UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

MÁSTER EN CIBERSEGURIDAD Y SEGURIDAD DE LA INFORMACIÓN

TRABAJO FIN DE MÁSTER

METRICS: Sistema de análisis de seguridad de aplicaciones Android

Oscar Martín

Sep, 2019

UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

Departamento de Sistemas Informáticos

TRABAJO FIN DE MÁSTER

METRICS: Sistema de análisis de seguridad de aplicaciones Android

Autor: Oscar Martín

Tutor: Pablo González

Sep, 2019

Resumen

Se presenta METRICS un sistema para estudiar la privacidad que tienen las aplicaciones de Android.

Para ello se ha creado una métrica en base a los permisos que se conceden en la instalación de una aplicación de Android.

Y conseguir de una manera sencilla un valor que nos ayude a medir como gestionan la privacidad las distintas aplicaciones que tenemos normalmente instaladas en nuestros dispositivos Android.

Ayudando a entender mejor el uso que hacen de los permisos concedidos al instalar una aplicación móvil y valorar si son adecuados al uso que podemos realizar de la aplicación en su uso cotidiano o poder valorar si instalamos o no esa aplicación, una vez que podemos conocer la implicación sobre la privacidad de nuestra información, que puede verse afectada con la aplicación instalada.

Siendo una mejora que podemos añadir en las herramientas que existen sobre el análisis de aplicaciones móviles que no contemplan la creación de una métrica sobre la privacidad.

Dedicatorias

A Alicia, Alberto y Beatriz por su paciencia en el tiempo que he estado con este master y trabajo final.

.

ÍNDICE

ÍNDICE DE FIGURAS

CAPÍTULO 1. Introducción

METRICS es una aplicación que mide la privacidad que tiene una aplicación móvil para el sistema operativo de Android mediante una métrica numérica de fácil comprensión por parte de un usuario corriente.

Imagen 1.METRICS

Que viene a completar otro tipo de proyectos que existen en la actualidad de análisis de aplicaciones móviles, que ayuden a un usuario corriente a poder medir como cuidan la privacidad las aplicaciones que puede tener instaladas en su dispositivo móvil Android.

1.1 MOTIVACIÓN

La elección de este proyecto tiene que ver con lo complicado que resulta para un usuario corriente el poder evaluar de una manera sencilla el tratamiento de la privacidad por parte de las aplicaciones móviles que se instalan para el uso cotidiano y sobre todo por lo complicado que lo tienen los padres a la hora de determinar de manera sencilla si instalan o no una aplicación móvil a sus hijos. Y poder explicar cómo afecta a la privacidad esos permisos que concedemos que mayoritariamente no solemos leer ... (o sí).

1.2 OBJETIVOS

Conseguir una métrica sencilla que los usuarios puedan utilizar para medir que grado de privacidad tiene una aplicación (App) descargada desde la Google Play.

Objetivo secundario: Mejorar mis conocimientos en Python para trabajar con aplicaciones de Android para su análisis y ver si se pueden sacar patrones de las aplicaciones a partir de la métrica. Para este punto tengo que analizar un alto numero de aplicaciones. Además, quiero aprovechar este trabajo para poder aprender a programar en Python y conocer el framework Flask para el desarrollo de aplicaciones web en Python, dado que mis conocimientos de este lenguaje eran prácticamente nulos al comienzo de este master y es una buena manera de aprenderlo y poder usarlo con herramientas de ciberseguridad.

Y por último poder emplearlo como ejemplo gráfico cuando se habla de privacidad en las charlas que espero retomar al terminar este master como cibercooperante.

1.3 ESTRUCTURA DE LA MEMORIA

A continuación, en los siguientes capítulos voy ir desarrollando como he llegado a la definición de la métrica como al desarrollo del entorno web.

Empezando por la definición de privacidad y viendo como analizar los permisos que tienen las aplicaciones Android y como crear una métrica para medir la implicación que tienen.

A continuación, se explica el entorno elegido para realizar la aplicación y se explica como se ha conseguido ajustar la métrica que se ha empleado finalmente.

Por último, se explica como se analiza una aplicación móvil y se presentan los datos de las aplicaciones analizadas para este trabajo exponiendo finalmente las conclusiones que se han obtenido y que propuestas pueden avanzar en la mejora de como poder utilizar esta métrica.

CAPÍTULO 2. La privacidad

Según la RAE la privacidad
es. Ámbito de la vida privada que se tiene derecho a proteger de cualquier intromisión.

Según Wikipedia.
La privacidad en Internet se refiere al control de la información que posee un determinado usuario que se conecta a la red, interactuando con diversos servicios en línea en los que intercambia datos durante la navegación. Implica el derecho o el mandato a la privacidad personal con respecto al almacenamiento, la reutilización, la provisión a terceros y la exhibición de información a través de Internet.

Según OSI
Todo lo que hacemos en Internet deja un rastro y nuestra información personal es muy valiosa, no solo para nosotros, también para otras personas, empresas e incluso para los ciberdelincuentes

Si cuando se ofrece una aplicación gratis el producto eres tú, frase que se repite en ámbitos de concienciación sobre la seguridad y privacidad en internet como podemos desde el punto de vista de un usuario normal saber si las aplicaciones que usamos habitualmente o las que necesitamos instalar para comunicarnos que grado de protección de la privacidad del usuario tienen.

Es lo que vamos a desarrollar en los siguientes capítulos.

CAPÍTULO 3. El análisis de las aplicaciones.

Para poder llevar a cabo este trabajo he tomado como referencia el trabajo realizado del framework MobSF ya que este framework ya realiza el análisis de las aplicaciones móviles, pero lo que no tiene es una clasificación de la privacidad de la aplicación que se analiza, por lo que este trabajo puede completar ese hueco.

Para conseguir el objetivo marcado en este trabajo y poder realizar el análisis de los permisos de las aplicaciones voy a utilizar la aplicación java Apktools, que nos permite descomprimir los archivos .apk que contienen la aplicación móvil y así poder acceder al contenido del archivo AndroidManifest.xml que es donde encontramos los permisos que se conceden a las aplicaciones móviles.

Lo principal es poder leer los permisos que se conceden a las aplicaciones, en los siguientes enlaces podemos encontrar información sobre los permisos de las aplicaciones y como están clasificados en función del nivel. Tiene 4 niveles: normal, dangerous, signature, signature system. Los permisos que una aplicación móvil necesite para instalarse y poder ejecutarse se tienen que poner de manera explícita en el archivo AdroidManifest.xml

https://play.google.com/intl/es/about/privacy-security-deception/permissions/

https://developer.android.com/guide/topics/permissions/overview#normal-dangerous

A parte de la información que se obtiene de la aplicación del fichero AdroidManifest, también se recoge cual es la categoría que tiene asignada en la tienda de Google Play la aplicación analizada, esta información se recoge directamente de la página web de Google Play si está activa dicha aplicación móvil en la tienda cuando se realice el análisis. Siendo un campo de clasificación que podríamos añadir a la métrica en un futuro junto a más información que se pudiera obtener de la tienda de Google Play como la edad a la que se

dirige la aplicación o el número de descargas, aunque esta parte no la he abordado en este trabajo y solo he mostrado que podemos añadir más características que puedan ayudar a una mejora de la métrica.

CAPÍTULO 4. La métrica

Realizar una métrica para poder medir la privacidad de las aplicaciones móviles no es una tarea sencilla, aunque lo parezca, lo que me ha llevado a enforcar el problema de realizar una métrica como el cálculo en valor negativo del uso de los permisos que se le conceden a una aplicación móvil para Android (recomendación realizada por Pablo), pero dejándola abierta a los ajustes que más nos puedan interesar para un ajuste más fino en la medición de los permisos concedidos de las aplicaciones.

El conseguir una única métrica requiere un análisis mucho más amplio en tiempo del que he empleado en este trabajo, para conseguir ponderar todas las posibles variables que pueden intervenir en la medición. Y analizar una gran cantidad de aplicaciones móviles.

Así que para poder realizarla con la suficiente validez he procedido de la siguiente manera.

