Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO FINAL DE GRADO

TÍTULO DEL TFG: Estado del arte de las técnicas digitales forenses para dispositivos Android

TITULACIÓN: Grado en Ingeniería Telemática

AUTOR: Adrián Couceiro Mato

DIRECTOR: Juan Hernández Serrano

FECHA: 15 de Febrero de 2017

TÍTULO DEL TFG: Estado del arte de las técnicas digitales forenses para dispositivos Android

TITULACIÓN: Grado en Ingeniería Telemática

AUTOR: Adrián Couceiro Mato

DIRECTOR: Juan Hernández Serrano

FECHA: 15 de Febrero de 2017

Resumen

Hoy en día, tanto los ordenadores como otros dispositivos digitales se han convertido en omnipresentes en nuestra sociedad. Tanto empresas como particulares utilizan ordenadores en su día a día, ya sea para ocio o trabajo. Además, actualmente, todo el mundo dispone de un Smartphone con conexión a internet, el cual llevamos encima todo el día. Los teléfonos inteligentes han supuesto un gran avance tecnológico. Los podemos utilizar para comunicarnos con amigos y conocidos, comprobar el estado de nuestra cuenta bancaria, trabajar o incluso pagar la factura del supermercado.

Lo que se conoce por investigación digital forense, pretende dar solución a dos problemas que nos encontramos en este mundo tan tecnológico. Por una parte, se producen centenares de ataques cada día a dispositivos móviles y ordenadores. Es primordial poder determinar cuáles han sido las causas de un ataque, vectores de entrada y consecuencias para poder protegernos de cara a futuros ataques. Además, la mayoría de estos ataques son delitos tipificados y, por tanto, resulta de gran importancia poder recuperar los diferentes datos de los dispositivos en busca de pruebas que puedan ser presentadas como evidencia en un posible juicio.

Por otro lado, tanto empresas como particulares utilizan los dispositivos tecnológicos para realizar diferentes funciones. Es inevitable que se produzcan problemas como la pérdida de datos. El análisis forense pretende dar la posibilidad de recuperar información importante de nuestros dispositivos para analizarla posteriormente. De esta forma, las empresas pueden analizar posibles negligencias de sus empleados y, tanto empresas como particulares, tiene la capacidad de recuperar datos que, aparentemente, se habían perdido.

En este proyecto, se pretende dar a conocer en que consiste un análisis forense digital y que consideraciones debemos tener en cuenta al realizar dicho análisis. Se hablará también de aspectos legales relacionados con los análisis forenses digitales. Una vez conocido el proceso general, se orientará el proyecto a la investigación y estudio de diferentes técnicas y herramientas para realizar investigaciones forenses digitales en dispositivos Android. De esta forma, se estudiará cómo debemos realizar un análisis forense a un terminal Android.

leapTitle: Art state in digital forensics techniques for Android devices

Author: Adrián Couceiro Mato

Director: Juan Hernández Serrano

Date: February 15th 2017

Overview

Nowadays, both computers and other digital devices have become ubiquitous in our society. Both companies and individuals use computers in their day to day, whether for leisure or work. In addition, currently, everyone has a smartphone with internet connection, which we carry all day. Smartphones have been a technological leap. We can use them to communicate with friends and acquaintances, check the status of our bank account, work or even pay the supermarket bill.

What is known as digital forensics aims to solve two problems of this technological world. In one hand, hundreds of attacks occur every day on mobile devices and computers. It is essential to be able to determine what have been the causes of an attack, vectors of entry and consequences to have the capacity of protect us in future attacks. Furthermore, most of them are typified crimes and, for this reason, it is very important to be capable to retrieve the different data of the devices in search of evidences that can be presented in a possible judicial process.

On the other hand, both companies and individuals use the technological devices to perform different functions. Problems such as data loss are inevitable. Forensic analysis pretends to give the ability to recover important information of our devices to analyse later. In this way, companies can analyse possible malpractice of their employees and, as much companies as individuals, has the capacity to recover data that, apparently, had been lost.

In this project, it is intended to make known what is a digital forensics analysis and what considerations we must take into account when performing such analysis. Legal issues related to digital forensics will also be discussed. Once the general process is known, the project will be oriented to the investigation and study of different techniques and tools to carry out digital forensics in Android devices. Therefore, we will study how to perform an analysis for an Android terminal.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

En los últimos años, se ha notado un considerable incremento de los denominados delitos informáticos. Los delitos siguen siendo los mismos que los de antaño (estafas, acoso, robo de información…), pero al igual que en multitud de aspectos cuotidianos, los delincuentes también han modernizado sus técnicas aprovechándose de la evolución tecnológica. La aparición y evolución de los ciberdelitos, así como de las técnicas y herramientas utilizadas por los ciberdelincuentes, han generado la necesidad de protocolos e instrumentos para poder analizar, proteger o prevenir este tipo de delitos.

En relación con los ciberdelitos, una parte fundamental hoy en día es el denominado análisis forense digital. Este análisis se lleva a cabo cuándo las medidas de prevención y protección frente a delitos y problemas informáticos no han tenido éxito. Es una parte esencial, ya que gracias a ella podemos realizar varias acciones importantes, como por ejemplo, recuperar información de un dispositivo que ha quedado inutilizable o conseguir evidencias sobre el delito para un hipotético juicio.

El objetivo principal de este proyecto es dar a conocer cómo debe llevarse a cabo una investigación forense digital en un terminal móvil. Es importante saber cómo debemos realizar dicha investigación para que las pruebas obtenidas sean válidas en un posible juicio. En este caso, el proyecto se centrará en el análisis forense digital para dispositivos Android.

Por tanto, se abarcarán las diferentes partes del análisis forense digital. Se tendrán en cuenta las distintas consideraciones legales que debemos conocer, y se estudiarán las diferentes formas de conseguir y analizar las evidencias del terminal. Otro objetivo importante de este proyecto es analizar y conocer las diferentes herramientas que existen en la actualidad para realizar el análisis forense digital de dispositivos Android.

El resto de este documento se estructura de la siguiente manera:

Capítulo2. Introducción al análisis forense digital: En este capítulo se pretende dar a conocer cómo debe llevarse a cabo un análisis forense digital válido. Para ello, primero se definirán una serie de conceptos clave y se explicaran las etapas básicas de dicho análisis. Finalmente, se comentarán ciertos aspectos legales a tener en cuenta para que todas las evidencias encontradas sean válidas en un posible juicio.

Capítulo 3. Estudio teórico de Android: Para poder llevar a cabo el análisis práctico, primero debemos realizar un estudio teórico. Este capítulo se centrará en explicar, con el mayor detalle posible, los aspectos técnicos de los dispositivos Android que son cruciales para poder llevar a cabo el análisis, como por ejemplo, la jerarquía de archivos, los sistemas de seguridad en Android o cómo debe llevarse a cabo la extracción de la información. Finalmente, también se analizarán las diferentes técnicas y herramientas que

existen, hoy en día, para poder realizar análisis forenses de dispositivos Android.

Capítulo 4. Caso práctico: Análisis forense en Android: Una vez realizado el estudio teórico, este capítulo se centrará en poner en práctica lo aprendido hasta el momento para conseguir recuperar la mayor información posible de un dispositivo Android. Se explicará cómo debe realizarse un backup del dispositivo y cómo conseguir evidencias a partir de dicha copia. Nos centraremos, sobre todo, en conseguir información relacionada con registros de llamadas y mensajes, imágenes, bases de datos de Whatsapp y correos electrónicos. También se explicará cómo debe rootearse un dispositivo, ya que para conseguir la mayor parte de la información debemos disponer de acceso root al terminal.

Capítulo 5. Tropiezos y escollos durante la realización del proyecto: Este capítulo está pensado para explicar todos aquellos tropiezos y problemas con los que me he encontrado durante la realización del caso práctico, y de esta forma poder dotar de referencias para no sufrir las mismas complicaciones en estudios futuros.

Capítulo 6. Conclusiones y líneas futuras: Finalmente, el último capítulo del documento pretende explicar si se han alcanzado, o no, los objetivos fijados en un principio y explicar, en caso de no alcanzarse, cuáles han sido los motivos. También se pretende analizar hacía donde avanza la seguridad en dispositivos Android y sus repercusiones en cuanto al análisis forense digital.

2. INTRODUCCIÓN AL ANÁLISIS FORENSE DIGITAL

En este capítulo se hará una breve introducción al mundo del análisis forense digital. Se definirán conceptos clave para entender todo el proceso, y al mismo tiempo, se marcarán y se explicarán las diferentes fases por las que pasa, o debe pasar, el análisis. Estas fases abarcan desde el momento en el que se necesita un investigador hasta que éste entrega un informe final de resultados. No existe un modelo o protocolo estándar para el análisis forense digital, pero sí que es cierto que a lo largo de los años se han definido una serie de pautas a seguir. Para acabar, se explicarán las consideraciones legales que debemos tener en cuenta para que el análisis sea válido y admisible en un juicio.

2.1. Conceptos básicos

Antes de conocer las diferentes fases que conforman el análisis forense digital debemos repasar y explicar algunos conceptos básicos que son clave para entender el proceso de análisis.

Análisis forense digital

El término análisis forense digital, o *digital forensics*, hace referencia al método científico en el cual, mediante el uso de técnicas y herramientas especializadas, se adquieren, analizan, validan e interpretan datos e información proveniente de dispositivos digitales. En el ámbito de un proceso judicial, el objetivo principal del análisis forense digital es la obtención de posibles evidencias o pruebas digitales con el fin de demostrar o refutar una hipótesis ante los tribunales.

La norma *ISO/IEC 27037:2012 "Information technology — Security techniques — Guidelines for identification, collection, acquisition and preservation of digital evidence"* [1], publicada en el año 2012, tenía la intención de renovar las directrices recogidas en el *RFC 3227* (2002) [2], orientándose hacía los dispositivos más actuales. Esta nueva norma está orientada a dar un conjunto de buenas prácticas para el procedimiento de la actuación pericial en el campo de la adquisición de potenciales evidencias digitales.

Evidencia digital

La evidencia digital, o *digital evidence*, es toda aquella información, almacenada o transmitida digitalmente, que se considera valiosa para cualquiera de las dos partes de un juicio y que se puede utilizar para probar o refutar acusaciones. Para que una evidencia digital pueda ser aceptada, el tribunal debe determinar si es relevante, auténtica (no se ha alterado la evidencia) y válida. La norma *ISO/IEC 27037:2012* de la que hablábamos anteriormente, recoge un conjunto de buenas prácticas que son realmente útiles y necesarias para que una evidencia digital sea totalmente admisible y válida en un juicio.

Cadena de custodia

Se conoce como cadena de custodia al protocolo de actuación que debe seguirse durante el tiempo de vida de la evidencia digital, es decir, desde que se consigue la evidencia hasta que esta se destruye o deja de ser necesaria. El protocolo debe controlar cómo, dónde y quién ha obtenido la evidencia, qué acciones se han llevado a cabo con ella, quién ha tenido acceso a la misma durante todo su tiempo de vida y, finalmente, dónde se encuentra la evidencia en cada momento.

Es muy importante que se mantenga la cadena de custodia intacta durante todo el proceso judicial, ya que si dicha cadena se rompe, la evidencia deja de ser válida y admisible dentro del proceso. La cadena de custodia debe empezar el día que se consiguen las evidencias, siempre que sea posible, ante la fe pública (secretario judicial o notario), la cual debe quedarse una copia de toda la información obtenida para garantizar, de esta manera, el derecho a la defensa de la otra parte del juicio.

2.2. Fases del análisis forense digital

El hecho de no contar con un modelo estandarizado, provoca que cada investigador realice el análisis a su manera y, por tanto, los resultados pueden llegar a ser diferentes. Por este motivo se han realizado diferentes estudios en busca de un conjunto de pautas o buenas praxis para llevar a cabo el análisis.

Para este proyecto, se ha tomado como referencia el *Integrated Digital Forensic Process Model* de Michael Donovan Köhn publicado en Noviembre de 2012 [3]. El estudio de Michael Donovan pretende analizar los modelos existentes hasta la fecha para así confeccionar un nuevo método que englobe las partes más importantes de cada uno de ellos. Además, dicho modelo cumple con todos los requisitos de la ley para que las evidencias obtenidas sean válidas y admisibles en un juicio.

Tal y como se muestra en la Fig. 2.1, en este estudio se definen 5 fases diferentes por las que debe pasar todo análisis forense digital.

Fig. 2.1 Fases del análisis [3]

2.2.1. Preparación

El proceso se inicia con una primera fase de preparación. En esta fase, como se muestra en la Fig. 2.2, el investigador forense tiene como claro objetivo conseguir un coste operativo mínimo, es decir, maximizar la recopilación de evidencias

digitales creíbles y, a su vez, minimizar el coste de una respuesta a incidencias forenses. Se trata de una fase que, aunque no corresponde a un análisis propiamente dicho, sirve al investigador para prepararse y marcarse una estrategia a seguir al inicio del análisis.

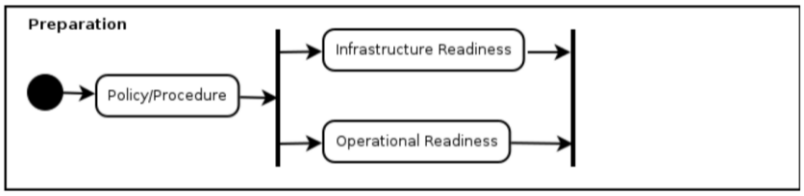


Fig. 2.2 Fase de preparación [3]

2.2.2. Incidencia

Se entiende por incidencia toda aquella acción realizada que compromete la confidencialidad, integridad y/o disponibilidad de un sistema de información o sistema informático. El objetivo principal de esta fase es detectar y confirmar o desmentir una posible incidencia.

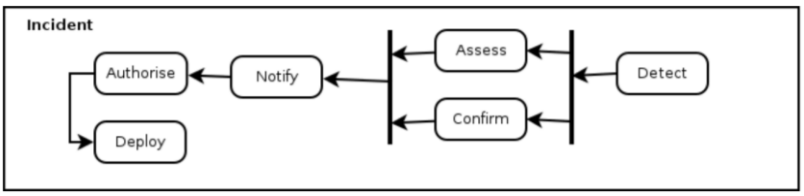


Fig. 2.3 Fase de incidencia [3]

Una vez se haya detectado la incidencia, ya sea de forma manual o automatizada, se debe comprobar si es real o se trata de un falso positivo. En el caso de confirmar la existencia de la incidencia, tal y como se observa en la Fig. 2.3, se debe notificar a los investigadores y autoridades para poder pasar a la siguiente fase, respuesta a la incidencia.

2.2.3. Respuesta a la incidencia

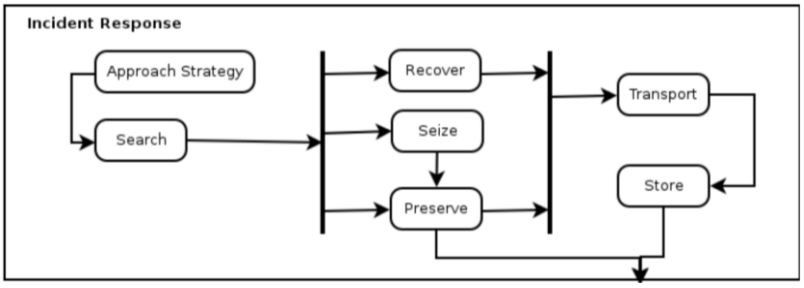


Fig. 2.4 Fase de respuesta a la incidencia [3]

La fase de respuesta a la incidencia, como vemos en la Fig. 2.4, se inicia justo en el momento en el que el primer investigador llega a la escena del crimen. En esta fase, el investigador debe analizar la escena con la que se encuentra y así definir una estrategia a seguir. Debe iniciar la cadena de custodia preservando todas las posibles evidencias y anotando todas las acciones que se lleven a cabo en la escena. Finalmente, una vez se encuentre presente la fe pública, se debe llevar a cabo una clonación de los dispositivos electrónicos que puedan ser considerados evidencias digitales. Dicha clonación, quedará en custodia de la fe pública y el investigador realizará, a partir de esta copia, una segunda clonación para un posterior análisis. Deberá obtenerse, también, el hash de la clonación que queda en disposición de la fe pública.

El hash es un algoritmo que nos permite obtener un resumen, de tamaño fijo, de una cierta información que pasamos como entrada. Una de las propiedades del hash, que lo hace tan interesante para estos casos, es que con una pequeña alteración de la información, el hash obtenido cambia completamente. Por tanto, mediante el hash se puede demostrar que no se han alterado las evidencias.

2.2.4. Investigación

La fase de investigación empieza cuando el investigador forense dispone de la clonación de los dispositivos electrónicos a analizar. A partir de la copia, el investigador debe conseguir el mayor número de evidencias posibles. Generalmente, se tratarán grandes cantidades de archivos e información, por esta razón, el investigador debe diferenciar la información importante para la investigación de aquella que no aporta datos de interés. Una vez diferenciada la información, el investigador debe documentar todas aquellas acciones que lleve a cabo para el tratamiento y el análisis de los datos para la obtención de evidencias. En la Fig. 2.5 se puede ver esta fase con mayor detalle.

