Universidad Nacional de La Plata

Facultad de Informática

Detección de ataques de seguridad en redes usando técnicas de ensembling

Tesis de Maestría en Redes de Datos
presentada por Paula Venosa

Dirigida por Dr. Sebastián García y Lic. Javier Díaz

Octubre de 2020

Resumen

En la actualidad el malware continúa representando una de las principales amenazas de seguridad informática. Aún resulta difícil contar con sistemas de detección eficientes para separar con precisión el comportamiento normal del malicioso, a partir del análisis del tráfico de red. Ello se debe a las características del tráfico malicioso y el normal ya que el tráfico normal es muy complejo, diverso y cambiante; y el malware también es cambiante, migra y se oculta simulando ser tráfico normal. Además hay gran cantidad de datos a analizar y se requiere que la detección sea en tiempo real para ser útil. Es necesario entonces contar con un mecanismo efectivo para detectar malware y ataques en la red.

A fin de beneficiarse de múltiples clasificadores diferentes, y explotar sus fortalezas, surge el uso de los algoritmos de ensembling, los cuales combinan los resultados de los clasificadores individuales en un resultado final para lograr una mayor precisión y así un mejor resultado. Ello también puede aplicarse a problemas de ciberseguridad, en particular a la detección de malware y ataques mediante el análisis de tráfico de red, desafío que hemos planteado en esta tesis.

Los trabajos de investigación realizados, en relación a ensemble learning de detección de ataques, apuntan principalmente a incrementar el rendimiento de los algoritmos de aprendizaje automático combinando sus resultados. La mayoría de los trabajos proponen el uso de alguna técnica, de ensemble learning existente o creada por los autores, para detectar algún tipo de ataque en particular y no ataques en general. Hasta el momento ninguno aborda el uso de datos de TI (Threat Intelligence por su sigla en inglés) en algoritmos de Ensemble Learning para mejorar el proceso de detección, como así tampoco se trabaja en función del tiempo, es decir teniendo en cuenta lo que ocurre en la red en un intervalo de tiempo acotado. El objetivo de

esta tesis es proponer una metodología para aplicar ensembling en la detección de hosts infectados considerando estos dos aspectos.

En función del objetivo planteado se han investigado y evaluado algoritmos de ensembling aplicables a seguridad en redes y se ha desarrollado una metodología de detección de hosts infectados aplicando ensembling, basado en experimentos diseñados y probados con datasets reales. Dicha metodología plantea realizar el proceso de detección de hosts infectados en tres fases. Dichas fases se llevan a cabo cada una determinada cantidad de tiempo (conocida como ventana de tiempo o TimeWindows). Cada una de ellas aplica ensembling con distintos objetivos. La primera fase lo hace para clasificar cada flujo de red perteneciente a la ventana de tiempo, como malware o normal. La segunda fase lo aplica para clasificar el tráfico entre un origen y un destino, como malicioso o normal, indicando si el mismo forma parte de una infección. Y por último, la tercer fase, con el objetivo de clasificar cada host como infectado o no infectado, considerando los hosts que originan las comunicaciones.

La implementación en fases permite resolver, en cada una de ellas, un aspecto del problema, y a su vez tomar las predicciones de la fase anterior, que se combinan con el análisis propio de la fase para lograr mejores resultados. Además, implica llevar a cabo el proceso de entrenamiento y testeo en cada fase. Dado que el mejor modelo se obtiene a partir del entrenamiento, cada vez que se realiza el mismo para una fase determinada, el modelo se ajusta para detectar nuevos ataques. Esto representa una ventaja frente a las herramientas basadas en firmas o reglas estáticas, donde hay que conocer el comportamiento para agregar nuevas reglas.

Las ventajas del uso de ensembling puede observarse en cada fase en particular. En la Fase 1, aplicando ensembling no hay falsos positivos al clasificar cada flujo de red, como malicioso o normal. Mientras que en dicha fase, sin aplicar ensembling y uusando un único algoritmo para la clasificación se tienen: 10366 falsos positivos en caso de usar Logistic Regression, 266 falsos positivos usando Naive Bayes, y 4 falsos positivos para el caso de Random Forest.

En la Fase 2, el aplicar ensembling para combinar criterios en relación a los distintos tipos de conexiones que se dan entre una IP origen y una IP destino, permite clasificar los flujos de red que van de un origen a un destino, y tener una única decisión para todo ese conjunto de flujos de red. En dicha fase se reducen los

posibles falsos positivos y falsos negativos de la Fase 1, lo cual se demuestra en los experimentos insertando errores aleatorios en el dataset resultante de la Fase 1.

En la Fase 3, el incluir la información de threat intelligence provista por el módulo VirusTotal de Slips (por su sigla en inglés Stratosphere Linux IPS) en el proceso de ensembling de esta fase, permite reducir los falsos negativos provenientes de la fase anterior. Ello también refuerza la decisión para el caso de las direcciones IPs destinos clasificadas como maliciosas. Sin embargo, el peso que se asigna a la información de TI debe ser poco significativo, para evitar falsos positivos en la clasificación de esta fase, donde se clasifica cada dirección IP origen como maliciosa o normal, indicando si está infectada o no.

A partir de los resultados obtenidos se propoone diseñar e implementar un nuevo módulo en Slips para detectar hosts infectados a través del ensembling, que incluye los datos de Threat Intelligence y trabaja en función del tiempo. Tanto la metodología desarrollada como la propuesta de diseño e implementación del módulo implementado constituyen los principales aportes de esta tesis de maestría.

Agradecimientos

Son muchos los agradecimientos que deseo hacer al final de este camino.

A mi familia y a mis amigos, por compartir y disfrutar mis logros, por su ayuda y por su aliento.

A mis Directores, Sebastián García, por dedicar su valioso tiempo y enseñarme tanto, por transmitirme pasión y lograr que dé lo mejor de mí; y Javier Díaz por darme la oportunidad de formar parte del LINTI y de trabajar en lo que me motiva creciendo día a día.

A mis amigas y amigos del LINTI. A Claudia Queiruga y Claudia Banchoff que me escucharon y me hicieron sentir confianza en mí misma. A Joaquín y Laura por su apoyo en los primeros meses de estudio, como un empujón inicial, y por seguir acompañandome y apoyandome hasta el final. En particular a Claudia Banchoff y a Laura por haber revisado mi tesis. A mis amigos Nicolás, Einar y Alejandro, con quienes formamos un grupo de trabajo del cual me siento orgullosa, en el cual priman el compañerismo y las ganas de crecer.

A todo el grupo del Laboratorio Stratosphere de la Universidad Técnica Checa de Praga, dirigido por Sebastián, porque fueron mi sostén, sobre todo en la etapa final de la tesis. Por hacerme sentir parte de ellos no sólo con palabras sino con hechos como compartir los martes de tesis, como sumarme a algunas charlas, como contarles día a día mis planes y avances sobre la tesis. A Verónica Valeros por el seguimiento, los consejos y su revisión. A María José Erquiaga por sus respuestas y su detallada revisión. A Kamila Babayeva por su soporte durante el desarrollo del módulo. A Dita Hollmannová porque además de ser mi compañera de martes de tesis me brindó su ayuda en relación al módulo de VirusTotal.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

La detección de malware y ataques mediante el análisis de tráfico de red sigue siendo un desafío para los responsables de monitorear la seguridad de la red y de gestionar los incidentes de seguridad [12]. Aunque existen varios mecanismos de detección bien conocidos para separar con precisión los comportamientos maliciosos de los normales, todavía es extremadamente difícil contar con sistemas de detección eficientes.

Los principales obstáculos para una buena detección de malware y ataques mediante el análisis de tráfico de red son cuatro. El primero, que el tráfico normal es extremadamente complejo, diverso y cambiante. El segundo, las acciones maliciosas cambian continuamente, adaptándose, migrando y ocultándose como tráfico normal. El tercero, la cantidad de datos para analizar es enorme, lo que obliga a los analistas a perder datos en favor de velocidad. Y el cuarto, la detección debe ocurrir casi en tiempo real para ser de alguna utilidad.

Desde hace algunos años, los sistemas de detección de intrusiones han incorporado paradigmas inteligentes como las técnicas de aprendizaje automático, para resolver estas dificultades. Hoy en día existen también algunas propuestas para implementar algoritmos de Ensemble Learning o *Ensembling*, a fin de combinar múltiples clasificadores para lograr una mejor precisión en la detección. Los algoritmos de Ensemble Learning implementan técnicas para usar, agregar y resumir información provista por varios detectores diferentes en una sola decisión final [25]. Estos permiten a los analistas de seguridad, usar detectores débiles en serie, votar para determinar si un

dominio es malicioso y decidir mejor la acción de bloqueo en base a datos contradictorios, entre otras funcionalidades.

Si bien ha habido algunas buenas propuestas de técnicas de ensembling aplicadas a la seguridad de la red [52], es un tema que se encuentra actualmente en desarrollo. En particular hay dos aspectos de los algoritmos de *Ensembling* que no se estudiaron completamente. Uno de ellos es el uso en algoritmos de Ensemble Learning, de datos de inteligencia de amenazas (más conocido como Threat Intelligence (TI)), por ejemplo, VirusTotal [54]. Y el segundo aspecto es que no hay algoritmos de Ensemble Learning que trabajen en función del tiempo en la detección, es decir que tengan en cuenta lo que ocurre en la red en un intervalo de tiempo dado.

El objetivo de esta tesis es proponer una metodología para aplicar ensembling en sistemas de detección de red usando TI y ventanas de tiempo. Para ello se trabajará en función de los siguientes objetivos:

- Investigar y evaluar algoritmos de ensembling aplicables a seguridad de redes.
- Desarrollar e implementar un método de detección de hosts infectados, basado en ensembling, que tenga en cuenta los resultados de detección de distintos clasificadores, que usen técnicas de aprendizaje automático y datos de Threat Intelligence; y pueda funcionar con ventanas de tiempo y detección a lo largo del tiempo.
- Proponer una integración del mecanismo implementado, al servicio de monitoreo proactivo de seguridad de CERTUNLP [9], CSIRT Académico de la Universidad Nacional de La Plata.

Los contribuciones de esta tesis son:

- El diseño de una metodología para detectar hosts infectados en la red aplicando Ensemble Learning.
- El establecimiento de un procedimiento para testear la metodología a través de experimentos usando datasets reales y sus resultados.
- El diseño y propuesta de implementación del módulo de Ensembling integrado a Slips [32], en el marco de un trabajo de colaboración con el Laboratorio Stratosphere de la Universidad Técnica de República Checa de Praga.

Los objetivos de esta tesis y sus resultados intermedios fueron publicados y presentados en el Congreso Argentino de Ciencias de la Computación (CACIC 2019) [20], y la ampliación de dicho trabajo, seleccionado entre los mejores, fue aceptada para su publicación en el libro Computer Science – CACIC 2019 [21].

Esta tesis está estructurada de la siguiente forma:

En el Capítulo 2 se describen los conceptos a utilizar en el desarrollo de este trabajo. Incluye la definición de flujo de red como unidad de información, la presentación de Slips, una introducción a Threat Intelligence y los conceptos principales de Ensemble Learning. En el Capítulo 3 se describe el estado del arte en relación a la aplicación de Ensemble Learning a problemas de ciberseguridad. El Capítulo 4 trata sobre la metodología que se propone para detectar hosts infectados en la red aplicando Ensemble Learning. En el Capítulo 5 se describen los datasets usados y en el Capítulo 6 los experimentos realizados y sus resultados. El Capítulo 7 trata sobre el diseño y la propuesta de implementación del módulo de ensembling. Finalmente, en el Capítulo 8, se presentan las conclusiones de la tesis.

Capítulo 2

Antecedentes

En este capítulo se presentan los conceptos usados en la tesis y se describe Slips, la herramienta utilizada y extendida en el presente trabajo.

2.1. Flujos de red

Con el fin de facilitar el análisis del tráfico de red, los paquetes son agrupados en flujos de tráfico de red, o conexiones. Un flujo se define como un conjunto de paquetes IP que pasan un punto de observación en la red durante un cierto intervalo de tiempo. Todos los paquetes que pertenecen a un flujo determinado tienen un conjunto de propiedades (atributos de los paquetes) comunes, conocidas como "flow keys" que son: dirección IP origen, dirección IP destino, puerto origen, puerto destino y protocolo, como puede verse en la Figura 2.1. Existen varios protocolos de flujo de red [35], entre ellos NetFlow, sFlow e IPFIX [16].

Los flujos son generados por distintas herramientas entre las que se encuentran Argus [55], Zeek [45] y NFDUMP [48]. La diferencia entre ellos es que Zeek genera flujos cuando la captura termina, mientras que Argus (como otros sensores de flujos de red) permite reportar el flujo cada una determinada cantidad de tiempo, la cual es configurable. Esto se llama flow-report-time y es importante porque "corta" el flujo cada vez que termina un intervalo de tiempo. Trabajar con flujos permite tener menos datos y escalar mejor. Si bien se pierden datos, brinda la posibilidad de resumir las características de la conexión.

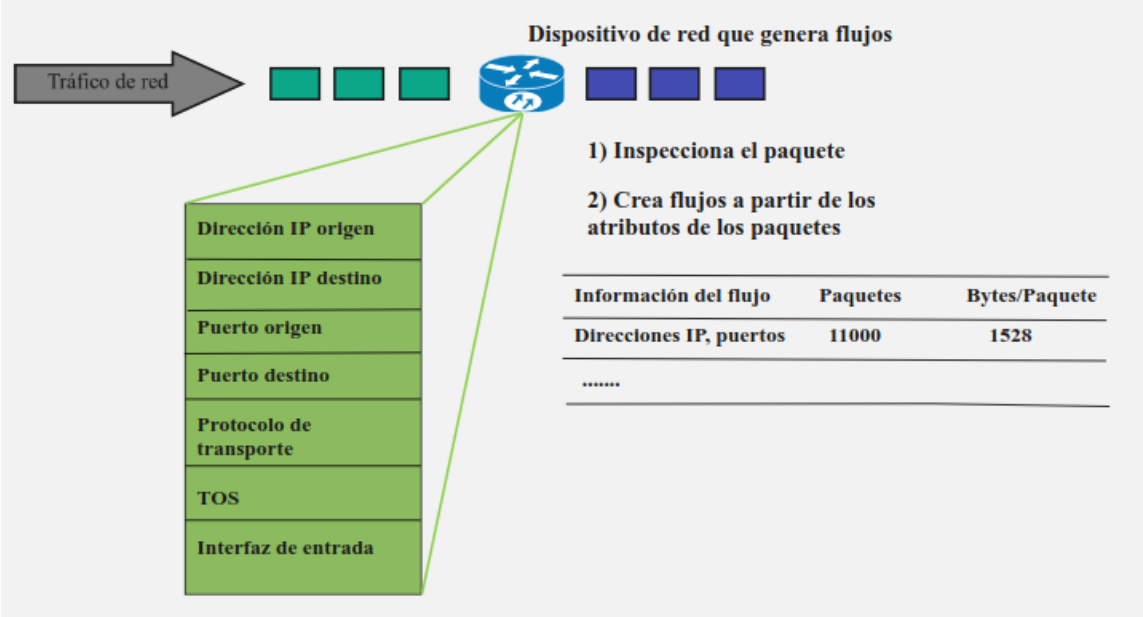

Figura 2.1: Proceso de creación de flujos de red, a partir de los atributos de los paquetes.

2.2. Threat Intelligence

Se llama Threat Intelligence (TI) o inteligencia de amenazas al conocimiento basado en evidencia, que incluye contexto, mecanismos, indicadores, implicancias y recomendaciones orientados a la acción sobre una amenaza o peligro existente o emergente para los activos. Este conocimiento se puede utilizar para tomar decisiones relacionadas a la respuesta que da cada organización frente a determinada amenaza [43].

Hoy en día, las amenazas evolucionan y se complejizan a un ritmo acelerado y muchas organizaciones las enfrentan continuamente. Por ello, el Threat Intelligence se ha convertido en un tema candente. A fin de contrarrestar el aumento de los ataques, muchas organizaciones intentan incorporar fuentes de datos de amenazas en su red (más conocidos como *feeds de información*).

Más allá de la relevancia de la información de Threat Intelligence, es importante una real comprensión de la misma para que su uso sea adecuado y provechoso. Sin embargo, las organizaciones carecen de esa comprensión y no saben qué hacer con

todos esos datos adicionales. Esto aumenta la carga de los analistas que pueden no tener las herramientas para decidir qué priorizar y qué ignorar [22].

Además se requiere que la inteligencia sobre amenazas sea accionable. Es decir debe ser oportuna y llegar en un formato que pueda ser entendido por quien la esté consumiendo. Una forma de lograrlo es cuando la información de Threat Intelligence se integra fácilmente con todas las soluciones de seguridad ya presentes en su entorno.

Hay tres categorías de fuentes de TI: internas, externas y comunitarias [5].

- Las fuentes internas recopilan la información de amenazas dentro de la organización, específicamente de los registros provistos por algunos servicios o aplicaciones de la red interna (registro de correo electrónico, alertas, informe de respuesta a incidentes, registros de eventos, registros DNS, registro de firewall, etc.) y de los sistemas SIEM (por sus siglas en inglés Security Information and Event Management) que se implementan en las mismas.

- Las fuentes externas son organizaciones que proveen información de amenazas y tienen una amplia cobertura de datos. Esta información requiere ser analizada en cada organización para para determinar su relevancia (en función del conocimiento de los servicios de la organización y el impacto de las amenazas sobre los mismos). Hay fuentes externas que proporcionan los datos sin costo y otras que son pagas.

- Las fuentes comunitarias son aquellas que comparten información de amenazas a través de un canal confiable entre miembros con el mismo interés.

VirusTotal es un feed de información que proporciona los resultados del análisis de archivos sospechosos y URLs. VirusTotal inspecciona esos elementos con una gran cantidad de antivirus y servicios de listas de bloqueos de URLs y dominios. Además utiliza un conjunto de herramientas para extraer señales del contenido analizado [54].

El análisis de archivos sospechosos y URLs puede realizarse a través de la página WEB, las extensiones del navegador o la API que VirusTotal provee. Al enviar un archivo o URL, los resultados se comparten con el solicitante, y también entre los socios examinadores, es decir otros usuarios de VirusTotal. Estos últimos utilizan los resultados para mejorar sus propios sistemas. En consecuencia, al enviar archivos,

URL, dominios, etc. a VirusTotal, se contribuye con la Comunidad VirusTotal y la seguridad global de Internet.

VirusTotal puede ser útil para detectar contenido malicioso y también para identificar falsos positivos: elementos normales e inofensivos detectados como maliciosos por uno o más motores de antivirus.

La API (Application Programming Interface) permite acceder a la información generada por VirusTotal sin la necesidad de utilizar la interfaz del sitio web HTML. La misma cuenta con dos versiones: una API pública y una API privada. Estas se diferencian en la cantidad máxima de solicitudes que pueden hacerse por minuto y la prioridad que dichas consultas tienen para el motor de VirusTotal. Actualmente la API pública permite hasta 4 consultas por minuto y provee un acceso de mínima prioridad. Mientras que en la API privada, la tasa de solicitudes y el número total de consultas permitidas solo están limitadas por los términos de servicio del usuario.

En particular, VirusTotal provee la siguiente información para una dirección IP o para un dominio [53]:

- Sistema autónomo (AS, por sus siglas del ingles Autonomous System) y país de localización para direcciones IP. La información del país está dada en código de país (por ejemplo ES para españa, CA para Canadá, etc)

- Información de replicación pasiva de DNS: todas las asignaciones de nombres de dominio IP que VirusTotal ha visto para el elemento por el cual se ha consultado.

- Búsquedas Whois: información de registro para el recurso por el cual se consulta, como ser el nombre de dominio, el bloque IP o el sistema autónomo.

- Subdominios observados: dominios vistos jerárquicamente bajo otro dominio almacenado en VirusTotal.

- Dominios hermanos: dominios en el mismo nivel jerárquico que el dominio que se está analizando.

- URLs: las últimas URLs vistas bajo el dominio o la dirección IP que se está analizando.

- Archivos descargados: últimos archivos que se han descargado de las URL ubicadas en el dominio o la dirección IP en análisis.
- Archivos de comunicación: archivos más recientes que, a través de su ejecución en un entorno virtual aislado, realizan algún tipo de comunicación con la dirección IP o el dominio en consideración.
- Referencias de archivos: VirusTotal inspecciona las cadenas contenidas en los archivos enviados al servicio y les aplica ciertas expresiones regulares para identificar dominios y direcciones IP. Registra entonces los archivos que han hecho referencia al dominio o la dirección IP en consideración.

2.3. Técnicas de Ensembling

Ensemble Learning es un paradigma de Machine Learning donde múltiples agentes (llamados *base learners* o algoritmos base) se combinan y se entrenan para resolver el mismo problema. A diferencia de las técnicas de Machine Learning clásicas las cuales tratan de aprender un modelo (hipótesis) a partir de los datos de entrenamiento, las técnicas de Ensemble Learning tratan de construir un conjunto de modelos y combinarlos para usarlos en la predicción [57].

El Ensemble Learning puede ser:

- Homogéneo: cuando utiliza un algoritmo de aprendizaje único, es decir combina algoritmos base del mismo tipo (homogéneos).
- Heterogéneo: cuando utiliza múltiples algoritmos de aprendizaje, es decir combina algoritmos base de distinto tipo (heterogéneos).

Los métodos de Ensemble Learning implican una mejora en lo que refiere a la habilidad de generalización y al poder predictivo del aprendizaje. A menudo el nivel de precisión de los algoritmos base es apenas superior al azar debido a que tienen alto sesgo o demasiada variación. El Ensemble Learning intenta reducir ambas variables combinando los *base learners* como se muestra en la Figura 2.2.

