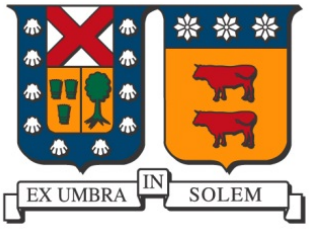
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE




# “IDENTIFICATION OF POTENTIALLY MALICIOUS BEHAVIOR THROUGH INFORMATION FLOW ANALYSIS TECHNIQUE FOR MOBILE APPLICATIONS IN ANDROID OPERATING SYSTEM”

DIEGO BARBOZA BUSTAMANTE

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Felipe Beroíza
Profesor Correferente: Marcello Visconti

Junio - 2024

# DEDICATORIA

En primer lugar, quiero dedicarle este trabajo a mi familia. A mi papá, a mi mamá y a mi hermana, quienes me apoyaron en cada parte del proceso y que sin su aliento esto no hubiera posible. Gracias por aguantarme en mis periodos de estrés, especialmente mis malas caras y mi mal genio. Gracias por darme un empujón cada vez que lo necesité. Este trabajo se lo debo a ellos por sobre todas las personas.

Por otro lado, quiero dedicarle este trabajo a mis amigos. Desde aquellos amigos que me han apoyado desde la época del colegio y que han estado siempre, hasta los amigos que conocí en mi proceso universitario y que definitivamente se ganaron un lugar especial en mi vida. Gracias a los Chatos, a los ANG, a los Jetones, al Grupo Paro y a los Juno.

Por último, quiero dedicarle este trabajo al Jaim. La persona que me invitó a un taller de programación en 4to Medio y que me ayudó a descubrir lo que me apasiona y el profesional que quiero ser. Te llevaré siempre en mis recuerdos. Un abrazo al cielo.

# RESUMEN

**Resumen —** Actualmente, la creciente adopción de los *smartphones* ha llevado a que estos dispositivos almacenen una cantidad cada vez mayor de información personal de los usuarios, que suele transmitirse sin un control riguroso. En particular, Android, siendo el sistema operativo con mayor cuota de mercado, naturalmente es un foco importante de esta problemática, presentando serias fallas de seguridad con respecto a sus contrapartes. En esta línea, el presente estudio plantea una metodología apoyada en técnicas de análisis dinámico basadas en el análisis del tráfico de red de aplicaciones de Android, capaz de detectar los casos en que la información sensible del usuario abandona el dispositivo y bajo qué contexto. Se seleccionaron y analizaron un total de 96 aplicaciones populares de la Play Store y se identificaron los aspectos más relevantes para detectar fugas de información, permitiendo así categorizar aquellas con un mayor nivel de riesgo y resaltar el bajo nivel de privacidad que presentan las aplicaciones en la actualidad.

**Palabras Clave —** Seguridad; Análisis Dinámico; Aplicaciones de Android; Fugas de Información; Privacidad.

# ABSTRACT

***Abstract* —** Nowadays, the growing adoption of smartphones has led to these devices storing an increasing amount of users' personal information, which is often transmitted without rigorous control. In particular, Android, being the operating system with the largest market share, is naturally a major focus of this problem, presenting serious security flaws with respect to its counterparts. In this line, the present study proposes a methodology supported by dynamic analysis techniques based on the analysis of Android application network traffic, capable of detecting the cases in which sensitive user information leaves the device and under what context. A total of 96 popular apps from the Play Store were selected and analyzed and the most relevant aspects for detecting information leaks were identified, thus allowing to categorize those with a higher level of risk and highlighting the low level of privacy that apps currently have.

***Keywords* —** Security; Dynamic Analysis; Android Applications; Information Leaks; Privacy.

# GLOSARIO

A continuación, se listan una serie de conceptos relevantes para facilitar la comprensión del informe.

**API**: Application Programming Interface
**APK**: Android Package Kit / Android Application Package
**DPI**: Deep Packet Inspection
**GKI**: Generic Kernel Image
**HTTP(S)**: Hypertext Transfer Protocol (Secure)
**IP**: Internet Protocol
**IPC**: Inter-Process Communication
**LTS**: Long Time Supported
**MITM**: Man In The Middle
**PII**: Personally Identifiable Information
**RAM**: Random Access Memory
**SSL**: Secure Sockets Layer
**TCP**: Transmission Control Protocol
**TLS**: Transport Layer Security
**UI**: User Interface

# ÍNDICE DE CONTENIDOS

# ÍNDICE DE FIGURAS



# ÍNDICE DE TABLAS

# CAPÍTULO 1
# INTRODUCCIÓN

## 1.1. Motivación y Contexto

En la última década, la tecnología se ha convertido indiscutiblemente en una parte integral de la vida cotidiana de las personas. Desde la aparición de las primeras redes sociales hasta el reciente aumento de popularidad de los asistentes virtuales, la tecnología se ha abierto paso en prácticamente cada aspecto de la vida del usuario común. A medida que ésta se ha vuelto más avanzada y con ello también más accesible, la sociedad ha ido dependiendo cada vez más de ella hasta el punto de volverse una necesidad al momento de realizar hasta las tareas más simples [Shahzad *et al.*, 2022]. A día de hoy y a causa de esto, la vida de las personas está intrínsecamente ligada al ecosistema digital. Mantener contacto y comunicarse con otras personas se ha vuelto cada vez más sencillo gracias a las redes sociales; una considerable parte de las tareas realizadas, tanto en el ámbito laboral como en el académico, se facilitan de manera notable con el apoyo de un ordenador; y en la actualidad, la gran mayoría de trámites y trabajos pueden realizarse perfectamente desde la comodidad del hogar, sin necesidad alguna de acudir a lugares físicos [Powell *et al.*, 2021].

Ahora bien, a pesar de que anteriormente se empleaban una serie de herramientas distintas para llevar a cabo tareas académicas, laborales, o bien, del día a día; en la actualidad, el uso se ha reducido exclusivamente a los denominados *smartphones*, dispositivos que irrefutablemente superan a los demás en términos de funcionalidad y accesibilidad [Ma'azer Al Fawareh y Jusoh, 2017]. De esta manera y con el fin de focalizar el alcance del presente informe, el estudio se focalizará en estos dispositivos en particular, que naturalmente corresponde al más utilizado hoy en día por el usuario promedio. Es más, según las estadísticas expuestas por [O'Dea, 2023], actualmente 9 de cada 10 personas tienen acceso a un dispositivo móvil en todo el mundo. Esto se debe principalmente a que los teléfonos inteligentes han logrado reemplazar casi por completo a otros dispositivos, liberando al usuario de contar con cámaras digitales, relojes, computadores, calculadoras, entre muchos otras herramientas. Esto transforma a los *smartphones* en dispositivos versátiles, indispensables y multifuncionales para los usuarios que deseen optimizar sus actividades cotidianas, además de permitirles estar conectados en todo momento [Saad, 2022]. No obstante, esto también implica disponer de un único dispositivo que concentra prácticamente toda la información personal de los usuarios y, en algunos casos, información sensible de las organizaciones a las que estos usuarios pertenecen. De esta manera, para bien o para mal, los *smartphones* han terminado por convertirse en un puente que conecta nuestra identidad física y nuestra identidad digital.

Inevitablemente, que las personas manejen prácticamente toda su información personal en su celular, los vuelve el blanco perfecto de ciberdelincuentes, personas malintencionadas que buscan hacerse con esa información sensible y usarla a su favor, y que, dado el

contexto actual, han aumentado en cantidad durante los últimos años [Shahzad *et al.*, 2022]. Como bien señala [Tasheva, 2021], eventos como la pandemia fueron los principales causantes de que innumerables organizaciones tuvieran que reinventarse tecnológicamente, integrando nuevas herramientas digitales a sus procesos y su forma de operar. Un cambio tan abrupto, naturalmente requirió que se dedicaran una enorme cantidad de esfuerzos y recursos a la digitalización de procesos y la capacitación del personal para desenvolverse correctamente con ellas. Esto causó que, en muchos casos, la seguridad de los sistemas en primera instancia se viera descuidada, volviéndolos altamente vulnerables a ataques cibernéticos y por lo tanto, creando el escenario perfecto para los ciberdelincuentes.

Ahora bien, el aumento de la ciberdelicuencia no se debe únicamente a la vulnerabilidad de la infraestructura tecnológica de las organizaciones, sino que también una gran parte radica en los usuarios, tanto a los que pertenecen a estas instituciones como a los usuarios comunes y corrientes. Dichos usuarios generalmente no cuentan con un conocimiento robusto de los entornos digitales de los que forman parte, por lo que no son realmente conscientes de la vulnerabilidad de la información que circula digitalmente. [Tasheva, 2021] hace énfasis en una encuesta realizada el año 2021 por *Pew Research Center*, que señaló que el 40 % de pequeñas empresas y el 65 % de medianas empresas habían sufrido ataques cibernéticos en los últimos doce meses. La encuesta destaca como una de las causas principales la falta de cuidado de parte del personal en términos de ciberseguridad, y de la misma manera, [Hourihan, 2022] menciona que actualmente el 95 % de las grandes filtraciones de datos son atribuidas a errores humanos. Es aquí entonces donde surge la gran problemática.

En primer lugar, nos encontramos con usuarios inexpertos que han tenido que adaptarse a la tecnología muy abruptamente, y por otro lado, se observa un incremento en la cantidad de ataques cibernéticos debido a la gran cantidad de información sensible que hoy en día se encuentra en circulación, y que, en muchas ocasiones, no está protegida correctamente. Tanto a nivel organizacional como a nivel de usuario, las fugas de información confidencial puede resultar en una serie de inconvenientes, entre ellos, suplantación de identidad, pérdidas de privacidad, daños de imagen y reputación, pérdida de propiedad intelectual e incluso en pérdidas económicas si es que los delincuentes exigen recompensas por el rescate de la información [The Yale Ledger, 2022]. Ahora, si bien salvaguardar debidamente la información personal es en parte responsabilidad del usuario, en otras ocasiones, puede depender exclusivamente de la robustez de los sistemas empleados. En el contexto de los dispositivos móviles, la seguridad puede atribuirse inicialmente al sistema operativo en uso, el cual puede desempeñar un rol fundamental en la protección de la información. Naturalmente, hay otras variables que pueden influir en el nivel de privacidad general, como las aplicaciones instaladas, las configuraciones, las conexiones de red y las acciones realizadas por el usuario. Ahora bien, la elección del sistema operativo puede tener implicaciones significativas en la exposición a riesgos de seguridad de manera más general. Por ejemplo, Android, cuya cuota de mercado alcanza un 70 %, como indica [Stat Counter, 2023], no sólo es el sistema operativo más utilizado a nivel mundial, sino que también es aquel que suma un mayor número de vulnerabilidades en comparación a otros sistemas operativos móviles según [CVE Details, 2022], estando condicionado, en parte, por su naturaleza *open-source*, implicando que gran parte

del código fuente esté disponible al público. Es más, un estudio realizado por [Rafter, 2022], señala que Android es efectivamente uno de los focos más grandes de ciberdelincuencia en comparación a otros sistemas operativos.

## 1.2. Justificación del Problema

El estudio realizado por [Shahzad *et al.*, 2022] indica que la causa principal de las fugas de información radica en las aplicaciones instaladas en los dispositivos, teniendo una mayor incidencia en el caso de Android. Naturalmente, los usuarios descargan e instalan aplicaciones, otorgándoles permisos sin tomar mayores precauciones, lo que puede resultar en la exposición involuntaria de datos sensibles. Así, aplicaciones maliciosas y/o con vulnerabilidades ponen en riesgo la información gestionada por un *smartphone*. Esta situación puede no solo afectar a nivel personal, sino también perjudicar directamente a las organizaciones a las que pertenecen los usuarios cuando éstos se convierten en víctimas de ciberdelincuentes.

Según [Powell *et al.*, 2021], cerca de un 70 % de las personas no confía en la protección de los datos que se encuentran en línea, agravando la situación aún más cuando se toma en consideración que un 63 % carece de las habilidades necesarias para proteger adecuadamente su información. Por consiguiente, esto sugiere que los usuarios efectivamente tienen inquietudes respecto a la privacidad de su información y de la misma manera, aparentan no tener claridad sobre qué datos pueden estar siendo comprometidos ni los métodos utilizados para comprometerlos. Por ende, es esencial fomentar la educación en ciberseguridad y concientizar a los usuarios, al menos para brindarles una mayor visibilidad de las amenazas que existen en la actualidad y las medidas que pueden tomar para proteger su información personal.

## 1.3. Propuesta de Solución

Con la debida finalidad de solucionar las problemáticas que fueron presentadas, el presente estudio tendrá como principal foco la búsqueda e implementación de una técnica o herramienta que permita detectar posibles fugas de información sensible en dispositivos móviles. Además, el estudio estará orientado específicamente en el comportamiento malicioso proveniente de aplicaciones instaladas en Android, siendo este uno de los mayores focos y principales causas de fugas de información en comparación a otros sistemas operativos móviles. Por lo tanto, se hará especial énfasis en las técnicas de análisis dinámico, es decir, técnicas que permitan analizar el comportamiento aplicaciones en ejecución. En resumen, se propone implementar una metodología completa de análisis de aplicaciones móviles basada en técnicas de análisis dinámico que permita detectar, categorizar y analizar comportamiento potencialmente malicioso, con la debida finalidad de estudiar y revelen una visión más completa de las amenazas que enfrentan los usuarios en términos de privacidad de su información personal.

## 1.4. Objetivos

En función de lo anteriormente expuesto, el presente informe tendrá como finalidad alcanzar los objetivos que se definen a continuación.

### 1.4.1. Objetivo General

Proponer una metodología para el análisis dinámico de flujos de información provenientes de las aplicaciones instaladas en un teléfono móvil, con la finalidad de identificar actividades potencialmente peligrosas o extrañas, contribuyendo así a la detección de fugas de información y a la protección de la privacidad de los usuarios.

### 1.4.2. Objetivos Específicos

- Resumir los aspectos y las etapas más relevantes del análisis dinámico y hacer énfasis en su funcionamiento y aplicaciones.
- Seleccionar una técnica de análisis dinámico que se adecúe al problema y definir una metodología que facilite su implementación.
- Detectar y categorizar las distintas vulneraciones y amenazas detectadas al momento de ejecutar una aplicación, según su comportamiento.
- Documentar y analizar los resultados de las pruebas realizadas, para obtener conclusiones sobre la efectividad de la propuesta metodológica de análisis dinámico de aplicaciones móviles.

## 1.5. Estructura del Informe

Con la debida finalidad de facilitar la comprensión del presente informe, este ha sido dividido en capítulos, donde cada uno buscará presentar una parte específica del estudio. En primer lugar, en el capítulo 2, el **Marco Teórico y Estado del Arte** proporcionarán el contexto sobre el cual se desarrolla el trabajo, destacando principalmente las formas que existen actualmente de combatir amenazas en un dispositivo móvil y haciendo especial hincapié en aquellas técnicas de análisis más relevantes y que más se adecúen a la problemática. A continuación, en el capítulo 3 se presentará la **Propuesta de Solución**, donde se detallará la metodología y enfoque que se seguirá para la implementación de una posible técnica de análisis dinámico, detallando las métricas que permitirán evaluar la propuesta. Luego, el capítulo 4, o bien, la **Implementación**, describirá el proceso en el que la técnica o herramienta implementada será puesta en práctica. El capítulo 5, es decir, el **Análisis de Resultados**, se

enfocará en la interpretación y discusión de los datos obtenidos, precisando tanto los aciertos como las limitaciones de la solución implementada, para luego en el capítulo 6, presentar la **Conclusión**, que busca resumir los principales hallazgos del estudio realizado y proponer posibles mejoras y recomendaciones para trabajos futuros. Cabe destacar que el informe contará con una sección de **Anexos**, que incluirá material relevante que permitirá respaldar el trabajo realizado, como tablas, gráficos, imágenes, entre otros.

# CAPÍTULO 2
# MARCO TEÓRICO Y ESTADO DEL ARTE

## 2.1. Marco Teórico

### 2.1.1. Información Sensible

Actualmente, las aplicaciones de Android, requieren una variedad de identificadores e información, ya sea para funcionar de manera óptima, o bien, para brindarle una experiencia personalizada a sus usuarios. Si bien, éstas requieren datos que no se relacionan directamente con el usuario en algunos casos, como lo serían datos de rendimiento del dispositivo y configuraciones generales, el estudio se focalizará en aquellos que sí aluden a él, ya que, por lo regular, es esta la información que se considera sensible. Para referirse a ella, comúnmente se utiliza el término **PII**, o por su traducción, información de identificación personal. Como su nombre lo indica y como bien lo define [International Organization for Standardization, 2022], esta información hace alusión a cualquier dato que confirme la identidad de una persona, como es el caso de ciertos identificadores, ubicación geográfica, nombres de usuario, correo electrónico, número de teléfono, entre otros. [Ren *et al.*, 2016] plantea una forma de clasificar dichos datos según su naturaleza, quedando agrupados de la siguiente manera:

- **Identificadores del Dispositivo**
  ICCID (*Integrated Circuit Card Identifier*), IMEI (*International Mobile Equipment Identity*) y IMSI (*International Mobile Subscriber Identity*), que permiten identificar, respectivamente, a la tarjeta SIM, al dispositivo móvil y el operador de telefonía. Además, se consideran identificadores más comunes como la dirección MAC, y por otro lado, identificadores propios de Android, como el *ID de Android* y el *ID de Publicidad*.
- **Identificadores del Usuario**
  Nombre, género, fecha de nacimiento, correo electrónico, entre otros.
- **Información de Contacto**
  Números de teléfono y dirección.
- **Ubicación**
  Coordenadas geográficas (longitud y latitud) y código postal.
- **Credenciales**
  Usuarios y contraseñas.

Si bien las categorías presentadas abarcan en cierta medida la información más importante y susceptible a filtrarse de un dispositivo, la antigüedad del estudio causó que cierta

información, especialmente la de tipo biométrica, no fuera tomada en cuenta. Actualmente, datos como las **huellas dactilares**, **escaneos de retina**, **patrones de voz** e **imágenes** en general, se han utilizado cada vez con más frecuencia, sobre todo como mecanismos de autenticación, por lo que es sumamente importante considerarlo para este y futuros estudios que trabajen con PII. De esta manera, la información biométrica quedaría debidamente cubierta por la categoría **Identificadores del Usuario**, o bien, por **Credenciales**, considerando que en la mayoría de casos el usuario la utiliza para autenticarse. Ahora bien, es crucial entender los riesgos asociados con la filtración de cada uno de estos PII. La exposición de datos sensibles no solo compromete la privacidad del usuario, sino que también puede llevar a una serie de peligros, incluyendo usos indebidos de la información personal [The Yale Ledger, 2022].

Tomando en cuenta las distintas categorías presentadas, aquellas que suponen un mayor riesgo en caso de filtrarse, corresponden a los **Identificadores de Usuario**, la **Información de Contacto**, la **Ubicación** y las **Credenciales**. Estos datos en su conjunto son especialmente sensibles, ya que su exposición podría permitir el acceso a cuentas personales, facilitando la suplantación de identidad y estafas. A su vez, las credenciales podrían entregar un acceso directo a una serie de cuentas y servicios importantes del usuario, y por otro lado, las coordenadas geográficas podrían llegar a revelar exactamente dónde se encuentra la persona, o bien los lugares que frecuenta, pudiendo comprometer su integridad física y resultando en posibles casos de acecho, amenazas y robos.

Si bien las categorías recién mencionadas suponen riesgos más directos que los **Identificadores del Dispositivo**, estos últimos pueden llegar a ser igual de peligrosos cuando son recuperados por algún tercero y asociados a otros datos. Como bien mencionan [Reyes *et al.*, 2018] y [Terkki *et al.*, 2017], dichos identificadores son utilizados en su mayoría por servidores de publicidad y analítica para rastrear la actividad de un dispositivo en múltiples aplicaciones y servicios, facilitando el seguimiento del comportamiento del usuario. En un principio los principales identificadores que cumplían esta tarea correspondían al IMEI y al ID de Android. Sin embargo, desde Agosto del 2014, Google prohibió explícitamente el uso del IMEI y el Android ID para fines publicitarios, por lo que se creó el ID de Publicidad en su reemplazo. Para mitigar posibles riesgos asociados este identificador en particular y seguimientos a largo plazo, Google dio la posibilidad de poder restablecer el ID de Publicidad, es decir, cambiar o ”reiniciar” el valor de dicho identificador. No obstante, en el caso de que éste fuera compartido en conjunto con información persistente, los datos aún podrían ser asociados a un usuario específico, causando que los esfuerzos de Google por resguardar la privacidad del usuario no tuvieran efecto alguno.

El ID de Android, el IMEI y los demás identificadores son más persistentes en comparación con el ID de Publicidad. Para modificar el ID de Android, es necesario realizar una restauración de fábrica o cambiar el dispositivo, mientras que el IMEI requiere cambiar de dispositivo [Reyes *et al.*, 2018]. Estas son acciones que los usuarios no realizan con frecuencia, por lo que estos PII en particular tienden a mantenerse constantes. En este contexto, la capacidad de restablecer el ID de Publicidad de manera relativamente sencilla pierde relevancia. Si un servidor tuviera registros previos de datos como el ID de Android o el IMEI,

sería sencillo asociar un nuevo ID de Publicidad con los valores previamente registrados de estos PII.

Ahora bien, la transmisión aislada del ID de Publicidad también conlleva sus riesgos. Si un actor malintencionado logra acceder a dicho identificador, éste podría emular un entorno que utilice el ID para solicitar anuncios personalizados a un servidor de publicidad, permitiendo inferir información basada en la naturaleza de los anuncios. Por ejemplo, el idioma de un anuncio podría indicar la posible procedencia del usuario, o bien, si se reciben anuncios de aerolíneas, podría sugerir que el usuario tiene un viaje programado, entre otras posibles deducciones.

### 2.1.2. Vías de Fugas de Información

En el contexto de posibles vías en las que la información puede abandonar el dispositivo, se identifican los paquetes de Internet como vía principal. Este destaca como el más propenso a fugas de información, no necesariamente debido a su falta de seguridad, sino por la frecuencia con la que las aplicaciones deben intercambiar paquetes para su correcto funcionamiento.

El gran volumen de datos que contienen estos paquetes es significativamente mayor al que mueven otros canales, como los mensajes SMS o las conexiones a través de Bluetooth [Conti *et al.*, 2018]. Además, en cuanto a transparencia, los paquetes de Internet que salen del dispositivo a menudo pasan desapercibidos para el usuario, quien puede no estar al tanto de la información que realmente se está transmitiendo. A diferencia de los SMS, donde el usuario puede acceder fácilmente al historial de mensajes enviados y recibidos, el tráfico de red no ofrece una forma sencilla de visualizarse directamente desde el dispositivo, lo que naturalmente dificulta su control. Por otro lado, en el caso de las comunicaciones a través de Bluetooth, estas suelen ser de corto alcance y se utilizan principalmente para el intercambio de contenido multimedia, mientras que los paquetes de Internet pueden transmitir una amplia gama de información, desde texto hasta archivos multimedia, lo que aumenta el potencial de filtración de datos. Cabe destacar que las aplicaciones modernas también son menos propensas a solicitar permisos para enviar SMS o utilizar el servicio Bluetooth. En cambio, los permisos para acceder a la red son esenciales para el correcto funcionamiento de la mayoría de las aplicaciones en la actualidad, lo que incrementa la probabilidad de que se filtren datos a través de esta vía. El acceso a Internet como requisito básico para la funcionalidad de las aplicaciones provoca intrínsecamente que los usuarios estén más expuestos a sufrir fugas de su información personal.

Ahora bien, para comprender a cabalidad cómo esta información abandona el dispositivo, se debe entender fundamentalmente cómo funcionan e interactúan los distintos componentes que conforman la arquitectura de Android. Es por esto que en la siguiente sección se explorarán aquellos componentes involucrados en las fugas de información, identificando canales potencialmente vulnerables y posibles causantes de las filtraciones.

### 2.1.3. Arquitectura de Android

A continuación, se presenta un diagrama elaborado por [Android Developers, 2024a] que resume los principales componentes que conforman la plataforma Android y que se detallan en la figura 1.

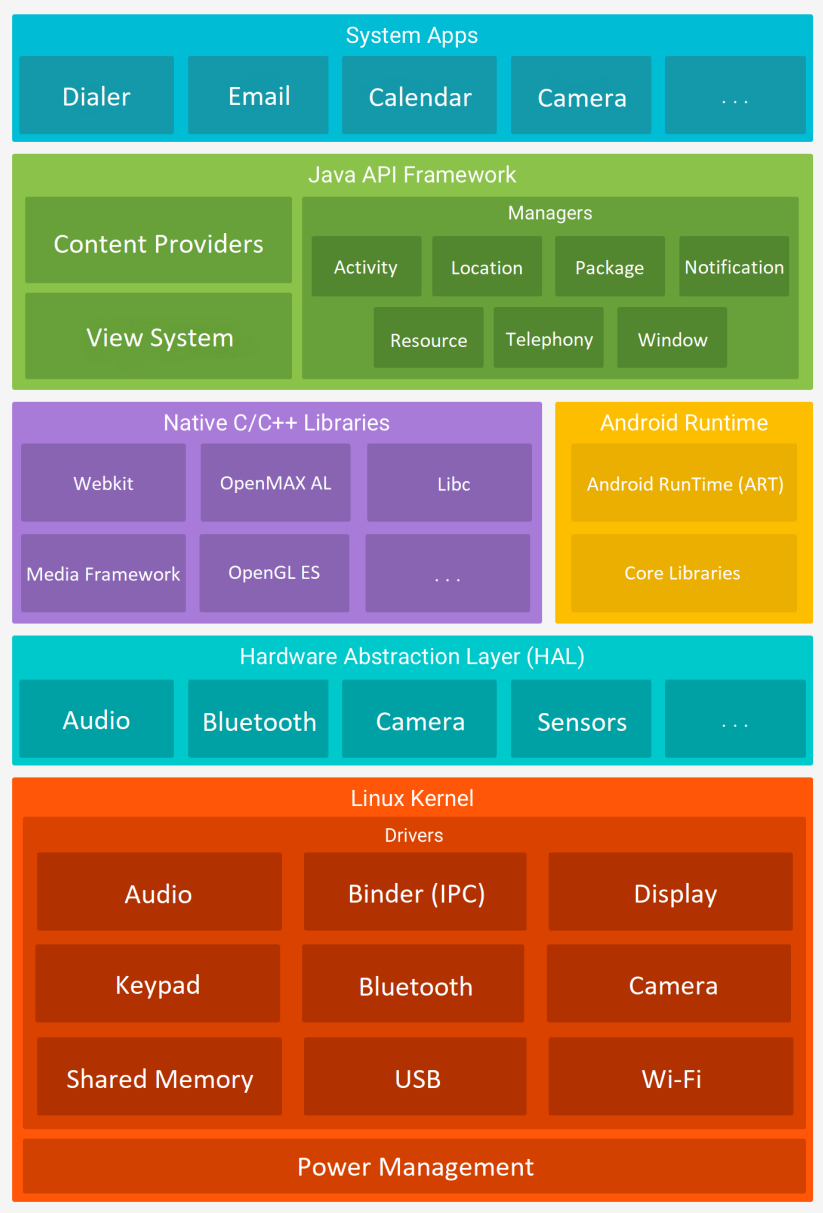



Figura 1: Arquitectura de Android. Fuente: [Android Developers, 2024a]

- **Kernel de Linux**

  Android está construido sobre la versión LTS[1] del kernel de Linux y una serie de parches específicos para Android desarrollados por Google, que hoy en día se conoce como GKI[2]. Gracias a esto, Android es capaz de aprovecharse no sólo de las funcionalidades específicas del kernel de Linux, sino que también cuenta con la robusta y madura seguridad que provee dicho sistema operativo.

  Entre las principales medidas de seguridad que se heredan del kernel de Linux, cabe destacar la capacidad de detectar y aislar los recursos utilizados en una aplicación. Esto se debe a lo que se denomina como *Application Sandbox*, entorno que permite aislar las aplicaciones unas de otras, con la debida finalidad de evitar que aplicaciones maliciosas puedan acceder indebidamente a archivos, permisos y datos gestionados por otras aplicaciones si es que éstas no cuentan con los privilegios apropiados. Dado que dicho entorno se encuentra en el kernel de Android, este modelo de seguridad se extiende tanto a código nativo como a aplicaciones del sistema operativo. Todo el software debajo del kernel, como las bibliotecas del sistema operativo, el *framework* de la aplicación y la aplicación en sí, se ejecutan dentro del *Application Sandbox*. En algunas plataformas, los desarrolladores están limitados a un *framework* específico, un conjunto de APIs o un lenguaje. En Android, dado que no hay restricciones en la forma en que se construye una aplicación, con el objetivo de garantizar la seguridad, el código nativo está aislado al igual que el código interpretado.

  Naturalmente, existen situaciones en que las aplicaciones deben comunicarse, por lo que Android también cuenta con un robusto IPC[3] que facilita la interacción segura de aplicaciones que se ejecutan en procesos distintos.

