1

Seguridad en sistemas y servicios móviles

ÍNDICE

MOTIVACIÓN

Los primeros terminales móviles funcionaban mediante software propietario, de tal manera que cada uno de ellos tenía su propio sistema de operar, ofreciendo, eso sí, una misma interfaz con el operador público de telefonía.

Esto ha sido así cuando la capacidad de las redes se limitaba al transporte de señal de voz, sin tener en cuenta los datos. Experiencias como WAP permitieron introducir la capacidad de ejecutar pequeños programas independientemente de la plataforma donde se ejecutaban, usando para ello el JAVA.

Ahora lo que nos encontramos es con una disociación entre software y hardware, donde los fabricantes de software son diferentes a los de hardware, y han dado lugar a auténticos Sistemas Operativos. Esto es lo que se denominan ecosistemas, donde APP, hardware y S.O. conviven en una única experiencia de usuario.

Si quieres entender los Sistemas Operativos actualmente existentes para plataformas móviles en toda su extensión, esta unidad te va a permitir iniciarte en este tipo de temas.

PROPÓSITOS

Al finalizar el estudio de esta unidad deberías ser capaz de poder explicar las siguientes cuestiones:

- Tener claro la evolución seguida con los sistemas operativos existentes para plataformas móviles.
- Conocer cómo funcionan los sistemas operativos móviles más importantes.
- Conocer los modelos de seguridad que se implementan en estos Sistemas Operativos.
- Tener criterio suficiente para elegir la plataforma más segura, o por lo menos poder valorar los riesgos existentes.

Pero sobre todo, como se verán en los diversos temas, la Seguridad en las Comunicaciones móviles es un aspecto esencial, con grandes expectativas de negocio para las mafias implicadas.

PREPARACIÓN PARA LA UNIDAD

En esta unidad vamos a tratar los siguientes temas:

1. Veremos los principales aspectos e introduciremos el Sistema Operativo Android, así como el modelo de Seguridad que implementa
2. Veremos los principales aspectos e introduciremos el Sistema Operativo iOS, así como el modelo de Seguridad que implementa
3. Veremos los principales aspectos e introduciremos el Sistema Operativo BlackBerry OS, así como el modelo de Seguridad que implementa
4. Por último haremos una comparativa de los modelos de seguridad entre las tres principales plataformas existentes: Android, IOS y BlackBerry OS.

1. SISTEMAS OPERATIVOS MÓVILES

Atrás han quedado los días en que los teléfonos móviles eran sólo un dispositivo para hacer llamadas y enviar mensajes de texto ocasionales. Los smartphones modernos están más cerca de las computadoras de mano y nos permiten enviar correos electrónicos, jugar, ver las noticias y hacer video llamadas, etc. Por tanto un sistema operativo móvil (OS) es un software que permite a los teléfonos inteligentes, tablet PCs y otros dispositivos ejecutar aplicaciones y programas.

Llamamos sistemas operativos al software que se ejecuta en nuestros PC de escritorio y portátiles que permiten administrar sus recursos y la memoria. Desde hace ya algún tiempo, los teléfonos inteligentes han utilizado sistemas operativos siguiendo el mismo camino que los PC y es este desarrollo ha llevado a disponer de las mismas funciones avanzadas en los móviles que antes sólo estaban disponibles en nuestros ordenadores. Un sistema operativo móvil normalmente se inicia cuando un dispositivo enciende, presentando una pantalla con iconos que presentan información y proporcionan acceso a las aplicaciones. Los sistemas operativos móviles también gestionan la conectividad de la red celular o Wi-Fi, así como el acceso telefónico.

También es una plataforma para que los desarrolladores puedan crear aplicaciones o 'apps' (programas de software desarrolladas para teléfonos inteligentes que pueden llevar a cabo funciones específicas). Hay cientos de miles de aplicaciones disponibles y constantemente se están desarrollando cada una con su propio propósito.

Por ejemplo, se puede descargar una aplicación de tiempo que indica la temperatura actual o de las posibilidades de lluvia en tu ciudad, una aplicación de noticias o widget que envía las últimas noticias directamente a la pantalla de inicio del dispositivo, o un juego para simplemente pasar el tiempo.

La mayoría de los sistemas operativos móviles están ligados a un hardware específico, con poca flexibilidad. Los usuarios sin embargo pueden hacer lo que se conoce como jailbreak o subir permisos (root) en algunos dispositivos, lo que les

permitirá instalar otro sistema operativo móvil o desbloquear aplicaciones restringidas.

Ejemplos de sistemas operativos de dispositivos móviles incluyen Apple iOS, Google Android, Research in Motion BlackBerry OS, Symbian de Nokia, webOS de Hewlett-Packard (anteriormente Palm OS) y Sistema operativo Windows Phone de Microsoft. Algunos, como los de Microsoft Windows 8, funcionan tanto como un sistema operativo de escritorio tradicional y un sistema operativo móvil.

1.1. OS MÓVILES

Debido a que los sistemas operativos de teléfonos inteligentes están tan integrados con la apariencia, sensación y la función de un teléfono móvil, muchas personas basan su elección del dispositivo según el sistema operativo que utiliza. Cualquier teléfono inteligente muestra el nombre y la versión de su software en el menú de configuración.

Algunos sistemas operativos móviles son software de código abierto, lo que significa que no hay restricciones en lo que se puede descargar en él, o que puede desarrollar su software (a menudo hay una "comunidad" de los desarrolladores). Los sistemas operativos de código abierto son totalmente personalizables, mientras que otros están restringidos en los tipos de software autorizadas para funcionar en el dispositivo.

1.1.1. IOS

El multi-touch de Apple iOS, el sistema operativo multitarea es lo que corre el iPhone, iPad y iPod de Apple. Una versión especial del software también se utiliza en el reloj de Apple. iOS responde al tacto del usuario lo que le permite pulsar en la pantalla para abrir un programa, juntar los dedos para minimizar o ampliar una imagen, o pasar el dedo por la pantalla para cambiar de página.

Sin embargo con el iOS de Apple no se permite al usuario utilizarlo en sistemas de terceros, por lo que sólo será capaz de utilizarlo en productos fabricados por Apple. Viene con el navegador web Safari para el uso de Internet, una aplicación de iPod para reproducir música y correo de Apple para gestionar sus correos electrónicos.

Existen multitud de aplicaciones (APP) disponibles, que pueden descargarse en la App Store directamente a cualquier dispositivo con iOS, ya sea un iPhone o un iPad. Estos abarcan de todo, desde recetas de cocina hasta tutoriales para aprender a tocar la guitarra o trucos para juegos.

1.1.2. Android

Android OS es propiedad de Google y funciona con el kernel de Linux, el cual se puede encontrar en una amplia gama de dispositivos. Android es un sistema operativo de código abierto que permite a los desarrolladores acceder a hardware desbloqueado y desarrollar nuevos programas como deseen. Esto significa acceso ilimitado a cualquier persona que quiera desarrollar aplicaciones para el teléfono con muy pocas restricciones en cuanto a temas de uso y licencia, lo que permite que los usuarios se beneficien de una gran cantidad de contenidos libres.

Android es actualmente la plataforma de teléfonos inteligentes dominante debido a su tremenda atracción y con un amplio espectro de usuarios.

Algunas de las mejores características de Android incluyen la posibilidad de personalizar múltiples pantallas de inicio con widgets y aplicaciones para poder disponer de acceso más fácil al contenido y funciones que más importa. También cuenta con una excelente capacidad para tareas múltiples - con la capacidad de cerrar los programas con sólo enviar a la basura.

Por último, pero no menos importante, el Android Market, que es el equivalente para Android del App Store de Apple es el hogar de millones de aplicaciones, muchas de las cuales son completamente gratis.

1.1.3. Teléfono de Windows

Microsoft lanzó una versión renovada de su enorme plataforma de Windows para móviles a finales de 2010, después que su software cayera detrás de iOS y Android como OS móvil. Rediseñado y reconstruido desde cero pone un mayor énfasis en la experiencia del usuario, y el resultado fue un sistema operativo llamado Windows Phone.

Windows Phone es reconocible desde su interfaz basada en “azulejos” - conocido como Metro - que cuenta con secciones desmontables e intercambiables en la pantalla principal, cada uno con su propio propósito y función.

También tiene agregadores llamados 'hubs', que agrupan todas las fotos de todas las aplicaciones, o toda la música en una biblioteca, es decir, las fotos de Facebook se pueden encontrar con las fotos de la cámara y los documentos de diferentes fuentes agrupadas en un único lugar lo que permite un fácil acceso localización.

Windows Phone viene con una versión optimizada para móviles de Internet Explorer para acceder a la web, y Exchange, que soporta cuentas de correos electrónicos corporativos seguras.

1.2. APLICACIONES (APP)

Se nos ofrecen tres principales enfoques de desarrollo. Podemos crear:

- Aplicaciones Móviles Nativas (iPhone, Android, BlackBerry, Windows Mobile)
- Aplicaciones Web para móviles (Web Apps)
- Aplicaciones Híbridas (Multiplataforma / Crossplatform)

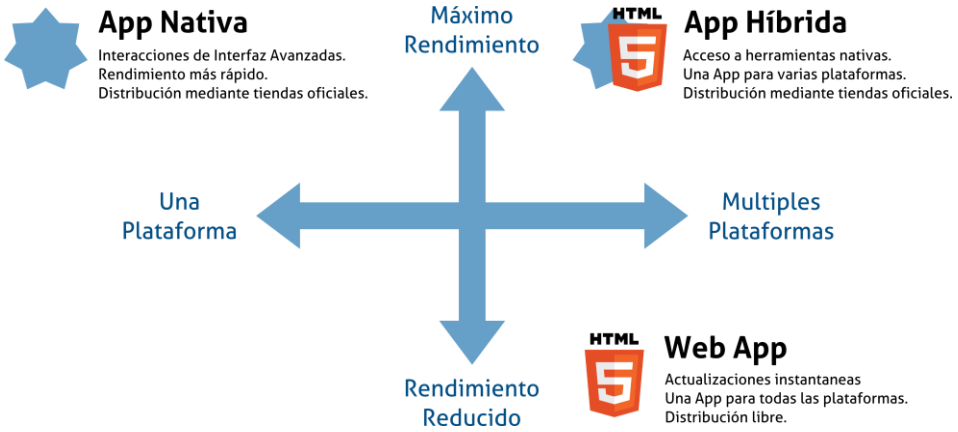

Figura 1: Aplicaciones web, nativas o híbridas. Fuente: http://www.contunegocio.es

Una aplicación "nativa", es una aplicación móvil desarrollada en el lenguaje específico para esa plataforma. En otras palabras, si deseamos que nuestra aplicación funcione en el iPhone, Android y BlackBerry, tenemos que desarrollar la misma aplicación en tres versiones distintas, una para cada plataforma.

Cada versión de esta aplicación se distribuye/comercializa a través de las tiendas online correspondientes: Apple Store (iOS), Google Play (Android), AppWorld (BlackBerry) y Windows Marketplace (Windows Mobile). Después de su descarga al móvil o tablet, esta aplicación se instala en el sistema de archivos de cada dispositivo.

El desarrollo de Aplicaciones Web (Web App) para dispositivos móviles es el desarrollo de páginas web que son optimizadas para ser visualizadas en las pantallas de dispositivos móviles y para ser utilizadas en pantallas táctiles.

El usuario accede a estas aplicaciones mediante el navegador web de su dispositivo, aunque en función del dispositivo puede crear un enlace directo en su escritorio y acceder a ella como si se tratara de cualquier otra aplicación. Ejemplos de Aplicaciones web móviles: Financial Times (web app con caché local), Yahoo Mail.

Figura 2: Comparativa App web, nativas v.s híbridas. Fuente: Desarrollo de Apps nativas multiplataforma - Slideshare

Una aplicación híbrida o multiplataforma, como su nombre bien indica, es una "mezcla entre una aplicación Nativa y una WebApp". Este tipo de aplicaciones se hicieron populares gracias al framework Phonegap (hoy Apache Cordova) pero existen varios que pueden ser utilizados: Kendo UI Mobile, Sencha Touch, Trigger.io o Titanium Appcelerator.

Al desarrollar aplicaciones híbridas utilizamos la tecnología nativa (conjunto de APIs) cuando es necesario o más nos conviene (para acceder a cámara, acelerómetro, contactos, etc.), y la tecnología web (como HTML5, CSS3 y JavaScript) para el desarrollo de la estructura e interfaz de la aplicación. De este modo maximizamos la base de código que es común a las distintas plataformas y limitamos el desarrollo de funcionalidad nativa a aquellos aspectos que no puedan ser desarrollados de otro modo.

Ejemplos de aplicaciones híbridas son:

- Instagram: Utiliza tecnología nativa para tomar, editar y publicar las fotos (incluso sin conexión a internet) y la tecnología web para desplegar las fotos y el perfil. Esto permite a los desarrolladores mejorar la lista de fotografías sin la necesidad de publicar una nueva versión. Si no hay conexión a internet la fotografía se está en espera para ser subida una vez haya la conexión.

- Linked In: Híbrida para iOS y nativa para Android.

- Facebook: Ha cambiado, de una aplicación totalmente híbrida a una nativa con funcionalidades híbridas.

La solución híbrida que pretende aprovechar las capacidades de las aplicaciones nativas permitiendo la re-utilización de la mayor parte del código para todas las plataformas.

2. INTRODUCCIÓN AL SISTEMA OPERATIVO ANDROID

2.1. INTRODUCCIÓN A LA PLATAFORMA

Android es una plataforma móvil creada por Google y el Open Handset Alliance. Se basa en un núcleo de Linux y normalmente se utiliza el lenguaje de programación Java para su desarrollo.

La. Open Handset Alliance (OHA) es una alianza comercial de 84 compañías que se dedica a desarrollar estándares abiertos para dispositivos móviles. Algunos de sus miembros son Google, HTC, Dell, Intel, Motorola, Qualcomm, Texas Instruments, Samsung, LG, T-Mobile, Nvidia y Wind River Systems.

Android proporciona un conjunto importante de abstracciones para los desarrolladores de aplicaciones, entre las que se incluyen interfaces de usuario, el ciclo de vida de la aplicación, eficientes mecanismos de IPC, permisos, etc. La plataforma también proporciona aplicaciones clave del sistema, como pueden ser el teclado numérico, la agenda telefónica, la pantalla de inicio, así como herramientas de

desarrollo y depuración para la integración con Eclipse. Todo esto basado en un modelo tradicional de seguridad Linux.

La plataforma Android se considera “abierta”, este es un concepto de amplia interpretación. Muchas empresas afirman que su patente e incluso su software de código cerrado es "abierto" simplemente, porque su API es de carácter público.

La plataforma Android también se considera abierta porque los desarrolladores pueden ver y modificar su código fuente sin licencias restrictivas. También se considera software libre, debido a que está diseñado para adaptar los niveles de seguridad y es capaz de ejecutar aplicaciones de terceros.

Otras plataformas con modelos de seguridad más débiles, en ocasiones necesitan disimular su debilidad a base de limitar el uso de las aplicaciones sólo a los "buenos conocidos”, por lo que levantan barreras para el desarrollo de aplicaciones y obstaculizan el libre acceso de los usuario.