Los tres métodos más populares para combinar las predicciones de diferentes modelos son [57]:

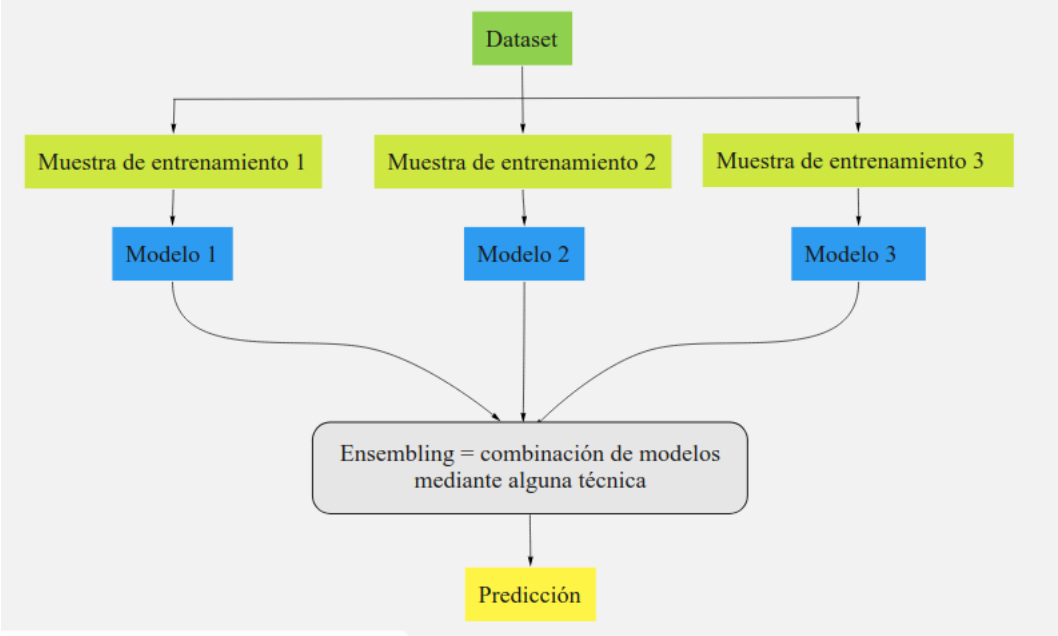

Figura 2.2: Arquitectura general del Ensemble Learning. Las predicciones son el resultado de aplicar alguna técnica de ensembling para combinar diferentes modelos.

- Bagging: construye múltiples modelos (típicamente del mismo tipo) a partir de diferentes submuestras del conjunto de datos de entrenamiento.

- Boosting: construye múltiples modelos (típicamente del mismo tipo) cada uno de los cuales aprende a corregir los errores de predicción del modelo anterior en la secuencia de modelos.

- Voting: construye múltiples modelos (típicamente de diferentes tipos) y usa estadísticas simples (como calcular la media) para combinar predicciones.

Otra técnica de Ensemble Learning, que algunos autores también consideran popular, es el Stacking o Stacked Generalization. Ésta utiliza un algoritmo de meta aprendizaje para aprender cómo combinar mejor las predicciones de dos o más algoritmos de Machine Learning.

2.3.1. Bagging

El *Bagging* o *Boostrap Agregation* toma múltiples muestras del conjunto de datos de entrenamiento (con reemplazo) y entrena un modelo para cada muestra. La Figura 2.3 muestra la arquitectura de esta técnica.

La misma utiliza lo que se conoce como muestra boostrap. Una muestra de bootstrap (no paramétrica) es una selección aleatoria de varios elementos del conjunto de datos con reemplazo. Es decir, una muestra de bootstrap puede contener múltiples copias de una fila de los datos originales [4].

La predicción final se obtiene promediando las predicciones de todos los submodelos, en los problemas de regresión. Mientras que en los problemas de clasificación, la predicción final se puede obtener tanto promediando las predicciones como usando probabilidades basadas en los porcentajes de las diferentes clases.

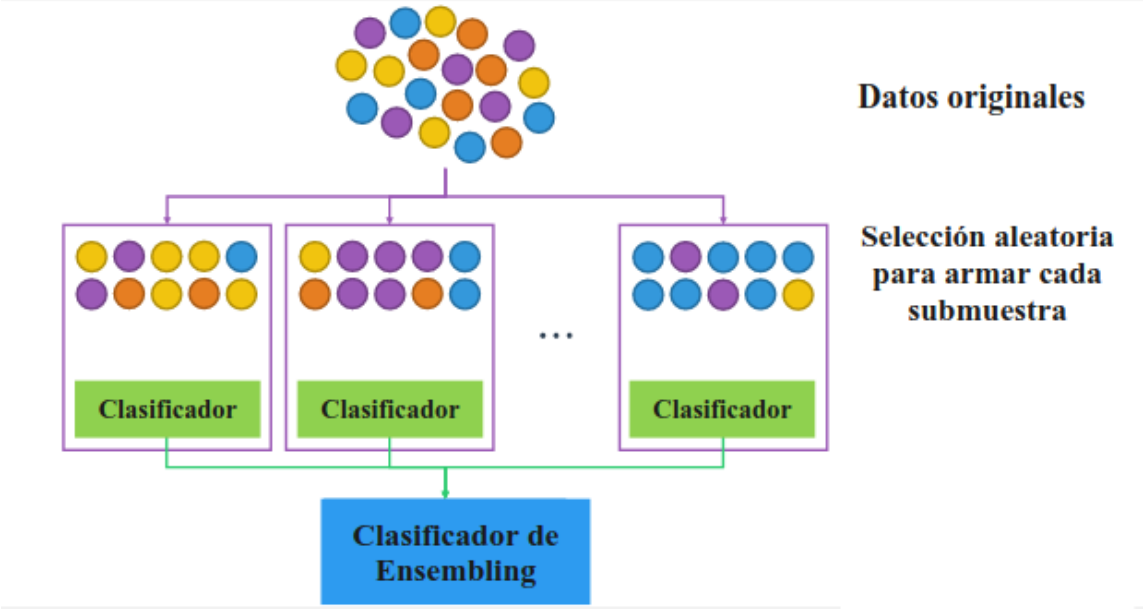

Figura 2.3: Arquitectura de la técnica de Ensemble Learning **Bagging** [47].

Entre los ejemplos de modelos que implementan Bagging se encuentran *Bagged Decision Trees*, *Random Forest* y *Extra Trees* [6].

2.3.2. Boosting

El *boosting* crea una secuencia de modelos que intentan corregir los errores de los modelos anteriores a ellos en la secuencia previa, como se refleja en la Figura 2.4. Una vez creados, los modelos hacen predicciones que pueden ser ponderadas por su precisión demostrada y los resultados se combinan para crear una predicción final [4]. En un primer paso, el algoritmo se entrena en todo el conjunto de datos. Los modelos posteriores se construyen ajustando los residuos del algoritmo anterior. Esto se hace dando mayor peso a las observaciones que el modelo anterior predijo incorrectamente.

Se basa en la creación de una serie de algoritmos débiles, cada uno de los cuales puede no ser apropiado para todo el conjunto de datos, pero sí lo es para una parte del mismo. Por lo tanto, cada modelo aumenta el rendimiento del conjunto.

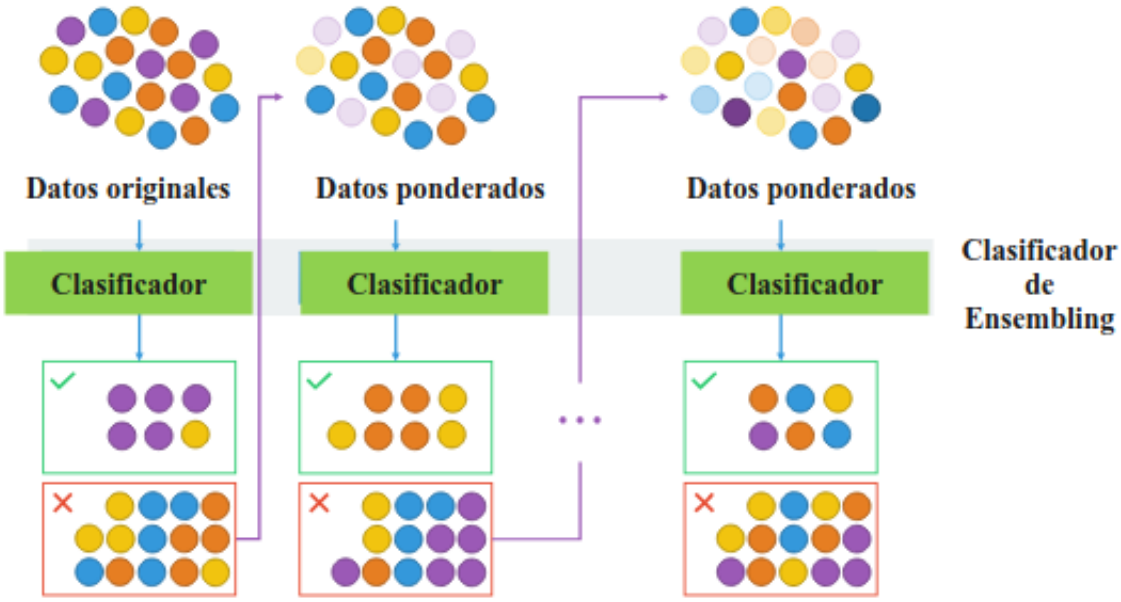

Figura 2.4: Arquitectura de la técnica de Ensemble Learning **Boosting** [46].

Los dos algoritmos más comunes que implementan boosting son Adaboost y Stochastic Gradient Boosting [6]:

2.3.3. Voting

Voting es una de las formas más simples de combinar las predicciones de múltiples algoritmos de aprendizaje automático, su funcionamiento se puede observar en la Figura 2.5. Funciona creando dos o más modelos independientes (heterogéneos) a partir de su conjunto de datos de entrenamiento. Luego se usa un método de votación para realizar predicciones [57].

Dentro de esta técnica existen variantes, como el *Majority Voting* o *Hard Voting*, el *Soft Voting* y el *Weighted Voting*.

En el *Majority Voting* se entrenan varios modelos con los mismos datos. Al momento de predecir, se obtiene una predicción de cada modelo. Cada modelo tendrá asociado un voto. Luego la predicción final estará determinada por lo que que voten la mayoría de los modelos [27].

En el *Soft Voting* se usa "voto suave". En esta variante, se le da más importancia a los resultados en los que algún modelo esté muy seguro. Es decir, cuando la probabilidad de la predicción está muy cercana a 1, se le da más peso a la predicción de ese modelo[27].

El *Weighted Voting* se usa cuando los clasificadores individuales tienen un rendimiento desigual. Dando más poder a los clasificadores más fuertes en la votación, se realiza una votación ponderada.

Las predicciones de los submodelos se pueden ponderar, pero es difícil especificar los pesos de los clasificadores de forma manual o incluso heurística. Los métodos más avanzados aprenden cómo ponderar mejor las predicciones de los submodelos.

2.3.4. Stacking

El *Stacking* o *Apilamiento* consiste en entrenar un modelo para realizar la agregación o combinación de las predicciones de todos los submodelos, en lugar hacerlo usando funciones triviales (como *Hard Voting*).

En un primer paso, la muestra se divide en una submuestra para training y una submuestra para testing. Luego se entrena un conjunto de algoritmos base con la muestra de training. Y los modelos resultantes se evalúan usando la muestra de testing.

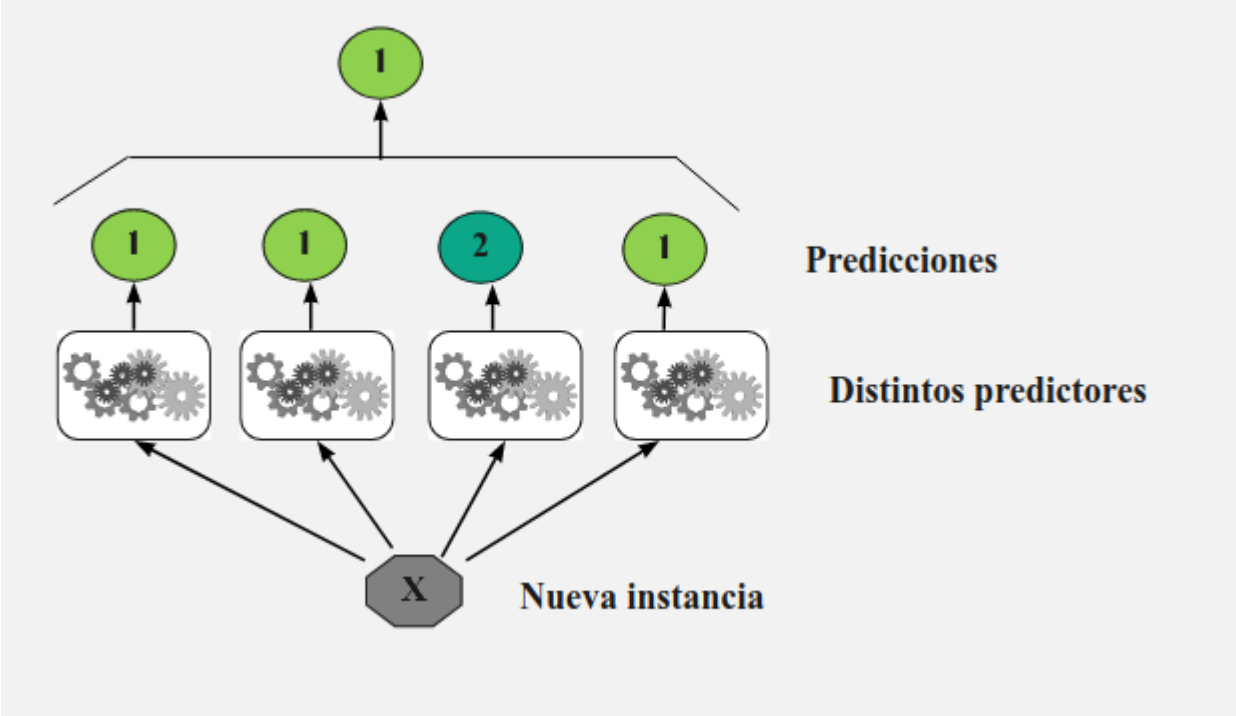

Figura 2.5: Arquitectura de la técnica de Ensemble Learning **Voting**.

Cada uno de los submodelos predice un valor diferente y finalmente el *ensemble learner* (también llamado *meta-learner*) toma estas predicciones como entradas y hace la predicción final. Su arquitectura puede observarse en la Figura 2.6.

Cuando se implementa *Stacking* el ensembling puede ser homogéneo u heterogéneo. Como se describió anteriormente en esta sección el ensembling homogéneo combina algoritmos base del mismo tipo mientras que el ensembling heterogéneo combina algoritmos base de distinto tipo.

2.4. Slips

Slips (Stratosphere Linux IPS) es un sistema de prevención de intrusiones modular desarrollado en Python. El mismo se basa en técnicas de aprendizaje automático para detectar comportamientos maliciosos en el tráfico de la red [32]. Slips fue diseñado para enfocarse en ataques dirigidos, como detección de canales de *Comand and Control*, para proporcionar una buena visualización para el analista de seguridad.

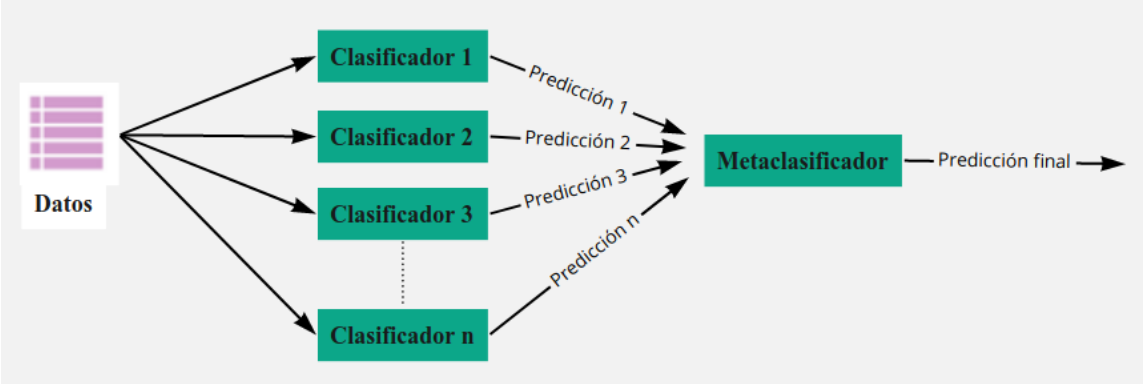

Figura 2.6: Arquitectura de la técnica de Ensemble Learning **Stacking** [50].

A partir del tráfico de red, crea perfiles para cada dirección IP, y luego divide el tráfico en ventanas de tiempo. Para cada ventana de tiempo, Slips extrae características del tráfico y luego las analiza de diferentes maneras a fin de detectar comportamiento malicioso [34].

Al momento de escribir esta tesis, la última versión publicada de Slips es la 0.6.8. La misma detecta escaneos de puerto horizontales y verticales, así como la existencia de varias conexiones de comando y control.

Slips es software libre y se encuentra disponible en la página de Stratosphere [33]. Slips puerde ser ejecutado sobre sistemas Linux basados en Debian y MacOS (10.9.5, 10.10.x, 10.12.x). En su versión actual Slips provee la facilidad de ejecutarse dentro de un contenedor Docker [41], para analizar archivos de flujos de red o archivos pcap. Sin embargo, para realizar el análisis de tráfico de red en tiempo real, la única opción es realizar la instalación tradicional de la herramienta.

2.4.1. Arquitectura de Slips

Slips trabaja a nivel de flujo de red, en lugar de inspección de paquete, obteniendo una visión de alto nivel de los comportamientos. Slips crea perfiles de tráfico para cada dirección IP que aparece en el tráfico. Un perfil contiene el comportamiento completo de cada una de ellas. Cada perfil se divide en ventanas de tiempo. Cada ventana de tiempo tiene una duración de 1 hora de manera predeterminada (puede establecerse por configuración) y contiene docenas de funciones calculadas para todas

las conexiones que comienzan en esa ventana de tiempo. Las detecciones se realizan en cada ventana de tiempo, lo que permite que el perfil se marque como no infectado en la ventana de tiempo siguiente.

Slips puede leer flujos de diferentes fuentes como archivos pcap o salidas de Zeek, Bro, Nfdump, entre otros. Una vez leídos, los datos son procesados e insertados dentro del perfil de cada dirección IP origen. Por cada dirección IP analizada, Slips crea esta estructura *Profile* que representa el perfil para dicha dirección y contiene tres tipos de logs: *Time Windows Files*, *Timeline file* y *Profile File*.

Además de la información de los perfiles, Slips crea algunos archivos con información sobre la captura completa, como Blocked.txt. Este archivo tiene información sobre todas las direcciones IP que se detectaron y bloquearon.

Slips usa una base de datos Redis donde guarda los flujos que procesa y la información asociada a los mismos. Además cuenta con una interfaz de consola, llamada Kalipso [2], basada en Node.js.

2.4.2. Módulos principales de Slips

Slips cuenta con módulos, que son archivos escritos en Python. Los mismos permiten a cualquier desarrollador ampliar su funcionalidad [28]. Procesan y analizan datos, realizan detecciones adicionales y almacenan datos en la base de datos Redis para que otros módulos los consuman.

Actualmente, Slips cuenta con los siguientes módulos:

- Módulo *asn*: permite cargar y encontrar el Número de Sistema autónomo (ASN, por sus siglas del ingles Autonomous System Number) de cada IP.
- Módulo *geoip IP*: su función es encontrar el país y la información de geolocalización de cada IP.
- Módulo *https*: permite entrenar o probar un clasificador RandomForest para detectar flujos HTTPS maliciosos.
- Módulo *Port Scan Detector*: su función es detectar escaneos de puertos horizontales y verticales.

- Módulo Threat Intelligence: sirve para verificar si cada dirección IP está en una lista de IPs maliciosas.
- Módulo *Timeline*: permite crear una línea de tiempo de lo que sucedió en la red, en función de todos los flujos y tipos de datos disponibles.
- Módulo *VirusTotal*: su función es buscar la dirección IP en VirusTotal.
- *Kalipso* [2]: es la interfaz gráfica de usuario para mostrar el tráfico analizado por Slips.

El núcleo de Slips no solo lo constituyen los algoritmos de aprendizaje automático, sino también los modelos de comportamiento que se utilizan para describir los flujos en función de la duración, el tamaño y la periodicidad de los mismos. Esto es muy importante porque los modelos están seleccionados para maximizar la detección.

Por último una característica que hace atractivo a Slips es que implementa una API, a partir de la cual se puede incorporar fácilmente un nuevo algoritmo de detección que cualquier desarrollador implemente, dando así la posibilidad de que la herramienta crezca a partir de los aportes de la comunidad.

Capítulo 3

Estado del arte

La detección de intrusiones en la red es un área de investigación importante, ya que los ciberataques aumentan día a día [12]. Existen numerosos estudios para proponer enfoques para detectarlos. Sin embargo, a medida que los ciberataques se vuelven más complejos, los enfoques existentes no logran abordar el problema de manera efectiva.

Es por ello que la detección de intrusiones en la red continúa siendo un desafío en la toma de decisiones, que se puede abordar mediante la aplicación de algoritmos de clasificación [23]. Estos algoritmos usan técnicas de aprendizaje automático para detectar ataques de red y malware. Ello ofrece las siguientes ventajas:

- La capacidad del aprendizaje automático para generalizar y así detectar nuevos tipos de intrusiones.
- Las firmas de ataque se pueden extraer automáticamente de los datos de tráfico etiquetados.
- La capacidad de adaptarse a nuevos ataques.

