

FACULTAD DE INGENIERÍA

PROGRAMA DE INGENIERÍA DE SISTEMAS

DEPARTAMENTO DE TECNOLOGÍAS DE INFORMACIÓN Y TELECOMUNICACIONES

PROYECTO DE GRADO  
SISTEMA DE ANÁLISIS DE TRÁFICO WEB PARA LA DETECCIÓN DE MALWARE EN DISPOSITIVOS ANDROID   
INTEGRANTE  
ANDRÉS FELIPE PÉREZ BELALCAZAR  
DIRIGIDO POR  
CHRISTIAN URCUQUI LÓPEZ, MSc

ANDRÉS NAVARRO CADAVID, PHD

Santiago de Cali, Mayo de 2018

**Tabla de contenido**

Resumen 4

Abstract 5

Lista de acrónimos 6

Glosario de términos 7

Índice de figuras 8

Motivación y antecedentes 9

Descripción del problema 13

Objetivos 14

Objetivo general 14

Objetivos específicos 14

Marco teórico 15

Sistema operativo Android 15

Seguridad en Android 16

Vulnerabilidades en Android 17

Dataset de malware y familias de malware 18

Tráfico en internet 19

Aprendizaje supervisado: árboles de decisión 21

Estado del arte 22

CREDROID: detección de malware en Android mediante el análisis de tráfico de red 22

Detección de malware mediante análisis de tráfico de red en dispositivos móviles basados en Android 23

DroidAlarm: una herramienta de análisis estático multifuncional para el malware escalado de privilegios en Android 24

Una primera mirada al tráfico de malware de Android en los primeros minutos 24

Resumen 26

Metodología 27

Fases de desarrollo del proyecto 27

Análisis e investigación 27

Diseño 28

Implementación 28

Pruebas y validación 32

Contribución y resultados del proyecto de grado 33

Resultados 33

Conclusiones y trabajo a futuro 39

Anexos 42

Árbol de problemas 42

Árbol de objetivos 43

Análisis de participación 44

Referencias bibliográficas 46

## Resumen

El sistema operativo Android se ha convertido poco a poco en uno de los más usados en todo el mundo; con el avance de la tecnología y del internet de las cosas, ahora es posible encontrar este sistema operativo en más que sólo dispositivos móviles. Por desgracia esta popularidad ha ocasionado que sea el objetivo principal de distintos tipos de software malicioso (malware), cuya principal función va desde el robo de información hasta el de dinero, provocando problemas en sus usuarios. Pese a que existen algunos métodos de detección de malware, como lo son el análisis estático y dinámico, éstos aún tienen muchos aspectos que mejorar en cuanto a la detección de malware, y por esta razón se ha tenido la necesidad de explorar otras formas de lograr una mejora en dichas detecciones. Este proyecto de grado se centra en el análisis de tráfico Web; analizando determinadas características de dicho tráfico y aplicando algoritmos de *machine learning* en las funciones de tráfico extraídas con el fin de identificar comportamientos de red maliciosos. Los resultados experimentales sugieren que el enfoque es notablemente preciso, y se espera que se detecte más del 90% de las muestras de tráfico malicioso.

## Abstract

## The Android operating system has gradually become one of the most widely used operating systems worldwide. With the advance of technology and the internet of things, it is now possible to find this operating system on more than just mobile devices. Unfortunately, this popularity has made it the main target of different types of malware, aiming to steal information and/or money, and causing problems for its users. Although there are some methods of malware detection, such as static and dynamic analysis, there is still much room for improvement in the detection of malware, and for this reason there has been a need to explore other ways of improving malware detection. This graduation project focuses on Web traffic analysis, analyzing certain characteristics of Web traffic, and running machine learning algorithms on the extracted traffic features to identify malicious network behaviors. Experimental results suggest that the approach is remarkably accurate, and more than 90% of malicious traffic samples are expected to be detected.

## Lista de acrónimos

SO Sistema operativo

ROM Memoria de sólo lectura (***R****ead* ***O****nly* ***M****emory*)

RIM Nombre de la compañía que fabricaba los dispositivos Blackberry (***R****esearch* ***I****n* ***M****otion*)

SMS Servicio de mensajes cortos (***S****hort* ***M****essage* ***S****ervice*)

GPS Sistema de posicionamiento global (***G****lobal* ***P****ositioning* ***S****ystem*)

APK Aplicación empaquetada de Android (***A****ndroid A****p****plication Pac****k****age*)

XML Lenguaje de Marcas Extensible (*e****X****tensible* ***M****arkup* ***L****anguage*)

UID Identificación de usuario (***U****ser* ***ID***)

SD Tipo de tarjeta de memoria (***S****ecure* ***D****igital*)

DNS Sistema de nombres de dominios (***D****omain* ***N****ame* ***S****ystem)*

DHCP Protocolo de configuración dinámica de host (***D****ynamic* ***H****ost* ***C****onfiguration* ***P****rotocol*)

HTTP Protocolo de transferencia de hipertexto (***H****ypertext* ***T****ransfer* ***P****rotocol*)

## Glosario de términos

**Técnicas de ofuscación [1]:** proceso mediante el cual se transforma a través de la aplicación de algoritmos de reescritura, un código perfectamente legible y entendible por una persona en otro de funcionalidad equivalente en un ciento por ciento, pero, en términos ideales, totalmente ilegible e incomprensible para un lector humano.

**Smartphone [2]**: Los modernos teléfonos móviles de gama alta que combinan la funcionalidad de un dispositivo de comunicación de bolsillo con funciones parecidas a las de un PC.

**AndroidManifest.xml**: es un archivo de configuración donde se pueden aplicar las configuraciones básicas de una aplicación Android.

**META-INF**: es una carpeta que contiene los "scripts" necesarios para realizar la instalación de una ROM, esta carpeta contiene archivos responsables de la carga de dicho ROM en el sistema o dispositivo.

**classes.dex**: este archivo contiene el código java, que será interpretado por la máquina virtual de Android, este archivo es una parte importante en la formación de un archivo APK.

**Malware [3]**: Aplicaciones maliciosas o “malware” es un software diseñado para afectar un sistema y a sus usuarios. Es un tipo de software que tiene como objetivo infiltrarse o dañar una computadora o sistema de información, por ejemplo muchos virus son diseñados para destruir archivos en disco duro o para corromper el sistema de archivos escribiendo datos inválidos. Otros tipos de malware como el spyware, el adware intrusivo y los secuestradores tratan de mostrar publicidad no deseada o redireccionar visitas hacia publicidad para beneficio del creador.

**Firma de malware**: son una secuencia continua de bytes comunes en cierta muestra de malware, lo que significa que se contiene dentro de este o del archivo infectado y no en archivos no afectados. Una firma puede ser un conjunto de instrucciones, un hash o un modo de comportamiento.

**Dataset**: es una base de datos empleada para realizar estudios en el ámbito investigativo con el fin de tener resultados en dicho estudio.

**Middleware [26]**: es un software que se sitúa entre un sistema operativo y las aplicaciones que se ejecutan en él. Básicamente, funciona como una capa de traducción oculta para permitir la comunicación y la administración de datos en aplicaciones distribuidas.

**Dalvik:** es la máquina virtual que se usaba para ejecutar aplicaciones java en Android.

## Índice de figuras

Figura 1: ventas globales de smartphones desde 2009 hasta 2017 …………….. Página 10

Figura 2: Arquitectura del sistema operativo Android.…………………. Página 15   
Figura 3: Componentes de un sistema de transmisión de datos…………….…. Página 20

Figura 4: Sistema implementado por CREDROID.………….…………………. Página 22

Figura 5: Fases del método utilizado…………………………………………… Página 23

Figura 6: Plataforma de captura y generación de tráfico de malware……… …. Página 25

Figura 7: Algoritmo de generación de tráfico…………………………………… Página 29

Figura 8: Particionamiento del sistema propuesto……………………………….. Página 33

Figura 9: Vista general del sistema propuesto…………………………………….Página 35

**Índice de tablas**

Tabla 1: Resumen de las características de algunos proyectos versus el propuesto.……………………..……………. …………………………………. Página 26

Tabla 2: Asociación de los requerimientos a los distintos subsistemas.… Página 34  
Tabla 3: resultados del análisis estadístico del proyecto……………………..Página 36  
Tabla 4: Promedios de las características obtenidas del estudio [36] realizado en i2t (Icesi)………………………………………………………………………………Página 37

Tabla 5: resultados en el desempeño individual de los 4 clasificadores…………...Página 38

## 

## Motivación y antecedentes

El uso de dispositivos móviles, tales como *smartphones* o *tablets,* ha tenido un crecimiento significativo en los últimos años y seguirá creciendo, ya que “se pronostica que los dispositivos móviles generarán el 98 por ciento del tráfico de datos móviles en el 2020. En tal sentido, los teléfonos inteligentes seguirán jugando un papel dominante, ya que representarán el 81 por ciento del tráfico móvil en cinco años, comparado con el 76 por ciento que se generó en el 2015” [4].

Android es un sistema operativo desarrollado y mantenido por Google para *smartphones* y tabletas, que ofrece un ambiente de desarrollo de software libre y tiene herramientas, aplicaciones y emuladores para desarrollar aplicaciones en Java. En el año 2016 la plataforma Android tuvo una participación en el mercado de 81,7%, la cual es superior con respecto a su competidor IOS que sólo tuvo un 17,9% [5]. Es uno de los sistemas operativos más famosos en el mundo, ya que es usado en teléfonos inteligentes, computadores portátiles, tabletas, Google TV, relojes de pulsera, auriculares​ y otros dispositivos.

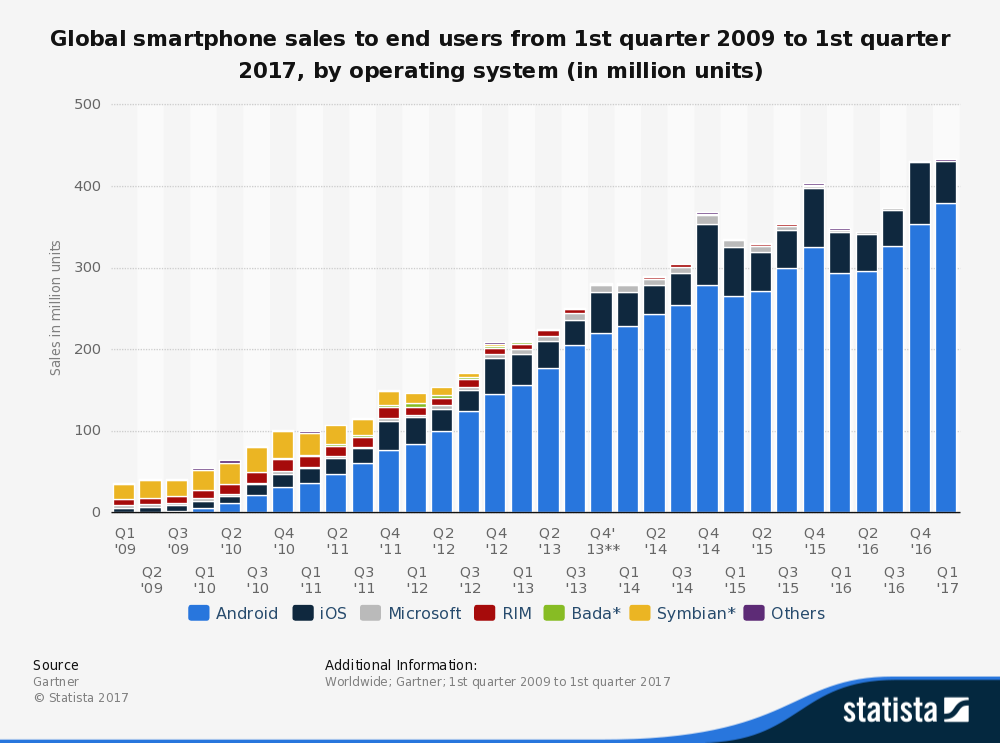


Figura 1 [5.2]: ventas globales de smartphones a usuarios finales desde 2009 hasta 2017

El crecimiento y la popularidad de Android ha ocasionado un gran interés por parte de los cibercriminales, de crear aplicaciones maliciosas que causan en los usuarios muchos problemas (robo de información y/o dinero) empleando todo tipo de código malicioso. Por ejemplo: “Los piratas informáticos pagan cerca de 2.000 dólares para obtener un programa cifrador de datos (ransomware) y piden 100 dólares al usuario para el desbloqueo de los datos. Sin embargo, el software que podría generarles mayores ingresos es el troyano que roba datos de cuentas bancarias. Programas de este tipo pueden costar alrededor de 3.000 dólares, mientras que los cibercriminales podrían ganar hasta 72.000 dólares. En este caso cada usuario que 'pica' perdería 722 dólares” [6].