La Open Handset Alliance lleva el concepto de software libre más allá, e incluso afirma que Android no hace diferencia entre aplicaciones “core” del teléfono y aplicaciones de terceros.

Esto podría ser cierto desde la perspectiva de Android, pero probablemente no desde la de un tercero que intenta cambiar el menú de configuración del sistema o el controlador de Wi-Fi en los terminales de los usuarios.

Android permite a los desarrolladores crear distribuciones empleando cualquier tipo de permiso que se considere oportuno, siempre y cuando sea un desarrollo propio. Sin embargo, en caso de que los desarrolladores deseen utilizar una distribución creada por Android, deberán adaptarse a los permisos establecidos en esa distribución. Es por eso, por lo que Android se considera una plataforma “igualitaria”.

Las distribuciones Android se pueden configurar para que sus propietarios no tengan acceso a la raíz, o no puedan cambiar ciertos aspectos del sistema o de su configuración. Hasta el momento, esta característica es común en todas las

plataformas, y puede ayudar a que los usuarios se sientan más seguros si pierden su teléfono, o instalan programas o aplicaciones de terceros.

Este aspecto también ayuda a tranquilizar a las empresas desarrolladoras de contenidos con licencia, como pueden ser tonos de llamada, juegos, etc. ya que dicha licencia se encuentra más "protegida".

Sin embargo, en cualquier terminal, una persona con los conocimientos técnicos necesarios y con acceso físico al dispositivo, probablemente pueda acceder a cualquiera de estas configuraciones, simplemente con un poco de tiempo y esfuerzo. Por lo tanto no se pueden considerar como fallos de seguridad de Android.

Sin embargo, los usuarios de terminales Android esperan tener un control total sobre sus dispositivos, y es probable que la gente sensata que accede a la raíz de su terminal no dañe el modelo de seguridad de Android, aunque habrá otros muchos que si lo hagan.

Los dispositivos con software libre como Android, no son los más indicados para controlar este tipo de accesos nos deseados o no controlados.

Uno de los aspectos más singulares de la plataforma Android es que los desarrolladores tienen la posibilidad de contribuir directamente a su desarrollo futuro. Si usted tiene algunas ideas inteligentes de cómo hacer mejorar la plataforma, puede agruparlas en un parche y presentarlo para su inclusión en la próxima versión de Android. Este proceso es similar a la forma en que otros proyectos de código abierto aceptan contribuciones y apoyos de la comunidad de desarrolladores.

2.2. MODELO DE SEGURIDAD DE ANDROID

2.2.1. ARQUITECTURA DE ANDROID:

A continuación se muestran las diferentes capas en las que se divide la arquitectura de Android, desde el nivel más bajo, que es el Kernel de Linux hasta el nivel

más alto donde se encuentran aplicaciones tales como el navegador, la agenda, etc.

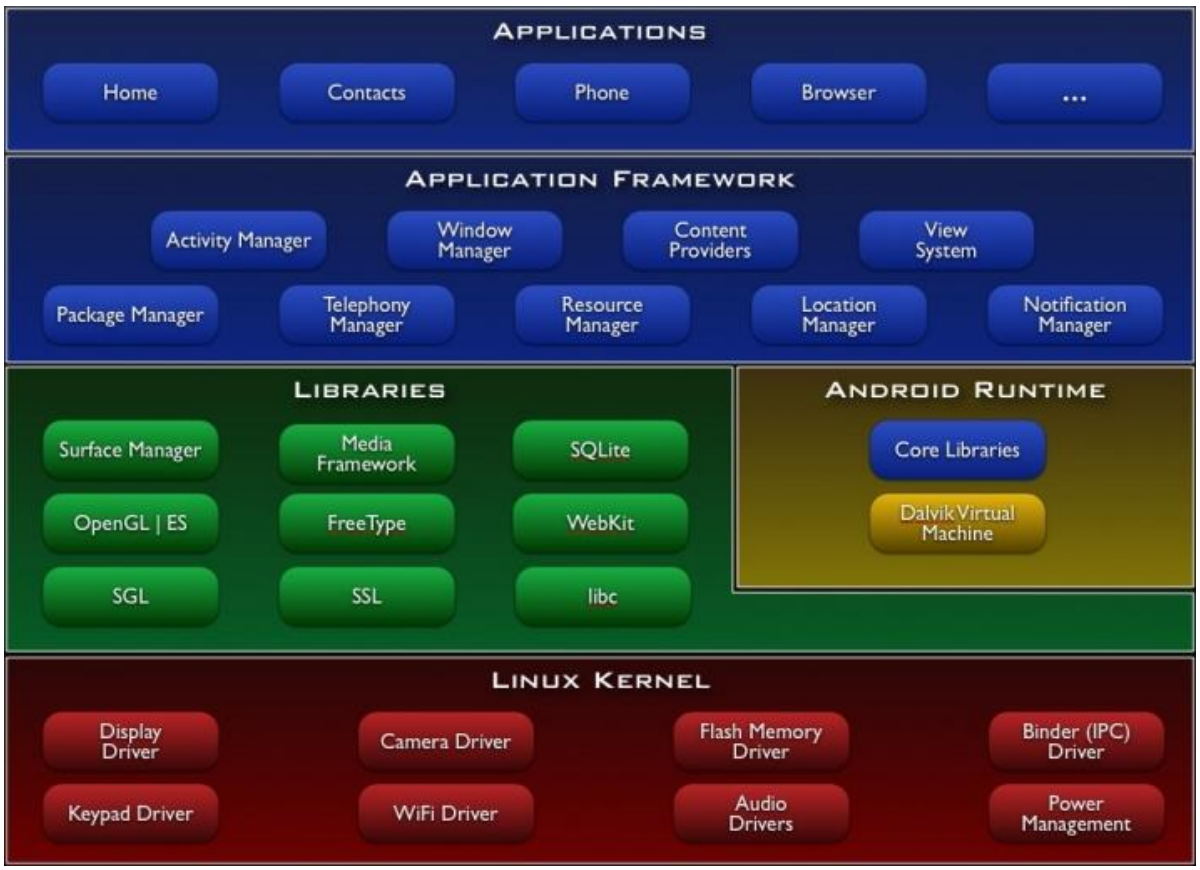

Figura 3: This image is reproduced from work created and HYPERLINK "http://code.google.com/policies.html" shared by the Android Open Source Project and used according to terms described in the HYPERLINK "http://creativecommons.org/licenses/by/2.5/" Creative Commons 2.5 Attribution License

2.2.2. KERNEL DE LINUX:

El núcleo del sistema operativo Android está basado en el kernel de Linux versión 2.6, similar al que puede incluir cualquier distribución de Linux, como Ubuntu, solo que adaptado a las características del hardware en el que se ejecutará Android, es decir, para dispositivos móviles.

El núcleo actúa como una capa de abstracción entre el hardware y el resto de las capas de la arquitectura. El desarrollador no accede directamente a esta capa, sino que debe utilizar las librerías disponibles en capas superiores. De esta forma también nos evitamos el hecho de quebrarnos la cabeza para conocer las características precisas de cada teléfono. Si necesitamos hacer uso de la cámara, el sistema operativo se encarga de utilizar la que incluya el teléfono, sea cual sea. Para cada elemento de hardware del teléfono existe un controlador (o driver) dentro del kernel que permite utilizarlo desde el software.

El kernel también se encarga de gestionar los diferentes recursos del teléfono (energía, memoria, etc.) y del sistema operativo en sí: procesos, elementos de comunicación (networking), etc.

Una de las características más interesantes a nivel de seguridad del Kernel de Linux que monta Android es que UID y GID son distintos para cada aplicación. UID y GID son los identificadores de usuario y grupo para ficheros/procesos en ejecución.

Figura 4: This image is reproduced from work created and HYPERLINK "http://code.google.com/policies.html" shared by the Android Open Source Project and used according to terms described in the HYPERLINK "http://creativecommons.org/licenses/by/2.5/" Creative Commons 2.5 Attribution License.

Ejecución de procesos en Sandbox. (Falta explicar qué es sandboxing…)

2.2.3. MIDDLEWARE (ANDROID RUNTIME):

Librerías de Android:

La siguiente capa que se sitúa justo sobre el kernel la componen las bibliotecas nativas de Android, también llamadas librerías. Están escritas en C o C++ y compiladas para la arquitectura hardware específica del teléfono. Estas normal-

mente están hechas por el fabricante, quien también se encarga de instalarlas en el dispositivo antes de ponerlo a la venta. El objetivo de las librerías es proporcionar funcionalidad a las aplicaciones para tareas que se repiten con frecuencia, evitando tener que codificarlas cada vez y garantizando que se llevan a cabo de la forma “más eficiente”.

Figura 5: This image is reproduced from work created and HYPERLINK "http://code.google.com/policies.html" shared by the Android Open Source Project and used according to terms described in the HYPERLINK "http://creativecommons.org/licenses/by/2.5/" Creative Commons 2.5 Attribution License.

Entre las librerías incluidas habitualmente encontramos OpenGL (motor gráfico), Bibliotecas multimedia (formatos de audio, imagen y video), Webkit (navegador), SSL (cifrado de comunicaciones), FreeType (fuentes de texto), SQLite (base de datos), entre otras.

Android Runtime (Entorno de Ejecución):

Como podemos apreciar en el diagrama, el entorno de ejecución de Android no se considera una capa en sí mismo, dado que también está formado por librerías. Aquí encontramos las librerías con las funcionalidades habituales de Java así como otras específicas de Android.

El componente principal del entorno de ejecución de Android es la máquina virtual Dalvik. Las aplicaciones se codifican en Java y son compiladas en un formato específico para que esta máquina virtual las ejecute. La ventaja de esto es que las aplicaciones se compilan una única vez y de esta forma estarán listas para distribuirse con la total garantía de que podrán ejecutarse en cualquier dispositivo Android que disponga de la versión mínima del sistema operativo que requiera la aplicación.

Cabe aclarar que Dalvik es una variación de la máquina virtual de Java, por lo que no es compatible con el bytecode Java. Java se usa únicamente como lenguaje de programación, y los ejecutables que se generan con el SDK de Android tienen la extensión .dex que es específico para Dalvik, y por ello no podemos correr aplicaciones Java en Android ni viceversa.

Figura 6: This image is reproduced from work created and HYPERLINK "http://code.google.com/policies.html" shared by the Android Open Source Project and used according to terms described in the HYPERLINK "http://creativecommons.org/licenses/by/2.5/" Creative Commons 2.5 Attribution License

También cabe destacar que la máquina virtual Dalvik no está enfocada ni diseñada para la proporcionar ningún tipo de seguridad, dado que los permisos de las aplicaciones Android se ejecutan a nivel de sistema operativo y no a nivel de la máquina virtual Dalvik.

2.2.4. CAPA DE APLICACIÓN:

Framework de Aplicaciones:

La siguiente capa está formada por todas las clases y servicios que utilizan directamente las aplicaciones para realizar sus funciones.

La mayoría de los componentes de esta capa son librerías Java que acceden a los recursos de las capas anteriores a través de la máquina virtual Dalvik.

Siguiendo el diagrama encontramos:

Figura 7: This image is reproduced from work created and HYPERLINK "http://code.google.com/policies.html" shared by the Android Open Source Project and used according to terms described in the HYPERLINK "http://creativecommons.org/licenses/by/2.5/" Creative Commons 2.5 Attribution License.

- **Activity Manager**. Se encarga de administrar la pila de actividades de nuestra aplicación así como su ciclo de vida.

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

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

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

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

- **Package Manager**. Esta biblioteca permite obtener información sobre los paquetes instalados en el dispositivo Android, además de gestionar la instalación de nuevos paquetes.

Con paquete nos referimos a la forma en que se distribuyen las aplicaciones Android, estos contienen el archivo .apk, que a su vez incluyen los archivos .dex con todos los recursos y archivos adicionales que necesite la aplicación, para facilitar su descarga e instalación.

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

- **Resource Manager**. Con esta librería podremos gestionar todos los elementos que forman parte de la aplicación y que están fuera del código, es decir, cadenas de texto traducidas a diferentes idiomas, imágenes, sonidos o layouts.

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

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

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

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

2.2.5. Aplicaciones:

En la última capa se incluyen todas las aplicaciones del dispositivo, tanto las que tienen interfaz de usuario como las que no, las nativas (programadas en C o C++) y las administradas (programadas en Java), las que vienen preinstaladas en el dispositivo y aquellas que el usuario ha instalado.

Figura 8: This image is reproduced from work created and HYPERLINK "http://code.google.com/policies.html" shared by the Android Open Source Project and used according to terms described in the HYPERLINK "http://creativecommons.org/licenses/by/2.5/" Creative Commons 2.5 Attribution License.

En esta capa encontramos también la aplicación principal del sistema: Inicio (Home) o lanzador (launcher), porque es la que permite ejecutar otras aplicaciones mediante una lista y mostrando diferentes escritorios donde se pueden colocar accesos directos a aplicaciones o incluso widgets, que son también aplicaciones de esta capa.

Como podemos ver, Android nos proporciona un entorno sumamente poderoso para que podamos programar aplicaciones que hagan cualquier cosa. Nada dentro de Android es inaccesible y podemos jugar siempre con las aplicaciones de nuestro teléfono para optimizar cualquier tarea.

El potencial de Android se sitúa en el control total que se le da al usuario para que haga de su teléfono un dispositivo a su medida.

2.2.6. PERMISOS. ¿QUÉ SON?

En Android, las aplicaciones también requieren de permisos para leer y escribir sobre la información contenida en el dispositivo. El sistema de permisos de Android resulta útil en escenarios “simples” como el manejo de la información de contactos, hasta el uso más complejo de content providers y servicios que requieran de información suministrada de fuentes externas al dispositivo, sobre todo de Internet dónde el escenario de ataques es más amplio.

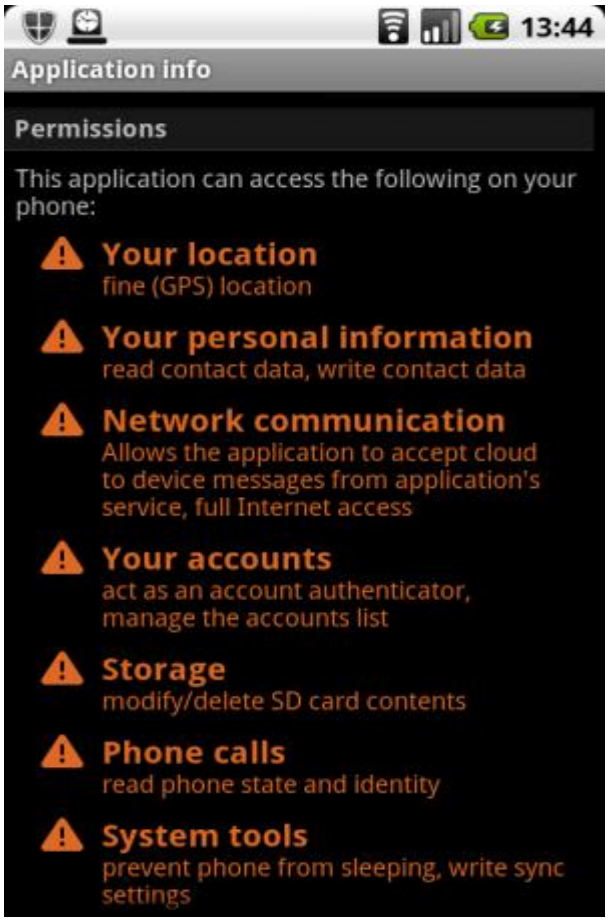

Figura 9: Extraído de http://applications.androidxiphone.com/android-permission-problems-free-privacy-protector/?lang=es

Las aplicaciones en Android funcionan en los dispositivos porque somos nosotros los que voluntariamente les otorgamos los permisos necesarios para que puedan funcionar.

