Investigación forense de dispositivos móviles

Android

Francisco Lázaro Domínguez

Ra-Ma®

Investigación forense de dispositivos móviles Android

Francisco Lázaro Domínguez

Ra-Ma®

La ley prohíbe
Copiar o Imprimir este libro

Investigación forense de dispositivos móviles Android
© Francisco Lázaro Domínguez

© De la Edición Original en papel publicada por Editorial RA-MA
ISBN de Edición en Papel: 978-84-9964-520-9

Editado por:
RA-MA, S.A. Editorial y Publicaciones
Calle Jarama, 33, Polígono Industrial IGARSA
28860 PARACUELLOS DE JARAMA, Madrid
Teléfono: 91 658 42 80
Fax: 91 662 81 39
Correo electrónico: editorial@ra-ma.com
Internet: www.ra-ma.es y www.ra-ma.com

Maquetación y diseño portada: Antonio García Tomé

ISBN: 978-84-9964-497-4

E-Book desarrollado en España en Octubre de 2015

La ley prohíbe
Copiar o Imprimir este libro

Investigación forense de dispositivos móviles Android
© Francisco Lázaro Domínguez

© De la Edición Original en papel publicada por Editorial RA-MA
ISBN de Edición en Papel: 978-84-9964-520-9

Editado por:
RA-MA, S.A. Editorial y Publicaciones
Calle Jarama, 33, Polígono Industrial IGARSA
28860 PARACUELLOS DE JARAMA, Madrid
Teléfono: 91 658 42 80
Fax: 91 662 81 39
Correo electrónico: editorial@ra-ma.com
Internet: www.ra-ma.es y www.ra-ma.com

Maquetación y diseño portada: Antonio García Tomé

ISBN: 978-84-9964-497-4

E-Book desarrollado en España en Octubre de 2015

A ti, apreciado lector,
por haber tenido la deferencia de
dedicar parte de tu presupuesto formativo
a la adquisición de este libro.
También por el esfuerzo que estás dispuesto
a hacer para leerlo,
y porque sospecho que eres de esas personas
capaces de trabajar mientras
el resto del mundo se divierte.

ÍNDICE

INTRODUCCIÓN **15**

CAPÍTULO 1. TECNOLOGÍA DE DISPOSITIVOS MÓVILES **21**

1.1 UN ORDENADOR EN EL BOLSILLO 21
1.2 HARDWARE 22
1.2.1 Microprocesador 24
1.2.2 Comunicaciones móviles de banda ancha 25
1.2.3 Conectividad wifi y Bluetooth 25
1.2.4 Memoria RAM/NAND 26
1.2.5 Tarjeta MicroSD 27
1.2.6 Pantalla 28
1.2.7 Teclado 28
1.2.8 GPS 28
1.2.9 Cámara 29
1.2.10 Micrófono 30
1.2.11 Giróscopo y acelerómetro 30
1.2.12 Batería 30
1.2.13 Interfaz USB 31
1.2.14 Componentes de otros tipos 33
1.3 SOFTWARE 33
1.3.1 Orígenes de Android 34
1.3.2 Open Handset Alliance 35
1.3.3 Intereses de Google 36
1.3.4 Android Open Source Project 37
1.3.5 Versiones de Android 38
1.3.6 API 39
1.3.7 Android Market 40

1.4 ARQUITECTURA DEL SISTEMA 42
1.4.1 Kernel 42
1.4.2 Binder IPC 42
1.4.3 Librerías 43
1.4.4 Android Runtime y Dalvik 44
1.4.5 Entorno de aplicaciones 45
1.4.6 Aplicaciones de usuario 45
1.5 ¿CÓMO ES UNA APLICACIÓN ANDROID? 46
1.5.1 Manifiesto 46
1.5.2 Activities 47
1.5.3 Intents 47
1.5.4 Broadcast receivers 48
1.5.5 Proveedores de contenidos 48
1.5.6 Servicios 48
1.6 SISTEMAS DE ARCHIVOS 49
1.6.1 YAFFS/YAFFS2 49
1.6.2 FAT32 50
1.6.3 Extended file system (ext) 51
1.6.4 rootfs, devtps, sysfs, proc, tmpfs 52

CAPÍTULO 2. PREPARATIVOS Y HERRAMIENTAS 53

2.1 MÁQUINAS VIRTUALES 53
2.1.1 ¿Por qué utilizar máquinas virtuales? 54
2.1.2 Principios de virtualización 55
2.2 VMWARE WORKSTATION 57
2.2.1 Características 58
2.2.2 Localización, descarga y recursos web 58
2.2.3 Creación de máquinas virtuales 59
2.3 VIRTUALBOX 61
2.3.1 Características 62
2.3.2 Instalación en Windows 63
2.3.3 Soporte USB 64
2.3.4 Instalación en Linux 65
2.3.5 Instalación en OSX 67
2.4 ANDROID SDK 67
2.4.1 Niveles de API y partes sobrantes 67
2.4.2 Instalación en Windows 69
2.4.3 Instalación en Linux 71
2.4.4 Instalación en OSX 72
2.4.5 Emulación de dispositivos Android 72
2.5 LINUX 74
2.5.1 Listado de directorios con ls 75

2.5.2 Ayuda mediante man 75
2.5.3 mkdir y rmdir 76
2.5.4 cd 76
2.5.5 less 77
2.5.6 grep 77
2.5.7 Canalización de comandos 78
2.5.8 Redireccionamiento 78
2.5.9 find 79
2.5.10 sudo 79
2.5.11 apt-get 80
2.5.12 cp y mv 81
2.5.13 chmod y chown 81
2.5.14 file 82

CAPÍTULO 3. TÉCNICAS DE INVESTIGACIÓN FORENSE 85

3.1 DESAFÍOS DE LA MOVILIDAD INFORMÁTICA 85
3.1.1 Privacidad 86
3.1.2 Estrategia gradual 87
3.2 PREVISUALIZACIÓN 88
3.2.1 Teléfono 89
3.2.2 Mensajes 90
3.2.3 Correo electrónico 91
3.2.4 Fecha y hora 91
3.2.5 Imágenes y vídeos 92
3.2.6 Mapas 92
3.2.7 Archivos 92
3.2.8 Aplicaciones 93
3.2.9 Administrador de tareas 95
3.2.10 Ajustes 95
3.2.11 Otros contenidos 96
3.3 TARJETA DE MEMORIA 97
3.3.1 Adquisición con dd 97
3.3.2 Hash 99
3.3.3 Montaje de la imagen 99
3.3.4 Análisis forense con TSK 100
3.4 DATA CARVING 103
3.5 ACCESO MEDIANTE ADB 105
3.5.1 Interfaces USB 105
3.5.2 Depuración USB 106
3.5.3 Funcionamiento de ADB 107
3.5.4 Empleo de ADB 109

3.6 EXTRAYENDO INFORMACIÓN CON ADB 113
3.6.1 dmesg 113
3.6.2 logcat 114
3.6.3 dumpsys 115
3.6.4 dumpstate 115
3.6.5 bugreport 116
3.7 ROOTING 116
3.7.1 Riesgos y precauciones del rooting 117
3.7.2 Norma fundamental de prudencia 118
3.7.3 ¿Qué es exactamente el rooting? 118
3.7.4 su y SuperUser 120
3.7.5 Rooting temporal 121
3.7.6 Psneuter 122
3.7.7 Rooting permanente 124
3.7.8 XDA-Developers 127

CAPÍTULO 4. TÉCNICAS ESPECIALES DE INVESTIGACIÓN 129
4.1 CADENA DE CUSTODIA 129
4.1.1 Aislamiento de redes 130
4.1.2 Terminales bloqueados 133
4.1.3 Smudge attack 135
4.1.4 Prueba de conexión ADB 136
4.1.5 Ejemplos 137
4.1.6 Eliminación del código de bloqueo 137
4.2 PARTICIONES RECOVERY 138
4.2.1 Particionado NAND 139
4.2.2 Modos de funcionamiento 143
4.2.3 Secuencia de arranque 144
4.2.4 ClockWorkMod 146
4.2.5 Instalación de CWM en un Samsung Galaxy S2 148
4.2.6 Adquisición lógica con CWM 149
4.2.7 Adquisición física con CWM 151
4.2.8 Adquisición física mediante ADB 154
4.3 ADQUISICIÓN POR HARDWARE 154
4.3.1 Chip-off 155
4.3.2 JTAG 155

CAPÍTULO 5. ANÁLISIS FORENSE 157
5.1 ESTRUCTURAS DE DATOS 157
5.1.1 Aplicaciones 157
5.1.2 Datos de usuario 158
5.1.3 Métodos de archivado 159

5.1.4 Jerarquía de directorios Linux 160
5.1.5 Datos de usuario 161
5.1.6 Formatos 163
5.2 XML 163
5.2.1 ¿Qué es XML? 163
5.2.2 Manipulación de archivos XML 164
5.2.3 Ejemplo de análisis 166
5.2.4 Apktool 168
5.3 INGENIERÍA INVERSA 169
5.3.1 Descompilando código con dex2jar 170
5.3.2 Análisis de malware 171
5.4 SQLITE 172
5.4.1 Carácter compacto y portable 173
5.4.2 Análisis de archivos .db 173
5.4.3 Ejemplo de utilización 175
5.5 ANÁLISIS DE APLICACIONES ANDROID 176
5.5.1 Mensajes SMS y MMS 177
5.5.2 Navegador de Internet 178
5.5.3 Contactos 179
5.5.4 Media Scanner 179
5.5.5 Youtube 181
5.5.6 Google Maps 181
5.5.7 Gmail 182
5.5.8 Otras aplicaciones 182
5.6 HERRAMIENTAS DE ANÁLISIS 183
5.6.1 Editores hexadecimales 183
5.6.2 Strings 186
5.7 YAFFS/YAFFS2 187

CAPÍTULO 6. SEGURIDAD 189
6.1 MODELO DE SEGURIDAD ANDROID 189
6.1.1 Retos de seguridad en la era móvil 190
6.1.2 Certificados digitales y supervisión 191
6.1.3 Máquina virtual Dalvik 192
6.1.4 Gestión de usuarios 193
6.1.5 Permisos de archivos 194
6.1.6 Permisos de aplicaciones 194
6.2 ATAQUES CONTRA ANDROID 195
6.2.1 Ataques contra la infraestructura 197
6.2.2 Ataques contra el hardware 197
6.2.3 Ataques contra el software 198

6.3 VULNERABILIDADES ANDROID 199
6.3.1 Vulnerabilidades de permisos nulos 200
6.3.2 Escalada de privilegios 202
6.3.3 Ejecución de shell inverso 203
6.3.4 Grietas funcionales (capability leaks) 203
6.4 SOFTWARE MALICIOSO 204
6.4.1 Motivación del delincuente 204
6.4.2 Tipos de malware 205
6.5 VECTORES DE ATAQUE 206
6.5.1 Google Play 206
6.5.2 Mercados no oficiales 207
6.5.3 Internet 208
6.5.4 Infecciones combinadas dispositivo móvil-PC 208
6.5.5 Problemática de los dispositivos rooteados 209
6.6 EJEMPLOS DE APLICACIONES MALICIOSAS 210
6.6.1 Suscripción a servicios SMS premium 211
6.6.2 Spyware 212
6.6.3 Adware 213
6.6.4 Pay-per-click 213
6.6.5 Envenenamiento de buscadores 213
6.6.6 Troyanos bancarios 214
6.7 ESPIONAJE CORPORATIVO 215
6.7.1 Hardware 215
6.7.2 Herramientas de software 216
6.7.3 Modus operandi 217
6.7.4 Implicaciones 220

CAPÍTULO 7. SOLUCIONES INTEGRADAS 221
7.1 VENTAJAS DEL SOFTWARE COMERCIAL 221
7.2 CELLEBRITE UFED TOUCH 223
7.2.1 Adquisición 224
7.2.2 Análisis forense 225
7.2.3 Cellebrite Touch y Android 226
7.3 OXYGEN FORENSIC SUITE 227
7.3.1 Instalación y manejo 228
7.3.2 Elementos recuperables 230
7.3.3 Conexión de dispositivos Android 230
7.4 ADEL 231
7.4.1 Principios de diseño 232
7.4.2 Implementación y funcionamiento 233
7.4.3 Otras características 234

CAPÍTULO 8. CONCLUSIÓN 235
8.1 ANÁLISIS DE DISPOSITIVOS APPLE IOS 236
8.1.1 Similitudes y diferencias 236
8.1.2 Carácter exclusivo 237
8.1.3 iTunes y AppStore 238
8.1.4 Jailbreaking 239
8.1.5 Criptografía hardware 240
8.1.6 Particiones y sistemas de archivos Apple HFS+ 241
8.1.7 Precaución con iTunes 242
8.2 EQUIPAMIENTO Y MATERIALES 243
8.2.1 Hardware 243
8.2.2 Software 246
8.2.3 Dispositivos para bloqueo de la señal electromagnética 246
8.3 INVESTIGACIÓN CONVENCIONAL 248
8.3.1 Mundo virtual y mundo físico 248
8.3.2 Interrogatorios y tomas de declaración 249
8.3.3 Objetivos de los procedimientos convencionales 254
8.3.4 Misión del investigador forense 255
8.4 OTROS ASPECTOS DE LA INVESTIGACIÓN 256
8.4.1 Realización y presentación de informes 256
8.4.2 Implicaciones jurídicas 256

BIBLIOGRAFÍA 259

ÍNDICE ALFABÉTICO 261

INTRODUCCIÓN

Potencia y capacidades de Android

¿Por qué un libro sobre investigación forense de dispositivos móviles Android? No haría falta siquiera explicarlo. El lector[1] hallará la respuesta cuando vaya al trabajo en metro mañana por la mañana y vea a toda esa gente inclinada sobre la pantalla de sus *smartphones*, hablando por teléfono, leyendo correos electrónicos o noticias, jugando, escuchando música, viendo una película, mandando mensajes WhatsApp o tuiteando. La mayor parte de los dispositivos que utilizan nuestros congéneres del ecosistema urbanita y globalizado de la segunda década del siglo XXI funcionan con un sistema operativo de código abierto que se llama Android. Android, lejos de ser un simple pasatiempo, se halla presente en numerosos ámbitos de la actividad moderna: empresa, educación, seguridad, servicios sanitarios, maquinaria, cámaras digitales, comercio electrónico, web 2.0 y un largo etcétera. A diferencia de otras plataformas para *smartphones* y teléfonos móviles que se distribuyen para hacer feliz a una comunidad de usuarios sin grandes ambiciones tecnológicas, Android admite gran variedad de usos y aplicaciones que convierten a un terminal en una potente herramienta para la delincuencia informática y el *hacking*.

También podemos liberar teléfonos móviles y hacer *jailbreaking* en un Apple iPhone, pero no es nada comparado con las posibilidades que son la norma en la escena Android para programar, manipular, customizar y dar un uso creativo a los terminales móviles. Además, el desarrollador del software y el fabricante se

1 Este libro está escrito con una intención neutra en cuanto al género. El empleo del masculino, cuando aparece asociado a personas, como por ejemplo el lector, el investigador, el sospechoso, usuarios y desarrolladores, etc., hace referencia a hombres y mujeres por igual.

muestran tolerantes ante cualquier intento de modificar el producto en favor del usuario, dejando que los dispositivos salgan desbloqueados de fábrica y facilitando información en foros y páginas web. Un Google Nexus o un Sony Xperia pueden funcionar como mando a distancia de televisores y otros aparatos, escáner de redes inalámbricas, *wardriver*,[2] punto de acceso para conexiones compartidas, servidor de archivos y páginas web, control remoto para drones y robots o cualquier otra aplicación que se nos ocurra, desde el lado correcto de la ley o desde el otro. ¿Le interesa ahora? Hace años vimos algo parecido con Linux, los escáneres de radiofrecuencia y *lockpicking* o arte popular de abrir cerraduras con ganzúas: si Android es importante para los hackers, también lo es para un investigador forense. Y por supuesto, para juristas, criminólogos o peritos tecnológicos.

Con un terminal Android el usuario está en condiciones de elegir entre diversos modos de vida. Puede ser un simple consumidor de tecnología para el entretenimiento. También puede trabajar, aprender, dar salida a una vocación de cooperante, formarse como experto en robótica y, lamentablemente, también vulnerar la ley. Android constituye una revolución auténticamente democrática: el libre albedrío en forma de dispositivo transportable. Este amplio espectro de posibilidades individuales tiene por detrás una interesante historia que ha condicionado en medida significativa el desarrollo de la sociedad y la economía.

Impacto de las tecnologías de la información

No hace falta insistir en el considerable impacto que las tecnologías digitales han tenido para la vida moderna. Desde 1970, aproximadamente, la aplicación de principios de abstracción y modularidad ha hecho posible la elaboración de soluciones de proceso de datos para todo tipo de problemas relacionados con el tratamiento de la información: sistemas de contabilidad y reservas de billetes, gestión de recursos humanos, censos, archivos policiales, bibliotecas, funcionamiento de las administraciones públicas y lucha contra el fraude. La aparición del ordenador personal a comienzos de los 80 capilarizó esta tendencia, permitiendo que los avances de la informática estuvieran al alcance no solo de gobiernos y grandes empresas —entonces únicos compradores potenciales de *mainframes* y miniordenadores—, sino también de pymes y particulares.

En los últimos años, Internet y la informática móvil han reinventado la revolución que comenzó hace cuatro décadas con la llegada al mercado de los primeros microprocesadores. El sueño visionario de Bill Gates de un ordenador en

2 El *wardriving* es una práctica consistente en localizar puntos de acceso inalámbrico desplazándose de un lado a otro con dispositivos móviles provistos de adaptadores wifi.

cada hogar se ha hecho realidad. Si además de los PC de sobremesa y ordenadores portátiles incluimos terminales móviles (*smartphones*, agendas electrónicas, tabletas, calculadoras programables, reproductores multimedia con acceso a Internet), electrodomésticos con controlador digital (televisores, equipos de alta fidelidad, frigoríficos, lavadoras, etc.), sistemas domóticos y de aire acondicionado, máquinas-herramienta, cajeros electrónicos, relojes de pulsera, cámaras digitales y otros aparatos, nos encontraremos con que en el mundo ya hay más ordenadores —considerando como tales a los dispositivos provistos de CPU, memoria RAM y un sistema de entradas y salidas— que seres humanos.

Podemos hablar de una una nueva esfera, superpuesta a las que constituyen el objeto de estudio de la geografía académica. Junto a la geosfera, la atmósfera, la biosfera, la litosfera y la magnetosfera propias del mundo natural, con sus ciclos característicos de las rocas, el carbono o el nitrógeno, y coronando la tecnosfera creada por el ser humano como resultado de la Revolución Industrial, existe ahora una infosfera, caracterizada por el flujo de una corriente de datos que genera valor económico y nos permite actuar sobre el mundo real en formas que hace años parecían inimaginables. Una oleada de innovaciones como la computación en la nube, la Internet de las cosas y la Internet industrial, con la posibilidad de manejar aviones, maquinaria, edificios, depuradoras y redes de suministro eléctrico a través de navegadores web, sugiere que nos encontramos a mitad de camino y tardaremos aún mucho tiempo en conocer el final del proceso.

Justificación de la obra

El análisis forense de dispositivos móviles es importante por varias razones, en primer lugar por su protagonismo en la escena delictiva y en todo tipo de conflictos entre particulares y empresas. Los *smartphones*, auténticos ordenadores de mano equipados con diversas interfaces de comunicación inalámbrica a redes públicas y privadas, memoria de almacenamiento interna y externa, giróscopo y sensor de inercia, tienen más potencia que los PC de sobremesa de hace cinco años y se ven involucrados en la escena criminal de variadas formas: como objetivo de ataque para la instalación de *malware*, el robo de datos y las intrusiones en redes. También como objeto de valor susceptible de ser sustraído y, sobre todo, como herramienta de piratería informática. Con un *smartphone* es posible enviar anónimos, transportar información confidencial y llevar a cabo otras acciones ilícitas.

El estudio forense de dispositivos móviles constituye una necesidad perentoria para las fuerzas de seguridad del Estado. Un *smartphone* plantea dificultades que afectan a principios fundamentales de la informática forense "clásica", como por ejemplo el requisito de no alterar las pruebas. En la mayor parte de los casos resulta imposible realizar adquisiciones forenses con un terminal que

no esté en funcionamiento, aparte de otras circunstancias que se irán viendo en los apartados correspondientes a la tecnología del *smartphone* y los procedimientos de investigación. Los terminales portátiles, por su proximidad a la persona que los utiliza, contienen información que afecta a su esfera de privacidad y a derechos fundamentales reconocidos por la Constitución.

Estructura y propósito

En un primer capítulo se estudia la tecnología de los *smartphones*, los orígenes de Android y la arquitectura del hardware y el sistema operativo. A continuación se describen los preparativos básicos para el análisis forense de dispositivos móviles. Posteriormente se hablará de métodos y técnicas utilizados para la extracción de datos y elementos de evidencia. También se incluye un capítulo dedicado a la seguridad y al modus operandi de los ciberdelincuentes en el ámbito de la informática móvil, con el objeto de que el investigador se haga idea de cómo son las cosas en un escenario real. Posteriormente se hablará de algunos productos comerciales destacados en el campo de la informática forense de dispositivos móviles, y, como término de comparación, también de ADEL, un modelo de concepto basado en código abierto. Para terminar, dentro del capítulo final dedicado a las conclusiones, se incluye un apartado sobre investigación de dispositivos móviles Apple iOS, exponiendo las principales diferencias y similitudes entre las dos plataformas.

El micromundo del *smartphone* —con Android, iOS o cualquier otro sistema operativo para dispositivos móviles— es complejo y bastante incómodo para el investigador. Si nos encontramos en proceso de transición desde una línea de actividad tradicional centrada en la adquisición de discos duros y soportes de datos mediante procedimientos de manual y herramientas certificadas, como FTK Imager o EnCase Forensics, a un escenario de investigación de dispositivos móviles, resulta inevitable la inquietud derivada de escenarios en los que no se puede garantizar la integridad absoluta de la prueba, donde el simple hecho de analizarla la altera y el sospechoso es capaz de destruir remotamente el contenido incriminador de su teléfono, aun después de que el dispositivo haya sido incautado. Al investigador también le conviene estar familiarizado con los aspectos jurídicos de su profesión. Un defecto de procedimiento en la búsqueda de elementos de evidencia puede traer consigo la anulación de la prueba, por considerarse que no se ha actuado con profesionalidad o que los derechos constitucionales del inculpado están siendo vulnerados. En las páginas que siguen, el lector encontrará indicaciones para manipular terminales y acceder al contenido de los mismos, y también motivos para no hacerlo, en caso de no darse las condiciones oportunas.

El lector con inquietudes técnicas, interesado en la extracción de datos o el funcionamiento de los sistemas, no debe saltarse los apartados jurídicos pensando

que se trata de algo meramente accesorio, como las referencias a directivas europeas o normas DIN. El ámbito de actividad del investigador informático forense, y de los peritos de cualquier especialidad tecnológica, no se rige por el pensamiento lineal del ingeniero, sino por las alambicadas categorías del derecho. El riesgo no se encuentra en los productos defectuosos sino en el interior de la sala. Los casos no naufragan por falta de pericia sino por fallas de procedimiento aparentemente imperceptibles, parecidas a aquellas grietas en el casco soldado de los barcos *Liberty* de la Segunda Guerra Mundial que en el astillero no eran detectadas por el control de calidad, pero que después, sometidas al vaivén del oleaje y a las tensiones internas de la estructura, se extendían en una fracción de segundo hasta romper el buque en dos, llevándoselo al fondo con toda su tripulación y su carga. El objetivo no consiste en explicar al investigador el funcionamiento de los dispositivos, sino en enseñarle a hacer uso de la prudencia y el sentido común en una tarea que, al margen de las herramientas utilizadas, consiste principalmente en un proceso mental basado en la coherencia lógica y la persecución de una meta esencial: aportar elementos de evidencia obtenidos de acuerdo con la ley, analizados de manera correcta y presentados en tal modo que los responsables de tomar decisiones judiciales puedan hacerlo en términos adecuados y con un conocimiento preciso de los problemas técnicos subyacentes.

Interés público de la materia

En la actualidad las tecnologías forenses resultan imprescindibles para atender a las necesidades de la administración de Justicia, las fuerzas de seguridad y los departamentos de seguridad de las empresas, tanto para la defensa de los derechos del ciudadano como de intereses particulares. También existen otras consideraciones que justifican la inversión de tiempo y recursos en el estudio de estas especialidades. A lo largo de los últimos años el lector habrá tenido ocasión de leer noticias relativas a los ciberataques contra las Repúblicas Bálticas y Chechenia, los daños causados por el gusano Stuxnet o la infiltración de hackers chinos en redes occidentales en busca de tecnología y activos de propiedad intelectual. Ciberguerra, espionaje informático, delincuencia organizada y la posibilidad de ataques contra infraestructuras críticas son fuentes constantes de preocupación para los gobiernos.

Siempre será necesario que alguien mantenga libres de piratas las rutas del comercio y la información. La autoridad pública debe disponer de medios para hacer frente a la amenaza cibernética. Pese a que vivimos en un mundo globalizado y regido por estructuras transnacionales, o quizás precisamente por ello, el moderno Estado de Derecho, con su administración centralizada, su capacidad coordinada de respuesta ante amenazas externas y para participar en compromisos de seguridad colectiva, es el único agente capaz de defender la civilización ante las agresiones del terrorismo y el crimen organizado. Por este motivo le conviene mejorar su capacidad

de respuesta con el apoyo de gremios especializados en tecnologías de la información, con el propósito de hacer frente a amenazas de todo tipo en el terreno del conflicto asimétrico, la delincuencia y el sabotaje. La seguridad de un país se verá mejorada por la existencia de un colectivo numeroso y bien formado de peritos judiciales e investigadores forenses. Un episodio que ilustra a la perfección esta necesidad es la derrota moral de la red china de hackers dedicada a la infiltración en redes europeas y norteamericanas en busca de tecnología y propiedad intelectual, detectada y puesta al descubierto por los profesionales de la empresa Mandiant.

La razón de haber escrito esta obra es la necesidad de disponer de información sobre el tema en una época en que una parte considerable de la actividad de los usuarios se traslada desde el ordenador de sobremesa al entorno móvil. ¿Por qué *smartphones* y tabletas son tan conflictivos y generan una escena delictiva tan compleja, obligando a investigadores forenses y departamentos de delitos tecnológicos a hacer grandes esfuerzos para ponerse al día? La movilidad digital es un proceso que acelera la denominada tercera Revolución Industrial, democratizándola y poniéndola al alcance de pequeñas empresas e individuos particulares. Este proceso comenzó hace tiempo y en última instancia no es más que el resultado de una singular alianza, establecida a comienzos de los años 70 del pasado siglo xx, entre máquinas capaces de procesar números y máquinas capaces de transmitirlos a distancia. Por sí solos ni el ordenador como dispositivo de tratamiento automatizado de la información, ni las redes de telecomunicaciones, con su entramado de fibra óptica y sus conmutadores de alta velocidad, explican las grandes transformaciones de nuestro tiempo. Es la combinación sinérgica de ambos elementos lo que da lugar a la revolución digital. Roma no alcanzó la grandeza por sus legiones o por sus calzadas, sino por las dos cosas *juntas*.

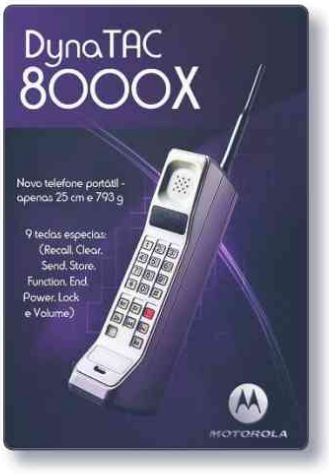

Figura 1.1. Motorola Dynatac 8000, primer teléfono móvil "transportable"

1

TECNOLOGÍA DE DISPOSITIVOS MÓVILES

El primer teléfono móvil comercial de tipo parecido a los que tenemos en la actualidad (Figura 1.1) fue el Dynatac 8000X de Motorola, homologado en 1983 por la Comisión Nacional de Telecomunicaciones de Estados Unidos. Pesaba 800 gramos y su autonomía era de 8 horas en espera. Se necesitaban 10 horas para cargar la batería; después su propietario podía hablar durante 30 minutos. Costaba casi 4.000 dólares (de los de entonces). Eran muy pocos los que se podían permitir un aparato como este, y, lógicamente, no llegó a convertirse en catalizador de una cultura popular de masas. Tres décadas más tarde, el desarrollo de la tecnología ha hecho con los teléfonos móviles algo parecido a lo que la evolución hizo con la fauna primitiva: convertir especímenes acorazados y torpones en organismos ligeros y gráciles provistos de finas carcasas de plástico y una inteligencia electrónica más desarrollada.

A comienzos del siglo XXI, más o menos a los dos tercios del trayecto recorrido entre el Dynatac 8000X y los últimos modelos de Nokia o Samsung que salieron al mercado antes de que el teléfono móvil se extinguiese para dejar paso al *smartphone*, es cuando sucede lo más interesante.

1.1 UN ORDENADOR EN EL BOLSILLO

Hacia el año 2000 la tendencia a la miniaturización, que parecía irreversible y conducía aparentemente al desarrollo de teléfonos del tamaño de un bolígrafo o una goma de borrar, se vio contrarrestada por necesidades de carácter puramente ergonómico: un enorme ojo fosforescente rectangular comenzó a desarrollarse en la parte frontal del dispositivo hasta adquirir las dimensiones de un pequeño monitor de alta definición. El teléfono móvil comenzó a incorporar características típicas de las

agendas digitales, incluyendo interfaces de comunicación de infrarrojos y Bluetooth, pantalla táctil, sistemas operativos con aplicaciones de productividad y conexión a Internet. Tras un período de experimentación, este proceso culminó con la salida al mercado del Apple iPhone en 2007 y de los primeros dispositivos Android en 2009. Algo más tarde llegaron los *tablets*, más grandes que el *smartphone* pero provistos de hardware análogo —pantalla táctil—, los mismos sistemas operativos (iOS o Android según el caso) y conectividad GSM, 3G y 4G para las redes telefónicas de datos móviles.

De pocos años a esta parte, los gustos del mercado parecen haberse decantado por Android en perjuicio de otras plataformas hasta entonces populares como Blackberry, Nokia, Windows Mobile e incluso el mismo iPhone de Apple. En el año 2013, Android lideró el mercado español con más de un 80 % de los terminales inteligentes vendidos. En otros países nos encontramos con cifras similares. Durante su trabajo con *smartphones* o tabletas es más probable que el investigador forense se encuentre con dispositivos Android que con cualquier otro tipo, de modo que conviene estar familiarizado con su diseño, tecnología y posibilidades de uso. Para realizar un análisis forense de estos aparatos es preciso saber qué posibilidades ofrecen y de qué modo los utilizan los usuarios, como consumidores normales, delincuentes o víctimas de delitos tecnológicos.

Vemos que una cosa es pequeña y nos hacemos una idea equivocada de su importancia. Pensamos en un *smartphone* como si fuera un teléfono móvil tradicional o una agenda electrónica, con calculadora, libreta de contactos, un teclado virtual para escribir mensajes SMS y poco más. Se nos ocurre que con algo de tan poco peso no es mucho lo que se puede hacer, y suponemos que ahí dentro hay un hardware y un software parecidos al de una calculadora científica programable. Pero las apariencias engañan. En potencia y prestaciones nuestro terminal móvil es comparable a un ordenador de sobremesa de hace cinco años, solo que algo más incómodo de manejar debido al tamaño. Su CPU ejecuta un árbol de procesos tan frondoso como el de un PC de sobremesa con Linux. Para convencernos no tenemos más que ejecutar el comando `ps` en un terminal de texto (Figura 1.2). En la imagen solo se ven las últimas líneas. El listado completo ocuparía cuatro páginas de este libro.

1.2 HARDWARE

Un dispositivo capaz de contener en memoria tantos procesos tiene que ser más complejo que una calculadora programable o una agenda digital. En las páginas siguientes figura una lista de los elementos de hardware que podemos encontrar en el interior de un típico *smartphone* Android. Nuestro ejemplo concreto es un Samsung Galaxy Ace 2, modelo algo anticuado pero todavía popular. Puesto que todos los

terminales Android tienen un diseño y un funcionamiento similares, lo aquí expuesto es extensible a gran parte de los dispositivos del mercado.

```
.....
.....
root      3254  2  0  0  ffffffff  00000000 S migration/1
root      3255  2  0  0  ffffffff  00000000 S ksoftirqd/1
root      3256  2  0  0  ffffffff  00000000 S watchdog/1
root      3257  2  0  0  ffffffff  00000000 S ext4-dio-unwrit
root      3258  2  0  0  ffffffff  00000000 S ext4-dio-unwrit
root      3259  2  0  0  ffffffff  00000000 S ext4-dio-unwrit
root      3260  2  0  0  ffffffff  00000000 S ext4-dio-unwrit
root      3261  2  0  0  ffffffff  00000000 S ext4-dio-unwrit
root      3262  2  0  0  ffffffff  00000000 S v9fs/1
root      3263  2  0  0  ffffffff  00000000 S svnetd/1
root      3264  2  0  0  ffffffff  00000000 S kconservative/1
root      3265  2  0  0  ffffffff  00000000 S kmpathd/1
root      3266  2  0  0  ffffffff  00000000 S ux500_wdog/1
root      3267  2  0  0  ffffffff  00000000 S shm_mod_reset_r
root      3268  2  0  0  ffffffff  00000000 S shm_ca_wake_req
root      3269  2  0  0  ffffffff  00000000 S shm_ac_wake_req
root      3270  2  0  0  ffffffff  00000000 S shm_audio_chann
root      3271  2  0  0  ffffffff  00000000 S B2R2/1
root      3272  2  0  0  ffffffff  00000000 S crypto/1
root      3273  2  0  0  ffffffff  00000000 S aio/1
root      3274  2  0  0  ffffffff  00000000 S kondemand/1
root      3275  2  0  0  ffffffff  00000000 S kblockd/1
root      3276  2  0  0  ffffffff  00000000 S events/1
app_43    3331 1533 171984 31880 ffffffff 00000000 S com.google.android.apps.plus
app_62    3346 1533 146128 23484 ffffffff 00000000 S com.google.android.apps.maps
app_65    3362 1533 138048 16736 ffffffff 00000000 S com.sec.android.app.controlpanel
app_50    3374 1533 155564 26220 ffffffff 00000000 S com.google.process.location
app_50    3382 1533 147080 20216 ffffffff 00000000 S com.google.android.gms
app_50    3388 1533 191180 31528 ffffffff 00000000 S com.google.process.gapps
app_50    3400 1533 154136 18608 ffffffff 00000000 S com.google.android.gsf.login
app_62    3500 1533 160872 27328 ffffffff 00000000 S
com.google.android.apps.maps:GoogleLocationService
root      3749  2  0  0  ffffffff 00000000 S flush-179:0
shell     3892  1562 800   328   c02da9e0 afd0c50c S /system/bin/sh
shell     3893  3892 956   320   00000000 afd0b58c R ps
compass   3894 1    900   316   ffffffff 00000000 D /system/bin/geomagneticd6x
compass   3895 1    872   308   ffffffff 00000000 D /system/bin/orientationd6x
```

Figura 1.2. Tabla de procesos Android

1.2.1 Microprocesador

La CPU (unidad central de proceso) es el corazón del sistema. Ejecuta instrucciones y almacena resultados en memoria. Controla los restantes elementos del ordenador y realiza todas aquellas funciones que no hayan sido delegadas a un chip especializado o a los módulos de hardware auxiliares que vamos a ver en los apartados siguientes. La mayoría de los dispositivos Android utiliza microprocesadores ARM basados en arquitectura RISC (*reduced instruction set computer*; ordenador con conjunto de instrucciones reducido), de uso frecuente en aparatos móviles, aplicaciones integradas y periféricos de ordenador, como por ejemplo impresoras y plotters, debido al favorable equilibrio entre potencia y bajo consumo, resultado de un diseño relativamente simple.

El Samsung Galaxy Ace 2, al igual que otros dispositivos móviles fabricados por la competencia, lleva dentro uno de estos microprocesadores ARM —concretamente Novathor U8500 de doble núcleo a 800 MHz—. Los modelos más recientes de Samsung y otras marcas incluyen una CPU de mayor potencia con velocidad superior a los dos gigahertzios. El procesador gráfico se halla integrado en el mismo chip. En general, la disposición de elementos y el grado de integración entre los diversos componentes son similares a los que podemos hallar en un PC de sobremesa. El *smartphone* también tiene su propia placa base en la que se hallan montados diversos componentes como la CPU, el procesador de comunicaciones de banda ancha, los chips de memoria NAND/RAM, etc. En la imagen se pueden apreciar los principales componentes electrónicos del Samsung Galaxy S3, otro modelo de la serie de teléfonos inteligentes de Samsung (Figura 1.3).

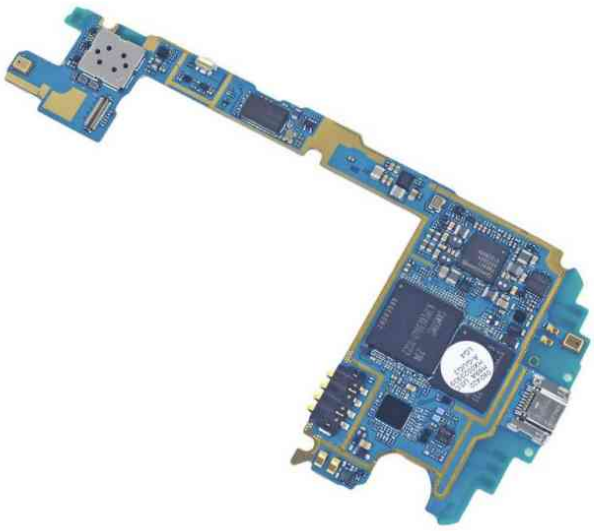

Figura 1.3. Placa base de un Samsung Galaxy S3

Existen proyectos para portar Android a otras arquitecturas, como los microprocesadores Atom de Intel. Algunos ordenadores portátiles provistos de procesadores Intel x86 también funcionan con versiones adaptadas de Android, como por ejemplo los Asus Eee PC o el ThinkPad x61 de Lenovo.

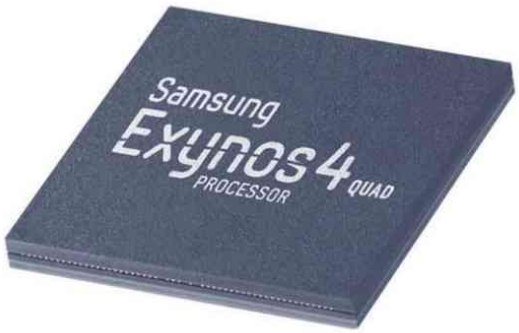

Figura 1.4. CPU Exynos 4 Quad utilizada en el Samsung Galaxy S3

1.2.2 Comunicaciones móviles de banda ancha

En principio la CPU de un dispositivo móvil, además de controlar el funcionamiento del ordenador, podría ocuparse también de gestionar la conexión a redes celulares de voz y datos. Sin embargo, por razones de eficacia y para simplificar el diseño del software, esta función es delegada en un componente específico que incluye la tecnología y los servicios necesarios para mantener una conexión a redes telefónicas y de datos: GSM 50/900/1800/1900. HSDPA 14.4 Mbit/s y HSUPA 5.76 Mbit/s. Aunque se trata de un elemento de hardware específico e independiente, suele ir integrado en el mismo bloque que la CPU.

1.2.3 Conectividad wifi y Bluetooth

Así mismo, los dispositivos móviles disponen de conectividad para otras tecnologías inalámbricas como wifi y Bluetooth, con el objeto de poder acceder a Internet por banda ancha. Para ello disponen de adaptadores inalámbricos basados en el estándar IEEE 802.11 con sus variantes b/n/g, y de una interfaz Bluetooth que hace posible la comunicación con impresoras, teclados, auriculares inalámbricos y otros periféricos. Algunos aparatos Android, entre ellos la mayor parte de los *tablets* y dispositivos multimedia para el hogar, prescinden de la conectividad GSM/G3/G4, incluyendo únicamente el adaptador para comunicaciones inalámbricas wifi/Bluetooth.

En la mayor parte de los *smartphones*, el teléfono móvil puede ser desactivado poniendo el terminal en modo avión o retirando de su interior la tarjeta SIM. En todo lo demás, el dispositivo seguirá funcionando como reproductor multimedia, soporte de almacenamiento de datos, GPS e incluso ordenador de bolsillo con capacidad de acceso wifi a Internet.

1.2.4 Memoria RAM/NAND

Siendo un ordenador, todo dispositivo Android precisa de una memoria de acceso aleatorio para guardar los programas de usuario y los datos procesados por la CPU. Suele producirse cierta confusión debido a la naturaleza de los dos tipos de memoria que utilizan los terminales: por un lado la RAM, construida a base de circuitos SDRAM muy rápidos y de contenido volátil, que se pierde al quedar interrumpido el suministro de corriente —en el caso de los dispositivos móviles, cuando apagamos el terminal o se agota la batería—; y por otro la memoria de almacenamiento o NAND, similar a la de las antiguas tarjetas CompactFlash y los actuales *pendrives* USB, que una vez grabada conserva su contenido aunque no circule la corriente.

No hay que confundir entre la memoria NAND de 2, 4, 8 o 16 GB, ampliable a 32 o 64 GB mediante tarjetas de tipo MicroSD, y la memoria RAM de un PC de sobremesa, que suele ser de 4, 6, 8 o 12 GB en ordenadores de gama media-alta. No quiere decir que un teléfono móvil esté mejor equipado que un PC en este aspecto. Se trata de cosas diferentes. Los datos del fabricante de terminales móviles por lo general hacen referencia a la memoria NAND. Los *smartphones* disponen de una RAM menor, de entre 512 y 2 GB. El Samsung Galaxy Ace 2, por ejemplo, posee una memoria RAM de 768 MB y una memoria NAND de 4 GB para guardar datos, programas y archivos del usuario. Podemos pensar en esta zona de almacenamiento de datos como si fuera un pequeño disco duro en estado sólido —de hecho, se encuentra dividida en particiones y formateada con los mismos sistemas de archivos que solemos encontrar en los discos duros—. Dependiendo de la marca y el modelo de terminal, la memoria NAND es ampliable mediante tarjetas MicroSD. Esta zona de almacenamiento extra funciona como una partición auxiliar de la memoria NAND del dispositivo. Por motivos de compatibilidad casi siempre se encuentra formateada con el sistema de archivos FAT32.

Las tecnologías empleadas en la elaboración de ambos tipos de memoria son diferentes. La RAM precisa de semiconductores muy rápidos con celdas de acceso enteramente aleatorio. En otras palabras, cada posición de memoria es direccionable individualmente. De ahí que su coste de fabricación sea más elevado que el de la memoria NAND. En la práctica, RAM y NAND vienen integradas dentro de un mismo componente, un circuito MCP (*multi chip package*) que incluye el subsistema completo de memoria además de los contactos para el acoplamiento a la placa base del terminal.

Hablaremos con más detalle de la memoria NAND en el capítulo correspondiente a sistemas de archivos para dispositivos Android.

1.2.5 Tarjeta MicroSD

La tarjeta MicroSD (Figura 1.5), utilizada por una gran variedad de aparatos portátiles —*tablets*, reproductores MP3, cámaras digitales, consolas de videojuegos, etc.— es de tipo no volátil y está fabricada también con tecnología NAND. Este elemento posee gran importancia para la investigación forense. La tarjeta MicroSD, insertada bajo la batería del dispositivo, o, más comúnmente, en una ranura lateral, supone una diferencia de diseño esencial con respecto a los terminales Apple iPhone, los cuales están provistos de una NAND interna de 16 GB a 64 GB sin posibilidad de ampliación. Esto permite al fabricante lograr un mayor grado de control del dispositivo móvil y del sistema operativo, a cambio de restringir la libertad del usuario —solo es posible añadir software y archivos nuevos a través del Apple Store y del software de sincronización iTunes— y las posibilidades de ampliación del terminal.

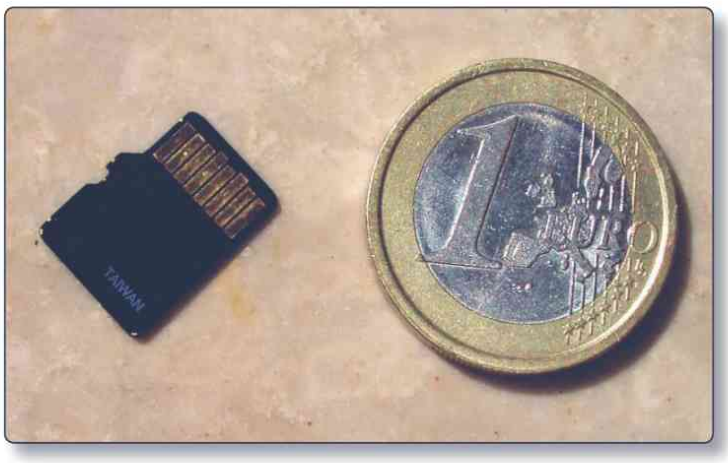

Figura 1.5. Tarjeta MicroSD del mismo tipo que las utilizadas en teléfonos móviles, smartphones, tabletas, reproductores MP3 y otros dispositivos portátiles

En algunos dispositivos Android comienza a imponerse desde hace tiempo una tendencia a la desaparición de las tarjetas MicroSD, reemplazadas por zonas de almacenamiento internas que funcionan del mismo modo que aquellas. En este caso se habla de tarjetas emuladas en memoria NAND. El investigador debe tenerlo en cuenta cuando analiza un terminal, tomando nota de la partición correspondiente, el sistema de archivo —generalmente FAT32—, el punto de montaje y otras características de configuración.

1.2.6 Pantalla

Las pantallas táctiles de alta definición son un campo de desarrollo muy dinámico en la tecnología de los dispositivos móviles. Precisamente esta naturaleza táctil, combinada con las prestaciones de otros componentes del terminal, como por ejemplo el procesador gráfico integrado (GPU), es lo que hace de un *smartphone* una experiencia diferente. Monitor, teclado y ratón al mismo tiempo, la pantalla táctil es algo más que un periférico de entrada y salida. A través de ella el internauta se hace dueño de la parte de red que le corresponde. Toma posesión no solo simbólicamente, sino también en sentido físico, de un modo similar al del hombre primitivo cuando pintaba bisontes en las cavernas para propiciar la caza. Esto condiciona su visión del mundo y su concepto de las relaciones sociales. El usuario toca los datos con los dedos e interacciona con el entorno a través de redes sociales, mapas y servicios de búsqueda avanzados, de tal modo que resulta difícil decir hasta qué punto el mundo en el que vive no comienza ya a ser parte de una realidad aumentada. A lo largo de la historia de los terminales móviles, la calidad de las pantallas, como por ejemplo las del tipo Super AMOLED, diseñadas para los dispositivos móviles Android de Samsung, ha ido en aumento hasta alcanzar en los últimos modelos características de alta definición comparables a la de un monitor de sobremesa.

1.2.7 Teclado

Otra característica relevante de los dispositivos móviles es la tendencia a reemplazar los teclados mecánicos por sistemas de entrada de datos virtuales, lo cual trae consigo un número de ventajas. El usuario ya no se encuentra limitado a un juego de caracteres sino que puede configurar el que más le convenga: inglés, español, griego, ruso o cualquier otro. La virtualización de teclados permite así mismo el empleo de otros tipos de apuntadores y sistemas de entrada de datos, incluyendo software de reconocimiento de escritura manual. También simplifica la fabricación del terminal, haciéndolo más compacto y resistente a roturas, partículas de suciedad e impactos. Finalmente, se puede aprovechar toda la parte frontal del dispositivo para dotarlo de una pantalla más grande.

1.2.8 GPS

El GPS (*global positioning system*) es probablemente la innovación más avanzada que se ha incorporado en los últimos años a los teléfonos móviles y otros dispositivos portátiles. Su principio de funcionamiento es el siguiente: a partir de la señal de radio emitida por cuatro satélites, un receptor calcula la posición geográfica

del terminal y su altura sobre el nivel del mar. Las coordenadas de longitud y latitud, que permiten establecer la posición geográfica actual del usuario con márgenes de error de unos pocos metros, asociándola a una fecha y hora determinadas, proporcionan una valiosa información al investigador forense.

1.2.9 Cámara

De las primitivas cámaras integradas en los teléfonos móviles, que permitían tomar fotos con una definición más o menos satisfactoria, se ha pasado en pocos años a sistemas de vídeo de alta definición y a la posibilidad de utilizar servicios de teleconferencia gracias a la instalación de una segunda cámara en la parte frontal del dispositivo. El valor forense de los archivos de imagen y de vídeo no solo reside en el contenido de los mismos, sino también en los metadatos de los archivos, los cuales proporcionan pistas valiosas para explicar los hechos: fechas, marcas de tiempo, software de retoque utilizado, etc. Analizando los datos de la cámara y del GPS resulta posible determinar la posición geográfica del dispositivo en el momento de ser tomadas y registradas las imágenes. También podemos reconstruir itinerarios y recorridos geográficos del sospechoso comparando diversas secuencias de datos. Añadiendo el historial de conexiones a puntos de acceso wifi y antenas GSM de las compañías telefónicas, información que convenientemente queda registrada en archivos XML y bases de datos SQLite, es posible obtener un material probatorio que resultará muy útil para los fines de la investigación. De todo esto se tratará con más detalle en un capítulo posterior.

La cámara de un *smartphone* también puede ser utilizada para interactuar con aplicaciones de realidad aumentada. Este es un campo de reciente desarrollo con perspectivas prometedoras para todo tipo de actividades económicas y recreativas. Por ejemplo, el usuario que está de visita en una ciudad desconocida enfoca un monumento y al instante aparece en la pantalla un texto descriptivo superpuesto a la imagen. Esto es posible porque un software de reconocimiento de imágenes y el GPS interactúan a través de Internet con una base de datos que suministra información relativa a localizaciones interesantes, como cafeterías, comercios, instalaciones públicas y otros objetos. Las posibilidades de esta tecnología son infinitas: formación, medicina, logística, reparación de vehículos, mantenimiento de equipos industriales y muchos otros usos. A lo anterior habría que añadir la capacidad de las dos cámaras del *smartphone* para suministrar material gráfico destinado a redes sociales, así como funciones de reconocimiento facial, de sonido, imágenes y objetos de todo tipo; y, finalmente, la posibilidad de leer códigos de barras y otros patrones de identificación. Todo ello se traduce en una enorme masa de datos a disposición del investigador.

1.2.10 Micrófono

Aunque no lo parezca, este componente del teléfono móvil también ha experimentado adelantos dignos de mención. Algunos dispositivos llevan no uno sino dos e incluso tres micrófonos, que controlados por software mejoran el sonido cancelando ruidos de fondo e incrementando la calidad de la percepción auditiva. Posiblemente, algún día la combinación de micrófonos de alta calidad y sistemas de reconocimiento de voz permita reescribir la historia de las interfaces hombre-máquina, proporcionando al usuario terminales manejables a través de la voz. Un ejemplo de lo que nos depara el futuro lo podemos ver ya por ejemplo en dispositivos como Google Glass.

1.2.11 Giróscopo y acelerómetro

Acelerómetros y giróscopos hacen posible que el *smartphone* cambie la posición de la imagen cada vez que el usuario gira el terminal para adaptar la pantalla a su perspectiva visual. El principio que hace posibles estos dispositivos no es precisamente novedoso, ya que se conoce desde los comienzos de la Revolución Industrial. James Watt utilizaba ya un rudimentario sistema inercial, inventado por él, para regular el funcionamiento de sus primeras máquinas de vapor. El giróscopo fue descubierto por Foucault a mediados del siglo XIX. Perfeccionado a continuación por un número de relevantes pioneros de la técnica, entre los cuales cabe mencionar al inventor español Leonardo Torres Quevedo, desde entonces ha sido utilizado en buques, torpedos, aviones, naves espaciales y otros vehículos. Los avances de la miniaturización han hecho posible incluir aparatos inerciales de este tipo en el interior de teléfonos móviles y dispositivos informáticos portátiles.

El acelerómetro y el giróscopo proporcionan soporte a gran variedad de aplicaciones y juegos de acción. Mediante software especial, el *smartphone* también puede emular diversos aparatos de registro para la investigación científica: sismógrafos, medidores de pulsaciones cardíacas o sensores de vibraciones para supervisar el funcionamiento de motores y máquinas eléctricas.

1.2.12 Batería

La batería cumple una función evidente y crucial por sus implicaciones para el trabajo forense: suministrar energía al dispositivo. El desarrollo de fuentes de alimentación fiables y de calidad constituye el apartado de mayor importancia estratégica en el mercado actual de dispositivos portátiles. En algunos países de Asia, los falsificadores de producto consiguen crear réplicas perfectas de aparatos

sofisticados, como calculadoras programables Texas Instrument y *smartphones* iPhone 5 o Samsung Galaxy S4. Para ello no tienen más que montar sobre una placa los mismos chips que los fabricantes de Taiwán venden a los fabricantes del equipo original. El resto es bricolaje. Sin embargo, no es tan fácil crear un suministro de energía que dure tanto como el de un dispositivo de marca original y no empiece a dar problemas al cabo de poco tiempo.

En su principio básico de funcionamiento pilas y acumuladores son algo tan simple que hasta un alumno de primaria los puede fabricar en el laboratorio de la escuela. Pero la investigación en tecnología de baterías eficientes y de alta calidad, capaces de alimentar componentes electrónicos durante días enteros, sin desprendimiento de calor ni pérdidas de capacidad por sulfatación, es algo que no está al alcance de ningún imitador por hábil que sea. El trabajo de inyección de plástico y montaje de chips sobre plaquitas se puede reproducir, pero las soluciones de química avanzada no están al alcance de talleres amateur. Los fabricantes han tenido que dedicar un gran esfuerzo industrial y científico, con décadas de estudio en ciencia de materiales e inversiones multimillonarias, para hacer posible el desarrollo de pequeñas pilas de petaca capaces de almacenar hasta 2.600 mW/h en un objeto del tamaño de un chicle, sin efecto memoria, con tiempos de carga cortos y una duración tan larga como el ciclo de vida del producto.

La fuente de energía de un *smartphone* es importante para el investigador forense por una razón que a veces no resulta evidente, pero que no conviene perder de vista: si se agota o falla por cualquier motivo, se pierden todos los datos existentes en la memoria RAM. En tiempos recientes han empezado a comercializarse baterías suplementarias que se conectan al dispositivo móvil a través del cable USB y permiten prolongar su funcionamiento durante una buena cantidad de horas. Estas baterías están pensadas para los usuarios adictos a redes sociales y servicios de mensajería, que consumen gran cantidad de energía, pero también pueden hacerle un buen servicio al investigador.

En algunos *smartphones*, la tarjeta MicroSD suele estar colocada detrás de la batería, y para llegar a ella es necesario quitarla, con lo cual se deja sin alimentación al terminal. Por tanto es preciso planificar minuciosamente la secuencia de operaciones cuando se tiene previsto adquirir un dispositivo móvil mediante técnicas forenses.

1.2.13 Interfaz USB

Un avance digno de mención en Android es que los terminales que funcionan con este sistema operativo prescinden en su mayoría de cables propietarios en favor de un estándar universal. Esto hace más fácil la conexión con ordenadores de sobremesa y dispositivos de adquisición forense por hardware. El conector

USB de un *smartphone* es una pasarela que permite llevar a cabo las funciones de conectividad que se detallan a continuación:

- Recarga de la batería a través de una fuente de alimentación enchufada a la red o a la interfaz USB de un PC portátil o de sobremesa. En la mayor parte de los casos basta conectar el terminal a la estación de trabajo,[3] y que esta a su vez esté enchufada a la corriente. La mayor parte de los cargadores que se enchufan a la red también disponen de una interfaz USB para conectar el cable de alimentación.

- Actualización del *firmware*, funcionamiento del software de sincronización y otros cometidos previstos por el fabricante.

- Depuración USB a través del puente ADB: se trata de una interfaz característica de los dispositivos Android, que permite acceder al terminal mediante un *shell* o intérprete de línea de comando para fines de desarrollo e instalación de aplicaciones. El investigador se sirve de ADB para examinar el dispositivo y extraer elementos de evidencia. Más adelante trataremos sobre ello.

- Conexión de soportes de datos y acceso a las particiones del dispositivo (memoria NAND, tarjeta simulada, expansión MicroSD) con posibilidad de copiar archivos directamente del ordenador al terminal móvil, y viceversa, como si se tratara de un disco duro externo.

Figura 1.6. Icono universal para conexión USB

3 A partir de este momento se denomina estación de trabajo al ordenador de sobremesa o portátil en el que el investigador tiene instaladas sus herramientas forenses y en el que guarda los datos de las adquisiciones y realiza la mayor parte de las tareas de análisis.

1.2.14 Componentes de otros tipos

Los elementos anteriores no son los únicos que se pueden encontrar en el interior de un dispositivo móvil Android. Consecuente con su naturaleza abierta y colaborativa, como resultado de su origen en el movimiento del software libre y en la creatividad empresarial de Google, a la hora de incluir nuevos accesorios de hardware en los terminales Android no hay más límites que la imaginación del desarrollador. La gama de aplicaciones y proyectos es inabarcable: tabletas, ordenadores miniportátiles, reproductores de medios audiovisuales, dispositivos Google TV, aparatos GPS, juguetes, consolas de videojuegos, lectores de libros electrónicos, cámaras digitales, robots, kits para el desarrollo de aplicaciones domóticas, fotocopiadoras e incluso electrodomésticos.

Cada vez que decide crear nuevos elementos de hardware o incluir funcionalidades adicionales en uno ya existente, el ingeniero cuenta con lo necesario para desarrollar interfaces de conexión, controladores y software de usuario. Los componentes enumerados no responden a definiciones canónicas de Android y reflejan tan solo las preferencias del mercado. En el futuro puede haber cambios debido a la invención de nuevas prestaciones o la incorporación de funcionalidades adicionales.

1.3 SOFTWARE

El mejor hardware no sirve de mucho si no se dispone de un software que lo haga funcionar: las calculadoras programables precisan rutinas perfectamente diseñadas y grabadas de modo indeleble en ROM; las máquinas-herramienta necesitan sus PLC y módulos de control numérico, y un PC no puede hacer nada sin la ayuda de Linux o Windows. Android es un sistema operativo completo, surgido como resultado de un fenómeno evolutivo que es preciso estudiar en sus orígenes para hacerse una idea de su potencial. Al contrario que los sistemas operativos tradicionales como VAX, OS2 o Windows NT, no fue creado por el equipo de ingenieros de una gran empresa en forma de producto listo para embalar y distribuir en cajas de cartón, junto a un manual de instrucciones bien encuadernado como parte integrante del volumen de suministro de un equipo informático. En Android confluyen diversas tecnologías e intereses: telefonía móvil, software libre, virtualización, Internet y, sobre todo, la intervención crucial de Google como patrocinador y estratega de una cruzada que en pocos años ha logrado conquistar la atención y el bolsillo del consumidor de informática móvil.

1.3.1 Orígenes de Android

Cuando el teléfono móvil dejó de ser un artículo exclusivo para convertirse en producto de masas, las empresas de telecomunicaciones se vieron obligadas a invertir ingentes recursos para fomentar la movilidad y abrir nuevos mercados. Este esfuerzo atrajo a emprendedores que hasta entonces habían estado trabajando para compañías telefónicas y empresas tecnológicas. Este fue el caso de Andy Rubin, antiguo empleado de Apple, que en 2003 fundó una *startup* llamada Android Inc. dedicada al desarrollo de software para dispositivos móviles. Después de largas e infructuosas negociaciones con fondos de capital riesgo, Rubin se puso en contacto con Google, quien decidió adquirir Android a mediados de 2005. Este movimiento de Google, complementado con la compra de patentes de tecnologías móviles y la participación en subastas de frecuencias del espectro, hizo sospechar que Google estaba desarrollando un modelo propio de *smartphone* similar al Apple iPhone con el propósito de entrar en el mercado de la telefonía móvil.

Esto era verdad pero solo en parte. Google creó, efectivamente, su propia gama de móviles inteligentes —la serie Google Nexus—, pero no como artículo exclusivo sino como modelo de concepto para establecer una directriz orientativa en el proyecto Android, de tipo general y desprovista de elementos propietarios de desarrolladores específicos. Google no fabrica el dispositivo, sino que lo licencia a diversas marcas que estén interesadas en producirlo de acuerdo con las especificaciones del proyecto Android. Hasta la fecha, el modelo Nexus ha sido fabricado por HTC, Samsung y LG. El terminal se vende sin características privativas del fabricante ni los operadores telefónicos, y con el gestor de arranque desbloqueado, para permitir que el usuario final lo adapte a sus necesidades. El propósito de crear terminales con estas características transmite el deseo de disponer de una plataforma genérica para ensayar perfeccionamientos y nuevas versiones del software. De hecho, los dispositivos Nexus son los primeros que reciben las actualizaciones del sistema cuando estas se encuentran listas para ser liberadas.

Figura 1.7. Dispositivos Nexus de diversos fabricantes (Fuente: Creative Commons)

El objetivo de Google, más que competir en un mercado de productos emblemáticos como el iPhone, el Blackberry o los nuevos terminales basados en Microsoft Windows Mobile, consistía desde el principio en crear una plataforma completa y abierta para dispositivos móviles, que incluya un sistema operativo, interfaz de usuario y suficientes aplicaciones. El software debía ejecutarse en una plataforma móvil del tamaño de una agenda electrónica similar a los antiguos Palm Pilot o HP iPaq y hallarse libre de los impedimentos propietarios que hasta la fecha entorpecían el desarrollo de innovaciones en el campo de la informática móvil.

Una vez completados los trabajos de desarrollo inicial, Google puso a disposición de la comunidad el primer kit de desarrollo de software (SDK) para Android. En agosto de 2008 se anunció el establecimiento del Android Market, una especie de repositorio central desde donde el usuario podía descargar sus aplicaciones. A finales del mismo año fue liberado el primer Android Open Source Project (AOSP). El código fuente de Android, como es norma en el movimiento del software libre, quedaba de este modo al alcance de cualquiera que estuviese interesado en modificarlo y adaptarlo a sus propias necesidades. Por aquel entonces salía al mercado el primer *smartphone* Android, el G1 de T-Mobile.

A partir de ese momento el ecosistema Android explotó, desencadenando la extinción masiva del teléfono móvil tradicional, que de la noche a la mañana perdió atractivo al lado de los novedosos y versátiles terminales móviles provistos de pantalla táctil y conectados permanentemente a Internet. De la noche a la mañana se desplomaron las cuotas de mercado de otros dispositivos propietarios hasta hace poco emblemáticos como el iPhone o el Blackberry. En la actualidad los dispositivos Android son los más populares del mercado. Este hecho habla no solo a favor de las ventajas del modelo de negocio establecido por Google. También testimonia el vigor de una estrategia de mercados basada en plataformas abiertas, principio hasta hace pocos años considerado inviable por la mayor parte de los analistas, tras el fracaso de las expectativas comerciales generadas en su momento por el código abierto y el fenómeno Linux.

1.3.2 Open Handset Alliance

La Open Handset Alliance (OHA) es una asociación de empresas de telecomunicaciones formada por proveedores de acceso a Internet, fabricantes de equipos, desarrolladores de software y otras compañías de servicios tecnológicos, logísticos y comerciales. Siendo Google el principal impulsor de la OHA, no extraña que los objetivos de la Alianza no solamente coincidan, sino que para formularlos se emplee una retórica similar: "acelerar la innovación y ofrecer al consumidor de telefonía móvil una experiencia mejor, más variada y asequible".

Se suele decir que la Alianza no es más que una plataforma de marketing que no crea valor para los miembros ni para el usuario. En cualquier caso constituye uno de los pilares de Android y, como herramienta corporativa de relaciones públicas, esto le otorga una respetabilidad que difícilmente habría logrado adquirir actuando desde la escena amateur y cooperativista del software libre, aun cuando para ello hubiese empleado los recursos publicitarios del todopoderoso Google.

Fundada en 2005, la Open Handset Alliance cuenta en la actualidad con más de 80 miembros.

1.3.3 Intereses de Google

¿Qué gana Google con todo esto? La creación de una plataforma móvil de código libre, junto con el hardware, la infraestructura y el desarrollo de mercados para nuevos productos y servicios es un proyecto muy costoso. ¿Se hace por amor al arte o también hay aquí algo de sano egoísmo empresarial? La respuesta tiene que ver con el negocio principal y la visión de Google: organizar la información del mundo haciéndola universalmente accesible. Actualmente hay en el mundo 7.000 millones de teléfonos móviles y una cobertura cercana al 100 % de la población, incluyendo a los países en vías de desarrollo. El mejor camino para llegar a los mercados globales ya no es el PC de sobremesa: es el teléfono móvil.

El modelo de negocio de Google está basado en la publicidad que acompaña a sus servicios de búsqueda de información. Aunque parezca paradójico, existe una estrecha relación entre el software libre y los modelos de negocio basados en publicidad. Las plataformas abiertas son una estrategia de distribución eficaz, puesto que ayudan al usuario a franquear barreras que tradicionalmente le impedían beneficiarse de las tecnologías de acceso a Internet.

Contrariamente a lo que se pensaba en el momento de la adquisición de Android, Google no tiene ambiciones en el campo de la telefonía móvil ni se propone desarrollar dispositivos de culto exclusivos como el Blackberry o el iPhone. Por encima de todas las visiones de consultoría y los mitos de Internet, el negocio principal de una empresa constituye su mejor garantía de continuidad, antes, hoy y en el futuro. He aquí el Santo Grial de la cruzada Android: apoyando la creación de un sistema operativo para móviles que satisfaga los requerimientos del fabricante y las necesidades del usuario, Google crea un entorno favorable para el desarrollo de su modelo de negocio basado en servicios de información con publicidad individualizada.

Esta estrategia no solo es aceptada por el consumidor, sino secundada con entusiasmo por las ventajas que comporta para los usuarios: un sistema operativo de código libre, posibilidades reales de elección dentro de una abundante gama de aplicaciones de pago y gratuitas, hardware asequible, potente y de calidad y, por si lo anterior fuese poco, conectividad ilimitada a Internet.

1.3.4 Android Open Source Project

Android es código libre. Los desarrolladores pueden examinarlo e introducir modificaciones para mejorar las características del dispositivo, sus prestaciones funcionales, ampliar el número de sistemas de archivo soportados y hacer posibles otros requisitos. El Android Open Source Project (AOSP) o Proyecto Android de Código Libre no consiste en un conjunto de archivos que implementan Android de modo orientativo o abstracto, antes bien se encuentra individualizado para diversos tipos de terminal (por ejemplo, de la serie Nexus). Los fabricantes introducen los cambios necesarios sobre una versión determinada del sistema mientras el equipo de desarrollo de AOSP trabaja en la próxima publicación (*release*).

El proyecto AOSP se rige por dos condicionados maestros del código libre: la Licencia General Pública GNU, versión 2 (GPLv2), y la Licencia de Software Apache 2.0 (ASL 2.0). GPL es más restrictiva, puesto que obliga a los desarrolladores a poner el código modificado a disposición de la comunidad en los mismos términos en que lo habían recibido antes de introducir los cambios. Google temía que este requisito pudiera inhibir la iniciativa de los productores comerciales, poco amigos de exhibir su *know-how* mediante CD de acompañamiento o páginas web con el código fuente de sus desarrollos de software. Por este motivo se decidió por la licencia Apache 2.0, que permite a las empresas sacar sus productos al mercado sin revelar secretos industriales vinculados con el proceso de elaboración de aquellos. De este modo se consigue un equilibrio entre la libertad del usuario y los intereses legítimos de los fabricantes.

AOSP es un proyecto complejo, que se coordina por Internet y distribuye el trabajo de acuerdo con objetivos específicos. La colaboración es voluntaria, a través de foros especializados o asumiendo voluntariamente una actividad que corresponda a la capacitación o los conocimientos técnicos del solicitante.

Dentro del AOSP existen diversos roles funcionales, como por ejemplo:

- **Líderes de proyecto**: por lo general pertenecientes al personal de Google y encargados de la gestión general del proyecto AOSP.
- **Desarrolladores/contribuidores**: todos aquellos que puedan aportar código nuevo al proyecto.
- **Comprobadores de código**: su misión consiste en examinar y verificar el código existente en busca de fallos y posibles mejoras.
- **Supervisores**: programadores con experiencia en desarrollo de proyectos de software con autoridad para decidir si se autorizan o introducen cambios.

El código fuente AOSP puede ser examinado, modificado, compilado y adaptado para usos propios por todo aquel que lo desee, y se encuentra libre para descarga en la página web del proyecto (*http://source.android.com*). Allí se puede encontrar también toda la información disponible sobre licencias, recursos, posibilidades de contribuir y otros temas relacionados.

1.3.5 Versiones de Android

La primera versión de Android (1.0, sin denominación publicitaria de proyecto) fue liberada a finales de 2008 para utilizar exclusivamente en *smartphones*. Incluía soporte para productos típicos de Google como Gmail, Youtube y Google Maps, además de telefonía y mensajes SMS. Seis meses más tarde Google hizo pública una primera revisión, denominada Cupcake. La nueva versión incorporaba mejoras de diseño y prestaciones adicionales, como grabación de vídeo y aplicaciones auxiliares de pantalla (*widgets*), como relojes, indicadores meteorológicos, *tickers* de bolsa, etc. En la versión 1.6 (Donut) se incluyó soporte para mayores resoluciones de pantalla y navegación gestual. Android 2.0 (Eclair), tras haber corregido algunos defectos de seguridad de las versiones anteriores, añadió soporte para Microsoft Exchange y Bluetooth 2.1. Al cabo de pocos meses, a mediados de 2010, era liberado Android 2.2 (Froyo) con nuevos avances: *tethering* o capacidad para habilitar el terminal como punto de acceso inalámbrico portátil, soporte para memoria RAM por encima de 256 MB, borrado remoto del terminal y posibilidad de instalar aplicaciones en la tarjeta SD.

Android 2.3 (Gingerbread) fue la primera versión de uso generalizado, y permitió incorporar el sistema operativo de Google a los terminales de mayor éxito del mercado. Gingerbread incluía mejoras que optimizaban la ejecución de código. Así mismo, substituyó el sistema de archivos originario de Android YAFFS2 por ext4, procedente del mundo Linux. Casi al mismo tiempo fue liberada la versión 3.0 (Honeycomb), primer Android creado específicamente para tabletas, que sin embargo no habría de alcanzar una presencia significativa en el mercado. Con la versión 4.0 de Android (Ice Cream Sandwich), Google unifica los sistemas operativos de *smartphones* y tabletas. Android 4.0, además de encriptación y bloqueos de pantalla, incorporaba una interfaz de usuario nueva, mejoraba la compatibilidad con diversos tipos de hardware y corregía fallos de seguridad de versiones anteriores.

La moderna y más utilizada versión 4.1 (Jelly Bean) añade encriptación de aplicaciones, *multicast* DNS y actualización inteligente de *apps*, que evita la reinstalación completa de paquetes de software, reemplazando únicamente aquellos archivos que hayan sido modificados. Jelly Bean se ha visto sometida a dos procesos de revisión (4.2 Gummi Bear y 4.3 Gominola) llevados a cabo con el objeto de mejorar el rendimiento de los dispositivos en un mercado cada vez más dinámico y competitivo por la irrupción de nuevas tecnologías (virtualización, *cloud computing*,

realidad aumentada) y la necesidad de mejorar el rendimiento energético de los terminales. Por supuesto, las consideraciones de seguridad también han tenido bastante que ver en el desarrollo de esta versión.

Finalmente, Android 4.4 (KitKat), liberada en noviembre de 2013, es, a la fecha de terminar este libro, la edición más reciente del sistema operativo de Google. Sin suponer adelantos significativos con respecto a las anteriores, incluye mejoras en rendimiento y estabilidad de las aplicaciones y prosigue en la labor de corrección de fallos de funcionamiento y seguridad.

1.3.6 API

Para fines de desarrollo de aplicaciones lo que interesa conocer no es la versión del sistema operativo sino el nivel de API (interfaz de programación de aplicaciones), un nivel funcional compuesto por una serie de procedimientos y recursos. Las API están implementadas en librerías con las que el software interactúa a la hora de solicitar acciones o servicios al sistema. El nivel de API es orientativo a la hora de localizar una determinada liberación (*release*) del *software development kit* (SDK) o conjunto de herramientas de desarrollo con el que los programadores escriben aplicaciones para Android. En la tabla 1.1 el lector podrá encontrar un resumen de las diferentes versiones de Android con sus fechas de liberación y números de API correspondientes.

Versión	Denominación	Nivel API	Fecha *release*	Destino
1.0	-		09/2008	Smartphone
1.5	Cupcake	3	04/2009	Smartphone
1.6	Donut	4	09/2009	Smartphone
2.0	Eclair	7	10/2009	Smartphone
2.1	Eclair	7	01/2010	Smartphone
2.2	Froyo	8	05/2010	Smartphone
2.3	Gingerbread	9 y 10	12/2010	Smartphone
3.0	Honeycomb	12 y 13	02/2011	Tablet
4.0	Ice Cream Sandwich	14 y 15	10/2011	*Smartphone/Tablet*
4.1	Jelly Bean	16	06/2012	*Smartphone/Tablet*
4.2	Jelly Bean	17	11/2012	*Smartphone/Tablet*
4.3	Jelly Bean	18	07/2013	*Smartphone/Tablet*
4.4	KitKat	19	09/2013	*Smartphone/Tablet*

Tabla 1.1. Versiones y niveles de API en Android

1.3.7 Android Market

Android Market, en la actualidad llamado Google Play, es la central oficial de distribución de aplicaciones de Google. En cuanto a estas últimas, las hay de pago y gratuitas. El blog de desarrolladores de Google define el Market como un "sistema de contenidos abiertos diseñado para ayudar al usuario a encontrar, adquirir, descargar e instalar elementos de los más diversos tipos en dispositivos Android". El planteamiento es análogo al del Apple Store, con algunas diferencias. En Android Market, por ejemplo, no existen procedimientos de autorización de aplicaciones. Cualquiera puede darse de alta y ofrecer su software gratuitamente o en modalidad de pago. En este último caso Google se queda con un 30 por ciento de los ingresos. Al comprador se le concede un período de 15 minutos para desinstalar la aplicación y recuperar su dinero en caso de no estar satisfecho con la compra. Para poder identificarlas de modo único en el Market, las aplicaciones deben estar firmadas con una clave privada del desarrollador.

Se puede pensar que este sistema es anárquico y poco fiable. Google opina que las preferencias del usuario, expresadas por medio de votos y comentarios, son un buen criterio para separar el trigo de la paja. Las firmas digitales permiten la trazabilidad de cualquier elemento que pudiera dar origen a problemas de seguridad o conflictos de propiedad intelectual. Google tiene poder para eliminar aplicaciones no solo del mercado oficial Android, sino también de manera remota en el terminal del usuario. Fiel a su planteamiento de contenidos abiertos y respeto a la libertad del usuario, Google no impide la instalación de aplicaciones desde otros mercados, y tampoco localmente a través del cable USB.

En Android no es necesario llevar a cabo prácticas de liberación similares al *jailbreak* de los terminales Apple iPhone. Si esto supone una desventaja o no, es algo sobre lo que no podemos juzgar sin haber estudiado el concepto de seguridad Android, lo cual se hace en un capítulo posterior de la presente obra. También Apple iOS es vulnerable a intrusiones y ataques de *malware*, pese al mopolio de descarga de aplicaciones y la estricta normativa de acceso del Apple Store. Se trata de filosofías de desarrollo distintas, con sus ventajas e inconvenientes.

La instalación de aplicaciones en Android es un proceso simple. Lo único que el usuario necesita es una cuenta en Gmail. Ejecutando desde su terminal la aplicación Playstore, incluida por defecto en todas las versiones de Android, podrá descargar e instalar la aplicación que desee en cuestión de sgundos. Para las aplicaciones gratuitas no se activa la pasarela de pagos. De hecho ni siquiera es preciso facilitar el número de una tarjeta de crédito. Android Market dispone de un sistema de búsqueda que permite al usuario encontrar la aplicación que necesita, orientándose a través de textos explicativos y opiniones de otros usuarios. Durante la instalación, que se lleva a cabo de manera automatizada, un menú muestra los

permisos de acceso a los recursos que la aplicación solicita —información personal, mensajes, teléfono, Internet— y pide al usuario que los confirme. Una vez hecho esto el software está listo para funcionar.

