Institución Universitaria

Metodología para la extracción distribuida de imágenes forenses incrementales de dispositivos móviles con sistema operativo Android a través de redes WIFI

Hernán Alonso Bernal Bernal

Juan Carlos Restrepo Balvín

Instituto Tecnológico Metropolitano

Facultad de Ingenierías

Medellín, Colombia

2019

Metodología para la extracción distribuida de imágenes forenses incrementales de dispositivos móviles con sistema operativo Android a través de redes WIFI

Hernán Alonso Bernal Bernal

Juan Carlos Restrepo Balvín

Trabajo de investigación presentada como requisito parcial para optar al título de:

Magíster en Seguridad Informática

Director:

Magíster en Automatización Industrial: Thomas Julián Ramírez Rozo

Codirector:

Magíster en Seguridad de la Información: Fernando Alonso Quintero Londoño

Línea de Investigación:

Ciencias computacionales

Grupo de Investigación:

Automática, Electrónica y Ciencias Computacionales AE y CC

Instituto Tecnológico Metropolitano

Facultad de Ingenierías

Medellín, Colombia

2019

“El aspecto más triste de la vida actual es que la ciencia gana en conocimiento más rápidamente que la sociedad en sabiduría”. Isaac Asimov

Agradecimientos

Nos gustaría agradecer en estas líneas la ayuda que muchas personas y colegas que nos han prestado durante el proceso de investigación y redacción de este trabajo. En primer lugar, a los profesores de la Maestría en Seguridad Informática del Instituto Tecnológico Metropolitano, porque sin su conocimiento y tutoría, no hubiéramos tenido la guía para construir la investigación; especialmente al profesor Héctor Fernando Vargas Montoya, Coordinador de la Maestría, por colaborarnos en el proceso de seguimiento y presentación de este documento. Igualmente, agradecemos a nuestro director del proyecto de investigación, Thomas Julián Ramírez Rozo, quien con su conocimiento y ayuda pudimos darle un hilo conductor a nuestro trabajo y al codirector Fernando Alonso Quintero Londoño, que con sus opiniones nos dimos cuenta que nuestra investigación sería mejor.

Asimismo, agradecemos a nuestros compañeros de clase, que, con sus aportes en las diferentes presentaciones, nos ayudaron a perfilar más la investigación.

A nuestros familiares y parejas, que siempre supieron qué palabras de ánimo decirnos.

Al Instituto Tecnológico Metropolitano, por ser la sede del conocimiento adquirido en todos estos años.

Resumen

Los dispositivos móviles, como los *smartphones* y las *tablets*, se convirtieron desde hace algunos años en instrumentos no sólo para uso doméstico o personal, sino como herramienta de trabajo. Al mismo tiempo comenzaron a ser también susceptibles para los delitos informáticos, ya sea como facilitador o como puente para llegar a información corporativa. Se han analizado las arquitecturas de las diferentes versiones de Android para conocer sus diferencias y poder hacer una copia exacta del contenido del dispositivo móvil. También se analizó un método apropiado de cifrado para la imagen extraída. Generalmente las copias exactas de los dispositivos se hacen de forma completa cada vez, pero en este trabajo se prueba la copia exacta de la información nueva o modificada en el dispositivo móvil, con el fin que en cada extracción los archivos sean de tamaño pequeño y facilitar el transporte vía WiFi. Y, por último, la validación de la metodología propuesta en un ambiente controlado que la confronta con la metodología tradicional alámbrica. Los resultados se muestran por cada uno de los componentes analizados, donde se puede concluir que es muy posible que se adopte esta metodología.

Palabras clave: móviles, WiFi, dispositivos, imágenes forenses, compresión, seguridad, algoritmos

Abstract

Mobile devices, such as smartphones and tablets, have for some years become instruments not only for domestic or personal use, but as a work tool. At the same time, they also began to be susceptible to computer crimes, either as a facilitator or as a bridge to reach corporate information. The architectures of the different versions of Android have been analyzed to know their differences and to be able to make an exact copy of the content of the mobile device. An appropriate encryption method for the extracted image was also analyzed. Generally, the exact copies of the devices are made completely every time, but in this work the exact copy of the new or modified information on the mobile device is tested, so that in each extraction the files are of small size and facilitate the transport via WiFi. And, finally, the validation of the proposed methodology in a controlled environment that confronts it with the traditional wired methodology. The results are shown for each of the components analyzed, where it can be concluded that it is very possible that this methodology is adopted.

Keywords: mobile, Wireless Fidelity, image forensics, compression algorithms, security, algorithms

Contenido

Lista de figuras

Pág.

Lista de tablas

Pág.

Lista de abreviaturas

Abreviatura	**Término**
4K	4000 píxeles de resolución horizontal en pantalla
ADB	*Android Debug Bridge* (Puente de Depuración de Android)
CPU	*Central Process Unit* (Unidad Central de Procesamiento)
DHCP	*Dynamic Host Configuration Protocol* (Protocolo de Configuración Dinámica del Equipo Anfitrión)
Gbps	*Gigabits per second* (Gigabits por Segundo)
GHz	*Giga Hertz*
H.264	*MPEG4* (Grupo de Expertos de Fotos en Movimiento versión 4)
HAL	*Hardware Abstraction Layer* (Capa de Abstracción de *Hardware*)
HGST	*Hitachi Global StorageTechnologies* (Tecnologías de Almacenamiento Globales de Hitachi)
IEEE	*Institute of Electrical and Electronics Engineers* (Instituto de Ingenieros Electricistas y Electrónicos)
IoT	*Internet of Things* (Internet de las Cosas)
IT	*Information Technologies* (Tecnologías de la Información)
Mbps	*Megabits per second* (Megabits por Segundo)
MIMO	*Multiple-input Multiple-output* (Múltiple entrada Múltiple salida)
MP3	*MPEG Audio Layer III* (Capa de Audio del Grupo de Expertos de Fotos en Movimiento versión 3)
NFC	*Near Field Communication* (Comunicación de Campo Cercano)
NIST	*National Institute of Standards and Technologies* (Instituto Nacional de Estándares y Tecnologías)
RAM	*Random Access Memory* (Memoria de Acceso Aleatorio)
ROM	*Read Only Memory* (Memoria de Solo Lectura)
SHA-1	*Secure Hash Algorythm version 1* (Algoritmo Hash Seguro versión 1)
SSD	*Solid-state Drive* (Unidad de Estado Sólido)
SSID	*Service Set IDentifier* (Identificador de Configuración del Servicio)
XML	*eXtensible Markup Language* (Lenguaje de Marcado Extensible)
USB	*Universal Serial Bus* (Bus Serial Universal)

WiFi	Wireless Fidelity (Fidelidad Inalámbrica)

Objetivo General

Proponer una metodología para la extracción distribuida de imágenes forenses incrementales de dispositivos móviles con sistema operativo Android a través de redes WIFI.

Objetivos Específicos

1. Determinar cuáles arquitecturas de los actuales sistemas operativos Android son apropiadas de cara a la extracción distribuida, el cifrado y formato de las imágenes forenses.

2. Seleccionar un método apropiado de cifrado, de cara al envío seguro de la imagen a través de redes WiFi.

3. Proponer las técnicas para la extracción de imágenes incrementales tomando en consideración su factor de compresión e integridad.

4. Validar la metodología propuesta bajo un ambiente controlado a partir de casos de estudio, estableciendo un marco comparativo para la medición de fortalezas y debilidades.

Introducción

Desde la primera década del siglo XXI, se ha presentado un cambio drástico en la utilización de los dispositivos móviles. Anteriormente, los teléfonos celulares eran de uso personal, ahora estos dispositivos son indispensables en cualquier ambiente corporativo ya que, se constituyen en un contacto inmediato con el funcionario de la empresa, así como una herramienta móvil desde donde se gestiona la información entre otras actividades laborales [1].

Una de las metas de este trabajo es proveer a las empresas con una metodología para auditar los teléfonos inteligentes que suministran a sus empleados en caso de ocurrencia de un hecho delictivo informático, para de esta manera contar con la información de los mismos, actualizada y con trazas en el tiempo. Teniendo en cuenta que el sistema operativo más usado en este momento a nivel mundial y en Colombia es Android [2] en todas sus versiones, en este trabajo se tendrá focalización en este sistema operativo. Dado que es sumamente importante actuar con prontitud ante incidentes de seguridad [3], se han mejorado estos tiempos tomando imágenes forenses digitales de forma incremental y distribuida, puesto que los métodos existentes punto a punto y alámbricos no son muy eficientes en este aspecto.

Uno de los problemas al momento de un evento o incidente, en el caso de pérdida o robo de información corporativa, es poder obtener la información de los dispositivos móviles en el menor tiempo posible y para esto, se requiere hacer una extracción o copia exacta de dicha información para su posterior análisis; este procedimiento puede demandar bastante tiempo según la cantidad de datos a copiar. Asimismo, esta información puede aumentar de tamaño en el tiempo significativamente y por tal motivo, se necesita de mayor capacidad de almacenamiento, lo que implica mayor cantidad de dispositivos para ese almacenamiento [4]. Los métodos existentes al momento de realizar las extracciones de las imágenes forenses digitales requieren de *kits* forenses costosos, compuestos de una gran cantidad de cableado, equipos y convertidores [5]. Igualmente, puede encontrarse otro inconveniente al momento de la conexión del dispositivo por medio cableado y es que, si el equipo tiene el puerto de comunicaciones averiado, no puede hacerse la extracción de la imagen digital forense de este modo y una alternativa es la conexión inalámbrica, concretamente WiFi (Fidelidad Inalámbrica por sus siglas en inglés) con el estándar IEEE 802.11 n/ac [6].

Otras de las dificultades encontradas al realizar la extracción de imágenes forenses digitales en su forma clásica como por medio alámbrico, es que se encuentran técnicas en el mercado que garantizan la extracción de cualquier información de un dispositivo móvil con sistema operativo Android, pero no cumplen con las normas establecidas para la cadena de custodia expedida por la Fiscalía General de la Nación de la República de Colombia [7] y por lo tanto esta información no podrá ser utilizada como evidencia a la hora de presentarla en una audiencia, por no asegurar la integridad de la información extraída.

Este trabajo se divide principalmente en 4 secciones. En el capítulo 1 se encuentra la información de la literatura encontrada de todos los componentes que tiene la investigación, tanto del sistema operativo Android, como de redes WiFi, técnicas de extracción y de imágenes incrementales, entre otros. El capítulo 2 anuncia el reconocimiento y prueba de las técnicas para obtener un resultado y evaluar las propuestas. En el capítulo 3 se consignan los resultados de las pruebas a las diferentes técnicas mencionadas anteriormente y en el capítulo 4 se dan las conclusiones y recomendaciones, teniendo en cuenta lo probado y los resultados.

1. Marco Teórico y Estado del Arte

La metodología tradicional para la extracción de imágenes forenses de dispositivos móviles en su fase de adquisición depende de múltiples dispositivos físicos para **realizar** este procedimiento. Habitualmente se utilizan cables para conexión a diferentes puertos USB dependiendo del fabricante del dispositivo móvil [8]. Por lo general, los fabricantes de software y hardware para adquisición de imágenes forenses utilizan variados dispositivos o codificación para tal procedimiento, como por ejemplo bloqueadores de escritura, lo cual aumentaría el costo en los requerimientos y también, de algún modo, el tiempo de esa adquisición de la imagen.

También en la metodología actual de la extracción de imágenes forenses, se necesita en la mayoría de las ocasiones, que ocurra algún evento de seguridad para hacer la adquisición de la imagen y como pueden ser varios dispositivos móviles los que están comprometidos, se necesita disponer de tiempo para la adquisición por cada uno de los dispositivos, ya que las soluciones actuales generalmente no se componen de adaptadores múltiples sino sencillos y el cambio de tipos de conexión USB es variada.

Igualmente, que, aunque los fabricantes de renombre para la extracción de imágenes forenses efectúan el procedimiento de manera estricta y válida, no utilizan por lo general ni compresión ni cifrado de las imágenes y en el caso de tener que almacenarlas por largo tiempo, no es conveniente que no se tengan estos dos controles.

1.1 Arquitecturas apropiadas del sistema operativo Android

1.1.1 Arquitectura Android

Android es un Sistema operativo móvil de código libre originalmente desarrollado por Android Inc. y basada en una versión modificada de Linux. Fue adquirida por Google en 2005, que la convierte en Android Open Source Project (AOSP), el cual se rige por dos condiciones de código abierto: GNU que es la licencia general pública (GPLv2) y ASL 2.0 que es la licencia de software de Apache 2.0, que permite a productores comerciales sacar los productos respetando los secretos industriales [9].

Gironés [9] indica que Android es un sistema operativo que combina las siguientes cualidades, que no las tienen los demás sistemas operativos para móviles y tabletas:

- Plataforma realmente abierta: es de desarrollo libre basada en Linux y de código abierto. Se puede usar y personalizar sin costo económico.
- Adaptable a cualquier tipo de *hardware*: no sólo es para teléfonos y tabletas. También se puede encontrar en relojes, cámaras, electrodomésticos e inclusive automóviles.
- Portabilidad asegurada: las aplicaciones son desarrolladas en Java. Por lo tanto, pueden funcionar en cualquier tipo de CPU ahora y en el futuro.
- Arquitectura basada en componentes inspirada en internet: el diseño de la interfaz permite que la aplicación se pueda ver en una pantalla reducida, por ser elaborada en XML.
- Filosofía de dispositivo siempre conectado a internet.
- Aceptable nivel de seguridad: basado en el kérnel de Linux, las aplicaciones tienen una serie de permisos que limitan el rango de actuación, como la localización y el acceso a internet, entre otros.
- Optimizado para poca potencia y poca memoria: utiliza Dalvik, que es una máquina virtual optimizada de la máquina virtual de Java que tiene más consumo de memoria y energía.
- Alta calidad de gráficos y sonido: animaciones inspiradas en Adobe Flash y gráficos basados en OpenGL. Incorpora códecs que se utilizan comúnmente para la reproducción de sonido como MP3, AAC y H.264.

La primera arquitectura con el inicio del sistema operativo Android está compuesta por capas que inician por el kérnel, seguida por las librerías, el Android Runtime, el *framework* de Aplicaciones y por último las aplicaciones. Esta arquitectura fue utilizada para las versiones iniciales del sistema operativo Android [10], como se muestra en la Figura 1.1-1.

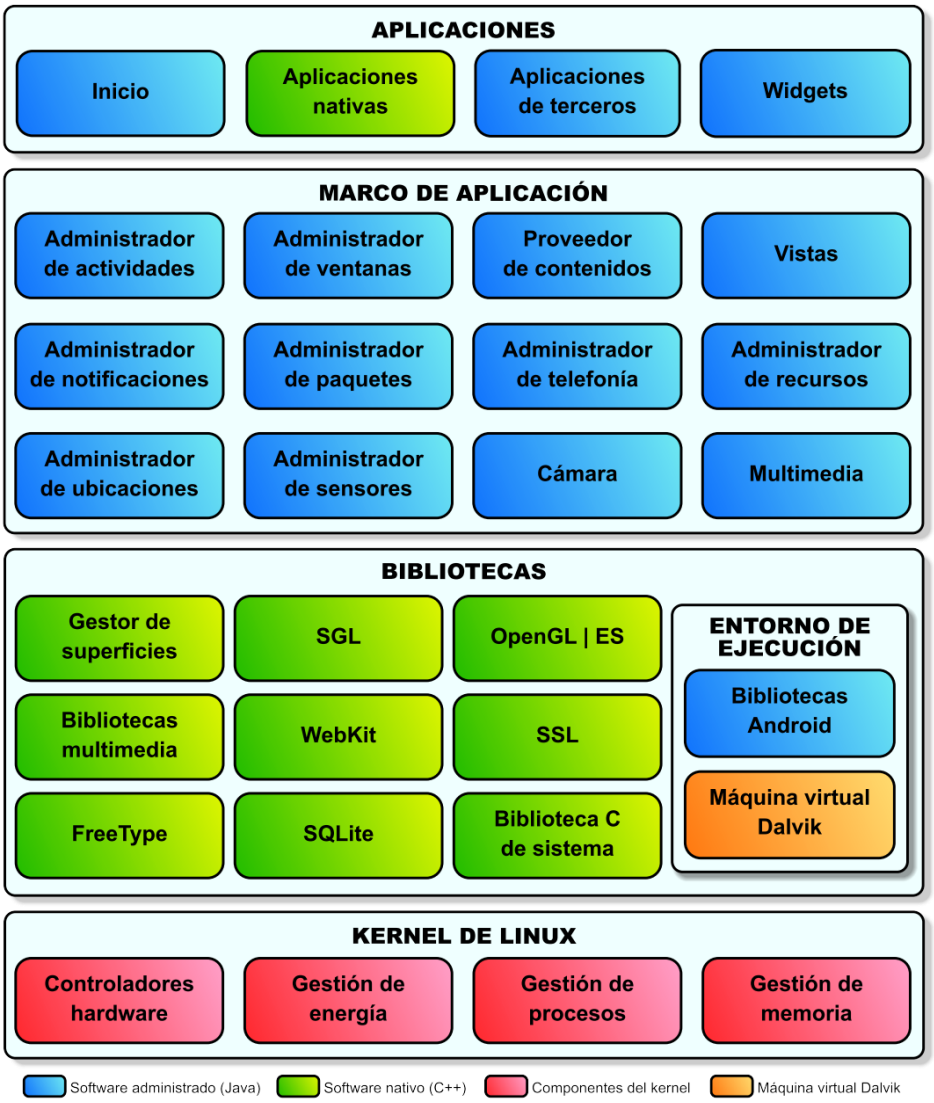

Figura 1.1-1 *Primera arquitectura de Android*

Para desarrollar en el sistema operativo Android, se requiere de las siguientes capas de la arquitectura:

- Aplicaciones: es la capa superior de la arquitectura, también llamada App, que es el *software* preinstalado en el sistema operativo o posteriormente instalado por los usuarios.
- Marco de aplicaciones: estas son desarrolladas en Java, en esta capa se encuentra los servicios y bibliotecas que necesitan las aplicaciones para su funcionamiento.

- Librerías: módulos desarrollados en lenguajes de programación en C y C++ (lenguajes de programación) que ofrecen servicios al marco de aplicaciones.
- Entorno de ejecución: cada aplicación ejecuta su propia instancia de una máquina virtual de Java llamada Dalvik VM, especificada de la plataforma de dispositivos móviles Android en las primeras versiones.
- *Kérnel*: es la capa más baja y compleja de la arquitectura del sistema operativo Android. En esta capa incluyen los controladores para el *hardware*, accesos al sistema de archivos, redes, entre otros. Cada que hay una ejecución de una aplicación, pasa a través de esta capa para obtener un resultado. El *kérnel* también es el encargado de gestionar procesos, memoria RAM y procesamiento [11].

La nueva arquitectura del sistema operativo Android es utilizada en las últimas versiones del sistema operativo Android (desde la versión 5 hasta la versión 9) y sigue la misma línea de la primera versión de la arquitectura, pero tiene un cambio significativo e importante y es la adición de una capa nueva llamada HAL (*Hardware Abstraction Layer*) [11]. En la Figura 1.1-2 se muestra la arquitectura actual del sistema operativo Android.

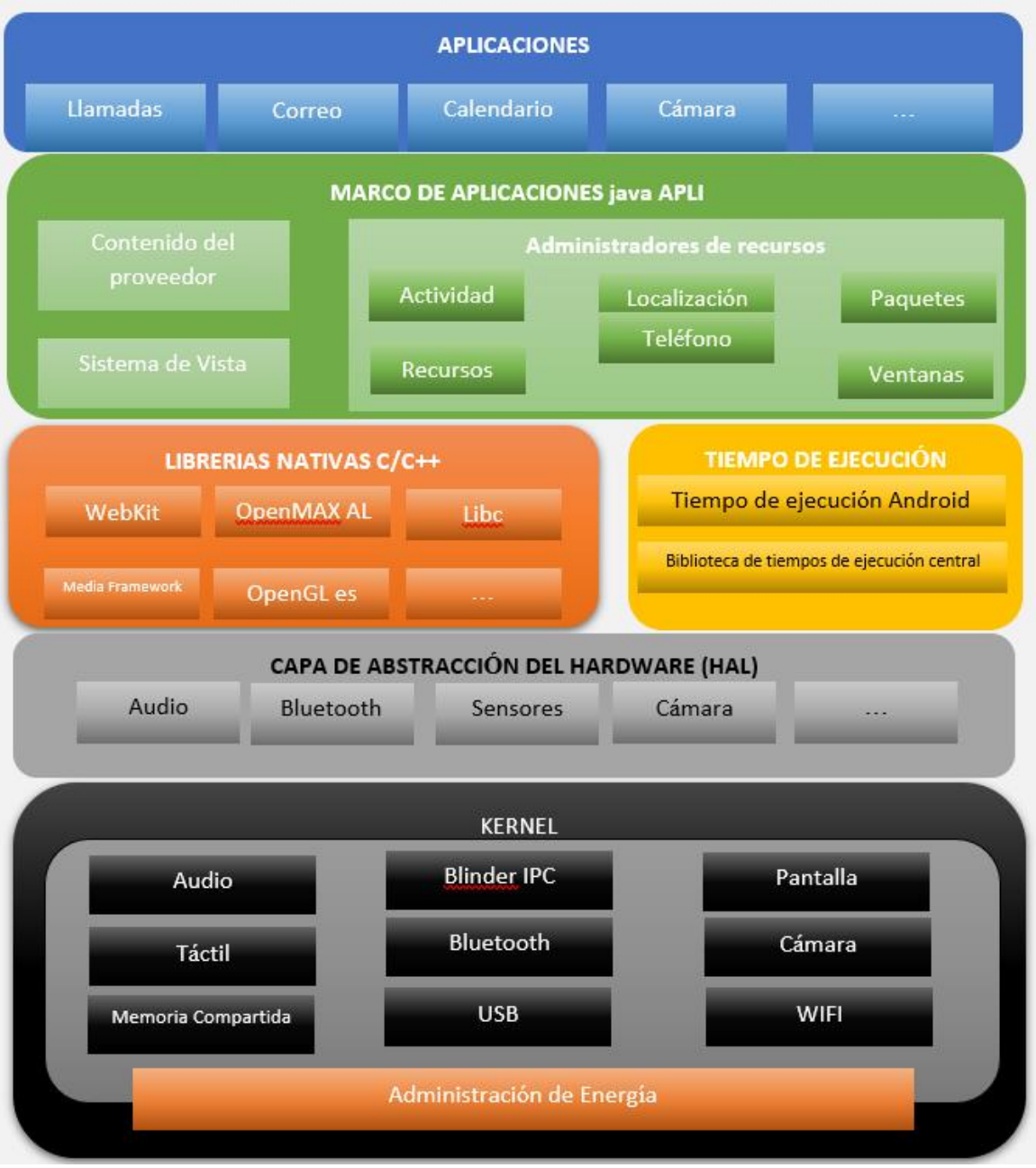

Figura 1.1-2 *Arquitectura actual de Android [12]*

La capa llamada HAL funciona como una interfaz entre el software y el hardware del sistema operativo Android, donde provee al sistema operativo de una plataforma de hardware en la que las aplicaciones corren. Las instrucciones de los controladores provenientes de la capa del *kérnel*,

pasan por la capa HAL convirtiendo estas solicitudes en unas instrucciones más genéricas para Android, permitiendo así a cualquier fabricante asignar una instrucción genérica para su *hardware*. Por ejemplo, un fabricante de pantallas para dispositivos con sistema operativo Android deberá hace una instrucción genérica en la capa de HAL "iniciar pantalla", pero la capa del *kérnel* la recibe "iniciar pantalla xxxx", donde xxxx es la versión de la pantalla del fabricante. En resumen, esta capa sirve para controlar el hardware de un dispositivo sin tener que cambiar las instrucciones para cada proveedor de hardware [12].

1.1.2 Niveles de seguridad de Android

A medida que va evolucionando el sistema operativo Android, también va mejorando la seguridad. En la versión de Android 1.0 llamada Apple Pie incluyen por primera vez opciones de seguridad en la pantalla de inicio del sistema operativo como un patrón de desbloqueo. Google le apunta a que la seguridad empieza en la capa de aplicaciones, donde se pueden evitar el uso de *software* malicioso. Tras avanzar en diferentes versiones, se nota el aumento del nivel de seguridad en la versión de Android 6.0 (Marshmallow), donde en la gestión de permisos granulares, los usuarios pueden observar y activar los diferentes permisos de las aplicaciones de forma manual. En esta versión se avanzó bastante en el tema de seguridad, ya que los usuarios pueden permitir o denegar los diferentes accesos que solicita una aplicación a ubicaciones donde se tiene información sensible u otros accesos en el sistema operativo. En esta versión aparecen las actualizaciones de seguridad mensual y también para estar a la vanguardia de los nuevos dispositivos de última generación, aumentó la seguridad en el acceso a estos equipos con un sistema biométrico llamado Fingerprint Applock.