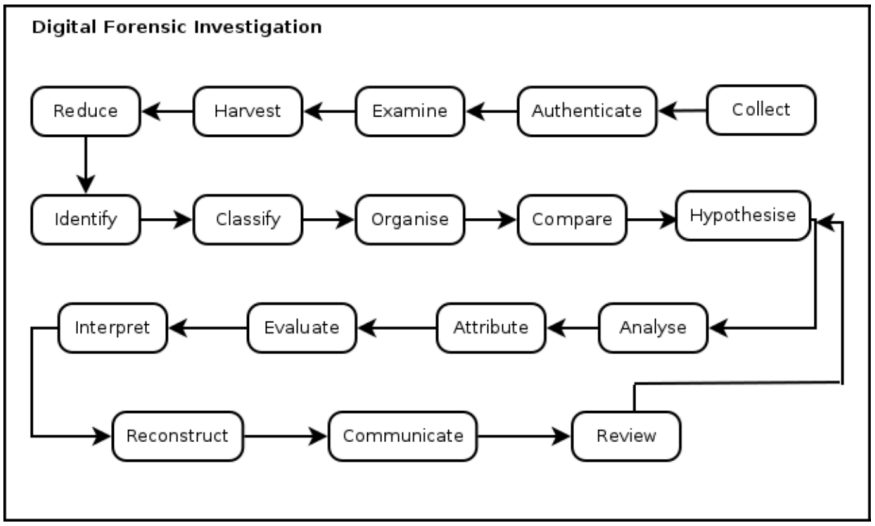


Fig. 2.5 Fase de investigación [3]

2.2.5. Presentación

Finalmente, el investigador forense debe realizar un informe para presentar todos los resultados obtenidos y las conclusiones a las que llega. Dicho informe debe estar dirigido a un público sin conocimientos técnicos, y por ello, toda evidencia o prueba que se incluya en él debe ir acompañada de una explicación e interpretación por parte del autor.

En el informe pericial deben incluirse tanto los resultados obtenidos, como todas aquellas acciones que se han llevado a cabo para llegar a los resultados. El informe debe permitir, por tanto, que otro investigador forense sea capaz de repetir todas y cada una de las acciones descritas y obtener los mismos resultados.

2.3. Adquisición y tratamiento de evidencias

En apartados anteriores ya se ha hecho referencia a la norma *ISO/IEC 27037:2012.* Su objetivo principal es el de establecer unas pautas de buena praxis para la obtención y análisis de evidencias digitales. Estas pautas se centran en los siguientes tipos de dispositivos y entornos:

- Equipos y medios de almacenamiento y dispositivos periféricos.
- Dispositivos móviles.
- Ordenadores y dispositivos conectados en red.
- Sistema de circuito cerrado de televisión digital.
- Sistemas críticos (alta disponibilidad).

La norma se basa en una serie de principios básicos que deben tenerse en cuenta para poder asegurar la integridad de la evidencia digital. Siguiendo estas pautas nos aseguramos que tanto la evidencia como el proceso llevado a cabo son totalmente válidos y admisibles por un tribunal judicial.

A continuación, podemos observar cuáles son los principios básicos que se detallan en la norma:

- **La adquisición:** Hace referencia al proceso por el cual se obtiene la información que se analizará en el futuro. Se debe recolectar y documentar aquellos dispositivos que puedan contener la evidencia y realizar la adquisición y la copia de toda la información de dichos dispositivos.

- **La preservación:** Debe preservarse la evidencia para poder garantizar su originalidad e integridad, es decir, debemos mantener la cadena de custodia para que pueda ser admisible en un proceso judicial.

- **Aplicación de métodos:** La evidencia debe ser adquirida, tratada y analizada aplicando herramientas y técnicas lo menos intrusivas posible, intentando preservar la originalidad de la prueba. También debemos

realizar, en la medida de lo posible, diferentes copias de las pruebas para trabajar sobre ellas y jamás sobre la original.

- **Proceso auditable, defendible y reproducible:** Todas las herramientas y acciones utilizadas deben haber sido contrastadas y validadas por las buenas prácticas profesionales. Debemos evitar, siempre que sea posible, utilizar herramientas propias, y en el caso de utilizarlas, debemos proporcionar junto con el informe final la herramienta utilizada, así como diferentes pruebas de su funcionamiento y resultados obtenidos. Tenemos que asegurarnos de que cualquier otro investigador forense sea capaz de reproducir el análisis forense digital que nosotros hemos llevado a cabo. Por tanto, debemos documentar con el mayor detalle posible todas las acciones que se realizan durante el proceso, las herramientas utilizadas, los resultados y las conclusiones obtenidas.

2.4. Consideraciones legales

Existen multitud de aspectos legales que debemos tener en cuenta cuando realizamos un análisis forense digital. Nos encontramos, por ejemplo, con el derecho a la intimidad y los derechos fundamentales de las personas investigadas. Por este motivo, cuando buscamos evidencias en un buzón de correo electrónico, por ejemplo, no podemos leer todo el buzón, sino que debemos realizar lo que se conoce como búsquedas ciegas, es decir, buscar mensajes concretos a partir de palabras clave.

Otro punto muy importante, al cual ya se ha hecho referencia en varias ocasiones, es la cadena de custodia y la integridad de las evidencias digitales. En el contexto de un proceso judicial, es imperativo mantener la cadena de custodia para asegurar la originalidad e integridad de las evidencias conseguidas. Por ello, se deben utilizar métodos de adquisición y análisis que sean lo menos intrusivos posibles para no alterar las evidencias. Es necesaria también, la documentación de quien está en posesión de la evidencia digital en todo momento para poder preservar la cadena de custodia.

La Ley Orgánica de Protección de Datos de Carácter Personal (LOPD) [4], Ley Orgánica 15/1999 que data del 13 de diciembre de 1999, establece las diferentes consideraciones legales que debemos tener en cuenta referentes al acceso y tratamiento de la información personal que encontremos en el dispositivo. No debemos llevar a cabo acciones que puedan considerarse delito o que violen cualquiera de los derechos fundamentales, detallados en la ley anteriormente mencionada, del sujeto investigado. Acciones como compartir información privada con gente ajena a la investigación, son catalogadas como delito.

Finalmente, en la Ley Orgánica 1/2015 del 30 de marzo [5], se detalla que no podemos llevar a cabo acciones ni utilizar herramientas informáticas que puedan comprometer las potenciales evidencias que estamos buscando. Estos hechos son considerados delito y en consecuencia, provocan que todo el análisis forense digital quede totalmente invalidado.

3. ESTUDIO TEÓRICO DE ANDROID

El presente capítulo pretende hacer un análisis del sistema operativo Android. Se estudiará dicho sistema, haciendo especial hincapié en aquellas partes que mantengan relación directa con el análisis forense digital.

Android es un sistema operativo, basado en el kernel de Linux, que fue diseñado principalmente para dispositivos móviles como los teléfonos inteligentes. Originalmente, pertenecía a la compañía Android Inc., actualmente, Google es la responsable de Android dado que en 2005 compró la compañía.

Hoy en día, Android se encuentra presente en teléfonos, tablets, relojes, televisores e incluso vehículos. El hecho de que se encuentre en la gran mayoría de dispositivos del mercado, que sea un proyecto de código abierto y que se base en el kernel de Linux, han provocado que la gran mayoría de ciberdelincuentes se interesen e intenten atacar los dispositivos Android. A continuación, veremos cómo funciona Android, estudiando su arquitectura, seguridad, sistema de fichero, etc., con la intención de buscar las mejores soluciones y la forma más adecuada de conseguir toda la información posible en un análisis forense digital.

3.1. Arquitectura

Tal y como se puede observar en la Fig. 3.1, Android se encuentra construido sobre una arquitectura formada por 4 capas diferentes. Estas capas están relacionadas entre ellas, ya que cada capa utiliza servicios de las capas anteriores y ofrece los suyos mismos a las capas superiores.

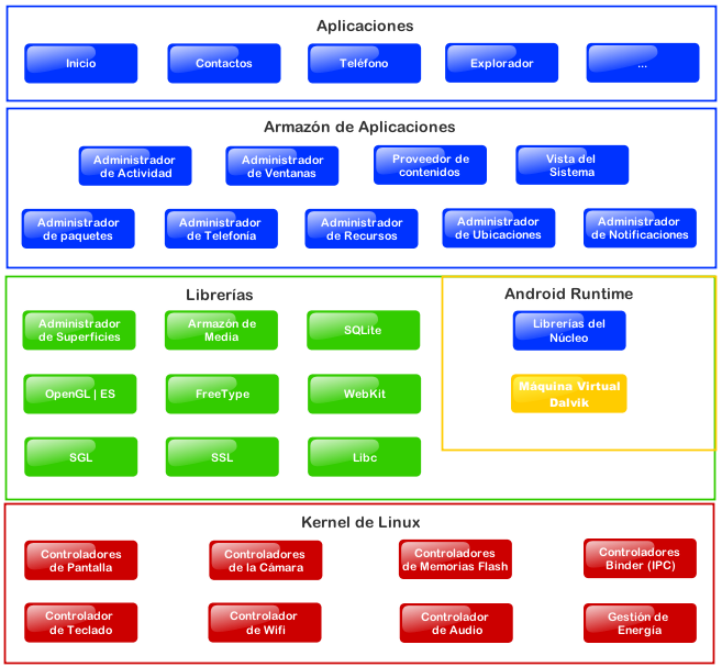


Fig. 3.1 Esquema de la arquitectura de Android

La capa básica y a partir de la cual se construye toda la arquitectura, es el Kernel de Linux, modificado y adaptado para entornos móviles. Android decidió basarse en Linux debido a la excelente portabilidad, seguridad y flexibilidad que este proporciona.

La capa de librerías constituye el segundo nivel de la arquitectura. En este nivel podemos encontrarnos tanto las librerías nativas como los controladores de tiempo de ejecución (Android Runtime). Dentro de esta capa, se distinguen las siguientes partes:

- **Librerías nativas:** Podemos ver diferentes librerías de código abierto como OpenGL para el renderizado de gráficos 3D, SQLite para la gestión de la base de datos, WebKit que permite el renderizado de los navegadores o SSL que proporciona servicios de cifrado para una conexión segura.

- **Hardware Abstraction Layer (HAL):** Se trata del componente esencial para que Android pueda ejecutarse en cualquier dispositivo móvil, independientemente de su hardware. Actúa como una arquitectura genérica para todos los posibles tipos de hardware del mercado. El HAL permite a los fabricantes que ajusten ciertos parámetros y características para que Android sea funcional en su tecnología.

- **Demonios (Daemons):** Son unos procesos que se ejecutan para ayudar a algún servicio del sistema. En el caso de Android, existen algunos daemons interesantes como *installd* que se ejecuta para administrar todo el proceso durante la instalación de aplicaciones. También nos encontramos con el daemon *adbd* que es el encargado de administrar el proceso del Android Debug Bridge (que veremos en detalle en apartados siguientes).

- **Consola:** Como en la mayoría de sistemas operativos, Android también permite la utilización de línea de comandos para explorar el sistema o ejecutar ciertos procesos.

- **Android Runtime:** Se basa en la misma idea que una máquina virtual de Java. Debido a las limitaciones que presentaban los dispositivos móviles (poca memoria y procesador), Android decidió crear Dalvik. Para la optimización de recursos, Dalvik ejecuta ficheros .dex (Dalvik Executables) que están diseñados para ahorrar memoria. Otro punto a favor, es que se basa en registros, es decir, cada aplicación utiliza su propio proceso Linux con su propia instancia de Dalvik. Con Android 5.0, Dalvik dejaría paso a ART que permite reducir el tiempo de ejecución en un 33%.

Después de la capa de librerías, nos encontramos con el framework o entorno de aplicaciones. Esta capa es la más interesante para los desarrolladores, ya que es la que proporciona todas las librerías de Java y los servicios utilizados por las aplicaciones. Podemos observar, por ejemplo, el administrador de notificaciones, servicio que permite a las aplicaciones mostrar alertas en la

pantalla del teléfono. Existen otros servicios importantes como el administrador de ventanas o el administrador de actividades. Cualquier desarrollador que realice una aplicación para Android, utilizará los servicios y librerías de esta capa para optimizar y reutilizar recursos.

Finalmente, llegamos a la capa de aplicaciones. Esta capa está formada por todas las aplicaciones Android instaladas en el dispositivo. Por medidas de seguridad y optimización, dichas aplicaciones deben ejecutarse en la máquina virtual Dalvik o ART. Esta capa utiliza todas las librerías, APIs y servicios de las capas explicadas anteriormente.

3.2. Jerarquía de archivos

Como ya se ha visto, Android es un sistema operativo basado en Linux, y por ello, la estructura de archivos es similar a la de los sistemas Unix. Los archivos se estructuran en secciones separadas, las cuales disponen de unos permisos de lectura y escritura. Por tanto, la accesibilidad a los archivos se determina teniendo en cuenta los permisos de las secciones, los directorios y finalmente, los archivos. Es decir, un archivo con permiso de lectura y escritura que se encuentre en una sección de solo lectura, no podrá ser modificado, ya que el permiso de la sección es prioritario frente al del archivo.

Estas secciones, conocidas como particiones, son independientes entre sí lo que nos permite aislarlas y tratarlas de forma totalmente independiente. Cuando el sistema monta las particiones se le asignan un conjunto de opciones, como los permisos de acceso (lectura y escritura), sistema de ficheros que contendrá (ext4, f2fs), o si se permite la ejecución de archivos root. Estas opciones no pueden modificarse posteriormente. Existen 3 particiones básicas que se encuentran presentes en todos los modelos de Android (system, data y sys) y en función del modelo se añaden otras particiones diferentes.

En la partición ***system*** o partición del sistema, encontramos aquellas aplicaciones y configuraciones que el fabricante o proveedor de telefonía instala en los dispositivos antes de entregarlos. Esta partición se monta en modo solo lectura, razón por la cual, el usuario es capaz de leer la información y utilizar las aplicaciones pero sin poder eliminarlas ni acceder a dicha partición.

Dentro de esta partición nos encontramos con diferentes directorios importantes. El directorio ***app*** contiene todas las aplicaciones del sistema (contactos, mensajes, telefonía…). También encontramos el directorio ***etc,*** que como en Linux, contiene los diferentes archivos de configuración del sistema. Por último, los directorios ***bin*** y ***xbin*** contienen los scripts y binarios que el sistema necesita para realizar algunas tareas y para ejecutar una serie de comandos.

La partición ***sys*** es la partición del kernel de nuestro dispositivo. Contiene el propio kernel y diferentes librerías y módulos necesarios para el correcto funcionamiento del dispositivo. También podemos encontrar archivos y datos relacionados con los diferentes componentes del teléfono como, por ejemplo, la CPU.

Finalmente, la partición ***data*** es la partición de datos. En esta partición se guardan todas las aplicaciones instaladas por el usuario y los datos de las mismas. También se guardan en esta partición los datos de las aplicaciones del sistema. Los permisos para la partición son de lectura y escritura (para usuarios con privilegios root como el usuario system), lo que permite una modificación dinámica de sus datos.

En el directorio ***app*** se guardan todas aquellas aplicaciones que instala el usuario, mientras que en el directorio ***data*** se almacenan todos los datos de aplicaciones tanto de sistema como de usuario. Este último directorio dispone de permisos de lectura y escritura, aunque Android crea un sub-directorio para cada una de las aplicaciones al que asignará los permisos correspondientes en función del tipo de aplicación.

Para acabar, podemos encontrarnos con otras particiones como ***sd-ext*** que nos permite la instalación de algunas aplicaciones en la tarjeta de memoria. Otro ejemplo, es la partición ***efs*** presente en los dispositivos Samsung, que almacena los datos relacionados con el hardware de comunicaciones del dispositivo o el IMEI del terminal.

3.3. Aplicaciones

Una aplicación Android es una aplicación móvil desarrollada para usarse en dispositivos de la plataforma Android. Google, apoya la idea de que sean sus propios clientes los que desarrollen las diferentes aplicaciones y, por este motivo, ponen a disposición del usuario multitud de documentación y diferentes APIs que pueden utilizarse en las aplicaciones desarrolladas por los usuarios. Las aplicaciones Android suelen programarse en lenguaje Java, aunque en los últimos meses están cogiendo mucha fama las llamadas aplicaciones móviles híbridas, como por ejemplo, las desarrolladas con Ionic Framework.

Una vez escrita la aplicación, las herramientas de Android SDK (Software Development Kit) nos permiten compilar el código, junto con los diferentes archivos de recursos y datos en un archivo de almacenamiento llamado APK. Este archivo incluye todos los contenidos necesarios de una aplicación Android y es el archivo que los dispositivos utilizan para instalar las aplicaciones en el terminal.

Al instalar la aplicación en el terminal, cada una de estas aplicaciones se aloja en su propia zona de pruebas de seguridad:

- Cada aplicación representa un usuario diferente para el sistema operativo Android.

- El sistema, le asigna a cada aplicación un ID de usuario de Linux diferente y establece permisos para los diferentes archivos que forman parte de la aplicación de modo que solo el ID de usuario asignado a la aplicación pueda acceder a ellos.

- Cada proceso cuenta con su propio Entorno Virtual (EV), por lo que cada aplicación se ejecuta de forma aislada a otras aplicaciones.

- Generalmente, cada aplicación ejecuta su propio proceso de Linux. Android inicia el proceso cuando requiere la ejecución de alguno de los componentes de la aplicación y lo cierra cuando dicho proceso ya no es necesario.

Basándonos en estos hechos, vemos que Android ejecuta lo que se conoce como el “Principio de mínimo privilegio”. Éste consiste en que cada aplicación tan solo tiene acceso a aquellos componentes y recursos que necesita para desarrollar sus funciones y que no puede acceder a partes del sistema para las cuales no tiene permisos.

No obstante, se puede desarrollar aplicaciones que utilicen recursos y/o datos de otras aplicaciones o incluso del propio sistema. Para ello, se pueden tener aplicaciones con el mismo ID de Linux y por tanto, estas aplicaciones tendrán los mismos permisos de acceso a los recursos. También podemos crear aplicaciones que necesiten acceder a datos del sistema como la agenda de contactos, los mensajes de texto o la cámara del dispositivo. El usuario debe garantizar de manera explícita estos permisos.

