



Situación del malware para Android

INSTITUTO NACIONAL DE
CIBERSEGURIDAD

SPANISH NATIONAL
CYBERSECURITY INSTITUTE

 **incibe**_

HISPASEC

Autor

Asier Martínez Retenaga

Este estudio ha sido elaborado con la colaboración de Jesús Díaz Vico y Héctor R. Suárez por parte de **INCIBE** y Francisco López Hernández y Javier Rascón Mesa por parte de **HISPASEC**.

Febrero 2015

La presente publicación pertenece a INCIBE (Instituto Nacional de Ciberseguridad) y está bajo una licencia Reconocimiento-No comercial 3.0 España de Creative Commons. Por esta razón está permitido copiar, distribuir y comunicar públicamente esta obra bajo las condiciones siguientes:

- Reconocimiento. El contenido de este informe se puede reproducir total o parcialmente por terceros, citando su procedencia y haciendo referencia expresa tanto a INCIBE o CERTSI como a su sitio web: <http://www.incibe.es>. Dicho reconocimiento no podrá en ningún caso sugerir que INCIBE presta apoyo a dicho tercero o apoya el uso que hace de su obra.
- Uso No Comercial. El material original y los trabajos derivados pueden ser distribuidos, copiados y exhibidos mientras su uso no tenga fines comerciales.

Al reutilizar o distribuir la obra, tiene que dejar bien claro los términos de la licencia de esta obra. Alguna de estas condiciones puede no aplicarse si se obtiene el permiso de CERTSI como titular de los derechos de autor. Texto completo de la licencia: <http://creativecommons.org/licenses/by-nc-sa/3.0/es/>

ÍNDICE

1	SITUACIÓN ACTUAL	5
2	SOBRE EL ESTUDIO	9
3	MODELO DE SEGURIDAD DE ANDROID	10
4	EL FORMATO APK	13
5	ANÁLISIS DE MUESTRAS	15
5.1.	Tipos de ficheros	15
5.2.	APKs Firmadas	16
5.3.	APKs monetizadas	18
5.4.	SDKVersion	25
5.5.	Tiempo de exposición a la amenaza	27
5.6.	Plataforma de desarrollo	29
5.7.	Nombre del package	30
5.8.	Package Name existentes en Google Play	31
5.9.	Permisos	33
5.10.	Services	39
5.11.	Receivers	39
5.12.	Providers	40
5.13.	Activities	41
5.14.	Complejidad	41
5.15.	Peligrosidad	42
5.16.	Idioma por defecto	43
5.17.	Idiomas alternativos	44
5.18.	Palabras más utilizadas	46
5.19.	Conexiones establecidas	47
5.20.	Detecciones de Antivirus	49
5.21.	Tipo de malware	50
5.22.	Privacidad	51
6	CONCLUSIONES	53
ANEXO 1 – COMPARATIVA DE LIBRERÍAS DE MONETIZACIÓN		57

ANEXO 2 – LISTADO DE MOTORES ANTIVIRUS	58
ANEXO 3 – LISTADO DE MÉTODOS QUE AFECTAN A LA PRIVACIDAD	60
ÍNDICE DE ILUSTRACIONES	61
REFERENCIAS	63

1 SITUACIÓN ACTUAL



Como reflejan los siguientes ejemplos, Android se ha convertido en la plataforma móvil más atacada por parte de los cibercriminales.

- El informe de malware correspondiente al segundo trimestre de 2014 elaborado por Kaspersky Lab, indica que Android acapara el 99% de los programas maliciosos móviles.
 - El informe «[Cisco's 2014 Annual Security Report](#)» indica, de igual modo, que el 99% de malware para plataformas móviles está orientado a Android.
 - [Kindsight Security Labs](#) estima que hay 15 millones de dispositivos móviles infectados con malware en todo el mundo, casi cuatro millones más que a finales de 2013, de los cuales el 60% utilizan sistema operativo Android.
 - El informe «[State of Mobile App Security](#)» de Arxan revela que existen versiones maliciosas del 97% del TOP 100 de aplicaciones de pago y el 80% del TOP 100 de aplicaciones gratuitas de Android, la cuales, son distribuidas por un gran número de mercados alternativos.

- El informe «[*Cheetah Mobile's security report*](#)» refleja que han identificado más de 2,2 millones de aplicaciones maliciosas correspondientes a la plataforma Android.
- El informe «[*Security Threat Report 2014*](#)» de Quick Heal revela que el malware para Android ha aumentado un 600% desde 2012.

Este hecho es debido principalmente a seis factores:

1. La mayoritaria elección por parte de los usuarios de Android como plataforma móvil: ya existen más de [900 millones de dispositivos](#).
2. Como se puede comprobar en el sistema de seguimiento de incidencias o «[*Android Open Source Project - Issue Tracker*](#)», la plataforma tiene multitud de fallos. Del mismo modo, los controles de seguridad que incorpora no son 100% fiables. Algunas de las noticias más relevantes al respecto son las siguientes:
 - [Investigadores de BlueBox](#) dieron a conocer «*MasterKey*», una vulnerabilidad que fue posteriormente detallada en la conferencia «[*Android: One Root to Own Them All*](#)», en la Black Hat USA 2013. El fallo de seguridad permitía incluir código arbitrario en cualquier APK firmada por un desarrollador sin alterar la firma. Para ello, se debía incluir en la APK dos ficheros con el mismo nombre, de modo que la verificación de Android se aplicaba únicamente sobre el primero, ejecutando posteriormente el segundo.
 - Un investigador chino hizo pública [una vulnerabilidad](#) mediante la cual era posible modificar el contenido de una APK sin que lo detecten los mecanismos de validación de Android. Pese a que dicha vulnerabilidad es difícil de explotar y sólo se da en ciertos casos (modificando aquellos ficheros classes.dex con un tamaño inferior a 64kb), podría permitir a un atacante obtener información sensible del usuario afectado.
 - [Investigadores de Bluebox](#) hicieron pública la vulnerabilidad conocida como «*FakeID*», que ha sido posteriormente detallada en la charla «[*Android Fake ID Vulnerability*](#)», en la Black Hat USA 2014. La vulnerabilidad, que lleva afectando a usuarios desde enero del año 2010, consiste en que el instalador de paquetes de Android no comprueba la cadena de certificación, de modo que si se falsea el emisor del certificado de una aplicación, Android no se percata de tal hecho y lo tomará como un certificado válido.
 - El investigador Rafay Baloch hizo pública [una vulnerabilidad](#) que afecta al navegador por defecto presente en cerca del 75% de los dispositivos Android. Esta vulnerabilidad permite eludir la política de SOP «*Same Origin Policy*», de manera que es posible obtener tanto información del resto de las páginas abiertas en el navegador como acceder a las cookies, y en el caso de que no estén protegidas, utilizarlas para suplantar a los usuarios afectados o secuestrar sus sesiones.

3. Los controles de seguridad del servicio «*Google Bouncer*» a la hora de aceptar la publicación de nuevas aplicaciones en Google Play no son infalibles.
4. La explotación de dichos fallos supone un gran beneficio económico con una mínima inversión. Se puede obtener gran cantidad de información sensible como fotografías, emails, cuentas y contraseñas, SMS, grabaciones, etc. que posteriormente puede ser vendida en el mercado underground para robos de identidad, chantajes, etc. Así mismo, la suscripción a servicios [SMS Premium](#) reporta grandes ganancias a los ciberdelincuentes, o incluso la subcontratación de servicios característicos de una botnet como por ejemplo ataques de denegación de servicio, o mailings masivos pertenecientes a campañas de phishing, o los 0days en Android por los que, según la revista [Forbes](#), se llega a pagar entre 30 mil y 60 mil dólares.
5. Un alto porcentaje de las aplicaciones para Android no se desarrollan utilizando metodologías cuya máxima sea el «*security by design*». Así mismo, no se utilizan otras como [OWASP](#) u [OASAM](#), las cuales permiten evaluar la seguridad de las aplicaciones y tienen como objetivo identificar los peligros correspondientes a la seguridad de dispositivos móviles y proporcionar los controles necesarios en el desarrollo para reducir su impacto y la probabilidad de explotación de los mismos. Atendiendo al informe «[Predicts 2014: Mobile Security Won't Just Be About the Device](#)» de Gartner más del 75% de las aplicaciones desarrolladas a lo largo del 2015 fallarán las pruebas de seguridad básicas. En este aspecto, no ayudan demasiado ciertas opciones por defecto de algunos de los componentes de las aplicaciones de Android. Por ejemplo, si no configura lo contrario, un «*provider*», permite que la información que comparte una aplicación sea accesible tanto por el sistema como por aplicaciones de terceros, salvo que se establezca el atributo *android:exported="false"*.

Así mismo, es importante tener en cuenta que a la hora de desarrollar una aplicación se acostumbra a reutilizar código; no vamos a reinventar la rueda. Según un [informe](#) de la compañía Codenomicon publicado en RSA Conference USA 2014, entre el 80% y el 90% del software desarrollado para dispositivos móviles se realiza a partir de código reutilizado. En la mayoría de los casos no se tienen en cuenta los fallos de seguridad que puede tener el código reutilizado.

6. Los usuarios no son conscientes del peligro real y, en muchos casos, no toman las medidas necesarias para mantener su seguridad y privacidad. El informe «[Mobile Security 2015: The Enterprise at Risk](#)» indica que el 34% de los usuarios no toma ningún tipo de medida relacionada con la seguridad de sus dispositivos.

Todo esto provoca que los cibercriminales generen constantemente aplicaciones que buscan explotar tanto la inexperiencia y credulidad de los usuarios como los fallos de seguridad del propio sistema. Dichas aplicaciones están orientadas principalmente a realizar las [siguientes actividades](#):

- Suscripción a servicios de tarificación especial «[SMS Premium](#)».
- Descarga e instalación de aplicaciones no deseadas, principalmente provenientes de mercados alternativos o ubicaciones no oficiales.
- Seguimiento de las ubicaciones del dispositivo y grabación del audio o video para monitorizar al usuario.
- Monitorización de SMS provenientes de servicios bancarios.
- Robo de información personal como contactos, imágenes, vídeos.
- Simulando ser aplicaciones de utilidad cuando realmente no tienen ninguna, como los [falsos antivirus](#).
- [Secuestro del dispositivo móvil](#).

2 SOBRE EL ESTUDIO

Ante el escenario descrito en la sección de Situación actual, surge la necesidad de analizar en profundidad las características técnicas del malware en Android: «Conoce a tu enemigo y conócete a ti mismo; en cien batallas, nunca saldrás derrotado.» (El arte de la guerra - Sun Tzu). De este modo, es posible conocer en profundidad ciertos aspectos comunes de este tipo de amenazas y así promover la aplicación de las medidas necesarias para mitigarlas y reducir su impacto y exposición.

El presente informe se ha podido realizar gracias a la **colaboración entre el Instituto Nacional de Ciberseguridad, INCIBE e HISPASEC**, en el que se han aunado esfuerzos para ofrecer una visión concreta desde un punto de vista analítico. Para ello, se han recopilado y analizado con la colaboración de HISPASEC un total de 76.000 muestras de aplicaciones maliciosas obtenidas en el último semestre del 2014.

Las muestras que se han analizado son detectadas por antivirus con grandes ratios de detección de malware para Android con el fin de dar un importante grado de fiabilidad al estudio y evitar incluir falsos positivos. Los antivirus seleccionados han sido: [ESET-NOD32](#), [BitDefender](#), [Commtouch](#), [Gdata](#) y [MicroWorld-eScan](#).

Para la realización del análisis se han utilizado por un lado los informes de Virustotal, dicho servicio además de analizar las muestras con multitud de motores antivirus, también utiliza la herramienta [Androguard](#) para realizar un análisis estático de las mismas. Con esta utilidad, al descompilar las aplicaciones, es posible acceder a gran cantidad de información como los permisos que solicitan, los «*services*», «*receivers*» o «*providers*», las librerías de publicidad que utilizan, el listado de «*activities*» o cadenas significativas, etc. Además, se han adaptado otras utilidades basadas también en Androguard, que extraen información adicional de las muestras, siendo toda esta información utilizada para generar el informe.

3 MODELO DE SEGURIDAD DE ANDROID

Para entender las características que tiene el malware para Android es importante tener claro el modelo de seguridad de este sistema operativo móvil para poder evaluar los puntos fuertes y débiles, y así hacerse una idea de las deficiencias o aspectos a mejorar que actualmente son explotados por los desarrolladores de malware. La arquitectura de Android está basada en un modelo multicapa como muestra la Ilustración 1:

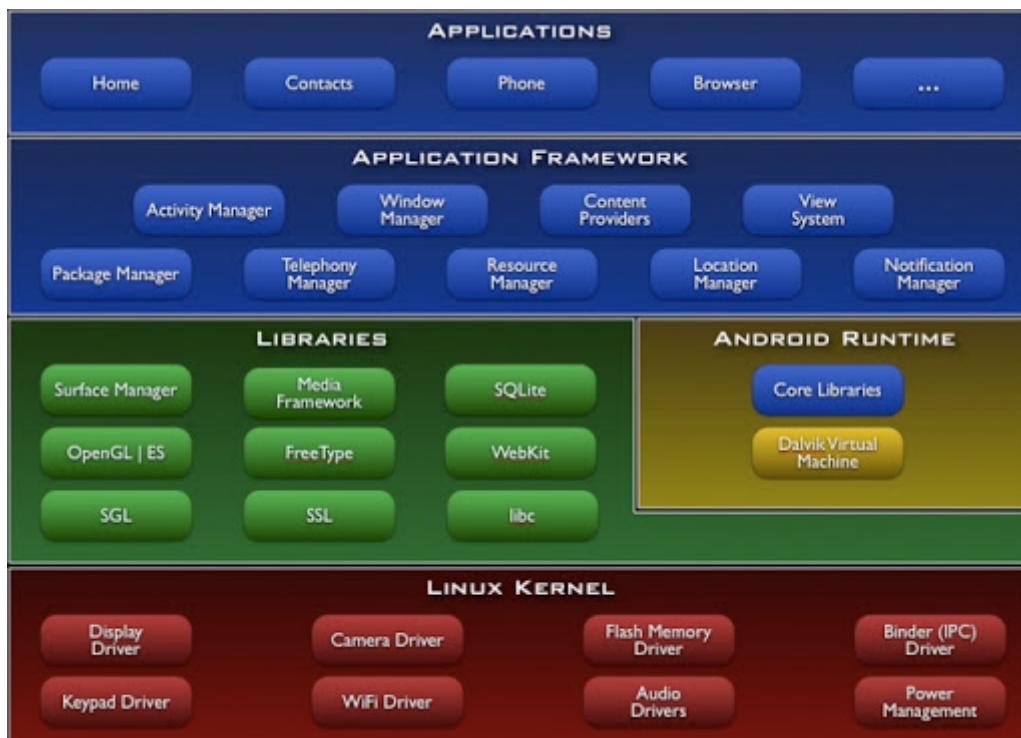


Ilustración 1: Arquitectura de Android

- **Aplicaciones** – corresponde al nivel más alto del modelo y está formado tanto por las aplicaciones que vienen por defecto en el dispositivo como aquellas que el usuario instale con posterioridad.
- **Framework** – es la capa que contiene las funcionalidades llave del sistema de Android como el Package Manager que permite de instalar/borrar aplicaciones Android, ActivityManager que maneja el ciclo de vida de cada actividad de cada aplicación, etc.
- **Librerías** – corresponde al conjunto de librerías que utilizan los diferentes componentes del sistema Android. Algunas de las más destacadas son las multimedia, el motor gráfico o el motor de base de datos SQLITE.
- **Android Runtime** – está formado por dos componentes: las librerías del CORE y la máquina virtual Dalvik o su sucesora, ART. Todas las aplicaciones son ejecutadas en una máquina virtual con el fin de dotar de un entorno de seguridad al modelo.

- **Kernel** – corresponde a una capa de abstracción entre el hardware y el software. Incluye servicios esenciales como la gestión de memoria o de procesos, o los drivers que permiten interactuar con la cámara, audio, Wi-Fi, etc.

El modelo de seguridad está diseñado para prevenir ataques habituales como los de ingeniería social, que buscan la instalación de aplicaciones de terceros. Para ello, incorpora una serie de protecciones que reducen la probabilidad de verse afectado por este tipo de amenazas, así como el impacto en el caso de que la afección se produzca. Algunas de las medidas más importantes que incorpora la plataforma para protegerse frente los diferentes tipos de amenazas son [las siguientes](#):

- Google Play incorpora una serie de controles como por ejemplo no permitir que varias aplicaciones tengan la misma imagen de avatar o el mismo nombre. Así mismo, bloquea la publicación de aplicaciones en los [siguientes casos](#):
 - Aplicaciones que incluyen contenido ilegal.
 - Aplicaciones que facilitan juegos de apuestas reales.
 - Aplicaciones que incluyen contenido de promoción del odio.
 - Aplicaciones que incluyen pornografía.
 - Aplicaciones que incluyen violencia real gratuita.

Así mismo, incluye un servicio conocido como «*Bouncer*», encargado de verificar las aplicaciones que se suben al repositorio oficial de Google. Para ello, realiza tanto un análisis dinámico de las mismas como una comprobación de firmas frente a distintas listas de malware.

