Universidad de Buenos Aires

Facultades de Ciencias Económicas, Ciencias Exactas y Naturales e Ingeniería

Carrera de Especialización en Seguridad Informática

RELEVAMIENTO DEL ESQUEMA DE SEGURIDAD DEL SISTEMA OPERATIVO ANDROID Y SU APLICACIÓN EN LAS PYMES DE LATINOAMERICA

Autor: Christian Hernán Montalvo Loza

Tutor: Juan Devincenzi

2021

Cohorte 2012

Declaración Jurada de origen de los contenidos

Por medio de la presente, el autor manifiesta conocer y aceptar el Reglamento de Trabajos Finales vigente y se hace responsable que la totalidad de los contenidos del presente documento son originales y de su creación exclusiva, o bien pertenecen a terceros u otras fuentes, que han sido adecuadamente referenciados y cuya inclusión no infringe la legislación Nacional e Internacional de Propiedad Intelectual.

Firmado electrónicamente por:
CHRISTIAN HERNAN MONTALVO LOZA

Christian Hernán Montalvo Loza

PASAPORTE: 1002922464

RESUMEN

Android es la estrategia de Google para masificar el acceso a internet a través de su plataforma, entre más personas usen sus servicios mayor data se genera y por ende mayores posibilidades de generar modelos de negocios.

La presente investigación tiene por fin conocer en profundidad el esquema de seguridad de Android desde el enfoque de las PYMES en Latinoamérica.

Para esto se plantea en primera instancia determinar los orígenes del sistema operativo y como el uso del software libre como rompimiento de la barrera de entrada ha garantizado el éxito de la estrategia de masificación. En este sentido, otro puntal de la estrategia es la Open Handset Alliance, una serie de empresas cuyo giro de negocios tiene que ver con la tecnología móvil.

Una vez entendidos los inicios de Android, se pasa a analizar la arquitectura del sistema operativo. En este apartado, se detalla las características de cada una de las cinco capas, que en una estructura de pila dan forma a su arquitectura. El impacto de Linux en el diseño, los conceptos de máquinas virtuales y la adopción de Java como lenguaje de programación terminan este análisis.

En un tercer capítulo se profundiza en el esquema de seguridad de Android, dando un vistazo a los niveles de protección, el entorno de trabajo, las formas que las aplicaciones tienen de intercomunicarse, entre otros serán los temas para tratarse.

Para finalizar, se topa la tendencia del Bring Your Own Dispositive BYOD en las pequeñas y medianas empresas desde un enfoque estadístico, de tal manera que permita diagnosticar la situación laboral actual desde la perspectiva del uso de los dispositivos móviles en los entornos laborales.

INDICE

INDICE DE IMÁGENES

INDICE DE TABLAS

INTRODUCCIÓN

La sociedad de la información permite el acceso ilimitado por parte de las personas a la información generada por otros, además de considerar al conocimiento como un valor agregado. En la revolución digital que vino de la mano con el internet, tiene mucho que ver Google, quien prácticamente domina a través de su buscador, con más del 90% del mercado según Statcounter.

El acelerado crecimiento en un poco más de una década con respecto al uso de dispositivos móviles ha cambiado totalmente la forma que nos interrelacionamos. El concepto del mando único, es decir, que desde un solo dispositivo podemos tener acceso a todo aquello que para nosotros resulta importante; ha provocado prácticamente una dependencia tecnológica en cada una de nuestras facetas.

La crisis sanitaria provocada por el SARS Cov 2, sin duda alguna ha conllevado vivir una nueva realidad, acelerando la adopción (que era inevitable) de las tecnologías de la información en todo ámbito: familiar, educativo, laboral, social y emocional. Seguramente una de las dudas recurrentes de los nuevos millones de usuarios de las tele-algo (teleeducación, teletrabajo, telemedicina, etc.) es como aseguro mi información personal, que hoy como nunca, está expuesta al mundo.

En este contexto, entendiendo que, Android es el principal sistema operativo para dispositivos móviles, se analiza el esquema de seguridad de este, mencionando sus raíces desde Google, su arquitectura y finalizando con un análisis de su estrategia de seguridad y su apego al marco de seguridad de Linux.

Adicional, para aterrizar en el caso práctico, se argumenta una tendencia desde el enfoque laboral actual y, se toma a las PYMES para analizar las implicancias del BYOD en los nuevos entornos laborales.

NOMINA DE ABREVIATURAS

BYOD	Bring Your Own Dispositive
OHA	Open Handset Alliance
IPC	Interprocess Communication
PYMES	Pequeñas y Medianas Empresas
APP	Aplicación
DVM	Dalvik Virtual Machine
JVM	Java Virtual Machine
API	Application Programming Interface
CEO	Chief Executive Officer
CIO	Chief Information Officer
ARM	Advanced RISC Machine
RISC	Reduced Instruction Set Computer
OpenGL	Open Graphics Library
BSD	Berkeley Software Distribution
AOT	Ahead-Of-Time
UID	User Identification
GID	Group Identity
VFAT	Virtual File Allocation Table
SD	Secure Digital

1 ANDROID, LA ESTRATEGIA DE GOOGLE

Resulta imposible someter la arquitectura del sistema Android a un análisis de seguridad sin previamente entender, por lo menos en lo general, cual es la visión del negocio que, desde Google tienen para este, y cuál fue la estrategia que ha impulsado la rápida escalada en el mercado de los dispositivos móviles.

El mismo Andy Rubin [1], define cual es el objetivo que inspira a Android: *"No quiero crear algún derivado de la Internet, no quiero tomar sólo una porción de la Internet, no quiero estar en la esquina en algún lugar con alguna versión idiotizada de Internet, quiero permanecer en Internet".*[1]

Posiblemente uno de los puntos de observación más llamativos es la variedad de marcas y modelos de dispositivos que hoy en día llevan a Android como su Sistema Operativo, teniendo los fabricantes incluso la posibilidad de personalizar ciertos aspectos del sistema operativo, por ejemplo, el teclado. Esto, de inicio brinda una muestra de la estrategia utilizada, asociarse y brindar variedad, sin someter al usuario a un solo diseño o empresa; en principio, la estrategia funciona.

Android es sin duda una poderosa plataforma que necesita de gran desarrollo y altísimos costos, pero ¿por qué desarrollarla como plataforma abierta? Bueno, la respuesta la encontramos en la página corporativa de Google, la que reza: "*La misión de Google es organizar la información del mundo y hacerla universalmente accesible y útil"* [2]

Los celulares son el dispositivo móvil más consumido, entonces al proveer una plataforma avanzada sin costo se hace más atractiva para los desarrolladores y obviamente al haber una mayor cantidad de celulares con

[1]RUBIN, Andy, ex CEO de Android

acceso a internet mayor será el número de usuarios con acceso a la red y por supuesto utilizando su buscador.

"Google tiene un gran modelo de negocio basado en la publicidad, la publicidad y el código abierto tienen una conexión natural. El código abierto es básicamente una estrategia de distribución, elimina completamente la barrera de entrada para la adopción"[2].

1.1 INTRODUCCIÓN HISTÓRICA

El primer teléfono móvil que atrajo seriamente la atención de los usuarios de tecnología fue sin duda el Black Berry de la canadiense Research in Motion RIM, el mismo que presentaba nuevas funcionalidades que le entregaban una utilidad adicional al teléfono celular, era el inicio de los Smart Phones, pero sobre todo del acceso a internet móvil lo cual lo hizo muy atractivo en los sectores empresariales, haciendo estos beneficios alcanzables a quien los pudiera pagar.

Pero la verdadera masificación de los teléfonos inteligentes vino a partir del 2007 cuando Apple presentó su primera versión del Iphone; esto suponía la presencia de un nuevo hito tecnológico, el Smart Phone ahora era más amigable y divertido, además de ser un icono estético, era más alcanzable para la clase media, económicamente hablando [3].Para entonces existían grandes marcas que disputaban el mercado de teléfonos celulares; y cada uno con su propio sistema operativo, entre ellos Nokia, Apple, RIM y Treo.

[2]RUBIN, Andy, ex CEO de Android

1.2 PLATAFORMA ANDROID

Android es una plataforma, principalmente, para dispositivos móviles, de código abierto basada en el Kernel 2,6 de Linux [4], es administrado por la **OHA** Open Handset Alliance, un grupo de 84 compañías de tecnología y dispositivos móviles [5].

Android ha tenido un impacto significativo en el mercado de dispositivos móviles, apenas dos años después de presentarse un primer dispositivo, Android era ya la segunda plataforma en importancia en Estados Unidos abarcando el 26% de ese mercado. Para el 2010 paso a ser la segunda a nivel mundial por debajo del Black Berry de RIM [4]. Para el 2013, según su sitio oficial, Android tenía ya 900 millones de activaciones convirtiéndola en líder a nivel mundial [6].

Siguiendo una evolución histórica de la compañía: Android Inc. fue fundada en Palo Alto California en octubre del 2003 por Andy Rubin, Rich Minero, Nick Sears y Chris White. Entre estos sobresale Rubin con experiencia trabajando para varias firmas de robótica, Apple, WebTV y Danger Inc. creadora de Danger OS, sistema operativo del T-Mobile Sidekick [4] y basado en java, tal vez este sea un primer origen. Todavía Rubin sigue trabajando como vicepresidente de ingeniería en Google.

Su intención original era crear un sistema operativo para ser utilizado por cámaras digitales, dándose cuenta de que lo que habían creado no era compatible con los recursos limitados que disponía en ese momento una cámara fotográfica pusieron todo su empeño en evolucionar su sistema operativo para ser implementado en teléfonos celulares [7].

En noviembre del 2007 aparece en escena la Open Handset Alliance un grupo de empresas de tecnología encabezada por Google, acompañados por los mayores fabricantes de dispositivos como HTC y Samsung, los operadores inalámbricos Sprint Nextel y T-Mobile, los fabricantes de chips Qualcomm y

Texas Instruments, con el objetivo de desarrollar estándares abiertos para dispositivos móviles [8].

Y es que, si algo hacía falta en un mercado en el cual abundaban las marcas y cada una con sus propias reglas y formas de trabajar, era precisamente estándares, algo que les permitiera trabajar libremente, pero con reglas mínimas de cumplimiento universal. En eso radica el éxito de Android en que no está atado a una marca de hardware o una compañía de servicios móviles, al contrario, brinda a sus usuarios la capacidad de elegir y para los desarrolladores, Google les brinda soporte directamente.

A partir de ahí, el sistema operativo ha tenido numerosas actualizaciones que han mejorado progresivamente sus prestaciones, agregándole nuevas características [6]. A partir del 2010 Google lanza con el nombre de Nexus una serie de dispositivos móviles (smartphones y tablets) incorporando nuevos socios a la Open Handset Alliance, grandes marcas como: HTC, LG, Samsung, Sony, Motorola, T-mobile, China Mobile, Telecom, Telefónica, Ebay, Broadcom, Intel, Nvidia, Synaptics, Texas, Vodafone, Softbank, Atheros, AKM, Ericcson, Assustek, Huawei, Garmin, Thoshiba, Dell, China Unicom, Svox, Acer, ZTE, Alcatel, Nec, Kyocera, Lenovo, Alcatel, Sharp [7] que le permiten ampliar la gama de productos ya no únicamente a dispositivos móviles.

1.3 EVOLUCIÓN DE VERSIONES

En junio de 2020 se presentó públicamente la primera beta de la más reciente versión de Android, la 11 denominada Lollipop. Esto marca un hecho histórico en la vida del Sistema Operativo, pasados alrededor de 15 años desde la creación de Android Inc., y 13 años desde que se presentara su primera versión, allá por noviembre de 2007 [9] .

Una de las características reconocibles en Android es la gran cantidad de versiones que se han publicado; además que cada una las versiones han debido ser actualizada al poco tiempo de lanzada al conocerse varios problemas de funcionamiento. Y es que son quince las versiones presentadas desde la 1.0 “Apple Pie” hasta la 11 y cada una con sus respectivas correcciones.

2 ARQUITETURA ANDROID

Android es un sistema operativo para dispositivos móviles, basado en Linux. Su arquitectura está estructurada en capas, de tal manera que, cada capa se sirve de la capa inmediatamente inferior, es decir, su arquitectura es de tipo pila. Además, esta distribución permite acceder a las capas más bajas mediante el uso de librerías para que así los desarrolladores no tengan que programar a bajo nivel las funcionalidades necesarias para que una aplicación haga uso de los componentes de hardware [10].

En muchas fuentes de consulta (publicaciones, revistas, sitios web, etc.) se menciona como una gran ventaja que el hecho que sea basado en Linux, pero ¿Android es lo mismo que Linux?

Bien, se pueden encontrar quienes entienden a Linux como únicamente un Kernel [11] y quienes lo identifican como un Sistema Operativo [12]. Lo objetivamente cierto es que en lo que colabora Linux en esta arquitectura es en el manejo de Hardware desde su Kernel.