Existen cuatro componentes esenciales en las aplicaciones Android. Cada uno de ellos tiene un fin específico y un ciclo de vida diferente que define la forma en la que se crea y se destruye el componente. Estos cuatro componentes son los siguientes:

- **Actividades**: Una actividad es el punto de entrada para la interacción con el usuario y representa una pantalla con una interfaz de usuario. Un ejemplo de actividad sería la pantalla de login o registro de una aplicación. Si bien es cierto que las diferentes actividades trabajan conjuntamente para proporcionar una experiencia de usuario óptima y consistente en la aplicación, cada una de ellas es independiente de las otras. Al ser independientes, permiten que otras aplicaciones puedan iniciar estas actividades, siempre y cuando, la aplicación lo permita. Por ejemplo, en una aplicación de correo electrónico, la cámara de nuestro dispositivo podría iniciar la actividad de enviar un nuevo mensaje de correo electrónico con la foto que hacemos gracias a la cámara.

- **Servicios:** Los servicios son componentes de la aplicación que se ejecutan en un segundo plano para realizar operaciones, por lo general, prolongadas. Un servicio no proporciona una interfaz de usuario. Las actividades pueden iniciar los servicios y conectarse a este para que el usuario pueda interactuar. Un claro ejemplo de servicio es el hecho de reproducir música en un segundo plano mientras realizamos otras operaciones con el dispositivo.

- **Receptores:** Se trata de un componente que responde a los anuncios de mensajes en todo el sistema. Algunos de estos mensajes pueden ser creados por el propio sistema, como mensajes de que la batería es baja

o que se ha apagado la pantalla del terminal. Las aplicaciones también pueden generar mensajes, como por ejemplo, un aviso para otras aplicaciones de que se ha realizado una descarga de datos y que estos están disponibles para usarse. Los mensajes no generan una interfaz de usuario, pero es cierto que a veces pueden crear notificaciones en la barra de estado para alertar al usuario de algún evento. Generalmente, los receptores de mensajes representan una puerta de enlace a otros componentes y están pensados para realizar una cantidad mínima de trabajo.

- **Proveedores:** Los proveedores de contenido son aquellos componentes encargados de gestionar un conjunto de recursos compartidos entre diferentes aplicaciones. A través del proveedor de contenido, las aplicaciones son capaces de acceder, por ejemplo, a la base de datos de contactos del terminal y poder leer estos datos e incluso, en caso de que le permita, modificarlos.

Una vez hemos visto todos los componentes que forman una aplicación Android, debemos explicar en qué consiste el fichero AndroidManifest.xml. Se trata de un fichero que encontramos en el contenedor APK, del que se habló anteriormente, y que se utiliza para instalar la aplicación en el dispositivo. En este fichero encontramos el nombre de la aplicación, versión, características de hardware y software requeridas o los diferentes permisos de acceso, entre otros. Es necesario definir los diferentes componentes vistos anteriormente dentro de este fichero para que la aplicación pueda inicializarlos.

Es necesario conocer el funcionamiento de las aplicaciones en Android para poder realizar un análisis forense detallado y conciso. Conociendo el funcionamiento general de las aplicaciones podemos determinar qué ficheros debemos analizar para poder buscar los diferentes datos que almacena la aplicación en el dispositivo o incluso para realizar un análisis en busca de malware.

3.4. Seguridad en Android

En este apartado se pretende dar una visión general de cómo implementa Android la Seguridad en sus dispositivos. Al tratarse de un sistema operativo con kernel de Linux, se trata de un sistema multi-usuario en el que cada aplicación cuenta con su propio entorno de seguridad. Ya hemos comentado que cada aplicación se ejecuta en su propio entorno virtual, de forma totalmente aislada a otras aplicaciones. Otro aspecto importante a tener en cuenta es que cada aplicación representa a un usuario diferente del sistema operativo. De esta forma, el sistema puede conceder permisos de acceso a recursos personalizados para cada aplicación.

Como Android es un sistema operativo basado en Linux, ofrece medidas de seguridad propias del mismo. Entre ellas podemos encontrar el hecho de que los usuarios solo pueden acceder a aquellos recursos y ficheros para los que dispongan de derechos de lectura, escritura o ejecución. Android también implementa el módulo de seguridad SELinux (Security-Enhanced Linux). Se trata

de un módulo de seguridad para el kernel de Linux que proporciona el mecanismo para soportar políticas de seguridad para el control de acceso. Su arquitectura está pensada para separar las decisiones de las aplicaciones de seguridad de las políticas de seguridad de las mismas.

3.4.1. Mecanismos de bloqueo del dispositivo

Otro aspecto fundamental de la seguridad en Android, que además tiene una implicación directa en el análisis forense, son los mecanismos de bloqueo de pantalla. Existen diferentes formas de bloquear la pantalla en los dispositivos Android, como por ejemplo: patrón de desbloqueo, contraseña, pin o huella dactilar. Estos mecanismos de protección dificultan, en gran medida, el análisis forense, ya que para poder realizar dicho análisis debemos, generalmente, tener acceso al terminal.

Se conocen diferentes formas de conseguir saltarse la pantalla de bloqueo del dispositivo, aunque normalmente se necesitan una serie de requisitos previos que en la mayoría de ocasiones no se cumplen. Algunos de estos requisitos pueden ser: tener el teléfono con acceso root, tener un recovery como TWRP instalado o tener la opción de depuración vía USB activada en el menú de opciones de desarrollador. Como podemos ver, son algunos requisitos que, generalmente, no se cumplirán debido a su componente técnica.

Sin embargo, tal y como se comenta en un artículo de El País del pasado 25 de Enero [6], un grupo de investigadores del Reino Unido y China han ideado un algoritmo con el que son capaces de desbloquear el 95% de móviles Android en menos de cinco minutos. Este grupo de investigadores han realizado un estudio en el que grabando a cientos de usuarios a una distancia de 2.5 metros mientras desbloqueaban su teléfono móvil con patrón de desbloqueo, son capaces de ejecutar su algoritmo y obtener el patrón de desbloqueo utilizado. Un dato revelador del artículo es que no por utilizar un patrón más complicado, es más difícil de averiguar, sino todo lo contrario, generalmente los patrones complejos son más fáciles de obtener.

3.4.2. SE, TEE y HCE

Uno de los grandes avances en cuanto a seguridad en Android, es la aparición del Secure Element (SE), el Trusted Execution Environment (TEE) y el Host-based Card Emulation (HCE).

SE consiste en una plataforma inviolable, generalmente un chip, capaz de alojar de forma segura diferentes aplicaciones y sus datos confidenciales. Apareció para poder proporcionar al dispositivo y al usuario un elemento para realizar operaciones criptográficas o relacionadas con la seguridad y de esta forma no comprometer al dispositivo con datos, normalmente sensibles, que pudieran ser recuperados teniendo acceso al terminal. Al tratarse de un elemento de hardware independiente, aunque tengamos acceso al dispositivo, no podemos recuperar los datos que se alojan en este chip.

El TEE se basa en una arquitectura de capas, ya que se trata de un área segura del procesador principal. Garantiza al usuario que los datos y operaciones que se realizan en este entorno seguro están totalmente aislados y, por tanto, protegen tanto su confidencialidad como su integridad. Se virtualiza un entorno aislado que nos proporciona una seguridad mayor que el propio sistema operativo y que nos ofrece más funcionalidades que el elemento seguro (SE).

Finalmente, el término Host-based Card Emulation (HCE), hace referencia a aquella arquitectura pensada fundamentalmente para realizar pagos seguros utilizando la tecnología NFC. Gracias a esta arquitectura, hoy en día somos capaces de pagar con nuestros dispositivos móviles en la mayoría de establecimientos. Permite a los dispositivos emular una tarjeta NFC utilizando la propia CPU del terminal. Tan solo el sistema operativo puede vincularse y comunicarse con la arquitectura HCE, por lo que nos proporciona una cierta seguridad. Además, la arquitectura HCE está pensada para que los datos sean transportados de forma segura desde nuestro terminal hasta el lector NFC.

3.4.3. Evolución de la seguridad en Android

A continuación, se analizarán las diferentes medidas de seguridad añadidas en las actualizaciones de Android. Para este estudio se ha tenido en cuenta la información publicada por los responsables de Android en su página web oficial [7]. Se analizarán en profundidad algunas de las diferentes medidas de seguridad que se añadieron tanto en la versión de Android 5.0 como en la 6.0, ya que considero que actualmente son las dos versiones de Android con más usuarios en todo el mundo. Finalmente, también se comentarán ciertos aspectos de seguridad que se introducirán con Android 7.0.

Android 5.0 (Lollipop):

- **Cifrado de forma predeterminada:** Los dispositivos que vienen con Android 5 de fábrica, cuentan con el cifrado de disco completo habilitado de forma predeterminada. Esta medida de seguridad nos permite mejorar la protección de los datos del dispositivo en caso de pérdida o robo del terminal. En caso de haber actualizado a esta versión de Android, debemos activar el cifrado predeterminado en la opción de seguridad del menú de ajustes.

- **Mejora en el cifrado de disco completo:** La contraseña de usuario se encuentra protegida contra ataques de fuerza bruta utilizando scrypt y, cuando está disponible, la clave se enlaza al almacén de claves de hardware para evitar ataques fuera del dispositivo. Además, tanto la clave de desbloqueo del dispositivo como la clave de desbloqueo de la pantalla no se envía nunca fuera del dispositivo ni se encuentran disponibles para ninguna aplicación.

- **Sandbox de Android reforzado con SELinux:** A partir de Android 5.0, se requiere SELinux en modo aplicación para todos los dominios. SELinux ayuda a aumentar el modelo de seguridad de control de acceso

discrecional (DAC). Esta nueva capa proporciona protección adicional contra posibles vulnerabilidades de seguridad en el dispositivo.

- **Bloqueo inteligente:** Se incorporan mecanismos que proporcionan mayor flexibilidad para desbloquear dispositivos. Algunos de estos mecanismos pueden permitir, por ejemplo, que los dispositivos se desbloqueen automáticamente cuando se encuentren cerca de otros dispositivos de confianza (a través de Bluetooth o NFC) o cuando sean utilizados por alguien con una “cara de confianza”.

- **Multiusuario, perfil restringido y modo de invitado:** Android añade los perfiles de invitado en sus dispositivos para permitir la utilización del terminal de forma rápida y sencilla pero protegiendo los datos y aplicaciones del usuario principal del terminal.

Android 6.0 (Marshmallow):

- **Permisos en tiempo de ejecución:** Las aplicaciones solicitan los permisos necesarios durante su ejecución en lugar de concederse en el momento de su instalación. Además, los usuarios son capaces de activar y desactivar permisos, tanto para aplicaciones de Android 6.0 como para aplicaciones anteriores.

- **Inicio verificado:** Se realizan un conjunto de comprobaciones criptográficas del software del sistema antes de su ejecución para asegurar que el teléfono se encuentra en un estado sano desde el cargador de arranque hasta el sistema operativo.

- **Seguridad de hardware aislada:** Una nueva capa de abstracción de hardware (HAL) se utiliza en diferentes servicios como la pantalla de bloqueo, el cifrado del dispositivo o la API de huellas dactilares, para proteger las claves frente a compromisos del kernel y/o ataques físicos locales.

- **Huellas dactilares:** Los dispositivos pueden desbloquearse con solo un toque. Además, esta nueva opción, permite a los desarrolladores el uso de las nuevas APIs para utilizar las huellas dactilares para bloquear o desbloquear claves de cifrado. Esta nueva medida de seguridad nos ofrece la posibilidad de usar medidas biométricas para desbloquear el terminal.

- **Control de acceso USB:** Los usuarios deben realizar una confirmación para poder permitir el acceso USB a archivos, almacenamiento u otras funcionalidades del dispositivo. El valor predeterminado al conectar vía USB un teléfono a dispositivos como ordenadores es de solo carga, por lo que si queremos acceder al almacenamiento del terminal, necesitamos una aprobación explícita del usuario.

Android 7.0 (Nougat):

En la nueva versión de Android aparecen nuevas medidas de seguridad importantes. Se cambia el cifrado de disco entero, que se implementaba hasta el momento, por el cifrado basado en archivos para proporcionar una mayor protección y mayor aislamiento de los usuarios y perfiles en un dispositivo.

Otra mejora importante es el inicio verificado. Se aplica estrictamente para evitar que los dispositivos comprometidos, tanto a nivel de hardware como de software, se inicien. Soporta la corrección de errores para mejorar la confiabilidad contra la corrupción de datos no maliciosos.

3.4.4. Samsung KNOX

Debemos hacer una mención especial a la solución de seguridad Samsung KNOX [8]. Se trata de una solución propietaria de la compañía Samsung que proporciona medidas de seguridad para proteger tanto hardware como software de los dispositivos y así ofrecer un nivel de seguridad integral al terminal.

Nos permite la separación de aplicaciones y datos en diferentes dominios según su nivel de confidencialidad. Implementa la arquitectura SE (Secure Element) conjuntamente con una arquitectura de procesador conocida como ARM TrustZone. TrustZone consta de tres componentes esenciales en cuanto a seguridad: depósito de claves de TIMA, protección de kernel en tiempo real y certificación. Además, en TrustZone existen dos mundos claramente diferenciados: el normal y el protegido. KNOX utiliza este mundo protegido para mantener seguros los datos confidenciales.

Para añadir mayor seguridad a los datos, nos encontramos con un sistema de cifrado de archivos que utiliza el algoritmo AES con una clave de 256 bits. KNOX cuenta con una tecnología de varias capas que se incorpora a nivel de hardware y software de los dispositivos Samsung. Se verifica constantemente la integridad del dispositivo y detecta cualquier manipulación para poder proporcionar una mayor protección de los datos.

Está pensando, fundamentalmente, para su utilización dentro del ámbito empresarial. Permite a las empresas ofrecer un entorno seguro dentro de los dispositivos de sus empleados para separar los datos personales de los empresariales. Además, ofrece la posibilidad de controlar estos datos de forma remota, y de esta manera, en caso de pérdida o robo del dispositivo, dejar totalmente inaccesibles e inutilizables los datos.

3.5. Android Debug Bridge (ADB)

Android Debug Bridge (ADB) es una herramienta de línea de comandos que nos permite comunicarnos con un dispositivo Android, a través de USB o Wi-Fi, o con una instancia de un emulador. Nos proporciona la capacidad de realizar diferentes acciones en los dispositivos como instalar o depurar aplicaciones y ejecutar comandos propios de sistemas Linux a través de una shell. Se trata de una herramienta cliente-servidor en el que podemos diferenciar tres partes:

- **Cliente:** Se ejecuta en nuestra máquina de desarrollo, generalmente, nuestro ordenador. Es la parte encargada de enviar los diferentes comandos que queremos ejecutar a nuestro dispositivo Android.

- **Demonio**: Se ejecuta como un proceso en segundo plano en nuestro dispositivo Android y es el encargado de ejecutar los comandos.

- **Servidor**: Administra la comunicación entre cliente y demonio y se suele ejecutar como proceso en segundo plano en nuestra máquina de desarrollo.

Ya se ha comentado que ADB nos permite realizar una gran cantidad de acciones, como instalar aplicaciones o ejecutar comandos de Linux en el dispositivo. A continuación, podremos ver algunos de los comandos más útiles y utilizados con esta herramienta:

- **adb devices:** Con este comando podemos listar las diferentes instancias de emulador o los dispositivos que tenemos conectados al servidor de ADB.

- **adb install *path_to_apk*:** Podemos instalar aplicaciones directamente desde nuestra máquina de desarrollo utilizando este comando y especificando la ruta donde se encuentra la apk que queremos instalar.

- **adb pull *remote local*:** Con este comando podemos realizar una transferencia de archivos y/o directorios desde el dispositivo Android hasta nuestra máquina de desarrollo.

- **adb push *local remote*:** Nos encontramos con el comando análogo al anterior. Se inicia una transferencia de archivos y/o directorios desde nuestra máquina hasta el dispositivo Android.

- **adb shell:** Este comando nos permite iniciar una shell en el dispositivo para poder ejecutar comandos propios de Linux como **ls** o **mount**, entre muchos otros.

- **adb backup:** Con la opción backup, podemos realizar una copia de seguridad lógica de toda la información y archivos del dispositivo. Se realizará la copia de seguridad de aquellos datos a los que tengamos acceso con nuestros permisos.

3.6. Extracción de información

Para poder realizar un análisis forense en condiciones, la extracción de información es una parte fundamental para la captación de evidencias. A continuación, se analizará tanto la adquisición lógica de información como la adquisición física.

Adquisición lógica: En la adquisición lógica de información se realiza una copia de los ficheros y de la información del dispositivo directamente desde la capa de

abstracción que proporciona el sistema de ficheros. Por tanto, no se accede físicamente al dispositivo y como consecuencia no tenemos acceso a la totalidad de la información. Además, al realizar una adquisición lógica de la información, el espacio “unallocated” del disco no se copia, imposibilitando así la recuperación de archivos eliminados.

Existen diferentes formas de realizar una adquisición lógica. Generalmente, las herramientas propias de cada compañía (Samsung, Sony…) para realizar copias de seguridad se basan en la adquisición lógica. Al ejecutar el comando **adb backup**, que vimos anteriormente, también se realiza una adquisición lógica de la información.

Adquisición física: También conocida como copia bit a bit, a diferencia de la adquisición lógica, nos permite replicar todo el sistema de ficheros disponible. Para poder realizar este tipo de adquisición de información, debemos contar con privilegios root en el dispositivo. Una de las formas más utilizadas para realizar una adquisición física es a través de la herramienta **dd** que ejecutamos a partir de **adb shell.**

Por una parte, su principal ventaja respecto a la adquisición lógica es que conseguimos una réplica idéntica del original, por lo que se preservan la totalidad de las potenciales evidencias del dispositivo. Además, este método nos permite la búsqueda de elementos eliminados, ya que al realizar una copia idéntica al original también se copia el espacio “unallocated”. No obstante, cuenta con desventajas evidentes. Se trata de un método mucho más lento que la adquisición lógica.

4. CASO PRÁCTICO: ANÁLISIS FORENSE EN ANDROID

Este capítulo constituye la parte más práctica del proyecto. En esta parte se realizará el análisis forense a dos dispositivos con versiones de Android diferentes. El dispositivo principal que analizaremos será el último modelo de la marca Samsung que viene con la última versión de Android disponible, la 6.0.1. Asimismo, también se analizará un terminal de la marca Sony con una versión de Android anterior como es la 5.1.1.

