EasyChair Preprint

№ 13374

A Data Warehouse for Mobile Application Privacy and Security Study

Alejandro Pérez-Fuente, M. Mercedes Martínez-González, Amador Aparicio and Quiliano Isaac Moro

May 19, 2024

Un *Data Warehouse* para el Estudio de la Privacidad y Seguridad de Aplicaciones Móviles

Alejandro Pérez-Fuente, M. Mercedes Martínez-González, Amador Aparicio, Quiliano Isaac Moro
Grupo de Investigación en Ingeniería de la Privacidad de la Universidad de Valladolid
apppi@infor.uva.es

***Resumen*—La privacidad juega un papel sustancial en las aplicaciones móviles ya que cada día se procesa y almacena una mayor cantidad de datos en los dispositivos móviles. Sin embargo, no se dispone de estudios sobre el impacto que generan las aplicaciones sobre la privacidad de sus usuarios, que estos puedan consultar para empoderarse en la protección de su privacidad. Entre las causas, la falta de repositorios fácilmente accesibles para los investigadores que almacenen metadatos relativos a la privacidad de las aplicaciones. En este artículo presentamos un repositorio con metadatos de aplicaciones Android capaz de aportar metadatos de calidad a quienes analizan la seguridad y privacidad de las aplicaciones móviles. La calidad de los datos se garantiza desde el diseño, tanto de los procesos de integración de datos, como su seguridad y la del repositorio. Actualmente, el repositorio es accesible y consta de metadatos de un total de 9068 aplicaciones.**

***Index Terms*—Seguridad, Privacidad, Warehousing, Repositorio, aplicaciones Android**

Tipo de contribución: *Investigación original*

I. Introducción

En la actualidad, los teléfonos móviles son un elemento de uso cotidiano. Millones de personas tienen acceso a un teléfono móvil que les permite navegar por internet, acceder a redes sociales... Ya no se trata solo de un medio para comunicarse mediante llamadas de voz o mensajes de texto, sino que también se han convertido en una herramienta esencial para el trabajo, el entretenimiento y la educación. La importancia que tienen los teléfonos móviles en la sociedad actual es innegable y todo apunta a que lo seguirá siendo.

Entre todas las aplicaciones que utilizamos merecen especial atención las que acceden a información personal, como fotografías, la ubicación y la lista de contactos, y las que utilizamos para realizar actividades privadas, como pedir citas médicas. En muchas ocasiones, pese a que la recopilación y utilización de datos personales está regulada por la Ley Orgánica de Protección de Datos y Derechos Digitales [1], las aplicaciones acceden a información sensible sin que el usuario sepa con claridad que uso se hará de esta [2].

Existen varios sistemas operativos de teléfonos móviles, pero los tres principales son: Android, iOS y Windows Phone. Android, desarrollado por Google, es el sistema operativo que mayor cuota de mercado abarca, un 86,2 % [3]. Por ello, este trabajo se centra en las aplicaciones Android.

Hasta donde conocemos, no hay herramientas que ayuden a detectar este problema en el ámbito de las aplicaciones móviles sin requerir instalación en los dispositivos como es el caso de CONAN mobile [4]. Por ello, en este trabajo se propone una solución similar a las que analizan la privacidad en sitios web, por ej., PrivacyScore [5], pero aplicado a aplicaciones móviles.

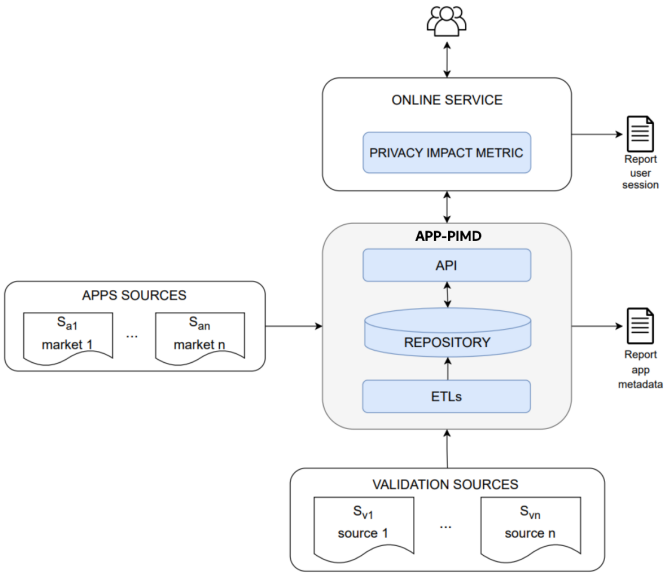

Figura 1. Arquitectura de App-PI.

Nuestra propuesta es *App Privacy Impact* (App-PI), un ecosistema para la evaluación del impacto de *apps* para dispositivos móviles sobre la privacidad y seguridad de sus usuarios. Este ecosistema consta de las siguientes partes (ver Fig. 1):

- Repositorio y los procesos ETL[1] asociados: un almacén de metadatos de aplicaciones Android. Se utilizan para calcular el impacto en la privacidad de los usuarios [6].
- API[2]: un acceso abierto de los contenidos del *warehouse* para servicios e investigadores [6].
- Una serie de métricas que calculen el impacto de las aplicaciones en la privacidad de los usuarios haciendo uso de la información del *warehouse*.
- Aplicaciones para usuarios finales que implementan las métricas y ofrecen servicios de usuario, permitiendo así empoderar al usuario final en la toma de decisiones respecto a su privacidad [7].

Dentro de este ecosistema, el principal objetivo del *warehouse* es poner a disposición de investigadores y servicios toda la información y metadatos necesarios para poder analizar el impacto de una aplicación sobre la privacidad y seguridad de sus usuarios. En este artículo nos centramos en

[1]ETL: *Extract, Transform, Load.*
[2]API: *Application Programming Inteface.*

este componente, App-PIMD (*App Privacy Impact MetaData*) donde se incluye el repositorio y la API. Actualmente[3], el repositorio es accesible[4] y consta de metadatos de un total de 9068 aplicaciones.