- La máquina virtual aísla a las aplicaciones cuando son ejecutadas, de manera que imposibilita que una aplicación acceda a otra.
- El framework de desarrollo de aplicaciones incorpora diferentes funcionalidades como el modelo de permisos o funciones criptográficas.
- Incorpora diferentes métodos como ASLR, [ProPolice](#) o [safe-iop](#) con el fin de mitigar riesgos asociados a la gestión de memoria.
- Sistema de cifrado tanto para el dispositivo móvil como para tarjeta SD externa.
- Por defecto, la plataforma Android no permite la instalación de aplicaciones de «Orígenes desconocidos».
- [Sistema de verificación de aplicaciones](#) que alerta al usuario en el momento de la instalación de aplicaciones ajenas al market oficial de Android.
- «Solicitar contraseña para compras» con el fin de prevenir compras accidentales o no deseadas.
- [Google Play Services recolecta las aplicaciones que todavía no conoce para mejorar la verificación de las mismas](#).

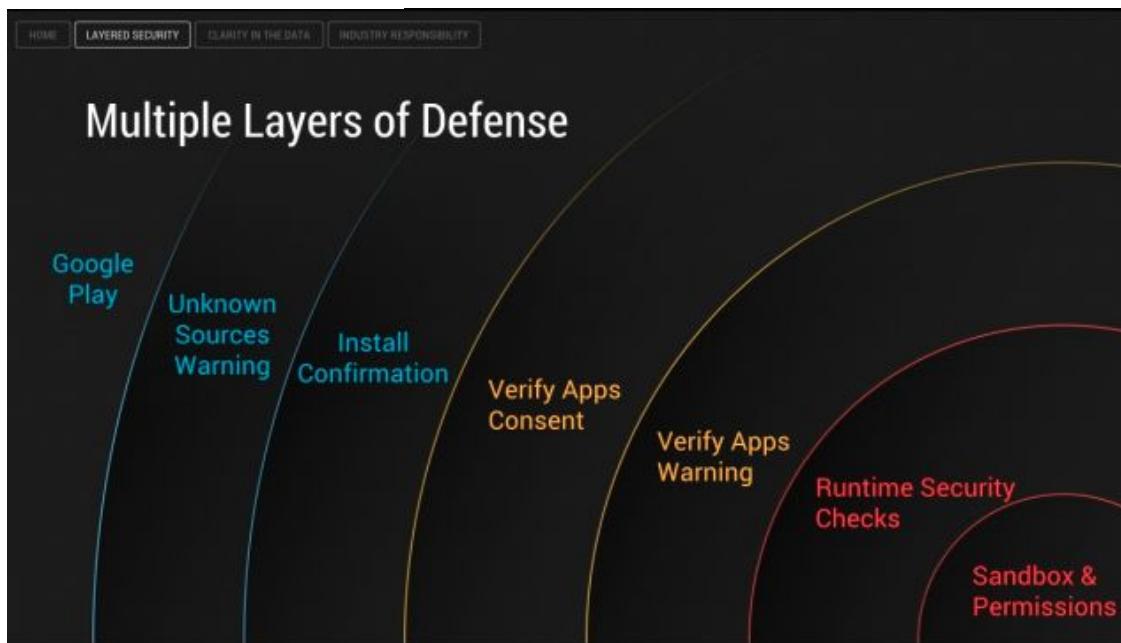


Ilustración 2: Android incorpora múltiples controles para proteger a los usuarios. Fuente: [Adrian Ludwig](#) - Jefe de seguridad de Android

Pese a todas estas capas, el modelo presenta algunas lagunas o aspectos que deberían ser mejorados:

- Las aplicaciones de manera habitual son firmadas con certificados autofirmados. Es decir, no requieren de ninguna autoridad certificadora que asegure que dicha aplicación no representa ningún riesgo para el usuario.
- La posibilidad de crear permisos personalizados puede suponer un riesgo para la privacidad.
- Los controles de seguridad del «*Bouncer*» deben de ser más exhaustivos.
- Con el último cambio en la política de permisos de Google Play Store, los usuarios ya no son notificados acerca de ciertos cambios en los permisos solicitados por las aplicaciones.

4 EL FORMATO APK

Todas las aplicaciones para dispositivos Android, y eso incluye al malware, están encapsuladas en un formato específico, conocido como APK «*Application PackAge File*». Dicho formato es el utilizado para la instalación y distribución de aplicaciones para esta plataforma móvil.

Los ficheros APK tienen estructura que se muestra en la Ilustración 3:

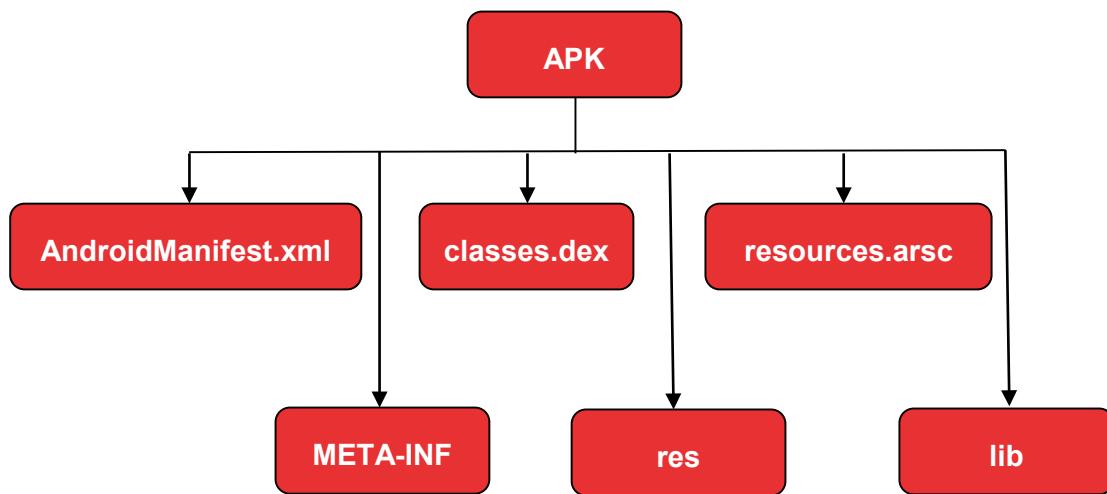


Ilustración 3: Estructura de un fichero APK.

- **AndroidManifest.xml** – es el fichero de configuración de la aplicación. En él se especifican diferentes aspectos como el identificador único de la aplicación, los componentes de la misma («*activities*», «*receivers*», «*content providers*», etc.) o los permisos que requiere la propia aplicación para un correcto funcionamiento.
- **classes.dex** – contiene el código compilado de la aplicación, en formato DEX, de modo que sea interpretable por la máquina virtual Dalvik o ART.
- **resources.arsc** – es el fichero correspondiente a los recursos compilados.
- **META-INF** – es el directorio que almacena información correspondiente a la firma digital de la aplicación y contiene los siguientes ficheros:
 - **MANIFEST.MF** – contiene un listado completo de los ficheros de la APK junto con su respectivo hash SHA-1.
 - **CERT.SF** – contiene el hash SHA-1 de cada 3 líneas que aparecen en el **MANIFEST.MF**.
 - **CERT.RSA** – almacena la firma del fichero **CERT.SF**, es decir, contiene la firma de la APK.

- **res** – es el directorio que almacena los recursos (imágenes, textos, ficheros XML, etc.) utilizados por la aplicación.
- **lib** – es el directorio que contiene el código compilado para diferentes arquitecturas: armeabi, armeabi-v7a, x86 o mips.

5 ANÁLISIS DE MUESTRAS

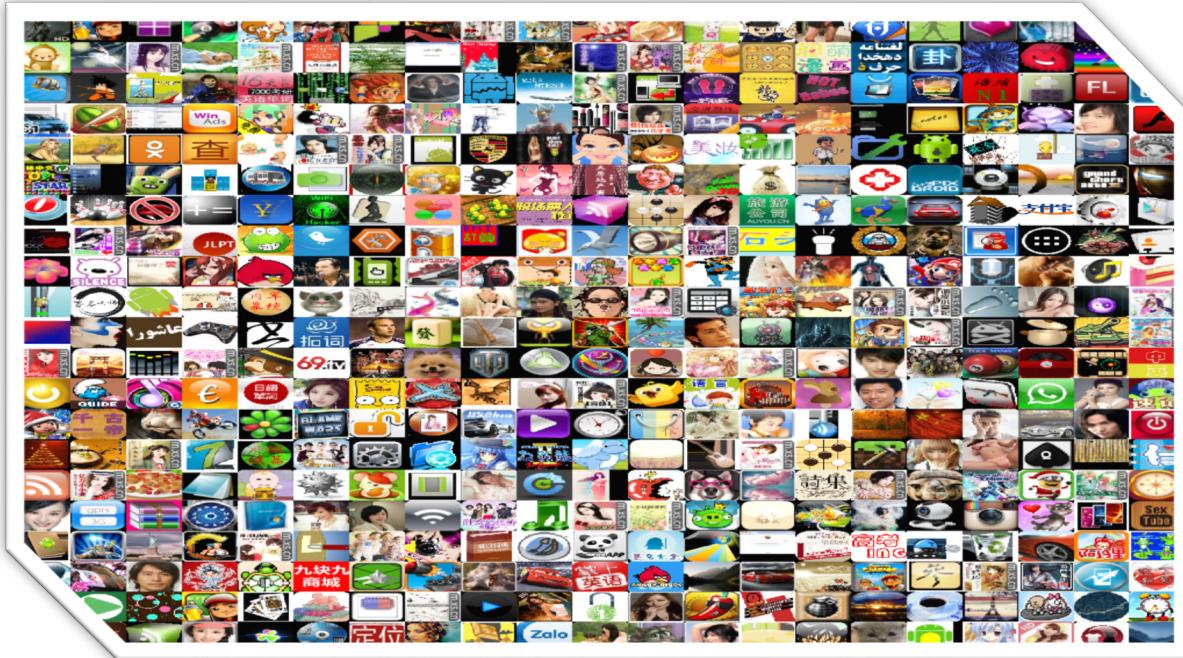


Ilustración 4: Iconos de algunas de las aplicaciones analizadas.

A continuación, se muestran los aspectos y componentes de las APKs maliciosas que se han analizado junto con los resultados obtenidos a partir de dicho análisis.

5.1. TIPOS DE FICHEROS

Se han identificado todos ficheros que se incluyen en las APKs analizadas con el fin de determinar cuáles son los tipos de ficheros más repetidos en los mismos.

En total se han registrado más de 8 millones de ficheros, lo que hace una media de 113,44 ficheros por aplicación. De todos ellos, existen más de 1.300 tipos distintos. En la Ilustración 5 se muestran los formatos de ficheros más repetidos.

Top 10 tipos de ficheros

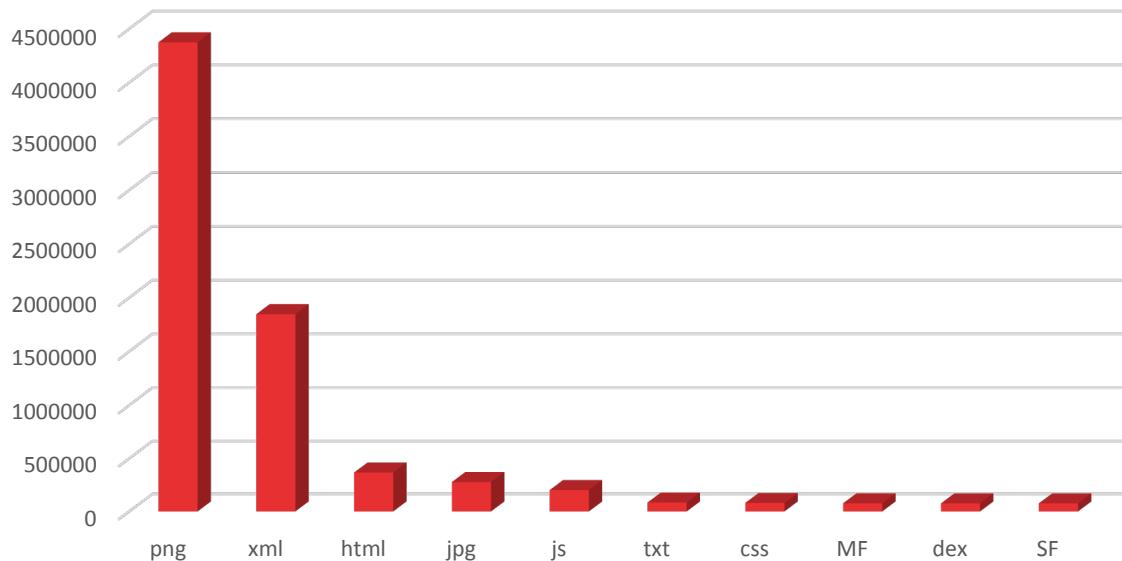


Ilustración 5: Top 10 tipos de ficheros

Además de los tipos de ficheros más repetidos, existen una serie datos reseñables:

- Se han identificado extensiones curiosas como .VIAGRA o .ACTOR.
- Hay un total de 11.925 ficheros con extensión .APK, es decir, son APKs que contienen a su vez APKs, lo que hace suponer que corresponden a instaladores de APKs que aprovechan a realizar alguna actividad maliciosa o bien directamente droppers de aplicaciones maliciosas.
- Se han identificado únicamente 26 ficheros con extensión .SQLITE o SQLITE3, que corresponden a bases de datos.
- Más de 200.000 corresponden a ficheros con extensión de audio: .MP3, .MP4, .OGG, .WAV, etc.
- Más de 10.000 corresponden a ficheros comprimidos: .RAR, .ZIP.
- Se han identificado únicamente 19 ficheros con extensión .PDF.

5.2. APKS FIRMADAS

A la hora de publicar una aplicación en Google Play, es necesario haberla firmado previamente ya que es una de las múltiples comprobaciones que realiza el «*Bouncer*» para determinar si una aplicación es maliciosa o no.

Atendiendo a la documentación de Android, en lo que se refiere a [este aspecto](#): «**Android requires that all apps be digitally signed with a certificate before they can be installed**», es decir, «**Android requiere que todas las aplicaciones sean firmadas digitalmente con un**

certificado antes de que puedan ser instaladas. Dicha firma permite comprobar la integridad de la APK, ya que contiene el hash de cada uno de los ficheros de la misma. De este modo, en el momento de la instalación, el gestor de aplicaciones de Android comprueba que dichos hashes sean correctos y así se cerciora de que la APK no haya sido modificada.

Tras analizar todas las aplicaciones se ha identificado que el 100% de las mismas están firmadas digitalmente. Este hecho no supone ninguna sorpresa, ya que los IDEs en el momento de la generación de una APK, generan así mismo un certificado en modo debug mediante las herramientas que incorpora Android SDK. Sin embargo, hay un aspecto referente a los certificados de las APKs que sí que es significativo: como se ha indicado anteriormente, los certificados no pasan obligatoriamente el control de una autoridad certificadora o CA, sino que pueden ser autofirmados, por lo que no supone un mecanismo de seguridad especialmente seguro.

Continuando con la documentación de Android: «*Android uses this certificate to identify the author of an app, and the certificate does not need to be signed by a certificate authority. Android apps often use self-signed certificates*», es decir, «*Android utiliza este certificado para identificar el autor de una aplicación, y el certificado no necesita ser firmado por una autoridad certificadora. Las aplicaciones de Android utilizan a menudo certificados autofirmados*».

En este caso, la inmensa mayoría de las aplicaciones analizadas poseen un certificado autofirmado. Únicamente 191 de la totalidad, poseen información distinta en el emisor «*Issuer*» con respecto al usuario final para el que se emite el certificado «*Subject*», lo que supone un 0,02%. Este hecho tampoco asegura que el certificado sea confiable ya que la información tanto del emisor del certificado como la del usuario final puede ser falseada fácilmente en el momento de la generación del mismo mediante la utilidad [Jarsigner](#) o alguna similar. Algunos de los nombres que se han utilizado como nombre común del emisor del certificado (CN) en el momento de la generación de los certificados son Bill Gates o Steve Jobs.

Google no fomenta la generación de certificados validados mediante autoridades certificadoras ya que solicita lo siguiente: «*If you plan to publish your apps on Google Play, the key you use to sign these apps must have a validity period ending after 22 October 2033*», es decir, «*Si planea publicar sus aplicaciones en Google Play, la clave utilizada para firmar estas apps debe tener un período de validad posterior al 22 de Octubre del 2033*».

La mayoría de las autoridades certificadoras no emiten certificados con períodos de validez tan largos, siendo lo habitual un período de 5 años de validez para usuarios finales. Esta política se contradice con los requisitos establecidos en el Foro de Autoridades de Certificación y Navegadores ([CAB forum](#)), en 2012, del que Google forma parte, y en el que se definió que, a partir del abril de 2015, los certificados emitidos por CAs a usuarios finales tendrían una validez máxima de 39 meses, en lugar de 60 meses.

5.3. APKS MONETIZADAS

Muchos desarrolladores incluyen publicidad en sus aplicaciones con el fin de rentabilizarlas y así obtener beneficio económico. La monetización es algo habitual, y en su justa medida no supone una experiencia negativa para los usuarios [1].

