Universidad de Buenos Aires Facultades de Ciencias Económicas, Ciencias Exactas y Naturales e Ingeniería

Maestría en Seguridad Informática

Tesis

Tema Análisis forense de dispositivos móviles con sistema operativo Android

> Título Desarmando al Androide

Autor: Maximiliano Bendinelli Director de Tesis: Hugo Pagola

> Año 2013 Cohorte 2009

Tabla de Contenidos

Resumenvi
Palabras Clavevii
Introducción1
Capítulo 1: Android
Arquitectura 3
Kernel Linux 4
Librerías5
Runtime de Android5
Capa de Aplicaciones6
Inicialización del dispositivo7
Boot ROM7
Bootloader 8
Kernel9
El proceso Init9
Zygote y Dalvik9
Servicios del Sistema o Servicios9
Boot completo 10
Particionamiento de la memoria de Android10
Filesystem 11
¿Extracción física o lógica?12
Extracción Física13
Extracción Lógica 13
Desafíos 14

Capítulo 2: Extracción de imagen forense de un Android 15
Metodología15
Recovery15
Fastboot17
Creando un firmware forense 17
Desarmando el firmware de Recovery 18
Ramdisk19
Alterando el Ramdisk 19
Preservación de la evidencia27
Capítulo 3: Análisis forense Android 33
Analizando SMS 35
Analizando Contactos 36
Analizando Llamadas 38
Analizando Redes WiFi 39
Analizando Bluetooth 41
Analizando Google Maps 41
Analizando Twitter 43
Analizando Whatsapp 44
Realizando Carving 45
Capítulo 4: ¿Virtualización? 47
Extrayendo las imágenes forenses 48
Ejecutando el entorno virtual 49
Resultado 51
Conclusiones
Anexos
Ejecución del emulador para virtualización54
Bibliografía61

Resumen

La penetración del sistema operativo Android en los usuarios de dispositivos móviles en los últimos años ha sido de un crecimiento exponencial, permitiendo una enorme interacción entre las personas mediante la utilización de distintas tecnologías y aplicaciones que se ofrecen hoy por internet o por redes de telefonía celular. Entre las más populares existen un gran número de mensajeros que han reemplazado a los SMS tradicionales por su bajo costo (o nulo), aquellas aplicaciones que permiten realizar llamadas sin costo como Skype, Line o Viber por mencionar sólo algunas y también la integración con redes sociales.

La creciente penetración en el mercado dispositivos móviles con sistema operativo Android y la falta de herramientas que sean de fácil acceso y que permitan realizar un análisis de estos dispositivos motivaron la realización de este trabajo.

Se comenzara en el primer capítulo conociendo su arquitectura , el kernel con el cual trabaja, su proceso de inicialización , se explicara que es el bootloader entre otras cuestiones técnicas que ayudaran a comprender el funcionamiento de Android por dentro y ayudaran a familiarizar al lector con distintos conceptos del funcionamiento de Android que serán necesarios para comprender el porqué del procedimiento adoptado para avanzar con la investigación que se llevó a cabo en este trabajo de tesis.

En siguiente capítulo se explicara cómo es posible realizar una extracción forense de un dispositivo con Android, se analizaran el uso de la herramientas y se demostrara cono es posible crear un firmware desde cero para poder realizar una copia bit a bit de la memoria del equipo en cuestión y poder extraer su contenido sin contaminar al evidencia allí contenida para un posterior análisis forense.

A lo largo de este trabajo también se demostrara cómo es posible realizar un análisis forense con la copia bit a bit de la memoria que fue extraída con la técnica desarrollada en el capítulo dos (2) y se podrá observar que de dicho análisis se puede recuperar información eliminada, historial de llamados, SMS, datos de geolocalización , historiales de chat y fotos por nombrar son solo algunos de los muchos huellas que dejan las aplicaciones por su utilización.

Como conclusión se intentara realizar una virtualización de un dispositivo con la imagen forense extraída del dispositivo por medio de un emulador del procesador ARM7 incluido dentro de las herramientas de desarrollo de Andorid.

Palabras Clave

Evidencia Digital – Informática Forense – Dispositivos Móviles – Android

Introducción

Los avances tecnológicos en telefonía celular (o móvil) y el crecimiento de Internet revolucionaron las comunicaciones de manera tal que se han convertido en dispositivos imprescindibles para la vida de muchas personas. Los dispositivos móviles se han trasformado de simples teléfonos a dispositivos de comunicación con más capacidades como las de una computadora. Éstos son equipos que poseen las capacidades de un teléfono y una computadora en un mismo aparato.

Estas innovaciones llevaron al desarrollo de teléfonos "inteligentes" con sistemas operativos capaces de realizar diferentes tareas. Una de estas plataformas es el sistema operativo Android: una plataforma de código abierto (open source) para dispositivos móviles desarrollada por Open Handset Alliance [1]. 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. La OHA se fundó el 5 de noviembre de 2007, liderada por Google con otros 34 miembros entre los que se incluían fabricantes de dispositivos móviles, desarrolladores de aplicaciones, algunos operadores de comunicaciones y fabricantes de chips.

En los últimos tiempos se ha generado una interacción de las personas con los dispositivos móviles a niveles impensados, logrando que los usuarios tomen fotografías, filmen videos, realicen intercambio de mensajes (o chat), hagan llamadas telefónicas, utilicen un GPS y puedan geolocalizarse y/o compartir su ubicación o llegar a algún destino por medio de aplicaciones para este fin.. Por esto es que los dispositivos móviles se han transformado en equipos casi indispensables para la vida cotidiana de quienes son usuarios y los ha llevados a estar "hiperconectados".

A partir de este acceso y conexión continua, el análisis de estos dispositivos comienza a ser clave en algunas situaciones como resolver

litigios o investigar un acto criminal. Por esta razón, el trabajo se concentra en analizar e investigar procedimientos que respeten las mejores prácticas para la preservación de la evidencia contenida en estos dispositivos, sin contaminarla.

Recientes estudios realizados por Lockout [2] muestran que al menos el 42% de los dispositivos con Android tenían corriendo alguna clase de malware que le realizaba gastos en la facturación de los usuarios de los dispositivos por medio del envío de SMS de alto costo, generación de *"clicks"* en banners publicitarios, fraude por llamados, son solo algunos de los propósitos de los malwares/virus que infectan a los equipo.

De esta manera, el análisis forense de dispositivos con Android como parte de la informática toma cada vez más importancia y ofrece, al mismo tiempo, muchos desafíos.

En este contexto, el objetivo del siguiente trabajo es realizar una extracción y análisis de información digital con bajo costo y sin alterar la evidencia. La motivación se encuentra debido a las escasas herramientas que le permiten a un investigador poder realizar una copia bit a bit del contenido de la memoria de estos dispositivos para su posterior análisis respetando las mejores prácticas para la preservación de la evidencia.

El crecimiento de dispositivos móviles con Android, su uso para diferentes actividades o el desconocimiento de estos dispositivos por parte de los delincuentes presenta un reto en relación a cómo hacer efectiva la extracción y análisis de datos con propósitos forenses. Para poder realizarla, es importante un buen entendimiento de la plataforma y de las herramientas

A continuación se expondrá un recorrido posible.

Capítulo 1: Android

Android es un sistema operativo basado en Linux y de código abierto , para dispositivos móviles, que inicialmente se desarrolló para arquitectura ARM pero que con el tiempo fue portado a x86. Comenzó a desarrollarse por la *Open Handset Alliance (OHA)* [1] en el año 2005 hasta que fue adquirido por Google en el año 2007. Es esta empresa la que creó el primer dispositivo con Android que recibió el nombre de G1 y al ser de código abierto fue adoptado por una gran cantidad de fabricantes que hoy lideran el mercado. Debido a que cualquier fabricante puede adaptar Android para sus dispositivos, la penetración que tiene en el mercado es enorme y continúa creciendo convirtiéndose en el sistema operativo para dispositivos móviles más utilizado con más de 900 millones de aparatos activos

Arquitectura

Una de las mejores características de Android, es que está diseñado para funcionar en una gran variedad de hardware. Esto se debe a su kernel de Linux el cual ofrece compatibilidad con una amplia variedad de hardware. Por esta razón, muchos fabricantes lo adoptan porque pueden adaptarlo con facilidad al hardware que cada uno desarrolla. Este beneficio del sistema operativo representa un problema para el análisis forense porque la gran diversidad de fabricantes con su respectivo hardware y sus constantes cambios generan modificaciones continuas sobre las distintas plataformas y versiones de Android, aunque existen características que se mantienen La arquitectura está formada por cuatro (4) capas en donde la primera está basada en un Kernel de Linux como se muestra en la figura que está a continuación:



Figura 1.1 Arquitectura de Android

Kernel Linux

El núcleo de Android está formado por el sistema operativo Linux con un Kernel 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. Esta capa proporciona servicios como la seguridad, el manejo de la memoria, el multiproceso, la pila de protocolos y el soporte de drivers para dispositivos. Además, actúa como capa de abstracción entre el hardware y el resto de la pila. Por lo tanto, es la única que es dependiente del hardware.

Librerías

La siguiente capa que se sitúa justo sobre el kernel la componen las bibliotecas nativas de Android, también llamadas librería. Incluye un conjunto de librerías en C/C++ usadas en varios componentes de Android, están compiladas en código nativo del procesador y muchas utilizan proyectos de código abierto. Normalmente están hechas por los fabricantes, quienes también se encarga de instalarlas en los dispositivos. El objetivo de las librerías es proporcionar funcionalidad a las aplicaciones para tareas que se repiten con frecuencia. Entre las librerías incluidas habitualmente se encuentran OpenGL, Bibliotecas multimedia, Webkit, SSL, FreeType, SQLite, entre otras.

Runtime de Android

Como se observa en la figura 1.1, el entorno de ejecución de Android no se considera una capa en sí mismo, dado que también está formado por librerías. Aquí se encuentran las librerías con las funcionalidades habituales de Java así como otras específicas de Android. Está basado en el concepto de máquina virtual utilizado en Java. El componente principal del entorno de ejecución de Android es la máquina virtual denominada **Dalvik.**

Las aplicaciones se codifican en Java y son compiladas en un formato específico para que esta máquina virtual las ejecute. La ventaja es que las aplicaciones se compilan una única vez y de esta forma estarán listas para distribuirse y podrán ser ejecutadas en cualquier dispositivo con Android que disponga de la versión mínima del sistema operativo que requiera la aplicación.

Capa de Aplicaciones

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

- Views: encargada de la parte visual de los componentes.
- Resource Manager: proporciona acceso a recursos que no son en código.
- Activity Manager: manipula el ciclo de vida de las aplicaciones y proporciona un sistema de navegación entre estas.
- Notification Manager: permite a las aplicaciones mostrar alertas personalizadas en la barra de estado.
- Content Providers: es el mecanismo desarrollado para acceder a datos de otras aplicaciones.
- Windows Manager: es el encargado de organizar lo que se mostrará en pantalla. En pocas palabras crea las superficies en la pantalla que a continuación pasarán a ser ocupadas por las actividades.
- Content Provider: es la librería que crea una capa que encapsula los datos que se compartirán entre aplicaciones para controlar cómo acceder a la información.
- Views:en Android, las vistas son los elementos que 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: es donde se encuentran los servicios que notifican al usuario cuando algo requiera su atención y lo realiza mostrando alertas en la barra de estado.
- Package Manager: esta biblioteca permite obtener información sobre los paquetes instalados en el dispositivo Android, así como también de gestionar la instalación de nuevos paquetes.

- Telephony Manager: por medio de esta librería es posible realizar llamadas telefónicas o intercambiar SMS/MMS.
- Resource Manager: es la que permite manejar 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: es por medio de esta librería con la cual el dispositivo puede determinar la posición geográfica mediante GPS o redes disponibles y también permite trabajar con mapas.

Inicialización del dispositivo

Para hacer un análisis forense es indispensable entender cómo inicializa el dispositivo móvil con Android para lo cual se demostrará el proceso de inicialización [3]. Como se mencionó con anterioridad Android está basado en un Kernel de Linux muy similar al que utiliza una computadora de escritorio, pero hay que tener en cuenta que el procesador que utilizan las gran mayoría de los dispositivos móviles es ARM y el hardware es muy distinto también, por lo tanto el Kernel está configurado y adaptado para el hardware del dispositivo.

A continuación se describirá la secuencia de inicialización de un dispositivo con Android.

Boot ROM

Cuando se inicializa el dispositivo y arranca el procesador, este tiene predefinida la dirección de memoria en donde se encuentra el Bootloader y procederá a cargar dicho programa en la RAM del dispositivo para comenzar su ejecución.

Bootloader

Es el primer programa que corre cuando se enciende un dispositivo móvil y es el encargado de gestionar el arranque del sistema. También se asegura de que todos los componentes de hardware estén en correcto funcionamiento y en ese caso se encargará de hacer iniciar Android en el caso de un inicio normal o en el caso de que sea solicitado por el usuario podrá iniciar con Recovery, Fastboot u otros.



Figura 1.2 Proceso de Inicialización de Android [4]

Kernel

El Kernel de Android inicializa de igual manera que cualquier computadora de escritorio con Linux: montará los filesystem necesarios, el cache, la memoria protegida, cargará los drivers y cuando termine todas estas tareas buscará entre los archivos del sistema el programa "init" el cual será el primero en ejecutar del sistema operativo que dará inicio a Android.

El proceso Init

Es el primero de todos los que corre el sistema operativo de base, se puede decir que es el proceso raíz. Se encuentra en la base del system "/" proceso encargado de montar los principales filesystems del sistema operativo para luego dar lugar a la ejecución del archivo "init.rc" el cual se encargará de cuatro clases bien definidas de operaciones: acciones, servicios, comandos y opciones, que servirán de base para el funcionamiento de los siguientes procesos en jerarquía de Andriod.

Al realizar todas estas tareas lo primero que se podrá visualizar en el teléfono es el logo de Android o su correspondiente logo de inicialización.

Zygote y Dalvik

El servicio de Zygote es inicializado desde Init y es el encargado de precargar e inicializar las librerías de clases del núcleo de Android y es la base sobre la cual se inicializan las máquinas virtuales Dalvik.

Dalvik es la máquina virtual que se generará para cada proceso y tiene un funcionamiento muy similar al de las VM de Java.

Servicios del Sistema o Servicios

Una vez que se haya completado la etapa anterior, Zygote se encargará de inicializar los servicios de sistema. Entre los principales servicios se encuentran: telefonía, teclado, batería, alarmas, sensores, administrador de ventanas, entre otros.