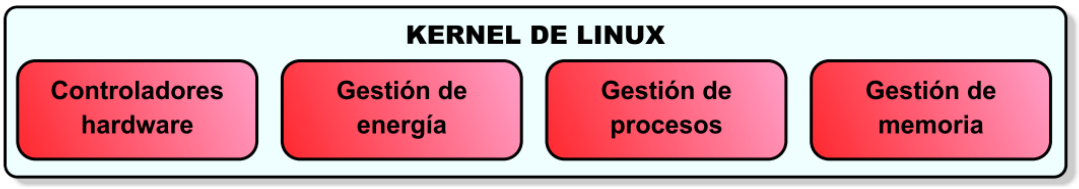
Para comprender mejor esta arquitectura, a continuación, se analizará cada una de sus capas.

[12]

Imagen 1 Arquitectura Android

2.1 KERNEL

Lo que el Kernel Linux le ofrece a Android es un entorno estandarizado que permite que la capa de abstracción de hardware, las librerías y las aplicaciones funcionen sobre un entorno controlado que no dependa directamente del hardware del dispositivo [10].



[10]

Imagen 2 Kernel Linux de Android

Esto es el puntapié inicial del éxito, en el mercado existen un sinfín de dispositivos con Android, si para cada uno de estos se tuviera que repensar todo el Sistema Operativo, esto se convertiría en una carga muy difícil de llevar y una odisea pensar en una estrategia de distribución de actualizaciones o nuevos desarrollos, pero sí en cambio, para cada uno de estos terminales tienes únicamente que pensar en pequeños retoques se simplifica significativamente esta labor.

Como contraparte, cada fabricante se responsabiliza de desarrollar y distribuir las personalizaciones al código fuente. Para esta tarea cuentan un Kernel y un sistema operativo base estable; además de una comunidad de desarrolladores y fuentes de información como respaldo.

El Kernel Linux fue adaptado para su funcionamiento en arquitectura ARM[3] y permite que se pueda acceder a los componentes de hardware sin necesidad

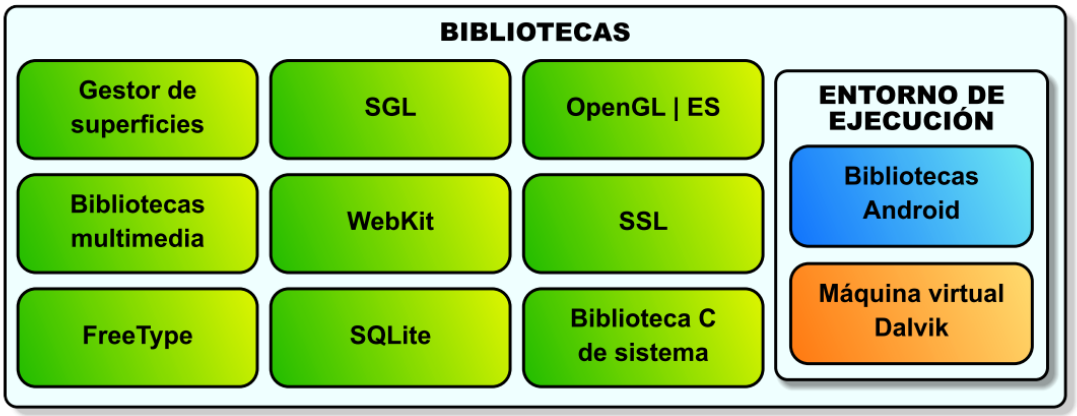
[3] ARM: es una arquitectura RISC (Reduced Instruction Set Computer=Ordenador con Conjunto Reducido de Instrucciones) de 32 bits.

de conocer el modelo o características precisas de cada teléfono o dispositivo. De esta forma, si una aplicación necesita, por ejemplo, la brújula, podrá utilizar la que incluya el teléfono, sea cual sea. Para cada elemento de hardware del teléfono existe un controlador (o driver) dentro del Kernel que permite utilizarlo desde el software.

Además de proporcionar controladores de hardware, el Kernel se encarga de gestionar los diferentes recursos del teléfono (por ejemplo, energía o memoria) y del sistema operativo en sí: procesos, elementos de comunicación (networking), etc. [13].

2.2 BILIOTECAS

En esta capa encontramos las bibliotecas nativas de Android (también llamadas Librerías), se encuentran escritas en C y C++ y compiladas para la arquitectura de hardware específico para cada dispositivo, esto por lo general lo realiza cada fabricante como parte de la personalización.



[10]

Imagen 3 Bibliotecas Nativas de Android

Esta capa tiene como objetivo ser base de funcionalidad para las aplicaciones que correrán sobre el Sistema Operativo, siempre tratando de hacer el rendimiento del dispositivo lo más eficiente posible.

Para los desarrolladores estas bibliotecas son presentadas a través de las API (Application Program Interface)[4], de esta manera se evita la interacción directa con la Bibliotecas, siendo estas invocadas desde las aplicaciones [14].

Algunas de las bibliotecas regularmente incluidas son:

2.2.1 SURFACE MANAGER:

Se encarga de componer las imágenes en pantalla a partir de capas graficas 2D y 3D. En lugar de mostrar directamente la imagen solicitada por una aplicación, esta realiza los cambios en los mapas de bits combinándolos con los que están en memoria, lo cual permite la realización de varios efectos.

2.2.2 SGL

La Scalable Graphics Layer es el motor de gráficos 2D utilizado tanto por Android como por Chrome.

[4] API: es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

2.2.3 OPENGL[5] | ES

OpenGL para Sistemas Embebidos, es un motor de gráficos 3D basado en la API de OpenGL, utiliza, si el dispositivo lo permite la aceleración de hardware, caso contrario lo manejará por software.

2.2.4 BIBLIOTECAS MULTIMEDIA

Proporciona todos los códecs[6] necesarios para el contenido multimedia soportado en Android (vídeo, audio, imágenes estáticas y animadas, etc. [15]

2.2.5 WEBKIT

Es el motor web utilizado tanto por el navegador como por las diferentes aplicaciones de forma embebida, este motor es el mismo utilizado por Chrome o Safari.

2.2.6 SSL

La Secure Sockets Layer, proporciona seguridad mediante cifrado en la navegación por internet.

[5] OpenGL (Open Graphics Library) es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D.

[6] Códec es la abreviatura de codificador-decodificador. Describe una especificación desarrollada en software, hardware o una combinación de ambos, capaz de transformar un archivo con un flujo de datos (stream) o una señal.

2.2.7 FREETYPES

Permite mostrar diferentes fuentes tipográficas, sean estas vectoriales o basadas en mapas de bits.

2.2.8 SQLLITE

Motor de bases de datos relacionales, disponible para todas las aplicaciones.

2.2.9 BIBLIOTECA C DE SISTEMA (LIBC)

Está basada en la implementación de Berkeley Software Distribución (BSD), pero optimizada para sistemas Linux embebidos. Proporciona funcionalidad básica para la ejecución de las aplicaciones.

2.3 ENTORNO DE EJECUCION

Cada aplicación que se inicializa en Android ejecuta sus procesos a través de sus propias instancias de máquina virtual. Según se describe en el gráfico a continuación, el entorno de ejecución conlleva el trabajo colaborativo entre las bibliotecas nativas Android y la máquina virtual Dalvik, sobre esta última de profundizará más adelante.

[10]

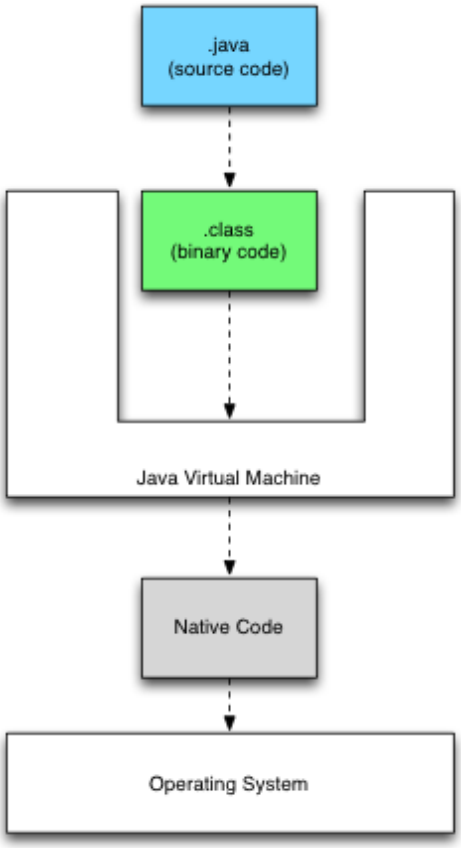
Imagen 4 Android Runtime

Este entorno de ejecución, a pesar de apoyarse en la capa de bibliotecas, no se considera una capa en sí misma, ya que también usa bibliotecas para su funcionamiento.

2.3.1 MAQUINA VIRTUAL DALVIK

El componente principal del entorno de ejecución de Android es la Máquina Virtual Dalvik (DVM), componente que ejecuta todas y cada una de las aplicaciones no nativas de Android [16].

Android y Dalvik son los encargados de trabajar con el código fuente Java y convertirlo en código nativo para el kernel de Linux que la plataforma Android soporta.



[17]

Imagen 5 Ciclos Compilación JVM

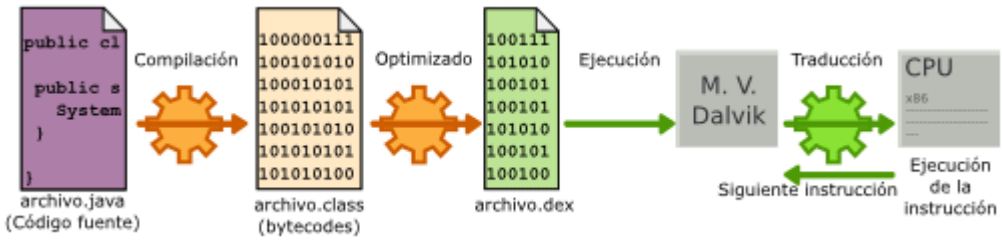
Para comprender el funcionamiento de la Máquina Virtual Dalvik es necesario referirse a la Máquina Virtual de Java, debido a su similar

funcionamiento, y a que el entorno de desarrollo de las aplicaciones a ejecutarse en Android es Java.

Cuando hablamos de Java se debe partir de un código fuente (.java), el cual luego debe ser compilado (javac) para convertirlo a código binario (.class), este código binario es el que la máquina virtual Java (JVM) es capaz de interpretar y convertir en código nativo que el sistema operativo puede interpretar.

Una vez examinado el proceso de compilación que realiza la JVM podemos definir más claramente la diferencia con el proceso que realiza la DVM.

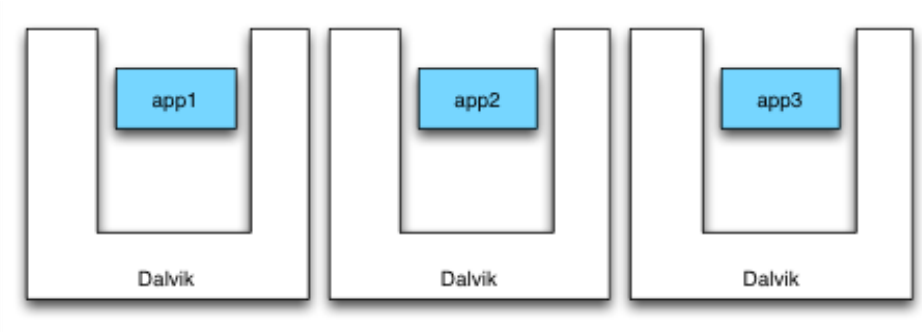
Como la siguiente imagen lo muestra, Android añade un paso intermedio. En este paso todos los ficheros .class de Java son pasados por otro proceso adicional de optimización (por los limitados recursos con lo que cuenta una plataforma móvil). En este proceso todos los ficheros .class son agrupados en un único fichero. dex (Dalvik Ejecutable). El grado de optimización es tal que un fichero. dex descomprimido ocupa la mitad que las mismas clases en formato .jar (comprimido). [17]



[18]

Imagen 6 Ciclos Compilación DVM

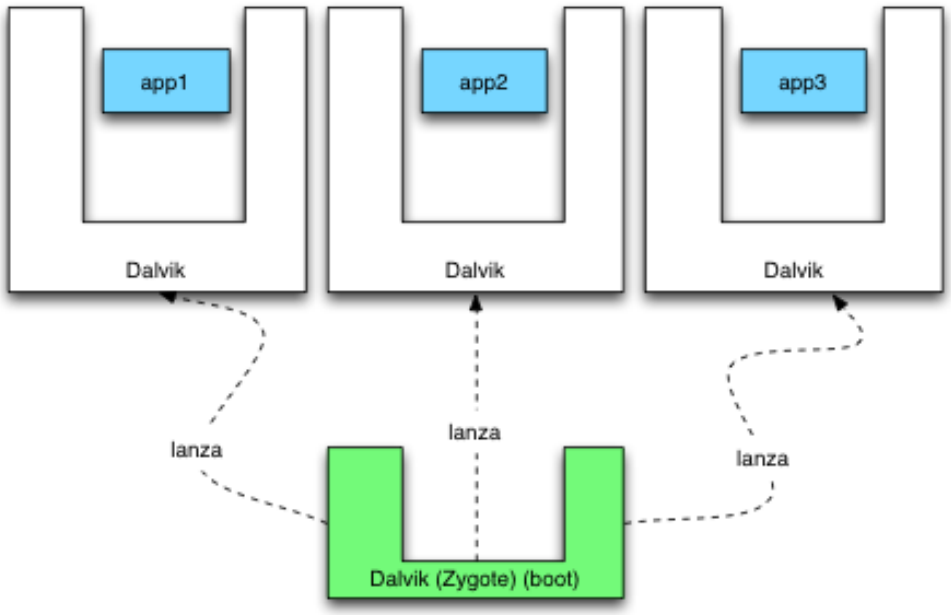
Otra diferencia importante que existe entre las aplicaciones Java y las aplicaciones Dalvik es que en un servidor de aplicaciones Java todas las aplicaciones corren sobre la misma máquina virtual. En Dalvik esto no es así ya que cada aplicación corre en su propia máquina virtual, consiguiendo un mayor aislamiento entre las distintas aplicaciones.



