ISec Lab #16

Seguridad en Android (I)

Septiembre 2011

Daniel Fernández Bleda
dfernandez <arroba> isecauditors.com

Introducción

Aunque la primera palabra que le viene a uno a la mente cuando oye la palabra Android es Sistema Operativo para móviles, es curioso que en la propia definición que Google Inc. da de Android no aparecen en ningún momento las palabras Sistema y Operativo, la palabra clave es "stack" o pila: Android es una pila de software de código abierto para teléfonos móviles y otros dispositivos ("Android is an open-source software stack for mobile phones and other devices" [1]). Lo curioso es que si tenemos en mente el concepto de "pila de software" entenderemos muy fácilmente el concepto tras la arquitectura de Android, el sistema que ha conquistado el mundo móvil más rápido que ningún otro con un detalle importante en la historia de la tecnología y es que se considera el verdugo de Microsoft en los terminales móviles. De cualquier modo, y con la venia de Google y de los más puristas, emplearemos el término Sistema Operativo para referirnos a Android.

En esta serie de ISecLabs pretendemos enseñar muchas cosas sobre Android centrándonos en el lado de la seguridad: como se implementan los controles de seguridad en este Sistema Operativo, cómo se define o regula la seguridad en el software que podemos instalar en un terminal, analizando de forma global el malware que lo afecta, como crear entornos para el análisis de este malware mediante el SDK de Android y algunos proyectos muy interesantes con cierta solera, pero más desconocidos, como Kirin y otros más recientes sobre los que se ha hablado mucho en el mundo del análisis de malware Android como DroidBox.

1. Las Aplicaciones en Android (mercado libre, o Free Market)

Para todo aquel acostumbrado a los sistemas operativos de escritorio (sobre todo a nivel doméstico) donde un usuario puede llevar acabo "cualquier" acción sobre su sistema (ya sea directamente –léase Windows XP- o bien respondiendo múltiples veces a un molesto aviso que dice algo a lo que hay que responder siempre "Aceptar" –léase Windows Vista/7- los terminales móviles basado en sistemas iOS cambiaron esta filosofía en gran manera. Conceptos como la firma digital de aplicaciones y, sobretodo, la existencia del "gran hermano" revisor que controla que el usuario sólo pueda instalarse aplicaciones previamente validadas por él modificaron el concepto de libertad del usuario.

Ante esa limitación de la libertad para el usuario, el enfoque de Android pretendía darle a este y, sobre todo a los desarrolladores, una capacidad de crear –a los segundos- y de instalar –a los primeros- infinita. La firma digital de las aplicaciones (archivos .apk, que no son más que un paquete comprimido de un conjunto predefinido de archivos con una estructura de directorios bajo una convención establecida) en Android tiene la única finalidad de identificar al creador no de fiscalizarle o controlar sus desarrollos.

Para el que no conozca muy bien lo que estamos explicando el resumen simplificado (menudo reto) es que en un iPhone únicamente podré instalar aplicaciones firmadas con un certificado de la propia Apple y esta sólo firmará aquel software que haya pasado su validación de calidad (e impuestos asociados para el desarrollador). En Android, el propio desarrollador firmará con su certificado, concedido por Google, su aplicación, pero únicamente él y con la finalidad de que el usuario identifique al creador. En este segundo caso Google no "garantizará" la aplicación. Pero lo realmente más importante es que esto permitirá que yo instale en mi sistema con Android (mediante un cambio de una opción por defecto creado a tal fin en el sistema) cualquier aplicación, creada por cualquier fuente. Esto ha permitido la existencia de mercados paralelos al "oficial" de Google o un mercado libre (Free Market) de aplicaciones. Y aquí han empezado una gran parte de los problemas. Pero esto lo veremos más adelante….

Precisamente por el hecho de tener un núcleo basado en Linux, Android aprovecha algunas de las capacidades de este sistema operativo para implementar la seguridad. Sin duda, una de las relevantes es la separación en la ejecución entre las aplicaciones de manera que cada una de ellas se ejecuta con un usuario y grupo diferente. De esta forma se consigue aislar lo que cada aplicación puede hacer en el sistema a su entorno y recursos así como los recursos comunes entre todas ellas pero impidiendo que una aplicación pueda afectar a otra.