Desarmando al Androide

Boot completo

Este es el último paso en la etapa de inicialización, cuando todas las etapas concluyeron se dispara una acción de BROADCAST denominada "ACTION_BOOT_COMPLETED" la cual indica la finalización del proceso.

Particionamiento de la memoria de Android

El particionamiento de la memoria de Android comúnmente se encuentra dividido en:

512	256K	bootloader	u-boot.bin
2048	8M	recovery	recovery.img
18432	8M	boot	boot.img
34816	512M	system	system.img
1083392	256M	cache	cache.img
1607680	512M	userdata	userdata.img

Figura 1.3 Tabla de particiones de Android

Como se observa en la tabla anterior se distinguen seis particiones: bootloader, recovery, boot, system, cache y userdata. Dependiendo del fabricante y el modelo del equipo esta estructura puede tener más particiones pero es la estructura básica sobre la cual se trabajará para el caso de estudio aquí planteado.

- Bootloader como bien su nombre lo indica tiene dentro de ella los programas necesarios para iniciar el bootloader que será el primer programa que cargará el teléfono apenas de inicializa.
- Recovery: es una partición de recuperación en la cual desde el bootloader puede elegirse para iniciar el dispositivo con el fin de realizar tareas de mantenimiento sobre el equipo. Al igual que la partición boot contiene un Kernel y un Ramdisk que actuará de manera distinta al ramdisk de la partición boot ya que sus funciones son limitadas. Cuando se inicia esta partición se puede visualizar una pantalla o consola que

permitirá al usuario realizar alguna actualización sobre el dispositivo o borrar el contenido de las particiones /data o /cache.

- Boot: esta partición contiene el kernel y el ramdisk de Android y es la que utiliza el arranque del teléfono cuando es inicializado en un modo "normal".
- System: dentro de esta partición se encuentra el framework de Android, o sea su entorno gráfico y también se están preinstaladas las aplicaciones del sistema.
- Cache: es la partición donde Android almacena información o datos que el usuario utiliza con más frecuencia, el fin de esta partición es agilizar la carga de aquellos a los que más se accede.
- Userdata: es en esta partición donde se almacenan todos los datos relacionados al usuario, es decir, todo aquello que es creado o modificado por él queda almacenado en esta partición:aplicaciones, contactos, información de llamadas (realizadas, perdidas y recibidas), mensajes, correos electrónicos, fondos de pantalla, fotos, cuentas de aplicaciones como Facebook, Google, Twiteer, etc.

Este último, desde el punto de vista forense, es el más rico en información y aquel desde donde podrá surgir toda la actividad del usuario con el dispositivo móvil.

Filesystem

Por medio de distintos Filesystem, Android organizará la información que contenga los dispositivos. Al estar basado en Linux soporta varios tipos de filesystems como EXT3, FAT, YAFFS2 y EXT4. El mayor reto desde el punto de vista forense son los Filesystems con YAFFS2 ya que no todas las herramientas forenses soportan este tipo de Filesystem. Lo positivo, es que con la evolución de Android cada vez son más los dispositivos que soportan EXT4 que sí está soportado por la mayoría de las herramientas.

Dentro de la estructura de Filesystems de Android se pueden distinguir las diferentes particiones con distintos Filesystems. A continuación se muestran dos (2) capturas de distintos dispositivos donde se observa con claridad que la primera utiliza yaffs2 para sus particiones de /system , /data y /cache y la segunda imagen se muestra que las principales particiones utilizan ext4, salvo la tarjeta micro-sd que por cuestiones de compatibilidad con otros sistemas operativos suele utilizar Fat o ExFat.

📼 C:\Windows\system32\cmd.exe - adb.exe -e shell	
C:\Users\maxi\Desktop\adt-bundle-windows-x86_64-20130514\sdk\platform-tools>adb. exe -e shell root@android:/ # mount mount rootfs / rootfs ro 0 0	
tmpfs /dev tmpfs rw,nosuid,mode=755 0 0 devpts /dev/pts devpts rw,mode=600 0 0 proc /proc proc rw 0 0 sysfs /sys sysfs rw 0 0 none /acct cgroup rw,cpuacct 0 0	
tmpfs /mnt/secure tmpfs rw,mode=700 0 0 tmpfs /mnt/asec tmpfs rw,mode=755,gid=1000 0 0 tmpfs /mnt/obb tmpfs rw,mode=755,gid=1000 0 0 none /dev/cpuctl cgroup rw,cpu 0 0 /dev/block/mtdblock0 /system yaffs2 ro 0 0	
/dev/block/mtdblockl /data garsz rw,nosuld,nodev 0 0 /dev/block/mtdblock2 /cache yaffs2 rw,nosuld,nodev 0 0 root@android:/ #	-

Figura 1.4 Particiones con Filesystem Yaffs2



Figura 1.5 Particiones con Filesystem EXT4

¿Extracción física o lógica?

Al momento de extraer información de un dispositivo se pueden establecer dos metodologías que se definen como extracción física o

extracción lógica. Estas alternativas dependiendo del alcance de la investigación brindaran al perito la información que requiera analizar del dispositivo. A continuación se hará una breve descripción de las bondades que ofrecen cada una de estas alternativas

Extracción Física: este tipo de extracción o copia forense permitirá a quien deba examinar el dispositivo cavar en lo profundo del Filesystem del sistema operativo, permitiendo el acceso a información *"invisible"* como archivos borrados o la información contenida en los *"slack spaces"*. Para poder llevar a cabo una extracción de este tipo se deberá tener un almacenamiento igual o mayor al de origen ya que una copia de estas características tendrá como resultado un archivo (o más) que totalizará el tamaño del disco origen, este tipo de copias también insumen una gran cantidad de tiempo ya que el copiado es completo y suele ocupar varios gigabytes.

Extracción Lógica: esta operación permitirá al investigador extraer información del dispositivo que está accesible desde el Filesystem, esto incluye todos los archivos que estén disponibles ya sean fotos, bases de datos, archivos de texto, correos electrónicos, videos, mp3, entre otros. Cabe destacar que gran parte de la información que maneja el sistema operativo Android la almacena en bases de datos del tipo SQLlite, por lo tanto acceder a estos archivos brindará al investigador información relativa a contactos, sms, mms, correos electrónicos, historial de llamados, configuraciones del dispositivo, entre otras. A diferencia de una extracción física el tiempo que insume es notablemente menor ya que solo se trabajará sobre los archivos que sean necesarios para la investigación y no la totalidad del dispositivo.

En los próximos capítulos se demostrará cómo toda esta información sale a la luz una vez que los datos son extraídos del dispositivo.

Desafíos

El principal desafío que se enfrenta en este trabajo es realizar una extracción forense (o copia bit a bit) de las particiones de la memoria que están en el dispositivo móvil y contienen información sensible del usuario sin contaminar la evidencia. Pero ¿cómo es posible extraer la información que el dispositivo posee si al iniciar el sistema operativo las funciones inherentes a su funcionamiento montan y escriben información sobre las particiones? No es intención de este trabajo hacer una simple copia del contenido que tienen las particiones, tampoco se ve viable la opción de hacer un "chip-off" y retirar la memoria del equipo. Por el contrario, se propone hacer una copia bit a bit de la partición userdata que es la que guarda información de la actividad de los usuarios de los dispositivos y que puede ser utilizada con fines legales a los efectos de resolver un caso. La ventaja que ofrece poder realizar una copia bit a bit es que se puede extraer toda la información de la partición, aquella que se ve por el usuario y también la que no se ve, como información borrada o información que se encuentra en lo que se denomina en informática forense "slack space" (espacio perdido).

Queda fuera del alcance de este trabajo realizar una imagen y un análisis forense sobre la tarjeta de almacenamiento externa micro-sd ya que puede ser realizada y analizada siguiendo los mismos pasos que para cualquier dispositivo interno o externo de una PC.

Capítulo 2: Extracción de imagen forense de un Android

Metodología

La metodología que se propone en este trabajo consiste en la creación de una imagen o firmware el cual será instalado en la partición de Recovery del Smartphone a ser peritado. Tiene como fin configurar el dispositivo de manera tal que inicie en modo de Recovery con un programa de inicialización realizado específicamente para realizar imágenes forenses del dispositivo teniendo en cuenta las mejores prácticas para realizar estas tareas.

El firmware desarrollado no montará ni realizará ninguna tarea que escriba o modifique datos sobre las particiones que deberán ser preservadas para un posterior análisis. Lo que aquí se propone es comparable con la tarea de iniciar una computadora personal con una unidad externa con un sistema operativo con fines forenses como por ejemplo el HELIX o CAINE (entre otros) para ordenadores.

Recovery

Los dispositivos con sistema operativo Android pueden inicializar no solo en el modo normal sino que también se le puede ordenar al Bootloader que inicialice el dispositivo en modo Fastboot, modo Flash (dependiendo del dispositivo) y modo Recovery.

El modo Recovery permite inicializar el dispositivo utilizando la información contenida en la partición de Recovery, la cual contiene el programa de inicialización (INIT) con la base del sistema operativo que permite al usuario realizar tareas de mantenimiento en el dispositivo, entre las que se encuentran actualizar el dispositivo y borrar las particiones /data y /cache como se muestra en la figura a continuación.



Android system recovery utility

apply update from sdcard wipe data/factory reset wipe cache partition



Figura 2.1 Captura de pantalla de un inicio con Recovery

Fastboot

Fastboot es un protocolo que es utilizado para actualizar los filesystems de dispositivos con Android desde una computadora por USB. A través suyo, se puede grabar en las particiones imágenes de firmware sin que estén firmadas. El principal problema que se enfrenta desde el punto de vista forense es que no todos los fabricantes incluyen fastboot en sus dispositivos por y esto es debido a que este no es un único protocolo que permite realizar estas tareas de actualización, sin embargo para este caso de estudio es posible utilizarlo.

Estructura de un firmware

La estructura de un firmware está conformada por un "boot header" (o encabezado de boot), un kernel, un ramdisk y un carga de "second stage", según surge de los paquetes de la SDK de Android y lo documenta y lo explica Android-dls [6], como se puede ver a continuación:

```
+----+
| boot header | 1 pagina
+----+
| kernel | n paginas
+----+
| ramdisk | m paginas
+----+
| second stage | o paginas
+----+
n = (kernel_tamaño + page_tamaño - 1) / pagina_tamaño
m = (ramdisk_tamaño + pagina_tamaño - 1) / pagina_tamaño
o = (second_tamaño + pagina_tamaño - 1) / pagina_tamaño
```

Figura 2.1.1 Estructura de un Firmware

Creando un firmware forense

Para dar comienzo a este trabajo se procedió a utilizar la imagen de un firmware oficial de Recovery del dispositivo a analizar. Para lograr esta tarea se procede a *"desarmar"* el firmware separando su Kernel del Ramdisk para alterar el funcionamiento original del ramdisk evitando que este último, por medio del proceso init, monte las particiones que se desean extraer del sistema operativo. Sin que se haya montado la partición *"userdata"* y *"cache"* y se procede a adicionarle al proceso INIT el servicio de *"Android Debug Bridge"* para que se ejecute con los permisos de *"root"* (con el fin de acceder vía línea de comandos al sistema operativo cuando este inicializado), para luego rearmar la imagen de recovery y guardarla en el teléfono en la partición de recovery utilizando fastboot. Una vez realizado este procedimiento se iniciará el dispositivo en modo recovery (con la imagen realizada para este fin) y se ingresará al dispositivo vía una consola provista por *"Android Debug Bridge"* [5].

Desarmando el firmware de Recovery

Para desarmar la imagen de Recovery del dispositivo se procede a descargar del sitio clockworkmod [6] el firmware necesario para realizar esta tarea, y siguiendo los pasos descriptos en android-dls [7] se utiliza un script escrito por William Enck [8] denominado *"split_boot_img.pl"* con el fin de dividir el firmware en dos archivos distintos, en uno estará localizado el Kernel y el otro será el Ramdisk del Recovery.

```
root@bt:~/tesis# ../split_boot_img.pl recovery.img
Page size: 2048 (0x0000800)
Kernel size: 2823432 (0x002b1508)
Ramdisk size: 451086 (0x0006e20e)
Second size: 0 (0x0000000)
Board name:
Command line: androidboot.carrier=wifi-only product_type=w
Writing recovery.img-kernel ... complete.
Writing recovery.img-ramdisk.gz ... complete.
root@bt:~/tesis#
```

Figura 2.2 Partiendo la imagen de Recovery

Ramdisk

Alterando el Ramdisk

Con el fin de cambiar el comportamiento original del Recovery para que realice las tareas forenses, se procede a alterar el comportamiento de los programas que se ejecutan desde el ramdisk. En primer lugar, se procederá a descomprimirlo, como se muestra en la figura a continuación:

```
root@bt:~/tesis/ramdisk# gunzip -c ../recovery.img-ramdisk.gz | cpio
-i --verbose
data
default.prop
dev
etc
etc/recovery.fstab
init
init.rc
proc
res
res/images
res/images/icon error.png
res/images/icon installing.png
res/images/indeterminate1.png
res/images/indeterminate2.png
res/images/indeterminate3.png
res/images/indeterminate4.png
res/images/indeterminate5.png
res/images/indeterminate6.png
res/images/progress empty.png
res/images/progress fill.png
res/keys
sbin
sbin/adbd
sbin/recovery
sbin/ueventd
sys
system
tmp
ueventd.goldfish.rc
ueventd.rc
ueventd.stingray.rc
1275 blocks
root@bt:~/tesis/ramdisk#
```

Figura 2.3 Descomprimiendo el Ramdisk de Recovery

Luego, se procede a analizar el contenido de init.rc para conocer cuáles son los servicios que este proceso inicia y cuál es el comportamiento del Ramdisk. Se puede observar lo siguiente:

```
root@bt:~/tesis/ramdisk# cat init.rc
on early-init
   start ueventd
on init
   export PATH /sbin
   export ANDROID ROOT /system
   export ANDROID DATA /data
    export EXTERNAL STORAGE /sdcard
   symlink /system/etc /etc
   mkdir /sdcard
   mkdir /system
   mkdir /data
   mkdir /cache
   mount /tmp /tmp tmpfs
on boot
   ifup lo
   hostname localhost
    domainname localdomain
    class start default
service ueventd /sbin/ueventd
   critical
service recovery /sbin/recovery
service adbd /sbin/adbd recovery
   disabled
on property:persist.service.adb.enable=1
   start adbd
on property:persist.service.adb.enable=0
   stop adbd
root@bt:~/tesis/ramdisk#
```

```
Figura 2.4 Estructura del archivo init.rc
```

Se puede observar que entre las tareas más relevantes que realiza el *init.rc* se encuentra la de crear la estructura de directorios sobre la base o la raíz del sistema de archivos (ver comandos mkdir). Luego entre los servicios más importantes que inicia es el GUI del Recovery en la línea *"service recovery /sbin/recovery"* que se corresponde como resultado a la figura 2.1 y al final de este archivo se distingue que dependiendo el valor de la propiedad *persist.service.adb.enable* iniciará o no el servicio que se corresponde con *"Android Debug Bridge"*, proceso necesario para realizar las tareas propuestas en este proyecto ya que proveerá al analista forense de un acceso por consola al dispositivo (también conocido como *shell*). El valor que tomará la propiedad *persist.service.adb.enable* estará configurado desde el archivo default.prop que se encuentra en la raíz del sistema de archivos. Como se muestra a continuación el valor que tiene esta propiedad (o variable) es 0 por lo que se interpreta desactivado.

```
root@bt:~/tesis/ramdisk# head default.prop
#
# ADDITIONAL_DEFAULT_PROPERTIES
#
ro.secure=1
ro.allow.mock.location=0
ro.debuggable=0
persist.service.adb.enable=0
# begin build properties
# autogenerated by buildinfo.sh
ro.build.id=H.6.2-21
root@bt:~/tesis/ramdisk#
```

Figura 2.5 Variables configuradas desde el archivo default.prop

Con el fin de lograr que el servicio de "*Android Debug Bridge*" inicie y poder acceder vía consola de comandos al dispositivo, se procede a alterar el valor del contenido del archivo "default.prop" y, para lograr que el acceso sea con *root* (o súper usuario), se cambiará el valor de la variable *ro.secure* a 0 ya que para poder realizar una imagen forense es necesario acceder a los dispositivos de modo directo (o *raw*) y solo es posible como *root*.