[17]

Imagen 7 MULTI DVM

Aunque disponemos de una máquina virtual por cada aplicación Android realiza una fuerte optimización y carga una máquina virtual inicial denominada Zygote que es la encargada de arrancar el resto.



[17]

Imagen 8 ZYGOTE DVM

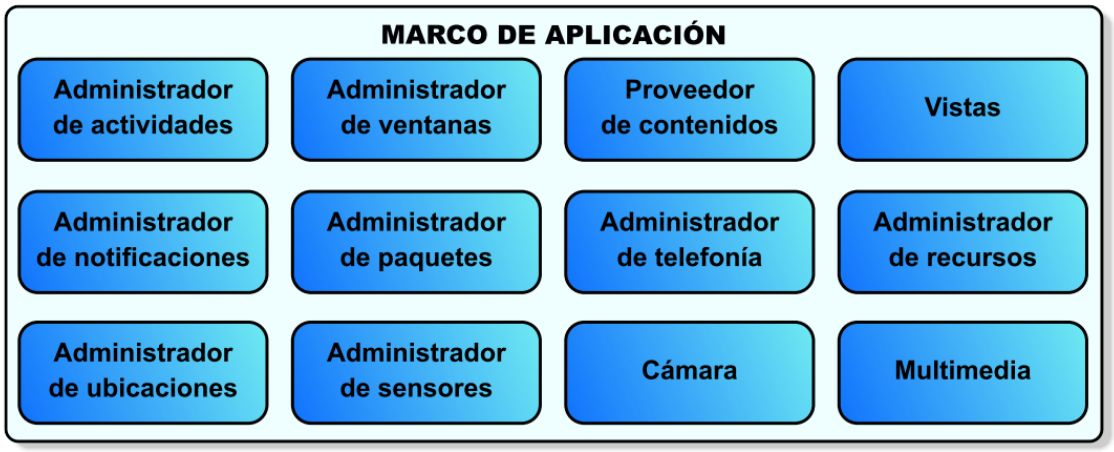
2.3.2 ART LA EVOLUCIÓN DE DALVIK

Cada vez que iniciamos una aplicación hay una serie de operaciones que se realizan en el código. Este proceso es el causante del hecho de que, según va pasando el tiempo y vamos usando el teléfono, cada vez vaya más lento. En cambio, ART es AOT (Ahead-Of-Time), es decir, comienza una pre-compilación al instalar la aplicación y, por tanto, al ejecutarla esta ya no requiere tanta carga de datos como antes y hace que iniciar una aplicación se produzca en menos tiempo [19], realizando estas operaciones una única vez en el momento de la instalación de la

aplicación. Esto supone un ahorro de recursos, pero a la vez supone un mayor tiempo en el proceso de instalación.

A su vez, se traduce también en un ahorro de batería al no tener que hacer este proceso cada vez que se inicia una aplicación. Para los dispositivos con Android 5.0 (nivel de API 21) o versiones posteriores, cada app ejecuta sus propios procesos con sus propias instancias del tiempo de ejecución de Android (ART) [20].

2.4 MARCO DE APLICACIÓN



[10]

Imagen 9 Marco de Aplicación

La siguiente capa está formada por todas las clases y servicios que utilizan directamente las aplicaciones para realizar sus funciones. La mayoría de los componentes de esta capa son librerías Java que acceden a los recursos de las capas anteriores a través de la máquina virtual Dalvik.

Los desarrolladores tienen acceso completo a los mismos APIs del framework usados por las aplicaciones base. La arquitectura está diseñada para

simplificar la reutilización de componentes; cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede luego hacer uso de esas capacidades (sujeto a reglas de seguridad del framework). Este mismo mecanismo permite que los componentes sean reemplazados por el usuario.

A continuación, se detallan los componentes que frecuentemente incluye esta capa [10]:

2.4.1 ACTIVITY MANAGER.

Se encarga de administrar la pila de actividades de nuestra aplicación, así como su ciclo de vida.

2.4.2 WINDOWS MANAGER.

Se encarga de organizar lo que se mostrará en pantalla. Básicamente crea las superficies en la pantalla que posteriormente pasarán a ser ocupadas por las actividades.

2.4.3 CONTENT PROVIDER.

Esta librería es muy interesante porque crea una capa que encapsula los datos que se compartirán entre aplicaciones para tener control sobre cómo se accede a la información.

2.4.4 VIEWS.

En Android, las vistas los elementos que nos ayudarán a construir las interfaces de usuario: botones, cuadros de texto, listas y hasta elementos más avanzados como un navegador web o un visor de Google Maps.

2.4.5 NOTIFICATION MANAGER.

Engloba los servicios para notificar al usuario cuando algo requiera su atención mostrando alertas en la barra de estado. Un dato importante es que esta biblioteca también permite jugar con sonidos, activar el vibrador o utilizar los LEDs del teléfono en caso de tenerlos.

2.4.6 PACKAGE MANAGER.

Esta biblioteca permite obtener información sobre los paquetes instalados en el dispositivo Android, además de gestionar la instalación de nuevos paquetes. Con paquete nos referimos a la forma en que se distribuyen las aplicaciones Android, estos contienen el archivo .apk, que a su vez incluyen los archivos. dex con todos los recursos y archivos adicionales que necesite la aplicación, para facilitar su descarga e instalación.

2.4.7 TELEPHONY MANAGER.

Con esta librería podremos realizar llamadas o enviar y recibir SMS/MMS, aunque no permite reemplazar o eliminar la actividad que se muestra cuando una llamada está en curso.

2.4.8 RESOURCE MANAGER.

Con esta librería podremos gestionar todos los elementos que forman parte de la aplicación y que están fuera del código, es decir, cadenas de texto traducidas a diferentes idiomas, imágenes, sonidos o layouts. En un post relacionado a la estructura de un proyecto Android veremos esto más a fondo.

2.4.9 LOCATION MANAGER.

Permite determinar la posición geográfica del dispositivo Android mediante GPS o redes disponibles y trabajar con mapas.

2.4.10 SENSOR MANAGER.

Nos permite manipular los elementos de hardware del teléfono como el acelerómetro, giroscopio, sensor de luminosidad, sensor de campo magnético, brújula, sensor de presión, sensor de proximidad, sensor de temperatura, etc.

2.4.11 CÁMARA.

Con esta librería podemos hacer uso de la(s) cámara(s) del dispositivo para tomar fotografías o para grabar vídeo.

2.4.12 MULTIMEDIA.

Permiten reproducir y visualizar audio, vídeo e imágenes en el dispositivo.

2.5 APLICACIONES

La capa superior de esta pila de software la forman las aplicaciones. En esta capa se incluyen todas las aplicaciones del dispositivo, tanto las que tienen interfaz de usuario como las que no, así mismo, las nativas (programadas en C++) como las administradas (programadas en Java), tanto las que vienen de serie con el dispositivo como las instaladas por el usuario [15]. Todas las aplicaciones han de correr en la máquina virtual Dalvik para garantizar la seguridad del sistema.

[10]

Imagen 10 Capa de Aplicaciones

Aquí está también la aplicación principal del sistema: inicio (home), también llamada a veces lanzador (launcher), porque es la que permite ejecutar

otras aplicaciones proporcionando la lista de aplicaciones instaladas y mostrando diferentes escritorios donde se pueden colocar accesos directos a aplicaciones o, incluso, pequeñas aplicaciones incrustadas o widgets, que son también aplicaciones de esta capa.

El aspecto principal para tener en cuenta de esta arquitectura es que todas las aplicaciones (las nativas de Android, las que proporciona Google, las que incluye de serie el fabricante del teléfono o las que instala después el usuario) utilizan el mismo marco de aplicación para acceder a los servicios que proporciona el sistema operativo. Esto implica dos cosas: que podemos crear aplicaciones que usen los mismos recursos que usan las aplicaciones nativas (nada está reservado o inaccesible) y que podemos reemplazar cualquiera de las aplicaciones del teléfono por otra de nuestra elección.

3 ESQUEMA DE SEGURIDAD

Seguramente uno de los comentarios más populares al momento de hablar de Android, por lo menos entre los amantes de la tecnología, es que está basado en Linux y que mediante esto hereda mucho de su esquema de seguridad, lo que es probablemente uno de los pilares más importantes para explicar el éxito obtenido.

Sin embargo, es común encontrar en foros de internet comentarios acerca de la débil seguridad que Android brinda a sus usuarios, pero bien vale realizar un análisis imparcial de su esquema de seguridad, ya que no olvidemos que, es hoy en día el sistema operativo con más activaciones a nivel mundial, con lo cual incomoda a su competencia quien podría tener mucho que ver con este desprestigio. O quizá sea verdad y tengamos en nuestro dispositivo móvil un agente infeccioso que ponga en riesgo nuestra información y terminemos pagando por un servicio no solicitado o incluso estemos siendo perseguidos por nuestros enemigos.

Pero, el esquema de seguridad de Android no únicamente se fundamenta en el exitoso esquema de seguridad de Linux, ya que por su naturaleza orientada a los dispositivos móviles trae implícitos otros ámbitos para tener en cuenta.

3.1 CARACTERÍSTICA DIFERENCIADORAS DE ANDROID

Android es un Sistema Operativo que presenta varios aspectos interesantes para ser estudiados, y que ha marcado diferencias con el resto de las plataformas móviles desde sus inicios.

Una de las características diferenciadoras es su ecosistema de desarrollo basado en software libre. Google permite a cualquier desarrollador la posibilidad de ofertar sus desarrollos de manera fácil mediante su Android Market. Abonando

al hecho que no hace falta herramientas sofisticadas o desconocidas para el desarrollo de aplicativos, teniendo como lenguaje estándar Java y herramienta de compilación Eclipse.

Quien quiera publicar sus desarrollos en el Android Market debe registrarse en la plataforma. Dicho registro debe ser validado con un pequeño pago mediante tarjeta de crédito, lo que para Google es una forma de validación al corroborar la identidad del propietario de esta tarjeta de crédito.

Esta característica es diferenciadora si se compara con el esquema que utiliza Apple con su Iphone, principal (o único) rival en el mercado de dispositivos móviles. Mientras Android permite la publicación libre de aplicaciones, Apple debe autorizar la publicación de cada aplicación en su tienda ITunes mediante una revisión manual de cada aplicación que solicita publicarse; esto fortalece la seguridad, pero vuelve excesivamente lento el proceso de publicación a la vez que es un freno para el caso de necesitar actualizar una aplicación por problemas de seguridad, por ejemplo.

Otra característica única de Android es el método automatizado que emplea para descubrir y remover software malicioso de Market. Google escanea y analiza la conducta de las aplicaciones a través del Android Market LifeCycle pudiendo incluso eliminar una aplicación del Market. Esto no garantiza una oportuna detección en todos los casos ya que pueden existir actividades que puedan presentarse como inofensivas, pero que en cierto contexto pudieran resultar maliciosas; por ejemplo, una aplicación necesita identificar cuando recibes un SMS para pausarse, pero pudiera reenviarlo a un sitio externo alojado en internet.