La estructura del artículo es la siguiente: la Sección II revisa otras investigaciones sobre almacenes de aplicaciones Android; la Sección III muestra los distintos componentes del repositorio así como las decisiones de diseño del mismo más relevantes, entre las que destacamos las relativas a la seguridad desde el diseño, y cómo contribuye a garantizar la calidad de los datos almacenados; la Sección IV aporta datos sobre el contenido del repositorio y unas pruebas del mismo; la Sección V realiza un análisis retrospectivo de alguna de las decisiones tomadas; la Sección VI presenta las conclusiones obtenidas de este trabajo; la Sección VII muestra las posibles líneas de trabajo futuro que plantea este proyecto.

II. Trabajo Relacionado

En la actualidad existen algunos almacenes de aplicaciones [8]. Entre los más voluminosos se tiene el conjunto de aplicaciones de Androzoo [9] y el de AndroVault [10]. El primero de estos almacena los archivos *.apk* de un gran número de aplicaciones, mientras que el segundo se centra en la extracción de conocimiento de aplicaciones por medio de diversos análisis. Aunque almacenan una gran cantidad de aplicaciones, no hay metadatos relativos a la privacidad y seguridad de las aplicaciones.

En el caso de Androzoo, en diciembre de 2023 se ha añadido la extracción de metadatos de Google Play. Esto es claramente un indicador del interés en el estudio de las aplicaciones móviles. No obstante, su repositorio sigue siendo de carácter general. No tiene en cuenta los grupos de permisos, posibles puntuaciones de privacidad, o la naturaleza estructurada de algunos de los metadatos como los AOSP[5]. Todos ellos datos que pueden ayudar a construir herramientas que empoderen a los usuarios en la protección de su privacidad.

Un proyecto que intentó abordar este problema fue Tacyt [11], de Telefónica España. Este pretendía clasificar millones de aplicaciones móviles mediante su análisis. No obstante, el proyecto fue descontinuado en 2021. Otro que sí continúa abierto es CONAN mobile, del INCIBE. Sin embargo, su enfoque es diferente, analiza los dispositivos en busca de amenazas. CONAN mobile requiere su instalación en nuestros dispositivos y solamente analiza aplicaciones ya instaladas en nuestro dispositivo [4].

También existen repositorios donde se puede encontrar el código de las aplicaciones [12]. No obstante, debido a la naturaleza de software propietario por parte de la mayoría de aplicaciones populares como Whatsapp, Microsoft Teams y TikTok en el ámbito de las redes sociales, o CapCut en el de edición de vídeo, estos repositorios sólo contienen el código de aplicaciones de código abierto. Esto limita mucho su potencial a la hora de realizar análisis de la intrusividad en la privacidad, ya que la mayoría de usuarios utiliza las aplicaciones más populares, que suelen ser las de software propietario.

Los repositorios de aplicaciones que almacenan información sobre la intrusividad en la privacidad son muy reducidos [13], [14]. Generalmente estos vienen asociados a un estudio de aplicaciones concretas, lo que dificulta la generalización de los resultados a otras aplicaciones.

III. El repositorio de App-PI

La construcción del repositorio se plantea como un servicio de consulta para aplicaciones que realizan análisis para conocer el impacto sobre la privacidad y seguridad de los usuarios a partir de sus propiedades características, tal como las han definido sus desarrolladores, independientemente del dispositivo en el que se instala.

En consecuencia, se diseñaron los componentes del ecosistema que recopilan metadatos, acorde a un modelo de *warehousing*. Los procesos ETL extraen metadatos de fuentes de datos. Estos datos se cargan en un repositorio, que se consulta utilizando una API (ver Fig. 1). Los datos se almacenan en una base de datos MySQL. La interacción se ofrece a través de una API que redirecciona las consultas a la base de datos. Se decidió no ofrecer acceso directo con consultas SQL por seguridad. La tecnología que soporta esta API se detalla en la sección III-B.

III-A. El Repositorio App-PIMD

Para alcanzar el objetivo anteriormente mencionado, se perfilan distintos tipos de usuario del repositorio:

- **Usuario general:** Es el tipo de usuario más frecuente y no es necesario estar registrado para trabajar con este perfil. Este tipo de usuario puede consultar los contenidos del *warehouse*. También puede solicitar la carga de nuevas aplicaciones para enriquecerlo.
- **Usuarios expertos:** Este tipo de usuario puede puntuar la intrusividad en la privacidad de las aplicaciones. De este modo, se aumenta la calidad de la información del *warehouse* en este aspecto gracias a tener múltiples fuentes heterogéneas de información. Está previsto para capturar la opinión verificada de expertos en privacidad y/o seguridad. La diferencia sustancial con el perfil anterior es que este tipo de usuario sí puede modificar el contenido -aunque, nunca los esquemas- del *warehouse* mientras que el usuario general solo puede solicitar la inserción de nuevos datos. Este tipo de usuario necesita estar registrado.
- **Usuario administrador:** Este tipo de usuario es el encargado de actualizar directamente el *warehouse*, así como asegurar la calidad de los datos del mismo, por ejemplo, llevando a cabo la carga masiva de aplicaciones. Por ello, está reservado para el grupo de investigación que mantiene el *warehouse*.

Se diseña el repositorio en base a los siguientes criterios: modularidad, escalabilidad y seguridad. Los dos primeros permiten que este se pueda reutilizar y ampliar según los cambios que vayan surgiendo en el mundo de las aplicaciones Android. En cuanto a la seguridad, se plantea como un criterio imprescindible para poder conseguir que haya datos de calidad almacenados en el repositorio.

[3] A fecha 26/02/2024.
[4] https://apkfalcon.infor.uva.es:8080
[5] AOSP: *Android Open Source Project*.

Atendiendo a estos criterios, se decide utilizar una arquitectura de *Data Warehouse* por ser la más limpia que cumple con las necesidades del repositorio. En esta, se separan los componentes de acceso al repositorio y carga de datos, llevándose a cabo estos por medio de la API y los procesos ETL respectivamente (ver Fig. 1). Añadiéndose de este modo seguridad al repositorio gracias a desacoplar las distintas responsabilidades en componentes.

Además, hacer uso de técnicas ETL permite extraer grandes cantidades de datos de fuentes heterogéneas, transformarlos en un esquema común y, finalmente, cargarlos en un repositorio centralizado, el *warehouse*.