Desarmando al Androide

Teniendo alterados estos valores, para continuar con el proceso de adaptación del Ramdisk forense, se procede a insertar algunos comandos básicos para realizar la copia bit a bit. Como se puede observar en la imagen 2.3 del listado de archivos Ramdisk básico carece de comandos (o binarios de programas) y a los fines de agilizar las tareas, se procede a extraer de un firmware de Android los binarios necesarios y son copiados en el subdirectorio /sbin del Ramdisk que se está adaptando. Entre las herramientas que se utilizan y se agregan al Ramdisk para la creación de la imagen forense se encuentran: dcfldd, dd y dc3dd. Una vez realizado este paso se procede a comprimir nuevamente la estructura de Ramdisk.

Figura 2.6 Rearmando el archivo Ramdisk

Habiendo logrado re empaquetar el archivo Ramdisk se procede a realizar pruebas en el emulador de Android para analizar su comportamiento y corroborar si está funcionando de manera esperada antes de ser grabado en la memoria del dispositivo móvil.



Figura 2.7 Probando Firmware Forense en el Emulador

Una vez que se ha verificado la iniciación del nuevo firmware en un entorno virtualizado para dicho fin, se procede a crear un firmware el cual contendrá el kernel del y el ramdisk creados.

```
mkbootimg --cmdline 'no_console_suspend=1 console=null' --kernel
recovery-s3.img-kernel --ramdisk newramdisk.cpio.gz -o recovery.img
```

Figura 2.7.1 Probando Firmware Forense en el Emulador

Una vez realizado el empaquetado del Kernel con el Ramdisk se procede a salvar la imagen en un dispositivo real, en este caso se utilizará una *Tablet* Marca Motorola modelo Xoom. Utilizando la herramienta Fastboot se guarda el firmware en la Tablet en la partición de Recovery y se procede a inicializar el equipo con la imagen de Recovery creada según se describió en los pasos anteriores.



Figura 2.8 Grabando el Firmware Forense usando fastboot

A continuación se puede apreciar la vista del dispositivo con el firmware siendo grabado.



Figura 2.9 Grabando el Firmware Forense en el dispositivo

Para verificar el correcto funcionamiento del firmware creado y salvado, se inicializa la Tablet en modo Recovery y se puede observar la siguiente pantalla en el dispositivo.



Figura 2.10 Firmware creado funcionando sobre la Tablet

Habiendo inicializado la Tablet con el firmware creado para realizar las imágenes forenses de las particiones, se procede a ingresar a la consola de comandos utilizando el ADB.

```
root@bt:~# /root/adt-bundle-linux-x86 64-20130219/sdk/platform-
tools/adb devices
List of devices attached
037c70884080e217 recovery
root@bt:~# /root/adt-bundle-linux-x86 64-20130219/sdk/platform-
tools/adb shell
~ # uname -a
Linux localhost 2.6.36.3 #1 SMP PREEMPT Wed Mar 23 12:15:44 CDT 2011
armv7l GNU/Linux
~ # id
uid=0(root) gid=0(root)
~ # cat /proc/cpuinfo
Processor : ARMv7 Processor rev 0 (v71)
processor
              : 0
              : 999.42
BogoMIPS
processor
              : 1
               : 999.42
BogoMIPS
```

Features: swp half thumb fastmult vfp edsp vfpv3 vfpv3d16CPU implementer: 0x41CPU architecture:7CPU variant: 0x1CPU part: 0xc09CPU revision: 0Hardware: stingrayRevision: 8300Serial: 037c70884080e217

Figura 2.11 Ingresando vía consola al dispositivo

Se comprueban las particiones del dispositivo y se verifican que no hayan sido montadas, lo que garantiza que **no se altera en ningún momento la integridad de la información** en dichas particiones.

```
~ # cat /proc/partitions
major minor #blocks name
 179 0 31162368 mmcblk0
            1 3072 mmcblk0p1
2 2048 mmcblk0p2
 179
 179
                      2048 mmcblk0p3
 179
            3
                     4096 mmcblk0p4
2048 mmcblk0p5
            4
 179
          5 2048 mmcblk0p5
6 12288 mmcblk0p6
7 8192 mmcblk0p7
 179
 179
 179
                  262144 mmcblk0p8
173056 mmcblk0p9
 259
           0

        259
        1
        173056
        mmcblk0p9

        259
        2
        30663168
        mmcblk0p10

~ # cat /etc/fstab
/dev/block/mmcblk1p9 /cache ext4 rw
/dev/block/mmcblk1p10 /data ext4 rw
/dev/block/mmcblk1p8 /system ext4 rw
/dev/block/mmcblk0p1 /sdcard vfat rw
~ # mount
rootfs on / type rootfs (rw)
tmpfs on /dev type tmpfs (rw,relatime,mode=755)
devpts on /dev/pts type devpts (rw,relatime,mode=600)
proc on /proc type proc (rw, relatime)
sysfs on /sys type sysfs (rw,relatime)
```

Figura 2.12 Verificación de particiones

Se procede a verificar la existencia y funcionamiento de los programas instalados para realizar la imagen forense, en particular dcfldd y dc3dd.

```
~ # ls -la /sbin/dcfldd /sbin/dc3dd
-rwxr-xr-x 1 root root 1021168 Aug 21 02:04 /sbin/dc3dd
-rwxr-xr-x 1 root root 804678 Aug 21 02:04
/sbin/dcfldd
~ # /sbin/dcfldd --version
dcfldd (dcfldd) 1.3.4-1
Written by: dcfldd by Nicholas Harbour, GNU dd by Paul Rubin, David
MacKenzie and Stuart Kemp.
Copyright (C) 1985-2006 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There
is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR
PURPOSE.
~ # /sbin/dc3dd --version
dc3dd (dc3dd) 7.1.614
Copyright (C) 2008 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Written by Paul Rubin, David MacKenzie, Stuart Kemp,
Jesse Kornblum, Andrew Medico, and Richard Cordovano.
```

Figura 2.13 Verificación de Binarios para la adquisición forense

Al llegar a este punto, se pudo crear un firmware (o imagen) de Recovery el cual una vez instalado en el dispositivo e inicializado en este modo, permitió el acceso al equipo y a ejecutar distintos comandos. En los siguientes capítulos se demostrarán distintas maneras de extraer la información contenida en el dispositivo siguiendo las mejores prácticas para la extracción de evidencia digital de un medio de almacenamiento.

Preservación de la evidencia

Previo a realizar una imagen forense de las particiones que son de interés para este trabajo (*"userdata" y "cache"*) se debe elegir el método y el

medio por el cual se almacenará o guardaráa las imágenes forenses creadas que son resultantes de la extracción de la información. Entre las opciones que se presentan existen la de almacenar la copia forense en una memoria micro-sd o trasferir la copia forense por el cable USB hacia una computadora.

Ambas opciones son válidas para resguardar y preservar las imágenes, pero presentan pros y contras. La principal ventaja que posee realizar una copia bit a bit sobre una memoria micro-sd es la velocidad de transferencia, pero debe disponerse de una memoria con la capacidad suficiente para realizar dicha tarea y con el crecimiento en almacenamiento de los teléfonos puede ser en algún caso difícil tener disponible una memoria con la capacidad requerida. La trasferencia por cable USB es más lenta pero el almacenamiento en las computadoras personales suele estar al alcance del investigador.

A continuación se demostrará las dos maneras aquí planteadas de extraer la evidencia y se podrá verificar mediante un hash del tipo MD5 que las copias, sin importar el método, son exactamente iguales.

Se procede a extraer una partición del dispositivo y guardarla en una computadora conectada vía USB y se toma el tiempo que se tarda en realizar la tarea.

root@b	:~/adt-bundle-linux-x86_64-20130219/sdk/platform-tools# t	lme
./adb	shell "dcfldd if=/dev/block/mmcblk0p9 hashlog=/tmp/hash.1	log
errlog:	/tmp/error.log 2>/tmp/stderr.log" sed 's/\r\$//' > mmcblk0	p9
<mark>real</mark> user sys	3m1.130s Om0.600s Om7.552s	

Figura 2.14 Realizando una copia bit a bit por medio de una conexión USB

El tiempo que llevó realizar este procedimiento fue de 3 minutos y 1,1 segundos para copiar una partición de 170 Megabytes por USB. Acto

Desarmando al Androide

seguido se procede a verificar el md5 de la partición del dispositivo y luego se verifica la el mismo hash de la copia.



Figura 2.15 Verificando la integridad de la copia.

Como se observa en recuadro anterior (Figura 2.15), la partición del dispositivo, así como también la copia realizada, son idénticas obteniéndose como resultado de esta prueba el mismo hash md5 el cual fue "cf1404228092bb1ab5300128d5bef774".-

Al chequear el contenido del archivo con el comando de Linux *"file*" se verifica que dicho archivo contiene una partición con un filesystem ext4.

```
# file mmcblk0p9
mmcblk0p9: Linux rev 1.0 ext4 filesystem data, UUID=57f8f4bc-abf4-
655f-bf67-946fc0f9f25b (extents) (large files)
```

Figura 2.16 Verificando el contenido de la copia.

A continuación se procederá a realizar la misma tarea sobre una memora micro-sd. Primero se identifican las particiones de la memoria, luego se monta y se realiza la copia bit a bit de una partición. Se muestra en los siguientes párrafos el procedimiento realizado:

~ # cat major m	t /proc/pa ninor #bl	artitions locks nar	ne
179	0	7761920	mmcb1k0
179	1	7757824	mmcblk0p1
179	8	31162368	mmcblk1
179	9	3072	mmcblk1p
179	10	2048	mmcblk1p
179	11	2048	mmcblk1p

~ #			
259	2	30663168	mmcblk1p10
259	1	173056	mmcblk1p9
259	0	262144	mmcblk1p8
179	15	8192	mmcblk1p7
179	14	12288	mmcblk1p6
179	13	2048	mmcblk1p5
179	12	4096	mmcblk1p4

Figura 2.17 Tabla de particiones

Como se observa en la figura 2.17, se identifican claramente dos dispositivos de almacenamiento, la memoria micro-sd identificada como mmcblk0, y la memoria interna del equipo de estudio que es mmcblk1.En el siguiente paso se montará la memoria y se verificará que haya montado correctamente y que exista espacio disponible para almacenar la imagen forense.

tmpfs on /dev type tmp	ofs (rw,nosu	uid,relat	cime,mode=	755)		
devpts on /dev/pts typ	e devpts (1	w,relat:	ime, mode=60)))		
proc on /proc type pro	oc (rw,relat	ime)				
sysfs on /sys type sys	sfs (rw,rela	atime)				
/dev/block/mmcblk0p1	on	/sc	lcard	tvr	e	vfat
act, brock, mucbrkopr				-11		
(rw,relatime,fmask=000	0,dmask=000	0,allow	_utime=0022	2, code	epage=cp4	437,i
(rw,relatime,fmask=000 ocharset=iso8859-1,shc	00,dmask=000 ortname=mixe	0,allow_	_utime=0022	2, code co)	epage=cp4	437,i
<pre>(rw,relatime,fmask=000 ocharset=iso8859-1,shc ~ # df -h</pre>	00,dmask=000 prtname=mixe)0,allow_ ed,errors	_utime=0022 s=remount-:	2, code ro)	epage=cp4	137,i
<pre>(rw,relatime,fmask=000 ocharset=iso8859-1,shc ~ # df -h Filesystem</pre>	00,dmask=000 ortname=mixe Size	00,allow_ ed,errors	_utime=0022 s=remount-r Available	2,code ro) Use%	epage=cp4 Mounted	437,i on
<pre>(rw,relatime,fmask=000 ocharset=iso8859-1,sho ~ # df -h Filesystem tmpfs</pre>	0,dmask=000 ortname=mixe Size 359.4M	00,allow_ ed,errors Used 32.0K	utime=0022 s=remount- Available 359.4M	2,code ro) Use% 0%	epage=cp4 Mounted /dev	137,i on

Figura 2.18 Montando la memoria externa

Los siguientes comandos se ejecutan con el fin de almacenar la imagen forense en la tarjeta micro-sd al mismo tiempo que se calcula su hash md5 y con fines puramente comparativos se calcula el tiempo que tarda en realizar las tareas. Cabe aclarar que la herramienta dcfldd que se introdujo dentro del firmware es comúnmente utilizada en tareas de adquisición forense con discos rígidos de computadoras, por lo tanto su utilización es la misma que a la de una herramienta con una madurez reconocida.



Figura 2.19 Realizando una copia bit a bit hacia una memoria externa

Del resultado obtenido, se observa el resultado md5 el cual dio **3026dc171cff839cb54765ef79babd36** y que luego con los fines de verificarlo se calculó el md5 de la partición la cual arrojo el mismo resultado y que el tiempo en realizar la tarea que vía USB demoró 3 minutos con una memoria micro-sd demoró apenas 13.61 segundos.

Se procede a insertar la memoria micro-sd en una computadora y se verifica la existencia del archivo contenedor de la imagen forense.