Existen diferentes modelos de negocio a la hora de monetizar una aplicación, entre los que destacan los siguientes:

- **Apps de pago:** es un modelo que está cayendo en desuso con el paso del tiempo ya que los usuarios tienden a preferir instalar aplicaciones gratuitas. Pese a ello, todavía genera un gran volumen de ganancias.
- **Apps gratuitas con publicidad:** Consiste en incluir publicidad de terceros.
- **Apps «freemium»:** son aquellas que ofrecen alguna funcionalidad adicional a cambio de dinero. Normalmente no llevan publicidad ya que se sustentan en las compras que realizan los propios usuarios. Es el modelo de negocio que genera más ganancias para los desarrolladores y está pensado para aquellas aplicaciones, normalmente juegos, con un gran volumen de usuarios.
- **Suscripción:** son aquellas aplicaciones que cobran periódicamente cierta cantidad de dinero y a cambio permiten su utilización al usuario que realiza el pago.

El modelo más utilizado tanto por aplicaciones normales como fraudulentas es el de «*Apps gratuitas con publicidad*», el cual funciona como se indica en Ilustración 6.



Ilustración 6: *Modelo de Apps gratuitas con publicidad*

A continuación, se muestra un ejemplo de comunicación entre una APK y un servidor correspondiente a una red de publicidad:

- La aplicación realiza la siguiente petición:

http://ads.wapx.cn/action/miniad/ad?app_id=106185f96d1a4106166a4f14aede5d9f&udid=XXXXXXXXXXXXXX&imsi=XXXXXXXXXXXXXX&net=internet&base=wapx.cn&app_version=2.8.9&sdk_version=1.8.8&device_name=Nexus%20S&device_brand=google&y=c98b7c327eef623890dda4b9f5a26de9&device_type=android&os_version=4.0.4&country_code=US&language=en&act=com.windrey.cuteroot=true&channel=mumayi&device_width=480&device_height=800&at=1349849007891

Como se puede observar, en la petición se incluyen una serie de parámetros correspondientes a información del terminal con el fin de personalizar la publicidad que va a ser mostrada en el mismo: *device_name*, *device_brand*, *os_version*, *country_code*, *language*, etc.

- El servidor devuelve una respuesta con información, en este caso, en formato XML. En ella, se incluye, entre otra información, las URLs de las imágenes correspondientes a la publicidad personalizada:



Ilustración 7: Ejemplo de publicidad en Android

El problema surge cuando un desarrollador inunda su aplicación con publicidad, cuando se monetizan aplicaciones sin ninguna utilidad y únicamente con el fin de conseguir enriquecer al desarrollador de la aplicación, o incluso cuando se utiliza este sistema para propagar malware mediante anuncios que redirigen a este tipo de amenazas y comprometer así la seguridad de los dispositivos.

Es importante tener en cuenta que las redes de publicidad generan anualmente un volumen de miles millones de dólares. Es por ello que son un «target» muy importante para los ciberdelincuentes, como se muestra en el informe [«The Bot Baseline: Fraud in Digital Advertising»](#) de White Ops, en el que se indica que el 11% de los anuncios en páginas web son vistos por bots y el 23% de las visitas que recibe la publicidad en vídeo llega de botnets, siendo

un 1% proveniente de redes de móviles. Este hecho provocará unas pérdidas estimadas de 6.000 millones de dólares de los anunciantes para 2015.

Así mismo, todas las librerías de monetización afectan en mayor o menor medida directamente a la privacidad de los usuarios ya que solicitan habitualmente una serie de permisos con el fin de poder acceder a información personal de los mismos y establecer así anuncios personalizados. Del mismo, como ha sido demostrado por investigadores de [FireEye](#), gran parte de este tipo de librerías de monetización son vulnerables y exponen directamente la información de los usuarios frente a ataques «*Man In The Middle*».

Existen diferentes métodos para identificar de manera automática las librerías de monetización que incorporan las APKs, como por ejemplo «[AdDetect: Automated detection of Android ad libraries using semantic analysis](#)» [2]. Para llevar a cabo el estudio, se han descompilado las aplicaciones y se han comparado todos los «*packages*» que integran con un listado de alrededor de 150 librerías que se han identificado previamente mediante un trabajo de investigación como las más utilizadas, es decir, se ha analizado la frecuencia de uso de todas las librerías de monetización identificadas. En la Ilustración 8 se indican los datos extraídos del análisis de las aplicaciones:

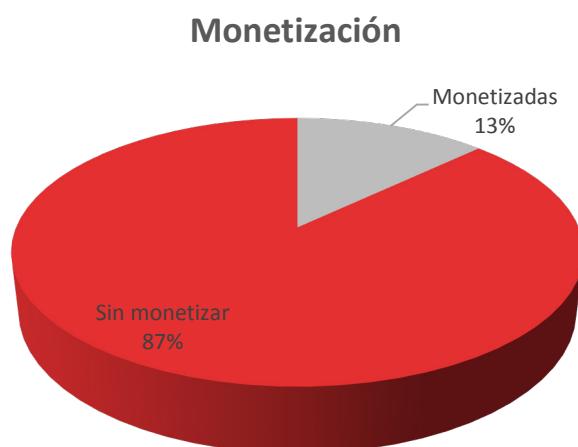
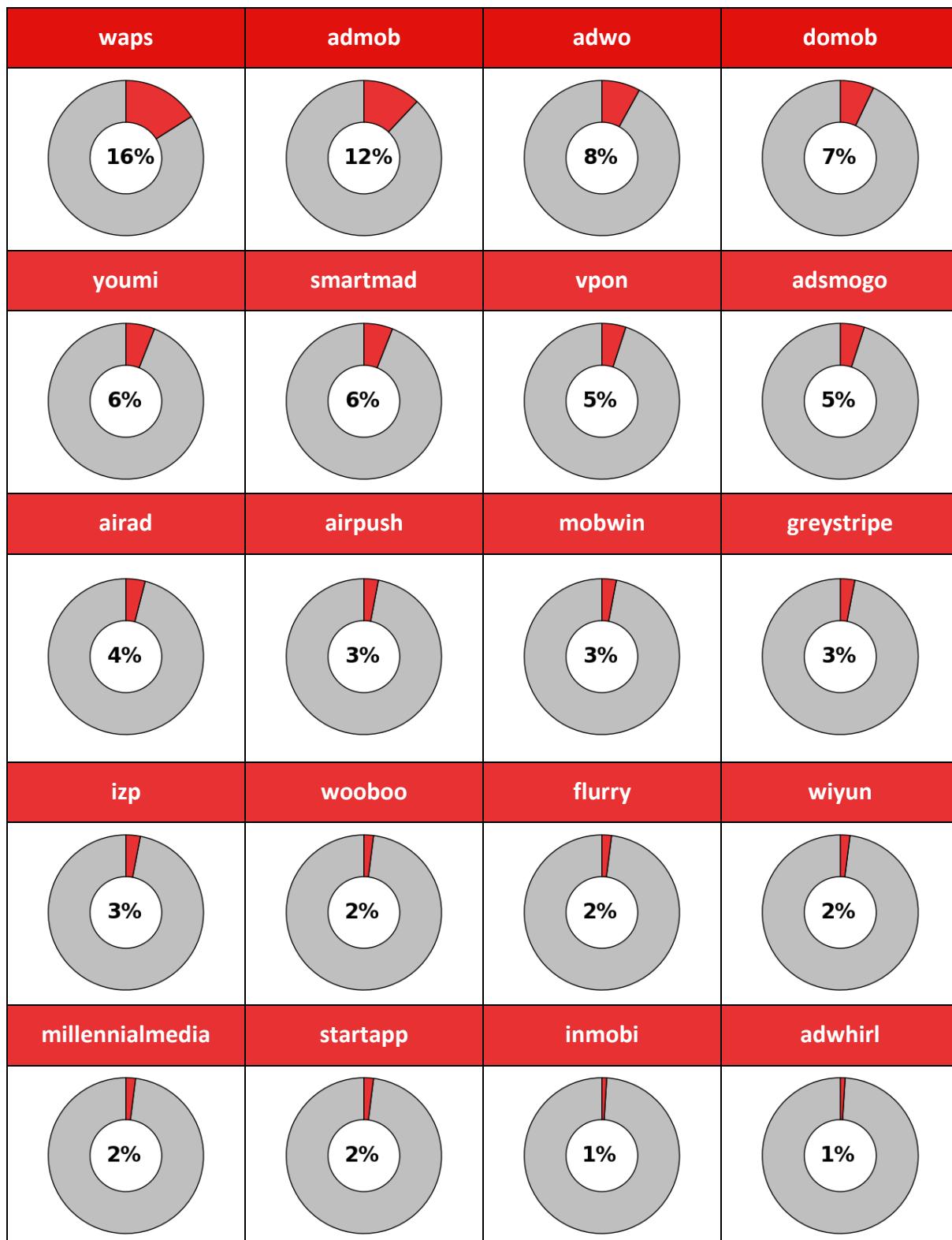


Ilustración 8: Monetización

El 13% de las aplicaciones analizadas incorpora alguna librería de monetización.

De la totalidad de librerías identificadas, únicamente 55 son distintas, siendo aquellas que más se repiten las siguientes:



Como reflejan los datos, los creadores de aplicaciones maliciosas no se decantan especialmente por un servicio en concreto, sino que acostumbran a utilizar diferentes redes de publicidad para obtener beneficio económico. Así mismo, acostumbran a integrar varias librerías de monetización en una misma APK. Las 9.840 APKs monetizadas incorporan un total de 26.894 librerías, lo que supone una media de 2,73 librerías por APK.

Existen comparativas de este tipo de librerías desde diferentes puntos de vista: «[A Comprehensive Guide To All Mobile Ad Networks](#)» [3], «[Unsafe Exposure Analysis of Mobile In-App Advertisements](#)» [4] o «[Investigating User Privacy in Android Ad Libraries](#)» [5].

A continuación, se muestra una tabla comparativa en cuanto al nivel de riesgo potencial para los usuarios basada en los requisitos de permisos que solicitan las últimas versiones del TOP 10 de librerías utilizadas, lo que está directamente relacionado con la peligrosidad y el nivel de intromisión de las mismas, y que afecta por tanto a la privacidad de los usuarios:

Librería	Nivel de intromisión
adsmogo	Alto
smartmad	
airad	
vpon	
domob	
youmi	Medio
airpush	
waps	
adwo	Bajo
admob	

Se puede visualizar el análisis de permisos en detalle en «*ANEXO 1 – Comparativa de librerías de monetización*».

Como se ha indicado, se han identificado 55 librerías de monetización distintas, si bien algunas de ellas incluían varias versiones, siendo las que más se repiten las que se muestran en la Ilustración 9:

Top 20 versiones de librerías de monetización

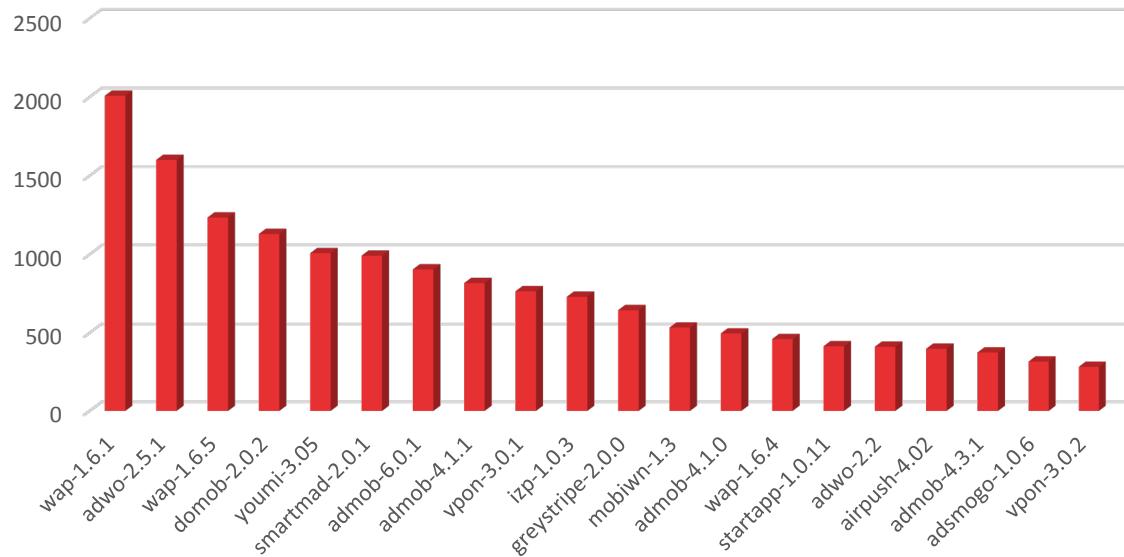


Ilustración 9: Top 20 versiones de librerías de monetización

Con el fin de determinar el riesgo para la privacidad que puede suponer para los usuarios la utilización de aplicaciones que integren librerías de este tipo se ha realizado un análisis de las mismas. Para ello, se han analizado sus correspondientes SDK «*Software Development Kit*» para identificar qué información recopilan. De manera genérica, las librerías de monetización obtienen de los dispositivos y del usuario los siguientes datos:

- Datos referentes al dispositivo móvil: número de teléfono, idioma, fabricante, modelo, número de serie del dispositivo, resolución, versión de Android, MAC, IP, IMEI o MEID, ubicación, información del GPS, tipo de conexión, operador de la red, preferencias de conexión, funcionalidades activas (NFC, Cámara, Bluetooth), listado de aplicaciones que se descargan o instalan en el dispositivo junto con su respectiva versión, cómo y cuándo se utilizan las aplicaciones instaladas, historial de navegación.
- Datos referentes al usuario: nombre de usuario, fecha de nacimiento, género, estudios, creencias religiosas, estado civil, orientación sexual, etnia, ideología política, código postal, edad, dirección de correo electrónico, aficiones, etc. Incluso en ocasiones recopilan la lista de contactos con su respectivo nombre, número de teléfono, email y fotografía, el listado de llamadas, calendario, histórico de mensajes de texto, identificadores de otros servicios como por ejemplo redes sociales, etc. Muchos de estos datos son solicitados al usuario mediante algún método como una encuesta, o para poder participar en algún sorteo, etc. Sin embargo, en algunos casos, otros son accedidos directamente sin interacción ni conocimiento por parte del usuario.

Mediante esta práctica se llegan a recopilar más de 50 aspectos que permiten generar el perfil de un usuario. A continuación, se indica un ejemplo del perfil básico, con datos falsos, que se puede obtener mediante la información que recopilan este tipo de librerías:

Josines Segurola, nacido el 13/01/1981 (34 años), residente en Guardo, casado y con dos hijos, católico, email: josinessegurola@gmail.com, licenciado en matemáticas, trabaja en un instituto, le gustan los deportes en general y las películas de Chuck Norris, dispone de un Nexus 5 no rootead, con Android 5.0 y su número de teléfono es 69XXXXXX, es usuario de diferentes servicios cloud y de varias redes sociales, principalmente facebook y su nombre de usuario es josinessegurola, tiene contratada una tarifa de datos con el proveedor XXXXX, visita de manera habitual páginas de deportes y periódicos de tirada nacional, suele llegar puntual al trabajo y recorre paseando siempre el mismo camino, su dirección es Calle XXXXX, número 5. Los fines de semana no los pasa en su domicilio habitual, sino en un pueblo a 20km de Guardo.

Toda esta información puede ser compartida con terceros en muchos casos por lo que el usuario pierde el control sobre la misma. Además, la información es almacenada en los servidores de las respectivas empresas (normalmente ubicados fuera de España e incluso en países sin una sólida legislación referente a la protección de datos), y en el caso de que éstos sufran algún incidente de seguridad, dicha información puede verse expuesta y utilizada por delincuentes para llevar a cabo actos delictivos como estafas o extorsión.

Es posible consultar la información que recolectan y para qué la utilizan en la política de privacidad disponible en sus respectivas páginas web, aunque es bastante común que referencien dicha información de manera superficial y que no la detallen.

En algunos casos, las compañías de antivirus detectan automáticamente mediante detecciones genéricas todas las aplicaciones que incluyen algunas de las librerías de monetización debido al elevado nivel de intrusión.

Actualmente, Android incorpora un sistema de permisos a nivel de aplicación, de modo que al otorgar permisos a las mismas, éstos son totalmente accesibles por las librerías que implementan, no habiendo una separación de privilegios. En ocasiones, se solicitan permisos que en realidad no requiere la aplicación, pero si la librería de monetización para su correcto funcionamiento lo que provoca [aplicaciones sobreprivilegiadas](#) [6]. Como se ha descrito, todo esto implica un gran riesgo para los usuarios, principalmente relacionado con la privacidad. Existen diferentes aproximaciones que buscan mitigar o solucionar el problema, como «[AdSplit: Separating smartphone advertising from applications](#)» [7], «[AdDroid: Privilege Separation for Applications and Advertisers in Android](#)» [8] o «[AFrame: Isolating Advertisements from Mobile Applications in Android](#)» [9]. Sin embargo, recae en Google la responsabilidad de implementar las mejoras necesarias para establecer una separación de permisos entre los componentes de una aplicación y que esto no suponga un problema para los desarrolladores.

5.4. SDKVERSION

Android es un sistema operativo fragmentado en varias versiones. A lo largo de los últimos años Google ha trabajado para reducir este problema con medidas como por ejemplo «*Google Play Services*», que permite actualizar las aplicaciones de Google y de Google Play sin necesidad de una actualización completa del sistema operativo. Aun así, como se observa en la Ilustración 10, los usuarios de Android todavía están repartidos en 5 versiones del sistema operativo, sin tener en cuenta los datos referentes a Android 5.0 «*Lollipop*», que todavía no están disponibles en la propia web de Android.

