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

DAVID ALEJANDRO HUERTAS TRUJILLO

BRAYAN JOSE VARGAS PLAZA

Universidad Icesi
Facultad de Ingeniería
Ingeniería de Sistemas
Cali
2021

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

DAVID ALEJANDRO HUERTAS TRUJILLO
BRAYAN JOSE VARGAS PLAZA

Proyecto de grado

Dirigido por:
Christian Camilo Urcuqui Lopez, MSc.

Universidad Icesi
Facultad de Ingeniería
Ingeniería de Sistemas
Cali
2021

Tabla de contenido

Lista de acrónimos

AT	***A****dversarial* ***T****raining.*
SL	***S****ecure* ***L****earning.*
DRL	***D****eep* ***R****einforcement* ***L****earning*
APP	***App****lication*
RL	***R****einforcement* ***L****earning*
GAN	***G****enerative* ***A****dversarial* ***N****etwork*

Glosario de términos

Ataques adversarios*:* modificaciones intencionales que generan errores en un modelo de Machine Learning.

Over-fitting: cuando un modelo de Machine Learning se ajusta excesivamente a los datos y no es capaz de funcionar correctamente para datos externos a los de entrenamiento

Malware: software diseñado para causar daño, invadir, deshabilitar o robar información de un sistema.

Secure Learning: término empleado para referirse a un modelo de Machine Learning, tiene un buen desempeño (defensa) ante ataques adversarios que tienen como propósito engañar al modelo.

Generative Adversarial Network: es una clase de marco de aprendizaje automático diseñada por Ian Goodfellow y sus colegas en 2014. Básicamente se trata de dos redes neuronales compitiendo en un juego en el cual una busca discriminar la entrada de ciertos datos mientras que la otra trata de engañarla y mejorar su desempeño.

Reinforcement Learning: es un área de aprendizaje automático que se ocupa de cómo los agentes inteligentes deben realizar acciones en un entorno para maximizar la noción de recompensa acumulativa

Deep Reinforcement Learning: es un subcampo del aprendizaje automático que combina el aprendizaje por refuerzo (RL por sus siglas en inglés) y el aprendizaje profundo. RL considera el problema de un agente computacional que aprende a tomar decisiones por ensayo y error.

Índice de figuras

Índice de tablas

Motivación y antecedentes

Contexto

Para el año 2020 *Android* alcanzó una cuota en el mercado de sistemas operativos del 41 %, alzándose con el primer puesto de sistemas operativos con mayor número de usuarios activos [1]. Esta ventaja pone a *Android* en la mira de la industria, por lo que cada vez es más común encontrar todo tipo de aplicaciones en la tienda oficial (*Play Store*) de *Android,* que cuenta ya con más de 2.9 millones de *APPs* [2]. A este número se suman muchas más aplicaciones que no están disponibles directamente desde la *Play Store*, pero que se pueden encontrar navegando en Internet. En este amplio mercado también ponen la mira los cibercriminales, que con variadas técnicas tales como la generación de aplicaciones maliciosas (*Malware*), buscan obtener desde los datos de la actividad del usuario, hasta el control total del dispositivo [3].

A raíz de la creciente amenaza del *malware* se desarrollaron sistemas con modelos de *machine learning (ML)* que tienen como objetivo la identificación de software malicioso. Esto permite reducir ataques exitosos que logran infectar los dispositivos, sin embargo, se han descubierto técnicas capaces de engañar y vulnerar estos modelos. Por tal, es de gran importancia implementar medidas de seguridad en los sistemas de ML con el fin de mejorar su resistencia frente a este tipo de ataques.

Antecedentes del problema

El sesgo de los modelos de clasificación de *malware Android,* debido a los ataques adversarios, se presenta cuando estos son vulnerados alterando algún componente, ya sean los datos de entrenamiento, la representación de datos o el algoritmo de aprendizaje [4]. Esto ocasiona que el sistema presente una alta tasa de falsos positivos y falsos negativos. Lo cual finalmente puede repercutir en los usuarios finales, que al instalar una aplicación aparentemente segura, terminan instalando un *malware,* o instalan una aplicación que realmente es malware pero no es detectado como tal *o,* por el contrario, que una aplicación bien intencionada se catalogue injustamente como malware.

Un anterior avance sobre la solución a este problema se desarrolló mediante un enfoque de aprendizaje seguro (*Secure Learning*), en el cual se abordó el adversarial training desde un punto de vista en el que un cibercriminal tenía conocimiento sobre los datos de aprendizaje, la representación de las características y el algoritmo del modelo (*White Box*) [5]. A través del *secure learning (SL)* y utilizando *reinforcement learning* (RL) se disminuyeron las vulnerabilidades del modelo de clasificación.

También se propuso para trabajos futuros un enfoque basado en el desconocimiento de uno a dos componentes (*Gray Box*) o incluso una aproximación en la se desconozcan los tres componentes (*Black Box*), y a través de *Deep reinforcement learning* (DRL) y *Generative Adversarial Networks* (*GAN's)* lograr disminuir aún más las vulnerabilidades del modelo clasificatorio [6]. Lo anterior es una motivación porque permite, desde una perspectiva más cercana al cibercriminal común, enfocar el *secure learning* para proteger modelos en los que se desconocen parcial o totalmente los componentes del mismo.

Justificación

Es importante considerar este problema, pues desde un enfoque más realista, no es común que los atacantes tengan acceso a toda una lista de información esencial de un sistema, o en este caso, a un modelo de aprendizaje. Aunque con el anterior proyecto [5] se logró conseguir un modelo de clasificación mucho más robusto, esto no significa que sea suficiente o definitivo, ya que no contempla los escenarios en donde el atacante tiene como única alternativa realizar un ataque de *black-box* o *gray-box*. Así como expone [7], es vital que se considere la robustez además de la precisión del modelo para así garantizar un proceso de aprendizaje y clasificación mucho más confiable y seguro contra un rango más amplio de posibles ataques.

Descripción del problema

Actualmente existen vulnerabilidades a ataques con enfoque adversario en algoritmos de *machine learning* para la clasificación de *malware Android*. Las soluciones propuestas en cubiertas por trabajos anteriores se basan en entornos de *White Box*, por lo que el amplio margen, *Gray Box* y *Black Box,* quedan sin cobertura, resultando en un riesgo de seguridad para la industria y los usuarios de los dispositivos móviles con Android.

Objetivos

Objetivo general

Reducir vulnerabilidades a ataques con enfoque adversario en algoritmos de *machine learning* para la clasificación de *malware android* utilizando *Secure Learning* y *Reinforcement Learning*.

Objetivos específicos

- Proponer un modelo de ataque adversario con enfoque *gray-box*.
- Evaluar un modelo de *deep reinforcement learning* para realizar perturbaciones más precisas en el proceso de aprendizaje del modelo.
- Proponer enfoques de desarrollo que cuenten con buenas medidas de seguridad.

Marco Teórico

En esta sección se abordan conceptos de gran importancia para el desarrollo del proyecto, en relación con el *machine learning* y la ciberseguridad.

Machine learning

El *machine learning* (ML) es una técnica que permite a los computadores aprender a través de datos, sin que sea necesario programarse específicamente para una solución en particular. En esta rama de la inteligencia artificial, el sistema mismo, a través de la generalización, puede adaptarse para encontrar soluciones en ambientes similares [8].

Es un mecanismo que emula el aprendizaje de los seres vivos, tal como algunas especies aprenden qué un determinado alimento es perjudicial, probando pequeñas cantidades de este, analizando la reacción que causa en sus organismos para posteriormente, en encuentros con alimentos nuevos que tengan sabores, olores o características similares, puedan predecir, hasta cierto punto, los efectos que pueda causar. Esto a partir de unas características que previamente fueron aprendidas y luego generalizadas, permitiendo la adaptación a través de la experiencia.