臰 Dispositivo de almace	enamiento digital seguro (G:)			_	
🕞 🖓 🖉 🗸 Equipo	o \checkmark Dispositivo de almacenamiento digital seguro (G:) \checkmark	▼ 65	Buscar Dispositivo de a	lmacenamiento	2
Organizar 🔻 F Abrir	Grabar Nueva carpeta			📰 👻 🛄	0
☆ Favoritos	Nombre	Fecha de modific 👻	Тіро	Tamaño	
🗼 Descargas	mmcblk1p9.dd	05/09/2013 02:54 a	Archivo DD	173.056 KB	
Escritorio	💽 system.img	21/08/2013 02:31 a	Archivo de imagen	262.144 KB	
Sitios recientes	isys.img	17/07/2013 11:16 p	Archivo de imagen	8.192 KB	
Cropbox	e boot.img	17/07/2013 11:12 p	Archivo de imagen	262.144 KB	
Google Drive	🔚 userdata.img.gz	13/06/2013 09:17 p	WinRAR archive	201.600 KB	
	ache.img.gz	13/06/2013 09:00 p	WinRAR archive	3.505 KB	
🥞 Bibliotecas	img recovery.img	13/06/2013 08:36 a	Archivo de imagen	3.928 KB	
Documentos	🚬 🗎 data.img.gz	09/06/2013 05:57 p	WinRAR archive	352.317 KB	
Imágenes	■ □		· · · · · · · · · · · · · · · · · · ·	4.100	
mmcblk1p9 Archivo DD	9.dd Fecha de modificación: 05/09/2013 02:54 a.m. Tamaño: 169 MB	Fecha de creación: 05/09/201	302:54 a.m.		

Figura 2.20 Archivo contenedor de la imagen forense

	Al abrir	el	mencionado	archivo	con	la	herramienta	FTK	Imager,	se
puede	ver el co	onte	enido de la pa	artición re	esgua	ard	ada.			

🔍 AccessData FTK Imager 3.1.1.8						
<u>File View M</u> ode <u>H</u> elp						
	s 200 200 ₽					
Evidence Tree	File List					×
E-g mmcbk1p9.dd	Name	Size	Туре	Date Modified	1	
B In NONAME [ext4]] [root]	4	Directory	03/09/2013 04:		
[unallocated space]	[unallocated space]	0	Unallocated Sp			
	bad blocks	0	Filesystem Met			
	block bitmap	8	Filesystem Met			
	boot record	1	Filesystem Met			
	group descriptor table	8	Filesystem Met			
	inode bitmap	8	Filesystem Met			
	inode table	2.704	Pliesystem Met Pequiar File			
	superblock	8	Filesystem Met			
			,.			
Custom Content Sources ×	09b66b0 F2 97 27 BC	46 79 82 B6-6E	3D B1 CB 6B B	3 8D C8 ò · '%Fy	¶n=±Ëk³ È	<u> </u>
Evidence:File System Path File Options	09b66c0 42 3C B3 8C	68 A4 F2 82-57	23 OB BD D1 9	A 6A 3C B< ³ h×d	ò-W≢-HaNi-j< an õi muos	_
	09b66e0 0C E5 5B 98	47 D6 37 3C-4A	19 B7 CB 63 6	B C2 CD -å[-GÖ]	10 104 104275 7 <j -="" eckâí<="" th=""><th></th></j>	
	09b66f0 F1 DC E8 15	B1 11 E2 1F-68	42 12 63 B6 71	E 18 E5 ñÜè·±·á	à·hB·c¶∼·å	
	09b6700 C6 8D A9 94	27 9C 5F 8E-F4	79 E2 43 B5 D	27DB8 E.@.'.	·ôyâCµÒ},	
	09b6710 D8 A9 1D 2F	F2 54 29 8C-C3	C2 5C 57 8C F	A 40 2E 200 ⋅/òT)	·ĂĂ\W·ú@.	
	0906720 EB AC 47 A8	12 45 BD 18-42 CO 37 ED 85-8D	EC 6E 24 17 B	4 DC 48 e→G ·E*	5-0 1-V-0H	
	09b6740 65 B9 51 DE	23 B9 81 C8-C3	BD E8 87 56 F	4 24 C6 e QP#	·ÈÃ‰è ·VôsE	
	09b6750 8B 1D CE 5C	FA 5B 54 3E-45	5A C5 06 9D E	8 63 88 - Ĩ\ú[]	r>EZÅ · ·èc ·	
	09b6760 5E E8 5A CA	4C 28 3B 25-EE	D8 57 1D 1D 6	B DD A2 ^èZÊL(;	\$îØW··kÝ≎	
	09b6770 C7 A1 EC CE	65 A7 C5 53-E6	6E B9 4B D2 00	C 18 78 C;iIeSF	ASen*KO··x	
	09b6790 A5 4B 44 2E	B5 C8 8C E4-2F	1A 3E 24 50 1	0 99 57 ¥KD.uÈ	·ä/·>sP··W	
	09b67a0 C8 B9 3C 45	C9 2A OC 58-69	A9 F8 11 E8 F	7 4A 6B ȳ <eé*< th=""><th>Xi©ø è+Jk</th><th></th></eé*<>	Xi©ø è+Jk	
	09b67b0 AE CF 99 33	DA 85 FA 07-E6	FF 43 8B F6 3	6 27 50 ®Ï·3Ú·ú	i-æÿC·ö6'₽	
Number of the Second States Second	09b67c0 6E 7A ED 47	F8 66 DA 62-CA 88 AC B3 21-F6	55 61 3C C3 C	6 DC 04 nziGøft	JDEUa <aeu Lö ji ééji</aeu 	
Tiene Fort Remove Keunive Bill Fleare ruege	09b67e0 1B 82 F5 B1	F4 E6 36 B1-EF	B6 A6 A9 59 F	4 47 0C ··õ±ôæ6	5±ï¶¦@YôG·	-1
Properties Hex Value Interpreter Custom Content Sources	Cursor pos = 170; dus = 0; lo	g sec = 0		· · ·		
mmcblk1p9.dd/NONAME [ext4]						

Figura 2.21 Abriendo la imagen forense con FTK Imager

Hasta aquí se ha demostrado que es posible mediante distintas herramientas de código abierto, realizar una imagen forense de un dispositivo con Android sin alterar la evidencia que contiene, manteniendo la integridad de la información en todos sus pasos y permitiendo que las copias realizadas puedan ser interpretadas por herramientas forenses para su posterior análisis.

Capítulo 3: Análisis forense Android

Para realizar un análisis forense de la copia bit a bit de la memoria interna en la cual se aloja el Filesystem del dispositivo con Android, se puede seguir las mismas prácticas forenses que para la copia de una imagen bit a bit del disco rígido de una computadora o notebook.

Hasta aquí se ha logrado resguardar y preservar la evidencia extraída de un dispositivo con Android así como también se obtuvo un hash de validación el cual le permitirá en un futuro al investigador validar la integridad de la información extraída.

Si bien objetivo de este trabajo consiste en lograr una adquisición forense de un dispositivo con Android, en este capítulo se intentará demostrar cómo con herramientas forenses se puede obtener información de la imagen (o copia forense) creada siguiendo los pasos y utilizando las herramientas utilizados en los capítulos anteriores.

Para continuar se procede a realizar una copia forense de la partición "/data" (o también conocida como "userdata"), que como se mencionó anteriormente aquí, se alojan los archivos que utilizan las distintas aplicaciones instaladas en el dispositivo como la información del usuario, contactos, mensajes, configuraciones, aplicaciones instaladas, fotos, videos, música y variada información que puede ser de utilidad para una investigación. También podrá surgir si el dispositivo a ser investigado fue víctima de alguna intrusión o malware el cual realizó actividades sin el conocimiento del usuario del dispositivo. Es por esto que a lo largo de este capítulo se demostrarán distintas formas de extraer información útil de la imagen forense realizada.

Como primer paso se procede a realizar una extracción física de una partición (unserdata) la memoria del dispositivo, para este caso de estudio se utilizó un teléfono celular Samsung i9300 (también conocido como S3). Una vez instalado el firmware creado en el Capítulo 2, se procede a ingresar al dispositivo vía *ADB* y por medio de los siguientes comandos que son

ejecutados desde una consola del dispositivo se logra la copia bit a bit y se la almacena en una memoria micro-sd:



Figura 3.1 Abriendo la imagen forense con FTK Imager

Habiendo realizado la extracción física del teléfono móvil se procede a verificar en una computadora la integridad de la copia por medio de un cálculo de hash del tipo MD5 y se verifica que los valores de cálculo hechos durante la extracción son los mismos que se obtienen en la computadora del investigador.



Figura 3.2 Verificando la integridad de la imagen forense

A modo de ejemplo se procede buscar en las distintas bases de datos de las aplicaciones instaladas en el sistema operativo.

Analizando SMS

La primera base que se va a verificar es la de sms realizados y recibidos desde el teléfono que está siendo investigado, para ello se abre la base llamada mmsms.db que se encuentra en "/data/com.android.providers.telephony/databases", en la figura 3.2 se puede visualizar la localización mencionada.



Figura 3.2 Localización de la Base de datos de SMS

Al abrir la base utilizando el programa "SQLite database Browser" (o cualquier otro cliente SQLite) se visualizan los mensajes enviados, recibidos, horarios de los mismos, si el SMS ha sido visto o no, entre otros datos de gran aporte para una investigación relativa a conocer el intercambio de SMS. Información que se puede observar en la figura 3.3.

atabase	e Structure Bro	owse Data Exe	cute SQL				
able:	sms	• 0				New Record	rd Delete Rec
	status	type	reply_path_pre	subject	body	service_center	locked
133	-1	1	0		Te llamo y si podes nis juntamos	+5434990138	0
134	-1	1	0		Podes morfar ahora?	+5434990142	0
135	-1	1	0		Toy en corrientes y florida	+5434990130	0
136	-1	2			Tengo q ir a reconquista al 500 queres q te avise cuan		0
137	-1	1	0		Dale. Tenes para mucho?	+5434990143	0
138	-1	1	0		Por donde tas?	+5434990130	0
139	-1	2			Ahora en libertad y cordoba y voy a donde te comente		0
140	-1	2			Me confundi voy a esmeralda al 700 te aviso apenas sa		0
141	-1	1	0		Dale . Estoy muy cerca de ahi.	+5434990145	0
142	-1	1	0		Como venis?	+5434990130	0
143	-1	2			Ya bajo		0
144	-1	1	0		Dale toy en la puerta	+5434990135	0
145	-1	2			Sundae		0
146	-1	1	0		Por ser socio Club Personal BLACK tenes un Sundae de	+5434990139	0
147	-1	1	0		Confirmado. Mañana 12.30 en Av de Mayo 651. Nos ve	+5434990145	0
148	-1	1	0		Estrada 3174 Victoria San Fernando	+5434990139	0
149	-1	1	0		8/6: 35% + cuotas en AKIABARA, LITTLE AKIABARA y W	+5434990118	0
150	-1	2			Edu te mande un mail , lo viste ?		0
151	-1	2			Toy en tu auto		0
152	-1	2			Bajas ?		0
153	-1	1	0		Oki	+5434990144	0
154	-1	1	0		VIDAL. EDUARDO ENRIQUE DNI 21.468.558.	+5434990142	0
155	-1	1	0		Hola Paja! Me debes \$270.	+5434990137	0
156	-1	1	0		Buen dia Maxi. Para cuando tendrias el informe???	+5434990133	0
157	-1	2			Buenos dias natalia durante el dia de hoy te la estoy er		0
158	-1	1	0		Buenisimo gracias	+5434990135	0
159	-1	1	0		El 11 y 12/06 -30% y hasta 3 cuotas en El Solar y Boul	+5434990141	0
160	-1	1	0		Maxi, a qué hora hoy? Me olvidé. Gracias	+5434990144	0
161	-1	2			segun me dijeron 20:45		0
162	-1	1	0		Ok!	+5434990130	0
163	-1	1	0		Okt	+5434990151	0

Figura 3.3 Contenido de la base de datos de SMS.

Analizando Contactos

Otra base de datos con información que puede ser de gran utilidad al investigador, es la base de contactos, la cual se encuentra en "/data/com.android.providers.contacts/databases" con el nombre mmssms.db como muestra la figura 3.4. Al abrir esta base se puede apreciar en la tabla "raw_contacts" los contactos agendados por el usuario del dispositivo como muestra la figura 3.5, así como también en esta misma base y en la tabla "Groups" se puede encontrar los grupos definidos por el usuario para relacionar contactos como ver observa en la figura 3.6.