Mostremos un ejemplo de la problemática. Se puede medir igual la privacidad desde el punto de vistas de un adulto que el de un menor o que la aplicación móvil se utilice para un uso profesional frente a un uso personal. La respuesta que doy es que no. Teniendo en cuenta que en la actualidad existen más de 200 permisos diferentes [1] en el universo de Android, con cuatro niveles de catalogación, solo con añadirle la característica de la edad y el uso (profesional, privado) las combinaciones para determinar una única métrica que nos evalúe la privacidad de la aplicación móvil requeriría un estudio de un número muy elevado de App's (aplicaciones móviles) y para este trabajo he analizado unas 300 que considero que es un número suficiente para este trabajo y que me permitiera defender la

[1] Manifest.permission | Android Developers. https://developer.android.com/reference/android/Manifest.permission. Accedido 12 de septiembre de 2019.

métrica elegida, pero que tendría que ser mucho más elevado para conseguir una mejor métrica.

Así que voy a utilizar dos valores llamados metrica1 y métrica2 cuya suma de los de sus valores en los permisos concedidos sea el valor que nos indique la privacidad de la App.

```
METRICA =∑permiso( metrica1 + metrica2)
```

El primer valor metrica1 se centra en el nivel del permiso asignado en Android, de los cuatro que existe, y el segundo valor metrica2 analiza el nombre del permiso asignado, de esta manera y utilizando el segundo valor podremos ajustar o parametrizar la métrica en función del uso que se haga y para el perfil al que se quiera dirigir (adulto, niños) para poder afinar el resultado de manera más eficiente y contemplar incluso los permisos personalizados que se pueden definir en la aplicaciones móviles.

Para calcular el primer valor (metrica1) voy a basarme en la clasificación que hace Android del nivel de un privilegio [2] y asignarle un valor a cada uno de los niveles como sigue:

- Nivel normal, metrica1 es 0
- Nivel dangerous, metrica1 es -1
- Nivel signature, metrica2 es -2
- Nivel signatureOrSystem, metrica1 es -3

De esta forma tenemos una primera aproximación de la métrica basada en los permisos, pero le hemos modificado unas excepciones para que nos ayude mejor en el cálculo de la privacidad y hemos modificado los siguientes permisos a valor 0 por considerar que no afectan a la privacidad de App. Cuando más aplicaciones se analicen se puede ampliar la lista.

```
excepciones =
['WAKE_LOCK','SYSTEM_ALERT_WINDOW','C2D_MESSAGE','CLEAR_APP_CAC
HE','SET_ALWAYS_FINISH','SET_ANIMATION_SCALE','SET_DEBUG_APP','
SET_PROCESS_LIMIT','SET_TIME_ZONE','SIGNAL_PERSISTENT_PROCESSES
','SUBSCRIBED_FEEDS_WRITE']
```

Estos permisos que he detectado que tienen algunas App que tienen un nivel distinto al normal los considero como normales a efecto de ponderación y les asignamos un valor de 0. Aquí tenemos la primera manera de poder afinar nuestra métrica utilizando el primer

[2] https://developer.android.com/guide/topics/permissions/overview?hl=es-419#normal-dangerous

valor de metrica1, podemos añadir o quitar en función de cómo queramos afinar más o menos nuestra métrica. Aunque según los resultados la mayoría de las App's consiguen valores negativos, sí que podemos hacer un ajuste más fino.

Y esta fue la primera aproximación que se me ocurrió, pero tras empezar a probarla me di cuenta que un mismo permiso con nivel de 'dangerous' no reflejaba el nivel de privacidad adecuado. Y con solo el valor de metrica1 no conseguía el resultado buscado.

Y es cuando decidí añadir el valor de metrica2 para ayudarme a afinar la métrica. Pongamos por ejemplo el permiso de INTERNET que se le concede a una App para que pueda acceder a Internet, es de nivel 'dangerous' por lo que tendría un valor de metrica1 igual a -1. Pero si lo comparo con el permiso de CAMERA que también tiene el mismo nivel (dangerous) y es el que permite acceder a la cámara del dispositivo Android, ¿los tengo que valorar de igual manera? Pues considero que no, que el permiso de INTERNET debería tener mayor peso, principalmente porque considero que cualquier permiso que implique una comunicación conlleva la divulgación de información que puede afectar a la privacidad. En este ejemplo el permiso CAMERA sí que afecta a la privacidad por el hecho de poder hacer una foto, pero si se quedará en el dispositivo no tendría la misma importación de si además existe el permiso de INTERNET que también afecta a la privacidad pero que puede permitir sacar la foto o cualquier otra información del dispositivo. Para resolver esta situación y que resultara sencillo ajustarse añadí el valor de metrica2 que se basa en la asignación del valor -1 a cualquier permiso que contenga alguna de las palabras de dos categorías definidas en el nombre del permiso.

Las dos categorías que he definido son:

- Comunicación: implica que existe comunicación.
- Personal: implica acceso a información considerada personal.

Y la lista de palabras que he elegido para determinarlas son basadas en las palabras utilizas en los permisos existentes en Android:

```
    comunicacion = ['PHONE', 'SMS','MMS','NETWORK', 'BLUETOOTH',
'INTERNET', 'WIFI', 'NFC']
        personal = ['CALENDAR', 'NUMBER', 'CALL', 'LOCATION',
'EXTERNAL_STORAGE', 'LOG','CONTACTS']
```

Cuando alguna de estas palabras se encuentra en el nombre del permiso concedido a la App se le suma -1 al valor metrica2.

De esta forma el permiso de INTERNET tendría una métrica de -2 y el de CAMERA de -1, o por ejemplo un permiso como ACCESS_WIFI_STATE que tiene un nivel normal tendría un valor de métrica de -1.

De esta manera puedo ajustar mejor la métrica y permite que tras el mayor análisis que se hagan de las App's se pueda afinar modificando bien en las excepciones del cálculo del valor metrica1 como añadiendo palabras para el cálculo del valor de metrica2.

Si se analizase un número elevado de App's se podría dejar definido los valores para metrica1 y emplear metrica2 para añadir por ejemplo la característica de seguridad de la App además de la privacidad, algo que ya no he contemplado en este trabajo como describiré más adelante.

Como ejemplo uno de los juegos de LEGO obtendría una métrica de -2

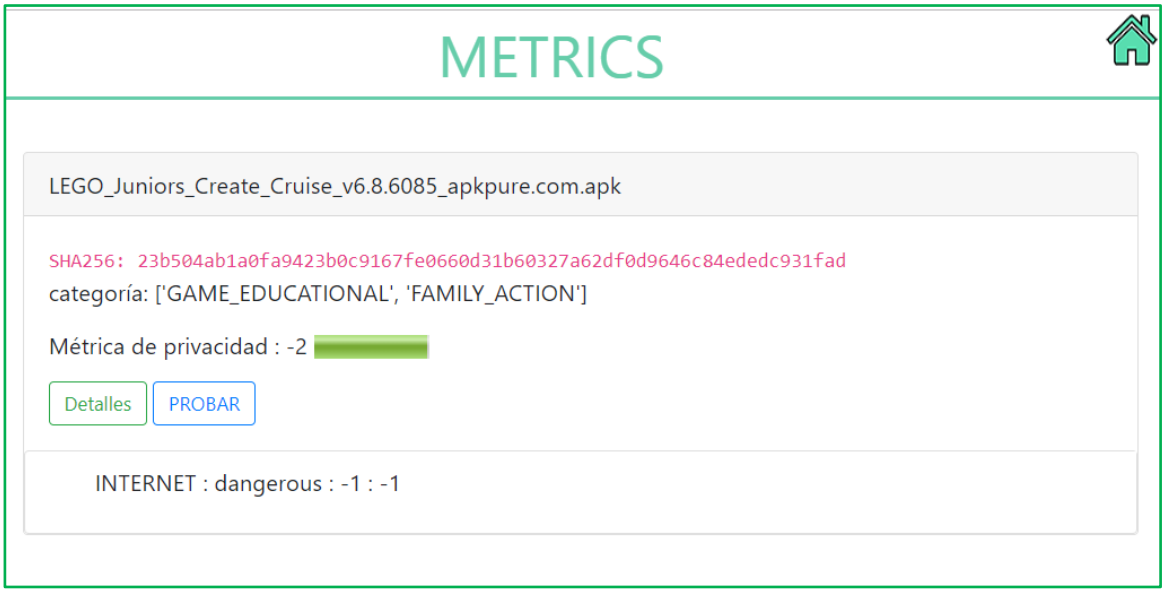


Imagen 2. Métrica App Lego

Imagen 3. Juego App Lego

Solo concede el permiso de Internet, que para ser un juego orientado a niños de hasta 8 años la métrica la considero adecuada. Decir que sorprende la diferencia de valores que se obtienen al analizar juegos de esta franja de edad.