Version	Codename	API	Distribution
2.2	Froyo	8	0.4%
2.3.3 - 2.3.7	Gingerbread	10	7.8%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	6.7%
4.1.x	Jelly Bean	16	19.2%
4.2.x		17	20.3%
4.3		18	6.5%
4.4	KitKat	19	39.1%

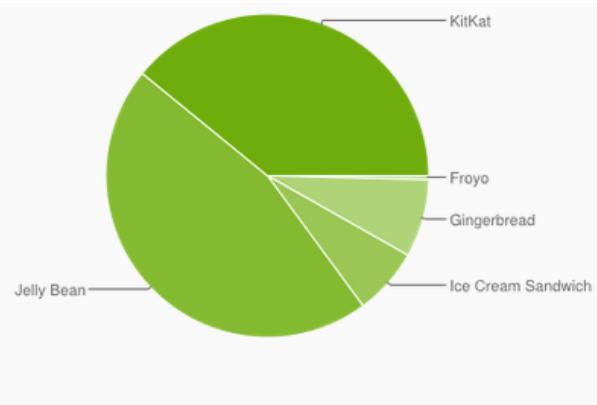


Ilustración 10: Distribución de las versiones de Android

Este hecho supone que el rendimiento de las aplicaciones sea distinto dependiendo de la versión de Android en la que se ejecute, o incluso el acceso a distintas características o funcionalidades del dispositivo. Es por todo ello que las aplicaciones en Android se desarrollan estableciendo diferentes parámetros como «*MinSDKVersion*», «*MaxSDKVersion*» y «*TargetSDKVersion*». Estos valores corresponden a la versión mínima de Android compatible con la aplicación, a la versión máxima de Android compatible con la aplicación y a la versión óptima, y normalmente con la que la aplicación ha sido testeada.

En la Ilustración 11 se indican los datos extraídos del análisis de las aplicaciones:

MinSDKVersion

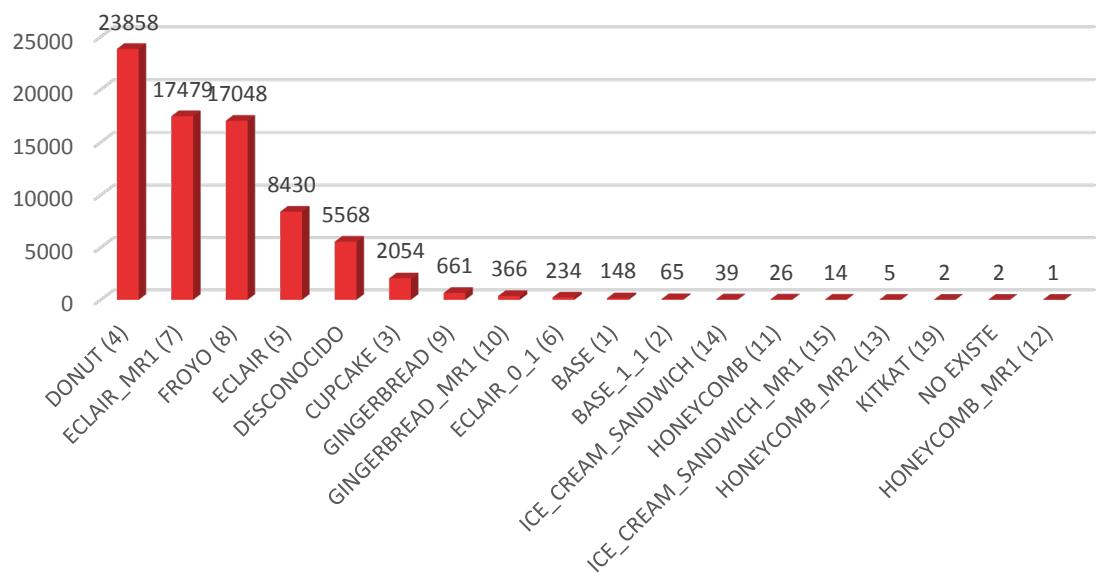


Ilustración 11: *MinSDKVersion*

De manera genérica, los creadores de malware para Android buscan que sus aplicaciones sean compatibles con el mayor número de dispositivos posibles por lo que establecen valores bajos en «*MinSDKVersion*». El 68,19% de las aplicaciones son compatibles con versiones de Android que ya no están en uso. Justamente todo lo contrario es el caso de las 2 únicas aplicaciones que se han identificado cuya versión mínima compatible es KitKat, lo que supone que no afecten a casi un 70% de la cuota de mercado.

En lo que se refiere a la versión mayor de Android compatible con las aplicaciones o «*MaxSDKVersion*», no se han obtenido datos suficientes como para reflejarlos. Este hecho es algo lógico ya que en la [documentación de Android](#) se recomienda lo siguiente: «Warning: Declaring this attribute is not recommended», es decir, «Advertencia: No se recomienda declarar este atributo».

En lo que se refiere a la versión de Android óptima para las que se han programado las aplicaciones, «*TargetSDKVersion*» los datos son bastante reveladores, como se puede observar en la Ilustración 12:

TargetSDKVersion

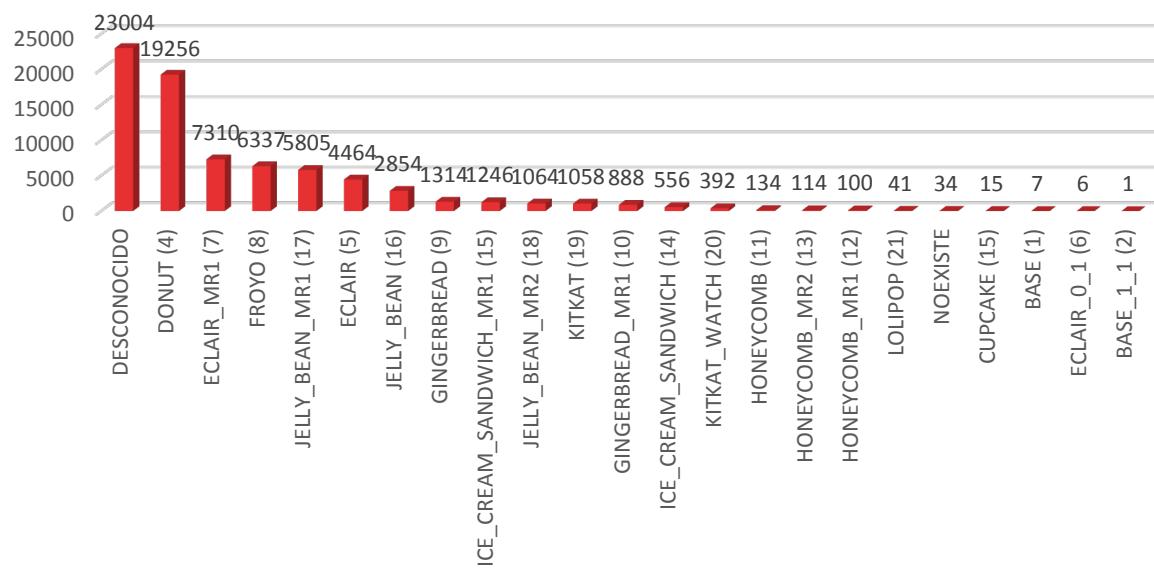


Ilustración 12: TargetSDKVersion

Un 34,96% de las aplicaciones analizadas tienen como «*TargetSDKVersion*» versiones de Android obsoletas. A este hecho hay que añadirle que un 8,34% está enfocada en Froyo, una versión de Android bastante antigua con una cuota de mercado del 0,6% y que probablemente desaparezca en un período no demasiado largo de tiempo. Todo esto hace pensar que las muestras no son muy nuevas. Sin embargo, todas ellas han llegado por primera vez al sistema de análisis de Virustotal a partir de julio del 2014, por lo que previsiblemente sí que son recientes, al menos en lo que a su identificación por el sistema se refiere. En el apartado «Tiempo de exposición a la amenaza» se ha analizado en profundidad este aspecto.

5.5. TIEMPO DE EXPOSICIÓN A LA AMENAZA

Para este informe hemos definido como tiempo de exposición a la amenaza al tiempo transcurrido desde que se ha creado una aplicación hasta que ha sido remitida al servicio de VirusTotal.

En este aspecto, no se tienen en cuenta las posibles heurísticas o detecciones genéricas de los antivirus, si bien este tipo de detecciones y protecciones no están muy extendidas en los antivirus para plataformas móviles. Tampoco se considera si las muestras han llegado previamente a las compañías a través de sus clientes o de cualquier otro modo, ya que no es posible determinar la fecha de inclusión de las correspondientes firmas en sus sistemas de detección. Para aportar valor a este dato, es importante tener presente que muchas de las

compañías de antivirus utilizan el servicio para nutrir sus sistemas de detección (ficheros de firmas, detección por nube, etc.), por lo que puede servir como una primera aproximación al tiempo transcurrido desde su creación hasta que es detectado por un gran número de los motores antivirus.

Con el fin de determinar este tiempo, se ha procedido a analizar las aplicaciones comprobando la fecha de última modificación del fichero *classes.dex*, que como se ha indicado con anterioridad corresponde al código fuente de la aplicación compilado. Es importante tener en cuenta que dicha fecha corresponde a la fecha local en el que se ha compilado y que por tanto puede haber sido alterada por el administrador del sistema, pese a que esta sea una práctica que a priori no parece muy habitual.

Una vez obtenidas las fechas de última modificación de las aplicaciones, se han comparado con respecto a la primera vez que fueron remitidas a Virustotal. Atendiendo a ese criterio se han identificado los siguientes niveles de riesgos en función del tiempo de exposición:

- Aplicaciones cuya diferencia entre fecha de compilación y fecha de primer envío a Virustotal es negativa → Error (Indica que la fecha en el sistema en el que se ha compilado la APK es errónea, no se puede determinar si accidental o deliberadamente).
- Aplicaciones cuya diferencia entre fecha de compilación y fecha de primer envío a Virustotal es menor de 1 día → Muy bajo.
- Aplicaciones cuya diferencia entre fecha de compilación y fecha de primer envío a Virustotal es menor de 7 días → Bajo.
- Aplicaciones cuya diferencia entre fecha de compilación y fecha de primer envío a Virustotal es menor de 31 días → Medio.
- Aplicaciones cuya diferencia entre fecha de compilación y fecha de primer envío a Virustotal es menor de 93 días → Alto.
- Aplicaciones cuya diferencia entre fecha de compilación y fecha de primer envío a Virustotal es mayor que o igual de 93 días → Muy alto.

A partir de esta clasificación se han obtenido los datos de la Ilustración 13:

Tiempo de exposición

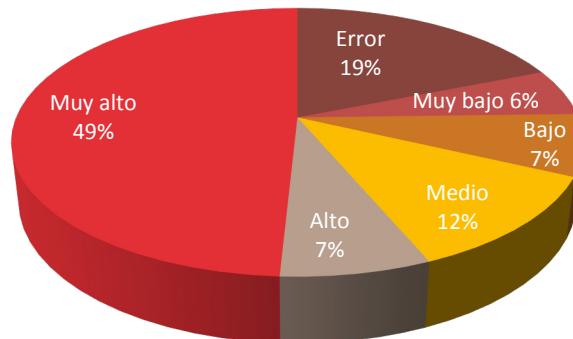


Ilustración 13: Tiempo de exposición

- Casi la mitad de las aplicaciones tienen un tiempo de exposición muy alto, de más de 3 meses.
- El 13% de las aplicaciones tienen un tiempo de exposición bajo o muy bajo, menor a 1 semana.
- En el 19% de las aplicaciones (14.384) se ha falseado la fecha de compilación del sistema.

5.6. PLATAFORMA DE DESARROLLO

Así mismo, se ha comprobado el sistema en el que se han desarrollado las APKs. Los resultados obtenidos se muestran en la Ilustración 14:



Ilustración 14: Plataforma de desarrollo

La inmensa mayoría de las aplicaciones analizadas han sido desarrolladas en entornos Windows, únicamente un 2% han sido desarrolladas en un entorno Linux. Y, según parece, los desarrolladores de aplicaciones maliciosas no son demasiado partidarios de OSX para llevar a cabo sus desarrollos.

5.7. NOMBRE DEL PACKAGE

Tomando como fuente de datos los «*package name*» de las aplicaciones analizadas, se ha comprobado cuáles son los más utilizados, lo que permite hacerse una idea de si hay familias de malware para Android especialmente activas y si los desarrolladores de malware hacen ligeras modificaciones sobre las APKs para saltarse las detecciones por hash de los antivirus. Los datos obtenidos se muestran en la Ilustración 15.



Ilustración 15: Package names

Además, mediante [NLTK](#), una librería para el procesamiento del lenguaje natural, se ha comprobado el número de veces que se repiten las palabras que forman los «*package name*», de modo que se han obtenido los resultados de la Ilustración 16.

Cadenas más utilizadas en los Package Names

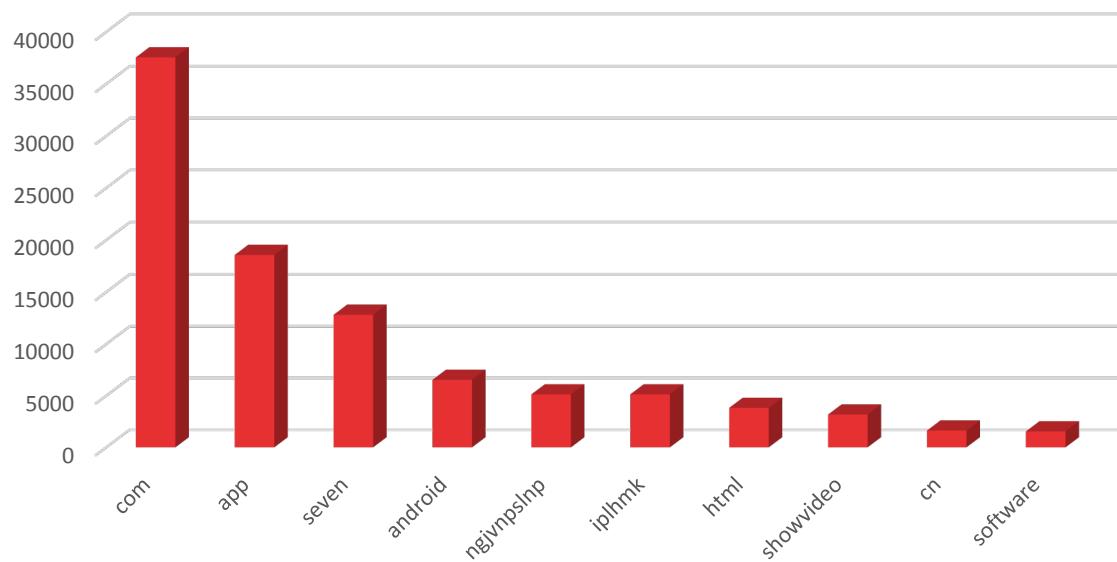


Ilustración 16: Cadenas más utilizadas en los Package names

Algunos de los términos es lógico que aparezcan como los más utilizados ya que son empleados de manera habitual por convención. Otros sin embargo, no son habituales y pueden ser utilizados para la detección de aplicaciones maliciosas.

Así mismo, se han identificado términos que permiten hacerse una idea de la temática de las aplicaciones como por ejemplo: showvideo, game, livewallpaper, playporn, pornow, smsreg, etc, y otros que nos resultan familiares por el artículo [«¿Sirven algunas aplicaciones de Android para estafar?»](#).

5.8. PACKAGE NAME EXISTENTES EN GOOGLE PLAY

Como se indica al comienzo del documento, el informe [«State of Mobile App Security»](#) de Arxan revela que el 97% del TOP 100 de aplicaciones de pago y el 80% del TOP 100 de aplicaciones gratuitas de Android tienen versiones modificadas correspondientes a malware, distribuidas por un gran número de mercados alternativos.

Tomando como origen los «*package name*» de las aplicaciones analizadas, se ha comprobado su existencia en Google Play con el fin de comprobar cuántas de las aplicaciones analizadas son probablemente aplicaciones originales que se han modificado para ser maliciosas. Puede haber alguna coincidencia accidental, es decir que distintas aplicaciones tengan el mismo «*package name*». Sin embargo, habitualmente son lo suficientemente característicos como para que esto no suceda. En la Ilustración 17 se pueden observar los datos obtenidos.

Package Name existentes en Google Play

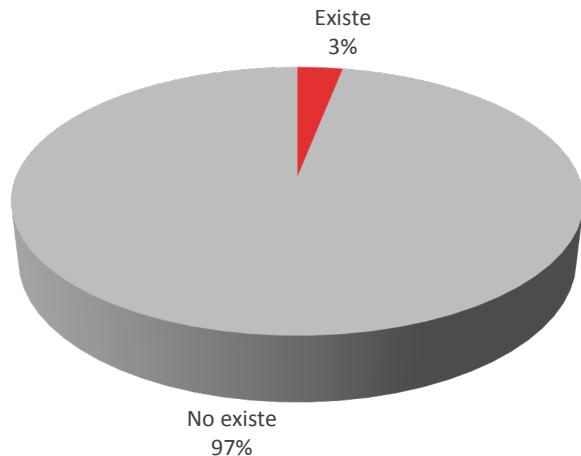


Ilustración 17: Package Name existentes en Google Play

Únicamente un 3% de las aplicaciones analizadas son previsiblemente aplicaciones alojadas en Google Play que han sido descargadas y modificadas para ser maliciosas.