Debido a que Android es muy usado por las personas (ésto se evidencia en la figura 1) se han creado métodos para el análisis de malware en dispositivos móviles. Según el artículo “Evolution of android malware and analysis and detection techniques” [7], dichos métodos son los siguientes:

* El análisis de la firma del malware, en el cual se extraen patrones o fragmentos aleatorios de una muestra. Sin embargo, este método es poco efectivo ya que se pueden obtener muchas firmas y resulta ser una tarea costosa, además de que algunas técnicas de ofuscación pueden lograr que ciertos tipos de malware no sean detectados usando este método.
* Otro método es el análisis estático, el cual examina el código y los datos de la aplicación, usando archivos tales como META-INF, AndroidManifest.xml, classes.dex, entre otros. Pero, al igual que la técnica anterior, los métodos de ofuscación hacen que este tipo de análisis sea poco efectivo.
* Un tercer método es el análisis dinámico, que consiste en ejecutar un programa y observar su comportamiento. Generalmente, esto se hace de una forma instrumentada o monitoreada para recolectar más información concreta de su comportamiento. La desventaja de este tipo de análisis es que cierto tipo de malware puede detectar el comportamiento emulado que muchas veces se usa en este método y, por lo tanto, toman medidas para no hacer nada cuando se encuentran en estos entornos.
* El último tipo de método es el análisis híbrido, el cual combina los últimos dos mencionados anteriormente, pero por desgracia también es falible.

Esto plantea la pregunta de si es posible encontrar una metodología diferente para obtener mejores resultados. Para ayudar a responder esto, de nuevo el texto “Evolution of android malware and analysis and detection techniques” [8] menciona que existen otras técnicas para el análisis de malware que han sido poco investigadas; entre ellas, el análisis de tráfico en internet. Esto es importante puesto que actualmente la mayoría de aplicaciones requiere de conexión a la red, es más, según un informe de Android Authority: “En 2012, solo un 40% de las conexiones a Internet se realizaban a través de dispositivos móviles. Este año, este dato llega hasta un 68%, y en 2018 alcanzaría un 79% del tráfico global” [9].

Además de esto, las aplicaciones maliciosas exigen mucha más conectividad a la red de las que no lo son [10]. Actualmente se emplean los métodos de análisis de tráfico en la red, tanto en PCs como en dispositivos móviles, y se estudia la forma en que estos diferentes análisis se pueden aplicar o complementar. Una de las aproximaciones orientadas a este tipo de análisis son algoritmos que utilizan el tráfico Web para extraer características de interés para ser examinadas, además muchas investigaciones se han empezado a enfocar en esta metodología tanto en aplicaciones maliciosas como las que no lo son y así poder realizar experimentos [11]. Este tipo de análisis ha tomado fuerza últimamente, pero se requiere de mucha investigación para lograr más avances.

## 

# Descripción del problema

El desarrollo de métodos para evitar la detección de malware, el bajo monitoreo del tráfico malicioso en la red, y las aplicaciones alteradas, entre otros, han generado un aumento de amenazas informáticas en dispositivos Android. Todo esto ha llevado a que haya pérdida de información y de dinero por parte de los usuarios, además de que haya más interés por parte de los cibercriminales en los ataques maliciosos y, por último, que se requiera una mayor investigación a la hora de plantear soluciones contra los ataques que se producen.

# Objetivos

### Objetivo general

Desarrollar un sistema de análisis de tráfico de red para detectar aplicaciones maliciosas en dispositivos Android.

### Objetivos específicos

1. Investigar, examinar y diseñar un método para la generación de tráfico Web de aplicaciones Android.
2. Implementar el método propuesto en el objetivo 1 y extraer características del tráfico generado.
3. Evaluar un algoritmo de clasificación que utilice los criterios obtenidos en el objetivo 2, con el fin de detectar si una aplicación es maliciosa.
4. Comparar el sistema propuesto y los resultados obtenidos con un estudio realizado anteriormente.

# Marco teórico

El marco teórico de este proyecto está enfocado en el sistema operativo Android, en su seguridad, sus vulnerabilidades, entornos virtualizados para este sistema operativo, y por último, algunas características tanto del tráfico en la red como en los dispositivos Android.

## Sistema operativo Android

Android es una plataforma de código abierto, con gran ventaja sobre los demás sistemas operativos para móviles tales como Nokia (Symbian), Apple (iOS) o RIM (Blackberry), ya que fabricantes, operadores y desarrolladores pueden dar mayor utilidad al *Smartphone* o tableta. Además de ser un sistema gratuito y multiplataforma, se puede instalar muy fácilmente en dispositivos móviles, inclusive de gama baja.

“Android es un software pensado para dispositivos móviles, que incluye el sistema operativo como middleware y diversas aplicaciones de usuario. Todas las aplicaciones para Android se programan en lenguaje Java y son ejecutables en una máquina virtual diseñada para esta

plataforma, llamada Dalvik (para versiones anteriores a la 5.0) o ART (Android Runtime, el cual sustituyó a Dalvik a partir de la versión 5.0)” [12].

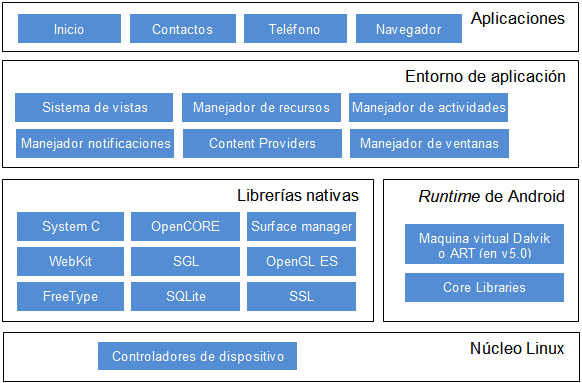
En la figura 2 se puede observar que la arquitectura de Android está compuesta de los siguientes elementos: el kernel de Linux, el cual provee los drivers para el hardware, redes, acceso a los archivos del sistema y gestión de los procesos; las librerías, que se usan en todo el sistema y proporcionan distintos módulos o componentes que pueden usar los desarrolladores; Android Runtime, que contiene el binario “init”, encargado de ejecutar el primer proceso en Android y de dar inicio al resto de procesos programados; y el framework de Java API. La mayor parte de Android se implementa en Java y es ejecutado por una Máquina Virtual Java (JVM), debido a que las funciones del SO Android están disponibles mediante un conjunto de APIs escritas en el lenguaje Java y, por último, las aplicaciones del sistema. Android incluye un conjunto de aplicaciones centrales para correo electrónico, mensajería SMS, calendarios, navegación en Internet y contactos, entre otros elementos [13].

Figura 2: Arquitectura del sistema operativo Android

Otra característica importante es el Android Virtual Device (AVD), el cual según la página de internet Android Developer [18], permite definir las características de un teléfono Android, una tablet, o un dispositivo Android Wear o Android TV que se desee simular en Android Emulator. Una máscara de emulador especifica la apariencia de un dispositivo. El Administrador de AVD proporciona algunas máscaras predefinidas. Al igual que con un dispositivo real, para que las aplicaciones usen determinadas funciones definidas en un AVD, como la cámara, ésta debe tener la correspondiente configuración <uses-feature> en su manifiesto.

## Seguridad en Android

Al igual que el resto del sistema, el modelo de seguridad de Android también aprovecha las características de seguridad que ofrece el núcleo de Linux. Linux es un sistema operativo multiusuario, y su kernel puede aislar los recursos de los usuarios entre sí, del mismo modo que aísla los procesos [14].

En los dispositivos móviles, la seguridad juega un papel importante, ya que las aplicaciones maliciosas pueden actuar de forma que lean la lista de contactos, averigüen la posición GPS, envíen toda esta información por internet, o envíen mensajes SMS.

La seguridad en Android [15] se fundamenta en los siguientes tres pilares:

* Android impide que las aplicaciones tengan acceso directo al hardware o interfieran con recursos de otras aplicaciones.
* Toda aplicación ha de ser firmada con un certificado digital que identifique a su autor. La firma digital también garantiza que el archivo de la aplicación no ha sido modificado. Android utiliza la firma del archivo APK para asegurarse de que las actualizaciones de una aplicación proceden del mismo autor (esto se denomina política de mismo origen) y para establecer relaciones de confianza entre aplicaciones.
* Si se desea modificar la aplicación, ésta tendrá que ser firmada de nuevo, y esto solo podrá hacerlo el propietario de la clave privada. No es preciso (ni frecuente) que el certificado digital de la aplicación sea firmado por una autoridad de certificación.

Un elemento de seguridad de Android [16] es el archivo de manifiesto (AndroidManifest.xml) que está incluido en el paquete de instalación de Android (archivo APK), junto con el bytecode Java y otros recursos relacionados. Este archivo está escrito en XML y proporciona toda la información necesaria a la plataforma Android para la ejecución de la aplicación. El archivo de manifiesto es crucial para el sistema, ya que en él se definen los permisos de cada aplicación. Estos permisos funcionan en las dos siguientes formas: la primera es cómo la aplicación interactúa con el sistema mediante el acceso a la API del mismo; y, en segundo lugar, la forma como el sistema y otras aplicaciones interactúan con la aplicación dada.

Según el proyecto “Seguridad en dispositivos móviles Android” [17], a cada aplicación se le asigna un directorio de datos dedicado, donde solamente ella tiene permiso para leer y escribir. De este modo, las aplicaciones son aisladas a través de entornos de pruebas (sandbox), tanto a nivel de proceso (haciendo que cada una se ejecute en un proceso dedicado) como a nivel de archivo (mediante el directorio de datos privado).

Para distinguir las aplicaciones instaladas para cada usuario, Android asigna un nuevo UID efectivo a cada aplicación cuando se instala para un usuario en particular. Este UID efectivo se basa en el ID de usuario físico de destino y el UID de la aplicación en un sistema de un solo usuario (el ID de aplicación).

