
Desarrollo de un sistema para la detección de intrusos en entornos domésticos utilizados para trabajos empresariales, mediante análisis del tráfico de red.

Diego José Franco Pacay



UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería



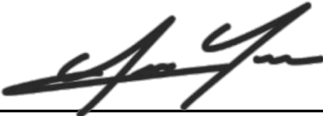
Desarrollo de un sistema para la detección de intrusos en entornos domésticos utilizados para trabajos empresariales, mediante análisis del tráfico de red.

Trabajo de graduación en modalidad de trabajo profesional presentado
por
Diego José Franco Pacay


Para optar al grado académico de Licenciado en Ingeniería En Ciencias
de la Computación y Tecnologías de la Información

Guatemala, diciembre del 2024


Vo.Bo.:

(f) 
MSc. Jorge Yass

Tribunal Examinador:

(f) 
PhD. Gabriel Barrientos

(f) 
Ing. Melinton Navas

(f) 
MSc. Jorge Yass

Fecha de aprobación: Guatemala, 03 de diciembre de 2,024.

Con el aumento del 13 % en la presencia de malware en entornos empresariales tras la pandemia del COVID-19[1], la preocupación por la seguridad de los datos en redes personales ha aumentado. En este contexto, el presente trabajo se centra en el desarrollo de un sistema de detección de intrusos diseñado específicamente para entornos domésticos utilizados para trabajos empresariales, con el objetivo de determinar paquetes de red que exhiban comportamientos inusuales dentro de ésta.

Para el desarrollo de la herramienta se incluyó la creación de un módulo de captura de paquetes, que permite la interceptación y almacenamiento del tráfico de datos entrante y saliente, así como un módulo de análisis que integra un algoritmo de aprendizaje automático para la detección de intrusos.

La combinación de estos módulos resultó en un sistema eficiente para la detección de intrusiones. Logrando identificar características críticas en los paquetes de datos, tales como direcciones IP, puertos de origen y destino, y volúmenes de tráfico. Para este propósito, se utilizó un dataset compuesto por tráfico de red etiquetado, lo que permitió realizar un proceso de selección de características que optimizó la identificación de patrones anómalos.

La implementación del algoritmo *Support Vector Machine* para la clasificación de los paquetes de red, obtuvo una exactitud del 82 %, destacándose por su capacidad para discriminar entre tráfico malicioso. Este enfoque es especialmente relevante para la detección de intrusiones en entornos domésticos utilizados para trabajos empresariales, donde la seguridad de los datos es primordial.

With the 13% increase in the presence of malware in enterprise environments following the COVID-19 pandemic[1], concern for data security in personal networks has increased. In this context, the present work focuses on the development of an intrusion detection system designed specifically for home environments used for business work. With the objective of determining network packets that exhibit unusual behavior inside the network.

The development of the tool included the creation of a packet capture module, which allows the interception and storage of incoming and outgoing data traffic, as well as an analysis module that integrates a machine learning algorithm for intrusion detection.

The combination of these modules resulted in an efficient intrusion detection system. Critical characteristics in the data packets, such as IP addresses, source and destination ports, and traffic volumes, were identified. For this purpose, a dataset composed of labeled network traffic was used, which allowed a feature selection process that optimized the identification of anomalous patterns.

The implementation of the *Support Vector Machine* algorithm for network packet classification obtained an accuracy of 82%, standing out for its ability to discriminate between malicious traffic. This approach is especially relevant for intrusion detection in home environments used for business work, where data security is paramount.

Tabla de Contenido

| | |
|---|-------------|
| Resumen | III |
| Abstract | IV |
| Lista de figuras | VII |
| Lista de cuadros | VIII |
| 1. Introducción | 1 |
| 2. Justificación | 2 |
| 3. Objetivos | 3 |
| 3.1. Objetivo General | 3 |
| 3.2. Objetivos Específicos | 3 |
| 4. Marco teórico | 4 |
| 4.1. Seguridad de redes | 4 |
| 4.2. Tipos de amenazas en redes domésticas | 4 |
| 4.2.1. Clasificación de malware | 5 |
| 4.2.2. Tipos de ataques cibernéticos | 5 |
| 4.2.3. Categorización de intrusos | 6 |
| 4.2.4. Estrategias de defensa y contramedidas | 7 |
| 4.3. Sistema de detección de intrusos | 7 |
| 4.4. Paquetes de Red | 8 |
| 4.4.1. Métodos de Análisis del Tráfico de Red | 9 |
| 4.5. Aplicación del Aprendizaje Automático | 9 |
| 4.5.1. Técnicas de aprendizaje automático | 10 |
| 4.5.2. Modelos de machine learning comúnmente utilizados | 10 |
| 4.6. Características de Paquetes de Red | 11 |
| 4.7. Conjunto de datos | 11 |
| 5. Metodología | 12 |
| 5.1. Exploración de datos | 12 |
| 5.2. Ingeniería de características | 12 |
| 5.2.1. Selección y Eliminación de Características | 14 |
| 5.2.2. Criterios para la Eliminación de Características | 15 |
| 5.2.3. Priorización de Información Esencial para Detección de Anomalías | 15 |

| | |
|---|-----------|
| 5.2.4. Estadísticas Descriptivas del conjunto de datos | 17 |
| 5.3. Desarrollo módulo de captura de paquetes | 17 |
| 5.3.1. Configuración de la Captura | 18 |
| 5.4. Desarrollo módulo para la detección de intrusos | 18 |
| 5.4.1. Support Vector Machine | 19 |
| 5.4.2. Naive Bayes | 19 |
| 5.5. Pruebas de algoritmos | 19 |
| 5.6. Integración de los módulos anteriores | 20 |
| 6. Resultados | 22 |
| 6.1. Resultados de los Modelos de Aprendizaje Automático | 22 |
| 6.1.1. Modelo Support Vector Machine | 22 |
| 6.1.2. Modelo Naive Bayes | 24 |
| 6.2. Resultados de la Implementación Híbrida | 26 |
| 6.2.1. Implementacion híbrida 1, SVM seguido de Naive Bayes | 26 |
| 6.2.2. Implementacion híbrida 2, Naive Bayes seguido de SVM | 27 |
| 7. Discusión | 29 |
| 7.1. Desempeño de los Modelos | 29 |
| 7.1.1. Modelo Support Vector Machine | 30 |
| 7.1.2. Modelo Naive Bayes | 32 |
| 7.1.3. Comparación entre Modelos | 34 |
| 7.2. Implementación híbrida | 35 |
| 7.2.1. SVM seguido de Naive Bayes | 35 |
| 7.2.2. Naive Bayes seguido de SVM | 37 |
| 8. Conclusiones | 39 |
| 9. Recomendaciones | 41 |
| Bibliografía | 43 |

Lista de figuras

| | |
|--|----|
| 5.1. Porcentaje de valores nulos por característica en el conjunto de datos inicial. | 13 |
| 5.2. Matriz de correlación de características del conjunto de datos. | 14 |
| 5.3. Flujo de trabajo de la herramienta de detección de intrusos. | 21 |
| 6.1. Matriz de Confusión del Modelo Support Vector Machine. | 23 |
| 6.2. Curva ROC del Modelo Support Vector Machine. | 23 |
| 6.3. Matriz de Confusión del Modelo Naive Bayes. | 24 |
| 6.4. Curva ROC del Modelo Naive Bayes. | 25 |
| 6.5. Matriz de Confusión de la implementación híbrida 1. | 26 |
| 6.6. Matriz de Confusión de la implementación híbrida 2. | 27 |
| 7.1. Matriz de Confusión del Modelo Support Vector Machine. | 31 |
| 7.2. Curva ROC del Modelo Support Vector Machine. | 32 |
| 7.3. Matriz de Confusión del Modelo Naive Bayes. | 33 |
| 7.4. Curva ROC del Modelo Naive Bayes. | 34 |
| 7.5. Matriz de Confusión de la implementación híbrida 1. | 36 |
| 7.6. Matriz de Confusión de la implementación híbrida 2. | 37 |

Lista de cuadros

| | |
|---|----|
| 5.1. Características finales. | 16 |
| 5.2. Ejemplo de Captura de Paquetes | 18 |
| 6.1. Métricas del Modelo Support Vector Machine | 22 |
| 6.2. Métricas del Modelo Naive Bayes | 24 |
| 6.3. Resumen de Resultados | 25 |
| 6.4. Métricas de la implementacion híbrida 1 | 26 |
| 6.5. Métricas de la implemetación hóibrida 2 | 27 |

Desde la pandemia del COVID-19, se ha notado un incremento del 13 % en la presencia de malwares en entornos empresariales [1], esto se debe a la falta de seguridad en las computadoras utilizadas para realizar tareas profesionales desde casa. Según un estudio realizado por ManageEngine, el 37 % de los dispositivos utilizados por los trabajadores para llevar a cabo sus tareas desde casa carecen de los elementos necesarios para conectarse de forma segura a los recursos corporativos y proteger la información gestionada en ellos[2]. La gran mayoría de veces, estas redes no poseen las medidas de seguridad necesarias para evitar o mitigar un ataques cibernéticos, dejando así la oportunidad para que intrusos entren y lancen ataques[3].

La falta de soluciones efectivas para detectar y reducir las amenazas que estos intrusos pueden suponer, lo antes posible, supone un riesgo importante para la integridad y confidencialidad de la información empresarial sensible[4]. Por lo tanto surge una preocupación crítica: Detectar de manera efectiva los intrusos en estos entornos y así poder mitigar las posibles consecuencias que éstos pueda causar dentro del sistema de la empresa. Para abordar esta necesidad, este proyecto se centra en desarrollar un sistema que pueda detectar amenazas en redes domésticas para el uso de tareas empresariales mediante la inspección de paquetes de red.

Este proyecto propone la implementación de un modelo de aprendizaje automático para detectar la presencia de intrusos, junto con un módulo eficiente y seguro para capturar paquetes de red. Además, se busca que el resultado final tenga la capacidad de detectar en su mayoría los paquetes de red que indiquen la posibilidad de ser generado por un intruso o actividad maliciosa. Esto con el fin de proteger los activos digitales de las organizaciones y los entes involucrados en la red, garantizar la continuidad de sus operaciones en entornos remotos por medio de la adición de una capa adicional de seguridad.

En la era en la que vivimos, las redes informáticas son indispensables para todas las personas, pero principalmente para el funcionamiento de muchas empresas. Sin embargo, estas redes también son vulnerables a ataques de ciberdelincuentes, lo cual puede recaer en grandes consecuencias: pérdidas económicas, robo de información personal, daños a la reputación.[5]

Actualmente existen diferentes formas de detectar estos ataques dentro de las redes, como el análisis de firmas, el análisis de comportamiento, el uso de sandboxes, entre otras. Pero, estas técnicas tienen diferentes limitaciones:

- Incapacidad de detectar ataques nuevos: la técnica de firmas consta de comparar la firma que identifica a un archivo del sistema con bases de datos previamente cargadas lo que significa capturar únicamente acciones conocidas.
- Falsos positivos: muchas técnicas generan gran cantidad de falsos positivos, haciendo que la detección de amenazas reales sea más compleja.
- Disposición de mucho tiempo y recursos: técnicas como el sandbox se centra en identificar formas de ataque nuevas, lo que lo hace una técnica muy específica, requiriendo de mucho esfuerzo tanto monetario como temporal.

Estas limitaciones conllevan a que la detección de los ataques dentro de la red sean muy tardías y que la institución o ente tenga serias consecuencias.[6]

Como se menciona en el estudio "Does Every Second Count? Time-based Evolution of Malware Behavior in Sandboxes", la detección temprana de malware es crucial para minimizar el daño potencial que puede causar. Si el malware se detecta y se elimina rápidamente, es menos probable que cause daños graves a los sistemas informáticos o a los datos[7]. Esta importancia se acentúa al considerar que "El tiempo promedio de detección de un ataque cibernético es de 206 días, lo que significa que los intrusos pueden permanecer en los sistemas de una empresa durante meses sin ser detectados[8]."

Todo esto sin mencionar que las técnicas de detección comúnmente no se implementan en las redes domésticas sino solo a redes empresariales, dejando las redes domésticas completamente vulnerables a ataques y estos serían prácticamente indetectables.

3.1. Objetivo General

Desarrollar un sistema de detección de intrusos en entornos domésticos utilizados para trabajos empresariales, capaz de detectar paquetes que exhiban comportamientos inusuales dentro de la red.

3.2. Objetivos Específicos

- Identificar las características de un paquete de red que indiquen la posibilidad de ser generado por un intruso.
- Implementar un módulo de captura de paquetes de red que pueda interceptar y almacenar el tráfico de red entrante y saliente de la computadora utilizada para realizar las tareas profesionales.
- Integrar algoritmos de análisis de paquetes para identificar patrones sospechosos que podrían indicar la presencia de intrusos, utilizando técnicas de aprendizaje automático.

4.1. Seguridad de redes

La seguridad de redes es un área crítica que se encarga de brindar seguridad de las redes es aquel que está enfocado en proteger los datos y recursos informáticos que transitan a través de dicha red informática. El cumplimiento de esto se logran a través de varias actividades y dispositivos dirigidos a impedir el acceso no autorizado, los ataques cibernéticos y los daños al sistema de información basándose en los pilares de la seguridad informática.[9]

Los tres pilares fundamentales de la seguridad de redes son:

- **La integridad de la información:** hace referencia a que la información sea adecuada, concreta y libre de errores y modificaciones.
- **La disponibilidad de la información:** hace referida a que la información esté siempre disponible cuando la necesitamos.
- **La confidencialidad:** implica que la información estará disponible solamente para el personal autorizado, de hecho, se conoce con el nombre de need-to-know, Esto hace referencia a que la información no estará en manos de personas, entidades o sistemas que no sea permitido necesario.

Teniendo definidos a que hacen referencias los pilares de la seguridad, se tiene un marco claro sobre cuál es el enfoque y los alcances que se deben cubrir cuando se está hablando sobre la seguridad de las redes. [10]

4.2. Tipos de amenazas en redes domésticas

Los elementos principales que en un sistema informático se debe proteger son tanto el hardware, como el software;

- El hardware, son los elementos físicos, provistos de elementos electrónicos, eléctricos, mecánicos, entre otros, que sustentan los computadores y sus periféricos, como: teclado, cámaras,

pantallas, etc. . .