Las aplicaciones Android necesitan permisos para interactuar con el dispositivo (wifi, datos, SMS, etc.)

Todos los permisos del sistema empiezan con android.permission y se enumeran en la Documentación oficial del SDK para la clase Manifest.permission. Este es el fichero donde se configuran los permisos requeridos por la aplicación. El desarrollador de la aplicación deberá indicar los permisos que necesita su apli-

cación en este fichero OBLIGATORIAMENTE. No hay forma de saltarse esta restricción.

```
    <application>
        <activity
            android:name="com.millennialmedia.android.MMActivity"
            android:configChanges="keyboardHidden|orientation|keyboard"
            android:theme="@android:style/Theme.Translucent.NoTitleBar" >
        </activity>
        <activity
            android:name="com.millennialmedia.android.VideoPlayer"
            android:configChanges="keyboardHidden|orientation|keyboard" >
        </activity>
    </application>

    <uses-sdk android:minSdkVersion="3" />

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
</manifest>
```

Figura 10: Ejemplo extraído de http://terminalesandroid.com/wp-content/uploads/2010/05/androidManifest.png

Figuras 11 (Izquierda) y 12 (Derecha): Extraído de http://www.androidzona.net/analisis-de-los-permisos-en-android/

Como se aprecia en las siguientes imágenes esta aplicación pide muchos permisos y posiblemente sea una de las aplicaciones más usadas en el mundo en todos los dispositivos móviles ya que es multi plataforma y no sólo funciona en Android, también se encuentra disponible para Blackberry, Nokia (alta gama) y Apple (Iphone y Ipod).

Los diferentes tipos de permisos existentes en la plataforma Android son los siguientes:

- **android.permission.ACCESS_CHECKIN_PROPERTIES**: Permite la lectura/escritura de la tabla de propiedades de la base de datos del registro pudiendo cambiar los valores que son actualizables.

- **android.permission.ACCESS_COARSE_LOCATION**: Permite la localización usando las redes inalámbricas a las que te encuentres como por ejemplo el wifi.

android.permission.ACCESS_COARSE_LOCATION: Permite la localización básicamente por ip.

- **android.permission.ACCESS_FINE_LOCATION**: Permite la localización usando la antena gps del dispositivo.

- **android.permission.ACCESS_LOCATION_EXTRA_COMMANDS**: Permite el acceso a la localización otorgada por los operadores de telefonía.

- **android.permission.ACCESS_MOCK_LOCATION**: Es un permiso de debug, emula a un operador de telefonía y sirve para la localización.

- **android.permission.ACCESS_NETWORK_STATE** Permite conocer el estado de la red.

- **android.permission.ACCESS_SURFACE_FLINGER**: Permite el acceso a las características de bajo nivel del deflector. Una aplicación normal no podrá usar este permiso.

- **android.permission.ACCESS_WIFI_STATE**: Permite leer información sobre las redes wifi.

- **android.permission.ACCOUNT_MANAGER**: Solo el sistema y aplica-

ciones del sistema pueden obtener este permiso. Permite el acceso a los identificadores de cuenta.

- **android.permission.AUTHENTICATE_ACCOUNTS**: Permite a una aplicación emular un identificador de cuenta para acceder al administrador de cuentas.

- **android.permission.BATTERY_STATS**: Permite el acceso a las estadísticas de la batería.

- **android.permission.BIND_APPWIDGET**: Permite la comunicación entre una aplicación y los datos que gestiona el servicio AppWidget. El AppWidget es el encargado de manejar los widgets de la pantalla y el permiso se usa para acceder a los datos que hayan sido generados por los distintos widgets.

- **android.permission.BIND_DEVICE_ADMIN**: Este permiso es requerido por el receptor de la administración del dispositivo para asegurar que sólo el sistema puede interactuar con él.

- **android.permission.BIND_INPUT_METHOD**: Este permiso es requerido por un InputMethodService para asegurarse que sólo el sistema puede unirse a él.

- **android.permission.BIND_REMOTEVIEWS**: Este permiso es requerido por un RemoteViewsService para asegurarse que sólo el sistema puede unirse a él.

- **android.permission.BIND_WALLPAPER**: Este permiso es requerido por WallpaperService para asegurarse que sólo el sistema puede unirse a él.

- **android.permission.BLUETOOTH**: Permite la conexión entre aplicaciones y dispositivos pareados por bluetooth.

- **android.permission.BLUETOOTH_ADMIN**: Permite a una aplicación buscar y parear dispositivos bluetooh.

- **android.permission.BRICK**: Formatea el terminal.

- **android.permission.BROADCAST_PACKAGE_REMOVED**: Permite a una aplicación difundir una notificación cuando una aplicación ha sido desinstalada.

- **android.permission.BROADCAST_SMS**: Permite a una aplicación di-

fundir una notificación de “mensaje SMS recibido”.

- **android.permission.BROADCAST_STICKY**: Permite la difusión de mensajes “sticky intents”.
- **android.permission.BROADCAST_WAP_PUSH**: Permite a una aplicación difundir notificaciones de mensajes wap entrantes.
- **android.permission.CALL_PHONE**: Permite a una aplicación realizar llamadas de teléfono sin necesidad de usar el dialer de Android.
- **android.permission.CALL_PRIVILEGED**: Igual que la anterior que además permite usar las llamadas de emergencia.
- **android.permission.CAMERA**: Permite usar la cámara de fotos/video.
- **android.permission.CHANGE_COMPONENT_ENABLED_STATE**: Permite a una aplicación cambiar de estado si un componente (que no sea de la aplicación) está iniciado o no. Dicho de otro modo es como tener un interruptor de dos posiciones ON/OFF y permite cambiar entre esos dos estados.
- **android.permission.CHANGE_CONFIGURATION**: Permite cambiar la configuración del sistema.
- **android.permission.CHANGE_NETWORK_STATE**: Permite el cambio de la conectividad de la red.
- **android.permission.CHANGE_WIFI_MULTICAST_STATE**: Permite a las aplicaciones acceder al modo “wifi multicast”.
- **android.permission.CHANGE_WIFI_STATE**: Permite el cambio de estado de la conexión wifi.
- **android.permission.CLEAR_APP_CACHE**: Permite a una aplicación a borrar la caché creada por otras aplicaciones.
- **android.permission.CLEAR_APP_USER_DATA**: Permite a una aplicación el borrado de los datos de usuario.
- **android.permission.CONTROL_LOCATION_UPDATES**: Permite activar/desactivar la ubicación sobre las actualizaciones de el proveedor de telefonía.
- **android.permission.DELETE_CACHE_FILES**: Permite el borrado de los ficheros en caché.

- **android.permission.DELETE_PACKAGES**: Permite el borrado de aplicaciones.
- **android.permission.DEVICE_POWER**: Permite el acceso de bajo nivel al administrador de energía del dispositivo.
- **android.permission.DIAGNOSTIC**: Permite el acceso de lectura/escritura a las herramientas de diagnóstico.
- **android.permission.DISABLE_KEYGUARD**: Permite a una aplicación el desbloqueo del “bloqueo de teclado”.
- **android.permission.DUMP**: Permite a una aplicación recuperar la información volcada por los servicios del sistema.
- **android.permission.EXPAND_STATUS_BAR**: Permite expandir o contraer la barra de estado.
- **android.permission.FACTORY_TEST**: Sirve para iniciar una aplicación en modo text. Solo el usuario root puede usar este permiso en una rom compilada en modo de pruebas.
- **android.permission.FLASHLIGHT**: Permite el acceso al flash.
- **android.permission.FORCE_BACK**: Permite a una aplicación volver atrás sea cual sea la aplicación que haya en primer plano.
- **android.permission.GET_ACCOUNTS**: Permite acceder al listado de cuentas registradas en el sistema.
- **android.permission.GET_PACKAGE_SIZE**: Permite a una aplicación saber el tamaño de cualquier paquete.
- **android.permission.GET_TASKS**: Permite saber las aplicaciones que hay en ejecución.
- **android.permission.GLOBAL_SEARCH**: Este permiso es complicado de explicar y no se usa en aplicaciones normales.
- **android.permission.HARDWARE_TEST**: Permite el acceso a los periféricos de hardware. Está diseñado para testear hardware.
- **android.permission.INJECT_EVENTS**: Permite a una aplicación insertar eventos tales como una pulsación en pantalla, uso de trackball, etc.

- **android.permission.INSTALL_LOCATION_PROVIDER:** Permite a una aplicación instalar un proveedor de localización en el “location manager”.
- **android.permission.INSTALL_PACKAGES**: Permite a una aplicación la instalación de otras aplicaciones.
- **android.permission.INTERNAL_SYSTEM_WINDOW**: Permite a una aplicación abrir ventanas usadas por la interfaz de usuario del sistema. No es usado por aplicaciones normales.
- **android.permission.INTERNET**: Permite a una aplicación abrir sockets.
- **android.permission.KILL_BACKGROUND_PROCESSES**: Permite a una aplicación llamar al killBackgroundProccesses.
- **android.permission.MANAGE_ACCOUNTS**: Permite a una aplicación administrar las cuentas almacenadas en el “AccountManager”.
- **android.permission.MANAGE_APP_TOKENS**: Sólo es usado por el sistema, permite crear o destruir tokens.
- **android.permission.MASTER_CLEAR**: No se pone nada sobre este permiso en la documentación de Android, aunque por su traducción podría servir para un borrado general.
- **android.permission.MODIFY_AUDIO_SETTINGS**: Permite la modificación de las características del sonido.
- **android.permission.MODIFY_PHONE_STATE**: Permite la modificación del estado del teléfono (encendido, apagado, etc.). No permite realizar llamadas.
- **android.permission.MOUNT_UNMOUNT_FILESYSTEMS**: Permite montar unidades de almacenamiento externo.
- **android.permission.MOUNT_FORMAT_FILESYSTEMS**: Permite formatear sistemas de almacenamiento externos.
- **android.permission.NFC**: Permite las operaciones de entrada/salida de los dispositivos NFC.
- **android.permission.PROCESS_OUTGOING_CALLS**: Permiso obsoleto, no lo uses.
- **android.permission.READ_CALENDAR**: Permite leer los datos almacenados en el calendario por el usuario.

- **android.permission.READ_CONTACTS**: Permite a una aplicación acceder a la lectura de los datos de tus contactos.
- **android.permission.READ_FRAME_BUFFER**: Permite a una aplicación tomar una captura de pantalla y más generalmente el acceso a los datos del “frame buffer”.
- **com.android.browser.permission.READ_HISTORY_BOOKMARKS**: Permite el acceso en modo lectura al historial y los favoritos del buscador.
- **android.permission.READ_INPUT_STATE**: Permite a una aplicación recuperar el estado de teclas o interruptores. Sólo es usado por el sistema.
- **android.permission.READ_LOGS**: Calificado como permiso peligroso. Permite el acceso a los logs creados por el sistema. En los logs puedes encontrar desde datos inútiles a claves de usuario.
- **android.permission.READ_PHONE_STATE**: Acceso de sólo lectura al estado del teléfono.
- **android.permission.READ_SMS**: Permite a una aplicación leer los SMS.
- **android.permission.READ_SYNC_SETTINGS**: Permite a una aplicación leer los datos de sincronización.
- **android.permission.READ_SYNC_STATS**: Permite a una aplicación leer estadísticas de la sincronización.
- **android.permission.RECEIVE_BOOT_COMPLETED**: Permite a una aplicación recibir el “ACTION_BOOT_COMPLETED“.
- **android.permission.RECEIVE_MMS**: Permite monitorizar los MMS recibidos.
- **android.permission.RECEIVE_SMS**: Permite monitorizar los SMS recibidos.
- **android.permission.RECEIVE_WAP_PUSH**: Permite a una aplicación monitorizar las notificaciones wap entrantes.
- **android.permission.RECORD_AUDIO**: Permite grabar audio.

- **android.permission.REORDER_TASKS**: Permite a una aplicación reorganizar las tareas.
- **android.permission.RESTART_PACKAGES**: Permiso obsoleto.
- **android.permission.SEND_SMS**: Permite a una aplicación enviar SMS.
- **android.permission.SET_ACTIVITY_WATCHER**: Sólo se usa en modo de depuración.
- **com.android.alarm.permission.SET_ALARM**: Permite a una aplicación crear una alarma en el sistema.
- **android.permission.SET_ALWAYS_FINISH**: Permite a una aplicación controlar si las actividades en segundo plano son finalizadas.
- **android.permission.SET_DEBUG_APP**: Configura una aplicación para ser depurada.
- **android.permission.SET_ORIENTATION**: Permite a una aplicación acceder a los parámetros de la rotación de pantalla. No se suele usar en aplicaciones normales.
- **android.permission.SET_POINTER_SPEED**: Acceso de bajo nivel al puntero (normalmente el puntero no se ve). No se usa en aplicaciones normales.
- **android.permission.SET_PREFERRED_APPLICATIONS**: Permiso obsoleto.
- **android.permission.SET_PROCESS_LIMIT**: Permite a una aplicación limitar el número de aplicaciones que se pueden ejecutar en un sistema Android.
- **android.permission.SET_TIME**: Permite cambiar la hora del sistema.
- **android.permission.SET_TIME_ZONE**: Permite cambiar la zona horaria.
- **android.permission.SET_WALLPAPER**: Permite a una aplicación poner un salvapantallas.
- **android.permission.SET_WALLPAPER_HINTS**: Permite a una aplicación poner texto en el salvapantallas.
- **android.permission.SIGNAL_PERSISTENT_PROCESSES**: Permite a

una aplicación enviar una señal a los procesos persistentes.

- **android.permission.STATUS_BAR**: Permite a una aplicación abrir, cerrar o deshabilitar la barra de estado y sus iconos.
- **android.permission.SUBSCRIBED_FEEDS_READ**: Permite a una aplicación acceder al feed del proveedor de contenido.
- **android.permission.SUBSCRIBED_FEEDS_WRITE**: No hay información disponible sobre este permiso.
- **android.permission.SYSTEM_ALERT_WINDOW**: Permite a una aplicación abrir ventanas usando TYPE_SYSTEM_ALERT.
- **android.permission.UPDATE_DEVICE_STATS**: Permite actualizar las estadísticas del dispositivo. No es usado por aplicaciones normales.
- **android.permission.USE_CREDENTIALS**: Permite a una aplicación solicitar tokens de autenticación del “account manager”.
- **android.permission.USE_SIP**: Permite a una aplicación usar el servicio SIP.
- **android.permission.VIBRATE**: Permite el acceso al vibrador del dispositivo.
- **android.permission.WRITE_APN_SETTINGS**: Permite a una aplicación modificar los ajustes APN.
- **android.permission.WAKE_LOCK**: Permite el acceso a los bloqueos de energía para mantener el procesador durmiendo o mantener la pantalla apagada.
- **android.permission.WRITE_CALENDAR**: Permite escribir datos en el calendario. No permite leer los datos.
- **android.permission.WRITE_CONTACTS**: Permite escribir contactos en la agenda. No permite leer los datos.
- **android.permission.WRITE_EXTERNAL_STORAGE**: Permite a una aplicación escribir en el almacenamiento externo.
- **android.permission.WRITE_GSERVICES**: Permite a una aplicación modificar el servicio de mapa de Google.
- **com.android.browser.permission.WRITE_HISTORY_BOOKMARKS**:

Permite a una aplicación añadir favoritos y modificar el historial del navegador. No permite la lectura.

