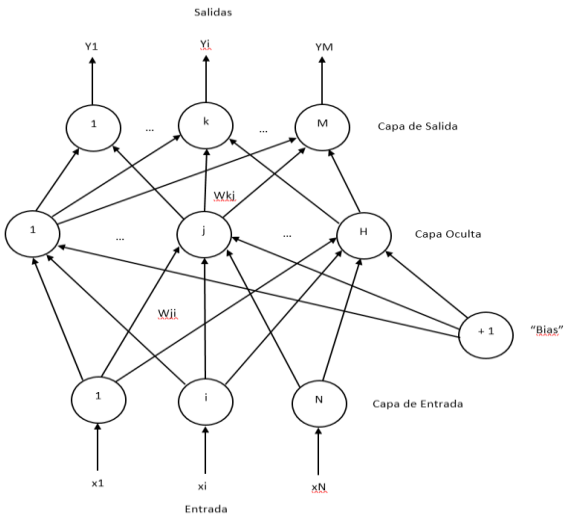


Figura 9. Arquitectura red Neuronal Perceptrón Multicapa(MLP)

Tomado de (Navarro, s.f)

Las conexiones entre las capas tienen que ser siempre hacia delante, así pasando hacia la siguiente capa, en este modelo no existen conexiones laterales ni conexiones hacia atrás, por lo cual la información va desde la capa de entrada hacia la capa de salida.

Siendo “Wij” el peso de conexión entre la neurona de entrada “i” y la neurona oculta “j”, y “Wkj” tiene el peso de conexión entre la neurona oculta “j” con la neurona de salida “k”.

2.1.2. Algoritmo de Propagación hacia atrás

Esta etapa de funcionamiento donde ante la red de entrada se presenta un patrón de entrada el cual se transmite por medio de las diferentes capas de neuronas hasta llegar a una salida, después una etapa de entrenamiento donde se irán modificando los pesos de la red de tal manera que se llegue a coincidir con la salida que desea el usuario y la salida que obtuvo la red. (Marin, uc3m, s.f)

2.1.3. Aplicación de un perceptrón multicapa

El perceptrón multicapa intenta resolver dos tipos de problemas, como principal es el problema de predicción los cuales se basan en una estimación de una variable de salida partiendo de un conjunto de datos de entrada, también otro problema son los de clasificación en los cuales asignan una categoría a ciertos patrones partiendo de un conjunto de datos de entrada. (Marin, uc3m, s.f)

2.1.4. Selección de variables, procesamientos en un perceptrón multicapa

Se tiene que elegir de manera cuidadosa las variables a ingresar a la red ya que deben ser las que realmente predigan la variable de salida, pero a la ves que no tengan relación con otras variables ya que esto podría realizar un reajuste innecesario a la red.

Las variables escogidas deben seguir una distribución uniforme siendo así los valores aproximados a un mismo valor e inmerso en el intervalo de trabajo de la función empleada en las capas tanto ocultas como la de salida de la red neuronal, así los valores de entrada y de salida suelen estar en valores dentro del rango comprendido entre 0 y 1 o entre -1 y 1. (Marin, uc3m, s.f)

2.2. Red Neuronal Memoria Asociativa Bidireccional

La red neuronal Memoria Asociativa Bidireccional (BAM) tiene una arquitectura de dos capas siendo la capa de entrada y la capa de salida que se encuentran

conectadas simétricamente, el aprendizaje de este modelo lo lleva de forma supervisada si es que se conocen las variables.

Este modelo nos permite trabajar toda clase de operaciones que tienen que ver con asociaciones de variables en el espacio o en el tiempo operando con variables binarias para así simular de mejor manera las excitaciones “+1” e inhibiciones “-1” de las redes neuronales. (Perez, 1994)

2.2.1. Arquitectura de la red neuronal Memoria Asociativa Bidireccional

La red Neuronal Memoria Asociativa Bidireccional (BAM) está formado por una capa de entrada y una capa de salida, siendo estas necesariamente conexiones simétricas, como se puede observar en la Figura 8, indica la arquitectura del modelo formado por una capa de entrada, y una de salida.

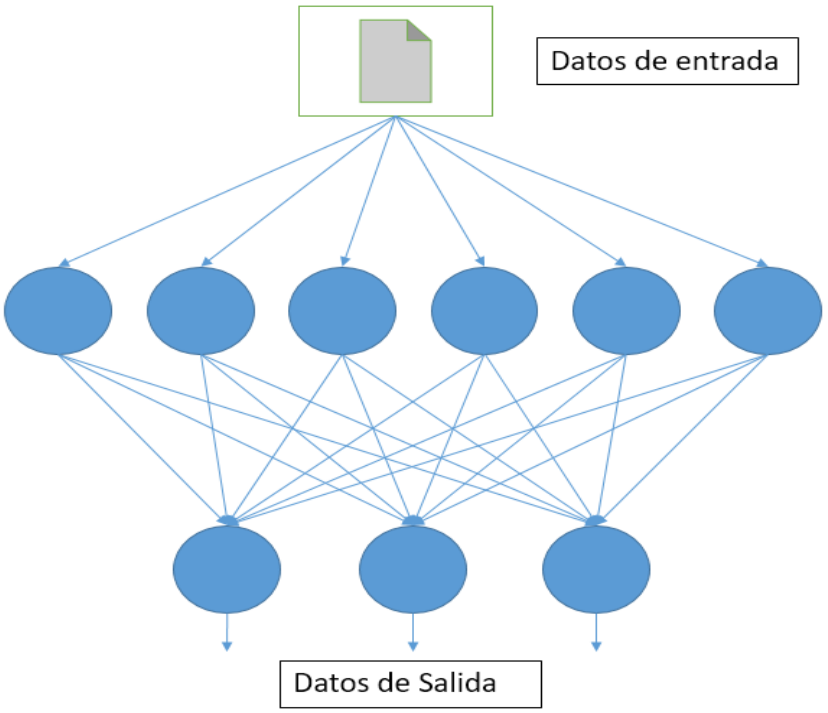


Figura 10.Arquitectura red neuronal BAM

Tomado de (Perez, 1994)

2.2.2. Memoria Heteroasociativa

La memoria Heteroasociativa está dada por $\phi(A_i) = B_i$, si existiese una entrada A cercana en distancia a otra entrada que se encuentra en A_i, entonces $\phi(A) = B_i$, este tipo de memoria es la que utiliza la red neuronal BAM, y se caracteriza ya que es resistente al ruido, por lo cual esta memoria se la utiliza en aplicaciones que desean filtrar una señal que contiene ruido. (Perez, 1994)

La letra ϕ significa una función que se da a la red para mapeo o de asociación

2.2.3. Aprendizaje de una red neuronal Memoria Asociativa Bidireccional

El aprendizaje de este modelo se forma sumando los productos externos o matrices entre las variables de salida y los opuestos de las variables de entrada formando así la memoria de la red neuronal BAM.

Las variables de entrada tendrán una dimensión n las cuales serán el número de neuronas que se encuentran en la capa de entrada y las variables de salida tendrán una dimensión m las cuales serán el número de neuronas que se encuentran en la capa de salida. Como por ejemplo teniendo un conjunto de patrones (A_i, B_i), por medio de estos patrones la memoria M se construye de la siguiente manera. (Perez, 1994)

$$M = B_1^t A_1 + B_2^t A_2 + .. + B_n^t A_n$$

Y para recalcular un nuevo vector A se tiene que multiplicar la memoria M por el vector de entrada A como se indica en la ecuación. (Perez, 1994)

$$\phi(A) = (B_1^t A_1 + B_2^t A_2 + .. + B_n^t A_n)A$$

2.3. Red Neuronal Learning Vector Quantization

La red Neuronal Learning Vector Quantization(LVQ) se la creo para poder formar mapas de características similares a las que realizamos en el cerebro, por lo que hay neuronas que se organizan en muchas zonas llegando a que las informaciones

captadas del entorno a través de los órganos sensoriales se representan en forma de mapas bidimensionales. Formado así mapas topológicos para poder establecer características comunes entre la información de entrada. (SAEM THALES, s.f)

2.3.1. Arquitectura de una red Neuronal Learning Vector Quantization

En la red neuronal Learning Vector Quantization (LVQ) podemos encontrar dos capas con N neuronas de entrada y M neuronas de salida, conectándose las N neuronas de entrada con las M neuronas de salida por medio de conexiones hacia a delante.

Una característica principal que tiene esta arquitectura es que en la capa de salida existen conexiones laterales las cuales tienen un peso negativo, cada neurona que se encuentre conectada lateralmente va a tener influencia con las neuronas conectadas lateralmente. (SAEM THALES, s.f)

El valor que se van a asignar a los pesos de las conexiones entre las capas de salida y, de entrada, al momento de aprendizaje de la red neuronal va a depender de las conexiones laterales de las neuronas, siendo así el valor dependerá de la distancia entre las neuronas ya que si se encuentra más lejos sería menor el valor, si se encuentra más cerca sería más alto. (SAEM THALES, s.f). En la Figura 9 podemos observar la arquitectura de la red neuronal LVQ con una capa de entrada y una capa de salida con sus respectivas conexiones laterales en la capa de salida.