Debido a que las aplicaciones de Android son *sandboxed*, sólo pueden acceder a sus propios archivos y a cualquier recurso con acceso general en el dispositivo. Sin embargo, una aplicación tan limitada no sería muy interesante, y Android puede conceder derechos detallados de acceso adicional a las aplicaciones para permitir una funcionalidad más completa. Estos derechos de acceso se denominan permisos y pueden controlar el acceso a dispositivos de hardware, conectividad a Internet, datos o servicios del sistema operativo. Antes de Android 6.0 (Marshmallow), una vez concedidos, los permisos no podían ser revocados y estaban disponibles para la aplicación sin ninguna confirmación adicional.

## Vulnerabilidades en Android

El libro “Android™ Hacker’s Handbook” [21], explica que los ataques contra DNS son mucho más fáciles, porque la baja latencia asociada con la adyacencia de red significa que los atacantes pueden responder más rápido que los hosts basados en Internet. Los ataques de espionaje contra DHCP también son bastante efectivos para ganar más control sobre un sistema de destino.

Los ataques de hombre en la mitad, o Man-in-the-Middle (MitM), son bastante poderosos. Los métodos para convertirse en un atacante en el medio incluyen comprometer enrutadores o servidores DNS, usar interceptaciones legales, manipular hosts mientras están conectados a la red y modificar las tablas de enrutamiento global de Internet.

Otro método, que parece menos difícil que el resto en la práctica, es secuestrar el servicio DNS a través de los registradores. Otra manera relativamente fácil de efectuar un ataque MiTM es específica para redes inalámbricas, tales como WiFi y celulares. Los ataques exitosos contra los navegadores Web se pueden llevar a cabo de varias maneras. El método más común consiste en persuadir a un usuario para que visite una URL que está bajo el control del atacante. Este método es probablemente el más popular debido a su versatilidad. Un atacante puede fácilmente entregar una URL por correo electrónico, medios sociales, mensajería instantánea u otros medios. Otra forma es insertando el código de ataque en sitios comprometidos que las víctimas pretenden visitar. Las redes publicitarias representan una pieza interesante y potencialmente peligrosa del rompecabezas por varias razones. La funcionalidad que genera los anuncios publicitarios se basa generalmente en un motor de navegación integrado (un WebView). Además del riesgo de ejecución remota de código, los marcos publicitarios también presentan un riesgo significativo para la privacidad.

Las aplicaciones de terceros (tiendas no oficiales) representa otra importante fuente de ataque remoto. Tal vez el mejor ejemplo es una aplicación Android. Como es evidente por ahora, las aplicaciones Android contienen código que se ejecuta directamente en un dispositivo Android. Por lo tanto, la instalación de una aplicación equivale a conceder una ejecución de código arbitraria (aunque dentro del entorno de Android a nivel de usuario) al desarrollador de la aplicación.

## Dataset de malware y familias de malware

Existen actualmente muchas familias de malware, algunas de ellas según Kaspersky [19] son:

* **Gusanos de red:** Emplean recursos de red para distribuirse, tales como redes locales, redes globales, etc. Su velocidad de propagación es muy alta, y tienden a consumir los recursos de la máquina a la que atacan.
* **Troyanos:** incluyen una gran variedad de programas que efectúan acciones sin que el usuario se dé cuenta y sin su consentimiento: recolectan datos y los envían a los criminales; destruyen o alteran datos con intenciones delictivas, causando desperfectos en el funcionamiento del computador, o usan los recursos del computador para fines criminales, como hacer envíos masivos de correo no solicitado.
* **Spyware:** software que permite recolectar la información sobre un usuario u organización de forma no autorizada. Puede recolectar los datos sobre las acciones del usuario, el contenido del disco duro, software instalado, calidad y velocidad de la conexión, etc.
* **Adware:** este tipo de malware está relacionado con la publicidad, que es mostrada al usuario. La mayoría de programas adware son instalados a software de distribución gratuita. La publicidad aparece en la interfaz. A veces pueden colectar y enviar los datos personales del usuario, lo que significa que en algunos casos pueden actuar como un spyware.
* **Rootkit:** es una colección de programas usados por un hacker para evitar ser detectado mientras busca obtener acceso no autorizado a un computador. Esto se logra reemplazando archivos o librerías del sistema; o instalando un módulo de kernel. Después de instalar el rootkit como administrador del sistema, se obtiene un acceso similar al del usuario: por lo general, hackeando una contraseña o explotando una vulnerabilidad, lo que permite usar otras credenciales hasta conseguir el acceso de raíz o administrador.

Un *dataset* de malware contiene varias muestras de aplicaciones que a lo largo de los años se han detectado como maliciosas, y se han almacenado en un conjunto de datos o bases del conocimiento para el análisis por parte de los investigadores [20]. Estos *datasets* permiten llevar a cabo experimentos con las aplicaciones y así desarrollar métodos de detección o mejoras en los métodos actuales. En el caso de Android existen muchos proyectos que proporcionan sus datasets al público investigador, tal es el caso de los proyectos Drebin y Android MalGenome.

## Tráfico en internet

Internet es una red interconectada de sistemas informáticos. Los computadores domésticos y los dispositivos móviles son los nodos más externos de la red. Entre estos nodos se encuentra un gran número de sistemas backend llamados routers. Cuando un *Smartphone* se conecta a un sitio Web, una serie de paquetes que utilizan varios protocolos atraviesan la red para localizar, contactar e intercambiar datos con el servidor solicitado (Figura 3).

Según el libro “Transmisión de datos y redes de comunicaciones” [20], el modelo OSI permite la comunicación entre sistemas distintos sin que sea necesario cambiar la

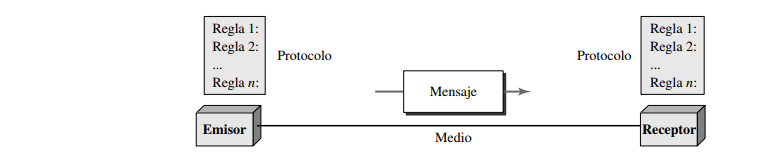


Figura 3 [21]: Componentes de un sistema de transmisión de datos.

lógica del hardware o el software subyacente. Este modelo está compuesto por 7 niveles ordenados: el físico (nivel 1), el de enlace de datos (nivel 2), el de red (nivel 3), el de transporte (nivel 4), el de sesión (nivel 5), el de presentación (nivel 6) y el de aplicación (nivel 7). Dentro de una máquina, cada nivel llama a los servicios del nivel que está justo por debajo. La comunicación se gobierna mediante una serie de reglas y convenciones acordadas que se denominan protocolos.

La transmisión de datos en el nivel físico significa mover los bits en forma de señal de un origen a un destino. A nivel de enlace, se empaquetan los bits en tramas, de forma que una trama se pueda distinguir de otra. En un protocolo orientado a bit, la selección de datos de una trama es una secuencia de bits a ser interpretados por el nivel superior como texto, gráficos, audio, etc.; esta secuencia está compuesta de un delimitador llamado flag, una cabecera, una dirección de origen, de destino y los datos.

Entre los protocolos usados en las comunicaciones en red se encuentran: el protocolo UDP, el cual es un protocolo sin conexión y no fiable, pero si un proceso quiere enviar un mensaje pequeño y no le preocupa mucho la fiabilidad, puede usar UDP y, por último, este protocolo es mucho más rápido que el TCP. El protocolo TCP es orientado a conexión; crea una conexión virtual entre dos procesos para enviar datos. Además, es fiable, ya que usa mecanismos de control de flujo y error a nivel de transporte.

El sistema de nombres de dominio (DNS) permite a un programa encontrar la dirección IP de otro computador. El DNS puede traducir un nombre a una dirección, y de forma inversa, una dirección a un nombre. Una estación que necesita traducir una dirección a un nombre o un nombre a una dirección llama al cliente DNS, denominado resolvedor. El resolvedor accede al servidor DNS más cercano con la petición de traducción. Si el servidor tiene la información, responde al resolvedor; en caso contrario, pregunta a otros servidores para obtener información.

## Aprendizaje supervisado: árboles de decisión

*Machine learning* es una disciplina científica del ámbito de la Inteligencia Artificial, que crea sistemas que pueden aprender automáticamente. Aprender en este contexto quiere decir identificar patrones complejos entre millones de datos. La máquina que realmente aprende es un algoritmo que revisa los datos y es capaz de predecir comportamientos futuros. Automáticamente, también en este contexto, implica que estos sistemas se mejoran de forma autónoma con el tiempo, sin intervención humana.

El aprendizaje supervisado es un enfoque aplicado en *machine learnin*g, en este se busca entrenar modelos a partir de un conjunto de datos, los cuales contienen la variable objetivo a predecir, es decir, éste tipo de aprendizaje busca crear una función capaz de predecir el valor correspondiente a cualquier objeto de entrada válida después de haber visto una serie de ejemplos (los datos de entrenamiento). Para ello, tiene que generalizar a partir de los datos presentados.

Un árbol de decisión es un modelo de predicción que hace el uso del aprendizaje supervisado, éste tipo de árbol está compuesto por nodos, vectores, ramas y etiquetas. Según un trabajo realizado en la Universidad Nacional de Colombia [28], en un árbol de decisión las ramas representan conjuntos de decisiones, y cada decisión genera  
reglas sucesivas para continuar la clasificación (partición), formando así grupos homogéneos respecto de la variable que se desea discriminar. Las particiones se hacen en forma recursiva hasta que se alcanza un criterio de parada, el método utiliza datos históricos para construir el árbol de decisión, y este árbol se usa para clasificar  
nuevos datos. El árbol se divide repetidamente en datos que constituyen grupos excluyentes, esta división o partición aplica tanto para los nodos padre como para los hijos.

Las divisiones se seleccionan de modo que la “impureza” (medida de la homogeneidad) de los hijos sea menor que la del grupo madre; están definidas por un valor de una variable explicativa. El objetivo es particionar la respuesta en grupos homogéneos y a la vez mantener el árbol razonablemente pequeño. Para dividir los datos se requiere un criterio de particionamiento que determinará la medida de impureza, esta última establecerá el grado de homogeneidad entre los grupos.

# Estado del arte

Para exponer el estado del arte del proyecto, se muestran diferentes proyectos, experimentos y sistemas desarrollados para el análisis de malware en dispositivos Android. Se analizan las distintas metodologías aplicadas y se explica por qué estas propuestas no solucionan el problema planteado o aún no son suficientes para resolverlo.

## CREDROID: detección de malware en Android mediante el análisis de tráfico de red

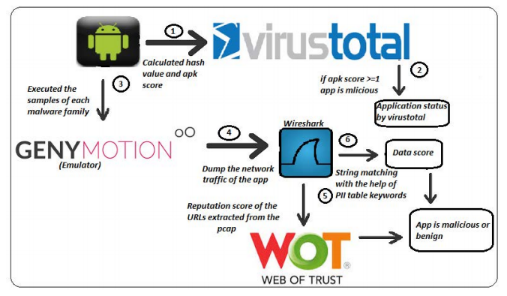
CREDROID es un método que identifica las aplicaciones maliciosas con base en sus consultas a los Servidor de Nombres de Dominio (DNS) y a los datos que transmite al servidor remoto, realizando un análisis en profundidad de los registros de tráfico de la red en modo offline. En lugar de realizar una detección basada en firmas que es incapaz de detectar malwares polimórficos, este método propone una detección basada en patrones. El patrón que se trabaja se refiere a la fuga de información sensible que se envía al servidor remoto.