Y un tercer diferenciador es la declaración que cada aplicación debe realizar acerca de las API que utiliza del Sistema para que el usuario las autorice, pudiendo inclusive incluir permisos para que parte de una

aplicación sea utilizada por otra. Esto deriva en una responsabilidad de parte del usuario de revisar que privilegios le está otorgando a una aplicación en el momento de la instalación de esta. [21]

3.2 MODELO DE SEGURIDAD DE LINUX

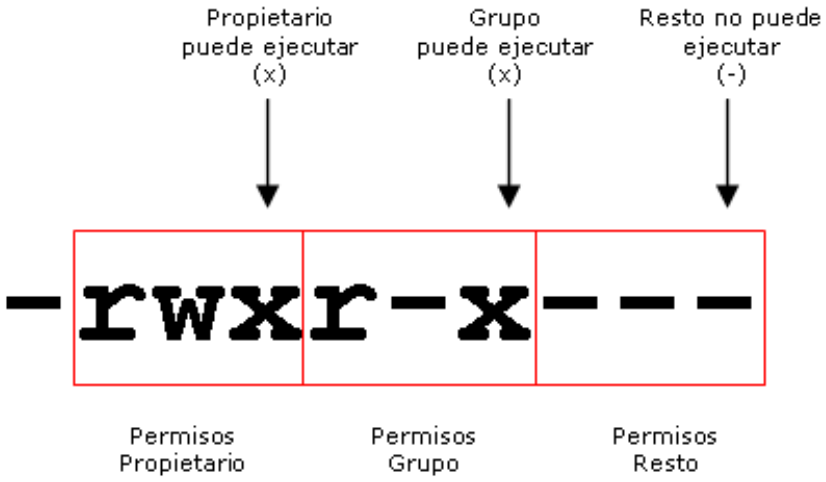
Como se ha mencionado anteriormente Linux es el núcleo de Android y gran parte del esquema de seguridad es una herencia del esquema manejado por Linux.

Cabe incorporar de manera inicial el concepto de que prácticamente todo en Linux es un archivo plano y como se va a describir más adelante estos están protegidos. Pero esto se traduce en una altísima responsabilidad sobre la persona que cuente con las credenciales de root [7] ya que un pequeño error puede cambiar por completo los permisos que poseen los grupos y usuarios. Es por esto por lo que por defecto se recomienda tener esta cuenta deshabilitada y acceder a esta lo mínimo posible. Y en esta parte se debe mencionar que también en Android este usuario esta deshabilitado por defecto.

La parte central del esquema de seguridad de Linux es el manejo de Usuarios y Grupos. A cada usuario en el momento de su creación se le asigna un número de identificación (UID) y se lo utiliza para diferenciar un usuario de otro. Adicionalmente cada usuario puede pertenecer a uno o más Grupos, y cada Grupo tiene también un número de identificación (GID) y que de igual manera que con los usuarios sirve para diferenciar un grupo de otro. Cada grupo puede tener varios usuarios y cada usuario puede pertenecer a varios grupos [21].

[7] En sistemas operativos del tipo Unix, root es el nombre convencional de la cuenta de usuario que posee todos los derechos en todos los modos (mono o multiusuario). root es también llamado súper usuario. Normalmente esta es la cuenta de administrador. El usuario root puede hacer muchas cosas que un usuario común no puede, tales como cambiar el dueño o permisos de archivos.

En Linux se realiza un control de acceso discrecional sobre los archivos por medio de la asignación de permisos. Se puede controlar al dueño del archivo, al grupo de usuarios al que pertenece el archivo, y al resto de los usuarios [22].



[23]

Imagen 11 Ejemplo permisos en Linux

Cada conjunto de permisos puede incluir leer (R), que permite a la entidad leer el archivo; escritura (W), que permite a la entidad a escribir o actualizar el archivo; y ejecución (X), que permite ejecutar el archivo. Se debe tener en cuenta que tener permiso de lectura no implica que tenga permiso de escritura, y viceversa.

Los permisos en Linux se basan adicionalmente en la idea que, si no se concede cierto derecho sobre un archivo, este no lo tiene. Esto quiere decir, por ejemplo, que, si se concede permisos de lectura y escritura para el dueño y su grupo, si el usuario no es el dueño o no pertenece a ese grupo, pues simplemente no tiene permisos sobre ese archivo.

3.3 MODELO RESULTANTE EN ANDROID

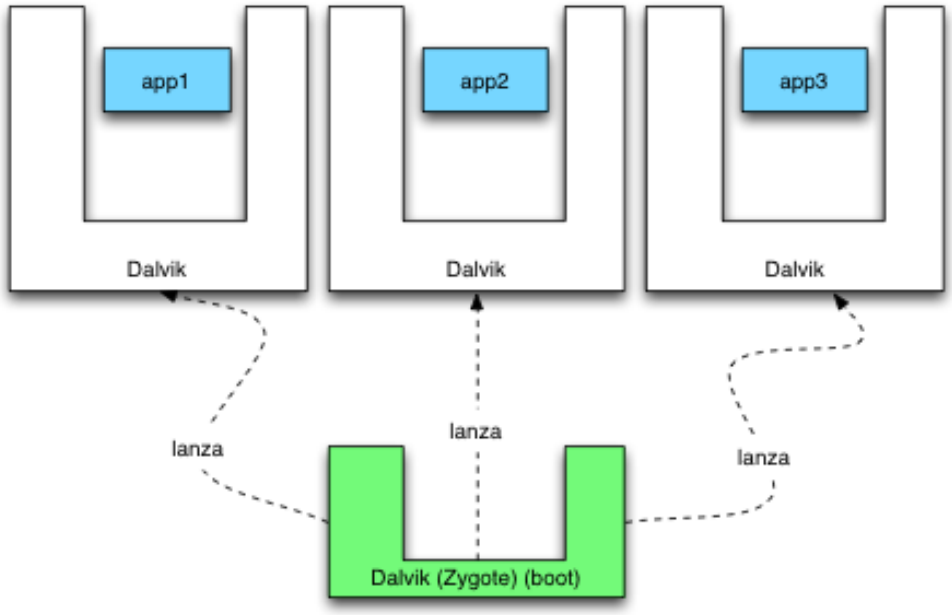
Como ya se mencionó el modelo de seguridad de Linux está basado en identificadores de grupos y usuarios (GID y UID), y también se observó que cada grupo puede tener varios usuarios y cada usuario puede pertenecer a varios grupos. Debido a que Android utiliza el Kernel de Linux como núcleo, estos conceptos también son aplicados.

Cuando se instala una nueva aplicación en Android, a esta se le asigna un identificador de usuario (UID) y esta nueva aplicación se ejecuta en función de este UID, además todos los recursos generados por esta aplicación se le asignan el mismo UID, es decir que, a estos recursos se podrá acceder únicamente si el ID del Usuario que lo solicita coincide con el de la aplicación que los genera. Evitando que aplicaciones con diferentes identificadores accedan a otros datos, procesos o espacios de memoria. Obviamente este UID asignado es diferente en cada dispositivo ya que se lo asigna directamente por el Sistema Operativo al momento de su instalación [21].

Este esquema de seguridad ha perdurado en el tiempo en las diferentes versiones, siendo de las pocas cosas que no se han tocado dentro del siempre cambiante mundo de Android.

Para poder entender este concepto de mejor manera vale recordar parte de lo explicado en la parte concerniente a la Arquitectura del Sistema Operativo, pues habíamos visto que la Máquina Virtual Dalvik ejecuta para cada aplicación una Máquina Virtual a partir de una Maquina Zygote.

Este concepto cabe recordarlo para reforzar la idea que tiene Android de aislamiento de cada aplicación para evitar la interacción con otra. De esta manera al igual que con la identificación de UID, para que únicamente la aplicación dueña de los recursos generados tenga acceso total sobre estos, el manejo de máquinas virtuales independientes ejecutándose por cada aplicación iniciada permite aislar las aplicaciones y sus recursos de otras.



[10]

Imagen 12 ZYGOTE DVM

Hemos visto que las aplicaciones en Android corren en procesos y máquinas virtuales diferentes; pero adicionalmente cada aplicación puede incluir código nativo, es decir, código que se ejecuta por fuera de la Máquina Virtual Dalvik, este código es incluido por ciertas aplicaciones para que se ejecute directamente en el procesador del dispositivo.

Cabe recalcar que este concepto de aislamiento de aplicaciones se aplica de igual manera en código ejecutado en la Máquina Virtual Dalvik o en código nativo.

Cabe recordar que las librerías nativas son las base que garantizan la ejecución de las aplicaciones, estas muchas veces son personalizadas por cada fabricante, al igual que cualquier aplicación que se ejecuta sobre Android goza del esquema de seguridad basado en identificadores de usuario.

3.3.1 AISLAMIENTO DEL SISTEMA DE ARCHIVOS[8]

El Kernel de Linux dota al sistema de archivos de un método de aislamiento, de tal manera que únicamente la aplicación con el UID de propietario pueda tener control total sobre los recursos generados.

La manera estándar en que Android establece el sistema de archivos dentro del dispositivo es crear una carpeta para cada aplicación dentro del directorio ***/data/data/app_package_name***, y es configurado de tal manera que se dan permisos sobre este únicamente para aplicación con el UID creado al momento de la instalación, y no existen otros permisos ya que no existen creados usuarios o grupos con permisos globales. Dentro del directorio antes descrito es en la carpeta ***/files*** donde se almacenarán los recursos generados por la aplicación, a estos recursos nuevos también se les vinculará con el UID propietario, siendo este el único con permisos, aislando de esta manera los archivos a otras aplicaciones.

Se pueden plantear cuatro advertencias importantes a tomar en cuenta:

[8] Un sistema de archivos son los métodos y estructuras de datos que un sistema operativo utiliza para seguir la pista de los archivos dentro del almacenamiento físico (por ejemplo, disco duro); es decir, es la manera en la que se organizan los archivos en el disco duro.

- ✓ Debido a que el aislamiento de archivos se basa en los UID, las aplicaciones que están configuradas para funcionar con el mismos UIDs pueden acceder a los archivos de cada uno. Esta excepción se plantea en el diseño, ya que las aplicaciones pueden funcionar con el mismo UID sólo si son configuradas de esa manera.

- ✓ Si un usuario accede al dispositivo utilizando el UID root será capaz de evitar restricciones de acceso en cualquier archivo, lo que les permitiría el acceso a los datos almacenados por alguna aplicación, sin importar con que UID fue creada. Esto es de alta importancia ya que es relativamente fácil encontrarse con tutoriales en internet que enseñan como “rootear” un dispositivo Android.

- ✓ Todos los datos escritos en la unidad externa, como tarjetas SD, carecen del control de acceso basado en Linux. Por lo tanto, cualquier archivo guardado en el almacenamiento externo es accesible por otra aplicación en el dispositivo.

- ✓ Como desarrollador, se puede especificar diferentes permisos en los ficheros.

3.4 PERMISOS DE LAS APLICACIONES

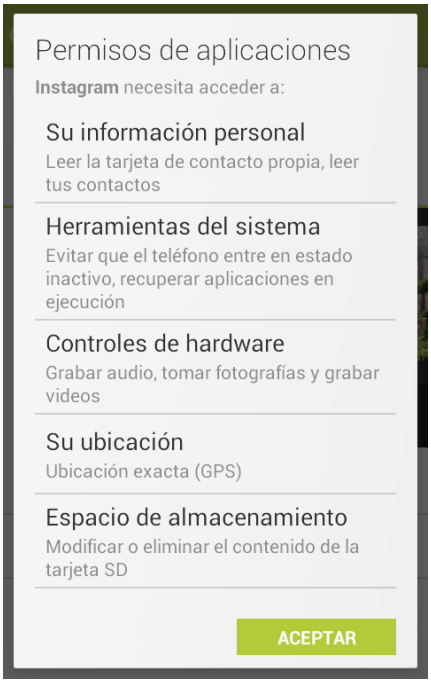
Android enfoca la ejecución de aplicaciones de manera diferente si lo comparamos con sistemas operativos tradicionales de escritorio. En un sistema operativo de escritorio (Windows, por ejemplo) las aplicaciones que se ejecutan corren con los mismos permisos que se hayan otorgado al usuario que ejecuta la aplicación, sin importar que tipo de aplicación sea la que se ejecute esta tendrá los mismos permisos de la cuenta de usuario que la inicia, es decir, tanto un editor de texto como un explorador de internet o una aplicación de VoIP se ejecutan con los mismos privilegios en una misma cuenta de usuario. Entonces si un usuario inicia sesión con privilegios de súper administrador, todas las aplicaciones que este usuario ejecute correrán con privilegios de súper administrador.

En el modelo de Android cada aplicación corre con su propia cuenta de usuario, lo cual tiene el efecto de separación y aislamiento entre aplicaciones, garantizando que los recursos almacenados en el dispositivo sean accedidos únicamente por la aplicación que los generó.

Luego Android va más allá incluyendo un sistema de permisos para los servicios que utilizan las aplicaciones instaladas en el dispositivo. Para hacer uso de otros pedazos código de Android que pudieran ser sensibles o peligrosos, como el acceso a los datos personales de un usuario o el acceso a internet, se debe conceder dicho permiso al momento de la instalación de la aplicación.