- Mientras que el software, son programas y aplicaciones incluidos los sistemas operativos que hacen funcionar el hardware o ejecutar actividades programadas a través de secuencias de instrucciones.

Son los datos que dentro de estos sistemas los que se deben proteger con mayor énfasis, ya que todos los elementos de un sistema informático están amenazados y en cualquier momento expuestos a que un ciberdelincuente ejecute un ataque sobre ellos con la finalidad de extraer información.[5]

4.2.1. Clasificación de malware

En este contexto, uno de los principales peligros que amenazan la integridad de los datos, así como del hardware y software, es el malware. El malware, o software malicioso, es cualquier programa o archivo que tiene como objetivo dañar, explotar o realizar acciones no deseadas en dispositivos, redes o servicios[11]. Los malwares pueden clasificarse en diversas categorías, cada una con sus propias características y métodos de ataque:

- **Virus:** programas que se adjuntan a otros archivos y se propagan al ejecutarse, causando daños o robando información.
- **Gusanos:** Similares a los virus, pero pueden propagarse por sí mismos sin necesidad de un archivo huésped.
- **Troyanos:** Programas maliciosos que se disfrazan de software legítimo para engañar a los usuarios y ejecutar actividades dañinas.
- **Spyware:** Software que recopila información sobre un usuario sin su conocimiento, generalmente para actividades de espionaje.
- **Adware:** Programas que muestran anuncios no deseados y pueden recopilar datos del usuario para fines publicitarios.
- **Ransomware:** Malware que cifra los archivos del usuario y exige un rescate para desbloquearlos.
- **Rootkits:** Herramientas diseñadas para ocultar la presencia de otros malwares y mantener el control de un sistema comprometido.

Basado en los peligros que estos tipos de malware plantean a la seguridad de los datos, así como al funcionamiento y la integridad del hardware y el software, es crucial que se tomen medidas de protección adecuadas para asegurarse de que todos los componentes del sistema tengan una capa de seguridad que los resguarde.[12]

4.2.2. Tipos de ataques cibernéticos

Los ataques cibernéticos pueden tomar una variedad de formas, cada una con métodos y objetivos distintos. Algunos de los más comunes son:

- **Phishing:** ataques que utilizan correos electrónicos o sitios web falsos para engañar a los usuarios y obtener información confidencial, como contraseñas y datos financieros[13].

- **Ransomware:** ataques que implican el uso de software malicioso para cifrar los datos de la víctima y exigir un rescate a cambio de la clave de descifrado[13].
- **Ataques Distribuidos de Denegación de Servicio (DDoS):** ataques que buscan sobrecargar un servicio o red enviando un gran volumen de tráfico desde múltiples fuentes, causando su falla[13].
- **Man-in-the-Middle (MitM):** ataques donde el ciberdelincuente intercepta y posiblemente altera la comunicación entre dos partes sin que estas se den cuenta[14].
- **Inyección de SQL:** ataques que explotan vulnerabilidades en aplicaciones web para ejecutar comandos SQL no autorizados y acceder a bases de datos sensibles[15].
- **Cross-Site Scripting (XSS):** ataques que inyectan scripts maliciosos en páginas web vistas por otros usuarios, lo que permite el robo de datos y la ejecución de comandos no autorizados por parte de los atacantes[16].

4.2.3. Categorización de intrusos

La seguridad de las redes se encuentra amenazada por diferentes tipos de intrusos, cada uno con objetivos, habilidades orígenes y métodos diferentes. El conocer de una mejor manera sus motivaciones y técnicas puede significar la formulación de distintas estrategias de defensa más impactantes. Entre los tipos más comunes se encuentran:

- **Script kiddies:** esto son personas con poco conocimiento sobre el hackeo, por lo mismo sus ataques se basan en herramientas y partes de código que fueron creados por otras personas. Al no tener un conocimiento profundo no suelen comprender por completo lo que está sucediendo. Generalmente buscan reconocimiento, entretenimiento o causar interrupciones menores sin un objetivo específico.
- **Hackers éticos:** personas que se dedican a la investigación de seguridad informática que utilizan su talento para descubrir deficiencias en los sistemas y redes. Trabajan de manera legal y ética. Su objetivo es ayudar a las organizaciones a mejorar su seguridad y proteger sus activos digitales. Llevan a cabo pruebas de penetración, auditorías de protección y análisis de vulnerabilidades con aprobación de la organización.
- **Cibercriminales:** estos son aquellos atacantes malintencionados que buscan detectar y utilizar las vulnerabilidades de un software para causar daño o para obtener algún beneficio propio. Además, son personas que roban información confidencial, extorsionar a las víctimas, buscan ganancias financieras o dañan el sistema. Utilizan técnicas como: phishing, ransomware, ingeniería social pero definitivamente sus acciones son ilegales.
- **Hackers de sombrero gris:** estos son personas que tienden estar entre lo legal y lo ilegal. Pueden llegar a sobrepasar algunas normas de ética o violentar la ley, pero sin una mala intención. Sus objetivos principales normalmente destacan por querer demostrar sus habilidades y dominio de un tema, alertar sobre vulnerabilidades pero sin tener los permisos o accesos debidos de los propietarios de un sistema. Las formas más comunes en las que actúan son penetraciones no autorizadas a los sistemas a los que se están enfocando.

Los diversos tipos de intrusos en la seguridad de redes, desde los scripts kiddies hasta los hackers de sombrero gris presentan un espectro amplio de motivaciones y habilidades, lo que resalta la necesidad de comprenderlos a fondo para formular estrategias de defensa más efectivas y adaptadas a cada amenaza específica.[17]

4.2.4. Estrategias de defensa y contramedidas

Para poder proteger las redes contra estos diversos tipos de ataques y los diferentes tipos de intrusos que pueden existir, es importante abordar la mejor estrategia de defensa y las contramedidas más efectivas. Estas estrategias pueden dividirse en varias categorías, cada una abordando diferentes aspectos de la seguridad de la red:

- **Medidas preventivas:**

Es sumamente importante la actualización de todos y cada uno de los sistemas y software con los últimos parches de seguridad para corregir las vulnerabilidades conocidas. Además de eso es recomendable asegurar que las configuraciones de sistemas y redes sean seguras, deshabilitando servicios innecesarios y utilizando contraseñas robustas. [18]

- **Medidas de respuesta:**

Desarrollar y mantener un plan de respuesta a incidentes que incluya procedimientos para la identificación, contención, erradicación y recuperación de ataques; realizar un análisis forense digital para investigar incidentes de seguridad, determinar el alcance de los daños y reunir pruebas para acciones legales; y establecer canales de comunicación claros y efectivos para la coordinación de la respuesta entre los equipos de seguridad y otras partes interesadas.[19]

- **Medidas de detección:**

El utilizar herramientas que constantemente monitoreen el sistema para detectar actividades sospechosas y anomalías en el tráfico de red y luego ayudar a revisar y analizar los registros de eventos y los registros de acceso para identificar patrones de comportamiento inusual e indicios de intrusiones. Así también es posible implementar sistemas de alerta que notifiquen a los dueños del sistema para que tomen las medidas necesarias.[20]

4.3. Sistema de detección de intrusos

Un sistema de detección de intrusos (IDS) es una herramienta de seguridad que se encarga de monitorear el tráfico de la red y los sistemas que componen el grupo de trabajo, para detectar actividades sospechosas o que sobrepasen las políticas de seguridad.[21] Entre las principales categorías de los IDS se encuentran los siguientes:

- **Basado en firmas:**

Este tipo de IDS suelen poseer grandes bases de datos en las que contienen las firmas o patrones informáticos que coinciden con ataques anteriores, estos se utilizan para lograr la detección de los intrusos. La manera en la que funcionan es bastante similar a la de un antivirus comparando el comportamiento a evaluar con los registros que ya poseen en sus datos almacenados. Estos IDS llegan a destacar por su alta precisión en la detección de ataques conocidos y el bajo número de falsos positivos que generan. Pero cabe destacar que son incapaces de reconocer un ataque con un nuevo formato hasta que se actualizan y se agreguen estas nuevas firmas a sus registros.[22]

- **Basado en anomalías:**

Este tipo de IDS suelen monitorear el comportamiento tanto del sistema como el de la red para así poder detectar patrones o comportamientos inusuales que lleguen a significar una intrusión. Normalmente se basan en técnicas estadísticas y heurísticas para determinar los rangos de que cosas se catalogan como normales o que cosas ya deberían de analizarse más a fondo. La mayor ventaja que este tipo de sistemas presentan es la detección de nuevos ataques sin la necesidad

de tener un conocimiento previo de sus características. Sin embargo suelen generar una mayor tasa de falsos positivos y requieren un periodo de aprendizaje para definir el comportamiento normal.[23]

Un IDS típico consta de varios componentes esenciales:

- **Sensores:** son dispositivos o incluso software que captura y monitorea el tráfico de red y los eventos del sistema. Pueden colocarse en diferentes ubicaciones disponibles en la red para proporcionar una cobertura más amplia. Básicamente son los encargados de recopilar y recolectar los datos en tiempo real para su análisis.
- **Motor de análisis:** es el componente central que se ocupa del tratamiento de los datos que fueron recolectados por los sensores. Se basa en utilizar diferentes algoritmos y técnicas para identificar posibles intrusiones. Este se encarga de analizar los datos en busca de comportamientos o patrones extraños que podrían ser inusuales.
- **Bases de datos:** son las encargadas de almacenar las firmas de los ataques conocidos, que el motor de análisis utiliza para hacer la comparación con los datos monitoreados. Su tarea es proporcionar patrones de referencia para la detección basada en firmas.
- **Interfaz:** herramienta que permite a los administradores de seguridad configurar el IDS, monitorear alertas y revisar los informes generados. Su función es facilitar la gestión y respuesta a incidentes de seguridad.
- **Alertas:** son aquellas notificaciones que el sistema haya generado a partir de sus hallazgos. Su función es proveer una respuesta rápida a amenazas detectadas.

Estos componentes trabajan juntos para detectar intrusiones de manera efectiva y más precisa.[24]

4.4. Paquetes de Red

Un paquete de red es la unidad básica de transferencia de datos en una red de computadoras. Se trata de un bloque de información que se envía desde un origen a un destino a través de una red, compuesto por varios campos que contienen tanto los datos útiles como la información necesaria para su correcta transmisión. Los paquetes de red juegan un papel fundamental en la comunicación entre dispositivos conectados a redes, ya que permiten fragmentar los datos en unidades manejables que son transmitidas y luego reensambladas en su destino final.[25]

Los paquetes se componen generalmente de un encabezado (header) y el mensaje a transmitir (payload). El encabezado contiene la información necesaria para gestionar la transmisión, mientras que la carga útil contiene los datos que realmente se están enviando. A continuación se detallan algunas de las características más comunes que suelen aparecer en los paquetes de red.[26]

- **Dirección IP:** Define la dirección del remitente y del destinatario. Esto es crucial para determinar el origen y destino de los datos en la red. Las direcciones IP pueden ser IPv4 (32 bits) o IPv6 (128 bits).
- **Puertos:** Los números de puerto especifican los servicios de origen y destino (como HTTP, FTP o SSH). Esto permite que un dispositivo reciba múltiples conexiones simultáneamente en diferentes puertos.
- **Protocolo:** El protocolo especifica el tipo de transporte que se utiliza, como TCP (Transmission Control Protocol) o UDP (User Datagram Protocol), que define la forma en que se entregarán los paquetes y cómo se manejarán los errores.

- **Tamaño del Paquete:** Los paquetes tienen un tamaño que incluye tanto el encabezado como los datos. El tamaño puede variar según la red y el protocolo. Si un paquete excede el tamaño máximo permitido, se fragmenta en varios paquetes.
- **Número de Secuencia:** Utilizado principalmente en protocolos como TCP, el número de secuencia asegura que los datos se entreguen en el orden correcto y sin duplicados.
- **TTL (Time to Live):** Es un valor que decrece con cada salto que da el paquete en su camino hacia el destino. Si el TTL llega a cero antes de llegar al destino, el paquete se descarta, lo que ayuda a evitar bucles infinitos en la red.
- **Checksum:** Es una suma de verificación que permite verificar la integridad de los datos. Si el checksum del paquete recibido no coincide con el valor calculado, el paquete se considera corrupto y se descarta.

4.4.1. Métodos de Análisis del Tráfico de Red

El análisis de paquetes es un método fundamental para identificar y prevenir intrusiones, especialmente utilizado en IDS. Este método se caracteriza por capturar y analizar el tráfico de red con gran detalle a nivel de paquete, lo que permite identificar patrones y anomalías que podrían indicar un comportamiento malicioso. La captura de paquetes es el primer paso en este proceso, y se puede realizar utilizando varias técnicas y herramientas.[27]

- **Captura de Paquetes:** La captura de paquetes implica interceptar y registrar los paquetes de datos que atraviesan una red. Esto se puede hacer mediante la escucha de puertos en switches o utilizando dispositivos de captura dedicados.
- **Análisis de Paquetes:** Una vez capturados, los paquetes se analizan para extraer información útil, como direcciones IP, puertos, protocolos, y patrones de tráfico. Este análisis puede ser manual o automatizado, utilizando algoritmos de detección de anomalías y firmas de ataque conocidas.

Hay muchas herramientas y programas de software diferentes disponibles para la captura y análisis de paquetes, con sus propias características y capacidades únicas, entre las más destacadas se encuentran:

- **Wireshark:** es una de las herramientas de captura y análisis de paquetes más utilizadas y versátiles, con la capacidad de ver, filtrar y analizar el tráfico de red en tiempo real y aplicaciones que van desde la resolución de problemas de red hasta el análisis de seguridad y su uso en el aula. [28]
- **Tcpdump:** es una herramienta de línea de comandos que se utiliza ampliamente en sistemas Unix y Linux. Tcpdump es un método eficiente para capturar e interpretar paquetes de red, ya que es apto para análisis rápidos y permite realizar capturas en scripts de manera sencilla.[29]
- **PyShark:** es una biblioteca de Python que utiliza las capacidades de Wireshark para la captura y el análisis de paquetes, lo que ayuda a la creación de scripts personalizados y la integración de aplicaciones para el análisis de tráfico.[30]

4.5. Aplicación del Aprendizaje Automático

El uso del aprendizaje automático para la detección de intrusos se ha convertido en una de las herramientas más fuertes en la actualidad. Esto se debe a que con la capacidad de aprender,

adaptarse y reconocer patrones de información pasada. Estos modelos pueden detectar acciones inusuales dentro de la red y así prevenir que se expandan y lleguen a causar potenciales daños. Estas herramientas suelen ser preparadas con antecedentes de diversos ataques cibernéticos, mejorando la eficiencia y efectividad de los sistemas de detección de intrusos (IDS).[31].