- **Capa de Abstracción de Hardware (HAL)**

  La Capa de Abstracción de Hardware proporciona interfaces estandarizadas que permiten la utilización y el acceso al hardware del dispositivo a través del *framework* de la API de Java en la capa superior. Dicha capa está compuesta por una serie de módulos pertenecientes a una biblioteca, donde cada uno implementa una interfaz específica para un componente de hardware en particular, por ejemplo, la cámara o el *Bluetooth*.

[1] Versiones de software que ofrecen soporte extendido, generalmente en términos de actualizaciones de seguridad y correcciones de errores, durante un período prolongado de tiempo. Estas versiones priorizan la estabilidad y seguridad sobre las características más recientes.

[2] Iniciativa de Android que estandariza la imagen del kernel en dispositivos, permitiendo actualizaciones más rápidas y consistentes al minimizar la variabilidad del kernel específico del hardware.

[3] Conjunto de métodos que permiten a diferentes procesos compartir datos y comunicarse entre sí dentro de un sistema operativo. Esta comunicación se utiliza para coordinar actividades y compartir recursos.

- **Entorno de Ejecución**

  En un principio *Android* contaba con *Dalvik Virtual Machine* como entorno de ejecución, sin embargo luego sería reemplazado por *Android RunTime* a causa de su ineficiente estrategia de compilación. [Sun *et al.*, 2016] ahonda en la forma en la que operan ambas estrategias y sus principales diferencias, resumiéndolas de manera general en la figura 2 y de manera más detallada a continuación.

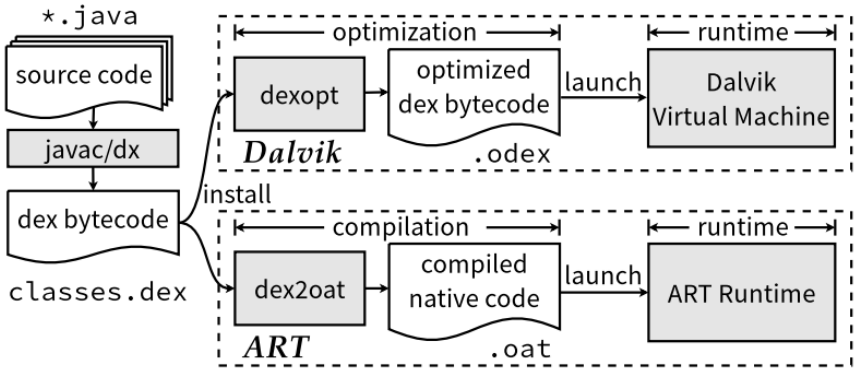



Figura 2: Comparación entre entornos de ejecución *Dalvik Virtual Machine* y *ART*. Fuente: [Sun *et al.*, 2016]

- **Dalvik Virtual Machine (DVM)**
  Las versiones de *Android* de legado (hasta su versión 4.4) contaban con este entorno de ejecución por defecto. En la etapa de instalación, *Dalvik* utiliza una herramienta para optimizar el *bytecode* original de la aplicación. De esta manera, se genera un archivo `odex` que, durante la ejecución, está siendo constantemente interpretado y ejecutado por *Dalvik*. A su vez, en pos de mejorar el rendimiento, se aplica la estrategia de compilación *Just-In-Time*, que consiste en compilar eficientemente ciertas partes del código, optimizándola según la arquitectura del dispositivo.
- **Android RunTime (ART)**
  Luego de que *ART* fuera implementado de manera experimental en la versión de *Android* 4.4, en el año 2015 con la salida de *Android* 5.0, se convirtió en el entorno de ejecución por defecto. A diferencia de *Dalvik*, en la fase de instalación de la aplicación *ART* hace uso de una herramienta que compila la aplicación completa en código nativo antes de su ejecución. Esta corresponde a la estrategia de compilación *Ahead-Of-Time* y destaca por su capacidad de ser utilizada múltiples veces para obtener un código con un mejor desempeño, siendo este enfoque mucho más eficiente que el de *Dalvik*, además de contar con un *Garbage Collection* optimizado.

- **Librerías Nativas de C/C++**

  Android cuenta con múltiples librerías nativas escritas en C y C++ que muchas veces son requeridas por las aplicaciones, ya que cuentan funcionalidades específicas. Estas generalmente son accesibles a través del *framework* de la API de Java.

- **Framework de la API de Java**

  API escrita en Java y que permite a las aplicaciones y a los desarrolladores acceder a todas las funcionalidades y servicios que provee el sistema operativo Android y que se listan a continuación:

  - Un sistema de vista extensible con múltiples herramientas para crear la interfaz de usuario de una aplicación.
  - Un administrador de recursos que brinda acceso recursos que no consisten de código. Entre ellos, *strings* localizadas, gráficos y archivos de diseño.
  - Un administrador de notificaciones que permite a las aplicaciones desplegar alertas personalizadas.
  - Un administrador de actividad que les provee de un ciclo de vida a las aplicaciones y administra la pila de actividades.
  - Proovedores de contenido que permite que las aplicaciones compartan datos y accedan a datos de otras aplicaciones.

- **Aplicaciones del Sistema**

  Corresponden a las aplicaciones que trae Android por defecto (Contactos, Mensajes SMS, Navegadores, Calendario, entre otros) y que pueden aportar con ciertas funcionalidades a aplicaciones construidas por otros desarrolladores. Cabe destacar que aplicaciones pertenecen a la misma capa que las aplicaciones proveídas por el sistema, pudiendo incluso llegar a reemplazarlas en caso de que el usuario lo estime conveniente. Además, es relevante destacar que Google Play Store organiza dichas aplicaciones en las categorías que se muestran en las tablas 1 y 2.

| **Categoría** | **Ejemplos** |
|---|---|
| Arte y Diseño | Cuadernos de Dibujo, Herramientas de Pintor, Herramientas de Arte y Diseño, Libros para Colorear |
| Autos y Vehículos | Compras de Autos, Seguros de Autos, Comparación de Precios de Autos, Seguridad Vial, Reseñas y Noticias de Autos |
| Belleza | Tutoriales, Herramientas y Simuladores de Maquillaje y de Cambio de Imagen, Peluquería, Compras de Belleza |
| Libros y Referencias | Lectores de Libros, Libros de Referencia, Libros de Texto, Diccionarios, Tesauros, Wikis |
| Negocios | Editores/Lectores de Documentos, Seguimiento de Paquetes, Escritorio Remoto, Gestión del Correo Electrónico, Búsqueda de Empleo |
| Cómics | Reproductor de Cómics, Títulos de Cómics |
| Comunicación | Mensajería, Chat, Marcadores, Libretas de Direcciones, Navegadores, Gestión de Llamadas |
| Conocer Personas | Búsqueda de Pareja, Noviazgo, Establecimiento de Relaciones |
| Educación | Preparación de Exámenes, Ayudas de Estudio, Vocabulario, Juegos Educativos, Aprendizaje de Idiomas |
| Entretenimiento | Streaming de Video, Películas, TV, Entretenimiento Interactivo |
| Eventos | Entradas de Conciertos, Entradas de Eventos Deportivos, Reventa de Entradas, Entradas de Cine |
| Finanzas | Banca, Pagos, Buscador de Cajeros Automáticos, Noticias Financieras, Seguros, Impuestos, Cartera/Operaciones, Calculadoras de Propinas |
| Comer y Beber | Recetas, Restaurantes, Guías Gastronómicas, Cata y Descubrimiento de Vinos |
| Salud y Fitness | Fitness Personal, Seguimiento de Entrenamientos, Consejos Dietéticos y Nutricionales, Salud y Seguridad |
| Inmuebles y Hogar | Búsqueda de Casas y Departamentos, Mejora del Hogar, Decoración Interior, Hipotecas, Inmobiliaria |
| Bibliotecas y Demostración | Bibliotecas de Software, Demostraciones Técnicas |
| Estilo de Vida | Guías de Estilo, Organización de Bodas y Fiestas, Guías Prácticas |
| Mapas y Navegación | Herramientas de Navegación, GPS, Cartografía, Herramientas de Tránsito, Transporte Público |
| Medicina | Medicamentos y Referencias Clínicas, Manuales para Profesionales Sanitarios, Revistas y Noticias Médicas |
| Música y Audio | Servicios Musicales, Radios, Reproductores de Música |
| Noticias y Revistas | Periódicos, Agregadores de Noticias, Revistas, Blogs |

Tabla 1: Categorías de Aplicaciones (Google Play Store). Fuente: [Google, 2024]

| **Categoría** | **Ejemplos** |
|---|---|
| Ser Padres | Embarazo, Cuidados y Seguimiento del Lactante, Puericultura |
| Personalización | Fondos de Pantalla, Fondos de Pantalla Animados, Pantalla de Inicio, Pantalla de Bloqueo, Tonos de Llamada |
| Fotografía | Cámaras, Herramientas de Edición Fotográfica, Gestión y Uso Compartido de Fotos |
| Productividad | Bloc de Notas, Lista de Tareas, Teclado, Impresión, Calendario, Copia de Seguridad, Calculadora, Conversión |
| Compras | Compras en Línea, Subastas, Cupones, Comparación de Precios, Listas de la Compra, Reseñas de Productos |
| Social | Redes Sociales, Check-In |
| Deportes | Noticias y Comentarios Deportivos, Seguimiento de Resultados, Gestión de Equipos de Fantasía, Cobertura de Partidos |
| Herramientas | Herramientas para Dispositivos Android |
| Viajes | Herramientas de Gestión y Reserva de Viajes, Viajes Compartidos, Taxis, Guías de Ciudades, Información sobre Empresas Locales, Reserva de Excursiones |
| Aplicaciones de Videos | Reproductores de Video, Editores de Video, Almacenamiento Multimedia |
| Tiempo | Informes Meteorológicos |

Tabla 2: Categorías de Aplicaciones (Google Play Store). Fuente: [Google, 2024]

### 2.1.4. Estructura de Aplicaciones de Android

El método de distribución e instalación de las aplicaciones de Android corresponde a archivos comprimidos en formato APK [Gajrani *et al.*, 2020]. Su contenido consta de cuatro componentes principales:

- **Archivo Manifest (AndroidManifest.xml)**
  Contiene la información esencial de la aplicación, incluyendo los permisos y configuraciones requeridas para su correcto funcionamiento.
- **Archivos DEX (classes.dex)**
  Contiene el código compilado de la aplicación, y por lo tanto, toda su lógica.
- **Recursos**
  Contiene toda la información de UI, imágenes, *strings*, diseños ded disposición y otros recursos necesarios para la aplicación.
- **Carpeta META-INF**
  Contiene toda la información referente a la firma digital de la aplicación y del contenido total de la misma. Se utiliza para verificar la autenticidad del paquete.

La forma en que la aplicación se ejecuta está explicada por la figura 2. Android compila los archivos DEX, que contienen la lógica de la aplicación y los convierte en código nativo al momento de su instalación, permitiendo que luego el entorno de ejecución, en este caso *Android RunTime*, pueda ejecutarlos de manera más eficiente. Cabe destacar que los recursos son cargados por la aplicación al momento que ésta es inicializada.

### 2.1.5. Análisis Estático y Dinámico

Habiendo comprendido las bases del sistema operativo Android, y sus principales componentes, así como la estructura y categorización de sus aplicaciones, es prudente comenzar a desarrollar el foco principal del estudio, es decir, las distintas metodologías que actualmente existen para realizar análisis del código y del comportamiento de las aplicaciones móviles, con el objetivo de detectar potenciales amenazas. Generalmente, dichas metodologías y técnicas pueden agruparse en dos categorías principales, análisis estático y análisis dinámico, respectivamente.

Por un lado, como es definido por [del Moral, 2016], el análisis estático es capaz de examinar exhaustivamente el código fuente (Archivos DEX) y el archivo Manifest de una aplicación sin que esta se encuentre en ejecución, permitiendo prever su comportamiento y el impacto que tendrá el código al momento de ejecutarse en el dispositivo. Si bien el análisis estático se considera eficiente y escalable, [Arp *et al.*, 2014] estipula que su capacidad de detectar *malware* se ve afectada al enfrentarse a aplicaciones cuyo código está ofuscado, es decir, cuando al código se le hayan aplicado modificaciones para dificultar su interpretación. Este puede ser el caso cuando dichas aplicaciones son descargadas e instaladas desde fuentes no oficiales y que posiblemente podrían estar escondiendo código malicioso, o bien, cuando las organizaciones buscan resguardar su código y, naturalmente, evitar que éste sea recuperado y utilizado por terceros. Por ende, el análisis estático generalmente se convierte en una alternativa poco robusta ante *malware* más moderno. Por otro lado, las técnicas de análisis dinámico son utilizadas con la aplicación en ejecución, ya sea en un entorno controlado, es decir, en una máquina virtual, o bien, en el mismo dispositivo. Esto permite analizar su comportamiento y hacer énfasis en las interacciones que ésta tiene con los demás componentes de la aplicación y el sistema de manera general. La tabla 3 que se presenta a continuación resume las principales diferencias entre el análisis estático y dinámico, dejando en evidencia las ventajas y desventajas de ambas alternativas.

| **Componentes** | **Análisis Estático** | **Análisis Dinámico** |
|---|---|---|
| **Ejecución de Código** | No es Posible | Posible |
| **Tipo de Ejecución** | Investigación de Código - Sin operaciones en *runtime* | Investigaciones y operaciones en *runtime* |
| **Tiempo Requerido** | Más tiempo requerido para revisar las líneas de código | Menos tiempo al utilizar un método de automatización |
| **Entrada** | Archivos binarios, archivos con *scripts*, etc | Capturas de memoria, datos de API en *runtime* |
| **Ventajas** | Análisis de todos los *execution tracks* del *AndroidManifest*<br>Puede ayudar a detectar una serie de vulnerabilidades: Fugas de datos sensibles, acceso no autorizado a recursos protegidos o privados y *intent injections*<br>Menor cantidad de recursos y tiempo | Análisis profundo y mayor tasa de descubrimiento de *malware*<br>Más preciso dada la ejecución actual del programa para alcanzar una cobertura de código adecuada |
| **Desventajas** | Base de datos restringida, pudiendo identificar solamente tipos de *malware* conocidos | Mayor consumo de energía y más tiempo para realizar el proceso |
| **Precisión de los Resultados** | Baja precisión en comparación al análisis dinámico | Alta precisión en comparación al análisis estático |

Tabla 3: Tabla comparativa entre Análisis Estático y Dinámico (Traducción Libre). Fuente: [Alkindi *et al.*, 2021]

A partir de la información entregada en la tabla 3 es que una serie de investigadores, entre ellos [Gajrani *et al.*, 2020], optan por un enfoque dinámico o híbrido a causa de su excepcional rendimiento en detectar comportamiento malicioso, a diferencia del *software* que se apoya en análisis únicamente estático, pues su efectividad y precisión es significativamente menor a la de los enfoques dinámicos, como estipula [Hande y Rao, 2017]. Por esta razón, las técnicas de análisis dinámico han sido objeto de estudio durante la última década, sin embargo, varias de ellas estaban construidas para funcionar con *Dalvik Virtual Machine*, el entorno de ejecución por defecto con el que contaba *Android* hasta la llegada del mucho más moderno y eficiente, *Android RunTime*. Si bien, esto significó un mejor rendimiento de los dispositivos, también generó que varias de las técnicas de análisis dinámico quedaran obsoletas, dejando de funcionar en versiones más actuales de *Android*, como bien menciona [Sun *et al.*, 2016].

[Kapratwar, 2016] define al análisis dinámico como un conjunto de técnicas que tienen como objetivo detectar comportamiento potencialmente malicioso de una aplicación de *Android* mientras esta se encuentra en ejecución. El estudio de las distintas interacciones que tiene la aplicación con el dispositivo y los registros que deja dicha aplicación posterior a su ejecución, son algunas de las herramientas con las que cuentan las técnicas de análisis dinámico para detectar conductas anómalas y por consiguiente, posibles fugas de información o fraudes. Ahora bien, una de las falencias que se le puede atribuir a este tipo de análisis, como bien se indica en la tabla 3, es la cantidad de recursos (en términos de tiempo y capacidad de procesamiento) que se necesitan para aplicarlas a un entorno real, es decir, a un dispositivo móvil. Es por esto que, para no incurrir en un consumo desmedido de los recursos del

dispositivo, las técnicas generalmente se implementan en un ambiente controlado fuera del dispositivo, evitando sobrecargarlo y contando con más recursos para el análisis.

### 2.1.6. Tipos de Técnicas de Análisis Dinámico

Durante el pasar de los años, se han explorado distintos enfoques para aplicar las técnicas de análisis dinámico presentadas en múltiples contextos y bajo distintas condiciones, por lo que, en un esfuerzo por organizar y precisar el alcance de las técnicas planteadas hasta la fecha, [Gajrani *et al.*, 2020] define una taxonomía para las técnicas de análisis dinámico, permitiendo clasificar cada una de las propuestas según su objetivo. Dicha taxonomía queda ilustrada en la figura 3.

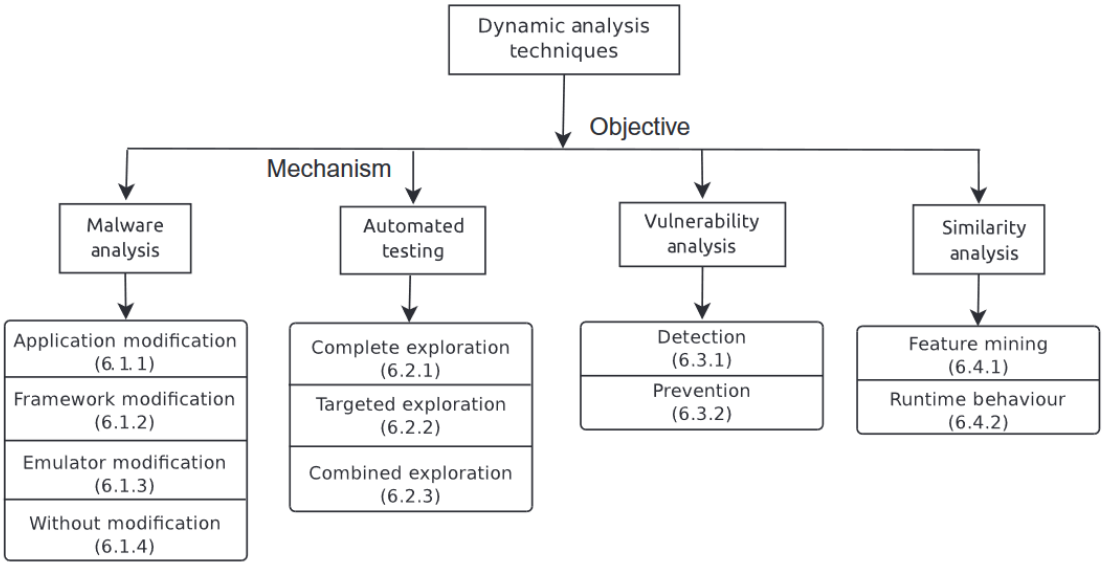



Figura 3: Taxonomía de técnicas de Análisis Dinámico. Fuente: [Gajrani *et al.*, 2020]

La primera categoría, es decir, aquellas técnicas cuyo objetivo es el análisis y detección de *malware* y que será el principal foco de este estudio, cuenta con cuatro subcategorías, que en este caso hacen referencia a modificaciones que se deben hacer ya sea a la aplicación, al *framework*, al emulador o entorno de prueba, o bien no requieren de modificación alguna. Si bien, esta manera de agrupar las técnicas de análisis dinámico es llamativa, se optará por otro enfoque propuesto por el mismo autor, que busca agrupar dichas técnicas según su metodología. Estas categorías se detallan a continuación y en la próxima sección.

- **Virtual Machine Introspection (VMI)**
  Técnica que hace uso de una máquina virtual como entorno de ejecución para replicar el contexto y el comportamiento de una aplicación (estructuras de datos, llamadas del sistema, procesos, entre otros). Su principal ventaja radica en que el software

potencialmente malicioso actuará en la máquina virtual y no en el sistema principal, volviéndola robusta frente a malware más elaborado.

- **API Monitoring**
  Su principal objetivo es obtener *logs* de APIs dentro del sistema, incluyendo sus parámetros y valores retornados, permitiendo detectar cuando se hagan llamadas a métodos específicos que podrían potencialmente acceder y filtrar información contenida en el dispositivo.

- **Behavior-based Monitoring**
  Se focaliza en monitorear el comportamiento de la aplicación cuando esta se encuentra en ejecución. Esto incluye llamadas sospechosas al sistema y APIs, ingreso y salida de paquetes, accesos a URLs y modificaciones a ficheros y/o memoria. Es común que para esta clase de análisis se adapten técnicas de *machine* y *deep learning*, y dado su auge, cada vez existen más estudios que abordan este tipo de técnicas.

- **Taint Analysis**
  La idea tras esta técnica es hacerle un seguimiento a la información que sale del dispositivo y detectar cómo ciertos datos (contactos, fotos, correos electrónicos, ubicación) fueron filtrados a través de la red o por medio de un SMS. Para esto, se utilizan métodos que permiten ”etiquetar” la información sensible, posibilitando su seguimiento y pudiendo identificar los canales por donde ésta transita y posiblemente se filtra.

- **Forensic Analysis**
  Esta técnica tiene un foco más en investigaciones de cibercrímenes y busca analizar el sistema de ficheros, *logs* del sistema y memoria física, permitiendo examinar procesos en ejecución, lectura de archivos, actividades de red y otras características que puedan ser obtenidas de las fuentes mencionadas.

Dentro de las categorías mencionadas, aquellas que cuentan con el respaldo de una mayor cantidad de estudios, corresponden a las técnicas basadas en *Taint Analysis* y *Behaviour-based Monitoring*, por lo que convertirán en el foco principal del presente estudio.

### 2.1.7. Detección de Fugas de Información

En primer lugar, como bien se mencionó en la sección anterior, las técnicas de *Taint Analysis* son capaces de detectar fugas a través del seguimiento de la información hasta que ésta abandona el dispositivo. Esto se logra ”etiquetando” los datos cuando éstos se encuentran en memoria, lo que naturalmente requiere de la previa instrumentación tanto de las aplicaciones, como del dispositivo, para introducir debidamente la lógica de seguimiento. De esta manera, cuando alguna aplicación solicite información sensible al dispositivo, el *Binder IPC* será capaz de agregar una etiqueta a dicha información, permitiendo su seguimiento a través del dispositivo. Ahora bien, la fuga es detectada cuando la información etiquetada

es enviada a través de la red, o bien, cuando es exportada a algún tipo de almacenamiento externo [Sun *et al.*, 2016].

Por el otro lado, las técnicas basadas en *Behaviour-based Monitoring* detectan fugas de información a través del análisis de los *payloads* y de la *metadata* de paquetes salientes, y de *logs* provenientes de distintas fuentes. En ambos casos, se requiere del análisis de grandes volúmenes de texto para la detección de información relevante y que, en este caso, pudiera indicar una posible filtración o vulneración de la privacidad de los usuarios. Por lo tanto, en la literatura destacan el uso de distintos algoritmos de *string matching* según el tipo de información que se está analizando y según el escenario específico del análisis. A continuación, se listan aquellas variaciones para algoritmos de *string matching* utilizados por [Shuba *et al.*, 2016] y [Novak *et al.*, 2020].

- **Aho-Corasick**
  El algoritmo *Aho-Corasick* es un algoritmo de *string matching* diseñado para encontrar múltiples patrones simultáneamente en un texto. Utiliza una estructura de datos denominada ”trie” (árbol de prefijos) para construir un autómata finito que represente todos los patrones, junto con enlaces para manejar coincidencias parciales, es decir, aquellos *strings* con el mismo prefijo [Vashchegin, 2024].

- **Boyer-Moore**
  El algoritmo *Boyer-Moore* es un algoritmo eficiente para la búsqueda de un único patrón en un texto. Utiliza varias heurísticas que permiten saltar secciones del texto y evitar comparaciones redundantes, lo que lo vuelve especialmente eficiente con grandes volúmenes de textos. Entre ellas destacan, por un lado, la heurística denominada ”desplazamiento del carácter malo” y por otro lado, la heurística denominada ”desplazamiento del sufijo bueno” [Pushkar, 2024].

- **Basado en Distancia de Levenshtein**
  La distancia de *Levenshtein* mide el número mínimo de operaciones necesarias para transformar un *string* en otro. Dichas operaciones incluyen inserciones, eliminaciones o sustituciones de caracteres. Esto lo vuelve útil para encontrar tanto coincidencias aproximadas en textos como coincidencias exactas. A su vez, esto permite establecer un grado de similaridad máximo entre dos *strings* a través de un *threshold*[4] que limite la diferencia que existe entre ambos para considerarse como coincidencias [Po, 2020].

Cabe destacar que los algoritmos presentados, pueden ser debidamente combinados para alcanzar un grado mayor de eficiencia y adaptabilidad según el escenario específico.

[4]Umbral o límite específico que, cuando se alcanza o se supera, provoca una acción o un cambio.

## 2.2. Estado del Arte

En esta sección, se presentarán algunas técnicas y *frameworks* basados en análisis dinámico y que han logrado combatir de alguna u otra forma las problemáticas que plantean los métodos de protección convencionales, suponiendo un avance en cuanto a la detección y contención de fugas de información. En particular, se enfatizará en las formas de protección más comunes y disponibles actualmente, además de ahondar en las dos categorías estudiadas en la sección anterior: *Taint Analysis* y *Behavior-based Monitoring*.

### 2.2.1. Protección Convencional

Como es de esperarse, hoy en día ya existen herramientas disponibles que actúan como una primera línea de defensa ante aplicaciones de carácter malicioso, y que implementan, de alguna u otra forma, técnicas tanto de análisis estático como de análisis dinámico.

- **Google Play Protect**

  En primer lugar, y quizás la más importante, existe *Google Play Protect*. Como bien menciona [Google, 2020], ésta corresponde a una herramienta integrada a *Google Play* que busca analizar todas las aplicaciones que sean publicadas en la plataforma antes de que estén disponibles para descargar. Como una gran parte de las técnicas modernas, *Google Play Protect* se apoya directamente en las técnicas de *machine learning* proveídas por Google, permitiendo discriminar si una aplicación es dañina, potencialmente dañina o segura. Solamente aquellas que sean categorizadas como seguras podrán ser descargadas a través de la plataforma, mientras que aquellas que no se puedan clasificar quedan categorizadas como potencialmente dañinas y tendrán que ser revisadas manualmente por un analista. Aquellas aplicaciones dañinas son bloqueadas automáticamente, además de sus respectivos desarrolladores.

  Naturalmente, este sistema no es perfecto, ya que con la sofisticación del *malware* más moderno, los ciberdelincuentes han encontrado formas de incluir aplicaciones maliciosas, traspasando la protección principal que brinda *Google Play*. Por ejemplo, [Muhammad *et al.*, 2023] da a conocer un método que permite anular por completo la herramienta de Google, publicando en primera instancia una aplicación benigna y agregarle código malicioso incrementalmente a través de actualizaciones. Ahora bien, también existen aplicaciones que efectivamente están libres de *malware* y por lo tanto no son bloqueadas por *Google Play Protect*, pero que de igual manera podrían llegar a presentar un comportamiento no deseado, recopilando datos y filtrándolos a entidades externas, como es el caso de algunas aplicaciones que cuentan con publicidad y pueden derivar muchas veces en la instalación desapercibida de malware.

- **Antivirus**

  Al tener motivos para dudar de la protección que podría considerarse como oficial, los usuarios optan por contar una segunda línea de defensa, los antivirus. Éstos, si bien pueden sonar recomendables y podrían llegar a jugar un rol clave en la detección de comportamiento malicioso en aplicaciones descargadas desde la plataforma de *Google Play*, la realidad es bastante distinta. Como bien menciona [Hammad *et al.*, 2018], los antivirus están basados en su mayoría en análisis de código estático, y por si fuera poco, bastante básico, pues las pruebas indicaban poca efectividad contra ofuscaciones de código, es decir, código modificado que dificulta su comprensión y que en muchos casos es usado para ocultar código malicioso. De la misma manera, [Aswal *et al.*, 2023] realiza un estudio que busca evaluar justamente eso, comparando la eficacia de los antivirus actuales ante malware ofuscado. Los resultados dejaron al descubierto el bajo rendimiento de dichos antivirus y cómo éstos no parecen ser capaces de adaptarse a técnicas de evasión relativamente simples.

  Ahora bien, en la mayoría de los casos en que los antivirus sí utilizan análisis dinámico, éste es realizado en un entorno externo al dispositivo dada la alta cantidad de recursos que comúnmente se requieren, pudiendo causar un *overhead*[5] demasiado elevado e interferir con el funcionamiento correcto del *smartphone*. El problema de aplicar análisis dinámico de manera externa es que, en primer lugar, no permite representar con exactitud el entorno que se quiere proteger, y por otro lado, no permite detectar fugas de información en tiempo real, lo que finalmente la vuelve una alternativa poco viable para evitar filtraciones de datos. Como se puede observar en el análisis realizado por [AV-Comparatives, 2023], lo que comúnmente se considera análisis en tiempo real por parte de los antivirus, corresponde a técnicas como la protección contra *phishing*, el monitoreo constante de los permisos con los que cuentan las aplicaciones (muchas veces llamados auditorías) y la protección del tráfico de red a través de VPN, siendo formas más indirectas de evitar las filtraciones de datos y por lo tanto, falibles.