Al momento de iniciarse la solicitud de instalación de una aplicación se presenta al usuario un manifiesto para que él revise detalladamente los permisos solicitados y decida aceptar o no la instalación. Esto tiene como objetivo mantener informado al usuario de los recursos que está brindando a las aplicaciones que instala.

Este modelo presenta dos ventajas principales en relación con otros sistemas tradicionales. Primero permite al usuario ver antes de la instalación todas las cosas peligrosas que la aplicación pudiera hacer ya que la aplicación debe incluir en el manifiesto los permisos solicitados o no podrá hacer uso de esos permisos. Digamos que el usuario desea instalar una aplicación que se ejecuta localmente y solicita permisos para enviar mensajes SMS, entonces el usuario puede decidir no instalar la aplicación al considerar que está solicitando permisos que no corresponden con el objetivo de la aplicación y por lo tanto la considera peligrosa. Esto en el caso de sistemas tradicionales de escritorio no sucede ya que el usuario instala la aplicación sin saber que permisos le está concediendo.



[24]

Imagen 13 Permisos de Aplicaciones

En segundo lugar, este modelo permite la contención de un posible atacante a una aplicación legítima. Para entender mejor esto usemos como ejemplo un navegador web en un tradicional sistema operativo de escritorio, supongamos que un atacante logra explotar una vulnerabilidad del explorador de internet y consigue ejecutar su propio código, este código se ejecutará con todos los permisos que se le hayan concedido a la cuenta de usuario que lo ejecuto, lo cual pone a nuestro equipo en un alto riesgo. Con el modelo de permisos utilizado por Android ahora supongamos que este mismo atacante logra ejecutar su propio código fuente explotando una vulnerabilidad del navegador de internet, este código se ejecutará únicamente con los permisos concedidos a la aplicación y además que por el aislamiento antes discutido se verá imposibilitado de utilizar recursos de otras aplicaciones.

Luego de haber discutido esto una inquietud cae por peso propio ***¿Los usuarios realmente leen los permisos que conceden a sus aplicaciones?***, y más allá de que si los leen o no ***¿Están conscientes del riesgo que conllevan estos permisos?*** Y estas preguntas son válidas ya que es en la decisión del usuario en lo que se basa el éxito del modelo de permisos que utiliza Android.

Fundamentalmente esto es un cambio en el paradigma de ejecución de aplicaciones y se lo ha tratado de hacer más comprensible acompañando un resumen básico y entendible de lo que implica cada permiso concedido. Básicamente esto se relaciona directamente con el nivel concientización que el usuario tenga con respecto a la seguridad de su dispositivo y con el uso de su sentido común.

También cabe recalcar que el método empleado es de carácter acepta todo o nada, es decir, no existe la posibilidad que un usuario “edite” los permisos que una aplicación le está solicitando. Todo se resume en aceptar los permisos solicitados e instalar la aplicación o simplemente no aceptar esta solicitud y por ende prohibir la instalación de la aplicación.

3.4.1 NIVELES DE PROTECCION

Antes de detallar los niveles de protección bien vale indicar que los desarrolladores pueden crear sus propios permisos, pero únicamente para el caso de que deseen compartir parte de su aplicación con otra, por ejemplo, reutilizar métodos o servicios.

Como se ha venido explicando los permisos que se consideran sensibles deben ser autorizados por el usuario al momento de la instalación, para identificar de mejor manera estos permisos más sensibles de otros menos sensibles Android los ha clasificado en niveles, siendo estos los siguientes [25]:

3.4.1.1 NORMALES

Es el valor por defecto. Un permiso de menor riesgo se da a las aplicaciones aisladas con un riesgo mínimo para otras aplicaciones, el sistema o el usuario. El sistema otorga de forma automática este tipo de permiso para una aplicación que solicita la instalación, sin pedir la aprobación explícita del usuario. Los usuarios pueden revisar ellos, pero pueden no ser advertidos explícitamente.

3.4.1.2 PELIGROSOS

Un permiso de mayor riesgo que le daría a las aplicaciones que solicitan acceso a los datos privados del usuario o el control sobre el dispositivo que puede influir negativamente en el usuario. Debido a que este tipo de permiso introduce riesgo potencial, el sistema no los puede

conceder automáticamente a la aplicación solicitante. Por ejemplo, los permisos peligrosos solicitados por una aplicación se pueden mostrar al usuario y requieren confirmación antes de proceder, o algún otro método para evitar que el usuario de forma automática conceda estos permisos.

3.4.1.3 FIRMADOS

Antes de conceder el permiso, el sistema verifica que el certificado digital que solicita el acceso es el mismo que, el de la aplicación que lo debe otorgar. Si los certificados coinciden, el sistema automáticamente asigna el permiso sin informar al usuario, es decir, no hace falta la aprobación explicita del mismo.

3.4.1.4 FIRMADOS O DE SISTEMA

Este es un permiso que el sistema concede sólo a las aplicaciones que se encuentran en la imagen del sistema Android o que están firmados con el mismo certificado de la aplicación que declaró el permiso. Este permiso se utiliza para ciertas situaciones especiales en las que múltiples proveedores tienen aplicaciones creadas en una imagen del sistema y necesitan compartir características explícitamente porque se están construyendo juntos.

3.5 IPC (INTERPROCESS COMMUNICATION) COMO MECANISMO DE SEGURIDAD

En Android, cada aplicación al iniciar su ejecución, lo hace con su propio proceso dentro de lo que se denomina Sandbox. Este proceso tiene un PID único, así como un usuario con UID. Esto permite a las aplicaciones desenvolverse en su propio entorno, aisladas del resto de aplicaciones. [26]

Este aislamiento ayuda a mejorar el rendimiento del sistema en general y a lograr una mayor seguridad a través de diferentes niveles de permisos y privilegios de acceso.

En el sitio oficial de Android para desarrolladores [27], se encuentra una forma práctica de definir IPC de la siguiente manera:

Por lo general, en Android, un proceso no puede acceder a la memoria de otro proceso. Por esto, para comunicarse, tienen que descomponer sus objetos en primitivas que el sistema operativo pueda comprender y tienen que ordenar los objetos entre esos límites para ti.

Esta forma de intercomunicación trae consigo varias ventajas, entre las más importantes:

- Compartir información entre dos o más aplicaciones
- Optimizar recursos del dispositivo
- Manejo modular de la información
- Resulta conveniente para los desarrolladores especificar que compartir

3.5.1 MODELOS DE COMUNICACIÓN

A través del IPC se puede intercambiar información o datos entre procesos que se ejecutan de manera independiente con diferentes identificadores de usuario.

En este contexto encontramos dos modelos de intercambio de información, por una parte, mediante la aplicación de espacios de memoria compartida. La segunda opción de modelo de comunicación que permite IPC es mediante el envío de mensajes.

Como hemos podido observar, en el primer método se abstrae a las aplicaciones del sistema operativo en el que se ejecutan, directamente utilizan un espacio de memoria compartida en el que se alojarán los recursos que entre ellos se quieran compartir. Por otra parte, a través del envío de mensajes, se utiliza al Kernel de Linux como un “negociador”.

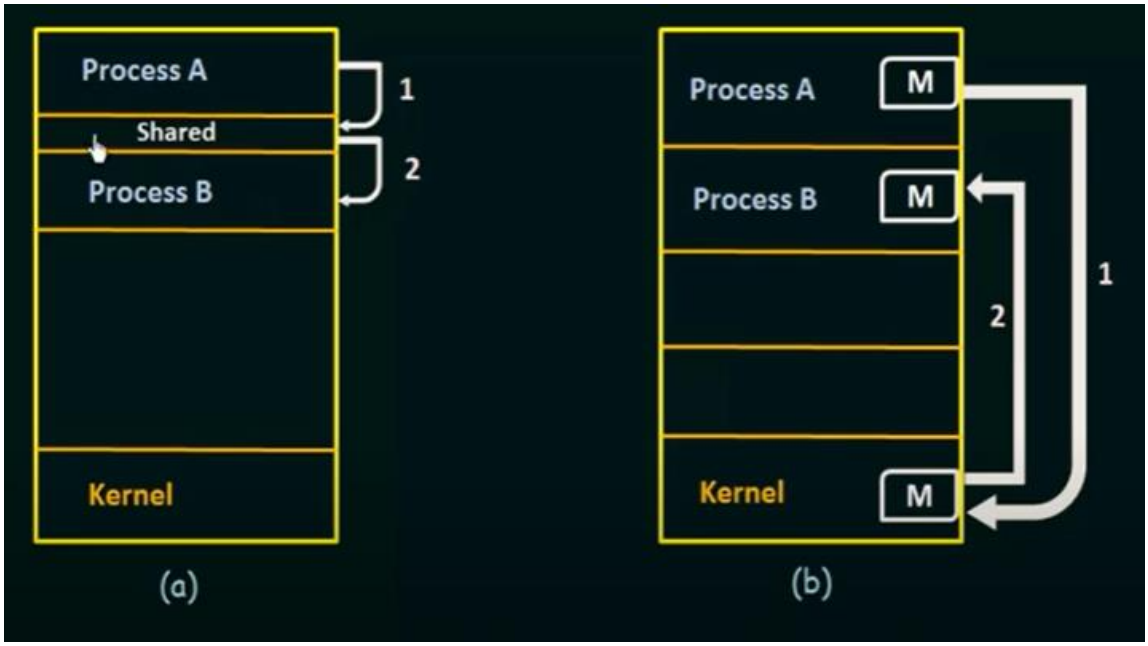
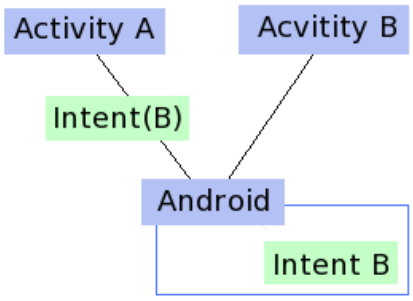


Imagen 14 Modelos de comunicación

Android implementa algunas herramientas clave que se utilizan para comunicarse entre las aplicaciones de forma segura. Estos mecanismos dan a las aplicaciones de Android la capacidad de ejecutar procesos en segundo plano, ofrecer servicios para ser consumidos por otras aplicaciones, compartir de forma segura los datos relacionales, iniciar otros programas y reutilizar los componentes de otras aplicaciones de forma segura.

Gran parte de la IPC (***Comunicación entre Procesos***) que se produce en Android se hace a través de la que pasa alrededor de una estructura de datos llamada Intenciones. Son un mecanismo que Android específica para mover datos entre procesos. La data en cada una de estas Intenciones es un hipervínculo y podría apuntar a un archivo, contacto, página web, número de teléfono, y así sucesivamente. Las Intenciones también pueden, potencialmente, tener una colección de pares de clave/valor llamado extras, así como banderas, componentes, y otras características más avanzadas.



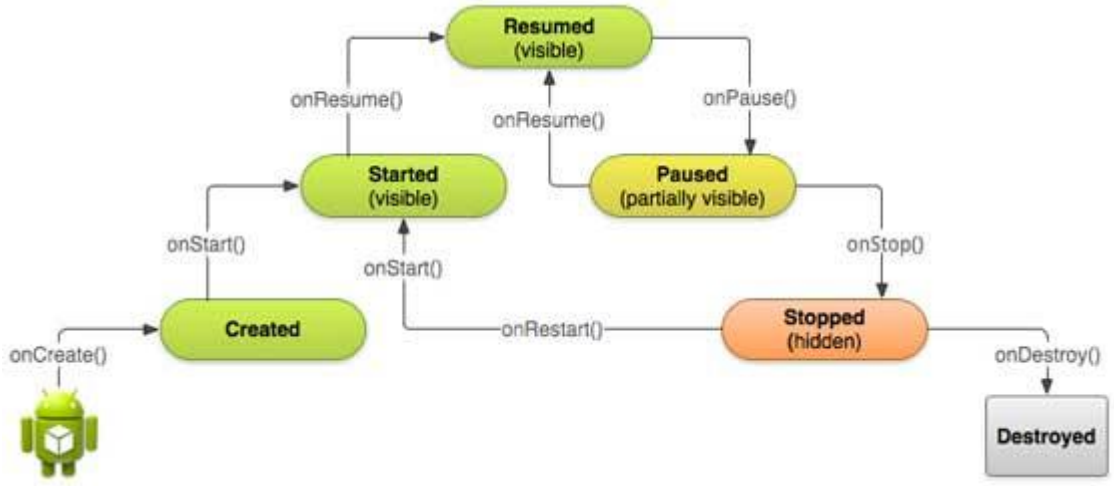
[28]

Imagen 15 Android Intent Activity

Cada uno de estos mecanismos de IPC utiliza Intenciones de alguna manera y es, probablemente, algo familiar para la mayoría de los desarrolladores de Android. Sin embargo, porque el uso de estos de forma segura es clave para la seguridad de Android, vamos a revisar brevemente cada mecanismo.