EnCase Forensic		_ 🗆 🗵
💷 👻 🚺 Case (Tesis) 🔻 🖏 View 👻 🎲 Tools 👻 🔧 EnScript 💌 🤮 Add Evid	dence 🔻	•
🐴 Home 🛛 🔗 Evidence 🗙 🔍 Search 🗙 🍕 Records 🗙 📃 Results 🗙		
G 🗿 🅈 Viewing (Entry) ▼ 🛄 Split Mode ▼ (ﷺ Condition ▼ 🍸 Filter ▼	- 🗢 Tags 🔻 👜 Review Package 👻 🐃 🔹 🐼 👻 💽 🚺 🐨 🐜 🐨 🖏 🐼 🖛 🔜	-
com.android.musicfx		
▶─ ○ □		
⊳− ⊳ □] com.android.noisefield	₩ ····································	· ·
▶ D com.android.packageinstaller	Name	Res B
com.android.phasebeam	1 nwk_info.db	
Com.android.phone	2 mmssms.db	
	3 telephony.db	
com and rold providers applications	4 nwk_info.db-journal	
com.android.providers.calendar	5 telephony.db-journal	
com.android.providers.contacts	G mmssms.db-shm	
cache	T mmssms.db-wal	
-D databases		
⊳_⊃ files		
-C- Lib		
Shared_prefs		
com.android.providers.downloads		
com android providers drm		
com.android.providers.media		
com.android.providers.partnerbookmarks		
com.android.providers.security		
		Þ
🕕 Fields 🖾 Report 🦰 Text 🕮 Hex 🗰 Decode 🗰 Doc 🗉 Trans	arrint 🕒 Picture 🖻 Console 🕬 File Extents 🐣 Permissions 📳 🕸 💿 Lo	ck 🖻
Continue #24 Contenues = A Test Stude = At Find Do Test Contenues		
Opacia	Subtressed view	· · · · ·
092748		····· -
092879		
093141		
093272		
093534		
093665		
093796		
094058		
094189		
094320		
094582		
•		
Tesis\Disk Image\/data\data\com.android.providers.telephony\databases\m	umssms.db (PS 8131144 LS 8131144 CL 1016393 SO 0 FO 0 LE 0)	



atabase St Table: rav t_ti	tructure	Browse Data	Execute SQL							
able: rav	w contacts									
t_ti	"_contacts	• • •	 					1	New Record	elete Recor
	ime_conta	starred	display_name	display_name_a	display_name_s	phonetic_name	phonetic_name	sort_key	sort_key_alt	name_ve
616		0	Carla Facebook	Facebook, Carla	40		3	Carla Facebook	Facebook, Carla	
617		0	EVEL-MAYRA	EVEL-MAYRA	40		3	EVEL-MAYRA	EVEL-MAYRA	
618		0	Aldana Guillerm	Guillermo, Aldar	40		3	Aldana Guillerm	Guillermo, Aldar	
619		0	info@fojas0.con	info@fojas0.con	10		0	info@fojas0.con	info@fojas0.con	
620		0	Pamela Melo Ac	Melo Ache, Pam	40		3	Pamela Melo Ac	Melo Ache, Pam	
621		0	Giselle Chanis	Chanis, Giselle	40		3	Giselle Chanis	Chanis, Giselle	
622		0	dolores.cash@n	dolores.cash@n	10		0	dolores.cash@n	dolores.cash@n	
623		0	juan	juan	40		3	juan	juan	
624		0	info@estudioaiz	info@estudioaiz	10		0	info@estudioaiz	info@estudioaiz	
625		0	Google Adwords	Promos, Google	40		3	Google Adwords	Promos, Google	
626		0	MARENDA@srt.	MARENDA@srt.	10		0	MARENDA@srt.	MARENDA@srt.	
627		0	Melina Paracino	Paracino, Melina	40		3	Melina Paracino	Paracino, Melina	
628		0	javi javi	javi, javi	40		3	javi javi	javi, javi	
629		0	Claudio Apebau	Apebaum, Clau	40		3	Claudio Apebau	Apebaum, Clau	
630		0	Francisco Paz W	Walther, Francis	40		3	Francisco Paz W	Walther, Franci	
631		0	ssidolinar@yahc	ssidolinar@yahc	10		0	ssidolinar@yaho	ssidolinar@yahc	
632		0	mmacebal@yah	mmacebal@yał	10		0	mmacebal@yał	mmacebal@yał	
633		0	silvinapalacio@ł	silvinapalacio@ł	10		0	silvinapalacio@ł	silvinapalacio@ł	
634		0	ssidolariar@yah	ssidolariar@yah	10		0	ssidolariar@yah	ssidolariar@yah	
635		0	Maria Elena Del	Del Corro, Maria	40		3	Maria Elena Del	Del Corro, Maria	
636		0	caf.290709@gr	caf.290709@gr	10		0	caf.290709@gr	caf.290709@gr	
637		0	marcela miguez	miguez, marcel	40		3	marcela miguez	miguez, marcel	
638		0	Fabiana Feuer	Feuer, Fabiana	40		3	Fabiana Feuer	Feuer, Fabiana	
639		0	Hcampos	Hcampos	40		3	Hcampos	Hcampos	
640		0	Cintia Romina A	Ayala, Cintia Ro	40		3	Cintia Romina A	Ayala, Cintia Ro	
641		0	Mica Darwin	Darwin, Mica	40		3	Mica Darwin	Darwin, Mica	
642		0	Agustina Darwir	Darwin, Agustin	40		3	Agustina Darwir	Darwin, Agustin	
643		0	MAGGIOLO Edu	Eduardo, MAGG	40		3	MAGGIOLO Edu	Eduardo, MAGG	
644		0	kerstinyan	kerstinyan	40		3	kerstinyan	kerstinyan	
645		0	Tiziano	Tiziano	40		3	Tiziano	Tiziano	
- e -		^	Baarlan 1 6	Contraction and	40		-	n	Cambrand: Band	

Figura 3.5 Contenido Contactos de la base de datos.

		ibase Browser - /mn	t/hgfs/Maxi/	Documents/EnCase/Cases/Te	esis/Temp/contacts2.db
Eile Edit	View Help				
🗋 📂	🔲 🍤 📑 🖬	t 🖬 📫 🗗 🔛 🛛	?		
Databas	Ctructure Pr				
Databas		owse bala Execute SQ			
Table:	groups	-			New Record Delete Record
	dirty	title	title_res	notes	
1	0	Family			
2	0	Friends			
3	0	Coworkers			
4	0	ICE			
5	0	My Contacts		System Group: My Contacts	
6	0	Starred in Android		Starred in Android	
7	0	Friends		System Group: Friends	
8	0	Family		System Group: Family	
9	0	Coworkers		System Group: Coworkers	
10	0	TODOS		TODOS	
11	0	Importado el 17/04/13		Importado el 17/04/13	
12	0	My Contacts		System Group: My Contacts	
13	0	Starred in Android		Starred in Android	
14	0	Friends		System Group: Friends	
15	0	Family		System Group: Family	
16	0	Coworkers		System Group: Coworkers	
17	0	Círculos de Google+			
11					
					<u>.</u>
<	1 - 17 of 1	7 >			Go to: 0

Figura 3.6 Contenido grupos de la base de datos de contactos.

Analizando Llamadas

A continuación se puede observar en la figura 3.7 la tabla "logs" de la "logs.db" base de datos ubicada en /data\data\com.sec.android.provider.logsprovider\databases contiene un registro de todas las llamadas y sms emitidas, recibidas y perdidas por el dispositivo así como también el tiempo de duración, el contacto vinculado con la llamada y la fecha y el horario entre otros datos de importancia. Se consultarápor medio de una sentencia sql: "SELECT datetime(date/1000, 'unixepoch', 'localtime') as fecha, name, number, duration, type FROM logs WHERE (logtype=100 OR logtype=500);".

e Edit View Help					
े 🚅 🖬 भ 📑 📸	# 🖬 🎽 🖺 🗞?				
atabase Structure Brow	vse Data Execute SQL				
SQL string:					
SELECT datetime(date/10	00, 'unixepoch','localtime') a	as fecha, name, number,duratio	on, type FROM logs WH	ERE (logtype=100 OR logtype=500);
Execute query					
rror message from databa	ase engine:				
No error					
)ata returned:					
				1	
2013 09 16 17:56:55	Mama	1152578002	490		
2013-08-16 19:45:10	Marria	+541168410600	90	2	
2013-08-10 19:45:10	ximona hondinolli	1157529540	3	1	
013 00 16 22:01:17	Mariano	+541145703749	2	2	
013-08-10 22.02.14	vimena bondinelli	T341143/93/46	2	2	
2013-08-16 22:03:17	ximena bendinelli	32434363	0	2	
2013-08-16 22:04:05	ximena bendinelli	1152454565	4	2	
2013-08-17 09:16:48	ximena bendinelli	+541157528540	0	3	
2013-08-17 09:24:48	wine and the stational lit	-2	0	3	
2013-08-17 10:11:20	ximena bendinelli	+541157528540	0	2	
2013-08-17 10:11:39	ximena bendinelli	+541157528540	218	2	
2013-08-17 10:17:14		+542944334600	37	1	
2013-08-17 13:27:25		-2	61	1	
2013-08-17 14:04:42	Pablo Temponi	+541156169431	115	1	
2013-08-17 20:03:00	ximena bendinelli	+541157528540	291	2	
2013-08-17 22:02:09	Mama	1152578902	262	2	
2013-08-18 11:30:03		1146750203	5	1	
2013-08-18 21:34:12	Mama	1152578902	94	2	
2013-08-18 21:37:19	Casa	1146741659	0	2	
2013-08-18 21:41:59	Casa	1146741659	2	2	
2013-08-18 21:42:57	ximena bendinelli	1157528540	2	2	
2013-08-18 21:44:02	Mariano	+541145793748	0	2	
2013-08-18 21:52:02		+5491159025101	0	2	
2013-08-18 21:53:19	Casa	1146741659	2	2	
2013-08-19 10:45:57		+541146750203	0	3	
2013-08-19 12:08:46	ximena bendinelli	+541157528540	0	3	
2013-08-19 12:14:15	ximena bendinelli	+541157528540	515	2	
		-2	63	1	
2013-08-19 14:22:57					

Figura 3.7 Registro de llamadas en el dispositivo.

Analizando Redes WiFi

Es posible también saber a cuáles redes WiFi estuvo conectado el dispositivo, ya que la información se encuentra dentro de la partición data (o userdata). Dentro del directorio /misc/wifi existe un archivo llamado wpa_supplicant.conf (/misc/wifi/supplicant.conf) que contiene información de las redes a las cuales estuvo conectado el dispositivo, asi como también sus claves de acceso en texto plano, como se muestra a continuación una porción de dicho archivo.

```
ctrl_interface=wlan0
update_config=1
device_name=m0ub
```

```
manufacturer=samsung
model name=GT-I9300
model number=GT-I9300
serial number=4df113e12c0f4f31
device type=10-0050F204-5
config_methods=physical_display virtual_push_button keypad
p2p listen reg class=81
p2p listen channel=1
p2p_oper_reg_class=115
p2p oper channel=48
bss_expiration_scan_count=1
network={
        ssid="Personal EAP"
        scan ssid=1
        key mgmt=WPA-EAP IEEE8021X
        eap=SIM
identity="1722341212108215@wlan.mnc034.mcc722.3gppnetwork.org"
        priority=1
       vendor_spec_ssid=1
}
network={
        ssid="casita"
        psk="estaesunaclavedeprueba"
       key mgmt=WPA-PSK
        priority=2
}
network={
        ssid="ING"
        scan ssid=1
        psk="1ngen1er1a"
        key mgmt=WPA-PSK
}
network={
        ssid="juani"
        key mgmt=NONE
        auth alg=OPEN SHARED
        wep key0=XXXXXXXXX
        priority=3
}
```

Figura 3.8 Listado de redes wifi donde se conectó el dispositivo.

Analizando Bluetooth

Al igual que en los anteriores casos de análisis, la actividad bluetooth queda almacenada en una base salite localizada en data/com.android.bluetooth/databases cuyo nombre es btopp.db. En su interior se puede identificar la actividad que tuvo el dispositivo bluetooth el cual muestra la dirección MAC del dispositivo con el que se conectó así como también cuál fue la actividad realizada. Estainformación puede ayudar a vincular el dispositivo en investigación con otros ya sean celulares, computadoras, tablets, entre otros. A continuación se muestra el contenido del archivo btopp.db:

```
sqlite> select * from btopp;
l|content://media/external/images/media/2063|20130805_215124.jpg||im
age/jpeg|0|00:1D:4F:87:F7:35|0|2|497|2764058||1375750344613|0
2|content://media/external/images/media/2062|20130805_215109.jpg||im
age/jpeg|0|00:1D:4F:87:F7:35|0|2|497|2646730||1375750360247|0
3|content://media/external/images/media/2073|20130806_101603.jpg||im
age/jpeg|0|00:1D:4F:87:F7:35|0|2|497|1913774||1375795051047|0
```

Figura 3.9 Actividad Bluetooth del dispositivo.

Analizando Google Maps

Es posible establecer alguna de las búsquedas o localizaciones del usuario del dispositivo por medio de las bases de datos de la aplicación Google Maps. En la base de datos localizada en data/com.google.android.apps.maps/databases *search_history.db* se puede encontrar búsquedas de posibles destinos del dispositivo como se observa en la figura 3.10.

También es posible establecer lugares con fecha y horario que fueron cargados como destino en el Google Navigator, dicha información surge de la base de datos localizada en el mismo lugar que la anterior pero denominada *da_destination_history*, que al abrilapermite apreciar esa información como se observa en la figura 3.11.

~ ~	× SQLite Da	atabase Brow	ser - /mnt/tes	timg/data/com.google.andro	oid.apps.map	s/databases/	search_history
<u>F</u> ile E	dit View Help						
	i 🖬 🕫 📑		≸ 🔚 [\ ?				
Data	base Structure	Browse Data	Execute SQL				
Tab	e: suggestions	• 🔍				New Record	d Delete Record
	_id	data1	singleResult	displayQuery	latitude	longitude	timestamp
1	16	lavalle 1737, 5ŗ		Lavalle 1737, 5ŗ piso, "F"	20000000	200000000	370444991773
2	17	los palitos, 224		Los Palitos, 2243 Arribeños, Belgr	20000000	20000000	370740619211
3	18	belgrano 1550			20000000	20000000	370865339799
4	19	moreno 1550			20000000	20000000	370865446810
5	20	25 de mayo 15			20000000	20000000	370877203716
6	21	maipú 12 12 ca		Maipú 12 12 Capital Federal	20000000	20000000	370881284038
7	22	emeralda 12 12		emeralda 12 12 Capital Federal	20000000	20000000	370881298039
8	23	esmeralda 1200		Esmeralda 1200 Capital Federal	20000000	20000000	370881316561
9	24	av de mayo 605			20000000	20000000	371036656019
10	25	colpayo 760 , c			20000000	20000000	371349161175
11	26	-34.614593505		-34.614593505859375,-58.3743	20000000	20000000	372111222012
12	29	crisologo larrald		Crisologo Larralde 3609, Oficina 1	20000000	20000000	372424412308
13	30	esmeralda 1111		Esmeralda 1111, Buenos Aires	-34599476	-58377478	372430414388
14	33	nogoya 3079			20000000	20000000	372863629984
15	34	-34.612289428			20000000	20000000	373073870861
16	35	lo de charlie		Lo de Charlie	20000000	20000000	373130022215
17	36	virrey melo 949		Virrey melo 949	20000000	20000000	373151406854
18	38	gurruchaga 222		-	20000000	20000000	373246314127
19	39	suiza1050, pac			20000000	20000000	373561165758
20	40	jorge luis borge		Jorge Luis Borges 2180, Buenos A	-34584843	-58424470	373589796396
	< 1 - 20 c	of 20 >				Go to:	D

Figura 3.10 Direcciones Buscadas por Google Map.

aba	ase Structure	Browse Data	Execute SQL							
ble	destination_h	istory 🚽 🔍							New Rec	ord Delete Re
	time	dest_lat	dest_ing	dest_title	dest_address	dest_token	source_lat	source_ing	day_of_week	hour_of_day
	369224561623	-34603052	-58389473		Montevideo 496	FdT_7_0dHwyF	-34583150	-58396924	4	9
	369237162309	-34919523	-57956114		Calle 49 876	FZ0r6_0d7qiL_0	-34624920	-58367985	4	12
	369757163417	-34735755	-58397524		Av Hipólito Yrigoyen 6935	FXX57f0drOyE_	-34612540	-58381784	3	13
	369788495894	-34611646	-58443552		Colpayo 760	FULe7_0d4DiE_	-34681801	-58385196	3	21
	369835426464	-34603109	-58391615		Lavalle 1737	FZv_7_0dwQOF	-34585222	-58390645	4	10
	370040869780	-34634680	-58417788	MegaBikes	Castro Barros 1832	FUiE7_0dhJ2E_(-34602642	-58443068	6	19
	370446354400	-34603109	-58391615		Lavalle 1737	FZv_7_0dwQOF	-34577313	-58411256	4	12
	370740622041	-34556675	-58451118		Los Palitos, 2243 Arribeños, Be	Ff208P0dUhuE	-34585458	-58434640	7	22
	370865468036	-34612442	-58388198		Moreno 1550	FSbb7_0dGhGF	-34615708	-58417251	2	8
0	370877244196	-34704593	-58414726		25 de Mayo 1568	FS9z7v0deqmE	-34611973	-58388250	2	12
1	372257080733	-34587085	-58409723		Beruti 3345 , capital federal	FTM-8P0dBb2E	-34593116	-58428300	4	11
2	372424472129	-34554832	-58478862		Av Crisólogo Larralde 3609	FTC88P0d8q6D	-34578356	-58459841	6	10
3	372864210043	-34602877	-58491839		Nogoyá 3079	FYMA8P0dQXyD	-34625194	-58471965	4	12
4	373151427019	-34628362	-58493830		Virrey Melo 949	Ffac7_0denSD_	-34628441	-58493234	7	19
5	373246350527	-34585218	-58423004		Gurruchaga 2222	FX5F8P0dJImE_	-34592915	-58432561	1	22
: 1	373247890182	-34585218	-58423004		Gurruchaga 2222	FX5F8P0dJImE_	-34592900	-58439092	1	22

Figura 3.11 Destino de Navegación cargados en Google Navigator

Analizando Twitter

Es posible conocer por las bases de datos de Twitter la actividad del dispositivo con esta red social, se pueden analizar los mensajes públicos así como los mensajes privados que son enviados a la cuenta configurada en el dispositivo. En el directorio data/com.twitter.android/databases se encuentran las bases de datos con los contenidos de interés como *"twits"* que eran leídos desde el dispositivo y también los mensajes privados del usuario con otros usuarios de la red social.