Con esto veremos, en caso de que existan, las diferencias existentes al tratar de realizar el análisis forense en los dos dispositivos. Queremos comprobar qué podemos recuperar en cada versión de Android para determinar posibles cambios de seguridad en las nuevas versiones. Además, como ya hemos visto, Samsung implementa el sistema KNOX en sus dispositivos desde la versión de Android 4.3 y por tanto, intentaremos descubrir qué tipo de barreras añade este sistema de seguridad.

4.1. Análisis del caso y adquisición de la información

El primer paso en un análisis forense digital es la preparación. Dentro de esta fase se incluye tanto la definición del caso y los objetivos como la creación de un plan de actuación. Debemos dejar lo más claro posible cuál es el objetivo principal del análisis y las razones por las cuales lo llevamos a cabo. Además, es necesario definir la metodología que se seguirá para realizar el análisis y conseguir recuperar el mayor número de pruebas. En esta fase de la investigación también se debe analizar en profundidad el dispositivo, decidir qué datos debemos recuperar y ver dónde se encuentran.

4.1.1. Estudio de la metodología adecuada al caso

El objetivo principal de este análisis es encontrar la mayor cantidad posible de pruebas de cara a un posible juicio. Para ello, como investigadores forenses, debemos decidir qué datos queremos obtener y descubrir dónde se encuentran. Generalmente, los datos más importantes que suelen considerarse pruebas potenciales son los siguientes:

- Registro de llamadas telefónicas y mensajes de texto.
- Agenda de contactos.
- Imágenes, vídeos y audios.
- Contenido de la aplicación de mensajería Whatsapp.
- Contenido de la aplicación de correo electrónico Gmail.

Por tanto, nuestro objetivo será conseguir todos estos datos de la forma menos intrusiva posible. Ahora, debemos analizar dónde podemos encontrar esta información para determinar la mejor forma de llegar a ellos sin que se nos pueda acusar de alterar las evidencias. A continuación, en la Tab. 4.1, podemos ver un listado de la localización de la información anteriormente comentada:

Tab. 4.1 Localización de la información más importante

FICHERO	UBICACIÓN
Registro de llamadas y mensajes de texto	data/data/com.android.providers.telephony
Agenda de contactos	data/data/com.android.providers.contacts
Imágenes y vídeos de la cámara	DCIM/Camera
Otras imágenes	Pictures
Otros vídeos	Movies
Audios	Music
BD de Whatsapp descifrada	data/data/com.whatsapp/databases
BD de Whatsapp cifrada	Whatsapp/Databases
Key file para descifrar la BD	data/data/com.whatsapp/files
Imágenes, vídeos y audios de Whatsapp	Whatsapp/Media
BD de Gmail	data/data/com.google.android.gm/databases

Analizando los diferentes directorios donde podemos encontrar la información, ya podemos observar algunos aspectos interesantes. Para obtener las imágenes, videos y audios no necesitamos tener rooteado el dispositivo, ya que solamente conectándolo a un ordenador ya se pueden recuperar estos datos. Por otra parte, para recuperar las bases de datos, tanto de Gmail como de Whatsapp, necesitamos ser usuario root en el dispositivo, y por tanto, debemos realizar el proceso de root de la manera menos intrusiva posible y documentando, paso por paso, el proceso realizado para que no se nos pueda acusar de alterar evidencias.

Finalmente, en cuanto a la obtención de datos relacionados con los contactos y el registro de llamadas y mensajes de texto, parece que también debemos ser usuario root en el terminal, ya que se almacenan en un directorio de la partición **/data** a la que solo podemos acceder siendo root. No obstante, al realizar una copia de seguridad del dispositivo, obtenemos los ficheros **calllog.ebk** y **contacts.spba** que son los registros de llamadas y la agenda de contactos respectivamente.

Una vez ya hemos decidido que datos del dispositivo queremos conseguir del dispositivo, debemos seleccionar la mejor forma de llegar a estos ficheros. El plan que se ha trazado y que debemos seguir para conseguir los datos del terminal es el siguiente:

1. Solicitar una autorización, ya sea al cliente o al juez del caso, para poder extraer toda la información del dispositivo para su posterior análisis.

2. Backup completo del terminal ante notario.

3. Como ya sabemos desde el principio que debemos ser usuario root para recuperar algunos ficheros importantes, podemos realizar el rooteo del dispositivo también ante notario, documentando todo el proceso.

4. Una vez seamos usuarios root, utilizar la herramienta **dd** para conseguir una imagen del dispositivo ante notario.

5. El notario debe quedarse una copia del backup inicial y otra copia de la imagen sacada con dd. De esta forma nos aseguramos la integridad de las pruebas.

6. Ya podemos ir a nuestro lugar de trabajo para conseguir las evidencias y analizarlas.

Como hemos visto, en este proceso se tiene muy en cuenta una tercera parte de confianza como es el notario. Ya se ha comentado en diferentes ocasiones que es especialmente importante no afectar a la integridad de las evidencias durante el proceso de extracción y análisis para que sean admisibles en un posible juicio.

4.1.2. Adquisición del backup ante notario

Uno de los primeros pasos que debemos realizar en nuestro análisis forense es conseguir un backup del dispositivo ante notario. Existen diferentes formas de realizar el backup del contenido, pero la más recomendable es utilizar la herramienta oficial de la marca del terminal analizado. En nuestro caso, como vamos a analizar un Samsung, utilizaremos la herramienta **Smart Switch** [9]. Al ejecutar la aplicación, podemos observar que se trata de una herramienta muy intuitiva y con pocas funcionalidades. Nos permite realizar la copia de seguridad del dispositivo, restaurar un backup creado anteriormente o hacer una sincronización con Microsoft Outlook.

Procedemos a realizar un primer backup del dispositivo (Fig. 4.1) antes de realizar el proceso para conseguir ser root en el terminal. Este backup nos servirá para devolver el dispositivo al estado inicial, en caso de necesidad, y además también servirá para obtener toda aquella información y ficheros en la que no necesitemos ser root.

Fig. 4.1 Copia de seguridad del dispositivo

Una vez realizada la copia de seguridad, veremos que ha guardado en nuestro ordenador un directorio con un nombre que sigue el siguiente esquema: **SM-G935F_YYYYMMDDHHMMSS**, es decir, el modelo del dispositivo seguido de un identificador único creado a través de la fecha exacta en la que se realizó la copia de seguridad. Dentro de este directorio encontramos diferentes carpetas con información importante y que necesitamos para analizar, como por ejemplo la carpeta **Photo** o **Video**. Además, también vemos los archivos **calllog.ebk** y **contacts.spba**, que hacen referencia a los registros de llamadas y la agenda de contactos del dispositivo móvil.

Para asegurar completamente la integridad de esta copia de seguridad, buscaremos el hash del backup con el algoritmo **Sha 256**. Para ello, primero comprimiremos el directorio resultante de la copia de seguridad y acto seguido buscaremos su hash. En la Fig. 4.2 podemos observar los comandos utilizados para la compresión y obtención del hash del backup:

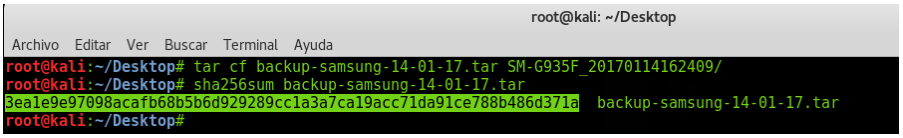


Fig. 4.2 Compresión y hash del backup

Una vez ya disponemos del backup correctamente realizado junto a su hash, ya podemos pasar al siguiente paso: rootear el dispositivo. Como ya hemos comentado, este proceso es especialmente crítico y debe realizarse con la mayor cautela posible, documentando absolutamente todo el progreso y, si es posible, realizándolo ante notario.

Lo primero que debemos hacer es analizar los diferentes procesos para rootear el dispositivo que existen. Suelen existir diferentes formas de realizar dicho proceso, siendo algunas de ellas menos intrusivas que las otras. Por este motivo, debemos seleccionar aquel proceso que consideremos menos intrusivo. En este caso, se ha seleccionado un rooteo que se lleva a cabo gracias a las herramientas **Odin** [10] y **CF-Auto-Root** [11].

CF-Auto-Root es una herramienta desarrollada por Chainfire que nos permite obtener acceso root al dispositivo de una forma fácil, rápida y segura sin necesidad de instalar ningún Kernel o Recovery personalizado en el terminal. Lo que hace esta herramienta es darnos acceso de SuperUsuario (su) en el dispositivo. Una de las ventajas de rootear el dispositivo siguiendo este método es que el sistema sigue siendo el original, por lo que el terminal continuará recibiendo las actualizaciones oficiales sin ningún problema. CF-Auto-Root instala en el dispositivo el Recovery Stock del modelo del dispositivo, junto con la aplicación SuperSu que nos da los privilegios de SuperUsuario. Odin es la herramienta que utlizaremos para flashear el CF-Auto-Root en nuestro dispositivo. A continuación, se detallará el proceso a seguir para conseguir con éxito el rooteo del dispositivo:

1. Descargamos y descomprimimos tanto CF-Auto-Root como Odin.

2. Como prerrequisito, debemos acceder a las opciones de desarrollador. Para ello, debemos presionar unas 7 o 8 veces seguidas en Número de Compilación dentro de Ajustes > Acerca del Dispositivo. Una vez activadas, debemos activar la depuración USB y el Desbloqueo OEM.

3. Apagamos el dispositivo y al encenderlo debemos entrar en modo Download (Fig. 4.3). Para acceder a este modo, debemos presionar de forma combinada las teclas **Encendido + Home + Bajar Volumen**. Cuando aparezca la pantalla para confirmar este modo, seguimos las instrucciones y presionamos **Subir Volumen**.

Fig. 4.3 Download Mode

4. A continuación, abrimos la herramienta Odin en nuestro ordenador y conectamos el teléfono al ordenador vía USB. Al conectarlo, veremos que Odin ya lo detecta. Presionamos en la opción **AP** y seleccionamos el archivo CF-Auto-Root que tiene **.tar.md5** como extensión. Una vez seleccionado, solo tenemos que darle a **Start** y esperar a que Odin termine con el proceso (Fig. 4.4).

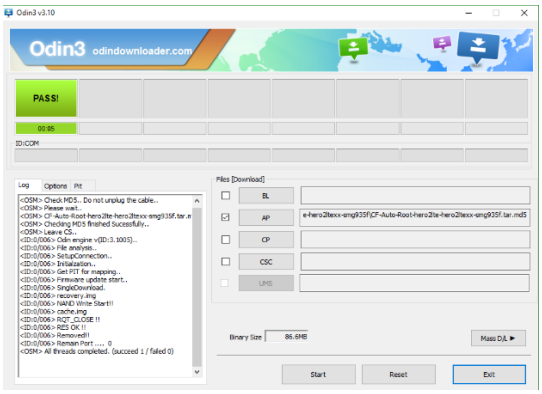


Fig. 4.4 Root con Odin

5. Una vez acabado el proceso, el teléfono se reiniciará solo y ya tendremos acceso root al dispositivo. Vemos que CF-Auto-Root ha instalado una apk llamada SuperSu, que no hace otra cosa que concedernos permisos de SuperUsuario. Podemos comprobarlo instalando la aplicación Root Checker en nuestro terminal y veremos que, efectivamente, somos root. Aprovechamos para instalar una aplicación llamada BusyBox que nos proporciona un conjunto de herramientas propias de Linux para poder utilizar en nuestro dispositivo.

Ahora que ya somos usuario root en el dispositivo, pasaremos a conseguir una imagen de la partición **data** del terminal, ya que es aquí donde se almacenan los datos de las diferentes aplicaciones del usuario. Para ello, utilizaremos la herramienta **dd** propia de Linux que ya viene integrada en el kernel de Android. Además, la transferiremos automáticamente a nuestro ordenador gracias a netcat. Esto lo hacemos para no tener que pasarla a una tarjeta de memoria SD y después pasarla al ordenador, dado que la imagen ocupa bastante espacio.

Para conseguir la imagen, debemos abrir una terminal en nuestro ordenador y ejecutar el comando **adb Shell** para conectarnos al dispositivo conectado. Una vez conectados, ejecutamos **su** para conseguir privilegios de SuperUsuario. Miramos las diferentes particiones del sistema utilizando el comando **mount** y, de esta forma, podremos ver el nombre de la que corresponde a la partición **/data,** que es la que nos interesa. Finalmente, una vez encontrada la partición, ejecutamos:

dd if=/dev/block/dm-0 bs=512 | busybox nc –l –p 9999

Este comando crea una imagen de la partición **/dev/block/dm-0** con un tamaño de bloque de 512 bytes. A su vez, activa el servicio de netcat y lo pone a la escucha en el puerto 9999. Para activar el servicio de netcat, utilizamos las herramientas que instalamos previamente con la aplicación BusyBox.

El siguiente paso es abrir otro terminal, que será el encargado de recuperar los datos a través del servicio netcat. Para ello debemos ejecutar dos comandos:

adb forward tcp:9999 tcp:9999

nc 127.0.0.1 9999 > android_adricouci.img

El primer comando permite el reenvío de puertos del ordenador a través de ADB. En el segundo comando, creamos una conexión netcat con el puerto 9999 de nuestra propia máquina (por eso realizamos el port forwarding con el primer comando) y guardamos los resultados en un fichero llamado android_adricouci.img.

En la Fig. 4.5, se puede observar una captura de pantalla donde se muestra todo el proceso que se ha explicado para conseguir la imagen del dispositivo.

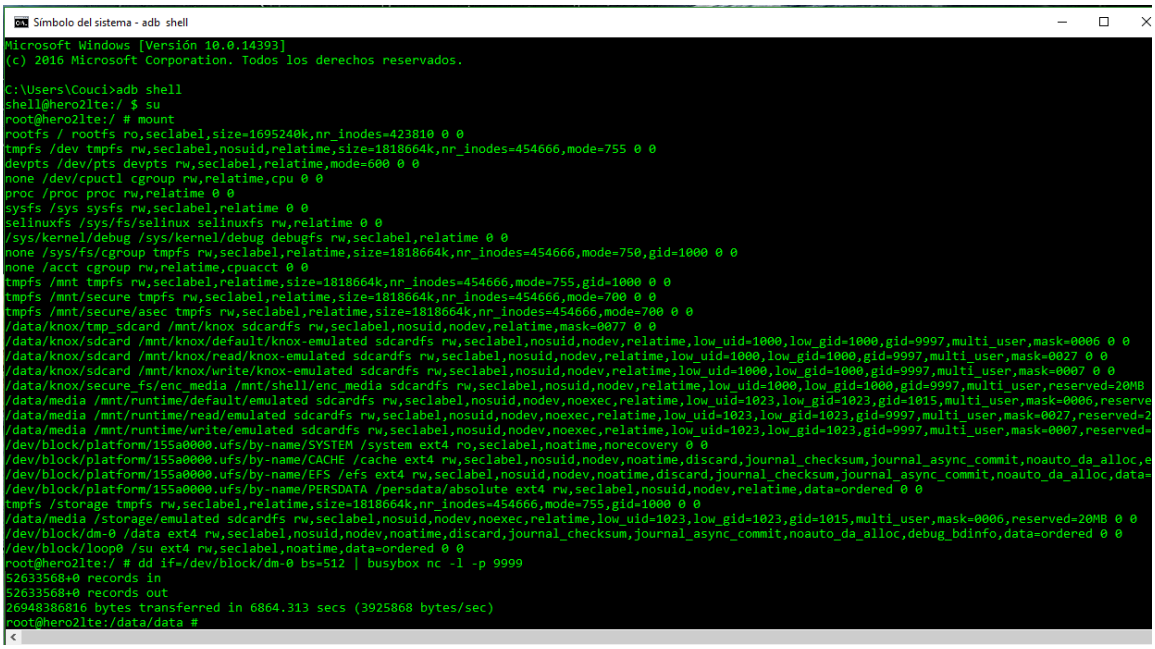


Fig. 4.5 Adquisición de una imagen de la partición data

Finalmente, solo nos queda conseguir el hash SHA 256 de esta imagen y depositar una copia de la imagen junto al hash ante notario. De esta forma, ya podemos irnos a nuestro lugar de trabajo con una copia del backup y otra de la imagen para analizarlo tranquilamente y obtener las evidencias.

4.2. Análisis de la información y adquisición de evidencias

En este nuevo apartado se explicará el proceso que se ha seguido para analizar la información, tanto de la copia de seguridad como de la imagen de la partición /data, y las diferentes formas de conseguir evidencias a partir de estas dos fuentes de información. Debemos tener muy claro qué información estamos buscando:

- Mensajes de texto, llamadas y contactos
- Imágenes, vídeos y audios
- Contenido de la aplicación de correo electrónico
- Contenido de la aplicación de mensajería instantánea Whatsapp

4.2.1. Mensajes de texto, llamadas y contactos

La primera información importante que queremos obtener es el registro de llamadas y mensajes de texto y la agenda de contactos. Existen diferentes formas de llegar a esta información. Entre las más comunes encontramos las provistas por software como **MOBILedit Forensic** [12] o **AF Logical OSE** [13].

MOBILedit Forensic nos da mucha información importante del dispositivo al conectarlo. Entre toda la información que proporciona, podemos encontrar el registro de llamadas (Fig. 4.6) y mensajes de texto y la agenda de contactos (Fig.

4.7). Además, nos permite exportar los resultados a un documento en formato XML o en formato Excel para analizarlo posteriormente.

