SEGURIDAD EN DISPOSITIVOS MÓVILES ANDROID

OSCAR BETANCUR JARAMILLO

SONIA ELIZABETH ERASO HANRRYR

UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA - UNAD
ESCUELA DE CIENCIAS BÁSICAS, TECNOLOGÍA E INGENIERÍA
ESPECIALIZACIÓN EN SEGURIDAD INFORMÁTICA
2015

SEGURIDAD EN DISPOSITIVOS MÓVILESANDROID

OSCAR BETANCUR JARAMILLO

SONIA ELIZABETH ERASO HANRRYR

Asesor:

Ing. DEIVIS EDUARD RAMIREZ MARTINEZ

Monografía presentada para optar el título de:
ESP. EN SEGURIDAD INFORMÁTICA

UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA - UNAD
ESCUELA DE CIENCIAS BÁSICAS, TECNOLOGÍA E INGENIERÍA
ESPECIALIZACIÓN EN SEGURIDAD INFORMÁTICA
2015

NOTA DE ACEPTACIÓN

_____

_____

_____

Septiembre 26 de 2015

LISTA DE FIGURAS

pág.

LISTA DE TABLAS

pág.

INTRODUCCIÓN

La presente monografía tiene como fin analizar las debilidades que pueden presentarse en el sistema operativo Android para posteriormente presentar recomendaciones a los usuarios de estos dispositivos con el objetivo de prevenir ataques que puedan permitir el robo, manipulación o pérdida de la información que con estos dispositivos se pueda manejar.

Los dispositivos móviles hoy en día se han convertido en uno de los elementos más comunes entre las personas. Muchas de ellas tienen uno y hasta dos dispositivos consigo todo el tiempo. Las compras en internet, los pagos bancarios y muchos otros tipos de transacciones, se están realizando por estos medios. Es por esto que los amantes de lo ajeno, han visto una oportunidad muy valiosa al tratar de vulnerar estos dispositivos.

La seguridad en Android estuvo muy mal vista en sus inicios y fue uno de los sistemas menos seguro que existía. Sin embargo, Google con el pasar de los años, de las versiones y con la madurez de este sistema operativo, ha logrado convertir a Android en una de las plataformas móviles más usadas y seguras. Aún siguen existiendo fallas, como suele ocurrir con todo sistema, pero las actualizaciones suelen ir corrigiendo estos problemas.

En vista de que los dispositivos móviles se han convertido en un elemento importante en la vida diaria de cada persona, es muy importante protegerlos de la mejor forma posible y esto se puede lograr, siguiendo sugerencias de seguridad que se detallarán de manera específica en este documento.

ABSTRACT

This document pretends to analyze the weaknesses that may occur in the Android operating system and then submit recommendations to the users of these devices, in order to prevent attacks that may allow theft, manipulation or loss of information.

Mobile devices today have become, in one of the most common elements used by people. They have one or even two devices with them all the time. Internet shopping, bank payments and many other types of transactions are being made by these mechanisms. That is why some indelicate people have been a valuable opportunity to try to violate these devices.

The Android security was seen in a bad way when the operating system starts and was one of the least secure systems existed. However, Google with over the years, versions and with the maturity of this operating system, has made of Android one of the most used and secure mobile platforms. Security issues still exist, as often happens with every system, but updates are usually correcting these problems.

Since mobile devices have become an important factor in the daily life of each person, is very important to protect them in the best way possible and this can be achieved by following a few safety tips specifically described in this document.

1. INFORMACIÓN DEL PROYECTO

1.1TÍTULO

LA SEGURIDAD EN DISPOSITIVOS MÓVILES CON SISTEMA OPERATIVO ANDROID.

1.2 DEFINICIÓN DEL PROBLEMA

1.2.1 Descripción del problema. Se vive en un tiempo donde la tecnología ha aumentado de manera inimaginable, donde los dispositivos móviles tales como los PDAS, Smartphones, tabletas, computadores portátiles etc., se han convertido en una herramienta de uso necesario para las personas, ya que la gran cantidad de información que se maneja hace obligatorio utilizar estos dispositivos que agilizan muchas tareas, acortan la barrera de la comunicación, son portables, permiten la navegación por internet, juegos, acceso a correo electrónico, multimedia, creación de documentos, comunicaciones inalámbricas (wi-fi, bluetooth, gps), entre otras.

De acuerdo al Ministerio de las Tecnologías de la Información y las comunicaciones (TIC), entre enero de 2012 y 2013 Colombia fue el país con mayor crecimiento de Smartphones en el mundo y el acceso a la web desde dispositivos móviles ascendió a 19 millones de personas, la mayoría de ellos abonados con servicios 3G.

> "Según un comunicado de Google, en 2014 el 40 por ciento de las consultas realizadas en su buscador por usuarios colombianos se hicieron desde dispositivos móviles, lo cual muestra la evolución de esta industria en el país. "Sólo la aplicación de Mercado Libre Colombia, lanzada hace unos meses, genera el 35 por ciento de sus ventas por medio de Smartphones", aseguró Martin Gallone, gerente de marketing de Mercado Libre"[1].

Con las anteriores cifras se puede afirmar que existe un gran crecimiento en la obtención de estos dispositivos asociado de igual manera a las vulnerabilidades relacionadas con los accesos por elementos externos que pueden atacar la disponibilidad e integridad de la información, servicios y recursos que se encuentran en este tipo de dispositivos.

[1]Redacción Tecnosfera el Tiempo. Dispositivos móviles cambian la forma de hacer negocios en el país. [Citado: 17 de mayo de 2015] Disponible en Internet: http://www.eltiempo.com/tecnosfera/novedades-tecnologia/uso-de-celulares-transformaron-la-forma-de-hacer-negocios-en-colombia/15372015

Actividades como recibir correo o navegar en la web implican un riesgo para el dispositivo siendo más vulnerable la información y aplicativos que contiene. Los atacantes han empezado a dirigirse a este mercado por ser uno de los de mayor consumo ya que las computadoras personales han pasado a un elemento alternativo en los hogares y las personas se conectan a internet por medio de sus Smartphones o tablets. "Android es la plataforma que más sufre el malware con el 92% del total, debido en parte a las distintas versiones que circulan del sistema operativo. Según Google, el 3 de junio 2013, solo el cuatro por ciento de los usuarios de Android poseen la versión más reciente del sistema, que está inmune a las vulnerabilidades que explotan alrededor del 77 por ciento del malware para Android"[2].

Anteriormente las amenazas se propagaban usando bluetooth, sms, y muchas veces a través de técnicas de ingeniería social; en la actualidad los Smartphones u otros dispositivos como las tablets, son mucho más vulnerables de ser atacados, tal como lo informan compañías antivirus como KasperskyLab, donde han aumentado las amenazas contra aplicaciones que tienen los dispositivos móviles con sistema operativo Android, una de las causas principales es el desconocimiento del usuario para proteger su dispositivo, ya que muchas veces se enfocan a utilizar aplicativos que se encuentran instalados relacionados con correo electrónico, juegos, servicios de mensajería, redes sociales, multimedia, entre otros, pero son muy pocos los que toman conciencia sobre las medidas de seguridad para la protección de la información.

Los usuarios de dispositivos móviles pueden ser en cualquier momento víctimas de un ataque, la mayor parte de los virus troyanos llegan vía mensajes SMS; medio que representa el 48% de las agresiones, le siguen con 29% las aplicaciones falsas y el 19% se origina en malware espía.

> Los ciberdelincuentes son conocidos infractores de la privacidad de los dispositivos móviles crean aplicaciones maliciosas como las de robo de datos, cuyo objetivo es conseguir información personal y financiera. Las aplicaciones gratuitas de alto riesgo también plantean varios problemas de privacidad, ya que compilan gran cantidad de información de tipología diversa. Por ejemplo, algunas de las aplicaciones para Android más populares en Alemania pueden revelar la ubicación, la identidad del equipo y la libreta de direcciones del usuario.[3]

---

[2] BASUALDO, Alejandro. El malware para los dispositivos móviles ha crecido un 600 por ciento, Android es el mayor objetivo. [Citado: Mayo 22 de 2015] Disponible en Internet: http://googlelizados.com/malware-dispositivos-moviles-crecido-600-ciento-Android-mayor-objetivo.

[3] Cinco preguntas claves sobre la privacidad de los dispositivos móviles. Guía electrónica para la vida digital de trendLabs. [Citado: Mayo 10 de 2015] Disponible en: http://www.trendmicro.es/media/br/4ws-and-1h-of-mobile-privacyes.pdfhttp://www.trendmicro.es/media/br/4ws-and-1h-of-mobile-privacy-es.pdf

Las funciones de conectividad de un dispositivo móvil también facilitan el acceso de los ciberdelincuentes a la información. Las conexiones Bluetooth e inalámbrica, cuya intención es facilitar la comunicación, pueden utilizarse también con fines maliciosos.

Otro factor importante relacionado con ataques a dispositivos móviles está relacionado el robo de los mismos por parte de delincuencia común, más aún si se tiene en cuenta el lucrativo mercado que existe en cuanto a dispositivos robados y la información que contienen.

Diferentes factores como no comprobar los permisos de las aplicaciones, hacer clic en enlaces maliciosos e inclusive una mala configuración del equipo, son formas de invitar a los ciberdelincuentes a infiltrarse en un dispositivo. Cabe destacar que a medida que se vayan haciendo más y más populares estos dispositivos, las probabilidades de ataque y creación de malware aumentarán.

1.2.2 Formulación del problema. De acuerdo con lo expuesto anteriormente y examinando los riesgos a los que están expuestos este tipo de dispositivos se hace necesario conocer: ¿Qué nivel de seguridad presenta el sistema operativo Android en su versión 4.4 en cuanto a disponibilidad, integridad y confidencialidad?

1.3 ALCANCE O PROPÓSITO

La presente monografía abarca temas relacionados con los dispositivos móviles con sistema operativo Android versión 4.4, haciendo un comparativo con versiones anteriores, describiendo su arquitectura, mejora y vulnerabilidades de seguridad con el fin de tener una visión más clara y estar más documentados acerca del uso de los mismos y elaborar un plan enfocado a sensibilizar al usuario acerca del uso adecuado de la información que se maneja, las medidas preventivas tendientes a evitar cualquier ataque que pueda alterar integridad, disponibilidad y confidencialidad de la información que se maneje.

1.4 JUSTIFICACIÓN

Hoy en día los dispositivos móviles están al alcance del 90% de la población. Las facilidades de adquisición y las utilidades que presentan estos dispositivos han hecho que mucha población adquiera cada vez más este tipo de elementos, puesto que han revolucionado de manera representativa la manera de comunicarse, disminuyendo así las fronteras de la comunicación y aprovechando dicha tecnología para compartir ideas, mensajes, y todo tipo de información que se quiera intercambiar.

Se ha acrecentado los usuarios de telefonía móvil quienes aprovechan al máximo los servicios que se prestan, leyendo el correo electrónico, descargando

aplicaciones o usando servicios de geolocalización, entre otras. Pero a medida que avanza la tecnología igualmente surgen nuevos ataques a este tipo de dispositivos. En la actualidad los dispositivos móviles son más vulnerables para ser atacados por creadores de malware, ya que son en sí pequeños computadores, pero aún la gente no tienen conciencia de que sus equipos pueden ser atacados y son muy pocos los fabricantes que han generado software antivirus para móviles, además el número de plataformas existentes para móviles es demasiado diverso (Android de Google, Windows Mobile, Symbian de Nokia, BlackBerry, iPhone IOS, etc.), haciendo una gran variedad dispuesta para que cualquier atacante pueda alterar estos sistemas. El uso de estas tecnologías sitúa a los dispositivos móviles como uno de los productos potencialmente deseados por los atacantes para efectuar las ciberamenazas. Las personas que utilizan este tipo de dispositivos no son conscientes de que si no se toman las medidas necesarias para proteger su información tarde o temprano serán blanco de uno de los tantos ataques que diariamente se producen con el objetivo de dañar, alterar o robar información.

1.5 OBJETIVOS

1.5.1 Objetivo general. Implementar políticas de seguridad de la información en dispositivos móviles con sistema operativo Android 4.4.

1.5.2 Objetivos Específicos

- Realizar un estado del arte de la evolución histórica del sistema operativo Android, identificando las mejoras realizadas entre cada versión.

- Comprender el funcionamiento general del sistema operativo Android y determinar factores de riesgo existen en este sistema.

- Investigar las vulnerabilidades y mejoras de seguridad en el Sistema Operativo Android versión 4.4.

- Simular un ataque a un dispositivo con sistema operativo Android versión 4.4 para obtener información de cómo surgen estos tipos de ataques.

- Aplicar medidas de prevención contra ataques para los usuarios de dispositivos móviles con sistema operativo Android 4.4.

2. MARCO DE REFERENCIA

2.1 ANTECEDENTES

2.2.1 Historia de los dispositivos móviles. Desde hace algunos años ha aparecido en los terminales la categoría de Smartphone: teléfonos de funcionalidad similar a la de computadores personales, lo que implica una importante capacidad de cálculo y memoria

En los años de la Segunda Guerra Mundial, la compañía Motorola lanzó el Handie Talkie H12-16, el cual permitía comunicarse a distancia entre las tropas, era un dispositivo que se basaba en la transmisión mediante ondas de radio.

Esta tecnología se aprovechó entre los años 50 y 60 para crear diversos aparatos de radio y comunicación a distancia (Walkie-Talkies), que eran utilizados en mayor parte por taxis, ambulancias o bomberos.

Estos dispositivos no se pueden considerar como teléfonos móviles pero su implementación supuso el comienzo de la evolución de los dispositivos que se conocen en la actualidad.

- **1-G Primera Generación:** A partir de 1973 surgieron los móviles de primera generación los cuales eran de gran tamaño y peso. Funcionaban de manera analógica, es decir la transmisión y recepción de datos se apoyaba sobre un conjunto de ondas de radio que cambiaban de modo continuo. La desventaja de que fueran análogos era que solo podían ser usados para la transmisión de voz, tenían muy baja seguridad y esto implicaba que una persona pudiera escuchar llamadas ajenas con un sintonizador de radio o incluso hacer uso de las frecuencias cargando el importe de las llamadas a otras personas.

- **2-G Segunda Generación:** Esta generación marca el paso de la telefonía analógica a la digital lo cual mejoró el manejo de llamadas, se pudieron hacer más enlaces al mismo tiempo en el mismo ancho de banda e integrar otros servicios adicionales aparte de la voz, como el servicio de mensajes cortos (Short MessageService).Los estándares más utilizados en esta generación fueron: GSM, CDMA, GPRS.

- **3-G Tercera Generación:** En el año 2001 fue revolucionando la telefonía móvil con la aparición de los primeros celulares que tenían pantalla LCD a color. También nacieron dispositivos con cámara fotográfica digital, grabación de vídeos los cuales se podían enviar por mensajería instantánea, juegos en 3d, sonido mp3, conversaciones por videoconferencia gracias a una tasa de

transferencia de datos más aceptable y a un soporte para internet correctamente implementado (correo electrónico, descargas, etc.).

Los estándares más utilizados en esta generación fueron: UMTS, 4-G Cuarta generación**.** En el año 2010 se lanzaron los primeros servicios 4G basados en la tecnología LTE en Tokio, Nagoya y Osaka, la red 4G está basada en el protocolo IP. Esta tecnología puede ser utilizada por módems inalámbricos, celulares inteligentes y otros dispositivos móviles. La principal característica de la red de esta generación es que tiene la capacidad de proveer velocidades de acceso mayores a los 100 Mbps en movimiento y 1 Gbps en reposo manteniendo una calidad de servicio (QoS) de alta seguridad que permitirá ofrecer servicios de cualquier clase en cualquier momento, en cualquier lugar.

2.2.2 Estado del arte de la evolución del Sistema operativo Android. En 2003, nace la empresa Android Inc., donde su principal actividad se centraba en el desarrollo de software para teléfonos móviles, fundada por Andy Rubin, Rich Miner, Nick Sears y Chris White.

> En el 2005 Google comenzó a reclutar a algunas startup o compañías con futuro prometedor del sector móvil, con el objetivo de replicar su éxito de la web en el futuro de las telecomunicaciones inalámbricas. En Agosto de 2005 le tocó el turno a Android Inc., la fecha clave para llegar a entender mejor el éxito de Android es el 5 de noviembre de 2007. Ese día se fundaba la OHA (Open Handset Alliance), una alianza comercial de 35 componentes iniciales liderada por Google, que contaba con fabricantes de terminales móviles, operadores de telecomunicaciones, fabricantes de chips y desarrolladores de software. El mismo día se dio a conocer por vez primera lo que hoy conocemos como Android, una plataforma de código abierto para móviles que se presentaba con la garantía de estar basada en el sistema operativo Linux. [4]

Ha sido Google quien ha publicado la mayor parte del código fuente del sistema operativo, gracias a Apache que da soporte a proyectos software de código abierto.

Aunque no fue hasta un año después, en Octubre de 2008 que se vio funcionando por primera vez en un HTC Dream. Veía la luz en los Estados Unidos un móvil con la primera versión final de Android 1.0. El modelo G1 de HTC quedará para la historia como el iniciador de este gigante llamado Android.

[4] Historia de la informática. Qué es Android. Disponible en: http://histinf.blogs.upv.es/2012/12/14/android/. Consultado: Mayo 23 de 2015.

Figura1. Móvil HTC G1

Fuente: Características técnicas de HTC G1. Disponible en: http://www.smart-gsm.com/moviles/htc-g1

El HTC G1 es un móvil deslizable hacia el costado con teclado QWERTY y una gran pantalla sensible al tacto. Posee una cámara de 3 megapixeles, ranura microSD, navegador de Internet y Email.

Muchas características han destacado a Android, como la integración de servicios de Google, navegador Web compatible con HTML y XHTML que ofrece la función de zoom integrado y permite ver múltiples páginas en diferentes ventanas.

Hasta 2005 Android era un sistema operativo desconocido, cuando google lo compró. En noviembre de 2007 se lanzó laOpen Handset Alliance, que agrupaba a muchos fabricantes de teléfonos móviles, chipsets y Google y se proporcionó la primera versión de Android, junto con el SDK para que los programadores empezaran a crear sus aplicaciones para este sistema. En el 2011 se lanzó la versión 3.0 de este sistema denominada Honeycomb que se utiliza más para tabletas que para dispositivos móviles. En 2010 Android se ubicó como el sistema operativo de móviles más vendido en el mundo.

Su creador Andy Rubyn, ingeniero en ciencias de la computación de la Universidad Utica de Nueva York quien llevaba trabajando como ingeniero de telecomunicaciones en el área de los dispositivos móviles, de donde nació Android Inc., que fue adquirida por google en 2005. Rubyn pasa a ser empleado de esta

empresa y forma parte del equipo técnico acabando como Vicepresidente de Ingeniería de Google supervisando el proyecto de Android.

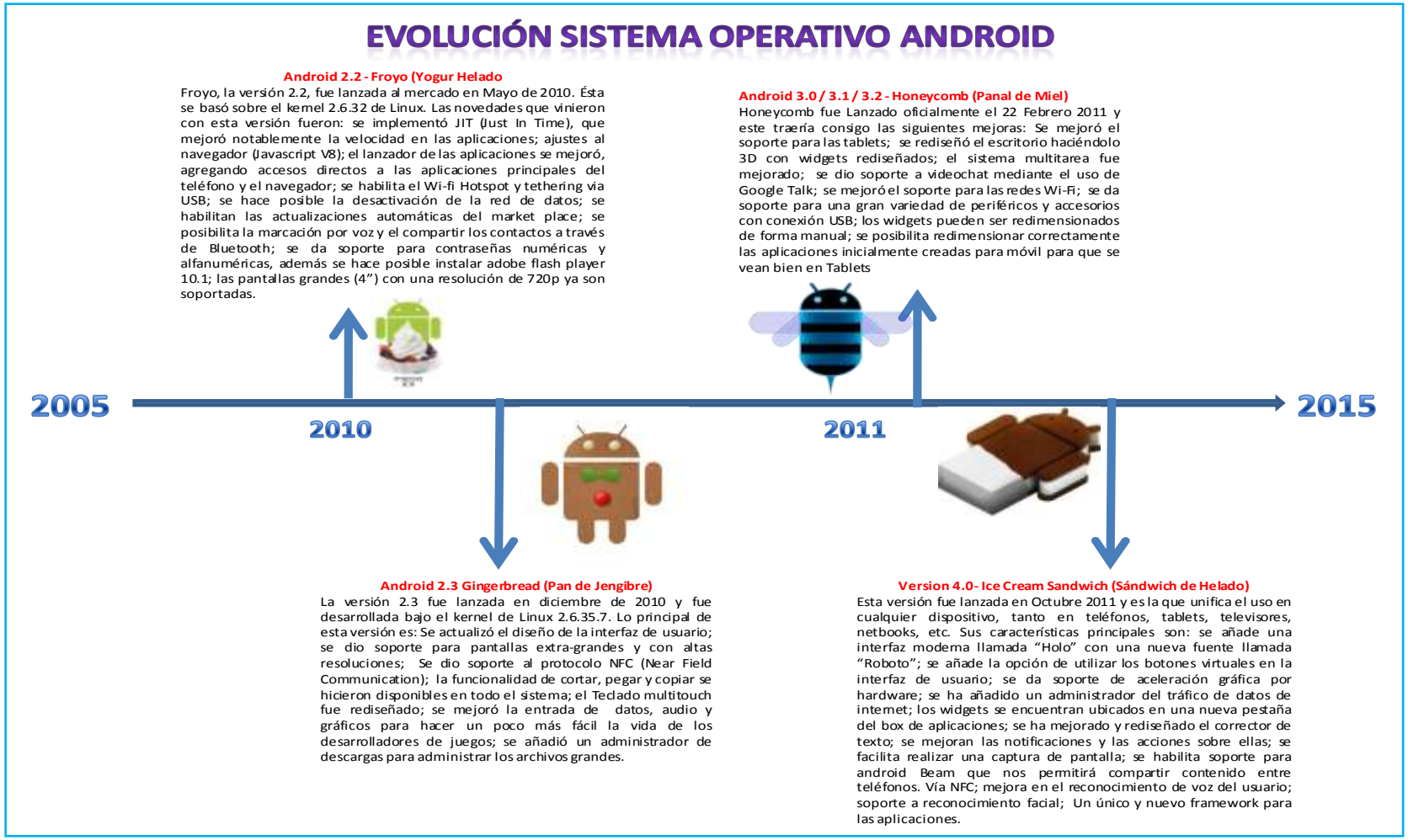
En cuanto a las versiones del sistema Operativo, cada versión desarrollada fue nombrada bajo un nombre específico de postres. Cada una de ella fue mejorando las funcionalidades permitidas y fue arreglando pequeños errores que se iban presentando en las anteriores versiones.

A continuación se relaciona mediante una línea de tiempo las diferentes versiones del sistema operativo Android con sus características principales.

Figura2. Evolución de las versiones del Sistema Operativo Android 2008-2009.

Fuente: Historia de Android. http://androidos.readthedocs.org/en/latest/data/historia/.

Figura3. Evolución de las versiones del Sistema Operativo Android 2009 – 2011.

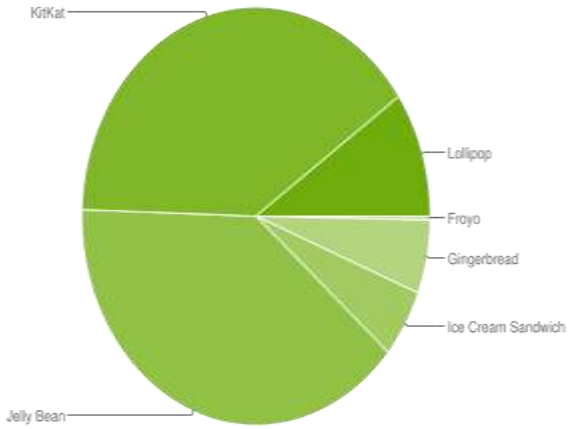


Fuente: Historia de Android. http://androidos.readthedocs.org/en/latest/data/historia/.

Figura4. Evolución de las versiones del Sistema Operativo Android 2012 - 2015.

Fuente: Historia de Android. http://androidos.readthedocs.org/en/latest/data/historia/.

Figura5. Distribución actual en versiones.



Version	Codename	API	Distribution
2.2	Froyo	8	0.3%
2.3.3 - 2.3.7	Gingerbread	10	5.7%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	5.3%
4.1.x	Jelly Bean	16	15.6%
4.2.x		17	18.1%
4.3		18	5.5%
4.4	KitKat	19	39.8%
5.0	Lollipop	21	9.0%
5.1		22	0.7%

Fuente: Android 4.4, sólo el 9,7% de los dispositivos Android están actualizados a Lollipop. Disponible en: [Último acceso: Mayo 2015]. Disponible en: http://www.xatakandroid.com/mercado/solo-el-9-7-de-los-dispositivos-android-estan-actualizados-a-lollipop.

2.3 MARCO LEGAL

La ley 1273 de 2009 creó nuevos tipos penales relacionados con delitos informáticos y la protección de la información y de los datos con penas de prisión de hasta 120 meses y multas de hasta 1500 salarios mínimos legales mensuales vigentes.