3.5.2 ACTIVIDADES

Este tal vez sea el componente más popular entre los desarrolladores de aplicaciones, se puede hacer la analogía con las pantallas que se muestran al usuario, es decir, se puede considerar una actividad a cada interface que se dibuje en la pantalla del dispositivo por parte de una aplicación. Cuando se programa una aplicación se deben construir las clases que derivan de la clase base actividad que contienen objetos con los que el usuario puede interactuar, por ejemplo, botones. En el argot de los desarrolladores se puede describir a estas como ***capa de aplicación.***



[29]

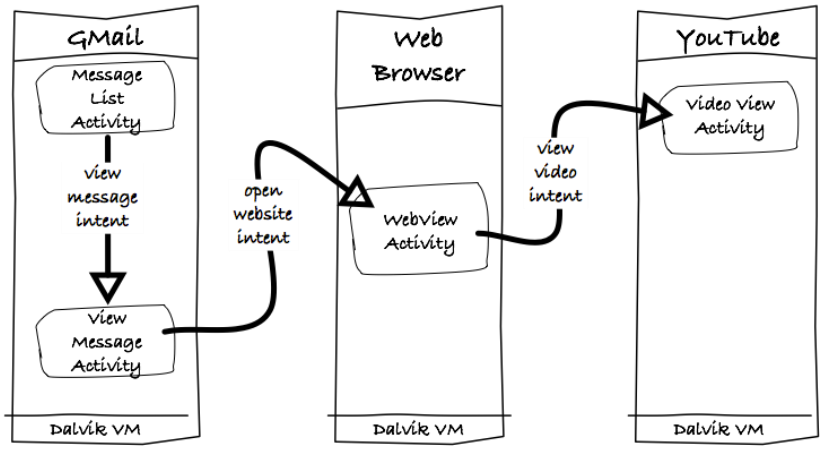
Imagen 16 Ciclo de Vida de las Actividades

Para poder asegurar las actividades, simplemente basta con incluir el permiso requerido dentro archivo manifiesto ***AndroidManifest.xml***, entonces cada vez que otra aplicación quiera iniciar la actividad se verificará que la misma tenga los correspondientes permisos.

3.5.3 DIFUSIONES (BROADCAST)

Las difusiones en forma de broadcast proporcionan una forma de enviar mensajes entre aplicaciones, por ejemplo, alertar a las aplicaciones “oyentes” durante el paso del tiempo de un mensaje entrante u otros datos. Cuando una aplicación envía una difusión coloca el mensaje que se enviará en una Intención.

Las aplicaciones se mantienen escuchando a la espera de un Intent que contenga información que sea de interés, supongamos la recepción de un mail, que contiene un link a un video de youtube, la primera difusión es de interés del administrador de correo, luego del explorador de internet y por último de existir la aplicación de youtube [30].



[31]

Imagen 17 Broadcast de Intent

Al recibir un correo se envía una difusión, con lo cual pudieran existir varios componentes que pudieran escuchar esta difusión, lo más seguro es que queramos limitar los componentes que puedan escuchar esta difusión de nuevo mail. El emisor puede solicitar un permiso a Android por cada emisión que envía, de tal manera que únicamente los receptores con el permiso pertinente puedan escuchar.

3.5.4 LENGUAJE DE DEFINICIÓN DE LA INTERFAZ DE ANDROID AIDL

La comunicación entre procesos o IPC, en resumen, es un mecanismo que permite que múltiples procesos independientes se comuniquen e intercambien datos. Esta comunicación es posible en muchos casos mediante el uso de interfaces compartidas definidas en el Lenguaje de Descripción de Interfaz IDL. En el caso particular de Android se utiliza el Lenguaje de Definición de la Interfaz de Android AIDL, mediante este, se establecen un conjunto de reglas a que los procesos interesados deberían seguir. [32]

Este método se debe utilizar si se necesita una estructura IPC multiproceso. Si no va a realizar operaciones simultáneas, puede elegir otro de los métodos disponibles en lugar de AIDL [33].

En este método, una aplicación llama al método de otra aplicación en el mismo dispositivo, a saber, RPC (llamada a procedimiento remoto). Estos métodos se pueden llamar desde muchos procesos o subprocesos diferentes simultáneamente.

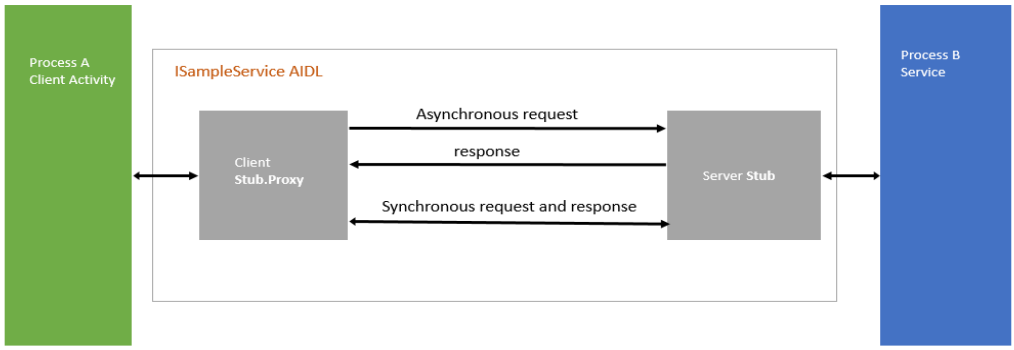


Imagen 18 Descripción comunicación AIDL

3.5.5 SERVICIOS

Los servicios son procesos en segundo plano, típicamente consiste en la actualización de bases de datos, siempre y cuando sean notificaciones de un evento externo, o realizar alguna otra tarea en beneficio de un componente que interactuará con el usuario, es importante asegurarse de que son accesibles sólo por los consumidores adecuados, esto nuevamente basado en el esquema de permisos.

Android define un servicio como un componente de una aplicación que puede realizar operaciones de larga ejecución en segundo plano y que no proporciona una interfaz de usuario [34].

Los servicios pueden, por ejemplo, reproducir música; otros manejan los mensajes instantáneos entrantes, transferencias de archivos, entre otros. Los servicios se pueden iniciar mediante un Intent.

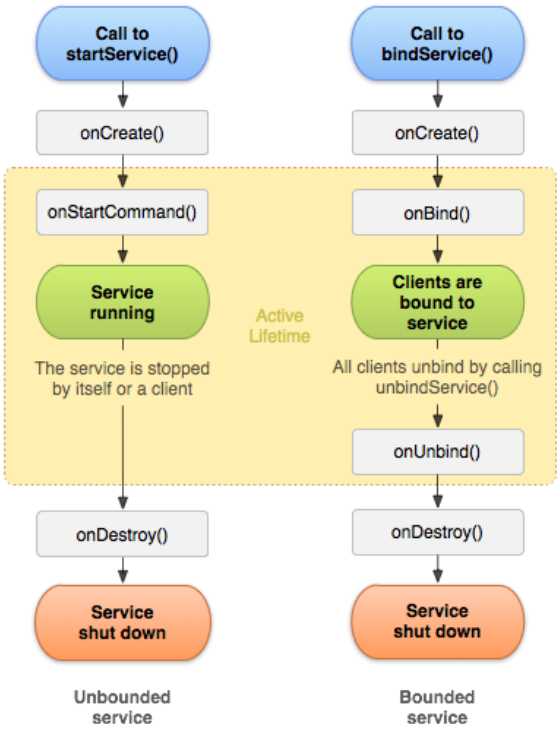


Imagen 19 Ciclo de Vida de un Servicio

Se pueden detallar dos grupos de servicios:

- Started: una vez iniciado el servicio se podrá ejecutar de forma indefinida realizando una sola operación cada vez. Se inicia llamando a "startService()".
- Bound: un servicio con el que podremos interactuar enviándole solicitudes, obteniendo resultados e incluso realizar comunicaciones IPC. Se inicia con "bindService()".

3.5.6 PROVEEDORES DE CONTENIDOS

Los proveedores de contenido son la forma estándar para que las aplicaciones puedan hacer que sus datos estén disponibles para otras aplicaciones. Como existen estos componentes para compartir datos entre

diferentes consumidores y proveedores, plantean la necesidad de un modelo de seguridad más compleja.

A diferencia de las actividades y servicios, los proveedores de contenidos pueden especificar dos tipos de permisos, lectura y escritura. Permitiendo a las aplicaciones ser configuradas con el principio de permiso mínimo.

Con los proveedores de contenidos, los permisos se verifican primero cuando se conecta al proveedor. Al hacerlo, puede conectarse a ella si se tiene cualquiera de los permisos de lectura o el permiso de escritura. Si no tiene ninguno, la solicitud de conexión fallará y se lanzará una Security Exception [21].

3.6 PROTECCION DE DATOS ALMACENADOS

Los dispositivos con Android, al igual que cualquier dispositivo móvil tienden a incluir una memoria interna limitada, es decir, sin capacidad de aumentar su capacidad de almacenamiento. En algunos dispositivos esta memoria interna puede almacenar grandes cantidades de información, pero en otros casos este almacenamiento resulta insuficiente; por lo que es habitual que la capacidad de almacenar deba ser incrementada en algunos casos con tarjetas de memoria externas.

El almacenamiento de datos en estos sistemas de archivos es un poco difícil. Para que sea más fácil para los usuarios mover los datos de un lado a otro entre cámaras, computadoras y Android, el formato de estas tarjetas es VFAT, que es un estándar que tiene ya mucho tiempo de funcionamiento y que no soporta los controles de acceso de Linux. Por lo tanto, los datos almacenados aquí no están protegidos y son de fácil acceso por cualquier aplicación en el dispositivo.

Se debería informar a los usuarios de que el almacenamiento a granel se comparte con todos los programas en su dispositivo, y disuadirlos de no poner datos muy sensibles allí. En caso de necesitar almacenar datos confidenciales, estos pueden cifrarse y almacenar la pequeña llave en el área de archivos de la aplicación y el texto cifrado grande en la tarjeta de memoria compartida. Siempre que el usuario no quiera utilizar la tarjeta de almacenamiento para mover los datos a otro sistema, esto debería funcionar.

3.6.1 AMENAZAS Y VULNERABILIDADES EN LOS DATOS ALMACENADOS

Ya hemos visto cómo el sistema Android segrega los datos almacenados por las aplicaciones de otras aplicaciones. También hemos visto como las aplicaciones pueden compartir datos entre ellas mismas y de cómo se utilizan los permisos y otros mecanismos de control de acceso para especificar qué tipo de dichos accesos están permitidos. Todas estas protecciones que se ofrecen a los datos almacenados o datos en reposo se basan en los mecanismos del sistema Android corriendo asumen que el sistema está en marcha y funcionando como debería y que todos los accesos a los datos almacenados sean a través de estos canales controlados. Por desgracia, esto no es siempre el caso [35].

3.6.1.1 VULNERABILIDADES

Como se mencionó antes, típicamente la mayor parte de información se almacena en un dispositivo externo (mayormente tarjeta SD), esto debido a que por lo regular tienen mucha mayor capacidad de almacenamiento. El problema en este caso radica en que este tipo de memorias son de fácil acceso desde cualquier computador; y el problema se profundiza al saber que por el sistema de archivos que estos manejan no se aplican las directivas de seguridad de Linux. Es decir, para cualquier atacante es fácil desmontar la tarjeta SD y apropiarse de toda la información que allí se almacene.

Otra vulnerabilidad de alto compromiso es la posibilidad existente, y no muy difícil, de acceder a la raíz del sistema con el usuario root. Esto debido a la relativa facilidad de ***"rootear"*** el dispositivo. Además, es importante mencionar que existe malware que se aprovecha de vulnerabilidades en el Kernel de ciertas versiones de Android para acceder

a la raíz del propio dispositivo. Al tener los privilegios de súper administrador de deja totalmente de lado el esquema de seguridad ya que se posee acceso total.

3.6.1.2 AMENAZAS

Aquí entra en juego el análisis de riesgo, que tan perjudicial puede resultar que un tercero aprovechando una vulnerabilidad tenga acceso a la información generada por una aplicación. No es lo mismo que la aplicación sea un juego que se utilice como simple entretenimiento, en este caso si un atacante logra acceder a la información de la aplicación, por ejemplo, puntuación, record o nombre de usuario; quizá esto no lo estimemos como perjudicial.

Pero, si la aplicación guarda información de mayor complejidad, digamos, un administrador de redes sociales; entonces si el mismo atacante tiene acceso a su perfil con información personal, ahí si estaríamos hablando de un mayor impacto y un nivel de riesgo más alto.

3.6.1.3 PRINCIPIOS DE PROTECCIÓN

La mejor forma de protección de los datos almacenados se logra utilizando una serie de mecanismos de protección que se refuerzan mutuamente. Por ejemplo, si los permisos adecuados se establecen en el almacén de datos que la aplicación está utilizando, otras aplicaciones no serán capaces de leer o modificar esos datos en un sistema de Android corriendo normalmente. Hemos visto que existen mecanismos que permitirían comprometer estos datos. Los datos sensibles también deben ser cifrados, de manera que, si un atacante es capaz de poner en peligro la protección ofrecida por el Sistema Operativo y obtener acceso a los datos, no pueda hacer nada con ella. La implementación de

diferentes capas como esta es una práctica común de seguridad de información conocida como la Defensa de Profundidad (DiD).

