Adversarial Machine Learning con enfoque de *Black-Box* para mejorar la robustez en modelos de *Machine Learning*

Realizado por

Juan Fernando Angulo Salvador

Danna Sofía García Trujillo

Miguel Andrés Sarasti Taguado

Universidad ICESI
Facultad de Ingeniería, Diseño y Ciencias Aplicadas
Ingeniería de Sistemas e Ingeniería Telemática
Cali
2023

Adversarial Machine Learning con enfoque de *Black-Box* para mejorar la robustez en modelos de *Machine Learning*

Realizado por

Juan Fernando Angulo Salvador
Danna Sofía García Trujillo
Miguel Andrés Sarasti Taguado

Proyecto de grado

Dirigido por
Christian Urcuqui Lopez, MSc.

Universidad ICESI
Facultad de Ingeniería, Diseño y Ciencias Aplicadas
Ingeniería de Sistemas e Ingeniería Telemática
Cali
2023

Índice general

Resumen

Actualmente, los algoritmos de *Machine Learning* (*ML*) son codiciados por su poder de cómputo, debido a esto, los cibercriminales buscan aprovechar las vulnerabilidades existentes en dichos algoritmos.

La idea de este proyecto nace de trabajos de grado previos [1, 2], donde desarrollaron estrategias de *Secure Learning* y *Reinforcement Learning* para aumentar la robustez de modelos de *ML*. En el siguiente proyecto se desarrolló una metodología de *Adversarial Machine Learning* con enfoque de *Black-Box* para mejorar la robustez de modelos de *ML* para la clasificación de *malware* con la estrategia de *Deep Reinforcement Learning*.

Para cumplir con estos objetivos se llevó a cabo lo siguiente. Primero, se generó unas muestras de *malware* en un entorno seguro. Segundo, la creación del modelo de defensa, siguiendo las fases de *CRISP-DM*. Tercero, la creación del modelo de ataque y la generación de datos adversarios. Por último, el *Adversarial Training*, en donde se re-entrenó el modelo de defensa para mejorar su *accuracy* con los datos adversarios.

Palabras clave: Inteligencia Artificial, *Machine Learning*, *Adversarial Machine Learning*, *Adversarial Training*, *Black-Box*, *Malware*.

Abstract

Currently, Machine Learning (ML) algorithms are coveted for their computing power. Because of this, cybercriminals seek to exploit existing vulnerabilities in these algorithms.

The idea of this project stems from previous graduate work [1, 2], where they developed Secure Learning and Reinforcement Learning strategies to increase the robustness of ML models. In the following project we developed an Adversarial Machine Learning methodology with a Black-Box approach to improve the robustness of ML models for malware classification with a Deep Reinforcement Learning strategy.

To meet these objectives, the following was carried out. First, we generated First, malware samples were generated in a secure environment. Second, the creation of the defense model, following the CRISP-DM phases. Third, the creation of the attack model and the generation of the attack model. and the generation of adversarial data. Finally, the Adversarial Training, where the defense model was re-trained to improve its performance.

Keywords: Artificial Intelligence, Machine Learning, Adversarial Machine Learning, Adversarial Training Learning, Adversarial Training, Black-Box, Malware.

Lista de acrónimos

- **AI:** Inteligencia artificial (***A**rtificial **I**ntelligence*)
- **AML:** Aprendizaje automático adversarial (***A**dversarial **M**achine **L**earning*)
- **DRL:** Aprendizaje por refuerzo profundo (***D**eep **R**einforcement **L**earning*)
- **GAN:** Red generativa adversaria (***G**enerative **A**dversarial **N**etwork*)
- **ML:** Aprendizaje automático (***M**achine **L**earning*)
- **RL:** Aprendizaje por refuerzo (***R**einforcement **L**earning*)
- **DL:** Aprendizaje profundo (***D**eep **L**earning*)

Glosario de términos

- ***Adversarial Machine Learning*:** Término empleado en el área de aprendizaje automático que considera los escenarios en los que los sistemas de *ML* pueden enfrentarse a potenciales atacantes adversarios.
- ***Black-Box*:** Término utilizado para referirse al ataque de un modelo de *ML* donde se desconoce previamente los parámetros, conjuntos de datos y algoritmos utilizados.
- ***Deep Learning*:** Término empleado en el área del aprendizaje automático en el que se habla de una técnica de aprendizaje automático basada en el modelo de red neuronal.
- ***Deep Reinforcement Learning*:** Término empleado en el área de aprendizaje automático en donde combina el *RL* y el aprendizaje profundo.
- ***Generative Adversarial Network*:** Término empleado en el área de aprendizaje automático en donde se emplean dos redes neuronales, un generador y un discriminador, que compiten para mejorar el desempeño de ataque, si el generador logra generar datos falsos que el discriminador clasifica como verdaderos, o defensivo, si el discriminador logra clasificar los datos del generador como falsos.
- ***Gray-Box*:** Término utilizado para referirse al ataque de un modelo de *ML* donde se conocen algunos atributos, como los parámetros, el conjunto de datos o los algoritmos utilizados, pero el resto se desconocen previamente.
- ***Machine Learning*:** Término empleado en el área de la *AI* en donde se busca que un sistema tenga la capacidad de aprender en entornos variables.
- ***Overfitting*:** Término empleado para referirse cuando un modelo de *ML* se entrena excesivamente sobre los datos y no logra generalizar el conocimiento.
- ***Reinforcement Learning*:** Término empleado en el área de aprendizaje automático en donde se busca que el algoritmo esté en la capacidad de encontrar el conjunto de operaciones más adecuadas para cumplir un objetivo.
- ***Secure Learning*:** Término empleado para referirse cuando un modelo de *ML* tiene un buen desempeño defensivo ante ataques adversarios, cuyo propósito es engañar al modelo.
- ***White-Box*:** Término utilizado para referirse al ataque de un modelo de *ML* donde se conoce los parámetros, los conjuntos de datos y los algoritmos utilizados.

Índice de figuras

Índice de tablas

1. Motivación y antecedentes

1.1. Contexto

La inteligencia artificial (*AI*) y, en particular, el aprendizaje automático (*ML*) han crecido rápidamente en los últimos años en el contexto del análisis de datos y la computación que suele permitir que las aplicaciones funcionen de forma inteligente [4]. Conforme avanza el desarrollo e implementación de nuevos algoritmos de *Machine Learning*, así mismo resulta inevitable que crezca el afán e interés por vulnerar y manipular estos algoritmos por parte de cibercriminales con el propósito de sacar ventaja y/o sabotear a la competencia. Los problemas de seguridad en los modelos de *ML* ocurren porque las técnicas de los modelos tradicionales no fueron desarrolladas originalmente para tratar con ataques inteligentes y adaptativos.

La mayoría de los escenarios en ciberseguridad, como la detección de *malware* y la detección de intrusos, pueden ser vistos como problemas de clasificación. El aprendizaje automático es eficaz en cuestiones de clasificación y muestra un excelente rendimiento en ciberseguridad, por lo que se aplica ampliamente en este ámbito. Sin embargo, con la aparición de ejemplos adversos, los sistemas de aprendizaje automático se enfrentan a nuevos desafíos en este campo tan crítico para la seguridad [5].

Como respuesta a esta creciente amenaza en la seguridad y privacidad a los modelos de *ML*, emerge el campo de investigación *Adversarial Machine Learning*. Mediante este campo, se exploran tres direcciones: (a) reconocimiento de las vulnerabilidades tanto en las etapas de entrenamiento como de inferencia; (b) desarrollando ataques correspondientes, evaluando el impacto estimado a los modelos; (c) idear contramedidas con el fin de mejorar la seguridad de los modelos frente a estos ataques [6]. Teniendo en cuenta las tres direcciones, deciden tomar la exploración dentro de un contexto *Black-Box*, en el que los atacantes no tienen acceso a las estructuras ni parámetros detallados de los modelos que atacan.

1.2. Antecedentes del problema

En [1] se exponen algunas de las vulnerabilidades que se pueden encontrar en modelos de *Machine Learning* (*ML*) y explican las causas de estas, en especial los que se enfocan en la clasificación de *Android Malware*. El problema se aborda desde el campo del Secure Learning dentro de un contexto *White-Box* utilizando *Adversarial Training* para el mejoramiento de la seguridad en los modelos clasificatorios de *ML*. Para

trabajos futuros se propone que continúen la misma línea de investigación, mejorar lo desarrollado desde el *Reinforcement Learning* e incluso explorar opciones como el *Deep Reinforcement Learning* y las *Generative Adversarial Networks* (*GAN*), en contextos de menor información (*Gray-Box* y *Black-Box*) sobre los modelos que se atacan.

Por esas propuestas, en [2] continúan el trabajo de [1], está vez también abordando el problema desde el campo del *Reinforcement Learning* y en un contexto *Gray-Box*, donde proponen para trabajos futuros la modificación de un *malware* real y abordar el problema en un contexto *Black-Box*.

1.3. Justificación

La importancia de resolver el problema de la robustez en modelos de *ML* recae en cubrir el enfoque más realista, un contexto de *Black-Box*, con este se lograría abordar todos los contextos posibles. Esto daría paso a modelos capaces de reconocer y soportar ataques adversos en escenarios en donde los atacantes tengan mucha, poca e incluso ninguna información. Lo anterior garantiza un proceso de aprendizaje seguro, y por tanto predicciones más acertadas y confiables, que contribuyan al desarrollo de nuevo conocimiento.

2. Descripción del problema

2.1. Formulación del problema

Actualmente, los algoritmos de *ML* son codiciados por su poder de cómputo, debido a esto, los cibercriminales buscan aprovechar las vulnerabilidades existentes en dichos algoritmos.