4.5.1. Técnicas de aprendizaje automático

Las técnicas de aprendizaje automático juegan un papel crucial en la seguridad de redes, permitiendo la identificación y mitigación de amenazas de manera más eficiente y precisa. A continuación, se presentan dos de las metodologías más utilizadas en este campo, cada una con enfoques y aplicaciones particulares que contribuyen a la detección de actividades maliciosas y la protección de la infraestructura digital.[32]

- **Clasificación Supervisada:** Los modelos se entrenan con datos etiquetados, donde cada una es identificada como benigna o maliciosa. Ejemplos incluyen los Support Vector Machines (SVM) y los árboles de decisión.
- **Clustering No Supervisado:** Los modelos agrupan datos en clusters basados en similitudes, sin ser etiquetas predefinidas. Esta técnica es útil para detectar comportamientos anómalos que no siguen patrones conocidos.

4.5.2. Modelos de machine learning comúnmente utilizados

En la detección de intrusos, se utilizan varios modelos de machine learning, cada uno con sus propias ventajas y aplicaciones específicas:

- **Support Vector Machines (SVM):** Los SVM son modelos de clasificación supervisada que buscan el hiperplano que mejor separa las clases de datos en un espacio multidimensional. Son efectivos para detectar intrusos en conjuntos de datos balanceados y de alta dimensión. Se caracteriza por su alta precisión y capacidad para manejar datos no lineales mediante el uso de kernels. Sin embargo, requieren tiempo y recursos computacionales significativos para el entrenamiento en grandes volúmenes de datos.
- **Árboles de Decisión:** Los árboles de decisión son modelos de clasificación y regresión que utilizan una estructura en forma de árbol para tomar decisiones basadas en el peso de los atributos de los datos. Son útiles para la detección de intrusos debido a su interpretabilidad y facilidad de uso. Su facilidad de interpretación y visualización se muestran como sus puntos fuertes, así como su capacidad para manejar tanto datos numéricos como categóricos. No obstante, pueden ser propensos al sobreajuste si no se podan adecuadamente.
- **Naive Bayes:** Naive Bayes es un modelo de clasificación supervisada basado en el teorema de Bayes, que asume la independencia entre las características de los datos. Es ampliamente utilizado en la detección de intrusos debido a su simplicidad, rapidez y eficiencia, incluso con grandes volúmenes de datos. Este modelo es particularmente efectivo cuando las variables predictoras no están fuertemente correlacionadas. Aunque su principal ventaja radica en su velocidad y bajo costo computacional, su rendimiento puede verse afectado si la suposición de independencia entre variables no se cumple, lo que puede llevar a resultados menos precisos en algunos escenarios.
- **Redes Neuronales:** Las redes neuronales imitan el funcionamiento del cerebro humano utilizando capas de neuronas artificiales para aprender representaciones complejas de los datos. Son extremadamente eficaces en la detección de intrusos debido a su capacidad para identificar

patrones no lineales y complejos. Al igual que con los árboles de decisión se destacan por su alta precisión y capacidad para aprender de grandes volúmenes de datos. Pero estos modelos requieren grandes cantidades de datos y recursos computacionales para el entrenamiento.

Cada uno de los modelos mencionados tiene sus aportes y la elección de que modelo a utilizar depende de las características del conjunto de datos y los objetivos que se tengan.[32]

4.6. Características de Paquetes de Red

Identificar las características de un paquete de red es fundamental para detectar posibles intrusos. Estas características, también conocidas como *features*, permiten a los modelos de aprendizaje automático aprender patrones asociados con comportamientos normales y anómalos. Al analizar un paquete de red, se pueden extraer atributos clave que indican si su origen es legítimo o si, por el contrario, presenta características típicas de un ataque. Algunos de estos atributos incluyen:

- **Marca Temporal (ts):** Permite rastrear cuándo ocurrió la actividad, esencial para detectar patrones de ataque en el tiempo.
- **Dirección IP del origen (id.orig_h):** Ayuda a identificar la fuente de la conexión; IPs desconocidas o no autorizadas pueden ser sospechosas.
- **Puerto del origen (id.orig_p):** Algunos puertos son más propensos a ser atacados; observar el uso inusual de estos puertos puede ser indicativo de intrusión.
- **Estado de la conexión (conn_state):** Los estados anormales, como conexiones que se abren y cierran rápidamente, pueden señalar un comportamiento intruso.
- **Etiquetas (label y detailed_label):** Proveen contexto adicional sobre el tipo de tráfico, lo que ayuda a distinguir entre tráfico benigno y malicioso.

4.7. Conjunto de datos

Un conjunto de datos es una colección organizada de información que normalmente es procesada y analizada por sistemas informáticos. Los conjuntos de datos suelen tener diferentes formas y tamaños, esto depende de la cantidad de información que se haya podido recopilar sobre el tema a tratar.[33]

En el ámbito del análisis de tráfico de red y la detección de intrusos, los conjuntos de datos juegan un papel crucial. Estos datos son fundamentales para que los modelos puedan aprender a identificar patrones, detectar anomalías, y diferenciar entre tráfico normal y el tráfico potencialmente malicioso sobre eventos que ya ocurrieron y dejaron detalles valiosos para identificar con una mejor certeza posibles intrusos.[34]

5.1. Exploración de datos

En esta fase, se llevó a cabo un análisis exhaustivo de los conjuntos de datos del tráfico de red, centrando el enfoque principalmente en el uso de datos reales siempre que fuera factible. El objetivo principal de este análisis fue obtener una comprensión detallada de la naturaleza y las características del tráfico de red en contextos tanto domésticos como empresariales. Cuando los datos reales no estaban disponibles, se utilizaron conjuntos de datos simulados diseñados para replicar escenarios típicos de tráfico, garantizando así una cobertura amplia de posibles situaciones y así una mejor comprensión del tráfico de datos.

El análisis preliminar proporcionó la base para la ingeniería de características y el desarrollo de algoritmos de detección, asegurando que los datos se adaptaran a las características específicas del tráfico observado. Finalmente, tras una evaluación detallada, se concluyó que el conjunto de datos más adecuado para el propósito de la investigación fue el “Malware Detection in Network Traffic Data”. Este dataset, proporcionado por Stratosphere Laboratory, fue seleccionado debido a sus características relevantes y su capacidad para representar tanto tráfico benigno como malicioso en entornos domésticos.[35]

5.2. Ingeniería de características

En la etapa de ingeniería de características, se llevó a cabo una meticulosa transformación de los datos recolectados con el objetivo de maximizar su utilidad en el análisis de detección de intrusos, con el fin de determinar cuál de las características resultaban más relevantes para el objetivo de la detección de malware en el tráfico de red.

El conjunto de datos utilizado contenía las siguientes características iniciales:

- **ts**: Marca temporal del paquete.
- **uid**: Identificador único de la conexión.
- **id.orig_h**: Dirección IP del origen.

- **id.orig_p**: Puerto del origen.
- **id.resp_h**: Dirección IP de destino.
- **id.resp_p**: Puerto de destino.
- **proto**: Protocolo utilizado (por ejemplo, TCP, UDP).
- **service**: Servicio al que se está conectando (por ejemplo, HTTP, DNS).
- **duration**: Duración de la conexión.
- **orig_bytes**: Bytes enviados desde el origen.
- **resp_bytes**: Bytes recibidos en el destino.
- **conn_state**: Estado de la conexión (por ejemplo, ESTABLISHED, CLOSED).
- **local_orig**: Indica si la conexión es local al origen.
- **local_resp**: Indica si la conexión es local al destino.
- **missed_bytes**: Bytes que se perdieron durante la conexión.
- **history**: Historial de la conexión (por ejemplo, conexión previa, reinicio).
- **orig_pkts**: Paquetes enviados desde el origen.
- **orig_ip_bytes**: Bytes IP enviados desde el origen.
- **resp_pkts**: Paquetes recibidos en el destino.
- **resp_ip_bytes**: Bytes IP recibidos en el destino.
- **tunnel_parents**: Identificadores de túneles asociados.
- **label**: Etiqueta general (por ejemplo, benigno, malicioso).
- **detailed_label**: Etiqueta detallada (por ejemplo, tipo específico de ataque).

Se realizó una exploración visual de los datos para identificar la cantidad de valores nulos presentes en cada columna, utilizando un gráfico de barras para representar visualmente la distribución de estos valores.

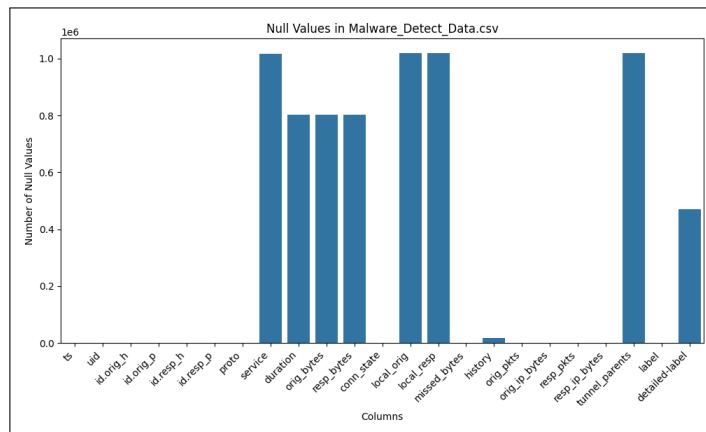


Figura 5.1: Porcentaje de valores nulos por característica en el conjunto de datos inicial.

Como se puede apreciar en la Figura 5.1 se presentan distintas características con un valor alto de datos faltantes. Esta exploración nos permitió identificar y eliminar esas características que contenían una cantidad significativa de valores nulos, ya que su presencia afectaría negativamente el rendimiento del modelo de detección.

Posteriormente, se efectuó una selección de características a partir de las variables restantes del dataset, conservando aquellas que ofrecían más información relevante. En este proceso, se generaron matrices de correlación para analizar las relaciones entre las variables.

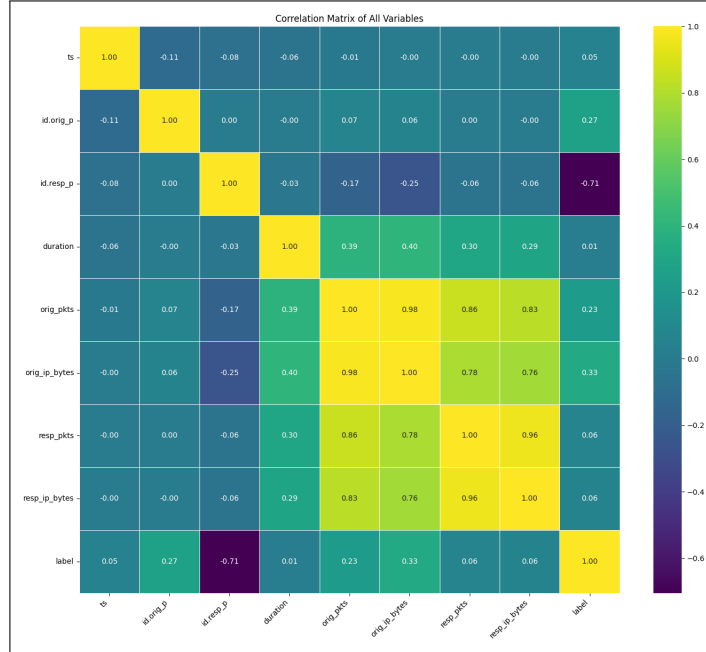


Figura 5.2: Matriz de correlación de características del conjunto de datos.

Como se muestra en la Figura 5.2, esta matriz nos permitió identificar cuáles variables eran más influyentes en la detección de intrusos. El valor mínimo para considerar una variable importante fue de 0.05 debido a que este umbral nos permite filtrar aquellas variables con una correlación muy baja o despreciable. Este límite fue seleccionado porque, en el contexto de machine learning, un coeficiente de correlación menor a 0.05 indica una relación muy débil entre la variable y el objetivo de estudio.

Además, se transformaron las variables restantes para mejorar su representación y relevancia en el análisis. Así mismo, la normalización de los datos se llevó a cabo para asegurar consistencia y preparar los datos para el análisis con algoritmos de aprendizaje automático. Luego del análisis y preprocesamiento, se redujo el conjunto de datos a las características más relevantes para la detección de malware.

5.2.1. Selección y Eliminación de Características

Las características iniciales del conjunto de datos fueron diseñadas para capturar diferentes dimensiones de cada conexión. El análisis exploratorio de datos permitió evaluar la relevancia de cada característica en el contexto de la detección de intrusos, considerando las correlaciones con patrones de comportamiento malicioso. En esta sección, se abordarán los fundamentos de la selección y eliminación de ciertas características en el conjunto de datos utilizado para la detección de malware.

en el tráfico de red.

5.2.2. Criterios para la Eliminación de Características

Redundancia Se eliminaron ciertas características como `local_orig` y `local_resp` debido a su naturaleza redundante. La información contenida en estas características podía deducirse fácilmente de las direcciones IP, lo que permitió reducir el ruido en los datos y mejorar la eficiencia computacional del modelo.

Valores Nulos Se descartaron características que presentaban una alta proporción de valores nulos, como `service`, `orig_bytes`, `resp_bytes`, `local_orig`, `local_resp`, `tunnel_parents` y `missed_bytes`. La gran cantidad de datos faltantes en estas columnas las hacía menos útiles para el análisis, y su eliminación ayudó a simplificar el conjunto de datos y a evitar problemas en el entrenamiento del modelo, mejorando así la eficiencia y precisión de este.