El 5 de enero de 2009, el congreso de la república de Colombia promulgó la ley 1273 “por medio del cual se modifica el código penal, se crea un nuevo bien jurídico tutelado – denominado “de la protección de la información y de los datos”- y se preservan integralmente los sistemas que utilicen las tecnologías de la información y las comunicaciones, entre otras disposiciones”

Ley estatutaria 1266 del 31 de diciembre de 2008

Artículo 4°. Principios de la administración de datos. En el desarrollo, interpretación y aplicación de la presente ley, se tendrán en cuenta, de manera armónica e integral, los principios que a continuación se establecen:

a) Principio de confidencialidad. Todas las personas naturales o jurídicas que intervengan en la administración de datos personales que no tengan la naturaleza de públicos están obligadas en todo tiempo a garantizar la reserva de la información, inclusive después de finalizada su relación con alguna de las labores que comprende la administración de datos, pudiendo solo realizar suministro o

comunicación de datos cuando ello corresponda al desarrollo de las actividades autorizadas en la presente ley y en los términos de la misma.

Artículo 18. Sanciones. Multas de carácter personal e institucional hasta por el equivalente a mil quinientos (1.500) salarios mínimos mensuales legales vigentes al momento de la imposición de la sanción, por violación a la presente ley, normas que la reglamenten, así como por la inobservancia de las órdenes e instrucciones impartidas por dicha Superintendencia. Las multas aquí previstas podrán ser sucesivas mientras subsista el incumplimiento que las originó.

2.4 MARCO TEÓRICO

2.4.1 Plataforma Android. Android es una plataforma de código abierto, con gran ventaja sobre los demás sistemas operativos para móviles como Nokia (Symbian), Apple (iOS) o RIM (Blackberry), ya que fabricantes, operadores y desarrolladores pueden dar mayor utilidad al Smartphone o tableta. Además de ser un sistema gratuito y multiplataforma, ha permitido instalarse de manera prácticamente fácil en dispositivos móviles aun con gamas bajas.

> "Android fue desarrollado y mantenido por Google para smartphones y tabletas que ofrece un desarrollo de software libre y tiene herramientas, aplicaciones y emuladores para desarrollar aplicaciones en Java. La plataforma Android aumentó su nivel de ventas del 3.5% en el 2009 a 25.5% en el 2010".[5]

El lanzamiento de Android como plataforma para el desarrollo de aplicaciones móviles ha tenido gran aceptación entre sus usuarios, de igual forma las industrias que lo distribuyen, convirtiéndose en una plataforma estándar frente a otras como iPhone, Windows Phone, Symbian, Blackberry, etc.

> "Android es un software pensado para dispositivos móviles que incluye el sistema operativo como middleware y diversas aplicaciones de usuario. Todas las aplicaciones para Android se programan en lenguaje java y son ejecutables en una máquina virtual diseñada para esta plataforma, llamada Dalvik".[6]

Las versiones anteriores de Android 4.0 se basan en Linux Kernel 2.6 mientras que las versiones más recientes se basan en 3.x. La siguiente figura se relaciona las versiones de Android con el kernel de Linux:

[5] Gartner. (2010). Gartner Press Releases. Disponible en Internet: http://www.gartner.com/it/ page.jsp?id=1466313

[6] SANTANA OROZCO, Alejandro. Una infraestructura de comunicaciones cliente-servidor para dispositivos móviles. México, 2011. Tesis Maestro en ciencias de la ingeniería de cómputo. Instituto Politécnico Nacional, 2011, 127 p.

Figura6. Linux Kernel en versiones de Android

Android 1.5	Linux Kernel 2.6.27
Android 1.6	Linux Kernel 2.6.29
Android 2.0/2.1	Linux Kernel 2.6.29
Android 2.2	Linux Kernel 2.6.32
Android 2.3.*	Linux Kernel 2.6.35
Android 3.*	Linux Kernel 2.6.36
Android 4.*	Linux Kernel 3.0.1
Android 4.1/4.2	Linux Kernel 3.0.31

Fuente: Joshua J. Drake, Pau Oliva Fora, Zach Lanier, Et al. Android Hacker's Handbook. Estados Unidos de América: John Wiley & Sons, Inc.

La licencia de distribución lo convierte en un software libre, se trabaja sobre una plataforma gratuita un SDK y la opción de plu-gin para el entorno de desarrollo llamado Eclipse, así como un emulador para su ejecución.

El proyecto de Android está dirigido por Google y otras empresas tecnológicas agrupadas bajo el nombre de Open Hanset Alliance (OHA) dentro de las cuales se encuentran Samsung, LG, Telefónica, Intel, Texas Instruments, etc., cuyo objetivo es desarrollar estándares abiertos para telefonía móvil para incentivar su desarrollo y para mejorar la experiencia del usuario.

Android cuenta con su propia máquina virtual DALVIK VIRTUAL MACHINE (DVM) que ejecuta código escrito en java, permite representación de gráficos 2D y 3D, soporta diferentes formatos multimedia, posibilita el uso de bases de datos, servicio de geolocalización, controla diferentes elementos de hardware como bluetooth, cámara, wifi, GPS, etc. Un aspecto básico de anotar es que a partir de la versión 4.4 existe otro entorno de ejecución llamado ART (Tiempo de Ejecución de Android) y el usuario es libre de cambiar entre DVM y ART, aunque para utilizar el ART se debe ser desarrollador y contar con un equipo Nexus.

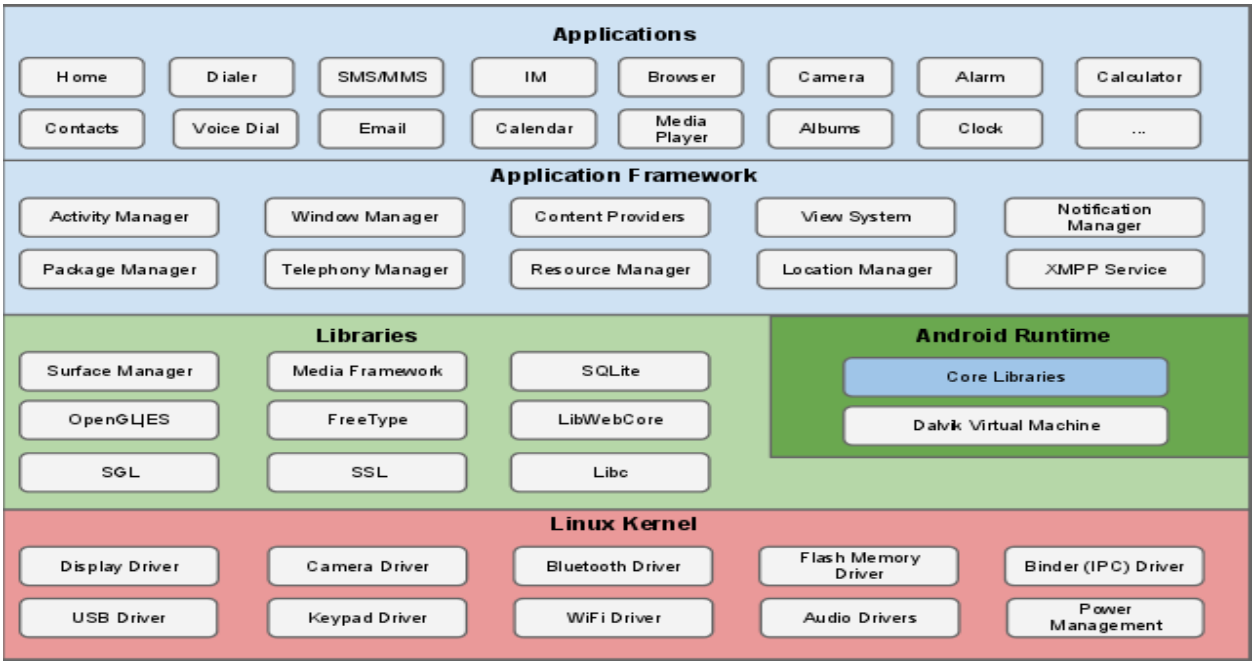
2.4.2 Estructura del sistema operativo Android. Este tema explicará las características principales del Sistema Operativo Android, su estructura, componentes e interacción entre los mismos.

Arquitectura. Es un sistema estratificado que conforma en sí una jerarquía, sus componentes son divididos en estratos, en donde los niveles más bajos agrupan componentes relacionados con la interacción del hardware del dispositivo, los estratos superiores corresponden a procesos de más alto nivel.

De forma general los componentes de un estrato utilizan los servicios provistos por el estrato inferior (si existe) y ofrecen sus servicios a los de estrato superior.

En la figura 7 se muestra la arquitectura general de cómo funciona el Sistema Operativo Android.La arquitectura se basa en capas, cada una de las capas utiliza servicios utilizados por la capa anterior y a la vez esta ofrece servicios a la capa superior.

Figura7. Descripción de la arquitectura de Android



Fuente: Android Open Source Project (Android Open Source Project: Android Security Overview. http:// source.Android.com/devices/tech/security/index.html. Ultimo acceso: Abril 2015.

1- Primera capa: Linux Kernel. El núcleo de Android está conformado por el sistema operativo Linux versión 2.6 ó 3.0, según la versión (Ver figura 7). Las funcionalidades que provee este estrato dependen de un kernel Linux multiusuario que se encarga, entre otras cosas, de la administración de la memoria, los procesos y los drivers de los distintos recursos como el hardware. En dicho kernel también se implementan aspectos básicos de seguridad.

Como es el primer estrato en la arquitectura, y la base de la estructura las características básicas del primero, dependen en gran medida del segundo nivel, puesto que si un fabricante incluye un nuevo elemento de hardware, lo primero

que se debe realizar para que pueda ser utilizado desde Android es crear las librerías de control o drivers necesarios dentro del kernel de Linux embebido en el propio Android.

2- Segunda capa – Librerías y tiempo de ejecución. Este estrato agrupa, básicamente, tres tipos de componentes:

> **Librerías nativas**. Se ofrece un conjunto de librerías C/C++ que proveen algunos servicios básicos para aplicaciones y otros programas. Dichos servicios son utilizados por los componentes del estrato superior. Entre sus funcionalidades se incluye facilitar el acceso al hardware (por ejemplo, manejo de gráficos) y a bases de datos. Estas librerías nativas corren en distintos procesos del kernel Linux subyacente[7].

Entre las librerías nativas[8] se encuentran:

- System C library: una derivación de la librería BSD de C estándar (libe), adaptada para dispositivos embebidos basados en Linux

- Media Framework: librería basada en PacketVideo's Apencare: soporta codecs de reproducción y grabación de multitud de formatos de audio, vídeo e imágenes MPEG4, H.264, MP3, AAC, AMR, JPG y PNG.

- Surface Manager: maneja el acceso al subsistema de representación gráfica en 2D y 3D.

- WebKit: soporta un moderno navegador web utilizado en el navegador Android y en la vista webview. Se trata de la misma librería que utiliza Google Chrome y Safari de Apple

- SGL: motor de gráficos 2D.

- Librerías 3D: implementación basada en OpenGL ES 1.0 API. Las librerías utilizan el acelerador hardware 3D si está disponible, o el software altamente optimizado de proyección 3D.

- FreeType: fuentes en bitmap y renderizado vectorial.

- SQLite: Potente y ligero motor de bases de datos relaciónales disponible para todas las aplicaciones.

[7] ROMANO, Carlos Luna. Descripción y análisis del modelo de seguridad Android. Reporte Técnico R-T 30-08.

[8] TOMAS GIRONES, Jesús. El Gran Libro de Android. Alfaomega Grupo Editor, S.A. de C.V. México: 2012. p. 28

- SSL: Proporciona servicios de encriptación Secure Socket Layer.

Máquina virtual. "Android cuenta con una máquina virtual, llamada Dalvik, para la ejecución de las aplicaciones programadas en Java. Esta ejecuta archivos con formato Dalvik Executable (.dex) que está optimizado para reducir al mínimo el consumo de memoria del dispositivo móvil" [9]

Cada aplicación Java es compilada a un formato **bytecode** y, posteriormente, es ejecutada en una máquina virtual Dalvik propia, distinta a la asignada a cualquier otra aplicación. Además de utilizar el lenguaje de programación Java en el desarrollo de una aplicación, también es posible incluir código nativo (como C o C++) que corra por fuera de la máquina virtual Dalvik. Este tipo de código, al compilarse, se ejecuta directamente en el procesador del dispositivo móvil. Sin embargo, la ejecución de dicho código nativo continúa siendo afectada por las restricciones del kernel Linux.

Librerías Estándar. Se provee, a nivel de ejecución de aplicaciones, la mayoría de las funcionalidades disponibles en las librerías estándar de Java, como por ejemplo, operaciones matemáticas, de texto, de entrada/salida, entre otras.[10]

3- Tercera capa - Framework de Aplicaciones. En este estrato se encuentra el framework que se encarga de, entre otras cosas, administrar el ciclo de vida de cada aplicación, proveer el conjunto de APIs necesarias para el desarrollo y manejar la interacción tanto entre aplicaciones como también entre las distintas partes que las componen.

Uno de los objetivos principales de la arquitectura de Android es la reutilización de componentes: cualquier aplicación puede ofrecer sus servicios y cualquier otra, con la autorización correspondiente, puede usarlos, es por ello que Android permite una comunicación entre diferentes componentes de aplicaciones.

"Otra característica del framework, es que un desarrollador puede tener acceso a las mismas APIs que utilizan las aplicaciones principales"[11] , es decir aquellas aplicaciones que ya vienen preinstaladas en el dispositivo. Todas las librerías del framework están escritas en Java y se encuentran en la máquina virtual **Dalvik** de cada aplicación. Si la aplicación requiere llevar a cabo alguna acción, las librerías se comunican con el sistema Linux base donde se verifica los permisos de acceso a los recursos del sistema de la aplicación solicitante.

[9] Mobarhan, Masoumeh Al. Haghighi: *Formal Specification of Selected Android Core Applications and Library Functions*. Suecia, 2011. Tesis de Licenciatura, Chalmers University of Technology, University of Gothenburg, Gotemburgo. Disponible en: http://publications.lib.chalmers.se/records/fulltext/136115 . p. 3
[10] Ibid. p. 4
[11] Ibid. p. 5

4- Cuarta capa – Aplicaciones. En este nivel se incluyen tanto las aplicaciones principales(por ejemplo, cliente de e-mail, calendario, libreta de contactos) como las nuevas aplicaciones escritas por otros desarrolladores.

Ambos tipos de aplicaciones se ejecutan en el marco impuesto por el framework del estrato inferior. Sin embargo, el acceso a las APIs está restringido a los fragmentos de aplicaciones escritos en el lenguaje de programación Java. De existir código nativo, no se podrá acceder directamente a las APIs a través a del mismo sin antes interponer código Java que actúe como intermediario[12]

Componentes de una aplicación Android. Una aplicación de Android se construye a partir de diferentes bloques llamados componentes. Cada componente es una entidad única y cumple una función específica, definiendo el comportamiento general de la aplicación. Un aspecto del diseño del sistema Android es que una aplicación puede iniciar un componente de otra si se tienen los permisos respectivos.

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

Existen cuatro tipos de componentes de una aplicación. Cada tipo se utiliza para un propósito diferente y tiene un ciclo de vida distinto"[13]

Estos tipos son:

1. Actividades: Una actividad representa una pantalla de la aplicación, en donde se provee una interfaz de usuario para interactuar con la misma. Típicamente, cada aplicación tiene una actividad principal que representa la primera pantalla que ve el usuario al ser iniciada desde la lista de aplicaciones disponibles. A partir de ese momento, es posible pasar a la siguiente pantalla (de existir) llamando a una nueva actividad

Cada actividad tiene asociada una vista. La vista constituye el conjunto de elementos ò interfaz gráfica que se presentan al usuario para que interaccione con el sistema. Por ejemplo, cajas de edición, etiquetas o botones, entre otros. Para crear actividades personalizadas es necesario definir una clase que herede de la clase base Activity.

---

[12] Felt, Adrienne Porter, Erika Chin, Steve Hanna, Dawn Song y David Wagner: Android permissions demysti_ed. En: Proceedings of the 18th ACM conference on Computer and communications security, CCS 11, pág. 627 - 638, New York, NY, USA, 2011. ACM, ISBN 978-1-4503-0948-6. http://doi.acm.org/10.1145/2046707.2046779.

[13] Android Developers: Application Fundamentals.[Citado: Febrero 28 de 2015]. Disponible en:http://developer.Android.com/guide/components/fundamentals.html.

A pesar de que una aplicación tenga una actividad principal, cada una de éstas es independiente de las otras, dando la posibilidad a una aplicación distinta de iniciar cualquier actividad que no sea, necesariamente, la principal (siempre y cuando se tenga la autorización apropiada), por ejemplo una aplicación de correo electrónico.

2. Servicios: "Un servicio es un componente cuya ejecución se desarrolla en segundo plano sin ofrecer ninguna interfaz con el usuario. Cualquier componente con los permisos adecuados puede iniciar un servicio o asociarse a uno en ejecución para interactuar con él".[14]

Si un componente inicia un servicio que ya se encuentra en ejecución, no se crea una nueva instancia sino que se interactúa con el ya existente.

El uso más frecuente de los servicios es para realizar tareas que demanden una gran cantidad de tiempo (y no requieran interacción con el usuario); sin embargo, también pueden ser utilizados con el fin de trabajar para procesos remotos. Por ejemplo, un servicio podrá reproducir música o descargar un archivo a través de internet sin impedir que el usuario siga utilizando su teléfono móvil normalmente

Los servicios pueden tomar dos formas esencialmente:

Started: Una vez iniciado el servicio, se ejecuta de forma indefinida en segundo plano, incluso si se destruye el componente que lo inició.

Bound: En este caso, el servicio ofrece una interfaz de tipo cliente-servidor a aquellos componentes que se asocian o enlazan (bind) a dicho servicio. Los componentes asociados o enlazados (bound) al servicio pueden enviar peticiones, conseguir resultados y respuestas o, incluso, llevar a cabo tareas de comunicación con otros procesos. En este caso, el servicio sólo se ejecuta cuando hay algún componente asociado o enlazado (bound) a él.

3. Content Providers: Un content provider es un componente diseñado para compartir información entre aplicaciones. Dicha información puede estar guardada, por ejemplo, en bases de datos SQLite, en la web o en cualquier otro medio de almacenamiento persistente que esté disponible"[15]. De esta forma, un content provider actúa como una interfaz entre los datos persistidos y el resto de las aplicaciones para que, estas últimas, puedan tanto acceder a los mismos como también modificarlos.

---

[14] Android Developers: Application Fundamentals. [Citado: Mayo 23 de 2015]. Disponible en: http://developer.Android.com/guide/components/fundamentals.html. Ultimo acceso: Abril de 2015.

[15]Mobarhan, Masoumeh Al. Haghighi: *Formal Specification of Selected Android Core Applications and Library Functions*. Suecia, 2011.Tesis de Licenciatura, Chalmers University of Technology, University of Gothenburg, Gotemburgo, http://publications.lib.chalmers.se/records/fulltext/136115. p. 3

Existen dos formatos posibles en los que un content provider puede presentar la información almacenada internamente al resto de las aplicaciones. Estos son: archivos y tablas. Por ejemplo, si se desea compartir el contenido de una base de datos SQLite, se presentaría la información en formato de tablas.

4. Broadcast Receivers: Un broadcast receiver es un componente cuyo objetivo es recibir mensajes, anuncios, emitidos por el sistema u otra aplicación, y disparar acciones a partir de los mismos. Dichos mensajes, llamados broadcasts, se transmiten a lo largo de todo el sistema y son los broadcast receivers los encargados de decidir cuáles de ellos se comunicarán a la aplicación a la que pertenecen.

Por ejemplo, el sistema podría generar broadcasts cuando la batería esté por agotarse o cuando se tome una foto. Dependiendo de la aplicación, se configuran o no a sus respectivos broadcast receivers para suscribirse o no a esta clase de mensajes, y si lo realiza realizar una acción determinada al recibirlos. Un ejemplo puede ser la descarga de un archivo.

2.4.3 Interacción entre componentes de Android. "Tres de los cuatro tipos de componentes, actividades, servicios y broadcast receivers, son activados mediante un mensaje asíncrono llamado intent. Estos mensajes hacen que distintos componentes individuales, pertenecientes tanto a una misma aplicación como a aplicaciones distintas, se relacionen entre sí en tiempo de ejecución. Existen, principalmente, dos formas de utilizar un intent: como un broadcast o como un mensaje para interactuar con actividades y servicios"[16].

Al crear un intent se pueden incluir los siguientes campos[17] :

- **Componente destino:** Especifica el nombre del componente al que va dirigido el intent. De no especificarse ningún destinatario, a la hora de enviar el intent, el sistema debe elegirlo basándose en la información contenida en otros campos.

- **Acción:** Describe la acción que se quiere realizar con el intent o, en el caso de ser utilizado como broadcast, el evento que sucedió. Algunas constantes predefinidas para este campo son: ACTION CALL, para iniciar una llamada telefónica desde una actividad; ACTION EDIT, para enviar información con el fin de ser editada en una actividad; o ACTION BATTERY LOW, para comunicar que la batería se está agotando.

- **Categoría:** Especifica, con una o varias constantes, las características del componente que debería recibir el intent. Por ejemplo, la contante predefinida

[16]Ibid. p. 6
[17]Ibid. p. 10

CATEGORY LAUNCHER indica que el componente destino debería ser la actividad principal de una aplicación

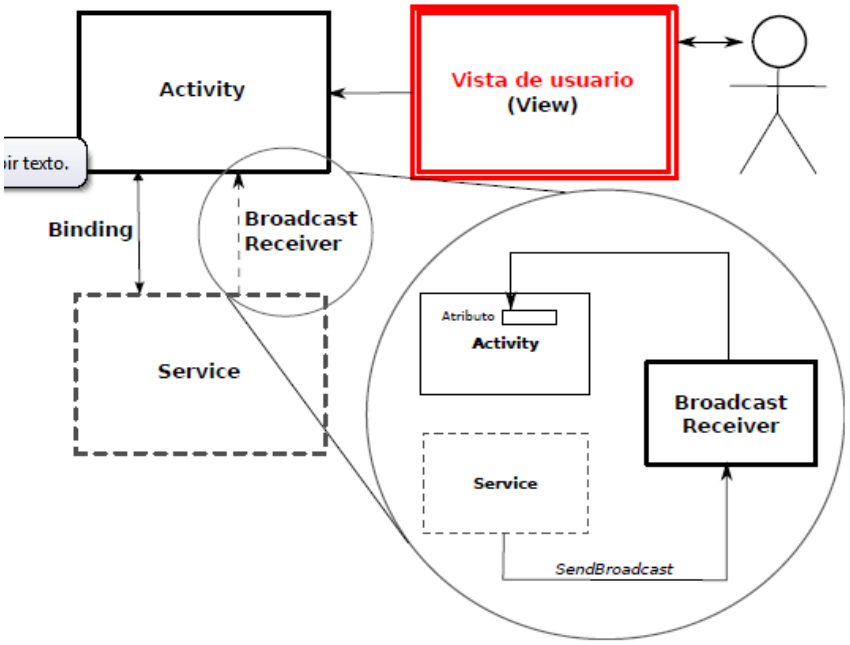
- **Datos:** Incluye el o los URIs que identifican los datos sobre los cuales se quiere actuar. Dichos URIs pueden ser referencias a recursos de algún content provider o a otro tipo de datos. Por ejemplo, si al campo acción se le asigna el valor ACTION CALL, se incluirá el URI del número de teléfono que se quiere llamar.

- **Banderas:** Incluye banderas opcionales para realizar tareas específicas como por ejemplo la delegación de permisos.

- **Extras:** Ofrece información adicional.

La siguiente figura muestra la interacción entre los diferentes componentes de una aplicación Android:

Figura 8. Interacciones entre diferentes componentes de una aplicación Android



Fuente: PAVON PULIDO, Nieves. Componentes básicos de Android. Cloud Incubator Hub, 2013.

2.4.4 Wireless Local Area Network (WLAN) en la Arquitectura Android

Figura9. Arquitectura Wlan en Android.

Fuente: Drake, Joshua J. Drake; Fora, Oliva Fora; Lanier, Zach Et al. Android Hackers HandBook.John Wiley & Sons, Inc. 577 p.

Android utiliza bibliotecas centrales que para ejecutar los servicios; las bibliotecas centrales también llamadas interfaces de programación de aplicaciones (API) se desarrollan utilizando el lenguaje de programación Java (SDK de Android). La API responsable de la gestión inalámbrica es la WifiManager y se ejecuta en su propio proceso llamado instancia de la DVM.

El SDK de Android ofrece un conjunto de aplicaciones para el acceso de datos sobre las redes Wi-Fi, tales como intensidad de la señal, descubrimiento de los puntos de acceso y la ejecución de procesos vinculados a puntos de acceso. Para permitir la conexión Wi -Fi gratuita, hay cierta información requerida que se logra mediante la creación de un permiso explícito mediante el archivo AndroidManifest.xml

a) CHANGE_WIFI_STATE - Se utiliza para autorizar establecer o desactivar el Wi-Fi
b) El ACCESS_WIFI_STATE - Se utiliza para solicitar cualquier información del servicio de Wi -Fi.

Transfer Control Protocol / Internet Protocol (TCP / IP). Consiste en:

1. Protocolo de Control de Transmisión (TCP), el intercambio de mensajes entre aplicaciones;

2. Protocolo de datagramas de usuario (UDP) de comunicación simple entre aplicaciones;

3. Protocolo de Internet (IP) el intercambio de mensajes entre computadoras;

4. Protocolo de mensajes de control de Internet (ICMP) para el envío de mensajes de error y las estadísticas de uso de los recursos;

5. Dynamic Host Configuration Protocol (DHCP) para direccionamiento dinámico de las tabletas o smartphones

2.4.5 Android 4.4 KitKat. Google anunció KitKat (versión 4.4 de Android) en septiembre de 2013 que funcionó inicialmente en un Smartphone Nexus 5, su nombre está basado en una barra de chocolate.