Algunas de estas son la linealidad, o poca flexibilidad de algunos modelos de *ML*, dado que el conjunto de datos de entrenamiento y el de evaluación pertenecen a la misma distribución estadística, lo que permite engañarlos con mayor facilidad [7]; el sobreajuste (*Overfitting*) causado cuando los modelos crecen en complejidad; y por último, los ataques exploratorios y causativos, donde los atacantes abusan de la naturaleza auto-adaptativa de los modelos de *ML* y modifican los componentes del modelo, como son los datos, representación o el mismo algoritmo.

Esto ha causado que los cibercriminales puedan: modificar los resultados de las predicciones del sistema de *ML*, extorsionar a las compañías por la información que obtienen de los ataques, entre otros. Por lo que también las organizaciones y los sistemas pierden la confianza de los usuarios.

3. Objetivos

3.1. Objetivo general

Desarrollar una metodología de *Adversarial Machine Learning* con enfoque de *Black-Box* para mejorar la robustez en modelos de *Machine Learning*.

3.2. Objetivos específicos

1. Proponer un modelo de ataque adversario con enfoque de *Black-Box*.
2. Evaluar un modelo de *Deep Reinforcement Learning* para realizar perturbaciones más precisas en el proceso de aprendizaje del modelo.
3. Proponer enfoques de desarrollo que cuenten con buenas medidas de seguridad.
4. Validar la metodología desarrollada aplicándola en el modelo de defensa seleccionado.

4. Marco teórico

El marco teórico del proyecto se dividirá en dos categorías. La primera, *Machine Learning* (*ML*), en donde se abordan los conceptos de Aprendizaje supervisado, Aprendizaje no supervisado, *Reinforcement Learning* y *Deep Reinforcement Learning*. La segunda, Ciberseguridad, en donde se abordan los conceptos de Ciberataques, *Adversarial Machine Learning*, *Generative Adversarial Network* y los tres tipos de enfoques de caja para pruebas.

4.1. *Machine Learning*

El *Machine learning* (*ML*) es la parte de la inteligencia artificial que busca que un sistema esté en la capacidad de aprender en entornos variables, sin que sea programado de forma explícita [6].

Los algoritmos de *ML* pueden resolver problemas difíciles de forma genérica. Estos algoritmos no requieren un diseño detallado explícito y en su lugar, aprenden el diseño detallado de un conjunto de datos etiquetados (es decir, un conjunto de ejemplos que ilustran el comportamiento del programa). En otras palabras, aprenden de los datos, y por eso, cuanto más grande es el conjunto de datos, más precisos se vuelven. El objetivo de un algoritmo de *ML* es aprender un modelo o un conjunto de reglas de un conjunto de datos etiquetado para que pueda predecir correctamente las etiquetas de los puntos de datos (por ejemplo, imágenes) que no están en el conjunto de datos [8].

4.1.1. Aprendizaje supervisado y no supervisado

En el aprendizaje supervisado se tiene una idea clara de las entradas, salidas y la relación de estas respecto al conjunto de datos. Además, abarca los problemas de regresión, donde se busca predecir un valor continuo, y clasificación, donde se busca obtener un valor discreto o categórico [6].

Por otro lado, en el aprendizaje no supervisado no se tienen etiquetas de salida, por lo que se debe identificar la estructura y los patrones que se encuentren en la información de los conjuntos de datos [6].

4.1.2. *Reinforcement Learning* y *Deep Reinforcement Learning*

El aprendizaje por refuerzo, también conocido como *Reinforcement Learning* (*RL*), tiene como objetivo encontrar el conjunto de operaciones más adecuadas para obtener un resultado deseado, y para ello hace uso del aprendizaje por reglas y acciones [6]. Dicho aprendizaje ocurre por medio de la interacción de un sistema o agente con el entorno, en donde el *learner* tiene que descubrir las acciones que debe tomar, por medio de un sistema de recompensas o premios; el cual, maximiza su valor si se acerca al resultado deseado y ocurre lo contrario si se aleja del objetivo [9].

En el *Deep Reinforcement Learning* (*DRL*) se usan las redes neuronales profundas para aproximar alguno de los componentes de *RL*, tales como la función de valor, la política y el modelo, empleando los pesos auto-calculados de las redes neuronales [10].

4.2. Ciberseguridad

La ciberseguridad es un término amplio que se refiere a las medidas adoptadas por entidades públicas y privadas, destinadas a garantizar la seguridad de las comunicaciones y los recursos en línea [11].

4.2.1. Ciberataques

Un ataque cibernético no es más que una forma de comprometer las funciones de la computadora en la red de una víctima o tener acceso digital no autorizado a la computadora de una víctima al eliminar las barricadas. Un ataque cibernético se considera como un ataque a un sistema informático que compromete la confidencialidad, integridad o disponibilidad de la información en ese sistema [12].

4.2.2. *Adversarial Machine Learning*

El *Adversarial Machine Learning* es un área de investigación que considera los escenarios en los que los sistemas de *ML* pueden enfrentarse a potenciales atacantes adversarios, los cuales sintetizan intencionadamente los datos de entrada para hacer que un modelo bien entrenado cometa un error. Este involucra la participación de dos partes ”adversarias”dentro de un juego denominado *minimax* (método de decisión para minimizar la pérdida máxima potencial), en el que la parte atacante aprende a producir ejemplos adversarios para engañar al defensor, mientras que el clasificador intenta defender los ataques de este tipo [13].

Los ejemplos adversos suelen definirse como entradas a un modelo de *ML* que están específicamente diseñadas para hacer que el modelo objetivo produzca salidas erróneas [14].

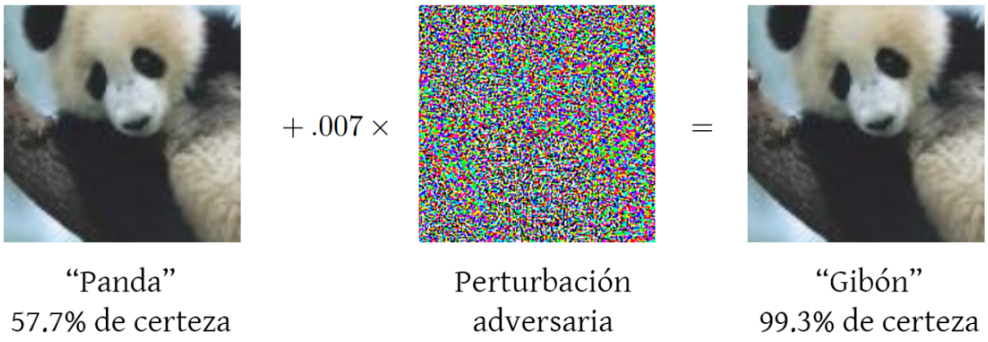

Figura 4.1: Ejemplo de ataque adversario

4.2.3. *Generative Adversarial Network*

Las redes generativas adversarias (*GAN*) son un tipo de algoritmo de inteligencia artificial diseñado para resolver el problema del modelado generativo. Las *GAN* se componen de dos modelos: un generador y un discriminador. El primero trata de capturar la distribución de los ejemplos verdaderos para la generación de nuevos ejemplos de datos, y el segundo suele ser un clasificador binario, que discrimina ejemplos generados de los ejemplos verdaderos con la mayor precisión posible como sea posible [15]. Estos dos modelos se implementan normalmente con redes neuronales, pero pueden implementarse con cualquier forma de sistema diferenciable que mapee los datos de un espacio a otro.

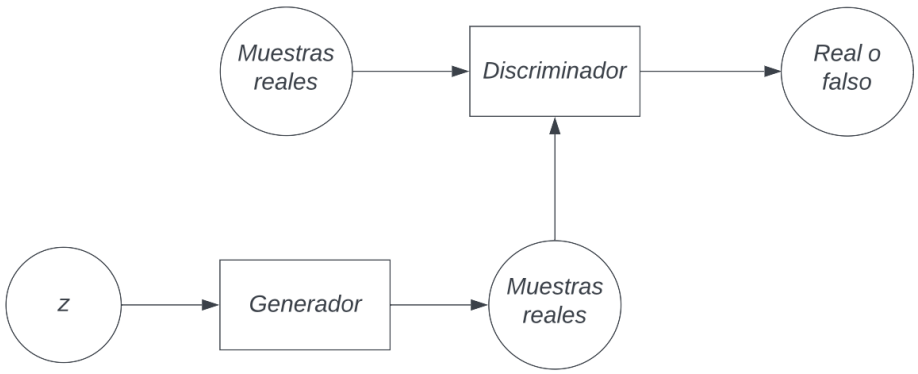

Figura 4.2: Arquitectura normal de una GAN

En el proceso de entrenamiento de las *GAN*, a medida que avanzan las iteraciones, los datos generados cada vez se acercan más a un resultado que puede engañar al discriminador, y eventualmente si el entrenamiento del generador funciona bien, el discriminante empeora al identificar entre el real y el falso, clasificando los datos falsos como reales, por lo que disminuye su exactitud.

4.2.4. *White-Box testing*

La prueba de caja blanca es una estrategia de prueba de productos en la que la estructura interna/exterior/uso de lo que se está probando es conocido por el analizador. El analizador escoge las entradas para practicar los caminos a través del código y decide los rendimientos de ajuste [16].

4.2.5. *Gray-Box testing*

La prueba de caja gris es un sistema de prueba de productos que es una mezcla de la estrategia de prueba de caja negra y la técnica de pruebas de caja blanca. En la prueba de caja gris, la estructura interna es algo conocida, y esto implica tener acceso a las estructuras de datos y algoritmos internos para diseñar los casos de prueba, pero se prueba a nivel de usuario o de caja negra, es decir, como si no se tuviese conocimiento de la estructura interna [16].