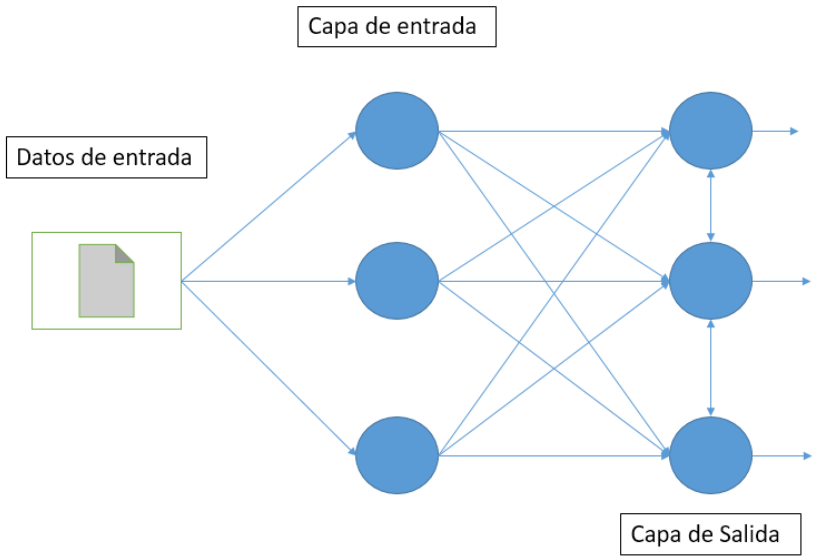


Figura 11.Arquitectura de red neuronal LVQ.

Tomado de (SAEM THALES, s.f)

2.3.2. Aprendizaje de una red neuronal Learning Vector Quantization

Este modelo de red neuronal es de tipo off line lo que quiere decir es que distingue una etapa de funcionamiento y otra etapa de aprendizaje, también suele utilizar el aprendizaje no supervisado. (EINATEC, s.f)

Solo una neurona de la capa de salida se activa correspondientemente con una neurona de la capa de entrada, así va a ajustarse los pesos de las conexiones en función de la neurona que tiene el valor más parecido a la de entrada llamándola neurona vencedora. (EINATEC, s.f)

En la etapa de aprendizaje p entrenamiento, se da a la red neuronal un conjunto de datos de entrada, y en función de la similitud de estos datos, las diferentes categorías que serán una por neurona de salida, estas servirán para el funcionamiento realizando las clasificaciones de los nuevos datos que se presentarán en la red, los valores finales de las conexiones entre la capa de salida

y entrada se corresponderán con los componentes del vector de aprendizaje que se utilice en las neuronas. (EINATEC, s.f)

Se tiene que volver a hacer el proceso de aprendizaje varias veces para que así el mapa topológico de salida sea más óptimo, de tal manera que si se presentan varias veces los datos se reducirán las zonas de las neuronas consiguiendo que la red realice una clasificación más selectiva. (EINATEC, s.f)

2.3.2.1. Algoritmo de aprendizaje de una red neuronal Learning Vector Quantization

El objetivo del algoritmo es identificar los valores de los pesos de las conexiones. Sea N el número de neuronas de entrada y M el de neuronas de salida, como primer paso se tiene que W_{ji} ponerlo con valores pequeños aleatorios, fijando así la zona inicial de conexión con las neuronas de salida. Se tiene que dar un conjunto de datos a la capa de entrada en una forma de vector que sería la siguiente $E_k = (e_1, \dots, e_N)$ los cuales deben ser valores continuos, por medio de esta se puede identificar cual será la neurona vencedora de la capa de salida siendo así que los pesos de W_j sean los más similares a los datos de entrada E_k,los componentes W_j son los pesos de conexión entre la neurona j con las neuronas de entrada, por lo cual se tiene que calcular las distancias entre los vectores E_k y W_j para cada neurona de salida de la siguiente manera

$$d_j = Sum_{i=1\dots N}(e_i - W_{ij})^2 \; para \; 1 \leq j \leq M$$

2.4. Red Neuronal Self-Organizings Maps

La red Neuronal Self-Organizings Maps(SOM) se la creo para poder formar una imagen de un espacio multidimensional de entrada a un espacio de salida de menor dimensión, este tipo de red puede ser entrenada por medio de aprendizaje no supervisado y puede producir un mapa. (Fernandez, 2017)

2.4.1. Arquitectura de una red neuronal Self-Organizings Maps

Las redes neuronales Self-Organizings Maps(SOM) están compuestas por dos capas de neuronas, siendo la capa de entrada que se encuentra formada por N neuronas, siendo una por cada variable de entrada cual función es de recibir y transmitir a la capa de salida, La capa de salida que se encuentra formada por M neuronas, estas se organizan de una manera de mapa bidimensional. En la Figura10 se puede observar la capa de entrada y la capa de salida que se encuentra organizada de manera de un mapa bidimensional.

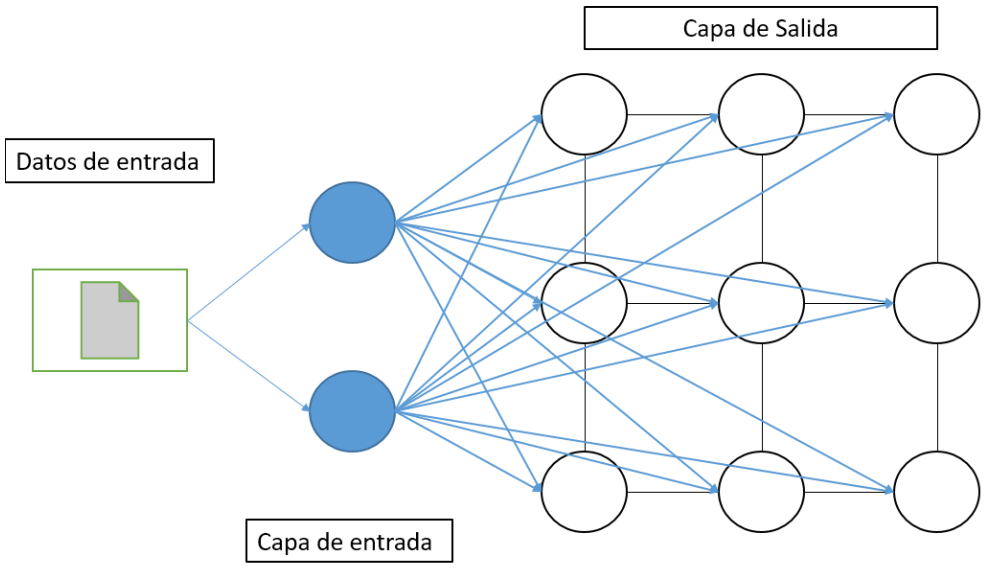


Figura 12. Arquitectura de red neuronal SOM

Tomado de (Marin, us3m, s.f)

Las conexiones en este modelo siempre tienen que ser hacia adelante ya que la información pasa desde la capa de entrada hacia la capa de salida, cada una de las neuronas se encuentran conectadas con cada una de las neuronas de salida mediante un peso W_{ji}. (Marin, us3m, s.f). La red neuronal Self-Organizings Maps (SOM) da una proyección de un espacio de datos en alta dimisión a un mapa bidimensional de neuronas.

2.4.2. Aprendizaje competitivo

Este aprendizaje se caracteriza ya que las neuronas compiten por el derecho a responder a un conjunto de datos de entrada, este aprendizaje tiene varios elementos básicos los cuales aplican a las neuronas, las neuronas tienen un límite impuesto para cada una de ellas, en el conjunto de neuronas en las que son todas iguales excepto las que son agregadas aleatoriamente, esto hace que respondan de diferente manera a los conjuntos de datos de entrada, y el aprendizaje competitivo da a las neuronas un modo de competir para responder a los datos de entra, de esta manera solo una neurona de salida estará activa por grupo a la vez. (Fernandez, 2017)

2.4.3. Algoritmo de una red neuronal Self-Organizings Maps

El objetivo del algoritmo es identificar los valores de los pesos de las conexiones. Sea N el número de neuronas de entrada y M el de neuronas de salida, como primer paso se tiene que W_{ji} ponerlo con valores pequeños aleatorios, fijando así la zona inicial de conexión con las neuronas de salida. Se tiene que dar un conjunto de datos a la capa de entrada en una forma de vector que sería la siguiente $E_k = (e_1, \dots, e_N)$ los cuales deben ser valores continuos, por medio de esta se puede identificar cual será la neurona vencedora de la capa de salida siendo así que los pesos de W_j sean los más similares a los datos de entrada E_k,los componentes W_j son los pesos de conexión entre la neurona j con las neuronas de entrada, por lo cual se tiene que calcular las distancias entre los vectores E_k y W_j para cada neurona de salida de la siguiente manera

$$d_j = Sum_{i=1\dots N}(e_i - W_{ij})^2 \; para \; 1 \leq j \leq M$$

2.5. Red Neuronal Cascada-Correlación

La red neuronal Cascada-Correlación fusiona dos ideas básicas que es el aprendizaje constructivo o de crecimiento de red donde se tiene que agregar una neurona oculta a la vez y estas no se modifican después de haberse agregado la combina con la arquitectura de cascada donde para cada nueva neurona oculta el algoritmo trata de magnificar la relación entre la nueva neurona oculta y el error residual que existe en la red.

2.5.1. Arquitectura de una red neuronal Cascada-Correlación

La arquitectura se basa en la de un perceptrón simple teniendo una capa de entrada y una de salida a su inicio, agregando las neuronas ocultas una a una en la red, para así formar una estructura multicapa, al agregar las nuevas neuronas ocultas estas se conectan con cada una de las neuronas de entrada y a las neuronas de salida que las preceden, así al agregar las nuevas neuronas ocultas los pesos de la entrada se quedan estáticos, mientras que los pesos de salida se entrenaran repetidamente. (Garzón, 2016)

En la Figura11 se puede observar la arquitectura de una red neuronal Cascada-Correlación, en la cual se observará la capa de entrada y la agregación de neuronas ocultas, siendo así los elementos con punteado los que se congelan al momento de agregar nuevas neuronas ocultas.