A fin de beneficiarse de múltiples clasificadores diferentes, y explotar sus fortalezas, surge el uso de los algoritmos de Ensembling [57] [56], los cuales combinan los resultados de los clasificadores individuales en un resultado final para lograr un mejor resultado. A partir de esta combinación de resultados, se puede mejorar el rendimiento de los algoritmos individuales. Algunos estudios han demostrado que

la aplicación del paradigma de Ensemble Learning en los sistemas de detección de intrusos puede resultar versátil y, sin duda, mejorar la precisión de la predicción y la velocidad de detección [13] [14]. En particular la mayor velocidad en la detección usando ensembling se puede lograr a partir de la utilización de arquitecturas paralelas.

Por otro lado, los ciberataques poseen distintos tipos de características: generales, de tráfico o asociadas a la conexión, de contenido o asociadas a los datos. La selección de características es esencial, por lo que es importante continuar investigando al respecto. También es fundamental evaluar qué clasificadores base usar y cómo deben combinarse de manera de diseñar arquitecturas donde los clasificadores múltiples colaboren entre sí en lugar de competir.

En ese sentido existen propuestas sobre la implementación de modelos, con datasets armados a partir de distintas características y diferentes clasificadores, que luego se combinan con técnicas de Ensemble Learning [25] [24]. Este problema se aborda de una manera más específica en [18], donde el Ensemble Learning se aplica para la detección de ataques DoS[1], R2L[2], U2R[3] y Probing[4].

MPLM (Multi-Perspective Machine Learning) es una técnica de Ensembling que se propone para analizar las diferentes facetas o perspectivas [38] y [37]. Las perspectivas están representadas por un conjunto de características de los datos. Los modelos se obtienen desde las diferentes perspectivas y luego se combinan. Un problema a resolver en este marco es el criterio y la implementación de la selección de características para construir las perspectivas. MPLM se puede aplicar tanto a la detección de varios ataques como DoS, R2L, U2R y Probing [38] como a la detección de actividad de botnets [37], donde las perspectivas representan las diferentes etapas de su ciclo de vida. En [38] se presenta un nuevo modelo para aplicar el aprendizaje en problemas multifacéticos, centrándose en la selección de características que se incluirán en cada perspectiva (perspectivas basadas en red, perspectivas basadas en

[1]Denial of Service o Denegación de servicio

[2]Root to local attack. Tipo de ataque que se realiza para acceder a una dirección de red particular de forma remota de forma ilegal.

[3]User to root. Tipo de ataque en el que el intruso intenta acceder a los recursos de la red como un usuario normal, y después escala privilegios convirtiendose en un usuario administrador o root.

[4]Probing o sondeo. Tipo de ataque donde el intruso escanea los dispositivos de red para descubrir puertos abiertos y/o vulnerabilidades, para lograr luego tener acceso ilegal a los servicios y su información (ejemplos de ello son nmap y portsweep.

host, perspectivas basadas en DNS). El criterio de selección de características debe tener en cuenta que la inclusión de características fuertemente correlacionadas puede no contribuir al proceso de clasificación.

También se han desarrollados propuestas basadas en la técnica de Stacking para detectar intrusiones en la red (Probe, DoS, UR2 y R2L) [39]. En este trabajo los modelos se generan utilizando muestras de la selección aleatoria de características del dataset. Se propone seleccionar los mejores modelos de acuerdo con un criterio definido (como precisión, tasa de verdaderos positivos, entre otros) y combinarlos con Stacking como técnica de Ensemble Learning.

Otros trabajos se enfocan en detección de botnets a partir de la clasificación de flujos de tráfico de red [19] [36]. En [19] se proponen realizar la detección en dos etapas. En la primer etapa aplican un algoritmo de clustering para generar clusters que agrupan flujos de red con características simililares. Y en la segunda etapa, los algoritmos de clasificación se aplican a cada cluster para separar los flujos de botnets y los flujos normales. Dada la inestabilidad de los algoritmos de clustering estos métodos presentan falencias, que intentan ser mejoradas con algunas variantes como se propone en [36] a través del uso de algoritmos link-based para agrupar los flujos de red en la etapa 1 en lugar de los algoritmos de clustering.

Más allá de los mecanismos y herramientas, para implementar una defensa eficaz, la organización necesita contar con información sobre los posibles adversarios, así como sus técnicas, tácticas y procedimientos. Esta llamada inteligencia de amenazas ayuda a la organización a comprender mejor su perfil de amenazas [15]. Los feeds de Threat Intelligence (TI) o fuentes de inteligencia de amenazas permiten a las organizaciones obtener indicardores que son usados por sus firewalls y sus sistemas de detección de intrusos para una reacción oportuna a las amenazas emergentes.

Las fuentes de inteligencia suelen estar formadas por indicadores simples. Pueden brindar información sobre dominios sospechosos, listas de hashes de malwares conocidos o direcciones IP asociadas con actividad maliciosa, entre otras cosas. Con la información proporcionada por estos fuentes de inteligencia, las organizaciones suelen optar por incluir en la lista negra, las comunicaciones y las solicitudes de conexión que se originan en fuentes maliciosas, por ejemplo [29])

Si bien las fuentes de inteligencia pueden ser fáciles de entender y utilizar, no son

una solución completa. Las mismas no brindan contexto ni priorizan las amenazas, por lo que es necesario contar con procedimientos y mecanismos para extraer valor de ellas y usarlas adecuadamente. Además, a pesar de que se las utiliza ampliamente en la industria como una herramienta útil para mitigar los ataques, hay estudios que afirman que su calidad no es tan alta como la esperada, así como tampoco su especificidad y su completitud [51] [10] [30].

En consecuencia, se puede pensar que un buen uso de la información provista por las fuentes de inteligencia es integrarla y combinarla con los mecanismos de detección de malware y ataques en la red que la organización posea. Sin embargo ninguna de las propuestas anteriormente descriptas en este capítulo incluye la información provista por las fuentes de TI en el proceso de clasificación.

En este contexto es que se plantea aportar al ámbito de la ciberseguridad, y a la detección de hosts infectados en particular, aprovechando las ventajas de las técnicas de Ensembling [9] [34], incluyendo la información de las fuentes de TI en el proceso de detección.

Capítulo 4

Ensembling para detectar hosts infectados en la red

La metodología propuesta tiene como fin mejorar el proceso de detección de hosts infectados en la red. Los hosts a analizar son aquellos que inician conexiones (flujos) de red. La detección se realiza mediante la implementación de Ensemble Learning. Las decisiones a tomar durante el proceso se basan en diferentes datos proporcionados por Slips [32].

Se trabaja con todos los flujos proporcionados por un dispositivo que genera flujos para la red objetivo. Para determinar si un host está infectado en una ventana de tiempo, se considera la siguiente información:

1. Las diferentes predicciones para cada flujo de red, uno para cada clasificador.

2. El conjunto de alertas de comportamiento malicioso asociadas con la IP dada (que tienen esta IP como origen).

3. Los datos provistos por de diferentes fuentes de Threat Intelligence que informan si los destinos de los flujos analizados son maliciosos o presentan indicios de serlo (con algún porcentaje de confianza).

En base a lo anteriormente detallado, se propone aplicar Ensemble Learning para tomar diferentes decisiones, como se muestra en la Figura 4.1. Primero, para determinar si cada flujo es malicioso o normal. Segundo, para determinar si el conjunto

de flujos que van de un origen a un destino son parte de una infección. Y tercero, para decidir si cada si dirección IP origen está infectada o no.

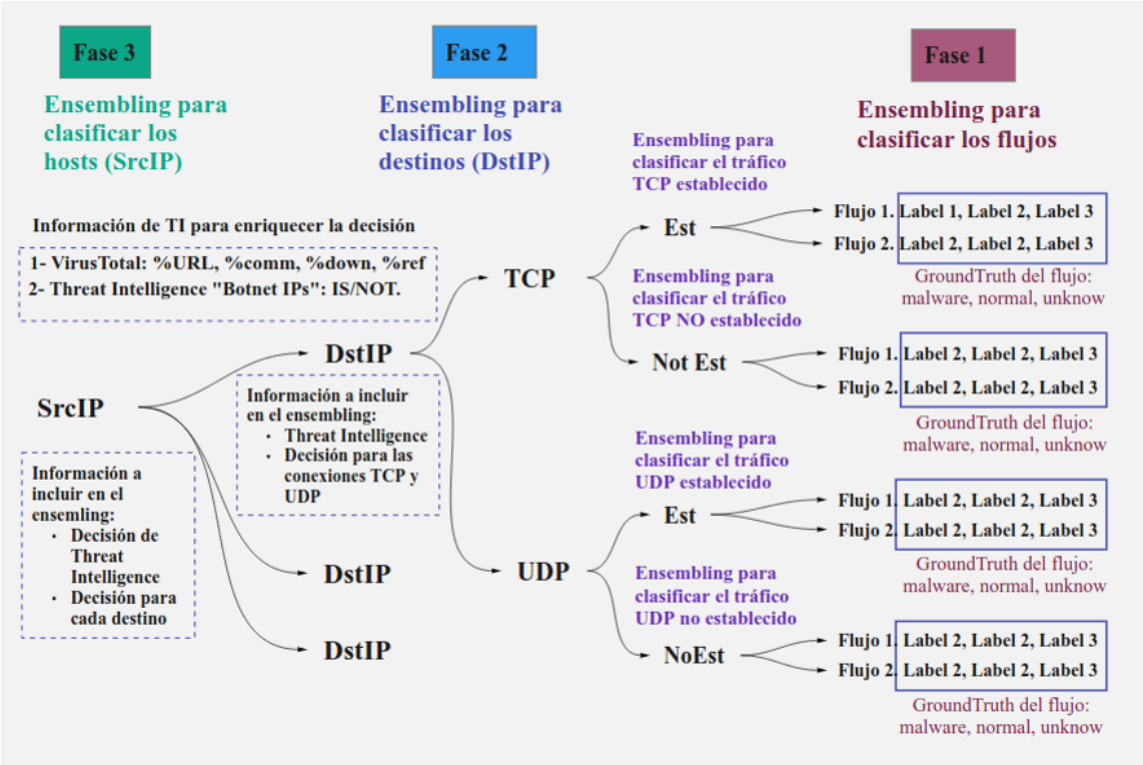

Figura 4.1: Metodología de ensembling propuesta para detectar hosts infectados en la red. En la Fase 1 se aplica ensembling para clasificar cada flujo, etiquetándolo como malware o normal. En la Fase 2 se aplica ensembling para clasificar el conjunto de flujos entre un origen y un destino, indicando si forma parte de una infección o no. En la Fase 3 se clasifican los hosts (identificados por la IP origen) etiquetándolos como infectado o normal.

El trabajo se ha realizado en fases, de manera de modularizar el análisis, realizar pruebas experimentales para cada una, obtener resultados para una parte del problema en particular y poder ajustar la solución en base a ello.

4.1. Fase 1: Ensemble Learning para clasificar flujos de red

El objetivo de esta primera fase es asignar una etiqueta (a la que haremos referencia como "label" por su significado en inglés utilizado en la mayoría de la bibliografía) para cada flujo de red. El valor será *malicious* (malicioso) o *normal*. Dicho valor es el resultado de aplicar Ensemble Learning, sobre las predicciones dadas por los distintos clasificadores de Slips, como se muestra en la Figura 4.2.

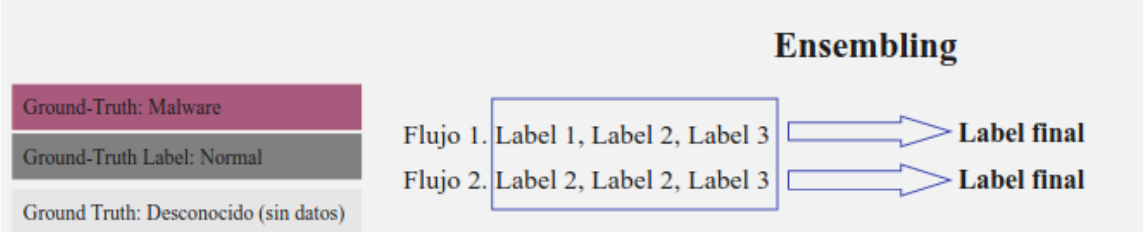

Figura 4.2: Fase 1 del ensembling. Se tiene un conjunto de labels para cada flujo, que se combinan aplicando *Weighted Voting* para obtener un único label por flujo.

Cada flujo de red posee un conjunto de n labels provenientes de las predicciones de los n clasificadores en operación de Slips, como puede verse en la Figura 4.3. Dichas predicciones se combinarán de manera tal de obtener una única decisión para cada flujo.

Los labels asignados por los clasificadores de Slips son resultado de aplicar algoritmos de aprendizaje automático base para clasificar flujos. Para combinar esos labels a fin de tomar una decisión es que se aplica *Ensemble Learning*.

Para determinar la técnica de Ensemble Learning más adecuada a implementar en esta fase se realizaron distintos experimentos y se evaluaron los resultados, descriptos en el próximo capítulo. A partir de ellos se decidió usar *Weighted Voting* en esta instancia.

Como resultado de esta fase se tiene entonces un label por flujo: malicious (por malicioso) o normal. La salida de esta fase es un dataset con los mismos campos que el dataset entrante, reemplazando los n labels por un único label, resultante del ensembling aplicado.

StartTime	Dur	Proto	SrcAddr	Sport	Dir
2011-08-18 11:54:06.982742	15,14016	tcp	147.32.84.170	42557	->

DstAddr	Dport	State	sTos	dTos
77.75.72.29	80	FSPA_FSPA	0.0	0.0

TotPkts	TotBytes	SrcBytes	SrcPkts	Label1
38	21570	4982	0.0	Normal

Label2	Label3	groundTruth
Normal	Normal	Normal

Figura 4.3: Ejemplo que muestra los campos de cada flujo. Los campos Label1, Label2 y Label3 representan los labels resultantes de las predicciones de los clasificadores. La Fase 1 recibe un conjunto de flujos con estos campos.

4.2. Fase 2: Ensemble Learning para clasificar conjuntos de flujos que van de un origen a un destino

El objetivo de esta segunda fase es decidir la asignación del label para cada conjunto de flujos de red que representa todas las conexiones entre un origen y un destino. Dicho label será malicioso si el conjunto de flujos forma parte de una infección, y normal en caso contrario. Como resultado de esta fase se tiene un conjunto de flujos de red etiquetados, como puede observarse en la Figura 4.4.

El ensembling se lleva a cabo de la siguiente forma:

1. Para el conjunto de flujos que van de una IP origen (SrcAddr) a una IP destino (DstAddr), se analizan las conexiones TCP establecidas, las conexiones TCP no establecidas, las conexiones UDP establecidas y las conexiones UDP no establecidas (se considera conexiones UDP establecidas a los flujos UDP que recibieron una respuesta y conexiones UDP no establecidas a aquellos que no la recibieron).

2. Para cada uno de los grupos (conexiones TCP establecidas, conexiones TCP no establecidas, conexiones UDP establecidas y conexiones UDP no establecidas)

se calcula el porcentaje y la cantidad de flujos etiquetados como malware en la Fase 1. Para cada grupo se tendrán entonces 2 valores: $Porcentaje_{Flujos=Malware}$ y $Cantidad_{Flujos=Malware}$.

En esta fase existen 2 niveles de decisión:

- Primer nivel de decisión: tiene como fin etiquetar cada grupo de flujos (Conexiones TCP Establecidas, Conexiones TCP no establecidas, Conexiones UDP establecidas y Conexiones UDP no establecidas) como malicioso o no malicioso.
- Segundo nivel de decisión: tiene como fin etiquetar el conjunto total de flujos de red que representa todas las conexiones entre el origen y el destino, como malicioso o no malicioso. Constituye la decisión final de la Fase 2.

4.2.1. Primer nivel de decisión

Para cada grupo de flujos se definen 2 criterios:

1. Criterio 1: se etiqueta un grupo como *malicious* (malicioso) si cumple la siguiente condición:

$$Porcentaje_{Flujos=Malware} > Threshold_{Porcentaje} \tag{4.1}$$

2. Criterio 2: se etiqueta un grupo como *malicious* (malicioso) si cumple la siguiente condición:

$$Cantidad_{Flujos=Malware} > Threshold_{Cantidad} \tag{4.2}$$

Los valores de $Threshold_{Porcentaje}$ y $Threshold_{Cantidad}$ se determinan en el entrenamiento de esta fase.

Cada grupo de flujos, que representa las conexiones TCP establecidas, conexiones TCP no establecidas, conexiones UDP establecidas y conexiones UDP no establecidas, será etiquetado como *malicious* (malicioso) si ambos criterios (4.1 y 4.2) son verdaderos, y normal en caso contrario.

El ensembling usado es comparable con una técnica de voting. En este caso, para que el label sea *malicious*, el resultado de los votos tiene que ser unánime (ambos

clasificadores tienen que votar malicioso), caso contrario la etiqueta que se asigna es normal.

Como resultado de este proceso de primer nivel de decisión se construye un dataset intermedio, que incluye para cada par de direcciones IP origen-IP destino, la etiqueta para cada grupo. Las etiquetas para cada grupo son:

- **TCPELabel:** etiqueta asignada al grupo de flujos TCP establecidos que van desde la dirección IP origen SrcAddr a la dirección IP destino DstAddr.
- **TCPNELabel:** label asignado al grupo de flujos TCP no establecidos que van desde la dirección IP origen SrcAddr a la dirección IP destino DstAddr.
- **UDPELabel:** etiqueta asignada al grupo de flujos UDP establecidos que van desde la dirección IP origen SrcAddr a la dirección IP destino DstAddr.
- **UDPNELabel:** label asignado al grupo de flujos UDP no establecidos que van desde la dirección IP origen SrcAddr a la dirección IP destino DstAddr.

La decisión de tener en cuenta la cantidad de flujos maliciosos por tipo de conexión, además del porcentaje, tiene como fin descartar casos donde un único flujo puede representar el 100 % de las conexiones de ese tipo, y tratarse solamente de un intento de ataque.

4.2.2. Segundo nivel de decisión

En este nivel se debe decidir el label para el conjunto de todos los flujos que van del origen al destino. Para ello se propone una técnica de ensembling similar al stacking. Se considera que tenemos 4 etiquetas para cada conjunto de flujos. Cada una de ellas se obtiene al analizar un subconjunto de características distintas del conjunto de flujos que van de un origen a un destino:

1. TCPELabel: es la etiqueta que se asigna a un conjunto de flujos que van de un origen a un destino en función de los flujos TCP establecidos. Tiene en cuenta la decisión tomada en el primer nivel respecto a este grupo de flujos.

2. TCPNELabel: es la etiqueta que se asigna a un conjunto de flujos que van de un origen a un destino en función de los flujos TCP no establecidos. Tiene en cuenta la decisión tomada en el primer nivel respecto a este grupo de flujos.

3. UDPELabel: es la etiqueta que se asigna a un conjunto de flujos que van de un origen a un destino en función de los flujos UDP establecidos. Tiene en cuenta la decisión tomada en el primer nivel respecto a este grupo de flujos.

4. UDPNELabel: es la etiqueta que se asigna a un conjunto de flujos que van de un origen a un destino en función de los flujos UDP no establecidos. Tiene en cuenta la decisión tomada en el primer nivel respecto a este grupo de flujos.

Para decidir la etiqueta para todo el conjunto de flujos se combinan estas 4 etiquetas. De esta manera, si alguno de los grupos es clasificado como malicioso, entonces todo el conjunto será clasificado como malicioso.

La salida de esta fase es un dataset resultante de agregar, al dataset intermedio, una columna con la predicción de la Fase 2, llamada PredictLabel. Dicha columna indica si el conjunto de flujos intercambiados entre la IP origen SrcAddr y la IP destino DstAddr son parte de una infección. En este caso el valor de PredictLabel será malicious, y normal en caso contrario.

4.3. Fase 3: Ensemble Learning para clasificar hosts

En esta fase el objetivo es decidir si cada IP origen está infectada o no. Para esto se aplica ensembling a partir de la información relacionada a los destinos con que ese host se conecta. Dicha información incluye la predicción de la Fase 2 y los datos de Threat Intelligence (TI), como puede verse en la Figura 4.5.

Para cada dirección IP destino se tiene en cuenta:

- El resultado de combinar la información provista por distintas fuentes de Threat Intelligence (TI), como por ejemplo VirusTotal.

- La decisión del Ensemble Learning de Fase 2, que nos indica si las conexiones desde el origen hacia ese destino forman parte de una infección.

Es necesario consultar la información de Threat Intelligence para cada dirección IP destino de los flujos a analizar. Para cada fuente de Threat Intelligence (feed) se define el criterio a aplicar para usar los datos que la misma provee. Este criterio es acorde al nivel de confianza que se tiene para esa fuente de TI.

Esto se hace para todas las fuentes de TI que se considere incluir. En este caso se usan los módulos de TI que Slips implementa. La lógica definida aquí permite incorporar fácilmente nuevas fuentes de TI al proceso. En este modelo se incorpora VirusTotal como feed. Como se describe en la Sección 2.4.2, Slips cuenta con un módulo llamado VirusTotal Module que se comunica con dicho feed. A partir de la información que obtiene del mismo, el módulo calcula los valores url_ratio, download_ratio, communicating_ratio referrer_ratio. Estos 4 valores son usados por esta fase en el proceso de clasificación.

Para cada IP, la API de VirusTtotal devuelve datos sobre 4 categorías: URL que se resolvieron en la IP, muestras (archivos) descargados de la IP, muestras (archivos) que contienen la IP dada, y muestras (programas) que contactan con la IP. La estructura de los datos es la misma para las 4 categorías. Para cada muestra de una categoría, VirusTotal consulta a los motores antivirus y cuenta cuántos de ellos encuentran la muestra maliciosa. La respuesta tiene dos campos para cada categoría. Estos son el campo "detected_category", que contiene una lista de muestras que fueron encontradas maliciosas por al menos un motor, y el campo "undetected_category", que contiene todas las muestras que ninguno de los motores encontró maliciosas. La respuesta tiene dos campos con puntaje (detectados y no detectados) para cada una de las 4 categorías.