4.2.6. *Black-Box testing*

La prueba de caja negra, también llamada prueba de comportamiento, es una estrategia de prueba de productos en la que el analizador desconoce la estructura interior/contorno/ejecución de lo que se está probando [16].

5. Estado del arte

5.1. *Secure Learning* para detección de *Android Malware*

En la tesis [1], se realizaron avances mediante la implementación del enfoque de aprendizaje seguro (*Secure Learning*) a un modelo que permitía predecir la clasificación de APKs *Android*. En donde se abordó el *Adversarial Training* desde el punto de vista en el que se tiene conocimiento sobre los datos de aprendizaje, la representación de las características y el algoritmo del modelo (*White-Box*). Adicionalmente, mediante el aprendizaje seguro (*SL*) y utilizando *Reinforcement Learning* (*RL*) se disminuyeron las vulnerabilidades del modelo de clasificación. En [1], el modelo pasó de reconocer 0 % de los datos adversarios al 77 % de ellos. Por tanto, se concluyó que el modelo era mucho más robusto ante ataques adversarios. Adicionalmente, mediante el ajuste de ciertos parámetros, se mejora el *accuracy* del modelo de defensa.

Como trabajo a futuro [1] propone mejoramiento de *RL* para realizar perturbaciones más precisas utilizando *Deep Reinforcement Learning* o *Generative Adversaria Networks* (*GAN*). Adicionalmente, [1] argumenta la necesidad de proponer un método de programación para realizar un *exploit* de un modelo de *ML* con enfoque de *Black-Box*.

[1] aporta bases fundamentales dado que proporciona una perspectiva diferente a la que se abordará en este proyecto, mediante la cual se puede analizar qué se requiere para poder lograr disminuir aún más las vulnerabilidades del modelo clasificatorio. Adicionalmente, los trabajos futuros que propone [1] significan una motivación para el presente proyecto, dado que son tomados en cuenta para el desarrollo del mismo y así poder tener una perspectiva más cercana al cibercriminal que la que se tuvo en [1].

5.2. *Secure Learning* y *Deep Reinforcement Learning* para la detección de *Android Malware*

En [2], se realizaron avances en la reducción de las vulnerabilidades a ataques con enfoque adversario en algoritmos de *Machine Learning* para la clasificación de *malware* en *Android* utilizando *Secure Learning* y *Reinforcement Learning*. Se presenta un modelo, que a través de *DRL* logra mejorar significativamente la robustez del clasificador de *malware*. Respecto a la precisión de los datos adversarios se encontró que el algoritmo de *DRL* alcanza una modificación mínima de 5,96628 respecto a los valores de entrada, permitiendo observar que los datos adversarios presentan una gran similitud con los datos de *malware* generados.

Como trabajo futuro, proponen un método de ataque adversario y posterior entrenamiento desde un enfoque de *Black-Box*.

[2] complementa las bases que posteriormente había proporcionado [1], dado que aborda la problemática desde un punto de vista diferente al que se abordará en el presente proyecto y al que aborda [1], permitiendo así analizar los avances y mejoras que se realicen en el presente proyecto frente a los resultados obtenidos tanto en [1] como en [2].

5.3. *NATTACK: Learning the Distributions of Adversarial Examples for an Improved Black-Box Attack on Deep Neural Networks*

En [17], se presenta un algoritmo de ataque adverso con enfoque de caja negra, explican cómo se planteó y cómo fue usado para vencer redes neuronales profundas (*DNN*) del estado del arte.

Proponen un algoritmo que encuentra la densidad de probabilidad sobre una región centrada en las entradas, sin la necesidad de acceder a las capas internas o pesos de las *DNN*s, clasificándose como un método de ataque adversario con enfoque de caja negra. Con un criterio de optimización motivado por la estrategia de evolución natural (*NES*). Los resultados obtenidos muestran un desempeño positivo, dado que logran vencer a las 13 *DNN*s del estado del arte propuestas, con un porcentaje de éxito mayor o igual al de otros métodos de ataque con enfoque de caja blanca.

Finalmente, proponen como futuros trabajos mejorar el entrenamiento por medio de un muestreo eficiente desde las distribuciones de los datos.

5.4. *A Brute-Force Black-Box Method to Attack Machine Learning-Based Systems in Cybersecurity*

En [5], se expone la vulnerabilidad a muestras adversarias que presentan los algoritmos de *Machine Learning*, explican el cómo funcionan este tipo de ataques y ahondan en el impacto de estos en problemas de ciberseguridad como lo son la detección de intrusos en la red y la detección de *Android malware*.

Proponen un nuevo y sencillo método de ataque *Black-Box* conocido como *brute-force attack method* (*BFAM*) para evaluar mejor la robustez de los sistemas basados en el aprendizaje automático en ciberseguridad contra muestras adversas. El método propuesto determina las características a perturbar basándose en las puntuaciones de confianza emitidas por los clasificadores objetivos y produce las muestras adversas

manipulando directamente las características de los vectores de entrada. El *BFAM* es sencillo de implementar, evita el inestable entrenamiento de los métodos de ataque basados en *GAN* (*AAM-GAN* como estado del arte) y tiene control completo sobre el proceso de generación de muestras adversas.

En cuanto a resultados, *BFAM* cuesta mucho menos tiempo para generar muestras adversas que *AAM-GAN* en varios escenarios de ciberseguridad. *BFAM* supera a *AAM-GAN* en la mayoría de los casos. Finalmente, para trabajo futuro, proponen mejorar *BFAM* para que pueda generar muestras adversas sólo con las etiquetas y pueda utilizarse para más escenarios y clasificadores.

5.5. Matriz de estado del arte

En la siguiente matriz se resumen las características principales de los *papers* en el estado del arte *versus* las características principales de nuestro proyecto.

Tabla 5.1: Matriz resumen del estado del arte

Características / Papers	[1]	[2]	[17]	[5]	**Este proyecto**
Black-Box			X	X	X
Machine Learning	X	X	X	X	X
Adversarial Machine Learning		X	X	X	X
Secure Learning	X	X			X
Reinforcement Learning	X				X
Deep Reinforcement Learning		X	X	X	X
Generative Adversarial Network		X		X	X
Malware (Ransomware y Troyanos)					X
Laboratorio - Ejecución Segura					X

En este proyecto, se combinarán las anteriores propuestas de solución enunciadas en [1] y [2], lo que incluye *SL*, *RL* y *DRL*, esta vez dentro de un contexto *Black-Box* y apoyado por el uso de *GAN*s, para dar solución a este problema de *ML* dentro del campo del *AML*. Además, es importante destacar que este trabajo aborda otro diferencial significativo en el campo de la ciberseguridad, específicamente en relación con el análisis de *ransomwares* y *malwares* troyanos. Para ello, se ha explorado y estudiado tres modelos de defensa que permite simular y analizar el comportamiento de estas familias de *malwares*. Esta investigación representa un avance importante en el desarrollo de soluciones más efectivas y completas para la detección y prevención de amenazas en el ámbito del *AML*.

6. Metodología

Para el desarrollo del proyecto, se seguirán los lineamientos planteados por la metodología *CRISP-DM* (*Cross-industry standard process for data mining*). Esta es una metodología especialmente diseñada para resolver problemas que conciernen a la ciencia de datos, por lo que se ajusta perfectamente a este proyecto.

6.1. Fases de desarrollo del proyecto

El ciclo vital de la metodología contiene seis fases con flechas que indican las dependencias más importantes y frecuentes entre fases. La secuencia de las fases no es estricta. De hecho, la mayoría de los proyectos avanzan y retroceden entre fases si es necesario [18].

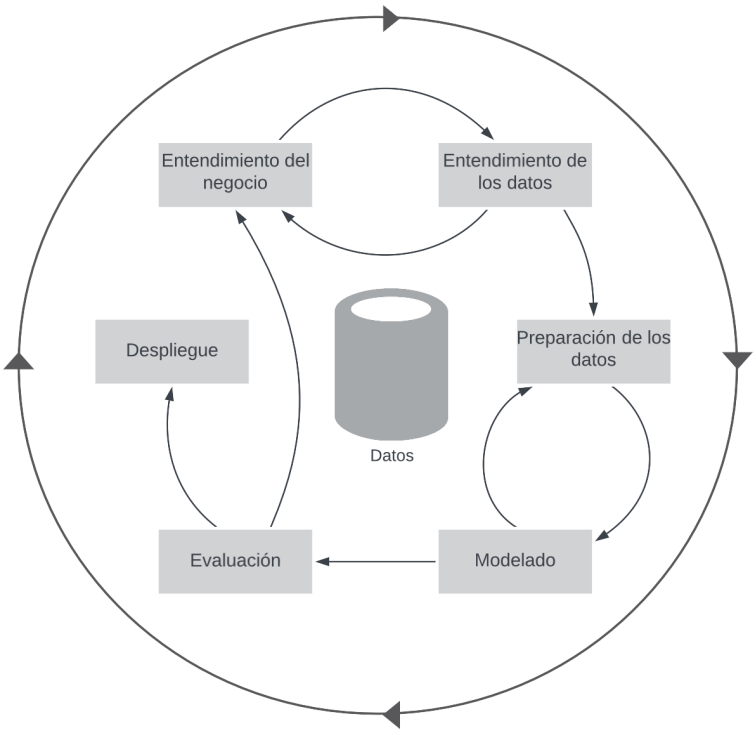

Figura 6.1: Fases de *CRISP-DM*

6.1.1. Entendimiento del negocio

Mediante el análisis de trabajos e investigaciones anteriores, logramos obtener una comprensión clara del negocio y la importancia de la ciberseguridad en el contexto de este proyecto. La revisión minuciosa de una amplia variedad de artículos y libros especializados en el tema fue esencial para completar esta fase con éxito.