Android KitKat comenzó con la versión 4.4 pero se desprendieron varios cambios como Android 4.4.1, 4.4.2, 4.4.3 y más recientemente Android 4.4.4 con el objeto de realizar ajustes y cambios de mejora del sistema operativo.

Diseñado para que se ejecute en una gran gama de dispositivos que pueden funcionar con un mínimo de 512 Mb de RAM, caracterizado por ser rápido, liviano y dinámico, introduce nuevas API's para ayudar a los desarrolladores a crear aplicaciones que gasten menos batería y memoria RAM. Esta es una característica importante en esta versión, ya que las anteriores versiones del sistema operativo requerían más memoria interna lo cual presentaba incompatibilidad con modelos de dispositivos más antiguos, esta fue una de las principales causas del problema de compatibilidad conocida como fragmentación de Android.

La interfaz de usuario de Android 4.4 es más limpia, el NFC (Near Field Communication) utiliza botones de acción desde la pantalla táctil que sustituyen el uso de los botones físicos que se encuentran en muchos dispositivos Android. Esta versión también cuenta con un módulo de Linux con seguridad mejorada con lo cual se previene accesos no autorizados de aplicaciones.

Mejoras de la versión 4.4

- **Google Now:** Actualización de tarjetas e inclusión de nuevas, en esta versión escanea las apps instaladas en el dispositivo, para que el resultado de la búsqueda sea más efectivo y utilice las aplicaciones para dar alternativas de resultados.

- Creación de videos con fotos almacenadas en el dispositivo.

- La multitarea es mucho más rápida en Android KitKat 4.4. Se han simplificado las aplicaciones que trabajan en segundo plano con el fin de que consuman menos memoria, y así dar más potencia a otras tareas y que no haya ralentización.[18]

- Más soporte para html5, java script y css3.

- Se ha incluido un nuevo soporte para pagos con NFC, acceso a la tarjeta de crédito y programas de fidelización. Además tiene un nuevo modo de lectura, para proporcionar una manera más rápida en el procesamiento de pagos NFC y las transferencias.

Mejoras en la seguridad:

Las anteriores versiones de Android tenían elementos de seguridad que fueron mejorando a medida que iban surgiendo nuevas versiones.

Además de los beneficios comentados, Android 4.4 trae varias mejoras de seguridad que refuerzan la seguridad del sistema[19], entre ellas están:

- **SELinux D:** En Android 4.4, se pasa de correr SELinux en modo permisivo (que sólo hace un log de los fallos) a enforcing mode. El núcleo SELinux en Android 4.4 KitKat funciona en modo “enforced”, en lugar del modo “permisive”. Esto niega el acceso a ciertos recursos, ayudando a frustrar los ataques de escalada de privilegios, como los exploits que quieran tener acceso root. SELinux fue introducido en Android 4.3 como sistema de control de acceso obligatorio al núcleo de Linux para ayudar a reforzar los derechos de acceso, por ejemplo permisos, y para evitar ataques de escalado de privilegios.

- **Soporte de claves de cifrado de curva elíptica (ECDSA) en AndroidKeyStore:** El almacén integrado de contraseñas Android incluye soporte para claves de firma de Curva Elíptica. Aunque este tipo de cifrado ha recibido mala crítica, aún sin demostrar, ECC es una forma viable como alternativa a otros algoritmos como RSA. Es otra opción más para los desarrolladores aunque para un almacenamiento de datos a largo plazo la selección de cifrado simétrico sigue siendo la mejor opción.

- **Avisos de Cifrados sospechosos SSL CA:** Muchos entornos corporativos incluyen software de monitorización SSL que añaden Certificado de Autoría (CA) al computador y/o navegador como medio para evitar ataques de tipo man-in-the-middle en sesiones HTTPS. Esto ha sido posible gracias a que

[18] http://computerhoy.com/listas/moviles/20-razones-que-actualizar-Android-kitkat-8577
[19] http://www.xatakAndroid.com/sistema-operativo/las-mejoras-de-seguridad-de-Android-4-4

Android 4.4 integra una clave CA adicional. Por ende, Android 4.4 avisará a los usuarios si su dispositivo recibe algún certificado SSL CA sospechoso.

- **Detección automática de desbordamiento de pila:** Android 4.4 está compilado con la opción FORTIFY_SOURCE configurada en nivel 2, lo que asegura que todo el código C está compilado con esta protección.

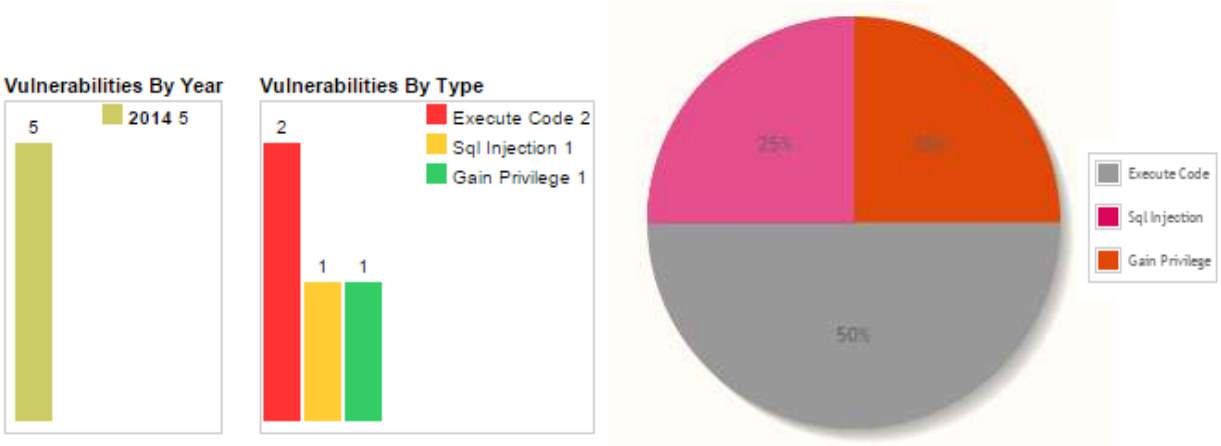
  FORTIFY_SOURCE es una característica del compilador que intenta identificar oportunidades de overflow que podrían ser explotadas por software malicioso intentando ganar acceso para una posible ejecución de código. Obviamente FORTIFY_SOURCE no elimina todas las posibilidades de que existan errores de tipo buffer overflow, pero sí que es mejor que nada.

- **Google Certificate Pinning:** Android 4.4 añade protección ante la sustitución de los certificados de google. Certificate Pinning implica que sólo se permite ejecutar certificados SSL concretos (lista blanca) frente a determinados dominios. Este sistema evita ataques de tipo man-in-the-middle a la hora de aceptar un certificado válido SSLCA, ya que sólo se pueden aceptar determinados certificados, haciendo más difícil este tipo de hackeo.

2.4.6 Vulnerabilidades del sistema operativo Android 4.4

De acuerdo con **CVE** (Common Vulnerabilities and Exposures) que es una lista oficial de vulnerabilidades de seguridad de la información públicamente conocidas, se han identificado las siguientes vulnerabilidades para Android 4.4:

Figura10. Vulnerabilidades de Android 4.4 KitKat



Fuente: http://www.cvedetails.com/version-list/1224/19997/1/Google-Android.html

Nombre: **CVE-2014-8610**
Tipo: Permisos, privilegios y/o control de acceso
Gravedad: Media

Descripción

En el archivo AndroidManifest.xml no se requiere el permiso Send_SMS para el receptor SmsReceiver, lo que permite a los atacantes enviar mensajes SMS almacenados, y por lo tanto transmitir nuevos SMS s o desencadenar cargos adicionales por mensaje de un operador de red para antiguos mensajes, a través de una aplicación diseñada que transmite la intención con la acción com.android.mms.transaction.MESSAGE_SENT, también conocido como Bug 17671795

Tipo de impacto: Afecta parcialmente a la integridad y confidencialidad del sistema, no hay impacto en la disponibilidad del sistema

Productos y versiones vulnerables:

Google Android 4.4.4
Google Android 4.4.3
Google Android 4.4.2
Google Android 4.4.1
Google Android 4.4

Nombre: **CVE-2014-8609**
Tipo: Permisos, privilegios y/o control de acceso
Gravedad: Crítica

Una vulnerabilidad clasificada como crítica fue encontrada en Google Android hasta 4.4.4. Una función desconocida del componente *Pendingintent* es afectada por esta vulnerabilidad. A través de la manipulación de un input desconocido se causa una vulnerabilidad de clase escalada de privilegios. Esto tiene repercusión sobre la confidencialidad, integridad y disponibilidad.

La vulnerabilidad fue publicada el 2014-11-26 por WangTao, WangYu y Zhang Donghui de Baidu X-Team con identificación *CVE-2014-8609 Android Settings application privilege leakage vulnerability / Android id 17356824* con un mailinglist post (Full-Disclosure) (corroboradas). La vulnerabilidad es identificada como CVE-2014-8609. Resulta fácil de explotar. El ataque se puede hacer desde la red. La explotación no necesita ninguna autentificación específica. No se conoce los detalles técnicos ni hay ningún exploit disponible

Nombre: **CVE-2014-8507**

Múltiples vulnerabilidades de inyección SQL en el método queryLastApp en los paquetes: /WAPPushManager/src/com/android/ smspush/WapPushManager.java en el módulo WAPPushManager que permite a atacantes remotos ejecutar comandos SQL de su elección, y por lo tanto lanzar una actividad o servicio , a través de (1) wapAppId o (2) campo contentType de una PDU para un mensaje WAPPush malformada.

Tipo de impacto: Permite que la divulgación no autorizada de la información; modificación no autorizada de información e interrupción del servicio

Nombre: **CVE-2014-7911**

Permite a los atacantes ejecutar código arbitrario a través de un método finalize elaborado por un objeto serializado en un ArrayMap dentro de una intención enviada al system_service, como lo demuestra el método finalize de android.os.BinderProxy.

Tipo de impacto: Permite que la divulgación no autorizada de la información; Permite la modificación no autorizada; Permite la interrupción del servicio

Nombre: **CVE-2013-6770**

El CyanogenMod/ClockWorkMod/Koush paquete superusuario 1.0.2.1 para Android 4.3 y 4.4 no restringe adecuadamente el conjunto de usuarios que pueden ejecutar /system/ xbin/su con la opción --daemon lo que permite a atacantes ganar privilegios mediante el aprovechamiento de ADB (Android Debug Britget, comando para ejecutar un emulador) y acceso al UID Linux para luego crear un script caballo de Troya.

En el 2013 se detectó una vulnerabilidad que afectaba a los dispositivos Android 4.4 llamada Android Master Key Vulnerability[20] que permite a los hackers modificar una aplicación legítima y firmada digitalmente con la finalidad de transformarla en un troyano para que pueda robar datos o tomar el control del dispositivo. De esta forma, el exploit de Saurik permite que el hacker obtenga acceso completo al dispositivo Android mediante la modificación de un APK del sistema sin que la firma se vea alterada. De este modo, un malware podría obtener acceso completo al dispositivo Android así como a las aplicaciones y datos del usuario.

[20]Android 4.4 KitKat afectada por Master Key: http://securityaffairs.co/wordpress/19400/hacking/android44-master-key-vulnerability.html

2.4.7 Dispositivos que vienen con el Sistema Operativo Android 4.4

- Google Nexus 5
- Google Nexus 10 (Versión 2013)
- Google Nexus Gem
- Google Glass

Para complementar esta información ver Anexo A del presente documento.

2.4.8 Seguridad en Android. La mayoría de las medidas de seguridad entre el sistema y las aplicaciones deriva de los estándares de Linux 2.6, cuyo kernel constituye el núcleo principal de Android. Ninguna aplicación tiene los permisos para realizar operaciones que puedan alterar el comportamiento del sistema, como leer o escribir archivos o ficheros privados, acceso a la red, habilitar hardware, no están per autoricen llevar a cabo una acción.

Según un reporte del Centro de Redes de Amenazas Juniper Mobile (MTC), se detectaron que los virus móviles crecieron un 614% en el último año. Existen 276.259 aplicaciones maliciosas para dispositivos móviles, indicó la consultora especializada internacional. Los usuarios de equipos con sistema operativo Android son los más perjudicados. Juniper detalla que el 92% de todas las amenazas van dirigidos a esta plataforma, puesto que tiene una participación dominante en el mercado mundial de los teléfonos inteligentes. Se detectaron más de 500 desarrolladores de aplicaciones para Android en todo el mundo, la mayoría proviene de China o Rusia[21].

Un elemento de seguridad de Android (Google, 2011c) es el archivo de manifiesto. El archivo de manifiesto está incluido en el paquete de instalación de Android (archivo .APK), junto con el código de bytes y otros recursos relacionados. El archivo sigue la estructura XML y proporciona toda la información necesaria a la plataforma Android para la ejecución de la aplicación. El archivo de manifiesto es crucial para el sistema, ya que es donde se definen los permisos de cada aplicación. Estos permisos funcionan en dos formas, la primera es como la aplicación interactúa con el sistema mediante el acceso a la API del sistema y en segundo lugar, la forma como el sistema y otras aplicaciones interactúan con la aplicación dada.

Por defecto, cada aplicación se ejecuta en un entorno de espacio aislado sin ningún tipo de permiso para realizar una acción que pueden afectar el sistema operativo en sí mismo.

[21] Descargar apps desconocidas pone en riesgo al Smartphone. Disponible en: http://m.ultimahora.com/descargar-apps-desconocidas-pone-riesgo-al-smartphone-n706187.html

Toda solicitud de permisos se realiza durante el momento de instalación, en donde el usuario aprueba o rechaza los permisos que solicita la aplicación. No hay más controles se realizan durante la ejecución de las aplicaciones, por lo tanto, si el usuario decide conceder permiso a la aplicación, a continuación, los recursos del sistema protegidos están disponibles para la aplicación, de lo contrario el acceso a los recursos es denegado. Es ahí donde muchas aplicaciones maliciosas aprovechan para instalarse en el sistema y acceder a la información del dispositivo o denegar algún servicio.

Los paquetes de instalación de Android deben ser firmados digitalmente por su creador, con una identificación única de cada aplicación. En el modelo de seguridad de Android no es necesario que el certificado del desarrollador esté firmado por un certificado de autoridad de confianza, por ende las aplicaciones son generalmente firmadas de manera digital con certificados auto-firmados, proporcionando se verifique el origen y la protección de la integridad.

Para establecer un permiso para una aplicación, es necesario declarar en el manifiesto uno o más elementos <uses-permission> donde se especifica el tipo de permiso que se desea habilitar.

En la clase Android.Manifest.permission se especifican todos los posibles permisos que se pueden conceder a una aplicación: utilización de wi-fi, bluetooth, llamadas telefónicas, cámara, internet, mensajes SMS y MMS, vibrador, etc.

2.4.9 Elementos de seguridad Android. En los dispositivos móviles la seguridad juega un papel importante, la descarga de aplicaciones maliciosas pueden actuar de forma que lean la lista de contactos, averiguar la posición GPS, mandar toda esta información por Internet y terminar enviando mensajes SMS.

La seguridad en Android[22] se fundamenta en los siguientes tres pilares:

- Android impide que aplicaciones tengan acceso directo al hardware o interfieran con recursos de otras aplicaciones.
- Toda aplicación ha de ser firmada con un certificado digital que identifique a su autor. La firma digital también nos garantiza que el fichero de la aplicación no ha sido modificado.
- Si se desea modificar la aplicación esta tendrá que ser firmada de nuevo, y esto solo podrá hacerlo el propietario de la clave privada. No es preciso (ni frecuente) que el certificado digital sea firmado por una autoridad de certificación.

[22]TOMAS GIRONES, Jesús. El Gran Libro de Android. Alfaomega Grupo Editor, S.A. de C.V. México: 2012. p. 28

Si se quiere que una aplicación tenga acceso a recursos del sistema se debe utilizar un modelo de permisos de forma que el usuario conozca los riesgos antes de instalar la aplicación.

Usuario Linux y acceso a archivos. Para proteger el acceso a recursos utilizados por otras aplicaciones, Android crea una cuenta de usuario Linux (user ID) nueva por cada paquete (APK) instalado en el sistema. Este usuario es creado cuando se instala la aplicación y permanece constante durante toda su vida en el dispositivo.

Cualquier dato almacenado por la aplicación será asignado al usuario Linux, por lo que normalmente no tendrán acceso otras aplicaciones. No obstante, cuando se crea un fichero se pueden usar los modos mode_world_readable y/o mode_world_writeable para permitir que otras aplicaciones puedan leer o escribir en el fichero. Aunque otras aplicaciones puedan escribir el fichero, el propietario siempre será el usuario asignado a la aplicación que lo creo.

Dado que las restricciones de seguridad se garantizan a nivel de proceso, el código de dos paquetes no puede, normalmente, ejecutarse en el mismo proceso. Para ello sería necesario usar el mismo usuario. Se puede utilizar el atributo shareduserid en AndroidManifest.xml para asignar un mismo usuario Linux a dos aplicaciones. Con esto se consigue que a efectos de seguridad ambas aplicaciones sean tratadas como una sola. Por razones de seguridad, ambas aplicaciones han de estar firmadas con el mismo certificado digital.

El esquema de permisos en Android. Para proteger ciertos recursos y características especiales del *hardware*, Android define un esquema de permisos. Toda aplicación que acceda a estos recursos está obligada a declarar su intención de usarlos. En caso de que una aplicación intente acceder a un recurso del que no ha solicitado permiso, se generara una excepción de permiso y la aplicación será interrumpida inmediatamente. Cuando el usuario instala una aplicación, este podrá examinar la lista de permisos que solicita la aplicación y decidir si considera oportuno instalar dicha aplicación.

Android también ha incorporado seguridad a los usuarios, proporcionando ayuda de cómo funcionan los programas , y el control sobre dichas aplicaciones, debido a que muchas veces los piratas informáticos han tratado de ejecutar ataques a través de la ingeniería social para convencer a los usuarios de configurar e instalar malware en sus dispositivos. El objetivo de seguridad de Android es proteger los datos del usuario, los recursos del sistema y las aplicaciones, es por ello que Android ha incorporado las siguientes características de seguridad:

- **Kernel de Linux:** Linux posee un núcleo estable y seguro. Kernel de Linux tiene varias características de seguridad a saber: modelos de permisos basados en el usuario, el aislamiento de procesos, la comunicación segura y la

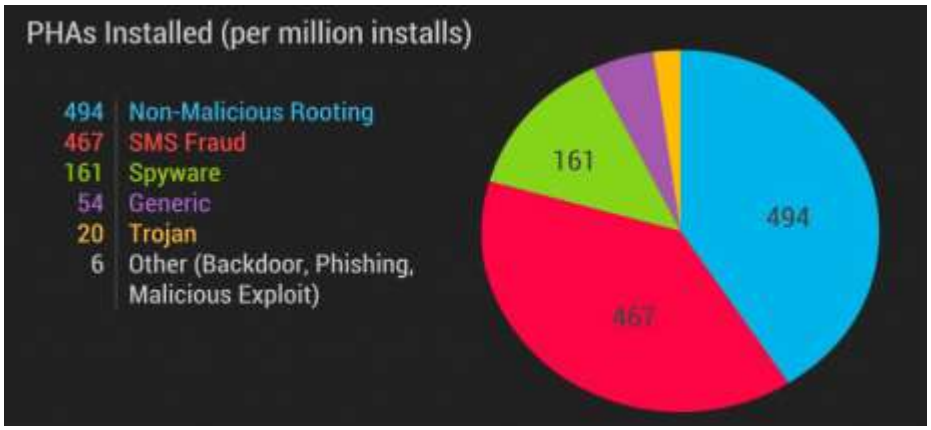
capacidad para eliminar partes innecesarias y potencialmente inseguras del kernel

- **Sandboxing:** Las aplicaciones en Android son independientes, por lo que cada proceso se ejecuta dentro de su propia máquina virtual. Android trabaja basado en programas sobre la exclusión del privilegio, es decir, cada aplicación se ejecuta con su propia ID de grupo (GID) y el ID de usuario (UID), Android evita que los usuarios recuperen los recursos de otro UID. Esto se traduce en tener aislamiento de aplicaciones.

2.4.10 Ataques a dispositivos con sistema operativo Android. Las primeras aplicaciones maliciosas dirigidas a las plataformas móviles más recientes, como iPhone y Android, aparecieron durante el año 2009. Pese a que en el caso del iPhone solo podían infectar a los usuarios que habían realizado el *jailbreak* del terminal, en el caso de Android podían afectar potencialmente a todos los usuarios. A finales de 2009 se descubrió que una nueva aplicación para el acceso a la banca online desde móvil publicada por Android, diseñada para dispositivos Android y disponible en el *Android Marketplace*, recopilaba información bancaria de los usuarios y sus cuentas. Varias entidades financieras alertaron a sus clientes de la existencia de dicha aplicación fraudulenta y de los riegos de su utilización. (Guía de seguridad en dispositivos móviles, 2013, p.28)

En la siguiente figura se muestra el informe publicado a finales de 2013 por el jefe de seguridad en Android, Adrián Ludwig, "*Android, practical security from the ground up*" donde se detallan las aplicaciones potencialmente dañinas que han sido instaladas en dispositivos Android.

Figura11. Estimación de 2013 de aplicaciones peligrosas por millón de instalaciones.



Fuente: Informe Adrián Ludwig. [Citado: Mayo 30 de 2015]. Disponible en: https://docs.google.com/a/ldelgado.es/presentation/d/1YDYUrD22Xq12nKkhBfwoJBfw2QOReMr0BrDfHyfyPw/pub#slide=id.g1202bd8e5_05

Destacan en un primer lugar aplicaciones que requieren de acceso root pero que no son maliciosas. En cambio le siguen, casi en mismo número, las aplicaciones con fraudes por mensaje SMS, seguido de aplicaciones de espionaje. En la cola están los troyanos, puertas traseras, fraudes phishing y exploits maliciosos

Entre las aplicaciones malware[23] conocidas se han destacado:

- **Tap Snake:** Primer caso de malware conocido, aplicación del juego conocido Snake que incluía permisos de acceso a las coordenadas GPS:

  android.permission.**ACCESS_COARSE_LOCATION**y
  android.permission.**ACCESS_FINE_LOCATION.**

- **Fake Player:** Malware creado en agosto del 2010 que pretendía ser un reproductor multimedia. Entre sus permisos estaban *android.permission.* ***SEND_SMS*** teniendo como objetivo el envío de mensajes en ruso SMS a números de tarificación especial a países como Kazajistán.

- **Gemini:** Markets de terceros de dudosa reputación que ofrecen aplicaciones del market de google las mismas aplicaciones gratuitas. Se trata de una aplicación procedente de China que a partir de aplicaciones legítimas modifican el código original incluyendo funcionalidades para **recolectar todo tipo de información privada**.

- **Droid Kungfu:** Primer malware troyano capaz de pasar inadvertido por los antivirus, publicado en 2011permitía infectar versiones de Android 2.2 Froyo y tomar el control del dispositivo. Una vez instalado, DroidKungFu intenta rootear el teléfono para obtener el control del sistema usando el uso de exploits como el método Rageagainstthecage de root. El código se almacena en el paquete del malware y se cifran con una clave para que no pueda ser detectado por un antivirus.

- **Obad:** En Junio de 2013 aparece el troyano Obad con multifuncionalidades como leer/enviar mensajes SMS sin notificar al usuario, iniciar llamadas de teléfono o realizar la instalación de exploits remotamente a través de un canal seguro para ocultarse de la lista de aplicaciones en el administrador. El fichero .dex está cifrado y ofuscado para dificultar la detección de los antivirus. La aplicación antes de ser instalada solicita hasta 24 permisos y realiza además la solicitud de administrador del dispositivo. Además usaba un protector comercial de código para prevenir el proceso de decompilación. En truco consistía en añadir una instrucciones *Goto* en el medio de un bloque *if* con el fin de

[23] ALBELA PEREZ, David. Aplicaciones para plataformas móviles seguridad en android. Facultad de Informática de Coruña, 2013.

confundir al desensamblador y hacer que se genere el código fuente equivocado.

- **AndroRAT:** Herramienta de administración remota para controlar dispositivos Android con la herramienta cliente instalada a través de un servidor en una aplicación Java/Swing. Incluye funcionalidades como obtener la lista de contactos, registro de llamadas, leer mensajes, localización GPS, monitorizar el teclado en tiempo real, sacar fotografías desde la pantalla frontal, grabar sonidos, enviar mensajes de texto, realizar llamadas, abrir urls en el navegador y hacer vibrar el teléfono. La aplicación se puede ejecutar como un servicio y se inicia en el arranque. La conexión con el servidor puede establecerse mediante SMS o llamadas.

Según estudios de securelist que afirma que el 98% va dirigido a dispositivos móviles con sistema operativo Android resumidos en anuncios no deseados, picos de datos para conectarse constantemente a un sitio web no deseado y enviar datos de gran volumen, facturas con costos elevados, aplicaciones no solicitadas de google play, robo de contraseñas, credenciales y datos guardados en el dispositivo, y aplicaciones que piden dinero para desbloquear un aplicativo.

