UNIVERSIDAD MAYOR DE SAN ANDRÉS

FACULTAD DE CIENCIAS PURAS Y NATURALES

CARRERA DE INFORMÁTICA

TESIS DE GRADO

"MODELO PARA LA DETECCIÓN DE ESCANEO DE PUERTOS DE LA COMPUTADORA EN UNA RED WLAN"

MENCION: CIENCIAS DE LA COMPUTACIÓN

POSTULANTE: LEONARDO JULIO RIOS ALIAGA

TUTOR METODOLÓGICO: P.Ph.D. YOHONI CUENCA SARZURI

LA PAZ – BOLIVIA

2021

HOJA DE CALIFICACIONES

UNIVERSIDAD MAYOR DE SAN ANDRES

FACULTAD DE CIENCIAS PURAS Y NATURALES

CARRRERA DE INFORMATICA

Tesis de grado:

MODELO PARA LA DETECCIÓN DE ESCANEO DE PUERTOS DE LA COMPUTADORA EN UNA RED WLAN

Presentado por: Leonardo Julio Rios Aliaga

Para optar al grado Académico de Licenciado en Informática

Mención Ciencias de la computación

Nota Numeral:……

Nota Literal:……

Ha sido:……

Director de la carrera de Informática: Ph.D. José María Tapida Baltazar

Tutor: P.Ph.D Yohoni Cuenca Sarzuri

Tribunal: ……..

Tribunal: ……..

Tribunal: ……..

UNIVERSIDAD MAYOR DE SAN ANDRÉS

FACULTAD DE CIENCIAS PURAS Y NATURALES

CARRERA DE INFORMÁTICA

LA CARRERA DE INFORMÁTICA DE LA FACULTAD DE CIENCIAS PURAS Y NATURALES PERTENECIENTE A LA UNIVERSIDAD MAYOR DE SAN ANDRÉS AUTORIZA EL USO DE LA INFORMACION CONTENIDA EN ESTE DOCUMENTO SI LOS PROPOSITOS SON ESTRICTAMENTE ACADEMICOS.

LICENCIA DE USO

El usuario está autorizado a:

- Visualizar el documento mediante el uso de un ordenador o dispositivo móvil
- Copiar, almacenar o imprimir si ha de ser de uso exclusivamente personal y privado
- Copiar textualmente partes de su contenido mencionado la fuente y/o haciendo la referencia correspondiente respetando normas de redacción e investigación.

El usuario no puede publicar, distribuir o realizar emisión o exhibición alguna de este material, sin la autorización correspondiente.

DEDICATORIA

A mi madre Zulema Aliaga y a mis hermanos, gracias por tanto y perdón por tan poco

A Danae Oruño Mamani por estar presente en muchos momentos importantes de mi vida.

A las personas que comparten conocimiento de forma absolutamente desinteresada.

AGRADECIMIENTO

A mi madre, hermanos y a mi pareja.

A mi docente de Taller II Ph.D. José María Tapia por sus consejos en el área de conocimiento necesarios para la elaboración de la tesis.

A mi tutor P.Ph.D Yohoni Cuenca Sarzuri por haberme guiado durante cada etapa de la creación de la tesis, por sus valiosas observaciones y consejos.

A la carrera de informática por la educación que me brindo durante mis años como estudiante.

leonardo_jlo@hotmail.com

RESUMEN

El escaneo de puertos es uno de los ataques más sencillos de ejecutar. Mediante esta técnica se puede saber que puertos están abiertos, filtrados o cerrados dentro de una red o una computadora. A su vez hoy en día una computadora en una WLAN de un hogar común es ahora uno de los puntos más vulnerables para una institución a este ataque pues el trabajo remoto es más popular que nunca.

La presente tesis combina dos áreas importantes en la actualidad, la seguridad informática y aprendizaje automático, en esta última área existen diferentes tipos de algoritmos de aprendizaje. Uno de ellos es el algoritmo de clasificación llamado Bagging. Este algoritmo se utiliza en nuestro estudio para la detección de escaneo de puertos, para lo cual se ha generado un conjunto de datos de transacciones de red que contienen diversos escaneos así como transacciones legítimas. Aplicando la metodología KDD, se aplicó una clasificación a los datos de entrenamiento, luego se realiza una codificación one-hot, y se reduce la dimensionaldad. Posteriormente se entrena el Algoritmo Bagging para detectar si una transacción es un escaneo o no.

Los resultados obtenidos después de entrenamiento es de 99.96% de forma general, lo que nos da idea que el algoritmo de Bagging puede clasificar si un flujo de red es un escaneo o no. Se aplicaron las pruebas estadísticas de análisis de la matriz de confusión, gmean y error fuera de la bolsa para verificar la fiabilidad de los resultados obtenidos.

Palabras clave: escaneo de puertos, aprendizaje automático, flujo de red, bagging, árbol de decisión

ABSTRACT

Port scanning is one of the simplest attacks to execute. Using this technique you can know which ports are open, filtered or closed within a network or a computer. In turn, today a computer in a common home WLAN is now one of the most vulnerable points for an institution to this attack since remote work is more popular than ever.

This thesis combines two important areas today, computer security and machine learning, in the latter area there are different types of learning algorithms. One of them is the classification algorithm called Bagging. This algorithm is used in our study for port scan detection, for which a set of network transaction data has been generated that contains various scans as well as legitimate transactions. Applying the KDD methodology, a classification was applied to the training data, then a one-hot coding is performed, and the dimensionality is reduced. Subsequently, the Bagging Algorithm is trained to detect whether a transaction is a scan or not.

The results obtained after training is 99.96% in general, which gives us an idea that the Bagging algorithm can classify whether a network flow is a scan or not. The statistical tests of analysis of the confusion matrix, gmean and out-of-bag error were applied to verify the reliability of the results obtained.

Keywords: port scanning, machine learning, network flow, bagging, decision tree

ÍNDICE GENERAL

CONTENIDO **Pag.**

MARCO INTRODUCTORIO…….....1

1.1. Introducción.....1

1.2 Estado del Arte.....3

1.2.1 Seguridad informática en instituciones.....3

1.2.2 Seguridad en entornos domésticos….....6

1.2.3 Trabajos relacionados a nivel global.....8

1.2.4 Trabajos relacionados a nivel nacional.....10

1.3 Planteamiento del problema…….....12

1.3.1 Problema general...….....12

1.3.2 Problemas específicos….....12

1.4 Hipótesis.....12

1.5 Objetivos.....13

1.5.1 Objetivo general.....13

1.5.1 Objetivos específicos.....13

1.6 Justificaciones.....13

1.6.1 Justificación social.....13

1.6.2 Justificación técnica.....14

1.6.3 Justificación económica.....14

1.6.4 Justificación científica.....14

1.7 Límites y alcances.....15

1.7.1 Delimitación temática.....15

1.7.2 Delimitación espacial.....15

1.7.3 Delimitación temporal.....15

1.8 Metodologías de investigación…….……16

1.8.1 Metodologías de investigación tradicionales…….…....16

1.8.2 Metodologías de diseño …….…….....17

1.9 Aportes…….….17

1.9.1 Aporte teórico…….....17

1.9.2 Aporte práctico……...17

MARCO TEÓRICO…….....18

2.1. Capa de transporte.....18

2.2 Puertos lógicos.....20

2.3 Protocolo de control de transporte (TCP).....22

2.3.1 Características y funcionamiento……..22

2.3.2 Estructura de un segmento TCP.…….….….23

2.3.4 HandShaking.…….…....24

2.4 Protocolo de datagramas de usuario (UDP).....25

2.4.1 Estructura de un datagrama UDP……..…26

2.5 Flujo de red.....27

2.6 Escaneo de puertos.....28

2.7 Aprendizaje automático.....30

2.8 Aprendizaje automático supervisado.....34

2.8.1 Predicción……..…..34

2.8.2 Inferencia……35

2.9 Árboles de decisión…….…….…36

2.9.1 Proceso de creación ……..…….…38

2.10 Bagging……40

2.11 Matriz de confusión…….…….....42

2.11.1 Indicadores de evaluación de rendimiento……..43

2.12 Gmean…….……44

2.13 Error fuera de la bolsa…….….45

2.14 Metodología KDD……...46

2.15 Diferencia entre modelo y algoritmo en aprendizaje automático……47

MARCO APLICATIVO…….....49

3.1. Recolección de los datos.....49

3.1.1 Generación de los datos……..49

3.1.2 Captura de los datos ……..52

3.2 Preparamiento de los datos.....53

3.2.1 Manipulación de archivos pcap……...53

3.2.2 Manipulación de los flujos de red……...60

3.3 Pre procesamiento de los datos.....64

3.3.1 Categorización de los datos.....64

3.3.2 Reducción de dimensionalidad.....65

3.4 Aplicación de metodología.....69

3.5 Modelo clasificador final….....70

ANÁLISIS DE DATOS Y RESULTADOS…….....72

4.1. Análisis de resultados mediante matriz de confusión.....72

4.2 Gmean.…….....75

4.3 Error fuera de la bolsa……......76

CONCLUSIONES Y RECOMENDACIONES.....78

5.1. Conclusiones.....78

5.2 Recomendaciones.....79

INDICE DE FIGURAS

Figura 1.1. 4

Figura 1.2. 4

Figura 2.1. 19

Figura 2.2. 24

Figura 2.3. 25

Figura 2.4. 27

Figura 2.5. 31

Figura 2.6. 32

Figura 2.7. 33

Figura 2.8. 36

Figura 2.9. 38

Figura 3.1.....49

Figura 3.2. 51

Figura 3.3. 52

Figura 3.4. 54

Figura 3.5. 62

Figura 3.6. 64

Figura 3.7.....67

Figura 3.8. 69

Figura 3.9. 70

Figura 3.10.....71

Figura 4.1. 72

INDICE DE TABLAS

Tabla 2.1. 21

Tabla 2.2. 43

Tabla 3.1. 58

Tabla 3.2. 61

Tabla 3.3. 62

Tabla 3.4. 65

Tabla 3.5. 66

Tabla 3.6. 68

CAPÍTULO I

Marco Introductorio

1.1. Introducción

El 2020 es un año que no pasará desapercibido en la historia de la humanidad. Fue en ese año cuando el Covid-19 se convirtió en una pandemia y que, para bien o para mal, forzó a la sociedad a adaptarse a un nuevo estilo de vida. Muchas fuentes de empleo se vieron afectadas más que otras no solo por la enfermedad en sí, sino del daño colateral causado por las medidas de contención en contra la pandemia más utilizadas globalmente: el distanciamiento social y las cuarentenas (International Labour Organization, 2020). Sin embargo hubo áreas de trabajo que lograron sobrevivir o incluso evolucionar frente a estas medidas de contención. Cualquier persona que realiza un trabajo mediante una computadora podría en teoría realizarlo desde cualquier otro lugar siempre y cuando se disponga del equipo adecuado para su trabajo y las herramientas necesarias (energía eléctrica, internet, etc.). Eso fue lo que ocurrió el 2020; millones de personas alrededor del mundo empezaron a trabajar desde su casa, y muchas otras fuentes laborales que tradicionalmente no se desempeñan mediante una computadora, como por ejemplo el área de la educación, se adaptaron para también trabajar de manera remota (Marinova, 2020).

El trabajo remoto demostró ser una alternativa para el desarrollo de actividades eficiente en muchas áreas laborales. Muchas organizaciones tanto públicas como privadas planean mantener este método de trabajo de forma permanente debido a la reducción de costos que brinda. Sin embargo, hablando de seguridad informática, muchas medidas que las instituciones tomaron previamente para proteger su información ahora quedaron obsoletas. Firewall en redes privadas o

sistemas de detección de intrusos funcionan adecuadamente si todos los equipos se encuentran dentro la misma red. Pero ese ya no es el caso, pues ahora muchas personas desempeñan sus actividades laborales desde la comodidad de sus casas, en redes de hogar comunes y con medidas de protección básicas o nulas.

El escaneo de puertos es uno de los ataques más sencillos de ejecutar. Mediante esta técnica se puede saber que puertos están abiertos, filtrados o cerrados dentro de una red o una computadora. Si bien califica más como un análisis, pues no hace daño al objetivo como tal, también puede ser utilizado como ataque en la forma de denegación de servicio (Henry-Stocker, 2018). El escaneo de puertos es algo a tomar en cuenta en el área de seguridad informática pues de este espionaje pueden derivar varios tipos de ataques haciendo uso de las vulnerabilidades descubiertas. Una computadora en una WLAN de un hogar común es ahora uno de los puntos más vulnerables para una institución a este ataque. Si bien estos equipos pueden poseer un firewall y/o un antivirus, estas serían configuraciones de seguridad básicas o incluso nulas si, por un motivo que se desconoce, el usuario decide apagar el firewall o si no actualiza su antivirus. Este es un problema reciente, pues en el trabajo presencial, siempre se tenía personal de IT encargado de recomendar el uso de firewall y de verificar que los equipos tengan con un antivirus actualizado, además de que una red de trabajo en si posee otras medidas de seguridad.

La presente tesis propone realizar un modelo, es decir un clasificador, de aprendizaje supervisado para la detección de escaneo de puertos en una computadora dentro una red de WLAN, asumiendo que el usuario no utiliza un firewall o un antivirus.

1.2 Estado del arte

1.2.1 Seguridad informática en instituciones.

Hoy en día existen varias formas de detectar o prevenir ataques o posible comportamiento malicioso en una red. Muchas infraestructuras de red utilizan sistemas de detección de intrusos.

Un sistema de detección de intrusos, denotado como IDS, es un software o hardware para detectar cualquier tipo de actividad maliciosa o ataque contra el sistema o la red. Un IDS recolecta datos de diferentes fuentes dentro de una computadora o una red como por ejemplo el historial del sistema, historial de la red, estado de funcionamiento del equipo, datos de los paquetes en la red, etc. Luego analiza la información obtenida para encontrar posibles violaciones de seguridad y finalmente dispara una alerta para que los administradores de sistemas para revisen la intrusión encontrada. Los IDS se clasifican tradicionalmente en dos categorías: Sistemas de detección de intrusos basados en red y sistemas de detección de intrusos basados en host, denotados como NIDS y HIDS respectivamente (Khraisat, ondal, Vamplew y Kamruzzaman, 2019).

Los NIDS son colocados en conjunto con la red para monitorear todo el tráfico existente dentro de esta como se observa en la Fig.1.1. Sirve para detectar escaneo de puertos, ataques de paquetes fragmentados, secuencia de comandos en sitios cruzados (XSS), etc.

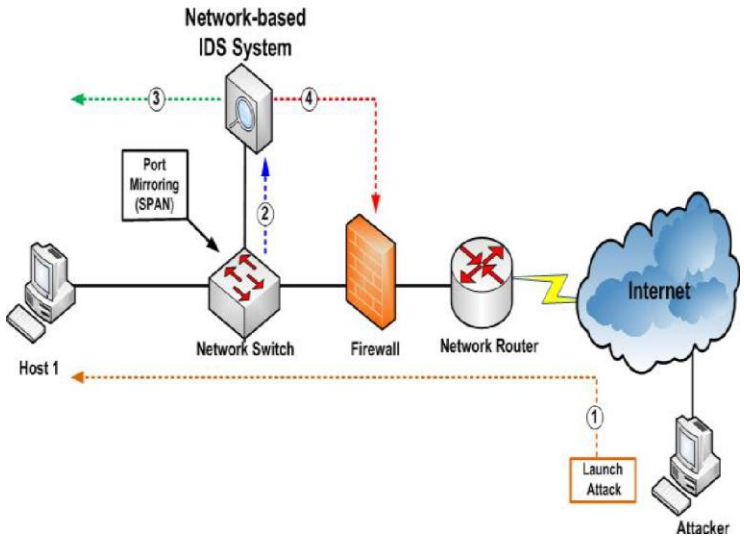


Figura 1.1 Sistemas de detección de intrusos basado en red (NIDS)

Los HIDS por otro lado son colocados en un host para escanear y monitorear todos los procesos dentro de un equipo como en la Fig 1.2. Sirven para detectar registradores de teclas, accesos no autorizados, actividad de bots, etc.

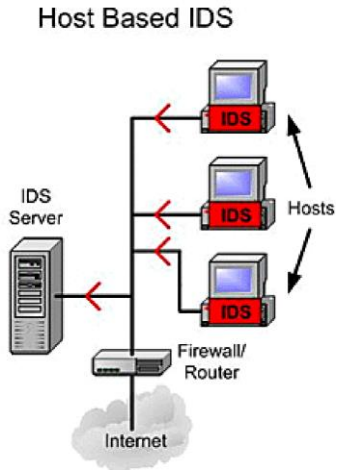


Figura 1.2. Sistema de detección de intrusos basado en host

Los IDS también pueden ser clasificados en base a su forma de funcionamiento. Entre las más comunes se tienen los sistemas de detección de intrusos basados en firmas y sistemas de

detección basados en anomalías, denotados como SIDS y AIDS respectivamente (Khraisat et al., 2019).

Los SIDS se enfocan en encontrar firmas o patrones de un posible evento malicioso en una base de datos de firmas de actividades o características maliciosas. La mayoría de los IDS son de este tipo. Estos necesitan una actualización constante de firmas maliciosas conocidas para garantizar seguridad. La desventaja de los SIDS es su naturaleza rígida. Pequeños cambios en la ejecución de un ataque conocido pueden hacer que este pase inadvertido sin que los SIDS puedan detectarlo, esto debido a que está ligeramente nueva firma no se encuentra en la base de datos. Por otra parte, cuando la base de datos crece, también lo hace el proceso de carga en el sistema al analizar cada conexión con los registros en la base.

En contraste a los SIDS, los sistemas de detección de intrusos basados en anomalías, AIDS, detectan ataques desconocidos que son difíciles de detectar con un método rígido. Debido al crecimiento acelerado en malware y tipos de ataques, los AIDS usan técnicas de aprendizaje automático para comparar modelos de comportamiento confiables con el nuevo comportamiento. Estos sistemas son relativamente nuevos y dan buenos resultados para determinar si alguien está probando, escaneando o analizando la red antes de que el ataque tome lugar. Sin embargo, es propenso a generar falsos positivos al calificar el comportamiento benigno previamente desconocido como una anomalía.

1.2.2 Seguridad en entornos domésticos.

Existen diversos esquemas de IDS en infraestructuras de red inalámbricas, algunas de ellas enfocadas exclusivamente a sensores móviles, mientras que otras a aplicaciones estacionarias donde existe una unidad centralizada de cómputo (Butun, Morgera, y Sankar, 2013).

Si bien todos estos sistemas de seguridad probaron ser efectivos en redes de diversa índole, no fueron pensados ni aplicados en redes WLAN dentro un hogar común. La mayoría de estas redes otorgadas por un proveedor de internet local funcionan mediante un router de espectro de radio de banda de 2.4GHz usualmente configurado con una red WiFi con programa de certificación de seguridad WPA2 AES (Jackson, 2018) en donde la seguridad se basa en una clave pre-compartida para garantizar el ingreso de usuarios conocidos a la red. Pese a estas medidas de seguridad, es posible que un individuo pueda obtener la contraseña de una red WiFi y así ingresar a esta, pues por su naturaleza abierta en comparación a redes LAN existen diversas herramientas y/o procedimientos para hacerlo (Griffith, 2019). En estos días es común que instituciones y/o negocios donde brinden una red WLAN publica para sus clientes; incluso es posible que un atacante tenga acceso a la red desde el principio, pues no es raro ver casas o edificios donde se comparte una sola red WiFi entre varios habitantes del mismo.

El nivel de protección máxima que se puede esperar de una computadora en una red WLAN de hogar común es un firewall y un antivirus, que si bien ofrecen un nivel de protección estándar aceptable para actividades comunes, resulta insuficiente comparado al nivel de seguridad que diversos IDS ofrecen en un red para la protección de información de una institución con su personal de IT. Además que muchos de estos IDS fueron pensados para redes privadas Ethernet, no redes WiFi con enrutadores genéricos de conocimiento público. Aparte de las herramientas de seguridad que se espera tener en una red WiFi común, una computadora

puede escoger su perfil de configuración en la red en la que se encuentra para así mejorar un poco la seguridad en esta. Entre los perfiles de red de una computadora, podemos encontrar los siguientes:

- **Público:** Este perfil es usado cuando el equipo se conecta a una red pública como un restaurante, librería o aeropuerto.
- **Privado***:* Este perfil es utilizado cuando el equipo es parte de una red privada. El equipo no está conectado directamente a internet. Una red de hogar califica como red privada.
- **Dominio:** Se aplica a un adaptador de red cuando se conecta a una red en la que puede detectar un controlador de dominio del dominio al que está unido el equipo. Una red corporativa califica para este perfil.