La concesión de permisos constituye una fase crítica a la hora de instalar aplicaciones. La mayor parte de los usuarios aceptan mecánicamente todas las solicitudes sin leerlas. No es una buena costumbre. Los permisos determinan lo que el software puede hacer dentro del contexto de ejecución asignado por el sistema de seguridad de Android. Los creadores de *malware* intentan persuadir al usuario, mediante estratagemas de ingeniería social, para que permita el acceso a funciones críticas del dispositivo. Conviene poner atención en lo que se hace a la hora de instalar programas y comprobar que lo que nos pide está en concordancia con las prestaciones del software. Si una aplicación para redes sociales como WhatsApp o Twitter solicita permiso para acceder a contactos, Internet o geolocalización no hay por qué alarmarse, pues forma parte de sus cometidos habituales, y sin estos privilegios el software no funcionaría. Pero si quien pide autorización para acceder al mecanismo de llamadas telefónicas o envío de mensajes SMS es un programa de pósits o un salvapantallas, entonces el usuario hará bien en rehusar.

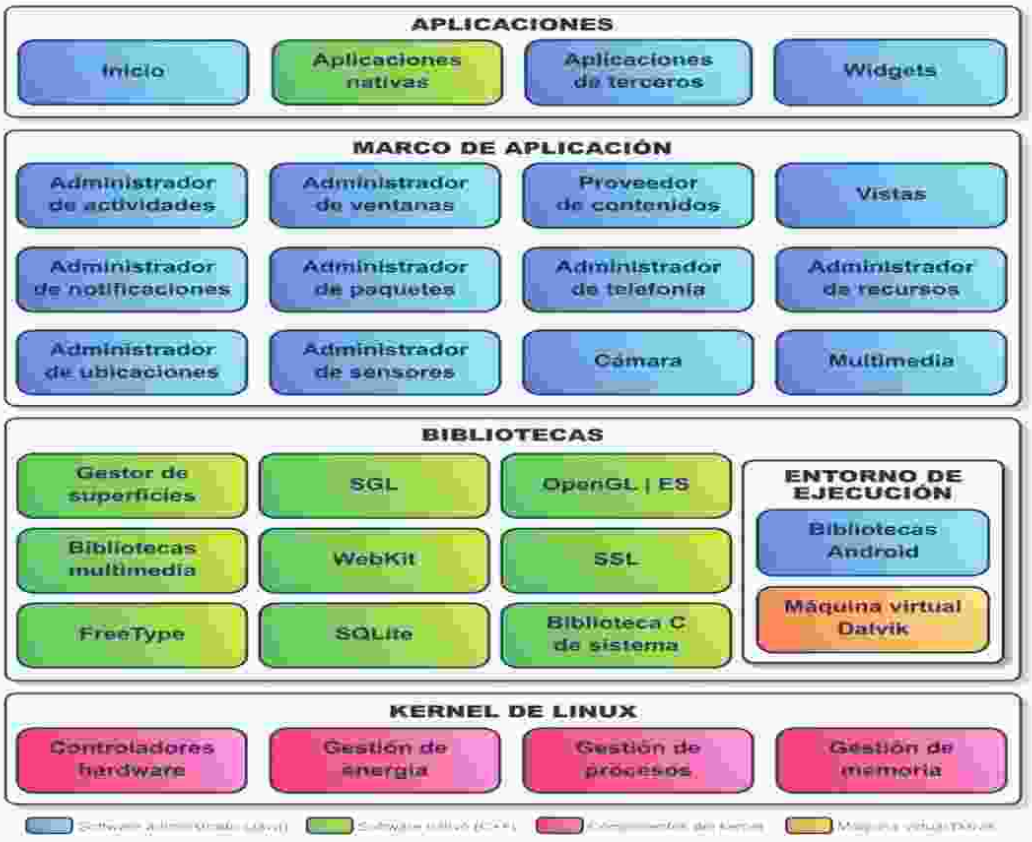

Figura 1.8. Arquitectura Android (Fuente: La Columna 80 – Blog de Ángel J. Vico)

1.4 ARQUITECTURA DEL SISTEMA

Android es un sistema operativo para plataformas móviles. Aunque el investigador no tenga previsto dedicarse a la programación y al desarrollo de aplicaciones, conviene que sepa cuáles son los elementos básicos de la estructura del sistema operativo para entender los procesos que tienen lugar en el interior del dispositivo. Este conocimiento no es tan detallado como el que precisa un programador, y alcanza hasta lo imprescindible para planificar adquisiciones forenses y analizar elementos de evidencia. La arquitectura del sistema operativo está organizada en un modelo de capas (Figura 1.8), con funciones concretas que desde el nivel inferior prestan servicios a las capas superiores hasta terminar en la interfaz gráfica de usuario.

1.4.1 Kernel

En la capa más baja y en contacto directo con el hardware hay una versión adaptada del *kernel* Linux 2.6.x, que, al igual que el resto de los componentes del proyecto AOSP, los desarrolladores pueden modificar y compilar para atender a requerimientos específicos. El *kernel* gestiona procesos, memoria y mecanismos de seguridad del sistema de archivos Linux. A través de diversos controladores —pantalla, teclado, cámara de vídeo, adaptador wifi, memoria *flash*, audio, Binder IPC y administrador de energía—, el *kernel* pone todas las funcionalidades del hardware al servicio del sistema operativo.

En la página web del AOSP el desarrollador hallará información detallada sobre las características del *kernel* Linux y las diferentes posibilidades de compilación para dispositivos móviles y la mayor parte de las plataformas existentes en el mercado de los terminales móviles.

1.4.2 Binder IPC

Dentro de la capa del *kernel* posee especial importancia un *driver* denominado Binder IPC, cuyo cometido consiste en administrar el intercambio de información entre procesos. En Android las aplicaciones no se ejecutan dentro del mismo espacio de memoria y recursos, sino que cada una de ellas tiene asignada su propia zona (*sandboxing*), para no interferir con otros programas y también por razones de seguridad. La comunicación se establece a través de mensajes IPC (*inter process communication*). Binder es el controlador que gestiona este intercambio de señales. Originalmente fue creado como proyecto de código abierto con la denominación OpenBinder. Android lo ha reescrito con el objeto de optimizarlo para sistemas

móviles y evitar las restricciones de propiedad intelectual impuestas por la Licencia General Pública (OPL).

Binder está basado en una arquitectura servidor-cliente: el cliente inicia la comunicación y espera una respuesta del servidor. Este último administra un *pool* de hilos de ejecución o *threads* y responde a las peticiones del cliente. Cada petición, procedente de un proceso determinado, es empaquetada por un *proxy* que proporciona el formato adecuado para el intercambio de datos entre cliente y servidor. Una vez el cliente recibe la respuesta desde el *pool* de hilos de ejecución o *threads* gestionados por el servidor, el *proxy* la desempaqueta (*unwrap*) y hace llegar los datos al proceso que había iniciado la comunicación (Figura 1.9).

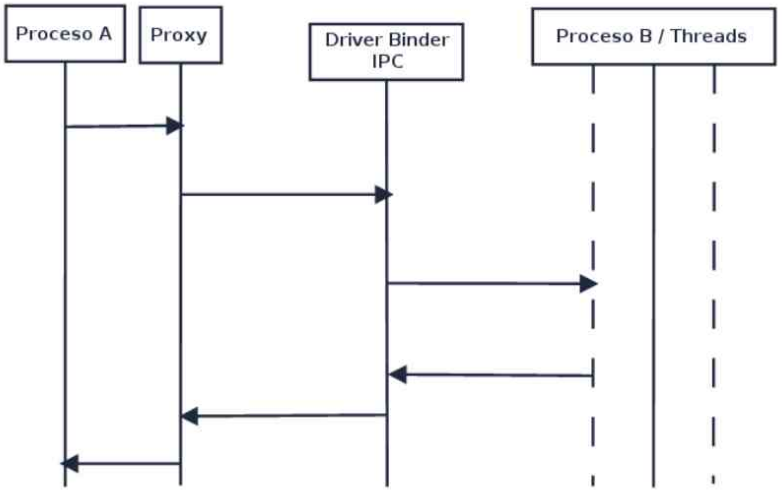

Figura 1.9. Binder IPC

1.4.3 Librerías

En el nivel inmediatamente superior se halla un grupo de librerías con servicios imprescindibles para el funcionamiento del software de usuario: bases de datos, entornos gráficos, de letra y tablas de código y motor de renderizado (WebKit) para el navegador de Internet. Las librerías son archivos que contienen rutinas de código reutilizables para la ejecución de funciones específicas, evitando al programador el tener que incluir el mismo código cada vez que escribe un programa. De este modo se evita la redundancia del software a la vez que se ahorra memoria RAM y espacio de almacenamiento en los soportes de datos. Estas librerías están programadas en C/C++ y son utilizadas tanto por el sistema operativo como por las aplicaciones.

1.4.4 Android Runtime y Dalvik

El *runtime* o entorno de ejecución Android, situado en la segunda capa funcional de la arquitectura del sistema, aloja las librerías básicas de Java y la máquina virtual Dalvik (DVM). Las aplicaciones Android están programadas en Java y, al igual que en este lenguaje, lo que se ejecuta no es código compilado para el hardware de la máquina, sino una especie de código intermedio en un formato especial, válido para la misma máquina virtual implementada en distintas plataformas hardware y, consiguientemente, portable.

Existen algunas diferencias importantes con respecto a Java. En el modelo de desarrollo Android, los archivos que componen el código ejecutable Java son traducidos a un código especial denominado DEX (Dalvik Execution Code). Las librerías y la máquina virtual Dalvik están diseñadas para consumo reducido de recursos (memoria RAM, ciclos de procesador, energía), lo cual resulta de vital importancia en dispositivos que han de ser alimentados mediante baterías. Inicialmente, el desarrollador escribe su aplicación en Java, generando una o varias clases que son compiladas mediante el procedimiento tradicional. Posteriormente, el código para Dalvik se obtiene a través de una recompilación de estas clases mediante la herramienta de conversión `dx`. El resultado del proceso es un archivo *.dex* que contiene todas las clases pertenecientes al programa y es agregado al paquete APK de la aplicación Android (Figura 1.10).

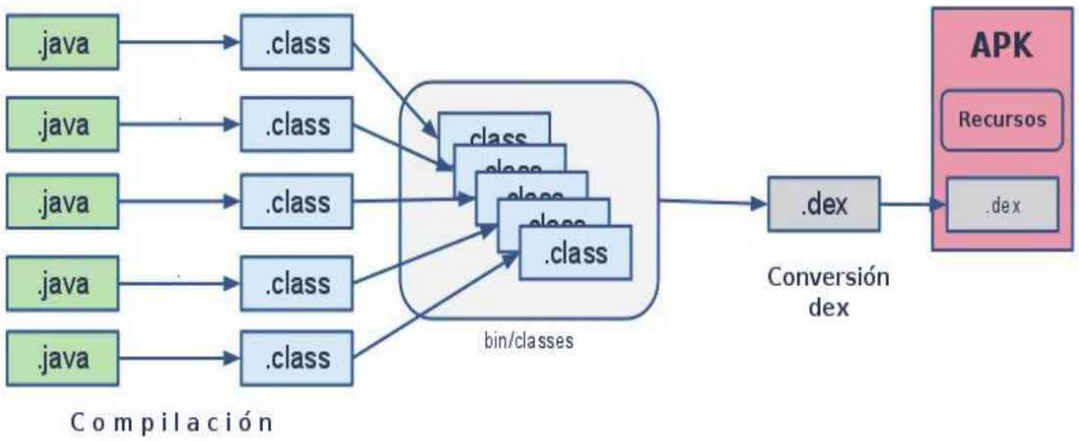

Figura 1.10. Proceso de desarrollo de código DEX para la máquina virtual Dalvik

1.4.5 Entorno de aplicaciones

El tercer nivel en la arquitectura Android lo constituye el *framework* o entorno de aplicaciones. En él se halla implementada la interfaz de programación de aplicaciones (API), elemento imprescindible para que los desarrolladores puedan escribir software destinado a plataformas Android. Mediante la API, el programador accede a una variedad considerable de características y servicios del sistema operativo Android sin tener que conocer los detalles de su funcionamiento interno. Gracias al *framework* las aplicaciones que funcionan en el último nivel de la pila Android tienen acceso a las funcionalidades básicas del sistema: administración de actividades, geolocalización, notificaciones, proveedores de contenido, etc.

1.4.6 Aplicaciones de usuario

El último estrato de la arquitectura Android lo compone la capa de aplicaciones. Este es el nivel con el que el usuario interactúa a través de los iconos de la pantalla y el teclado virtual. Como se ha dicho antes, las aplicaciones están escritas en Java y se ejecutan en la máquina virtual Dalvik. Además de las que vienen preinstaladas en el sistema (Contactos, Gmail, reproductor de medios, navegador de Internet), el usuario dispone de gran variedad de utilidades gratuitas y de pago: programas de gestión, entretenimiento, herramientas del sistema, etc. Solo en Google Play (anteriormente denominado Android Market) había registradas en agosto de 2014 más de 1.300.000 aplicaciones (Figura 1.11).

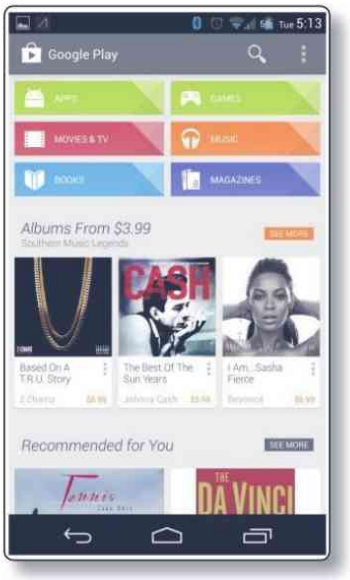

Figura 1.11. Interfaz de Google Play

1.5 ¿CÓMO ES UNA APLICACIÓN ANDROID?

El SDK (*software development kit*) o conjunto de recursos de desarrollo de Android hace posible que incluso desarrolladores con un nivel elemental de conocimientos estén en condiciones de crear cualquier tipo de software, en poco tiempo y con una curva de aprendizaje llevadera. Lo único que se necesita es tener nociones de Java y algo de experiencia en el manejo de clases.

Posteriormente, las aplicaciones son subidas a Google Play, desde donde se descargan empaquetadas en archivos APK que incluyen todo lo necesario para la instalación: archivo `AndroidManifest.xml`, ejecutables *.dex* y datos del programa. A continuación se presentan algunos componentes básicos que el investigador encontrará en todas las aplicaciones Android.

1.5.1 Manifiesto

Toda aplicación Android viene acompañada de un archivo llamado `AndroidManifest.xml` que contiene información general sobre ella: nombre del programa, versión de SDK utilizada por el desarrollador, permisos solicitados durante la instalación, requerimientos hardware y software y otros datos por el estilo. Este archivo es de tipo texto y está escrito en formato XML. La información incluida en el manifiesto se utiliza para instalar el programa. A través de un menú se informa al usuario sobre los permisos que debe autorizar. En función de lo que se haya decidido, el sistema determina si aquella es instalada o no. La lectura de estos avisos y la autorización de permisos es un elemento clave en el sistema de seguridad de los dispositivos Android. El manifiesto también contiene un listado de las diferentes pantallas o *activities* que integran la aplicación.

AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example"
    android:versionCode="1"
    android:versionName="1.0">

    <uses-sdk android:minSdkVersion="15" />

    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:label="@string/app_name"
        android:icon="@drawable/ic_launcher">
        <activity
            android:name="MyActivity"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Figura 1.12. Aspecto del archivo AndroidManifest.xml

El manifiesto proporciona al investigador forense información que puede resultar valiosa para detectar infecciones de *malware*, puertas traseras, software espía o cualquier tipo de programas utilizados con alguna finalidad delictiva.

1.5.2 Activities

En el entorno Android, una *activity* consiste en una plantilla perteneciente a una aplicación que ocupa una pantalla del terminal, representando una unidad funcional vinculada al programa. En todo momento durante el manejo de una aplicación Android, el elemento con el que el usuario interactúa directamente, con sus ojos, sus dedos, un *smart pen* o la voz, es en primera instancia la *activity* actual. Por ejemplo: cuando iniciamos la aplicación y se nos muestra la pantalla principal con un menú o mandos de control que permiten cargar otras pantallas. Cada una de estas pantallas es una *activity*. Todas ellas han de estar declaradas en el manifiesto: en caso contrario el sistema operativo no permitirá su ejecución.

Estas *activities* utilizan recursos —principalmente memoria RAM— con una prioridad que viene determinada por el sistema operativo. Dentro de cada aplicación existe un ciclo de *activities* establecido por la programación y la política del sistema. Un cambio de pantalla no significa que la *activity* con la que interactuábamos en un momento anterior se haya perdido, sino que permanece suspendida hasta el momento en que sea necesario traerla nuevamente a primer plano. Cuando los datos correspondientes a la *activity* ya no son necesarios, aquellos se borran. Y si la actividad permanece suspendida durante algún tiempo, el sistema la elimina liberando recursos que pueden dedicarse a otras tareas.

1.5.3 Intents

Los *intents* son paquetes de datos necesarios para que las aplicaciones puedan establecer comunicación cada vez que intercambian datos, se llaman unas a otras o ejecutan acciones pertenecientes al ámbito de otro programa. Esta funcionalidad no debe ser confundida con la del IPC Binder, que tiene lugar entre procesos de bajo nivel en el ámbito reservado de ejecución del sistema operativo. Cada *intent* posee dos atributos principales —datos y acciones— y tres de carácter opcional —tipos, categorías y extras—. La acción representa lo que la aplicación debe hacer con los datos transportados por el *intent*. Los atributos opcionales indican tareas que se han de ejecutar en caso de que los datos no respondan a la sintaxis del URI (identificador uniforme de recursos).

Los *intents* pueden ser de dos clases: explícitos o implícitos. Un *intent* explícito va dirigido a un módulo de software receptor que ya espera la recepción

de aquel por estar indicado en el destinatario del *intent*. Por el contrario los *intents* implícitos, que no tienen asignado un destino, son puestos a disposición del gestor de paquetes de Android para que sea este quien decida a qué aplicaciones debe hacerlos llegar. Siempre que un *intent* no sea descartado por ambigüedad o incoherencia de datos es transferido por el sistema a un *broadcast receiver* o receptor de multidifusión.

1.5.4 Broadcast receivers

Algunas *activities* pueden actuar como receptoras de eventos y procesar directamente la información contenida en los *intents*. En caso de que no sea así, el elemento adecuado para cumplir dicha función es un *broadcast receiver* o receptor de multidifusión. Los *broadcast receiver*s recogen el *intent*, analizan los datos transportados por el mismo y en función de aquellos lanzan la aplicación o *activity* encargada de procesarlos.

1.5.5 Proveedores de contenidos

Los *content providers* (proveedores de contenido) son estructuras de datos a las que han de acceder en algún momento las aplicaciones Android. Ejemplos típicos son los contenedores para mensajes de correo electrónico, contactos, coordenadas del sistema de geolocalización o archivos audiovisuales almacenados en un soporte de datos y accesibles a través de peticiones o enlaces. En todos estos casos los datos son guardados por lo general en tablas SQLite, accesibles desde una aplicación a través de funciones exportadas por librerías.

1.5.6 Servicios

Los servicios (*services*) son elementos de software que se ejecutan en segundo plano sin una interfaz explícita de interacción con el usuario. También vienen declarados en el manifiesto. En sentido estricto no son imprescindibles para el funcionamiento del terminal, pero sin ellos resulta imposible realizar acciones como descargar archivos de Internet en segundo plano o escuchar música mientras el usuario ejecuta una aplicación o navega por Internet. Los servicios manejan archivos, monitorizan procesos de comunicación, gestionan el tráfico de la red y suministran información —por ejemplo, coordenadas geográficas— a otras aplicaciones. Para acceder a los servicios las aplicaciones no se sirven de *intents*, sino de recursos de bajo nivel de IPC Binder.

1.6 SISTEMAS DE ARCHIVOS

El *kernel* de Linux, a través de módulos adecuados, puede manejar la mayor parte de los sistemas de archivos utilizados en la actualidad: no solamente los típicos del entorno Unix (ext2, ext3 y ext4) sino también sistemas de archivos Microsoft (FAT16, FAT32 y NTFS) y de red (JFS) o para aplicaciones especiales, como por ejemplo el sistema utilizado para montar las particiones internas del sistema, el árbol de procesos `/proc` y otras estructuras de datos virtuales necesarias para el funcionamiento del sistema operativo. Android reconoce gran cantidad de sistemas de archivos, aunque en la práctica solo utiliza tres o cuatro.

Estos sistemas de archivos constituyen objetivo prioritario de la investigación forense. Dentro de ellos suele haber documentos, imágenes, archivos de sonido y/o vídeo, datos de archivos borrados pero aún accesibles bajo nivel y otros elementos de evidencia. En las páginas siguientes se hablará sobre los diferentes sistemas de archivos que por lo general el investigador tiene que analizar durante la investigación de un dispositivo móvil Android.

1.6.1 YAFFS/YAFFS2

El sistema de archivos nativo de Android YAFFS (*yet another flash file system*) fue desarrollado para adaptarse a las características específicas de las memorias NAND, dispositivo de almacenamiento de datos utilizado por la mayor parte de los terminales Android y otros aparatos móviles con sistemas operativos empotrados. YAFFS es código libre, sujeto a la licencia GPL con posibilidad de incorporar cláusulas menos restrictivas para fabricantes que por diversas razones no estén en situación de aceptar las condiciones de la Licencia General Pública. Incluye mecanismos que optimizan los procesos de lectura en un entorno de recursos limitado, distribuyendo de manera homogénea el desgaste producido en los circuitos del chip por las operaciones de escritura y lectura. También es capaz de gestionar bloques defectuosos de memoria NAND, generando una lista de ellos para que el sistema prescinda de ellos en operaciones de escritura y borrado de datos. Finalmente, es un sistema de archivos rápido, gestionado por un controlador que no sobrecarga la memoria RAM. YAFFS2 es una versión más avanzada, que se caracteriza por la capacidad para manejar particiones y archivos de mayor tamaño que su predecesora.

En los primeros tiempos, Android se servía de YAFFS para aquellas particiones en las que viene instalado el sistema operativo con aplicaciones de fábrica. También guardaba los datos del usuario —contactos, cuentas de correo, aplicaciones instaladas desde el Android Market, archivos temporales, etc.—. En la actualidad, YAFFS es un sistema de archivos en desuso, y ha sido reemplazado

por Linux ext4, principalmente por razones de rendimiento y compatibilidad con las nuevas CPU *multicore*. Sus escasas perspectivas de futuro no constituyen el mejor incentivo para el desarrollo de herramientas de análisis forense. En la práctica el contenido de una partición YAFFS solo puede ser estudiado con la ayuda de un editor hexadecimal o mediante aplicaciones creadas específicamente para tal fin por alguien que posea un conocimiento profundo de las estructuras de datos utilizadas por YAFFS.

1.6.2 FAT32

Es habitual que los dispositivos Android dispongan de una o varias particiones formateadas con sistemas de archivos Microsoft FAT32, generalmente en la tarjeta de expansión SD o en la tarjeta interna eMMC simulada en la memoria NAND. La razón de que un fósil informático que data de los comienzos de MS-DOS haya sobrevivido hasta la era móvil, y que además continúe siendo utilizado con éxito —no solo en *smartphones* sino también en cámaras digitales, reproductores MP3, GPS y otros dispositivos portátiles— es, principalmente, la compatibilidad. Al no disponer de los mecanismos de seguridad y sistemas de permisos característicos de sistemas de archivos más profesionales como NTFS, ext3 o ReiserFS, resulta posible cambiar tarjetas de memoria de unos aparatos a otros, y con el ordenador de sobremesa, sin que haya problemas de acceso ni de permisos. Esto explica la persistente popularidad de FAT32, que convierte a los soportes de datos en receptáculos ideales para guardar archivos de gran tamaño —con la limitación inevitable de los 4 GB—, como multimedia, documentos PDF y descargas de Internet, incluyendo los paquetes APK de las aplicaciones.

Las particiones FAT32 pueden ser adquiridas y analizadas con herramientas forenses de uso habitual. El procedimiento es simple, y en la práctica las únicas dificultades se presentan cuando la tarjeta está colocada bajo la batería y para extraerla es necesario apagar el terminal, arriesgándonos a perder la información volátil que pudiera existir en el dispositivo móvil. Como alternativa, el investigador forense puede intentar la extracción de datos de la tarjeta SD conectando el terminal a un ordenador mediante el cable USB. En el caso de los sistemas de archivo FAT32 emulados en el interior de la NAND en forma de tarjeta virtual eMMC, el proceso de adquisición forense requiere inevitablemente la ayuda de la interfaz USB.

1.6.3 Extended file system (ext)

El sistema extendido de archivos (ext) es un estándar muy antiguo que forma parte por defecto de todas las plataformas Unix y Linux. Desde la versión original creada en 1992 ha habido tres grandes perfeccionamientos: ext2, ext3 y el actual ext4, provisto de *journaling*[4] y otras características avanzadas. Hasta hace poco ext no formaba parte del mundo Android, pero a partir de 2010 Google hizo pública su intención de reemplazar YAFFS por ext4. Las razones para dar este importante paso fueron varias:

- ext4 es un sistema de archivos estándar con soporte para permisos, comandos y procedimientos característicos de Linux.
- Se trata de un sistema estable y de alto rendimiento.
- YAFFS carece de soporte multihilo, lo cual genera cuellos de botella en el rendimiento de los nuevos terminales provistos de procesadores multinúcleo.
- Cada vez son más los *smartphones* que reemplazan sus memorias NAND convencionales por otras más avanzadas y capaces de funcionar como dispositivos de bloque.

En la actualidad no abundan las herramientas forenses que implementan la posibilidad de acceso a particiones ext4. Es de esperar que con la difusión del nuevo sistema de archivos, tanto en terminales móviles como en ordenadores de sobremesa provistos de Linux, se desarrollen módulos correspondientes para el software forense comercial y se incluya soporte ext4 también en utilidades de código libre como The Sleuth Kit. Hasta ese momento, será necesario recurrir al editor hexadecimal y a herramientas de *data carving*.

---

4 El *journaling* es una funcionalidad basada en un registro de transacciones atómicas que permite reparar de manera automática las inconsistencias producidas en las estructuras de datos por un apagado irregular del ordenador. Casi todos los sistemas de archivos modernos disponen de *journaling*. Ejemplos típicos son Microsoft NT, Linux ext3, etx4, ReiserFS o Btrfs. En la práctica forense el *journaling* es conflictivo porque produce cambios en los archivos de registro, que, generalmente, no afectan a la integridad del software ni a los datos de usuario, pero modifican las sumas de verificación de las imágenes de discos duros y particiones adquiridas a bajo nivel. Por ello es necesario realizar este tipo de operaciones a través de conectores de bloqueo o herramientas de software que aseguren el acceso al soporte de datos en modo de solo lectura.

1.6.4 rootfs, devtps, sysfs, proc, tmpfs

Android, como cualquier otro sistema operativo basado en Linux, requiere sistemas de archivos virtuales necesarios para el arranque del sistema o el almacenamiento de información utilizada en tiempo real por los procesos que están siendo ejecutados por la CPU. Por lo general, estos sistemas no contienen datos relevantes para el investigador forense. Su análisis tiene interés a efectos de estudiar defectos de programación de los cuales pudieran derivarse fallos de seguridad aprovechables por virus, troyanos, puertas traseras o cualquier otra forma de software malicioso.

Así, por ejemplo, `rootfs` es el sistema de archivos raíz o vértice en el que coinciden todos los árboles de directorios de un sistema Linux (indicado por el típico *slash* a la derecha [/]). Linux lo necesita para acceder a archivos de configuración y librerías básicas que le permiten completar el proceso de arranque. Por su parte, `devpts` es un sistema de archivos que permite simular sesiones de terminal en un dispositivo Android, por ejemplo durante el establecimiento de una conexión a través de `adb` —se hablará de ello en otro capítulo—. El sistema `sysfs` contiene archivos de configuración y otras estructuras de datos necesarias para el control del dispositivo.

De gran importancia es el sistema de archivos `proc`, complejo árbol poblado con directorios y archivos que incluyen información estructurada sobre el estado de la CPU y los procesos en ejecución, RAM utilizada, parámetros de configuración y otros datos de funcionamiento. Para el investigador forense quizás se trate de la información más valiosa que pueden aportar los sistemas de archivos virtuales de Android. El único inconveniente es que resulta difícil acceder a dicha información, ya que la mayor parte de los sistemas Android, por motivos de seguridad, se inician con permisos limitados que impiden la lectura y escritura de cualquier zona de memoria NAND que no haya sido asignada de manera explícita al usuario normal para instalar aplicaciones o guardar datos. El acceso a `/proc` y otros sistemas de archivos virtuales de Android requiere procedimientos de *rooting*, lo cual significa modificar el terminal en mayor o menor medida. Esto implica un riesgo de alterar medios probatorios, a resultas de lo cual la parte contraria podría impugnar los elementos de evidencia presentados.

Finalmente, `tmpfs` es una ubicación virtual pensada para archivos temporales, y también para alojar áreas de intercambio o caché. Todos estos datos son transferidos a la RAM para que el sistema pueda acceder a ellos con la menor latencia posible. El análisis de estas áreas exige precaución y ha de hacerse con el dispositivo encendido. Los elementos de evidencia no están respaldados en la memoria de estado sólido NAND y pueden perderse en caso de cuelgue del sistema operativo o apagado accidental del *smartphone*.

2

PREPARATIVOS Y HERRAMIENTAS

Antes de proceder a la investigación forense de dispositivos móviles es preciso habilitar en el ordenador de sobremesa —por lo general un PC convencional con Windows o Linux— un área dedicada a operaciones de adquisición, análisis de datos y realización de informes. Esto incluye instalar herramientas de software de las que hablaremos en las páginas siguientes. Primeramente se repasarán conceptos básicos de virtualización. Después veremos el entorno de desarrollo de Android (SDK), imprescindible para acceder al contenido de *smartphones* y *tablets*. Finalmente hablaremos de Linux, sistema operativo que tiene mucho que ver con Android, y haremos un repaso de los comandos habitualmente utilizados por el investigador durante su trabajo.

2.1 MÁQUINAS VIRTUALES

En los apartados correspondientes a Android *runtime* y Dalvik el lector ya ha tenido oportunidad de entrar en contacto con el mundo de la virtualización. ¿Por qué las aplicaciones Android no se ejecutan directamente sobre el sistema operativo, sino en un entorno separado, con su propia zona de memoria a la que ninguna otra aplicación puede acceder de manera directa? Existen para ello razones de seguridad: el desarrollador de teléfonos inteligentes no quiere que en estos minúsculos dispositivos que millones de personas llevan de aquí para allá, conectándose a todos los puntos de acceso wifi que les dan la bienvenida a lo largo del camino, se repita el mismo escenario de vulnerabilidad y caos que ha acompañado al PC desde los comienzos de su historia. Separar funciones es necesario para evitar que los programas interfieran unos con otros.

El concepto de virtualización utilizado por el investigador forense para sus fines es algo más amplio que el de los entornos de ejecución protegidos para aplicaciones Android. En este capítulo se examinan algunas aplicaciones de usuario, instalables en un PC de sobremesa, que el investigador puede utilizar para tareas de análisis. Existen numerosas soluciones de virtualización que permiten ejecutar software fuera de su contexto habitual (Xen, VirtualPC, Cygwin, Wine, Cross Over Office, etc.). En las páginas que siguen nos centraremos en dos utilidades de amplio uso: VMware Workstation, producto comercial con versiones de evaluación gratuitas, y el software de código libre Oracle VirtualBox.

2.1.1 ¿Por qué utilizar máquinas virtuales?

El software de virtualización no es en realidad imprescindible para el análisis forense de dispositivos móviles. Las herramientas y los procedimientos que se describen en los próximos capítulos funcionarían directamente sobre plataformas Windows, Linux y Mac OSX. Sin embargo, la utilización de máquinas virtuales proporciona algunas ventajas. Por ejemplo:

- **Flexibilidad**: a la hora de utilizar herramientas forenses el investigador no está condicionado por una plataforma concreta. Es posible que en determinados casos necesite recurrir a una versión diferente de Android SDK o emplear comandos de Linux cuyas versiones para Windows no ofrecen las mismas prestaciones. En tales circunstancias las máquinas virtuales pueden ser de ayuda.

- **Aislamiento de aplicaciones**: en investigaciones sobre software malicioso interesa que el material peligroso esté separado del entorno anfitrión para evitar contagios. Aunque es poco probable que un *malware* desarrollado para Android pueda infectar una estación de trabajo, la prudencia nunca está de más. La utilización de máquinas virtuales permite tener los elementos de evidencia confinados en un entorno específico dedicado al caso correspondiente, evitando una dispersión incontrolada de los mismos por otras áreas de trabajo.

- **Portabilidad**: habiendo preparado una máquina virtual para investigar un caso, podremos trasladarla fácilmente a otra estación de trabajo provista del mismo software de virtualización con solo copiar a un disco duro externo los archivos correspondientes. De este modo también resulta posible facilitar material a los peritos de la parte contraria o a otros investigadores. Lo único que necesitarán para verlo es un PC con VMware o VirtualBox.

- **Archivado**: si guardamos la máquina virtual correspondiente a un caso, con la misma configuración, herramientas de análisis, imágenes a bajo nivel de los medios adquiridos y otros elementos de evidencia que se hayan podido encontrar, resultará fácil volver al trabajo en cualquier momento. La máquina virtual puede dejarse suspendida en un estado de funcionamiento determinado. Si al cabo de varios meses es necesario revisar el caso, bastará reiniciarla pulsando un botón y recuperaremos nuestro entorno de trabajo en el instante en que lo habíamos dejado, con las aplicaciones en ejecución y los datos cargados.

2.1.2 Principios de virtualización

La virtualización hace posible que las aplicaciones de software funcionen con independencia de plataformas de hardware y sistemas operativos subyacentes. También permite optimizar el rendimiento de la infraestructura y ahorrar energía en centros de datos y servidores empresariales. Aunque parezca un invento innovador vinculado al desarrollo de Internet y el *cloud computing*, la virtualización es una tecnología conocida desde los primeros tiempos de la revolución informática, cuando se utilizaban ordenadores de gran tamaño (*mainframes*) para atender a necesidades de gestión en organismos públicos y grandes empresas. La razón para ello era principalmente de rentabilidad: de este modo se evitaba tener costosos equipos de hardware aprovechados en un porcentaje muy bajo de su capacidad.

El elemento central de todo sistema de virtualización lo constituye un elemento denominado hipervisor o VMM (monitor de máquinas virtuales), el cual establece una capa de abstracción entre el hardware de la máquina física y el sistema operativo de la máquina virtual, distribuyendo los recursos disponibles —CPU, memoria RAM, espacio de almacenamiento en los discos duros— entre varios entornos de ejecución (Figura 2.1). Esto permite que se puedan tener varios ordenadores virtuales ejecutándose en la misma plataforma de hardware.

Existen diversas tecnologías para convertir plataformas informáticas en entornos virtualizados: podemos virtualizar hardware de servidor, software de servidor y otros recursos. También se pueden virtualizar sesiones de usuario, aplicaciones y elementos de red, incluso es posible crear una red local formada por varias máquinas virtuales dentro de un PC de sobremesa. Microsoft, por ejemplo, a través de sus soluciones basadas en Hyper-V, es un destacado proveedor de software de virtualización integral para entornos empresariales que abarca todos los ámbitos de computación.

En la actualidad, con los modelos de negocio basados en la nube, las tecnologías de virtualización se han convertido en un componente fundamental de las modernas infraestructuras de red.

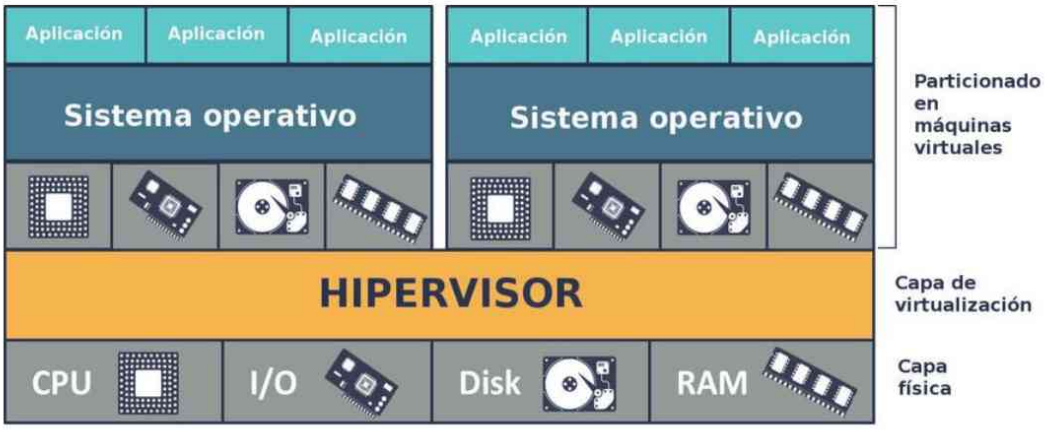

Figura 2.1. Así funciona la virtualización

El concepto de virtualización que el investigador utilizará a lo largo de las páginas siguientes es específico y adaptado al tipo de tareas que interesa llevar a cabo. Se trata de instalar en un ordenador una aplicación que permite simular una plataforma de hardware con todos los recursos de un PC convencional: CPU, memoria, discos duros, adaptadores de red, puertos USB e interfaces para periféricos de entrada y salida. Aunque podemos pensar en esta plataforma como en algo real, compuesto por placas de circuitos y cables, en realidad se trata de un conjunto de parámetros de funcionamiento que hacen posible un uso compartido de recursos proporcionados por el sistema anfitrión.

Sobre esta plataforma virtualizada instalamos un sistema operativo del tipo y la versión que nos permitan las características de la máquina virtual creada por el usuario, con sus aplicaciones y ajustes específicos. Este entorno —sistema huésped (*guest*)— funcionará con cierto grado de independencia respecto de la plataforma original —anfitrión (*host*)— instalada sobre el hardware. Así, podemos tener una máquina virtual Windows 7 o XP ejecutándose sobre Linux Ubuntu, o viceversa; o diferentes versiones de Windows o Linux funcionando sobre un sistema anfitrión del mismo tipo. También podemos habilitar máquinas virtuales con el mismo o con diferentes sistemas operativos para que funcionen de manera concurrente.

2.2 VMWARE WORKSTATION

VMware Inc. es una empresa de California que desarrolla soluciones de virtualización para la empresa y el usuario particular. Su gama de productos cubre todo el espectro de necesidades de cualquier entorno productivo moderno: desde servidores empresariales e hipervisores *bare-metal*, capaces de funcionar directamente sobre el hardware del ordenador sin necesidad de un sistema operativo anfitrión, destinados a centros de datos y entornos de alta productividad, hasta aplicaciones para PC y usuarios particulares. También dispone de versiones de evaluación que permiten someter a prueba sus herramientas comerciales durante un período de tiempo determinado —generalmente un mes— y productos gratuitos (como VMware Server).

El más útil para el investigador, por sus prestaciones, su versatilidad y el coste relativamente bajo de las licencias, es VMware Workstation. Este software dispone de versiones para Windows y Linux que se pueden descargar desde la página web de VMware. Admite como huéspedes la mayor parte de los sistemas operativos, desde el arcaico MS-DOS hasta Windows 8 (en VMware Workstation 9.0), incluyendo Linux, Solaris, BSD, IBM OS2 y sistemas OSX desarrollados para las CPU utilizadas por Apple en sus nuevas líneas de ordenadores iMac Core Intel y MacBook Pro. VMware Workstation requiere desde su versión 8.0 que la plataforma que vaya a ser utilizada como anfitrión sea de 64 bits o compatible.

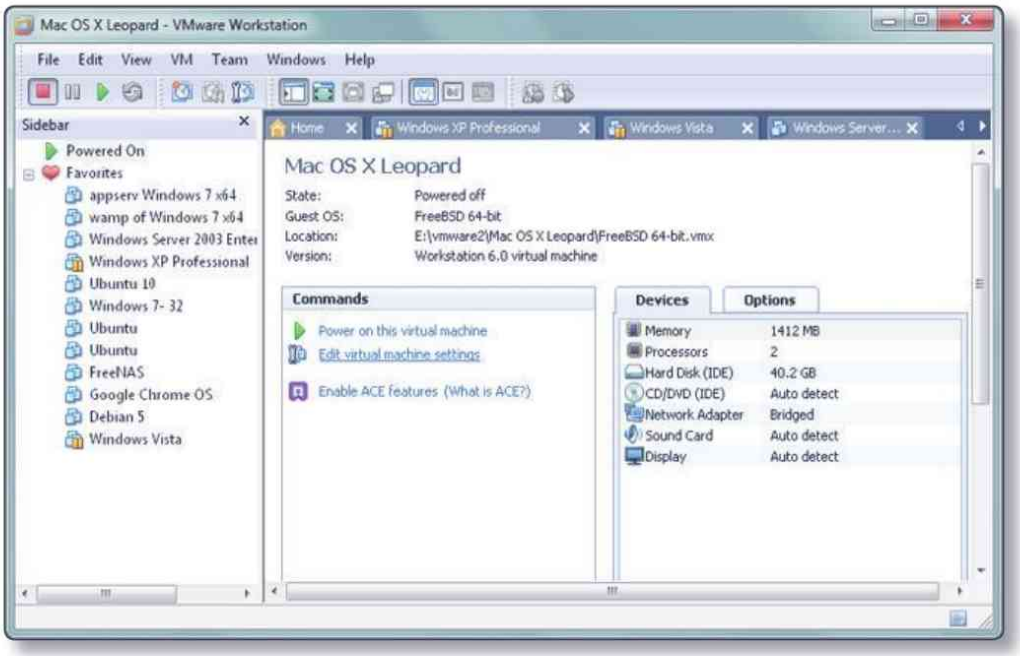

Figura 2.2. Ventana principal de VMware Workstation

2.2.1 Características

VMware Workstation (Figura 2.2) es un hipervisor que funciona sobre plataformas Intel y AMD con CPU de 64 bits. Permite al usuario crear y gestionar múltiples máquinas virtuales y utilizarlas de modo simultáneo con el software que corre directamente sobre el sistema operativo anfitrión. Cada una de estas máquinas virtuales ejecuta su propio sistema operativo con el conjunto de herramientas que el usuario ha instalado sobre él —suites ofimáticas, navegadores de Internet, software de análisis forense—. VMware Workstation permite compartir carpetas con el sistema anfitrión y soporta interfaces de red en diferentes modos de acceso al hardware de red físico: *bridging*, NAT o aislamiento total. También es capaz de montar imágenes ISO y acceder a las mismas como si fueran soportes de datos físicos. Tanto las máquinas virtuales VMware como los discos duros virtualizados se guardan en archivos con extensión *.vmdk* y pueden ser llevados con facilidad de unos ordenadores a otros. VMware Workstation permite interrumpir el funcionamiento de las máquinas en un estado de funcionamiento concreto, guardando la RAM en un archivo especial, de modo que resulte posible reiniciar posteriormente una máquina en el mismo punto (*snapshot* o momento fotográfico) en que el usuario la dejó congelada.

Se trata de una herramienta útil para todo aquel que en un momento dado necesite disponer de una plataforma distinta a la que está utilizando pero no pueda permitirse un PC adicional para instalar nuevo software, ni esté en situación de llevar a cabo modificaciones trabajosas y complejas, como instalar un nuevo disco duro o reparticionar los existentes para trabajar con otro sistema operativo o con una configuración distinta. Las herramientas de virtualización resultan de gran ayuda, por ejemplo, para desarrolladores de páginas web que necesiten probar su proyecto bajo diversas plataformas, o para técnicos de servicio al cliente de un banco encargados de atender consultas procedentes de usuarios de banca *on line* que acceden a sus cuentas desde sistemas operativos diferentes.

Gracias al entorno de ejecución aislado de VMware Workstation, a la facilidad de manipulación de archivos *.vmdk* y al soporte USB, el investigador puede conectar dispositivos Android, realizar con ellos todo tipo de operaciones y extraer elementos de evidencia para analizarlos con herramientas forenses.

2.2.2 Localización, descarga y recursos web

VMware Workstation, junto con otros productos para empresas y usuarios finales, se encuentra disponible en la página web del desarrollador, la cual está organizada en modo similar a un portal de Internet que el usuario podrá explorar en busca de información sobre soluciones informáticas, tecnologías de virtualización

y *cloud computing*, direciones de contacto, soporte técnico y otros recursos *on line* como documentación técnica, foros, eventos y notas de prensa.

Desde la pestaña **Soporte y descargas** un menú permite al usuario trasladarse directamente a la página de descargas de VWware Workstation. Aquí podrá decidir si adquiere el producto o prefiere probar antes la versión de evaluación. Para cualquiera de estas opciones debe conseguir una clave de producto. Requisito para ello, lo mismo que para descargar el software, es disponer de una cuenta en VMware. Si el usuario aún no la tiene, puede crearla a través de un formulario. Finalmente, no tiene más que seleccionar el sistema operativo anfitrión (Windows o Linux) sobre el que tiene previsto instalar VMware Workstation y descargar la aplicación en forma de un *bundle* o paquete que se instala haciendo doble clic encima. En la misma página de descargas el usuario hallará otros recursos: FAQ, condiciones de compra, soporte técnico, recomendaciones de uso, guías de instalación rápida, manuales en varios idiomas, hojas de datos del producto con requerimientos de hardware y otras informaciones de interés.

La instalación de VMware Workstation sobre Windows no ofrece mayores dificultades que la de cualquier otro software de productividad. En Linux, por el contrario, el proceso suele ser algo más complejo. Si el investigador trabaja con una distribución popular como Ubuntu o Fedora no deberían presentarse dificultades, salvo alguna que otra incompatibilidad en los módulos de compilación del *kernel*. Para solucionarlas el usuario puede recurrir a la ayuda en línea de VMware o a foros de Internet.

2.2.3 Creación de máquinas virtuales

De manera aproximada podemos describir el funcionamiento de VMware en términos de un apilamiento cuya base estaría constituida por el hardware de la máquina física, en nuestro caso un PC de sobremesa con CPU Intel de 64 bits. Sobre esta plataforma de hardware corre un sistema operativo Windows o Linux. Encima de este nos encontramos con la aplicación VMware Workstation, que es el software hipervisor de nuestro entorno de virtualización. Sobre este, a su vez, funciona una máquina virtual que emula un hardware con determinadas especificaciones y recursos, y que soporta el sistema operativo huésped —Windows, Linux, BSD o cualquier otro compatible con procesadores x64—. Finalmente estarían las aplicaciones utilizadas por el usuario en su entorno virtualizado. Yendo en sentido contrario, es decir, de arriba abajo, la disposición sería esta:

Aplicación → Sistema operativo *guest* → hardware virtualizado (máquina virtual) → VMware Workstation → Sistema operativo *host* → Hardware físico.

Para que todo esto pueda funcionar es preciso que el usuario haya creado la máquina virtual o la haya traído de otras plataformas provistas de una versión igual o inferior de VMware Workstation. También existe la posibilidad de virtualizar el funcionamiento de sistemas instalados directamente sobre el hardware del ordenador, crear discos duros virtualizados o convertir máquinas virtuales creadas por otras aplicaciones de virtualización —como VirtualBox o Qemu— al formato de VMware. La creación de una máquina virtual se lleva a cabo a través de un asistente mediante el cual el usuario establece las características del hardware que quiere virtualizar: memoria RAM, tamaño del disco duro, tipo de las conexiones de red con otras máquinas virtuales y con el anfitrión, carpetas compartidas, interfaces USB y acceso a la grabadora DVD/CD o a otros periféricos.

Figura 2.3. Asistente para la creación de máquinas virtuales en VMware

El proceso de creación de una máquina virtual es simple y se lleva a cabo en pocos minutos. VMware dispone de un asistente para simplificar la tarea (Figura 2.3). Después viene la parte más tediosa, que consiste en instalar el sistema virtualizado y las aplicaciones de productividad. Esta etapa suele ser, salvo incompatibilidades muy específicas de aplicaciones que requieren acceso directo al hardware, indistinguible de una instalación del software sobre ordenadores reales. Los medios utilizados para ello —discos ópticos, llaves USB, ejecutables, paquetes descargados desde repositorios o páginas web— son también los mismos.

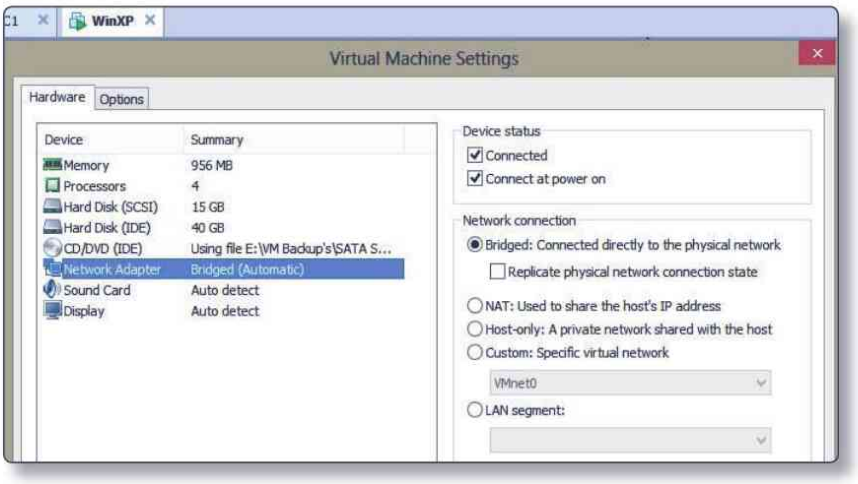

Figura 2.4. Configuración de máquinas virtuales en VMware

El usuario debe tener en cuenta que los fabricantes de software propietario no hacen distinción entre máquinas reales o virtuales. Por tanto hay que cumplir los términos de las licencias y tener a mano la documentación correspondiente. No hacerlo puede causar problemas si se piensa comparecer en juicio o poner a disposición de los peritos de la otra parte elementos de evidencia adquiridos por medio de herramientas que funcionan sobre entornos virtualizados.

VMware puede manejar varias máquinas virtuales de manera simultánea. El funcionamiento del hipervisor es compatible con la ejecución paralela de cualquier otro software —por ejemplo, Office, aplicaciones de contabilidad, retoque fotográfico, etc.— que el usuario tenga instalado sobre su sistema operativo anfitrión. De este modo resulta posible tener en una misma máquina varios sistemas operativos ejecutándose al mismo tiempo, y mantener abiertas aplicaciones desarrolladas para plataformas diferentes e incluso incompatibles, como Office, Photoshop, LibreOffice, utilidades Unix, etc. Los parámetros de la máquina virtual son configurables y pueden modificarse de acuerdo con las necesidades y preferencias del usuario (Figura 2.4).

2.3 VIRTUALBOX

Además de las diferencias de diseño y las características de funcionamiento, VirtualBox presenta algunas ventajas con respecto a VMware: inicio más rápido del hipervisor, mayor velocidad, instalación sencilla y libre de complicaciones burocráticas (VirtualBox se halla disponible en los repositorios de numerosas distribuciones Linux). El propietario de la aplicación —Oracle Corporation— ofrece el sofware bajo dos modalidades de licencia: una privativa para fines empresariales y otra pública para uso particular. VirtualBox, creado por la empresa alemana Innotek,

es un software de virtualización para arquitecturas x86/amd64 que soporta sistemas operativos GNU/Linux, Mac OSX, OS/2 Warp, Microsoft Windows, Solaris/ OpenSolaris, FreeBSD, Solaris, MS-DOS y otros.

2.3.1 Características

Originalmente la aplicación era distribuida bajo licencia privativa, pero en enero de 2007 Oracle puso VirtualBox OSE (Open Source Edition) a disposición del público bajo licencia GPL 2. El usuario, de acuerdo con sus preferencias, puede escoger la versión que más le convenga: Oracle VM VirtualBox, privativa y gratuita siempre que se cumpla esta cláusula fundamental de la licencia: "para uso personal y de evaluación" (VirtualBox Personal Use and Evaluation License [PUEL]), y VirtualBox OSE (Open Source Edition), versión de software libre y sometida a los términos de la licencia GPL.

Figura 2.5. Ventana principal de VirtualBox

VirtualBox permite la ejecución remota de maquinas virtuales a través del RDP (*remote desktop protocol*). También dispone de soporte iSCSI. Estas opciones, junto con el soporte USB, no están disponibles en la versión OSE. Más adelante trataremos sobre el modo de incluir soporte USB en máquinas virtuales, algo imprescindible para la conexión de terminales Android con cable USB. En el PC anfitrión, los discos duros de los sistemas invitados se guardan como archivos pertenecientes a un contenedor denominado VDI (*virtual disk image*). VirtualBox puede montar imágenes ISO como discos duros o unidades virtuales ópticas de CD o DVD. La aplicación dispone de controladores que permiten emular aceleración 3D, visión en pantalla completa y adaptadores PCI Ethernet.

El manejo del programa y el procedimiento para crear máquinas virtuales son similares a los de VMware, mediante una interfaz en la que se muestran las máquinas virtuales disponibles con sus características y su estado de funcionamiento. A continuación se describen los procedimientos habituales para instalar VirtualBox en los sistemas operativos de uso más general: Microsoft Windows, GNU Linux y Apple OSX.

2.3.2 Instalación en Windows

Instalar VirtualBox en Windows resulta tan sencillo como instalar cualquier otra aplicación. El único requisito es disponer de la versión 1.1 o superior del Windows Installer, lo cual está asegurado si nuestro sistema operativo es Windows 7 o Vista con el Service Pack de Microsoft más reciente y las últimas actualizaciones. Lo único que se necesita es descargar el paquete desde la página web de VirtualBox y hacer doble clic con el ratón. Otra posibilidad es abrir una consola de texto en carpeta de descarga y descomprimirlo a mano con el comando:

```
virtualbox.exe –extract
```

Esto extrae el contenido en un directorio temporal. El paquete incluye software para PC x86/amd64. De los dos archivos con extensión *.msi* el usuario debe elegir la que le convenga. Si tiene un sistema de 64 bits cualquiera de ellas sirve, aunque la versión recomendada, por supuesto, es amd64:

```
msiexec /i virtualbox-<version>-multiarch_<x86|amd64>.msi
```

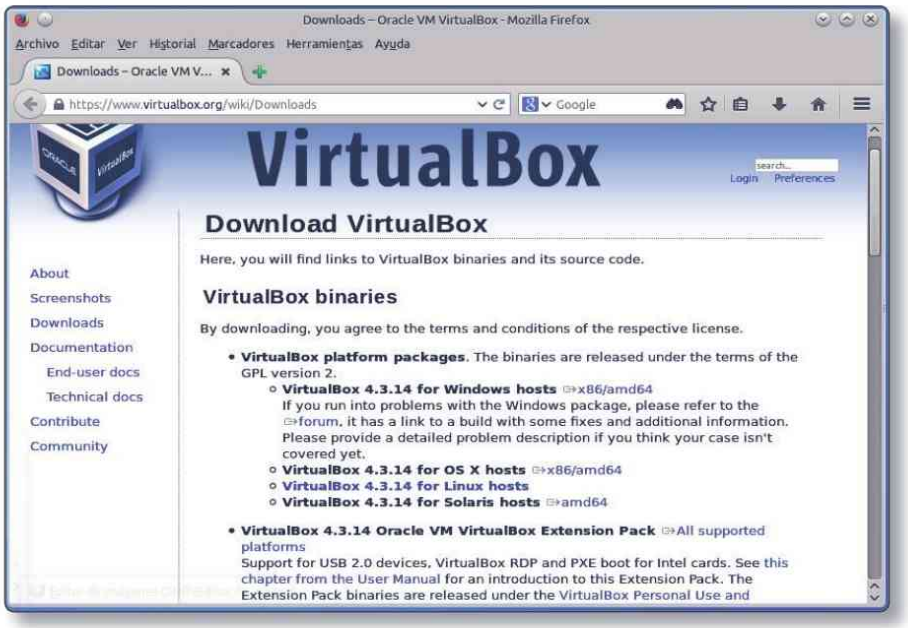

Figura 2.6. Página web del proyecto VirtualBox

Es probable que Windows envíe mensajes de advertencia al detectar controladores no firmados por Microsoft (*unsigned drivers*). Responda pulsando con el ratón en **Continuar**, ya que de lo contrario VirtualBox podría no funcionar correctamente tras la instalación. Windows crea en el menú de inicio un grupo de programas llamado **Virtualbox**, que permite al usuario iniciar la aplicación y acceder a los archivos de ayuda.

Por defecto Virtualbox se encuentra a disposición de todos los usuarios del sistema. En el supuesto de que el investigador quiera restringir el uso del programa a su propia cuenta (usuario actual), el paquete descargado desde Oracle deberá instalarse mediante la consola de comandos con los parámetros siguientes:

```
virtualbox.exe –msiparams ALLUSERS=2
```

O bien, primero extraer el contenido:

```
virtualbox.exe –extract
```

Y después:

```
msiexec /i virtualbox-<version>-multiarch_<x86|amd64>.msi ALLUSERS=2
```

2.3.3 Soporte USB

Antes de la versión 4, VirtualBox se distribuía en paquetes diferentes de acuerdo con el tipo de licencia necesario para utilizar el producto. OSE no incluía por defecto soporte para interfaz USB. Si el usuario necesitaba esta funcionalidad se veía obligado a descargar el paquete de la versión privativa y aceptar las condiciones de la licencia PUEL (Personal Use and Evaluation License). Este requisito legal aún existe, pero el procedimiento de instalación se ha simplificado.

Ahora solo existe una versión. Las características especiales —como VBoxUSB, VBoxNetwork, etc.— y cualesquiera otras que pudieran hallarse sometidas a las condiciones de la licencia privativa, se incluyen en paquetes complementarios descargables con los archivos de cada versión o activables durante la instalación. Por ejemplo, para incluir soporte USB durante la instalación de los binarios, el procedimiento en Windows sería algo parecido a esto:

```
virtualbox.exe –msiparams ADDLOCAL=VBoxApplication,VBoxUSB
```

O bien, como ya se ha visto, primero descomprir el paquete:

```
virtualbox.exe –extract
```

Y a continuación:

```
msiexec /i virtualbox-<version>-multiarch_<x86|amd64>.msi ADDLOCAL=
VBoxApplication,VBoxUSB
```

Si el usuario tiene dudas, o trabaja con configuraciones atípicas de Windows o con versiones anteriores de VirtualBox, antes de hacer nada deberá informarse adecuadamente en la página web de Oracle.

2.3.4 Instalación en Linux

Si el investigador trabaja con distribuciones populares de Linux como Ubuntu o Fedora, probablemente los paquetes de VirtualBox ya están disponibles en el repositorio. Con Ubuntu no tiene más que abrir un terminal de comandos y teclear lo siguiente:

```
sudo apt-get install virtualbox
```

Antes de la versión 4, la versión existente en repositorios Linux solía ser OSE. Para disponer de soporte USB y otras características, el usuario tenía que desinstalar OSE y descargar desde la web de Oracle los paquetes de la versión PUEL. En la actualidad lo único que hace falta es añadir las extensiones. En cada versión de VirtualBox existe un paquete de extensiones válido para todas las plataformas. Basta descargarlo desde la página web de Oracle e instalarlo desde la misma aplicación VirtualBox.

Si por las razones que sea (por ejemplo disponibilidad de una distribución antigua de Ubuntu sobre plataforma de 32 bits) el investigador se ve obligado a instalar una versión concreta, existe la posibilidad de localizar el paquete de Debian e instalarlo manualmente. Supongamos por ejemplo que alguien necesita software para la distro Ubuntu Karmic de 32 bits:

```
sudo dpkg –i VirtualBox-3.2.4.2.12,Ubuntu.Karmic.i386.deb
```

No olvide aceptar los términos de la licencia PUEL. Tras la instalación, el programa intentará seleccionar un módulo compatible con la versión actual del *kernel*. Si entre los que vienen incluidos en el paquete no hay uno que sirva, es el propio usuario quien debe compilarlo. Si se producen errores deberá consultar el manual de instalación de VirtualBox. Una vez solucionado el problema, tan solo queda ejecutar las rutinas de configuración de módulos:

```
sudo /etc/init.d/vboxdrv setup
```

Alternativamente, VirtualBox puede instalarse a partir de un archivo con extensión *.run* descargado desde la página de Oracle y ejecutable bajo diversas distribuciones Linux. Lo que hace este instalador, tras pulsar sobre él, es descomprimir la aplicación en un directorio de destino, por ejemplo `/opt/VirtualBox/`. A continuación compila e instala los módulos para el *kernel* (`vboxdrv`, `vboxnetflt` y `vboxnetadp`). Después crea un grupo de usuarios denominado `vboxusers`, genera los enlaces simbólicos necesarios para el inicio de los ejecutables, facilita a `udev` las instrucciones necesarias para que todos estos servicios estén disponibles tras el arranque del sistema y registra el directorio de instalación en el archivo `/etc/vbox/vbox.cfg`.

Sobra añadir que el archivo *.run* ha de ser ejecutado con permisos de administrador o `root`. Para seguir el procedimiento anterior es necesario abrir un navegador de archivos con privilegios de superusuario, tecleando en línea de comando:

```
sudo konqueror
```

Si es usted un usuario de los de viejo cuño, probablemente piense que hacer esto no tiene mucho sentido, y no le falta razón. ¿Para qué vamos a abrir un navegador si ya tenemos la consola? Todo experto Linux que se considere digno de tal título prefiere buscarse la vida a golpe de teclado:

```
sudo ./VirtualBox.run install|unistall [...Opciones]
```

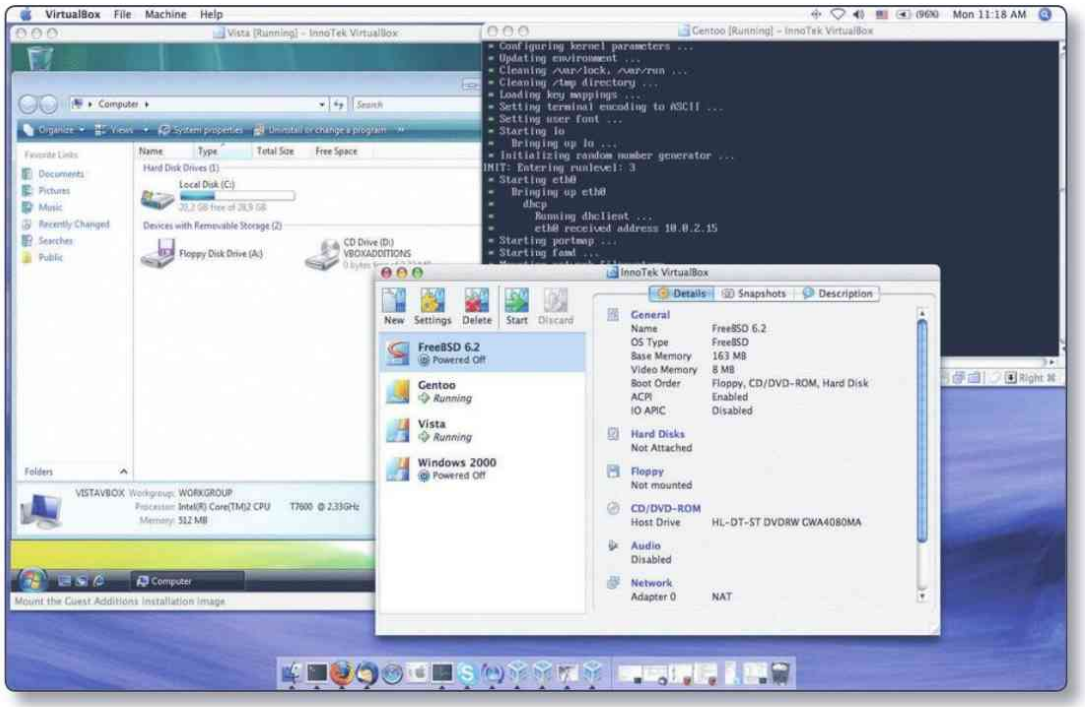

Figura 2.7. VirtualBox también funciona en un Mac… de los nuevos

2.3.5 Instalación en OSX

Para entornos Mac, VirtualBox se distribuye en archivos de imagen de disco DMG. El procedimiento de instalación es similar al de las aplicaciones OSX:

1. Montaje de la imagen DMG haciendo doble clic sobre el archivo.

2. En la ventana abierta, pulsar dos veces con el ratón sobre el instalador **VirtualBox.mpkg**.

3. Seleccionar una ubicación para el software y seguir las instrucciones en pantalla.

Una vez finalizado el proceso habrá un icono VirtualBox en la carpeta de aplicaciones del Finder. Entonces el usuario podrá utilizar en su sistema operativo OSX máquinas virtuales capaces de emular el funcionamiento de plataformas Linux, Windows o de cualquier otro tipo (Figura 2.7).

2.4 ANDROID SDK

Android SDK (*software development kit*) es el conjunto de herramientas de desarrollo que los programadores utilizan para crear e instalar aplicaciones Android, probar su funcionamiento en dispositivos virtuales, depurar código y otras tareas de desarrollo. El objetivo consiste en elaborar, de manera unificada y más o menos estándar, programas que habrán de ejecutarse sobre Android una vez descargados de Google Play u otros mercados de aplicaciones. El paquete completo SDK incluye librerías y API, documentación técnica, un emulador de terminales y diversas herramientas. SDK es multiplataforma: dispone de versiones para Windows, Linux y OSX, y puede descargarse desde la página web del proyecto SDK en *http://developer.android.com/sdk/index.html*. Aunque SDK fue creado pensando en los desarrolladores de software, también es, como vamos a ver, una herramienta imprescindible para el investigador forense. Disponer de una instalación plenamente funcional constituye requisito necesario para la adquisición y análisis de dispositivos móviles Android.

2.4.1 Niveles de API y partes sobrantes

La versión de SDK que el investigador necesita depende del sistema operativo que quiere analizar y de lo antiguo que sea el dispositivo investigado.

Lo que hay que tener en cuenta no es el número de la versión de Android, sino el nivel de API correspondiente. Por ejemplo, para la investigación de dispositivos Android 2.3 Gingerbread, de los cuales aún quedan muchos en circulación, se debe utilizar una versión de SDK con nivel de API 10 o superior. Para los terminales más modernos —con KitKat 4.4 y nivel de API 19— lo recomendable es disponer de una versión de SDK para plataformas de 64 bits. Con ella se podrán analizar dispositivos Android 4.X (Jelly Bean) y superiores, como por ejemplo terminales de las gamas Samsung Galaxy o Sony Xperia.

Los objetivos profesionales del investigador raramente coinciden con los del desarrollador de software. Es preciso tenerlo en cuenta porque la documentación técnica en la que se describe el proceso de instalación de SDK está dirigida a programadores y contiene instrucciones para instalar y configurar, además del SDK, un entorno de desarrollo completo —como por ejemplo Eclipse— con herramientas que facilitan la elaboración, compilación y depuración de programas Android. Estos entornos ayudan a escribir código fuente en Java, compilan el ejecutable *.dex* dejándolo listo para instalar en el teléfono e incluyen soporte para otros lenguajes de programación como C++, PHP o Perl. Las recomendaciones de la página de descarga pueden confundir al investigador, haciéndole pensar que Android SDK no podrá funcionar bien si no se instala también el entorno de desarrollo Eclipse.

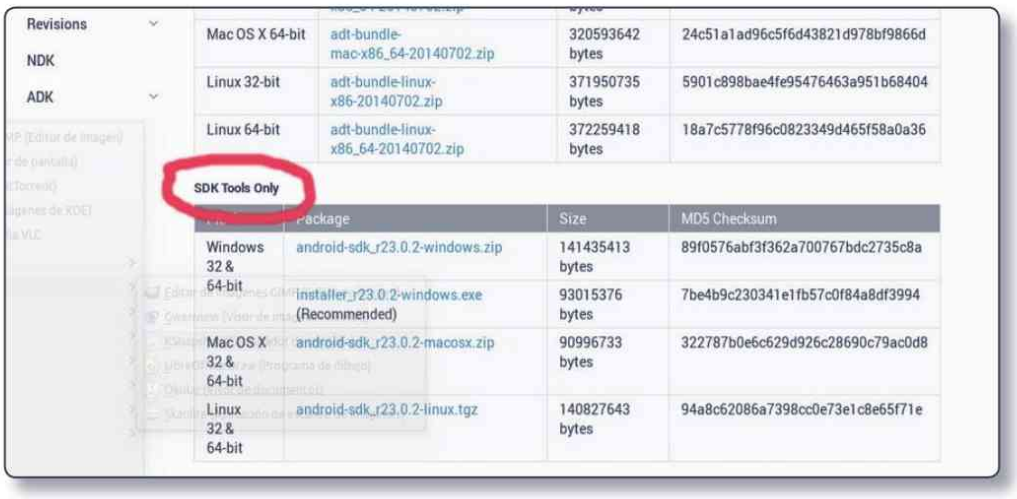

Figura 2.8. Recuerde que el entorno de desarrollo Eclipse no es imprescindible

Para cometidos forenses tales como la adquisición y el análisis de elementos de evidencia no es necesario tener Eclipse ni ningún otro entorno de desarrollo (Figura 2.8). El investigador no va a escribir ni depurar programas de ordenador, a no ser que se trate de un caso muy particular de análisis de aplicaciones de

software malicioso. Lo único que hará es conectar dispositivos a su estación forense, visualizar el contenido de aquellos, extraer archivos y llevar a cabo otras operaciones por el estilo. Si en la web de SDK selecciona el paquete básico **—SDK tools only—** correspondiente a su sistema operativo (Windows, OSX o Linux) ignorando el paquete que contiene Eclipse y las herramientas de desarrollo, podrá llevar a cabo la instalación de SDK sin ningún problema y dispondrá de todo lo necesario para trabajar. Además ahorrará medio gigabyte de espacio en el disco duro.

2.4.2 Instalación en Windows

La última versión de Android SDK en el momento de escribir estas líneas (r23.0.2) puede descargarse en dos formatos: archivo comprimido ZIP o instalador Windows. Esta última es la más recomendable. No es preciso elegir entre arquitecturas porque el paquete contiene código para ambas y es capaz de detectar automáticamente si el sistema es de tipo x86 o amd64. También comprueba si existen las necesarias dependencias Java, y de no ser así las descarga por su cuenta. En algunas plataformas es posible que no se descargue de manera automática la versión de 64 bits del JDK (entorno de desarrollo de Java). En tal caso el usuario tendrá que instalar el SDK manualmente. Con el archivo ZIP el procedimiento es similar, solo que antes habrá que crear una carpeta de instalación, a la que llamaremos `C:\android-sdk-windows`, para después extraer en ella el archivo comprimido.

Una vez hecho esto, el usuario debe hacer doble clic sobre la utilidad **SDK Manager.exe**. Si ha utilizado el instalador, también puede iniciar el programa desde el menú de inicio de Windows, los iconos del escritorio o la barra de acceso rápido. Esta acción hace que se abra la ventana del administrador SDK, en la que se le pedirá que especifique algunas opciones y seleccione paquetes para el proceso inicial de actualización, mediante el cual se bajan de Internet componentes adicionales necesarios para el funcionamiento de SDK y se actualizan otros en los que el desarrollador pudiera haber realizado correcciones de errores o cambios de cualquier otro tipo.

Si el investigador trabaja con Windows Vista o 7, lo más probable es que sufra un pequeño contratiempo. El programa arranca con normalidad pero no es capaz de instalar los paquetes bajados de Internet. El remedio es simple. Para que la actualización se lleve a cabo con éxito, el SDK Manager debe ejecutarse con privilegios de superusuario. En vez de hacer doble clic, el usuario tiene que abrir el menú contextual con el botón derecho del ratón y seleccionar la opción **Ejecutar como administrador** (Figura 2.9).

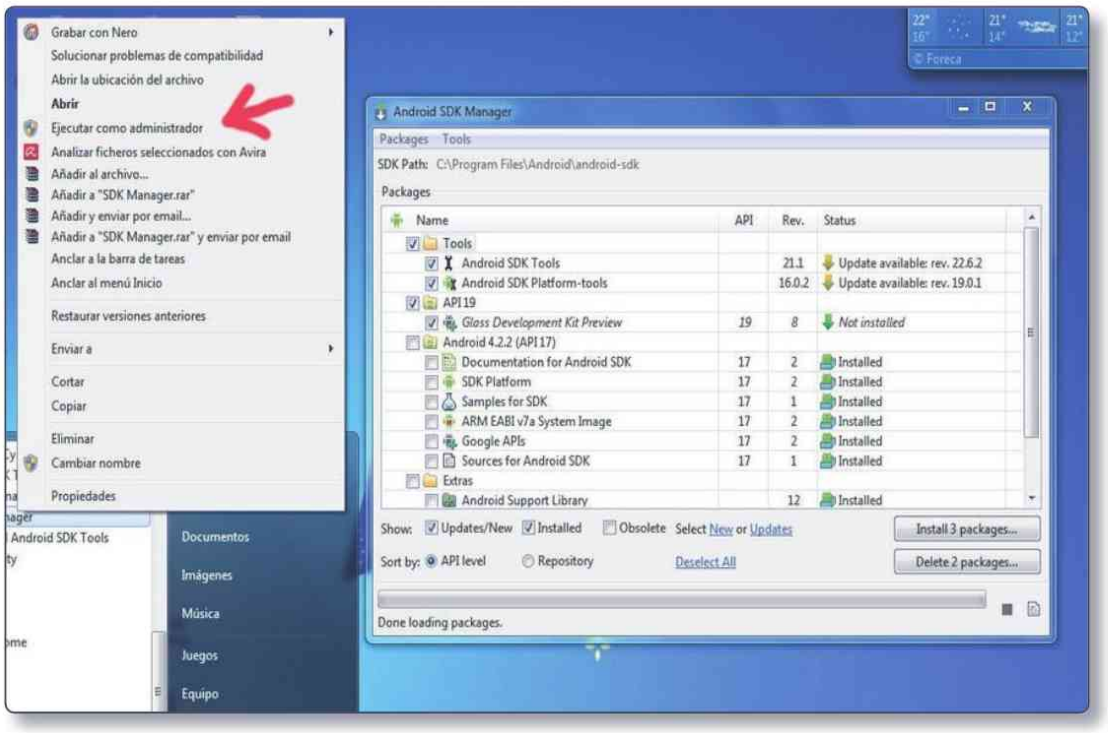

Figura 2.9. Android SDK Manager

El último paso para disponer de un Android SDK plenamente operativo consiste en instalar los controladores del dispositivo móvil que queremos investigar. Sin ellos, SDK solo podrá establecer la conexión USB con el terminal a un nivel básico —en modo almacenamiento, como si fuera un disco duro externo o un *pendrive*— y no será posible acceder a la partición de **Sistema Android** ni ejecutar las funciones especializadas de `adb`—. Estos *drivers* son específicos del modelo y la marca del terminal (Sony, HTC, etc.). Si no los encuentra en las actualizaciones facilitadas por Google a través del SDK Manager, deberá buscarlos en la página web del fabricante.

En el caso de los dispositivos móviles de la serie Samsung Galaxy, utilizados como ejemplo a lo largo de esta obra, el investigador puede conseguir el paquete de *drivers* descargándoselo de la página de soporte de Samsung o copiándolo desde el DVD de acompañamiento. También puede instalar el software de sincronización Samsung Kies, donde vienen incluidos por defecto. Esta última opción es la más aconsejable, ya que en caso de que los objetivos de la investigación solo especifiquen una exploración a bajo nivel, podrá utilizar Kies para adquisiciones lógicas del dispositivo (contactos, mensajes, imágenes, etc.) sin tener que recurrir a los procedimientos intrusivos que se llevan a cabo mediante SDK, y que serán objeto de explicación detallada en las páginas siguientes.

2.4.3 Instalación en Linux

Android SDK para Linux se encuentra disponible para descarga en *http://developer.android.com/sdk/index.html*, con las versiones correspondientes a otros sistemas operativos. Al igual que en el procedimiento descrito para Windows, lo que interesa aquí es disponer del paquete básico, que incluye código para arquitecturas i86 y amd64. Una vez obtenido, el investigador deberá ir con él a su directorio personal (`home/usuario`) y extraer el contenido mediante cualquier utilidad Linux, por ejemplo en línea de comando con `tar`:

```
tar xfzv android-sdk_r22.0.1-linux.tgz
```

La descompresión crea un árbol de directorios que contiene el SDK y sus componentes. Vaya a la carpeta donde se encuentran los ejecutables:

```
cd android-sdk-linux/tools
```

Desde aquí podrá arrancar directamente el SDK Manager y proceder a la actualización de componentes y la descarga de actualizaciones. También puede arrancar el gestor de dispositivos virtuales y llevar a cabo otras tareas características del programa:

```
./android
```

Marque en el menú los paquetes que desea actualizar. Si la investigación tiene que ver con dispositivos antiguos y necesita un nivel de API obsoleto, también puede seleccionar esta opción desde el Manager. La actualización no es simplemente una posibilidad: una vez instalado el entorno de desarrollo es preciso llevarla a cabo porque en versiones recientes de SDK las herramientas necesarias para establecer comunicación con los terminales Android se encuentran en un directorio llamado `platform-tools`. Si este directorio no existe, SDK Manager mostrará un mensaje de error al iniciarse. Dependiendo de la distribución Linux y de la configuración del entorno, es posible que también tenga que ejecutar SDK Manager con privilegios de administrador:

```
sudo ./android
```

Finalmente, si desea poder iniciar el SDK Manager desde otra carpeta —por ejemplo, un directorio especialmente creado para el caso que investiga— y no quiere trasladarse con `cd` hasta la ubicación de los binarios ni escribir largas rutas de archivo, puede modificar sus variables de entorno y hacer que su trabajo sea más cómodo durante el manejo de SDK. Para ello es necesario añadir las siguientes líneas al archivo *.bashrc* de su directorio de usuario:

```
export PATH=$PATH:/home/usuario/android-sdk-linux/tools/
export PATH=$PATH:/home/usuario/android-sdk-linux/platform-tools/
```

Puede servirse de un editor de texto cualquiera, como `kate`, `joe` o `nano`. Una vez hecho esto, lo único que tiene que hacer es guardar *.bashrc*, cerrar la consola de comandos y abrir otra nueva.