Es así como han aparecido malware móvil destinado a dispositivos con Android, donde se ha encontrado una botnet de spam que está hecha de tabletas y teléfonos Android infectados.

Se han llevado a cabo estudios que analizan los ataques realizados por malware para móviles y Android es, el objetivo principal del software malicioso, el sistema operativo de Google se lleva el 96.5% de los ataques a teléfonos celulares según un informe emitido por Fortinet.

2.5 MARCO CONCEPTUAL

2.5.1 Definición de dispositivo móvil. Los dispositivos móviles son aquellos dispositivos que usan personas o empresas, que permite manejar la información desde cualquier sitio donde se encuentre, acceso a redes de comunicaciones tanto de voz como datos. Entre los dispositivos móviles de uso más común se encuentran:

- Teléfonos móviles
- Smartphones (Teléfonos avanzados o inteligentes)
- Agendas electrónicas (PDA)
- Tabletas
- Paginadores
- Comunicadores de bolsillo
- Sistemas de navegación de automóviles
- Sistemas de televisión en Internet (web TV)

2.5.2 Características de un dispositivo móvil

- Portables, facilidad de traslado a cualquier sitio por ser generalmente de pequeño tamaño.
- Permiten conexión permanente a Internet o a una red
- Memoria limitada
- Con algunas capacidades de procesamiento
- Generalmente se asocian al uso individual de una persona, tanto en posesión como operación, y fácilmente configurables a gusto del usuario

2.5.3 Sistema operativo. En ingles (OperatingSystem OS) es el programa o conjunto de programas que efectúan la gestión de los procesos básicos de un sistema informático, y permite a un usuario interactuar con el sistema [24]

Qué sistemas operativos utilizan los Smartphones?. Atendiendo los requerimientos de la población mundial, los fabricantes han diseñado sistemas operativos móviles que permiten gestión de procesos dentro del dispositivo, así como el funcionamiento de programas y aplicaciones que se instalan posteriormente. Dentro de los sistemas operativos más utilizados en la actualidad se encuentran:

- Android: Diseñado por Google, es el sistema operativo con mayor uso a nivel mundial, con casi el 50,9% del volumen de ventas. Es de código abierto y se lo puede configurar a gusto del usuario.
- iOs: Creado por Apple para IPhone, es de carácter cerrado y el segundo sistema operativo en uso.
- Symbian: Perteneciente a la empresa Nokia, y utilizado actualmente en terminales de gama media y baja.
- BlackBerry: Enfocado específicamente al mundo empresarial, desarrollado por RIM. Es multitarea y con soporte para uso de computadoras de mano como la trackwhell, trackball, touchpad y pantallas táctiles.

2.5.4 Máquina virtual. Es una implementación de Software de una máquina (por ejemplo, una computadora) que ejecuta programas como una máquina física. Específicamente, una máquina virtual de procesos está diseñada para ejecutar un único programa, lo que quiere decir que soporta un único proceso. Uno de le ejemplos más conocidos de Máquina Virtual es la utilizada para Java. En el caso de Android, la máquina virtual utilizada para ejecutar las aplicaciones se conoce como Dalvik.

[24] Real Academia Española. Sistema Operativo. [en línea]. http://buscon.rae.es/draeI/SrvltObtenerHtml?origen=RAE&LEMA=sistema&SUPIND=0&CAREXT=10000&NE DIC=No#sistema_operativo..

2.5.5 API. Conjunto de funciones o métodos (en la programación orientada a objeto) que le permiten a un programador acceder a las características de hardware de un dispositivo.

2.5.6 Aplicación nativa. Es aquella que se instala en el propio dispositivo y se desarrolla utilizando un lenguaje de programación compatible con el sistema operativo del dispositivo.

2.5.7 Aplicación WEB. Es aquella que se encuentra instalada en un servidor tipo web o un browser y necesita de él para ejecutarse, dicho navegador debe de ser compatible con las tecnologías con las cuales se realiza el aplicativo WEB y que correrán de parte de este, tales como AJAX, JSON, CSS, entre otros.

2.5.8 Kernel[25]**.** El kernel o núcleo de Linux se puede definir como el corazón de este sistema operativo. Es el encargado de que el software y el hardware del computador puedan trabajar juntos.

Las funciones más importantes del mismo, aunque no las únicas, son:

- Administración de la memoria para todos los programas y procesos en ejecución.
- Administración del tiempo de procesador que los programas y procesos en ejecución utilizan.
- Es el encargado de que se pueda acceder a los periféricos/elementos del dispositivo.

2.5.9 ADB[26]**.** El Adb (Android Depuration Bridge) es una herramienta incluida en el SDK de Android. Permite hacer cambios en el dispositivo en con un emulador, mediante la línea de comandos usando comandos Unix compatibles incluidos en el sistema

2.5.10 Dalvik. Máquina virtual que usa Android. Optimizada para requerir poca memoria y permite ejecutar varias versiones simultáneamente, aliviando el sistema. Las versiones modernas de Android usan de forma predeterminada ART (Android Runtime).

2.5.11 NFC. Near-Field Communication o Comunicación de Campo Cercano, es un estándar creado para la comunicación sin cables de corto alcance para la realización de pagos a través de dispositivos móviles mayoritariamente.

2.5.12 Rootear. Modificar el sistema para poder disfrutar de permisos de root (administrador, superusuario).Es imprescindible para poder usar ciertas

[25] Kernel. Disponible en: http://www.linux-es.org/kernel. Ultimo acceso: Mayo 28 de 2015
[26] http://norfipc.com/celulares/diccionario-terminos-tecnicos-usados-android.php

aplicaciones. Es algo similar al Jailbreak que se realiza en los dispositivos de Apple para tener un control total del dispositivo.

2.5.13 SDK. Un paquete de programas de desarrollo de software o SDK (siglas en inglés de software development kit). Es un conjunto de herramientas de desarrollo que le permite a un programador o aficionado con conocimientos básicos, crear aplicaciones para Android.

3. DISEÑO METODOLÓGICO

La investigación sobre seguridad en dispositivos móviles con sistema operativo Android depende del desarrollo de varias etapas en la adquisición e investigación del conocimiento, realización de entrevistas y análisis de información recolectada.

Etapa 1: Apropiación del conocimiento en dispositivos móviles y sistema operativo Android. Con el objeto de crear un referente teórico que sirva de base para el desarrollo del proyecto de grado y del documento a presentar.

Etapa 2: Investigación de los diferentes ataques informáticos e incidentes de seguridad presentados a este tipo de sistema operativo y detección de vulnerabilidades en el mismo.

Etapa 3: Práctica para establecer y analizar las vulnerabilidades encontradas en el sistema operativo Android 4.4.

Etapa 4: Elaboración del documento final con análisis de resultados y con el desarrollo de una estrategia donde se describa la aplicación de buenas prácticas y para la utilización de este tipo de dispositivos que será publicada igualmente en las redes sociales para conocimiento y sensibilización hacia los usuarios de estos dispositivos.

3.1 FUENTES DE INFORMACIÓN

Se utilizará todas las fuentes bibliográficas como libros, revistas, artículos, tesis que aborden el tema y toda la información relacionada con los dispositivos móviles con sistema operativo Android.

4. ANÁLISIS SISTEMA OPERATIVO ANDROID

4.1 INTRODUCCIÓN

Existen varias razones por las que se ha elegido esta plataforma para hacer pruebas de seguridad, por tratarse de un sistema operativo de código abierto lo que permite obtener código fuente del mismo para poder examinarlo. Múltiples partes del sistema son también de código abierto, por lo que es posible acceder a él. Se optó por Android 4.4 ya que es el sistema operativo más utilizado en la actualidad en dispositivos móviles y por ende el más vulnerable de ser atacado.

En el presente capítulo se realizará prueba de pent-testing, se simulará un ataque man in the middle a un dispositivo android; también se analizará mediante técnicas de ingeniería inversa una aplicación APK para establecer las vulnerabilidades del sistema y analizar si el sistema y aplicaciones se consideran seguras, aunque ninguna aplicación o sistema operativo es 100% seguro, pero conocida la importancia de cada activo para el usuario se determinará las posibles vulnerabilidades existentes.

4.2 OBJETIVO

Analizar el Sistema Operativo Android 4.4 a través de herramientas de pent-testing que permitan establecer vulnerabilidades existentes para generar políticas de seguridad a ejecutar que protejan la información que se almacena en este tipo dispositivos.

4.3 METODOLOGÍA

El presente capítulo recogerá todo el proceso de creación y comprobación del método propuesto para realizar auditoria de seguridad al Sistema y a algunos de sus componentes. En primer lugar, se describirá los elementos que hay que tener en cuenta en el análisis y las herramientas empleadas para obtener resultados en las pruebas, en este caso se conocerá el sistema, sus componentes principales, acceso a bases de datos, luego se hará una simulación de ataque a un dispositivo móvil mediante la instalación de una APK maliciosa para poder alterar la cámara web y se analizará una aplicación APK para determinar debilidades en el código fuente.

A continuación, se detallará el diseño de las pruebas, es decir, qué pruebas se van a llevar a cabo, con qué herramientas, qué datos se van a estudiar y qué información se espera obtener. Posteriormente, se mostrarán los resultados obtenidos en las pruebas sobre las aplicaciones y se determinará si cada

aplicación es lo suficientemente segura. Para concluir, se realizará un comentario crítico analizando y resumiendo los diferentes resultados obtenidos.

4.4 PRUEBAS A EJECUTAR

La siguiente tabla muestra las pruebas que se ejecutarán con el fin de establecer que vulnerabilidades se encuentran encontrar elementos sensibles y en qué estado se encuentran.

Tabla 1. Pruebas a ejecutar

PRUEBA	DESCRIPCION
ANALISIS DEL SISTEMA OPERATIVO	Se accederá a la estructura del sistema operativo mediante técnicas de pent-testing, para analizar los siguientes elementos: - Localización de archivos importantes del sistema - Localización de aplicaciones, - Localización de bases de datos, extracción de datos - Permisos del sistema - Bootloader - Análisis de tráfico
EJECUCION DE UN ATAQUE A UN DISPOSITIVO ANDROID	Se realizará una simulación de ataque a una dispositivo Android versión 4.4 mediante técnicas Man in the Middle, para verificar vulnerabilidades existentes en el mismo, en este caso se utilizará Metasploit Framework.
ANALISIS DE UNA APLICACIÓN APK	Se realizará análisis de una .APK para determinar vulnerabilidades, mediante técnica de ingeniería inversa.

Fuente: Esta investigación

4.5 HERRAMIENTAS DE DEPURACIÓN

La SDK de Android ofrece suficientes herramientas para realizar una depuración completa.

ADB: es una herramienta que se sitúa entre el dispositivo y el sistema de desarrollo. Provee al desarrollador de funcionalidades de gestión como

sincronización de archivos, consola UNIX y comunicación entre dispositivos conectados y emuladores.

DbBrowser SQLite: Es una herramienta de código abierto para crear, diseñar y editar archivos de bases de datos compatibles con SQLite. Sirve además para crear bases de datos, importar y exportar registros, emite consultas SQL, exportar tablas a archivos CSV y otras funciones que la hacen una herramienta potente para bases de datos.

Kali Linux: Kali Linux es una distribución de Linux avanzada para pruebas de penetración y auditorías de seguridad. Kali es una completa re-construcción de BackTrack Linux desde la base hacia arriba, y se adhiere completamente a los estándares de desarrollo de Debian.

APKtool: Las aplicaciones Android se empaquetan en archivos "*APK*", que al igual que los archivos "*jar*" no son más que archivos "*zip*" sin encriptación, por lo que simplemente usando herramientas como APKtool se los puede descomprimir y extraer todos sus recursos. APKtool Compila y decompila aplicaciones de android .APK,

Dex2jar: Toma un archivo **APK** o el ***classes.dex*** y devuelve un archivo ***.jar***, que se puede abrir con decompiladores de Java como **JD-GUI** y acceder al código de la aplicación en **Java**. El principal objetivo de realizar este tipo de operaciones para acelerar el análisis de la amenaza, o evitar pasar grandes cantidades de tiempo comprendiendo la estructura de la aplicación. Las aplicaciones de Android, se desarrollan principalmente en Java, y como se trabaja sobre una máquina virtual es posible obtener una salida con un alto nivel de similitud al código original.

Wireshark*:* Está basado en la API *pcap* diseñada para la captura de paquetes de red y añade interfaz de usuario. La aplicación es capaz de filtrar más de 1100 protocolos y mostrar la información de manera estructurada, con todos los campos de las cabeceras y capas de los paquetes capturados.

4.6 ANÁLISIS SISTEMA OPERATIVO ANDROID 4.4

Para realizar el pent-testing se utilizó una serie de herramientas como Android SDK, ADB, con las cuales se pueden realizar test de penetración y verificar tanto la estructura del sistema como las vulnerabilidades que se pueden encontrar realizando distintas pruebas.

ANÁLISIS DEL CELULAR CONECTADO AL PC

Herramienta: ADB

Objetivo: Conocer las características del celular y poder ingresar al sistema operativo para observar sus componentes.

Descripción del proceso

ADB Es una herramienta de línea de comandos versátil que permite comunicarse con una instancia de emulador o dispositivo con Android conectado al equipo. Es un programa cliente - servidor

Componentes del ADB

La comunicación del ADB se basa en un modelo cliente-servidor, dónde sus componentes son:

- Cliente: se ejecuta en el entorno de desarrollo desde la consola de comandos. El DDMS también ejecuta clientes ADB y es más sencillo de utilizar.
- Servidor: se ejecuta como un proceso en background en el entono de desarrollo. El servidor controla la comunicación entre el cliente y el demonio que se ejecuta en el dispositivo.
- Demonio: se ejecuta en background sobre cada instancia de un dispositivo.

Ciclo de ejecución de ADB

El ciclo de ejecución del ADB es el siguiente:

1. Se ejecuta el cliente ADB
2. Se comprueba si hay procesos de servidor ADB en marcha.
2.1. Si no, se crea un proceso de servidor ADB.
2.2. El servidor hace un bind al puerto local 5037 TCP.
2.3. El servidor escucha los comandos enviados por los clientes (todos se comunican por ese puerto)
2.4. El servidor configura todos los dispositivos en ejecución (se les aginan puertos impares desde el 5555 al 5585).
2.5. Cuando el servidor encuentra un demonio ADB, configura la conexión del puerto.
3. Una vez conectado el servidor, se configuran las conexiones de todas las instancias para poder usar comandos de control y acceso a las instancias. Se puede controlar cualquier dispositivo desde cualquier cliente.

Tipo de celular analizado:

MODELO: Ilium S3130
Versión de Android: 4.4.2
Versión de núcleo: 3.4.67
Versión de software: ILIUM S3130_CLARO_SW_01

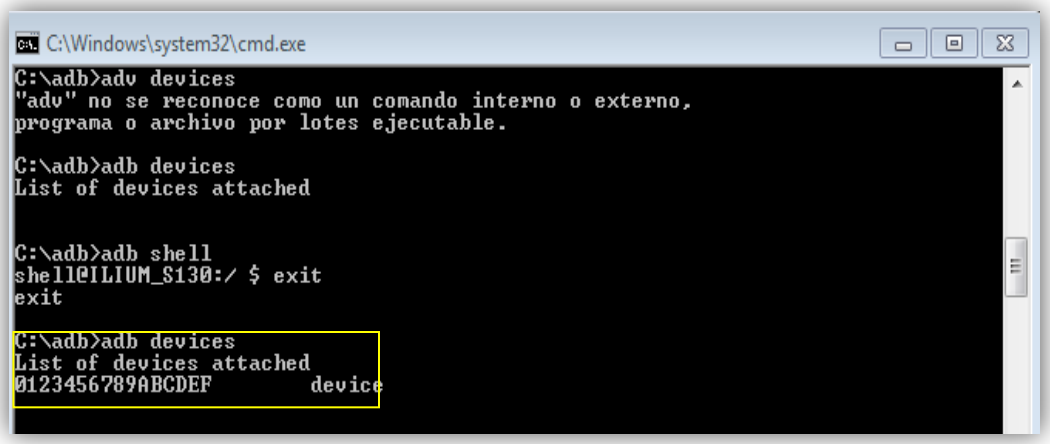
Figura 12. Características Móvil Ilium S3130

Fuente: Esta investigación

1. Se ejecuta el comando ***adb devices*** dentro de la carpeta Adb para verificar la existencia de la conexión.

C:/ adb/ adb devices

Figura 13. Conexión con el comando adb



Fuente: Esta investigación

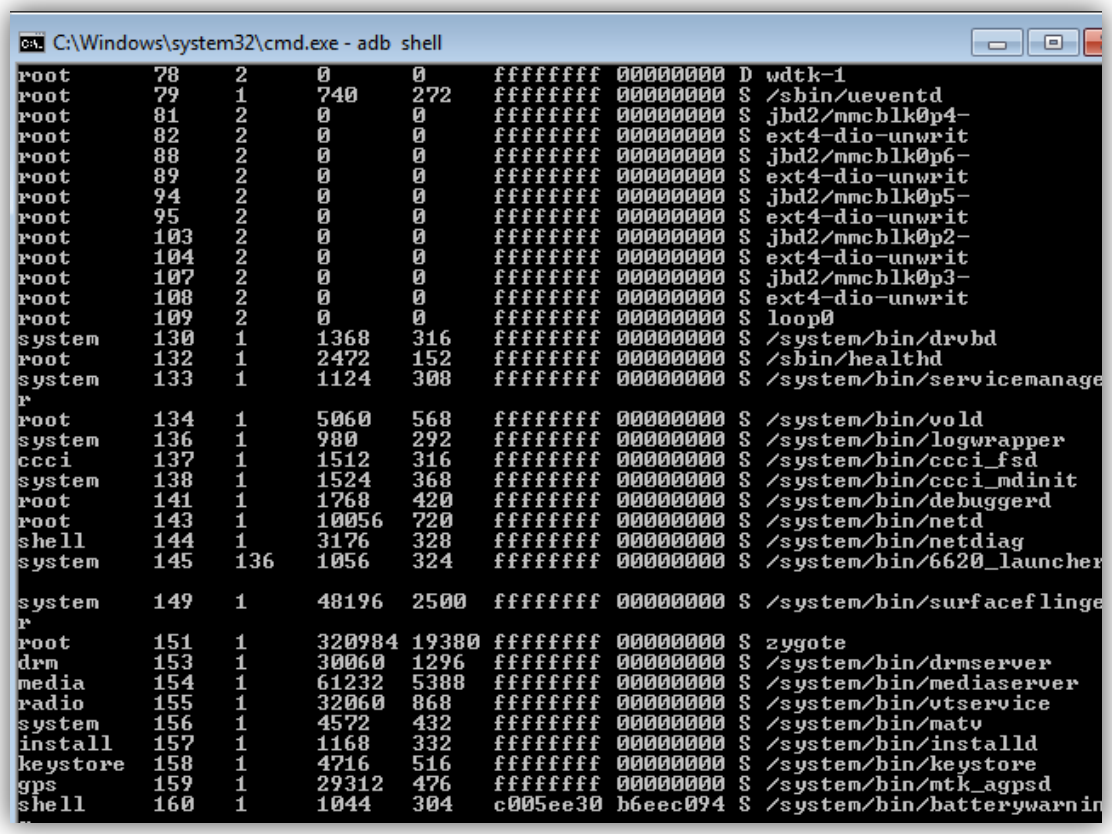
2. Una vez conectado se ejecuta el comando **adb shell** el cual da una directa interacción con el dispositivo y donde se pueden ejecutar comandos y desarrollar acciones para analizar la información del dispositivo. En la figura 14 se puede observar la conexión del dispositivo al equipo.

Figura 14. Conexión a la Shell adb

Fuente: Esta investigación

3. Una vez en el shell se ejecuta el comando **ps** para correr la lista de procesos

Figura 15. Lista de procesos



Fuente: Esta investigación

En este caso se observa que ps lista todos los procesos que actualmente se ejecutan en el Sistema Android, de igual manera se observa la columna de usuarios y la gran variedad que tiene como root, system, gps, etc. Los procesos en ejecución del sistema son propiedad del sistema ejecutados en la raíz o root, otros como radio son procesos relacionados con la telefonía y app son aplicaciones que se han descargado e instalado en el dispositivo, es así como en Android un usuario identifica una aplicación/proceso que se ejecuta en su propio entorno.

El modelo de seguridad de Android consiste en la separación de privilegios, donde cada vez que se inicia una nueva aplicación se le asigna un identificador de usuario único (UID) que pertenece además a algún u otros grupos. Además, los datos de la aplicación que se instalan desde Play Store o cualquier otra fuente se encuentra en /data/data (Ver figura 16), mientras que su archivo de instalación original, es decir, .APK se almacenará en /data/app (Ver figura 17). También, hay algunas aplicaciones que necesitan ser compradas en la tienda Play en lugar de sólo descargarlas de forma gratuita. Estas aplicaciones se almacenan en / data /

app- privada /.Si se observan las diferentes subcarpetas dentro de data, se puede observar archivos, de bases de datos, cache el cual se podrán observar posteriormente en herramientas de auditorías para aplicaciones.

Figura 16. Datos de aplicaciones

Fuente: Esta investigación

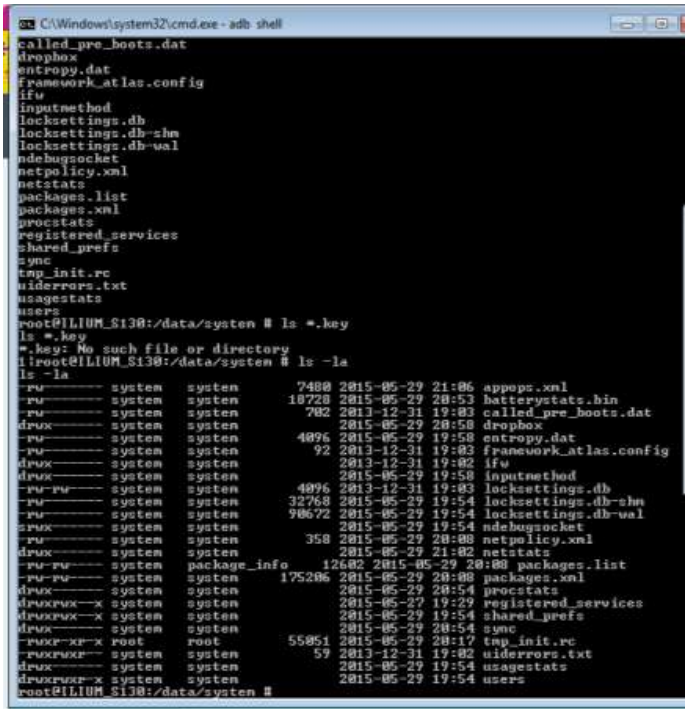
Figura 17. Archivos de instalación de Android 4.4

Fuente: Esta investigación

Un aspecto importante es que si el teléfono esta rooteado, se puede modificar cualquier archivo del sistema, por lo que se tiene acceso pleno y el control sobre todo el dispositivo y modificar los archivos que se deseen.

Otra de las cosas que se pueden realizar es desbloquear el patrón de bloqueo, conectando el teléfono ya que la contraseña se almacena en /data/system (figura 16) con el nombre de pasword.key o gesture.key.

Figura 18. Estructura del directorio /data/system



Fuente: Esta investigación

Como el celular analizado no tiene patrón de bloqueo, el archivo *.key no se encuentra en la carpeta system.