CAPÍTULO 5. El Entorno

El entorno elegido para desarrollar la web y los programas que nos permitan calcular el cálculo de la métrica es utilizando el microframework FLASK escrito en Python.

Para el desarrollo de este trabajo he utilizado un entorno virtual de Python en su versión 3, con el microframework de FLASK y apoyándonos con una libreta de jupyter notebook que me servirá como medio de explicación del trabajo realizado y el ajuste inicial de la métrica.

También hago uso de la herramienta apktool (escrita en java) para poder descomprimir los archivos apk y poder leer el archivo de permisos de cada aplicación móvil (AndroidManifest.xml) que es lo que realmente necesitamos para trabajar con la métrica.

Por lo tanto, creo dos entornos virtuales de Python, uno para la aplicación Flask que nos permite evaluar las App's y otro para poder utilizar la libreta de jupyter-notebook con pandas para el manejo de las tablas de definición de los valores asignados a los permisos y que ayuden al análisis de los permisos en las aplicaciones.

La libreta de jupyter-notebook solo la utilizo como ayuda para analizar los resultados y poder ajustar de manera más sencilla la métrica, siendo únicamente necesario para poder utilizar el proyecto utilizar el entorno de Flask y tener habilitado java para poder ejecutar la herramienta de Apktools

En el Anexo B se incluyen los requerimientos de ambos entornos.

CAPÍTULO 6. Preparando la Métrica

Una vez que ya sabemos cómo va a ser la métrica la vamos a preparar para poder utilizarla en nuestro proyecto.

Para ello he utilizado una libreta de Jupyter Notebook para ayudarme a configurarla y así luego poder emplearla para el análisis de la misma y conseguir afinarla como he ido comentando anteriormente.

Lo primero que tenemos que tener es la lista de los permisos de Android con el nivel que tienen asignado y para ello hago uso del proyecto MobSF (Mobile-Security-Framework-MobSF) [3]

Imagen 4. MobSF

Este framework es un todo en uno para análisis de aplicaciones móviles donde ya tiene implementado todo lo que necesitaría para poder aplicar la métrica, pero lo que yo voy a utilizar es la tabla de permisos que ya tienen definida que a su vez la obtiene del proyecto Androguard[4], aunque solo voy a utilizar los permisos individuales. Y así poder modificarla para contemplar los dos valores de metrica1 y metrica2.

[3] https://github.com/MobSF/Mobile-Security-Framework-MobSF
[4] https://androguard.readthedocs.io/en/latest/intro/installation.html

Para poder ajustarla utilizo la librería Pandas de Python para obtener una tabla como esta.

	PERMISO	LEVEL	LITERAL	DESCRIPTION
0	ACCEPT_HANDOVER	dangerous		Allows a calling app to continue a call which ...
1	ACCESS_CACHE_FILESYSTEM	signatureOrSystem	access the cache file system	Allows an application to read and write the ca...
2	ACCESS_CHECKIN_PROPERTIES	signatureOrSystem	access check-in properties	Allows read/write access to properties uploade...
3	ACCESS_COARSE_LOCATION	dangerous	coarse (network-based) location	Access coarse location sources, such as the mo...
4	ACCESS_FINE_LOCATION	dangerous	fine (GPS) location	Access fine location sources, such as the Glob...
5	ACCESS_LOCATION_EXTRA_COMMANDS	normal	access extra location provider commands	Access extra location provider commands. Malic...
6	ACCESS_MOCK_LOCATION	dangerous	mock location sources for testing	Create mock location sources for testing. Mali...
7	ACCESS_NETWORK_STATE	normal	view network status	Allows an application to view the status of al...
8	ACCESS_NOTIFICATION_POLICY	normal		Marker permission for applications that wish t...
9	ACCESS_SURFACE_FLINGER	signature	access SurfaceFlinger	Allows application to use SurfaceFlinger low-l...

Imagen 5. Tabla de permisos Android

Donde partimos de:

```
Permisos totales evaluados : 198
Permisos peligrosos : 69
Permisos normales : 45
Permisos signature : 58
Permisos signatureOrSystem : 26
```

Imagen 6. Número de permisos iniciales

Con cuatro niveles: dangerous, signatureOrSystem, normal, signatura

Aplicamos las excepciones que hemos comentado para el valor de la métrica1 y creamos la columna con su valor.

	PERMISO	LEVEL	LITERAL	DESCRIPTION	METRICA
58	C2D_MESSAGE	signature	Allows cloud to device messaging	Allows the application to receive push notific...	0
71	CLEAR_APP_CACHE	dangerous	delete all application cache data	Allows an application to free phone storage by...	0
155	SET_ALWAYS_FINISH	dangerous	make all background applications close	Allows an application to control whether activ...	0
156	SET_ANIMATION_SCALE	dangerous	modify global animation speed	Allows an application to change the global ani...	0
157	SET_DEBUG_APP	dangerous	enable application debugging	Allows an application to turn on debugging for...	0
160	SET_PROCESS_LIMIT	dangerous	limit number of running processes	Allows an application to control the maximum n...	0
162	SET_TIME_ZONE	dangerous	set time zone	Allows an application to change the phone's ti...	0
167	SIGNAL_PERSISTENT_PROCESSES	dangerous	send Linux signals to applications	Allows application to request that the supplie...	0
172	SUBSCRIBED_FEEDS_WRITE	dangerous	write subscribed feeds	Allows an application to modify your currently...	0
173	SYSTEM_ALERT_WINDOW	dangerous	display system-level alerts	Allows an application to show system-alert win...	0
182	WAKE_LOCK	dangerous	prevent phone from sleeping	Allows an application to prevent the phone fro...	0

Imagen 7. Excepciones métrica1

Y por último calculamos los valores para metrica2 según las categorías de comunicación y personal que hemos definido en la métrica, creando la columna con los valores de mtetrica2.

PERMISO	LEVEL	LITERAL	DESCRIPTION	METRICA	METRICA2
ACCEPT_HANDOVER	dangerous		Allows a calling app to continue a call which ...	-1	0
ACCESS_CACHE_FILESYSTEM	signatureOrSystem	access the cache file system	Allows an application to read and write the ca...	-3	0
ACCESS_CHECKIN_PROPERTIES	signatureOrSystem	access check-in properties	Allows read/write access to properties uploade...	-3	0
ACCESS_COARSE_LOCATION	dangerous	coarse (network-based) location	Access coarse location sources, such as the mo...	-1	-1
ACCESS_FINE_LOCATION	dangerous	fine (GPS) location	Access fine location sources, such as the Glob...	-1	-1

Imagen 8.creamos la columna metrica2

Ya tenemos calculada la tabla de permisos que vamos a utilizar para el cálculo de la métrica que guardamos en formato json para emplearla en la aplicación web de nuestro proyecto (metrica.json).

La libreta de Jupyter Notebook se incluye en la instalación de la aplicación para que se utilice para realizar un ajuste diferente si se quiere y resulte más sencillo manejarla.

CAPÍTULO 7. El análisis de la App

Para realizar el análisis de la App (aplicación móvil Android) he realizado una web muy sencilla que nos permita subir la App, su fichero .apk que podemos localizar de varios modos[5]. Una vez que ya la tenemos subida se realiza la descompresión del archivo .apk utilizando la herramienta Apktools (programa en java) y se realiza la lectura del archivo AndroidManifest.xml donde está definidos los permisos que va a solicitar la aplicación al usuario para poder instalarse en el dispositivo para su posterior uso.

Una vez tenemos los permisos que necesita los comprobamos en la tabla de permisos de la métrica conseguida en el capítulo anterior para realizar la suma de los valores de los permisos solicitados.

En el video de presentación de este trabajo se describe paso a paso su utilización para evaluar las App.

Una vez analizada la App guardamos los resultados para un posterior análisis que nos ayude a mejorar la definición de la métrica.

Como para poder obtener el archivo AndroidManifest.xml hay que descomprimir la App se crea para cada App un directorio cuyo nombre es el hash sha256 del fichero .apk y también se guarda en la carpeta "app" el archivo .apk que hemos subido para analizar.