Aunque no parezca intuitivo, el perfil público es el más seguro de todos, seguido del privado y por último del de domino. Una computadora en una red WiFi de hogar se espera que tenga un perfil privado, bajando así su nivel de seguridad. Una desventaja, consecuencia de este perfil, es que la configuración del firewall recae en perfiles privados para facilitar el uso de archivos compartidos y otras ventajas en la red. Otra desventaja de un firewall es su naturaleza rígida, un problema similar al de los SIDS, donde una variante diminuta en la ejecución de un ataque podría pasar desapercibida.

Por otra parte, un antivirus también funciona de manera similar a un SIDS, pues bloquea la instalación de posibles malwares comparando esta con una base de datos de firmas conocidas. Se especializan en eventos que ocurren internamente, no en la red. Es por eso que tiene parentescos un HIDS pues monitorea distintas fuentes de información dentro de una computadora. Pero si el

hecho de mantener la base de datos actualizada era una desventaja en un SIDS, en un antivirus de computadora en un hogar común es más que eso, pues la gran mayoría de personas no invierten recursos en mantener su antivirus actualizado, prefiriendo usar solo el tiempo de prueba que estos ofrecen.

1.2.3 Trabajos relacionados a nivel global.

- Uso de un modelo de aprendizaje profundo para la detección de escaneo de redes

  "En los últimos años, los ciberataques nuevos y devastadores amplifican la necesidad de prácticas sólidas de ciberseguridad. La prevención de nuevos ciberataques requiere la invención de sistemas de detección de intrusiones (IDS), que pueden identificar ataques nunca antes vistos. Muchos investigadores han intentado producir IDS basados en anomalías, sin embargo, todavía no pueden detectar el tráfico de red malicioso de manera suficientemente consistente como para garantizar su implementación en redes reales. Obviamente, sigue siendo un desafío para la comunidad de seguridad producir IDS que sean adecuados para su implementación en el mundo real. En este documento, proponemos un nuevo enfoque que utiliza una Deep Belief Network con una combinación de métodos de aprendizaje automático supervisados y no supervisados para la detección de ataques de escaneo de puertos: la tarea de sondear redes empresariales o servicios de Internet, buscar vulnerabilidades o formas de infiltrarse en activos de IT . Nuestro enfoque propuesto se probará con conjuntos de datos de seguridad de red y se comparará con métodos existentes anteriormente." (Viet, Nguyen, Thi Trang, Shone, 2018).

Un punto débil de esta investigación, radica en el hecho que se trabajó con datos generados artificialmente. Además que como muchos otros modelos ya desarrollados, estos fueron pensados para redes empresariales, no entornos domésticos.

- Encuesta sobre la detección de intentos de escaneo de puertos con análisis combinado de máquina de vectores de soporte y algoritmos de aprendizaje profundo.

"En comparación con el pasado, la seguridad de los sistemas en red se ha convertido problema universal crítico que influye en las personas, las empresas y los gobiernos. La tasa de ataques contra sistemas en red ha aumentado drásticamente, y las estrategias utilizadas por los atacantes continúan evolucionando. Por ejemplo, la privacidad de la información importante, seguridad de las plataformas de datos almacenados, disponibilidad de conocimientos, etc. Sobre estos problemas, el ciber terrorismo es uno de los temas más importantes en el mundo actual. Terror cibernético, que ha causado muchos problemas a personas e instituciones, ha alcanzado un nivel que podría amenazar al público y la seguridad del país por parte de diversos grupos como organizaciones criminales, profesionales y ciber activistas. La detección de intrusiones es una de las soluciones contra estos ataques. Un enfoque gratuito y eficaz para diseñar. Los sistemas de detección de intrusiones (IDS) son aprendizaje automático. En este estudio, se utilizaron algoritmos de aprendizaje profundo y la máquina de vectores de soporte (SVM) para detectar intentos de escaneo de puertos basados en el nuevo conjunto de datos CICIDS2017" (Kurkure, 2019).

Este trabajo se enfoca más en la comparativa de dos tipos de modelos diferentes para la detección de escaneos de puertos. Si bien se obtuvieron buenos resultados, los datos trabajados fueron generados artificialmente.

1.2.4 Trabajos relacionados a nivel nacional.

- Auditoría de seguridad de redes inalámbricas de área local wireless local area Network (WLAN) - Tesis de Grado

  "Hoy en día las redes de computadoras se han convertido en el soporte de cualquier entidad, ya que estas ayudan a un mejor desarrollo de actividades y servicios. Estos servicios brindan información sensible y de vital importancia para la organización, generando diversas acciones relacionadas con la toma de decisiones, políticas de administración, informes económicos entre otros, siendo ésta susceptible a cualquier variación, modificación o pérdida. "Las redes inalámbricas aparecen como una nueva solución de comunicación en la organización, las redes inalámbricas de área local van tomando un papel muy importante en las organizaciones. Una conexión inalámbrica se ha convertido en una alternativa para ofrecer conectividad en lugares donde resulta complicado o imposible brindar servicio con una red cableada"1. Pero toda nueva tecnología trae consigo nuevos problemas, y en el caso de las redes inalámbricas una de las principales dificultades es la seguridad. Se debe considerar que el desconocimiento de las herramientas de seguridad disponibles para redes inalámbricas, hace que estas redes de comunicación sean inseguras. Estos problemas de seguridad abarcan cuestiones como vulnerabilidades en los protocolos de comunicación inalámbrica y mecanismos de autenticación y protección de datos débiles dentro de la red de la organización. Es por

ello que se plantea la necesidad de un control sobre estas redes LANs inalámbricas, ya que la administración de una organización desconoce la magnitud del problema con el que se enfrenta y, generalmente no se invierte ni el capital humano ni económico necesarios para prevenir daños y/o pérdidas de información; siendo esta, usualmente, el conocimiento con que se cuenta." (Monzón, 2011).

En esta investigación queda expuesta la vulnerabilidad más grande de las redes inalámbricas, su fácil acceso por intrusos no autorizados. Se recomendó no solo mejorar la seguridad de acceso, sino también la seguridad dentro de estas WLAN.

- IPSVOFSL: Sistema Inteligente de Prevención de Intrusiones - Tesis de Grado

  "La amplia gama de ataques a redes de computadoras y sus variantes han hecho de la detección y prevención de intrusiones una ardua tarea, sobre todo, teniendo en cuenta que además de la implantación de herramientas diseñadas para tal cometido, es necesario actualizar constantemente la base de conocimientos o reajustar los parámetros de configuración de dichas herramientas si la red o el contexto de operaciones cambia en determinado momento. Es por esto que la incorporación de técnicas de inteligencia artificial para adicionar cierto grado de adaptabilidad a los detectores de intrusos está tomando cada vez más fuerza. En la presente tesis de grado "IPSVOFSL: Sistema Inteligente de prevención de intrusiones" se describe el análisis realizado y los resultados obtenidos de introducir un elemento inteligente, basado en redes neuronales, orientado a la detección del problema específico del escaneo de puertos." (Villegas Pacasi, 2009).

  Este sistema logró resultados óptimos a la hora de la detección y manejo de diversos tipos de escaneos de puertos. Sin embargo el sistema fue pensado para redes LAN en entornos empresariales, no las redes WLAN de hoy en día. En la actualidad también existen

nuevas herramientas que facilitan el análisis exploratorio de datos para decidir mejor qué algoritmo(s) utilizar para la creación del modelo.

1.3 Planteamiento del problema

1.3.1 Problema general.

Las medidas de seguridad actuales contra escaneo de puertos a computadoras en redes WLAN son insuficientes.

1.3.2 Problemas específicos.

- El escaneo de puertos es una de las amenazas más comunes para descubrir vulnerabilidades en una computadora y generalmente conlleva a ataques más severos.
- A causa del trabajo remoto, hoy en día muchas computadoras dentro de una red WLAN manejan información sensible de diversa índole.
- Existe escases de técnicas y algoritmos de aprendizaje automático aplicados en la seguridad de WLANs.
- La mayoría de clasificadores actuales carecen de formación en entornos de red WLAN con datos actualizados.

1.4 Hipótesis

Se puede crear un modelo clasificador de aprendizaje automatizado para detectar escaneos de puertos en una computadora dentro de una WLAN.

1.5 Objetivos

1.5.1 Objetivo General.

Desarrollar un modelo clasificador de aprendizaje supervisado para la detección de escaneo de puertos de una computadora dentro una red WLAN.

1.5.2 Objetivos específicos.

- Detectar si un flujo de red pertenece a un escaneo de puertos.
- Minimizar el riesgo que el trabajo remoto conlleva en cuanto a la protección de información.
- Aplicar técnicas de aprendizaje automático para la elaboración de un modelo clasificador.
- Generar un conjunto de datos con información real para el desarrollo del modelo.

1.6. Justificaciones

1.6.1 Justificación Social.

Se puede argumentar que la mejor solución a estas problemáticas sería la de volver entornos de trabajo presencial, con personal de IT encargados de la seguridad para evitar riesgos. Pero esa solución sería beneficiosa solamente en términos de seguridad y no laborales, pues de ser así no se estaría tomando en cuenta los beneficios que trae el trabajo remoto para los empleados. Reducción del estrés, comodidad de trabajo y tiempo mejor administrado son algunas de las ventajas que brinda trabajar desde casa. Incluso áreas laborales que usualmente se desempeñan netamente de manera presencial están dando resultados óptimos de manera remota. Un ejemplo de esto es el área de salud donde gracias a la telemedicina, un doctor puede tratar a

sus pacientes de forma más eficaz y hasta tratar pacientes que antes no podía debido a problemas de distancia. La solución ideal es mejorar la seguridad informática en entornos WLAN.

1.6.2 Justificación técnica.

Las herramientas que se utilizaran en este proyecto son WhireShark para la recolección de los paquetes, Argus para el preparamiento de los mismos, Nmap para generar diversos tipos de escaneo de puertos, y por ultimo diversas librerías Python para el manejo de datos como ser Numpy, Pandas, etc. Se hará uso también de una conexión a internet en una WLAN proveída por la empresa de telecomunicaciones TIGO

1.6.3 Justificación Económica.

Todas las tecnologías de software que se piensan utilizar son gratuitas, así que el costo por parte de ellas es 0 Bs. Para el experimento se cuenta también con 2 computadoras conectadas en una red WLAN a través de un enrutador Arris, por lo que no se invertirá en hardware. Sin embargo el servicio de internet provisto por TIGO tiene un coste mensual de 254 bs. Se hará uso de este servicio no solo para la recolección y prueba de datos, sino durante los 12 meses de investigación, por lo que el costo aproximado sería de 3048 bs.

1.6.4 Justificación científica.

Independientemente de la problemática del trabajo remoto, existen otros motivos por el cual se quiere implementar este modelo. Un firewall es capaz de detectar ciertos tipos de escaneo de puertos. Uno de los más comunes es un escaneo ping que hace uso del protocolo ICMP.

Deshabilitar escaneos ping es fácil de hacer mediante un firewall, sin embargo hay escaneos que son difíciles de detectar como por ejemplo un escaneo XMAS. Este tipo de análisis se basa en el flag FIN del protocolo TCP para perpetrar el escaneo debido a que los firewall están esperando por paquetes SYN la mayor parte del tiempo, por este motivo los escaneos XMAS envían paquetes FIN esperando ninguna respuesta, lo cual indicaría que el puerto está abierto. Recibir una respuesta RST significa que el puerto está cerrado. Lo que se necesita son métodos más avanzados para detectar este tipo de escaneos, que no se basen en parámetros tan reglas rígidas.

1.7 Límites y alcances

1.7.1 Delimitación temática.

Los tipos de escaneo de puertos que se tomarán en cuenta en la investigación son ataques SYN, conexiones TCP, escaneos UDP y ataques XMAS en una red WLAN, en diversas formas de ejecución.

1.7.2 Delimitación espacial.

La investigación se llevará a cabo en una WLAN con dos equipos, uno realizando escaneos y el otro como víctima. La recolección de los datos se la realizara en la computadora víctima y no desde un nodo central en la red, debido que no se cuenta con el presupuesto para adquirir el hardware especializado que permite esto.

1.7.3 Delimitación temporal.

El tiempo que tomará escanear la red WLAN será aproximadamente de un mes.

1.8 Metodologías de investigación

1.8.1 Metodologías de investigación tradicionales.

Como metodología de la investigación se denomina el conjunto de procedimientos y técnicas que se aplican de manera ordenada y sistemática en la realización de un estudio. En un proceso de investigación, la metodología es una de las etapas en que se divide la realización de un trabajo (Castillero, 2017).

Basándonos en el objetivo, la presente tesis califica como una investigación aplicada, pues se busca minimizar una problemática existente, la falta de seguridad necesaria de una computadora en una WLAN.

Tomando en cuenta el nivel de profundización del objeto de estudio se puede decir que la investigación a realizar es exploratoria, porque la recolección, pre procesamiento y análisis de los datos se aplicarán en un área específica donde la seguridad informática no se profundizó lo suficiente.

Según el tipo de datos empleados se puede decir que la investigación será de tipo cuantitativa, pues se recolectaran diversas variables y características numéricas de la transacción de paquetes para su posterior uso y análisis.

Basándonos en el grado de manipulación de las variables la investigación será de tipo cuasi experimental, pues si bien se puede controlar algunas variables, por ejemplo el tipo de cabecera del protocolo TCP en los ataques a simular, existen otras variables y factores que están fuera de control, como ser la latencia de la red, o el tráfico de paquetes generados por el sistema operativo de la computadora víctima.

Tomando en cuenta el tipo de inferencia en el que se basará la investigación, se puede decir que la metodología es deductiva, pues se aplicarán diversas técnicas y herramientas del estado del arte del aprendizaje automático particularmente al tráfico de paquetes de una computadora en una red WLAN.

1.8.2 Metodologías de diseño

La metodología KDD, o descubrimiento de bases de datos en español, se usa día a día cuando se realizan aplicaciones relacionadas con el aprendizaje automático o aprendizaje profundo.

1.9 Aportes

1.9.1 Aporte teórico

La investigación realizará un análisis minucioso de diversos parámetros y variables que son relevantes para la detección del escaneo de puertos. Estos servirán para cualquier futura investigación que se realice en el área.

1.9.2 Aporte práctico

El producto final de esta tesis es no solo un modelo clasificador ya entrenado para la detección de escaneo de puertos de una computadora dentro de una WLAN, sino también las herramientas y código creado para facilitar la creación de cualquier otro modelo con metodologías diferentes en entornos distintos.

CAPÍTULO II

Marco Teórico

2.1 Capa de transporte

En redes computacionales, la capa de transporte es una división conceptual de métodos en la familia de protocolos de internet y el modelo OSI. Los protocolos de esta capa proveen servicios de comunicación de un host hacia otro para diversas aplicaciones. En esta capa se proveen servicios como comunicación orientada a conexión, confiabilidad, control de flujos y multiplexación (Braden, 1989). La capa de transporte es responsable de entregar datos a las respectivas aplicaciones en la computadora. Esto involucra multiplexación estadística de datos de diferentes procesos de aplicaciones, por ejemplo formando segmentos de datos y añadiendo los puertos de origen y destino en el encabezado de cada uno de estos. Junto con las direcciones IP de origen y destino, los puertos constituyen un socket de red, una dirección de identificación de la comunicación entre ambos procesos (Braden, 1989). Los servicios de los protocolos de transporte pueden ser los siguientes:

- **Comunicación orientada a la conexión**: Para una aplicación, es más fácil interpretar una conexión como un flujo de datos que tener que lidiar con modelos de conexión que no manejan estados.
- **Entrega en el mismo orden**: La capa de red no siempre garantiza que los paquetes de datos llegarán en el mismo orden que fueron enviados, como usualmente es requerido.
- **Confiabilidad**: Muchos paquetes pueden perderse durante una transferencia de un host a otro, debido a congestión en la red u otros factores. Por medio de un código de detección

de errores, como la suma de verificación, los protocolos de transporte pueden verificar si los datos no están corruptos o incompletos.

- **Control de flujos**: A veces, la velocidad de transmisión entre dos nodos debe ser administrada para prevenir que un emisor envíe más datos de los que el receptor puede manejar. Esto se logra proveyendo un mecanismo al receptor que le permite controlar la velocidad de transmisión.
- **Evitar congestiones**: Se pueden evitar congestiones tratando de evitar sobre subscripciones de cualquiera de los nodos intermediarios y realizando un control de flujos.
- **Multiplexado**: Los puertos pueden proveer puntos finales de comunicación múltiples en un solo nodo. Por ejemplo, el nombre en una dirección postal es un tipo de multiplexado, y distingue diferencias entre distintos receptores de la misma ubicación.

La figura 2.1 muestra a la interacción de la capa de transporte con las demás capas, algo que también se conoce como encapsulación:

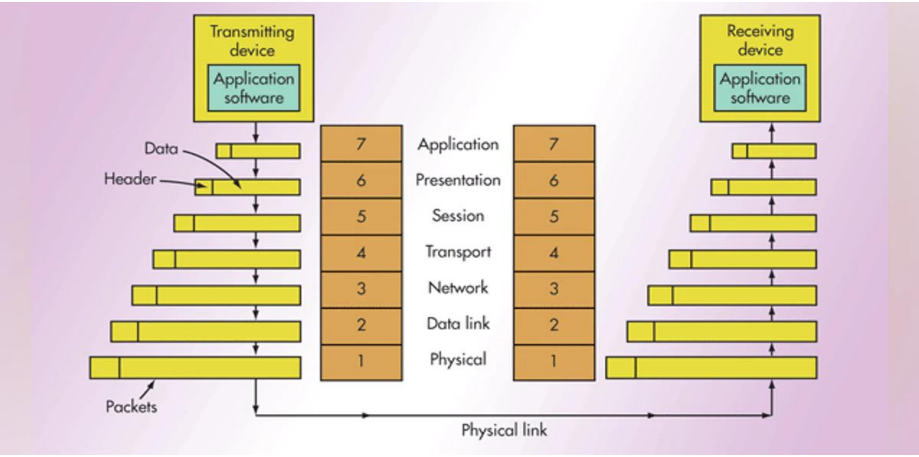


Figura 2.1 Encapsulación entre diferentes capas

2.2 Puertos lógicos

En telemática, un puerto es un punto para la entrada o salida de datos. Un puerto es identificado para cada combinación de una dirección IP y un protocolo en la capa de transporte o capa 4 del modelo OSI con un número de 16 bits, conocido como número de puerto, por lo que una computadora puede tener puertos que van desde 0 a 65535 (Datacademia, 2020).

Los puertos son utilizados por los procesos que se ejecutan en la computadora. Un proceso asocia sus canales de entrada y salida a través de un socket de internet, asociado con un protocolo de transporte, una dirección IP y un número de puerto. Esto se conoce como enlazamiento. Luego, el proceso usa un socket para enviar y recibir datos a través de la red. El software de red del sistema operativo tiene la tarea de transmitir datos de salida de todos los puertos de aplicaciones hacia la red, y distribuir paquetes de red entrantes a los procesos de acuerdo la IP del paquete y el número de puerto del socket. En el caso de TCP, solo un proceso puede ligar una dirección IP y puerto en específico. Los fallos de aplicación comunes, algunas veces llamados conflictos de puerto, ocurren cuando múltiples programas intentan usar el mismo puerto en la misma dirección IP con el mismo protocolo (Cotton y Eggert, 2011). Se puede categorizar a los puertos en de la siguiente manera:

- Los puertos del 0 al 1023 son **puertos del sistema**. Son empleados para uso interno del sistema o programas ejecutados por usuarios privilegiados, un ejemplo sería el puerto 80 para HTTP
- Los puertos del 1024 al 49151 son los **puertos registrados**. Estos fueron registrados por compañías u otros usuarios con la Corporación de Internet para el Asignado de Nombres

y Puertos, denotado como ICANN, para su uso en distintas aplicaciones. Como ejemplo se tiene el uso del puerto 8080 para el protocolo HTTP para tráfico web.

- Los puertos del 49152 al 65535 son los **puertos dinámicos**. Un puerto que puede ser utilizado por cualquier aplicación dentro de la computadora.