En la nueva a versión del sistema operativo Android 7.0 llamada (Nougat), se realizó un cambio en el almacenamiento de datos del sistema donde se tendrían dos particiones: una partición principal para el almacenamiento de información y la otra partición que estaría deshabilitada y se tendría solo para recibir actualizaciones. La seguridad del sistema Android es mejorada con las actualizaciones especificas en seguridad. También aprovechan el *hardware* de los dispositivos para tener más controles en el acceso y en la manipulación. La protección en los componentes internos

del sistema operativo es muy importante para evitar errores y sean aprovechados para degradar la seguridad del sistema operativo [13].

Hay diferentes vulnerabilidades en las versiones del sistema operativo Android, donde pueden ser aprovechadas por un atacante informático para vulnerar la seguridad del sistema operativo. Estas vulnerabilidades se pueden encontrar en diferentes bases de datos, como son la de Vulnerabilidad Nacional (NVD) o la página Web http://www.cvedetails.com, donde se encuentran las vulnerabilidades del sistema operativo Android y su actualización para la corrección. Los aspectos más importantes en la seguridad del sistema operativo son los permisos, privilegios y control de acceso. Hay diferentes formas para vulnerar la seguridad del sistema operativo Android y una de ellas es cuando el dispositivo no tiene activada la depuración USB y se puede ejecutar un ataque de fuerza bruta para la extracción del código de bloqueo [14].

Vulnerability Trends Over Time

Year	# of Vulnerabilities	DoS	Code Execution	Overflow	Memory Corruption	Sql Injection	XSS	Directory Traversal	Http Response Splitting	Bypass something	Gain Information	Gain Privileges	CSRF	File Inclusion	# of exploits
2009	5	3								1					
2010	1	1	1												
2011	9	1	1		1					3	2	3			
2012	8	5	4	2							1				1
2013	7	1	2	2	2					1	1	3			
2014	13	2	4	1		1				1	2	2			1
2015	125	56	70	63	46					20	19	17			
2016	525	106	73	92	38					48	99	250			
2017	843	87	206	162	32			1		31	115	36			
2018	611	32	84	150	12	3	1	1		17	64	3			
2019	72	4	18	3	7					8	1	1			
Total	2219	298	463	475	138	4	1	2		130	304	315			2
% Of All		13.4	20.9	21.4	6.2	0.2	0.0	0.1	0.0	5.9	13.7	14.2	0.0	0.0	

Tabla 1.1-2 *Vulnerabilidades de Android [15]*

En la tabla 1.1-1 se observa el tipo y la cantidad de vulnerabilidades encontradas en el periodo comprendido entre el año 2009 y 2019. Es importante resaltar que estas vulnerabilidades son debilidades que tiene el sistema operativo y pueden ser aprovechadas por un atacante.

1.1.3 Sistema de archivos de Android

Dentro del sistema operativo Android se almacenan los archivos principales del sistema. Los archivos de configuración y las bibliotecas se almacenan en la memoria interna, como también las aplicaciones, algunos archivos de usuarios, las bases de datos de las aplicaciones; y por último en la memoria externa se almacenan copias de seguridad y datos del usuario. Es importante identificar estas rutas de almacenamiento a la hora de extraer una imagen bit a bit, para que no se pase por alto ninguna información que puede ser valiosa a la hora de hacer un análisis forense de la imagen digital extraída [16].

El sistema de archivos del sistema operativo de Android tuvo un cambio significativo a partir de la versión 2.3 llamada Gingerbread ya que en las versiones anteriores por lo general utilizaba el sistema de archivos YAFFA (Yet Anoter Flash System) que es un sistema de desarrollo óptimo para el almacenamiento Flash y actúa con un solo subproceso en ejecución y esto dificulta el procesamiento para sistemas de doble núcleo. El cambio a sistema de archivos en las últimas versiones fue por el Ext4 (EXTended) que son los estándares para el sistema de archivos Linux con mejoras en rendimiento, características y confiabilidad en la sincronización para no tener perdidas de datos [17][18].

Después de dar un enfoque general a la arquitectura de Android, se entiende que es un sistema operativo basado en Linux y su estructura de sistema de archivos no es una excepción [19]. Android emplea varias particiones para organizar los archivos y carpetas en el dispositivo. Principalmente existen seis particiones, sin embargo, se debe contar con la existencia de particiones adicionales que difieren entre un modelo y otro. A medida que se actualiza el sistema operativo también van cambiando su forma de almacenamiento en las particiones, por lo tanto, es importante conocerlas, tener claro su funcionamiento y su contenido:

- /boot: esta es la partición utilizada para iniciar el sistema operativo.
- /system: es la partición donde se almacena la memoria temporal ROM, tiene los formatos de archivos que se emplean en la mayoría de los dispositivos. Incluye las bibliotecas del sistema y aplicaciones preinstaladas.

- /recovery: partición donde se va a almacenar la recuperación del Sistema operativo. Contiene un sistema operativo liviano que contiene un *kérnel*, una memoria RAM que contiene varias utilidades de mantenimiento de bajo nivel.
- /data: también llamada UserData. En esta partición se almacena la información de los usuarios incluyendo la información de las aplicaciones.
- /cache: en la partición cache se almacenan los datos temporales de las aplicaciones y del sistema operativo.
- /sd card: esta partición es utilizada para almacenar información del usuario y también para guardar las fotos y videos tomados desde la aplicación de la cámara del dispositivo [20].

1.2 Métodos de cifrado

Estos son algunos de los métodos de cifrados conocidos a nivel mundial y que se tratarán en el trabajo.

1.2.1 3DES (Triple Data Encryption Standard)

También se conoce como DES_EDE3. Fue desarrollado para hacer mejoras al algoritmo de cifrado DES. Triple DES aumenta el tamaño de la llave de DES (56 bits) mediante la aplicación de tres veces el algoritmo DES con tres llaves diferentes. El tamaño de la clave combinada tiene entonces un total de 168 bits [21].

1.2.2 AES (Advanced Encryption Standard)

En 1997, NIST comenzó con el esfuerzo del desarrollo de AES, recibiendo varias propuestas y en 1995 cerró la convocatoria con 5 finalista: MARS, RC6, RIJNDAEL, SERPENT y TWOFISH, quedando RIJNDAEL como ganador [22]. Este es un cifrado simétrico de bloques que puede cifrar grupos de datos de 128 bits utilizando claves simétricas de 128, 192, o 256. AES sustituye al algoritmo DES, pero ha tenido ataques de fuerza bruta [23].

1.2.3 IDEA (International Data Encryption Algorithm)

Este algoritmo también utiliza el sistema de cifrado simétrico de bloques. Se desarrolla con el propósito de reemplazar el algoritmo DES. En su inicio fue desarrollado con el nombre de IPES (Improved Proposed Encryption Standard). El algoritmo IDEA funciona con bloques de 64 Bits y llaves de 128 bits, presenta 8 transformaciones iguales llamadas rondas y una trasformación de salida denominada media ronda. Este algoritmo hace mejoras en la seguridad ya que por ataque de fuerza bruta es difícil de romper [24]. Permite varias formas de gestión de los bloques: CBC (Cipher-block Chaining), OFB (Output Feedback) y CFB (Cipher Feedback) [42].

1.2.4 BLOWFISH

Es también un método de cifrado por bloques que puede ser usado como reemplazo de DES o IDEA. Usa una llave de longitud variable entre 32 y 448 bits. Fue diseñado en 1993 como una alternativa rápida y gratuita a los métodos de cifrado existentes en ese momento. Existen errores en algunas implementaciones [25].

1.2.5 TWOFISH

Es un método de cifrado por bloques con una llave que va desde 128 bits a 256 bits. Es rápido en unidades de procesamiento de 8 y 32 bits. Hizo parte de la convocatoria de NIST para reemplazar a DES y convertirse en AES [26].

1.2.6 THREEFISH

Es otro cifrado por bloques que puede ser retocado. Maneja tamaños de bloque de 256 bits, 512 bits y 1024 bits y la llave maneja los mismos tamaños que el bloque, siendo retocable el de 128 bits para todos los tamaños de bloque. Fue creado en 2008 con base en SHA3. Consiste en 3 operaciones: adición, XOR y 72 u 80 rotaciones (72 para bloques de 256 y 512 bits; y 80 para bloques de 1024). No hay criptoanálisis efectivo hasta el momento [27].

1.2.7 CAST

Pertenece a una clase de cifrados por bloques por cajas de sustitución (S-boxes). Similar a DES, utiliza una serie de rondas de sustitución para crear confusión y difusión [28].

1.2.8 ICE ENCRYPT (Information Concealment Engine)

Utiliza el mismo sistema Feistel de DES, que consiste en un cifrado por bloques reversible, es decir, que utiliza el cifrado y el descifrado con la misma técnica invirtiendo sólo las subclaves. Este método de cifrado toma bloques de 64 bits y los divide en 2 mitades. La mitad derecha con un clave de 60 bits es unida con el resultado XOR de la mitad izquierda hasta completar todas las rondas [29].

1.2.9 R6

Es un cifrado por bloques que fue finalista en la convocatoria de NIST para crear AES. Fue una evolución de su antecesor RC5. Fue creado en 1995 por Ronald Rivest y consiste en la operación de 3 operaciones matemática: adición, sustracción y multiplicación. Esta última se usa para crear difusión y para que aumente la rapidez del cifrado. Para las rotaciones opera un XOR en cada una de las operaciones. Utiliza llaves de 128 bits, 192 bits, 256 bits, 384 bits y 512 bits [30].

1.2.10 SALSA20

Es un método de cifrado por chorro (*stream*) diseñado en 2005. Según el número de rondas que tenga, se le da el nombre de Salsa20/20, Salsa20/12 y así sucesivamente. Salsa20/20 es más rápido que el actual AES. Utiliza las operaciones de adición, XOR y una rotación cada una de bloques de 32 bits. No se conoce criptoanálisis de Salsa20/20, pero sí para los de menos rondas [31].

1.2.11 SERPENT

Es un método de cifrado por bloques que terminó de segundo en la convocatoria de NIST para determinar el nuevo AES. Utiliza 32 rondas uniformes con una llave de 128 bits. La última ronda utiliza 2 llaves, siendo 33 en total [32].

1.2.12 SHACAL2

Fue uno de los 4 métodos escogidos en la NESSIE. Es un cifrado por bloques de 256 bits que se ayuda de SHA256. Acepta llaves de 128 bits hasta de 512 bits. Es rápido y muy seguro [33]. El algoritmo incluye el texto plano a la función de compresión como la variable de cadena y luego la llave como bloque del mensaje. El texto de 256 bits se divide en palabras de 32 bits. Luego la función de estado actualiza a 8 palabras de 32 bits. Después la salida es de 8 palabras de 32 bits luego de 64 pasos [34].

La idea entonces con el cifrado de la imagen es proteger la información, atendiendo la normativa de protección de datos.

Sin embargo, en la metodología tradicional de extracción de imágenes de forma alámbrica, el cifrado no es un proceso inherente a la toma de la imagen que se extrae. Este proceso acarrea un tiempo adicional. Pero muchas veces se toma en consideración que el dispositivo de almacenamiento se cifre con alguna herramienta como TrueCrypt [35] [36], ésta descontinuada desde 2014.

Consultados algunos grandes fabricantes de software y para la adquisición de imágenes forenses, tanto a computadores como a móviles, no se menciona que se cifra la imagen extraída [37] [38] [39] [40], ya que no sólo se demora un poco más la realización del análisis forense, sino que también se presupone que la cadena de custodia está asegurada. Pero algunos de los fabricantes de hardware para extracción de imágenes forenses sí utilizan cifrado de la información como el método XTS-AES256 [41].

1.3 Técnicas vigentes para las imágenes incrementales, compresión e integridad

1.3.1 Copias de seguridad

Las copias de seguridad son una parte fundamental de todo proyecto IT. Las copias de seguridad son esenciales para prevenir la pérdida de datos si se produce algún imprevisto. No necesariamente

tienen que hacerse de forma manual, sino que, por el contrario, pueden hacerse de manera automática y con alguna periodicidad y que pueden ser tanto a nivel de archivo (*file level*) o basados en imágenes (*image based*):

- Copias de seguridad a nivel de archivos: los *backup* de archivos se hacen en forma selectiva, de tal modo que el usuario puede escoger los archivos o directorios que quiera respaldar.
- Copias basadas en imágenes: se pueden ejecutar copias de seguridad basados en imágenes a nivel de *bit*. La imagen se refiere a la copia exacta de un medio de almacenamiento en concreto, como un disco, una memoria *flash*, un medio óptico, entre otros. Se crea una copia idéntica con todos los datos seleccionados.

La copia de seguridad basada en imagen tiene la ventaja que, en caso de pérdida total de los datos, sólo debe restaurarse la última imagen tomada en el sistema productivo o en otro.

Las copias de seguridad pueden ser de varios tipos:

- Copia completa: se realiza una copia completa de los archivos o del disco duro (o dispositivo de almacenamiento) completo.
- Copia diferencial: se copian todos los datos, sea a nivel de archivos o de *bits* que se han modificado o añadido desde la última copia completa. Si se hace necesario restaurar el sistema, sólo se requiere la copia completa subyacente y la última copia parcial.
- Copia incremental: todos los datos a nivel de archivo o de *bits* que se han creado o modificado en la última copia parcial o total se transfieren al medio de almacenamiento seguro. Como cada copia incremental se superpone a la copia parcial anterior, se reduce el rendimiento por cada copia, así como el espacio de almacenamiento, comparándolo con la copia diferencial. La desventaja de este método es que, en el caso de hacer una restauración, se necesita de cada una de las copias parciales. Si falta una copia, no es posible la restauración. [42].

En una imagen incremental, todo el contenido copiado está encapsulado en un único archivo y que es administrado por el programa que creó la imagen. Pero si el sistema fue contaminado por cualquier tipo de *malware*, estos serán incluidos en la siguiente imagen incremental que se haga,

porque son archivos nuevos o modificados. Por lo tanto, la imagen ya no estará sana y limpia, sino alterada [43].

La herramienta ZBACKUP está basado en las ideas de **rsync** y realiza copias de seguridad incrementales. Este programa soporta cualquier formato, por lo que se le puede añadir cualquier tipo de archivo e inclusive imágenes de disco RAW. Cuenta con las siguientes características:

- Compresión paralela LZMA o LZO de los datos almacenados.
- Cifrado AES-128.
- Posibilidad de borrar datos de copia de seguridad antiguos.
- Uso de *rolling hash* de 64 bits.
- El repositorio consta de archivos inmutables.
- Está escrito en lenguaje C++.
- Posibilidad de intercambiar datos sin recompresión [44].

Rsync es una herramienta especialmente presente en los sistemas Linux y Unix para transferir archivos de forma incremental [45]. Es una línea de comando rápida y versátil que sincroniza archivos y carpetas entre dos lugares sobre un s*hell* remoto. Rsync puede ser usado para datos espejo, *backups* incrementales, copiar archivos entre sistemas y como un reemplazo de comandos de uso común como **cp** (copia de archivos), **scp** (copia remota de archivos por medio de **ssh**) y **sftp** (transferencia de archivos por **ssh**). Se puede utilizar con varias opciones en la sentencia de ejecución, como, por ejemplo **-a** para conservar el grupo, los permisos y la autoría de la jerarquización de carpetas y archivos; **-z**, para comprimir los datos que se van a sincronizar; **--delete** para borrar los datos de la fuente que se copiaron; entre otros [56].

1.3.2 Compresión

Para trabajar en la compresión de datos o imágenes en un sistema operativo Linux o Android, existen varias herramientas para hacer estas compresiones:

- ZIP: es una muy buena opción si lo que se quiere es que el comprimido sea portable a otros sistemas operativos. Si el archivo a comprimir se llama prueba, la sintaxis es zip prueba.zip prueba.

- GZIP: se utiliza sobre todo para la portabilidad del archivo comprimido entre los sistemas operativos Unix y Linux. La sintaxis para utilizarlo con el archivo a comprimir llamado prueba es, gzip prueba. El resultado es un archivo comprimido llamado prueba.gz. Esta herramienta tiene ligeramente mejor tasa de compresión que la herramienta Zip.
- BZIP2: también se utiliza para la portabilidad entre Unix y Linux. Pero no se utiliza tanto porque la velocidad de compresión es más lenta que las anteriormente mencionadas y la tasa de compresión no es muy buena. El archivo comprimido es más grande que los resultantes de utilizar Zip y Gzip. La sintaxis para comprimir un archivo llamado prueba es, bzip2 prueba y el archivo resultante es prueba.bz2.
- XZ: es el formato preferido para archivos grandes, ya que su tasa de compresión es mucho mejor que cualquiera de las anteriores, pero su velocidad de compresión es lenta. La sintaxis para comprimir el archivo prueba es, xz prueba y el archivo resultante es prueba.xz.
- RAR: no es común usarlo en sistemas de tipo Linux o Unix y el tiempo de compresión no es tampoco el mejor. La sintaxis para comprimir el archivo prueba es rar a prueba.rar prueba [47].
- ZPAQ: es un formato estándar basado en Paq, que logran mayores tasas de compresión a cambio de utilización de bastante CPU, de memoria [48] y tiempo. La sintaxis es zpaq c prueba.zpaq prueba.
- TAR: esta herramienta es bastante práctica, porque se puede empaquetar y comprimir y se puede utilizar el algoritmo de cualquiera de las herramientas mencionadas anteriormente. Se debe tener en cuenta que con la opción “c” se empaqueta y con la opción “x” se desempaqueta o se extrae. Si se empaqueta y comprime con tar y utilizando el algoritmo de gzip, entonces para el archivo prueba sería tar czvf prueba.tar.gz prueba. La opción “v” es que trabaje en modo *verbose* (que muestre todo lo que está haciendo mientras empaqueta y comprime) y la opción “f” indica el archivo con el que se trabajará. Para empaquetar y comprimir con el algoritmo XZ, la sintaxis sería tar cJvf prueba.tar.xz prueba [47].

1.3.3 Integridad

Siendo uno de los pilares de la seguridad informática en conjunto con la disponibilidad y la confidencialidad, la integridad es la que garantiza que los datos no han sido modificados desde su creación sin autorización [49].

Por lo tanto, se entiende como integridad de la información cuando se considera que es exacta, completa, homogénea, sólida y coherente con la intención de los creadores de los datos que la conforman. La integridad va ligada al propio dato y no de donde se almacena; se obtiene cuando se impide eficazmente que el contenido de una información, de un sistema o de un proceso se vea accidental o intencionalmente [50].

La integridad de los datos garantiza que los usuarios autorizados pueden modificar o acceder a la información. Un ataque a esta integridad puede ser para obtener ganancias financieras o simplemente para invalidar los datos, que no sirvan para nada. De hecho, existen varias técnicas corporativas para preservar la integridad de la información como, por ejemplo, el cifrado de datos confidenciales o utilizar autenticación multifactor [51].

Los disparadores más comunes que atentan contra la integridad de los datos son:

- Modificación de los permisos y privilegios de acceso.
- Imposibilidad de rastrear el uso de contraseñas privilegiadas, en especial cuando es compartido.
- Errores del usuario final que afectan los datos de producción.
- Aplicaciones vulnerables a la introducción de códigos ocultos (como los *backdoors*).
- Procesos de control de cambios y acreditación deficientes o no desarrollados plenamente.
- Fallas en la configuración de *software* y dispositivos de seguridad.
- Aplicación de parches en forma incorrecta o incompleta.
- Conexión de dispositivos no autorizados a la red corporativa.
- Uso de aplicaciones no autorizadas en dispositivos conectados a la red corporativa.
- Segregación de funciones (SoD) inadecuada o no aplicada [52].

Para preservar la integridad de los datos, es conveniente utilizar el método *Hash*, que es una función de cifrado con un algoritmo matemático que transforma cualquier bloque arbitrario de datos en una nueva serie de caracteres con una longitud fija. Sin importar la longitud de los datos de entrada, el valor *hash* de salida siempre tendrá la misma longitud [53].

Hace algunos años se reportó que una de las técnicas de *hash* llamada SHA-1 fue atacada y por lo tanto NIST realizó una convocatoria pública para seleccionar el algoritmo SHA-3 para el nuevo estándar. Se presentaron los algoritmos Blake, Grøstl, JH, Keccak y Skein y el ganador fue Keccak para ser el nuevo estándar SHA-3 [54].

En los últimos años, la mayoría del mundo digital fue forzado a moverse del defectuoso SHA-1 a SHA-2 a causa de los incrementales exitosos a los *hashes.* Cualquiera que estuviera involucrado en estas migraciones, sabe que muchas de las horas de investigación y trabajo llevaron a interrupciones mínimas de las operaciones. En muchos casos se admite que el proceso se preparó de la mejor manera y después de apagar el interruptor esperamos por lo mejor. Las migraciones de *hash* no son de los mejores procesos para hacer. ¿Por qué no se ha migrado lo suficiente en estos momentos a SHA-3? Porque hasta ahora no hay mucho *software* ni *hardware* que lo soporten. Además, SHA-3 no es un estándar oficial todavía. SHA-2 provee protección suficiente por ahora. Y algo bien importante, los que lo han probado dicen que es más lento que SHA-2. SHA-3 es lento en *software*, en *hardware* es un poco más rápido que SHA-1 y SHA-2 y generalmente se utilizan rutinas de cifrado en *hardware* y va en aumento; se ha demostrado que SHA-512 es más rápido que SHA-3 [55].

Se debe tener en cuenta la premisa que, a mayor número de *bits* de salida, es mayor la seguridad, pero no hay que descuidar la velocidad de cifrado. MD5, ya no se tiene en cuenta algunas veces, porque ha sido comprometido por ataques. Entonces, dentro de la familia SHA-2 se encuentran SHA-256 y SHA-512, que han reportado muy buenas velocidades de cifrado y seguridad [56].

1.3.4 Tratamientos de datos sensibles y privados

La ley estatutaria 1581 de 2012 de la República de Colombia trata de las disposiciones generales para la protección de datos personales [57].

El decreto 1377 de 2013 de la República de Colombia reglamenta parcialmente la ley 1581 de 2012 que también trata sobre las disposiciones generales para la protección de datos personales, autorizaciones, políticas de tratamiento y transferencias y transmisiones internacionales de datos personales, como también de la responsabilidad en el tratamiento de esos datos [58].

1.4 Validación de la metodología propuesta bajo un ambiente controlado

Para la validación de la metodología propuesta en este trabajo, se necesitan varios elementos para llevarla a cabo, que serán explicados a continuación.

1.4.1 Ambiente o entorno controlado

Estos ambientes son comúnmente llamados *sandbox,* es un entorno de prueba aislado que se utiliza para la ejecución de aplicaciones o archivos sin afectar el sistema de base. Los desarrolladores de código lo utilizan para probar programas y los profesionales de ciberseguridad lo utilizan para probar *software* posiblemente malicioso [59].

La utilidad de un ambiente controlado es variada, dependiendo del objetivo de la prueba. Por ejemplo, puede ser muy útil para:

- Instalar y usar de forma segura programas que no son fiables y que pueden contener código malicioso.
- Usar programas que pueden comprometer la seguridad del sistema operativo.
- Crear un entorno de pruebas para ejecutar código creado [60].

El objetivo entonces de los ambientes controlados o *sandbox* es mejorar la seguridad en muchos aspectos, aislando una aplicación o una ejecución de aplicaciones para evitar que *malware*, intrusos o recursos del sistema interactúen con la aplicación protegida. Cuando un desarrollo no se quiere

que sea tocado por influencias externas, se puede aislar en una máquina virtual, al cual se le llama microvirtualización [61].

1.4.2 Máquinas virtuales

Una máquina virtual es un *software* que permite emular el funcionamiento de un computador dentro de otro computador, por medio de un encapsulamiento que aísla a ambos. Esa emulación permite tener todos los componentes necesarios de un computador, como son el procesador, la memoria RAM, el disco duro, la tarjeta de red y de gráficas, entre otros [62].

Unos de los usos más frecuente de las máquinas virtuales es la de probar diferentes sistemas operativos, programas o configuraciones con total seguridad para el computador real, ya que, si falla en la máquina virtual, no lo afecte. Como estas máquinas virtuales no son "conscientes" que lo son, se comportan igual que si fueran reales [63].

Las tecnologías de virtualización son muy buenas, pero tienen un problema. Se necesita cierto nivel de *hardware* para cederles parte de los recursos de la máquina donde se vaya a instalar las máquinas virtuales, sobre todo recursos de procesador, memoria y almacenamiento. Como es difícil obtener el mismo nivel de rendimiento de una máquina local en una máquina virtual, hay características a tener en cuenta para tener una mejor experiencia, como son:

- Asegurarse que Intel VT-x o AMD-V esté disponible y activado. Algunos computadores no tienen disponible esta característica y por lo tanto no es posible que funcione una máquina virtual.
- Se deben crear en las máquinas virtuales discos de tamaño fijo y no dinámico. Esto hace que se tenga más rendimiento y menos fragmentación. Sin embargo, existen aplicaciones críticas instaladas en las máquinas virtuales, los cuales exigen que el disco sea de crecimiento dinámico.
- Elegir la unidad de almacenamiento más rápida. Generalmente se utiliza una unidad de disco secundario de mayor capacidad, pero a veces es mejor utilizar un dispositivo SSD por ser de transferencia de información mucho más rápida que un disco duro.