Los datos que se guardan en el archivo app_metricas.json (aplicaciones analizadas) para cada aplicación son:

- Métrica

[5] https://apkpure.com/es/; https://apps.evozi.com/apk-downloader ; de la copia de seguridad del dispositivo.

- Packagename, nombre del paquete
- Categoría, categoría de la apk en la tienda de Google Play
- Fichero, nombre del fichero de la apk
- Hash, has sha256 del fichero de la apk
- Versión, versión de Android de la apk
- Version_name, nombre de la versión de Android
- Fecha, fecha del análisis
- Perm, los permisos de la apk

Aunque en la web solo mostramos el nombre del archivo .apk, su hash, las categorías a la que pertenece la App en la tienda de Google Play, este campo puede aparecer vacío por no encontrarse la App ya en la tienda de Google Play o ser de otro market, la métrica con un indicador de color que lo tengo establecido en color verde de 0 a -3, amarillo de -3 a -15 y rojo más de -15 definido en el archivo template/app_permiso.html.

Respecto a los colores el fijar en -3 el límite para el color verde viene por que las App's con esta puntuación suele ser porque suelen tener el permiso de INTERNET con valor -2 y luego algún otro permiso con puntuación en -1 que salvo en el caso de que sea el permiso de CAMERA con valor -1 donde podría ser confuso esa elección de color y que no me he encontrado ningún caso decidí dejarlo así fijado, pero que esta excepción se puede solventar dando al permiso de CAMERA valor -2 bien usando las excepciones del valor metrica1 bien añadiendo la palabra CAMERA en la lista de "personal" para el cálculo del valor metrica2. De ahí que podamos afinar nuestra métrica según vayamos analizando cuanto mayor número de aplicaciones mejor.

La inclusión de añadir las categorías de clasificación de la App de la tienda de Google Play fue para poder tener una variable más a la hora de ajustar mejor la métrica de privacidad, por no ser lo mismo que una App este en la categoría de juegos, familia, que en productividad o herramientas. Pero que al final no añadí a la métrica por no disponer de análisis suficientes. Pero quería mostrar que podemos enriquecer la métrica con características ajenas al propio fichero .apk.

Y sí que nos permite posteriormente al analizar los resultados de las aplicaciones analizadas el ver cómo podemos añadir esa característica en la métrica.

CAPÍTULO 8. Aplicaciones analizadas

Para poder realizar el ajuste de la métrica como he expuesto anteriormente realice varias pruebas y analicé diferentes aplicaciones.

Utilice un canal de Telegram (https://t.me/aapks) para facilitarme la descarga. que contenía algunas más y me descargue unas 300 aplicaciones de las que luego solo analice 255, la métrica resultante está en el documento (metrica_255_apls.xlsx), lo que me sirvió para ver que sí que existen aplicaciones con un valor de 0 como la aplicación:

Imagen 9. App con métrica 0

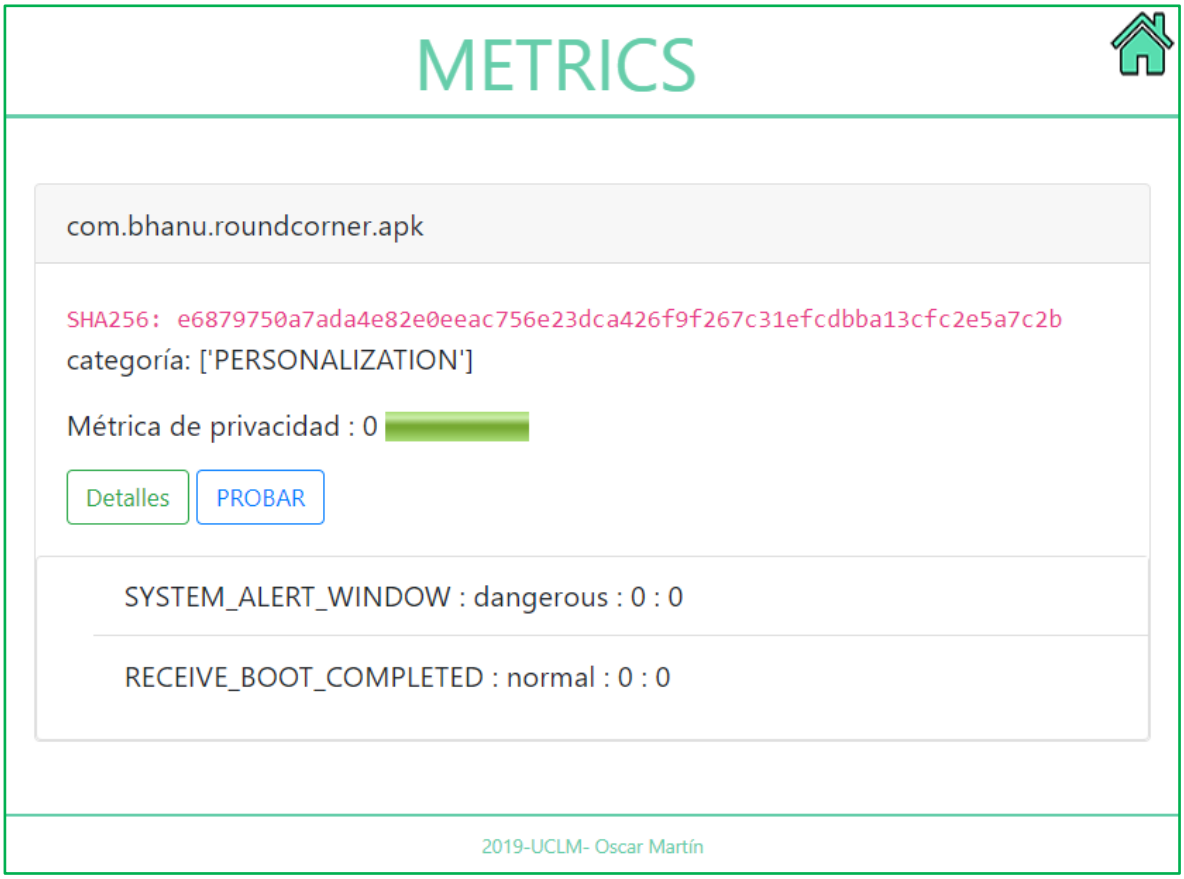


Imagen 10. Métrica de una App con valor 0

Y otras como la siguiente que llegan a tener un valor de -166 aunque estas ya no están en la tienda de Google (y no tenemos la categoría asociada)

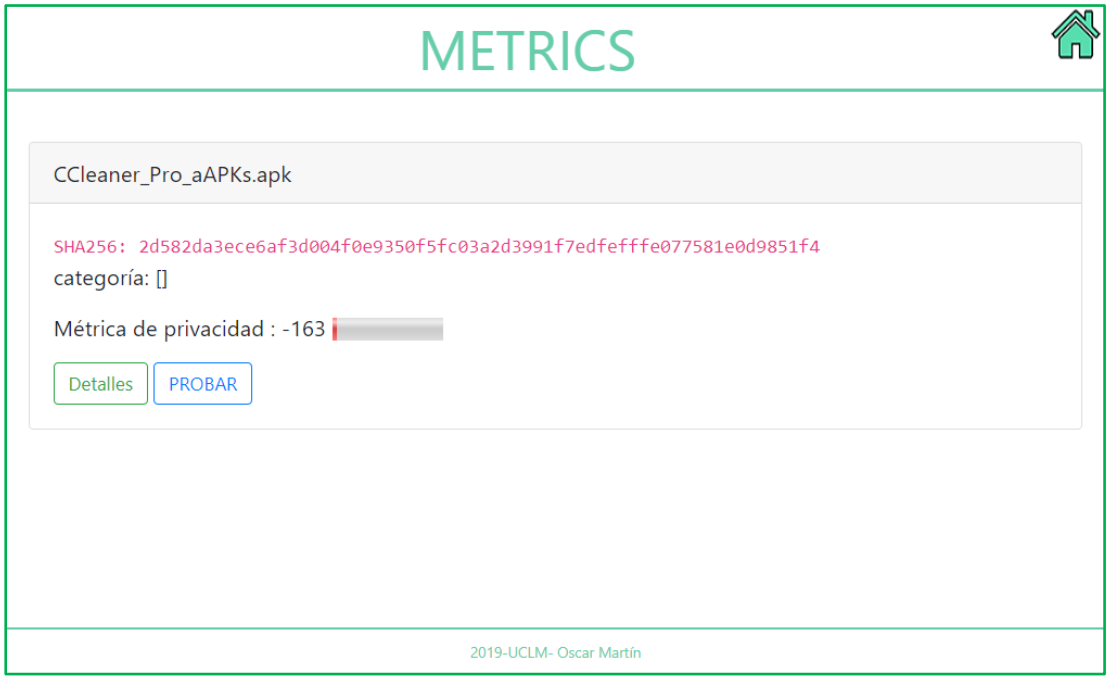


Imagen 11. App con una métrica de -163

También encontramos aplicaciones con valores de -123 que sí que encontramos en la tienda de Google.