También hemos visto que las medidas adoptadas para proteger los datos almacenados deben basarse en el riesgo de compromiso.

La criptografía moderna nos da amplias y seguras soluciones, como el cifrado simétrico y asimétrico y funciones de hash, que son perfectamente compatibles con las últimas versiones de Android. Aquí vale mencionar que se han visto dispositivos con Android que como parte de la personalización decidida por el fabricante han eliminado la función de cifrado de datos.

4 ANDROID EN LAS PYMES EN LATINOAMERICA

Para septiembre de 2013 se contaba con 1,5 millones de activaciones diarias de Android, se estima que en total serían ya 1000 millones los dispositivos que cuentan como Sistema Operativo Android [36].

Esta realidad se traslada efectivamente al ambiente laboral, sobre todo en las PYMES[9], ya que como es bien conocido, este tipo de organizaciones se nutre laboralmente hablando de jóvenes profesionales, justamente el sector de jóvenes entre 18 y 30 años es el mercado que más se ve atraído a consumir dispositivos con alguna versión de Android.

Adicional algo que diferencia dentro de la cultura organizativa a las nuevas generaciones es el uso masificado del acceso a internet; según una encuesta realizada por Telefónica, cuyos resultados son expuestos por la agencia de noticias argentina Télam [37], en américa latina el 70% de usuarios que usan internet tienen menos de 35 años y el 78% de jóvenes entre 18 y 30 años poseen un Smartphone. Entonces, queda claro que el impacto de estos dispositivos en todas las actividades socioeconómicas es significativo.

Ahora bien, el uso masificado de dispositivos móviles en el trabajo ha logrado cambiar drásticamente los métodos con los que los empleados interactúan con la empresa, su información y entre ellos. Pues actualmente podemos acceder con nuestro Smartphone o Tablet a un archivo necesario para cumplir con nuestras funciones sin necesidad de estar físicamente en nuestro lugar de trabajo, o mantener largas reuniones de planificación con nuestros compañeros de trabajo usando alguna de las tantas aplicaciones disponibles para chat o videoconferencia.

[9] PYMES: Pequeñas y Medianas Empresas

Para las nuevas generaciones es impensable un día de trabajo sin revisar un mail o coordinar actividades sin el uso de su dispositivo móvil, es decir, nuestros dispositivos son ya una importante herramienta de trabajo.

4.1 BRING YOUR OWN, NUESTRO DISPOSITIVO COMO HERRAMIENTA DE TRABAJO

Se cuentan ya por millones los consumidores que están comprando dispositivos móviles avanzados, como smartphones y tablets, para uso personal, a los mismos que dotan de aplicaciones que hacen su vida más fácil, dispositivos que tienen interfaces de usuario intuitivas, listos para usarlos en videoconferencias, por ejemplo.

Estos dispositivos no son solo para usos personales y de entretenimiento, sino también para fines laborales. Cada vez más, las personas están llevando estos dispositivos a su trabajo para integrarlos en su flujo laboral diario. A esta tendencia se la conoce como “Bring Your Own Device" o BYOD (Traiga su propio dispositivo) [38].

Esta realidad que afecta a todos los tipos de empresa sin importar su tamaño o modelo de negocio, y se ha convertido en una preocupación para la alta gerencia, pero sobre todo para los responsables de la TICs.

Y es que, según Kaspersky[10] el 60% de empresas en América Latina piensa que BYOD es una amenaza creciente para su seguridad, esto ocasionado por el uso indebido de dispositivos móviles, lo cual causas incidentes de seguridad, por ejemplo, perdida de información crítica [39].

[10] Kaspersky: empresa proveedora de herramientas de seguridad como antivirus, por ejemplo.

Para ISACA[11], las Compañías en América Latina están más Conscientes del Riesgo del BYOD, esto derivado del resultado de la edición 2012 de su encuesta IT Risk/Reward Barometer en la cual participaron más de 4,500 miembros de la asociación en todo el mundo. En la misma destaca que existe un aumento de un 16% de responsables de las TICs que creen que son más los riesgos que los beneficios de aplicar el BYOD en las empresas [40].

Sin embargo, a pesar de que las organizaciones miran como un riesgo el uso del BYOD en sus entornos, están muy lejos de restringir esta práctica, esto debido a que se destacan beneficios de eficiencia en la productividad del empleado, el aumento en la satisfacción y la reducción de costos.

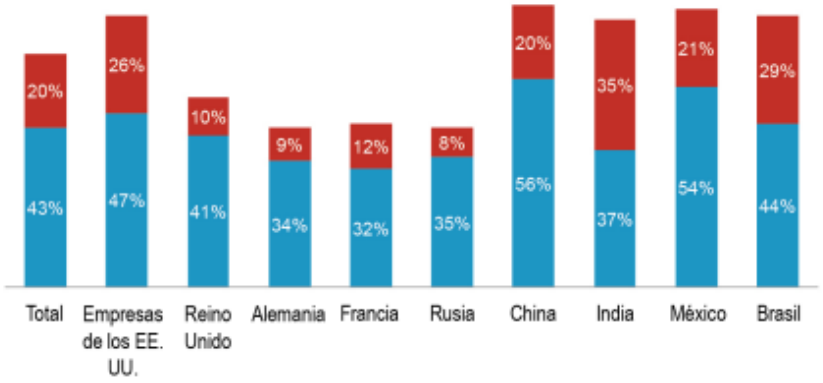
El 39% de los encuestados por Kaspersky en América Latina planea fomentar el uso de teléfonos inteligentes y tabletas personales en el trabajo.

La práctica del BYOD está modificando la forma en que las empresas administran sus redes, sus dispositivos móviles e incluso sus empleados, quienes están redefiniendo lo que significa estar “en la oficina” [38].

El estudio realizado por Cisco, a través de su centro de investigaciones “Horizons” va más allá en su análisis y asegura que BYOD es ya una tendencia bien afianzada en las Corporaciones de todo el mundo, y es que la cantidad de dispositivos por usuario en aumento es, en gran medida, consecuencia de BYOD. Por ejemplo, el 42% de los smartphones y el 38% de las computadoras portátiles utilizadas en el lugar de trabajo actualmente pertenecen a los empleados.

[11] ISACA es una asociación global sin ánimo de lucro de 140 000 profesionales en 180 países, gestiona varias certificaciones en el uso de las TICs

La tendencia marcada va para el alza y se espera un incremento hasta el 63% del uso de esta, según lo detalla el siguiente gráfico:



[38]

Imagen 20 Empresas que esperan que aumente el porcentaje de dispositivos pertenecientes a los empleados en los próximos años.

Para las empresas todavía queda mucho por hacer para que la tendencia BYOD sea segura, sobre todo en lo que tiene que ver a concientización. Este reto es todavía más complicado para las PYMES, que debido a los limitados presupuestos se ven relegados en temas de innovación.

CONCLUSIONES

Desde su irrupción, Android ha provocado un completo cambio en como las personas se interrelacionan con la tecnología. La estrategia de Google para masificar el acceso a la información contenida en internet necesita de un inmenso número de dispositivos conectados todo el tiempo. La estrategia es generar suficiente data de cada usuario, de tal manera, que sea propensa de comercializarse.

La necesidad de masificación de dispositivos móviles provocó una acelerada adaptación de las interfaces desde el punto de vista estético. Competir con un icono de la modernidad como el iphone, tuvo como consecuencia, que se priorizará el aspecto estético sobre la seguridad.

La arquitectura del Sistema Operativo Android está estructurada en una pila dividida en capas, de tal manera que permite a cada capa, servirse de las capas anteriores. Esto ha logrado incrementar el número de desarrolladores interesados en Android ya que abstrae a las aplicaciones del entorno en el que se van a ejecutar.

El hecho que tanto el sistema operativo como las aplicaciones sean desarrollados mediante software libre, elimina la barrera de entrada. El Kernel de Linux se responsabiliza de la gestión de controladores, recursos y procesos. Con lo cual el entorno de ejecución solamente debe preocuparse de la gestión de sus propios recursos sin importarle marca, modelo o cualquier otra característica en la que se va a ejecutar.

Android basa su esquema de seguridad en las fortalezas que le brinda el propio esquema que maneja Linux, creando para cada aplicación un id de usuario, de tal manera que solamente se pueda acceder a los recursos creados sí coinciden las identificaciones. En este contexto es vital el concepto de que si un permiso no está declarado simplemente no existe.

A breves rasgos pudiéramos resumir el esquema de seguridad de Android en tres pilares fundamentales:

- ✓ Cada aplicación tiene su UID,
- ✓ Separación y asilamiento entre aplicaciones y sus recursos
- ✓ Los permisos deben ser concedidos durante la instalación

Así mismo, desde el punto de vista de la protección a los datos almacenados se deben considerar ciertas cuestiones para garantizar su seguridad, a decir:

- ✓ Solamente si así se configuran, dos aplicaciones pueden compartir el mismo UID y por lo tanto compartir recursos
- ✓ Por defecto los sistemas Android deshabilitan el usuario root, ya que este podría acceder a los datos de cualquier aplicación rompiendo con la separación y el aislamiento
- ✓ Las tarjetas externas de almacenamiento tienen un formato VFAT que escapa de los controles establecidos por el Kernel Linux
- ✓ El desarrollador es quien específica los permisos de cada aplicación

El principio de protección de datos de Linux se basa en el refuerzo mutuo de varios mecanismos de autoprotección a decir los más importantes: otorgamiento de permisos por parte del usuario, posibilidad de cifrar datos sensibles y buenas prácticas del usuario. Este conjunto de acciones puede conocerse como defensa en profundidad.

Para la intercomunicación entre aplicaciones, Android establece procedimientos dentro del entorno de trabajo, la herramienta principal se denomina IPC (Interprocess Communication). La información entre procesos se pasa alrededor de una estructura de datos llamada intent (intención). La data es un hipervínculo.

Las PYMES en general son el primer empleo de jóvenes que ingresan al mercado laboral. La nueva realidad incluye el uso de dispositivos móviles dentro de los entornos laborales, la mayoría de estos dispositivos son de propiedad de los mismos empleados, a esta tendencia se la conoce como Bring Your Own Dispositive BYOD. Esta tendencia ha cambiado la forma en que interactúan empleados y empresas y trae consigo muchas preocupaciones a ser resueltas en el futuro inmediato:

- ✓ 60% de las empresas ven al BYOD como una amenaza
- ✓ 16% de los responsables de tecnologías en las PYMES ven más riesgos que beneficios en esta tendencia
- ✓ 39% de las empresas planean fomentar el BYOD
- ✓ Actualmente al menos un 63% de dispositivos en los entornos laborales no son de propiedad de las empresas u organizaciones

Para finalizar es de destacar que el fuerte impacto que ha tenido la irrupción de la tecnología móvil en los mercados laborales en conjunto con la alta tendencia a utilizar Android como sistema operativo base para dispositivos crea una realidad preocupante, la concientización como base de prevención es la principal herramienta con la que se puede contar considerando los limitados recursos con los que cuentan las pequeñas o medianas empresas.

ANEXOS

EVOLUCIÓN ANDROID

El objetivo de este trabajo no es presentar un análisis en profundidad de cada versión de Android por lo que se resume en la siguiente tabla la evolución histórica de sus versiones destacando sus principales características.

EVOLUCION VERSIONES DE ANDROID [41] y [9]	
VERSION	CARACTERISTICAS
1.0 APPLE PIE	Primer sistema operativo móvil gratuito basado en el Kernel 2.6 de Linux. No movió mucho el mercado, pero introdujo características que hasta hoy son estándares para dispositivos móviles: :: Menú desplegable de notificaciones :: Widgets de escritorio :: Tienda de apps (Android Market) :: Integración con Google Mail, Contactos y Calendar :: Navegador, Maps, Google Talk, reproductor de YouTube y soporte para cámaras
1.5 CUPCAKE	Aparece luego de sólo 6 meses de Apple Pie y tres meses de Banana Bread (actualización de la 1.0), presenta mejoras enfocadas a la usabilidad como la inclusión de un teclado qwerty, mejora en la reproducción de videos, bluetooth, búsqueda en Google desde el escritorio y un SDK para desarrollo de Widgets para terceros.
1.6 DONUT	Presenta varias mejoras a la versión anterior como soporte a varios tipos de redes móviles, actualización del Android Market y mejora de la aplicación de cámara.
2.0 ENCLAIR	Presentada a finales del 2009 y supuso uno de los cambios más notorios, integrada en una alta gama de teléfonos móviles que todavía hoy son utilizados. Incluía Navegación GPS, compatibilidad con MS Exchange, reconocimiento de voz para escritura, soporte html5, entre otras. Con la venta de diez millones del Samsung Galaxy S pudo por primera vez superar la cuota de mercado de Apple.