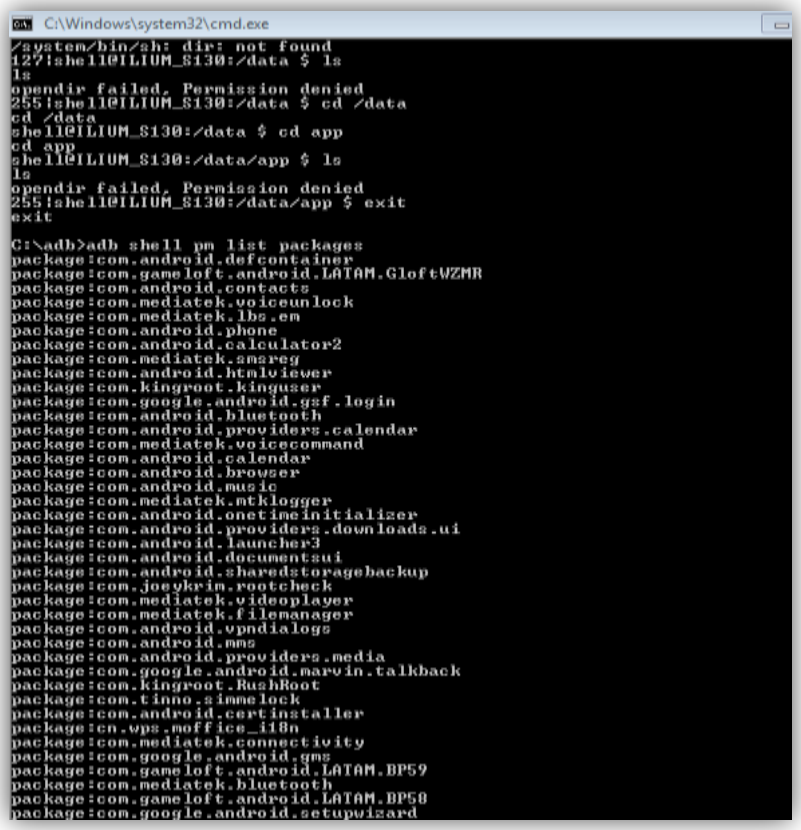
4. Listar paquetes instalados en Android

Se pueden listar los paquetes instalados en android, mediante el comando:

adb shell pm list packages

Lo cual retornará un listado de paquetes instalados en el sistema como se muestra en la figura 19:

Figura 19. Lista de paquetes instalados en Android



Fuente: Esta investigación

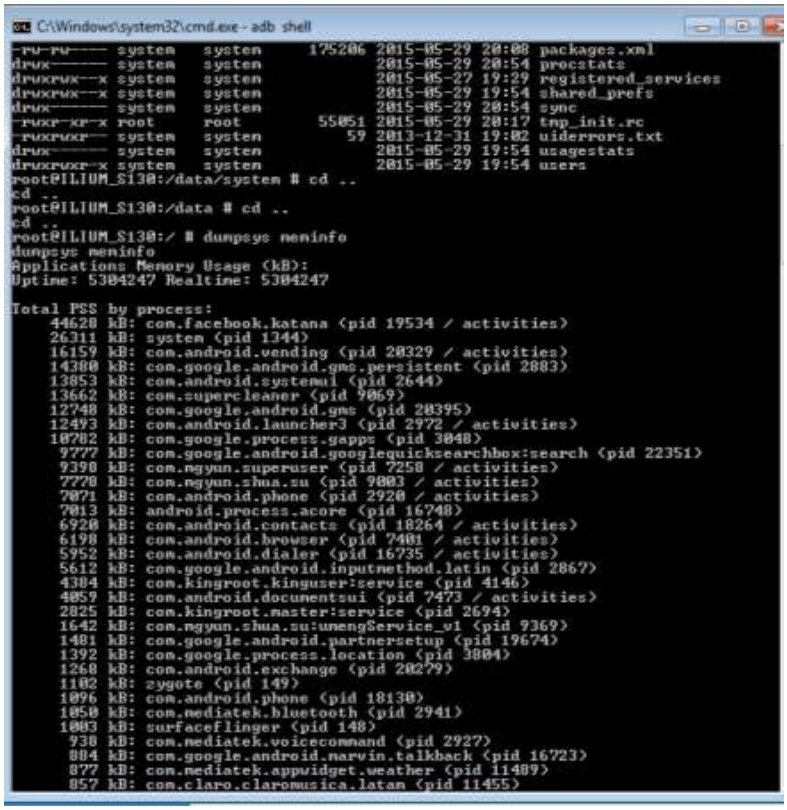
Figura 20. Contenido de los paquetes

Fuente: Esta investigación

5. Lista de aplicaciones en memoria actual

También se puede conseguir una lista de aplicaciones en la memoria actual y su consumo utilizando el comando **dumpsys meminfo** que arroja la siguiente información:

Figura 21. Lista de aplicaciones en memoria actual



Fuente: Esta investigación

6. Extracción de datos

Para extraer datos el dispositivo debe estar totalmente rooteado como se ha indicado en las figuras anteriores, hay dos maneras de extraer datos:

1. **Por medio de ADB:** Como se ha venido explicando, adb es un protocolo que ayuda a conectarse a un dispositivo android.
2. **Extracción por medio del gestor de arranque:** Esto puede hacerse cuando el dispositivo está en modo de gestor de arranque o bootloader.

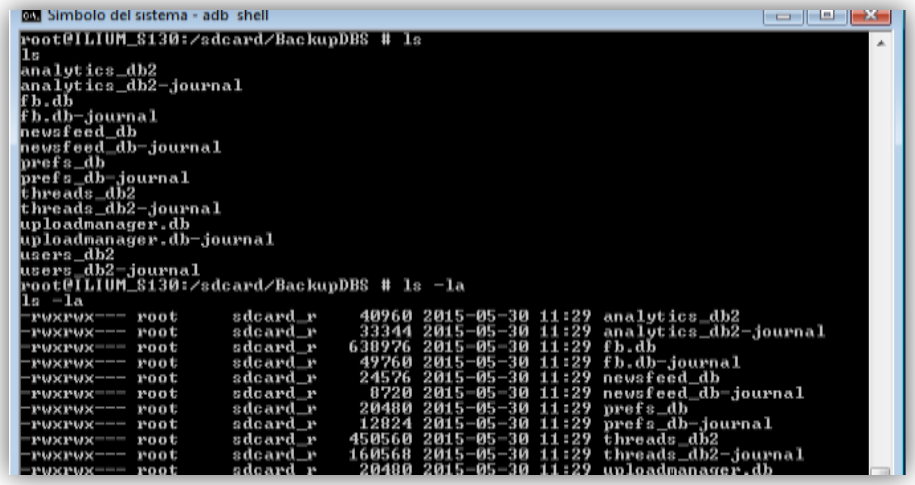
Antes de la extracción de los datos, es importante saber cómo se almacenan los datos en el dispositivo Android para entender dónde buscar y qué datos extraer.

Los datos de Android se encuentran en los siguientes sitios:

- **Compartir Preferencias:** Los datos se almacenan en pares clave-valor. Archivos de preferencias compartidas se almacenan en el directorio 'datos' de aplicación en la carpeta 'shared_pref'.
- **Almacenamiento interno:** Almacena datos privados del dispositivo en una memoria interna (por ejemplo flash NAND).
- **Almacenamiento externo:** Almacena datos públicos en la memoria externa del dispositivo que podría no contener mecanismos de seguridad. Estos datos están disponibles en el directorio / sdcard.
- **SQLite:** Esta es una base de datos que contiene los datos estructurales. Estos datos están disponibles en / data / data / Paquete / base de datos.

Bases de datos. Para la presente prueba se realiza una extracción de la base de datos de Facebook localizada en el dispositivo móvil del presente análisis y de propiedad de la autora del presente documento. Como se mencionó anteriormente las bases de datos de cada paquete se encuentran almacenadas en la ruta: /data/data/paquete/bases de datos. En el dispositivo del presente análisis se dirige a la ruta /data/data/com.facebook.katana/databases tal como muestra la siguiente figura:

Figura 22. Acceso a bases de datos de aplicaciones



Fuente: Esta investigación

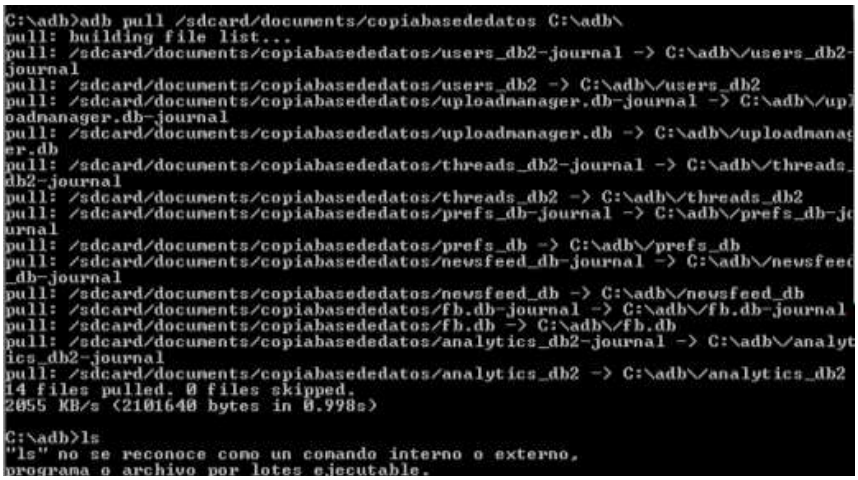
Todos los archivos *.db indican las bases de datos presentes para la aplicación Facebook.

Ahora si se quiere mirar el contenido de esas bases de datos se puede extraer el archivo *.db y mirarlos a través de del comando adb pull, se copia los archivos al adb para tenerlos localizados en esta carpeta de prueba mediante el comando:

adb pull /sdcard/documents/copiabasededatos/*.db C:\adb

Previamente se había realizado una copia de las bases de datos de facebook a esta ruta. Por lo tanto, nos copia a la carpeta de adb, todas las bases de datos extraídas del teléfono, como se muestra la figura 23:

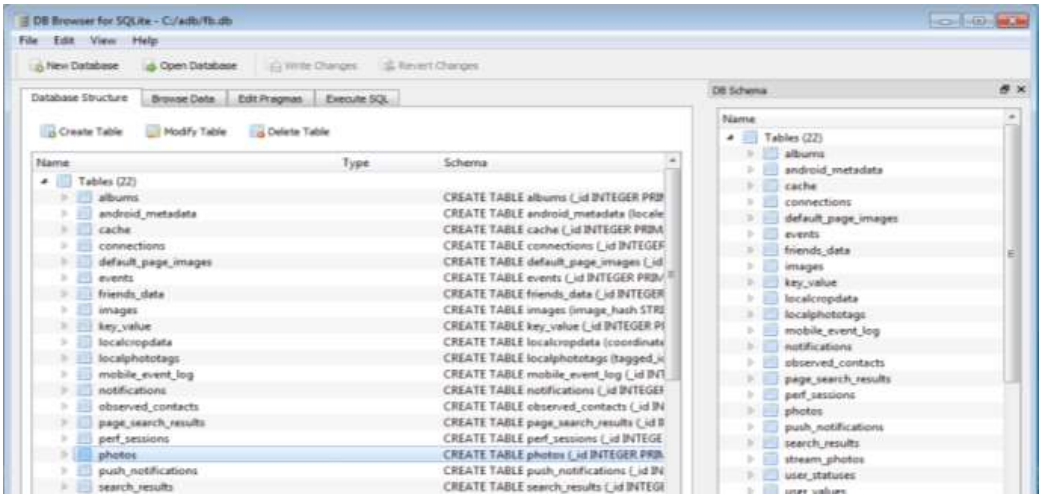
Figura 23. Extracción de datos de las bases de datos en android



Fuente: Esta investigación

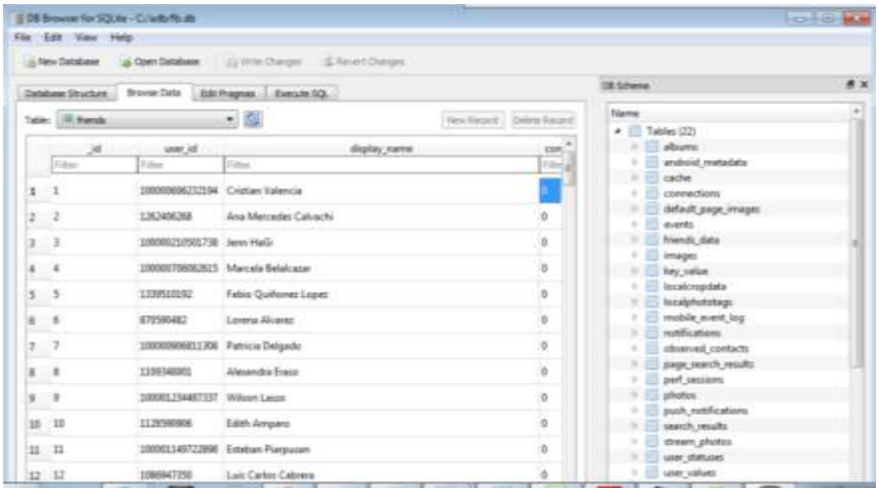
Mediante la herramienta DB Browser for SQlite, se observa la base de datos extraída, que es la base de Facebook:

Figura 24. Estructura de las tablas en facebook



Fuente: Esta investigación

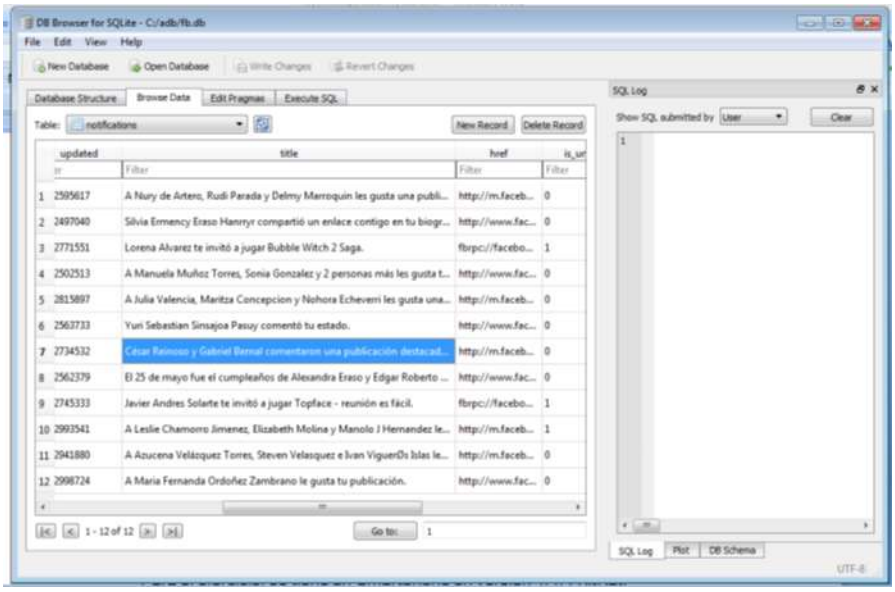
Figura 25. Lista de contactos de Facebook – tabla friends.



Fuente: Esta investigación

En la lista de contactos claramente aparece el id, nombre, correo electrónico, fecha de nacimiento y celular de quienes hayan agregado este tipo de información.

Figura 26. Tabla de notificaciones de facebook



Fuente: Esta investigación

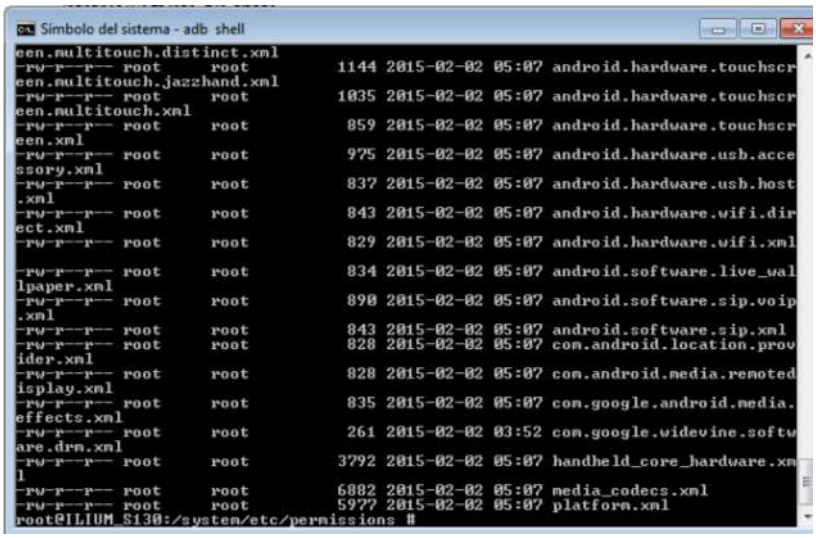
Con esto queda demostrado que se puede acceder fácilmente a un dispositivo, que aunque tiene aspectos positivos, ya que se puede acceder a código fuente y verificar la estructura del sistema, existe la vulnerabilidad que pueda ser accedido

por un ciberdelincuente y realizar modificaciones de datos, información, denegación de algún servicio, etc.

7. Permisos del sistema

De igual forma se puede ver en la ruta /system/etc/permission los diferentes permisos con los que cuenta el sistema, cada uno viene con un archivo xml por separado que se podría analizar para establecer cómo está el nivel de seguridad del dispositivo

Figura 27. Esquema de permisos en Android

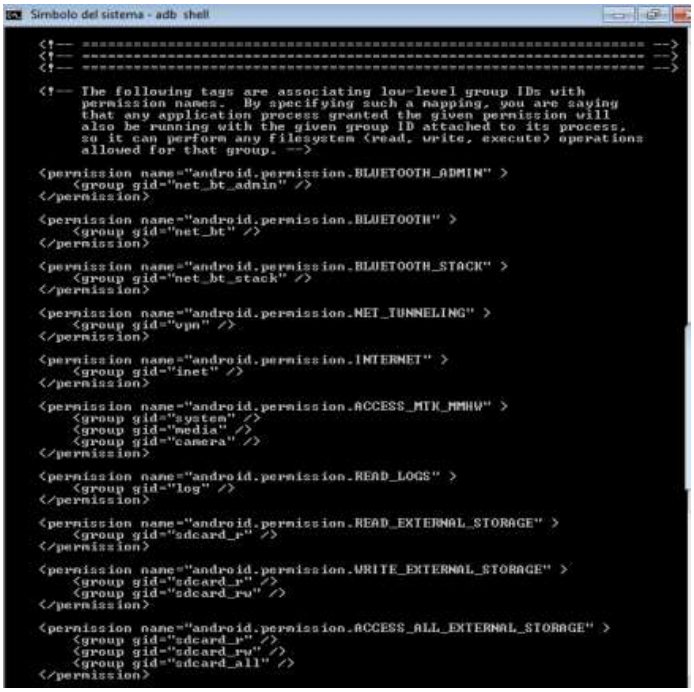


Fuente: Esta investigación

Para el caso anterior se puede observar que existen permisos para acceso a usb, wi-fi, llamadas telefónicas, cámara, localización, etc., donde tiene permisos el root como propietario de estos archivos de lectura y escritura, para grupos y el resto solo hay permisos de lectura. Cada usuario puede tener múltiples grupos y cada grupo múltiples usuarios, un grupo tiene un único nombre o identificador llamado Group Id (GID), por lo que es importante establecer un buen esquema de permisos. También cada usuario tiene un único identificador ID.

En la figura anterior también se observa un archivo llamado **platform.xml**que al observar su contenido se encuentran los permisos y de qué forma distribuye los permisos entre usuarios y grupos tal como se observa a continuación:

Figura 28. Distribución de permisos entre usuarios



Fuente: Esta investigación

Como se observó anteriormente, cada aplicación almacena sus datos en /data/data/nombre_del_paquete que tendrán el mismo ID de usuario, lo que forma el modelo de seguridad de Android. Depende del de UID y los permisos de archivos que otras aplicaciones puedan permitir o restringir el acceso. Sin embargo, se puede leer el contenido desde una tarjeta SD sin necesidad de ningún tipo de permiso, y una vez el atacante tenga los datos puede abrir el navegador y enviar los datos con un POST/GET haciendo petición a un servidor remoto, donde se guardará, espacio propicio para realizar un malware.

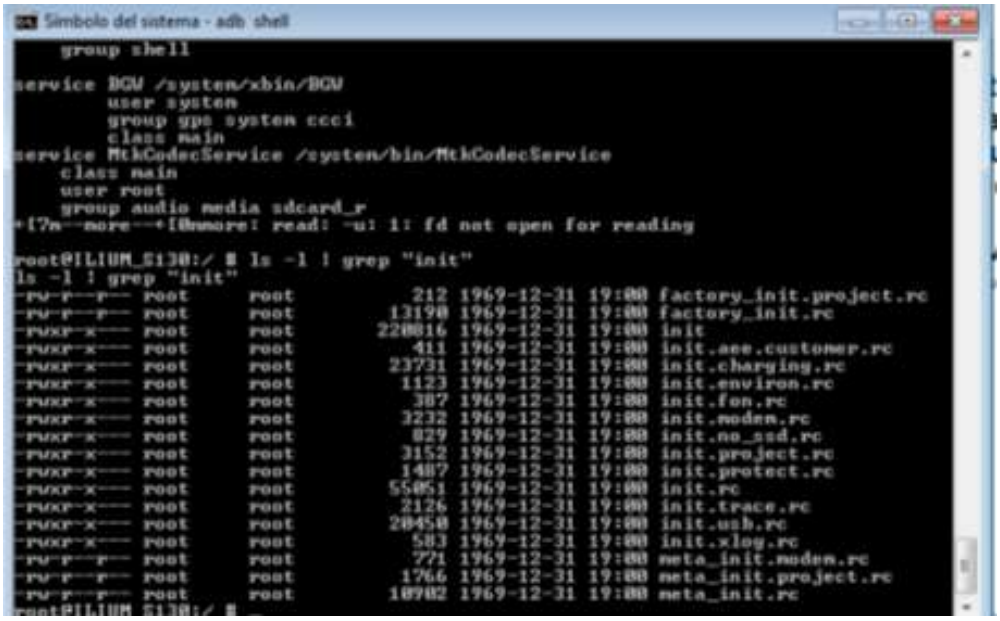
8. Bootloader

El bootloader es el gestor de arranque del Sistema Operativo Android y uno de los elementos de seguridad más importantes para analizar, donde se ejecutan los principales procesos para que el sistema pueda funcionar. Estos procesos se montan sobre algunos directorios importantes como /dev /sys y /proc.

También se toma la configuración de los archivos init.rc einit . [devicename].rc, y en algunos casos a partir de los archivos .sh para el arranque

Se puede listar los archivos * init mediante el siguiente comando: ls-l I grep “init” que mostrará los archivos de inicio

Figura 29. Archivos .init localizados en el dispositivo.



Fuente: Esta investigación

El código en el bootloader es diferente en cada Android. Muchas empresas no permiten desbloquear el bootloader porque permite modificar el sistema operativo del teléfono, y ellas consideran que ese es el idóneo para ese dispositivo, pero desbloquearlo puede servir para muchas cosas.

9. Análisis de tráfico

Para el análisis de tráfico se utiliza la herramienta Wireshark hacia la dirección ip 192.168.1.3 como muestra la figura 30.

Figura 30. Dirección Mac e Ip del dispositivo analizado

Fuente: Esta investigación

Las pruebas de comunicación tienen como objetivo detectar como se transmite la información mediante la captura de paquetes transmitidos. Es primordial realizar un seguimiento de los diferentes protocolos y tipos de paquetes para detectar el inicio o fin de ciertas acciones. En primer lugar, los datos en claro se transmiten, normalmente, con el protocolo HTTP. Por lo tanto, hay que filtrar los paquetes HTTP y analizar la sección de datos en busca de elementos marcados como activos de la aplicación. A continuación, hay que comprobar que los datos están siendo transmitidos por HTTPs. Existen varias posibilidades, pero la más sencilla es buscar los paquetes que tiene como puerto destino el 443. Las aplicaciones se prueban sin información almacenada en la base de datos del dispositivo o en caché. Es decir, las pruebas se realizan como si la aplicación estuviera recién instalada.

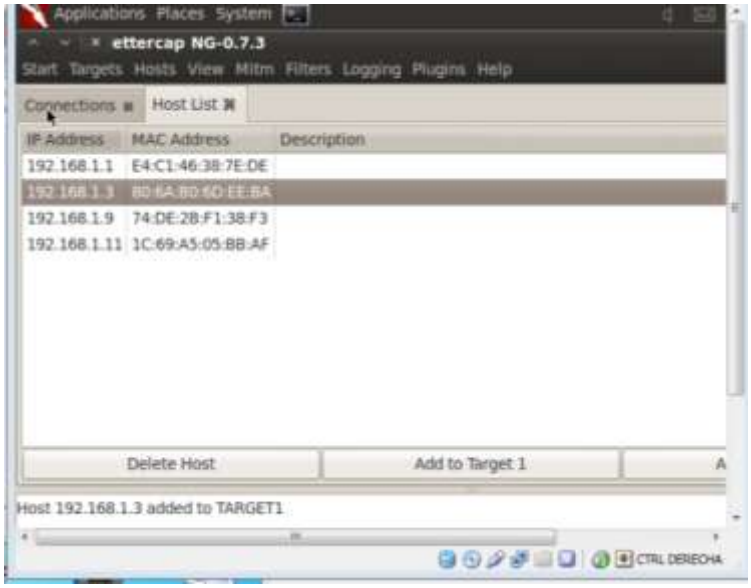
Sin embargo al realizar las pruebas con wireshark no se obtuvo resultado alguno, ya que el celular no tiene puertos abiertos tal como lo muestra la figura 31 donde se realizó un nmap, a través de backtrack a la dirección 192.168.1.3

Figura 31. Búsqueda de puertos abiertos en el dispositivo

Fuente: Esta investigación

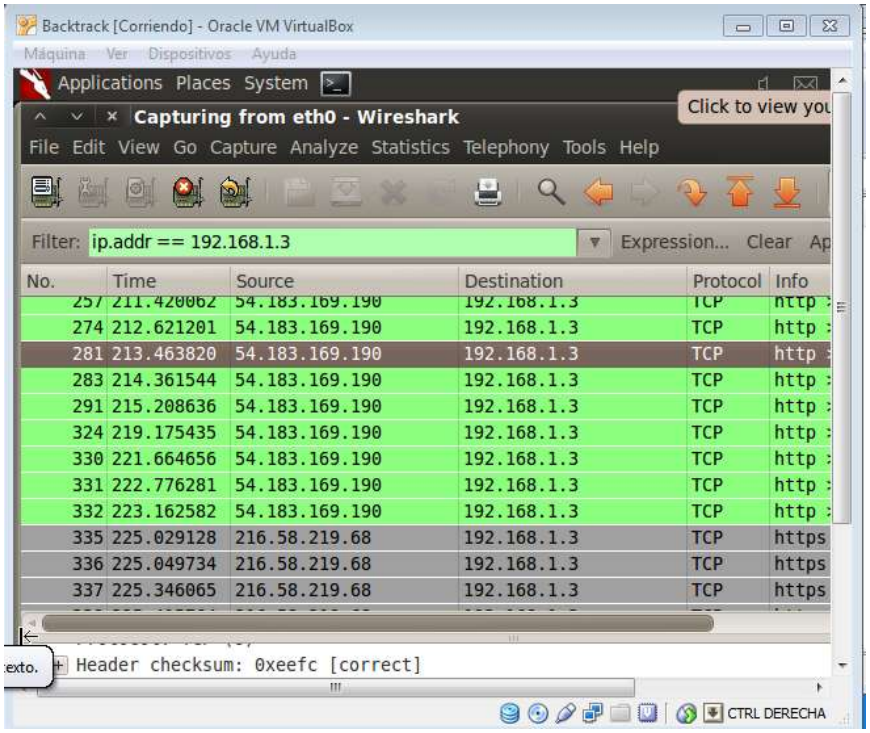
Pero al intentar realizar un ataque tipo ARP-spoofing donde reconoció la ip del dispositivo 192.186.1.3 a través de la herramienta ettercap, empezó el wireshark a capturar los paquetes del dispositivo móvil ya que los redirigió hacia la máquina atacante (backtrack), lo que indica que se puede acceder al dispositivo y hacer un ataque tipo spoofing como se muestra en las siguientes figuras:

Figura 32. Ataque spoofing con la herramienta ettercap de backtrack Linux.