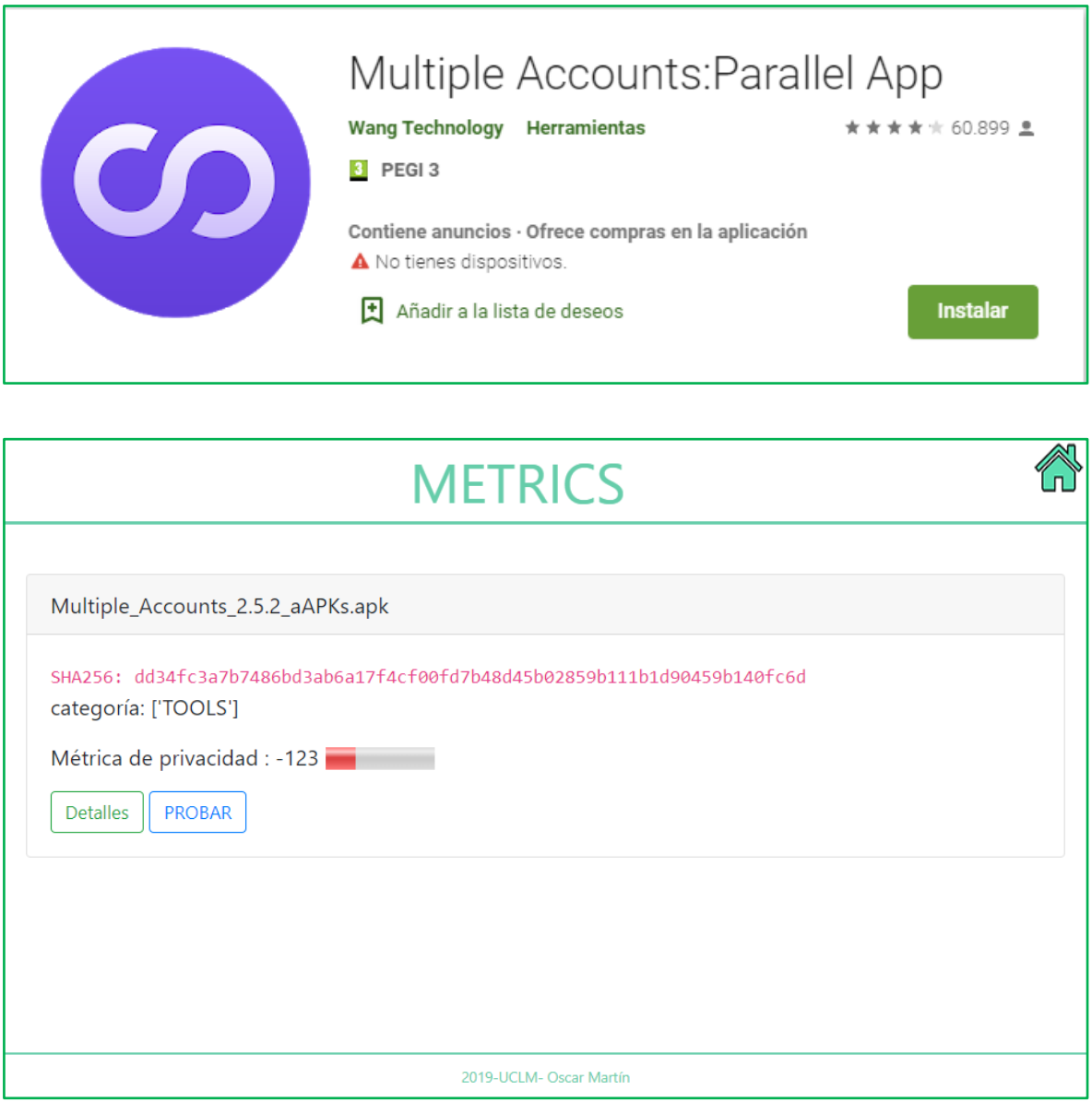


Imagen 12 App con una métrica de -123

Está claro que depende mucho del uso que tenga la aplicación móvil puede solicitar más o menos permisos, pero sí que es curioso que aplicaciones del mismo desarrollador como Whatsapp e Instagram difieran tanto en la métrica.

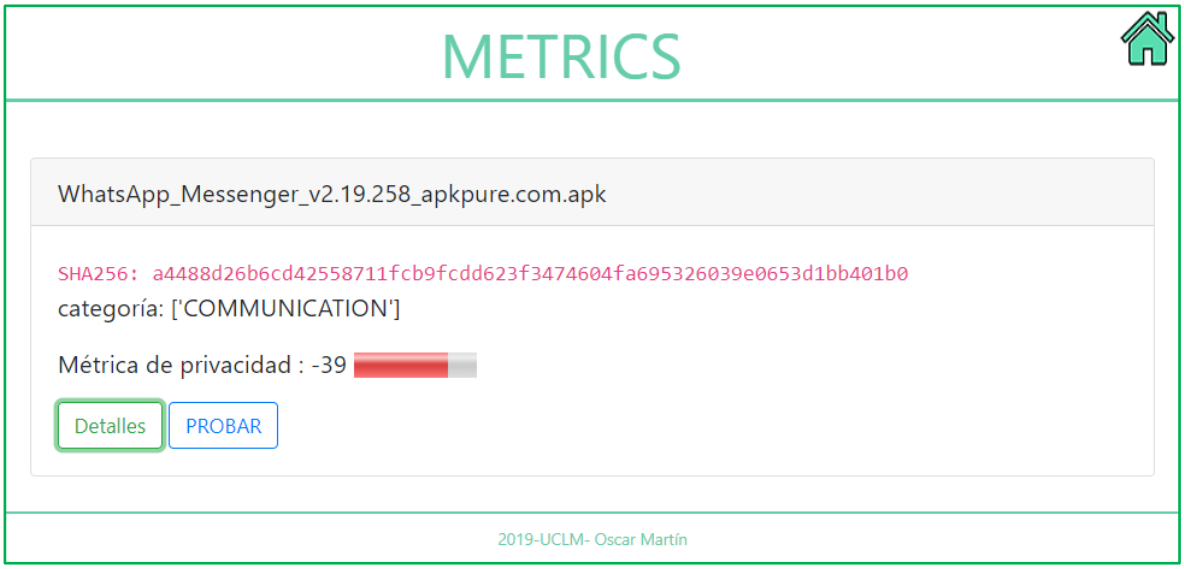


Imagen 13. Métrica de WhatsApp

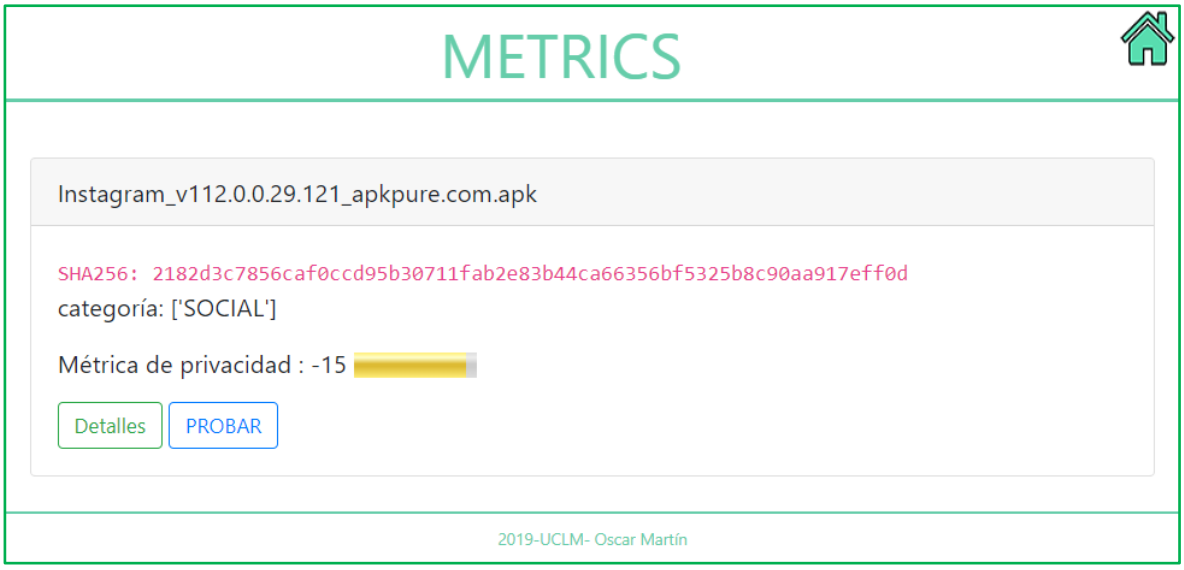


Imagen 14. Métrica de Instagram

Sobre todo, porque son dos aplicaciones que los jóvenes usan como medio de comunicación.