Aprendizaje supervisado

Se refiere al tipo de aprendizaje en ML el cual requiere intervención, lo qué se consigue con unos datos de entrada debidamente etiquetados, indicando al modelo qué es lo que debe aprender específicamente. Existen dos tipos de técnicas bastante conocidas dentro de esta modalidad:

- modelos de clasificación
- modelos de regresión

Aprendizaje no supervisado

En aprendizaje no supervisado la información se aprende sin utilizar necesariamente etiquetas, por lo cual no se pueden aplicar directamente a problemas qué requieran la clasificación de datos. Uno de los objetivos del aprendizaje no supervisado es la exploración, en esta el modelo aprende la estructura de los datos, las características qué no se tenían previamente en cuenta y agrupa la información según su similitud [22].

Modelos de este tipo pueden ser utilizados para la toma de decisiones en las qué no se cuentan con datos categorizados, o se necesita encontrar relaciones entre las variables de una base de datos qué aparentemente no tienen conexión.

Reinforcement learning

El *reinforcement learning* (RL) es un área del *machine learning*, que consiste en un mecanismo de prueba y error. En este, un agente interactúa con un ambiente o entorno, con el objetivo de obtener cierto estímulo. Para alcanzar esta meta el agente realiza un determinado número de acciones sobre el entorno, que le permiten obtener información a partir de la exploración. Así es posible entonces optimizar sus interacciones para obtener la mayor cantidad de estímulo posible [10]. "Esto era simplemente la idea de un sistema de aprendizaje que quiere algo y que adapta su comportamiento para maximizar una señal especial de su entorno." [11]. Los elementos esenciales para componer un modelo de *reinforcement learning* son los siguientes:

- **El agente**: es el que realiza las acciones sobre el entorno modificando los estados en la búsqueda del objetivo.
- **Las acciones**: son aquellas interacciones que realiza el agente sobre el modelo del entorno.
- **Una política**: ésta define qué acciones puede realizar el agente en un estado determinado del modelo.
- **La señal de recompensa**: evalúa y entrega un determinado resultado al agente lo que le permite, basándose en esta recompensa, seleccionar acciones más adecuadas para mejorar el desempeño.

- **El Modelo de entorno**: es una representación del comportamiento real de un entorno que en determinadas circunstancias o acciones, tiene ciertos cambios o estados.

Deep learning

El *deep learning* (DL) es una técnica de *machine learning* en la que se utilizan más de 2 capas de redes neuronales artificiales. Estas redes se basan en algoritmos que están inspirados en el funcionamiento del cerebro. En las interacciones de las neuronas que en conjunto y a través de diferentes partes de este, permiten que aprendamos y generalicemos a través de datos qué se reciben de los órganos sensoriales.

En el DL resulta fundamental la interacción entre las capas, de esta forma en el caso de un sistema que permita reconocer ratones en imágenes, el primer nivel aprende algo sencillo, como los tonos del fondo. Luego pasa a un siguiente nivel en el que se determinan detalles cada vez más complejos que en conjunto terminan identificando un ratón en una imagen proporcionada [9].

Deep reinforcement learning

El RL es muy útil para la resolución de tareas en las que se requiere ejecutar acciones, es decir controlar el comportamiento de un agente en un modelo de entorno. El DL es ideal para el tratamiento y procesamiento de datos complejos. De esta forma, entonces es una buena elección utilizar el RL como un *framework* que implementa algoritmos de DL para manejar los datos de una forma muy eficiente [10].

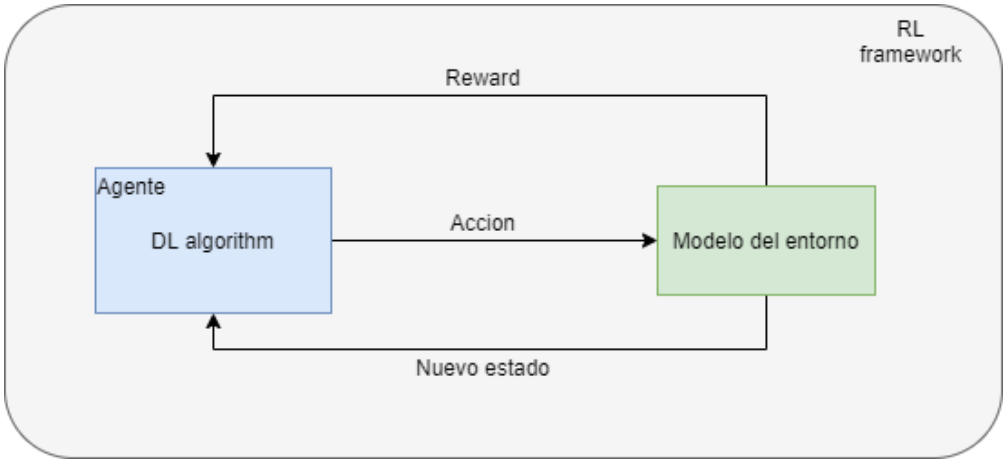


Fig. 1. Estructura de un sistema de *Deep Reinforcement Learning*, en este se muestra la interacción del agente con el modelo de entorno.

Red generativa adversaria (GAN)

Una red generativa adversaria es una estructura en la que interactúan dos modelos *deep learning* para mutuamente evaluarse y mejorar. Funciona asignando roles; discriminador y generador. A partir de esto el generador es evaluado por el discriminador, que a través de las respuestas y las interacciones permite construir un mecanismo de mejora [23]. Las *GANs* son comúnmente usadas para generar datos nuevos de una forma precisa. Estos datos se encuentran en el llamado espacio latente, el cual es el espacio de diferencias entre dos muestras conocidas de algún dato. Por ejemplo 2 rostros, en donde el espacio latente es una cantidad inmensa de muestras qué comparten características de los 2 rostros originales.

Inicialmente en el proceso de entrenamiento de una GAN, los datos del modelo generador son aleatorios. A medida qué este recibe el feedback del discriminador, los datos cada vez se parecen más a los datos reales y la tarea del discriminador se dificulta, lo cual a través de varias interacciones le permite mejorar.

Ciberseguridad

La ciberseguridad incluye un conjunto de prácticas que buscan robustecer y proteger los sistemas de información. Es vital en la era de la informática proteger los servicios y recursos que los medios electrónicos proveen, así como mantener la privacidad de la comunicación e información privilegiada. También es importante en la ciberseguridad que todos estos datos, comunicaciones y servicios se mantengan inalterados por terceros sin autorización [12].

Para alcanzar estos objetivos de mantener la confidencialidad, la integridad de los datos y la disponibilidad, se han desarrollado variedades muy amplias de técnicas como la criptografía y el *secure learning*. Este último enfocado en la seguridad de los sistemas de *machine learning*.

Cyber kill chain

Es un término qué tiene por origen el concepto militar de *kill chain,* qué tiene como objetivo definir los pasos que un enemigo sigue para atacar un objetivo. En el contexto actual recibe el nombre de *cyber kill chain* e incluye una serie de pasos qué comúnmente recorren los atacantes cuando intentan vulnerar un sistema [21].

- Reconocimiento: el atacante obtiene información del objetivo y de su entorno.
- Construcción de la amenaza: el atacante escoge el medio para llevar la amenaza a la víctima y la crea, esta puede ser, por ejemplo, un *malware*.
- Entrega: se hace entrega o despliegue de la amenaza en el blanco.
- Explotación: se aprovechan de vulnerabilidades para correr la amenaza en el objetivo
- Instalación o finalización de la explotación: una vez la amenaza se ejecuta, se instala el *malware* o en el caso de otro tipo de ataques como los realizados en este contexto de ciencia de datos se localizan los datasets o el modelo a vulnerar.
- Comando y control: se refieren al nivel de control qué obtiene el atacante sobre el objetivo y las posibilidades de ejecutar más acciones.
- Acciones en objetivos: se culmina el ataque y el atacante ejecuta acciones sobre el objetivo.

