UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES

TITULO DEL TRABAJO DE TITULACIÓN

Aplicación de modelos de aprendizaje para la detección y prevención de ataques, aplicando pentesting de nueva generación

AUTOR

Prudente Del Pezo Eddie Josue

TRABAJO DE INTEGRACIÓN CURRICULAR

Previo a la obtención del grado académico en
INGENIERO EN TELECOMUNICACIONES

TUTOR

Ing. Chamba Macas Fernando Vinicio

La Libertad, Ecuador

Año 2024

DECLARACIÓN DE DOCENTE TUTOR

En mi calidad de Docente Tutor del Trabajo de Integración Curricular, **"Aplicación de modelos de aprendizaje para la detección y prevención de ataques, aplicando pentesting de nueva generación"**, elaborado por **Prudente Del Pezo Eddie Josue,** estudiante de la Carrera de Telecomunicaciones, Facultad de Sistemas y Telecomunicaciones de la Universidad Estatal Península de Santa Elena, previo a la obtención del título de **Ingeniería en Telecomunicaciones**, me permito declarar que, tras supervisar el desarrollo y estructura final del trabajo, este cumple y se ajusta a los estándares académicos. En consecuencia, lo considero apto en todos sus aspectos y listo para ser evaluado por el docente especialista.

Atentamente,

Ing. Fernando Chamba Macas, Mgt

DOCENTE TUTOR

DECLARACIÓN AUTORÍA DEL ESTUDIANTE

El presente trabajo de Integración Curricular con el título, **"Aplicación de modelos de aprendizaje para la detección y prevención de ataques, aplicando pentesting de nueva generación"**, elaborado por **Prudente Del Pezo Eddie Josue,** declaro que la concepción, análisis y resultados son originales a la actividad educativa en el área de Telecomunicaciones

Atentamente,

Eddie Prudente

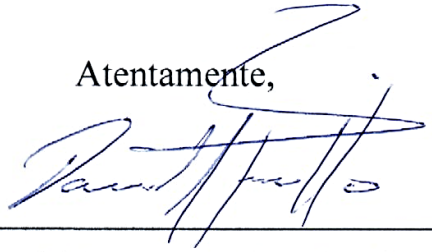
Prudente Del Pezo Eddie Josue

C.I: 2400130411

DECLARACIÓN DE DOCENTE ESPECIALISTA

En mi calidad de Docente Especialista del Trabajo de Integración Curricular, **"Aplicación de modelos de aprendizaje para la detección y prevención de ataques, aplicando pentesting de nueva generación"**, elaborado por **Prudente Del Pezo Eddie Josue,** estudiante de la Carrera de Telecomunicaciones, Facultad de Sistemas y Telecomunicaciones de la Universidad Estatal Península de Santa Elena, previo a la obtención del título de **Ingeniería en Telecomunicaciones**, me permito declarar que, tras supervisar el desarrollo y estructura final del trabajo, este cumple y se ajusta a los estándares académicos. En consecuencia, lo considero apto en todos sus aspectos y listo para la sustentación del trabajo.

Atentamente,



Ing. Daniel Jaramillo Chamba, Mgt.

DOCENTE ESPECIALISTA

TRIBUNAL DE SUSTENTACIÓN

Ing. Ronald Rovira Jurado, PhD.

DIRECTOR DE LA CARRERA

Ing. Fernando Chamba Macas, Mgt

DOCENTE TUTOR

Ing. Daniel Jaramillo Chamba, Mgt

DOCENTE ESPECIALISTA

Ing. Luis Amaya Fariño, Mgt.

DOCENTE GUÍA UIC

Ing. Corina Gonzabay DeLaA, Mgt

SECRETARIA

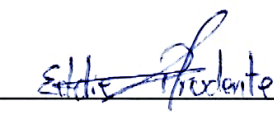
DECLARACIÓN DE RESPONSABILIDAD

Yo, **PRUDENTE DEL PEZO EDDIE JOSUE**

DECLARO QUE:

El trabajo de Titulación, **"Aplicación de modelos de aprendizaje para la detección y prevención de ataques, aplicando pentesting de nueva generación"** previo a la obtención del título en Ingeniero en Telecomunicaciones, ha sido desarrollado respetando derechos intelectuales de terceros conforme las citas que constan en el documento, cuyas fuentes se incorporan en las referencias o bibliografías. Consecuentemente este trabajo es de mi total autoría.

Atentamente,



Prudente Del Pezo Eddie Josue

C.I: 2400130411

AGRADECIMIENTO

Le agradezco a Dios en primer lugar, por haberme dado salud y la fortaleza necesaria para culminar esta etapa de mi formación académica y culminar con éxito este proyecto.

A mi madre, Reyna, por su apoyo incondicional y dedicación a lo largo de estos años, la motivación constante para avanzar y mejorar está representados en este trabajo, con esfuerzo y dedicación.

A mi padre, Javier, por inculcarme valores que han sido esenciales en mi desarrollo personal y académico. Sin su sacrificio y confianza no hubiese podido llegar tan lejos.

A mi tutor, el ingeniero Fernando Chamba, por la orientación y dedicación. Su sabiduría y paciencia han sido otro motivo para culminar con éxito este proyecto.

A los docentes de la carrera en general, que dedicaron su tiempo en inculcarme sus conocimientos, les agradezco por su compromiso y pasión por la enseñanza. Cada uno de ellos me incentivaron a querer motivarme cada día.

Eddie Josue Prudente Del Pezo

DEDICATORIA

Dedico este proyecto a mi padre, Javier, por enseñarme el valor del trabajo duro y que con perseverancia se puede llegar a la meta. Sus consejos de vida y ejemplo me han guiado en cada paso hasta el día de hoy, agradecido por ser un pilar fundamental en mi vida.

A mi querida madre, Reyna, quien ha sido mi fuente de motivación constante, quien depositó su fe y confianza en mí, le dedico este logro por todas esas palabras de aliento que me dieron fuerzas para superar cualquier obstáculo. Gracias por confiar en mí y por enseñarme a no rendirme ante las dificultades.

A mi querida Adriana, por su paciencia y cariño al estar a mi lado desde el inicio de la carrera, por apoyarme en cada etapa a lo largo de estos años, por brindarme palabras de ánimo cuando más lo necesitaba convirtiéndose en una persona muy importante en mi vida. Gracias por creer en mí hasta el final, por ser mi fortaleza y apoyo incondicional en todo este proceso.

A mis hermanos, Danna, Dave y Elkin, gracias por el apoyo, aliento y motivación que me dieron para terminar este proyecto y por ser una fuente de inspiración.

A mis amigos más cercanos, Joarkel y Carolina, este logro también es para ustedes y por su gran amistad a pesar de la distancia, cada risa compartida y cada consejo brindado son un tesoro invaluable en mi vida.

Eddie Josue Prudente Del Pezo

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE ACRÓNIMOS

ANN:	Artificial Neutral Network (Red Neuronal Artificial)
AP:	Access Point (Punto de Acceso)
API:	Application Programming Interface (Interfaz de Programación de Aplicaciones
ARCOTEL:	Agencia de Regulación y control de telecomunicaciones
ARP:	Address Resolution Protocol (Protocolo de Resolución de Direcciones)
COIP:	Código Integral Penal
DDoS:	Distributed Denial of Service Attack (Ataque de Denegación de Servicios Distribuido)
DL:	Deep Learning (Aprendizaje Profundo)
DNS:	Domain Name System (Sistema de Nombres de Dominio)
DoS:	Denial of Service (Ataque de Denegación de Servicios)
DT:	Decision Tree (Árbol de Decisión)
EAPOL:	Extensible Authentication Protocolo over LAN (Protocolo de Autenticación Extensible sobre LAN)
FN:	False Negative (Falso Negativo)
FP:	False Positive (Falso Positivo)
HTTP:	Hypertext Transfer Protocol (Protocolo de Transferencia de Hipertexto)
IA:	Artificial Intelligence (Inteligencia Artificial)
IDS:	Intrusion Detection System (Sistema de Detección de Intrusos)
IEEE:	Institute of Electrical and Electronics Engineers (Instituto de Ingenieros Eléctricos y Electrónicos)
IoT:	Internet of Things (Internet de las Cosas)
IP:	Internet Protocol (Protocolo de Internet)
KNN:	K-Nearest Neighbour (K-Vecinos Más Cercanos)
MAC:	Media Access Control (Control de Acceso a Medios)
MAE:	Mean Absolute Error (Error Absoluto Medio)
MitM:	Man in the middle (Hombre en el Medio)
ML:	Machine Learning (Aprendizaje Automático)
NB:	Naive Bayes

Nmap:	Network Mapper (Mapeador de Red)
PCAP:	Packet Capture (Captura de Paquetes)
QoS:	Quality of Service (Calidad de Servicio)
RF:	Random Forest (Bosque Aleatorio)
RL:	Logistic Regression (Regresión Logística)
RMSE:	Root Mean Square Error (Error Cuadrático Medio)
SML:	Supervised Machine Learning (Aprendizaje Automático Supervisado)
SRL:	Simple Linear Regression (Regresión Lineal Simple)
SSID:	Service Set Identifier (Identificador de Red de Servicio)
SVM:	Support Vector Machine (Máquina de Vectores de Soporte)
TCP:	Transmission Control Protocol (Protocolo de Control de Transmisión)
TN:	True Negative (Verdadero Negativo)
TP:	True Positive (Verdadero Positivo)
UML:	Unsupervised Machine Learning (Aprendizaje Automático No Supervisado)
WAN:	Wide Area Network (Red de Área Amplia)
WLAN:	Wireless Local Area Network (Red de Área Local Inalámbrica)
WPA:	Wi-Fi Protected Access (Acceso Wi-Fi Protegido)
XGBoost:	Extreme Gradient Boosting (Refuerzo de Gradientes Extremo)

ÍNDICE DE ANEXOS

RESUMEN

En la actualidad, la seguridad de las redes de telecomunicaciones se ha convertido en un tema de vital importancia debido al creciente número de ataques cibernéticos. En este contexto, este proyecto se centra en la aplicación de modelos de aprendizaje supervisado para la detección y prevención de ataques, utilizando técnicas de pentesting de nueva generación. Este estudio se lleva a cabo en un entorno de prueba controlado, específicamente en el laboratorio de telecomunicaciones.

Los objetivos son distinguir los tipos de ataques de red que pueden ser realizados mediante el dispositivo "WiFi Pineapple Mark VII", identificar y seleccionar los modelos de aprendizaje supervisado más efectivos, y demostrar exitosamente su implementación.

Se creó una pequeña red para pruebas de penetración usando módulos del Pineapple, capturando el tráfico con Wireshark y almacenándolo en un dataset para entrenamiento y pruebas de predicciones. Los modelos probados incluyen Regresión Lineal, Regresión Logística, Random Forest, Extreme Gradient Boosting, Decision Tree, K-Vecinos más cercanos (KNN), y Máquina de Vectores de Soporte (SVM).

En los resultados obtenidos se demostró que el modelo KNN mostró mayor eficiencia, detectando correctamente los ataques. Para prevenir estos ataques, se implementó un bot de Telegram que envía alertas con detalles del ataque detectado, permitiendo al usuario tomar acciones preventivas.

Palabras claves:

Aprendizaje supervisado, detección de ataques, pentesting, WiFi Pineapple, modelo KNN

ABSTRACT

Nowadays, the security of telecommunication networks has become a vitally important issue due to the increasing number of cyber-attacks. In this context, this project focuses on the application of supervised learning models for the detection and prevention of attacks, using new generation pentesting techniques. This study is carried out in a controlled test environment, specifically in the telecommunications laboratory.

The objectives are to distinguish the types of network attacks that can be performed using the "WiFi Pineapple Mark VII" device, identify and select the most effective supervised learning models, and successfully demonstrate their implementation.

A small penetration test network was created using Pineapple modules, capturing traffic with Wireshark, and storing it in a dataset for training and prediction testing. The models tested include Linear Regression, Logistic Regression, Random Forest, Extreme Gradient Boosting, Decision Tree, K-Nearest Neighbours (KNN), and Support Vector Machine (SVM).

In the results obtained, it was shown that the KNN model showed higher efficiency, correctly detecting attacks. To prevent these attacks, a Telegram bot was implemented that sends alerts with details of the detected attack, allowing the user to take preventive actions.

Keywords:

Supervised learning, attack detection, pentesting, Pineapple WiFi, KNN model

INTRODUCCIÓN

En la era actual, la seguridad de las redes se ha convertido en una preocupación crítica debido al incremento constante de ciberataques sofisticados. Las organizaciones enfrentan desafíos significativos para proteger sus infraestructuras de red contra una amplia variedad de amenazas, que van desde ataques de denegación de servicio hasta intrusiones más avanzadas que pueden comprometer datos sensibles. La creciente dependencia de las tecnologías de la información y la comunicación ha amplificado la necesidad de desarrollar y aplicar métodos eficaces para la detección y prevención de estos ataques. Los profesionales conocidos como hackers éticos o pentesters, suelen utilizar métodos de intrusión para evaluar la seguridad de alguna infraestructura de red, principalmente de empresas grandes en donde se requiere un manejo de la información en confidencialidad. Es por ello que existen equipos especializados para este campo, como lo son los equipos de la marca Hak5, ofrece múltiples herramientas para realizar ataques, uno de ellos es el dispositivo “WiFi Pineapple Mark VII”, el cual es un router con 3 antenas que permite al usuario escanear las redes cercanas para realizar ataques de desautenticación, clonar redes y utilizar portales cautivos para sustraer las credenciales de las víctimas, también cuenta con múltiples módulos que le añaden una arquitectura más completa al dispositivo y una versatilidad a la hora de utilizarlo debido a su interfaz completa [1]. Otra herramienta que sirve como complemento es el conocido software Wireshark, este es un analizador del tráfico de red ya que permite capturar los paquetes que transcurren dentro de una infraestructura de red, utilizando esta herramienta se puede realizar un análisis profundo de los paquetes capturados [2]. El crecimiento de las tecnologías como el Machine Learning, Deep Learning e Inteligencia Artificial suelen ser un complemento a la hora de detectar anomalías en la red, ya que existen múltiples modelos de aprendizaje que pueden ser implementados, dependiendo del contexto en el que se quiera utilizar, cada modelo puede ser eficiente [3]. Este proyecto busca no solo identificar los modelos más efectivos para la detección de ataques, sino también proporcionar una base metodológica robusta para la implementación de medidas preventivas en tiempo real.

CAPÍTULO 1: Generalidades de la propuesta

1.1 Antecedentes

Desde la aparición de las redes inalámbricas hasta la época actual han tenido un gran impacto en la sociedad debido al avance de las nuevas tecnologías puesto que han transformado la manera en cómo las personas se comunican, interactúan entre sí y el acceso a la información ilimitada que un individuo puede poseer [4].

El principio de esta historia se remonta en la primera transmisión que se realizó por ondas electromagnéticas siendo esta, el primer medio inalámbrico ejecutada en el año de 1888 por el físico alemán Rudolf Hertz [4].

Conforme pasaban las décadas iban surgiendo nuevas necesidades llevando así a las tecnologías mecánicas, electrónicas y analógicas a convertirse en procedimientos poco eficientes, por ello entre los años de 1950 y 1970 surgió la denominada revolución digital en donde se vivió el periodo del cambio mejorando y hasta sustituyendo a los procedimientos tradicionales de la época con tecnología digital [5].

En el año de 1997 se establece la primera norma IEEE (Institute of Electrical and Electronics Engineers) para el Wifi el cual definió estándares para las redes WAN (Red de Área Amplia) y fue la base de la tecnología Wifi que hoy en día conocemos, esta norma es conocida como IEEE 802.11 el cual tuvo la función de transmitir inalámbricamente datos por medio de una banda de frecuencia de 2.4GHz proporcionando velocidades de hasta 2Mbps para la transferencia, eso llevó a la conexión sin cableados a dispositivos por medio de ondas de radio como las computadoras. Dicho estándar ha logrado evolucionar con el paso del tiempo obteniendo significativas mejoras y derivando nuevas normas, entre ellas están la 802.11a, 802.11b y este estándar que fue publicado en el año de 1999 fue quien extendió la utilización del Wifi en las empresas, casas e incluso en lugares públicos ya que este estándar tenía velocidades de 11Mbps, un gran avance considerando la época en la que se popularizó [6].

El gran crecimiento que presentan las redes inalámbricas ha hecho posible la conexión e interacción entre dispositivos que son usados diariamente permitiendo

enviar y recibir datos simultáneamente, este desarrollo se lo conoce como IoT (Internet of Things) [5].

El uso de los dispositivos con esta tecnología alcanzó un 54.2% en el año 2020, cifra que se lo considera como un hito histórico ya que recauda un aproximado de 11.7 billones de dispositivos IoT repartidos en el mundo y dicho porcentaje se estima que crecerá para el 2025 llegando a un 75% con 31 billones de equipos IoT en la sociedad [5].

La manera en cómo se propagó la tecnología digital por el mundo permitiendo nuevas tecnologías que prácticamente nos mantiene conectados simultáneamente también trajo consigo una amenaza global, los ataques a las redes llevadas a cabo por individuos con intenciones maliciosas que buscan el beneficio propio comprometiendo y vulnerando la seguridad de las redes cuyo fin es robar información interceptando las redes, ya sea de un área local o un área extensa [7].

Por ello la seguridad de las redes también han ido actualizándose, teniendo una defensa más robusta para proteger los datos accesibles que se encuentran en la red, existen muchos tipos de amenazas, entre ellos los más conocidos son el virus, phishing, suplantación de identidad, troyanos [7].

Para contrarrestar la problemática de las intrusiones en las redes, se desarrollaron técnicas especializadas en la detección de estas anomalías, existe una rama conocida como hacking ético en donde los desarrolladores realizan pruebas de intrusión con fines profesionales sin cruzar la línea de la legalidad, con el fin de exponer las vulnerabilidades de las redes.

Estas técnicas y modelos se centralizan en detectar patrones inusuales en los datos del tráfico de red, existen dos tipos de técnicas conocidas como supervisadas y no supervisadas. Para saber cuál de los dos es óptimo usar, es importante comprender la labor que ejecuta cada una [8].

Las técnicas de aprendizaje supervisado trabajan con datos etiquetados, los cuales viajan por la red para indicar una respuesta correcta en la evaluación del tráfico, es decir, identifica y predice las etiquetas correctas y en caso de no tener registro de alguno, se lo considera como una anomalía [9].

Las técnicas de aprendizaje no supervisado no requieren de etiquetar los datos, mediante un entrenamiento de los modelos que posee, es capaz de encontrar patrones, desde los más comunes que fluyen por la red, hasta los que tienen un margen de cambio, para que posteriormente pueda clasificarlos en grupos y determinar qué comportamiento es inofensivo y qué comportamiento puede presentar un ataque [9].

En el artículo denominado "Learning Intrusion Detection Supervised or Unsupervised" [8], hace un análisis de estas técnicas de aprendizaje y hacen uso de algoritmos que cada uno posee, a continuación, se muestran estos algoritmos usados en su investigación. Para el aprendizaje supervisado se expone seis tipos de algoritmos que son fuertemente utilizados [Fig. 1].

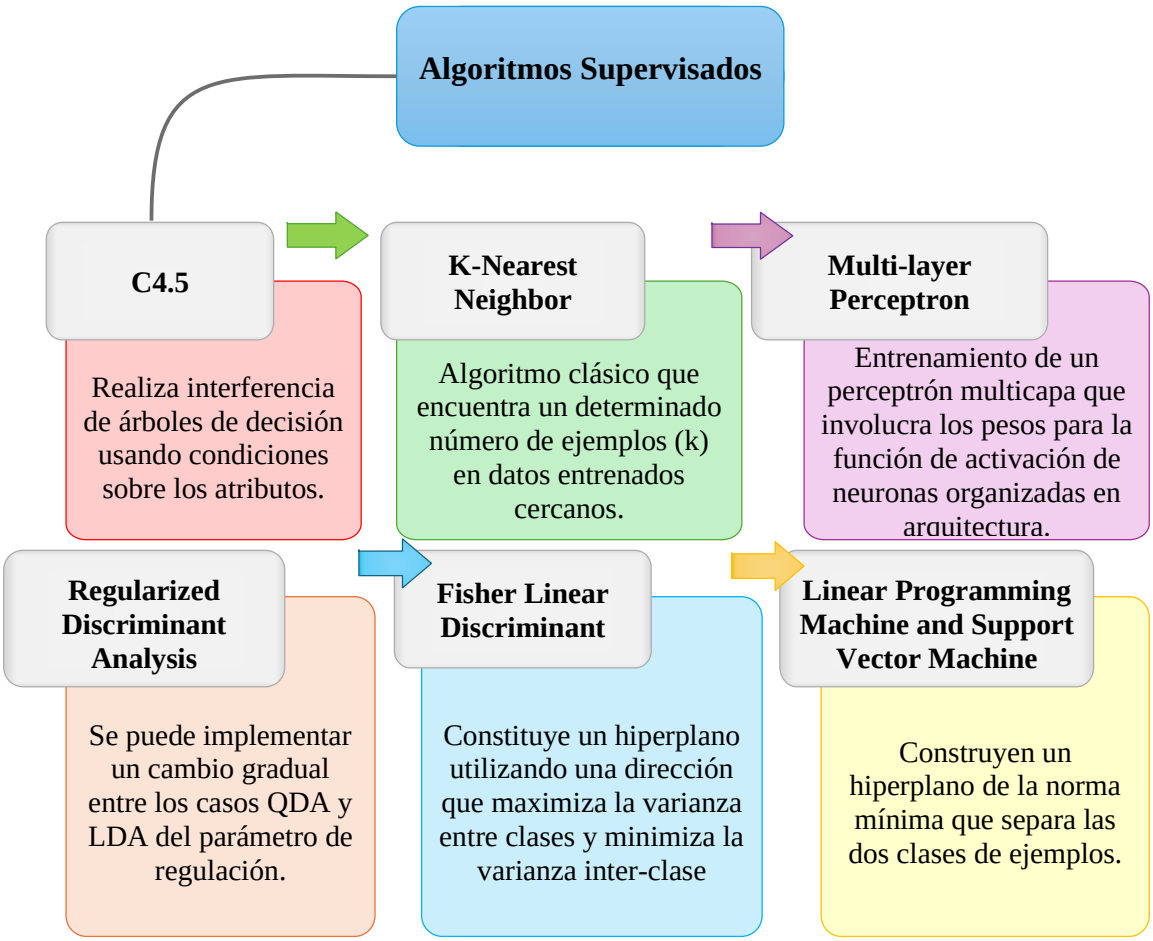


Fig 1. Métodos de algoritmos supervisados

Fuente: Elaborado por autor

En el caso del aprendizaje no supervisado, se presentan cuatro algoritmos relevantes [Fig. 2]:

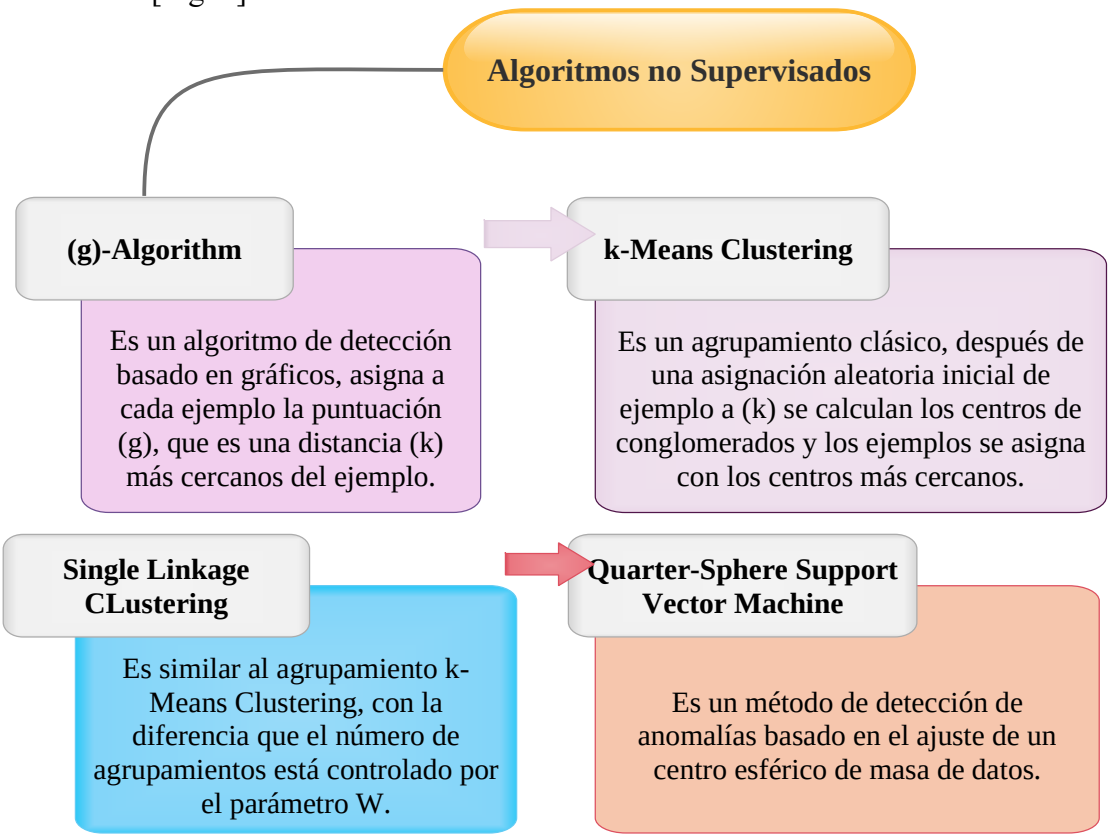


Fig 2. Métodos de algoritmos no supervisados

Fuente: Elaborado por autor

Por ello, se busca implementar un modelo de aprendizaje en cuanto a la detección de actividades sospechosas para poder identificar las vulnerabilidades en una red y prevenir fugas de información futuros permitiendo así un mejoramiento en la seguridad de las redes inalámbricas haciendo uso de técnicas de aprendizaje supervisados.

Para lograr la implementación de los modelos de aprendizaje, se requieren varios factores a considerar, como lo es el dispositivo o dispositivos que servirán de apoyo para la recolección de datos, inicialización de las pruebas, ataques premeditados para identificar las vulnerabilidades de una red, entorno en donde se ejecutarán dichas simulaciones.

1.2 Descripción del proyecto

Mediante el uso de herramientas de pentesting de las nuevas generaciones

se pretende evaluar la seguridad de las redes inalámbricas con el fin de encontrar vulnerabilidades expuestas a ataques.

En este trabajo, se propone el uso de modelos de aprendizaje para la detección y prevención de ataques en redes inalámbricas mediante el uso de herramientas de pruebas de penetración. Se recopilará la información encontrada en la web, libros y artículos relacionados al tema que complementen la investigación a realizar, para analizar las tendencias y avances en la detección de ataques de red utilizando técnicas de aprendizaje automático.

Se analizarán las técnicas de aprendizaje supervisado y no supervisado para detectar patrones en el tráfico de red y detectar la presencia de ataques de red. También se utilizará el análisis de comportamiento para detectar actividades maliciosas en la red y prevenir ataques futuros.

Se realizarán experimentos en un entorno de prueba utilizando herramientas de pentesting de la nueva generación para validar la efectividad de los modelos de aprendizaje propuestos. Los resultados de esta investigación pueden ser útiles para mejorar la seguridad en redes inalámbricas y prevenir ataques maliciosos tanto en entornos empresariales como en entornos domésticos.

Para ello se considera un equipo de esta categoría el cual cumplirá con esta función, se pretende utilizar uno de los equipos de pentesting de nueva generación para la implementación de este trabajo ya que es un dispositivo que entra en una categoría que permite la realización de ataques de red en entornos de prueba y mejora de la seguridad en redes inalámbricas.

El entorno en donde se realizará la ejecución del proyecto será en el laboratorio de telecomunicaciones, ahí se tiene que implementar una red privada, la cual servirá para realizar las distintas pruebas de pentesting.

1.3 Identificación y justificación del problema

En muchas ocasiones las redes inalámbricas sufren por la falta de un sistema de monitoreo autónomo que les permitan la detección de intrusos en la red y actividades de carácter maliciosos que vulneran la seguridad de esta con accesos no

autorizados, facilitando a los atacantes la extracción de información valiosa en ese lapso que el sistema queda vulnerable y sin supervisión.

El desarrollo del Ecuador en el ámbito tecnológico con respecto a las telecomunicaciones se disparó desde el año 2020 debido a la pandemia ya que, en consecuencia, el internet tuvo un papel principal en la vida cotidiana de los ecuatorianos, desde las clases virtuales hasta el teletrabajo, por lo que a nivel nacional el porcentaje de los clientes que contrataban servicio de internet en sus hogares aumentó un 7.7% [10].

De acuerdo con los datos de la ARCOTEL (Agencia de Regulación y control de telecomunicaciones), sólo en la provincia de Santa Elena se crearon 46.055 cuentas y usuarios del servicio de acceso a internet en el mes de marzo del año 2022 [11].

Al existir este incremento de usuarios, las víctimas de los ciberdelincuentes aumentan por los equipos conectados a la red que están propensos a diferentes tipos de amenazas significativas tales como ataques, robo, inyección de malware en las redes inalámbricas, entre otros. Y que, según el tipo de configuración, su seguridad puede ser no tan eficiente ya que su arquitectura se basa en reglas estáticas o predefinidas lo cual genera un problema puesto que no se adaptaría a las técnicas de ataque más avanzadas [10]. Para poder hacerle frente a un problema tan grande se pretende hacer uso de herramientas que permitan realizar pruebas de penetración, este tipo de pruebas tienen como función principal identificar las vulnerabilidades expuestas en un entorno de seguridad y categorizar el nivel de amenaza que representa, ya que según un artículo de la revista IEEE, en relación a las aplicaciones web, cerca del 91% son propensos a ser violentados y en donde un 84% de los datos confidenciales extraídos pueden ser divulgados [12].

De acuerdo con el tipo de herramienta que se pretende utilizar, las técnicas o modelos de aprendizaje tienen que ser capaces de identificar comportamientos o patrones extraños en el tráfico de la red de manera automática y que puedan aprender de sí mismas para mejorar la eficiencia de detección de atacantes generando una respuesta rápida a estos eventos.

1.4 Objetivos del Proyecto

1.4.1 Objetivo General

Aplicar un modelo de aprendizaje supervisado para la detección y prevención de ataques, utilizando herramientas de pentesting de nueva generación mediante la realización de experimentos en un entorno de prueba.

1.4.2 Objetivos Específicos

- Distinguir los tipos de ataques de red que pueden ser realizados mediante el uso del dispositivo de pentesting de nueva generación para entornos inalámbricos.
- Identificar los principales modelos de aprendizaje supervisado para implementar en el proyecto, a través del análisis de la información bibliográfica.
- Seleccionar uno o varios de los modelos de aprendizaje supervisado que cumpla con los requisitos del proyecto y su aplicación en el desarrollo de soluciones.
- Demostrar la exitosa implementación del modelo de aprendizaje seleccionado y la validación de su efectividad.

1.5 Alcance del Proyecto

Para el siguiente proyecto, la propuesta se centra en la implementación de un modelo de aprendizaje automático supervisado, en el campo de machine learning existen gran variedad de estos aprendizajes por lo que, en esta ocasión se trabajará con la técnica de aprendizaje supervisado con el fin de no extender la complejidad del proyecto ya que esto conllevaría a manejar muchos recursos e información.

Para poder llevar con éxito este proyecto, se requiere de varias componentes y recursos, en este caso se usará un equipo de pentesting de nueva generación el cual contribuirá con las pruebas a realizar y el montaje de una red cuyo fin es el mismo, el análisis del tráfico de datos y las pruebas de penetración a esa red. Estas pruebas se realizarán en el laboratorio de Telecomunicaciones de la Universidad Estatal Península de Santa Elena el cual cuenta con varios equipos para la implementación de una red.

Cada fase del proyecto está documentada en un cronograma indicando los capítulos a realizar en un tiempo límite, la primera parte del proyecto consta de este lapso, se extiende desde el inicio del periodo 2023-1 del séptimo semestre y continúa hasta culminar el octavo semestre de la carrera, en donde se debe completar las actividades dispuestas por el mismo docente el cual consta de las presentaciones o avances periódicas por cada semana al profesor de la materia y exposiciones de los avances efectuados.

1.6 Metodología

- **Investigación aplicada**

Este proyecto se enmarca en una investigación aplicada, cuyo principal objetivo es la resolución de un problema práctico mediante el desarrollo de una solución innovadora en el ámbito de la seguridad en redes. La investigación aplicada se centra en utilizar conocimientos y teorías preexistentes, específicamente en el campo del aprendizaje automático, para diseñar e implementar un sistema de detección de ataques que sea efectivo en entornos reales.

- **Investigación cuantitativa**

El enfoque de esta investigación es cuantitativo, ya que se basa en la recopilación y análisis de datos numéricos provenientes del tráfico de red para entrenar y evaluar modelos de aprendizaje supervisado. Se emplean métricas estadísticas para medir el rendimiento de los algoritmos de detección de ataques, como la precisión, la sensibilidad, y la tasa de falsos positivos.

- **Método experimental**

Además, el proyecto utiliza un método experimental, en el que se simulan distintos escenarios de ataques de red para probar la efectividad del sistema propuesto. Este enfoque experimental permite ajustar los parámetros de los modelos de aprendizaje supervisado en función de los resultados obtenidos, optimizando su capacidad de detección.

Finalmente, se busca la aplicabilidad práctica de los resultados, ya que el sistema desarrollado no solo tiene valor teórico, sino que está diseñado para

integrarse en infraestructuras de red reales, brindando una herramienta útil para mejorar la seguridad y la capacidad de respuesta ante ataques cibernéticos.

Para la ejecución de la propuesta metodológica del proyecto, se enfatiza el tema de la investigación e implementación que se requiere, en base a esto, en el siguiente diagrama se muestra el flujo del desarrollo de la propuesta [Fig. 3]:

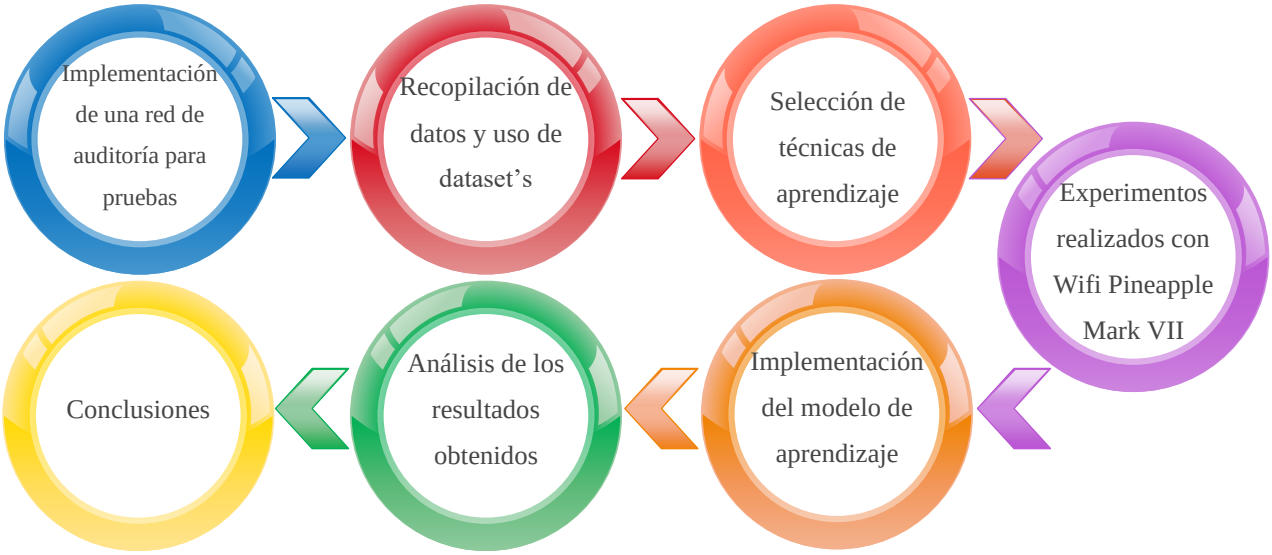


Fig 3. Cronología del desarrollo de la propuesta

Fuente: Elaborado por autor

- **Implementación de una red de auditoría para pruebas:** Al tener en claro los conceptos generales expuestos en el capítulo 2 mediante una revisión exhaustiva de la literatura científica disponibles en la web, revistas, artículos, libros, entre otros, se procede a la implementación de una red inalámbrica para las pruebas de pentesting que se pretende ejecutar mediante el uso del equipo.
- **Recopilación de datos y uso de dataset's:** Se recopilará la información extraída del tráfico de red en el entorno de prueba utilizando softwares que permitan dicha acción como el Wireshark y usando el propio equipo Wifi Pineapple Mark VII, para usarlos en los experimentos.
- **Selección de técnicas de aprendizaje:** De acuerdo con la información recopilada de varios artículos y revistas, la técnica de aprendizaje seleccionada será la supervisada puesto que se considera adecuada para la detección de anomalías mediante el etiquetado de datos del tráfico de la red

inalámbrica, y posterior a esto se seleccionará un algoritmo de aprendizaje supervisado óptimo para la identificación de ataques.

- **Experimentos o pruebas realizados con WiFi Pineapple Mark VII:** Mediante el uso del dispositivo Wifi Pineapple Mark VII, se propone realizar múltiples ataques hacia la red de prueba para entrenar los algoritmos del modelo de aprendizaje supervisado.
- **Implementación de modelos de aprendizaje:** Se realizará un análisis entre los algoritmos para verificar quién obtuvo un mejor resultado y así evaluar la efectividad del modelo mediante las pruebas de penetración.
- **Evaluación de resultados:** Se evaluarán los resultados obtenidos de los experimentos para determinar la eficacia de los modelos de aprendizaje implementados en la detección y prevención de ataques a la red.
- **Conclusiones y recomendaciones:** Una vez analizado los resultados, se elaborarán conclusiones y recomendaciones en donde se discutirán las implicaciones de los hallazgos para la mejora de la seguridad en redes inalámbricas.

1.7 Retos y limitaciones

El proyecto está enfocado en la representación de una red dentro de un entorno controlado para no rebasar los límites de la legalidad de la práctica del pentesting con los equipos disponibles, dentro de la práctica se espera que se presenten desafíos a la hora de implementar los modelos de aprendizaje, de acuerdo con las investigaciones realizadas, los algoritmos de aprendizaje supervisado y en el contexto de su implementación puede variar de acuerdo a las necesidades de la red, por ello se plantea este desafío, proporcionar una adecuada implementación de estos algoritmos será el reto a cumplir.

1.8 Análisis de la información bibliográfica

La importancia de la problemática se da por las vulnerabilidades que presentan las redes, al ser objetos de robos de información es importante tener una contramedida para evitar estos riesgos. Por ello es importante determinar el tipo de modelo y técnicas a usar a lo largo de la investigación, mediante las herramientas de estudio e información recopilada de la web y artículos se hace un análisis de los

artículos relacionados a la seguridad de las redes y modelos de aprendizaje propuestos en cada investigación.

En el artículo titulado “A comparative Analysis of Supervised and Unsupervised Models for Detecting Attacks on the Intrusion Detection Systems”[13], hacen mención del análisis de los modelos supervisados y no supervisados realizando una comparativa del rendimiento de dichos modelos aplicados a un sistema de detección de intrusiones.

Los autores hacen uso de un conjunto de datos para dicha evaluación, el cual se encontró que los modelos supervisados son más apropiados en términos de precisión y exhaustividad en comparación a los modelos no supervisados[13].

Y una de las técnicas de este modelo denominado árbol de decisión fue el más apropiado en la detección de ataques por lo que, para los autores los modelos supervisados son más efectivos en cuanto a la detección de ataques siendo útiles en futuros sistemas de detección de intrusiones en la seguridad informática.

En otro artículo denominado “A Hybrid Approach for Network Intrusion Detection” [14] interesante relata un enfoque híbrido en cuanto al sistema de detección de intrusos en redes mediante las técnicas de aprendizaje supervisado, aquí presentan varias técnicas, entre ellas las que son basadas en firmas, técnicas basadas en anomalías y en el aprendizaje automático, la técnica más relevante en la que se hace énfasis es en el aprendizaje supervisado presentando una arquitectura híbrida el cual une dos algoritmos de este modelo denominados “algoritmo Naive Bayes” y “algoritmo K-Nearest Neighbor” [14].

Este artículo redacta con detalle la implementación de la arquitectura híbrida junto con los pasos realizados en la detección de intrusos, en donde los resultados experimentales presentan que esta arquitectura es bastante efectiva en la detección de intrusiones, teniendo una tasa de detección elevada en comparación a otros enfoques que se basan en el mismo aprendizaje, aunque no está exento de los errores, presenta una tasa muy baja con los falsos positivos, concluyendo que esta técnica es menos propensa a identificar cualquier actividad apacible como un intruso[14].

En el artículo “A machine learning-based intrusion detection for detecting internet of things network attacks” [15], proponen un sistema de detección que se basa en un aprendizaje autónomo para detectar ataques en las redes de IoT utilizando una combinación de técnicas de preprocesamiento de datos junto con algoritmos de aprendizaje autónomo supervisado para la identificación de patrones de comportamiento anómalos presentes en el tráfico de red con el fin de detectar posibles ataques, los datos son recopilados gracias al monitoreo de la red conectado a un enrutador IoT.

Dicho estudio experimental sucede en un entorno de red simulado con lo cual se ejecutan varios ataques para evaluar la eficiencia del sistema de detección propuesta en el artículo, logrando como resultado una detección de varios tipos de ataques a la red IoT exitosos mediante la detección de intrusiones basado en el aprendizaje automático supervisado. El artículo logra destacar el uso de técnicas de aprendizaje automático supervisado en la detección de intrusiones para mejorar la seguridad en redes IoT [15].

En el artículo titulado “An Effective Intrusion Detection System using Supervised Machine Learning Techniques”[3] se explican el uso de las técnicas de aprendizaje supervisado para los sistemas de detección de intrusiones para el mejoramiento de la seguridad de la red comparando tres tipos de algoritmos de aprendizaje supervisado:

- Arboles de decisión
- Redes neuronales
- Máquina de Vectores de Soporte (SVM)

El estudio de este análisis concluye que el uso de SVM fue más efectivo considerando la precisión de detección de ataques que realizó [3]. Los autores de este artículo destacan la efectividad del aprendizaje supervisado en la detección de intrusos y señalan un enfoque híbrido para la mejora del preprocesamiento de datos mejorando así la precisión de los sistemas de detección.

En el artículo “Learning Intrusion Detection Supervised or Unsupervised” [8] se hace una comparación de las técnicas de detección de intrusos supervisado y no supervisado destacando las ventajas y desventajas que conllevan ambos tipos,

que, como los artículos anteriores, dejan como sugerencia el enfoque híbrido con el fin de mejorar la precisión en las detecciones.

Para el análisis del aprendizaje supervisado, en el artículo se discute la utilización de un conjunto de datos etiquetados para el entrenamiento de un modelo de detección de intrusos y que dicho aprendizaje es apto para detectar ataques conocidos, sin embargo, para ataques desconocidos este aprendizaje deja de ser efectivo.

Por otro lado, se menciona al aprendizaje no supervisado como un conjunto de datos sin etiquetar para entrenar un modelo de detección de intrusiones lo cual es muy factible detectar patrones anómalos en los datos indicando un ataque, aunque también tiene sus falencias ya que puede generar muchos falsos positivos haciéndolo una opción menos fiable para la detección de ataques conocidos [8].

La información revisada del artículo “Network Intrusion Detection using Supervised Machine Learning Technique with Feature Selection” [16] presenta un enfoque hacia el aprendizaje automático supervisado en relación con la clasificación del tráfico de red siendo estas de carácter benévolo o malicioso, aquí usan algoritmos de aprendizaje automático como el SVM (Support Vector Machine) y el ANN (Artificial Neutral Network). También lo que hace el documento es señalar las características más importantes el cual logra hacer una mejora en la tasa de éxito. Mediante una tabla realizado por los autores de ese documento, presentan una tabla resumida el cual consta de un número de características de entrada y salida para cada técnica [16].