Figura 4: Sistema implementado por CREDROID

CREDROID es un enfoque semi-automático que trabaja sobre diversos factores, tales como el servidor remoto donde se está conectando la aplicación, los datos que se envían y el protocolo utilizado para la comunicación para identificar la fiabilidad de la aplicación. En este trabajo, se observó que el 63% de las aplicaciones de un conjunto de datos estándar de malware están generando tráfico de red que ha sido el foco de este trabajo [22].

A pesar de que CREDROID es un método que ha implementado un sistema de monitoreo muy completo, en este trabajo no se enfocaron mucho en obtener características del tráfico sino en monitorear las interacciones de las aplicaciones con el servidor, que es solamente una de las características para analizar con relación a la detección de malware usando en tráfico en la red.

## Detección de malware mediante análisis de tráfico de red en dispositivos móviles basados en Android

Este trabajo analiza las características del tráfico de red, y crea un clasificador basado en reglas para la detección de malware Android. Los resultados experimentales sugieren que el enfoque es notablemente preciso e identifica correctamente más del 90% de las muestras de tráfico.

El objetivo principal del trabajo es detectar malware que es controlado remotamente por algún servidor, y que obtiene comandos desde dicho servidor o filtra información privada de los usuarios. El funcionamiento del marco de detección se divide principalmente en dos fases. La primera fase consiste en analizar el tráfico de red del software malicioso y encontrar la lista de aquellas características que distinguen completamente el tráfico de malware del tráfico móvil normal. La segunda fase consiste en la construcción de un clasificador basado en una regla que incluye las características distintivas encontradas, y en la ejecución del clasificador sobre los datos de prueba para demostrar su exactitud. La figura 5 muestra los pasos de la aproximación [23].

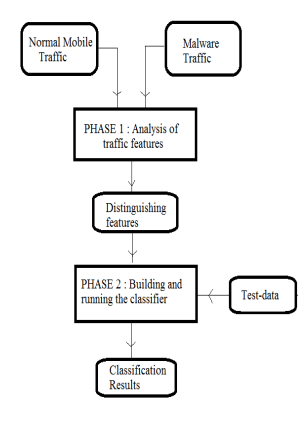


Figura 5: Fases del método utilizado

Las fases de este método permiten que la detección de malware mediante el tráfico en la red sea más eficiente. La desventaja más grande que posee este método es el *dataset* utilizado, ya que este trabajo fue hecho en el año 2014 y por lo tanto el *dataset* es de ese mismo año. Actualmente el malware ha evolucionado y por tanto este trabajo no es suficiente para aplicarlo en la actualidad.

## DroidAlarm: una herramienta de análisis estático multifuncional para el malware de escalado de privilegios en Android

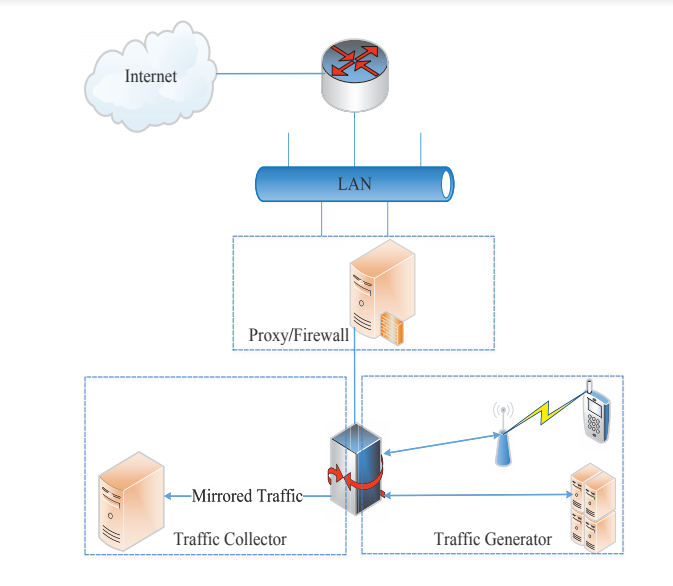
El problema central de este trabajo está enfocado en que Android contiene recursos sensibles a los que sólo se puede acceder a través de las API correspondientes, y éstas sólo se pueden invocar cuando el usuario tiene permisos autorizados en el modelo de permisos de Android. Sin embargo, una nueva amenaza llamada ataque de escalamiento de privilegios puede pasar por alto éste mecanismo de seguridad. Se presenta cuando una aplicación con menos permisos accede a recursos sensibles a través de interfaces públicas de una aplicación más privilegiada, lo que es especialmente útil para que el malware oculte funciones sensibles dispersándolas en múltiples programas. En este trabajo se exploran las técnicas de evolución del software malicioso de escalamiento de privilegios en muestras del proyecto Android Malware Genome Project. Y han demostrado una gran eficacia frente a un conjunto de potentes herramientas antivirus proporcionadas por VirusTotal. Las relaciones de detección presentan una reducción diferente y diferenciada, comparada con una relación de detección media del 61% antes de la transformación. Con el fin de conquistar este modelo de amenaza, se desarrolló una herramienta llamada DroidAlarm para llevar a cabo un análisis de espectro completo, con el fin de identificar posibles fugas de seguridad y presentar las rutas de fuga mediante el análisis estático en aplicaciones Android [24].

Una de las ventajas de este trabajo es la reducción de falsos positivos, que consisten en la clasificación errónea de software no malicioso como malware. Aunque DroidAlarm tiene muchas ventajas, es un método de análisis estático y por lo tanto no proporciona una metodología muy aplicable al análisis de tráfico en la red, debido a que este tipo de método está más orientado al análisis dinámico.

## Una primera mirada al tráfico de malware de Android en los primeros minutos

En este trabajo, se diseña un esquema de monitoreo del comportamiento del tráfico de malware Android para capturar datos de tráfico generado por muestras de malware en un entorno real de Internet. Se captura el tráfico de la red a partir de 5560 muestras de malware en los primeros 5 minutos y analizan las principales composiciones de los datos de tráfico.

Para capturar y analizar el tráfico de la red de malware Android, se diseñó una plataforma de generación y captura de tráfico de malware (Figura 7). Este método emplea una plataforma para monitorear el tráfico, y luego ejecuta un gran número de muestras de malware Android. Se encontraron algunos elementos clave que afectan la generación de tráfico; con base en ellos se diseñó el algoritmo automatizado de generación y recolección de tráfico.

Mediante este método se descubrió que el tráfico HTTP y DNS representa más del 99% del tráfico de la capa de aplicación. Se presenta un análisis de las características de red relacionadas: consulta DNS, longitud del paquete HTTP, relación entre la cantidad de tráfico de enlace de subida y bajada, petición HTTP y característica de tráfico publicitario. Los resultados estadísticos que usan ilustran que: (1) más del 70% de los programas maliciosos generan tráfico malicioso en los primeros 5 minutos; (2) la consulta DNS y la solicitud HTTP se pueden utilizar para identificar el malware, y las tasas de detección alcanzan el 69,55% y el 40,89% respectivamente [25].

Pese a las características que se proporcionan y a los hallazgos realizados, este trabajo sólo tuvo en cuenta un *dataset* de aplicaciones maliciosas, lo que no permite una comparación de las características obtenidas versus aplicaciones no maliciosas.

Figura 6: Plataforma de captura y

generación de tráfico de malware

## Resumen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sistema/Características | CREDROID | Detección mediante al análisis de tráfico | DroidAlarm | Monitoreo del tráfico en los primeros minutos | Proyecto propuesto |
| Dataset para pruebas | Si | Si | Si | Si | Si |
| Uso de dataset actualizado (entre 2016 y 2017) | No | No | No | No | Si |
| Extracción e identificación de características | Si | Si | Si | Si | Si |
| Tipo de método de análisis | Dinámico | Dinámico | Estático | Dinámico | Dinámico |
| Uso del tráfico en la red | Si | Si | No | Si | Si |
| Entorno virtualizado | Si | Si | No | Si | Si |
| Análisis del tráfico de aplicaciones no maliciosas | Si | Si | No | No | Si |
| Uso de herramientas de simulación de movimientos | No | No | No | No | Si |

Tabla 1: Resumen de las características de algunos proyectos versus el propuesto.

En conclusión, uno de los principales diferenciadores del proyecto propuesto es el uso de un dataset más actualizado, tanto para el caso de las aplicaciones benignas como el de las maliciosas, además del uso de una herramienta que permitirá la simulación de movimientos del usuario con una aplicación, ya que al parecer esto no fue realizado en los trabajos evaluados, y esta característica es importante debido a que la interacción resultante puede agregar resultados interesantes.

# Metodología

Para el desarrollo de éste proyecto se empleó el modelo de ciclo de vida incremental, puesto que se enfatizan las demostraciones frecuentes de progreso, verificación y validación del trabajo hasta la fecha. Los requerimientos se asignan a varios elementos de la arquitectura de software, y esta se divide en una secuencia priorizada de construcciones. Cada compilación añade nuevas capacidades al producto en crecimiento incremental. En cada iteración se obtiene una versión i del proyecto. El proceso de desarrollo finaliza cuando la versión N (la versión final) es verificada, validada, demostrada y aceptada por el cliente [27].

# Fases de desarrollo del proyecto

## Análisis e investigación

Este proyecto incluye una metodología enfocada a la investigación, y en la implementación del sistema de detección se incluye la metodología de ingeniería de software. En la primera fase se realizó toda la investigación anterior al desarrollo del proyecto, es decir, es la base de todas las herramientas usadas, los ambientes de desarrollo, tecnologías, *dataset* de malware, etc.

La investigación comenzó con el marco teórico y el estado del arte, en esta investigación se analizó los distintos trabajos realizados y los *frameworks* propuestos. Además, en ésta parte se identificaron posibles *datasets* para cumplir con el primer objetivo, al final para el caso de aplicaciones maliciosas se decidió usar el *dataset* (de aproximadamente 650 aplicaciones) proporcionado por el proyecto *Drebin* [29] y para el caso de las aplicaciones benignas se decidió usar el *dataset* de la universidad de Columbia [30] para aproximadamente el 80% de dichas aplicaciones, y para el 20% restante se usaron aplicaciones obtenidas de la página *APKPure* [31] para un total de aproximadamente 530 aplicaciones benignas. Para el caso de las aplicaciones benignas se utilizó la herramienta online *VirusTotal* [32] para verificar que en efecto no fueran aplicaciones que pudieran contener algún tipo de malware. Luego se comenzó con la investigación de la metodología para el desarrollo del sistema propuesto.

Inicialmente se evaluaron distintos *frameworks* creados para el análisis dinámico de aplicaciones Android, tales como CuckooDroid, Androl4b, Android Security Evaluation Framework y Mobile Security Framework (MobSF). Al final de la investigación de estos *frameworks* se concluyó que por motivos de control, complejidad y la limitación de recursos no se podría usar ninguna de las herramientas estudiadas y por tanto se decidió el uso de un AVD (Android Virtual Device). La descripción de las herramientas usadas se encuentra en uno de los entregables del proyecto [33].