Malware

Se conoce como *malware* al *software* o programa informático que tiene como objetivo causar daño a un usuario, un computador o una red [13]. Las categorías más comunes son las siguientes:

- *Backdoors*: es un código malicioso que es instalado en un sistema comprometido para que posteriormente un atacante pueda tener acceso de forma rápida y sencilla.
- *Botnets*: permite que un atacante controle remotamente un grupo de equipos infectados.
- *Downloaders*: código malicioso que tiene solo el fin de permitir la descarga automática de otros códigos maliciosos.
- *Spywares*: recolecta información de la víctima y la envía a un atacante.
- *Launcher*: usa técnicas no tradicionales para ejecutar otros *malware* o servicios que permitan la explotación de una vulnerabilidad.
- *Rootkits*: son difíciles de detectar y están diseñados para ocultar otros *malware*.
- *Scarewares*: son programas que informan la existencia de virus en el sistema y requieren un pago, que generalmente solo hace que se desinstale el *scareware* mismo.
- Gusanos o virus: estos infectan el equipo y se auto replican para dispersarse en la mayor cantidad de dispositivos posibles.

Pentesting

El *pentesting* o pruebas de penetración es una práctica que se realiza sobre un sistema con el objetivo de verificar las medidas de seguridad y confirmar su correcta aplicación. En caso de encontrar vulnerabilidades medir el alcance que pueden tener. Una vez se encuentran vulnerabilidades se aplican los parches correspondientes, las actualizaciones o las medidas complementarias que se requieran, esto con el objetivo de disminuir los riesgos y construir un sistema robusto, resistente al mayor número posible de ataques que existan en el momento [14].

Debido a que es una práctica que requiere estar al tanto de las últimas vulnerabilidades descubiertas, es necesario que se realice periódicamente para garantizar la actualización de los componentes del sistema.

Este tipo de test se realiza desde 3 puntos de vista, llamados caja blanca, caja gris y caja negra. Se refieren a la cantidad de conocimiento y alcance qué se tiene sobre un sistema. En caja blanca o caja de cristal se conoce todo, la arquitectura, las tecnologías, la base de datos y todos los demás factores involucrados en la construcción del sistema. En la caja gris se conocen solo algunos de los factores. Finalmente en la caja negra no se conoce ninguno, solo se puede interactuar con el sistema como un usuario final.

Secure learning

Los sistemas de ML no están exentos de los riesgos que están presentes en el mundo digital. Los atacantes se enfocan en la extracción de datos, en la posible copia del modelo, en el engaño a través de entradas modificadas utilizando *adversarial attacks* o en la corrupción del modelo. Esto último implica una penetración que causa el mal funcionamiento interno, de tal forma que de entradas "limpias" se generan salidas incorrectas [4]. Los tipos de ataques a los modelos de ML según un informe de Microsoft y MITRE [24] se pueden agrupar en 7 categorías:

- Reconocimiento: en esta se busca encontrar información acerca del modelo y su entorno, tecnologías empleadas y versiones.
- Acceso inicial: se ubican aquí ataques que buscan obtener un acceso no autorizado al uso del modelo.
- De ejecución: aquellos ataques que buscan ejecutar los modelos en ambientes no seguros con el fin de comprometerlos.

- Persistencia: implantación de backdoors y envenenamiento de datos.
- Evasión del modelo: en esta categoría se agrupan diversas técnicas tales como los ataques adversarios, pieza fundamental en este proyecto. Estos ataques están destinados a causar fallas u obtener información qué permita evadir los modelos de ML de clasificación.
- Filtración de información: cuando se tiene el fin de robar los datos de entrenamiento o incluso clonar el modelo.
- De impacto: tiene el objetivo de alterar las configuraciones, denegar el servicio y en general de paralizar el correcto funcionamiento del modelo de ML.

Por todo lo anterior es importante robustecer el modelo, de esta forma se reducen las posibilidades de un ataque exitoso, al disminuir las vulnerabilidades. Este proceso de robustecimiento se puede alcanzar, en el caso del engaño del modelo, a través de *adversarial training*, en el cual se entrena otro modelo de machine *learning* para que genere entradas modificadas específicamente para engañar el modelo [15]. Con estos resultados posteriormente se realiza el entrenamiento mezclando los datos anteriores, más estos nuevos datos adversos, dando como resultado un modelo más robusto.

Estado del arte

Robust Android Malware Detection System Against Adversarial Attacks Using Q-Learning

En [16] se señala principalmente que a pesar de que los modelos de ML y DL, en los que se basan los diferentes sistemas de detección de *malware* en *Android,* brindan un gran desempeño, estos suelen ser muy vulnerables a ataques de tipo adversario.

Primeramente, se desarrollan ocho modelos de detección de *malware* en Android basados en ML Y DNN para así poner a prueba su robustez contra ataques de tipo adversario.

Entre los ataques que se han propuesto dentro de la investigación se encuentra uno de tipo *white-box.* Se trata de un ataque de política única que busca la modificación del vector de características extraído de una app maliciosa de tal forma que llegue a ser identificada como benigna por el clasificador de tipo *gray-box*. Un ataque de política múltiple que consiste en la recolección de un conjunto de políticas a partir de Q-tablas para luego usarlas paralelamente en ataques de adversario.

Además, se trata de la primera investigación de ataques de tipo adversario con un enfoque *gray-box* en el área de la detección de *malware*. Finalmente, se propone una estrategia defensiva contra este tipo de ataques la cual reduce significativamente la tasa de engaños promedio contra ataques de política única y múltiple. Así se mejora la robustez general del Sistema de detección de *malware Android*.

Deep Reinforcement Learning for Black-Box Testing of Android Apps

En [17] se expone que uno de los grandes retos en el desarrollo de aplicaciones móviles, debido a su creciente complejidad, es el diseño e implementación de pruebas del funcionamiento de estas. Señalan que la implementación de una efectiva fase de pruebas es de suma importancia para minimizar la aparición de fallas durante la ejecución. Sin embargo, los distintos procedimientos y exploraciones llevadas a

cabo por pruebas automáticas no llegan a ser suficientes como para cubrir la extensa cantidad y complejidad de los posibles estados de la aplicación.

La aproximación que se propone es una ejecución de pruebas con enfoque *black-box* basado en *deep RL* llamada ARES. Los resultados mostraron que el alcance y revelación de fallas es mayor en comparación con procedimientos convencionales de pruebas automáticas que se basan en una exploración aleatoria, que, si bien cubre gran parte del código y detección de bugs, se puede quedar corta al tratar con transiciones mucho más complejas.

Evading Anti-malware Engines with Deep Reinforcement Learning

Este proyecto [18] se realizó con el objetivo de demostrar que los sistemas de clasificación de Android malware basados en ML, de aprendizaje supervisado, son vulnerables, específicamente a técnicas adversarias. En este tipo de técnicas se entrena un modelo de ML. En este caso específico se utilizó uno de DRL y se nombró DQEAF. Este se encargó de interactuar con los datos de entrenamiento y a través de estos generó "*malware*" que pasó por alto el sistema de clasificación. Estos datos también fueron testeados en Virustotal, evidenciando una mejora en la habilidad para evadir antivirus.

Aunque este proyecto demostró exitosamente qué en la actualidad existe este tipo de vulnerabilidades en muchos sistemas de *machine learning*, su alcance no partió de un enfoque de caja negra o gris, sino que se le permitió al DQEAF conocer a profundidad el algoritmo de clasificación. Es decir, se basó en un enfoque de caja blanca. Cabe recalcar que el fin de este proyecto fue la demostración de la vulnerabilidad, así que no se realizaron aplicaciones de *secure learning* con el objetivo de robustecer el modelo.

Adversarial-Example Attacks Toward Android Malware Detection System

En [19] se menciona que así como los sistemas existentes de detección de *malware* en *Android* pueden ser burlados por ataques de tipo adversario basados en redes adversarias generativas (GAN), también pueden ser fácilmente defendidas con

la implementación de un cortafuego a un detector de *Malware Android* conectado remotamente en la nube.