- **VPN**

  Finalmente, es importante mencionar el rol que cumplen las redes privadas virtuales, o VPNs, en la protección de datos. Como bien mencionan [Jyothi y Reddy, 2018], dentro de sus principales ventajas, figura su capacidad de enmascarar la dirección IP del usuario, lo que dificulta el seguimiento de las actividades que realiza el usuario en la web. Además, las VPN son capaces de cifrar el tráfico de datos, lo que hace que sea más difícil para los ciberdelincuentes llevar a cabo ataques como *data interception*[6], *network eavesdropping*[7] y *session hijacking*[8], permitiendo

[5] Costo adicional o recursos extras necesarios para realizar una tarea más allá de su función principal.

[6] Proceso de capturar, acceder o monitorear información transmitida a través de redes, a menudo sin autorización, con el fin de obtener datos confidenciales o información sensible en general.

[7] También llamado "espionaje de red". Acto de escuchar pasivamente las comunicaciones en una red, sin alterar los datos, generalmente para obtener información sin el conocimiento de los usuarios.

[8] Ataque en el que un intruso toma el control de una sesión activa entre un usuario y un sistema, pudiendo acceder a información privada o realizar acciones en nombre del usuario legítimo.

proteger la integridad y la confidencialidad de la información perteneciente a un usuario en un entorno cada vez más amenazado.

Aun así, las VPN no pueden proteger contra todas las amenazas potenciales. A pesar del cifrado, los ciberdelincuentes pueden utilizar métodos más sofisticados que no dependen del acceso no autorizado a datos en tránsito. Por ejemplo, el cifrado de extremo a extremo en una aplicación puede garantizar la seguridad de los datos durante el trayecto desde el dispositivo del usuario hasta el servidor. Sin embargo, una vez en el destino, los datos pueden acabar en un servidor malicioso que finalmente los descifre y haga mal uso de ellos. Asimismo, las aplicaciones maliciosas son capaces de omitir la autenticación del servidor o falsificar certificados de seguridad, lo que también permite eludir la protección que intrínsecamente entrega una VPN.

### 2.2.2. Taint Analysis

Como bien se mencionó en la sección anterior, el denominado *Taint Analysis* corresponde a un tipo de técnica que se basa en una serie de métodos para ”etiquetar” información sensible, permitiendo hacer un seguimiento de los datos al moverse a través del dispositivo. Esto permite identificar los canales por donde dicha información circula y eventualmente se filtra. El principal desafío que enfrentan estas técnicas es la instrumentación requerida en el dispositivo para realizar el etiquetado, por lo que a continuación se detallarán distintas formas de hacer frente a dichas dificultades.

- **TaintDroid**

  [Enck *et al.*, 2010] plantea sin duda una de las primeras y más influyentes técnicas de análisis dinámico que incursionan en el denominado *Taint Analysis*. Una de sus principales ventajas, corresponde a que los investigadores tenían como objetivo generar la menor cantidad de *overhead* posible, lo que luego se tradujo en la obtención de resultados sumamente prometedores en comparación a las técnicas desarrolladas hasta la fecha que no tomaron en consideración este enfoque en particular. La arquitectura y funcionamiento de *TaintDroid* se ilustra en la figura 4 y se detalla a continuación.

  En primer lugar, la información es etiquetada por una aplicación de confianza (1). Luego, el intérprete de Dalvik invoca un método nativo (2) que permite almacenar la etiqueta en un *Virtual Taint Map* (3). En este punto y con ayuda de la librería *Binder IPC*, que debe estar previamente modificada, *TaintDroid* es capaz de reflejar la información con su respectiva etiqueta a otras aplicaciones, a través del kernel (5). Al momento que la información es correctamente recibida por otra aplicación, en este caso poco confiable, nuevamente la librería *Binder IPC* es la encargada de recuperar la información etiquetada (6) para que luego el intérprete de Dalvik la propague según corresponda (7). Finalmente, cuando la aplicación poco confiable invoca un método con el objetivo de enviar información fuera del dispositivo (8), *TaintDroid* reporta el evento como

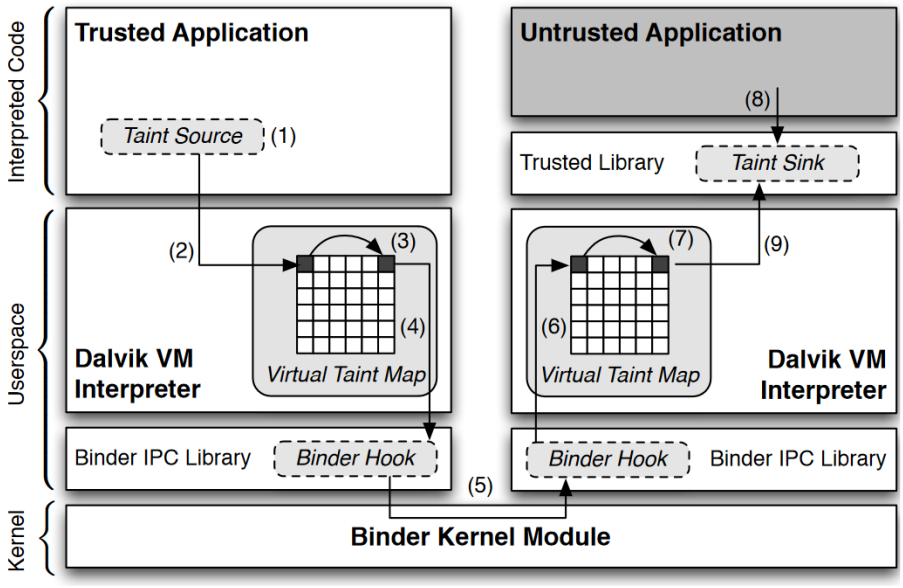



Figura 4: Arquitectura de TaintDroid. Fuente: [Enck *et al.*, 2010]

una fuga de información sensible, o *taint sink*, cuando los datos que están tratando de abandonar el dispositivo estaban debidamente etiquetados (9).

A lo largo del proceso de detección de fugas de información, los desarrolladores de *TaintDroid* utilizaron una serie de métodos que permitieron reducir al máximo el uso del almacenamiento, memoria y capacidad de procesamiento del dispositivo, logrando así minimizar el *overhead* causado por propagar las etiquetas a través de distintas aplicaciones, alcanzando sólo un 14 % de aumento en el uso de la CPU. Esto revolucionaría las técnicas de *Taint Analysis* y sentaría las bases para aquellas que se implementarían posteriormente, especialmente su sucesor.

- **TaintART**

Por su parte, *TaintDroid* estaba diseñado para funcionar con el entorno de ejecución *Dalvik* y naturalmente quedó obsoleto con la llegada de *Android RunTime* y su nueva estrategia de compilación *Ahead-Of-Time*. En respuesta a esta problemática [Sun *et al.*, 2016] planteó *TaintART*, sistema que buscaba replicar la metodología planteada por su predecesor, pero con una serie de mejores en términos de eficiencia y compatibilidad con nuevas versiones de *Android*.

Para la implementación de esta técnica, el primer paso es la modificación del compilador de *Android RunTime* que se ilustra en la figura 5. Estructuralmente las primeras dos etapas se mantienen iguales, es decir, las etapas donde se construye y se optimiza la aplicación, por lo que su desempeño no se verá afectado cuando ésta se encuentre en ejecución. Es en la etapa final donde se realizará la modificación. La lógica de *taint* será introducida cuando esté por terminar el proceso de compilado y que permitirá manejar las variables etiquetadas y su paradero. *TaintART*, a diferencia de

*TaintDroid* se vale directamente de los registros del procesador para facilitar el rápido almacenamiento y acceso de dichas etiquetas.

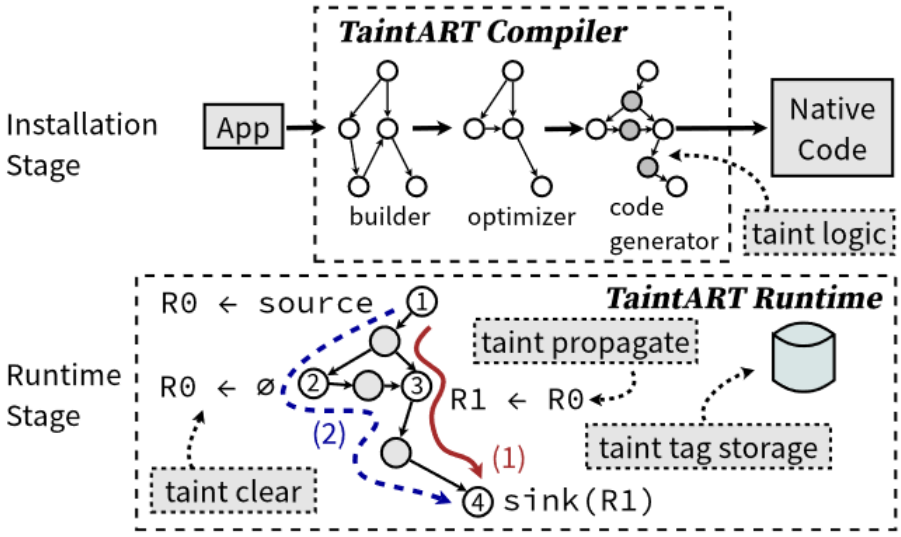



Figura 5: Arquitectura de TaintART. Fuente: [Sun *et al.*, 2016]

La figura 5 ilustra la idea general que sigue este *framework*, presentando un ejemplo a grandes rasgos de dos posibles flujos de información sensible, donde cada nodo corresponde a un procedimiento que sigue el programa. El nodo 1, trae información sensible de alguna fuente en concreto y lo guarda en el registro `R0`. En el nodo 2, se ejecuta un procedimiento que borra la información del `R0` y por consiguiente, su etiqueta. Mientras tanto, el nodo 3 replica la información del `R0` al `R1` y realiza la propagación de la etiqueta antes de pasar al siguiente procedimiento (representada por la lógica de los nodos intermedios de color gris). Por último, el nodo 4 representa un *taint sink*, es decir, que el registro `R1` fue enviado a través de canal poco confiable, por ejemplo a la red Wi-Fi. Si es que se sigue el flujo denotado por la línea roja, el registro `R1` estará etiquetado como información sensible cuando llegue al nodo 4, indicando una fuga de dicha información. Por el contrario, si es que se sigue la línea azul, se entiende que la información sensible fue eliminada del registro cuando pasó por el nodo 2, con lo que éste no estaría etiquetado y por lo tanto, la información que sale del dispositivo no sería de carácter privado o sensible.

- **TaintMan**

  Tomando en cuenta las limitaciones con las que aún contaba *TaintART*, [You *et al.*, 2017] planteó *TaintMan*, una técnica cuya principal diferencia con respecto a las presentadas previamente, corresponde a que no requiere orquestar el dispositivo.

  Fundamentalmente, *TaintMan* está basado en *TaintDroid*, haciendo un manejo de las etiquetas muy similar al de dicha técnica. La diferencia radica en el tipo de ins-

trumentación necesaria para añadir la lógica de *taint*, que en este caso, sucede a nivel de aplicación. *TaintMan* cuenta con dos grandes componentes, por un lado, una herramienta de instrumentación que se ejecuta en un computador, y por otro lado, una herramienta de *hijacking* que se ejecuta en el dispositivo. La primera herramienta se encarga de realizar todos los cambios necesarios a la aplicación cuando el dispositivo es conectado al computador, además de importar las librerías modificadas que contienen la lógica de *taint*, almacenándolas en un directorio espécifico para su posterior acceso. Mientras tanto, la segunda herramienta permite forzar que las aplicaciones utilicen dichas librerías en lugar de las oficiales, logrando implementar la técnica sin necesidad de obtener mayores privilegios por parte de Android.

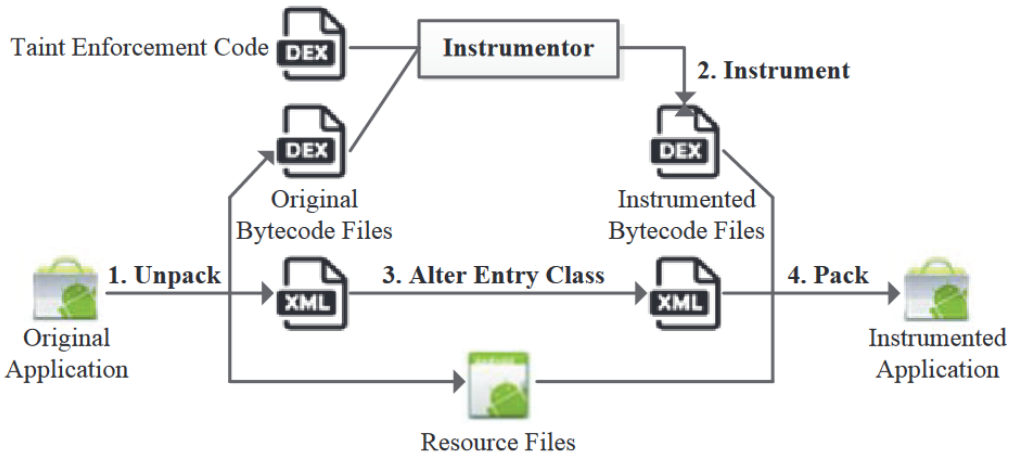



Figura 6: Instrumentación de *TaintMan*. Fuente: [You *et al.*, 2017]

La figura 6 ilustra de mejor manera la instrumentación de esta técnica. La idea principal es conectar el dispositivo al computador que se encuentre ejecutando el instrumentador y luego, a través de una herramienta que permite realizar ingeniería inversa, se descomprime la aplicación en cuestión. Después, se agrega el código que contiene la lógica de *taint* y se modifica el *Manifest* para lograr que la aplicación apunte a la herramienta de *hijacking*, que será la encargada de ejecutar las librerías previamente modificadas. Para finalizar, se vuelve a empaquetar la aplicación y se instala en el dispositivo, permitiendo a éste realizar el etiquetado respectivo de la información y detectar filtraciones.

### 2.2.3. Behavior-based Monitoring

Continuando con el estudio de las técnicas de análisis dinámico que se adecúan de mejor manera a la problemática planteada, aquellas basadas en *Behavior-based Monitoring* se posicionan como la otra alternativa viable. Esta categoría en particular, como bien se mencionó previamente, se centra en monitorear el comportamiento en *runtime* de la aplicación, es decir, las llamadas realizadas por la aplicación ya sea directamente al sistema o a APIs,

el flujo de red y modificaciones a ficheros o memoria. En particular, las técnicas que serán detalladas a continuación se enfocan en el análisis del flujo de red. Para ello, dichas técnicas utilizan algún tipo de VPN o servidor proxy que permite interceptar dicho flujo y analizarlo en base a distintos enfoques.

- **ReCon**

  [Ren *et al.*, 2016] plantea *ReCon*, una técnica que correspondería al punto de partida para aquellas que la sucedieron. Esto se debe a que fue una de las primeras en utilizar *machine learning*[9] para la detección de PII desconocidos y que hasta la fecha sólo se habían detectado utilizando *string matching* entre el payload (y en algunos casos, los headers) de los paquetes y PII predefinidos dentro del dispositivo. Este último sería el enfoque adoptado por [Song y Hengartner, 2015] con *PrivacyGuard*, [Shuba *et al.*, 2016] con *AntMonitor* y [Razaghpanah *et al.*, 2015] con *Haystack*, técnicas que se desarrollaron de manera simultánea a *ReCon*.

  Una de sus principales características es el uso de una VPN en un servidor remoto, realizando el análisis fuera del dispositivo a diferencia de las técnicas que serían planteadas más adelante. Es por esto que el clasificador implementado debe realizar el bloqueo de leaks de manera automática, no permitiendo al usuario bloquear la salida de los paquetes manualmente. Naturalmente, esto se debe a que la conexión remota introduce un *delay* en el todo el proceso de detección y posterior bloqueo de los paquetes que contengan información sensible, haciendo que una posible intervención del usuario fuera más difícil de ejecutar. Ahora bien, este modelo de *machine learning* se alimenta de flujos de red etiquetados y cuenta con la flexibilidad de ser re-entrenado según el feedback del usuario, por lo que, si bien éste no es capaz de intervenir en el momento que se detecta la fuga, sí es capaz de retroalimentar el modelo en base a sus preferencias. A partir de dicha retroalimentación del usuario, el servidor remoto es capaz de bloquear los leaks o incluso modificar la información sensible detectada dentro de un paquete. La figura 7 detalla de manera general el funcionamiento de esta técnica.

  *ReCon* utiliza un modelo de *machine learning* basado en *Decision Trees* generados por el algoritmo C4.5, mostrando resultados incluso mejores resultados que algoritmos convencionales que requieren de un mayor tiempo de procesamiento. Para la obtención de *features*, naturalmente se debe inspeccionar el contenido de los paquetes, que en múltiples ocasiones están encriptados, para lo que se usan herramientas como *SSLsplit*, *Tcpdump* y *Bro*. Luego de obtener el *payload* del paquete, los datos son divididos a partir de una serie de delimitadores cuidadosamente seleccionados para abarcar la mayoría de los contextos. Una observación significativa es que, en general, cuando un PII se filtra, tiende a hacerlo repetidas veces. Por lo tanto, *ReCon* prioriza aquellos datos con mayor frecuencia de aparición que finalmente conformarán los

[9] Técnicas diseñadas para encontrar una secuencia (o ”patrón”) dentro de un texto más extenso, permitiendo la detección rápida de coincidencias exactas o aproximadas entre cadenas de caracteres.

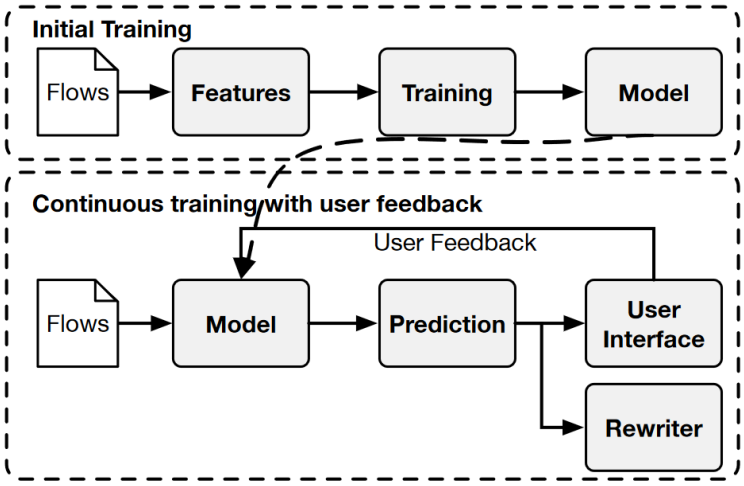



Figura 7: Funcionamiento de *ReCon*. Fuente: [Ren *et al.*, 2016]

*features* del modelo, resultando en una detección de PII más precisa.

- **AntMonitor**

  Como se mencionó previamente, [Shuba *et al.*, 2016] crea *AntMonitor*, que a diferencia de *ReCon*, utiliza una VPN dentro del dispositivo y se focaliza en la detección de PII predefinidos. Al igual que *PrivacyGuard*, *AntMonitor* limita su detección a leaks del IMEI, número telefónico, contactos, ubicación, entre otros. Es decir, información estática que puede obtenerse del dispositivo. No obstante, incluye una interfaz gráfica que facilita la definición de *strings* a proteger, donde se pueden agregar credenciales e información sensible que el usuario estime conveniente.

  Dado que *AntMonitor* hace uso de una VPN dentro del dispositivo, una gran parte de los recursos son utilizados por este componente en particular, por lo que se hace especial énfasis en la importancia de una implementación eficiente para este tipo de técnicas. Donde suele haber un mayor impacto es en el *downlink* y *uplink*, es decir, la bajada y subida de datos a través de la red. A nivel de rendimiento, mientras técnicas como *PrivacyGuard* usan un hilo por cada conexión TCP, lo que resulta en un rendimiento promedio, el *forwarder* de *AntMonitor* actúa como un *proxy*, lo que le permite manejar múltiples conexiones TCP simultáneamente. Además, emplea *multiplexing* para los *sockets* UDP, en comparación con *Haystack* que utiliza una cantidad significativamente mayor de *sockets* por conexión.

  *AntMonitor* lidia con estos desafíos y consigue superar técnicas con enfoques similares, logrando un *downlink* 2 veces mejor y un *uplink* 8 veces mejor. Además, se enfatiza que *AntMonitor* que el *throughput* general es sólo un 6 % más bajo que un dispositivo que no cuenta con una VPN, validando la eficiencia de su implementación. De manera complementaria, se desarrolló una aplicación personalizada llamada *AntEvaluator*, con el objetivo de evaluar y afinar la eficacia de *AntMonitor*.

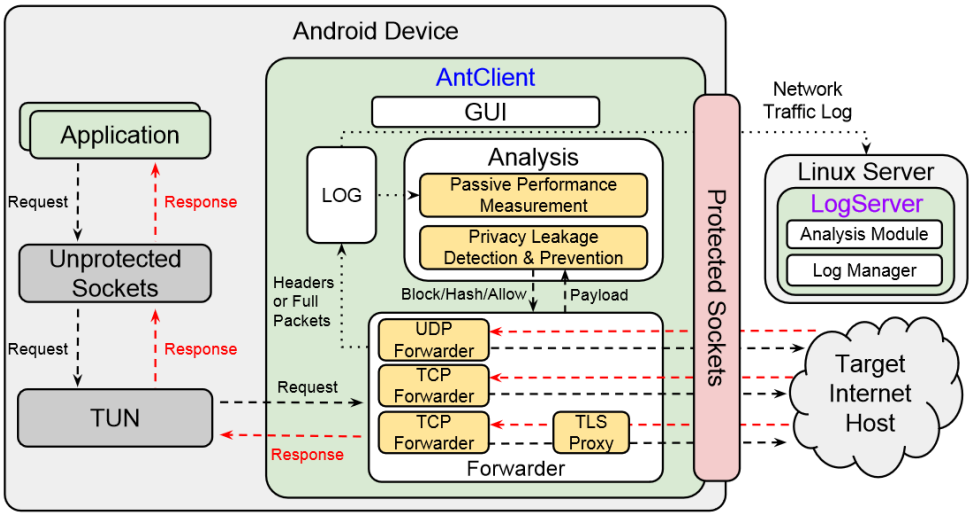



Figura 8: Arquitectura de *AntMonitor*. Fuente: [Shuba *et al.*, 2016]

Si bien, *AntMonitor* se plantea como una técnica que funciona completamente dentro del dispositivo, también posee un componente en particular que actúa fuera de éste y que se muestra en la figura 8. El componente denominado *Linux Server* o *Log Server*, funciona específicamente para recibir y analizar el tráfico de red a partir de los *logs* enviados por la aplicación. Esto con el objetivo de extraer *features* a partir de los headers TCP/IP y otro tipo de información contextual (hora, ubicación) para el entrenamiento de modelos de *machine learning* que permitan estudiar especialmente los flujos de red que contengan filtraciones.

En línea con la eficiencia intrínseca que busca lograr *AntMonitor*, para detectar aquella información sensible dentro de los paquetes se utiliza el algoritmo *Aho-Corasick*. Para no incurrir en un uso de recursos tan elevado que teóricamente podría traer dicho algoritmo, éste es implementado en C nativo, entregando mejores resultados generales que su implementación en Java u otros algoritmos de *string matching*.

- **AntShield**

[Shuba *et al.*, 2018] presenta una evolución significativa de técnicas previas de detección de PII con su técnica *AntShield*. Esta técnica propone un enfoque híbrido que combina el uso de *machine learning* para identificar PII desconocidos, y algoritmos de *string matching* para aquellos predefinidos. Fundamentalmente, *AntShield* es una extensión de *AntMonitor*, puesto que la única diferencia radica en la incorporación de código capaz de extraer *features* del *payload* de los paquetes y alimentar a un modelo de *machine learning* para detectar PII que no hayan sido previamente definidos por el usuario.

Naturalmente, la arquitectura presentada en la figura 9 es muy similar a la de su predecesor pero con las modificaciones correspondientes a la nueva funcionalidad. En este caso, la sección denotada como ”Online Leak Detection”, contiene la nueva lógica

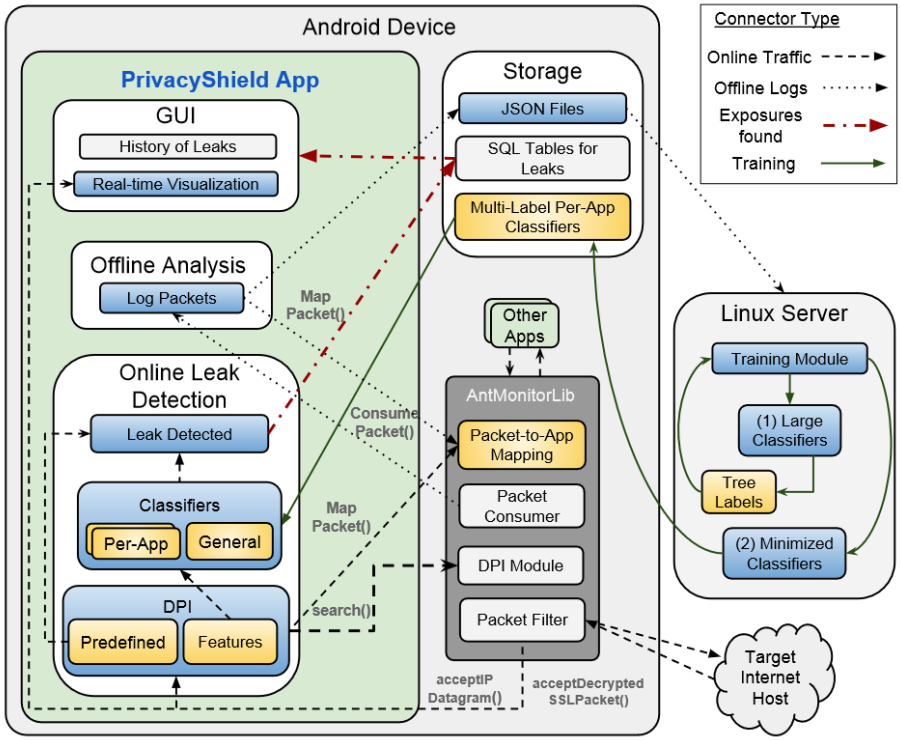



Figura 9: Arquitectura de *AntShield*. Fuente: [Shuba *et al.*, 2018]

de la herramienta, donde destaca DPI, que corresponde al proceso donde se utiliza el algoritmo de *string matching* y se obtienen los *features* respectivos que luego serán utilizados por el clasificador para potencialmente detectar fugas. Todo este proceso está sustentado por el componente *AntMonitorLib* que constituyen las herramientas más importantes para el correcto funcionamiento de *AntMonitor* y ahora de *AntShield*.

Su metodología, en lugar de orientarse por dominio como *ReCon*, se enfoca en las aplicaciones, empleando *Binary Relevance*[10] con *Decision Trees*. Para esto, se construyeron dos tipos clasificadores distintos. En primer lugar se cuenta con uno que funciona para aplicaciones específicas, y en caso de no existir dicho clasificador, se utiliza uno general. Ambos clasificadores están constantemente siendo actualizados por medio de *crowdsourcing*[11], lo que sugiere una mayor efectividad general. En un esfuerzo por optimizar la memoria, *AntShield* fue diseñado para cargar únicamente los modelos de las aplicaciones presentes en el dispositivo, y realizar una modificación al árbol generado del clasificador general, con el objetivo de reducir la carga excesiva de *features*, previniendo así llamadas bloqueantes al *garbage collector*[12].

[10]Técnica de clasificación multi-etiqueta que trata cada etiqueta como un problema de clasificación binaria independiente, transformando tareas multi-etiqueta en múltiples tareas de clasificación binaria.

[11]Práctica de obtener ideas, servicios o contenido solicitando contribuciones de una gran comunidad, generalmente en línea.

[12]Componente encargado del proceso de identificación y liberación de memoria que ya no está en uso por un programa, evitando así fugas de memoria.

- **VPN+**

  [Novak *et al.*, 2020] presenta una de las innovaciones más recientes en lo que respecta a técnicas basadas en *Behavior-based Monitoring*. Dicha técnica, denominada *VPN+* está inspirada fundamentalmente en las capacidades de *PrivacyGuard*. En particular, *VPN+* introduce dos mejoras a las técnicas más dominantes hasta la fecha. En primer lugar, una modificación al algoritmo de *string matching*, que permite detectar pequeñas variaciones de los *strings* predefinidos, y por otro lado, la implementación de un modelo de clasificación *Multinomial Naive Bayes*[13], que tiene la capacidad de predecir de manera automática basándose en las preferencias del usuario.

  Para la tarea de *string matching*, *VPN+* utiliza un algoritmo que se basa en *Boyer-Moore* que incorpora el cálculo de la distancia *Levenshtein* entre los *strings* que corresponden a PII y el *payload* del paquete. Esto permite a *VPN+* buscar coincidencias aproximadas en lugar de centrarse exclusivamente en coincidencias exactas, logrando detectar información sensible que haya sido ligeramente ofuscada y que supone un problema en aquellas técnicas que hacen uso de otros algoritmos para la detección.