6.1.2. Entendimiento de los datos

En la fase de entendimiento de los datos, se tomó el *dataset* de tráfico de red utilizado en la construcción e implementación de [3], nuestro modelo de defensa de referencia. El *dataset* [19] consta de capturas de tráfico (.pcap) de aplicaciones tanto benignas como malignas (*malware*) de las cuales nos quedamos solo con uno de sus campos:

- Paquetes UDP

El *dataset* completo, consta de 14 capturas de tráfico benigno y 10 capturas de tráfico *malware*, que en conjunto pesan aproximadamente 20 GB.

Adicionalmente se incluyeron capturas de las siguientes muestras de *malware*: *Tesla Crypt*, *Trojan Asprox New*, *Trojan Asprox Old* y *AllElectroRat*. Estas fueron generadas a través de una simulación controlada en un laboratorio de tráfico de red seguro.

Por último, incluimos una variable llamada *label*, que puede tomar valores de 0 para indicar tráfico de una aplicación benigna, o de 1 para indicar tráfico de una aplicación de *malware*.

6.1.3. Preparación de los datos

Para la preparación de los datos, estas capturas .pcap se pasaron por un algoritmo de extracción de características, y se exportan los datos en un formato .csv, para finalmente realizarles una transformación a su tipo de dato correcto y eliminar aquellas filas que pudieran causar error a la hora de entrenar el modelo.

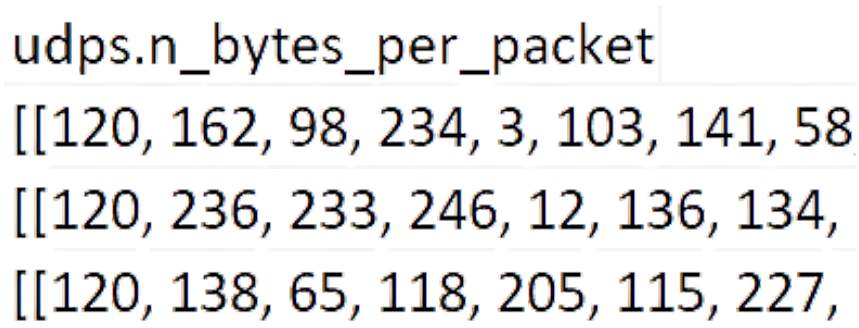

Figura 6.2: Salida del algoritmo de extracción de características

6.1.4. Modelado

En la fase de modelado, se entrenaron 3 modelos de *ML*, entre los cuales se encuentran:

- *Naive Bayes*
- *Random Forest*
- DeepMAL

DeepMAL es un modelo de *DL* que es capaz de capturar las estadísticas subyacentes del tráfico malicioso, entrenado con un flujo de bytes que no requieren de pasos de preprocesamiento o intervención de un experto en el dominio, esto con el fin de que el enfoque sea genérico y flexible [3].

En el análisis y entrenamiento de DeepMAL, se evitó utilizar los encabezados de los protocolos IP y de transporte debido a su falta de representatividad y sesgos en capturas de tráfico generadas en entornos controlados. En su lugar, se enfoca únicamente en la información de la carga útil (*payload*) de cada paquete de red capturado; ésta es la parte de los datos del paquete que lleva información específica de la aplicación, o contenido transmitido. Al enfocarse en la carga útil de los paquetes, se evita el sesgo permitiendo una representación más realista, y potencialmente, una mejor detección y clasificación.

DeepMAL establece la cantidad de bytes por cada paquete mediante un análisis simple del conjunto de datos. La distribución del tamaño de la carga útil de los paquetes para las muestras de tráfico tanto maligno como benigno es similar para tamaños de carga útil por debajo de 750 bytes y por encima de 1250 bytes, pero marcadamente diferentes entre estos dos límites. Por tanto, se establece que el número de bytes de los paquetes será 1024 bytes, lo cual representa la cantidad adecuada para capturar la diferencia entre malware y actividad benigna.

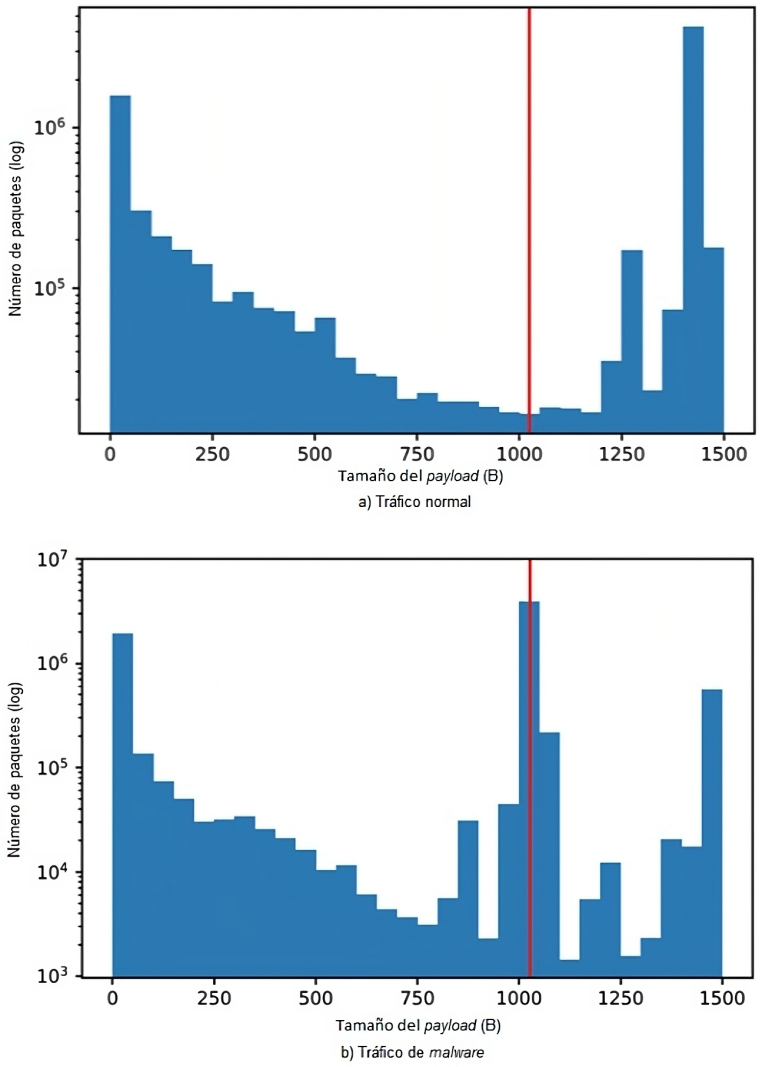

Figura 6.3: Carga útil de paquetes con *malware* y normales [3]

Para el entrenamiento de estos modelos, se realizó una partición de los datos con un 80 % destinados para el entrenamiento, y un 20 % para la validación.

6.1.5. Evaluación

Para la evaluación se tuvieron en cuenta principalmente las métricas de *Accuracy* y su matriz de confusión para cada modelo, pues nos interesaba especialmente que fuera baja la tasa de falsos positivos, tanto como la de falsos negativos. Adicionalmente se tuvieron en cuenta durante este proceso otras métricas como:

- *Precision*
- *Recall*

- *F1-score*

Finalmente, teniendo en cuenta estos criterios se selecciona el mejor modelo que se ajuste a nuestro problema.

6.1.6. Despliegue

La fase de despliegue no se incluye en los objetivos ni en el alcance de este proyecto, ya que se busca abarcar solamente las fases anteriores con el fin de generar una metodología segura y robusta antes de llegar al despliegue del modelo.

6.2. *Secure Learning* y *Deep Reinforcement Learning*

6.2.1. Ataque al modelo de defensa

El ataque se realizó desde la perspectiva de un atacante que solo tiene acceso a consultar el modelo clasificatorio, lo que permite entregar una muestra de variables desconocidas y recibe una respuesta de si es *malware* o benigno. Adicionalmente, el atacante conoce el tipo de datos de entrada del modelo de defensa, qué es el tráfico de red, en específico los paquetes UDP. El ataque se divide en los siguientes pasos:

Generación de muestras malware

Se realiza la generación de datos malware, mediante un ambiente controlado dentro de un laboratorio; este es constituido por dos máquinas virtuales, la primera cuyo sistema operativo es Linux con distribución Ubuntu y otra Windows. La máquina virtual Linux será denominada como "Máquina de análisis" y la máquina virtual Windows será denominada como "Máquina víctima".

Ambas máquinas virtuales se encuentran en una red privada virtual a la que pertenecen únicamente estas dos, esto con el fin de proteger una disipación de los *malware* a los demás dispositivos que se encuentren en la red del computador host. Ambas máquinas no cuentan con acceso a internet.

Generación de datos adversarios

La generación de datos sigue el camino trazado por el proyecto [2], donde se emplea el algoritmo de *Deep Q-learning* y se transforma en una red generativa adversaria utilizando el modelo de defensa como discriminador. En el entrenamiento, se genera un conjunto de datos con modificaciones, con la finalidad de engañar al clasificador, haciendo que clasifique muestras de *malware* como benignas.

Adversarial training

El proceso que se sigue en el *adversarial training* consiste en entrenar de nuevo el modelo de defensa, pero ahora agregando los datos adversarios generados y con su etiqueta (*label*) real de *malware*, de forma que el modelo aprenda a diferenciar muestras adversas y resulte más robusto frente a este tipo de ataques.