Falta de Relevancia Directa Características como `history` y `tunnel_parents` fueron consideradas menos relevantes para la detección inicial de intrusos. Aunque en el conjunto de datos utilizado se pudo obtener esta información con facilidad debido a que se trataba de un ambiente controlado, en situaciones reales y no controladas, la obtención de estos datos es considerablemente más compleja. Por esta razón, se decidió enfocarse en un conjunto más simple pero significativo de características, priorizando aquellas que sean más accesibles y consistentes en entornos de producción.

Reducción de la Dimensionalidad en base a correlación La priorización de un conjunto de datos más sencillo facilita el entrenamiento de modelos de aprendizaje automático. La reducción de dimensionalidad mejora la precisión del modelo al eliminar información irrelevante o redundante. En este proceso, se evaluaron las correlaciones y se eliminaron características que mostraron baja correlación con la variable objetivo, contribuyendo a un modelo más robusto.

5.2.3. Priorización de Información Esencial para Detección de Anomalías

Se decidió mantener características clave que proporcionan información fundamental sobre las conexiones de red, lo que permite identificar patrones anómalos y posibles intrusiones. Características como las direcciones IP, puertos y estados de conexión son cruciales para la identificación y monitoreo del tráfico. Además, se priorizaron las características relacionadas con los volúmenes de tráfico, ya que estas ofrecen información detallada sobre la cantidad de datos transferidos, facilitando así la detección de comportamientos sospechosos. Se optó por simplificar la representación de protocolos, eliminando la columna `proto` y, en su lugar, creando las columnas `tcp` y `udp` para indicar de manera binaria la presencia de estos protocolos. Esto permite al modelo enfocarse en las características más relevantes sin perder información crítica, optimizando así la detección de anomalías en el tráfico de red.

| Campo | Descripción |
|----------------------|--|
| ts | Marca temporal del paquete. |
| uid | Identificador único de la conexión. |
| id.orig_h | Dirección IP del origen. |
| id.orig_p | Puerto del origen. |
| id.resp_h | Dirección IP de destino. |
| id.resp_p | Puerto de destino. |
| conn_state | Estado de la conexión. |
| orig_pkts | Paquetes enviados desde el origen. |
| orig_ip_bytes | Bytes IP enviados desde el origen. |
| resp_pkts | Paquetes recibidos en el destino. |
| resp_ip_bytes | Bytes IP recibidos en el destino. |
| label | Etiqueta general. |
| tcp | Indica si el protocolo de transporte es TCP. |
| udp | Indica si el protocolo de transporte es UDP. |

Tabla 5.1: Características finales.

La Tabla 5.1 presenta las características finales seleccionadas luego de haber realizado el análisis del conjunto inicial de datos. Estas características fueron elegidas por su relevancia en la identificación de patrones de tráfico de red y posibles intrusos.

- **ts**: Proporciona la marca temporal del paquete, esencial para analizar el momento exacto de la conexión y detectar patrones de tiempo relacionados con ataques.
- **uid**: Identificador único de la conexión, utilizado para rastrear y correlacionar eventos a través de sesiones de red.
- **id.orig_h**: Dirección IP del origen, fundamental para identificar la fuente de la conexión y rastrear posibles intrusos o dispositivos comprometidos.
- **id.orig_p**: Puerto del origen, que ayuda a determinar el servicio o aplicación que inicia la conexión, lo cual es útil para identificar actividades inusuales.
- **id.resp_h**: Dirección IP de destino, necesaria para entender hacia dónde se dirigen las conexiones y detectar posibles destinos maliciosos o inusuales.
- **id.resp_p**: Puerto de destino, que proporciona información sobre los servicios a los que se accede, permitiendo detectar intentos de acceder a puertos específicos o vulnerables.
- **conn_state**: Estado de la conexión, utilizado para evaluar el estado de cada sesión de red y detectar conexiones que no se comportan de manera esperada.
- **orig_pkts**: Cantidad de paquetes enviados desde el origen, indicador clave de actividad de red que permite detectar picos anómalos en el tráfico.
- **orig_ip_bytes**: Bytes IP enviados desde el origen, proporciona información sobre el volumen de datos transmitidos, útil para identificar comportamientos sospechosos.
- **resp_pkts**: Paquetes recibidos en el destino, permite monitorear la cantidad de respuestas y posibles patrones de comunicación no habituales.
- **resp_ip_bytes**: Bytes IP recibidos en el destino, otra métrica crucial para analizar el flujo de datos y detectar anomalías en la transmisión.
- **label**: Etiqueta que clasifica la conexión, indispensable para evaluar la precisión del modelo al clasificar y detectar conexiones maliciosas.

- **tcp:** Columna binaria que indica si el protocolo utilizado es TCP, proporcionando información crítica sobre el tipo de transporte y facilitando el análisis de conexiones sospechosas.
- **udp:** Columna binaria que indica si el protocolo utilizado es UDP, relevante para identificar tráfico que utiliza este protocolo y evaluar su comportamiento.

Al mantener estas características finales, se garantiza que el modelo se enfoque en un conjunto de datos optimizado que permite la detección eficiente de anomalías sin sobrecargar el procesamiento. Este enfoque asegura un equilibrio entre la precisión y la eficiencia computacional, facilitando la implementación de un sistema de detección de intrusos efectivo en entornos de red realistas y de alto tráfico.

5.2.4. Estadísticas Descriptivas del conjunto de datos

El dataset final consta de un total de 218,887 registros, que representan los paquetes de red recopilados y procesados para la detección de intrusos. Este tamaño de muestra es adecuado para entrenar y evaluar los modelos de aprendizaje automático, permitiendo una evaluación robusta de su rendimiento. Al dividir el dataset en conjuntos de entrenamiento y prueba. El 80 % de los datos se destina para el entrenamiento del modelo y el 20 % para su evaluación.

La distribución en el conjunto de entrenamiento es la siguiente:

- **Tráfico Benigno (0): 75,132** registros
- **Tráfico Malicioso (1): 99,977** registros

Esta distribución indica un balance entre las muestras de tráfico benigno y malicioso, lo cual es crucial para entrenar un modelo que pueda distinguir efectivamente entre ambos tipos de tráfico.

5.3. Desarrollo módulo de captura de paquetes

Se desarrolló un módulo específico para capturar paquetes de red, diseñado para interceptar y guardar el tráfico entrante y saliente de la computadora utilizada para realizar las tareas profesionales. Este módulo fue esencial para la recopilación de datos en tiempo real y proporcionó una base sólida para el análisis de detección de intrusos. Este componente juega un rol fundamental al capturar y monitorear el tráfico entrante y saliente en la red del usuario. La información recopilada sirve como base para el análisis de detección de intrusos.

Para el desarrollo de este módulo, se seleccionaron las siguientes herramientas y tecnologías, cada una justificada por su funcionalidad y eficiencia en el contexto del proyecto:

- **Pyshark:** Esta biblioteca de Python proporcionó una interfaz para trabajar con Wireshark en Python, permitiendo la captura y análisis de paquetes en tiempo real. Su uso fue justificado por su capacidad de acceder a detalles profundos de los paquetes de red y su integración directa con Python, lo que facilitó el procesamiento automatizado.[30]
- **CSV:** El uso del formato CSV para almacenar las características extraídas de los paquetes permitió una fácil manipulación y análisis de datos en hojas de cálculo y otras herramientas analíticas. Este formato es ampliamente soportado y facilitó la exportación y visualización de los resultados.[36]

- **Ippaddress:** La biblioteca estándar de Python para manipulación y validación de direcciones IP se utilizó para determinar si las IPs involucradas en las conexiones eran locales o externas, un factor importante para identificar tráfico potencialmente sospechoso.[37]
- **Datetime:** Se utilizó para manejar marcas de tiempo y calcular la duración de las conexiones, proporcionando métricas temporales clave para el análisis de patrones de tráfico.[38]
- **Collections (Defaultdict):** Permite organizar y manipular los datos de manera eficiente, facilitando el seguimiento y conteo de paquetes y bytes transmitidos, así como el cálculo de estadísticas sobre las conexiones.[39]

A continuación se muestra un ejemplo de los datos capturados durante una de las pruebas del primer módulo de la herramienta, que incluyen las características de la Tabla 5.1:

Tabla 5.2: Ejemplo de Captura de Paquetes

| ts | uid | id.orig_h | id.orig_p | id.resp_h | id.resp_p | conn_state | orig_pkts | orig_ip_bytes | resp_pkts | resp_ip_bytes | tcp | udp |
|----------------------------|-----|---------------|-----------|---------------|-----------|------------|-----------|---------------|-----------|---------------|------|-------|
| 2024-09-21 17:28:55.378297 | 1 | 140.82.113.22 | 443 | 192.168.1.15 | 58738 | SO | 1 | 516 | 1 | 54 | True | False |
| 2024-09-21 17:28:55.424380 | 2 | 192.168.1.15 | 58749 | 51.11.192.48 | 443 | SO | 5 | 6031 | 1 | 66 | True | False |
| 2024-09-21 17:28:55.424475 | 3 | 192.168.1.15 | 58738 | 140.82.113.22 | 443 | SO | 1 | 54 | 0 | 0 | True | False |
| 2024-09-21 17:28:55.569004 | 4 | 51.11.192.48 | 443 | 192.168.1.15 | 58749 | SO | 1 | 66 | 4 | 5976 | True | False |
| 2024-09-21 17:28:55.876782 | 5 | 13.89.178.27 | 443 | 192.168.1.15 | 57950 | SO | 1 | 154 | 1 | 54 | True | False |
| 2024-09-21 17:28:55.917641 | 6 | 192.168.1.15 | 57950 | 13.89.178.27 | 443 | SO | 1 | 54 | 0 | 0 | True | False |
| 2024-09-21 17:28:55.993455 | 7 | 192.168.1.15 | 58749 | 51.11.192.48 | 443 | SO | 5 | 6031 | 1 | 66 | True | False |
| 2024-09-21 17:28:55.993455 | 8 | 192.168.1.15 | 58749 | 51.11.192.48 | 443 | SO | 5 | 6031 | 1 | 66 | True | False |
| 2024-09-21 17:28:55.993455 | 9 | 192.168.1.15 | 58749 | 51.11.192.48 | 443 | SO | 5 | 6031 | 1 | 66 | True | False |
| 2024-09-21 17:28:55.993455 | 10 | 192.168.1.15 | 58749 | 51.11.192.48 | 443 | SO | 5 | 6031 | 1 | 66 | True | False |

La Tabla 5.2 muestra un conjunto de 10 registros de paquetes, donde cada fila corresponde a una conexión de red específica. En cada registro, se incluyen datos como la marca temporal (**ts**), identificadores únicos de conexión (**uid**), direcciones IP y puertos de origen y destino, así como el estado de la conexión y la cantidad de paquetes e información transferida. Además, se indica si el protocolo de transporte es TCP o UDP, proporcionando una visión detallada y estructurada de cada evento capturado en la red, facilitando así un análisis exhaustivo y preciso de las actividades de red.

5.3.1. Configuración de la Captura

La captura de paquetes se realizó en la interfaz de red Wi-Fi con un total de **10** paquetes por ejecución del script, capturando las características que se muestran en la Tabla 5.1. Este número fue seleccionado para equilibrar la eficiencia y precisión en la captura de datos. Capturar un número menor de paquetes podría resultar en una muestra insuficiente para identificar patrones relevantes, mientras que un número mayor podría sobrecargar el procesamiento de datos en cada iteración. Por lo tanto, ajustar a 10 paquetes por iteración permitió obtener una muestra representativa sin comprometer la velocidad de ejecución ni sobrecargar el sistema.

Este script fue diseñado para extraer información detallada de cada paquete. Esto permitió una recopilación precisa y estructurada de datos relevantes para el análisis del tráfico red. Estos datos serían la entrada al segundo módulo de la herramienta.

5.4. Desarrollo módulo para la detección de intrusos

En esta etapa, se desarrollaron y ajustaron dos modelos de aprendizaje automático específicamente diseñados para identificar intrusos en el tráfico de red: Support Vector Machine (SVM) y Naive Bayes. Ambos modelos fueron elegidos por su capacidad para manejar datos con alta dimensionalidad y su eficacia en tareas de clasificación binaria.

Los modelos se entrenaron utilizando los conjuntos de datos de entrenamiento previamente preparados. Estos modelos fueron desarrollados en el lenguaje de programación Python y se implementaron utilizando librerías especializadas como Scikit-learn y TensorFlow. Durante el proceso de desarrollo.

Tras un análisis exhaustivo de los resultados obtenidos, se decidió seleccionar uno de los dos modelos para formar parte del proceso final. Este modelo fue elegido por su capacidad superior para detectar intrusiones de manera efectiva en el tráfico de red, asegurando así una mayor precisión y fiabilidad en el sistema.

Este fue catalogado como el segundo módulo del proyecto, que tuvo como función principal recibir y analizar los paquetes obtenidos de la red en tiempo real. Los paquetes fueron evaluados por el modelo entrenado para determinar si su origen estaba asociado a comportamientos sospechosos o intrusos.

5.4.1. Support Vector Machine

Este modelo se basa en la teoría de márgenes máximos, buscando encontrar el hiperplano óptimo que separa las clases en el espacio de características. La elección del núcleo (kernel) es fundamental para su rendimiento. En esta implementación, se utilizó el kernel RBF (Radial Basis Function), que permite modelar relaciones no lineales entre las características del tráfico de red. Los parámetros a ajustar incluyen:

- **C:** Controla el trade-off entre un margen más amplio y la clasificación correcta de los puntos de datos. Se utilizó un valor de $C=0.001$.
- **Gamma:** Influye en la forma del kernel, afectando la complejidad del modelo. En este caso, se utilizó la configuración 'scale', que ajusta el valor de gamma en función de las características del dataset.

5.4.2. Naive Bayes

Este modelo se basa en el teorema de Bayes y asume que las características son independientes entre sí, lo que simplifica el cálculo de probabilidades. Existen diferentes variantes, como Gaussian Naive Bayes y Multinomial Naive Bayes, elegidas según la distribución de los datos. En esta implementación, se utilizó la variante Gaussian. Las configuraciones específicas incluyeron:

- **Variante de distribución:** Se eligió Gaussian Naive Bayes, adecuado para datos continuos y normalmente distribuidos.
- **Ajuste de Laplace:** Aunque no se aplicó en este caso, este ajuste se utiliza comúnmente para evitar problemas de cero probabilidades en casos de características no observadas, si se usara una variante como Bernoulli.