Así mismo, se ha comprobado qué tipo de aplicaciones son, es decir, dentro de qué categoría están organizadas en Google Play. De este modo, es posible realizar una pequeña aproximación sobre cuáles son el tipo de aplicaciones que más se plagan y se modifican para incluir código malicioso. Se han identificado aplicaciones pertenecientes a 40 categorías distintas, siendo top 10 el mostrado en la Ilustración 18.

Top 10 categorías

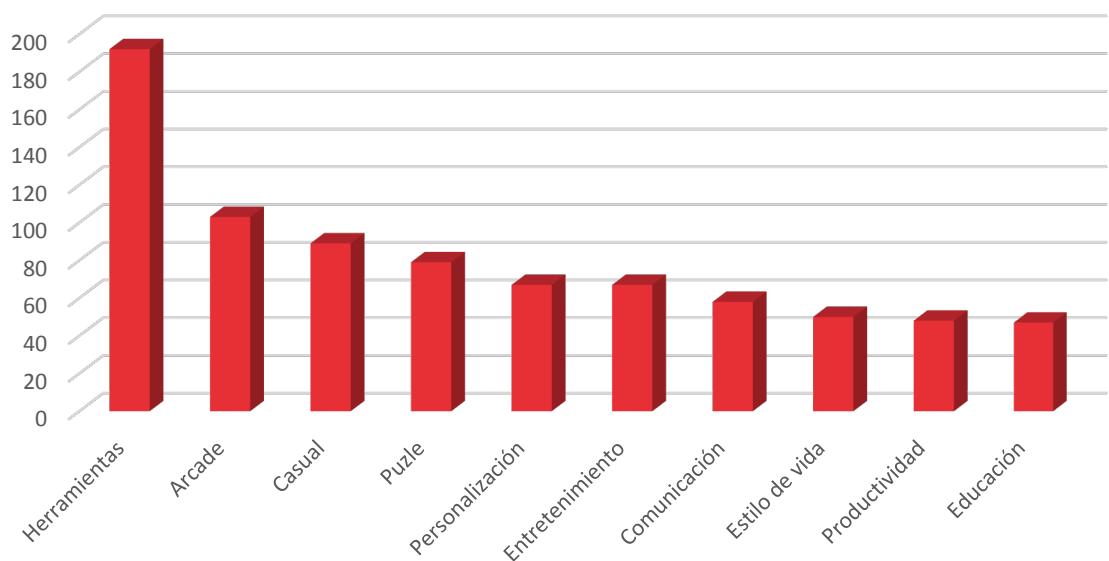


Ilustración 18: Top 10 categorías

Finalmente, se ha obtenido el número de descargas que tienen dichas aplicaciones con el fin de ver el impacto potencial de las aplicaciones maliciosas. Los datos obtenidos se pueden ver en la Ilustración 19.



Ilustración 19: Número de descargas de Google Play

El impacto potencial de las aplicaciones es muy variado: hay aplicaciones maliciosas con el mismo «package name» que otras disponibles en Google Play que únicamente se han descargado entre 1 y 5 veces y otras que lo han hecho más de 1.000.000.000, como ese el caso de Google Maps y de Youtube. Este hecho, puede dar pie a analizar en profundidad, principalmente en las aplicaciones con pocas descargas, si se trata de aplicaciones maliciosas que han evadido los controles del «bouncer» y están disponibles en el «market» o si son aplicaciones legítimas que han sido utilizadas como base para modificarlas, incluir componentes maliciosos y propagarlas por otros «markets».

5.9. PERMISOS

Android incorpora un [modelo de permisos](#) gracias al cual es posible limitar las operaciones que puede realizar una aplicación en el dispositivo móvil. En el momento de la instalación, se le informa al usuario de los permisos que solicita la misma, y que han sido previamente establecidos por el desarrollador en el fichero AndroidManifest.xml. Es necesario aceptar todos los permisos solicitados en el caso de querer instalar la aplicación, no es posible realizar una instalación con funcionalidades reducidas de modo que se puedan aceptar únicamente algunos permisos y rechazar otros.

Esto supone un riesgo ya que en muchos casos los desarrolladores añaden permisos adicionales, que no son necesarios para el funcionamiento de la aplicación, para obtener información como el listado de contactos o los lugares visitados mediante el seguimiento GPS, como demuestra el [informe](#) de la Agencia Española de Protección de Datos en el que se refleja 31% de las aplicaciones solicitan permisos excesivos para la funcionalidad que ofrecen.

Con el fin de intentar mejorar el modelo de permisos y para permitir al usuario tener un mayor control sobre la privacidad de sus datos, Google desarrolló una herramienta nativa conocida como «[App Ops](#)», disponible a partir de Android 4.3, la cual permitía a un usuario revocar los permisos que considerase oportunos a una aplicación. Sin embargo, esta aplicación fue eliminada con la actualización de Android 4.4.2. Es importante tener en cuenta que la revocación de algún permiso puede suponer la diferencia entre que una aplicación funcione o no, por lo que se debe ser especialmente cuidadoso en este aspecto.

Los permisos tienen distinto nivel de peligrosidad según las funciones que permitan realizar a la aplicación. Es por ello que son clasificados en cuatro grandes grupos «[Protection Level](#)». Así mismo, mediante este atributo es posible establecer qué aplicaciones tienen acceso al permiso:

- **Normal:** en teoría no suponen un riesgo real ni para el dispositivo, ni para el usuario. Es por ello que son aceptados automáticamente. Así mismo, en el momento de la instalación no son mostrados por defecto, es necesario expandirlos para poder visualizarlos. Algunos de los permisos ubicados dentro de esta categoría son: vibrar o permitir conocer el estado de la red.
- **Dangerous:** pueden suponer un riesgo real para el usuario, como por ejemplo tarificación de servicios SMS Premium o el acceso a información personal. Es por ello que en el momento de la instalación de la aplicación son mostrados al usuario y deben ser aceptados. Algunos de los permisos encuadrados dentro de esta categoría son: enviar SMS, recibir SMS o realizar llamadas.
- **Signature:** el permiso es accesible únicamente por aquellas firmadas con el mismo certificado que la aplicación que declara el permiso.
- **Signature / System:** es como «*Signature*», pero también puede ser utilizado por el sistema.

Así mismo, los permisos están clasificados dentro de varios grupos «[permission group](#)» dependiendo para qué sirvan, siendo los siguientes algunos de los más destacables:

- **android.permission-group.ACCOUNTS:** agrupa los permisos que solicitan acceso directo a las cuentas de Google.
- **android.permission-group.COST MONEY:** agrupa los permisos que pueden ser utilizados para hacer al usuario gastar dinero sin su participación directa.
- **android.permission-group.LOCATION:** agrupa los permisos que permiten acceder a la posición actual del usuario.

- **android.permission-group.PERSONAL_INFO:** agrupa los permisos que facilitan el acceso a información privada del usuario como contactos, eventos del calendario, emails, etc.

Se han identificado un total de 1.221.350 permisos, lo que supone un promedio de 16,1 permisos por aplicación.

En la Ilustración 20 se pueden ver los permisos más solicitados en conjunto por las aplicaciones:

Top 20 permisos utilizados

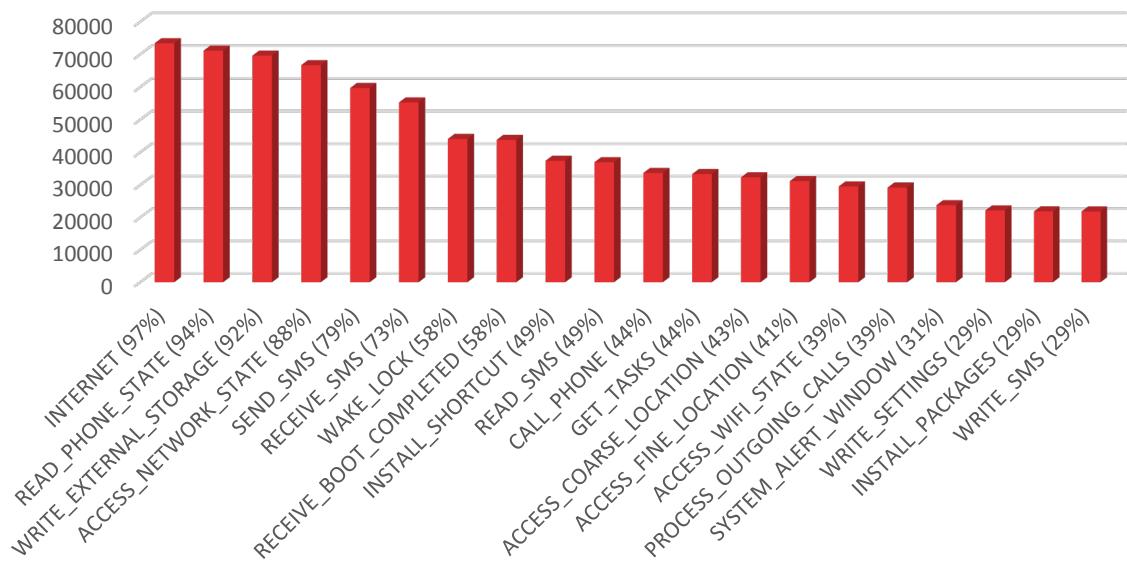
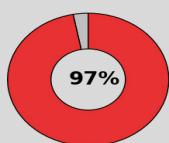
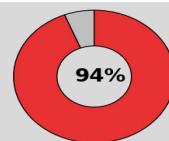


Ilustración 20: Permisos



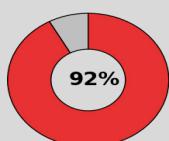
android.permission.INTERNET

- Permite a las aplicaciones acceso total a Internet.
- Dangerous



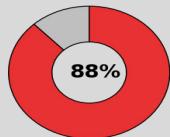
android.permission.READ_PHONE_STATE

- Permite el acceso de sólo lectura al estado del teléfono.
- Dangerous



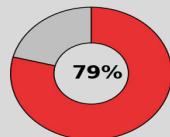
android.permission.WRITE_EXTERNAL_STORAGE

- Permite a una aplicación escribir en el almacenamiento externo.
- Dangerous



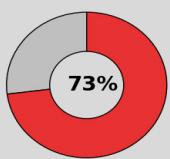
android.permission.ACCESS_NETWORK_STATE

- Permite que las aplicaciones accedan a información sobre las redes.
- Normal



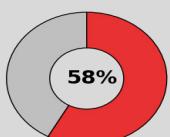
android.permission.SEND_SMS

- Permite a una aplicación enviar sms.
- Dangerous



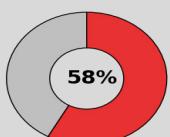
android.permission.RECEIVE_SMS

- Permite que una aplicación para monitorear los mensajes SMS entrantes, para grabar o realizar el procesamiento en ellos.
- Dangerous



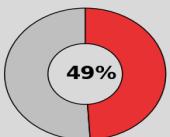
android.permission.WAKE_LOCK

- Permite el acceso a los bloqueos de energía para mantener el procesador durmiendo o mantener la pantalla apagada.
- Dangerous



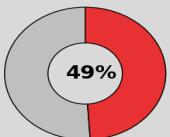
android.permission.RECEIVE_BOOT_COMPLETED

- Permite que una aplicación reciba el ACTION_BOOT_COMPLETED que se emite después de que el sistema termine de iniciarse.
- Normal



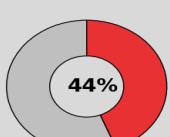
com.android.launcher.permission.INSTALL_SHORTCUT

- Permite a una aplicación instalar un acceso directo.
- signatureOrSystem



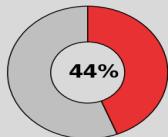
android.permission.READ_SMS

- Permite a una aplicación leer los sms.
- Dangerous



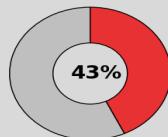
android.permission.CALL_PHONE

- Permite a una aplicación realizar llamadas de teléfono sin necesidad de usar el dialer de Android.
- Dangerous



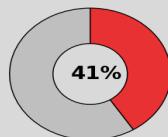
android.permission.GET_TASKS

- Permite a una aplicación obtener información sobre las tareas en ejecución actual o reciente.
- Dangerous



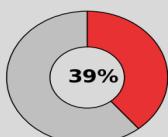
android.permission.ACCESS_COARSE_LOCATION

- Permite que una aplicación acceda a la ubicación aproximada derivada de fuentes de ubicación de red, como las torres de celulares y Wi-Fi.
- Dangerous



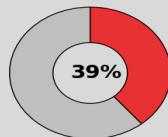
android.permission.ACCESS_FINE_LOCATION

- Permite que una aplicación acceda a la ubicación precisa derivada de fuentes de localización como el GPS, las torres celulares y Wi-Fi.
- Dangerous



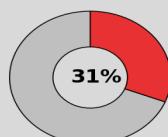
android.permission.ACCESS_WIFI_STATE

- Permite que las aplicaciones accedan a información sobre redes Wi-Fi.
- Normal



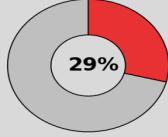
android.permission.PROCESS_OUTGOING_CALLS

- Permite que una aplicación visualice el número al que se está llamando con la opción de redirigir la llamada a un número diferente o cancelarla.
- Dangerous



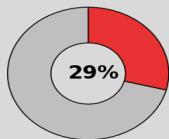
android.permission.SYSTEM_ALERT_WINDOW

- Permite que una aplicación abra ventanas utilizando el tipo TYPE_SYSTEM_ALERT, que se muestra en la parte superior de todas las demás aplicaciones.
- Dangerous



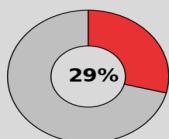
android.permission.WRITE_SETTINGS

- Permite que una aplicación lea o escriba la configuración del sistema.
- Dangerous



android.permission.INSTALL_PACKAGES

- Permite que una aplicación instale paquetes.
- signatureOrSystem



android.permission.WRITE_SMS

- Permite a una aplicación escribir sms.
- Dangerous

En base a los datos estadísticos se obtienen las siguientes conclusiones sobre los permisos utilizados:

- El 99,93% de las aplicaciones solicitan al menos un permiso catalogado como peligroso.
- El 97% solicitan acceso a ilimitado Internet.
- El 94% solicitan acceso a información sensible.
- Al menos el 79% pueden suponer cargos adicionales a la factura de los usuarios. Teniendo en cuenta que los servicios de tarificación adicional SMS PREMIUM de manera habitual informan que el coste es de 1,45€ por SMS recibidos, «Máx 25 sms/mes», lo que puede suponer al usuario un coste económico de hasta 36,25€ al mes por usuario, las aplicaciones analizadas pueden generar un beneficio económico de más de 2 millones de euros al mes, sin tener en cuenta otros aspectos como la venta de información confidencial, etc.
- El 58% se autoejecutan al iniciar el sistema.

De los 20 permisos más solicitados por las aplicaciones, el 15% corresponden a permisos con «*ProtectionLevel*» «*normal*», el 75% «*dangerous*» y el 10% «*signatureOrSystem*». Es por ello que es muy importante fijarse en los permisos que solicita una aplicación en el momento de la instalación, ya que pueden resultar bastante significativos. Sin embargo, este control, al recaer sobre el usuario, supone un riesgo principalmente por la credulidad, confianza y desconocimiento de gran parte de los mismos.

El nuevo sistema de permisos de Google Play aumenta notablemente la peligrosidad que supone la aceptación de los mismos. Mediante este sistema, en el momento de la actualización se aceptan de manera implícita, sin interacción por parte del usuario, permisos pertenecientes al mismo grupo que alguno que haya sido previamente aceptado y no se detallan los mismos en el momento de la instalación desde Google Play, sino que se muestra una explicación genérica referente al grupo al que pertenecen.

Así mismo, se ha comprobado la validez de los permisos comparando éstos con los indicados en la [documentación](#) de Android con el fin de realizar una aproximación acerca de los permisos mal declarados, principalmente provocado por una sintaxis incorrecta, y de los que son personalizados.

- Un 5,19% del total de los permisos declarados corresponden a permisos personalizados o mal declarados.
- Un 43,49% de las aplicaciones analizadas contienen permisos personalizados o mal declarados.
- El 13,70% de estos permisos tienen relación directa con el servicio [Google Cloud Messaging for Android](#), el servicio a través del cual un desarrollador puede establecer comunicación bidireccional entre un servidor y los clientes de la aplicación.