- **android.permission.WRITE_SECURE_SETTINGS**: Permite a una aplicación leer o escribir los ajustes de seguridad.
- **android.permission.WRITE_SETTINGS**: Permite a una aplicación leer o escribir los ajustes del sistema.
- **android.permission.WRITE_SMS**: Permite a una aplicación escribir SMS.
- **android.permission.WRITE_SYNC_SETTINGS**: Permite a una aplicación escribir los ajustes de sincronización.

Comparativa de permisos utilizados en aplicaciones legitimas frente a permisos utilizados en malware (Izquierda: Malware – Derecha: Apps Legítimas):

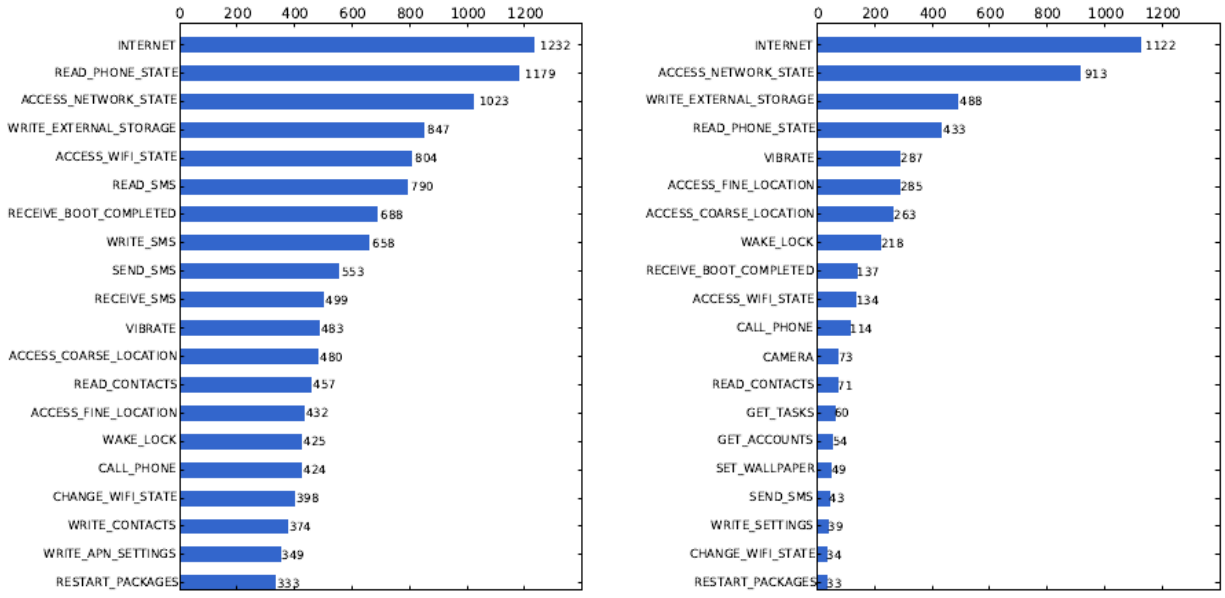

Tabla 1: Extraído de
http://www.csc.ncsu.edu/faculty/jiang/pubs/OAKLAND12.pdf

Cuando uno se encuentra con su primer proyecto Android o con los posteriores, va a encontrar un archivo suelto entre todo el árbol de paquetes, carpetas y otras hiervas. Ese archivo que tanto llama la atención y que se denomina AndroidManifest.xml, es uno de los archivos más importantes de cada aplicación.

Fichero donde se configuran los permisos requeridos por la aplicación. El desarrollador de la aplicación deberá indicar los permisos que necesita su aplicación en este fichero OBLIGATORIAMENTE. No hay forma de saltarse esta restricción.

Este XML se genera automáticamente al crear un proyecto y en él se declaran todas las especificaciones de nuestra aplicación. Cuando hablamos de especificaciones hacemos mención a las Activities utilizadas, los Intents, bibliotecas, el nombre de la aplicación, el hardware que se necesitará, los permisos de la aplicación, etc.

Este sería un ejemplo de este tipo de archivo:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="apt.tutorial"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".FirstApp"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>
    <uses-sdk android:minSdkVersion="3" />