5.5. Pruebas de algoritmos

Luego de haber desarrollado varias versiones los algoritmos de detección, variando las configuraciones anteriormente mencionadas y las features utilizadas. Se evaluó su rendimiento utilizando métricas clave como accuracy, precision, recall, F1-score y la curva ROC. Estas métricas permiten determinar la efectividad de los modelos en distinguir entre tráfico legítimo y malicioso.

Las pruebas se llevaron a cabo utilizando las librerías `scikit-learn`, las mismas librerías con las que se implementaron los modelos. Estas evaluaciones se centraron en determinar la capacidad de los modelos para distinguir entre tráfico benigno (label 0) y malicioso (label 1). Se implementaron los siguientes procedimientos con el conjunto de datos determinado para las pruebas:

- **Accuracy:** Para calcular la precisión general de ambos modelos, se utilizó la función `accuracy_score`. Esta métrica mide el porcentaje de clasificaciones correctas en relación con el total de predicciones realizadas, proporcionando una visión general de la efectividad del modelo.
- **Reporte de clasificación:** Se generó un informe detallado utilizando `classification_report`, que incluye métricas críticas como precisión, recall (sensibilidad) y F1-score. Estas métricas evaluaron la capacidad de cada modelo para identificar correctamente las clases de interés, resaltando su rendimiento en términos de verdaderos positivos y negativos.
- **Matriz de confusión:** Se utilizó esta herramienta para visualizar y analizar los resultados de la clasificación. La matriz de confusión mostró los verdaderos positivos (TP), verdaderos negativos (TN), falsos positivos (FP) y falsos negativos (FN) para ambos modelos. Esta visualización facilitó el cálculo de los errores Tipo I (falsos positivos) y Tipo II (falsos negativos), proporcionando una visión clara de los errores de clasificación.
- **Curva ROC y AUC:** Se evaluó la tasa de verdaderos positivos frente a la tasa de falsos positivos mediante el uso de `roc_curve` para obtener la curva ROC. Se utilizó `roc_auc_score` para calcular el área bajo la curva (AUC), lo que permitió medir la capacidad de ambos modelos para diferenciar entre tráfico legítimo y malicioso en diferentes umbrales de clasificación.

Como resultado de estas pruebas, se seleccionó uno de los dos modelos evaluados para ser implementado como el modelo definitivo para el sistema. Este modelo fue el que se integró en el segundo módulo, asegurando así una detección efectiva del tráfico malicioso y optimizando el rendimiento del sistema en su conjunto.

5.6. Integración de los módulos anteriores

Finalmente, todos los módulos desarrollados anteriormente se integraron en un sistema unificado de detección de intrusos. Este proceso incluyó la verificación de la compatibilidad entre los distintos componentes, así como la realización de pruebas de integración para asegurar que el sistema funcionara de manera cohesiva y eficaz.

A continuación, se presenta el flujo general de la herramienta.

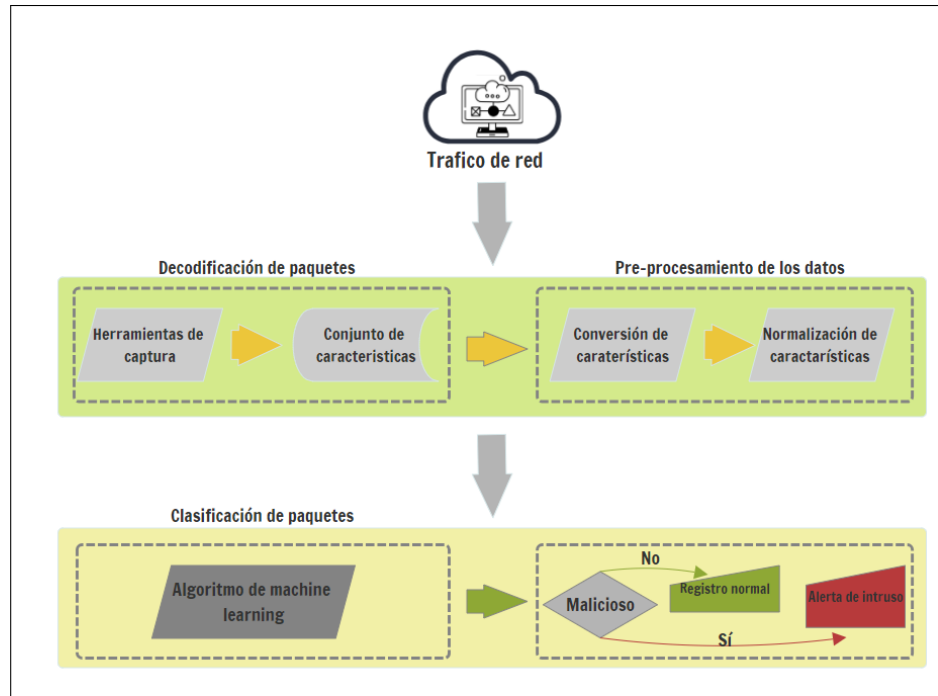


Figura 5.3: Flujo de trabajo de la herramienta de detección de intrusos.

Como se puede observar en la Figura 5.3, la herramienta se estructura en dos módulos principales para el análisis y clasificación del tráfico de red. El primer módulo abarca la decodificación de paquetes y el pre-procesamiento de los datos. En esta fase, el tráfico de red capturado se procesa mediante herramientas que extraen un conjunto de características representativas. Estas características luego se convierten y normalizan para garantizar que estén en un formato estandarizado adecuado para el análisis posterior. El segundo módulo corresponde a la clasificación de paquetes, donde un algoritmo de machine learning utiliza las características normalizadas para identificar si el tráfico es malicioso o no. Si se detecta actividad maliciosa, se genera una alerta de intruso. De lo contrario, se clasifica el tráfico como registro normal.

6.1. Resultados de los Modelos de Aprendizaje Automático

Estos fueron los resultados obtenidos de las versiones con las mejores métricas que se realizaron a los modelos Support Vector Machine (SVM) y Naive Bayes tras utilizar las librerías de `scikit-learn`.

6.1.1. Modelo Support Vector Machine

Métricas utilizadas

El modelo Support Vector Machine obtuvo las siguientes métricas tras ser evaluado:

Tabla 6.1: Métricas del Modelo Support Vector Machine

| Métrica | Valor |
|-----------------------|-------|
| Precisión (Benigno) | 0.89 |
| Recall (Benigno) | 0.69 |
| F1-Score (Benigno) | 0.78 |
| Precisión (Malicioso) | 0.79 |
| Recall (Malicioso) | 0.94 |
| F1-Score (Malicioso) | 0.86 |
| Accuracy | 0.83 |
| Error Tipo I | 0.31 |
| Error Tipo II | 0.06 |

La Tabla 6.1 muestra las métricas obtenidas por el modelo Support Vector Machine, incluyendo precisión, recall, F1-Score, y accuracy para las clases de tráfico benigno y malicioso, así como los errores Tipo I y Tipo II.

Matriz de Confusión

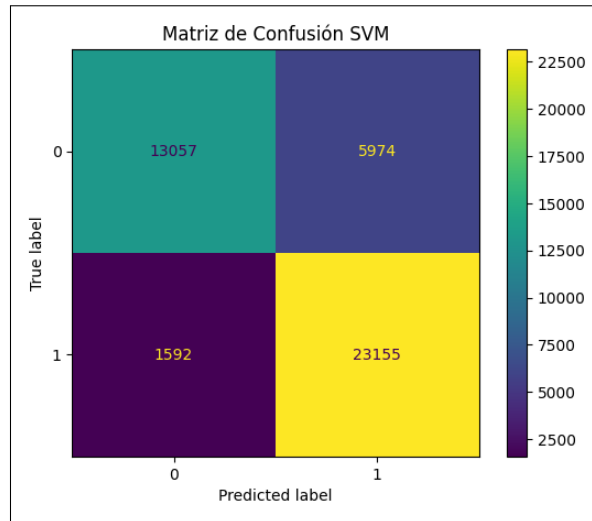


Figura 6.1: Matriz de Confusión del Modelo Support Vector Machine.

La Figura 6.1 muestra la matriz de confusión del modelo Support Vector Machine, representando la distribución de predicciones correctas e incorrectas para las clases de tráfico benigno y malicioso. En esta matriz se observan 13,057 verdaderos positivos (tráfico malicioso correctamente clasificado) y 23,155 verdaderos negativos (tráfico benigno correctamente clasificado). También se identifican 1,592 falsos negativos, donde instancias de tráfico malicioso fueron clasificadas incorrectamente como benignas. Además, hay 5,974 falsos positivos, donde tráfico benigno fue clasificado erróneamente como malicioso.

Curva ROC

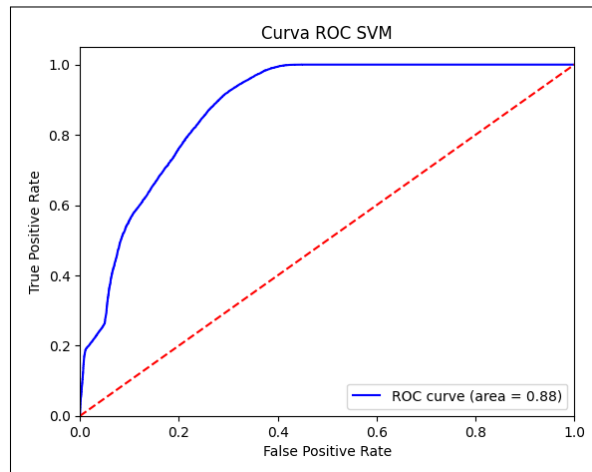


Figura 6.2: Curva ROC del Modelo Support Vector Machine.

La Figura 6.2 presenta la curva ROC (Receiver Operating Characteristic) del modelo Support Vector Machine, donde se evalúa el rendimiento del modelo en términos de la tasa de verdaderos

positivos frente a la tasa de falsos positivos. Esta curva presenta un AUC (Área Bajo la Curva) de 0.88. Inicialmente, la curva asciende de manera vertical, y a medida que avanza, se observa un aumento más moderado en la tasa de verdaderos positivos (TPR) frente a la tasa de falsos positivos (FPR). En un punto específico, la curva alcanza un TPR de 0.28 con un FPR de 0.07. Posteriormente, la curva se vuelve más horizontal, especialmente después de un FPR de 0.4.

6.1.2. Modelo Naive Bayes

Métricas utilizadas

El modelo Naive Bayes obtuvo las siguientes métricas tras ser evaluado:

Tabla 6.2: Métricas del Modelo Naive Bayes

| Métrica | Valor |
|-----------------------|-------|
| Precisión (Benigno) | 0.70 |
| Recall (Benigno) | 0.94 |
| F1-Score (Benigno) | 0.80 |
| Precisión (Malicioso) | 0.94 |
| Recall (Malicioso) | 0.69 |
| F1-Score (Malicioso) | 0.79 |
| Accuracy | 0.80 |
| Error Tipo I | 0.06 |
| Error Tipo II | 0.31 |

La Tabla 6.2 presenta las métricas correspondientes al modelo Naive Bayes, con los mismos indicadores de desempeño para ambas clases y las tasas de error.

Matriz de Confusión

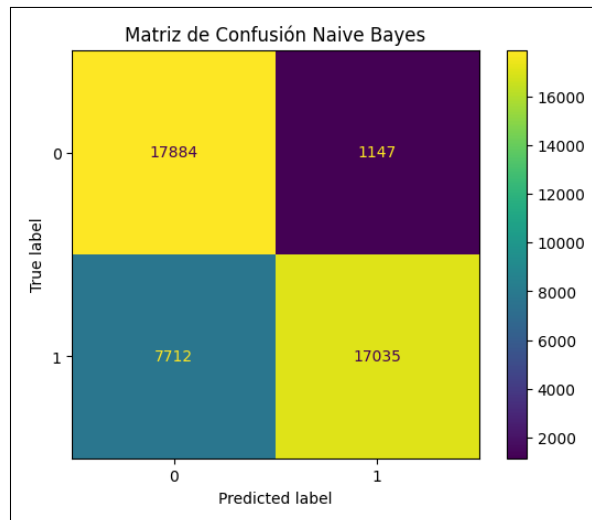


Figura 6.3: Matriz de Confusión del Modelo Naive Bayes.

La Figura 6.3 muestra la matriz de confusión del modelo Naive Bayes, indicando el número de predicciones correctas e incorrectas para ambas clases, benigno y malicioso. En esta matriz se observan 17,884 verdaderos positivos, 17,035 verdaderos negativos y 7,712 falsos negativos, donde instancias de tráfico malicioso fueron clasificadas incorrectamente como benignas. Además, hay 1,147 falsos positivos, donde tráfico benigno fue clasificado erróneamente como malicioso.

Curva ROC

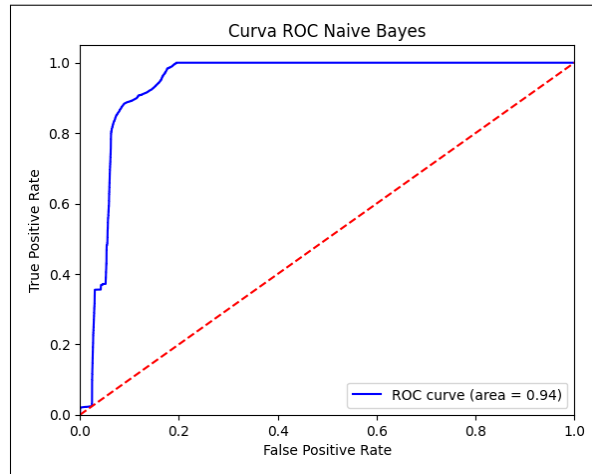


Figura 6.4: Curva ROC del Modelo Naive Bayes.

La Figura 6.4 presenta la curva ROC del modelo Naive Bayes, ilustrando su capacidad para discriminar entre las clases benigno y malicioso según la tasa de verdaderos y falsos positivos. La curva tiene un AUC de 0.94 y muestra subidas muy verticales en ciertas secciones. Esto se traduce en una curva que se aproxima al vértice superior izquierdo. A partir de un FPR de 0.2, la curva comienza a estabilizarse en una línea horizontal.