5.10. SERVICES

Los «services» o servicios son componentes de una aplicación que pueden realizar, en segundo plano, operaciones de larga duración sin necesidad de interacción por parte del usuario. Se han identificado un total de 201.826 «services», lo que supone una media de 2,66 servicios por aplicación. En la Ilustración 21 se muestran los servicios más declarados:

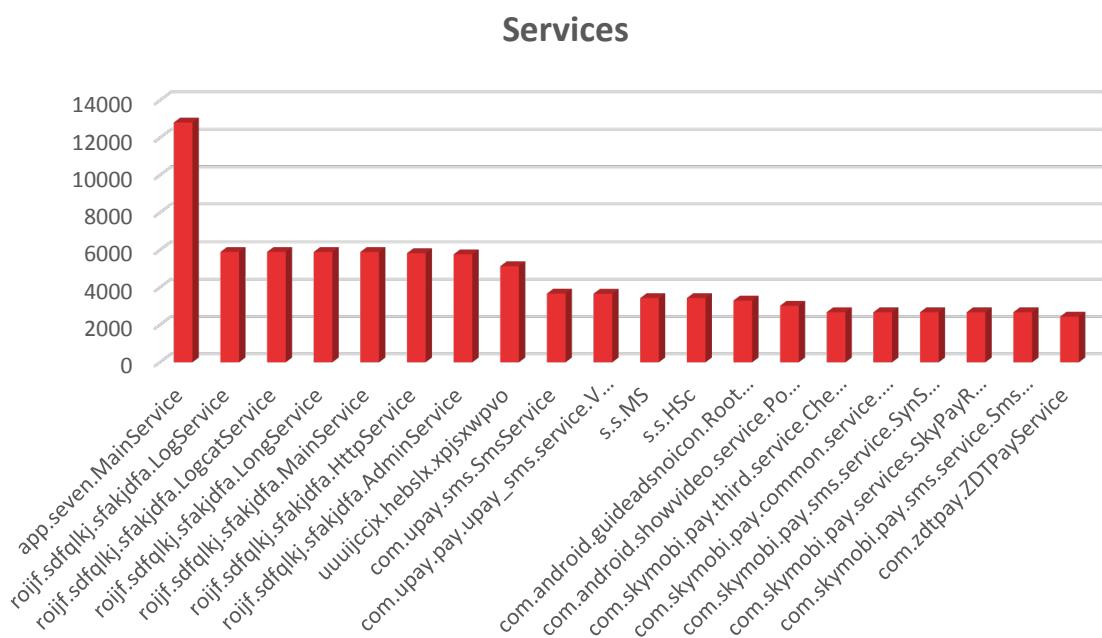


Ilustración 21: Services más declarados en las apps analizadas

5.11. RECEIVERS

Los «receivers» o receptores son componentes de una aplicación que permiten la recepción de acciones que son transmitidas por el sistema o por otras aplicaciones, incluso cuando otros componentes de la aplicación no están funcionando. En la Ilustración 22 se muestra el porcentaje de aplicaciones analizadas que incluyen este tipo de mecanismos.

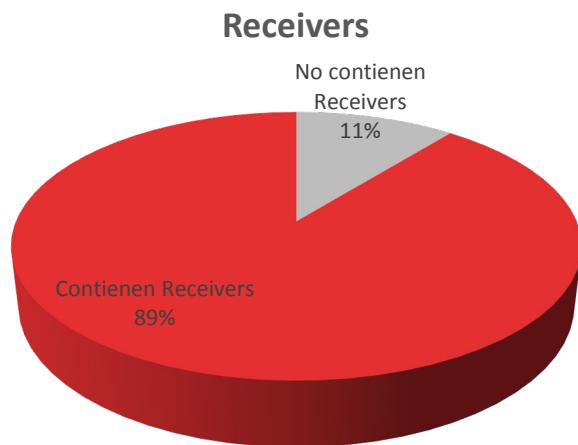


Ilustración 22: Porcentaje de las apps analizadas que contienen Receivers

Se han identificado un total de 228.951 receptores, lo que supone una media de 3,01 «receivers» por aplicación. Entre todos los identificados, destacan algunos como los relacionados con mensajes de texto recibidos, los cuales, tienen de manera habitual una relación directa con la suscripción a servicios SMS Premium, siendo el proceso como a continuación se concreta:

1. Un usuario instala una aplicación, aceptando previamente los permisos solicitados, siendo uno de ellos el de envío de mensajes de texto.
2. La aplicación envía, sin conocimiento del usuario, un mensaje de suscripción a un servicio de SMS Premium.
3. Este tipo de servicios requieren un código de confirmación que es recibido por el usuario también mediante mensaje de texto.
4. Un receptor expresamente diseñado para ello, identifica la recepción del código, lo lee, elimina el mensaje, y envía otro mensaje al servicio confirmando la contratación del mismo mediante dicho código. De este modo, un usuario se ve suscrito a un servicio de tarificación especial sin su conocimiento ni consentimiento.

5.12. PROVIDERS

Los «providers» o proveedores son componentes de una aplicación gracias a los cuales es posible compartir información con otras aplicaciones. En la Ilustración 23 se muestra el porcentaje de aplicaciones que incorporan este tipo de mecanismos.



Ilustración 23: Porcentaje de las apps analizadas que utilizan Providers

Se han identificado un total de 2010 «*providers*», lo que supone una media de 0,03, es decir, su uso no está demasiado extendido en las aplicaciones maliciosas, algo totalmente razonable ya que no suele ser necesario que este tipo de aplicaciones implementen mecanismos para compartir su información, sino que más bien, en algunos casos, buscan explotar una configuración incorrecta de otras aplicaciones para obtener información sensible.

5.13. ACTIVITIES

Las «*Activities*» o actividades corresponden al interfaz de usuario de una aplicación, de tal modo que éste pueda interactuar con la misma.

El 0,30% de las aplicaciones analizadas no incluyen ninguna «*Activity*», es decir, no disponen de ningún interfaz gráfico. Este hecho, no está demasiado claro si permite a los usuarios que se den cuenta de que algo no va bien con la aplicación, o si por el contrario les hace pensar que no se ha instalado correctamente y no le dan importancia como sucede con malware para otros entornos como es el caso de Windows. El resto de aplicaciones, el 99,70% incluyen un total de 409.819 «*activities*», lo que supone una media de 5,39 «*activities*» por aplicación.

5.14. COMPLEJIDAD

La complejidad de una APK se ha calculado a partir del número de «*Permisos*», «*Services*», «*Receivers*», «*Providers*», «*Activities*» y “*Ads*” que tienen las propias APKs.

Para ello, se han utilizado los siguientes rangos:

Complejidad	Rango
Baja	0 – 10

Media	11 – 25
Alta	26 – 50
Muy alta	> 50

A partir de toda la información obtenida y en base a los rangos indicados, se han obtenido los datos de la Ilustración 24.

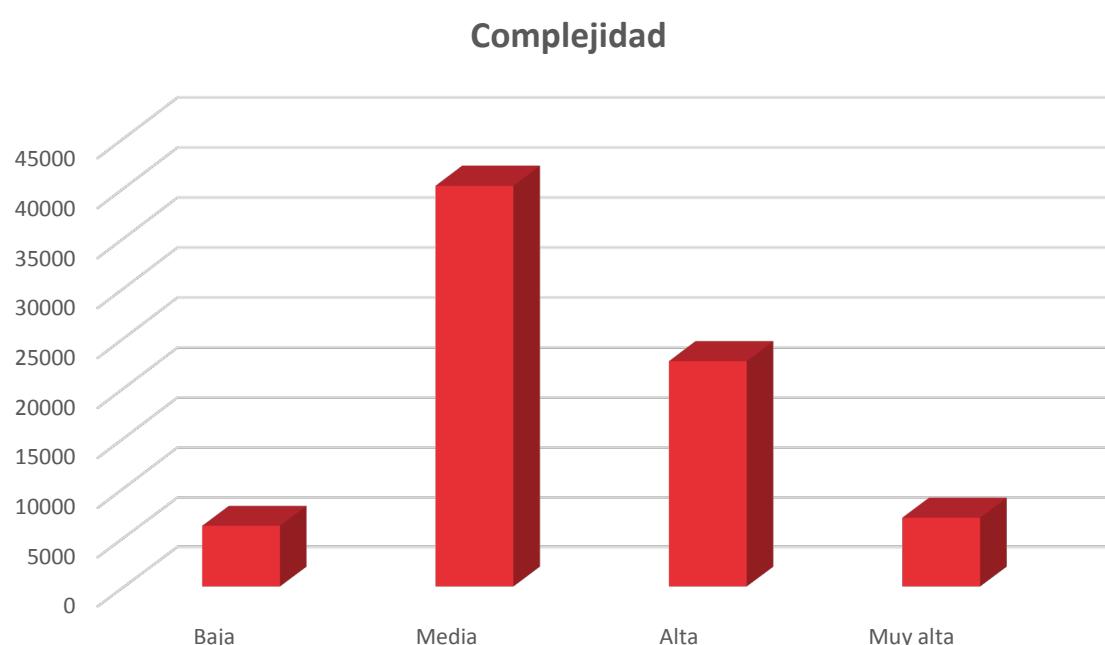


Ilustración 24: Complejidad de las apps analizadas

- El 8,11% de las aplicaciones tienen una complejidad baja.
- El 52,89% de las aplicaciones tiene una complejidad media.
- El 29,81% de las aplicaciones tiene una complejidad alta.
- El 9,19% de las aplicaciones tiene una complejidad muy alta.

5.15. PELIGROSIDAD

Androguard, implementa una serie de métodos que permiten establecer la peligrosidad de una APK, basándose para ello en varios aspectos como el tipo de permisos solicitados por la aplicación. A partir de las aplicaciones analizadas se ha obtenido la información que se muestra en la Ilustración 25.



Ilustración 25: Peligrosidad de las apps analizadas

Menos del 1% de las aplicaciones maliciosas analizadas tienen una peligrosidad menor al 50%.

5.16. IDIOMA POR DEFECTO

El fichero *strings.xml* contiene las cadenas de texto que son utilizadas por una aplicación. A partir de dicho fichero, una vez procesado para excluir las etiquetas características de la definición del mismo, y utilizando la librería [guessLanguage](#) de Python se ha identificado el idioma por defecto de cada una de las aplicaciones analizadas. Los resultados obtenidos son los que se muestran en la Ilustración 26.

Idiomas por defecto

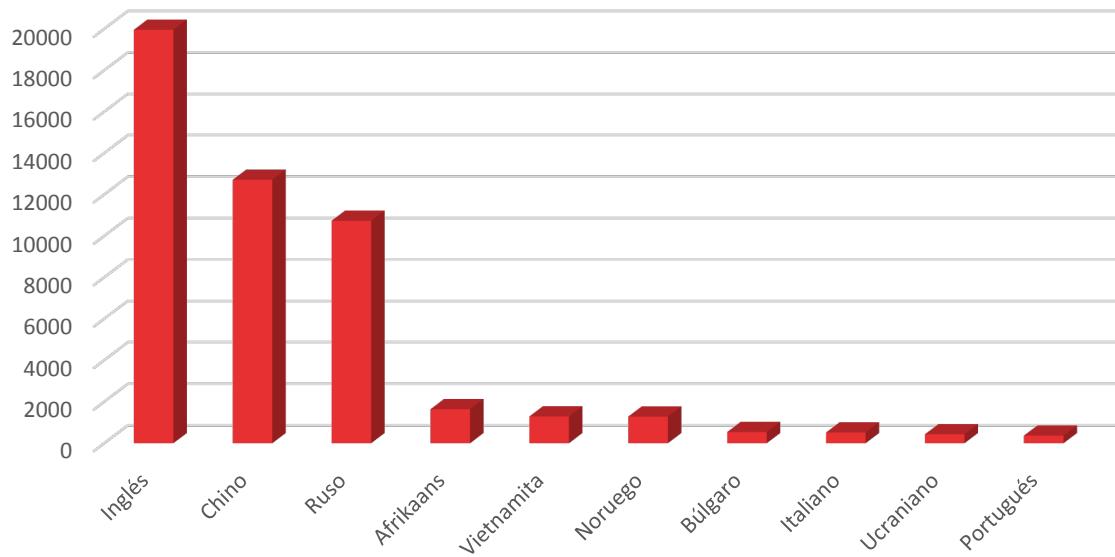


Ilustración 26: Idiomas por defecto de las apps analizadas

Los idiomas más extendidos son inglés, chino y ruso, algo lógico ya que son los públicos objetivos con mayor cuota de mercado.

En muchas ocasiones, no es posible determinar el idioma de una aplicación si el fichero *strings.xml* contiene muy pocos literales o si las cadenas de texto están directamente definidas de manera estática mediante las propiedades de los elementos de la propia aplicación y no a través de dicho fichero. Ese es el motivo de que en un 30,19% de las aplicaciones analizadas no haya sido posible concretar el idioma.

5.17. IDIOMAS ALTERNATIVOS

Extrayendo de los recursos de las aplicaciones los valores correspondientes a los idiomas establecidos en las aplicaciones mediante el análisis y comparándolos con los incluidos en la [ISO 639-1](#) que recoge los idiomas principales del mundo y permite identificarlos mediante 2 letras, se han obtenidos los datos que se muestran en la Ilustración 27.

Multi-Idioma

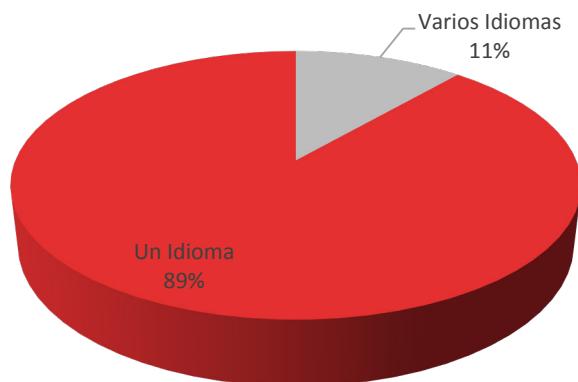


Ilustración 27: Soporte multi-Idioma en las apps analizadas

De ese 11% de aplicaciones con idiomas alternativos, los idiomas más utilizados son los que se muestran en la Ilustración 28.

Idiomas alternativos

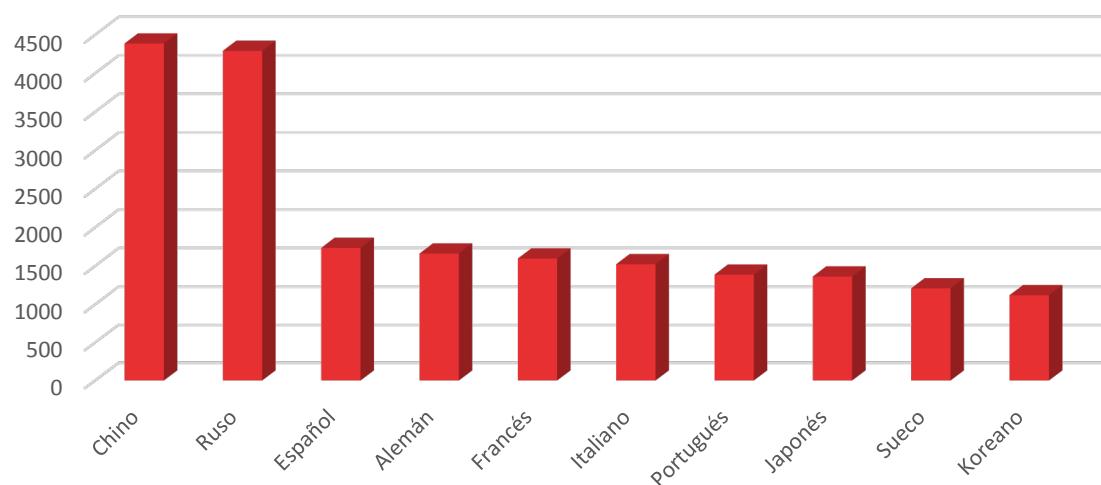


Ilustración 28: Idiomas alternativos

Agrupando las APKs por el número de idiomas alternativos que incluyen se obtienen los datos de la Ilustración 29.

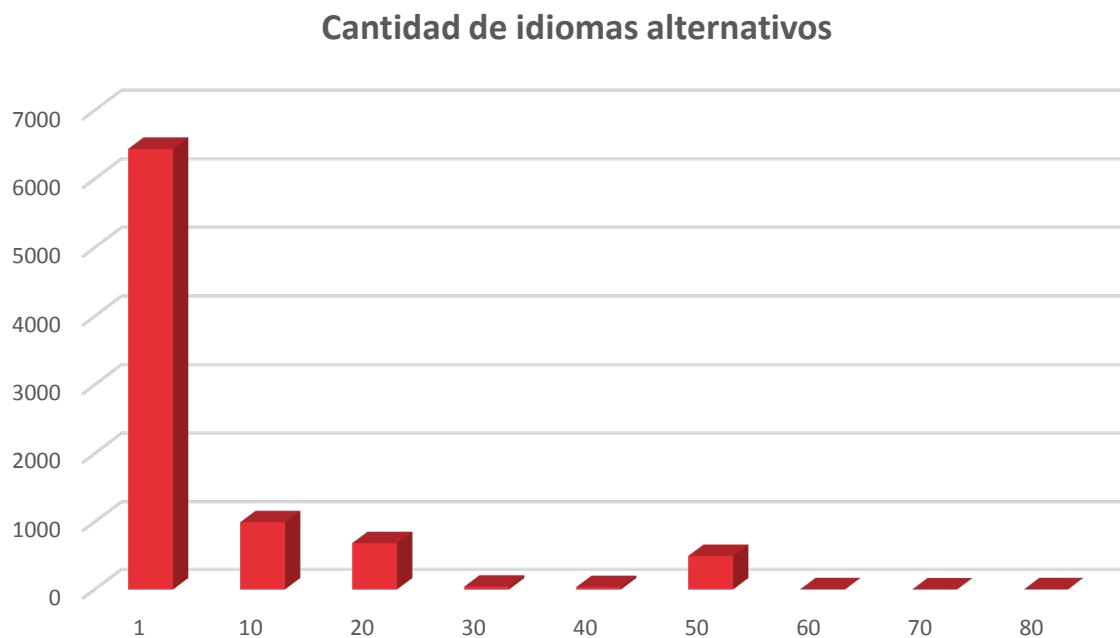


Ilustración 29: Cantidad de idiomas alternativos

El 74,09% de las aplicaciones multi-idioma tienen un único idioma alternativo y el 14,48% tienen configurados más de 20 idiomas.