Los protocolos de transporte más comunes que utilizan estos números de puertos son el protocolo de control de transmisión y el protocolo de datagramas de usuario, denotados como TCP o UDP respectivamente (Datacademia, 2020). La tabla 2.1 muestra algunos puertos conocidos:

Tabla 2.1
Puertos del sistema, o puertos bien conocidos, populares

Puerto	**Protocolo**
20	Protocolo de transferencia de archivo (FTP), transferencia de datos
21	Protocolo de transferencia de archivo (FTP), control de comandos
22	Seguridad shell (SSH)
23	Telnet
25	Protocolo de transferencia de correo simple (SMTP)
53	Sistema de nombres de dominio (DNS)
67, 68	Protocolo de configuración dinámica de host (DHCP)
80	Protocolo de transferencia de hipertexto (HTTP)
110	Protocolo de oficina de correo (POP3)
119	Protocolo de transferencia de noticias de red (NNTP)
123	Protocolo de tiempo de red (NTP)
143	Protocolo de acceso a mensajes de internet (IMAP)
161	Protocolo de administración de red simple (SNMP)
194	Internet Relay Chat (IRC)
443	HTTP con TLS/SSL (HTTPS)

2.3 Protocolo de control de transmisión (TCP).

El protocolo de control de transmisión TCP es uno de los principales protocolos de la Familia de protocolos de internet. Funciona con el protocolo de internet (IP), quien define como las computadoras envían paquetes de datos entre ellas. TCP se caracteriza porque verifica si existen datos perdidos en la comunicación, y de ser ese el caso los vuelve a enviar, lo que causa latencia, pero garantizando la entrega de datos de forma completa entre el servidor y el cliente (Rouse, 2020). Otra característica importante de TCP es que para iniciar una transferencia de paquetes se realiza un establecimiento de conexión ("handshaking" en inglés).

2.3.1 Características y funcionamiento

Una aplicación no necesita saber los mecanismos particulares para enviar datos via link a otro host, como fragmentación IP para acomodar la unidad máxima de transmisión del medio de transmisión. En niveles más bajos de la pila de protocolos, debido a congestión de la red, balanceamiento de carga del tráfico, o comportamiento de red impredecible, los paquetes IP pueden perderse, duplicarse, o ser entregados fuera de orden. TCP detecta estos problemas, solicita re-transmisión de datos perdidos, re-acomoda los datos fuera de orden e incluso ayuda a minimizar la congestión en la red (Comer, 2006). TCP es usado extensamente por muchas aplicaciones de internet, incluyendo la red informática mundial (WWW), email, FTP, Seguridad de Shell, etc. TCP está optimizado para una entrega precisa en lugar de una entrega rápida, por lo que puede tener tiempos de espera largos (en el orden de segundos) mientras espera por mensajes fuera de orden o retransmisión de mensajes perdidos. Por lo tanto, no es ideal para aplicaciones en tiempo real (Comer, 2006). TCP es un servicio de entrega confiable que garantiza que todos los bytes recibidos serán idénticos y estarán en el mismo orden que los enviados. Como la

transferencia de paquetes en muchas redes no es confiable, TCP garantiza la transferencia usando una técnica conocida como reconocimiento positivo con retransmisión. Esto requiere que el receptor responda con un mensaje de reconocimiento (ACK) cuando recibe datos (Comer, 2006). El emisor maneja un registro de cada paquete que envía y mantiene un temporizador de cuando un paquete fue enviado. El emisor retransmite un paquete si el temporizador expira antes de recibir el reconocimiento. El temporizador es necesario en el caso de que un paquete se pierda o corrompa (Comer, 2006). Mientras IP maneja la entrega real de los datos, TCP realiza un control de los segmentos; las unidades de datos individuales en las que se divide un mensaje para un enrutamiento eficiente a través de la red (Comer, 2006). Por ejemplo, cuando un archivo HTML es enviado desde un servidor web, la capa de software TCP de ese servidor divide el archivo en segmentos y los dirige individualmente a la capa de internet en la pila de red. La capa de internet encapsula cada segmento TCP en un paquete IP añadiendo un encabezado que incluye (entre otros datos), la dirección IP del destino. Cuando el programa del cliente los recibe, el software TCP en la capa de transporte re ensambla los segmentos y se asegura que están correctamente ordenados y libre de errores mientras envía el contenido a la aplicación receptora (Comer, 2006).

2.3.2 Estructura de un segmento TCP

El Protocolo de control de transmisión acepta datos de un flujo de datos, los divide en trozos y agrega un encabezado así creando un segmento o paquete TCP. Un segmento TCP consta de un encabezado de segmento y una sección de datos (Comer, 2006). El encabezado del segmento contiene 10 campos obligatorios y un campo de extensión opcional (Opciones). La sección de datos sigue al encabezado y son los datos de carga útil transportados para la aplicación. La longitud de la sección de datos no se especifica en el encabezado del segmento; Se puede calcular restando la longitud combinada del encabezado del segmento y el encabezado IP

de la longitud total del datagrama IP especificado en el encabezado IP. La figura 2.2 detalla las partes de un paquete TCP.

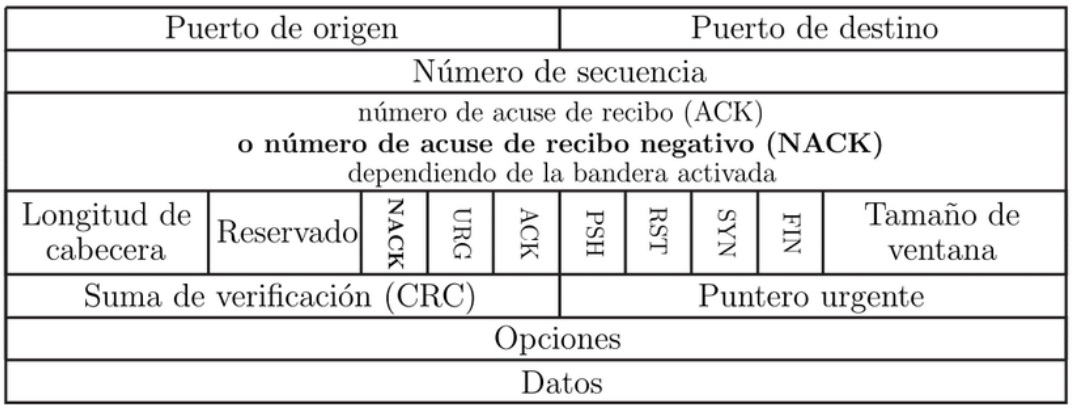


Figura 2.2. Estructura de un paquete o segmento TCP

2.2.3 HandShaking

Para iniciar una conexión TCP tiene un proceso llamado "3-way handshaking" como se ve en la figura 2.3. Mediante estos pasos se establece una conexión acordada entre el cliente y el servidor:

1. **SYN**: El cliente envía un paquete con el flag SYN al servidor. El cliente envía también un número de secuencia de segmento aleatorio x.
2. **SYN-ACK**: En respuesta, el servidor responde con un paquete con flag SYN-ACK. El número enviado por el cliente se devuelve con un incremento de 1 i.e. x+1, y por su parte, el servidor también envía su propio número aleatorio al cliente, i.e y.
3. **ACK**: Finalmente, el cliente envía el flag ACK, junto con el número de secuencia de segmento que el servidor instantánea incrementado en 1 i.e. y+1.

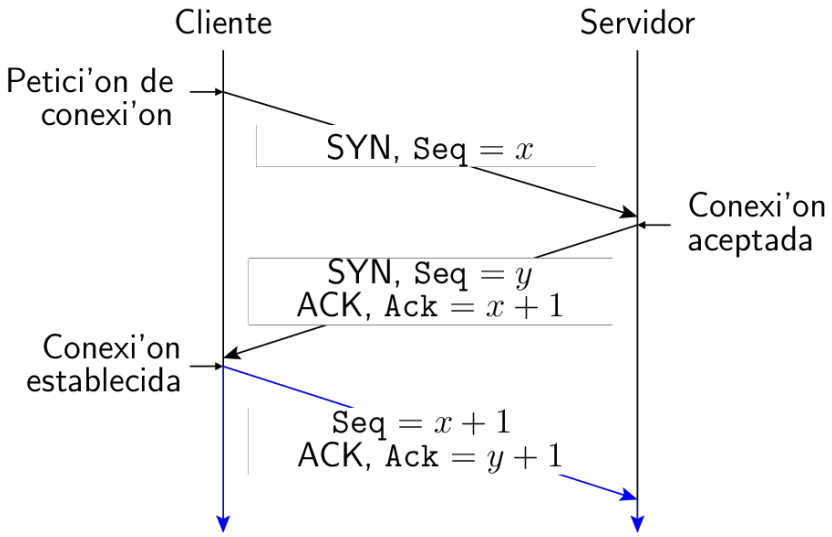


Figura. 2.3. Establecimiento de conexión o "handshaking" del protocolo TCP

2.4 Protocolo de datagramas de usuario (UDP).

El protocolo de datagramas de usuario UDP es uno de los miembros principales de la familia de protocolos de internet. Con UDP, las aplicaciones de computadora pueden enviar mensajes, llamados datagramas, a otros hosts en una red IP. Este protocolo no establece ningún tipo de comunicación previa para la transmisión de datos. UDP es utilizado para propósitos donde la verificación y corrección de errores no son necesarias o son manejadas por la aplicación; UDP evita la sobrecarga de estos procesos en la pila de protocolos. Aplicaciones como streaming o juegos en linea usan UDP frecuentemente porque es preferible tener paquetes perdidos a esperar que estos sean enviados de nuevo por el emisor (Kumar, 2020). Numerosas aplicaciones clave de internet usan UDP. Algunos atributos de UDP hacen que sea un protocolo adecuado para ciertas aplicaciones son:

- Está orientado a transacción, ideal para protocolos de consulta-respuesta simple como DNS o NTP.
- Provee datagramas, que pueden ser usados para modelar otros protocolos como túneles IP o llamadas de procedimiento remoto.
- Es simple, ideal para propósitos que no requieren una pila de protocolos como ser DHCP y TFTP.
- No maneja estado, lo cual beneficia en el manejo de muchos clientes, como aplicaciones de streaming.
- La ausencia de tiempo de espera de retransmisiones lo hace ideal para aplicaciones de tiempo real como juego en línea y muchos protocolos que usan RTSP.
- Cómo soporta multidifusión, es aplicable para difusión masiva de información como en muchos tipos de servicios de detección y la información compartida como PTP.

2.4.1 Estructura de un datagrama UDP

Un datagrama UDP consiste de un datagrama encabezado y una sección de datos. La figura 2.4 detalla las partes de un datagrama UDP. El encabezado consta de 4 campos, cada uno de 2 bytes (16 bits). El puerto de origen indica el puerto del emisor, y se asume que el puerto al cual responder si se necesita. El puerto de destino indica el puerto del receptor. El campo longitud específica el tamaño en bytes del encabezado y el payload juntos y su longitud mínima es 8 bytes. Este campo tiene un límite teórico de 65535 bytes por datagrama (Clark, 2003). Sin embargo, el límite de datos real, es impuesto por protocolo IPv4 y es de 65508. El campo checksum se utiliza para detectar errores en el encabezado y en los datos. El uso de los campos checksum y puerto de origen es opcional en IPv4. En IPv6 solo el puerto de origen es opcional. La sección de datos es la que sigue al encabezado y es el payload de datos de la aplicación.

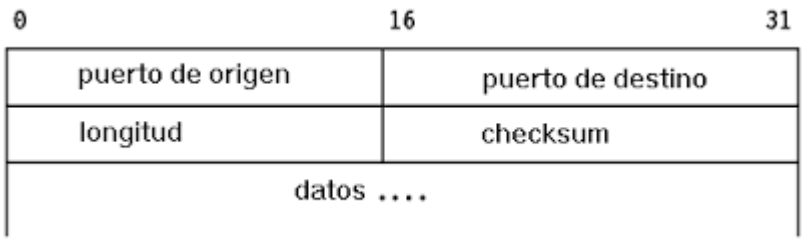


Figura 2.4 Estructura de un paquete o datagrama UDP

2.5 Flujo de red.

En las redes de conmutación de paquetes, el flujo de tráfico, flujo de paquetes, o el flujo de la red es una secuencia de paquetes desde un ordenador de origen a un destino, que puede ser otro host, un grupo de multidifusión o un dominio de difusión. RFC 2722 define el flujo de tráfico como un equivalente lógico artificial a una llamada o conexión (Brownlee, Mills, y Ruth, 1999). RFC 3697 define el flujo de tráfico como una secuencia de paquetes enviados desde una fuente en particular a un destino de unidifusión, unicast o multidifusión particular que el origen desea etiquetar como un flujo (Rajahalme, Conta, Carpenter y Deering, 2004). Un flujo podría consistir en todos los paquetes en una conexión de transporte específica o en un flujo de medios; sin embargo, un flujo no está necesariamente mapeado 1:1 a una conexión de transporte. Una definición más simple es dada en RFC 3917 como un conjunto de paquetes IP pasando un punto de observación en la red durante cierto intervalo de tiempo (Quittek, Zseby, Claise y Zander, 2004).

2.6 Escaneo de Puertos.

El escaneo de puertos es una de las técnicas más populares que los atacantes usan para descubrir servicios que ellos pueden explotar para acceder a sistemas. El escaneo de puertos no es un ataque hostil como tal, sino más bien como el primer paso que un individuo realiza al intentar infiltrarse en una computadora para robar o destruir información (Henry-Stocker, 2018). Este ataque clasifica los puertos de una computadora en tres categorías:

- Abierto: La computadora escaneada responde con un paquete indicando que se está escuchando en ese puerto. También indica que el servicio que fue usado para el escaneo (usualmente TCP o UDP) está en uso.
- Cerrado: La computadora escaneada recibió el paquete pero responde de vuelta indicando que no hay un servicio escuchando en ese puerto.
- Filtrado: Un escaneo en un puerto clasifica a este como filtrado cuando un paquete es enviado pero no se recibió ninguna respuesta. Esto generalmente indica que el paquete enviado ha sido interceptado y rechazado por un firewall.

Si bien existen varias formas de escanear puertos, existen 7 formas bien conocidas (Lyon, 2020):

- **TCP medio abierto o escaneo SYN**: Es una de las técnicas de escaneo de puertos más comunes y populares. Este tipo de escaneo es rápido porque nunca termina el "handshaking" o establecimiento de conexión de TCP. Se envía un mensaje SYN al objetivo, luego este responde con el mensaje SYN-ACK y es ahí donde el emisor termina la transacción no enviando el correspondiente mensaje ACK, pues ya se sabe que el puerto está abierto.

- **Conexión TCP**: Es similar al escaneo SYN solo que emisor de escaneo termina la transacción TCP enviando el correspondiente mensaje ACK al objetivo.
- **Escaneos UDP**: Funciona enviando un paquete vacío o con un payload distinto por puerto, dependiendo del caso de uso. Si el puerto está cerrado, entonces se obtiene una respuesta, en cambio si el puerto está abierto o filtrado, ninguna respuesta es enviada al emisor.
- **Escaneo sigiloso FIN**: Este tipo de escaneo envía un paquete con el flag FIN, esperando así ninguna respuesta si el puerto está abierto, pero si se obtiene una respuesta con el flag RST significa que el puerto está cerrado.
- **Escaneo sigiloso XMAS**: La forma de funcionamiento es similar a la del escaneo FIN, pero con la diferencia que los paquetes se envían con los flags FIN, PSH, y URG.

La herramienta más popular para realizar escaneos de puertos fue y sigue siendo Nmap. Nmap es una herramienta gratuita y de código abierto para la abstracción de información en redes y auditoria de seguridad. Si bien Nmap permite realizar todos los escaneos ya mencionados, también tiene la capacidad de interpretar estos mediante una base de datos de los patrones de respuesta. Es por esto que se puede agregar al menos dos nuevos tipos de escaneos de puertos a los ya mencionados (Lyon, 2020):

- **Escaneo de versión de servicios**: Después que los puertos TCP y UDP son descubiertos usando uno los escaneos ya mencionados, es posible seguir interrogando estos puertos para obtener más información sobre los servicios en funcionamiento. Utilizando pruebas para consultar varios servicios y comparando sus respuestas, es posible obtener el

protocolo (por ejemplo, FTP, SSH, Telnet, HTTP), el nombre de la aplicación (e.g. ISC BIND, Apache httpd, Solaris telnetd), la versión, el nombre del host, etc.

- **Escaneo de sistema operativo**: Funciona enviando una serie de paquetes TCP y UDP y examinando cada bit en las respuestas. Para lograr esto se realizan decenas de pruebas como muestreo de TCP ISN, opciones TCP soportadas, muestreo de IP ID y verificando el tamaño inicial de la ventana, entre otras pruebas. Luego se los compara con una base de datos de más de 2600 respuestas de diferentes sistemas operativos y así escoger los que tengan más coincidencias.

2.7 Aprendizaje automático

El aprendizaje automático es el estudio de algoritmos que mejoran automáticamente a través de la experiencia y el uso de datos. Es visto como un subcampo de estudio de la inteligencia artificial. Los algoritmos de aprendizaje automático construyen un modelo basado en "datos de entrenamiento", para poder realizar predicciones o decisiones sin ser explícitamente programados para hacerlo (Kenyon , 2020). Los algoritmos de aprendizaje automático son usados en una gran variedad de aplicaciones, como ser por ejemplo el filtrado de correo electrónico y visión artificial, donde es difícil o inviable el desarrollar algoritmos convencionales para realizar las tareas necesitadas. Los objetivos del aprendizaje automático son divididos tradicionalmente en tres grandes categorías, dependiendo de la naturaleza de la retroalimentación disponible del sistema de aprendizaje.

- **Aprendizaje supervisado:** Es el proceso de aprendizaje automático de aprender o inferir una función que mapea una entrada a una salida basada en diversas muestras de entrada-salida. En aprendizaje supervisado, cada muestra es un par que consiste de una entrada, generalmente un vector, y un valor de salida deseado (Kenyon, 2020). Un algoritmo de aprendizaje supervisado analiza el conjunto de datos de entrenamiento y produce una función inferida, que puede ser usada para mapear nuevas muestras. En un escenario óptimo el algoritmo podría determinar correctamente el valor de salida para muestras previamente no vistas. En la figura 2.5 se puede apreciar un ejemplo de aprendizaje supervisado, donde las muestras de entrenamiento están previamente clasificadas; por lo que el algoritmo de aprendizaje supervisado encontró una frontera o línea de decisión de color rojo para poder clasificar futuras muestras.

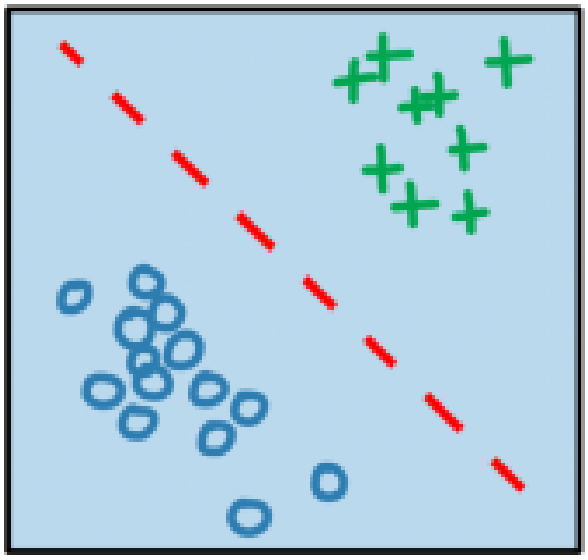


Figura 2.5 Ejemplo de aprendizaje supervisado

- **Aprendizaje no supervisado:** Son tipos de algoritmos que aprenden patrones de datos que no están previamente clasificados. Lo que se espera es que a través de la repetición e imitación, la máquina se vea forzada a construir una representación compacta del universo del cual aprendió (Kenyon, 2020). En contraste con el aprendizaje supervisado, donde los datos de entrenamiento están previamente clasificados, el aprendizaje no supervisado exhibe auto-organización del conjunto de datos y los patrones que se registran ahí. Su habilidad de descubrir similitudes y diferencias en información los hace la solución ideal para el análisis exploratorio de datos, estrategias de ventas cruzadas, segmentación de clientes, y reconocimiento de imágenes. En la figura 2.6 muestra un ejemplo de aprendizaje no supervisado, donde las muestras no están categorizadas, por lo que el algoritmo de agrupamiento busca patrones o conjuntos de datos similares para poder realizar una categorización respectiva.