2.4.4 Instalación en OSX

Para instalar Android SDK en un entorno Apple OSX debe descargar el paquete correspondiente desde la misma página web donde se encuentran las versiones para Windows y Linux. Extraiga su contenido en OSX. Al igual que en el apartado anterior, vaya hasta la carpeta de los ejecutables e inicie SDK Manager. Una vez completada la actualización, el investigador dispone de las herramientas necesarias para conectar dispositivos móviles. Finalmente debe expandir el repositorio Android, seleccionar el directorio **`platform-tools`** y al menos una plataforma Android. La instalación se completa tras haber aceptado la licencia.

Finalmente, para hacer que las herramientas puedan iniciarse desde la carpeta Android SDK, es preciso poner los ejecutables en una ruta por defecto. Modifique el archivo `.bash_profile` de manera similar a como se ha hecho antes en Linux, añadiendo la siguiente línea mediante un editor de texto:

```
directory PATH=$PATH:/Users/yingyang/android-sdk-mac_86/platform-tools
```

2.4.5 Emulación de dispositivos Android

Una vez instalado Android SDK y actualizados los paquetes, el usuario podrá crear un AVD o dispositivo móvil virtual, capaz de ejecutarse en el ordenador en lugar de sobre una tableta o un *smartphone*. Los AVD (Android Virtual Device) son máquinas virtuales, verdaderas plataformas simuladas que a través de una capa de software interpuesta entre el sistema anfitrión y el software de usuario emulan dispositivos físicos provistos de sistemas operativos y aplicaciones Android. Estos dispositivos son genéricos y no reproducen las características propias de marcas y modelos concretos de ningún terminal. El emulador de Android es una herramienta muy útil para el desarrollador porque le permite comprobar el funcionamiento de sus programas en condiciones cercanas a la realidad sin tener que instalarlos sobre un terminal hardware.

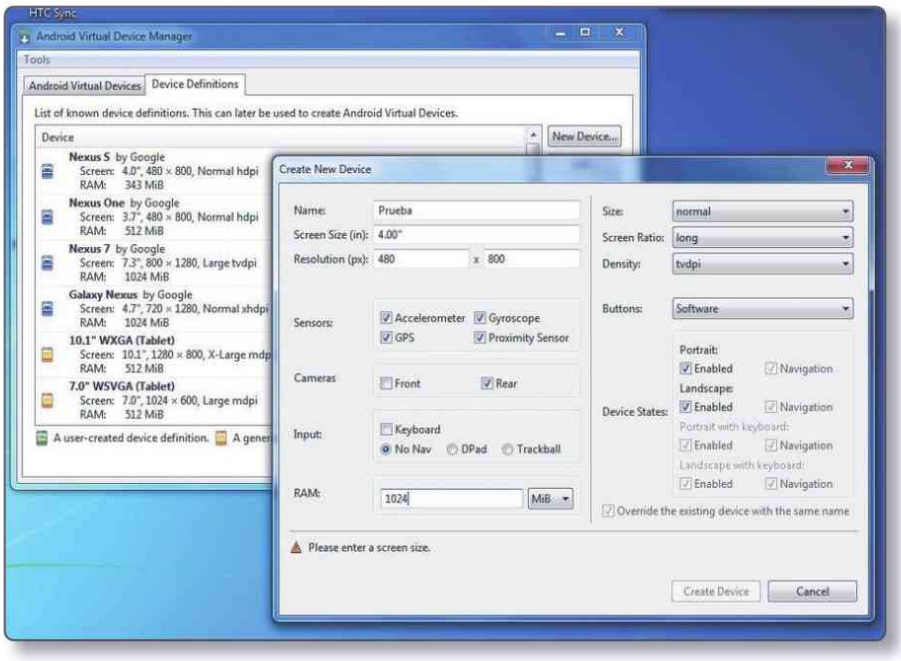

Figura 2.10. Emulación de dispositivos virtuales Android

Esto es algo de lo que el investigador también podrá sacar partido, por ejemplo cuando tenga que estudiar ataques de software malicioso o reconstruir el modus operandi de un sospechoso en casos de robo de datos, ciberacoso u otros delitos. El emulador permite analizar el funcionamiento de una aplicación y situar una hipótesis forense en un contexto de circunstancias plausibles. En otras palabras, se trata de hacer un uso creativo de herramientas diseñadas para cometidos de ingeniería y no para fines de investigación.

Para crear un dispositivo virtual, el investigador debe ir en primer lugar al menú **Herramientas** (**Tools**) del SDK Manager y abrir el **Administrador de dispositivos virtuales** (**AVD Manager**). Repase las definiciones de dispositivos (en la pestaña **Device definitions**) y decida cuál le conviene. El proceso es autoexplicativo (botón **New** a la derecha) y consta de los pasos siguientes: asignación de nombre al dispositivo, selección de marca y modelo, determinación del nivel de API con que quiere realizar las pruebas, establecimiento de parámetros básicos: teclado, hardware, cámaras trasera y/o frontal, memoria RAM, espacio de almacenamiento, tarjeta SD, etc. Finalmente, pulse **OK** (Figura 2.10).

En pocos segundos el dispositivo virtual estará configurado y el usuario podrá iniciarlo, detenerlo, guardar su estado de funcionamiento, borrarlo o trasladarlo a otro ordenador con solo copiar los archivos correspondientes. El dispositivo virtual posee las mismas funcionalidades que un terminal físico. Podemos instalar aplicaciones,

navegar por Internet, descargar correo electrónico, etc. Los datos generados por estos entornos virtuales pueden ser adquiridos y analizados de la misma manera que los de un *smartphone* o una tableta.

Para un investigador la característica más interesante del emulador es que no obliga a trabajar con entornos de última generación. Descargando componentes software antiguos resulta posible reconstruir dispositivos obsoletos que nos ayuden a esclarecer particularidades relativas al funcionamiento de una plataforma determinada, la ubicación de datos dentro del árbol de directorios y otras particularidades que en un momento dado pudieran ser de interés.

El emulador consume gran cantidad de recursos, algo a tener en cuenta sobre todo en entornos que su vez ya están virtualizados, como los que posiblemente esté utilizando el investigador tras la instalación de VMware o VirtualBox en su estación de trabajo forense. El código libre —Android no es una excepción— funciona bien en plataformas antiguas. No obstante, para analizar dispositivos móviles con los procedimientos que se han explicado anteriormente resulta aconsejable disponer de un equipo potente, con procesador Intel o AMD de 64 bits, sistema operativo Windows 7 o Linux y toda la RAM que uno se pueda permitir (en ningún caso menos de 4 GB). Aunque no tengamos previsto recurrir al emulador, nunca se sabe cuándo habrá que realizar operaciones intensivas en cálculo con herramientas de *data carving* y ruptura de contraseñas por fuerza bruta, simular una red local con varias máquinas virtuales VMware u otras operaciones intensivas en ciclos de CPU y consumo de RAM.

2.5 LINUX

Linux es importante para la investigación forense de dispositivos Android. En primer lugar, aunque las aplicaciones están escritas en Java y funcionan sobre un entorno virtualizado haciendo llamadas estándar a librerías y servicios a través de una interfaz de programación de aplicaciones (API), que en algunos aspectos recuerda a la estructura del entorno y los procedimientos de trabajo utilizados por los desarrolladores de aplicaciones Microsoft, todo el proyecto Android está basado en Linux: funciona sobre un *kernel* de Linux con controladores y módulos insertados en el núcleo del sistema; la estructura jerárquica de directorios es típica de los sistemas Linux/Unix, al igual que el modelo de seguridad POSIX, los sistemas de archivos soportados, la existencia de permisos y un esquema de gestión de usuarios y grupos característico de los entornos Linux/Unix. Algunas de las herramientas que va a utilizar el investigador para la adquisición y análisis de evidencias son también de tipo Linux/Unix.

No entraremos en la historia de Linux; tampoco en la filosofía del software libre, sus diferencias con respecto al código propietario y otros temas relacionados. Para los fines a los que esta obra aspira, interesa sobre todo que el investigador se familiarice con comandos imprescindibles que permiten ejecutar tareas en Linux y otros sistemas operativos desarrollados a partir de Unix.

2.5.1 Listado de directorios con ls

¿Cuál es el primer comando que aprende todo usuario al iniciarse en el manejo de la consola de texto Linux? Si tiene edad para haber conocido los tiempos pioneros del vetusto MS-DOS, un reflejo instintivo le llevará a averiguar qué hay en el directorio actual tecleando `dir` y pulsando a continuación la tecla **Enter**. Sorprendentemente, esto también funciona en Linux. El resultado es una lista de archivos con sus nombres, tamaño en bytes, fecha y hora de la última modificación y a la izquierda del todo unas letras extrañas, que no aparecían en los listados de MS-DOS, y cuyo significado desconoce porque aún no está familiarizado con nociones como grupos, permisos y dispositivos de bloque.

Si lo desea puede arreglárselas con esto. Pero en el mundo Unix existe una forma más ortodoxa de hacer las cosas. El comando que nos permite listar el contenido de los directorios es `ls`. Ejecutado sin opciones proporciona una lista simple de archivos y directorios. Acompañado de parámetros (`-l`, `-a`, `--color`, etc.) podremos ver el contenido del directorio —archivos y otros directorios— en forma tabulada, con permisos de ejecución, nombre del propietario del archivo, tamaño, marcas de tiempo e incluso un color característico para cada tipo de archivo).

2.5.2 Ayuda mediante man

Por ejemplo, para saber más de las opciones de `ls`, el usuario puede llamar a la página de ayuda tecleando:

```
man ls
```

El significado de `man` es "manual". Con este comando podrá llamar a las páginas de ayuda de la mayor parte de los comandos Linux, obteniendo información relativa a sintaxis, parámetros, opciones, reglas de empleo y ejemplos de uso.

2.5.3 mkdir y rmdir

Hoy día casi todo el mundo crea las carpetas por medio de un navegador de archivos. Para hacerlo en línea de comando se emplea `mkdir`:

```
mkdir trabajo
```

Y para suprimirla, `rmdir`:

```
rmdir trabajo
```

Antiguamente, a las carpetas se las llamaba directorios. En Linux aún se emplea este término por varias razones: por un lado existe cierta reticencia a dejarse influenciar por conceptos Microsoft. Quien se formó con la línea de comando posee una visión de las cosas secuencial y analítica, muy diferente a la perspectiva intuitiva y visual de los que han aprendido a manejar un ordenador en la época de los entornos gráficos de finales del siglo xx y comienzos del presente. Por otra parte, la línea de comando sigue siendo muy utilizada por los usuarios de código libre y constituye una asignatura obligatoria dentro de la escena Linux. En las páginas que siguen, y cada vez que hagamos referencia a conceptos de Linux, en lugar de hablar de carpetas nos serviremos del término "directorio".

El comando `rmdir` solo elimina directorios vacíos. Si en el directorio que queremos borrar hay archivos, aunque sea ocultos, el sistema responde con un mensaje de error.

2.5.4 cd

Para trasladarnos de unos directorios a otros nos servimos de `cd`. En Linux el carácter ~ representa el directorio `home` o propio del usuario actual. Con `cd` podremos navegar en sentido ascendente o descendente a otros directorios del árbol Linux, y una vez allí, dependiendo de los permisos que el usuario tenga, examinar el contenido del directorio, abrir sus archivos, modificarlos, copiarlos, borrarlos, crear y borrar nuevos directorios, etc.

En todo directorio nos encontraremos con dos entradas de archivo especiales: el punto (.) y el doble punto (..). El punto (.) representa el directorio actual, es decir, aquel en el que nos encontramos, mientras que los dos puntos (..) son el símbolo del directorio de nivel inmediatamente superior. Esta convención también se daba en MS-DOS, y continúa existiendo en la interfaz de comandos de Windows Vista/7/8. Por ejemplo, si tecleamos…

```
cd .
```

… nos quedamos donde estamos. Esta orden, literalmente, quiere decir: "desplazarse al directorio actual". Es como un salto hacia arriba para caer en el mismo sitio. Por el contrario, el comando…

```
cd ..
```

… permite al usuario trasladarse al directorio de nivel inmediatamente superior. Como es lógico, cuando estemos en el directorio raíz del sistema de archivos no podremos utilizar los dos puntos. Ni siquiera los podremos ver al listar el contenido con `ls`, ya que el directorio raíz (/) es el punto más alto de la jerarquía y no cuelga de ningún otro directorio.

2.5.5 less

El comando `less` muestra en pantalla el contenido de un archivo, permitiendo al usuario desplazarse arriba y abajo mediante las teclas ↑ [flecha arriba] y ↓ [flecha abajo]. Su utilidad es limitada a la hora de examinar código ejecutable o formatos binarios como los de las bases de datos o documentos Microsoft, pero resulta de gran ayuda para visualizar archivos de texto o XML, habituales en Internet y el mundo del Android.

2.5.6 grep

El comando `grep` permite realizar búsquedas de caracteres en el interior de un archivo, en un directorio determinado, un árbol de directorios o incluso en todo el sistema de archivos. Una particularidad de `grep` es que puede distinguir entre mayúsculas y minúsculas. Si el usuario no está seguro de cómo se escribe exactamente la palabra que busca, puede indicar a `grep` que ignore la distinción entre mayúsculas y minúsculas, empleando para ello la opción `"-i"`. He aquí algunos ejemplos de uso de `grep`:

```
grep password dumpfile_12-07-2013.pcap
```

Este comando intenta localizar contraseñas en un archivo de paquetes capturados en la red local mediante utilidades como `tcpdump` o Wireshark. La salida del comando se muestra inmediatemente después de la línea de ejecución. Con esta otra variante…

```
grep -i password dumpfile_12-07-2013.pcap
```

… se consigue lo mismo, con la certeza de que si la palabra "password" comienza o está escrita con mayúsculas no podrá escapar a la detección.

Las comillas, al igual que en los buscadores de Internet, acotan el término buscado con el objeto de localizar una expresión literal en la que distinguimos entre mayúsculas y minúsculas de una manera explícita:

```
grep “List of passwords” dumpfile_12-07-2013.pcap
```

2.5.7 Canalización de comandos

La canalización o *pipelining* es una técnica originaria de los primeros entornos Unix. Toma el resultado de un comando y lo introduce en otro, pudiendo hacerse esto varias veces, redirigiendo la salida hacia etapas sucesivas de proceso que refinan el tratamiento de la información. De este modo se consiguen efectos de filtrado que pueden ayudar al investigador en su búsqueda. El operador de canalización es el signo de teclado barra vertical (|).

Veamos un ejemplo: nos interesa saber en qué líneas de un archivo de gran tamaño, como el volcado de paquetes de red procedente del ejemplo anterior, figuran unos términos de búsqueda concretos. Para ello podemos combinar los comandos `less` y `grep`:

```
less dumpfile_12-07-2013.pcap | grep POP3
```

El *pipelining* es posible siempre que la salida de un comando pueda ser introducida como *input* para el siguiente.

2.5.8 Redireccionamiento

Mediante redireccionamiento se consigue que el resultado de un comando no sea dirigido a la salida estándar, es decir, mostrado en pantalla, sino canalizado hacia otro destino, como, por ejemplo, un archivo de texto para fines posteriores de análisis y documentación. Una forma habitual de redireccionar la salida de un comando es mediante el signo > [mayor que]; por ejemplo, de este modo:

```
dmesg | grep –i USB > info_hardware.txt
```

Este comando, en el que también se hace uso de concatenación, explora el volcado de depuración del *kernel* Linux en busca de líneas que contengan información relativa a dispositivos USB, aunque los mismos figuren en minúsculas. A continuación el resultado es redirigido a un archivo de texto.

Supongamos que nos interesa cierta información relacionada con dispositivos Bluetooth, pero no queremos que aparezca recogida en un segundo archivo, sino

añadir la salida al archivo anterior, sin borrar nada de lo escrito en él. Esto se consigue con los operadores >>, que conducen la información hacia un archivo de destino (en este caso de texto) añadiéndola a la ya existente:

```
dmesg | grep –i bluetooth >> info_hardware.txt
```

2.5.9 find

El comando `find` se utiliza para localizar archivos dentro de un árbol de directorios. Si hemos perdido algo lo podremos encontrar de modo rápido y simple de la siguiente manera:

```
find –name sancion.pdf
./Wardrivin/Capturas/sancion.pdf
```

La búsqueda realizada por el comando `find` no se limita a nombres de archivos. Gracias a un amplio número de operadores y opciones también admite posibilidades más sofisticadas, incluyendo concatenación con otros comandos, *pipelining* y redireccionamiento. Por ejemplo:

```
find -name *.sqlite -exec sha256sum  {} \; > sqlitefiles.txt
```

Este comando, ejecutado desde el directorio `home` del usuario, localiza todos los archivos bases de datos sqlite existentes en su árbol de carpetas, calcula el *hash* SHA 256 de cada uno de ellos y almacena el resultado en un archivo de texto.

2.5.10 sudo

En Linux el usuario trabaja normalmente con permisos restringidos que no le permiten ejecutar operaciones reservadas al administrador. Cuando sea necesario acceder a recursos para los cuales se requiere una autorización de nivel más alto —cambios en los archivos de configuración del sistema, utilización de programas de copia a bajo nivel como `dd`, instalación de paquetes de software, etc.— existen dos posibilidades: reentrada desde la cuenta de superusuario o ejecución de comandos con el prefijo `sudo`.

En el primer caso estaremos cambiando nuestro contexto de uso del sistema. Seremos el administrador, con nuestro propio directorio base —en este caso `/root`— y un conjunto de variables de entorno diferentes a las que son asignadas al resto de los usuarios. Tendremos poder para hacer lo que queramos con cualquier recurso del sistema. Por el contrario, ejecutando comandos con `su`, seguiremos

siendo usuarios normales, solo que provistos de una autorización provisional para llevar a cabo tareas con un nivel de privilegios más elevado.

El acceso como administrador o `root` aumenta nuestras atribuciones de poder de modo considerable, y con ello nuestra capacidad para producir daños en el sistema, por lo que en todo momento es preciso andarse con cuidado y pensar las cosas dos veces antes de teclear nada.

En ambos casos el sistema solicitará la contraseña del administrador:

```
sudo apt-get install scalpel
[sudo] password for igandekoa:
```

2.5.11 apt-get

En distribuciones Linux basadas en Debian —Ubuntu, Backtrack y otras— la gestión de paquetes de software se lleva a cabo a través de utilidades que permiten llevar a cabo todo el mantenimiento del sistema a través de la línea de comando: descargar paquetes desde los repositorios, instalarlos en el disco duro, resolver dependencias con librerías y otros programas, configurar el software, actualizarlo, repararlo y desinstalarlo cuando ya no hace falta.

Una de estas utilidades es `apt-get` (Advanced Packaging Tool), cuyas opciones de uso trascienden el ámbito explicativo de este libro. Tan solo hablaremos de su empleo más frecuente a la hora de instalar software en el sistema:

```
sudo apt-get install sleuthkit
[sudo] password for igandekoa:
.....
```

La gestión de software a través de estos comandos, y otros como `dpkg` o `aptitude`, es por lo general automática. No solamente se descargarán del repositorio y se instalarán los paquetes solicitados, sino también aquellos que sean necesarios para resolver sus dependencias.[5] El usuario no tiene más que pulsar una o dos veces la tecla de retorno para confirmar la asignación de espacio en disco duro a los nuevos programas y autorizar la instalación.

---

5 Una dependencia, sobre todo en entornos Linux/Unix, tiene lugar cuando un programa necesita para su funcionamiento que haya otros programas instalados, por verse obligado en algún momento a acceder a recursos, librerías o datos pertenecientes a aquellos.

2.5.12 cp y mv

Copiar y mover archivos en Linux es una operación que se lleva a cabo de modo similar a como se hacía bajo el antiguo MS-DOS. De hecho, el comando `copy` fue en sus orígenes una adaptación de su homólogo `cp` en Unix. La sintaxis es también muy parecida:

```
cp ./hashes_MD5.txt /media/LLAVEUSB/Evidencia
```

Con esta orden copiamos un archivo de texto desde el directorio actual a una carpeta perteneciente a una unidad USB montada en el directorio `/media`. Si en lugar de utilizar `cp` nos servimos de `mv`, estaremos trasladando el archivo, y el original en el directorio actual será borrado después de la copia.

Con `cp` también es posible cambiar de nombre al archivo durante el proceso de copia. Para ello basta asignar un nombre diferente en la carpeta de destino.

```
cp ./hashes_MD5.txt /media/LLAVEUSB/Evidencia/sumas_verificacion.txt
```

A diferencia del antiguo MS-DOS y la interfaz de texto de Windows, en Linux/Unix no existe un comando específico para renombrar archivos, ya que `mv` cumple esa función cuando se le utiliza sobre un archivo sin especificar otro directorio de destino:

```
mv hashes_MD5.txt sumas_verificacion.txt
```

2.5.13 chmod y chown

El comando `chmod` (*change mode*) se utiliza para modificar permisos de archivos y carpetas, algo importante cuando el usuario necesita acceder a ellos, modificarlos o ejecutarlos si se trata de programas o *scripts*. Este comando, al igual que `chown`, debe ser utilizado sobre un archivo que se encuentre en el directorio actual. En caso contrario habrá de indicarse la ruta completa:

```
chmod 755 hashes_MD5.txt
chmod 755 /home/igandekoa/evidencia/hashes_MD5.txt
```

Esta orden concede permisos de lectura, escritura y ejecución al propietario del archivo, y solo de lectura y ejecución a los restantes miembros del grupo y a otros usuarios.

La posibilidad de gestionar los permisos de un archivo depende de quién sea su propietario. Si el archivo pertenece al administrador y un usuario sin privilegios quiere tener sobre él un permiso del que carece, el sistema responde con un mensaje

de error. En ocasiones resulta más práctico —siempre que se tengan privilegios suficientes para ello— cambiar el propietario del archivo en lugar de los permisos. Con esto el nuevo propietario podrá hacer con el archivo lo que quiera dentro del contexto de privilegios que le haya sido asignado por el administrador.

Para cambiar el propietario de un archivo se utiliza el comando `chown` (*change owner*). Por ejemplo, si el archivo de texto de los ejemplos anteriores ha sido copiado desde una llave USB que en lugar de estar formateada con una partición FAT32 tiene un sistema de archivos NTFS o ext3, la copia del archivo pertenece al administrador. Este, para permitir el acceso a usuarios normales, cambia la propiedad del archivo mediante `chown`. También puede hacerlo el propio usuario con la ayuda de `sudo` si conoce la contraseña del administrador:

```
sudo chown igandekoa hashes_MD5.txt
```

2.5.14 file

¿Cómo sabemos que un archivo es lo que parece y no otra cosa, por ejemplo información confidencial o software malicioso, disfrazado bajo la apariencia de algo normal como un documento de trabajo o una fotografía digital? En Windows, donde la norma es creer lo que dice la extensión del archivo, resulta complicado, a no ser que hagamos doble clic sobre él —opción poco aconsejable si se sospecha que puede tratarse de código malicioso—, o utilicemos un editor hexadecimal o herramientas forenses como EnCase o FTK.

En Linux resulta posible identificar un archivo mediante el comando `file`. Por lo general cada archivo suele llevar en su parte inicial una secuencia de caracteres o firma distintiva que indica si se trata de un documento de Word, un archivo gráfico JPG, código ejecutable o datos en bruto para ser utilizados por un programa determinado. Por ejemplo, el archivo que se muestra en la figura 2.11 parece a simple vista un inofensivo audiovisual. Hemos intentado ejecutarlo con un reproductor de medios —por ejemplo, Windows Media o VLC— y no funciona. Quizás está corrupto o no ha sido extraído de forma correcta por nuestro software de recuperación de archivos. Ante la duda exigimos que nos muestre sus credenciales a través de `file`:

```
file singinintherain.mp4
singinintherain.mp4: Zip archive data, at least v2.0 to extract
```

Al final nuestro archivo resulta ser un contenedor de datos comprimidos. No sabemos lo que hay dentro, pero si alguien se tomó la molestia de camuflarlo bajo la apariencia de una canción MP3, quizás hubiera una razón para ello.

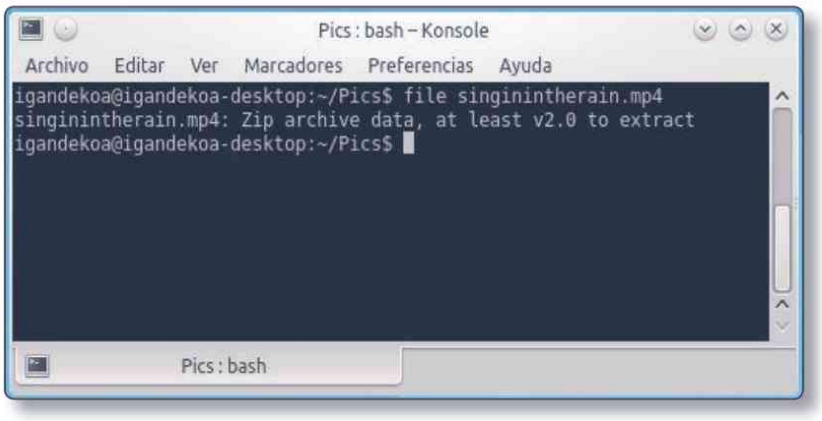

Figura 2.11. Al parecer, alguien nos la quiere pegar cambiando la extensión a un archivo de datos comprimido

El funcionamiento de `file` es simple: cuando se le ordena examinar un archivo lo que hace es comparar sus caracteres iniciales con las entradas existentes en un archivo de firmas ubicado en `/usr/share/misc/magic`. Si los encuentra, mostrará la línea de texto que incluye la información correspondiente al tipo de archivo, independientemente de la extensión que pueda llevar. El archivo `Magic` es ampliable: el usuario podrá añadir líneas correspondientes a firmas características (también llamadas "números mágicos": de ahí el nombre Magic) de archivos desconocidos o de nuevos tipos con los que pudiera ir encontrándose en el transcurso de su investigación.

3

TÉCNICAS DE INVESTIGACIÓN FORENSE

Hace unos pocos años la labor del informático forense era relativamente simple. Bastaba seguir procedimientos establecidos: extracción del disco duro de un PC de sobremesa o portátil, copia a bajo nivel con bloqueador de escritura, análisis de los datos con herramientas forenses, recuperación de archivos borrados y redacción del informe. Todos los elementos de evidencia se hallaban contenidos en equipos estándar y en ubicaciones predecibles: herramientas ofimáticas, documentos electrónicos, archivos de contabilidad y otros elementos orientados a la actividad empresarial o administrativa. La incautación de ese material no suponía excesivas complicaciones de trámite o interpretación jurídica, puesto que se trataba de un todo coherente en cuanto a funcionalidad y vinculación con objetivos profesionales y poco significativos para el ámbito de los derechos personales y la privacidad. El riesgo de que se produjeran alteraciones en los elementos de evidencia parecía estar bajo control.

3.1 DESAFÍOS DE LA MOVILIDAD INFORMÁTICA

El paso del módem telefónico a la banda ancha y la llegada de los dispositivos móviles cambian la situación. El ordenador ya no es, ni siquiera en los entornos empresariales y administrativos, tan solo una herramienta de trabajo, sino una interfaz personal que el usuario lleva consigo para hacer todo tipo de cosas relacionadas con su vida laboral y particular: gestionar correo electrónico, acceder a Internet, administrar sus cuentas bancarias, descargar archivos y mandar mensajes SMS o WhatsApp. Todo esto dificulta de manera notable el trabajo de los investigadores forenses.

Figura 3.1. Difícil panorama

3.1.1 Privacidad

Adquirir un soporte de datos para llevar a cabo una extracción masiva de elementos de evidencia a través de procedimientos normalizados y desprovistos de riesgo jurídico ya no es una opción practicable, por más que existan medios para mantener la cadena de custodia. En primer lugar existe el peligro de que nuestras actuaciones se consideren lesivas para la intimidad del acusado. Las empresas tampoco escapan a este riesgo, sobre todo cuando no existen sistemas de seguridad que protejan la red contra intrusiones, o al menos una política de uso de dispositivos digitales que conste por escrito y sea suficientemente conocida por el personal.

El mercado de la movilidad, con su amplia oferta de ordenadores portátiles, *smartphones*, reproductores multimedia y dispositivos de bolsillo de todo tipo, complica aún más el panorama. En cualquier entorno de la vida moderna —profesional o de ocio— el teléfono móvil ofrece la posibilidad de mantener al usuario conectado permanentemente a Internet y las redes de las empresas u organizaciones en las que trabaja. El hardware del dispositivo móvil codifica y almacena una parte significativa de la vida personal de su propietario: contactos telefónicos, historiales de llamadas y navegación por Internet, mensajes SMS, fotografías y mucho más.

Examinar el *smartphone* de una persona es algo parecido a practicar un registro corporal. No extraña que juristas constitucionales, abogados, expertos en seguridad y plataformas cívicas observen con tanta minuciosidad la metodología aplicada por los investigadores forenses en el ámbito de la informática móvil.

3.1.2 Estrategia gradual

Ya no es posible aplicar soluciones de manual basadas en listas de chequeo. El investigador debe aplicar una estrategia eficaz y tener claros los límites de su intervención. No es un simple experto que recurre a su recetario para extraer conclusiones de una manera mecánica: antes bien, debe explicar qué hace, por qué y con qué derecho. Su trabajo se basa no solo en la técnica, sino en presupuestos jurídicos firmes, puesto que ha de desenvolverse dentro de los límites establecidos por una orden judicial que le faculta a examinar unas cosas sin poder tocar otras. Por si lo anterior no fuese poco, en la práctica resulta imposible investigar un dispositivo móvil sin alterar de algún modo sus contenidos.

Por consiguiente se impone la necesidad de aplicar un enfoque gradual, yendo desde lo simple a lo complejo y desde lo externo a lo interno. Una vez incautado el dispositivo móvil lo primero que ha de hacerse es examinarlo de manera no intrusiva, como hace un usuario normal cuando busca números de teléfono o consulta su correo electrónico. Cada paso ha de quedar documentado por medio de notas o fotografías digitales sacadas a la pantalla del dispositivo.

Únicamente después de haber agotado esta vía sin resultados prácticos, se puede considerar la utilización de técnicas más intrusivas, como por ejemplo el examen del sistema de archivos, la realización de imágenes a bajo nivel y la recuperación de archivos borrados.

La última etapa del proceso, cuando otros procedimientos no hubiesen permitido alcanzar los objetivos de la investigación, implica el uso de técnicas sofisticadas que podrían llegar a requerir el desmantelamiento del terminal y la lectura directa de sus chips de memoria. Esto, por la capacitación que requiere y el elevado coste de los equipos de hardware necesarios para la operación, resulta viable tan solo en casos delictivos graves o cuando hay en juego un valor económico considerable.

Figura 3.2. Lápiz capacitivo para pantallas táctiles

3.2 PREVISUALIZACIÓN

Una vez que el terminal llega a nuestras manos lo primero que debemos hacer es llevar a cabo un proceso de previsualización. El objetivo no consiste en localizar elementos de evidencia, sino en verificar los ajustes y el estado inicial del *smartphone*. En un capítulo posterior se hablará de las dificultades prácticas del proceso de adquisición —aislamiento de la red, bloqueos de código y patrones, etc.—. Por el momento basta decir que este proceso de examen preliminar dista de ser trivial. La manipulación del dispositivo debe hacerse con precaución y economía de movimientos para no alterar, o al menos hacerlo en la menor medida posible, los elementos de evidencia. Se aconseja utilizar un *pen stylus* o lápiz capacitivo con punta de goma (Figura 3.2) en lugar de los dedos. De este modo se evita que queden huellas dactilares sobre la pantalla.

El investigador —o el encargado de manipular el dispositivo durante la intervención— debería estar familiarizado con los manuales de instrucciones de algunos de los dispositivos más populares del mercado. Se aconseja haber hecho prácticas con algún terminal de su propiedad. Tampoco estaría de más hacerse una pequeña colección de dichos manuales en PDF y llevarlos en un *tablet* para poder consultarlos cuando sea necesario. En Internet se pueden encontrar libros de instrucciones para gran cantidad de marcas y modelos. Adquirir la experiencia de un usuario normal, además de los conocimientos técnicos en profundidad del perito forense, ayudará a ejecutar con precisión y rapidez las operaciones necesarias para una correcta previsualización de los terminales.

Figura 3.3. Aplicaciones telefónicas

3.2.1 Teléfono

Aunque un *smartphone* tiene más de ordenador que de teléfono, muchos usuarios adquieren su primer terminal Android con la intención de reemplazar su antiguo móvil por otro nuevo, y en la mayor parte de los casos las llamadas telefónicas siguen consituyendo el uso principal del dispositivo. Por ello la labor de previsualización —también denominada *triage*— debe dirigirse en un primer momento a aplicaciones relacionadas con la telefonía.

Si el terminal no está bloqueado por código o patrón de seguridad —más adelante se habla de los problemas que esto plantea—, no costará llegar hasta ellas. Se trata principalmente de dos aplicaciones de sistema llamadas **Teléfono** y **Contactos**, situadas en la parte inferior izquierda de la pantalla táctil (Figura 3.3).

Dentro de este grupo de aplicaciones el investigador hallará un menú de cinta (parte superior de la pantalla) con pestañas correspondientes a diversas opciones. Interesan sobre todo la de **Registro** o **Historial de llamadas entrantes** (marcadas con una flecha verde dirigida a la izquierda), **salientes** (flecha roja señalando hacia la derecha) y **perdidas** (círculo azul con un trazo).

Así mismo tienen interés las dos pestañas siguientes: **Contactos**, con información sobre personas, números de teléfono, organizaciones y empresas, correos electrónicos, direcciones postales, etc.; y **Favoritos**, que incluye los números marcados con mayor frecuencia (Figura 3.4). Finalmente se encuentra la pestaña **Grupos**, que permite clasificar los contactos de manera automática en función de los datos introducidos en los diferentes apartados de la lista.

Figura 3.4. Menú de cinta de aplicaciones telefónicas

El icono de **Contactos**, situado a la derecha de la aplicación **Teléfono** en la pantalla táctil del *smartphone*, no es más que un enlace al menú de cinta descrito anteriormente.

3.2.2 Mensajes

La aplicación **Mensajes** incluye los SMS enviados y recibidos, agrupados en conversaciones y con una interfaz simple para redacción, envío e inclusión de adjuntos multimedia (fotos, vídeos, audios, notas).

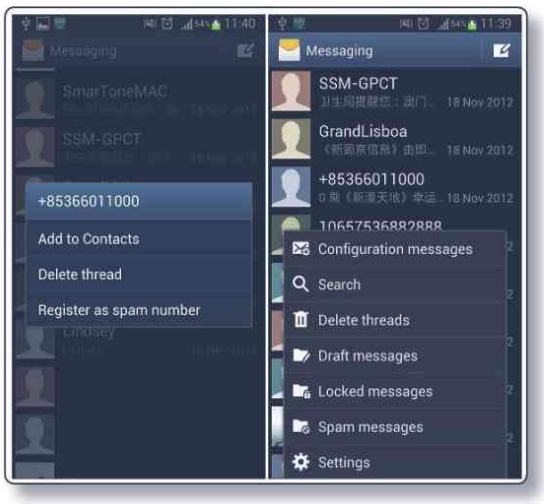

Figura 3.5. Mensajes SMS

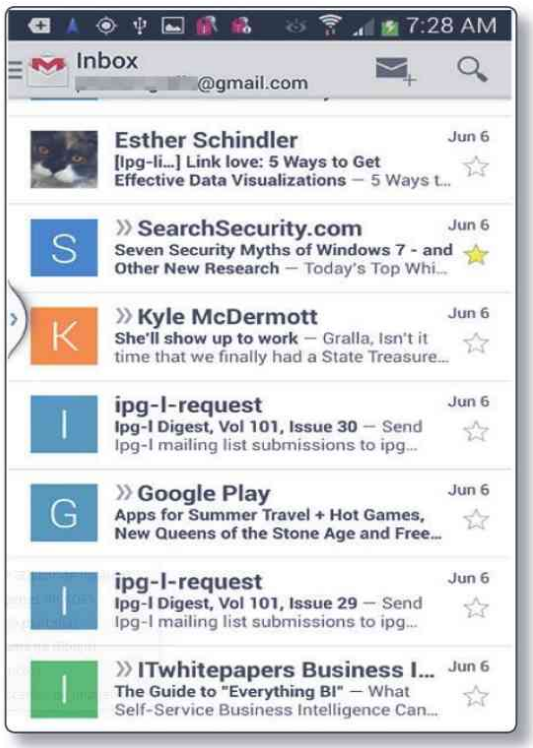

Figura 3.6. Interfaz de Gmail en un Samsung Galaxy S3

3.2.3 Correo electrónico

Como es de esperar en un desarrollo impulsado por Google, Gmail es una aplicación de sistema y viene preinstalada en todos los teléfonos móviles Android (Figura 3.6). Es ahí donde el investigador encontrará el correo del usuario, sin que pueda excluirse la existencia de otros clientes de correo electrónico. Cuando el dispositivo está conectado a Internet, Gmail accede al servicio de correo web de Google en tiempo real, de modo que antes de pulsar el icono conviene pensarlo dos veces para no alterar elementos de evidencia en el dispositivo o en el servidor de Google —por ejemplo, mediante la actualización automática de los contenidos del buzón entrante o el marcado de mensajes con una etiqueta de ya leído—. También es importante mantener a salvo la privacidad del sospechoso en asuntos no contemplados por la orden judicial o que no tengan que ver con los objetivos de investigación para el caso en curso.

3.2.4 Fecha y hora

La aplicación **Reloj** no solamente indica la fecha y la hora del sistema; también contiene otros datos cronológicos que pueden resultar de interés: alarmas, despertador y husos horarios de otras ciudades.

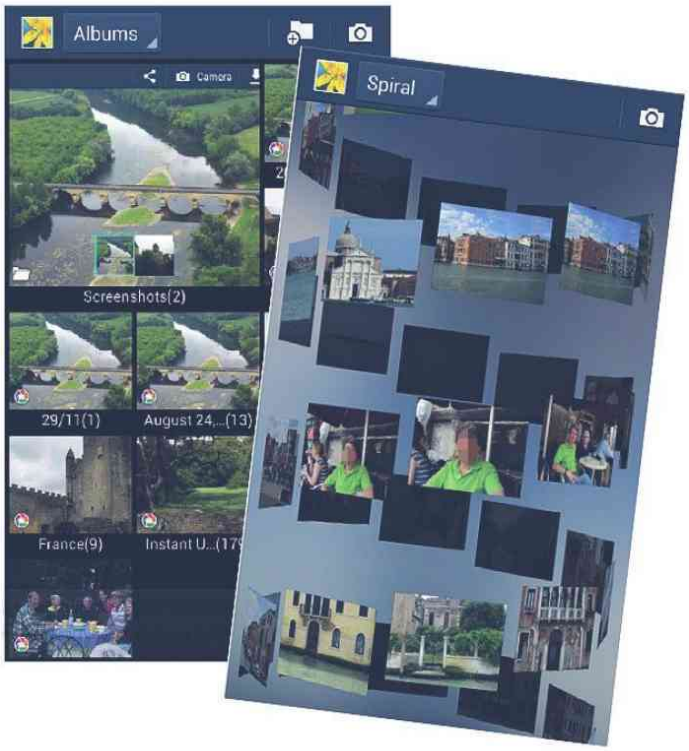

Figura 3.7. La galería de medios de Android es altamente configurable

3.2.5 Imágenes y vídeos

Están incluidos en la aplicación **Galería** (Figura 3.7), donde se podrán encontrar no solo tomas fijas y de vídeo realizadas con la cámara del terminal, sino también *podcasts*, películas y otros materiales descargados de la Red. Los metadatos incluidos en las fotografías —fechas, horas y coordenadas geográficas— constituyen uno de los elementos de evidencia más valiosos en la forénsica de dispositivos móviles Android.

3.2.6 Mapas

La aplicación **Mapas** (Android Google Maps) es importante porque proporciona datos para elaborar la línea de tiempo del caso (Figura 3.8). Estos datos hacen posible la trazabilidad de personas y objetos y pueden cruzarse con información procedente de otras aplicaciones —cámara fotográfica, navegador de Internet— y fuentes —antenas de telefonía móvil y puntos de acceso wifi—.

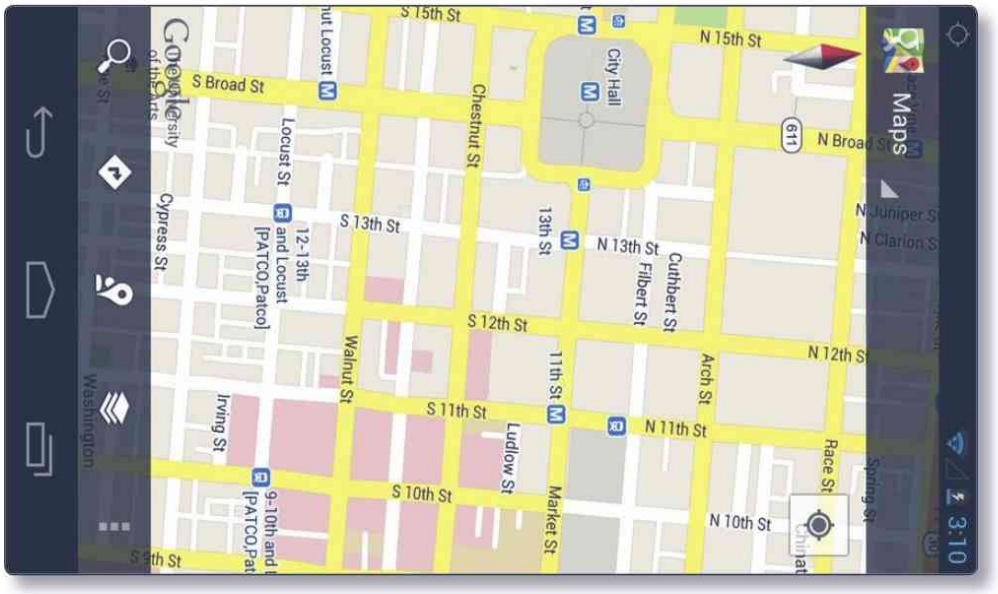

Figura 3.8. Android Google Maps

3.2.7 Archivos

Android dispone de un navegador de archivos con el que el investigador puede examinar el contenido de aquellas partes del terminal que son accesibles al usuario, y que este utiliza para administrar su información: documentos, imágenes, multimedia, aplicaciones descargadas de la Red, etc. Con este navegador podemos

movernos entre carpetas de un modo similar a como lo haríamos en un ordenador con OSX o Windows, accediendo a la tarjeta de expansión MicroSD (carpeta `/sdcard/external_sd`) y al área de almacenamiento interna (`/sdcard/DCIM`) donde se encuentran las fotografías sacadas con la cámara del dispositivo (Figura 3.9).

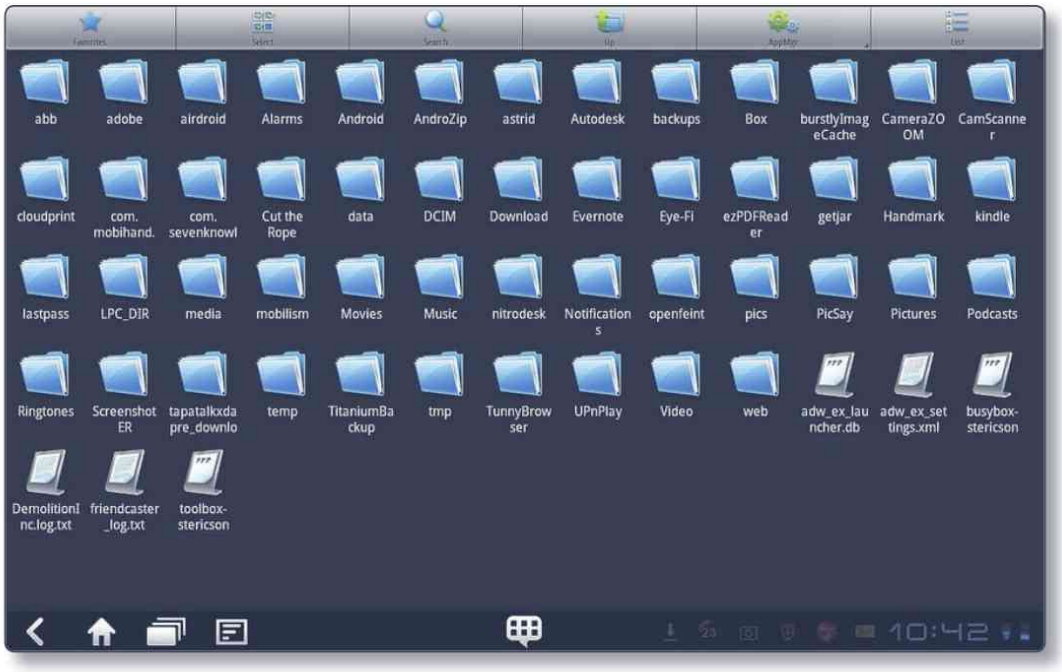

Figura 3.9. Navegador de archivos en una tableta Android

Hasta el momento, el uso de la pantalla táctil nos ha permitido obtener una imagen superficial del dispositivo. Para llegar a las partes no accesibles al usuario y a las zonas ocultas del sistema es necesario utilizar herramientas especiales y, si fuera necesario, realizar cambios en la configuración del dispositivo. Más adelante se verá que la exploración profunda de los contenidos de un *smartphone* no se lleva a cabo mediante el explorador de archivos, sino en línea de comando, obteniendo listados de directorios y copiando los elementos a la estación de trabajo forense.

3.2.8 Aplicaciones

Hay dos tipos de aplicaciones: las que vienen preinstaladas en el sistema, algunas de las cuales se han visto en el apartado anterior, y son necesarias para el funcionamiento del sistema; y las que el usuario descarga desde el mercado oficial de Android mediante Google Play (Figura 3.10) o desde diversos sitios de Internet. A estas últimas podemos añadir otras que el usuario quiera instalar directamente desde un ordenador. El examen de las aplicaciones permite conocer el perfil de actividad

del usuario y hacerse una idea de sus aficiones, intereses, costumbres y nivel de conocimientos informáticos.

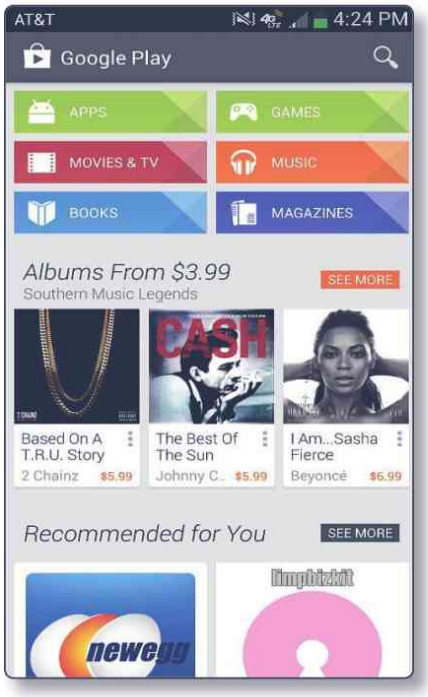

Figura 3.10. Interfaz para descarga de aplicaciones Android

Figura 3.11. Administrador de tareas con la pestaña de descargas activa

Por ejemplo: Youtube nos habla de lo que le gusta ver. Un programa de bolsa nos puede facilitar pistas sobre el tipo de inversor que es. Si además dispone de Wifipass, Intercepter-ng y otras herramientas de *sniffing* y monitorización de redes, o se detectan prácticas de *rooting* en el dispositivo con el objeto de adquirir privilegios de superusuario, entonces nos encontraremos con una información de contexto que puede ser de gran interés. Convenientemente configurado, Android se convertirá en una poderosa herramienta de *hacking* y espionaje. Más adelante veremos una prueba de concepto que nos permitirá hacernos una idea de las potencialidades delictivas de Android.

3.2.9 Administrador de tareas

La aplicación **Administrador de tareas** presenta un resumen del estado de funcionamiento del terminal y de los programas que se encontraban en ejecución en el momento de ser encontrado. En su menú principal, de tipo cinta, encontramos varias pestañas: **Aplicaciones**, con una lista de los programas activos; **Descargas**, donde se pueden ver las aplicaciones descargadas por el usuario; **RAM**, **Gestor de memoria**, con una barra de progreso que indica ocupación y disponibilidad de la memoria del dispositivo, así como un botón para terminar aplicaciones liberando espacio en memoria; y, finalmente, **Almacenamiento**, donde se indica el grado de utilización de los espacios de memoria NAND destinados a guardar datos: almacenamiento del sistema, almacenamiento USB y tarjeta SD. Desde esta interfaz se puede controlar el consumo de recursos por las aplicaciones, detenerlas para liberar memoria RAM o desinstalarlas (Figura 3.11).

3.2.10 Ajustes

Existen dos modos de llegar al apartado de **Ajustes del sistema** (Figura 3.12): pulsando el icono en forma de rueda dentada que hay en la ventana de **Aplicaciones**, o a través del menú contextual (botón luminoso del dispositivo situado en la parte inferior izquierda junto al botón mecánico de origen [**Home**]). **Ajustes** es importante por la misma razón que el administrador de tareas: proporciona información de contexto sobre el estado del teléfono. Dentro de este apartado encontraremos gran cantidad de secciones que permiten configurar el funcionamiento del dispositivo, adaptándolo a las necesidades del usuario y optimizando su rendimiento en función de lo que se quiera hacer con él: podemos configurar conexiones inalámbricas, sonidos de llamada y alarma, opciones de ubicación, seguridad y ahorro de energía, cuentas utilizadas para el acceso a Internet, correo electrónico y otros usos, privacidad, etc.

De gran interés es la sección correspondiente a **Conexiones inalámbricas**, con ajustes wifi, Bluetooth y redes de datos 3/4G, además de una lista de puntos de acceso (**Ajustes Wi-Fi**) a los que el usuario ha estado conectándose en el transcurso de sus desplazamientos geográficos con el terminal.

Figura 3.12. Ajustes

3.2.11 Otros contenidos

Finalmente, los terminales móviles Android incluyen otros tipos de datos como los que se detallan a continuación:

- Mensajes de aplicaciones de chat y mensajería instantánea.
- Coordenadas GPS.
- Historial de búsquedas de Google.
- Direcciones y rutas de navegadores de carretera.
- Entradas en Facebook, Twitter y otras redes sociales.
- Colecciones de música.
- Anotaciones en el calendario.
- Notas de texto.
- Archivos compartidos.
- Historial de compras y subastas.
- Información financiera y de bolsa.
- Etcétera.

3.3 TARJETA DE MEMORIA

Se puede investigar un naufragio examinando los restos que flotan en la superficie, pero si queremos hacernos idea de lo que realmente ha sucedido es necesario llamar a un buzo. Solo así llegaremos hasta el barco hundido y podremos averiguar contra qué chocó, cómo se hundió y qué ha pasado con el cargamento. Además de los elementos de evidencia directamente visibles, en un dispositivo Android hay más material que puede ser aprovechado por el investigador.

Ha llegado la hora de aplicar técnicas forenses en el mejor sentido del término.

3.3.1 Adquisición con dd

La tarjeta de memoria MicroSD suele hallarse localizada en el lateral del dispositivo. En determinadas marcas y modelos es preciso quitar la batería, con los inconvenientes técnicos —apagado forzoso del terminal— y procesales —alteración del dispositivo y pérdida de elementos de evidencia potenciales— que ello supone. En tales situaciones se ha de considerar la posibilidad de adquirir los contenidos de la tarjeta a través de la interfaz USB. Si el medio de almacenamiento consiste en un espacio simulado en NAND y el terminal no dispone de bahía para insertar tarjetas SD, la extracción tendrá que llevarse a cabo a través del cable USB.

Casi todas las tarjetas MicroSD utilizadas en dispositivos móviles Android están formateadas con el sistema de archivos Microsoft FAT32. Una vez retirada del terminal, la tarjeta puede adquirirse directamente en una estación de trabajo Windows o Linux a través de un adaptador.

En teoría no hace falta un bloqueador de escritura, ya que las particiones FAT32 carecen de *journaling* y no experimentan cambios al ser montadas. Pero nunca se sabe lo que un ordenador mal configurado puede hacer con los soportes de datos. Por otra parte, el abogado de la parte contraria se servirá de cualquier pretexto para atacar nuestra argumentación. Si disponemos de un bloqueador de escritura con puertos para interfaz USB, debemos utilizarlo en cualquier caso (Figura 3.13).

Figura 3.13. Bloqueador de escritura para conexión USB

Para adquirir la tarjeta se puede emplear cualquier herramienta comercial (EnCase Forensics, FTK, SMART) o de código libre (por ejemplo `dd`). Nos vamos a servir de `dd` por tratarse de una utilidad veterana y fiable que desde hace más de tres décadas viene incluida de serie en Unix/Linux. También existen versiones para Windows y OSX:

```
igandekoa@igandekoa:~$ sudo dd if=/dev/sdb of=tarjetaSD.dd bs=1024
```

La sintaxis de `dd` está compuesta por dos elementos principales —origen y destino— y varias opciones. Mediante `if=/dev/sdb` especificamos el origen de los datos que se quieren copiar, en este caso el dispositivo software creado por el sistema para representar el soporte de datos. Dependiendo de las características del sistema, dicho dispositivo puede variar. En nuestra situación, `/dev/sdb` indica la segunda unidad de disco duro conectada al sistema, aunque en la práctica la denominación es asignada a cualquier tipo de soporte que se conecte una vez encendido el ordenador. Si tuviésemos más discos duros, al soporte de datos conectado por USB se le habría adjudicado el disposivo `/dev/sdc`, o `/dev/sdd`, etc. Con las versiones de `dd` para Windows los nombres de los volúmenes y las particiones son diferentes, ya que deben adaptarse a los esquemas de denominación de unidades de disco típicos de Microsoft.

En el presente ejemplo estamos trabajando en Linux. Es importante tenerlo porque para utilizar `dd` a menudo nos veremos obligados a asumir privilegios de superusuario o `root`. Será necesario repasar atentamente todo lo que se escribe en línea de comando, ya que de lo contrario existe la posibilidad de dañar el sistema o borrar por accidente datos valiosos. Para estar seguro, el investigador deberá consultar antes la salida del comando `dmesg`, el cual le informa sobre la denominación exacta de los dispositivos y las características de los mismos.

El parámetro `of=tarjetaSD` indica a `dd` que los datos procedentes del origen deben empaquetarse en un archivo de imagen con extensión `dd`. La opción `bs=1024` uniformiza el flujo de datos en bloques de tamaño definido para optimizar el proceso de copia. En la práctica no es tan importante, sobre todo si se dispone de una plataforma potente.

Las tarjetas SD tienen capacidades comprendidas entre varios cientos de megabytes y 64 o 128 GB, dependiendo del límite soportado por el modelo de *smartphone* y la versión de Android. El tiempo que tarda en completarse una adquisición física depende del tamaño del medio sin importar el número de archivos que contenga, puesto que `dd` extrae también el espacio sin asignar, con todos los archivos borrados, datos preexistentes al último formateado del soporte, zonas ocultas y sin particionar y estructuras administrativas del sistema de archivos.

3.3.2 Hash

Una vez finalizada la adquisición y precintada la tarjeta MicroSD, para que pueda ser presentada como prueba, es preciso obtener la suma de verificación criptográfica (*hash*) del archivo. Un *hash* es una función unidireccional que a partir de un flujo de datos —texto, imagen, código o información digital en cualquier formato— calcula una firma característica compuesta por un número fijo de caracteres hexadecimales. El *hash* es único para cada archivo, de manera que sirve para demostrar que aquel no ha sufrido alteraciones, garantizando así la integridad de la cadena de custodia. Tanto en Linux como en Windows podemos utilizar los algoritmos MD5 o SHA256. Este último se recomienda por su robustez criptográfica y su mayor rendimiento.[6]

```
igandekoa@igandekoa:~$ sha256sum tarjetaSD.dd
25fdb84a1cea2dbf28ebe613dff88bf2ba9b749af145f49906a41817520a000d  tarjetaSD.dd
```

3.3.3 Montaje de la imagen

Tras la extracción de los *hashes* se procede a realizar copias de seguridad de la imagen para trabajar sobre ellas con herramientas de análisis forense. Una de estas copias habrá de ser enviada a la parte contraria (junto con el *hash* de la primera imagen, que debe coincidir con los *hashes* de todas las copias obtenidas). Otra copia deberá ser puesta a disposición de la autoridad judicial. Una vez que la imagen ha sido copiada al disco duro de la estación de trabajo forense, podemos examinar sus contenidos montándola como si fuera una unidad de disco normal. Para ello es preciso crear un directorio, por ejemplo:

```
igandekoa@igandekoa:~$ mkdir evidencia
```

Y a continuación:

```
igandekoa@igandekoa:~$ sudo mount –o loop tarjetaSD.dd zzzz
```

En nuestro ejemplo seguimos trabajando con Ubuntu Linux. Dependiendo de la distribución que se utilice, puede que el comando `mount` exija una sintaxis diferente. Tras esto, y simplemente con trasladarse al directorio `/home/evidencia`,

---

6 Y también porque los criptógrafos opinan que MD5 es vulnerable a determinados ataques que en teoría permitirían obtener dos *hashes* idénticos a partir de archivos diferentes. Esta vulnerabilidad no tiene apenas consecuencias en la práctica, pero el hecho de que exista, y de que SHA256 se vea de manera comprobada totalmente libre de tales ataques, resulta suficiente para dar prioridad a este último método.

el investigador podrá explorar los contenidos de la imagen y extraer elementos de evidencia, sirviéndose para ello de un navegador de archivos o ejecutando los comandos Linux estudiados en el capítulo anterior.

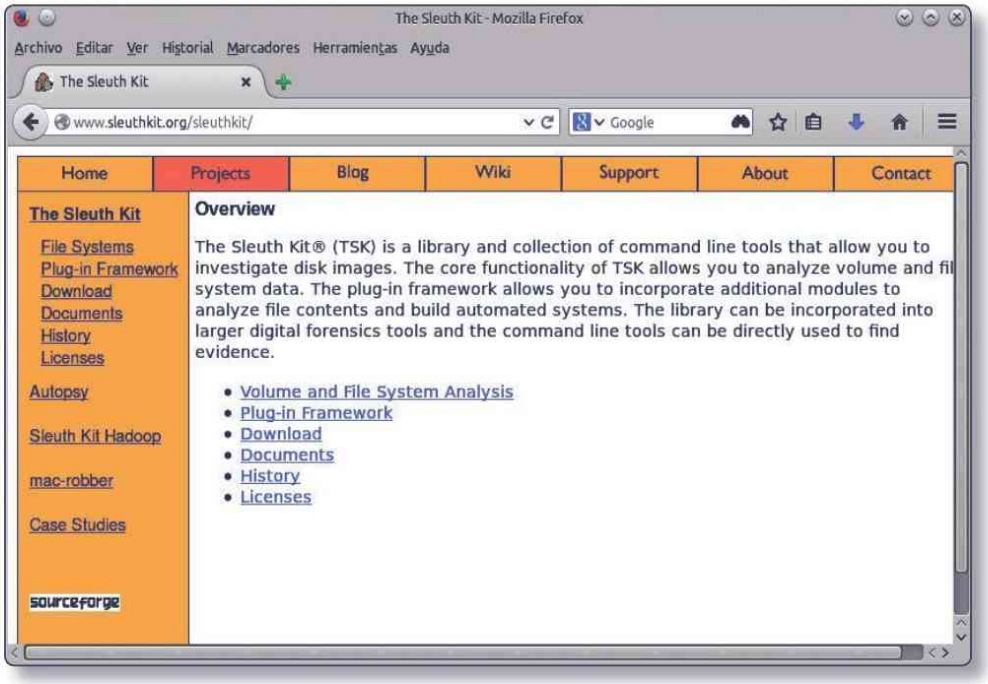

Figura 3.14. Página web de TSK

3.3.4 Análisis forense con TSK

El análisis forense de la imagen se puede llevar a cabo con herramientas comerciales o de código libre. También se puede utilizar software convencional para la recuperación de los archivos borrados. Sin embargo, si se tiene previsto comparecer ante un tribunal o una autoridad pública, se recomienda el empleo de herramientas homologadas o que dispongan de algún grado de aceptación oficial.

Entre las herramientas de código libre destaca la suite de comandos The Sleuth Kit (TSK), desarrollada por Brian Carrier, autor de un libro clásico sobre análisis forense de sistemas de archivos (véase *Bibliografía*). TSK puede descargarse desde la página web del autor en *http://www.sleuthkit.org*, donde existen versiones para Windows y Linux (Figura 3.14). Se pueden obtener tanto los archivos ejecutables como el código fuente. En distribuciones populares, como por ejemplo Ubuntu, TSK se encuentra incluido en los repositorios oficiales, por lo que para instalarla no habrá más que teclear:

```
igandekoa@igandekoa:~$ sudo apt-get install sleuthkit
```

Las versiones del software almacenado en los repositorios suelen permanecer estáticas de una versión de Ubuntu a la siguiente, y a veces durante períodos más largos, por lo que en caso de ser necesaria una actualización de TSK, con características mejoradas —como corrección de fallos, comandos adicionales o soporte para nuevos sistemas de archivos— existe la posibilidad de descargar los archivos de la página web e instalarlos directamente. El usuario con conocimientos de programación también puede modificar el código para incluir funcionalidades nuevas, compilándolo después para obtener los ejecutables.

TSK es una colección de comandos diseñados para el análisis de soportes de datos y sistemas de archivos a diferentes niveles, desde el medio físico hasta características avanzadas de *journaling* en sistemas NTFS, ReiserFS y ext3/4. Sus opciones y posibilidades —incluyendo la integración con el entorno gráfico Autopsy y la creación reciente de un *framework* que permite estructurar el flujo de análisis forense sobre módulos especializados— son demasiado extensas para ser tenidas en cuenta dentro de la presente obra. Para un conocimiento detallado de las mismas remitimos a la página web del autor. Brian Carrier mantiene al día el software y los contenidos del sitio. Su página web es el lugar indicado para documentarse sobre los progresos de TSK y otros temas de informática forense.

Veamos algunos ejemplos de uso con imágenes adquiridas a partir de tarjetas MicroSD, procedentes de dispositivos Android y formateadas con sistemas de archivos FAT32. Con `fsstat` se pueden observar las características generales del sistema de archivos:

```
igandekoa@igandekoa:~$ fsstat –o 63 tarjetaSD.dd
```

He aquí un ejemplo típico de salida de `fsstat`:

```
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: FAT16
OEM Name: android
Volume ID: 0x6a0f1f0b
Volume Label (Boot Sector): NO NAME
Volume Label (Root Directory):
File System Type Label: FAT16
Sectors before file system: 0
File System Layout (in sectors)
Total Range: 0 - 3987455
* Reserved: 0 - 23
** Boot Sector: 0
* FAT 0: 24 - 267
* FAT 1: 268 - 511
```

```
* Data Area: 512 - 3987455
** Root Directory: 512 - 543
** Cluster Area: 544 - 3987423
** Non-clustered: 3987424 - 3987455

METADATA INFORMATION
------------------------------------------
Range: 2 - 63791110
Root Directory: 2

CONTENT INFORMATION
------------------------------------------
Sector Size: 512
Cluster Size: 32768
Total Cluster Range: 2 - 62296

FAT CONTENTS (in sectors)
------------------------------------------
608-671 (64) -> EOF
672-735 (64) -> EOF
736-799 (64) -> EOF
800-863 (64) -> EOF
864-927 (64) -> EOF
928-991 (64) -> EOF
992-1055 (64) -> EOF
1056-1119 (64) -> EOF
1120-1183 (64) -> EOF
1184-1247 (64) -> EOF
1248-1311 (64) -> EOF
1312-1375 (64) -> 47904
.............
```

Esto es solo una parte. Truncamos la salida por conveniencias de espacio. Toda esta información, que incluye la distribución de los bloques de datos con sus sectores correspondientes, puede ser útil por ejemplo a la hora de recuperar datos borrados. Otro elemento de la caja de herramientas de TSK es el comando `fls`, que lista archivos y carpetas con sus números de entrada en la tabla de asignación de archivos (denominados *inodes* en Linux):

```
igandekoa@igandekoa:~$ fls tarjetaSD.dd
```

Ejemplo típico de *output*:

```
d/d 3:  LOST.DIR
d/d 6:  .android_secure
```

```
d/d 8:  .downloadTemp
d/d 10: download
d/d 13: com.cjbridge.smartpodcast
d/d 14: DCIM
d/d 15: FOTOS
d/d 19: Liszt_Rapsodias_Hungaras_Cziffra_FLAC
r/r * 21:       mtptemp.tmp
v/v 63791107:   $MBR
v/v 63791108:   $FAT1
v/v 63791109:   $FAT2
d/d 63791110:   $OrphanFiles
```

Finalmente, en caso de que nos interese recuperar alguno de los archivos borrados —marcados con un asterisco en el listado de `fls`—, quizás podamos hacerlo con el comando `icat`:

```
igandekoa@igandekoa:~$ icat tarjetaSD.dd 764438 > imagen.jpg
```

En el ejemplo anterior hemos supuesto, por la extensión del archivo correspondiente al *inode* facilitado como parámetro, que se trata de un archivo de imagen. Si tenemos dudas, tras la extracción debemos examinarlo con el comando `file` estudiado en el capítulo anterior.

3.4 DATA CARVING

El montaje de la imagen y las herramientas de TSK nos permiten acceder a los contenidos visibles del soporte de datos y también —aunque no siempre es posible— a los archivos borrados de la tarjeta SD. Pero ¿qué hay del período anterior al último reformateado del soporte de datos? Reformatear un soporte equivale a una reinicialización total del sistema de archivos. TSK no podrá decirnos nada de lo que había antes, ya que únicamente analiza las estructuras del sistema de archivos que se creó después de poner a cero todos los parámetros del que existía con anterioridad. Para saber si puede haber elementos de evidencia fuera del ámbito de control del sistema de archivos actual, el único recurso de que disponemos son las técnicas de *data carving* (tallado de archivos).

El principio de funcionamiento es simple: explorar secuencialmente el soporte de datos en busca de cadenas de caracteres características de cada tipo de archivos, y, en caso de encontrar una coincidencia, extraer al disco duro de la estación de trabajo forense un número de sectores arbitrario y lo suficientemente grande para incluir el archivo completo, según estimaciones habituales.

Esto presupone que el archivo se encuentra guardado en una zona de bloques contigua, ya que las herramientas de *data carving* no son capaces de distinguir entre archivos continuos y fragmentados. La contigüidad de los archivos no está asegurada en ninguno de los casos. Por suerte, el tamaño cada vez mayor de los soportes de datos aumenta la probabilidad de que los archivos estén guardados en zonas de bloques contiguas. Los archivos de gran tamaño —vídeos, audios, bases de datos— son difíciles de recuperar mediante técnicas de *data carving*, ya que raramente se guardan sin experimentar algún grado de fragmentación. Sin embargo, con archivos pequeños (documentos Word o PDF, imágenes, código fuente, etc.) suele haber más suerte.

La recuperación de archivos puede llevarse a cabo con la ayuda de numerosas utilidades comerciales y de código libre. EnCase, FTK y SMART Forensics incluyen herramientas para operaciones de *data carving*. Existen así mismo utilidades *shareware* muy interesantes: EasyRecovery Pro, GetDataBack, PC-Inspector, R-Studio, etc. La mayor parte han sido desarrolladas para ayudar a programadores y técnicos de sistemas a recuperar datos perdidos como consecuencia de fallos o manejo inadecuado de equipos informáticos.

El investigador puede recurrir a otras herramientas de software libre que no han sido desarrolladas por expertos en forénsica, pero resultan de gran interés por su rapidez, facilidad de manejo, elevado rendimiento y bajo consumo de recursos. Este es el caso de Testdisk y Photorec (Figura 3.15), utilidades de fácil instalación (disponibles en repositorios Ubuntu y Debian con solo teclear "`sudo apt-get install testdisk`"). La interfaz de usuario es de tipo `ncurses`, con opciones seleccionables mediante las teclas de flecha y retorno.

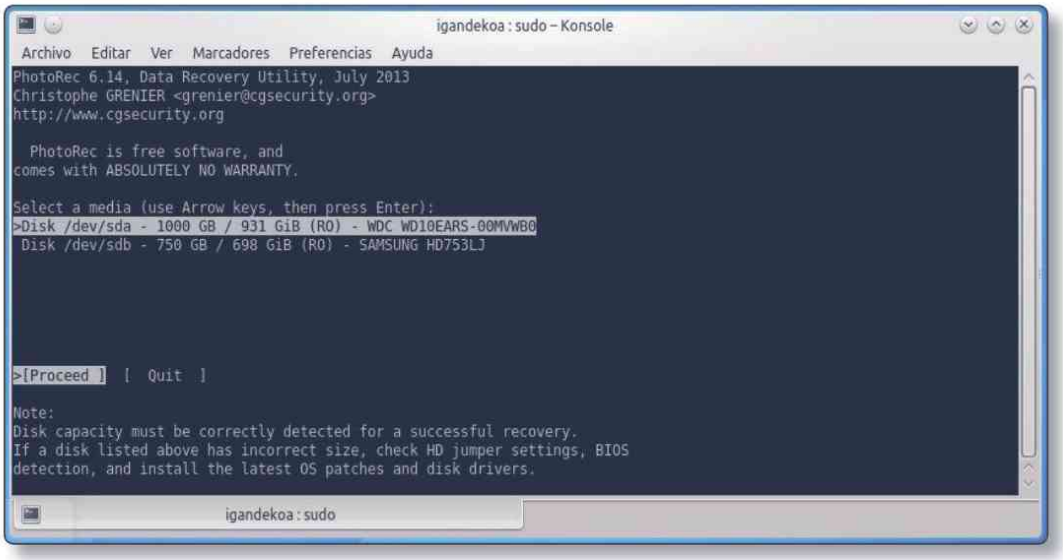

Figura 3.15. Photorec en Linux Kubuntu 13.04.

Por su parte, Foremost y Scalpel (el segundo es una versión perfeccionada del primero) son herramientas Linux en línea de comando. Foremost fue creado en 2001 por dos investigadores de las Fuerzas Aéreas de EEUU, Kris Kendall y Jesse Kornblum. Scalpel, publicado en 2005, aplica una estrategia optimizada de *data carving* con soporte para entornos distribuidos. En ambos casos el uso es simple y directo:

```
igandekoa@igandekoa:~$ foremost tarjetaSD.dd
```

La herramienta recorre el archivo de imagen en busca de cadenas de caracteres y guarda el resultado en un árbol de carpetas diferenciado por tipos, con un directorio para cada clase de archivos que consigue recuperar. Foremost y Scalpel se controlan mediante archivos de configuración que incluyen las firmas de caracteres para gran cantidad de tipos de archivo. El usuario puede ampliar el inventario incluyendo secuencias de caracteres correspondientes a archivos nuevos o poco habituales, o excluyendo determinados tipos de archivo —mediante un carácter almohadilla (#) al principio de la línea de texto correspondiente—. Esto permite que la herramienta se concentre en los elementos de evidencia que interesan al investigador, aumentando así la velocidad de ejecución.

3.5 ACCESO MEDIANTE ADB

Tras un examen superficial del dispositivo y después de la adquisición de la tarjeta MicroSD, el investigador dispone de elementos de evidencia que pueden resultar suficientes, o no, para las finalidades de su trabajo. En caso de que no sea así, se verá en la necesidad de emplear métodos más intrusivos y con mayor riesgo desde el punto de vista legal. En este viaje al interior de Android estamos a punto de iniciar la etapa más interesante y peligrosa, donde se trata sobre técnicas de adquisición y análisis que posibilitan el acceso directo a elementos que por lo general solo se encuentran al alcance de un desarrollador familiarizado con tareas de programación y depuración de código. También será necesario realizar cambios en la configuración y las estructuras de datos del dispositivo. Por ello se requiere prudencia. Es preciso entender lo que se hace y llevar un registro detallado de las actuaciones con el objeto de poder explicarlas después delante del tribunal.

3.5.1 Interfaces USB

En el momento de conectar un terminal Android al ordenador a través del cable USB, queda expuesto al PC de sobremesa o portátil donde tenemos instalada nuestra estación de trabajo forense un número de interfaces necesarias para la gestión

y el mantenimiento del dispositivo. Las funciones principales de una conexión USB son las siguientes:

- Sincronización con el software del ordenador de sobremesa y realización de copias de seguridad.
- Utilización del terminal como disco de almacenamiento externo USB.
- Recarga de la batería a través del puerto USB.
- Interfaz para comunicaciones móviles de banda ancha.
- Depuración USB.

Los tres primeros no necesitan explicación. La interfaz para comunicación a través de redes de datos o MBC, habitual en dispositivos móviles Android, se utiliza para conectar un ordenador portátil o de sobremesa a Internet u otras redes a través de la red telefónica inalámbrica.

El investigador puede servirse de la sincronización para adquirir —a través de Samsung Kies u otro software por el estilo— copias de respaldo de los datos de usuario, incluyendo fotografías, archivos multimedia y otros elementos de evidencia. La conexión de disco sirve para realizar adquisiciones de tarjetas MicroSD mediante el procedimiento descrito en el apartado anterior, en caso de que no hubiera sido posible extraerlas físicamente (por encontrarse bajo la batería, estar atascadas dentro de la ranura o cualquier otra razón). También permite adquirir los soportes de almacenamiento virtuales simulados en memoria NAND.

3.5.2 Depuración USB

La interfaz de depuración USB a través de Android Debug Bridge (ADB) es la más importante de todas las funcionalidades que se mencionan en el apartado anterior. La utilizan los desarrolladores para comunicarse con dispositivos virtuales emulados por SDK y con dispositivos reales con el objeto de comprobar el funcionamiento de sus programas, visualizar estructuras internas de datos, copiar archivos, instalar aplicaciones en línea de comando y otros cometidos. Todas estas posibilidades hacen de ADB la herramienta perfecta para que un investigador forense acceda a zonas de un terminal Android que resultarían imposibles de alcanzar a través de los procedimientos examinados en el apartado precedente, como por ejemplo particiones del sistema, archivos de configuración, caché, historiales, bases de datos y directorios de almacenamiento interno.

Los dispositivos virtuales generados con el emulador tienen activado por defecto el puente USB y no necesitan cable, ya que la conexión al ordenador tiene lugar internamente a través de Android SDK. Para acceder a un terminal físico —*smartphone*, tableta, reproductor MP3— es necesario habilitar la opción **Depuración USB** (**USB Debugging**) en **Ajustes del sistema** (Figura 3.16).

Figura 3.16. Opciones de desarrollo y Depuración USB

En Android 2.3 Gingerbread, por ejemplo, esto se consigue yendo a **Ajustes → Aplicaciones → Desarrollo**, para marcar la casilla **Depuración USB**. En Android 4.1/4.2 Jelly Bean el procedimiento es más intrincado. A Google no le interesa que los usuarios enreden en el cuarto de máquinas, a no ser que tengan una buena razón para hacerlo. Vaya a **Ajustes → Información del dispositivo**. Una vez allí, marque varias veces sobre el número de compilación del sistema. Insista lo que haga falta. Al final aparecerá un mensaje diciendo: "Ahora eres un desarrollador Android" y la depuración USB estará activa en el terminal.

3.5.3 Funcionamiento de ADB

Android Debugging Bridge (ADB) es una herramienta en línea de comando que permite al usuario establecer comunicación con dispositivos físicos o virtuales provistos de sistemas operativos Android. Su diseño, como el de cualquier otra aplicación típica de software de red, está basado en una arquitectura servidor-cliente y consta de tres componentes básicos que interactúan entre sí:

- Un **cliente** que se ejecuta en la máquina de desarrollo, en nuestro caso la estación de trabajo forense, y que se pone en funcionamiento —tanto en Windows como en Linux— a través del comando `adb`.

- Un **servicio local `adb`** en el dispositivo móvil, que funciona en segundo plano como demonio (*daemon*) Unix/Linux, y que tiene la misión de atender llamadas procedentes del cliente `adb` que se ejecuta en la estación de trabajo.