</manifest>
```

Figura 13: Extraído de http://terminalesandroid.com/wp-content/uploads/2010/05/androidManifest.png

3. INTRODUCCIÓN A IPHONE

3.1. INTRODUCCIÓN A LA PLATAFORMA

Probablemente el terminal móvil más influyente en el mercado durante los últimos años ha sido el iPhone de Apple. Podría decirse que es el primer Smartphone que ha sabido atraer al mercado de masas. El iPhone combina un diseño elegante y pulcro con una pantalla multi-táctil; alta capacidad multimedia; una navegación funcional y rápida en Internet e impresionantes efectos visuales. Siempre manteniendo una increíble autonomía de la batería.

El iPhone ha cambiado significativamente desde su introducción, creciendo hasta convertirse en la primera plataforma con una distribución de aplicaciones centralizada y de uso sencillo, generando así un importante aumento del mercado de las aplicaciones móviles.

Debido a su uso de Objetivo-C, el desarrollo en iPhone implica el riesgo de fallas de seguridad tradicionalmente asociadas al software C. Sin embargo, algunas de ellas, son enmascaradas por los altos niveles de aplicaciones Cocoa Touch, y por tanto, requieren especial atención. Apple ofrece relativamente poca documentación a cerca de las prácticas de seguridad más adecuadas para llevar a cabo en sus dispositivos, es por ello que los desarrolladores normalmente no son conscientes de dichos riesgos.

El desarrollo de software para iPhone se lleva a cabo con Xcode y el Iphone SDK. El código se puede ejecutar en el emulador o en un dispositivo de desarrollo. Para algunas aplicaciones (por ejemplo, los que utilizan el "Keychain"), se requiere un dispositivo físico. La depuración se realiza dentro de Xcode a través de "gdb", aunque para los dispositivos con jailbreak, puede ser instalada un tercer "gdb" en el propio dispositivo para depurar cualquier aplicación.

3.2. MODELO DE SEGURIDAD IPHONE:

3.2.1. CODE SIGNING:

Las aplicaciones para el iPhone deben estar certificadas por una licencia válida o "code signing", una tecnología de seguridad que permite certificar la autoría de una aplicación.

Una vez que la app está certificada, el sistema OS X puede detectar cualquier cambio en ella, tanto si este cambio ha sido introducido de forma accidental o a través de un código malicioso.

Algunas aplicaciones, en particular los que utilizan el "Keychain" o criptografía primitivos, son diseñadas sólo para ser ejecutadas en un dispositivo real, en lugar de un emulador. Para obtener un "code signing", se puede utilizar la herramienta de Acceso a Keychain para crear una solicitud de licencia (CSR), tal y como se describe en la guía "Apple's Code Signing" (http://developer.apple.com/documentation/Security/Conceptual/CodeSigningGuide/Procedures/Procedures.html)

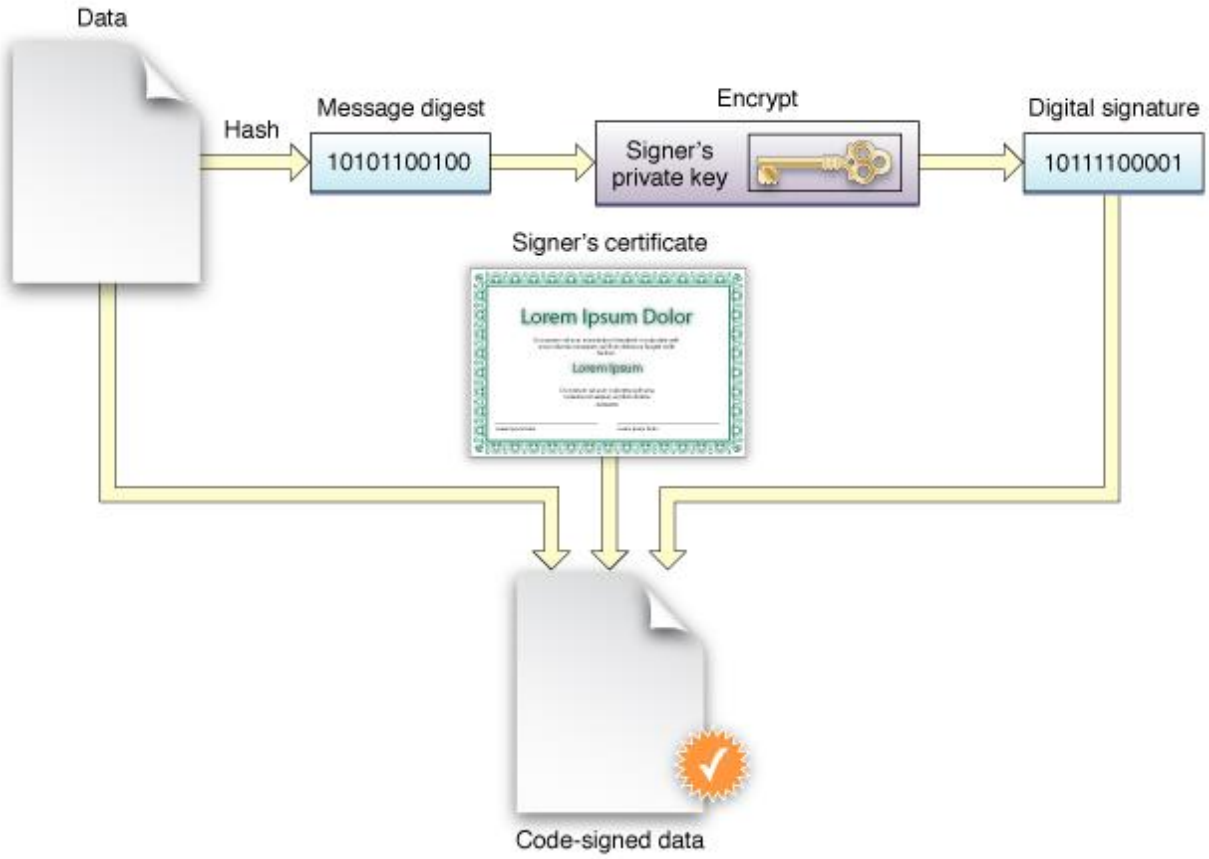

Figura 14: Extraído de <https://developer.apple.com/library/mac/#documentation/Security/Conceptual/CodeSigningGuide/Introduction/Introduction.html>

Una vez que el certificado ha sido emitido, un proceso por cierto, bastante rápido, se puede importar al "Keychain".

En el caso de no ser miembro del "iPhone Developer Program" (programa de desarrolladores de Iphone), sigue siendo posible utilizar certificados auto-firmados para validar aplicaciones. Sin embargo, esto implica deshabilitar ciertas comprobaciones de seguridad en el dispositivo, por lo que no es aconsejable para los terminales que no son utilizados exclusivamente para el desarrollo.

3.2.2. Permisos y Controles de usuario:

Los distribuidores han diseñado diferentes formas de abordar el aislamiento de procesos en aplicaciones móviles Apple ha optado por utilizar los controles obli-

gatorios de acceso (en inglés Mandatory Access Controls ó MAC) como su mecanismo para restringir las capacidades de las aplicaciones, lo que tiene la ventaja de ser extremadamente flexible y además, jugar con las siglas "MAC", seña de identidad de la marca. Los permisos adicionales se exponen directamente al usuario en forma de instrucciones.

3.2.3. PERMISOS:

La concesión de permisos para funciones específicas, se solicita a través de pop-ups mientras el usuario está haciendo uso de la API (en lugar de durante la instalación, como otros sistemas como Android).

La solicitud más común suele ser para las funciones de geo-localización. Otra petición habitual es solicitar el consentimiento del usuario para leer los datos de la cámara. Curiosamente, es notable la ausencia de una solicitud de permiso para hacer grabaciones de audio.

Figura 15: Extraído de http://sanziro.com/2009/06/photo-geotagging-in-iphone.html

4. INTRODUCCIÓN A WINDOWS PHONE

Para saber más sobre este tema, del cual hemos extractado parte de este capítulo, te recomendamos que vayas a los blogs de Hispasec y de Winphonemetro, donde encontrarás más información (ver bibliografía)

Una de las primeras cosas a tener en cuenta es que Windows Phone no es Windows Mobile. Si bien ambos son sistemas operativos que se ejecutan por encima de un kernel CE desde una perspectiva de la API de Windows Mobile y Windows Phone ambos tienen muy poco en común.

Windows Phone es un sistema operativo propietario para móviles desarrollado por Microsoft. Windows Phone introdujo un nuevo lenguaje de diseño, anteriormente llamada Metro UI, pero más tarde cambió el nombre a simplemente Modern.

Inicialmente se trató del Windows Phone 7 (originalmente llamado "Windows Phone 7 Series"), cuyo nombre clave durante su desarrollo era "Photon",y es el sucesor de la versión del sistema operativo móvil Windows Mobile, desarrollado por Microsoft y basado en el núcleo Windows Embedded CE 6.0. Microsoft mostró Windows Phone 7 el 15 de febrero, en el Mobile World Congress 2010 en Barcelona y reveló más detalles del sistema en el MIX 2010 el 15 de Marzo. La versión final de Windows Phone 7, se lanzó el 1 de septiembre de 2010, y la versión final del SDK estuvo disponible el 16 de septiembre de 2010. WP7 se lanzó en Europa y Asia el 21 de octubre de 2010 y en EEUU el 8 de noviembre de 2010. Inicialmente, Windows Phone 7 estaba destinado para lanzarse durante el 2009, pero varios retrasos provocaron que Microsoft desarrollara Windows Mobile 6.5 como una versión de transición.

Durante el Mobile World Congress 2010 en Barcelona, Microsoft reveló detalles de Windows Phone 7, mostrándolo como un nuevo sistema operativo que incluye funciones de integración con los servicios Xbox Live y Zune. La interfaz, conocida como "Metro", ha sido revisada en su totalidad y comparte características visuales similares a la interfaz del dispositivo Zune HD.

Microsoft declaró que pedirá a los fabricantes que los requerimientos de hardware sean "altos, pero justos", con la obligatoriedad de que todos los dispositivos con Windows Phone 7 dispongan de al menos tres botones (Atrás, Inicio y Buscar) y un receptor de radio FM.14

El inmediato sucesor fue Windows Phone 8 que supuso todo un paso adelante en la integración de los terminales Windows con Windows RT o Windows 8. La nueva versión, desgraciadamente, no se podía instalar en los terminales Windows Phone 7.

Microsoft otorga licencias del software a los fabricantes de hardware de terceros, pero mantiene una lista estricta de los requisitos mínimos para el hardware que se ejecuta en asegurar la mejor experiencia de usuario.

En 2011 Nokia anunció que había elegido Windows Phone como sistema operativo para todos sus futuros teléfonos inteligentes que proporcionan un sólido respaldo para el sistema operativo en ciernes y en el mismo tiempo apostando todo su negocio de telefonía móvil en su éxito.

4.1. PLATAFORMA WINDOWS OS

Android, iOS, Windows Phone, cada una de estas plataformas móviles tenían que empezar en alguna parte, y ninguno fue ni de lejos perfecta en el primer intento. Afortunadamente, cada sistema operativo mejora con cada iteración, lo que en opinión de analistas sucedió con la versión 8.1 a partir de la cual se transformó en un OS móvil a tener en cuenta.

El OS Windows Movile 8.1 posee características comunes con otros OS como son el centro de notificaciones y teclado en pantalla y otras particulares como Live Tiles. Además incorpora Cortana, un asistente virtual similar a Siri, Google Now y de Samsung S Voice. Windows Phone 8.1 también soporta geolocalización que se integra con el asistente Cortana.

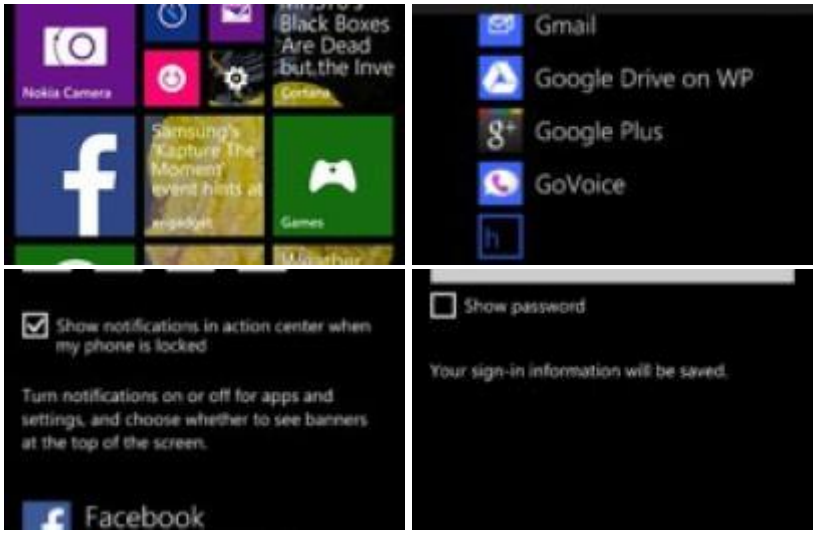

Figura 16: Fuente capturas propias

La última actualización de firmware de otoño de 2014 (conocido simplemente como Actualización 3) añadió soporte para pantallas más grandes y de mayor resolución, así como los nuevos procesadores. Esta actualización fue crucial para Microsoft, ya que hasta entonces, la empresa se esforzó por convencer a los fabricantes y los consumidores que un buque insignia de Windows Phone podría ser tan bueno como un dispositivo Android o iOS, al mismo precio. Aunque Microsoft tenía algunos requisitos de hardware estrictos para asegurar que los dispositivos WP8 se optimizaran a su gusto, como sucedió con Nokia encontró una manera de diferenciar el Lumia 1020 al presentar una cámara PureView de 41 megapíxeles, pero por lo demás no había mucha diferencias prácticas sobre hardware que montara otros OS como Android (HTC o Samsung Galaxy S4).

La versión 8.1 ofrece soporte para dispositivos de doble SIM que son los más importantes sobre todo para los mercados emergentes donde esa funcionalidad es muy apreciada. También permite guardar las aplicaciones en una tarjeta SD, lo que si es diferencial frente a otros OS, sobre todo en hardware con poca capacidad de almacenamiento. Curiosamente, Microsoft dice que las aplicaciones aún están encriptados para asegurar el acceso no controlado al dispositivo.

Figura 17: Fuente Microsoft

Lanzamiento de Microsoft es Windows 10 al que también da soporte para smartphones. Windows 10 es el intento más reciente de Microsoft para unificar sus sistemas operativos en uno solo para que esté disponible en toda clase de dispositivos -- incluyendo celulares, tabletas y computadoras -- ofreciendo una experiencia similar.

La versión de Windows 10 Mobile (para celulares) fue habilitada en febrero de 2015, pero la versión previa de Windows 10 habilitada el 10 de abril de 2015, es la que realmente incluye la mayoría de novedades, como por ejemplo las apps universales que prometen funcionar en diferentes dispositivos

La interfaz de Windows 10 Mobile no cambia drásticamente de Windows Phone 8.1 y a primera vista no es muy fácil distinguir la diferencia. Sin embargo, Windows 10 para celulares trae algunas modificaciones que ofrecen mayor personalización y una mejor experiencia.

Otra novedad es la aparición del sucesor de Internet Explorer, que hizo su aparición por primera vez en la versión previa de Windows 10 para computadoras y posteriormente en la segunda versión previa de Windows 10 para celulares. Conocido anteriormente como Project Spartan, Microsoft Edge usa el mismo motor de procesamiento que la versión para computadora, mientras que también intenta mantener el mismo concepto de diseño.

Las únicas nuevas funciones que trae integrado Edge para Windows 10 Mobile son las listas de lectura y el modo de lectura. Mientras que la primera permite añadir páginas web a una lista que se sincronizará en un futuro para poderlas leer después en cualquier dispositivo con Windows 10, el modo de lectura simplifica el diseño de la página Web y quita las gráficas innecesarias para que el contenido sea más fácil de leer y visualizar.

Los celulares con Windows 10 Mobile pueden conectarse a un monitor a través de un cable para que la interfaz y los apps universales automáticamente se ajusten y puedan funcionar como, casi, una computadora de escritorio. Además permiten conectar un teclado y mouse inalámbrico (Bluetooth) para poder usar el terminal con Windows 10 como si fuera una computadora de escritorio.

4.2. SEGURIDAD EN WINDOWS PHONE 8

Cuando hablamos del modelo de seguridad de Windows Phone lo primero es hablar del Secure Boot que ayuda a garantizar la integridad de todo el sistema operativo.

La ejecución de código no-confiable es lo que crea la mayor parte de los problemas y representa importantes vulnerabilidades en las aplicaciones, sistemas, infraestructura, así como los datos personales. El Secure Boot agrega seguridad a un procesador que no tiene ningún tipo de seguridad incorporado y puede evitar instalaciones indeseadas en SO. Por tanto el Secure Boot ayuda a evitar que el malware se instale en el teléfono.

Lo anterior se integra en el UEFI (Unified Extensible Firmware Interface), que es una interfaz de firmware estándar para PCs, diseñada para reemplazar el BIOS (sistema básico de entrada y salida). Es un estándar creado por más de 140 compañías tecnológicas que forman parte del consorcio UEFI, en el que se incluye Microsoft. Se ha diseñado para mejorar la interoperabilidad del software y solucionar las limitaciones del BIOS. Esto proporciona, entre otras cosas, mayor seguridad, ya que ayuda a proteger el proceso previo al inicio (o pre arranque) frente a ataques de bootkit.

Con Windows 8 Microsoft ha adoptado totalmente una arquitectura de seguridad en la que se llevaba trabajando mucho tiempo: Unified Extensible Firmware Interface (UEFI). En esencia, UEFI hace todo lo que la BIOS hacía, pero también trabaja como una especie de sistema operativo independiente, haciendo que el sistema operativo sea accesible, esté intacto y legítimo antes de iniciarlo.