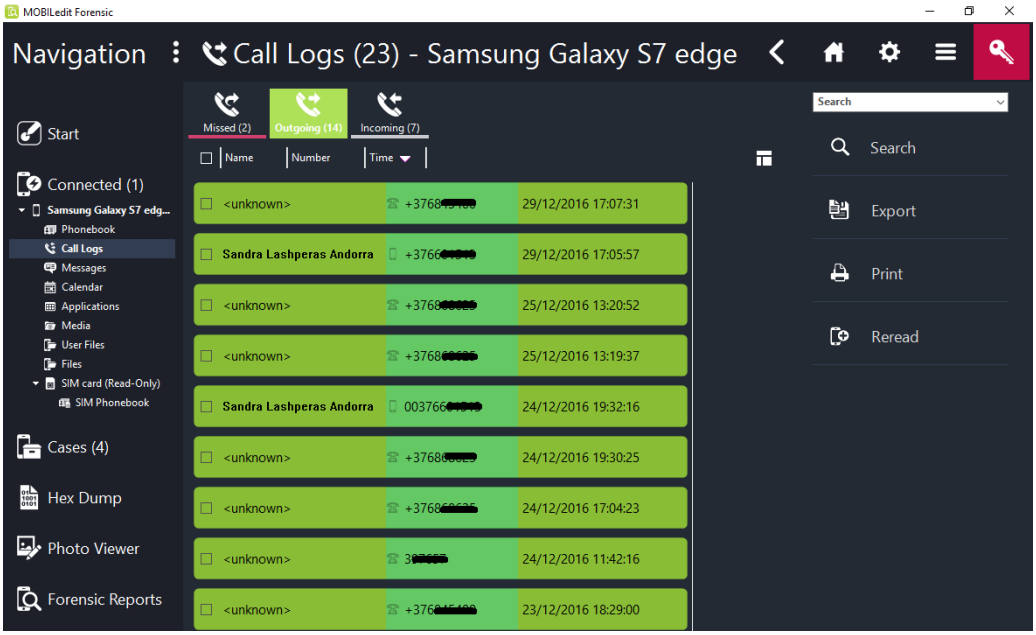


Fig. 4.6 Registro de llamadas con MOBILedit Forensics

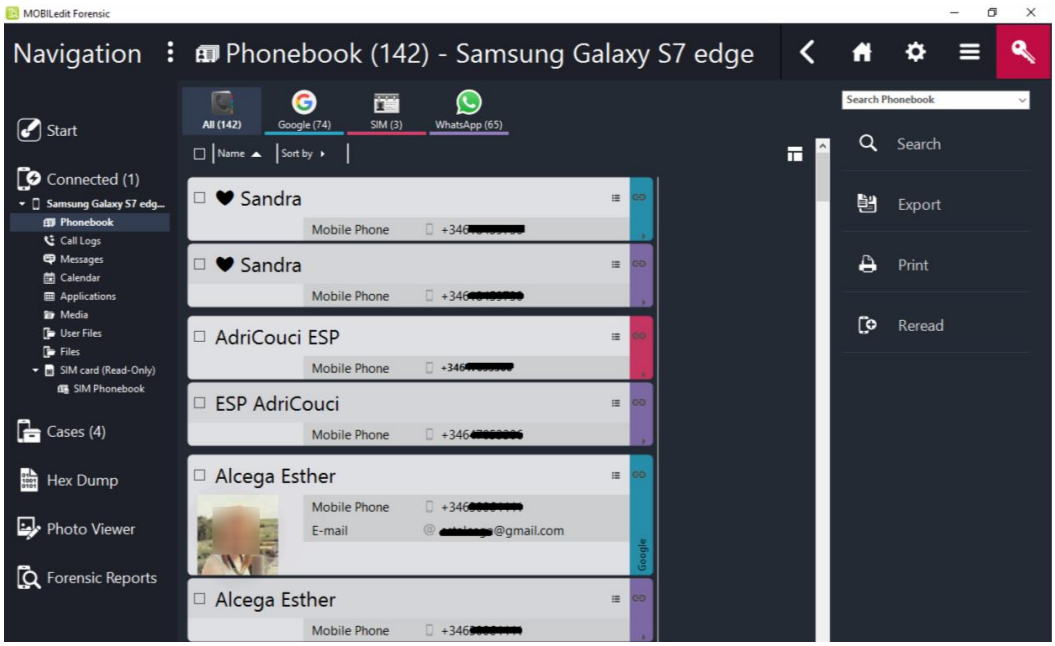


Fig. 4.7 Contactos con MOBILedit Forensics

Por otro lado, AF Logical OSE, instala una aplicación propia en el dispositivo que permite al usuario seleccionar los datos que quiere transferir al ordenador. Una vez seleccionados los datos, pulsamos capturar y estos se transfieren automáticamente a nuestro ordenador. En este caso, los datos se guardan en

nuestro ordenador en archivos **csv**. En la Fig. 4.8, se muestra una imagen de los datos obtenidos:

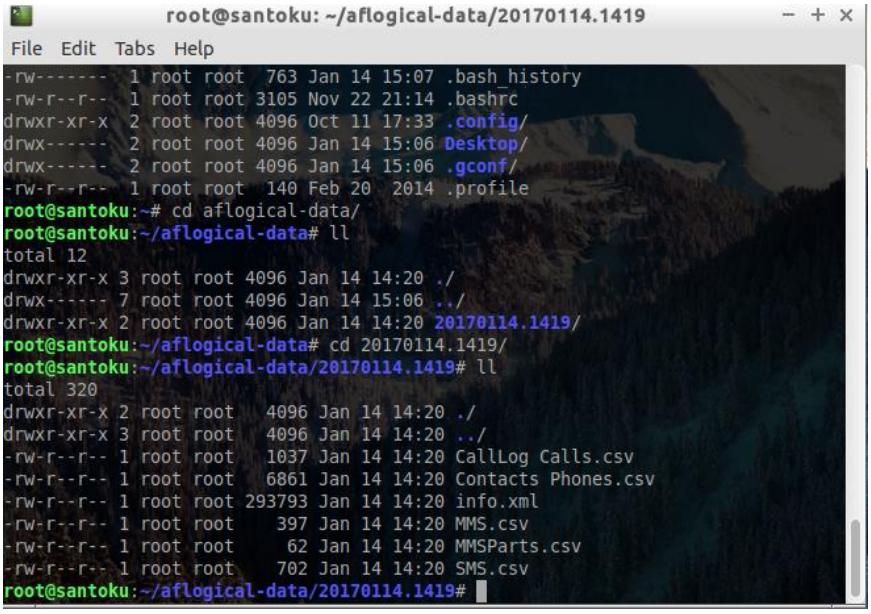


Fig. 4.8 Aplicación de AF Logical OSE

Como ya se ha comentado, existen una gran multitud de herramientas y procedimientos diferentes que nos permiten conseguir esta información. Otra opción de la que disponemos, consiste en analizar la copia de seguridad del dispositivo y visualizar los archivos **calllogs.ebk** y **contacts.spba**. Para visualizar dichos archivos necesitamos alguna herramienta que nos permita abrir estas extensiones, como por ejemplo **FileViewPro** [14].

4.2.2. Imágenes, vídeos y audios

Ahora, pasamos a analizar la forma en que se adquiere el contenido multimedia del dispositivo, es decir, imágenes, vídeos y audios. Ya hemos comentado que este contenido se encuentra bastante repartido y que tenemos diferentes formas de conseguirlo. El contenido multimedia suele guardarse en directorios diferentes dependiendo de la aplicación de la que provenga. Además, debemos diferenciar la forma de llegar a este contenido, si lo buscamos directamente con el teléfono conectado o si lo hacemos a través de una copia de seguridad.

Empezaremos con las imágenes. Nos centraremos en aquellas imágenes que provienen de la propia cámara del dispositivo y de aquellas otras que provienen de la aplicación de mensajería Whatsapp. Como ya hemos comentado, podemos recuperar las imágenes con el dispositivo conectado o a través de la copia de seguridad que realizamos en la primera fase. En esta copia de seguridad, todas las imágenes se encuentran en el directorio **Photo**, diferenciando **DCIM** para las de la propia cámara o **Whatsapp** para aquellas que provengan de los chats de la aplicación. En las siguientes capturas se mostrarán las distintas imágenes que recuperamos del backup de la aplicación Whatsapp (Fig. 4.9) y las imágenes

provenientes de la cámara del dispositivo conseguidas cuando éste se encontraba conectado (Fig. 4.10):

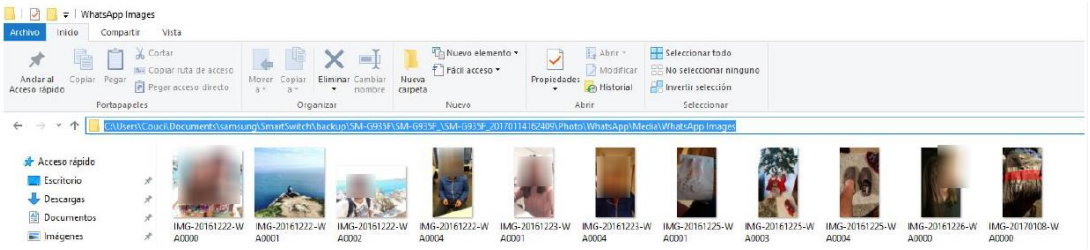


Fig. 4.9 Imágenes de Whatsapp recuperadas a través de la copia de seguridad

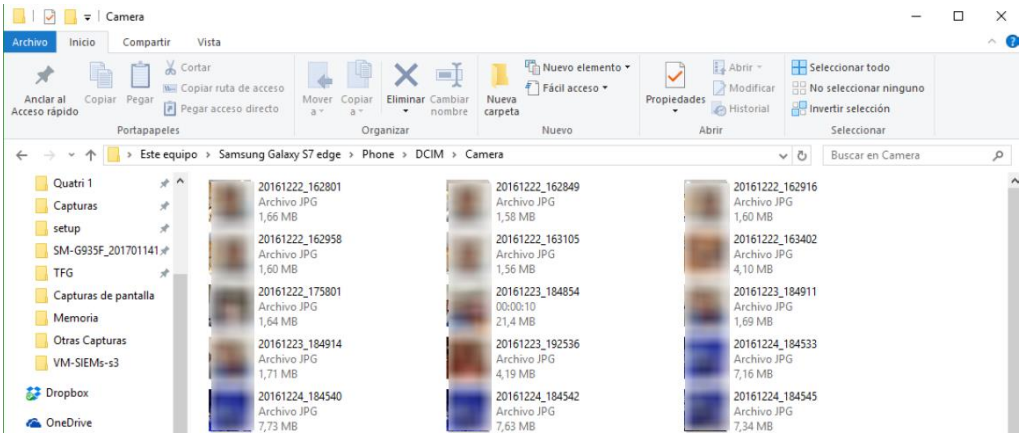


Fig. 4.10 Imágenes de la cámara recuperadas directamente del dispositivo

El caso de los vídeos es bastante similar al de las imágenes. Cuando analizamos el backup nos damos cuenta que todos aquellos vídeos que provienen directamente de la cámara del dispositivo se almacenan en **Video/DCIM**, mientras que aquellos vídeos que provienen de Whatsapp se almacenan en **Video/Whatsapp**. Por otro lado, con el terminal conectado, podemos acceder a los vídeos de la cámara a través del directorio **DCIM/** y a aquellos vídeos enviados o recibidos a través de Whatsapp en el directorio **Whatsapp/Media/Whatsapp Video**. En la Fig. 4.11, se podrá ver cómo se almacenan estos vídeos en la copia de seguridad:

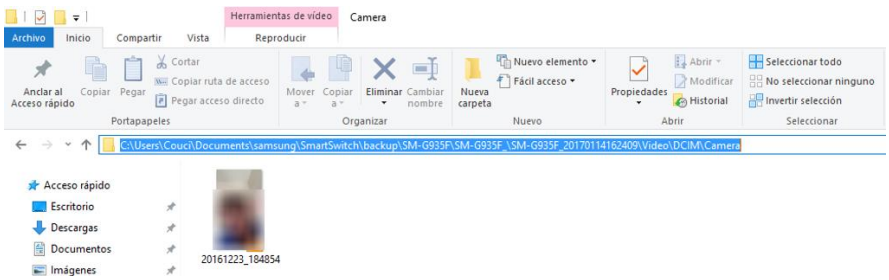


Fig. 4.11 Vídeos del dispositivo en la copia de seguridad

Finalmente, el caso de los audios es ligeramente diferente. Debemos diferenciar entre archivos de audio y notas de voz. Las notas de voz que enviemos por Whatsapp no se almacenan al realizar una copia de seguridad, y por esta razón, debemos recuperarlas con el dispositivo conectado. Se pueden encontrar en el directorio **Whatsapp/Media/Whatsapp Voice Notes** y agrupadas según las fecha. Por otra parte, los archivos de audio los encontramos en el directorio **Whatsapp/Media/Whatsapp Audio**, y al realizar una copia de seguridad se guardarán en el directorio **Music**. En la Fig. 4.12, se muestran las diferentes notas de voz de la aplicación Whatsapp:

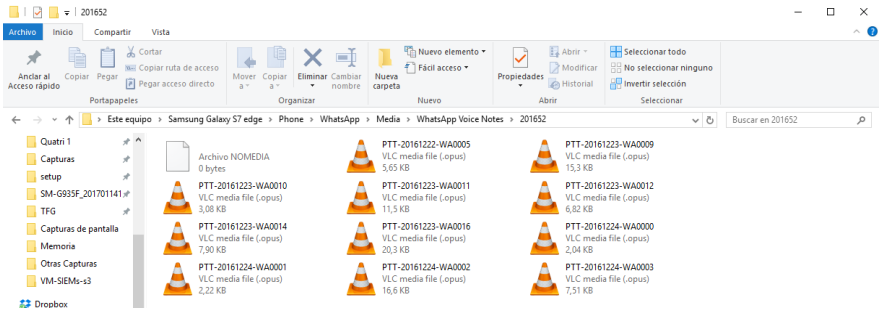


Fig. 4.12 Notas de voz de Whatsapp

Como hemos podido ver, existen diferentes formas de recuperar el contenido multimedia del dispositivo. Se trata de un contenido especialmente importante de cara a un proceso judicial, ya que suele considerarse una evidencia clara y admisible. Además, analizando los metadatos de las diferentes imágenes podemos obtener más datos como la fecha en que se realizó o la localización, si la aplicación de la cámara tiene el tag de geolocalización activado. Para analizar los metadatos de una imagen podemos hacer uso de prácticamente cualquier herramienta de visualización de imágenes. También existen herramientas dedicadas exclusivamente a esta tarea como, por ejemplo, **Exiftool** de Linux. En la Fig. 4.13, observamos cómo podemos leer los metadatos gracias a esta herramienta:

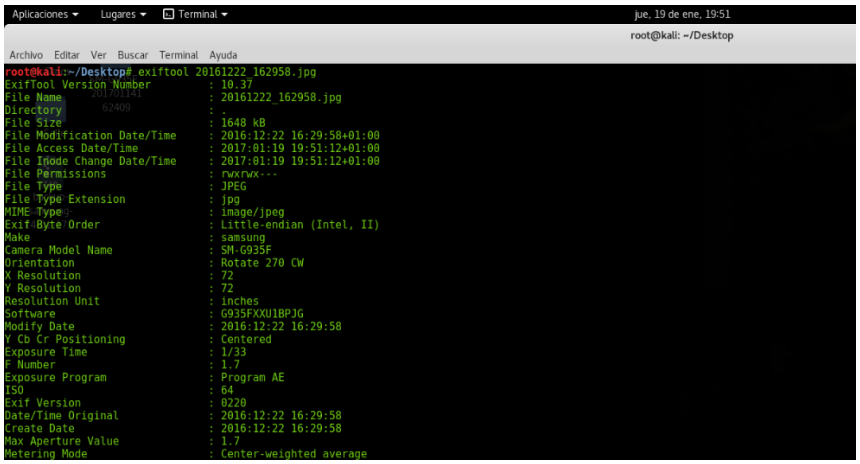


Fig. 4.13 Metadatos imagen

Finalmente, tenemos que hacer una mención especial a Autopsy, una de las herramientas más conocidas para análisis forense digital. Se trata de la plataforma forense y la interfaz gráfica de la herramienta **The Sleuth Kit**, librería y colección de herramientas de línea de comandos para el análisis e investigación de imágenes de disco. Al analizar la imagen de la partición /data con esta herramienta podemos recuperar algunos archivos que se borraron en el terminal, entre los cuales encontramos algunas fotografías, tal y como demuestra la Fig. 4.14.

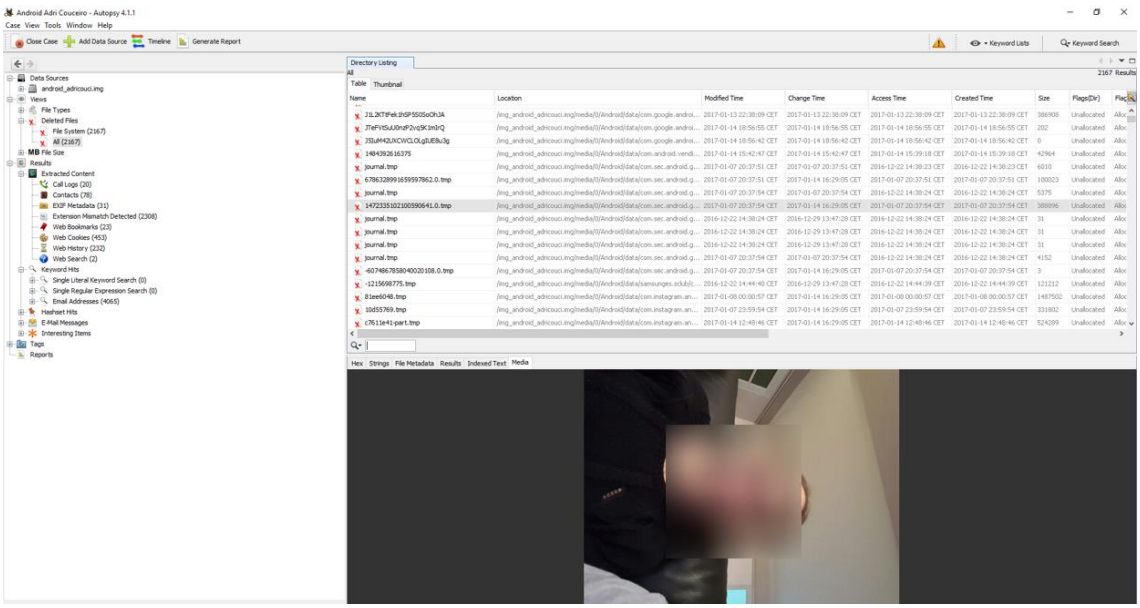


Fig. 4.14 Recuperación de imágenes eliminadas

4.2.3. Contenido de la aplicación Gmail

Otra parte importante del análisis forense es la recuperación del contenido de la aplicación de correo electrónico, en este caso Gmail. Gracias tanto a la agenda de contactos como a la propia base de datos de conversaciones, podemos recolectar multitud de evidencias. Se trata de un canal de comunicación que debe ser revisado, ya que podemos encontrar contenido y conversaciones útiles para nuestra investigación forense.