III-B. La Seguridad de App-PIMD

El repositorio App-PIMD introduce estas estrategias para contemplar la seguridad desde el diseño:

- Uso de distintos roles de usuarios que previenen cargas de datos malintencionadas que podrían afectar negativamente a la calidad de los datos.
  Gracias a los distintos tipos de usuario, se puede limitar la funcionalidad asociada a cada uno de estos en función de su calidad como fuente de información. Por ejemplo, los usuarios generales que no están registrados serían una fuente potencial de datos de baja calidad si se dedicasen a subir información falsa al *warehouse*. Por ello, estos no pueden actualizar de manera directa el contenido del *warehouse*. Sin embargo, los usuarios expertos (especialistas en privacidad y/o seguridad) sí que representan una fuente de información de calidad. Por ello, sí pueden actualizar directamente los contenidos del *warehouse*.
- Se incorpora una API para el acceso al *warehouse*. De este modo, se protege el repositorio contra ataques de tipo SQLi [15] y, se independiza el componente de acceso haciendo que los usuarios solo puedan modificar los datos almacenados a través de las interfaces que aporta la API. Gracias a que toda la información modificada del repositorio es gestionada por la API, se aumenta la uniformidad y calidad de los datos almacenados.
- Se hace uso de fuentes de datos de calidad. En referencia a los metadatos sobre el sistema operativo Android y su sistema de permisos, se usan las webs oficiales del sistema operativo: AndroidDevelopers [16], [17] y GoogleGit [6], donde se encuentra respectivamente la documentación y el código fuente de Android. Para los metadatos de aplicaciones se usan múltiples fuentes: Androzoo [9], Apkpure [7], Evozi [8]... Para las medidas de intrusividad de las aplicaciones se usan también múltiples fuentes: Tosdr [18], métrica App-PI [19], estudios pasados a este respecto [13], [14]...
- Todos los metadatos se extraen de los archivos empaquetados de la aplicación (*.apk*). De este modo, al extraer la información directamente de las aplicaciones, se asegura la uniformidad entre las distintas fuentes de datos y la calidad de estos. Además, esto permite realizar comparaciones entre datos de distintas fuentes.
- Se almacenan los *hash* SHA256 de los archivos de aplicación (*.apk*). Esto es muy importante ya que por medio del *hash* se puede comprobar si una aplicación ha sido comprometida.
  Por ejemplo, digamos que se tiene una versión *X* de la aplicación *com.whatsapp* obtenida de Google Play y de una página web de dudosa reputación. Si los *hash* de ambos archivos son diferentes, significará que pese a ser la misma versión de aplicación, su contenido es diferente y, por tanto, una de las dos aplicaciones ha sido comprometida.
- Se almacena el origen de las aplicaciones, así como de las puntuaciones que estas reciben sobre lo intrusivas que son en la privacidad. De este modo, se tiene una trazabilidad completa de todos sus metadatos, lo que permite subsanar posibles deficiencias en la calidad de los mismos. Por ejemplo, si se descubre que una fuente de información resulta ser de baja calidad, se podrían omitir las aplicaciones de dicha fuente.
- Se hace uso de los protocolos y estándares habituales de los servicios web. Para la comunicación, se utiliza el protocolo HTTPS que garantiza la integridad y la seguridad en la transferencia de los datos gracias al cifrado de estos. En concreto, se ha utilizado una clave RSA de 4096 bits. Para la autenticación de los usuarios, el estándar OAuth 2.0 [20], el cual emplea la cabecera HTTP *Authorization*. Finalmente, para la definición de la API, se utiliza el estándar OpenAPI 3 [21]. Todo esto facilita la interacción segura entre los usuarios y el repositorio.
- Se han implementado protecciones necesarias para evitar posibles sobrecargas del sistema y ataques de denegación de servicio distribuidos (DDoS [9]). En este respecto, se limita la cantidad de recursos máxima que se puede asociar a las peticiones de tal modo que nunca se supere dicho límite. También se han limitado la cantidad máxima de peticiones concurrentes permitidas en las funcionalidades especialmente pesadas, como la carga de aplicaciones nuevas al repositorio.

III-C. El Warehouse y los ETL

Como se ha mencionado anteriormente, el repositorio se nutre de múltiples fuentes de datos heterogéneas para incrementar la calidad de la información que almacena. A continuación, se describe el flujo general de datos del *warehouse* el cual incluye el proceso completo de carga de datos y, los distintos procesos ETL que gestionan el flujo de cada fuente de datos.

Tal y como se aprecia en la Fig. 2, el *warehouse* almacena 3 tipos de datos diferentes provenientes de distintos tipos de fuentes de datos (HTML, APIs y archivos):

- **Aplicaciones.** Las fuentes de datos de aplicaciones son: la fuente API de Androzoo [9] y, las fuentes HTML: Apkpure, Evozi, Apkmonk [10] y Apkfollow [11].

[6]https://android.googlesource.com/
[7]https://apkpure.net/es/
[8]https://apps.evozi.com/apk-downloader/

[9]DDoS: *Distributed Denial of Service.*
[10]https://www.apkmonk.com/
[11]https://www.apkfollow.com/es/

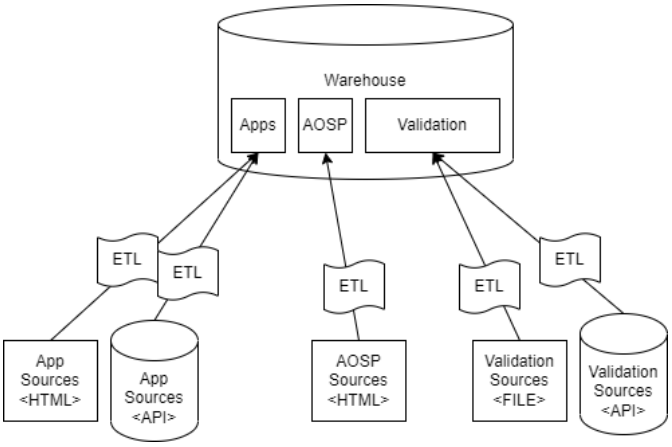

Figura 2. Modelo ETL y flujo de datos del *warehouse*.