5.18. PALABRAS MÁS UTILIZADAS

Siguiendo un procedimiento similar al utilizado en el apartado «Nombre del package», se han obtenido las palabras más utilizadas en todas las aplicaciones a partir de las cadenas del fichero *strings.xml*, una vez procesado para excluir las etiquetas características de la definición del mismo. Así mismo, se han excluido los signos de puntuación, exclamación, interrogación, artículos, preposiciones, etc. En la Ilustración 30 se pueden visualizar las palabras más utilizadas.

Palabras más utilizadas

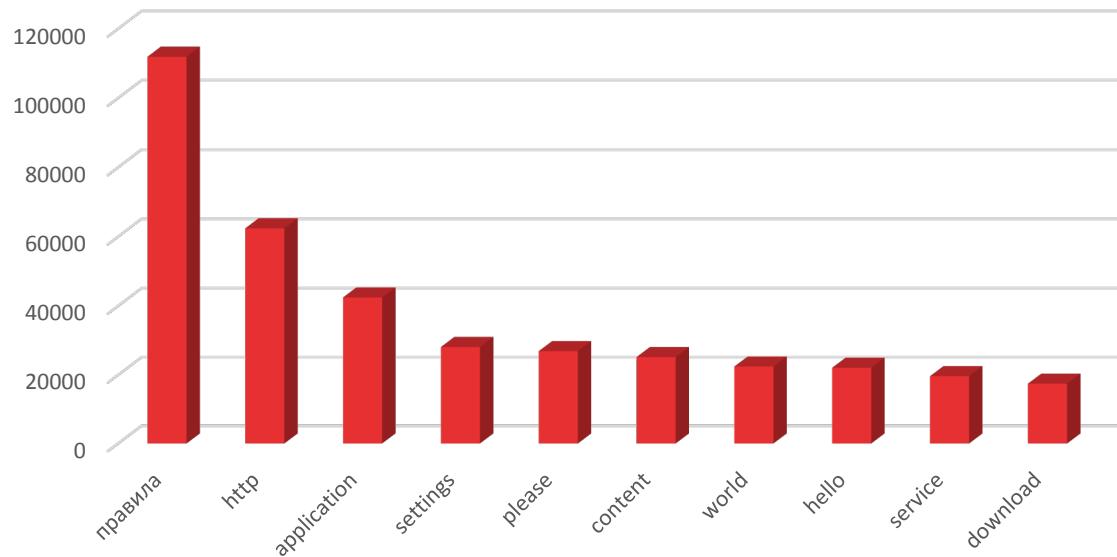


Ilustración 30: Palabras más utilizadas

5.19. CONEXIONES ESTABLECIDAS

Tras extraer las URLs con las que las aplicaciones establecen conexión se ha comprobado dónde están alojados dichos dominios, obteniéndose los datos de la Ilustración 32 e Ilustración 32.

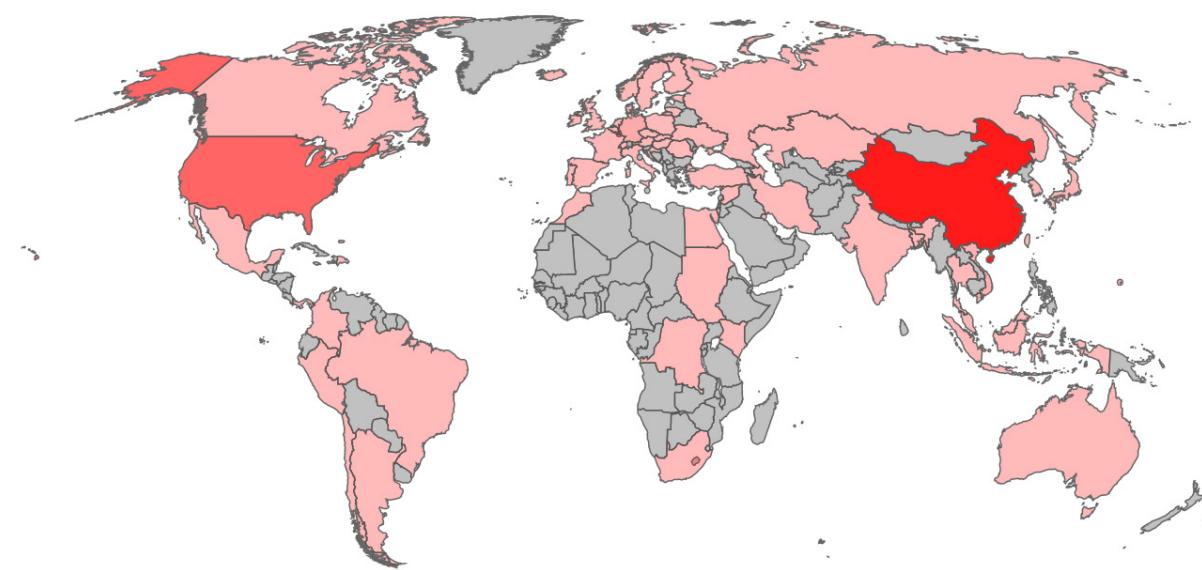


Ilustración 31: Mapa mundial de conexiones establecidas por aplicaciones maliciosas.

Ubicaciones de dominios

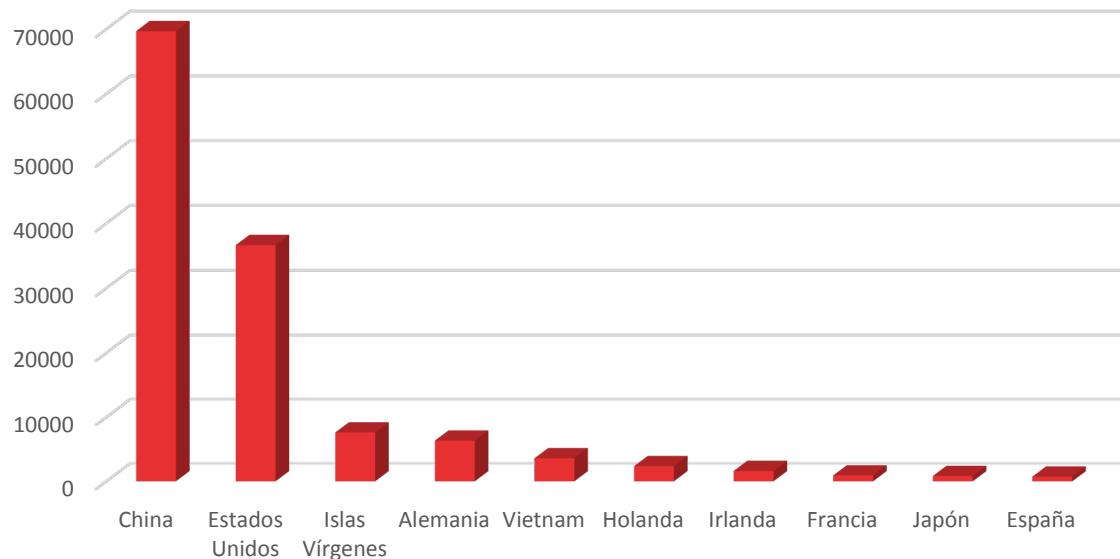


Ilustración 32: Ubicaciones de dominios

Así mismo, se ha comprobado cuáles son los dominios / subdominios con los que las aplicaciones maliciosas establecen más conexiones, siendo los resultados los de la Ilustración 33.

Dominios / Subdominios

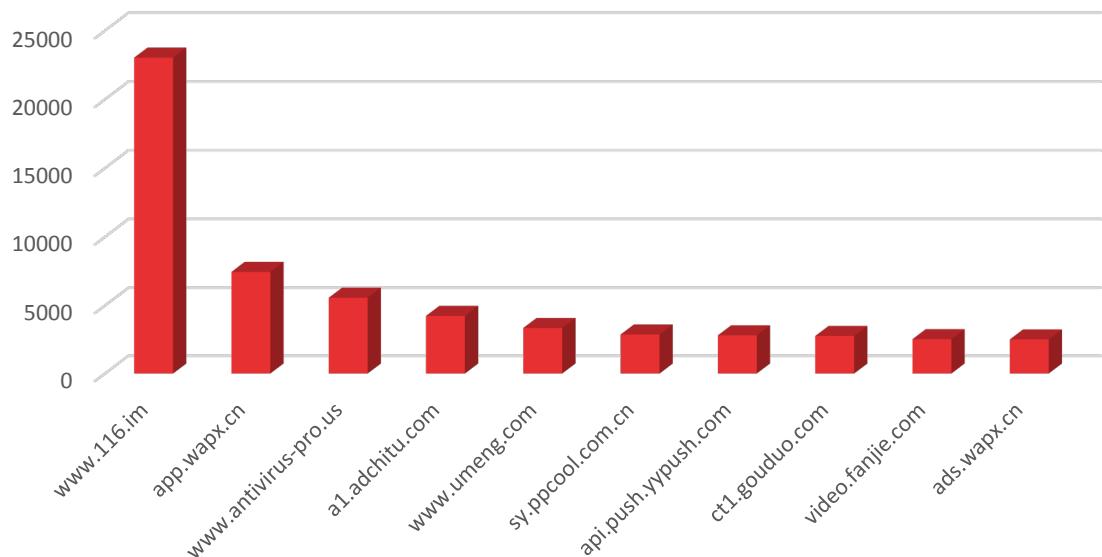


Ilustración 33: Dominios / Subdominios

Es importante tener en cuenta que el hecho de que una aplicación maliciosa establezca conexión con una URL, no implica que dicha URL sea maliciosa: en muchos casos corresponden a los servicios de monetización anteriormente mencionados. Es por ello que se han cruzado los dominios de las URLs obtenidas con diferentes listas de reputación.

No se han comprobado las URLs completas ya que, en muchos casos, incluyen parámetros que varían en cada caso y por lo tanto es muy complicado que estén catalogadas como malware. En la mayoría de los casos, los dominios no estaban catalogados pero en algunos casos sí que estaban identificados como maliciosos, como se muestra en la Ilustración 34.

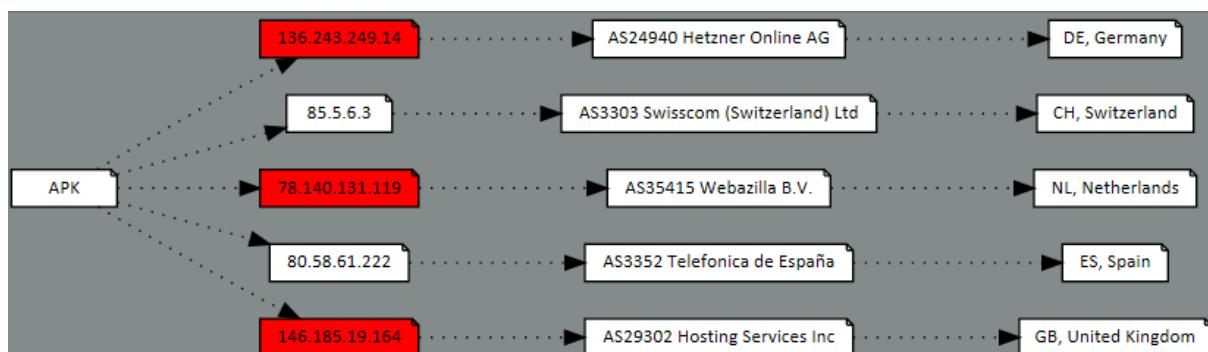


Ilustración 34: Dominios analizados

5.20. DETECCIONES DE ANTIVIRUS

Se ha comprobado el número de antivirus que detectan cada una de las muestras, sobre un total de 55 motores (para visualizar la lista completa de los mismos consultar el «ANEXO 2 – Listado de motores Antivirus»). Los resultados obtenidos se muestran en la Ilustración 35.

Número de detecciones por APK

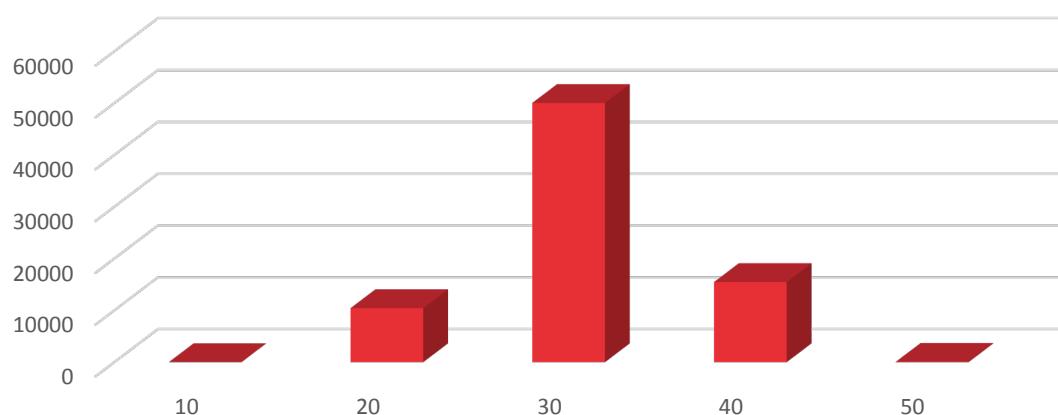


Ilustración 35: Número de detecciones por APK

Únicamente hay 6 muestras de 76.000 que son detectadas por 10 antivirus o menos, lo que confirma la fiabilidad del muestreo seleccionado para el informe.

5.21. TIPO DE MALWARE

Utilizando las detecciones de los motores seleccionados para la obtención de las muestras analizadas para el informe y teniendo en cuenta el formato de la nomenclatura de los alias asignados a cada una de las mismas, se han comprobado los tipos de malware para Android más extendidos, obteniéndose los datos de la Ilustración 36.

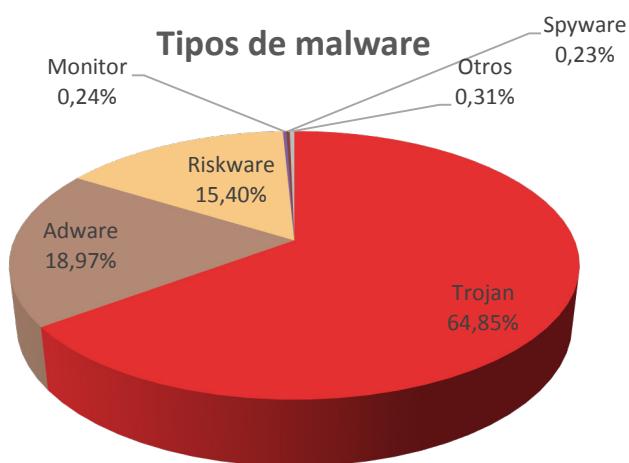


Ilustración 36: Tipos de malware

Al igual que sucede en otras plataformas, como es el caso de Windows, el tipo de malware más extendido son los troyanos.

Respecto a las familias de malware más activas se ha obtenido la clasificación de la Ilustración 37.

Familias de malware más activas

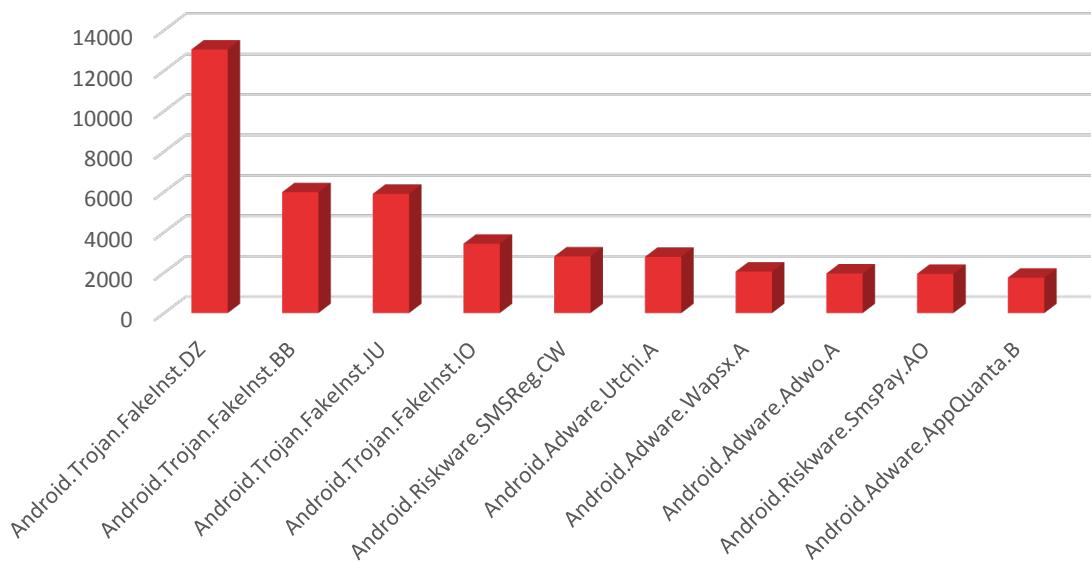


Ilustración 37: Familias de malware más activas

Las familias más activas son las correspondientes a FakelInst, es decir falsos instaladores, que corresponden a supuestos instaladores de otras aplicaciones y que se tratan en realidad de aplicaciones que suscriben a los usuarios a servicios de tarificación especial.

5.22. PRIVACIDAD

Se han analizado todos los métodos de las APKs y se han comparado con un listado, disponible en «ANEXO 3 – Listado de métodos que afectan a la privacidad», de métodos potencialmente peligrosos y que pueden atentar directamente contra la privacidad de los usuarios. A continuación se muestran los datos obtenidos:

- Se han identificado 46.083 aplicaciones con código que afecta directamente a la privacidad de los usuarios que la instalan, lo que supone un 60,64% de las aplicaciones analizadas.
- Se han identificado 181.305 métodos que afectan a la privacidad de los usuarios, lo que supone una media de 2,38 métodos por aplicación.