- Instalar herramientas de *software* adicionales, según la aplicación de máquina virtual a utilizar.
- Las máquinas virtuales utilizan mucha memoria RAM. Mínimo se debe asignar 2 GB de RAM para sistemas Windows y Linux.
- Se debe asignar suficiente cantidad de núcleos de procesamiento. Mientras más máquinas virtuales se pongan en ejecución, más procesamiento utilizará el equipo anfitrión.
- Igualmente, la configuración de video puede mejorar el rendimiento de la máquina virtual, inclusive ajustando la cantidad de memoria de video a utilizar.
- Como el antivirus no puede ver el interior de la máquina virtual, se debe colocar una excepción en el antivirus de no escanear los directorios de la máquina virtual y así aumentar el rendimiento [63].

1.4.3 Redes inalámbricas (WiFi)

WiFi es un conjunto de estándares que permiten la conexión inalámbrica a una red de datos o Internet con dispositivos adaptados a WiFi, como por ejemplo los *smartphones* [64].

WiFi es una marca de la WiFi Alliance, anteriormente WECA (*Wireless Ethernet Compatibility Alliance*), creada en 1999 por 3Com, Airones, Intersil, Lucent Technologies, Nokia y Symbol Technologies. Esta organización está encargada que los equipos cumplan con la norma IEEE 802.11 [65].

Este estándar IEEE 802.11 incluye una familia de especificaciones que comenzó en la década de 1990. Acá se mencionan los estándares de esa familia:

- 802.11-1997: es el primer estándar, que admite velocidades de hasta 2 Mbps en la frecuencia de 2.4 GHz. Tiene un alcance de 20 metros.
- 802.11b: fue lanzado en 1999 y admite velocidades de hasta 11 Mbps en la frecuencia de 2.4Ghz.
- 802.11a: fue aprobado en 1997, pero salió al mercado después del estándar 802.11b. Éste admite velocidad de hasta 54 Mbps y opera en la frecuencia de 5 Ghz.
- 802.11g: fue aprobado en 2003 y es sucesor de 802.11b, capaz de alcanzar velocidad de hasta 54 Mbps en la frecuencia de 2.4 GHz.

- 802.11n: es el primer estándar que especifica las antenas MIMO en las bandas de 2.4 GHz y 5 GHz. Admite velocidades hasta de 600 Mbps.
- 802.11ac: actualmente muchos de los enrutadores domésticos son compatibles con el estándar 802.11ac en la frecuencia de 5 GHz y con múltiples antenas de envío y recepción. Este estándar admite velocidades hasta de 3.46 Gbps.
- 802.11ad: fue aprobado en 2012. Admite velocidades hasta de 6.7 Gbps en la frecuencia de 60 GHz, pero con sólo hasta de 3.3 metros de distancia [66].
- 802.11ah: se conoce como White WiFi y es una modificación de 802.11-2007. Está diseñado para trabajar en las bandas en blanco del VHF y UHF, comúnmente asociadas a las transmisiones de televisión. Opera en las bandas entre 54 MHz y 790 MHz.
- 802.11af: se conoce como HaLow. Trabaja en la banda de 900 MHz. Consume poca energía y la intención es que trabaja para IoT [67].

Están pendientes de aprobación los estándares 802.11aj, 802.11ak, 802.11ax, 802.11ay, 802.11az y 802.11b [66].

1.4.4 Kali Linux

El ecosistema GNU/Linux consiste en varios sistemas operativos Linux, cada uno para necesidades específicas y trabajos numerosos, tanto domésticos como súper complejos.

Kali Linux es una distribución de Linux usada para la seguridad informática y no para la realización de ilegales. Kali está basado en Debian, que es otra distribución de Linux. Actualmente es mantenida por Offensive Security Ltd. que desarrolló la distribución a partir de la reescritura de BackTrack, que fue predecesora de Kali. Tiene preinstalados una gran cantidad de programas relacionados con la seguridad informática, entre ellas NMap (escáner de puertos), Wireshark (sniffer), John The Ripper (crackeador de contraseñas) y Aircrack-ng (pruebas de seguridad en redes inalámbricas).

Este sistema operativo es muy fácil de instalar, pero no tan fácil de manejar, aunque hay mucha gente que comparte conocimiento para el manejo del mismo [68].

Sin embargo, Kali es un sistema operativo que puede funcionar perfectamente en vivo (*live*), ejecutarse desde una memoria USB y actualizarla periódicamente, ya que no es frecuente que se grabe algo en el disco que tenga instalado Kali Linux. Tiene la flexibilidad de ejecutar una máquina Kali en cualquier lugar y en cualquier red [69]. Hasta el momento la última actualización es la 2019.2 [68].

Una de las particularidades que tiene Kali Linux con relación a varias de las otras distribuciones de Linux, es que se maneja con usuario *root* siempre. Esto se debe a que las herramientas de seguridad que provee este sistema operativo necesitan privilegios a ese nivel. Esta funcionalidad es de manejo cuidadoso, ya que, si se necesita ver las vulnerabilidades de un *software* en especial, un atacante podría también tomar el control del equipo [70].

1.4.5 Android Debug Bridge (ADB)

Es una herramienta utilizada para establecer una comunicación entre una estación y los dispositivos móviles con sistema operativo Android. Además, por medio de esta herramienta se pueden ejecutar diferentes tareas, instalar aplicaciones, deshabilitar el micrófono del dispositivo, entre otros. Por otra parte, es una aplicación que establece la comunicación cliente-servidor y está comprendido por tres elementos:

- El cliente: este se encarga de enviar los comandos desde una estación hacia el dispositivo móvil.
- El *daemon*: este ejecuta los comandos en un segundo plano del dispositivo móvil.
- El servidor: es la estación de trabajo que se encarga de establecer la comunicación con el dispositivo móvil.

La comunicación con el dispositivo móvil puede ser de forma cableada (USB) o de forma inalámbrica. Para fijar la comunicación de modo cableado se debe activar en el sistema operativo Android la acción USB *debugging* y para fijar la comunicación WiFi debe haber una comunicación establecida entre la estación de trabajo y el dispositivo móvil por medio de una red. El puerto para la comunicación del cliente -servidor en modo WIFI es por defecto el 5555 [71].

1.4.6 Transacciones distribuidas

Un sistema distribuido es aquel cuando sus componentes pueden estar en varios dispositivos y coordinan sus acciones a través de la red. Se caracterizan por tener un funcionamiento concurrente, no existe una definición global del tiempo y las fallas son independientes. Los sistemas distribuidos están asociados a una amplia de gama de aplicaciones desde servicios centralizados hasta alta computación, desde sistemas de cómputo pequeños hasta sistemas de gran procesamiento [72].

Los recientes avances tecnológicos en la miniaturización de dispositivos y redes inalámbricas han llevado a la integración de dispositivos portátiles pequeños hacia un sistema distribuido. Estos dispositivos incluyen computadoras portátiles, teléfonos móviles, teléfonos inteligentes, relojes inteligentes, dispositivos con GPS, entre otros. La portabilidad de estos dispositivos, junto con su capacidad de conectarse a redes en diferentes lugares, hace posible la computación móvil. La computación móvil se define como la ejecución de tareas de computación, mientras que el usuario está en movimiento o en lugares distintos de su entorno habitual de trabajo [72].

Para poder tener control de varios dispositivos, un agente debe ser enviados a esos equipos y aquí entra el concepto de distribución. Un agente transaccional está compuesto por subagentes (bloques de programación) de enrutamiento y de manipulación. Cuando este agente llega al dispositivo, los objetos son manipulados según lo programado en el módulo de manipulación. Estos objetivos obtenidos tienen que ser enviados a un repositorio donde el agente, previamente programado, dice a dónde debe enviarse. La programación para que el agente visite esos dispositivos es programado desde el bloque de programación de manipulación [73].

Por lo tanto, debido a las características intrínsecas de los entornos inalámbricos y distribuidos, resulta de especial interés la capacidad de los agentes software para reaccionar y adaptarse autónomamente al estado del entorno, permitiendo así ajustar su actuación a los diferentes estados de la red, el consumo de energía, el contexto concreto de ejecución, el comportamiento del usuario, y las características técnicas del dispositivo en el que se ejecuten. Cualquier petición realizada por el cliente y la respuesta del servidor asociado con la aplicación se comunica a través del agente específicamente creado para ello. En este caso el dispositivo móvil tiene que estar

conectado con tantos agentes como servicios estén siendo accedidos. A esto se le llama arquitectura cliente/agente/servidor [74].

En general, las transacciones distribuidas sólo ocurren en entornos distribuidos. En el caso de este trabajo, un script será distribuido hacia varios dispositivos móviles, cuando sea necesario, para obtener una imagen digital de la memoria de esos dispositivos y llevada a un repositorio.

2. Metodología

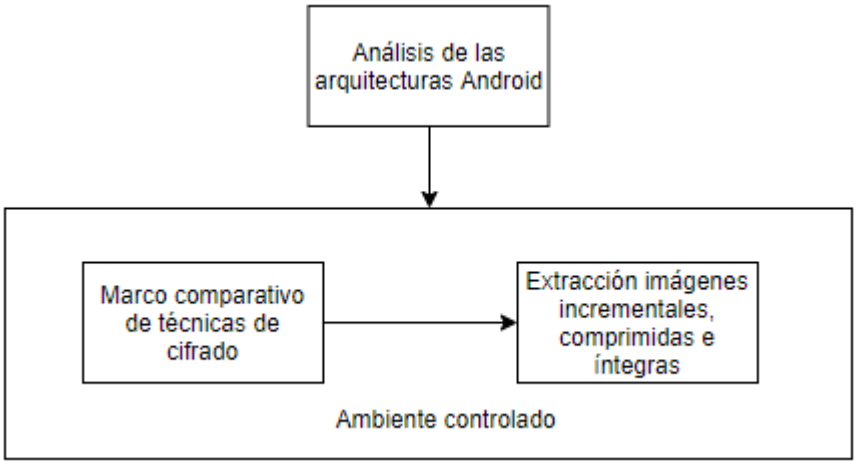

Figura 2-1 *Metodología desarrollada*

En la figura 2-1 se propuso esta metodología de investigación para cada uno de los objetivos específicos de este trabajo. Los tres primeros procesos de la figura, de izquierda a derecha fue trabajado en ambientes individuales, para que en el ambiente controlado se validara la propuesta completa.

La metodología utilizada en esta investigación para probar los diferentes métodos y en el ambiente controlado, es utilizar la simulación de un entorno real, hacer diferentes análisis exhaustivos, no comprometer dispositivos físicos, la independencia de cada máquina virtual y la facilidad de manejo de los sistemas operativos. Para la simulación del ambiente real y la obtención de datos fidedignos al momento de la extracción de la imagen digital forense, se cuenta con interfaces de red inalámbrica físicas (AP y tarjetas de red WIFI) independientes para cada máquina virtual.

2.1 Análisis de las arquitecturas de Android a utilizar en la investigación

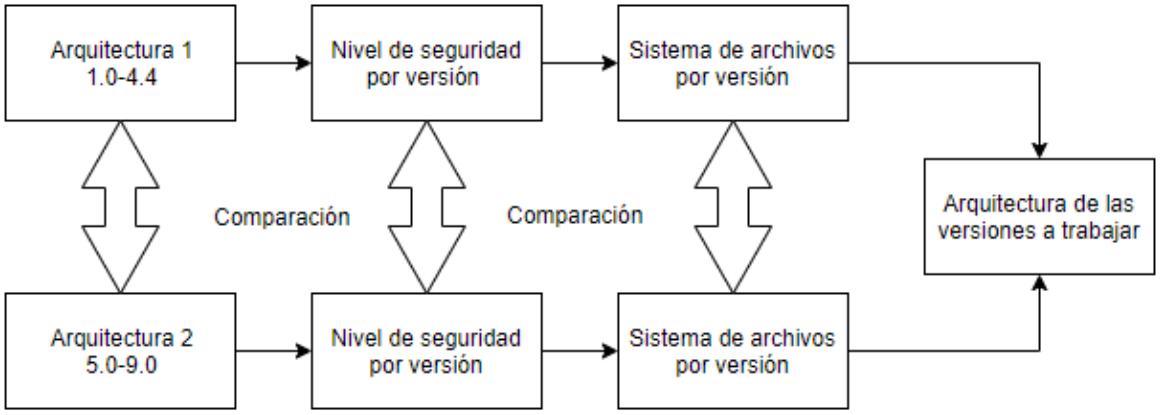

Figura 2.1-1 *Diagrama de bloques del proceso de selección de arquitectura*[1]

En la figura 2.1-1 se representa el análisis de las arquitecturas de Android. En forma general, se detectaron dos arquitecturas y que difieren entre ellas por los sistemas de archivos y el nivel de seguridad. Este análisis es importante para el desarrollo de la aplicación que tomará la imagen y la ubicación de los archivos cuando la imagen vaya a ser analizada. El producto final de este análisis es la lista de versiones a trabajar en el ambiente controlado.

Básicamente y teniendo en cuenta que en el marco teórico fue cubierta la mayor parte de la determinación de las arquitecturas de Android, se pueden diferenciar dos arquitecturas: la que va desde la primera versión de Android (1.0, Apple Pie) hasta la última versión de Kit Kat (4.4); y la que salió con la versión de Lollipop (5.0) hasta la versión 9 (Pie).

Para probar la premisa anterior, se revisó mucha de la literatura especializada en Android y se compararon tanto la arquitectura, el nivel de seguridad y el sistema de archivos entre las diferentes versiones. Igualmente, al tratar de generar un código de aplicación en Android Studio y dependiendo de la versión del mismo, se hace una validación de la versión de Android al que se quiere enviar ese código en forma de App. Para las versiones de Android probadas anteriores a Kit

[1] Este diagrama es de realización propia de los autores de este trabajo, basados en los materiales usados para la prueba

Kat inclusive, no había mensaje de advertencia; entre las versiones de Kit Kat y siguientes, tampoco mostraba la advertencia que no se usara el código para versiones anteriores.

Así entonces, se hicieron las identificaciones que se mencionan a continuación.

2.1.1 Identificación de la arquitectura Android

Esta identificación de la arquitectura de Android se logró con la teoría recopilada y que se puede resumir con las gráficas mostradas en la Figura 1.1-1 y Figura 1.1-2.

Se revisaron todas las capas que conforman el sistema operativo en todas sus versiones, pero poniendo énfasis desde Kit Kat, que, aunque ya no es compatible con varias de las aplicaciones actuales, hasta Marshmallow (versión 6), Nougat (versión 7) y Oreo (versión 8), siendo estas últimas, las versiones más utilizadas en el mundo a mayo de 2019 [75].

Por lo tanto, se identificaron todas las capas que conforman la arquitectura, incluyendo la HAL, para ser tenidas en la extracción de la imagen posteriormente.

2.1.2 Identificación de los niveles de seguridad de Android

En esta etapa de identificación, se toma en cuenta los niveles de seguridad de algunas versiones de Android, teniendo en cuenta lo descrito en las arquitecturas de ese sistema operativo.

Existe una base de datos que puede ser consultada por internet, Mitre Corp [14], que periódicamente se está actualizando mostrando las diferentes vulnerabilidades de Android y cuántos ataques ha recibido por año. Esta información es muy útil para el caso de esta investigación, ya que se observa que Android tiene una vulnerabilidad bastante apreciable de ejecución de código. Este código no necesariamente tiene que ser malicioso, porque en el caso de esta investigación, ese código será para tomar y extraer una imagen forense del dispositivo.

Se hicieron comparaciones con otras fuentes y la capacidad de poder ejecutar un código en Android es alta, como lo menciona Bilić [76] y una buena oportunidad para la investigación.

2.1.3 Identificación del sistema de archivos de Android

Se pudo evidenciar en la literatura consultada y sobre todo con la misma Android, que el sistema de archivos no ha cambiado sustancialmente en todas las versiones más usadas del sistema operativo. Por lo tanto, ya se tiene claro cuáles son sus nombres y qué tipo de información se encuentra en cada una de las particiones y ficheros.

Lo más importante en la extracción de la imagen, es que no sólo los datos estén ahí, sino la metadata, que es la que identifica un recurso específico [77].

2.2 Marco comparativo de técnicas de cifrado

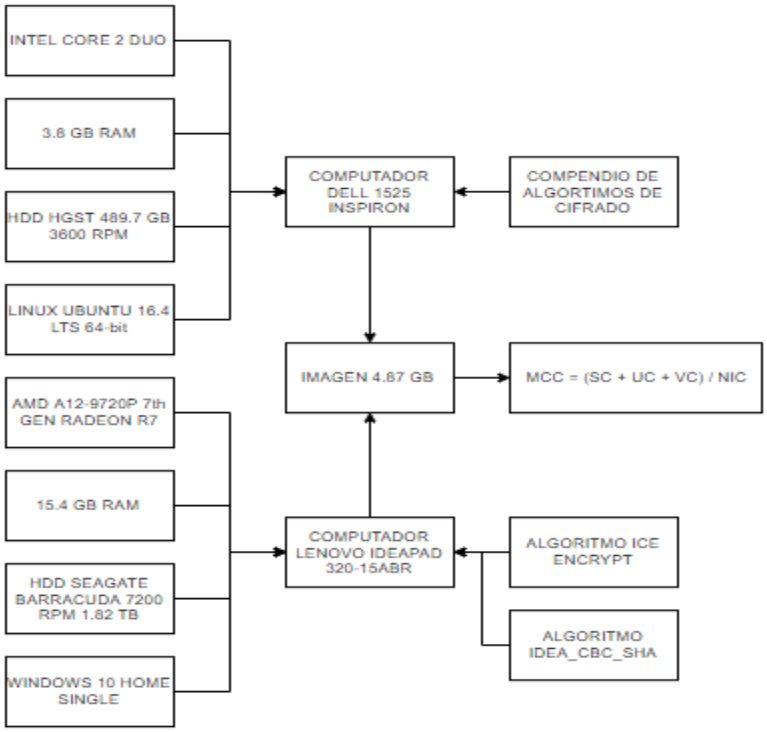

Figura 2.2-1 *Diagrama de bloques de elementos utilizados en la prueba de selección de método de cifrado*[2]

En la figura 2.2-1 se presenta el diagrama de lo que se utilizó para hacer el comparativo entre técnicas de cifrado y poder llegar a una conclusión del que se utilizará para la metodología. Se están

[2] Este diagrama es de realización propia de los autores de este trabajo, basados en los materiales usados para la prueba.

representado los equipos y características, la imagen que ayudó a hacer la comparación, los algoritmos de cifrado y un producto final que es la métrica que ayudó a la selección de un método de cifrado. Se tiene la premisa que un proceso añadido a la extracción puede afectar el tiempo y la rapidez, pero se quiere probar que, en el caso de necesitarse, cuál sería el mejor método de cifrado. Siempre se debe tener en cuenta la seguridad de la información.

En la actualidad, asegurar la información es de suma importancia, ya sea al enviar la información a través de una red insegura o al momento de almacenar los datos en un dispositivo. No es preciso confiar que todo canal de WiFi está asegurado o que está cifrado y por lo tanto al momento de enviar una información, existe el riesgo que dicha información pueda ser interceptada por una persona diferente al receptor original, por lo que se debe mejorar la seguridad [78].

Existen diferentes formas para asegurar la información y la más asertiva es la utilización de un método de cifrado, que consta de hacer el mensaje incomprensible a personas ajenas. Cuando alguien diferente al receptor intenta acceder a la información y no cuenta con el permiso o llave de cifrado, no comprenderá la información original del mensaje. Entre las diferentes formas de asegurar la información están los métodos de cifrado simétrico con sus diferentes algoritmos como es IDEA, que es muy utilizado a nivel mundial por el software PGP (*Pretty Good Privacy*) [79].

Hay 2 formas de cifrado. El simétrico, que consiste en tener una misma llave para cifrar y descifrar el mensaje y la otra forma es asimétrica, que cuenta con una llave pública y sólo el receptor al que va dirigido el mensaje, podrá descifrar con una llave privada. En este trabajo se usa el cifrado simétrico, el costo computacional es menor, reflejado en su alta velocidad. Adicionalmente, se trabajará con una llave de longitud grande y confiable para asegurar que la información es correcta [69]. Cifrar la información extraída en una imagen para enviarla por WiFi, no es considerada como una técnica antiforense [78].

Las variables a tener en cuenta a escoger un método de cifrado son velocidad, tamaño, seguridad, uso en la comunidad y número de imágenes de 48 GB cifradas en 30 minutos.

Se utilizó un computador Dell Inspiron 1525, con procesador INTEL CORE 2 DUO CPU T5750 2.00 GHZ X 2 Centrino, 3.8 GB de RAM, disco duro HGST de 489.7 GB de 3600 RPM y sistema operativo Linux Ubuntu 16.04 LTS 64-bit. Se ejecutaron en este equipo los algoritmos de cifrado de 3DES-EDE3, AES, Blowfish , Twofish, Threefish, Cast, R6, Salsa20, Serpent y Shacal2, hechos en lenguaje C++ [80].

ICE Encrypt e IDEA_CBC_SHA, hechos en lenguaje Java, se ejecutaron en un equipo Lenovo Ideapad 320-15ABR, con procesador AMD A12-9720P 7th GEN RADEON R7 4C + 8G 2.70 GHZ, 15.4 GB de RAM, disco duro Seagate Barracuda 1.82 TB de 7200 RPM y sistema operativo Windows 10 Home Single.

Se utilizó también un archivo imagen de 4.87 GB de control para hacer las pruebas de cifrado y descifrado.

El indicador para esta etapa ayudará a seleccionar el método de cifrado más conveniente para la metodología propuesta y sería el siguiente:

MCC = (SC + UC + VC) / NIC

Mientras más cerca esté a 0 el resultado, el método es más conveniente.

MCC= Método conveniente de cifrado. Este es un modo para cuantificar el desempeño del método de cifrado según unas variables asignadas a la fórmula y que si el resultado tiende a 0 (cero), será un método más conveniente de cifrado que los demás estudiados.

SC= Seguridad del cifrado. Teniendo en cuenta que (2^n)-1 claves necesita el atacante para encontrar la clave correcta, donde n es el tamaño en bits de la clave, entonces se da la siguiente escala:

- 0-64 bits= 1
- 65-128 bits = 2
- 129-256 bits = 3
- 257-512 bits = 4

- 513 + bits =5

UC= Uso del cifrado en la comunidad. Mientras más conocido y utilizado por la comunidad informática, más puntos tendrá.

- Ya no se utiliza ni fue reemplazado por otro método = 1
- No se utiliza porque se reemplazó por otro método = 2
- Se utiliza, pero se le han encontrado vulnerabilidades o bugs (fallos) = 3
- Se sigue utilizando actualmente, pero se está reemplazando por otro método o no ha sido estandarizado = 4
- Se sigue utilizando en aplicaciones actuales = 5

VC= Velocidad del cifrado en minutos. Teniendo en cuenta los procesadores a utilizar en el cifrado y el descifrado, las 2 operaciones pueden tomar más o menos minutos entre métodos de cifrado. Con procesadores comunes en equipos de cómputos caseros o de trabajo en oficina, lo ideal es que cada operación no dure más de 3 minutos para una imagen de 5 GB.

- 6:01 + minutos = 1
- 4:31-6:00 minutos = 2
- 3:01-4:30 minutos = 3
- 1:31-3:00 minutos = 4
- 0:00-1:30 minutos = 5

NIC= Número de imágenes cifradas en media hora. Se tiene en cuenta que muchos de los móviles tienen una memoria interna de 32 GB más una memoria RAM de 16 GB, siendo un total de 48 GB. Se toma como tiempo determinado 30 minutos.

- Menos de 1 = 1
- De 1.00 a 1.19 = 2
- De 1.20 a 1.39 = 3
- De 1.40 a 1.59 = 4
- Más de 1.6 = 5

En la tabla 3.2-1 se podrán ver los resultados de los tiempos y uso de los recursos para cada uno de los métodos de cifrado probados. Igualmente, en la tabla 3.2-2, se verán los resultados para el proceso de descifrado con cada uno de los métodos utilizados. Por último, en la tabla 3.2-3 están reflejados los resultados para cada variable utilizada en la ecuación utilizada para la escogencia del método conveniente de cifrado. Los tiempos se miden tanto con el comando TOP en Linux como con cronómetro (time cat) para ayudar en la precisión de los mismos.