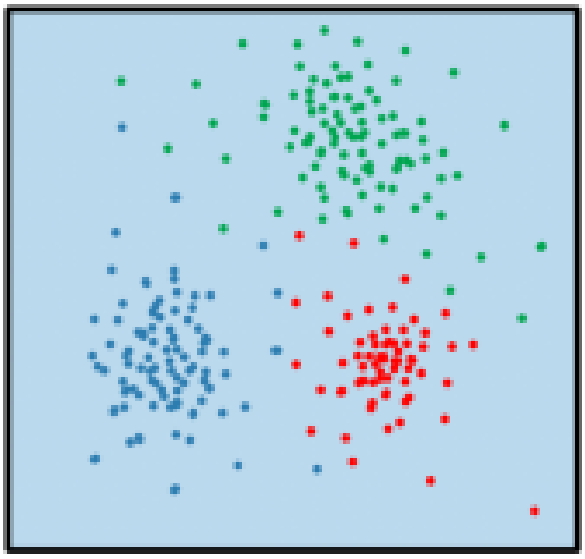


Figura 2.6 Ejemplo de aprendizaje no supervisado

- **Aprendizaje reforzado:** es un área del aprendizaje automático encargada de cómo los agentes inteligentes podrían tomar acciones para así maximizar las recompensas acumuladas en un determinado entorno. El aprendizaje reforzado se diferencia del supervisado al no necesitar pares de entradas y salidas categorizadas (Kenyon, 2020). Su enfoque es encontrar un balance entre la exploración del territorio no explorado y la explotación de conocimiento actual. El entorno es tradicionalmente establecido en la forma de procesos de decisiones de Markov (MPD), porque muchos algoritmos de aprendizaje reforzado en este contexto usan técnicas de programación dinámica. La principal diferencia entre los métodos clásicos de programación dinámica y los algoritmos de aprendizaje reforzado es que los últimos no asumen el conocimiento de un modelo matemático MDP, además de tener como objetivo MDPs grandes donde métodos exactos se vuelven inviables. En la figura 2.7 se tiene un ejemplo de un MDP, con diferentes estados conectados a través de acciones *ai*, donde cada acción tiene una recompensa *R*.

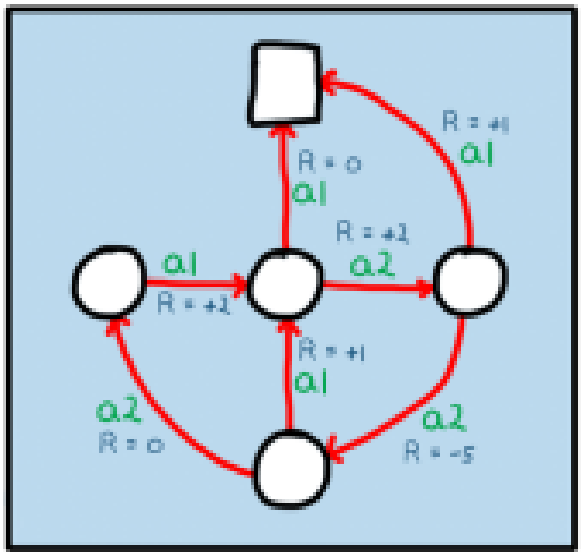


Figura 2.7 Ejemplo de aprendizaje reforzado

2.8 Aprendizaje automático supervisado.

En esencia, el aprendizaje automático se refiere al conjunto de técnicas que ayudan a representar la relación entre un conjunto de variables de entrada o predictores $X = (X1, X2, \ldots Xn)$ y una variable de salida Y a través de una función f (James, Witten, Hastie, y Tibshirani, 2013):

$$Y = f(X) + e$$

Donde f es una función desconocida, pero que puede ser estimada y la variable e es un error aleatorio que es independiente de X y tiene una media de cero. Existen 2 razones principales por las cuales se quiere estimar f: **predicción** e **inferencia**.

2.8.1 Predicción

Usualmente se tiene acceso a los predictores de entrada X, pero la variable de salida Y no puede obtenerse fácilmente. En este escenario, y considerando que los errores aleatorios e tienen una media de cero, se puede predecir Y usando la siguiente ecuación

$$\hat{Y} = \hat{f}(X)$$

Donde $\hat{f}$ representa nuestra **estimación de *f***, y $\hat{Y}$ representa la **predicción resultante de *Y***. En este escenario, $\hat{f}$ a menudo se trata como una caja negra porque uno no suele preocuparse por la forma de $\hat{f}$ siempre y cuando provea predicciones precisas de Y (James et.al, 2013). Por ejemplo, si se quisiera predecir el riesgo de una reacción adversa de un paciente al tomar cierto medicamento, no nos interesa la forma que tenga el modelo sino obtener predicciones precisas en

base a los datos sobre la salud del paciente, o *X*, sobre qué tan riesgoso es dicho medicamento para esa persona**.**

La precisión de $\hat{Y}$ cómo predicción de *Y* depende de dos variables conocidas como error reducible y error irreducible. En el mundo real, $\hat{f}$ no será una estimación idéntica de f y esta imprecisión introduce un error al que se llama reducible por que puede ser potencialmente minimizado usando una técnica de aprendizaje automatizado apropiada. Sin embargo, incluso si se tuviera la función f, la predicción *Y* tendría aún un error de precisión; esto debido a que *Y* es también una función de *e*, que, al ser un error *aleatorio* por definición, no puede ser predicha con *X*.

2.8.2 Inferencia

Muchas veces el interés radica en cómo los cambios en *X1, X2,… Xn* afectan a *Y*. En este caso el objetivo principal no es obtener predicciones de Y, sino entender la relación entre *Y* y *X*, o más específicamente, entender cómo cambia *Y* como función de *X*. En este escenario, *f* no puede ser tratado como una caja negra, porque se necesita conocer su forma exacta (James et al., 2013). Por ejemplo se puede hablar de una compañía que hace publicidad a través de periódicos, radio y televisión para mejorar sus ventas. Aquí, uno podría estar interesado en saber ¿qué medio de publicidad contribuye a las ventas y cuáles no?, ¿Cuál contribuye más? O ¿cuánto incremento en las ventas está asociado a un cierto incremento en la publicidad por televisión?

Como regla general, los modelos lineales son fáciles de interpretar, pero no son tan precisos como modelos no lineales más complejos. En contraste existen modelos no lineales mucho más flexibles que brindan predicciones precisas, pero son mucho menos interpretables. Se puede decir que si el objetivo es inferir el caso de estudio, pues un modelo interpretable es mejor

que uno flexible, pero si lo que se quiere es una predicción precisa, un modelo flexible puede ser un mejor candidato (James et.al, 2013). En la figura 2.8 se puede ver la relación gráfica de diversas metodologías de aprendizaje automático posicionadas de acuerdo a su flexibilidad e interpretabilidad.

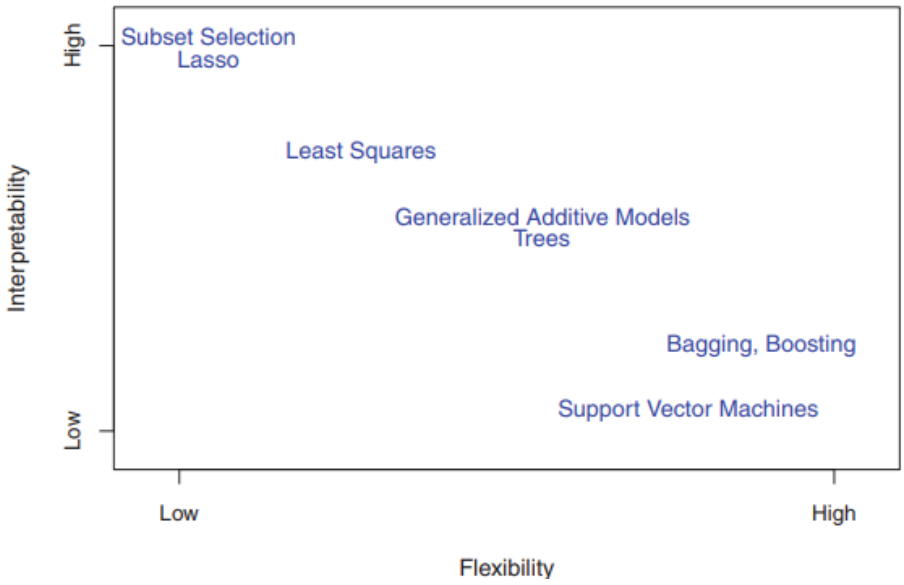


Figura 2.8 Flexibilidad vs interpretabilidad de diferentes métodos de aprendizaje automático

2.9 Árboles de decisión.

Los árboles de decisión son métodos de aprendizaje automático basados en distintos tipos de probabilidades de un conjunto de datos de entrenamiento para crear una estructura de categorización de los parámetros de entrada para poder clasificar futuras muestras. Es una de las técnicas más populares que existen debido a la interpretabilidad que ofrecen (Kubat 2017). Su funcionamiento se basa en cuanta **información** provee un determinado atributo. Son usados en la

toma de decisiones no lineales. Los árboles de decisión tienen un número de ventajas mayor sobre algunos métodos de aprendizaje automático:

- Los árboles son muy fáciles de explicar a las personas. De hecho son más fáciles de explicar que una regresión lineal.
- Algunas personas creen que los árboles de decisión imitan más de cerca la toma de decisiones hechas por humanos que otros métodos de regresión y clasificación.
- Los árboles pueden ser mostrados gráficamente y son fáciles de interpretar incluso por alguien no experto en el área de estudio, especialmente si son pequeños.
- Los árboles pueden manejar predictores cualitativos sin la necesidad de variables de apoyo

Por el contrario, sus entre sus desventajas se pueden encontrar las siguientes:

- Los árboles generalmente no tienen el mismo nivel de precisión de la predicción como otros métodos de regresión y clasificación existentes.
- Adicionalmente, los árboles pueden ser no robustos. En otras palabras, un pequeño cambio en los datos puede causar un gran cambio en el árbol estimado.

La figura 2.9 muestra un ejemplo de un árbol de decisión. Este método de aprendizaje automático es como un grafo a cíclico con nodos representando el lugar donde se escoge un atributo y se hace una pregunta; las aristas son las respuestas a la pregunta y los nodos hoja representan la decisión final de clasificación de acuerdo a la ruta escogida.

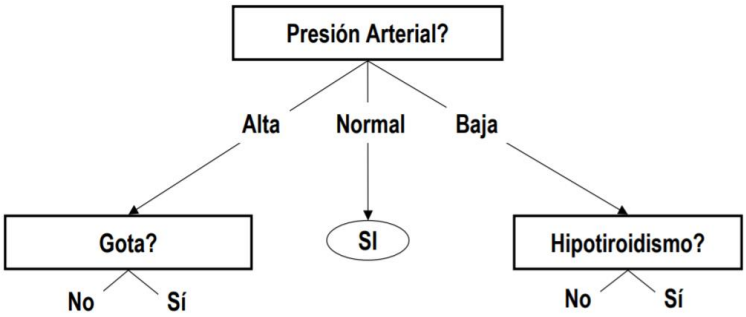


Figura 2.9 Ejemplo de árbol de decisión para determinar el estado de salud de una persona

2.9.1 Proceso de creación

A continuación se procederá a explicar de manera resumida cómo se forma un árbol de decisión cualitativo, precisamente el cómo se determina las variables para los nodos del árbol. Inicialmente se supone que se tiene un conjunto de datos con parámetros de entrada $X_1, X_2, \ldots, X_n$ y un vector *Y* como la categorización de cada muestra. Se define como el conjunto de entrenamiento T una matriz con las columnas X y Y donde t_i es una tupla, una muestra de *T*, con valores $x_1, x_2, \ldots x_n$ y con una clasificación y_i. Se define también $Y_c = \{pos, neg\}$ como el conjunto de posibles valores para cualquier y_i. Si bien es posible que y_i tenga muchos valores, para esta explicación se asume que solo existen las dos ya mencionadas (Kubat, 2017). Con estas variables se puede calcular:

$$p_{pos} = \frac{|pos|}{|T|} \qquad (2.1)$$

Donde p_{pos}indica la probabilidad de que un registro aleatorio t_i sea positivo. De acuerdo a la frecuencia de pos en *T*, se puede definir la información de cada valor en Y_c como:

$$I_{pos} = -log_2(p_{pos}) \quad (2.2)$$

La información I_{pos} es inversamente proporcional a p_{pos} . Si la gran mayoría de los registros en *T* son positivos, la probabilidad p_{pos} será cercana a uno, y la información I_{pos} será cercana a cero; en otras palabras, sabiendo que la mayoría de tuplas son positivas, el hecho de escoger una al azar y que sea positiva nos brinda poca información. En cambio sí pocas tuplas en *T* son positivas, la probabilidad p_{pos} será baja y la información I_{pos}será alta; pues escogiendo una tupla aleatoria y que sea positiva brinda mucha información (Kubat, 2017). La información promedio de este conjunto de datos se la conoce como entropía, definida por:

$$H(T) = -p_{pos} * log_2(p_{pos}) - p_{neg} * log_2(p_{neg}) \quad (2.3)$$

$H(T)$ es la entropía de *T.* Su valor máximo es 1 y lo alcanza cuando p_{pos} = p_{neg} = 0.5. Su valor mínimo es 0 y lo alcanza cuando p_{pos} =1 o p_{neg}= 1. El concepto de entropía ayudará a tratar con la siguiente pregunta: Qué cantidad de información brinda un predictor X_i ? Para saber esto se tiene que dividir *T* en subconjuntos $T_i$donde cada subconjunto tiene **uno de todos** los posibles valores para X_i. Naturalmente cada subconjunto tiene sus probabilidades p_{ipos} y p_{ineg}(Kubat, 2017). Con base en este conocimiento, la ecuación (2.3) dará $H(T_i)$. Ahora sea $|T_i|$ el número de muestras en T_i. La probabilidad que una muestra aleatoria de *T* esté en T_i seria:

$$P_i = \frac{|T_i|}{|T|} \quad (2.4)$$

Con (2.4) ya se tiene lo necesario para calcular la entropía de *T* en relación a un predictor X_i:

$$H(T, X_i) = \sum_i P_i * H(T_i) \quad (2.5)$$

El resultado obtenido es la entropía del sistema donde no solo Y es conocido, sino los valores de X_i son conocidos para cada muestra en *T*. La cantidad de información contribuida por X_i es la diferencia entre la entropía antes de que se considere X_i y la entropía después de que se considera X_i .

Existe otro criterio de decisión aparte de la entropía mostrado en (2.3) y es el índice Gini definido en la ecuación (Aznar, 2020):

$$Gini = 1 - \sum_i p_i^2 \quad (2.6)$$

Los pasos restantes para la bifurcación del árbol es similar al mostrado en las ecuaciones (2.4) y (2.5), donde el rango de valores que el índice *Gini* abarca es de 0 a 0.5. Si bien ambos criterios de decisión demostraron óptimos al crear el árbol, a veces el índice Gini es preferido sobre la entropía por cuestiones de tiempo de ejecución al entrenar el modelo, pues al no utilizar logaritmos para ser calculado es ligeramente más rápido que la entropía (Aznar, 2020).

2.10 Bagging

A veces los árboles de decisión pueden sufrir de mucha varianza. Esto significa que si dividimos los datos de entrenamiento en dos conjuntos aleatoriamente, y entrenamos un árbol de decisión a ambas mitades, los resultados obtenidos podrían diferir bastante (James et al., 2013). En contraste, un procedimiento con poca varianza podría llevar a resultados similares si se aplica

a distintos conjuntos de datos. Agregación de bootstrap, o también llamado bagging es un procedimiento de propósito general para reducir la varianza de un método de aprendizaje en particular.

Suponiendo que dado un conjunto de n observaciones independientes, $Z_1, \ldots, Z_n$, cada una con varianza σ^2, la varianza de la media Z de observaciones es dada por la ecuación (2.7):

$$\frac{\sigma^2}{n} \tag{2.7}$$

En otras palabras, promediar un conjunto de observaciones reduce la varianza (James et al., 2013). Por lo tanto una forma natural de reducir la varianza y así incrementar la exactitud de la predicción es tomar muchos conjuntos de entrenamiento de la población, construir un clasificador por cada conjunto de entrenamiento, y promediar el resultado de las predicciones o en el caso de necesitar una respuesta cuantitativa, tomar la clase con mayoría de votos entre todos los clasificadores.

En otras palabras, se puede calcular $\widehat{f^1}(x)$, $\widehat{f^2}(x)$, …. , $\widehat{f^B}(x)$ usando B conjuntos de entrenamiento separados, y promediarlos para obtener un solo modelo con baja varianza (James et al., 2013). Cuando se trabaja con datos reales, se puede tomar muestras repetidas del único conjunto de entrenamiento disponible dividiendo el conjunto de entrenamiento en B subconjuntos distintos, luego se entrena un clasificador por cada subconjunto y finalmente se promedia todas las predicciones como se ve en la ecuación 2.8:

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \quad \hat{f}^{*b}(x) \tag{2.8}$$

Si bien Bagging puede mejorar la predicción para muchos métodos de regresión, es particularmente útil para los árboles de decisiones (James et al., 2013). Simplemente construimos B árboles usando B subconjuntos de entrenamiento, y se promedian los resultados predichos o se escoge la clase con más cantidad de votos. Estos árboles crecen profundamente, y no son podados. Por lo tanto cada árbol individual tiene mucha varianza pero poco bias. Al obtener el promedio de estos árboles se reduce la varianza (James et al., 2013). Bagging ha demostrado brindar mejoras sorprendentes en la predicción al combinar cientos o hasta miles de árboles juntos como un solo modelo.

2.11 Matriz de confusión

En el campo de aprendizaje automático y especialmente el problema de clasificación estadística, una matriz de confusión, también conocida como matriz de error, es una tabla específica que permite una visualización del rendimiento de un algoritmo, típicamente uno de aprendizaje supervisado (Stehman, 1997). Cada fila de la matriz representa las instancias en una clase clasificada, mientras que cada columna representa las instancias de una clase predicha. El nombre viene del hecho que es más fácil ver donde el sistema está confundiendo las clases (clasificando una muestra con una clase errónea). En una matriz de confusión de dos dimensiones existen cuatro variables que muestran el rendimiento del modelo clasificador sobre un conjunto de datos de prueba como se ve en la tabla 2.2

Tabla 2.2.
Matriz de confusión

	Predicciones negativas	**Predicciones positivas**
Muestras negativas	TN	FP
Muestras positivas	FN	TP

Donde las variables en cada casilla de la matriz representan:

- TP (verdaderos positivos): es el número de muestras positivas y que el modelo predijo correctamente como positivas.
- TN (verdaderos negativos): es el número de muestras negativas y que el modelo predijo correctamente como negativas.
- FP (falsos positivos): es el número de muestra negativas y que el modelo predijo erróneamente como positivas.
- FN (falsos negativos): es el número de muestras positivas y que el modelo predijo erróneamente como negativas.

2.11.1 Indicadores de evaluación de rendimiento

La tasa de error de un clasificador, E, es la frecuencia de errores hechos por el clasificador sobre un conjuntos de datos (Kubat, 2017). Es calculada dividiendo el número de errores FP + FN entre el número total de muestras como se ve en la ecuación 2.7:

$$E = \frac{FP + FN}{TP+TN+FP+FN} \tag{2.7}$$

A veces es preferible trabajar con la medida opuesta, es decir la frecuencia de clasificaciones correctas hechas por el clasificador sobre un conjunto de datos, o también conocida como exactitud (Kubat, 2017). Es calculada dividiendo el número de clasificaciones correctas, TP + TN entre el número total de muestras como se ve en la ecuación 2.8:

$$Acc = \frac{TP + TN}{TP+TN+FP+FN} = 1 - E \quad (2.8)$$

Por precisión se habla del porcentaje de positivos verdaderos TP sobre todas las observaciones que el clasificador indicó positivas TP + FP (Kubat, 2017):

$$Pr = \frac{TP}{TP+FP} \quad (2.9)$$

Por precio positivo falso o recall se habla del porcentaje de positivos verdaderos TP sobre todas las observaciones que son realmente positivas TP + FN (Kubat, 2017):

$$Recall = \frac{TP}{TP+FN} \quad (2.10)$$

2.12 Gmean

Al inducir un clasificador en un dominio con donde la representación de clases no está balanceada, a veces se quiere conseguir un rendimiento en ambas clases, positivas y negativas

(Kubat, 2017). En esas circunstancias se usa la media geométrica, gmean, de la precisión clasificación de ambas clases:

$$gmean = \sqrt{\frac{TP}{TP+FN} * \frac{TN}{TN+FP}} \tag{2.11}$$

Se puede observar que el gmean es la raíz cuadrada del producto de dos números; recall en muestras positivas y el recall en muestras negativas. Si estos dos parámetros difieren mucho pues su media geométrica tenderá a ser más pequeña (Kubat, 2017).