  Por el otro lado, para el caso de recordar las preferencias del usuario, como bien se mencionó, se utiliza un modelo de clasificación simple. Este clasificador utiliza como *features* solamente el nombre de la aplicación, la categoría del PII y el destino al que se dirige la información. De esta manera, cada vez que se detecte un *leak*, el usuario será capaz de permitir o bloquear la salida del paquete. El modelo irá entrenándose conforme el usuario vaya seleccionando consistentemente la opción que crea correcta y eventualmente el modelo será capaz de seleccionar dicha opción de manera automática.

  Para ilustrar de mejor manera el funcionamiento de VPN+, se presenta su arquitectura en la figura 10.

[13]Variante del algoritmo Naive Bayes utilizada principalmente para clasificación de texto. Está basado en la distribución multinomial y es adecuado para características discretas, como contar palabras en documentos

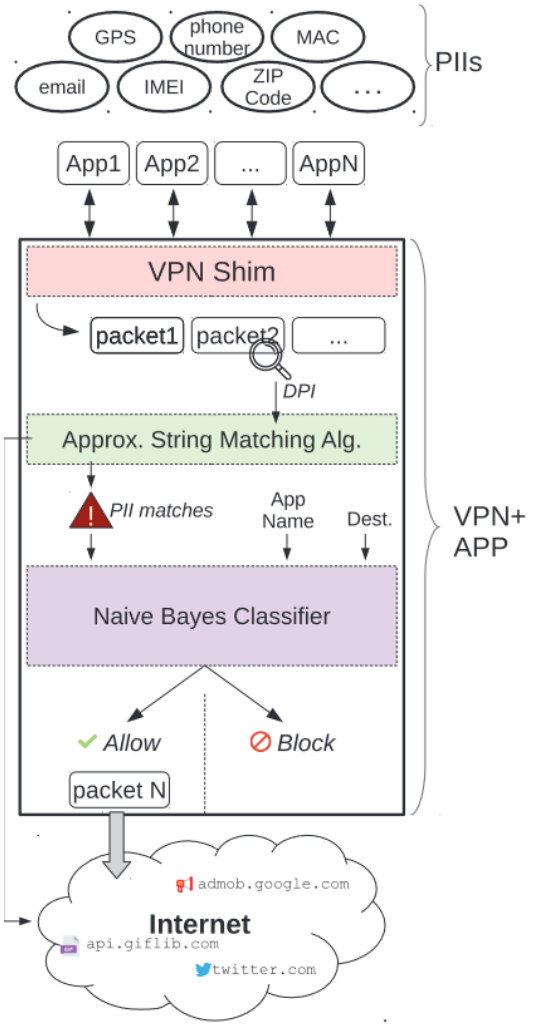



Figura 10: Arquitectura de *VPN+*. Fuente: [Novak *et al.*, 2020]

# CAPÍTULO 3
# PROPUESTA DE SOLUCIÓN

## 3.1. Análisis de Técnicas Modernas

Condensando lo mencionado en el capítulo anterior, a día de hoy, no parece haber una técnica de *Taint Analysis* ni de *Behaviour-based Monitoring* que pueda ser implementada en el entorno actual de Android. Las técnicas expuestas, si bien proponen alternativas interesantes, generalmente requieren de algún tipo de instrumentación importante, ya sea a nivel de dispositivo o a nivel de aplicación, o bien, requieren de funcionalidades que, a la fecha, se encuentran obsoletas.

En primer lugar, el gran problema tanto de *TaintART* como de su predecesor, es que su instrumentación requiere de un dispositivo *rooteado*, es decir, un dispositivo que haya sido modificado para obtener un control privilegiado del mismo y que, de esta forma, permita modificar las limitaciones establecidas por el fabricante. Si bien la idea de *rootear* un *smartphone* puede sonar llamativa, es un proceso delicado y que debe ser realizado por personas capacitadas. Sin mencionar que además, conlleva una serie de riesgos y desventajas, siendo la principal que el dispositivo se vuelve más vulnerable a ciertos tipos de *malware*. Por consiguiente, estas técnicas en particular no presentan una alternativa interesante al problema.

Ahora bien, *TaintMan* definitivamente dio un paso adelante en el desarrollo de una técnica de *Taint Analysis* más moderna e incluso apta para los usuarios finales, ya que no depende de un dispositivo rooteado para su funcionamiento. Lamentablemente, la instrumentación que propone esta técnica no es lo suficientemente sencilla. Además, el proceso de ingeniería inversa requerido para el re-empaquetado de las aplicaciones a analizar, puede traer complicaciones para aquellas que necesiten validar la integridad de sus paquetes en *runtime*. Si bien, [You *et al.*, 2017] propone una forma de eludir dicha validación, significaría complicar aún más la instrumentación necesaria. Además, las técnicas que aparecieron posteriormente, se construyeron siguiendo enfoques similares a *TaintMan* y por lo tanto, no aportan con nuevas alternativas para abordar este tipo de análisis. Por si fuera poco, *TaintMan* fue construida para funcionar con la versión 5.0.2 de Android y corresponde a una de las alternativas más actuales en cuanto a técnicas basadas en *Taint Analysis*, lo que dificulta aún más su uso y el de sus predecesores.

Tomando en consideración lo anterior, se opta por la implementación de una técnica basada en *Behaviour-based Monitoring*, que se adecúa en mayor medida a las necesidades del problema y resultaría en una solución más idónea. Con el objetivo de contar con una visión un poco más detallada de las diferencias de las técnicas que fueron analizadas en la sección anterior, se elaboró la tabla 4 que se presenta a continuación.

| **Criterio\Técnica** | **Recon** | **AntMonitor** | **AntShield** | **VPN+** |
|---|---|---|---|---|
| **Año** | 2016 | 2016 | 2018 | 2020 |
| **Versión de Android** | 5.1.1 | 5.0 | 7.1.1 | 8.1 |
| **VPN / Proxy** | Remoto | Local | Local | Local |
| **Detección de PII** | Desconocidos | Predefinidos | Predefinidos y Desconocidos | Predefinidos |
| **Uso de Machine Learning / String Matching** | Machine Learning | String Matching | Ambos | Ambos |
| **Modelo / Algoritmo** | Decision Trees C4.5 | Aho-Corasick | Binary Relevance con Decision Trees y Aho-Corasick | Boyer-Moore (Levenshtein Distance) y Naive Bayes |
| **Dataset** | Sí | No aplica | Sí | No |

Tabla 4: Comparación de técnicas basadas en Behavior-based Monitoring. Fuente: Elaboración Propia.

Naturalmente, las técnicas más nuevas corresponden a alternativas más interesantes para la implementación de una nueva técnica de análisis dinámico. Si bien, cada una aporta con estudios e informaciones relevantes, *AntShield* se posiciona como la técnica más llamativa tanto en términos de resultados como de alcance, ya que es la única que cuenta con un enfoque híbrido en lo que respecta a identificación de PII. Sin embargo, no deben ignorarse las mejoras propuestas por *VPN+*, tanto para la detección de *strings* como en términos de usabilidad, ya que éstas la convierten en una alternativa interesante al momento de plantear una propuesta de solución.

Ahora bien, cabe señalar que en la actualidad la mayor parte del tráfico de red se encuentra cifrado de extremo a extremo mediante protocolos como SSL/TLS[14], lo que añade una capa de seguridad adicional. En esta línea, [Android Developers, 2024b] indica una serie de cambios que fueron introducidos en Android 10.0 y 11.0 en lo que respecta a SSL y TLS, con el objetivo de volver más robusta la seguridad de la información en tránsito. Tomando esto en cuenta, el gran problema radica en que ambas técnicas mencionadas fueron desarrolladas para versiones más antiguas de Android, lo que supone ciertos problemas de compatibilidad con versiones más actuales. En particular, *AntShield* fue desarrollado para funcionar con la versión 7.1.1 de Android y en su repositorio oficial se indica que para asegurar su funcionamiento con versiones más recientes a esa, son requeridas herramientas alternas para poder interceptar correctamente el tráfico SSL/TLS. Por otro lado, *VPN+*, si bien es más reciente y data del 2020, fue desarrollada para la versión 8.1 de Android, por lo que lógicamente presenta los mismos problemas que *AntShield* con las versiones más actuales. Esto se debe a que dichas técnicas utilizaban en cierta medida las vulnerabilidades presentes en versiones

[14]Secure Sockets Layer / Transport Layer Security. Protocolos de seguridad para cifrar la comunicación en Internet, garantizando la privacidad de los datos transmitidos entre un cliente y un servidor. TLS es la versión más segura y actualizada de SSL.

antiguas de Android para hacer posible la intercepción y análisis del tráfico saliente. Naturalmente, el no poder explotar dichas vulnerabilidades dejaron completamente inhabilitadas a técnicas como *AntShield* y *VPN+* luego de que la seguridad se robusteciera. Una mayor restricción en el uso de certificados externos para establecer conexiones con servidores y el uso de nuevas técnicas de seguridad como el *Certificate Pinning*[15], ayudó a reducir enormemente los ataques MITM[16], pero trajo consigo la incapacidad de filtrar el tráfico y así evitar fugas de información. No obstante, si bien la información en tránsito puede verse más resguardada con las medidas de seguridad actuales sobre todo cuando existe algún tipo de cifrado de extremo a extremo, aún corre el riesgo de ser comprometida cuando los datos son recibidos por un servidor malicioso.

Considerando que [Stat Counter, 2024] indica que a la fecha más de un 85 % de las personas que cuentan con un dispositivo con Android utilizan versiones más nuevas a la 8.1, se dificulta la utilización de cualquiera de las técnicas mencionadas, ya que éstas se restringen a versiones más antiguas. Por lo tanto, la solución que se proponga debe ser necesariamente compatible con versiones más recientes de Android, preferiblemente con versiones más recientes a la 11.0, y que en lo posible no requiera de un análisis del *payload* de los paquetes.

## 3.2. Consideraciones

Tomando en cuenta lo anteriormente mencionado, para la propuesta de solución se deben tener en consideración los siguientes puntos:

- Las técnicas de análisis dinámico que pertenecen a la categoría de *Behaviour-based Monitoring*, específicamente las que se basan en el análisis de tráfico de red, sugieren ser las más efectivas para detectar fugas de información.
- En la actualidad, la mayor parte del tráfico de red se encuentra encriptado con la debida finalidad de resguardar la información de posibles ciberdelincuentes.
- Las actualizaciones introducidas las versiones de Android 10.0 y 11.0 hacen imposible el análisis de tráfico de red encriptado en un teléfono no rooteado, causando que las técnicas que utilizaban este método quedaran obsoletas.
- Rootear un dispositivo real generalmente introduce nuevos vectores de ataque, exponiendo al usuario y su información a nuevas amenazas.

[15] Técnica de seguridad en la que una aplicación móvil almacena información específica sobre el certificado digital de un servidor al que se conecta, y verifica que el certificado presentado coincida con la información almacenada, reduciendo así el riesgo de ataques de intermediarios malintencionados.

[16] Man-in-the-Middle. Ataque cibernético donde un atacante se interpone en la comunicación entre dos partes para interceptar o manipular la información sin que ellas lo sepan.

## 3.3. Propuesta

Teniendo en cuenta el análisis y las consideraciones previas, se ha determinado que es inviable implementar cualquiera de las técnicas revisadas, puesto que ninguna cumple con todos los requisitos del estudio. La obsolescencia de estas técnicas hace imposible su adaptación directa a versiones más recientes de Android y plantea la necesidad de desarrollar una solución desde cero. Teniendo esto en cuenta, es crucial que la propuesta al menos sea compatible con las versiones recientes de Android, pues es de suma importancia que los resultados obtenidos sean representativos del estado actual de la privacidad de las aplicaciones. Por esto, también es esencial tener acceso al *payload* de los paquetes salientes, puesto que garantiza una mayor precisión en la identificación de la información que se está filtrando, evitando así depender de predicciones basadas en otras métricas.

Por lo tanto, se propone implementar una técnica de análisis dinámico en un entorno controlado, específicamente en un dispositivo virtual, siendo esta la única forma de cumplir con las restricciones planteadas. Este acercamiento permitirá superar las limitaciones de trabajar directamente en un dispositivo real, aunque signifique renunciar a la posibilidad de implementar una técnica directamente disponible para el usuario general y que funcione en tiempo real. No obstante, de esta manera el estudio podrá focalizarse en comprender a cabalidad el comportamiento y el contexto en el que las aplicaciones filtran información.

A modo general, la propuesta de la técnica de análisis dinámico a implementar se separa en dos etapas principales. La primera se compone de cuatro procesos distintos y se repetirá para todas las aplicaciones que se pretendan analizar.

1. **Descarga de la Aplicación**
   Naturalmente, el primer paso consiste en descargar la aplicación que se desea analizar desde la Google Play Store.

2. **Instrumentación**
   Generalmente, las aplicaciones implementan prácticas de seguridad que inhabilitan cualquier tipo de captura de tráfico de red. Es por esto que, habiendo realizado la descarga, se procede a realizar un proceso de instrumentación de la aplicación con el apoyo de una herramienta previamente seleccionada. Esta herramienta se presenta más en detalle en la sección 3.4 y permitirá deshabilitar aquellas restricciones, y por ende, habilitar la captura del tráfico de red.

3. **Ejecución de la Aplicación**
   Realizado lo anterior, se procede a ejecutar la aplicación durante un intervalo de tiempo que permita realizar los flujos principales y, de esta manera, generar una cantidad suficiente de tráfico de red.

4. **Obtención del Tráfico de Red**
   Habiendo pasado el intervalo de tiempo necesario, el tráfico de red generado será debidamente exportado en un archivo de texto para su posterior análisis.

Por el otro lado, la segunda etapa está más enfocada en trabajar los datos obtenidos y se separa en cinco distintos procesos que se detallan a continuación.

5. **Recolección del Tráfico de Red**
   Primero, se recolecta todo el tráfico de red de las aplicaciones analizadas.

6. **Ejecución de Algoritmo de String Matching**
   Cada archivo de tráfico de red generado es debidamente examinado mediante un algoritmo de *string matching* que detecta si los *payloads* de los paquetes contienen información sensible en formato de PII. Para cada ejecución del algoritmo se elabora un archivo de salida que resume todas las potenciales fugas de información que fueron identificadas en el archivo de entrada.

7. **Construcción de Reporte de Fugas de Información**
   A partir de los archivos de salida generados por el algoritmo de *string matching* para cada aplicación, se construye un reporte general que reúna los hallazgos de todas las aplicaciones analizadas, con la debida finalidad de analizar dichos datos de manera más centralizada.

8. **Depuración del Reporte y Clasificación de Fugas**
   Este informe es refinado y perfeccionado utilizando información de diversas fuentes, facilitando la categorización de los PII encontrados, los servidores a los que se conecta la aplicación y las prácticas de seguridad seguidas por cada aplicación según lo indicado en Google Play Store. Posteriormente, con el informe consolidado, se le asigna un nivel de riesgo a cada evento según sus características, permitiendo clasificar los eventos como fugas efectivas de información, o bien, como tráfico benigno.

9. **Análisis de Resultados**
   Finalmente, se analizan los resultados obtenidos con el objetivo de estudiar y comprender el comportamiento actual de las aplicaciones en cuanto a la filtración de datos sensibles, así como el nivel de privacidad con el que cuentan en la actualidad.

Antes de presentar más en detalle la metodología que se empleará en la propuesta de solución, se ofrece una visión general de cómo se organizan y coordinan los distintos procesos y técnicas utilizadas en la figura 11.

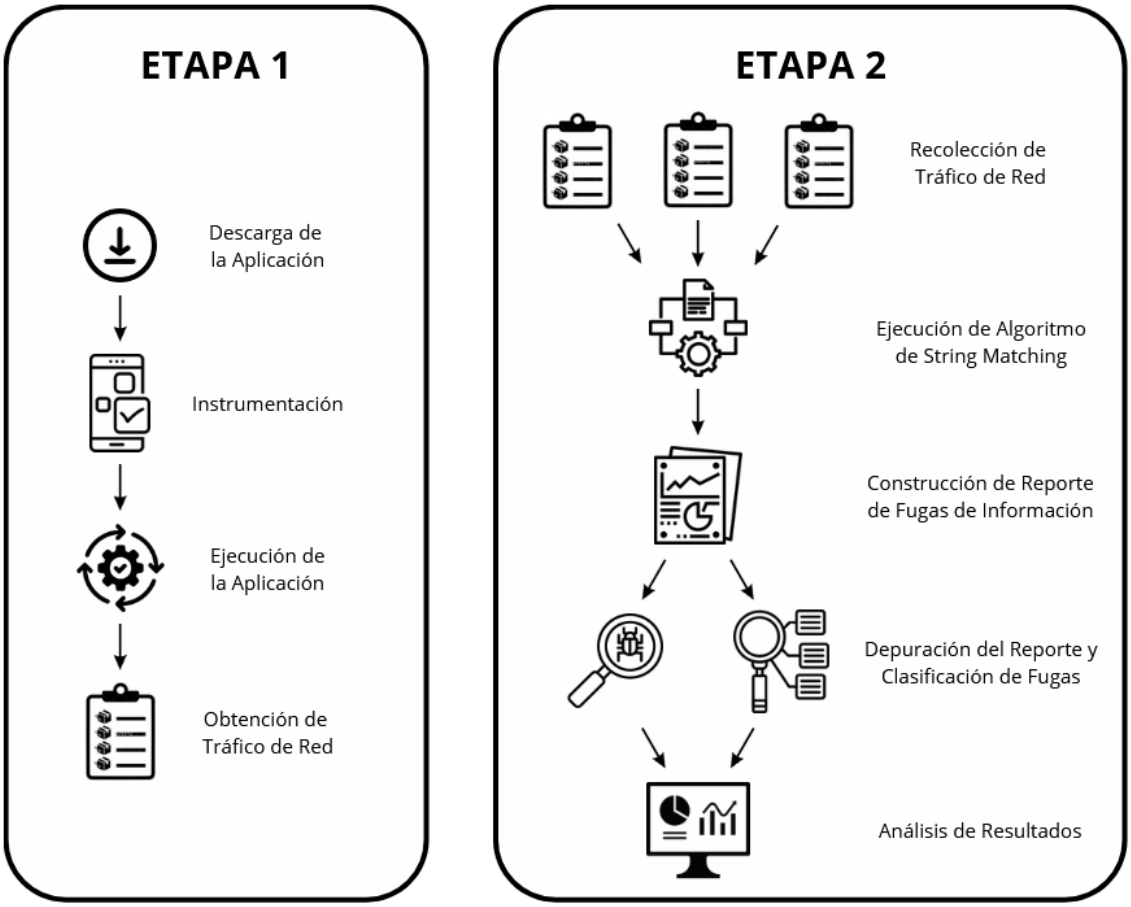



Figura 11: Propuesta de Solución. Fuente: Elaboración Propia.

## 3.4. Selección de Tecnologías

A continuación, se presentan las herramientas que constituirán la base de la propuesta de solución propuesta.

- **MobSF**

  Mobile Security Framework (MobSF, por sus siglas en inglés) es una plataforma de investigación de seguridad para aplicaciones móviles en Android, iOS y Windows Mobile. MobSF se puede utilizar para múltiples casos de uso, como seguridad de aplicaciones móviles, pruebas de penetración, análisis de malware y análisis de privacidad, contando con herramientas tanto de análisis estático como de análisis dinámico. Éstas son compatibles con aplicaciones tanto de Android como de iOS, y ofrece una plataforma para pruebas interactivas instrumentadas, análisis de datos en tiempo de

ejecución y tráfico de red. Además, esta plataforma cuenta con una forma automatizada de instrumentar aplicaciones con la ayuda de la herramienta *Frida*[17], lo que permite *bypassear* el *certificate pinning* y *root detection*[18] implementado por algunas aplicaciones. Asimismo, si bien ofrece la capacidad de obtener el tráfico de red descifrado solamente en casos específicos, sí permite obtener una serie de otros resultados y datos que pueden ser de gran utilidad para la investigación y el estudio de la seguridad de las aplicaciones que se analicen.

- **Genymotion**

  Genymotion es una plataforma de virtualización que permite a los desarrolladores probar y depurar aplicaciones en una amplia gama de dispositivos virtuales, ofreciendo emulación de dispositivos con diferentes versiones de Android y configuraciones de hardware. Su principal ventaja radica en su compatibilidad con herramientas de desarrollo populares y en este caso en particular, con MobSF. De hecho, Genymotion figura como el único emulador que permite virtualizar dispositivos con versiones de Android superiores a Android 9.0 a través de MobSF, llegando hasta Android 11.0, adecuándose correctamente a los requerimientos de la propuesta.

Aunque las herramientas mencionadas previamente constituyen la base de la propuesta, es importante destacar aquellas que también complementarán la solución final. En esta línea, se confecciona la tabla 5 que resume las tecnologías que se utilizarán a lo largo del estudio.

| Nombre | Tipo | Versión | Uso |
|---|---|---|---|
| **MobSF** | Framework | 3.9.7 | Instrumentación de las aplicaciones a analizar y recolección del tráfico de red. |
| **Genymotion** | Software | 3.6.0 | Emulador de Android para la ejecución de las aplicaciones a analizar. |
| **Docker** | Software | 26.1.0 | Levantamiento y ejecución de MobSF en un contenedor local. |
| **Python** | Lenguaje de Programación | 3.10.12 | Implementación de algoritmo de String Matching. |
| **MatPlotLib** | Librería de Python | 3.8.2 | Creación y personalización de gráficos. |

Tabla 5: Resumen de Tecnologías seleccionadas. Fuente: Elaboración Propia.

[17]Herramienta que permite analizar y modificar aplicaciones Android en tiempo de ejecución, facilitando tareas como la depuración, el análisis de seguridad y la manipulación del comportamiento de las aplicaciones.

[18]Proceso de verificar si un dispositivo tiene permisos de *root* o acceso de superusuario. Se utiliza para proteger las aplicaciones de posibles riesgos de seguridad asociados con dispositivos *rooteados*.

## 3.5. Selección de Aplicaciones

Para las pruebas, los estudios revisados en la sección 2.2.3, generalmente seleccionaban una cantidad restringida de aplicaciones, otorgando una vista más reducida de las fugas de información que ocurrían a través de ellas. En cambio, para este estudio serán consideradas las aplicaciones más populares de cada categoría definida por Google Play Store. Esto, con el objetivo de evaluar el posible alcance que tendrían fugas de información en las aplicaciones más ampliamente utilizadas por los usuarios. Además, el hecho de considerar toda la gama de categorías permite enriquecer el estudio al tomar en cuenta aplicaciones de distintas naturalezas, lo que comúnmente se traduce en fugas de distintos PII. Para garantizar una comparación más justa entre categorías, se consideró el mismo número de aplicaciones por cada una y solamente aquellas que no presentaran ningún inconveniente para su análisis. Ahora bien, con el objetivo de aclarar el proceso de selección de aplicaciones, a continuación se detallan los criterios utilizados tanto para incluir como para excluir ciertas aplicaciones de ser analizadas.

- **Criterios de Inclusión**
  - Aplicaciones pertenecientes al *top* de cada categoría definida por Google Play Store. Se utilizarán las primeras 3 aplicaciones que no tengan conflicto con alguno de los criterios de exclusión.
- **Criterios de Exclusión**
  - Aplicaciones que requieran algún tipo de suscripción.
  - Aplicaciones que dependan de herramientas adicionales como *smartwatches*, al igual que aquellas diseñadas específicamente para vehículos.
  - Aplicaciones cuyo uso se puede ver restringido por las limitaciones intrínsecas que conlleva la utilización de un dispositivo virtual.

Tomando en cuenta lo anterior, se evaluarán un total de 96 aplicaciones. Éstas se encuentran debidamente detalladas en el anexo A.

## 3.6. Metodología

Antes de detallar los distintos procesos de la metodología propuesta, y con el objetivo de entregar una visión más general y completa de la interacción entre los componentes que la conforman, se presenta el diagrama de la figura 12. Cabe destacar, que el diagrama detalla los pasos como fueron definidos en la sección 3.3.

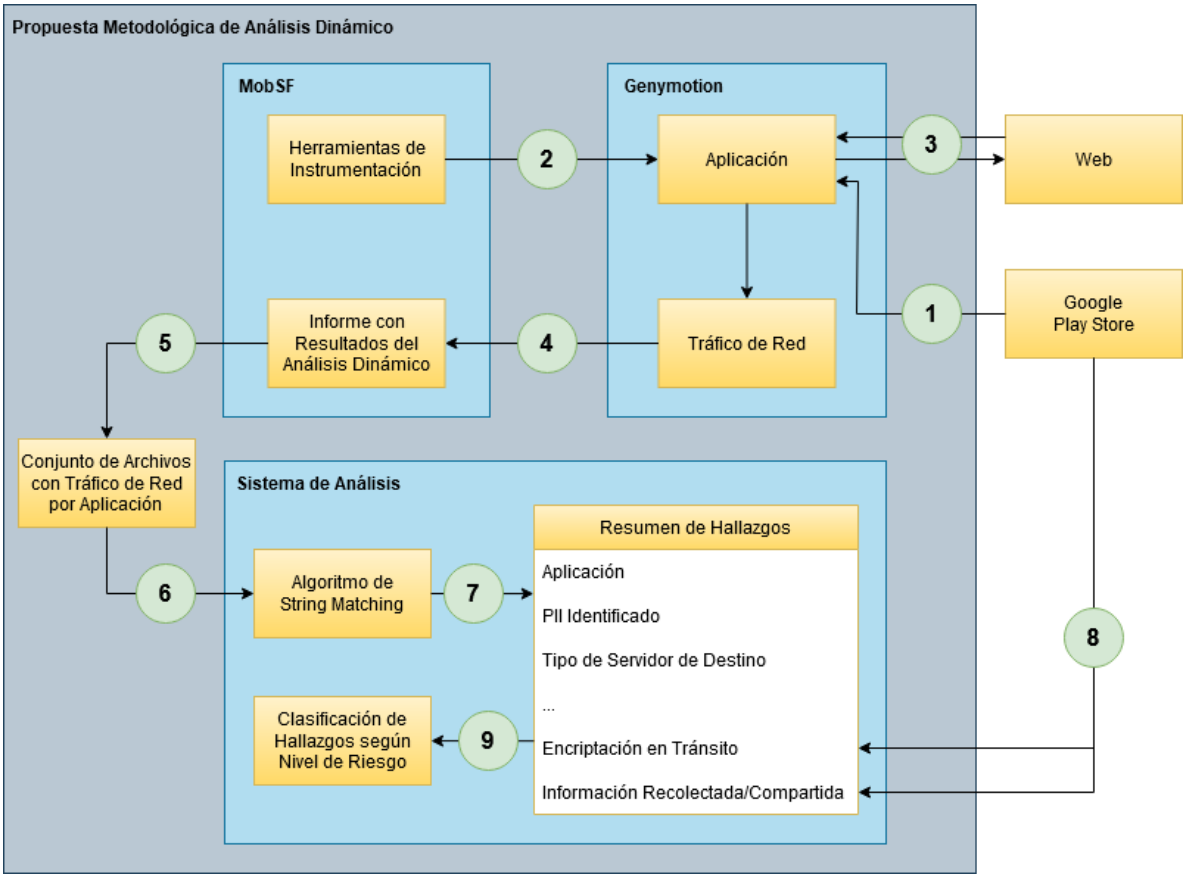



Figura 12: Interacción de Componentes. Fuente: Elaboración Propia

### 3.6.1. Ejecución de Aplicaciones y Obtención de Tráfico de Red

Como bien se mencionó anteriormente, lo primero es descargar la aplicación que se pretende analizar. Hecho esto, se procede a instrumentar la aplicación seleccionada a través de MobSF. Esta herramienta tiene la capacidad de orquestar dicha aplicación, inhabilitando el *root detection* y el *certificate pinning* presentes en la mayoría de aplicaciones. Como se mencionó en la sección 3.4, este proceso es realizado automáticamente por MobSF, sin embargo, consiste fundamentalmente de la ejecución de *Frida*, una herramienta que permite modificar las aplicaciones de Android en tiempo de ejecución a través de una serie de comandos, lo que finalmente permite *bypassear* aquellas restricciones que impiden la

captura del tráfico. Al terminar el proceso, la aplicación se ejecuta automáticamente y la herramienta comience a recolectar el tráfico de red generado mientras ésta se encuentra en uso. Ahora bien, para esta parte del estudio, se optó que por cada aplicación se realizara este procedimiento un total de dos veces, separando el proceso de creación de cuenta y el uso normal de la aplicación. Este paso es crucial debido a que durante la creación de una cuenta, se requiere que el usuario proporcione una serie de datos e información personal, los cuales posteriormente son enviados fuera del dispositivo y aparecerán en el tráfico de red capturado. Diferenciar estos flujos permitirá categorizar de manera más sencilla aquella información que fue ingresada explícitamente por el usuario cuando éste crea su cuenta, y por otro lado, la información que abandona el dispositivo de manera más pasiva cuando el usuario hace uso de la aplicación después de haber creado su cuenta. Naturalmente para aquellas aplicaciones que no requieran de una cuenta para ser utilizadas, el procedimiento se realizará una sola vez. En cuanto al tiempo necesario para generar tráfico suficiente, el proceso de creación de cuenta se extenderá por el tiempo que éste demore, mientras que para el uso general de la aplicación, se consideró un intervalo de entre 3 y 5 minutos, lo cual se estimó como un tiempo adecuado para cubrir los flujos esenciales. De esta manera, la idea principal es interactuar con la aplicación durante el intervalo de tiempo fijado como lo haría un usuario normal, preferiblemente tratando de realizar la mayor cantidad de acciones distintas posibles.