2.3 Extracción de imágenes digitales, comprimidas e íntegras

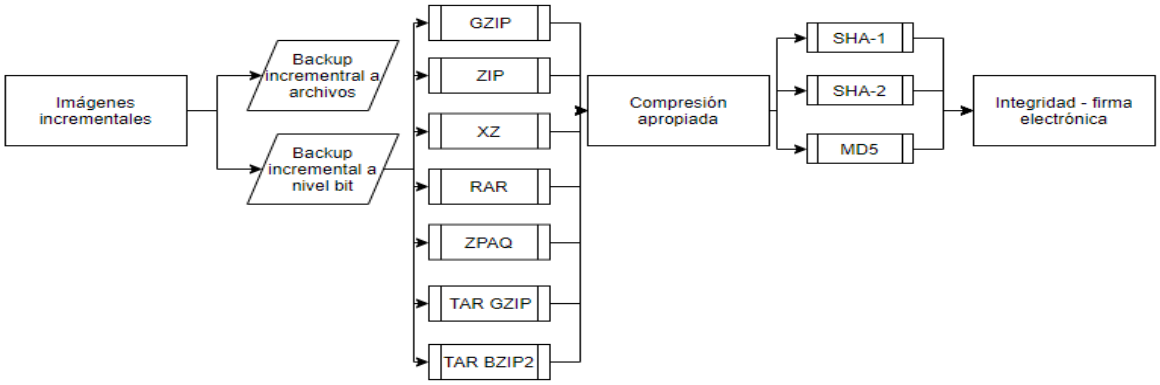

Figura 2.3-1 *Diagrama de bloques de elementos utilizados en la prueba de extracción de imágenes incrementales, compresión e integridad*[3]

En la figura 2.3-1 se muestran los pasos a seguir para esta metodología. Se trabajó en primera instancia en cómo hacer hacer imágenes incrementales; se partió de un método de *backup* incremental a archivos y luego con un *backup* incremental a nivel de *bit*. Ya teniendo esas imágenes incrementales, se procedió a hacer una comparación entre métodos de compresión utilizados en Android, para medir tanto su costo informático como el tiempo utilizado. Luego, se procedió con hacer las métricas entre procedimientos de *hash* para asegurar la integridad de esas imágenes.

Cuando se refiere a técnicas para la extracción de imágenes incrementales, se tienen en cuenta tres factores: que se pueda hacer una imagen incremental en sí, el método de compresión de la

[3] Este diagrama es de realización propia de los autores de este trabajo, basados en los materiales usados para la prueba.

imagen y el método de que asegure la integridad de los datos, para luego enviar la imagen por una red WiFi.

2.3.1 Posibilidad de crear imágenes incrementales

Como ZBACKUP se basa en **rsync** y es una aplicación de terceros, se optó por ejecutar el método de crear imágenes incrementales con **rsync**, que es un paquete que se instala en Linux y en Android, de código abierto y que está en constante mejora.

La emulación del dispositivo Android es hecha con una máquina virtual con privilegios de *root*, donde fue instalada la versión 4.4 de este sistema operativo.

En primer lugar se instaló **rsync server** en una máquina virtual con Kali Linux que será el repositorio de la imagen, como se puede observar en la figura 2.3-2.

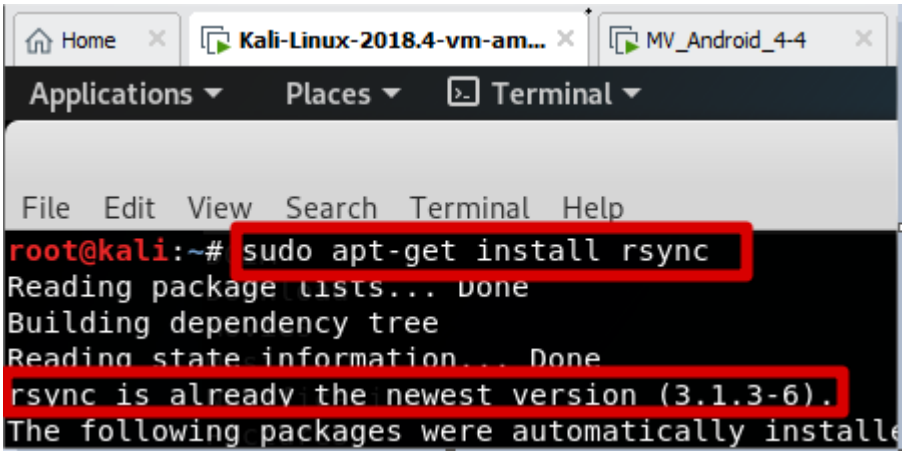

Figura 2.3-2 *Instalación rsync*

Luego se creó el archivo de configuración para el servicio del servidor de **rsyncd** llamado rsyncd.conf, como se muestra en la figura 2.3-3.

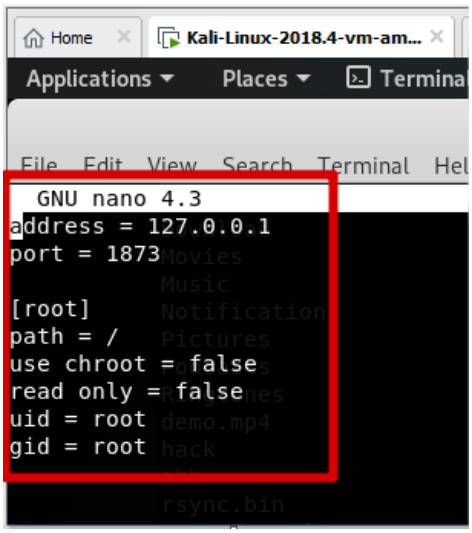

Figura 2.3-3 *Creación archivo rsyncd.conf*

Después se promueve el puerto de **rsyncd** del dispositivo Android al puerto de escucha de la máquina del repositorio, así:

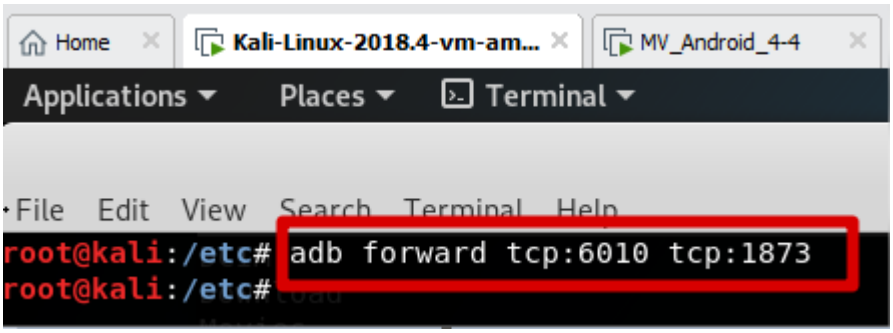

Figura 2.3-4 *Forward del puerto de Android al host*

Se lleva el archivo binario de **rsync** al dispositivo Android, como se muestra en la siguiente figura.

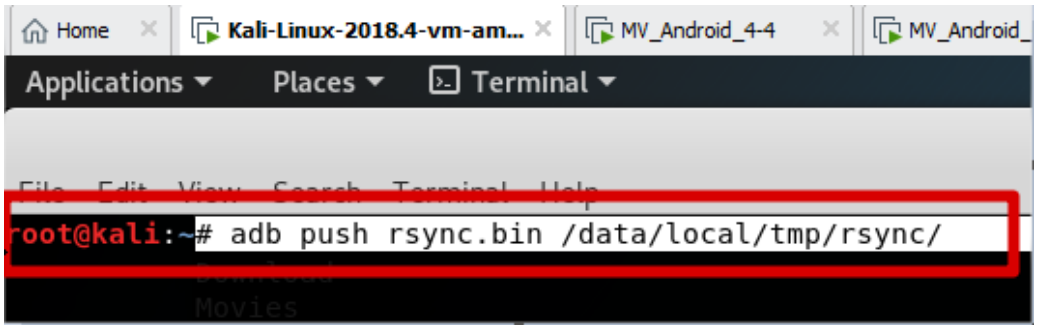

Figura 2.3-5 *Envío del archivo rsync.bin a Android*

Se le concede permisos de escritura, lectura y ejecución al propietario del archivo y a los demás usuarios sólo de lectura y ejecución.

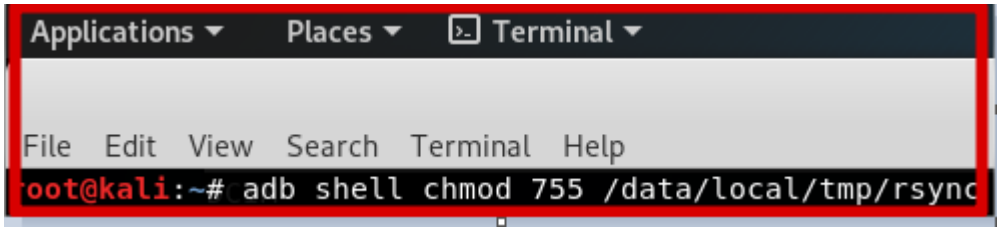

Figura 2.3-6 *Otorgamiento de permisos al archivo*

Como el servicio de **rsyncd** establece la comunicación con el dispositivo remoto por medio del servicio SSH, por motivos de seguridad se debe cambiar el puerto predeterminado 22 del servicio SSH por un puerto no estándar, para que solo sea conocido por el servicio de sincronización rsync. Por lo tanto, para esta investigación se asignó el puerto 2222 al servicio SSH. Entonces se debe hacer la configuración respectiva del servicio SSH en el archivo que está en la ruta /etc/ssh/sshd_config/.

Se verifica la comunicación entre la máquina de Kali Linux y la máquina virtual con Android, como muestra la figura 2.3-6.

Figura 2.3-7 *Comunicación entre Kali y Android*

Se prueba localmente el funcionamiento de **rsync** haciendo una copia de seguridad incremental.

Figura 2.3-8 *Prueba de rsync localmente*

2.3.2 Identificación del método de compresión apropiado

Para identificar el método de compresión apropiado para la extracción de la imagen, se utilizó un computador Dell Inspiron 1525, con procesador INTEL CORE 2 DUO CPU T5750 2.00 GHZ X 2 Centrino, 3.8 GB de RAM, disco duro HGST de 489.7 GB de 3600 RPM y sistema operativo Linux Ubuntu 18.04 LTS 64-bit y se instalaron los paquetes de compresión en el sistema operativo, como son ZIP, GZIP, BZIP2, XZ, RAR y ZPAQ. Igualmente se instaló el empaquetador TAR. La imagen con que se probaron todas estas técnicas de compresión tiene un tamaño de 5.2 GB y tenía por nombre imagen.img. Es de aclarar que los paquetes de compresión utilizados fueron validados también en Android e igualmente son operativos como en Linux. Se prueban en Linux su funcionamiento y luego se realizan en el ambiente controlado del que se hablará más adelante.

En el computador antes mencionado, se colocó un archivo imagen y se ejecutó cada uno de los métodos de compresión a la vez, midiendo tiempos y consumos de CPU y memoria. Igualmente se observó el tamaño del archivo resultante de la compresión y se comparó con el archivo original de la imagen.

Para probar el método Zip, se ejecutó la siguiente sentencia:
zip windows2003a.img.zip windows2003a.img

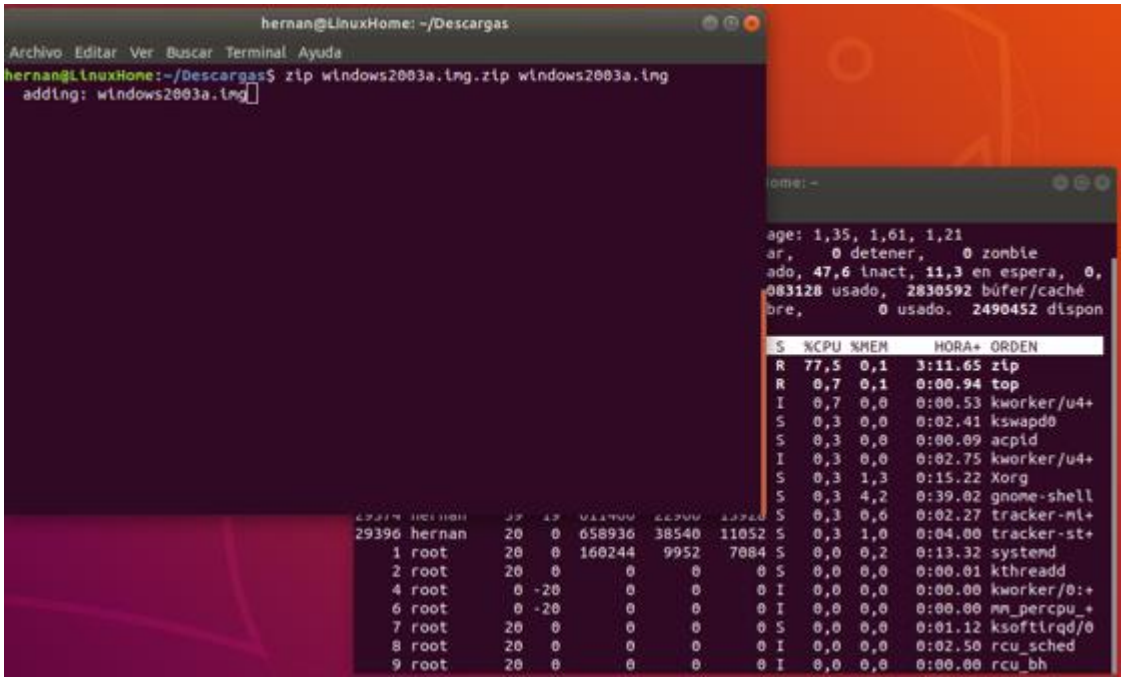

Figura 2.3-9 *Compresión de la imagen con ZIP*

Figura 2.3-10 *Resultado de la compresión con ZIP*

Con el método GZIP, se ejecutó así:

Se hizo primera una copia del archivo windows2003a.img: cp windows2003a.img windows2003a2.img

Luego: gzip windows2003a2.img. El archivo resultante es windows2003a2.igm.gzip

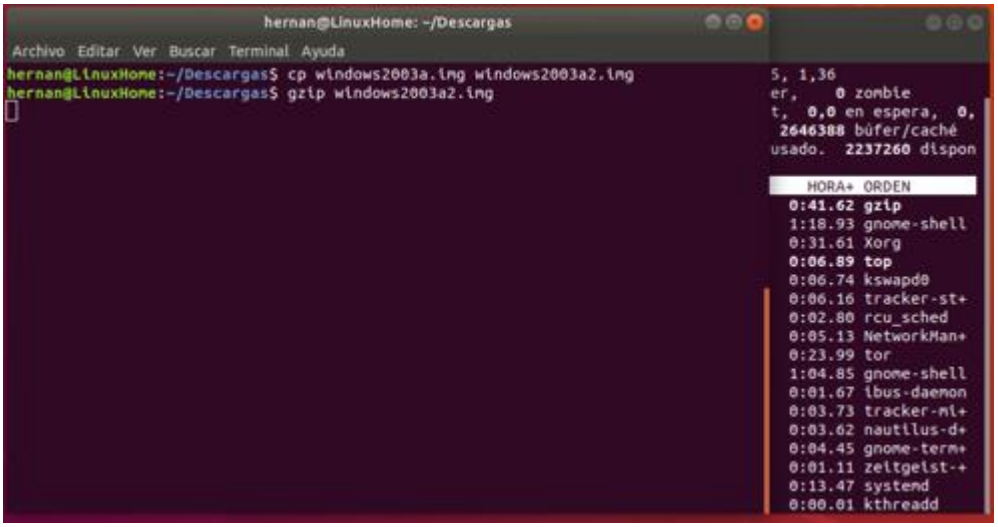

Figura 2.3-11 *Compresión de la imagen con GZIP*

Figura 2.3-12 *Resultado de la compresión con GZIP*

Se ejecutó el método BZIP2:

Se hizo una copia de windows2003a.img: cp windows2003a.img windows2003a3.img

Luego: bzip2 windows2003a3.img. El archivo resultante fue windows2003a3.bz2

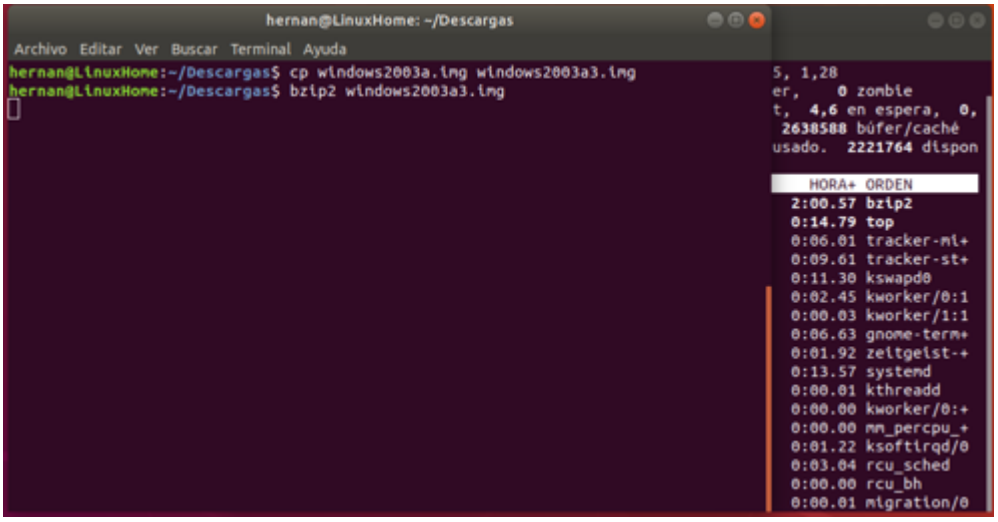

Figura 2.3-12 *Compresión con BZIP2*

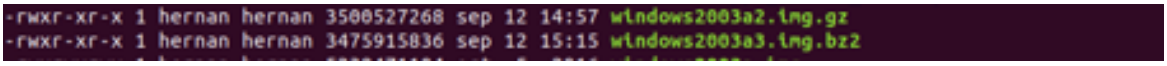

Figura 2.3-13 *Resultado compresión con BZIP2*

Se probó el método XZ así:

Se hizo una copia del archivo windows2003a.img: cp windows2003a.img windows2003a4.img

Se ejecutó el comando: xz windows2003a4.img. El archivo final fue windows2003a4.xz

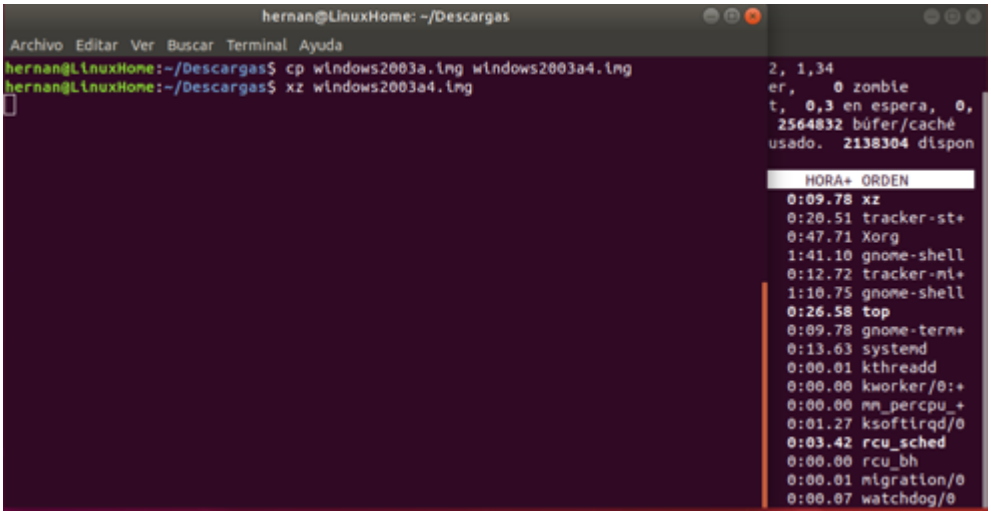

Figura 2.3-14 *Compresión con XZ*

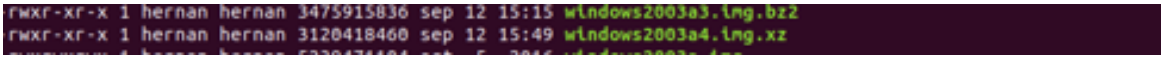

Figura 2.3-15 *Resultado de la compresión con XZ*

Se probó el método RAR:

rar a -y windows2003a.rar windows2003a.img

Figura 2.3-16 *Compresión con RAR*

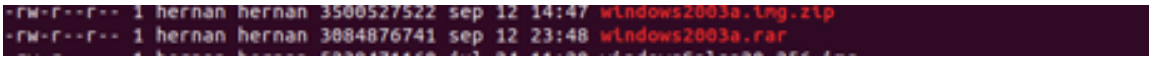

Figura 2.3-17 *Resultado de la compresión con RAR*

Por último, se probó ZPAQ:

Zpaq qc windows2003a5.zpaq windows2003a5.img

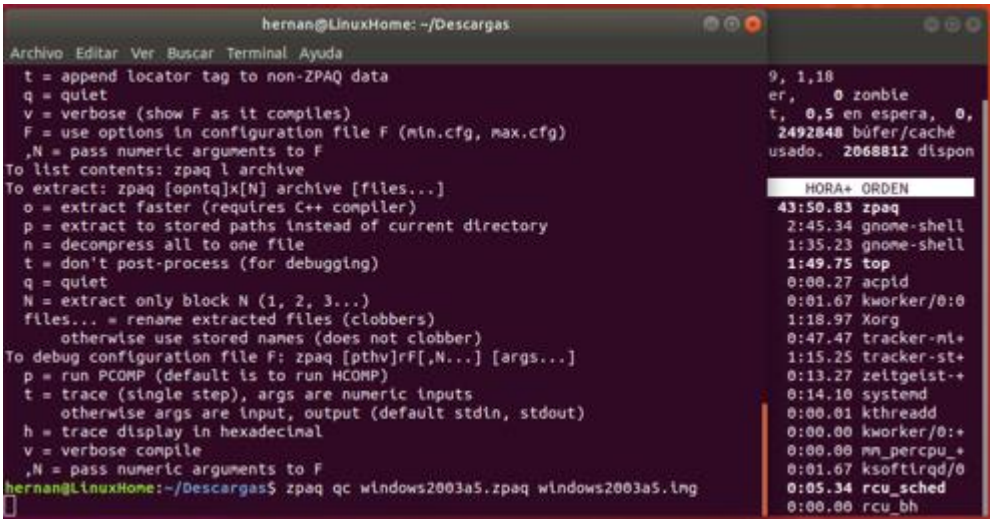

Figura 2.3-18 *Compresión con ZPAQ*

Figura 2.3-19 *Resultado de la compresión con ZPAQ*

También se probó la combinación de un empaquetador y un compresor:

- Primero se probó TAR con GZIP:

  tar czvf windows2003a.tar.gz windows2003a.img

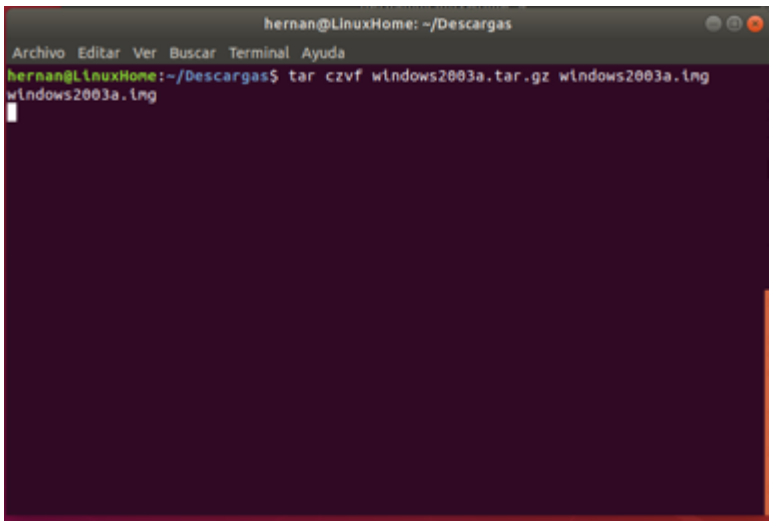

Figura 2.3-20 *Compresión y empaquetamiento con TAR-GZIP*

Figura 2.3-21 *Resultado de la compresión y empaquetamiento con TAR-GZIP*

- Luego se probó TAR con BZIP2:

  tar cjvf windows2003a.tar.bz2 windows2003al.img

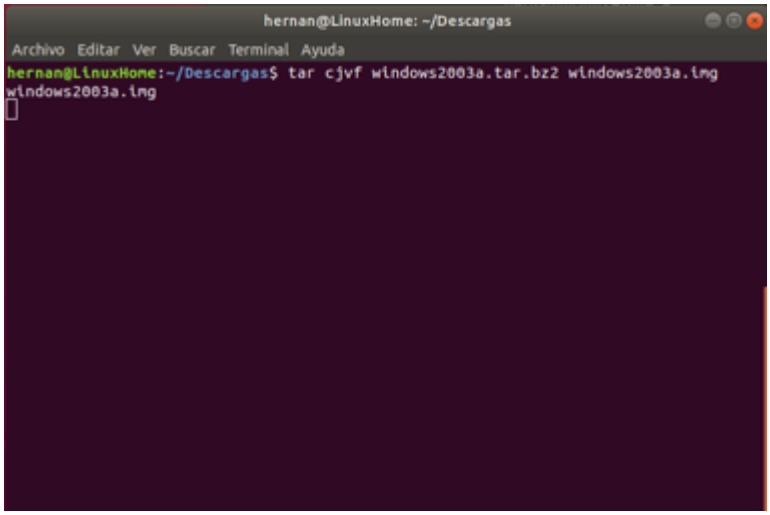

Figura 2.3-22 *Compresión y empaquetamiento con TAR-BZIP2*

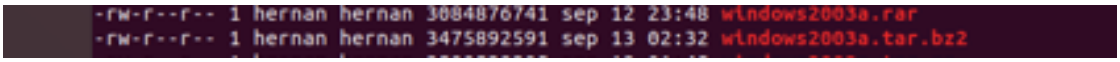

Figura 2.3-23 *Resultado de la compresión y empaquetamiento con TAR-BZIP2*

Para medir la tasa de compresión se utilizó la cuantificación [Valor Absoluto] (100 x Tamaño imagen comprimida) / (Tamaño original de la imagen).