Resumen de Resultados

Tabla 6.3: Resumen de Resultados

| Métrica | Support Vector Machine | Naive Bayes |
|-----------------------|------------------------|-------------|
| Accuracy | 0.8271 | 0.7976 |
| Macro Avg Precision | 0.84 | 0.82 |
| Weighted Avg F1-Score | 0.84 | 0.83 |

La Tabla 6.3 presenta un resumen de las métricas de desempeño comparativas entre los modelos Support Vector Machine y Naive Bayes. Para el modelo SVM, el valor de accuracy es de 0.8271, mientras que la precisión promedio macro se sitúa en 0.84, al igual que el valor del F1-Score ponderado. Mientras que para el modelo Naive Bayes, los valores son de 0.7976 para el accuracy, 0.82 para la precisión promedio macro y 0.83 para el F1-Score ponderado.

6.2. Resultados de la Implementación Híbrida

Estos fueron los resultados obtenidos al realizar una implementación híbrida de los modelos Support Vector Machine y Naive Bayes, como parte de profundizar en el desempeño final de la herramienta.

6.2.1. Implementacion híbrida 1, SVM seguido de Naive Bayes

Métricas utilizadas

La primera implementación obtuvo las siguientes métricas tras ser evaluado:

Tabla 6.4: Métricas de la implementacion híbrida 1

| Métrica | Valor |
|-----------------------|-------|
| Precisión (Benigno) | 0.93 |
| Recall (Benigno) | 0.84 |
| F1-Score (Benigno) | 0.88 |
| Precisión (Malicioso) | 0.86 |
| Recall (Malicioso) | 0.94 |
| F1-Score (Malicioso) | 0.90 |
| Accuracy | 0.89 |
| Error Tipo I | 0.16 |
| Error Tipo II | 0.06 |

La Tabla 6.4 muestra las métricas obtenidas por la implementación híbrida 1, incluyendo precisión, recall, F1-Score, y accuracy para las clases de tráfico benigno y malicioso, así como los errores Tipo I y Tipo II.

Matriz de Confusión

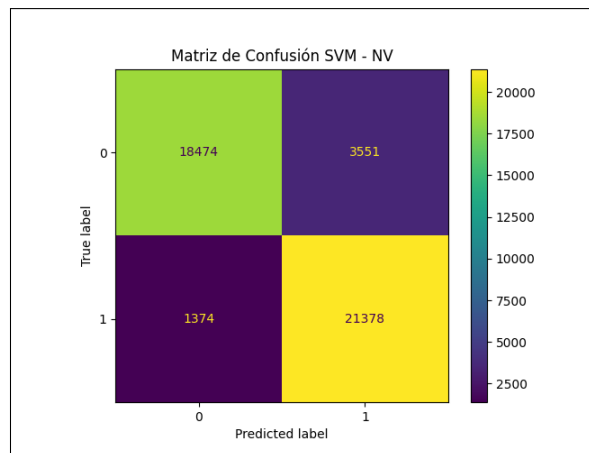


Figura 6.5: Matriz de Confusión de la implementación híbrida 1.

La Figura 6.5 muestra la matriz de confusión de la implemetación híbrida 1, representando la distribución de predicciones correctas e incorrectas para las clases de tráfico benigno y malicioso. En esta matriz se observan 18,474 verdaderos positivos (tráfico malicioso correctamente clasificado) y 21,378 verdaderos negativos (tráfico benigno correctamente clasificado). También se identifican 1,374 falsos negativos, donde instancias de tráfico malicioso fueron clasificadas incorrectamente como benignas. Además, hay 3,551 falsos positivos, donde tráfico benigno fue clasificado erróneamente como malicioso.

6.2.2. Implementacion híbrida 2, Naive Bayes seguido de SVM

Métricas utilizadas

La implemetación híbrida 2 obtuvo las siguientes métricas tras ser evaluado:

Tabla 6.5: Métricas de la implemetación hóibrida 2

| Métrica | Valor |
|-----------------------|-------|
| Precisión (Benigno) | 0.77 |
| Recall (Benigno) | 0.95 |
| F1-Score (Benigno) | 0.85 |
| Precisión (Malicioso) | 0.95 |
| Recall (Malicioso) | 0.76 |
| F1-Score (Malicioso) | 0.84 |
| Accuracy | 0.84 |
| Error Tipo I | 0.05 |
| Error Tipo II | 0.24 |

La Tabla 6.5 muestra las métricas obtenidas por la implementación híbrida 2, incluyendo precisión, recall, F1-Score, y accuracy para las clases de tráfico benigno y malicioso, así como los errores Tipo I y Tipo II.

Matriz de Confusión

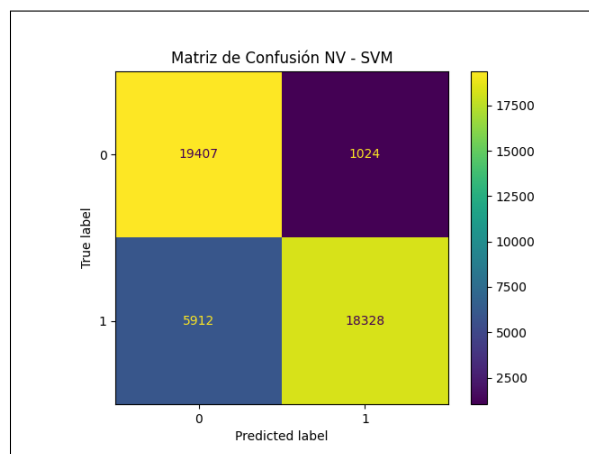


Figura 6.6: Matriz de Confusión de la implementación híbrida 2.

La Figura 6.6 muestra la matriz de confusión de la implementación híbrida 2, representando la distribución de predicciones correctas e incorrectas para las clases de tráfico benigno y malicioso. En esta matriz se observan 19,407 verdaderos positivos (tráfico malicioso correctamente clasificado) y 18,328 verdaderos negativos (tráfico benigno correctamente clasificado). También se identifican 5,912 falsos negativos, donde instancias de tráfico malicioso fueron clasificadas incorrectamente como benignas. Además, hay 1,024 falsos positivos, donde tráfico benigno fue clasificado erróneamente como malicioso.

7.1. Desempeño de los Modelos

En este apartado, se analiza el rendimiento de las mejores versiones los modelos implementados para la detección de malware en tráfico de red, los cuales fueron Support Vector Machine (SVM) y Naive Bayes. Cabe mencionar que las versiones de los modelos que no fueron incluidas en el análisis final presentaron resultados que carecían de la efectividad necesaria para la detección de intrusiones, estas versiones abarcaron enfoques como:

- En la primera versión de los modelos, tanto Support Vector Machine como Naive Bayes se implementaron utilizando todas las features disponibles sin realizar un análisis previo de su relevancia. Para los dos modelos las configuraciones fueron las descritas en la metodología. Ambos modelos presentaron un rendimiento deficiente, con un accuracy de 0.54 para SVM y 0.49 para Naive Bayes. Estos resultados iniciales indicaron que la inclusión indiscriminada de características no era efectiva para la detección de intrusiones, lo que llevó a la necesidad de ajustar los modelos.
- En la segunda versión, se realizaron cambios significativos en los parámetros de ambos modelos, así como un análisis detallado de las features seleccionadas. Para el modelo SVM, el parámetro $C = 0.001$ fue ajustado para controlar el margen de error permitido. Un valor pequeño de C permite que el modelo sea más tolerante a los errores en el entrenamiento, buscando un margen más amplio entre las clases. Además, se utilizó la configuración de Gamma como "*scale*", lo que ajusta automáticamente la influencia de cada dato según el número de features, permitiendo un mejor manejo de la dispersión de los datos. En el caso de Naive Bayes, se continuó utilizando la variante Gaussian, que asume que los datos siguen una distribución normal. Sin embargo, se empezó a normalizar los datos para ajustar las escalas de las variables y mejorar la precisión del modelo, ya que este algoritmo es sensible a las diferencias de escala entre las features. A pesar de estas mejoras en la configuración de los modelos, el SVM alcanzó un accuracy de 0.69, mientras que Naive Bayes logró un accuracy de 0.58. Aunque ambos modelos mostraron mejoras respecto a la primera versión, estos resultados seguían siendo insuficientes para garantizar una detección confiable de intrusiones en el sistema.
- En la tercera versión, se aplicaron las features finales tras un análisis exhaustivo de correlación. Se revisaron y ajustaron nuevamente los parámetros de ambos modelos para que quedaran

configurados como los descritos en la metodología, buscando optimizar su rendimiento. Sin embargo, los resultados no fueron satisfactorios: el modelo SVM logró un accuracy de 0.79, mientras que Naive Bayes alcanzó un accuracy de 0.76. A pesar de la selección cuidadosa de características y ajustes en los parámetros, ambos modelos continuaron presentando valores bajos, lo que evidenció la necesidad de una versión extra que se describe a continuación.

7.1.1. Modelo Support Vector Machine

El modelo Support Vector Machine se implementó como parte de la estrategia para clasificar el tráfico de red en categorías de benigno y malicioso, gracias a que estos modelos son conocidos por su capacidad para encontrar el hiperplano óptimo que maximiza la separación entre las clases, lo cual resulta crucial en escenarios de detección de intrusos.

Métricas utilizadas

La Tabla 6.1 resume las mejores métricas obtenidas para el modelo SVM. A continuación, se destacan algunos puntos clave:

- **Precisión:** Con un valor de 0.89 para tráfico benigno, el modelo muestra una alta capacidad para identificar correctamente las instancias benignas.
- **Recall:** Sin embargo, el recall para tráfico benigno es de 0.69, lo que indica que el modelo no captura todas las instancias benignas presentes en el conjunto de datos.
- **F1-Score:** El F1-Score para tráfico benigno es de 0.78, reflejando un balance entre precisión y recall.
- **Accuracy:** La tasa de precisión general del modelo es de 0.83, lo que sugiere un buen rendimiento en la clasificación general.
- **Error Tipo I y II:** Los errores Tipo I (falsos positivos) y Tipo II (falsos negativos) son de 0.31 y 0.06 respectivamente, lo que implica que el modelo es más propenso a clasificar erróneamente el tráfico benigno como malicioso.

Matriz de Confusión

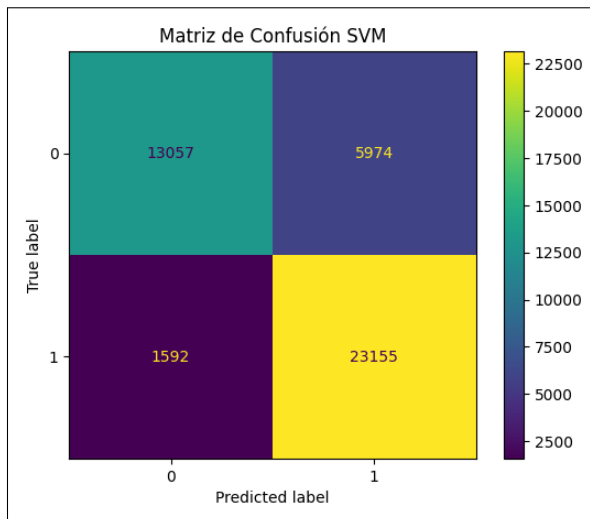


Figura 7.1: Matriz de Confusión del Modelo Support Vector Machine.

La Figura 7.1 muestra la matriz de confusión del modelo Support Vector Machine en su última versión, representando la distribución de predicciones correctas e incorrectas para las clases de tráfico benigno y malicioso. En esta matriz de confusión del modelo Support Vector Machine, se observan 13,057 verdaderos positivos (tráfico malicioso correctamente clasificado) y 23,155 verdaderos negativos (tráfico benigno correctamente clasificado). Sin embargo, también se identifican 1,592 falsos negativos, lo que implica que el modelo no detectó ciertas instancias de tráfico malicioso, clasificándolas incorrectamente como benignas. Además, hay 5,974 falsos positivos, donde tráfico benigno fue clasificado erróneamente como malicioso. Este patrón sugiere que, si bien el modelo es efectivo en la identificación del tráfico malicioso, tiene una tasa significativa de falsos positivos, esto indica que, aunque el modelo tiene un buen desempeño general, tiende a clasificar erróneamente una cantidad significativa de tráfico malicioso como benigno (falsos negativos) y tráfico benigno como malicioso (falsos positivos). lo que puede generar alertas innecesarias.

Curva ROC

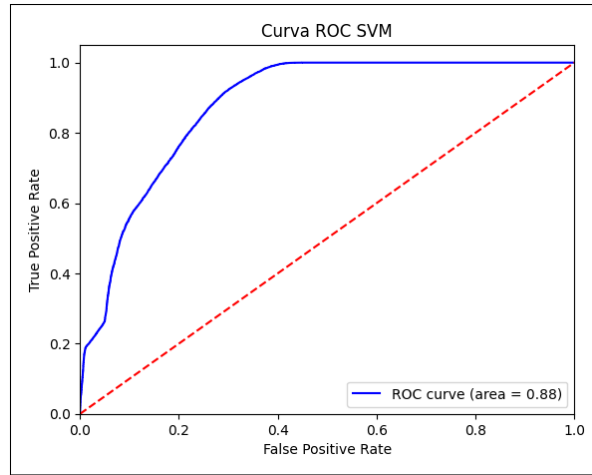


Figura 7.2: Curva ROC del Modelo Support Vector Machine.

La Figura 7.2 presenta la curva ROC del modelo Support Vector Machine en su última versión, mostrando la capacidad del modelo para discriminar entre las clases benigno y malicioso a través de la relación entre la tasa de verdaderos positivos y falsos positivos. Esta curva ROC del modelo Support Vector Machine presenta un AUC (Área Bajo la Curva) de 0.88, lo que indica un buen desempeño general en la discriminación entre tráfico benigno y malicioso. La curva inicialmente sube de manera bastante vertical, indicando así que el modelo es bastante preciso en estas primeras etapas de clasificación. Sin embargo, conforme la curva avanza, se observa un aumento más moderado en el TPR frente al FPR, llegando a un punto donde el TPR es 0.28 con un FPR de 0.07. A partir de ahí, la curva se vuelve más horizontal, especialmente después de un FPR de 0.4, lo que indica que, a medida que se incrementa la sensibilidad, el modelo comienza a generar más falsos positivos. Este comportamiento refleja un buen rendimiento en general, pero con limitaciones en la precisión a medida que se intenta capturar más casos positivos.