A partir de Windows Vista, cada driver que se metía en modo kernel necesita venir firmado por una entidad con un certificado especial emitido por Microsoft, lo que hizo que muchos vieran cómo tenían que cambiar sus procesos para poder seguir haciendo drivers para Windows Vista. Esta medida, aunque sí dificultó y acabó con muchos de los canales de malware tradicionales siendo una buena medida de fortificación, no fue una medida definitiva para siempre, ya que aparecieron formas de saltarse esas protecciones. Entre las técnicas usadas para meter drivers firmados fue hacerse con certificados de fabricantes de drivers autori-

zados y que incluso permitió utilizarlos para firmar malware, algo que pasó y que llegó hasta el propio servicio de Windows Update, con lo que la propia Microsoft distribuyó malware.

Debido a esto, la industria trabajó en un sistema de protección contra este tipo de actividades maliciosas, y generó lo que hoy en día se conoce como Secure Boot, una protección que viene implementada y basada en el propio firmware del equipo, de serie en todos los que vengan con UEFI. Esta protección lo que hace es evitar que se cargue un MBR de un sistema no conocido, por lo que el arranque de los sistemas operativos vendrán firmados digitalmente, y el equipo comprobará la firma del MBR antes de lanzar el arranque del sistema o liberar las claves de cifrado desde el chip TPM para que se descifren los discos con Bitlocker.

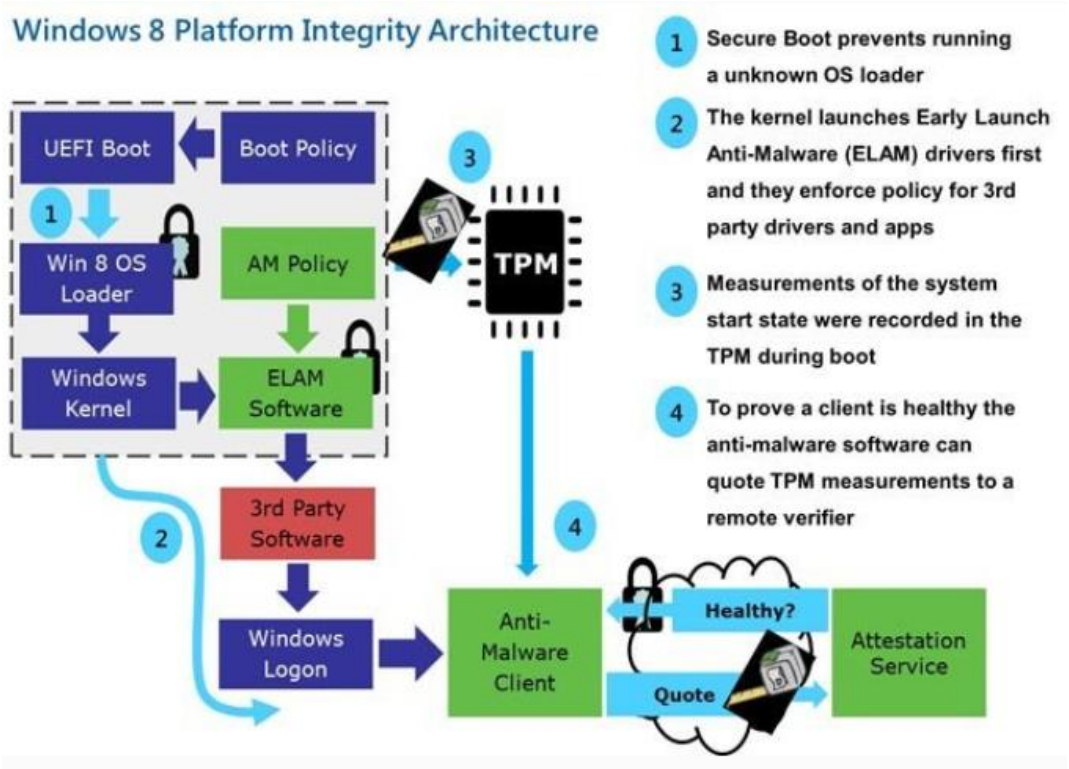

Figura 18: Fuente http://www.mobilejaw.com

La función de Secure Boot es impedir la ejecución de cualquier software no firmado y certificado por el fabricante, por lo que cualquier amenaza que intentara atacar durante el inicio se vería frustrada, pues se detendría el arranque del sistema. Esto impide también la posibilidad de instalar distribuciones diferentes a las aceptadas por Microsoft.

La función de Secure Boot es ser totalmente restrictivo con cualquier software no certificado. No se hacen distinciones de ningún tipo; si no está certificado no se ejecuta. Sin embargo, el usuario siempre podrá elegir desactivarlo desde el panel de control.

Esta arquitectura para el arranque seguro es proporcionada por el SoC (Sistema en un chip) o firmware integrado y se realiza en dos fases:

- Gestores de arranque pre-UEFI[1] (Unified Extensible Firmware Interface) para inicializar el hardware
- UEFI arranque seguro ayuda a garantizar la integridad de las aplicaciones y el sistema operativo Windows UEFI

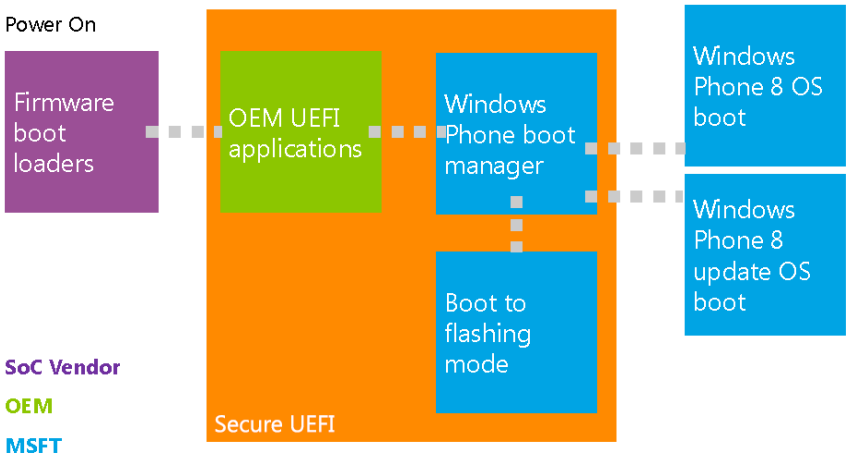

Figura 19: Fuente Microsoft

Todos los binarios de Windows Phone 8 deben tener firmas digitales firmados por Microsoft para funcionar. Durante el proceso de fabricación se inserta el hash de la clave pública utilizada para firmar los gestores de arranque iniciales, lo que asegura el proceso descrito.

Por otro lado tenemos Trusted Boot, cuya función es proteger el resto del proceso de inicio del sistema, verificando que todos los componentes de inicio tienen un mínimo de integridad y son de confianza. El boot loader verifica la firma digital del kernel de Windows Phone antes de cargarlo, y este a su vez verifica a cada uno del resto de componentes que influyen en el proceso de inicio. En caso de que un archivo haya sido modificado con código malicioso, Trusted Boot protegería al resto de componentes y evitaría su inicio.

[1] La Interfaz de Firmware Extensible Unificada, Unified Extensible Firmware Interface (UEFI), es una especificación que define una interfaz entre el sistema operativo y el firmware. UEFI remplaza la antigua interfaz del Sistema Básico de Entrada y Salida (BIOS).

A continuación se procedería a cargar los componentes del sistema y cualquier aplicación que deba ser iniciada en el arranque. Esto no se llevará a cabo si una aplicación no contiene la firma que toda aplicación requiere para estar en la tienda Windows Phone, o la firma del grupo de desarrolladores de una gran empresa. Esto hace que sea extremadamente difícil para un hacker ejecutar código malicioso en Windows Phone.

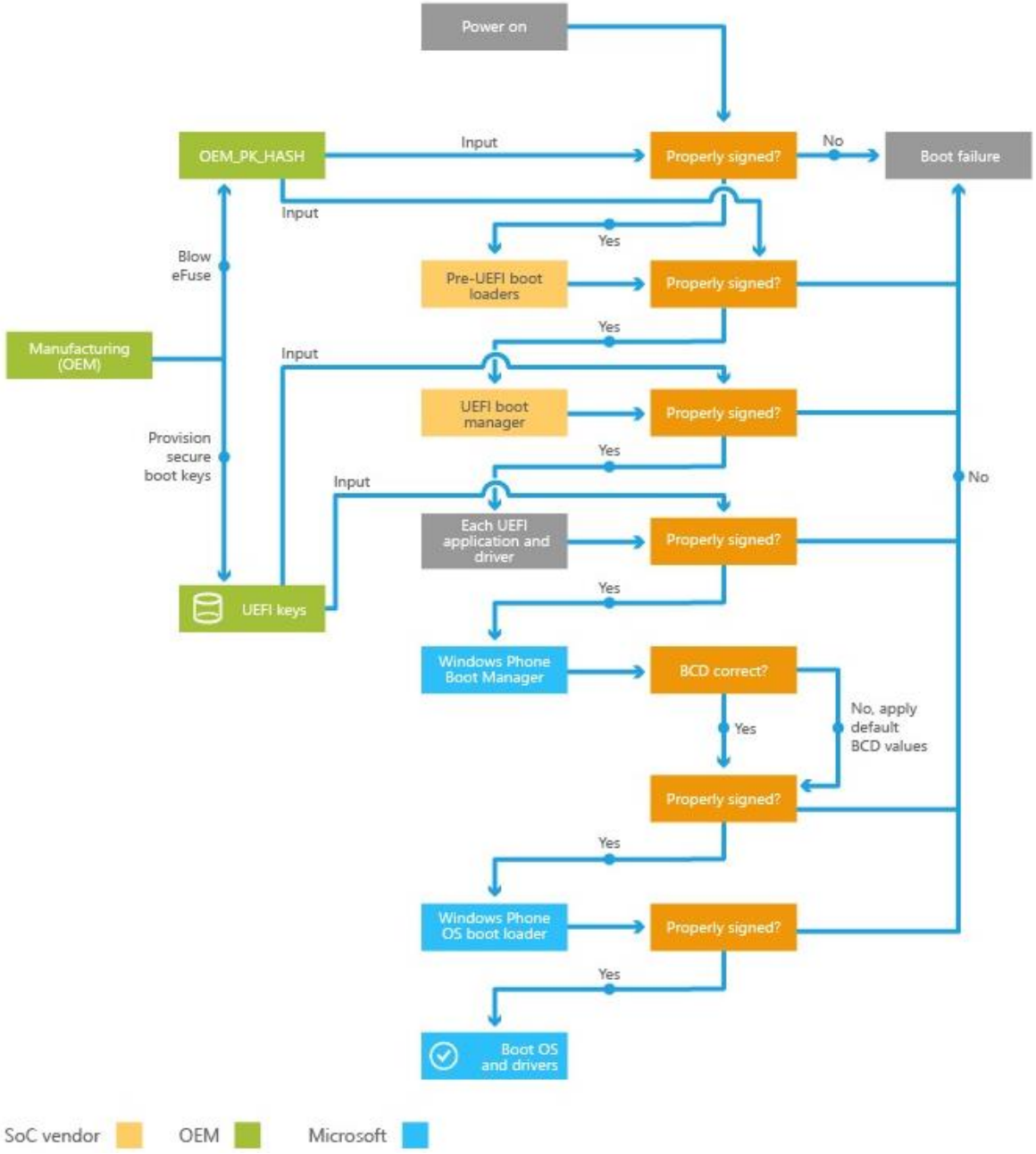

Figura 20: Fuente Microsoft

Por tanto, como hemos visto como primer punto del modelo de seguridad de Microsoft Phone es cargar al propio sistema operativo de una política de seguridad bastante férrea. Para las aplicaciones sucede lo mismo y por tanto cada aplicación que se ejecuta tenga un espacio de trabajo independiente. Así es como en Windows Phone cada aplicación se instala en un directorio independiente y sólo puede tener acceso a sus ficheros y no a los de otra aplicación. Incluso no se tenía acceso ni a las librerías como música o vídeos, lo que se cambia con Windows Phone 8.1 al haber creado un concepto de librería parecido al de Windows.

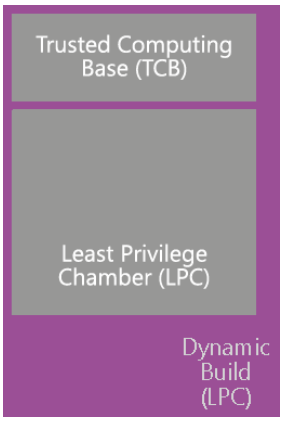

Figura 21: Fuente Microsoft

Además, el propio sistema operativo realiza preguntas ante cambios de configuración si se realizan desde programas. Es por ello que podemos hacer enlaces a la configuración de Wi-Fi, red, etc. pero no se puede cambiar de forma automática por aplicaciones externas. Esto hace que una aplicación no pueda hacer operaciones como cambiar la contraseña de bloqueo por su cuenta, o bloquear el terminal para secuestrarlo, cómo hacen algunas aplicaciones en Android.

El segundo punto es controlar las aplicaciones que se pueden instalar en el sistema operativo. Es como tener un antivirus gigante por parte de Microsoft a la entrada de Windows Phone. Así, se tiene la seguridad de que la aplicación instalada, los permisos que solicita, no son para algo peligroso, como robar información y enviarla a otros servidores. Esta misma política la tiene Apple con su tienda de aplicaciones, pero Microsoft cuenta con una experiencia dilatada en detectar virus, lo que le da una ventaja.

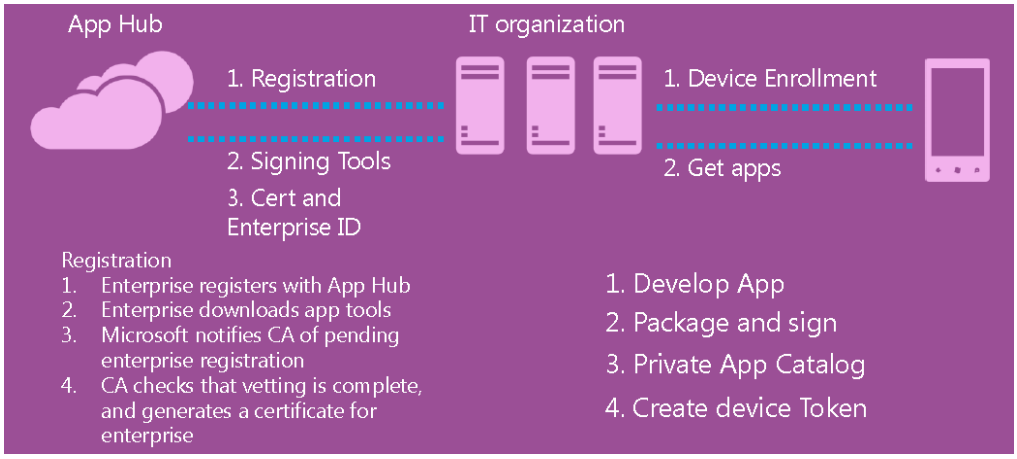

Figura 22: Fuente Microsoft

Para el desarrollo por terceros Microsoft hace uso del proceso Microsoft Security Development Lifecycle (SDL) a la hora de desarrollar software, del cual también se puede beneficiar cualquier desarrollador. Esto se traduce en aplicaciones que ya desde su propio diseño son realmente seguras y son capaces de eliminar o al menos mitigar riesgos potenciales de seguridad. De hecho, esta característica se mencionaba en el informe CISCO de 2014 como una de las que han marcado la diferencia entre Windows Phone respecto a Android y iOS.

Figure 1. Microsoft Security Development Lifecycle (Simplified)

Figura 23: Fuente Microsoft

Todas estas medidas han hecho que en casi los cuatros de andadura que tiene Windows Phone no se haya conocido problema alguno de seguridad. Muchos dirán que será porque no interesa, pero tampoco tenemos que infravalorar todas estas medidas.

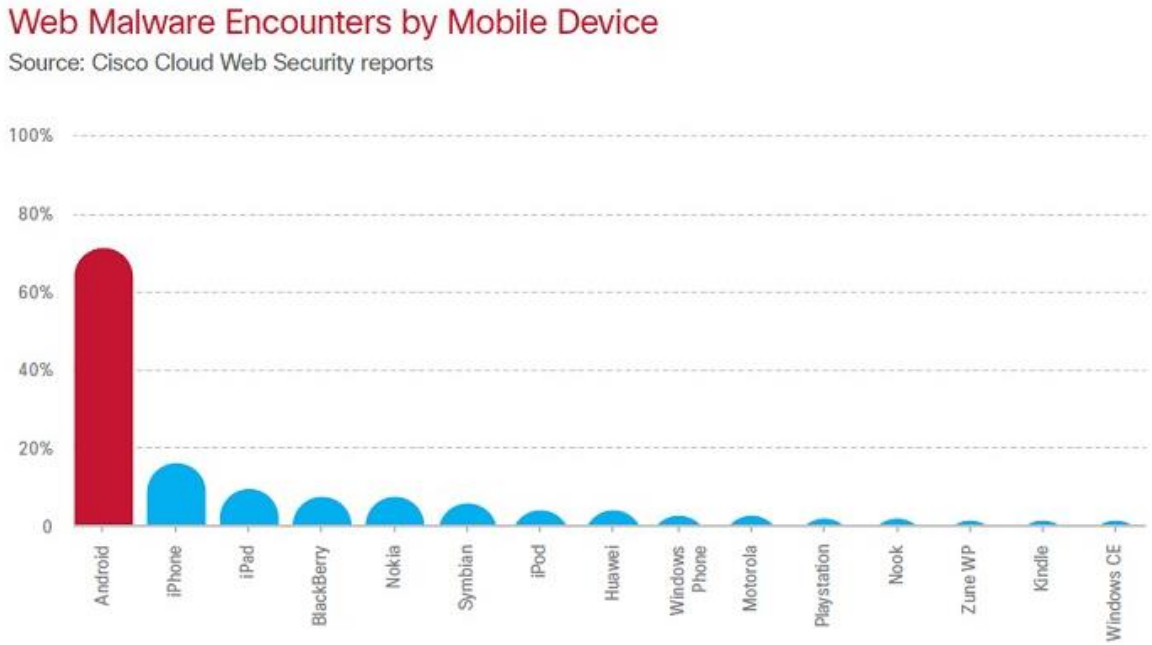

Figura 24: Fuente CISCO

Precisamente, este aspecto de la seguridad, Microsoft lo está aprovechando para diferenciarse de iOS y Android. Para así atacar los nichos de mercados gubernamentales y empresariales. Dos mercados en los que justamente Black-Berry tenía una presencia importante. Y en los que Microsoft puede entrar gracias a esa calidad en la seguridad y a su integración con otras muchas aplicaciones empresariales.

5. INTRODUCCION A BLACKBERRY:

Los dispositivos BlackBerry son producidos por Research In Motion (RIM), una compañía canadiense que presentó la primera BlackBerry en 1999 como un localizador de mensajería (un “busca” o “beeper”) y PDA, que se podía utilizar también para acceder al correo electrónico corporativo.

En 2002, la BlackBerry 5810 fue el primer dispositivo en incluir funciones de teléfono. RIM diseña todos sus terminales y produce su propio sistema operativo BlackBerry. Los dispositivos tienen, desde sus inicios, una distintiva forma cuadrada con un teclado QWERTY completo y una rueda de desplazamiento lateral o central.

La combinación de un teclado completo, sus funciones para la administración empresarial, y la robusta integración de correo electrónico han hecho de las BlackBerry, unos terminales muy populares.

Las modernas BlackBerry son más “amigables” que sus predecesoras y tienen características de consumo como GPS, cámara fotográfica, navegador web completo, y reproductor multimedia. RIM lanzó su primer teléfono con pantalla táctil, el BlackBerry Storm, en 2008.

5.1. INTRODUCCIÓN A LA PLATAFORMA:

Las versiones del sistema operativo BlackBerry posteriores al 4.6 incluyen un completo y potente navegador HTML/JavaScript/CSS2 con el cual, se puede acceder a la mayoría de los sitios de Internet, incluyendo los que utilizan tecnología AJAX. Las versiones del navegador anteriores a la 4.2 son incompletas y no son compatibles con funciones web avanzadas.

RIM promueve el desarrollo de aplicaciones por parte de terceros y proporciona documentación bastante completa y apoyo a los desarrolladores a través de foros. El sistema operativo BlackBerry es fundamentalmente Java y soporta J2ME Mobile Information Device Profile (MIDP) 1,0, un subconjunto de MIDP 2,0, Wireless Application Protocol (WAP) 1.2, además de perfiles de Configuración Limitada de Dispositivos Conectados (CLDC) por defecto.

Para aprovechar totalmente la plataforma BlackBerry, se requiere una Api Java propiedad de RIM que permite hacer uso de determinadas funciones específicas.

Las aplicaciones pueden utilizar al mismo tiempo Apis de RIM, MIDP y las CLDC, pero las de tipo UI (interfaz de usuario) de RIM sólo se pueden usar dentro de las aplicaciones CLDC debido a sus conflictos de subprocesamiento GUI con las aplicaciones MIDP. Por esa razón, la mayoría de las aplicaciones Java específicas para Blackberry se construyen sobre base CLDC y emplean Apis de RIM. A estas aplicaciones se les conoce, comúnmente como "RIMlets"

Los desarrolladores también pueden diseñar aplicaciones utilizando tecnologías alternativas, incluyendo modelos de servicios web basados en datos, dirigidos al tiempo de ejecución del Mobile Data System (MDS).

La mayoría de dispositivos móviles "encuestan" al servidor de forma intermitente para comprobar si hay nuevos mensajes. Sin embargo, BlackBerry utiliza la tecnología "Push", por la cual el servidor inicia la comunicación inmediatamente después de que llegue un mensaje. El software del servidor propio de RIM monitorea las cuentas de correo electrónico de los usuarios e inicia el “Push”.

Funcionamiento del servicio Push de BlackBerry:

Figura 25: Extraído de http://es.blackberry.com/developers/javaappdev/pushapi.jsp#tab_tab_works

5.1.1. BLACKBERRY PUSH PLUS

- El proveedor de contenidos envía una solicitud Push.
- La infraestructura de BlackBerry® devuelve una respuesta.
- La infraestructura de BlackBerry envía los datos al smartphone BlackBerry.
- El smartphone BlackBerry devuelve una respuesta a la infraestructura de BlackBerry.

- La infraestructura de BlackBerry reenvía la confirmación al proveedor de contenidos.

- La notificación de lectura se devuelve a la infraestructura de BlackBerry.

La política, las aplicaciones y otros mensajes también pueden ser enviados utilizando este mecanismo. Para ahorrar en ancho de banda, el servidor comprime el contenido antes de enviarlo al dispositivo. La arquitectura “Push” prolonga la duración de la batería y reduce el tiempo de entrega de mensajes porque el dispositivo no sobrecarga la batería comprobando constantemente la disponibilidad de la red y solicitando nuevos mensajes.

Cada dispositivo BlackBerry tiene un número de identificación personal único en el mundo (PIN) tanto para funciones de mensajería como de gestión. El PIN de BlackBerry es público. Los usuarios utilizan los PINs para encontrarse en el BlackBerry Messenger, y los administradores pueden usar su PIN para identificar los dispositivos están manejando.

5.1.2. BlackBerry Enterprise Server (BES)

La mayoría de las organizaciones que facilitan Blackberry a sus empleados, suelen instalar el BlackBerryEnterprise Server (BES). BES se integra con los servidores del correo electrónico corporativo servidores (incluyendo Exchange, Lotus Notes y Novell Groupware), monitorea las cuentas de los usuarios, y exhibe el correo electrónico y los archivos adjuntos en cuanto llegan. Los administradores también pueden utilizar BES para controlar los dispositivos y aplicaciones de despliegue, la política de autoría del dispositivo e incluso forzar una limpieza remota del dispositivo.

El alto nivel de control que permite BES satisface las necesidades de los administradores más exigentes y lo convierte en el actual líder en gestión empresarial de dispositivos.

Una vez que un dispositivo está asociado a una instancia de BES, se crea un túnel encriptado entre el dispositivo y sus BES. Todo el tráfico de información fluye a través de este túnel, con el BES actuando como un puente entre la red de la compañía móvil, Internet, y la intranet de la empresa. El Mobile Data System (MDS), componente de BES, es el responsable real del enrutamiento interno y esa función de puente o enlace.

La investigación en seguridad para BlackBerry se ha centrado principalmente en la relación entre BES y el dispositivo debido a que BES proporciona un puente entre la "trinidad" de Internet, Intranet y redes móviles. En este capítulo se adopta un enfoque diferente y se trata la seguridad del dispositivo en sí, especialmente en lo que se refiere a las aplicaciones.

5.2. MODELO DE SEGURIDAD BLACKBERRY, DESARROLLO Y PRUEBAS DE SEGURIDAD:

Todas las aplicaciones de terceros desarrolladas para BlackBerry deben estar programadas en Java o emplear una de las aplicaciones alternativas de RIM.