Lo primero que haremos, en este caso, es buscar los diferentes contactos de correo electrónico. Como ya hemos comentado, se trata de una información de mucha relevancia, y por este motivo, la herramienta que estamos utilizando para el análisis, Autopsy, ya se encarga automáticamente de analizar el diferente contenido de la imagen que nosotros le damos, para encontrar los contactos de correo electrónico. Para mostrarnos estas cuentas de correo electrónico, Autopsy analiza la información de diferentes bases de datos. Para empezar, analiza la base de datos **contacts2.db**, donde encontramos el contenido de la agenda de contactos del usuario. En esta base de datos suelen almacenarse también algunos contactos de la aplicación Gmail. También analiza las diferentes bases de datos que encuentra en la ruta **data/data/com.google.android.gm**

/databases donde se pueden encontrar todos los correos enviados y recibidos con la aplicación Gmail y, por tanto, las diferentes direcciones de correo electrónico. La Fig. 4.15 nos demuestra como Autopsy es capaz de recuperar una lista de direcciones de correo electrónico:

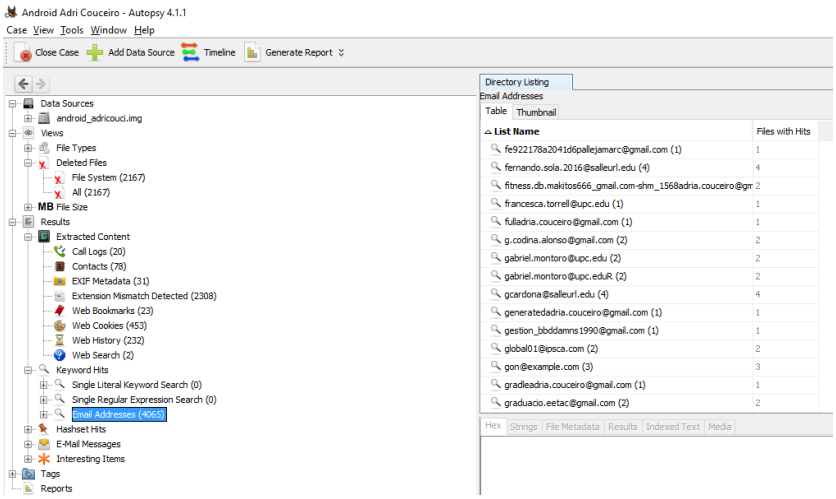


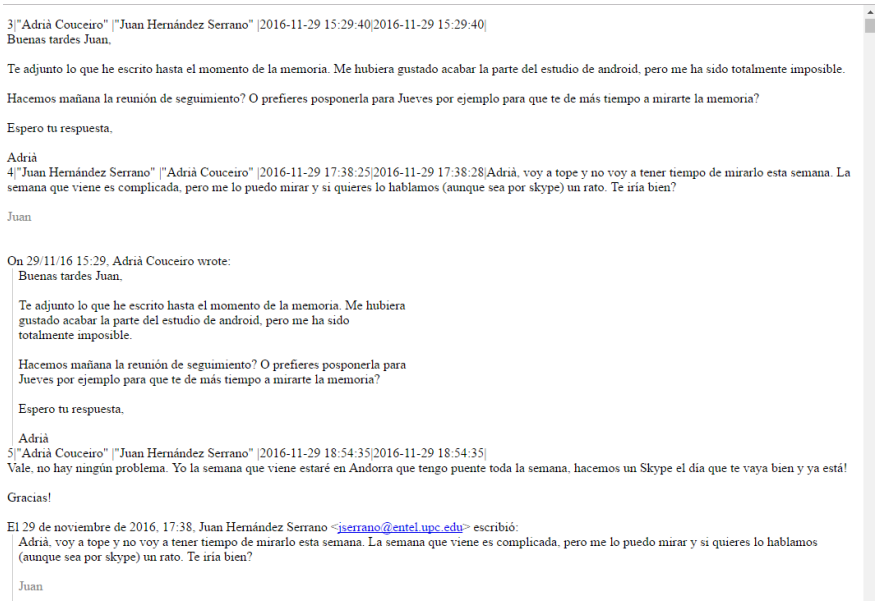
Fig. 4.15 Cuentas de correo electrónico

No obstante, la parte más importante a analizar es el propio contenido de los diferentes correos electrónicos. Dicho contenido se encuentra en la base de datos **mailstore.{dirección_usuario}@.gmail.com.db**, localizada en la ruta de bases de datos que comentamos anteriormente. En esta base de datos, existen diferentes tablas de datos, pero la más importante es la tabla **Messages**. Esta guarda los diferentes correos electrónicos junto a: dirección de correo, origen y destino, fecha de envío y recepción, asunto y el propio contenido. También encontramos una columna llamada **Snippet** que nos muestra el pequeño fragmento del contenido que se muestra en las notificaciones de nuestro terminal.

La columna que contiene los datos más reveladores es la llamada **bodyCompressed**, que como su propio nombre indica, almacena el contenido del correo electrónico en formato BLOB. Este formato recibe sus siglas de Binary Large Objects y comprime todo el contenido del mensaje, incluyendo imágenes y archivos, y lo almacena en binario dentro de la base de datos. Por tanto, para poder visualizar el contenido de los diferentes mensajes que hay en la base de datos debemos pasar este contenido en formato BLOB a algún formato que podamos visualizar.

Para realizar esta tarea, se ha creado un pequeño script en Python que nos ayuda a visualizar el contenido correctamente. Básicamente se recuperan ciertos campos de la base de datos, entre los que encontramos la columna bodyCompressed, y con la ayuda de la librería **zlib** los descomprimimos para visualizarlos como texto. Se analizará este sencillo script en mayor profundidad en el Anexo 1 de este mismo documento. La Fig. 4.16 muestra una captura de

pantalla donde se observa el resultado de ejecutar este script y cómo visualizamos el contenido del correo electrónico en formato html:


3|"Adrià Couceiro" |"Juan Hernández Serrano" |2016-11-29 15:29:40|2016-11-29 15:29:40|
Buenas tardes Juan,

Te adjunto lo que he escrito hasta el momento de la memoria. Me hubiera gustado acabar la parte del estudio de android, pero me ha sido totalmente imposible.

Hacemos mañana la reunión de seguimiento? O prefieres posponerla para Jueves por ejemplo para que te de más tiempo a mirarte la memoria?

Espero tu respuesta,

Adrià
4|"Juan Hernández Serrano" |"Adrià Couceiro" |2016-11-29 17:38:25|2016-11-29 17:38:28|Adrià, voy a tope y no voy a tener tiempo de mirarlo esta semana. La semana que viene es complicada, pero me lo puedo mirar y si quieres lo hablamos (aunque sea por skype) un rato. Te iría bien?

Juan

On 29/11/16 15:29, Adrià Couceiro wrote:
> Buenas tardes Juan,
>
> Te adjunto lo que he escrito hasta el momento de la memoria. Me hubiera gustado acabar la parte del estudio de android, pero me ha sido totalmente imposible.
>
> Hacemos mañana la reunión de seguimiento? O prefieres posponerla para Jueves por ejemplo para que te de más tiempo a mirarte la memoria?
>
> Espero tu respuesta,
>
> Adrià
5|"Adrià Couceiro" |"Juan Hernández Serrano" |2016-11-29 18:54:35|2016-11-29 18:54:35|
Vale, no hay ningún problema. Yo la semana que viene estaré en Andorra que tengo puente toda la semana, hacemos un Skype el día que te vaya bien y ya está!

Gracias!

El 29 de noviembre de 2016, 17:38, Juan Hernández Serrano <jserrano@entel.upc.edu> escribió:
> Adrià, voy a tope y no voy a tener tiempo de mirarlo esta semana. La semana que viene es complicada, pero me lo puedo mirar y si quieres lo hablamos (aunque sea por skype) un rato. Te iría bien?
>
> Juan

Fig. 4.16 Contenido correo electrónico

Como hemos podido observar, recuperar información relacionada con los correos electrónicos es relativamente sencillo, siempre y cuando tengamos acceso al terminal como usuario root. Una vez consigamos acceso root al terminal, solo debemos recuperar la base de datos que almacena todo el contenido de los mensajes de correo electrónico. Las mayores dificultades que nos encontramos son las de conseguir acceso al terminal y poder recuperar el contenido del cuerpo del correo electrónico en un formato que podamos visualizar y entender.

4.2.4. Contenido de la aplicación Whatsapp

Hoy en día todo el mundo tiene una cuenta de Whatsapp y utiliza esta aplicación en su vida cuotidiana para comunicarse. Este hecho hace que durante un análisis forense a dispositivos móviles sea totalmente indispensable recuperar el contenido de dicha aplicación. Recuperando la base de datos de Whatsapp, podemos recuperar todas las conversaciones del usuario y de esta forma, analizar estas conversaciones para buscar posibles evidencias.

Al igual que en el caso de la aplicación de correo electrónico Gmail, para poder acceder a las diferentes bases de datos y otro contenido de esta aplicación, necesitamos tener acceso como usuario root al dispositivo. Como usuarios normales podemos recuperar la base de datos de Whatsapp, eso sí, cifrada con **crypt12**. Actualmente, este es el formato de cifrado que tienen las diferentes bases de datos de Whatsapp. Antiguamente, ya se habían utilizado **crypt5, crypt7** o **crypt8.** Para poder descifrar con éxito esta base de datos en formato crypt12 necesitamos un archivo llamado **key** que se almacena en el directorio

data/data/com.whatsapp/files/. En esta ruta encontramos el fichero key, que es totalmente necesario para descifrar la base de datos.

Sin embargo, nos damos cuenta que existe una copia de la base de datos de Whatsapp ya descifrada en la ruta **data/data/com.whatsapp/databases.** Aquí encontramos la base de datos **msgstore.db**, que contiene todo el contenido de las diferentes conversaciones de los usuarios.

Por tanto, como estamos viendo, existen dos formas diferentes de recuperar el contenido de las conversaciones de Whatsapp y para ambas necesitamos tener acceso root al terminal. Una vez tenemos acceso como SuperUsuario, podemos recuperar la base de datos ya descifrada. A pesar de ello, existen ciertos casos en los que esta base de datos se encuentra completamente vacía o con errores que no nos permiten visualizar su contenido, y por este motivo, la mejor forma de recuperar el contenido de las conversaciones de Whatsapp es descifrar la base de datos que conseguimos directamente del terminal.

Hasta el pasado mes de octubre de 2016, la única forma de poder descifrar la base de datos era subiéndola junto con el fichero key a la página web http://whatcrypt.com/, en la cual no se explica ni se puede ver el proceso que seguían para descifrar la base de datos. Se trata de un método que, pese a no poder asegurar que el proceso se esté realizando correctamente, y donde además, no sabemos qué hacen los dueños de la página web con el contenido que nosotros subimos, se aceptaba en ciertos casos como procedimiento válido en algunos procesos judiciales, debido que no existía otra forma de descifrar la base de datos.

No obstante, en Octubre de 2016 se hicieron públicas otras formas de conseguir descifrar esta base de datos. La herramienta opensource **WhatsApp Viewer,** cuyo autor es el usuario de Github **andreas-mausch** [15], llevaba mucho tiempo utilizándose para poder visualizar correctamente las diferentes conversaciones de Whatsapp con un formato parecido al de la propia aplicación. Dicha herramienta, además de facilitar la visualización del contenido, también permite descifrar la base de datos si se encuentra cifrada con crypt5, crypt7, crypt8 o crypt12. Para ello, como se ve en la Fig. 4.17, debemos subir tanto la base de datos cifrada como el archivo que contiene la clave (key file) y la aplicación se encarga de proporcionarnos la base de datos ya descifrada (Fig. 4.18).

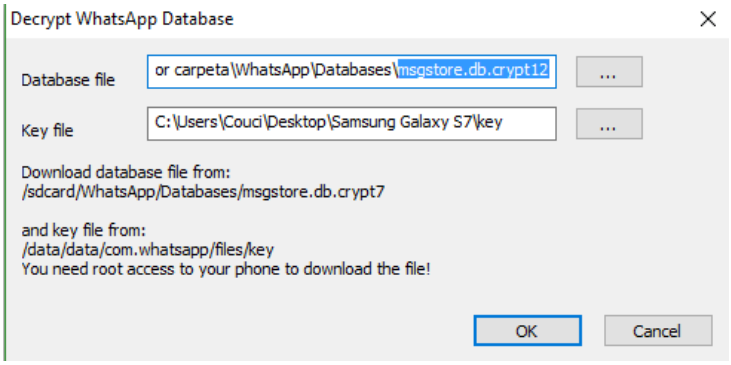


Fig. 4.17 Ficheros para descifrar la base de datos

Fig. 4.18 Visualización de la base de datos descifrada

Como podemos observar en las anteriores capturas de pantalla, es realmente sencillo conseguir visualizar el contenido de la base de datos. Al tratarse de una herramienta opensource, podemos investigar qué es exactamente lo que estamos haciendo y, de esta forma, demostrar que no manipulamos los datos.

Existen otras alternativas para poder realizar este proceso de descifrado. Entre las cuales, encontramos una sencilla herramienta desarrollada en Java por el usuario de GitLab **Ibrahim** [16], que haciendo uso del fichero que contiene la clave y de la base de datos cifrada, nos devuelve el contenido de base de datos para poder visualizarlo con cualquier visualizador de bases de datos como **SQLite** o herramientas como WhatsApp Viewer. En el Anexo 2 de la memoria analizaremos detalladamente el funcionamiento de este programa en Java.

En la siguiente captura de pantalla, Fig. 4.19, se podrá ver el procedimiento para descifrar la base de datos y el resultado obtenido al ejecutar el programa:

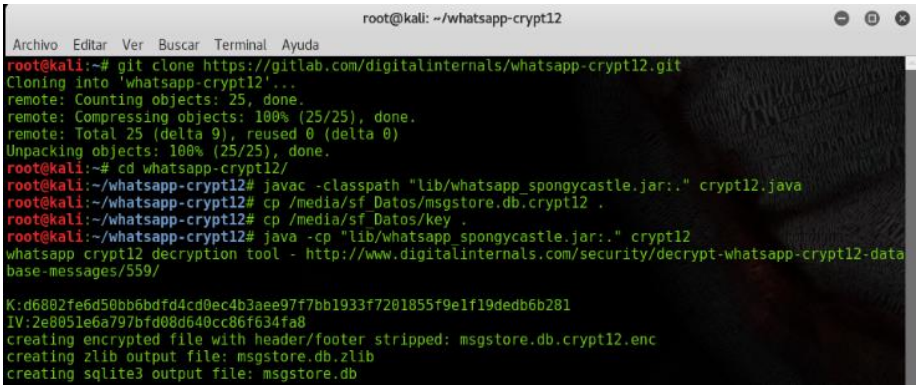


Fig. 4.19 Descifrar base de datos de Whatsapp

4.3. Diferencias análisis forense Android 5.1 y Android 6.0

En este apartado se analizarán las posibles diferencias entre un análisis digital forense a un dispositivo Android 5.1 y otro con la versión Android 6.0. Para ello, se ha realizado el mismo procedimiento que se ha seguido en el caso del Samsung Galaxy S7 Edge, con un terminal Sony Xperia M2 que cuenta con una versión inferior de Android.

Al buscar diferencias entre ambas versiones de Android, nos encontramos con dos aspectos especialmente interesantes. Para empezar, con Android 5.1, existen diferentes métodos para saltarse el bloqueo de pantalla. Algunas formas se basan en instalar herramientas externas desde la tarjeta SD gracias al recovery mode de Android. De esta forma, las herramientas ejecutan una serie de comandos que eliminan la pantalla de bloqueo del dispositivo, ya sea patrón, pin o contraseña. Esta “vulnerabilidad” se solucionó con la actualización a Android 6.0 al aislar la seguridad del Hardware, que protege las claves del dispositivo frente a ataques contra el Kernel o ataques físicos.

Otra forma que encontramos para saltarnos la contraseña de desbloqueo es a través de un fallo de implementación. Esta opción se basa en “romper” el proceso de la aplicación de la cámara y conseguir así acceso al dispositivo, tal y como vemos en la Fig. 4.20. Con el terminal bloqueado, debemos hacer uso del inicio rápido de la aplicación de la cámara y, una vez abierta, acceder a la pestaña de ajustes del dispositivo. A continuación, se nos pedirá la contraseña del dispositivo y nosotros debemos escribir una gran cantidad de asteriscos (*). Al escribir unos cuantos, copiamos y pegamos varias veces hasta que, al cabo de un rato, el proceso de la cámara se rompe y conseguimos acceso al terminal desbloqueado. Este método funciona en algunas versiones de Android 5.0 y 5.1, pero no en todas, ya que en la gran mayoría se arregló el fallo.