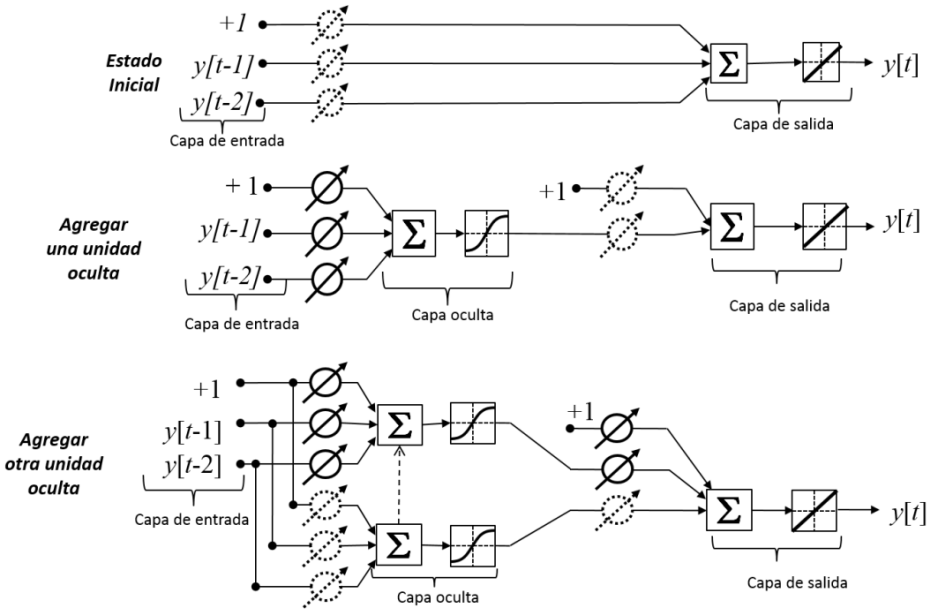


Figura 13.Arquitectura Cascada-Correlación

Tomado de (Garzón, 2016)

2.5.2. Aprendizaje Constructivo

Este tipo de aprendizaje permita una adaptación continua del modelo la cual se basa en los datos que llegan de manera continua, por lo cual este aprendizaje se lo utiliza cuando los sistemas actúan de forma autónoma como en las redes neuronales, este tipo de aprendizaje investiga el cómo aprender en las configuraciones de transmisión de datos permitiéndole así al modelo actualizarse cada vez que lleguen datos de entrenamiento. (Gepperth, 2016)

En la Tabla No.2 se puede observar las diferencias entre las características de cada una de las redes neuronales explicadas.

Tabla 2. Comparación redes Neuronales

Red Neuronal	Retro Propagación	N Capas	Conexión Lateral
Perceptrón Multicapa	ok	Ok	-
Memoria Asociativa Bidireccional	-	-	-
Learnin Vector Quantization	-	-	ok
Self Organizings Map	-	-	ok
Cascada-Correlación	ok	Ok	ok

Tomado de (Ballesteros, s.f)

Se concluye que el modelo de red neuronal Perceptron Multicapa (MLP), es el más utilizado en el Machine Learning, ya que con su estructura con N capas, nos permite ser más asertivos en los resultados deseados a un problema. Se tiene una aceptación en varios campos científicos como es la medicina, ya que se ha comprobado la utilización en varios casos médicos para poder conseguir predicciones de pacientes que padecen con dispepsia dando un resultado alto en sus aciertos en la predicción del caso, siendo así se ha mostrado que la red neuronal perceptron multicapa(MLP) es una de las más utilizadas en todo campo por ello se

la utilizo en el presente proyecto por su alta tasa de efectividad en las predicciones que se desea realizar.

3. CAPÍTULO III: CONJUNTO DE DATOS USNW-BoT-IoT

En este capítulo se detallará como se construyó el conjunto de datos USNW-BoT-IoT abarcando las herramientas que se utilizaron y el proceso realizado para su construcción.

3.1. Redes Internet de las Cosas

Las redes IoT se componen de varios elementos de red típicos los cuales incluyen estaciones de trabajo, laptops, enrutadores y dispositivos IoT, las aplicaciones de los sistemas IoT se los puede encontrar en ciudades inteligentes, IoT industrial, estas llegan a ser complejas ya que necesitan potencia de procesamiento y computación para así permitir que los usuarios puedan controlar los elementos IoT en las redes.

Las vulnerabilidades de las redes IoT han aumentado con los ciberataques, como los Botnets, siendo los botnets redes sincronizadas de equipos comprometidos a los cuales se los denomina como bots, en los botnets, el atacante al cual se lo denomina botmaster, este mantiene un canal de comunicación bidireccional hacia la red atacada, a través de una infraestructura que se llama Comando y Control.

Al usar el comando y control, un botmaster manda comandos a sus botnet y así es capaz de recibir comentarios sobre ataques que se están realizando o datos robados. Una botnet puede lanzar una serie de ataques como son los de denegación de servicios, registro de teclas, phishing, spam, fraude de clics, robo de identidad e incluso puede hacer llegar más malware de bot, los bots infectados están controlados por el botmaster.

3.2. Conjunto de datos BoT-IoT

El banco de pruebas consta de tres componentes, que son plataformas de red, servicios de IoT simulados, funciones de extracción y análisis forense, en las plataformas de red se incluyeron máquinas virtuales y dispositivos de red adicionales como por ejemplo un firewall. (Nickolaos Koroniotis, 2018)

3.2.1. Plataformas de red

Las máquinas virtuales que se prepararon se las trasfirieron a un clúster ESXi configurado y estas se administraron por medio de la plataforma vSphere.

En la Figura 14 podemos observar gráficamente el banco de pruebas del conjunto de datos BoT-IoT.

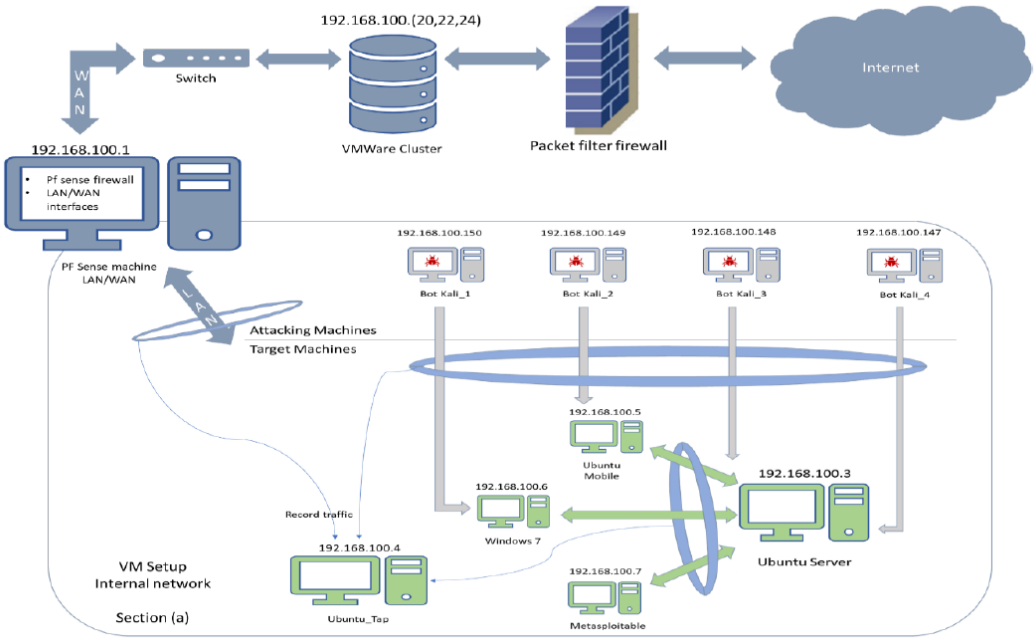


Figura 14. Entorno del banco de pruebas del conjunto de datos BoT-IoT

Tomado de (Nickolaos Koroniotis, 2018)

Las máquinas virtuales se encuentran conectadas a interfaces LAN y WAN en el cluster, se encuentran conectadas al internet por medio de la maquina PFSense.

En la maquina Ubuntu se utilizó la herramienta de Red-Node para poder simular sensores IoT que están conectadas con el centro público IoT de Amazon. Dentro del entorno también se proporcionó un firewall y dos tarjetas de interfaz de red las cuales fueron configuradas en el entorno, una de ellas se la configuro para LAN y la otra para WAN, se agregó el firewall para poder garantizar la validez del etiquetado de los paquetes de las redes entrantes y salientes. (Nickolaos Koroniotis, 2018)

Dentro de la red de máquinas virtuales se utilizó cuatro máquinas que son Kali, Bot Kali 1, Bot Kali 2, Bot Kali 3, Bot Kali 4, un servidor Ubuntu, Ubuntu Mobile, Windows 7, Metasploitable y una maquina Ubuntu Tap, las maquinas Kali que son las que pertenecen a las maquinas atacantes, realizaban escaneos de puertos, DDoS hacia las máquinas virtuales Ubuntu, Windows 7 y Metasploite, dentro de la mauqina virtual Ubuntu server se implementó servicios de DNS, correo electrónico, FTP, HTTP, SSH, también se agregó los servicios simulados de IoT, para así poder imitar sistemas de redes reales. (Nickolaos Koroniotis, 2018)

3.2.2. Servicios IoT simulados

Para poder simulas servicios IoT se utilizó la herramienta de Red-Node el cual funciona como un middleware que permite conectar dispositivos físicos IoT con sus servidores y aplicaciones en la nube, dentro de la herramienta se desarrolló un código JavaScript imitando los sensores IoT como de temperatura, presión y humedad. En la Figura 15 podemos observar la simulación IoT con la herramienta Red-Node que se utilizó en el conjunto de datos.