Fuente: Esta investigación

Figura 33. Redireccionamiento del tráfico hacia la máquina atacante y captura de paquetes del dispositivo



Fuente: Esta investigación

4.7 SIMULACIÓN ATAQUE A UN DISPOSITIVO ANDROID

Ya que en muchas ocasiones las personas son un poco confiadas y creen que es poco probable, que su dispositivo con Android sea vulnerado por alguien más, a continuación se evidenciará, lo sencillo que puede ser tomar control remoto de un Smartphone y obtener de él lo que se quiera.

TIPO DE ATAQUE: MAN IN THE MIDDLE. Los ataques “Man in the Middle” son ataques en los que una tercera persona adquiere la posibilidad de leer, insertar y modificar a voluntad los paquetes entre dos partes sin que ninguna de ellas conozca que el enlace entre ellos ha sido alterado.

Para realizar dichos ataques desde dSploit se ejecutara la aplicación en el dispositivo Android y esperar a que el programa analice la red.

Una vez que detecte todos los equipos de la red se debe elegir quien será la víctima. Se puede elegir realizar el ataque a toda la máscara de subred, a la puerta de enlace del router o a un determinado equipo de la red. Para ello se debe seleccionar el destinatario del ataque.

Para el ejercicio, se tiene un Smartphone en versión 4.4.4 KitKat y se utilizará la herramienta Kali Linux para las diferentes pruebas, con metasploit que es una herramienta que permite ejecutar y desarrollar ***exploits*** contra sistemas objetivos.

Figura 34. Características del dispositivo

Fuente: Esta investigación

Se tienen varias aplicaciones instaladas en él y la dirección IP para la prueba de conexión al final es la 192.168.1.14.

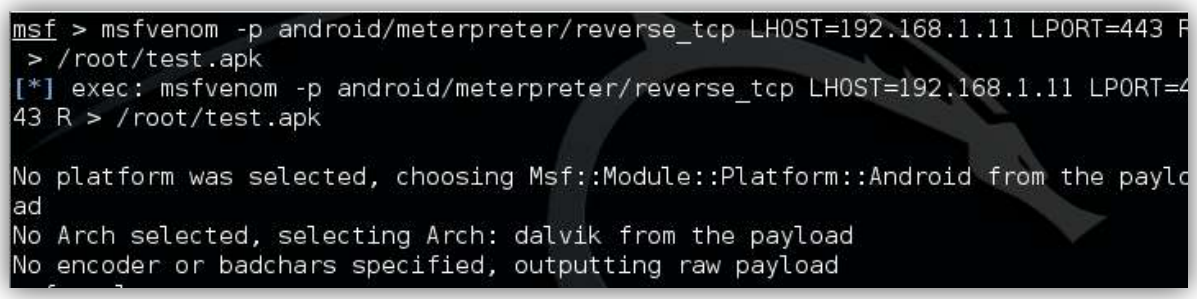
Figura 35. Aplicaciones instaladas en el dispositivo – IP Address

Fuente: Esta investigación

La herramienta que se usará para realizar el ataque es “metasploit”. Un framework que se usa para exponer las vulnerabilidades de muchos sistemas, no sólo Android.

El comando que se ejecutará, generará un APK (Aplicación para Android), y por debajo se le mapeará la dirección IP y el puerto de la máquina atacante, a la que se conectará el dispositivo móvil al abrir el APK, luego de instalado. La dirección IP y el puerto en este caso son 192.168.1.11 y 443 respectivamente.

Figura 36. Generación de una APK maliciosa



Fuente: Esta investigación

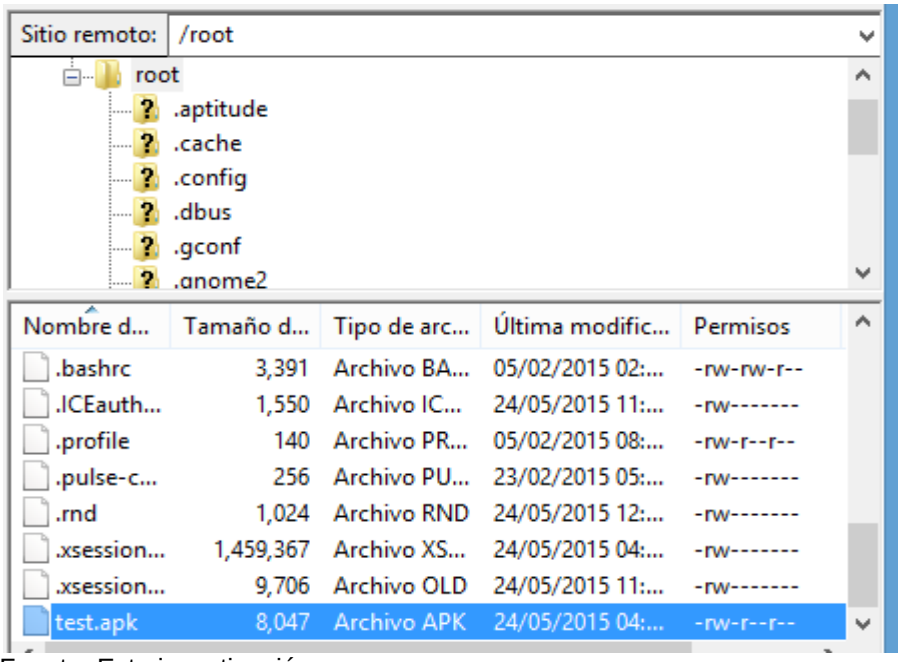
Desde el framework de metasploit se levantará el hilo en la máquina atacante, para que al instalar el APK en el dispositivo, pueda haber una conexión.

Figura 37. Generación de APK para ataque

Fuente: Esta investigación

El APK que se generó en la máquina atacante, se debe distribuir (Internet es una forma fácil de hacerlo – Google Play también). En el caso del ejercicio se instalará directamente en el dispositivo.

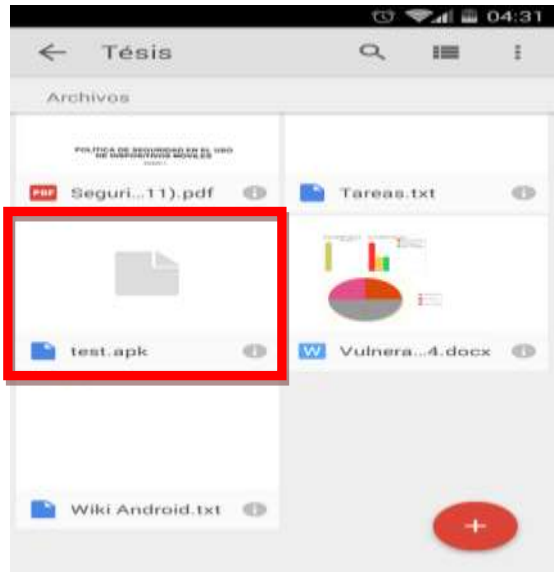
Figura 38.Copiado de la APK en el dispositivo



Fuente: Esta investigación

Se abre el APK desde un explorador de archivos, luego de haberlo copiado al dispositivo:

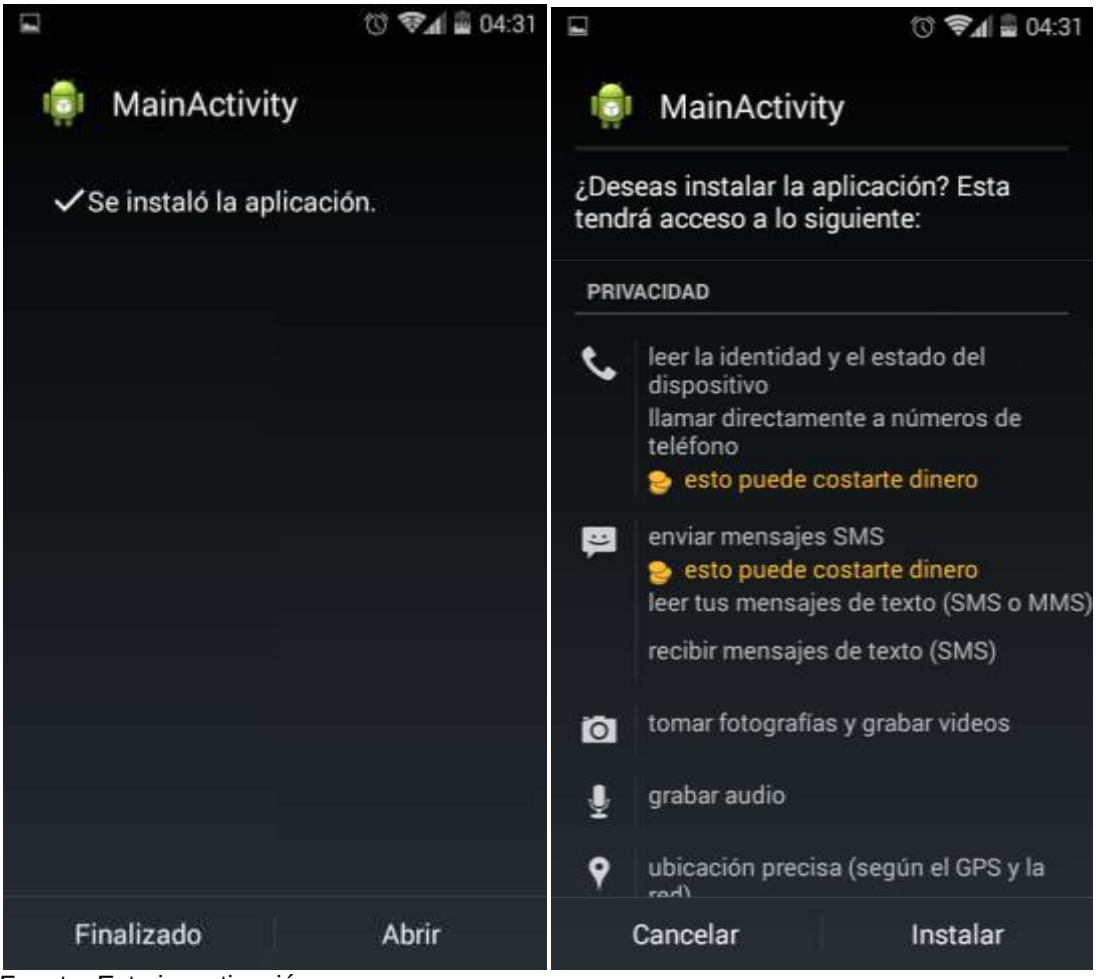
Figura 39. Instalación de la APK en el dispositivo



Fuente: Esta investigación

Aparecerá la advertencia del sistema operativo, en dónde se especifica el nivel de privacidad y de acceso que tendrá la aplicación en el dispositivo. Por lo general nadie lee esto y se pasa por delante, pero no debería ser así. Es muy importante validarlo. Para el caso del ejercicio se debe pisar el botón de instalar.

Figura 40. Instalación de la APK en el dispositivo

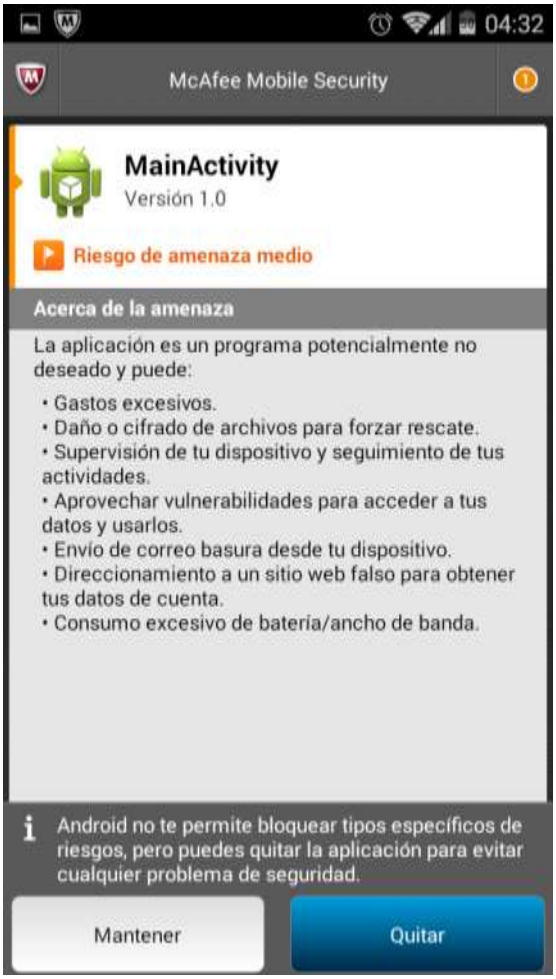


Fuente: Esta investigación

Luego de instalar el APK, aparecerá la confirmación.

El dispositivo al tener un antivirus – antimalware instalado, alertará sobre la presencia extraña de una aplicación. Es algo que suele pasar cuando las aplicaciones son de origen desconocido, aunque en muchas ocasiones son confiables, y por esto muchas personas suelen mantener la aplicación instalada. Pero es un riesgo que no siempre se debe correr, por el contrario debería evitarse.

Figura 41. Alerta del antivirus por aplicaciones maliciosas



Fuente: Esta investigación

Al pisar sobre el botón abrir de la aplicación, inmediatamente se lanza la conexión a la máquina atacante, permitiendo así el total control remoto del dispositivo desde ésta:

Figura 42. Acceso desde la máquina atacante

```
[*] Started reverse handler on 192.168.1.11:443
[*] Starting the payload handler...
[*] Sending stage (43586 bytes) to 192.168.1.14
[*] Meterpreter session 1 opened (192.168.1.11:443 -> 192.168.1.14:57620) at 201
5-05-24 17:32:36 -0400
```

Fuente: Esta investigación

Si se ejecuta el comando “sysinfo” muestra exactamente la versión 4.4.4 que se veía en un principio directamente desde el dispositivo.

Figura 43. Información desde el dispositivo a la máquina atacante

Fuente: Esta investigación

Además teniendo el control del dispositivo, se puede hacer algo tan delicado cómo manipular las cámaras y por medio del comando webcam_snap, se puede tomar una foto, que quedará almacenada directamente en la máquina atacante:

Figura 44. Acceso a la cámara del dispositivo

Fuente: Esta investigación

Figura 45. Imagen tomada

Fuente: Esta investigación

Esto es un tema de conciencia y de saber que existen muchos peligros en el medio, y que se deben preparar para afrontarlos. En un tema más adelante se encontrarán las recomendaciones pertinentes para poder mejorar la seguridad en los Smartphone con Android. Los sistemas nunca serán 100% seguros, pero se pueden realizar labores, que seguro dificultarán el acceso a personas que quieren hacerte daño.

4.8ANÁLISIS DE UNA APLICACIÓN APK

Para este análisis se realizará metodología de ingeniería inversa que permitirá verificar el código de la aplicación y las vulnerabilidades encontradas. Se toma la práctica a partir de un laboratorio práctico y actividad realizado en la materia Dispositivos Móviles de la presente especialización.

> La ingeniería inversa consiste en: “El análisis de un sistema para identificar sus componentes actuales y las dependencias que existen entre ellos para extraer y crear abstracciones de dicho sistema e información de su diseño” (Chifosky, 1990)

Se procederá a descargar una aplicación APK y a través de herramientas como el APKtool y el dex2jar analizar el código fuente y los permisos de dicha aplicación para encontrar vulnerabilidades.

DESCRIPCIÓN:

SNAPDEAL es un sitio de compras de la India, ofreciendo productos como teléfonos móviles, computadores portátiles, televisores, cámaras digitales, ropa, calzado, reloj, etc. Su interfaz es fácil de usar y rápida, la aplicación permite navegar por múltiples productos, ver comentarios, comparar precios, ver las especificaciones y características antes de realizar cualquier compra. Esta aplicación es gratuita, disponible en la tienda Google Play, compatible con Android 2.2 y sistemas operativos superiores, hasta el 4.4. Tiene servicio de entrega gratuito a más de 4000 ciudades de la India.

a) Desarrollador de la aplicación: jasperindiapvtltd@gmail.com
b) Otras APPS desarrolladas: Existe una aplicación del mismo desarrollador Snapdeal Seller Zone, igualmente comercial.

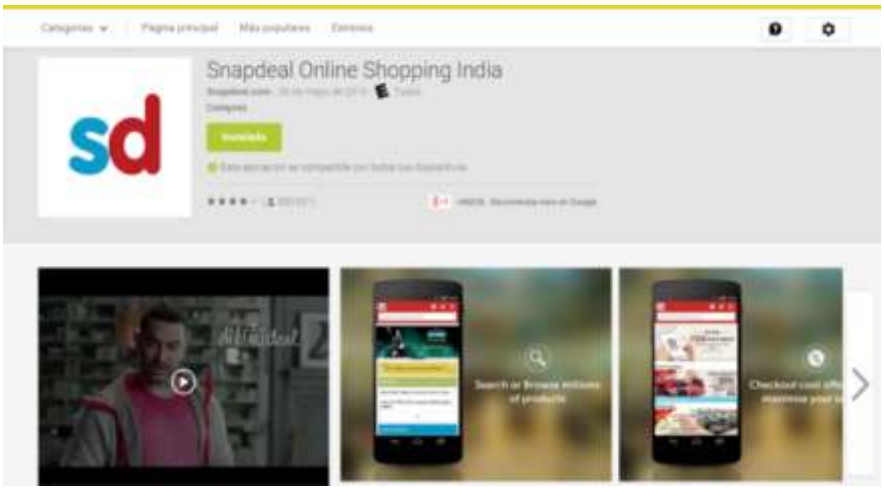
Permisos. En la ficha de la app se observan los permiso que supuestamente solicita, más adelante se comprobara estos requisitos cuando se analice la APK.

La versión 5.2.8b puede acceder a lo siguiente: Dispositivo e historial de aplicaciones. Recuperar aplicaciones en ejecución

Identidad. Buscar cuentas en el dispositivo, Agregar o eliminar cuentas Fotos/Medios/Archivos, Modificar o eliminar el contenido del almacenamiento USB, Probar el acceso al almacenamiento protegido, Información de la conexión Wi-Fi, Ver conexiones Wi-Fi.

Otros. Recibir datos desde Internet, Ver conexiones de red, Acceso completo a la red, Ejecutarse al inicio, Impedir que el dispositivo entre en modo de suspensión, Controlar la vibración

Figura46. Aplicación Snapdeal



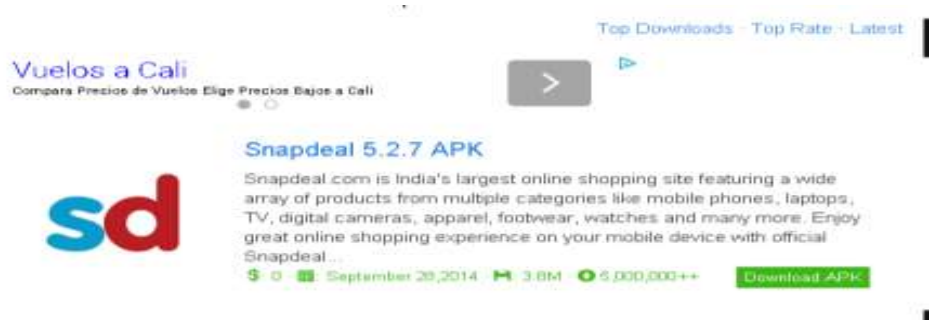
Fuente: Esta investigación

ANÁLISIS ESTÁTICO En esta fase se analizará el código fuente por medio de ingeniería inversa, para ello se realizará las siguientes actividades:

- Análisis de código fuente
- Posibles errores en la lógica de código

Mediante http://directAPKs.com indicar el nombre del paquete de la aplicación que se puede que es la url de Play Store de la app. De esta forma se descarga el APK a analizar.

Figura47. Snapdeal 5.2.7 APK for Android

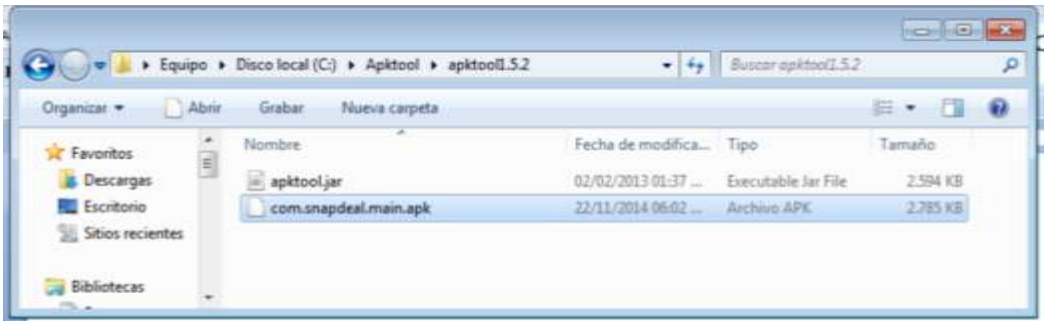


Fuente: Esta investigación

Snapdeal 5.2.7 APK for Android

Free Shopping Apps by Snapdeal.com
Downloads: 5,000,000++ Updated: 2014-09-28
Require: 4.0 and up Content Rating: Low Maturity
License: Free Installed: 0 Size: 3.8M

Figura48. Descarga de la APK para ser analizada



Fuente: Esta investigación

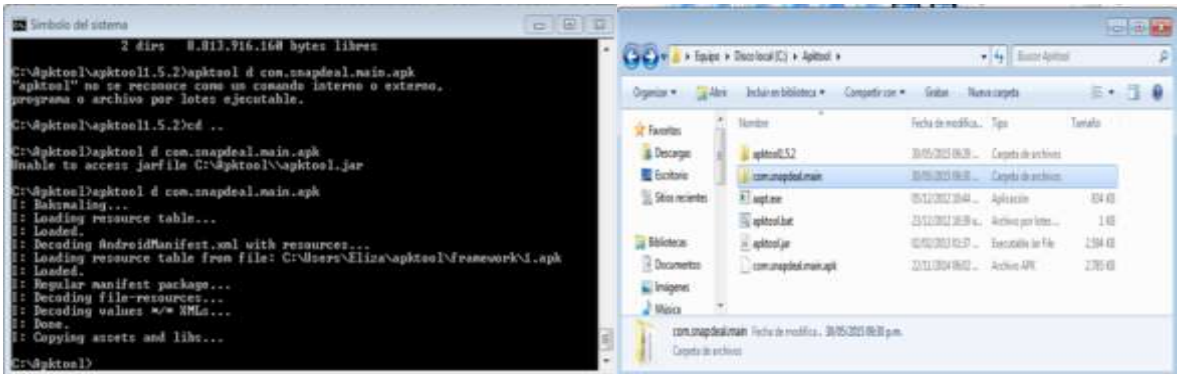
PROCEDIMIENTO: Para decompilar la aplicación se procede a instalar la herramienta APKtool en disco duro, y mediante línea de comandos se ejecuta:

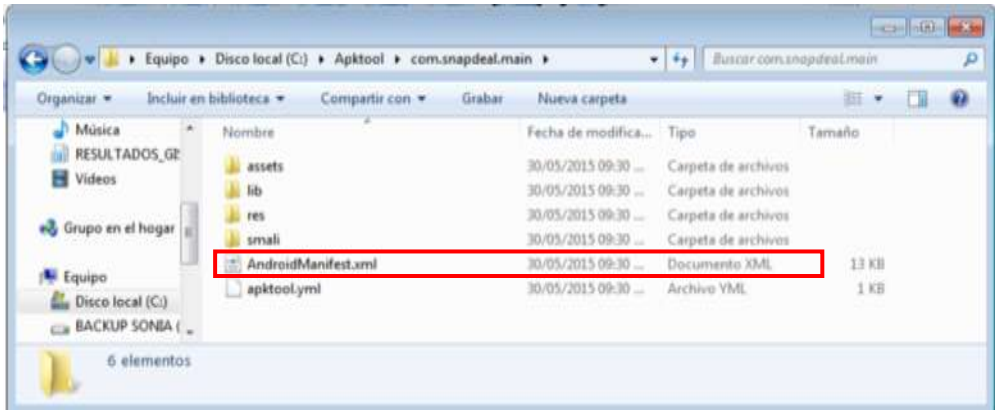
APKtool d com.snapdeal.main.APK

Obtención del fichero de manifiesto

La ejecución del anterior comando crea automáticamente una carpeta llamada com.snapdeal.main dentro de la cual se puede encontrar el archivo de manifiesto **AndroidManifest.xml**

Figura49. Obtención del archivo Android.manifest.xml





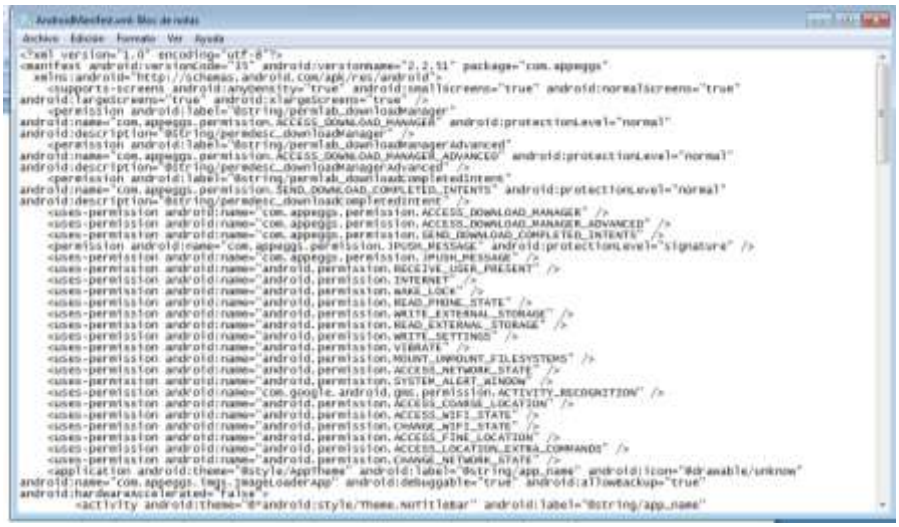
Fuente: Esta investigación

Como se había mencionado anteriormente, el AndroidManifest.xml contiene todos los permisos que la aplicación tiene activados. En la siguiente figura se muestra el contenido del archivo AndroidManifest (Ver anexo 1).