De cada aplicación se almacenan metadatos identificativos de la misma como: su nombre de paquete, su *hash* SHA256, su versión, su categoría y metadatos sobre su extracción como la fuente y fecha entre otros. En referencia a los metadatos relacionados con la intrusividad en la privacidad se almacenan los permisos que esta utiliza, los que define, los grupos de permisos y las puntuaciones de intrusividad obtenidas por la aplicación ente otros.

- **AOSP.** Las fuentes de los metadatos de Android y su sistema de permisos son las páginas web HTML de documentación [12] y código del sistema Android [13].
  Se almacenan los permisos y grupos de permisos que declara el sistema operativo Android (ejemplo de permiso: *android.permission.READ_SMS*). De cada permiso se almacena el nivel de protección o tipo, su identificador o nombre, en qué versión del sistema se incluyó y, el grupo de permisos al que pertenece. Además, de cada permiso de tipo *dangerous* se almacena una puntuación sobre el impacto en la privacidad que este tiene [22]. Estos son los metadatos que utilizan las métricas que evalúan la privacidad [19], y que también están relacionados con la seguridad de los datos que Google incorpora en su *market* para informar a sus usuarios sobre la seguridad y privacidad de sus datos en las aplicaciones [23].
- **Validación.** Las fuentes de datos que miden el nivel de intrusividad en la privacidad de las aplicaciones son: la API de Tosdr [18], los archivos de texto de estudios pasados a este respecto [13], [14], y nuestra propia API por medio de la opinión de los expertos. Estos datos son puntuaciones asociadas a cada aplicación concreta.

Cada tipo de dato almacenado puede provenir de diversas fuentes de datos de distinto tipo HTML, API o archivo. Por tanto, tienen un proceso ETL distinto asociado.

III-C1. Flujos ETL de aplicaciones: La Fig. 3 muestra el flujo de datos en el proceso ETL que carga los metadatos de una aplicación en el *warehouse*. La entrada es el nombre de paquete de la aplicación a descargar (ej. *net.universia.uva*). La categoría se extrae del *market* oficial de Android, Google Play.

Google se protege frente a las descargas masivas de aplicaciones para evitar que otras personas creen *markets* que compitan contra ellos [9]. Para solventar las trabas que este *market* tiene a las extracciones automáticas masivas se ha trabajado con las URL, haciendo búsquedas de expresiones regulares. Aunque en nuestro caso no aplicamos descargas masivas de aplicaciones a este *market*, los mecanismos de defensa de Google también dificultan nuestra tarea.

[12]https://developer.android.com/
[13]https://android.googlesource.com/

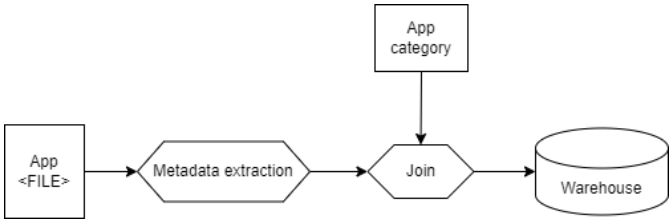

Figura 3. Flujo de datos de subida de una aplicación al *warehouse*.

A continuación, se descarga el archivo empaquetado de la aplicación (*.apk*) de una de las distintas fuentes de datos del *warehouse*. En el caso de la API de Androzoo, se realiza una petición a la API con el *hash* de la aplicación a descargar como parámetro. En el caso de las páginas web anteriormente mencionadas, se aplica un proceso de *scrapping* sobre las mismas que da como resultado el enlace de descarga directo de dicha aplicación.

Estos procesos de *web scrapping* se han implementado mayoritariamente con búsquedas de patrones haciendo uso de expresiones regulares sobre los *links* de cada página, de tal modo que se independiza de su interfaz visual. De esta manera se prolonga la vida de los *scrapping* desarrollados ya que, por norma general, las interfaces web son mucho más cambiantes que los esquemas de los recursos (los esquemas de las URL).

Una vez se tiene el archivo de aplicación empaquetado, se extrae el archivo *AndroidManifest.xml* el cual contiene todos los metadatos sobre permisos y grupos de permisos de forma semiestructurada (XML) [16], [17]. Tal y como se ve en la Fig. 3, el paso final es integrar la información extraída del *.apk* con la categoría de la aplicación, pudiéndose así insertar todos los metadatos extraídos en el *warehouse*.

III-C2. Flujos ETL de AOSP: La Fig. 4 muestra el flujo de datos para cargar los metadatos de Android y su sistema de permisos y grupos de permisos.

A partir del fichero *AndroidManifest.xml* del *core/res/* del código fuente del sistema operativo, se extraen los permisos y grupos de permisos declarados por este. Este fichero tiene la misma estructura que el *AndroidManifest.xml* que contienen las aplicaciones lo que facilita enormemente la extracción de información del mismo [16], [17]. A partir del identificador de los permisos y grupos de permisos, se consigue mediante *scrapping* en la página web de documentación del sistema operativo (Android Developers) la versión de la API en que se añadieron. Una vez se tiene, se integran los datos y son insertados en el *warehouse*.

III-C3. Flujos ETL de validación: Para cargar la información relativa al nivel de intrusividad de una aplicación en la privacidad de los usuarios, se busca si las fuentes de validación (Tosdr [18] o los estudios pasados [13], [14]) aportan una puntuación para esta.

En el caso de Tosdr se realiza por medio de una petición

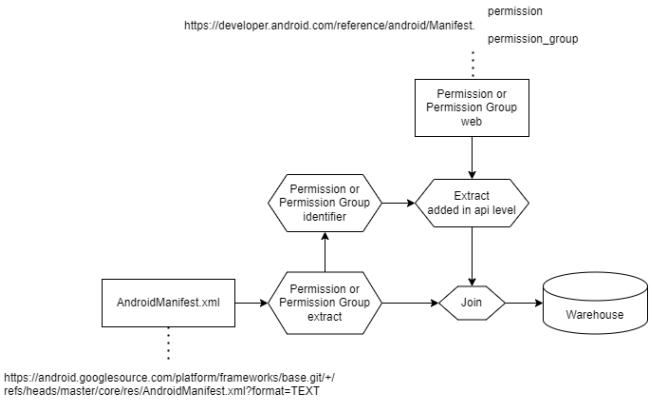

Figura 4. Flujo de datos AOSP.

a su API, mientras que en el caso de los estudios pasados se busca si estos contenían la aplicación. Si es así, se escala cada una de las puntuaciones obtenidas entre 0 y 10 y, se cargan al *warehouse*. Este escalado nos permite comparar las distintas puntuaciones obtenidas ya que se encuentran en un mismo rango de valores.