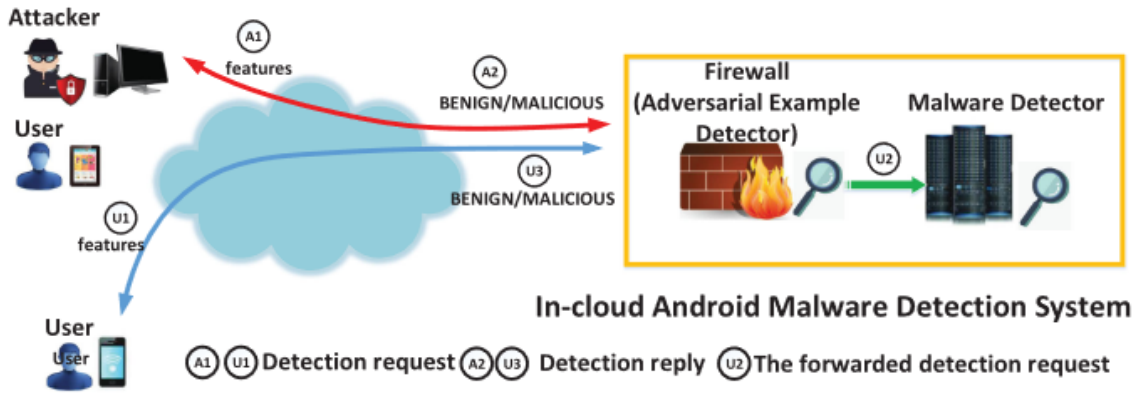


Fig. 2. Resumen del sistema [19]

En el trabajo mencionado, se propone el desarrollo de un nuevo método de ataque de ejemplo de adversario, llamado E-MalGAN, el cual básicamente intenta enviar una entrada maliciosa con la capacidad de eludir las mencionadas formas de detección de *malware*. Esto a partir de su propuesta de variante de *GANs* llamada *bi-objective GAN,* la cual es capaz de generar ejemplos de adversario con el objetivo de engañar tanto al *firewall* como al detector de *malware.* Los resultados muestran que el 95% de los ejemplos de adversario producidos por E-MalGAN evadieron exitosamente los dos detectores trabajados en esta investigación.

Black box analysis of android malware detectors

En [20] exponen que, en vista del rápido aumento de Android en la captura del mercado, era de esperarse un gran aumento en cuanto a ataques a su sistema por medio de *Malware*. Se menciona que uno de los retos para crear formas de protección ante este tipo de ataques radica en las técnicas de ofuscación de código.

Esto permite camuflar las verdaderas intenciones de un código malicioso sin que se vea afectado el rendimiento o funcionalidad de este.

En el trabajo mencionado previamente, se propone un análisis con enfoque *black-box*, a la aplicación de ofuscación de código, a determinadas y conocidas características de Malware Android y a su efecto al enfrentarse con diferentes detectores de *Malware Android*. Esto para luego aplicar ingeniería inversa en estos algoritmos y así determinar en qué característica se basa un detector según sus capacidades.

Resumen del estado del arte

En la siguiente tabla (Tabla 1) se resumen las características principales relacionadas a nuestro proyecto versus a los documentos seleccionados en el estado del arte.

Características. /papers	[16]	[17]	[18]	[19]	[20]	Este proyecto
DRL		x	x			x
Gray-Box	x		x	x		x
Secure Learning	x					x
Malware Classifier	x		x	x	x	x

Tabla 1: Matriz resumen del estado del arte

En este proyecto se propone un esquema de secure learning aplicado a un modelo clasificador de malware android en un ambiente de caja gris y negra para el cual se utiliza aprendizaje profundo reforzado.

Metodología

Para este proyecto se llevará a cabo bajo la metodología *Cross-industry standard process for data mining* (CRISP-DM) [20], debido a que se ajusta al manejo de los datos de este proyecto y al desarrollo y entrenamiento de modelos de *machine learning*. Posteriormente, se incluyen un conjunto de actividades de *secure learning* para mejorar la robustez del modelo construido en la primera aproximación.

CRISP-DM

1. Entendimiento del negocio

A través del estudio de trabajos anteriores, que han sido los fundamentos de este proyecto, fue posible entender el negocio, en este contexto, comprender la importancia de la ciberseguridad y la manera en que se aplica aquí. Fue de gran utilidad en este apartado, la revisión de variedad de artículos y libros de esta temática qué hicieron posible culminar esta fase.

2. Entendimiento de los datos

Con base en el dataset escalado y los datos adversarios resultantes del proyecto anterior [5] fue posible analizar los valores, para comprender cómo estaban estructurados a través de una descripción de los datos. En el primer dataset original se incluyen 7832 muestras del comportamiento en red de aplicaciones android de las cuales 3141 tienen la etiqueta de *"malicious"* y el resto *"benign"*.

El segundo dataset contiene los datos adversarios generados, de gran importancia para este proyecto porque se deben incluir para generar la mayor robustez posible en el modelo final. Ambos *datasets* cuentan con 11 variables. 10 de estas correspondientes al comportamiento de una aplicación *android* en red, la

variable 11 es la etiqueta dada, la cual informa si la muestra es *malware* o si por el contrario no lo es. Las variables que componen el *dataset* son:

- Bytes enviados por la aplicación.
- Bytes recibidos por la aplicación.
- Paquetes tcp.
- Paquetes recibidos por la aplicación.
- Paquetes enviados por la aplicación.
- Volumen de bytes.
- Número de consultas al DNS.
- Paquetes distintos TCP.
- Paquetes UDP.
- IP externas.
- Tipo maligno o benigno.

Cabe mencionar que hubo 2 datasets de tipo csv relativamente recientes (2020) que se lograron encontrar en la web. No obstante, uno de ellos solo era una recopilación de datos tomados en 2018, los cuales ya se tenían. El otro, aunque sí contiene datos actualizados, no contaba con las variables que el clasificador en cuestión analiza para determinar si una muestra es un malware o no. Es importante que las variables fueran las mismas ya que con el fin de robustecer y mejorar el clasificador android, se necesita una coherencia en los datos ya estudiados y los nuevos. Por tanto, se descartaron dichos datasets y se procedió con la preparación de los inicialmente mencionados.

3. Preparación de los datos

Durante esta fase se confirmó la integridad de los datos, verificando la no inclusión de valores no permitidos según su columna. Una vez completada la evaluación de calidad, se realizó la mezcla de los dos *datasets* anteriores y se verificó nuevamente su integridad.

4. Modelado

Se entrenaron nuevamente 6 modelos *machine learning* (Tabla 2) con el objetivo de actualizar a versiones más recientes y verificar como la mejor opción como clasificador al modelo *Random forest*. Este fue usado posteriormente para desarrollar el modelo de *Deep Reinforcement Learning* basado en *Deep Q Learning* para generar los datos adversarios.

Algoritmo	Precisión		Recall		F1-score		Accuracy
	benign	malign	benign	malign	benign	malign	
Naive Bayes	0.81	0.41	0.12	0.96	0.20	0.58	0.4468
Random Forest	0.93	0.90	0.94	0.88	0.93	0.89	0.9172
KNN: K=4	0.89	0.89	0.93	0.83	0.91	0.86	0.8922
SVM	0.62	0.90	1	0.06	0.76	0.11	0.6271
Logistic regression	0.72	0.68	0.86	0.47	0.78	0.56	0.7063
Decision Tree	0.90	0.85	0.90	0.84	0.90	0.84	0.8773

Tabla 2: Modelos candidatos a clasificador actualizados.

5. Evaluación

La evaluación se basa en la precisión, el *F1-score,* el *recall* y la pérdida. Estas métricas se seleccionaron con el objetivo de verificar la disminución de los falsos negativos, por eso se toma especialmente en cuenta el *recall.* Resultados qué una vez graficados como en el caso del precisión qué en una matriz de confusión permiten verificar el correcto comportamiento del modelo a evaluar.

6. **Despliegue**

La fase de despliegue no se incluye en el alcance de este proyecto ya que se busca abarcar sólo las fases anteriores con el fin de generar un modelo más seguro y resistente a ciberataques justo antes del despliegue.

Secure learning

Ataque al modelo de defensa

Este ataque se realizó en un enfoque *Gray box*, desde la perspectiva de un supuesto atacante qué solo tiene acceso a consultar el modelo clasificatorio. Es decir, a entregarle una muestra de cierta cantidad de variables que desconoce y recibir una respuesta de si esta muestra entra en la categoría de malware o no. El atacante es conocedor del tipo de datos que recibe el modelo clasificatorio, qué en este caso son datos del comportamiento dinámico de una aplicación, más específicamente del comportamiento de la comunicación en red. Con base en la llamada *cyber kill chain [21]* el ataque se divide en varios pasos*:*

Exploración

1. El primer paso, fue realizar una exploración del modelo de defensa. Es decir partiendo del desconocimiento del número de variables que el modelo clasificatorio recibe y el tipo de dato de cada una de estas, se consultó de manera aleatoria e interactiva cada una de las variables.