En la Ilustración 38 se muestran los métodos potencialmente peligrosos más repetidos.

Métodos que afectan a la privacidad

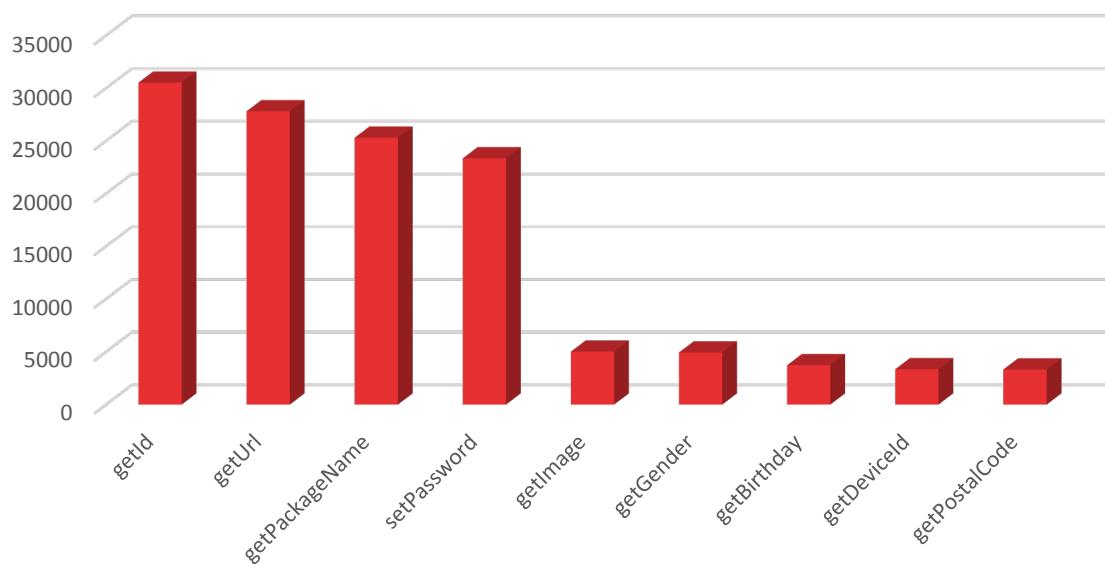


Ilustración 38: Métodos potencialmente peligrosos

Muchos de estos métodos tienen una relación directa con las librerías de monetización, ya que son utilizados por éstas para obtener la información necesaria para personalizar la publicidad en los dispositivos.

6 CONCLUSIONES

Como reflejan los datos, el malware para Android sigue aumentando día tras día y no se prevé un descenso a lo largo de los próximos años, sino todo lo contrario.

Un hecho irrefutable es que, al igual que sucedió en el desarrollado para otro tipo de plataformas como Windows, ha adquirido un nivel de sofisticación muy importante, lo que denota que el nivel de profesionalización de este tipo de servicios es elevado principalmente provocado por el gran beneficio económico que obtienen sus desarrolladores con una inversión muy baja.

Pese a este notable aumento y el nivel de sofisticación alcanzado, los usuarios no son conscientes de los peligros a los que están expuestos y en muchas ocasiones no toman las medidas necesarias para dificultar el trabajo a los desarrolladores de malware. Hay que tener en cuenta que la seguridad empieza por uno mismo, y que el conocimiento y el sentido común son unas de las mejores armas para estar protegidos.

En relación al estudio, se han obtenido gran cantidad de datos significativos, algunos de los más destacados se muestran en la Ilustración 39.



Ilustración 39: Algunos datos relevantes del estudio

- Los desarrolladores de malware para Android prefieren **Windows como plataforma de desarrollo** como demuestra el dato de que el 98% de las aplicaciones analizadas han sido compiladas en dicho sistema operativo.
- El 99,98% de las aplicaciones están **autofirmadas**, aspecto común con las aplicaciones legítimas. Este hecho, hace que la legitimidad de las mismas no esté respaldada por una autoridad certificadora o CA, algo totalmente lógico en el caso de las aplicaciones maliciosas, pero sobre lo que se debería trabajar para dotar de una capa de seguridad adicional a las aplicaciones y por tanto a los usuarios de la plataforma. Es importante encontrar un equilibrio entre la operatividad y la seguridad, siendo necesario plantearse en ocasiones aplicar políticas restrictivas.
- Las aplicaciones poseen 16,1 permisos de media, siendo algunos de los datos más significativos los siguientes:
 - El 99,93% solicitan al menos un permiso catalogado como **peligroso**.
 - El 97% solicitan acceso indiscriminado a Internet, lo que aprovechan para **exfiltrar información**.
 - El 94% solicitan **acceso a información sensible**.
 - Al menos el 79% de las aplicaciones tienen permisos que derivan en **cargos adicionales a la factura** de los usuarios. Este hecho puede suponer un beneficio económico de más de 2 millones de euros al mes, sin tener en cuenta otros aspectos como la venta de información confidencial, etc.
- Basándose en la cantidad de «*Permisos*», «*Services*», «*Receivers*», «*Providers*», «*Activities*» y «*Ads*» se puede determinar que el 39% de las aplicaciones tienen una **complejidad alta o muy alta**.
- Atendiendo a los idiomas soportados por las mismas, se puede concretar que están orientadas en mayor medida a los mercados **chino, ruso y angloparlante**.
- Establecen conexiones principalmente con **China y Estados Unidos**.
- Poseen una media de 2,38 métodos por aplicación que afectan a la privacidad de los usuarios.
- Evaluando el impacto que tienen sobre los usuarios que las instalan, el 99,72% de las aplicaciones tienen una **peligrosidad media / alta**.
- Teniendo en cuenta el número de descargas de las aplicaciones con «*package name*» existente en Google Play, parece que los creadores de malware para Android seleccionan indiscriminadamente las aplicaciones a clonar y modificar añadiendo funcionalidades maliciosas o que, al menos, no se limitan exclusivamente a aquellas con un alto ratio de descargas.

Ante esta situación, es necesario que las compañías con soluciones antivirus, en este caso para sistema operativo Android, hagan un esfuerzo notable e implementen mecanismos que puedan otorgar un mayor nivel de seguridad a los usuarios. Para ello, tomando como punto de partida la información obtenida una vez finalizado el estudio es posible utilizarla para

mejorar los sistemas encargados de realizar la identificación y categorización de aplicaciones potencialmente peligrosas o maliciosas.

Con toda la información obtenida, se pueden generar firmas en base a un gran número de criterios: «*package name*», nombres de componentes de «*Activities*», «*Services*», «*Receivers*», «*Providers*», hash de los componentes de las aplicaciones, conexiones establecidas, una combinación de lo anterior, etc. O incluso utilizar otras vías de estudio como hashes de los métodos peligrosos de las aplicaciones, firmas basadas en el fichero *AndroidManifest.xml*, etc.

De este modo, se pueden realizar detecciones genéricas para diferentes familias de malware y así identificar las nuevas variantes del mismo que vayan surgiendo. Como se ha demostrado, y al igual que sucede en otras plataformas como Windows, los desarrolladores utilizan en muchas ocasiones generadores «*constructors*» que varían ligeramente las aplicaciones con el fin de evadir los sistemas de detección por firma, por lo que la implementación de dichas firmas sería de gran utilidad.

Por otra parte, es necesario que se tomen medidas como ha ocurrido con Android One, a partir del cual Google se responsabiliza de enviar las actualizaciones directamente a los dispositivos móviles sin necesidad de interacción por parte de los fabricantes de los mismos, y así asegurar que la gama de terminales que lo incluyan no sufren el acusado problema de la [fragmentación](#).

Es importante tener presente que **es responsabilidad de todos** los agentes implicados tomar las medidas oportunas para mitigar las amenazas a las que estamos expuestos diariamente.

ANEXO 1 – COMPARATIVA DE LIBRERÍAS DE MONETIZACIÓN

	waps	admob	adwo	domob	youmi	smartmad	vpon	adsmogo	airad	airpush
internet	X	X	X	X	X	X	X	X	X	X
access_network_state	X	O	X	X	X	X	X	X	X	X
read_phone_state	X			X	X	X	X	O	X	X
write_external_storage	X			X	X	O	X	O	X	X
access_coarse_location				X		O	X	O		O
access_wifi_state	X			X	X	O	X	O	X	O
vibrate				O	O		O	O	X	
install_shortcut					O					
access_FINE_location						O	O	O	X	O
camera						O		O		
record_audio						O				X
send_sms						O				
call_phone						O		O		
read_calendar						O		O		
write_calendar						O		O		
read_external_storage						O				
get_tasks	X						O			
mount_unmount_filesystems							O			
system_alert_window							O			
read_logs							O			
write_settings							O			
change_configuration							O			
broadcast_sticky							O			
receive_boot_completed							O			
read_sms							O			
read_contacts							O			

(X) Obligatorio (O) Opcional

ANEXO 2 – LISTADO DE MOTORES ANTIVIRUS

Motor Antivirus	Web	País
Ad-Aware	http://lavasoft.com	Canada
AegisLab	http://www.aegislab.com	Taiwan
Agnitum	http://www.agnitum.com	Rusia
AhnLab-V3	http://www.ahnlab.com	Korea
Antiy-AVL	http://www.antiy.net	China
Avast	http://www.avast.com	República Checa
AVG	http://www.avg.com	República Checa
Avira	http://www.avira.com	Germany
AVware	http://www.avware.com	Brasil
Baidu-International	http://antivirus.baidu.com	China
BitDefender	http://www.bitdefender.com	Rumania
Bkav	http://www.bkav.com	Vietnam
ByteHero	http://www.bytehero.com	China
CAT-QuickHeal	http://www.quickheal.com	India
ClamAV	http://www.clamav.net	Estados Unidos
CMC	http://www3.cmcinfosec.com	Vietnam
Comodo	http://www.comodo.com	USA
Commtouch	http://www.cyren.com	USA
DrWeb	http://www.drweb.com	Rusia
Emsisoft	http://www.emsisoft.com	Nueva Zelanda
ESET-NOD32	http://www.eset.com	Eslovaquia
Fortinet	http://www.fortinet.com	USA
F-Prot	http://www.f-prot.com	Islandia
F-Secure	http://www.f-secure.com	Finlandia
GData	http://www.gdata.de	Alemania
Ikarus	http://www.ikarus.at	Austria
Jiangmin	http://www.jiangmin.com	China
K7AntiVirus	http://www.k7computing.com	India
K7GW	http://www.k7computing.com	India
Kaspersky	http://www.kaspersky.com	Rusia
Kingsoft	http://www.kingsoft.com	China
Malwarebytes	http://www.malwarebytes.org	USA
McAfee	http://www.mcafee.com	USA
McAfee-GW-Edition	http://www.mcafee.com	USA
Microsoft	http://windows.microsoft.com/MSE	USA
MicroWorld-eScan	http://www.escanav.com	India
NANO-Antivirus	http://www.nanoav.ru	Rusia

Norman	http://www.norman.com	Noruega
nProtect	http://www.nprotect.com	Korea
Panda	http://www.cloudantivirus.com	España
Qihoo-360	http://360safe.com	China
Rising	http://www.rising.com.cn	China
Sophos	http://www.sophos.com	UK
SUPERAntiSpyware	http://www.superantispyware.com	USA
Symantec	http://www.symantec.com	USA
Tencent	http://guanjia.qq.com	China
TheHacker	http://www.hacksoft.com.pe	Perú
TotalDefense	http://www.totaldefense.com	USA
TrendMicro	http://www.trendmicro.com	Japón
TrendMicro-HouseCall	http://www.trendmicro.com	Japón
VBA32	http://anti-virus.by	Bielorrusia
VIPRE	http://www.vipreantivirus.com	Malta
ViRobot	http://www.hauri.net	Korea
Zillya	http://zillya.com	Ucrania
Zoner	http://www.zonerantivirus.com	República Checa

ANEXO 3 – LISTADO DE MÉTODOS QUE AFECTAN A LA PRIVACIDAD

getDeviceId	getManufacturer
getLine1Number	getAboutMe
getDeviceSoftwareVersion	getBirthday
getNetworkOperator	getCircledByCount
getNetworkOperatorName	getCover
getSimSerialNumber	getCurrentLocation
getActiveNetworkInfo	getDisplayName
getNetworkPreference	getGender
getDisplayLanguage	getImage
getDisplayCountry	getId
getSubscriberId	getLanguage
getLongitude	getNickname
getLatitude	getOrganizations
getCellLocation	getPlacesLived
getPhoneType	getPlusOneCount
getAccounts	getRelationshipStatus
setPassword	getTagline
getNetworkInfo	getUrl
getAllNetworkInfo	getUserName
getPackageName	getPostalCode
getApiKey	

ÍNDICE DE ILUSTRACIONES

Ilustración 1: Arquitectura de Android	10
Ilustración 2: Android incorpora múltiples controles para proteger a los usuarios. Fuente: Adrian Ludwig - Jefe de seguridad de Android	12
Ilustración 3: Estructura de un fichero APK	13
Ilustración 4: Iconos de algunas de las aplicaciones analizadas	15
Ilustración 5: Top 10 tipos de ficheros	16
Ilustración 6: Modelo de Apps gratuitas con publicidad	18
Ilustración 7: Ejemplo de publicidad en Android	19
Ilustración 8: Monetización	20
Ilustración 9: Top 20 versiones de librerías de monetización	23
Ilustración 10: Distribución de las versiones de Android	25
Ilustración 11: MinSDKVersion	26
Ilustración 12: TargetSDKVersion	27
Ilustración 13: Tiempo de exposición	29
Ilustración 14: Plataforma de desarrollo	29
Ilustración 15: Package names	30
Ilustración 16: Cadenas más utilizadas en los Package names	31
Ilustración 17: Package Name existentes en Google Play	32
Ilustración 18: Top 10 categorías	32
Ilustración 19: Número de descargas de Google Play	33
Ilustración 20: Permisos	35
Ilustración 21: Services más declarados en las apps analizadas	39

Ilustración 22: Porcentaje de las apps analizadas que contienen Receivers	40
Ilustración 23: Porcentaje de las apps analizadas que utilizan Providers	41
Ilustración 24: Complejidad de las apps analizadas	42
Ilustración 25: Peligrosidad de las apps analizadas.....	43
Ilustración 26: Idiomas por defecto de las apps analizadas	44
Ilustración 27: Soporte multi-Idioma en las apps analizadas	45
Ilustración 28: Idiomas alternativos.....	45
Ilustración 29: Cantidad de idiomas alternativos.....	46
Ilustración 30: Palabras más utilizadas.....	47
Ilustración 31: Mapa mundial de conexiones establecidas por aplicaciones maliciosas.....	47
Ilustración 32: Ubicaciones de dominios.....	48
Ilustración 33: Dominios / Subdominios.....	48
Ilustración 34: Dominios analizados	49
Ilustración 35: Número de detecciones por APK	49
Ilustración 36: Tipos de malware	50
Ilustración 37: Familias de malware más activas	51
Ilustración 38: Métodos potencialmente peligrosos	52
Ilustración 39: Algunos datos relevantes del estudio	54

REFERENCIAS

- [1] I. J. Mojica Ruiz, M. Nagappan, B. Adams, T. Berger, S. Dienst y A. E. Hassan, «Impact of Ad Libraries on Ratings of Android Mobile Apps,» 2014. [En línea]. Disponible: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6811104>.
- [2] A. Narayanan, L. Chen y C. K. Chan, «AdDetect: Automated detection of Android ad libraries using semantic analysis,» 2014. [En línea]. Disponible: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6827639&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6827639.
- [3] M. Fidelman, «A comprehensive guide to all mobile ad networks,» 2010. [En línea]. Disponible: <https://www.flickr.com/photos/fidelman/5282509966/>.
- [4] M. C. Grace, W. Zhou, X. Jiang y A.-R. Sadeghi, «Unsafe Exposure Analysis of Mobile In-App Advertisements,» 2012. [En línea]. Disponible: http://www4.ncsu.edu/~mcgrace/WISEC12_ADRISK.pdf.
- [5] R. Stevens, C. Gibler, J. Crussell, J. Erickson y H. Chen, «Investigating User Privacy in Android Ad Libraries,» 2012. [En línea]. Disponible: <http://mostconf.org/2012/papers/27.pdf>.
- [6] SnoopWall, «SnoopWall flashlight apps threat assessment report,» 2014. [En línea]. Disponible: <http://www.snoopwall.com/threat-reports-10-01-2014/>.
- [7] P. Pearce, A. Porter Felt, G. Nunez y D. Wagner, «AdDroid: privilege separation for applications and advertisers in Android,» 2012. [En línea]. Disponible: <https://www.eecs.berkeley.edu/~daw/papers/addroid-asiaccs12.pdf>.
- [8] S. Shekhar, M. Dietz y D. S. Wallach, «AdSplit: Separating smartphone advertising from applications,» 2012. [En línea]. Disponible: <https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/shekhar>.
- [9] X. Zhang, A. Ahlawat y W. Du, «AFrame: Isolating Advertisements from Mobile Applications in Android,» 2013. [En línea]. Disponible: http://www.cis.syr.edu/~wedu/Research/paper/aframe_acsac2013.pdf.