Si hablamos de juegos nos podemos encontrar con una métrica de -2 en la aplicación del juego de Lego comentado anteriormente o el Candy Crush donde nos da una métrica de -4 y como un juego de números para niños con una métrica de -8.

Viendo que para una misma categoría y diferente objetico en la edad indicada se emplean permisos que considero que no son necesarios y desde luego afectan a la privacidad, encontrándome con algunas aplicaciones de otra índole donde el número de permisos solicitados son excesivos.

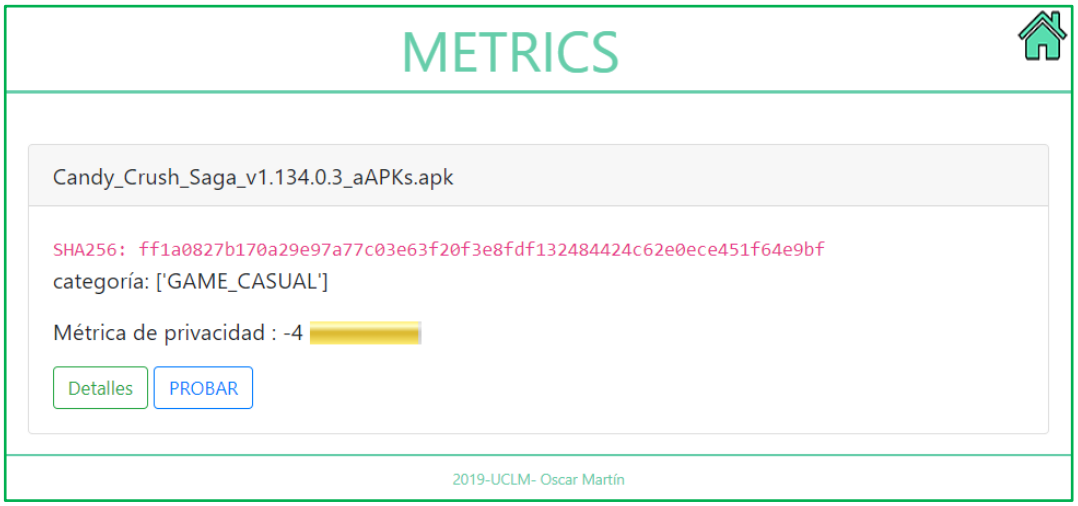
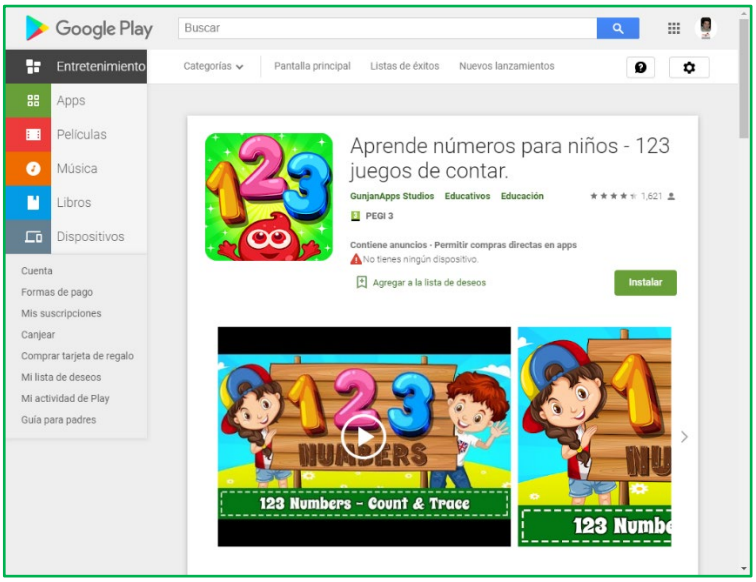


Imagen 15. Métrica de Candy Crush



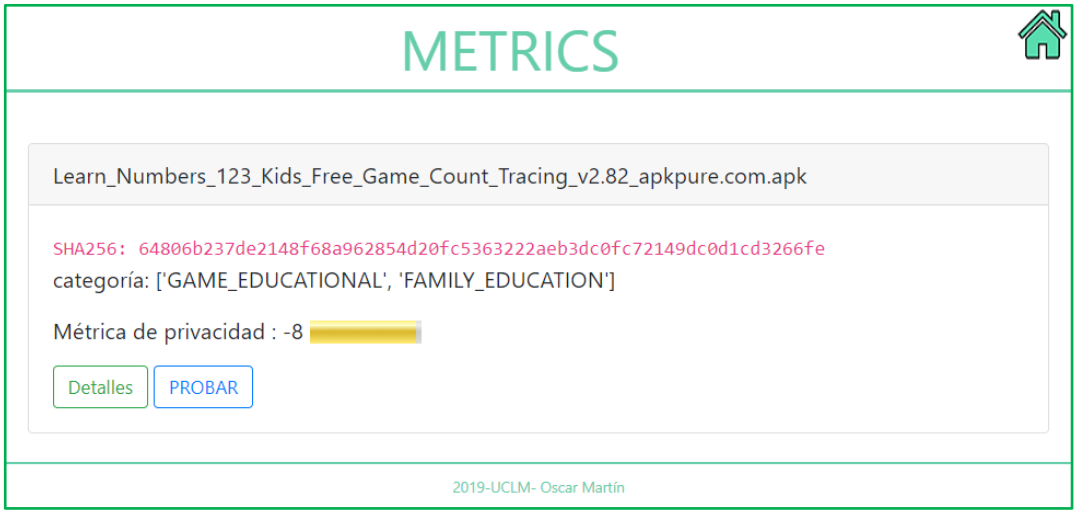


Imagen 16. Métrica juego de números para niños

CAPÍTULO 9. Conclusiones y propuestas

Como he ido describiendo a lo largo de este trabajo el realizar una métrica que mida como cuida la privacidad las aplicaciones móviles es complejo ya que depende de factores como el propósito de la aplicación, la edad a la que está dirigida y otros que hacen que afinar la métrica requiere de un análisis elevado de aplicaciones, pero que para poder hacerse una idea de cómo afecta a la privacidad el uso de las aplicaciones móviles Android que llevamos en nuestros dispositivos esta métrica propuesta puede servir para entender fácilmente como afecta a nuestra privacidad, además se puede mejorar y ajustar.

9.1 CONCLUSIONES

Que los objetivos que definí para este proyecto los he conseguido, aunque me ha llevado más tiempo del que pensaba y he podido dedicar.

Que sería bueno que un indicador como este estuviera presente en las tiendas de aplicaciones móviles para ayudar al usuario a entender mejor como afectan a la privacidad.

.

9.2 TRABAJO FUTURO Y POSIBLES AMPLIACIONES

Para finalizar con este último capítulo, comentaré alguna de las propuestas que se podrían añadir al trabajo actual en trabajos futuros:

- Crear un plugin para los navegadores que permitan ver la métrica cuando se accede a la tienda de Google a ver una aplicación, igual que se pude consultar los permisos que tiene.
- Como ya existen framework para el análisis de aplicaciones móviles como MobSF, poder añadirle en los análisis estáticos la métrica de privacidad.
- Realizar un análisis masivo de App's para procesar los resultados y permitir realizar un ajuste fino de la métrica y poder añadirla como valor en la primera propuesta.