## Diseño

En ésta etapa, en cada uno de los incrementos, se añadieron elementos al diagrama de *deployment* (ver sección de Anexos). Debido a que el primer incremento estuvo enfocado a la generación de tráfico, el diseño solamente estuvo conformado por el dispositivo AVD y algunos módulos necesarios para dicho proceso. En el incremento 2 se añadieron componentes nuevos, tales como el componente de captura de tráfico y el componente de simulación de acciones del usuario de la aplicación en ejecución, y en el tercer incremento se incluyó el módulo de clasificación de aplicaciones.

## Implementación

*Etapa inicial: preparación del ambiente de ejecución de aplicaciones Android*

Para ésta etapa se usó el IDE de Android llamado Android Studio (versión 3.0), ésta herramienta contiene un módulo que permite hacer uso de un emulador Android para la gestión de aplicaciones, bien sea para aplicaciones en desarrollo o aplicaciones descargadas.

Una vez se creó el emulador, se configuraron y realizaron pruebas con aplicaciones, y se investigaron distintos tipos de comandos para realizar tareas por consola usando el lenguaje de programación Python (versión 2.7) y librerías tales como *os* y *subprocess*.

*Etapa 2: Generación de tráfico*

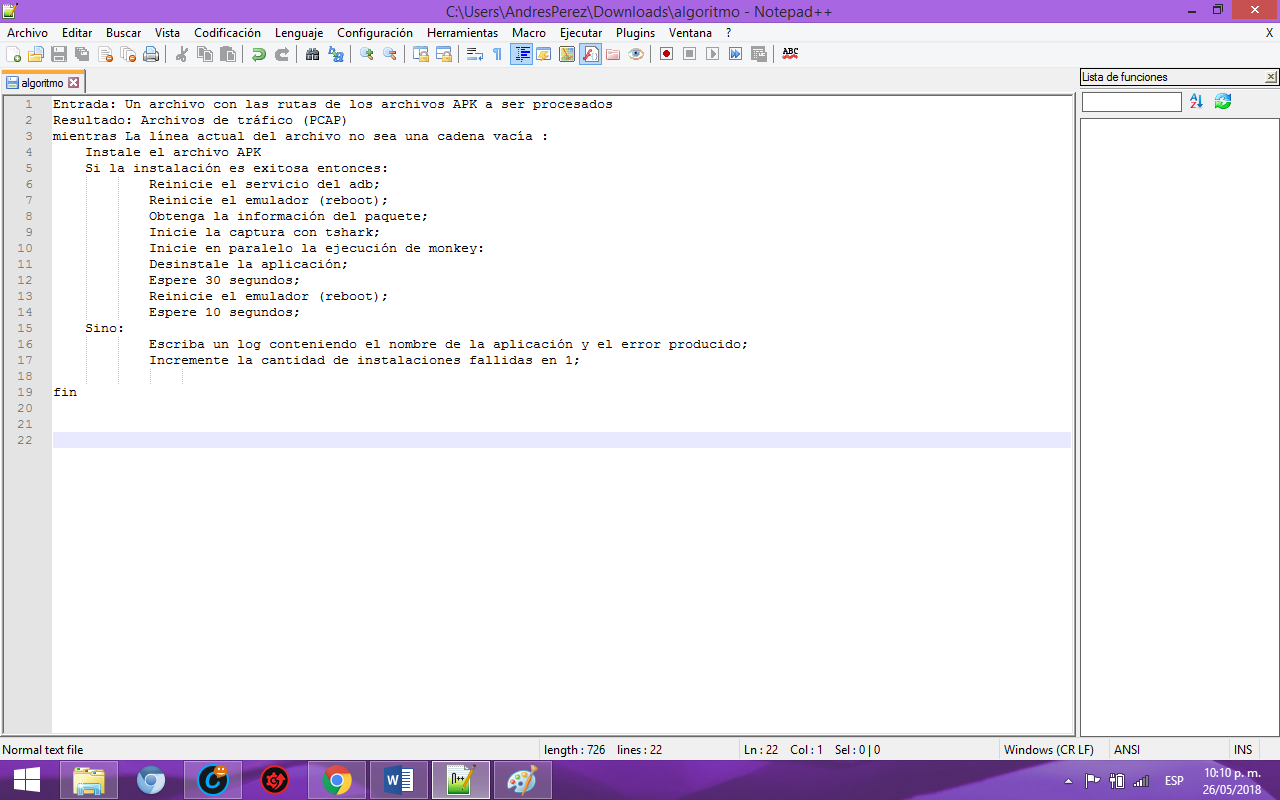
Para la generación de tráfico se usó un script creado en Python para realizar la instalación, búsqueda de paquetes y desinstalación de aplicaciones, encendido y apagado del emulador, ejecución de la herramienta Monkey y captura de tráfico mediante TShark. El tiempo que permanece una aplicación instalada e interactuando con Monkey es de 5 minutos, y siempre que se desea instalar una aplicación, y posteriormente a la desinstalación de la misma se hace un reinicio (reboot) del emulador, ésta metodología es recomendada por académicos como Dong Cao, Shanshan Wang y Qun Li [25], entre otros, quienes ya han realizado y presentado estudios de análisis de tráfico Web. La figura 7 muestra el proceso que se lleva a cabo para la generación de tráfico:

Figura 7: Algoritmo de generación de tráfico

A continuación, se proporciona una descripción de algunas de las herramientas mencionadas anteriormente:

* **AVD (Android Virtual Device):** un AVD contiene un perfil de hardware, una imagen de sistema, un área de almacenamiento y una máscara, entre otras propiedades. Este dispositivo está cargado previamente con determinados perfiles de hardware, tales como dispositivos de teléfono Nexus, y se pueden definir e importar perfiles de hardware cuando sea necesario. Para este proyecto se usó el perfil de un teléfono Google Pixel 2 XL con Android versión 5.1 (Lollipop) ya que según Andro4all es la más usada hasta el momento por los dispositivos móviles.
* **Monkey**: es una herramienta de línea de comando que puede ser utilizada en una instancia del emulador o en un dispositivo real, esta herramienta envía flujos de eventos aleatorios al sistema, con el fin de simular la interacción de las aplicaciones con el usuario.
* **TShark**: es un analizador de protocolos de red. Permite capturar datos de paquetes desde una red en vivo, o leer paquetes desde un archivo de captura previamente guardado, ya sea imprimiendo una forma decodificada de esos paquetes a la salida estándar o escribiendo los paquetes a un archivo. El formato de archivo de captura nativo de TShark es el formato pcap, que es también el formato utilizado por tcpdump y varias otras herramientas.

Una de las principales dificultades a la hora de instalar las aplicaciones fue un error de configuración llamado INSTALL\_FAILED\_NO\_MATCHING\_ABIS, el cual impidió la instalación de aproximadamente 150 aplicaciones maliciosas y 12 benignas. Según algunas opiniones en foros y páginas como Stack Overflow, este error ocurre cuando se intenta instalar una aplicación que no es compatible con la arquitectura de un emulador. Existen muchas aplicaciones compiladas para ser usadas en arquitectura ARMv7. Si se intenta instalarlas en un emulador con arquitectura Intel (o el caso contrario) no funcionará, es decir, que la arquitectura de la compilación de las aplicaciones debe ser igual a la del emulador, o de lo contrario, la aplicación no se podrá instalar. Cabe aclarar que dicho error sucedió tanto con aplicaciones benignas como maliciosas, y por tanto, el dataset de aplicaciones tuvo una reducción de más del 15% de las aplicaciones que se tenían en el *dataset* original.

Para realizar todo este proceso se utilizó el dataset del proyecto *Drebin,* que es libre para uso académico y contiene aplicaciones Android (APK) de tipo malicioso (malware), el tamaño del dataset es de más de 500 aplicaciones provenientes de distintas familias de malware que incluyen troyanos, spyware, adware, entre otros. El proceso de generación y captura de tráfico tarda aproximadamente 8 minutos por aplicación y el resultado final de todo este proceso fueron 1053 archivos pcap que se obtuvieron con TShark (518 benignos y 535 maliciosos), a partir de estos archivos se puede concluir que los objetivos 1 y parte del 2 fueron cumplidos.

*Etapa 3: características y clasificación*

Para esta etapa se creó un script en Python, que hace uso de la librería *Pyshark* para la extracción de características del tráfico Web. Haciendo un estudio del estado del arte de los trabajos relacionados con el proyecto tales como *Malware Detection Using Network Traffic Analysis in Android Based Mobile Devices*, *DroidCollector: A High Performance Framework for High Quality Android Traffic Collection, A First Look at Android Malware Traffic in First Few Minutes*, entre otros, se consideraron las siguientes características:

* Cantidad de paquetes TCP
* Número de puertos distintos a los puertos TCP
* Número de IP externas
* Cantidad de bytes de la capa de aplicación
* Número de paquetes UDP
* Número de paquetes originados
* Número de paquetes recibidos
* Cantidad de bytes enviados por la aplicación
* Cantidad de bytes recibidos por la aplicación
* Número de paquetes HTTP
* Número de consultas DNS

Algunos de los algoritmos usados para la extracción de características están basados en el proyecto de grado anterior llamado *Sistema open source para la detección de páginas Web maliciosas* [34], con ellos fue posible obtener algunas de las características mencionadas anteriormente.

A partir de un archivo CSV (Comma-Separated Values) generado por el script, se procedió a evaluar los distintos árboles de decisión haciendo uso de la herramienta WEKA (Waikato Environment for Knowledge Analysis) [35], que es un entorno para la experimentación de análisis de datos que permite aplicar, analizar y evaluar las técnicas más relevantes de análisis de datos, principalmente las provenientes de *machine learning*, sobre cualquier conjunto de datos del usuario. Para ello únicamente se requiere que los datos a analizar se almacenen en un formato conocido como ARFF (Attribute-Relation File Format). WEKA se distribuye como software libre desarrollado en Java. Está constituido por una serie de paquetes de código abierto con diferentes técnicas de preprocesado, clasificación, agrupamiento, asociación, y visualización, así como facilidades para su aplicación y análisis de prestaciones cuando son aplicadas a los datos de entrada seleccionados.

## Pruebas y validación

Una vez terminado cada módulo del sistema las pruebas consistieron en un tipo de prueba llamado Smoke Test o prueba de humo, en este tipo de pruebas se corrobora el correcto funcionamiento del módulo haciendo pruebas de funcionalidad. Para este sistema se hicieron las siguientes pruebas de humo:

* Una prueba de humo para el módulo de generación y captura de tráfico, en la que se verificó la correcta instalación, interacción, captura de datos y desinstalación de la aplicación, haciendo pruebas con software compatible y no compatible y asegurándose que los errores que pudiesen surgir quedaran registrados en un archivo de log.
* Una prueba para la extracción de características, esta prueba consistió en la verificación de los datos que se tomaban en las capturas y compararlas con los archivos PCAP originales y validar que los datos extraídos fueran consistentes con los de los archivos.