El método de compresión a elegir será el que mejor uso de CPU y memoria y en relación con la tasa de compresión tenga. Como se utilizará en dispositivos móviles, el uso de los recursos de CPU y memoria es crítico y teniendo en cuenta que la tasa de compresión no es variable en grandes proporciones entre métodos, los valores de esos recursos son lo más valiosos para esta investigación. En la tabla 3.3-1 se podrán ver los resultados en la comparación de los métodos de compresión.

Los tiempos de compresión se miden con el comando TOP en Linux y con cronómetro para ayudar en la precisión de los mismos.

2.3.3 Identificación de la técnica de integridad de los datos

Se probaron varios métodos de *hash*, que por definición esta técnica mantiene la integridad de los datos. Este es un paso después del método cifrado probado en el numeral 2.2 del presente trabajo.

Se utilizaron para la prueba SHA-1, la familia SHA-2 (224, 256, 384 y 512 *bits*) y MD5. Todas las pruebas reportaron para el archivo de imagen de 5.2 GB, un tiempo de procesamiento del *hash* entre los 50 y 60 segundos. Aquí la longitud de la firma del cifrado es la que varía según los *bits* que utilice el método de *hash*. Aunque el tiempo y el uso de CPU son factores importantes, lo que prima acá es la seguridad y por lo tanto mientras la firma sea más larga y con menos uso de CPU y tiempo, se considerará para usar en la investigación. MD5 ya ha sido reportado como inseguro desde hace algún tiempo y la firma es de sólo 128 bits, sin embargo, se utilizó para la prueba y poder hacer la comparación. En las figuras 2.3-24 hasta la 2.3-31 se evidencian las ejecuciones de estos algoritmos. Igualmente, en la tabla 3.3-2 se muestran los resultados en cuanto a uso de CPU, uso de memoria y tiempo de ejecución.

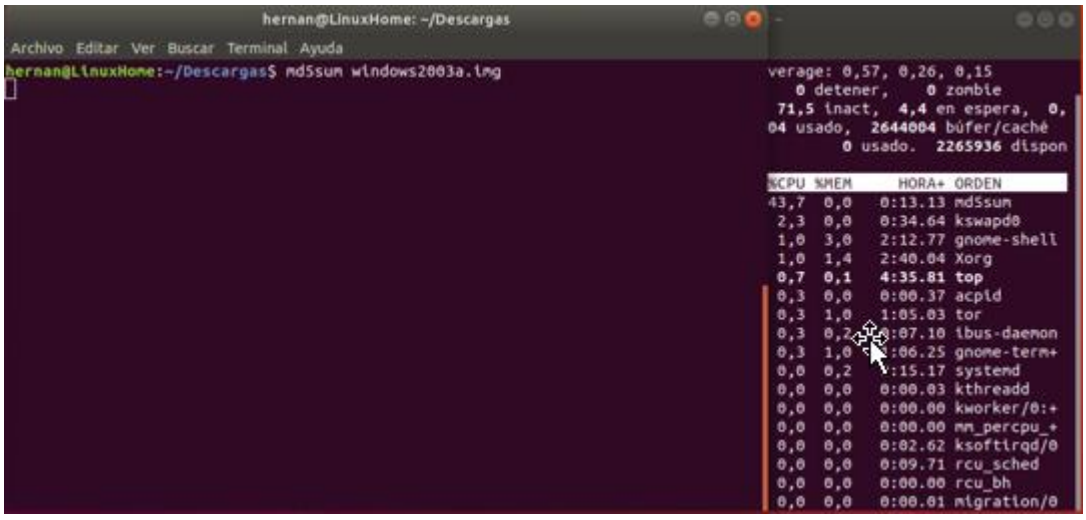

Figura 2.3-24 *Ejecución de MD5*

Figura 2.3-25 *Resultado de MD5*

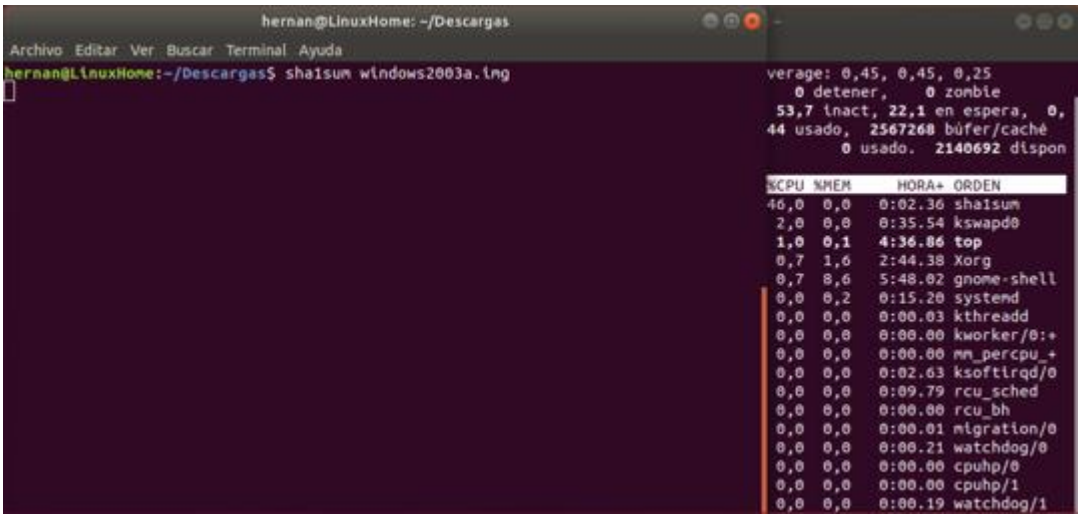

Figura 2.3-26 *Ejecución de SHA1*

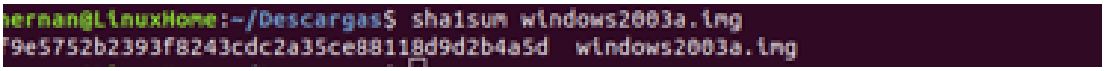

Figura 2.3-27 *Resultado de la firma con SHA1*

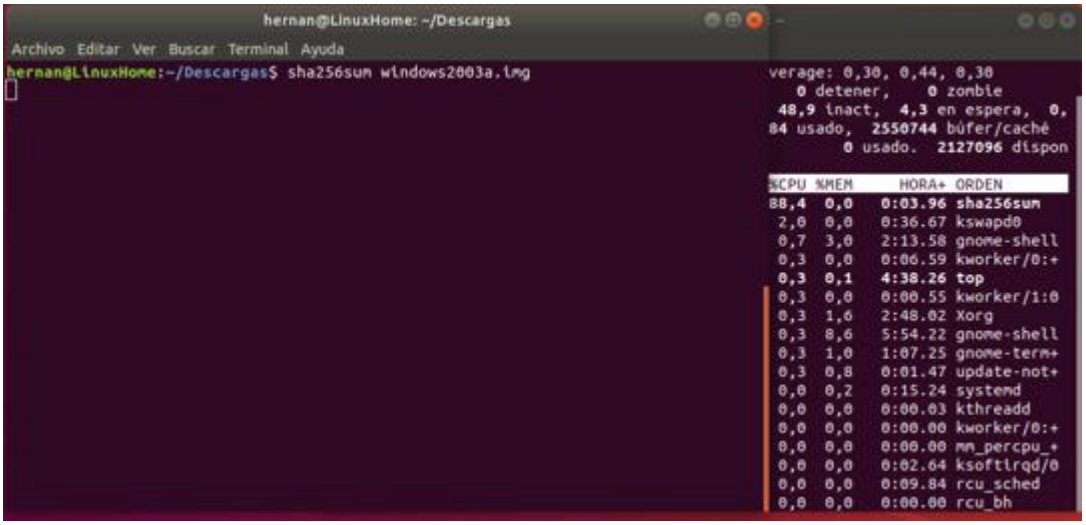

Figura 2.3-28 *Ejecución de SHA256*

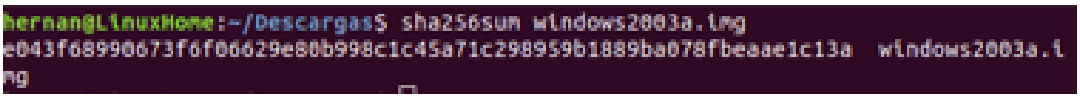

Figura 2.3-29 *Resultado de SHA256*

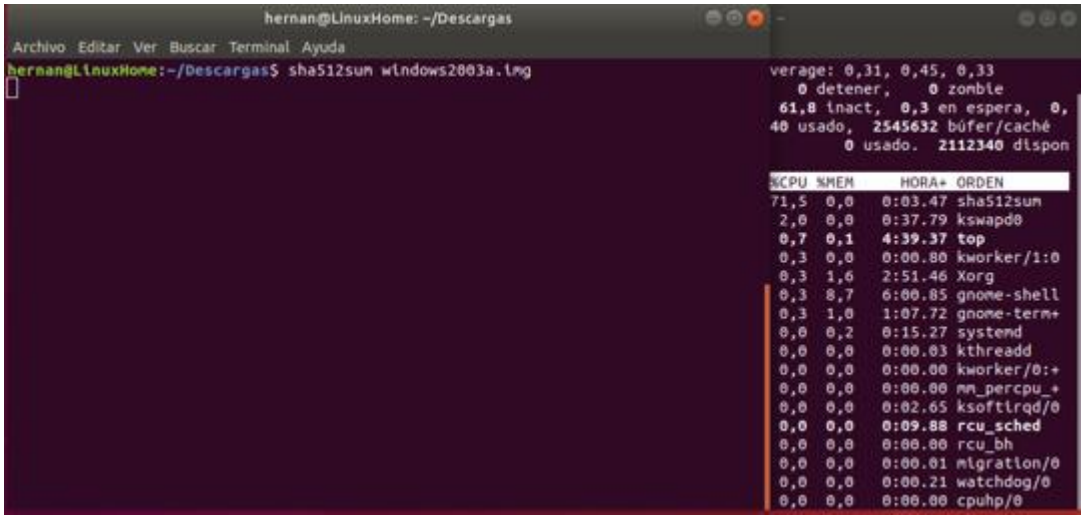

Figura 2.3-30 *Ejecución de SHA512*

Figura 2.3-31 *Resultado de SHA512*

2.4 Validación de la metodología propuesta bajo un ambiente controlado

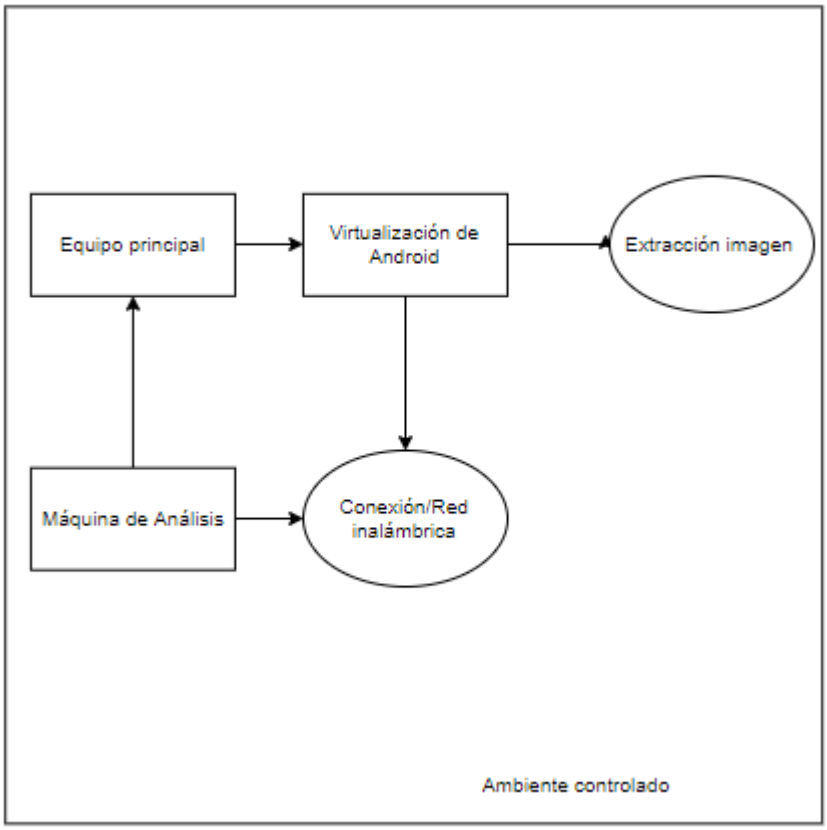

Figura 2.4-1 *Proceso de validación en ambiente controlado*

En la figura 2.4-1 se puede visualizar que se creó un ambiente controlado para validar la metodología propuesta en este trabajo. Se necesitaron de unas máquinas con unas características que se mencionan más adelante, para poder hacer las pruebas y el control de lo realizado. Igualmente, una conexión a una red inalámbrica para los dispositivos móviles y el producto final es la extracción de la imagen completa y segura.

Se propone un ambiente controlado virtualizado por los siguientes motivos: la simulación de un entorno real, hacer diferentes análisis exhaustivos, no comprometer dispositivos físicos, independencia de cada máquina virtual y la facilidad de manejo de los sistemas operativos.

Además, para la simulación del ambiente real y obtener datos fidedignos al momento de la extracción de la imagen digital forense, se cuenta con interfaces de red inalámbrica físicas (AP y tarjetas de red WiFi) independientes para cada máquina virtual.

2.4.1 Diagrama de la arquitectura del ambiente controlado

En la siguiente imagen se detalla el diagrama del ambiente controlado donde se tiene una máquina nativa y la visualización del sistema operativo Kali Linux que sirvió para prestar diferentes servicios como repositorio de la imagen, servicio de DHCP, entre otros. También se tiene virtualizado el sistema operativo Android en la versión 4.4 de forma distribuida.

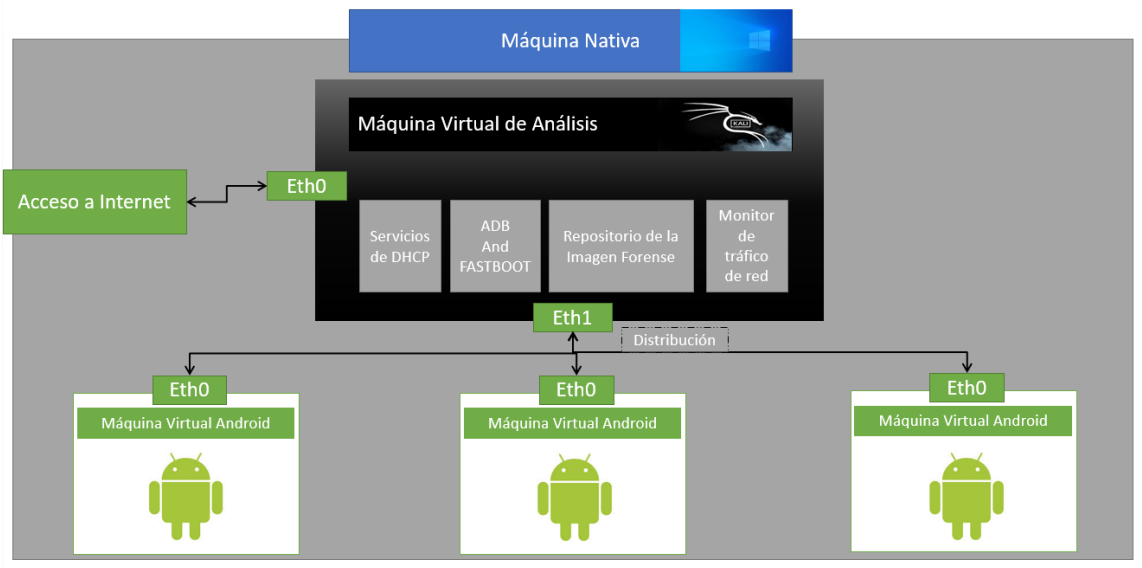

Figura 2.4-1 *Arquitectura del ambiente controlado*[4]

2.4.2 El equipo principal

Se cuenta con un computador portátil de la marca Hewlett-Packard (HP), que tendrá la tarea de ser la plataforma física para la virtualización de los diferentes sistemas operativos utilizados en el

[4] Este diagrama es de realización propia de los autores de este trabajo, basados en los materiales usados para la prueba

ambiente controlado. Este equipo debe contar con unos requerimientos mínimos de *hardware* y *software* para poder estabilizar el ambiente controlado.

El dispositivo o equipo principal es un Laptop HP Elite Book -Folio 1040. Para este entorno se le nombrará como HP y tiene como función soportar la virtualización de los diferentes sistemas operativos utilizados en el ambiente controlado y cuenta con las siguientes características:

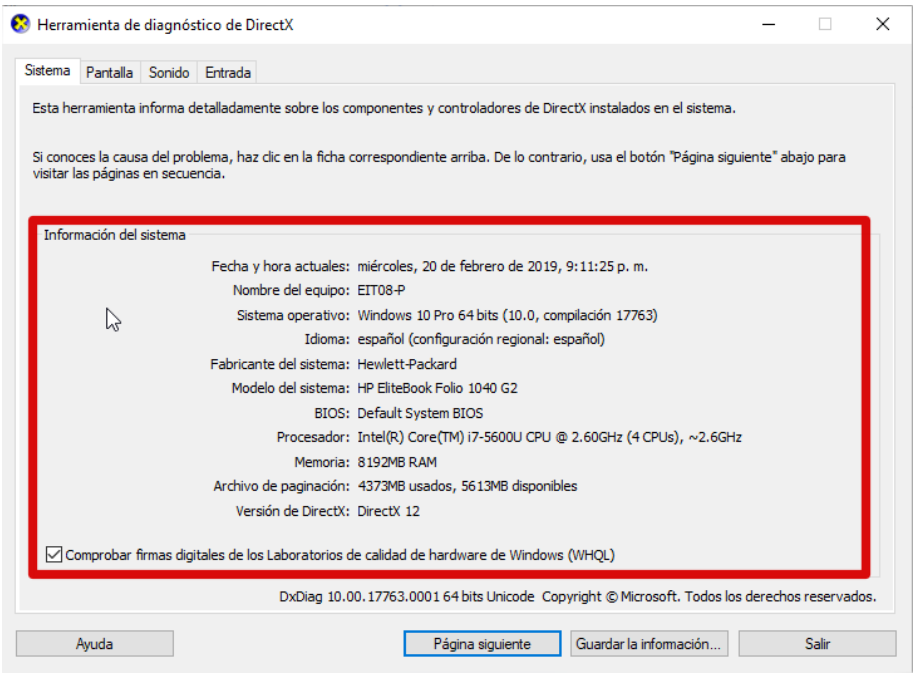

Figura 2.4-2 *Descripción de hardware, software y equipo utilizados*

Seguidamente se parametrizó la interfaz de red inalámbrica para contar con una estabilización en las telecomunicaciones del ambiente controlado. Se realizó la configuración de la interfaz red inalámbrica dejando la banda de 5 GHZ como preferida y en el protocolo de internet IP versión 4 como DHCP.

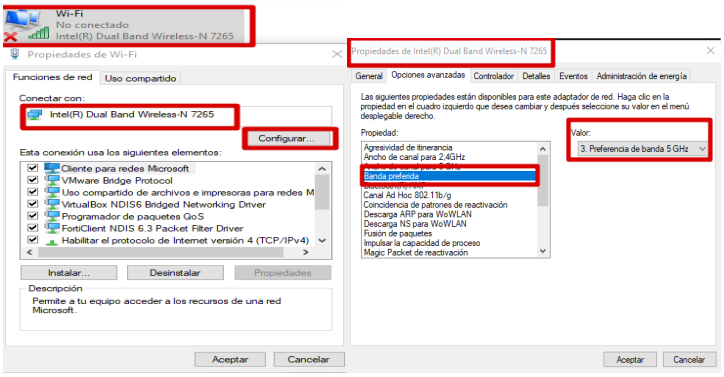

Figura 2.4-3 *Parámetro de la red inalámbrica*

2.4.3 Red inalámbrica

Para tener un ambiente controlado con óptimas condiciones en el tema de comunicaciones, se parametrizó con un *Access Point* (AP) de la marca Ubiquiti modelo UNIFI AC LITE con doble banda (2.4 GHz y 5GHz). Este laboratorio se trabajó con el estándar de WiFi 802.11 n/ac en la banda 5 Ghz, para mayor velocidad de transferencia al momento de hacer la extracción de la imagen digital forense.

Este dispositivo es un AP UBIQUITI modelo UNIFI AC LITE con doble banda y servirá para crear una red inalámbrica donde se conectará el equipo HP. Los parámetros designados para el UniFI son los siguientes:

Nombre SSID (Servicio del identificador de nombre de red): Forense_Android

IP: Dinámica

Protocolo: 802.11.n y 802.11.ac

Banda: 5 Ghz

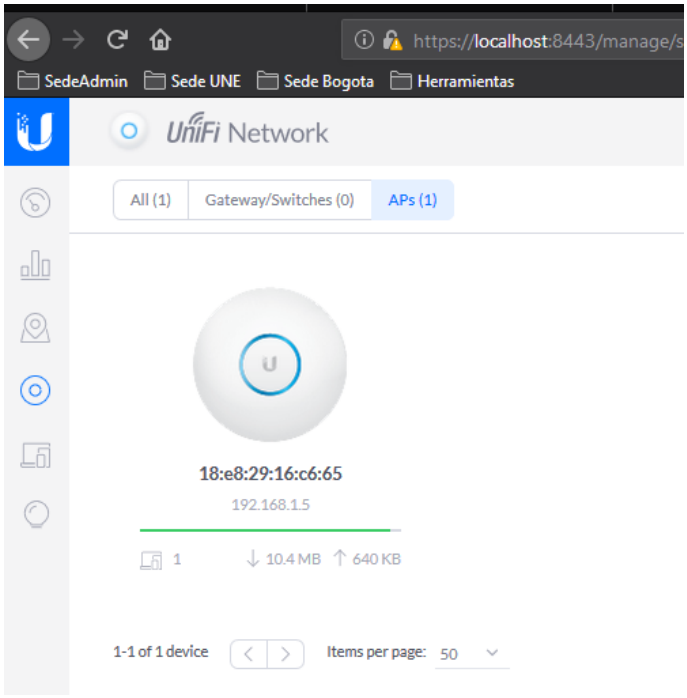

Figura 2.4-4 *Características y configuración de la red inalámbrica*

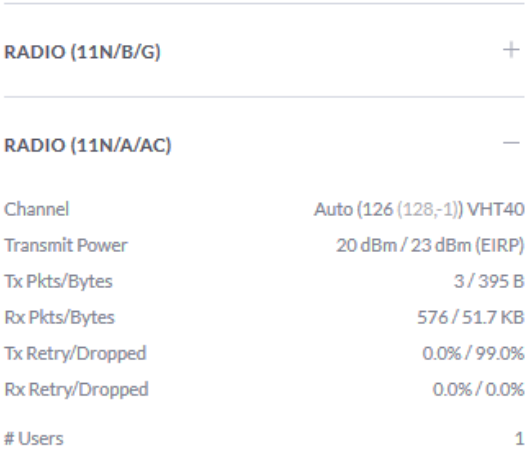

Figura 2.4-5 *Verificaciones del estado de configuración del dispositivo de comunicaciones*

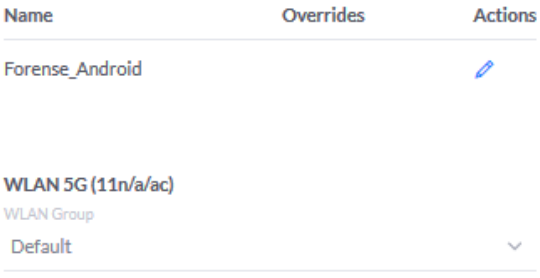

Figura 2.4-6 *Servicio del identificador de nombre de red*

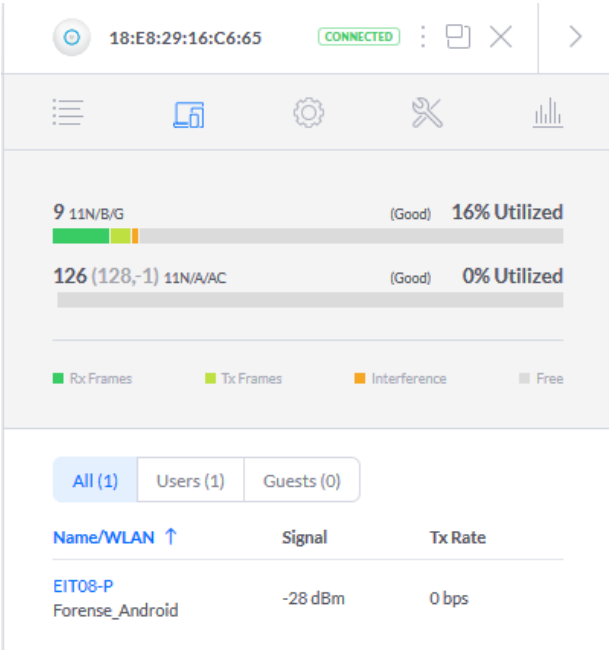

Figura 2.4-7 *Conexión máquina HP al SSID nombrado Forense_Android*

Se descargó el manual de configuración de este dispositivo de la siguiente página web: https://dl.ubnt.com/guides/UniFi/UniFi_AP-AC-Lite_QSG.pdf.