El uso universal de tiempos de ejecución "gestionados" sacrifica ligeramente la velocidad en favor de una reducción de la posibilidad de sufrir un ataque en el dispositivo además de aumentar la productividad del desarrollador.

Además de la ejecución de la aplicación Java, hay que tener en cuenta la ejecución del MDS. Las aplicaciones MDS se construyen utilizando un plug-in Visual Studio, una presentación basada en el lenguaje JavaScript, y servicios web especialmente diseñados para ello. Las empresas desarrollan aplicaciones MDS para interactuar con sistemas de back-end, tales como sus inventarios o sus sistemas de ventas. El desarrollo de estas aplicaciones es muy especializado y por tanto, no nos extenderemos más a cerca de ello en este capítulo.

5.2.1. CÓDIGO DE SEGURIDAD

Sólo el BlackBerry JVM y el firmware de nivel más bajo están programados en código nativo(C / C + +, ASM), lo que elimina una gran parte de la susceptibilidad de ataque de las BlackBerry que sí podrían ser vulnerables a desbordamientos de búfer y otras cuestiones de corrupción de memoria.

La prueba es el hecho de que no existen evidencias públicas de vulnerabilidades de corrupción de la memoria de una BlackBerry, lo que supone un historial impresionante para cualquier fabricante de terminales.

Para detener los desbordamientos de búfer y controlar el comportamiento de aplicaciones BlackBerry Java, RIM no permite la invocación de Java nativa (JNI) ni la reflexión de Java. JNI permite que la conexión del código de Java con los nativos de C / C + +, y permitiendo su uso facilitaría a las aplicaciones Java acceder a funciones no deseadas o, más aún, corromper la memoria. La reflexión de Java puede ser utilizada para eludir las restricciones de acceso público/privado en las clases de Java, y su uso podría permitir a las aplicaciones invocar métodos internos del sistema. La desactivación de ambas propiedades de Java es estándar para dispositivos de JME.

5.2.2. Permisos y controles de usuario

Los permisos se determinan por aplicación y se asignan basándose en la licencia de cada una de ellas o en una política especificada por el usuario. La mayoría de las API no son consideradas susceptibles y se posibilita el acceso por parta de cualquier aplicación sin firmar. El conjunto de APIs sensibles incluye aquellas APIS que permiten internarse en datos de información personal (PIM) o en las características del teléfono; las que operan en la configuración del sistema y/o la red. Las aplicaciones que utilizan estas API pueden tener que ser firmadas, en función de si son MIDP o CLDC, o propiedad de RIM. Las APIs sensibles que son propiedad de RIM y que no forman parte de MIDP2 se conocen como RIM Controlled APIs. Las RIM Controlled APIs se dividen en conjuntos de API diferentes, cada una con una firma única autoridad.

Las tres autoridades de firma más comunes se abrevian como sigue:

- RCR (Runtime RIM criptográfica) Incluye la mayor parte de RIM API criptográfica. La criptografía de clave pública API requieren una firma diferente clave de Certicom.

- TSR (RIM Runtime API) Proporciona acceso a la funcionalidad de la plataforma sensible, como el Administrador de autorización de aplicaciones.

- RBB (RIM BlackBerry Apps API) Proporciona control de una función de BlackBerry para aplicaciones (por ejemplo, el navegador de Blackberry).

6. COMPARATIVA: ANDROID, IPHONE, BLACKBERRY ¿CUÁL ES MÁS SEGURA Y MEJOR?

6.1. ANDROID

El sistema operativo Android de Google es la plataforma más utilizada en las tabletas y los smart-phones en la actualidad, con un gran número de vendedores que ofrecen sus propias versiones personalizadas. Debido a su facilidad de integración con muchos servicios de Google, Android está evolucionando rápidamente.

Por desgracia, cuando se trata de seguridad, Android todavía tiene un largo camino por recorrer. Los elevados plazos que son necesarios para el lanzamiento de soluciones sobre problemas de seguridad influye el hecho que se requiere un lanzamiento diferente y específico para cada compañía, fabricante y modelo.

Como resultado, muchos teléfonos Android están estancados usando versiones antiguas e inseguras del sistema operativo.

En relación a las aplicaciones, la principal fuente es el Android Market, el cual contiene miles de aplicaciones, la mayoría de ellas gratuitas. Dichas aplicaciones son subidas por los desarrolladores y no pasan ninguna revisión antes de ser publicadas, lo que permite ponerlas velozmente a disposición de los usuarios, pero dejando una puerta abierta a aplicaciones maliciosas que penetran para quedarse hasta que Google descubre el origen remoto de la amenaza y consigue eliminar la app de los dispositivos (como ha ocurrido ya en numerosas ocasiones).De forma alternativa, otros mercados fiables como Amazon Appstore ofrecen muestras de prevención de acceso malicioso, sin embargo, esto también ha provocado quejas por el lento despliegue de las actualizaciones de las aplicaciones.

Debido a que Android usa un muy modelo flexible, las aplicaciones pueden hacer cosas que no pueden ser hechas en otras plataformas. El usuario es informado de lo que la aplicación podría realizar en el momento de la instalación y él mismo pude elegir instalarlo o no.

Una vez descargada, las aplicaciones de terceros tienen la posibilidad (siempre que haya sido autorizado con anterioridad) leer y enviar mensajes, hacer y recibir llamadas, acceder a internet y encender o apagar el micrófono o la cámara.

Dado que los usuarios no suelen leer detenidamente e intentar entender lo que implican estos permisos, se ha descubierto que algunas App de Android envían y reciben mensajes de tarificación adicional; graban las pulsaciones de teclas de los usuarios o los sonidos; monitorean la geo-localización de los usuarios, o incluso contienen malware “botnet”, como el que podría encontrarse en una máquina de escritorio. Hay muy pocas soluciones de terceros disponibles que aseguren el dispositivo, y además su eficacia es, en muchos casos, cuestionable.

La flexibilidad de Android, hace de él una gran elección para los usuarios más competentes, pero requiere algún conocimiento para garantizar la seguridad durante el uso. Habitualmente esto requiere “rootear” el dispositivo e instalar actualizaciones personalizadas directamente si el operador no lo facilita. Por supuesto, ¡no apto para novatos técnicos!

6.2. BLACKBERRY

Mientras Android ha encontrado su nicho de mercado en el consumidor común, BlackBerry se ha convertido en la joya de la corona del mundo de los negocios. Con sus usuarios tildados de “adictos” por su dependencia a su BlackBerry, los terminales de RIM se han ganado el sobrenombre de “Crackberry”.

¡Incluso el presidente Obama no podía deshacerse de su aparato! al parecer ante la irritación de los servicios secretos y el deleite de Research in Motion-RIM.

La seguridad es uno de los puntos fuertes de BlackBerry, con la posibilidad de ofrecer datos completamente encriptados; un control estricto de lo que se hace con el dispositivo; restringir lo que las aplicaciones individuales pueden o no pueden hacer; requerir “tunneling” a todo el tráfico que pasa a través de los servidores de la compañía; controlar las aplicaciones y mucho más. El lado oscuro de dicho control conlleva un coste; mantener el terminal seguro puede consumir mucho tiempo a un usuario que no haga un uso “empresarial” de una BlackBerry.

BlackBerry App World, el mercado de aplicaciones de terceros, ofrece cierto grado de revisión a todas las licencias. Sin embargo, el código fuente no pasa el filtro de RIm, y sólo puede deducirse parte del comportamiento de la aplicación. Blackberry no ha sufrido, ni remotamente, tantos ataques de spyware y malware como Android, aunque sí existen evidencias de aplicaciones maliciosas y actualizaciones portadoras de troyanos-spyware.

Definitivamente la capacidad de bloquear y asegurar los terminales Blackberry, es un plus. Pero, ya que fue diseñado con los operadores en mente, puede resultar algo complicado para los usuarios comunes, a menos que sean especialmente cuidadosos. El lanzamiento de Blackberry 10, más orientado a un uso standard de los terminales, promete hacer prosperar más aún el uso de este sistema por parte de usuarios comunes.

6.3. IOS-IPHONE

En un mercado donde el líder está representado por un robot verde y el vagón de cola por una especie de “tecnodroga” adictiva (Blackberry), la compañía con el segundo puesto cuenta con la fidelidad y satisfacción de un público “de culto”. Estamos refiriéndonos, por supuesto, al IOS de Apple, la plataforma donde parece que cada nueva mejora vende aún más que su predecesora, hagan lo que hagan.

IOS es una plataforma de evolución más lenta y vigilancia más rigurosa que Android, con funciones diseñadas para dar una experiencia consistente, fluida y controlada. Como resultado de esto, la plataforma es ideal para los diseñadores de Apple, sin embargo, es poco flexible para otros desarrolladores. Debido al nivel de control de Apple sobre IOS, los usuarios no pueden parchear las vulnerabilidades hasta que Apple publique una actualización, lo que a veces lleva meses y en muchos casos los dispositivos más antiguos no son compatibles con ellas y por tanto, nunca pueden ser parcheados.

Para aplicaciones existe el App Store, en el que Apple suele ser bastante restrictivo. De hecho, hay muchas instancias de que algunas aplicaciones han sido denegadas por razones desconocidas o misteriosas. La más famosa fue la aplicación de voz de Google en 2009.

Dado que se concede la completa capacidad a las aplicaciones de hacer todo lo permitido (con la excepción de algunas cosas como las notificaciones y Reading Location), no existen permisos complejos para que los usuarios gestionen o administren los dispositivos. Si bien ha existido más de un ejemplo de una aplicación maliciosa accediendo a la App Store, el más notable fue sólo una prueba de un investigador.

También cabe destacar, el ecosistema paralelo y circundante de los dispositivos de Apple, “jailbreaking” (donde los usuarios han retirado conscientemente el software de protección de Apple). Hacer “jailbreaking” ofrece a los usuarios la capacidad de dotar a los dispositivos de nuevas características; protegerse de los problemas que Apple aún no ha reparado, e instalar aplicaciones no autorizadas (o piratas). No obstante, la eliminación de estas protecciones deja a los usuarios más vulnerables desde el punto de vista de seguridad, como ocurrió con el gusano Ikee en 2008.

Los dispositivos iOS muestran un gran nivel de seguridad, pero esto se cobra un costo de flexibilidad que tal vez no agrade a los usuarios más exigentes de smartphone/ Tablet.

6.4. WINDOWS PHONE

Windows Phone trata de hacerse un sitio como OS empresarial incorporando funcionalidades demandadas desde este sector. Con Windows Phone 8, Microsoft admite la posibilidad de revocar aplicaciones, restringir el reenvío de correo electrónico, de forma remota inscribirse o dispositivos Anular inscripción, y de forma remota actualizar aplicaciones empresariales.

Otra capacidad de Windows Phone 8 que no está disponible para otros sistemas operativos móviles es su integración con un Active Directory. Esto significa que las herramientas de MDM compatibles pueden acceder a los grupos de Active Directory, y a continuación asignar políticas a los grupos en lugar de mantener un conjunto separado de los grupos en la herramienta MDM del conjunto en Active Directory. La función reduce el riesgo que los empleados no estén en los grupos correctos para las políticas que deben aplicarse o que caen en el olvido, cuando se eliminan del Active Directory, pero no en la base de datos de usuario de la herramienta MDM.

Para esto Microsoft utiliza un gestor central en Windows Phone 8 se llama DM Client que contiene todos los usuarios relevantes y perfiles corporativos (como el Registro de Windows, en efecto), en lugar de confiar en un conjunto de perfiles de configuración instalados por separado (como la carpeta del sistema OS X, en efecto).

Table 2: Other native management capabilities compared

(Typically requires a mobile device management server to use)

	Apple	Google	Samsung	BlackBerry	Microsoft
Capability	iOS 7, 8	Android 4, 5	Android 5 + Knox 2.4	BlackBerry 10 + BES12	Windows Phone 8, 8.1
Encryption	AES 256, user has no disable option	AES 128, user has disable option, only some models support encryption	AES 256, user has disable option, only Knox devices support encryption	AES 256, user has disable option in personal workspace	AES 256, user has no disable option
FIPS 140-2 certification	Yes (Level 1)	No	Some models (Level 1)	Yes (Level 2)	Yes (Level 1)
Over-the-air data encryption	Yes	Yes	Yes	Yes	Yes
S/MIME	Yes	No	Yes	Yes	Yes [2]
VPN	Yes	Yes	Yes	Yes	Yes [2]
Configure VPN	Yes	Yes	Yes	Yes	Yes [2]
Per-app VPN	Yes	Yes [3]	Yes	Yes	Yes [2]
Restrict/block app stores	Yes	No	Yes	Yes	Yes
Business licensing and provisioning	Yes	Yes [3]	Yes [3]	Yes	No
Restrict/block wireless LANs	Yes	No	Yes	Yes	Yes [2]
Configure allowable access points	Yes	Yes	Yes	Yes	Yes [2]
Signed apps required	Yes	No	Yes	Yes	Yes
Selective wipe of business apps and data only	Yes	Yes [3]	Yes	Yes	Yes [2]
Remotely update business apps	Yes	Yes [3]	Yes	Yes	Yes
Secure boot	Yes	Yes [1]	Yes	Yes	Yes
Active Directory container signin	NA	No	Yes [3]	No	No
App sandboxing	Yes	Yes	Yes	Yes	Yes
Disable copy and paste	Yes	Yes	Yes	Yes	Yes [2]
Disable iCloud/Microsoft Account/Google Account sync and storage	Yes	No	Yes	Yes	Yes [2]