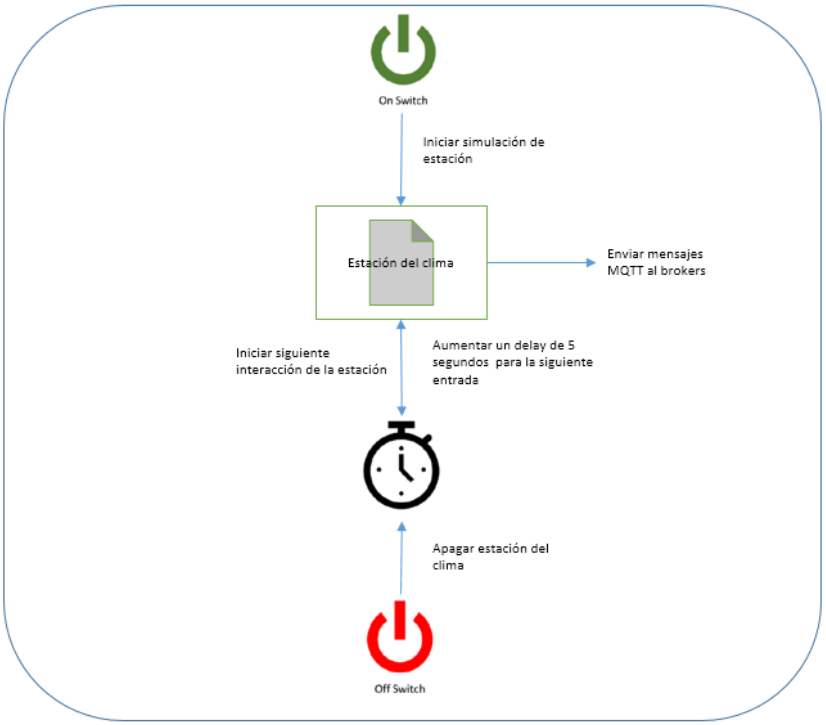


Figura 15. Simulación IoT con Red-Node.

Tomado de (Nickolaos Koroniotis, 2018)

El protocolo MQTT se utilizó como protocolo de comunicación ligero, vinculando las culminaciones de máquina a máquina, aplicando en varios escenarios simulados, estación meteorológica que genera información sobre la presión del aire, humedad y temperatura, un frigorífico inteligente que mide la temperatura del frigorífico, luces que se activan por movimiento las cuales se activan o apagan basado en una señal generada aleatoriamente, una puerta de garaje activada a distancia que se abre o se cierra, basado en una entrada probabilística, un termostato inteligente que regula la temperatura, activando el sistema de aire acondicionado. (Nickolaos Koroniotis, 2018)

Se tomó todos los archivos pcap de las máquinas virtuales, por medio de la herramienta Argus se logró transformar a archivos “.argus” obteniendo solo la

información de flujo de red para así poder subirlo a la base de datos MSQL y transfórmalos en csv para su procesamiento. (Nickolaos Koroniotis, 2018)

3.2.3. Escenarios Benignos

Para la construcción del conjunto de datos BoT-IoT se diseñó una configuración de un hogar inteligente, con cinco dispositivos los que fueron simulados y manejados de manera local, por medio de un botón Node-Red se conectó los dispositivos inteligentes y la infraestructura de nube para así generar tráfico de red normal o también llamada benigno, utilizando la herramienta Ostinato se generó una gran cantidad de tráfico benigno para las máquinas virtuales y los sistemas de producción de red. (Nickolaos Koroniotis, 2018)

La red creada representa una red smarth realista con los cinco dispositivos IoT: Refrigerador Inteligente, Puerta Garaje inteligente, Monitor de Clima Inteligente, Luces Inteligentes, Termostato Inteligente, los mensajes generados por los dispositivos se los transfirió a un proveedor de la nube que es Amazon utilizando un protocolo MQTT. (Nickolaos Koroniotis, 2018). En la tabla 3 podemos observar las estadísticas de trágico normal que se utilizó en el conjunto de datos.

Tabla 3.Estadisticas de Trafico del Conjunto de datos

Protocolo	Numero
UDP	7225
TCP	1750
ARP	468
IPV6-ICMP	88

ICMP	9
IGMP	2
RARP	1
Total	9543

Tomado de (Nickolaos Koroniotis, 2018)

3.2.4. Escenarios Botnet

Se utilizaron varios tipos de ataques de botnets como son los ataques de son los escaneos de puertos, en este ataque se utilizó las herramientas Nmap y Hping3 para así realizar varios tipos de escaneos de puertos, el Nmap se lanzó para escanear servicios y puertos en las máquinas virtuales, y Hping3 para poder realizar una exploración de puertos, también se realizó un ataque de huella digital de sistema operativo en este ataque se utilizó la herramienta Nmap y Xprobe2 para poder lanzar el ataque , la herramienta Nmap se utilizó para poder identificar los sistemas operativos de las máquinas virtuales , la herramienta Xprobe2 se lo utilizo a la par con Nmap para realizar los análisis, utilizando las operaciones básicas de Xprobe2, en los ataques DDoS y DoS se utilizaron la herramienta Hping3, en el robo de datos se utilizó Metasploit para así poder explorar las debilidades en las máquinas virtuales , como por ejemplo en la máquina de Windows 7 se explotó la vulnerabilidad azul SMB, pero para Ubuntu Server se aprovechó las credenciales de administrador, después de la explotación se configuro una conexión tcp invertido por el cual se pudo extraer directorios completos. (Nickolaos Koroniotis, 2018)

En la tabla 4 se puede observar la estadística de los ataques sutilizados en el conjunto de datos IoT-BoT, describiendo la cantidad de cada uno y el total de todos ellos.

Tabla 4. Estadística de ataques en el conjunto de datos

Recopilación de información	**Servicio de Escaneo**		**nmap, hping3**	**1463364**
	OS Fingerprinting		Nmap, xprobe2	358275
Negación de Servicio	DDoS	TCP	hping3	19547603
		UDP	hping3	18965106
		HTTP	Golden-eye	19771
	DoS	TCP	hping3	12315997
		UDP	hping3	20659491
		HTTP	Golden-eye	29706
Robo de información	Keylogging		Metasploit	1469
	Data Theft		Metasploit	118
Total				73360900

Tomado de (Nickolaos Koroniotis, 2018)

3.3. Banco de Pruebas

En la siguiente tabla se puede observar todos los conjuntos de datos que se utilizaron para crear el conjunto de datos BoT-IoT e identificar las características que tiene cada uno de ellos para compararlos.

Tabla 5.Comparacion de conjuntos de datos

Conjunto de datos	Configuración realista	Trafico Realista	Datos Etiquetados	Trazos IoT	Diferentes Escenarios de ataque	Captura completa de paquetes	Características Nuevas
Darpa 98	ok	-	Ok	-	ok	ok	-
KDD99	ok	-	Ok	-	ok	ok	ok
DEFCON8	-	-	-	-	ok	ok	-
UNIBS	ok	ok	Ok	-	-	ok	-
CAIDA	ok	ok	-	-	-	-	-
LBNL	-	ok	-	-	ok	-	-
UNSW-NB15	ok	ok	Ok	-	ok	ok	ok
ISCX	ok	ok	Ok	-	ok	ok	ok
CICIDS 2017	ok	ok	Ok	-	ok	ok	ok
TUIDS	ok	ok	Ok	-	ok	ok	ok
BoT-IoT	ok	ok	Ok	ok	ok	ok	ok

Tomado de (Nickolaos Koroniotis, 2018)

3.3.1. Conjunto de datos DARPA 98

EL conjunto de datos Darpa 98 fue creado por MITS Lincoln Lab el cual evalúa sistemas de detección de intrusos, el resultado se produjo alrededor de 7 semanas, esta tenía 4 GB de datos binarios y una simulación de una pequeña red de la fuerza aérea conectada al internet, que luego se lo mejoro en el año 1999 para dar nuevas funciones al conjunto de datos KDD99. (Nickolaos Koroniotis, 2018)

3.3.2. Conjunto de datos KDD99

El conjunto de datos KDD99 se lo creo a partir del conjunto de datos DARPA 98 para poder evaluar los sistemas de detección de intrusos que pueden distinguir las conexiones normales y ataques, este conjunto de datos se lo sigue utilizando hasta el día de hoy, tiene varios problemas como distribuciones anormales de ataque y datos normales a los cuales se los llama aprendizaje desequilibrado. (Nickolaos Koroniotis, 2018)

Por lo cual se creó un nuevo conjunto de datos llamado NSL-KDD, el cual se lo creo para solucionar las limitaciones que tenía el conjunto de datos KDD99. (Nickolaos Koroniotis, 2018)

3.3.3. Conjunto de datos DEFCON-8

EL conjunto de datos DEFCON-8 escanea los puertos y los ataques de sobre flujo de buffer, los cuales se registraron en una competencia de Capture flag, este carece de una cantidad significativa de tráfico por lo cual no se aplica mucho para la evaluación de sistemas forenses. (Nickolaos Koroniotis, 2018)