· · ·	∨ × Edit	SQLite Data View Help	abase Browse	r - /mnt/testir	ng/data/com.twitter.android/databases/130828878-3.db		
. 🗅 I	2	🗐 🍤 📑 🖩	ð 🖬 🖬 👘	🖺 🏹			
Dat	abaco	Structure Br	rowse Data				
Dat	abase	Structure bi	owse baca JEX	ecute sqt			
Ta	ble:	statuses	• @				New Record Delete Record
		_id	status_id	author_id	content	source	source_url
1	45	145	438352433153	69416519	Los sueldos de agosto deberán liquidarse con las nuevas exenciones de Gananci	dlvr.it	http://dlvr.it
1	46	146	545024196611	56179836	[web applications] - WordPress plugins KBoard SQLi/XSS Vulnerabilities http://t.c	dlvr.it	http://dlvr.it
1	47	147	845219319809	594809093	Hermosa noche con mis compa de practico profesional, ahora a casa y DORMIR!	Twitter for Black	http://blackberry.com/twitter
1	48	148	315378479104	2241661	Que aguacero!	Twitter for Andr	http://twitter.com/download/and
1	49	149	942854311936	242347719	Iran's Teaching Hacking in High School - http://t.co/aUsO7HI1bG	HootSuite	http://www.hootsuite.com
1	50	150	985788526592	242347719	#Pentesting WITHOUT spending \$1,000's #APT L-1 > - http://t.co/XJyWR3K71r	HootSuite	http://www.hootsuite.com
1	51	151	201364471808	242347719	Woman says hacker accessed computer, videotaped her - http://t.co/yM0wD97Y	HootSuite	http://www.hootsuite.com
1	52	152	593147729921	101166742	Windows Técnico - Windows Assessment and Deployment Kit (Windows ADK) htt	Twitter for Mac	http://itunes.apple.com/us/app/
1	53	153	526675775488	101166742	Windows Técnico - Extraer la última palabra de un texto en Microsoft Excel http	Twitter for Mac	http://itunes.apple.com/us/app/
1	54	154	535300386816	101166742	Seguridad Apple - Bug en CoreText crashea aplicaciones en iOS y Mac OS X http	Twitter for Mac	http://itunes.apple.com/us/app/
1	55	155	335279382529	5027791	Today stats: 3 followers, No unfollowers and followed 2 people via http://t.co/1H3	Unfollowers.me	http://unfollowers.me
1	56	156	290230517760	242347719	NSA has super secret hacker collective according to newly revealed Snowden do	HootSuite	http://www.hootsuite.com
1	57	157	443602731008	572964305	Hoy os traemos La Trinchera de Flinskin 2 donde se explica cómo Kim Dot Com e	Twitter for Mac	http://itunes.apple.com/us/app/
1	58	158	812425109504	109488608	El lado del mal - No Lusers 165 - El nuevo Batman http://t.co/RFISLJCzxx #Batma	Twitter for Mac	http://itunes.apple.com/us/app/
1	59	159	995017752576	144183456	#ReciclandoPodcast - 040. Crimen Digital LIVE, en podcast. Segunda parte - htt	podcast crimen	http://crimendigital.com
1	fn	160	360466408060	2/23/7710	KEYNOTE: @nrivacyneek > Are Your Audience.Ruilding Efforts Suffering Due to (HootSuite	http://www.hootsuite.com
	<	1 - 1000 (of 3105 >				Go to: 0

Figura 3.12 Lectura de "Twits" del usuario.

			* 🖭 💦						
_									
taba	ase Structure	Browse Data	Execute SQL						
able	messages	<u>•</u> (0),						New Record	Delete Reco
	_id	type	msg_id	content	created	sender_id	recipient_id	thread	is_unread
	1	1	404835008512	Gracias por seguirme	372259877000	97739625	130828878	97739625:130	
2	2	1	436375859200	http://t.co/UPOQPGivhj	369683057000	269622219	130828878	130828878:26	
3	3	1	771758448640	http://t.co/cFwehMQiAY	369247784000	269622219	130828878	130828878:26	
L I	4	1	504035977217	http://t.co/1OP1goesoC	369159291000	269622219	130828878	130828878:26	
i	5	1	480002646017	Gracias Max	366078417000	60486251	130828878	60486251:130	
i	6	1	513654059008	Max, no hemos tenido contacto, quieres que	366061497000	60486251	130828878	60486251:130	
·	7	1	703912878080	Buenos días, te llego mi email?	365769956000	60486251	130828878	60486251:130	
3	8	1	770454732800	A qué otro email puedo contactarte? Si te po	365731348000	60486251	130828878	60486251:130	
)	9	1	175588454400	Gracias por seguir a @inforenses el primer eq	355749098000	58443187	130828878	58443187:130	
.0	10	1	823596470276	Desde ahora la comunidad es http://t.co/ieLM	338095310000	387327748	130828878	130828878:38	
1	11	1	159353790465	Gracias x seguirnos! Conocé promos, benefic	331190788000	211977538	130828878	130828878:21	
2	12	0	510938759168	Andres, yo les pase tu contacto y el procedin	366077709000	130828878	60486251	60486251:130	
.3	13	0	514413551616	Buenos días Andres , si te lo acabo de respon	365770388000	130828878	60486251	60486251:130	
.4	14	0	928777125888	maximiliano@bendinelli.com.ar o maximiliano	365731386000	130828878	60486251	60486251:130	
.5	15	0	091935322114	Mi Linea es 1152578889 y tengo un Atrix ,	331149526000	130828878	211977538	130828878:21	
1									
	< 1 - 15 0	of 15 >	1					Go to: 0	

Figura 3.13 Mensajes privados del usuario con otros de la red social.

Analizando Whatsapp

Al igual que en los casos de análisis anterior es posible por medio de las bases de datos de esta aplicación conocer el intercambio de mensajes de usuario del dispositivo móvil, dicha información se encuentra en base de datos denominada msgstore.db directorio la el /data/com.whatsapp/databases, allí se podrá observar el horario, el mensaje, el status del mensaje y quién es el que envía o recibe dicho mensaje. Es importante aclarar que si los mensajes no fueron archivados se encuentran en texto en la base de datos, como se puede apreciar en la imagen que se encuentra a continuación:

- 12 -	≥ ⊞"⊞									
tabase Stru	cture Br	owse Data Execute SQL								
able: mess	ages	J 🔍						New	Record Dele	ete Recor
_id		key_remote_jid	key_from_me	key_id	status	needs_push	data	timestamp	media_url	med
20	520	5491121805359@s.whatsapp.net	0	1369011039-2	0	0	Yo no soy vaga	369076325000		
21	521	5491121805359@s.whatsapp.net	0	1369011039-2	0	0	Jaja	369076325000		
522	522	5491121805359@s.whatsapp.net	1	1369075692-7	5	0	JajajajJ	369076335658		
23	523	5491121805359@s.whatsapp.net	0	1369011039-2	0	0	Quiero comida :(369076410000		
24	524	5491121805359@s.whatsapp.net	0	1369011039-2	0	0	Cuando salgo de acá me voy	369076448000		
525	525	5491121805359@s.whatsapp.net	1	1369075692-8	5	0	Y compra	369076451361		
26	526	5491121805359@s.whatsapp.net	1	1369075692-9	5	0	Ahhb rico	369076457948		
527	527	5491121805359@s.whatsapp.net	1	1369075692-10	5	0	No almorzaste ?	369076469881		
28	528	5491121805359@s.whatsapp.net	0	1369011039-2	0	0	Nop	369076480000		
29	529	5491121805359@s.whatsapp.net	0	1369011039-2	0	0	Tome un té con tostadas que	369076500000		
i30	530	5491121805359@s.whatsapp.net	0	1369011039-2	0	0	Jaja	369076501000		
31	531	5491121805359@s.whatsapp.net	1	1369075692-1	5	0	Come algo	369076512228		
i32	532	5491153176282@s.whatsapp.net	0	1368093155-7	0	0	00	369079458000		
i33	533	5491153176282@s.whatsapp.net	1	1369077780-1	5	0	Q paso ?	369080146919		-
i34	534	5491121805359@s.whatsapp.net	0	1369011039-24	0	0	Estas por acá?	369085819000		
35	535	5491121805359@s.whatsapp.net	0	1369011039-2-	0	0	Porque me anda mal el facé	369085823000		
i36	536	5491121805359@s.whatsapp.net	1	1369080232-1	5	0	Aca toy	369085891049		
37	537	5491121805359@s.whatsapp.net	1	1369080232-2	5	0	A dinde t escapas ?	369086032588		
i 38	538	5491121805359@s.whatsapp.net	0	1369011039-2	0	0	A mi casa	369086348000		
i39	539	5491121805359@s.whatsapp.net	0	1369011039-2	0	0	Jaja na creo que hay clases p	369086355000		
40	540	5491121805359@s.whatsapp.net	0	1369011039-2	0	0	Así que me quedo	369086356000		
41	541	5491121805359@s.whatsapp.net	1	1369080232-3	5	0	Bue	369086517339		
i42	542	5491121805359@s.whatsapp.net	1	1369080232-4	5	0	T volvio la lyz?	369086523655		
i43	543	5491121805359@s.whatsapp.net	0	1369011039-2	0	0	La que??	369086954000		
44	544	5491121805359@s.whatsapp.net	1	1369080232-5	5	0	Luz	369086969248		
45	545	5491121805359@s.whatsapp.net	0	1369011039-2	0	0	deja de escribir con las patas	369086980000		
				j						•

Figura 3.14 intercambio de mensajes de Whatsapp.

Tambiénfueposiblehallarendirectoriodata\media\WhatsApp\Databasesbackupsdistintosarchivoscorrespondientes a backups (o copias de resguardo) de las bases de datoscontenedorasde mensajes como se aprecia en la figura 3.15, pero los

mismos se encuentran encriptados y debido a que no es parte de este trabajo no se continúa con la investigación de dichos archivos.

🗊 EnCase Forensic			
💷 🔻 🚺 Case (Tesis) 🔻 🖏 View 👻 🍈 Tools 💌	🛠 EnScript 🔻 📇 Add Evidence 👻		 •
🐴 Home Records 🗙 🚑 Evidence 🗙			
🔇 🌍 🅈 Viewing (Entry) 👻 🛄 Split Mode 👻 [E Condition 🔻 🍸 Filter 💌 🕫 Tags 🔻 🙀 Review Package 💌 🐃 🔹 🕥 💌 👰 🛙 🚳 💌 🐜 💌	(i) • 🖏 🕥 • 🎜	-
Playlists	Table Timeline E Gallery		
-D Podcasts			
			File
Samsung	Name	Kel Kel Fol Ign	Ext
- SamsungSelecciones	1 msgstore-2013-09-09.1.db.crypt		crypt
-D Snapchat	2 msgstore-2013-09-02.1.db.crypt		crypt
-D tmp	3 msgstore-2013-09-03.1.db.crypt		crypt
► ► ► ► ► ► ► ► ► ► ► ► ► ► ► ► ► ► ►	4 msgstore-2013-09-05.1.db.crypt		crypt
⊳⊃ viber	5 msgstore-2013-09-04.1.db.crypt		crypt
wdh_update	6 msgstore-2013-09-08.1.db.crypt		crypt
A WhatsApp	7 msgstore-2013-09-06.1.db.crypt		crypt
Databases	8 msgstore.db.crypt		crypt
▶ D Media			
Profile Pictures			
kbmc 🚽			
			F
() Fields 🔛 Report 🎦 Text 🔠 Hex 🇰 De	code 📧 Doc 📳 Transcript 📲 Picture 🖭 Console 🕬 File Extents 🔒 Permissions 📱	1 🕑	🗆 Lock 🛛 📑
* Zoom In 🛛 Zoom Out 🔍 100% 👻 📑 🛄			-
Name msgstore-2013-09-09.1.	db.crypt		
File Ext crypt			
Logical Size 4.808.720			
Signature Analysis Unknown			
Last Accessed 08/09/13 04:00:00 a m			<u> </u>
] Tesis\Disk Image\/data\media\WhatsApp\Databa	ses\msgstore-2013-09-09.1.db.crypt		

Figura 3.15 Backups de bases de datos encriptadas de Whassup.

Realizando Carving

Dado que en el capítulo anterior se procedió a realizar una copia forense (o copia bit a bit), este tipo de extracciones en las cuales toda la información contenida en la memoria interna del teléfono (visible o no para el Filesystem) es extraída, permite al investigador encontrar archivos que hayan sido eliminados y que no estén siendo mostrados por el Filesystem. Por lo tanto, por medio de la utilización de herramientas de "data carving" se procederá a recuperar el contenido que haya sido eliminado de la memoria. A continuación se muestra cómo luego de haber realizado el procedimiento para la recuperación de imágenes, se puede acceder a aquellas que no podrían haber identificarsea simple vista por quien esté realizando el análisis del dispositivo, en este ejemplo se pudo recuperar de la copia forense un total de 1669 imágenes como se puede apreciar en la figura 3.16.