El *adversarial training* incluye los datos de entrenamiento originales [19], más los datos adversarios generados a partir del ataque al modelo de defensa. Para este *dataset* combinado, de nuevo se toma el 80 % de los datos para el entrenamiento y un 20 % para pruebas.

7. Resultados

7.1. Generación de muestras *malware*

La máquina víctima será la encargada de actuar como tal, por lo que será la máquina que ejecute los diferentes tipos de *malware* con los que se realizarán las pruebas. Esta máquina tiene una dirección IP perteneciente a la red privada mencionada anteriormente y como DNS se le asignó la dirección de la máquina de análisis; por lo tanto, cuando la máquina víctima intente acceder a sitios web mediante su navegador, en lugar de utilizar un servidor DNS público o el DNS proporcionado por el proveedor de servicios de Internet (ISP), la máquina víctima enviará las solicitudes de resolución de nombre de dominio a la máquina de análisis. La máquina de análisis se encargará de resolver los nombres de dominio solicitados.

La máquina de análisis tiene instaladas herramientas como Burp Suite, Inetsim y Wireshark. Inetsim es un paquete de software para simular servicios de Internet en un entorno de laboratorio y así poder analizar el comportamiento de la red ante muestras de malware; ésta herramienta admite la simulación de servicios como: HTTP, SMTP, POP3, DNS, entre otros [20]. Burp Suite es una suite de herramientas de seguridad utilizada para realizar pruebas de seguridad y análisis de tráfico web. Mediante el uso de Burp Suite junto con Inetsim se crea un entorno de pruebas de seguridad altamente controlado para analizar y evaluar el tráfico web generado por la máquina víctima.

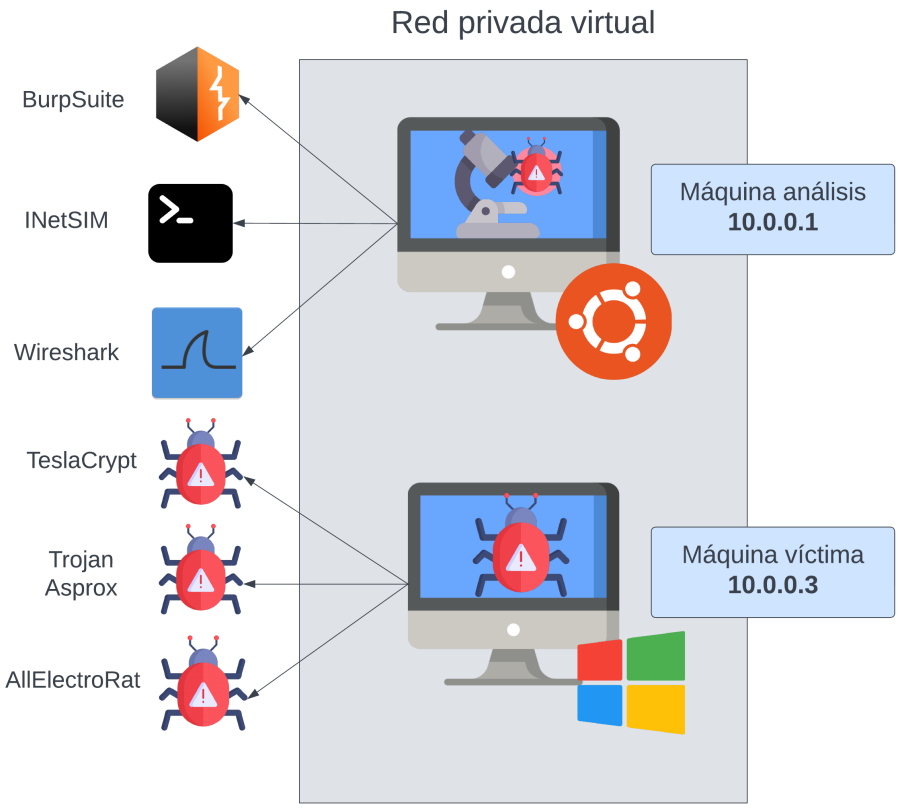

Figura 7.1: Laboratorio controlado para análisis y ejecución de *malware*.

Las muestras de *malware* utilizadas fueron *Tesla Crypt*, *ransomware* enfocado en cifrar los archivos en el sistema comprometido y luego solicitar rescate para desbloquearlos; *Trojan Asprox New* y *Trojan Asprox Old*, enfocado en infectar sistemas y convertirlos en parte de redes de computadoras infectadas y controladas de manera remota por ciberdelincuentes o entidades maliciosas, utilizada para llevar a cabo actividades maliciosas, como envío de spam, ataques de fuerza bruta y propagación de *malware* adicional; *AllElectroRat*, *malware* de acceso remoto que se utiliza para tomar el control no autorizado de sistemas comprometidos. Esto con el fin de robar información, instalar otros *malware* o realizar acciones maliciosas en el sistema comprometido.

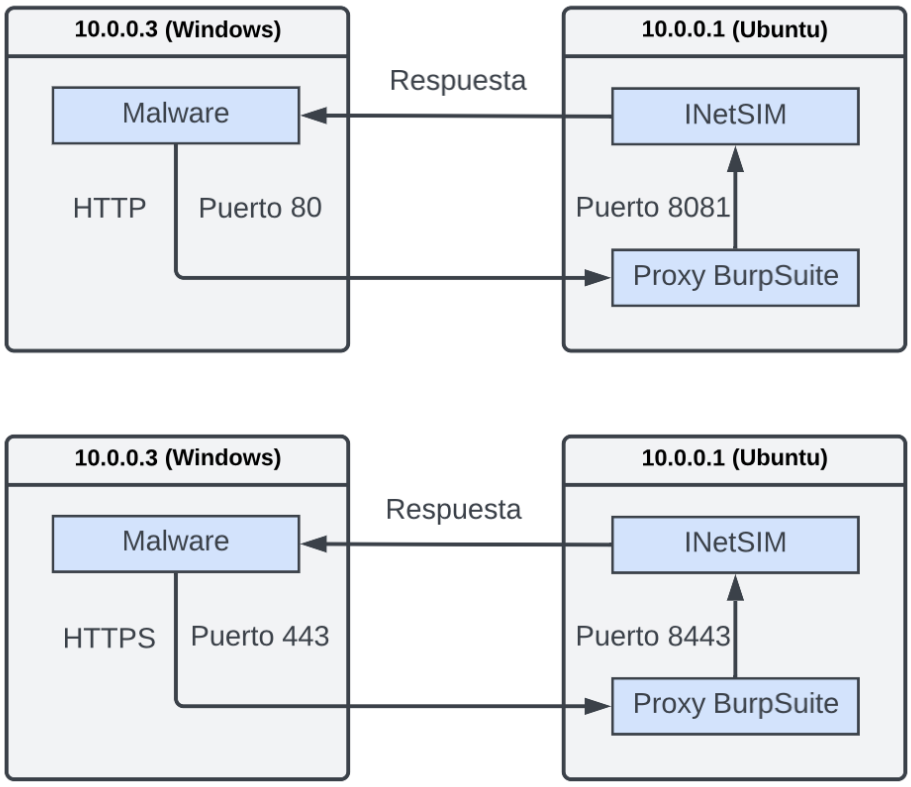

Figura 7.2: Comportamiento del laboratorio.

La figura 7.2 presenta la arquitectura del laboratorio en la que el *malware* dentro de la máquina víctima genera tráfico HTTP o HTTPS mediante el puerto 80 o 443 respectivamente, donde el Proxy dentro de BurpSuite redirigirá este tráfico al puerto 8081 o 8443 donde estará respondiendo INetSIM. La respuesta por parte de INetSIM tanto para HTTP como HTTPS es como se muestra en la figura 7.3

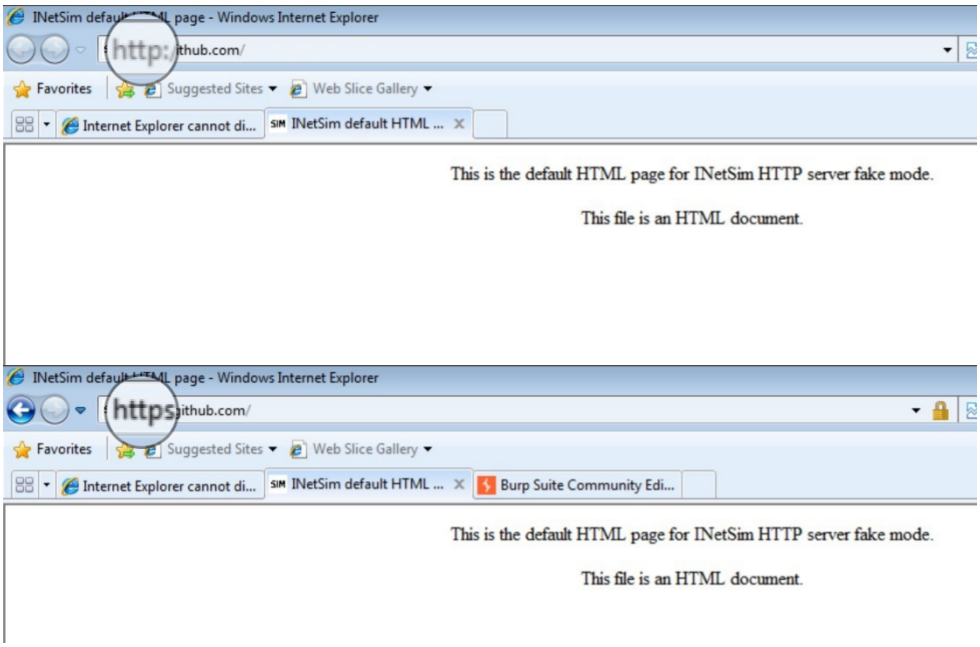

Figura 7.3: Respuesta INetSIM a tráfico HTTP y HTTPS.