3.3.4. Conjunto de datos UNIBS

El conjunto de datos UNIBS lo desarrollo la universidad de Brescia, Italia. Para este conjunto de datos se instalaron 20 lugares de trabajo los cuales ejecutaban el demonio Ground Truth y se recogió el tráfico a través de tcpdump por medio del

enrutador al cual fueron conectados, este conjunto de datos se limitó solo a ataques DoS y no tiene etiquetas en los ataques. (Nickolaos Koroniotis, 2018)

3.3.5. Conjunto de datos CAIDA

El conjunto de datos CAIDA tiene agrupaciones de tipos de datos variados, componiéndose de tráfico encabezado anónimo, los ataque son muy específicos como ataques DDoS, una agrupación de datos característica es el CAIDA DDoS 2007, el cual incluye una hora de ataques anónimos DDoS que se los realizo el 4 de agosto del 2007, el problema de este conjunto de datos es que no tiene instancias en los ataques, así los datos recopilados no tuvieron mejoras por lo que no pudieron mejorar la diferenciación entre ataques y trafico normal. (Nickolaos Koroniotis, 2018)

3.3.6. Conjunto de datos LBNL

El conjunto de datos LBNL tiene tráfico anónimo los cuales solo tienen datos de encabezado, este se lo realizo en el laboratorio Lawrence Berkley recopilando tráfico real tanto de entra como de salida en los routers de borde, este tiene el mismo fallo que el conjunto de datos UNIBS ya que tampoco cuenta con el proceso de etiquetado y sin generar nuevas características como por ejemplo colecciones de archivos PCAP. (Nickolaos Koroniotis, 2018)

3.3.7. Conjunto de datos UNSW-NB15

EL conjunto de datos UNSW-NB15 se lo creo en UNSW Camberra, el creador empleo una tormenta IXIA para así generar tráfico benigno y de ataque, el resultado de esto fue de 100GB en archivos PCAP con un alto número de características generadas, el propósito de este conjunto de datos fue para la generación y validación de detección de intrusos, pero este conjunto de datos solo fue creado en un entorno sintético generando ataques. (Nickolaos Koroniotis, 2018)

3.3.8. Conjunto de datos ISCX

El conjunto de datos ISCX lo creo el instituto canadiense de seguridad cibernética, este utilizo técnicas de ataque y distribución en un entorno de red, analizando varios rastros de ataques reales para poder evaluar los sistemas de detección de intrusos. (Nickolaos Koroniotis, 2018)

3.3.9. Conjunto de datos TUIDS

El conjunto de datos TUIDS fue creado por la universidad de Tezpur, India, este conjunto de datos tiene ataques como DDoS, DoS, Scan, estos ataques fueron realizados en un banco de pruebas físicas, los niveles de flujo no incluyeron nuevas características generadas a parte de las generadas por la captura del flujo. (Nickolaos Koroniotis, 2018)

4. CAPÍTULO IV: IMPLEMENTACION DE MODELO DE RED NEURONAL PERCEPTRON MULTICAPA

En este capítulo se detallará la normalización del conjunto de datos UNSW BoT-IoT y la creación de la red neuronal perceptrón multicapa (MLP) abarcando los procesos utilizados para su creación.

4.1. Descarga Conjunto de Datos

Los creadores del conjunto de datos recomiendan descargar el 5% del conjunto de datos ya que con estos tienen las mejores características para el entrenamiento de una red neuronal. (Nickolaos Koroniotis, 2018)

El conjunto de datos se puede descargar mediante el link https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/bot_iot.php, aquí podremos encontrar todo el conjunto de datos y también el 5% con el cual se va a trabajar, dentro de la carpeta del conjunto de datos hay varios “csv” los cuales contienen los datos como se muestra en la figura 16.

4.2. Procesamiento de Datos

Para procesar los datos se utilizará el software Pycharm en el cual cargaremos los “csv” anteriormente descargados para el procesamiento de los mismos, se utilizará la librería pandas con el que vamos a leer cada uno de los “csv” y también la librería glob con la que vamos a unir todos los “csv” en uno solo. Todos estos procesos realizados se irán guardando en una carpeta local.

4.2.1. Procesar campo “Category”

Para procesar el campo category, se tiene que llamar por medio del path general el archivo que se va a leer con la librería pandas que es el “Csvjoin.csv” el cual contiene todos los datos del conjunto de datos, se va unir el campo “category” con el campo “subcategory” en uno solo que se llamara “category” por medio del siguiente código que se encuentra en la Figura 17.

4.2.2. Procesar campo “sport”

Para procesar el campo “sport” se tuvo un error ya que la red neuronal solo lee datos enteros y flotantes, este campo contenía en ciertas filas números negativos por lo cual se tuvo que leer cada fila para poder eliminarla, esto no afectara al conjunto de datos, ya que se tiene una gran cantidad de datos, además se tenía que algunos datos de esta columna estaban expresado en exponenciales lo cual no podía leer la red neuronal, se lo realizo con el siguiente código que se encuentra en la Figura 18. Todos los archivos “csv” se van guardando en una carpeta local.

4.3. Mapeo del conjunto de datos

Ya que la red neuronal no puede leer texto se tiene que convertir las columnas y filas en números por lo cual se tiene que mapear cada una de ellas dándoles valores numéricos.

4.3.1. Mapeo “category”

Para mapear el campo category se tuvo que realizar la lectura del conjunto de datos para identificar los diferentes datos del campo para poder cambiarlos a números enteros a cada uno de ellos respectivamente como se encuentra en la Tabla 6.

Tabla 6.Mapeo campo “Category”

Categoría	Mapeo
DoSHTTP	1
DoSTCP	2
DoSUDP	3
DDoSHTTP	4
DDoSTCP	5
DDoSUDP	6
ReconnaissanceService_Scan	7
TheftKeylogging	8
TheftData_Exfiltration	9
ReconnaissanceOS_Fingerprint	10
NormalNormal	11

4.3.2. Mapeo “daddr”

Para mapear el campo “daddr”, se tuvo que hacer un conteo por columna de las diferentes ips que tiene la columna para así saber todas las ips que se encuentran utilizadas en el conjunto de datos y poder cambiarlas a cada una de ellas por un numero entero respectivamente como se encuentra detallado en la Tabla 7.

Tabla 7.Mapeo campo “daddr”

daddr	mapeo
192.168.100.3	1
192.168.100.5	2
192.168.100.7	3
192.168.100.6	4
192.168.100.150	5
192.168.100.147	6
192.168.100.149	7
192.168.100.148	8
8.8.8.8	9
192.168.217.2	10
224.0.0.251	11
192.168.100.1	12
192.168.100.255	13
27.124.125.250	14
ff02::fb	15
192.168.100.4	16
ff02::2	17
192.5.5.241	18
192.12.94.30	19
192.41.162.30	20
13.55.154.73	21

ff02::1	22
255.255.255.255	23
199.7.83.42	24
172.217.25.170	25
129.250.35.250	26
91.189.91.157	27
52.11.124.117	28
52.201.147.106	29
199.19.57.1	30
205.251.198.91	31
192.168.100.27	32
192.35.51.30	33
205.251.195.97	34
199.7.91.13	35
192.42.93.30	36
192.52.178.30	37
205.251.193.2	38
192.33.4.12	39
192.36.148.17	40
192.31.80.30	41
192.58.128.30	42
192.168.100.55	43
205.251.196.236	44
216.239.34.10	45
96.7.49.66	46
192.112.36.4	47
91.189.92.40	48
128.63.2.53	49
202.12.27.33	50

192.203.230.10	51
192.55.83.30	52
192.33.14.30	53
205.251.196.160	54
205.251.199.148	55
192.54.112.30	56
156.154.100.3	57
205.251.194.86	58
205.251.194.201	59
205.251.198.239	60
205.251.193.205	61
205.251.196.32	62
199.19.56.1	63
156.154.101.3	64
205.251.197.206	65
205.251.192.170	66
205.251.195.59	67
205.251.194.84	68
192.26.92.30	69
205.251.194.102	70
184.85.248.65	71
205.251.194.167	72
52.35.35.13	73
216.239.38.10	74
224.0.0.252	75
35.165.2.252	76
216.239.36.10	77
205.251.198.119	78
205.251.199.194	79

205.251.199.61	80
205.251.195.185	81
198.41.0.4	82
192.48.79.30	83
205.251.194.154	84
vacío	85

Por medio del siguiente código que se encuentra en la figura 19, se pudo cambiar respectivamente el texto a un número entero como está especificado en la Tabla 7.

4.3.3. Mapeo “flgs”

Para mapear este campo se tuvo que realizar la lectura del conjunto de datos para saber que diferentes datos tiene para poder cambiarlas a cada una de ellas por un número entero respectivamente como se encuentra detallado en la Tabla 8.

Tabla 8.Mapeo campo “flgs”

flgs	mapeado
e	1
e s	2
e g	3
e *	4
eU	5
e d	6
e &	7
e t	8
e D	9

Por medio del siguiente código que se encuentra en la figura 20, se pudo cambiar respectivamente el texto a un número entero como está especificado en la Tabla 8.

4.3.4. Mapeo “saddr”

Para mapear “daddr”, se tuvo que hacer un conteo por columna de las diferentes ips que tiene la columna para así saber todas las ips que se encuentran utilizadas en el conjunto de datos y poder cambiarlas a cada una de ellas por un numero entero respectivamente como se encuentra detallado en la Tabla 9.