Para las puntuaciones otorgadas por los expertos, estas se reciben por medio de la API del *warehouse* ya escaladas entre 0 y 10, por lo que simplemente se almacenan junto al resto.

En caso de que ninguna fuente de validación tuviera información relativa a una aplicación, siempre se seguiría teniendo la puntuación obtenida por la aplicación de la métrica de App-PI [19].

III-D. La API de Acceso al Warehouse

La API sigue el estándar OpenAPI. Se facilita la interoperabilidad del *warehouse* con otros sistemas de terceros gracias a la documentación web interactiva que este estándar aporta. La funcionalidad de la API se encuentra segmentada según el tipo de usuario. Teniendo cada uno de estos sus respectivos puntos de entrada.

El punto de entrada de los usuarios generales es: https://apkfalcon.infor.uva.es:8080/. La Fig. 5 muestra la página web https://apkfalcon.infor.uva.es:8080/docs que contiene la documentación de toda la funcionalidad permitida para este tipo de usuario, siendo las operaciones más habituales:

1. **Descargar los metadatos de una aplicación.**
   Esta se realiza por medio de una petición de tipo GET a la URL: https://apkfalcon.infor.uva.es:8080/get/app/package con el nombre de paquete de la aplicación solicitada con el parámetro *package*.
   Por ejemplo, la Fig. 6 muestra la consultada realizada para obtener los metadatos de WhatsApp: https://apkfalcon.infor.uva.es:8080/get/app/package?package=com.whatsapp.
2. **Solicitar la carga de una aplicación.**
   Esta se realiza por medio de una petición de tipo POST a la URL: https://apkfalcon.infor.uva.es:8080/post/app/package teniendo el nombre de paquete de la aplicación solicitada en el cuerpo de la petición.

El punto de entrada de los usuarios expertos es: https://apkfalcon.infor.uva.es:8080/expert/. La documentación de toda la funcionalidad permitida para este tipo de usuario

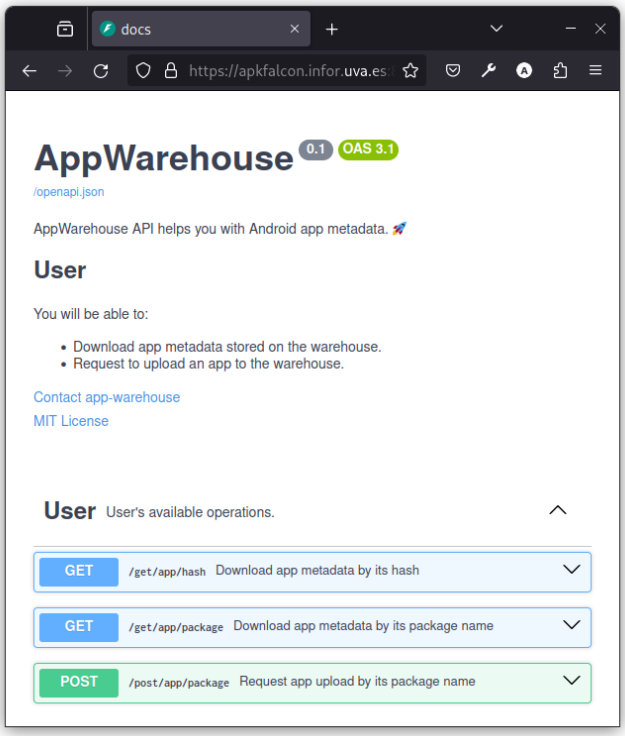

Figura 5. Documentación de la funcionalidad de los usuarios generales.

se encuentra en la página web: https://apkfalcon.infor.uva.es:8080/expert/docs, siendo la operación más habitual la subida de una puntuación de una aplicación.

Esta se realiza por medio de una petición de tipo POST a la URL: https://apkfalcon.infor.uva.es:8080/expert/post/score. Es necesario especificar en el cuerpo de la petición el *hash* SHA256 de la aplicación que se está puntuando y la puntuación escalada entre 0 y 10.

Se ha utilizado el formato JSON en la implementación de la API para el intercambio de información con los usuarios ya que es la notación más utilizada en los servicios web. Además, se ha diseñado para que pueda servir concurrentemente a un gran número de usuarios de forma segura, teniendo en cuenta alguno de los riesgos y vulnerabilidades del *top* 10 de OWASP API [24].

IV. Estado actual del repositorio

El *warehouse* almacena datos de un total de 9068 aplicaciones, 912 permisos del sistema, 16 grupos de permisos del sistema y 5 fuentes de información de intrusividad (ver tabla I)[14]. Además, se encuentra abierta la API de acceso al *warehouse* para que pueda ser utilizada por servicios de terceros e investigadores en el enlace https://apkfalcon.infor.uva.es:8080.

Las aplicaciones contenidas en el *warehouse* se encuentran altamente distribuidas entre distintas categorías. La Fig. 7 muestra la distribución de aplicaciones entre el top 10 de categorías con más *apps*. Estas aplicaciones representan un 51,15 % del contenido en el *warehouse*. Como se puede observar los contenidos del *warehouse* no se encuentran sesgados a

[14]A fecha 26/02/2024.