BIBLIOGRAFÍA

-*CONAN mobile | Oficina de Seguridad del Internauta*. https://www.osi.es/es/conan-mobile. Accedido 14 de agosto de 2019.

-«Decálogo de buenas prácticas en seguridad móvil». *INCIBE*, 13 de enero de 2016, https://www.incibe.es/protege-tu-empresa/blog/decalogo-buenas-practicas-seguridad-movil.

-*ANÁLISIS DE LOS FLUJOS DE INFORMACIÓN EN ANDROID HERRAMIENTAS PARA EL CUMPLIMIENTO DE LA RESPONSABILIDAD PROACTIVA estudio-flujos-informacion-android.pdf*. https://www.aepd.es/media/estudios/estudio-flujos-informacion-android.pdf. Accedido 14 de agosto de 2019.

-«Manifest.Permission». *Android Developers*, https://developer.android.com/reference/android/Manifest.permission. Accedido 27 de agosto de 2019.

-*Protege tu móvil: revisa los permisos de tus apps | Oficina de Seguridad del Internauta*. https://www.osi.es/es/actualidad/blog/2015/01/07/protege-tu-movil-revisa-los-permisos-de-tus-apps. Accedido 14 de agosto de 2019.

-*Reverse engineering, Malware and goodware analysis of Android applications ... and more (ninja !): androguard/androguard*. 2014. androguard, 2019. *GitHub*, https://github.com/androguard/androguard.

-*Tips on privacy*. https://www.botfree.eu/en/tutorials/privacy.html. Accedido 14 de agosto de 2019.

-*Welcome to Androguard's documentation! — Androguard 3.4.0 documentation*. https://androguard.readthedocs.io/en/latest/. Accedido 17 de agosto de 2019.

-*Cómo implementar facturación integrada | Android Developers*. https://developer.android.com/google/play/billing/billing_integrate.html?hl=es-419.

Accedido 12 de septiembre de 2019.

-Derr, Erik. *axplorer: Android Permission Mappings. Contribute to reddr/axplorer development by creating an account on GitHub*. 2017. 2019. *GitHub*, https://github.com/reddr/axplorer.

-Gibler, Clint, et al. «AndroidLeaks: Automatically Detecting Potential Privacy Leaks in Android Applications on a Large Scale». *Trust and Trustworthy Computing*, editado por Stefan Katzenbeisser et al., vol. 7344, Springer Berlin Heidelberg, 2012, pp. 291-307. *DOI.org (Crossref)*, doi:10.1007/978-3-642-30921-2_17.

-*GitHub - secure-software-engineering/FlowDroid: FlowDroid Static Data Flow Tracker*. https://github.com/secure-software-engineering/FlowDroid. Accedido 14 de agosto de 2019.

-*Manifest.permission |AndroidDevelopers*. https://developer.android.com/reference/android/Manifest.permission. Accedido 12 de septiembre de 2019.

-*SuSi: our tool to automatically discover sources and sinks in the Android framework - secure-software-engineering/SuSi*. 2013. Secure Software Engineering Group at Paderborn University and Fraunhofer IEM, 2019. *GitHub*, https://github.com/secure-software-engineering/SuSi.

-Zimmeck, Sebastian. *Using Machine Learning to improve Internet Privacy*. 2017. *Semantic Scholar*, doi:10.7916/D8862N5F.

GLOSARIO DE TÉRMINOS

App

Aplicación Móvil Android.

FLASK

MicroFramework escrito en Pytihon para desarrollar aplicaciones web.

Apktool

Herramienta escrita en lenguaje java para descomprimir el archivo .apk de las aplicaciones móviles.

ANEXO A. **Archivos del proyecto.**

Estructura de las carpeta y archivos del proyecto:

Carpetas:

- app: donde se suben las aplicaciones cuando se analizan
- descompilar: donde se guarda por cada App un directorio con el hash sha256 donde está la App descomprimida para acceder a archivo AndroidManifest.xml
- Include,Lib,Scripts carpetas del entorno de Python creado
- Static : carpeta que contienen los estilos e imágenes de la web
- Templates: contiene las plantillas para Flask que se usan en la web
- Tools: contienen la herramienta apktools_24.0.jar

Archivos:

- app_metricas.json: archivo donde guardamos los resultados de las App analizadas para su posterior estudio.
- metrica.json: archivo en formato json que contiene los valores de los permisos para aplicar la métrica.
- metrica.xlsx contiene la misma información que el anterior archivo en formato Excel
- metrica_app.py archivo Python que contiene la aplicación web
- tfm_final.ipynb libreta de jupiter-notebook para ayudar a crear metrica.json

Contenido del archivo metrica_app.py

```
    import os
    import io
    import re
    import json
    import requests
    import hashlib
    import sys
    import subprocess
    import logging
    import datetime
    import pandas as pd
    from flask import Flask, flash, render_template, request,
redirect, url_for, send_from_directory

    from werkzeug.utils import secure_filename
    from xml.dom import minidom


    logger = logging.getLogger(__name__)

    UPLOAD_FOLDER = './app/'
    ALLOWED_EXTENSIONS = {'apk','xml'}
    APKTOOL = "./tools/apktool_2.4.0.jar"
    PERMISOS = './descompilar/'

    app = Flask(__name__)
    app.config['SECRET_KEY']                                    =
'7110c8ae51a4b5af97be6589caef90e4bb9bdcb3380af888f90b23a5d1616bf
319bc298105da20fe'
    app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
    app.config['PERMISOS'] = PERMISOS
    app.config['APKTOOL'] = APKTOOL

    df3 = pd.read_json('metrica.json')
    logger.info("Cargamos JSON...")
    logger.info(df3.index)

    def ficheros_permitidos(fichero):
    return '.' in fichero and fichero.rsplit('.',1)[1].lower() in
ALLOWED_EXTENSIONS


    @app.route('/')
    def inicio():
    return render_template("index.html")

    @app.route('/upload/',methods=['GET','POST'])
```

```
    def subir_app():
    if request.method == 'POST':
         if 'App' not in request.files:
              print('No se ha subido ningún fichero.')
              return redirect(request.url)
         f = request.files['App']
         if f.filename == '':
              print('No se ha seleccionado ningún fichero.')
              return redirect(request.url)
         if f and ficheros_permitidos(f.filename):
              fich = secure_filename(f.filename)
              f.save (app.config['UPLOAD_FOLDER']+fich)
              app_hash                                        =
str(hash_calc(app.config['UPLOAD_FOLDER']+fich))+'.apk'
              if ~os.path.isdir(app_hash):
                   descompilar = "java -jar " + APKTOOL + " d -
-output " + \
                   PERMISOS+app_hash[:-4]    +    "    "    +
UPLOAD_FOLDER+fich
                   # Si la carpeta ya existe da error.
                   os.system(descompilar)
              return
redirect(url_for('ver_permisos',filename=fich))
    return render_template("subir_ficheros.html")

    @app.route('/upload/<filename>')
    def ver_app(filename):
    return
send_from_directory(app.config['UPLOAD_FOLDER'],filename)


    @app.route('/permisos/<filename>')
    def ver_permisos(filename):

    ruta = UPLOAD_FOLDER+filename
    app_hash = hash_calc(ruta)
    carpeta = app_hash+"/AndroidManifest.xml"
    f = app.config['PERMISOS']+ carpeta
    app.logger.info('fichero:' + f)
    manifest = minidom.parse(f)
    res,      metric,      package      ,      metrica_app      =
sacar_permisos(manifest,app_hash)
    categoria = app_cat(package)
    metrica_app['metricas'][0]['categoria']=categoria
    metrica_app['metricas'][0]['fichero'] = filename
    df = pd.DataFrame(metrica_app)
    if os.stat("app_metricas.json").st_size != 0:
         df2 = pd.read_json('app_metricas.json')
         i = df2['metricas'].count()
```

```
          print(df2['metricas'].count())
          df2 = df2.append(df)
          print(df2['metricas'].count())
          df2.to_json('app_metricas.json', orient='records')
     else:
          df.to_json('app_metricas.json', orient='records')


     #logger.info(res)
     return     render_template("app_permisos.html",     dat=res,
met=metric, app=filename, apphash=app_hash, cat=categoria)

     # Recupera la categoria de la App desde google Play
     def app_cat(package):
     try:
          payload = {'id': package, 'hl': 'en'}
          response                                                =
requests.get("https://play.google.com/store/apps/details",
params=payload)
          numero = 0
          lista_categorias = {"categorias": []}
          regex                =                'itemprop="genre"
href="\/store\/apps\/category/(?P<categoria_en>[\w]+)'
          for match in re.finditer(regex, response.text):
               numero += 1

     lista_categorias["categorias"].append(match.group('categoria
_en'))
          return (lista_categorias["categorias"])
     except:
          return("Sin categoría")


     # Calcula el hash SHA256 de la APK subida.
     def hash_calc(filename):

     try:
          sha256 = hashlib.sha256()
          block_size = 65536
          with io.open(filename, 'rb') as appfile:
               buf = appfile.read(block_size)
               while buf:
                    logger.info('calculando el hash ...')
                    sha256.update(buf)
                    buf = appfile.read(block_size)
          sha256val = sha256.hexdigest()
          print(sha256val)
          return sha256val
     except:
```

```
          print("Calculando hashes, falla")


    def get_metrica2(x):
        total = 0
        comunicacion = ['PHONE', 'SMS', 'MMS','NETWORK',
                        'BLUETOOTH', 'INTERNET', 'WIFI', 'NFC']
        personal = ['CALENDAR', 'NUMBER', 'CALL',
                    'LOCATION',              'EXTERNAL_STORAGE',
'LOG','CONTACTS']
        palabras_permiso = x.split('_')
        for i in palabras_permiso:
            if i in comunicacion:
                total += -1
            if i in personal:
                total += -1
        return total

    def sacar_permisos(mfxml,h):
    try:
          perm = []
          dvm_perm = {}
          permissions     =     mfxml.getElementsByTagName("uses-
permission")
          manifest = mfxml.getElementsByTagName("manifest")
          suma = 0
          for node in manifest:
                package = node.getAttribute("package")
                # No saca los valores , revisarlo.
                androidversioncode                                =
node.getAttribute("platformBuildVersionCode")
                androidversionname                                =
node.getAttribute("platformBuildVersionName")

          for permission in permissions:

    perm.append(permission.getAttribute("android:name"))
          for i in perm:
                prm = i
                pos = i.rfind(".")
                if pos != -1:
                      prm = i[pos + 1:]
                try:
                      m = df3['METRICA'][prm]
                      n = df3['METRICA2'][prm]
                      level = df3['LEVEL'][prm]
                      suma += m+n
                      dvm_perm[prm] = (level,m,n)
```

```
                except KeyError:
    # Permisos no definidos
    # Utiliza la metrica2 para definir la ponderación cuando no
se tiene definido.
                    n = get_metrica2(prm)
                    dvm_perm[prm] = [
                                        "no-defined",0, n
                                    ]
                    suma += n
          x = datetime.datetime.now()

          metrica_app = {'metricas':[{'metrica': suma,
                      'packagename': package,
                      'categoria': '',
                      'fichero': '',
                      'hash': h,
                      'version': androidversioncode,
                      'version name': androidversionname,
                      'fecha': x,
                      'perm': dvm_perm
                      }]
                      }



          logger.info('Manifest parseado.....')
          logger.info('Metrica: %s',suma)

          return(dvm_perm, suma, package, metrica_app)
    except:
          print("Extracting Manifest Data")
```