7.1.2. Modelo Naive Bayes

El modelo Naive Bayes se ha implementado como parte del enfoque de clasificación para diferenciar entre tráfico benigno y malicioso. Su alta eficiencia y velocidad se deben a que se basa en la probabilidad condicional y en la suposición de independencia entre las características, lo que lo hace útil en entornos de detección de intrusos en tiempo real.

Métricas utilizadas

La Tabla 6.2 presenta las mejores métricas obtenidas para el modelo Naive Bayes. A continuación, se resaltan las observaciones más relevantes:

- **Precisión:** La precisión para tráfico benigno es de 0.70, lo que indica que este modelo tiene una menor capacidad para clasificar correctamente las instancias benignas en comparación con SVM.
- **Recall:** Sin embargo, el recall para tráfico benigno es elevado (0.94), lo que significa que el modelo es efectivo en la identificación de casi todas las instancias benignas.

- **F1-Score:** El F1-Score para tráfico benigno es de 0.80, mostrando un balance razonable entre precisión y recall.
- **Accuracy:** La tasa de precisión general del modelo Naive Bayes es de 0.80, ligeramente inferior a la del modelo SVM.
- **Error Tipo I y II:** Los errores Tipo I y II son de 0.06 y 0.31 respectivamente, sugiriendo que este modelo tiene un mejor desempeño en términos de falsos negativos en comparación con SVM.

Matriz de Confusión

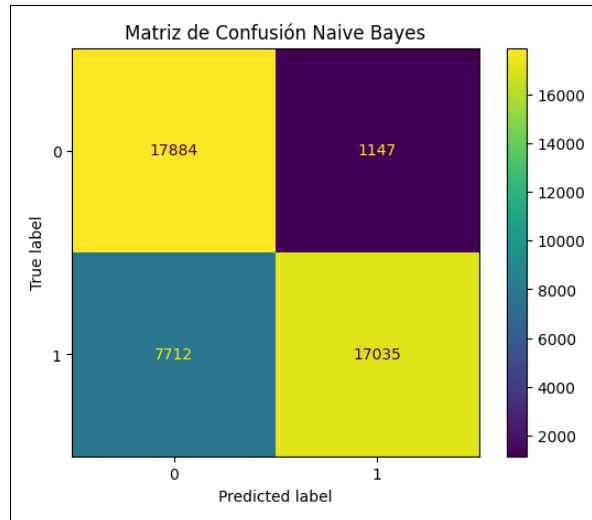


Figura 7.3: Matriz de Confusión del Modelo Naive Bayes.

La Figura 7.3 muestra la matriz de confusión del modelo Naive Bayes en su última versión, indicando el número de predicciones correctas e incorrectas para ambas clases, benigno y malicioso. La matriz de confusión del modelo Naive Bayes muestra 17,884 verdaderos positivos, indicando una alta capacidad para identificar correctamente el tráfico malicioso. No obstante, el número de falsos negativos es considerablemente alto, con 7,712 instancias de tráfico malicioso clasificadas incorrectamente como benignas, lo que podría ser un riesgo importante en entornos de seguridad. Por otro lado, los falsos positivos son bajos, con solo 1,147 instancias de tráfico benigno clasificadas erróneamente como malicioso. Esto sugiere que el modelo es más conservador en la clasificación del tráfico benigno, pero menos efectivo para detectar todo el tráfico malicioso. Por lo tanto el modelo tiene un desempeño desigual, mostrando una tendencia a ser menos estricto con las predicciones de tráfico malicioso.

Curva ROC

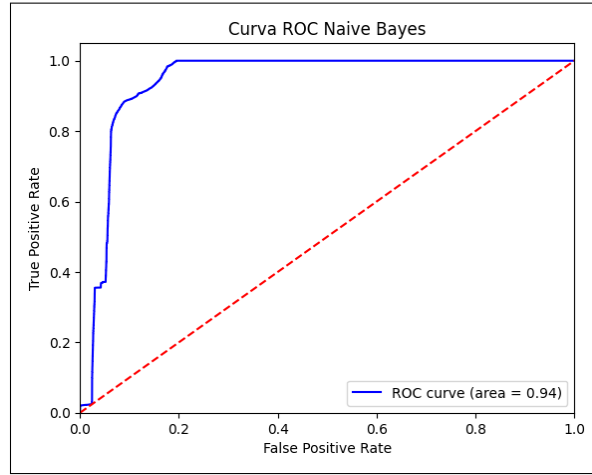


Figura 7.4: Curva ROC del Modelo Naive Bayes.

La Figura 7.4 presenta la curva ROC del modelo Naive Bayes en su última versión, ilustrando su capacidad para discriminar entre las clases benigno y malicioso según la tasa de verdaderos y falsos positivos. Esta curva ROC del modelo Naive Bayes tiene un AUC de 0.94, lo cual es excelente y sugiere que el modelo es muy efectivo en la discriminación entre clases. La curva muestra subidas muy verticales en ciertas partes, lo que implica que el modelo puede aumentar rápidamente el TPR con un incremento mínimo en el FPR en esas zonas específicas. Esto resulta en una curva que se aproxima bastante al vértice superior izquierdo, lo cual es ideal en una curva ROC. No obstante, la naturaleza de la curva, con segmentos que parecen más rectos y menos fluidos, podría indicar que hay ciertas áreas donde el modelo mejora drásticamente, pero de manera menos continua. A partir de un FPR de 0.2, la curva comienza a estabilizarse en una línea horizontal, mostrando que el modelo mantiene su alta precisión, aunque podría estar alcanzando su límite en cuanto a la discriminación adicional sin aumentar falsos positivos.

7.1.3. Comparación entre Modelos

La Tabla 6.3 resume la versión con el mejor rendimiento de ambos modelos. Aunque ambos modelos ofrecen un rendimiento razonable, existen diferencias significativas en la precisión y recall para las clases benignas y maliciosas.

Al comparar los resultados de los modelos Support Vector Machine (SVM) y Naive Bayes, se observan varias diferencias clave en su desempeño. En primer lugar, el modelo SVM muestra un mejor rendimiento general en términos de precisión (accuracy) y puntuación F1 ponderada, con un valor de 0.8271 y 0.84 respectivamente, en comparación con Naive Bayes, que alcanza una precisión de 0.7976 y un F1 ponderado de 0.83. Esto sugiere que SVM tiene una mayor capacidad para clasificar correctamente tanto el tráfico benigno como el malicioso de manera consistente.

La precisión media macro (macro avg precisión) también es ligeramente superior en el modelo SVM (0.84 frente a 0.82 para Naive Bayes), lo que indica que SVM es más equilibrado en la predicción de ambas clases. Esto es importante, especialmente en un contexto de seguridad, donde una correcta clasificación de ambas clases (benignas y maliciosas) es crítica para evitar tanto falsos positivos como falsos negativos.

Pero en el contexto de seguridad, donde el objetivo principal es detectar intrusos con la mayor

efectividad posible, es fundamental priorizar un modelo que minimice la cantidad de falsos negativos, incluso si esto resulta en un mayor número de falsos positivos.

La precisión general del modelo SVM es de 0.8271, comparada con 0.7976 del modelo Naive Bayes. Aunque esta diferencia puede parecer modesta en términos globales, la diferencia en el desempeño al clasificar tráfico malicioso es clave. La matriz de confusión y la curva ROC del modelo SVM muestran que, aunque pueda generar más falsos positivos, su tasa de verdaderos positivos (TPR) es mayor y, por lo tanto, es más efectivo para identificar correctamente el tráfico malicioso. El AUC de la curva ROC del SVM es de 0.88, lo que indica una alta capacidad para discriminar entre tráfico benigno y malicioso.

Por otro lado, el modelo Naive Bayes, a pesar de ser rápido y menos complejo, presenta un mayor número de falsos negativos, lo cual es inaceptable en un entorno donde la detección de intrusos es prioritaria. La matriz de confusión para Naive Bayes muestra 7,712 instancias de tráfico malicioso que fueron clasificadas erróneamente como benignas. Esto representa un riesgo significativo, ya que cualquier intruso que no sea identificado correctamente podría comprometer la seguridad del sistema. Aunque el modelo Naive Bayes tiene una alta tasa de recall para la clase benigna, esto se logra a costa de pasar por alto tráfico malicioso, lo cual es un punto crítico en la seguridad de la red.

En contraste, el modelo SVM, a pesar de tener una precisión ligeramente menor en la identificación de tráfico benigno (lo que resulta en un mayor número de falsos positivos), logra minimizar de manera más efectiva los falsos negativos en la clase maliciosa. Este enfoque es preferible en el contexto de detección de intrusos, ya que es más seguro clasificar un tráfico benigno como malicioso y luego investigarlo más a fondo, que ignorar una amenaza real. La curva ROC del SVM respalda esta ventaja, ya que muestra que el modelo mantiene un alto TPR incluso a medida que aumenta el FPR.

En un contexto donde la seguridad y la detección precisa de intrusos son las máximas prioridades, el modelo SVM es la elección más adecuada. Su capacidad para identificar correctamente el tráfico malicioso, minimizando falsos negativos, lo convierte en una herramienta más confiable y efectiva para proteger la integridad del sistema, aunque esto implique la necesidad de gestionar un mayor número de falsos positivos. Este modelo fue seleccionado para ser implementado en el segundo módulo del proyecto, el cual se encarga de clasificar el tráfico capturado por el primer módulo el cual se encargaba de interceptar y almacenar el tráfico de red entrante y saliente en entornos domésticos utilizados para trabajos empresariales.

7.2. Implementación híbrida

7.2.1. SVM seguido de Naive Bayes

La implementación híbrida 1, que consistió en aplicar primero el modelo Support Vector Machine (SVM) seguido de Naive Bayes (NB), se realizó con el objetivo de aprovechar la capacidad de SVM para identificar patrones bien definidos y maximizar la separación entre las clases. Una vez clasificados los datos por SVM, se buscó complementar el proceso con Naive Bayes para refinar las clasificaciones utilizando su eficiencia en el manejo de probabilidades, especialmente en casos donde las clases tienen distribuciones más difusas.

Métricas utilizadas

La Tabla 6.4 resume las métricas obtenidas para la implementación híbrida 1. A continuación, se destacan algunos puntos clave:

- **Precisión:** Con un valor de 0.93 para la clase benigna y 0.86 para la clase maligna, el modelo híbrido 1 muestra una alta capacidad para identificar correctamente las instancias de ambas categorías.
- **Recall:** El recall alcanzó 0.84 para la clase benigna y 0.94 para la clase maligna, evidenciando una excelente capacidad para capturar las instancias maliciosas, aunque se observan leves omisiones en las instancias benignas.
- **F1-Score:** El F1-Score es de 0.88 para la clase benigna y 0.90 para la clase maligna, lo que refleja un buen equilibrio entre precisión y recall en ambas clases.
- **Accuracy:** La precisión general del sistema híbrido 1 fue de 0.89, indicando un rendimiento sólido en la clasificación global del tráfico de red.
- **Error Tipo I y II:** Los errores Tipo I y Tipo II fueron de 0.16 y 0.06 respectivamente. Esto demuestra que el sistema tiene un mayor sesgo hacia clasificar tráfico benigno como malicioso, pero mantiene un excelente desempeño en minimizar las instancias maliciosas clasificadas erróneamente.

Matriz de Confusión

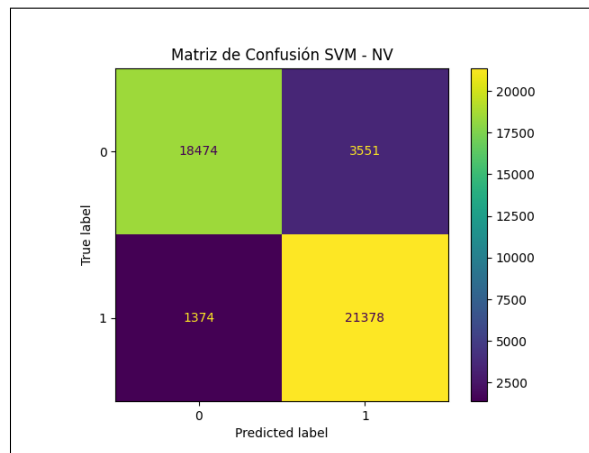


Figura 7.5: Matriz de Confusión de la implementación híbrida 1.

La Figura 7.5 muestra la matriz de confusión del modelo híbrido 1. En esta se observan 21,378 verdaderos positivos y 18,474 verdaderos negativos. Sin embargo, también se identifican 1,374 falsos negativos, lo que indica que el modelo no detectó ciertas instancias de tráfico malicioso, clasificándolas erróneamente como benignas. Además, se presentan 3,551 falsos positivos, donde tráfico benigno fue clasificado de manera incorrecta como malicioso.

Estos resultados evidencian una mejora significativa en el rendimiento general en comparación con los modelos individuales. La implementación híbrida 1 combina las fortalezas de ambos modelos, logrando una mayor precisión y una reducción notable en los errores más críticos, como los falsos negativos. Esto demuestra el potencial de las estrategias híbridas para superar las limitaciones inherentes a los modelos individuales y optimizar la detección de intrusos en el tráfico de red.

7.2.2. Naive Bayes seguido de SVM

La implementación híbrida 2, que invirtió el orden aplicando primero el modelo Naive Bayes (NB) seguido de Support Vector Machine (SVM), se llevó a cabo para evaluar cómo un modelo probabilístico inicial podría prefiltrar el tráfico de red, permitiendo a SVM trabajar con un conjunto de datos más reducido y definido.