### 3.6.2. Análisis de Tráfico y Elaboración de Reporte

Habiendo finalizado con el proceso de recolección del tráfico HTTP/HTTPS de todas las aplicaciones, se procederá a recolectar y analizar dicho tráfico mediante un algoritmo de *string matching*. Éste se empleará para detectar si los paquetes salientes contienen o no alguno de los PII que serán utilizados para el estudio. En el caso de encontrarse alguno de ellos, dicho evento se categorizará debidamente como una posible fuga de información.

Si bien en las secciones 2.1.7 y 2.2.3, se hizo alusión a una serie de algoritmos de *string matching* que podrían ser utilizados para la detección de PII, es importante destacar que estos algoritmos fueron implementados para un análisis en tiempo real y en un dispositivo común y corriente, donde los recursos generalmente son limitados. *Aho-Corasick* y *Boyer-Moore* se implementaron puesto que correspondían a alternativas más eficientes dado el contexto, pero no son más eficaces que un algoritmo de *string matching* sin modificaciones. Por lo tanto, el algoritmo en cuestión se construirá basándose en la técnica utilizada por *VPN+*, que sustenta su algoritmo de *string matching* en el cálculo de la distancia Levenshtein, medida que permite identificar similitudes entre *strings* incluso si presentan pequeñas variaciones y cuya implementación sí entrega valor. Este enfoque resultará útil para manejar casos en los que se utilicen codificaciones que modifiquen levemente los datos, como podría serlo el caso del *percent encoding*[19].

[19]Método que codifica caracteres especiales en URLs usando el signo de porcentaje seguido de dos dígitos hexadecimales, asegurando su correcta transmisión e interpretación en la web.

Con el objetivo de consolidar la información recabada, ésta será depurada y resumida en un informe final. Cada entrada de dicho informe incluirá la información detallada de cada evento, la cual permitirá establecer la probabilidad de que los hallazgos efectivamente sean considerados como fugas de información. La información que se recolectará de cada evento se resume a continuación.

- **Nombre de la Aplicación.**
- **Flujo.** Campo que indica si el hallazgo corresponde al proceso de creación de cuenta o al uso normal de la aplicación. Puede tener dos valores: Cuenta o Uso.
- **Destino.** URL a la que se envía la solicitud HTTP/HTTPS del paquete en cuestión.
- **Tipo de Servidor.** Campo que indica si el servidor corresponde a uno propio de la aplicación o a un servidor de terceros. Puede tener dos valores: Servidor Propio o Servidor de Terceros.
- **Categoría del Servidor.** Categoría asignada al servidor según su naturaleza. Puede tener cinco valores: Servidor Propio, Utilidades, Analítica, Publicidad u Otro.
- **Encriptación en Tránsito.** Campo que indica si la información cuenta con encriptación en tránsito, según lo estipulado por la página de la aplicación en Google Play Store. Puede tener dos valores: Verdadero o Falso.
- **PII.** Dato encontrado.
- **Categoría del PII.** Categoría asignada al PII. Puede tener cinco valores: Información Personal (Más Sensible), Información Personal (Menos Sensible), Ubicación Precisa, Ubicación Estimada e Identificadores del Dispositivo/Usuario. Detallados en la tabla 6.
- **Input del Usuario.** Campo que indica si el PII específico fue ingresado explícitamente por el usuario, generalmente en el proceso de creación de cuenta. Puede tener dos valores: Verdadero o Falso.
- **Información Recolectada/Compartida. (Google Play Store)** Campo que indica si los desarrolladores de la aplicación señalan explícitamente que el PII en cuestión podría ser recolectado o compartido. Esta información está disponible en la página de la aplicación en Google Play Store y se considera que fue recolectado si se trata de un servidor propio y recolectado si se trata de un servidor de terceros. Puede tener dos valores: Verdadero o Falso.
- **Total.** Cantidad de apariciones de dicho evento. Se considerará como idéntico un evento cuando una aplicación filtre el mismo PII a un mismo destino.

### 3.6.3. Clasificación de Eventos

Como último paso, se propondrá una forma de categorizar los hallazgos identificados durante el proceso de análisis. La intención de proponer un sistema de categorización para los eventos identificados, nace de la necesidad de diferenciar el tráfico potencialmente malicioso del tráfico benigno. En la literatura estudiada es común que los resultados se focalicen únicamente en mostrar la cantidad de hallazgos. Si bien, dichos hallazgos generalmente sí son categorizados según el PII en particular que se filtró, y a veces según el servidor al que fue filtrada la información, no asignan un nivel de criticidad a éstos y tampoco ahondan mucho más en otros aspectos relacionados al evento. Es más, [Shuba *et al.*, 2018] plantea esto como trabajo a futuro y valida la falta de algún tipo de categorización para distinguir el riesgo asociado a cada hallazgo.

La estrategia principal consiste en asignar un grado de importancia a cuatro de los atributos detallados en la sección anterior. Éstos funcionarán como indicadores para determinar si el hallazgo respectivo representa o no una fuga de información, o más bien, para determinar el riesgo que supone dicho evento. Cada atributo tendrá una ponderación distinta en el cálculo del nivel de riesgo final, como también lo tendrán los valores de dichos atributos, que oscilarán entre 0 y 1. Por su parte, 0 indicará un bajo nivel de criticidad o riesgo, mientras que 1 indicará un nivel alto. Si un atributo sólo tiene dos valores posibles, se asignarán los valores extremos (0 y 1). En caso de tener más de dos valores, se utilizarán valores intermedios entre 0 y 1 para representar su nivel de riesgo de manera más precisa. Dicho esto, los atributos seleccionados, sus posibles valores y sus respectivos niveles de riesgo, se presentan más en detalle a continuación.

- **Encriptación en Tránsito**

  Este atributo es, indudablemente, el más importante de los cuatro que fueron seleccionados. Si la aplicación en cuestión no implementa cifrado en tránsito, la información transmitida es susceptible a ser interceptada fácilmente por algún tercero, frecuentemente con intenciones maliciosas. La falta de encriptación en tránsito aumenta la vulnerabilidad de los datos frente a accesos no autorizados y potencia el riesgo de ataques MITM, donde un posible atacante es capaz de alterar o robar datos mientras éstos se transmiten. Considerando que los únicos valores posibles para este atributo son de Verdadero y Falso, se les asigna un riesgo de 0.0 y 1.0, respectivamente.

- **Categoría del PII**

  En este caso, para determinar de mejor manera el riesgo general, los PII fueron agrupados como se muestra a continuación, ordenados según su nivel de criticidad, de manera creciente y en línea con lo mencionado en la sección 2.1.1. Además, éstos se presentan de manera más detallada en la tabla 6.

  - **Información Personal (Menos Sensible)**
    Aunque el género y la fecha de nacimiento corresponden a datos personales, son

menos peligrosos en términos de consecuencias inmediatas de una filtración. La única manera en que la fuga de estos datos significara una verdadera amenaza, sería en combinación con otros datos.

- **Ubicación Estimada**
  Esta categoría, mientras que puede indicar un área general en la que vive una persona, no proporciona necesariamente una ubicación específica y suele ser accesible públicamente de distintas fuentes, incluyendo guías telefónicas o registros electorales.
- **Identificadores del Dispositivo/Usuario**
  Estos identificadores representan un nivel más elevado de riesgo que las categorías previamente mencionadas, puesto que pueden emplearse para monitorear la actividad de un dispositivo a través de distintos servicios. Esto no solamente permite hacer un seguimiento detallado del usuario, sino que también se puede hacer un perfilamiento completo si se asocia a otros tipos de datos.
- **Ubicación Precisa**
  Datos como la longitud y latitud corresponden a datos altamente sensibles y con justa razón, ya que la integridad física de una persona puede verse comprometida al revelar exactamente dónde se encuentra y los lugares que frecuenta.
- **Información Personal (Más Sensible)**
  Esta categoría es potencialmente la más peligrosa en caso de filtrarse, puesto que podrían permitir a alguien acceder a cuentas y servicios personales de un usuario, resultando en posibles suplantaciones de identidad o incluso en la realización de estafas.

| **Categoría** | **PII** | **Riesgo** |
|---|---|---|
| Información Personal (Menos Sensible) | Género<br>Fecha de Nacimiento | 0.1 |
| Ubicación Estimada | Código Postal<br>Ciudad<br>Región<br>País | 0.25 |
| Identificadores del Dispositivo/Usuario | ID de Publicidad<br>ID de Android<br>IMEI | 0.5 |
| Ubicación Precisa | Latitud / Longitud | 0.75 |
| Información Personal (Más Sensible) | Nombre<br>Número de Teléfono<br>Mail<br>Nombre de Usuario<br>Contraseña | 1.0 |

Tabla 6: Categorización de PII. Fuente: Elaboración Propia.

- **Categoría de Servidor**

  Al igual que en el atributo anterior, las categorías de los servidores fueron debidamente agrupadas. A continuación, se detalla brevemente a qué servidores alude cada una de las categorías indicadas y la razón por la que representan su nivel respectivo de riesgo en la tabla 7.

  - **Propio**
    Los servidores propios, al ser gestionados por la misma entidad que recopila los datos, suelen tener políticas de seguridad y privacidad diseñadas específicamente para proteger la información. Por lo tanto, el riesgo suele ser relativamente bajo.
  - **Utilidades**
    Los servidores enfocados en utilidades generalmente suelen utilizarse para tareas específicas como verificación de identidad, obtención de recursos multimedia, integración de servicios con terceros, entre otros. Si bien, suelen seguir estrictos estándares de seguridad, en ocasiones pueden manejar información sensible, por lo que la dependencia de terceros lleva consigo un riesgo de exposición.
  - **Analítica / Publicidad**
    Los servidores dedicados a analítica y publicidad a menudo recopilan y procesan grandes volúmenes de datos personales con el objetivo de realizar estudios de mercado y personalizar la experiencia de los usuarios. Ahora bien, en este caso el riesgo es alto debido al amplio alcance de la recopilación de datos y la posibilidad de que la información sea compartida con otras entidades. La privacidad del usuario puede verse altamente comprometida si estos datos son mal gestionados.
  - **Otro**
    Esta categoría engloba aquellos servidores cuya naturaleza no pudo ser determinada con certeza. Ante la duda, se considera este tipo de servidores como los más peligrosos y con el mayor riesgo de incurrir en fugas de información y uso mal intencionado de datos.

| **Categoría** | **Riesgo** |
|---|---|
| Propio | 0.0 |
| Utilidades / API | 0.4 |
| Analítica / Publicidad | 0.7 |
| Otro | 1.0 |

Tabla 7: Categorización de Servidores. Fuente: Elaboración Propia.

- **Información Recolectada/Compartida (Google Play Store)**

  Como se mencionó en la sección 3.6.2, este atributo especifica si los desarrolladores de la aplicación declararon explícitamente que cierto PII podría ser recolectado o compartido. La ausencia de tal declaración en la descripción de la aplicación en Google Play Store sugiere que los desarrolladores podrían no estar cumpliendo con las políticas establecidas por Google. De esta manera, lo mencionado podría interpretarse como un intento de ocultar la recolección y/o compartición de datos personales, posiblemente con la intención de utilizarlos para fines maliciosos. Nuevamente, considerando que los valores posibles para este atributo son de Verdadero y Falso, se les asigna un riesgo de 0.0 y 1.0, respectivamente.

### 3.6.4. Cálculo del Riesgo

Tomando en cuenta los atributos seleccionados, se procede a calcular un valor final que refleje el riesgo asociado al evento en el que, potencialmente, se filtró información. Como se mencionó anteriormente, a cada atributo se le asigna un peso específico, para cuantificar su importancia y evaluar más precisamente su impacto en el riesgo total.

Dada la importancia de que una aplicación implemente algún tipo de encriptación en tránsito, se han desarrollado dos métodos distintos para calcular el riesgo final. Por un lado, el caso en que sí exista dicha encriptación y por el otro lado, el caso en que no. Para ambos casos, la importancia de los atributos quedaría distribuida como se muestra en la tabla 8. Naturalmente, cuando los datos se transmiten sin encriptación, el número de actores potenciales que pueden interceptarlos durante su transmisión incrementa considerablemente, volviendo irrelevante la identidad del receptor final. En estos escenarios, el peso de la Categoría del Servidor se reduce a 0.0, mientras que Encriptación en Tránsito se mantiene en 0.5. Por el contrario, cuando sí existe cifrado en tránsito, el riesgo principal se concentra en la categoría del servidor que los recibe y en la posibilidad de que éste tenga intenciones maliciosas y/o utilice los datos para fines más allá de los estrictamente necesarios. En estos casos, la importancia de los atributos se invierten, es decir, el peso asignado a **Encriptación en Tránsito** se minimiza a 0.0 y el asignado a **Categoría del Servidor** aumenta a 0.5.

| **Atributo** | **Peso (Con Encriptación en Tránsito)** | **Peso (Sin Encriptación en Tránsito)** |
|---|---|---|
| Encriptación en Tránsito | 0.0 | 0.5 |
| Categoría del PII | 0.35 | 0.35 |
| Categoría del Servidor | 0.5 | 0.0 |
| Información Recolectada/Compartida | 0.15 | 0.15 |

Tabla 8: Pesos por Atributo. Elaboración Propia.

Después de haber detallado el peso y el valor asociado al riesgo de cada categoría, se puede proceder a calcular el valor del riesgo final. Este cálculo se realiza sumando los pro-

ductos del peso de cada atributo por el valor de riesgo correspondiente a ese atributo. Así, el valor final del riesgo queda definido por la siguiente expresión.

$$R_T = W_E \cdot (1 - R_E) + W_P \cdot R_P + W_S \cdot R_S \cdot R_E + W_{RC} \cdot R_{RC} \quad (1)$$

donde

$R_T \rightarrow$ Riesgo Total del Evento
$W_E \rightarrow$ Peso Asociado al Atributo **Encriptación en Tránsito**
$W_P \rightarrow$ Peso Asociado al Atributo **Categoría del PII**
$W_S \rightarrow$ Peso Asociado al Atributo **Categoría del Servidor**
$W_{RC} \rightarrow$ Peso Asociado al Atributo **Información Recolectada/Compartida**
$R_E \rightarrow$ Riesgo Asociado al Valor del Atributo **Encriptación en Tránsito**
$R_P \rightarrow$ Riesgo Asociado al Valor del Atributo **Categoría del PII**
$R_S \rightarrow$ Riesgo Asociado al Valor del Atributo **Categoría del Servidor**
$R_{RC} \rightarrow$ Riesgo Asociado al Valor del Atributo **Información Recolectada/Compartida**

Aunque esta metodología para calcular el riesgo se aplicará a la mayoría de los casos, se tendrá en cuenta el siguiente supuesto.

- Se considerará una pequeña variación en los casos que los datos se envían a un servidor propio, la aplicación cuente con cifrado en tránsito y que se especifique en la Play Store que éstos son recolectados/compartidos. Para estos escenarios, se considerará $W_P = 0{,}0$, ya que se asume que éstos son gestionados y resguardados debidamente.

Habiendo explicado a cabalidad la manera de calcular el nivel de riesgo de cada evento, se definió un diagrama de flujo que permite realizar el cálculo de manera más sencilla en la figura 13, además de una categorización por intervalos para el valor final, que se presenta a continuación en la tabla 9.

| **Nivel de Riesgo Total** | |
|---|---|
| 0 % - 25 % | Bajo |
| 25 % - 50 % | Moderado |
| 50 % - 75 % | Alto |
| 75 % - 100 % | Crítico |

Tabla 9: Categorización de Niveles de Riesgo. Fuente: Elaboración Propia.

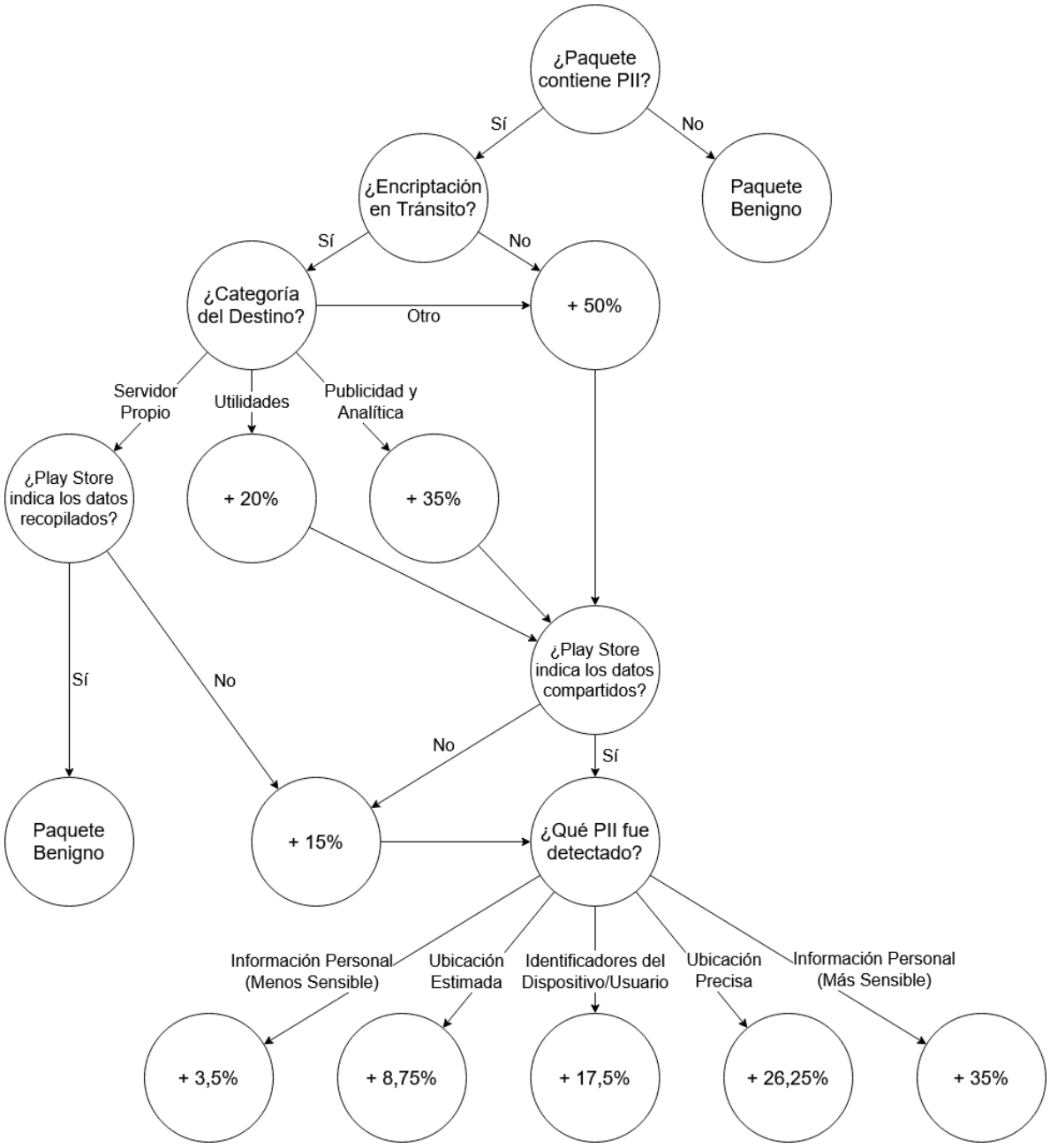



Figura 13: Diagrama de Flujo para el Cálculo del Nivel de Riesgo. Fuente: Elaboración Propia.

# CAPÍTULO 4
# IMPLEMENTACIÓN

Los procedimientos realizados en este capítulo fueron llevados a cabo en un dispositivo con sistema operativo Ubuntu 22.04.4 LTS (64-bit), con un procesador AMD Ryzen 5 5500U @ 2.1 GHz con Radeon RX Vega 7 Graphics y 16 GB de RAM. A su vez, cabe destacar que se utilizó el periodo de prueba de la versión de pago de Genymotion, con la debida finalidad de desbloquear todas sus capacidades y herramientas disponibles.

## 4.1. Creación de Perfil de Prueba

Considerando que las pruebas a realizar buscan identificar fugas de información, es importante tener en cuenta que los datos utilizados posiblemente logren filtrarse y por lo tanto podrían sean vulnerados. Por ende, se creó un perfil ficticio con datos creados específicamente para ser utilizados en el análisis de las aplicaciones y con el objetivo de no exponer información real. Dicho perfil está compuesto por los datos indicados en las tablas 10, 11, 12 y 13.

| **PII** | **Valor** |
|---|---|
| **Nombre** | Juanito Castellaneta |
| **Género** | Masculino |
| **Fecha de Nacimiento** | 4 de Mayo de 1977 |
| **Número de Teléfono** | 946661089 |

Tabla 10: Información Personal. Fuente: Elaboración Propia.

| **PII** | **Valor** |
|---|---|
| **Latitud / Longitud** | -33.0171 / -71.5544 |
| **Código Postal** | 2520000 |
| **Ciudad / Región / País** | Viña del Mar / Valparaíso / Chile |

Tabla 11: Ubicación. Fuente: Elaboración Propia.

| **PII** | **Valor** |
|---|---|
| **Mail** | memoriatest77@gmail.com |
| **Nombre de Usuario** | AngaraUser56 |
| **Contraseña** | Starwars987 / Starwars987c3po! |

Tabla 12: Credenciales. Fuente: Elaboración Propia.

| PII | Valor |
|---|---|
| **Nombre 1** | Wahid Nicoletta |
| **Número de Teléfono 1** | 5744523433 |
| **Nombre 2** | Priyanka Vishnu |
| **Número de Teléfono 2** | 5748745454 |
| **Nombre 3** | Nevena Aditi |
| **Número de Teléfono 3** | 5749787667 |

Tabla 13: Contactos. Fuente: Elaboración Propia.

## 4.2. Preparación del Entorno de Análisis

En primer lugar, para el proceso de análisis de aplicaciones móviles se habilita un dispositivo virtual con Genymotion. Como bien se mencionó, al utilizar esta herramienta a la par con MobSF, la versión más reciente de Android que puede emplearse para las pruebas corresponde a Android 11. Para este sistema operativo en particular, el dispositivo más actual que ofrece Genymotion corresponde al **Google Pixel 3a**, siendo éste el escogido para la realización de las pruebas y análisis. Ahora bien, se debe preparar el entorno antes de poder realizar las pruebas de seguridad, por lo que es necesario seguir una serie de pasos que aseguren la configuración adecuada del dispositivo virtual. En primer lugar, se crea una cuenta de Google utilizando los datos especificados previamente, siendo esta la cuenta vinculada al dispositivo. Cabe destacar que el número de teléfono que se utilizó para validar dicha cuenta pertenece a una SIM adquirida específicamente para este estudio. Por consiguiente, aquellas aplicaciones que soliciten confirmar y/o activar las cuentas a través de un número de teléfono activo se realizarán con dicho número. Seguidamente, se procede a crear contactos en el dispositivo, ingresando sus nombres y números de teléfono conforme a lo indicado en la sección anterior. Estos datos fueron creados con la debida finalidad de explorar si las fugas de información se limitan a la información personal del usuario a quien pertenece el dispositivo, o bien, también puede extenderse a información de terceros como sugieren algunos de los estudios revisados en el Estado del Arte. Naturalmente, los contactos también son ficticios para evitar filtraciones no deseadas. Posteriormente, se obtienen los identificadores únicos del dispositivo y la cuenta de Google, específicamente el ID de Publicidad, el ID de Android y el IMEI, que están sujetos a la instancia particular que se levantó, además de un número de teléfono extra que es asignado automáticamente al dispositivo. En este escenario específico, los identificadores corresponden a los que se muestran en la tabla 14.

| PII | Valor |
|---|---|
| **ID de Publicidad** | 86cfb259-46e4-446e-8e0c-3ffafff07d1d |
| **ID de Android** | 76931fac9dab2b36 |
| **IMEI** | 574859485918234 |
| **Número de Teléfono** | 15555218135 |

Tabla 14: Identificadores del Dispositivo/Usuario. Fuente: Elaboración Propia.

A través de Genymotion, se pueden obtener y modificar los valores del ID de Android y del IMEI, sin embargo el ID de Publicidad está dado por la cuenta de Google, por lo que fue recuperado a través de los ajustes del dispositivo, al igual que el número de teléfono asignado al dispositivo. La obtención de dichos datos se ilustran en las figuras 14, 15 y 16. Asimismo, a través de la configuración de Genymotion se emuló una ubicación particular dada por una cierta longitud y latitud, cuyos valores fueron establecidos de forma que representaran el centro geográfico del sector cuyo código postal corresponde al indicado en la tabla 11. De esta manera, si alguna aplicación fuera capaz de obtener el código postal a partir de la ubicación del dispositivo, dicho código postal sería el más probable en aparecer, pudiendo identificarlo más fácilmente en el análisis del tráfico de red.

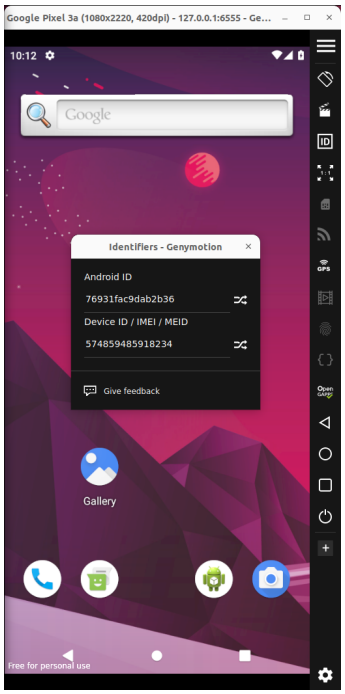



Figura 14: Modificación de ID de Android e IMEI. Fuente: Elaboración Propia.

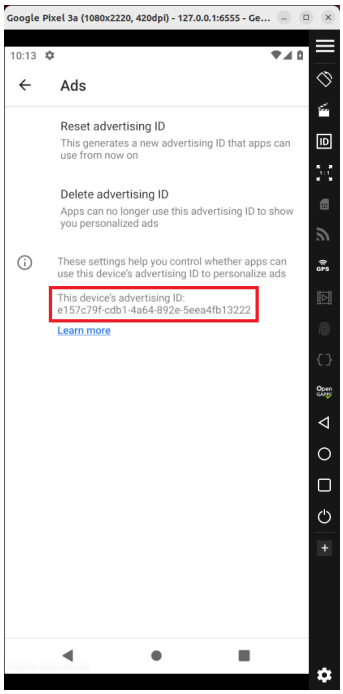



Figura 15: Obtención de ID de Publicidad. Fuente: Elaboración Propia.

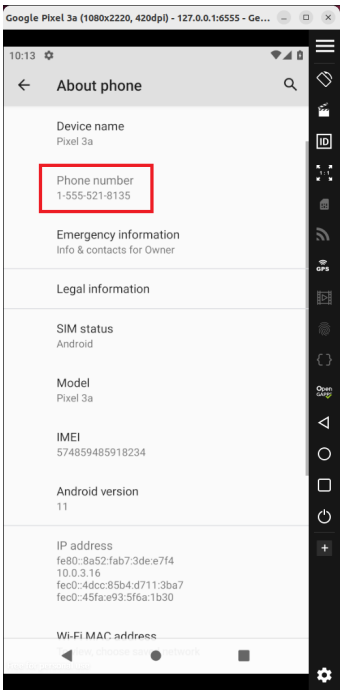



Figura 16: Obtención de Número de Teléfono. Fuente: Elaboración Propia.

## 4.3. Ejecución de Aplicaciones y Obtención de Tráfico de Red

Habiendo preparado completamente el dispositivo y sus respectivos PII para el análisis, se puede proceder con la descarga y prueba de las aplicaciones. Naturalmente, el primer paso corresponde en realizar la descarga de la aplicación que se pretende analizar a través de la Play Store. Una vez instalada en el dispositivo, MobSF la reconoce, permitiendo orquestar la aplicación. En esta etapa, se inhabilitan las funciones de *root detection* y *certificate pinning* de la aplicación en caso de tenerlas, además de habilitar la captura del tráfico de red. Al final de la instrumentación, la aplicación comienza a ejecutarse de manera automática y se da inicio a dicha captura. Como se mencionó en la sección anterior, este proceso se realiza dos veces por cada aplicación. La primera ejecución corresponde al proceso de creación de cuenta y se extenderá por el tiempo que se requiera para completarlo, mientras que la segunda ejecución se extenderá por un intervalo de 3 a 5 minutos, permitiendo realizar los flujos principales de la aplicación y emulando un uso normal de la misma.

Sin importar el tipo de ejecución realizada, el proceso de captura del tráfico de red continúa hasta que se seleccione la opción **Generate Report**, como se muestra en la interfaz de la figura 17. En ese momento, la aplicación se detiene automáticamente y MobSF construye un informe que compila toda la información obtenida a través del análisis dinámico, incluyendo el tráfico HTTP/HTTPS, llamadas al sistema y otras métricas relevantes. El informe generado incluye el tráfico de red condensado en un archivo de texto (`.txt`) que contiene la información respectiva de cada paquete enviado y recibido, donde su información se estructura de la manera presentada a continuación y ejemplificada en la figura 18.