7.2. Modelo de defensa *(CRISP-DM)*

En la fase de preparación de los datos el algoritmo de extracción de características recorre las distintas capturas de tráfico y nos deja solamente la columna deseada, que es la de paquetes UDP, y recorta este arreglo de bytes hasta su posición 1024, que es por donde se logra evidenciar el comportamiento que permite diferenciar entre un una aplicación benigna y un *malware*, tal y como se menciona en [3], estas capturas de tráfico ahora recortadas, son luego exportadas a un formato .csv que luego son cargados para el entrenamiento del modelo.

Antes del entrenamiento nos aseguramos que las columnas se encuentren en su tipo de dato correcto aplicando una transformación *eval* de la mano de la librería *swifter* para agilizar el proceso. Una vez transformados los datos, estos se exportan en un formato .pickle de forma que conserven su estructura y tipo, para que luego puedan ser cargados sin necesidad de aplicarles las transformaciones de nuevo.

Después de realizados estos pasos, recortamos los datos benignos a su 80 % y los datos de *malware* a su 35 %, esto lo hicimos primero, por las capacidades limitadas de hardware con las que contábamos, y segundo, para mitigar la alta tasa de falsos positivos, que hacía que capturas que eran benignas fueran clasificadas como *malware*.

En la fase de modelado se entrenaron y evaluaron 3 modelos de *ML* y *DL*, entre los cuales se obtuvieron los mejores resultados con el *Random Forest*, logrando un *accuracy* de 0.84 para la clasificación binaria. Para este se realizaron algunos ajustes de

hiperparámetros, quedando con los siguientes: *n_estimators*=200, *max_depth*=50 y *random_state*=42, de forma que los resultados sean replicables.

Del modelo de defensa en cuestión, también se obtuvo su matriz de confusión, ya que esta proporciona una visión general de la precisión y el rendimiento del modelo, permitiendo evaluar su nivel de acierto y error en la predicción de las clases.

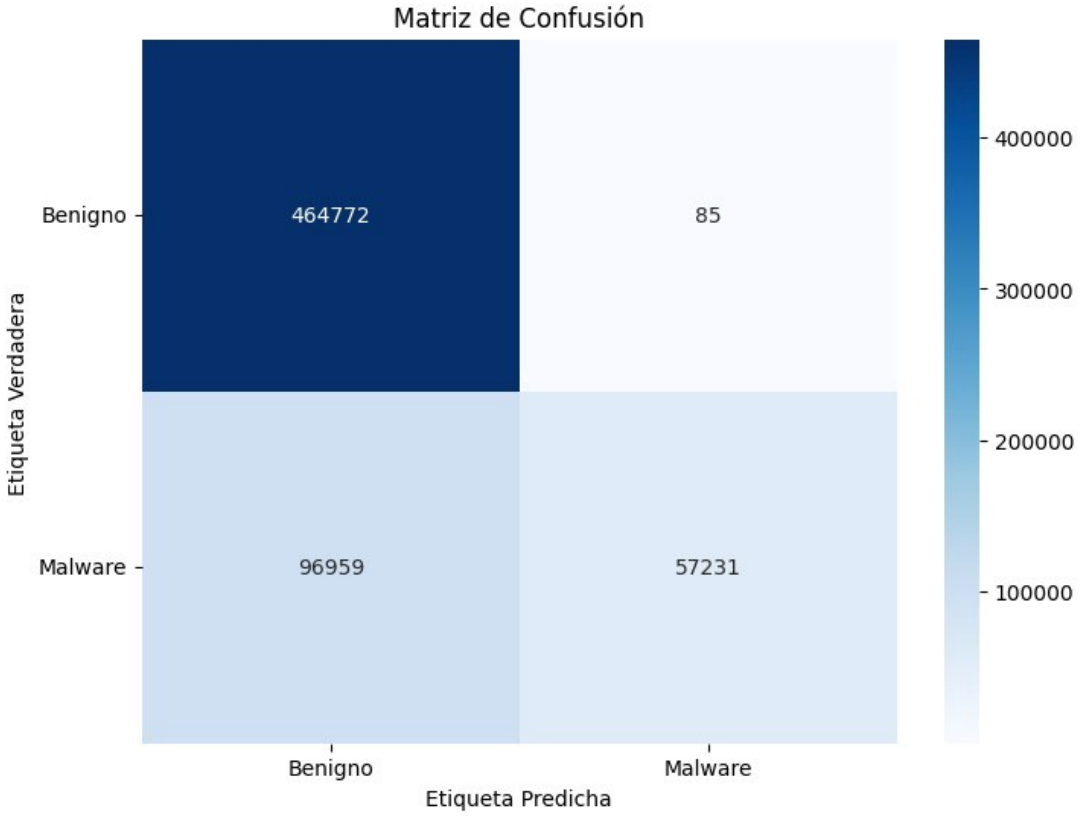

Figura 7.4: Matríz de confusión del modelo de defensa

En la figura 7.4 podemos observar que el modelo presenta una alta tasa de Falsos Negativos, lo que probablemente se deba a la cantidad de muestras de *malware* con las que se entrenó en comparación con las muestras benignas, por otro lado, vemos que casi no presenta problemas para identificar el tráfico benigno. Cabe resaltar que se hicieron extensivas pruebas con los distintos algoritmos, de las que resultaron alrededor de 30 modelos de defensa, y se escogió este por ser el que mejor interacción lograba con el algoritmo de *DRL* de ataque.

7.3. *Secure Learning* y *Deep Reinforcement Learning*

7.3.1. Ataque al modelo de defensa

Se adaptó el modelo de ataque del proyecto [2], para trabajar con los diferentes modelos de defensa creados. En donde adaptamos el algoritmo de *Deep Q-Network* para tener una entrada de tamaño 1024, y una salida de tamaño 1024 o 2048, dependiendo del modelo de defensa. Con una salida de 1024 significa que la red neuronal puede realizar 1024 acciones sobre el estado siguiente, a partir del anterior estado, por otro lado, con 2048, se duplica la cantidad de acciones. Las acciones permiten aumentar o disminuir cada una de las 1024 variables del paquete UDP independientemente entre 10 y 25 unidades.

Los hiperparámetros utilizados fueron un *learning rate* de 0.0001, un gamma de 0.9 y un epsilon de 1, adicionalmente el entrenamiento se dejó a 1000 épocas, en donde cada una se realizan modificaciones con el fin de clasificar erróneamente al *malware*.

Se trabajó con dos variantes, una en donde el 100 % del estado inicial es *malware*, y una en donde solo 10 de los 1024 parámetros son *malware* y el resto es una media de datos benignos. Esto se adaptó de tal forma que cuando es el primer caso modifica cualquiera de las 1024 variables, y en el segundo solo modifica las 10 variables no conocidas.

En el entrenamiento del modelo de ataque se usaron los datos generados en el laboratorio. Específicamente, se emplearon los *malware Trojan Asprox Old*, *Trojan Asprox New* y *AllElectroRAT*. Además, para el modelo de ataque con 1014 parámetros de datos benignos, se tomó la media del tráfico de las aplicaciones Facetime, Bittorrent, WOW y Weibo. Como discriminadores, se emplearon los modelos de defensa de *DeepMAL* y *Random Forest*. Cabe destacar que el mejor resultado se obtuvo con el segundo modelo como discriminador, lo cual dio lugar a la generación de aproximadamente 530 datos adversarios.

7.3.2. *Adversarial training*

Luego de realizar el *adversarial training*, se obtuvo que su *accuracy* fue de 0.8441, lo que representa una mejora muy pequeña respecto al modelo original (0.8371) y seguía reconociendo de casi igual manera tanto las muestras benignas como las de *malware*, lo que se evidencia en su matriz de confusión, en la figura 7.5.