Métricas utilizadas

La Tabla 6.5 resume las métricas obtenidas para la implementación híbrida 2. A continuación, se destacan algunos puntos clave:

- **Precisión:** Con un valor de 0.77 para la clase benigna y 0.95 para la clase maligna, el modelo híbrido 2 muestra una alta capacidad para identificar correctamente las instancias de ambas categorías.
- **Recall:** El recall alcanzó 0.95 para la clase benigna y 0.76 para la clase maligna, evidenciando una excelente capacidad para capturar las instancias maliciosas, aunque se observan leves omisiones en las instancias benignas.
- **F1-Score:** El F1-Score es de 0.85 para la clase benigna y 0.84 para la clase maligna.
- **Accuracy:** La precisión general del sistema híbrido fue de 0.84, alcanzando casi el mismo rendimiento que el modelo SVM individual, lo que indica un rendimiento sólido en la clasificación global del tráfico de red.
- **Error Tipo I y II:** Los errores Tipo I y Tipo II fueron de 0.05 y 0.24 respectivamente. Esto demuestra que el sistema híbrido 2 mantiene la tendencia del modelo naive bayes de omitir la detección de tráfico malicioso.

Matriz de Confusión

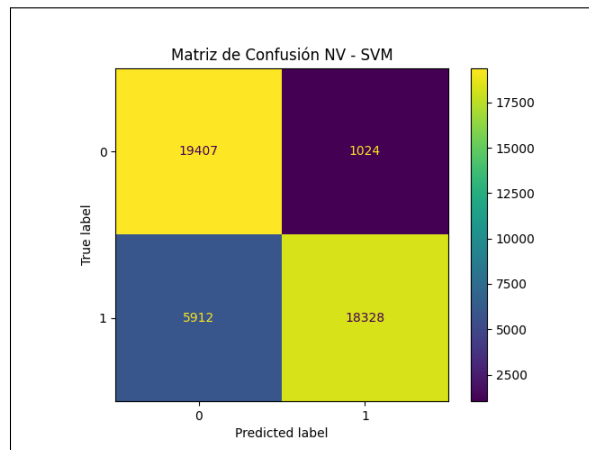


Figura 7.6: Matriz de Confusión de la implementación híbrida 2.

La Figura 7.6 muestra la matriz de confusión del modelo híbrido 2. En esta se observan 18,328 verdaderos positivos y 19,407 verdaderos negativos. Sin embargo, también se identifican 5,912 falsos

negativos, lo que implica que el modelo no detectó ciertas instancias de tráfico malicioso, clasificándolas incorrectamente como benignas. Además, se presentan 1,024 falsos positivos, donde tráfico benigno fue clasificado erróneamente como malicioso. Este patrón sugiere que, aunque el modelo híbrido 2 mejoró el rendimiento en comparación con el modelo SVM individual, todavía muestra una tasa significativa de falsos positivos y mantiene la tendencia de omitir la detección de tráfico malicioso (falsos negativos). A pesar de mejorar el rendimiento de los modelos individuales, el modelo híbrido 2 se mantuvo muy cercano al rendimiento del SVM individual, lo que indica un desempeño sólido pero con limitaciones en la detección del tráfico malicioso.

El desarrollo de un sistema de detección de intrusos en entornos domésticos utilizados para trabajos empresariales ha permitido avanzar en la identificación de características críticas en el tráfico de red. A través de un análisis exhaustivo, se identificaron elementos esenciales en los paquetes de datos, tales como direcciones IP, puertos de origen y destino, y volúmenes de tráfico. Estas características resultaron ser determinantes para detectar patrones anómalos que pueden indicar la presencia de intrusos. La capacidad de distinguir estos aspectos es fundamental para mitigar riesgos y proteger la integridad de la red, especialmente en entornos donde se manejan datos sensibles.

El módulo de captura de paquetes de red implementado permite la interceptación y almacenamiento efectivo del tráfico de datos entrante y saliente en las computadoras utilizadas para tareas profesionales. Este módulo no solo posibilita una recolección eficiente de datos, sino que también facilita un análisis en tiempo real del tráfico de red, lo cual es esencial para identificar comportamientos inusuales y detectar posibles intrusiones. Además, este módulo captura las características denominadas como más relevantes, como las mencionadas anteriormente. Al centrarse en estas características clave, se optimiza el proceso de detección, permitiendo al sistema identificar patrones anómalos y comportamientos sospechosos de manera más efectiva, lo que refuerza la seguridad en entornos laborales.

Se implementó un módulo de análisis que integró uno de los algoritmos de aprendizaje automático para la detección de intrusos, seleccionando entre Naive Bayes y Support Vector Machine (SVM). Tras evaluar las métricas de desempeño de ambos modelos, el SVM se destacó por su alta exactitud 82% y su capacidad para manejar conjuntos de datos con características complejas. Aunque Naive Bayes ofreció una clasificación razonable con una exactitud del 79%, el SVM mostró una mejor capacidad para reducir los falsos negativos, lo que es esencial en la detección de intrusiones. Las métricas obtenidas, incluyendo la curva ROC y la matriz de confusión, respaldaron esta elección, evidenciando que el SVM no solo identificó correctamente los paquetes maliciosos, sino que también brindó un equilibrio adecuado entre precisión y sensibilidad, lo que lo convierte en la opción más efectiva para el sistema propuesto.

El desarrollo de un sistema de detección de intrusos en entornos domésticos utilizados para trabajos empresariales fue posible gracias a la implementación de dos módulos fundamentales. El primero es un módulo de captura de paquetes, que permite la interceptación y almacenamiento del tráfico de datos. Mientras que el segundo es un módulo de análisis que integra un algoritmo de aprendizaje automático, específicamente Support Vector Machine (SVM). A través de un análisis exhaustivo, se identificaron elementos esenciales en los paquetes de datos, como direcciones IP,

puertos de origen y destino, y volúmenes de tráfico, características que resultaron determinantes para detectar patrones anómalos que pueden indicar la presencia de intrusos. Gracias a esta selección, el SVM pudo realizar una clasificación más limpia, reflejada en sus métricas de desempeño. La alineación de estos dos módulos hicieron posible el desarrollo de un sistema enfocado en la detección de intrusos, capaz de identificar paquetes que exhiban comportamientos inusuales dentro de la red de forma eficiente, asegurando así una mayor protección en el entorno laboral.

El modelo SVM ha demostrado ser efectivo para minimizar los falsos negativos y detectar tráfico malicioso, mientras que el modelo Naive Bayes ha sido más conservador en la clasificación del tráfico benigno pero menos efectivo en la detección de tráfico malicioso. Por lo tanto se implementó una estrategia de integración en la que, tras la identificación de tráfico malicioso por el modelo SVM, se realizó una segunda revisión utilizando el modelo Naive Bayes. Esto permitió confirmar la clasificación y reducir la posibilidad de falsos positivos, asegurando que el tráfico clasificado como malicioso fuera efectivamente peligroso antes de tomar medidas adicionales. La implementación híbrida 1, basada en la combinación de SVM seguido de Naive Bayes, alcanzó un 89 % de precisión general y redujo significativamente los falsos negativos. Este enfoque mejoró el desempeño general de la detección de intrusos, logrando un equilibrio entre precisión y sensibilidad, y permitió una mayor seguridad en entornos laborales al mitigar tanto falsos positivos como falsos negativos.

Recomendaciones

Se recomienda aplicar validación cruzada en ambos modelos, tanto en el de Support Vector Machine como en el de Naive Bayes, para mejorar la robustez de la evaluación del rendimiento. La validación cruzada permite dividir el conjunto de datos en varios pliegues y entrenar el modelo múltiples veces, proporcionando una medida más precisa de la efectividad del modelo y reduciendo el riesgo de sobreajuste. Sin embargo, la implementación de esta técnica no fue factible en este caso debido a limitaciones de recursos computacionales. La validación cruzada puede ser intensiva en términos de procesamiento, ya que requiere que el modelo se entrene varias veces sobre diferentes subconjuntos de los datos, lo que puede resultar en tiempos de ejecución prolongados, especialmente con conjuntos de datos grandes. Estas restricciones dificultaron la identificación de configuraciones óptimas y la posibilidad de ajustes en la selección de características o en los parámetros del modelo.

Se sugiere optimizar el rendimiento del sistema implementando una estrategia de lista blanca (whitelist), donde ciertos paquetes previamente identificados como seguros sean excluidos del análisis. Esta estrategia reduciría la carga de procesamiento, lo que podría ser beneficioso en entornos domésticos donde los recursos de hardware son limitados. De esta manera, se podría asegurar que solo los paquetes más relevantes sean clasificados por el algoritmo, permitiendo una detección más eficiente y rápida sin comprometer la capacidad del sistema para detectar intrusiones.

Asimismo, se sugiere complementar el análisis probando con otros algoritmos de aprendizaje automático como Random Forest o Gradient Boosting. Aunque Support Vector Machine y Naive Bayes ofrecen ventajas en términos de simplicidad y rendimiento en ciertos escenarios, explorar estos modelos adicionales podría proporcionar nuevas perspectivas y mejorar el rendimiento general, especialmente en contextos donde se requiere manejar relaciones más complejas entre las variables o se desea robustez adicional frente a diferentes configuraciones de datos.

Bibliografía

- [1] Cyberzaintza . El malware cada vez más orientado al entorno empresarial e industrial | actualidad | bcsc, 2021.
- [2] Juan Ranchal. Los 10 peores incidentes de ciberseguridad en 2020, 12 2020.
- [3] Citrix. What is remote work security? how to secure the remote workforce - citrix, 07 2022.
- [4] Maria Pérez-Ramírez, Miguel-Ángel Díaz, and Juan-Antonio Lopez. The impact of cyber threats on businesses: a review of the current landscape. *information management*, 64, 162-175, 2017.
- [5] García Monje and Robert Alexander. Seguridad informatica y el malware.
- [6] Faster Capital . Deteccion de malware mejora de la deteccion de malware mediante tecnicas de terceros, 2024.
- [7] Alexander Küchler, Alessandro Mantovani, Yufei Han, Leyla Bilge, and Davide Balzarotti. Does every second count? time-based evolution of malware behavior in sandboxes. *Proceedings 2021 Network and Distributed System Security Symposium*, 2021.
- [8] Semana. Una empresa puede tardar hasta 7 meses en detectar un ataque cibernético, 09 2020.
- [9] Esteban Forero. Seguridad en redes.
- [10] Juliana Tejada Berrio. Datos personales y pilares de la seguridad de la información, 2021.
- [11] Jairo Eduardo Márquez Díaz. Armas cibernéticas. malware inteligente para ataques dirigidos. *Ingenierías USBMed*, 8:48, 07 2017.
- [12] Rivera Guevara. Universidad internacional de la rioja máster universitario en seguridad informática detección y clasificación de malware con el sistema de análisis de malware cuckoo trabajo de fin de master, 09 2018.
- [13] Javier Guña-Moya, Andrea Sánchez-Zumba, Paúl Chérrez-Vintimilla, Lorena Chulde-Obando, Paulina Jaramillo-Flores, and Cristian Pillajo-Rea. Ataques informáticos más comunes en el mundo digitalizado, 11 2022.
- [14] Antonio Jesús and Caro Alonso-Rodríguez. MÁster interuniversitario de seguridad de las tecnologías de la informaciÓn y de las comunicaciones (misitic) trabajo fin de máster "man in the middle attacks on ssl/tls", 2013.
- [15] Carrera De Informática. Universidad mayor de san andrÉs facultad de ciencias puras y naturales postulante: Efrain rojas tutor metodolÓgico: M.sc. franz cuevas quiroz asesor: M.sc. moises martin silva choque nuestra seÑora de la paz -bolivia 2018, 2018.

- [16] Katherine Pereira. Cross-site scripting.
- [17] Carlos Alberto Flores Quispe. Tipos de hackers. *Revista de Información, Tecnología y Sociedad*, 8:16, 2013.
- [18] Jorge Mieres. Buenas prácticas en seguridad informática, 06 2009.
- [19] A. Gómez Vieites. La lucha contra el ciberterrorismo y los ataques informáticos.
- [20] Guillermo Roberto Solarte Martinez, Carlos Alberto Ocampo, and Yanci Viviana Castro Bermúdez. Sistema de detección de intrusos en redes corporativas. *Scientia et technica*, 22(1):60–68, 2017.
- [21] Mario Ávila Pérez. Implementación de un ids. *Gestión Competitividad e Innovación*, 8:11–23, 07 2020.
- [22] R Alonso, J Diaz-Verdejo, A Alonso, G Madinabeitia, F Muñoz, and Daniel Saucedo Aranda. Generación automática de firmas para detección de ciberataques basados en uri, 2021.
- [23] Alejandro Barrera García-Orea. Presente y futuro de los ids.
- [24] Edgar Leonardo Prieto Perilla. Sensores de red para un sistema de detección de intrusos distribuido. <https://repositorio.uniandes.edu.co/>, 2002.
- [25] CloudFlare. What is a packet? | network packet definition | cloudflare. *Cloudflare*, 2024.
- [26] M. Bykova, S. Ostermann, and B. Tjaden. Detecting network intrusions via a statistical analysis of network packet characteristics, 2001.
- [27] KeepSolid Inc. ¿qué es la captura de paquetes? - términos y definiciones de ciberseguridad, 2024.
- [28] Wireshark Foundation. Wireshark, 2024.
- [29] tcpdump. Tcpdump/libpcap public repository. *Tcpdump.org*, 2017.
- [30] Dor Green. pyshark, 11 2023.
- [31] Rivero Pérez and Jorge Luis. Técnicas de aprendizaje automático para la detección de intrusos en redes de computadoras. *Revista Cubana de Ciencias Informáticas*, 8:52–73, 12 2014.
- [32] Gonzalo Valdezate Álvarez. Escuela de ingeniería informática.
- [33] Ella Siman. ¿qué es un conjunto de datos? definición.
- [34] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41:1–58, 07 2009.
- [35] Agustin Parmisano, Sebastian Garcia, and Maria Jose Erquiaga. Iot-23 dataset: A labeled dataset of malware and benign iot traffic., 2020.
- [36] Python Software Foundation. csv — csv file reading and writing — python 3.8.1 documentation, 2020.
- [37] Python Software Foundation License. ipaddress — ipv4/ipv6 manipulation library — python 3.9.0 documentation.
- [38] Python Software Foundation. Datetime — basic date and time types — python 3.7.2 documentation, 2002.
- [39] Python Software Foundation. collections — container datatypes — python 3.10.0 documentation.