- **Tipo de Paquete.** Naturalmente si el paquete figura como REQUEST corresponderá a un paquete enviado. El paquete que aparecerá inmediatamente después figurará como RESPONSE y corresponderá al paquete recibido como respuesta.
- **Tipo de Solicitud.** Indica el método HTTP y el protocolo utilizado junto a la dirección a la que se envía dicha solicitud.
- **Headers.** Metadata esencial para la transmisión y correcto enrutamiento de datos.
- **Payload.** Conjunto de datos transmitidos por el paquete respectivo. Es aquí donde generalmente se encontrará la información personal filtrada en caso de existir.

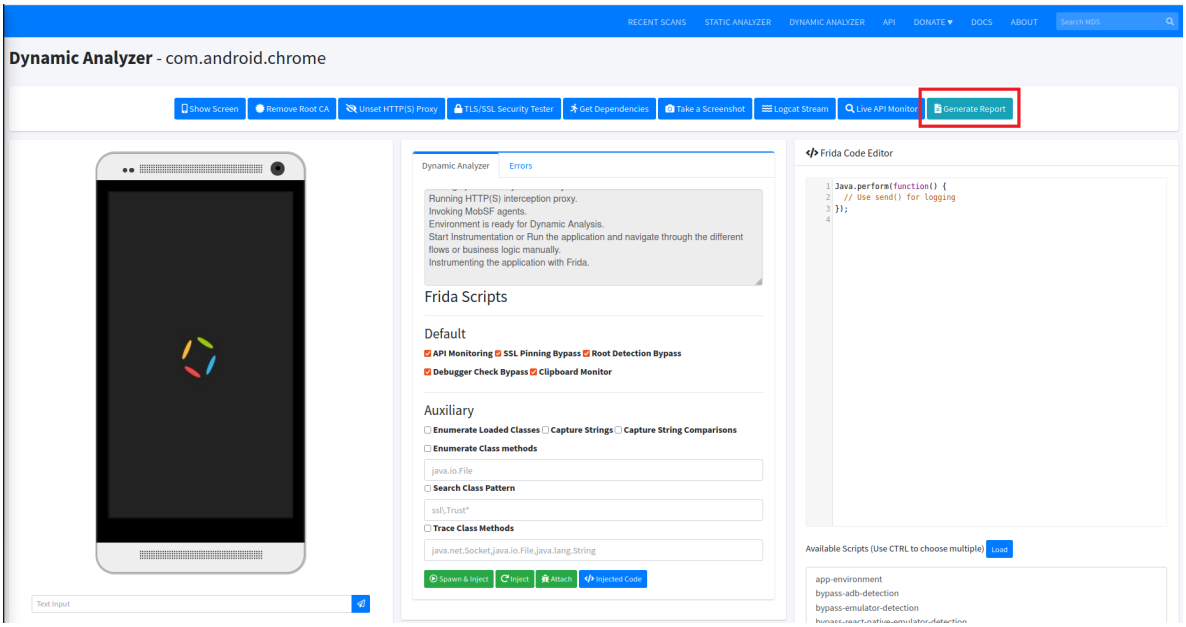



Figura 17: Interfaz de Análisis Dinámico de MobSF (Generate Report). Fuente: Elaboración Propia.

```
========
REQUEST
========
POST https://graph.facebook.com/v16.0/1102889023621608/activities HTTP/1.1
User-Agent: FBAndroidSDK.17.0.0
Accept-Language: en_US
Content-Type: application/x-www-form-urlencoded
Content-Encoding: gzip
Transfer-Encoding: chunked
Host: graph.facebook.com
Connection: Keep-Alive
Accept-Encoding: gzip

b'format=json&sdk=android&custom_events=%5B%7B%22_eventName%22%3A%22fb_mobile_deactivate_app%22%2C%22_eventName_md5%22%3A%2292255b491a4e25b5d809edcf3665affe%2
abe2-
b19776efc25c%22%2C%22fb_mobile_time_between_sessions%22%3A%22session_quanta_19%22%2C%22fb_mobile_launch_source%22%3A%22Unclassified%22%2C%22fb_mobile_app_inte
abe2-
b19776efc25c%22%2C%22fb_mobile_launch_source%22%3A%22Unclassified%22%2C%22fb_mobile_pckg_fp%22%3A%22f73dac0fd7c5cf466690b79efd0a540f%22%2C%22fb_mobile_app_cer
a1b7-06d7571a4c81&application_tracking_enabled=true&advertiser_id_collection_enabled=true&advertiser_id=3017b12d-751a-47e6-a060-
fef59f0b84ad&advertiser_tracking_enabled=true&installer_package=com.android.vending&extinfo=%5B%22a2%22%2C%22com.peqconsultores.lipigas%22%2C1112%2C%222.17.0%
=========
RESPONSE
=========
HTTP/1.1 200 OK
Content-Type: text/javascript; charset=UTF-8
Vary: Origin
Access-Control-Allow-Origin: *
facebook-api-version: v16.0
Strict-Transport-Security: max-age=15552000; preload
Pragma: no-cache
Cache-Control: private, no-cache, no-store, must-revalidate
Expires: Sat, 01 Jan 2000 00:00:00 GMT
x-fb-request-id: AHvWFQ0yCw-79m4s3WptOS_
x-fb-trace-id: F0JDjcWYoOj
x-fb-rev: 1012708192
X-FB-Debug: gPrfurjfsTRUavybqEL///MZ3XtlmDntIPWfF0bvtcrsv84auPuROASReY/OterRxUKSKgMsdZ4oVN2Ew9AuZg==
Date: Thu, 11 Apr 2024 17:18:23 GMT
X-FB-Connection-Quality: EXCELLENT; q=0.9, rtt=23, rtx=0, c=10, mss=1380, tbw=7032, tp=-1, tpl=-1, uplat=330, ullat=5
Alt-Svc: h3=":443"; ma=86400
Connection: keep-alive
Content-Length: 16
```

Figura 18: Ejemplo de Par de Paquetes REQUEST y RESPONSE. Fuente: Elaboración Propia.

Este proceso se repite para las 96 aplicaciones seleccionadas. Sin embargo, sólo 43 de ellas incluían un proceso de creación de cuenta, causando que las 53 aplicaciones restantes se ejecutaran una sola vez, y por lo tanto, se generaran un total de 139 archivos de tráfico de red.

## 4.4. Algoritmo de String Matching

Habiendo ejecutado las aplicaciones seleccionadas y recolectado su tráfico de red respectivo, se procede a utilizar el algoritmo de *string matching* elaborado para detectar aquellos paquetes que contengan información sensible.

Al emplear la distancia de Levenshtein en el algoritmo, se optimiza la detección de *strings* que contengan ligeras variaciones. En el análisis de tráfico de red, es frecuente encontrar que los PII presenten pequeñas modificaciones, principalmente inserciones y sustituciones de caracteres. Por lo general, es menos común que los PII sufran eliminaciones de caracteres, ya que esto implicaría una transmisión incompleta de la información. Por esta razón, se asigna un ”peso” mayor a esta operación en particular, lo que reduce la probabilidad de que estas coincidencias surjan. Por el contrario, las coincidencias que no incluyen eliminaciones de caracteres tienen mayores probabilidades de ser detectadas. Por ejemplo, en el caso de utilizarse *percent encoding*, el algoritmo es capaz de detectar que los siguientes *strings* son prácticamente idénticos, puesto que la distancia de Levenshtein entre ellos tendría un valor de 3, considerando una sustitución (”@” por ”%”) y dos inserciones (”4” y ”0”). Mientras mayor sea el valor total, menores las probabilidades de considerar el hallazgo como una coincidencia.

memoriatest77@gmail.com

↓

memoriatest77%40gmail.com

Cabe destacar que para algunos de los PII definidos en la sección 4.1, se consideraron múltiples variaciones en sus valores con el objetivo de adecuarse a una gama más amplia de convenciones. Específicamente, las fechas de cualquier tipo suelen tener distintas representaciones, por lo que se utilizaron las siguientes variaciones indicadas en la tabla 15.

| **Formato** | **Variación** |
|---|---|
| **Estadounidense** (MM/DD/YYYY) | 05/04/1977 |
| **Global** (DD/MM/YYYY) | 04/05/1977 |
| **ISO** (YYYY-MM-DD) | 1977-05-04 |

Tabla 15: Variaciones de PII (Fecha de Nacimiento). Fuente: Elaboración Propia.

Tomando en cuenta el funcionamiento del algoritmo, éste es ejecutado para todo el tráfico de red recolectado, elaborando un reporte con las posibles fugas de información detectadas para cada una de 139 archivos de tráfico de red recolectados. Para cada archivo, el algoritmo elabora un resumen de las posibles fugas, categorizándola según su PII, la URL a la que se hizo la solicitud y la cantidad de apariciones. Este formato se detalla a continuación en la figura 19.

```
 1 ========= All Leaked Data =========
 2 [Advertiser ID]
 3 1 graph.facebook.com
 4 1 api.papajohns.cl
 5 4 app-measurement.com
 6 [Android ID]
 7 1 codepush.appcenter.ms
 8 1 api.papajohns.cl
 9 1 o4506615877795840.ingest.us.sentry.io
10 [Mail]
11 70 api.papajohns.cl
12 1 mobile.slgnt.eu
13 [City]
14 3 app-measurement.com
15 [Latitude]
16 1 api.papajohns.cl
17 [Longitude]
18 1 api.papajohns.cl
19 [Region]
20 1 api.papajohns.cl
21 [Country]
22 1 api.papajohns.cl
```

Figura 19: Salida del Algoritmo (App: Papa John's Chile). Fuente: Elaboración Propia.

## 4.5. Elaboración de Reporte y Clasificación de Eventos

El último paso para elaborar el informe consiste en juntar los 139 resúmenes generados por el algoritmo de *string mathing* y resumirlos en una plantilla donde cada columna corresponda a los atributos indicados en la sección 3.6.2. Naturalmente, existe información que no era posible obtener a partir del algoritmo, por lo que fue completada de manera manual. Dichos atributos corresponden a los que se indican a continuación.

- **Tipo de Servidor.** La categorización de los servidores involucró una investigación de la naturaleza de los *endpoints*[20] para determinar si éste correspondía a un servidor propio, o bien, a un servidor de terceros.
- **Categoría del Servidor.** De manera similar al primer atributo, se realizó una investigación aún más exhaustiva para determinar de manera más específica la categoría que mejor se ajustara a cada URL. En caso de tratarse de un servidor de terceros, se determina si éste correspondía a servidores de publicidad, analítica, utilidades u otro.
- **Categoría del PII.** Categoría asignada según lo indicado en la tabla 6.
- **Encriptación en Tránsito.** Información recuperada de la página de la aplicación en Google Play Store. Ejemplificado en la figura 20. Cabe destacar que, existieron casos en los que la Play Store indicaba que no existía encriptación en tránsito, sin embargo, revisando el tráfico generado sí se evidenciaba tráfico HTTPS. En estos escenarios, se supuso que la causa de la discrepancia era el uso de una versión desactualizada del protocolo

[20] En contextos de desarrollo web y API's, designa la URL específica donde los servicios están disponibles para ser accedidos por clientes o aplicaciones.

TLS/SSL, lo que explicaría las advertencias de la Play Store. Ante la incertidumbre, se optó por considerar la información proporcionada por la Play Store.

- **Información Recolectada/Compartida.** Información recuperada de la página de la aplicación en Google Play Store. En caso de tratarse de una conexión con un servidor propio, se considera el dato como ”recopilado” y en caso contrario, como ”compartido”. Ejemplificado en las figuras 21 y 22.

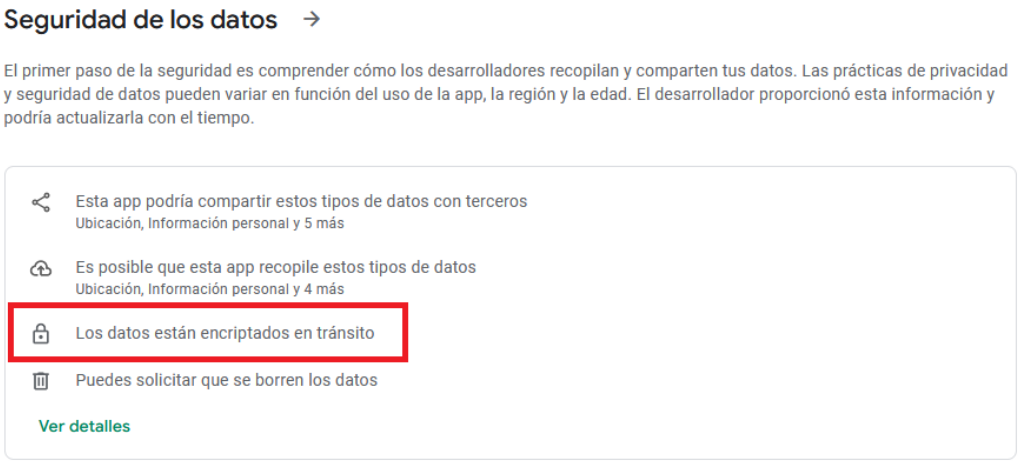



Figura 20: Encriptación en Tránsito (App: Papa John’s Chile). Fuente: Elaboración Propia.

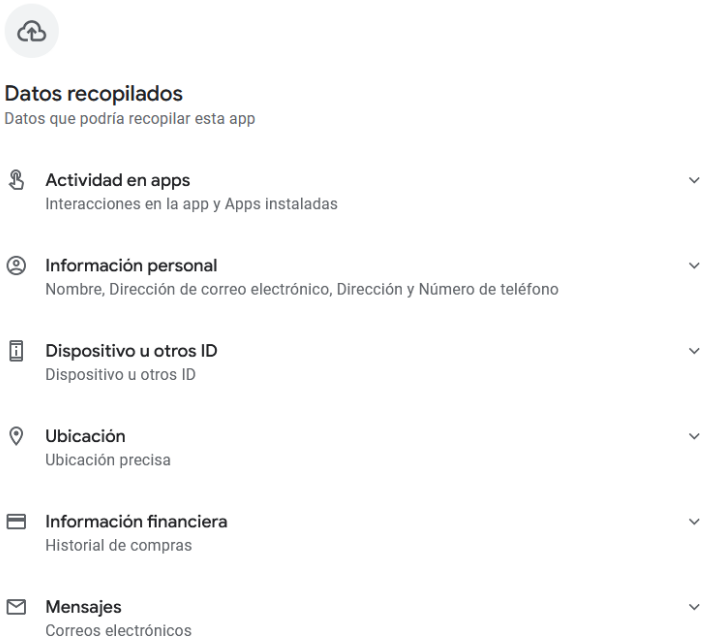



Figura 21: Información Recolectada (App: Papa John’s Chile). Fuente: Elaboración Propia.

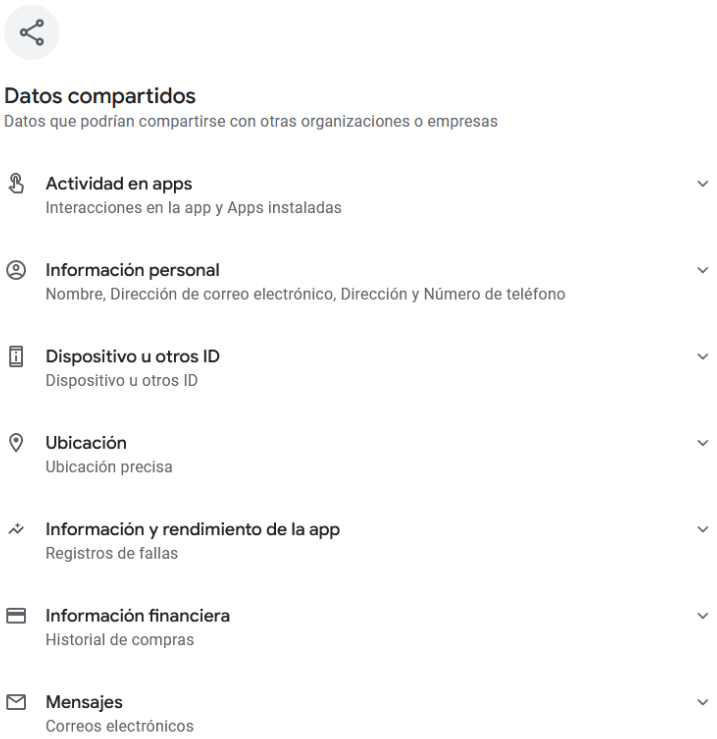



Figura 22: Información Compartida (App: Papa John's Chile). Fuente: Elaboración Propia.

Con el informe completo, se puede proceder a calcular el riesgo total de cada evento. Por ejemplo, si en uno de los hallazgos se identificó una fuga en una aplicación que cuenta con **Encriptación en Tránsito**, donde el PII filtrado corresponde a la categoría **Ubicación Precisa**, el servidor de destino corresponde a la categoría de **Analítica / Publicidad** y además, la página de la aplicación no especifica el PII en cuestión dentro de la **Información Recolectada/Compartida**, el valor se calcularía de la siguiente forma.

$$
\begin{aligned}
R_T &= W_E \cdot (1 - R_E) + W_P \cdot R_P + W_S \cdot R_S \cdot R_E + W_{RC} \cdot R_{RC} \\
R_T &= 0{,}5 \cdot (1 - 1{,}0) + 0{,}35 \cdot 0{,}75 + 0{,}5 \cdot 0{,}7 \cdot 1{,}0 + 0{,}15 \cdot 1{,}0 \\
R_T &= 0{,}2625 + 0{,}35 + 0{,}15 \\
R_T &= 0{,}7625
\end{aligned}
$$

De esta manera, el valor total del riesgo en este escenario sería de 0.7625, lo que en el estudio se definirá como un 76.25 % de probabilidad que el evento suponga una fuga, siendo categorizado como un evento con una probabilidad "alta" de riesgo según la tabla 9. Dicho cálculo se repite para cada uno de los hallazgos resumidos en el informe. Una vez finalizado este proceso, el informe se considera consolidado y se puede proceder a analizar los datos obtenidos.

# CAPÍTULO 5
# ANÁLISIS DE RESULTADOS

En la presente sección se expondrán los diversos resultados y estadísticas que se obtuvieron de la información recabada en el reporte final, considerando el nivel de riesgo calculado para cada uno de los hallazgos. Este análisis busca entender el comportamiento de las aplicaciones cuando éstas filtran información, así como determinar y comprender el nivel de privacidad que ofrecen en la actualidad. Además, con el objetivo de facilitar la comprensión, se aclara que a lo largo de la presente sección, se utilizarán las palabras ”evento” y ”hallazgo” de manera indistinta.

## 5.1. Análisis General del Tráfico Saliente

Habiendo finalizado con el análisis del tráfico de red correspondiente a las 96 aplicaciones abordadas, es importante cuantificar la cantidad de tráfico potencialmente malicioso que fue identificado. Por consiguiente, se busca calcular para cada aplicación, la proporción entre la cantidad de paquetes salientes que contenían alguno de los PII indicados en la sección 4.1, y la cantidad de paquetes salientes totales. En esta línea, se elaboró la tabla 16, donde se presentan las 15 aplicaciones con la mayor proporción de tráfico potencialmente malicioso.

| Top | Aplicación | Tráfico Saliente con PII |
|---|---|---|
| 1 | Kismia - Meet Singles Nearby | 75.08 % |
| 2 | Dating with Singles - iHappy | 16.94 % |
| 3 | PedidosYa | 16.21 % |
| 4 | Duolingo | 15.81 % |
| 5 | SOSAFE - City Social Network | 13.49 % |
| 6 | LinkedIn: Jobs & Business News | 11.95 % |
| 7 | Club Dominó | 11.04 % |
| 8 | Be Closer: Family Location | 10.47 % |
| 9 | Dating and Chat - Sweetmeet | 9.08 % |
| 10 | Pregnancy Tracker: Amma | 8.96 % |
| 11 | Bible App Lite - NIV Offline | 8.93 % |
| 12 | Spotify: Music and Podcasts | 8.93 % |
| 13 | Viajes Falabella | 8.85 % |
| 14 | Pluto TV: Watch TV & Movies | 7.74 % |
| 15 | Planner 5D: Home Design, Decor | 7.34 % |

Tabla 16: Top 15 Aplicaciones con Mayor Proporción de Tráfico Saliente con PII. Fuente: Elaboración Propia.

De la tabla, se desprende una serie de información interesante. En primer lugar, destaca la aplicación que encabeza la lista, es decir, *Kismia*, que indica que de todos los paquetes que ésta envió, un 75.08 % contenían PII. Esto adquiere más relevancia cuando se observa que el valor que lo sigue es del 16.94 % (*iHappy*), para luego seguir una tendencia a la baja sin tanta diferencia entre los demás valores. Además, la presencia de las aplicaciones en los puestos 1, 2 y 10, podría estar justificada, puesto que corresponden aplicaciones de citas, es decir, que generalmente operan con la información del perfil del usuario, con el objetivo de relacionarlo con personas de características similares. El mismo análisis puede ser realizado para las demás aplicaciones pertenecientes a la lista. No obstante, destacan aplicaciones como *Duolingo*, *Bible App Lite*, *Pluto TV* y *Planner 5D* que, dada su naturaleza, no justifican un porcentaje tan alto de tráfico con PII, pudiendo sugerir un comportamiento potencialmente malicioso.

Con el objetivo de aclarar cómo se distribuyen los porcentajes de manera general, se elabora el diagrama de la figura 23, indicando que la mayoría de aplicaciones envían PII en un 1 % - 5 % del total de paquetes salientes aproximadamente. Como acotación, se excluyó el porcentaje de *Kismia* para poder visualizar mejor el diagrama, pero aún así debe considerarse un *outlier* del 75.08 %.

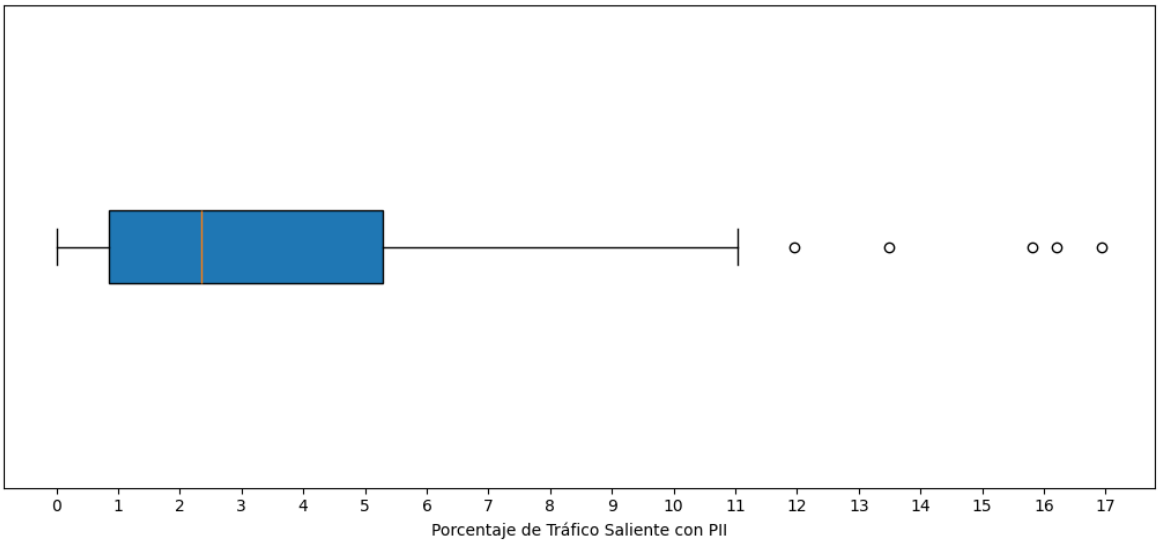



Figura 23: Distribución de la Proporción de Tráfico Saliente con PII. Fuente: Elaboración Propia.

Ahora bien, estos resultados pueden sugerir comportamientos anómalos pero no necesariamente representarán a las aplicaciones más ”peligrosas”, puesto que sólo indican que enviaron una mayor cantidad de PII a través de la red en relación al tráfico saliente total. Con la intención de determinar la naturaleza de los eventos en los que sí se identificaron PII, en la siguiente sección se estudiará el nivel de riesgo que representan estos eventos. Esto permitirá diferenciar el tráfico con una mayor probabilidad de filtrar información del tráfico con una mayor probabilidad de considerarse como benigno.

## 5.2. Análisis por Nivel de Riesgo

Considerando el tráfico recuperado de todas las aplicaciones, se identificaron un total de 6395 hallazgos. Ahora bien, se observaron múltiples ocasiones en el que un mismo PII era enviado a un mismo destino y por la misma aplicación, naturalmente resultando en eventos repetidos. Por lo tanto, para el análisis, cada evento estará definido por la combinación **Aplicación - PII - Destino**. Los eventos que difieran en cualquiera de estos atributos se considerarán como eventos distintos unos de otros y, análogamente, si los tres atributos son equivalentes, los eventos se considerarán idénticos y se agruparán. De esta manera, al realizar la agrupación respectiva, se identificaron un total de 963 eventos únicos, es decir, cerca de un 15 % en relación a los hallazgos totales. A su vez, dichos hallazgos fueron categorizados según los niveles de riesgo indicados en la tabla 9, resumiéndose de mejor manera en la figura 24 y en la tabla 17. Cabe destacar que los siguientes análisis se realizarán en base a los 963 hallazgos únicos y no al total.

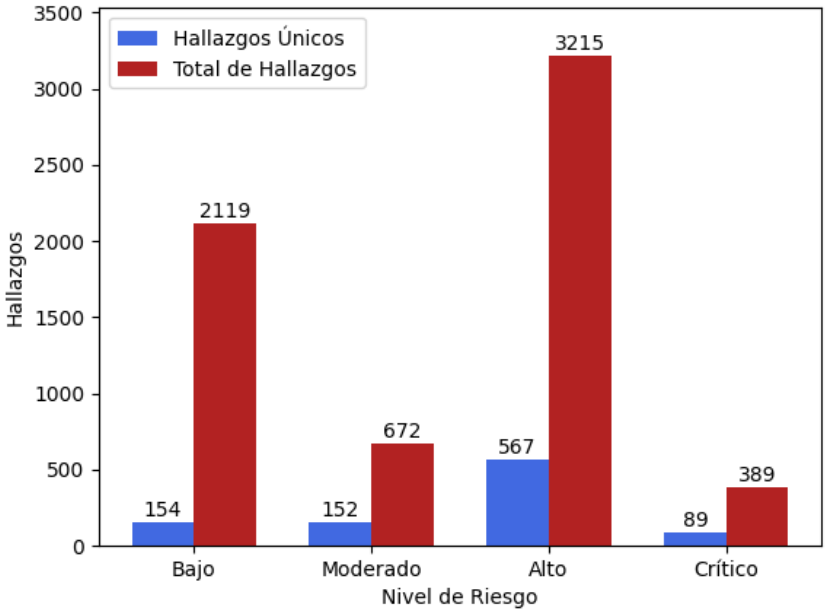



Figura 24: Hallazgos Identificados

| Nivel de Riesgo | Hallazgos Únicos | Total de Hallazgos | Proporción |
|---|---|---|---|
| Bajo (0 % - 25 %) | 154 | 2119 | 0.07 |
| Moderado (25 % - 50 %) | 152 | 672 | 0.23 |
| Alto (50 % - 75 %) | 567 | 3215 | 0.18 |
| Crítico (75 % - 100 %) | 89 | 389 | 0.23 |
| **Total** | 962 | 6395 | 0.15 |

Tabla 17: Hallazgos Identificados. Fuente: Elaboración Propia.

Como bien se puede apreciar, la mayor cantidad de eventos fueron categorizados con un nivel de riesgo ”alto”, seguido de ”bajo”, ”moderado” y ”crítico”, en orden decreciente. Si se considera la proporción entre hallazgos únicos identificados y el total de hallazgos, el nivel de riesgo ”bajo” es el que destaca por sobre las demás categorías, puesto que sólo un 7 % de ellos corresponden a hallazgos únicos. Aquellos eventos con una baja probabilidad de representar una fuga, en este escenario se repiten un promedio de 15 veces aproximadamente, lo que podría indicar que se trata de eventos más convencionales y/o rutinarios. Por el contrario, aquellos eventos con una probabilidad moderada o mayor de representar una fuga de información tienden a repetirse con menos frecuencia, indicando eventos poco comunes y lo que podría sugerir un esfuerzo por pasar más desapercibidos.

En línea con lo anteriormente mencionado, las tablas 19, 20, 21 y 22 presentan las 10 aplicaciones con mayor cantidad de hallazgos por nivel de riesgo. Ahora bien, con el objetivo de facilitar la comprensión de las tablas mencionadas, se presenta un ejemplo en la tabla 18 de cómo son contabilizados los PII y los destinos totales de los hallazgos identificados.

| **PII’s Filtrados** | **Destinos** |
|---|---|
| Nombre | `app-measurement.com` |
| Mail | `android.googleapis.com` |
| Nombre | `android.googleapis.com` |
| ID de Android | `adjust.com` |
| IMEI | `adjust.com` |

Tabla 18: Hallazgos de Aplicación de Ejemplo. Fuente: Elaboración Propia.