- Finalmente, un **servidor** que se ejecuta en segundo plano en la estación de trabajo forense y gestiona el proceso de comunicación entre el cliente y los servicios `adb` que corren en los dispositivos móviles Android.

Lo primero que el cliente `adb` hace nada más iniciarse en la estación de trabajo, al ser llamado por el usuario mediante el comando `adb`, es comprobar si hay un servidor `adb` ejecutándose en segundo plano. En caso de que no sea así, lo pone en funcionamiento. Una vez en marcha, el servidor se conecta al puerto TCP 5037 y permanece a la escucha de órdenes enviadas desde los clientes `adb`, los cuales utilizan por defecto el mismo puerto TCP 5037. A continuación el servidor establece conexiones para todos los dispositivos físicos, tanto reales como emulados, que puedan estar conectados al sistema —a través de la red virtual instalada por encima de la interfaz USB—, asignando puertos disponibles dentro del rango comprendido entre el 5555 y el 5585.

Cada dispositivo necesita dos puertos, uno para la conexión `adb` y otro para la consola de comandos. El servidor asigna puertos pares a `adb` e impares a la consola. Una vez establecida la conexión, el usuario puede utilizar los comandos `adb` para comunicarse con el terminal, trasladar datos entre la estación de trabajo y el terminal o viceversa, ejecutar comandos locales del terminal y realizar otras operaciones (Figura 3.17).

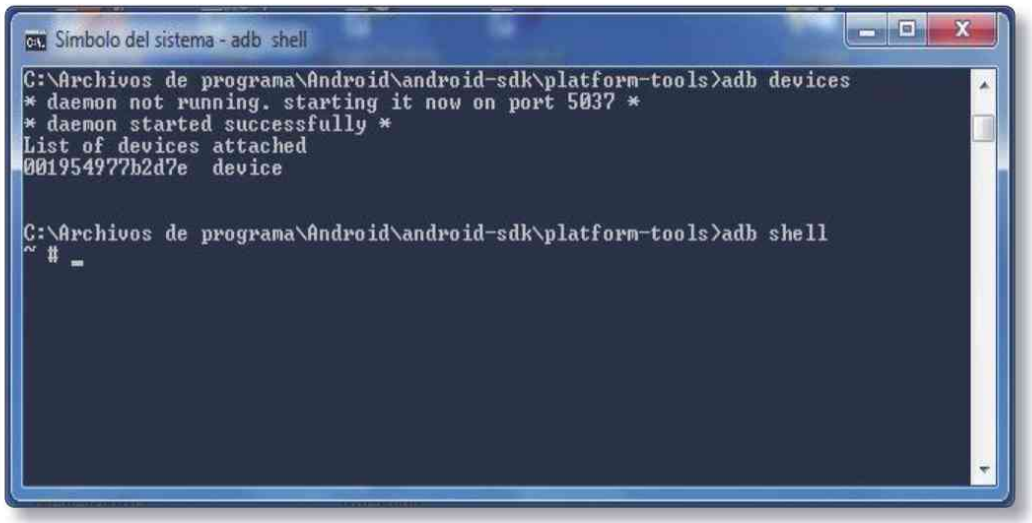

Figura 3.17. Consola ADB en Windows 7

- Un **servicio local `adb`** en el dispositivo móvil, que funciona en segundo plano como demonio (*daemon*) Unix/Linux, y que tiene la misión de atender llamadas procedentes del cliente `adb` que se ejecuta en la estación de trabajo.

- Finalmente, un **servidor** que se ejecuta en segundo plano en la estación de trabajo forense y gestiona el proceso de comunicación entre el cliente y los servicios `adb` que corren en los dispositivos móviles Android.

Lo primero que el cliente `adb` hace nada más iniciarse en la estación de trabajo, al ser llamado por el usuario mediante el comando `adb`, es comprobar si hay un servidor `adb` ejecutándose en segundo plano. En caso de que no sea así, lo pone en funcionamiento. Una vez en marcha, el servidor se conecta al puerto TCP 5037 y permanece a la escucha de órdenes enviadas desde los clientes `adb`, los cuales utilizan por defecto el mismo puerto TCP 5037. A continuación el servidor establece conexiones para todos los dispositivos físicos, tanto reales como emulados, que puedan estar conectados al sistema —a través de la red virtual instalada por encima de la interfaz USB—, asignando puertos disponibles dentro del rango comprendido entre el 5555 y el 5585.

Cada dispositivo necesita dos puertos, uno para la conexión `adb` y otro para la consola de comandos. El servidor asigna puertos pares a `adb` e impares a la consola. Una vez establecida la conexión, el usuario puede utilizar los comandos `adb` para comunicarse con el terminal, trasladar datos entre la estación de trabajo y el terminal o viceversa, ejecutar comandos locales del terminal y realizar otras operaciones (Figura 3.17).

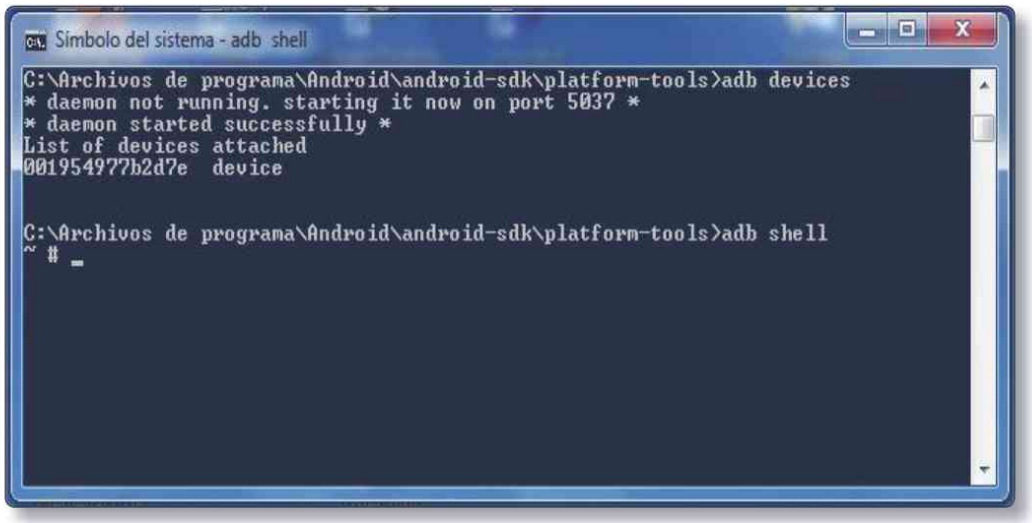

Figura 3.17. Consola ADB en Windows 7

En determinadas circunstancias puede ser necesario establecer una conexión por vía inalámbrica. Los métodos se explican en diversas páginas de Internet y resultan algo complejos, ya que requieren un trabajo previo de redireccionamiento de puertos y configuración de redes tanto en el dispositivo móvil como en la estación de trabajo.

3.5.4 Empleo de ADB

ADB es una herramienta compleja y muy potente. Con ella los programadores solucionan la mayor parte de los problemas que se presentan en el día a día del desarrollo de aplicaciones Android. Una vez establecido por cable el enlace entre nuestra estación de trabajo y un terminal Android que tenga activada la depuración USB, lo primero que el investigador suele hacer es ejecutar el más simple de todos los comandos de ADB, que sirve para comprobar que el dispositivo está en línea y preparado para el acceso:

```
igandekoa@igandekoa:~/android-sdk-linux/platform-tools$ adb devices
List of devices attached
4696C2228D44BF63489A743B485C8EC device
```

A veces puede ser necesario rearrancar el servidor ADB. Para lograrlo lo único que hace falta es terminar el proceso en ejecución con el comando `kill`. En la siguiente ocasión en que el usuario llame a `devices`, el servidor se reinicia:

```
igandekoa@igandekoa:~/android-sdk-linux/platform-tools$ adb kill-server
igandekoa@igandekoa:~/android-sdk-linux/platform-tools$ adb devices
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
4696C2228D44BF63489A743B485C8EC device
```

La denominación de los dispositivos varía con la marca y el modelo de terminal. En este caso tenemos conectado al ordenador de sobremesa un Samsung Galaxy Ace 2. Si la salida del comando es nula, eso quiere decir que no hay ningún dispositivo en línea, que el sistema no ha podido reconocerlo por falta de un controlador específico, o que el terminal no tiene activada la depuración USB. Si el investigador utiliza Android SDK en Windows, debe conseguir los controladores en la página web del fabricante o buscar en los CD de soporte del producto. En Linux es preciso añadir algunas líneas al archivo de reglas de `udev` para que el hardware sea reconocido durante la conexión o el inicio del sistema operativo.

Como se ha explicado anteriormente, ADB establece una conexión con el dispositivo a través del *daemon* que se ejecuta localmente en el terminal y el cliente `adb` de la estación de trabajo. Podemos realizar diversas operaciones desde el

directorio local de Android SDK en la estación de trabajo, enviando los comandos al terminal para que este los ejecute y nos devuelva el resultado:

```
igandekoa@igandekoa:~/android-sdk-linux/platform-tools$ adb shell ls -l var
drwxr-xr-x root     root          2013-07-23 00:35 lib
drwxr-xr-x root     root          2013-07-23 00:35 run
```

También podemos abrir en el terminal una sesión mediante el comando `adb shell`. Entonces el investigador estará trabajando sobre el propio dispositivo y podrá desplazarse a través de su árbol de directorios, listar sus contenidos y copiar a la estación de trabajo aquellos archivos que sean necesarios para su trabajo de análisis forense:

```
igandekoa@igandekoa:~/android-sdk-linux/platform-tools$ adb shell
$ ls -l
drwxr-xr-x root     root              2013-07-23 00:35 var
drwxrwxr-x radio    system            2013-06-27 16:57 efs
dr-x------ root     root              2013-07-23 00:35 config
drwxrwx--- system   cache             2013-07-23 01:36 cache
drwxrwxr-x system   system            2013-07-23 00:54 modemfs
lrwxrwxrwx root     root              2013-07-23 00:35 sdcard -> /mnt/sdcard
drwxr-xr-x root     root              2013-07-23 00:35 acct
drwxrwxr-x root     system            2013-07-23 00:35 mnt
lrwxrwxrwx root     root              2013-07-23 00:35 vendor -> /system/vendor
lrwxrwxrwx root     root              2013-07-23 00:35 d -> /sys/kernel/debug
lrwxrwxrwx root     root              2013-07-23 00:35 etc -> /system/etc
-rw-r--r-- radio    shell     5103    2012-04-23 22:14 ueventd.rc
drwxr-xr-x radio    shell             2012-04-23 22:34 res
drwxr-xr-x radio    shell             2012-04-23 22:34 lib
-rwxr-xr-x radio    shell     33836   2012-04-23 22:13 init.rc
drwxr-xr-x root     root              2013-07-23 00:35 dev
drwxr-xr-x radio    shell             2013-07-23 00:35 tmp
-rw-r--r-- radio    shell     1677    2012-04-23 22:14 init.goldfish.rc
drwxrwxr-x system   system            2013-07-23 00:05 system
dr-xr-xr-x root     root              1970-01-01 01:00 proc
-rw-r--r-- radio    shell     0       2012-04-23 22:14 ueventd.goldfish.rc
-rw-r--r-- radio    shell     466     2012-04-23 22:13 init.ux500.post_boot.sh
-rwxr-xr-x radio    shell     189104  2012-04-23 22:26 init
-rw-r--r-- radio    shell     1684    2012-04-23 22:27 recovery.rc
-rw-r--r-- radio    shell     118     2012-04-23 22:20 default.prop
drwxr-xr-x root     root              2013-07-23 00:35 sys
drwxr-xr-x radio    shell             2012-04-23 22:34 sbin
-rw-r--r-- radio    shell     1752    2012-04-23 22:13 init.u8500.rc
-rw-r--r-- radio    shell     3239    2012-04-23 22:13 lpm.rc
-rw-r--r-- radio    shell     3872    2012-04-23 22:13 prerecovery.rc
```

```
drwxrwx--x system system              2013-07-23 23:04 data
$
```

Recordemos que Android no es más que una distribución Linux adaptada a los requerimientos de un dispositivo móvil. En principio, a través de ADB resulta posible ejecutar todos los comandos Unix del terminal, con independencia de que el *shell* haya sido abierto desde Windows o Linux. Por ejemplo, con la ayuda del comando `df` se puede comprobar el grado de uso de la partición, los puntos de montaje y el espacio disponible:

```
$ df
Filesystem                          Size     Used     Free     Blksize
/dev                                277M     88K      277M     4096
/mnt/asec                           277M     0K       277M     4096
/dev/shm                            277M     0K       277M     4096
/mnt/obb                            277M     0K       277M     4096
/system                             602M     572M     29M      4096
/modemfs                            15M      4M       11M      4096
/cache                              301M     4M       296M     4096
/efs                                9M       4M       5M       4096
/data                               1G       276M     952M     4096
/mnt/.lfs: Function not implemented
/mnt/sdcard                         1G       116M     1010M    4096
/mnt/sdcard/external_sd             1G       954M     991M     32768
/mnt/secure/asec: Permission denied
/mnt/asec/com.scoreloop.games.geared-1 6M    4M       1M       4096
$
```

También podemos examinar los puntos de montaje y los sistemas de archivos correspondientes a cada una de las particiones de la memoria de estado sólido NAND. El comando `mount` permite localizar la partición de datos de usuario (montada en el directorio `/data`), estableciendo su asignación directa a un dispositivo de bloques en caso de que posteriormente sea necesaria una adquisición a bajo nivel con `dd` u otras herramientas forenses:

```
$ mount
rootfs / rootfs rw 0 0
tmpfs /dev tmpfs rw,relatime,mode=755 0 0
devpts /dev/pts devpts rw,relatime,mode=600 0 0
proc /proc proc rw,relatime 0 0
sysfs /sys sysfs rw,relatime 0 0
none /sys/kernel/debug debugfs rw,relatime 0 0
none /acct cgroup rw,relatime,cpuacct 0 0
tmpfs /mnt/asec tmpfs rw,relatime,mode=755,gid=1000 0 0
tmpfs /dev/shm tmpfs rw,relatime,mode=755,gid=1000 0 0
```

```
tmpfs /mnt/obb tmpfs rw,relatime,mode=755,gid=1000 0 0
/dev/block/mmcblk0p3 /system ext4 ro,relatime,barrier=1,data=ordered 0 0
/dev/block/mmcblk0p2 /modemfs ext4 rw,nosuid,nodev,noatime,barrier=1,
data=ordered 0 0
/dev/block/mmcblk0p4 /cache ext4 rw,nosuid,nodev,noatime,barrier=1,
data=ordered 0 0
/dev/block/mmcblk0p7 /efs ext4 rw,nosuid,nodev,noatime,barrier=1,
data=ordered 0 0
/dev/block/mmcblk0p5 /data ext4 rw,nosuid,nodev,noatime,barrier=1,
data=ordered,noauto_da_alloc 0 0
```

Hallándonos en Linux, también podemos investigar las conexiones de red abiertas, utilizando para ello el comando `netstat`, del mismo modo que hacemos en Windows o Linux:

```
$ netstat
Proto Recv-Q Send-Q Local Address          Foreign Address        State
tcp        0      0 127.0.0.1:7777         0.0.0.0:*              LISTEN
tcp        0      0 127.0.0.1:7203         0.0.0.0:*              LISTEN
tcp        0      0 127.0.0.1:32500        0.0.0.0:*              LISTEN
tcp        0      0 127.0.0.1:7777         127.0.0.1:39901        ESTABLISHED
tcp        0      0 127.0.0.1:7777         127.0.0.1:39900        ESTABLISHED
tcp        0      0 127.0.0.1:7777         127.0.0.1:39899        ESTABLISHED
tcp        0      0 127.0.0.1:7777         127.0.0.1:36194        ESTABLISHED
tcp        0      0 127.0.0.1:39899        127.0.0.1:7777         ESTABLISHED
tcp        0      0 127.0.0.1:39901        127.0.0.1:7777         ESTABLISHED
tcp        0      0 127.0.0.1:39900        127.0.0.1:7777         ESTABLISHED
udp        0      0 127.0.0.1:43000        0.0.0.0:*              CLOSE
udp        0      0 127.0.0.1:43001        0.0.0.0:*              CLOSE
tcp6       0      0 ::ffff:127.0.0.1:36194 ::ffff:127.0.0.1:7777  ESTABLISHED
$
```

Esto resultará de gran ayuda si sospechamos que el dispositivo móvil puede estar infectado por troyanos o atrapado en una red de zombis (*botnet*). En resumen, ADB es una herramienta poderosa. Sus funcionalidades han ido ampliándose con cada nueva versión de Android SDK. Los ejemplos anteriores resultan explicativos con respecto al tipo de tareas que se pueden llevar a cabo. ADB también permite realizar operaciones complejas, como instalar aplicaciones en línea de comando, redireccionar puertos entre la estación de trabajo y el terminal, copiar archivos de manera recursiva y listar archivos de registro (*logs*) tanto del sistema como de las aplicaciones.

3.6 EXTRAYENDO INFORMACIÓN CON ADB

Antes de pasar a la adquisición de archivos y elementos de evidencia a partir de un terminal Android es preciso conocer algunas utilidades de ADB que permitirán al investigador obtener una valiosa información contextual sobre el dispositivo móvil que está analizando. Estos comandos aún no permiten llegar a lo que nos interesa: los datos del usuario. Pero gracias a ellos quizás podamos averiguar algo interesante sobre el funcionamiento y la configuración del terminal.

3.6.1 dmesg

La primera de estas utilidades es `dmesg`, comando muy utilizado en Linux para hacer volcados del archivo de registro del *kernel*. Siendo el elemento clave de la arquitectura Android, la información que genera el *kernel* resulta de gran importancia para conocer la configuración del sistema y las actividades del usuario.

Podemos ejecutar `dmesg` de dos formas: primeramente desde el propio terminal Android una vez establecida la conexión y abierto el *shell*:

```
$ dmesg
```

O bien desde la estación de trabajo forense a través del comando *shell* de ADB. Este segundo procedimiento hace posible la captura de la salida en un archivo de texto que posteriormente podrá ser analizado fuera de línea, o bien agregado al inventario de evidencias o al informe:

```
igandekoa@igandekoa:~/android-sdk-linux/platform-tools$ adb shell dmesg >
kernel_log.txt
```

Entiéndase que cuando se habla de ejecutar un comando en la estación de trabajo o en el terminal, no se está haciendo referencia a ubicaciones físicas. Tanto en un caso como en otro el investigador trabaja desde el teclado y el monitor de su estación de trabajo forense. La ejecución de comandos en el terminal tiene lugar de forma remota, a no ser que se abra una consola de texto en el mismo dispositivo, lo cual también es posible, aunque requiere instalar aplicaciones de *hacking* y realizar en el terminal modificaciones de las cuales nos ocuparemos en un apartado posterior.

La salida del comando `dmesg` es más bien farragosa y está compuesta por gran número de líneas con información relativa a numerosas funciones, desde el proceso de arranque del dispositivo hasta el estado de carga de la batería. La ejecución de `dmesg` no requiere permisos especiales. Lo único que se necesita, como para cualquier otro comando de Linux que podamos pasar por ADB, es una conexión por cable y que esté activada la depuración USB.

3.6.2 logcat

El comando `logcat` (en el directorio `/system/bin`) proporciona un listado de mensajes emitidos por el sistema y las aplicaciones con el fin de informar al desarrollador sobre posibles errores. Su sintaxis de utilización es parecida a la de `dmesg`, con una verbosidad comparable en la salida de texto, y proporciona datos que por haberse producido como resultado de las acciones del usuario poseen un evidente interés para el investigador.

Por ejemplo, `logcat` puede mostrar información relativa a coordenadas geográficas —longitud y latitud— generada por el GPS, marcas de tiempo correspondientes a diversas funciones, y, finalmente, detalles de configuración de las aplicaciones. Cada línea viene marcada con un código inicial representativo del tipo de mensaje o aviso de fallo que el sistema transmite al desarrollador:

Método o tipo	Significado
I	Informativo
V	Comentario (*verbose*)
D	Depuración (*debug*)
W	Advertencia (*warning*)
E	Error
F	Fallo catastrófico (fatal)
S	Silencioso

Tabla 3.1. Tipos de mensaje logcat

Logcat es una herramienta sofisticada y muy potente. Fue creada para facilitar al programador datos relativos al funcionamiento del código sobre terminales concretos e información de contexto relacionada con aquellos aspectos funcionales de Android que tienen relevancia en el desarrollo de aplicaciones orientadas a la comunicación móvil. Con `logcat`, por ejemplo, es posible capturar mensajes de registro generados por el subsistema de telefonía celular:

```
igandekoa@igandekoa:~/android-sdk-linux/platform-tools$ adb shell logcat –b radio
```

El análisis de un volcado obtenido con `logcat` permite obtener elementos de evidencia e información contextual relacionada con la hora exacta de un suceso (registrado mediante marcas de tiempo Unix), comandos AT utilizados por el teléfono móvil, mensajes SMS, ubicación del terminal dentro de redes GSM, CDMA o inalámbricas, información de la compañía operadora, etc.

Finalmente, con `logcat` se pueden obtener listados de eventos:

```
igandekoa@igandekoa:~/android-sdk-linux/platform-tools$ adb shell logcat –b events
```

La cantidad de texto resultante es enorme. En la mayor parte de los casos el investigador se verá obligado a emplear *scripts* de búsqueda de caracteres, hojas de cálculo o las herramientas habituales de Linux para estos menesteres, como `grep` o `find`. Existen diversos tipos de evento que pueden aportar informaciones útiles en una investigación, como por ejemplo los comandos `INSERT` y `SELECT` para gestión de bases de datos SQLite.

3.6.3 dumpsys

El comando `dumpsys` suministra información referente a servicios del sistema en ejecución —con posibilidad de hacer volcados selectivos de los mismos—, *intents* en cola de proceso, actividades pertenecientes a las aplicaciones en memoria, números de proceso (PID), archivos abiertos y otros elementos. La salida de `dumpsys` es tan abundante como la de los comandos anteriores, pero está estructurada en un formato más claro, con secciones separadas que facilitan el trabajo del investigador. He aquí un ejemplo típico de uso de `dumpsys`:

```
igandekoa@igandekoa:~/android-sdk-linux/platform-tools$ adb shell dumpsys
```

Con `dumpsys` el investigador podrá refinar su labor de selección de datos hasta el más preciso detalle, incluyendo análisis cruzados de información referente a diversas funcionalidades del sistema: apertura de bases de datos, localización geográfica, señales enviadas por las torres de telefonía móvil, acciones producidas por la recepción de un mensaje de correo electrónico o un SMS, etc.

3.6.4 dumpstate

La utilidad `dumpstate` es una combinación de los comandos anteriores y otros varios característicos del entorno Linux (`ps`, `top`, `df`, etc.). Su salida, al igual que la de `dumpsys` —cuya ejecución está incluida en `dumpstate`—, se encuentra estructurada por secciones y proporciona información destinada a servir como material de referencia a los programadores de aplicaciones Android. El investigador forense puede aprovechar estos datos para su búsqueda de elementos de evidencia, la elaboración de informes periciales y otros fines.

3.6.5 bugreport

Finalmente, el comando `bugreport` no precisa mucho comentario, ya que se trata de una compilación de los anteriores. Es el investigador quien tiene que decidir si lo utiliza o, por el contrario, prefiere llevar a cabo operaciones más específicas. La ejecución de `bugreport`, al igual que la de `dumpstate` y `dumpsys`, solo tiene sentido cuando se hace desde la estación de trabajo forense a través de `adb shell` y no de manera local en el dispositivo móvil. La razón de ello es que el elevado consumo de recursos de estas herramientas altera la memoria RAM y los directorios de caché del terminal, con lo cual se puede causar la pérdida o sobreescritura de elementos de evidencia importantes para la investigación.

Por idénticas razones se desaconseja redireccionar la salida de comandos a archivos de texto guardados en el terminal, ya que ello supone alterar el objeto de la prueba. En un capítulo anterior se ha visto que en el mundo de la informática móvil resulta difícil realizar adquisiciones de elementos de evidencia sin riesgo de alterarlos. Siendo así, conviene hacer todo lo posible para no influir sobre el objeto de la investigación en una medida superior a lo estrictamente necesario.

La salida de comandos como `dumpstate` o `bugreport` está constituida por listados de texto que alcanzan tamaños considerables, con decenas de miles de líneas y un volumen de datos lo suficientemente grande para ocupar cuatro o cinco megabytes de espacio en memoria NAND, lo que plantea el riesgo de que datos borrados y otros elementos de evidencia potencialmente aprovechables queden sobreescritos y se pierdan de manera irreversible.

3.7 ROOTING

ADB pone a disposición del investigador forense gran cantidad de información relacionada con la configuración y el contenido de terminales Android. Lo que no permite, sin embargo, es llegar de manera inmediata a aquellas zonas del dispositivo que más nos interesan, como la partición de datos del usuario. Cuando intentamos trasladarnos hasta el directorio `/data/data`, donde se guardan los datos de las aplicaciones y la información personal del usuario, nos damos cuenta de que no es posible. Al no tener privilegios de superusuario, quizás el sistema nos deje hacer un listado de contenidos con `ls`, pero no podremos visualizar el contenido de los archivos mediante el comando `cat`, y tampoco será posible copiarlos a nuestra estación de trabajo forense. En algunos casos resulta imposible incluso acceder a la partición de datos con el comando `cd`.

3.7.1 Riesgos y precauciones del rooting

Estas limitaciones de acceso se presentan porque en un teléfono móvil o tableta Android, según salen al mercado con sus ajustes de fábrica, no están activados los permisos de administrador del sistema. Los desarrolladores tampoco contemplan un modo fácil de adquirir privilegios de superusuario, por ejemplo activando una opción en **Ajustes del sistema**. Es más, con frecuencia avisan de que la garantía de compra del terminal queda automáticamente anulada si el usuario utiliza técnicas de *rooting*. Hacen esto por variadas razones:

- **Rentabilidad**. En primer lugar, existe el riesgo de que los procedimientos de *rooting* dejen el terminal inutilizado. El fabricante no quiere que su servicio técnico se llene de aparatos enladrillados como consecuencia de intentos de *rooting* llevados a cabo por usuarios inexpertos, porque eso supone para la empresa un coste adicional.

- **Marketing**. Las compañías telefónicas subsidian la venta de dispositivos a cambio de que, además de que vengan configurados para acceder exclusivamente a sus redes de datos, también lleven instalado algún tipo de software publicitario, como aplicaciones del operador, demos de productos comerciales, etc. En un terminal "rooteado" resulta posible librarse de estos extras, en la mayor parte de los casos inútiles para el usuario (*bloatware*), con solo ordenar la desinstalación de los mismos en el apartado de **Ajustes**. El fabricante, como es de esperar, no puedee tolerar un perjuicio como este para la empresa de telefonía móvil.

- **Seguridad**. Los terminales rooteados, sobre todo en manos de usuarios inexpertos, son más vulnerables a ataques de todo tipo: virus, troyanos, *botnets*, estafas telefónicas, etc. También resultan más peligrosos porque con ellos se pueden utilizar herramientas de *hacking* para interceptar tráfico en redes locales y zonas wifi. El fabricante quiere que los usuarios naveguen por Internet, consuman servicios de información y se diviertan sanamente. No le interesa que anden por ahí mirando lo que hacen sus vecinos, secuestrando sesiones web o interceptando contraseñas de correo electrónico.

Más adelante hablaremos de las potencialidades criminales de un terminal Android con privilegios de *root*. De momento el interés del investigador forense reside en cómo lograr acceso a la partición de datos del usuario. Para ello es preciso poner el dispositivo en un estado especial que haga posible la ejecución de aplicaciones con permisos de superusuario en el terminal.

3.7.2 Norma fundamental de prudencia

En Internet existe gran cantidad de literatura amateur acerca del *rooting*. Impulsado por el deseo de conseguir un terminal con privilegios de superusuario a través de recetas fáciles, el usuario tiende a confundir conceptos que, aunque tengan que ver entre sí, significan cosas distintas: liberación de particiones *boot*, *rooting*, flasheado de *kernels*, particiones Recovery, ROM cocinadas, etc. Muchas veces este desinformado y frenético trajín termina en el fracaso, cuando no con un terminal inutilizado. No es la estrategia más adecuada para un investigador forense que haya de presentar los resultados de su trabajo ante un tribunal.

Antes de manipular dispositivos móviles que hayan de servir como prueba en un proceso judicial, conviene que el investigador tenga claros algunos conceptos elementales sobre el *rooting*. Solo así podrá alcanzar un triple objetivo: obtener elementos de evidencia de modo impecable y profesional, realizar la intervención con la menor alteración posible en el objeto de la prueba y explicar su intervención de manera convincente ante el juez o el destinatario de los informes periciales.

Se recomienda estudiar el funcionamiento de Android y Linux lo más a fondo que sea posible. También es conveniente hacer prácticas con algunos modelos de *smartphone* y tableta populares en el mercado. Si el caso que nos ocupa es importante, no está de más extremar la prudencia, aunque se entiendan bien los conceptos fundamentales de *rooting* y se haya adquirido destreza en la manipulación de teléfonos móviles. Una vez hayamos averiguado cuál es el método que debemos emplear para el rooteado de un terminal, se recomienda ensayarlo previamente con otro dispositivo de la misma marca y del mismo modelo para estar seguros de que funcionará.

3.7.3 ¿Qué es exactamente el rooting?

El término *root* procede del entorno multiusuario y multitarea de Linux/Unix, donde un número de usuarios accede al sistema e interactúa con él desde una configuración personalizada sin que ninguno pueda interferir con la actividad de los otros. Un usuario puede hacer lo que quiera con los recursos que él mismo crea —archivos, carpetas, etc.— o le han sido asignados por el sistema. Sin embargo, a no ser que pertenezca a un grupo con privilegios especiales o encuentre la forma de adquirirlos mediante algún truco de *hacking*, ni siquiera podrá salir de su directorio `home`, donde se encuentran sus documentos y los archivos de configuración del software que está autorizado a utilizar. El sistema de permisos, asignados a cada uno de los archivos del sistema, es la base de un modelo de seguridad que mantiene dentro de cauces ordenados la actividad de los usuarios.

En el modelo de administración Android existen algunas diferencias fundamentales con respecto a Linux. Los usuarios no son personas registradas en el sistema, sino aplicaciones que se ejecutan sobre la máquina virtual Dalvik con unos privilegios determinados por los permisos que se le asignan a dicha aplicación en el momento de instalarla.

El usuario con el nivel de privilegios más elevado no es una aplicación, sino una interfaz abstracta diseñada para ser manejada por una persona real a la que se denomina administrador del sistema, superusuario o *root*. La función es similar a la del Administrador o la cuenta de sistema en Windows. Este usuario privilegiado dispone de poder para gestionar sin límite todos los recursos del sistema: actualizaciones, grupos de usuarios, permisos de archivos, acceso a particiones y soportes de datos, controladores de hardware, configuraciones de red y cualquier otro elemento que podamos imaginar.

En otras palabras: dentro del micromundo del *smartphone*, el superusuario es una especie de dictador con poderes absolutos. En Android el *rooting* permite obtener acceso al nivel de privilegios que define a este usuario omnipotente, y con ello se obtiene la capacidad para administrar el sistema sin las limitaciones impuestas a los usuarios normales. Mediante *rooting* un usuario consigue el control absoluto de su terminal y la capacidad de hacer con él lo que quiera.

¿Por qué motivos un usuario desearía ser administrador? Por ejemplo, para instalar aplicaciones de terceros, desinstalar *bloatware* del fabricante o *apps* publicitarias del operador telefónico, y también para hacer cosas que resultan inviables desde una configuración de fábrica. En los últimos tiempos se ha desarrollado una cultura de *modding* o tuneado de dispositivos móviles que consiste en instalar sistemas alternativos con interfaces atractivas y características de funcionamiento especiales, en lugar de las versiones estándar con que los fabricantes lanzan sus terminales al mercado. Se trata de las célebres ROM de diseño o "cocinadas". Para instalar estos sistemas alternativos en el espacio ocupado por el software de fábrica es necesario que el dispositivo móvil, además de tener la partición de arranque desbloqueada (*unlocked*), funcione con privilegios de supersuario, ya que de lo contrario no resultará posible acceder a las particiones reservadas del sistema.

En foros de Internet es habitual la confusión entre términos como *unlocking*, *rooting* e instalación de ROM cocinadas. El investigador forense debe tener bien claro lo que le interesa: adquirir privilegios de superusuario para acceder a la partición de datos, que es donde se guarda la información generada por las aplicaciones de usuario: contactos, historial de llamadas, mensajes SMS, correos electrónicos, etc. Se trata de eso y de nada más. No buscamos desbloquear el arranque del dispositivo ni instalar una ROM cocinada —a no ser que las circunstancias de la investigación lo hagan necesario—. Ni siquiera interesa que el *rooting* sea permanente, sino que dure

lo bastante para extraer elementos de evidencia o realizar una imagen a bajo nivel de las zonas reservadas del sistema.

Existen técnicas de *rooting* que permiten hacer precisamente eso: obtener permisos de superusuario temporales que desaparecen tras un rearranque del sistema, dejando el dispositivo en el mismo estado en que se encontraba con anterioridad. En una investigación forense tales técnicas, suponiendo que estén disponibles y hayan sido debidamente verificadas, deben tener prioridad frente al *rooting* permanente, puesto que los cambios que producen en la configuración de un sistema que tenemos previsto adquirir mediante procedimientos forenses son temporales y fácilmente reversibles.

El proceso de *rooting* varía de unas versiones de Android a otras, así como entre los dispositivos de distintas marcas y modelos. A veces consiste en explotar un fallo de programación que anula el sistema de permisos de Android. Si este es el caso, el procedimiento puede ser tan simple como copiar un archivo ejecutable llamado `su` a una ubicación situada en la ruta de acceso de las aplicaciones del sistema (normalmente el directorio `/system/bin`) o a cualquier otra ubicación que disponga de permisos de acceso para todas las aplicaciones, como por ejemplo un directorio temporal.

Con el sistema de permisos de Android desactivado, una aplicación que solicite acceso a la memoria, los contactos o cualquier otro recurso lo puede conseguir sin restricciones de ningún tipo, simplemente ejecutando el comando `su`.

Figura 3.18. Icono típico para el superusuario en Android

3.7.4 su y SuperUser

No hay que confundir el binario `su` con el superusuario del sistema. Este último no está implementado mediante archivos ejecutables sino que forma parte del concepto de ingeniería del sistema operativo Linux, basado en estructuras jerárquicas y de privilegios: `su` (*switch user*) significa "conmutar usuario", y su misión consiste en hacer que las aplicaciones que recurren a él se ejecuten con privilegios de administración del sistema. Para controlar esta ampliación de priviliegios es por lo que se creó SuperUser.

SuperUser gestiona el acceso de las aplicaciones a los privilegios de superusuario, presentando en pantalla un mensaje de advertencia con los botones correspondientes para que el usuario decida si autoriza o no el acceso a determinados recursos del sistema o a proveedores de contenido. No es que este programa resulte de gran interés para los fines de una investigación forense. Sin embargo, en la mayor parte de los procedimientos de *rooting* se instala por defecto.

SuperUser figura representado mediante un icono característico cuyo diseño puede diferir dependiendo del procedimiento y las utilidades de *rooting* que se hayan utilizado. Normalmente, el usuario lo identificará por la presencia del carácter almohadillado (#) típico del indicador de superusuario en los indicadores de consola Unix/Linux (Figura 3.18).

3.7.5 Rooting temporal

De acuerdo con lo expuesto en el apartado 3.7.3, en un dispositivo Android existen tantas maneras de adquirir permisos de superusuario como posibilidades de explotar fallos de programación (*bugs*) que hagan posible un acceso privilegiado. En los primeros modelos del *smartphone* HTC Dream, que salieron al mercado a finales de 2008, no tardó en descubrirse que cualquier texto introducido a través del teclado físico era interpretado como un comando `shell` con privilegios de `root`. Google distribuyó un parche para eliminar el *bug*, pero algunos usuarios ya habían extraído una imagen del sistema defectuoso, que fue puesta en Internet a disposición de todo el que quisiera descargarla para disfrutar de acceso ilimitado a sus dispositivos móviles.

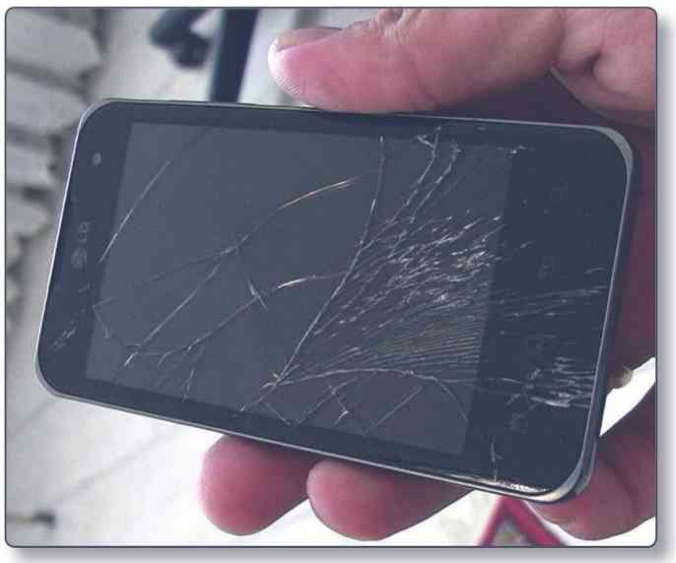

Figura 3.19. Ejemplo de rooting temporal

Posteriormente se han ideado otros procedimientos aplicables a versiones sucesivas de Android y a diferentes modelos de terminal. Algunos de estos métodos son temporales. Una vez ejecutado el *hack*, mediante conexión ADB o wifi, el dispositivo permanece rooteado mientras está encendido, pero tras apagarlo o hacer un *reset* regresa a su configuración básica con permisos de usuario normal. Tales procedimientos, si se puede recurrir a ellos, resultan útiles para acceder a la partición de datos con el propósito de extraer elementos de evidencia o realizar una imagen a bajo nivel mediante `dd`.

Los archivos del *hack*, copiados a una zona de la memoria NAND no privilegiada, cuya escritura no afecte a los datos del usuario y que tampoco sea vital para el funcionamiento del sistema—como por ejemplo una partición temporal o de caché— se pierden al apagar el dispositivo, una vez realizada la adquisición forense. De este modo se cumple un requisito fundamental de la práctica forense: alterar lo menos posible el objeto de la prueba.

3.7.6 Psneuter

Por ejemplo, el terminal que se muestra en la imagen (figura 3.19) es un LG Optimus 2X encontrado por la policía en el transcurso de una operación contra traficantes de droga. Una vez llevado al laboratorio forense, un examen superficial de las aplicaciones y archivos de usuario permitió hallar contactos telefónicos, mensajes SMS y un historial de llamadas. La galería de imágenes proporcionaba abundante información sobre el propietario del teléfono y las personas de su entorno, así como escenarios frecuentados durante las actividades delictivas, todo ello atestiguado por metadatos correspondientes a fechas y coordenadas geográficas: un auténtico cofre del tesoro repleto de elementos de evidencia suficientes para comprometer al sospechoso y desmantelar su red criminal.

Animados por este éxito inicial, los investigadores quisieron averiguar si este dispositivo podía dar más de sí. Interesaba conocer la existencia de mensajes SMS borrados —algo que requiere un análisis de los espacios intersticiales en las bases de datos SQLite—, historiales de Internet y restos de correo electrónico en las cachés del dispositivo. Por suerte la versión de Android con la que este dispositivo móvil viene provisto de fábrica (2.2.2 Froyo) es vulnerable a un ataque de rooteado temporal que permite el acceso con privilegios de superusuario durante la sesión en curso, hasta que el terminal se desconecta o es reinicializado.

He aquí el procedimiento. En primer lugar se localiza en la página web *http://forum.xda-developers.com* un archivo llamado `SuperOneClick.zip`. Utilice el formulario de búsquedas para llegar hasta él. Una vez descomprimido en una carpeta de la estación de trabajo forense, cópielo al directorio donde está instalado el ejecutable ADB de Android SDK, por lo general en la carpeta `/home/#usuario#/`

`android-sdk-linux/platform-tools`, o bien —si trabaja con Windows— en `Archivos de programa\Android\android-sdk\platform-tools`.

El contenido de este archivo ZIP está compuesto por las herramientas siguientes: `su-v3`, `busybox`, `Superuser.apk`, `psneuter` y `GingerBreak`. Antes de conectar el terminal a la estación de trabajo debe estar activada la opción de **Depuración USB**. Una vez hecho este ajuste se cambia el nombre del ejecutable `su-v3` a `su` y se ejecutan los comandos siguientes:

```
igandekoa@igandekoa:~/android-sdk-linux/platform-tools$ adb devices
igandekoa@igandekoa:~/android-sdk-linux/platform-tools$ adb push psneuter /
data/local/tmp
igandekoa@igandekoa:~/android-sdk-linux/platform-tools$ adb shell
$ cd /data/local/tmp
$ chmod 777 psneuter
$ ./psneuter
```

A continuación el *exploit* se ejecuta cerrando el *shell*. Entonces reiniciamos el servidor:

```
igandekoa@igandekoa:~/android-sdk-linux/platform-tools$ adb kill-server
igandekoa@igandekoa:~/android-sdk-linux/platform-tools$ adb devices
```

Finalmente volvemos a abrir el *shell*:

```
igandekoa@igandekoa:~/android-sdk-linux/platform-tools$ adb shell
```

Sabremos que la operación ha tenido éxito si en lugar del *prompt* o indicador del usuario normal en forma de signo del dólar ($) aparece el indicador del `root` consistente en una almohadilla (#). Después de esto el investigador podrá moverse sin limitaciones por todo el árbol de directorios del terminal, visualizar la partición de datos y copiar a la estación de trabajo forense los archivos que le interesen mediante el comando `push`. También podrá realizar imágenes a bajo nivel de cualquiera de las particiones del dispositivo, incluyendo el espacio SD simulado en memoria. Para ello extraemos la tarjeta MicroSD —si el dispositivo incautado llevaba una, podrá ser adquirida sin dificultad por el procedimiento descrito en el apartado 2.1—.

A continuación insertamos otra tarjeta MicroSD que esté limpia de datos —preferiblemente formateada— y que tenga capacidad suficiente para contener la partición entera. Sobre ella grabaremos la imagen a bajo nivel realizada con `dd`. La partición de datos está alojada en la zona de memoria NAND correspondiente al dispositivo de bloque `mmcblk0p8`. Esto lo hemos averiguado con la ayuda del comando `mount`. Su punto de montaje es `/data/data`. Por consiguiente:

```
# cd /mount/sdcard/_ExternalSD
# dd if=/dev/block/mmcblk0p8 of=data.dd
```

Una vez terminada la operación, la imagen se copia al PC insertando la tarjeta MicroSD a través de un adaptador. También se puede trasladar directamente a la estación de trabajo con ADB. Cerramos el *shell* (comando "`exit`") y tecleamos:

```
igandekoa@igandekoa:~/android-sdk-linux/platform-tools$ adb pull
/mnt/sdcard/_ExternalSD/data.dd
```

3.7.7 Rooting permanente

Dependiendo de la marca, el modelo y la versión de Android, también se puede conseguir que el *rooting* sea permanente, es decir que la capacidad de manejar el dispositivo con privilegios de superusuario se mantenga tras la reinicialización del terminal. Esto no es algo que nos interese de modo particular. Lo que es bueno para un hacker no tiene por qué serlo para el investigador forense. Si ya tenemos lo que queríamos —una imagen de la partición de datos—, deberíamos dejar las cosas como están, ya que así podremos explicar delante de un juez que si hemos tomado la decisión de introducir cambios en la configuración del dispositivo era porque no había otro camino para llegar a los datos, y, en cualquier caso, se hizo todo lo posible para reducir al mínimo la alteración de la prueba.

Otras veces no será posible recurrir al rooteado temporal, porque el dispositivo no lo admite o porque los procedimientos no nos parecen fiables, o bien porque existe el riesgo de modificar la prueba en mayor medida que mediante la aplicación de técnicas de *rooting* permanente.

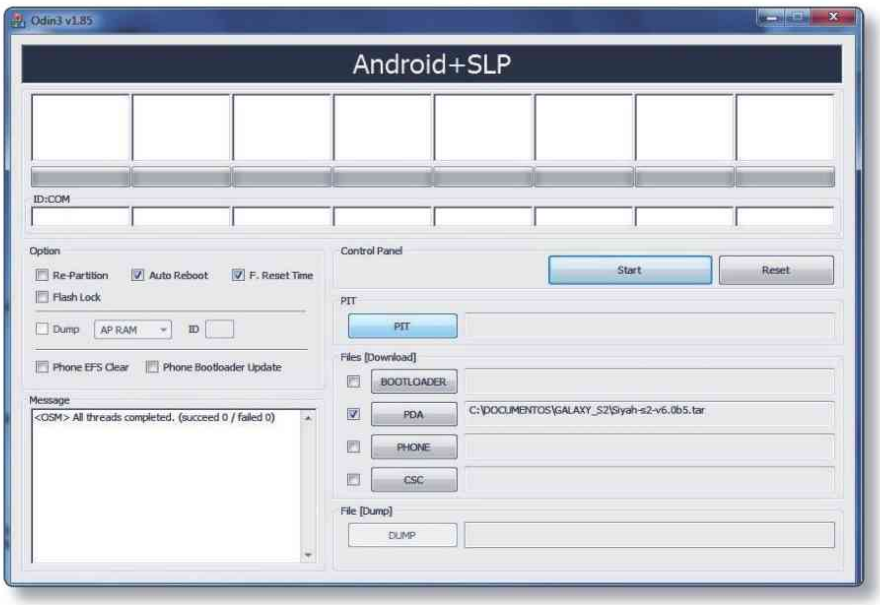

Figura 3.20. Interfaz de Odin

Supongamos, por ejemplo, que queremos rootear nuestro Samsung Galaxy Ace 2 del capítulo 1. Con este dispositivo no resulta posible utilizar `psneuter`, ya que la versión del sistema operativo (2.3 Gingerbread o superior) no es vulnerable al *exploit*. Existen otros procedimientos de *rooting* temporal, pero por lo que hemos podido averiguar sobre ellos no nos parecen lo bastante fiables. Por suerte para este terminal se dispone de métodos que permiten obtener un rooteado permanente en condiciones razonables de seguridad.

El procedimiento consiste en grabar en la NAND una imagen modificada de la partición Recovery, que incluye los binarios `su` y SuperUser. Este método tiene el inconveniente de que el *rooting* es definitivo y persiste tras el rearranque del teléfono, con lo cual produciremos de manera inevitable algunas modificaciones en el objeto de prueba.

No obstante también tiene sus ventajas: es seguro, fácil de realizar y está documentado. Las imágenes de software, los archivos de las aplicaciones y el programa flasheador Odin para dispositivos Samsung (Figura 3.20) se encuentran disponibles en la página web de XDA-Developers. En esta ocasión necesitamos trabajar desde Windows 7. El procedimiento consta de los pasos siguientes:

1. Antes de nada es preciso instalar en la estación de trabajo forense los *drivers* del terminal. Los podremos encontrar en la página de Internet del fabricante. Si en nuestro ordenador tenemos instalado el software de sincronización Samsung Kies, entonces no necesitamos hacer nada. Los *drivers* son imprescindibles. Sin ellos no será posible la conexión del dispositivo móvil al ordenador mediante USB.

2. Copiar el archivo update_su.zip a la memoria SD interna del terminal.

3. Estando el teléfono conectado mediante el cable USB, hay que apagarlo y encenderlo nuevamente manteniendo pulsadas las teclas que permiten entrar en el modo download o descarga (Volumen abajo + botón principal o home + botón de encendido). En Windows 7: iniciar Odin. Esperar a que Windows haya cargado el controlador y Odin reconozca el dispositivo.

4. Flashear la imagen I8160XXLD8_ready_to_root_ANT.tar.md5, seleccionándola en Odin dentro del recuadro PDA y pulsando después Start. Las casillas Re-Partition y Flash-Lock deben estar desmarcadas. Esperar a que la operación finalice con éxito, lo cual estará indicado por una señal de color verde en la parte superior izquierda de la ventana de Odin.

5. Apagar el teléfono y reiniciarlo en modo recovery o recuperación (Volumen arriba + botón principal o home + botón de encendido).

6. Seleccionar Apply update from sdcard (aplicar actualización desde SD interna) y seleccionar update_su.zip como archivo origen.

7. Apagar el teléfono y volver a encenderlo en modo de funcionamiento normal. Si el procedimiento ha sido aplicado de modo correcto, el terminal estará funcionando ahora en modo root.

Existe un procedimiento similar para eliminar el *rooting* y devolver el teléfono a su estado anterior. La diferencia reside en que la imagen flasheada con Odin en la partición Recovery es la original del dispositivo.

No existe razón que justifique la molestia de restablecer en un dispositivo móvil los ajustes de fábrica, a no ser que que sea necesario devolvérselo a su propietario y esto tenga que hacerse en el mismo estado en que aquel lo entregó. Para no empeorar el estado de modificación de la prueba, sin embargo, lo más aconsejable es dejarlo rooteado incluyendo en el informe las razones que llevaron a tomar esta medida y una explicación detallada del procedimiento.

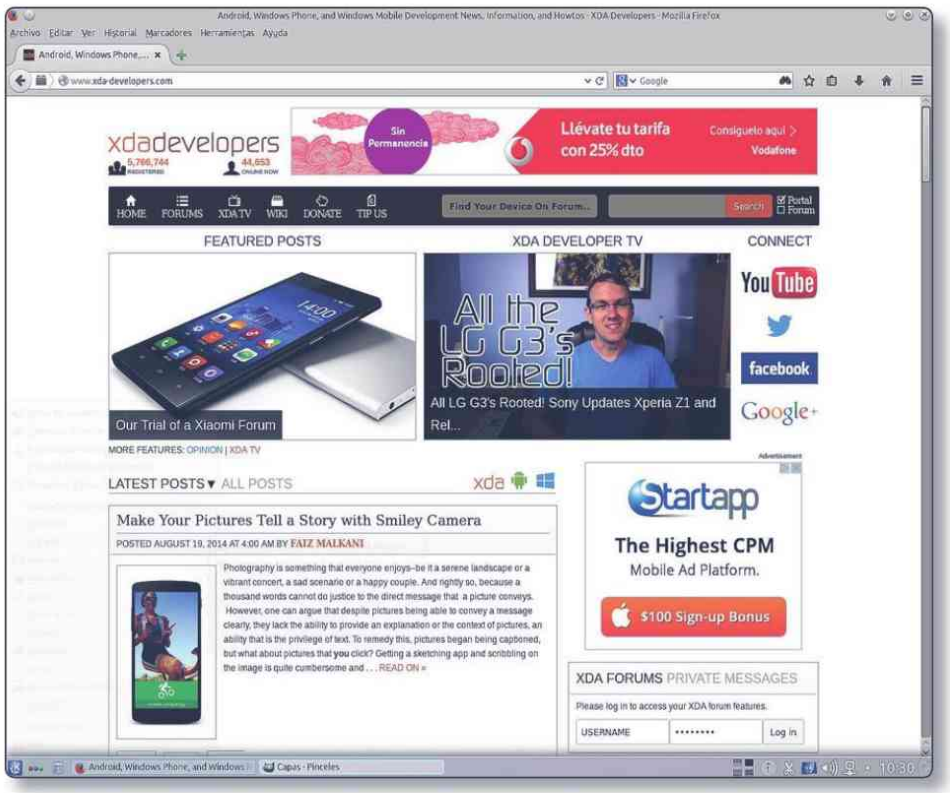

Figura 3.21. Página web de XDA-Developers

3.7.8 XDA-Developers

La investigación forense de dispositivos móviles es una especialidad en la que por fuerza de las circunstancias —abundancia de marcas y modelos, diversidad de aplicaciones técnicas, rápida sucesión de las versiones Android con sus API correspondientes y un rápido avance de la tecnología— se impone la necesidad de una formación continuada. Hay que hacer esfuerzos considerables para marchar al paso de los tiempos. En páginas web de aficionados y hackers existe información abundante sobre técnicas de *rooting*, ROM cocinadas, herramientas y otros temas de actualidad en la escena Android. Como referencia autorizada se recomienda la página de XDA-Developers: *http://www.xda-developers.com/*. XDA-Developers (XDA) es una comunidad de desarrolladores y aficionados en Internet. Este sitio fue habilitado en 2003 y en sus primeros tiempos no tenía que ver con Android —que por aquel tiempo ni siquiera existía—, sino que estaba dedicado a un dispositivo llamado O2 XDA con sistema operativo Windows Mobile del fabricante taiwanés High Tech Computer Corporation —en la actualidad más conocido como HTC—.

En aquellos días pioneros, hackers y usuarios experimentados se dieron cuenta de que el O2 XDA ofrecía unas prestaciones que podían ir mucho más allá de los estrechos límites establecidos por la configuración estándar del fabricante. Por este motivo abrieron un foro técnico destinado a explotar nuevas posibilidades técnicas del dispositivo, poniéndolas a disposición de la comunidad. Con el tiempo fueron abriéndose foros para dispositivos nuevos, y XDA creció como comunidad abierta de desarrolladores, en la que se reunía gente con las mismas aficiones y conocimientos técnicos relacionados con el mundo de la informática móvil y, más concretamente, Android. En la actualidad, XDA-Developers constituye la primera referencia para temas relacionados con la modificación de dispositivos móviles y el descubrimiento de *hacks* que permitan extender la funcionalidad de *smartphones* y tabletas equipados con el sistema operativo Android.

Los temas de discusión están organizados por foros. Para llegar hasta ellos basta introducir la marca y el modelo del dispositivo en un formulario de búsquedas. En XDA el investigador no solo podrá encontrar explicaciones detalladas sobre diversas cuestiones relacionadas con el *rooting* y la instalación de particiones Recovery; también hallará noticias sobre tecnología, nuevas versiones de Android, guías y tutoriales. También podrá consultar opiniones de expertos y dispondrá de enlaces a sitios de descarga de ROM cocinadas y herramientas necesarias para actualizar binarios, flashear imágenes y otras operaciones similares.

4

TÉCNICAS ESPECIALES DE INVESTIGACIÓN

Hasta el momento hemos visto los principios básicos en los que se basa la adquisición e investigación forense de dispositivos móviles. Teléfonos móviles, *smartphones* y tabletas fueron diseñados para una conexión permanente a la Red. Su mayor proximidad al usuario y su capacidad para contener datos personales hacen que sean difíciles de manejar como objetos de prueba. Resulta imposible cumplir el principio fundamental de la informática forenseinformática forense de tipo "clásico", es decir, realización de copias a bajo nivel (*bitstream* o flujo de bits) con el dispositivo apagado y de manera que no experimente cambios durante la adquisición. Mantener la cadena de custodia es problemático. En la etapa final del proceso, a la hora de redactar los informes y defenderlos ante el juez, el investigador forense se encontrará además con otras dificultades que deberá superar mediante la aplicación de procedimientos adecuados.

4.1 CADENA DE CUSTODIA

Los elementos de evidencia digital son algo frágil y efímero. Demostrar que pueden haber sido manipulados es tan fácil como descubrir el más mínimo hueco en la cadena de custodia. Un formulario mal rellenado, un dispositivo de bloqueo defectuoso o un *hash* que no coincide bastan para que la parte contraria solicite la anulación de la prueba con una probabilidad de éxito muy alta. No solo hay riesgo de alteración y pérdida de datos: también existe la posibilidad de que el sospechoso haya instalado aplicaciones capaces de eliminar datos que le incriminan, o de que pueda llevar a cabo un borrado remoto del dispositivo a través de Internet. Para garantizar el mantenimiento de la cadena de custodia existen procedimientos y

buenas prácticas. La amenaza de una actuación a distancia por parte del sospechoso se conjura aislando el dispositivo de las redes informáticas a las que está configurado para conectarse durante su uso habitual. Así mismo, antes de llevar a cabo cualquier tipo de manipulaciones al azar, es necesario saber qué hacer en caso de que el terminal esté protegido por códigos de acceso u otros medios de seguridad.

4.1.1 Aislamiento de redes

Supongamos que nos entregan un terminal incautado a un sospechoso. Un examen inicial del dispositivo revela que no está protegido por código. Al parecer el usuario no conoce los ajustes de seguridad de Android o no ha querido servirse de ellos por comodidad. Para acceder a las aplicaciones y los datos podemos desbloquear la pantalla con un movimiento de dedo. En tales condiciones podemos explorar sin problemas el contenido del terminal siguiendo el orden de prioridades descrito en el capítulo anterior: historial de llamadas, contactos telefónicos, mensajes SMS y de correo electrónico, etc. También podemos activar la depuración USB y ver el sistema de archivos del teléfono a través del *shell* ADB. Finalmente, si las circunstancias lo hacen necesario, podemos rootear el dispositivo y acceder a todos sus datos mediante las herramientas forenses habituales: `strings`, Scalpel, clientes SQLite y otras.

Figura 4.1. Bolsa de Faraday para aislamiento de dispositivos móviles

La vida del investigador forense no suele ser tan fácil. No sabemos si el usuario ha instalado una aplicación capaz de borrar los datos, mediante temporizador —al pasar un tiempo sin introducir algún código de seguridad—, o remotamente a

través de mensajes SMS o una conexión de Internet. Esta es precisamente una opción que Google ofrece al usuario para proteger su privacidad. No requiere conocimientos avanzados de informática y se puede habilitar a través de un formulario web. Por consiguiente, no debe descartarse la posibilidad de que un sospechoso recurra a este mecanismo de protección en caso de que el terminal caiga en poder de la policía. Para evitar que tal cosa suceda es preciso dejar el dispositivo móvil aislado de la red.

En teoría, cualquier cosa que sirva para bloquear transmisiones inalámbricas puede ser de ayuda. En la práctica, para separar un teléfono móvil de las redes de comunicaciones a las que se encuentra conectado —GSM/CDMA, Internet, dispositivos Bluetooth— se recurre a diversos procedimientos. Cada uno tiene ventajas e inconvenientes. He aquí los más utilizados:

- **Apagar el dispositivo**. Esta opción es efectiva al 100 %, al precio de producir un cambio substancial en el estado del teléfono: pérdida de toda la información volátil contenida por la RAM, además de la posibilidad de que el usuario haya activado un bloqueo por código al reiniciar el terminal.

- Introducir el teléfono dentro de una **jaula de Faraday**, caja metálica, bolsa blindada o similar (Figura 4.1). El propósito que se persigue es obstaculizar el flujo de ondas electromagnéticas entre el terminal y las torres de telefonía móvil o los puntos de acceso wifi. Es un método infalible, aunque plantea algunos problemas, como evitar que el dispositivo se apague una vez agotada la batería. No se puede hacer pasar el cable de alimentación a través de un hueco en la bolsa, ya que los hilos de cobre actúan como una guía de ondas anulando la función bloqueadora del receptáculo. Algunos laboratorios forenses disponen de salas aisladas electromagnéticamente. Este tipo de instalaciones son difíciles de construir y muy caras.

  Un problema que se presenta de manera inevitable es el siguiente: al perder la conexión, el dispositivo intenta restablecerla enviando señales en busca de torres de telefonía móvil y puntos de acceso wifi. Cada pulso de energía que emite es un pequeño tirón a la reserva de energía almacenada en la batería y hace que esta se agote antes de lo esperado. El terminal actúa de este modo cada vez que sus circuitos detectan dificultades en la conexión. En zonas con cobertura débil se pueden percibir los efectos de este esfuerzo por restablecer la conexión —sobre todo si lleva activado el acceso a Internet a través de 3G— en el modo en que se recalienta el terminal. Si la batería es de baja calidad, existe el riesgo de que el exceso de calor provoque daños en los chips del terminal. Se sabe de baterías

que han llegado incluso a explotar, inutilizando el terminal y provocando quemaduras al usuario.

- **Suspender la línea del usuario** en la compañía telefónica. Esta medida deja interrumpidas las comunicaciones de voz, el tráfico de mensajes SMS y el acceso a Internet a través de redes 3G, pero no las transmisiones wifi. Otro inconveniente reside en el tiempo que se necesita para obtener una orden judicial y notificarla al proveedor de acceso telefónico.

- **Quitar la tarjeta SIM**. Basta retirar la tarjeta SIM de un teléfono GSM para interrumpir su contacto con cualquier red de voz y datos 3G. Este remedio no es aplicable a dispositivos configurados para redes CDMA[7] (que no disponen de ranura para tarjeta SIM), y tampoco sirve para aislar el terminal del acceso a Internet por wifi. Además puede darse la circunstancia de que el teléfono sea de una marca y/o modelo tales que impida la extracción de la SIM sin tener que quitar la batería, lo cual obligaría a apagar el terminal.

- **Modo avión**. Este es el procedimiento más práctico. Normalmente se puede recurrir a él sin producir alteraciones significativas en el terminal. El modo avión, incluido desde hace años en teléfonos y dispositivos móviles de diversas marcas para facilitar su manejo a bordo de aviones, en hospitales, salas de conciertos y otros recintos donde el empleo de teléfonos móviles se considera perjudicial o molesto, deja totalmente interrumpidas las comunicaciones inalámbricas de todo tipo, sin que el terminal entre en rutinas de reconexión consumidoras de energía. En Android se activa mediante **Ajustes → Conexiones inalámbricas**, marcando la casilla correspondiente. También existe la posibilidad de activarlo manteniendo pulsado el botón **ON/OFF** del dispositivo durante un rato. Esto se puede hacer también sin quitar la pantalla de protección o el bloqueo por código (Figura 4.2). El modo avión implica algunos cambios menores en la configuración del dispositivo, pero es un precio muy bajo que podemos pagar sin problemas por los riesgos que conseguimos evitar.

7 Este problema puede presentarse en Estados Unidos y otros países del continente americano, no así en Europa, donde todas las redes telefónicas son GSM.

4.1.2 Terminales bloqueados

Otro problema se presenta cuando el usuario del terminal ha configurado un bloqueo de seguridad. Entonces no solo será imposible visualizar datos, aplicaciones y otros elementos de evidencia, sino que tampoco podremos conectar el dispositivo a un ordenador para sincronizar contenidos o hacer copias de seguridad. Existen procedimientos para acceder a dispositivos móviles protegidos por bloqueo. Cada uno de ellos supone riesgos técnicos y jurídicos. La posibilidad de sortear con éxito un código de bloqueo depende de las circunstancias, la marca y modelo del terminal, y también de si este ha sido rooteado o sometido a manipulaciones que puedan llegar a alterar los datos o poner en peligro la cadena de custodia.

Figura 4.2. Modo Avión

Existen tres formas básicas de proteger la privacidad en los dispositivos móviles Android. La configuración de estos diferentes sistemas se lleva a cabo desde el apartado **Ajustes → Ubicación y seguridad → Definir bloqueo de pantalla**.

- **Bloqueo mediante patrón**. Este sistema era utilizado por defecto en los primeros terminales Android. Consiste en definir un recorrido con los dedos o con un dispositivo táctil que una diversos puntos de modo continuo sobre una cuadrícula. El sistema deja paso libre a aquel usuario que conozca el patrón y sea capaz de trazarlo sobre la pantalla con un gesto dactilar adecuado (Figura 4.3-a).

- **Código PIN**. Similar al que se emplea en los cajeros automáticos de los bancos o para impedir el acceso a un teléfono móvil mediante código de cuatro dígitos (Figura 4.3-b). Este código no tiene por qué ser el mismo que se ha configurado previamente para la tarjeta SIM.

- **Contraseña**. El último tipo de bloqueo para dispositivos móviles Android es una contraseña alfanumérica parecida a las que utilizamos para acceder a ordenadores, redes informáticas, cuentas de correo electrónico o cualquier tipo de servicio *on line*.

(a) (b)

Figura 4.3. Bloqueos de patrón (a) y PIN (b)

En un dispositivo protegido mediante cualquiera de estos tres sistemas de bloqueo, nuestra investigación habrá llegado a un punto muerto si no somos capaces de hallar una manera que nos permita acceder al terminal sin conocer el código. De momento lo único que se puede hacer, como ya se ha dicho antes, es poner el terminal en modo avión, impidiendo así el acceso desde el exterior.

Para llegar a los datos es preciso disponer de conocimientos sobre seguridad Android. Los métodos que se describen a continuación no son los únicos: existen otros, y con el tiempo van surgiendo procedimientos diferentes basados en técnicas nuevas. Cada uno plantea dificultades y riesgos legales. Antes de hacer nada es necesario estar bien informado y actuar con prudencia.

4.1.3 Smudge attack

El ataque de suciedad (*smudge attack*) es una técnica ideada por un equipo de investigadores de la Facultad de Ciencias de la Computación de la Universidad de Pennsylvania. Está explicada en un documento disponible en Internet (Aviv, Gibson, Mossop, Blaze, Smith: *Smudge attacks on smartphone touch screens*; en español: *Ataques de suciedad en pantallas táctiles de teléfonos móviles inteligentes*). Consiste en examinar las marcas existentes sobre una pantalla táctil para descubrir trazas del patrón de bloqueo que el usuario ha utilizado para proteger su terminal.

El fundamento de esta técnica reside en la forma en que los residuos de grasa corporal y sudor mezclados con partículas de humo, suciedad ambiente y otras substancias, modifican las propiedades reflectantes de la pantalla. Como norma general, cuando está limpia, una superficie es reflectante y tiene una difusividad baja. A medida que la pantalla se ensucia con el uso, la reflectividad va disminuyendo mientras la difusividad aumenta. A través de iluminación oblicua y una selección de valores extremos en los ajustes de brillo y contraste de cualquier software de retoque fotográfico (por ejemplo Adobe Photoshop o GIMP), a menudo resulta posible descubrir trazas del patrón de bloqueo (Figura 4.4).

En este caso, más que técnicas de *hacking*, se aplican métodos basados en el proceso digital de imágenes. El éxito del procedimiento requiere evitar todo contacto manual con los dispositivos incautados. Es buena idea acudir al escenario de los hechos provisto de bolsas para recogida de pruebas y de un lápiz capacitivo con punta de goma para realizar las manipulaciones más urgentes (como por ejemplo activar el modo avión), antes de trasladar el dispositivo al laboratorio en el que se van a hacer las fotografías digitales.

Figura 4.4. Smudge attack o "ataque de suciedad"

4.1.4 Prueba de conexión ADB

Aunque esté protegido por bloqueo —PIN, patrón o contraseña—, en caso de tener habilitada la depuración USB el dispositivo será accesible a través de Android SDK. Conectando el terminal a su estación de trabajo, el investigador puede establecer un puente ADB y examinar los contenidos del teléfono mediante un *shell*. Para comprobar si la depuración USB está habilitada en terminales protegidos por bloqueo, lo único que hay que hacer es conectar el dispositivo a nuestra estación de trabajo y, desde una consola de texto abierta en el directorio correspondiente de nuestra instalación SDK, ejecutar el comando:

```
igandekoa@igandekoa:~/android-sdk-linux/platform-tools$ adb devices
```

Si tenemos suerte y la depuración USB está activada, ADB devolverá una salida con códigos de serie o cualquier otro tipo de información textual representativa del dispositivo móvil:

```
List of devices attached
4696C2228D44BF63489A743B485C8EC device
```

En tales circunstancias, al menos una parte de los contenidos del terminal estará disponible para ser adquirida por medios forenses. Si además el dispositivo está rooteado, entonces habremos tenido mucha suerte. No hace falta decir, sin embargo, que la depuración USB activa en un teléfono —y no digamos el *rooting*— no es un caso que se presente con frecuencia. En ocasiones es esperable este escenario, por motivos como los que se indican a continuación:

- El usuario es un programador profesional o aficionado con experiencia que dedica parte de su tiempo a desarrollar y probar aplicaciones móviles.
- El usuario tuvo en algún momento la intención de de rootear su dispositivo móvil, y después de llevar a cabo varios intentos, exitosos o fallidos, ha olvidado desmarcar la casilla **Depuración USB** en el apartado **Ajustes**.
- Alguien ha instalado particiones Recovery alternativas mediante herramientas de flasheado.
- Instalación de ROM cocinadas.
- *Hacking* de teléfonos móviles.
- O bien, simplemente, porque determinadas aplicaciones requieren el activado previo de la depuración USB.

4.1.5 Ejemplos

Pudimos comprobar la conectividad ADB en los dos teléfonos móviles que se han utilizado como ejemplo en el capítulo anterior —un Samsung Galaxy Ace 2 y un LG Optimus 2X—, tal y como se encontraban en el momento en que fueron puestos a disposición para las prácticas. Previamente se configuró en ambos un bloqueo de pantalla por código PIN. Acto seguido, cada uno de elllos fue conectado a nuestra estación de trabajo forense, con SDK Android instalado sobre Linux Kubuntu 13.04, y se intentó establecer conexión mediante ADB.

El Galaxy Ace 2 había pertenecido a un usuario experto aficionado a la escena del *modding*, las ROM cocinadas y el *hacking* ético. No fue una sorpresa descubrir que el dispositivo estaba rooteado. Aparentemente, su propietario lo mantenía en este estado para facilitar la manipulación del dispositivo y la grabación de imágenes mediante programas como Odin o Heimdall. Al estar habilitada la depuración USB no hubo ningún problema para abrir el *shell* e ingresar al sistema en modo superusuario (indicado por un *prompt* con el cuadradillo [#] en lugar del signo del dólar [$]). El acceso a la memoria NAND era total, incluyendo la partición de datos del usuario.

El LG Optimus 2X, por el contrario, no tenía la depuración USB habilitada, por lo que fue imposible abrirse camino hasta su interior por este método. Como contraprueba activamos la depuración USB en el apartado de **Ajustes** y volvimos a establecer conexión a través del cable. Esta vez SDK reconoció el dispositivo y pudo abrir un *shell* con acceso a una parte restringida del sistema de archivos del terminal, por no hallarse este rooteado. En cualquier dispositivo de estas características provisto de sistema operativo con una versión inferior a Android 2.3 Gingerbread, el estado activo de la **Depuración USB** permite el acceso a través de ADB y, como consecuencia de ello, un rooteado temporal mediante `psneuter` y el procedimiento descrito en 3.7.6.

4.1.6 Eliminación del código de bloqueo

Si se dispone de acceso a través de ADB, también existe la posibilidad de eliminar el bloqueo borrando los archivos que contienen los *hashes* del patrón o la contraseña/PIN. Esto produce cambios en la configuración del dispositivo, y además implica el riesgo de dejar inutilizado el sistema. Obviamente operaciones de este tipo, aunque resulten de gran utilidad a un desarrollador de software, no son prioritarias en una investigación forense. Lo que sigue es únicamente para ampliar conocimientos en este apartado técnico o para tenerlo en cuenta si se sospecha que alguien ha tenido acceso no autorizado a un teléfono móvil y es necesario explicarlo en el informe.

Los códigos de bloqueo se guardan en dos archivos cuyas rutas en el árbol de directorios Android son `/data/system/gesture.key` (para patrón) y `/data/`

`system/password.key` (para PIN o contraseña). Estos archivos pertenecen al administrador del sistema: para modificarlos se necesita que el dispositivo esté rooteado. El procedimiento es simple y consiste en abrir un *shell* ADB, borrar el archivo correspondiente y reemplazarlo por otro vacío con el mismo nombre. Por ejemplo, para eliminar un bloqueo por patrón teclearíamos lo siguiente en una consola abierta mediante ADB:

```
# rm gesture.key
# touch gesture.key
```

Y de manera análoga, para quitar la contraseña o el PIN:

```
# rm password.key
# touch password.key
```

4.2 PARTICIONES RECOVERY

El acceso a terminales protegidos por códigos PIN, contraseña o patrones que no tengan activada la depuración USB ni estén rooteados requiere métodos imaginativos y no exentos de riesgo. Por el momento estos nuevos métodos están siendo objeto de debate, pero cabe suponer que con el tiempo se irán abriendo camino ante la necesidad de contar con herramientas de investigación y análisis en un escenario de actuaciones forenses caracterizado por la abundancia de marcas, modelos y plataformas de software, así como por la inexistencia de estándares y por un entorno legal garantista que no avanza a la misma velocidad que la tecnología. Para entender los procedimientos de los que se va a hablar en las páginas siguientes es necesario adquirir antes algunas nociones sobre el esquema de particionado de memoria NAND en los dispositivos Android.

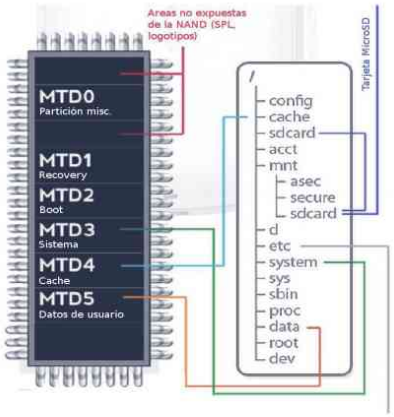

Figura 4.5. Esquema general de particionado NAND

La instalación de particiones Recovery implica el empleo de técnicas intrusivas que producen cambios en la configuración del dispositivo, aunque por lo general estas modificaciones afectan al sistema operativo y no a aquella área del terminal que constiyuye el objetivo de la investigación forense: la partición de datos del usuario. Es por ello que pese a incumplir el principio maestro de la informática forense —no alteración de elementos que puedan servir como prueba— el uso de estas técnicas de investigación debería estar autorizado cuando no exista otro medio de llegar hasta los elementos de evidencia electrónica.

4.2.1 Particionado NAND

Las memorias NAND (fabricadas a base de puertas lógicas NO-Y) que se utilizan para guardar el software del sistema, aplicaciones y datos del usuario en *smartphones*, tabletas, reproductores multimedia y otros artefactos portátiles, son dispositivos de almacenamiento similares a los discos duros de estado sólido o las llaves USB. No hay que confundirlas con la memoria RAM, aunque en la mayor parte de los chips modernos ambos tipos de memoria, NAND y RAM, vienen integradas en un mismo componente tipo sándwich.

Las arquitecturas de base de un dispositivo de estado sólido y de una memoria NAND son las mismas. Sin embargo, en los dispositivos Android, a diferencia de los *pendrives* USB o las tarjetas de almacenamiento para cámaras digitales, la memoria no dispone de controlador integrado en el chip de memoria. Es el sistema operativo el que debe gestionar la memoria mediante software. Por esta razón resulta imposible leer este tipo de soportes de datos con el teléfono apagado —salvo excepciones basadas en técnicas avanzadas de extracción mediante equipos especiales de hardware que luego se explicarán—.

```
rootfs on / type rootfs (ro,noatime,nodiratime)
proc on /proc type proc (rw,noatime,nodiratime)
sysfs on /sys type sysfs (rw,noatime,nodiratime)
tmpfs on /dev type tmpfs (rw,nosuid,relatime,mode=755)
devpts on /dev/pts type devpts (rw,relatime,mode=600)
tmpfs on /mnt/asec type tmpfs (rw,relatime,mode=755,gid=1000)
tmpfs on /mnt/obb type tmpfs (rw,relatime,mode=755,gid=1000)
none on /dev/cpuctl type cgroup (rw,relatime,cpu)
/dev/block/mmcblk0p9 on /system type ext4 (ro,relatime,barrier=1,data=ordered)
/dev/block/mmcblk0p7 on /cache type ext4
(rw,nosuid,nodev,noatime,errors=panic,barrier=1,journal_async_commit,data=ordered)
/dev/block/mmcblk0p1 on /efs type ext4
(rw,nosuid,nodev,noatime,errors=panic,barrier=1,journal_async_commit,data=ordered)
```

```
/dev/block/mmcblk0p12 on /preload type ext4
(ro,nosuid,nodev,noatime,barrier=1,data=ordered)
/dev/block/mmcblk0p10 on /data type ext4 (rw,nosuid,nodev,noatime,errors=panic,
barrier=1,journal_async_commit,data=ordered,noauto_da_alloc,discard)
/dev/block/mmcblk0p4 on /mnt/.lfs type j4fs (rw,noatime,nodiratime)
/sys/kernel/debug on /sys/kernel/debug type debugfs (rw,noatime,nodiratime)
tmpfs on /mnt/ntfs type tmpfs (rw,relatime,mode=777,gid=1000)
/dev/block/vold/259:3 on /storage/sdcard0 type vfat (rw,dirsync,nosuid,nodev,
noexec,noatime,nodiratime,uid=1000,gid=1015,fmask=0002,dmask=0002,
allow_utime=0020,codepage=cp437,iocharset=iso8859-1,shortname=
mixed,utf8,errors=remount-ro,discard)
/dev/block/vold/179:25 on /storage/extSdCard type vfat (rw,dirsync,nosuid,nodev,
noexec,noatime,nodiratime,uid=1000,gid=1023,fmask=0002,dmask=0002,
allow_utime=0020,codepage=cp437,iocharset=iso8859-1,shortname=
mixed,utf8,errors=remount-ro)
/dev/block/vold/179:25 on /mnt/secure/asec type vfat (rw,dirsync,nosuid,nodev,
noexec,noatime,nodiratime,uid=1000,gid=1023,fmask=0002,dmask=
0002,allow_utime=0020,codepage=cp437,iocharset=iso8859-1,shortname=
mixed,utf8,errors=remount-ro)
tmpfs on /storage/extSdCard/.android_secure type tmpfs
(ro,relatime,size=0k,mode=000)
```

Listado 4.1. Salida del comando adb shell mount

En un esquema típico de particionado NAND para dispositivos Android (Figura 4.5) el espacio de almacenamiento está dividido en un número de áreas, cada una de ellas correspondiente a un apartado funcional del sistema: `boot`, `recovery`, `system`, `userdata`, `sdcard`, `cache`, `efs`, etc. Este espacio se ve ampliado con la capacidad de almacenamiento de las tarjetas de memoria SD insertables. La disposición exacta y el tamaño de las particiones dependen de la marca, el modelo y la versión de Android.