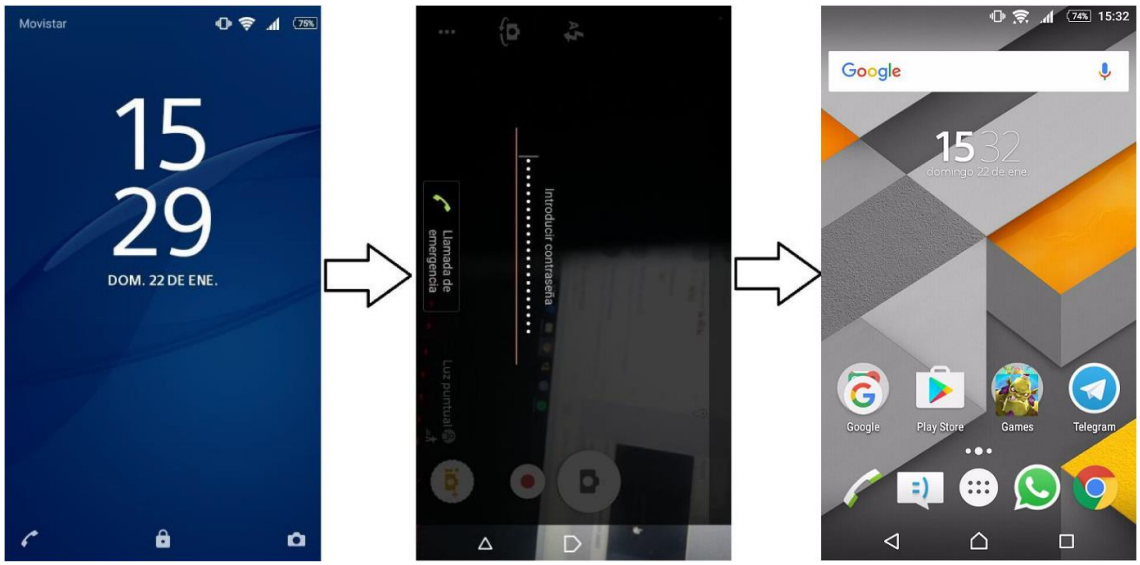


Fig. 4.20 Bypass Lockscreen rompiendo el proceso de la cámara

Existen herramientas de terceros que también permiten eliminar la pantalla de bloqueo de dispositivos Android. La herramienta **Dr. Fone** [17] (Fig. 4.21), por ejemplo, entre otras opciones, nos permite eliminar la pantalla de bloqueo. No funciona con todos los dispositivos, ya que actualmente solo es compatible con ciertos modelos de la marca Samsung, pero es una opción a tener en cuenta cuando realicemos un análisis forense digital. Dependiendo del dispositivo a analizar, si se trata de un modelo de Samsung, puede ser una buena idea probar de eliminar la pantalla de bloqueo con esta herramienta. Se trata de una herramienta de pago en la que debemos contratar los servicios que nos interese por separado. En este caso, podríamos comprar una licencia para hacer uso del módulo de eliminación de la pantalla de desbloqueo. Se ha probado con diferentes terminales en la versión de prueba para comprobar si conseguíamos eliminar la pantalla de desbloqueo, pero ha sido totalmente imposible.

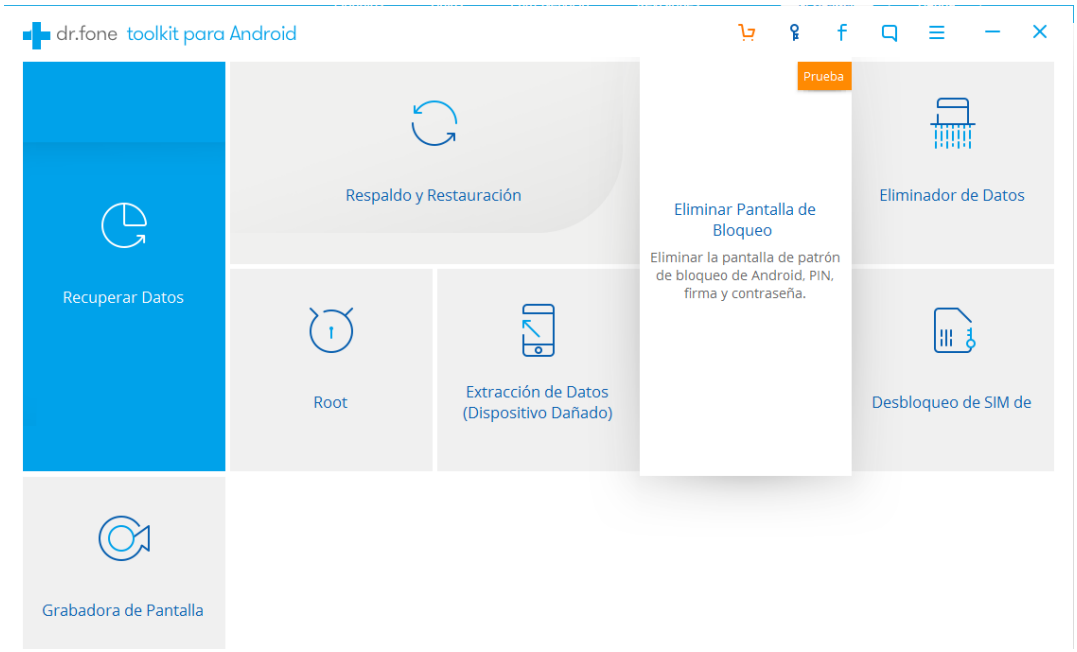


Fig. 4.21 Eliminar pantalla de bloqueo con Dr. Fone

Otra gran diferencia que vemos entre Android 5.0 y Android 6.0, es que en la actualización a la versión 6.0, Android decidió que al conectar el teléfono vía USB a un ordenador, por defecto se encontrará en modo “Solo Carga”. De esta forma, para poder recuperar datos del teléfono debemos desbloquearlo y aceptar que se pueda acceder vía USB a los diferentes datos del dispositivo. Con versiones anteriores de Android, al conectar el teléfono podíamos recuperar información sin tener que desbloquear el dispositivo para aceptar la transferencia USB. Se trata de una característica muy importante, ya que con Android 6.0 si no tenemos acceso al terminal es completamente imposible recuperar información.

Finalmente, se puede asegurar que teniendo acceso al terminal, podemos recuperar la misma cantidad de datos, tanto en Android 6.0 como en Android 5.0. Las principales diferencias que encontramos están relacionadas, por lo tanto,

con el desbloqueo del dispositivo y con la recuperación de datos sin desbloquear el terminal.

4.4. ¿Qué implica KNOX en el análisis forense digital?

Ahora que ya hemos visto cómo debemos analizar la distinta información más relevante de un dispositivo Android en busca de evidencias, debemos centrarnos en qué implicaciones tiene el sistema de seguridad propietario de Samsung conocido como **Samsung KNOX**. Como se ha visto en el capítulo 3 de este documento, Estudio Teórico de Android, el sistema Samsung KNOX está pensado claramente para usos empresariales.

Este sistema, proporciona un entorno cifrado y aislado de los datos personales en el que almacenar los datos y aplicaciones que se desee. De esta forma, el usuario es capaz de ejecutar aquellas aplicaciones que considere oportunas en un contenedor totalmente cifrado con el Algoritmo AES usando una clave de 256 bits, y así consigue que, tanto los datos protegidos por KNOX como los de las aplicaciones que se ejecutan en el contenedor de KNOX, queden totalmente protegidos frente a un posible robo de información.

Samsung KNOX incorpora medidas de seguridad también a nivel de Hardware. El llamado Bit de Garantía permite que, en caso de detectarse cualquier tipo de pirateo o de acceso root al dispositivo, salte un fusible electrónico de un solo uso impidiendo totalmente el acceso por parte de cualquier usuario a los datos protegidos por Samsung KNOX. Gracias a este fusible, los datos quedan cifrados y totalmente inutilizables e inaccesibles, lo que hace que sea muy complicado obtener información del dispositivo.

Imaginemos, por ejemplo, que el usuario tiene las aplicaciones de Whatsapp o Gmail protegidas por Samsung KNOX. Ya hemos visto que para poder visualizar el contenido de ambas aplicaciones necesitamos tener acceso root al dispositivo, por tanto, al entrar como root todos los datos de la aplicación y, por consiguiente, las diferentes bases de datos, quedarán totalmente inaccesibles y cifradas.

Otro aspecto fundamental de Samsung KNOX que puede acarrear diferentes problemas durante un análisis forense digital, es el hecho que permite al dueño del terminal gestionar todo su contenido remotamente a través de la aplicación Samsung SDS EMM, que ejecuta un cliente MDM (Mobile Device Management). De esta forma, el dueño del terminal móvil es capaz de administrar el contenido del mismo, y por tanto, de eliminar evidencias remotamente en caso de que estas existan. Es un hecho que debe tenerse muy en cuenta. Debemos aislar el dispositivo a analizar y evitar a toda costa que el terminal se conecte a Internet para minimizar las probabilidades de que su dueño lo administre a remotamente.

Es muy importante conocer las diferentes medidas de seguridad que implementa el sistema Samsung KNOX puesto que, como vemos, crea una serie de barreras que impiden un correcto análisis forense. Debemos considerar este sistema de seguridad, ya que en caso de estar activado en el terminal móvil, debemos actuar con cautela y no podemos seguir los pasos habituales para no dañar o dejar inaccesibles los datos que queremos analizar.

4.5. Herramientas para el análisis forense en Android

En este último apartado, se darán a conocer diferentes herramientas que podemos utilizar para el análisis digital forense en general, y que también son útiles para Android. Podemos encontrar un gran número de herramientas y distribuciones basadas en Linux que nos permiten realizar un análisis digital forense completo. Si bien es cierto que muchas de estas herramientas son de pago, también podemos encontrar algunas de código abierto que pueden ser de ayuda en nuestra investigación.

Para realizar este proyecto se ha trabajado con algunas de las herramientas más conocidas, como por ejemplo: **The Sleuth Kit (+Autopsy)**, **MOBILedit Forensics,** el propio **ADB** de Android o distribuciones como **Kali Linux** o **Santoku**. A lo largo del documento, ya se ha ido viendo un poco cómo funcionan y qué opciones nos ofrecen en las diferentes capturas de pantalla, y por esta razón, en este apartado nos centraremos en otras herramientas que no se han comentado hasta el momento.

Empezamos con la distribución de **SANS Investigative Forensic Toolkit (SIFT)** [18]. Se trata de una distribución basada en Ubuntu y creada por la organización SANS, una de las organizaciones más importantes en el mundo de la Seguridad Informática en cuanto a formación, investigación y certificaciones. En esta distribución (Fig. 4.22), encontramos, por una parte una serie de documentación muy útil para empezar a realizar un análisis forense, y por otra parte, también contiene algunas herramientas importantes como The Sleuth Kit, Autopsy, **log2timeline,** para crear líneas temporales de la utilización del dispositivo o **Volatility,** para realizar análisis de memoria. Es una distribución muy completa y que cuenta con el valor añadido de estar creada por una organización importante como es el caso de SANS.

Fig. 4.22 SANS Investigative Forensic Toolkit (SIFT)

En cuanto a distribuciones basadas en Linux para el análisis forense digital, también podemos encontrar **Santoku** [19], **Kali Linux** [20], **Caine** [21] o **DEFT**

[22]. Las ventajas de instalarse una distribución ya preparada para el análisis forense es que encontraremos las herramientas más utilizadas ya instaladas y configuradas para su uso. Sin embargo, al tratarse de distribuciones del mismo sistema operativo, podremos instalar nosotros manualmente las diferentes herramientas en nuestro propio sistema.

También podemos encontrar herramientas de pago muy útiles, como las desarrolladas por la empresa israelí **Cellebrite** [23]. Esta empresa crea soluciones, tanto a nivel de hardware (Fig. 4.23) como de software, para la extracción y análisis de los datos de dispositivos móviles. Se centran en el campo del análisis digital forense para terminales móviles y ofrecen productos muy completos. En concreto, Cellebrite UFED (Unified Digital Forensics Platform), permite la adquisición lógica o física de la información del dispositivo y su exportación directa a una tarjeta SD o un dispositivo USB.

Asimismo, nos proporciona métodos para romper códigos de bloqueo del dispositivo o incluso para descifrar la información protegida del dispositivo. Ha ganado el premio “Phone Forensic Hardware Tool of the Year” durante varios años consecutivos, y por tanto, se convierte en una muy buena opción para realizar análisis forenses completos.

Fig. 4.23 Solución Hardware de Cellebrite

Debemos hablar también de la herramienta **Oxygen Forensic Suite** [24] (Fig. 4.24). Se trata de una herramienta de pago, muy completa, que nos permite realizar todas las fases del análisis digital forense. Ofrece módulos para la extracción física y lógica de la información del dispositivo y es compatible con más de 15.000 modelos de dispositivos móviles diferentes. Otra característica muy interesante de esta herramienta, es que nos proporciona la habilidad de analizar algunas de las aplicaciones móviles más populares en busca de datos interesantes e incluso nos permite conseguir las contraseñas de las cuentas del usuario.
Esta herramienta nos permite analizar los datos obtenidos para crear líneas temporales de la utilización del dispositivo y de las comunicaciones del usuario, lo cual es especialmente útil para poder analizar en profundidad un espacio

temporal concreto. También nos ayuda a crear gráficos sociales a través de los datos obtenidos, y de esta forma, conocer las posibles conexiones del usuario del teléfono con sus contactos y las diferentes aplicaciones utilizadas. Finalmente, nos permite exportar los resultados obtenidos a los formatos más comunes, como PDM o XML, para poder visualizarlos con mayor facilidad o incluirlos en nuestro informe final.

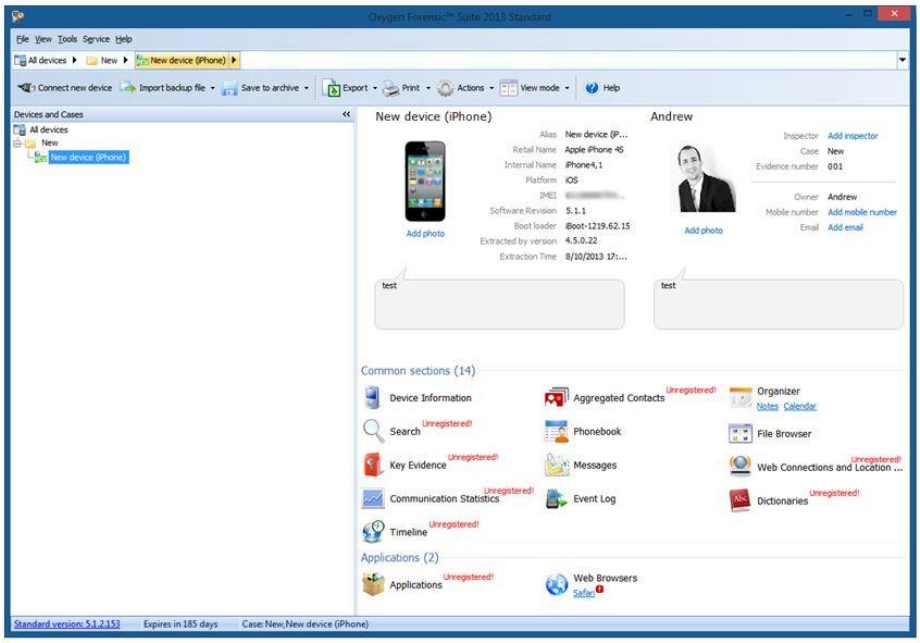


Fig. 4.24 Oxygen Forensic Suite

5. TROPIEZOS Y ESCOLLOS DURANTE LA REALIZACIÓN DEL PROYECTO

Durante este proyecto se han seguido diferentes procedimientos para conseguir realizar un análisis forense digital lo más completo posible. La gran mayoría de procedimientos se han llevado a cabo con éxito. A pesar de ello, algunas partes del proyecto han presentado más problemas de los esperados, e incluso en ocasiones no se obtenía el resultado imaginado cuando, a priori, se realizaba correctamente el procedimiento. Por este motivo, se ha decidido añadir este capítulo en la memoria del proyecto, el cual tiene la intención de dar a conocer aquellos problemas con los que me he encontrado durante la realización del proyecto para que pueda ser de ayuda en posteriores estudios en los que aparezcan estos problemas.

La primera dificultad importante que he encontrado durante el proyecto, ha sido el hecho de ser totalmente incapaz de conseguir acceso al terminal móvil sin conocer la contraseña o patrón de desbloqueo del mismo. Puede parecer una tontería, pero se han dedicado muchas horas de búsqueda y prueba de diferentes métodos, que en principio permitían realizar un bypass de la pantalla de bloqueo del dispositivo. Finalmente, se ha llegado a la conclusión que con versiones de Android 6.0 y superiores no existe forma, al menos gratuita, de conseguirlo. Ya hemos comentado que existen diferentes herramientas como Dr. Fone o Cellebrite que permiten saltarse el bloqueo de pantalla del dispositivo móvil, pero en estos casos, se trata de herramientas de pago que no se han probado en este proyecto.

Otro problema con el que me he encontrado a lo largo del proyecto, hace referencia a la adquisición de la información. Ya hemos visto que para realizar la adquisición física del dispositivo, o copia bit a bit, que nos permite hacer una copia completa de todo el sistema de ficheros, hacemos uso de la herramienta propia de Linux **dd** que viene integrada en el Android Debug Bridge (ADB). Al ser una herramienta propia de Linux, supuse en un principio, que su comportamiento sería el mismo y que se utilizaría igual. Por tanto, la primera vez que intenté realizar la copia, escribí el siguiente comando:

 # dd if=/dev/block/platform/…/by-name/userdata of=/sdcard/data.dd

La parte importante es la salida. En principio, la herramienta dd realiza la copia bit a bit indiferentemente de la extensión que le demos a nuestro fichero de salida. Una vez hecha la copia e intentar visualizarla con Autopsy, me di cuenta que ni Autopsy ni The Sleuth Kit eran capaces de analizar la imagen del dispositivo Android. Intenté modificar la extensión manualmente a .img del fichero .dd para ver si así solucionaba los problemas, pero no se solucionaban.