En este ejemplo, se tienen 5 hallazgos únicos, puesto que cada hallazgo cuenta con una combinación **Aplicación - PII - Destino** distinta. Dentro de los 5 hallazgos, se contabilizan 4 PII distintos (Nombre, Mail, ID de Android e IMEI) que a su vez, fueron filtrados a 3 destinos distintos (App-Measurement, GoogleApis y Adjust). La misma lógica fue aplicada para cada aplicación presente en las tablas que se muestran a continuación.

| **Top** | **Aplicación** | **Hallazgos** | **PII’s Filtrados** | **Destinos** |
|---|---|---|---|---|
| 1 | PedidosYa - Delivery Online | 12 | 5 | 4 |
| 2 | Microsoft Teams | 11 | 4 | 8 |
| 3 | Papa John’s Chile | 10 | 9 | 2 |
| 4 | Samsung Shop | 10 | 3 | 8 |
| 5 | LinkedIn: Jobs & Business News | 8 | 7 | 2 |
| 6 | Pluto TV: Watch TV & Movies | 7 | 5 | 3 |
| 7 | Spotify: Music and Podcasts | 7 | 3 | 4 |
| 8 | Wattpad - Read & Stories | 7 | 3 | 3 |
| 9 | Dating with Singles - iHappy | 5 | 5 | 1 |
| 10 | Duolingo | 5 | 3 | 2 |

Tabla 19: Top 10 Aplicaciones con más Hallazgos de Nivel ”Bajo”. Fuente: Elaboración Propia.

| Top | Aplicación | Hallazgos | PII's Filtrados | Destinos |
|---|---|---|---|---|
| 1 | Dating with Singles - iHappy | 9 | 4 | 7 |
| 2 | Skincare and Face Care Routine | 6 | 4 | 5 |
| 3 | SOSAFE - City Social Network | 6 | 4 | 4 |
| 4 | Battery Charging Animation | 6 | 3 | 5 |
| 5 | Remini - AI Photo Enhancer | 6 | 3 | 5 |
| 6 | OneFootball Resultados en Vivo | 6 | 2 | 6 |
| 7 | AttaPoll - Paid Surveys | 6 | 2 | 5 |
| 8 | Despegar: Vuelos y Hoteles | 6 | 2 | 5 |
| 9 | Pregnancy Tracker: Amma | 5 | 3 | 3 |
| 10 | Samsung Shop | 5 | 1 | 5 |

Tabla 20: Top 10 Aplicaciones con más Hallazgos de Nivel "Moderado". Fuente: Elaboración Propia.

| Top | Aplicación | Hallazgos | PII's Filtrados | Destinos |
|---|---|---|---|---|
| 1 | OneFootball Resultados en Vivo | 38 | 5 | 33 |
| 2 | Vix: TV, Deportes y Noticias | 19 | 7 | 15 |
| 3 | PedidosYa - Delivery Online | 19 | 7 | 10 |
| 4 | Radio Chile - FM, Online Radio | 18 | 5 | 15 |
| 5 | Pregnancy Tracker: Amma | 16 | 7 | 11 |
| 6 | Western Union Send Money | 16 | 7 | 10 |
| 7 | Moovit: Horario de Bus y Tren | 15 | 6 | 11 |
| 8 | Viajes Falabella | 14 | 5 | 10 |
| 9 | Dating with Singles - iHappy | 14 | 3 | 13 |
| 10 | Santiago Bus Checker | 12 | 6 | 6 |

Tabla 21: Top 10 Aplicaciones con más Hallazgos de Nivel "Alto". Fuente: Elaboración Propia.

| Top | Aplicación | Hallazgos | PII's Filtrados | Destinos |
|---|---|---|---|---|
| 1 | Club Dominó | 11 | 4 | 5 |
| 2 | PedidosYa - Delivery Online | 9 | 4 | 4 |
| 3 | AR Drawing Sketch Paint | 9 | 2 | 8 |
| 4 | Digital Compass for Android | 8 | 4 | 4 |
| 5 | Pelisplay Ver Peliculas HD | 8 | 2 | 7 |
| 6 | INFRACCIÓN DE MULTAS - CHILE | 7 | 5 | 6 |
| 7 | Language Translator: Translate | 6 | 4 | 4 |
| 8 | CUPI CHAT: Dating, Flirt, Meet | 5 | 2 | 5 |
| 9 | ComunidadFeliz | 4 | 4 | 2 |
| 10 | Mod Menu For RBX | 3 | 1 | 3 |

Tabla 22: Top 10 Aplicaciones con más Hallazgos de Nivel "Crítico". Fuente: Elaboración Propia.

Como se puede apreciar en la tabla 19, figuran aplicaciones ampliamente utilizadas por los usuarios como *Microsoft Teams*, *Samsung Shop*, *LinkedIn*, *Spotify* y *Duolingo*. Estas aplicaciones presentan un nivel de riesgo relativamente bajo en sus hallazgos, generalmente debido a que los PII son transmitidos a los servidores propios de las empresas responsables de dichas aplicaciones, las cuales gestionan la autenticación y el intercambio de datos sensibles de manera más controlada. Naturalmente, esto genera que la cantidad de destinos a los que se envían estos datos sea generalmente baja, incluso para *Microsoft Teams* y *Samsung Shop* que si bien ambos figuran con 8 destinos distintos, solamente se debe a la cantidad de subdominios con los que cuentan sus servidores.

Por otro lado, se hará especial énfasis en aquellas que se abordan las aplicaciones con más hallazgos de nivel alto y crítico, respectivamente. La tabla 21 es particularmente relevante debido al volumen de datos que representa. Como bien se observó en el gráfico de la figura 24, los eventos con un nivel de riesgo alto son los más numerosos, lo que sugiere que aplicaciones como *OneFootball*, *Vix* y *PedidosYa* presenten faltas significativas de privacidad. Considerando que un evento está dado por la combinación **Aplicación - PII - Destino**, *OneFootball* resalta por sobre los demás, pues la aplicación envía datos sensibles a 33 destinos distintos, siendo éste el valor más alto de todas las aplicaciones evaluadas.

Aunque el número de hallazgos puede indicar el nivel de privacidad de una aplicación, los eventos clasificados como críticos representan una amenaza considerablemente mayor. La detección de un solo evento en esta categoría es suficiente para catalogar una aplicación como altamente riesgosa. En este sentido, en la tabla 22 se pueden observar aplicaciones como *Club Dominó* y *PedidosYa*, ambas dentro de la categoría ”Comer y Beber”, lo que podría sugerir un patrón de comportamiento recurrente en este tipo de aplicaciones, sobre todo considerando que *PedidosYa* ya tiene una cantidad significativa de eventos con un nivel de riesgo alto. Además, resulta interesante encontrar aplicaciones como *AR Drawing Sketch Paint*, cuya principal funcionalidad es la de dibujar, y por otro lado, *Language Translator: Translate*, que es básicamente un traductor. Estas son aplicaciones que en un principio no deberían acceder a datos personales del usuario para funcionar adecuadamente, mucho menos transmitirlos. Ahora bien, mientras estas aplicaciones presentan fugas de información injustificadas, existen casos como el de *CUPI CHAT*, que al ser una aplicación de citas, al menos resulta más coherente que tenga acceso a datos referentes al perfil del usuario.

Como último punto y en contraste con lo anteriormente expuesto, se identificaron tres aplicaciones sin hallazgos. Estas incluyen *Sketchbook* de la categoría ”Arte y Diseño”, *Zello PTT Walkie Talkie* de ”Comunicación” y *Reproductor TV Chilena* de ”Aplicaciones de Video”. Es importante mencionar que esta situación tendrá incidencia en las próximas secciones del análisis, donde estas tres categorías mostrarán resultados basados en solamente dos aplicaciones cada una, en lugar de tres como las demás categorías.

A modo de resumen, el análisis entrega la siguiente información:

- Los eventos con un nivel de riesgo bajo suelen repetirse más que los moderados, altos y críticos, pudiendo indicar eventos más rutinarios.
- Los eventos correspondiente a aplicaciones sustentadas por grandes empresas (*Microsoft Teams*, *Samsung Shop*, *LinkedIn*, *Spotify*, *Duolingo*) suelen presentar niveles de riesgo más bajos, puesto que los PII suelen ser gestionados por servidores propios y no de terceros.
- *One Football*, *Vix* y *PedidosYa* encabezan la lista de aplicaciones con más eventos de nivel alto.
- *Club Dominó* y *Pedidos Ya* encabezan la lista de aplicaciones con más eventos críticos y ambas pertenecen a la categoría ”Comer y Beber”, lo que podría indicar un comportamiento propio de este tipo de aplicaciones.
- La naturaleza *AR Drawing Sketch Paint* y *Language Translator* vuelve sospechosa su aparición en el top de aplicaciones con más hallazgos críticos, puesto que ninguna debería requerir datos personales para funcionar correctamente.
- Las aplicaciones *Sketchbook*, *Zello PTT Walkie Talkie* y *Reproductor TV Chilena* no presentaron ningún hallazgo.

## 5.3. Análisis por Categorías

Con el objetivo de identificar patrones o características comunes entre las aplicaciones que presentan un mayor número de fugas de información, se analizaron los resultados agrupados en las 32 categorías definidas por la Play Store. La figura 25 que se muestra a continuación, presenta la cantidad de hallazgos identificados para cada una de estas categorías.

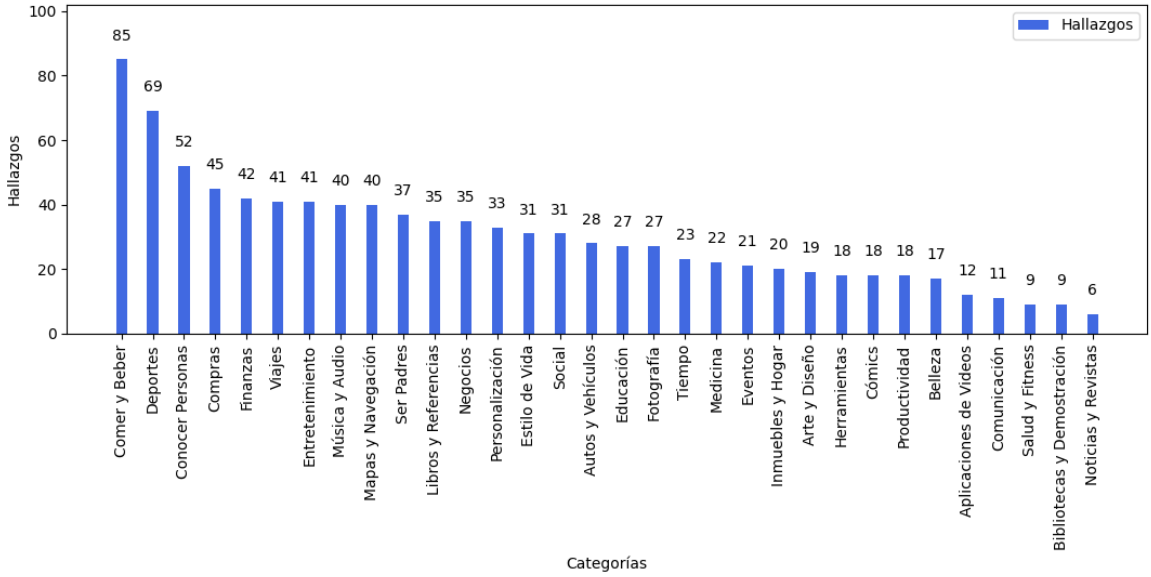



Figura 25: Categorías con más Hallazgos Identificados. Fuente: Elaboración Propia.

Como se puede observar en el gráfico y en línea con lo mencionado en la sección anterior, la categoría ”Comer y Beber” presenta el mayor número de hallazgos, con una proporción considerable de eventos de nivel alto y crítico. Como se puede observar en el anexo A, las aplicaciones analizadas de esta categoría en particular corresponden a *PedidosYa*, *Papa John’s Chile* y *Club Dominó*, todas relacionadas con el servicio de *delivery*. Estas aplicaciones suelen requerir datos como la ubicación y el número de teléfono para operar correctamente. Considerando que ambos datos fueron categorizados como de los más sensibles, no es sorprendente que los hallazgos estén asociados a un nivel alto de riesgo. El mismo análisis se puede hacer para categorías como ”Mapas y Navegación”, e incluso ”Viajes”, que comúnmente solicitan datos como la ubicación.

De la misma manera, categorías como ”Conocer Personas” que corresponden a aplicaciones de citas, exigen que los usuarios creen perfiles detallados para emparejarlos con personas de rangos etarios similares, en áreas cercanas y con intereses en común. Esto nuevamente genera que las aplicación tengan a disposición múltiples datos del usuario que luego son transmitidos a múltiples destinos sin un control demasiado riguroso. Por otro lado, al tope de la lista también aparecen las categorías ”Deportes”, ”Compras”, ”Entretenimiento” y ”Viajes”. Actualmente, éstas albergan una gran cantidad de aplicaciones cuyo modelo de negocio está basado en la publicidad y además se apoyan en servidores de analítica con

el principal objetivo de fomentar el uso de su aplicación. Esto, generalmente conlleva una intensa interacción con servidores que rastrean el comportamiento del usuario con el objetivo de ofrecer una experiencia más personalizada. Sin embargo, también implica que la información relacionada con la cuenta asociada a la aplicación sea compartida a un serie de servidores.

Para entregar un poco más de información con respecto al nivel de riesgo que tienen los hallazgos según la categoría de la aplicación, se elaboró el gráfico de la figura 26 que se presenta a continuación.

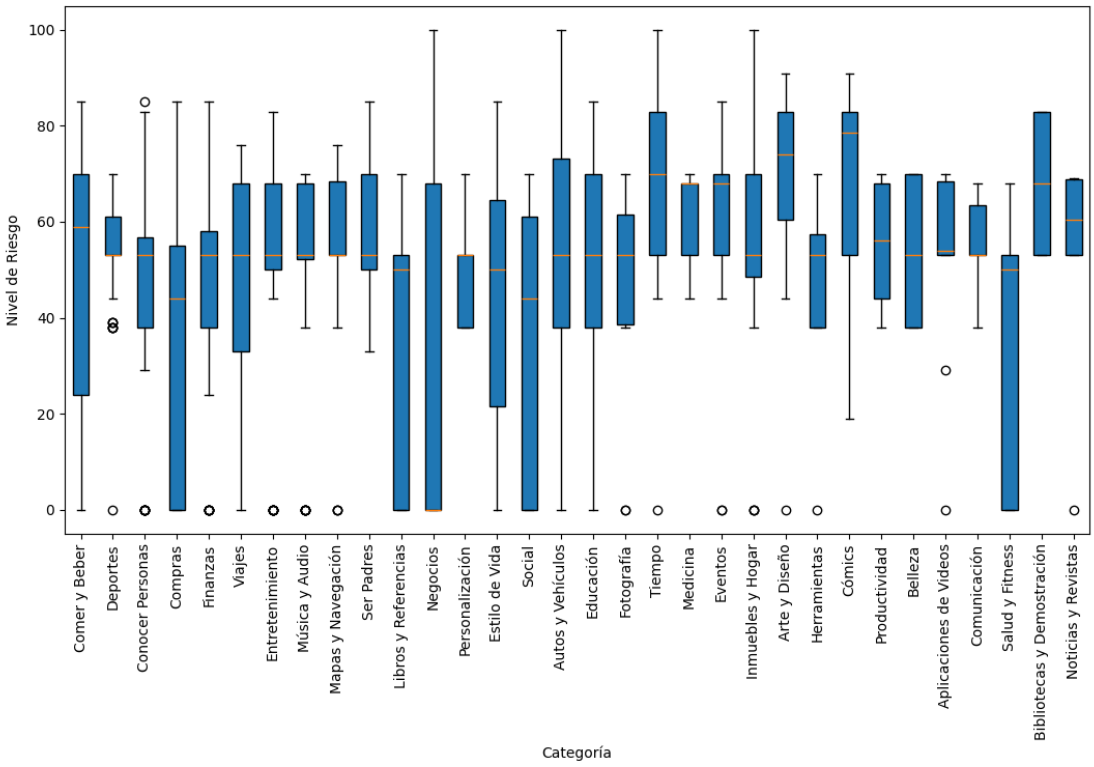



Figura 26: Distribución del Riesgo de los Hallazgos Identificados por Categoría. Fuente: Elaboración Propia.

El gráfico de la figura 26 reafirma lo que se ha expuesto a lo largo del análisis. La mayoría de los hallazgos fueron clasificados con un alto nivel de riesgo, es decir, entre un 50 % y un 75 %. El diagrama ilustra cómo los datos se agrupan alrededor de estos valores para la mayoría de las categorías, que están ordenadas por cantidad de hallazgos, siendo las de más a la izquierda aquellas más representativas por contener una mayor proporción de datos en comparación con las demás. Dicho esto, en el gráfico son evidentes aquellas categorías cuyos hallazgos tienden a tener niveles de riesgo más bajos, destacando ”Compras”, ”Libros y Referencias”, ”Negocios”, ”Social” y ”Salud y Fitness”. En contraste, ”Tiempo”, ”Arte y Diseño”, ”Cómics” y ”Bibliotecas y Demostración” parecen tener valores superiores. Sin embargo, dado que estas últimas presentan un menor número total de hallazgos, esto podría afectar la percepción de su peligrosidad en comparación con aquellas con más incidentes. En es-

te contexto, prestar atención a las categorías donde el límite superior es mayor puede ser indicativo de una mayor presencia de eventos de nivel crítico.

A modo de resumen, el análisis entrega la siguiente información:

- La categoría ”Comer y Beber” presenta la mayor cantidad de hallazgos, con una proporción considerable de eventos de nivel alto y crítico.
- Las categorías suelen indicar una mayor cantidad de hallazgos según los datos que requieren para operar correctamente, y por lo tanto, de su naturaleza.
- Las categorías ”Comer y Beber” (*Delivery)*, ”Mapas y Navegación” y ”Viajes” suelen solicitar y compartir la ubicación, lo que sugiere mayor cantidad de hallazgos, especialmente altos y críticos.
- La categoría de ”Conocer Personas” (Aplicaciones de Citas) suelen requerir acceso al perfil completo del usuario, resultando en mayor cantidad de hallazgos.
- Las categorías ”Deportes”, ”Compras”, ”Entretenimiento” y ”Viajes” suelen basar su modelo de negocio en publicidad y la oferta de productos/servicios personalizados, lo que generalmente se traduce en conexiones con servidores de analítica y publicidad, resultando en una mayor cantidad de hallazgos, especialmente altos y críticos.
- La mayoría de hallazgos tienen un nivel de riesgo alto. Esto significa, entre un 50 % y 75 % de probabilidad de representar una fuga de información.
- Las categorías ”Compras”, ”Libros y Referencias”, ”Negocios”, ”Social” y ”Salud y Fitness” suelen tener hallazgos con niveles de riesgo generalmente inferiores.
- Las categorías ”Tiempo”, ”Arte y Diseño”, ”Cómics” y ”Bibliotecas y Demostración” suelen tener hallazgos con niveles de riesgo generalmente superiores.

## 5.4. Análisis por Destinos

Como último análisis, se busca hacer énfasis en el tipo de servidores a los que se conectan las aplicaciones evaluadas, además de profundizar en los peligros intrínsecos de dichos servidores. En este contexto, se elaboró la figura 27 donde se muestra cómo se distribuyen los tipos de servidores según la cantidad de hallazgos identificados.

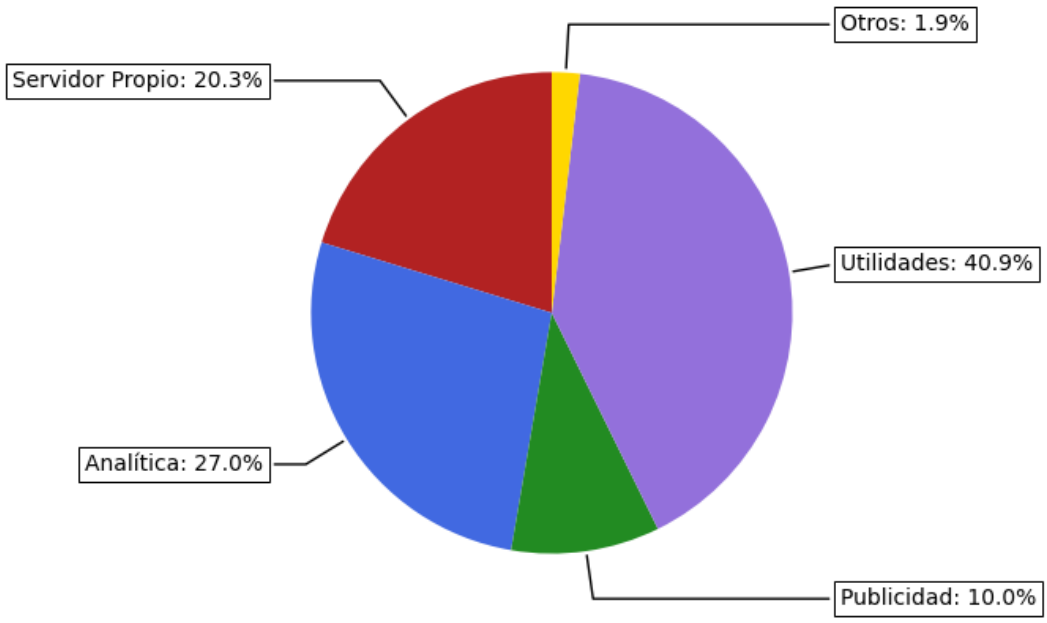



Figura 27: Hallazgos por Categoría del Destino. Fuente: Elaboración Propia.

El análisis del gráfico revela que las conexiones a servidores de analítica superan incluso a aquellas realizadas con servidores propios de las aplicaciones, lo que indica un importante intercambio de datos del usuario con entidades dedicadas a la recopilación de información para seguimiento. Esto resulta especialmente preocupante al considerar que los servidores de analítica, junto con los servidores de publicidad y aquellos cuya naturaleza no pudo ser identificada con certeza, abarcan casi un 40 % de los hallazgos identificados. Se esperaría que la mayoría de los eventos detectados se concentrasen en los servidores propios, los cuales deberían manejar los datos adecuadamente y sólo para fines esenciales, sin embargo, el gráfico evidencia lo contrario. Por otro lado, aquellos destinos que fueron catalogados como ”Utilidades”, comúnmente utilizados para tareas específicas como verificación de identidad e integración con servicios de terceros, también representan un porcentaje significativo, sobre todo considerando el riesgo de exposición que lleva consigo la dependencia de terceros. Ahora bien, la tabla 23 da un poco más de información respecto a estos últimos mencionados y su alcance en cuanto a las aplicaciones analizadas.

| Servidor de Destino | Categoría | Hallazgos | Aplicaciones |
|---|---|---|---|
| googleapis.com | Utilidades | 249 | 78 |
| app-measurement.com | Analítica | 73 | 63 |
| graph.facebook.com | Analítica | 46 | 43 |
| fundingchoicesmessages.google.com | Utilidades | 28 | 28 |
| adjust.com | Analítica | 25 | 19 |
| api.papajohns.cl | Servidor Propio | 10 | 1 |
| crashlyticsreports-pa.googleapis.com | Analítica | 9 | 4 |
| unityads.unity3d.com | Publicidad | 9 | 9 |
| wallet.google.com | Utilidades | 9 | 9 |
| ingest.sentry.io | Analítica | 8 | 5 |
| linkedin.com | Servidor Propio | 7 | 1 |
| adx.ads.vungle.com | Publicidad | 6 | 6 |
| api2.branch.io | Analítica | 6 | 4 |
| google-analytics.com | Analítica | 6 | 3 |
| api.amplitude.com | Analítica | 6 | 1 |

Tabla 23: Top 15 Servidores de Destino con más Hallazgos Identificados. Fuente: Elaboración Propia.

La tabla presentada clarifica la razón de por qué existe una cantidad tan elevada de hallazgos asociados a servidores de ”Utilidades”. Se observa que gran parte de estos pertenecen exclusivamente al dominio de `googleapis.com`, el cual incluye varios subdominios que ofrecen distintas funcionalidades. Cabe destacar que sólo fueron agrupados aquellos dominios debidamente catalogados como ”Utilidades”. Aquellos dominios que se ajustaban mejor a otra categoría, fueron excluidos de esta agrupación, como fue el caso de `crashlyticsreports-pa.googleapis.com`, que se especializa en analítica. Dicho esto, como puede observarse, de las 96 aplicaciones evaluadas, un total de 78 se conectan a alguno de estos subdominios y con justa razón, puesto que Android, al tener un ecosistema basado en Google, está ligado a la utilización de herramientas y funcionalidades con las que opera el sistema operativo del dispositivo.

Por otro lado, al analizar la tabla en su conjunto, es evidente la presencia de diversos servidores de analítica entre los 15 principales en términos de hallazgos identificados, especialmente entre los cinco primeros, sobre todo considerando que éstos son visitados por una proporción significativa de las aplicaciones evaluadas. Lo más alarmante es que, potencialmente, los tres primeros servidores de analítica podrían acumular casi toda la información sobre el usuario y su comportamiento a través de distintas aplicaciones, y aún más si se consideran todos los servidores que fueron identificados. En este contexto, el riesgo que esto puede significar en cuanto a la privacidad requiere de un análisis más exhaustivo y focalizado, por lo que se aborda más en detalle en la próxima sección.

### 5.4.1. Servidores de Analítica/Publicidad

Como se mencionó en secciones anteriores, tanto los servidores enfocados en publicidad como aquellos enfocados en analítica, se centran en entender el comportamiento del usuario con el objetivo de proporcionar información valiosa a las empresas y mejorar la efectividad de las campañas publicitarias, respectivamente. Funcionalmente, ambos tipos de servidores requieren de la integración de librerías (SDK[21]) para poder operar en aplicaciones de Android y recopilar la información necesaria para hacer el análisis respectivo. Ahora bien, la información que dichas librerías pueden recopilar y compartir siempre estará sujeta a la información que la aplicación tiene a disposición, considerando datos que el usuario ingresó directamente, o bien, datos obtenidos mediante algún permiso que fue previamente aceptado por el usuario. En este contexto, para profundizar en los PII que suelen filtrarse, se incluye el gráfico de la figura 28. Éste muestra los PII que se identificaron con mayor frecuencia durante las pruebas.

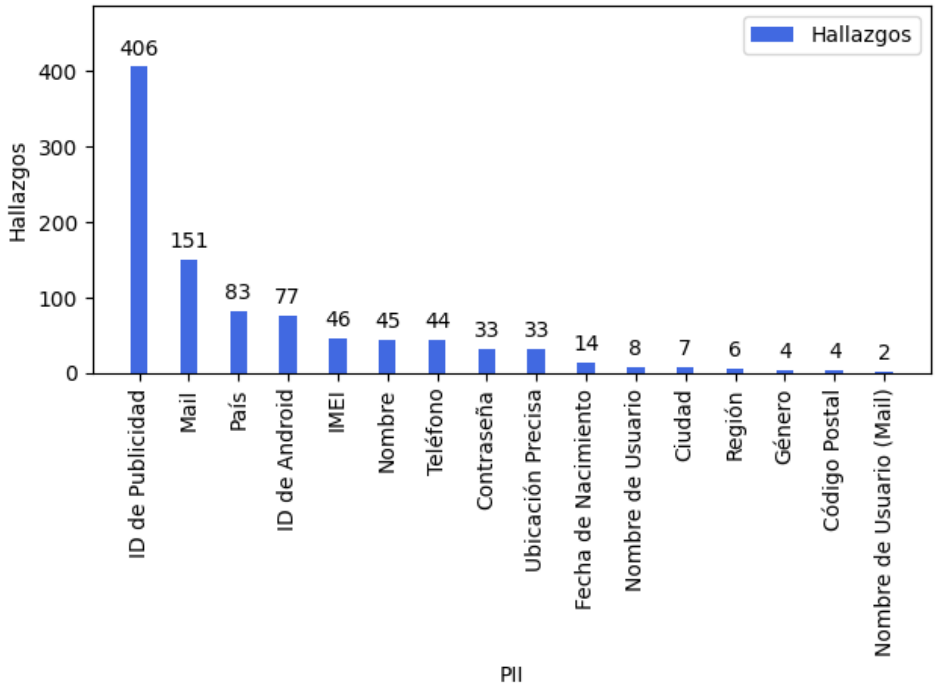



Figura 28: PII Identificados. Fuente: Elaboración Propia.

Como bien se puede apreciar en el gráfico, el PII que más apareció en el *payload* de los paquetes salientes corresponde al principal identificador que usan los servidores de publicidad y analítica para distinguir a los diferentes usuarios, el ID de Publicidad. Esto puede deberse a múltiples factores. Sin embargo, en este contexto, el motivo específico pasa a segundo plano al compararse con las implicaciones que esto tiene en la privacidad de los datos.

[21]SDK (Software Development Kit) es un conjunto de herramientas y recursos que permiten a los desarrolladores crear aplicaciones para una plataforma específica. Incluye bibliotecas, documentación y ejemplos para facilitar el desarrollo.

Si bien, los resultados obtenidos indican que se identificaron 406 eventos donde se filtra el ID de Publicidad, existen 70 casos donde dicho identificador se filtra en conjunto con otros PII. Éstos se muestran más en detalle en la figura 29.