2.4.4 Máquina virtual Kali Linux (Máquina de Análisis)

Esta máquina virtual cuenta con el sistema operativo Kali Linux, que prestó el servicio de DHCP para la asignación de direcciones IP, de ADB Shell, Netcat y *root* para la conexión al sistema operativo

Android y de un monitoreo del tráfico de red, para analizar el tráfico enrutado y además tiene la función de repositorio donde se almacenaron las imágenes digitales forenses extraídas.

La imagen ISO se descargó del siguiente repositorio: https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-image-download/.

Se contó con la virtualización del sistema operativo Kali Linux en la versión kali-linux-2018.4-vm-amd64 (4.18.0-kali2-amd64). Para esta investigación se nombró Máquina de Análisis y tiene las siguientes configuraciones en la virtualización:

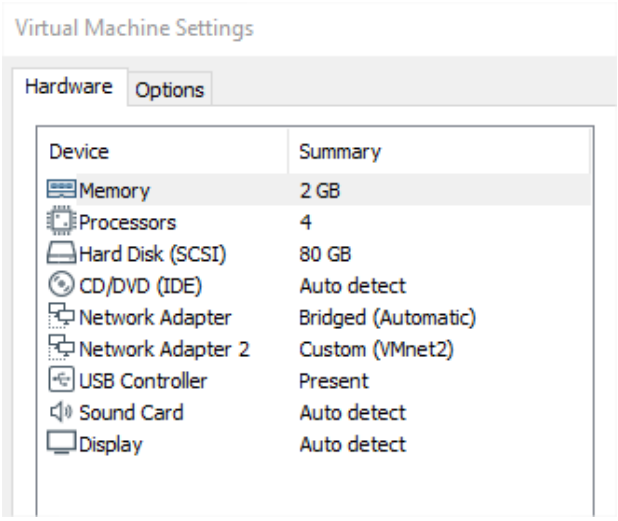

Figura 2.4-8 *Resumen de la virtualización de la máquina de análisis*

Los componentes de la máquina virtual son 2 GB de memoria RAM para un mejor procesamiento de la información y el *software* instalado, cuatro núcleos del procesador físico para mejorar el rendimiento y velocidad, un disco de 80 GB para el almacenamiento y dos tarjetas de red donde utiliza diferentes modos de configuración para la comunicación entre dispositivos.

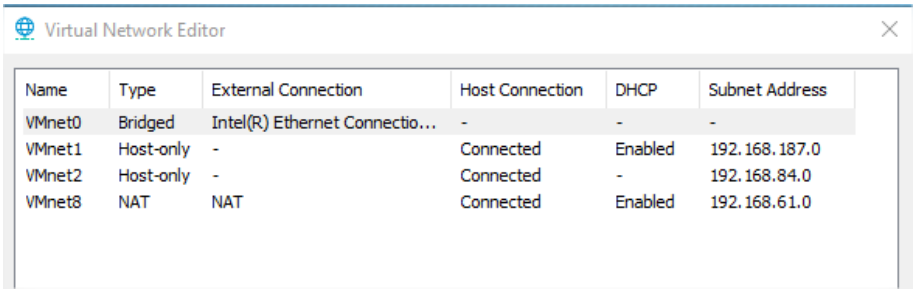

Figura 2.4-9 *Configuración de VMWare para las interfaces de red*

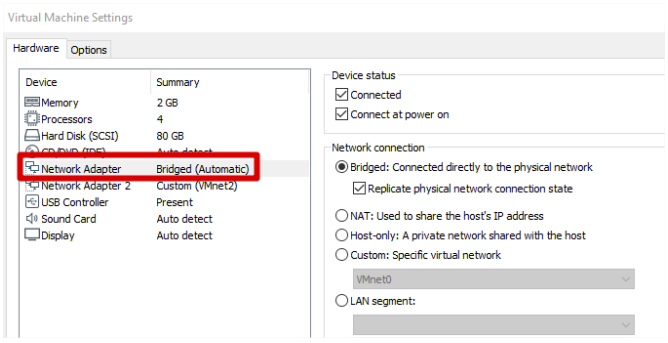

Figura 2.4-10 *Interfaz de red en modo puente*

La interfaz de red en puente crea una interfaz de red virtual, donde replica la configuración del dispositivo de red anfitrión, en este caso de la máquina HP. Esta interfaz debe tener acceso a internet [81].

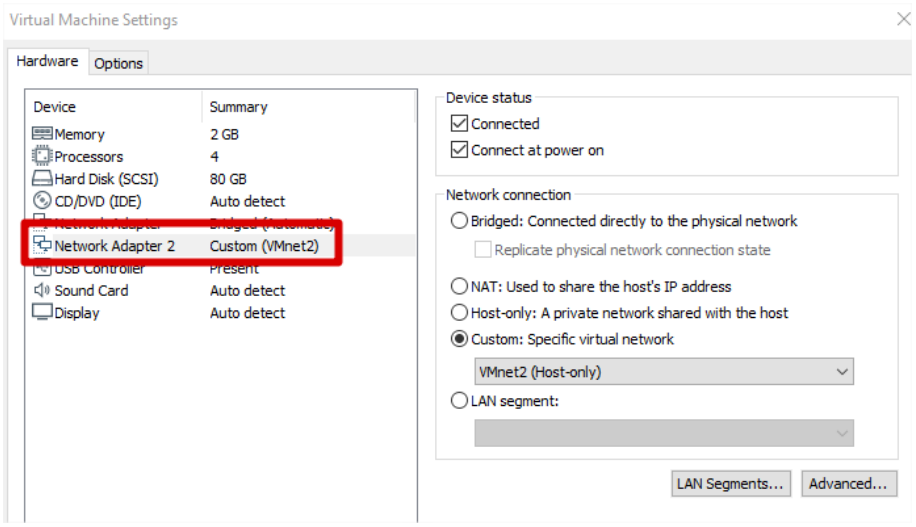

Figura 2.4-11 *Interfaz de red en modo Host-Only*

En el modo Host-Only se crea una red virtual interna para comunicación solo en esta red, no se tendrá acceso a otras redes. Inclusive se deshabilitará el servicio de DHCP de esta configuración.

Posteriormente se da inicio a la máquina virtual descargada con anterioridad, luego de la inicialización del sistema operativo va a solicitar el acceso con un usuario y una contraseña las que corresponde a usuario: root y contraseña: toor.

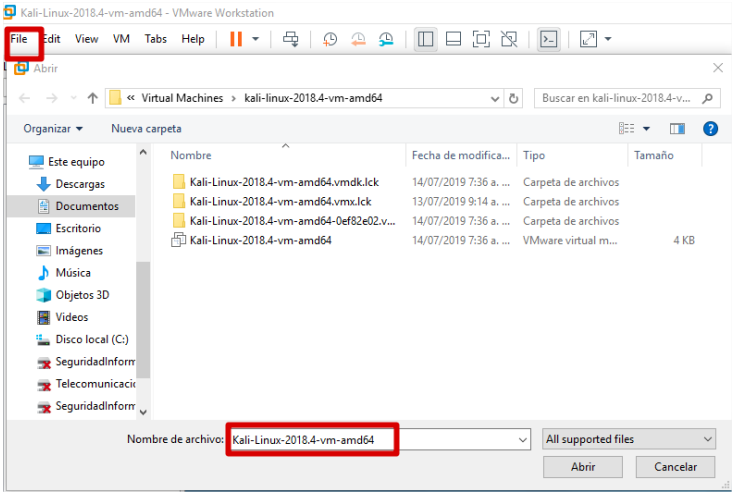

Figura 2.4-12 *Apertura de la máquina virtual Kali Linux*

2.4.5 Virtualización Android

La versión del sistema operativo Android utilizada para este entorno es la 4.4 llamada Kit Kat, donde se aprovechará la compatibilidad con el ambiente controlado. Además, la pruebas que se realizaron al sistema operativo Android, pueden ocasionar daños que comprometen el dispositivo físico y por lo tanto se virtualizará. También se tienen dos máquinas virtuales con sistema operativo Android con el fin de hacer una simulación de un ambiente distribuido.

Las imágenes ISO para la instalación se descargaron del siguiente repositorio: https://www.android-x86.org/download.

Se contó con la virtualización del sistema operativo Android en la versión 4.4 que se llama en esta investigación MV_Android_4-4 y tiene las siguientes características y configuraciones:

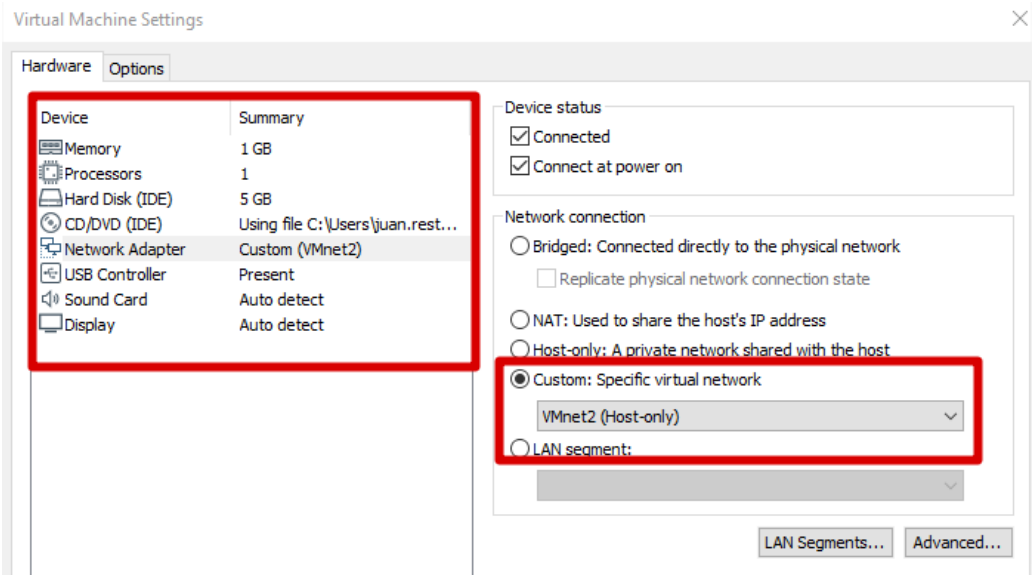

Figura 2.4-13 *Resumen de la virtualización de Android*

La configuración de la máquina virtual Android contó con un GB de memoria RAM para un mejor procesamiento de la información y el software instalado, un núcleo del procesador físico para mejorar el rendimiento y velocidad, un disco de cinco GB para el almacenamiento y una tarjeta de red donde utiliza el modo Host-Only. En esta ocasión se debe crear una nueva máquina virtual ya que la de Android fue en formato ISO. Para esto se ingresa a VMWare, ir a la pestaña archivo y abrir una nueva máquina virtual. En la opción de boot se carga la imagen android-x86-4.4-r5.iso e se inicia la máquina virtual para hacer la instalación del sistema operativo.

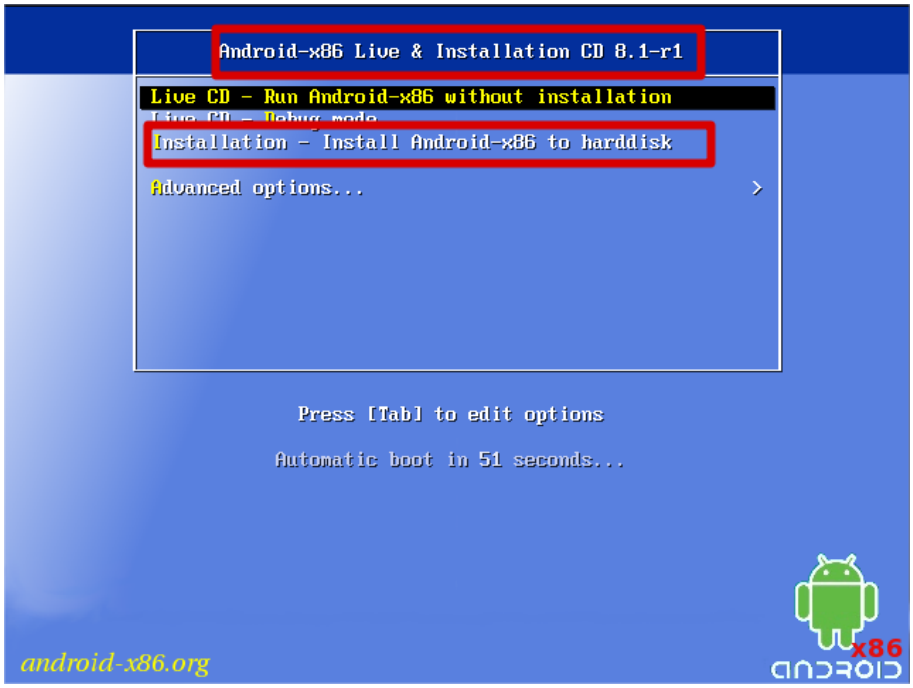

Figura 2.4-14 *Instalación de Android en la máquina virtual*

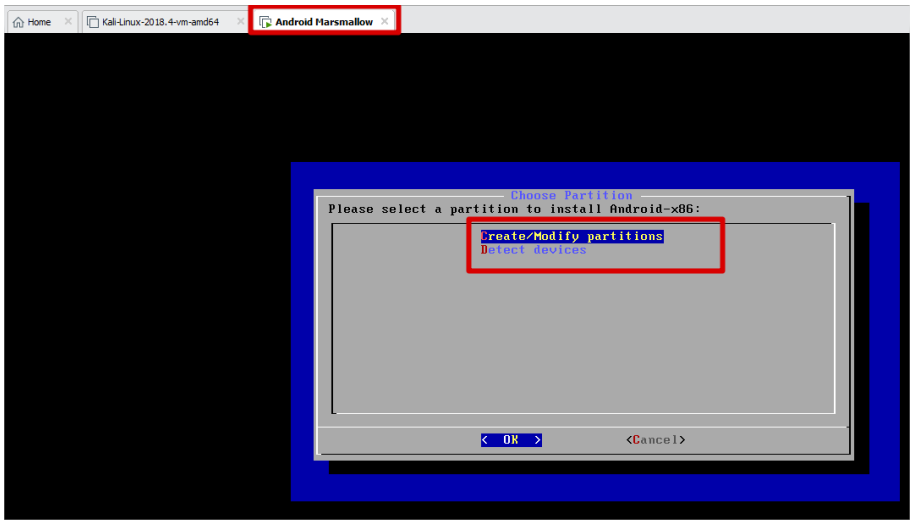

Figura 2.4-15 *Creación de la partición*

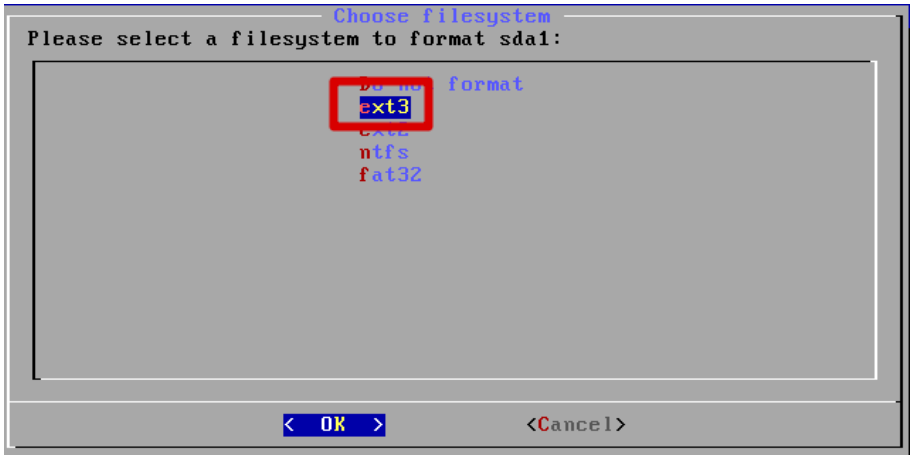

Figura 2.4-16 *Selección de la partición creada*

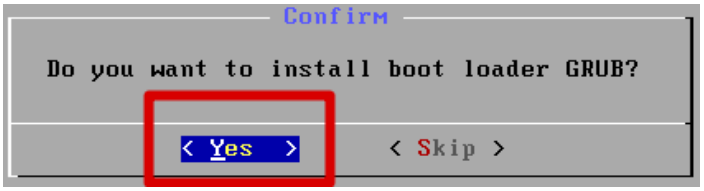

Figura 2.4-17 *Gestor de inicio*

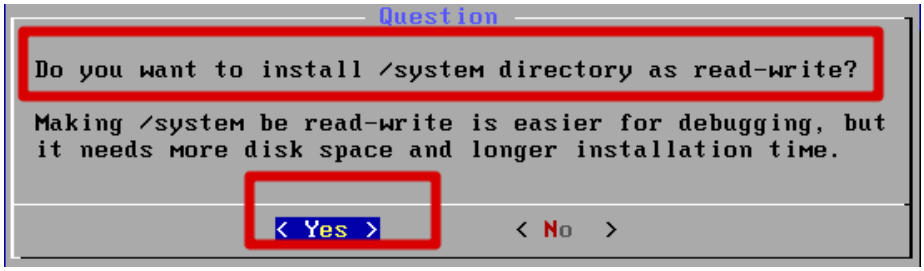

Figura 2.4-17 *Confirmación de la instalación en el directorio indicado*

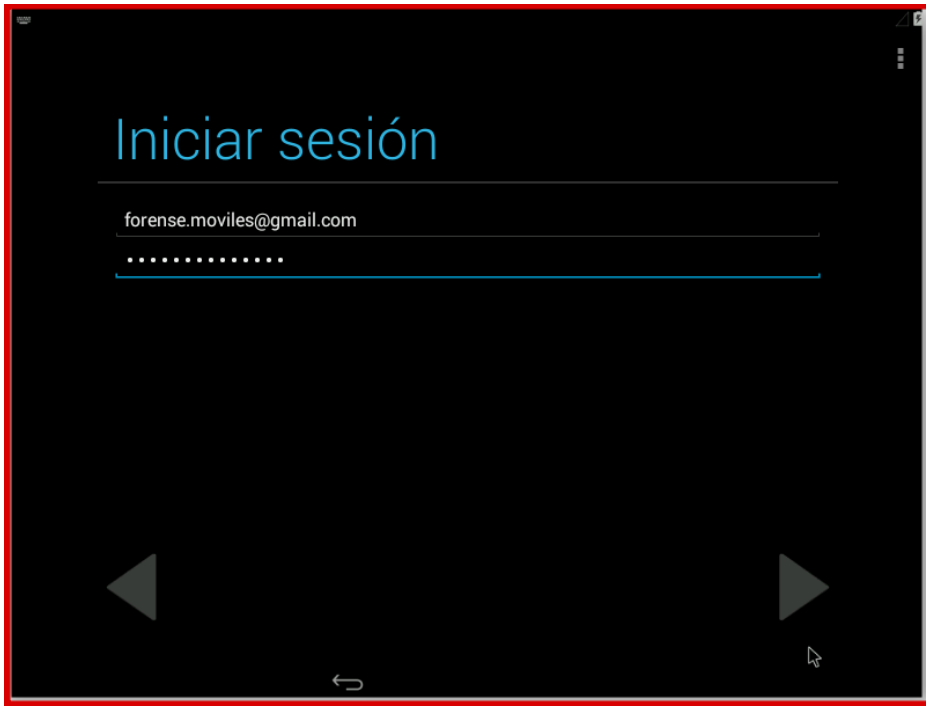

Figura 2.4-18 *Creación de la cuenta de Google*

Así mismo, se puede clonar la máquina virtual creada anteriormente para completar los otros sistemas operativos Android del ambiente controlado propuesto. La máquina virtual clonada se llamó en esta investigación MV_Android_4-4(2). Para clonar la máquina virtual solo basta copiar los archivos originales a un nuevo directorio.

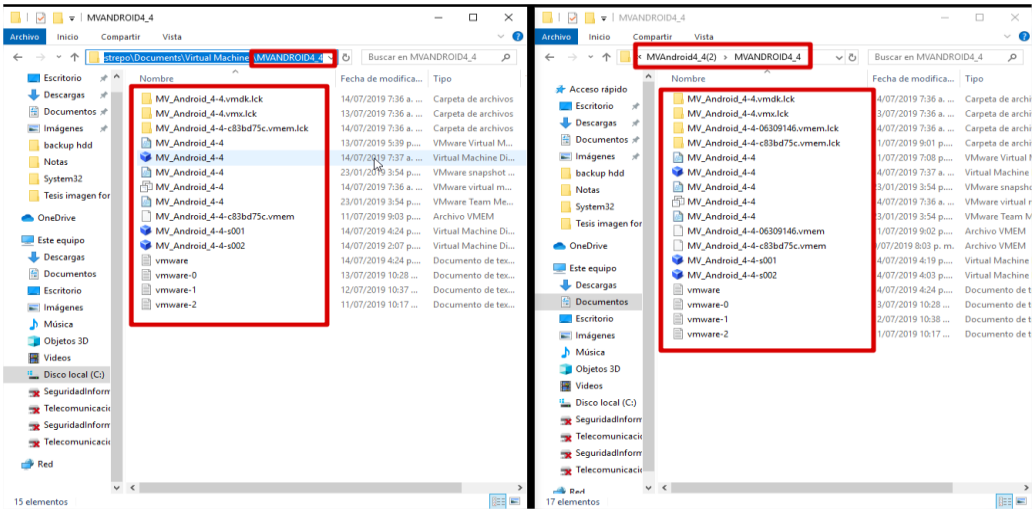

Figura 2.4-19 *Clonación de la máquina virtual de Android*

Por lo tanto, ya creados los equipos en el ambiente controlado, se continua con la instalación y configuración de los componentes necesarios en cada uno de los sistemas operativos para realizar la extracción de la imagen digital forense.

En la máquina de análisis se inició la actualización del sistema operativo y sus aplicaciones con el comando apt-get update && apt-get upgrade && apt-get dist-upgrade.

Para validar si la actualización se realizó con éxito se ingresó el comando donde muestra la versión del sistema operativo, *lsb_release -a*

Seguidamente en la máquina de análisis se propuso un sistema de DHCP para la asignación de las direcciones IP de forma dinámica a los otros sistemas operativos del ambiente controlado y a su vez puedan comunicarse con el repositorio. Por consiguiente, se realizó la instalación del servicio de DHCP con el comando #apt-get install isc-dhcp-server.

Para que el servicio de DHCP funcione de forma correcta, se hicieron los siguientes cambios en la configuración del servicio de red. En este caso las interfaces de red deben estar configuradas de estas formas, la eth0 debe estar de modo DHCP y la interfaz eth1 de modo estática.