En la etapa de validación se pudo comprobar que los datos propios obtenidos en la universidad Icesi eran muy distintos a los ofrecidos por Drebin, ya que existe la posibilidad de que en el proceso de generación de tráfico no usaran herramientas de simulación de las acciones de un usuario en las aplicaciones, a diferencia de éste estudio. Esto claramente tiene una incidencia en la cantidad de tráfico recolectado. En esta fase se realizó el objetivo número 4, para esto se llevó a cabo una comparación con el estudio realizado en la universidad Icesi llamado *Features to detect Android malicious applications* [36] que consistió en un análisis de características tanto a nivel de red como de permisos en aplicaciones Android, y cuyos resultados y la comparación con éstos serán presentados en la siguiente sección.

## Contribución y resultados del proyecto de grado

## Resultados

**Fase 1: Requerimientos generales del sistema**

R1: El sistema debe permitir la clasificación de aplicaciones Android (APK) como maliciosas o benignas.

R2: El sistema debe generar, capturar y almacenar datos de tráfico Web de aplicaciones benignas y maliciosas provenientes de un dataset de prueba.

R3: El sistema debe permitir la instalación y desinstalación de aplicaciones Android.

R4: El sistema debe contar con un módulo de simulación para las acciones realizadas por un usuario.

R4: El sistema debe permitir la automatización del requerimiento dos (R2).

R5: El sistema debe procesar datos de captura de tráfico (PCAP) y realizar la extracción de características en un archivo CSV.

**Fase 2: Particionamiento final del sistema (análisis Dorfman paso 1)**

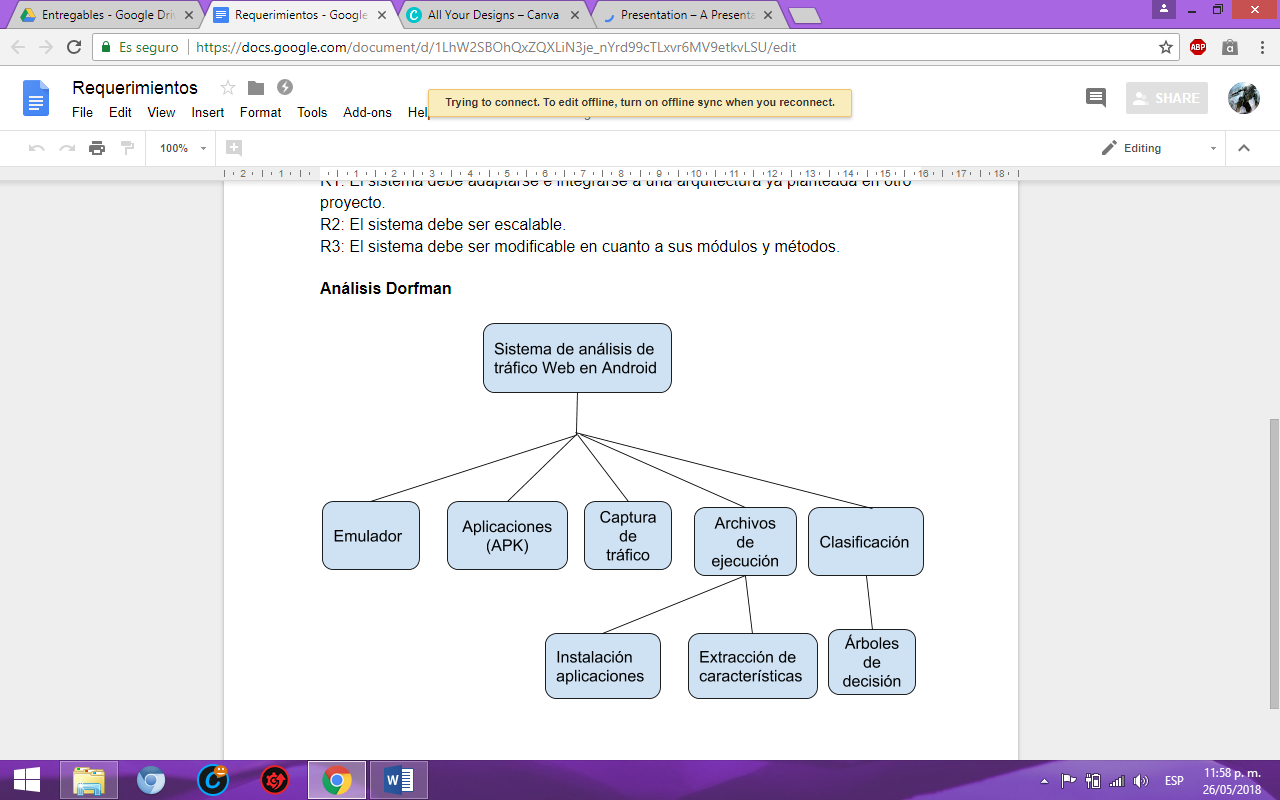


Figura 8: Particionamiento del sistema propuesto.

**Con respecto a la figura 8, el subsistema emulador es la entidad que representa el entorno emulado donde las aplicaciones van a ser ejecutadas, el subsistema aplicaciones se refiere al dataset de aplicaciones tanto benignas como maliciosas, el subsistema de captura de tráfico hace referencia al proceso de captura y generación de tráfico y a los archivos que allí se generan, los subsistemas que se desprenden de “archivos de ejecución” son los responsables tanto de la automatización de tareas a la hora de generar tráfico como de obtener las características deseadas de dicho tráfico, y por último, el subsistema de arboles de decisión que deriva de “clasificación” es el encargado de la clasificación de las aplicaciones como maliciosas o benignas.**

**Fase 3: Asignación de requerimientos a subsistemas (análisis Dorfman paso 2)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Emulador | Aplicaciones | Captura tráfico | Instalación  aplicaciones | Extracción  características | Árboles de decisión |
| R1 |  |  |  |  |  | X |
| R2 |  |  | X |  |  |  |
| R3 | X | X |  |  |  |  |
| R4 | X |  |  | X |  |  |
| R5 |  |  |  |  | X |  |

Tabla 2: Asociación de los requerimientos a los distintos subsistemas.

**Fase 4: Subespecificación (análisis Dorfman paso 3)**

R1\_arbolDes\_1: El sistema debe evaluar 4 distintos tipos de árboles de decisión, utilizando la herramienta WEKA.

R2\_capturaTrafico1: El sistema debe capturar el tráfico generado por cada una de las aplicaciones tanto benignas como maliciosas, haciendo uso de filtros por la IP del dispositivo.

R2\_captura\_trafico2: El sistema debe almacenar los datos de las capturas de tráfico, al igual que los logs de errores que puedan surgir.

R3\_emulador1: El sistema debe hacer uso de la consola de comandos para la instalación y desinstalación no visual (interfaz gráfica) de las aplicaciones.

R3\_aplicaciones1: El sistema debe descartar las aplicaciones que no sean compatibles con el emulador y dejar un registro de dichas aplicaciones en un archivo de texto plano.

R4\_emulador1: El sistema, por medio de consola de comandos Linux, debe permitir el encendido, apagado, reinicio, reboot y suspensión de los servicios del emulador Android.

R4\_instalacionAplicaciones1: El script de automatización debe automatizar los procesos de instalación y desinstalación de aplicaciones, generación de tráfico, gestión del emulador y del simulador de movimientos.

R4\_instalacionAplicaciones2: El script debe registrar los errores surgidos en cualquier proceso en un archivo de log en formato txt.

R5\_extraccionCaracteristicas1: El sistema debe contar con un módulo de extracción de características ya definidas en la fase de investigación y generar un archivo CSV y un archivo txt de log para los errores surgidos.

**Fase 5: Diseño**

A continuación, la figura 9 muestra una vista general del sistema propuesto. Muestra el proceso desde que el archivo APK se instala en el entorno emulado, hasta que llega a la fase de clasificación. En la sección de anexos se puede observar el diagrama de deployment del sistema.



Figura 9: Vista general del sistema propuesto.

**Fase 6: Implementación**

Los resultados de esta fase se pueden observar en el repositorio de *Github* donde se encuentran los archivos de generación, captura y extracción de características, a continuación se proporciona dicho enlace: [https://github.com/andres-180/Traffic-algorithm](about:blank)

**Fase 7: Validación**

Para este proceso se tomaron los datos del estudio realizado por [36] para realizar la comparación, tanto en el análisis descriptivo como en el análisis de los métodos de clasificación.

**Fase 7.1: Análisis descriptivo de los resultados**

En este aspecto, primero se muestra en la figura 7 el promedio de las características obtenidas, tanto de las aplicaciones maliciosas como de las benignas del proyecto propuesto.

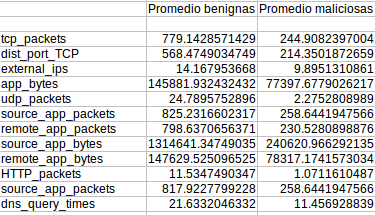


Tabla 3: resultados del análisis estadístico del proyecto.

Basándose en estos resultados se puede concluir que, al igual que el experimento realizado por [37], se evidencia en las capturas de tráfico que el promedio del tamaño de los paquetes tanto enviados como recibidos en el caso del malware es menor que las aplicaciones benignas, lo mismo sucede con los bytes tanto enviados como recibidos.

Una de las características más notoria fue el envío de paquetes TCP, ya que en las aplicaciones benignas esta cantidad fue más de tres veces mayor al de las aplicaciones maliciosas (779 versus 244). Pese a que la cantidad de llamados o consultas DNS en el caso de las aplicaciones benignas es casi el doble que el de las maliciosas, al hacer una inspección más detallada se puede observar que para cada aplicación la cantidad de consultas fue un factor muy variable. Para el caso del protocolo HTTP hubo una diferencia notoria a nivel individual entre las aplicaciones maliciosas y las benignas, ya que si bien la tabla nos muestra que el promedio de las aplicaciones maliciosas es 11 veces menor en comparación, al observar más detalladamente se puede notar que sólo en 9 casos el protocolo HTTP superó los 4 paquetes mientras que en el caso de las aplicaciones benignas este protocolo fue usado en 108 paquetes.

Continuando con los resultados de la validación, en la tabla 4 se pueden observar los datos obtenidos del estudio objetivo de la comparación del proyecto [36], y se encuentra que para este estudio la cantidad de tráfico que se generó fue menor que la del proyecto propuesto. Esto puede ser debido a que no se emplearon herramientas de simulación de movimientos del usuario.



Tabla 4: Promedios de las características obtenidas del estudio [36] realizado en i2t (Icesi).

Cabe resaltar que la característica relacionada con la cantidad de paquetes HTTP no aparece en la tabla anterior, y es debido a que dicha característica es diferenciadora en el proyecto propuesto y en el estudio anterior no se tuvo en cuenta. Por último, al realizar un análisis estadístico, podemos observar que aquí también se cumplen algunos de los hallazgos del proyecto, como lo es la cantidad de tráfico generado por las aplicaciones maliciosas versus las benignas, la cantidad de paquetes TCP, los *Bytes* enviados y recibidos, entre otras.