Tabla 9.Mapeo campo “saddr”

saddr	mapeado
192.168.100.147	1
192.168.100.148	2
192.168.100.150	3
192.168.100.149	4
192.168.100.3	5
192.168.100.5	6
192.168.100.6	7
192.168.100.7	8
192.168.100.4	9
192.168.100.1	10
192.168.100.46	11
192.168.100.27	12
fe80::250:56ff:febe:254	13
fe80::c0c0:aa20:45b9:bdd9	14
192.168.100.55	15
fe80::250:56ff:febe:26db	16
fe80::250:56ff:febe:e9d9	17
fe80::250:56ff:febe:89ee	18
fe80::2c6a:ff9b:7e14:166a	19
fe80::250:56ff:febe:bf1a	20
fe80::250:56ff:febe:c038	21

Por medio del siguiente código que se encuentra en la figura 21, se pudo cambiar respectivamente el texto a un número entero como está especificado en la Tabla 9.

4.3.5. Mapeo “proto”

Para mapear este campo se tuvo que realizar la lectura del conjunto de datos para saber que diferentes datos tiene para poder cambiarlas a cada una de ellas por un número entero respectivamente como se encuentra detallado en la Tabla 10.

Tabla 10.Mapeo campo “proto”

proto	mapeado
tcp	1
udp	2
icmp	3
arp	4
ipv6-icmp	5

Por medio del siguiente código que se encuentra en la figura 22, se pudo cambiar respectivamente el texto a un número entero como está especificado en la Tabla 10.

4.3.6. Mapeo “state”

Para mapear este campo se tuvo que realizar la lectura del conjunto de datos para saber que diferentes datos tiene para poder cambiarlas a cada una de ellas por un número entero respectivamente como se encuentra detallado en la Tabla 11.

Tabla 11.Mapeo campo “state”

state	mapeado
RST	1
REQ	2
INT	3

CON	4
URP	5
FIN	6
ACC	7
NRS	8
ECO	9
TST	10
MAS	11

Por medio del siguiente código que se encuentra en la figura 23, se pudo cambiar respectivamente el texto a un número entero como está especificado en la Tabla 11.

4.4. Normalizar conjunto de datos

Para normalizar el conjunto de datos se utilizará la librería pandas, numpy y la sklear.preprocessing importando la funcionalidad MinMaxScaler, la función MinMaxScaler nos permitirá normalizar el conjunto de datos ya mapeado, dando valores entre 0 a 1, hay que tomar en cuenta que el campo “category” es el campo con el que vamos a evaluar el funcionamiento de la red neuronal por ende, este campo no se tiene que agregar a la normalización y dejarlo con sus números normalmente, como se muestra en el código que se encuentra en la Figura 25.

Se tiene que poner la cabecera que se desea tener en el csv, después se va aguardar el archivo en la carpeta local.

4.5. Creación red neuronal MLP

Para crear la red neuronal MLP vamos a utilizar las librerías sklearn.neural_network import MLPClassifier, sklearn.model_selection import train_test_split, por medio de MLPClassifier nos permite crear directamente la red neuronal MLP, con train_test_split podemos dividir el conjunto de datos en dos partes tanto en la parte de entrenamiento para la red neuronal y el test, pero antes se tiene que transformar

por medio de numpy los datos a matrices para que así se pueda entrenar la red neuronal, dándole el 80 % para entrenamiento y el 20 % para el test, después de eso se puede entrenar la red neuronal especificando las neuronas de entrada, salida y ocultas, utilizando el comando “fit” se da a conocer con que datos se va a entrenar la red neuronal. Como se muestra el código en la Figura 26.

4.6. Muestreo de datos

Para mostrar el resultado de la red neuronal se va a realizar una matriz de confusión por medio de las librerías sklearn.metrics import confusion_matrix, sklearn.metrics import classification_report, por medio de la propiedad confusión_matrix por medio de la predicción de la red neuronal ya entrenada y el test, así se crea la matriz de confusión pero se tiene que interpretar, para poder interpretar se utilizara la propiedad de classification_report, el cual nos permitirá interpretar los datos y el rendimiento de la red neuronal el cual se muestra en la Figura 27.

Al interpretar los datos se tiene la precisión para cada categoría siendo así que mientras más cerca del 1 se encuentre la predicción de la red neuronal significa que es exitosa, en cuanto recall es el porcentaje entre 1 y 0 de que tan acertada fue la clasificación de los datos

5. CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

Se utilizó la red neuronal perceptron multicapa(MLP) por tener una alta tasa de efectividad en las predicciones en saber el tipo de ataque de botnet que se encuentra en el conjunto de datos UNSW-Bot-Iot.

Se realizó un análisis de las diferentes herramientas para procesar conjuntos de dados, de acuerdo al análisis que se realizó se optó por una programación en

Python ya que dentro de ella existe la herramienta pandas la cual nos permite clasificar de la mejor manera los datos dando así que el resultado de este permita el entrenamiento optimo a la red neuronal perceptron multicapa (MLP)

Se utilizó el 5 % de toda la data por la razón que estas contienen las mejores características dentro del etiquetado de los ataques, así como el creador del conjunto de datos sugiere descargar el mismo

En el entrenamiento de la red neuronal se tiene que utilizar el 80 % de los datos para el entrenamiento y el 20 % para el testeo, ya que de esta forma la red neuronal tiene la capacidad de detectar ataques de una forma eficiente.

En el entrenamiento de la red neuronal se fueron ajustando los números de neuronas en cada capa ya que al principio al entrenarla no se obtenía un score adecuado para realizar la matriz de confusión por lo cual se cambió el número de neuronas en cada capa hasta que el score sea aceptable

5.2. Recomendaciones

Para un buen funcionamiento de la red neuronal, es recomendable instruir la red en un futuro con nuevos ataques que exista de tipo botnet.

Se recomienda probar el modelo de red neuronal con datos de una red en producción para la eficiencia del mismo

El lenguaje de programación Python es indispensable en la creación de las redes neuronales, ya que tiene librerías que nos permite crear las mismas de una manera fácil, ahorrando recursos y tiempo.

Se recomienda instruir la red neuronal con todo el conjunto de datos para más asertividad en las predicciones del mismo

6. Referencias

A. Requena, R. Q. (s.f). *Universidad de Murcia*. Obtenido de Universidad de Murcia: https://www.um.es/LEQ/Atmosferas/Ch-VI-3/F63s5p1.htm

Agncia de Regularizacion y Control de las Telecomunicaciones . (sf). *Agncia de Regularizacion y Control de las Telecomunicaciones* . Obtenido de Agncia de Regularizacion y Control de las Telecomunicaciones : https://www.ecucert.gob.ec/assets/botnet2.pdf

Alfonso Palmer, J. J. (2002). *Researchgate*. Obtenido de Researchgate: https://www.researchgate.net/publication/331641249_Tutorial_sobre_Redes_Neuronales_Artificiales_Los_Mapas_Autoorganizados_de_Kohonen

Amazon Web Services. (15 de Abril de 2020). *aws*. Obtenido de aws: https://aws.amazon.com/es/what-is-cloud-computing/

APD, R. (04 de Marzo de 2019). *apd.* Obtenido de apd: https://www.apd.es/que-es-machine-learning/

avast. (10 de Abril de 2020). *avast.* Obtenido de avast: https://www.avast.com/es-es/c-botnet

AVG Internet Security. (16 de Febrero de 2019). *AVG*. Obtenido de AVG: https://www.avg.com/es/signal/what-is-botnet

Ballesteros, A. (s.f). *Neural Networks Frameworks*. Obtenido de Neural Networks Frameworks: http://www.redes-neuronales.com.es/tutorial-redes-neuronales/Las-redes-neuronales-multicapa.htm

BBVA. (08 de Noviembre de 2019). *bbva*. Obtenido de bbva: https://www.bbva.com/es/machine-learning-que-es-y-como-funciona/

Calvo, D. (08 de Diciembre de 2018). *Diego Calvo*. Obtenido de Diego Calvo: https://www.diegocalvo.es/perceptron/

Campos, O. (01 de Febrero de 2012). *GENBETA*. Obtenido de GENBETA: https://www.genbeta.com/desarrollo/testeando-nuestras-aplicaciones-de-red-con-scapy

Caparrini, F. S. (14 de Diciembre de 2019). *Universidad de Sevilla*. Obtenido de Universidad de Sevilla: http://www.cs.us.es/~fsancho/?e=72

COPIMAR. (s.f). *COPIMAR growing together*. Obtenido de COPIMAR growing together: https://copimar.net/tipos-de-nubes/

Cyberseguridad. (s.f). *Cyberseguridad*. Obtenido de Cyberseguridad: https://www.cyberseguridad.net/index.php/ataques-informaticos

Delgado, D. O. (Septiembre de 15 de 2017). *OpenWebinars*. Obtenido de OpenWebinars: https://openwebinars.net/blog/que-es-tensorflow/

EINATEC. (s.f). *einatec*. Obtenido de einatec: https://einatec.com/tipos-de-cloud-computing/#Nube_publica

EY. (s.f). *EY*. Obtenido de EY: https://www.ey.com/es_ec/internet-of-things-iot

Fernandez, D. O. (Septiembre de 2017). *Rua repositorio institucional de la universidad de alicante*. Obtenido de Rua repositorio institucional de la universidad de alicante: https://rua.ua.es/dspace/bitstream/10045/70314/1/TFG-Daniel-Ortega.pdf

Galiano, F. B. (s.f). *elvex*. Obtenido de elvex: http://elvex.ugr.es/software/nc/help/spanish/nc/classification/lvq.html

Garzón, F. A. (2016). *Universidad Nacional de Colombia*. Obtenido de Universidad Nacional de Colombia: http://bdigital.unal.edu.co/50820/1/8061440.2016.pdf