2.13 Error fuera de la bolsa

Hay una forma fácil de estimar el error de prueba de un modelo que utiliza bagging, sin la necesidad de realizar validaciones cruzadas. La clave de bagging es que los arboles sean entrenados repetidamente con subconjuntos de datos de entrenamiento. En promedio, cada árbol creado con bagging hace uso de dos tercios de los datos (James et al., 2013). El tercio restante de los datos no usados para entrenar un determinado árbol son conocidos como datos fuera de la bolsa (OOB). Se puede predecir la respuesta para la observación *i* usando los arboles donde en los que esa observación era OOB. Esto resulta en una B/3 predicciones para la observación *i*. Para obtener una predicción final para la observación *i*, se puede tomar una mayoría de votos sobre a cual clase pertenece (James et al., 2013). Esto lleva a una sola predicción OOB para cada uno de los datos de entrenamiento, con lo que se puede calcular un error de clasificación general del modelo. El error OOB resultante es un estimador valido del error de prueba del modelo, porque la respuesta para cada observación *i* es predicha usando solo árboles que no fueron entrenados con esta (James et al., 2013).

2.14 Metodología KDD

El descubrimiento de bases de datos o KDD es el proceso de búsqueda de conocimiento en datos, y enfatiza las aplicaciones de alto rendimiento de métodos particulares de minería de datos. El objetivo de KDD es extraer conocimientos de los datos en el contexto de grandes bases de datos. Esta metodología funciona mediante el uso de métodos de extracción de datos (algoritmos) para identificar lo que se considera conocimiento de acuerdo con las especificaciones de medidas y umbrales utilizando unas bases de datos junto con cualquier pre procesamiento, sub-muestreo y transformaciones necesarias de bases de datos (Lior, 2014). Las etapas de esta metodología son:

- **Recolección de datos**: En la primera etapa se recolectan los datos para su posterior utilización. Estos datos pueden provenir de sensores físicos, bases de datos, conjuntos de datos recolectados de internet, etc.
- **Preparamiento de datos**: Es el proceso de convertir los datos crudos en un conjunto de datos limpio para ser utilizado en las etapas siguientes. Es aquí donde se tratan datos inconsistentes, incompletos o mal formados.
- **Pre procesamiento de datos**: En esta etapa se analizan los datos para identificar diversas pistas, que nos ayuden a decidir qué modelo es el mejor para el propósito.
- **Aplicación de la metodología**: Es aquí donde el conjunto de datos se divide en los conjuntos de entrenamiento y el de pruebas. El primero se utiliza para entrenar en sí el modelo o los modelos candidatos. Con el segundo conjunto se realizan pruebas para

verificar el rendimiento de las predicciones que el modelo realiza con datos nunca antes vistos.

- **Evaluación de los resultados**: En la etapa final se observan los resultados obtenidos para la posterior toma de decisiones. Es aquí donde, si se requiere, se vuelven a los pasos anteriores para tratar de aplicar otro enfoque al análisis de los datos, mejorar el rendimiento del modelo mediante optimización de híper parametrización, etc.

2.15 Diferencia entre modelo y algoritmo en aprendizaje automático

Si bien al hablar de un modelo se piensa en una entidad abstracta que puede ser implementada y/o utilizada de diversas formas, en aprendizaje automático esta palabra adquiere un nuevo significado. Primero, un algoritmo en aprendizaje automático es un procedimiento que funciona con datos para crear un modelo de aprendizaje automático (Brownlee, 2016). Por ejemplo existen algoritmos para clasificación, como los k vecinos más próximos. Se tienen algoritmos para regresión, como regresión lineal, y se tienen también algoritmos para agrupamiento como k medias. Los algoritmos de aprendizaje automático se pueden describir con seudocódigo o fórmulas matemáticas en publicaciones de investigación, libros, etc (Brownlee, 2016).

Un modelo en aprendizaje automático es la salida de un algoritmo de aprendizaje automático que trabajo con datos (Brownlee, 2016). Un modelo representa lo que fue aprendido por el algoritmo, es la salida “fisica” que se obtiene después de utilizar un algoritmo de aprendizaje automático con datos de entrenamiento (Cowley, 2021). Se lo puede considerar un programa con reglas, números, y cualquier otra estructura de datos algorítmica con las que realizara predicciones (Cowley, 2021). Por ejemplo un algoritmo de regresión lineal resulta en

un modelo compuesto de un vector de coeficientes con valores específicos. Un algoritmo de árboles de decisión devuelve un modelo compuesto de un árbol con nodos condicionales y valores específicos. Un algoritmo para una red neuronal devuelve un modelo compuesto de una estructura similar a un grafo con vectores o matrices de pesos con valores específicos. La ecuación 2.12 muestra la relación entre un algoritmo y modelo en aprendizaje automático (Brownlee, 2016).

$$MODELO = ALGORITMO\left(\begin{matrix}DATOS\ DE \\ ENTRENAMIENTO\end{matrix}\right) \tag{2.12}$$

CAPÍTULO III

Marco Aplicativo

3.1 Recolección de datos.

Como ya se ha explicado, un escaneo de puertos funciona mandando múltiples paquetes a computadoras dentro de una red, para recolectar e interpretar las respuestas obtenidas de estas. Para la recolección de estos datos se requieren al menos 2 máquinas dentro una misma red, una que genera diversos tipos de escaneo de puertos, y otra que actúe como receptora como se ve en la figura 3.1.

Figura 3.1 Red inalámbrica de trabajo

3.1.1 Generación de los datos

La generación de estos escaneos se realizó en una máquina Ubuntu 20.04 mediante un script Shell que genera un escaneo aleatorio con el software Nmap cada cierto intervalo de tiempo. Los comandos generados tienen el siguiente formato:

nmap -Pn <scanType> <ip>

Donde:

-Pn: Omite el ping de prueba para verificar que los hosts de destino estén activos.

<scanType>: Selecciona un escaneo aleatorio de entre los siguientes:

-sS: Escaneo SYN o TCP medio abierto

-sT: Escaneo CONNECT o TCP completo

-sU: Escaneo UDP

-sF: Escaneo sigiloso FIN

-sX: Escaneo sigiloso XMAS

-sV: Escaneo de versión de los servicios

-O: Escaneo de versión del sistema operativo

<ip>: La dirección ip destino.

Adicionalmente, se incluyeron parámetros de configuración opcionales en cada escaneo realizado. Esto para generar diversidad en el comportamiento y detalles de las transacciones:

-F: Indica si se debe ejecutar un escaneo rápido. Un escaneo de puertos en una red puede ser lento. Con esta opción se puede indicar a nmap que solo envíe paquetes a los puertos más populares o comúnmente usados y no así a los 1000 puertos que analiza por defecto.

--scan-delay: Indica el tiempo de espera entre paquetes enviados. Por defecto, Nmap tratará de enviar los paquetes de prueba lo antes posible, lo que puede causar que se pierdan muchos paquetes si el host de destino implementa un ratio estricto límite.

Este parámetro es útil también para evadir IDS e IPS basados en un umbral del tiempo entre paquetes para clasificar comportamiento inusual en una red.

--data-length: Indica la cantidad de bytes aleatorios que se enviaran en el payload de cada paquete. Esto causa que el escaneo sea ligeramente más lento a cambio de que sea menos sospechoso.

Es necesario aclarar que muchas de estas opciones solo son aplicadas de acuerdo al tipo de escaneo que se esté realizando. Por ejemplo, cuando se hace un escaneo de versión del sistema operativo, añadir la opción --data-length no tiene efecto alguno porque Nmap necesita manejar el payload para obtener una respuesta precisa. Los logs de un escaneo, como se ve en la figura 3.2, se guardaron en diversos archivos txt.

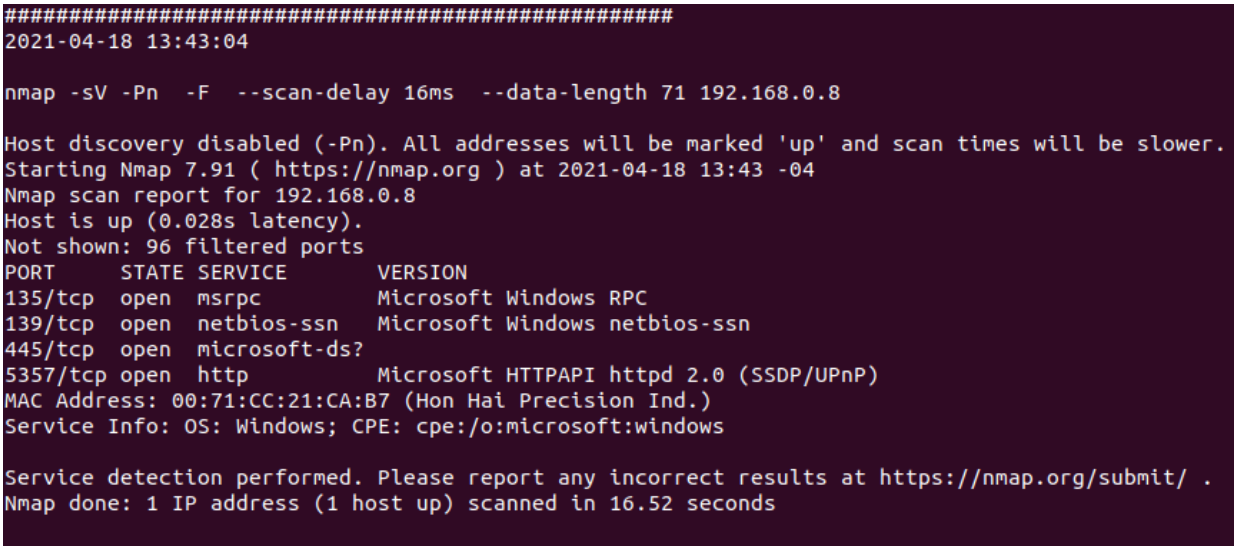


Figura 3.2 Ejemplo de escaneo de servicios y versión.

3.1.2 Captura de los datos

En la computadora de destino se necesita registrar absolutamente todas las transacciones que ocurren en la red, tanto tráfico normal, como los escaneos de puertos que este estaría recibiendo. Para esto se hizo uso del software WireShark. WireShark es el analizador de protocolos más popular que existe actualmente. Es utilizado para realizar análisis y solucionar problemas en redes de comunicaciones, para análisis de datos y protocolos, y como una herramienta didáctica. Todas las transacciones de paquetes son grabadas en archivos .pcap como se ve en la figura 3.3. La recolección de los datos se realizó durante distintos intervalos de tiempo, con una cantidad aproximada de tiempo total de un mes, en donde se capturaron los paquetes de 2 computadoras distintas dentro la misma red, una con sistema operativo Windows 10 y la otra con un dual boot entre Windows 10 y Ubuntu 20.04, con las cuales el usuario estaba realizando sus actividades cotidianas (navegar por internet, jugar juegos en línea, etc).

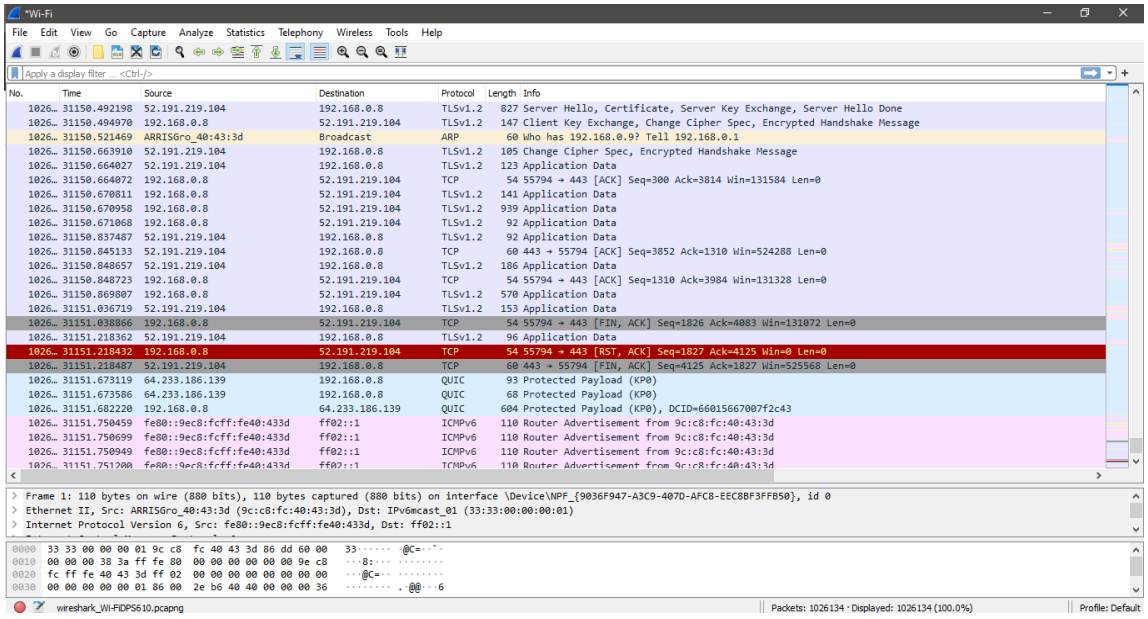


Figura 3.3 WireShark capturando paquetes

3.2 Preparamiento de los datos

Una vez que se tienen suficientes datos recopilados estos deben ser pre procesarlos para que puedan ser leídos por las librerías de aprendizaje automático que se utilizaran. No se puede manejar los archivos pcap directamente, pues como capturaron todo el tráfico de la red en un día cada archivo pcap puede llegar ser realmente pesado, además de que la granularidad de los datos en este estado es muy alta, lo cual no nos permite una interpretación general del comportamiento de los diversos escaneos realizados. Para sacar los atributos más importantes y obtener una versión resumida de cada transacción, se hizo uso de la herramienta Argus.

3.2.1 Manipulación de archivos pcap

Argus es un sistema y aplicación de monitoreo del tráfico de una red desarrollado por Carter Bullard en los principios de los 1980s en el Georgia Tech, y fue adoptado por propósitos de ciberseguridad en el instituto de Ingeniería de Software Carnegie Mellon hacia finales de los 80. La tecnología de monitoreo de flujos de red se ha vuelto una parte crítica de la ciberseguridad moderna y Argus está siendo usado en algunas de las más importantes redes en el mundo. A diferencia de WireShark, Argus permite leer un archivo pcap y extraer información resumida y detallada en la forma de flujos de red. Argus está compuesta de un sensor de flujos de red de procesamiento de paquetes, que genera archivos ARGUS, y una colección de programas de procesamiento de estos archivos en la forma de diversos clientes Argus. En la figura 3.4 se ve un ejemplo de la diferencia de la información abstraída con Argus en comparación a un analizador de paquetes clásico como tcpdump.

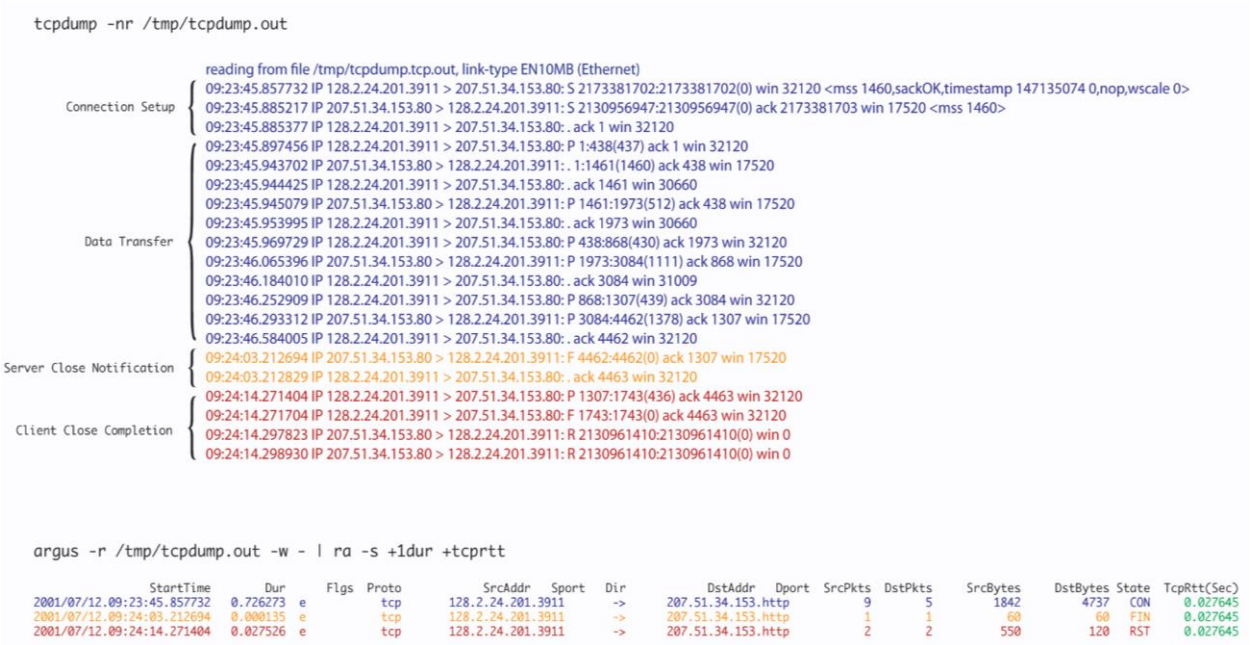


Figura 3.4 Transacción de paquetes resumida a un flujo de red

Por defecto, cuando se quiere mostrar los flujos de red, el cliente de Argus muestra solo algunas propiedades de la transacción de paquetes por defecto. Esto puede ser configurado a través de un archivo rarc donde se puede especificar qué propiedades de la transacción de paquetes se quiere ver resumida en los flujos de red. Los campos relevantes que nos interesan obtener de un flujo de red son:

- **Time:** Es el tiempo en el cual se inició y término la transacción. Por defecto Argus solo imprime el tiempo de inicio.
- **Flags:** Es un indicador con un tamaño fijo. Este reporta varios identificadores del flujo, protocolo, estados y atributos. Este campo es de longitud 8 donde cada carácter puede tomar uno de múltiples valores. Abajo se tiene la lista de sus posibles valores:

- El primer campo indica información general del flujo de red, los posible valores son:

  T - Tiempo corregido/ajustado

  N - Data originada de un flujo de red

- En el segundo campo se tiene información sobre el encapsulamiento de los paquetes:

  * - Encapsulaciones sub-ip múltiples

  e - Flujo Ethernet encapsulado

  E - Encapsulación ERSPAN

  M - Múltiple direcciones Mac vistas

  m - Flujo MPLS encapsulado

  l - Flujo LLC encapsulado

  v - Encapsulación/etiquetas 802.1Q

  w - Encapsulación inalámbrica 802.11

  p - PPP sobre flujo encapsulado Ethernet

  i - Flujo ISL encapsulado

  G - Encapsulación GRE

  a - Encapsulación AH

  P - Encapsulación de túnel IP

  6 - Encapsulación de túnel IPv6

  H - Encapsulación HDLC

  C - Encapsulación Cisco HDLC

A - Encapsulación ATM

S - Encapsulación SLL

F - Encapsulación FDDI

s - Encapsulación SLIP

R - Encapsulación ARCNET

- En el tercer campo se tiene información del protocolo ICMP, con los posibles valores:

  I - Eventos ICMP mapeados a este flujo

  U - Evento ICMP inalcanzable mapeado a este flujo

  R - Evento ICMP redirigido mapeado a este flujo

  T - Tiempo ICMP excedido mapeado a este flujo

- En el cuarto campo brinda información sobre el estado de pérdida y transmisión o anomalías en la transacción de paquetes.