El sistema operativo gestiona las particiones mediante un controlador de software o MTD (*memory technology device*) que emula dispositivos de bloques a partir de los sectores de borrado característicos de los soportes de almacenamiento NAND. Los archivos correspondientes a estos dispositivos se encuentran alojados en el directorio `/dev`. Cada uno de ellos dispone de un punto de montaje en la jerarquía de directorios Linux, como se puede ver en los listados 4.1 y 4.2, generados por la ejecución de los comandos `mount` y `df` a través de una consola ADB.

```
Filesystem                1K-blocks  Used    Available  Use%  Mounted on
tmpfs                     400500     76      400424     0%    /dev
tmpfs                     400500     0       400500     0%    /mnt/asec
tmpfs                     400500     0       400500     0%    /mnt/obb
/dev/block/mmcblk0p9      516040     515992  48         100%  /system
/dev/block/mmcblk0p7      100784     4156    96628      4%    /cache
/dev/block/mmcblk0p1      20144      8460    11684      42%   /efs
/dev/block/mmcblk0p12     516040     431340  84700      84%   /preload
/dev/block/mmcblk0p10     2064208    794808  1269400    39%   /data
df: /mnt/.lfs: Function not implemented
tmpfs                     400500     0       400500     0%    /mnt/ntfs
/dev/block/vold/259:3     12063840   176576  11887264   1%    /storage/sdcard0
/dev/block/vold/179:25    1954560    754720  1199840    39%   /storage/extSdCard
/dev/block/vold/179:25    1954560    754720  1199840    39%   /mnt/secure/asec
```

Listado 4.2. Salida del comando adb shell df

En los ejemplos anteriores, la ejecución de los comandos `adb` se ha llevado a cabo sobre un dispositivo móvil Samsung Galaxy S2. Al investigador forense le interesan principalmente aquellas zonas en las que se almacenan los datos de usuario y las aplicaciones instaladas por aquel, mediante descarga del Play Store o de otros sitios. También es necesario conocer el contenido de las tarjetas externas. Aunque hay diferencias determinadas por la marca, el modelo de terminal y la versión de Android, la información buscada por el investigador forense se halla en las ubicaciones siguientes:

- **`/data/data`**: información del usuario y utilidades con ajustes y datos de configuración, alojados en directorios que corresponden a cada una de las aplicaciones. En esta partición es donde se encuentra la mayor parte de los elementos de evidencia potenciales: contactos telefónicos, historiales de llamadas e Internet, mensajes SMS y de correo electrónico, etc. Este directorio, en el ejemplo que antecede, es el punto de montaje de la partición contenida en el dispositivo `/dev/block/mmcblk0p10`.

- **`/sdcard`**: tarjeta SD interna emulada en la memoria principal NAND del dispositivo, donde se suelen guardar documentos, archivos multimedia, fotografías y vídeos realizados con la cámara, contenedores APK de las aplicaciones descargadas y otros elementos. Este directorio, en el ejemplo propuesto, es el punto de montaje del dispositivo `/dev/block/vold/259:3`.

- **`/sdcard/external_sd`**, **`/sdcard/sdcard`** —la denominación difiere de acuerdo con la marca, el modelo y la versión de Android, pero en cualquier caso es similar a la que aquí se indica—: este es el directorio de montaje de la tarjeta externa, utilizada por el usuario para ampliar la capacidad de su terminal Android, pudiendo guardar en ella cualquier tipo de contenidos. La tarjeta externa, como ya se ha visto, puede ser adquirida a través de un adaptador. Este directorio, en los listados 4.1 y 4.2, corresponde al dispositivo `/dev/block/vold/179:25`.

Cuando se investigan casos de software malicioso interesa también llevar a cabo un estudio de la partición del sistema operativo para analizar sus contenidos y compararlos con el software de un terminal no manipulado. En tales circunstancias, y en el ejemplo que nos ocupa, la partición que debe ser adquirida por medios forenses se encuentra alojada en el dispositivo `/dev/block/mmcblk0p10`.

Las particiones están formateadas con diferentes sistemas de archivos. Android reconoce gran cantidad de ellos, aunque en la práctica solo utiliza tres o cuatro. Como se aprecia en los listados, y por las razones de compatibilidad mencionadas en el capítulo 1, `sdcard` y `sdcard/extSdCard` están formateadas con sistemas de archivos VFAT.

En la época en que los primeros dispositivos Android salieron al mercado, los sistemas de archivos habituales para las particiones de sistema y datos de usuario eran YAFFS (*yet another flash file system*) y YAFFS2, desarrollados para interactuar de manera óptima con memorias de estado sólido. Este sistema de archivos aún se puede encontrar en terminales antiguos. En la actualidad, sin embargo, ha caído en desuso debido a problemas de compatibilidad con las CPU de núcleos múltiples, siendo reemplazado en la mayor parte de los dispositivos modernos por el ext4 estándar de Linux.

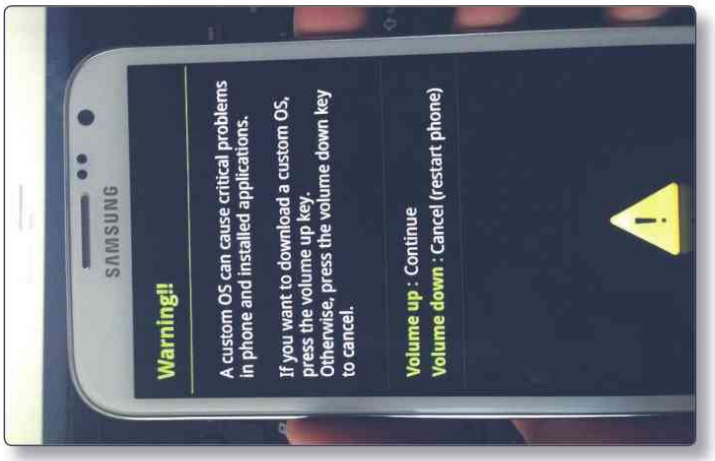

Figura 4.6. Pantalla inicial típica del modo download en Android

4.2.2 Modos de funcionamiento

Los dispositivos Android disponen de tres modos básicos de funcionamiento:

1. **Modo normal**. Se inicia tras la activación del botón de encendido y la carga del sistema operativo. En este modo tiene lugar el uso convencional de un *smartphone* o una tableta por el usuario, con todas sus aplicaciones y servicios de conexión a redes de voz y de datos. Este es el modo en el que el dispositivo pasa la mayor parte de su vida útil, y en la práctica el único que conocen la mayor parte de los usuarios.

2. **Modo download o de descarga**. Si el dispositivo es arrancado en modo *download*, la carga del sistema operativo queda suspendida y el terminal permanece en espera de que alguien, a través de la conexión USB, modifique su configuración interna, transfiera actualizaciones del *firmware*, herramientas software, imágenes de particiones completas o incluso un nuevo sistema operativo (ROM cocinadas) (Figura 4.6).

3. **Modo recovery**. El arranque del dispositivo en modo *recovery* permite ejecutar una versión reducida del sistema operativo con un conjunto de herramientas y opciones para tareas de mantenimiento del terminal, recuperación de ajustes de fábrica o realización de copias de respaldo de las aplicaciones del sistema y los datos del usuario. Pronto se verá la importancia que tiene este modo para los fines de la investigación forense de dispositivos móviles Android (Figura 4.7).

Figura 4.7. Smartphone con recovery de fábrica. Opciones limitadas

Los métodos de acceso a los modos de funcionamiento *download* y *recovery* varían en función de la marca y el modelo de dispositivo Android. Para ello existen determinadas combinaciones de botones, teclas y mandos de volumen. En la mayor parte de los casos la operación se lleva a cabo con el terminal apagado. Por ejemplo, pulsando simultaneamente el botón de inicio (**Power**), el mando de volumen (**Bajar**) y el botón de base (**Home**), se consigue iniciar el modo *download*. La misma combinación, pero con el mando de volumen en la posición de subir, inicia el modo *recovery*.

En dispositivos que carecen de un botón mecánico para llamar a la pantalla de inicio —como por ejemplo en modelos de la marca LG—, la combinación de botones se reduce a **Power** + **Volumen abajo** (*download*) / **Power** + **Volumen arriba** (*recovery*). Para estar seguro es preciso consultar el manual de instrucciones del fabricante del terminal.

4.2.3 Secuencia de arranque

Android es Linux, y no solo posee la misma estructura básica que este sistema operativo de código libre, sino también unas características de funcionamiento similares, entre ellas la forma de iniciarse desde que se activa el botón de encendido hasta que el terminal se halla listo para admitir comandos y ejecutar aplicaciones. Conocer la secuencia de arranque de Android ayuda a entender el proceso de recuperación de datos basado en el empleo de particiones Recovery modificadas.

Cuando un dispositivo apagado comienza a funcionar, lo primero que hace es inicializar el hardware y ejecutar un código grabado en la ROM. Este código solo hace dos cosas: localizar el medio de almacenamiento de datos (memoria NAND) donde se encuentra el software de arranque del sistema operativo, y, una vez encontrado, entregarle el control. A continuación el software de arranque (*boot loader*) es copiado a la memoria RAM y desde ese momento se hace cargo de todo lo demás. Esta rutina es en todo similar a la que se verifica en el inicio de otros sistemas operativos como OSX o Windows.

Los cargadores típicos, como Linux/GRUB o el que viene integrado en cualquier versión de Windows, permiten al usuario seleccionar el sistema operativo con el que quiere arrancar, en caso de que haya más de uno instalado. En Android el software de carga consta de dos partes: un cargador inicial de programa (Initial Program Loader [IPL]), que habilita la RAM para el proceso de arranque y ejecución del sistema operativo, y un cargador secundario (Second Program Loader [SPL]). El IPL copia el SPL a la memoria RAM y le transfiere el control. Entonces el SPL transfiere a la RAM el sistema operativo Android y activa funciones de arranque

alternativo como `fastboot`, `recovery` o cualquier otro recurso que pudiera resultar necesario para fines de configuración, mantenimiento o recuperación del dispositivo.

Etapa esencial en este proceso es la inicialización del *kernel* de Linux una vez que este ha sido transferido a la RAM. Será entonces cuando comience a funcionar el sistema operativo propiamente dicho, ya que el *kernel* es responsable de regular todas las funciones a bajo nivel, administrar procesos del sistema y permitir o denegar recursos a las aplicaciones que los soliciten, teniendo en cuenta el nivel de privilegios de cada una de ellas.

Para funcionar en estas etapas iniciales de ejecución, el *kernel* necesita un número de módulos que le permitan acceder al hardware y ejecutar funciones básicas, como montar sistemas de archivos, poner en marcha los procesos iniciales del sistema y otras por el estilo. Como el *kernel* tiene que arrancar de algún modo, con controladores o sin ellos, nos encontramos ante el típico dilema del huevo o la gallina. Los desarrolladores lo resuelven mediante la carga en memoria —efectuada con anterioridad por el SPL— de una imagen que contiene un sistema de archivos en miniatura junto con los módulos necesarios para la puesta en funcionamiento del *kernel* y otras utilidades básicas.

Inmediatamente después se ejecutan los *scripts* o guiones de comandos encargados de montar los sistemas de archivos característicos de Linux. El sistema provisional contenido en el disco RAM que se creó al cargar la imagen de inicio es desmontado y, finalmente, el *kernel* ejecuta el proceso padre (`Init`), que a su vez pone en marcha los restantes procesos del sistema, así como las aplicaciones, según vayan siendo solicitados por el usuario o por otros programas. En Android la secuencia culmina con la puesta en marcha del proceso Zygote, el cual finalmente habilita la máquina virtual Dalvik dejándola lista para ejecutar aplicaciones de usuario compiladas en código dex.

Para el investigador forense estas imágenes iniciales que carga el sistema en el momento del arranque tienen un gran interés. En las distribuciones antiguas de Linux tanto la imagen como el sistema de archivos provisional estaban contenidos en el archivo `initrd`. Desde la versión 2.6 del *kernel* se utiliza el más eficiente sistema `initramfs`. Es ahí donde el fabricante del terminal aloja las utilidades que permiten realizar funciones de mantenimiento y recuperación durante el arranque del dispositivo en modo *recovery*. Cuando arrancamos Android en modo *recovery*, el terminal se queda estacionado en `initramfs` sin montar los sistemas de archivo normales de Linux ni iniciar el proceso `Init`.

Por lo general estas imágenes de fábrica son bastante limitadas: únicamente incluyen la capacidad de borrar cachés, ejecutar archivos de actualización o restablecer ajustes de fábrica. Si se consigue modificar esta imagen añadiendo

herramientas que permitan una adquisición forense de los datos del usuario mediante la realización de *backups* o el acceso directo a las particiones a través de un *daemon* `adb` con privilegios de superusuario, entonces dispondremos de una vía para acceder a la práctica totalidad de los datos del dispositivo.

La nueva imagen, provista de herramientas de adquisición forense, ha de ser inyectada en la partición Recovery de la memoria NAND mediante una utilidad de grabación a bajo nivel como `fastboot`, Heimdall u Odin —esta última para terminales Samsung Galaxy—. Posteriormente se arranca el terminal en modo *recovery*, ejecutándose tan solo el *kernel* y las aplicaciones de la imagen modificada. Al no iniciarse en modo normal, los bloqueos por código no aparecerán y se dispondrá de acceso ilimitado al contenido de la memoria NAND.

Inevitablemente, esto implica una alteración del soporte de datos y, por lo tanto, de la prueba. A cambio de vulnerar este principio de la ciencia forense —si bien de manera limitada y por razones justificadas, que deben explicarse convenientemente en caso de recurrir a los procedimientos que se describen algo más adelante— dispondremos de acceso total a los datos del dispositivo móvil, aunque esté bloqueado por código o no tenga habilitada la depuración USB.

Queda por ver si el procedimiento podría ser admisible ante cualquier tribunal. En cualquier caso incumbe a la autoridad judicial resolver sobre la procedencia del mismo en función de factores como la ausencia de alternativas, la gravedad del caso o el valor económico de los intereses disputados. Teniendo en cuenta estas difíciles consideraciones es por lo que se ha escrito el apartado de este libro correspondiente a la instalación de particiones Recovery.

En cualquier caso, los cambios afectan únicamentge a la partición Recovery procedente de fábrica, la cual se pierde al grabar sobre ella la nueva imagen. El resto de las particiones —sistema, caché, `efs`, datos de usuario— no se ven afectadas.

4.2.4 ClockWorkMod

Elaborar una imagen modificada de la partición Recovery es algo que en principio está al alcance de cualquiera. Lo único que se necesita es una plataforma Linux, un editor de texto, algunas aplicaciones fácilmente localizables en páginas de desarrolladores y software flasheador como Odin o Heimdall. Todo este material es código libre y se halla disponible gratuitamente en Internet. Sin embargo, para trabajar con él se requieren conocimientos técnicos avanzados y experiencia con la línea de comando. Las dificultades y el riesgo de dejar inutilizado el terminal pueden disuadir al investigador. Por fortuna existen imágenes Recovery elaboradas por la comunidad de desarrolladores Android que sirven para alcanzar el objetivo que nos hemos propuesto.

Tales imágenes no fueron creadas para fines forenses, sino educativos y experimentales. Pese a ello resultan de gran de utilidad para el investigador. Basta localizar —preferiblemente a través de XDA-Developers— una de estas imágenes que sirva para la marca, modelo y sistema operativo del terminal investigado, y que disponga de utilidades que permitan realizar una copia de respaldo de la partición de datos del usuario en la tarjeta de memoria externa SD.

Una de estas particiones Recovery modificadas es ClockWorkMod (CWM), creada por Koushik Dutta y disponible en *http://www.clockworkmod.com/* junto con otras aplicaciones convenientes para mantenimiento y *hacking* de sistemas Android. CWM soporta instalación *on line*, verificando previamente su compatibilidad con la marca y el modelo de terminal. En sus versiones recientes, además de las opciones habituales —borrado de cachés, restablecimiento de ajustes de fábrica, etc.—, dispone de utilidades que permiten realizar un *backup* completo de las particiones de sistema y de datos (Figura 4.8).

CWM también incluye un *daemon* ADB que se ejecuta con privilegios de `root` y varios programas de sistema Linux —`dd`, `nandump`, `cpio`, etc.— integrados en un único ejecutable denominado `busybox`, con los cuales se puede obtener una imagen a bajo nivel de las particiones del sistema, volcándola sobre una tarjeta SD o trasladándola a la estación de trabajo del investigador forense.

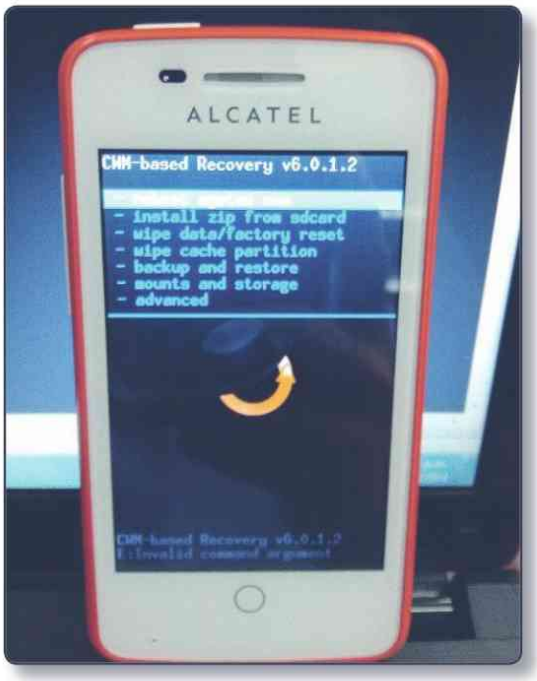

Figura 4.8. Recovery CWM con opciones de backup y restore

4.2.5 Instalación de CWM en un Samsung Galaxy S2

La manipulación de particiones Recovery requiere prudencia. No solo se está modificando la configuración del terminal: también existe el riesgo de dejarlo temporalmente inutilizado. Esto obligaría a ejecutar operaciones adicionales tales como un restablecimiento de ajustes de fábrica, o la búsqueda e instalación de una ROM original del fabricante del dispositivo, lo cual supondría alteraciones adicionales en el objeto de prueba, con las complicaciones jurídicas que son de esperar. Si el caso es importante, se recomienda hacer con anterioridad pruebas sobre un terminal de la misma marca y modelo que el incautado al sospechoso.

Obviamente, CWM no es la única partición Recovery que existe; en caso necesario, sobre todo si la marca y el modelo de terminal lo requieren, se puede recurrir a otras, como TWRP (Figura 4.9). También hay distintas formas de instalar CWM dependiendo del modelo del terminal y/o de la versión de Android. No nos resulta posible ocuparnos aquí de todas las variantes. Sin embargo, hemos de comenzar por algún sitio. Nuestro prototipo, en este caso, es un Samsung Galaxy S2 con Android 4.1.2 Jelly Bean. La imagen que necesitamos ha sido localizada a través de Google y los foros de XDA-Developers. La operación debe llevarse a cabo en un PC de sobremesa o portátil con Windows, puesto que hacen falta la utilidad de flasheado Odin —propietaria de Samsung— y los *drivers* del dispositivo, descargables desde la página de soporte de Samsung o incluidos en el software de sincronización Kies. También podríamos utilizar el programa Heimdall, con posibilidad de elegir entre interfaz gráfica (en Windows) y línea de comando en las versiones para Linux.

Para trasladar ClockWorkMod al dispositivo móvil es necesario seguir los pasos siguientes:

1. Reiniciar el terminal en modo *download*, apagándolo primero y encendiéndolo a continuación con los botones **Principal** + **Bajar volumen** + **Power** pulsados de manera simultánea. La pantalla mostrará la opción de reiniciar el terminal (**Volumen abajo**) o iniciar descarga (**Volumen arriba**).

2. Seleccionar esta última opción.

3. Conectar el teléfono al ordenador a través del cable USB.

4. Arrancar Odin. Si estamos en Vista o Windows 7, quizás sea necesario ejecutar la aplicación como administrador. El recuadro alargado de la parte superior izquierda de Odin se volverá amarillo, lo que indica que el dispositivo ha sido detectado.

5. Seleccionar el archivo de imagen en la casilla **PDA** (**Files/Download**).

6. Hay que asegurarse de que las opciones **Flash-Lock** y **Re-Partition** se encuentran desmarcadas. Manteniendo marcadas las opciones **Auto Reboot** y **F. Reset Time**, pulsar sobre **Start** y esperar a que termine el proceso, lo cual será indicado con una luz verde (PASS).

En ese momento tendrá lugar un reinicio automático del dispositivo en modo normal. Para utilizar el modo *recovery* no hay más que apagarlo de nuevo y volver a encenderlo con la combinación de botones siguiente: **Principal** + **Volumen alto** + **Power**.

Si la operación se ha completado con éxito, al final se podrá ver, en lugar del *recovery* minimalista del fabricante, la interfaz de ClockWorkMod con su logotipo característico y un completo menú de opciones que nos permitirá manipular el dispositivo del modo que nos interesa.

Figura 4.9. Partición Recovery TWRP

4.2.6 Adquisición lógica con CWM

Lo expuesto en estos apartados no constituye una lista de instrucciones válida para cualquier *smartphone* o *tablet* provisto de Android. El propósito del autor consiste en lograr que los investigadores forenses entiendan el funcionamiento y los peligros de las tecnologías subyacentes. Queda a discreción de cada cual el gestionar oportunidades y riesgos de una manera adecuada, así como hacerse cargo de la responsabilidad relativa al desarrollo de estrategias para una adquisición de

datos limpia y jurídicamente admisible. El acceso a terminales mediante particiones Recovery modificadas es un asunto delicado que ha de resolverse con prudencia y sentido común para evitar que la integridad de la prueba y la cadena de custodia se vean comprometidas.

Tras haber hecho estas advertencias, he aquí un ejemplo de adquisición lógica de los datos del dispositivo a través de ClockWorkMod. Como se ha explicado antes, no importa que el terminal esté protegido con bloqueos de patrón, contraseña o PIN, o que no tenga habilitada la depuración USB. Una vez instalado CWM, y con el teléfono apagado, el procedimiento sería este:

1. Con la ayuda de otro dispositivo Android formateamos una tarjeta MiniSD de 32 GB de capacidad y la insertamos en el dispositivo investigado. La tarjeta también puede formatearse directamente desde ClockWorkMod. La tarjeta SD que venía insertada en el lateral del dispositivo ha sido previamente retirada para realizar una adquisición directa y un análisis de la partición de datos FAT32 por medios convencionales: adaptador USB y `dd` en un PC con Linux Ubuntu.

2. Encendemos el dispositivo en modo *recovery* apretando de manera simultánea las teclas **Principal** + **Volumen Arriba** + **Power**. Mantenemos estos mandos pulsados hasta que aparezca en pantalla el logotipo de ClockWorkMod (sombrero sobre una flecha curvada). Insistir lo que haga falta para conseguirlo.

3. Mediante el mando de volumen nos desplazamos hasta llegar a la opción de menú (**Backup and restore**) y pulsamos.

4. Esto nos llevará hasta un nuevo menú compuesto por tres opciones: **Copia de seguridad** ("- backup"), **Restablecimiento de ajustes** ("- restore") y **Recuperación avanzada** ("- advanced restore"). Seleccionamos "**- backup**".

5. Conviene estar seguros de lo que se hace antes de pulsar nada, porque la copia de respaldo comienza sin pedir confirmación. Esperamos hasta que terminen de extraerse todos los datos.

6. Una vez finalizada la operación, hay que regresar al menú principal pulsando el botón **Power**. Desde ahí podemos apagar el dispositivo, reiniciarlo en modo normal o dejarlo estacionado de momento en modo *recovery*, algo de todos modos recomendable, por si aún hiciera falta llevar a cabo una exploración detallada del dispositivo a través de ADB.

CWM realiza un *backup* completo de las particiones de sistema, caché y datos de usuario. El proceso puede durar algún tiempo a causa del tamaño de los archivos comprimidos que han de copiarse. Por ello resulta conveniente vigilar el nivel de carga de la batería (es preferible que esté a más de un 70 % en el momento de comenzar el trabajo) o mantener el terminal conectado a una fuente de alimentación.

Aunque los datos se guardan por defecto en la tarjeta externa SD, CWM puede elegir como punto de destino el espacio SD interno del dispositivo, con el peligro de que la adquisición fracase por falta de espacio o se sobreescriba un espacio del cual más adelante hubiéramos podido extraer evidencia útil en forma de archivos borrados por el usuario. Esto puede suceder —aun disponiendo de espacio suficiente— cuando la tarjeta externa MicroSD contiene archivos multimedia o datos de cualquier otro tipo. De ahí la necesidad de formatearla antes de proceder a la adquisición forense. Para estar seguros de que CWM trabaja como queremos es recomendable realizar antes pruebas con un terminal de la misma marca y modelo.

La copia de respaldo queda guardada en un directorio denominado *backup*, y dentro de este en una carpeta que en el momento de crearse recibe un nombre descriptivo de la fecha y la hora en que se llevó a cabo la operación. El contenido de esta carpeta, por lo general, consistirá en tres archivos comprimidos en el formato **.tar*, típico de Linux, y uno de texto:

- `system.ext4.tar`: partición del sistema.
- `data.ext4.tar`: partición de datos del usuario.
- `cache.ext4.tar`: partición de caché.
- `nandroid.md5`: sumas de verificación md5 de los archivos **.tar*.

Estos archivos **.tar* pueden ser descomprimidos y analizados con las utilidades del sistema y las herramientas de investigación forense habituales.

4.2.7 Adquisición física con CWM

La adquisición lógica de un soporte de datos incluye tan solo aquello que está referenciado explícitamente por el sistema de archivos, sin incluir los archivos borrados ni el espacio de almacenamiento no utilizado. En estas zonas podría haber información correspondiente a archivos antiguos o a un formateado anterior del medio. En realidad, una adquisición lógica no se diferencia de una copia de respaldo realizada por el software de sincronización del dispositivo o cualquier otra utilidad de *backup*.

En el supuesto de necesitar algo más que los archivos accesibles por el sistema, o si la investigación comprende entre sus objetivos la recuperación de material borrado, o el adquirir una partición completa (sistema, datos o caché),

incluyendo los sectores no asignados por el sistema de archivos, también es posible llevar a cabo adquisiciones físicas con CWM. Esto es posible porque la partición Recovery, al iniciarse, ejecuta un *daemon* `adbd` con privilegios de superusuario; y también porque el investigador, gracias a `busybox`, dispone de las herramientas necesarias para realizar copias a bajo nivel.

Para realizar una adquisición física es preciso emplear herramientas de Linux. He aquí los pasos:

1. Inicio del dispositivo móvil en modo *recovery* con los botones **Principal** + **Volumen arriba** + **Power**.

2. Conexión a la estación de trabajo forense a través del cable USB. En este ejemplo estamos trabajando con Linux Ubuntu, pero también puede hacerse desde Windows o Apple OSX.

3. Con la consola de comandos nos situamos en el subdirectorio de Android SDK donde se encuentre el ejecutable del cliente `adb` (en el caso presente: `/home/<nombre de usuario>/android-sdk-linux/platform-tools`), o bien en nuestra carpeta de trabajo si tenemos configuradas las variables de entorno del sistema para que `adb` pueda ser lanzado desde cualquier ubicación.

4. Comprobamos la conexión mediante el comando siguiente:

```
igandekoa@igandekoa:~/android-sdk-linux/platform-tools$ adb devices
```

   Si la instalación de CWM es correcta, deberá mostrarse en pantalla la misma salida que cuando se tiene activada la depuración USB con el dispositivo funcionando en modo normal:

```
List of devices attached
4696C2228D44BF63489A743B485C8EC device
```

5. Aunque el terminal esté bloqueado por código, el investigador dispone de acceso ilimitado, puesto que Android no llega a arrancar en su modo de funcionamiento normal y por lo tanto no se ejecutan las rutinas de protección. Para comprobarlo, el investigador no tiene más que abrir un *shell* y comprobar que puede trasladarse sin restricciones a través del árbol de directorios del dispositivo móvil:

```
igandekoa@igandekoa:~/android-sdk-linux/platform-tools$ adb shell
~ #
```

   El *prompt* nos indica que disponemos de un *shell* en modo `root`.

6. Para realizar una adquisición física de la partición de datos primero tenemos que ir a nuestra tarjeta SD para adquisiciones forenses, que, previamente formateada, debe hallarse ya insertada en el dispositivo y tener capacidad suficiente para admitir un archivo de imagen generado por el volcado de la partición completa que pensamos adquirir, generalmente la de datos de usuario. Cuidado a la hora de seleccionar la tarjeta: hablamos de un flujo de datos de unos cuantos gigabytes. Si la partición tiene un tamaño superior a 4 GB, entonces no será posible adquirirla con `dd`, puesto que el medio de destino está formateado con un sistema de archivos FAT32 y el tamaño máximo de archivo que este admite es, precisamente, 4 GB.

```
~ # cd sdcard
```

   El investigador debe decidir sus necesidades de espacio en función del dispositivo y el tamaño de memoria NAND que aquel lleva instalada. A continuación listamos las particiones del dispositivo. Esto nos permite averiguar cómo se llama el dispositivo de bloques que utilizaremos como origen de datos para el volcado de la imagen.

```
~ # mount
```

   Si en el listado no figura el dispositivo correspondiente a la partición de datos es porque ClockWorkMod no la ha montado durante el arranque. Esto lo puede hacer el investigador a través de la interfaz CWM. Sin embargo, debido a que en los terminales modernos la partición de datos de usuario suele estar formateada con un sistema de archivos ext4, al igual que las de sistema y caché, existe el riesgo de que el sistema de *journaling* altere mínimamente la partición al ser montada. Para evitar cambios es aconsejable adquirir la partición sin haberla montado previamente. Lo podemos hacer aunque no veamos el dispositivo de bloques, ya que por otros teléfonos de la misma marca y modelo conocemos la denominación del archivo correspondiente. Tan solo queda generar la imagen a bajo nivel estableciendo como destino para `dd` un archivo creado en la tarjeta externa SD:

```
~ # dd if=/dev/block/mmcblk0p5 of=imagen_data.dd
```

7. Traslado del archivo de imagen obtenido con `dd` a la estación de trabajo forense. Para esto el investigador puede extraer la tarjeta del dispositivo móvil y copiar su contenido directamente a través de un adaptador, o bien volcar la imagen directamente con `adb`:

```
~ # exit
igandekoa@igandekoa:~/android-sdk-linux/platform-tools$ adb pull
/sdcard/imagen_data.dd
```

Esta imagen a bajo nivel podrá ser examinada con las herramientas forenses habituales del investigador, o sometida a una operación de *data carving* en busca de archivos borrados, utilizando para ello Testdisk, Foremost, Scalpel, o cualquier otro software de código libre o propietario.

4.2.8 Adquisición física mediante ADB

La imagen a bajo nivel con `dd` también puede realizarse directamente a través de `adb` sin necesidad de utilizar una tarjeta externa SD insertada en el dispositivo. Esta posibilidad resulta de interés en el caso al que antes nos referíamos de las particiones con un tamaño superior a 4 GB. Como guardarlas en una tarjeta formateada con un sistema de archivos FAT32 resulta imposible —a no ser que la imagen sea cortada en fragmentos durante el proceso de adquisición—, la única opción consistirá en transferirla al disco duro de la estación de trabajo, formateado con ext3/4. Para ello nos servimos de las posibilidades de redireccionamiento de los comandos Linux mediante el operador "|":

```
igandekoa@igandekoa:~/android-sdk-linux/platform-tools$ adb shell dd
if=/dev/block/mmcblk0p5 | of=imagen_data.dd
```

Esta combinación de comandos establece un conducto a través del cual los datos recogidos por el `dd` local del dispositivo son canalizados hasta el `dd` que funciona en la estación de trabajo forense, el cual a su vez los deposita en un archivo de imagen a bajo nivel.

Se trata de una solución simple y elegante, que, llegado el caso, puede ayudar al investigador forense a resolver las dificultades derivadas de la imposibilidad de utilizar tarjetas SD, bien por daños en el hardware, porque el terminal solamente dispone de SD VFAT emulada en NAND o por el problema de los tamaños máximos de archivo en particiones FAT32 anteriormente mencionado.

4.3 ADQUISICIÓN POR HARDWARE

Existen situaciones en las que la adquisición por medios y herramientas convencionales resulta totalmente inviable, debido a daños de hardware o a otras causas. Entonces, la única alternativa que queda es intentar acceder a los contenidos del chip de memoria NAND a través de técnicas que implican el desmantelamiento físico del dispositivo. Debido a la complejidad técnica de los procedimientos, al riesgo de dañar los objetos de prueba y al elevado coste de los equipos de hardware que se necesitan, estas técnicas no están al alcance de cualquiera. Por lo general se recurre a ellas en la investigación de casos criminales graves, situaciones de amenaza contra la seguridad interior o espionaje industrial, cuando la importancia o el valor de los intereses económicos en litigio es muy elevado.

4.3.1 Chip-off

En el enfoque gradual que hemos seguido al exponer nuestra estrategia de adquisición de dispositivos móviles, progresando desde una simple exploración visual a métodos cada vez más intrusivos, que deben emplearse en función de las circunstancias y la gravedad de las diferentes situaciones, el *chip-off* es sin duda el procedimiento más agresivo, puesto que consiste en extraer físicamente el circuito de memoria NAND para analizarlo con equipos de hardware especiales. Esta técnica no solo permite investigar terminales dañados. También invalida cualquier bloqueo de seguridad basado en patrones o códigos.

Se trata de un método destructivo, que trae consigo cambios substanciales e irreversibles en el estado del dispositivo. En la práctica, después de una extracción física del chip, resulta imposible reparar el terminal y volver a dejarlo en condiciones de funcionamiento. Existen diversos procedimientos para separar el chip NAND de la placa madre del dispositivo. Se puede utilizar un soldador de precisión y pinzas para disipar el exceso de calor, evitando de este modo que sufran daño las conexiones internas y los circuitos de silicio. El componente puede ser así mismo extraído mediante succión por vacío después de haber lanzado un chorro de aire muy caliente para ablandar los conectores. Todos estos procesos son críticos y requieren de un control preciso para no quemar el chip.

Finalmente, el componente electrónico es conectado a un aparato de lectura que permite obtener una imagen a bajo nivel de los contenidos. El resultado de este proceso es similar a la realización de una adquisición física con herramientas de software. Los riesgos son mucho mayores debido a la posibilidad de destruir el chip. También son elevados los costes, ya que para llevar a cabo estas operaciones se requiere hardware especial, salas limpias y la ayuda de personal especializado en electrónica de precisión.

4.3.2 JTAG

JTAG (Joint Test Action Group) es un estándar desarrollado en los años ochenta para verificar circuitos impresos y componentes electrónicos de acuerdo con la norma IEEE 1149.1. La idea consiste en crear, mediante conexiones eléctricas con las patillas del circuito, una puerta trasera que los técnicos puedan utilizar para comprobar el funcionamiento del componente y depurar aplicaciones en ROM. Esto incluye la posibilidad de leer los contenidos de un circuito de memoria sin extraerlo de la placa. Los componentes diseñados conforme al estándar JTAG disponen, a través de sus patillas de conexión, de funciones de lectura y escritura, sincronización de reloj y selección de modos operativos.

La extracción mediante JTAG resulta complicada, al requerir equipos especiales y personal experimentado en comprobación de circuitos. También existe el riesgo de producir daños en el objeto de la prueba, ya que la lectura del chip requiere soldaduras provisionales. Equivocarse de patilla o aplicar voltajes incorrectos trae consigo un riesgo de destrucción del chip y con ello de perder todos los elementos de evidencia digital contenidos en el dispositivo móvil.

5

ANÁLISIS FORENSE

En la investigación forense de ordenadores de sobremesa y soportes de datos convencionales (discos duros, CD/DVD, cintas de *backup*, etc.), el procedimiento de análisis por lo general se lleva a cabo mediante determinadas herramientas de software (EnCase Forensics, FTK, The Sleuth Kit, utilidades de recuperación de archivos borrados, etc.) en un proceso de caracterización de archivos y búsqueda de información que pueda ser reveladora de actividades irregulares o delictivas. En Android, el análisis forense no requiere herramientas particularmente sofisticadas. Nuestro arsenal está compuesto por editores hexadecimales, clientes de bases de datos, visores de archivos XML y otras utilidades del mundo Linux/Unix tales como `strings`, que permite localizar líneas de texto en el interior de archivos binarios, o `file`, capaz de identificar tipos de archivo evitando que nos engañen mediante un cambio de extensión. Para llevar a cabo un análisis correcto de los elementos de evidencia es preciso conocer el modo en que la información es almacenada, gestionada y eliminada en Android.

5.1 ESTRUCTURAS DE DATOS

5.1.1 Aplicaciones

Los dispositivos móviles Android acumulan grandes cantidades de información. La fuente primaria de dicha información son las aplicaciones, que también funcionan como agente fundamental en todo proceso de gestión y almacenamiento de datos, al menos desde la perspectiva del usuario. Cada una de estas aplicaciones consiste en un programa escrito en Java y traducido a código especial optimizado para ejecución en la máquina virtual Dalvik.

Existen diferentes tipos de aplicaciones Android. Entre las más corrientes cabe mencionar:

- Aplicaciones del sistema.
- Software preinstalado por el fabricante del dispositivo.
- Utilidades preinstaladas por la compañía telefónica.
- Aplicaciones instaladas por el usuario mediante descarga de sitios oficiales de Google: en una primera época Android Market y Google Play en la actualidad.
- Aplicaciones procedentes de sitios no oficiales e instaladas por el usuario mediante descarga o directamente desde el ordenador.
- Aplicaciones desarrolladas por el usuario, en caso de que este sea un programador.

5.1.2 Datos de usuario

Los datos del usuario se generan como resultado de una interacción directa con las aplicaciones del dispositivo móvil. Al igual que en el análisis tradicional de discos duros y ordenadores de sobremesa, estos datos constituyen el objetivo principal del investigador. Esto no quiere decir que las propias aplicaciones no sean importantes como objeto de investigación, sobre todo en casos de software malicioso, espionaje industrial o ciberguerra. Por lo que respecta a la información relacionada con la actividad del usuario, en un dispositivo móvil Android interesan los elementos de evidencia que se indican a continuación:

- Contactos telefónicos.
- Historial de llamadas entrantes y salientes.
- Mensajes de texto (SMS y MMS).
- Mensajes de correo electrónico (Gmail, Yahoo, cuentas facilitadas por el proveedor de acceso telefónico).
- Mensajes WhatsApp.
- Fotografías y vídeos realizados por el usuario.
- Archivos multimedia (películas, canciones, etc.).
- Enlaces favoritos guardados en el navegador de Internet.
- Historial de navegación de Internet.
- Historial de consultas en Google y otros buscadores.
- Coordenadas geográficas.
- Rutas consultadas en el GPS.
- Datos geográficos de Google Maps.
- Entradas en Facebook, Twitter, Linkedin, Xing y otras redes sociales.
- Archivos almacenados en el dispositivo.
- Archivos procedentes de programas de descarga y *file sharing*.

- Información bursátil, datos de banca *on line* y servicios financieros.
- Compras en eBay, Amazon y otros sitios.
- Notas de texto.
- Listas de actividades y anotaciones en el calendario.

5.1.3 Métodos de archivado

En un PC de sobremesa los procedimientos para guardar información dependen de las aplicaciones utilizadas, pero en general son configurables con un amplio margen de flexibilidad. Es el propio usuario el que decide dónde se guardan los datos, estableciendo para ello ubicaciones arbitrarias además de las carpetas predeterminadas del sistema. Android, debido a las limitaciones típicas de las plataformas móviles y a requerimientos de seguridad más estrictos, funciona en este aspecto con un mayor grado de rigidez, determinado por el automatismo de las API del sistema. En la tarjeta de expansión MicroSD, el usuario puede hacer lo que quiera; no así en las restantes zonas del terminal, donde es Android quien decide dónde y de qué modo se guardan las cosas.

El almacenamiento persistente de aquellos datos que han de permanecer disponibles tras un apagado del sistema, sin que se pierdan al ser borrados de la RAM, se lleva a cabo en la memoria NAND del dispositivo móvil, que en esta función resulta del todo similar a un disco duro o a un soporte de datos regrabable. Existen sin embargo algunas diferencias importantes con respecto a discos duros y llaves USB, como tuvimos ocasión de explicar en el capítulo anterior. Para archivar y gestionar datos, Android utiliza uno de los cinco métodos siguientes, dependiendo del tipo de información del que se trate:

1. Almacenamiento interno.
2. Preferencias compartidas (*shared preferences*).
3. Bases de datos SQLite.
4. Almacenamiento externo.
5. Almacenamiento en red (o en la nube).

El almacenamiento interno es administrado por las API del sistema, y en general comprende el segundo y el tercer método (preferencias compartidas y bases de datos SQLite), así como otros más específicos definidos por el desarrollador para el funcionamiento de su aplicación. Pueden emplearse formatos especiales, pero nunca fuera de la zona del sistema de archivos reservada por Android para la aplicación correspondiente. Por el contrario, el almacenamiento externo (en tarjeta SD o FAT simulada en NAND) no plantea requisitos tan estrictos. Tanto el desarrollador como el usuario pueden crear las estructuras de datos y los árboles de directorios que necesiten, incluso hacer cambios en la disposición de los mismos, copiando carpetas, moviéndolas o eliminándolas según convenga.

Finalmente, el almacenamiento en Red es el que se lleva a cabo a través de utilidades de *backup*, con destino a la nube o mediante servicios de carpetas compartidas en Internet como Dropbox, Free Hidrive y otros parecidos.

5.1.4 Jerarquía de directorios Linux

Android funciona sobre Linux. Y siendo Linux un sistema operativo con más de cuatro décadas de existencia —si incluimos a su predecesor Unix— resulta esperable que la funcionalidad básica de nuestro terminal móvil gravite sobre la estructura de datos más importante del mundo Unix: la jerarquía de directorios estándar, con su sistema de permisos y su diseño basado en la abstracción de recursos. En el terminal móvil, el usuario ve sus datos en cómodos recuadros que puede manipular de manera fácil e intuitiva a través de una pantalla táctil. Pero realmente la información se encuentra alojada dentro de bases de datos y archivos XML en algún lugar del árbol de directorios Linux.

Algunas aplicaciones, principalmente navegadores de archivos, nos permiten ver parte del árbol de directorios Linux dispuesto en forma de carpetas amarillas sobre un fondo negro al mejor estilo de las interfaces gráficas Windows o Apple OSX. Estas utilidades solo muestran las partes accesibles al usuario en la tarjeta MicroSD o el espacio FAT simulado en memoria NAND. Como norma general, la libertad de movimiento queda restringida a aquellas zonas en las que se guardan las imágenes tomadas con la cámara del terminal o los archivos multimedia descargados de Internet.

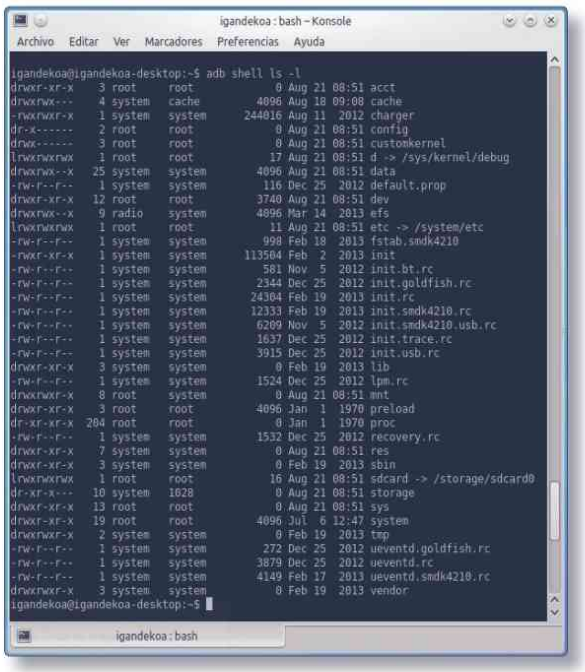

Figura 5.1. Estructura de directorios Android en un Samsung Galaxy S2

Para examinar árboles de directorios es necesario establecer una conexión ADB. Esto se puede hacer a través del cable USB o mediante un enlace inalámbrico. En un dispositivo rooteado podremos acceder con permisos de superusuario a todos los directorios. Abriendo un *shell*, o con los comandos `pull` y `push`, que permiten copiar y mover archivos del terminal a la estación de trabajo forense y viceversa, haremos y desharemos a capricho: podremos extraer archivos del terminal, trasladar a este herramientas necesarias para un rooteado temporal, instalar aplicaciones o realizar imágenes forenses de particiones de la memoria NAND.

Con el comando `ls` se muestran en la raíz del sistema algunos directorios característicos de la jerarquía estándar de Linux (Figura 5.1), como `/dev`, `/bin`, `/mnt`, `/var`, `/sbin`, o `/lib`. Las funciones y cometidos que se les asignan en Android son generalmente los mismos. La disposición suele experimentar ligeras variaciones según se trate de una versión u otra de Android. Cada fabricante tiene su libro de especificaciones, pero el principio fundamental es el mismo.

5.1.5 Datos de usuario

Si buscamos información sobre el propietario del dispositivo móvil interesan sobre todo dos ubicaciones: el punto de montaje de la tarjeta SD (o del espacio FAT simulado en NAND) y la partición de datos del usuario. Estos directorios existen en todos los sistemas Android, independientemente del nivel de API y de las configuraciones del fabricante. He aquí la localización habitual:

Espacio de almacenamiento	Punto de montaje	Sistema de archivos
SD simulada en NAND	/mnt/sdcard	vfat
SD externa	/mnt/sdcard/external_sd	vfat
Datos del usuario	/data/data	ext4

En las dos primeras ubicaciones la disposición de archivos es arbitraria y el usuario puede establecer su propio esquema de carpetas. Generalmente, estas zonas se utilizan para almacenar archivos multimedia liberando espacio en la partición de datos del usuario mediante el traslado a la tarjeta SD de parte de las aplicaciones descargadas de Google Play o de otros sitios. Por el contrario, en la partición de datos del usuario la disposición de directorios y archivos sigue un esquema más rígido, determinado por las API del sistema, sin que el usuario pueda hacer gran cosa para modificarlo.

Como norma general existe un directorio para cada aplicación, ya sea esta de sistema o instalada por el usuario. Dentro de este directorio encontraremos otros que contienen los componentes de la aplicación: ejecutables, parámetros de configuración en archivos XML, recursos de la aplicación (gráficos, mapas, mensajes

de texto, documentación) y, finalmente, la materia prima que el investigador estaba buscando: información y elementos de evidencia, alojados convenientemente dentro de archivos de bases de datos SQLite.

Veamos un ejemplo. Abrimos un *shell* en un terminal rooteado y pasamos a superusuario mediante el comando `su`. A continuación nos situamos sobre el directorio de la partición de datos de usuario correspondiente al navegador web (`./com.android.browser`) y examinamos sus contenidos. Lo primero que vamos a ver es el árbol de subdirectorios de la aplicación:

```
igandekoa@igandekoa:/$ adb shell
$ su
# cd /data/data/com.android.browser
# ls -l
drwxr-xr-x system   system        2013-05-14 09:18 lib
drwxrwx--x app_91   app_91        2013-09-18 21:46 databases
drwxrwx--x app_91   app_91        2013-05-14 10:04 app_appcache
drwxrwx--x app_91   app_91        2013-05-14 10:04 app_databases
drwxrwx--x app_91   app_91        2013-05-14 10:06 app_geolocation
drwxrwx--x app_91   app_91        2013-09-10 21:49 shared_prefs
drwxrwx--x app_91   app_91        2013-05-14 10:04 app_thumbnails
drwxrwx--x app_91   app_91        2013-08-23 17:15 app_icons
drwxrwx--x app_91   app_91        2013-05-14 14:16 cache
```

Observe la presencia de algunos directorios correspondientes a los métodos de almacenamiento expuestos en el apartado 5.1.3: preferencias compartidas (`./shared_prefs`), donde se encuentran los archivos XML que contienen los parámetros de configuración de las aplicaciones, y la carpeta para las bases de datos SQLite (`./databases`). Estas ubicaciones y algunas otras (como `./cache` y `./lib`) son estándar y nos las encontraremos en casi todas las aplicaciones. Algunas de las denominaciones son autoexplicativas. Examinando el contenido de los archivos se puede llegar a saber lo que el sospechoso ha estado haciendo con la aplicación.

```
# ls -l ./shared_prefs
-rw-rw---- app_91   app_91   479 2013-09-10 21:49 com.android.browser_preferences.xml
-rw-rw---- app_91   app_91   119 2013-05-14 10:06 BrowserBookmarksPage.xml
-rw-rw---- app_91   app_91   118 2013-07-08 21:17 WebViewSettings.xml
```

O bien:

```
# ls -l ./databases
-rw-rw---- app_91   app_91   69632 2013-07-10 16:42 webview.db
-rw-rw---- app_91   app_91   428512 2013-09-10 21:53 webview.db-wal
-rw-rw---- app_91   app_91   32768 2013-09-12 21:49 webview.db-shm
-rw-rw---- app_91   app_91   282624 2013-08-23 17:15 webviewCache.db
```

```
-rw-rw---- app_91  app_91  527392 2013-09-10 21:53 webviewCache.db-wal
-rw-rw---- app_91  app_91  32768 2013-09-12 21:49 webviewCache.db-shm
-rw-rw---- app_91  app_91  28672 2013-06-26 21:56 browser.db
-rw-rw---- app_91  app_91  412032 2013-09-10 21:53 browser.db-wal
-rw-rw---- app_91  app_91  32768 2013-09-10 21:53 browser.db-shm
-rw-rw---- app_91  app_91  16384 2013-05-14 09:22 launcher.db
```

5.1.6 Formatos

En la adquisición forense de ordenadores de sobremesa y soportes de datos convencionales se emplean técnicas analíticas directas: fotografías, archivos multimedia y documentos son abiertos con las aplicaciones correspondientes —visores gráficos, reproductores de medios, suites ofimáticas— y examinados en busca de elementos de evidencia explícitos y de metadatos; los archivos de código se cargan en entornos aislados —máquinas virtuales, ordenadores aislados de la red o entornos de desarrollo seguros— para ser examinados en busca de instrucciones o cadenas de texto características de aplicaciones maliciosas. Se recuperan los archivos borrados con herramientas de software y se llevan a cabo búsquedas de cadenas de texto y expresiones regulares con las herramientas forenses que se utilizan habitualmente para estos cometidos, como por ejemplo `strings`.

Con el material procedente de un dispositivo móvil se puede hacer lo mismo, pero esto no es solo una parte del trabajo. El empleo de formatos de datos estructurados por parte del desarrollador de aplicaciones exige procedimientos de trabajo específicos y focalizados. ¿Cómo hacemos para llegar a los datos del usuario? ¿Cómo realizar las operaciones de búsqueda y filtrado correspondiente? ¿Cómo extraer los elementos de evidencia de un modo adecuado para incluirlos más tarde en el informe forense? Para dar respuesta a todas estas cuestiones es necesario adquirir nociones básicas de XML y bases de datos SQLite.

5.2 XML

5.2.1 ¿Qué es XML?

XML es un lenguaje de marcas desarrollado por el consorcio W3W, la misma entidad que creó el HTML y la World Wide Web en los años noventa, para satisfacer una necesidad de la que adolecían las primitivas páginas de Internet. El marcado HTML, con su conjunto de etiquetas básico para definir párrafos e insertar recursos —tablas, imágenes, elementos multimedia— resulta ideal para dar formato

a un documento, pero no es el más adecuado para el procesamiento mecánico de la información. Una cosa es mostrar páginas web en pantalla, con texto, imágenes y algún efecto animado de carácter básico; otra muy distinta es conseguir que sean realmente interactivas y convertirlas en herramientas eficaces para la gestión de todo tipo de servicios de productividad.

Basándose en sus propios trabajos de desarrollo con etiquetas de texto, evolucionados a partir de proyectos pioneros de IBM y otras empresas durante la década de 1960, los técnicos del WWW se dieron cuenta de que un lenguaje de marcas solo puede llegar a ser eficaz si antes es rediseñado para incluir marcas que definan no solamente el aspecto visual de las páginas web, sino también la definición, tipología, clasificación y destino de los datos que la componen. Solo así es posible desarrollar tecnologías que sirvan para crear páginas interactivas, redes sociales, aplicaciones de inteligencia artificial, incluso una web semántica capaz de gestionar no solo contenidos sino también el significado de los mismos.

Para evitar incompatibilidades con el estándar HTML, XML fue definido como un lenguaje de marcado de tipo general para todo tipo de caracterizaciones de datos, incluyendo al propio HTML como caso particular compuesto por un subconjunto de las etiquetas disponibles en XML. Aunque desde este punto de vista ambos lenguajes son lo mismo, entre XML y HTML existen notables diferencias. Una de ellas consiste en que mientras el conjunto de etiquetas de HTML viene definido en las especificaciones de la versión correspondiente de este lenguaje de marcas, en XML es posible crear etiquetas nuevas en función de las necesidades de presentación y proceso de datos.

El éxito de XML se hace visible no solo en la creación de nuevas tecnologías orientadas a la Red —páginas dinámicas, servicios web, nuevos conceptos de interacción con el usuario como el XHTML, etc.—, sino en la aparición de un rango de aplicaciones prácticamente ilimitado para escenarios que requieren el manejo de grandes cantidades de información no destinada al consumo humano, sino dirigida a sistemas de proceso automatizados. XML es ampliamente utilizado en sistemas operativos para teléfonos y dispositivos móviles. Con XML se pueden configurar aplicaciones informáticas, máquinas-herramienta, dispositivos domóticos e incluso electrodomésticos. También hace posible el desarrollo de bases de datos directamente integradas en aplicaciones de Internet sin el intermediario de un sistema convencional servidor-cliente.

5.2.2 Manipulación de archivos XML

Aunque los archivos XML están escritos en texto plano, su lectura no resulta fácil debido a la abundancia de etiquetas de apertura y cierre y otros elementos

necesarios para definir el formato de un modo que resulte eficaz para los procesos a los que está destinado. Esto en el uso habitual no tiene importancia porque XML no fue desarrollado para una lectura por usuarios humanos sino para gestión automatizada por máquinas y programas de software. Los desarrolladores no trabajan directamente con el código XML, a no ser que necesiten realizar modificaciones puntuales. Para generar y manipular archivos XML se sirven de las API correspondientes al entorno de desarrollo y al sistema operativo para el cual programan. Los entornos de desarrollo Java y las API de Android manipulan el código XML principalmente a través de dos estándares: SAX y DOM.

SAX (Simple API for XML) es una interfaz de programación de aplicaciones que permite acceder al contenido de archivos XML de modo rápido y sencillo. Su funcionamiento es secuencial: procesa las etiquetas una por una y de principio a fin. El intérprete SAX analiza el documento XML en busca de marcas de apertura y cierre, comprueba que el código está escrito correctamente y gestiona los espacios de nombres. Gracias al carácter lineal del procedimiento se consigue rapidez en la ejecución y economía de recursos —no es necesario cargar previamente en memoria la totalidad del archivo—. Esto permite que SAX, tras haberse convertido en un estándar de Java, sea portado a otras plataformas, como por ejemplo Python. El inconveniente de SAX es que carece de estructura en árbol, lo cual dificulta el proceso de información y la introducción de cambios en el código. Cada vez que se quiera buscar algo nuevo dentro del archivo habrá que comenzar otra vez desde el principio.

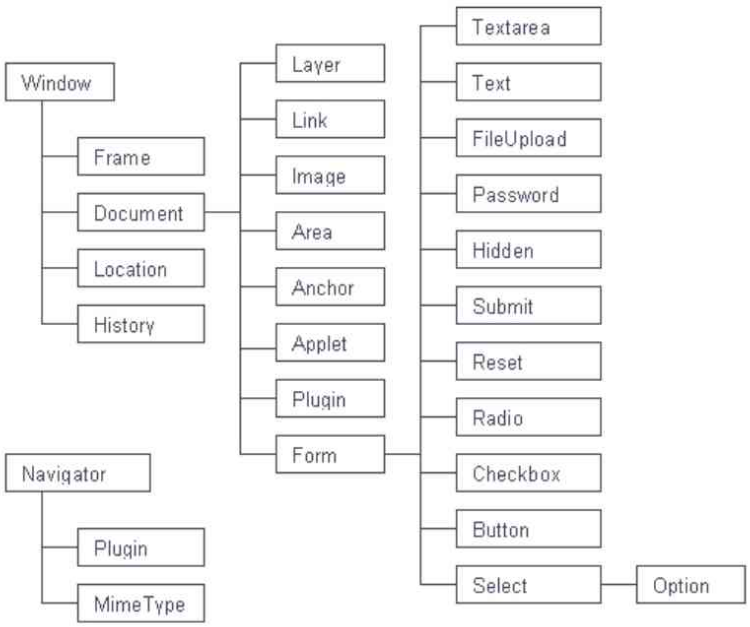

Figura 5.2. Especificación del DOM

Por su parte DOM (Document Object Model; en español: Modelo de Objetos del Documento) no solo interpreta el código sino que proporciona además un conjunto de objetos estándar para la representación de documentos HTML y XML, así como reglas para definir el modo en que dichos objetos han de combinarse e interaccionar entre ellos, y una interfaz para manipularlos. A diferencia de SAX, que admite tan solo lectura secuencial, DOM permite que las aplicaciones accedan a archivos HTML o XML y modifiquen aleatoriamente su contenido y estructura.

DOM, lo mismo que HTML y XML, es un desarrollo del World Wide Web Consortium. Sus ventajas funcionales, sin embargo, se ven contrarrestadas por los inconvenientes derivados de la complejidad, que de manera inevitable se hacen notar en un mundo limitado por la escasez de recursos como es el de la informática móvil: lentitud en el procesado de la información y una necesidad considerable de memoria RAM.

5.2.3 Ejemplo de análisis

En los directorios de las aplicaciones Android, los archivos XML suelen tener nombres explicativos de su función y del tipo de datos que contienen. Podemos examinar su contenido con cualquier editor de texto. Fijémonos por ejemplo en el archivo `AndroidManifest.xml`, el cual contiene los parámetros de funcionamiento y datos relativos a la estructura, actividades, servicios y permisos de una aplicación Android. Este archivo se genera automáticamente en el entorno de desarrollo. A medida que se añaden páginas (*Activities*) y nuevos elementos (*Views*, *Layouts*, etc.), las diferentes secciones de `AndroidManifest.xml` se van poblando con los parámetros correspondientes. Finalmente, tras la compilación del código dex y la formación del paquete completo, `AndroidManifest.xml` es añadido al archivo *.apk* que contiene la aplicación. Este archivo es lo que se descarga desde GooglePlay u otras fuentes para instalar la aplicación en el dispositivo móvil. Los archivos *.apk* se guardan en ubicaciones diferentes dependiendo de la versión de Android y del fabricante del terminal. En un teléfono Samsung Galaxy el investigador los encontrará por lo general en el directorio `/system/app`.

Un archivo *.apk* no es en último grado otra cosa que un contenedor ZIP. Su contenido puede extraerse con cualquier herramienta de descompresión. Probamos, por ejemplo, con el archivo `Gmail.apk`, trasladándolo desde el terminal hasta la estación de trabajo a través de ADB:

```
igandekoa@igandekoa:~/android-sdk-linux/platform-tools$ adb pull
/system/app/Gmail.apk
3920 KB/s (1919261 bytes in 0.478s)
```

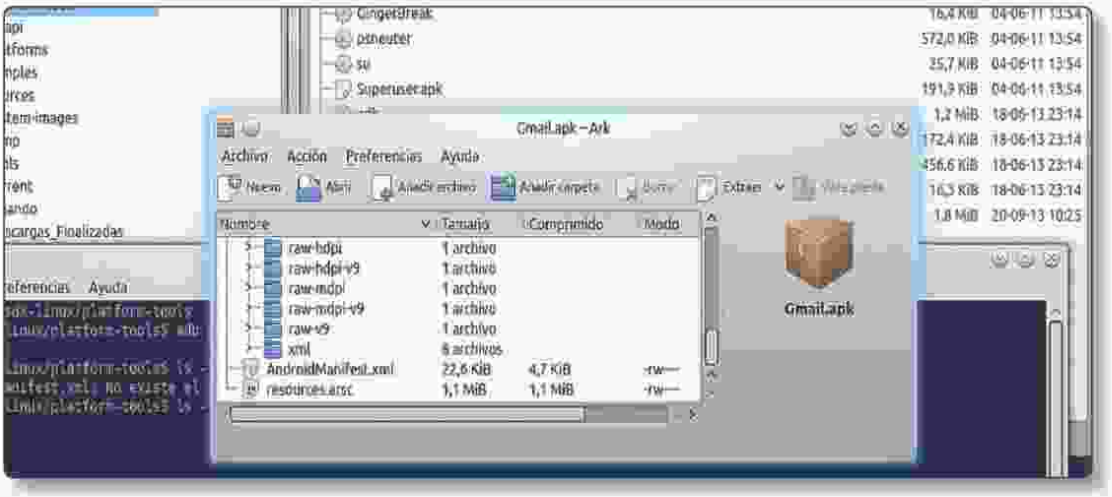

Figura 5.3. Archivo AndroidManifest.xml en el interior del paquete .apk

Extraemos del contenedor el archivo `AndroidManifest` (Figura 5.3) e intentamos visualizarlo con un editor de texto. No lo conseguiremos, porque el archivo no está en un formato de texto legible. El programador de la aplicación, cuando accede al archivo desde su entorno de desarrollo, puede ver sus secciones y trabajar con ellas, añadiendo actividades o, más importante aún, estableciendo los permisos que la aplicación pide al usuario para poder ejecutarse en el terminal. Pero nosotros, en el paquete `apk` únicamente disponemos de los datos compilados.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest android:versionCode="585" android:versionName="5.8.5" android:installLocation="internalOnly"
  xmlns:android="http://schemas.android.com/apk/res/android">
    <supports-screens android:anyDensity="true" android:smallScreens="true" android:normalScreens="tru
    <uses-permission android:name="android.permission.ACCESS_SUPERUSER" />
    <uses-permission android:name="com.android.vending.CHECK_LICENSE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.GET_TASKS" />
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
    <uses-permi
    <permission android:name="com.jrummy.liberty.toolboxpro.permission.C2D_MESSAGE" android:protection
    <uses-permission android:name="com.jrummy.liberty.toolboxpro.permission.C2D_MESSAGE" />
    <uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
    <permission android:label="@string/parm_run_script" android:name="jackpal.androidterm.permission.R
    <permission android:label="@string/parm_append_to_path" android:name="jackpal.androidterm.permissi
    <permission android:label="@string/parm_prepend_to_path" android:name="jackpal.androidterm.permiss
    <uses-feature android:name="android.hardware.touchscreen" android:required="false" />
    <uses-feature android:name="android.hardware.telephony" android:required="false" />
    <application android:label="@string/app_name" android:icon="@drawable/appicon" android:hardwareAcc
        <service android:name="com.jrummy.billing.BillingService" />
        <receiver android:name="com.jrummy.billing.BillingReceiver">
            <intent-filter>
                <action android:name="com.android.vending.billing.IN_APP_NOTIFY" />
                <action android:name="com.android.vending.billing.RESPONSE_CODE" />
                <action android:name="com.android.vending.billing.PURCHASE_STATE_CHANGED" />
            </intent-filter>
        </receiver>
        <activity android:name="com.google.ads.AdActivity" android:configChanges="keyboard|keyboardHid
```

Figura 5.4. Aspecto típico del archivo AndroidManifest.xml después de descompilar

El archivo `AndroidManifest.xml` es un objeto de evidencia importante para el investigador forense, sobre todo para el análisis de intrusiones y ataques con aplicaciones maliciosas descargadas desde sitios poco fiables o instaladas de manera intencionada por un delincuente informático.

Imagine que tenemos una aplicación de ayuda para la enseñanza secundaria que convierte grados Celsius en Fahrenheit. Un programa como este requiere permisos de ejecución elementales, y esto es lo que cabría esperar que especifiquen las secciones correspondientes de `AndroidManifest.xml`. Pero si el investigador halla evidencia que demuestre que se han activado permisos que posibilitan el acceso a datos del usuario, la utilización de funciones del teléfono o el envío de mensajes SMS a servicios *premium* con elevadas tarifas de pago, hay motivos para sospechar que esta aplicación hace algo más que ayudar a resolver problemas de física a los estudiantes de la ESO. Los cambios del escritorio también resultan sospechosos, puesto que permiten disfrazar la actividad de las aplicaciones en primer plano. He aquí algunos ejemplos de permisos que deberían inducir a la desconfianza, en caso de que los veamos especificados en el archivo `AndroidManifest.xml` (Listado 5.1):

```
android.permission.READ_PHONE_STATE
android.permission.CALL_PRIVILEGED
android.permission.READ_OWNER_DATA
android.permission.SET_WALLPAPER
android.permission.DEVICE_POWER
```

Listado 5.1. Permisos sospechosos

5.2.4 Apktool

Volvamos al galimatías que estábamos contemplando a través de nuestro editor de texto. No se alarme: no necesita instalar un entorno de desarrollo para poder visualizar los contenidos de `AndroidManifest.xml` del mismo modo en que aparecen mostrados en la figura 5.4. Vaya a Google y consiga una aplicación llamada Apktool. Mucho mejor si puede descargar una versión reciente a través de XDA-Developers. Hágase con los archivos correspondientes a su plataforma forense (Windows, OSX o Linux). Si trabaja con Windows, descomprímalos en un directorio de trabajo, o en la carpeta del sistema operativo si desea utilizarlos desde cualquier ubicación. Después copie el paquete *.apk* a ese directorio de trabajo y ejecute el comando siguiente:

```
apktool d Gmail.apk ./app_decrypt
```

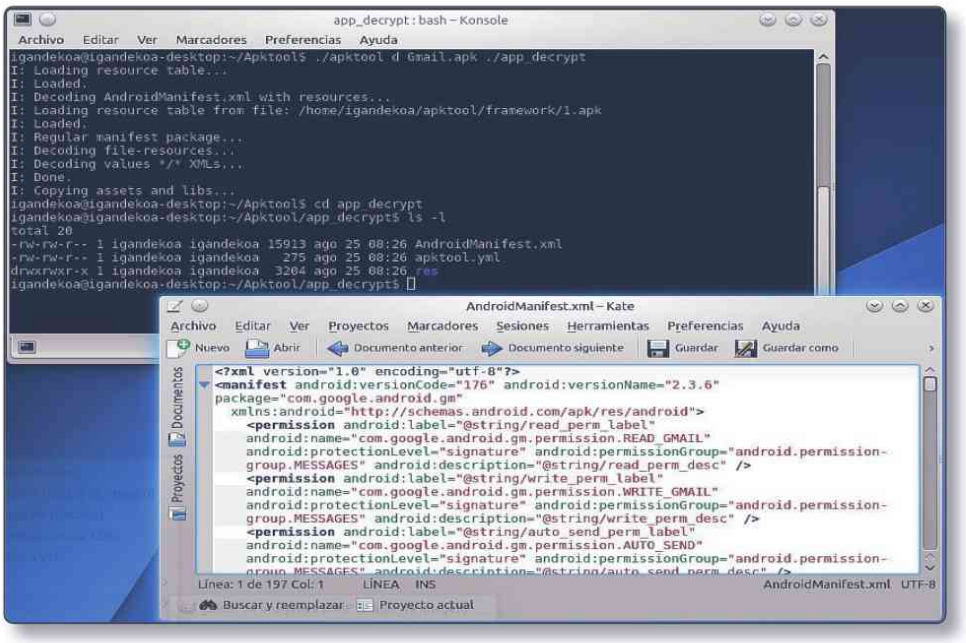

Figura 5.5. Uso de Apktool y visualizado de AndroidManifest.xml en texto claro

`Gmail.apk` es la aplicación que hemos copiado desde el directorio `/system/app` del terminal. La herramienta apktool, al ejecutarse, descompila los contenidos del paquete y crea una carpeta denominada `app_decrypt` en la que también hay una versión en texto claro del archivo `AndroidManifest.xml`, que podrá ser examinada con cualquier editor de texto o incluso con el Bloc de Notas de Windows (Figura 5.5). Apktool también extrae del archivo *.apk* un árbol de directorios, que coloca en la subcarpeta `./res` (recursos). En estos directorios podremos encontrar archivos XML correspondientes a otros elementos de la aplicación descompilada por Apktool.

Apktool consta de dos partes: un ejecutable y un archivo de recursos Java. Los ejecutables son específicos de cada sistema operativo. La parte escrita en Java, al ser código portable, sirve para todas las plataformas. Para un funcionamiento correcto de Apktool, tanto en Linux como en Windows, lo único que se necesita es que las dos partes de la aplicación se encuentren en el mismo directorio y esté instalado el *runtime* de Java.

5.3 INGENIERÍA INVERSA

En ocasiones, para caracterizar una aplicación maliciosa no basta saber qué permisos solicita. También puede resultar interesante conocer su funcionamiento,

concretamente si contiene instrucciones que son típicas del *malware*. No esperamos ver a una *app* amistosa, como por ejemplo un salvapantallas o nuestro viejo amigo el conversor de medidas, ejecutando instrucciones raras.

Figura 5.6. Página web de dex2jar

Para un análisis en profundidad es necesario recurrir a técnicas de ingeniería inversa. En otras palabras: se trata de descompilar no solo el texto de los archivos de configuración sino también el código de la aplicación, para examinarlo en busca de instrucciones sospechosas e indicios que apunten a funcionamientos impropios del software.

5.3.1 Descompilando código con dex2jar

La ingeniería inversa es utilizada principalmente por los desarrolladores con el objeto de corregir errores de programación y perfeccionar el software. También resulta de interés para analizar aspectos de seguridad relacionados con el software, concretamente en alguna de las situaciones siguientes:

1. **Identificación de código malicioso**: se trata de saber qué hace un componente de software, en particular si durante su funcionamiento ejecuta instrucciones sospechosas, como, por ejemplo, leer o transmitir credenciales de usuario, historiales de Internet, direcciones de correo electrónico u otros elementos a ubicaciones externas, o establecer

conexiones entre el dispositivo móvil y un servidor remoto con el objeto de tomar el control del terminal y convertirlo en parte de una *botnet*.

2. **Descubrimiento de defectos de programación** que puedan traducirse en fallos de seguridad aprovechables por un atacante.

3. **Identificación de funcionalidades parásitas** en el interior de aplicaciones legítimas, como resultado de manipulaciones llevadas a cabo por ciberdelincuentes.

Sobra decir que la ingeniería inversa es ilegal, salvo que se trate de aplicaciones maliciosas o software elaborado por el propio usuario. Descompilar el código de un programa desarrollado en código propietario vulnera derechos de propiedad intelectual. El investigador debe tenerlo en cuenta antes de ponerse a experimentar con sus herramientas.

Realizar ingeniería inversa sobre Android es relativamente fácil, sobre todo si ya se ha utilizado Apktool para extraer el archivo `AndroidManifest.xml` y otros recursos de la aplicación. En uno de los directorios de salida de Apktool el investigador hallará el binario de la aplicación en código **.dex*, un formato que, como ya se dijo en apartados anteriores, sirve para ser ejecutado en la máquina virtual Dalvik. Este archivo puede convertirse a código Java mediante la herramienta dex2jar, obtenible en la web de recursos de Google: *http://code.google.com/p/dex2jar/downloads/list* (Figura 5.6).

El código Java generado por dex2jar es meramente funcional, con una presentación espartana y desprovisto de comentarios y de cualquier otro tipo de documentación. Tener nociones de Java es imprescindible para entender el funcionamiento del *malware*. No hace falta ser un programador experto ni tener instalado el entorno de desarrollo Eclipse. Se trata de saber lo suficiente para identificar instrucciones sospechosas relacionadas con operaciones ilegítimas, como por ejemplo el robo de información personal del usuario o el establecimiento de conexiones con servidores remotos.

5.3.2 Análisis de malware

Aunque en la práctica no es necesario conocer paso por paso el funcionamiento de todas las instrucciones que componen una aplicación maliciosa, la identificación y caracterización de la misma debe hacerse de acuerdo con una estrategia adecuada que permita atestiguar su naturaleza maligna sin que haya lugar a dudas. No es necesario entender el funcionamiento del código línea por línea, pero, como norma general,

para estar seguros del carácter perverso de un elemento de software es preciso dar respuesta a las cuestiones siguientes:

1. ¿**De dónde procede y qué es lo que hace** una vez instalado en el dispositivo? Si está disponible en mercados no oficiales o en páginas desconocidas es conveniente analizar su funcionamiento, bien mediante ingeniería inversa bien obligándolo a ejecutarse en una *sandbox*.

2. ¿**Qué permisos solicita al usuario**? ¿Van más allá de lo que el programa requiere para un funcionamiento normal y proporcionado a las prestaciones que aparenta ofrecer?

3. ¿**A qué datos tiene acceso**? ¿Toca información personal del usuario? ¿Su correo electrónico? ¿Sus mensajes SMS? ¿Ejecuta alguna rutina de búsqueda de expresiones regulares para localizar números de teléfono, direcciones de correo electrónico, contraseñas, números de tarjeta de crédito, códigos TAN de entidades bancarias, etc.? ¿Hay en el código descompilado por dex2jar expresiones similares a `GetContactInfo()`, `GetCurrentLocation()` o `GetBrowserHistory()`? ¿Se puede considerar normal para una aplicación de esta clase?

4. **Conectividad**. ¿Aparecen en el código descompilado direcciones de páginas web rarillas, correos electrónicos extraños o direcciones IP sospechosas? ¿Instrucciones del tipo `ConnectToServer`, `SendDataToMaster` o cosas parecidas?

5.4 SQLITE

Android utiliza archivos XML para guardar parámetros de configuración del sistema operativo y las aplicaciones. Sin embargo la mayor parte de los datos creados por la actividad del usuario se archivan automáticamente en bases de datos SQLite. Este formato y sus archivos característicos, reconocibles por la extensión *.db*, constituyen otro de los objetivos prioritarios de la investigación forense. Para llegar a ellos es por lo que nos hemos molestado en estudiar el sistema de archivos y las estructuras internas de Android, y también para lo que hemos aprendido a rootear dispositivos móviles, instalar particiones Recovery y llevar a cabo otras manipulaciones que comprometen el principio de no alteración de las pruebas, si bien en una medida tolerable y por una razón justificada.

5.4.1 Carácter compacto y portable

SQLite es un sistema de bases de datos que, a diferencia de los que se utilizan en entornos empresariales e Internet, no está basado en una arquitectura servidor-cliente. La funcionalidad de red se halla encapsulada en una librería de tan solo unos centenares de kilobytes que se enlaza en tiempo de ejecución con cualquier programa que solicite servicios de bases de datos. Esto hace de SQLite una herramienta muy eficaz, puesto que las llamadas a funciones exportadas por librerías son más rápidas y fiables que la comunicación entre procesos, sobre todo cuando se realizan dentro de un mismo entorno de ejecución sin tener que transitar la red en viajes de ida y vuelta hasta el servidor de bases de datos.

Eficiencia y caracter compacto hacen de SQLite un elemento imprescindible para el manejo de bases de datos en navegadores de Internet (por ejemplo, Mozilla Firefox) y otras aplicaciones de usuario (como LibreOffice, de Sun). SQLite también se utiliza en teléfonos móviles, reproductores multimedia e incluso maquinaria industrial, por su capacidad para organizar los ingentes caudales de información generados por el funcionamiento de todo tipo de aparatos conectados en red.

5.4.2 Análisis de archivos .db

Instalar SQLite en OSX o Windows es fácil y no requiere ningún preparativo especial. Tratándose de código libre, basta ir a la página del desarrollador y descargar los archivos ZIP con los binarios ejecutables para el sistema correspondiente. Con Linux ni siquiera tenemos que preocuparnos de ello, puesto que la mayor parte de las distribuciones incluyen una versión actualizada de SQLite. No obstante, para no tener que manejar las tablas de datos con los comandos de consola SQLite, quizás interese instalar un cliente gráfico, como por ejemplo Sqliteman. La versión para Windows se instala de la manera que ya conocemos. Para Linux —en este caso Ubuntu— tecleamos:

```
igandekoa@igandekoa:~$ sudo apt-get install sqliteman
```

Una vez que el investigador forense ha adquirido los archivos de bases de datos —por ejemplo a través de una conexión ADB— desde sus directorios correspondientes en la partición de datos del dispositivo móvil, lo único que tiene que hacer es explorar su contenido con las utilidades en línea de comando. La sintaxis es similar a la de otros sistemas de bases de datos relacionales. Quien posea nociones de SQL tiene hecha la mitad del camino.

Más práctico, sin embargo, es abrir los archivos *.db* con el cliente gráfico. De este modo podremos visualizar de manera inmediata la estructura de la base de datos

y las diferentes tablas que la componen. En Sqliteman (Figura 5.7) la parte inferior del recuadro izquierdo, bajo el epígrafe **sqlite_master**, informa sobre la composición de la base de datos con las tablas que contiene, registros y otras características. En la parte superior se pueden encontrar (bajo **Tables**) todas las tablas de la base.

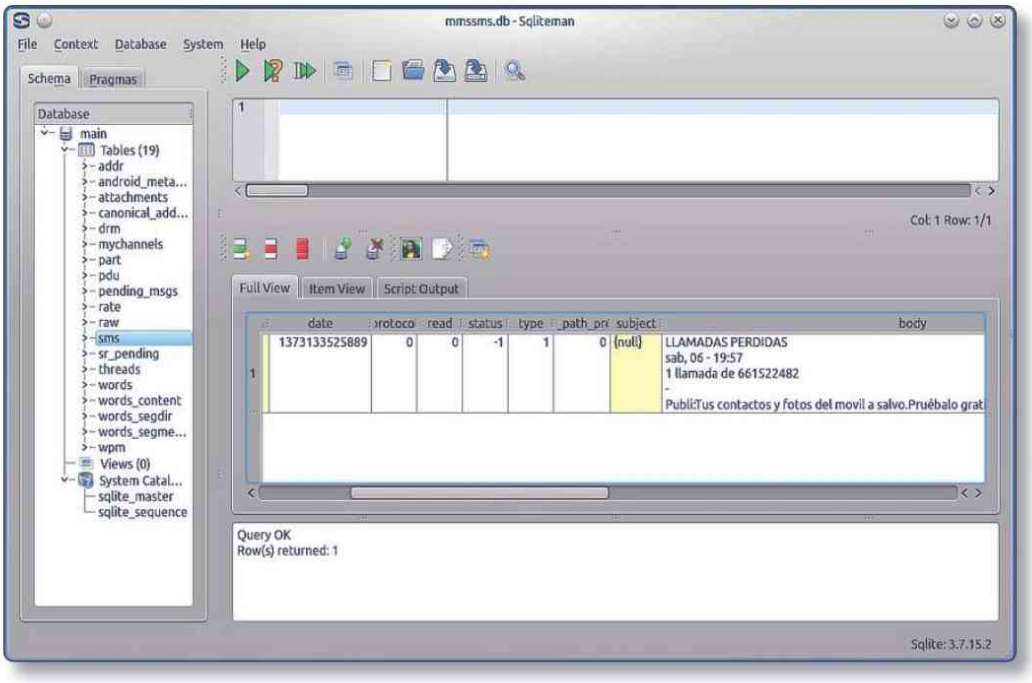

Figura 5.7. Interfaz de Sqliteman

En un terminal móvil, el uso cruzado de todos estos datos, gestionados por el entorno de aplicaciones de Android y las API del sistema, es lo que produce el milagro visual de un dispositivo que parece saberlo todo y es capaz de ofrecer la información perfectamente estructurada en una pantalla no más grande que la palma de la mano. Los archivos *.db* contienen gran cantidad de tablas. Algunas están vacías, otras incluyen información auxiliar. Los datos que interesan al investigador se encuentran incluidos, por lo general, en una o dos tablas.

Un elemento de gran importancia lo constituyen las marcas de tiempo. Estas son de gran utilidad a la hora de explicar cómo tuvieron lugar y se sucedieron los hechos, ayudando así a establecer la línea de tiempo del caso. Dentro de las bases de datos SQLite las marcas de tiempo se hallan incluidas en formatos específicos del software que las procesa, y no necesariamente legibles para un observador humano. Para facilitar la lectura se recurre a la ayuda de conversores de hora y fecha. En Linux, por ejemplo, podemos utilizar el comando de sistema `date`:

```
igandekoa@igandekoa:~$ date -d @1374139059
jue jul 18 11:17:39 CEST 2013
```

5.4.3 Ejemplo de utilización

Supongamos que durante la investigación de un dispositivo móvil hemos conseguido extraer archivos de bases de datos que contienen el registro de navegación en Internet. Estos archivos son `browser.db` (Figura 5.8) y `webview.db` (Figura 5.9), y han sido copiados mediante ADB. Puede abrir `browser.db` pulsando sobre él con el botón derecho del ratón y seleccionando el programa **Sqliteman**. También puede iniciar primero el cliente y cargar el archivo. Vaya al recuadro de la izquierda y fíjese en la estructura de la base de datos. Podrá ver que está compuesta por tres tablas. Observe sus denominaciones junto con los comandos y parámetros utilizados para crearlas (sección inferior de la ventana). En la tabla **Bookmarks**, por ejemplo, hallará información sobre sitios visitados por el sospechoso, además del número de visitas y las marcas de tiempo que fechan los accesos.

Analizando de manera análoga el archivo `webview.db`, también relacionado con el navegador, hallaremos las *cookies* depositadas en el navegador desde los diferentes sitios de Internet visitados por el usuario. Esta colección de elementos de evidencia no solo resulta útil en una investigación forense; también puede interesar a un delincuente informático que haya robado el dispositivo y tenga intención de utilizarlo para acceder a información personal, servicios de banca *on line* u otros recursos del usuario en Internet.

Las tablas pueden exportarse a archivos de texto CSV, un formato simple con columnas separadas por comas o tabuladores. También pueden ser exportadas a hojas de datos Excel, para llevar a cabo un análisis detallado con Office y poder incluir esta información en el informe.

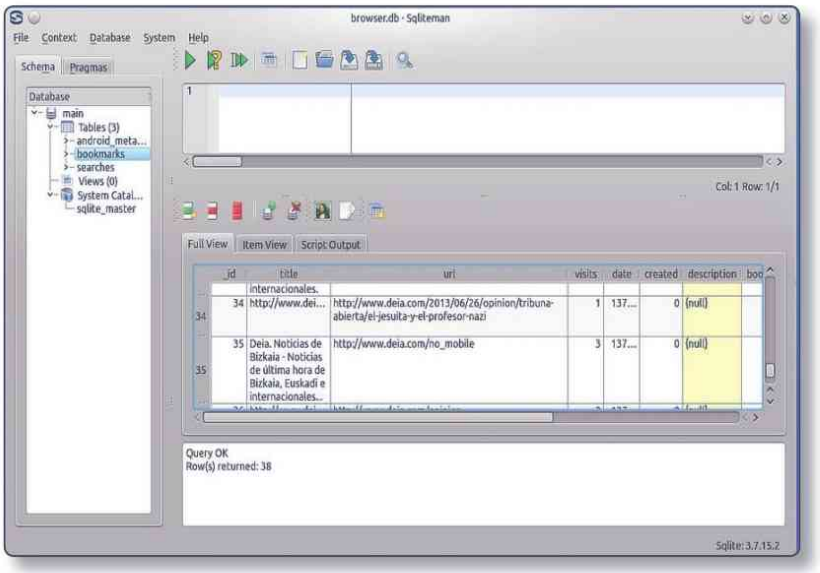

Figura 5.8. Archivo de base de datos browser.db visto a través de Sqliteman

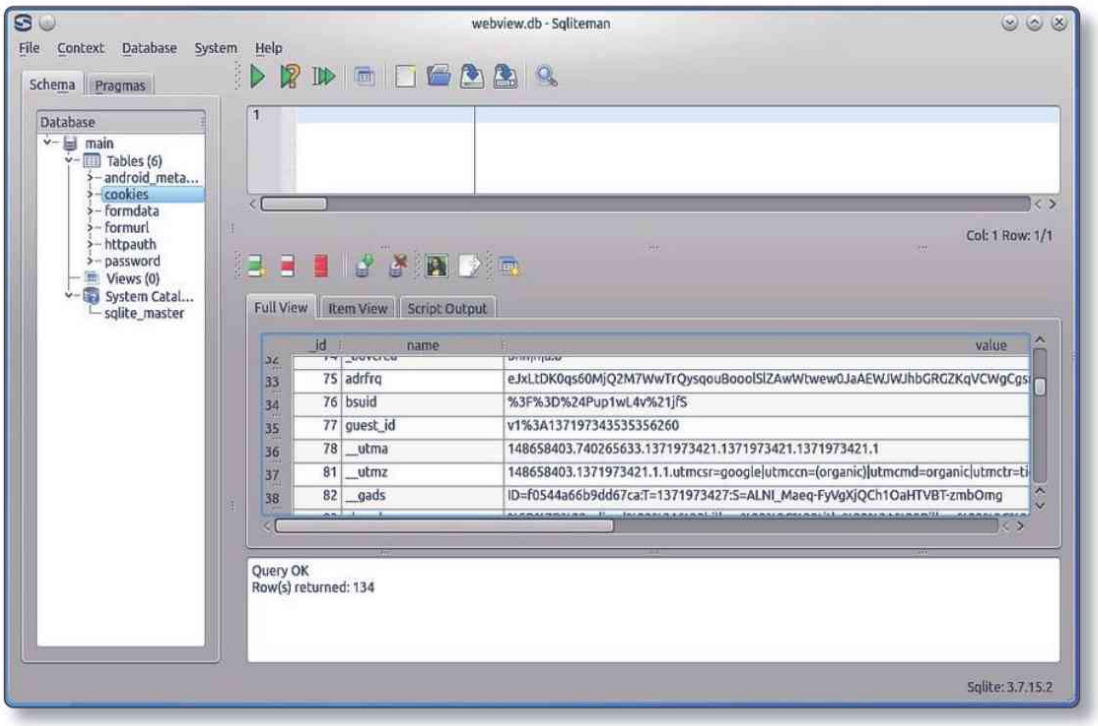

Figura 5.9. Archivo de base de datos webview.db visto a través de Sqliteman

5.5 ANÁLISIS DE APLICACIONES ANDROID

En Android cada aplicación tiene su propio árbol de subdirectorios, equivalente a la zona `/home` de un usuario de Linux. Dentro de este árbol, y en los subdirectorios correspondientes, encontraremos, además del código y otros recursos, los parámetros de configuración y los datos del usuario convenientemente alojados en archivos XML y bases de datos SQLite. A continuación se citan algunas aplicaciones que vienen instaladas de fábrica en dispositivos móviles Android. Con ellas el usuario dispone de lo necesario para utilizar su terminal de modo productivo.

En cuanto a las aplicaciones descargadas desde Google Play y *markets* alternativos, o escritas en un entorno de desarrollo Eclipse, el esquema de organización es similar. Cada directorio lleva el mismo nombre que el paquete de la aplicación. En algunos casos, dependiendo de la versión de Android o del fabricante que haya realizado la compilación de los paquetes, las denominaciones de archivo pueden variar. Para estar seguro de que el investigador forense adquiere los datos correctos, es necesario prestar atención a todos los paquetes y directorios con denominaciones sospechosas o variantes en el nombre. En caso de duda es preciso consultar en foros técnicos y páginas de Internet especializadas.

En los apartados siguientes, las ubicaciones que se mencionan para archivos y carpetas son relativas. Cuando se indique, por ejemplo, un directorio llamado `./cache`, esto se refiere a la ubicación relativa correspondiente al directorio en que nos encontramos en ese momento. La ruta completa, partiendo desde la raíz de directorios del sistema, puede ser `/data/data/com.android.browser/cache`, `/data/data/com.android.mms/cache`, u otra por el estilo, dependiendo del programa que se quiere analizar.

5.5.1 Mensajes SMS y MMS

Android incluye la aplicación Messaging para gestionar y archivar mensajes SMS y multimedia. Siguiendo el esquema de nombres de Android, el directorio de esta aplicación lleva la misma denominación que el paquete APK, `com.android.providers.telephony` (Figura 5.10), y dispone de varias subcarpetas para guardar los datos de configuración y funcionamiento. En algunas de ellas el investigador encontrará las imágenes, los vídeos y otros elementos que forman parte de los mensajes multimedia.

Las bases de datos en las que se guardan los mensajes de texto se encuentran en el subdirectorio `databases` y constan de dos archivos: `msmsms.db` y `telephony.db`. Dentro del primer archivo, la tabla `sms` contiene el texto de todos los mensajes y debería constituir el objetivo principal de la adquisición forense. El archivo `telephony.db` incluye datos técnicos de configuración del operador.

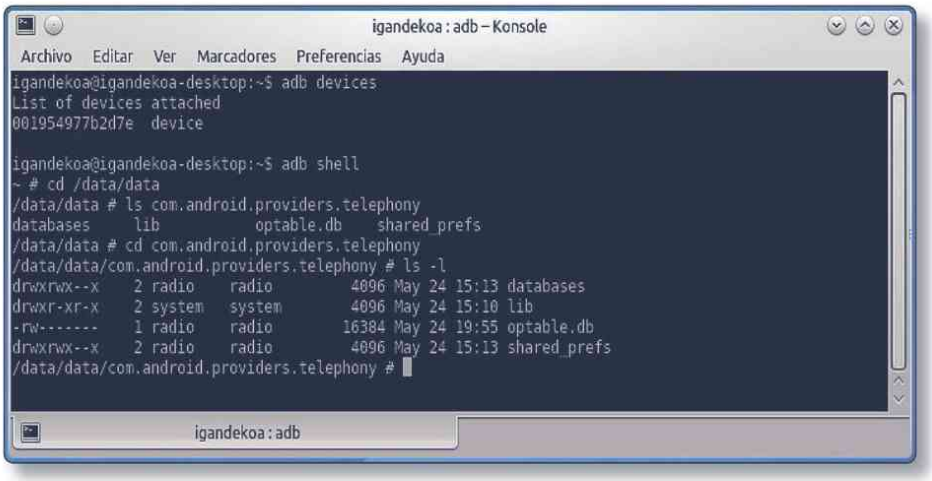

Figura 5.10. Directorios de la aplicación de mensajes SMS/MMS

5.5.2 Navegador de Internet

Para acceder a la WWW, el software de navegación de Android se sirve de WebKit, plataforma programada en código libre WebKit que a su vez constituye una evolución del motor de renderizado KHTML del navegador Linux KDE Konqueror. El nombre oficial de la aplicación es Internet. En casi todas las versiones, y en la mayor parte de los dispositivos móviles, el paquete correspondiente al navegador se llama `com.android.browser`, y sus datos se encuentran alojados en la carpeta del directorio `/data/data` con el mismo nombre.

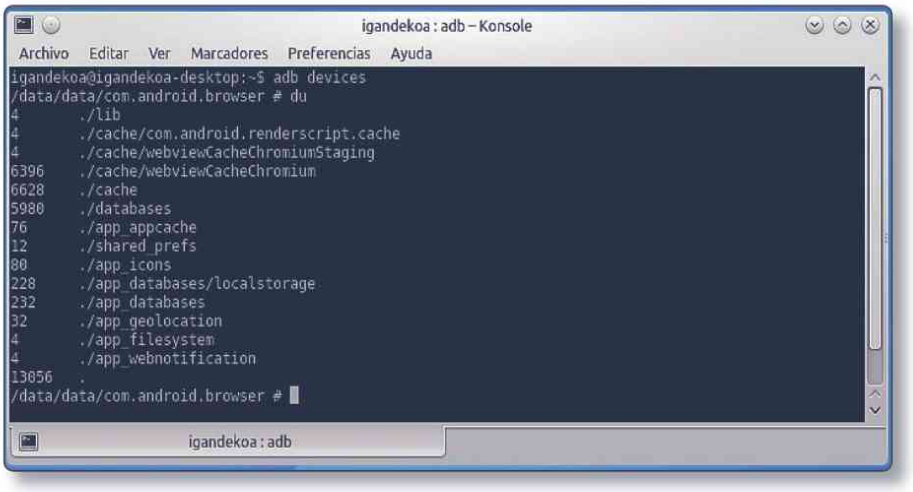

Figura 5.11. Navegador de Internet

El árbol de carpetas para una aplicación tan compleja es inevitablemente complejo y comprende gran cantidad de subdirectorios y archivos (Figura 5.11). Conociendo el esquema de denominaciones Android, sin embargo, no cuesta mucho trabajo moverse por él. Lo que nos interesa se encuentra en el directorio `./databases`. Allí el investigador podrá encontrar elementos de evidencia como los archivos `browser.db`, `webview.db` y `webviewCache.db`. Todos son bases de datos SQLite. El archivo `webview.db`, además, puede incluir información relativa a dominios, nombres de red y contraseñas utilizadas por el usuario.

En el directorio `./cache` el investigador hallará rastros de la actividad de navegación del usuario y otros elementos de evidencia potenciales: archivos de código HTML y texto ASCII, imágenes JPG, GIF y PNG, etc. Otros datos de la aplicación que pueden tener interés para el investigador son los relativos a geolocalización, guardados dentro de la carpeta `./app_geolocation` en dos archivos SQLite denominados `CacheGeoposition.db` y `GeolocationPermissions.db`.

Figura 5.12. Contactos

5.5.3 Contactos

La aplicación Contactos, oficialmente llamada Contacts y perteneciente a un paquete que lleva la denominación `com.android.providers.contacts`, se aloja en el subdirectorio del mismo nombre (figura 5.12), y es la utilidad suministrada por Android para que el usuario guarde los datos relativos a su agenda telefónica. Este componente suministra dicha información a las aplicaciones que la soliciten y tengan permiso para acceder a ella.

Los datos se guardan en un archivo del directorio `./databases` llamado `contacts2.db` o con una denominación parecida, el cual contiene gran número de tablas. Aquí el investigador podrá encontrar el registro de llamadas entrantes y salientes, así como información personal, postal, de negocios y cuentas de correo electrónico —Gmail, Yahoo, Microsoft Exchange— y redes sociales —Twitter, Facebook, Linkedin, etc.—.

Las fotografías de los perfiles y otras imágenes relacionadas con estos datos se alojan en el directorio `./files`.

5.5.4 Media Scanner

El escaneador de medios (Media Scanner) es una aplicación de sistema que se encarga de actualizar la lista de archivos multimedia de imagen y sonido encontrados en las diferentes particiones del dispositivo móvil. La gestión de estos archivos se lleva a cabo mediante los metadatos incluidos en los mismos. El nombre oficial de

la aplicación es Media Store y el paquete y/o directorio donde se encuentran sus archivos es `com.android.providers.media`.

En la carpeta `./databases` el investigador hallará tres archivos SQLite: uno con los datos de los medios almacenados en el espacio de almacenamiento interno del dispositivo móvil (generalmente denominado `internal.db`); otro para la tarjeta externa SD y el tercero para el espacio FAT emulado en memoria NAND. Dependiendo de la marca y modelo de dispositivo las denominaciones de estos archivos pueden cambiar, pero, en cualquier caso, no debería costar mucho trabajo dar con ellos.

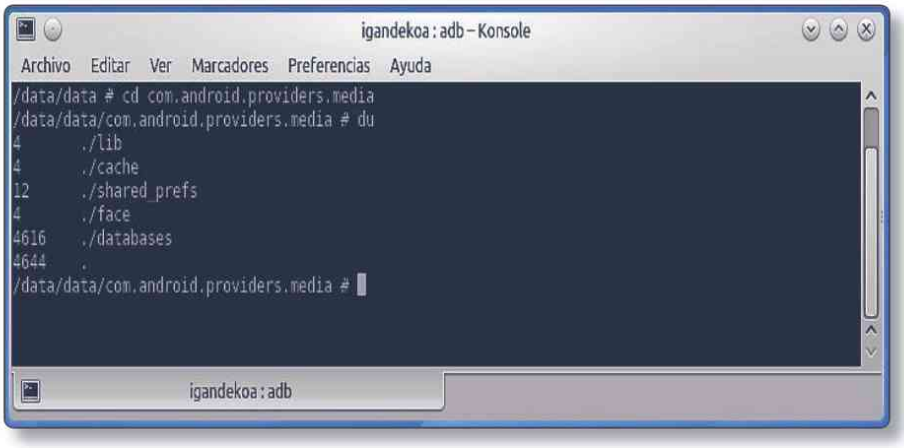

Figura 5.13. Media Scanner

El análisis de los archivos que se encuentran alojados en estos directorios (Figura 5.13) puede aportar una valiosa información. Es preciso tener en cuenta que además de Media Store puede haber otros escaneadores de medios instalados en un sistema Android, por lo cual conviene examinar el dispositivo en busca de otras aplicaciones de tipo similar.

En segundo lugar, lo que en un momento dado podamos encontrar dentro de un archivo *.db* no tiene por qué ser lo único que hay o ha habido en él. Debido a la estructura de registros indexados de las bases de datos SQLite, es posible que el archivo contenga entradas borradas correspondientes a elementos de información antiguos guardados en el terminal móvil (imágenes, vídeos, canciones, mensajes de voz, etc.). Si se está buscando algo muy específico y no se encuentra con el cliente SQLite, quizás merezca la pena llevar a cabo una criba minuciosa del archivo con el comando `strings` o un editor hexadecimal.

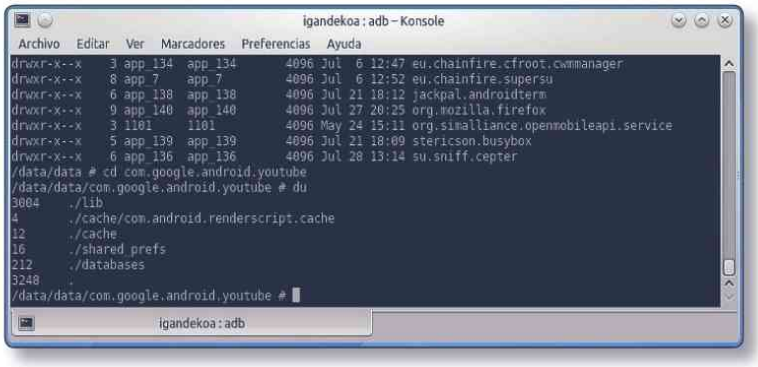

Figura 5.14. Youtube

5.5.5 Youtube

Youtube no necesita presentación. Dentro de su carpeta —previsiblemente denominada `com.google.android.youtube`— no hay bases de datos, pero sí un subdirectorio `./cache` que contiene dentro de archivos XML la información de los vídeos buscados en Internet (Figura 5.14).

5.5.6 Google Maps

Google Maps es la aplicación integrada de mapas de Google, con funcionalidad adicional para búsqueda de direcciones, trazado de rutas, información sobre lugares interesantes y servicios GPS. El paquete correspondiente se llama `com.google.android.apps.maps` y se halla en un directorio del mismo nombre. Aquí hay varias carpetas que pueden resultar interesantes en una investigación: en `./databases`, por ejemplo, encontraremos historiales de búsquedas, itinerarios, destinos favoritos. Las carpetas `./files` y `./cache` también pueden contener mucha información, algo previsible en vista de las grandes cantidades de datos descargadas de Internet y procesadas por la aplicación. Vivimos en la era del *big data*, y eso se hace notar en los directorios temporales y las cachés.

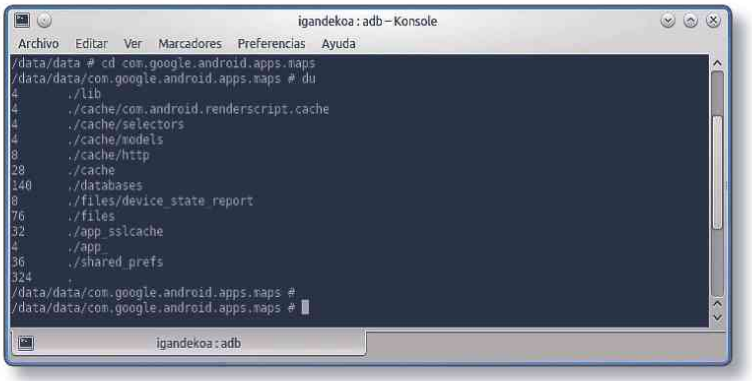

Figura 5.15. Google Maps

Como característica interesante de esta aplicación es necesario mencionar su capacidad para escribir datos, principalmente de caché, en la tarjeta SD externa. Caso de existir una ubicación de volcado externa, lo más probable es que se encuentre en `/mnt/sdcard/Android/data/com.google.android.apps.maps/cache`.

5.5.7 Gmail

Los datos de la aplicación Gmail (Google Mail) se encuentran disponibles en el directorio `com.google.android.gm` (Figura 5.16), aunque en la tarjeta SD también puede haber información relacionada con el correo electrónico. Gmail genera archivos de bases de datos con gran cantidad de tablas. Por lo general cada cuenta dispone de un archivo *.db* que contiene el correo del usuario. En otros archivos, como por ejemplo `gmail.db`, `downloads.db`, `suggestions.db`, el investigador podrá encontrar información adicional y datos de gestión.

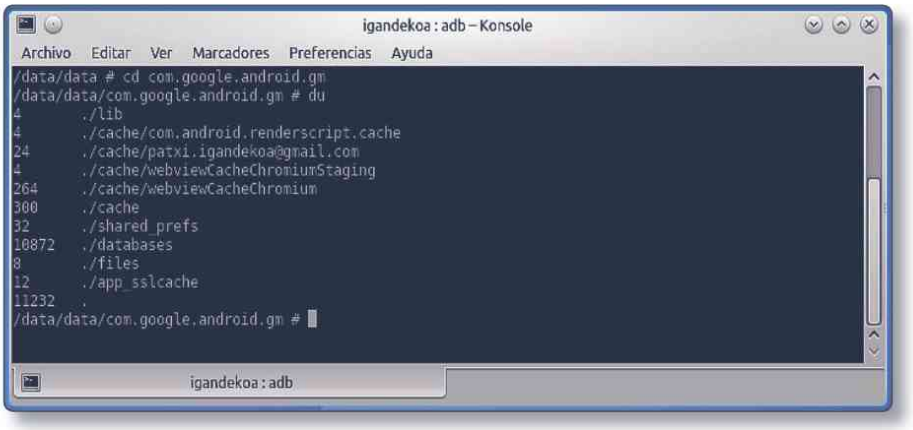

Figura 5.16. Aplicación de correo electrónico Gmail

5.5.8 Otras aplicaciones

Lo que hemos visto en apartados anteriores da una idea del esquema que Android aplica para organizar los datos generados por las aplicaciones del usuario. En caso de que la investigación estuviese centrada en redes sociales, documentos ofimáticos, información geográfica o cualquier otro tipo de objeto, lo único que hay que hacer es ir al directorio correspondiente para buscar los archivos de configuración XML y las bases de datos SQLite de la aplicación. En las otras carpetas se podrá encontrar cachés, archivos de código, páginas HTML, imágenes y otros recursos utilizados por la aplicación.

Las bases de datos SQLite constituyen objetivos prioritarios en el análisis del material procedente de la adquisición forense de dispositivos móviles Android. Casi todo lo que el sospechoso hace con su terminal se encuentra guardado en estos archivos de bases de datos. Cuando la información es abundante, el cliente SQLite puede resultar una herramienta de difícil manejo. Por ello, para un análisis en profundidad, se aconseja exportar los datos a archivos de texto CSV o, mejor todavía, a una hoja de datos Excel.

5.6 HERRAMIENTAS DE ANÁLISIS

En ocasiones no es suficiente limitarse a visualizar el contenido de las bases de datos o los archivos XML por medio de utilidades capaces de gestionar estos formatos. Si el investigador intuye que en una adquisición forense puede haber más de lo que parece, o si necesita elementos de evidencia adicionales para incluir en a su informe, puede llevar a cabo un registro a bajo nivel con la ayuda de dos herramientas imprescindibles para la actividad forense, no solo con dispositivos móviles, sino también con cualquier otro tipo de soporte digital de datos. Es más, tarde o temprano tendrá que utilizar un editor hexadecimal y el comando nativo de Linux `strings`.

5.6.1 Editores hexadecimales

Desde los comienzos del PC, el editor hexadecimal (Figura 5.17) ha prestado servicios inapreciables a desarrolladores, investigadores forenses, hackers y ciberdelincuentes por igual. Entre los lectores habrá seguramente quien aún se acuerde de los viejos tiempos del MS-DOS, cuando con la ayuda de un editor Norton buscábamos en el interior de un archivo ejecutable con extensión EXE o COM secuencias de caracteres reveladoras de la presencia del virus de la pelotita o del infame Viernes 13. La investigación en el campo de la informática forense ha permitido recuperar parte de aquel espíritu. Seguimos escudriñando el código en bruto como parte de nuestra operativa habitual, y seguirá siendo así en el futuro. A la hora de acometer el análisis de una aplicación maliciosa, examinar archivos de bases de datos con registros borrados o buscar metadatos, ninguna herramienta automatizada puede reemplazar la perspicacia del investigador experimentado.

La utilidad del editor hexadecimal es doble: con él podemos visualizar archivos de cualquier tipo independientemente del software utilizado para crearlos. En segundo lugar, el editor revela toda la información que contiene el archivo, incluyendo datos ocultos que no se muestran en la aplicación de usuario e información correspondiente a versiones antiguas del documento que permanece guardada por si

acaso el usuario deseara restaurar su documento al estado que tenía en un momento anterior.

No es fácil llevarse bien con los editores hexadecimales. Una cosa es ver el texto y las imágenes bien presentadas en un procesador de textos; otra, tener que leer en crudo una ristra de caracteres hexadecimales o ASCII, con símbolos crípticos, secuencias de escape y otras rarezas por el estilo. Es normal que los usuarios formados con posterioridad a la aparición de las interfaces gráficas OSX o Windows encuentren el editor hexadecimal aún más incómodo que la línea de comando. La dificultad se ve acrecentada por el hecho de que el editor hexadecimal es una de esas utilidades cuyo funcionamiento no se enseña en ningún sitio, ni siquiera en libros de informática básica. Es el usuario quien debe aprender a base de práctica, consultas en Internet y paciente experimentación.

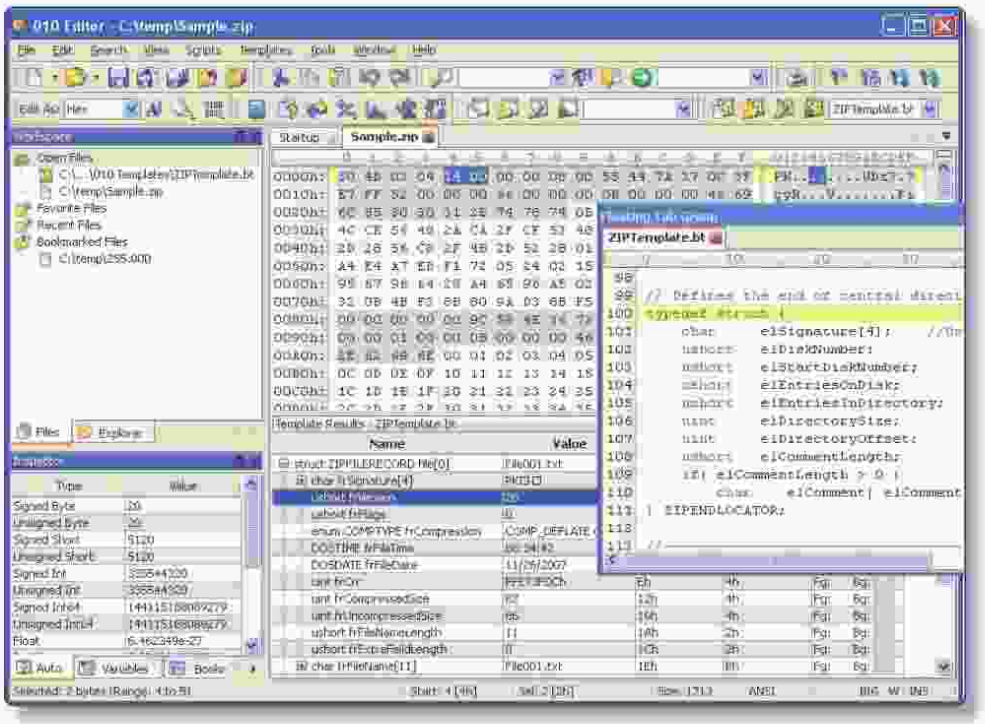

Figura 5.17. Editor hexadecimal

En realidad, el funcionamiento de un editor hexadecimal es muy sencillo: a la izquierda una escala de posiciones, en el centro un campo con los caracteres hexadecimales, a la derecha el texto equivalente ASCII. En la parte superior, finalmente, un menú con opciones, apartados de configuración y herramientas de búsqueda y configuración.

El análisis de archivos con un editor hexadecimal requiere paciencia y cierto grado de conocimiento de las estructuras de datos subyacentes. Cuando se le toma afición también puede resultar muy entretenido, y, en cualquier caso, nuestra paciencia se verá recompensada con resultados que ningún aficionado podría obtener. En la investigación forense de dispositivos Android existen situaciones que requieren el uso del editor hexadecimal:

- **Obtención de metadatos**: aunque existen herramientas automatizadas para esta tarea, no funcionan con todos los tipos de archivo, o no son capaces de extraer todo lo que hay en el interior de un documento o una imagen. No hay etiqueta que pueda esconderse del editor hexadecimal.
- **Examen de archivos de bases de datos** en busca de registros eliminados que el cliente no es capaz de mostrar.
- **Investigación de archivos borrados** en una tarjeta SD reformateada o en una imagen forense adquirida físicamente.
- **Examen de particiones y sistemas de archivos** (por ejemplo YAFFS2), para las cuales no existen herramientas de análisis.
- **Búsqueda de caracteres** en zonas del soporte de almacenamiento de datos no asignadas por el sistema de archivos.
- **Análisis de *malware*** y código malicioso.
- **Reconocimiento y análisis de archivos camuflados** mediante un cambio de extensión o provistos de firmas (números mágicos) manipuladas.

Hay muchos editores hexadecimales y cualquiera de ellos sirve, con tal que cumpla algunos requerimientos básicos: mostrar los datos del archivo en bruto, permitir búsquedas de caracteres —tanto en texto como en código hexadecimal—, marcar selecciones y guardar los datos cortados en el disco duro.

Todos los entornos Linux/Unix disponen de un editor hexadecimal en línea de comando, pero su manejo puede resultar complicado, sobre todo para el principiante. Por esta razón se recomienda empezar con una utilidad gráfica como HxD para Windows (Figura 5.18) o Ghex para Linux.

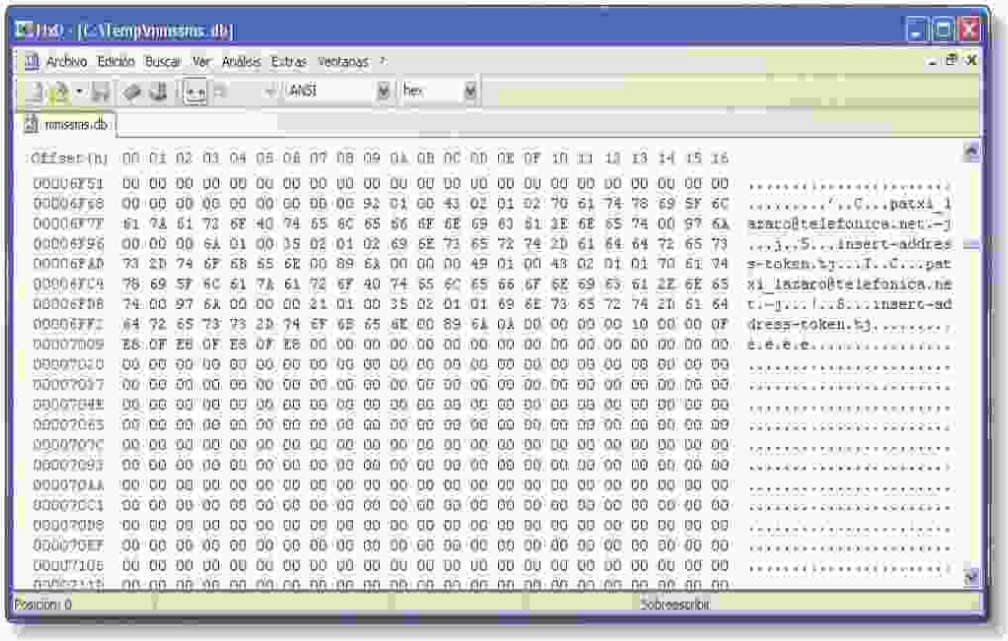

Figura 5.18. Archivo de bases de datos SQLite mmssms.db visto a través de HxD

5.6.2 Strings

El comando `strings`, incluido por defecto en la mayor parte de las distribuciones Linux, extrae secuencias de caracteres de cualquier archivo binario o de texto. En archivos de gran tamaño, la salida de `strings` consiste en una larga serie de líneas que nos veremos obligados a redirigir a un archivo de texto. También podremos examinar la salida mediante concatenación con el comando `less`:

```
igandekoa@igandekoa:~/FORENSICS/LIBRO_ANDROID/SQLite$ strings
mmssms.db | less
```

Existen varias posibilidades de uso para el comando `strings`. Mediante la opción `-radix`, por ejemplo, se muestra el *offset* o desplazamiento (un número en código hexadecimal) de la línea en que aparece la secuencia de caracteres. Esto permite combinar `strings` con un editor hexadecimal para localizar datos específicos y llevar a cabo búsquedas complejas. Otras opciones permiten la búsqueda de caracteres de texto en diversos estándares de codificación. Por defecto `strings` trabaja con texto ASCII simple de 7 bits. Mediante `--encoding` podemos buscar texto en formatos especiales, como, por ejemplo, ASCII de 8 bits, caracteres de tablas extendidas (UTF) o correspondientes a alfabetos no europeos. Para una información más amplia sobre las posibilidades de `strings` se recomienda consultar la página de manual en Linux:

```
igandekoa@igandekoa:~$ man strings
```

5.7 YAFFS/YAFFS2

Si este libro hubiera sido escrito hace dos o tres años, el autor habría dedicado mayor número de páginas al sistema de archivos YAFFS (*yet another flash file system*), desarrollado a propuesta de la compañía de software neozelandesa Aleph One en 2001. YAFFS es el primer sistema de archivos optimizado para soportes de memoria NAND. Su versión perfeccionada YAFFS2 se utilizaba en las particiones de almacenamiento de los primeros dispositivos Android. Actualmente, YAFFS2 está desfasado y apenas se emplea en *smartphones* o tabletas. Los fabricantes de terminales modernos como Samsung, HTC o Sony, y también Google en su línea de dispositivos Nexus, prefieren utilizar el sistema de archivos ext4 originario del entorno Linux, principalmente por razones de compatibilidad funcional con el hardware de nueva generación. YAFFS/YAFFS2 no funciona bien con los procesadores de núcleo múltiple. Esto lo vuelve obsoleto para la práctica totalidad de los dispositivos que salen al mercado en la actualidad. Sin embargo, es posible que en el transcurso de su trabajo el investigador se vea en la necesidad de analizar alguno de los numerosos dispositivos antiguos que todavía están en uso. Por este motivo conviene saber algo sobre este sistema de archivos.

YAFFS/YAFFS2 está diseñado para realizar operaciones de escritura, borrado y actualización de datos de acuerdo con las características funcionales de los chips de memoria NAND. Asegura la integridad de los archivos mediante un registro de transacciones con reparto uniforme de actividad que garantiza un desgaste homogéneo de las puertas lógicas del chip, alargando de este modo la vida útil del hardware. Su interacción con el controlador de memoria MTD es simple y directa, con un flujo de datos adaptado al tamaño de los fragmentos y los bloques que forman la memoria NAND. Esto le permite alcanzar altos niveles de rendimiento y fiabilidad.

Aunque YAFFS/YAFFS2 dispone de soporte en Linux, su abandono en dispositivos nuevos ha hecho que nadie se esfuerce en desarrollar herramientas de investigación forense destinadas al análisis de este sistema de archivos. Las particiones de sistema y datos de usuario adquiridas desde terminales antiguos y formateadas con YAFFS2 deben ser exploradas con un editor hexadecimal o sometidas a operaciones de *data carving* con Scalpel o Foremost. De hecho, nos vemos obligados a interactuar con particiones YAFFS/YAFFS2 de modo similar a cuando recuperamos archivos borrados. Incluso en este último caso la operación dista de ser trivial, puesto que antes es preciso eliminar los sectores intermedios con metadatos que YAFFS incluye para tareas de gestión del sistema de archivos. Para este fin se utilizan *scripts* en Python u otros lenguajes de programación, que eliminan dichas partes y reensamblan la imagen creando un espacio continuo ocupado únicamente por los datos de los archivos.

6

SEGURIDAD

Aunque la labor del investigador forense consiste en extraer elementos de evidencia, conocer el concepto de seguridad Android y los métodos de los que los delincuentes informáticos se sirven para lanzar sus ataques resulta útil por varias razones. En primer lugar, el conocimiento de técnicas delictivas facilita la localización de los datos del sospechoso, y, por supuesto, también de la información robada, oculta o eliminada por aquel. Además se podrá valorar la posibilidad de que hayan podido producirse determinados acontecimientos —intrusiones en redes, troyanización de terminales, robos de identidad—. Las nociones de seguridad resultan muy útiles para todo aquel que tenga que ver con el entorno de los dispositivos móviles, desde el desarrollador hasta el usuario.

6.1 MODELO DE SEGURIDAD ANDROID

La informática móvil plantea problemas de seguridad más complejos que los PC de sobremesa y las redes locales. Los dispositivos utilizados por sospechosos y víctimas ofrecen prestaciones y capacidades de almacenamiento similares a las de los PC de hace cinco o seis años (que ya eran unas máquinas muy potentes), pero caben en la palma de la mano y son tan fácilmente transportables como cualquier artículo de uso personal. Sus precios asequibles y su facilidad de manejo los ponen al alcance de todo tipo de usuarios, incluyendo personal no cualificado, escolares, marginados sociales y colectivos de los que hace diez años jamás habríamos pensado que algún día pudieran llegar a tener algo que ver con el mundo de las telecomunicaciones y la informática. La conectividad es ilimitada: GSM, redes 3G/4G, wifi, Bluetooth, cable USB, interfaces HDMI, etc. El usuario permanece todo el tiempo conectado a redes públicas y privadas.

Hablar de las vulnerabilidades de las comunicaciones móviles es algo que trasciende el alcance temático y la intención de la presente obra. La inseguridad no se limita al sistema operativo y a las aplicaciones: comienza con el despliegue de la misma tecnología de base que hace posible los enlaces telefónicos por vía aérea y a través de redes GSM, CDMA, 3G y 4G. Tampoco es nuestro propósito contribuir a la polémica en torno al tema ni generar preocupación. Opinamos que el público ya tiene suficiente ración de susto con lo que puede ver por ahí todos los días. Por consiguiente, nos limitaremos al objeto principal de nuestro estudio, con el propósito de apoyar lo explicado hasta el momento sobre la investigación forense de plataformas Android, más que con el de ayudar al usuario a aumentar sus conocimientos en materia de seguridad. Con respecto a esto último, baste decir que con la facilidad de uso y el carácter social de los dispositivos móviles, peca de ingenuo quien no espere para el futuro mayores riesgos relacionados con incidentes de seguridad.

6.1.1 Retos de seguridad en la era móvil

En un PC de sobremesa con Windows, Apple OSX o Linux, el usuario que ha completado de manera correcta la instalación de su software trabaja dentro de un entorno de permisos ajustado a los requerimientos de su actividad y a un nivel de privilegios suficientemente restrictivo. Dentro de este entorno, lo normal es que utilice un conjunto de aplicaciones informáticas de productividad más o menos interdependientes (Office, programas de contabilidad, entornos de desarrollo, herramientas de edición de vídeo o retoque fotográfico) que comparten un mismo espacio de memoria RAM y almacenamiento en disco duro.

El riesgo de que las aplicaciones interfieran o se ataquen unas a otras, por ejemplo con el objeto de llevar a cabo un proceso de ampliación de privilegios que permita el acceso a un atacante externo, es reducido, o en todo caso está siempre bajo control, debido a la ubicación relativamente fija del ordenador y a la circunstancia de que en la mayor parte de los casos el software procede de fuentes fiables: viene incluido de fábrica en el PC, ha sido instalado por el administrador del sistema o lo descarga el usuario desde un repositorio oficial de Linux.

Sería desastroso dar por supuesto que este también es el caso en entornos de informática móvil, donde los dispositivos están diseñados para que conexión con Internet sea permanente y el usuario pasa mucho tiempo descargando aplicaciones o comunicándose con sus conocidos a través del correo electrónico y programas de mensajería. Los problemas de seguridad se resuelven de diferentes modos según la filosofía y el diseño de software de cada fabricante.

Apple solamente permite la descarga de aplicaciones desde un sitio autorizado —el Apple Store— en el que los desarrolladores están identificados y el software pasa a través de los filtros de control de calidad de la empresa. Las técnicas de *jailbreaking* permiten manipular terminales para obtener privilegios de superusuario y autorizar la instalación de aplicaciones procedentes de fuentes no fiables. Pero, en términos generales, este monopolio articulado sobre el Apple Store resulta suficiente para garantizar la seguridad de un usuario normal que se limite a disfrutar de una experiencia móvil poco ambiciosa y sea respetuoso con los términos de la garantía. También mantiene a la empresa a salvo de responsabilidades civiles y daños de imagen.

En Android el concepto de seguridad resulta algo más complejo y ecléctico. Esto es inevitable debido a las características del software libre, como la libertad del usuario para modificar la configuración de los dispositivos y utilizar todo tipo de aplicaciones, desarrolladas por él mismo o descargadas de los sitios que desee, independientemente de que estén o no bajo el control de Google y la Open Handset Alliance.

Para conseguir este difícil equilibrio entre seguridad y libertad, los desarrolladores de Android se han visto obligados a coordinar todo el herramental disponible en el entorno Linux/Unix y las diferentes tecnologías en las que se basa el proyecto (Java, máquinas virtuales, etc.). El resultado es un concepto de seguridad en el que el sistema de permisos Android se combina con las firmas digitales y la ejecución en entornos restringidos de características similares a las de una máquina virtual Java.

6.1.2 Certificados digitales y supervisión

Toda aplicación puesta a disposición del público en el sitio oficial de Android —originariamente Android Market, luego Play Store y en la actualidad Google Play— dispone de un certificado que identifica al autor. Esta firma digital, basada en el sistema de doble clave (pública y privada) tiene la misión de atestiguar que los archivos de la aplicación no han sido modificados. Siempre que el desarrollador introduzca cambios en su aplicación deberá crear un nuevo certificado, algo que solo está al alcance de quien posee la clave privada.

Google no exige que los certificados para Google Play tengan que haber sido creados por una autoridad de certificación. El desarrollador es libre para utilizar los suyos, aunque en la práctica se emplean los que el entorno de desarrollo —por ejemplo Eclipse— genera al introducir la aplicación en el paquete *.apk*.

El sistema de certificados sirve para validar la procedencia del software y controlar las descargas desde sitios no oficiales. También impide que una aplicación pueda acceder de manera no autorizada a los datos de otras. Como norma general solo se permite compartir datos a aquellas aplicaciones cuya firma digital coincida y puedan así demostrar que tienen la misma procedencia o han sido escritas por el mismo desarrollador. Igualmente, existen permisos de ejecución y de acceso a recursos que son exclusivos de las aplicaciones del sistema operativo o del fabricante del terminal. Cualquier programa que solicite un nivel de privilegios alto sin disponer de un certificado que coincida con el de la aplicación suplantada será bloqueado de inmediato por Android.

Los certificados digitales permiten aplicar una política eficaz de vigilancia y son útiles para combatir el software no autorizado. Google borra de su mercado oficial las aplicaciones que considera sospechosas o las que no hayan cumplido determinados requisitos de calidad. También puede eliminarlas directamente del terminal del usuario.

6.1.3 Máquina virtual Dalvik

Cada aplicación Android se ejecuta en su propia máquina virtual Dalvik. La razón para no utilizar el *runtime* o entorno de ejecución Java es que Dalvik está diseñada para entornos con recursos reducidos, como dispositivos móviles y aparatos alimentados con baterías. Los microprocesadores consumen gran cantidad de energía, sobre todo si tienen que gestionar varias instancias de un software de virtualización en una plataforma multitarea, no digamos ya en condiciones aceptables de seguridad, con sistemas de permisos y mecanismos adicionales de vigilancia. Todo ello funciona en última instancia mediante líneas de código que tienen que ser ejecutadas por la CPU. Un diseño esmerado, racionalización del código y bajo consumo de recursos tienen una importancia crucial a la hora de crear productos de alta calidad y atractivos para el consumidor.

Dentro de su máquina virtual Dalvik la aplicación crea su propio entorno de ejecución con un espacio de direcciones en memoria al cual solo ella puede acceder. De este modo el software funciona dentro de una zona reservada (*sandbox*), protegido contra otras aplicaciones concurrentes. En caso de que algún proceso externo quiera acceder a los datos, deberá cumplir determinados requisitos de autorización y poseer los certificados digitales necesarios.

6.1.4 Gestión de usuarios

Android dispone de dos mecanismos de seguridad basados en el diseño tradicional de los sistemas operativos Linux/Unix: gestión de usuarios POSIX (*portable operating system interface*; interfaz de sistemas operativos portables) y un régimen de autorizaciones de uso de archivos con control discrecional de acceso (Discretionary Access Control [DAC]). El funcionamiento coordinado de estos dos mecanismos resulta posible mediante la asociación de un identificador de usuario (UID) a cada proceso que se ejecuta en el sistema (PID).

En un PC de sobremesa convencional el UID se asocia a un usuario que por haber sido dado de alta en el sistema dispone de autorización para utilizar determinadas aplicaciones dentro del entorno de ejecución que se le asigna. La originalidad del concepto de seguridad Android reside en el hecho de que el sistema operativo considera como usuarios no a las personas sino a las aplicaciones informáticas. Cuando una aplicación es instalada, el sistema le asigna un identificador de usuario POSIX. Todos los procesos iniciados por la aplicación pertenecen al mismo UID y se ejecutan en su espacio de direcciones de memoria. Esta caja de arena (*sandbox*), que constituye la principal base de recursos para la máquina Virtual Dalvik, mantiene a la aplicación a salvo de cualquier proceso externo ejecutado por las otras aplicaciones del sistema.

Si dos procesos pertenecientes a aplicaciones distintas necesitan intercambiar datos, podrán hacerlo a través de un método denominado `sharedUserId`. Para ello es preciso que Android asigne a las dos aplicaciones, durante su instalación, el mismo UID.

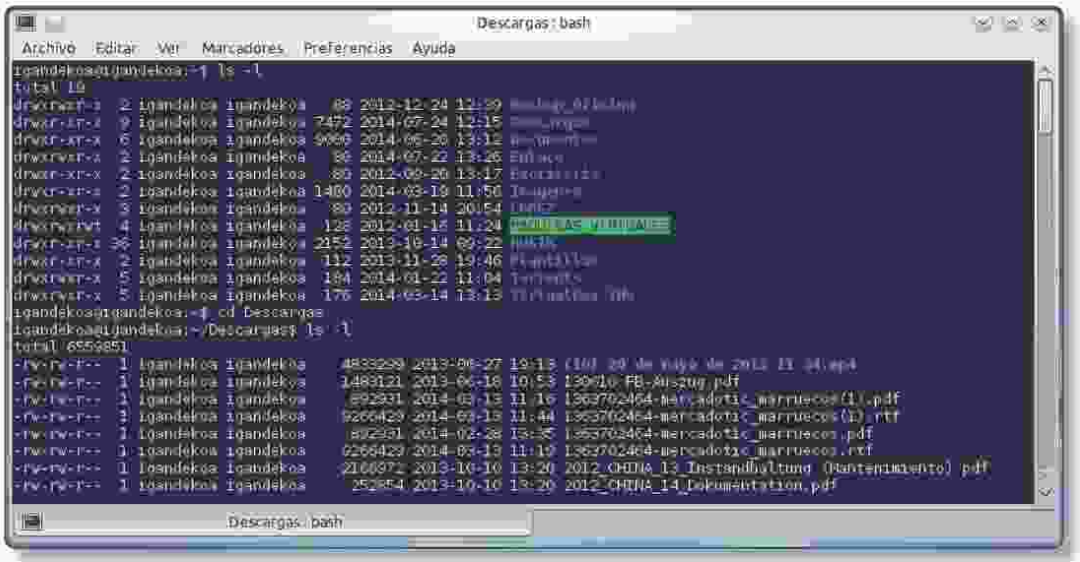

Figura 6.1. Sistema de permisos Unix/Linux

6.1.5 Permisos de archivos

En un nivel básico de acceso a ejecutables, controladores, librerías, archivos de bases de datos y otros recursos del sistema, Android utiliza el sistema de permisos Linux (Figura 6.1). Cada archivo o directorio dispone de identificadores de usuario y grupo con marcas que definen los derechos de lectura, escritura y ejecución para el propietario del archivo, los miembros del grupo y el resto del mundo. Android asigna estos derechos con criterio restrictivo: todos los archivos del sistema operativo son propiedad del superusuario (`root`), mientras que los archivos descargados con las aplicaciones pertenecen a los UID respectivos, es decir, a las aplicaciones en cuanto que usuarias del sistema.

Mediante este sistema de permisos de archivos, Android crea un entorno seguro para la ejecución de aplicaciones. Solamente aquellas que posean el mismo UID podrán acceder a los mismos datos —salvo que se trate de archivos con permisos de acceso para todo el mundo—. Los archivos del superusuario están protegidos por un mecanismo adicional: durante el arranque de Android la partición del sistema operativo se monta con permisos de solo lectura.

6.1.6 Permisos de aplicaciones

En el mundo Android, los permisos de los archivos y los permisos de las aplicaciones son cosas diferentes. Los permisos de las aplicaciones constituyen un nivel de seguridad superpuesto al sistema de permisos Unix/Linux. Su función consiste en especificar lo que una aplicación puede hacer y lo que no. Se configuran en el entorno de desarrollo y aparecen listados en el archivo `AndroidManifest.xml` para que Android sepa cuál es el grado de funcionalidad al que solicitan autorización para acceder. Esta lista de permisos establece lo que el sistema debe pedir al usuario que autorice en el momento de la instalación. Los permisos de aplicaciones no pueden especificarse con el mismo grado de detalle que los permisos de los archivos. El usuario tiene la opción de aceptarlos o rechazarlos en bloque, tras haber leído el mensaje de advertencia que le presenta en pantalla el instalador de Android.

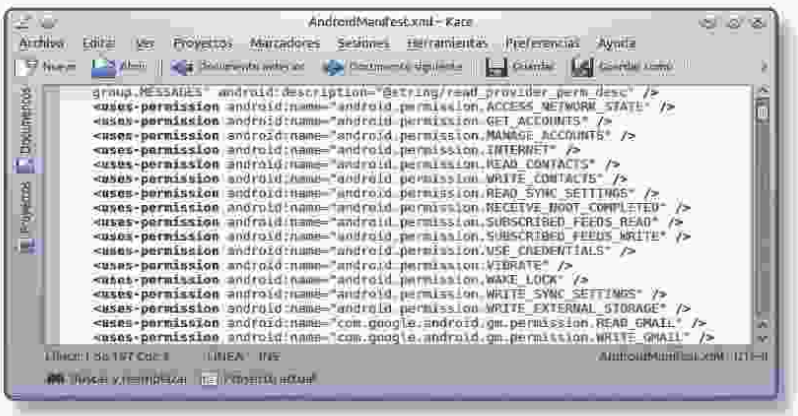

Figura 6.2. Permisos de las aplicaciones Android

El investigador forense, cuando tenga que examinar una aplicación sospechosa, deberá prestar atención a los permisos de las aplicaciones. Es normal que una aplicación de correo electrónico o mensajería solicite autorización para consultar la base de datos de contactos, utilizar el teléfono y acceder a Internet. Pero si quien pide esos permisos es un juego de marcianos o un salvapantallas, entonces hay algo que no va bien.

Android dispone de un centenar de permisos para controlar y restringir la actividad de las aplicaciones instaladas por el usuario, con posibilidad de definir otros adicionales según las necesidades del desarrollador. Cada uno de estos permisos se encuentra asociado a un nivel de protección siguiendo un orden de prioridad en cuatro niveles, de menor a mayor seguridad:

- **Permisos de nivel 0 (Normal)**: se conceden a recursos necesarios pero no tienen un carácter crítico. El sistema ni siquiera se molesta en consultar al usuario. Se asignan de manera automática, puesto que no es de esperar que originen problemas de ningún tipo.
- **Permisos de nivel 1 (Peligroso)**: estos permisos son necesarios para que una aplicación pueda acceder a recursos considerados críticos, como cambiar la configuración del sistema, realizar llamadas telefónicas y conexiones a Internet o leer contactos y datos personales del usuario.
- **Permisos de nivel 2 (Firma digital)**: se llama así a aquellos permisos que solo pueden ser asignados a las aplicaciones que dispongan del mismo certificado digital que el UID de las aplicaciones que controlan los datos o servicios a los que se pretende acceder.
- **Permisos de nivel 3 (Firma digital/Sistema)**: este tipo de permisos solo puede ser asignado a aquellas aplicaciones que estén firmadas por el mismo certificado digital que las aplicaciones del sistema operativo.

6.2 ATAQUES CONTRA ANDROID

En general, los dispositivos Android se hallan expuestos al mismo tipo de ataques que los PC de sobremesa y los ordenadores portátiles. Lo único que cambia es el escenario: si antes era estático y estaba en la oficina, sujeto a un horario laboral estándar con largas pausas nocturnas y un día de la semana establecido para copias de seguridad y mantenimiento del sistema, ahora el entorno es móvil, imprevisible, heterogéneo, caótico. El golpe de mano puede tener lugar a cualquier hora del día o de la noche. En el ámbito laboral fijo y en el hogar las intrusiones y los abusos podían ser combatibles hasta cierto punto, mediante cortafuegos, políticas de uso,

un vigilante nocturno o, simplemente, dándole al botón de apagado. Pero ¿cómo hacer frente a una amenaza que puede presentarse en cafeterías, gimnasios, aviones y palacios de congresos? ¿Cómo sobrevivir en un mundo donde el límite a la insidia del delincuente no viene determinado por la capacitación del administrador de sistemas, sino por la fantasía del delincuente?

Aunque parezca una idea extraña, el único modo de hallar la respuesta a este interrogante consiste en seguir los pasos del malhechor. Hay que ponerse en su lugar y razonar como él, aprender con el mismo empeño, por supuesto aplicado a la causa del bien, y mantenerse al día sobre el avance de las tecnologías informáticas. El atacante es un hacker de sombrero negro o algo peor. Nosotros llevamos sombrero blanco y luchamos para que prevalezcan la justicia y el Estado de Derecho. Entre ambos extremos existe una zona de sombrías transiciones en la que es posible experimentar con enfoques de *hacking* ético y psicología criminalista.

Haciendo un esfuerzo pedagógico de sombrero gris, y tras haber llevado a cabo el esfuerzo de ponernos en el lugar de un ciberdelincuente que intenta abrirse paso a través de las defensas de la red, no tardaremos en darnos cuenta de que existen tres vectores o modalidades principales de amenaza contra plataformas Android: ataques de infraestructura, ataques de hardware y ataques de software (Figura 6.3). Cada tipo de agresión requiere metodos específicos y plantea al delincuente la necesidad de hacer las cosas de un modo determinado.

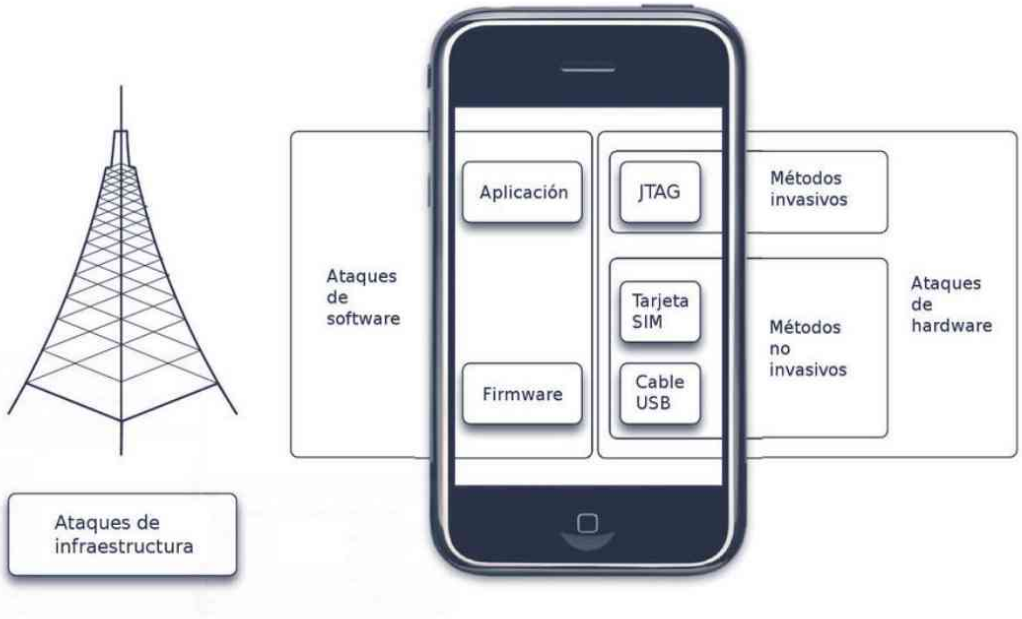

Figura 6.3. Tipología de los ataques contra plataformas Android

6.2.1 Ataques contra la infraestructura

Este tipo de ataque consiste en una agresión contra cualquiera de aquellos elementos de infraestructura utilizados por los dispositivos móviles para comunicarse entre ellos o con otros sistemas conectados a una red de comunicaciones: teléfonos fijos, servidores de Internet, *proxies*, *hosts* de instalaciones corporativas, etc. Independientemente de que se encuentren situados en domicilios particulares, bloques de oficinas o dentro de un tren de alta velocidad, todos estos aparatos se comunican entre sí a través de un entramado compuesto por antenas de telefonía móvil, tendidos de fibra óptica, concentradores, *routers*, *switches* multinivel y otros elementos de hardware típicos de cualquier red de telecomunicaciones moderna.

Los ataques contra la infraestructura permiten a un agresor causar daños de la más variada índole: interceptar transmisiones, suplantar identidades o dejar sin servicio al usuario. Además de un riesgo para la privacidad, el ataque contra elementos de infraestructura hace posible que un atacante amplifique el efecto de su agresión dejando a un gran número de dispositivos desconectados de la red. Esto va más allá del robo de datos y las molestias individuales; son palabras mayores, pues supone un peligro cierto para los mismos fundamentos de la vida moderna, al interrumpir comunicaciones y servicios que pueden necesitarse con urgencia.

6.2.2 Ataques contra el hardware

El ataque contra el hardware es fácil de definir, puesto que está dirigido hacia algo que se puede ver y tocar. Las implicaciones son complejas en función de los usos particulares o institucionales a los que están dedicados los dispositivos. Sumadas con otras, cuando gran número de terminales se ven afectados por prácticas de robo o manipulación, o simplemente terminan perdiéndose por ahí, el problema adquiere al final dimensiones macroeconómicas y sociales. De acuerdo con una encuesta de la empresa de seguridad informática Avast, durante el año 2012 un 22 % de los usuarios españoles extravió un teléfono móvil o un *smartphone*. En un parque móvil de 25 millones de terminales esto significa que al menos 5 millones de dispositivos cambiaron de manos de un modo irregular.

Pensar en las pérdidas reduciéndolas al valor económico de reposición resulta ingenuo: los datos también tienen valor, en ocasiones considerable. ¿Qué hay de la información personal del titular y de los usuarios con los que se comunicaba y que estaban almacenados en el terminal sustraído? ¿A qué manos han ido a parar? La indiferencia con que una inmensa mayoría de usuarios encara este problema y el escaso eco alcanzado en los medios sugiere que nos hallamos lejos de superar el viejo prejuicio materialista según el cual el valor económico reside en objetos con

propiedades de masa y volumen. Esta forma de ver las cosas ya no es aplicable en la sociedad de la información, donde lo que importa son activos intangibles, como datos personales, identidades, tecnología y *know-how*.

Dentro de los ataques físicos se puede diferenciar entre dos variantes:

- **Intrusivos**: implican la modificación física del dispositivo en una escala tal que con frecuencia lo dejan inutilizado o hacen imposible que pueda seguir funcionando en las condiciones originales de fábrica. Cuando se trata de llegar al contenido del terminal, el ejemplo típico de este tipo de ataques sería la extracción de chips o la soldadura de cables a las patillas de aquellos para realizar lecturas de memoria a través de JTAG.

- **No intrusivos**: un ataque es de carácter no intrusivo cuando el agresor no altera el estado de fábrica del dispositivo, sino que se limita a introducir modificaciones en la configuración del software o del sistema operativo con la intención de alterar el funcionamiento del terminal. Hemos visto ejemplos de estas manipulaciones al tratar sobre técnicas de *rooting*, ROM cocinadas y particiones Recovery alternativas. También son ataques no intrusivos la retirada de tarjetas SIM y/o de la memoria SD con fines de adquisición de datos, el intento de descifrar códigos de bloqueo por tanteo, fuerza bruta u otros procedimientos y la adquisición de datos a través de una conexión USB. El objetivo de un ataque no intrusivo consiste en conseguir acceso a una información almacenada en el dispositivo que se considera valiosa o interesante, reservando el hardware para valorizarlo en mercados clandestinos o dedicarlo a usos delictivos de algún otro tipo.

6.2.3 Ataques contra el software

Un ataque de software es aquel que va dirigido contra el sistema operativo o las aplicaciones que se ejecutan en el dispositivo móvil. Algunas de estas agresiones pueden ser tan burdas como persuadir al usuario, mediante maniobras de ingeniería social, para que lleve a cabo él mismo las manipulaciones que comprometen su terminal (por ejemplo, mediante descarga de aplicaciones malignas y concesión de permisos demasiado amplios). En el aspecto puramente técnico los ataques contra el software están basados en la explotación de un defecto de programación (*bug*) en el sistema operativo del terminal, en las aplicaciones de usuario o en los *drivers*.

Más adelante hablaremos sobre las vulnerabilidades de Android que hacen posible la existencia de los diferentes tipos de ataque utilizados por los delincuentes

informáticos. En los ataques contra el software de dispositivos móviles existen dos variantes principales:

- **Ataques de aplicación**. El agresor engaña al usuario legítimo de un dispositivo para que descargue desde sitios no homologados por Google una aplicación maliciosa que se hace pasar por software legítimo. Una vez instalada, esta aplicación cumple su propósito maligno por uno de los tres métodos siguientes: a) a través de los permisos excesivos concedidos por un usuario inexperto; b) explotando un *bug* de la versión de Android sobre la que se instala; o c) aprovechándose de los permisos que poseen otras aplicaciones legítimas.

- **Ataques desde la Web**. Se trata de una modalidad de agresión cada vez más utilizada por los delincuentes informáticos. En este caso el vector de acceso al dispositivo suele ser el motor de navegación WebKit. Los navegadores de Internet, además de interpretar código HTML, incluyen funcionalidades complejas para la ejecución de *scripts* en diversos lenguajes de programación, así como para visualizar archivos multimedia, realizar llamadas de videoconferencia, utilizar servicios web y otros cometidos. Para implementar estos extras son necesarios componentes de software adicionales (*plugins*). También se precisa la cooperación concurrente de otras aplicaciones vinculadas al programa atacado mediante relaciones de confianza y permisos compartidos. Pulsando sobre los enlaces de páginas web controladas por el atacante, o simplemente por el mismo hecho de visualizarlas, el usuario puede estar descargando una aplicación maligna o ejecutando un *script* con capacidad para explotar vulnerabilidades que hacen posible el robo de datos, la troyanización del terminal o cualesquiera otros efectos indeseables.

6.3 VULNERABILIDADES ANDROID

Con lo expuesto en el apartado anterior nos podemos hacer cierta idea de cómo emplea su talento un pirata especializado en el ataque contra objetivos móviles. Lo que aún no sabemos es cómo lo hace. Para ello es preciso explicar con cierto detalle las condiciones que hacen posible la existencia de los diferentes tipos de agresión. Interesan principalmente los ataques de software.

Lo que sucede en la infraestructura de red pertenece al ámbito de otras tecnologías informáticas y existen métodos de investigación específicos para ocuparse del problema. Las manipulaciones del hardware, por su parte, requieren

enfoques de alto nivel tecnológico, de los cuales ya hemos hablado al tratar sobre los métodos de investigación intrusivos como *chip-off* o JTAG. El software, por el contrario, está al alcance de cualquier investigador que lo conozca y sepa cómo detectar dentro de él las anomalías y los fallos que impiden un funcionamiento correcto de las aplicaciones.

Pese a su carácter enteramente lógico y determinista, el software es un producto hecho por el hombre y por tanto sujeto a imperfecciones. Tanto en Android como en cualquier otra plataforma informática, la complejidad y el tamaño de los programas hacen posible la existencia de vulnerabilidades con diversos grados de severidad que, explotadas por un atacante, abren camino a infinidad de maniobras maliciosas cuyo último objetivo consiste en robar datos, asociar el terminal a una *botnet* o extraer beneficios económicos ilícitos.

Como en cualquier otro sistema operativo, en Android no dejan de aparecer vulnerabilidades nuevas. La naturaleza de Android como código libre hace posible solucionar estos defectos en menos tiempo del que se tarda en reparar fallos similares en el software propietario. La razón de ello es que todo el mundo puede examinar el código fuente y siempre hay un programador dispuesto a poner a disposición de la comunidad nuevas versiones del software con los fallos corregidos.

En los apartados siguientes vamos a ver algunos puntos débiles que los delincuentes informáticos utilizan para atacar plataformas móviles equipadas con el sistema operativo de Google.

6.3.1 Vulnerabilidades de permisos nulos

En el momento de instalarse una aplicación, esta solicita los permisos que necesita para realizar sus funciones. Si el usuario no está dispuesto a concedérselos, la aplicación es rechazada por el sistema operativo. Las aplicaciones que no necesitan permisos, y que por tanto no han de solicitarlos, se instalan de manera inmediata y sin pedir confirmación al usuario, al darse por entendido que este tipo de aplicaciones no plantean ninguna amenaza potencial.

Dicha suposición es, sin embargo, más arriesgada de lo que parece. Las vulnerabilidades de permisos nulos (*zero permission flaws*), descubiertas por Tim Wyatt en 2010, permiten acceder a Internet y llevar a cabo un reinicio del terminal, una broma molesta que en determinadas circunstancias puede llegar a ser peligrosa. Y ello aunque la aplicación maliciosa permanezca confinada dentro de la máquina virtual Dalvik. Veamos por qué:

- **Acceso a Internet**. El objativo malicioso consiste en establecer una comunicación entre el software instalado en el dispositivo y un servidor remoto controlado por el atacante. En condiciones normales, para hacer esto una aplicación necesita el permiso `INTERNET`, o bien tener acceso a otra aplicación que lo tenga. En el caso de las aplicaciones de permisos nulos, la forma en que aquellas consiguen abrirse paso hasta el exterior consiste en pasar una URL (la dirección de un objeto situado en Internet) a través de un *intent*. El navegador de Internet transfiere a la red la URL acompañada de sus parámetros en forma de una petición `HTTP GET`. De este modo, el agresor establece el enlace ascendente (*uplink*), y, por consiguiente, una vía de comunicación desde el dispositivo hasta el servidor remoto.

  Para hablitar el enlace descendente (*downlink*) desde el servidor remoto hasta el terminal, el atacante define un receptor estándar para identificadores uniformes de recursos (URI) y lo da de alta mediante la entrada correspondiente en el archivo `AndroidManifest.xml` de su presuntamente inofensiva aplicación. Este receptor inicia la aplicación cada vez que se establece una conexión con el servidor remoto, transfiriendo a la aplicación todos los parámetros de la URL.

  Mediante este proceso en dos etapas, solo identificable mediante un análisis detallado del código de la aplicación maliciosa, resulta posible establecer conexiones remotas —similares a las de troyanos y *botnets*— sin que sea necesario disponer del permiso `INTERNET`.

- **Reinicio del dispositivo**. Toda aplicación que desee reiniciar el sistema necesita disponer del permiso `REBOOT`. Sin embargo, aunque el usuario lo conceda, el sistema no lo aceptará porque `REBOOT` es un permiso perteneciente al nivel de protección 3, propio de las aplicaciones de sistema provistas de firma digital. Por consiguiente, las aplicaciones de permisos nulos no son capaces de forzar directamente un reinicio del sistema. Lo que sí pueden hacer, sin embargo, es emitir una notificación de tipo especial, denominada `Toast`, capaz de persuadir al usuario para que lleve a cabo él mismo un reinicio del sistema operativo. Las notificaciones `Toast` son mensajes emergentes utilizados por los desarrolladores con fines de depuración de software o para mostrar algún aviso al usuario. Se puede conseguir que una de estas notificaciones aparezca de modo intermitente. Al no existir otra posibilidad de anularla, ya que no admite interacción, el usuario cree que se trata de un fallo de software y reinicia el terminal para librarse de las molestias por el inoportuno y tozudo mensaje de advertencia.

El reinicio del dispositivo permite completar procesos de instalación del software malicioso descargado por las aplicaciones de permisos nulos, haciendo permanentes los cambios que aquellas hayan podido producir en la configuración del sistema operativo.

6.3.2 Escalada de privilegios

Otra categoría de vulnerabilidades está compuesta por aquellos fallos de programación o configuración del software que hacen posible conseguir un aumento de privilegios mediante la ejecución de determinados fragmentos de código malicioso o *exploits*. Este defecto no es exclusivo de Android sino que está presente en todos los sistemas operativos desde los inicios de la computación. En Android los *exploits* de escalada de privilegios más conocidos son los que se muestran en la tabla 6.1. Se utilizan para rootear dispositivos móviles sin que la víctima se dé cuenta, con el objeto de descargar software malicioso, robar datos del usuario o controlar el terminal desde una ubicación remota.

Denominación	Vulnerabilidad	Fecha de publicación	Versión Android
Droidsploit	CVE-2009-2692	Ag. 2009	<= 1.6
moosecox	CVE-2009-3547	Nov. 2009	<= 1.6
exploid	CVE-2009-1185	Jul. 2010	2.1
RageAgainstTheCage	CVE-2010-EASY	Ag. 2010	2.2
Zimperlich	CVE-2010-EASY	Dic. 2010	2.2
zysploit	CVE-2010-EASY	Dic. 2010	2.2
KillingInTheNameOf	CVE-2011-1149	Ene. 2011	<= 2.1
psneuter	CVE-2011-1149	Ene. 2011	<= 2.1
GingerBreak	CVE-2011-1823	Abr. 2011	<= 2.3.7
zergRush	CVE-2011-3874	Oct. 2011	<= 2.3.7
Levitator	CVE-2011-1350 / 1352	Nov. 2011	<= 2.3.6
mempodroid	CVE-2012-0056	Ene. 2012	>=4.0
-	CVE-2013-4777	Sept. 2013	<= 2.3.7
-	CVE-2013-5933	Sept. 2013	<= 2.3.7

Tabla 6.1. Vulnerabilidades de escalada de privilegios

6.3.3 Ejecución de shell inverso

En este caso el objetivo malicioso consiste en abrir en el terminal un *shell* o cuenta que permita introducir comandos de Linux. Los ataques suelen tener éxito a causa de determinados defectos de programación vinculados al navegador de Internet WebKit, que no es capaz de validar entradas relacionadas con el procesamiento de código JavaScript. Existe, por ejemplo, una función encargada de interpretar datos numéricos en coma flotante que produce desbordamientos de *buffer*. A través de una página web dominada por el atacante, el *exploit* puede ser activado cuando el usuario la visita, atraído por un enlace en el correo electrónico o engaños de ingeniería social.

La ejecución del *exploit* produce un desbordamiento de *buffer* con comandos que abren el *shell* y establecen una conexión entre el terminal y el servidor remoto, así mismo en poder del atacante. A partir de este momento, el dispositivo móvil de la víctima queda bajo el control del intruso y preparado para ejecutar los comandos que el mismo introduce a través del *shell* inverso.

La lógica de este ataque es similar a la que emplean *botnets* y troyanos para expandirse a través de redes protegidas con cortafuegos. Al ser la misma víctima la que propicia el ataque cuando se conecta desde el interior de la red a una ubicación remota, sirviéndose para ello de canales HTTP que el cortafuegos no puede bloquear —ya que de lo contrario sería imposible la navegación por Internet—, no es necesario hacer barridos de puertos en busca de una fisura que permita acceder al interior de la red. Si la apertura del *shell* inverso tiene éxito en un terminal rooteado, el atacante podrá acceder a la totalidad de su contenido al disponer de permisos de superusuario. Aunque no sea así, el *shell* remoto le permitirá apoderarse de todo lo que esté guardado en la tarjeta externa SD y en la zona de almacenamiento FAT emulada en NAND.

6.3.4 Grietas funcionales (capability leaks)

Estos puntos débiles se hallan vinculados a aplicaciones de sistema que vienen instaladas de fábrica en el terminal y exportan receptores de acceso público a los cuales cualquier aplicación puede enviar mensajes por medio de *intents*. Todo aquello que llega al receptor es ejecutado por la aplicación de sistema correspondiente. Es necesario aclarar que la intención original del programador es buena: se trata de facilitar el funcionamiento de ciertos servicios y rutinas sin que la supervisión del administrador suponga un obstáculo permanente. Sin embargo, estos atajos abren brechas a través de las cuales un atacante puede acceder a la información del dispositivo, mandar mensajes SMS sin el permiso del usuario o determinar la posición geográfica del terminal.

6.4 SOFTWARE MALICIOSO

El *malware* es el pecado original de la informática de usuario. De un tiempo a esta parte, como resultado del desarrollo de Internet y de las constantes innovaciones en el desarrollo de la tecnología informática, el software malicioso ha experimentado una mutación tanto cuantitativa como cualitativa. Antes teníamos un ecosistema clásico formado por virus, gusanos y troyanos, que afectaba casi en exclusiva al PC de sobremesa y a los ordenadores portátiles provistos de alguna versión del sistema operativo Windows. Detrás de la creación de estos elementos de software maligno había, en la mayor parte de los casos, un ansia infantil de autoafirmación o motivaciones vandálicas. Ahora, gracias a las redes móviles, el entorno ha evolucionado y vivimos en una jungla digital infestada de especímenes a cual más raro y dañino: *rootkits*, puertas traseras, *botnets*, registradores de teclado o *keyloggers*, programas espía, ladrones de identidad y otras aplicaciones perversas. Esta fauna evoluciona a través de un proceso de recombinación constante de código que hace posible la aparición de nuevas variantes. No pasa día sin que aparezca un nuevo ejemplar.

6.4.1 Motivación del delincuente

Entre aquella tropa de espontáneos y las hordas bárbaras digitales de nuestros días existe una diferencia. Los parásitos liberados por los hackers de antaño eran toscos y escandalosos. Constantemente se hacían notar con efectos exhibicionistas cuya única finalidad consistía en producir la mayor cantidad de molestia posible. Al moderno software malicioso lo que le interesa es pasar desapercibido. No solo evita fastidiar al usuario, sino que se oculta haciéndose pasar por procesos legítimos o falseando la información entregada por los comandos de administración del sistema. Su objetivo no es destruir información, sino robarla. La estrategia consiste en llevar a cabo acciones que reporten un beneficio económico al delincuente: sustracción de datos y credenciales bancarias, enganche del ordenador a una red de zombis para el envío de *spam* o el lanzamiento de ataques de denegación de servicio, creación en el disco duro de sus víctimas de repositorios ocultos de *warez* y pornografía infantil, etc.

Con la llegada del *smartphone* la problemática del software malicioso entra en una nueva dimensión a causa del traslado masivo de servicios al ámbito de las redes móviles. La disponibilidad de plataformas abiertas basadas en código libre y la existencia de un enorme parque de dispositivos móviles equipados con el mismo sistema operativo proporcionan dos de los tres ingredientes necesarios para un éxito tan sonado como el que el software malicioso ha conocido en el entorno Windows. El único factor que falta para propiciar la aparición de escenarios de contaminación masiva en Android es la existencia de oportunidades de negocio y, por consiguiente, de posibilidades para que los delincuentes informáticos puedan monetizar la actividad de sus parásitos.

Más adelante veremos algunos conceptos de negocio fraudulento en Android. Conviene familiarizarse con ellos, porque si la delincuencia informática tiene una razón de existir es que con ella se puede ganar dinero. De la viabilidad de esas oportunidades de explotación económica depende el alcance de la amenaza del software malicioso en el ámbito de las redes móviles. En las páginas que siguen pasamos revista al panorama actual del *malware* Android. Aquí solo se facilitarán al investigador algunos conceptos elementales. Si quiere mantenerse al tanto de las últimas novedades en este campo deberá informarse por su cuenta a través de Internet y los medios especializados.

6.4.2 Tipos de malware

De acuerdo con el tipo principal de amenaza se pueden distinguir algunas clases fundamentales de software malicioso. He aquí las más importantes, por orden decreciente de letalidad:

- **Malware clásico** que todavía persigue causar daños en el dispositivo o hacerse con los datos del usuario. Existen diversos métodos o vectores de instalación para programas malignos. Por lo general los delincuentes informáticos consiguen introducir sus parásitos en un dispositivo móvil explotando una vulnerabilidad de Android o persuadiendo al usuario para que él mismo los instale, por medio de mensajes de correo electrónico engañosos, falsas páginas web o alguna otra maniobra de ingeniería social.

- **Software fisgón** (*spyware*). La misión del *spyware* consiste en recolectar información del usuario o vigilar su comportamiento. La instalación se lleva a cabo de manera inadvertida mediante descargas de Internet. También existe la posibilidad de que el *spyware* haya sido instalado en el dispositivo por medios manuales durante un descuido del propietario.

- **Software gris** (*grayware*). El propósito de este tipo de aplicaciones parásitas consiste en espiar al usuario, quien generalmente las descarga pensando que se trata de software inocuo. Una vez instalado, el software parásito se dedica a observar qué otros programas tiene el usuario instalados en su terminal, sus hábitos de navegación en Internet, sus desplazamientos geográficos o cualquier otra información susceptible de ser registrada en el transcurso del funcionamiento normal de Android. Posteriormente, estos datos son transmitidos a una ubicación remota para fines publicitarios o de marketing.

- **Software fraudulento** (*fraudware*). Esta modalidad de software malicioso, al no producir cambios en la configuración del sistema, carece de efectos adversos directos. Lo único que hace es engañar al usuario para que envíe mensajes SMS de pago con el objeto de obtener el código de descarga de una aplicación que le permita ampliar la funcionalidad de su dispositivo móvil. Con frecuencia el *fraudware* es incluso legal y cumple el requisito de informar —en letra pequeña— sobre los gastos adicionales. Su objetivo no es la CPU del teléfono sino la mente del usuario.

6.5 VECTORES DE ATAQUE

Se denomina vector a cualquier procedimiento o canal que sirve para instalar un programa maligno en equipos informáticos. A veces el atacante alcanza su objetivo por medios tan simples como tomar prestado el terminal y conectarlo a un ordenador por cable aprovechando un descuido del usuario. En otras ocasiones se utilizan sofisticadas maniobras de ingeniería social o páginas web controladas por el agresor. En cualquier caso, el resultado final es siempre el mismo: huéspedes no deseados a bordo del dispositivo móvil.

6.5.1 Google Play

Los usuarios instalan preferentemente sus aplicaciones descargándolas del mercado oficial Android mediante una aplicación de sistema denominada Google Play. El sitio oficial de Android es una plataforma muy grande, con más de un millón de aplicaciones de pago y gratuitas. Si el atacante consigue darse de alta en el sistema de desarrolladores de este mercado oficial y poner su mercancía envenenada a disposición de la comunidad, disfrutará de considerables ventajas. Siempre que no sea descubierto, claro está. Por suerte no lo tiene tan fácil. Todos los ofertantes del Android Market deben identificarse y firmar su software con certificados digitales que aseguran la trazabilidad en caso de problemas.

En los primeros tiempos del Android Market sí hubo problemas de infección. A través de un sistema de monitorización y gracias a la aplicación de estándares de seguridad y calidad, Google ha logrado reducir notablemente las intrusiones de código malicioso. La vigilancia en el Market es proactiva y continua, lo cual da idea de la seriedad con que Google se toma la amenaza del *malware*. Las aplicaciones son revisadas por sistemas automáticos que examinan el código en busca de instrucciones sospechosas y rastros de de *malware*. Las aplicaciones problemáticas son eliminadas. Google no solo suprime las aplicaciones maliciosas en su mercado oficial, sino que en versiones recientes de Android también puede desinstalarlas de los dispositivos móviles.

Gracias a estas medidas, el mercado oficial Android se ha convertido en un vector de infección incómodo para los ciberdelincuentes y, por esta razón, cada vez menos utilizado en los ataques contra dispositivos móviles.

6.5.2 Mercados no oficiales

Android Market no es el único repositorio de software. La naturaleza abierta del sistema operativo abre otras vías de infección a través de los numerosos portales que ofrecen software para dispositivos móviles. A diferencia de Apple iOS, donde existe tan solo una fuente autorizada para descargas de software —salvo que el iPhone haya sido liberado mediante *jailbreaking*, en cuyo caso el usuario podrá descargar también aplicaciones de una tienda no oficial denominada Cydia—, en Android el usuario puede aprovisionarse donde quiera, incluso desde un PC a través del cable USB. Los mercados no oficiales escapan al control de Google. En ellos la seguridad depende principalmente del sentido común del usuario.

Las aplicaciones maliciosas no se muestran como lo que realmente son, sino que intentan pasar desapercibidas mediante diversas estrategias de disimulo. El límite está en la imaginación del delincuente. He aquí algunos procedimientos habituales de ocultación:

- **Inserción de código maligno en aplicaciones legítimas** o *repackaging* (reempaquetado). El agresor descarga del mercado oficial Android una aplicación de pago legítima, y después de descompilarla inserta en ella su código maligno. A continuación la recompila, vuelve a introducirla en un paquete APK y la coloca en un sitio de Internet poniéndola a disposición del usuario. Este llega hasta ella atraído por la posibilidad de disfrutar gratuitamente de algo por lo que en el Android Market le piden algunos euros y la instala sin pensar. El usuario piensa que alguien la compró en el mercado oficial y la puso en ese sitio web con la generosa intención de compartirla con otros internautas, y no se le ocurre que este programa reempaquetado pueda contener líneas de código capaces de leer sus datos personales o engancharle a un programa de SMS *premium*.

  Cuanto más sofisticada es la aplicación de pago utilizada como cebo, mayor probabilidad hay de que el software malicioso pase desapercibido. Esto permite camuflar no solo las líneas de código extra necesarias para conseguir el efecto maligno, sino también los permisos de ejecución adicionales. El usuario los autoriza dando por supuesto que cuanto más potente y versátil es un programa, mayores son también los privilegios de acceso que necesita para funcionar.

- **Actualizaciones**. El programador del *malware* inserta en una aplicación legítima componentes en apariencia inofensivos que al cabo del tiempo se actualizan descargando el código malicioso.

- **Ingeniería social**. A veces es el propio delincuente el que logra persuadir al usuario de que se descargue e instale el software malicioso, mediante mensajes de correo electrónico, recomendaciones en foros y demás. La capacidad de instalar software procedente de fuentes no oficiales, combinada con la inventiva del delincuente a la hora de discurrir todo tipo de pretextos para colocar su mercancía envenenada, es algo que no debe infravalorarse en ningún momento.

6.5.3 Internet

Al igual que sucede en el caso de los PC de sobremesa y portátiles, Internet constituye un vector de ataque a causa de las vulnerabilidades del navegador y la capacidad de este para ejecutar código de procedencia externa, generalmente en forma de *scripts* incluidos en componentes dinámicos de páginas web. Imaginemos un sitio de apariencia inofensiva, con una versión para dispositivos móviles que se activa al detectar que el *host* que solicita la conexión es un terminal Android. El servidor HTTP envía al navegador una página web aparentemente normal, pero que va provista de un diminuto *frame* o marco que por su tamaño diminuto —en ocasiones hasta de un solo píxel— ni siquiera se ve en la pantalla del navegador. Este *frame* contiene código malicioso que al ejecutarse de manera automática produce los efectos previstos: alterar la configuración del dispositivo, obtener privilegios o descargar otras aplicaciones de *malware*.

Una modalidad de ataque reciente consiste en la descarga dinámica de código para ser ejecutado por aplicaciones Android. Programas en apariencia inofensivos pueden convertirse en agentes de descarga de código malicioso con una probabilidad de detección muy baja, ya que los antivirus convencionales solo pueden identificar las aplicaciones nocivas durante la descarga directa de las mismas, o, en todo caso, examinando los archivos descargados en busca de instrucciones sospechosas. En un ataque de descarga dinámica el código malicioso no es guardado en NAND ni en la tarjeta MicroSD, sino que se almacena transitoriamente en memoria RAM y es eliminado una vez cumplida su misión.

6.5.4 Infecciones combinadas dispositivo móvil-PC

Tarde o temprano el usuario de un dispositivo móvil, tanto de Android como de cualquier otro sistema, se verá obligado a utilizar el cable USB para sincronizar

con el software del fabricante o realizar algún tipo de operación a través del SDK —si se trata de un desarrollador—. El *malware*, al instalarse en una de las dos plataformas, puede generar mensajes que advierten al usuario sobre la conveniencia de actualizar componentes de software. Una vez establecida la conexión, el código maligno pasa desde el dispositivo móvil al PC o viceversa.

En la actualidad esta técnica la utilizan diversas aplicaciones malignas basadas en el concepto ZitMo (Zeus in the Mobile), aunque de momento el ataque no está automatizado y requiere una fase de ingeniería social para ejecutarlo con éxito. ZitMo intenta llevar a cabo infecciones cruzadas mostrando al usuario mensajes SMS falsos en los que se le urge a llevar a cabo actualizaciones de diversos componentes de software, facilitando al mismo tiempo el enlace de Internet para descargarlas. La actualización consiste en un archivo APK que el propio usuario debe instalar manualmente.

Los ataques de plataformas cruzadas establecen un puente entre el *malware* para ordenadores de sobremesa y las aplicaciones maliciosas destinadas a dispositivos móviles. Su capacidad para comprometer la seguridad de las redes corporativas es apreciable, sobre todo dentro de organizaciones en las que está permitido el BYOD (*bring your own device*: acceso a la red local con ordenadores portátiles y dispositivos pertenecientes a los empleados). Las infecciones cruzadas permiten crear redes de zombis híbridas Android-Windows, lo cual implica una amenaza para aplicaciones web corporativas o servicios de banca *on line* con sistemas de autenticación doble, por ejemplo del tipo mTAN, en los que el *smartphone* es utilizado para confirmar claves de acceso.

6.5.5 Problemática de los dispositivos rooteados

La adquisición de privilegios de superusuario amplía la funcionalidad del terminal, con posibilidad de acceder a todas las particiones de la memoria NAND, desinstalar componentes y *adware* del fabricante, y cambiar la partición Recovery estándar por otra customizada o con características especiales. También se pueden instalar ROM "cocinadas", con funcionalidad ampliada y un aspecto visual más atractivo que las de fábrica. La postura de Google y los desarrolladores frente al fenómeno del *rooting* es muy diferente de la de Apple en relación con las prácticas de *jailbreak*. No solo se toleran escaladas de privilegios en el dispositivo —a cambio de que el usuario renuncie a sus derechos de garantía—, sino que en cierto modo también se fomentan, haciendo pública gran cantidad de información técnica relativa a componentes de hardware y de software.

No resulta exagerado afirmar que, tarde o temprano, un usuario experimentado terminará rooteando su terminal Android. Esto no es necesariamente malo si se conocen los riesgos y se toman las necesarias precauciones a la hora de navegar por Internet. Tener un terminal rooteado no significa que uno pueda descargar e instalar cualquier cosa. El conocimiento de la amenaza y el sentido común del usuario son el mejor antídoto contra el software malicioso y las amenazas de seguridad.

Un usuario normal, sin grandes ambiciones técnicas y que solo desee utilizar su terminal para una experiencia de uso típica, no debe rootear su terminal sin más, porque haya leído en Internet o en alguna revista que se trata de algo que está de moda. El *rooting* no añade nada al pleno disfrute de la experiencia móvil. Además, los permisos de los archivos constituyen una de las bases del modelo de seguridad Android. Rootear el terminal implica la desactivación de dicho sistema de seguridad, con lo cual el dispositivo, con todos sus datos y recursos, queda expuesto al software malicioso. Los delincuentes ni siquiera tendrán que ejecutar *exploits* para obtener privilegios de administrador porque ya los tienen nada más entrar en el terminal. Todo esto agrava la problemática de la seguridad privada y empresarial, con las repercusiones negativas que son de esperar.

Ni que decir tiene que, en una investigación forense, el hecho de que un dispositivo móvil esté rooteado o no tiene un gran interés, tanto a la hora de aplicar técnicas de adquisición y análisis de elementos de evidencia como a la de elaborar hipótesis que expliquen lo sucedido. Con frecuencia el *rooting*, como elemento de evidencia en sí mismo, solo por estar el dispositivo funcionando con privilegios de superusuario, apunta a la utilización de aplicaciones maliciosas sofisticadas o a la existencia de conocimientos avanzados por parte del sospechoso.

6.6 EJEMPLOS DE APLICACIONES MALICIOSAS

En estas páginas tan solo resulta posible mencionar una pequeña parte de todo el arsenal de *malware* creado para la plataforma Android. Las molestas criaturas de este ecosistema virtual no solo se multiplican de manera incontrolada, también experimentan mutaciones. Con frecuencia los mismos productos se presentan con distintos nombres. La mayor parte de los programas parásitos derivan de unos pocos *exploits* originales. Los ciberdelincuentes copian fragmentos de código de otros programadores y los insertan en sus aplicaciones. Esto facilita en numerosos casos la identificación del *malware*. En las páginas que siguen se ofrecen descripciones de algunas funcionalidades malignas que el investigador podrá encontrar implementadas en el *malware* Android.

6.6.1 Suscripción a servicios SMS premium

Este tipo de ataque estuvo muy extendido durante los años 2012 y 2013. Debido al impacto negativo en la opinión pública, era de esperar que las compañías telefónicas tomaran medidas para evitarlos. Sin embargo, a la fecha de publicación de este libro, las esperanzas de mejora siguen estando depositadas en el sentido común del usuario y en una información adecuada acerca del tema. En cualquier caso, si un operador ofrece la opción de desactivar el marcado automático de números *premium*, el usuario debe solicitarla de inmediato, aunque todavía no haya sufrido un caso de infección. Solo así estará a salvo de las desagradables consecuencias económicas de este *malware*.

Así es como funcionan estas aplicaciones maliciosas: el atacante da de alta una línea telefónica *premium*, pagando la tarifa correspondiente al país donde tenga previsto desplegar su actividad. Luego desarrolla una aplicación capaz de marcar el número contratado y recibir mensajes a la tarifa oficial establecida por el prestador del servicio, mucho más alta que la de un SMS normal. Una vez que el importe de la llamada ha sido cargado en la cuenta del abonado, los ingresos se reparten entre la compañía telefónica y el arrendatario del número *premium*.

Durante la instalación, el software pide permisos para enviar mensajes SMS, algo que después hará sin esperar confirmación del usuario. Resulta fácil detectar la presencia de una aplicación de este tipo. Una vez descompilado el archivo `AndroidManifest.xml`, dentro de él encontraríamos algunas líneas muy parecidas a estas:

```
<uses-permission>
android:name="android.permission.SEND_SMS"
</uses-permission>
```

Para que el engaño funcione es preciso que el usuario haya facilitado su número de teléfono móvil a través de un formulario de Internet o mediante una aplicación maliciosa descargada de la web. Esto no es fácil de lograr por medios automáticos. Sin embargo, aquí es donde interviene la ingeniería social. Elementos de diseño capaces de transmitir respetabilidad corporativa, letra pequeña, premios y bonificaciones, cualquier cosa con tal de alcanzar el objetivo: conseguir que el usuario introduzca su número de teléfono y acepte las condiciones del contrato.

A partir de ese momento, el usuario comenzará a recibir mensajes *premium*, y esto se reflejará de manera desagradable en su factura telefónica. Timos de esta clase son frecuentes sobre todo en verano, ya que muchos usuarios pasan largo tiempo lejos de su residencia habitual y las facturas se acumulan en el buzón sin que tengan ocasión de revisarlas hasta que regresan de las vacaciones. Reclamar al operador

telefónico no sirve de nada, porque se trata de contratos diferentes y el servicio es del todo legal. Lo más que podrá conseguir es que la recepción de mensajes *premium* quede bloqueada por defecto. Pero el dinero no se devuelve.

6.6.2 Spyware

Existe gran número de aplicaciones Android capaces de extraer información del usuario, monitorizar el uso que este hace del dispositivo móvil o controlar sus desplazamientos geográficos mediante la recuperación de datos correspondientes a la posición de antenas de telefonía móvil, puntos de acceso wifi y coordenadas GPS. Un software fisgón también puede exportar a un destino desconocido la lista de contactos, los mensajes SMS, el correo electrónico y, si el terminal está rooteado, cualquier archivo que se encuentre almacenado en el mismo. Algunas aplicaciones pueden incluso grabar conversaciones telefónicas.

Actuación	Permiso requerido	API
Intercepción de mensajes SMS	RECEIVE_SMS	BroadcastReceiver
Lectura de mensajes SMS	READ_SMS	getContentResolver() content://sms
Grabación de audio	RECORD_AUDIO	AudioRecord. startRecording()
Lectura del registro de llamadas telefónicas	READ_CONTACTS	CallLog:calls
Extracción de coordenadas GPS	ACCESS_FINE_LOCATION ACCESS_COARSE LOCATION	LocationManager

Tabla 6.2. Permisos y API utilizadas por el software espía (Fuente: Symantec)

Dentro del sistema de seguridad Android existen numerosas funciones que pueden ser activadas si el usuario concede los permisos correspondientes. En la tabla 6.2 se puede encontrar un resumen de los principales permisos y API utilizadas por el software espía. Las prestaciones incluyen la lectura de la lista de contactos y la extracción de coordenadas geográficas.

El modelo de negocio de este tipo de software malicioso no consiste tanto en vender los datos obtenidos como en generar dinero para el desarrollador de las aplicaciones espía. Ejemplo típico de utilidad fisgona, camuflada en forma de juego, es un *malware* llamado Android.Tapsnake que se vende por varios cientos de dólares en el mercado negro.

6.6.3 Adware

Muchas plataformas publicitarias pagan a los proveedores de contenido una cantidad determinada en función de la cantidad de veces que un anuncio aparece dentro de sitios web. Una aplicación maliciosa diseñada para aprovecharse de este negocio puede mostrar el anuncio abriendo un navegador en el dispositivo móvil o haciendo aparecer la publicidad en la pantalla de modo aleatorio. Algunos desarrolladores de software malicioso se descargan del mercado Android aplicaciones de pago y mediante ingeniería inversa insertan en ellas los anuncios publicitarios del programa al cual están suscritos, para después volverlas a poner en circulación desde un *market* no oficial o a través de páginas web. Cada vez que la aplicación es utilizada en un dispositivo móvil se muestran los anuncios, generando ingresos para el atacante.

6.6.4 Pay-per-click

Las plataformas publicitarias del tipo *pay-per-click* (pago de una cantidad por clic de ratón) abonan un importe determinado cada vez que alguien pulsa sobre el enlace correspondiente a sus anuncios. Esta cantidad no suele ser muy elevada, por lo general de unos pocos céntimos, pero cuando el número de clics es alto, los céntimos se apilan hasta hacer sumas importantes. Al atacante no le basta con los clics de aquellos usuarios que marcan legítimamente el enlace publicitario. Por el contrario, sirviéndose de aplicaciones maliciosas genera tráfico falso para inflar su gratificación. El *pay-per-click* es un delito de poca monta y difícil de combatir. La persistencia de estos modelos de negocio ilícitos obedece a que, siendo tan poco productivos, el coste que supone combatirlos es superior a las pérdidas que generan.

En esta modalidad de *malware* existen sin embargo variantes dañinas que suponen oportunidades más lucrativas para el ciberdelincuente. Una aplicación maliciosa puede activar la descarga de aplicaciones, música u otros contenidos, de manera automática y sin autorización del usuario, generando de este modo ingresos para un atacante suscrito a programas de pago por volumen de descargas o número de visitas a páginas web.

6.6.5 Envenenamiento de buscadores

Algunos motores de búsqueda muestran páginas web y elaboran rankings de recomendaciones basándose en una monitorización de visitas a sitios de Internet. Mediante aplicaciones maliciosas se puede desencadenar una avalancha de visitas a determinadas páginas web con el objeto de favorecer su posicionamiento en el ranking del buscador. Con ello un atacante generaría de manera artificial ingresos

suplementarios en aquellos sistemas publicitarios de *pay-per-view* y *pay-per-click* a los cuales esté suscrito. Como ejemplo podemos mencionar al troyano Android. Adrd, activo en algunos países asiáticos, que es capaz de perturbar el funcionamiento normal del buscador chino Baidu generando tráfico artificial.

6.6.6 Troyanos bancarios

El sector bancario utiliza diversos procedimientos para autentificar transacciones por Internet. Un sistema reciente, desarrollado para móviles, consiste en el empleo de códigos de verificación no reutilizables, de modo que en caso de caer en manos extrañas después de haber efectuado la transacción, dichos códigos resulten inútiles y no supongan ningún riesgo para el titular de la cuenta. El procedimiento es más o menos el que se describe a continuación: utilizando su PC de sobremesa o portátil, el usuario establece contacto con su banco a través de la página web de la entidad y solicita acceso a su cuenta para realizar una operación. El banco le manda un código de verificación, pero no directamente al navegador, sino fuera de línea en forma de un mensaje SMS que recibe en su teléfono móvil o en su *smartphone*. Con este sistema, utilizado por entidades financieras de países como Austria, la República Checa, Alemania, Sudáfrica y España, se evitan riesgos en caso de que las credenciales de acceso se pierdan o la comunicación entre el cliente y el banco sea interceptada mediante ataques del tipo *man-in-the middle*.[8]

El problema está en que en un dispositivo móvil infectado, los SMS pueden ser retransmitidos a un atacante si este ha instalado una aplicación de software malicioso capaz de leerlos. Lo único que se necesita es que durante el proceso de instalación el usuario haya autorizado los permisos `READ_SMS` y `RECEIVE_SMS`. En un escenario de infección combinada como el descrito en el apartado 6.5.4, tanto el mTAN como las credenciales de acceso del usuario (identificador y contraseña) pueden ser fácilmente interceptadas.

Si el usuario acostumbra a realizar sus transacciones desde el *smartphone*, y este se halla infectado por una aplicación maliciosa, el problema se agrava, puesto que ya no será necesario controlar también el ordenador de sobremesa. Tanto las credenciales como el mTAN caerán en poder del atacante.

---

8 Modalidad de intrusión informática en la que un atacante intercepta las transmisiones entre dos ordenadores haciéndose pasar al mismo tiempo por el emisor y el destinatario, y retransmitiendo el mensaje a su destino legítimo después de haberse apoderado de la información.

6.7 ESPIONAJE CORPORATIVO

Hasta ahora hemos visto casos en los que un dispositivo móvil Android era víctima de ataques procedentes de la Red. ¿Qué sucede cuando la situación se invierte y es el usuario el que utiliza su terminal para llevar a cabo actos ilícitos con el objeto de entorpecer el funcionamiento de equipos informáticos o robar información perteneciente a particulares o empresas? ¿Puede hacerlo? Para demostrar que sí, en las páginas que siguen se describe un modelo de concepto cuyo propósito consiste en facilitar al investigador algunas pistas útiles a la hora de examinar elementos de evidencia y elaborar el informe correspondiente, supuesto que llegue a darse el caso. Android no solo es una pieza muy sofisticada de software susceptible de sufrir todo tipo de *exploits* y agresiones remotas. Con facilidad y una inversión reducida también puede convertirse en una potente herramienta de *hacking* para fines éticos o maliciosos.

Conviene recordar que la utilización de dispositivos electrónicos y software para interceptar tráfico de redes que no son propiedad del usuario constituye un delito. Lea los artículos 197 a 201 del Código Penal —sobre revelación de secretos y sus consecuencias—. Lo que se describe en los apartados siguientes no es una prueba de *pentesting*, sino un escenario delictivo que puede darse en la realidad y de hecho se da, con mayor frecuencia de la que nos gustaría imaginar.

6.7.1 Hardware

El sospechoso adquiere en una tienda de artículos de segunda mano un Samsung Galaxy S2. Ha pagado en efectivo sin dejar datos personales que permitan seguir pistas cruzando información con los archivos de registro de los elementos de hardware con los que el terminal pueda llegar a interaccionar, como *routers*, concentradores, puntos de acceso wifi o servidores web. Como tampoco piensa darse de alta en ninguna línea de teléfono móvil ni comprar tarjetas de prepago, no tiene que facilitar datos personales de ningún tipo. En este sentido el anonimato es total.

Este usuario piensa emplear su terminal para interceptar tráfico inalámbrico. Lo primero que hace es rootear el dispositivo por el procedimiento descrito en el apartado 3.7.7. Este es un paso necesario para que funcionen las utilidades de *hacking* que tiene previsto instalar para utilizarlas con fines delictivos. La nueva partición Recovery CWM además le permitirá realizar diversas tareas de mantenimiento como copias de seguridad, borrado de cachés, etc.

6.7.2 Herramientas de software

Provisto del software necesario, un Samsung Galaxy S2 —o cualquier otro dispositivo Android de gama media— puede convertirse en un potente analizador de tráfico inalámbrico sin necesidad de gastar dinero en hardware adicional. El hacker de sombrero negro al cual estamos siguiendo la pista tiene ideas muy claras en este sentido. También posee una gran habilidad en el manejo de utilidades de *wardriving*[9] y ataque inalámbrico. Todas estas aplicaciones se pueden descargar con Google Play del mercado oficial Android:

Figura 6.4. Wifi Analyzer

- **Wifi Analyzer**. Escáner de redes inalámbricas que proporciona información detallada sobre puntos de acceso, canales de transmisión, potencia de señal, sistemas de encriptación y otras características (Figura 6.4). Bien utilizado sirve para elegir la conexión adecuada, estudiar posibles interferencias o seleccionar canales poco congestionados. En manos de un ciberdelincuente, Wifi Analyzer permite que aquel se haga una idea del tráfico circundante y averigüe cuáles son las zonas menos protegidas de una red corporativa —por ejemplo mediante puntos que emiten en abierto o sistemas de encriptación WEP—.

9 Ver nota 1 en pág. 16.

- **Intercepter-ng**. *Sniffer* inalámbrico con versiones para Windows, Linux y Android. Permite la captura de paquetes con posibilidad de grabarlos en el terminal o en la tarjeta de memoria SD. Las *cookies* se muestran automáticamente en pantalla, por lo que no es preciso examinar el archivo de capturas con Wireshark. También muestra al vuelo los pares de identificación USER/PASSWORD de numerosos servicios como correo electrónico POP3, páginas web, foros, etc., y dispone de una utilidad para lanzar ataques contra implementaciones inseguras de SSL. Utilizar Intercepter-ng es ilegal, por lo que si se puede probar el uso de esta herramienta, por ejemplo mediante los archivos de captura almacenados en el dispositivo móvil, sería factible añadirlo a la cuenta del sospechoso ante la administración de Justicia.

- **Terminal de comandos**. Sin comentarios, salvo que no puede faltar en el arsenal de todo hacker que se precie. Existen programas muy útiles para tareas a bajo nivel con archivos y directorios, sobre todo si su instalación se complementa con la del paquete `busybox`, que instala en `/system/xbin` una versión reducida de todos los comandos típicos del entorno Linux/Unix.

- **SAMBA Server**. Esta utilidad requiere rooteado previo del dispositivo, y hace posible acceder desde un PC de sobremesa a un terminal Android como si este fuese una unidad de red más. SAMBA es útil para trasladar archivos desde máquinas que funcionan en entornos Windows, que constituyen la mayor parte del parque de estaciones de trabajo y servidores en las actuales instalaciones de red local.

6.7.3 Modus operandi

El sospechoso trabaja en un proyecto de espionaje corporativo y está interesado en adquirir información confidencial en zonas próximas a su objetivo, por ejemplo una gran empresa del sector de la defensa. Para ello se moverá por cafeterías próximas, universidades y centros de investigación, la zona wifi del aeropuerto o la estación de ferrocarril, etc. Para lograr sus fines se conecta a redes abiertas o protegidas cuya contraseña haya logrado averiguar mediante programas como Wifipass, explotando vulnerabilidades WEP, ingeniería social u otros trucos por el estilo. Luego pone en marcha su *sniffer* y comienza a absorber tráfico.

Intercepter-ng no funciona del mismo modo que una tarjeta inalámbrica como las que utilizan los aficionados al *wardriving*. La interfaz wifi de los dispositivos

Android no admite el modo Monitor,[10] y tampoco puede llevar a cabo ataques de inyección de paquetes ni otras operaciones parecidas. Para que el ataque tenga éxito es necesario estar autentificado dentro de la red cuyo tráfico se quiere interceptar. Una vez dentro, la aplicación escanea el medio en busca de dispositivos inalámbricos y lanza contra ellos un ataque de envenenamiento de caché ARP, haciéndoles creer que el dispositivo del agresor es la puerta de enlace por defecto (*gateway*) y obligándoles a enviarle a él todos sus paquetes de datos.

Figura 6.5. Intercepter-ng captando el tráfico inalámbrico circundante

Estos paquetes son analizados al vuelo y acto seguido reenviados a la verdadera puerta de enlace de la red local. Las respuestas desde Internet siguen el recorrido inverso. Una vez terminada la captura, y cuando el sospechoso se ha desconectado de la red, las cachés ARP de la puerta de enlace y los restantes terminales se reconsituyen con sus valores legítimos y queda eliminado todo rastro de esta manipulación de aparatos que, sobra decirlo, es ilícita y perseguible en cualquier sistema legal.

10 Modo de funcionamiento característico de los adaptadores inalábricos en el cual el dispositivo puede transmitir a las capas superiores de la pila de protocolos todas las tramas inalámbricas que detecta en el aire y no solo las que van dirigidas a su dirección de hardware o MAC.

Figura 6.6. Intercepter-ng capturando conexiones HTTP y, posiblemente, también las cookies de los usuarios

Figura 6.7. Intercepter-ng capturando cookies

Los paquetes capturados pueden guardarse en archivos PCAP para ser analizados en un ordenador de sobremesa o portátil mediante Wireshark. Intercepter-ng no solo captura usuarios, contraseñas, *cookies* y otros elementos de información valiosos, también los muestra en pantalla, de modo que el agresor puede comenzar su labor de intrusión de manera inmediata y sin tener que alejarse del escenario de sus capturas.

6.7.4 Implicaciones

¿A dónde nos lleva esto? Antes hemos dicho que los límites los pone la fantasía del delincuente. El agresor dispone de hardware y software que le permiten secuestrar sesiones de banca *on line*, correo web, tiendas virtuales, plataformas de pago o redes sociales como Facebook, Linkedin o Xing. Con unos cuantos pares de identificación USER/PASSWORD en intranets y servidores de correo electrónico IMAP y POP3 puede acceder a correspondencia profesional, archivos adjuntos, documentos profesionales e información reservada. Está en condiciones de utilizar estos recursos como base para consolidar su presencia dentro de la red, sembrar *exploits* o ejecutar maniobras de ingeniería social.

Cuando hablamos de temas como APT (en español: amenaza avanzada persistente) nos puede parecer que se trata de algo lejano y de alto nivel, solo accesible a las grandes potencias o a bandas criminales dotadas de medios para procurarse los equipos y el talento necesario. No nos imaginamos que pueda estar a la puerta de nuestras casas o de las empresas en las que trabajamos. Conviene que nos vayamos acostumbrando a verlo como algo cercano a nuestro entorno personal y profesional. Android y las tecnologías móviles confieren una nueva dimensión a la problemática del espionaje corporativo. A partir de aquí el investigador debe extraer sus propias conclusiones.

7

SOLUCIONES INTEGRADAS

La pregunta que el lector viene quizás planteándose desde el comienzo va a ser finalmente contestada. Y la respuesta es que sí: existen programas que lo hacen todo, o casi todo, con solo conectar el teléfono móvil al ordenador a través de un cable USB. La razón de haber empleado tantas páginas en explicar procedimientos de adquisición y análisis forense de sistemas Android, así como el funcionamiento del hardware y software subyacentes, es sobre todo pedagógica. El investigador necesita formación, no un entrenamiento superficial en el manejo de herramientas específicas para cada tarea. El objetivo de este aprendizaje teórico consiste en que el investigador llegue a saber cómo funcionan las cosas para poder tomar las decisiones que convengan sin verse limitado por unos medios y unos métodos que se aplican de modo mecánico sin saber qué es lo que hacen.

7.1 VENTAJAS DEL SOFTWARE COMERCIAL

Pese a lo anteriormente dicho, en el contexto de una investigación real las cosas pueden ser diferentes. Lo normal es que el investigador forense no esté en el lugar de los hechos, o que el encargado de la adquisición sea una persona sin formación técnica ni experiencia en el tema. De un modo u otro al final resulta inevitable el empleo de soluciones integradas, ya sean comerciales o de código libre. En el mercado existe un buen número de productos destinados a la adquisición y el análisis forense de dispositivos móviles. Su utilización, pese al coste de las licencias y los cursillos de formación, reporta ventajas indudables:

- Rapidez en la ejecución de las tareas.
- Facilidad de aprendizaje y uso de las herramientas.
- Posibilidad de adquirir gran parte de los modelos de teléfonos móviles, tabletas y otros dispositivos portátiles existentes en el mercado.
- Compatibilidad con los principales sistemas operativos móviles: Android, iOS, Windows Mobile, teléfonos chinos, Symbian y otros.
- Realización automatizada de informes.
- Detección de virus, aplicaciones maliciosas y otros parásitos.
- Mayor productividad con los presupuestos y las plantillas de personal existentes.
- Aplicación de métodos repetibles y comprobables.
- Estos productos disponen de certificaciones reconocidas internacionalmente.
- Amplia aceptación por parte de departamentos de policía y administraciones de Justicia de todo el mundo.

No es intención del autor manifestar en las páginas que siguen una preferencia especial por ninguno de estos productos. La selección es incompleta, incluso arbitraria si se quiere, y desde luego existen alternativas. Si no se citan es por varias razones. En primer lugar, no disponemos de suficiente papel. Por otro lado, si el lector ha sido capaz de llegar hasta aquí, eso quiere decir que ya posee cierta competencia en la materia y el tema le interesa, por lo cual no tendrá dificultad en localizar por su cuenta otros productos que sirvan a sus fines mejor que los que se describen en el presente capítulo. Finalmente, se ha decidido seguir un criterio de popularidad. Las soluciones técnicas de las que vamos a hablar son aceptadas por juzgados y departamentos de policía en entornos jurídicos y culturales similares al nuestro. Se trata de software europeo o israelí, utilizado por las fuerzas de seguridad, ejércitos, compañías de seguros, investigadores privados, departamentos de seguridad de las grandes empresas y otras organizaciones.

En primer lugar se hablará de Cellebrite UFED Touch, plataforma hardware transportable para la adquisición de evidencia electrónica en cualquier entorno. Posteriormente pasaremos revista a Oxygen Forensic, una suite rusa de software para PC (en sus últimas versiones también ofrece la posibilidad de instalación sobre una plataforma portátil para investigaciones sobre el terreno). Finalmente, y como término de comparación implementado exclusivamente a base de software libre, se presenta ADEL, modelo de concepto elaborado por el investigador alemán Michael Spreitzenbarth para la adquisición automatizada de bases de datos SQLite, imágenes y otros elementos de evidencia existentes en el interior de dispositivos Android.

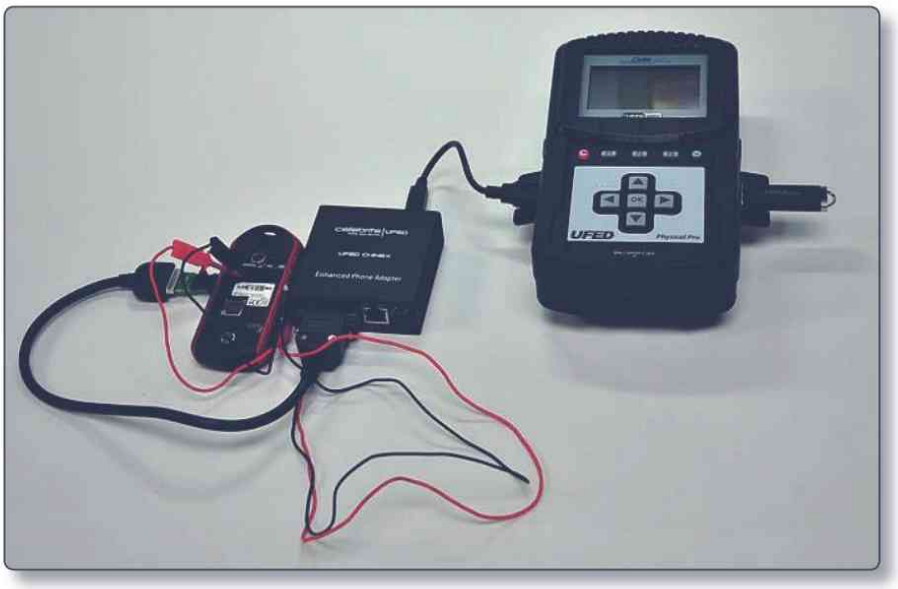

Figura 7.1. Cellebrite UFED equipado para la adquisición de teléfonos móviles chinos

7.2 CELLEBRITE UFED TOUCH

La empresa israelí Cellebrite produce y distribuye dispositivos transportables y compactos, compuestos por una solución de hardware y software integrada dentro de una plataforma con carcasa resistente (*ruggedized*), para la extracción de datos de terminales móviles. Desde hace años los aparatos del tipo UFED forman parte del equipo estándar de los departamentos de delitos tecnológicos de la policía en gran número de países. Entretanto, el UFED (Figura 7.1) se ha quedado anticuado y comienza a ser reemplazado por el más moderno y versátil UFED Touch. Estos aparatos utilizan técnicas sofisticadas para realizar su cometido: extracción de memoria física, lectura de los sistemas de archivos, rescate de imágenes y objetos multimedia, e incluso recuperación de contraseñas.

Cellebrite, con sede en Petah Tikva (Tel Aviv) y delegaciones en Estados Unidos y diversos países europeos, entre ellos España, tiene una división de tecnologías y servicios para cometidos civiles. Sus clientes suelen ser compañías telefónicas y empresas interesadas en operaciones de recuperación y trasvase de datos. Por su parte, la sección de clientes procedentes del sector público tiene en cartera a fuerzas armadas, administraciones de justicia y departamentos de policía de numerosos países.

Figura 7.2. Cellebrite UFED Touch

7.2.1 Adquisición

La versión de Cellebrite UFED Touch para entornos de acción (Figura 7.2) se transporta dentro de un maletín blindado, que además contiene accesorios útiles para cualquier contingencia, incluyendo escenarios de combate: bolsas de Faraday, batería de reserva con capacidad para cinco horas de funcionamiento, adaptador de tarjetas y un set completo de cables y conectores para la mayor parte de las marcas y modelos de terminales móviles existentes en el mercado (Figura 7.3). Cellebrite Touch también dispone de una interfaz Bluetooth.

Figura 7.3. Set de cables y conectores para Cellebrite UFED Touch

Cellebrite Touch utiliza una RAM interna para almacenar provisionalmente los datos adquiridos antes de grabarlos en el medio de destino, que puede ser un disco duro portátil, un *pendrive* u otro dispositivo móvil al cual se haya previsto realizar una transferencia de datos. Una vez finalizado el proceso de adquisición, o en caso de que aquel se vea interrumpido por cualquier causa, como por ejemplo el agotamiento de la batería, el contenido de esta memoria se borra. Así la extracción es siempre segura en términos forenses: cada nueva operación comienza sin datos de la anterior y la evidencia se conserva intacta. Tampoco hace falta utilizar bloqueadores de escritura.

Con estos dispositivos es posible realizar adquisiciones físicas y análisis forenses de más de 2.500 modelos de teléfonos móviles, *smartphones* y tabletas de diversas marcas, incluyendo teléfonos chinos. Si se trata de llevar a cabo una lectura lógica, el número de modelos aumenta hasta más de 5.000. Cellebrite Touch también puede realizar duplicados de tarjetas SIM. Por estas razones, además de los investigadores forenses de la policía y las fuerzas armadas, lo utilizan las empresas de telefonía móvil para transferir datos de un terminal antiguo a otro nuevo en los puntos de venta y atención al cliente.

7.2.2 Análisis forense

Cellebrite incluye UFED Physical Analyzer (Figura 7.4), un software de análisis provisto de herramientas y métodos para la exploración, detección y desencriptado de elementos de evidencia. Así mismo dispone de un módulo especial —UFED Phone Detective— para la detección rápida de marcas y modelos de teléfonos móviles. Facilidad de uso, sencillez del concepto, robustez de los equipos y versatilidad para el trabajo de campo apuntalan la popularidad de Cellebrite en el sector de la protección de datos y la informática forense.

He aquí algunas posibilidades de análisis junto con los elementos de evidencia que pueden recuperarse mediante el empleo de dispositivos Cellebrite. Parte de lo que se enumera ya estaba incluido en la gama de prestaciones de los antiguos UFED. Cellebrite Touch extiende la capacidad operativa de la plataforma y la adapta a un número mayor de escenarios:

- Extracción física y lógica de teléfonos y dispositivos móviles.
- Volcado de datos a discos duros y llaves USB.
- Recuperación de contactos telefónicos, historial de llamadas, mensajes SMS y de correo electrónico, fotografías, archivos multimedia, coordenadas GPS, contraseñas, etc.

- Extracción física de dispositivos móviles Apple (iPod, iTouch, iPad) y diversos modelos de *smartphone* hasta iPhone 4 y la versión 4.X de iOS.
- Clonado de tarjetas SIM.
- Detección automática de marcas y modelos.
- Estudio de elementos de evidencia a través de UFED Physical Analyzer y Phone Detective.
- Extracción de datos correspondientes a puntos de acceso inalámbricos, como antenas de telefonía móvil y redes wifi a las que se ha conectado un dispositivo Apple o Android.
- Visualización de informaciones geográficas facilitadas por el GPS señaladas sobre Google Earth o Google Maps.
- Realización automatizada de informes en diferentes formatos: PDF, HTML, XMS o XML.
- Cálculo de *hashes* MD5 y SHA256.
- Posibilidad de incluir logotipos y emblemas en los informes.

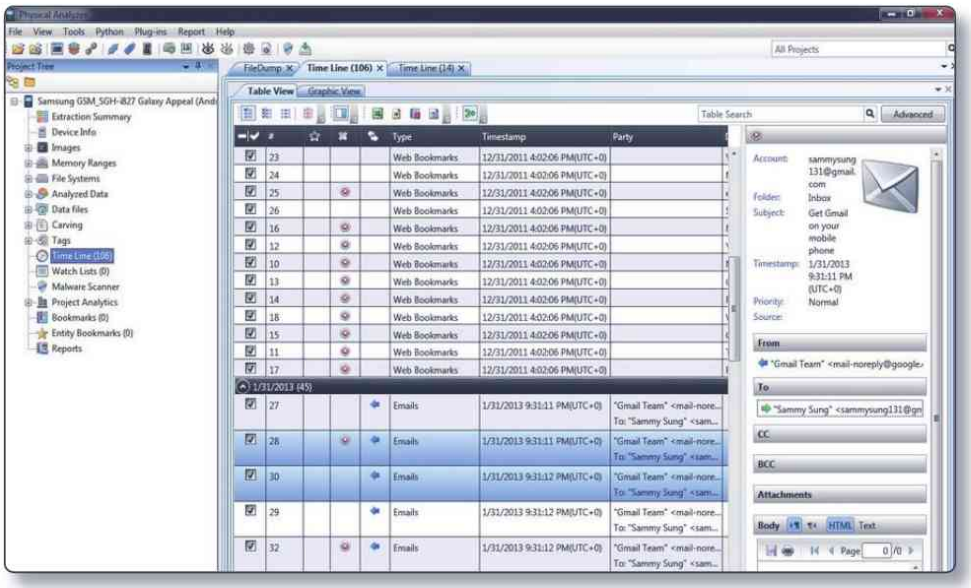

Figura 7.4. UFED Physical Analyzer

7.2.3 Cellebrite Touch y Android

Cellebrite Touch permite adquirir físicamente más de 200 modelos de dispositivos Android, en condiciones de higiene forense y sin que sea necesario

rootear el terminal. Entre los *smartphones* compatibles de última generación que pueden ser adquiridos a través de Cellebrite Touch y analizados por el software UFED Physical Analyzer se encuentran los siguientes:

- Samsung: Galaxy S2, Galaxy S3, Galaxy Note, Galaxy Note II y otros.
- HTC: Evo, Incredible, Desire.
- LG: Optimus, Optimus One, Optimus 3D, Optimus Black, etc.
- Motorola: Milestone, Milestone 2, Droid, Droid 2, Droid X, y otros.

También puede realizar adquisiciones lógicas de elementos de evidencia procedentes de Facebook, Google+, Twitter, Skype, Yahoo Messenger, Dropbox, ICQ, Evernote, WhatsApp y otras muchas. Según asegura el desarrollador en la página web de la empresa, Cellebrite Touch es capaz de deshabilitar bloqueos en algunos modelos Samsung.

7.3 OXYGEN FORENSIC SUITE

Oxygen Forensic Suite es un software ruso de investigación forense que combina el uso de protocolos propietarios de alto nivel y API de los sistemas operativos móviles para extraer datos en una extensión y profundidad más amplias de lo que resultaría posible por medio de herramientas convencionales basadas en protocolos estándar. Utilizado por departamentos de policía, fuerzas armadas, autoridades arancelarias y empresas de seguridad de todo el mundo, su objetivo consiste en adquirir aquellos datos descriptivos de la actividad del usuario que quedan registrados localmente en un dispositivo móvil: contactos telefónicos, listas de tareas, navegadores de Internet, correo electrónico, redes sociales, etc.

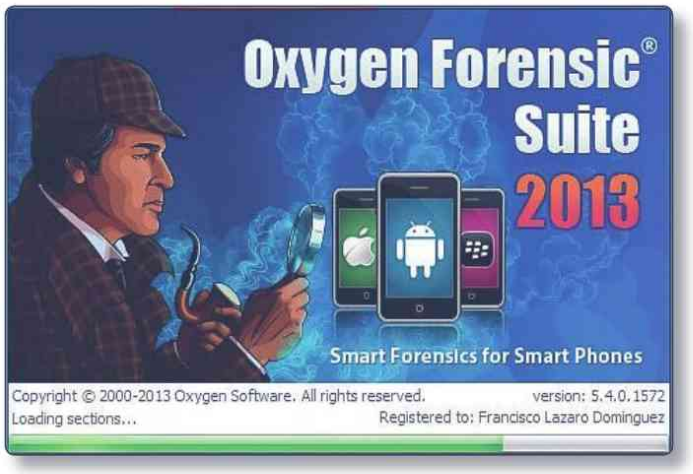

Figura 7.5. Pantalla de presentación del producto

Oxygen Forensic Software es una empresa con sede oficial en Alexandria (Virginia, EEUU). Su suite de investigación forense reconoce más de 6.000 dispositivos móviles de todas las marcas y los principales sistemas operativos, incluyendo Android. En su actual versión, Suite 2014, soporta conexiones por cable USB, Bluetooth e infrarrojos. Oxygen Forensic funciona sobre plataformas de 32 y 64 bits con sistemas operativos Windows XP, Vista, 7 y Server 2003.

Figura 7.6. Oxygen Forensic Extractor en acción

7.3.1 Instalación y manejo

Oxygen Forensic Suite 2013 se instala igual que cualquier otra aplicación estándar para Windows. La versión básica del producto se entrega con una licencia activable a través de dos sistemas: Internet o mochila (*dongle*) insertada en uno de los conectores USB del ordenador. La adquisición forense de dispositivos móviles se lleva a cabo con la ayuda de asistentes. Todo el proceso de gestión de casos, trasvase de datos, análisis y realización de informes tiene lugar a través de interfaces gráficas. Existe una versión del software traducido al español que se distribuye comercialmente a través de la empresa de seguridad Informática64.

Para trabajar con un teléfono móvil, en primer lugar es preciso asegurarse de que el modelo está soportado por el software y el ordenador dispone de los controladores necesarios para la conexión por cable o por vía inalámbrica. Algunos dispositivos requieren procedimientos de conexión específicos. En cualquier caso, debe leerse la documentación de Oxygen Forensic Suite antes de trabajar con *smartphones* de marcas extrañas o procedencia desconocida.

El asistente dirige todo el proceso, incluyendo la apertura de una ficha de gestión del caso con campos para introducir los descriptores de costumbre: nombre del investigador, descripción y número de caso, algoritmo de *hash* aplicado, observaciones y notas. A continuación se especifica el objetivo —la lista de elementos recuperables depende de la marca, el modelo y el sistema operativo del teléfono móvil— y se inicia el proceso pulsando sobre el botón **Extraer**.

Una vez completado el proceso, el investigador podrá elegir entre un análisis detallado de los elementos adquiridos o la elaboración de un informe sobre el dispositivo en PDF para sacar por impresora o utilizar en gestiones dirigidas a presentación de pruebas, cumplimentación de trámites judiciales o administrativos o de cualquier otro tipo.

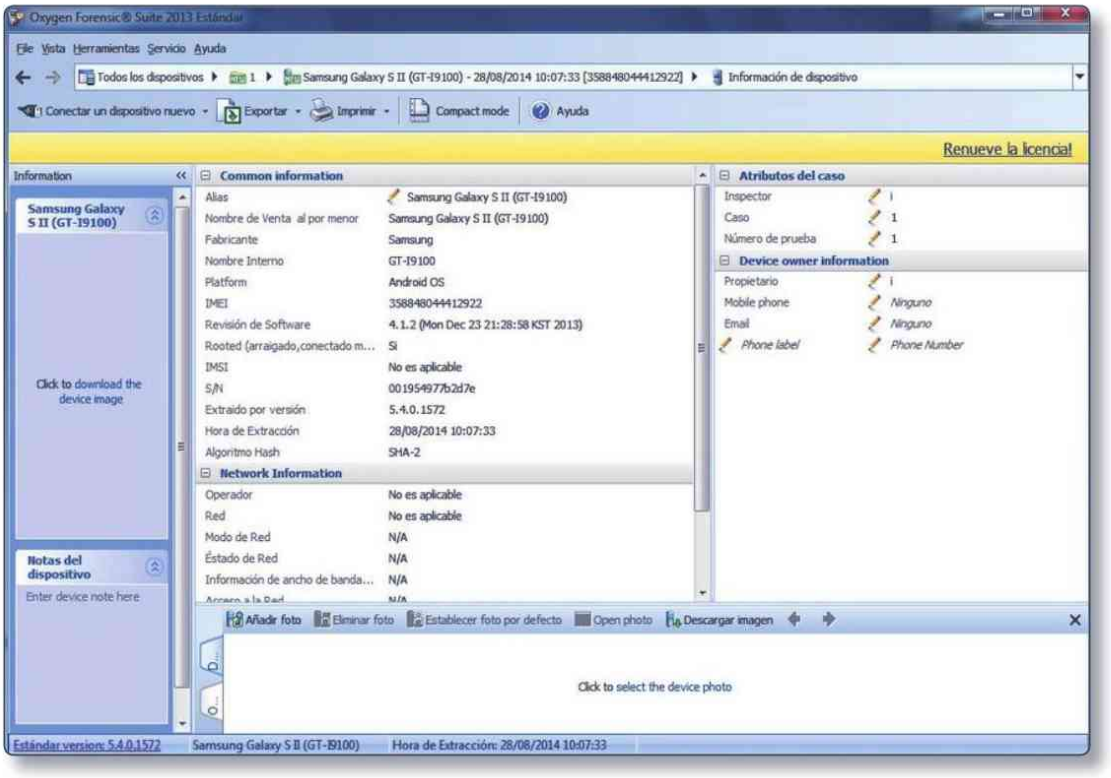

Figura 7.7. Detalles del dispositivo adquirido

7.3.2 Elementos recuperables

Oxygen Forensic Suite 2013 puede adquirir gran variedad de elementos de evidencia relacionados con la actividad del usuario, como por ejemplo:

- Información del teléfono y la tarjeta SIM (en terminales GSM).
- Lista de contactos (números de teléfono, direcciones postales, correos electrónicos, fotografías de contacto, etc.).
- Llamadas entrantes, salientes y fallidas.
- Listas de tareas, calendarios, notas, aniversarios, agenda.
- Mensajes SMS.
- Mensajes telefónicos multimedia MMS.
- Correo electrónico y archivos adjuntos.
- *Logs* de conexiones wifi, GPRS con datos relativos a puntos de acceso inalámbricos y volumen de tráfico.
- Fotografías y vídeos tomados con la cámara del teléfono.
- Archivos multimedia y documentos almacenados por el usuario en la memoria del teléfono o en la tarjeta de expansión externa.
- Información geográfica y coordenadas GPS.
- Enlaces archivados e historial de navegación de Internet.
- *Backups* de iPhone protegidos por contraseña.
- Datos de usuario de las aplicaciones.

7.3.3 Conexión de dispositivos Android

La adquisición y el análisis forense de dispositivos móviles Android a través de Oxygen Forensic no es tan simple como conectar el teléfono al ordenador y pulsar un botón. El proceso requiere la instalación previa de un programa local —OxyAgent— en la memoria del teléfono. Esta operación resulta mínimamente intrusiva, puesto que es provisional, funciona con pocos recursos y no altera los datos del usuario.

Para establecer la conexión de un dispositivo Android en Oxygen Forensic Suite 2013 se necesita por lo general lo siguiente:

- Una instalación de Oxygen Forensic Suite 2013 plenamente funcional y con licencia activada.
- Cable USB original del dispositivo.
- Controladores del fabricante.
- Una tarjeta de memoria externa soportada por el terminal (necesaria para almacenar los datos temporales de OxyAgent durante la extracción de datos).

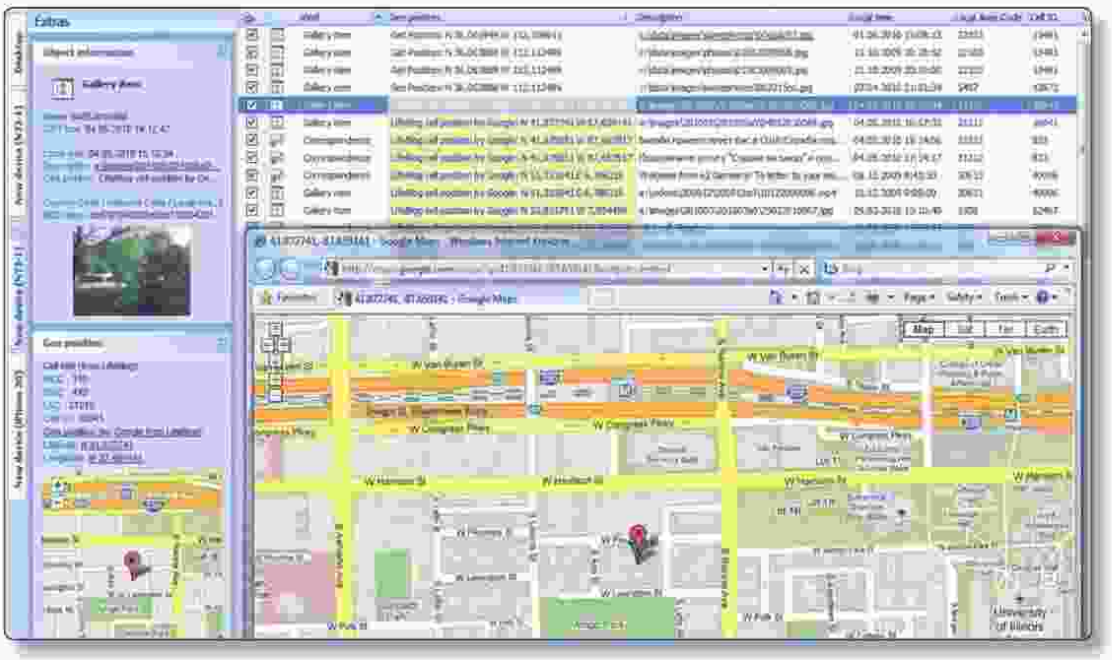

Figura 7.8. Procesamiento de información geográfica con Oxygen Forensic Suite

7.4 ADEL

ADEL (Android Data Extractor Lite) es una herramienta desarrollada por Michael Spreitzenbarth, experto en forénsica de sistemas Android y profesor de la Facultad de Informática de la Friedrich-Alexander Universität Erlangen-Nüremberg. El propósito de ADEL consiste en llevar a cabo adquisiciones automatizadas de las principales bases de datos SQLite de dispositivos móviles Android, extrayendo con posterioridad el contenido de los archivos copiados. ADEL también puede utilizarse para el análisis de bases de datos SQLite adquiridas manualmente. Esta opción fue incluida para dar soporte a aquellos *smartphones* que no admiten el empleo directo de ADEL debido a la existencia de características de seguridad como particiones de arranque bloqueadas.

Las descripciones funcionales y los diagramas que siguen proceden de documentos elaborados y publicados por el desarrollador en la página web de la Friedrich-Alexander Universität y en su blog sobre investigación forense de sistemas Android: *http://forensics.spreitzenbarth.de/*. No quiero continuar sin expresarle mi agradecimiento por haber autorizado su publicación en este libro. También recomiendo encarecidamente la lectura de su tesis sobre investigación forense de dispositivos Android, con el título *Dissecting the droid*; documento imprescindible para todo aquel que se dedique a la seguridad móvil y publicado por el autor en Internet.

7.4.1 Principios de diseño

Inicialmente ADEL fue desarrollada para las versiones 2.x de Android, pero a través de una serie de actualizaciones ahora también es aplicable a dispositivos móviles equipados con versiones 4.x. Durante la creación de esta herramienta se tuvieron en cuenta tres objetivos de carácter general:

- **Cumplimiento de directrices forenses**. La suite de código libre ADEL está pensada para un tratamiento correcto de los datos obtenidos en el transcurso de una adquisición forense. Este propósito se cumple mediante una ejecución de las operaciones de análisis a partir de copias adquiridas y guardadas en la estación de trabajo forense, y no sobre la evidencia original. Este método de trabajo evita que los datos se vean alterados accidentalmente por el usuario de ADEL. Previamente y con posterioridad a cada una de las tareas de análisis, se extraen *hashes* para demostrar que los datos no han sido modificados durante la operación.
- **Modularidad y carácter ampliable**. ADEL está integrada por dos módulos independientes, uno de análisis y otro para la realización de informes. Entre ambos módulos existe una interfaz con puntos de conexión definidos de manera estricta, que en caso necesario pueden complementarse mediante funciones adicionales. La estructura modular del software hace posible una actualización permanente con el objeto de incorporar nuevos archivos de bases de datos que resulten útiles en futuras adquisiciones forenses. De este modo resulta más fácil adaptar el sistema a necesidades derivadas de la aparición de nuevos dispositivos en el mercado.
- **Usabilidad**. El propósito del desarrollador consistía en hacer el uso de ADEL tan fácil como fuera posible, tanto para personal cualificado como para usuarios sin experiencia. El análisis de los dispositivos móviles se lleva a cabo de forma automática, sin que el usuario reciba notificaciones relativas a procesos internos. El módulo de informes genera un documento legible que incluye un anexo con la información decodificada durante el proceso de análisis. ADEL genera un archivo de registro completo con una lista de todas las operaciones realizadas con el objeto de asegurar la trazabilidad de las mismas.

7.4.2 Implementación y funcionamiento

En la figura 7.9 se muestra un diagrama con la estructura y los principios de funcionamiento de ADEL. ADEL se vale de Android SDK para realizar volcados de bases de datos a la estación de trabajo del investigador forense. Al extraer el contenido de las bases de datos SQLite, ADEL examina estructuras de bajo nivel en busca de información existente en los campos de los encabezamientos de los que están compuestos los archivos SQLite. Todo este proceso se lleva a cabo en modo de solo lectura. Tras haber descifrado las cabeceras del archivo, ADEL interpreta el árbol con estructura en *b* que contiene la tabla `'sqlite-master'` como raíz principal. A continuación es procesada toda la estructura en árbol *b* para cada una de las tablas, y el contenido correspondiente a sus filas extraído para servir como elemento de evidencia.

Las bases de datos SQLite procesadas por ADEL son las correspondientes a los elementos de evidencia que se detallan a continuación:

- Información de la tarjeta SIM (código IMSI y número de serie).
- Contactos telefónicos y lista de llamadas.
- Anotaciones en el calendario.
- Mensajes SMS.
- Entradas en Twitter y Facebook.
- Coordenadas GPS procedentes de diversas aplicaciones del terminal.

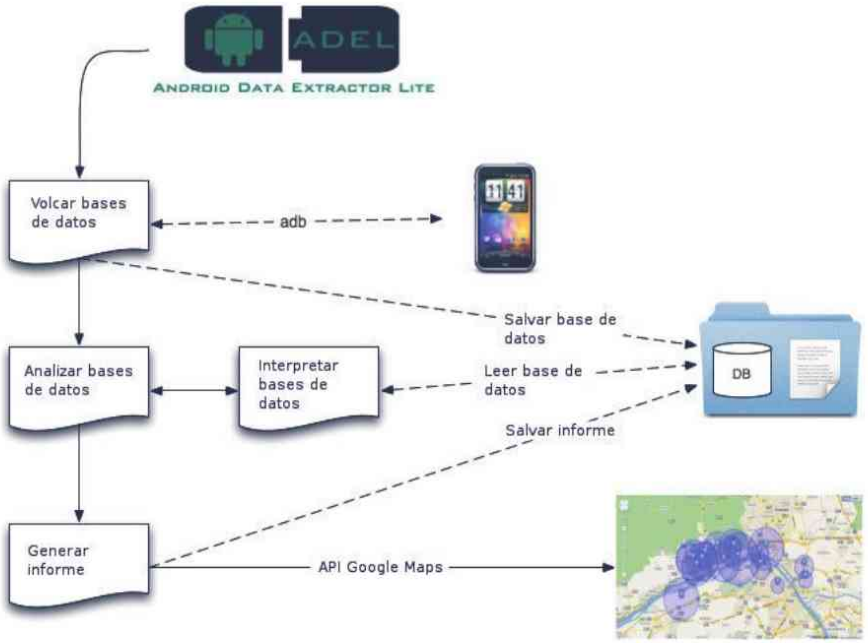

Figura 7.9. Principio de funcionamiento de ADEL

Los datos recuperados a través de los procedimientos que se describen con anterioridad son introducidos en un archivo XML para facilitar el empleo posterior de la información en documentos electrónicos relacionados con el caso. El módulo de análisis es configurable en función de los objetivos del análisis y los nuevos requerimientos resultantes de los cambios que puedan introducirse en nuevas versiones del sistema. El módulo de informes genera archivos XML para cada categoría de datos. Todos los archivos producidos por ADEL se guardan en sus respectivas subcarpetas dentro del directorio correpondiente al caso en curso.

7.4.3 Otras características

Para tener acceso a las bases de datos y los directorios correspondientes del dispositivo móvil ADEL se precisan privilegios de superusuario. Esto se consigue a través de procedimientos de *rooting* como los explicados en el capítulo 4. Podemos rootear el terminal de manera temporal o permanente, o bien instalar una partición Recovery. Este último método resulta más práctico en la etapa preparatoria de una intervención con ADEL, ya que no es necesario preocuparse por posibles bloqueos o de si está activada la depuración USB.

En la actualidad, ADEL utiliza una partición Recovery adaptada a partir de ClockWorkMod. Esto evita tener que hacer modificaciones en el *kernel* o en el *daemon* ADB con el objetivo de adquirir privilegios de superusuario. Es más, en algunos dispositivos móviles de reciente fabricación existe la posibilidad de grabar la partición Recovery en la RAM del dispositivo móvil mediante una opción de la utilidad `fastboot`, lo cual proporciona acceso al terminal sin que este sufra ninguna alteración permanente.

La versión *open source* de ADEL se encuentra disponible en *https://github.com/mspreitz/ADEL*.

*** *** *** *** ***

8

CONCLUSIÓN

Android no es el único sistema operativo para *smartphones* y tabletas. Hasta hace poco tampoco era el más utilizado. En la actualidad, los dispositivos Apple iOS —iPhone 5, iPad de nueva generación e iPad Mini, sumados a los numerosos iPhone 4 y 4S y otros de generaciones anteriores— suponen una proporción bastante respetable del parque de terminales móviles. Se estima que hasta comienzos del año 2013 se habían vendido cerca de 500 millones de unidades de todos los modelos desde que el primer iPhone salió al mercado en 2007. Por consiguiente existen buenos motivos para prestarles atención. ¿Cuáles son las diferencias entre las plataformas móviles Android y Apple iOS? ¿Se pueden emplear los mismos métodos de adquisición y análisis? A estas cuestiones y otras es a lo que se intenta dar respuesta en las páginas que siguen.

En este último capítulo también se plantean consideraciones relativas al lado práctico del oficio, así como a estrategia y procedimientos de trabajo. ¿Qué necesita el investigador forense para alcanzar su objetivo y hacer una buena presentación de elementos de evidencia? ¿De qué materiales, herramientas, accesorios, documentación y otros elementos auxiliares debe disponer en su laboratorio? La labor del informático forense implica el manejo de unas tecnologías muy potentes para el proceso de datos electrónicos, pero que, pese a su poder, son incapaces de llegar más allá del objeto específico que estudian —los propios dispositivos móviles—. ¿Cómo dar el salto desde el teclado de un ordenador hasta los dedos del sospechoso culpable, consiguiendo así lo que realmente interesa a fiscales, jueces y letrados: disponer de elementos que permitan demostrar una atribución de autoría de actos delictivos con nombres y apellidos?

Finalmente se pasará revista a algunas implicaciones jurídicas del trabajo de investigación forense de dispositivos móviles. Este último capítulo no ha de ser

considerado como el típico vagón de cola, con una lista de conclusiones y consejos. No por quedar para lo último tiene menos importancia. Leer con atención los últimos apartados de esta obra puede resultar más útil que cualquier otra cosa explicada hasta el momento.

8.1 ANÁLISIS DE DISPOSITIVOS APPLE IOS

Todos los dispositivos móviles plantean dificultades de tipo similar. Las agendas digitales y los teléfonos móviles anteriores a la aparición del *smartphone* tal y como hoy lo conocemos (un terminal de diseño atractivo con pantalla táctil y sistema operativo iOS o Android) no eran diferentes en este sentido. Con aquellos dispositivos que hoy se nos antojan antediluvianos, pese a haber pasado tan solo unos pocos años, también resultaba difícil extraer elementos de evidencia sin producir cambios en la configuración del terminal. Análogamente, existía la necesidad de aislar los dispositivos de la red para evitar manipulaciones remotas. Era preciso conectarlos a una fuente de alimentación para evitar que se apagaran al agotarse la batería y había que tomar otras precauciones por el estilo que nunca han sido necesarias a la hora de analizar un PC de sobremesa con discos duros y soportes de datos convencionales. Así mismo, para realizar adquisiciones forenses, la mayor parte de los dispositivos móviles, ya sean *smartphones*, tabletas, teléfonos móviles, agendas digitales u otros aparatos, requiere un SDK o kit de desarrollo de software específico del fabricante y, por supuesto, cables. Aunque el investigador disponga de un laboratorio bien equipado, en el momento de la intervención debe asegurarse de que todos los conectores y accesorios específicos del dispositivo móvil han sido recogidos en la escena del crimen junto con el resto de las pruebas.

8.1.1 Similitudes y diferencias

Los *smartphones* de Apple fueron los primeros en aparecer en el mercado, y con ello establecieron las pautas del actual ecosistema móvil, con sus mercados de aplicaciones, su esquema de división de memoria NAND en particiones de sistema y de usuario y un modelo de seguridad basado en la restricción de privilegios del usuario. En cuanto a los aspectos funcionales, Android es muy parecido a iOS: misma filosofía de uso —interfaces gráficas y pantalla táctil—, la misma gama de prestaciones —telefonía móvil, correo electrónico, Internet, multimedia, *apps*—, parecido hardware —microprocesadores y memorias NAND— e incluso las mismas ideas de diseño, que con frecuencia dan lugar a conflictos de patentes entre las empresas.

En lo que se refiere a capacidades de servicio al consumidor, se puede decir que la disyuntiva Apple/Android carece de sentido. Ambas cosas son lo mismo o al menos algo parecido. La única diferencia importante, más perceptible para el usuario de a pie que para los analistas de las publicaciones técnicas, se encuentra en el el precio.

Tanto Apple como Google, en última instancia, han desarrollado su software para dispositivos móviles a partir de un mismo sistema operativo (Unix). Aunque difieran los sistemas de archivos, la disposición estratificada de componentes —*kernel*, controladores, librerías, entorno de aplicaciones—, la organización de los directorios y el desarrollo de aplicaciones, en ambos casos se utilizan las mismas tecnologías básicas de proceso de información: bases de datos SQLite y XML. ¿Dónde están entonces las diferencias, si es que las hay?

Ciertamente las hay. Android y Apple iOS poseen características específicas que obligan a modificar la estrategia y los métodos de una investigación forense. El propósito de las páginas que siguen no consiste en ofrecer procedimientos de adquisición forense adaptados a la hoja de datos de un dispositivo Apple con sistema operativo iOS —iPhone, iPad, iTouch o las generaciones sucesivas del iPod—. El objetivo consiste en aportar claridad acerca de aspectos que pueden confundir al investigador y que, en caso de tener que llevar a cabo una investigación sobre dispositivos iOS, exigen una aproximación diferente a la que emplearíamos con Android a la hora de realizar adquisiciones forenses y análisis de elementos de evidencia.

8.1.2 Carácter exclusivo

Lo primero que llama la atención en un iPhone o un iPad —y que es extensivo a cualquier producto de marca Apple— es el carácter exclusivo del género. Tal vez un terminal Apple, visto por dentro, no sea tan diferente de cualquier aparato de la competencia. Visto desde fuera sí lo parece, y además de un modo que va más allá del propósito de causar una impresión óptica peculiar. Prueba de ello la tenemos en los tornillos con ranura en forma de estrella de cinco puntas que dificultan el acceso al interior del dispositivo, y que requien de una herramienta especial para trabajar con ellos. También en la agresividad con que la empresa defiende ante los tribunales el diseño de sus dispositivos y sus conceptos de usabilidad.

En el mundo Apple, la diferencia tiene un carácter fundamental, casi de naturaleza filosófica, y comienza con el planteamiento estratégico desde el que se lanzó al mercado el primer iPhone en el año 2007: el desarrollo de modelos de negocio innovadores que revolucionan los esquemas de comercialización estableciendo nuevas pautas de venta y de consumo.

8.1.3 iTunes y AppStore

A diferencia del ecosistema pluralista y anárquico de Google, donde el denominador común lo constituyen únicamente Android y las recomendaciones de la Open Handset Alliance, y donde todo el mundo puede hacer con sus terminales lo que le da la gana, en Apple toda la actividad de distribución de sistemas operativos, aplicaciones y contenidos tiene lugar a través de un software de sincronización exclusivo —iTunes (Figura 8.1)— y un único mercado *on line* controlado por la empresa. Las actualizaciones del sistema operativo y de software, incluyendo nuevas versiones de las aplicaciones instaladas, se llevan a cabo a través de iTunes. Apple recomienda actualizar iTunes a la última versión en cuanto esta se encuentre disponible, y, una vez instalada, no permite volver a poner en el iPhone o en el iPad una versión de de iOS anterior a la que en ese momento está siendo distribuida. En cuanto al hardware, solo existen los modelos del momento (iPhone 4, 4S, 5, etc.) con un número de variantes reducido (8, 16, 32 o 64 GB).

Figura 8.1. iTunes

En el mundo Apple no existe la misma tolerancia de la que Google hace gala con respecto a la liberación de terminales o las modificaciones personalizadas del sistema operativo. En Apple iOS no hay *rooting*, ni ROM cocinadas, ni mercados no oficiales. Los desarrolladores que no cumplen las directrices de Apple ni siquiera llegan a entrar en el sistema. Aunque desde hace años las sentencias del Tribunal Supremo de Estados Unidos reconocen el derecho de los usuarios a aplicar a sus terminales técnicas de *jailbreaking* si lo desean, con el objeto de ampliar su funcionalidad y

permitir la instalación de software externo al App Store, todo intento de intrusión en las interioridades del iPhone mediante técnicas de *hacking* o ingeniería inversa es visto con hostilidad por Apple. No hace falta decir que la garantía del terminal queda anulada.

El objetivo de la política de Apple no solo consiste en proteger los beneficios y la posición de mercado de la empresa, también busca garantizar la seguridad de las redes informáticas sobre las que funcionan su infraestructura de comunicaciones y su sistema de distribución de productos y servicios comerciales. El *jailbreaking* abre en el modelo de seguridad de Apple una brecha insalvable. Debido a la falta de información relativa a los escenarios de privilegios ampliados, así como al desentendimiento del fabricante en caso de que el usuario decida saltarse sus recomendaciones de uso, las consecuencias del *jailbreaking* pueden resultar más graves para la seguridad de un dispositivo Apple que las del *rooting* en un terminal Android.

8.1.4 Jailbreaking

En foros de Internet es habitual que se hable del *jailbreaking* como si fuera el equivalente en iOS del rooteado de dispositivos Android. Nada nos impide verlo de ese modo, aunque existen diferencias. Gracias al *jailbreaking*, que se puede lograr por numerosos métodos en función del modelo y de la versión de iOS, el usuario consigue ampliar las prestaciones de su terminal mediante el acceso a plataformas de aplicaciones independientes y externas al App Store de Apple. El *jailbreaking* no es lo mismo que la liberación del teléfono móvil con el objeto de utilizar este en las redes de un operador diferente al que nos lo vendió. Para esto se emplean otros procedimientos.

Figura 8.2. Cydia Store en un iPhone liberado mediante jailbreaking

En el *jailbreaking* principalmente se trata de desbloquear el dispositivo móvil para poder instalar en él software que permita la descarga de aplicaciones que extienden la funcionalidad del terminal, como por ejemplo consolas de comandos, utilidades tipo Linux como `dd` y `netcat` y otras herramientas de *hacking*. Estas aplicaciones no están disponibles en el mercado oficial de Apple y han de conseguirse a través de una tienda virtual denominada Cydia (Figura 8.2).

En ocasiones el investigador forense puede verse obligado a efectuar prácticas de *jailbreaking* sobre un terminal, por ejemplo cuando tenga que realizar una adquisición física de la partición de datos de usuario. Entonces, al igual que cualquier usuario avanzado del iPhone, se encontrará con los obstáculos que Apple opone para dificultar el *jailbreaking*, como por ejemplo la imposibilidad de realizar un *downgrading*[11] del sistema operativo. Esta medida es muy eficaz, o al menos obliga a trabajar más a los hackers, ya que es necesario cambiar una versión actualizada de iOS para la que aún no existen procedimientos autorizados de *jailbreaking* por otra anterior que sí dispone de ellos. Entonces habrá que recurrir a estratagemas rebuscadas: búsqueda de imágenes de *firmware* descontinuadas, instalación de versiones antiguas de iTunes que no sean capaces de detectar que la versión de iOS ha sido actualizada, o manipulación del archivo `hosts` del ordenador para que los intentos de comprobar la validez del *firmware* sean redireccionados a páginas de Internet especialmente diseñadas para dar el OK.

Antes hemos mencionado que el *jailbreaking* es incompatible con la seguridad de las plataformas iOS. El único sistema de protección del que Apple dispone es el canal blindado compuesto por iTunes y Apple Store, que hace posible la existencia de un monopolio de aplicaciones con niveles de calidad garantizados. Una vez levantada la barrera, nuestro iPhone o iPad no dispondrá de un sistema de permisos eficaz para las aplicaciones, ni de una máquina virtual Dalvik que aísle su ejecución. Como resultado de ello el terminal queda expuesto a ataques, robos de datos e intrusiones de todo tipo.

8.1.5 Criptografía hardware

Apple iPhone/iPad incorpora en sus modelos más recientes un sistema de encriptación de datos basado en hardware. Todos los dispositivos disponen de un motor criptográfico AES 256 alojado en un chip interpuesto entre la memoria NAND

11 Se denomina *downgrading* a la práctica de instalar un sistema operativo o un software de una versión anterior a la que en esos momentos está siendo distribuida, mantenida o recomendada por la empresa. La razón para ello, en la mayor parte de los casos, es el deseo de aprovechar un *hack* que ha sido corregido en las versiones más recientes.

y la RAM. Este componente realiza operaciones de cifrado en tiempo real sobre todos los datos que se graban en la memoria de estado sólido del dispositivo o salen de ella para ser utilizados por las aplicaciones de usuario. Este sistema, que fue introducido en el iPhone 3GS y ahora viene incorporado de serie en el iPhone 4 y modelos superiores, se apoya en dos claves AES de 256 bits grabadas de manera indeleble en el hardware durante la fabricación del chip. En las operaciones de cifrado y descifrado se utilizan dos claves: un identificador único del dispositivo (UID) y otro identificador único de grupo para todos los dispositivos pertenecientes a una misma clase (GID).

Ninguna aplicación puede acceder directamente a estas claves: desde el exterior lo único que se ve son los resultados de las operaciones criptográficas realizadas por el chip. El UID es exclusivo de cada dispositivo y no lo conoce nadie, ni siquiera Apple. Su función consiste en crear un vínculo criptográfico único con el terminal que haga imposible el descifrado de las claves que se derivan del mismo y son utilizadas para proteger los archivos del usuario. Estos códigos están grabados en el hardware de modo indeleble y en teoría ni siquiera es posible acceder a ellos extrayendo físicamente la memoria o mediante procedimientos de lectura JTAG o *chip-off* (véase capítulo 4). El GID es común a los dispositivos de una clase determinada —con la misma CPU— y proporciona un grado de seguridad adicional en la instalación de versiones nuevas de iOS, así como durante las actualizaciones de software.

En los dispositivos Apple iOS el cifrado hardware no solo mejora la seguridad de los datos, también dificulta la labor del investigador forense. Una vez reinicializado el terminal con los ajustes de fábrica es casi imposible recuperar los datos. Las técnicas de *data carving* no podrán detectar en el soporte de almacenamiento firmas características ni información aprovechable de ningún tipo. Las adquisiciones físicas al estilo tradicional, con `dd` y una conexión de puertos TCP por vía inalámbrica mediante `netcat`, son posibles únicamente con modelos anteriores al iPhone 3GS.

En la práctica nada impide realizar adquisiciones a bajo nivel en un iPhone 4 o 5 —previo *jailbreaking* del dispositivo e instalación de las herramientas Cydia correspondientes—. Las imágenes a bajo nivel que podamos obtener a través de técnicas convencionales de adquisición forense, además de los datos recientes del usuario, incluirán los espacios no asignados por el sistema de archivos. Pero de ellos no se podrá sacar gran cosa porque la información está encriptada.

8.1.6 Particiones y sistemas de archivos Apple HFS+

Un dispositivo móvil iPhone o iPad dispone de dos particiones principales: una de sistema, donde van alojados el *firmware* y el sistema operativo del terminal,

y otra para los datos de usuario. En este aspecto no se diferencia notablemente de Android, salvo por el hecho de que las particiones se hallan formateadas con el sistema de archivos HFS+ característico de los ordenadores Apple. Esto supone una dificultad adicional a la hora de examinar los contenidos. Es cierto que algunas utilidades forenses de código libre, como The Sleuth Kit, incluyen soporte para el análisis de sistemas de archivos HFS+. Sin embargo, y salvo que se utilicen herramientas de *data carving* para extraer los archivos de las imágenes, el único modo de llegar hasta ellos será montando las particiones en un ordenador Apple OSX o a través de EnCase.

Obviamente, también se pueden utilizar herramientas comerciales como Oxygen Forensics o Cellebrite Touch. Aun así, en la mayor parte de los casos esto solo permitiría realizar adquisiciones lógicas. Recuérdese lo que se ha dicho antes acerca del cifrado hardware.

8.1.7 Precaución con iTunes

Un procedimiento habitual para el análisis forense de dispositivos Apple consiste en la adquisición de *backups* a través del software de sincronización iTunes. A menudo esto resulta suficiente para obtener elementos de evidencia aprovechables ante un tribunal. Sin embargo, existe un riesgo: el manejo inadecuado de iTunes puede traer consigo la pérdida de los datos del terminal. Este suele ser el caso cuando la configuración del software no es la adecuada o se activa por error un proceso de reinicialización del dispositivo. ¿Quién se lee todos esos mensajes de advertencia que iTunes muestra en pantalla a la hora de sincronizar contenidos o inicializar un iPhone recién comprado? Usted, como investigador forense competente, no debería saltárselos. Si esto le sucede durante el análisis forense de un iPhone 4 u otro dispositivo que tenga cifrado hardware, entonces tendrá un serio problema: los elementos de evidencia se habrán perdido y será imposible recuperarlos.

Tales incidencias son poco frecuentes en Android. Aunque para facilitar el trabajo del usuario los fabricantes desarrollan sus propios programas de sincronización —por ejemplo Samsung Kies—, en todo momento existe la posibilidad de establecer una conexión directa entre el dispositivo y la estación de trabajo. No se necesitan programas de sincronización para extraer datos del terminal. El investigador controla la operación y no depende de un software intermediario cuyas opciones y ajustes de configuración le puedan jugar malas pasadas. No subestime los riesgos de iTunes. El peligro de perder elementos de evidencia debido a un manejo amateur es mayor de lo que cree. Para combatir a Murphy con eficacia será preciso ensayar con dispositivos de prueba hasta haberse familiarizado con Apple iTunes. Si el caso es importante, los elementos de evidencia deben adquirirse a través de herramientas comerciales certificadas.

8.2 EQUIPAMIENTO Y MATERIALES

Lo que se ha explicado en capítulos anteriores sobre adquisición y análisis de dispositivos Android se puede repetir sobre nuestro escritorio con materiales estándar sin necesidad de realizar gastos adicionales: basta un PC de sobremesa o portátil, un par de *smartphones* de marca Samsung Galaxy, HTC o cualquier otra, el cable USB de sincronización del terminal y algunas utilidades de software descargadas de Internet. Más adelante, sin embargo, el investigador puede llegar a necesitar un pequeño laboratorio equipado con algo de material. Para ello se requiere una inversión que no tiene por qué ser cuantiosa.