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp

auto eth1
iface eth1 inet static
address 10.0.0.1
netmask 255.255.255.0
network 10.0.0.0
broadcast 10.0.0.255
gateway 192.168.1.254
```

Figura 2.4-20 *Configuración interfaz de red*

Antes de validar los cambios realizados, se reinició el servicio de red del sistema operativo Kali Linux. Esto se hace con el fin de evitar errores en el servicio de red. El comando utilizado para esta acción es /etc/init.d/networking restart. Posteriormente al reinicio se validaron los cambios de la configuración con el comando ifconfig y por último un diagnóstico de la comunicación de red con el comando ping x.x.x.x.

Ya configurado el servicio de red y realizado las pruebas pertinentes, se inicia la configuración del servicio del DHCP y para esto se hicieron los cambios con el comando nano /etc/dhcp/dhcpd.conf, quedando así el archivo:

```
default-lease-time 600;
max-lease-time 7200;
option domain-name-servers 8.8.8.8, 8.8.4.4;

#option domain-name "yourdomainname.com";

subnet 10.0.0.0 netmask 255.255.255.0 {
range 10.0.0.10 10.0.0.30;
option subnet-mask 255.255.255.0;
option broadcast-address 10.0.0.255;
option routers 10.0.0.1;
}
```

A continuación, se reiniciaron los servicios de red y DHCP. Además, se constató en la tabla ARP, la ubicación de los clientes del servicio DHCP y en los sistemas operativos Android la asignación de las direcciones IP:

Comando reinicio servicio de red: /etc/init.d/networking restart
Comando reinicio servicio de DHCP: /etc/init.d/isc-dhcp-server restart,
Comando en el servicio DHCP: cat /var/lib/dhcp/dhcpd.leases

Adicionalmente se requiere la instalación de la herramienta fastboot para que exista comunicación con el dispositivo Android. La diferencia con esta herramienta a la herramienta ADB, es que esta sólo se comunica cuando Android inicia en modo *bootloader*. Comando de Instalación: sudo apt-get install android-tools-fastboot.

2.4.6 Procedimiento para la conexión entre la máquina de análisis y los dispositivos Android

En esta investigación, se realiza la instalación de forma remota de algunas aplicaciones para el sistema operativo Android de la máquina virtual MV_Android_4-4 para *"rootear"* el sistema operativo Android y tener más privilegios [82]; se hace con el fin de aprovechar las vulnerabilidades del sistema operativo Android. Por otra parte, la máquina virtual MV_Android_4(2) no contó con estas aplicaciones y se realizó en mismo procedimiento para la extracción de la imagen digital forense.

Existen dos formas de realizar la instalación de estas aplicaciones y es utilizando la herramienta ADB. Una forma de instalación es copiando los archivos a una ruta o directorio del sistema operativo Android y la otra es realizando la instalación por medio de comandos desde la máquina de análisis. Pero antes de realizar la instalación de las aplicaciones, se necesita la conexión con los otros sistemas operativos Android y se realizó con los siguientes comandos:
Comando para conectarse al a la maquina virtual MV_Android_4-4:
root@kali:/# adb connect 10.0.0.10
Comando para listar los dispositivos conectados:
root@kali:/# adb devices
List of devices attached
10.0.0.10:5555 device

Teniendo el dispositivo MV_Android_4-4 conectado procederemos se procedió a realizar las difentes formas de hacer la intalación de una aplicación en el sistema operativo Android:
Comandos para pasar archivos para su posterior instalación en el sistema operativo Android
adb push /root/Desktop/BusyBox.apk /sdcard/hack

adb push /root/Desktop/KingoRoot.apk /sdcard/hack

El otro modo es la instalación directa, esta se realiza mediante los siguientes comandos:

adb install BusyBox.apk

adb install KingoRoot.apk

adb install root-checker-6-4-6.apk

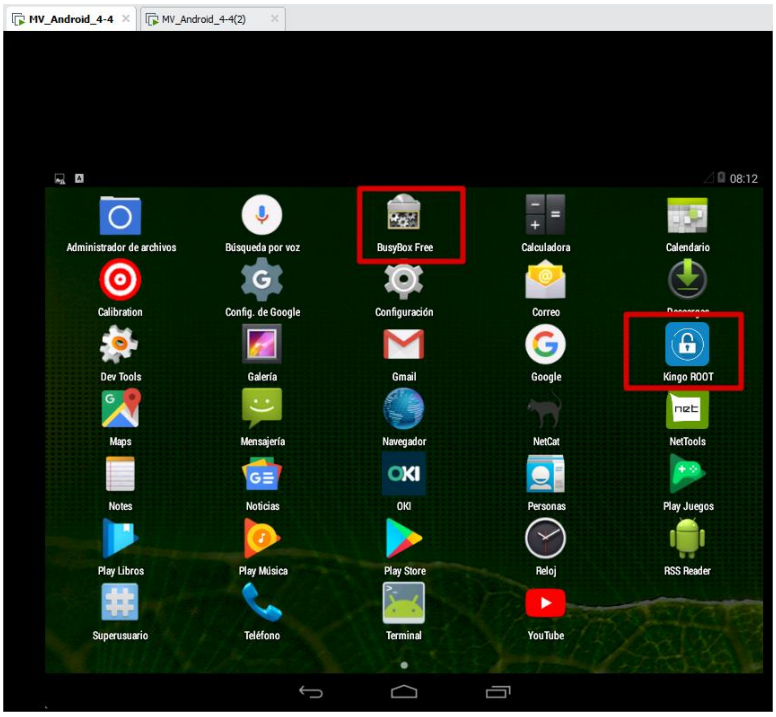

Figura 2.4-21 *Aplicaciones Busy Box y Kingo Root instaladas*

2.4.7 Procedimiento para la extración de la imagen digital forense

Finalmente, se realizó la extracción de la imagen lógica digital forense a las máquinas virtuales MV_Android_4-4 y MV_Android_4-4(2), pero primero se identificó la partición principal y se hizo de la siguiente manera y con los siguientes comandos:

- Se ingresó a la consola de la máquina virtual android con el comando adb shell.

root@kali:/# adb shell

uid=2000(shell)gid=2000(shell)groups=1003(graphics),1004(input),1007(log),1011(adb),1015(sdcard_rw),1028(sdcard_r),3001(net_bt_admin),3002(net_bt),3003(inet),3006(net_bw_stats)@x86:/$

- Se listaron las diferentes particiones del sistema operativo con el comando cat /proc/partitions.

Otra forma de identificar las particiones y de identificar la principal es con el comando buxybox df -f, previa puesta en ejecución de Busy Box.

Se identificó la partición sd1 de los dispositivos MV_Android_4-4 y MV_Android_4-4(2) ubicada en la ruta /dev/block/sd1, pero es de anotar que para cada marca de dispositivo es diferente el nombre de la partición. Entonces se extrajo la imagen lógica digital forense de los dispositivos móviles de la siguiente forma, iniciando en la máquina de análisis con la redirección de puerto, es decir, la comunicación con el sistema operativo se establece con el puerto 5555. Para trasmitir la imagen digital lógica forense se hace por el puerto 5552.

Así mismo, en las maquinas MV_Android_4-4 y MV_Android_4-4(2) se envió la imagen de la partición identificada a la máquina de análisis con el siguiente comando dd if=/dev/block/sda1 | busybox nc -l -p 5552 y posteriormente a la ejecución del comando anterior, se recibió en la máquina de análisis con el siguiente comando nc 127.0.0.1 5552 > imagen.android.dd.

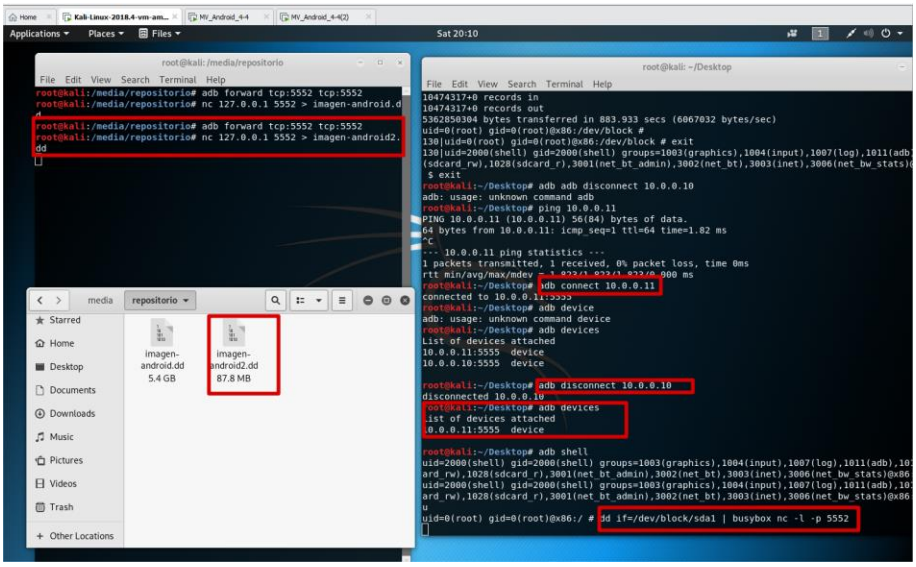

Figura 2.4-22 *Procedimiento de transferencia para MV_Android_4-4(2)*

Por último, cuando finalizó el proceso de transmisión de la imagen digital forense,
se procedió a desconectar las máquinas Android remotamente con el siguiente comando:
adb disconnect 10.0.0.10
adb disconnect 10.0.0.11

Es muy importante a la hora de realizar una extracción de una imagen digital forense mantener la cadena de custodia para ofrecer un soporte evidente y confiable ante una investigación forense y para esto se debe firmar la imagen obtenida mediante un hash, que es un valor único generado por un algoritmo matemático de cifrado y sirve para demostrar la integridad y asegurar que no se ha comprometido la imagen digital forense extraída [83].

Para esta investigación se realizó el hash con el algoritmo criptográfico SHA-512, porque a la fecha no se han encontrado vulnerabilidades que lo comprometan o afecte el nivel de seguridad. Por otra parte, en el proceso de obtención del hash se obtuvo un tiempo estimado de 35 segundos por cada imagen digital forense extraída.

Integridad de la imagen digital forense:

Con el comando sha512sum /ruta/archivo se hace el cálculo de la integridad de las diferentes imágenes digitales forenses extraídas.

Comando: sha512sum imagenandroidwifi.dd
Comando: sha512sum imagenandroidwifi2.dd
Comando: sha512sum imagenandroidwifiredinterna.dd

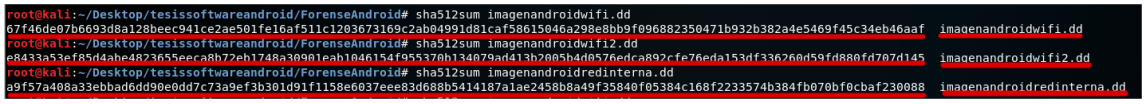

Figura 2.4-23 *Comprobación de la integridad*

Confirmación del hash en cada imagen extraída:

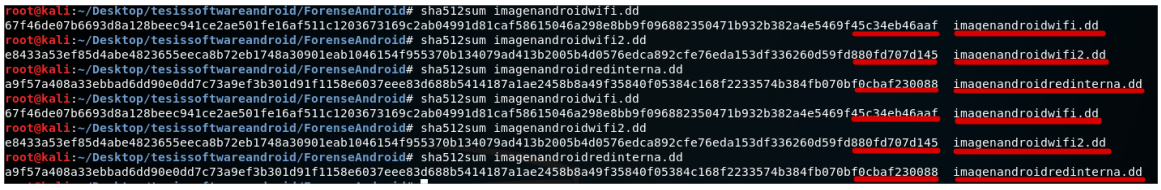

Figura 2.4-24 *Confirmación del hash*

En la imagen anterior se evidencia la creación de un nuevo hash a las diferentes imágenes digitales forenses extraídas y se valida la igualdad del resultado de los hash, por lo tanto, el funcionamiento de este procedimiento es efectivo.

2.4.8 Optimización de la extracción de la imagen forense

Para hacer la optimización de la extracción de la imagen lógica digital forense, se realizó un *script* donde se reúnen todos los pasos mencionados anteriormente de forma manual. Además, se adicionaron unas nuevas recomendaciones que se deben tener en cuenta para que la imagen forense no se pueda corromper.

Una de las recomendaciones más importantes es la desactivación de los otros medios de comunicación del dispositivo para transferir archivos, como son la conexión USB, *Bluetooth* y la red de datos móviles del dispositivo, ya que se requiere que la imagen forense sea los más confiable posible. También se debe detener la función de los botones de apagado, volumen arriba y abajo, cámara, notificaciones y por último el bloqueo de la pantalla.

Teniendo estas recomendaciones atendidas, se procede a realizar la extracción de la imagen digital forense del dispositivo. Después de esto, cuando finalice el proceso de extracción, se habilita de nuevo las opciones que anteriormente se deshabilitaron.

Se inicial el proceso iniciando el script y la captura de tráfico:

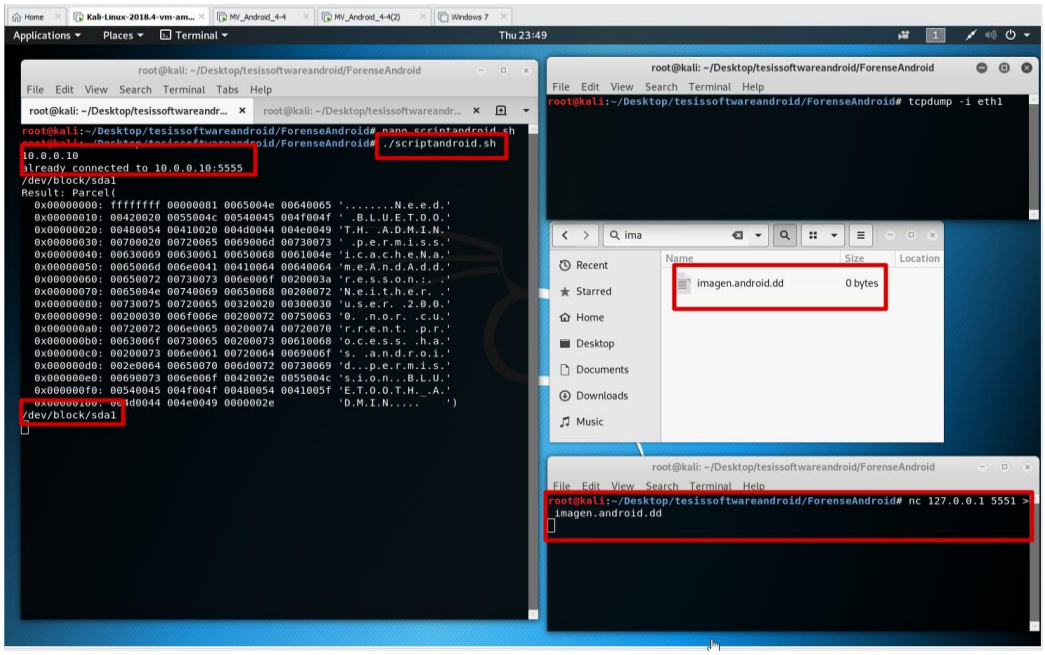

Figura 2.4-25 *Inicio de la extracción de la imagen digital forense optimizada*

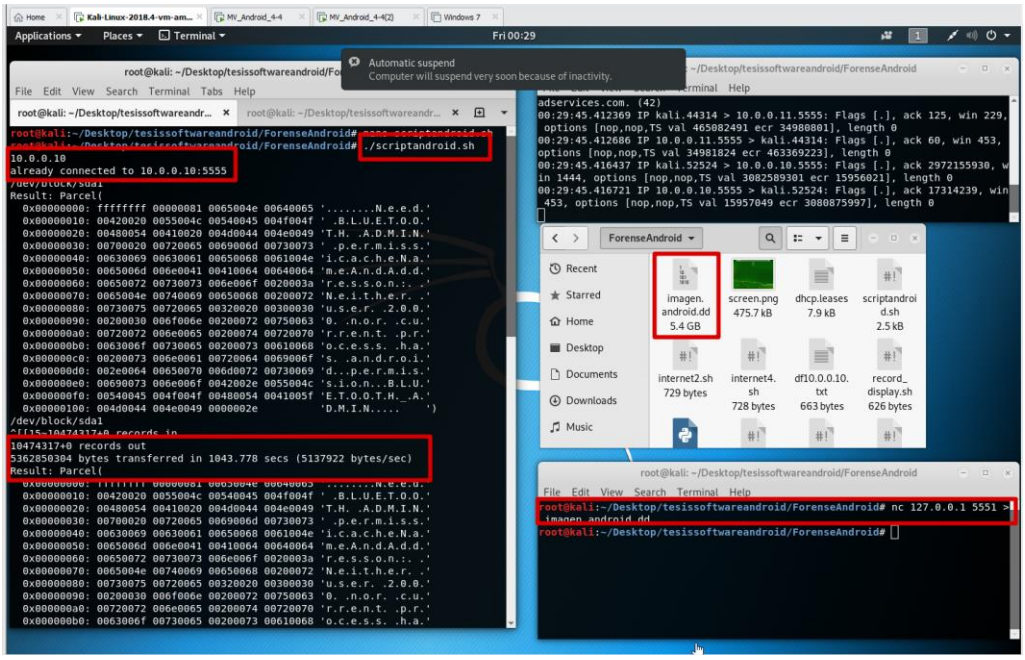

Figura 2.4.26 *Finalización de la extracción de la imagen digital forense optimizada*

El código creado y utilizado para la extracción de la imagen forense con lenguaje Bash, fue el siguiente:

```
#!/bin/bash

#Borrar algun archivo con los siguientes nombres, ya que se van a utilizar.

rm filtroleases.txt

rm leases.txt

rm sortleases.txt

        # Expresion regular para identificar las IP de los dispositivos conectados al servicios del DHCP

        grep -o 'lease.[0-9.]*' /var/lib/dhcp/dhcpd.leases | grep -o '[0-9.]*' > leases.txt

        # Llevar las IPs encontradas a un archivo txt

        # Organizar y retirar lineas duplicadas

        sort leases.txt > sortleases.txt

        uniq sortleases.txt filtroleases.txt

        leases=$(cat filtroleases.txt)

        #echo $leases

        #reemplazar espacios por,reemplazo=$(echo $leases | tr ' ' ,)

        #generar array con scripts a aplicar para contar el contenido

        IFS=', ' read -a ascript <<< "$reemplazo"

        #recorrer array creando dispositivos nuevos

for aplicar in "${ascript[@]}"
```

```
do
    echo "$aplicar"
                # Comando  para conectar con los dispositivos filtrados anteriormente
                  adb connect $aplicar
                # Buscar la particion principal
                adb shell busybox df -h > df$aplicar.txt
                particion=$(grep .*system df$aplicar.txt | grep -o ^/..... | sed 's/ //g')
                # Mostrar la particion
                echo $particion
                 #Deshabilitar componentes que puedan interferir en la data
                #bloquear pantalla
                adb shell input keyevent 26
                          #deshabilitar notificaciones
                            adb shell input keyevent 83
                           #deshabilitar tecla volumen arriba
                          adb shell input keyevent 24
                           #deshabilitar tecla volumen abajo
                          adb shell input keyevent 25
                          #deshabilitar tecla CAMARA
                          adb shell input keyevent 27
                          #Deshabilitar bluetooth
```

```
adb shell service call bluetooth_manager 8

#datos moviles

adb shell svc data disable

 #bloquear  pantalla

adb shell input keyevent 26

 sleep 2

 #Comunicacion para recibir la imagen forense

 adb forward tcp:5551 tcp:5551

# Abrir na terminar nueva y la ejecucion para nombrar la imagen

#gnome-terminal -e nc 127.0.0.1 5551 > imagen.android.dd

 #nc 127.0.0.1 5551 > imagen.android.dd

          gnome-terminal -x bash -c "nc 127.0.0.1 5551 >
          imagenandroid.dd; exec bash"

          #Comando dd para copiar y convertir a bajo

          adb shell "su -c 'dd if=/dev/block/sda1 | busybox nc -l -p 5551'"