Figura 3.16 Recuperación de Archivos de Imagen de la copia Forense

Las técnicas de análisis mencionadas en este capítulo son sólo algunas de las tareas que se pueden realizar para analizar información del dispositivo móvil que ayudará a investigadores a la recopilación de información que será de utilidad para una causa.

Capítulo 4: ¿Virtualización?

La virtualización en entornos de informática forense es de utilidad para poder inicializar o "encender de manera virtual" con la copia bit a bit (o imagen forense) el dispositivo, computadora, notebook, pc, etc que se este analizándo. Además, debido a que en ciertas situaciones es necesario poder inicializar el dispositivo sin escribir información en el disco o contaminar la evidencia, se procede a virtualizar el entorno y trabajar con la copia forense de manera que ningún dato sea alterado. Para realizar las tareas con imágenes de computadoras (notebooks, PCs, netbooks) existe una herramienta denominada LiveView [9] que permite poder virtualizar dichos entornos.

Los usos de esta clase de herramientas son múltiples, entre algunas situaciones donde se puede necesitar se encuentran:investigar el comportamiento del dispositivo con la red con el fin de identificar si alguna aplicación o malware esta interactuando con el equipo y la red, tener una visión del escritorio de usuario y sus accesos directos o simplemente tener una visión general de la utilización del usuario con el dispositivo que está en análisis.

Semejantes herramientas no existen momentáneamente para dispositivos con sistema operativo Android, por esta razón, en este capítulo se intentará realizar una virtualización del entorno. Para lograr este cometido, será necesario realizar una imagen forense de la partición *system*, la cual contiene el sistema operativo o firmware que se está utilizando en el dispositivo, y *userdata* que es donde se encuentra toda la parametrización del usuario así como toda la información y aplicaciones del dispositivo a analizar y se utiliza el emulador de Android [10] cambiando su ejecución *"normal"* con el fin de que tome como memoria (o disco interno) la que previamente se extrajo como la adquisición forense, y se intentará virtualizar el entorno con las imágenes del dispositivo analizado.

A continuación se demostrará paso a paso las tareas que serán realizadas para lograrlo.

Extrayendo las imágenes forenses

Para dar comienzo a esta etapa se procede a realizar una copia bit a bit de las particiones *userdata* y *system*, la cuales se almacenaron en una memoria micro-sd para luego ser copiadas el equipo que será el encargado de ejecutar la máquina virtual. Como se explicó en el punto 2 utilizando las herramientas dd, dfcldd o d3cdd se procede a extraer las particiones mencionadas almacenándolas en una memoria externa micro-sd.

La primer partición a extraer es la denominada system, la cual tiene el entorno de ejecución de Android.

```
~ # dcfldd if=/dev/block/mmcblk1p9 of=/sdcard/mmcblk1p9.dd hash=md5
5376 blocks (168Mb) written.Total (md5):
3026dc171cff839cb54765ef79babd36
5408+0 records in
5408+0 records out
```

Figura 4.1 Extracción de la partición system

La segunda partición a extraer es la denominada data, la cual posee todos los datos relacionados con el usuario y las customizaciones del dispositivo.

```
~ # dcfldd if=/dev/block/mmcblk1p10 of=/sdcard/mmcblk1p10.dd
hash=md5
937500 blocks (30Gb) written.Total (md5):
2a1946fc3bb0a952d9a1006f0d911a07
```

Figura 4.2 Extracción de la partición data

Ejecutando el entorno virtual

Una vez que las copias han sido almacenadas en el equipo encargado de la virtualización se procede a crear un perfil de emulación en la herramienta de desarrollo denominada ADV (Android Virtual Device). En este caso será llamada "tesis3" como se puede observar en la figura 4.1.

× Edit Android Vi	irtual Device (AVD)
AVD Name:	tesis3
Device:	4.7" WXGA (1280 × 720: xhdpi) ▼
Target:	Google APIs (Google Inc.) - API Level 15 🔹
CPU/ABI:	ARM (armeabi-v7a)
Keyboard:	Hardware keyboard present
Skin:	Display a skin with hardware controls
Front Camera:	None
Back Camera:	Emulated 🔻
Memory Options:	RAM: 1024 VM Heap: 48
Internal Storage:	2 GiB v
SD Card:	o Size: MiB v
	O File: Browse
Emulation Options:	Snapshot Use Host GPU
Override the exis	sting AVD with the same name
	Cancel OK

Figura 4.3 Creando un Virtual Device

Luego, se procede a ejecutar la línea de comando como se muestra en la figura 4.2, utilizando como parámetros el sistema extraído del dispositivo invocándolo por medio del parámetro –system y parametrizando por medio del parámetro *"datadir"* el archivo que contendrá la copia de la partición userdata y se agregan los parámetros "-verbose", "-show-kernel" y "-shell" con fines de depuración y de ingreso a la plataforma por medio de una consola (o Shell).

```
emulator @tesis3 -ramdisk /root/roms/emulator/newramdisk.cpio.gz -
system /root/roms/xoom/system-test.img -datadir /mnt/datadir -
verbose -show-kernel -shell
```

Figura 4.4 Comando ejecutado para la inicialización de la "Virtualizacion"

Habiendo ejecutando la máquina virtual se puede observar desde la ventana del emulador que inicia el sistema operativo y queda el logo animado mostrándose en la pantalla sin que el estado cambie en ningún otro momento.



Figura 4.5 Emulador iniciando con las imágenes forenses

Al verificar el log de ejecución por medio del comando "*adb -e logcat*" se puede observar la siguiente salida:

/dalvikvm(138): threa	adid=11	: thread	exi	ting	with	uncaught
exception (group=02	40a191f8)					
/AndroidRuntime(138):	*** FA	TAL EXCE	PTION	IN	SYSTEM	PROCESS
ndroid.server.Serv	verThread						
/AndroidRuntime(1	38):	dalvik.	system	n.Sta	leDexCa	cheError
system/framework/	com.motor	ola.and	roid.fram	lework	s.ja:	ſ	
/AndroidRuntime(1	38):					at

```
138):
E/AndroidRuntime(
                                                                  at.
com.android.server.pm.PackageManagerService.<init>(PackageManagerSer
vice.java:990)
E/AndroidRuntime(
                         138):
                                                                  at
com.android.server.pm.PackageManagerService.main(PackageManagerServi
ce.java:838)
E/AndroidRuntime(
                        138):
                                                                  at
com.android.server.ServerThread.run(SystemServer.java:172)
E/AndroidRuntime( 138): Error reporting crash
E/AndroidRuntime( 138): java.lang.NullPointerException
E/AndroidRuntime(
                         138):
                                                                  at
com.android.internal.os.RuntimeInit$UncaughtHandler.uncaughtExceptio
n(RuntimeInit.java:72)
E/AndroidRuntime(
                         138):
                                                                  at
java.lang.ThreadGroup.uncaughtException(ThreadGroup.java:693)
                         138):
E/AndroidRuntime(
                                                                  at
java.lang.ThreadGroup.uncaughtException(ThreadGroup.java:690)
I/Process ( 138): Sending signal. PID: 138 SIG: 9
I/Zygote
          ( 37): Exit zygote because system server (138) has
terminated
```

4.6 Salida del debug de la máquina virtual

De esta depuración se observa con claridad que el proceso Zygote se termina por una excepción y por lo tanto no puede continuar.

Resultado

Como resultado se observa que al configurar el emulador con las particiones que se extrajeron como parte de su entorno real inicia el proceso de emulación, se visualiza el logo del fabricante del firmware (Motorola) quedando el proceso de inicialización detenido en esta fase del inicio. Al analizar los logs del entorno se puede apreciar que el sistema comienza su proceso de iniciación pero el DAVLINK comienza a tener un gran número de errores que provocan la "*caída*" del proceso en sí. Se vuelve a ejecutar automáticamente y se obtiene el mismo resultado del comienzo, una y otra vez, entrando en un loop (o ciclo) infinito sin que la máquina virtual logre terminar su ejecución "normal". Por este motivo, se decide cancelar el proceso de emulación sin lograr el objetivo planteado al comienzo de este capítulo.

Dado que no era el principal objetivo de este trabajo realizar una virtualización y por cuestiones que exceden las metas planteadas, al no ser posible logar el funcionamiento de esta tarea, se deja en este estado de estudio.-

Desarrollo del software

Actualmente se está trabajando en realizar una distribución gratuita de este software para que pueda ser utilizada por toda la comunidad de investigadores forenses y fuerzas de la ley sin costo alguno. Dicho proyecto ha sido llamado "Android Forensic Toolkit".-

Conclusiones

La falta de herramientas de código abierto o gratuito que ayuden a peritos o investigadores a extraer la información que está contenida en dispositivos móviles con sistema operativo Android siguiendo las mejores prácticas forenses, fue la inquietud que motivó la realización de este trabajo. A partir de este cuestionamiento, se procedió a verificar la factibilidad de realizar la extracción y, al constatarse que dicha tarea era posible, se desarrolló una metodología que mediante el uso de herramientas nativas de Unix así como también herramientas de código abierto, de código libre o gratuito, permitió la extracción forense de la información contenida en la memoria interna del dispositivo o teléfono. Al mismo tiempo, se demostró que con la extracción es posible realizar un análisis de la actividad del dispositivo y extraer información sensible como usuario con el goeposicionamiento, llamados, mensajes, sms, chat, imágenes, videos, elementos eliminados y actividad en redes sociales.

Adicionalmente se intentó realizar una virtualización del dispositivo mediante la utilización de las imágenes forenses extraídas del equipo en conjunto con el emulador de Android provisto por las herramientas de desarrollo de esta plataforma. Como se pudo observar en el capítulo 4, si bien es posible hacer iniciar un entorno virtual con las copias de la memoria, no es factible por el momento emular todo el hardware del equipo original. Además, debido a que algunas aplicaciones instaladas requieren acceso directo a equipamiento físico que, por razones inherentes a la emulación (o virtualización) no se encuentran disponibles, esta tarea se torna por el momento difícil ya que el sistema es inestable cuando logra arrancar y en algunos casos *no arranca*, Si bien no es objetivo inicial de este trabajo, se deja en este estado de situación y se abre la posibilidad de que la cuestión sea objeto de investigación a futuro por quien desee realizarlo ya que se convertiría en una excelente herramienta para investigadores forenses que interactúen con esta clase de tecnología.