Otro artículo interesante es el “A comparative study of supervised Machine Learning classifiers for Intrusion Detection in Internet of Things” [17] el cual menciona de manera generalizada los retos de seguridad asociados al crecimiento exponencial de las redes informáticas e internet, también el estudio realizado se centra en el desarrollo de un sistema de detección de intrusos utilizando el aprendizaje automático como técnica de clasificación, su diseño consiste en la identificación de intrusos multiclase en el entorno IoT.

Dentro de este artículo también se discute la implementación y comparación de varios clasificadores de aprendizaje automático, los cuales son: KNN (K-Nearest Neighbour), SVM (Support Vector Machine), NB (Naive Bayes), RF (Random Forest), DT (Decision Tree) y SGD (Stochastic Gradient Descent) [17].

El artículo llamado “Comparison of Machine Learning and Deep Learning Models for Network Intrusion Detection Systems” [18] es un trabajo de investigación cuyo objetivo es comparar el rendimiento de diferentes modelos de aprendizajes automáticos junto con el aprendizaje profundo orientado a un sistema de detección de intrusos en una red, el enfoque principal que se puede notar mediante el análisis del artículo es en el conjunto de datos en la detección de intrusiones de Coburg (CIDDS) el cual evalúa dichos modelos tomando en consideración la exactitud, precisión, recuperación y puntuación de la misma [18].

Los resultados al analizar este estudio muestran que la clasificación y árbol de regresión fue quien obtuvo mejores resultados que otros modelos implementados del aprendizaje profundo con una medida del 99, 31% de precisión media y una desviación estándar de 0,15.

CAPÍTULO 2: Fundamentación Teórica

2.1 Marco contextual

Dentro del campo de las telecomunicaciones, las comunicaciones inalámbricas y la seguridad de las redes es un tema que ha ganado terreno debido al crecimiento de los avances tecnológicos que conforme al aumento de dispositivos inalámbricos surgen nuevas amenazas que ponen en peligro la información de los usuarios.

En este contexto, el desarrollo de prácticas que sobrelleven esta problemática es de suma importancia ya que capacita a la comunidad estudiantil a contrarrestar ciberataques, por lo tanto, el entorno en donde se desarrollará los métodos prácticos de la demostración del funcionamiento del proyecto será en el laboratorio de telecomunicaciones de la Universidad Estatal Península de Santa Elena (UPSE) perteneciente a la provincia de Santa Elena y que a su vez se encuentra ubicada en el cantón La Libertad, con la siguiente propuesta se busca implementar un método de aprendizaje supervisado aplicando herramientas de pentesting de nueva generación cuya finalidad busca contribuir al mejoramiento de la seguridad en las redes inalámbricas.

2.2. Marco conceptual

2.2.1. Fundamentos teóricos de la seguridad informática

Dentro del campo de la seguridad informática es importante conocer varios fundamentos teóricos que permitan la comprensión y dimensionar todo lo que se quiere abarcar con este tema, esto hace referencia a un conjunto de técnicas y medidas utilizadas para proteger la información contra amenazas o ataques maliciosos de fuentes externas, en términos generales y como parte de la introducción, este concepto trata de un conjunto de medidas que logran impedir operaciones sin autorización dentro de un sistema o red local. El objetivo de esta seguridad es la protección de la información, ya sea de una persona, entidad pública u organización privada, todos cuentan con el mismo objetivo y es respaldar la seguridad de sus datos contra operaciones externas que no cumplan con las autorizaciones debidas del sistema [19].

Los sistemas de seguridad han ido evolucionando con el paso del tiempo permitiendo estructuras más robustas en sus aplicaciones, dicho tema es crucial en esta era digital el cual ha involucrado un sinnúmero de estrategias o técnicas que garanticen la protección de información confidencial de suma importancia [19].

Para entrar más a fondo en este tema es importante recordar varios temas básicos que servirán de apoyo para la comprensión y análisis de estos temas.

2.2.2 Seguridad

La seguridad tiene muchas aplicaciones y en términos generales, puede decirse que esta idea procede de la palabra latina “securitas” quien se centra en la propiedad de la seguridad o en mejorar la propiedad de cualquier cosa en la que exista riesgo, daño o evaluación de riesgos. Algo seguro es algo sólido cierto e innegable, en consecuencia, la seguridad puede considerarse como “algo dado o una acción con certeza” [20].

Ampliando este sentido, la seguridad se refiere a la protección de algo o alguien frente a posibles riesgos, peligros o amenazas. Esto implica asegurarse de que solo aquellos con permisos pueden acceder a información confidencial, que los datos no pueden cambiarse o corromperse sin permiso, de que los sistemas están disponibles y en correcto funcionamiento cuando se lo requiera [20].

2.2.3 Información

Otro tema importante por mencionar es la información, dentro del contexto general puede decirse que es el acto de presentar informe basados en noticias sobre algo y declarar los conocimientos adquiridos. Esta palabra también es utilizada para hacer referencia a los conocimientos adicionales que se añaden a los ya existentes en un área determinada, por lo que también se los denomina así a estos conocimientos [20].

También es posible definir el conocimiento en términos de cómo causó o contribuyó a una ocasión o circunstancia que fue creada por una parte en la mente de la gente y apoyada por otra parte de alguna manera, ya sea información expresada por televisión, periódicos, radios, entre otros [20].

La información como tal puede encontrarse en diversas formas, entre ellas son:

- De manera física mediante la escritura en papel o por impresión.
- Almacenamiento por medios electrónicos.
- Enviada por correo electrónico u otros medios del mismo tipo.
- Presentada de forma visual o mencionada en un diálogo.

2.2.4 Seguridad informática, de información y ciberseguridad

Existen varios tipos de seguridad fundamentales en los que se puede enfocar una persona y que se aplican en diferentes campos y contextos, en esta ocasión se mencionan 3 tipos de seguridad y estas son:

- Seguridad informática
- Seguridad de la información
- Ciberseguridad

Se pueden relacionar entre sí, pero conservan un contexto diferente, en la figura 4 se explica el concepto básico de cada seguridad, en donde se puede observar las diferencias que poseen cada una:

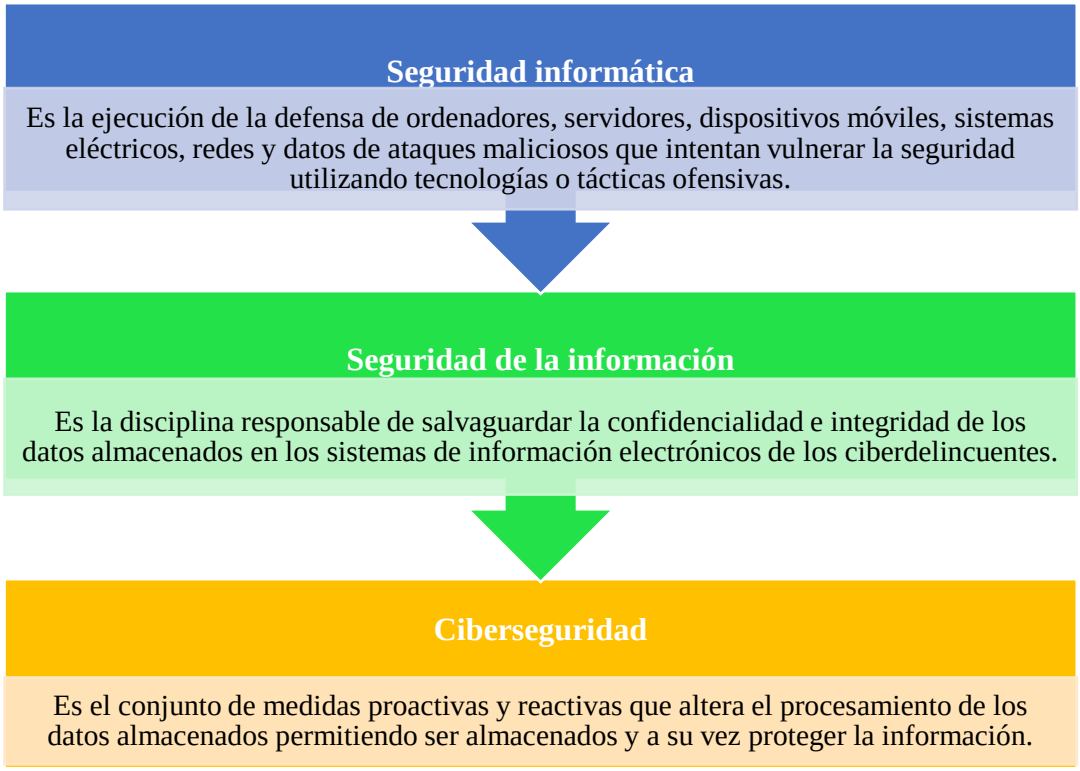


Fig 4. Tipos de seguridades

Fuente: Elaborado por autor

2.3 Redes inalámbricas

Al definir el significado de una red inalámbrica, surge el concepto general en donde se explica que es una conexión por ondas electromagnéticas el cual permite intercambiar información sin utilizar un canal físico como el cableado estructurado. Siempre que se encuentren en la misma zona de red, los dispositivos remotos pueden conectarse fácilmente vía Wifi, bluetooth, infrarrojos y microondas, estos son algunos ejemplos de las tecnologías de comunicación inalámbrica que pueden utilizarse en estas redes [21].

La popularidad de las redes inalámbricas ha crecido gracias a su adaptabilidad y sencillez por la facilidad en su uso siendo implementadas en una amplia gama de dispositivos móviles y computadoras portátiles. Son perfectas tanto para entornos residenciales como comerciales, ya que permiten a los usuarios conectarse a internet y a otros dispositivos sin necesidad de cables siendo esta su principal ventaja. Las redes inalámbricas son perfectas para entornos empresariales porque pueden ampliarse y cubrir áreas enormes. En la figura 5 se puede observar cómo se comporta una red inalámbrica [21].

Fig 5. Esquema de una red inalámbrica ideal

Fuente: Elaborado por autor

Sin embargo, las redes inalámbricas incluyen fallos de seguridad que pueden ser utilizados por los atacantes para acceder a datos privados o llevar a cabo actos nefastos. Las causas de estas vulnerabilidades pueden ser diversas, desde la falta de

cifrado en su tráfico de red, contraseñas poco seguras, hasta malas configuraciones en el sistema o la falta de actualizaciones en su seguridad dentro de la infraestructura [22].

2.3.1 Vulnerabilidades en las redes inalámbricas

Como se menciona en el tema anterior, las redes inalámbricas tienden a presentar fallos o bugs en sus sistemas y quedan expuestos a ataques, este concepto se lo conoce como vulnerabilidades en la red ya que son puntos débiles o fallos en la seguridad dentro de sus sistemas de comunicación, dichas vulnerabilidades suelen ser aprovechados por los atacantes que buscan acceder a información confidencial de una empresa u organización con el fin de malograr la infraestructura a su antojo [7]. En la figura 6 se muestran distintos tipos de vulnerabilidades, entre estas se consideran las más comunes:

Fig 6. Vulnerabilidades principales

Fuente: Elaborado por autor

Sin embargo, estas vulnerabilidades pueden detectarse a tiempo para poder tomar medidas de prevención utilizando técnicas y herramientas de seguridad como el uso de pentesting, la detección de intrusiones y la supervisión del tráfico de red. Así mismo, las otras medidas a tomar van desde el uso de contraseñas más seguras hasta el cifrado de las mismas [7].

2.4 Ataques de red

Se define a un ataque de red como un intento malicioso de acceder a un servicio en línea con la interrupción de su funcionamiento [3], dichos ataques aprovechan de las capacidades limitadas dentro de los recursos de la red, tales como la infraestructura de algún sitio web para ser vulnerado desbordando su capacidad y evitando su funcionalidad de manera correcta [2]. Estos ataques tienen varios objetivos, entre ellos robo de información confidencial, daño a la reputación de una empresa u organización grande o la interrupción al acceso de servicios online.

2.4.1 Tipos de ataques de red

De acuerdo con la información bibliográfica consultada, se logra identificar una variedad de ataques de red que son perjudiciales para las organizaciones grandes y que son comúnmente usadas por atacantes del ámbito, estas son:

- **Ataques de denegación de servicios (DoS)**

  El objetivo de estos ataques es sobrecargar un servicio o infraestructura en la red como por ejemplo un sitio web o también un servidor para hacerlo inaccesible a los usuarios legítimos. Los ataques DoS pueden ser llevados a cabo por un único dispositivo o por numerosos dispositivos que trabajan juntos para lanzar un ataque de denegación de servicios distribuido (DDoS) [23].
- **Ataques de intrusión**

  Este tipo de ataque pretende comprometer la seguridad de una red o sistema de información mediante la intrusión sin autorización, para acceder a los sistemas y obtener información confidencial, los atacantes pueden emplear técnicas como la explotación de fallos de seguridad al igual que el uso de contraseñas deficientes o débiles, también el robo de credenciales de acceso [24].
- **Ataques de phishing**

Por lo general, estos ataques están basados en la ingeniería social con el cual pretenden engañar a personas para que revelen su información privada como por ejemplo contraseñas o claves de sus tarjetas de crédito. Los atacantes por lo general suelen enviar correos electrónicos de manera virtual que contienen mensajes falsos que parecen legítimos pidiendo a las personas que verifiquen o confirme sus datos personales en sitios web fraudulentos [24].

- **Ataques de inyección de código**

  Estos ataques instalan códigos dañinos en las aplicaciones web aprovechando sus puntos débiles dentro de su infraestructura. Los atacantes suelen utilizarlo para ejecutar comandos u órdenes no autorizadas para obtener datos o conseguir un control total de la aplicación [24].

- **Ataques de spoofing**

  Estos ataques consisten en alterar la dirección IP de un paquete de datos para que parezca que procede de una fuente fiable. De este modo los atacantes pueden eludir las medidas de seguridad y llevar a cabo acciones destructivas de forma encubierta [25].

El uso de firewalls, sistemas de detección de intrusos y actualizaciones periódicas del software para corregir vulnerabilidades conocidas son solo algunas de las medidas de protección y mitigación que pueden aplicarse para defenderse de estos ataques. La prevención de los ataques a la red también depende en gran medida de la educación y concienciación de los usuarios sobre los procedimientos de seguridad en línea [26].

2.5 Ataques pasivos en las redes inalámbricas

Este tipo de ataque es conocido debido a la complejidad en cuanto a su detección ya que el atacante no realiza ninguna modificación o alteración al momento en que observa o monitorea la información transmitida, por lo que, para evitarlo es importante realizar la técnica de cifrado de información. Dentro de los ataques pasivos se encuentran los siguientes:

- **Ataques de tipo espionaje**

  Este es un tipo de ataque en donde el atacante se aprovecha de las conexiones de red inseguras o con falta de contraseñas para filtrar datos mientras haya

comunicación, estos individuos operan buscando conexiones que sean débiles entre los servidores y sus clientes, por ejemplo, las que no se encuentran encriptadas, dispositivos o software desactualizados o los que guardan un programa maligno instalado. Al lograr vulnerar dichas conexiones, los atacantes logran interceptar paquetes de datos que viajan por la red leyendo el tráfico o medio electrónico que no se encuentre cifrado [23].

De acuerdo con el artículo [7], los atacantes de espionaje de redes también pueden lanzar ataques DoS o de otras categorías, en las redes deficientes de una institución universitaria.

- **Descubrimiento de SSID**

  En una red inalámbrica, este tipo de ataque se define como la identificación y visualización de la identificación pública por nombre de una WLAN (Wireless Local Area Network) y quien se diferencia de las demás redes que se encuentren en su entorno. Las siglas SSID significan Service Set Identifier y es un conjunto de valores alfanuméricos que permite identificar las configuraciones correctas para la conexión a la red de dispositivos. El SSID puede ocultarse para que la red no sea visible dentro de las listas de las redes en la zona, este proceso se lo conoce como red Wifi oculta, sin embargo, esta acción no garantiza una seguridad real ya que los atacantes pueden detectarla mediante un análisis y escaneo de paquetes dentro del área [27].

- **Ataques de descubrimiento de contraseña**

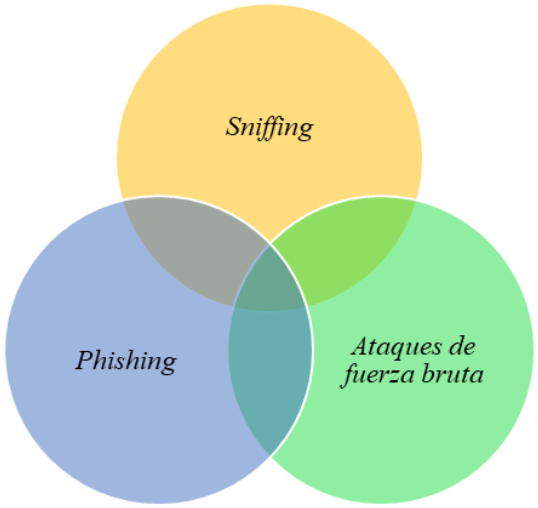


Fig 7. Técnicas para descubrimiento de contraseñas

Fuente: Elaborado por autor

Este es un tipo de ataque que abarca varios tipos utilizados por los atacantes que aplican técnicas para obtener contraseñas de los usuarios cuyo objetivo es acceder a su información personal o confidencial para actividades malintencionadas, entre las técnicas utilizadas constan el tipo sniffing de red, ataques de fuerza bruta y el phishing, como se muestra en la figura 7.

- **Ataques de Sniffing**

  El ataque de sniffing es considerada una técnica capaz de "escuchar" el tráfico de datos dentro de la red, es decir que los atacantes utilizan estas herramientas para monitorear el tráfico y capturar los paquetes de datos transmitidos por los dispositivos de la red. Al monitorear la red determinan varios parámetros como lo son las direcciones MAC (Media Access Control) de los dispositivos conectados, direcciones IP o tasas de transmisión. Para que este ataque pueda ejecutarse, el atacante debe contar con tarjetas externas inalámbricas al igual que programas específicos requeridos. El sniffing no solo es pasivo, también es un tipo de ataque sniffing activo quien busca inundar la memoria de direcciones CAM del switch desviando el tráfico legal hacia otros puertos [26].

2.6 Ataques activos en las redes inalámbricas

A diferencia de los ataques pasivos, este si ejecuta una modificación dentro del flujo de datos que se transmite en la red, ya sea cambiar la corriente de transferencia de datos o también generar un falso flujo de datos. Para estos tipos de ataques existen varios de los cuales se encuentran los siguientes:

- **Ataque Mac Spoofing**

  Este tipo de ataque sirve para cambiar la dirección MAC que tiene un dispositivo de fábrica y que se encuentra conectado a la red. Esta dirección es un tipo de identificación único, que se encuentra en el hardware del dispositivo al igual que de manera física, por lo que para los usuarios no es posible ser cambiado, sin embargo, existen herramientas que permiten modificar esta dirección y supone un problema ya que se puede utilizar para suplantar el identificador de la red [27].

  Ahora, el funcionamiento de esta técnica consiste en modificar la dirección MAC de un dispositivo para hacerse pasar por otro, permitiendo a los atacantes eludir las medidas de seguridad de la red como lo son el filtrado de las

direcciones o autenticación de la MAC, también con esto se puede evadir restricciones de los servicios impuestas por la red para cometer actos ilegales.

- **Ataque DNS Spoofing**

  Este tipo de técnica también se la conoce como envenenamiento de caché DNS el cual consiste en inyectar datos corruptos en la caché del servidor, DNS (Domain Name System) son todas aquellas direcciones IP de páginas webs en donde una persona solicita su acceso haciendo uso de los navegadores web. Este ataque tiene como objetivo engañar a los usuarios dirigiéndolos a una página ficticia haciéndose pasar como una página legal sin tener conocimiento de que la dirección IP fue alterada dándole al atacante un control total de los datos que el usuario digite dentro de dicha página fraudulenta [23].

- **Man in the Middle**

  El ataque de MitM (Man in the middle) o en español Hombre en el medio, es una de las técnicas más utilizadas para la ejecución de los ataques cibernéticos ya que consiste en interceptar y hasta modificar las comunicaciones que existe entre dos individuos que se estén comunicando entre ellos, como su nombre mismo lo indica, el atacante si sitúa a la mitad de ambos puntos para poder leer y hasta introducir mensajes dentro de la comunicación en donde las personas se convierten en víctimas de robo de información personal para cometer actos maliciosos [24].

- **Sesión Hijacking**

  El ataque hijacking o secuestro de sesiones, es un tipo de técnica el cual consiste en que los atacantes obtienen el acceso a la autenticación de sesión de algún sitio web de su víctima para poder robar su identidad [24]. También es conocido como cookie hijacking y su función se la observa en la figura 8:

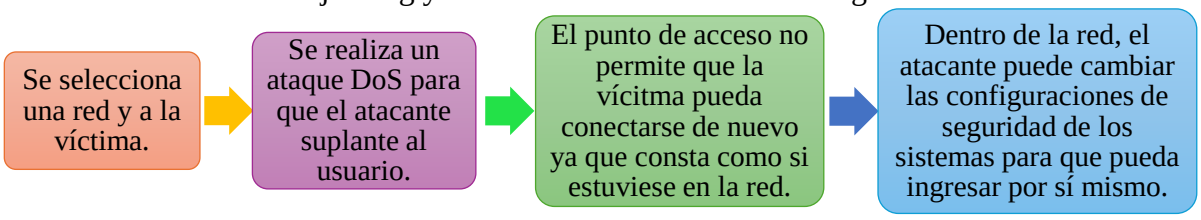


Fig 8. Funcionamiento de la sesión hijacking

Fuente: Elaborado por autor

- **Puntos de acceso no autorizados**

También conocidos como Rogue Access Point, se refiere a un punto de acceso inalámbrico que no tienen la autorización del administrador de la red, es decir que fue agregado de manera ilegal por algún atacante, este tipo de ataque es una amenaza grave para la seguridad de las redes y que tienen el poder de permitir que personas malintencionadas puedan ejecutar diferentes tipos de escáner que buscan las vulnerabilidades de la red, todo esto de manera remota ya que no se necesita estar presente en el lugar atacado. Para poder contrarrestar este ataque es importante utilizar herramientas que detecten estos puntos de accesos no autorizados o bien la segmentación de la red y autenticar a los usuarios [5].

2.7 Sistemas de detección de intrusos (IDS)

Un Sistema de detección de intrusos o por sus siglas en inglés IDS (Intrusion Detection System), son aquellos sistemas cuya función principal es identificar, detectar y prevenir actividades no autorizadas dentro de la red, un activo o sistema informático [22].

Los activos por lo general son los dispositivos de uso personal de una organización o de la misma red. El mal uso de estos puede considerarse un ataque o sabotaje por lo cual se definen reglas que aportan a la configuración de los IDS [24]. Los IDS pueden encontrarse implementados en los firewalls, el término WIDS hace referencia a los IDS conectados al Wifi y estos al momento son alternativas de paga ya que no existen alternativas libres desarrolladas al momento [24].

2.7.1 Funcionamiento de los IDS

Como ya se mencionó anteriormente, su función está basada en detectar un comportamiento de algún usuario o dentro del tráfico de la red que sea considerado una actividad sospechosa o fuera de lugar. Dentro de las conexiones Wifi los paquetes suelen viajar por la red sin un cifrado adecuado y también pueden ser visibles por usuarios que se encuentren conectados a esa red, por lo que es necesario para organizaciones grandes querer implementar un IDS vía Wifi, para ello se necesitará establecer una red inalámbrica para analizar quien podrá controlar las diversas redes que se encuentren alrededor de la principal [27].

Dentro de la estructura y funcionamiento de un IDS se diferencian 3 partes fundamentales como se observa en la figura 9:

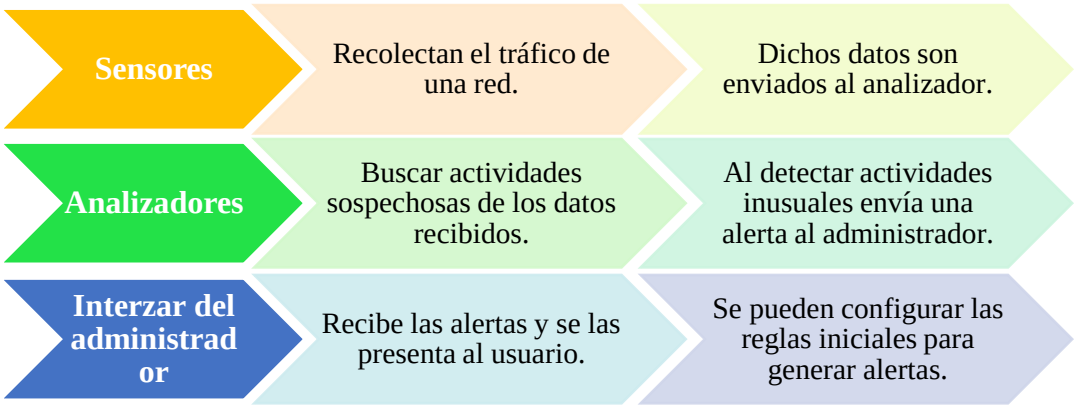


Fig 9. Estructura de los IDS

Fuente: Elaborado por autor

Los sistemas de detección de intrusos también son categorizados en dos sistemas, pasivos y reactivos:

- **Sistema pasivo:** Se define como un sistema de detección del uso inadecuado de activos de una organización, con el objetivo de almacenar la información recolectada y emitir una alerta al administrador o bien para ser almacenado en una base de datos.
- **Sistema reactivo:** Este sistema, a diferencia del anterior, se encarga de responder a las alertas y no se limita solo a almacenar la información, es decir, puede tomar acciones como reprogramar las reglas definidas para impedir el acceso al usuario que comete irregularidades en la organización con el activo, este tipo de sistemas tienen una reacción inmediata a la detección de posibles ataques y se las conocen como sistema de prevención de intrusos (IPS).

Algunos de los aspectos claves dentro de los IDS se observa en la figura 10:

Fig 10. Aspectos básicos de los IDS

Fuente: Elaborado por autor

- **Detección de firmas:** Los sistemas de detección de intrusos que se basan en firmas utilizan entradas de base de datos que contienen patrones conocidos de ataques o intrusiones. Al detectarse una coincidencia de carácter anómalo se activa una alarma. Este tipo de método es funcional para los tipos de intrusiones que se conocen, pero una desventaja es que cuando se presentan ataques nuevos que no han sido identificados, el sistema puede fallar al momento de detectar dichos ataques [23].
- **Detección de anomalías:** Este método de detección de anomalías consiste en monitorear el comportamiento de la red que por lo general tiene que ser normal, también está en busca de desviaciones de información significativamente anormales que indiquen intrusión o filtraciones de datos. Esta medida es de mucha utilidad para la detección de nuevos y desconocidos ataques permitiendo tomar medidas de corrección, sin embargo, esto también puede dar paso a la generación de falsos positivos en sus alertas [23].
- **Evaluación de rendimiento:** Los IDS constan de una evaluación exhaustiva en cuanto a la precisión de la detección identificando la cantidad de los falsos positivos al igual que los falsos negativos con respecto a los diversos ataques que puedan ser efectuados por un agente externo. El porcentaje de alarmas generadas por el IDS falsas son conocidos como tasa de falsos positivos, esto da lugar a una gran cantidad de alarmas inútiles que disminuyen la eficiencia del IDS. Por otro lado, el porcentaje de incursiones que el IDS pasa por alto se

los denomina tasa de falsos negativos, esto podría estar comprometiendo la seguridad y datos privados de la red al generar un alto índice de falsos negativos. También se encarga del rendimiento del sistema que hace referencia a la capacidad de los sistemas de detección de intrusos que tienen para el procesamiento de una gran cantidad de datos sin comprometer el rendimiento de la mima. Esto se puede medir de acuerdo con varias métricas tales como la capacidad de almacenamiento, el uso de recursos o la velocidad de procesamiento [23].

2.8 Técnicas de IA usadas en detección de intrusiones

Las técnicas de inteligencia artificial usadas en la detección de intrusiones se basan en el manejo del aprendizaje automático y el aprendizaje profundo dentro de dichos sistemas con la finalidad de mejorar exponencialmente la capacidad de detectar problemas con una mayor precisión. Estas técnicas consisten en analizar una gran cantidad de datos y con esto lograr la identificación de patrones que son sutilmente incongruentes y que pasarían desapercibidos por cualquier otro método exponiendo a todo el sistema a una intrusión. Estas técnicas utilizadas para detectar intrusión en los sistemas incluyen a los siguientes [16]:

- Lógica difusa
- Razonamiento basado en casos.
- Deep Learning (DL)
- Machine Learning (ML)

2.8.1 Lógica difusa

Es un método usado para tratar la imprecisión e incertidumbre de los datos y se utiliza también en la detección de intrusiones para modelar y evaluar el grado en que un suceso encaja dentro de la categoría de intrusión mejorando la precisión de la detección de anomalías y reduce los falsos positivos que puedan generarse. La lógica difusa parte del campo de la IA (Inteligencia Artificial), ocupándose de la imprecisión y ambigüedad de los datos permitiendo asignar distintos grados de pertenencia a diversas categorías, a diferencia de la lógica binaria clásica que clasifica las entradas como verdaderas o falsas.

Los conjuntos difusos son un grupo de elementos cuyos grados de pertenencia pueden oscilar entre 0 y 1 y que dentro de la lógica difusa se utilizan para representar valores. Las reglas difusas son enunciados condicionales que vinculan entradas y salidas difusas, para poder definir dichas reglas se utilizan tanto operadores como unión difusa, la conjunción y la negación. Entre los usos de la lógica difusa se encuentran sistemas de control, toma de decisiones, identificación de patrones y modelado de sistemas complejos [19].

2.8.2 Razonamiento basado en casos

Este razonamiento basado en casos es una técnica que utiliza una base de datos de ejemplos anteriores para localizar y categorizar situaciones completamente nuevas, dentro del método de detección de intrusiones, utiliza casos anteriores conocidos para comparar dichos eventos pasados con los actuales con la finalidad de detectar patrones similares en el presente, mediante esta comparación se logra identificar si el evento que está sucediendo en el momento es una intrusión o se la descarta. En la figura 11 se observa el razonamiento basado en casos sigue una serie de pasos:

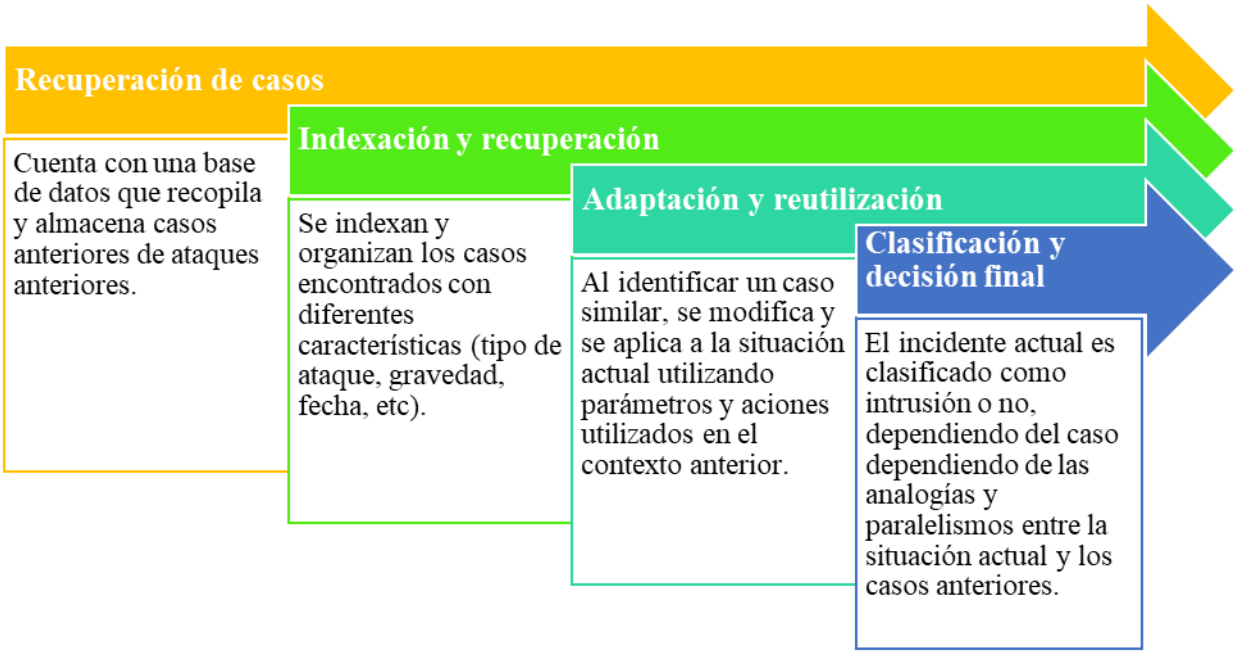


Fig 11. Pasos para aplicar el razonamiento basado en casos

Fuente: Elaborado por autor

2.8.3 Deep Learning (DL)

Es una rama del aprendizaje automático y su traducción al español significa aprendizaje profundo, dentro de este subcampo se hace uso de las redes neuronales para procesar y aprender de datos complejos, las redes neuronales profundas son capaces de detectar anomalías y patrones sutiles en el tráfico de red o la actividad de algún activo que pueda haber sido víctima de la intrusión [14].

A través de un conjunto de datos de tráfico de red o actividad de un activo, se entrena una red neuronal profunda para la detección de intrusiones mediante Deep Learning; esta red puede detectar intrusiones en tiempo real, aprender y reconocer patrones o anomalías en los datos registrados, es por esto que el aprendizaje profundo presenta varias ventajas por sobre los métodos tradicionales de detección, por ejemplo una cualidad a destacar es su capacidad de reconocer patrones intrincados y no identificados [14].

A medida que se recopilan nuevos datos, se descubren nuevos peligros, por ello Deep Learning también puede ajustarse y mejorar con el tiempo. Es crucial tener en cuenta que el uso del aprendizaje profundo para la detección de intrusiones requiere una inversión significativa tanto en datos como en potencia informática, además el calibre y volumen de los datos utilizados para entrenar la red neuronal afectan a la precisión y eficiencia de la detección [3].

2.8.4 Machine Learning (ML)

Conocido también como aprendizaje automático, es un subcampo de la inteligencia artificial que se centra en crear modelos y algoritmos que permitan a los ordenadores a aprender de los datos y actuar en consecuencia sin tener que ser programados explícitamente. Los algoritmos que utilizan el aprendizaje automático aprenden por experiencia y exposición a conjunto de datos, en lugar de seguir instrucciones predefinidas [28].

2.9 Estructura del Machine Learning

En la figura 12 muestra las categorías del maching learning, en donde los enfoques de aprendizaje automático pueden clasificarse en varios grupos, en función del grado de supervisión humana del algoritmo.

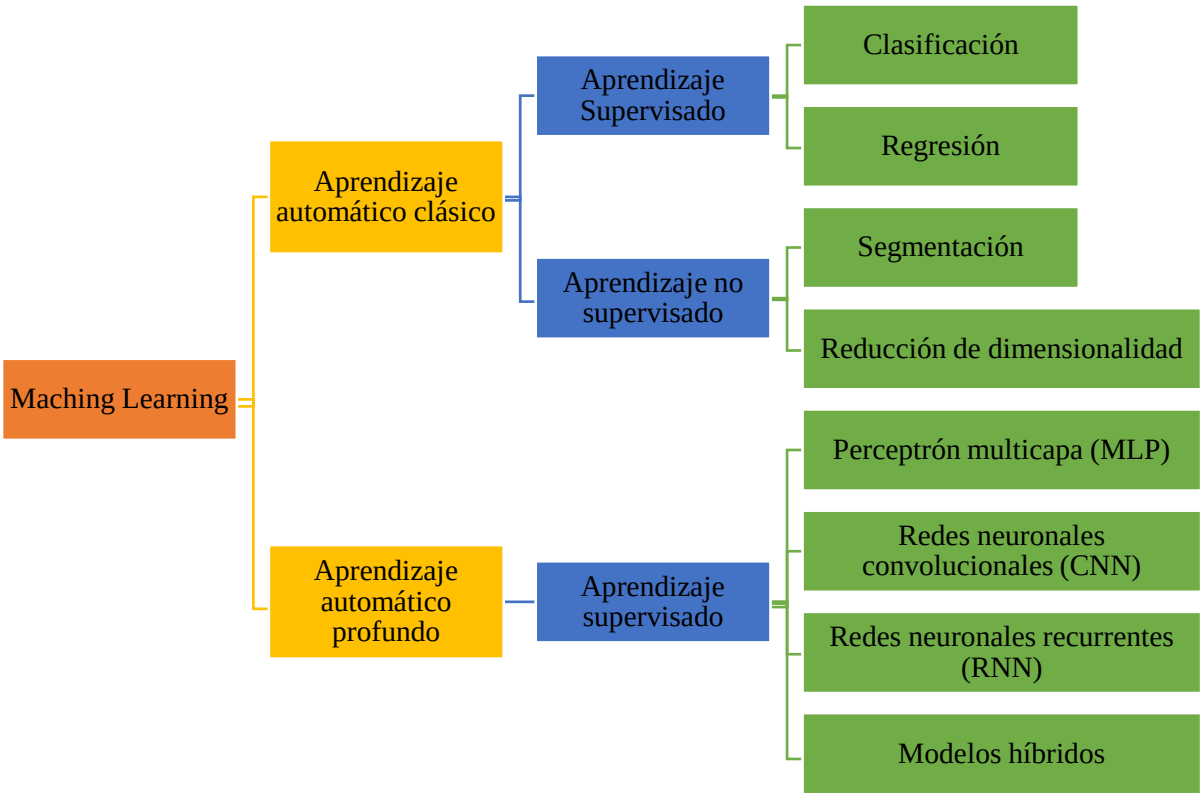


Fig 12. Estructura jerárquica del aprendizaje automático.

Fuente: Elaborado por autor

2.9.1 Unsupervised Machine Learning (UML)

El aprendizaje no supervisado (UML) es una técnica encontrada dentro del contexto del machine learning, quien hace uso de un modelo el cual es entrenado con un conjunto de datos no etiquetados, es decir, no se le proporciona al modelo ningún tipo de ejemplos de entrada y salida esperados ya que el modelo escogido capta patrones y estructuras inherentes a los datos y aprende de ellos. Como el algoritmo debe aprender a representar la estructura de datos, las principales técnicas en este ámbito son la agrupación y el análisis de componentes principales denominados (ACP) [28]. Esta técnica es utilizada en la detección de intrusiones para encontrar e identificar patrones anormales en el tráfico de la red o en el sistema que pueda dar indicios de una intrusión. Los algoritmos que se usan en esta sección

tienen la capacidad de agrupar eventos relacionados y encontrar patrones diminutos que pasan desapercibidos y que podrían apuntar a una intrusión [8].

2.9.2 Supervised Machine Learning (SML)

Se conoce al aprendizaje supervisado (SML) como una técnica derivada del aprendizaje automático el cual se basa en entrenar un modelo usando una colección de datos previamente entrenados y etiquetados. A diferencia del aprendizaje no supervisado, dentro del aprendizaje supervisado se presentan modelos como ejemplos para una entrada o salida prevista, dicho modelo es capaz de aprender usando estos ejemplos como base para predecir nuevos datos con una mayor precisión [28]. Con el fin de entrenar modelos que puedan reconocer patrones y aspectos distintivos de los ataques, el aprendizaje supervisado es utilizado en el contexto de la detección de intrusiones, para entrenar los modelos se hace uso de un conjunto de datos proporcionados por el tráfico de la red o del comportamiento del sistema que son catalogados como normales o anormales. Los parámetros internos del modelo se modifican cuando se expone a más datos para mejorar la generalización y precisión de la predicción con datos nuevos [8]. En la tabla 1 se muestran dos ejemplos que esta técnica utilizada en la detección de intrusos:

Clasificación	**Binaria:** El modelo es entrenado para clasificar los sucesos como benignos o maliciosos, para ello el modelo recibe ejemplos de eventos clasificados como normales o maliciosos y aprende de ellos para generar predicciones precisas sobre nuevos datos.
	Multiclase: El modelo es entrenado para segmentar los eventos en distintas las categorías de intrusión, el modelo recibe ejemplos de estos estos eventos con etiquetas de las distintas categorías para que pueda aprender y generar predicciones precisas sobre nuevos datos.
Regresión	La regresión consiste en entrenar un modelo para predecir una variable numérica como la cantidad de intentos fallidos de inicios de sesión o del tráfico de la red.

Tabla 1. Tipos de técnicas orientadas a la detección de intrusos

Fuente: Elaborado por autor

Estas técnicas se denominan aprendizaje supervisado porque el modelo se construye a partir de los valores conocidos de las observaciones, es decir, el ordenador “aprende” de los datos conocidos para predecir resultados futuros, este

tipo de aprendizaje utiliza predictores y variables de respuesta para construir modelos matemáticos con la intención de predecir o clasificar observaciones que se obtienen posteriormente.

Al querer implementar un modelo de aprendizaje supervisado, es necesario seguir varias etapas para que el resultado sea el esperado y estas se observan en la figura 13:

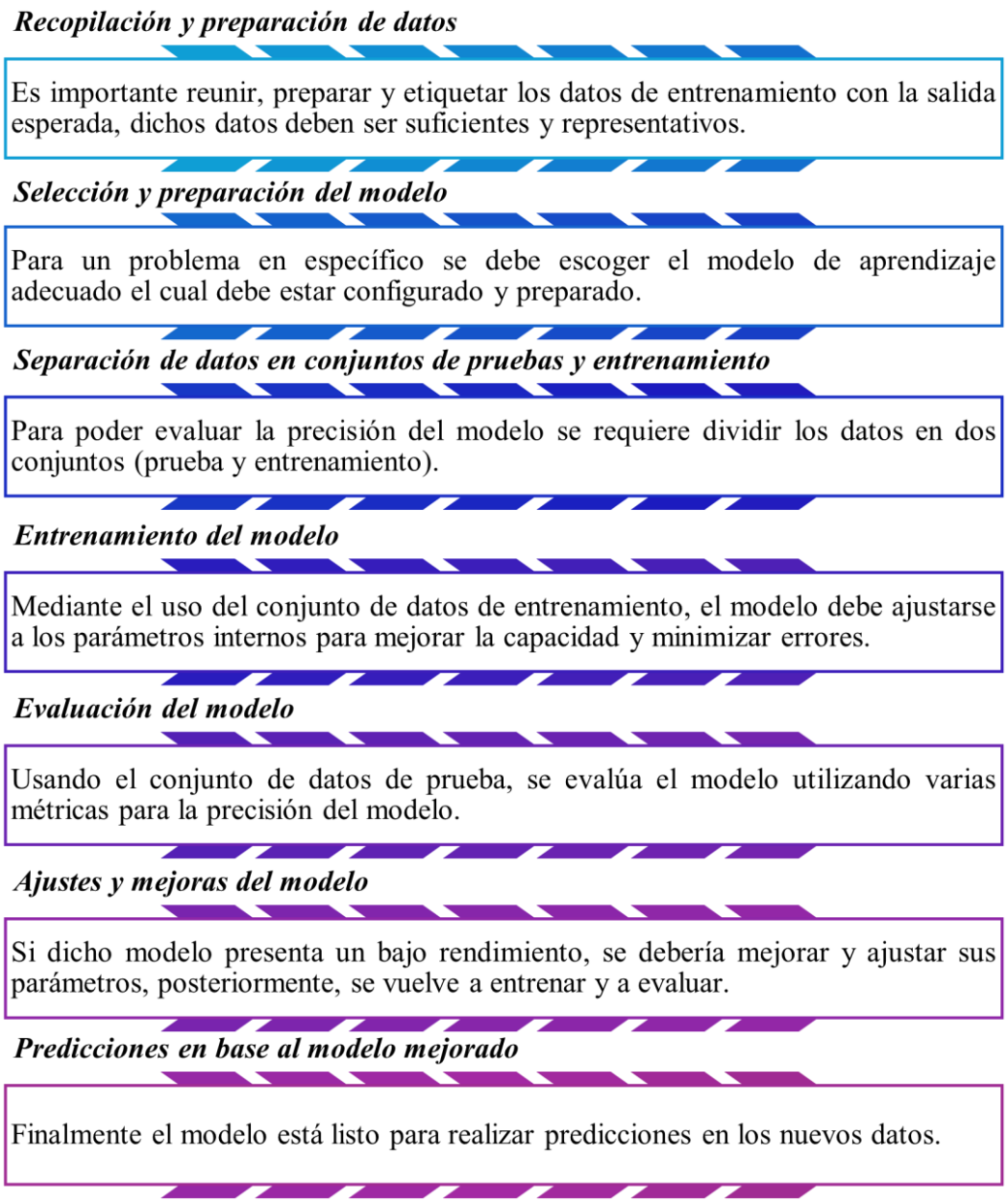


Fig 13. Etapas de la técnica de aprendizaje

Fuente: Elaborado por autor

2.9.3 Algoritmos del aprendizaje supervisado

Existen distintos algoritmos o tipos de modelos utilizados en el aprendizaje automático, y dentro del SL se encuentran muchos con distinto tipo de estructuras, tipos de entrada y salida, o de un nivel de complejidad computacional únicos [16].