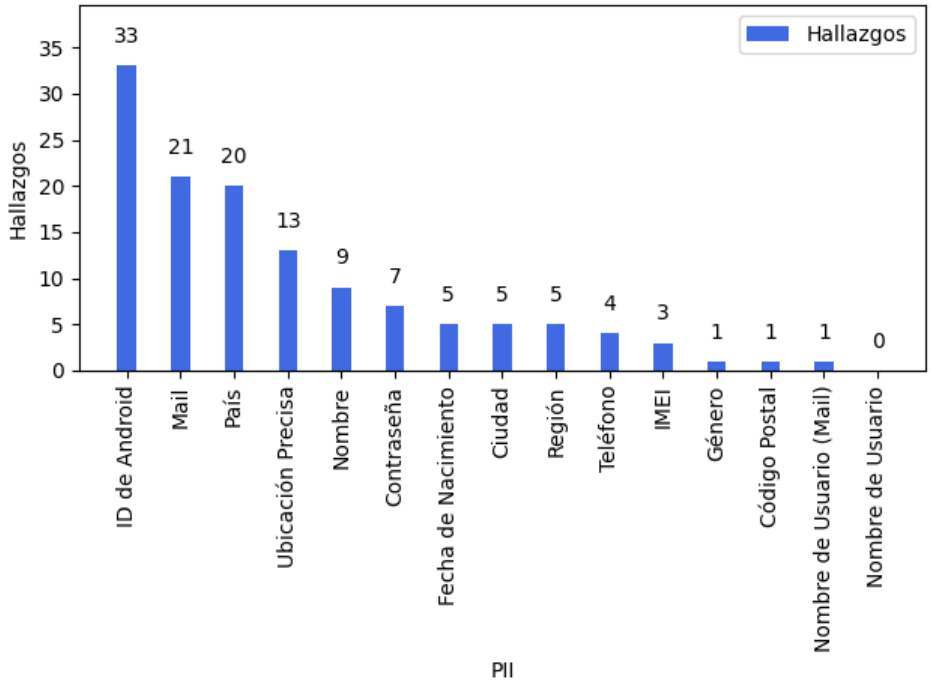



Figura 29: PII Filtrados en conjunto con el ID de Publicidad. Fuente: Elaboración Propia.

Como se puede apreciar en el gráfico de la figura 29, el ID de Android y el correo electrónico encabezan la lista de PII filtrados en conjunto con el ID de Publicidad, es decir, que fueron enviados a un mismo destino y por la misma aplicación. Esto sugiere que las preocupaciones vistas previamente sobre la asociación con otros datos más persistentes siguen siendo relevantes. En este caso, un servidor que recopile estos datos tendría acceso no sólo al ID de Publicidad, sino que también a otros datos personales, siendo sencillo asociar un nuevo ID de Publicidad asumiendo que los otros datos no se ven modificados. Por tanto, este análisis no se limitaría solo al ID de Android y al correo electrónico, sino que también aplicaría para otros datos presentes en el gráfico, como la ubicación, el nombre, el teléfono, el IMEI y el nombre de usuario. Aunque la asociación entre estos y el ID de Publicidad puede variar en complejidad, lo cierto es que el ID de Publicidad debería transmitirse de manera aislada y no junto con otros PII, como sucedió en un porcentaje considerable de los hallazgos.

A modo de resumen, el análisis entrega la siguiente información:

- Casi un 40 % de los hallazgos identificados, corresponden a conexiones con servidores de publicidad, analítica o con servidores cuya naturaleza no pudo ser determinada con certeza, lo que se traduce en hallazgos de niveles superiores de riesgo.
- El servidor `googleapis.com` encabeza la lista de servidores de destino con más hallazgos identificados. Un total de 78 aplicaciones distintas se conectan a alguno de sus subdominios, puesto que Android, al estar basado en Google, está ligado a la utilización de herramientas proveídas por este servidor en particular.
- Dentro de los 15 servidores de destino con más hallazgos identificados, los servidores de analítica son los que tienen una mayor presencia.
- El ID de Publicidad corresponde al PII más presente en los hallazgos identificados, seguido del Correo Electrónico, el País y el ID de Android.
- El ID de Android y el Correo Electrónico corresponden a los PII que más se filtran en conjunto con el ID de Publicidad, lo que indica que incluso cuando se este último es restablecido para evitar el seguimiento a largo plazo, aún puede ser asociado a identificadores más persistentes e invalidando la opción que se les da a los usuarios para cuidar su privacidad.

# CAPÍTULO 6
# CONCLUSIONES

Una vez finalizado el análisis de los resultados obtenidos, se procede a verificar si los objetivos planteados al principio del estudio fueron cumplidos en su totalidad, además de identificar áreas de mejora y proponer nuevos acercamientos para trabajos futuros.

## 6.1. Impacto y Limitaciones de la Solución Propuesta

El presente estudio tuvo como foco la propuesta e implementación de una metodología para la identificación de fugas de información a través de una gama de aplicaciones. Dicha propuesta busca responder a la necesidad de ahondar en las prácticas de privacidad que siguen las aplicaciones de Android en la actualidad y entender su comportamiento y contexto al momento de manejar la información sensible de los usuarios. En este contexto, la solución propuesta se establece como un primer paso hacia el desarrollo de nuevas metodologías basadas en análisis dinámico, ofreciendo además una mayor visibilidad y entendimiento de aplicaciones y servidores con comportamientos potencialmente peligrosos.

En cuanto a las limitaciones, fueron aquellas restricciones identificadas durante el desarrollo del estudio las que finalmente permitieron adaptar y estructurar la solución que fue implementada. Las principales limitaciones se resumen a continuación.

- Una de las más grandes restricciones fue identificada al momento de estudiar las técnicas de análisis dinámico creadas hasta la fecha, en conjunto con la evolución del sistema operativo de Android durante los años. Todas las técnicas basadas en *Behaviour-based Monitoring* que fueron analizadas, funcionaban simulando un ataque MITM para recuperar y analizar el tráfico de red saliente. Esto, si bien era posible en versiones más antiguas de Android, no lo es en aquellas más modernas a la versión 10.0 y 11.0, donde se introducen nuevas prácticas de seguridad que impiden recuperar el tráfico, y por lo tanto, imposibilitan cualquier tipo de análisis en tiempo real en dispositivos no rooteados.
- Las limitaciones mencionadas anteriormente hicieron necesario el uso de un entorno externo para la metodología propuesta. Esto llevó a la implementación de un dispositivo virtual, que trae consigo restricciones propias de este tipo de emulación. Por ejemplo, aplicaciones que requieren una conexión Bluetooth con otro dispositivo o aquellas que dependen de sensores específicos de dispositivos reales, tuvieron que ser debidamente excluidas de las pruebas.
- Por último, existieron casos en los que la instrumentación realizada por MobSF no pudo realizarse con éxito. Ésta corresponde a otra de las limitaciones del estudio y la ra-

zón principal por la que algunas aplicaciones llamativas no fueron capaces de ser analizadas. Esto es atribuido principalmente a las restricciones propias de la herramienta y a la adopción de nuevas prácticas por parte de las aplicaciones, lo que complicó su orquestación y, por ende, su análisis.

## 6.2. Objetivos

### 6.2.1. Objetivo General

El objetivo general que se buscaba alcanzar con el desarrollo de la presente memoria, era el de proponer una metodología que a través de un análisis dinámico de aplicaciones fuera capaz de contribuir a la detección de fugas de información y a la protección de la privacidad de los usuarios. A través del presente estudio, el objetivo se considera cumplido, proponiendo satisfactoriamente una metodología capaz de analizar aplicaciones y categorizar exitosamente aquellos hallazgos donde la información personal de los usuarios abandona el dispositivo y suponen eventos potencialmente peligrosos.

### 6.2.2. Objetivos Específicos

De la misma manera, a continuación se abordan nuevamente los cuatro objetivos específicos planteados al principio del estudio.

- **Resumir los aspectos y las etapas más relevantes del análisis dinámico y hacer énfasis en su funcionamiento y aplicaciones.**
  Este objetivo específico se considera como logrado al considerar el estudio que se hizo en los primeros capítulos. Tanto el marco conceptual y el estado del arte buscan ahondar en los aspectos más importantes del análisis dinámico y sus ventajas frente al análisis estático. En estas secciones, se exploró la clasificación de las técnicas de análisis dinámico, identificando cuáles son las más influyentes y cuáles ofrecen los mejores resultados. Además, se realizó un estudio exhaustivo de sus requerimientos, permitiendo identificar qué tan factible son sus implementaciones en entorno más modernos.

- **Seleccionar una técnica de análisis dinámico que se adecúe al problema y definir una metodología que facilite su implementación.**
  A partir del estudio realizado, fue posible explorar las posibilidades y la factibilidad de las técnicas de análisis dinámico que más se adecuaran al contexto actual. En esta línea, se optó por elaborar una propuesta que estuviera basada en una técnica de *Behaviour-based Monitoring*, específicamente aquellas focalizadas en el análisis del tráfico de red. De la misma manera, se seleccionaron herramientas como Genymotion y MobSF

para emular un entorno de Android y capturar el tráfico de red, respectivamente. Este par de herramientas son actualmente las alternativas más compatibles y fáciles de usar para el análisis dinámico, facilitando así su implementación y contribuyendo al cumplimiento exitoso del objetivo.

- **Detectar y categorizar las distintas vulneraciones y amenazas detectadas al momento de ejecutar una aplicación, según su comportamiento.**
  Este objetivo fue satisfactoriamente cumplido a través del algoritmo de *string matching* implementado y la categorización propuesta para los hallazgos identificados. El algoritmo es capaz de detectar correctamente si los paquetes salientes contenían datos sensibles pertenecientes al usuario, incluso si éstos presentaban pequeñas variaciones. Esto, en conjunto con la propuesta que se elaboró para categorizar aquellos eventos potencialmente peligrosos, permitió evaluar las amenazas que suponen las aplicaciones analizadas, lo que finalmente permitió identificar ciertos patrones en común entre aplicaciones que filtran información.

- **Documentar y analizar los resultados de las pruebas realizadas, para obtener conclusiones sobre la efectividad de la propuesta metodológica de análisis dinámico de aplicaciones móviles.**
  Finalmente, este último objetivo se cumplió exitosamente en la última sección del presente estudio, donde se resumen y categorizan todos los hallazgos identificados. En esta sección, se estudiaron los aspectos principales de los eventos en los que se filtra información y se lograron definir aquellas aplicaciones con mayor nivel de riesgo y los aspectos más relevantes al momento de categorizar un evento como peligroso. A su vez, estos resultaron revelaron el nivel de privacidad con el que cuentan las aplicaciones más utilizadas en la actualidad y lo lejos que éstas están de brindar un entorno seguro para los datos sensibles que son compartidos.

## 6.3. Resultados Obtenidos

Como primer punto, según los resultados obtenidos, las aplicaciones que podrían presentar fugas de información parecen ser relativamente fáciles de detectar, ya que el problema subyacente no corresponde a que las aplicaciones obtengan datos a espaldas del usuario, sino más bien, que éstas transmitan los datos que el mismo usuario puso a disposición de la aplicación, ya sea de manera directa o a través de un permiso. Los resultados sugieren que las aplicaciones no acceden a datos que el usuario no ha entregado, sino que se aprovechan de la información que éstas requieren para operar correctamente y la manejan de manera incorrecta. Aunque los usuarios tienen acceso a información sobre las aplicaciones, incluyendo los datos que estas comparten y recopilan según las políticas de privacidad, los análisis revelan que esta información a menudo están alejados de la realidad. Esto sugiere que las políticas de privacidad de Google no se están cumpliendo de manera estricta, lo que indica la necesidad de implementar auditorías más rigurosas que garanticen su cumplimiento y que fortalezcan el ecosistema de Android y Google en general.

En esta línea, hay que destacar lo peligroso que puede llegar a ser la filtración de ciertos PII, específicamente el ID de Publicidad. A simple vista este dato parece insignificante, pero considerando la cantidad de hallazgos asociados a este identificador y que éste puede ser la llave al perfil completo de un usuario, no debería tomarse a la ligera. Al tomar en cuenta que los servidores que más interactúan con él son los servidores de analítica y publicidad, y que éstos almacenan una cantidad importante de información sobre el comportamiento de los usuarios, el panorama actual se torna preocupante. La realidad es que, aunque estos servicios ofrecen a los usuarios la opción de solicitar la eliminación de su información, resulta imposible rastrear cada uno de los servidores que han accedido a dichos datos una vez que abandonan el dispositivo.

En resumen, en cuanto al nivel de privacidad de la información que ofrecen las aplicaciones en la actualidad, los resultados obtenidos sugieren que es considerablemente deficiente. El poco cumplimiento de prácticas establecidas por Google, indican que la única forma de asegurar que los datos sensibles no abandonen el dispositivo, o más bien, que no sean utilizados para otros propósitos, es preocuparse activamente de que no caigan en manos de terceros. Sin embargo, la sociedad actual y su dependencia en los dispositivos móviles en conjunto con un bajo conocimiento de prácticas básicas de seguridad hacen esto básicamente imposible. Aún cuando existen personas con los conocimientos necesarios para reducir al máximo las fugas de información, no existe un método que sea completamente infalible.

## 6.4. Trabajo Futuro

Como último punto, es importante resaltar aquellos aspectos en los que el estudio no profundizó lo suficiente, o bien, aquellas maneras en las que técnica de análisis puede servir como base para próximos proyectos y aportes.

- En primer lugar, el estudio es capaz de detectar solamente aquellos PII que se transmiten de manera completa o con variaciones relativamente pequeñas. Ahora bien, la realidad es que existen una serie de diversos tipos de *encodings* utilizados para transmitir información, además de una serie de métodos de ofuscación que en ciertos casos esconden las posibles fugas de información. Robustecer el algoritmo de *string matching* para considerar estos escenarios permitiría enriquecer el estudio y resultar en la detección de más hallazgos de alto nivel de riesgo.
- En el contexto del punto anterior, otro gran aspecto que podría ser abordado corresponde al formato de los PII que fueron utilizados. En el presente estudio se redujo su uso a información solamente en formato de texto y de datos personales, sin profundizar mucho en archivos multimedia, o información relacionada al comportamiento del usuario y sus interacciones con las aplicaciones. Un primer acercamiento a este problema en particular requiere que se identifique en primer lugar, la forma en que éstos son transmitidos a través del *payload* de los paquetes salientes, y cómo identificar

patrones que posteriormente puedan ser interpretados y debidamente catalogados como fugas.

- Finalmente, una última mejora que se le podría hacer al estudio realizado, es extenderlo a un sistema dentro de dispositivos reales que utilicen la información recabada del análisis de las aplicaciones. Dado que actualmente no hay ninguna aplicación que pueda acceder al *payload* de los paquetes salientes, el análisis no se puede realizar en tiempo real de la manera en la que se hizo en el estudio. Sin embargo, es factible utilizar las directrices seguidas por el presente estudio para construir una herramienta que base su análisis en la metadata del paquete y la información disponible de las aplicaciones en la Play Store para categorizar el tráfico malicioso y de esta forma, hacerlo factible en tiempo real.

# REFERENCIAS BIBLIOGRÁFICAS

[Alkindi *et al.*, 2021] Alkindi, Z. R., Sarrab, M., y Alzeidi, N. (2021). User privacy and data flow control for android apps: Systematic literature review. *Journal of Cyber Security and Mobility*.

[Android Developers, 2024a] Android Developers (2024a). Android platform guide. `https://developer.android.com/guide/platform`. Accessed: 2023-06-02.

[Android Developers, 2024b] Android Developers (2024b). Security with network protocols. `https://developer.android.com/privacy-and-security/security-ssl`. Accessed: 2024-01-08.

[Arp *et al.*, 2014] Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., Rieck, K., y Siemens, C. (2014). Drebin: Effective and explainable detection of android malware in your pocket. En *Ndss*, volumen 14, pp. 23–26.

[Aswal *et al.*, 2023] Aswal, K., Pathak, H., Singh, N., y Gupta, N. (2023). Strength and limitations of publicly available anti-malware tools against obfuscated malware. En *2023 5th International Conference on Inventive Research in Computing Applications (ICIRCA)*, pp. 1261–1266. IEEE.

[AV-Comparatives, 2023] AV-Comparatives (2023). Mobile security review 2023. `https://www.av-comparatives.org/tests/mobile-security-review-2023/`. Accessed: 2023-05-18.

[Conti *et al.*, 2018] Conti, M., Li, Q. Q., Maragno, A., y Spolaor, R. (2018). The dark side (-channel) of mobile devices: A survey on network traffic analysis. *IEEE communications surveys & tutorials*, 20(4):2658–2713.

[CVE Details, 2022] CVE Details (2022). Top 50 products by total number of 'distinct' vulnerabilities. `https://www.cvedetails.com/top-50-products.php?year=2022`. Accessed: 2023-05-18.

[del Moral, 2016] del Moral, M. G. (2016). *Malware en Android: Discovering, Reversing & Forensics*. ZeroXword Computing.

[Enck *et al.*, 2010] Enck, W., Gilbert, P., Chun, B.-G., Cox, L., Jung, J., McDaniel, P., y Sheth, A. (2010). Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. volumen 57, pp. 393–407.

[Gajrani *et al.*, 2020] Gajrani, J., Laxmi, V., Tripathi, M., Gaur, M. S., Zemmari, A., Mosbah, M., y Conti, M. (2020). Effectiveness of state-of-the-art dynamic analysis techniques in identifying diverse android malware and future enhancements. En *Advances in Computers*, volumen 119, pp. 73–120. Elsevier.

[Google, 2020] Google (2020). Cloud-based protections. `https://developers.google.com/android/play-protect/cloud-based-protections`. Accessed: 2023-06-02.

[Google, 2024] Google (2024). Google play console. `https://play.google.com/console`. Accesed: 2024-06-20.

[Hammad *et al.*, 2018] Hammad, M., Garcia, J., y Malek, S. (2018). A large-scale empirical study on the effects of code obfuscations on android apps and anti-malware products. En *Proceedings of the 40th international conference on software engineering*, pp. 421–431.

[Hande y Rao, 2017] Hande, N. y Rao, V. (2017). A comparative study of static, dynamic and hybrid analysis techniques for android malware detection. *International Journal of Engineering Development and Research*, 5:1433–1436.

[Hourihan, 2022] Hourihan, D. (2022). 15 alarming cyber security facts and stats. `https://www.cybintsolutions.com/cyber-security-facts-stats/`. Accessed: 2023-05-18.

[International Organization for Standardization, 2022] International Organization for Standardization (2022). *ISO/IEC 27002:2022: Information security, cybersecurity and privacy protection – Information security controls*. International Organization for Standardization.

[Jyothi y Reddy, 2018] Jyothi, K. K. y Reddy, B. I. (2018). Study on virtual private network (vpn), vpn's protocols and security. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 3(5):919–932.

[Kapratwar, 2016] Kapratwar, A. (2016). Static and dynamic analysis for android malware detection.

[Ma'azer Al Fawareh y Jusoh, 2017] Ma'azer Al Fawareh, H. y Jusoh, S. (2017). The use and effects of smartphones in higher education. *Ijim*, 11:103.

[Muhammad *et al.*, 2023] Muhammad, Z., Amjad, M., Iqbal, Z., Javed, A. R., y Gadekallu, T. (2023). Circumventing google play vetting policies: a stealthy cyberattack that uses incremental updates to breach privacy. *Journal of Ambient Intelligence and Humanized Computing*.

[Novak *et al.*, 2020] Novak, E., Aung, P. T., y Do, T. (2020). Vpn+ towards detection and remediation of information leakage on smartphones. En *2020 21st IEEE International Conference on Mobile Data Management (MDM)*, pp. 39–48. IEEE.

[O'Dea, 2023] O'Dea, S. (2023). Forecast number of mobile users worldwide 2020-2025. `https://www.statista.com/statistics/218984/number-of-global-mobile-users-since-2010/`. Accessed: 2023-09-16.

[Po, 2020] Po, D. K. (2020). Similarity based information retrieval using levenshtein distance algorithm. *Int. J. Adv. Sci. Res. Eng*, 6(04):06–10.

[Powell *et al.*, 2021] Powell, L. M., Swartz, J., y Hendon, M. (2021). Awareness of mobile device security and data privacy tools. *Issues in Information Systems*, 22(1).

[Pushkar, 2024] Pushkar, A. (2024). Boyer moore algorithm: Approaches, complexity, and applications. `https://intellipaat.com/blog/boyer-moore-algorithm/`. Accessed: 2024-06-20.

[Rafter, 2022] Rafter, D. (2022). Android vs. ios: Which is better for security? `https://us.norton.com/blog/mobile/android-vs-ios-which-is-more-secure`. Accessed: 2023-05-18.

[Razaghpanah *et al.*, 2015] Razaghpanah, A., Vallina-Rodriguez, N., Sundaresan, S., Kreibich, C., Gill, P., Allman, M., y Paxson, V. (2015). Haystack: In situ mobile traffic analysis in user space. *arXiv preprint arXiv:1510.01419*, pp. 1–13.

[Ren *et al.*, 2016] Ren, J., Rao, A., Lindorfer, M., Legout, A., y Choffnes, D. (2016). Recon: Revealing and controlling pii leaks in mobile network traffic. En *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 361–374.

[Reyes *et al.*, 2018] Reyes, I., Wijesekera, P., Reardon, J., Elazari Bar On, A., Razaghpanah, A., Vallina-Rodriguez, N., y Egelman, S. (2018). "won't somebody think of the children?" examining coppa compliance at scale.

[Saad, 2022] Saad, L. (2022). Americans have close but wary bond with their smartphone. `https://news.gallup.com/poll/393785/americans-close-wary-bond-smartphone.aspx`. Accessed: 2023-05-18.

[Shahzad *et al.*, 2022] Shahzad, L., Ahmad, E., Sadiq, T., y Sohail, F. (2022). A survey paper on smartphones data security, challenges and awareness. En *2022 International Conference on Decision Aid Sciences and Applications (DASA)*, pp. 1653–1658.

[Shuba *et al.*, 2018] Shuba, A., Bakopoulou, E., Mehrabadi, M. A., Le, H., Choffnes, D., y Markopoulou, A. (2018). Antshield: On-device detection of personal information exposure. *arXiv preprint arXiv:1803.01261*.

[Shuba *et al.*, 2016] Shuba, A., Le, A., Alimpertis, E., Gjoka, M., y Markopoulou, A. (2016). Antmonitor: A system for on-device mobile network monitoring and its applications. *arXiv preprint arXiv:1611.04268*.

[Song y Hengartner, 2015] Song, Y. y Hengartner, U. (2015). Privacyguard: A vpn-based platform to detect information leakage on android devices. En *Proceedings of the 5th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices*, pp. 15–26.

[Stat Counter, 2023] Stat Counter (2023). Mobile operating system market share worldwide. `https://gs.statcounter.com/os-market-share/mobile/worldwide`. Accessed: 2024-01-08.

[Stat Counter, 2024] Stat Counter (2024). Mobile android version market share worldwide. https://gs.statcounter.com/android-version-market-share/mobile/worldwide/2023. Accessed: 2024-01-08.

[Sun *et al.*, 2016] Sun, M., Wei, T., y Lui, J. C. (2016). Taintart: A practical multi-level information-flow tracking system for android runtime. En *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 331–342.

[Tasheva, 2021] Tasheva, I. (2021). Cybersecurity post-covid-19: Lessons learned and policy recommendations. *European View*, 20(2):140–149.

[Terkki *et al.*, 2017] Terkki, E., Rao, A., y Tarkoma, S. (2017). Spying on android users through targeted ads. pp. 87–94.

[The Yale Ledger, 2022] The Yale Ledger (2022). What are the risks of a data leak? https://campuspress.yale.edu/ledger/what-are-the-risks-of-a-data-leak/. Accessed: 2023-05-18.

[Vashchegin, 2024] Vashchegin, R. (2024). Conquista la búsqueda de cadenas con el algoritmo aho-corasick. https://www.toptal.com/algorithms/conquista-la-busqueda-de-cadenas-con-el-algoritmo-aho-corasick. Accessed: 2024-06-20.

[You *et al.*, 2017] You, W., Liang, B., Shi, W., Wang, P., y Zhang, X. (2017). Taintman: An art-compatible dynamic taint analysis framework on unmodified and non-rooted android devices. *IEEE Transactions on Dependable and Secure Computing*, 17(1):209–222.

# ANEXOS

## A. Aplicaciones analizadas

| Categoría | Aplicación | |
|---|---|---|
| **Arte y Diseño** |  | Canva: Design, Photo & Video |
| |  | AR Drawing Sketch Paint |
| |  | Sketchbook |
| **Autos y Vehículos** |  | Chileautos |
| |  | Driving Instructor - Theory Test |
| |  | INFRACCIÓN DE MULTAS - CHILE |
| **Belleza** |  | Face Shape - Pretty Scale |
| |  | Beauty Mirror, The Mirror App |
| |  | Skincare and Face Care Routine |
| **Libros y Referencias** |  | FamilySearch Tree |
| |  | Wattpad - Read & Write Stories |
| |  | Bible App Lite - NIV Offline |
| **Negocios** |  | Microsoft Teams |
| |  | LinkedIn: Jobs & Business News |
| |  | Language Translator: Translate |
| **Cómics** |  | Manta: Comics & Graphic Novels |
| |  | Mango: Lector de Manga Español |
| |  | Pelisplay ver peliculas hd |

| Categoría | Aplicación | |
|---|---|---|
| **Comunicación** |  | WhatsApp Messenger |
| |  | Zello PTT Walkie Talkie |
| |  | Stickers 2024 - WASticker |
| **Conocer Personas** |  | Dating with Singles - iHappy |
| |  | Dating and Chat - SweetMeet |
| |  | CUPI CHAT: Dating, Flirt, Meet |
| **Educación** |  | Duolingo |
| |  | Headway: 15-Min Book Summaries |
| |  | Test de Conducir PracticaTest |
| **Entretenimiento** |  | Vix: TV, Deportes y Noticias |
| |  | Hair Cliper Prank: Fun Sounds |
| |  | Pluto TV: Watch TV & Movies |
| **Eventos** |  | Invitation Maker: Card Creator |
| |  | Fever: Local Events & Tickets |
| |  | Epic Freebie Games Radar |
| **Finanzas** |  | XTB - Precios, análisis y más |
| |  | Salcobrand |
| |  | Western Union Send Money |
| **Comer y Beber** |  | PedidosYa - Delivery Online |
| |  | Papa John’s Chile |
| |  | Club Dominó |

| Categoría | Aplicación | |
|---|---|---|
| **Salud y Fitness** |  | Flo Period & Pregnancy Tracker |
| |  | Weight Loss for Women: Workout |
| |  | Jawline Exercises - Face Yoga |
| **Inmuebles y Hogar** |  | ComunidadFeliz |
| |  | Planner 5D: Home Design, Decor |
| |  | ColorSnap Visualizer |
| **Bibliotecas y Demostración** |  | Mod Menu For RBX |
| |  | Biblia de estudio en español |
| |  | PSP Ultimate Database Game Pro |
| **Estilo de Vida** |  | LipiApp MiCilindro |
| |  | Nicequest |
| |  | AttaPoll - Paid Surveys |
| **Mapas y Navegación** |  | Moovit: Horario de bus y tren |
| |  | Santiago Bus Checker |
| |  | Radar GO-X: HUD, GPS, Maps |
| **Medicina** |  | My Calendar - Period Tracker |
| |  | Medicamentos via parenteral |
| |  | Body Temperature - Fever Tracker |
| **Música y Audio** |  | Spotify: Music and Podcasts |
| |  | Radio Chile - FM, online radio |
| |  | Learn Piano - Real Keyboard |

| Categoría | Aplicación | |
|---|---|---|
| **Noticias y Revistas** |  | LUN.COM |
| |  | DIARIO FINANCIERO |
| |  | BBC Mundo |
| **Ser Padres** |  | Be Closer: Family Location |
| |  | Pregnancy Tracker: amma |
| |  | Phone Tracker By Number |
| **Personalización** |  | Control Center Simple |
| |  | Battery Charging Animation |
| |  | Edge Lightning: LED Borderlight |
| **Fotografía** |  | Remini - AI Photo Enhancer |
| |  | Time Wrap Scan: Face Scan |
| |  | Sticker Maker - WAStickers |
| **Productividad** |  | PDF Reader - PDF Viewer |
| |  | Antivirus: Virus Cleaner, Junk |
| |  | Basic Calculator: GPA & Math |
| **Compras** |  | SHEIN - Compras en línea |
| |  | Samsung Shop |
| |  | Cashback - LetyShops |
| **Social** |  | Kismia - Meet Singles Nearby |
| |  | SOSAFE - City Social Network |
| |  | Pinterest Lite |

| Categoría | Aplicación | |
|---|---|---|
| **Deportes** |  | 365Scores: Live Scores & News |
| |  | UFC |
| |  | OneFootball Resultados en vivo |
| **Herramientas** |  | AVG AntiVirus & Security |
| |  | Clean Master Ultra Security |
| |  | Phone Cleaner - AI Cleaner |
| **Viajes** |  | Viajes Falabella |
| |  | Despegar: Vuelos y Hoteles |
| |  | Google Earth |
| **Aplicaciones de Videos** |  | HD Video Player All Formats |
| |  | Reproductor TV Chilena |
| |  | Provid - Video Player |
| **Tiempo** |  | Daily Weather |
| |  | Windy.com - Weather Forecast |
| |  | Digital Compass for Android |