A partir de esta respuesta de la API de VirusTotal, el módulo VirusTotal de Slips calcula la proporción (ratio) de cada categoría. Para calcular la proporción de una categoría se calcula el número global de detecciones (suma de todas las detecciones positivas (detected) de la lista de todas las muestras) y se calcula el número global de pruebas (suma de todas las detecciones positivas y no positivas (detected y undetected) de la lista de todas las muestras). Para la IP consultada se tiene entonces el total de detecciones positivas y total de pruebas para cada categoría. Por lo tanto:

$$url\_ratio = suma\_de\_detecciones\_positivas\_de\_todas\_las\_muestras_{categoriaURL} / numero\_global\_de\_pruebas_{categoriaURL} \tag{4.3}$$

$$url\_ratio = suma\_de\_detecciones\_positivas\_de\_todas\_las\_muestras_{categoriaDownload} / numero\_global\_de\_pruebas_{categoriaDownload} \tag{4.4}$$

$$url\_ratio = suma\_de\_detecciones\_positivas\_de\_todas\_las\_muestras_{categoriaCommunicating} / numero\_global\_de\_pruebas_{categoriaCommunicating} \tag{4.5}$$

$$url\_ratio = suma\_de\_detecciones\_positivas\_de\_todas\_las\_muestras_{categoriaReferrer} / numero\_global\_de\_pruebas_{categoriaReferrer} \tag{4.6}$$

Para incluir la información provista por el módulo VirusTotal, en esta fase se define la variable *VTConfidence*. La cual se calcula de la siguiente manera:

$$VTConfidence = w1 * VTSumUrl + w2 * VTSumDownload + w3 * VTSumCommunic + w4 * VTSumReferrer \tag{4.7}$$

Donde:

- w1, w2, w3, w4 son los pesos asignados para la información provista por VT: url_ratio, download_ratio, communicating_ratio y referrer_ratio respectivamente. Los valores de dichos pesos se determinan en la etapa de entrenamiento.
- VTSumUrl: es la suma de los valores de *url_ratio* que brinda VirusTotal para

todos los destinos del host a clasificar.

- VTSumDownload: es la suma de los valores de *download_ratio* que brinda VirusTotal para todos los destinos del host a clasificar.
- VTSumCommunicating: es la suma de los valores de *commnicating_ratio* que brinda VirusTotal para todos los destinos del host a clasificar.
- VTSumReferrer es la suma de los valores de *referrer_ratio* que brinda VirusTotal para todos los destinos del host a clasificar.

Además se debe cumplir que:

- $0 \leq w1, w2, w3, w4, P2Confidence \leq 1$.
- $w1 + w2 + w3 + w4 = 1$ por ser una distribución de probabilidad.
- $0 \leq VTConfidence \leq 1$.

Se debe normalizar VTConfidence para que varíe entre 0 y 1. Para ello se elige min-max que es mecanismo simple, por lo tanto para calcular VTConfidence:

$$\begin{aligned} VTConfidence = VTConfidence - min(VTConfidence) \\ /(max(VTConfidence) - min(VTConfidence) \end{aligned} \tag{4.8}$$

Si se sabe que:

- $min(VTConfidence) = 0$
- $max(VTConfidence) = 100$

Entonces al normalizar el valor de VTConfidence se tiene:

$$VTConfidence = VTConfidence/100 \tag{4.9}$$

Se define también P2Confidence, a fin de incluir los resultados de la Fase 2 para los destinos del host a analizar. P2Confidence se calcula en base a la cantidad de conjuntos de flujos etiquetados como maliciosos en la Fase 2, de acuerdo al siguiente criterio:

- Si $Cantidad_{ConjuntosMaliciosos} \leq threshold3 \rightarrow P2Confidence = 0{,}59$
- Si $threshold2 \leq Cantidad_{ConjuntosMaliciosos} \leq threshold3 \rightarrow P2Confidence = 0{,}55$
- $Sithreshold1 \leq Cantidad_{ConjuntosMaliciosos} \leq threshold2 \rightarrow P2Confidence = 0{,}50$
- Si $Cantidad_{ConjuntosMaliciosos} \leq threshold1 \rightarrow P2Confidence = 0$

Donde los valores a usar para threshold1, threshold2 y threshold 3 son los umbrales usados para para asignar un valor a P2Confidence basado en la cantidad conjuntos de flujos (asociados a un destino) cuyo label es malicioso. Dichos valores se determinan en la etapa de entrenamiento de esta fase. P2Confidence es un valor $0 \leq P2Confidence \leq 1$, como puede observarse en la figura 4.6.

Por último en esta fase, se realiza el ensembling entre la información de TI, respresentada en este caso por VTConfidence, y la resultante de la Fase 2. De esta manera se decide el label para cada dirección IP origen, teniendo en cuenta lo que se conoce de sus conexiones hacia cada uno de sus destino (Fase 2) y la información de TI de los mismos. Se tiene entonces:

$$Phase3Confidence = P2Confidence + VTConfidence \tag{4.10}$$

Para determinar el label de Fase 3 se evalúa el valor de Phase3Confidence. Si el mismo es mayor a un peso representado por la variable p3weight, se clasificará al host como infectado (malicious). Mientras que si el valor de Phase3Confidence es menor o igual a dicho peso, se clasificará al host como no infectado (normal). Es decir:

- Si $Phase3Confidence \geq p3weight \rightarrow Phase3Label = malicious$
- Si $Phase3Confidence < p3weight \rightarrow Phase3Label = normal$

Donde p3weight también se determina en la etapa de entrenamiento de esta fase, siendo el valor adecuado para el mismo aquel que dé los mejores resultados, reduciendo los falsos positivos.

Se tiene como resultado de esta fase la decisión para cada IP Origen. El label será *malicious* o *normal*, indicando que el host está infectado o no. Esta decisión de la Fase 3, respecto a la IP origen, representa el resultado del proceso completo.

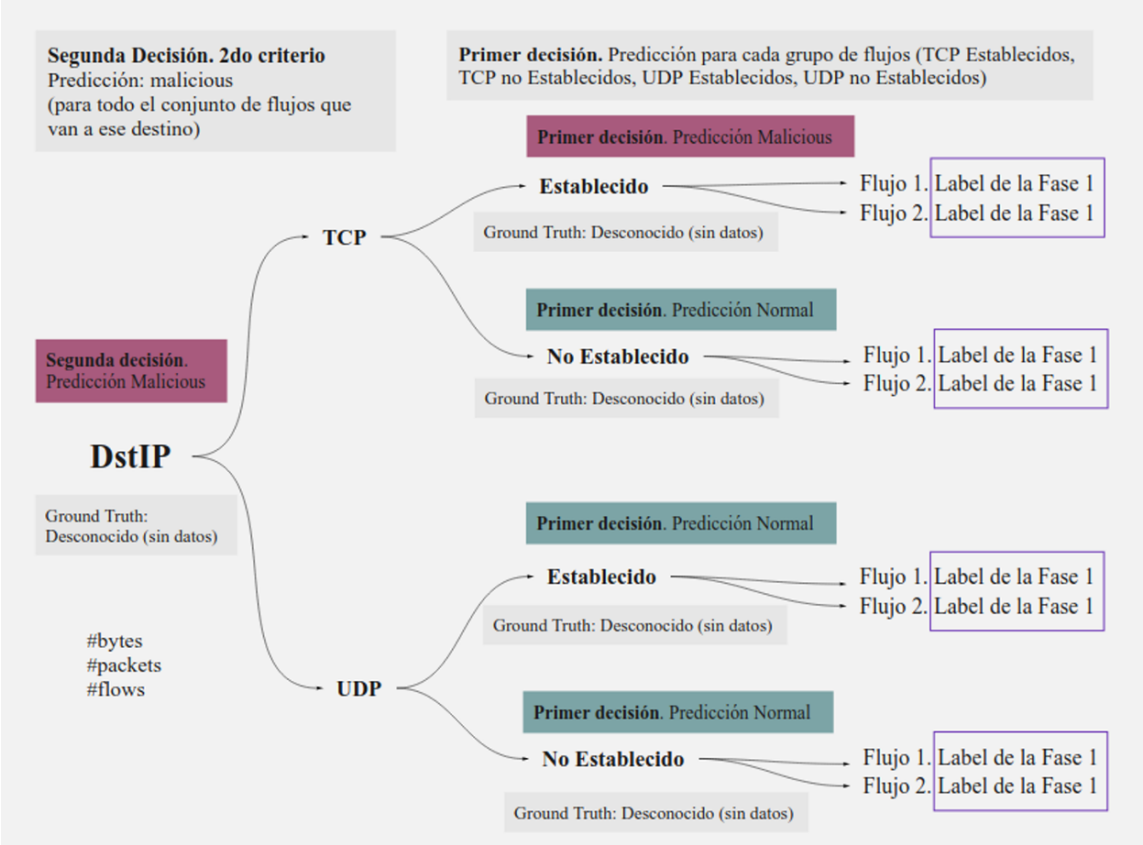

Figura 4.4: Fase 2 del Ensembling. En la Fase 2 se aplica *Ensembling* en dos etapas (o niveles de decisión. En la primer etapa se toma una decisión por grupo de flujos que representan el tráfico TCP establecido, TCP no establecido, UDP establecido y UDP no establecido. En la segunda etapa se toma una decisión respecto a todas las conexiones entre un origen y un destino. El label se asocia a cada uno de los destinos de un mismo host origen.

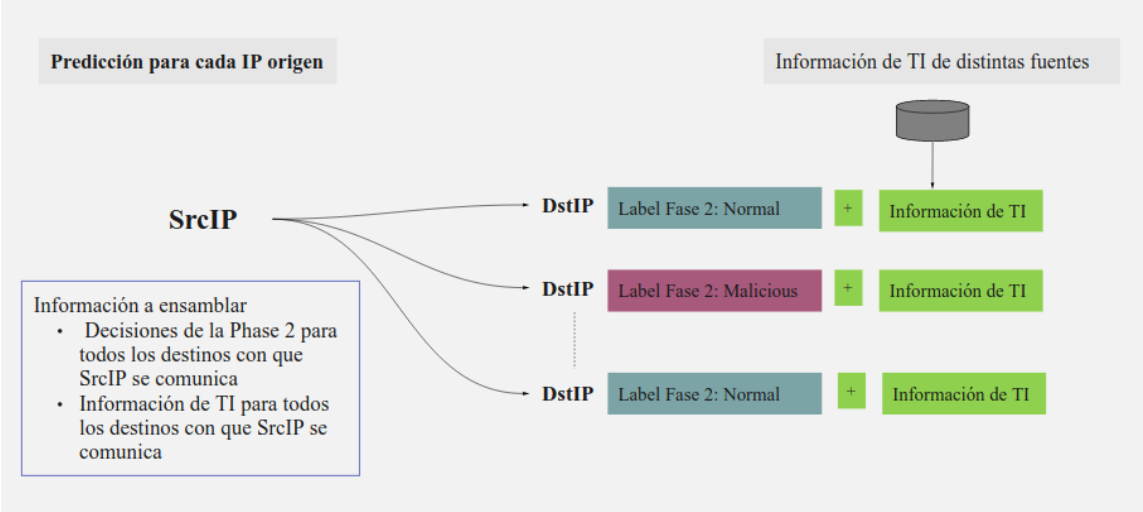

Figura 4.5: Fase 3 del Ensembling. En esta fase se combinan la información provista por la Fase 2 y la información provista por TI, de los destinos con que cada origen se comunica. En función de ese ensembling se decide si cada host origen está infectado o no.

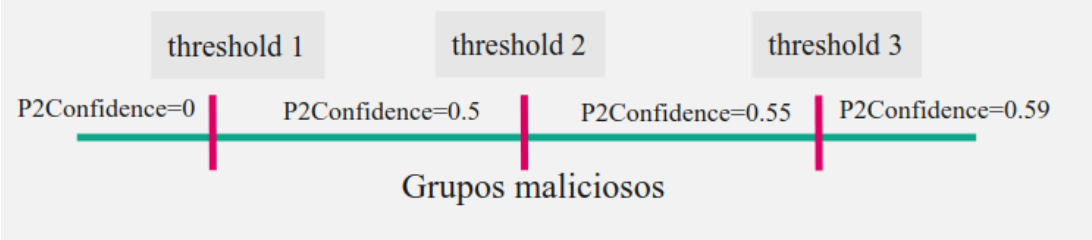

Figura 4.6: Relación entre P2Confidence y la cantidad de grupos marcados como maliciosos. Los thresholds threshold1, threshold2 y threshold3 se usan para determinar el valor de P2Confidence en función de la cantidad de destinos marcados como maliciosos en la Fase 2.

Capítulo 5

Datasets

Para realizar los experimentos se utilizaron datasets del Laboratorio Stratosphere [49] y datasets "ad hoc". Estos últimos fueron generados en las distintas fases, a fin de utilizarse para el procesamiento de los datos (intermedios) o como interfaz con la siguiente fase (resultantes) y serán descriptos en el Capítulo 6.

Los datasets de Stratosphere son creados a partir de capturas reales de tráfico normal, malicioso y mixto. Esta última categoría incluye capturas de tráfico de red normal y malicioso simultáneamente.

Las capturas de malware se llevaron a cabo en el marco del proyecto *Malware Capture Facility Project* [31], en el cual se monitorea continuamente el panorama de amenazas para detectar nuevas amenazas emergentes, recuperar muestras maliciosas y ejecutarlas en sus instalaciones a fin de capturar el tráfico. Se crean escenarios de los cuales participan máquinas infectadas y máquinas no infectadas, y se captura tráfico de red utilizando distintas herramientas. A partir de las capturas se generan datasets que son publicados para la comunidad. Cada escenario tiene asociado un identificador que lo distingue dentro del repositorio de datasets de Stratosphere.

El rendimiento de los algoritmos de aprendizaje automático debe verificarse con datos reales. Especialmente en ciberseguridad, es realmente importante contar con datos representativos del tráfico de red, que incluya actividad normal y actividad maliciosa, que incluya distintos tipos de ataques y las distintas fases de los mismos.

Para hacer una buena verificación, necesitamos tres tipos de tráfico: malware, normal y "background". El tráfico de malware incluye todo lo que se quiere detec-

tar, especialmente las conexiones de "comando y control". El tráfico normal es muy importante para descubrir el rendimiento real de nuestros algoritmos calculando los falsos positivos y los verdaderos negativos. El tráfico de "background" es tráfico no malicioso que se transmite al mismo tiempo que el tráfico malicioso. El mismo necesario para saturar los algoritmos, verificar su rendimiento de memoria y velocidad, y además probar si el algoritmo se confunde con los datos.

Las capturas mixtas proporcionan un escenario real donde una máquina no está infectada, luego se infecta y después de un tiempo deja de estar infectada. Este tipo de escenario facilita la prueba de los algoritmos y modelos de aprendizaje automático.

Para realizar el training y el testing de la Fase 1 se usaron tres datasets provistos por Stratosphere [49], a partir de ahora Dataset 1, Dataset 2 y Dataset 3.

Estos datasets contienen información de flujos de red. Para cada flujo se obtienen los siguientes datos:

- Dir: sentido de la conexión.
- Dport: puerto destino.
- DstAddr: dirección IP destino.
- Dur: duración de la conexión.
- Proto: protocolo de capa de transporte que usa la conexión.
- Sport: puerto origen.
- SrcAddr: dirección IP origen.
- SrcBytes: cantidad de bytes que envía el origen.
- SrcPkts: cantidad de paquetes que envía el origen.
- StartTime: timestamp correspondiente al inicio de la conexión.
- State: estado de la conexión. Los posibles valores para este campo se detallan en [44].
- TotBytes: cantidad de bytes transmitidos a través de esa conexión.

- TotPkts: cantidad de paquetes transmitidos a través de esa conexión.
- dTos: type of service del destino.
- sTos: type of service del origen.
- Label: etiqueta que indica malware o normal.

En el caso del Dataset 1, además de los campos detallados, se agrega un campo CCDetector (con información adicional del nombre de la botnet a la que se relaciona la conexión).

5.1. Dataset 1

Características del Dataset 1:

1. Es un dataset creado a partir de capturas mixtas (*Mixed dataset*)
2. Nombre del archivo: capture20110818-binetflow.csv
3. Identificador del escenario: CTU-Malware-Capture-Botnet-51
4. Disponible en: *https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-51/detailed-bidirectional-flow-labels/capture20110818.binetflow.labeled*
5. Los flujos que forman parte del dataset provienen de:
   - 6 máquinas no infectadas.
   - 10 máquinas infectadas.
6. Período de tiempo que dura la captura de tráfico: 4.75hs.

Se analizó el contenido de este dataset. El mismo contiene flujos de red obtenidos de la actividad de hosts infectados, hosts no infectados y hosts cuyo estado es desconocido. Se denomina hosts desconocidos a aquellos hosts para los cuales no se ha establecido con certeza si están o no infectados), como puede observarse en la Tabla 5.1.

Hosts infectados		
Con flujos maliciosos	Con flujos normales	Con flujos normales y maliciosos
10 hosts	0 hosts	0 hosts
Hosts no infectados		
Con flujos maliciosos	Con flujos normales	Con flujos normales y maliciosos
0 hosts	6 host	0 hosts
Hosts desconocidos		
Con flujos maliciosos	Con flujos normales	Con flujos normales y maliciosos
52 hosts	4 hosts	3 hosts

Tabla 5.1: Análisis de los hosts y sus flujos en del Dataset 1. En primer lugar se muestra la cantidad de hosts infectados que solo tienen flujos maliciosos, la cantidad de hosts infectados que solo tienen flujos normales y la cantidad de hosts infectados que tienen tanto flujos maliciosos como normales. En segundo lugar se muestra la cantidad de hosts no infectados que solo tienen flujos maliciosos, la cantidad de hosts no infectados que solo tienen flujos normales y la cantidad de hosts no infectados que tienen tanto flujos maliciosos como normales. En tercer lugar se muestra la cantidad de hosts desconocidos que solo tienen flujos maliciosos, la cantidad de hosts desconocidos que solo tienen flujos normales y la cantidad de hosts desconocidos que tienen tanto flujos maliciosos como normales.

5.2. Dataset 2

Características del Dataset 2:

1. Es un dataset creado a partir de capturas mixtas (*Mixed dataset*)
2. Nombre del archivo: capture20110819.binetflow
3. Identificador del escenario: CTU-Malware-Capture-Botnet-53
4. Disponible en: *https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-53/detailed-bidirectional-flow-labels/capture20110819.binetflow*
5. Los flujos que forman parte del dataset provienen de:
   - 6 máquinas no infectadas.
   - 3 máquinas infectadas.
6. Período de tiempo que dura la captura de tráfico: 1 hora, 13 minutos y 21 segundos.

Se analizó el contenido de este dataset. El mismo contiene flujos de red obtenidos de la actividad de hosts infectados, hosts no infectados y hosts cuyo estado es desconocido, como puede observarse en la Tabla 5.2.

5.3. Dataset 3

Este dataset fue usado para realizar el training y el testing de la Fase 1.
Características del Dataset 3:

1. Es un dataset creado a partir de capturas mixtas (*Mixed dataset*)
2. Nombre del archivo: 2018-05-03_win11.binetflow
3. Identificador del escenario: CTU-Malware-Capture-Botnet-351-1
4. Disponible en: *https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-351-1/2018-05-03_win11.binetflow*

Hosts infectados		
Con flujos maliciosos	Con flujos normales	Con flujos normales y maliciosos
3 hosts	0 hosts	0 hosts
Hosts no infectados		
solo flujos maliciosos	solo flujos normales	Flujos normales y maliciosos
0 hosts	6 host	0 hosts
Hosts desconocidos		
solo flujos maliciosos	solo flujos normales	Con flujos normales y maliciosos
58 hosts	6 hosts	2 hosts

Tabla 5.2: Análisis de los hosts y sus flujos en el Dataset 2. En primer lugar se muestra la cantidad de hosts infectados que solo tienen flujos maliciosos, la cantidad de hosts infectados que solo tienen flujos normales y la cantidad de hosts infectados que tienen tanto flujos maliciosos como normales. En segundo lugar se muestra la cantidad de hosts no infectados que solo tienen flujos maliciosos, la cantidad de hosts no infectados que solo tienen flujos normales y la cantidad de hosts no infectados que tienen tanto flujos maliciosos como normales. En tercer lugar se muestra la cantidad de hosts desconocidos que solo tienen flujos maliciosos, la cantidad de hosts desconocidos que solo tienen flujos normales y la cantidad de hosts desconocidos que tienen tanto flujos maliciosos como normales.

Hosts infectados		
Con flujos maliciosos	Con flujos normales	Con flujos normales y maliciosos
0 hosts	0 hosts	1 host
Hosts no infectados		
Con flujos maliciosos	Con flujos normales	Flujos normales y maliciosos
0 hosts	1 host	0 hosts
Hosts desconocidos		
Con flujos maliciosos	Con flujos normales	Con flujos normales y maliciosos
0 hosts	294 hosts	6 hosts

Tabla 5.3: Análisis de los hosts y sus flujos en el Dataset 3. En primer lugar se muestra la cantidad de hosts infectados que solo tienen flujos maliciosos, la cantidad de hosts infectados que solo tienen flujos normales y la cantidad de hosts infectados que tienen tanto flujos maliciosos como normales. En segundo lugar se muestra la cantidad de hosts no infectados que solo tienen flujos maliciosos, la cantidad de hosts no infectados que solo tienen flujos normales y la cantidad de hosts no infectados que tienen tanto flujos maliciosos como normales. En tercer lugar se muestra la cantidad de hosts desconocidos que solo tienen flujos maliciosos, la cantidad de hosts desconocidos que solo tienen flujos normales y la cantidad de hosts desconocidos que tienen tanto flujos maliciosos como normales.

5. Los flujos que forman parte del dataset provienen de:

   - 1 máquina no infectada.
   - 1 máquina infectada.