Anexos

Ejecución del emulador para virtualización

```
root@bt:~/adt-bundle-linux-x86 64-20130219/sdk/tools#./emulator
@tesis3 -ramdisk /root/roms/emulator/newramdisk.cpio.gz -system
/root/roms/xoom/system-test.img -datadir /mnt/datadir -verbose -
show-kernel -shell
emulator: found SDK root at /root/adt-bundle-linux-x86_64-
20130219/sdk
             Android virtual device
                                                  file at:
emulator:
/root/.android/avd/tesis3.ini
emulator: virtual device content at /root/.android/avd/tesis3.avd
emulator: virtual device config file:
/root/.android/avd/tesis3.avd/config.ini
emulator: using core hw
                                             config
                                                         path:
/root/.android/avd/tesis3.avd/hardware-gemu.ini
emulator: Found AVD target API level: 17
emulator: 'magic' skin format detected: 1280x720
emulator: autoconfig: -skin 1280x720
emulator: autoconfig: -skindir (null)
emulator: keyset loaded from: /root/.android/default.keyset
         found SDK root at /root/adt-bundle-linux-x86_64-
emulator:
20130219/sdk
emulator: trying to load skin file '/root/adt-bundle-linux-x86 64-
20130219/sdk/tools/lib/emulator/skins/dynamic//layout'
emulator: loaded dynamic skin width=1280 height=720 bpp=16
emulator:
         autoconfig: -kernel /root/adt-bundle-linux-x86 64-
20130219/sdk/system-images/android-17/armeabi-v7a//kernel-qemu
emulator: Using initial system image: /root/roms/xoom/system-
test.img
emulator: autoconfig: -data /root/.android/avd/tesis3.avd/userdata-
qemu.img
emulator:
                          autoconfig:
                                                       -initdata
/root/.android/avd/tesis3.avd/userdata.img
emulator: autoconfig: -cache /root/.android/avd/tesis3.avd/cache.img
emulator: Physical RAM size: 1024MB
Content of hardware configuration file:
 hw.cpu.arch = arm
 hw.cpu.model = cortex-a8
 hw.ramSize = 1024
 hw.screen = touch
 hw.mainKeys = yes
 hw.trackBall = no
 hw.keyboard = yes
 hw.keyboard.lid = no
 hw.keyboard.charmap = qwerty2
 hw.dPad = no
 hw.gsmModem = yes
 hw.gps = yes
```

```
hw.battery = yes
  hw.accelerometer = yes
  hw.audioInput = yes
  hw.audioOutput = yes
 hw.sdCard = no
  disk.cachePartition = yes
  disk.cachePartition.path = /root/.android/avd/tesis3.avd/cache.img
  disk.cachePartition.size = 66m
  hw.lcd.width = 1280
  hw.lcd.height = 720
  hw.lcd.depth = 16
 hw.lcd.density = 320
 hw.lcd.backlight = yes
 hw.gpu.enabled = no
 hw.camera.back = emulated
 hw.camera.front = none
  vm.heapSize = 48
 hw.sensors.proximity = yes
 hw.sensors.magnetic field = yes
 hw.sensors.orientation = yes
 hw.sensors.temperature = yes
  kernel.path = /root/adt-bundle-linux-x86 64-20130219/sdk/system-
images/android-17/armeabi-v7a//kernel-gemu
  kernel.parameters = androidboot.console=ttyS2 android.checkjni=1
  disk.ramdisk.path = /root/roms/emulator/newramdisk.cpio.gz
  disk.systemPartition.initPath = /root/roms/xoom/system-test.img
  disk.systemPartition.size = 253m
  disk.dataPartition.path = /root/.android/avd/tesis3.avd/userdata-
qemu.img
  disk.dataPartition.size = 2g
  avd.name = tesis3
QEMU options list:
emulator: argv[00] = "./emulator64-arm"
emulator: argv[01] = "-show-kernel"
emulator: argv[02] = "-serial"
emulator: argv[03] = "stdio"
emulator: argv[04] = "-android-hw"
           argv[05]
                      = "/root/.android/avd/tesis3.avd/hardware-
emulator:
qemu.ini"
Concatenated QEMU options:
 ./emulator64-arm -show-kernel
                                   -serial stdio -android-hw
/root/.android/avd/tesis3.avd/hardware-gemu.ini
emulator: registered 'boot-properties' qemud service
                                                       nand add dev:
emulator:
system,size=0xfd00000,initfile=/root/roms/xoom/system-test.img
emulator: mapping 'system' NAND image to /tmp/android-root/emulator-
WLLjr3
emulator: rounding devsize up to a full eraseunit, now fd0b000
```

```
emulator:
                                                    nand add dev:
userdata, size=0x80000000, file=/root/.android/avd/tesis3.avd/userdata
-gemu.img
emulator: rounding devsize up to a full eraseunit, now 80010000
emulator: registered 'boot-properties' qemud service
emulator: Adding boot property: 'dalvik.vm.heapsize' = '48m'
emulator: Adding boot property: 'qemu.sf.lcd density' = '320'
emulator: Adding boot property: 'qemu.hw.mainkeys' = '1'
emulator: Adding boot property: 'qemu.sf.fake camera' = 'back'
emulator:
                                                   nand add dev:
cache, size=0x4200000, file=/root/.android/avd/tesis3.avd/cache.img
emulator: Initializing hardware OpenGLES emulation support
Failed to create Context 0x3005
emulator: Can't start OpenGLES renderer?
emulator: WARNING: Could not initialize OpenglES emulation, using
software renderer.
emulator: Kernel parameters: qemu.gles=0 qemu=1 console=ttyS0
android.qemud=ttyS1 androidboot.console=ttyS2 android.checkjni=1
ndns=1
emulator: Trace file name is not set
emulator: autoconfig: -scale 1
emulator: Could not open file: (null)/system/build.prop: No such
file or directory
emulator: control console listening on port 5554, ADB on port 5555
emulator: sent '0012host:emulator:5555' to ADB server
            ping program: /root/adt-bundle-linux-x86 64-
emulator:
20130219/sdk/tools/ddms
emulator: ping command: /root/adt-bundle-linux-x86 64-
20130219/sdk/tools/ddms ping emulator 22.0.4.0 "" "" ""
Uncompressing
Linux.....
..... done, booting the kernel.
goldfish fb get pixel format:167: display surface, pixel format:
 bits/pixel: 16
 bytes/pixel: 2
            16
 depth:
            bits=5 mask=0xf800 shift=11 max=0x1f
 red:
 green:
            bits=6 mask=0x7e0 shift=5 max=0x3f
            bits=5 mask=0x1f shift=0 max=0x1f
 blue:
 alpha: bits=0 mask=0x0 shift=0 max=0x0
Initializing cgroup subsys cpu
Linux version 2.6.29-gea477bb (kroot@kennyroot.mtv.corp.google.com)
(gcc version 4.6.x-google 20120106 (prerelease) (GCC) ) #1 Wed Sep
26 11:04:45 PDT 2012
CPU: ARMv7 Processor [410fc080] revision 0 (ARMv7), cr=10c5387f
CPU: VIPT nonaliasing data cache, VIPT nonaliasing instruction cache
Machine: Goldfish
Memory policy: ECC disabled, Data cache writeback
```

```
Truncating RAM at 0000000-3fffffff to -33ffffff (vmalloc region
overlap).
Built 1 zonelists in Zone order, mobility grouping on. Total pages:
211328
Kernel
        command line: qemu.gles=0 qemu=1 console=ttyS0
android.qemud=ttyS1 androidboot.console=ttyS2 android.checkjni=1
ndns=1
Unknown boot option `qemu.gles=0': ignoring
Unknown boot option `android.qemud=ttyS1': ignoring
Unknown boot option `androidboot.console=ttyS2': ignoring
Unknown boot option `android.checkjni=1': ignoring
PID hash table entries: 4096 (order: 12, 16384 bytes)
Console: colour dummy device 80x30
Dentry cache hash table entries: 131072 (order: 7, 524288 bytes)
Inode-cache hash table entries: 65536 (order: 6, 262144 bytes)
Memory: 832MB = 832MB total
Memory: 840320KB available (2900K code, 707K data, 124K init)
Calibrating delay loop... 519.37 BogoMIPS (lpj=2596864)
Mount-cache hash table entries: 512
Initializing cgroup subsys debug
Initializing cgroup subsys cpuacct
Initializing cgroup subsys freezer
CPU: Testing write buffer coherency: ok
net namespace: 936 bytes
NET: Registered protocol family 16
bio: create slab <bio-0> at 0
NET: Registered protocol family 2
IP route cache hash table entries: 32768 (order: 5, 131072 bytes)
TCP established hash table entries: 131072 (order: 8, 1048576 bytes)
TCP bind hash table entries: 65536 (order: 6, 262144 bytes)
TCP: Hash tables configured (established 131072 bind 65536)
TCP reno registered
NET: Registered protocol family 1
checking if image is initramfs... it is
Freeing initrd memory: 160K
goldfish new pdev goldfish interrupt controller at ff000000 irg -1
goldfish new pdev goldfish device bus at ff001000 irg 1
goldfish_new_pdev goldfish_timer at ff003000 irq 3
goldfish new pdev goldfish rtc at ff010000 irq 10
goldfish new pdev goldfish tty at ff002000 irq 4
goldfish new pdev goldfish tty at ff011000 irq 11
goldfish new pdev goldfish tty at ff012000 irq 12
goldfish new pdev smc91x at ff013000 irq 13
goldfish new pdev goldfish fb at ff014000 irq 14
goldfish new pdev goldfish audio at ff004000 irq 15
goldfish new pdev goldfish memlog at ff006000 irq -1
goldfish new pdev goldfish-battery at ff015000 irg 16
goldfish new pdev goldfish events at ff016000 irq 17
goldfish new pdev goldfish nand at ff017000 irq -1
goldfish new pdev qemu pipe at ff018000 irq 18
goldfish new pdev goldfish-switch at ff01a000 irq 19
                       _____
```

Desarmando al Androide

```
goldfish pdev worker registered goldfish interrupt controller
goldfish pdev worker registered goldfish device bus
goldfish pdev worker registered goldfish timer
goldfish pdev worker registered goldfish rtc
goldfish pdev worker registered goldfish tty
goldfish pdev worker registered goldfish tty
goldfish pdev worker registered goldfish tty
goldfish pdev worker registered smc91x
goldfish_pdev_worker registered goldfish fb
goldfish pdev worker registered goldfish audio
goldfish pdev worker registered goldfish memlog
goldfish pdev worker registered goldfish-battery
goldfish pdev worker registered goldfish events
goldfish pdev worker registered goldfish nand
goldfish pdev worker registered qemu pipe
goldfish pdev worker registered goldfish-switch
goldfish pdev worker registered goldfish-switch
ashmem: initialized
Installing knfsd (copyright (C) 1996 okir@monad.swb.de).
yaffs Sep 26 2012 11:04:43 Installing.
msgmni has been set to 1642
alg: No test for stdrng (krng)
io scheduler noop registered
io scheduler anticipatory registered (default)
io scheduler deadline registered
io scheduler cfq registered
allocating frame buffer 1280 * 720, got ffa00000
console [ttyS0] enabled
brd: module loaded
loop: module loaded
nbd: registered device at major 43
goldfish audio probe
tun: Universal TUN/TAP device driver, 1.6
tun: (C) 1999-2004 Max Krasnyansky <maxk@qualcomm.com>
smc91x.c: v1.1, sep 22 2004 by Nicolas Pitre <nico@cam.org>
eth0 (smc91x): not using net device ops yet
eth0: SMC91C11xFD (rev 1) at f480c000 IRQ 13 [nowait]
eth0: Ethernet addr: 52:54:00:12:34:56
goldfish nand dev0: size f560000, page 2048, extra 64, erase 131072
goldfish nand dev1: size 7c200000, page 2048, extra 64, erase 131072
goldfish nand dev2: size 4000000, page 2048, extra 64, erase 131072
mice: PS/2 mouse device common for all mice
*** events probe ***
events probe() addr=0xf4814000 irq=17
events probe() keymap=qwerty2
input: qwerty2 as /devices/virtual/input/input0
goldfish rtc goldfish rtc: rtc core: registered goldfish rtc as rtc0
device-mapper: uevent: version 1.0.3
device-mapper: ioctl: 4.14.0-ioctl (2008-04-23) initialised: dm-
devel@redhat.com
logger: created 64K log 'log main'
```

```
logger: created 256K log 'log events'
logger: created 64K log 'log radio'
Netfilter messages via NETLINK v0.30.
nf conntrack version 0.5.0 (13312 buckets, 53248 max)
CONFIG NF CT ACCT is deprecated and will be removed soon. Please use
nf_conntrack.acct=1 kernel paramater, acct=1 nf_conntrack module
option or
sysctl net.netfilter.nf conntrack acct=1 to enable it.
ctnetlink v0.93: registering with nfnetlink.
NF TPROXY: Transparent proxy support initialized, version 4.1.0
NF TPROXY: Copyright (c) 2006-2007 BalaBit IT Ltd.
xt time: kernel timezone is -0000
ip tables: (C) 2000-2006 Netfilter Core Team
arp tables: (C) 2002 David S. Miller
TCP cubic registered
ip6 tables: (C) 2000-2006 Netfilter Core Team
IPv6 over IPv4 tunneling driver
NET: Registered protocol family 17
NET: Registered protocol family 15
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
802.10 VLAN Support v1.8 Ben Greear <greearb@candelatech.com>
All bugs added by David S. Miller <davem@redhat.com>
VFP support v0.3: implementor 41 architecture 3 part 30 variant c
rev 0
goldfish rtc goldfish rtc: setting system clock to 2013-10-06
21:42:47 UTC (1381095767)
Freeing init memory: 124K
init: cannot open '/initlogo.rle'
yaffs: dev is 32505856 name is "mtdblock0"
yaffs: passed flags ""
yaffs: Attempting MTD mount on 31.0, "mtdblock0"
yaffs read super: isCheckpointed 0
save exit: isCheckpointed 0
yaffs: dev is 32505857 name is "mtdblock1"
yaffs: passed flags ""
yaffs: Attempting MTD mount on 31.1, "mtdblock1"
hrtimer: interrupt too slow, forcing clock min delta to 26386311 ns
yaffs read super: isCheckpointed 0
yaffs: dev is 32505858 name is "mtdblock2"
yaffs: passed flags ""
yaffs: Attempting MTD mount on 31.2, "mtdblock2"
yaffs read super: isCheckpointed 0
eth0: link up
shell@android:/ $ warning: `rild' uses 32-bit capabilities (legacy
support in use)
goldfish fb pan display: timeout wating for base update
request suspend state: wakeup (3->0) at 120060793619 (2013-10-06
21:44:46.941441593 UTC)
init: untracked pid 33 exited
```

<pre>request_suspend_state: w 21:45:32.521654980 UTC)</pre>	vakeup	(0->0)	at	165640770376	(2013-10-06
init: untracked pid 146 e	exited				
request_suspend_state: w	akeup	(0->0)	at	212188669882	(2013-10-06
init: untracked pid 216 e	exited				
init: untracked pid 215 e	exited				

Bibliografía

- [1] "Open Handset Alliance," 12 11 2007. [Online]. Available: http://www.openhandsetalliance.com/. [Accessed 12 03 2013].
- [2] Lookout, "State of Mobile Security," 2012. [Online]. Available: https://www.lookout.com/static/ee_images/lookout-state-of-mobilesecurity-2012.pdf. [Accessed 10 07 2013].
- [3] K. Parmar, "In Depth : Android Boot Sequence / Process," 08 11 2012.
 [Online]. Available: http://www.kpbird.com/2012/11/in-depth-android-boot-sequence-process.html. [Accessed 02 04 2013].
- [4] K. Parmar, "In Depth : Android Boot Sequence / Process," 08 11 2012.
 [Online]. Available: http://www.kpbird.com/2012/11/in-depth-android-boot-sequence-process.html. [Accessed 20 03 2013].
- [5] Google, "Android Debug Bridge," [Online]. Available: http://developer.android.com/tools/help/adb.html. [Accessed 15 03 2013].
- [6] "ClockworkMod," [Online]. Available: http://www.clockworkmod.com/.[Accessed 20 03 2013].
- [7] Desconocido, "HOWTO: Unpack, Edit, and Re-Pack Boot Images," 1 2
 2013. [Online]. Available: http://androiddls.com/wiki/index.php?title=HOWTO:_Unpack,_Edit,_and_Re-Pack_Boot_Images. [Accessed 20 03 2013].
- [8] W. Enck, "William Enck Tools," [Online]. Available: http://www.enck.org/tools.html. [Accessed 16 03 2013].
- [9] tvidas, "Live View," Carnegie Mellon University, [Online]. Available: http://liveview.sourceforge.net/.

[1 Google, "Android Emulator,"[Online].Available:0] http://developer.android.com/tools/help/emulator.html.

A. Hoog, Android Forensics: Investigation, Analysis, and Mobile Security
 for Google Android, ELSEVIER, 2011.

J. Lessard and G. C. Kessler, "Android Forensics: Simplifying Cell Phone
 Examinations VOL. 4, NO.1," Septiembre 2010.

- [1 W. Jansen and R. Ayers, "Guidelines on Cell Phone Forensics,
- Recommendations of the National Institute of Standards and Technology,
 Special Publication 800-101," [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-101/SP800-101.pdf.
- [1 M. Bjornheden, "The Android boot process from power on," 11 06 2009.
- 4] [Online]. Available: http://www.androidenea.com/2009/06/android-bootprocess-from-power-on.html. [Accessed 01 03 2013].
- [1 V. VIJAYAN, "Android Forensic Capability and Evaluation of Extraction
- 5] Tools," 04 2012. [Online]. Available: http://academia.edu/1632597/Android_Forensic_Capability_and_Evaluati on_of_Extraction_Tools. [Accessed 10 03 2013].
- [1 Laurent, "Xoom," 15 03 2012. [Online]. Available:
- 6] http://p.quinput.eu/qwiki/Wiki.jsp?page=Xoom. [Accessed 25 03 2013].
- [1 "Under the hood of Android Emulator (appcert)," 21 07 2011. [Online].7] Available:

https://wiki.diebin.at/Under_the_hood_of_Android_Emulator_(appcert). [Accessed 15 04 2013].

- [1 "ISO/IEC 27037:2012 Information technology -- Security techniques --
- 8] Guidelines for identification, collection, acquisition and preservation of digital evidence," ISO, 15 10 2012. [Online]. Available: http://www.iso.org/iso/catalogue_detail?csnumber=44381. [Accessed 01

03 2013].