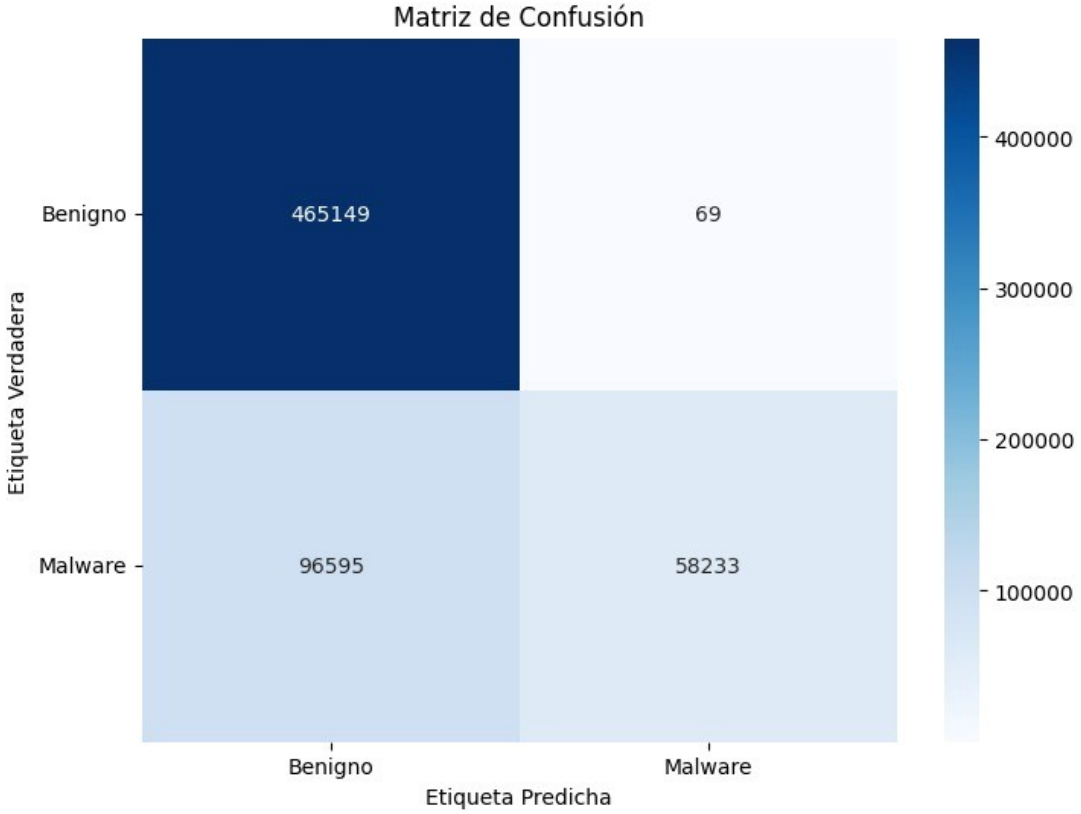

Figura 7.5: Matríz de confusión del modelo re-entrenado

De esta se puede observar que la diagonal de Falsos Positivos y Falsos Negativos redujo sus números, lo que demuestra una mejor clasificación en comparación con la matriz de confusión del modelo original, que la encontramos en la figura 7.4.

Luego de comprobar que el modelo seguía teniendo un comportamiento similar al original, se procedió a evaluar su desempeño con respecto al conjunto de evaluación de datos adversarios. Para esto se empleó la técnica de *Cross-Validation* con un k=5.

Tabla 7.1: *Cross-Validation* para la evaluación del modelo con datos adversarios

Cross-Validation	*Accuracy*
k=1	0.8445
k=2	0.8451
k=3	0.8451
k=4	0.8446
k=5	0.8444
$\bar{k}$	0.8448

Como se puede observar en la tabla 7.1, se obtuvo que el modelo pasó de reconocer el 0 % de los datos adversarios, a reconocer ahora un 85 % en promedio de estos. De

modo que podemos concluir que este es ahora un modelo mucho más robusto frente a ataques adversarios.

7.3.3. Enfoque de desarrollo seguro para modelos de *ML*

Durante el desarrollo del actual proyecto se interactuó con diversas áreas de ciberseguridad, en las que se trabajó con *malware* y modelos de ataque que generan datos adversarios, lo que nos lleva a proponer las siguientes medidas para el desarrollo seguro de proyectos de *Machine Learning*:

- Mantener privadas las entradas de los modelos, de forma que se dificulte la exploración y el descubrimiento de estas para los atacantes.

- Limitar el acceso a los datos de entrenamiento de los modelos, pues estos sirven de insumo para conocer las distribuciones y patrones en los datos, que luego sirven para engañar al modelo.

- Realizar *Adversarial Training* en la medida de lo posible, dado que este permite aumentar la robustez de los modelos frente a ataques adversarios.

- Tomar medidas que incrementen la dificultad de los intentos de reconocimiento de posibles atacantes. Algunas estrategias son (a) establecer límites en el número de consultas; (b) implementación de tiempos de espera; (c) evitar la entrega de información específica en los log de error; y por último, (d) cifrar partes sensibles de los datos para que solamente el modelo pueda interpretarlos correctamente.

8. Contribuciones y entregables

8.1. Contribuciones

La idea del presente proyecto nace de trabajos de grado previos [1, 2], en los cuales se propone a trabajo futuro mejorar el algoritmo *RL* para realizar perturbaciones más precisas utilizando *Deep Reinforcement Learning* y proponer un método de ataque adversario y posterior entrenamiento desde un enfoque *Black-Box*. Dentro de los aportes del proyecto se encuentran:

- Generación de muestras de *malware* mediante la creación de un laboratorio para la ejecución controlada de diferentes *malwares*.
- Creación de algoritmos *DRL* con enfoque de *Gray-Box*, conociendo 1014 parámetros de la entrada al modelo, y *Black-Box*, donde no se conocían los parámetros de entrada.
- Mejorar la robustez de un modelo clasificatorio a través de un algoritmo de *DRL*.
- Propuesta de enfoque de desarrollo seguro para proyectos de *ML* desde la experiencia.

8.2. Entregables

- Laboratorio para ejecución de *malware*.
- Conjuntos de datos utilizados para el modelo de defensa y ataque.
- Conjuntos de datos de *malware* generados en el ambiente seguro.
- Conjuntos de datos de muestras adversarias.
- Diferentes modelos de defensa.
- Diferentes modelos de ataque.
- *Jupyter Notebook* con el algoritmo de extracción de características.
- *Jupyter Notebook* con el algoritmo para obtener la media de datos benignos.
- *Jupyter Notebooks* con el entrenamiento del modelo de defensa.
- *Jupyter Notebooks* con el entrenamiento del modelo de ataque.
- *Jupyter Notebook* con el *Adversarial Training*.

9. Conclusiones y trabajo futuro

9.1. Conclusiones

En este proyecto se presenta una metodología que resulta como guía del proceso de desarrollo y entrenamiento de un modelo de defensa, clasificador de tráfico de red. El modelo es atacado utilizando un algoritmo de *Deep Reinforcement Learning* que busca engañarlo con muestras adversarias. Estas muestras son modificaciones de capturas de tráfico de *malware* obtenidas en un entorno de laboratorio seguro que permite la ejecución controlada de muestras de *malware*.

Una vez generadas las muestras adversarias, estas son usadas para reentrenar el modelo de defensa en un proceso conocido como *Adversarial Training*, el cual busca aumentar la robustez del modelo en cuestión, frente a ataques adversarios.

De lo anterior nos quedamos con las siguientes conclusiones:

- A través de la implementación de un laboratorio que proporciona un entorno seguro se logra emular el comportamiento en la red de diferentes muestras de *malware*, lo que permite mediante herramientas como BurpSuite, INetSIM y Wireshark capturar y analizar el tráfico de red generado. Lo cual brinda la oportunidad de entrenamiento del modelo con datos mucho más precisos y reales.
- El mejor modelo para la detección de *malware* con base únicamente en su campo de Paquetes UDP y las limitadas capacidades de hardware fue con el algoritmo *Random Forest*, que obtuvo un accuracy de 0.8371 para la clasificación binaria.
- Tras evaluar los modelos de defensa *DeepMAL* y *Random Forest*, se determinó que el modelo de defensa basado en el algoritmo *Random Forest* destacó como el mejor discriminador. Este modelo logró obtener aproximadamente 530 datos adversarios, lo que demuestra su capacidad para identificar de manera efectiva y clasificar correctamente las muestras adversarias generadas por el algoritmo de ataque basado en *Deep Reinforcement Learning*.
- Entre los diversos modelos de ataque evaluados, se encontró que el modelo que tuvo el mejor desempeño fue aquel en el que se utilizó como entrada el paquete UDP de *malware* desconocido, en contraste con el que desconocía solo 10 parámetros. Este enfoque demostró ser altamente efectivo para engañar al modelo de defensa, generando muestras adversarias que lograron evadir su detección y engañarlo exitosamente.
- El Adversarial Training ayuda a robustecer modelos de *ML* frente a ataques

adversarios, y en ocasiones incluso puede tener otros efectos positivos como un incremento en su rendimiento general de clasificación, que se ve reflejado en el *accuracy*.

9.2. Trabajo futuro

Teniendo en cuenta el alcance del proyecto y lo resultados obtenidos, se propone lo siguiente:

- Llevar a cabo nuevos entrenamientos para el modelo de defensa, posiblemente agregando columnas a la entrada original del modelo.
- Explorar la generación de datos adversos variando el tamaño de datos de *malware*.
- Continuar con el entrenamiento desde el enfoque de *Black-Box*.

10. Anexos

Tabla 10.1: Análisis de participación

Participante	Beneficiario directo	Beneficiario indirecto	Perjudicado	Neutral
Organizaciones y desarrolladores con buenas intenciones	X			
Sistemas de *ML*		X		
Usuarios de sistemas de *ML*		X		
Cibercriminal			X	