6. Período de tiempo que dura la captura de tráfico: 16 días aproximadamente. Desde el 17 de abril 19:18:29 al viernes 4 de mayo 00:17:48 CEST 2018).

Se analizó el contenido de este dataset. El mismo contiene flujos de red obtenidos de la actividad de hosts infectados, hosts no infectados y hosts cuyo estado es desconocido, como puede observarse en la Tabla 5.3.

Capítulo 6

Experimentos

A fin de validar el modelo definido se llevaron a cabo experimentos que permiten testear las distintas fases del mismo con datasets reales, obtener conclusiones a partir de los resultados y ajustar los criterios de clasificación de cada fase.

Los experimentos son el núcleo de este trabajo. A partir de sus resultados se analizan las ventajas de aplicar ensembling con el objetivo de mejorar la detección de hosts infectados. Esto se realiza como etapa previa a la implementación de un módulo que integra las 3 fases.

En nuestro modelo, cada fase consta de dos etapas: una de *training* (entrenamiento) y una de *testing* (es decir, pruebas de verificación de resultados). En la etapa de *training* se entrenan los algoritmos diseñados, a fin de establecer el mejor modelo para cada una de las fases. En cada fase hay distintas variables que forman parte del criterio de clasificación, de cuyos valores dependen los resultados. Para establecer los mismos es que se realiza el entrenamiento, usando distintos valores, y se analizan los resultados.

Para determinar el mejor modelo se usan métricas. Las mismas pueden calcularse a partir de la matriz de confusión resultante. Una matriz de confusión es una representación del rendimiento de los modelos de clasificación [40]. La matriz muestra el número de casos clasificados correcta e incorrectamente, en comparación con labels reales, conocidos como *Ground Truth* o verdad conocida).

Una de las ventajas de usar la matriz de confusión como herramienta de evaluación es que permite un análisis detallado a partir del cual se obtienen distintas

métricas. De este conjunto de métricas se debe elegir cuál utilizar para la comparación de los modelos.

El rendimiento de un clasificador binario se resume en una matriz de confusión como se muestra en la Figura 6.1. La misma tabula en forma cruzada casos predichos (predicción) y observados (verdad conocida o Ground Truth) en cuatro opciones:

1. Verdadero Positivo (TP): se predice correctamente un label positivo.
2. Verdadero negativo (TN): se predice correctamente un label negativo.
3. Falso positivo (FP): se predice un label positivo, y era falso.
4. Falso negativo (FN): se predice un label negativo, y era verdadero.

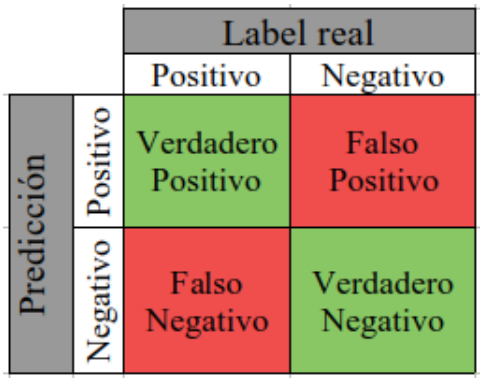

Figura 6.1: Matriz de confusión para clasificación binaria. Las predicciones representan el resultado del clasificador y el label real es el que se conoce para los datos de prueba (también llamado Ground Truth). En los cuadrantes se representan los siguientes valores: Verdadero Positivo representa la cantidad de predicciones positivas correctas, Falso Positivo representa la cantidad de predicciones positivas incorrectas, Falso Negativo representa la cantidad de predicciones negativas incorrectas, Verdadero Negativo representa la cantidad de predicciones negativas correctas

Las métricas usadas, que se obtienen a partir de cada matriz de confusión son:

1. Accuracy $= \dfrac{TP + TN}{TP + FP + TN + FN}$
2. F1Score (Balanced option) $= \dfrac{2 * TP}{2TP + FP + FN}$

3. FPR (False positive rate) = $\frac{FP}{FP+TN}$
4. TPR (True positive rate) = $\frac{TP}{TP+FN}$

En la etapa de testing se valida el modelo elegido, con otra muestra de datos.

Los clasificadores implementados en esta tesis son clasificadores binarios. En cada fase el criterio es el siguiente:

- En la Fase 1 se usan los labels malicious y normal. En la misma una detección se considera positiva cuando un flujo se clasifica como malicious.
- En la Fase 2 se usan los labels malicious y normal. En esta etapa una detección se considera positiva cuando el conjunto de flujos que van de un origen a un destino se clasifica como malicious.
- En la Fase 3 se usan los labels infected y normal. En la misma una detección se considera positiva cuando una IP origen se clasifica como infected.

En las siguientes secciones se describen para cada fase: los experimentos realizados, detallando su objetivo, las etapas de entrenamiento y testing, y los resultados obtenidos.

6.1. Experimentos de la Fase 1

6.1.1. Primera etapa de pruebas de la Fase 1

Como primera etapa de este proyecto, se realizan pruebas de rendimiento de los algoritmos de Ensemble Learning para detectar flujos maliciosos, y se comparó su precisión con la de un conjunto de algoritmos de aprendizaje base.

Para llevar a cabo las pruebas, se utiliza el dataset Dataset 1 descripto en la Sección 5.1. Es un conjunto de datos mixto con labels correspondientes a tráfico normal y a tráfico de malicious, de una botnet conocida como Rbot.

Se prueban los siguientes algoritmos de ML: *Logistic Regression (LR)*, *Naive Bayes* (NB), *Random Forest* (RF), *K Nearest Neighbor* (KNN) y *Decision Tree* (DT) [3]. Y las técnicas de ensembling learning utilizadas fueron: *Hard Voting* o

Algoritmo	**Precisión (Accuracy)**
Logistic Regression	99,45 % (+/-0,641)
Random Forest	99,999 % (+/-0,004)
Naive Bayes	98,989 % (+/-1,246)
K Nearest Neighbor	99,999 % (+/-0,022)
Decision Tree	99,999 % (+/-0,006)

Tabla 6.1: Resultados de los experimentos aplicando los algoritmos base *Logistic Regression, Random Forest, Naive Bayes, K Nearest Neighbor y Decision Tree*, sin *Ensemble Learning*, usando el Dataset 1.

Majority Voting, *Soft Voting* (usando la suma de las probabilidades pronosticadas), *Weighted Voting* o *Votación Ponderada*, *Bagging* y *Boosting*.

Se implementan pruebas con la biblioteca Scikit-learn [17] y se utiliza como métrica la *Accuracy* o precisión, obtenida al aplicar la función cross_val_score al modelo. Esta función realiza una división simple de los datos del dataset en un subconjunto para entrenamiento y un subconjunto para el testeo [11].

En la Tabla 6.1 se muestra la precisión obtenida para los siguientes algoritmos de aprendizaje automático: *Logistic Regression, Random Forest, Naive Bayes, K Nearest Neighbor y Decision Tree*. Luego se comparan estos valores con los resultantes de aplicar distintas técnicas de Ensemble Learning para combinarlos.

En la Tabla 6.2 se puede observar que al aplicar *Majority Voting* y *Soft Voting* se obtienen resultados similares. Ambos mejoran el *Logistic Regression* y el *Naive Bayes* pero no el *Random Forest*. Esto sucede porque, cuando se combina *Random Forest* que da buenos resultados con otros dos algoritmos que son peores, al aplicar *Majority Voting* se decide mal. Mientras que al aplicar *Weighted Voting* no se presentan mejoras para *Random Forest*, aunque sí muestra mejoras en comparación con las otras técnicas de votación probadas. Sin embargo la precisión es más cercana a la del Random Forest sin técnicas de ensembling. Los mejores resultados en la clasificación se obtienen dando el mayor peso a *Random Forest* y el menor peso a los otros dos algoritmos.

Luego se hacen experimentos con *Weighted Voting* reemplazando *Random Forest* por *K Nearest Neighbor* y *Decision Tree*. Los resultados de combinar *Logistic Regression*, KNN y *Naive Bayes* pueden observarse en la Tabla 6.3 y aquellos de combinar

Técnica de Ensemble Learning	**Precisión (Accuracy)**
Majority Voting	99,731 % (+/- 0,286)
Soft Voting	99,753 % (+/- 0,295)
Weighted Voting (LR=1, RF=3 y NB=1)	99,993 % (+/-0,004)
Weighted Voting (LR=1, RF=3 y NB=2)	99,991 % (+/-0,007)
Weighted Voting (LR=2, RF=3 y NB=1)	99,988 % (+/-0,009)

Tabla 6.2: Resultados de los Experimentos usando como algoritmos base *Logistic Regression*, *Random Forest* y *Naive Bayes* y aplicando como Ensemble Learning distintas variantes de las técnicas de votación. La primera fila muestra el valor de Accuracy al aplicar Majority Voting. La segunda fila muestra muestra el valor de Accuracy al aplicar Soft Voting. Las siguientes filas muestran el valor de Accuracy para *Weighted Voting* con distintos pesos asignados a Logistic Regression, Random Forest y Naive Bayes. Estas filas corresponden a los pesos para los que se obtuvieron mejores valores de Accuracy.

Técnica de Ensemble Learning	**Precisión (Accuracy)**
Weighted Voting (LR=1, KNN=2 y NB=1)	99,975 % (+/-0,025)
Weighted Voting (LR=1, KNN=3 y NB=1)	99,975 % (+/-0,025)
Weighted Voting (LR=2, KNN=3 y NB=1)	99,974 % (+/-0,025)

Tabla 6.3: Resultados de los experimentos usando como algoritmos base *Logistic Regression*, *K Nearest Neighbor* y *Naive Bayes*, y aplicando *Weighted Voting* como técnica de Ensembling. En estos experimentos se reemplazó el algoritmo Random Forest por K Nearest Neighbor.

Logistic Regression, *Decision Tree*y *Naive Bayes* se muestran en la Tabla 6.4. En ambos casos se pueden observar mejoras al aplicar Ensemble Learning con respecto a no aplicarlo. Si bien las mejoras al aplicar *Weighted Voting* no son significativas, las hay para todos los algoritmos involucrados.

Al aplicar Bagging se obtienen mejoras poco significativas para *Random Forest*, KNN y DT, y ninguna mejora para *Logistic Regression* y *Naive Bayes*. como puede observarse en la Tabla 6.5.

Por otro lado, *Adaboost* mejora para *Decision Tree*, es igual para *Random Forest* y no mejora ni *Naive Bayes* ni *Logistic Regression*, como se muestra en la Tabla 6.6.

Al cierre de esta fase preliminar se concluye que usar *Weighted Voting* o *Vota-*

Técnica de Ensemble Learning	**Precisión (Accuracy)**
Weighted Voting (LR=1, DT=3 y NB=1)	99,993 % (+/-0,006)
Weighted Voting (LR=1, DT=3 y NB=2)	99,991 % (+/-0,007)
Weighted Voting (LR=2, DT=3 y NB=1)	99,993 % (+/-0,006)

Tabla 6.4: Resultados de los experimentos usando como algoritmos base *Logistic Regression*, *Decision Tree* y *Naive Bayes* y aplicando *Weighted Voting* como técnica de Ensembling. En estos experimentos se reemplazó el algoritmo Random Forest por Decision Tree.

Algoritmo	**Accuracy (sin bagging)**	**Accuracy (con bagging)**
Logistic Regression	99,457 % (+/- 0,641)	99,419 % (+/-1,731)
Random Forest	99,995 % (+/-0,004))	99,997 % (+/-0,013)
Naive Bayes	98,997 % (+/-1,246)	99,165 % (+/-2,727)
K Nearest Neighbor	99,975 % (+/-0,022)	99,98 % (+/-0,033)
Decision Tree	99,993 % (+/-0,006))	99,997 % (+/-0,013)

Tabla 6.5: Experimentos usando como algoritmos base Logistic Regression, Random Forest, Naive Bayes, K Nearest Neighbor y Decision Tree y aplicando *Bagging* como técnica de Ensembling (usando un seed value = 8).

Algoritmo	**Accuracy (sin boosting)**	**Accuracy (con boosting)**
Logistic Regression	99,457 % (+/-0,641)	98,557 % (+/-1,707)
Random Forest	99,995 % (+/-0,004)	99,995 % (+/-0,005)
Naive Bayes	98,997 % (+/-1,246)	77,884 % (+/-29,319)
Decision Tree	99,993 % (+/-0,006)	99,993 % (+/-0,013)

Tabla 6.6: Experimentos usando como algoritmos base *Logistic Regression*, *Random Forest*, *Naive Bayes* y *Decision Tree* y aplicando *Boosting (Adaboost)* como técnica de Ensembling.

ción Ponderada resulta la técnica más adecuada de ensembling dado que mejora los resultados de los algoritmos base, salvo el Random Forest. Para que esta mejora se produzca, es necesario elegir los pesos en forma adecuada en la fase de entrenamiento. Los resultados de esta etapa forman parte del trabajo publicado en el Congreso Argentino de Ciencias de la Computación (CACIC 2019) [20] durante la realización de esta tesis.

En esta fase se agrega el campo "GroundTruth" (verdad conocida) el cual indica lo que se sabe respecto al flujo. Al mismo se le asigna el valor: normal, malicious o unknow. Esta información se establece a partir del valor de los campos "Label" y "CCDetector" del dataset original, *Dataset 1.*

6.1.2. Segunda etapa de pruebas de la Fase 1

La primera etapa de pruebas permite llevar a cabo los primeros experimentos con algoritmos de aprendizaje automático, en particular de Ensemble Learning, con el objetivo de detectar flujos maliciosos.

Con estos resultados y a fin de avanzar en el proceso, se realizan otros experimentos con un dataset más grande. El mismo se arma a partir de los datasets *Dataset 1, Dataset 2 y Dataset 3* descriptos en el Capítulo 5, y lo llamaremos *Dataset 1-2-3* . Ello permite contar con mayor cantidad y heterogeneidad de datos, a fin de obtener mejores resultados. Dicho dataset constituye la entrada para los experimentos de todo el proceso de ensembling implementado en esta tesis.

En los experimentos descriptos en esta sección se usa la función cross_validation o validación cruzada. La misma se utiliza principalmente en el aprendizaje automático aplicado para estimar la habilidad de un modelo en datos no vistos [7]. Es un método popular porque es simple de entender y generalmente resulta en una estimación menos sesgada que otros métodos, como la división simple de entrenamiento y testeo.

En primer lugar, se realizan nuevos experimentos con el algoritmo *Weigthed Voting*, para determinar los pesos a asignar a cada clasificador. Para ello se corre un proceso probando distintos pesos para cada uno de los clasificadores. Se compararon los resultados para elegir el mejor modelo.

En la Tabla 6.7 se puede observar que los mejores resultados, al igual que en la etapa anterior, se dan asignando los pesos de la siguiente manera: 1 a Logistic

Peso para *LR*	**Peso para *RF***	**Peso para *NB***	**Accuracy**
1	1	2	59 % (+/- 0.00441)
1	1	3	57,2 % (+/- 0.00321)
1	2	1	99,8 % (+/- 0.00025)
1	2	2	99,5 % (+/- 0.00020)
1	2	3	59,7 % (+/- 0.00402)
1	**3**	**1**	**99,8 % (+/- 0.00024)**
1	3	2	99,6 % (+/- 0.00023)
2	1	1	99,4 % (+/- 0.00026)
2	1	2	99,4 % (+/- 0.00096)
2	1	3	58,6 % (+/- 0.00401)
2	2	1	99,5 % (+/- 0.000315)
2	2	3	99,4 % (+/- 0.000988)
2	3	1	99,8 % (+/- 0.000239)
2	3	2	99,6 % (+/- 0.000282)
2	3	3	99,5 % (+/- 0.000176)
3	1	3	93,2 % (+/- 0.00139)
3	1	1	99,4 % (+/- 0.00095)
3	1	2	99,2 % (+/- 0.00102)
3	2	3	99,4 % (+/- 0.00039)
3	2	1	99,5 % (+/- 0.00015)
3	2	2	99,5 % (+/- 0.00093)
3	3	1	99,5 % (+/- 0.00042)
3	3	2	99,5 % (+/- 0.00029)

Tabla 6.7: Experimentos usando *Logistic Regression (LR)*, *Random Forest (RF)* y *Naive Bayes (NB)* como algoritmos base y aplicando *Weighted Voting* como técnicas de Ensembling. Se prueban todas las combinaciones posibles de pesos con valores 1, 2 y 3. El mejor valor de Accuracy se obtiene para la combinación de pesos 1 para *Logistic Regression*, 3 para *Random Forest* y 1 para *Naive Bayes*.

	Label real Malicious	**Label real Normal**
Predicción Malicious	86.136	10.362
Predicción Normal	4.136	233.507

Tabla 6.8: Matriz de confusión para los experimentos realizados usando Logistic Regression para el dataseet *Dataset1-2-3*.

	Label real Malicious	**Label real Normal**
Predicción Malicious	127.497	4
Predicción Normal	0	243.865

Tabla 6.9: Matriz de confusión para los experimentos usando *Random Forest* para el dataset *Dataset1-2-3*.

Regression, 3 a Random Forest y 1 a Naive Bayes, al usar Weighted Voting como algoritmo de Ensemble Learnig. Por lo tanto, ese es el modelo elegido para realizar el ensembling de la Fase 1.

Una vez decidido el modelo a usar se corren nuevamente experimentos para los clasificadores *Logistic Regression*, *Random Forest* y *Naive Bayes*, esta vez usando el *Dataset1-2-3* en lugar de *Dataset1* para evaluar los resultados obtenidos sin Ensemble Learning y compararlos con aquellos resultantes de aplicar Ensemble Learning con el modelo elegido.

Las matrices de confusión obtenidas para los clasificadores *Logistic Regression*, *Random Forest* y *Naive Bayes* se muestran en las Tablas 6.8, 6.9 y 6.10, respectivamente. Mientras que en la Tabla 6.11 se puede observar aquella resultante de aplicar el modelo elegido, Weighted Voting con pesos: 1 (para *Logistic Regression*), 3 (para *Random Forest*) y 1 (para NB).

A partir de las mismas se calculan las métricas Accuracy, F1Score, FPR y TPR,

	Label real Malicious	**Label Normal**
Predicción Malicious	1.767	266
Predicción Normal	125.730	243.603

Tabla 6.10: Matriz de confusión para los experimentos usando Naive Bayes para el dataset *Dataset1-2-3*.

	Label real Malicious	**Label real Normal**
Predicción Malicious	127.456	0
Predicción Normal	41	243.869

Tabla 6.11: Matriz de confusión para los experimentos usando *Weighted Voting* con pesos: 1 (para *Logistic Regression*), 3 (para *Random Forest*) y 1 (para *Naive Bayes*) para el dataset *Dataset1-2-3*.

Algoritmo Logistic Regression			
Accuracy	F1Score	False Positive Rate	True Positive Rate
95,66 %	92,24 %	4,25 %	95,42 %
Algoritmo Random Forest			
Accuracy	F1Score	False Positive Rate	True Positive Rate
100 %	100 %	0 %	100 %
Algoritmo Naive Bayes			
Accuracy	F1Score	False Positive Rate	True Positive Rate
66,07 %	2,73 %	0,11 %	1,39 %
Algoritmo Weighted Voting (con pesos (LR=1, RF=3, NB=1)			
Accuracy	F1Score	False Positive Rate	True Positive Rate
99,99 %	99,98 %	0 %	99,97 %

Tabla 6.12: Valores de las métricas Accuracy, F1Score, False Positive Rate y True Positive Rate, resultantes de aplicar los algoritmos *Logistic Regression, Random Forest, Naive Bayes*, y de combinarlos con *Weighted Voting*, para el dataset *Dataset1-2-3*.

las cuales se muestran en la Tabla 6.12. Se concluye que usar Weighted Voting asignando los pesos 1 (para *Logistic Regression*), 3 (para *Random Forest*) y 1 (para *Naive Bayes*), da buenos resultados. Si bien no mejora el Random Forest cuyos resultados son óptimos, mantiene la tasa de falsos positivos. Minimizar los falsos positivos es uno de los objetivos de nuestras detecciones. Los experimentos muestran también que, del entrenamiento, surge una distribución de pesos que determina que decida el algoritmo base que mejor predice, siendo *Random Forest* en este caso.

StartTime	Dur	Proto	SrcAddr	Sport	Dir
2011-08-18 10:22:00.042929	0,312377	tcp	147.32.84.170	53856	->
2011-08-18 11:04:04.942692	3575,900391	tcp	147.32.84.208	1039	->
2011-08-18 12:33:13.109033	0	udp	147.32.80.9	53	->
2011-08-18 14:43:52.601800	1198,607056	icmp	147.32.84.208	0x0008	->

DstAddr	Dport	State	sTos	dTos
199.7.59.72	80	FSPA_FSPA	0.0	0.0
213.179.58.83	6667	SPA_SPA	0.0	0.0
147.32.84.25	44072	INT	0.0	0.0
147.32.96.69	0x004b	ECO	0.0	0.0

TotPkts	TotBytes	SrcBytes	SrcPkts	Label	groundTruth
12	2.614	798	0.0	malicious	malicious
168	28.134	5.579	0.0	malicious	malicious
1	152	152	0.0	malicious	malicious
25	26.650	26.650	0.0	malicious	normal

Figura 6.2: Dataset resultante de la Fase 1 (ejemplo que muestra 4 filas) con los campos del dataset original: StartTime, Dur, Proto, SrcAddr, Sport, Dir, DstAddr, Dport, State, sTos, dTos, TotPkts, TotBytes, SrcBytes, SrcPkts). Label es la predicción final para el flujo resultante de aplicar el Ensembling en esta Fase y el GroundTruth representa el label real.