**Fase 2: Análisis de resultados usando el método de clasificación**

Con 4 clasificadores (J48, LMT, Random Forest y Random Tree) y con el dataset compuesto por 535 aplicaciones maliciosas y 518 aplicaciones benignas, se procedió a evaluar los 4 algoritmos de clasificación, utilizando parámetros generales como lo son la validación cruzada con un k (número de iteraciones) igual a 12, es decir que en cada iteración los datos de muestra se dividieron en un subconjunto de entrenamiento y otro de datos de prueba (más pequeño que el de entrenamiento). Los resultados de este proceso se presentan en la siguiente tabla:

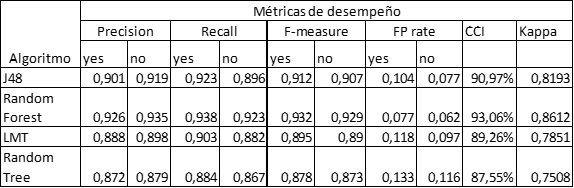


Tabla 5: resultados en el desempeño individual de los 4 clasificadores.

Podemos observar que el mejor modelo es Random Forest, el cual contó con un desempeño de clasificación de 93,06% de CCI (Correctly Classified Instances), con una capacidad de 92,6% en identificación de comunicaciones benignas, un indicador Kappa (coherencia o concordancia entre las variables) de 0,86 y finalmente con una baja tasa de falsos positivos (FP rate).

Cabe resaltar que los distintos algoritmos tenían como condición inicial en el árbol de decisión distintas características. Por ejemplo, el algoritmo J48 tenía como primera condición la cantidad de IP externas, al igual que el algoritmo LMT, pero en el caso de Random Tree la característica principal fue el número de consultas DNS.

En cuanto a la comparación con el estudio anterior, el método de Random Forest fue el que mejor desempeño tuvo, con una precisión de 0,93 para los casos de aplicaciones benignas, que es muy cercano al de este proyecto, pero el kappa del estudio es de sólo 0,74 que es menor al de éste proyecto, esto indica que hay una mayor concordancia entre las variables del proyecto ya que nuestro kappa es más cercano a 1. Por último, se puede concluir con respecto al estudio realizado y a este proyecto, que Random Forest es el mejor modelo de predicción a nivel de red, y aunque el estudio anterior tiene muy buenos resultados, el proyecto propuesto le añade algunas mejoras que han ayudado a obtener algunos resultados aún más interesantes, tales como la cantidad de tráfico y la característica del tráfico HTTP.

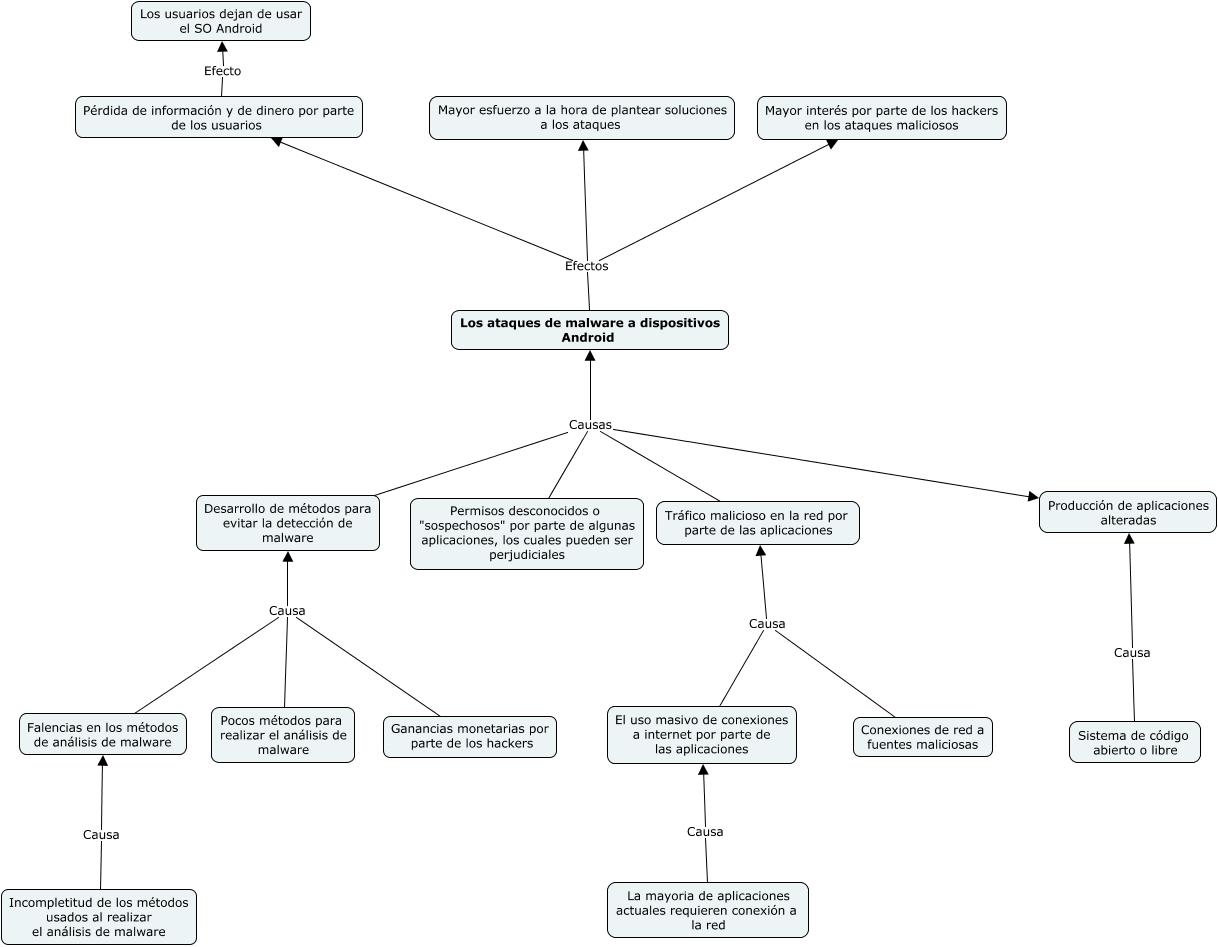
## Conclusiones y trabajo a futuro

Basados en los resultados del análisis descriptivo, podemos concluir que en las características seleccionadas, las aplicaciones maliciosas presentan un porcentaje muy bajo de tráfico a diferencia de las aplicaciones benignas. Esto puede tener explicaciones tales como lo son el hecho de que la cantidad de información que debe ser consultada por algunas de estas aplicaciones es muy recurrente, como por ejemplo el caso de aplicaciones sociales, juegos, aplicaciones de reproducción de contenido, etc. En el caso de la clasificación, se puede concluir que es posible utilizar algoritmos de clasificación para la detección de aplicaciones maliciosas basados en algunas características del tráfico Web. Los resultados muestran que los mejores modelos fueron Random Forest y J48, siendo el primero el más eficiente. En lo personal y profesional puedo concluir que éste proyecto de grado me ayudó mucho a comprender algunos conceptos de ciberseguridad, programación en Python, procesos en sistemas operativos, Android, entre otros aspectos que claramente pueden ser aplicados en la práctica y en la vida cotidiana. Aunque no fue un proyecto fácil, debo decir que el resultado fue muy bueno y espero que este proyecto se vuelva a retomar y se le apliquen algunas mejoras que se mencionaran en el trabajo a futuro.

En cuanto al trabajo a futuro, aún se pueden mejorar muchos aspectos como lo son el seguimiento de los sitios a los cuales las aplicaciones maliciosas acceden. Ya es de mucho interés conocer los sitios en Internet a los cuales se hacían las conexiones y así obtener otro tipo de resultados interesantes. Otro de los aspectos a mejorar es la emulación de las acciones del usuario que se usaron para cada aplicación, ya que pese a que la herramienta Monkey hace un buen trabajo, en algunas ocasiones no era muy efectiva y eso disminuye la cantidad de tráfico generado por algunas aplicaciones (en especial las que requieren un login o en las que se requieren datos reales). Por último, existen estudios [38] que demuestran que algunas aplicaciones pueden detectar entornos emulados, y por tanto, estas aplicaciones restringen sus actividades, haciendo que trabajar en estos entornos sea muy distinto del entorno real, y por tanto, se propone llevar a cabo el mismo estudio, pero con dispositivos reales, y realizar una comparación para detectar las diferencias entre ambos ambientes.