En esta sección de la figura 14 se presentan los modelos de aprendizaje automático más conocidos en el SL:

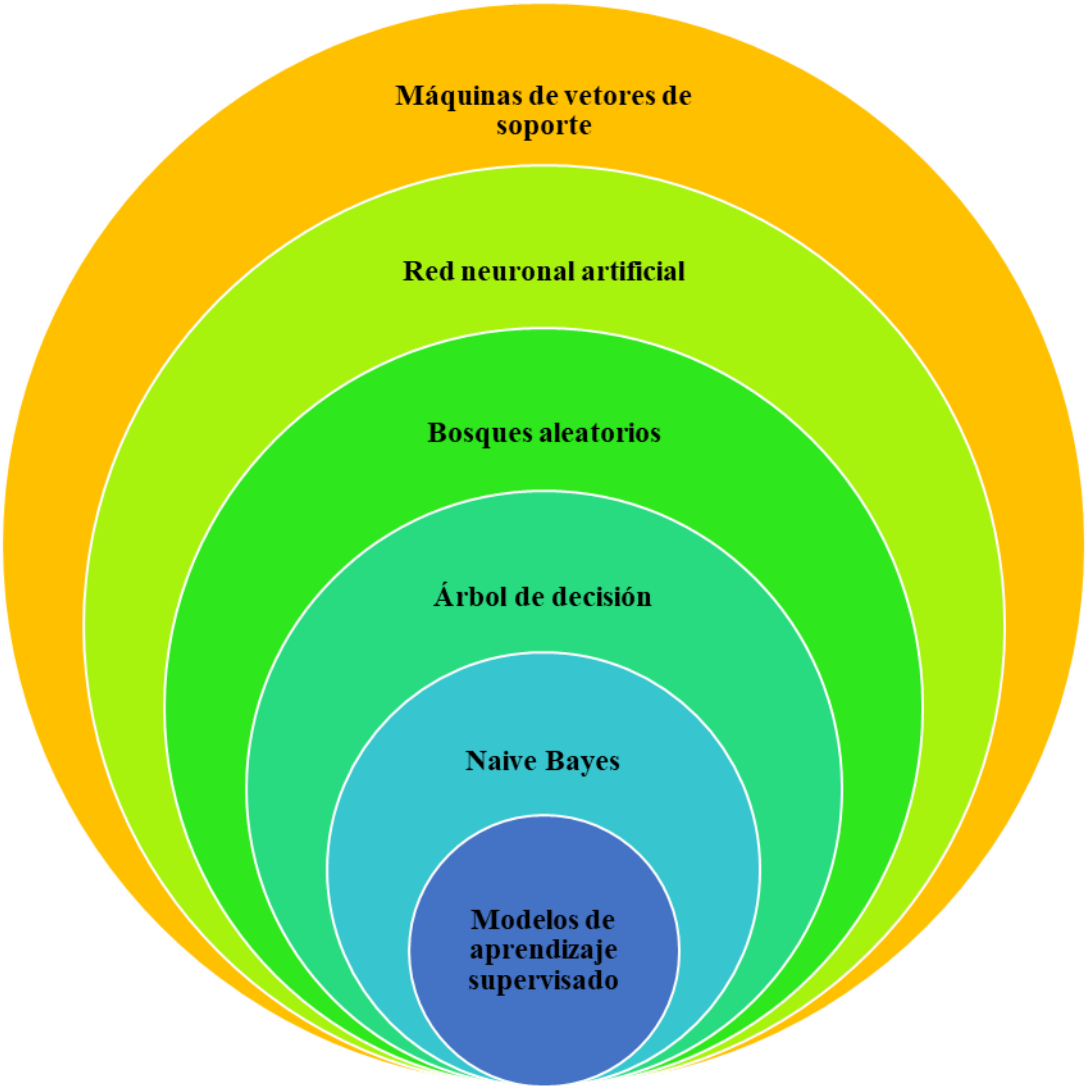


Fig 14. Algoritmos de ML supervisado

Fuente: Elaborado por autor

- **Naive Bayes:** El NB (Naive Bayes) es un logaritmo es considerado como un clasificador basado en la teoría de Bayes, en donde cada característica de una predicción es independiente de las demás y no influye en la probabilidad

cualquier característica adicional. Además, posee una ventaja considerable con el uso de las varianzas de cada variable ya que los parámetros de clasificación del modelo pueden ser determinados con unos datos de entrada relativamente mínimos, dichas varianzas sirven como parámetros del modelo [16].

- **Árbol de decisión:** El DT (Decision Tree), se lo conoce como un algoritmo supervisado más popular dentro de la categoría del aprendizaje automático. Mediante la aplicación de un conjunto de reglas logra emplear la clasificación y regresión de datos para formar una estructura tipo árbol con base o raíz presentadas en la tabla 2:

Nodos	Ramas	Hojas
Representan un aspecto diferente de un determinado conjunto de datos.	Representan cada regla o elección que se aplica para la clasificación un árbol.	Representan los posibles resultados de los datos o bien los estados finales.

Tabla 2. Estructura del árbol de decisión

Fuente: Elaborado por autor

En general, este árbol de decisión se encarga de eliminar las ramas (decisiones) que son irrelevantes y elige las mejores características en base a la información que posee. Se utiliza con frecuencia en algoritmos más complejos como lo es el bosque aleatorio [13].

- **Bosques aleatorios:** RF o Random Forest es la combinación de muchos árboles de decisión cuya función es predecir un determinado tipo de resultado o salida de manera individual y quien obtenga mayores votos será considerada como la predicción de ese algoritmo. Para evitar errores individuales y valores atípicos, el conjunto de árboles de decisiones debe poseer una correlación muy baja para que el bosque aleatorio sea fiable [13].
- **Red neuronal artificial:** También se la conoce por sus siglas ANN (Artificial Neutral Network) y es un método de aprendizaje supervisado que se esfuerza por imitar el funcionamiento de las neuronas que posee el sistema nervioso humano. Está conformado por una agrupación de nodos organizados en varias

capas que van desde la entrada, interno y salidas. Esta red neuronal está basada en el algoritmo de perceptrón junto con el denominado mecanismo de retro propagación permitiendo a las neuronas artificiales aprender de manera independiente utilizando los datos de entrada con los que se les enseña para aprender por sí mismas. Cada nodo de la red neuronal artificial es entrenado mediante el algoritmo de perceptrón para determinar los coeficientes de la función de peso, de acuerdo con la combinación de los datos de entrada, logra ofrecer un resultado de salida más centralizado y compacto. En el transcurso del entrenamiento, los coeficientes son modificados por la neurona para hallar los patrones ocultos situados dentro de los datos de entrada y que pasaban desapercibidos permitiendo identificar datos determinados [28].

- **Máquinas de vectores de soporte:** El algoritmo SVM (Support Vector Machine) es un enfoque que se encuentra dentro del aprendizaje automático supervisado y consiste en la creación de un modelo que se basa en la mayor separación factible entre puntos de muestra que se encuentra en el espacio vectorial dividiendo las clases en distintos espacios como sea posible haciendo uso de una separación de hiperplano dándole su nombre de Vector Machine encontradas en la mitad de los puntos más cercanos de cada clase, este hiperplano es utiliza al vector de soporte como un límite de decisión para la clasificación de las tomas de muestras [8].

2.10 Definición de Hacking

En el campo de la ciberseguridad, se encuentran términos como “Hackers” o “Hacking” el cual está muy relacionado a diferentes tipos de acciones, entre los más sonados es conocido como una actividad que explora o manipula los sistemas informáticos y por consiguiente, accede desde cualquier lugar del ciberespacio hacia un dispositivo informático o activo (computadoras, redes inalámbricas, etc) aprovechándose de las diversas vulnerabilidades que una red pueda tener o bien, extrayendo contraseñas de acceso suplantando al usuario legal [19]. El hacker como tal, es un individuo que cuenta con mucha experiencia en el campo de la seguridad inalámbrica, su propósito es vulnerar la seguridad de una organización a nivel de la red para infiltrarse en sus sistemas informáticos. En esta categoría existen distintos tipos de hackers, entre ellos los 3 más conocidos son:

- **Sombrero negro:** Personas que vulneran la seguridad informática de una corporación con el fin de cometer actividad maliciosa para su propio beneficio que en muchos casos es económico.
- **Sombrero gris:** Son aquellas personas que se infiltran ilegalmente en la red de las empresas, pero sin intenciones maliciosas ya que lo hacen con el fin de exponer las vulnerabilidades que dicha empresa pueda tener para dar a conocer a los propietarios y que estos tomen acciones con respecto a su seguridad.
- **Sombrero blanco:** Por otro lado, este tipo de hackers suelen ser llamados como hackers éticos, la diferencia con respecto al anterior es que estos trabajan bajo el margen de la ley u organización en la que se encuentran buscando vulnerabilidades dentro de esa red y reforzando los sistemas de seguridad.

El hacking es la actividad que realizan las personas o los hackers en sí, como se mencionó antes existen distintos tipos de hackers que laboran en dos campos:

- **Hacking ilegal**
  Se define como la actividad delictiva que tiene fines maliciosos como lo es la manipulación de sistemas o la exploración de esta, la profundidad de este tema abarca muchos aspectos ya que los hackers que laboran en este campo suelen lucrarse con el robo de información de carácter personal o a nivel financiero, también llegan a extremos como lo es la extorsión haciendo uso de la información robada. Esta actividad es condenada ya que rompe las reglas éticas del uso de las redes por lo que el uso indebido de estas habilidades puede ser sujetas a penalización por la ley [20].
- **Hacking ético**
  Es la contraparte del hacking ilegal, puesto que es una actividad que realizan profesionales y es un término muy usado por ellos, el trabajo que emplean es confiable y seguro. El hacker ético que realiza esta actividad por lo general no es un peligro, más bien es quien proporciona un sistema seguro ya que su objetivo es evaluar, identificar e informar a las organizaciones acerca de las vulnerabilidades que encuentra en la estructura de sus redes [29].

La opción de aplicar el hacking ético surge de varias motivaciones, estas se observan en la figura 15:

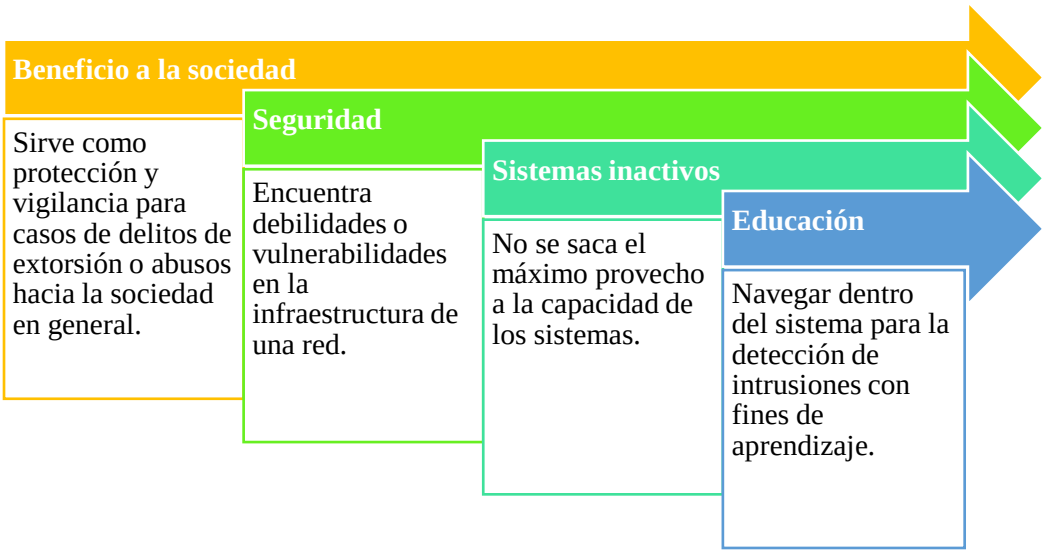


Fig 15. Motivos del uso del hacking ético

Fuente: Elaborado por autor

2.11 Definición de pentesting

El pentesting o pruebas de penetración, surge su nombre por sus palabras en inglés (penetration, test), y se la define como una técnica o metodología aplicada a la seguridad informática para realizar pruebas que permitan encontrar vulnerabilidades o errores dentro de un sistema [26]. Este tipo de pruebas son utilizados por empresas quienes contratan a profesionales en la materia con el fin de comprobar y evaluar si su sistema de seguridad aplicada a sus organizaciones son lo suficientemente robustas o carecen de protección contra intrusiones. el objetivo principal de la realización de las pruebas de penetración es identificar y corregir los sistemas débiles que son fácilmente hackeadas por personas maliciosas [26]. Su diseño se basa en la clasificación de las falencias de seguridad determinando el alcance y repercusiones de estas, esto proporciona a las empresas información fundamental para contrarrestar posibles ataques a sus dominios mejorando la defensa y eficiencia de su seguridad [26].

2.11.1 Clasificación de pentesting

El pentesting puede clasificarse en dos partes conocidas como el origen de las pruebas, aquí se determina el tipo de origen, estos pueden ser de manera interna

o externa. Al referirnos al origen interno el pentester se centra en el interior de la empresa, es decir las pruebas se ejecutan con el uso de dispositivos como hardware y software y junto con eso se procede a la ejecución de las pruebas [2].

Por otro lado, el origen externo tiene más relación a una prueba de manera remota o a entornos virtuales como lo son las páginas web, en donde no se requieren dispositivos físicos para la realización del pentesting [30]. Con este contexto se puede definir los tipos de pentesting existentes que entran en la clasificación del conocimiento del objetivo, este concepto indica que al conocer el objetivo se puede agilizar el proceso de pentesting mediante 3 tipos de pruebas manteniendo un alto porcentaje de éxito. En la figura 16 se observa estos tipos de pruebas de penetración:

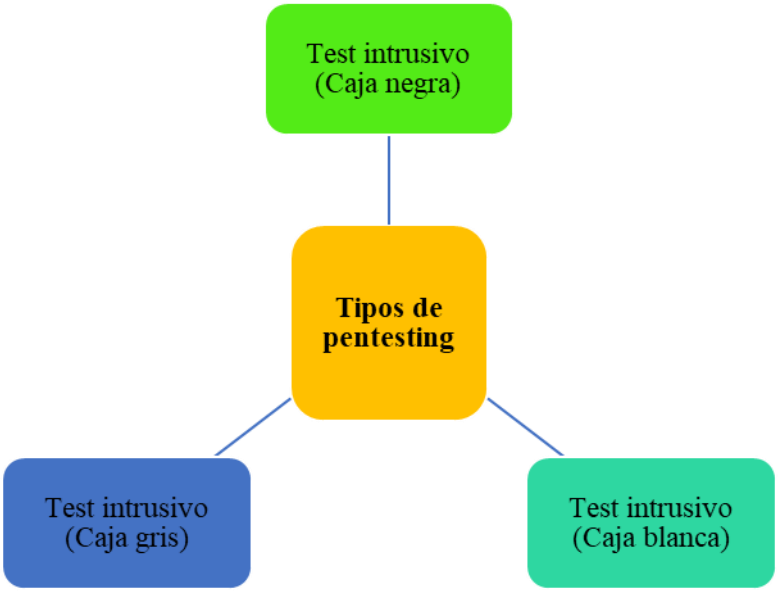


Fig 16. Tipos de pruebas de penetración

Fuente: Elaborado por autor

Test intrusivo externo o caja negra

Tiene un enfoque dentro del nivel de seguridad de la red en el campo externo de las empresas. Es decir que el atacante viene desde afuera para lograr obtener información, también se puede decir que es un tipo de pruebas a ciegas ya que no se cuenta con información de la estructura central de la empresa. Es un tipo de prueba que se asemeja a una práctica real puesto que, por lo general los hackers no

tienen el conocimiento de la infraestructura del sistema de seguridad con la que cuentan sus víctimas [20].

Test intrusivo interno (caja blanca)

Estas pruebas se centran en la seguridad privada de la empresa, por lo que se la puede definir como del tipo de prueba más completa. Esto gracias a que la empresa es quien contrata el servicio, puede proporcionar la información requerida por el pentester, ya sea la cantidad de equipos con las que cuenta la empresa, cómo funciona su seguridad, la infraestructura de su red, los servidores, entre otros. Todo con el fin de conseguir un análisis más profundo para dar marcha a las pruebas y poder detectar con exactitud las falencias del sistema informático. En la figura 17 se observan características adicionales de esta prueba.

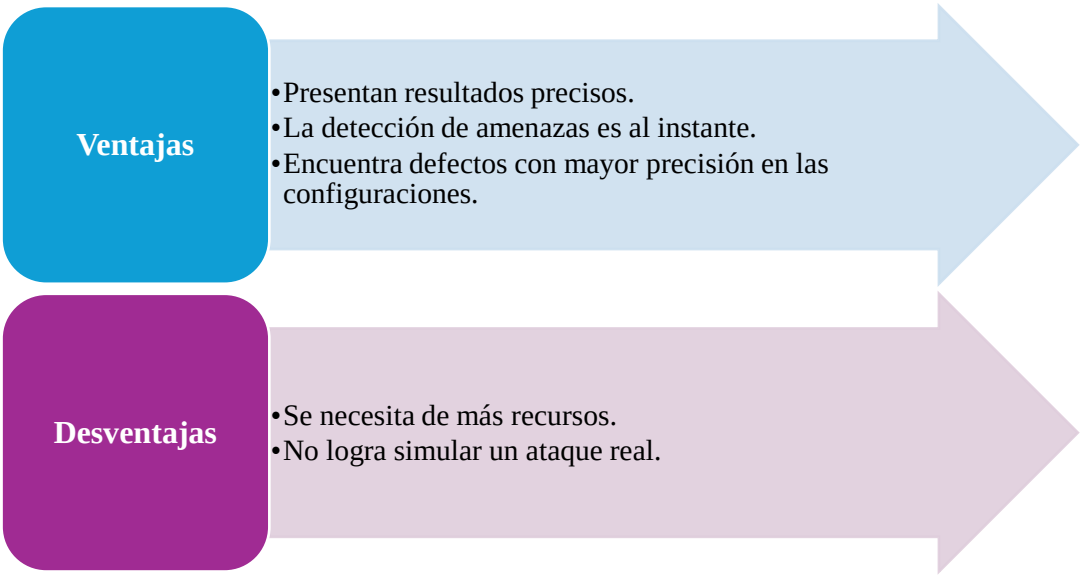


Fig 17. Ventajas y desventajas del uso del test intrusivo interno (caja blanca)

Fuente: Elaborado por autor

Test intrusivo interno (Caja gris)

Esta prueba se la puede definir como un híbrido entre la caja negra y caja blanca puesto que para la realización del pentesting es necesario recolectar cierta información que servirá para la identificación de vulnerabilidades en ciertos lugares. Un ejemplo sería el uso de contraseñas en donde el administrador se lo proporcione al pentester para que este pueda tener el control parcial de la información a la que necesite acceder. Este tipo de prueba es ideal al momento de

hacer estimaciones con respecto a las amenazas presentes y es muy rentable en comparación a las otras, pero como en el anterior, este no proporciona una simulación de un ataque real.

2.11.2 Etapas del pentesting

En la siguiente figura 18 se muestra la cronología que se debe seguir al momento de ejecutar una prueba de penetración:

Fig 18. Pasos para realizar pentesting

Fuente: Elaborado por autor

Recolección de la información

Se la considera como la primera etapa y más importante en la prueba de penetración puesto que se recopila toda la información existente que servirá de ayuda para cumplir el objetivo principal, es decir, el pentester se centra en la identificación de opciones que faciliten su intrusión a la red [7].

Análisis de vulnerabilidades

Una vez se recolecta la información, el siguiente paso es analizar dichos datos a través de los resultados obtenidos con el fin de encontrar vulnerabilidad en la red o fracturas en la seguridad. Este proceso es el encargado de identificar falencias que puedan estar sujetos a un ataque inminente [5].

Explotación de las vulnerabilidades

Cuando el pentester tenga identificadas las vulnerabilidades del sistema, empieza a realizar el ataque a la red para intentar comprometer la información mediante el uso de técnicas, contraseñas y usuarios extraídos en el proceso anterior. Un escenario que podría presentarse es que, al momento de intentar explorar una vulnerabilidad encontrada, puede que no sea posible hacerlo, esto debido a las otras capas de seguridad que pueda poseer la infraestructura de la red [5].

Informe del procedimiento

Esta es la fase final de toda la prueba, el pentester se encarga de documentar todo el procedimiento llevado a cabo junto con los resultados anexados, con la finalidad de dar a conocer a detalle los posibles riesgos a los que están expuesto debido a las vulnerabilidades encontradas, así mismo se detallan los puntos fuertes de la seguridad que maneja la empresa y a los sectores deficientes se hace un énfasis para su corrección. El informe es entregado al técnico encargado del personal de TI, el informe debe ser detallado y escrito en un lenguaje no tan formal para que pueda ser entendido por todo el personal [4].

2.12 Herramientas de pentesting de la nueva generación

Existen múltiples herramientas en el mercado con el cual se pueden realizar pruebas de penetración y de los cuales permiten obtener y mejorar los resultados dentro de sus posibilidades. Al agregar la palabra “nueva generación”, hace referencia a que son aquellos equipos que hacen uso de tecnología avanzar permitiendo realizar estas pruebas a los sistemas informáticos de una manera más exacta con respecto a los de la generación anterior. Una de sus características más destacables son la precisión y eficiencia con la que realizan su trabajo. Algunas de las herramientas con software libres para realizar pentesting son los siguientes:

- **Metasploit:** Esta herramienta tiene a su disposición la opción de usar una base de datos que contienen vulnerabilidades definidas para poder identificarlo dentro de una red y así explotar sus vulnerabilidades. Esta herramienta cuenta con un código abierto por lo que es más común su uso en la seguridad informática [31].
- **Nmap:** La función principal de esta herramienta es el escaneo de redes cuyo fin, como el anterior es encontrar vulnerabilidades, su nombre completo es

Network Mapper. Debido a que también es de código abierto, su uso se extiende más por administradores de redes para la gestión de las actualizaciones de programas y servicios. Nmap suele utilizar paquetes de IP [30].

- **Nessus:** Es utilizada para evaluar las vulnerabilidades de la red conociendo las problemáticas que afectan a la empresa y que son catalogados como críticos. Esta herramienta es compatible con múltiples plataformas puesto que en su interior cuenta con varias plantillas previamente configuradas con el fin de hallar vulnerabilidades [32].

Existen otros tipos de herramientas que cumplen la misma función, estos equipos pueden ser encontrados en la tienda online "Hak5" o bien por amazon, entre sus productos se encuentran los siguientes:

- Wifi Pineapple Mark VII
- Hak5 LAN Turtle
- Hak5 Packet Squirrel
- Aursinc deauther wifi

2.13 Aspectos legales del hacking ético en Ecuador

La seguridad informática se ha convertido en un pilar fundamental para el éxito de toda empresa puesto que el mundo cada vez se moderniza con la tecnología y Ecuador, al igual que muchos países, no es ajeno a los crecientes desafíos cibernéticos que ponen en riesgo la información confidencial de los internautas. Por ello el pentesting se hace presente como una herramienta útil para las organizaciones, ya que ofrece una solución proactiva y estratégica para proteger sus activos digitales [33]. Existen dos tipos de usos de los equipos informáticos, el primero es la generación de información legal y correcta conforme a la ley, pero la segunda roza la ilegalidad puesto que son usados para cometer cibercrímenes para robar la información legal, existen dos formas en las que estos equipos suelen estar involucrados en delitos informáticos:

Crímenes efectuados por el ordenador	El equipo es utilizado como herramienta para realizar actos como fraudes, suplantación de identidad, robo de archivos.

Crímenes en donde el ordenador es el objetivo	Dificulta la ubicación del criminal al igual que la víctima, evadiendo la jurisdicción del crimen ya que el modo de recolección y manipulación de las evidencias es diferente a la forencia de datos tradicional.

Tabla 3. Tipos de crímenes utilizando el ordenador

Fuente: Elaborado por autor

En el Ecuador, la legislación ha definido en el COIP (Código Integral Penal) varios artículos indicando la gravedad del delito para no incurrir en ellos, estos son:

> ***Artículo 190.- Apropiación fraudulenta por medios electrónicos. -*** *Este artículo establece la sanción de tres años de prisión para quien, mediante el uso fraudulento de sistemas informáticos o redes, se apropie de información ajena* [34].
>
> ***Artículo 229.- Revelación ilegal de base de datos. -*** *Este artículo establece la sanción de uno a tres años de prisión para quien, sin autorización legal, revele o difunda información secreta, íntima o privada contenida en una base de datos* [34].
>
> ***Artículo 231.- Transferencia electrónica de activo patrimonial. -*** *En este artículo se menciona la pena de reclusión de tres a cinco años por la modificación del funcionamiento de programas, sistemas informáticos o telemáticos* [34].
>
> ***Artículo 234. Acceso no consentido a un sistema informático, telemático o de telecomunicaciones. –*** *En este artículo expresa la pena de 3 a 5 años para una persona que acceda en parte o a todo sistema informático sin autorización, ya sea telemático o de telecomunicaciones y manteniéndose en contra de la persona que tiene el legítimo derecho* [34].

Luego de haber revisado el ámbito legal de lo que conllevan los delitos informáticos, es prescindible no cometer estos delitos al momento de efectuar las pruebas de penetración del proyecto.

CAPÍTULO 3: Desarrollo de la propuesta

En este capítulo se presenta el proceso de implementación de la propuesta planteada para la detección y prevención de ataques en redes utilizando modelos de aprendizaje supervisado. El desarrollo de la solución está estructurado en diversas fases y se utiliza una metodología de investigación aplicada a la implementación tecnológica dentro del campo del pentesting, basándose en el entendimiento de procesos prácticos dentro de una red controlada, en este capítulo se desglosa el desarrollo inicial de la propuesta, así mismo, los materiales utilizados en el entorno práctico dentro del laboratorio de telecomunicaciones.

3.1 Componentes físicos de la propuesta

Para la ejecución del presente proyecto, es necesario adquirir equipos profesionales que faciliten el proceso de pentesting, así mismo aquellos equipos que permitirán la creación de una red de prueba y para una buena ejecución hay que tener conocimientos de las características técnicas y capacidades de los equipos que se ocuparán en la implementación.

3.1.1 Switch HPE Office Connect serie 1920

El Switch HPE OfficeConnect serie 1920 son una solución de red Gigabit de alto rendimiento y fácil administración para pequeñas y medianas empresas. Ofrecen velocidades de transmisión de datos de hasta 1 Gbps en cada puerto, enrutamiento estático para una interconexión de redes simplificada, administración web intuitiva para un control completo y funciones de seguridad sólidas para proteger sus datos [35]. A continuación, se destacan sus características principales:

- Rendimiento Gigabit: Velocidades de hasta 1 Gbps en cada puerto para redes de alta velocidad.
- Enrutamiento estático: Interconexión de redes sencilla y eficiente.
- Administración web: Control completo del switch y del servidor virtual a través de una interfaz web fácil de usar.
- Seguridad robusta: Cifrado de datos para mayor protección.
- Calidad de servicio (QoS): Priorización del tráfico de aplicaciones críticas para un rendimiento fluido.

- Eficiencia mejorada: Tecnología half dúplex y full dúplex para duplicar la capacidad de los puertos.

Fig 19. Switch HPE Officeconnect serie 1920

Fuente: sieuthimaychu.vn [35]

Los beneficios que ofrece este enrutador es su fácil configuración ya que es rápida y sencilla a través de un servidor web, ofrece máxima seguridad en la protección de datos confidenciales con cifrado de red, aplicaciones fluidas en la priorización del tráfico para una experiencia de red sin interrupciones, también un rendimiento excepcional ofreciendo una mayor capacidad de puerto para satisfacer las demandas de la red. Los switches HPE OfficeConnect serie 1920 son la solución ideal para pequeñas y medianas empresas que buscan una red confiable, segura y de alto rendimiento.

3.1.2 Router Mikrotik RB2011UiAS-2HnD-IN

Es un equipo que ofrece una amplia gama de funciones y opciones de conectividad, principalmente diseñado para uso en interiores, funciona con el sistema operativo RouterOS, ofreciendo funcionalidades como enrutamiento, firewall, VPN, calidad de servicios y más. Este sistema cuenta con una curva de aprendizaje más centrada que los routers domésticos genéricos, pero ofrece un mayor control y flexibilidad para los usuarios con mayor experiencia [36]. A continuación, se muestra una tabla de especificaciones del modelo:

Especificación	Descripción
Arquitectura	MIPSBE
CPU	AR9344 @ 600 MHZ
Memoria RAM	128 MB
Puertos Ethernet	5 x 10/100/1000 Gigabit LAN, 5 x 10/100 Fast Ethernet LAN

Puertos SFP	1 x SFP (módulo SFP no incluido)
Wi-Fi	Punto de acceso inalámbrico 802.11 b/g/n de doble polarización, 2.4 GHz, hasta 1000 mW, antenas integradas de 4 dBi
Puertos USB	1 x USB 2.0
Características adicionales	Botón de reinicio, pantalla LCD, ranura para tarjeta microSD, licencia RouterOS L5

Tabla 4. Especificaciones del Router Mikrotik

Fuente: Elaborado por autor

Para efectos del proyecto, se selecciona este router (figura 20) ya que sus cualidades la convierten en una solución versátil para redes domésticas y de pequeñas empresas, ideales para pruebas de pentesting, esto la convierte en una opción ideal debido a sus opciones de conectividad y precio asequible.

Fig 20. Router Mikrotik RB2011UiAS-2HnD-IN

Fuente: mikrotik.com [36]

3.1.3 WiFi Pineapple Mark VII

Es un dispositivo diseñado por la empresa Hak5, cuya funcionalidad está centrada en la utilización para auditorías de redes WiFi y ataques Man in the Middle (MitM), aunque no solo se entra en un único ataque ya que este equipo puede imitar redes preferidas para vulnerar los datos de los dispositivos conectados a ella, también captura handshakes WPA (Wi-Fi Protected Access) y simular puntos de acceso empresariales permitiendo extraer credenciales de acceso dentro de redes empresariales. Es ideal para la evaluación de vulnerabilidades. Está catalogada como un dispositivo de red de la nueva generación ya que se combina con múltiples

radios basadas en roles [1]. A continuación, en la tabla 4 se presentan las características principales del equipo:

WiFi Pineapple Mark VII	
Interfaz	(3) Wifi, (1) USB 2.0, (1) USB-C Ethernet
Standards	$USB\ 2.0, 802.11b/g/n$
Dimensiones	$107 \times 93 \times 21\ mm$
Rango de frecuencia	$2.412{\sim}2.4835\ GHz$
Power	$10W\ (USB\ 5V\ 2A)$
Temperatura de funcionamiento	$35°C \sim 45°C$
Temperatura de almacenamiento	$-20°C \sim 50°C$
Humedad relativa	0% a 90% (noncondensing)

Tabla 5. Especificaciones del WiFi Pineapple Mark VII

Fuente: Elaborado por autor

Este equipo fue elegido para el proyecto porque tiene la factibilidad de integrar varios módulos en un solo equipo, también tiene un firmware actualizado a la última versión que permite estar al día con nuevas funcionalidades dentro de la interfaz. Ofrece un alto rendimiento desde una interfaz web asequible desde cualquier dispositivo, ya sea laptop o desde un celular, con un ecosistema comprensible para el usuario, convirtiéndose en una herramienta versátil para las pruebas de penetración.

Fig 21. WiFi Pineapple Mark VII

Fuente: shop.hak5.org [1]

Dentro de una red, ya sea empresarial, doméstica o de pruebas, los dispositivos electrónicos con acceso a internet usualmente tienen la opción del Wifi habilitado para conectarse automáticamente estableciendo una comunicación con dicha red y es ahí cuando el dispositivo Pineapple Mark VII entra en escena, con una de sus funciones clona el SSID de la red redirigiendo a todos esos dispositivos al AP (Access Point) falso recolectando la información de cada uno de ellos, la información se guarda dentro de la misma interfaz para su posterior análisis.

3.2 Componentes lógicos de la propuesta

Dentro del desarrollo del proyecto, se utilizarán recursos tecnológicos como el uso de diferentes tipos de software que facilite la proyección del modelado e implementación del diseño propuesto.

3.2.1 SketchUp

Es un software de diseño que crea modelos en 3D implementados en proyectos a gran escala, ya sea de arquitectura o diseño de interiores y hasta para videojuegos. En la ingeniería es utilizado para modelar productos, piezas o maquinarias en 3D, este software nació para consolidar datos de manera visual y atractiva. Cuenta con una interfaz sencilla y de fácil manejo de aprendizaje, así como instructivos y guías para poder crear un primer modelo.

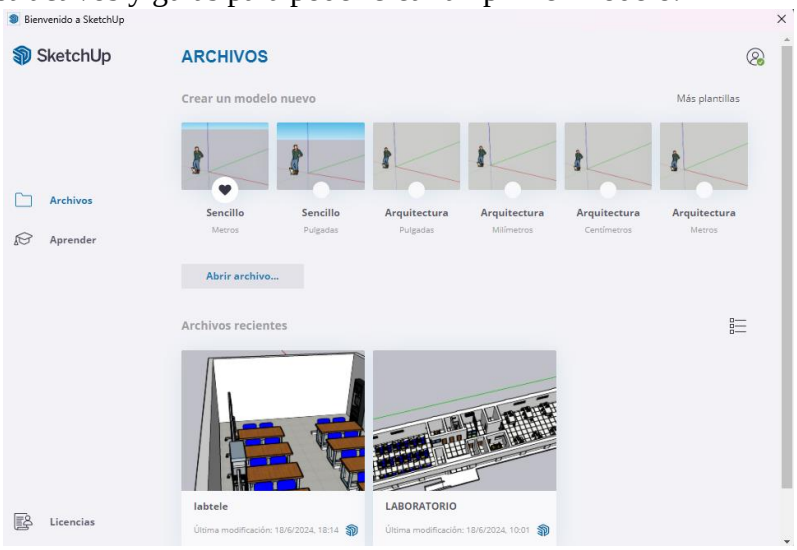


Fig 22. Interfaz visual de SketchUp

Fuente: Elaborado por autor

Para representar la implementación de este proyecto de una forma más llamativa se optó por el uso de este software, cuya finalidad fue crear un diseño de la arquitectura de la red mediante la simulación del laboratorio de telecomunicaciones en un modelado 3D, en donde se especifica la ubicación de los equipos a utilizar. SketchUp es ideal para plasmar las ideas del diseño planteado ya que cuenta con modelos predefinidos de objetos descargables como personas, sillas, mesas, entre otros; todos estos recursos que ofrece lo convierten en una elección confiable para la visualización de la implementación.

3.2.2 Módulos del WiFi Pineapple para las pruebas de pentesting

El dispositivo WiFi Pineapple Mark VII ofrece varios módulos para realizar pruebas de penetración en las redes inalámbricas que amplían las capacidades del usuario en las pruebas de penetración, así mismo sirve como herramienta extra para el análisis de tráfico de la red, a continuación, se explican los módulos a utilizar en este proyecto.

3.2.2.1 Módulo “Evil Portal”

El módulo más conocido que es el Evil Portal. Este es una herramienta robusta que permite la realización de pruebas de concepto y ataques de phishing de manera controlada, su funcionalidad consiste en interceptar el tráfico de la red de múltiples dispositivos que son redirigidos a una red falsa creada por el Pineapple y llevándolos a un portal web falso con la finalidad de capturar información confidencial como credenciales. Los portales cautivos no vienen por defecto en el dispositivo, por lo que se recurre a utilizar repositorios en GitHub para la instalación de estas librerías.

3.2.2.2 Módulo HTTPeek

Este es un tipo de analizador web que inspecciona y modifica el tráfico HTTP (Hypertext Transfer Protocol) que pasa por el AP creado por el WiFi Pineapple, intercepta y registra todas las solicitudes HTTP incluyendo encabezados, datos de formulario, cookies y contenido de la página. Su aplicación en el proyecto es realizar un ataque de Sniffing para identificar las vulnerabilidades en aplicaciones web interceptando tráfico HTTP.

3.2.2.3 Módulo “MDK4”

Este módulo converge una serie de ataques contra redes WiFi que permite realizar pruebas de penetración exhaustivas, automatizando la ejecución de ataques y con una interfaz gráfica fácil de utilizar la realización de ataques. Estos incluyen:

- **Ataques de Hombre en el medio (MitM):** Intercepta y modifica el tráfico entre dispositivos en la red.
- **Ataques de falsificación de puntos de acceso (AP):** Crea múltiples puntos de acceso WiFi falsos para atraer usuarios y capturar credenciales.
- **Ataques de denegación de servicio (DoS):** Inunda la red WiFi con tráfico falso para desautenticar a los clientes impidiéndoles conectarse a la red.

Para el proyecto se utilizará el ataque DoS para poder desautenticar a los dispositivos conectados a la red y a su vez, redirigirlos a un AP Evil, con esto el usuario al intentar reconectarse a internet se capturará los Handshakes.

3.2.2.4 Módulo Nmap

Es un escáner de red que permite descubrir host y servicios en una red, su función es enviar paquetes a los hosts permitiendo analizar las respuestas y determina qué servicios están activos y qué puertos se encuentran abiertos, el acrónimo Nmap significa Network Mapper. Para efectos del proyecto se detectará vulnerabilidades de seguridad comunes, en este caso los puertos abiertos vulnerables.

3.2.2.5 Módulo TCPDump

Es una herramienta de captura de paquetes similar al Wireshark, principalmente utilizados en sistemas operativos de Linux, captura y analiza el tráfico de datos permitiendo observar la información de la red incluyendo protocolos, también permite filtrar los paquetes y contenido de datos, los datos recolectados se exportan en archivos PCAP (Packet Capture), que son compatibles para observarlo en el software Wireshark. Esta herramienta es muy útil para el proyecto ya que será un complemento perfecto para analizar capturar comportamientos de intrusión proporcionados por el WiFi Pineapple, y combinarlo

con los datos recopilados en Wireshark y obtener un conjunto de datos más robusto para comprobar la efectividad de los modelos de aprendizajes.

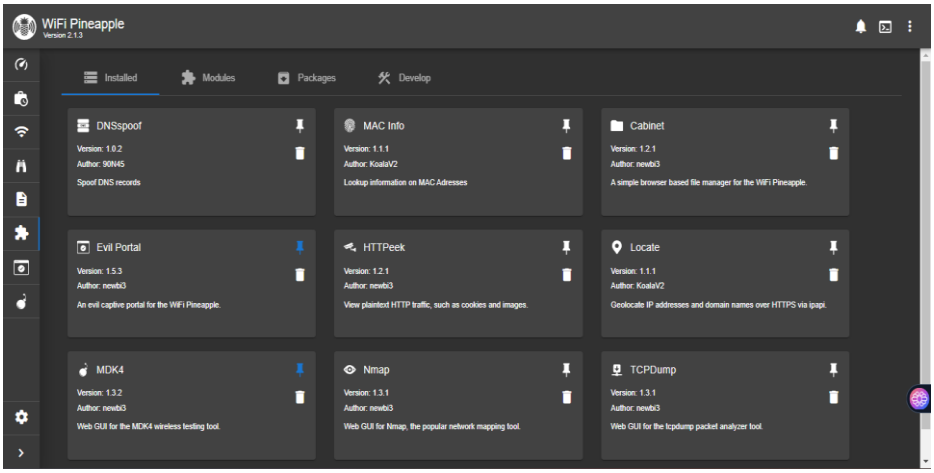


Fig 23. Interfaz de los módulos del WiFi Pineapple Mark VII

Fuente: Elaborado por autor

3.2.3 Wireshark

Es un software orientado al análisis de redes, en términos coloquiales funciona como un microscopio para el tráfico de datos ya que permite observar con detalles la información que fluye a través de las redes, ya sean domésticas o a nivel empresarial. El trabajo que realiza esta herramienta es capturar paquetes de datos, los disecciona y presenta de forma organizada en una interfaz gráfica en donde se visualiza la información importante como las direcciones IP (Internet Protocol) de origen, destino, contenido de cada paquete, protocolos que se usaron en la transmisión, en general se ve el comportamiento de la red [2].

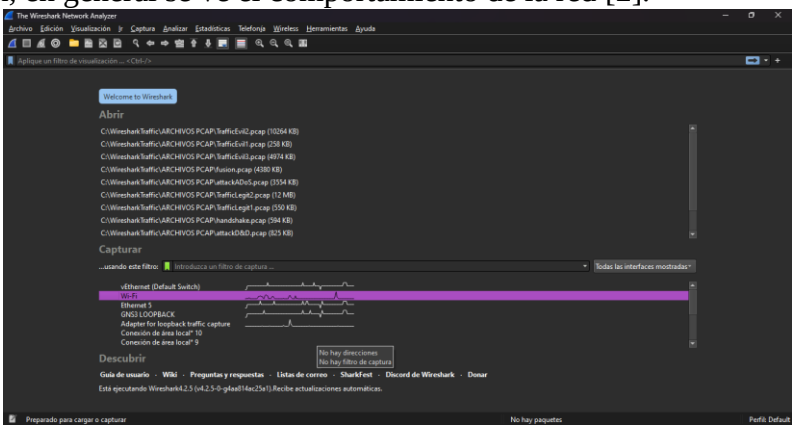


Fig 24. Interfaz del Software Wireshark

Fuente: Elaborado por autor

Se opta por usar esta herramienta debido a la facilidad intuitiva de observar los datos, posee una interfaz completa y la facilidad de agregar más campos para analizar permitiendo identificar problemas, analizar rendimiento, detectar intrusiones y comprender mejor el funcionamiento de los protocolos de red. Además, es una herramienta utilizada fuertemente por profesionales en redes, administradores, desarrolladores, e incluso estudiantes del campo de la tecnología en redes.

3.2.4 Google Colab

Conocido también como "Colaboratory", es una plataforma basada en la nube que proporciona un entorno interactivo para escribir y ejecutar códigos en lenguaje Python, funciona como un cuaderno virtual que combina opciones como texto, código y resultados en un solo entorno sin necesidad de configuraciones previas, los códigos que se crean quedan almacenados en la nube. A continuación, se muestran los campos en donde es comúnmente utilizado [37]:

Aprendizaje automático	Entrena y evalúa modelos de aprendizaje automático
Educación	Herramienta utilizada por estudiantes en proyectos
Ciencia de datos	Analiza y visualiza datos complejos con herramientas integradas
Desarrollo de Software	Se realizan prototipos de aplicaciones sin necesidad de configurar un entorno de desarrollo local

Tabla 6. Aplicaciones de Google Colab en diversos campos

Fuente: Elaborado por autor

Se optó por el uso de esa herramienta ya que es accesible para cualquier persona sin la necesidad de instalar cualquier dependencia en la computadora para su funcionamiento, tiene acceso gratuito a los recursos como los potentes GPUs y TPUs de Google, esto permite la ejecución de códigos como el entrenamiento de modelos de aprendizaje, que es el enfoque de este proyecto.