6.1.3. Dataset resultante de la Fase 1

Al dataset *Dataset 1-2-3*, se agregan 4 columnas, 3 de ellas representan el label resultante de cada clasificador: Label-LogisticRegression, Label-RandomForest, Label-NaiveBayes. Y la última, GroundTruth, la verdad conocida para el flujo, que proviene del label para ese flujo en los dataset originales.

Los valores de Label-LogisticRegression, Label-RandomForest y Label-NaiveBayes se combinan usando *Weighted Voting* con peso para así obtener el label resultante del ensembling, el cual se agrega al dataset *Dataset 1-2-3* como puede verse en la Figura 6.2. De esta manera se tiene una única decisión por flujo. Dicho dataset constituye la entrada para la Fase 2.

6.2. Experimentos Fase 2

Esta fase tiene como fin decidir si el conjunto de flujos que van desde un origen a un destino forman parte de una infección. Es decir, si hay flujos asociados a actividad maliciosa, entre los flujos de red que intercambian esos hosts.

En esta fase se trabaja con el dataset resultante de la Fase 1, *Dataset 1-2-3*. En este dataset se tiene ahora un único label (malicious o normal por flujo).

Se crea otro dataset, a partir del resultante de la Fase 1, al cual se insertan errores aleatoriamente en los valores de los labels, para simular posibles errores en la Fase 1. De esta manera se puede chequear si el ensembling de Fase 2 da buenos resultados a pesar de que haya fallas en las predicciones de la fase anterior. A este dataset lo referenciaremos en esta sección como *Dataset 1-2-3ConErrores*.

Ambos datasets se dividen para ser usados en las etapas de entrenamiento y testing. El criterio para ello es tomar el 80 % de los datos para entrenamiento y el 20 % de los datos para testing. En este caso la división de los datos se hace en base a la dirección IP Origen de los flujos porque se necesita tener todos los datos relacionados al tráfico de una IP origen en el mismo dataset (ya sea el de entrenamiento o el de testeo). Los flujos pertenecientes a un 80 % de IPs origen del dataset de entrada se usaron para training y los flujos pertenecientes al 20 % de IPs origen restantes del dataset de entrada se usan para testing.

6.2.1. Entrenamiento de la Fase 2

Dado que se tienen etiquetas para cada flujo, y que a partir de los flujos de red se puede distinguir las conexiones de los distintos protocolos de capa de transporte usados para las distintas aplicaciones y servicios, se analizan los distintos tipos de conexiones a fin de obtener datos referentes a comunicación entre dos hosts. A partir de dicho análisis se clasifica el conjunto de flujos entre un origen y un destino, para decidir si el label es malicioso o normal. Para ello se analizan:

- Las conexiones TCP establecidas entre el origen y el destino.
- Las conexiones TCP no establecidas entre el origen y el destino.
- Las conexiones UDP establecidas entre el origen y el destino.

- Las conexiones UDP establecidas entre el origen y el destino.

Para cada grupo de flujos se calcula:

- La cantidad de flujos maliciosos.
- El porcentaje de flujos maliciosos.

Se toma una decisión por cada grupo, teniendo en cuenta el porcentaje de flujos etiquetados como maliciosos y la cantidad de flujos etiquetados como maliciosos, con el criterio definido para esta fase en la Sección 4.2.

Para tomar la decisión es necesario establecer el límite (umbral o threshold), tanto para el porcentaje ($Threshold_{Porcentaje}$) como para la cantidad ($Threshold_{Cantidad}$), de manera tal que si el porcentaje de ese tipo de tráfico y la cantidad de ese tipo de tráfico superan esos valores, ello indica la existencia de tráfico malicioso, y si no lo superan no se considera un indicio de infección.

Dado que entre un origen y un destino puede haber distintas conexiones, algunas relacionadas a ataques o tráfico malicioso, y otras relacionadas a actividad normal, es necesario establecer qué porcentaje sobre el total puede determinar que las conexiones entre un origen y un destino forman parte de una infección. Para ello se define $Threshold_{Porcentaje}$, límite para el porcentaje de flujos maliciosos. El mismo se usa para evaluar si el porcentaje de flujos entre un origen y un destino marcados como maliciosos supera ese límite, y en ese caso clasificar a todo el conjunto como malicioso.

$Threshold_{Cantidad}$, límite para la cantidad de flujos maliciosos, se usa para analizar si la cantidad de flujos entre el origen y el destino etiquetados como malicosos supera dicho límite. Se decide establecer este threshold debido a que el porcentaje por sí solo puede no ser suficiente, como es el caso de que tengamos solo un flujo o muy pocos entre un origen y un destino y los mismos sean maliciosos. Por esta razón es que se decide incluir ambos thresholds en el criterio de decisión de esta fase.

En esta fase, entonces, el entrenamiento consiste en correr el proceso para etiquetar los conjuntos de flujos utilizando distintos valores para los thresholds. Dichos valores determinan el modelo a usar en las predicciones.

Para determinar el mejor resultado la métrica elegida es F1Score. Esta métrica es la recomendada cuando se desea tener un modelo con buena precisión y re-

call(recuperación) [1]. La misma mantiene un equilibrio entre la precisión y el recall de su clasificador.

Se entrenó con los siguientes valores:

- $Threshold_{Porcentaje} = [0; 25; 50; 75]$
- $Threshold_{Cantidad} = [0; 1; 5; 10; 50]$

Los resultados del entrenamiento para el dataset *Dataset 1-2-3* se muestran en la Tabla 6.13. En la misma puede observarse que las combinaciones que dan mejores resultados son:

- $Threshold_{Cantidad} = 0$ y $Threshold_{Porcentaje} = 0$
- $Threshold_{Cantidad} = 0$ y $Threshold_{Porcentaje} = 25$

Para ambas el valor de F1Score es de 98,799 %. Para *Dataset 1-2-3* se elige usar como modelo la combinación $Threshold_{Cantidad} = 0$ y $Threshold_{Porcentaje} = 25$. Estos valores aseguran que cuando el porcentaje de flujos maliciosos sea menor a 25 % no se van a generar detecciones, evitando así sumar falsos positivos en esta etapa del proceso. Este modelo será referenciado como *Modelo 1* de ahora en adelante.

Mientras que para el dataset *Dataset 1-2-3ConErrores* los valores $Threshold_{Cantidad} = 0$ y $Threshold_{Porcentaje} = 75$ son los que dan mejores resultados. Ello puede observarse en la Tabla 6.14. Para dichos thresholds el valor de F1Score es de 98,72 %. Para *Dataset 1-2-3ConErrores* se elige usar como modelo la combinación $Threshold_{Cantidad} = 0$ y $Threshold_{Porcentaje} = 75$. Este modelo será referenciado como *Modelo 2* de ahora en adelante.

6.2.2. Testeo de la Fase 2

Se realiza el testeo de los modelos *Modelo 1* y *Modelo 2* obtenidos para *Dataset 1-2-3* y *Dataset 1-2-3ConErrores*. Para ello se usan las muestras correspondientes a datos de testing para ambos datasets.

Los resultados para el *Modelo 1* son los que se muestran en la matriz de confusión de la Figura 6.3. A partir de la misma se calculan las métricas FalsePositiveRate, TruePositiveRate, F1Score y Accuracy, presentes en la Tabla 6.15.

Threshold Percentege Malicious Flows PerIPDst	**Threshold Counter Malicious Flows PerIPDst**	**FP**	**FN**	**TP**	**TN**	**FPR**	**TPR**	**F1 Score**	**Accuracy**
0 %	**0**	**0**	**16**	**658**	**227**	**0**	**97,626 %**	**98,799 %**	**98,224 %**
25 %	**0**	**0**	**16**	**658**	**227**	**0**	**97,626 %**	**98,799 %**	**98,224 %**
50 %	0	0	17	657	227	0	97,478 %	98,723 %	98,113 %
75 %	0	0	17	657	227	0	97,478 %	98,723 %	98,113 %
0 %	5	0	663	11	227	0	1,632 %	3,212 %	26,415 %
25 %	5	0	663	11	227	0	1,632 %	3,212 %	26,415 %
50 %	5	0	663	11	227	0	1,632 %	3,212 %	26,415 %
75 %	5	0	663	11	227	0	1,632 %	3,212 %	26,415 %
0 %	1	0	620	54	227	0	8,012 %	14,835 %	31,188 %
25 %	1	0	620	54	227	0	8,012 %	14,835 %	31,188 %
50 %	1	0	620	54	227	0	8,012 %	14,835 %	31,188 %
75 %	1	0	620	54	227	0	8,012 %	14,835 %	31,188 %
0 %	10	0	670	4	227	0	0,593 %	1,180 %	25,638 %
25 %	10	0	670	4	227	0	0,593 %	1,180 %	25,638 %
50 %	10	0	670	4	227	0	0,593 %	1,180 %	25,638 %
75 %	10	0	670	4	227	0	0,593 %	1,180 %	25,638 %
0 %	25	0	673	1	227	0	0,148 %	0,296 %	25,305 %
0 %	50	0	673	1	227	0	0,148 %	0,296 %	25,305 %
25 %	25	0	673	1	227	0	0,148 %	0,296 %	25,305 %
25 %	50	0	673	1	227	0	0,148 %	0,296 %	25,305 %
50 %	25	0	673	1	227	0	0,148 %	0,296 %	25,305 %
50 %	50	0	673	1	227	0	0,148 %	0,296 %	25,305 %
75 %	25	0	673	1	227	0	0,148 %	0,296 %	25,305 %
75 %	50	0	673	1	227	0	0,148 %	0,296 %	25,305 %

Tabla 6.13: Resultados del entrenamiento de Fase 2 para el dataset *Dataset 1-2-3*. Los mejores resultados se obtienen para las combinaciones ($Threshold_{Cantidad} = 0$, $Threshold_{Pocentaje} = 0$) y ($Threshold_{Cantidad} = 0$, $Threshold_{Pocentaje} = 25$), cuyo valor de F1Score es 98,799.

Threshold Percentege Malicious Flows PerIPDst	**Threshold Counter Malicious Flows PerIPDst**	**FP**	**FN**	**TP**	**TN**	**FPR**	**TPR**	**F1 Score**	**Accuracy**
75 %	**0**	**0**	**17**	**657**	**227**	**0 %**	**97,478 %**	**98,723 %**	**98,113 %**
50 %	0	1	17	657	226	0,441 %	97,478 %	98,649 %	98,002 %
25 %	0	4	17	657	223	1,762 %	97,478 %	98,427 %	97,669 %
0 %	0	6	17	657	221	2,643 %	97,478 %	98,280 %	97,447 %
75 %	1	0	620	54	227	0 %	8,012 %	14,835 %	31,188 %
50 %	1	1	620	54	226	0,441 %	8,012 %	14,815 %	31,077 %
0 %	1	3	620	54	224	1,322 %	8,012 %	14,774 %	30,855 %
25 %	1	3	620	54	224	1,322 %	8,012 %	14,774 %	30,855 %
50 %	5	0	663	11	227	0 %	1,632 %	3,212 %	26,415 %
75 %	5	0	663	11	227	0 %	1,632 %	3,212 %	26,415 %
0 %	5	1	663	11	226	0,441 %	1,632 %	3,207 %	26,304 %
25 %	5	1	663	11	226	0,441 %	1,632 %	3,207 %	26,304 %
50 %	10	0	670	4	227	0 %	0,593 %	1,180 %	25,638 %
75 %	10	0	670	4	227	0 %	0,593 %	1,180 %	25,638 %
0 %	10	1	670	4	226	0,441 %	0,593 %	1,178 %	25,527 %
25 %	10	1	670	4	226	0,441 %	0,593 %	1,178 %	25,527 %
50 %	25	0	673	1	227	0 %	0,148 %	0,296 %	25,305 %
50 %	50	0	673	1	227	0 %	0,148 %	0,296 %	25,305 %
75 %	25	0	673	1	227	0 %	0,148 %	0,296 %	25,305 %
75 %	50	0	673	1	227	0 %	0,148 %	0,296 %	25,305
0 %	25	1	673	1	226	0,441 %	0,1 %	0,296 %	25,194 %
0 %	50	1	673	1	226	0,441 %	0,148 %	0,296 %	25,194 %
25 %	25	1	673	1	226	0,441 %	0,148 %	,296 %	25,194 %
25 %	50	1	673	1	226	0,441 %	0,148 %	0,296 %	25,194 %

Tabla 6.14: Resultados del entrenamiento de Fase 2, para *Dataset 1-2-3ConErrores*.mejores resultados se obtienen para las combinaciones ($Threshold_{Cantidad} = 0$, $Threshold_{Pocentaje} = 0{,}75$, cuyo valor de F1Score es 98,722 %.

FalsePositiveRate	TruePositiveRate	F1Score	Accuracy
0 %	97,77 %	98,87 %	98,36 %

Tabla 6.15: Métricas obtenidas para el testeo del *Modelo 1* de Fase 2 que usa la combinación $Threshold_{Cantidad} = 0$ y $Threshold_{Porcentaje} = 25$ con la muestra de datos de testeo del dataset *Dataset 1-2-3*.

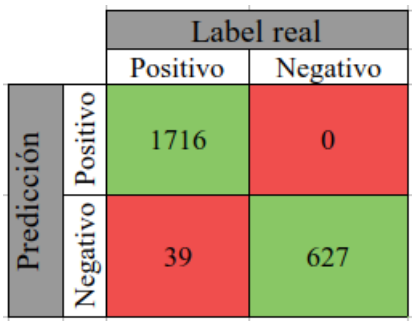

Figura 6.3: Matriz de confusión para el testeo del *Modelo 1* de Fase 2 que usa la combinación $Threshold_{Cantidad} = 0$ y $Threshold_{Porcentaje} = 25$ con la muestra de datos de testeo del dataset *Dataset 1-2-3*

Siendo el valor de F1Score = 98.87 %, mejor que el obtenido para esa métrica en el entrenamiento, donde F1Score = 98.79 %.

Los resultados para *Dataset 1-2-3ConErrores* fueron los que se muestran en la matriz de confusión de la Figura 6.4. A partir de ella se obtienen los siguientes valores para las métricas que se detallan en la Tabla 6.16.

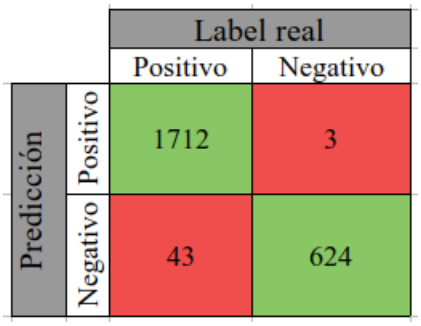

Figura 6.4: Matriz de confusión para el testeo del *Modelo 1* de Fase 2 que usa la combinación $Threshold_{Cantidad} = 0$ y $Threshold_{Porcentaje} = 75$ con la muestra de datos de testeo del dataset *Dataset 1-2-3ConErrores*

Siendo el valor de F1Score = 98,67 %, similar al obtenido para esa métrica en el entrenamiento, donde F1Score = 98,72 %.

6.2.3. Resultados de la Fase 2

Como resultado de esta fase se decide usar la combinación $Threshold_{Cantidad} = 0$ y $Threshold_{Porcentaje} = 25$. Estos valores aseguran que cuando el porcentaje de

FalsePositiveRate	TruePositiveRate	F1Score	Accuracy
0,47	97,54	98,67	98,06

Tabla 6.16: Métricas obtenidas para el testeo del Modelo 1 de Fase 2 que usa la combinación $Threshold_{Cantidad} = 0$ y $Threshold_{Porcentaje} = 25$ con la muestra de datos de testeo del dataset *Dataset 1-2-3ConErrores*

flujos maliciosos sea menor a 25 % no se van a generar detecciones, evitando así sumar falsos positivos en esta etapa del proceso.

Los experimentos realizados a partir del dataset con errores random, tienen como fin analizar si la clasificación de Fase 2 mejora los posibles errores que puede tener el proceso de clasificación de la Fase 1. En el training con este dataset se determina el uso de la combinación de valores $Threshold_{Cantidad} = 0$ y $Threshold_{Porcentaje} = 75$. Y en el testing se obtiene un valor de 98,87 % para F1Score. Esto muestra que, eligiendo un modelo adecuado, los falsos positivos y falsos negativos provenientes de la Fase 1, se reducen aplicando los criterios de decisión de esta fase.

Los experimentos de la Fase 2 y sus resultados fueron publicados el libro Computer Science – CACIC 2019 [21].

Como resultado del procesamiento de esta fase, se genera el *Dataset Resultante de la Fase 2* con las siguientes columnas:

- ClaveSrcIPDstIP: campo clave, identifica al conjunto de flujos que van del origen SrcAddr al destino DstAddr. Los campos que se describen a continuación se refieren a ese conjunto de flujos.
- SrcAddr: dirección IP origen.
- DstAddr: dirección IP destino.
- TCPEstablishedPercentageMW: Porcentaje de flujos TCP establecidos etiquetados como malicious.
- TCPNotEstablishedPercentageMW: porcentaje de flujos TCP no establecidos etiquetados como malicious.
- UDPEstablishedPercentageMW: porcentaje de flujos UDP establecidos etiquetados como malicious.

- UDPNotEstablishedPercentageMW: porcentaje de flujos UDP no establecidos etiquetados como malicious.
- cantTCPEMW: cantidad de flujos TCP establecidos etiquetados como malicious.
- cantTCPNEMW: cantidad de flujos TCP no establecidos etiquetados como malicious.
- cantUDPEMW: cantidad de flujos UDP establecidos etiquetados como malicious.
- cantUDPNEMW: cantidad de flujos UDP no establecidos etiquetados como malicious.
- cantTCPE: cantidad total de flujos TCP establecidos.
- cantTCPNE: cantidad total de flujos TCP no establecidos.
- cantUDPE: cantidad total de flujos UDP establecidos.
- cantUDPNE: cantidad total de flujos UDP no establecidos.
- totalFlows: cantidad total de flujos que se transmiten entre el origen y el destino.
- totalPackets: cantidad total de paquetes que se transmiten entre el origen y el destino.
- totalBytes: cantidad total de bytes que se transmiten entre el origen y el destino.
- TCPELabel: label asignado al grupo de flujos TCP establecidos.
- TCPNELabel: label asignado al grupo de flujos TCP no establecidos.
- UDPELabel: label asignado al grupo de flujos UDP establecidos.
- UDPNELabel: label asignado al grupo de flujos UDP no establecidos.

- PredictLabel: label para el conjunto de todos los flujos que van de un origen a un destino (malicioso o normal).
- GroundTruth: valor de verdad (se asigna de acuerdo a lo que se conoce del host origen).

El formato del dataset resultant puede verse en la Figura 6.5, donde se incluyen algunas filas del mismo a modo de ejemplo.

SrcIPDstIP	SrcAddr	DstAddr	TCPEstablished PercentageMW	TCPNotEstablished PercentageMW	UDPEstablished PercentageMW	UDPNotEstablished PercentageMW
147.32.84.204-147.32.96.69	147.32.84.204	147.32.96.69	0	0	0	0
147.32.84.205-147.32.96.69	147.32.84.205	147.32.96.69	0	0	0	100
147.32.80.9-147.32.84.59	147.32.80.9	147.32.84.59	0	0	0	0
147.32.84.170-199.7.59.72	147.32.84.170	199.7.59.72	0	0	0	0

cantTCPE MW	cantTCPNE MW	cantUDPE MW	cantUDPNE MW	CantTCPE	CantTCPNE	CantUDPE	CantUDPNE
0	0	0	0	0	0	0	0
0	0	0	3	0	0	0	3
0	0	0	0	0	0	0	160
0	0	0	0	1	0	0	0

totalFlows	totalPackets	totalBytes	TCPE Label	TCPNE Label	UDPE Label	UDPNE Label	Predict Label	Ground Truth
360	8.536	9.099.376	normal	normal	normal	normal	normal	malicious
11.701	81.972	87.379.134	normal	normal	normal	malicious	malicious	malicious
160	160	32.948	normal	normal	normal	normal	normal	normal
1	12	2.614	normal	normal	normal	normal	normal	normal

Figura 6.5: Dataset resultante de la Fase 2 (ejemplo que muestra 4 filas) con los campos: SrcAddr, DstAddr, TCPEstablishedPercentegeMW, TCPNotEstablishedPercentegeMW, UDPEstablishedPercentegeMW, UDPNotEstablishedPercentegeMW, cantTCPEMW, cantTCPNEMW, cantUDPEMW, cantUDPNEMW, totalPackets, totalBytes, TCPELabel, TCPNELabel, UDPELabel, UDPNELabel, PredictLabel y GroundTruth. PredictLabel es la predicción de la Fase 2 para el conjunto de flujos entre el origen y el destino y GroundTruth representa el label real para la dirección IP origen.

6.3. Experimentos Fase 3

La Fase 3 es la encargada de tomar la decisión final respecto a la IP origen, etiquetando a la misma como infectada o normal. Para ello cuenta con los resultados de la Fase 2, que clasifica cada conjunto de flujos desde ese origen a sus distintas IPs destinos. Puede decirse entonces que tenemos un label por IP destino.

En la Fase 3 el proceso se alimenta de la información provista por diferentes fuentes de Threat Intelligence. En los experimentos realizados la información de TI usada es la provista por el módulo VirusTotal de Slips.

El criterio de ensembling usado fue el detallado en Sección 4.3. A continuación se describen los experimentos que se llevaron a cabo en las etapas de entrenamiento y testeo de esta fase, así como los resultados obtenidos al aplicar ensembling incluyendo la información de TI para los destinos.

6.3.1. Entrenamiento de la Fase 3

Para los experimentos de esta etapa se usó el dataset resultante del entrenamiento de Fase 2. Para cada destino se consulta la información de VirusTotal.