Gepperth, A. (29 de Abril de 2016). *Universidad de Paris Saclay*. Obtenido de Universidad de Paris Saclay: https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2016-19.pdf

González, L. (21 de Septiembre de 2018). *ligdigonzales*. Obtenido de ligdigonzales: https://ligdigonzalez.com/introduccion-a-numpy-python-1/

Gonzalez, M. V. (2015). *Universitat de les Illes Balears*. Obtenido de Universitat de les Illes Balears: https://dspace.uib.es/xmlui/bitstream/handle/11201/1126/TFG%20Marta%20Vidal%20Gonz%C3%A1lez.pdf?sequence=1

Google cloud. (s.f). *Google cloud*. Obtenido de Google cloud: https://cloud.google.com/solutions/iot?hl=es

Gutiérrez, A. (03 de Abril de 2020). *robolabo*. Obtenido de robolabo: http://www.robolabo.etsit.upm.es/asignaturas/irin/transparencias/redesNeuronales.pdf

Informatica, S. (10 de Noviemnre de 2013). *Amenazas y fraudes en los sistemas de la información*. Obtenido de https://infosegur.wordpress.com/2013/11/10/amenazas-y-fraudes-en-los-sistemas-de-la-informacion/

Instituto Nacional de Ciberseguridad. (14 de Junio de 2018). *Incibe-cert*. Obtenido de Incibe-cert: https://www.incibe-cert.es/blog/honeypot-herramienta-conocer-al-enemigo

IONOS España S.L.U. (s.f). *IONOS*. Obtenido de IONOS: https://www.ionos.es/digitalguide/servidores/seguridad/honeypot-seguridad-informatica-para-detectar-amenazas/

IONOS. (s.f). *Digital Guide IONOS*. Obtenido de Digital Guide IONOS: https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/bases-de-datos/

Kaspersky. (10 de Marzo de 2020). *Kaspersky.* Obtenido de Kaspersky: https://latam.kaspersky.com/resource-center/threats/ddos-attacks

Marin, J. M. (s.f). *uc3m*. Obtenido de Universidad Carlos 3 de Madrid: http://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/DM/tema3dm.pdf

Marin, J. M. (s.f). *us3m*. Obtenido de us3m: http://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/DM/tema5dm.pdf

Matich, D. J. (Marzo de 2001). *Universidad Tecnologica Nacional* . Obtenido de Universidad Tecnologica Nacional : https://www.frro.utn.edu.ar/repositorio/catedras/quimica/5_anio/orientadora1/monograias/matich-redesneuronales.pdf

Mercado Polo, D., Pedraza Caballero, L., & Gómez, M. (Julio de 2015). *Redalyc*. Obtenido de Redalyc: https://www.redalyc.org/pdf/4962/496250642011.pdf

Microsoft Azure. (20 de Marzo de 2019). *Microsoft Azure*. Obtenido de Microsoft Azure: https://azure.microsoft.com/es-es/overview/what-is-saas/

Microsoft. (s.f). *Microsoft Azure*. Obtenido de Microsoft Azure: https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/

Mieres, J. (25 de Enero de 2009). *evil fingers.* Obtenido de evil fingers: https://s3.amazonaws.com/academia.edu.documents/39004327/01_Ataques_informaticos.pdf?response-content-disposition=inline%3B%20filename%3D01_Ataques_informaticos.pdf&X-

Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=ASIATUSBJ6BAK747VEVI%2F20200411%2Fus-eas

missinglink. (13 de Abril de 2020). *missinglink*. Obtenido de missinglink: https://missinglink.ai/guides/neural-network-concepts/perceptrons-and-multi-layer-perceptrons-the-artificial-neuron-at-the-core-of-deep-learning/

Montoro, A. F. (2012). *Python 3 al descubierto.* Madrid: RC libros.

Navarro, V. B. (s.f). *ResarchGate*. Obtenido de ResarchGate: https://www.researchgate.net/figure/Figura-3-Ejemplo-de-perceptron-multicapa_fig1_228815505

Nickolaos Koroniotis, N. M. (2018). *Bot-IoT Dataset.* Canberra, Australia: University of New South Wales .

Olabe, X. B. (s.f). *Universidad del país Vasco*. Obtenido de Universidad del país Vasco: https://ocw.ehu.eus/pluginfile.php/9047/mod_resource/content/1/redes_neuro/contenidos/pdf/libro-del-curso.pdf

Pare, R. C. (Mayo de 2005). *Universidad Oberta de Catalunya*. Obtenido de Universidad Oberta de Catalunya: https://www.uoc.edu/pdf/masters/oficiales/img/913.pdf

Peña, M. (21 de noviembre de 2019). *DIGITAL TRENDS*. Obtenido de DIGITAL TRENDS: https://es.digitaltrends.com/tendencias/que-es-el-internet-de-las-cosas/

Perez, R. S. (1994). *Redes Neuronales una aplicacion para la prediccion de ozono.* Monterrey: Instituto tecnologico de monterrey.

SAEM THALES. (s.f). *SOCIEDAD ADALUZA DE EDUCACION MATEMÁTICA THALES*. Obtenido de SOCIEDAD ADALUZA DE EDUCACION

MATEMÁTICA THALES: https://thales.cica.es/rd/Recursos/rd98/TecInfo/07/capitulo6.html

Salas, R. (2004). Redes Neuronales Artificiales. En R. Salas, *Redes Neuronales Artificiales* (págs. 1-7). Valparaiso: Universidad de Valparaiso.

Sociedad andaluza de matematicas. (s.f). *SAEM Thales*. Obtenido de SAEM Thales: https://thales.cica.es/rd/Recursos/rd98/TecInfo/07/capitulo4.html

Soriano, M. (2014). *Seguridad en redes y seguridad de la información.* Republica Checa: České vysoké učení technické v Praze .

Torfi, A. (13 de Abril de 2020). *Python Machine Learning*. Obtenido de Python Machine Learning: https://machine-learning-course.readthedocs.io/en/latest/content/deep_learning/mlp.html#what-defines-a-multilayer-perceptron

T-Systems International GmbH. (10 de Abril de 2019). *T-Systems* . Obtenido de T-Systems : https://www.t-systemsblog.es/que-es-un-honeypot/

Universidad de Valencia. (24 de Abril de 2018). *Universidad Internacional de Valencia*. Obtenido de Universidad Internacional de Valencia: https://www.universidadviu.com/las-4-claves-la-seguridad-la-informacion/

USS. (20 de Febrero de 2019). *USS*. Obtenido de USS: https://uss.com.ar/preguntas-frecuentes/sistema-de-deteccion-de-intrusos/

YellowBrick. (s.f). *YellowBrick*. Obtenido de YellowBrick: https://www.scikit-yb.org/en/latest/api/classifier/classification_report.html

Anexo

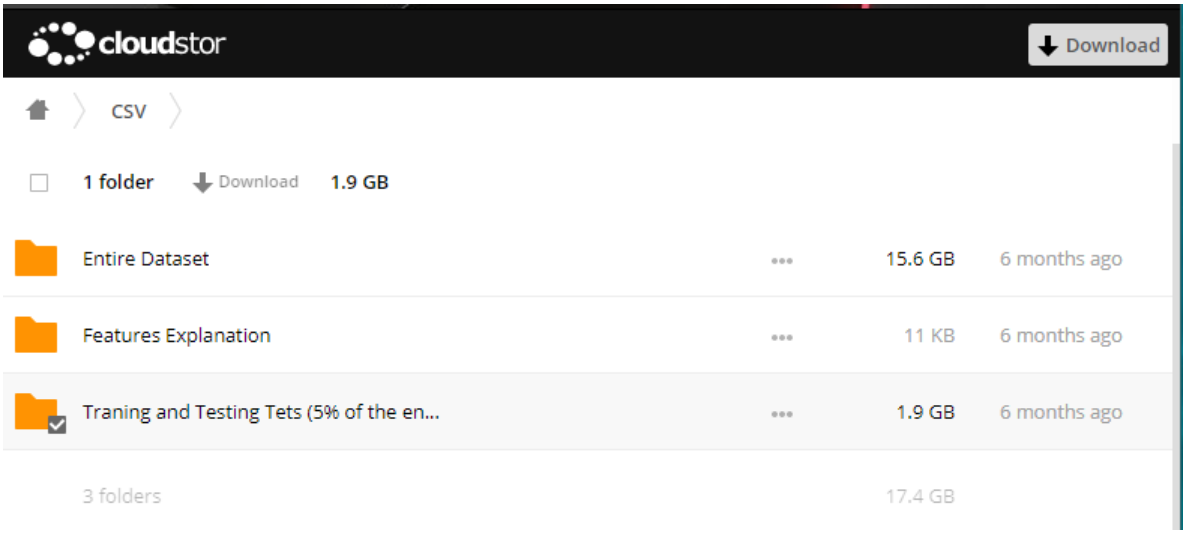


Figura 16. 5% del conjunto de datos

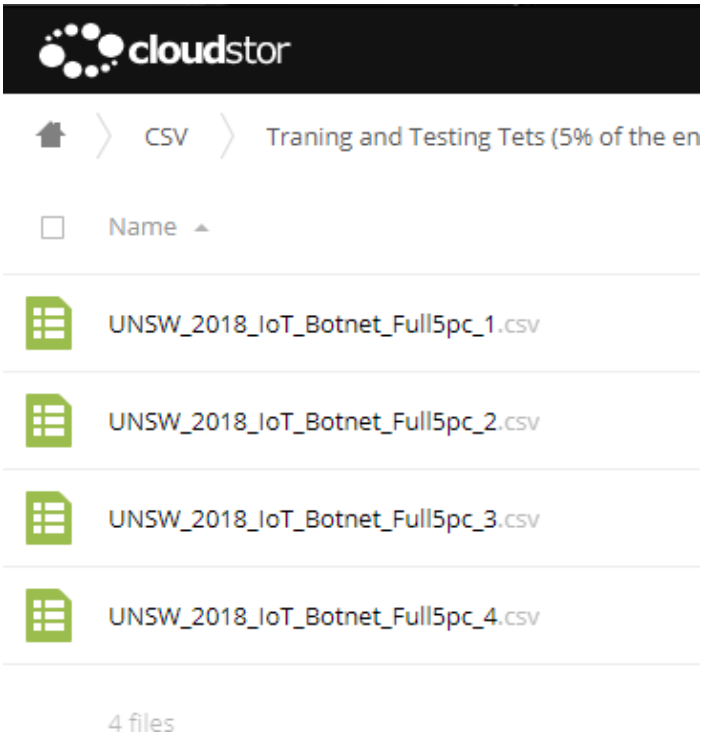


Figura 17.Conjunto de datos

```
import pandas as pd
path_saida = r"C:/Users/Victor Totoy/Documents/nuevo/"
df = pd.read_csv(path_saida + 'CSVjoin.csv')
df['category']=df['category']+df['subcategory']
```