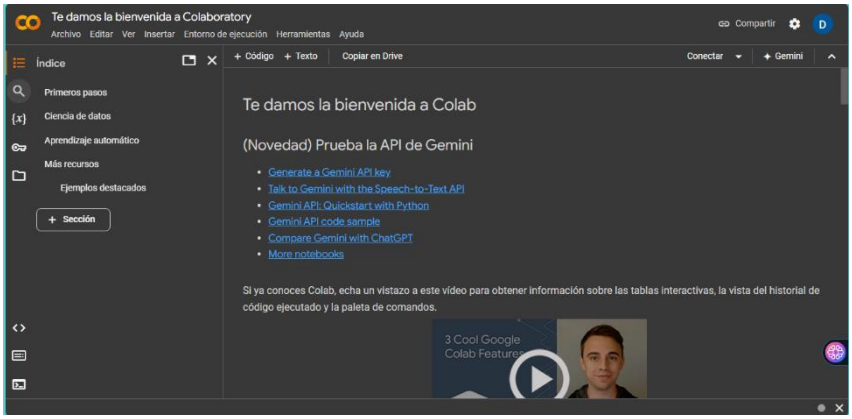


Fig 25. Interfaz de Google Colab

Fuente: Elaborado por autor

3.2.5 Telegram

Esta herramienta es una plataforma de mensajería instantánea y comunicación en tiempo real, que permite a los usuarios enviar mensajes, archivos multimedia y realizar llamadas de voz. Es conocida por su enfoque en la seguridad y privacidad, ofreciendo características como chats secretos que tienen cifrado end-to-end, lo que significa que solo los participantes pueden leer los mensajes intercambiados. Además de ser una plataforma de mensajería, Telegram también soporta la creación de bots y canales, lo que la convierte en una herramienta versátil para desarrolladores y usuarios que desean automatizar tareas, recibir actualizaciones automatizadas o crear comunidades de usuarios alrededor de intereses comunes [38].

Fig 26. Telegram

Fuente: ambito.com [38]

En el contexto de este proyecto, el uso de telegram facilitará la visualización de los resultados de una manera más didáctica y limpia, mediante un código en Google Colab, luego del proceso de la implementación del modelo de aprendizaje supervisado, se pretende recoger los resultados obtenidos y enviarlos a un bot de

telegram, quien recibirá esa alerta para que el usuario adopte medidas de prevención mediante acciones predefinidas en el mismo bot.

3.3 Diseño de la propuesta física

Mediante el uso del software SketchUp se diseña un modelo 3D del esquema de la red que se va a implementar, tal como se ve en la figura 27, ese sería el escenario propuesto para la implementación del proyecto.

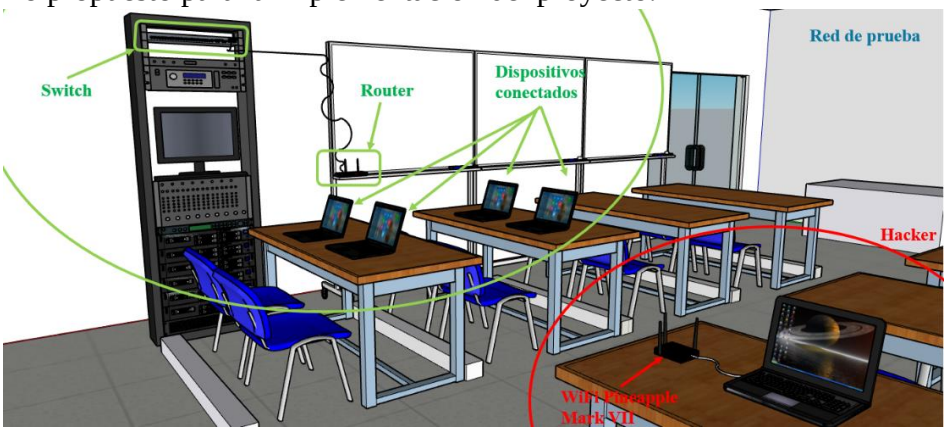


Fig 27. Escenario propuesto de la red a implementar

Fuente: Elaborado por autor

La red está conformada por un switch quien proporcionará el internet del proveedor, a ese switch se conecta un router para habilitar un AP para que los dispositivos se conecten a esa red como se puede observar en la figura 28(a). A su vez, se implementa una estación para realizar los ataques con el equipo de pentesting y una laptop como se observa en la figura 28(b).

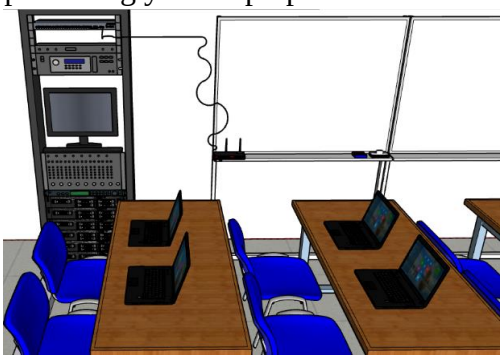


Fig 28.a. Conexión de los equipos de la red

Fig 28.b. Estación para los ataques con el WiFi Pineapple Mark VII

Fuente: Elaborado por autor

3.4 Esquema del modelo de aprendizaje supervisado

Una vez diseñada la red de prueba, se debe tener en claro el proceso que llevará el modelo de aprendizaje supervisado para que detecte los ataques en la red de prueba, este esquema se observa en la figura 29.

Aquí se observa un dataset con características (Features) que el modelo usará para aprender, posteriormente están las etiquetas que se le dará a cada característica del dataset, en el contexto del proyecto, se etiquetarán los paquetes con diferentes tipos de ataques específicos. El modelo entrenado es en donde entra en acción el modelo de aprendizaje supervisado pues se entrena con un conjunto de datos y etiquetas, aprendiendo a mapear las características de entrada y salida.

Una vez que el modelo haya sido entrenado, se procede a proporcionar un conjunto de datos para pruebas que no se usó en el entrenamiento para evaluar el rendimiento y realizar las predicciones validando la efectividad del modelo entrenado.

Una vez introducido los datos de prueba, el modelo empieza a realizar predicciones que son el resultado del proceso de aprendizaje con las etiquetas correspondientes, la clase o el tipo de ataque que el modelo identifica se basa en las características del conjunto de datos de prueba y, por último, las predicciones del modelo se interpretan para identificar el tipo de ataque, dependiendo del resultado se tomarán las acciones pertinentes.

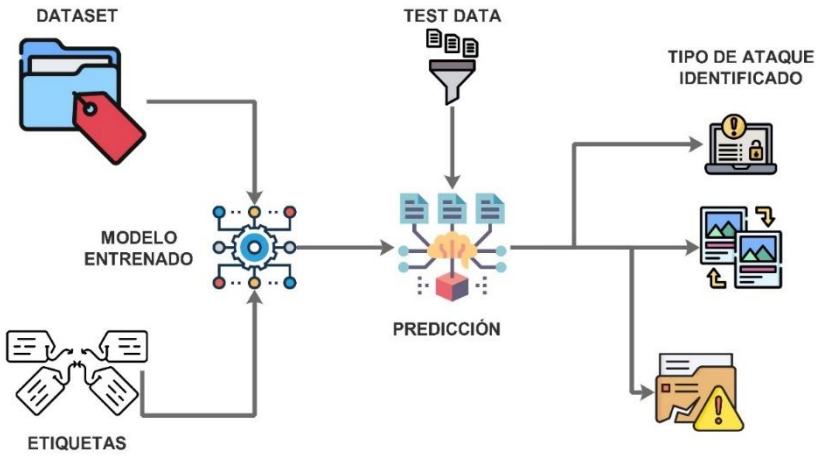


Fig 29. Esquema del modelo de aprendizaje a implementar

Fuente: Elaborado por autor

3.5 Esquema del sistema de alertas para la prevención de ataques

Como se mencionó anteriormente, luego del proceso que realiza el modelo de aprendizaje supervisado, es necesario optar por acciones pertinentes ante un ataque, es por ello por lo que, para el proyecto se pretende implementar un sistema de alertas dirigido a la aplicación de telegram mediante un bot. En la figura 30 se muestra el procedimiento que realizará.

Dentro de la red, el atacante intentará vulnerar las medidas de seguridad, ejecutando diferentes tipos de ataques, posteriormente, se utiliza la plataforma de Google Colab para la implementación y ejecución del modelo de aprendizaje, ahí capturan el tráfico de la red para pasarlo al modelo de aprendizaje supervisado y que determine si el tráfico que se está generando presenta anomalías.

Si el modelo KNN detecta alguno de los ataques previamente etiquetados, se envía una alerta inmediata al bot de Telegram, proporcionando detalles sobre el tipo de ataque identificado. El propósito de este sistema de alertas es únicamente informar sobre posibles amenazas para que el personal técnico calificado, como los técnicos de TI, pueda tomar las medidas correctivas necesarias. Es importante destacar que estas alertas no implican una solución automática a dichos ataques; su función es notificar a las personas competentes para que ellas se encarguen de mitigar los daños.

Fig 30. Esquema del sistema de alertas

Fuente: Elaborado por autor

3.6 Configuración y conexión de los equipos

El acceso a la red de datos es gestionado por el departamento de TIC de la Universidad Estatal Península de Santa Elena, que proporciona internet a cada facultad de la entidad educativa UPSE. Para dar acceso a internet a la red proyectada de la propuesta, el switch HPE 1920 será considerado el servidor que proveerá de internet al router mikrotik y esta a su vez a los dispositivos que se conecten a la red, la conexión de estos equipos se muestra en la figura 31:

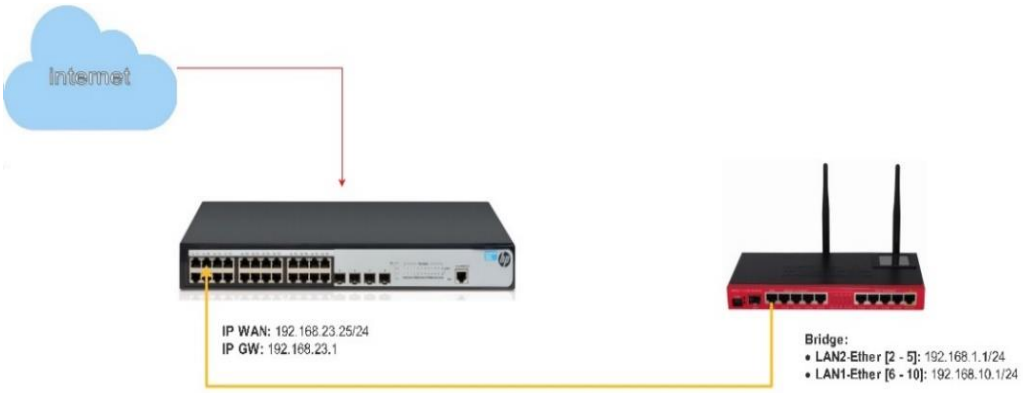


Fig 31. Conexión entre el Switch y Router

Fuente: Elaborado por autor

3.6.1 Configuración de la red en el Router Mikrotik

Para configurar la red que se pretende usar para la auditoría, es necesario usar el software WinBox para ingresar a la interfaz del router mikrotik, una vez dentro se configura al puerto Ether(1) con el nombre “WAN” ya que será quien se conecte mediante cable UTP al proveedor, en este caso el switch, se segmentan los puertos en dos Bridge con el nombre “LAN1 y LAN2”, en la figura 31 se observan las IP’s que se deben configurar para el direccionamiento del puerto WAN y los dos Bridge. Una vez realizada la configuración básica, se procede a configurar la red que llevará como nombre “LAB_TELECO” como se muestra en la figura 32:

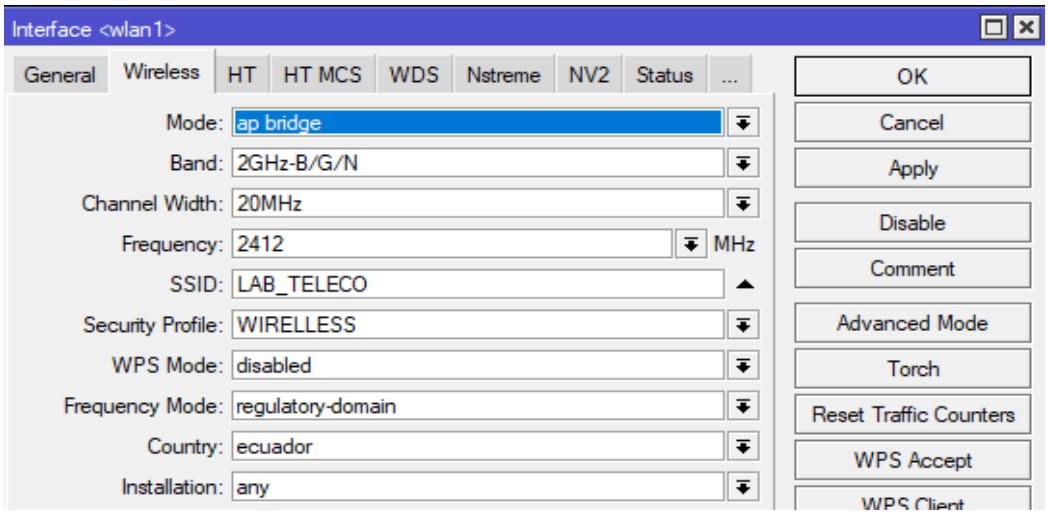


Fig 32. Configuración de la red "LAB_TELECO" en el router

Fuente: Elaborado por autor

3.6.2 Configuración del WiFi Pineapple

Para emplear técnicas de penetración a la red de prueba, es necesario configurar el equipo por lo que, se conecta el equipo mediante cable USB-C a la computadora, y en un navegador se digita la dirección: "http://172.16.42.1:1471", el usuario y contraseña del equipo por defecto será "root".

Dentro de la Pineapple es necesario configurar el Wireless Client Mode quien se encargará de proporcionar de internet a la piña directamente del proveedor como se observa en la figura 33, en este caso se toma como proveedor a la red "UPSE" para evitar cualquier inconveniente al momento de realizar las pruebas.

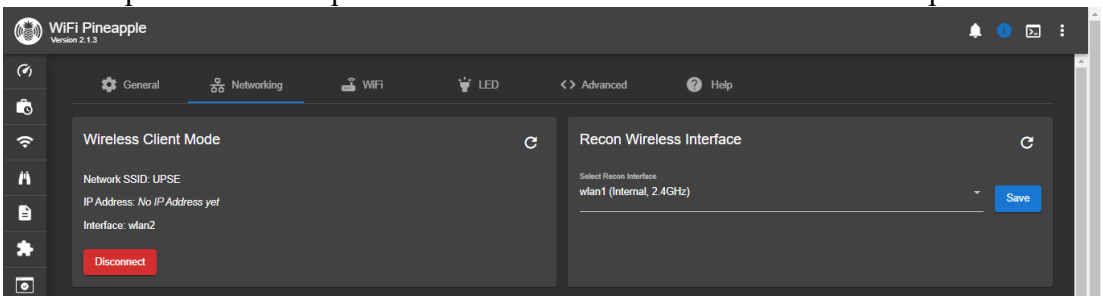


Fig 33. Configuración para proporcionar al Pineapple de internet

Fuente: Elaborado por autor

A su vez, se instalan varios módulos que servirán para las pruebas de penetración, uno de los módulos a utilizar es el conocido “Evil Portal” en donde se alojarán las librerías de los portales cautivos para realizar un ataque tipo fishing. Esto es posible al configurar un Open AP con el nombre “WiFi-Free”, una red abierta al público para capturar credenciales importantes de la víctima.

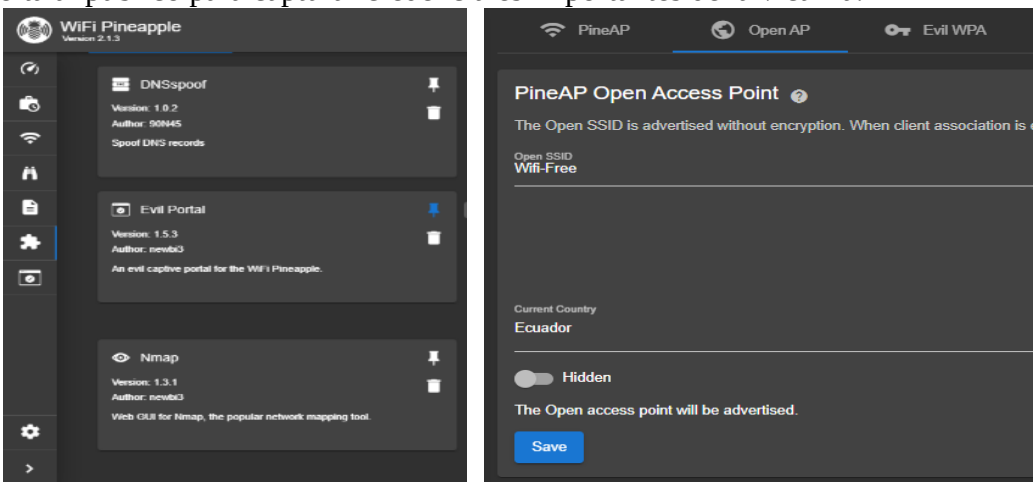


Fig 34. Configuraciones básicas para la preparación del ataque de Fishing

Fuente: Elaborado por autor

3.6.3 Creación del Bot en Telegram

Para comenzar con la creación de un bot en Telegram, el primer paso es localizar al usuario “BotFather”, que es el administrador oficial encargado de gestionar la creación y administración de bots en la plataforma. Una vez encontrado, se inicia una conversación enviando el comando “/start”, lo que desplegará un menú con varias opciones. Entre estas opciones, se debe seleccionar “/newbot” para proceder con la creación de un nuevo bot.

En este punto, el “BotFather” solicitará asignar un nombre al bot, este será el nombre visible que los usuarios verán al interactuar con él. En este caso, se eligió el nombre “Js_eddie_iot_bot” (como se muestra en la figura 35).

Tras completar este proceso, “BotFather” generará un mensaje importante que incluye el token de autenticación. Este token es una clave única que permite a tu bot conectarse con la API de Telegram y realizar acciones automatizadas, como el envío de mensajes o la recepción de comandos. El token debe guardarse de forma segura, ya que es esencial para que tu bot funcione correctamente. Para este bot el

token asignado es el siguiente: **“7322906771:AAFdPn9_TJTNLfg U5PLddFYVSHyOQy8hvqw”**. Además del token, es importante obtener el ID del bot y el ID del chat en donde se va a recibir o enviar mensajes. Para obtener el ID del bot, se solicita al usuario “userinfobot” la información con el comando “/start”, la información que proporciona es el **“ID: 7030670232”** como se muestra en la figura 35. También es útil identificar el ID de los usuarios o grupos con los que el bot interactuará, ya que esto permitirá una comunicación directa entre el bot y el usuario o grupo de destino. Este proceso asegura que el bot está correctamente configurado para interactuar con los usuarios de Telegram mediante la API, utilizando el token como medio de autenticación y el ID para identificar las sesiones de chat.

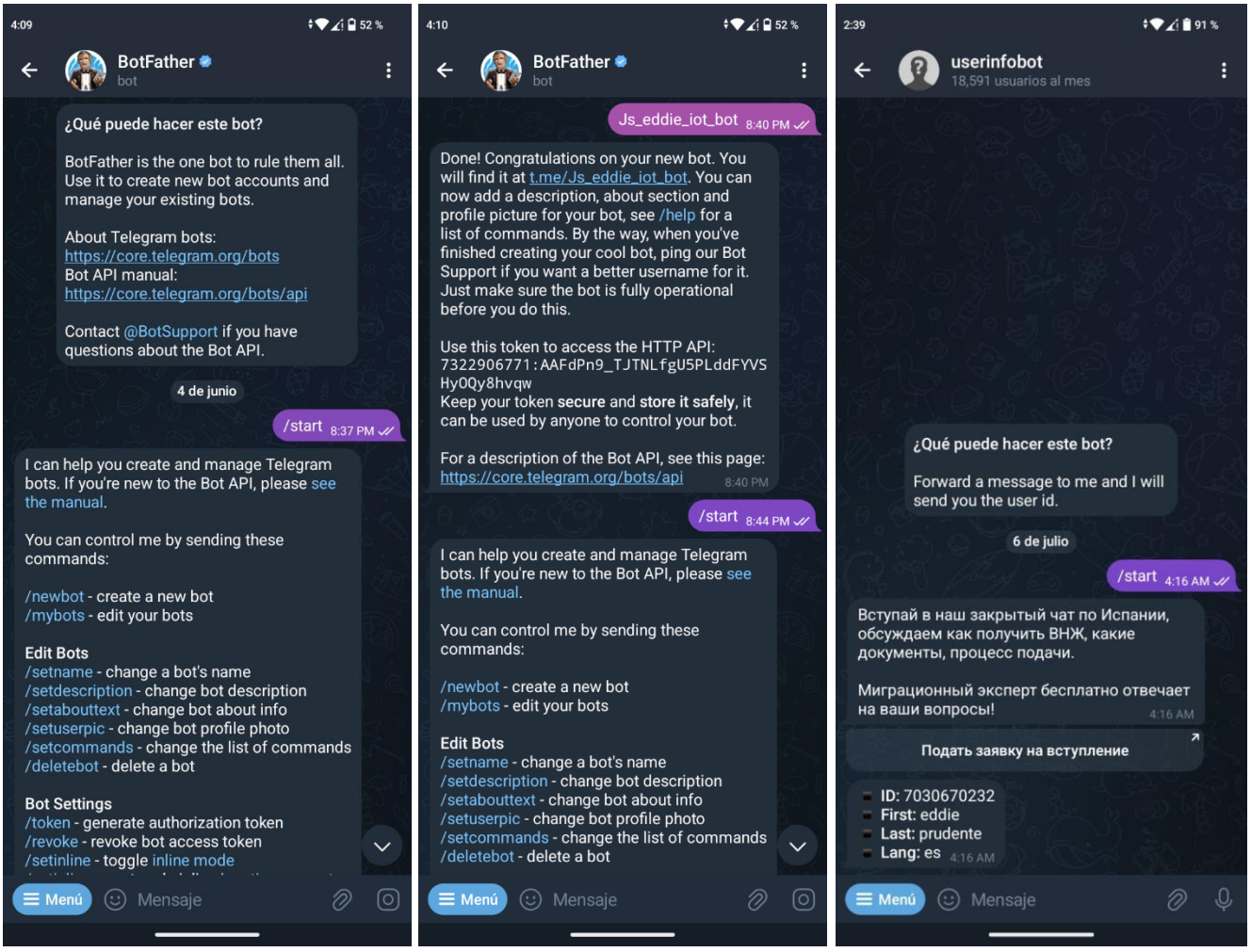


Fig 35. Creación del bot en Telegram

Fuente: Elaborado por autor

Con esos pasos el bot está creado y listo para usarse, sin embargo, se requiere conectar este bot con Google Colab y para eso se necesitará el token asignado. Para finalizar con la configuración del bot, se crea un menú de acciones preventivas que se va a realizar dependiendo del ataque como se ve en la figura 36.

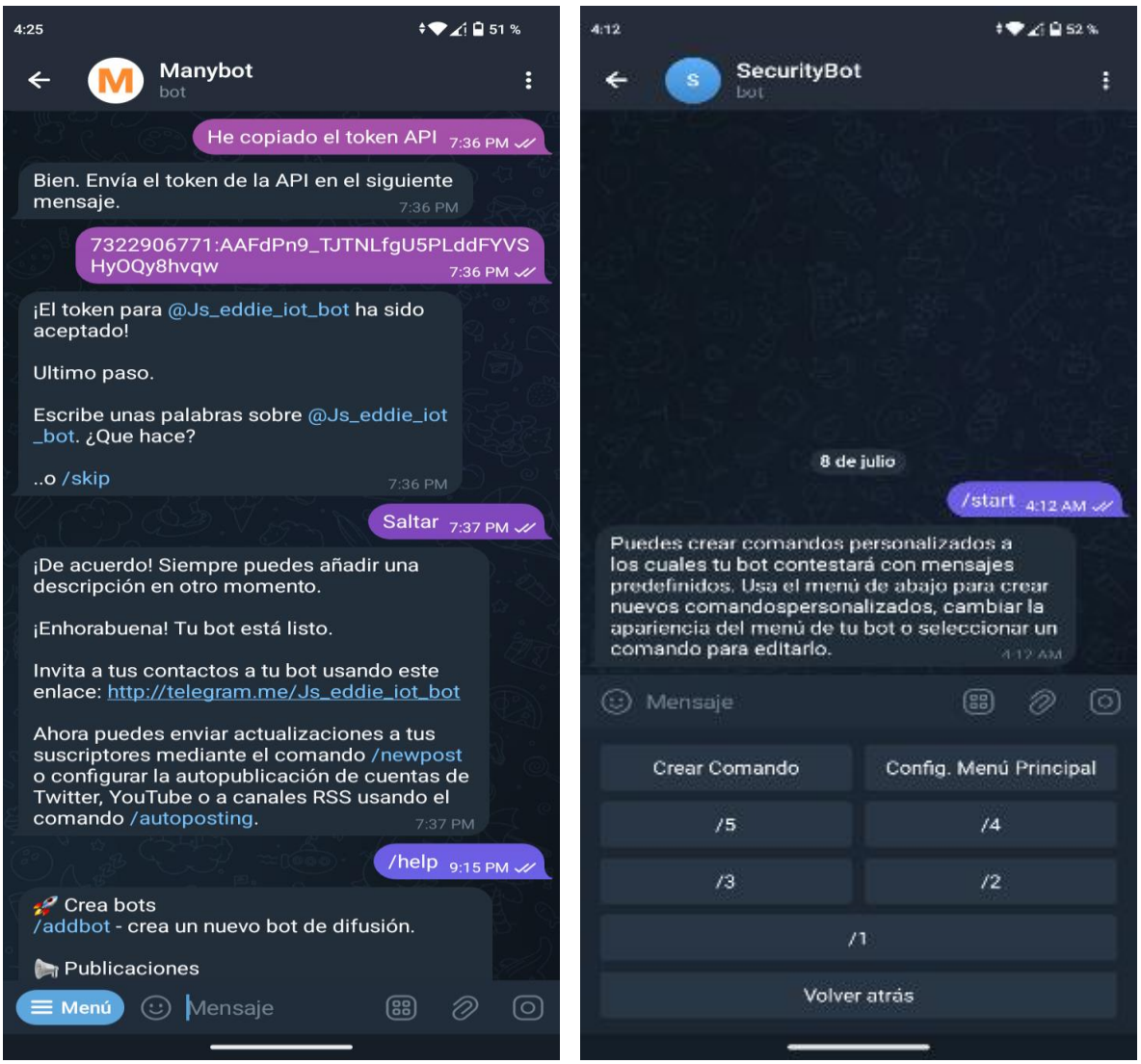


Fig 36. Configuración de comandos para el bot de Telegram

Fuente: Elaborado por autor

Para la creación del menú con las acciones preventivas en cada caso de ataque, se realiza mediante una serie de comandos digitados dentro del bot. A continuación, en la tabla 7 se muestran las configuraciones realizadas por cada ataque:

Tipo de Ataque	Comandos configurados	Pregunta	Respuestas
DDoS	/1		Bloquear temporalmente el acceso a la red

		¿Qué medidas desea implementar?	Activar servicios de mitigación DDoS
			Priorizar el tráfico legítimo
			Escalar la capacidad de la red
Phishing	/2		Alertar a los usuarios potenciales
			Bloquear los enlaces al sitio web falso
			Habilitar la autenticación de dos factores (2FA)
			Deshabilitar cuentas comprometidas
MitM	/3		Revocar los certificados de seguridad
			Cambiar a HTTPS
			Utilizar un VPN
			Escanear la red en busca de dispositivos comprometidos
Malware	/4		Actualizar parches de seguridad
			Aislamiento del sistema infected
			Bloquear puertos abiertos
			Restaurar copias de seguridad
Intrusión	/5		Generar avisos para modificar credenciales
			Activar firewalls
			Enviar reporte al servicio de TI
			Aislar el sistema comprometido

Tabla 7. Configuración de comandos para medidas preventivas

Fuente: Elaborado por autor

El menú de acciones preventivas ante ataques se puede observar en la figura 37:

Fig 37. Menú de acciones preventivas contra ataques configurado

Fuente: Elaboración por autor

3.7 Pruebas de pentesting usando el WiFi Pineapple Mark VII

En esta sección se documenta las pruebas de penetración realizadas utilizando el equipo WiFi Pineapple Mark VII. El objetivo de estas pruebas es evaluar la seguridad de las redes inalámbricas y la susceptibilidad a diversos tipos de ataques. Las pruebas se llevaron a cabo siguiendo un conjunto de procedimientos y técnicas descritas a continuación.

3.7.1 Escaneo de las redes cercanas con "Recon"

Para empezar, en la interfaz del equipo hay que ingresar en la sección "Recon", ahí básicamente se realiza un escaneo de todas las redes WiFi más cercanas al Pineapple, se lo configura para que escanee en un lapso de 30 segundos, al finalizar se mostrarán las redes, ahí se busca la red "LAB_TELECO" quien será a quien se le realice las pruebas de pentesting. En la figura 38 se muestra el resultado del escaneo la red junto a toda la información que esta posee como los dispositivos conectados y al seleccionar la red para hacer las pruebas se muestran opciones que podría realizar a la red y a los dispositivos conectados.

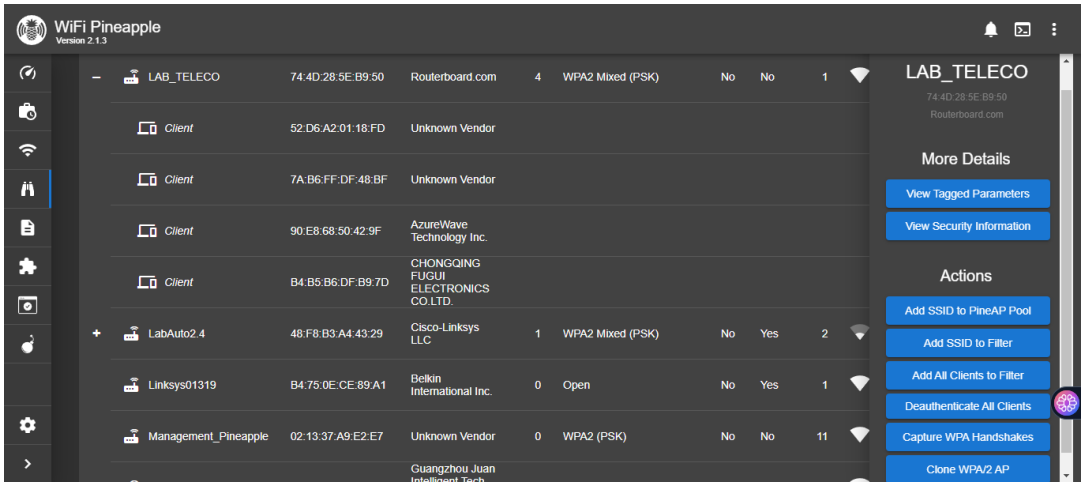


Fig 38. Escaneo de las redes disponibles en el área

Fuente: Elaboración por autor

Como se mencionó anteriormente, el objetivo será la red llamada "LAB_TELECO". Con la opción "Clone WPA/2 AP" se crea un "Evil WPA" que básicamente es un punto de acceso falso que imita a la red normal, esto con el objetivo de utilizar la opción "Deauthenticate all clients" para desautenticar a todos

los que están conectados en la red y poder redirigirlos al Evil WPA. Estos pasos principales deben considerarse al momento de realizar las pruebas de penetración.

3.7.2 Captura de Handshakes

Como primer paso se habilita el Evil WPA para que esté disponible, para pasar a desautenticar a todos los clientes de la red legítima y que se intenten conectar a la red clonada. Se activa la opción de capturar handshakes y al momento en el que el cliente intenta ingresar a la red, automáticamente empieza la captura, en las notificaciones de la Pineapple se mostrará si el handshake se capturó correctamente. En la figura 39 se muestra el historial de los Handshakes capturados:

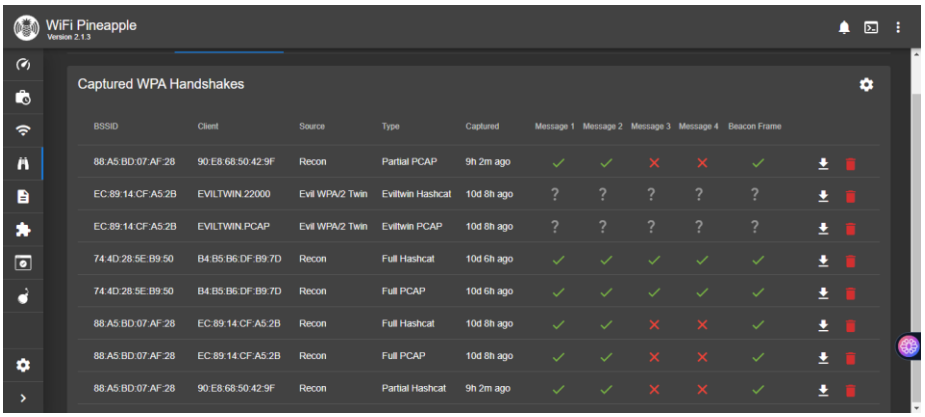


Fig 39. Captura de los Handshakes en el equipo

Fuente: Elaborado por autor

Para comprobar esto se realizó un pequeño código en Google Colab para descifrar uno de esos archivos, con la ayuda de la herramienta Hascat y el diccionario “rockyou.txt” se logró descifrar la contraseña de la red, esto se observa en la figura 40.

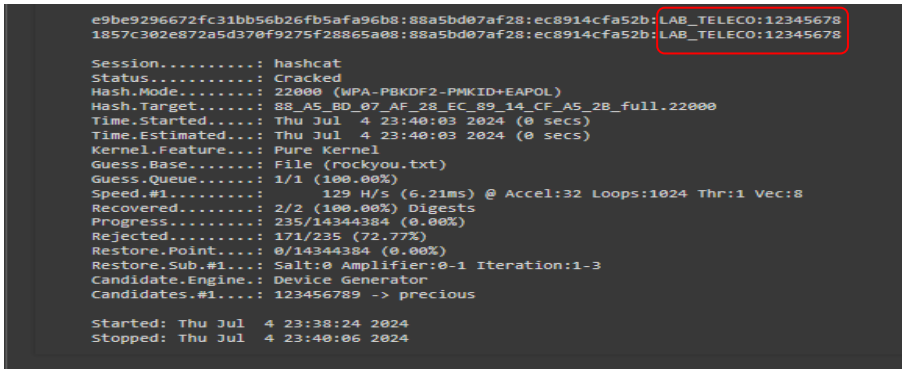


Fig. 40. Handshake crackeado

Fuente: Elaborado por autor

3.7.3 Uso del módulo Evil Portal

Primero se empieza habilitando el AP que imita a la red legítima y se desautentica a los clientes de su red para que sean redirigidos a la red del atacante, para posteriormente iniciar el módulo Evil Portal y activar uno de los portales cautivos, en este caso se activa al portal de “Google” para que la víctima ingrese sus credenciales como se muestra en la figura 41(a). El Pineapple muestra una alerta para indicar que se han capturado datos desde el portal cautivo en el formato LOGS como se muestra en la figura 41(b).

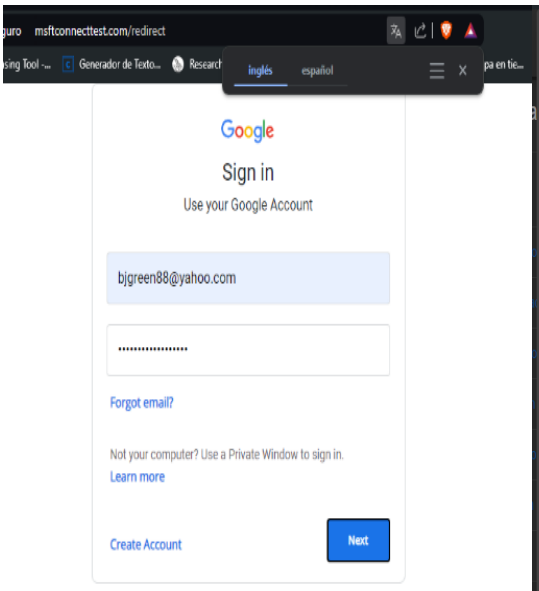


Fig 41.a. Portal cautivo de Google **Fig 41.b.** Credenciales extraídas

Fuente: Elaborado por autor

3.7.4 Uso del módulo HTTPeek

Para empezar, el Evil WPA debe estar habilitado y que las victimas sean redirigidas a dicha red, una vez completado ese paso, se ingresa en el módulo HTTPeek y se habilita la opción “Sniffer” que permitirá observar todas las páginas e imágenes que el cliente visite, así como las cookies de estas. La captura y visualización se encuentra en la figura 42.

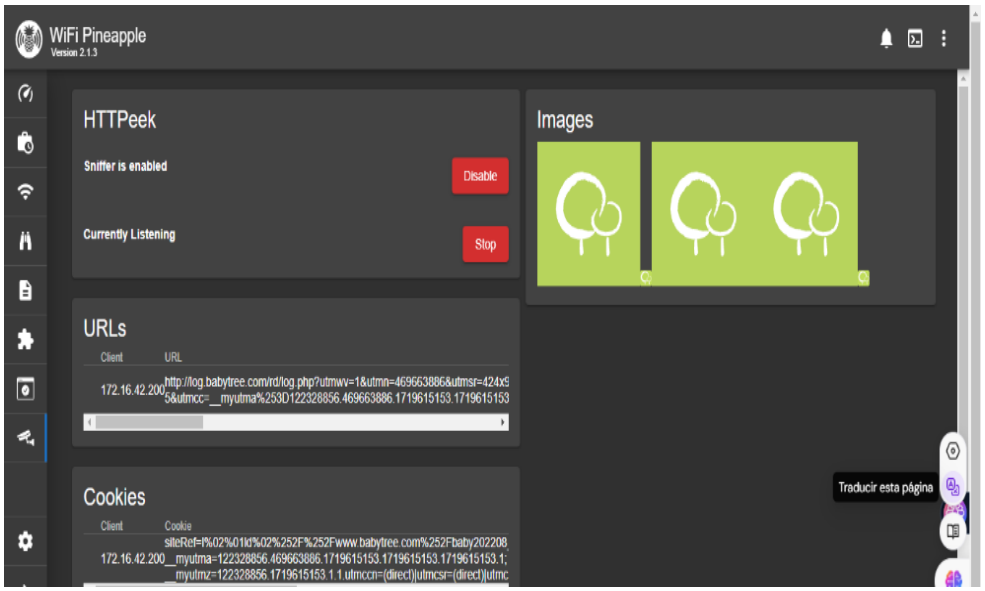


Fig 42. Captura de URLs, Cookies e imágenes

Fuente: Elaborado por autor

3.8 Ataques DoS a la red usando el módulo MDK4 del WiFi Pineapple

Como se mencionó anteriormente, este módulo realiza un tipo de ataque conocido como DoS, permitiendo atacar directamente a un cierto objetivo, ya sea la red en general o simplemente centrarse en un dispositivo, por lo que no es necesario utilizar el AP falso, gracias a los datos extraídos en el escaneo de redes se tiene la dirección MAC de la red y de varios dispositivos. A continuación, se detallan 3 ataques realizados con este módulo.

3.8.1 Primer ataque: Deauthentication and Disassociation

Se seleccionan las interfaces de entrada y salida “wlan1mon” en ambos casos y para mayor precisión se indica que el objetivo se encuentra en el canal 11, se agrega la dirección MAC del objetivo y se empieza a ejecutar el ataque como se muestra en la figura 43(a).

El proceso que realiza es desconectar a la víctima de la red de forma controlada y activa, ya que una vez se inicia el ataque, el dispositivo no podrá conectarse a la red como se muestra en la figura 43(b).

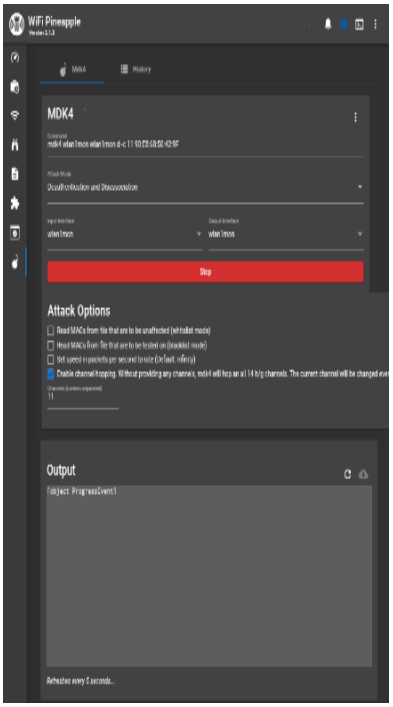


Fig. 43.a. Ejecución del ataque 1

Fig. 43.b. Red sin conexión

Fuente: Elaborado por autor

3.8.2 Segundo ataque: Authentication Denial of Service

La configuración de las interfaces son las mismas que en la anterior, sin embargo, aquí se puede controlar la velocidad de paquetes que se transmite, en este caso se deja por defecto (infinito), al ejecutar el ataque, la víctima presentará problemas con la velocidad del internet ya que este ataque es dirigido a sistemas de autenticación con el objetivo de impedir que los usuarios legítimos accedan a un sistema o servicio. Esto se puede observar en la figura 44(a) en donde la velocidad de descarga es de 79.86Mbps y la de subida es de 63.97Mbps antes de realizar el ataque, indicando una buena conexión, mientras que en la figura 44(b) se muestra una disminución significativa en la velocidad de descarga con un valor de 27.03Mbps y la velocidad de subida con 38.16Mbps cuando se está empleando el ataque.

Fig 44.a. Velocidad de la red antes del ataque

Fig 44.b. Velocidad de la red cuando se emplea el ataque

Fuente: Elaborado por autor

3.8.3 Tercer ataque: EAPOL Start and Logoff Packet Injection

En este ataque la víctima presenta una modificación o inserción de paquetes para provocar desconexiones debido a la inyección de paquetes de inicio y cierre del protocolo de autenticación extensible (EAPOL), en la figura 45(a) se muestra el empleo de dicha técnica, con el fin de realizar un análisis de seguridad. Para finalizar, el equipo Pineapple guarda todos los registros de los ataques realizados, esto se puede observar en la figura 45(b).

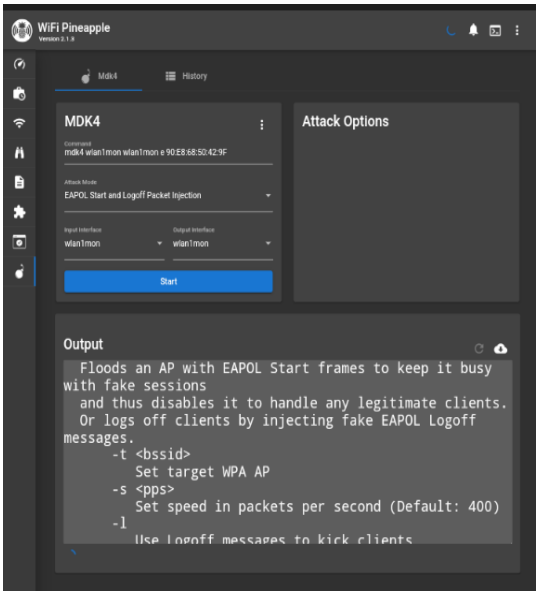


Fig 45.a. Ejecución del ataque 3

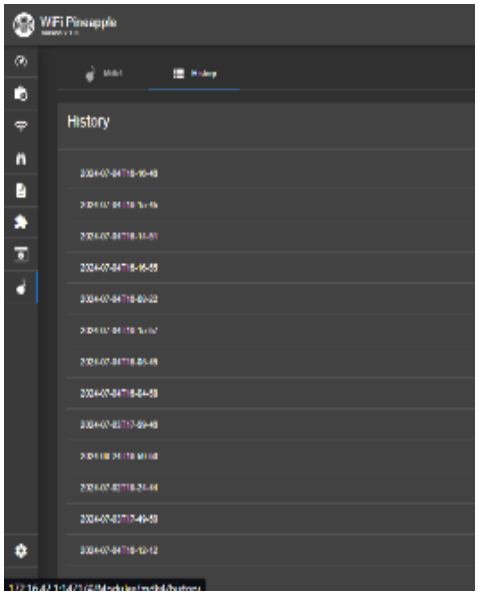


Fig 45.b. Registros de los ataques almacenados

Fuente: Elaborado por autor

3.9 Análisis del tráfico de red usando módulos WiFi Pineapple

Para la realización de esta sección también se utilizaron módulos del Pineapple, ya que el equipo no solo realiza ataques, sino que también permite analizar el tráfico de la red de manera controlada y centralizada en objetivos específicos, a continuación, se detallan los módulos usados y el proceso que se efectuó en cada uno.