Generación de muestras de malware

2. El segundo paso fue generar muestras de *malware* que luego iban a ser el insumo qué se iba a convertir en datos adversarios. Se inició con un recorrido aleatorio en busca de muestras que el modelo de defensa considerará como *malware*.

Generacion de datos adversarios

3. Posteriormente, se implementa el algoritmo de *deep Q-learning*, y luego se modifica convirtiéndolo en una red generativa adversaria, en la cual el discriminador es el modelo de defensa. Después de realizar numerosas épocas, finalmente, el resultado es un dataset que tiene el mismo número de muestras que el *dataset* de *malware* generado pero en el cual, todas las entradas han sufrido perturbaciones mínimas. De tal forma qué, ahora, el modelo clasificatorio a pesar de que contiene muestras muy similares a las de tipo *malware,* las detecta como si fueran de tipo benigno.

Entrenamiento adversario

Durante esta sección se preparó el *dataset* final, con datos qué incluyen los generados en el proyecto anterior [5], más los datos originales y a estos se les agregó los adversarios generados en este proyecto. De cada uno de los *dataset* se tomó 75% para entrenamiento y 25% para las pruebas, a razón de qué los datos se encontraban balanceados.

El paso siguiente fue el entrenamiento del modelo clasificatorio, para el cual se generó un modelo de *random forest* y se entrenó con el *dataset* final. Luego se evaluó utilizando la porción de pruebas de los datasets con el fin de comparar las métricas, para así determinar la robustez alcanzada.

Resultados

Reentrenamiento y actualización de los 6 modelos de defensa

Durante el proyecto [5] se entrenaron 6 modelos de aprendizaje supervisado con el objetivo de determinar el clasificador qué obtuviera mejores resultados. El modelo seleccionado fue finalmente el *random forest* qué obtuvo un *accuracy* de 0.9172 en ambas versiones. Sin embargo, en algunos de los modelos se obtuvieron resultados diferentes.

Los modelos qué variaron en los resultados al pasar de la versión de *scikit-learn* 0.20.1 a la versión 1.0.1 fueron; el árbol de decisión pasó de un *accuracy* de 0.8845 a 0.8799. El SVM disminuyó su *accuracy* de 0.7359 a 0.6271 mientras que la regresión logística aumentó su *accuracy* al pasar de 0.7053 a 0.7063.

Ataque al modelo de defensa

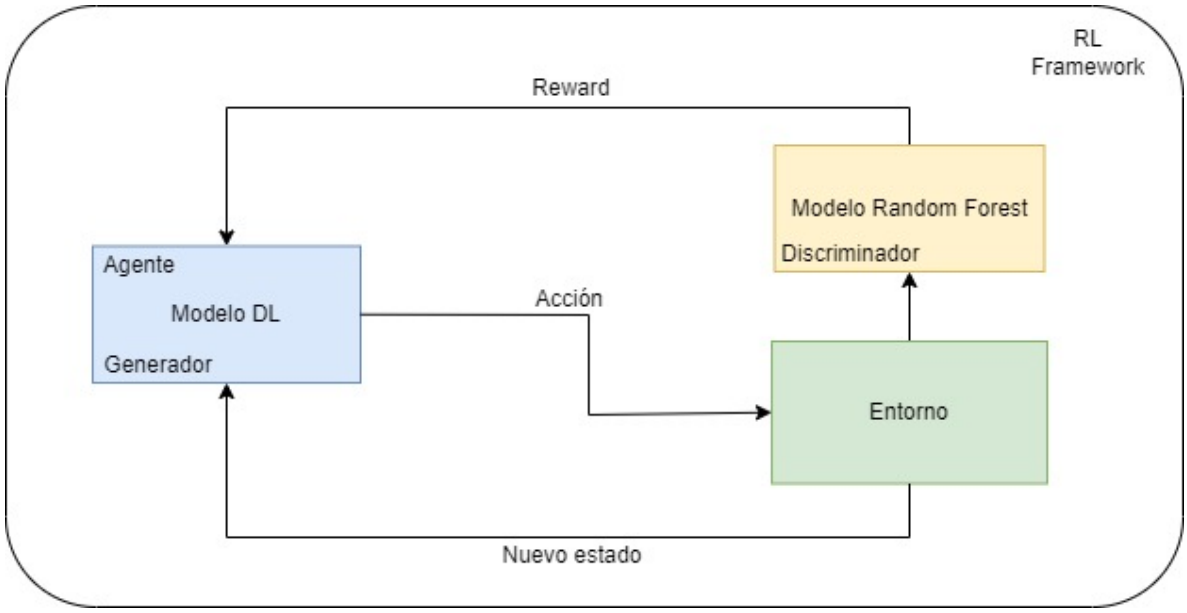


Fig. 3. Framework propuesto durante este proyecto para el ataque adversario.

Los resultados de la primera fase del ataque al modelo de defensa se basó en la exploración. De aquí se obtuvo qué eran 10 variables las qué recibía el modelo clasificatorio. Este paso se realizó a través de la captura de errores, es decir que después de correr un algoritmo de fuerza bruta y verificar las excepciones se obtuvo finalmente la combinación de 10 variables de carácter numérico. Como recomendación se propone minimizar la información que se entrega al momento de capturar los errores ya que en este caso aunque se ignoró, al entregarle al modelo un número errado de variables, se lanza un error qué informa el tipo y cantidad de variables qué requiere el modelo, facilitando significativamente la fase de reconocimiento del atacante.

El segundo paso fue la construcción del generador de *malware* basándose en el conocimiento obtenido en el primer paso. Para esto también se utilizó un algoritmo de fuerza bruta que de manera aleatoria génerо muestras de *malware*. Para cada una de las variables al desconocerse los rangos qué podrían tomar, se escogió como máximo 100 y como mínimo -100. La elección del rango se realizó teniendo en cuenta qué en este intervalo se podían hallar muestras de malware más rápidamente qué en otros rangos descartados como -10, 10 y -50, 50. A partir de esto se generó entonces un total de 1500 datos de malware. Este malware se guardó en un archivo csv para entregarlo en el siguiente paso.

A continuación se implementó el algoritmo de DQN el cual es una modificación del *framework* básico de RL utilizando *Deep Q-learning* en donde la parte profunda se realiza a través de un agente de red neuronal.

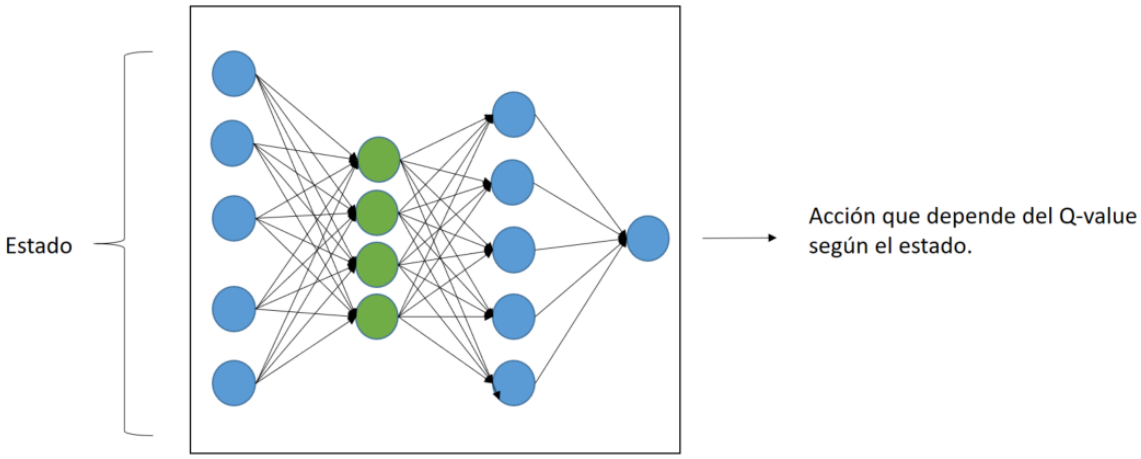


Fig. 4. Agente DQN

En este caso la salida tiene un tamaño de 20, lo cual significa qué existen 20 acciones que el agente puede realizar sobre el estado siguiente a partir del estado anterior. Las acciones son aumentar o disminuir cada una de las 10 variables independientemente en 10 unidades.