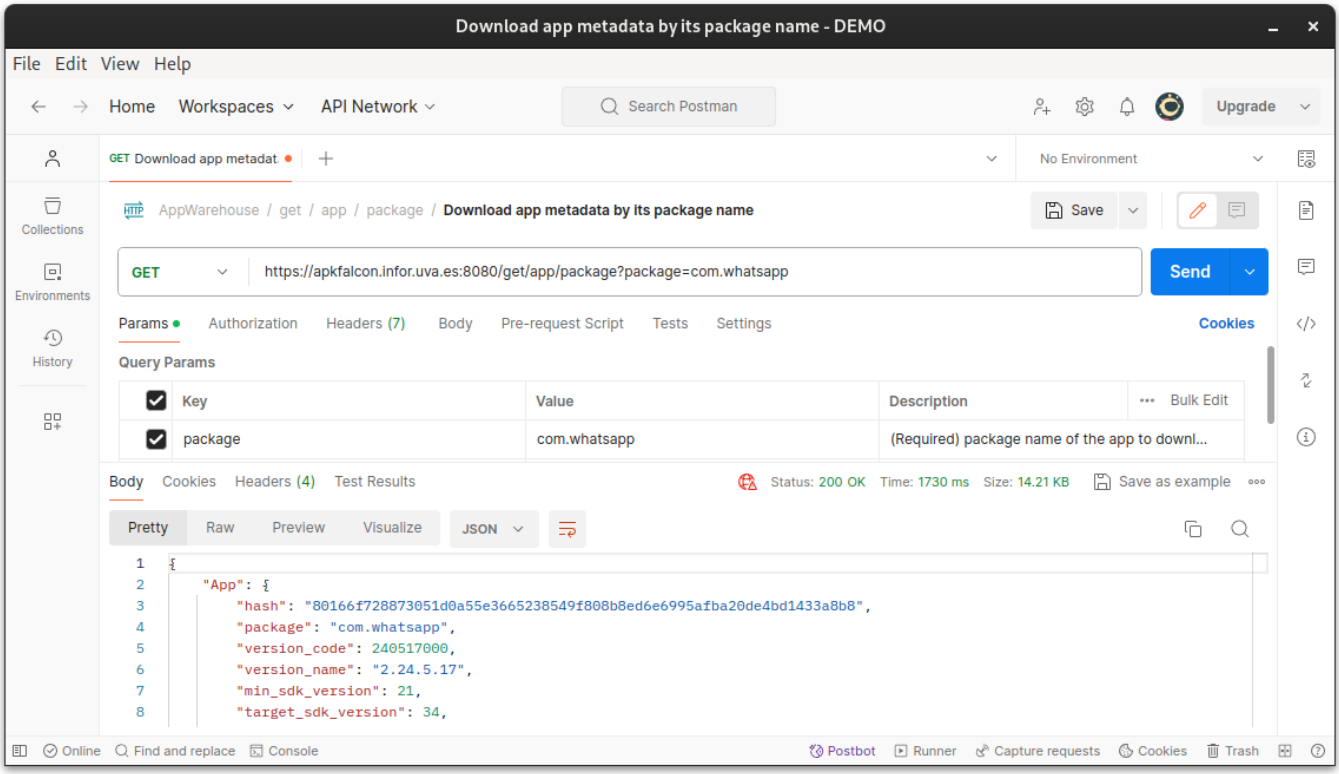

Figura 6. Solicitud de los metadatos de WhatsApp en Postman.

Aplicaciones	9068
Permisos de Android	912
Grupos de Permisos Android	16
Medidas de Intrusividad en la Privacidad	5

Tabla I
INDICADORES SOBRE EL CONTENIDO DEL REPOSITORIO.

Hardware	**Capacidad**
CPU	4 núcleos a 2 GHz
RAM	5 GB
Almacenamiento	50 GB

Tabla II
ESPECIFICACIONES TÉCNICAS DEL SERVIDOR.

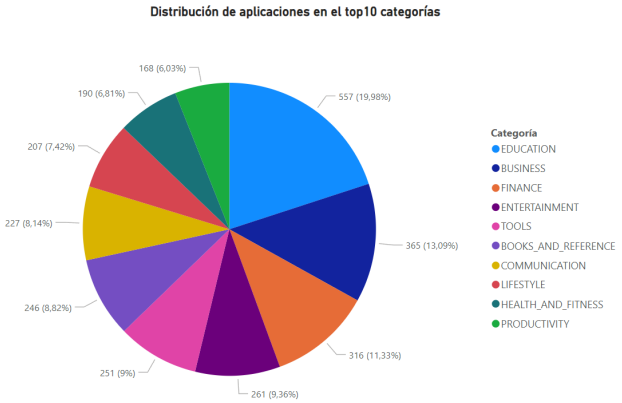

Figura 7. Distribución de aplicaciones en el top 10 categorías.

ninguna categoría en concreto. En el top 3 de categorías con más aplicaciones se encuentran las de educación, negocios y financieras; estos tipos de aplicaciones van dirigidas a un gran público y, por tanto, la intrusividad en la privacidad de sus usuarios afecta a un mayor número de estos.

Se ha monitorizado el tiempo que tardan algunas de las operaciones más importantes del repositorio: carga masiva de aplicaciones (8 *apps*/min), descarga de metadatos de una *app* almacenada (381 ms) y descarga de metadatos de una *app* no almacenada (42 ms). Los resultados que se muestran son muy prometedores, demuestran la viabilidad del proyecto en recursos de coste moderado (ver tabla II).

V. DISCUSIÓN

En esta sección se realiza un análisis retrospectivo de alguna de las decisiones tomadas en la sección III.

Después de sopesar varios tipos de diseños para la recogida y el almacenamiento de datos, se decide optar por una arquitectura de *Data Warehouse*. Si bien es cierto que con una arquitectura más general como *Data Lake* se podrían cubrir las necesidades del repositorio y, además, se podría dejar abierta la puerta a ampliaciones futuras de los tipos de información almacenados en el mismo, esta incrementaría la complejidad del sistema de manera innecesaria para el propósito planteado. Por tanto finalmente, debido a que la naturaleza de los metadatos que se almacenan es textual y estructurada, se decide utilizar la arquitectura que, cumpliendo con los requisitos de almacenamiento del proyecto, recopilar y almacenar metadatos de aplicaciones, es a su vez la más

fácil de entender y mantener, el *Data Warehouse*.