3.9.1 Escaneo profundo de puertos con el módulo Nmap del WiFi Pineapple

El módulo Nmap tiene la capacidad de realizar escaneos exhaustivos en una red, brindando información detallada sobre los puertos y servicios en ejecución. En este proyecto, el enfoque se centró en un escaneo intensivo de todos los puertos de la red, con el objetivo de identificar qué puertos están abiertos en los dispositivos conectados y qué servicios (incluyendo sus versiones) se ejecutan en esos puertos. Este proceso es fundamental para identificar posibles vulnerabilidades que puedan ser explotadas.

Para llevar a cabo el escaneo, se accede a la interfaz del módulo Nmap y se configuran los parámetros necesarios, como se muestra en la tabla 8. Uno de los primeros campos a configurar es el "Target", donde se introduce una dirección MAC ficticia. El uso de una dirección MAC falsa permite enmascarar la identidad del atacante en caso de ser detectado durante el proceso, ayudando a ocultar la verdadera dirección MAC del dispositivo desde el cual se ejecuta el ataque.

En el campo "Profile", se selecciona el tipo de escaneo a realizar. En este caso, se eligió el perfil "Intense scan, all TCP ports", que realiza un escaneo profundo y exhaustivo de todos los puertos TCP (Transmission Control Protocol) disponibles en la red. Este perfil es ideal para capturar la mayor cantidad de información posible, ya que examina cada puerto (desde el 1 hasta el 65,535), buscando cualquier posible vulnerabilidad en los dispositivos conectados.

Finalmente, en el campo "Nmap Command", se genera el comando que se ejecutará para realizar el escaneo. Este comando se forma a partir de los parámetros configurados previamente, e incluye tanto la dirección MAC ficticia como la dirección IP del dispositivo o red objetivo. En este caso, el comando es `nmap -p 1-65535 -T4 -A -v 90:E8:68:50:42:9F 192.168.1.1`, donde se especifica que se escanearán todos los puertos TCP (del 1 al 65,535), y se utilizan opciones avanzadas

para obtener información detallada sobre los servicios, versiones, y sistema operativo del host, con la dirección IP objetivo de 192.168.1.1.

Campos	Configuración	Descripción
Target	90:E8:68:50:42:9F	Se configura una dirección MAC ficticia para despistar al usuario si es descubierto
Profile	Intense scan, all TCP ports	El tipo de escaneo que se va a realizar
Nmap Command	Nmap -p 1-65535-T4-A-v 90:E8:68:50:42:9F 192.168.1.1	Es el comando final utilizado para escanear, aquí se especifica el IP de la red a escanear.

Tabla 8. Configuración de los campos en Nmap

Fuente: Elaborado por autor

Una vez ejecutado el comando, empieza el escaneo, dependiendo de la configuración y la dirección IP tomará su tiempo para realizarlo correctamente, al finalizar el Pineapple envía una notificación y el escaneo se guarda en el historial del mismo módulo, se puede visualizar ahí mismo o también está la opción de descargar, en la figura 46 se muestra los datos que recolectó:

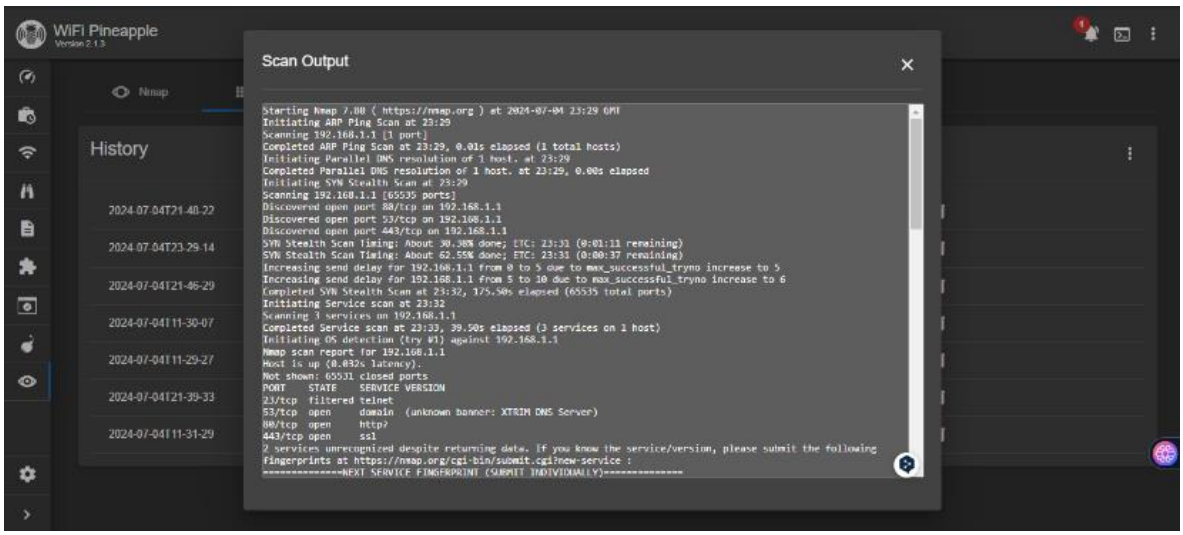


Fig 46. Registro del escaneo realizado

Fuente: Elaborado por autor

El escaneo realizado con Nmap sobre el host 192.168.1.1 reveló información detallada sobre el estado y los servicios activos en dicho dispositivo. El proceso de escaneo se desarrolló en varias etapas, cada una con un objetivo

específico para identificar la disponibilidad, los servicios, y las características del sistema de la máquina. A continuación, se describen las fases clave del escaneo, los tiempos empleados y los resultados obtenidos, tal como se resume en la tabla 9:

- **ARP Ping Scan:** En esta etapa, Nmap envió paquetes ARP (Address Resolution Protocol) al host para determinar si estaba activo en la red local. El resultado mostró que el host estaba en línea y accesible para escaneos posteriores. Este método es muy efectivo en redes locales ya que el ARP no puede ser bloqueado por firewalls.
- **DNS Resolution:** A continuación, se realizó una resolución de DNS para obtener el nombre de dominio asociado a la dirección IP 192.168.1.1. Este paso se completó rápidamente debido a la configuración adecuada del servidor DNS y su proximidad a la red local.
- **SYN Stealth Scan:** Nmap utilizó un escaneo SYN (semiabierto) para sondear los 65,535 puertos TCP del host. Este tipo de escaneo es conocido por ser sigiloso y menos detectable por los sistemas de seguridad, ya que no completa el handshake TCP. El escaneo SYN permitió identificar los puertos abiertos y aquellos que estaban filtrados o cerrados, proporcionando una visión general del estado de la red.
- **Service Scan:** En esta fase, se profundizó en los puertos que fueron identificados como abiertos. Nmap intentó identificar los servicios que se ejecutaban en cada uno de esos puertos, recopilando información clave sobre las versiones de software en uso. Esta etapa es crucial para detectar vulnerabilidades específicas de servicios conocidos.
- **OS Detection:** Finalmente, se intentó determinar el sistema operativo del host mediante el análisis de las respuestas a las solicitudes enviadas. Aunque la detección del sistema operativo no siempre es 100% precisa, proporciona pistas valiosas sobre la plataforma que está siendo utilizada en el host, lo que puede ayudar en la planificación de medidas de seguridad o en la identificación de posibles vulnerabilidades asociadas al sistema.

Etapa del escaneo	Detalles
ARP Ping Scan	Host activo

DNS Resolution	Completado rápidamente
SYN Stealth Scan	65535 puertos escaneados
Service Scan	Servicios escaneados en puertos abiertos
OS Detection	Intento de detección del sistema operativo

Tabla 9. Tabla de etapas del escaneo

Fuente: Elaborado por autor

Los resultados también muestran los estados de los puertos escaneados y los servicios identificados en ellos, con esta información se podría implementar inyección de malware en la red mediante estos puertos. A continuación, se muestra la tabla 10 detallando esta sección:

Puertos y servicios	**Estado**	**Servicio**	**Detalles**
23/tcp	Filtrado	Telnet	---
53/tcp	Abierto	Domain	XTRIM DNS Server
80/tcp	Abierto	HTTP	Respuesta: HTTP/1.0200 OK, múltiples encabezados de seguridad y cookies
443/tcp	Abierto	SSL/HTTPS	---

Tabla 10. Tabla de puertos y servicios

Fuente: Elaborado por autor

También identificó el sistema operativo y red y otros detalles que se reflejan en la tabla 11. Con esto se concluye que el Nmap permite una visión clara del estado y seguridad del host en la red, permitiendo tomar acciones adecuadas de administración y seguridad.

Sistema operativo y red	**Detalles**
Sistema operativo	Linux
Versión de kelnet	Entre 3.10 y 4.11
Distancia de red	1 salto

MAC Address	A0:09:2E:58:5ª:52 (Fabricante desconocido)
Predicción de secuencia TCP	Dificultad 262 (Relativamente segura)
Generación de secuencia IP ID	Todos ceros (baja carga de red o firewall)

Tabla 11. Configuración de los campos en Nmap

Fuente: Elaborado por autor

3.9.2 Captura del tráfico con el módulo TCPDump del WiFi Pineapple

El uso que se le da a este módulo en el proyecto es capturar y mostrar los paquetes de datos que se transmiten y reciben en la red, esto en tiempo real para detectar anomalías o actividades sospechosas, cuenta con filtros para capturar paquetes específicos y exportación de captura del tráfico en formato pcap. La configuración empieza por configurar la interfaz que se analizará, en este caso es la "wlan1mon", en el apartado de Verbose se configura "very very verbose" para aprovechar al máximo la captura con detalle del tráfico, el timestamp configurado por el sistema y sin filtros para capturar todo lo que transita en la red como se muestra en la figura 47:

Fig 47. Configuración y ejecución del escaneo

Fuente: Elaborado por autor

El archivo capturado se guarda en el historial del módulo con extensión PCAP, como se muestra en la figura 48, para visualizar estos datos existen múltiples herramientas, sin embargo, para el proyecto se opta por utilizar Wireshark ya que posee una interfaz intuitiva para realizar el análisis del tráfico capturado.

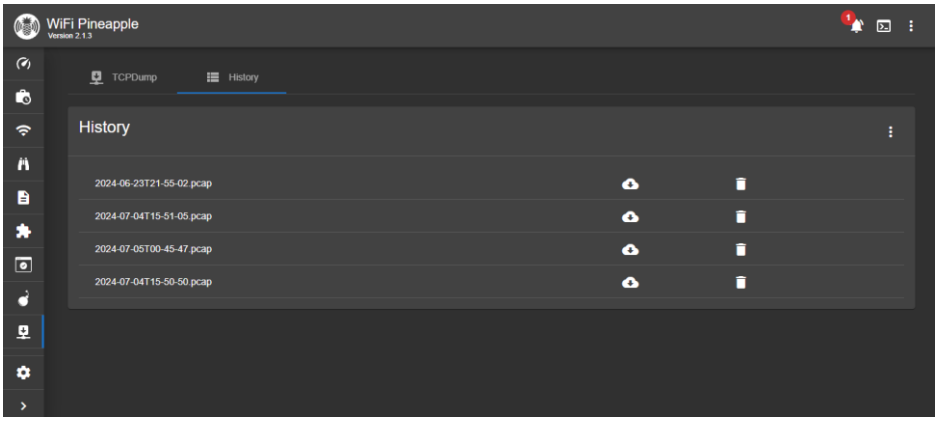


Fig 48. Registro del escaneo realizado

Fuente: Elaborado por autor

3.10 Uso del Wireshark para visualización de archivos PCAP

Para visualizar los archivos PCAP generados por el módulo TCPDump, simplemente hay que descargarlo desde el historial, una vez realizado eso, se debe abrir wireshark y buscar el archivo en el directorio en donde se guardó, en la figura 49 se muestra el tráfico capturado con el módulo TCPDump

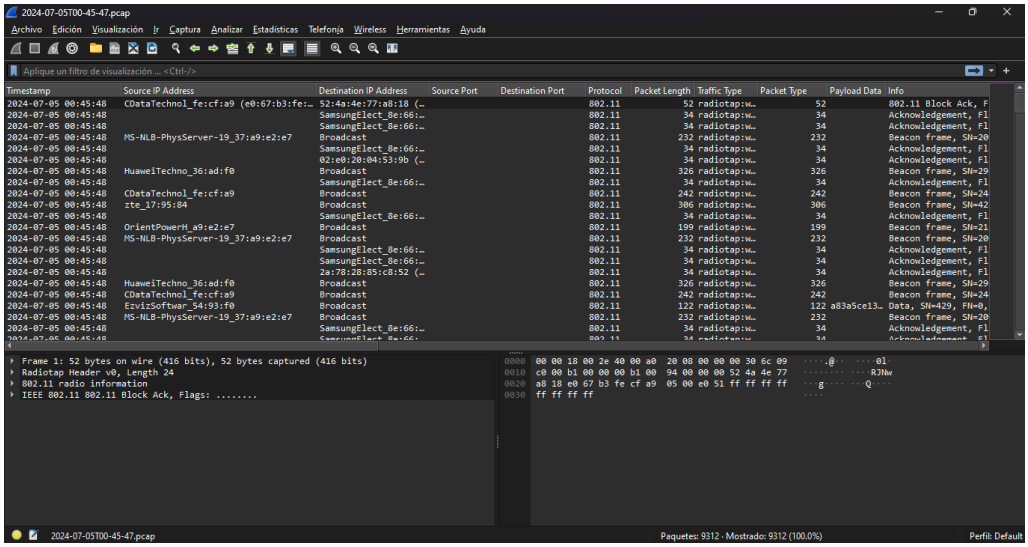


Fig 49. Visualización de la captura de tráfico en Wireshark

Fuente: Elaborado por autor

3.10.1 Análisis de los ataques visualizados en Wireshark

Para identificar los datos relevantes en la ejecución y recolección de tráfico en el ataque “Evil Portal”, se aplicó el filtro **(tcp.stream eq 127)**, el cual permitió aislar un flujo de datos específico asociado a la comunicación entre el cliente y el portal malicioso como se observa en la figura 50.

Al examinar los paquetes correspondientes dentro de este flujo, se identificaron las credenciales ingresadas por la víctima en el portal. Estas credenciales estaban contenidas en el cuerpo de la solicitud HTTP enviada al servidor, específicamente en el método “POST”. Dentro del payload, se observó información sensible como el nombre de usuario y la contraseña, enviados en texto claro debido a la ausencia de cifrado (HTTP en lugar de HTTPS).

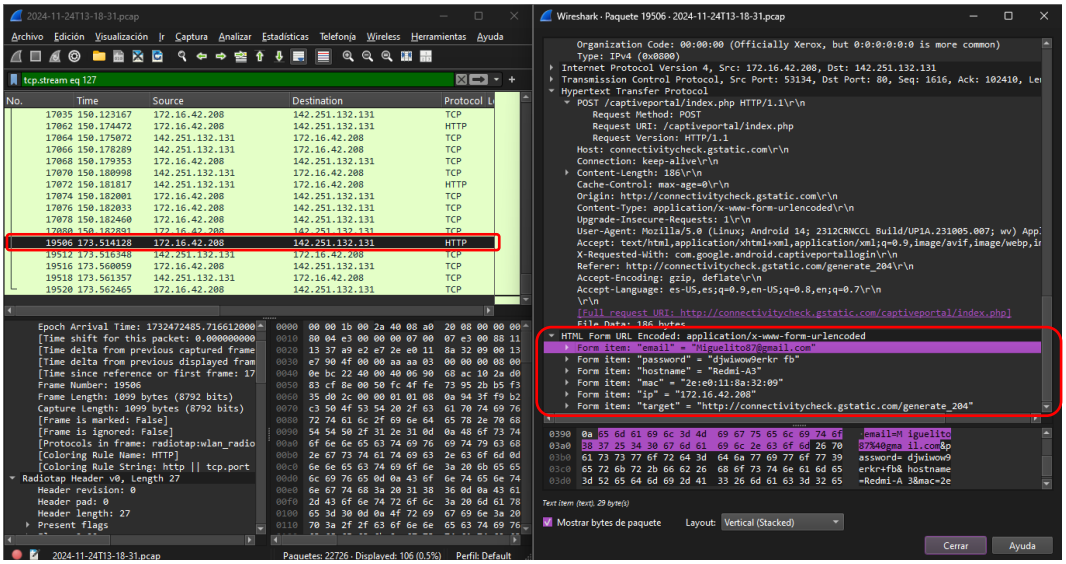


Fig 50. Captura del tráfico del ataque Evil Portal en Wireshark

Fuente: Elaborado por autor

Para los ataques del módulo MDK4 se utilizó primero el filtro **(wlan.fc.type_subtype==0x0c)** dentro del marco IEEE 802.11 en el tráfico del ataque “Deauthentication and Disassociation”, el cual permite visualizar exclusivamente los paquetes de desautenticación.

Cada paquete incluye información como la dirección de destino ya que es el cliente que recibe la instrucción de desautenticación y en la dirección del transmisor se observa la dirección MAC del punto de acceso (AP), simulando ser el emisor legítimo. En la dirección de origen coincide con la del transmisor, reforzando la

legitimidad aparente del paquete y el “Reason Code” está mostrado (0x0000) lo cual indica una desconexión general sin un motivo específico como se muestra en la figura 51.

Los dispositivos afectados intentan reconectarse de manera recurrente al AP. Sin embargo, los paquetes de desautenticación interceptan el proceso, evitando que los clientes restablezcan la conexión, esto genera un bucle continuo de desconexiones, afectando tanto la experiencia de usuario como el rendimiento de la red.

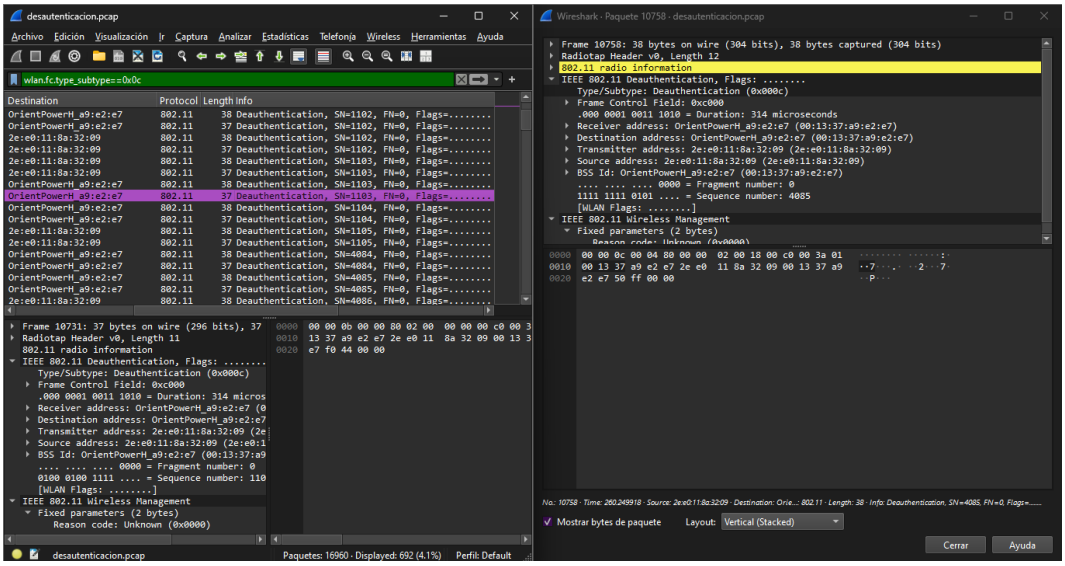


Fig 51. Captura del tráfico del primer ataque del MDK4 en Wireshark

Fuente: Elaborado por autor

Durante el análisis del tráfico capturado en Wireshark que se muestra en la figura 52, se utilizó el filtro **(wlan.fc.type_subtype==0x0B)** para el segundo ataque del módulo MDK4, diseñado para identificar exclusivamente las tramas de autenticación. Se identificaron tramas de autenticación repetitivas y se observa un patrón de solicitudes de autenticación hacia el punto de acceso **“OrientPowerH_a9:e2:e7”**, lo que evidencia un comportamiento anómalo generado por el ataque y las direcciones de origen cambian constantemente, simulando múltiples dispositivos (direcciones MAC spoofed). Esto refuerza la carga sobre el AP al obligarlo a procesar cada solicitud como si proviniera de dispositivos legítimos y los números de secuencia progresivos indican la frecuencia de las solicitudes y las tramas de 54 bytes son consistentes con el tamaño típico de

una solicitud de autenticación en el protocolo 802.11. En el tráfico, los tiempos entre las tramas de autenticación son extremadamente cortos, lo cual no es típico en escenarios de tráfico normal y el AP objetivo puede llegar a colapsar debido al exceso de solicitudes, resultando en una denegación de servicio para los clientes legítimos.

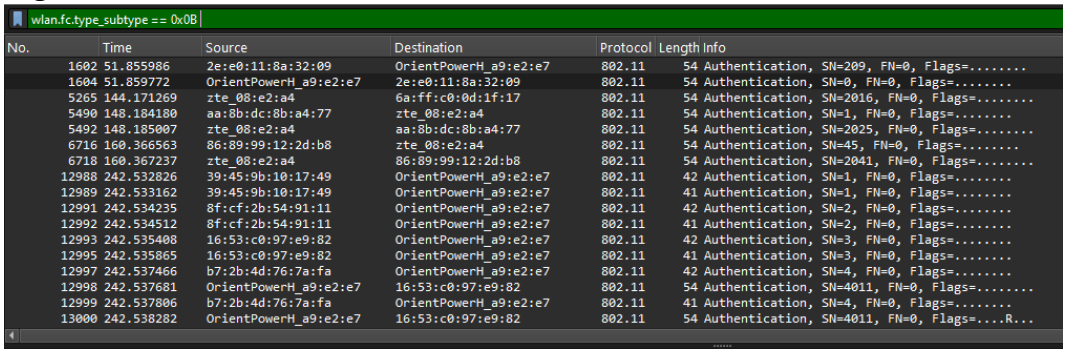


Fig 52. Captura del tráfico del segundo ataque del MDK4 en Wireshark

Fuente: Elaborado por autor

Para el último ataque llamado “EAPOL Start and Logoff Packet Injection”, se inyectan paquetes falsificados EAPOL Start y EAPOL Logoff con el objetivo de interrumpir el proceso de autenticación o desconectar dispositivos de la red inalámbrica. El filtro **(eapol)** en Wireshark muestra exclusivamente tramas relacionadas con el protocolo EAPOL.

La presencia recurrente de paquetes “EAPOL Logoff” que se muestra en la figura 53 es un comportamiento inusual en condiciones normales, este tipo de paquetes son utilizados para indicar la finalización de una sesión, pero en este ataque se inyectan de manera forzada para interrumpir la conexión de los clientes con el punto de acceso (AP).

Las direcciones de origen y destino refuerzan la identificación del ataque ya que en la dirección de origen aparece como Broadcast o la dirección MAC del cliente y en la dirección de destino apunta al AP, en este caso “OrientPowerH_a9:e2:e7”. por otro lado, los paquetes Logoff presentan una duración de 314 microsegundos, lo cual es consistente con tramas falsificadas de desconexión, también en la captura se observa una alta frecuencia de paquetes Logoff, lo que evidencia la inyección continua como parte del ataque.

En el detalle de la trama EAPOL Logoff (paquete 4679), se puede observar el Type/Subtype: Data (0x0020), lo que corresponde a datos de gestión y en BSSID y dirección de transmisión apuntan al AP “OrientPowerH_a9:e2:e7”, el paquete tiene como objetivo desconectar múltiples dispositivos.

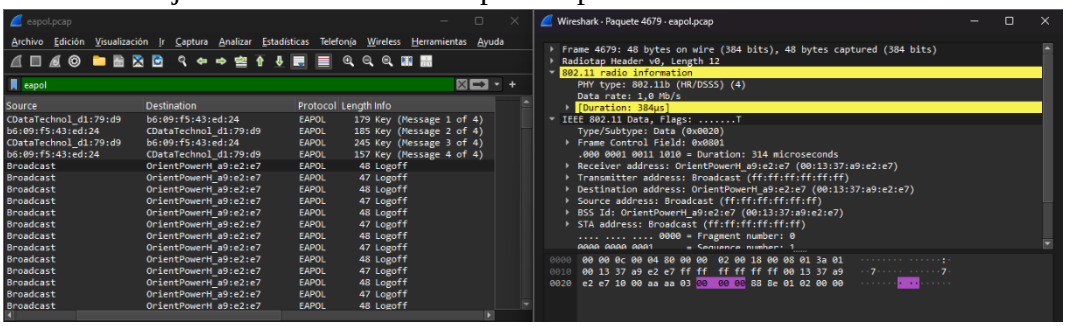


Fig 53. Captura del tráfico del tercer ataque del MDK4 en Wireshark

Fuente: Elaborado por autor

A continuación, se muestra en la tabla 12 de los parámetros a considerar para determinar que en una captura de tráfico de red existe un ataque como los que se realizó.

Parámetros	Descripción
Protocolo	Se considera el tipo de tráfico (802.11, EAPOL, HTTP, TCP, etc)
Tipo/Subtipo de tramas	Identificación específica de las tramas, por ejemplo “wlan.fc.type_subtype” o “tcp.stream”.
Tamaño de las tramas	Consistencia en tamaños anómalos para cierto tipo de tráfico.
Dirección MAC (Origen y Destino)	Existen presencia de spoofing (Direcciones MAC aleatorias o falsas)
Dirección IP (Origen y Destino)	Tráfico dirigido a servidores maliciosos o inusuales como el de un portal cautivo.

Puertos	Uso de puertos estandar o no estándar en actividad sospechosa
Broadcast	Tráfico dirigido a todos los dispositivos de la red con dirección ff:ff:ff:ff:ff:ff
Números de secuencia	Incrementos anómalos o repeticiones sospechosas
Información (códigos de estado o razón)	Razones de desconexión o mensajes específicos.

Tabla 12. Especificaciones que determinan un ataque en la red

Fuente: Elaborado por autor

Una vez capturado el tráfico de red tanto de wireshark como de los módulos del Pineapple, se procede a exportarlo en archivo CSV, puesto que ese es el formato que se manejará para el entrenamiento de los algoritmos de machine learning, en la figura 54 se muestran carpetas con los archivos en ambos formatos (CSV y PCAP).

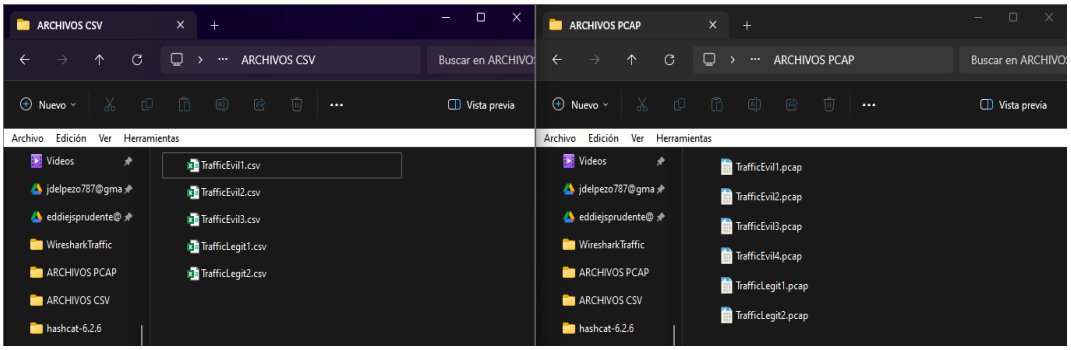


Fig 54. Exportación del formato PCAP a formato CSV

Fuente: Elaborado por autor

3.11 Consideraciones para el modelo de aprendizaje supervisado

Para llevar a cabo la implementación de modelos de aprendizaje supervisado, hay que considerar ciertos factores, en primera instancia, el entorno en donde se llevará a cabo, para ello se usará la plataforma de Google Colab, también se deben considerar las métricas de evaluación que se detallan en la tabla 13:

Métrica	Descripción	Fórmula
Matriz de confusión	Cantidad de datos correcta e incorrectamente clasificados (TN, TP altos; FN, FP bajos).	N/A
Exactitud (Accuracy)	Proporción de datos clasificados correctamente.	$\frac{(TP + TN)}{(TP + TN + FP + FN)}$
Precisión	Proporción de positivos predichos que son realmente positivos.	$\frac{TP}{(TP + FP)}$
Recuperación (sensibilidad)	Capacidad del modelo para identificar casos positivos.	$\frac{TP}{(TP + FN)}$
Puntuación F1	Media ponderada de la precisión y la recuperación.	$2 \times \frac{(Precisión \times Recuperación)}{(Precisión + Recuperación)}$
Error Absoluto Medio (MAE)	Media de los errores absolutos entre predicciones y valores reales.	N/A
Error Cuadrático Medio (RMSE)	Desviación estándar de los errores de predicción.	N/A
Puntuación de Varianza	Dispersión de los errores en los datos; alta varianza indica datos dispersos.	N/A
Validación Cruzada	Evalúa el rendimiento del modelo y ayuda a encontrar el mejor modelo.	N/A

Tabla 13. Tabla de métricas de evaluación

Fuente: Elaborado por autor

En el aprendizaje supervisado la máquina posee datos iniciales asignados y un solo dato de salida para que el modelo logre aprender, la fase final es cuando se generan nuevas variables de entrada capaz de predecir la salida con una gran precisión. Dentro de esta categoría incluye la clasificación que saca una conclusión a partir de datos observados previamente, regresión que comprende la relación entre dos objetos y la previsión que hace forecasting tomando como referencia datos históricos. Las librerías por utilizar en el desarrollo de todo el algoritmo son:

- Pandas
- Numpy
- Matplotlib.pylot
- Seaborn

- sklearn

También, se ha considerado el uso de los siguientes algoritmos de aprendizaje automático supervisado:

- Regresión lineal (SRL)
- Regresión logística (RL)
- Random Forest (RF)
- Extreme Gradient Boosting (XGBoost)
- Decision Tree (DT)
- K-Vecinos más cercanos (KNN)
- Máquina de vectores de soporte (SVM)

Por último, también se considera el tipo de ataque a identificar, para la realización del proyecto se realizó varias pruebas de penetración y ataque a la red, esos ataques se agruparán en estos 3 campos como se muestra en la tabla 14 para adaptar de mejor manera el tráfico de la red y el dataset de entrenamiento

Tipo de Ataque	Ataques realizados con la Pineapple
DDoS	MDK4: Deauthentication and Disassociation, Authentication Denial of Service, EAPOL Start and Logoff Packet Injection
Phishing	Captura de Handshake. Evil Portal
Man in the Middle	HTTPeek

Tabla 14. Clasificación de los ataques realizados

Fuente: Elaborado por autor

3.12 Preparación de los datos del Dataset

El primer paso por seguir para el análisis de datos es realizar una preparación de estas, con la finalidad de minimizar los riesgos de ruido o datos innecesarios que podrían ocasionar fallos en el entrenamiento. Este proyecto utiliza el dataset para el entrenamiento llamado "cybersecurity_attacks.csv", este es un conjunto de datos en donde se han registrado ataques de ciberseguridad, esta información se encuentra en la plataforma Kaggle [27]. El dataset cuenta con 25 campos encontrados y un total de 40000 registros de muestra en la tabla 15.

#	Campo	Tipo de Datos	Descripción del Campo
1	*Timestamp*	*datetime*	Fecha y hora en que ocurrió un evento
2	*Source IP Address*	*string*	Dirección IP de Origen
3	*Destination IP Address*	*string*	Dirección IP de Destino
4	*Source Port*	*int*	Puerto de Origen
5	*Destination Port*	*int*	Puerto de Destino
6	*Protocol*	*string*	Protocolo [ICMP, UDP, Otro]
7	*Packet Length*	*int*	Longitud del Paquete [64-1500 bytes]
8	*Packet Type*	*string*	Tipo de Paquete [Control, Dato]
9	*Traffic Type*	*string*	Tipo de Trafico [DNS, HTTP, Otro]
10	*Payload Data*	*string*	Datos de Carga útil
11	*Malware Indicators*	*string*	Indicador de *Malware*
12	*Anomaly Scores*	*float*	Puntuaciones de anomalía
13	*Alerts/Warnings*	*string*	Alertas/Advertencias [Alerta activada]
14	*Attack Type*	*string*	Tipo de ataque [DDoS, *Malware*, Otro]
15	*Attack Signature*	*string*	Firma del Ataque [Patrón conocido]
16	*Action Taken*	*string*	Acción Tomada [Bloquear, Ignorar, Otro]
17	*Severity Level*	*string*	Nivel de Severidad [Media, Alta, Otro]
18	*User Information*	*string*	Información del Usuario
19	*Device Information*	*string*	Información del Dispositivo
20	*Network Segment*	*string*	Segmento de Red [Segmento A, B, C, Otro]
21	*Geo-location Data*	*string*	Datos de Geolocalización
22	*Proxy Information*	*string*	Información del Proxy
23	*Firewall Logs*	*string*	Registro del Firewall [Si, No]
24	*IDS/IPS Alerts*	*string*	Alerta IDS/IPS [Datos de Alerta]
25	*Log Source*	*string*	Origen del Log [Firewall, Server]

Tabla 15. Campos y tipos de datos

Fuente: Kaggle.com [27]

3.13 Análisis exploratorio del Dataset obtenido

Luego de cargar el dataset en el entorno de Google Colab, se empieza a digitar una serie de comandos para poder visualizar las dimensiones del dataset y la cantidad de datos disponibles para el análisis, así confirmamos que tiene 40000 registros y 25 campos diferentes, estos campos también se visualizan sus nombres confirmando los datos de la tabla 15, también se visualiza el tipo de datos de cada columna para saber si se está trabajando con números o datos de tipo objeto y al final se imprime en una tabla los 4 primeros registros para verificar que no haya ningún problema en el formato de presentación.

Se identifican y se realiza un conteo de valores nulos en el dataset para determinar si es necesario realizar imputaciones o eliminar datos faltantes, lo mismo para con el registro de datos duplicados, se corrobora que no existan duplicados en el dataset para evitar sesgos en el análisis y por último se realiza una tabla estadística de los datos numéricos antes de cualquier tratamiento para entender mejor su distribución y detectar posibles anomalías.

3.13.1 Análisis del dataset mediante gráficos

Se crean histogramas para visualizar la distribución de las variables numéricas y detectar posibles patrones en los datos, también se realizan gráficas circulares en donde se muestra mejor la distribución de cada campo, como lo es el protocolo, el tipo de paquete, tipo de tráfico, niveles de seguridad, segmentos de la red para identificar áreas más vulnerables, distribución de los tipos de ataques y la visualización de acciones tomadas para evaluar la respuesta ante incidentes de seguridad.

3.14 Preprocesamiento de datos del dataset

Una vez analizado todo lo que contiene el dataset que se va a utilizar, se empieza realizando un tratamiento de valores nulos para asegurar que el dataset no contenga campos vacíos, si en dado caso existen datos nulos se procede a rellenar con valores numéricos que no afecten al análisis, este proceso se realiza con los campos que servirán como detección de anomalías, se le asignan valores de 1 y 0 dependiendo del caso.

Etiqueta	Malware Indicators	Alerts/Warnings	Firewall Logs	IDS/IPS Alerts
0	None Detected	No Alert	No Logs	Normal
1	IoC Detected	Alert Triggered	Log data	Alert Data

Tabla 16. Etiqueta de valores nulos

Fuente: Elaborado por autor

3.14.1 Diccionario de mapeo de datos obtenidos del dataset

Antes de realizar este procedimiento, se plantea una codificación de las variables categóricas para facilitar el uso de algoritmos de machine learning, posteriormente se empiezan a crear diccionarios para mapear valores categóricos a numéricos permitiendo una interpretación más fácil y uso eficiente de algoritmos

3.14.2 Gestión de datos anómalos del dataset

Se emplea esta acción para eliminar columnas con datos no numéricos o anómalos para limpiar el dataset de elementos no relevantes y mejorar la precisión del modelo, se realiza un escalado de variables numéricas para asegurar que todas tengan una escala comparable y mejorar el rendimiento del modelo.

3.15 Uso de variables para los modelos SML

Como se mencionó anteriormente, el proyecto incluye el uso de varios modelos de aprendizaje automático supervisado, pero antes de implementar cada uno de ellos, es necesario seleccionar las variables del modelo, se definen las variables de entrada que en este caso son entradas numéricas y entradas categóricas, para un mayor impacto se define una entrada mixta de ambos. No existe un límite máximo de entradas que se pueden evaluar, sin embargo, hay que tener en cuenta los recursos computacionales ya que la complejidad del entrenamiento y la inferencia con el número de variables deben ser sujetas a los recursos que ofrece la plataforma, también la dimensionalidad de los datos ya que un número de datos elevados puede generar un sobreajuste que podría afectar a la generalización del modelo y por último considerar la redundancia, si existen valores correlacionadas es mejor evitarlas ya que podrían afectar negativamente el rendimiento del entrenamiento.

3.15.1 División de datos para entrenamiento y pruebas del dataset

Se realiza una separación del dataset en conjuntos de entrenamiento y prueba para evaluar la capacidad predictiva del modelo, en este caso se divide en:

25% para pruebas	10k/40k registros
75% para entrenamiento	30k/40k registros

Tabla 17. Distribución de datos para entrenamiento y prueba

Fuente: Elaborado por autor

3.16 Creación y entrenamiento de los modelos SML

Una vez segmentado el porcentaje de datos a utilizar para el entrenamiento y prueba, se empieza la creación de los códigos llamando a la función de cada algoritmo mediante el recurso Sklearn, este contiene librerías con todos los algoritmos a utilizar.

3.16.1 Regresión lineal [SLR]

Se implementa la regresión lineal para predecir valores continuos de datos, permitiendo trazar una línea sobre los datos. Esta "línea" puede ser recta, con picos o con formas redondeadas, pero mantendrá la continuidad en el gráfico resultante. Se puede obtener n coeficientes (cada uno corresponde a las n variables predictivas), también se define la proporción de datos asignados al conjunto de pruebas que será de un 0.25. El código para crear y entrenar el modelo es el siguiente:

```
Crear el modelo
model_slr = linear_model.LinearRegression()
Entrenar el modelo
model_slr.fit(X_train, y_train)
```

3.16.2 Regresión Logística [LR]

Este algoritmo de aprendizaje automático se utiliza para clasificar datos en dos categorías (0 o 1). Se le presenta un conjunto de datos de entrenamiento con ejemplos ya etiquetados como 0 o 1. El algoritmo aprende de estos datos y crea una curva en forma de S (llamada sigmoide) que puede usarse para predecir la categoría de nuevos datos. La curva sigmoide asigna a cada nuevo punto de datos una

probabilidad entre 0 y 1 de pertenecer a la categoría 1. El algoritmo utiliza técnicas matemáticas para identificar relaciones entre las variables de los datos y luego usa esas relaciones para predecir el valor de una variable en función de la otra. El código para crear y entrenar el modelo es el siguiente, aquí se configura el 1000 como número máximo de iteraciones necesarias para que los solucionadores converjan:

```
Crear el modelo
model_lr = LogisticRegression(max_iter=1000)
Entrenar el modelo
model_lr.fit(X_train, y_train)
```

3.16.3 Random Forest [RF]

Este es un algoritmo de tipo ensamble que se utiliza para clasificación y regresión, algo así como imaginar a un grupo de expertos que votan para tomar una decisión, funciona de manera similar, pero en lugar de expertos, utiliza varios árboles de decisión. Cada árbol de decisión es como un experto individual, y cada uno analiza una parte diferente de los datos. Al final, se suman los "votos" de todos los árboles de decisión para obtener la predicción final. Este enfoque, llamado "Bosque Aleatorio" (RF), es más preciso y confiable que depender de un solo árbol de decisión. El código para crear y entrenar el modelo es el siguiente:

```
Crear el modelo
model_rf
=RandomForestClassifier(n_estimators=100,criterion='gini')
Entrenar el modelo
model_rf.fit(X_train, y_train)
```

El estimador indica la cantidad RF y el criterio “Gini” funciona para medir la cantidad de una división, el grado de impureza, 'log_loss' y 'entropy', ambos para la ganancia de información de Shannon. Esta función mide el desorden o mezcla que quedan en los nodos una vez divididos.

3.16.4 Extreme Gradient Boosting [XGBoost]

Al igual que el anterior, se utiliza para la clasificación y regresión, pero a un nivel diferente, la analogía que se podría usar es imaginando a un equipo de expertos trabajando juntos para resolver un problema, cada experto tiene su propia perspectiva y forma de analizar la información. XGBoost funciona de manera similar, pero en lugar de expertos, utiliza árboles de decisión. A medida que se

agregan más árboles de decisión al equipo, este aprende a corregir los errores de los árboles anteriores y mejora la precisión general. Al final, el equipo de árboles de decisión trabaja en conjunto para hacer la mejor predicción posible. El código para crear y entrenar el modelo es el siguiente, aquí se estima el número de rondas de refuerzo:

```
Crear el modelo
model_xgb = XGBClassifier(n_estimators=100)
Entrenar el modelo
model_xgb.fit(X_train, y_train)
```

3.16.5 Árbol de decisión [DT]

Se compone principalmente de nodos y ramas, cada rama representa una posible decisión que se puede tomar en un punto determinado del proceso. Las hojas del árbol representan las predicciones finales. Para llegar a una predicción, el algoritmo recorre el árbol de arriba hacia abajo, haciendo preguntas sobre las variables del problema. En cada nodo, se toma una decisión en función de la respuesta a la pregunta. Este proceso se repite hasta que se llega a una hoja del árbol, que nos da la predicción final. A continuación, se muestra la creación y entrenamiento del modelo:

```
Crear el modelo
model_dt = DecisionTreeClassifier()
Entrenar el modelo
model_dt.fit(X_train, y_train)
```

3.16.6 K-Vecinos más cercanos [KNN]

Este algoritmo es de una estructura simple pero efectiva para resolver una amplia variedad de problemáticas, al imaginar que se tiene un conjunto de datos con puntos de diferentes colores, cada punto representa un ejemplo de datos y su color representa la clase a la que pertenece. Así se tiene un nuevo punto de datos que quieres clasificar. El algoritmo KNN funciona como si preguntara a sus vecinos más cercanos (los K puntos de colores más cercanos al nuevo punto) qué clase creen que es. La clase que más se repite entre esos vecinos es la que se asigna al nuevo punto. A continuación, se muestran las líneas de código para la creación del modelo y entrenamiento, pero antes se definen los hiperparámetros como el número de vecinos para la consulta y el peso auniforme, es decir, la ponderación de los vecinos:

```
Definir los hiperparámetros
n_neighbors = 5
weights = 'uniform'
Crear el modelo
model_knn = KNeighborsClassifier(n_neighbors=n_neighbors,
weights=weights)
Entrenar el modelo
model_knn.fit(X_train, y_train)
```

3.16.7 Máquina de Vector de Soporte [SVM]

El algoritmo SVM es como un entrenador que aprende a identificar diferentes tipos de ataques en base a ejemplos. Para ello, el entrenador recibe un conjunto de datos de entrenamiento (X_train) que contiene ejemplos de ataques junto con su tipo correspondiente (y_train). El entrenador analiza estos ejemplos y aprende a reconocer patrones que diferencian un tipo de ataque de otro. Una vez que el entrenador está entrenado, se le presenta un nuevo conjunto de datos (X_test) que contiene ataques sin clasificar. El entrenador utiliza lo que ha aprendido para predecir el tipo de ataque para cada uno de estos nuevos ejemplos. Finalmente, se evalúa el rendimiento del entrenador utilizando diferentes medidas, al igual que se evalúa el rendimiento de otros entrenadores. A continuación, se muestran las líneas de código para la creación y entrenamiento del modelo:

```
Crear el modelo
model_svm = SVC()
Entrenar el modelo
model_svm.fit(X_train, y_train)
```

3.17 Predicción de los modelos SML entrenados

Una vez creado y entrenado los modelos, se realiza el proceso de predicción con el conjunto de pruebas para evaluar el rendimiento de los modelos, en la tabla 18 se muestran los detalles luego de la ejecución de esta sección:

Modelo ML	CPU Times	Wall time	Información
SLR	3.66 ms (usuario), 17 µs (sistema), 3.67 ms (total)	3.71 ms	Este modelo tiene un tiempo de predicción muy rápido, lo que sugiere que es computacionalmente eficiente.