[1] Added by some smartphone makers. [2] In Windows Phone 8.1 only (and VPN support is partial). [3] In secured container only.

Figura 26: Extraído de http://www.cso.com.au/article/574311/mobile-security-ios-vs-android-vs-blackberry-vs-windows-phone/

6.5. CONCLUSIÓN

Por todo ello, ¿cuál es la mejor plataforma en lo que respecta a la seguridad? Para la mayoría de los consumidores la respuesta sería iOS, pero para los que

tienen mayor experiencia técnica, Android funciona, siempre que se sea cuidadoso.

Sin embargo, si un usuario está dispuesto a hacer jail-breaking también puede obtener beneficios de Android.

Blackberry puede ser también una buena opción en cuanto a seguridad, aunque generalmente, los que eligen dispositivos de esta compañía no lo hacen por motivos de seguridad. Windows Phone y el resto de plataformas podrían presentarse como buenos competidores en el futuro pero en el presente no, aún no han tenido una amplia penetración, sobre todo después de lo ocurrido con el touchpad.

En resumen, la recomendación según el tipo de usuario es la siguiente:

- Una persona poco técnica: iOS (iPhone, iPad…)
- Un “techie”: iOS/Android
- Un usuario de negocios: Blackberry/iOS

Aun así, tratar de dar un diagnóstico de seguridad de un OS es demasiado simplista, ya que existen múltiples aspectos de seguridad en los sistemas modernos como vemos en la figura 27, que van a influir en la decisión. En la actualidad los OS son suficientemente robustos, por lo que más que decir cuál es el más seguro, es mejor hablar de cuál es el que mejor se ajusta a mis necesidades.

Table 1: Exchange ActiveSync (EAS) policy support compared

("MDM" means a separate mobile device management server is required)

Policy	Apple iOS 7, 8	Google Android 4, 5	Samsung Android 5 + Knox	BlackBerry BlackBerry 10	Microsoft Windows Phone 8, 8.1
Allow device encryption	Yes	Yes	Yes	Yes	Yes
Require device encryption	Yes	Yes [1]	MDM	Yes	Yes
Encrypt storage card	NA	Yes	Yes	Yes	Yes
Minimum password length	Yes	Yes	Yes	Yes	Yes
Minimum number of complex characters (password)	Yes	Yes	Yes	Yes	Yes
Password history	Yes	Yes	Yes	Yes	Yes
Device wipe threshold	Yes	Yes	Yes	Yes	Yes
Disable removable storage	MDM	No	MDM	MDM	No
Disable camera	Yes	Yes	Yes	MDM	No
Disable SMS text messaging	No	No	MDM	MDM	No
Disable Wi-Fi	MDM	No	MDM	MDM	Yes [2]
Disable Bluetooth	MDM	No	MDM	MDM	No
Disable IrDA	NA	No	No	No	No
Require manual sync while roaming	Yes	No	Yes	MDM	No
Allow Internet sharing from device	MDM	No	MDM	MDM	MDM
Allow desktop sharing from device	MDM	No	MDM	No	No
Disable email attachment access	Yes	MDM	Yes	No	Yes
Disable POP3/IMAP4 email	MDM	No	MDM	Yes	No
Allow consumer email	No	No	MDM	Yes	No
Allow browser	Yes	MDM	MDM	No	MDM
Configure message formats (HTML or plain text)	No	No	MDM	No	No
Include past email items (days)	Yes	No	MDM	Yes	Yes
Email body truncation size (KB)	No	No	MDM	No	Yes [2]
HTML email body truncation size (KB)	No	No	MDM	No	Yes [2]
Include past calendar items (days)	Yes	No	MDM	Yes	No
Require signed S/MIME messages	Yes	No	MDM	MDM	Yes [2]
Require encrypted S/MIME messages	Yes	No	MDM	MDM	Yes [2]
Require signed S/MIME algorithm	Yes	No	MDM	MDM	Yes [2]
Require encrypted S/MIME algorithm	Yes	No	MDM	MDM	Yes [2]
Allow S/MIME encrypted algorithm negotiation	Yes	No	MDM	MDM	Yes [2]
Allow S/MIME soft certs	No	No	Yes	MDM	Yes [2]

[1] Storage areas only. [2] Windows Phone 8.1 only.

Figura 27: Extraído de http://www.cso.com.au/article/574311/mobile-security-ios-vs-android-vs-blackberry-vs-windows-phone/

CONCLUSIONES

A lo largo del tema hemos podido comprobar que el malware en terminales móviles es actualmente el mayor problema de seguridad en esta tecnología. Como hemos podido ver, los terminales móviles no han logrado establecer un modelo de seguridad claro y eficiente para sus usuarios.

Aunque en su origen, estos ecosistemas tuvieron a la seguridad como un parámetro de diseño, se ha podido ver a lo largo del tema que la usabilidad ha sido otra vez más enemiga de la seguridad, relegando la facilidad de uso a la seguridad en el dispositivo, por lo que estamos en un momento en el que la seguridad empieza a impactar en los usuarios.

Este problema resulta más acuciante en entornos normalmente seguros, como el ámbito empresarial, donde se ha abierto una fisura por donde documentos, correos, agendas, etc., han quedado expuestas al mundo exterior, sin el adecuado control por parte de los responsables de seguridad.

La gran versatilidad y facilidad de uso que han proporcionado los terminales móviles han permitido que estén actualmente en manos de todo el mundo, lo que ha traído consigo que el problema de la seguridad se haya extendido. Por tanto hay que tener claros los conceptos aquí tratados con el fin de poder dar una correcta valoración de los riesgos que se han de afrontar.

RECAPITULACIÓN

Tras lo visto en el tema, como recapitulación, vamos a indicar lo que INTECO comenta sobre esta problemática:

> "Al mismo tiempo que crece la demanda de cantidad y calidad de funcionalidades así como más servicios y más aplicaciones, ha comenzado a aumentar la preocupación en torno a la seguridad de las tecnologías móviles. Los ciber-delincuentes y las organizaciones criminales no son ajenas a esta revolución y se han dado cuenta del enorme potencial criminal que encierran estas tecnologías.
>
> Un claro ejemplo de lo anterior es que las amenazas de seguridad asociadas a las tecnologías móviles no han dejado de crecer desde la aparición del primer código malicioso para dispositivos móviles, que data de 2004. La tendencia parece que no va a cambiar a medio plazo.
>
> Paralelamente a todo lo anterior, se ha producido un aumento del comercio electrónico a través de Internet y, por extensión, también ha llegado a los dispositivos móviles, a través de los cuales se realizan cada vez más transacciones comerciales. El aumento de las operaciones comerciales realizadas a través de terminales móviles preocupa enormemente a los expertos en seguridad. Los dispositivos móviles han demostrado ser un blanco perfecto para los ciber-criminales, que han descubierto la facilidad (relativa)

para darle valor económico a los ataques, obteniendo beneficios económicos rápidamente.''

Por otro lado, hemos de tener en cuenta que los dispositivos móviles presentan una mayor diversidad y oferta que los PC tradicionales. Además cuando pensamos en terminales móviles, solemos pensar erróneamente en teléfonos smartphones, pero hay que tener en cuenta que las tablets son el mismo problema, pero con una mayor penetración que los terminales móviles en todos los segmentos de la sociedad y de la empresa, debido a que en gran parte emulan o incluso mejoran las capacidades y características que los PC tradicionales ofrecen.

AUTOCOMPROBACIÓN

1. La Open Handset Alliance (OHA) es:

a) Una alianza de fabricantes de software para lanzar un sistema operativo móvil universal.

b) Una alianza comercial de 84 compañías que se dedica a desarrollar estándares abiertos para dispositivos móviles.

c) El fabricante de los dispositivos BlackBerry.

d) Ninguna de las demás respuestas es válida.

2. El núcleo del sistema operativo Android está basado en:

a) El Kernel de Linux versión 2.6, similar al que puede incluir cualquier distribución de Linux, como Ubuntu, solo que adaptado.

b) Un desarrollo abierto entre varios grupos de desarrolladores.

c) Una variación de Windows CE.

d) Está desarrollado íntegramente en Java.

3. La capa de las bibliotecas nativas de Android, también llamadas librerías, están:

a) Escritas en JAVA y compiladas para la arquitectura hardware específico del teléfono.

b) Escritas en C o C++ y compiladas para la arquitectura hardware específico del teléfono.

c) Escritas en Visual Basic y compiladas para la arquitectura hardware específico del teléfono.

d) Escritas en código máquina y compiladas para la arquitectura hardware específico del teléfono.

4. El componente principal del entorno de ejecución de Android es:

a) El lanzador o launcher.

b) La máquina virtual Dalvik

c) Las librerías de aplicación.

d) Ninguna de las demás respuestas es válida.

5. El archivo que se denomina AndroidManifest.xml, es uno de los archivos más importantes de cada aplicación porque…

a) Es el fichero donde se configuran los permisos para desarrolladores.

b) Es el fichero donde se configuran los procesos que se lanzan al iniciar la aplicación.

c) Es el fichero donde se firma la aplicación.

d) Es el fichero donde se configuran los permisos requeridos por la aplicación.

6. El desarrollo de software para iPhone se lleva a cabo con Xcode y:

a) El Java SDK. El código se puede ejecutar en el emulador o en un dispositivo de desarrollo.

b) El iPhone SDK. El código sólo se puede ejecutar en un dispositivo de desarrollo.

c) El iPhone SDK. El código se puede ejecutar en el emulador o en un dispositivo de desarrollo.

d) Ninguna de las demás respuestas es válida.

7. Las aplicaciones para el iPhone deben estar certificadas por:

a) Certificado de desarrollador una tecnología de seguridad que permite certificar quien es un desarrollador autorizado.

b) Un HASH del código que permite certificar la autoría de una aplicación.

c) Una licencia válida o “code signing”, una tecnología de seguridad que permite certificar la autoría de una aplicación.

d) Ninguna de las demás respuestas es válida.

8. RIM promueve el desarrollo de aplicaciones por parte de terceros y proporciona documentación bastante completa y apoyo a los desarrolladores a través de foros. Para aprovechar totalmente la plataforma BlackBerry, se requiere:

a) Una Api Java propiedad de RIM que permite hacer uso de determinadas funciones específicas.

b) Una Api C++ propiedad de RIM que permite hacer uso de determinadas funciones específicas.

c) Una Api propietaria y desarrollada por RIM que permite hacer uso de determinadas funciones específicas.

d) Una Api Visual Basic propiedad de RIM que permite hacer uso de determinadas funciones específicas.

9. En relación a las aplicaciones, la principal fuente es el Android Market, el cual contiene miles de aplicaciones, la mayoría de ellas gratuitas. Dichas aplicaciones son subidas por:

a) Personal de Google Play y no pasan ninguna revisión antes de ser publicadas.

b) Los desarrolladores y no pasan ninguna revisión antes de ser publicadas.

c) Los desarrolladores pero pasan una revisión antes de ser publicadas.

d) Ninguna de las demás respuestas es válida.

10. El ecosistema paralelo y circundante de los dispositivos de Apple, es el:

a) “Jailbreaking” donde los usuarios han retirado conscientemente el software de protección de Apple.

b) “Jailbreaking” donde los usuarios configuran el iphone para aceptar otros Markets al margen del de Apple.

c) Market denominado Google Play, donde se encuentran cientos de APP gratuitas.

d) Creado por comunidades de desarrolladores libres, al que los operadores de telefonía otorgan acceso sólo a sus clientes.

SOLUCIONARIO

1.	b	**2.**	a	**3.**	b	**4.**	b	**5.**	d
6.	c	**7.**	c	**8.**	a	**9.**	b	**10.**	a

PROPUESTAS DE AMPLIACIÓN

Tras leer este tema, estás en condiciones de poder empezar a analizar el comportamiento de malware en terminales móviles.

1. Busca en Internet información de un malware, por ejemplo The Android DDoS 1 origin.
2. Trata de hacerte con una muestra del código, para lo cual tendrás que buscar en páginas de consultores de seguridad.
3. Trata de explicar con un gráfico cómo funciona dicho código, y esboza alguna contramedida que pudieras ser efectiva.

Otra cuestión sobre la que deberías empezar a ampliar conocimientos es con el Sistema Operativo Android. Busca en Internet algún emulador y analizador del mismo, para poder tener un laboratorio donde poder probar el malware que puedas ir encontrando.

BIBLIOGRAFÍA

- Rodríguez, A. (2011). "Arquitectura de Android". Recuperado de http://androideity.com/2011/07/04/arquitectura-de-android/
- Hollmann, S. (2010). “Android Manifest, una introducción”. Recuperado de http://piedralibre.wordpress.com/2010/11/25/android-manifest-una-introduccion/
- CervanteX, (2011). “Análisis de los permisos en Android”. Recuperado de http://www.androidzona.net/analisis-de-los-permisos-en-android/
- Porras, E. (2012). “Sistemas operativos móviles: Android”. Recuperado de http://www.eve-ingsistemas-u.blogspot.com.es/2012/04/sistemas-operativos-moviles-android.html
- Instituto Nacional de Tecnologías de la Comunicación (INTECO), (2012). “Seguridad en Dispositivos Móviles”. Recuperado de http://cert.inteco.es /extfrontinteco/img/File/demostrador/monografico_seg_disp_moviles.pdf
- INTECO, (2012). Seguridad en dispositivos móviles Ed. INTECO, Monográficos de Seguridad del Catálogo STIC. Número 5. Recuperado de http://cert.inteco.es/extfrontinteco/img/File/demostrador/monografico_seg_disp_moviles.pdf
- Nachenberg, C (2011) Una Mirada a la Seguridad de los Dispositivos Móviles. Ed Symantec Corporation, Análisis de los enfoques de seguridad en las plataformas iOS de Apple y Android de Google. Resumen ejecutivo. Recuperado de http://www.sadvisor.com/downloads/InformeSymantecSET.pdf, Informe SYMANTEC, SIMANTEC
- Caracciolo, C y Sallis, E (2011). Seguridad en Dispositivos Móviles; Smartphone- Pocket PC. Ed. Root-Secure. Recuperado de http://www.root-secure.com/arch/Seguridad%20en%20Dispositivos%20Moviles%20 Smartphone%20y%20Pocket%20PC.pdf
- Blog de http://unaaldia.hispasec.com/2014/05/microsoft-detalla-las-caracteristicas.html

- Blog de http://winphonemetro.com/2015/06/windows-phone-sistema-mas-seguro-opinion