2.2 FROYO	Lanzada en mayo del 2010 también presento numerosos cambios incluyendo una mentalidad para el uso empresarial. Incluía desbloqueo por PIN, rediseño de la pantalla Home, nuevo compilador para mejora de velocidad. Esta versión fue incluida en la Galaxy Tab, llamada a ser la competencia del IPad.
2.3 GINGERBREAD	Para fines de 2010 se presenta esta versión con un rediseño estético total, acompañado de varios modelos y marcas de teléfonos móviles, convirtiéndose en una de las versiones más extendidas siendo hasta hoy la segunda con más terminales activos. Además incluía soporte para tecnología NFC, uso de archivos ext4, y varias mejoras de usabilidad.
3.0 HONEYCOMB	Exclusiva para tablets, presentaba un nuevo rediseño en lo estético y eliminaba los botones físicos.
4.0 ICE CREAM SANDWICH	Optimización de Honeycomb para smartphones. Presentaba el desbloqueo mediante el reconocimiento de rostro, nueva función para gestión de consumo de datos.
4.1 JELLY BEAN	Es la versión con mayor número de activaciones, destaca la eliminación de soporte para flash, nuevas mejoras en el rendimiento, nuevas notificaciones en el escritorio.
	La versión 4.2 mantuvo el mismo nombre y presentaba nuevas mejoras en el rendimiento, soporte para varios perfiles de usuario, gestual mode para personas invidentes. Una nueva versión la 4.3 presentaba :: Soporte multiusuario y de perfiles mejorado :: Soporte OpenGL ES 3.0 :: Compatible con TRIM :: Bluetooth Smart :: Plataforma Google Games :: Servicios de localización WiFi mejorados
4.5 KIT KAT	Es la última versión y pretende corregir el principal problema para los usuarios, la gran cantidad de versiones y que los fabricantes no siempre ofrecen las actualizaciones. Reduce los requisitos de hardware, compatible con terminales de 512 MB de RAM.
5.0 LOLLIPOP	En marzo de 2015, la empresa anunció oficialmente la versión Android 5.1 Lollipop, trayendo algunas mejoras en el desempeño, nuevas animaciones y funciones, poco después de que la empresa habilitara Android 5.01 y Android 5.02 con algunas correcciones del sistema operativo.

6.0 Marshmallow	Android M, introdujo una característica importante: el usuario podía conceder o denegar permisos a las aplicaciones en función de sus necesidades. :: Se implementa el soporte para el USB Type-C y para huellas dactilares :: Esta nueva versión incorpora ya Android Pay.
7.0 Nougat	Se agiliza el proceso de instalación de las aplicaciones, ya que no se compilan durante la instalación, sino que este proceso se realiza una vez que se están ejecutando o cuando están en reposo. A esto se suma la implementación del compilador Just inTime (JIT), con creación de perfiles de código para ART, que reporta una mejora constante en el rendimiento de las apps a medida que se van ejecutando.
8.0 Oreo	Esta actualización, cuyo lanzamiento se produjo en agosto de 2017, marcó un antes y un después. Sobre todo, por lo relativo al Project Treble, que supuso el mayor cambio en los cimientos de Android desde sus inicios. Se implementó con este una arquitectura modular cuyo cometido era facilitar y agilizar a los fabricantes de hardware la entrega de actualizaciones de Android.
9.0 Pie	En agosto de 2018 se produjo el lanzamiento de esta novena versión de Android. Las principales características de esta actualización eran la importancia que adquiría la inteligencia artificial para el comportamiento predictivo del uso de aplicaciones, de la batería del dispositivo y algunas otras funcionalidades.
Android 10	Uno de los campos en los que Android se está volviendo más inteligente es en el de las respuestas que ofrece cuando trata de responder a todas y cada una de las comunicaciones, notificaciones o mensajes.
Android 11	En Junio de 2020 Google lanzó la primera beta de Android 11. Sus principales novedades fueron mejorar su compatibilidad con smartphones plegables, conectividad 5G, Project Mainline y HEIFs animados, además incluir soporte para la autenticación de llamadas STIR/SHAKEN. Otra de sus novedades es la nueva opción de permiso de ubicación, que permite otorgar acceso de ubicación (solo esa vez) a una aplicación.

Tabla 1 evolución android

BIBLIOGRAFÍA

[1] T. Krazit, «CNET,» [En línea]. Available: http://news.cnet.com/8301-1023_3-10245994-93.html. [Último acceso: 26 agosto 2013].

[2] G. Corp., «Google Empresa,» [En línea]. Available: http://www.google.com.ar/intl/es/about/company/. [Último acceso: 29 Agosto 2013].

[3] A. Chiclayo, «Android Chiclayo,» [En línea]. Available: http://android.cix.pe/lecciones/la-historia-de-android/. [Último acceso: 2013 Agosto 29].

[4] A. Hoog, Android Forensics Investigation, Analysis and Mobile Security for Google Android, Waltham: Elsevier, Inc., 2011.

[5] O. H. Alliance, «Open Handset Alliance,» [En línea]. Available: http://www.openhandsetalliance.com/. [Último acceso: 29 Agosto 2013].

[6] Android, «Android,» [En línea]. Available: http://www.android.com/. [Último acceso: 29 Agosto 2013].

[7] Fotostocker, «Foto Stocker,» [En línea]. Available: http://photographystocker.wordpress.com/2013/06/09/breve-historia-de-android-s-o-evolucion-en-el-mercado/. [Último acceso: 29 Agosto 2013].

[8] admin, «Android Zone,» [En línea]. Available: http://androidzone.org/2013/05/historia-de-android-la-evolucion-a-lo-largo-de-sus-versiones/. [Último acceso: 29 Agosto 2013].

[9] NTS solutions, «nts-solutions.com,» 3 Junio 2020. [En línea]. Available: https://www.nts-solutions.com/blog/versiones-android.php. [Último acceso: 24 Enero 2021].

[10] Condesa. [En línea]. Available: http://androideity.com/2011/07/04/arquitectura-de-android/. [Último acceso: 17 Marzo 2014].

[11] T_EVILL. [En línea]. Available: http://www.taringa.net/posts/linux/11592911/Android-y-Linux-la-delgada-linea-verde.html. [Último acceso: 17 Marzo 2014].

[12] A. F. Sanchez. [En línea]. Available: http://apkforandroid.es/android-y-linux-es-realmente-lo-mismo/. [Último acceso: 17 Marzo 2014].

[13] Á. J. Vico, 17 Marzo 2014. [En línea]. Available: http://columna80.wordpress.com/2011/02/17/arquitectura-de-android/.

[14] Ó. Mejia. [En línea]. Available: http://www.izt.uam.mx/newpage/contactos/revista/83/pdfs/android.pdf. [Último acceso: 17 Marzo 2014].

[15] U. C. I. MADRID, «Programación en dispositivos móviles portables,» [En línea]. Available: https://sites.google.com/site/swcuc3m/home/android/generalidades/2-2-arquitectura-de-android. [Último acceso: 25 Marzo 2014].

[16] U. O. d. Catalunya, 27 Marzo 2014. [En línea]. Available: http://cv.uoc.edu/~javiercg/practica_final/modulo5_1.2.3.html.

[17] C. Á. Caules, 27 Marzo 2014. [En línea]. Available: http://www.arquitecturajava.com/arquitectura-android-y-dalvik/.

[18] @condesa_sama, «Androity,» 22 Octubre 2014. [En línea]. Available: http://androideity.com/2011/07/07/la-maquina-virtual-dalvik/.

[19] D. Sánchez, 3 Abril 2014. [En línea]. Available: http://www.elandroidelibre.com/2013/11/art-la-nueva-maquina-virtual-de-google-que-sustituira-a-dalvik.html.

[20] Java, «developers,» 07 Mayo 2020. [En línea]. Available: https://developer.android.com/guide/platform?hl=es-419. [Último acceso: 21 Febrero 2021].

[21] J. Six, Application Security for the Android Platform, Primera ed., A. O. y. M. Hendrickson, Ed., O´Reilly Media Inc., 2012.

[22] J. Ardita, *Clase 11 Seguridad en UNIX / Linux,* 2013.

[23] www.adslzone.net. [En línea]. Available: http://i.imgur.com/LJ7C3.png. [Último acceso: 22 Abril 2014].

[24] C. Montalvo, *Fuente propia,* 2014.

[25] A. Inc., 12 Mayo 2014. [En línea]. Available: http://developer.android.com/guide/topics/manifest/permission-element.html.

[26] V. Sharma, «medium.com,» 29 Marzo 2019. [En línea]. Available: https://medium.com/@vardaansh1/an-introduction-to-android-interprocess-communication-and-common-pitfalls-ac4dfeddf89b. [Último acceso: 30 Marzo 2021].

[27] Android, «Adroid developers,» 27 Diciembre 2019. [En línea]. Available: https://developer.android.com/guide/components/aidl#kotlin.

[28] sepiablue, «Sepi Android,» 24 Octubre 2014. [En línea]. Available: http://sepiandroid.blog100.fc2.com/blog-entry-13.html.

[29] K. Islam, «Stack Overflow,» 22 Octubre 2014. [En línea]. Available: http://stackoverflow.com/questions/15904061/android-start-activity-without-creating-new-instance.

[30] M. Gargenta. [En línea]. Available: https://thenewcircle.com/s/post/1178/architecting_android_apps. [Último acceso: 30 Mayo 2014].

[31] M. Gargenta, «StackOverflow,» 23 Octubre 2014. [En línea]. Available: http://stackoverflow.com/questions/15904061/android-start-activity-without-creating-new-instance.

[32] Z. Ahmad, «dev.to,» 25 Junio 2018. [En línea]. Available: https://dev.to/xuhaibahmad/inter-process-communication-in-android-lessons-learnings-1p6n. [Último acceso: 30 Marzo 2021].

[33] P. Mirkelam, «https://proandroiddev.com/,» 2 Enero 2021. [En línea]. Available: https://proandroiddev.com/ipc-techniques-for-android-aidl-bb03ed62adaa. [Último acceso: 13 Abril 2021].

[34] Android, «Descripción general de los servicios,» 27 Diciembre 2019. [En línea]. Available: https://developer.android.com/guide/components/services?hl=es-419.

[35] C. C. y. D. T. Himanshu Dwivedii, Mobile Application Security, C. Chung, Ed., New York: McGraw Hill, 2010.

[36] @jcosmoscc, «Xataca,» [En línea]. Available: http://www.xatakandroid.com/mercado/ya-hay-mas-de-mil-millones-de-dispositivos-android-en-el-mundo. [Último acceso: 4 Octubre 2015].

[37] A. N. d. N. TELAM, «Telam Tecnología,» [En línea]. Available: http://www.telam.com.ar/notas/201410/83695-el-78--de-los-jovenes-en-america-latina-tiene-un-smartphone.html. [Último acceso: 11 Diciembre 2014].

[38] Cisco, «BYOD: una perspectiva global,» [En línea]. Available: https://www.cisco.com/web/about/ac79/docs/re/byod/BYOD_Horizons-Global_LAS.pdf. [Último acceso: 5 Octubre 2015].

[39] Kaspersky, «Centro de Prensa Kaspersky,» [En línea]. Available: http://latam.kaspersky.com/mx/sobre-kaspersky/centro-de-prensa/comunicados-de-prensa/60-de-empresas-en-am%C3%A9rica-latina-piensa-que-b. [Último acceso: 5 Octubre 2015].

[40] ISACA, «News Releases,» [En línea]. Available: http://www.isaca.org/About-ISACA/Press-room/News-Releases/2012/Pages/ISACA-Survey-Latin-American-Companies-Increasingly-Worried-About-BYOD-Risk-Spanish.aspx. [Último acceso: 5 Octubre 2015].

[41] H. Hernández, «Malavida.com,» [En línea]. Available: http://www.malavida.com/blog/47063/la-historia-de-android. [Último acceso: 11 Febrero 2014].

[42] DANYKUROSAK, 3 Abril 2014. [En línea]. Available: http://androtecnios.com/como-funciona-dalvik-vs-nuevo-art-maquinas-virtuales-android/.

[43] D. N. Arnao. [En línea]. Available: http://www.uv.es/~montanan/redes/trabajos/PKI.ppt. [Último acceso: 2014 Mayo 2014].

[44] Z. TIC. [En línea]. Available: http://zonatic.usatudni.es/aprendizaje/aprende-sobre-el-dnie/57-aspectos-tecnicos/196-criptografia-y-esquemas-de-clave-publica.html. [Último acceso: 1 Mayo 2014].

[45] wordreference.com, «Word Reference,» [En línea]. Available: http://www.wordreference.com/definicion/conciencia. [Último acceso: 21 Noviembre 2015].