Por unos cientos de euros se puede conseguir una suite comercial para la adquisición de teléfonos móviles y *smartphones*. Si desea un Cellebrite UFED Touch, el presupuesto sube de modo apreciable. Y si el objetivo es llevar a cabo operaciones más sofisticadas, como desensamblado de componentes electrónicos y extracciones físicas de chips de memoria, el coste de salas limpias y hardware especial de lectura de chips ascendería ya a importes que solo están al alcance de la Administración pública o de empresas especializadas en seguridad.

Algunos de los elementos que se mencionan en las páginas siguientes no son imprescindibles. También habrá otros que no figuran en la lista y que pueden llegar a hacer falta: depende de los requerimientos específicos del trabajo, las características del caso, el entorno legal y otros factores.

8.2.1 Hardware

PC de sobremesa o portátil. Android es un sistema operativo compatible con cualquier plataforma. Para seguir las explicaciones y ejemplos de los capítulos anteriores lo único que se necesita es un PC de sobremesa o portátil con Windows Vista/7 o una distribución de Linux. Se aconseja que tanto la máquina como el sistema operativo no sean demasiado antiguos. Conectores y *drivers* USB han de funcionar a la perfección. Es mejor utilizar las tomas integradas en la placa base, por la parte trasera del PC, que las que hay en la parte frontal de la carcasa, puesto que estas no son más que simples derivaciones conectadas por cable a la placa.

También necesitaremos gran cantidad de espacio libre en el disco duro para guardar datos —los archivos de imágenes forenses suelen alcanzar tamaños considerables, frecuentemente de varios gigabytes—, y una CPU con potencia suficiente para manejar con soltura aplicaciones de software que no destacan precisamente por su frugalidad en el consumo de recursos: Android SDK, emulador de dispositivos móviles, software de virtualización y otras herramientas por el estilo.

Lo ideal sería disponer de un PC con CPU AMD o Intel, sistema operativo de 64 bits y toda la RAM que admita la placa base: idealmente 8, 12 o 16 GB. No se inhiba a la hora de gastar dinero en memoria.

Si tiene previsto dedicarse también al análisis forense de terminales iPhone, iPad, iTouch, iPod y otros dispositivos de la marca Apple —lo cual no es mala idea, teniendo en cuenta el protagonismo que estos aparatos tienen todavía en el mercado, y que seguramente lograrán mantener durante los próximos años— compensa adquirir un ordenador de sobremesa Apple o un MacBook con OSX 10 o superior. El acceso a las particiones HFS+ de los dispositivos móviles Apple es más eficaz desde una plataforma que utiliza la misma tecnología. Por supuesto, en un Mac también se puede ejecutar el SDK Android y realizar conexiones ADB.

Cellebrite Touch. Para realizar adquisiciones forenses en el lugar de los hechos, así como en dependencias judiciales, despachos de notarios, establecimientos de telefonía móvil o cualquier otro lugar en el que se requiera demostrar profesionalidad mediante un adecuado despliegue de capacidades operativas, vale la pena adquirir uno de estos dispositivos, pese al elevado coste que supone. El UFED Touch de Cellebrite reconoce el 95 % de todos los teléfonos móviles existentes en el mercado, incluyendo modelos chinos. Dispone de pantalla táctil y menús gráficos de fácil manejo. Viene acompañado de un set de cables y conectores que permiten al investigador adquirir prácticamente cualquier cosa, extrayendo los datos de usuario y copiándolos a otro teléfono móvil, una tarjeta de memoria o un PC. También puede realizar duplicados de tarjetas SIM.

Lectores de tarjetas. Imprescindibles para adquirir las tarjetas SD de los dispositivos móviles. No olvide incluir en su equipo un adaptador USB para la lectura de las tarjetas SIM de diferentes formatos que llevan los teléfonos GSM. Los lectores de tarjetas son baratos y se pueden conseguir en cualquier comercio de material informático.

Figura 8.3. Fuente de alimentación regulable

Fuente de alimentación DC. Más importante de lo que parece, sobre todo cuando no se dispone de un alimentador adecuado para el dispositivo móvil investigado o se presentan problemas con la batería. Las fuentes de alimentación de corriente continua, programables por ordenador y provistas de *display* digital (Figura 8.3) permiten regular la energía suministrada, cargar baterías de cualquier clase y mantener con vida un teléfono móvil el tiempo necesario para llevar a cabo tareas de adquisición y análisis. Se pueden encontrar en cualquier tienda de electrónica industrial por menos de cien euros.

Cámara fotográfica digital. Muy importante para registrar información contextual del lugar de los hechos o el entorno en que fue encontrado el dispositivo móvil. También para realizar imágenes de la pantalla y agregarlas a la información disponible sobre el caso. Para situaciones en las que la calidad de la imagen sea decisiva, por ejemplo a la hora de realizar informes y presentaciones, se recomienda adquirir un pequeño trípode.

Discos duros externos. Para almacenar y transportar datos e imágenes procedentes de adquisiciones y análisis forenses, y también para archivar casos, guardar copias de respaldo de nuestro trabajo y otros fines.

Destornilladores de precisión. Nunca se sabe cuándo pueden hacer falta. Ideal sería disponer de un mango magnético con juego completo de puntas intercambiables. Si además de Android piensa dedicarse a la investigación de dispositivos Apple, conviene tener a mano puntas de tipo pentalobe. Se pueden adquirir por poco dinero por Internet o en cualquier tienda de teléfonos chinos.

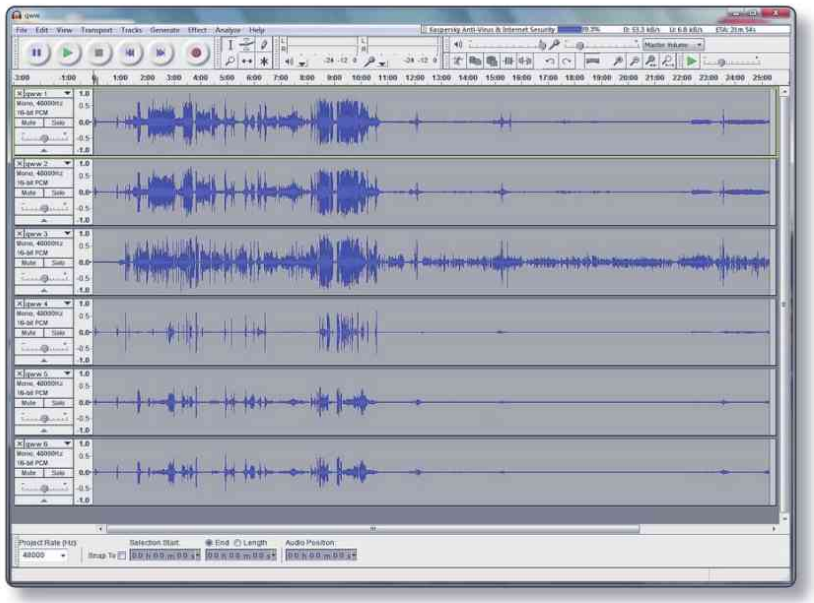

Figura 8.4. Audacity

8.2.2 Software

El software de desarrollo de Android —gratuito y disponible en Internet— y las herramientas de código libre de las que se ha hablado en capítulos anteriores son útiles para resolver la mayor parte de las situaciones. Su dominio es condición imprescindible para dedicarse profesionalmente a la investigación de dispositivos móviles Android. No obstante, es posible que un investigador profesional necesite adquirir uno o varios de los productos comerciales que se mencionan a continuación.

Oxygen Forensic Suite. Software forense para el análisis de teléfonos móviles, *smartphones* y agendas electrónicas.

Compelson MOBILedit! Forensic. Software forense para la adquisición de evidencia digital y el análisis de datos procedentes de teléfonos GSM.

Paraben Device Seizure. Herramienta para la adquisición física y lógica de numerosos modelos de teléfono móvil, con capacidad para descifrar contraseñas e interpretar (*parsing*) estructuras de datos.

Audacity. Software gratuito para analizar archivos de sonido y convertirlos a un formato adecuado para la investigación mediante el empleo de herramientas integradas para realce, amplificación y normalizado de onda (Figura 8.4). Audacity también resulta útil para convertir a archivos de onda el sonido de mensajes de voz guardados en el teléfono.

8.2.3 Dispositivos para bloqueo de la señal electromagnética

Un problema frecuente en la investigación de teléfonos móviles se presenta cuando hay que aislarlos de las redes de comunicaciones de modo que no puedan transmitir información a un usuario remoto, o para evitar que este pueda activar un borrado remoto de los datos almacenados en el dispositivo. El aislamiento se consigue por diversos métodos, por ejemplo introduciendo el teléfono en un recipiente metálico que impida el paso de las ondas de radio. He aquí algunos de los accesorios que los investigadores utilizan para este fin:

Bolsas especiales. Son envoltorios de tela metalizada que producen un efecto similar al de las jaulas de Faraday, y en los que el terminal es introducido para evitar que establezca comunicación con torres de telefonía y puntos de acceso wifi. Estas bolsas distan de ser perfectas, por lo que el investigador debería tomar la precaución de envolver el teléfono en papel de plata —tres capas como mínimo— para apantallar por completo la señal.

Papel de aluminio. Si no tiene a mano una bolsa con costuras metalizadas que formen una jaula de Faraday, vaya al supermercado y compre un rollo de papel de aluminio para envolver alimentos. En alguna página web sobre temas de *hacking* y seguridad informática se cuenta que el mismo efecto se logra con una bolsa de patatas fritas. No se fíe del folclore de Internet: antes haga pruebas con papel de plata de buena calidad, cuanto más grueso mejor. Si con una capa no basta, envuelva el terminal en dos vueltas de papel de aluminio. Experimente con diferentes *smartphones* y modelos de teléfono móvil y asegúrese de que el truco funciona, por si alguna vez tiene que recurrir a él.

No olvide que algunos terminales intentan restablecer el contacto con la red a base de emitir repetidamente señales electromagnéticas en busca de torres telefónicas cercanas. Esto puede hacer que en ocasiones el dispositivo se recaliente hasta el punto de sufrir deterioros en su circuitería u otros componentes. En cualquier caso disminuirá la duración de la batería, por lo que resulta conveniente llevarlo al laboratorio lo antes posible para proceder a la adquisición forense. No envuelva el dispositivo en papel de aluminio con el cable de alimentación puesto, ya que este actuaría como una antena externa dejando pasar la señal de radio.

Inhibidor de frecuencias. Con algo más de presupuesto se puede adquirir a través de Internet un dispositivo capaz de bloquear transmisiones de telefonía móvil y redes inalámbricas (Figura 8.5), impidiendo así que el teléfono utilice los canales de control para establecer comunicación con el exterior. Uno de estos aparatos puede generar hasta 6 vatios de potencia, haciendo imposible el funcionamiento de cualquier teléfono móvil o cliente wifi en un radio de 10 metros. Antes de adquirir un inhibidor de frecuencias es conveniente estar al tanto de lo que leyes y normativas locales establecen sobre el uso de estos dispositivos.

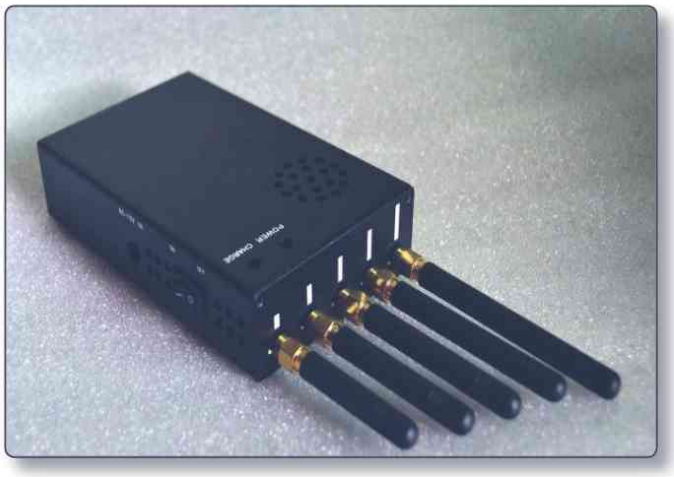

Figura 8.5. Inhibidor de frecuencias portátil para teléfonos móviles y wifi

8.3 INVESTIGACIÓN CONVENCIONAL

En este apartado trataremos de una problemática común a todos los campos de la investigación informática forense, y que tiene que ver con la dificultad de establecer una relación entre la secuencia de acontecimientos detectada en el interior de un sistema digital y la persona concreta que los causa. En el contexto de toda investigación forense, tanto convencional como tecnológica, la atribución de autoría es un asunto de importancia capital, porque de ello depende la evolución del proceso y otras gestiones importantes ante la administración de Justicia. Habiendo intereses considerables en juego, víctimas y damnificados a quienes resarcir, y existiendo la necesidad de resolver de modo inequívoco la inocencia o culpabilidad de un acusado, resulta imposible exagerar la importancia del tema. Un trabajo de investigación metódico y profesional es algo imprescindible para establecer de manera fundada una atribución de los hechos delictivos.

8.3.1 Mundo virtual y mundo físico

Para entender bien la naturaleza del problema jurídico y metodológico del cual estamos hablando, imagine la siguiente situación: un automóvil sale a velocidad excesiva de un túnel y un vehículo radar de la policía hace una foto. La imagen es procesada por un software especial que lee la matrícula, identifica al titular en una base de datos y cursa el aviso de multa correspondiente. Otro programa escribe la notificación, añadiendo las señas del titular, referencias relativas a las normas legales vulneradas, la velocidad real a la que circulaba el vehículo y el importe de la sanción. Luego añade el plazo de pago y los descuentos contemplados por puntualidad en el abono, advertencias en caso de incumplimiento y, finalmente, una funcionarial y cortés fórmula de despedida. Otro sistema automático se encarga de imprimir la carta, meterla en un sobre y ponerla en el correo. Poco después el titular del vehículo y presunto infractor la recibe en su domicilio. Se trata de un proceso impecable y totalmente objetivo. Ninguna persona ha intervenido en él, y por lo tanto la posibilidad de un error humano es prácticamente nula. Un perito al que contrataran para supervisar esta secuencia de operaciones mecánicas no tiene más remedio que llegar a una sola conclusión: hay que pagar la multa.

Sin embargo, la foto sacada por el vehículo radar no muestra la imagen del conductor. El titular puede alegar que se trataba de otra persona, que le robaron el coche —presentando como prueba la denuncia correspondiente— o incluso, en caso de tratarse de un infractor habitual y que esta vez la falta haya sido lo suficientemente grave como para hacerle perder su carnet de conducir, convencer a otro individuo para que declare que era él quien iba al volante del vehículo en el momento en que se sacó la foto. En torno a este ángulo muerto de la vigilancia automatizada florece

una picaresca de suplantaciones retribuidas y compraventa de puntos de carnet que habría resultado imposible en otros tiempos, cuando era un motorista el que daba el alto al conductor para examinar la documentación y extender una papeleta rellena a bolígrafo y arrancada de un bloc. Las máquinas son inteligentes, pero no tanto como para evitar que unos cuantos pícaros se rían del sistema.

Otro ejemplo: un día descubre que alguien ha estado sacando dinero de su cuenta bancaria mediante disposiciones de efectivo en cajeros electrónicos. Usted tiene sospechas de quién puede ser, pues no hay muchas personas que tengan acceso a su tarjeta de débito y conozcan el PIN. Sin embargo los extractos bancarios, las fechas de las disposiciones y los importes, aunque proporcionan datos precisos para elaborar una línea de tiempo, no dicen nada concluyente acerca del autor. Para salir de dudas solo hay una opción: denunciar los hechos ante la policía y conseguir las grabaciones de las cámaras de seguridad.

Es como si existieran dos mundos: uno virtual, cerrado, predecible, compuesto por datos y procesos digitales, que se despliega en el interior de la memoria RAM de los ordenadores y obedece a la lógica determinista de los programas y algoritmos informáticos; otro real, abierto y gobernado por la incertidumbre, donde hay una parte de la información que no es conocida ni podrá serlo jamás, poblado por delincuentes que cometen crímenes con armas reales, palanquetas de acero y dinero falso; un mundo con policías que utilizan medios de investigación convencionales e interrogan pacientemente a los testigos, donde hay jueces y fiscales que viven en mundos de papel todavía no informatizados.

En cualquiera de estos mundos, así como en la zona de transición entre ambos, que pasa por el teclado del ordenador, el éxito de la investigación no depende de la pericia individual ni de la disponibilidad de herramientas adecuadas. Más decisiva resulta la existencia de procesos eficaces de trasvase de información entre los analistas de las distintas especialidades. El ámbito virtual es el dominio del informático forense. Si lo pensamos bien nos daremos cuenta de que este entorno no tiene absolutamente nada que ver con el ámbito físico, donde actúa la policía y la suerte de víctimas y delincuentes es decidida por el personal de la administración de Justicia.

8.3.2 Interrogatorios y tomas de declaración

Un procedimiento eficaz para obtener información en todo tipo de investigaciones consiste en hacer preguntas a la gente. Los interrogatorios y las declaraciones de víctimas, acusados y testigos permiten reconstruir sucesos, descubrir motivaciones y circunstancias relacionadas que puedan ayudar a esclarecer los acontecimientos delictivos que constituyen el objeto de la investigación. Las

respuestas a una batería de preguntas convenientemente planteadas constituyen la más valiosa fuente de información a la que podemos recurrir. Pese a todos los avances de la tecnología, en el futuro esta seguirá siendo la base de la investigación criminológica.

En el ámbito del delito digital las informaciones aportadas en interrogatorios y tomas de declaración sirven para complementar los datos obtenidos por el investigador forense durante su investigación y sus tareas de análisis. Su cometido consiste en salvar la brecha que existe entre el teclado del ordenador y los dedos culpables que se sirvieron de él para perpetrar los actos delictivos. Sin una información precisa al respecto resulta imposible establecer atribuciones de autoría que puedan probarse más allá de cualquier duda razonable. Las preguntas para casos de investigación de la criminalidad informática varían según la gravedad del delito, las circunstancias o las personas a quienes van dirigidas. En este último sentido podemos distinguir varias categorías:

a) **Preguntas dirigidas a la víctima**. La víctima de un delito tecnológico, en su condición como objetivo explícito del ataque, o bien como damnificado, denunciante o testigo, es la primera persona con la que la policía o el investigador establecen contacto. No solo necesita que alguien le preste ayuda tras la agresión sufrida. Los acontecimientos de que ha sido involuntario y pasivo protagonista aún se hallan recientes en su memoria. La intensidad con que la víctima los ha vivido convierte a aquella en un primer punto de referencia.

Las preguntas dirigidas a la víctima no tienen por objeto determinar su competencia en el manejo de ordenadores, sino dilucidar el papel desempeñado por los dispositivos digitales en la comisión del delito. Lo normal es que las víctimas no tengan mucha experiencia en tecnologías de la información. Por este motivo suelen constituir una presa fácil para estafadores, pedófilos y ciberdelincuentes. Poniendo en claro el modo en que una víctima ha sido atacada quizás encontremos pistas que nos ayuden a averiguar la identidad de su agresor.

Las preguntas para la víctima, por lo general y de modo preferente, tratarán de las siguientes cuestiones:

- Frecuencia con que la víctima usa el ordenador o el teléfono móvil.
- Ataques de virus sufridos en el pasado y en el presente; sistemas de defensa con los que cuenta el usuario (antivirus, cortafuegos, etc.).
- Personas a las que ha facilitado información o datos personales.
- Recepción de mensajes de *phishing* solicitando claves de acceso, y en qué medida el usuario ha respondido a los mismos.

- Quejas de otros usuarios por haber recibido *spam* o mensajes ofensivos desde la cuenta de la víctima.
- Actividad extraña en cuentas bancarias, correo electrónico y otros servicios *on line*.
- Actividad extraña en perfiles de Facebook, Twitter y otras redes sociales.
- Recepción de mensajes ofensivos, amenazas, anónimos, extorsiones o injurias de cualquier tipo.
- Publicación en Internet de imágenes personales o información privada perteneciente a la víctima.
- Suplantaciones y robo de identidad en foros, redes sociales o servicios *on line*.
- Personas que en opinión de la víctima pueden ser sospechosas de los actos anteriores.
- Existencia de pruebas (correos electrónicos, mensajes, copias impresas en papel) que respalden las sospechas de la víctima.

b) **Preguntas para administradores de sistemas**, **compañías telefónicas o proveedores de acceso a Internet**. El responsable de la infraestructura constituye un eslabón crucial en el proceso de esclarecimiento de los delitos tecnológicos. Esto resulta aún más cierto en el caso de unos dispositivos que, como los teléfonos móviles y los *smartphones*, fueron diseñados para estar conectados en todo momento a redes de telecomunicaciones. Los registros de la compañía telefónica o el proveedor de acceso, que en algunos países la ley obliga a guardar durante un período de tiempo determinado, proporcionan información sobre números marcados y entrantes, fechas y horas de las llamadas y posición geográfica de las torres de telefonía móvil con las que el sospechoso establece contacto en el transcurso de sus desplazamientos.

Así mismo quedan registradas las direcciones IP desde las cuales el usuario se conecta a Internet, y que pueden corresponder a *routers* domésticos o de empresa, puntos públicos de acceso wifi o antenas telefónicas, en caso de que la conexión se lleve a cabo a través de redes de datos 3G o 4G.

Para recabar información de compañías telefónicas y proveedores de acceso se necesita una orden judicial. Numerosos países han modificado sus leyes para liberar de responsabilidad civil a los operadores telefónicos y los proveedores de acceso a Internet que colaboran con la policía facilitando información sobre sus clientes, por lo que un investigador debidamente autorizado no debería encontrar ninguna dificultad para acceder a informaciones de esta categoría. Las preguntas

dirigidas al responsable de la infraestructura tendrán que ver con datos de registro, titulares de líneas y cuentas de acceso y otras informaciones por el estilo.

c) **Preguntas dirigidas al sospechoso**. El interrogatorio destinado al sospechoso constituye un elemento crucial de la investigación. El escenario más favorable se presenta cuando el sospechoso coopera voluntariamente. Como es lógico, también puede hacer uso de su derecho a no declarar. Entre ambas situaciones existe un amplio rango de posibilidades circunstanciales y, en ocasiones, negociables. Al ser la única persona implicada en el caso que no está obligada a decir la verdad, no hay que hacerse muchas ilusiones sobre el carácter objetivo de lo que nos pueda contar. Esto, en cualquier caso, y aunque pueda parecer paradójico, tampoco es importante. Tanto si el sospechoso dice la verdad como si miente, el objetivo del investigador no consiste en poner de manifiesto contradicciones ni incoherencias con el propósito de empeorar su situación ante la justicia, sino en obtener informaciones contrastables con los resultados de la investigación o el testimonio de otras personas.

El cuestionario dirigido a un sospechoso debe plantearse de acuerdo con la naturaleza de los delitos. No es lo mismo el espionaje industrial que los crímenes de pederastia, extorsión o acoso moral. Al efecto de esclarecer los hechos y poder llevar a cabo atribuciones de autoría fundadas interesa, sobre todo, obtener respuestas a preguntas relacionadas con los extremos siguientes:

- Preguntas relativas a la **competencia** del acusado **en tecnologías de la información**: si tiene estudios de informática, dónde los realizó, qué sistemas operativos conoce, si domina algún lenguaje de programación, si tiene costumbre de desfragmentar su disco duro, etc.

- Preguntas sobre conocimiento y empleo de técnicas de **encriptación**, **borrado seguro de datos** y otras cuestiones relacionadas: aquí el sospechoso con toda probabilidad se va a mostrar reticente. Se recomienda introducir este bloque de preguntas como mero formalismo, sin darle demasiada importancia. No conviene utilizar un tono imperativo o directo a la hora de dirigirse al acusado. No es inteligente plantear preguntas del tipo: "Díganos esto o lo otro", o "¿Por qué hizo lo de más allá?". Formule sus peticiones de un modo indirecto, con perífrasis que atenúen el efecto coercitivo del interrogatorio, como por ejemplo: "¿Nos podría decir si alguna vez se ha visto en situación de…?". Pregunte al acusado si emplea técnicas esteganográficas o de ocultación de datos, si tiene volúmenes ocultos, qué software utiliza. Y si para proseguir las investigaciones fuera necesario disponer de contraseñas que solo el sospechoso conoce, pregúnteselas con amabilidad y franqueza, sin dramatismos de ningún tipo.

- Preguntas relativas a **dispositivos informáticos** en poder del sospechoso. No solo de su propiedad, sino también aquellos que están bajo su control o a los que tiene acceso: ordenadores de sobremesa y portátiles, teléfonos móviles, soportes de almacenamiento, etc. Intente así mismo averiguar para qué utiliza estos dispositivos y si otras personas tienen acceso a ellos.

- Preguntas relativas al **software** utilizado por el sospechoso: sistemas operativos, suites ofimáticas, aplicaciones Android o iOS. También interesa conocer si acostumbra a descargar programas de Internet, *apps* de mercados no oficiales u otros contenidos; si alguna vez ha modificado el código de algún programa informático, si es capaz de desarrollar sus propias aplicaciones y otras cuestiones por el estilo que guarden relación con los hechos que se investigan.

- Preguntas relativas al **uso de Internet** por el sospechoso: si dispone de acceso desde su hogar u otros sitios, y qué tipo de conexiones tiene: wifi, 3G, 4G, etc. Así mismo qué navegadores de Internet utiliza, además del *browser* por defecto (MS Explorer en Windows, Safari en OSX/iOS o WebKit en Android). Interesa conocer si el sospechoso emplea algún sistema para ocultar direcciones IP, si al final de cada sesión borra el historial de Internet, qué tipo de contenidos busca en la Red y si realiza compras en sitios de comercio *on line*. Finalmente, es importante saber si ha utilizado redes virtuales, servicios en la nube o redes anónimas del tipo Tor.

- Preguntas sobre **soportes de almacenamiento**: si el acusado guarda archivos en el disco duro de su ordenador o posee discos duros externos, llaves USB, CD/DVD, unidades ZIP, cintas de datos, etc. Así mismo, si los archivos son compartidos con otras personas o hay alguien más que tenga acceso a ellos.

- Preguntas sobre utilidades de descarga y **software P2P**. Estas cuestiones no solamente tienen sentido en relación con ordenadores de sobremesa portátiles, sino también, y cada vez en mayor medida, en el ámbito de la informática móvil, ya que algunos programas de intercambio de archivos —entre ellos varios clientes bittorrent— han sido portados a Android.

- **Chat**, **foros**, **telefonía IP**: la información obtenida en el análisis de las respuestas a este tipo de preguntas puede resultar útil para comprobar coartadas y localizar a otras personas relacionadas con los hechos, como cómplices, testigos, víctimas o personas sospechosas de haber cometido o estar cometiendo delitos similares.

- **Pederastia y pornografía infantil**. El cuestionario dirigido a pedófilos y traficantes de pornografía infantil es el más delicado y problemático. A la hora de elaborarlo el investigador debe estar atento a la naturaleza y la gravedad de los hechos criminales que se investigan, la estrategia de la investigación y aspectos de atención prioritaria señalados por el ministerio fiscal, el juez o el personal de apoyo —médicos forenses, psicólogos criminalistas, peritos en psicología infantil y otros especialistas—. Habrán de plantearse al sospechoso preguntas de carácter general relativas a sus hábitos de consumo de pornografía infantil y su grado de conciencia acerca del carácter grave de esos crímenes, así como a su conocimiento sobre las implicaciones delictivas de lo que hace. También se deberán plantear cuestiones más concretas relacionadas con sus hábitos de navegación, la forma en que obtiene, guarda o redistribuye el material y si conoce en persona a los menores que salen en las imágenes o a los cómplices que pudiera tener en los actos delictivos.

8.3.3 Objetivos de los procedimientos convencionales

El empleo de métodos de investigación convencionales obedece al propósito de superar las limitaciones de los procedimientos específicos de la informática forense. Basándose tan solo en lo que sabe por el análisis de los elementos de evidencia adquiridos a través de herramientas digitales, resulta problemático determinar con certeza la autoría de la acción criminal cuando no se dispone de información complementaria procedente del mundo físico. El cometido del investigador consiste en obtener información que permita decidir acerca de la existencia o no de un acto delictivo cometido de manera intencionada y culpable. Todo procedimiento que sirva para arrojar claridad sobre los hechos, esté basado en métodos convencionales o de alta tecnología, es útil para los fines que se persiguen, siempre que se aplique de modo correcto y eficaz en el ámbito que le corresponde.

En general los métodos de investigación que se emplean son lo bastante eficaces para resolver una parte importante de los casos. En ocasiones se cometen errores de atribución con graves consecuencias. Afortunadamente estos errores son cada vez más raros debido al continuo avance en las tecnologías y los métodos de investigación, pero no por ello dejan de tener, cuando ocurren, consecuencias nefastas. El individuo falsamente acusado de pederastia por culpa de un teléfono troyanizado o un punto de acceso wifi abierto no recupera su reputación por mucho que se reconozcan públicamente los errores judiciales cometidos.

La realización de entrevistas adaptadas a la investigación de delitos tecnológicos aspira a resolver el problema de las falsas atribuciones y lograr una

conclusión correcta de la labor de investigación y de las diligencias judiciales. Sus objetivos son concretos: adquirir informaciones contextuales útiles para la investigación, cubrir lagunas, ordenar y sistematizar de manera adecuada los conocimientos relacionados con el hecho delictivo.

En resumidas cuentas: el proceso de investigación no consiste en un despliegue de tecnologías sofisticadas que permitan en obtener la mayor cantidad de información posible sobre los hechos, sino en llevar a cabo una actividad intelectual basada en fundamentos científicos que permita hallar respuestas claras a los interrogantes fundamentales de todo proceso de esclarecimiento de actos delictivos: **qué**, **quién**, **cuándo**, **dónde**, **cómo** y **por qué**. Responder de modo fundado a estas cuestiones es explicar los hechos. Con ello hacemos posible que la autoridad encargada de sancionarlos tome decisiones libres de error.

8.3.4 Misión del investigador forense

En todo proceso de investigación forense existe una contradicción: por un lado se aspira a alcanzar conclusiones objetivas al margen de elementos subjetivos o intereses personales. Por otro, el mecanismo de planteamiento de interrogantes, análisis de evidencias y deducciones lógicas se lleva a cabo en la cabeza de un ser humano. En su aspiración ideal, la ciencia forense posee un carácter tan impersonal como las pesquisas de la policía y el trabajo de los magistrados. No se trata de que alguien decida por su propia voluntad si un acusado es culpable o no. Idealmente, en un Estado de Derecho nadie debe tener facultad para resolver acerca de tal cosa basándose en criterios propios. El objetivo consiste más bien en hacer que las respuestas acerca de la culpabilidad o inocencia del sospechoso surjan por sí mismas a partir de una explicación fundada y completa de los hechos.

Se trata de llegar a un punto en que las situaciones estén explicadas con la claridad suficiente para que puedan resolverse por sí mismas. Que un juez tome la decisión es inevitable, pero, en último grado, accidental. Lo deseable sería que el trabajo de investigación esté tan bien hecho que las respuestas caigan por su propio peso. Quizás sea mucho pedir, pero al menos aporta énfasis a la cuestión esencial de que en todo trabajo de investigación forense lo que se necesita no son opiniones sino hechos comprobables. Se trata de una labor acumulativa. El investigador no debe creer que la solución del caso depende de una pieza fundamental que él pueda localizar en un momento dado. Solo hace su parte del trabajo y debe hacerla bien, para que las piezas encajen y se encuentre la respuesta a los seis interrogantes básicos: qué, quién, cuándo, dónde, cómo y por qué.

8.4 OTROS ASPECTOS DE LA INVESTIGACIÓN

8.4.1 Realización y presentación de informes

El producto final de la actividad del investigador es un informe con los resultados de su trabajo, que posiblemente tenga que ser expuesto y defendido ante un tribunal. Dicho documento debe incluir una descripción detallada y exacta de los hechos. En él se habrá de dejar constancia de las operaciones practicadas, los elementos de evidencia hallados y cualquier otro extremo que pudiera resultar de interés. El investigador no debe hacer conjeturas ni aventurar conclusiones sobre las cuales corresponda decidir a jueces y fiscales. Su labor consiste en hallar elementos de evidencia que habrán de ser interpretados de modo profesional en el contexto técnico dentro del cual desempeña su labor.

Cada actuación debe ajustarse a derecho y a catálogos de buenas prácticas, para que el informe no pueda ser impugnado por omisiones o defectos de forma. Debe existir una trazabilidad adecuada de todo aquello que haya servido como material para elaborarlo. Especial cuidado ha de ponerse en el elemento fundamental de toda la ciencia forense: el mantenimiento de la cadena de custodia. El informe no solo es un resumen del proceso de investigación; forma parte de él, y como tal ha de cumplir unos determinados requisitos de solvencia: aceptabilidad, integridad, credibilidad, existencia de una relación causa-efecto, carácter repetible y una documentación coherente y completa. Numerosos investigadores elaboran sus informes de acuerdo con estándares reconocidos, como por ejemplo la Directiva RFC 3227 o la Norma ISO/IEC 27037.

Los tribunales comprueban minuciosamente los elementos de evidencia presentados por los investigadores forenses, y también sus métodos de trabajo. Un profesional contribuye al sostenimiento de su credibilidad cuando hace afirmaciones solventes, se cuida de que la presentación de las pruebas sea impecable y no hace nada que implique una vulneración de derechos fundamentales ni un ataque contra la privacidad ni las leyes de protección de datos. Este es un aspecto delicado en la investigación de dispositivos móviles, que nos lleva al último, pero no por ello menos importante, apartado de este capítulo final.

8.4.2 Implicaciones jurídicas

El autor no quiere terminar esta obra sin hacer un último esfuerzo de honestidad. Vaya por delante que todo lo que el lector ha aprendido en este libro puede tener un plazo de caducidad decepcionantemente corto. Culpa de ello no es mía sino

del progreso técnico. A la velocidad con que se suceden los avances en informática móvil y en el sector de las telecomunicaciones, todo lo que aprendemos se queda obsoleto en cuestión de pocos años, a veces incluso de pocos meses. Permanezca alerta y dedique la mayor parte de su esfuerzo a la formación continua. El resto deberá emplearlo en mantenerse en guardia contra contingencias e imprevistos. Recuerde que Murphy jamás descansa, y que en el ámbito de la informática forense tiene la desagradable costumbre de atacar no tanto a través de fallos en software o fuentes de alimentación defectuosas como desde los más inesperados recovecos del sistema legal.

Por consiguiente, será necesario que permanezca al tanto de lo que dicen la Agencia de Protección de Datos y las sentencias de los tribunales. Para poner a prueba lo aprendido o adquirir práctica en el manejo de herramientas forenses se recomienda utilizar dispositivos que sean de su exclusiva propiedad. Y si en el transcurso de una investigación oficial tiene que probar uno de los métodos de adquisición por medio de particiones Recovery alternativas descritos en el capítulo 4, produciendo así una alteración limitada en los medios de prueba, ensaye antes con un dispositivo de marca, modelo y características similares. Asegúrese de que el terminal no queda inutilizado. Aunque después pueda volver a ponerlo en funcionamiento, existe el riesgo de perder o invalidar elementos de evidencia que podrían resultar cruciales para un juicio. Finalmente, no piense que las particiones Recovery pueden ser la panacea. En esto la última palabra no la tienen los expertos en tecnología sino los juristas. Y ello por buenas razones.

La protección de la privacidad es un derecho fundamental reconocido por la Constitución. De ello se derivan consecuencias importantes para la investigación forense. Si adquirir el correo electrónico o los archivos personales de un usuario de ordenadores de sobremesa o portátiles supone una delicada operación que requiere de una orden judicial, el acceso a los datos guardados en un dispositivo móvil es el equivalente electrónico de un registro corporal. Es de prever que en el futuro la jurisprudencia se irá poblando de casos con todo tipo de situaciones conflictivas en torno a dispositivos móviles, protagonizadas por agentes de policía, traficantes de droga, empleados desleales, hackers, cónyuges celosos o simples gamberros provistos de tabletas y teléfonos móviles inteligentes.

Veremos cómo los jueces resuelven estos conflictos con una lógica que a veces nos parecerá comprensible, pero otras entrará en conflicto con lo que la opinión pública considera que debería ser justo y de sentido común. Descubriremos que un simple error de trámite, un exceso de celo en las pesquisas o una transgresión accidental de la intimidad pueden arruinar la prueba principal de un juicio, dejando en libertad a alguien que, en el supuesto de haberse hecho las cosas como se debe, habría ido a parar a prisión. Por su parte, la condena de un inocente por culpa de

un tratamiento equivocado de la evidencia, aunque menos probable, sería aún más grave. De cualquier modo, y poco a poco, la doctrina legal irá puliéndose con cada caso sucesivo y adaptándose a un contexto delictivo caracterizado por el uso masivo e irreversible de las nuevas tecnologías.

De modo que hay que extremar las precauciones. Con frecuencia los motivos para examinar lo que hay dentro de un dispositivo móvil no son tan importantes como los que existen para no tocarlo.

BIBLIOGRAFÍA

ANDROULIDAKIS, I. A.; *Mobile phone security and forensics. A practical approach*, Springer, 2012.

BERGMAN, STANFIELD, ROUSE, SCAMBRAY; *Hacking exposed – Mobile: Security secrets & solutions*, McGraw Hill, 2013.

CARRIER, B.; *File system forensic analysis*, Addison-Wesley, 2005.

FEDLER, BANSE, KRAUSS, FUSENING; "Android OS security – Risks and limitations. A practical evaluation", AISEC Reports, Fraunhofer Research Institution AISEC, mayo 2012.

GALÁN-GARCÍA, SANZ URQUIJO, LAORDEN GÓMEZ, GARCÍA BRINGAS; *Eludiendo la concesión de permisos de administrador en Android mediante una vulnerabilidad en SuperAgent*, documento digital editado por los autores en Internet, 2012.

GHOST; "Ingenería inversa en aplicaciones de Android I y II", en RedInfoCol (*http://www.redinfocol.org*), 18 y 20 de noviembre de 2011.

HOOG, A.; *Android forensics – Investigation, analysis and mobile security for Google Android*, Syngress Elsevier, 2011.

HOOG, A. y STRZEMPKA, K.; *iPhone and iOS forensics – Investigation, analysis and mobile security for Apple iPhone, iPad and iOS Devices*, Syngress Elsevier, 2011.

JOVANOVIC, Z.; *Android forensics techniques*, International Academy of Design and Technology, January 2012.

LESSARD, J. y KESSLER, G. C.; "Android forensics: Simplifying cell phone examinations", *Small Scale Digital Device Forensics Journal*, N.º 4, Sept. 2010.

MORENO ÁLVAREZ, M. Á.; *Desarrollo de aplicaciones Android seguras*, Informática64, 2012.

SHAVERS, B., *Placing the suspect behind the keyboard – Using digital forensics and investigative techniques to identify cybercrime suspects*, Syngress Elsevier, 2013.

SIX, J.; *Application security for the Android platform*, O'Reilly, 2012.

SPREITZENBARTH, M.; *Dissecting the droid. Forensic analysis of Android and its malicious applications* (dissertation), Technische Fakultät der Universität Erlangen-Nürnberg, 19.12.2012 (disponible para descarga en Internet).

SPREITZENBARTH, M. y FREILING, F.; *Android malware on the rise – Technical report CS-2012-04* (dissertation), Technische Fakultät der Universität Erlangen-Nürnberg, abril 2012 (disponible para descarga en Internet).

TOMÁS GIRONÉS, J.; *El gran libro de Android*, Marcombo, 2013.

TYLER, J. con VERDUZCO, W.; *XDA-Developers' Android hacker's toolkit – The complete guide to rooting ROMs and theming*, Wiley, 2012.

Páginas web:

Estas son solo algunas de las páginas web de referencia más importantes citadas en el texto de la obra. Para todo aquel que desee ampliar sus conocimientos sobre forénsica de Android existe en Internet gran número de enlaces disponibles directamente a través de Google y otros buscadores.

- ✓ *https://source.android.com/*
- ✓ *http://developer.android.com/sdk/index.html*
- ✓ *http://www.vmware.com/es*
- ✓ *https://www.virtualbox.org/*
- ✓ *http://www.ubuntu.com/*
- ✓ *http://www.sleuthkit.org/*
- ✓ *http://www.xda-developers.com/*
- ✓ *http://www.clockworkmod.com*
- ✓ *http://www.cellebrite.com/es/homepage*
- ✓ *http://www.oxygen-forensic.com/en/*
- ✓ *http://forensics.spreitzenbarth.de/*
- ✓ *http://columna80.wordpress.com/category/android/*

ÍNDICE ALFABÉTICO

Símbolos

3G, 22, 131, 132, 189, 190, 251, 253
4G, 22, 96, 189, 190, 251, 253

A

Acelerómetro, 30
Activities, 46, 47, 48
Activity, 47, 48
Adb, 32, 52, 70, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 122, 123, 124, 130, 136, 137, 138, 140, 141, 146, 147, 150, 152, 153, 154, 161, 162, 166, 173, 175, 234, 244
Adbd, daemon, 152
Adb shell, 110
ADEL, 18, 222, 231, 232, 233, 234
Administrador, 42, 66, 69, 71, 79, 80, 81, 82, 95, 117, 119, 138, 190, 196, 203, 210, 259
Administrador de tareas, 94, 95
Adquisición lógica, 150, 151
Adware, 209, 213
AES 256, 240
Agencia de Protección de Datos, 257
Ajustes, Android, 95, 96, 107, 117, 132, 133, 136, 137
Aleph One, 187
Almacenamiento en red, 159
Almacenamiento externo, 159
AMD, 58, 74, 244
AMOLED, 28
Android.Adrd, 214
Android Inc., 34
AndroidManifest.xml, 46, 47, 166, 167, 168, 169, 171, 194, 201, 211
Android Market, 35, 40, 45, 49, 158, 191, 206, 207, 213
Android Open Source Project (AOSP), 35, 37
Android.Tapsnake, 212
Anfitrión, 54, 56, 57, 58, 59, 60, 61, 62, 72
API, 39, 45, 67, 68, 71, 73, 74, 159, 161, 165, 174, 212
APK, paquete, 44, 50, 141, 207, 209
Apktool, 168, 169, 171
Aplicaciones, 15, 22, 24, 29, 30, 32, 33, 35, 36, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 52, 53, 54, 55, 56, 57, 59, 60, 61, 67, 68, 72, 73, 74, 89, 92, 93, 94, 95, 96, 106, 109, 112, 113, 114, 115, 116, 117, 119, 120, 121, 122, 125, 127,

129, 130, 133, 136, 137, 139, 141, 142, 143, 144, 145, 146, 147, 155, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 168, 171, 172, 173, 176, 178, 179, 180, 182, 190, 191, 192, 193, 194, 195, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 216, 222, 230, 233, 236, 237, 238, 239, 240, 241, 243, 253
App_decrypt, carpeta, 168, 169
Apple, 15, 18, 22, 27, 34, 40, 57, 72, 152, 160, 190, 191, 207, 209, 226, 235, 236, 237, 238, 239, 240, 241, 242, 244, 245, 259
Apple Store, 27, 40, 191
APT, Advanced Persistent Threat, 220
Apt-get, 65, 80, 100, 104, 173
Aptitude, 80
Archivo .dex, 44
ARM, 24
ARP, caché, 218
Arquitectura servidor-cliente, 43, 107
ASCII, 178, 184, 186
Asus, 25
Atom, 25
Audacity, 246
Audiovisuales, 33, 48, 85
Autopsy, 101
Avast, 197
AVD (Android Virtual Device), 72

B

Backup, 147, 150, 151, 157, 160
Baidu, 214
Big data, 181
Binder IPC, 42, 43
Blackberry, 22, 35, 36
Bloatware, 117
Bloqueador de escritura, 85, 97
Bloqueo por código, 131, 132
Bluetooth, 25, 38, 78, 96, 131, 189, 224, 228
Bookmarks, 175
Botnet, 112, 171, 200
Bridging, 58
Broadcast receivers, 48
Browser.db, 163, 175, 178
Bug, 121, 198, 199
Bugreport, 116
Busybox, 123, 147, 152
BYOD, 209

C

Caché, 52, 116, 122, 146, 151, 153, 182, 218
Canalización, pipelining, 78
Cat, comando, 116
C/C++, 43
Cd, 71, 76, 77, 116, 123, 153, 162
CDMA, 114, 131, 132, 190
Cellebrite, 222, 223, 224, 225, 226, 227, 242, 243, 244
Chip-Off, 155, 156, 241
Chmod, 81, 123
Chown, 81, 82
Cifrado hardware, 241
ClockWorkMod, 146, 147, 148, 149, 150, 153, 234
Cloud computing, 38, 55, 59
com.android.browser, 162, 177, 178
com.android.providers.contacts, 179
com.android.providers.media, 180
com.android.providers.telephony, 177
com.google.android.gm, 182
com.google.android.youtube, 181
CompactFlash, 26
Conexiones inalámbricas, 132
Conexiones inalámbricas, 96

Consola, 63, 64, 66, 72, 75, 105, 108, 113, 121, 136, 138, 140, 152, 173
Contactos, 45, 89, 158, 179, 233
Contacts2.db, 179
Content providers, 48
Cookies, 175, 217, 219
Cortafuegos, 195, 203, 250
Cp, 81
Cpio, 147
CPU, 17, 22, 24, 25, 26, 50, 52, 55, 56, 57, 58, 59, 74, 142, 206, 241, 243
CSV, archivos de texto, 175, 183
Cupcake, 38, 39
CWM, 147, 148, 149, 150, 151, 152, 153, 215
Cydia, 207, 239, 240, 241
Cygwin, 54

D

DAC, 193
Daemon, 108, 109, 147, 152, 234
Dalvik, 44, 45, 53, 119, 145, 157, 171, 192, 193, 200, 240
Data carving, 51, 74, 103, 104, 105, 187, 241, 242
Dd, 79, 97, 98, 99, 101, 102, 103, 105, 111, 122, 123, 124, 147, 153, 154, 240, 241
Dependencias, 69, 80, 244
Depuración USB, 106, 137
Desbordamiento de buffer, 203
Devpts, 52, 111, 139
DEX, 44
Dex2jar, 170, 171, 172
DEX (Dalvik Execution Code), 44
Df, comando, 111, 115, 140, 141
Dmesg, 78, 79, 98, 113, 114
DMG, archivo de imagen, 67
DOM, 165, 166
Donut, 38, 39
Download, modo, 103, 125, 143
Downloads.db, 182
Dpkg, 65, 80
Dropbox, 227
DropBox, 160
Dumpstate, 115, 116
Dumpsys, 115, 116
Dx, 44
Dynatac 8000X, 21

E

EasyRecovery Pro, 104
Eclair, 38, 39
Eclipse, 68, 171, 176, 191
eMMC, 50
EnCase, 18, 82, 98, 104, 157, 242
EnCase Forensics, 18, 98, 157
Enlace ascendente (uplink), 201
Enlace descendente (downlink), 201
Entorno de aplicaciones, 45, 174, 237
Envenenamiento, 218
Estructura en árbol b, 233
Ethernet, 62
Evernote, 227
Excel, 175, 183
Exchange, MS, 179
Exploit, 123, 125, 203
Ext2, 49, 51
Ext3, 49, 50, 51, 82, 101
Ext4, 23, 38, 49, 50, 51, 112, 139, 140, 142, 151, 153, 161, 187
Exynos 4 Quad, 25

F

Facebook, 96, 158, 179, 220, 227, 233, 251
Faraday, bolsa, 130, 224, 246
Fastboot, 145, 146, 234
FAT16, 49, 101
FAT32, 26, 27, 49, 50, 82, 97
Favoritos, 89

Fedora, 59, 65
File, comando, 82, 83, 101, 103, 154, 157
File sharing, 158
Find, 79, 115
Finder, 67
Firefox, 173
Firmware, 32, 143, 240, 241
Fls, 102, 103
Foremost, 105, 154, 187
Frame, 208
Framework, 45
Fraudware, 206
FreeBSD, 62
Free Hidrive, 160
Froyo, 38, 39, 122
Fsstat, 101
FTK, 18, 82, 98, 104, 157
FTK Imager, 18

G

G1, 35
Galería, aplicación, 92
Gateway, 218
Geolocalización, 41, 45, 48, 178
Gestor de Memoria, 95
Gesture.key, 137, 138
GetDataBack, 104
GET, petición HTTP, 201
GIF, 178
Gingerbread, 38, 39, 68, 107, 125, 137
GingerBreak, 123, 202
Giróscopo, 17, 30
Gmail, 38, 40, 45, 90, 91, 158, 166, 168, 169, 179, 182
Gmail.db, 182
Gominola, 38
Google, 16, 30, 33, 34, 35, 36, 37, 38, 39, 40, 45, 46, 51, 67, 70, 91, 92, 93, 96, 107, 121, 131, 148, 158, 161, 168, 171, 176, 181, 182, 187, 191, 192, 199, 200, 206, 207, 209, 216, 226, 227, 237, 238, 259, 260
Google Glass, 30
Google Maps, 38, 226
Google Nexus, 16, 34
Google Play, 191
GPL, licencia, 37, 49, 62
GPRS, 230
GPS, 26, 28, 29, 33, 50, 96, 114, 158, 181, 212, 225, 226, 230, 233
GPU, 28
Grep, 77, 78, 79, 115
GRUB, 144
Grupos, 89
GSM, 22, 25, 29, 114, 131, 132, 189, 190, 230, 244, 246
Guest, 56, 59
Gummi Bear, 38

H

Hash, 79, 99, 129, 229
HDMI, 189
Heimdall, 137, 146, 148
Hexadecimal, 50, 51, 82, 180, 183, 184, 185, 186, 187
HFS+, sistema de archivos, 241, 242, 244
Hilos de ejecución, 43
Hipervisor, 55, 58, 59, 61
Historial de llamadas, 89, 158
Home, botón, 95
Honeycomb, 38, 39
Host, 56, 59, 208
HP iPaq, 35
HSDPA, 25
HSUPA, 25
HTC, 34, 70, 121, 127, 187, 227, 243
HTML, 163, 164, 166, 178, 182, 199, 226
Hyper-V, 55

Fedora, 59, 65
File, comando, 82, 83, 101, 103, 154, 157
File sharing, 158
Find, 79, 115
Finder, 67
Firefox, 173
Firmware, 32, 143, 240, 241
Fls, 102, 103
Foremost, 105, 154, 187
Frame, 208
Framework, 45
Fraudware, 206
FreeBSD, 62
Free Hidrive, 160
Froyo, 38, 39, 122
Fsstat, 101
FTK, 18, 82, 98, 104, 157
FTK Imager, 18

G

G1, 35
Galería, aplicación, 92
Gateway, 218
Geolocalización, 41, 45, 48, 178
Gestor de Memoria, 95
Gesture.key, 137, 138
GetDataBack, 104
GET, petición HTTP, 201
GIF, 178
Gingerbread, 38, 39, 68, 107, 125, 137
GingerBreak, 123, 202
Giróscopo, 17, 30
Gmail, 38, 40, 45, 90, 91, 158, 166, 168, 169, 179, 182
Gmail.db, 182
Gominola, 38
Google, 16, 30, 33, 34, 35, 36, 37, 38, 39, 40, 45, 46, 51, 67, 70, 91, 92, 93, 96, 107, 121, 131, 148, 158, 161, 168, 171, 176, 181, 182, 187, 191, 192, 199, 200, 206, 207, 209, 216, 226, 227, 237, 238, 259, 260
Google Glass, 30
Google Maps, 38, 226
Google Nexus, 16, 34
Google Play, 191
GPL, licencia, 37, 49, 62
GPRS, 230
GPS, 26, 28, 29, 33, 50, 96, 114, 158, 181, 212, 225, 226, 230, 233
GPU, 28
Grep, 77, 78, 79, 115
GRUB, 144
Grupos, 89
GSM, 22, 25, 29, 114, 131, 132, 189, 190, 230, 244, 246
Guest, 56, 59
Gummi Bear, 38

H

Hash, 79, 99, 129, 229
HDMI, 189
Heimdall, 137, 146, 148
Hexadecimal, 50, 51, 82, 180, 183, 184, 185, 186, 187
HFS+, sistema de archivos, 241, 242, 244
Hilos de ejecución, 43
Hipervisor, 55, 58, 59, 61
Historial de llamadas, 89, 158
Home, botón, 95
Honeycomb, 38, 39
Host, 56, 59, 208
HP iPaq, 35
HSDPA, 25
HSUPA, 25
HTC, 34, 70, 121, 127, 187, 227, 243
HTML, 163, 164, 166, 178, 182, 199, 226
Hyper-V, 55

I

Icat, 103
Ice Cream Sandwich, 38, 39
ICQ, 227
IEEE 802.11, 25
IEEE 1149.1, 155
IMAP, 220
Informática64, 228, 260
Infosfera, 17
Ingeniería inversa, 170, 171, 172, 213, 239
Ingeniería social, 41, 198, 205, 206, 209, 211, 217, 220
Init, 145
Initramfs, 145
Initrd, 145
Innotek, 61
Inodes, 102
Insert, 115
Intel, 25, 57, 58, 59, 74, 244
Intent, 47, 48
Intercepter-ng, 95, 217, 218, 219
Internal.db, 180
Internet de las cosas, 17
Internet industrial, 17
Internet, permiso, 201
iOS, 18, 22, 40, 207, 222, 226, 235, 236, 237, 238, 239, 240, 241, 253, 259
iPhone, 15, 22, 27, 34, 35, 36, 40, 207, 226, 230, 235, 237, 238, 239, 240, 241, 242, 259
IPL, Initial Program Loader, 144
iSCSI, soporte, 62
ISO, imagen, 58, 62, 256
iTunes, 27, 238, 240, 242

J

Jailbreaking, 15, 191, 207, 238, 239, 240
Jaula de Faraday, 131, 247
Java, 44, 45, 46, 68, 69, 74, 165, 169, 171, 191, 192
JavaScript, 203
JDK (entorno de desarrollo de Java), 69
Jelly Bean, 38, 39, 68, 107, 148
Jerarquía de directorios standard, 160
JFS, 49
Journaling, 51, 97, 101, 153
JPG, 82, 178
JTAG, 155, 156, 198, 200, 241

K

Kernel, 42, 49, 59, 65, 66, 74, 78, 110, 111, 113, 140, 145, 146, 234, 237
Kies, Samsung, 70, 106, 125, 148, 242
KitKat, 39
Konqueror, KDE Linux, 178
Kubuntu, 104, 137

L

Lenovo, 25
Less, 77, 78, 186
LG, 34, 122, 137, 144, 227
LibreOffice, 61, 173
Librerías, 39, 43, 44, 48, 52, 67, 74, 80, 194, 237
Licencia General P˙blica (OPL), 43
LinkedIn, 158, 179, 220
Linux, 16, 22, 33, 35, 38, 42, 49, 50, 51, 52, 53, 54, 56, 57, 59, 61, 63, 65, 66, 67, 69, 71, 72, 74, 75, 76, 78, 79, 80, 81, 82, 97, 98, 99, 100, 102, 104, 105, 107, 108, 109, 111, 112, 113, 115, 118, 120, 121, 137, 140, 142, 144, 145, 146, 147, 148, 151, 152, 153, 154, 157, 160, 161, 168, 169, 173, 174, 176, 178, 183, 185, 186, 187, 190, 191, 193, 194, 203, 217, 240, 243

Lockpicking, 16
Logcat, 114, 115
Lápiz capacitivo, 88
Ls, 75, 77, 110, 116, 161, 162

M

MacBook, 57, 244
MacBook Pro, 57
Magic, archivo, 83
Malware, 17, 40, 41, 47, 54, 170, 171, 185, 204, 205, 206, 208, 209, 210, 211, 213
Man, 75, 186, 214
Manager.exe, 69
Mandiant, 20
Man-in-the middle, 214
Mapas, aplicación, 92
Máquina virtual, 59, 60
MBC, interfaz, 106
MCP (Multi Chip Package), 26
MD5, hash, 81, 82, 99, 226
Media Scanner, 179
Media Store, 180
Mensajes, 90, 96, 158, 177, 230, 233
Messaging, aplicación, 177
Metadatos, 29, 92, 122, 163, 179, 185, 187
MicroSD, 26, 27, 31, 32, 93, 97, 99, 101, 105, 106, 123, 124, 151, 159, 160
Microsoft, 35, 38, 49, 50, 51, 55, 62, 63, 64, 74, 76, 77, 97, 98
Microsoft Exchange, 38
Mkdir, 76, 99
Modo avión, 26, 132, 134, 135
Monitor, 28, 218
Motorola, 20, 21, 227
Mount, 99, 111, 123, 140, 153
MP3, 27, 50, 106
Máquina virtual, 44, 45, 54, 55, 56, 59, 60, 61, 72, 119, 145, 157, 171, 192, 200, 240
MS-DOS, 50, 62, 75, 76, 81, 183
Msmsms.db, 177
MS-Windows, 33, 53, 63, 108, 144, 148, 152, 190, 204, 217, 228
mTAN, código, 209, 214
MTD (Memory Technology Device), 140
Multicast DNS, 38
Mv, 81

N

NAND, 24, 26, 27, 32, 49, 50, 51, 52, 95, 97, 106, 111, 116, 122, 123, 125, 137, 138, 139, 140, 141, 144, 146, 153, 154, 155, 159, 160, 161, 180, 187, 203, 209, 236, 240
nandump, 147
NAT, 58
Navegador de archivos, 66, 69, 76, 92, 100
Ncurses, 104
Netstat, 112
Nokia, 21, 22
Norton, editor, 183
Novathor U8500, 24
NTFS, 49, 50, 82, 101
Nube, 17, 56, 159, 160, 253

O

O2 XDA, 127
Odin, 124, 125, 126, 137, 146, 148
Offset, 186
OpenBinder, 42
Open Handset Alliance, 35, 36, 191, 238
Open Handset Alliance (OHA), 35
Oracle, 54, 61, 62, 64, 65, 66
Orden judicial, 87, 91, 132, 251, 257
OS/2 Warp, 62
OSE, licencia, 62, 64, 65
OxyAgent, 230, 231

Oxygen Forensic, 222, 227, 228, 229, 230, 231, 246

P

Palm Pilot, 35
Password.key, 138
Pay-per-click, 213, 214
PC-Inspector, 104
PDF, 50, 88, 104, 182, 226, 229
Pelotita, virus, 183
Perl, 68
Permisos, 41, 46, 50, 51, 52, 66, 74, 75, 76, 78, 79, 81, 113, 117, 118, 119, 120, 121, 122, 160, 161, 166, 167, 168, 169, 172, 190, 191, 192, 193, 194, 195, 198, 199, 200, 201, 202, 203, 207, 210, 211, 212, 214, 240, 259
Permisos nulos, vulnerabilidad, 200
Photorec, 104
Photoshop, 61, 135
PHP, 68
PID, 115, 193
PIN, 134, 136, 137, 138, 150, 249
Pipelining, canalización, 78, 79
Platform-tools, 71, 72, 109, 110, 113, 114, 115, 123, 124, 136, 152, 153, 154, 166
Playstore, 40
Play Store, 141, 191
PNG, 178
POP3, 78, 217, 220
Pornografía infantil, 204, 254
POSIX, 74, 193
Preferencias compartidas, 159
Previsualización, 88, 89
Proc, 49, 52, 110, 111, 139
Proveedores de contenido, 45, 48, 121, 213
Proxy, 43
Ps, 22, 23, 115
Psneuter, 123, 125, 137, 202
PUEL, licencia, 62, 64, 65
Punto de montaje, 27, 123, 140, 141, 161
Python, 165, 187

Q

Qemu, 60

R

RAM, 17, 24, 26, 31, 38, 43, 44, 47, 49, 52, 55, 58, 60, 73, 74, 95, 116, 131, 139, 144, 145, 159, 166, 190, 208, 225, 234, 241, 244, 249
RDP, 62
READ_SMS, permiso, 214
REBOOT, permiso, 201
RECEIVE_SMS, permiso, 214
Recovery, 125, 126, 127, 138, 139, 143, 144, 145, 146, 147, 149, 150, 172, 209, 215, 234, 257
Redireccionamiento, 78, 79, 109, 154
ReiserFS, 50, 51, 101
Release, 37, 39
Reloj, aplicación, 91
Repackaging, 207
RFC 3227, 256
RISC, 24
Rmdir, 76
ROM, 33, 118, 119, 127, 136, 137, 143, 144, 147, 148, 155, 198, 238
Root, 23, 66, 79, 80, 98, 110, 117, 118, 119, 121, 123, 125, 126, 147, 152, 194
Rootfs, 52, 111, 139
Rooting, 52, 95, 117, 118, 119, 120, 121, 124, 125, 126, 127, 136, 161, 198, 209, 210, 234, 238, 239
R-Studio, 104
Rubin, 34
Runtime, 44, 53, 169

S

Salida standard, 78
SAMBA, 217
Samsung, 21, 24, 25, 26, 28, 31, 34, 68, 70, 90, 106, 109, 125, 137, 141, 146, 148, 166, 187, 215, 216, 227, 242, 243
Samsung Galaxy Ace 2, 22
Samsung Galaxy S3, 24
Sandbox, 172, 192, 193
Sandboxing, 42
SAX, 165, 166
Scalpel, 105, 154, 187
Scripts, inicio Linux, 81, 115, 145, 187, 199, 208
SDK, 35, 39, 46, 53, 54, 67, 68, 69, 70, 71, 72, 73, 106, 109, 110, 112, 122, 136, 137, 152, 209, 233, 236, 243, 244
SDRAM, 26
SELECT, 115
Service Pack, 63
Services, 48
Servicios, 15, 25, 28, 29, 31, 35, 36, 39, 42, 43, 45, 48, 66, 74, 108, 115, 117, 143, 159, 160, 164, 166, 168, 173, 175, 181, 183, 195, 199, 203, 204, 209, 211, 217, 223, 239, 251, 253
SHA 256, 79
Shared Preferences, 159
SharedUserId, 193
Shell inverso, 203
SIM, 26, 132, 134, 198, 225, 226, 230, 233, 244
Sistema Extendido de Archivos (ext), 51
Sistemas de archivos, 26, 27, 49, 50, 51, 52, 74, 100, 101, 111, 142, 145, 185, 223, 237, 241, 242
Skype, 227
SMART, 98, 104
SMS, 22, 38, 41, 85, 86, 90, 114, 115, 119, 122, 130, 131, 132, 141, 158, 168, 172, 177, 203, 206, 207, 209, 211, 212, 214, 225, 230, 233
Smudge attack, 135
Sniffer, 217
Software maligno, 52, 54, 82, 204, 206, 213
Solaris, 57, 62
Sony Xperia, 16, 68
SPL, Second Program Loader, 144, 145
SQLite, 29, 48, 115, 122, 130, 159, 162, 163, 172, 173, 174, 176, 178, 180, 182, 183, 186, 222, 231, 233, 237
Sqliteman, 173, 174, 175, 176
SSL, 217
Strings, 130, 157, 180, 183, 186
Stuxnet, 19
Sudo, 65, 66, 71, 79, 80, 82, 98, 99, 100, 104, 173
Suggestions.db, 182
SuperOneClick.zip, 122
SuperUser, 120, 121, 125
Superuser.apk, 123
Superusuario, 66, 69, 79, 95, 98, 116, 117, 118, 119, 120, 121, 122, 124, 137, 161, 162, 191, 194, 203, 209, 234
Su-v3, 123
Symbian, 222
Sysfs, 52, 111, 139

T

Tar, comando Linux, 71, 125, 151
TCP, 108, 241
Tcpdump, 77
Tecnosfera, 17
Telephony.db, 177

TestDisk, 104
Tethering, 38, 136
The Sleuth Kit, 51, 100, 242
ThinkPad, 25
Threads, 43
T-Mobile, 35
tmpfs, 52, 111, 112, 139, 140, 141
Toast, 201
Top, comando, 115
Tor, 253
Troyano, 112, 214
Tuneado, modding, 119
Twitter, 15, 41, 96, 158, 179, 227, 233, 251
TWRP, 148, 149

U
Ubuntu, 56, 59, 65, 80, 99, 100, 101, 104, 152, 173
Ubuntu Karmic, 65
Udev, 66, 109
UFED Phone Detective, 225
UFED Physical Analyzer, 225, 226, 227
UFED Touch, 222, 223, 224, 243, 244
UID, 193, 194, 195, 241
Unidad Central de Proceso, 24
Unix, 49, 51, 61, 74, 75, 78, 80, 81, 98, 108, 111, 114, 121, 157, 160, 185, 191, 193, 194, 217, 237
Update_su.zip, 125, 126
URI, 47, 201
URL, 201
USB, 26, 31, 32, 40, 50, 56, 58, 60, 62, 64, 65, 70, 78, 81, 82, 95, 97, 98, 105, 106, 107, 108, 109, 113, 123, 125, 130, 136, 137, 138, 139, 143, 146, 148, 150, 152, 161, 189, 198, 207, 208, 221, 225, 228, 231, 234, 243, 244, 253

V
VBoxNetwork, 64
VBoxUSB, 64, 65
Vboxusers, 66
VDI, 62
Víctima, 202, 203, 215, 250, 251
Viernes 13, virus, 183
VirtualBox, 54, 60, 61, 62, 63, 64, 65, 66, 67, 74
Virtualización, 28, 33, 38, 53, 54, 55, 56, 57, 58, 59, 60, 62, 192, 243
VirtualPC, 54
Virus, 52, 117, 183, 204, 222, 250
VLC, 82
Vmdk, 58
VMM (Monitor de Máquinas Virtuales), 55
VMware Server, 57
VMware Workstation, 57

W
W3W, Consorcio, 163
Warez, 204
WebKit, 43, 178, 199
Web semántica, 164
WebviewCache.db, 162, 163, 178
Webview.db, 162, 175, 176, 178
WEP, cifrado WiFi, 216, 217
WhatsApp, 15, 41, 85
WiFi, 16, 25, 26, 29, 42, 53, 92, 96, 117, 122, 131, 132, 189, 212, 215, 217, 226, 230, 246, 247, 251, 253, 254
Wifi Analyzer, 216
Wifipass, 95, 217
Windows 8, 57
Windows Media, 82
Windows Mobile, 22, 35, 127, 222
Wine, 54
Wireshark, 77, 217, 219
World Wide Web, 166

X

XDA Developers, 127
Xen, 54
XHTML, 164
Xing, 158, 220
XML, 29, 46, 77, 157, 160, 161, 162, 163, 164, 165, 166, 169, 172, 176, 181, 182, 183, 226, 234, 237

Y

YAFFS2, 38, 49, 142, 185, 187
Yahoo, 158, 179, 227
Youtube, 38, 95

Z

ZIP, archivo, 69, 123, 166, 173, 253
ZitMo, 209
Zygote, 145

Investigación forense de dispositivos móviles Android

La investigación forense de dispositivos móviles Android es un campo de reciente desarrollo en el que la disponibilidad de conocimientos técnicos, metodologías, herramientas y criterios de actuación están en proporción inversa al interés generado hacia los mismos. Esto se debe al vertiginoso desarrollo de este conocido sistema operativo de Google. *Smartphones*, tabletas, reproductores de medios e incluso electrodomésticos inteligentes Android plantean al investigador problemas difíciles de resolver que no se dan en el análisis forense tradicional de ordenadores de sobremesa y soportes de datos convencionales.

La presente obra trata temas de interés relacionados con el análisis forense en dispositivos Android, entre muchos otros:

- Tecnología de dispositivos móviles: hardware y software para plataformas Android.
- Empleo de máquinas virtuales en la investigación forense de dispositivos móviles Android.
- Adquisición forense basada en el empleo del SDK.
- *Rooting* y particiones *Recovery* alternativas.
- Análisis forense de bases de datos SQLite, archivos XML, aplicaciones, documentos, ejecutables .dex y sistemas de archivos ext4, FAT32 y YAFFS/YAFFS2.
- Modelo de seguridad Android, delincuencia informática móvil, espionaje industrial y aspectos criminológicos de la investigación.
- Soluciones comerciales utilizadas en la investigación forense de dispositivos móviles.
- Aplicación del análisis forense móvil en el contexto de la investigación convencional.

9 788499 645209

Ra-Ma®