En cuanto a los hiperparametros se utilizó un learning rate de 0.001, un gamma de 0.9 y un epsilon de 1, con el fin de iniciar con una exploración totalmente aleatoria e ir realizando explotación a medida qué se actualizan los pesos de la red neuronal. El entrenamiento se ajustó a 1000 épocas en cada una de las cuales se realizaban N modificaciones con el fin de lograr finalmente el resultado esperado.

Gráficamente se pudo analizar qué tanto alteró finalmente el modelo DRL el *malware* generado, al compararlo con los datos adversarios, es decir, los datos que a pesar de ser inicialmente malware y tener mínimas modificaciones el modelo clasificatorio ahora detecta como muestras benignas (ver fig. 4). La media de los cambios realizados en cada variable fue de 5,96 unidades.

En la figura 4 se puede apreciar en azul la media de los datos qué el modelo consideraba como malware y en rojo los datos modificados qué ahora el modelo detecta como benignos.

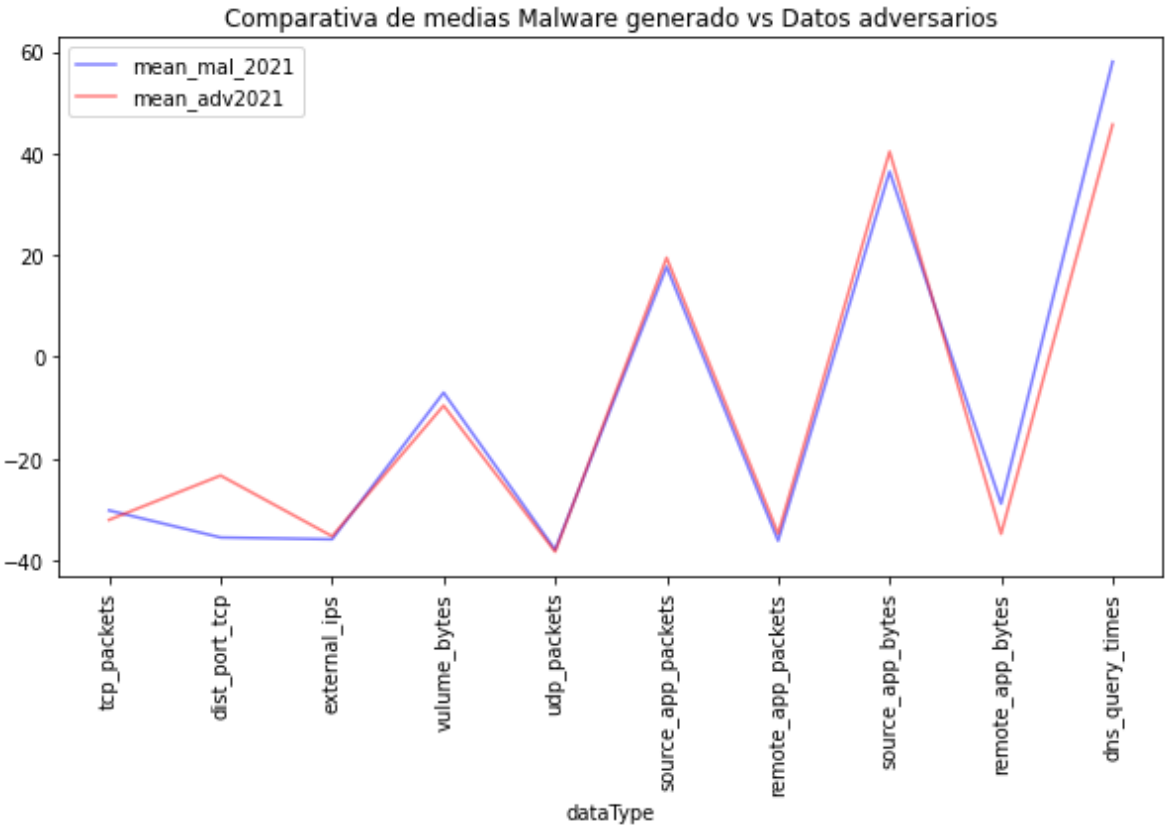


Fig. 5. Malware generado vs Datos adversarios

Con esta información es posible para un atacante determinar rangos para cada una de las variables, con el supuesto de que el atacante conoce que el discriminador realiza una evaluación a partir del comportamiento en red de la aplicación.

Entrenamiento adversario

El primer paso fue verificar la robustez del modelo clasificatorio anterior sobre los datos adversarios generados a partir de caja gris. Para esto se utilizó el total de los datos, que mezclan los valores originales, los datos adversarios del proyecto anterior y los generados en este proyecto.

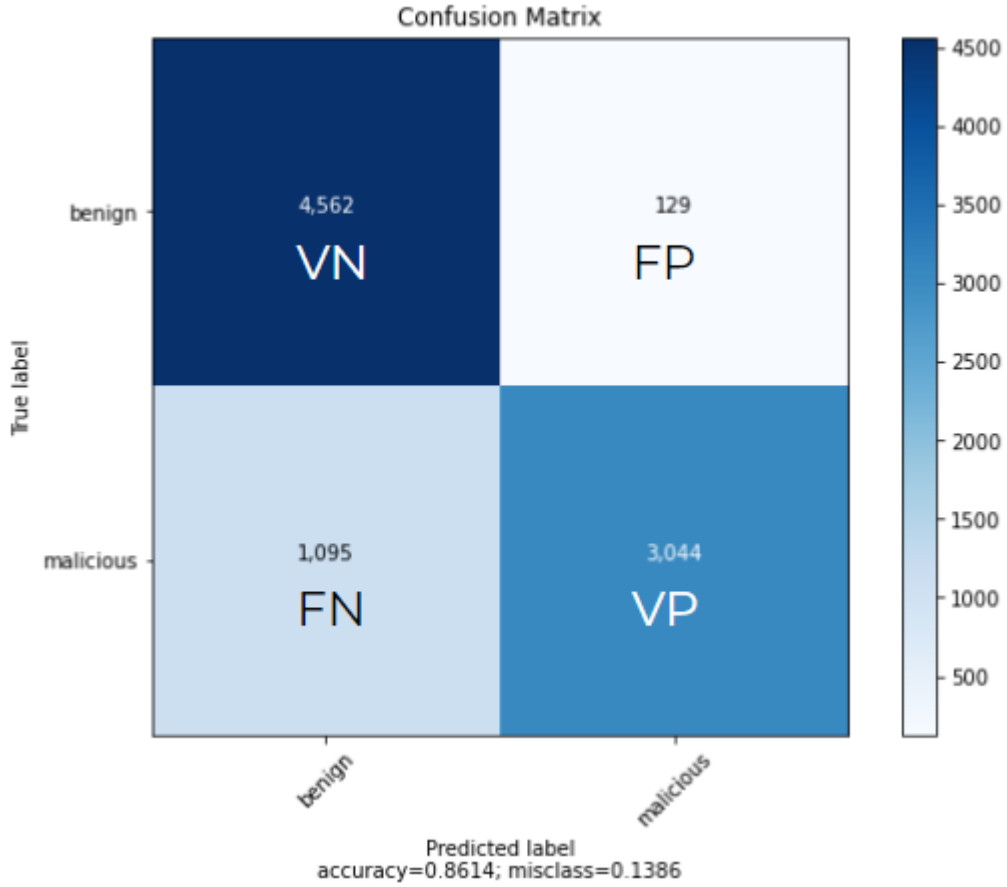


Fig. 6. Matriz de confusión del modelo vulnerable

En la figura 5 se puede apreciar a través del *accuracy* del 0.8614 el nivel de vulnerabilidad de este modelo clasificatorio qué falla en detectar 1095 muestras maliciosas esto en contraste con los datos de la figura 6 evidencian una mejora significativa, lo qué resulta en un modelo clasificatorio notablemente más robusto qué alcanza una *accuracy* de 0.973.