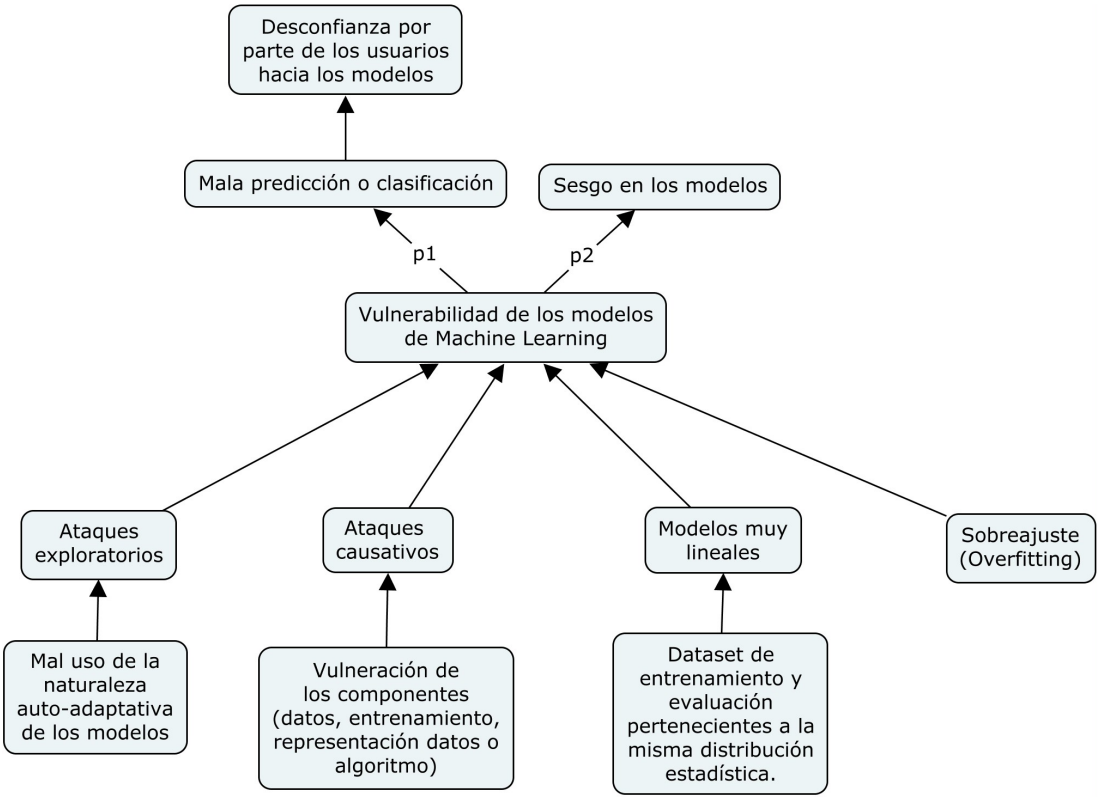

Figura 10.1: Árbol del problema

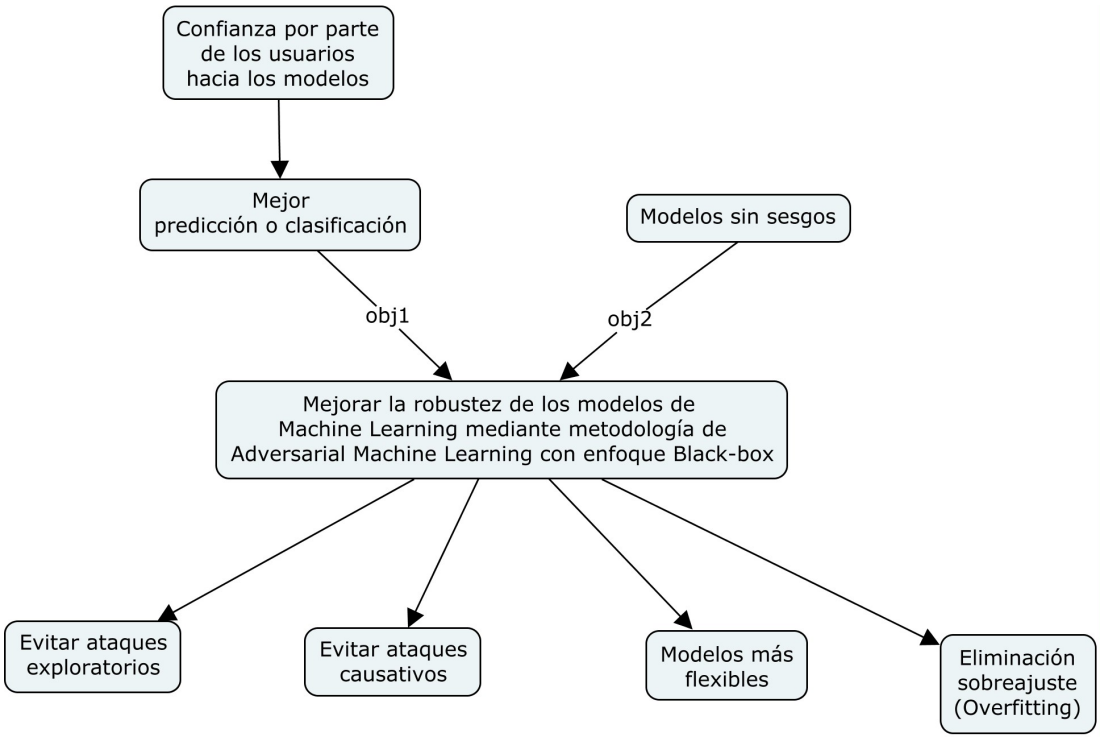

Figura 10.2: Árbol de objetivos

Referencias bibliográficas

[1] J. Delgado. *Secure Learning para detección de Android Malware*. Tesis de grado, Universidad ICESI, 2019.

[2] D. Huertas y B. Vargas. *Secure Learning y Deep Reinforcement Learning para la detección de Android Malware*. Tesis de grado, Universidad ICESI, 2021.

[3] G. Marín, P. Casas, y G. Capdehourat. DeepMAL – Deep Learning Models for Malware Traffic Detection and Classification. *CoRR*, abs/2003.04079, 2020. URL https://doi.org/10.48550/arXiv.2003.04079.

[4] I. H. Sarker, M. H. Furhad, y R. Nowrozy. AI-Driven Cybersecurity: An Overview, Security Intelligence Modeling and Research Directions. *SN Computer Science*, 2(3), 2021. doi: 10.1007/s42979-021-00557-0. URL https://doi.org/10.1007/s42979-021-00557-0.

[5] S. Zhang, X. Xie, y Y. Xu. A Brute-Force Black-Box Method to Attack Machine Learning-Based Systems in Cybersecurity. *IEEE Access*, 8:128250–128263, 2020. doi: 10.1109/ACCESS.2020.3008433. URL https://doi.org/10.1109/ACCESS.2020.3008433.

[6] C. C. Urcuqui, M. G. Peña, O. Q. J. Luis, y A. N. Cadavid. *Ciberseguridad: Un enfoque desde la Ciencia de Datos*. Universidad Icesi, 2018. ISBN 9789588936550. URL https://doi.org/10.18046/EUI/ee.4.2018.

[7] I. Goodfellow, P. McDaniel, y N. Papernot. Making machine learning robust against adversarial inputs. *Communications of the ACM*, 61(7):56–66, 2018. ISSN 0001-0782. doi: 10.1145/3134599. URL https://doi.org/10.1145/3134599.

[8] G. Rebala, A. Ravi, y S. Churiwala. Machine Learning Definition and Basics. In *An Introduction to Machine Learning*, pages 1–17. Springer International Publishing, 2019. ISBN 978-3-030-15729-6. doi: 10.1007/978-3-030-15729-6_1. URL https://doi.org/10.1007/978-3-030-15729-6_1.

[9] R. Sutton y A. Barto. *Reinforcement Learning*. Adaptive Computation and Machine Learning series. The MIT Press, 2 edition, 2018. ISBN 9780262039246.

[10] Y. Li. Deep Reinforcement Learning: An Overview. *arXiv.org*, 2017. doi: 10.48550/ARXIV.1701.07274. URL https://doi.org/10.48550/arXiv.1701.07274.

[11] J. Kulesza. Cybersecurity. In *Encyclopedia of Big Data*, pages 1–5. Springer International Publishing, 2018. ISBN 978-3-319-32001-4. doi: 10.1007/978-3-319-32001-4_53-1. URL https://doi.org/10.1007/978-3-319-32001-4_53-1.

[12] D. Dasgupta, Z. Akhtar, y S. Sen. Machine learning in cybersecurity: a comprehensive survey. *The Journal of Defense Modeling and Simulation*, 19(1): 57–106, 2022. doi: 10.1177/1548512920951275. URL https://doi.org/10.1177/1548512920951275.

[13] G. Li, P. Zhu, J. Li, Z. Yang, N. Cao, y Z. Chen. Security Matters: A Survey on Adversarial Machine Learning. *arXiv.org*, 2018. doi: 10.48550/ARXIV.1810.07339. URL https://doi.org/10.48550/arXiv.1810.07339.

[14] R. R. Wiyatno, A. Xu, O. Dia, y A. de Berker. Adversarial Examples in Modern Machine Learning: A Review. *arXiv.org*, 2019. doi: 10.48550/ARXIV.1911.05268. URL https://doi.org/10.48550/arXiv.1911.05268.

[15] J. Gui, Z. Sun, Y. Wen, D. Tao, y J. Ye. A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications. *IEEE Transactions on Knowledge and Data Engineering*, 35(4):3313–3332, 2023. ISSN 1041-4347. doi: 10.1109/TKDE.2021.3130191. URL https://doi.org/10.1109/TKDE.2021.3130191.

[16] T. Hussain y S. Singh. A Comparative Study of Software Testing Techniques Viz. White Box Testing Black Box Testing and Grey Box Testing. *International Journal of Allied Practice, Research and Review*, II:1–8, 2015. ISSN 2350-1294. URL http://www.ijaprr.com/download/1440480921.pdf.

[17] Y. Li, L. Li, L. Wang, T. Zhang, y B. Gong. NATTACK: Learning the Distributions of Adversarial Examples for an Improved Black-Box Attack on Deep Neural Networks. *arXiv.org*, 2019. doi: 10.48550/ARXIV.1905.00441. URL https://doi.org/10.48550/arXiv.1905.00441.

[18] IBM Documentation. Conceptos básicos de ayuda de CRISP-DM, 2021. URL https://www.ibm.com/docs/es/spss-modeler/saas?topic=dm-crisp-help-overview.

[19] D. Lu y W. Wang. USTC-TFC2016. https://github.com/yungshenglu/USTC-TFC2016, 2019.

[20] INetSim Documentation. INetSim: Internet Services Simulation Suite, 2023. URL https://www.kali.org/tools/inetsim/.