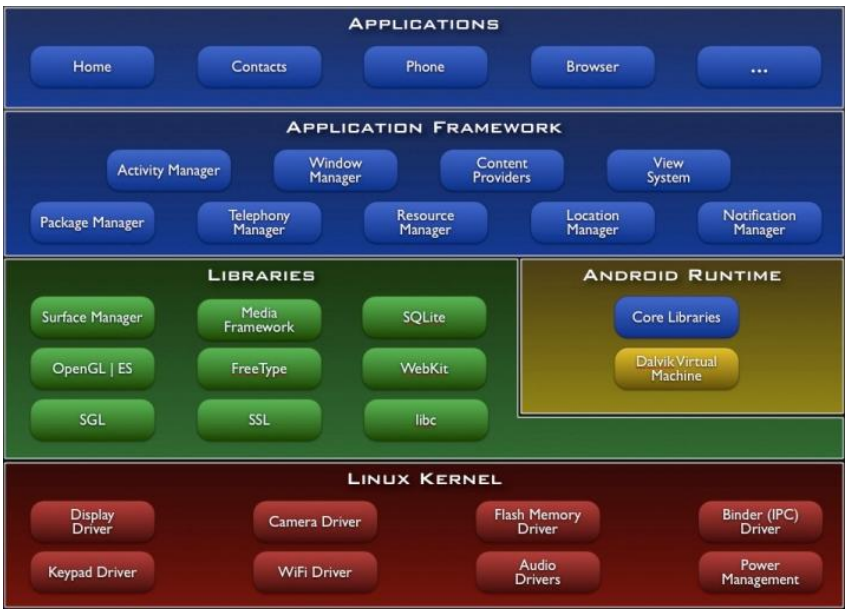

Figura 1: Estructura de capas de Android.

Cada aplicación, por tanto, se ejecutará con un id de usuario e id de grupo distinto impidiendo que una aplicación acceda a información o recursos empleados por otra, más allá de aquellos definidos para un uso común (como la extensión habitual de almacenamiento con la memoria SD de las ranuras de expansión). Además de esta separación entre aplicaciones, el sistema, empleando las propias capacidades de la arquitectura de Linux, aísla las aplicaciones del propio sistema, a modo de *sandboxes*, impidiendo que estas realicen acciones sobre el sistema, los datos o el usuario que puedan ser potencialmente peligrosas.

Como hemos comentado anteriormente Android es un sistema operativo compuesto por un conjunto de capas en forma de una pila que se agrupan principalmente en Sistema Operativo, middleware y aplicaciones. El esquema de estas capas es de la **Figura 1**.

Y la pregunta que surge entonces es pero entonces las aplicaciones ¿cómo realizan operaciones sobre el sistema, léase, hacen llamadas, envían mensajes, acceden a Internet o acceden a la información del GPS integrado?

Aquí entra en juego uno de los puntos más importantes en la seguridad de las aplicaciones, la petición al usuario de los permisos para la aplicación en tiempo de instalación.

2. Permisos de las aplicaciones: manifest.xml

En Android, y de una forma muy particular, los permisos de las aplicaciones son confirmadas con el usuario en el momento de la instalación. Estos permisos no volverán a preguntarse, validarse o serán modificables tras la instalación. La única forma de instalar una aplicación que requiera ciertos permisos (veremos más adelante que permisos son éstos) será aceptarlos todos, no se podrán aceptar algunos y otros no. Además, el usuario no podrá revocar los permisos concedidos a una aplicación: si, por ejemplo, una aplicación puede acceder (porque así se le concedió en la instalación) a la información del GPS, no habrá posibilidad de revocar ese permiso. La única forma de revocarlo será mediante la desinstalación y nueva instalación de la aplicación. Pero habrá que tener en cuenta que la aplicación que requiera de ciertos permisos y estos no sean aceptados por el usuario, no podrá ser instalada.

En este punto es cuando podemos rescatar de nuestros mails de hace años un concepto mediante el cual se troyanizan la mayoría de terminales con Android (permítanme la licencia humorística los nacidos en mi estimada Galicia) y es la del concepto del virus gallego. Para el que no lo recuerde, una imagen vale más que mil palabras:

La cuestión es que la mayoría de troyanizaciones en Android se producen porqué el usuario del terminal, sin prestar atención a los permisos que demandan las aplicaciones aceptan su propia troyanización, así de simple.