LR	4.65 ms (usuario), 5.92 ms (sistema), 10.6 ms (total)	5.3 ms	Ligeramente más lenta que la regresión lineal simple debido a la naturaleza iterativa de su optimización.
RF	407 ms (usuario), 101 ms (sistema), 508 ms (total)	394 ms	Toma más tiempo en comparación con los modelos anteriores debido a la necesidad de evaluar múltiples árboles de decisión.
XGBoost	121 ms (usuario), 0 ns (sistema), 121 ms (total)	63.9 ms	Más rápido que el anterior, lo que refleja su optimización para eficiencia computacional debido a sus mejoras en algoritmos de boosting.
DT	13.7 ms (usuario), 0 ns (sistema), 13.7 ms (total)	6.75 ms	Los árboles de decisión individuales son muy rápidos en la predicción. debido a su estructura de evaluación directa de reglas de decisión.
KNN	710 ms (usuario), 72.3 ms (sistema), 782 ms (total)	722 ms	El KNN es considerablemente más lento porque requiere calcular la distancia a todos los puntos de datos en el conjunto de entrenamiento para cada predicción, lo que es computacionalmente costoso.
SVM	22 s (usuario), 18.9 ms (sistema), 22 s (total)	22.1 s	La SVM es la más lenta de todos los modelos presentados. Esto es típico, ya que las SVM pueden ser muy costosas computacionalmente, especialmente con conjuntos de datos grandes y complejos.

Tabla 18. Resultados de la predicción de los modelos SML

Fuente: Elaborado por autor

3.17.1 Evaluación de modelos SML

Para entrenar y validar el modelo de forma robusta, se divide los datos en dos conjuntos: entrenamiento (75%) y prueba (25%). El conjunto de entrenamiento se utilizará para entrenar el modelo, mientras que el conjunto de prueba se utilizará para evaluar su rendimiento final, esto se explicó en secciones anteriores.

Sin embargo, en lugar de utilizar una simple validación con el conjunto de prueba, emplearemos una técnica llamada Validación Cruzada K-fold. Esta técnica consiste en dividir el conjunto de entrenamiento en 5 subconjuntos (folds) de igual tamaño. En la figura 55 se observa un diagrama de flujo típico de la validación cruzada en los entrenamientos de diversos modelos de ML.

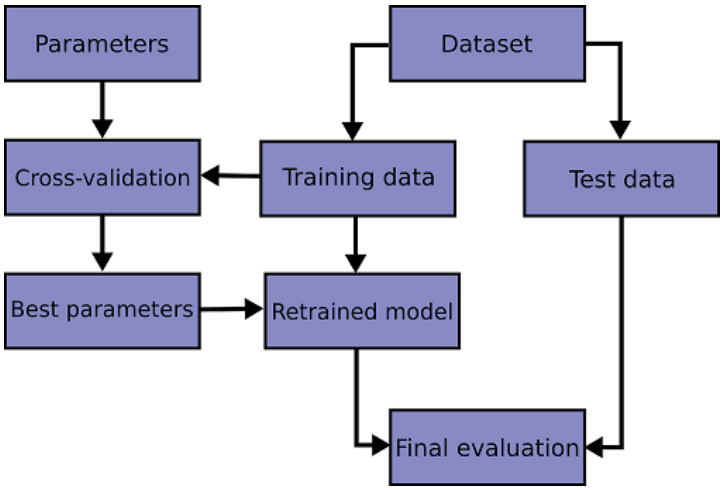


Fig 55. Diagrama de flujo de Validación Cruzada

Fuente: scikit-learn.org [39]

A continuación, se entrena el modelo 5 veces, utilizando 4 de los folds para el entrenamiento y el fold restante para la validación. Esto permite obtener un promedio de precisión más preciso, ya que no depende de un solo conjunto de validación. En cada iteración de la Validación Cruzada K-fold:

- Se aparta un fold de 8.000 muestras del conjunto de entrenamiento.
- Se entrena el modelo con las 32.000 muestras restantes.
- Se evalúa la precisión del modelo sobre las 8.000 muestras apartadas.

Al completar las 5 iteraciones, se obtiene 5 valores de precisión. Si estos valores son similares entre sí, indica que el modelo está funcionando bien y que no está sobreajustándose a los datos de entrenamiento. En la figura 56 se muestra el funcionamiento de la técnica k-fold con 5 splits:

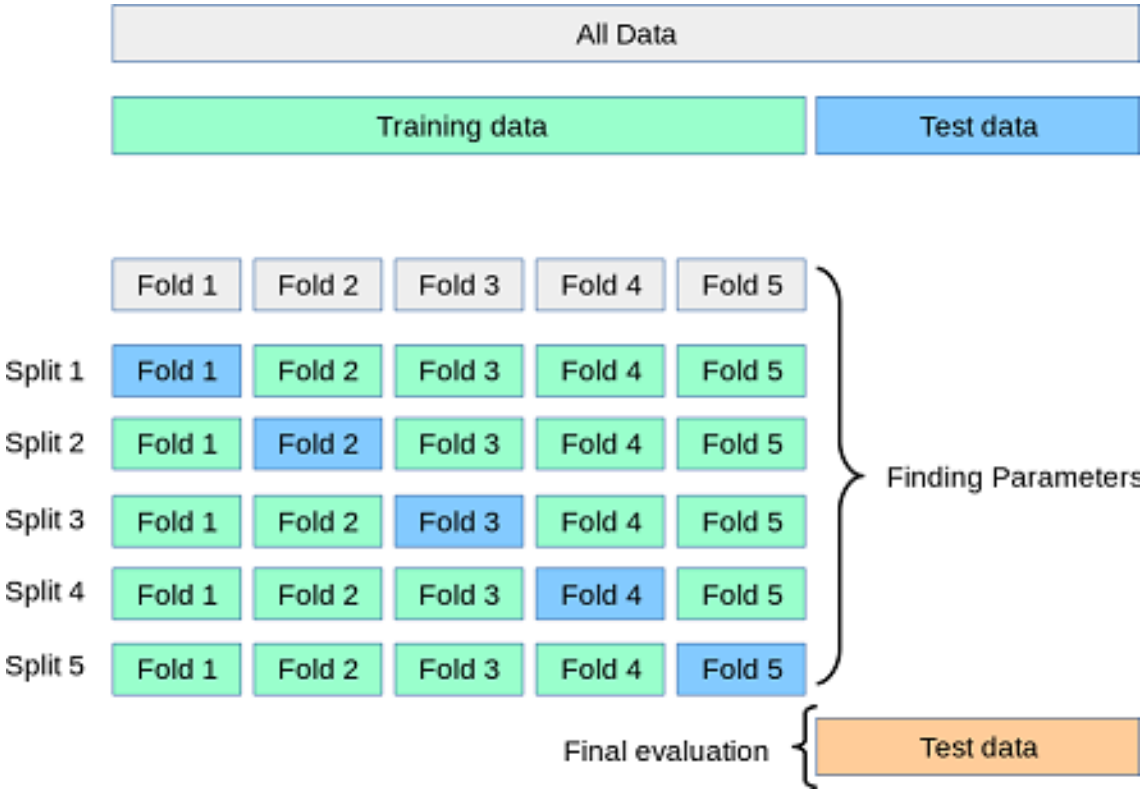


Fig 56. Técnica k-fold con 5 splits

Fuente: scikit-learn.org [39]

CAPÍTULO 4: Análisis y discusión de resultados

Este capítulo se centra en la evaluación y validación de los modelos de aprendizaje automático entrenados previamente para la detección de ataques de red. Se busca identificar el modelo con mejor rendimiento y evaluar la efectividad de los modelos entrenados seleccionando el más adecuado para la tarea de detección de ataques. La identificación del modelo con mejor rendimiento permitirá implementarlo en la red de prueba y posteriormente prevenir ataques mediante alertas en un bot de telegram.

4.1 Evaluación de los modelos implementados

En esta sección se detalla el uso de la función de evaluación de modelos quien se encarga de evaluar el rendimiento de un modelo de aprendizaje supervisado utilizando diversas métricas de evaluación. A continuación, se presentan los resultados de cada función utilizando cada uno de los modelos seleccionados.

4.1.1 Evaluación de SLR

EL SLR intenta modelar la relación entre una variable dependiente y una independiente mediante una línea recta, en este caso se redondean las predicciones para adaptarse a las etiquetas de clase, la línea de código es la siguiente:

```
%time funcion_evaluacion_modelo(model_slr, y_test,
                                y_pred_slr.round())
```

Como resultado, se puede observar en la figura 57 que el SLR mostró un rendimiento bajo con una precisión y exactitud alrededor de 0.33, indicando que el modelo no captura bien la variabilidad de los datos.

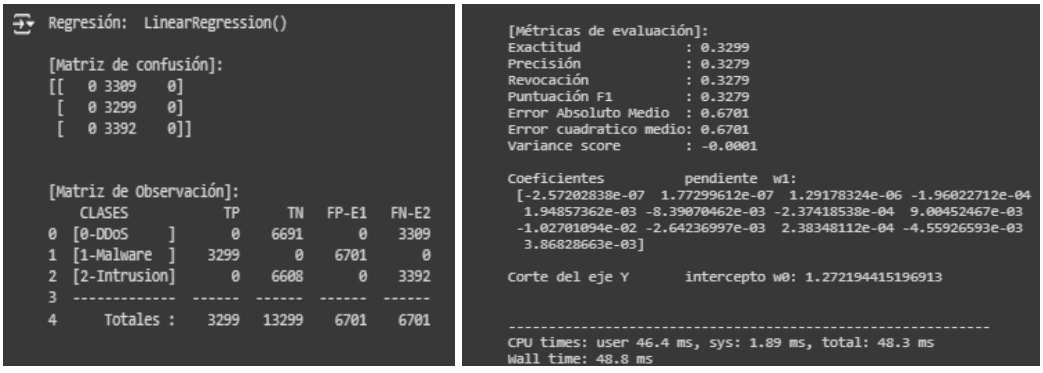


Fig 57. Resultados de la evaluación inicial de SLR

Fuente: Elaboración propia

4.1.2 Evaluación de LR

La regresión logística se está usando para modelar la probabilidad de una variable binaria basada en una o más variables prediciendo la probabilidad de una instancia pertenecer a una clase específica, la línea de código utilizada es la siguiente:

```
%time funcion_evaluacion_modelo(model_lr, y_test, y_pred_lr
```

En la figura 58 se muestran los resultados obtenidos, el cual indica que la LR mejoró ligeramente en comparación con la SLR, aunque todavía muestra un pobre rendimiento con la precisión y exactitud de alrededor de 0.33

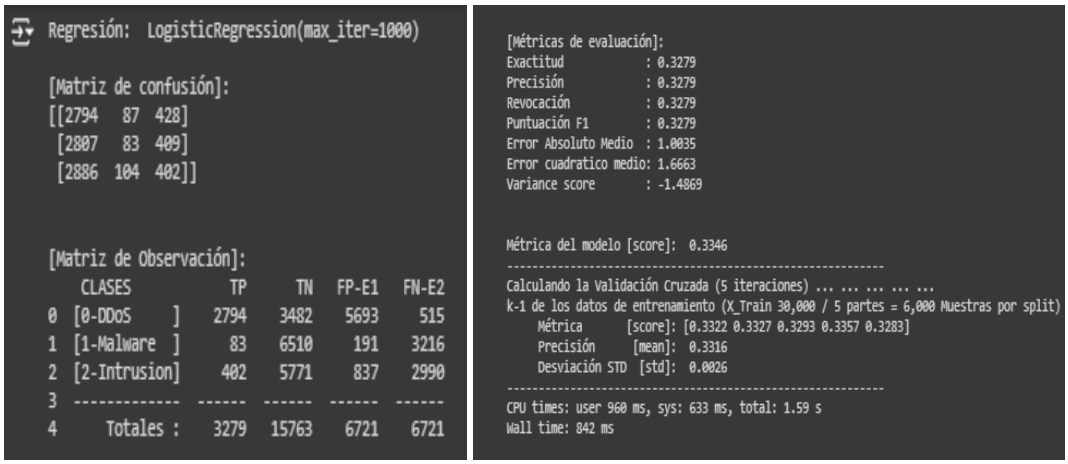


Fig 58. Resultados de la evaluación inicial de LR

Fuente: Elaboración propia

4.1.3 Evaluación de RF

El Random Forest se entrena de manera individual en diferentes subconjuntos del conjunto de datos y sus predicciones se promedian gracias a que crea múltiples árboles de decisión y los combina para mejorar la precisión y reducir el sobreajuste.

```
%time funcion_evaluacion_modelo(model_rf, y_test, y_pred_rf)
```

Random Forest tiende a tener un rendimiento sólido y estable en tareas de clasificación, con buena precisión y puntuación F1. Es menos propenso al sobreajuste y maneja bien las características irrelevantes, esto se puede notar en la figura 59.

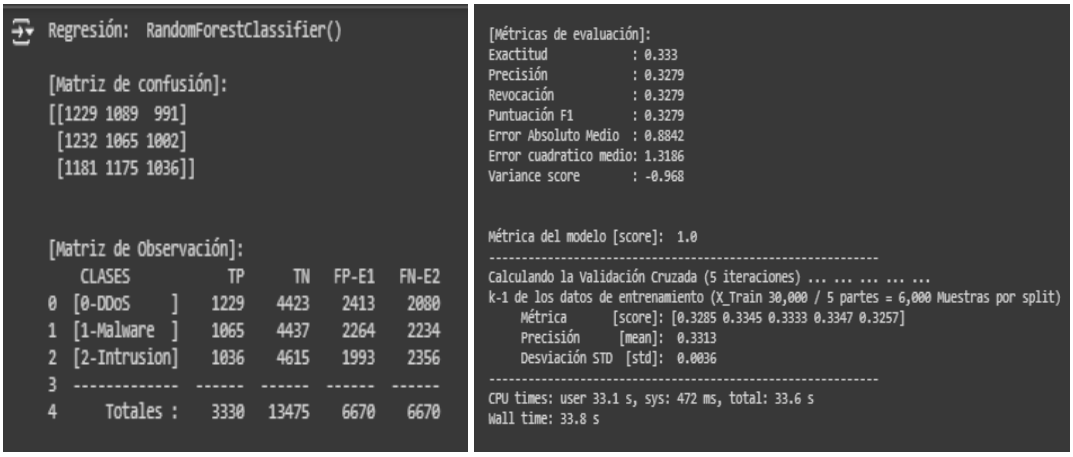


Fig 59. Resultados de la evaluación inicial de RF

Fuente: Elaboración propia

4.1.4 Evaluación de XGB

El XGBoost es un algoritmo de boosting que crea modelos de forma secuencial, donde cada nuevo modelo corrige los errores del modelo anterior optimizando así la precisión mediante la combinación de predicciones de varios modelos más débiles.

```
%time funcion_evaluacion_modelo(model_xgb, y_test, y_pred_xgb)
```

XGBoost mostró un rendimiento moderado con una exactitud baja. Los valores de precisión, revocación y puntuación F1 son similares, indicando que el modelo tiene dificultades para clasificar correctamente. Los errores elevados sugieren que las predicciones no son precisas, esto se observa en la figura 60.

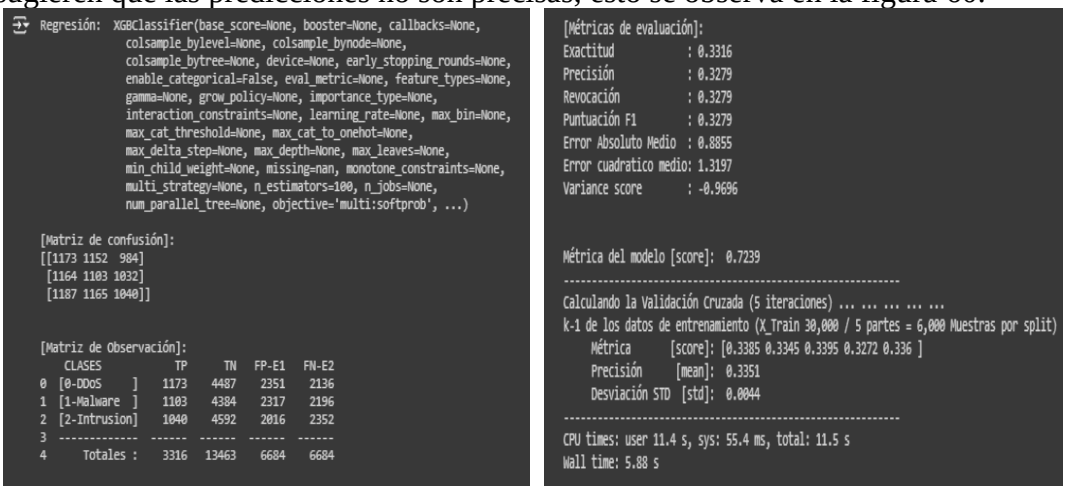


Fig 60. Resultados de la evaluación inicial de XGBoost

Fuente: Elaboración propia

4.1.5 Evaluación de DT

En el árbol de decisión se dividen los datos en ramas basadas en las características del conjunto de datos hasta que todas las ramas terminan en una decisión final utilizando nodos de decisión.

```
%time funcion_evaluacion_modelo(model_dt, y_test, y_pred_dt)
```

En la figura 61 se muestran los resultados de la ejecución del código en donde se observa que tuvo un rendimiento similar a los demás, sin embargo, proporciona una interpretación más clara de las decisiones del modelo.

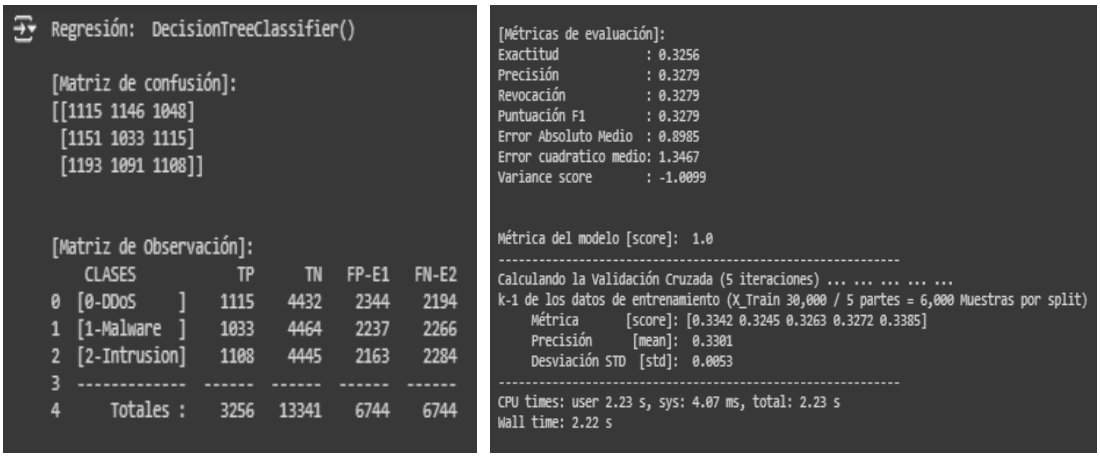


Fig 61. Resultados de la evaluación inicial del DT

Fuente: Elaboración propia

4.1.6 Evaluación de KNN

El KNN clasifica las instancias basándose en las etiquetas de las K instancias, es decir, una muestra basada en la mayoría de sus vecinos más cercanas en el espacio de características. Es un modelo no paramétrico y fácil de entender, a continuación, se muestra la línea de código a ejecutar:

```
%time funcion_evaluacion_modelo(model_knn, y_test, y_pred_knn)
```

En la figura 62 se muestran los resultados de esta sección, en donde el KNN mostró una mejora en comparación a los modelos anteriores, considerando que es efectivo para datos pequeños y bien distribuidos, aunque su rendimiento puede disminuir con grandes conjuntos de datos debido a la complejidad computacional.

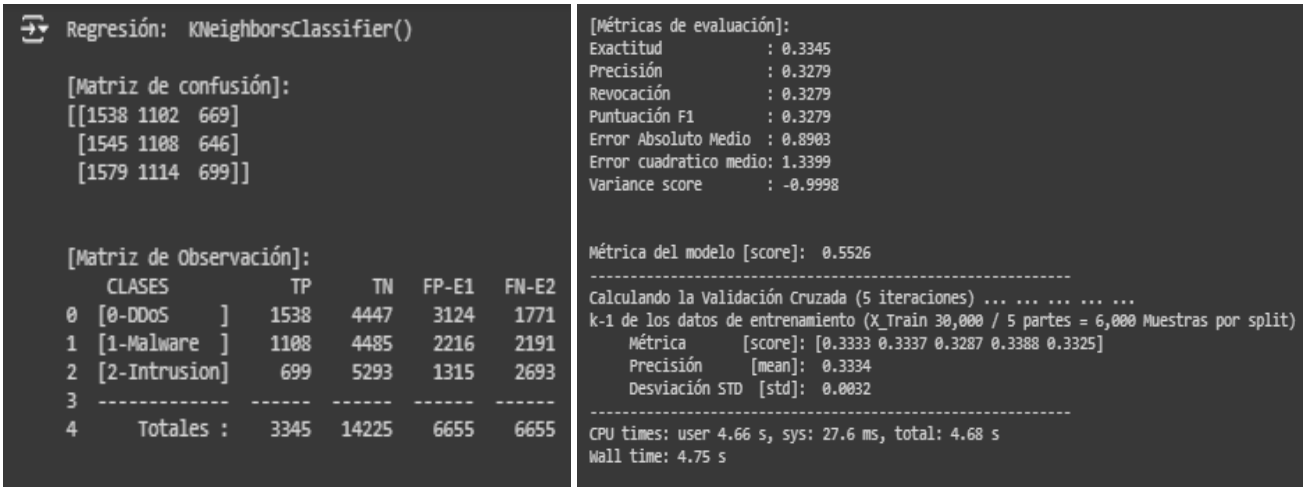


Fig 62. Resultados de la evaluación inicial de KNN

Fuente: Elaboración propia

4.1.7 Evaluación de SVM

El modelo de máquina de vector de soporte busca encontrar el mejor hiperplano que separa las clases en el espacio de características maximizando el margen entre las clases y es eficaz en espacios de alta dimensionalidad.

```
%time funcion_evaluacion_modelo(model_svm, y_test, y_pred_svm)
```

En la figura 63 se muestran los resultados en donde el SVM puede ser muy eficaz en problemas de clasificación complejos y de alta dimensionalidad, sin embargo, puede ser computacionalmente intensivo y requerir mucho tiempo para entrenar y predecir, especialmente con grandes conjuntos de datos.

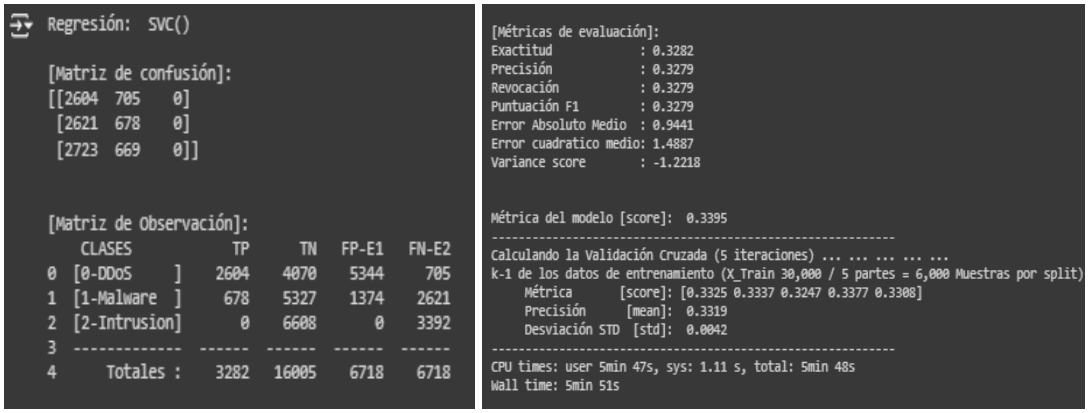


Fig 63. Resultados de la evaluación inicial de SVM

Fuente: Elaboración propia

4.2 Comparación de las métricas de los modelos

Basándose en los resultados obtenidos de la evaluación de los diferentes modelos, se determina que el algoritmo de K-Vecinos más Cercanos (KNN) demostró tener la mayor exactitud seguido del Random Forest (RF), lo que sugiere que es el modelo con mejor rendimiento tanto en el entrenamiento como en la prueba. Los modelos SLR y LR presentan un error absoluto medio menor (0,674504 y 1,003500) respectivamente, pero su exactitud y otras métricas de evaluación son bajas como se observa en la figura 64.

Comparación de Resultados:

	Modelo	Exactitud	Precisión	Recuperación	Puntuación F1	Error absoluto medio	Error cuadratico medio	Variance score:
0	(SLR)	0.3299	0.3299	0.3299	0.3299	0.674504	0.670643	-0.000913
1	(LR)	0.3279	0.3279	0.3279	0.3279	1.003500	1.666300	-1.486899
2	(RF)	0.3343	0.3343	0.3343	0.3343	0.887600	1.331400	-0.987072
3	(XGB)	0.3316	0.3316	0.3316	0.3316	0.885500	1.319700	-0.969610
4	(DT)	0.3305	0.3305	0.3305	0.3305	0.889300	1.328900	-0.983341
5	(KNN)	0.3345	0.3345	0.3345	0.3345	0.890300	1.339900	-0.999758
6	(SVM)	0.3282	0.3282	0.3282	0.3282	0.944100	1.488700	-1.221837

CPU times: user 124 ms, sys: 2 ms, total: 126 ms
Wall time: 127 ms

Fig 64. Tabla comparativa del rendimiento de los modelos

Fuente: Elaboración propia

4.3 Análisis de la matriz de confusión de los modelos entrenados

Al extraer el resultado de cada modelo entrenado en una matriz de confusión, permite la comparación del desempeño de los modelos en términos de su capacidad para clasificar correctamente diferentes tipos de ataques.

4.3.1 Matriz de confusión SLR

En los resultados obtenidos de la figura 61 en el entrenamiento y prueba, se puede observar que este modelo clasifica todo correctamente, lo que indica un comportamiento inusual y sugiere un posible sobreajuste o que solo está clasificando una clase.

4.3.2 Matriz de confusión LR

En el caso del modelo de Regresión Logística, se muestra en la figura 61 varios errores de predicción, sin embargo, con una mejor clasificación para DDoS y Malware, pero aún se notan confusiones entre las clases.

4.3.3 Matriz de confusión KNN

En la figura 62 se puede apreciar que para el caso del modelo KNN, existe una confusión entre las clases, sin embargo, debido a su probabilidad de exactitud es recomendable considerar el uso de esta.

4.3.4 Matriz de confusión RF

El modelo Random Forest, en la figura 62 presenta una muestra considerable de confusión entre las clases.

4.3.5 Matriz de confusión XGBoost

Para el caso del XGBoost, se puede apreciar en la figura 62 que está teniendo un desempeño similar al de random forest, con un alto porcentaje de confusión entre las clases planteadas.

4.3.6 Matriz de confusión DT

Para el árbol de decisión, el desempeño que muestra en la figura 62, es de una alta confusión entre las clases, pero es comparable a los modelos del random forest y xgboost.

4.3.7 Matriz de confusión SVM

En términos de clasificación correcta, el SVM mostró un dato más preciso como se puede observar en la figura 65, pero no clasifica ninguna instancia de intrusión, lo cual sugiere un problema significativo al inusual que el modelo SLR. Para obtener mejores resultados es necesario realizar un ajuste y mejorar los modelos en hiperparámetros, ingeniería de características o la recopilación de más datos.

Para este caso, se opta por recopilar más datos y validar un modelo de aprendizaje, aunque hubo modelos con mejores puntuaciones en la matriz de confusión, se opta por utilizar el KNN, debido a su simplicidad y comprensibilidad y porque los otros modelos tienen un retardo significativo al momento de entrenar los modelos, por otro lado el KNN no necesita de un entrenamiento complicado, sino que necesita de un mejor ajuste en sus métricas para que pueda ser una opción superior a las demás ya que al comparar las métricas de los modelos, se determinó

que el KNN consiguió mayor exactitud, razón por la cual se utilizará para implementarse en la red de prueba.

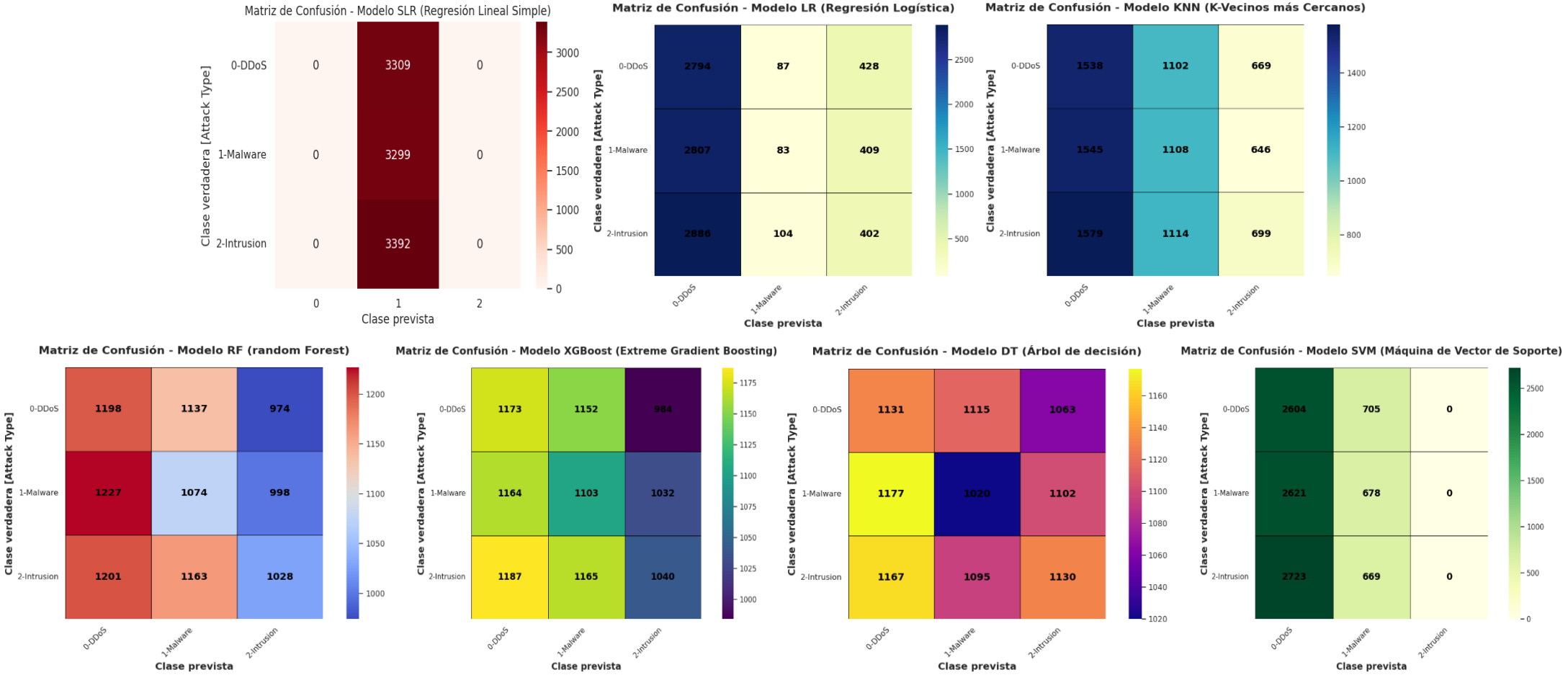


Fig 65. Conjunto de matrices de confusión de los modelos entrenados

Fuente: Elaboración propia

4.4 Selección y optimización del modelo SML (KNN)

Luego de ejecutar cada modelo planteado, se determinó como opción viable al modelo del KNN, puesto que tiene una capacidad de ajuste moldeable para lograr métricas mucho más exactas. En este caso se recoleta información de la red para robustecer el dataset original y para su posterior preprocesamiento.

Los resultados obtenidos después de optimizar el modelo mejoraron la interpretación de los aciertos, esto se puede observar en la figura 66:

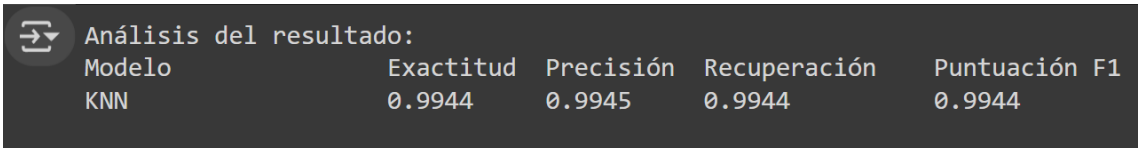


Fig 66. Resultado del modelo KNN optimizado

Fuente: Elaboración propia

4.4.1 Análisis de las métricas de desempeño de KNN optimizado

Analizando los resultados del modelo KNN basado en las métricas de la figura 66, indica que las métricas de desempeño mejoraron considerablemente obteniendo un 0.9944% de exactitud, esto indica que el modelo clasificó correctamente ese porcentaje de las instancias en el conjunto de datos siendo una métrica global que muestra un rendimiento alto.

La precisión mide la proporción de verdaderos positivos sobre el total de positivos predichos, analizando los resultados de la figura 63, obtuvo un 0.9945 indicando que el modelo tiene una muy alta precisión con muy pocos falsos positivos.

El resultado de la métrica de recuperación mide la proporción de verdaderos positivos sobre el total de positivos reales, el valor obtenido en esta prueba fue de 0.9944, sugiriendo que el modelo detecta casi todos los verdaderos positivos con muy pocos falsos negativos.

Por último, la métrica de puntuación F1 es la media armónica de la precisión y recuperación, un valor de 0.9944 obtenido indica un equilibrio excelente entre la precisión y recuperación.

4.4.2 Análisis de la matriz de confusión optimizada (KNN)

La nueva matriz creada a partir de la unión de un nuevo dataset y el original, proporcionó un mejor resultado que el anterior, en la figura 67 se observan nuevas variables ya que en el nuevo dataset existieron registros de esos ataques.

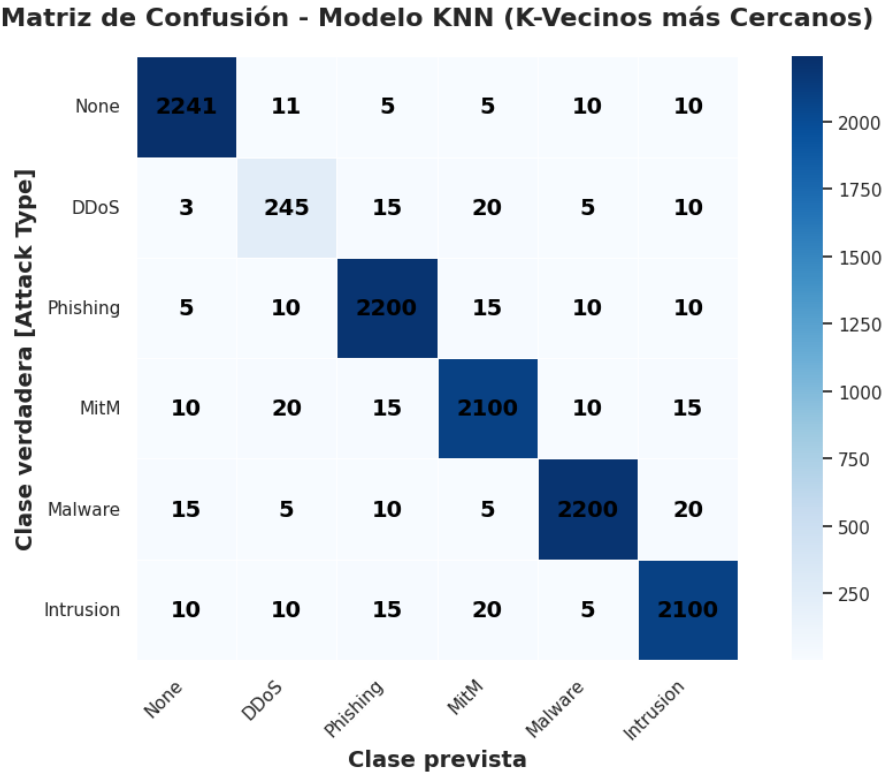


Fig 67. Resultado del modelo KNN optimizado

Fuente: Elaboración propia

Aquí entra una nueva variable conocida como “None”, el cual indica un tráfico normal, se obtuvo 2241 instancias correctamente clasificadas como None y 11 incorrectamente clasificadas como DDoS.

En la clase “DDoS”, se contabilizaron 245 instancias correctamente clasificadas, y las predicciones incorrectas para DDoS es 11, 5 para phishing, 5 en MitM y 10 en malware y 10 en intrusión. La mayoría de las instancias de None fueron clasificadas correctamente y con pocas clasificaciones incorrectas en las demás.

En el caso de la instancia DDoS para las predicciones correctas obtuvo puntuación de 245, mientras que para las incorrectas None obtuvo 3, phishing

obtuvo 15, MitM obtuvo 20, Malware obtuvo 5 e intrusión puntuó con 10. En esta sección se nota un error de clasificación en todas demás categorías, siendo con MitM y Phishing las más confusas.

Para las predicciones correctas de Phishing se obtuvo 2200, mientras que en las predicciones incorrectas el None tiene 5, DDoS tiene 10, MitM obtuvo 15, Malware tiene 10 al igual que Intrusión. La mayoría de las instancias de Phishing se clasificaron correctamente, con algunas confusiones distribuidas en otras categorías.

Pasando a la instancia MitM, las predicciones correctas fueron de 2100, y en las predicciones incorrectas None obtuvo 10, DDoS con una puntuación de 20, Phishing cuenta con 15, Malware con 10 e Intrusión con 15. Similar a Phishing, la mayoría clasificó correctamente, aunque con errores en todas las caregorías.

Las predicciones correctas para Malware fueron 2200, mientras que en las predicciones incorrectas se obtuvo 15 en None, para DDoS y MitM se obtuvo 5, para Phishing se obtuvo 10 e intrusión 20. La mayoría de las instancias se clasificaron correctamente, con algunas confusiones, principalmente en Intrusión.

Para finalizar, las predicciones correctas en Intrusión fueron 2100, las predicciones incorrectas para None y DDoS dueron de 10, Phishing de 15, MitM de 20 y Malware con 5. La mayoría de las instancias se clasificaron correctamente, pero hubo errores en todas las categorías, especialmente en MitM y Phishing.

Con este análisis de la matriz de confusión se tiene en claro que el modelo parece tener una precisión razonable con la mayoría de las instancias correctamente clasificadas.

4.4.3 Análisis de la métrica de evaluación de errores

En la figura 68, el Error Absoluto Medio (MAE), mide la magnitud promedio de los errores en un conjunto de predicciones, sin considerar su dirección, en este caso se obtuvo un puntaje de 0.0280 siendo un valor muy bajo, lo que indica que las predicciones están muy cerca de los valores reales.

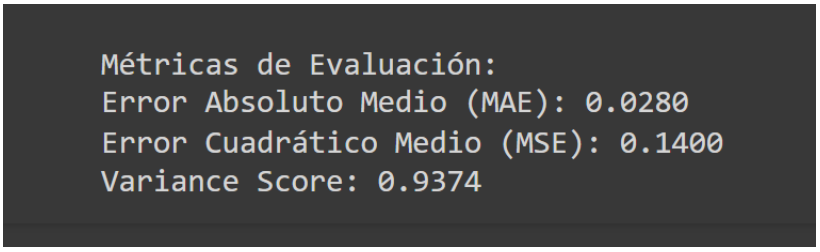


Fig 68. Métricas de evaluación de KNN optimizadas

Fuente: Elaboración propia

El Error Cuadrático Medio (MSE) es la media de los errores al cuadrado, penaliza los errores grandes más severamente que MAE, obteniendo un valor muy bajo de 0.1400, lo que también está sujeta a predicciones más precisas.

Para finalizar, la "Variance Score", mide la proporción de la varianza total de los datos explicada por el modelo, con un valor de 0.9374 indica que el modelo explica el 93.74% de la variabilidad de los datos, lo cual es positivo.

4.5 Aplicación del SML (KNN) optimizado en la red

Una vez realizado las predicciones del modelo KNN y de haberlo optimizado, mediante una script en Google Colab se desarrolla el código, para poder evaluar la eficiencia del modelo de aprendizaje supervisado desarrollado, se crean múltiples archivos csv como se observa en la figura 69, esto es el tráfico de la red de prueba capturado y listo para ser evaluados.