También se obtiene una carpeta con los siguientes directorios:

- **Assets:** Este directorio contiene todo los assets de la aplicación: html, fuentes, etc
- **Red:** Todo lo que va en el directorio de recursos: cadenas, videos, layouts, drawables, etc
- **Smali:** Todo el código fuente en .smali

Figura50. Permisos de la aplicación Snapdeal



Fuente: Esta investigación

Al abrir el archivo AndroidManifest.xml se encuentran los siguientes elementos:
Paquete: “package="com.appeggs"”

En el archivo Xml se declara “Activity” unidades en android encargadas de visualizar interfaces de usuario, lo que indica que tiene interfaz gráfica:

*<activity android:theme="@*android:style/Theme.NoTitleBar"*
android:name=".activity.EggsDetail" android:screenOrientation="portrait" />
<uses-permission
android:name="com.appeggs.permission.ACCESS_DOWNLOAD_MANAGER" />
<uses-permission
android:name="com.appeggs.permission.ACCESS_DOWNLOAD_MANAGER_AD
VANCED" />
<uses-permission
android:name="com.appeggs.permission.SEND_DOWNLOAD_COMPLETED_INT
ENTS" />
<permission android:name="com.appeggs.permission.JPUSH_MESSAGE"
android:protectionLevel="signature" />
<uses-permission
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission
android:name="android.permission.READ_PHONE_STATE" />
<uses-permission
<uses-permission
android:name="android.permission.SYSTEM_ALERT_WINDOW" />

<uses-permission android:name="com.google.android.gms.permission.ACTIVITY_RECOGNITION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS" />
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />

4.9 ANÁLISIS DE CÓDIGO FUENTE

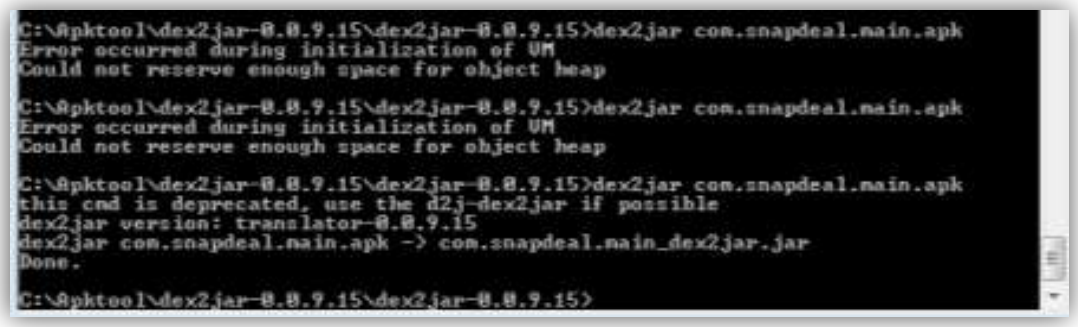
Para ver los ficheros código fuente se utilizará el archivo clases.dex que está dentro del APK.

Para ello se utilizarán las herramientas dex2jary por ventana de comandos se ejecuta:

C:/ dex2jar com.snapdeal.main.APK

Esto lo dejará en la misma carpeta con la extensión .jar

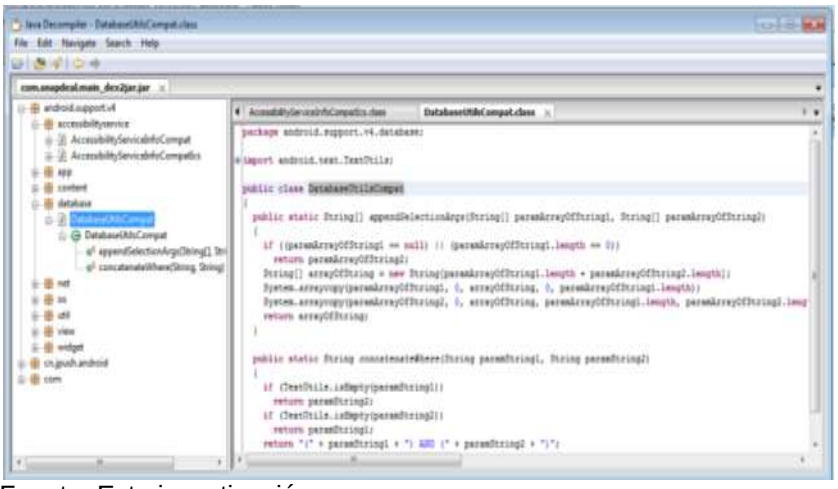
Figura 51. Utilización de la herramienta dex2jar



Fuente: Esta investigación

Ya se tiene el archivo *.jar, así que se puede visualizar mediante la herramienta jd-gui-0.36 que es un visor de este tipo de archivos y que muestra el código fuente de la aplicación con sus diferentes clases.

Figura 52. Código fuente de la aplicación snapdeal

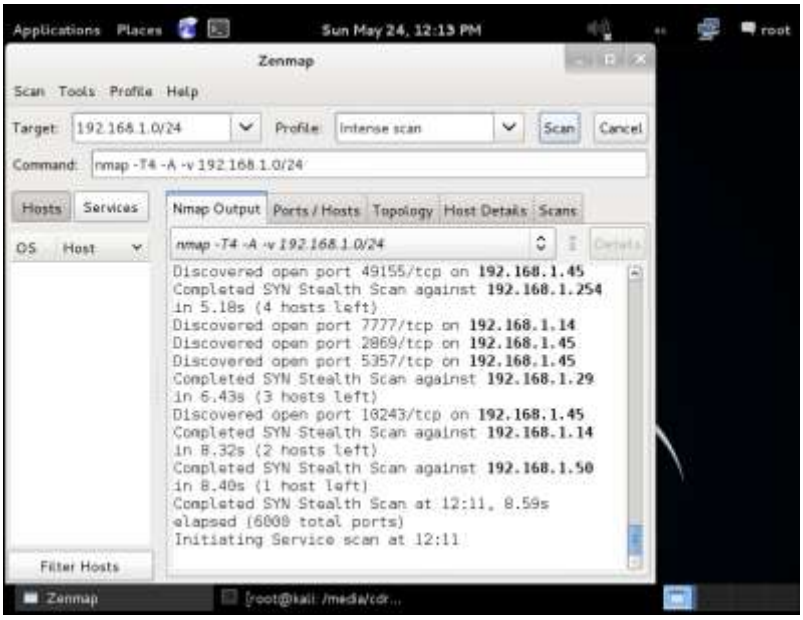


Fuente: Esta investigación

4.10 ANÁLISIS AL SISTEMA OPERATIVO ANDROID CON OPENVAS

Ejecutando el Zenmap en la red. En la simulación del ataque su observó que la IP del dispositivo es la 192.168.1.14:

Figura 53. Análisis con Zenmap

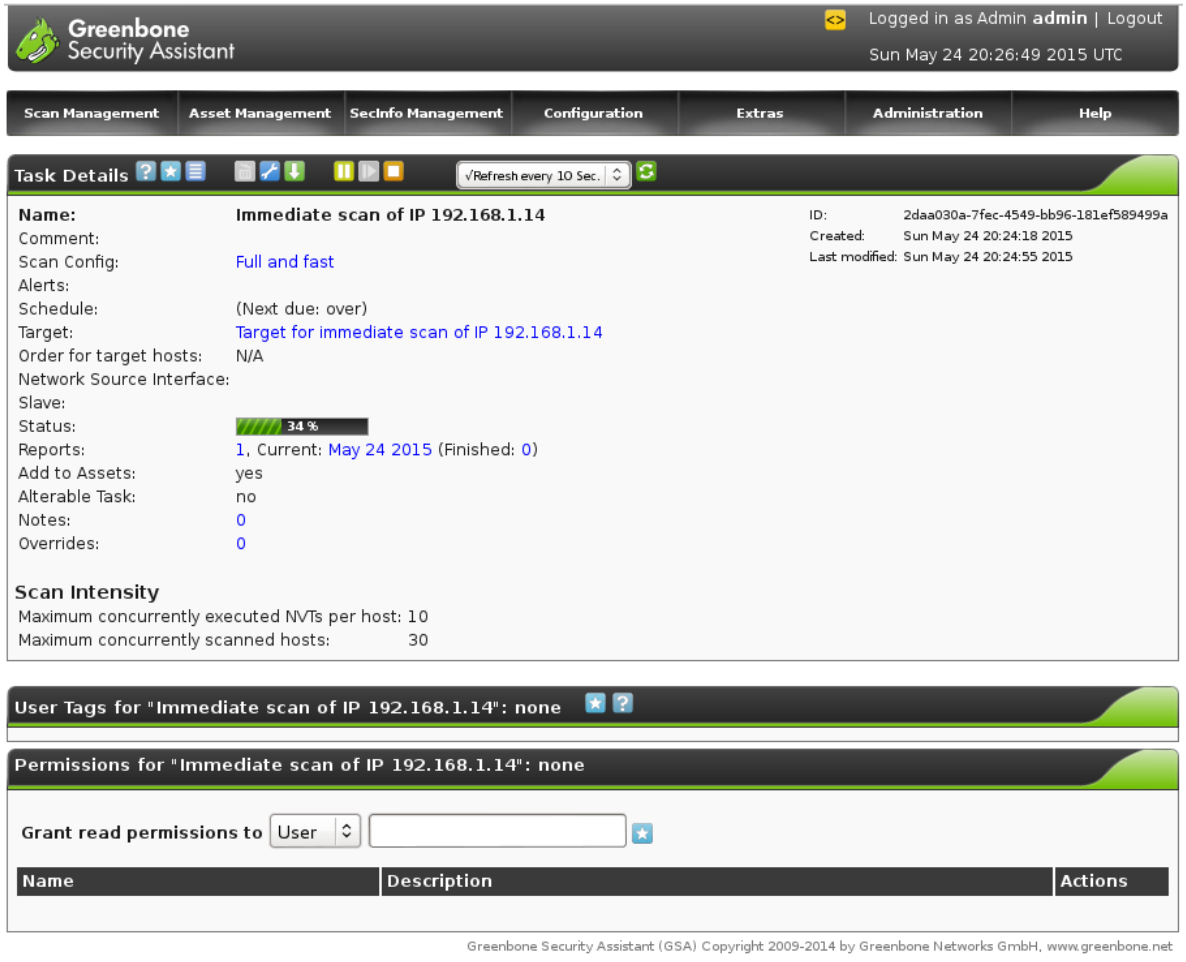


Fuente: Esta investigación

En la imagen se pueden observar algunos puertos abiertos. Se deben validar siempre que puertos se dejan abiertos, porque por medio de estos se podría explotar alguna falla.

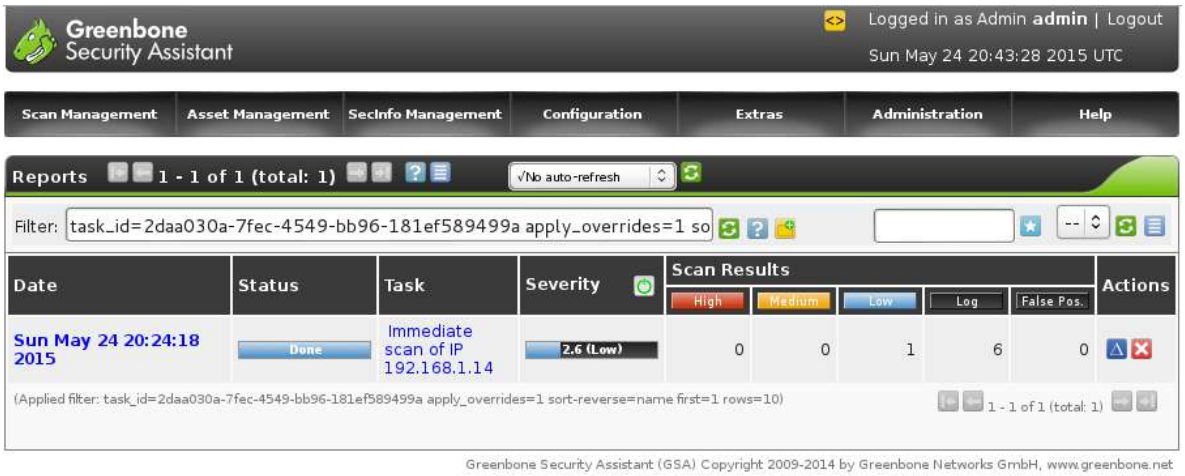
En la siguiente imagen se ejecuta OpenVas para identificar posibles vulnerabilidades en el dispositivo Android.

Figura 54. Análisis con OpenVas



Luego de ejecutado el análisis se abre el reporte a continuación:

Figura 55. Reporte OpenVas



El reporte muestra una vulnerabilidad clasificada como “low”.

Figura 56. Reporte detallado OpenVas

Al validar la descripción esta la explicación:

Figura 57. Vulnerabilidad OpenVas

Es un problema con los timepstamps del protocolo TCP. Al estar habilitados es posible calcular el “uptime” del sistema. Es por esto que recomiendan deshabilitarlos por medio de una variable en el Linux de Android.

5. ANÁLISIS DE RESULTADOS

El presente capítulo establece los resultados y las vulnerabilidades encontradas en las pruebas ejecutadas en el capítulo 5, para establecer que tan seguros son los elementos que maneja el sistema.

Luego de ejecutar las diferentes pruebas para los dispositivos Android 4.4 se llegó a los siguientes resultados:

5.1 RESULTADOS PENT-TEST

Tabla 2. Extracción de archivos por medio del sdcard

PRUEBA EJECUTADA	ANÁLISIS DEL SISTEMA OPERATIVO
TIPO DE PRUEBA	Extracción de datos, almacenamiento interno y/o externo
VULNERABILIDAD	Todos los archivos del sdcard pueden ser leídos por cualquier aplicación
ATAQUE	Robo y alteración de información Dentro del directorio /sdcard se encuentran todos los archivos de documentos, videos, música, etc., que pueden ser extraídos, ya que son públicos
TECNICA PREVENTIVA	El almacenamiento interno es mejor cuando se quiere estar seguro de que ni el usuario ni otras aplicaciones pueden tener acceso a sus archivos, utilizar programas para cifrar archivos

Fuente: Esta investigación

Tabla 3. Extracción de archivos de paquetes

PRUEBA EJECUTADA	ANÁLISIS DEL SISTEMA OPERATIVO
TIPO DE PRUEBA	Localización de archivos, almacenamiento inseguro
VULNERABILIDAD	Todos los paquetes de aplicaciones se encuentran en la ruta /data/data/nombre.del.paquete/ donde se encuentran los siguientes directorios Cache Database Libs Shareprefs
ATAQUES	Dentro del directorio /data/data/nombre_del paquete se encuentran todos los datos del paquete

PRUEBA EJECUTADA	ANÁLISIS DEL SISTEMA OPERATIVO
	como bases de datos, preferencias y archivos que mediante código pueden ser alterados como shareprefs (preferencias de la aplicación) Otra de las cosas que se pueden realizar es desbloquear el patrón de bloqueo, conectando el teléfono ya que la contraseña se almacena en /data/system con el nombre de pasword.key o gesture.key.
TECNICA PREVENTIVA	Comprobar que las aplicaciones están firmadas digitalmente, autenticación de credenciales

Fuente: Esta investigación

Tabla 4. Extracción de archivos de bases de datos

PRUEBA EJECUTADA	ANALISIS DEL SISTEMA OPERATIVO
TIPO DE PRUEBA	Localización de archivos, almacenamiento inseguro
VULNERABILIDAD	Todos los paquetes de aplicaciones se encuentran en la ruta /data/data/nombre.del.paquete/ donde se encuentran los siguientes directorios Cache Database Libs Shareprefs En el directorio Database se pueden extraer datos de las bases de datos de cada aplicación y hacer un sql injection
ATAQUES	Dentro del directorio /data/data/nombre_del paquete se encuentran todos los datos del paquete como bases de datos, archivos con extensión db, que pueden ser alterados mediante técnicas de sql injection, robo y/o alteración de información
TECNICA PREVENTIVA	Comprobar que las aplicaciones están firmadas digitalmente y no guardar datos sensibles como direcciones , teléfonos, números de cuenta dentro de una base de datos

Fuente: Esta investigación

Tabla 5. Análisis de Trafico

PRUEBA EJECUTADA	ANÁLISIS DEL SISTEMA OPERATIVO
TIPO DE PRUEBA	Intento de acceso al celular para capturar paquetes de datos
VULNERABILIDAD	Aunque se comprobó que todos los puertos estaban cerrados, se intentó una segunda opción basada en el ataque ARP-Spoofing que detectó el dispositivo móvil y realizó el ataque redirigiendo los paquetes hacia el equipo atacante, e inclusive denegando el servicio de acceso a internet.
ATAQUES	ARP-Spoofing, DNS Spoofing, Web Spoofing
TÉCNICA PREVENTIVA	Conectarse redes seguras, tener ips dinámicas, evitar utilizar protocolos de transmisión insegura como http (texto en claro), Para estar seguros de realizar transmisiones seguras con información sensible se debe emplear HTTPS, que es HTTP más SSL/TLS para añadir cifrado al canal. El protocolo HTTPS se dirige siempre al puerto 443. El autenticado y cifrado se realiza a nivel de socket con la clase SSLSocket. Es muy recomendado emplear SSLSocket para mitigar riegos al emplear redes públicas (muy común en los usuarios).

Fuente: Esta investigación

5.2 RESULTADOS ATAQUE A UN DISPOSITIVO ANDROID CON METASPLOIT FRAMEWORK

Tabla 6. Prueba de penetración a dispositivo móvil

PRUEBA EJECUTADA	PRUEBA DE PENETRACIÓN
TIPO DE PRUEBA	Creación e instalación de una APK maliciosa en el móvil para acceder remotamente a él y alterar el comportamiento de la cámara
VULNERABILIDAD	Es una prueba peligrosa, ya que al acceder al dispositivo de la víctima se puede robar y alterar la información, el comportamiento de los dispositivos e ingresar al Shell Linux para extraer ficheros, vulnerabilidad en cuanto a puertos abiertos,
ATAQUES	Metasploit Framework

PRUEBA EJECUTADA	PRUEBA DE PENETRACIÓN
TÉCNICA PREVENTIVA	Emplear un programa de cifrado de archivos, guardar información sensible en sitio seguro, realizar copia de seguridad de los datos, no abrir correos electrónicos de los cuales se desconozca el remitente, verificar bien antes de instalar cualquier aplicación

Fuente: Esta investigación

5.3 ANÁLISIS DE CÓDIGO

Tabla 7. Análisis del archivo Android.manifest.xml

PRUEBA EJECUTADA	**ANÁLISIS DEL ARCHIVO ANDROID MANIFEST**
TIPO DE PRUEBA	Se analizaron los permisos existentes en el archivo Android manifest, encontrando los siguientes permisos más relevantes en la aplicación:
VULNERABILIDAD	Permisos de la aplicación que si llegaren a ejecutarse podrían alterar el comportamiento del móvil, estos permisos son: - android.permission.**INTERNET**: Permite a las aplicaciones abrir sockets de red. - android.permission.**ACCESS_WIFI_STATE**: Permite que las aplicaciones accedan a información sobre redes inalámbricas. - android.permission.**ACCESS_COURSE_LOCATION**: Permite que una aplicación acceda a la ubicación aproximada derivado de las fuentes de ubicación de red, tales como torres de telefonía móvil y Wi-Fi. - android.permission.**ACCESS_FINE_LOCATION**: Permite que una aplicación acceda a la ubicación precisa de fuentes de localización como GPS, antenas de telefonía móvil y Wi-Fi. - android.permission.**READ_PHONE_STATE**: Permite acceso al estado del teléfono - android.permission.**ACCESS_NETWORK_STATE**: Permite que las aplicaciones accedan a información sobre redes. La seguridad de la intercomunicación entre componentes puede verse afectada debido a la falta de permisos y

PRUEBA EJECUTADA	ANÁLISIS DEL ARCHIVO ANDROID MANIFEST
	declaraciones públicas de los componentes. Si el valor "exported" es verdadero esto permite a cualquier componente comunicarse. (Anexo B) ***<receiver android:name="com.inmobi.commons.analytics.androidsdk.IMAdTrackerReceiver" android:enabled="true" android:exported="true">*** Los proveedores de contenido son vulnerables a sql injection y revelan información sensible. (Ver anexo B) ***<provider android:name="com.appeggs.downloads.DownloadProvider" android:authorities="com.appeggs.downloads" />***
ATAQUES	Si se analiza detenidamente todos los procesos, se puede hacer casi cualquier cosa con estos permisos, y sobre todo de manera remota, ya que se tiene permiso para abrir socket y cambiar configuración de sistema. Se puede hacer llamadas a números de tarificación especial, obtención de fotografías en vivo, venta de contactos a terceros, grabar las conversaciones, localizar a la persona por su posición GPS y permitir el acceso al estado del teléfono.
TECNICA PREVENTIVA	Emplear un programa de cifrado de archivos, guardar información sensible en sitio seguro, realizar copia de seguridad de los archivos, utilizar conexiones seguras, verificar los permisos que pide una aplicación antes de instalarse.

Fuente: Esta investigación

Tabla 8. Análisis de código de la aplicación snapdeal.com

PRUEBA EJECUTADA	ANALISIS DEL CODIGO Y LAS CLASES
TIPO DE PRUEBA	Análisis de código fuente de la aplicación
VULNERABILIDADES ENCONTRADAS	

VULNERABILIDAD	CLASE	EJEMPLO DE CODIGO

openFileInput: método que permite leer ficheros de la memoria interna.	Paquete:cn.jpush (android
	
	c.class
	
	pushservice.class
	
	j.class
		private static JSONArray d(Context paramContext)
		
		{
		
		if (!new File(paramContext.getFilesDir(), z[7]).exists())
		
		return null;
		
		JSONArray localJSONArray;
		
		try
		
		{
		
		FileInputStream localFileInputStream = paramContext.**openFileInput**(z[7]);
		
		byte[] arrayOfByte = new byte[1024];
		
		StringBuffer localStringBuffer = new StringBuffer();

Webview: clase que permite visualizar una Web dentro de una aplicación Android y por tanto la ejecución de código javascript, posible ejecución de sql injection	AddtrackerWebView Loader.class	public void onReceivedError**(WebView** paramWebView, int paramInt, String paramString1, String paramString2)
		
		{
		
		try
		
		{
		
		if (AdTrackerWebViewLoader.b().com

VULNERABILIDAD	CLASE	EJEMPLO DE CODIGO

		pareAndSet(true, false))
		
		AdTrackerNetworkInterface.setWebviewUploadStatus(false);
		synchronized (AdTrackerNetworkInterface.getNetworkThread())
		{
		AdTrackerNetworkInterface.getNetworkThread().notify();
		
		AdTrackerUtils.reportMetric(AdTrackerEventType.GOAL_FAILURE, AdTrackerWebViewLoader.b(AdTrackerWebViewLoader.this), 0, 0L, paramInt, null);
		Log.internal("[InMobi]-[AdTracker]-4.4.3", "Webview Received Error");
		super.onReceivedError(paramWebView, paramInt, paramString1, paramString2);
		return;
		}
		catch (Exception localException)
		{
		while (true)
		Log.internal("[InMobi]-[AdTracker]-4.4.3", "Exception onReceived Error", localException);
		}
		}

VULNERABILIDAD	CLASE	EJEMPLO DE CODIGO

Función que pueden ser usada inseguramente: putString ya que:La información privada del usuario entra en el programa.
 	