Volviendo a la materia (y a la seriedad del tema), las aplicaciones (los referidos archivos .apk) contienen un archivos denominado manifest.xml. Este archivo contendrá información sobre la

aplicación, entre ellos, los permisos que deberá aceptar el usuario en el momento de instalación para el funcionamiento correcto de la aplicación.

Para aquel que tenga curiosidad en echar un vistazo a este fichero en cualquier archivo .apk tendrá que tener presente que este y otros archivos XML contenidos en un APK contienen un XML compilado y, por tanto, no legible directamente con un editor sino que se ha convertido a un formato binario previamente. Esto se hace por una cuestión de optimización del propio archivo APK y su procesado. Existen scripts en Internet y algunas herramientas para convertir este XML binario a un XML en texto.

La estructura de un fichero manifest.xml, en lo que concierne a los permisos, contendrá líneas de este tipo, por ejemplo para monitorizar la recepción de SMS:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.app.myapp" >
    <uses-permission android:name="android.permission.RECEIVE_SMS" />
    ...
</manifest>
```

La pregunta que se estará haciendo el que no conociera esto es ¿entonces es el usuario el que se troyaniza o instala un malware en su terminal y encima éste le avisa y pide permiso para hacer algo malévolo? La respuesta es tristemente afirmativa.

Uno de los casos significativos reciente fue el de una aplicación para Android denominada Steamy Window. En algunos mercados (*markets*) o repositorios de aplicaciones se modificó y colgaron aplicaciones troyanizadas que requerían mayores permisos sobre el terminal (ver Figura 2).

Figura 2: Permisos de la legítima y troyanizada Steamy Windows

La justificación a esto es que para un usuario no avanzado (e incluso para uno que lo sea) resulta complejo discernir entre un permiso “excesivo” o uno “normal”. Las razones son desde el no prestar la atención necesaria a la hora de instalar las aplicaciones hasta el hecho de la gran cantidad de posibles permisos para los que una aplicación puede llegar a demandar concesión.

En una de las últimas versiones del sistema, los permisos incluyen 117 opciones [2]. Algunos podrán ser tan simples como “android.permission.VIBRATE” para poder actuar sobre el sistema de vibrado del terminal, “FLASHLIGHT” para encender la luz flash de la cámara, hasta algunos

que en la propia guía para desarrolladores clasifica de muy peligrosos, como “BRICK”, que permite deshabilitar el terminal y “READ_LOGS” que permite un acceso a bajo nivel de los archivos de registro del sistema, donde puede haber información sensible del usuario.

Las aplicaciones, en muchos casos podrán pedir permiso para acceder a los contactos de la agenda “READ_CONTACTS”, acceso a Internet abriendo sockets de comunicación “INTERNET”, monitorizar la llegada de SMS “RECEIVE_SMS”, enviar SMS “SEND_SMS” y un largo etcétera.

Pasándonos al lado oscuro, pensemos que pueden hacer con estos permisos en alguna de sus combinaciones y a más de uno se le pondrán los pelos de punta: RECEIVE_SMS, RECORD_AUDIO, READ_CONTACTS, READ_SMS, WRITE_SMS, SEND_SMS, CALL_PRIVILEGED, INTERNET, etc.

A que más de uno empieza a pensar si su móvil está grabando el sonido ambiente (“RECORD_AUDIO”) o grabando con la cámara de vídeo/fotos integrada (“CAMERA”, aunque el uso de hardware del terminal requerirá una definición extra de permisos en el archivo manifest.xml) y enviando el audio por Internet (“INTERNET”) o leyendo la información de sus contactos (“READ_CONTACTS”) y enviándola por SMS a otro terminal (“WRITE_SMS”+”SEND_SMS”), una mente perspicaz o retorcida habrá identificado que 117 permisos diferentes y un ordenador de bolsillo será muy versátil o vulnerable.

Seguiremos profundizando en el próximo ISecLab.

[1] http://source.android.com/about/philosophy.html
[2] http://developer.android.com/reference/android/Manifest.permission.html