La principal ventaja de la solución planteada es el uso de un sistema gestor de bases de datos como MySQL ya que es poco costoso en términos de infraestructura, coste económico y curva de aprendizaje para su mantenimiento. Además, en un SGBD relacional es posible realizar consultas más complejas sobre la información almacenada en el *warehouse*. Por ejemplo, el cruce de información común entre las distintas aplicaciones o buscar todas las aplicaciones que utilizan un cierto grupo de permisos... Mientras que la solución general basada en *Data Lakes* tendría que hacer una búsqueda sobre el total de los datos almacenados, en la solución MySQL se aprovecha la combinación de la información almacenada (*join*) para obtener una respuesta sin la necesidad de la exploración de todo el conjunto de datos. Esto incrementa la velocidad de respuesta del sistema.

Como se comentó en la sección III, se ha tenido en cuenta la seguridad del repositorio desde el diseño. En este aspecto, se ha decidido utilizar una API de acceso en lugar de acceso directo a SQL. Esta decisión limita las posibilidades del repositorio a las que aporta la API. No obstante, es una decisión necesaria si queremos mantener un repositorio seguro con datos de calidad, ya que tener acceso directo a SQL permitiría introducir información sin validar como puntuaciones sin escalar, aplicaciones ficticias...

Aun así, todavía cabe la posibilidad de que el método de acceso al repositorio, la API, no sea seguro. Para intentar que sea lo más seguro posible se ha tenido en cuenta el *top* 10 de vulnerabilidades y riesgos de seguridad que hay en las APIs según la fundación OWASP [24]. Algunos de los riesgos que se mencionan en el *top* 10 se relacionan con: el uso no restringido de recursos o acceso no restringido a flujos sensibles, los cuales, como se ha visto en la sección III han sido tratados.

VI. Conclusiones

Pese a las dificultades inherentes al problema de integración de distintas fuentes de datos, se ha conseguido alcanzar el objetivo de poner a disposición de investigadores y servicios metadatos para analizar el impacto en la privacidad de las aplicaciones. Todo ello teniendo en cuenta la seguridad del repositorio, así como la calidad de los datos desde el diseño del mismo. Las ventajas de nuestra propuesta son: aportar información sobre aquello que los usuarios puede manejar (grupos de permisos) y proporcionar información sobre las valoraciones del impacto en la privacidad conseguidos por distintas métricas, lo cual puede ser útil a los investigadores para realizar comparaciones.

Gracias a que solo se almacenan metadatos sobre las aplicaciones, se ha conseguido almacenar toda la información de la tabla I en un espacio inferior a 50 MB. Esto nos asegura una buena escalabilidad y viabilidad del proyecto de cara al futuro, sin la necesidad de un incremento sustancial del coste del mismo.

Se espera un crecimiento en el uso del repositorio y sus contenidos por la comunidad investigadora, ya que se esta difundiendo en congresos, talleres, etc. Para que sea conocido y utilizado por terceros y permita arrojar luz y seguir entendiendo el problema de la intrusividad en la privacidad de las aplicaciones móviles. Un problema del cual los usuarios finales a pesar de ser los principales afectados, siguen tomando conciencia poco a poco, es necesario insistir en la formación y creación de herramientas que contribuyan a este fin.

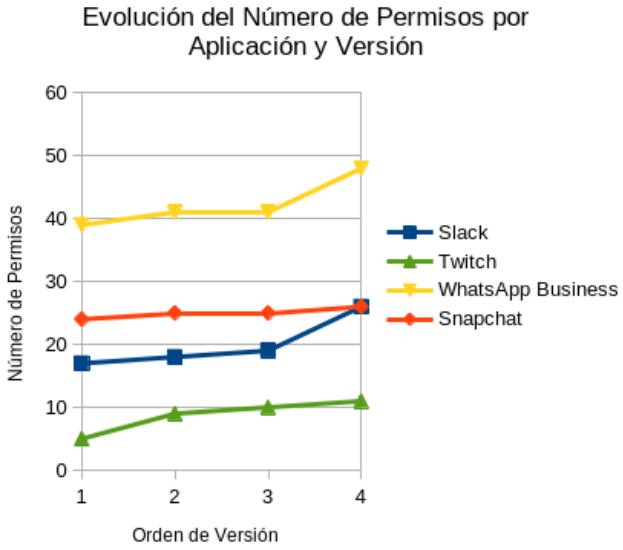

Figura 8. Gráfico de evolución del número de permisos solicitados.

[Se esta difundiendo el repositorio en congresos, talleres, etc para que sea conocido y se utilice por terceros]

VII. Trabajo Futuro

El trabajo que aquí se presenta permite múltiples líneas de trabajo futuro gracias a su diseño. Directamente relacionadas con el *warehouse* surgen posibilidades de ampliar y/o complementar su funcionalidad de distintos modos:

- Permitir la carga de aplicaciones al *warehouse* por medio de su archivo empaquetado (*.apk*). De este modo, se permitiría a investigadores y desarrolladores subir aplicaciones que no se encuentren publicadas en las fuentes de información mencionadas en la sección III-C.
- Permitir consultas avanzadas para investigadores. Por ejemplo, ver todas las aplicaciones que utilizan cierto permiso.
- Mejoras basadas en el *feedback* de los usuarios.
- Continuar con la monitorización del repositorio por medio de evaluaciones de rendimiento y auditorías surgidas a raíz de su apertura a la comunidad investigadora para garantizar que se cumplen los objetivos de diseño.
- Sistemas de medida y control de la calidad de los datos almacenados en el *warehouse*.

Por otro lado, esperamos poder hacer análisis acerca de la evolución del número de permisos que solicitan las aplicaciones a medida que aumente el número de usuarios. Estos estudios permiten ratificar otros ya existentes [25]. Si nos fijamos en algunas de las aplicaciones más utilizadas actualmente, como Slack, Twitch, WhatsApp Business y Snapchat, tienden a aumentar el número de permisos que solicitan con el paso de las versiones, tal como se muestra en la

Fig. 8. Uno de los permisos utilizado por la mayoría de estas aplicaciones que es más crítico para nuestra privacidad es "*android.permission.READ_PHONE_STATE*", el cual permite acceder al estado del teléfono (por ejemplo, conocer si se encuentra en una llamada telefónica). En nuestra opinión, esto representa un peligro potencial en la privacidad de sus usuarios. Este tipo de evoluciones justifica el interés en el estudio del impacto en la privacidad de los usuarios de las aplicaciones y, la utilidad de un repositorio que proporcione datos de calidad que se puedan utilizar por los investigadores a este respecto.