A partir de ello se genera el *Dataset Intermedio de la Fase 3* que incluye los campos del dataset resultante de la Fase 2, ya descriptos. Y se agregan campos con la información de Threat Intelligence para cada destino. Dicha información se usa para el ensembling de esta fase y es la siguiente:

- vtInfoUrl: valor de *url_ratio* provisto por el módulo VirusTotal para el destino.
- vtInfoDownload: valor de *download_ratio* provisto por el módulo VirusTotal para el destino.
- vtInfoReferrer: valor de *referrer_ratio* provisto por el módulo VirusTotal para el destino.
- vtInfoCommunic: valor de *communicating_ratio* provisto por el módulo VirusTotal para el destino.

La información de TI no sirve para detectar hosts infectados, sino para analizar qué ocurre con los destinos a los que se conectan los hosts. Que un destino al que un host se conecta presente indicios de maliciocidad, no implica necesariamente que el host en cuestión esté infectado.

Como se analiza el tráfico saliente, los datos de VirusTotal, que es la fuente de TI usada, tienen menos valor o peso que si se tratara de tráfico entrante. En esta fase se toma una decisión para cada IP origen teniendo en cuenta la información de TI

para todos los destinos con que un host se comunica. Para esto se calcula un único valor en base a la información que VirusTotal provee para todos ellos, incluyendo los normales. De esta manera podremos distinguir entre un host que tiene mucho tráfico normal de otro que tiene mucho tráfico normal y también está infectado.

Por otro lado, puede ocurrir que un host infectado se conecte a destinos normales para llevar a cabo alguna acción maliciosa, como puede ser un escaneo. En esos casos la información de TI no aporta nuevas evidencias a la información provista por la Fase 2.

Como se describió en la Sección 4.3, se define la variable *VTConfidence*, para incluir la información provista por el módulo VirusTotal. Y su cálculo, también visto en dicha sección se realiza de la siguiente forma:

$$\begin{aligned} VTConfidence = w1 * VTSumUrl + w2 * VTSumDownload \\ + w3 * VTSumCommunic + w4 * VTSumReferrer \end{aligned} \tag{6.1}$$

Donde w1, w2, w3, w4 son los pesos asignados para la información provista por VT (w1 es el peso para la suma de los valores de url_ratio de todos los destinos, w2 es el peso para la suma de los valores de download_ratio de todos los destinos, w3 es el peso para la suma de los valores de communicating_ratio de todos los destinos y w4 es el peso para la suma de los valores de referrer_ratio de todos los destinos. Los valores de dichos pesos se determinan en esta etapa de entrenamiento.

A fin de incluir los resultados de la Fase 2 en el ensembling de esta fase, se ha definido P2Confidence, descripta también en la Sección 4.3. P2Confidence se calcula en base a la cantidad de conjuntos de flujos etiquetados como maliciosos en la Fase 2, de acuerdo al siguiente criterio:

- Si $Cantidad_{GruposMaliciosos} \geq threshold3 \rightarrow P2Confidence = 0{,}59$
- Si $threshold2 \leq Cantidad_{GruposMaliciosos} \leq threshold3 \rightarrow P2Confidence = 0{,}55$
- Si $threshold1 \leq Cantidad_{GruposMaliciosos} \leq threshold2 \rightarrow P2Confidence = 0{,}50$

- Si $Cantidad_{GruposMaliciosos} \leq threshold1 \rightarrow P2Confidence = 0$

Donde los valores a usar para threshold1, threshold2 y threshold 3 son los umbrales usados para para asignar un valor a P2Confidence basado en la cantidad conjuntos de flujos (asociados a un destino) cuyo label es malicioso. Dichos valores se determinan en esta etapa de entrenamiento.

También se ha definido la variable Phase3Confidence para realizar el ensembling, cuyo valor como se ha descripto en la Sección 4.3 se calcula de esta forma:

$$Phase3Confidence = P2Confidence + VTConfidence \tag{6.2}$$

La decisión final para clasificar un host se hace asignando un valor al label de la Fase 3 en función del valor de Phase3Confidence. Si el mismo es mayor a un peso representado por la variable p3weight, se clasificará al host como infectado (malicious). Mientras que si el valor de Phase3Confidence es menor o igual a dicho peso, se clasificará al host como no infectado (normal). El valor de p3weight también se determina en esta estapa de entrenamiento.

En el training de esta fase se determinan entonces los valores de w1, w2, w3, w4, threshold1, threshold2, threshold3 y p3weight, cuya combinación representa el mejor modelo. Al haber varios thresholds y cada uno de ellos puede tomar muchos valores posibles, es necesario definir el mecanismo a utilizar para determinar qué valores de los thresholds optimizan el objetivo. Existen distintas técnicas para ello. Si bien su estudio y análisis detallado se encuentran fuera del alcance de esta tesis, se tienen en cuenta las siguientes:

- Búsqueda de cuadrícula: prueba todos los valores para encontrar el mejor. De esta manera garantiza encontrar el mínimo global.

- Búsqueda heurística: prueba un conjunto valores basados en algún criterio y encuentra el mejor. No garantiza encontrar el mínimo global, solo un mínimo local.

- Gradient descendent: permite localizar un mínimo en una función de forma iterativa.

SrcAddr	DstAddr	vtInfoUrl	vtInfoDownload	vtInfoReferrer	vtInfoCommunic	PredictLabel	GroundTruth
147.32.84.165	222.56.50.66	0	0	0	0	normal	malicious
147.32.84.165	115.206.29.133	0	0	0	0	normal	malicious
147.32.84.165	121.230.189.119	0	0	0	0	normal	malicious
147.32.84.165	175.141.153.32	0	0	0	0	normal	malicious
147.32.84.165	183.7.81.213	0	0	0	0	normal	malicious
147.32.84.165	118.101.146.169	0	0	0	0	normal	malicious
147.32.84.165	217.43.57.174	0	0	0	0	normal	malicious
147.32.84.165	210.35.99.254	0	0	0	0	normal	malicious
147.32.84.165	221.91.224.201	0	0	0	0	normal	malicious
147.32.84.165	58.247.135.20	0	0	0	0	normal	malicious
147.32.84.165	70.160.106.191	0	0	0	0	normal	malicious
147.32.84.165	92.35.196.184	0	0	0	0	normal	malicious
147.32.84.165	222.72.138.150	0	0	0	0	normal	malicious
147.32.84.165	219.142.132.196	0	0	0	0	normal	malicious
147.32.84.165	110.16.155.247	0	0	0	0	normal	malicious
147.32.84.165	121.113.200.247	0	0	0	0	normal	malicious

Tabla 6.17: En esta tabla se muestran los falsos negativos de la Fase 2. Todas ellas tienen como origen a la IP 147.32.84.165. Para cada fila se pueden ver las direcciones IP destino correspondientes a esas conexiones y los valores que se obtienen del Módulo VirusTotal para cada una de ellas.

El método usado en esta fase para entrenar los thresholds se basa en la búsqueda heurística, donde:

$$0 \leq w_i \leq 1 \tag{6.3}$$

$$w1 + w2 + w3 + w4 = 1 \tag{6.4}$$

En cada fase se busca minimizar los falsos positivos y falsos negativos en la clasificación. En particular como resultado de los experimentos de la Fase 2 existen 16 falsos negativos y ningún falso positivo. Los falsos negativos están asociados a una única IP origen, la 147.32.84.165, como puede observarse en la Tabla 6.17 Dicha IP se comunica con 653 direcciones IP destino distintas. De acuerdo a la clasificación realizada en la Fase 2, de esas conexiones 637 son maliciosas, es decir los flujos forman parte de una infección, y 16 son normales.

Para los 16 destinos asociados a los falsos negativo, se analiza la información brindada por el módulo VirusTotal. Para todos ellos los valores de url_ratio, download_ratio, communicating_ratio y referrer_ratio son 0, por lo tanto se debe tener en cuenta que incluir dicha información no reduce los falsos negativos. Para la IP 147.32.84.165, son 637 los destinos clasificados como maliciosos en la Fase 2. En este

IPSrc	VTSumUrl	VTSumDownload	VTSumReferrer	VTSumCommunic	GroundTruth
147.32.84.170	0,971103075941896	0,027835165895313246	3,4981125973305587	24,947710605590185	normal
147.32.84.165	0,8688088629949348	0,8938986171212779	0,978195803502977	17,498817823955388	malicious
147.32.84.206	0,04524886877828054	0	0	0,7922817159189407	malicious
147.32.84.209	0,02564102564102564	0	0,1598173515981735	0,3303027462742525	malicious
147.32.84.193	0,019230769230769232	0	0	0,6934673366834171	malicious
66.249.72.228	0,018867924528301886	0	0	0	Unknow
88.103.19.195	0,018867924528301886	0	0	0	Unknow
212.71.150.246	0,018867924528301886	0	0	0	Unknow
66.249.72.27	0,018867924528301886	0	0	0	Unknow
72.14.199.57	0,018867924528301886	0	0	0	Unknow

Tabla 6.18: En esta tabla se muestran las 10 direcciones IP origen del dataset intermedio de la Fase 3 con la suma de los valores de url_ratio, download_ratio, communicating_ratio y referrer_ratio, para todos los destinos de cada una de ellas. Los mismos están representados por las columnas: VTSumUrl, VTSumDownload, VTSumCommunicating y VTSumReferrer. Estas filas son aquellas que tienen los valores más altos para dichas columnas.

caso los falsos negativos se eliminan al incluir P2Confidence en el criterio de esta fase, ya que su cálculo tiene en cuenta la cantidad de destinos cuyas conexiones forman parte de una infección, es decir aquellos clasificados como maliciosos en la Fase 2.

A fin de determinar los pesos aproximados que colaboren en la decisión, evitando sumar falsos positivos al incluir la información de VirusTotal en el proceso de ensembling de esta fase, se analiza la información del dataset intermedio de Fase 3. En este se tienen para todas las IPs origen la suma de los valores de url_ratio, download_ratio, communicating_ratio y referrer_ratio, para todos los destinos. De dicho análisis surge que se deben tener en cuenta los valores para la IP 147.32.84.170, ya que tiene los máximos valores del conjunto para 3 de ellos (la suma de los valores de url_ratio, communicating_ratio y referrer_ratio para todos los destinos), como puede observarse en la Tabla 6.18. Dado que el GroundTruth de dicha IP es normal, se deben considerar estos valores tanto para la asignación de pesos y la determinación de los thresholds en en el cálculo de Phase3Confidence, de manera tal que la misma sea clasificada como normal para evitar sumar falsos positivos. Se prueba entonces con las combinaciones de valores de pesos que se muestran en la Tabla 6.19 para entrenar el modelo.

La asignación de pesos se hace también en función de lo que representa cada uno de los valores de VirusTotal. Se tiene en cuenta que se están clasificando direcciones IP origen y se está usando en el ensembling información de VirusTotal de los destinos

w1	**w2**	**w3**	**w4**
0,19	0,8	0,01	0
0,18	0,8	0,02	0
0,17	0,8	0,03	0
0,16	0,8	0,04	0
0,15	0,8	0,05	0
0,29	0,7	0,01	0
0,28	0,7	0,02	0
0.27	0.7	0,03	0
0,26	0,7	0,04	0
0,25	0,7	0,05	0
0,39	0,6	0,01	0
0,38	0,6	0,02	0
0,37	0,6	0,03	0
0,36	0,6	0,04	0
0,35	0,6	0,05	0

Tabla 6.19: En esta tabla se muestran las combinaciones usadas para los pesos de w1, w2, w3 y w4 para entrenar el modelo.

con que las mismas se comunican. Se da un mayor peso a la suma de los valores de url_ratio, porque se considera importante tener en cuenta el grado de maliciocidad de las urls con las que se pudo haber conectado la dirección IP origen. También se da un peso más significativo a la suma de los valores de download_ratio, ya que está asociado a la posibilidad de que el origen haya descargado archivos maliciosos de las IPs destino.

Se decide incluir con menor grado de importancia la suma de los valores de communicating_ratio para considerar el hecho de que haya archivos maliciosos que se comunican con alguno de los destinos. Esto puede terminar infectando los mismos para luego afectar a las comunicaciones que éstos tengan con otros hosts, entre los cuales se encuentra la IP origen que se está clasificando. Se da peso 0 a la suma de los valores de referrer_ratio dado que este valor tiene que ver solo con la aparición de alguna de las IPs destino en archivos maliciosos, lo cual no da evidencia certera de que dichas IPs destino sean maliciosas.

Para los thresholds de las cantidades se prueban los siguientes valores:

- threshold1 = (1; 2; 3; 4)
- threshold2 = (5; 10)
- threshold3 = (20; 50; 100)

Para p3weight se usan los siguientes valores en el entrenamiento: 0,50; 0,55; 0,60; 0,65; 0,70; 0,75 y 0,80.

Las combinaciones de valores que dan los mejores resultados se muestran en Tabla 6.20. Entre ellos se elige la combinación de valores que se muestra en la Tabla 6.21 de manera tal que:

- se da peso a la predicción de Fase 2, la cual hemos mostrado que tiene un alto grado de precisión, para determinar su influencia en la decisión final.
- se fija el valor de p3weight en un valor 0,55, en este caso, para:
  - Requerir que haya al menos 5 destinos marcados como maliciosos en la Fase 2.
  - Tener en cuenta la información de TI (VirusTotal) en caso de que dicho feed TI provea indicios de maliciocidad para los destinos con los que la IP origen se comunicó.

6.3.2. Testeo de la Fase 3

Para los experimentos de esta etapa se usa el dataset resultante del testeo de Fase 2. Para cada destino se consultó la información de TI al módulo VirusTotal.

El testing se realiza usando el mejor modelo, es decir fijando los valores de las variables a la combinación resultante de la etapa de training. Dichos valores son los siguientes:

- w1 = 0,19, w2 = 0,8, w3 = 0,1, w4 = 0
- threshold1 = 1, threshold2 = 5, threshold3 = 20
- p3weight = 0,55

w1	w2	w3	w4	Threshold1	Threshold2	Threshold3	p3weight	FP	FN	TP	TN	FPR	TPR	F1Score	Accuracy
0.19	0.8	0.01	0	1	5	20	0.5	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	1	5	20	0.55	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	1	5	50	0.5	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	1	5	50	0.55	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	1	5	100	0.5	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	1	5	100	0.55	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	1	10	20	0.5	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	1	10	50	0.5	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	1	10	100	0.5	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	2	5	20	0.5	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	2	5	20	0.55	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	2	5	50	0.5	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	2	5	50	0.55	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	2	5	100	0.5	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	2	5	100	0.55	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	2	10	20	0.5	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	2	10	50	0.5	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	2	10	100	0.5	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	3	5	20	0.5	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	3	5	20	0.55	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	3	5	50	0.5	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	3	5	50	0.55	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	3	5	100	0.5	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	3	5	100	0.55	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	3	10	20	0.5	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	3	10	50	0.5	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	3	10	100	0.5	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	4	5	20	0.5	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	4	5	20	0.55	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	4	5	50	0.5	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	4	5	50	0.55	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	4	5	100	0.5	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	4	5	100	0.55	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	4	10	20	0.5	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	4	10	50	0.5	0	0	4	3	0 %	100 %	100 %	100 %
0.19	0.8	0.01	0	4	10	100	0.5	0	0	4	3	0 %	100 %	100 %	100 %

Tabla 6.20: Conjunto de combinaciones de w1, w2, w3, w4, threshold1, threshold2, threshold3 y p3weight que mejor predicen, en base a las métricas evaluadas. Puede observarse que para los valores (w1, w2, w3, w4) = (0,19; 0,80; 0,01; 0), y para todas las combinaciones probadas para Threshold1(1; 2; 3; 4), Threshold2(5; 10), Threshold3(20; 50; 100) y p3weight(0,50; 0,55).

w1	**w2**	**w3**	**w4**	**threshold1**	**threshold2**	**threshold3**	**p3weight**
0,19	0,8	0,1	0	1	5	20	0,55

Tabla 6.21: Valores elegidos para cada una de las variables (w1, w2, w3, w4, threshold1, threshold2, threshold3 y p3weight) en función de los resultados del entrenamiento.

w1	w2	w3	w4	Threshold1	Threshold2	Threshold3	p3weight	FP	FN	TP	TN	FPR	TPR	F1Score	Accuracy
0,19	0,8	0,01	0	1	5	20	0,55	1	0	8	4	20 %	100 %	94,11764 %	92,30769 %

Tabla 6.22: Valores de las métricas resultantes del testeo de la Fase 3, donde se puede observar que en este proceso se suma un falso positivo con los valores determinados por el entrenamiento.

IPSrc	P2Confidence	VTConfidence	Phase3Confidence	Phase3Label	GroundTruth
147.32.84.207	0.55	0,03258418116614783	0,5825841811661479	malicious	malicious
147.32.84.192	0.59	1,1873946355975045	1,7773946355975045	malicious	malicious
147.32.84.204	0.55	0,008257055522507654	0,5582570555225077	malicious	malicious
147.32.84.205	0.55	0,013397900592930188	0,5633979005929303	malicious	malicious
147.32.84.165	0.59	1,0675007454101344	1,6575007454101343	malicious	malicious
147.32.84.208	0.55	0,005135135135135135	0,5551351351351351	malicious	malicious
147.32.84.191	0.59	1,3523028620642592	1,9423028620642593	malicious	malicious
147.32.84.134	0	0,7436957316113059	0,7436957316113059	malicious	normal
192.168.1.121	0.55	0,07791129102495792	0,627911291024958	malicious	malicious

Tabla 6.23: Direcciones IP clasificadas como maliciosas (es decir que están infectadas) durante el testeo de la Fase 3. Para cada una se muestra su valor de P2Confidence y VTConfidence, usados para calcular el P3Confidence y determinar el label de esta fase. Se puede observar que la dirección IP 147.32.84.134 es clasificada como maliciosa y se trata de un falso positivo.

Como resultado del testeo se obtienen para las métricas los valores que se muestran en la tabla 6.22. En este caso no son valores óptimos como los del entrenamiento ya que se produce un falso positivo, el cual corresponde a la IP 147.32.84.134 como puede verse en la tabla 6.23, donde se muestran las IPs clasificadas como maliciosas (es decir que están infectadas) como resultado del testeo de la Fase 3.

6.3.3. Resultados de la Fase 3

Como resultado del procesamiento de esta fase, se genera el *Dataset Resultante de la Fase 3* que incluye los siguientes campos:

- IPSrc: IP origen.
- P2Confidence: grado de confianza que se tiene en que la IP está infectada, en base a la información brindada por la Fase 2.
- VTSumUrl: suma de los valores de *url* para todos los destinos.

- VTSumDownload: suma de los valores de *download* para todos los destinos.
- VTSumReferrer: suma de los valores de *referrer* para todos los destinos.
- VTSumCommunic: suma de los valores de *communicating* para todos los destinos.
- totalFlows: cantidad total de flujos intercambiados que tienen como origen el host IPSrc.
- totalMaliciousFlows: cantidad de flujos etiquetados como malicious, que tienen como origen IPSrc.
- VTConfidence: grado de confianza que se tiene en que la IP está infectada, en base a la información brindada por el módulo VirustTotal para sus destinos.
- Phase3Confidence: grado de confianza que se tiene en que la IP está infectada, en base a los resultados del ensembling de la Fase 3.
- infectedDestinations: cantidad de destinos marcados como infectados que se comunican con la IP.
- totalDestinations: cantidad total de destinos que se comunican con la IP.
- Phase3Label: label resultante de la Fase 3, que indica si la IP está infectada o no.
- GroundTruth: valor de verdad asignado, de acuerdo a lo que se conoce del host origen.

El formato del dataset resultante puede verse en la Figura 6.6, donde se incluyen algunas filas del mismo a modo de ejemplo.

De acuerdo a los resultados del training se define que la mejor combinación para los valores de los pesos y thresholds:

- w1 = 0,19; w2 = 0,8; w3 = 0,01; w4 = 0
- Threshold1 = 1, Threshold2 = 5; Threshold3 = 20

IPSrc	Phase2Confidence	VTSumUrl	VTSumDownload	VTSumReferrer
147.32.84.191	0,59	0,92704600	0,87321847	0,45098235
147.32.84.165	0,59	0,87030001	0,89739700	0,97695338
147.32.84.192	0,59	0,93323767	1,08023097	0,67068872
147.32.84.170	0,00	25,93677883	0,02779267	3,49057496

VTSumCommunic	totalFlows	totalMalwareFlows	VTConfidence	Phase3Confidence
892,11785418	11049	776	9,79589206	10,38589206
18,42261413	1203	824	1,06750075	1,65750075
14,58947005	10817	585	1,18739464	1,77739464
26,83559595	14002	0	5,21857808	5,21857808

Phase 3 Label	GroundTruth	MaliciousGroups	TotalGroups
malicious	malicious	613	619
malicious	malicious	637	653
malicious	malicious	423	427
malicious	normal	0	206

Figura 6.6: Dataset resultante de la Fase 3 (ejemplo que muestra 4 filas) con los campos: IPSrc, P2Confidence, VTSumUrl, VTSumDownload, VTSumReferrer, VTSumCommunic, totalFlows, totalMaliciousFlows, VTConfidence, Phase3Confidence, Phase3Label, GroundTruth, MaliciousGroup y TotalGroup. Phase3Label es la predicción de la Fase 3 y GroundTruth representa el label real para la dirección IP.

- p3weight = 0,55

Con estos valores en el training se tiene que el FPR (False Positive Rate) = 0 %, el TPR (True Positive Rate) = 100 %, el F1Score = 100 % y el Accuracy = 100 %; siendo estos valores óptimos. Los resultados son similares para el testeo, donde las métricas tienen los siguientes valores: FPR (False Positive Rate) = 20 %, el TPR (True Positive Rate = 100 %, el F1Score = 94,11764 % y el Accuracy = 92,30769 %. Si bien no son las óptimas como en el training, prueban que la combinación elegida de pesos y thresholds da buenos resultados.

Los experimentos muestran que se cumple el objetivo de contar con una metodología de clasificación de direcciones IP origen como maliciosas o normales, indicando si están infectadas o no, minimizando los falsos positivos y falsos negativos. El ensembling de la Fase 1, que combina los resultados de los clasificadores usando la técnica de Voting con peso, cuenta con una alta precisión (99,8 %) para clasificar los flujos de red.