2. Los datos se escriben en una ubicación externa, como la consola, el sistema de archivos o la red.
	m.class	public final void a(boolean paramBoolean, String paramString)
		
		{
		
		Message localMessage = this.a.obtainMessage(this.b);
		
		Bundle localBundle = new Bundle();
		
		localBundle.putBoolean(z[1], paramBoolean);
		
		localBundle.**putString(z[2],** paramString);
		
		localBundle.**putString**(z[0], this.c.c);
		
		localMessage.setData(localBundle);
		
		localMessage.sendToTarget();
		
		}

TECNICA PREVENTIVA	Aunque pueden ser falsos positivos, los códigos deben ser revisados de tal manera que no se afecte el uso de la aplicación y que no sean peligros potenciales para el usuario que instalen la aplicación, ya que al acceder al código se pueden manipular varios elementos que pueden acceder a la información del dispositivo donde se instala.	

6. POLÍTICAS DE SEGURIDAD

Buscando proteger de mejor manera los dispositivos móviles, se deben de tener en cuenta las siguientes recomendaciones:

- Usar métodos de bloqueo en el dispositivo, ojalá que sean múltiples para hacer más difícil el acceso a éste. Existen métodos como contraseñas, patrones, bloqueos biométricos, de reconocimiento de voz, entre otros.

- Realizar respaldo periódico de los elementos importantes almacenados en el dispositivo. Existen herramientas de backup en la nube (No local) que permiten gestionar los respaldos y mantenerlos seguros, por si se requiere alguna recuperación.

- Procurar aplicar cifrado a la memoria de almacenamiento, para hacer imposible la copia de los datos y por lo tanto la extracción de estos.

- Usar algún método de borrado remoto, que puede configurarse a través de múltiple métodos. Por medio de Google Sync o Google Apps es posible realizar esta función. De esta manera si se pierde o es robado el dispositivo se puede eliminar la información, para evitar su divulgación o manipulación.

- Instalar antivirus en los dispositivos, que apoyará no solo el tema de infecciones, sino que podrá controla el acceso no autorizado al dispositivo, haciendo las veces de firewall y además contienen algunas otras funcionalidades como capturecam, por si se intenta desbloquear el dispositivo de forma errónea en múltiples ocasiones , tomará una foto para conocer quién trata de desbloquearlo. Los antivirus tienen múltiples funciones y a nivel de seguridad es muy importante contar con ellas.

- Se debe tratar de evitar conexiones a redes públicas y otras de dudosa procedencia. Esto incluye acceso a Wi-Fi pero también por medio de Bluetooth e infrarrojo. Es posible que a través este tipo de conexiones, pueda ser robada información del dispositivo.

- Evitar instalar aplicaciones desconocidas, de fuentes no identificadas, incluso si provienen de Google Play. La mayoría de ellas contienen malware y elementos que infectan el dispositivo, incluso permitiendo el control remoto, cómo se pudo evidenciar en un tema expuesto anteriormente.

- Las actualizaciones de las aplicaciones y del sistema operativo son fundamentales para proteger los dispositivos, pues en su mayoría traen

parches de seguridad de bugs (Fallas) encontradas durante su operación y por esto siempre se deben de instalar.

- Procurar por no conectar los dispositivos vía USB en equipos públicos y desconocidos, pues también se pueden tener brechas de seguridad al realizar esto.

- Es importante leer manuales e instruirse en cuanto al uso del dispositivo y del sistema operativo. En muchas ocasiones se comenten errores y se dejan de habilitar funciones de seguridad por la falta de conocimiento.

- Hacer lo posible por evitar la instalación de ROMS (Imágenes del sistema operativo) custom o personalizados. Estos por lo general no tienen los niveles óptimos de seguridad y pueden contener elementos maliciosos que pueden generar problemas en el dispositivo, así como el robo de información.

- Si un usuario es inexperto en el manejo de Android, se recomienda evitar hacer “root” en el sistema operativo para obtener permisos de súper usuario. Muchas aplicaciones maliciosas se aprovechan de esto para hacer daños.

7. GESTIÓN DEL PROYECTO

7.1 CRONOGRAMA DE ACTIVIDADES

El proyecto de investigación se desarrolló en un tiempo de 3 meses de acuerdo al siguiente cronograma de actividades:

ACTIVIDAD	MES 1				MES 2				MES 3			
	1	2	3	4	1	2	3	4	1	2	3	4
Etapa 1: Recolección de información												
Etapa 2: Conocimiento del SO Android												
Etapa 3: Practica para establecer y analizar las vulnerabilidades encontradas en el sistema operativo Android 4.4												
Etapa 4: Documento final con resultados												

7.2 RECURSOS

Para el desarrollo del proyecto se contó con los siguientes recursos:

Lógicos

- Internet
- Material bibliográfico
- Simulador del sistema operativo Android para diferentes pruebas para Smartphones

Humanos

- Responsables del proyecto: Oscar Betancur, Sonia Eraso
- Asesor del proyecto Ing. Deivis Ramírez

Físicos

- Equipos de cómputo
- Dispositivo móvil con sistema operativo Android 4.4

Financieros

ACTIVIDAD	CANTIDAD	COSTO
Costo mensual de internet 3 Mb para investigaciones	4 meses $ 55.000	$ 220.000
Celular Smartphone Android 4.4	1 por integrante c/u $300.000	$ 600.000
Plan de datos para celular x 4 meses	4 meses ($ 160.000) Por 2 integrantes	$ 320.000
Computador por cada integrante para pruebas	$ 1.500.000.	$ 3.000.000
TOTAL		$ 4.140.000

8. CONCLUSIONES

- Al analizar la versión 4.4 de Android se logró identificar que el sistema operativo tiene vulnerabilidades y estas normalmente van siendo corregidas por el fabricante (Google), pero se debe tener precaución en el manejo de los datos y la información que se almacena en los dispositivos para evitar posible daño y/o pérdida de la información.

- Las vulnerabilidades que se presentan en los dispositivos móviles con Android, son en su mayoría por la falta de conocimiento de los usuarios y por la poca precaución que tienen al instalar aplicaciones.

- Existen numerosas herramientas para el análisis del sistema operativo Android, como las vistas en el presente documento, son herramientas útiles tanto para técnicas de pent-testing como análisis forense que permiten analizar los datos almacenados en el dispositivo, buscar vulnerabilidades, pero mal utilizadas permiten igualmente modificar o dañar la información del dispositivo.

- Se ha cumplido con los objetivos planteados en el presente documento, encontrando los principales componentes del sistema operativo Android y analizando vulnerabilidades en el mismo, lo que ha permitido concluir que todo sistema por más avanzado que esté presenta vulnerabilidades sean pocas o muchas y que como usuarios y especialistas en seguridad informática se deben tomar medidas preventivas y sensibilizar a las personas en nuestro entorno para ser consciente sobre el uso seguro para este tipo de dispositivos.

9. RECOMENDACIONES

- Los desarrolladores de aplicaciones móviles deben de ser conscientes de la creación de código seguro, basarse en proyectos como OWASP, OSSTM, pretenden a determinar y combatir las causas de software inseguro y crear aplicaciones cada vez más confiables.

- Es importante seguir unas políticas de seguridad referentes al manejo del sistema operativo Android 4.4, que sean accesibles y aplicables por parte del usuario, sensibilizando acerca de las medidas preventivas que se necesita para tener una información que aplique los principios de disponibilidad, confiabilidad e integridad.

- Como recomendación final es importante para los especialistas en seguridad informática indagar cada vez más sobre las vulnerabilidades no solo en este tipo de sistemas operativos sino en todos aquellos que están en nuestro entorno, cada día hay aplicaciones muy llamativas usadas por miles de usuarios, pero así mismo aparecen malware, ataques y todo tipo de acción que intenta alterar la información que se maneja.

REFERENCIAS BIBLIOGRÁFICAS

ANDROID OS (2012). HISTORIA DE ANDROID. [Citado: Mayo 28 de 2015]. Disponible en: http://androidos.readthedocs.org/en/latest/data/historia/

CARACTERÍSTICAS DEL SISTEMA OPERATIVO ANDROID (1995). [Citado Mayo 25 de 2015] Disponible en: http://www.samsung.com/co/article/android-2-2-os-explained/

DRAKE, Joshua; FORA; Pau Oliva; ZA Lanier; COLLIN, Mulliner. Etal. Android Hackers's handbook. Indianapolis: John Wiley & Sons, Inc., 2013, 577 p.

EL ANDROID LIBRE. *LA HISTORIA DE ANDROID EN IMÁGENES: DESDE SUS INICIOS HASTA AHORA.*[Citado Abril 10 de 2015].Disponible en: http://www.elandroidelibre.com/2014/06/la-historia-de-android-en-imagenes-desde-sus-inicios-hasta-ahora.html

EVOLUCIÓN DE MÓVILES[Citado: Mayo de 2014] Disponible en:*http://dispositivosmobilesits.blogspot.com/2012/02/evolucion-de-moviles.html* http://www.hacking-tutorial.com/hacking-tutorial/hacking-android-smartphone-tutorial-using-metasploit/#sthash.9RNHAwR0.dpbs

FERNANDES, Jerónimo. Banco de pruebas de seguridad para plataformas móviles Android. Cartagena, 2014. Tesis de Grado: Universidad Politécnica de Cartagena.

GIRONES, Jesús Tomas. El gran libro de Android. México: Alfaomega, 2012. 403p.

INSTITUTO NACIONAL DE TECNOLOGÍAS DE LA COMUNICACIÓN INTECO. (2008). [Citado: Febrero de 2015] *Estudio sobre seguridad en dispositivos móviles y Smartphones.* Disponible en: https://www.inteco.es/file/rdj_9ad_DHM_jZcyjTYRlw

INSTITUTO NACIONAL DE TECNOLOGÍAS DE LA COMUNICACIÓN INTECO. (2011).*Estudio sobre hábitos seguros en el uso de Smartphones por los niños y adolescentes españoles.* Disponible en: http://www.inteco.es/Estudios/Estudio_smartphones_menores

LABORATORIO DE REDES Y SEGURIDAD. *Ataques a dispositivos móviles*. Recuperado de:

http://redyseguridad.fip.unam.mx/proyectos/buenaspracticas/ataquesadispositivosmoviles.html

LA HISTORIA DE ANDROID. [Citado: Mayo 28 de 2015]. Disponible en: http://www.android.com/intl/es-419_mx/history/

NORMA TECNICA COLOMBIANA NTC 1486. Documentación, presentación de tesis, trabajos de grado y otros trabajos de investigación. Bogotá: Icontec.

SANTANA, A. (Enero de 2011). "*Una infraestructura de comunicaciones cliente-servidor para dispositivos móviles*". Disponible en: http://tesis.bnct.ipn.mx/dspace/bitstream/123456789/9891/1/210.pdf

TODO ANDROID (2015).*Qué es Android 5.0 Lollipop: novedades y características de esta versión de Android*. Disponible en: http://www.todoandroid.es/index.php/faq-de-android/65-versiones/1672-que-es-android-5-0-lollipop-novedades-y-caracteristicas-de-esta-version-de-android.html

ANEXOS

ANEXO A

Dispositivos donde se puede instalar Android 4.4
Fuente: http://www.movilcelular.es/sistemas-operativos/android-442-kitkat/113

Tabletas y Smartphones con Android 4.4 KitKat 2015

Tabletas y Smartphones con Android 4.4 KitKat 2014

 Samsung Galaxy Grand Prime Duos SM-G530H	 Samsung Galaxy Grand Prime SM-G530M	 Samsung Galaxy Young 2 Duos SM-G130H	 Samsung Galaxy Young 2 Duos SM-G130M	 ZTE Grand X Max Z787
 ZTE Kis 3 Max	 LG Tribute LS660	 LG G3 Beat Dual D724	 LG G3 S D722	 LG G3 Vigor D725
 Positivo S480	 Micromax A310 Canvas Nitro	 Bq Aquaris E6	 HTC Desire 510	 HTC Desire 510 Boost Mobile
 HTC Desire 510 Sprint	 NGM Dynamic Jump	 Samsung Galaxy V Duos SM-G313HZ	 LG L Prime Dual TV D337	 Alcatel One Touch Fierce 2 7040T
 Xolo Q2100	 Lenovo Vibe Z2 Pro K920	 Sony Xperia E3 3G	 Sony Xperia E3 4G LTE	 Sony Xperia E3 Dual Sim
 Sony Xperia E3 TV	 Sony Xperia M2 Aqua	 ZTE ZMax Z970	 Blu Studio 5.0 CE D536	 Blu Studio C mini D670
 Xiaomi Redmi Note 4G	 Sharp Aquos Crystal 305SH	 Sharp Aquos Crystal 306SH	 LG G Vista VS880	 LG G2 Lite Dual D295F
 LG L Bello Dual D335	 LG L Fino D290N	 LG L Fino Dual D295F	 LG L60 Dual X147	 LG L60 X145
Blu Life Play 2 L170	Blu Studio 5.0 C HD D534	Blu Studio 5.0 K D530	Sony Xperia C3 4G LTE	Sony Xperia C3 Dual Sim
Bq Aquaris E5 FHD	Huawei Ascend P7	Huawei Ascend P7 Dual Sim	Kyocera Brigadier E6782	NGM Forward Xtreme
LG G Pad 10.1 WiFi V700	LG G Pad 7.0 WiFi V400	LG G Pad 8.0 WiFi V480	Samsung Galaxy Avant SM-G386T	Samsung Galaxy Avant SM-G386T1
Samsung Galaxy S5 Duos SM-G900MD	Samsung Galaxy S5 mini 3G SM-G800H	Samsung Galaxy S5 mini Duos SM-G800H	Samsung Galaxy S5 mini LTE SM-G800F	Samsung Galaxy S5 mini LTE SM-G800Y
Samsung Galaxy Tab S 8.4 4G SM-T705 16GB	Samsung Galaxy Tab S 8.4 4G SM-T705 32GB	Samsung Galaxy Tab S 8.4 WiFi SM-T700 16GB	Samsung Galaxy Tab S 8.4 WiFi SM-T700 32GB	LG L20 D100

Dispositivos que pueden actualizar a Android 4.4

- Google Nexus 4
- Google Nexus 7 2012
- Google Nexus 7 2013
- Google Nexus 10 2012
- Samsung Galaxy S4
- Samsung Galaxy S4 Active
- Samsung Galaxy S4 Mini
- Samsung Galaxy Note 2
- Samsung Galaxy Note 3
- Samsung Galaxy Note 8.0
- Samsung Galaxy Tab 3 (todas las versiones)
- Samsung Galaxy Note 10.1 Edición 2014
- Sony Xperia Z
- Sony Xperia ZL
- Sony Xperia ZR
- Sony Xperia Z1
- Sony Xperia Z Ultra
- Sony Xperia Z Tablet
- LG G2
- LG T Pad 8,3
- LG T Pro
- LG VU 3
- LG Optimus 4X HD
- HTC One
- HTC One Max
- HTC One X
- Asus Fonepad Notes 6
- Asus Fonepad 7
- Asus Padfone Infinity
- Oppo Find 5
- Oppo N1
- Huawei Ascend P6
- Huawei Ascend G700
- Motorola Moto X
- Motorola Moto G

ANEXO B.

Contenido del Archivo AndroidManifest.xml de la aplicación Snapdeal

ANDROID MANIFEST.XML

```
<?xml version="1.0" encoding="utf-8"?>
<manifest android:versionCode="35" android:versionName="2.2.51" package="com.appeggs"
  xmlns:android="http://schemas.android.com/APK/res/android">
<supports-screens android:anyDensity="true" android:smallScreens="true" android:normalScreens="true"
android:largeScreens="true" android:xlargeScreens="true" />
<permission android:label="@string/permlab_downloadManager"
android:name="com.appeggs.permission.ACCESS_DOWNLOAD_MANAGER" android:protectionLevel="normal"
android:description="@string/permdesc_downloadManager" />
<permission android:label="@string/permlab_downloadManagerAdvanced"
android:name="com.appeggs.permission.ACCESS_DOWNLOAD_MANAGER_ADVANCED" android:protectionLevel="normal"
android:description="@string/permdesc_downloadManagerAdvanced" />
<permission android:label="@string/permlab_downloadCompletedIntent"
android:name="com.appeggs.permission.SEND_DOWNLOAD_COMPLETED_INTENTS" android:protectionLevel="normal"
android:description="@string/permdesc_downloadCompletedIntent" />
<uses-permission android:name="com.appeggs.permission.ACCESS_DOWNLOAD_MANAGER" />
<uses-permission android:name="com.appeggs.permission.ACCESS_DOWNLOAD_MANAGER_ADVANCED" />
<uses-permission android:name="com.appeggs.permission.SEND_DOWNLOAD_COMPLETED_INTENTS" />
<permission android:name="com.appeggs.permission.JPUSH_MESSAGE" android:protectionLevel="signature" />
<uses-permission android:name="com.appeggs.permission.JPUSH_MESSAGE" />
<uses-permission android:name="android.permission.RECEIVE_USER_PRESENT" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_SETTINGS" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />
<uses-permission android:name="com.google.android.gms.permission.ACTIVITY_RECOGNITION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS" />
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
<application android:theme="@style/AppTheme" android:label="@string/app_name" android:icon="@drawable/unknow"
android:name="com.appeggs.imgs.ImageLoaderApp" android:debuggable="true" android:allowBackup="true"
android:hardwareAccelerated="false">
<activity android:theme="@*android:style/Theme.NoTitleBar" android:label="@string/app_name"
android:name="com.appeggs.activity.HomeActivity" android:screenOrientation="portrait">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<activity android:theme="@*android:style/Theme.NoTitleBar" android:name=".activity.EggsDetail"
android:screenOrientation="portrait" />
<activity android:theme="@*android:style/Theme.NoTitleBar" android:name=".activity.WallPaperJpushActivity"
android:screenOrientation="portrait" />
<activity android:theme="@*android:style/Theme.NoTitleBar" android:name=".activity.GagHotAvtivity"
android:screenOrientation="portrait" />
<activity android:theme="@*android:style/Theme.NoTitleBar" android:name=".activity.Wallpaper_FunnyGagActivity"
android:screenOrientation="portrait" />
<activity android:theme="@*android:style/Theme.NoTitleBar" android:name=".activity.Gag_Activity"
android:screenOrientation="portrait" />
<activity android:theme="@*android:style/Theme.NoTitleBar" android:name=".activity.Gag_detail_Webview"
android:screenOrientation="portrait" />
<activity android:theme="@*android:style/Theme.NoTitleBar" android:name=".activity.Detail_Search_webview_Activity"
android:screenOrientation="portrait" />
<activity android:theme="@*android:style/Theme.NoTitleBar" android:name=".activity.SettingActivity"
android:screenOrientation="portrait" />
<activity android:theme="@*android:style/Theme.NoTitleBar" android:name=".activity.CasualGamesActivity"
android:screenOrientation="portrait" />
<activity android:theme="@*android:style/Theme.NoTitleBar" android:name=".activity.TopImage"
android:screenOrientation="portrait" />
<activity android:theme="@*android:style/Theme.NoTitleBar" android:name=".activity.SearchActivity"
android:screenOrientation="portrait" />
```

```
<activity android:theme="@*android:style/Theme.NoTitleBar" android:name=".activity.CategoriesActivity"
android:screenOrientation="portrait" />
<activity android:theme="@*android:style/Theme.NoTitleBar" android:name=".activity.BestChoiceActivity"
android:screenOrientation="portrait" />
<activity android:theme="@*android:style/Theme.NoTitleBar" android:name=".activity.ImagersSecetor"
android:screenOrientation="portrait" />
<activity android:theme="@*android:style/Theme.NoTitleBar" android:name=".activity.DisplayActivity"
android:screenOrientation="portrait" />
<activity android:theme="@*android:style/Theme.NoTitleBar" android:name="com.appeggs.fragment.app_SelectFragment"
android:screenOrientation="portrait" />
<activity android:theme="@*android:style/Theme.NoTitleBar" android:name="com.appeggs.fragment.Must_have_apps_Fragment"
android:screenOrientation="portrait" />
<activity android:theme="@*android:style/Theme.NoTitleBar" android:name="com.appeggs.downloadsui.DownloadList"
android:screenOrientation="portrait" />
<activity android:name="com.appeggs.activity.ShowActivity" />
<activity android:theme="@*android:style/Theme.NoTitleBar" android:name="com.appeggs.activity.LiveWallPaperActivity"
android:screenOrientation="portrait" />
<activity android:theme="@*android:style/Theme.NoTitleBar"
android:name="com.appeggs.activity.LiveWallPaperCategoriesActivity" android:screenOrientation="portrait" />
<activity android:theme="@*android:style/Theme.NoTitleBar" android:name="com.appeggs.activity.WallPaperCategoriesActivity"
android:screenOrientation="portrait" />
<activity android:theme="@*android:style/Theme.NoTitleBar" android:name="com.appeggs.activity.WallPaperActivity"
android:screenOrientation="portrait" />
<activity android:theme="@*android:style/Theme.NoTitleBar" android:name="com.appeggs.activity.AdultActivity"
android:screenOrientation="portrait" />
<activity android:theme="@*android:style/Theme.NoTitleBar" android:name="com.appeggs.activity.TopImageResultActivity"
android:screenOrientation="portrait" />
<activity android:theme="@*android:style/Theme.NoTitleBar" android:name="com.appeggs.activity.Must_Have_apps_Result"
android:screenOrientation="portrait" />
<activity android:theme="@*android:style/Theme.NoTitleBar" android:name="com.appeggs.activity.BestPicksActivity"
android:screenOrientation="portrait" />
<activity android:theme="@*android:style/Theme.NoTitleBar" android:name="com.appeggs.activity.WebViewActivity"
android:screenOrientation="portrait" />
<activity android:name="com.google.ads.AdActivity"
android:configChanges="keyboard|keyboardHidden|orientation|screenLayout|uiMode|screenSize|smallestScreenSize" />
<activity android:name="com.inmobi.androidsdk.IMBrowserActivity"
android:configChanges="keyboard|keyboardHidden|orientation|screenLayout|uiMode|screenSize|smallestScreenSize"
android:hardwareAccelerated="true" />
<receiver android:name="com.inmobi.commons.analytics.androidsdk.IMAdTrackerReceiver" android:enabled="true"
android:exported="true">
<intent-filter>
<action android:name="com.android.vending.INSTALL_REFERRER" />
</intent-filter>
</receiver>
<service android:name="com.inmobi.commons.internal.ActivityRecognitionManager" android:enabled="true" />
<provider android:name="com.appeggs.downloads.DownloadProvider" android:authorities="com.appeggs.downloads" />
<service android:name="com.appeggs.downloads.DownloadService" />
<receiver android:name="com.appeggs.downloads.DownloadReceiver" android:exported="false">
<intent-filter>
<action android:name="android.intent.action.BOOT_COMPLETED" />
<action android:name="android.net.conn.CONNECTIVITY_CHANGE" />
</intent-filter>
</receiver>
<meta-data android:name="UMENG_APPKEY" android:value="537aa5d456240b8368034a44" />
<meta-data android:name="UMENG_CHANNEL" android:value="AppEggsMarket 20140520" />
<activity android:name="com.appeggs.activity.TestActivity">
<intent-filter>
<action android:name="jpush.testAction" />
<category android:name="jpush.testCategory" />
</intent-filter>
</activity>
<activity android:theme="@*android:style/Theme.Translucent.NoTitleBar" android:name="cn.jpush.android.ui.PushActivity"
android:configChanges="keyboardHidden|orientation">
<intent-filter>
<action android:name="cn.jpush.android.ui.PushActivity" />
<category android:name="android.intent.category.DEFAULT" />
<category android:name="com.appeggs" />
</intent-filter>
</activity>
<service android:name="cn.jpush.android.service.DownloadService" android:enabled="true" android:exported="false" />
<service android:name="cn.jpush.android.service.PushService" android:enabled="true" android:exported="false">
<intent-filter>
<action android:name="cn.jpush.android.intent.REGISTER" />
<action android:name="cn.jpush.android.intent.REPORT" />
<action android:name="cn.jpush.android.intent.PushService" />
<action android:name="cn.jpush.android.intent.PUSH_TIME" />
</intent-filter>
```

```
</service>
<receiver android:name="cn.jpush.android.service.PushReceiver" android:enabled="true">
<intent-filter android:priority="1000">
<action android:name="cn.jpush.android.intent.NOTIFICATION_RECEIVED_PROXY" />
<category android:name="com.appeggs" />
</intent-filter>
<intent-filter>
<action android:name="android.intent.action.USER_PRESENT" />
<action android:name="android.net.conn.CONNECTIVITY_CHANGE" />
</intent-filter>
<intent-filter>
<action android:name="android.intent.action.PACKAGE_ADDED" />
<action android:name="android.intent.action.PACKAGE_REMOVED" />
<data android:scheme="package" />
</intent-filter>
</receiver>
<receiver android:name="cn.jpush.android.service.AlarmReceiver" />
<receiver android:name="com.appeggs.activity.MyReceiver" android:enabled="true">
<intent-filter>
<action android:name="cn.jpush.android.intent.REGISTRATION" />
<action android:name="cn.jpush.android.intent.UNREGISTRATION" />
<action android:name="cn.jpush.android.intent.MESSAGE_RECEIVED" />
<action android:name="cn.jpush.android.intent.NOTIFICATION_RECEIVED" />
<action android:name="cn.jpush.android.intent.NOTIFICATION_OPENED" />
<action android:name="cn.jpush.android.intent.ACTION_RICHPUSH_CALLBACK" />
<category android:name="com.appeggs" />
</intent-filter>
</receiver>
<meta-data android:name="JPUSH_CHANNEL" android:value="developer-default" />
<meta-data android:name="JPUSH_APPKEY" android:value="a1f881c2ff3074aba6ae6268" />
</application>
</manifest>
```