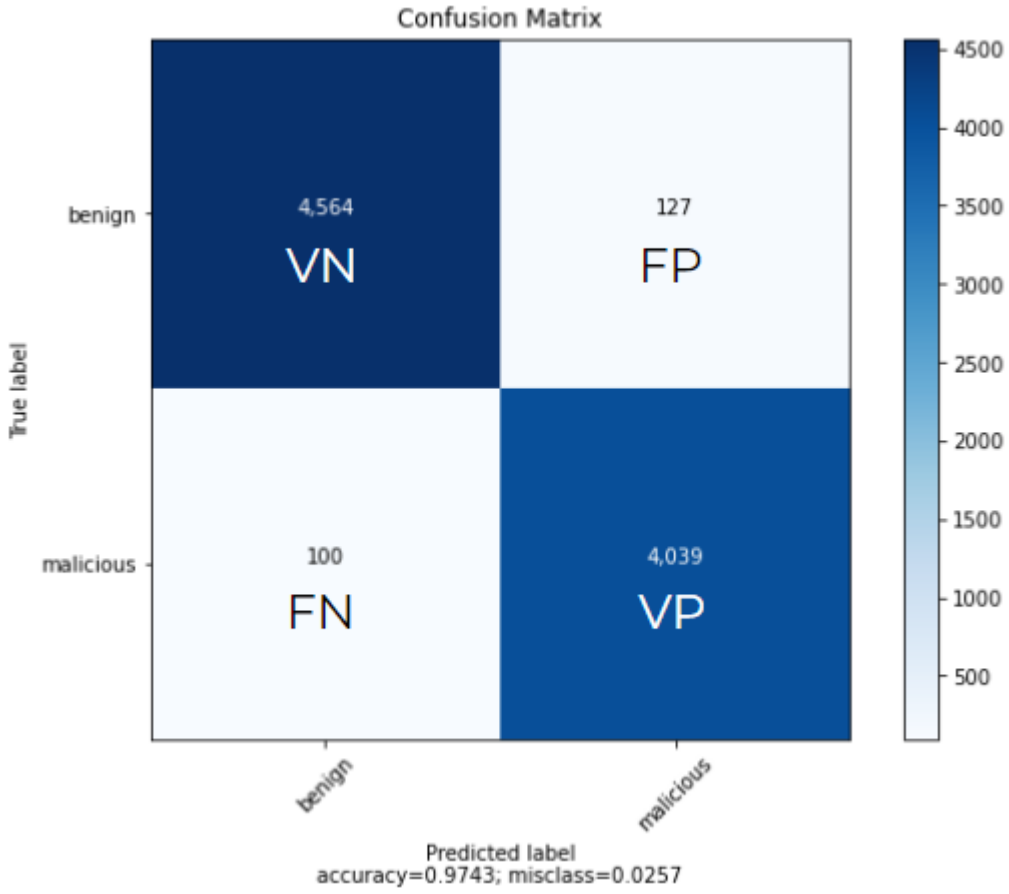


Fig. 7. Matriz de confusión del modelo robusto

Enfoque para el desarrollo seguro de modelos de ML

En primer lugar se recomienda realizar un modelo de base teniendo en cuenta los datos originales. Posteriormente es una buena medida mejorar la robustez mediante la inclusión de datos adversarios generados a través de RL utilizando o no los datos originales.

Una vez generados estos datos adversarios, mezclados y entrenado el modelo, se verifican las mejoras respecto al modelo inicial.

Finalmente al momento del despliegue, es de gran importancia dificultar los intentos de reconocimiento que los atacantes pueden intentar realizar. Para esto,

acciones como limitar el número de consultas, la adición de tiempos de espera y evitar dar información muy específica en el momento de capturar errores pueden hacer gran diferencia frente a ataques a modelos de ML.

Contribuciones y entregables

Contribuciones

Las principales contribuciones alcanzadas en este proyecto se dividen en las secciones de secure learning y la perspectiva de un atacante con fines ilustrativos para dificultar posibles ataques futuros.

Secure learning

- Aumento de la robustez del modelo clasificatorio a través de un algoritmo DRL.
- Proponer medidas qué mejoren la seguridad durante el desarrollo del modelo incluyendo la fase del despliegue.

Atacante

- Plantear un ataque adversario de tipo *gray-box* para el clasificador de *malware* android usado en el proyecto.
- Propuesta de escaneo de rangos en donde se hallan muestras de *malware Android* según el resultado del clasificador.

Entregables

- Dataset de datos adversarios.
- Dataset de *malware* generado.
- Generador de *malware* basado en el modelo de clasificación.
- Explorador de variables y tipos.
- Algoritmo de DQN y GAN para generar los datos adversarios.
- Modelo de defensa robusto.

Conclusiones y trabajo futuro

Conclusiones

En este proyecto se presentó un modelo, el cual a través de DRL logra mejorar significativamente la robustez del clasificador de malware. Inicialmente no se reconocieron 1095 muestras de datos adversarios, pero después del proceso de adversarial training tan solo 100 datos se mantuvieron mal clasificados. Por otro lado, respecto a la *accuracy* de los datos adversarios se encontró qué el algoritmo de DRL alcanzó una modificación mínima de 5,96628 respecto a los valores de entrada, lo qué permite observar qué los datos adversarios presentan una gran similitud con los datos de malware generados.

El uso de *Deep reinforcement learning* en combinación con una estructura de una red generativa adversaria es de gran utilidad al momento de realizar un proceso de entrenamiento adversario ya qué permite reducir los tiempos qué se tardarían en generar estos datos de otras formas. Los tiempos oscilan entre los 4 minutos 50 segundos y los 5 minutos 10 segundos mientras que, tomando como ejemplo la exploración aleatoria del modelo clasificatorio para obtener 500 muestras de malware, se toma alrededor de 30 minutos.

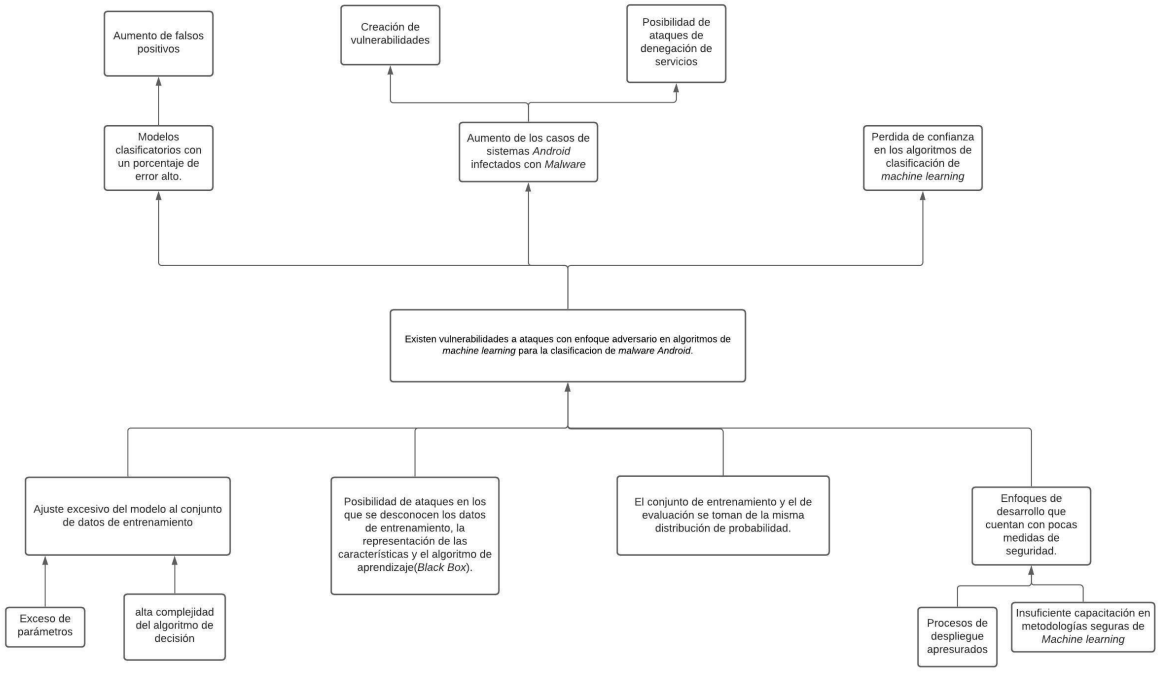
Trabajo futuro

Teniendo en cuenta el alcance de este proyecto y los objetivos conseguidos plantamos para un trabajo futuro:

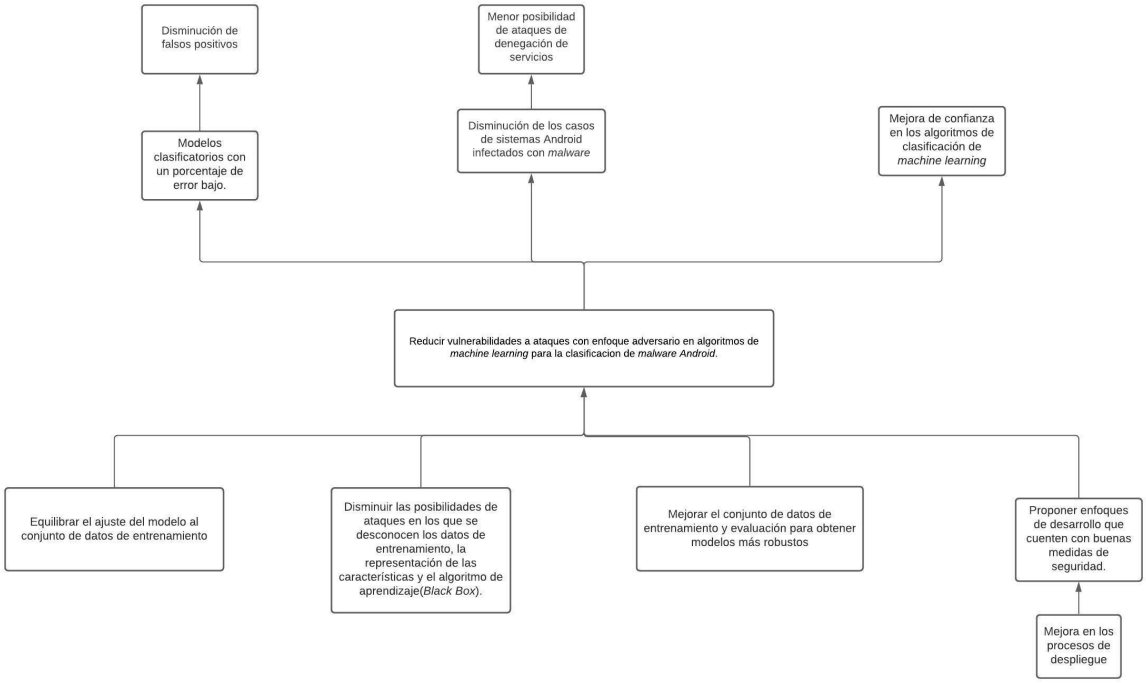
- Llevar a cabo el proceso de modificación de un *malware* real con el fin de demostrar los riesgos para un modelo de clasificación poco seguro.
- Proponer un método de ataque adversario y posterior entrenamiento desde un enfoque *black box*.
- Continuar con el análisis de los rangos para determinar el intervalo en el que un atacante podría modificar un malware.

Anexos

Anexo 1.1: Árbol del problema



Anexo 1.2: Árbol de objetivos



Anexo 1.3: Análisis de Participación

PARTICIPANTE	AFECTADO	BENEFICIARIO DIRECTO	BENEFICIARIO INDIRECTO	NEUTRAL
Cibercriminal	X			
Usuario de aplicaciones móviles Android		X		
Organizaciones y desarrolladores con buenas intenciones		X		
Sistema Android			X	
Otros sistemas operativos				X
Usuarios de otros sistemas operativos				X

Referencias bibliográficas

[1] “Operating System Market Share Worldwide,” *StatCounter Global Stats*, Feb-2021. [Online]. Available: https://gs.statcounter.com/os-market-share. [Accessed: 16-Mar-2021].

[2] P. by S. R. Department and F. 4, “Google Play Store: number of apps 2020,” *Statista*, 04-Feb-2021. [Online]. Available: https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/. [Accessed: 16-Mar-2021].

[3] C. Urcuquí , M. García , J. Osorio , and A. Navarro , Ciberseguridad: un enfoque desde la ciencia de datos, 1 ed. Cali, Valle Del Cauca: Editorial Universidad Icesi, 2018.

[4] S. Chaieb, “Machine Learning Systems: Security,” Sahbi Chaieb, 03-Feb-2021. [Online]. Available: https://sahbichaieb.com/mlsystems-security/. [Accessed: 16-Mar-2021].

[5] J. Delgado, “Secure Learning para detección de Android Malware,” thesis, 2019.

[6] B. Dickson, “Robust AI: Protecting Neural Networks Against Adversarial Attacks,” *Experfy Insights*, 18-Nov-2020. [Online]. Available: https://www.experfy.com/blog/ai-ml/robust-ai-protecting-neural-networks-against-adversarial-attacks/. [Accessed: 16-Mar-2021].

[7] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar, “The security of machine learning,” Machine Learning, vol. 81, no. 2, pp. 121–148, 2010.

[8] S. Shalev-Shwartz and S. Ben-David, Understanding Machine Learning: From Theory to Algorithms, 1st ed. New York: Cambridge University, 2014, pp. 19-25.

[9] J. Patterson and A. Gibson, Deep learning, 2nd ed. pp. 1-15.

[10]B. Brown and A. Zai, Deep reinforcement learning in action, 1st ed. NY: Manning Publications Co, 2020.

[11] R. Sutton and A. Barto, *Reinforcement learning*, 2nd ed.

[12] J. Graham, R. Howard and R. Olson, CYBER SECURITY ESSENTIALS, 1st ed. [Place of publication not identified]: CRC Press, 2017.

[13] M. Sikorski and A. Honig, Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software, 1st ed. Manassas Park, Virginia: No Starch Press, 2012, pp. 24-30.

[14] G. Weidman, Penetration Testing - A hands-on introduction to Hacking, 1st ed. San Francisco: no statch press, 2014, pp. 1-10.

[15]M. Barreno, A. D. Joseph, and J. D. Tygar, "Can Machine Learning Be Secure?," pp. 16–25, 2006.

[16] Rathore H, Sahay SK, Nikam P, Sewak M. Robust Android Malware Detection System Against Adversarial Attacks Using Q-Learning. Information Systems Frontiers. 2020 Nov 15:1-6.

[17] Romdhana A, Merlo A, Ceccato M, Tonella P. Deep Reinforcement Learning for Black-Box Testing of Android Apps. arXiv preprint arXiv:2101.02636. 2021 Jan.

[18] Fang Z, Wang J, Li B, Wu S, Zhou Y, Huang H. Evading anti-malware engines with deep reinforcement learning. IEEE Access. 2019 Mar 28;7:48867-79.

[19] Li H, Zhou S, Yuan W, Li J, Leung H. Adversarial-example attacks toward android malware detection system. IEEE Systems Journal. 2019 Apr 11;14(1):653-6.

[20] "CRISP-DM" [Online], Available: http://crisp-dm.eu/ [Accessed: 9-Jun-2021].

[21] "Cyber kill chain" [Online], Available: https://www.sans.org/blog/applying-security-awareness-to-the-cyber-kill-chain/ [Accessed: 10-nov-2021].

[22] Sarkar, D., Bali, R., Sharma, T. (2018). Practical Machine Learning with Python. Berkeley, CA: Apress.

[23] Langr, J., & Bok, V. (2019). GANs in action (2nd ed.). Shelter Island, New York: Manning Publications.

[24]Siva, R., & Johnson, A. (2020). Cyberattacks against machine learning systems are more common than you think - Microsoft Security Blog. Retrieved 8 December 2021, from
[25]https://www.microsoft.com/security/blog/2020/10/22/cyberattacks-against-machine-learning-systems-are-more-common-than-you-think/