  * - Src and Dst pérdida/retransmisión

  s - Src pérdida/retransmisión

  d - Dst pérdida/retransmisión

  g - Se observaron brechas en los números de secuencia

  & - Paquetes de Src y Dst fuera de orden

  i - Paquetes Src fuera de orden

  r - Paquetes Dst fuera de orden

- En el quinto campo es utilizado por protocolos basados en TCP para mostrar información sobre congestión en la transacción. También es utilizado para mostrar tipos de supresión del protocolo RTP:

  @ - Cierre de Ventana TCP en src y dst

  S - Cierre de Ventana TCP en src

  D - Cierre de Ventana TCP en dst

  * - Supresión de silencio usada por src y dst (RTP)

  s - Supresión de silencio usada por src

  d - Supresión de silencio usada por dst

- En el sexto campo se utiliza para mostrar información del protocolo de Notificación de Congestión Explícita:

  E - ECN en Src y Dst

  x - ECN en src

  t - ECN en dst

- El séptimo campo brinda información en el caso de existir fragmentación IP

  V - Se vieron fragmentos superpuestos

  f - Se vieron fragmentos parciales

  F - Se vieron fragmentos

- El octavo y último campo es utilizado para mostrar información de las opciones IP:

O	-	Se detectó múltiples opciones de ip
S	-	Opción IP de ruta de origen estricta
L	-	Opción IP de ruta de origen flexible
T	-	Opción IP de marca de tiempo
+	-	Opción IP de seguridad
R	-	Opción IP de grabación de rutas
A	-	Opción IP de alerta de enrutamiento
U	-	Opción IP desconocida

- **TcpOpt:** Son las opciones TCP vistas al inicio de la conexión. consiste de un campo de longitud fija que reporta la presencia de cualquier opción TCP que Argus detectó. Sus posibles valores se ven en la tabla 3.1:

Tabla 3.1
Opciones TCP disponibles

Valor	Descripción
M	Tamaño máximo de segmento
w	Escala de ventana
s	SACK requerido aceptado
S	SACK requerido
e	Eco TCP
E	Respuesta de eco TCP
T	Habilita TCP Timestamp
c	Congestión de control (CC)
N	Nueva CC
O	Eco CC
S	Notificación de congestión explícita del origen
D	Notificación de congestión explícita del destino

- **Proto:** Indica el protocolo superior usado en la transacción. Este campo contiene los primeros 4 caracteres del nombre oficial del protocolo usado.
- **SAddr:** Indica de donde se originó la transacción de datos. El valor de este campo depende del protocolo. Para protocolos IP contiene la dirección de origen. Para protocolos TCP y UDP también contiene el número o nombre del puerto.
- **DAddr:** Indica el receptor de la transacción de datos. El valor de este campo depende del protocolo, similar al campo SAddr
- **State:** Indica el estado principal del reporte de la transacción. Es dependiente del protocolo. Para todos los protocolos, excepto ICMP, este campo reporta el estado básico de la transacción.
    - **REQ|INT:** Este valor indica que este es el reporte de estado inicial para la transacción. Para TCP el valor es REQ e indica que una conexión está siendo requerida. en UDP es INT.
    - **ACC:** Indica que una transacción ha sido detectada entre dos hosts. En TCP indica que una solicitud de conexión ha sido respondida y que la conexión será aceptada. Para protocolos no orientados a conexión, este campo indica que existió una transacción de un paquete entre dos hosts.
    - **EST|CON:** Este campo indica que la transacción en cuestión sigue activa y ha sido establecida o está continuando. Para TCP, el estado EST indica que el handshake ha sido completado.

- **CLO:** Específico de TCP, indica que la conexión se cerró con normalidad.

- **TIM:** Indica que no se registró actividad en esta transacción durante un tiempo mayor al límite de tiempo excedido que Argus maneja para ese protocolo.

- **Sum**: Indica la duración acumulada total de grabaciones agregadas
- **Sttl**: Indica el tiempo de vida del origen
- **Dttl**: Indica el tiempo de vida del destino
- **SynAck:** Indica el tiempo entre los paquetes SYN y SYN_ACK.
- **AckDat**: Indica el tiempo entre los paquetes SYN_ACK y ACK.
- **SrcWin**: Aviso de la ventana TCP del origen.
- **DstWin**: Aviso de la ventana TCP del destino.

3.2.2 Manipulación de los flujos de red

En este punto, todas las transacciones de paquetes fueron convertidas a su equivalente en flujos de red, cada archivo .pcap está resumido en un archivo .argus. De estos últimos se imprimió su contenido el cual fue almacenado en archivos .txt donde cada uno de estos archivos tiene la información en el formato establecido en el archivo de configuración rarc. En este punto es necesario el preparamiento de los datos con un script debido a la gran cantidad de flujos de red que se tienen dispersos en varios archivos, además que se necesita aplicar cierto tratamiento a cada flujo antes de poder crear el conjunto de datos como tal.

Como se observó anteriormente, el campo Flags que nos brinda argus se muestra en la forma de una cadena de ocho caracteres donde cada uno representa una propiedad en particular

del flujo (tipo de encapsulación, opciones de IP, etc) y si no aplicaba entonces el respectivo campo es llenado con un campo vacío. Fue necesario dividir estos ocho caracteres en distintas columnas con sus respectivos valores para poder manejar todos estos predictores de manera independiente como se ve en la tabla 3.2. El primer campo de Flag no se considera relevante para este porque solo brinda información de cómo los datos fueron capturados, no del estado de las transacciones como tal.

Tabla 3.2.
Campo Flag convertido a columna

Flag	**Columna**
_ **X** _ _ _ _ _ _	Encapsulación
_ _ **X** _ _ _ _ _	ICMP Info
_ _ _ **X** _ _ _ _	Retransmisión/Perdida
_ _ _ _ **X** _ _ _	Cierre de ventana
_ _ _ _ _ **X** _ _	ECN Info
_ _ _ _ _ _ **X** _	Fragmentación
_ _ _ _ _ _ _ **X**	Opciones de IP

Otro campo que presenta problemas similares es el de TcpOpt, donde cada posición representa una opción TCP en particular en el flujo de la red por lo que se realizó un proceso similar al anterior indicado. También se tuvo que completar o rellenar con valores nulos algunos campos que no aplicaban al protocolo de ese flujo; por ejemplo en transacciones UDP el campo SrcWin está vacío debido a que UDP no tiene un control de congestión. En la figura 3.5 se puede observar el proceso realizado para la obtención del archivo CSV final.

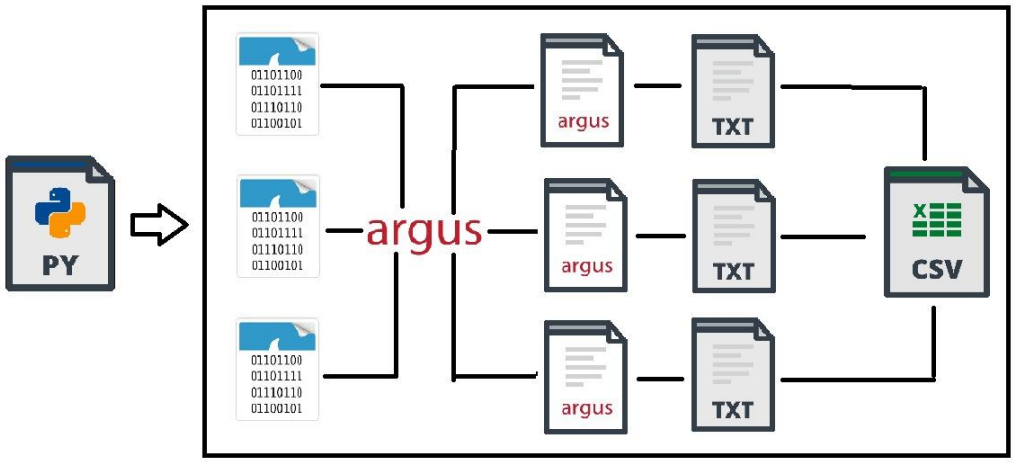


Figura 3.5 Conversión de los archivos .pcap a un archivo .csv

Las columnas generadas en el archivo csv se detallan en la tabla 3.3:

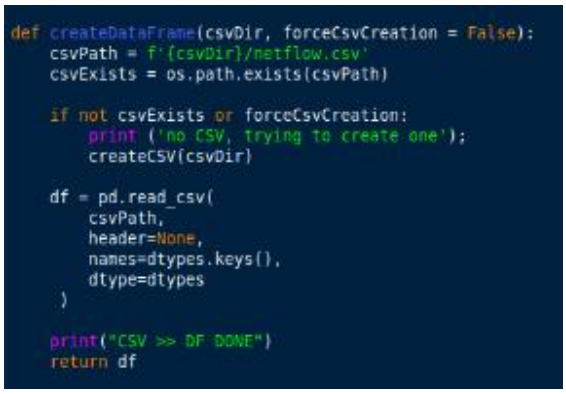
Tabla 3.3.
Columnas del archivo CSV

Predictor	**Descripción**
fecha	La fecha de la transacción
encapsulacion	El tipo de encapsulación de los paquetes
icmp_info	Información de los eventos ICMP
retransmicion_perdida	Información de transmisión y/o pérdidas de paquetes
cierre_de_ventana	Información sobre congestiones en la transacción
ecn_info	Notificaciones de congestión explícitas
fragmentacion	Información de fragmentación en los paquetes
opciones_ip	Opciones de IP registradas
tcp_maximum_segment_size	Tamaño máximo de segmento
tcp_window_scale	Escala de ventana
tcp_selective_ack_ok	SACK requerido aceptado
tcp_selective_ack	SACK requerido
tcp_echo	TCP Echo
tcp_echo_reply	Respuesta de TCP Echo

tcp_timestamp	Habilitación de TCP Timestamp
tcp_cc	Control de Congestión (CC)
tcp_cc_new	Nuevo CC
tcp_cc_echo	Eco CC
tcp_source_ecn	Notificación de congestión explícita del origen
tcp_destination_ecn	Notificación de congestión explícita del destino
protocolo	Protocolo de la transacción
srcaddr_sport	Dirección IP del origen
dstaddr_dport	Dirección IP del origen
srcpkts	Número de paquetes de origen a destino
dstpkts	Número de paquetes de destino a origen
state	Estado final de la transacción
sum	Duración acumulada total de grabaciones agregadas
sttl	Tiempo de vida del origen
dttl	Tiempo de vida del destino
synack_time	Tiempo entre los paquetes SYN y SYN_ACK
ackdat_time	Tiempo entre los paquetes SYN_ACK y ACK
srcwin	Anuncio de congestión de ventana del origen
dstwin	Anuncio de congestión de ventana del destino

3.3 Pre procesamiento de los datos

Este proceso puede ser moroso, pues puede exigir conocimiento de otras áreas relacionadas con la estadística y el área de telemática. Por suerte existen herramientas que facilitan bastante el trabajo realizado en esta etapa. Estas herramientas existen como librerías en el lenguaje de programación Python, pero para poder utilizarlas se tiene que transformar los datos obtenidos a una estructura dataframe. Con el archivo CSV se procedió a la creación del dataframe como se ve en la figura 3.6.



```
def createDataFrame(csvDir, forceCsvCreation = False):
    csvPath = f'{csvDir}/netflow.csv'
    csvExists = os.path.exists(csvPath)

    if not csvExists or forceCsvCreation:
        print ('no CSV, trying to create one');
        createCSV(csvDir)

    df = pd.read_csv(
        csvPath,
        header=None,
        names=dtypes.keys(),
        dtype=dtypes
     )

    print("CSV >> DF DONE")
    return df
```

Figura 3.6 Creación del dataframe

3.3.1 Categorización del conjunto de datos

Los datos en este estado aún no cuentan con una clasificación de cuáles son escaneos y cuales son flujos legítimos, por lo que se utilizó como discriminante la dirección IP del origen de los escaneos y la dirección IP de la víctima agregando la así la variable de salida Y a cada registro del conjunto de datos. La categorización es binaria, con 1 si es un escaneo y 0 si no. En la tabla 3.4 se puede apreciar un ejemplo.

Tabla 3.4
Clasificación de los flujos de red, en base a las IP del origen y destino

Flujo de red				**es_escaneo**
tcp	192.168.0.5.35382	192.168.0.8.domain	REQ	1
udp	192.168.0.8.55813	200.73.96.146.domain	CON	0
arp	192.168.0.4	192.168.0.5	INT	0

3.3.2 Reducción de dimensionalidad

Para mejorar el rendimiento del modelo clasificador se tiene que realizar la eliminación de predictores redundantes o innecesarios bajo el criterio de lo que se está buscando, pues estos pueden perjudicar la creación del modelo. Los predictores eliminados fueron:

- **Fecha:** En el universo de datos en el que se trabajó, la fecha y hora es una variable fuera de control, pues un escaneo de puertos puede ocurrir cuando el atacante lo decida. Es entonces un valor completamente aleatorio.
- **IP de origen**: Un escaneo de puertos puede venir de cualquier dirección IP dentro de la misma red, por lo que esta variable está también fuera de control. Además como los escaneos generados provienen de una única dirección IP, dejar esta variable causaría que el árbol de decisión clasifique en base a la IP lo cual es erróneo.
- **IP de destino:** La dirección IP del destino se debe eliminar por las mismas razones que la IP de origen.

Continuando con el pre procesamiento para poder clasificar datos cualitativos con árboles de decisiones necesario la codificación de estos lo que se conoce como codificación one-hot. La

codificación one-hot es un proceso en el procesamiento de datos que se aplica a datos cualitativos o categóricos, para convertirlos a una representación de varios vectores binarios para luego poder ser utilizados por los algoritmos de aprendizaje supervisado. En el conjunto de datos se pudo identificar 8 columnas cualitativas a las cuales se les aplicó codificación one-hot como se ve en la tabla 3.5:

Tabla 3.5.
Columnas convertidas a binarias

Columna	**Máximo posible de columnas binarias**
ICMP Info	**4**
Retransmisión/Perdida	**7**
Cierre de ventana	**6**
ECN Info	**3**
Fragmentación	**3**
Opciones de IP	**8**
Protocolo	**6**
Estado	**15**

Si bien la columna TcpOpt también es de tipo cualitativa, cada carácter en esta puede tener solo dos valores, presente o ausente, por lo que su conversión a valor cuantitativo fue realizada en el preparamiento de los datos.

El paso que se realizó luego fue el de la eliminación de columnas de las que no se puede abstraer información nueva en lo absoluto. Es el caso de las columnas que tienen el mismo valor

para una gran mayoría de los registros en el conjunto de datos, es decir las columnas con muy poca varianza. Para esto se utilizó un umbral de 0.01 donde todas las columnas con una varianza menor a este umbral son removidas. Luego para evitar redundancia de información en el modelo se procedió con eliminación de predictores correlacionados. La figura 3.7 muestra la matriz de correlación de los datos donde un color amarillo representa un valor cercano a 1, lo que significa una muy alta correlación entre esos dos predictores. El umbral utilizado aquí fue del 0.95.

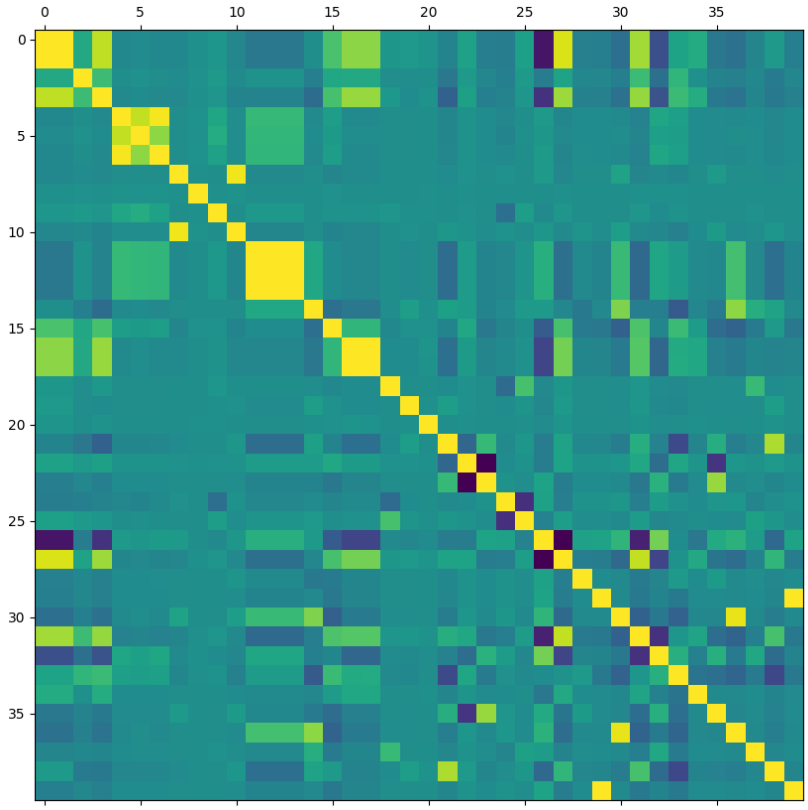


Figura 3.7 Matriz de correlación.

En la tabla 3.6 se puede ver 31 columnas finales del dataframe con las cuales se trabajara.

Tabla 3.6.
Columnas finales para entrenar el modelo

Columnas
tcp_maximum_segment_size
tcp_window_scale
tcp_selective_ack
tcp_timestamp
pkts
srcpkts
srcload
dstload
loss
mean
sttl
dttl
synack_ackdat_time
ackdat_time
srcwin
dstwin
es_escaneo
icmp_info_-
retransmicion_perdida_-
retransmicion_perdida_s
protocolo_arp
protocolo_icmp
protocolo_ipv6-icmp
protocolo_tcp
protocolo_udp
state_CON
state_FIN
state_INT
state_NRS
state_REQ
state_RST

3.4 Aplicación de metodología

En el conjunto de datos se tuvo un total de 667676 registros donde cada tupla representa un flujo de red. El objetivo es saber si un flujo de red es o no un escaneo, por lo que la columna a clasificar es "es_escaneo". En el aprendizaje automático es común que el conjunto de datos se divida en dos subconjuntos, datos de entrenamiento y de prueba. La librería de python scikit-learn proporciona una función que sirve para esta división, precisamente con el método train_test_split. Los parámetros para utilizar esta función son los datos de entrada y el porcentaje de cuántos de esos datos serán para el test. Para el conjunto de datos se tomó un 20% de estos como datos de prueba. En la figura 3.8 se muestran los parámetros de configuración utilizados en la función train_test_split.

```
train, test = train_test_split(df, test_size=0.2)
Y_train = train[constants.ESCANEO]
X_train = train.drop(columns=[constants.ESCANEO]);
Y_test = test[constants.ESCANEO]
X_test = test.drop(columns=[constants.ESCANEO]);
```

Figura 3.8. División del conjunto de datos

La selección del algoritmo de clasificación es uno de los pasos más importantes para resolver el problema, para el presente caso se utilizó el método de bagging con un numero de 50 árboles de decisión. Cada árbol nos permite hacer la clasificación basada en la información brindada por un predictor en particular. Finalmente para entrenar un modelo clasificador usando bagging se hace uso de la función fit. En la figura 3.9 se pueden observar estos dos pasos.

```
model = BaggingClassifier(random_state=0, n_estimators=50, oob_score=True)
model = model.fit(X_train,Y_train)
```

Figura 3.9. Creación de los árboles de decisión y entrenamiento de los datos.

3.5 Modelo clasificador final

Si bien en este punto aún no se realizaron las pruebas para verificar el rendimiento, ya se obtuvo un modelo clasificador con 30 parámetros de entrada (tabla 3.6), que corresponden a distintas propiedades relevantes de un flujo de red. La salida de este clasificador es una variable que indica si el flujo de red respectivo es un escaneo o no.

El producto final, es decir el modelo clasificador, es un archivo “bag.pkl” que posee 50 árboles de decisión sin podar. Para cada muestra nueva, es decir para cada flujo de red capturado, se procesara el flujo para obtener los predictores, relevantes, o parámetros, y en el formato que el modelo clasificador requiere. Luego se lo pasara al modelo clasificador y este devolverá una predicción. El modelo clasificador no recolecta los paquetes directamente de la red. Estos pueden ser recolectados con cualquier herramienta o librería, sin embargo se tiene que proveer al modelo clasificador los mismos parámetros de entrada pre-procesados que se ven en la tabla 3.6 para poder utilizarlo y obtener resultados fiables. La figura 3.10 muestra la estructura del modelo clasificador y su funcionamiento ejecutado en un IDS.