Para ejecutar el proyecto se necesita tener instalado java y Python 3 y ejecutar el archivo arrancar_flask.bat en un ordenador con Windows

```
REM
SET FLASK_APP=metrica_app.py
SET FLASK_ENV=development
.\Scripts\python.exe -m flask run --host 0.0.0.0
```

Para ejecutarlo en un ordenador Linux o MacOS hay que llamar al archivo arrancar_flask.sh

```
source ./bin/activate
export FLASK_APP="metrica_app.py"
export FLASK_ENV="development"
./bin/flask run --host 0.0.0.0
```

Los pasos para instalar el proyecto son los siguientes:

Descomprimir el archivo METRICS.zip

En un ordenado con Windows

```
python -m venv METRICS
cd METRICS
Scripts\activate
pip install -r requirments
```

En un ordenado con Linux o MacOS

```
python -m venv METRICS
cd METRICS
source ./bin/activate
./bin/pip install -r requirements
```

Y abrir el navegador en la url http://localhost:5000

ANEXO B. **Entornos virtuales en Python**

B.1 Flask

Para el desarrollo de la aplicación web creo un primer entorno virtual.

Python -m venv metrics

Python install -r requirements.txt

El fichero de requerimientos utilizados es el siguiente:

```
astroid==2.2.5
certifi==2019.9.11
chardet==3.0.4
Click==7.0
Flask==1.1.1
Flask-WTF==0.14.2
get==2019.4.13
idna==2.8
isort==4.3.21
itsdangerous==1.1.0
Jinja2==2.10.1
lazy-object-proxy==1.4.1
MarkupSafe==1.1.1
mccabe==0.6.1
numpy==1.17.1
pandas==0.25.1
post==2019.4.13
public==2019.4.13
pylint==2.3.1
```

```
python-dateutil==2.8.0
pytz==2019.2
query-string==2019.4.13
requests==2.22.0
six==1.12.0
typed-ast==1.4.0
urllib3==1.25.3
Werkzeug==0.15.5
wrapt==1.11.2
WTForms==2.2.1
```

B.2 Jupyter-notebooks y Pandas

Para ayudarme en el análisis de los resultados y preparar el archivo que contiene los valores de la métrica de los permisos de las aplicaciones creo un entorno el siguiente entorno virtual.

En este caso los requerimientos empleados son los que utilicé para la asignatura de CiberInteligencia y Análisis de Datos que ya tenía montado y me ha servido para este trabajo.

La libreta que utilizo es el archivo: tfm_final.ipynb

Python -m venv metrica

Python install -r requirements.txt

El fichero de requerimientos utilizados es el siguiente:

```
absl-py==0.7.1
astor==0.8.0
attrs==19.1.0
backcall==0.1.0
bleach==3.1.0
certifi==2019.6.16
chardet==3.0.4
colorama==0.4.1
cycler==0.10.0
decorator==4.4.0
defusedxml==0.6.0
entrypoints==0.3
et-xmlfile==1.0.1
gast==0.2.2
```

```
google-pasta==0.1.7
grpcio==1.21.1
h5py==2.9.0
idna==2.8
ipykernel==5.1.1
ipython==7.5.0
ipython-genutils==0.2.0
ipywidgets==7.4.2
jdcal==1.4.1
jedi==0.13.3
Jinja2==2.10.1
joblib==0.13.2
jsonschema==3.0.1
jupyter==1.0.0
jupyter-client==5.2.4
jupyter-console==6.0.0
jupyter-core==4.4.0
Keras==2.2.4
Keras-Applications==1.0.8
Keras-Preprocessing==1.1.0
kiwisolver==1.1.0
Markdown==3.1.1
MarkupSafe==1.1.1
matplotlib==3.1.0
mistune==0.8.4
nbconvert==5.5.0
nbformat==4.4.0
notebook==5.7.8
numpy==1.16.4
openpyxl==2.6.3
pandas==0.24.2
pandocfilters==1.4.2
parso==0.4.0
pickleshare==0.7.5
prometheus-client==0.7.0
prompt-toolkit==2.0.9
protobuf==3.8.0
Pygments==2.4.2
pyparsing==2.4.0
pyrsistent==0.15.2
python-dateutil==2.8.0
pytz==2019.1
pywinpty==0.5.5
PyYAML==5.1.1
pyzmq==18.0.1
qtconsole==4.5.1
requests==2.22.0
scikit-learn==0.21.2
scipy==1.3.0
```

```
seaborn==0.9.0
Send2Trash==1.5.0
six==1.12.0
sklearn==0.0
tensorboard==1.14.0
tensorflow==1.14.0
tensorflow-estimator==1.14.0
termcolor==1.1.0
terminado==0.8.2
testpath==0.4.2
tornado==6.0.2
traitlets==4.3.2
urllib3==1.25.3
wcwidth==0.1.7
webencodings==0.5.1
Werkzeug==0.15.4
widgetsnbextension==3.4.2
wrapt==1.11.2
xlrd==1.2.0
```