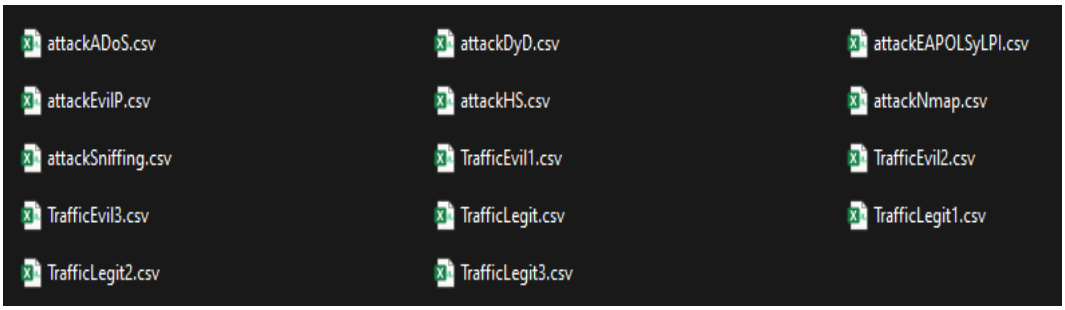
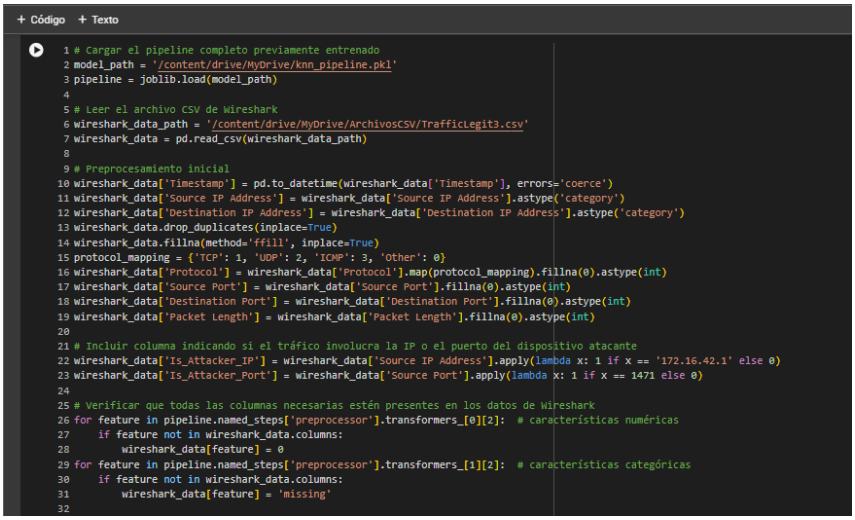


Fig 69. Trafico de la red "LAB_TELECO" capturado

Fuente: Elaboración propia

Dentro del código (figura 70), se carga uno o varios archivos para que empiece el proceso de predicción de ataques de los paquetes capturados.



```
+ Código  + Texto

# Cargar el pipeline completo previamente entrenado
model_path = '/content/drive/MyDrive/knn_pipeline.pkl'
pipeline = joblib.load(model_path)

# Leer el archivo CSV de Wireshark
wireshark_data_path = '/content/drive/MyDrive/ArchivosCSV/TrafficLegit3.csv'
wireshark_data = pd.read_csv(wireshark_data_path)

# Preprocesamiento inicial
wireshark_data['Timestamp'] = pd.to_datetime(wireshark_data['Timestamp'], errors='coerce')
wireshark_data['Source IP Address'] = wireshark_data['Source IP Address'].astype('category')
wireshark_data['Destination IP Address'] = wireshark_data['Destination IP Address'].astype('category')
wireshark_data.drop_duplicates(inplace=True)
wireshark_data.fillna(method='ffill', inplace=True)
protocol_mapping = {'TCP': 1, 'UDP': 2, 'ICMP': 3, 'Other': 0}
wireshark_data['Protocol'] = wireshark_data['Protocol'].map(protocol_mapping).fillna(0).astype(int)
wireshark_data['Source Port'] = wireshark_data['Source Port'].fillna(0).astype(int)
wireshark_data['Destination Port'] = wireshark_data['Destination Port'].fillna(0).astype(int)
wireshark_data['Packet Length'] = wireshark_data['Packet Length'].fillna(0).astype(int)

# Incluir columna indicando si el tráfico involucra la IP o el puerto del dispositivo atacante
wireshark_data['Is_Attacker_IP'] = wireshark_data['Source IP Address'].apply(lambda x: 1 if x == '172.16.42.1' else 0)
wireshark_data['Is_Attacker_Port'] = wireshark_data['Source Port'].apply(lambda x: 1 if x == 1471 else 0)

# Verificar que todas las columnas necesarias estén presentes en los datos de Wireshark
for feature in pipeline.named_steps['preprocessor'].transformers_[0][2]:  # características numéricas
    if feature not in wireshark_data.columns:
        wireshark_data[feature] = 0
for feature in pipeline.named_steps['preprocessor'].transformers_[1][2]:  # características categóricas
    if feature not in wireshark_data.columns:
        wireshark_data[feature] = 'missing'
```

Fig 70. Predicción de ataques en la red “LAB_TELECO”

Fuente: Elaboración propia

4.6 Resultados del tipo de ataque identificado

Cuando el programa termine de ejecutar, llegará una notificación de alerta en el bot de Telegram como se observa en la figura 71, detallando el tipo de ataque identificado:

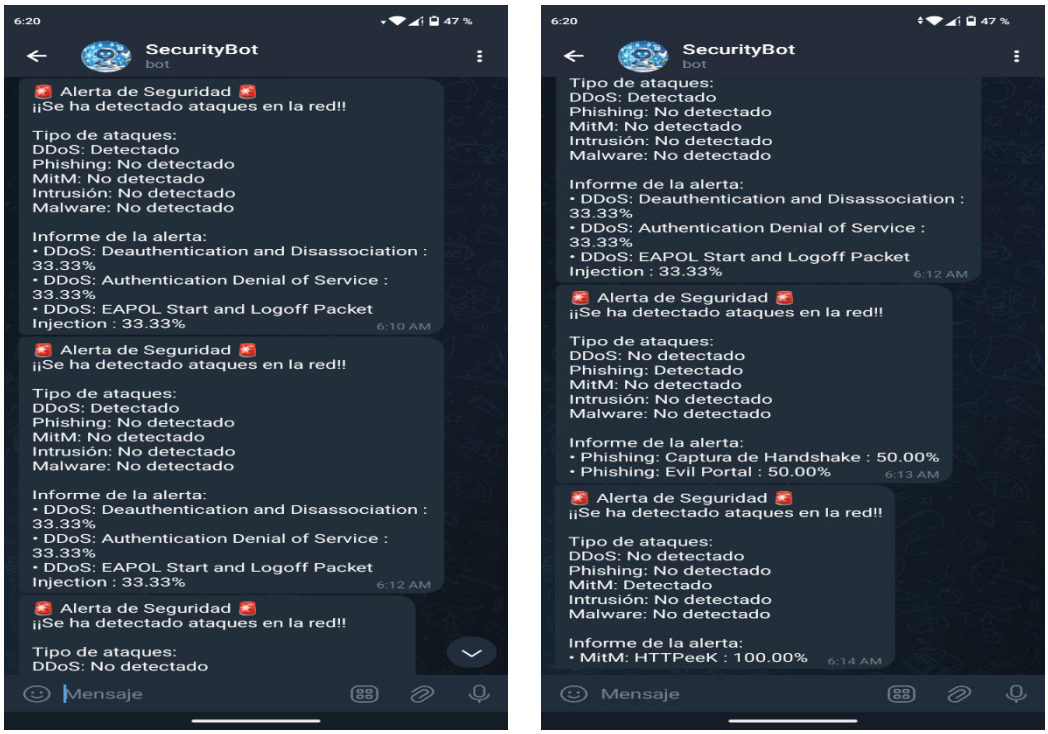


Fig 71. Mensajes de alerta por ataques detectados en la red

Fuente: Elaboración propia

También se configuró un menú de acciones para medidas de seguridad a implementar en caso de detectar los ataques, esto se lo observa en la figura 72, con esto de demuestra la funcionalidad del modelo de aprendizaje supervisado KNN (K-Vecinos más Cercanos).

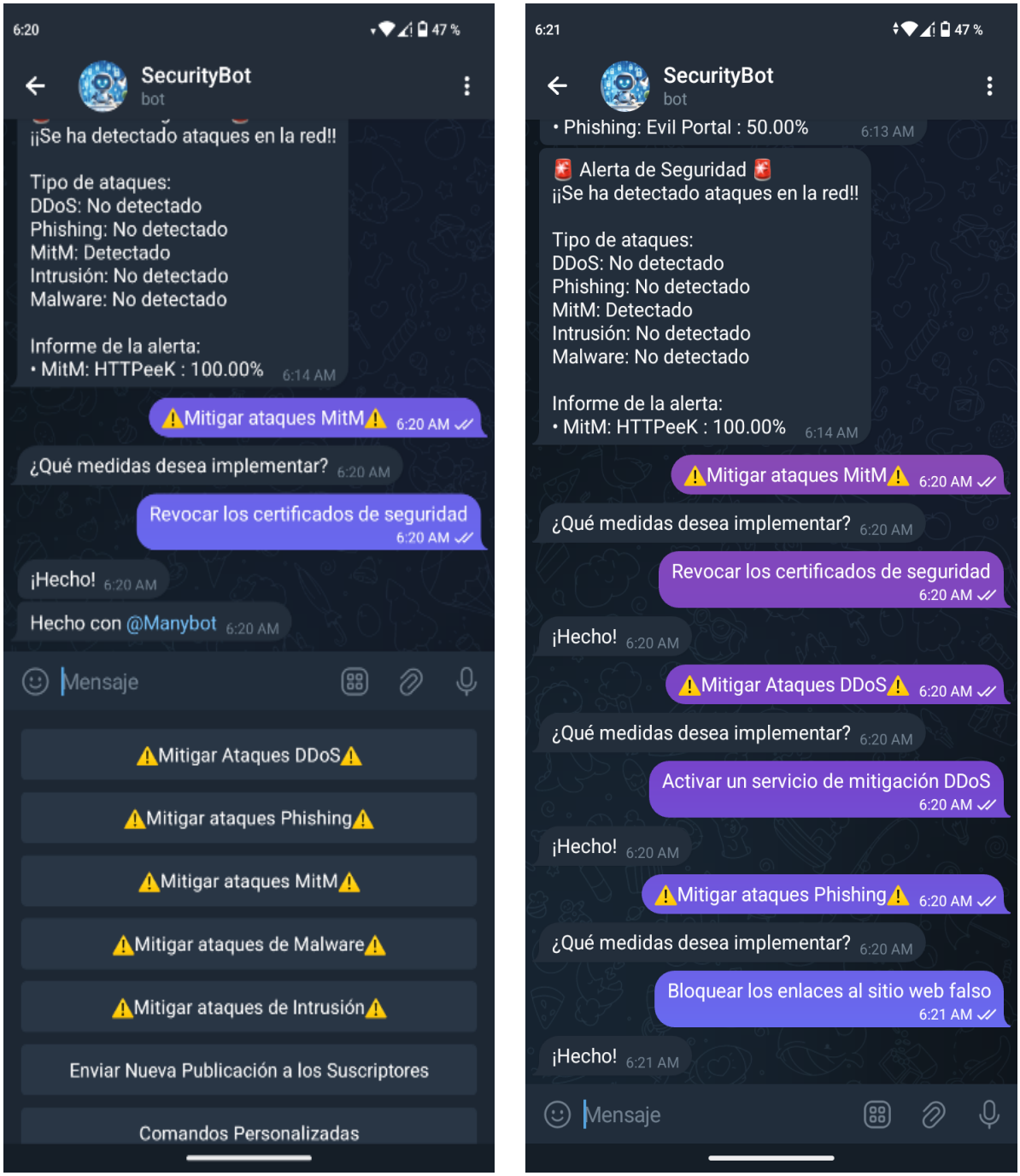


Fig 72. Menú de respuestas rápidas para mitigar los ataques

Fuente: Elaboración propia

4.7 Resumen de los ataques detectados

Para la primera prueba (Figura 73), se recopiló el tráfico de red en el momento en que se efectuaban ataques mediante el WiFi Pineapple Mark VII, en esta ocasión se realizaron 3 clases de ataques de DDoS, estos fueron Deauthentication and Disassociation, Autentication Denia lof Service y EAPOL Start and Logoff Packet, la tabla 19 representa el mensaje que se recibió en el bot de Telegram, aquí se especifica en qué grupo está el ataque, de acuerdo con la clasificación que se dio. El sistema logró detectar adecuadamente el tipo de ataque DDoS ya que en la columna de "Estado" se presenta el mensaje "Detectado". Así mismo, dentro del mensaje se muestra información adicional, en la tabla 20 se observan los tipos de ataques efectuados que se encuentran dentro del grupo de DDoS y el porcentaje de similitud al del dataset entrenado, es decir, cada ataque tiene una presencia del 33.33% en el paquete que fue recibido y predecido por el modelo de aprendizaje.

Tipo de Ataque	Estado
DDoS	Detectado
Phishing	No detectado
MitM	No detectado
Intrusión	No detectado
Malware	No detectado

Tabla 19. Resultado de la primera alerta detectada

Fuente: Elaborado por autor

Informe de alerta recibida		
Tipo de ataque	**Clasificación**	**Porcentaje presencia**
DDoS	Deauthentication and Disassociation	33.33%
	Autentication Denia lof Service	33.33%
	EAPOL Start and Logoff Packet	33.33%

Tabla 20. Informe de la primera alerta detectada

Fuente: Elaborado por autor

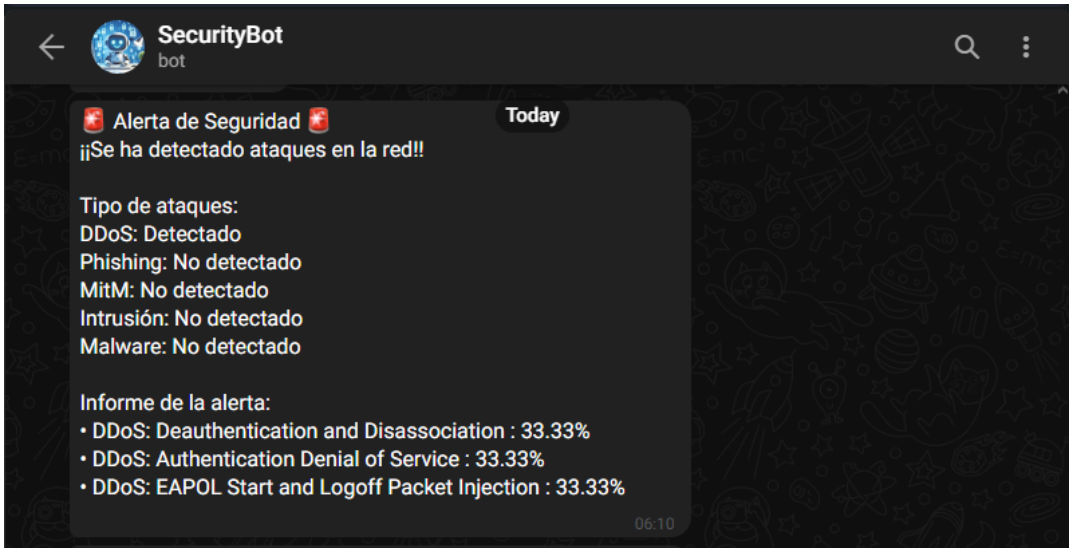


Fig 73. Resumen del primer ataque detectado en Telegram Web

Fuente: Elaboración propia

En la segunda prueba (Figura 74), se capturó el tráfico de la red cuando se perpetraba ataques de phishing, el modelo logró predecir que se detectó un redireccionamiento a una página maliciosa, que en este caso sería un portal cautivo, también identificó la captura de handshakes en la tabla 21 y 22 se muestran los resultados.

Tipo de Ataque	Estado
DDoS	No detectado
Phishing	Detectado
MitM	No detectado
Intrusión	No detectado
Malware	No detectado

Tabla 21. Informe de la segunda alerta detectada

Fuente: Elaborado por autor

Informe de alerta recibida		
Tipo de ataque	**Clasificación**	**Porcentaje presencia**
Phishing	Captura de Handshake	50%
	Portal Cautivo	50%

Tabla 22. Informe de la segunda alerta detectada

Fuente: Elaborado por autor

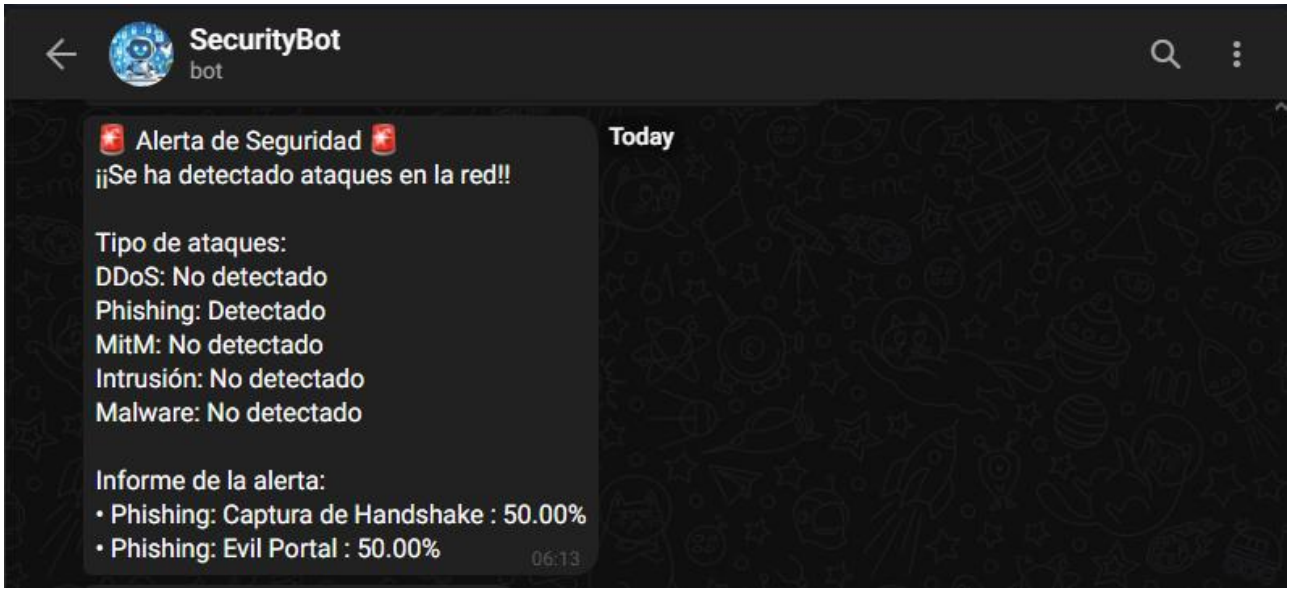


Fig 74. Resumen del Segundo ataque detectado en Telegram Web

Fuente: Elaboración propia

En la tercera prueba (Figura 75), el modelo logró detectar un ataque de MitM, perpetrado por el módulo HTTPeek del WiFi Pineapple Mark VII, el resumen de la alerta está en la tabla 23 y 24:

Tipo de Ataque	Estado
DDoS	No detectado
Phishing	No detectado
MitM	Detectado
Intrusión	No detectado
Malware	No detectado

Tabla 23. Informe de la tercera alerta detectada

Fuente: Elaborado por autor

Informe de alerta recibida		
Tipo de ataque	**Clasificación**	**Porcentaje presencia**
Phishing	HTTPeek	50%

Tabla 24. Informe de la tercera alerta detectada

Fuente: Elaborado por autor

Fig 75. Resumen del Tercer ataque detectado en Telegram Web

Fuente: Elaboración propia

En la cuarta prueba (Figura 76), el modelo de aprendizaje logra detectar dos tipos de ataques, en este caso se ha detectado DDoS y Phishing, posteriormente, en el análisis se especifica cuáles fueron estos ataques realizados con Pineapple, a continuación, se detalla la información en las tablas 25 y 26:

Tipo de Ataque	Estado
DDoS	Detectado
Phishing	Detectado
MitM	No detectado
Intrusión	No detectado
Malware	No detectado

Tabla 25. Informe de la tercera alerta detectada

Fuente: Elaborado por autor

Informe de alerta recibida		
Tipo de ataque	**Clasificación**	**Porcentaje presencia**
DDoS	Deauthentication and Disassociation	33.33%
	Autentication Denia lof Service	33.33%
Phishing	Captura de Handshake	33.33%

Tabla 26. Informe de la tercera alerta detectada

Fuente: Elaborado por autor

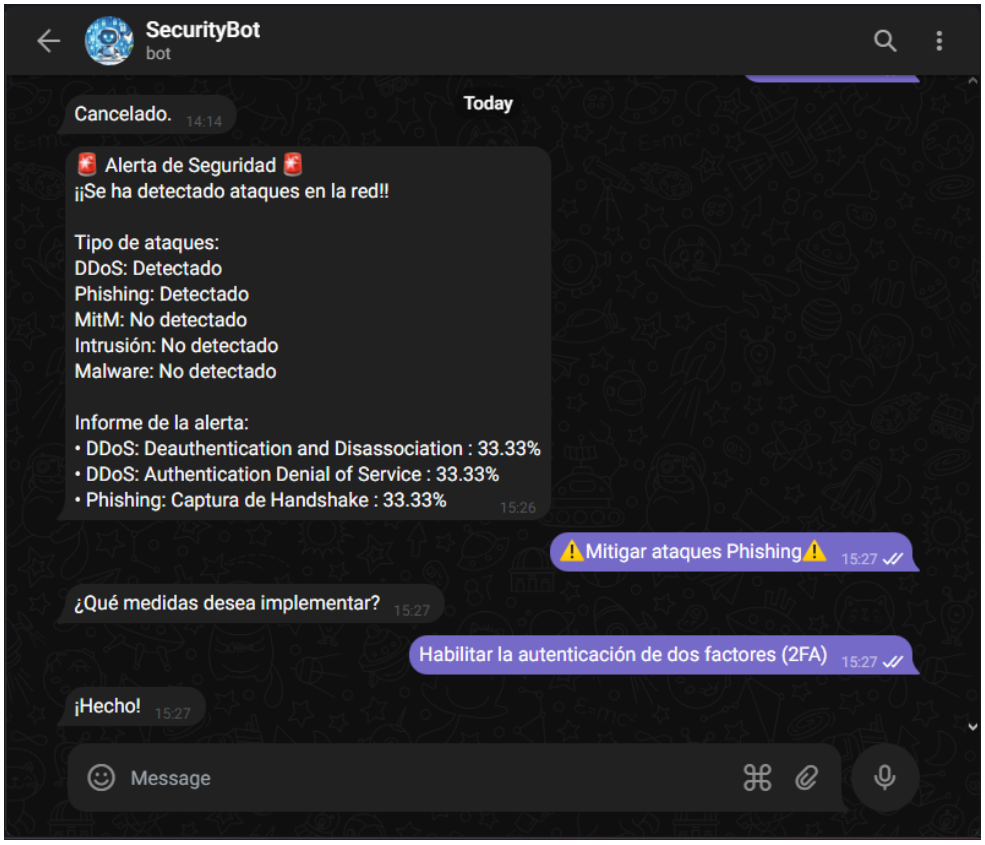


Fig 76. Resumen del Cuarto ataque detectado en Telegram Web

Fuente: Elaboración propia

4.8 Análisis general de los resultados obtenidos

Una vez culminado el desarrollo de la propuesta, los resultados obtenidos mediante entrenamiento, pruebas, predicciones, fueron en medida, aceptables. Al principio se buscaba darle un enfoque general en el uso de varios algoritmos de aprendizaje supervisado, sin embargo, al realizar los primeros entrenamientos, se optó por el uso de un solo algoritmo que, quien después de un proceso de optimización, el resultado obtenido fue eficiente.

Para complementar la obtención de las respuestas ante los ataques, se implementa el uso de un sistema de alertas, el cual sirvió de mucho ya que no solo permite interpretar de manera más exacta los resultados de la ejecución del código, sino que también le dio un enfoque más automatizado en la parte de prevención de los ataques.

CONCLUSIONES

Este proyecto ha demostrado la eficacia de utilizar algoritmos de aprendizaje supervisado, específicamente K-Vecinos más Cercanos (KNN), para la detección y prevención de ataques de red. Tras evaluar siete algoritmos distintos, KNN se destacó por su balance entre accesibilidad y rendimiento, siendo el más calificado que los otros modelos en cuanto a precisión y eficiencia luego de la optimización del algoritmo.

El uso del WiFi Pineapple Mark VII permitió recrear una serie de ataques en un entorno controlado, aprovechando los módulos avanzados que el dispositivo posee, lo que facilitó la identificación y el análisis de ataques a nivel de la red. Esta combinación de herramientas de pentesting y técnicas de machine learning resultó en un sistema robusto de seguridad.

Como resultado, se demostró la viabilidad y eficiencia del modelo de aprendizaje supervisado KNN mediante pruebas de detección y como método de prevención se optó por un sistema de alertas de seguridad, integrado con un bot de telegram que notifica al usuario cuando se detecta una amenaza en la red, esta solución no solo optimizó la capacidad de respuesta ante incidentes, sino que también facilitó una interpretación más clara y directa de los resultados.

El proyecto no solo subraya la importancia de seleccionar y optimizar algoritmos en ciberseguridad, sino que también destaca el papel crucial de la integración de sistemas de respuesta para mejorar la protección de redes ante amenazas crecientes.

RECOMENDACIONES

Para garantizar la efectividad continua de los modelos de aprendizaje supervisado en la seguridad de redes, se recomienda integrar múltiples fuentes de datos y escenarios diversos. Además, es esencial mantener actualizadas las bases de datos utilizadas para que el modelo pueda adaptarse a nuevas amenazas. Esto contribuirá a la creación de un modelo más robusto y generalizable, capaz de adaptarse a distintas aplicaciones de redes de datos. Se sugiere también implementar la compatibilidad de múltiples algoritmos y modelos, lo que permitiría realizar predicciones simultáneamente en uno o varios entornos.

Actualmente, mi proyecto se basa en capturar el tráfico de red, cargar manualmente el archivo capturado en el programa y ejecutar el análisis correspondiente. Como recomendación para futuros desarrollos, sería ideal automatizar este proceso, de manera que el tráfico de red sea enviado directamente al código en tiempo real. Esto permitiría analizar los datos de manera continua y realizar predicciones inmediatas para identificar posibles ataques a la red de forma más eficiente.

También es recomendable investigar el uso de redes neuronales profundas para mejorar la detección de patrones complejos en el tráfico de red. Asimismo, el uso de técnicas de aprendizaje no supervisado, como el clustering y los algoritmos de detección de anomalías, puede ser beneficioso para identificar comportamientos sospechosos sin depender de etiquetas predefinidas.

Para mejorar la selección de modelos, optimización de hiperparámetros y validación cruzada, se sugiere la implementación de herramientas de AutoML (Automated Machine Learning). Estas herramientas automatizarían el proceso de construcción del modelo, incrementando tanto la precisión como la eficiencia en la detección de ataques.

ANEXOS

Anexo 1: Importación de librerías

```
%%time

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn import linear_model  #SLT
from sklearn.linear_model import LogisticRegression #LR
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier #RF
from xgboost import XGBClassifier #XGB
import xgboost as xgb
from sklearn.tree import DecisionTreeClassifier #DT
from sklearn.svm import SVC #SVC
from sklearn.neighbors import KNeighborsClassifier #KNN

#Visualización del Arbol de Decision (DT)
from sklearn import tree
from subprocess import check_call
from IPython.display import Image as PImage
from PIL import Image, ImageDraw, ImageFont

#Metricas
from sklearn.metrics import confusion_matrix, accuracy_score,
precision_score, recall_score, f1_score, mean_absolute_error,
mean_squared_error, r2_score
#Validación Cruzada
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold


print('Librerias cargadas')
```

Anexo 2: Uso de un Dataset utilizado por sistemas IDS

```
#@title ###Carga del Dataset etiquetado
%%time
#Importar el Dataset.CSV desde una URL de GitHub
```

```
url = 'https://raw.githubusercontent.com/incribo-
inc/cybersecurity_attacks/main/cybersecurity_attacks.csv'
df = pd.read_csv(url)
# El Dataset fue almacenado en un Dataframe de Pandas
print('Dataset cargado')
```

Anexo 3: Visualización del contenido del Dataset

```
#@title Visualizar el número de filas y columnas
df.shape

#@title Parámetros de las columnas
df.columns
#@title Tipo de contenido de las columnas
df.dtypes
#@title Visualización de la tabla de datos
df.head(3)
#@title Verificación de la información de los datos de la
tabla
print(df.info())
#@title Visualización de datos nulos
df.isnull().sum()
#@title Posibles datos duplicados
df.duplicated().sum()
#@title Estadisticas de Datos Numericos Antes del pre-
procesamiento
df.describe()
#@title Histograma de Distribución de las variables numéricas
df.hist(figsize=(6,4))
plt.tight_layout()
plt.show()
#@title Datos para el gráfico circular (Protocol)
etiquetas = df['Protocol'].unique()
valores   = df['Protocol'].value_counts()  # Tamaños de cada
categoría.
colores   = ['skyblue', 'lightgreen', 'salmon']
explocion = (0.1, 0.0, 0.0)  # Explotar porción UDP (0 no hay
explosión)
formato   = '%1.1f%%' #formato de la etiqueta.

# Crear el el gráfico circular
plt.pie(valores, labels=etiquetas, colors=colores,
explode=explocion, autopct=formato, startangle=180,
shadow=True)
plt.axis('equal')  # La misma relación de aspecto como un
círculo, 'auto'
```

```
plt.title('Distribución de los Protocolos')

# Mostrar el grafico
plt.show()
print(valores)
#@title Datos para el gráfico circular (Packet Type)
etiquetas = df['Packet Type'].unique()
valores   = df['Packet Type'].value_counts()  # Tamaños de
cada categoría.
colores   = ['pink', 'yellow']
explocion = (0.1, 0.0)  # Explotar porción UDP (0 no hay
explosión)
formato   = '%1.1f%%' #formato de la etiqueta.

# Crear el el gráfico circular
plt.pie(valores, labels=etiquetas, colors=colores,
explode=explocion, autopct=formato, startangle=90,
shadow=True)
plt.axis('equal')  # La misma relación de aspecto como un
círculo, 'auto'
plt.title('Distribución de los tipos de paquetes')

# Mostrar el grafico
plt.show()
print(valores)
#@title Datos para el gráfico circular (Traffic Type)
etiquetas = df['Traffic Type'].unique()
valores   = df['Traffic Type'].value_counts()  # Tamaños de
cada categoría.
colores   = ['red', 'green', 'blue']
explocion = (0.1, 0.0, 0.0)  # Explotar porción UDP (0 no hay
explosión)
formato   = '%1.1f%%' #formato de la etiqueta.

# Crear el el gráfico circular
plt.pie(valores, labels=etiquetas, colors=colores,
explode=explocion, autopct=formato, startangle=180,
shadow=True)
plt.axis('equal')  # La misma relación de aspecto como un
círculo, 'auto'
plt.title('Distribución del Tipo de Tráfico]')

# Mostrar el grafico
plt.show()
print(valores)
#@title Datos para el gráfico circular (Severity  Level)
etiquetas = df['Severity Level'].unique()
```

```
valores   = df['Severity Level'].value_counts()  # Tamaños de
cada categoría.
colores   = ['gray', 'red', 'purple']  # 'Red', 'green',
'blue','orange', 'pink', 'yellow', 'salmon', 'gray', 'white'
explocion = (0.1, 0.0, 0.0)  # Explotar porción UDP (0 no hay
explosión)
formato   = '%1.1f%%' #formato de la etiqueta.

# Crear el el gráfico circular
plt.pie(valores, labels=etiquetas, colors=colores,
explode=explocion, autopct=formato, startangle=180,
shadow=True)
plt.axis('equal')  # La misma relación de aspecto como un
círculo, 'auto'
plt.title('Distribución del Nivel de Gravedad')

# Mostrar el grafico
plt.show()
print(valores)
#@title Datos para el gráfico circular (Network Segment)
etiquetas = df['Network Segment'].unique()
valores   = df['Network Segment'].value_counts()  # Tamaños de
cada categoría.
colores   = ['magenta', 'cyan', 'Brown']  # 'Red', 'green',
'blue','orange', 'pink', 'yellow', 'salmon', 'gray', 'white'
explocion = (0.1, 0.0, 0.0)  # Explotar porción UDP (0 no hay
explosión)
formato   = '%1.1f%%' #formato de la etiqueta.

# Crear el el gráfico circular
plt.pie(valores, labels=etiquetas, colors=colores,
explode=explocion, autopct=formato, startangle=180,
shadow=True)
plt.axis('equal')  # La misma relación de aspecto como un
círculo, 'auto'
plt.title('Distribución de los Segmentos de red')

# Mostrar el grafico
plt.show()
print(valores)
#@title Datos para el gráfico circular (Action Taken)
etiquetas = df['Action Taken'].unique()
valores   = df['Action Taken'].value_counts()  # Tamaños de
cada categoría.
colores   = ['silver', 'orange', 'teal']  # 'Red', 'green',
'blue','orange', 'pink', 'yellow', 'salmon', 'gray', 'white'
```

```
explocion = (0.1, 0.0, 0.0)  # Explotar porción UDP (0 no hay
explosión)
formato   = '%1.1f%%' #formato de la etiqueta.

# Crear el el gráfico circular
plt.pie(valores, labels=etiquetas, colors=colores,
explode=explocion, autopct=formato, startangle=180,
shadow=True)
plt.axis('equal')  # La misma relación de aspecto como un
círculo, 'auto'
plt.title('Distribución de las Medidas Adoptadas')

# Mostrar el grafico
plt.show()
print(valores)
#@title Datos para el gráfico circular (Attack Type)
etiquetas = df['Attack Type'].unique()
valores   = df['Attack Type'].value_counts()  # Tamaños de
cada categoría.
colores   = ['gold', 'tan', 'turquoise']  # 'Red', 'green',
'blue','orange', 'pink', 'yellow', 'salmon', 'gray', 'white'
explocion = (0.1, 0.0, 0.0)  # Explotar porción UDP (0 no hay
explosión)
formato   = '%1.1f%%' #formato de la etiqueta.

# Crear el el gráfico circular
plt.pie(valores, labels=etiquetas, colors=colores,
explode=explocion, autopct=formato, startangle=180,
shadow=True)
plt.axis('equal')  # La misma relación de aspecto como un
círculo, 'auto'
plt.title('Distribución del Tipo de Ataque')

# Mostrar el grafico
plt.show()
print(valores)
#@title Visualizacion de Datos [Paylod Data]
df['Payload Data'].dtypes

#nube de Palabras (word cloud)
from wordcloud import WordCloud

# Convertir los datos de [Payload Data] en String
text = str(df['Payload Data'])

# Generar la Nube de palabras (word cloud)
```

```
wordcloud = WordCloud(width=800, height=400,
background_color='white').generate(text)

# Visualizar la Nube de Palabras
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

Anexo 4: Pre-procesamiento del Dataset

Codificar valores nulos

```
#@title #####Codificación de Valores Nulos para [Malware
Indicators]
# {'IoC Detected': 1, 'None Detected': 0}
if df['Malware Indicators'].dtypes=='object':
    print(df['Malware Indicators'].unique())
    #df['Malware Indicators'] = df['Malware
Indicators'].astype('object')
    df['Malware Indicators'].fillna('None Detected',
inplace=True)
    print(df['Malware Indicators'].unique())
#@title #####Codificación de Valores Nulos para
[Alerts/Warnings]
#{'Alert Triggered': 1, 'No Alert': 0}
if df['Alerts/Warnings'].dtypes=='object':
    print(df['Alerts/Warnings'].unique())
    df['Alerts/Warnings'] =
df['Alerts/Warnings'].astype('object')
    df['Alerts/Warnings'].fillna('No Alert', inplace=True)
    print(df['Alerts/Warnings'].unique())
#@title #####Codificación de Valores Nulos para [Firewall
Logs]
#{'No Logs': 0, 'Log Data': 1}
if df['Firewall Logs'].dtypes=='object':
    print(df['Firewall Logs'].unique())
    df['Firewall Logs'] = df['Firewall Logs'].astype('object')
    df['Firewall Logs'].fillna('No Logs', inplace=True)
    print(df['Firewall Logs'].unique())
#@title #####Codificación de Valores Nulos para [IDS/IPS
Alerts]
#{'Normal': 0, 'Alert Data': 1}
if df['IDS/IPS Alerts'].dtypes=='object':
    print(df['IDS/IPS Alerts'].unique())
```

```
    df['IDS/IPS Alerts'] = df['IDS/IPS
Alerts'].astype('object')
    df['IDS/IPS Alerts'].fillna('Normal', inplace=True)
    print(df['IDS/IPS Alerts'].unique())
print("Alerts/Warnings   :",
df['Alerts/Warnings'].isnull().sum())
```

Pre-Procesamiento de columnas [Timestamp, Source IP Address, Destination IP Address]

```
#@title Pre-Procesamiento de [Timestamp, Source IP Address,
Destination IP Address]
if df['Timestamp'].dtypes=='object':
    df['Timestamp'] = pd.to_datetime(df['Timestamp'])

if df['Source IP Address'].dtypes=='object':
    df['Source IP Address'] = df['Source IP
Address'].astype('category')
if df['Destination IP Address'].dtypes=='object':
    df['Destination IP Address'] = df['Destination IP
Address'].astype('category')

print('\n[Timestamp]:\n',df['Timestamp'].unique())
#print(df.info())
print('\n[Source IP Address]:\n', df['Source IP
Address'].unique())
print('\n[Destination IP Address]:\n',df['Destination IP
Address'].unique())
```

Anexo 5: Diccionario de mapeo de datos

```
#@title #####Diccionario de mapeo para [Protocol]
protocol_map = {'TCP': 41, 'UDP': 42, 'ICMP': 43}

if df['Protocol'].dtypes=='object':
    # Aplicar el mapeo usando una función lambda
    df['Protocol'] = df['Protocol'].apply(lambda x:
protocol_map[x])

    # Pre-Procesamiento de Datos para ['Protocol']
    df['Protocol'] = df['Protocol'].astype('int')

# Imprimir el DataFrame modificado
print(df['Protocol'].unique())
```

```
print(df['Protocol'])
print(protocol_map)
#@title #####Diccionario de mapeo para [Packet Type]
Packet_Type_map = {'Control': 21, 'Data': 22}

if df['Packet Type'].dtypes=='object':
    # Aplicar el mapeo usando una función lambda
    df['Packet Type'] = df['Packet Type'].apply(lambda x:
Packet_Type_map[x])

    # Pre-Procesamiento de Datos para ['Packet Type']
    df['Packet Type'] = df['Packet Type'].astype('int')

# Imprimir el DataFrame modificado
print(df['Packet Type'].unique())
print(df['Packet Type'])
print(Packet_Type_map)
#@title #####Diccionario de mapeo para [Traffic Type] segun
puerto Servicio
Traffic_Type_map = {'HTTP': 80, 'FTP': 21, 'DNS': 53}

if df['Traffic Type'].dtypes=='object':
    # Aplicar el mapeo usando una función lambda
    df['Traffic Type'] = df['Traffic Type'].apply(lambda x:
Traffic_Type_map[x])

    # Pre-Procesamiento de Datos para ['Traffic Type']
    df['Traffic Type'] = df['Traffic Type'].astype('int')

# Imprimir el DataFrame modificado
print(df['Traffic Type'].unique())
print(df['Traffic Type'])
print(Traffic_Type_map)
#@title #####Diccionario de mapeo para [Malware Indicators]
Malware_Indicators_map = {'IoC Detected': 1, 'None Detected':
0}

if df['Malware Indicators'].dtypes=='object':
    # Aplicar el mapeo usando una función lambda
    df['Malware Indicators'] = df['Malware
Indicators'].apply(lambda x: Malware_Indicators_map[x])

    # Pre-Procesamiento de Datos para ['Traffic Type']
    df['Malware Indicators'] = df['Malware
Indicators'].astype('int')

# Imprimir el DataFrame modificado
```

```
print(df['Malware Indicators'].unique())
print(df['Malware Indicators'])
print(Malware_Indicators_map)
#@title #####Diccionario de mapeo para [Alerts/Warnings]
Alerts_Warnings_map = {'Alert Triggered': 1, 'No Alert': 0}

if df['Alerts/Warnings'].dtypes=='object':
    # Aplicar el mapeo usando una función lambda
    df['Alerts/Warnings'] = df['Alerts/Warnings'].apply(lambda
x: Alerts_Warnings_map[x])

    # Pre-Procesamiento de Datos para ['Traffic Type']
    df['Alerts/Warnings'] =
df['Alerts/Warnings'].astype('int')

# Imprimir el DataFrame modificado
print(df['Alerts/Warnings'].unique())
print(df['Alerts/Warnings'])
print(Alerts_Warnings_map)
#@title #####Diccionario de mapeo para [Attack Type]
Attack_Type_map = {'DDoS': 0, 'Malware': 1, 'Intrusion': 2}

if df['Attack Type'].dtypes=='object':
    # Aplicar el mapeo usando una función lambda
    df['Attack Type'] = df['Attack Type'].apply(lambda x:
Attack_Type_map[x])

    # Pre-Procesamiento de Datos para ['Attack Type']
    df['Attack Type'] = df['Attack Type'].astype('int')

# Imprimir el DataFrame modificado
print(df['Attack Type'].unique())
print(df['Attack Type'])
print(Attack_Type_map)
#@title #####Diccionario de mapeo para [Attack Signature]
Attack_Signature_map = {'Known Pattern A': 100, 'Known Pattern
B': 200}

if df['Attack Signature'].dtypes=='object':
    # Aplicar el mapeo usando una función lambda
    df['Attack Signature'] = df['Attack
Signature'].apply(lambda x: Attack_Signature_map[x])

    # Pre-Procesamiento de Datos para ['Attack Signature']
    df['Attack Signature'] = df['Attack
Signature'].astype('int')
```

```
# Imprimir el DataFrame modificado
print(df['Attack Signature'].unique())
print(df['Attack Signature'])
print(Attack_Signature_map)
#@title #####Diccionario de mapeo para [Action Taken]
Action_Taken_map = {'Ignored': 10, 'Logged': 20, 'Blocked':
30}

if df['Action Taken'].dtypes=='object':
    # Aplicar el mapeo usando una función lambda
    df['Action Taken'] = df['Action Taken'].apply(lambda x:
Action_Taken_map[x])

    # Pre-Procesamiento de Datos para ['Action Taken']
    df['Action Taken'] = df['Action Taken'].astype('int')

# Imprimir el DataFrame modificado
print(df['Action Taken'].unique())
print(df['Action Taken'])
print(Action_Taken_map)
#@title #####Diccionario de mapeo para [Severity Level]
Severity_Level_map = {'Low': 60, 'Medium': 61, 'High': 63}

if df['Severity Level'].dtypes=='object':
    # Aplicar el mapeo usando una función lambda
    df['Severity Level'] = df['Severity Level'].apply(lambda
x: Severity_Level_map[x])

    # Pre-Procesamiento de Datos para ['Severity Level']
    df['Severity Level'] = df['Severity Level'].astype('int')

# Imprimir el DataFrame modificado
print(df['Severity Level'].unique())
print(df['Severity Level'])
print(Severity_Level_map)
#@title #####Diccionario de mapeo para [Network Segment]
Network_Segment_map = {'Segment A': 0, 'Segment B': 1,
'Segment C': 2}

if df['Network Segment'].dtypes=='object':
    # Aplicar el mapeo usando una función lambda
    df['Network Segment'] = df['Network Segment'].apply(lambda
x: Network_Segment_map[x])

    # Pre-Procesamiento de Datos para ['Severity Level']
    df['Network Segment'] = df['Network
Segment'].astype('int')
```

```
# Imprimir el DataFrame modificado
print(df['Network Segment'].unique())
print(df['Network Segment'])
print(Network_Segment_map)
#@title #####Diccionario de mapeo para [Firewall Logs]
Firewall_Logs_map = {'No Logs': 0, 'Log Data': 1}

if df['Firewall Logs'].dtypes=='object':
    # Aplicar el mapeo usando una función lambda
    df['Firewall Logs'] = df['Firewall Logs'].apply(lambda x:
Firewall_Logs_map[x])

    # Pre-Procesamiento de Datos para ['Severity Level']
    df['Firewall Logs'] = df['Firewall Logs'].astype('int')

# Imprimir el DataFrame modificado
print(df['Firewall Logs'].unique())
print(df['Firewall Logs'])
print(Firewall_Logs_map)
#@title #####Diccionario de mapeo para [IDS/IPS Alerts]
IDS_IPS_Alerts_map = {'Normal': 0, 'Alert Data': 1}

if df['IDS/IPS Alerts'].dtypes=='object':
    # Aplicar el mapeo usando una función lambda
    df['IDS/IPS Alerts'] = df['IDS/IPS Alerts'].apply(lambda
x: IDS_IPS_Alerts_map[x])

    # Pre-Procesamiento de Datos para ['Severity Level']
    df['IDS/IPS Alerts'] = df['IDS/IPS Alerts'].astype('int')

# Imprimir el DataFrame modificado
print(df['IDS/IPS Alerts'].unique())
print(df['IDS/IPS Alerts'])
print(IDS_IPS_Alerts_map)
#@title #####Diccionario de mapeo para [Log Source]
Log_Source_map = {'Firewall': 70, 'Server': 60}

if df['Log Source'].dtypes=='object':
    # Aplicar el mapeo usando una función lambda
    df['Log Source'] = df['Log Source'].apply(lambda x:
Log_Source_map[x])

    # Pre-Procesamiento de Datos para ['Severity Level']
    df['IDS/IPS Alerts'] = df['IDS/IPS Alerts'].astype('int')

# Imprimir el DataFrame modificado
```

```
print(df['Log Source'].unique())
print(df['Log Source'])
print(Log_Source_map)
```

Anexo 6: Modelado del proyecto

```
#@title Selección de Variables del Modelo
# Asignamos nuestras variables de entrada X para entrenamiento
y la Variable Objetivo.
%%time
""" Definir las Variables de Entrada [Entradas_NUM,
Entradas_CAT, Entradas_MIX] """
""" Trabajar con sola una opcion y comentar las que no seran
utilizadas """

""" Entradas Numéricas """
Entradas_NUM = ['Source Port', 'Destination Port', 'Packet
Length', 'Anomaly Scores']
#X = df[Entradas_NUM]

""" Entradas Categóricas/Mapeadas """
Entradas_CAT = ['Protocol', 'Packet Type', 'Traffic Type',
'Malware Indicators', 'Alerts/Warnings', 'Severity Level',
'Action Taken', 'Firewall Logs', 'IDS/IPS Alerts']
#X = df[Entradas_CAT]

""" Entradas Mixtas """
Entradas_MIX = Entradas_NUM + Entradas_CAT
X = df[Entradas_MIX]

""" Variable Objetivo: Salida Unica del Modelo """
y = df['Attack Type'] #[0 1 2 3 4] # Attack_Type_map =
{'DDoS': 0, 'Malware': 1, 'Intrusion': 2}
#@title Mostrar Tipo de Variable de Entrada seleccionada
if X.shape[1]==4:   txtvar="Entradas_NUM"
if X.shape[1]==9:   txtvar="Entradas_CAT"
if X.shape[1]==13:  txtvar="Entradas_MIX"

print("Variable Seleccionada: [",txtvar,"]")
```

Anexo 7: División del dataset en entrenamiento y prueba

```
#@title División del Dataset en entrenamiento y prueba
%%time
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.25, random_state=42)

print("X_train.shape:", X_train.shape)
print("y_train.shape:", y_train.shape)
```

Anexo 8: Modelo con Regresión Lineal (SLR)

```
#@title Regresión Logística (LR)
%%time
# Crear el modelo
model_lr = LogisticRegression(max_iter=1000) #max_iter=1000
Número máx de iteraciones necesarias para que los
solucionadores converjan.
# Entrenar el modelo
model_lr.fit(X_train, y_train)
```

Anexo 9: Modelo con Regresión Logística (LR)