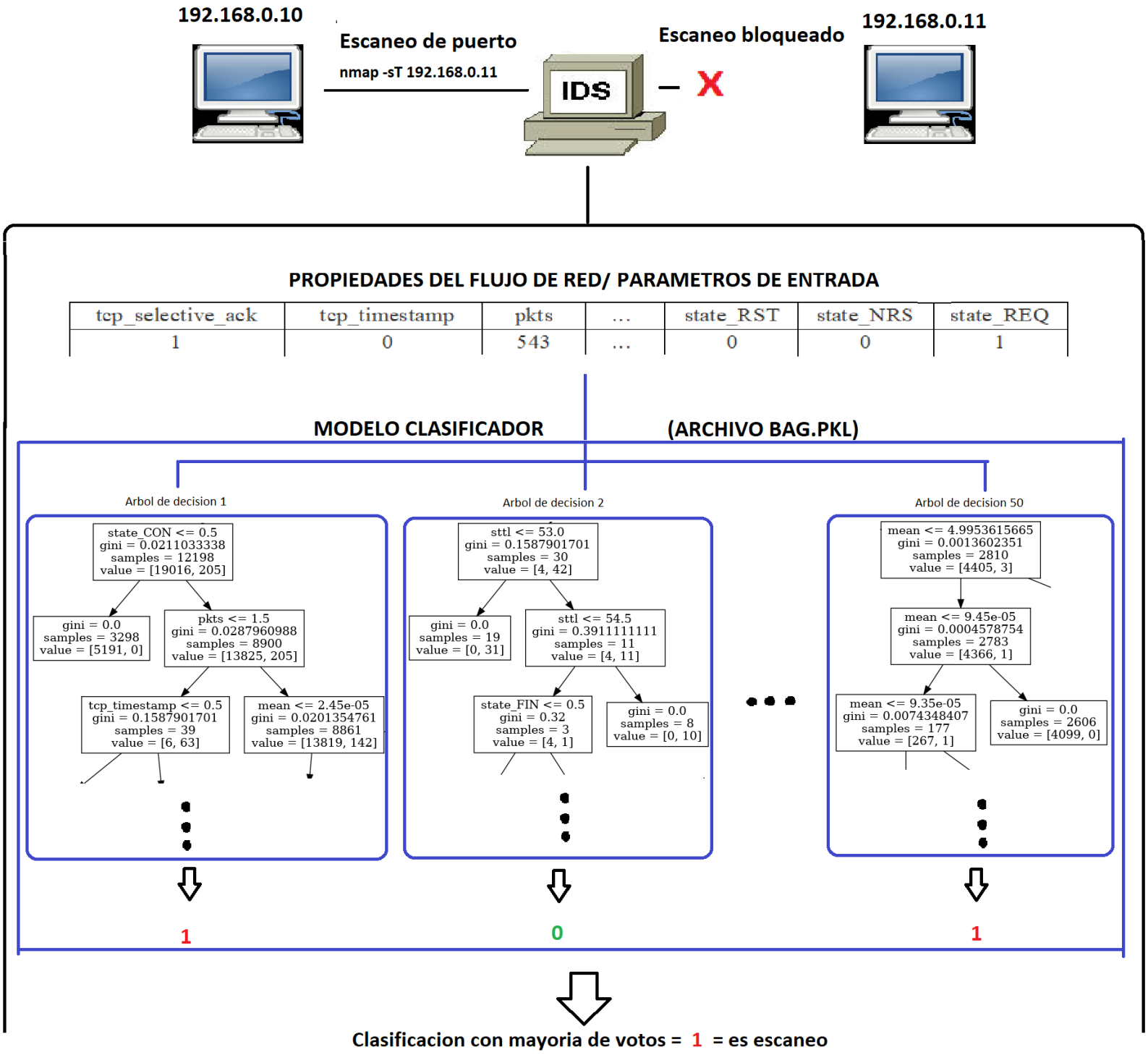


Figura 3.10. Estructura del modelo clasificador en funcionamiento.

CAPÍTULO IV

Análisis de datos y resultados

4.1 Análisis de resultados mediante matriz de confusión

Cuando ya se tiene el modelo entrenado, se procede a verificar su rendimiento con datos previamente no vistos, es decir, datos de prueba sin categorizar. Para el conjunto de prueba se utilizaron 133536 flujos de red, el 20% del total. El modelo va a generar un vector de predicciones, con una clasificación para cada registro en los datos de prueba. Luego para poder visualizar el rendimiento del modelo, se utilizó una matriz de confusión a través de la función de scikit-learn plot_confusion_matrix como se ve en la figura 4.1.

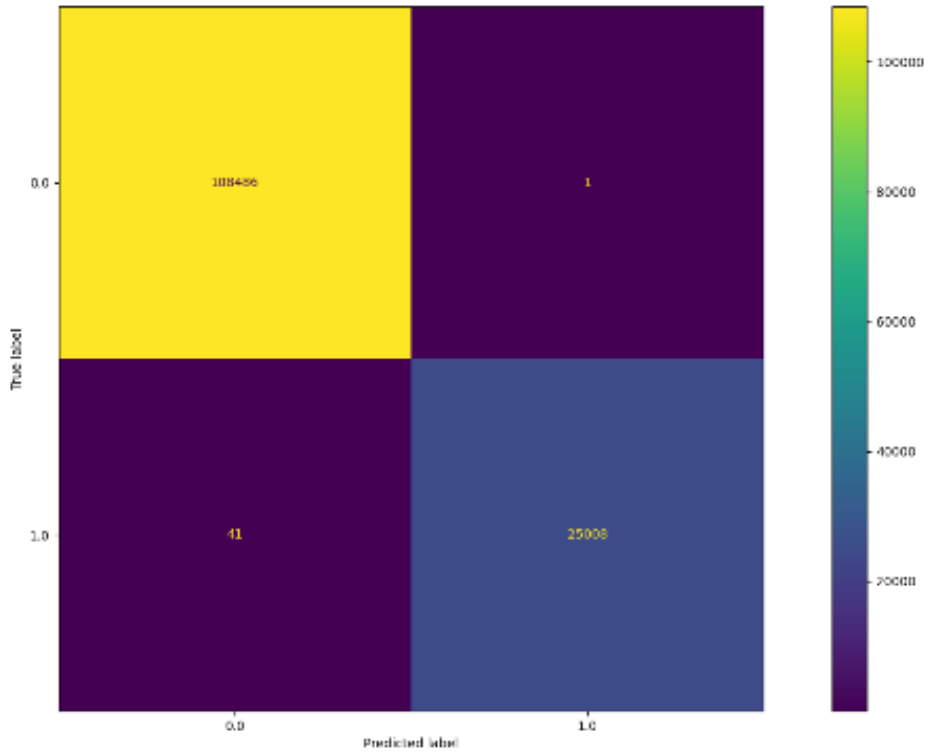


Figura 4.1 Resultados de la matriz de confusión.

Se puede observar una gran mayoría de escaneos de puertos detectados, precisamente 25008 registros que representan el número de verdaderos positivos TP. A su vez unos 108486 de flujos legítimos en la red también fueron correctamente clasificados; este número representa la cantidad de verdaderos negativos TN. Se puede observar también que no todos los flujos de red fueron correctamente catalogados. El número de falsos negativos FN o también el número de escaneos que no fueron detectados es de 41 y el número de falsos positivos FP, flujos de red erróneamente catalogados como escaneos es de 1. Si bien el hecho de que existan flujos que no fueron correctamente clasificados es preocupante, se pueden aplicar pruebas estadísticas de precisión, recall, tasa de error y exactitud para tener una idea de la magnitud que estos números representan. Para obtener la frecuencia de errores realizados por el modelo se usó la fórmula de la tasa de error:

$$\frac{FP + FN}{TP+TN+FP+FN} = E$$

$$\frac{41+1}{25008+108486+41+1} = 0.000315$$

Con la tasa de error se puede obtener la medida opuesta, la frecuencia de clasificaciones correctas hechas por el modelo sobre los datos de prueba.

$$1 - E = Acc$$

$$1 - 0.001333 = 0.99968$$

Se puede apreciar que la tasa de error es prácticamente irrelevante y a su vez la exactitud general del modelo es bastante alta. Dicho de manera simple el 99.968 % de las veces el modelo está catalogando los flujos correctamente y solo comete errores en un 0.0315 %. Analizando más a detalle, para saber la probabilidad de que el modelo está detectando los escaneos correctamente y no está confundiendo flujos legítimos como escaneos se tiene que calcular la precisión:

$$\frac{TP}{TP+FP} = Pr$$

$$\frac{25008}{25008+1} = 0.99996$$

Para saber la probabilidad de que el modelo está detectando los escaneos correctamente y no está dejando pasar muchos escaneos como si no lo fueran, se tiene que calcular el recall:

$$\frac{TP}{TP+FN} = Re$$

$$\frac{25008}{25008+41} = 0.99836$$

Dicho de una forma más sencilla, el 99.996 % de los flujos catalogados como escaneos son verdaderos y el porcentaje de escaneos verdaderos detectados es de 99.836 %. El porcentaje de flujos legítimos confundidos con escaneos es del 0.004 % y el porcentaje de que un escaneo no sea detectado es del 0.164 %.

4.2 Gmean

En el conjunto total de datos se tienen 667676 registros donde solo 125991 de estos son escaneos, es decir solo el 18.87 %, mientras que el 81.13 % restante son flujos de red legítimos. Esto se conoce como un conjunto de datos desbalanceado y en algunos casos podría causar que el modelo tenga preferencia sobre una clase en particular. La prueba de la media geométrica o Gmean ayuda a medir el rendimiento en ambas clases, positivas y negativas, cuando el conjunto de datos no está balanceado. Uno de los aspectos más importantes de la prueba Gmean es que no solo depende del producto de los valores dentro de su raíz sino también de que tanto difieren entre ellos, lo cual indicaría un desbalance en el rendimiento del modelo. Para utilizar esta prueba, se necesita calcular el recall de verdaderos positivos:

$$\frac{TP}{TP+FN} = acc_{pos}$$

$$\frac{25008}{25008+41} = 0.99836$$

Y el recall en verdaderos negativos:

$$\frac{TN}{TN+FP} = acc_{neg}$$

$$\frac{108486}{108486+1} = 0.99999$$

Luego se procede a calcular el Gmean:

$$\sqrt{acc_{pos} * acc_{neg}} = gmean$$

$$\sqrt{0.99836 * 0.99999} = 0.99917$$

Con este resultado se supo que pese a que el conjunto de datos es desbalanceado, el rendimiento en la clasificación de ambas clases no lo es.

4.3 Error fuera de la bolsa

Como cada árbol creado con bagging hace uso en promedio de dos tercios de los datos, se pueden usar los datos fuera de la bolsa (OOB) restantes para pruebas. Al tener 667676 registros, cada árbol entrenado dispone en promedio de un tercio de estos para pruebas:

$$\frac{667676}{3} \approx 222{,}558$$

Para predecir la respuesta para la observación *i*, se utilizó los árboles en los que esa observación era OOB. Esto resulta en un promedio de 50/3 ≈16 árboles que pueden ser usados para predecir la observación *i*. Para obtener una predicción final para la observación *i*, se tomó la mayoría de votos sobre a cual clase pertenece. Esto llevo a una sola predicción OOB para cada uno de los datos de entrenamiento, con lo que se pudo calcular un error de clasificación general del modelo. Al ser una prueba de rendimiento exclusiva del método de bagging, esta es calculada internamente por la librería de sklearn. El resultado obtenido fue:

$$OOB = 0.9997$$

Se puede apreciar que el error fuera de bolsa es bastante óptimo.

CAPÍTULO V

Conclusiones y recomendaciones

5.1 Conclusiones

Durante la investigación se ha seguido un proceso basado en la metodología KDD que guio las diferentes etapas para la obtención del modelo final. El primero fue la generación y recolección de los paquetes en la red, luego se preparó los paquetes para abstraer la información en forma de flujos de red, luego en el pre procesamiento de los datos se eliminaron predictores que no brindaban información alguna, así como se transformaron algunos otros predictores para finalmente implementar del algoritmo de bagging y crear el modelo. En el proceso se utilizaron las herramientas de Nmap, Wireshark y Argus para la obtención y preparamiento de los datos, luego se utilizaron las librerías de Python sklearn, matplotlib, Pandas y Numpy para el pre procesamiento y creación del modelo.

Con los pasos para la elaboración del modelo de la metodología KDD detallados en el capítulo 3 y con los resultados de la matriz de confusión, prueba de Gmean y del error fuera de la bolsa detalladas en el capítulo 4 se demostró la hipótesis: es posible crear un modelo de aprendizaje automatizado para detectar escaneos de puertos en una computadora dentro de una WLAN, lo que a su vez concluye satisfactoriamente el objetivo general de crear un modelo capaz de detectar un escaneo de puertos en una computadora. Precisamente, los objetivos logrados son:

- Con una precisión del 99.996%, un recall del 99.836% obtenida de la matriz de confusión en datos de prueba no vistos, se concluye que el modelo puede detectar si un flujo de red es un escaneo de puertos o no.
- Con una exactitud del 99.968% obtenida en el análisis de la matriz de confusión, se concluye que con el modelo creado minimiza el riesgo que el trabajo remoto conlleva en cuanto a la protección de información.
- Con un error fuera de la bolsa del 0.9997 y el valor de la prueba de Gmean de 0.99917, se puede concluir que la aplicación de las técnicas de aprendizaje automático supervisado detalladas en el capítulo 3 para la creación del modelo fue satisfactoria.
- Al utilizar las herramientas de Nmap para la producción de diversos escaneos de puertos, Wireshark para la captura de datos y Argus para el preparamiento de estos durante el periodo aproximado de un mes como se detalla en el capítulo 3, se cumplió el objetivo de trabajar con datos recolectados reales.

5.2 Recomendaciones

Cuando se quiere recolectar paquetes para su posterior análisis, se recomienda colocar el analizador de paquetes un punto en la red por el que todo el tráfico transite para no tener información parcializada al momento de realizar el análisis. Para esto se requiere hardware especializado, como routers o concentradores que permitan capturar paquetes, los cuales estaban fuere del presupuesto de esta investigación, pero se recomienda utilizarlos si se cuenta con acceso a estos.

Otro de los problemas que se encontró es la obtención de los datos, pues al tratarse de identificar diversas propiedades de un escaneo de puertos a una computadora en una determinada

red, los datos deben ser recolectados de una actividad real y no pueden ser obtenidos de otra forma, lo que conlleva a que la generación y recolección de los datos sea considerablemente lenta. Por eso se recomienda trabajar en redes grandes con varios equipos en ella para recolectar más información de paralelamente y así ahorrar algo de tiempo.

Existen muchas variables que pueden afectar el comportamiento de un escaneo de puertos que dependen plenamente de la red en la que se ejecute y de las computadoras que participen en esta, por lo que si se desea obtener un mayor rendimiento al detectar escaneos en una red en particular se recomienda generar un modelo en la red donde se planea detectar los escaneos. Esto no debería ser un problema pues la presente tesis no solo presenta un modelo clasificador sino también diversas herramientas que realizan la generación de diversos escaneos de puertos, así como la captura, preparado y pre procesamiento de los flujos en la red, como también la creación y pruebas de un modelo.

Bibliografía

Braden, R. (1989). *Requirements for Internet Hosts – Communication Layers*. doi:10.17487/RFC1122. RFC 1122. Recuperado de https://tools.ietf.org/html/rfc1122

Brownlee, J. (2016). *Master Machine Learning Algorithms.* Machinelearningmastery.com. Melbourne, Australia.

Brownlee, N., Mills, C. y Ruth, G. (1999). *RFC 2722 - Traffic Flow Measurement: Architecture*. IETF. Recuperado de https://www.ietf.org/rfc/rfc2722.txt

Butun, I., Morgera, S. y Sankar, R. (2013). *A Survey of Intrusion Detection Systems in Wireless Sensor Networks*. IEEE Communications Surveys & Tutorials. doi: 10.1109/SURV.2013.050113.00191 Recuperado de https://tools.ietf.org/html/rfc6335

Castillero, O. (2017). *Los 15 tipos de investigación (y características)*. Barcelona, España.: Psicología y Mente. Recuperado de https://psicologiaymente.com/miscelanea/tipos-de-investigacion

Clark, M.P. (2003). *Data Networks IP and the Internet, 1st ed*. West Sussex, England: John Wiley & Sons Ltd. USA

Comer, D. (2006). *Internetworking with TCP/IP: Principles, Protocols, and Architecture*. **1** (5th

ed.). Prentice Hall. USA.

Cotton, M. y Eggert, L. (2011). *Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry*. IETF. doi:10.17487/RFC6335. BCP 165. RFC 6335. Recuperado de https://tools.ietf.org/html/rfc6335

Cowley, E. (2021). *What is a machine learning model?*.EU: Documentación oficial de Microsoft. USA: Recuperado de https://docs.microsoft.com/en-us/windows/ai/windows-ml/what-is-a-machine-learning-model

Datacadamia. (2020). *Network - Port (Computer networking)*, EU.: Datacadamia. Recuperado de https://datacadamia.com/network/port

Griffith, E. (2020). *How to Hack Wi-Fi Passwords*. Ithaca, New York, EU.: pcmag. Recuperado de https://www.pcmag.com/

Henry-Stocker, S. (2018). *Using nmap on your home network*. Washington, USA.: Network World. Recuperado de https://www.networkworld.com/article/3314832/using-nmap-on-your-home-network.html

International Labour Organization. (2020). *The impact of the COVID-19 pandemic on jobs and incomes in G20 economies*. Global.: ILO. Recuperado de

https://www.ilo.org/global/docs/WCMS_753607/lang--en/index.html

Jackson, M. (2018). *A Comparison of Home Broadband Router Specs from the Big UK ISPs*. Dorset, UK.: Recuperado de https://www.ispreview.co.uk

James, G. Witten, D. Hastie, T. y Tibshirani, R. (2013). *An introduction to Statistical Learning. Springer*, USA.

Khraisat, A., Gondal, I., Vamplew, P. y Kamruzzaman, J. (2019). *Survey of intrusion detection systems: techniques, datasets and challenges*. SpringerOpen. doi: 10.1186/s42400-019-0038-7

Kumar, V. (2020). *User Datagram Protocol (UDP)*. EU.: geek for geeks. Recuperado de https://geeksforgeeks.org/user-datagram-protocol-udp/

Kurkure, M. (2019). *Survey on Detecting Port Scan Attempts with Combined Analysis of Support Vector Machine and Deep Learning Algorithms*. IOSR Journal of Computer Engineering, 21(3) , 42-46. Recuperado de http://iosrjournals.org/iosr-jce/papers/Vol21-issue3/Series-4/F2103044246.pdf

Marinova, I. (2020). *28 Need-To-Know Remote Work Statistics Of 2020*. USA.: Review 42. Recuperado de https://review42.com/remote-work-statistics/

Monzon, C. (2011). *Auditoría de seguridad de redes inalámbricas de área local wireless local area Network (WLAN)*. Universidad Mayor de San Andrés, La Paz, Bolivia.

Lior, R. (2014). *Data mining with decision trees: theory and applications (Vol. 81)*. World scientific.

Lyon, G. (2020). *Nmap reference guide*. Nmap org Recuperado de https://nmap.org/book/man-author.html

Kenyon M, (2020) *Supervised v. Unsupervised v. Reinforcement Learning: An Introduction*. USA.: The Sharper Dev. Recuperado de https://thesharperdev.com/supervised-v-unsupervised-v-reinforcement-learning-an-introduction/

Pant, A. (2019). *Workflow of a Machine Learning project*. USA.: Towards Data Science. Recuperado de https://towardsdatascience.com/workflow-of-a-machine-learning-project-ec1dba419b94

Quittek, J. Zseby, JT. Claise, B. y Zander, S. (2004). *RFC 3917 - IPFIX Requirements*. IETF. Recuperado de https://www.ietf.org/rfc/rfc3917.txt

Rajahalme, J. Conta, A. Carpenter, B. y Deering, S. (2004). *RFC 3697 - IPv6 Flow Label Specification*. IETF. Recuperado de https://www.ietf.org/rfc/rfc3697.txt

Rouse, M. (2020). *TCP (Transmission Control Protocol)*. East Coast, USA.: TechTarget. Recuperado de https://searchnetworking.techtarget.com/definition/TCP

Singer, D. (2019). *Windows Firewall Basics. Michigan*, EU.: liquidweb. Recuperado de https://www.liquidweb.com/

Viet, N. Nguyen, Q., Thi Trang, L. y Shone, N. (2018) *Using Deep Learning Model for Network Scanning Detection. Acm Digital Library*. doi: 10.1145/3233347.3233379

Villegas Pacasi, L. F. (2009). *IPSVOFSL: SISTEMA INTELIGENTE DE PREVENCIÓN DE INTRUSIONES*. Universidad Mayor de San Andrés, La Paz, Bolivia.