Figura 18. Código para “Category”

```
import pandas as pd
path_saida = r"C:/Users/Victor Totoy/Documents/nuevo/"
df = pd.read_csv(path_saida + 'CSVjoin.csv')
for index, row in df.iterrows():
    x=row['sport']
    if str(x)[:2] == '0x':
        df.loc[df['sport'] == x, 'sport'] = int(x,0)
    if x == -1:
     df=df.drop([index])
df=df.drop(['subcategory'], axis=1)
print(df.head(10))
df.to_csv(path_saida+"CSVjoin-pulido.csv", index=None)
```

Figura 19.Código para el campo “sport”

```
df.loc[df['daddr'] == '192.168.100.3', "daddr"] = int(1)
df.loc[df['daddr'] == '192.168.100.5', "daddr"] = int(2)
df.loc[df['daddr'] == '192.168.100.7', "daddr"] = int(3)
df.loc[df['daddr'] == '192.168.100.6', "daddr"] = int(4)
df.loc[df['daddr'] == '192.168.100.150', "daddr"] = int(5)
df.loc[df['daddr'] == '192.168.100.147', 'daddr'] = int(6)
df.loc[df['daddr'] == '192.168.100.149', 'daddr'] = int(7)
df.loc[df['daddr'] == '192.168.100.148', 'daddr'] = int(8)
df.loc[df['daddr'] == '8.8.8.8', 'daddr'] = int(9)
```

Figura 20.Código mapeo “daddr”

```
df.loc[df['flgs'] == 'e', 'flgs'] = int(1)
df.loc[df['flgs'] == 'e s', 'flgs'] = int(2)
df.loc[df['flgs'] == 'e g', 'flgs'] = int(3)
df.loc[df['flgs'] == 'e *', 'flgs'] = int(4)
df.loc[df['flgs'] == 'eU', 'flgs'] = int(5)
df.loc[df['flgs'] == 'e d', 'flgs'] = int(6)
df.loc[df['flgs'] == 'e &', 'flgs'] = int(7)
df.loc[df['flgs'] == 'e   t', 'flgs'] = int(8)
df.loc[df['flgs'] == 'e  D', 'flgs'] = int(9)
```

Figura 21.Código mapeo “flgs”

```
df = pd.read_csv(path_saida + 'CSVjoin-map.csv')
df.loc[df['saddr'] == '192.168.100.147', "saddr"] = int(1)
df.loc[df['saddr'] == '192.168.100.148', "saddr"] = int(2)
df.loc[df['saddr'] == '192.168.100.150', "saddr"] = int(3)
df.loc[df['saddr'] == '192.168.100.149', "saddr"] = int(4)
df.loc[df['saddr'] == '192.168.100.3', "saddr"] = int(5)
df.loc[df['saddr'] == '192.168.100.5', 'saddr'] = int(6)
df.loc[df['saddr'] == '192.168.100.6', 'saddr'] = int(7)
df.loc[df['saddr'] == '192.168.100.7', 'saddr'] = int(8)
df.loc[df['saddr'] == '192.168.100.4', 'saddr'] = int(9)
df.loc[df['saddr'] == '192.168.100.1', 'saddr'] = int(10)
df.loc[df['saddr'] == '192.168.100.46', 'saddr'] = int(11)
```

Figura 22.Código mapeo “saddr”

```
df.loc[df['proto'] == 'tcp', "proto"] = int(1)
df.loc[df['proto'] == 'udp', "proto"] = int(2)
df.loc[df['proto'] == 'icmp', "proto"] = int(3)
df.loc[df['proto'] == 'arp', "proto"] = int(4)
df.loc[df['proto'] == 'ipv6-icmp', "proto"] = int(5)
df.to_csv(path_saida + "CSVjoin-map.csv", index=None)
```

Figura 23.Código mapeo “proto”

```
df.loc[df['state'] == 'RST', 'state'] = int(1)
df.loc[df['state'] == 'REQ', 'state'] = int(2)
df.loc[df['state'] == 'INT', 'state'] = int(3)
df.loc[df['state'] == 'CON', 'state'] = int(4)
df.loc[df['state'] == 'URP', 'state'] = int(5)
df.loc[df['state'] == 'FIN', 'state'] = int(6)
df.loc[df['state'] == 'ACC', 'state'] = int(7)
df.loc[df['state'] == 'NRS', 'state'] = int(8)
df.loc[df['state'] == 'ECO', 'state'] = int(9)
df.loc[df['state'] == 'TST', 'state'] = int(10)
df.loc[df['state'] == 'MAS', 'state'] = int(11)
```

Figura 24.Código mapeo “state”

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
path_saida = r"C:/Users/Victor Totoy/Documents/nuevo/"
df = pd.read_csv(path_saida + 'CSVjoin-map-final1.csv')
scaler= MinMaxScaler()
transformed_df=scaler.fit_transform(df.iloc[:, :44])
transformed_df=np.round(transformed_df,decimals=5)
transformed_df = pd.DataFrame(transformed_df)
transformed_df["category"] = df["category"]
transformed_df.columns = transformed_df.columns
print(transformed_df)
keep_col = ['Unnamed: 0','pkSeqID','stime','flgs','flgs_number','proto','proto_number','saddr','sport','daddr'
            ,'dport','pkts','bytes','state','state_number','ltime','seq','dur','mean','stddev','sum','min','max','spkts','dpkts'
            ,'sbytes','dbytes','rate','srate','drate','TnBPSrcIP','TnBPDstIP','TnP_PSrcIP','TnP_PDstIP','TnP_PerProto','TnP_Per_Dport',
            'AR_P_Proto_P_SrcIP','AR_P_Proto_P_DstIP','N_IN_Conn_P_DstIP','N_IN_Conn_P_SrcIP','AR_P_Proto_P_Sport',
            'AR_P_Proto_P_Dport','Pkts_P_State_P_Protocol_P_DestIP','Pkts_P_State_P_Protocol_P_SrcIP','category']
transformed_df.to_csv(path_saida+"CSVjoin-normalizado.csv", index=None, header=keep_col)
```

Figura 25.Código de normalización

```
import pandas as pd
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from joblib import dump, load #guarda y carga un modelo
path_saida = r"C:/Users/Victor Totoy/Documents/nuevo/"
df = pd.read_csv(path_saida + 'CSVjoin-normalizado-final.csv')
matriz=df.values
y=matriz[:,42]
X=matriz[:,0:42]
X_train,X_test,y_train,y_test = train_test_split(X, y,random_state=8,train_size=0.8,test_size=0.2)
mlp= MLPClassifier(hidden_layer_sizes=(44,20,20,7), solver='adam').fit(X_train, y_train)
score=mlp.score(X_test,y_test)
print(score)
dump(mlp,path_saida+'modelomlp.joblib')
```

Figura 26.Código creación MLP

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from joblib import dump, load #guarda y carga un modelo
path_saida = r"C:/Users/Victor Totoy/Documents/nuevo/"
df = pd.read_csv(path_saida + 'CSVjoin-normalizado-final.csv')
matriz=df.values
y=matriz[:,42]
X=matriz[:,0:42]
X_train,X_test,y_train,y_test = train_test_split(X, y,random_state=8,train_size=0.8,test_size=0.2)
mlp=load(path_saida+'modelomlp.joblib')
prediction=mlp.predict(X_test)
cm =confusion_matrix(y_test,prediction)
print (cm)
print(classification_report(y_test,prediction))
```

Figura 27.Matriz de Confusión

	precision	recall	f1-score	support
1.0	0.97	0.97	0.97	292
2.0	1.00	1.00	1.00	123459
3.0	1.00	1.00	1.00	206504
4.0	0.96	0.93	0.95	204
5.0	1.00	1.00	1.00	195485
6.0	1.00	1.00	1.00	189334
7.0	1.00	1.00	1.00	14724
8.0	0.83	1.00	0.91	10
9.0	0.00	0.00	0.00	2
10.0	1.00	1.00	1.00	3578
11.0	0.99	0.99	0.99	97
accuracy			1.00	733689
macro avg	0.89	0.90	0.89	733689
weighted avg	1.00	1.00	1.00	733689

Figura 28.reporte de conjunto de datos