Se volvió a ejecutar el comando dd indicando que la salida era **data.img** y de esta forma se puede, ahora sí, analizar y visualizar su contenido con Autopsy. El problema reside en que la herramienta dd que viene integrada en ADB, no realiza el mismo proceso de copia para cualquier extensión del fichero de salida, sino

que realiza un proceso diferente dependiendo del formato de salida del fichero resultante.

Finalmente, el hecho de visualizar el contenido de la aplicación de correo electrónico Gmail se ha convertido en un contratiempo considerable. Como ya hemos visto, el contenido de los diferentes mensajes de correo electrónico se encuentra en la base de datos en formato BLOB. Se trata de un formato propio de algunos gestores de bases de datos que permite almacenar objetos y texto de forma mucho más eficiente, ya que se encuentran comprimidos. El hecho de recuperar este contenido en su formato original (texto, imagen…), se ha convertido en una auténtica adversidad.

Al tratarse de un formato de compresión, mi primera idea fue la de intentar descomprimir este campo de texto con aplicaciones como 7-Zip. Como era de esperar, el resultado no fue para nada satisfactorio. El siguiente paso fue buscar información por Internet para intentar recuperar esta columna de la base de datos en formato texto. Después de algún tiempo de búsqueda, se encontraron sentencias SQL como:

SELECT hex(bodyCompressed) FROM messages;

SELECT CONVERT(VarChar(40), bodyCompressed) FROM [messages];

SELECT CONVERT(bodyComrpessed USING utf8) FROM messages;

Estas sentencias SQL tampoco me permitieron visualizar el contenido de los mensajes en formato texto. Por más que buscaba información para poder realizar este proceso, no encontraba ninguna solución que funcionara, hasta que finalmente, encontré un pequeño script en formato Python que parecía que podía realizar el proceso deseado. Después de modificar algunas partes del script, para adaptarlo a mi entorno y a los datos que yo quería recuperar y modificar, conseguí que el script funcionara y que me devolviera un archivo .html con el contenido íntegro de todos los mensajes de la aplicación de correo electrónico. Dicho script se puede ver en el Anexo 1 de este documento con una breve explicación de su funcionamiento.

6. CONCLUSIONES Y LÍNEAS FUTURAS

El resultado final del presente proyecto, en cuanto a los objetivos propuestos en un principio, ha sido satisfactorio. Como objetivos principales para este proyecto teníamos el dar a conocer cómo debe llevarse a cabo una investigación forense digital en terminales móviles y el orientar dicha investigación a dispositivos Android, para analizar las diferentes herramientas existentes en la actualidad para lleva a cabo este proceso.

En el segundo capítulo de este documento se realiza un estudio para determinar cómo debe llevarse a cabo un análisis forense digital y que consideraciones, legales y técnicas, debemos tener en cuenta. Si además valoramos que en el tercer capítulo se realiza un estudio teórico sobre el sistema operativo Android, somos capaces de empezar nuestra propia investigación digital forense a terminales Android.

Durante la investigación propiamente dicha, se ha conseguido realizar una adquisición física de los datos del dispositivo. Esto conlleva recuperar una réplica exacta del estado del terminal en el momento de la adquisición. Para ello, previamente se ha lleva a cabo un proceso para conseguir privilegios root en el dispositivo. El proceso utilizado para obtener estos privilegios, se ha escogido dentro de las diferentes posibilidades debido a su carácter poco intrusivo y que, además, nos permitía conseguir estos privilegios sin tener que realizar un reset de fábrica al dispositivo.

Una vez adquirida la copia física del dispositivo, se han utilizado diferentes herramientas como Autopsy, Exiftool o WhatsApp Viewer para analizar toda la información en busca de posibles evidencias. Se ha podido ver, durante todo el proceso, que es relativamente sencillo realizar un análisis forense digital a dispositivos Android, siempre y cuando hayamos podido acceder al terminal, que es lo más complicado. Las diferentes herramientas existentes nos proporcionan la capacidad de analizar toda la información y nos ofrecen muchas facilidades para recuperar archivos eliminados.

No obstante, no todo ha sido tan sencillo. También nos hemos encontrado con algunas dificultades durante el caso práctico. La primera de todas ellas y la más importante, es que sin tener acceso al terminal no podemos realizar el análisis forense. Hemos visto que no se ha podido saltar la pantalla de bloqueo del dispositivo con los diferentes problemas que esto conlleva. En un caso real, si no somos capaces de desbloquear el dispositivo, nuestras opciones de realizar un análisis forense conciso y exhaustivo se reducen en gran medida.

Actualmente, empiezan a aparecer estudios en los que grupos de investigación intentan analizar diferentes formas de romper el bloqueo del dispositivo. Estos estudios están pensados para concienciar a la sociedad sobre las medidas de seguridad que deben tomar cuando quieren proteger su terminal móvil. Sin embargo, podemos utilizar estas investigaciones para conseguir nuevos métodos o para tener ideas diferentes para intentar saltarnos la restricción que crea el no tener acceso al dispositivo.

Finalmente, se han visto también los próximos avances en cuanto a seguridad, que incorporará Android en su versión Android 7.0. Además, se ha estudiado en qué consisten las soluciones de seguridad propietarias como Samsung KNOX y qué niveles de seguridad ofrecen. Tanto las medidas de seguridad propias de Android como las propietarias de los diferentes fabricantes de teléfonos móviles, son de gran ayuda para proporcionar un nivel de seguridad mucho mayor al usuario. Sin embargo, añaden nuevas dificultades a la investigación digital forense que debemos afrontar y para las que debemos buscar soluciones con urgencia.

REFERENCIAS BIBLIOGRÁFICAS

[1] ISO 27037:2012. Information technology - Security techniques - Guidelines for identification, collection, acquisition and preservation of digital evidence. Recuperado de **https://www.iso.org/obp/ui/es/#iso:std:iso-iec:27037:ed-1:v1:en**

[2] IETF. (Febrero 2002). Guidelines for Evidence Collection and Archiving (rfc3227).Recuperado de **https://www.ietf.org/rfc/rfc3227.txt**

[3] Donovan, M. (Noviembre 2012). Integrated Digital Forensic Process Model. Master's dissertation, University of Pretoria, Department of Computer Science: *Computers & Security*, *38*, 103-115. Recuperado de **http://repository.up.ac.za/bitstream/handle/2263/25433/dissertation.pdf?sequence=1**

[4] Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal, BOE 298 (1999). Recuperado de **https://www.boe.es/buscar/act.php?id=BOE-A-1999-23750**

[5] Ley Orgánica 1/2015, de 30 de marzo, por la que se modifica la Ley Orgánica 10/1995, de 23 de noviembre, del Código Penal, BOE 77 (2015). Recuperado de **https://www.boe.es/diario_boe/txt.php?id=BOE-A-2015-3439**

[6] Mendiola Zuriarrain, José (Enero 2015). ¿Usas patrón de desbloqueo en el móvil? Pues ten cuidado, han descubierto cómo descifrarlo. *El País.* Recuperado de **http://tecnologia.elpais.com/tecnologia/2017/01/25/actualidad/1485336951_413986.html**

[7] Android. (s.f.). Security Enchancements. Recuperado de **https://source.android.com/security/enhancements/index.html**

[8] Samsung. (2016). Documentos Samsung Knox. Recuperado de **https://www.samsungknox.com/es/knox-technology/white-papers**

[9] Samsung. (2016). Smart Switch de Samsung. Recuperado de **http://www.samsung.com/es/support/smartswitch/**

[10] Odin. (2014). Download Odin 3.09: Samsung Odin 3.12.3 with Android ROM Flashing Tool. Recuperado de **http://odindownload.com/**

[11] Chainfire (2016). CF-Auto-Root Repository. Recuperado de **https://autoroot.chainfire.eu/**

[12] MOBILedit, Compelson. (2017). Forensic solutions: MOBILedit Forensic. Recuperado de **http://www.mobiledit.com/forensic-solutions/**

[13] GitHub.(2017). AFLogical OSE: Open source Android Forensics app and framework. Recuperado de **https://github.com/nowsecure/android-forensics**

[14] Solvusoft. (2014). Fileviewpro: Open any File with one program, Developed by Microsoft Partner. Recuperado de **http://www.fileviewpro.com/**

[15] GitHub.(2017). Andreas- mausch/whatsapp-viewer: Small tool to display chats from the Android msgstore.db database (crypt12). Recuperado de **https://github.com/andreas-mausch/whatsapp-viewer**

[16] GitLab. (2016). Digital Internals/ whatsapp – crypt12. Recuperado de **https://gitlab.com/digitalinternals/whatsapp-crypt12**

[17] Dr. Fone, Wandershare. (2017). Eliminar pantalla de bloqueo Android. Recuperado de **https://drfone.wondershare.com/es/android-lock-screen-removal.html**

[18] SANS DFIR. (2016). SANS Investigative Forensic Toolkit (SIFT) Workstation Version 3. Recuperado de **https://digital-forensics.sans.org/community/downloads**

[19] Santoku. (2016). Santoku 0.5. Recuperado de **https://santoku-linux.com/**

[20] Kali by Offensive Security. (2016). Our Most Advanced Penetration Testing Distribution, Ever. Recuperado de **https://drfone.wondershare.com/es/android-lock-screen-removal.html**

[21] Caine. (2016). Computer Forensics Linux Live Distro. Recuperado de **http://www.caine-live.net/**

[22] DEFT. (2016). Yes, we are alive! Recuperado de **http://www.deftlinux.net/**

[23] Cellebrite. (2016). Cellebrite, delivering Mobile expertise. Recuperado de **http://www.cellebrite.com/**

[24] Oxygen Forensic. (2016). Recuperado de **https://www.oxygen-forensic.com/es/**

[25] Android. (s.f.). Aspectos fundamentales de la aplicación. Recuperado de **https://developer.android.com/guide/components/fundamentals.html**

[26] Android. (s.f.). Trusty TEE. Recuperado de **https://source.android.com/security/trusty/**

[27] Android. (s.f.). Host-based Card Emulation. Recuperado de **https://developer.android.com/guide/topics/connectivity/nfc/hce.html**

ANEXO 1: SCRITP EN PYTHON PARA RECUPERAR EL CONTENIDO DE GMAIL

```python
import sqlite3
import zlib
import sys
reload(sys)
sys.setdefaultencoding('utf-8')


connection = sqlite3.connect('mailstore.adria.couceiro@gmail.com.db')
c = connection.cursor()
c.execute("select _id, fromAddress, toAddresses, datetime(dateSentMS/1000, 'unixepoch', 'localtime'),
datetime(dateReceivedMS/1000, 'unixepoch', 'localtime'), case when body not Null then body else
bodyCompressed end from messages ")
rows = c.fetchall()

for row in rows:
    id, _from, _to, sent, recv, body = row
    try:
        body = zlib.decompress(body)
    except:
        pass
    print('{}|{}|{}|{}|{}|{}'.format(id, _from, _to,  sent, recv, body).encode('utf-8').strip())
```

Fig. A1.1 Script en python

La anterior captura de pantalla (Fig. 1.1) muestra el pequeño script en python que se ha utilizado para recuperar el contenido de los correos de la aplicación Gmail.

El primer fragmento del script incluye las librerías sqlite3 y zlib. La primera, para realizar todas las acciones que tengan relación con la base de datos. La segunda, para descomprimir el campo de la base de datos bodyCompressed que, como ya hemos visto, se encuentra en formato BLOB. Una vez incluidas estas dos librerías, incluimos la librería sys, lo recargamos y finalmente, le decimos que lo queremos codificado por defecto con utf-8 (para poder visualizarlo sin problemas).

Si nos fijamos en el segundo fragmento, vemos que se realizan las operaciones con la base de datos. Gracias a la librería sqlite3, creamos una conexión con la base de datos que queremos analizar, que previamente hemos conseguido a través de los datos del dispositivo. Una vez creada la conexión, ejecutamos una sentencia SQL en la que decidimos que campos necesitamos para poder visualizar correctamente el contenido. En este caso, se ha decidido seleccionar el campo id (identificador del correo), direcciones origen y destino, fechas de envío y recepción y, finalmente, el cuerpo del mensaje. Vemos que en caso de que la columna **body** no esté vacia, se selecciona esta columna. En caso de ser nula, se recoge la columna **bodyCompressed**, todo ello de la tabla **messages**.

Finalmente, en el tercer fragmento de código, vemos cómo hacemos un for para recorrer todos los resultados que anteriormente guardamos en el objeto rows. En este for, se intenta descomprimir el cuerpo del mensaje utilizando la librería zlib. Una vez descomprimido, en cada iteración del for se imprimen por pantalla (o en un fichero si así lo preferimos) los diferentes campos que hemos seleccionado separados por el símbolo | y codificados en utf-8. De esta manera, conseguimos recuperar los correos electrónicos y visualizarlos como más nos convenga.

ANEXO 2: PROGRAMA EN JAVA PARA DESCIFRAR LA BASE DE DATOS .CRYPT12

A continuación, se analizarán algunas de las partes del programa en Java utilizado para descifrar la base de datos de Whatsapp.

```
public class crypt12 {
    public static void main(String[] args) {
        String crypt12File    = "msgstore.db.crypt12";      // input file
        String crypt12FileEnc = "msgstore.db.crypt12.enc";  // encrypted file with header and footer stripped
        String decryptedZlibFile = "msgstore.db.zlib";      // zlib output file
        String decryptedDbFile = "msgstore.db";             // sqlite3 db output file
        String keyFile = "key";

        byte[] keyData = new byte[158];
        byte[] aesK = new byte[32];
        byte[] aesIV = new byte[16];
        byte[] crypt12Header = new byte[67];
        byte[] buffer = new byte[8192];
        int read;

        BufferedInputStream is;
        RandomAccessFile raf;
        FileOutputStream os;
        CipherInputStream isCipher;
        Cipher cipher;
```

Fig. A2.1 Crypt12 en java – Declaración de variables

En esta primera parte del programa (Fig. 2.1), el autor define todas las variables que utilizará durante la ejecución del programa. Podemos ver que el código es muy limpio y ordenado, lo que nos facilita mucho su análisis y comprensión. Vemos que la variable **crypt12File** hace referencia a nuestra base de datos cifrada con Crypt12.

```
// get K
is = new BufferedInputStream(new FileInputStream(keyFile));
is.read(keyData);
System.arraycopy(keyData, 126, aesK, 0, 32);
is.close();

// get IV
is = new BufferedInputStream(new FileInputStream(crypt12File));
is.read(crypt12Header);
System.arraycopy(crypt12Header, 51, aesIV, 0, 16);
is.close();

//System.out.println("K:" +Arrays.toString(aesK));
//System.out.println("IV:"+Arrays.toString(aesIV));
System.out.println("K:" +ListToHex(aesK));
System.out.println("IV:"+ListToHex(aesIV));

// create enc file by stripping header and footer
System.out.format("creating encrypted file with header/footer stripped: %s%n",crypt12FileEnc);
is = new BufferedInputStream(new FileInputStream(crypt12File));  // read msgstore.db.crypt12
is.skip(67);    // 67 byte header
int available = is.available();
raf = new RandomAccessFile(new File(crypt12FileEnc), "rw");

while((read=is.read(buffer))!=-1) {
    raf.write(buffer, 0, read);
}
raf.setLength(available - 20);  // 20 byte footer
raf.close();
```

Fig. A2.2 Crypt12 en java – Vector de inicialización y key

En la Fig. 2.2, podemos ver cómo el programa recupera tanto la clave para descifrar la base de datos como el vector de inicialización (IV). Una vez recuperado estos dos valores, crea un nuevo fichero, aún cifrado con Crypt12, en el que elimina los 67 bytes de Header y los 20 bytes del Footer. De esta forma, el contenido íntegro de esta nueva base de datos son propiamente datos.

```
// create zlib output file
System.out.format("creating zlib output file: %s%n",decryptedZlibFile);

is = new BufferedInputStream(new FileInputStream(crypt12FileEnc));
cipher = Cipher.getInstance("AES/CBC/NoPadding", "SC");
cipher.init(Cipher.DECRYPT_MODE, new SecretKeySpec(aesK, "AES"), new IvParameterSpec(aesIV));
isCipher = new CipherInputStream(is, cipher);
os = new FileOutputStream(decryptedZlibFile);

while((read=isCipher.read(buffer))!=-1) {
    os.write(buffer, 0, read);
}

os.close();
is.close();


// create sqlite3 db output file
System.out.format("creating sqlite3 output file: %s%n",decryptedDbFile);

is = new BufferedInputStream(new FileInputStream(crypt12FileEnc));
cipher = Cipher.getInstance("AES/CBC/NoPadding", "SC");
cipher.init(2, new SecretKeySpec(aesK, "AES"), new IvParameterSpec(aesIV));
isCipher = new CipherInputStream(is, cipher);
InflaterInputStream isInflater = new InflaterInputStream(isCipher, new Inflater(false));
os = new FileOutputStream(decryptedDbFile);

while((read=isInflater.read(buffer))!=-1) {
    os.write(buffer, 0, read);
}

os.close();
is.close();
```

Fig. A2.2 Crypt12 en java – Descifrar y guardar

Finalmente, este fragmento de código descifra la base de datos ayudándose de la clase **cipher** de Java, y crea dos archivos de salida: uno en formato zlib (comprimido) y otro en formato sqlite3 (base de datos). Vemos que todo lo necesario para descifrar la base de datos son: la clave AES utilizada en el cifrado (se consiguió en el paso anterior) y el vector de inicialización de AES, que también conseguimos anteriormente. Con estos dos valores, y la base de datos sin Header ni Footer, ya podemos recuperar el contenido de esta base de datos totalmente descifrado.