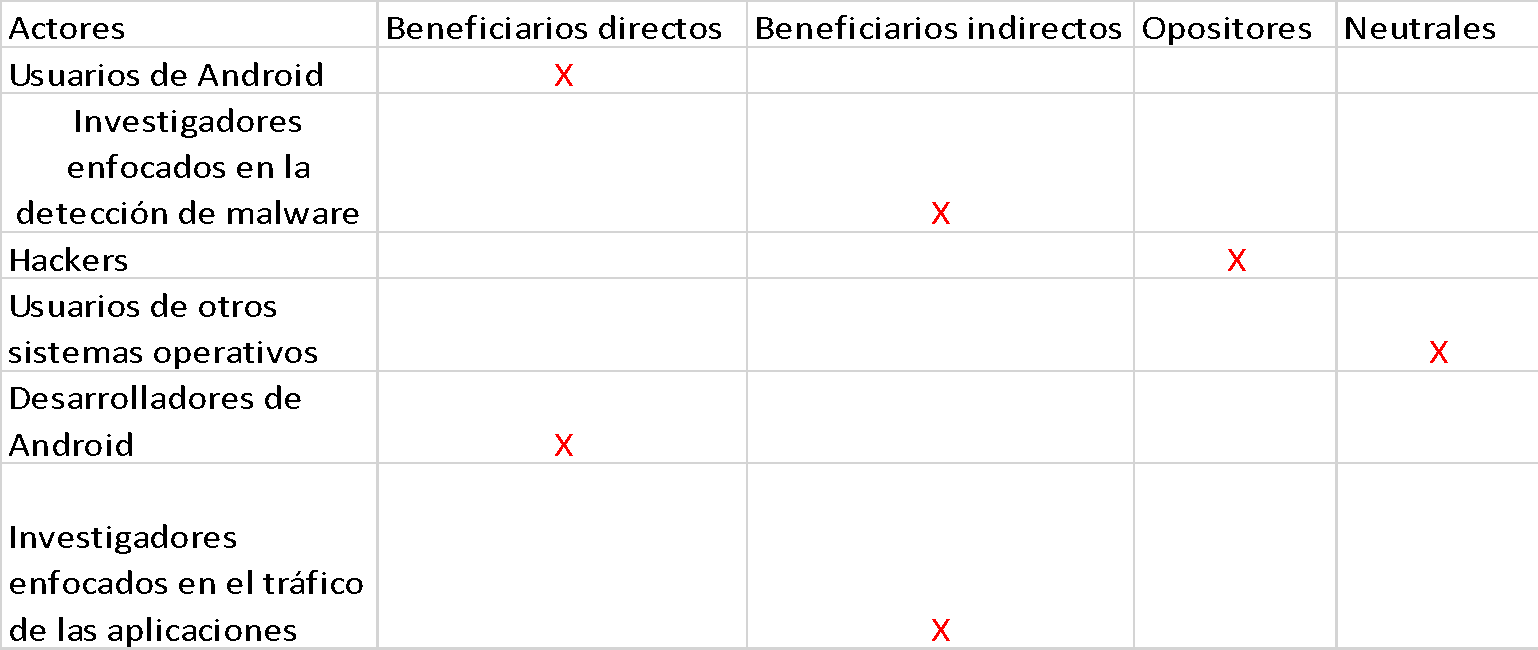
# Anexos

## Árbol de problemas



## Árbol de objetivosC:\Users\1143866754\Downloads\PDG_arbol_de_objetivos.jpg

## Análisis de participación

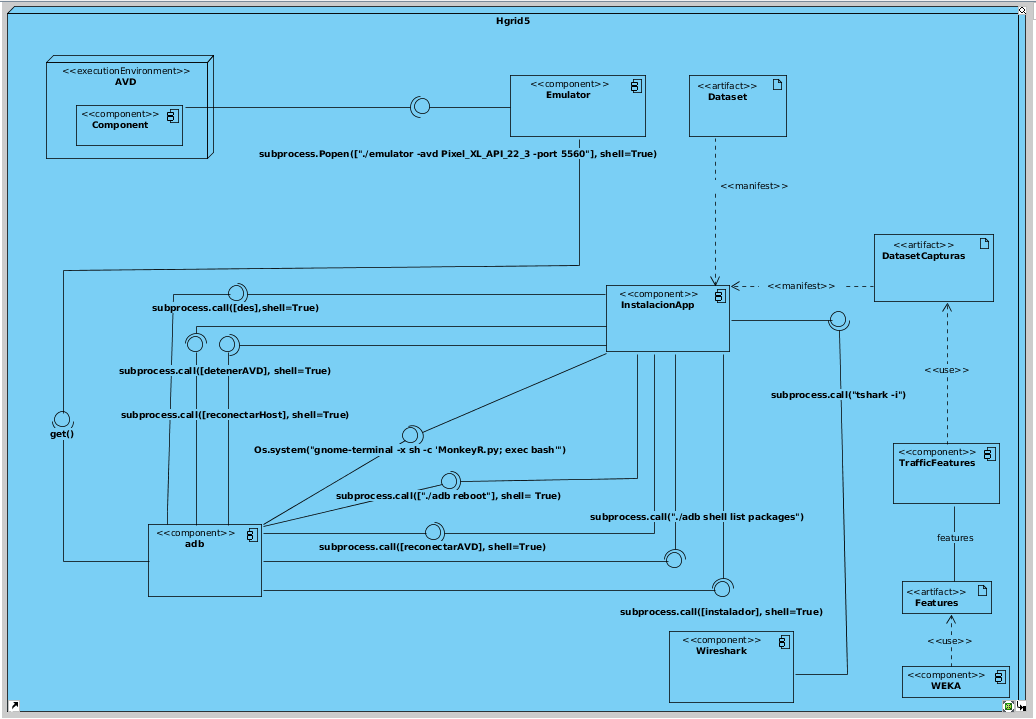


# 

# 

# 

**Diagrama de deployment del sistema propuesto**



# 

# Referencias bibliográficas

[1] Dolz, D., Parra, G. (2006). Ofuscadores de Código Intermedio. Reporte Preliminar. Disponible en*:* http://sedici.unlp.edu.ar/handle/10915/20866

[2] Carroll, A., Heiser, G. (2010). An Analysis of Power Consumption in a Smartphone. Disponible en: https://www.usenix.org/legacy/event/usenix10/tech/full\_papers/Carroll.pdf

[3] Prowse, D. L. (2011). CompTIA Security+. Pearson Education, Indianapolis, IN, 2nd ed edition.

[4] The security ledger. (2017). Android Malware Doubled in 2016, Adding to Mobile Malware Problem. [*Online*]. Disponible en: [https://securityledger.com/2017/03/android-malware-doubled-in-2016-adding-to-mobile-malware-problem/](about:blank)

[5] Gartner. (2017). Gartner Says Worldwide Sales of Smartphones Grew 7 Percent in the Fourth Quarter of 2016. [*online*]. Disponible en: https://www.gartner.com/newsroom/id/3609817

[5.2] Statista. "Global smartphone sales to end users from 1st quarter 2009 to 1st quarter 2017, by operating system (in million units)," 2017. [Online]. Disponible en: https://www.statista.com/statistics/266219/global-smartphone-sales-since-1st-quarter-2009-by-operating-system/.

[6] RT. (2014). Infografía: ¿Cuánto ganan los 'hackers'? Kaspersky explica el lucro de la piratería. [*Online*]. Disponible en: [https://actualidad.rt.com/actualidad/view/138751-ganan-hackers-expertos-kaspersky-explican](about:blank)

[7] Tam, K., Feizollah, A., Anuar, B., Salleh, R., & Cavallaro, L. (2017). The Evolution of Android Malware and Android Analysis Techniques. *Article*, *49*(76). https://doi.org/10.1145/3017427. P 10-15

[8] Tam, K., Feizollah, A., Anuar, B., Salleh, R., & Cavallaro, L. (2017). The Evolution of Android Malware and Android Analysis Techniques. *Article*, *49*(76). https://doi.org/10.1145/3017427. P 15-16

[9] Carlon, K., Mobile to represent 75% of internet traffic next year. [*Online*]. Disponible en: [http://www.androidauthority.com/mobile-internet-usage-predictions-725382/](about:blank)

[10] Chen, Z., Han, H., Yan, Q., Yang, B., Peng, L., Zhang, L., & Li, J. (2015). A first look at android malware traffic in first few minutes. In *Proceedings - 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2015*. https://doi.org/10.1109/Trustcom.2015.376. P 8.

[11] Arora, A., Garg, shree., peddoju, sateesh,. (2014). Malware Detection Using Network Traffic Analysis in Android Based Mobile Devices. Article, Disponible en: http://ieeexplore.ieee.org/document/6982893/?reload=true

[12] Santana, A. (2011). Una infraestructura de comunicaciones cliente-servidor para dispositivos móviles. *Tesis Maestro en ciencias de la ingeniería de cómputo*. Instituto Politécnico Nacional, p 127.

[13] Elenkov, N. “Android's security model,” in *Android security internals: An in-depth guide to Android's security architecture*, No Starch Press, Inc. 2015, p 1- 11.

[14] Elenkov, N. “Android's security model,” in *Android security internals: An in-depth guide to Android's security architecture*, No Starch Press, Inc. 2015, p 12.

[15] Tomas, J. *El Gran Libro de Android*. Alfaomega Grupo Editor, S.A. de C.V. México: 2012. p. 28.

[16] Android Developer. Manifest of the app. [*Online*]. Disponible en: [https://developer.android.com/guide/topics/manifest/manifest-intro.html?hl=es-419](about:blank).

[17] Jaramillo, O., Eraso, S, “Seguridad en dispositivos móviles Android”. “Proyecto de grado”, seguridad informática, Universidad nacional abierta y a distancia, Bogotá, 2015.

[18] Android Developer. Create and Manage Virtual Devices. [*Online*]. Disponible en: [https://developer.android.com/studio/run/managing-avds.html](about:blank)

[19] KasperskyLab. Los tipos de malware. [*Online*]. Disponible en: [https://support.kaspersky.com/sp/viruses/general/614](about:blank)

[20] Bhrouz, A. (2007). “Transmisión de datos y redes de comunicaciones”. 4 edición.

[21] Bhrouz, A. (2007). ”TRANSMISIÓN DE DATOS,” en *Transmisión de datos y redes de comunicaciones*. 4 edición, p 4.

[21] Drake, J., Fora, P., Lanier, Z., Mulliner, C., Ridley, S., Wicherski, G. “Understanding Android’s Attack Surface,” in *Android™ Hacker’s Handbook*, No Starch Press, Inc. 2015, p 129- 160.

[22] Malik, J., Kaushal, R. (2016). *CREDROID: Android Malware Detection by Network Traffic Analysis*. https://www.researchgate.net/publication/304995756\_CREDROID\_Android\_malware\_detection\_by\_network\_traffic\_analysis.

[23] Anshul, A., Shree, G., Sateesh, P. (2014). *Malware Detection Using Network Traffic Analysis in Android Based Mobile Devices*. http://ieeexplore.ieee.org/document/6982893/?reload=true

[24] Yibing, Z., Zhi, X., Bing, M., Li, X. (2013). *DroidAlarm: An all-sided static analysis tool for Android privilege-escalation malware*. https://www.researchgate.net/publication/262403167\_DroidAlarm\_An\_all-sided\_static\_analysis\_tool\_for\_Android\_privilege-escalation\_malware

[25] Chen, Z., Han, H., Yan, Q., Yang, B., Peng, L., Zhang, L., & Li, J. (2015). A first look at android malware traffic in first few minutes. In *Proceedings - 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2015*. https://doi.org/10.1109/Trustcom.2015.376.

[26] Azure. ¿Qué es middleware?. [*Online*]. Disponible en: [https://azure.microsoft.com/es-es/overview/what-is-middleware/](about:blank)

[27] R.E. Fairley, Managing and Leading Software Projects, Wiley-IEEE Computer Society Press, 2009.

[28] Pineda, S. (2009). Comparación de Árboles de Regresión y Clasificación y regresión logística. *Trabajo presentado como requisito para optar al título de Magíster en Estadística.* Universidad Nacional de Colombia Sede Medellín. Disponible en: [http://www.bdigital.unal.edu.co/671/1/42694070\_2009.pdf](about:blank)

[29] Technische Universität Braunschweig. (2016). *The Drebin Dataset*. Disponible en: [https://www.sec.cs.tu-bs.de/~danarp/drebin/](about:blank)

[30] Columbia University Department of Computer Science. (2015). *PlayDrone: A Measurement Study of Google Play*. Disponible en: [http://systems.cs.columbia.edu/projects/playdrone/](about:blank)

[31] APKPure. (2018). Sitio Web para la descarga de aplicaciones Android. Disponible en: [https://apkpure.com/es/](about:blank)

[32] VirusTotal. (2018). *VirusTotal: Analice archivos y URL sospechosos para detectar tipos de malware y compártelos automáticamente con la comunidad de seguridad.* Disponible en: [https://www.virustotal.com/#/home/upload](about:blank#/home/upload)

[33] Pérez, A. (2018). Herramientas para el análisis dinámico de aplicaciones Android. Entregable de proyecto de grado. Disponible en: [https://docs.google.com/document/d/1kmp8K7jLGl-LciBZdwhUulKT9-mehPblZ6-y3NtHBGM/edit](about:blank)

[34] Peña, M., Quintero, J. (2017). *Sistema open source para la detección de páginas Web maliciosas*. Proyecto de grado. Aún no disponible.

[35] Universidad de Waikato. (2018). Weka 3: Data Mining Software in Java. Proyecto open source. Disponible en: [https://www.cs.waikato.ac.nz/ml/weka/](about:blank)

[36] Urcuqui, C., Delgado, J., Pérez, A., Navarro, A., Díaz, J. (2018). Features to detect Android malicious applications. presentado en IEEE COLOMBIAN CONFERENCE ON COMMUNICATIONS AND COMPUTING (COLCOM 2018).

[37] Aurora, A., Garg, S., & Peddoju, S. K. (2014). Malware detection using network traffic analysis in Android based mobile devices. In next generation mobile apps, services and technologies (NGMAST), 2014 eighth international conference on (pp. 66-71). IEE.

[38] Alzaylaee, M. k., Yerima, S. Y., & Sezer, S. (2017). Emulator vs real phone: Android malware detection using machine learning. In proceedings of the 3rd ACM on International Workshop on security And Privacy Analytics (pp. 65-72). ACM.