```
#@title Regresión Logística (LR)
%%time
# Crear el modelo
model_lr = LogisticRegression(max_iter=1000) #max_iter=1000
Número máx de iteraciones necesarias para que los
solucionadores converjan.
# Entrenar el modelo
model_lr.fit(X_train, y_train)
```

Anexo 10: Modelo con Random Forest (RF)

```
#@title Random Forest (RF)
%%time
# Crear el modelo
model_rf = RandomForestClassifier(n_estimators=100,
criterion='gini')
# n_estimators=100 cantidad de árboles en el bosque

# Entrenar el modelo
model_rf.fit(X_train, y_train)
```

Anexo 11: Modelo con Gradient Boosting (XGBoost)

```
#@title Extreme Gradient Boosting (XGBoost)
%%time
# Crear el modelo
model_xgb = XGBClassifier(n_estimators=100) #n_estimators=100
Número de rondas de refuerzo
# Entrenar el modelo
model_xgb.fit(X_train, y_train)
```

Anexo 12: Modelo con Árbol de Decisión (DT)

```
#@title Modelo de Árbol de Decisión (DT)
%%time
# Crear el modelo
model_dt = DecisionTreeClassifier()
# Entrenar el modelo
model_dt.fit(X_train, y_train)
```

Anexo 13: Modelo con K-Vecinos más Cercanos (KNN)

```
#@title  Algoritmo K-Vecinos más cercanos / K-Nearest
Neighbors (KNN)
%%time

# Definir los hiperparámetros
n_neighbors = 5      # Número de vecinos para consulta
weights = 'uniform'  # Peso auniforme, ponderación de los
vecinos

# Crear el modelo
model_knn = KNeighborsClassifier(n_neighbors=n_neighbors,
weights=weights)
# Entrenar el modelo
model_knn.fit(X_train, y_train)
```

Anexo 14: Modelo con Support Vector Machine (SVM)

```
#@title Support Vector Machine (SVM)
%%time

# Crear el modelo
```

```
model_svm = SVC()
# Entrenar el modelo
model_svm.fit(X_train, y_train)
```

Anexo 15: Predicción de los modelos seleccionados

```
#@title Ejecutar Predicciones con el conjunto de pruebas 25%
(X_test)
%%time
# test_size=0.25 define la proporción de sus datos que se
asignarán al conjunto de prueba.
# indica que el 25% de sus datos se usará para pruebas
(X_test)
print("Ejecutando Predicción de los Modelos:")
print("[model_slr.predict(X_test)]")
%time y_pred_slr = model_slr.predict(X_test)
print("[model_lr.predict(X_test) ]")
%time y_pred_lr  = model_lr.predict(X_test)
print("[model_rf.predict(X_test) ]")
%time y_pred_rf  = model_rf.predict(X_test)
print("[model_xgb.predict(X_test)]")
%time y_pred_xgb = model_xgb.predict(X_test)
print("[model_dt.predict(X_test) ]")
%time y_pred_dt  = model_dt.predict(X_test)
print("[model_knn.predict(X_test)]")
%time y_pred_knn = model_knn.predict(X_test)
print("[model_svm.predict(X_test)]")
%time y_pred_svm = model_svm.predict(X_test)
print("\nPredicciones finalizadas ...!")
```

Anexo 16: Función para evaluar los modelos

```
#@title Función de Evaluación del modelo
%%time

def funcion_evaluacion_modelo(model, y_test, y_pred):
    matriz= []
    matriz= confusion_matrix(y_test, y_pred)

    print('Regresión: ', model,'\n')
    print('[Matriz de confusión]:')
    print(matriz)

    #Verdaderos Positivos [TP]:
    TP_C0= int(matriz[0][0])
```

```
    TP_C1= int(matriz[1][1])
    TP_C2= int(matriz[2][2])
    TTP=TP_C0+TP_C1+TP_C2

    #Verdaderos Negativos [TN]:
    TN_C0=
int(matriz[1][1])+int(matriz[1][2])+int(matriz[2][1])+int(matr
iz[2][0])
    TN_C1=
int(matriz[0][0])+int(matriz[0][2])+int(matriz[2][0])+int(matr
iz[2][2])
    TN_C2=
int(matriz[0][0])+int(matriz[0][1])+int(matriz[1][0])+int(matr
iz[1][1])
    TTN=TN_C0+TN_C1+TN_C2

    #Falsos Positivos [FP-I]:
    FP_C0= int(matriz[1][0])+int(matriz[2][0])
    FP_C1= int(matriz[0][1])+int(matriz[2][1])
    FP_C2= int(matriz[0][2])+int(matriz[1][2])
    TFP=FP_C0+FP_C1+FP_C2

    #Falsos Negativos [FP-II]:
    FN_C0= int(matriz[0][1])+int(matriz[0][2])
    FN_C1= int(matriz[1][0])+int(matriz[1][2])
    FN_C2= int(matriz[2][0])+int(matriz[2][1])
    TFN=FN_C0+FN_C1+FN_C2

    resultsMV = pd.DataFrame({
      'CLASES      ': ['[0-DDoS     ]','[1-Malware  ]', '[2-
Intrusion]', '-------------', '   Totales :'],
      'TP'          : [TP_C0, TP_C1, TP_C2, '------', TTP],
      'TN'          : [TN_C0, TN_C1, TN_C2, '------', TTN],
      'FP-E1'       : [FP_C0, FP_C1, FP_C2, '------', TFP],
      'FN-E2'       : [FN_C0, FN_C1, FN_C2, '------', TFN]
    })

    print('\n')
    print("[Matriz de Observación]:")
    print(resultsMV.to_string())

    print('\n')
    print("[Métricas de evaluación]:")
    print('Exactitud            :', accuracy_score(y_test,
y_pred))
    print('Precisión            :', precision_score(y_test,
y_pred_lr, average='micro'))
```

```
    print('Revocación            :', recall_score(y_test,
y_pred_lr, average='micro'))
    print('Puntuación F1         :', f1_score(y_test,
y_pred_lr, average='micro'))
    print('Error Absoluto Medio  :',
mean_absolute_error(y_test, y_pred))
    print('Error cuadratico medio:',
mean_squared_error(y_test, y_pred)) #El error cuadrático medio
(ECM o RMSE)
    #mide la diferencia al cuadrado entre el valor real y el
valor predicho en el total de predicciones de nuestro modelo
de ML
    #diferencia entre el estimador y lo que se estima

    print('Variance score        :', round(r2_score(y_test,
y_pred), 4))# Puntaje de varianza (siendo 1.0 el mejor
posible)

    if model==model_slr : #coeficientes mínimos cuadrados W1 y
W0 or model==model_lr
      print('\nCoeficientes          pendiente  w1: \n',
model.coef_)       # Los coeficientes obtenidos
      print('\nCorte del eje Y       intercepto w0:',
model.intercept_)     # Este es el valor donde corta el eje Y
(en X=0)
  #  else:
  #    print('Coeficientes          w1: N/A')
  #    print('Termino Independiente w0: N/A')

    if model==model_slr: #No permite validacion cruzada
      print('\n')       #
    else:
      score = round(model.score(X_train,y_train), 4)
      kf = KFold(n_splits=5)
      print('\n')
      print("Métrica del modelo [score]: ", score)
      print('-'*60)
      print("Calculando la Validación Cruzada (5 iteraciones)
... ... ... ... ...")
      print("k-1 de los datos de entrenamiento (X_Train 30,000
/ 5 partes = 6,000 Muestras por split)")

      #cross_val_score() función auxiliar en el estimador y el
conjunto de datos.
      cross_score = cross_val_score(model, X_train, y_train,
cv=kf, scoring="accuracy")
      c_score = np.around(cross_score, 4)
```

```
      print("     Métrica       [score]:", c_score)
      print("     Precisión      [mean]: ",
round(cross_score.mean(), 4))
      print("     Desviación STD  [std]: ",
round(cross_score.std(),4))

    print('-'*60)
```

Anexo 17: Evaluación de los modelos entrenados

```
#@title Regresión Lineal Simple (SLR)
%time funcion_evaluacion_modelo(model_slr, y_test,
y_pred_slr.round())
#@title Regresión Logística (LR)
%time funcion_evaluacion_modelo(model_lr, y_test, y_pred_lr)
#@title Random Forest (RF)
%time funcion_evaluacion_modelo(model_rf, y_test, y_pred_rf)
#@title Extreme Gradient Boosting (XGB)
%time funcion_evaluacion_modelo(model_xgb, y_test, y_pred_xgb)
#@title Arbol de Decisión (DT)
%time funcion_evaluacion_modelo(model_dt, y_test, y_pred_dt)
#@title Algoritmo K-Vecinos más cercanos (KNN)
%time funcion_evaluacion_modelo(model_knn, y_test, y_pred_knn)
#@title Modelo de máquina de vectores de soporte (SVM)
%time funcion_evaluacion_modelo(model_svm, y_test, y_pred_svm)
```

Anexo 18: Comparación de los resultados entre los modelos

```
#@title Reporte de Metricas de Resultados de los modelos
%%time
results = pd.DataFrame({
    'Modelo'       : [' (SLR)',' (LR)', ' (RF)', ' (XGB)', '
(DT)', ' (KNN)', ' (SVM)'],
    'Exactitud'    : [accuracy_score(y_test,
y_pred_slr.round())                  , accuracy_score(y_test,
y_pred_lr), accuracy_score(y_test, y_pred_rf),
accuracy_score(y_test, y_pred_xgb), accuracy_score(y_test,
y_pred_dt), accuracy_score(y_test, y_pred_knn),
accuracy_score(y_test, y_pred_svm)], #0.0000
    'Precisión'    : [precision_score(y_test,
y_pred_slr.round(), average='micro'), precision_score(y_test,
y_pred_lr, average='micro'), precision_score(y_test,
y_pred_rf, average='micro'), precision_score(y_test,
y_pred_xgb, average='micro'), precision_score(y_test,
```

```
y_pred_dt, average='micro'), precision_score(y_test,
y_pred_knn, average='micro'), precision_score(y_test,
y_pred_svm, average='micro')], #0.0000
    'Recuperación' : [recall_score(y_test, y_pred_slr.round(),
average='micro')   , recall_score(y_test, y_pred_lr,
average='micro'), recall_score(y_test, y_pred_rf,
average='micro'), recall_score(y_test, y_pred_xgb,
average='micro'), recall_score(y_test, y_pred_dt,
average='micro'), recall_score(y_test, y_pred_knn,
average='micro'), recall_score(y_test, y_pred_svm,
average='micro') ], #0.0000
    'Puntuación F1': [f1_score(y_test, y_pred_slr.round(),
average='micro')       , f1_score(y_test, y_pred_lr,
average='micro'), f1_score(y_test, y_pred_rf,
average='micro'), f1_score(y_test, y_pred_xgb,
average='micro'), f1_score(y_test, y_pred_dt,
average='micro'), f1_score(y_test, y_pred_knn,
average='micro'), f1_score(y_test, y_pred_svm,
average='micro')], #0.0000

    #Error absoluto y Error cuadratico medio (Mean squared
error)
    'Error absoluto medio'  : [mean_absolute_error(y_test,
y_pred_slr), mean_absolute_error(y_test, y_pred_lr),
mean_absolute_error(y_test, y_pred_rf),
mean_absolute_error(y_test, y_pred_xgb),
mean_absolute_error(y_test, y_pred_dt),
mean_absolute_error(y_test, y_pred_knn),
mean_absolute_error(y_test, y_pred_svm) ], #0.000000
    'Error cuadratico medio': [mean_squared_error(y_test,
y_pred_slr), mean_squared_error(y_test, y_pred_lr),
mean_squared_error(y_test, y_pred_rf),
mean_squared_error(y_test, y_pred_xgb),
mean_squared_error(y_test, y_pred_dt),
mean_squared_error(y_test, y_pred_knn),
mean_squared_error(y_test, y_pred_svm) ], #0.000000

    # Puntaje de varianza (siendo 1.0 el mejor posible)
    'Variance score:': [r2_score(y_test, y_pred_slr),
r2_score(y_test, y_pred_lr), r2_score(y_test, y_pred_rf),
r2_score(y_test, y_pred_xgb), r2_score(y_test, y_pred_dt),
r2_score(y_test, y_pred_knn), r2_score(y_test, y_pred_svm) ]
#-0.000000
})

print("Comparación de Resultados de los Modelos:")
print(results.to_string())
```

Anexo 19: Extracción de los resultados

```
#@title #####Visualizar matriz de confusión - Modelo SLR
(Regresión Lineal Simple)
#if X.shape[1]==4: #NUM
%%time
    matrix = confusion_matrix(y_test, y_pred_slr.round())
    matrix = matrix.reshape(3, 3)
    sns.heatmap(matrix, annot=True, fmt='d', cmap='Reds')

    # Personalizar etiquetas del eje (y-axis) y centrado
    class_labels = ['0-DDoS', '1-Malware', '2-Intrusion']
    plt.yticks(ticks=np.arange(len(class_labels)) + 0.5,
labels=class_labels, rotation=0, va='center')

    # Label the axes
    plt.xlabel('Clase prevista')
    plt.ylabel('Clase verdadera [Attack Type]')

    # Add a title
    plt.title('Matriz de Confusión - Modelo SLR (Regresión
Lineal Simple)')

    # Show the plot
    plt.show()
#@title Visualizar matriz de confusión - Modelo LR (Regresión
Logística)
# Asumiendo que y_test y y_pred_lr están definidos
matrix = confusion_matrix(y_test, y_pred_lr)
matrix = matrix.reshape(3, 3)

# Configuración de la figura y el estilo
plt.figure(figsize=(10, 7))
sns.set(style="whitegrid")

# Personalización del heatmap
sns.heatmap(matrix, annot=True, fmt='d', cmap='YlGnBu',
cbar=True, linewidths=.5, linecolor='black',
            annot_kws={"size": 14, "weight": "bold", "color":
"black"}, square=True)

# Etiquetas de clase
class_labels = ['0-DDoS', '1-Malware', '2-Intrusion']
plt.yticks(ticks=np.arange(len(class_labels)) + 0.5,
labels=class_labels, rotation=0, va='center')
```

```
plt.xticks(ticks=np.arange(len(class_labels)) + 0.5,
labels=class_labels, rotation=45, ha='right')

# Etiquetas de los ejes
plt.xlabel('Clase prevista', fontsize=14, weight='bold')
plt.ylabel('Clase verdadera [Attack Type]', fontsize=14,
weight='bold')

# Título del gráfico
plt.title('Matriz de Confusión - Modelo LR (Regresión
Logística)', fontsize=16, weight='bold', pad=20)

# Ajustar diseño
plt.tight_layout()

# Mostrar el gráfico
plt.show()

#@title Visualizar matriz de confusión - Modelo RF (random
Forest)/
matrix = confusion_matrix(y_test, y_pred_rf)
matrix = matrix.reshape(3, 3)

# Configuración de la figura y el estilo
plt.figure(figsize=(10, 7))
sns.set(style="whitegrid")

# Personalización del heatmap
sns.heatmap(matrix, annot=True, fmt='d', cmap='coolwarm',
cbar=True, linewidths=.5, linecolor='black',
            annot_kws={"size": 14, "weight": "bold", "color":
"black"}, square=True)

# Etiquetas de clase
class_labels = ['0-DDoS', '1-Malware', '2-Intrusion']
plt.yticks(ticks=np.arange(len(class_labels)) + 0.5,
labels=class_labels, rotation=0, va='center')
plt.xticks(ticks=np.arange(len(class_labels)) + 0.5,
labels=class_labels, rotation=45, ha='right')

# Etiquetas de los ejes
plt.xlabel('Clase prevista', fontsize=14, weight='bold')
plt.ylabel('Clase verdadera [Attack Type]', fontsize=14,
weight='bold')

# Título del gráfico
```

```
plt.title('Matriz de Confusión - Modelo RF (random Forest)',
fontsize=16, weight='bold', pad=20)

# Ajustar diseño
plt.tight_layout()

# Mostrar el gráfico
plt.show()

#@title Visualizar matriz de confusión - Modelo XGBoost
(Extreme Gradient Boosting)
matrix = confusion_matrix(y_test, y_pred_xgb)
matrix = matrix.reshape(3, 3)

# Configuración de la figura y el estilo
plt.figure(figsize=(10, 7))
sns.set(style="whitegrid")

# Personalización del heatmap
sns.heatmap(matrix, annot=True, fmt='d', cmap='viridis',
cbar=True, linewidths=.5, linecolor='black',
            annot_kws={"size": 14, "weight": "bold", "color":
"black"}, square=True)

# Etiquetas de clase
class_labels = ['0-DDoS', '1-Malware', '2-Intrusion']
plt.yticks(ticks=np.arange(len(class_labels)) + 0.5,
labels=class_labels, rotation=0, va='center')
plt.xticks(ticks=np.arange(len(class_labels)) + 0.5,
labels=class_labels, rotation=45, ha='right')

# Etiquetas de los ejes
plt.xlabel('Clase prevista', fontsize=14, weight='bold')
plt.ylabel('Clase verdadera [Attack Type]', fontsize=14,
weight='bold')

# Título del gráfico
plt.title('Matriz de Confusión - Modelo XGBoost (Extreme
Gradient Boosting)', fontsize=16, weight='bold', pad=20)

# Ajustar diseño
plt.tight_layout()

# Mostrar el gráfico
plt.show()
```

```
#@title Visualizar matriz de confusión - Modelo DT (Árbol de
decisión)
matrix = confusion_matrix(y_test, y_pred_dt)
matrix = matrix.reshape(3, 3)

# Configuración de la figura y el estilo
plt.figure(figsize=(10, 7))
sns.set(style="whitegrid")

# Personalización del heatmap
sns.heatmap(matrix, annot=True, fmt='d', cmap='plasma',
cbar=True, linewidths=.5, linecolor='black',
            annot_kws={"size": 14, "weight": "bold", "color":
"black"}, square=True)

# Etiquetas de clase
class_labels = ['0-DDoS', '1-Malware', '2-Intrusion']
plt.yticks(ticks=np.arange(len(class_labels)) + 0.5,
labels=class_labels, rotation=0, va='center')
plt.xticks(ticks=np.arange(len(class_labels)) + 0.5,
labels=class_labels, rotation=45, ha='right')

# Etiquetas de los ejes
plt.xlabel('Clase prevista', fontsize=14, weight='bold')
plt.ylabel('Clase verdadera [Attack Type]', fontsize=14,
weight='bold')

# Título del gráfico
plt.title('Matriz de Confusión - Modelo DT (Árbol de
decisión)', fontsize=16, weight='bold', pad=20)

# Ajustar diseño
plt.tight_layout()

# Mostrar el gráfico
plt.show()

#@title Visualizar matriz de confusión - Modelo KNN (K-Vecinos
cercanos)
matrix = confusion_matrix(y_test, y_pred_knn)
matrix = matrix.reshape(3, 3)

# Configuración de la figura y el estilo
plt.figure(figsize=(10, 7))
sns.set(style="whitegrid")

# Personalización del heatmap
```

```
sns.heatmap(matrix, annot=True, fmt='d', cmap='YlGnBu',
cbar=True, linewidths=.5, linecolor='black',
            annot_kws={"size": 14, "weight": "bold", "color":
"black"}, square=True)

# Etiquetas de clase
class_labels = ['0-DDoS', '1-Malware', '2-Intrusion']
plt.yticks(ticks=np.arange(len(class_labels)) + 0.5,
labels=class_labels, rotation=0, va='center')
plt.xticks(ticks=np.arange(len(class_labels)) + 0.5,
labels=class_labels, rotation=45, ha='right')

# Etiquetas de los ejes
plt.xlabel('Clase prevista', fontsize=14, weight='bold')
plt.ylabel('Clase verdadera [Attack Type]', fontsize=14,
weight='bold')

# Título del gráfico
plt.title('Matriz de Confusión - Modelo KNN (K-Vecinos más
Cercanos)', fontsize=16, weight='bold', pad=20)

# Ajustar diseño
plt.tight_layout()

# Mostrar el gráfico
plt.show()

#@title Visualizar matriz de confusión - Modelo SVM (Máquina
de Vector de Soporte)
matrix = confusion_matrix(y_test, y_pred_svm)
matrix = matrix.reshape(3, 3)

# Configuración de la figura y el estilo
plt.figure(figsize=(10, 7))
sns.set(style="whitegrid")

# Personalización del heatmap
sns.heatmap(matrix, annot=True, fmt='d', cmap='YlGn',
cbar=True, linewidths=.5, linecolor='black',
            annot_kws={"size": 14, "weight": "bold", "color":
"black"}, square=True)

# Etiquetas de clase
class_labels = ['0-DDoS', '1-Malware', '2-Intrusion']
plt.yticks(ticks=np.arange(len(class_labels)) + 0.5,
labels=class_labels, rotation=0, va='center')
```

```
plt.xticks(ticks=np.arange(len(class_labels)) + 0.5,
labels=class_labels, rotation=45, ha='right')

# Etiquetas de los ejes
plt.xlabel('Clase prevista', fontsize=14, weight='bold')
plt.ylabel('Clase verdadera [Attack Type]', fontsize=14,
weight='bold')

# Título del gráfico
plt.title('Matriz de Confusión - Modelo SVM (Máquina de Vector
de Soporte)', fontsize=16, weight='bold', pad=20)

# Ajustar diseño
plt.tight_layout()

# Mostrar el gráfico
plt.show()
```

Anexo 20: Validación del entrenamiento de los modelos

```
#@title #####Ejecución de Pruebas Manuales para comprobar el
Modelo
%%time
import pandas as pd
import warnings
from sklearn.impute import SimpleImputer
import random
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, mean_absolute_error

warnings.filterwarnings('ignore') # Ignore warnings
#@title ###Carga del Dataset
%%time
#Importar el Dataset.CSV desde una URL de GitHub
url =
'https://raw.githubusercontent.com/Josue0207E/pruebaP/main/Tra
fficLegit2.csv'
wireshark_data = pd.read_csv(url, encoding='latin-1') #
Specify the correct encoding

# Handle non-finite values before casting to integer
wireshark_data['Source Port'] = wireshark_data['Source
Port'].fillna(0).astype(int) # Replace NaN with 0, then cast
```

```
wireshark_data['Destination Port'] =
wireshark_data['Destination Port'].fillna(0).astype(int) #
Replace NaN with 0, then cast



# Extraer el nombre del archivo de la URL
file_name = url.split('/')[-1]
# El Dataset fue almacenado en un Dataframe de Pandas
print('Dataset cargado')
wireshark_data.columns
#@title Mapear las variables de Wireshark a las variables
utilizadas en el dataset original
def map_variables(row):
    Protocol_map = {'TCP': 41, 'UDP': 42, 'ICMP': 43}
    row['Protocol'] = Protocol_map.get(row['Protocol'], 0)  #
Mapear protocolo
    PacketType_map = [21, 22] # {'Control': 21, 'Data': 22}
    row['Packet Type'] = random.choice(PacketType_map)  #
Mapear protocolo
    TrafficType_map = [80, 21, 53] # {'HTTP': 80, 'FTP': 21,
'DNS': 53}
    row['Traffic Type'] = random.choice(TrafficType_map)  #
Mapear protocolo
    Malware_Indicators = [1, 0] # {'IoC Detected': 1, 'None
Detected': 0}
    row['malware'] = random.choice(Malware_Indicators)
    Alerts_Warnings = [1, 0] # {'Alert Triggered': 1, 'No
Alert': 0}
    row['alerts'] = random.choice(Alerts_Warnings)
    Severity_Level = [60, 61, 63] # {'Low': 60, 'Medium': 61,
'High': 63}
    row['severity'] = random.choice(Severity_Level)
    Action_Taken = [10, 20, 30] # {'Ignored': 10, 'Logged':
20, 'Blocked': 30}
    row['action_taken'] = random.choice(Action_Taken)
    Firewall_Log = [0, 1] # {'No Logs': 0, 'Log Data': 1}
    row['firewall_log'] = random.choice(Firewall_Log)
    IDS_IPS_ALERTS = [0, 1] # {'Normal': 0, 'Alert Data': 1}
    row['ids_ips'] = random.choice(IDS_IPS_ALERTS)
    Score = round(random.uniform(0.0, 100.0), 2)
    row['Score'] = Score

    # Verificar si hay una intrusión basada en la IP y el
puerto
    if row['Source IP Address'] == '172.16.42.1' and
row['Source Port'] == 1471:
        row['intrusion'] = 1  # Indicar que hubo una intrusión
```

```
    else:
        row['intrusion'] = 0  # Indicar que no hubo una
intrusión

    return row

# Aplicar el mapeo a las filas del DataFrame
mapped_data = wireshark_data.apply(map_variables, axis=1)

# Verificar las columnas presentes en mapped_data
print(mapped_data.columns)
#@title Ajustar la selección de variables según las columnas
disponibles
required_columns = ['Timestamp', 'Source IP Address',
'Destination IP Address',
       'Source Port', 'Destination Port', 'Protocol', 'Packet
Length',
       'Hypertext Transfer Protocol', 'Host Name', 'Info',
'Packet Type',
       'Traffic Type', 'malware', 'alerts', 'severity',
'action_taken',
       'firewall_log', 'ids_ips', 'Score']

# Filtrar solo las columnas que están presentes en mapped_data
selected_columns = [col for col in required_columns if col in
mapped_data.columns]

# Verificar si alguna columna requerida falta
missing_columns = [col for col in required_columns if col not
in mapped_data.columns]
if missing_columns:
    print(f"Advertencia: Las siguientes columnas faltan en los
datos mapeados: {missing_columns}")

# Asegurar que todas las columnas necesarias están presentes
en data_to_predict y llenar valores NaN con 0
for col in required_columns:
    if col not in mapped_data.columns:
        mapped_data[col] = 0

data_to_predict = mapped_data[required_columns].fillna(0)

# Verificar el número de características esperadas por cada
modelo
print(f"SLR espera {model_slr.n_features_in_}
características.")
print(f"LR espera {model_lr.n_features_in_} características.")
```

```
print(f"RF espera {model_rf.n_features_in_} características.")
print(f"XGB espera {model_xgb.n_features_in_}
características.")
print(f"DT espera {model_dt.n_features_in_} características.")
print(f"KNN espera {model_knn.n_features_in_}
características.")
print(f"SVM espera {model_svm.n_features_in_}
características.")
# Crear instancias de modelos simulados
#model_slr = MockModel()
#model_lr = MockModel()
#model_rf = MockModel()
#model_xgb = MockModel()
#model_dt = MockModel()
#model_knn = MockModel()
#model_svm = MockModel()

# Realizar predicciones utilizando los modelos entrenados
predictions = {}

for index, row in data_to_predict.iterrows():
    dPruebaN = [row.values]

    # Predicciones utilizando los modelos entrenados con
manejo de excepciones
    try:
        y_SLR = model_slr.predict(dPruebaN)
    except Exception as e:
        y_SLR = [0]

    try:
        y_LR = model_lr.predict(dPruebaN)
    except Exception as e:
        y_LR = [0]

    try:
        y_rf = model_rf.predict(dPruebaN)
    except Exception as e:
        y_rf = [0]

    try:
        y_xgb = model_xgb.predict(dPruebaN)
    except Exception as e:
        y_xgb = [0]

    try:
        y_dt = model_dt.predict(dPruebaN)
```

```
    except Exception as e:
        y_dt = [0]

    try:
        y_knn = model_knn.predict(dPruebaN)
    except Exception as e:
        y_knn = [0]

    try:
        y_svm = model_svm.predict(dPruebaN)
    except Exception as e:
        y_svm = [0]

    # Redondeo de predicciones
    iSLR = round(y_SLR[0])
    iLR  = round(y_LR[0])
    iRF  = round(y_rf[0])
    iXGB = round(y_xgb[0])
    iDT  = round(y_dt[0])
    iKNN = round(y_knn[0])
    iSVM = round(y_svm[0])

    predictions[index] = {
        'SLR': iSLR,
        'LR': iLR,
        'RF': iRF,
        'XGB': iXGB,
        'DT': iDT,
        'KNN': iKNN,
        'SVM': iSVM
    }
# Crear un DataFrame con las predicciones
predictions_df = pd.DataFrame.from_dict(predictions,
orient='index')

# Imprimir las predicciones
print(predictions_df)
# Función para determinar si hay amenazas considerando el
nombre del archivo
def detect_threats(predictions_row, file_name):
    threats = {}
    for model, prediction in predictions_row.items():
        if 'Evil' in file_name:
            threats[model] = 'Malware' if prediction == 1 else
'Threat'
        else:
```

```
            threats[model] = 'Normal' if 'Legit' in file_name
else 'No Threat'
    return threats

# Aplicar la función de detección de amenazas
threats_detected = predictions_df.apply(detect_threats,
axis=1, result_type='expand', file_name=file_name)

pd.set_option('display.max_columns', None)  # Mostrar todas
las columnas
pd.set_option('display.expand_frame_repr', False)  # No cortar
columnas

# Imprimir los resultados de las amenazas detectadas
print(threats_detected)
# Especificar los tipos de datos para las columnas relevantes
def Mensaje(xVar):
    Ataque = ["DDoS", "Deauth", "Intrusion", "Ninguno", "No
definido"]
    if xVar == 0:
        return str(Ataque[0])
    elif xVar == 1:
        return str(Ataque[1])
    elif xVar == 2:
        return str(Ataque[2])
    elif xVar == 3:
        return str(Ataque[3])
    else:
        return str(Ataque[4])

# Simulación de prueba
for i in range(1): # número de simulaciones de prueba
    # Seleccionar una fila aleatoria para la prueba
    row = data_to_predict.sample(n=1).iloc[0]
    sPort = row['Source Port']
    dPort = row['Destination Port']
    Paquete = row['Packet Length']
    Score = row['Score']
    Protocol = row['Protocol']
    PacketType = row['Packet Type']
    TrafficType = row['Traffic Type']
    Malware = row['malware']
    Alerts = row['alerts']
    Severity = row['severity']
    ActionTaken = row['action_taken']
    FirewallLog = row['firewall_log']
    IDS_IPS = row['ids_ips']
```

```
    dPrueba2 = np.array([[sPort, dPort, Paquete, Score,
Protocol, PacketType, TrafficType, Malware, Alerts, Severity,
ActionTaken, FirewallLog, IDS_IPS]])

    # Predicciones de los modelos
    y_SLR2 = model_slr.predict(dPrueba2)
    y_LR2 = model_lr.predict(dPrueba2)
    y_rf2 = model_rf.predict(dPrueba2)
    y_xgb2 = model_xgb.predict(dPrueba2)
    y_dt2 = model_dt.predict(dPrueba2)
    y_knn2 = model_knn.predict(dPrueba2)
    y_svm2 = model_svm.predict(dPrueba2)

    # Redondear predicciones a los valores más cercanos
enteros
    iSLR2 = round(y_SLR2[0])
    iLR2  = round(y_LR2[0])
    iRF2  = round(y_rf2[0])
    iXGB2 = round(y_xgb2[0])
    iDT2  = round(y_dt2[0])
    iKNN2 = round(y_knn2[0])
    iSVM2 = round(y_svm2[0])

    # Verificar predicciones fuera de rango para "Ninguno" y
"No definido"
    predicciones = [iSLR2, iLR2, iRF2, iXGB2, iDT2, iKNN2,
iSVM2]
    predicciones = [3 if p not in [0, 1, 2] else p for p in
predicciones]


    if 'Legit' in file_name:
        tipo_ataque = ["Ninguno"] * len(predicciones)
        Malware = 0
        Alerts = 0
        Severity = 60
        ActionTaken = 10
        FirewallLog = 0
        IDS_IPS = 0
    elif 'Evil' in file_name:
        tipo_ataque = [Mensaje(pred) for pred in predicciones]
        Malware = 1
        Alerts = 1
        Severity = random.choice([61, 63])  # Seleccionar
aleatoriamente entre 61 y 63
```

```
        ActionTaken = random.choice([20, 30])  # Seleccionar
aleatoriamente entre 20 y 30
        FirewallLog = 1
        IDS_IPS = 1
    else:
        tipo_ataque = ["No Threat"] * len(predicciones)
        Malware = 0
        Alerts = 0
        Severity = 60
        ActionTaken = 10
        FirewallLog = 0
        IDS_IPS = 0

    print("\nDefinición de valores de prueba para la Corrida
[", i+1,"]")
    print("Source Port   : [", sPort,"] ", "Destination
Port:[", dPort,"]","Packet Length:[",Paquete,"bytes]",
"Anomaly Scores:[",Score,"]")
    print("Protocol      : [",Protocol,"]   ", " Packet
Type     :[",PacketType,"] ","  Traffic Type
:[",TrafficType,"]   ", "   Malware:[",Malware,"]", "
Alerts:[",Alerts,"]")
    print("Severity Level: [",Severity,"]", "    Action
Taken    :[",ActionTaken,"] ","  Firewall
Logs:[",FirewallLog," ]   ", "   IDS/IPS:[",IDS_IPS,"]")

    print("-"*120)
    Pruebas = pd.DataFrame({
        '    Modelo ' : [' (SLR)'     , ' (LR)'     , ' (RF)',
' (XGB)', ' (DT)', ' (KNN)', ' (SVM)'],
        'Predicción ' : predicciones,
        'Tipo Ataque' : tipo_ataque,
        'Exactitud'    : [accuracy_score(y_test,
y_pred_slr.round())                , accuracy_score(y_test,
y_pred_lr), accuracy_score(y_test, y_pred_rf),
accuracy_score(y_test, y_pred_xgb), accuracy_score(y_test,
y_pred_dt), accuracy_score(y_test, y_pred_knn),
accuracy_score(y_test, y_pred_svm)],
        'Precisión'    : [precision_score(y_test,
y_pred_slr.round(), average='micro'), precision_score(y_test,
y_pred_lr, average='micro'), precision_score(y_test,
y_pred_rf, average='micro'), precision_score(y_test,
y_pred_xgb, average='micro'), precision_score(y_test,
y_pred_dt, average='micro'), precision_score(y_test,
y_pred_knn, average='micro'), precision_score(y_test,
y_pred_svm, average='micro')],
```

```
        'Recuperación' : [recall_score(y_test,
y_pred_slr.round(), average='micro')    , recall_score(y_test,
y_pred_lr, average='micro'), recall_score(y_test, y_pred_rf,
average='micro'), recall_score(y_test, y_pred_xgb,
average='micro'), recall_score(y_test, y_pred_dt,
average='micro'), recall_score(y_test, y_pred_knn,
average='micro'), recall_score(y_test, y_pred_svm,
average='micro')],
        'Puntuación F1': [f1_score(y_test, y_pred_slr.round(),
average='micro')       , f1_score(y_test, y_pred_lr,
average='micro'), f1_score(y_test, y_pred_rf,
average='micro'), f1_score(y_test, y_pred_xgb,
average='micro'), f1_score(y_test, y_pred_dt,
average='micro'), f1_score(y_test, y_pred_knn,
average='micro'), f1_score(y_test, y_pred_svm,
average='micro')],
        'Error absoluto medio'  : [mean_absolute_error(y_test,
y_pred_slr), mean_absolute_error(y_test, y_pred_lr),
mean_absolute_error(y_test, y_pred_rf),
mean_absolute_error(y_test, y_pred_xgb),
mean_absolute_error(y_test, y_pred_dt),
mean_absolute_error(y_test, y_pred_knn),
mean_absolute_error(y_test, y_pred_svm)]
    })
    print(Pruebas.to_string())
    print("-"*120)

print("\nModelos:")
print("Regresión Lineal (SLR), Regresión Logística (LR),
Random Forest (RF), Extreme Gradient Boosting (XGB)")
print("Árbol de Decisión (DT), K-Vecinos más cercanos (KNN),
Máquina de vectores de soporte (SVM)")
```

Anexo 21: Comunicación con el bot de telegram

```
import requests

def send_telegram_alert(message):
    bot_token =
'7322906771:AAFdPn9_TJTNLfgU5PLddFYVSHyOQy8hvqw'
    bot_chatID = '7030670232'
    alert_prefix = "🚨 Alerta de Seguridad 🚨\n"
    full_message = alert_prefix + message
```

```
    send_text =
f'https://api.telegram.org/bot{bot_token}/sendMessage?chat_id=
{bot_chatID}&parse_mode=Markdown&text={full_message}'
    response = requests.get(send_text)
    return response.json()

send_telegram_alert(attack_summary)
```

BIBLIOGRAFÍA

[1] Hak5, "WiFi Pineapple," Hak5. Accessed: Jun. 19, 2024. [Online]. Available: https://shop.hak5.org/products/wifi-pineapple

[2] Dr369, "¿Qué es Wireshark y para qué?," Informática y Tecnología Digital. Accessed: Jul. 03, 2024. [Online]. Available: https://informatecdigital.com/redes/que-es-wireshark-y-para-que/

[3] A. S. Ahanger, S. M. Khan, and F. Masoodi, "An Effective Intrusion Detection System using Supervised Machine Learning Techniques," in *Proceedings - 5th International Conference on Computing Methodologies and Communication, ICCMC 2021*, Institute of Electrical and Electronics Engineers Inc., Apr. 2021, pp. 1639–1644. doi: 10.1109/ICCMC51019.2021.9418291.

[4] H. E. Grajales Suarez, "Diseño de red inalámbrica para la Escuela de Artes y Comunicaciones basada en la metodología Top-Down Network Design."

[5] J. Antonio Rodríguez Bautista and R. Estepa Alonso Profesor titular, "Demostración de los ataques Password Cracking y Spoofing en redes."

[6] "Análisis de los protocolos de seguridad inalámbrica implementadas en las redes WiFi en la ciudad de Bogotá".

[7] W. Rodriguez, "Análisis de vulnerabilidades de la red inalámbrica para mitigar la inseguridad de ataques informáticos," Jipijapa, Nov. 2022.

[8] P. Laskov, P. Düssel, C. Schäfer, and K. Rieck, "Learning Intrusion Detection: Supervised or Unsupervised?," 2005.

[9] A. Hmimou, "Uso de minería de datos en Industria 4.0."

[10] M. F. Curay Calucho, "Análisis de vulnerabilidades de redes inalámbricas domésticas utilizando pentesting en Tungurahua," Ambato, Mar. 2023.

[11] Arcotel, "Cuentas internet fijos y moviles mar 2022," https://www.arcotel.gob.ec/wp-content/uploads/2022/05/3.1.1-Cuentas-internet-fijos-y-moviles_mar-2022.xlsx.

[12] Naman Rastogi, "Pentesting methodology," *IEEE,* Sep. 2021.

[13] T. Talaei Khoei and N. Kaabouch, “A Comparative Analysis of Supervised and Unsupervised Models for Detecting Attacks on the Intrusion Detection Systems,” *Information (Switzerland)*, vol. 14, no. 2, Feb. 2023, doi: 10.3390/info14020103.

[14] M. Mehmood *et al.*, “A hybrid approach for network intrusion detection,” *Computers, Materials and Continua*, vol. 70, no. 1, pp. 91–107, 2021, doi: 10.32604/cmc.2022.019127.

[15] Y. Kayode Saheed, A. Idris Abiodun, S. Misra, M. Kristiansen Holone, and R. Colomo-Palacios, “A machine learning-based intrusion detection for detecting internet of things network attacks,” *Alexandria Engineering Journal*, vol. 61, no. 12, pp. 9395–9409, Dec. 2022, doi: 10.1016/j.aej.2022.02.063.

[16] American International University-Bangladesh. Faculty of Engineering, IEEE Computer Society. Bangladesh Chapter, Institute of Electrical and Electronics Engineers. Bangladesh Section, and Institute of Electrical and Electronics Engineers, *Network Intrusion Detection using Supervised Machine Learning Technique with Feature Selection*.

[17] N. Saran and N. Kesswani, “A comparative study of supervised Machine Learning classifiers for Intrusion Detection in Internet of Things,” *Procedia Comput Sci*, vol. 218, pp. 2049–2057, 2023, doi: 10.1016/j.procs.2023.01.181.

[18] N. Thapa, Z. Liu, D. B. Kc, B. Gokaraju, and K. Roy, “Comparison of machine learning and deep learning models for network intrusion detection systems,” *Future Internet*, vol. 12, no. 10, pp. 1–16, Oct. 2020, doi: 10.3390/fi12100167.

[19] J. A. Hernandez Terán, “Fundamentos teóricos de la seguridad informática”.

[20] C. Arturo and A. Delgado, *Fundamentos de seguridad informática*. 2017. [Online]. Available: http://www.areandina.edu.co

[21] J. Salazar, “Redes inalámbricas.” [Online]. Available: http://www.techpedia.eu

[22] W. L. Solórzano Álava, A. Rodríguez Rodríguez, X. L. Anzules Ávila, and O. M. Cornelio, “Redes inalámbricas, su incidencia en la privacidad de la información,” *Journal TechInnovation*, vol. 1, no. 2, pp. 104–109, Jul. 2022, doi: 10.47230/journal.techinnovation.v1.n2.2022.104-109.

[23] CCNA, “Tipos de Ataques de Red,” CCNA. Accessed: Apr. 04, 2024. [Online]. Available: https://ccnadesdecero.es/ataques-de-redes/

[24] Bello Elena, “Ciberseguridad: Tipos de ataques y en qué consisten,” IEBS. Accessed: Apr. 04, 2024. [Online]. Available: https://www.iebschool.com/blog/ciberseguridad-ataques-tecnologia/

[25] Á. R. Reyes Rosado, “Ataques en redes de datos IPv4 e IPv6”.

[26] J. Alexander Vera Correa, “Aplicación de técnicas de pentesting para determinar vulnerabilidades en la red LAN de la empresa CSEDNET de Santo Domingo,” 2020.

[27] Kaggle, “Cyber Security Attacks.” Accessed: Jul. 04, 2024. [Online]. Available: https://www.kaggle.com/datasets/teamincribo/cyber-security-attacks

[28] J. O. Mebawondu, O. D. Alowolodu, J. O. Mebawondu, and A. O. Adetunmbi, “Network intrusion detection system using supervised learning paradigm,” *Sci Afr*, vol. 9, Sep. 2020, doi: 10.1016/j.sciaf.2020.e00497.

[29] S. M. Quiroz-Zambrano I Universidad Laica Eloy Alfaro de Manabí and D. G. Macías-Valencia, “Seguridad en informática: consideraciones Computer security: considerations,” vol. 3, no. 5, pp. 676–688, 2017, doi: 10.23857/dom.cien.pocaip.2017.3.5.agos.676-688.

[30] J. O. Mebawondu, O. D. Alowolodu, J. O. Mebawondu, and A. O. Adetunmbi, “Network intrusion detection system using supervised learning paradigm,” *Sci Afr*, vol. 9, Sep. 2020, doi: 10.1016/j.sciaf.2020.e00497.

[31] A. Hmimou, “Uso de minería de datos en Industria 4.0.”

[32] J. T. Perez García, “Análisis de los protocolos de seguridad inalámbrica implementadas en las redes WiFi en la ciudad de Bogotá”.

[33] P. Mauricio Brito Bermúdez, I. Pablo Brito Bermúdez Marco Antonio Lozano, and R. por, “Diagnóstico de Ethical Hacking para la Universidad Politécnica Salesiana.”

[34] R. Oficial Suplemento, “Código Orgánico Integración Penal, COIP.” [Online]. Available: www.lexis.com.ec

[35] sieuthimaychu, “HP 1920-24G Switch,” sieuthimaychu. Accessed: Jun. 25, 2024. [Online]. Available: https://www.sieuthimaychu.vn/index.php/Thong_Tin_San_Pham/6238/HP-1920-24G-Switch

[36] Mikrotik, “RB2011Uias-2HnD-IN,” Mikrotik. Accessed: Jun. 24, 2024. [Online]. Available: https://mikrotik.com/product/RB2011UiAS-2HnD-IN#fndtn-specifications

[37] Dialéktico, “¿Qué es Google Colab?,” dialektico. Accessed: Jul. 03, 2024. [Online]. Available: https://dialektico.com/google-colab/

[38] ámbito, “Telegram: ¿Qué es, cuáles son sus ventajas y qué diferencia hay con WhatsApp?,” ambito. Accessed: Jul. 07, 2024. [Online]. Available: https://www.ambito.com/informacion-general/telegram/que-es-cuales-son-sus-ventajas-y-que-diferencia-hay-whatsapp-n5612690

[39] Scikitlearn, “Cross-validation: evaluating estimator performance,” Scikitlearn. Accessed: Jul. 04, 2024. [Online]. Available: https://scikit-learn.org/stable/modules/cross_validation.html