En la Fase 2, eligiendo adecuadamente los thresholds para cantidades y porcen-

tajes usados para clasificar todas las conexiones dadas entre una dirección IP origen y una dirección IP destino, se reducen los falsos positivos y falsos negativos que provengan de la Fase 1. Esto se demuestra a través de los experimentos donde se insertan aleatoriamente errores en los resultados de la fase 1 y se tienen los siguientes valores para las métricas: FPR (FalsePositiveRate) = 0,47 %, TPR (TruePositiveRate) = 97,54 %, F1Score = 98,67 % y Accuracy = 98,06 %.

En la Fase 3, las combinaciones de pesos y thresholds se definen de manera tal de dar relevancia a la clasificación de la Fase 2 respecto a los destinos. Los resultados muestran que la información de TI, en particular provista por el módulo de VirusTotal sirve para reforzar la decisión de la Fase 2, en el ensembling que se hace considerando todos los destinos para una dirección IP origen dada. Esto se tiene en cuenta al determinar los valores de los pesos y thresholds relacionados a la información de VirusTotal, usados en el criterio de clasificación. Se les da un valor tal que se requiera que haya evidencia suficiente para marcar una dirección IP origen como infectada en caso de que la fase 2 clasifique como maliciosos solo 1 o ninngún destino con que esa dirección IP origen se comunica.

Los scripts implementados para realizar los experimentos se encuentran disponibles para la comunidad [33]. Dicho código constituye una base para el desarrollo de herramientas que apliquen ensembling en problemas de ciberseguridad.

Capítulo 7

Implementación del Módulo Ensembling

A partir del trabajo de investigación realizado, se desarrolla el módulo de Ensembling. El mismo detecta hosts infectados en la red, a través de la implementación de las 3 fases de ensembling definidas en la presente tesis.

La arquitectura del módulo se muestra en la Figura 7.1. El mismo se integra a Slips, combinando resultado de sus clasificadores, información de los flujos almacenada por Slips e información de TI provista por su módulo VirusTotal.

El módulo se activa cada vez que se termina un TimeWindows. Cuando esto ocurre recibe la llamada de un canal. Al activarse el mismo obtiene el conjunto de flujos de ese TimeWindows. Y entonces realiza el procesamiento necesario para decidir el label para cada IP origen. Para ello implementa el siguiente comportamiento:

Arregloflujos = Obtener los flujos del Timewidows TW_i

Ensembling en Fase 1

PARA TODOS *los flujos del Timewindows* TW_i

- Se obtienen todas las predicciones del flujo f_i.
- Se aplica ensembling a las predicciones para obtener la decisión final para el flujo f_i.

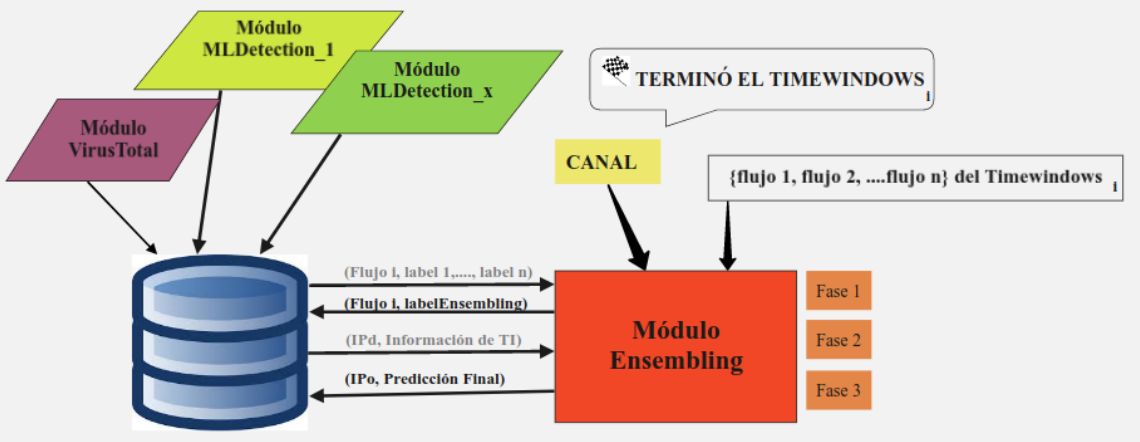

Figura 7.1: Arquitectura del Módulo Ensembling. Integración de dicho módulo a Slips e interacción con los distintos componentes del sitema. El mismo se activa cada vez que finaliza un Timewindow. Al activarse realiza el procesamiento del conjunto de flujos pertenecientes a ese Timewindows para clasificar los hosts como infectados o no infectados.

- Se guarda la predicción final para f_i en el arreglo.
- Se guarda la predicción para f_i en la BD.

Ensembling en Fase 2 y Fase 3

PARA CADA IP origen SrcAddr del conjunto de flujos

- **PARA CADA** *destino DstAddr con que se comunica SrcAddr*
  - Se calculan porcentajes y cantidades de flujos maliciosos para cada grupo de flujos (TCP establecidas, TCP no establecidas, UDP establecidas y UDP no establecidas).
  - Si algún grupo supera el threshold de cantidad y porcentaje, se etiqueta como malicioso, sino como normal.
  - Se guarda Phase2PredictLabel en el arreglo (***Ensembling en Fase2***).
  - Se obtiene la información de VirusTotal para DstAddr.
  - Se suma el valor URLl de VT para ese destino a VTSumUrl, que guarda la suma de los valores de URL para todos los destinos.

- Se suma el valor download de VT para ese destino a VTSumDownlad, que guarda la suma de los valores de download para todos los destinos.
- Se suma el valor communicating de VT para ese destino a VTSumCommunicating, que guarda la suma de los valores de communicating para todos los destinos.
- Se suma el valor referrer de VT para ese destino a VTSumReferrer, que guarda la suma de los valores de referrer para todos los destinos.

- Se calcula VTConfidence (grado de confianza que aporta VirusTotal para etiquetar la IP origen como maliciosa) a partir de la la información que el feed brinda de los destinos.
- Se establece el valor de P2Confidence (grado de confianza que aporta la Fase 2 para etiquetar la IP origen como maliciosa) en función de la cantidad de conjuntos de flujos marcados como maliciosos.
- Se calcula Phase3Confidence (grado de confianza calculado en la Fase 3 para etiquetar la IP origen como maliciosa), en función de Phase2Confidence y VTConfidence.
- Si (Phase3Confidence es mayor que el umbral establecido (p3weight), entonces la IP origen se marca como infectada (Phase3Label=Infected), sino se considera no infectada (Phase3=normal).
- Se guarda la predicción final para la IP origen en la BD.

7.0.1. Propuesta de integración del módulo a CERTUNLP

En base a los objetivos de la tesis, se han analizado los procesos actuales de CERTUNLP, CSIRT Académico de la Universidad Nacional de La Plata, en relación a la detección de hosts infectados. Actualmente, CERTUNLP recibe información de distintos feeds externos, como Shadow Server [26] y Spamhaus [42], que indican cuando un host de la red de la UNLP (su comunidad objetivo) forma parte de una botnet. También se usa Zeek [45], como feed interno, el cual se ha configurado para

detectar algunos patrones conocidos en relación a hosts infectados, como por ejemplo hosts que están minando criptomonedas o hosts que realizan escaneando.

Actualmente no se correlaciona la información provista por los diferentes feeds. Dichos feeds se encuentran integrados a Ngen, sistema de gestión de incidentes de CERTUNLP [8]. A partir de la información provista por los mismos se generan automáticamente incidentes, que son tratados como tal. Los feeds de TI no se usan para mejorar los datos que se tienen para decidir si un host está infectado o no, dado que no se entrecruza información, sino que se toma la brindada por un feed externo como suficiente para tratar el incidente.

El módulo de Ensembling desarrollado se puede integrar como un feed interno de CERTUNLP. De este modo se tendría en cuenta también la información de VirusTotal para las decisiones, feed que no es utilizado por el CSIRT. También es posible plantear como proyecto a futuro sumar los feeds hoy usados independientemente, como fuentes de entrada para la Fase 3 de nuestro proceso de ensembling. De esta manera se enriquecería el ensembling implementado. Y además, a través del ensembling se combinaría la información provista por los diferentes feeds, mejorando así los resultados del proceso actual para detectar hosts infectados en la red de la UNLP.

Capítulo 8

Conclusiones

Al finalizar la tesis se ha cumplido con los objetivos planteados en su inicio, a partir de la creación de una metodología para detectar hosts infectados en la red, que aplica Ensemble Learning en las distintas fases del proceso. Y su implementación en un módulo integrado a Slips que aporta a la herramienta los beneficios del *Ensembling.*

La metodogía definida en este trabajo fue validada a través de experimentos usando datasets con datos de tráfico real, provistos por Stratosphere [49] y disponibles para la comunidad. Ello permite ajustar el proceso teniendo en cuenta flujos que representan el comportamiento real de un conjunto de hosts y aprender a partir de ello.

Esta propuesta de implementación del ensembling en fases permite resolver en cada una de ellas un aspecto del problema, y a su vez tomar las predicciones de la fase anterior, que se combinan con el análisis propio de la fase para lograr mejores resultados.

El entrenamiento que se realiza en cada fase para elegir el mejor modelo, permite que el modelo se ajuste para detectar nuevos ataques. Esto lo pone en ventaja frente a las herramientas basadas en firmas o reglas estáticas.

Dicho mecanismo resulta novedoso al combinar: la información que proveen los datasets generados a partir del tráfico de máquinas infectadas y máquinas no infectadas (Fase 1), la información que provee el análisis de las conexiones TCP establecidas, TCP no establecidas, UDP establecidas y UDP no establecidas con cada destino con

las que un host se comunica (Fase 2), y la información que las distintas fuentes de TI proveen en relación a los destinos con los que un host se conecta (Fase 3).

Esta combinación que se realiza para tomar la decisión final de si un host está infectado o no, es aplicar ensembling en sí mismo. Se decide teniendo en cuenta los resultados de las 3 fases.

Existen ventajas propias de cada fase. En la Fase 1, al usar ensembling para clasificar los flujos, se implementa una decisión que combina los hallazgos de los clasificadores base implementados en Slips. En particular la técnica seleccionada de votación con peso, elige para este caso la decisión del algoritmo base que mejor clasifica.

En la Fase 2, el ensembling se hace agrupando flujos por destino, y en cada uno de estos grupos, por protocolo y estado. Esto permite detectar conexiones maliciosas en función de las características de las conexiones entre el origen y el destino. A su vez, que la decisión sea por destino, da lugar a sumar en la Fase 3 la información de Threat Intelligence para cada uno de ellos.

El aplicar ensembling en la Fase 3 con la información de TI enriquece el conocimiento del comportamiento del host origen, en cuanto a lo informan los feeds de Threat Intelligence respecto a los destinos con los que el mismo se conecta.

Al dividir el problema en fases, el modelo se hace adaptable porque es posible mejorar la técnica de ensembling en una de las fases sin afectar a las otras. El entrenamiento de cada fase es independiente, con lo cual se puede cambiar el modelo de una fase para mejorar la detección sin afectar la definición de las otras fases, aunque sí mejorando mejorando sus resultados como consecuencia.

En lo que hace a mi desarrollo profesional, este trabajo me motiva a continuar esta línea de trabajo, relacionada a la inteligencia de datos y a la ciberseguridad, dentro del ámbito de CERTUNLP y del LINTI. En ese marco se proponen un conjunto de posibles líneas futuras de investigación y desarrollo:

- Incluir otras fuentes de threat intelligence en la Fase 3 de la metodología, a fin de mejorar la detección de hosts infectados.

- Integrar Slips y el módulo de ensembling como feed a CERTUNLP, a partir de la propuesta realizada en esta tesis.

- Analizar la efectividad de la metodología de ensembling propuesta como una línea de investigación para formar recursos humanos.
- Analizar la aplicabilidad mecanismo propuesto para tráfico IOT, con el fin de detectar dispositivos infectados.
- Generar datasets a partir de información provista por distintas herramientas de monitoreo de red, que puedan ser usados para aplicar técnicas de aprendizaje automático y ensemble learning en particular, a fin de detectar hosts infectados.
- Estudiar la aplicabilidad de ensembling learning para detectar distintos ataques de seguridad en redes.

La realización de esta tesis también me ha permitido adquirir experiencia en cuanto a metodología de trabajo e investigación, que es una habilidad fundamental para aportar en la formación de recursos humanos en mi rol docente de grado y postgrado.

Bibliografía

[1] Rahul Agarwal. The 5 classification evaluation metrics every data scientist must know. and when exactly to use them? 2019. *https://towardsdatascience.com/the-5-classification-evaluation-metrics-you-must-know-aa97784ff226* (accedido en enero de 2020).

[2] Kamila Babayeva. Introducing kalipso: the new interactive gui of the stratosphere linux ips. *Stratosphere Research Blog*, 2019. *https://www.stratosphereips.org/blog/introducing-kalipso-the-gui-of-slips*.

[3] Giuseppe Bonaccorso. *Machine Learning Algorithms*. Packt Publishing Ltd, 2018.

[4] Michael Bowles. *Machine Learning in Python: Essential Techniques for Predictive Analysis*. Wiley, 2015.

[5] Matt Bromiley. Threat intelligence: What it is, and how to use it effectively. *SANS.edu Graduate Student Research*, 2016.

[6] Jason Brownlee. *Machine Learning Mastery With Python*. Google Books (ebook), 2017.

[7] Jason Brownlee. A gentle introduction to k-fold cross-validation. 2018. *https://scikit-learn.org/stable/* (accedido en agosto de 2019).

[8] CERTUNLP. Ngen - csirt incident report system. *https://github.com/CERTUNLP/NgenBundle* (accedido en agosto de 2020).

[9] CERTUNLP. Sitio institucional de certunlp. *http://www.cert.unlp.edu.ar*(accedido en julio de 2020).

[10] Anirudh Ramachandran et al. Filtering spam with behavioral blacklisting. *Proceedings of the 14th ACM conference on Computer and communications security*, pág. 342–351, 2007.

[11] Caleb Neale et al. Cross validation: A beginner's guide. 2019. *https://towardsdatascience.com/cross-validation-a-beginners-guide-5b8ca04962cd* (accedido en enero de 2020).

[12] Emmanouil Vasilomanolakis et al. Taxonomy and survey of collaborative intrusion detection. *ACM Computing Surveys (CSUR)*, 47:1–33, 2015.

[13] Emna Bahri et al. Approach based ensemble methods for better and faster intrusion detection. *Computational Intelligence in Security for Information Systems*, pág. 17–24, 2011.

[14] Emna Bahri et al. A survey of intrusion detection systems based on ensemble and hybrid classifiers. *Computers Security*, 65:135–152, 2017.

[15] Harm Griffioen et al. Quality evaluation of cyber threat intelligence feeds. *International Conference on Applied Cryptography and Network Security (ACNS)*, 2020.

[16] J. Quittek et al. Requirements for ip flow information export (ipfix), rfc 3917 (informational). 2008. *http://www.ietf.org/rfc/rfc3917.txt*.

[17] Lars Buitinck et al. Design for machine learning software: experiences from the scikit-learn project. *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, págs. 108–122, 2013.

[18] Luca Didaci et al. Ensemble learning for intrusion detection in computer networks. 2002.

[19] Omar Y Al-Jarrah et al. Data randomization and cluster-based partitioning for botnet intrusion detection. *IEEE Trans Cybern*, 46:1796–1806, 2016.

[20] Paula Venosa et al. Ensembling to improve infected hosts detection. *CACIC 2019*, págs. 1251–1260, 2019. doi:https://doi.org/10.1007/978-3-030-48325-8.

[21] Paula Venosa et al. A better infected hosts detection combining ensemble learning and threat intelligence. *Computer Science - CACIC 2019 - Springer*, págs. 354–365, 2020.

[22] Sahrom Abu et al. Cyber threat intelligence – issue and challenges. *Indonesian Journal of Electrical Engineering and Computer Science*, 10:371–379, 2018. doi: 10.11591/ijeecs.v10.i1.pp371-379.

[23] Sannasy Ganapathy et al. Intelligent feature selection and classification techniques for intrusion detection in networks: a survey. *EURASIP Journal on Wireless Communications and Networking*, pág. 271, 2013.

[24] Tsaia Chih-Fong et al. Intrusion detection by machine learning: A review. *Expert Systems with Applications*, 36:11994–12000, 2009.

[25] Gianluigi Folino y Pedro Sabatino. Review ensemble based collaborative and distributed intrusion detection systems: A survey. *Journal of Network and Computer Applications. Elsevier Journal*, 66:1–16, 2016.

[26] Shadowserver Foundation. ShadowServer. *https://www.shadowserver.org/* (accedido en agosto de 2020).

[27] Jose Martinez Heras. Ensembles: voting, bagging, boosting, stacking. *https://iartificial.net/ensembles-voting-bagging-boosting-stacking/* (accedido en julio de 2020).

[28] Dita Hollmannová. Writing slips module. *Stratosphere Research Blog*, 2020. *https://www.stratosphereips.org/blog/2020/1/31/writing-a-slips-module.*

[29] Sam Humphries. Threat intelligence feeds: Keeping ahead of the attacker. *https://www.exabeam.com/siem/threat-intelligence-feeds/*(accedido en agosto de 2020).

[30] Ponemon Institute. The second annual study on exchanging cyber threat intelligence: There has to be a better way. *https://www.ponemon.org/blog/the-second-annual-study-on-exchanging- cyber-threat-intelligence-there-has-to-be-a-better-way* (accedido en julio de 2020).

[31] Stratosphere Lab. Malware Capture Facility Project. . *https://www.stratosphereips.org/datasets-malware*.

[32] Stratosphere Lab. Stratosphere IPS. . *https://www.stratosphereips.org/stratosphere-ips-suite*.

[33] Stratosphere Lab. Stratosphere Linux IPS (Slips) version 0.6.8. . *https://github.com/stratosphereips/StratosphereLinuxIPS/tree/develop*.

[34] Stratosphere Lab. Stratosphere research laboratory. . *https://www.stratosphereips.org/*(accedido en junio de 2019).

[35] Rostislav Listvan. Introducing flow formats and their differences. 2018. *https://www.flowmon.com/en/blog/introducing-flow-formats-and-their-differences* (accedido en julio de 2020).

[36] Long Mai y Dong Kun Noh. Cluster ensemble with link-based approach for botnet detection. *Journal of Network and Systems Management*, pág. 616–639, 2017.

[37] Sean Tavarez Miller y Curtis Busby-Earle. Multi-perspective machine learning a classifier ensemble method for intrusion detection. págs. 7–12. 2017. doi: 10.1145/3036290.3036303.

[38] Sean Tavarez Miller y Curtis Busby-Earle. Multi-perspective machine learning (mpml) a machine learning model for multi-faceted learning problems. págs. 363–368. 2017. doi:10.1109/CSCI.2017.60.

[39] Demir Necati y Dalkilic Gökhan. Modified stacking ensemble approach to detect network intrusion. *Turkish Journal of Electrical Engineering & Computer Sciences*, 26:418 – 433, 2018.

[40] onlineconfusionmatrix project. Confusion matrix. *http://onlineconfusionmatrix.com/* (accedido en julio de 2019).

[41] opensource.com project. What is docker? *https://opensource.com/resources/what-docker* (accedido en junio de 2020).

[42] Spamhaus Organization. The Spamhaus Project. *https://www.spamhaus.org/* (accedido en agosto de 2020).

[43] Zane Pokorny. What is threat intelligence? definition and examples. 2019. *http://www.ietf.org/rfc/rfc3917.txt* (accedido en julio de 2020).

[44] Zeek Project. base/protocols/conn/main.zeek. . *https://docs.zeek.org/en/current/scripts/base/protocols/conn/main.zeek.html* (accedido en mayo de 2020).

[45] Zeek Project. zeek an open source network security monitoring tool. . *https://zeek.org/* (accedido en julio de 2019).

[46] Sirakorn. Illustration of a boosting method for ensemble learning. 2020. *https://commons.wikimedia.org/wiki/File:Ensemble_Boosting.svg* (accedido en julio de 2020).

[47] Sirakorn. Illustration of a bootstrap aggregating (bagging) method for ensemble learning. 2020. *https://commons.wikimedia.org/wiki/File:Ensemble_Bagging.svg* (accedido en julio de 2020).

[48] SOURCEFORGE. Nfdump. *http://nfdump.sourceforge.net/* (accedido en abril de 2020).

[49] Stratosphere. Stratosphere laboratory datasets. 2015. *https://www.stratosphereips.org/datasets-overview* (accedido en junio de 2019).

[50] Spung Stunga. Figure stacking. 2016. *https://commons.wikimedia.org/wiki/File:Stacking.png* (accedido en julio de 2020).

[51] Wiem Tounsi y Helmi Rais. A survey on technical threat intelligence in the age of sophisticated cyber attacks. *Computers security*, 72:212–233, 2018.

[52] Juan Vanerio y Pedro Casas. Ensemble-learning approaches for network security and anomaly detection. *Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks - ACM*, 1:1–16, 2017.

[53] VirusTotal. Reports. . *https://support.virustotal.com/hc/en-us/articles/115002719069-Reports* (accedido en abril de 2020).

[54] VirusTotal. Virustotal home page. . *https://www.virustotal.com* (accedido en marzo de 2019).

[55] Jeff Weisberg. Argus - the all seeing. *http://argus.tcp4me.com/* (accedido en julio de 2019).

[56] Cha Zhang y Yunqian Ma. *Ensemble Machine Learning: Methods and Applications*. Springer, 2012.

[57] Zhi-Hua Zhou. *Ensemble Methods: Foundations and Algorithms. 1st Edition*. Chapman and Hall/CRC, 2012.