AGRADECIMIENTOS

Este trabajo se incluye en las actividades del proyecto estratégico de ciberseguridad "App-PI (*App Privacy Impact*): Un ecosistema para la evaluación del impacto de apps para dispositivos móviles sobre la privacidad y seguridad de sus usuarios", el cual se realiza al amparo de un convenio de colaboración entre la Universidad de Valladolid y la S.M.E. Instituto Nacional de Ciberseguridad de España M.P., S.A. para la promoción de proyectos estratégicos de ciberseguridad en España, en el marco de los fondos del Plan de Recuperación, Transformación y Resiliencia, financiados por la Unión Europea (Next Generation), el proyecto del Gobierno de España que traza la hoja de ruta para la modernización de la economía española, la recuperación del crecimiento económico y la creación de empleo, para la reconstrucción económica sólida, inclusiva y resiliente tras la crisis de la COVID19, y para responder a los retos de la próxima década.

REFERENCIAS

[1] España. (2018) Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales. [Online]. Available: https://www.boe.es/buscar/pdf/2018/BOE-A-2018-16673-consolidado.pdf

[2] P. G. Kelley, S. Consolvo, L. F. Cranor, J. Jung, N. Sadeh, and D. Wetherall, "A Conundrum of Permissions: Installing Applications on an Android Smartphone," in *Financial Cryptography and Data Security*, J. Blyth, S. Dietrich, and L. J. Camp, Eds., 2012, pp. 68–79.

[3] Rina, "A Comparative Analysis of mobile Operating Systems," in *International Journal of Computer Sciences and Engineering*, 2018, pp. 69–73.

[4] I. N. de Ciberseguridad. CONAN mobile. [Online]. Available: https://www.incibe.es/ciudadania/herramientas/conan-mobile

[5] M. contributors. (2017) Privacyscore. [Online]. Available: https://privacyscore.org/

[6] A. Pérez-Fuente, "Data Warehouse para el estudio de la privacidad de aplicaciones móviles," Trabajo fin de Grado, Universidad de Valladolid, 2023.

[7] J. Crespo Guerrero, "APKFalcon: Servicio de usuarios para la evaluación y comprensión del impacto sobre la privacidad de aplicaciones móviles," Trabajo fin de Grado, Universidad de Valladolid, 2023.

[8] F.-X. Geiger and I. Malavolta, "Datasets of Android Applications: a Literature Review," *ArXiv*, vol. abs/1809.10069, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:52845379

[9] K. Allix, T. F. Bissyandé, J. Klein, and Y. Le Traon, "AndroZoo: Collecting Millions of Android Apps for the Research Community," in *Proceedings of the 13th International Conference on Mining Software Repositories*, 2016, pp. 468–471.

[10] G. Meng, Y. Xue, J. K. Siow, T. Su, A. Narayanan, and Y. Liu, "AndroVault: Constructing Knowledge Graph from Millions of Android Apps for Automated Analysis," *ArXiv*, vol. abs/1711.07451, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:1736293

[11] Telefónica. (2018) Tacyt. [Online]. Available: https://www.movistar.cl/web/corporaciones/tacyt

[12] D. E. Krutz, M. Mirakhorli, S. A. Malachowsky, A. Ruiz, J. Peterson, A. Filipski, and J. Smith, "A Dataset of Open-Source Android Applications," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, 2015, pp. 522–525.

[13] M. Hatamian, N. Momen, L. Fritsch, and K. Rannenberg, "A Multilateral Privacy Impact Analysis Method for Android Apps," in *Privacy Technologies and Policy*, M. Naldi, G. F. Italiano, K. Rannenberg, M. Medina, and A. Bourka, Eds., 2019, pp. 87–106.

[14] S. Barth, M. de Jong, M. Junger, P. H. Hartel, and J. C. Roppelt, "Putting the privacy paradox to the test: Online privacy and security behaviors among users with technical knowledge, privacy awareness, and financial resources," in *Telematics Informatics*, vol. 41, 2019, pp. 55–69.

[15] H. Benita. (2019) Preventing SQL Injection Attacks With Python. [Online]. Available: https://realpython.com/prevent-python-sql-injection/

[16] Android Developers. (2023) Archivo de manifiesto de la app: permission. [Online]. Available: https://developer.android.com/reference/android/Manifest.permission

[17] ——. (2023) Archivo de manifiesto de la app: permission group. [Online]. Available: https://developer.android.com/reference/android/Manifest.permission_group

[18] H. Roy. (2012) Terms of Service Didn't Read. [Online]. Available: https://tosdr.org/

[19] A. Aparicio, M. Martínez González, and V. Cardeñoso, "Métrica basada en grupos de permisos para entender el impacto de las aplicaciones Android sobre la privacidad," in *2022 17th Iberian Conference on Information Systems and Technologies (CISTI)*, 2022, pp. 1–5.

[20] D. Hardt, "The OAuth 2.0 Authorization Framework," RFC 6749, 2012. [Online]. Available: https://tools.ietf.org/html/rfc6749

[21] O. I. (OAI). (2022) OpenAPI. [Online]. Available: https://www.openapis.org/

[22] M. Upadhayay, A. Sharma, G. Garg, and A. Arora, "RPNDroid: Android Malware Detection using Ranked Permissions and Network Traffic," in *2021 Fifth World Conference on Smart Trends in Systems Security and Sustainability (WorldS4)*, 2021, pp. 19–24.

[23] Android Developers. (2021) Declara el uso de datos de tu app. [Online]. Available: https://developer.android.com/privacy-and-security/declare-data-use?hl=es-419

[24] T. O. Foundation. (2023) OWASP API Security Project. [Online]. Available: https://owasp.org/www-project-api-security/

[25] P. Calciati, "Understanding the Evolution of Android Applications," Tésis Doctoral, Universidad Politécnica de Madrid, 2019. [Online]. Available: https://oa.upm.es/57886/1/PAOLO_CALCIATI.pdf