ANEXOS

CODIGO PARA LA CREACION DEL MODELO

ARCHIVO train.py

```
import matplotlib.pyplot as plt
import matplotlib.image as pltimg
import pydotplus
from sklearn.model_selection import train_test_split
from sklearn import tree
from sklearn.ensemble import BaggingClassifier
from utils import preprocesDataFrame
from utils import plotConfusionMatrix
from utils import createCSV
from utils import plotFeatureImportance
from utils import createDataFrame
from utils import categorizeFlows
from utils import filterByIP
from utils import plotCorrelation
from utils import dimensionalityReduction
from bashUtils import createArgusFilesOutput
import constants

commonPaths=open(f'paths.txt', "r").readlines();
trainDataPath=commonPaths[1].strip()


createArgusFilesOutput(trainDataPath)
createCSV(trainDataPath)
df=createDataFrame(trainDataPath)
df[constants.ESCANEO] = categorizeFlows(df)
df=preprocesDataFrame(df)
df=dimensionalityReduction(df)
df=df.sample(frac=1)

print(df)

#### TRAINING THE MODEL  #####
```

```
train, test = train_test_split(df, test_size=0.2)
Y_train = train[constants.ESCANEO]
X_train = train.drop(columns=[constants.ESCANEO]);
Y_test = test[constants.ESCANEO]
X_test = test.drop(columns=[constants.ESCANEO]);


model = BaggingClassifier(random_state=0, n_estimators=50, oob_score=True)
model = model.fit(X_train,Y_train)

joblib.dump(model, 'bag.pkl')


#####  DISPLAY ONE TREE #####
df = df.drop(columns=[constants.ESCANEO]);
print("Y column dropped")
joblib.dump(df.columns.tolist(), 'columns.txt')
print ("out of bag score",model.oob_score_)
print(model.estimators_)
data = tree.export_graphviz(model.estimators_[0], out_file=None,
feature_names=df.dtypes.keys(), precision = 10)
graph = pydotplus.graph_from_dot_data(data)
graph.write_png('mydecisiontree.png')

print("now plotting")
img=pltimg.imread('mydecisiontree.png')
imgplot = plt.imshow(img)


plotConfusionMatrix(model, X_test, Y_test)
plotFeatureImportance(model, X_train)
plt.show();
```

ARCHIVO utils.py

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
import re
import csv
import os
from sklearn.metrics import plot_confusion_matrix
from os import walk
from os import path
from sklearn import tree
from sklearn.feature_selection import VarianceThreshold
import constants
```

```
dtypes={
    constants.FECHA:        "string",
    constants.ENCAP:        "string",
    constants.ICMP:         "string",
    constants.RE_PERD:      "string",
    constants.C_VENTANA:    "string",
    constants.ECN:          "string",
    constants.FRAG:         "string",
    constants.OP_IP:        "string",
    constants.TCP_M:        "int64",
    constants.TCP_w:        "int64",
    constants.TCP_s:        "int64",
    constants.TCP_S:        "int64",
    constants.TCP_e:        "int64",
    constants.TCP_E:        "int64",
    constants.TCP_T:        "int64",
    constants.TCP_c:        "int64",
    constants.TCP_N:        "int64",
    constants.TCP_O:        "int64",
    constants.TCP_S:        "int64",
    constants.TCP_D:        "int64",
    constants.PROTO:        "string",
    constants.SRCADDR:      "string",
    constants.DIR:          "string",
    constants.DSTADDR:      "string",
    constants.PKTS:         "int64",
    constants.SRCPKTS:      "int64",
    constants.DSTPKTS:      "int64",
    constants.STATE:        "string",
    constants.SRCLOAD:      "float64",
    constants.DSTLOAD:      "float64",
    constants.LOSS:         "float64",
    constants.RATE:         "float64",
    constants.MEAN:         "float64",
    constants.STDDEV:       "float64",
    constants.RUNTIME:      "float64",
    constants.IDLE:         "float64",
    constants.TRANS:        "int64",
    constants.SUM:          "float64",
    constants.STTL:         "int64",
    constants.DTTL:         "int64",
    constants.PCR:          "float64",
    constants.TCPRTT:       "float64",
    constants.SYNACK:       "float64",
    constants.ACKDAT:       "float64",
    constants.SRCWIN:       "int64",
    constants.DSTWIN:       "int64"
};


windowsIps = ["192.168.0.8", "192.168.43.196"] #, "192.168.0.5"]
linuxIps = ["192.168.0.5", "192.168.43.203"] #, "192.168.0.8"]
```

```
#######  PREPARE DATA ##########
def addFlagsSeparators(flowStr,start, length, binary = False):
    newFlags=""
    for i in range(start, start+length):
        if(flowStr[i]==' '):
            newFlags+= '0' if binary else '-'
        else:
            newFlags+= '1' if binary else flowStr[i]
        newFlags+=' '
    return newFlags


def createCSV(dataDir, filterCriteria = None):
    csvPath = f'{dataDir}/netflow.csv'
    try:
        os.remove(csvPath);
        print('...deleting prev CSV')
    except:
        var=0


    open(csvPath, "w").close()
    for (dirpath, dirnames, filenames) in walk(dataDir):
        for fname in filenames:
            match = re.search(".+.txt$", fname);
            if match is not None:
                print(fname)
                argusOutputFile = open(f'{dirpath}/{fname}', "r");

                #do nothing with header
                argusOutputFile.readline();

                fileContent = [i.strip() for i in
argusOutputFile.readlines()]

                if filterCriteria is not None:
                    fileContent = list(filter(filterCriteria, fileContent))

                with open(f'{csvPath}', "a") as csvFile:
                    for line in fileContent:
                        # Flags field
                        flags = addFlagsSeparators(line, 27, 7)
                        #TCP opt field
                        tcpOpt = addFlagsSeparators(line, 37, 12, True)
                        line = line[:27] + flags + " " + tcpOpt +
line[37+12:]
                        line=line.split()
                        line[1] = line[0]+" "+line[1]
                        line.pop(0)
                        while (len(line) <= len(dtypes.keys())):
                            line.append(0);
                        writer = csv.writer(csvFile)
                        writer.writerow(line)
    print("txt's >> CSV DONE")
```

```
#######  PREPROCESS DATA ##########


def createDataFrame(csvDir, forceCsvCreation = False):
    csvPath = f'{csvDir}/netflow.csv'
    csvExists = os.path.exists(csvPath)

    if not csvExists or forceCsvCreation:
        print ('no CSV, trying to create one');
        createCSV(csvDir)

    df = pd.read_csv(
        csvPath,
        header=None,
        names=dtypes.keys(),
        dtype=dtypes
     )

    print("CSV >> DF DONE")
    return df


def preprocesDataFrame(df):
    df = df.drop(columns=[
            constants.FECHA,
            constants.SRCADDR,
            constants.DSTADDR,
            constants.DIR
        ])

    df= pd.get_dummies(df,
columns=df.select_dtypes("string").columns.tolist());
    print("get dummies DONE")
    return df


def dimensionalityReduction(df):
    lowVarianceFilter = VarianceThreshold(0.01)
    lowVarianceFilter.fit(df, df[constants.ESCANEO])

    df=pd.DataFrame(data=lowVarianceFilter.transform(df),
                    columns=df.columns[lowVarianceFilter.get_support()])

    print(df.columns.tolist())
    print("low variance columns REMOVED")


    corr  = df.corr()
    correlated=[]
    plotCorrelation(df)
    for i in range(len(corr.columns)):
        for j in range(i):
            if abs(corr.iloc[i, j]) > 0.95:
                colname = corr.columns[i]
                correlated.append(colname)
```

```
    df = df.drop(columns=correlated)
    print("corralated columns REMOVED")

    return df


def categorizeFlows(df):
    es_escaneo = []
    counter = 0
    for row in df.itertuples():
        scan = 0
        for i in range (0, len(windowsIps)):
            if (row[21].find(windowsIps[i])>=0 and
row[23].find(linuxIps[i])>=0):
                scan = 1
                break;
        counter +=  scan
        es_escaneo.append(scan)
    print("data categorization DONE", counter)
    return es_escaneo



#######  PLOT FUNCTIONS ##########
def plotConfusionMatrix(model, X_test, Y_test):
    disp = plot_confusion_matrix(model, X_test, Y_test);
    print(disp.confusion_matrix)

    FN=0
    TP=0
    if len(disp.confusion_matrix) == 2:
        FN = disp.confusion_matrix[1][0]
        TP = disp.confusion_matrix[1][1] if len(disp.confusion_matrix[1]) ==
2 else 0

    TN = disp.confusion_matrix[0][0]
    FP = disp.confusion_matrix[0][1] if len(disp.confusion_matrix[0]) == 2
else 0


    print("Precision: ", (TP/(TP + FP)) if TP+FP > 0 else 1);
    print("Recall: ", (TP/(TP + FN)) if TP + FN > 0 else 1);
    print("Tasa de error: ", (FP + FN)/(TP + TN + FP + FN));
    print("Precision total: ", (TP + TN)/(TP + TN + FP + FN));
    plt.show()


def plotFeatureImportance (clf, X_train):
    f_importances=[]
    if not hasattr(clf, 'feature_importances_'):
        f_importances = np.mean([
            tree.feature_importances_ for tree in clf.estimators_
        ], axis=0)
    else:
        f_importances = clf.feature_importances_
```

```
    feat_importances = pd.Series(f_importances, index=X_train.columns)
    plot = feat_importances.nlargest(len(X_train.columns)).plot(kind='barh')
    plt.show()



def plotCorrelation(df):
    corr=df.corr()
    print(corr)
    plt.matshow(corr)
    plt.show()
```

ARCHIVO bashUtils.py

```
import subprocess
import time
from subprocess import  PIPE, STDOUT

commonPaths=open(f'paths.txt', "r").readlines();
argusConfPath= commonPaths[0].strip()

pwd='unounodostres'

def createArgusFilesOutput (dataDir):
    print("argus >> txt")
    subprocess.run(['bash', f'{argusConfPath}/argus_conversion.sh',
f'{argusConfPath}', f'{dataDir}'])
    print("argus >> txt DONE")


def createArgusDaemonOutput (outputDir):
    subprocess.run(f'echo {pwd} | sudo -S pkill argus', shell=True,
capture_output=True)
    time.sleep(5)
    print("init argus daemon")
    subprocess.run(f'echo {pwd} | sudo -S argus -P 561 -d', shell=True,
capture_output=True)
    time.sleep(5)
    print("start netflows capture")
    p = subprocess.Popen(f'exec ra -F {argusConfPath}/rarc -S localhost >
{outputDir}/realData.txt', shell=True)
    return p
```

ARCHIVO script generador de escaneos

```
#!/bin/bash
ip=192.168.0.5;   #192.168.43.203;
maxTimeSinceMin=1;
minTimeBetweenScans=1;
outputDirName='./quickScanGenerationFiles';
SYNFile="${outputDirName}/syn_output.txt";
ConnectFile="${outputDirName}/connect_output.txt";
UDPFile="${outputDirName}/udp_output.txt";
FINFile="${outputDirName}/fin_output.txt";
XMASFile="${outputDirName}/xmas_output.txt";
OSFile="${outputDirName}/os_output.txt";
VersionFile="${outputDirName}/version_output.txt";


mkdir -p  "${outputDirName}"
chmod ugo+rwx  "${outputDirName}"

if [ ! -e "$SYNFile" ] ; then
   touch $SYNFile
   chmod ugo+rwx  $SYNFile
fi

if [ ! -e "$ConnectFile" ] ; then
   touch $ConnectFile
   chmod ugo+rwx  $ConnectFile
fi

if [ ! -e "$UDPFile" ] ; then
   touch $UDPFile
   chmod ugo+rwx  $UDPFile
fi

if [ ! -e "$FINFile" ] ; then
   touch $FINFile
   chmod ugo+rwx  $FINFile
fi

if [ ! -e "$XMASFile" ] ; then
   touch $XMASFile
   chmod ugo+rwx  $XMASFile
fi

if [ ! -e "$OSFile" ] ; then
   touch $OSFile
   chmod ugo+rwx  $OSFile
fi

if [ ! -e "$VersionFile" ] ; then
   touch $VersionFile
   chmod ugo+rwx  $VersionFile
fi
```

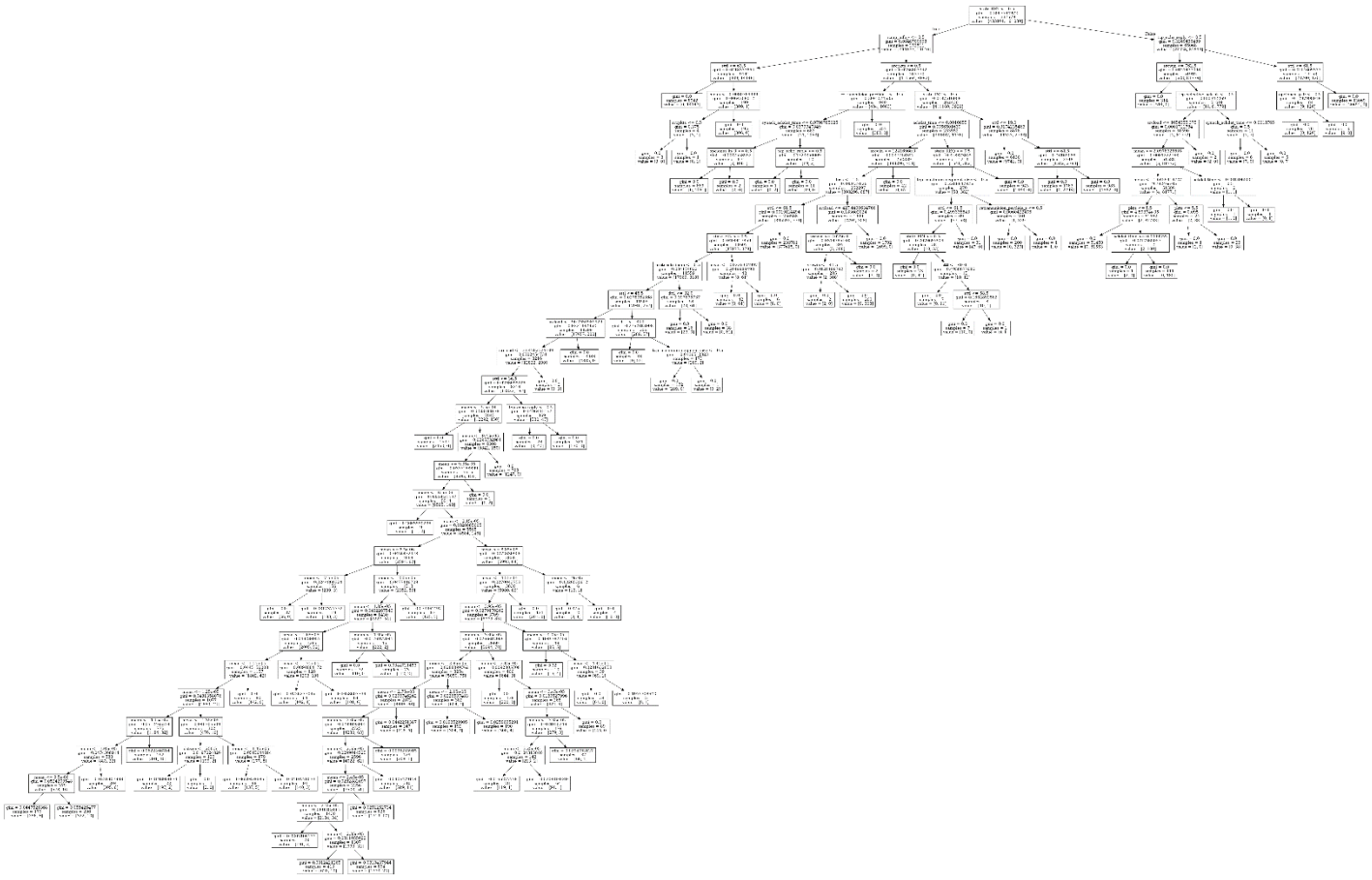
```
today=`date '+%Y-%m-%d %H:%M:%S'`;
getDate() {
    today=`date '+%Y-%m-%d %H:%M:%S'`;
}

fast='';
addFastScan() {
    fast='';
    addParameter=$(($RANDOM%2))
    if [ "$addParameter" -eq 1 ] ; then
      fast=' -F';
    fi
}

delay='';
addScanDelay() {
    d=$(($RANDOM%500))
    delay=" --scan-delay ${d}ms";
}

payload=''
addRandomPayload() {
    bytes=$(($RANDOM%256))
    payload=" --data-length ${bytes}";
}


doSYNScan () {
  echo -e
'##################################################'$"\n${today}\n" | tee -
a  $SYNFile
  echo -e  "nmap -sS -Pn ${fast} ${delay} ${payload} ${ip}"$"\n"
  nmap -sS -Pn $fast $delay $payload $ip| tee -a  $SYNFile
}

doCONNECTscan () {
  echo -e
'##################################################'$"\n${today}\n" | tee -
a  $ConnectFile
  echo -e "nmap -sT -Pn ${fast} ${delay} ${payload} ${ip}"$"\n"
  nmap -sT -Pn $fast $delay $payload $ip | tee -a  $ConnectFile
}

doUDPScan() {
  echo -e
'##################################################'$"\n${today}\n" | tee -
a  $UDPFile
  echo -e "nmap -sU -Pn ${fast} ${delay} ${payload} ${ip}"$"\n"
  nmap -sU -Pn $fast $delay $payload $ip  | tee -a  $UDPFile
}

doFINScan() {
  echo -e
'##################################################'$"\n${today}\n" | tee -
a  $FINFile
  echo -e "nmap -sF -Pn ${fast} ${delay} ${payload} ${ip} "$"\n"
```

```
  nmap -sF -Pn $fast $delay $payload $ip  | tee -a  $FINFile
}

doXMASScan() {
  echo -e
'#####################################################'$"\n${today}\n" | tee -
a  $XMASFile
  echo -e "nmap -sX -Pn ${fast} ${delay} ${payload} ${ip}"$"\n"
  nmap -sX -Pn $fast $delay $payload $ip  | tee -a  $XMASFile
}

doOSScan() {
  echo -e
'#####################################################'$"\n${today}\n" | tee -
a  $OSFile
  echo -e "nmap -O  -Pn ${fast} ${delay} ${payload} ${ip}"$"\n"
  nmap -O  -Pn $fast $delay $payload $ip  | tee -a  $OSFile
}

doVersionScan() {
  echo -e
'#####################################################'$"\n${today}\n" | tee -
a  $VersionFile
  echo -e "nmap -sV -Pn ${fast} ${delay} ${payload} ${ip}"$"\n"
  nmap -sV -Pn $fast $delay $payload $ip  | tee -a  $VersionFile
}

randomScanGenerator () {
  echo  "Scanning host ${ip}"
  while :
  do
    scanToDo=$(($RANDOM%7))
    getDate
    addFastScan
    addScanDelay
    addRandomPayload
    case $scanToDo in
    0)
      doSYNScan
      ;;
    1)
      doCONNECTscan
      ;;
    2)
      doUDPScan
      ;;
    3)
      doFINScan
      ;;
    4)
      doXMASScan
      ;;
    5)
      doOSScan
      ;;
    6)
```

```
      doVersionScan
      ;;
    esac
    sleepTime=$(((($RANDOM%$2)+$1))
    echo $'\n\n';
    echo   "next scan will start in ${sleepTime} minutes :)"
    echo $'\n\n';
    sleep  $(($sleepTime*60))
  done
}
```

randomScanGenerator minTimeBetweenScans maxTimeSinceMin

ARBOL DE DECISION PARA UN NODO



DOCUMENTACION