          #sleep 2

          #habilitar componentes desactivados anteriormente

          #habilitar notificaciones

          adb shell input keyevent 83

          #habilitar tecla volumen arriba

          adb shell input keyevent 24
```

```
                    #habilitar tecla volumen abajo
                    adb shell input keyevent 25
                    #habilitar tecla CAMARA
                    adb shell input keyevent 27
                    #habilitar bluetooth
                    adb shell service call bluetooth_manager 8
                    #habilitar datos moviles
                    adb shell svc data disable
                    # debloquear pantalla
                    adb shell input keyevent 26
                    adb shell input keyevent 82
                #Comandos utilizados para pruebas
                #gnome-terminal -x bash -c nc 127.0.0.1 5551 > imagen.android.dd
                #konsole -e nc 127.0.0.1 5551 > imagen.android.dd
                #sleep 2
                #nc 127.0.0.1 5551 > imagen.android.dd
                 #echo "voy a ejecutar el comando"
                 #exec busybox df -h
                #sleep 2
                sha512sum imagenandroidwifi.dd
# Muestra de nuevo los dispositivos
```

adb devices

Inicia nuevamente el ciclo de extracción a los otros dispositivos en el servicio de DHCP encontrados con sistema operativo Android

Ya validando que en el ambiente controlado se puede realizar la extracción de la imagen digital forense de formar exitosa, se procederá a realizar un cambio en la configuración de la arquitectura propuesta para realizar la conexión entre la máquina de análisis y los equipos Android de forma inalámbrica. Para esto se presentará tres escenarios de configuración para determinar los tiempos que tarda el proceso de extracción de la imagen lógica digital forense.

- Escenario uno:

  No se realiza ningún cambio a la arquitectura presentada anteriormente. La máquina de análisis en la interfaz de red virtualizada eth0 sigue de modo puente con la interfaz de red inalámbrica del dispositivo HP y la interfaz de red virtualizada eth1 está en la red interna de la herramienta VMware. Así mismo, los sistemas operativos Android virtualizados siguen con la interfaz de red interfaz eth0 de modo red interna VMware.

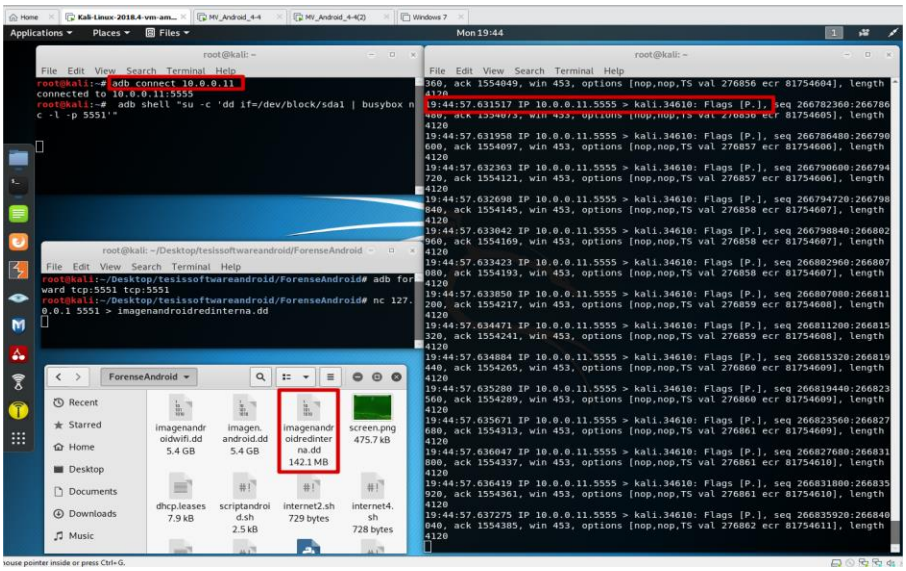

Figura 2.4-27 *Extracción de imagen digital forense escenario uno*

- Escenarios dos:

En la máquina de análisis la interfaz de red virtualizada eth0 sigue en modo puente con la interfaz de red inalámbrica del dispositivo HP y a los sistemas operativos Android virtualizados se hace el cambio en la interfaz eth0 virtualizada en modo puente con la interfaz de red inalámbrica del dispositivo HP.

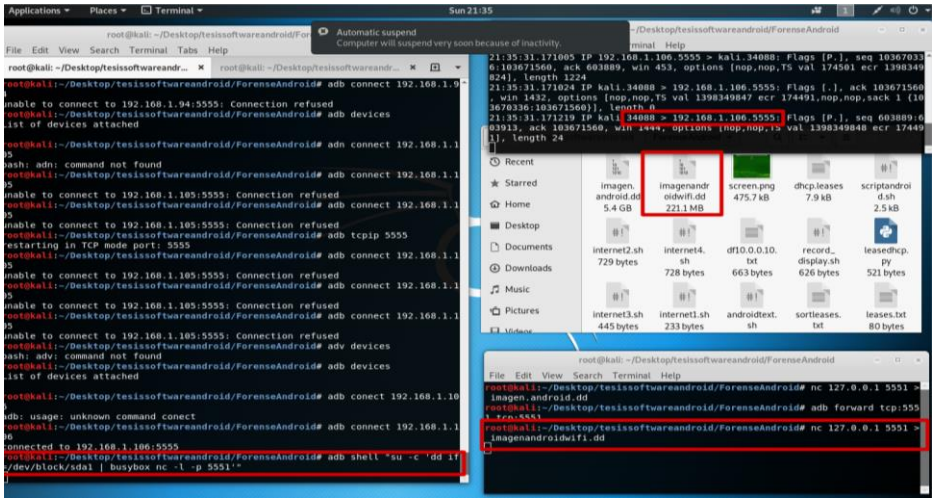

Figura 2.4-28 *Extracción de imagen digital forense escenario dos*

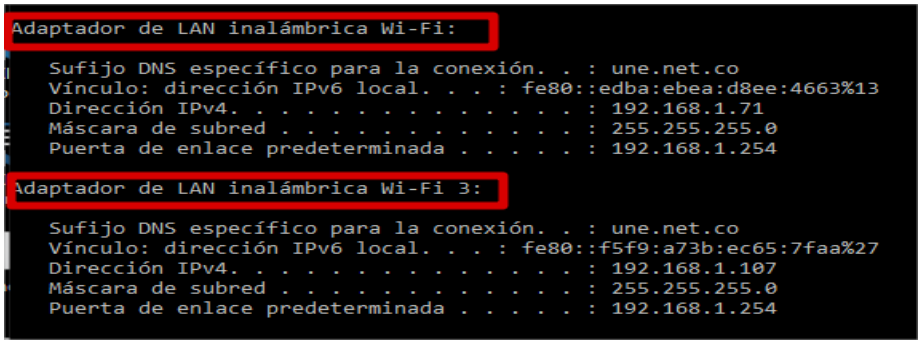

Figura 2.4-29 *Adición de nueva conexión red inalámbrica en el escenario dos*

- Escenario tres:

En la máquina de análisis no se realiza ningún cambio, la interfaz virtualizada de red eth0 está de modo puente con la interfaz de red inalámbrica del dispositivo HP y los sistemas operativos Android virtualizados se hace el cambio en la interfaz eth0 virtualizada en modo puente con la nueva interfaz de red (Marca ALFA Network modelo AWUS036ACH). Por otra parte, es de

indicar que la configuración de estos dispositivos se realizará en la de 5 GHz con el fin de determinar los tiempos al momento de realizar la extracción de la imagen digital forense. Este escenario replica un escenario real, todos los dispositivos físicos conectado mediante WIFI.

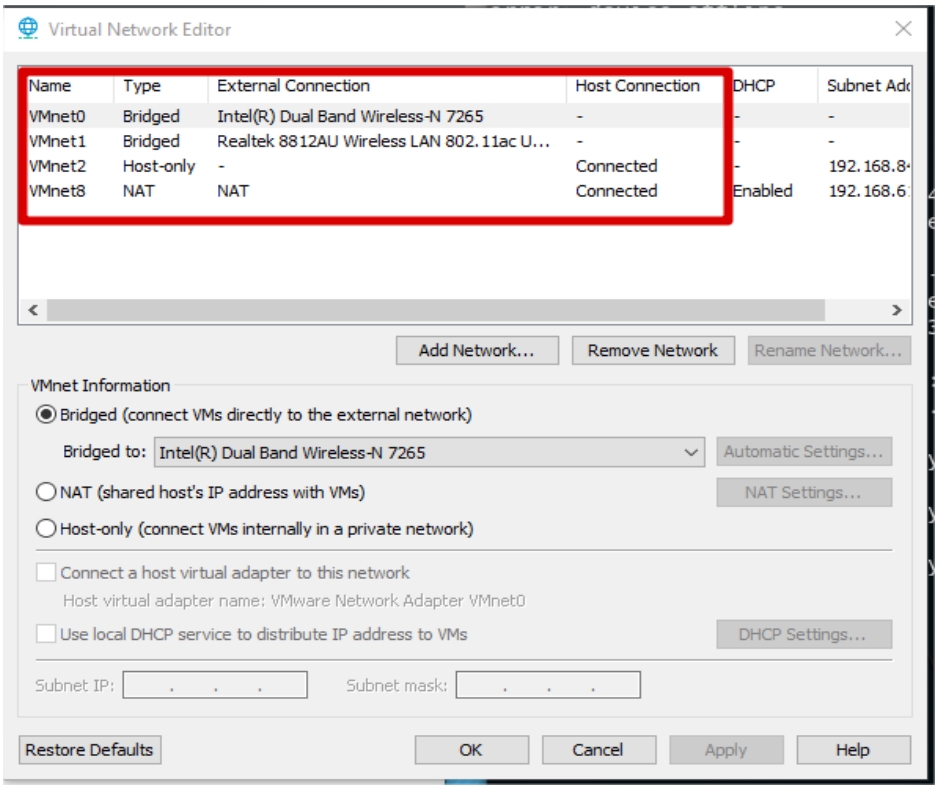

Figura 2.4-30 *Cambio en el escenario tres, configuración en ambiente virtual*

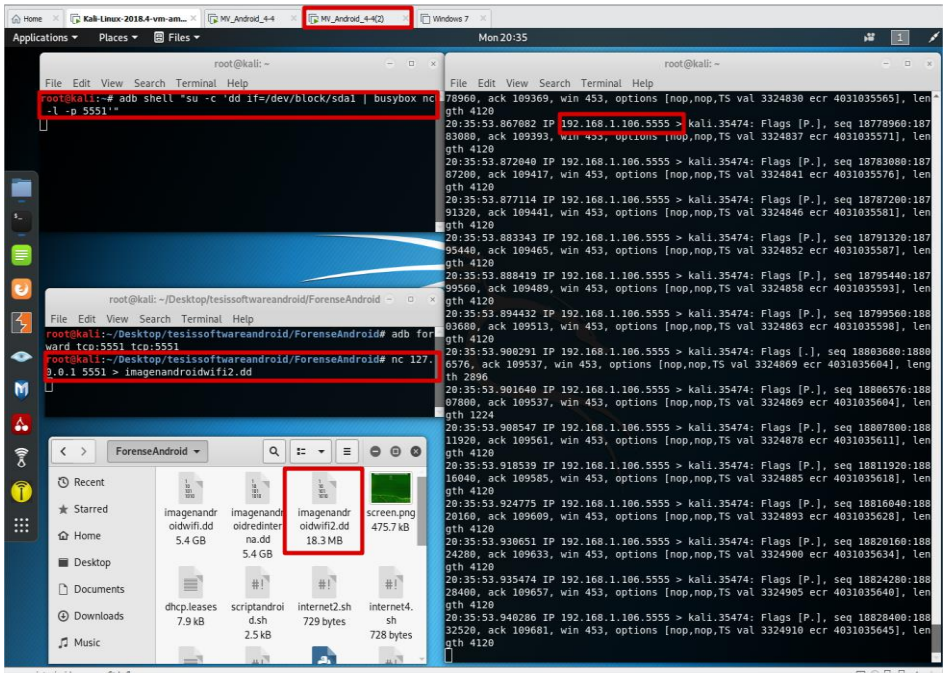

Figura 2.4-31 *Extracción de imagen digital forense escenario tres*

2.4.9. Caso de uso real

Para esta investigación se realizó una extracción de una imagen digital forense de modo convencional cableada USB con la ayuda de la herramienta especializada en casos forenses Magnet AXIOM Process, esto con el fin de hacer una comparación en tiempos en relación con la investigación realizada. Esta extracción se realizó a un dispositivo móvil SAMSUNG Galaxy J3, modelo SM-J320M, con serie 42008937aad44400 y con capacidad de almacenamiento 4.71 GB.

Inicialmente se debe conectar el dispositivo móvil con sistema operativo Android al equipo principal, como se muestra en la Figura 2.4-30, donde está instalado la herramienta AXIOM magnetic. Asimismo, se debe cambiar los parámetros en el dispositivo móvil para activar el modo desarrollador y depuración USB. Para activar el modo desarrollador en versiones superiores a Android 4.4 se debe ir al Menú—Ajustes—Acerca del dispositivo—Número de compilación (Dar clic en el número de compilación de 4 a 8 repeticiones hasta que indique que está en modo desarrollador). Con respecto a la activación de la depuración USB se debe validar en Menú—Ajustes—Opciones Desarrollador—Depuración USB.

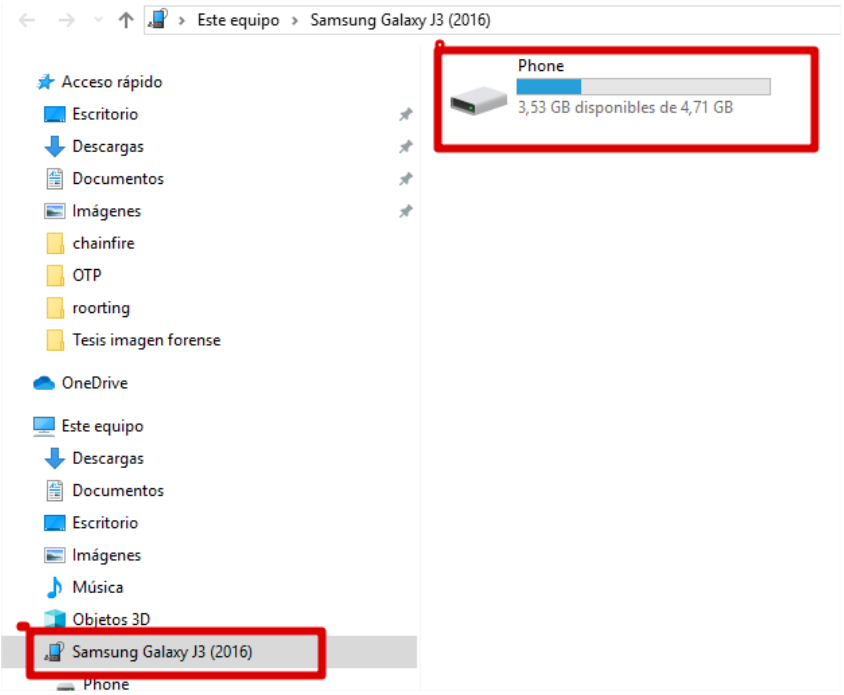

Figura 2.4-32 *Conexión de dispositivo Samsung Galaxy J3*

Con la herramienta Magnetic Axiom se debe crear un incidente de seguridad donde se detalle el número de caso, fecha de creación, ubicación donde se almacenará la evidencia recolectada y descripción donde se indicará la información del dispositivo a realizar la extracción de la imagen forense.

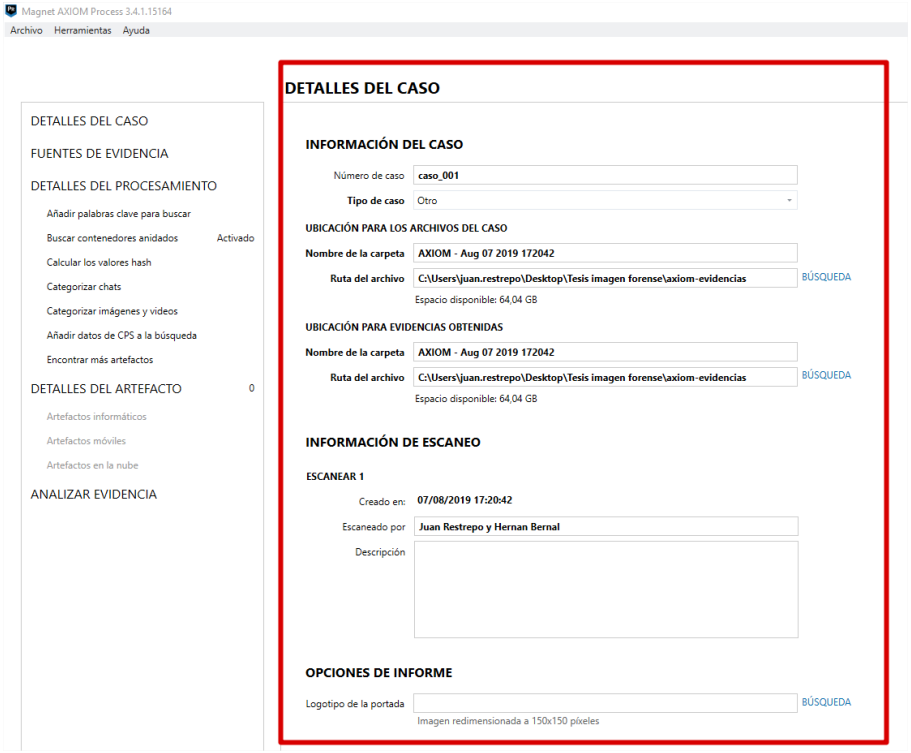

Figura 2.4-33 *Creación del caso*

Seguidamente se debe indicar a qué tipo de dispositivo se le debe realizar la extracción de la imagen forense. Además, la forma de recolección de la evidencia que para este caso se realizó la extracción de la imagen digital forense a un dispositivo móvil con sistema operativo Android y una adquisición rápida con ADB (desbloqueado).

Figura 2.4-34 *Tipo dispositivo*

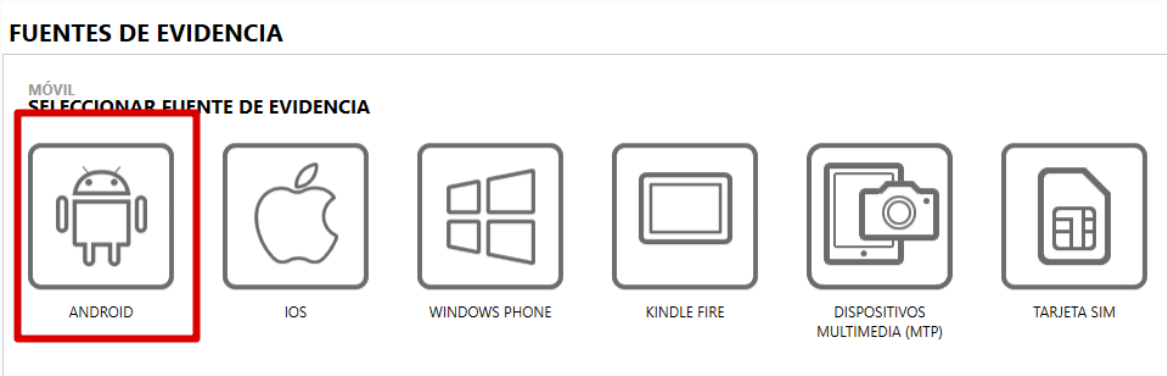

Figura 2.4-35 *Indicación de sistema operativo Android*

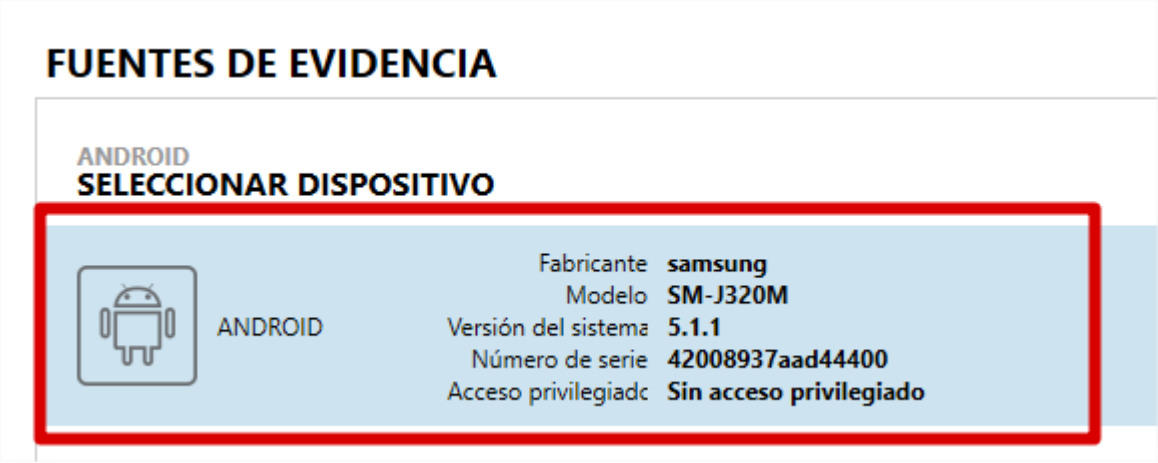

Figura 2.4-36 *Dispositivo Móvil reconocido mediante conexión USB*

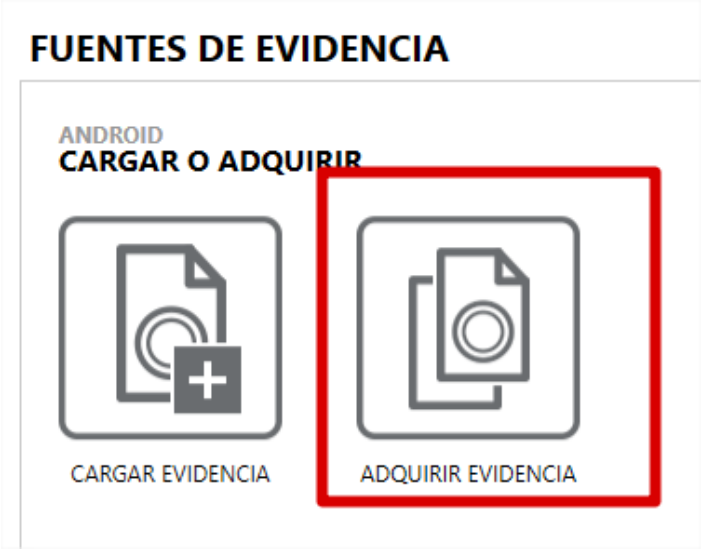

Figura 2.4.37 *Adquisición de la imagen forense*

Figura 2.4.38 *Adquisición de la imagen forense*

Figura 2.4.39 *Formato de la imagen forense*

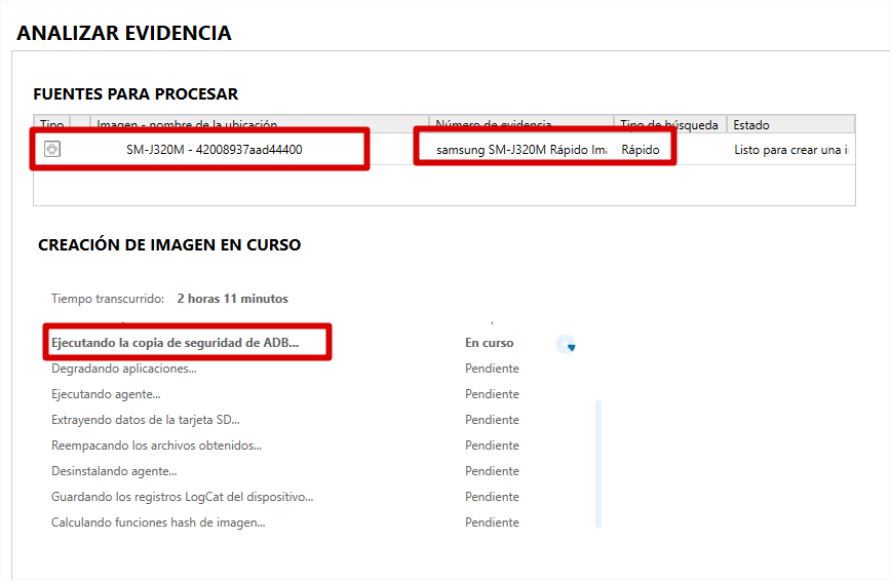

Figura 2.4-40 *Proceso de la adquisición de la imagen*

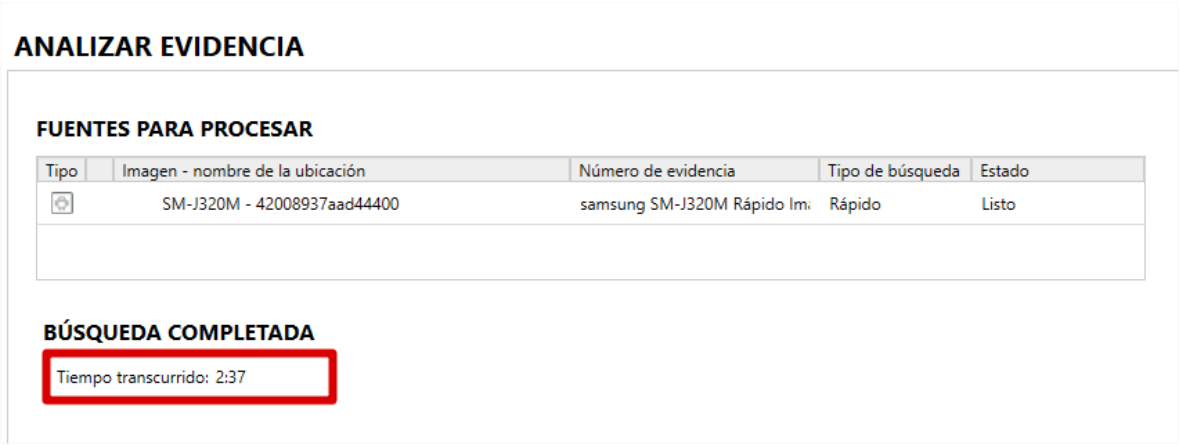

Figura 2.4-41 *Tiempo de extracción de la imagen forense*

3. Resultados

3.1 Determinación de las arquitecturas de Android a utilizar

El resultado obtenido con el listado de cambios significativos por versión dio cuenta, que no se impide la toma de imágenes digitales para un eventual análisis forense ni de programaciones especiales que hará esta operación. Sin embargo, se debe tener en cuenta que, al realizar una App para Android con Android Studio, se debe usar una versión adecuada dependiendo de la versión del sistema operativo donde se vaya a instalar, con el fin que no se presente el mensaje de advertencia por no soportar cierta versión.

En la siguiente tabla se muestran los cambios significativos por versión, para tenerlo en cuenta en la extracción de las imágenes.

Resumen de las versiones del Sistema Operativo Android						
Número de versión	**Nombre de la versión**	**Año de lanzamiento**	**Cambios significativos**	**Arquitectura**	**Sistema de archivos**	**Seguridad**
1.0	Apple Pie	2008	Primera versión comercial, no estética pero funcional. Introdujo Android Market e integró a Gmail.	Primera Arquitectura	YAFFS	N/A
1.1	Banana Bread	2009	Corrección de los errores de la versión anterior.	Primera Arquitectura	YAFFS	N/A
1.5	Cupcake	2009	Mejora su diseño en pantalla principal, rotación en la pantalla	Primera Arquitectura	YAFFS	N/A

			automáticamente, teclado virtual en la pantalla principal y soporte de Bluetooth.			
1.6	Donut	2009	SDK de Android, donde los desarrolladores podrían crear aplicaciones, Android Market y diversidad en los tamaños de pantalla.	Primera Arquitectura	YAFFS	N/A
2.1	Eclair	2009	Navegación GPS con Google Maps, agregó fondos de pantallas animadas y agregó la opción de activación de voz en el teclado.	Primera Arquitectura	YAFFS	N/A
2.2	Froyo	2010	WiFi, *hostspot*, paso de contactos por Bluetooth, acciones con comando de voz. Mejora el rendimiento	Primera Arquitectura	YAFFS	Android introduce la extensión ASEC para admitir funcionabilidades de aplicaciones en

			con la incorporación de Dalvik en el procesamiento y el motor V8 JavaScript en la navegación del sistema operativo.			tarjetas SD y cifrarlas.
2.3	Gingerbread	2010	API para creación de juegos con gráficos avanzados, lanzamiento con la compatibilidad de la comunicación cercana (NFC) y administración de la batería donde muestra de forma detallada el consumo de ésta.	Primera Arquitectura	EXT4	Control de seguridad para eliminación de forma remota de aplicaciones maliciosas para evitar la exposición de cuentas de usuarios.

3.0	Honeycomb	2011	Soporte a diferentes tamaños de pantallas, compatible con *tablets*, barra del sistema y configuración rápida para inicial a la información principal.	Primera Arquitectura	EXT4	En esta versión se enfatiza en el cifrado completo de los datos como las aplicaciones, cuentas de usuarios, archivos etc. Además, descifrar el dispositivo mediante una contraseña.
4.0	Ice Cream Sandwich	2011	Pantallas personalizables, carpetas de las apps, administración del control del uso de datos de red. Comunicación entre dispositivos por medio de la tecnología NFC.	Primera Arquitectura	EXT4	Facilidad para que las apps administren las autenticaciones y las sesiones de forma segura.

4.2	Jelly Bean	2012	Nuevos idiomas, asistencia al móvil, metereología con Google Now y multiusuario (en un mismo dispositivo tener varias cuentas de usuarios).	Primera Arquitectura	EXT4	* API KeyChain proveedor de almacenamiento y generación de claves exclusivas y privadas que otras apps no pueden utilizar ni ver. * Posibilidad de inhabilitar comprobaciones de las apps para hacer transferencias de apps por medio de USB. *. Verificación de aplicaciones antes de su instalación. * Utilización de OpenSSL para tener compatibilidad con TLS en las

						versiones 1.1 y 1.2.
4.4	Kitkat	2013	Resoluciones en 4K, multitareas, marcador Inteligente (prioriza contactos y lugares con más aperturas, oculta el texto para ver la pantalla completa y comandos de voz).	Primera Arquitectura	EXT4	* Mejoras en seguridad con la incorporación de mejores algoritmos criptográficos para la firma de aplicaciones y protección de clave criptográficas para el cifrado del disco.

5.0	Lollipop	2014	Notificaciones en la pantalla de bloqueo, multipantalla, conexión a la nube, mejora el rendimiento y el consumo de batería, *Material Desing (*mejora el desplazamiento en el dispositivo).	Arquitectura Actual	EXT4	* Cambios en los perfiles de usuarios con privilegios administrativos. * Actualización de Web View a Chomium M37 para corregir los permisos que se otorgaban a las apps para acceder a la cámara y el micrófono.
6.0	Marshmallow	2015	Múltiples usuarios, configuración rápida, permisos manuales que soliciten las aplicaciones para acceder a información sensible, batería más inteligente y eficiente, botón de inicio para activar	Arquitectura Actual	EXT4	* Usuarios pueden controlar los permisos que solicite la aplicación. * Disponibilidad de tecnología de almacenamiento adaptables para reemplazar ASEC.

			Google Now.			
7.0	Nougat	2016	Biométrico, administración en la energía con menos consumo, carga rápida, miltiventanas y realidad virtual.	Arquitectura Actual	EXT4	* Cifrado a nivel de archivos para aislar y proteger cada cuenta de usuario. * Acceso restringido del directorio de apps a archivos privados.
8.0	Oreo	2017	Actualizaciones más rápidas, los íconos redondos pueden tomar otras formas (iconos adaptativos). Notificaciones con multimedia y posibilidad de silenciarlas, restricciones en las	Arquitectura Actual	EXT4	* Incompatibilidad con SSLv3. * Cambio de directorio de las APK siempre debe de usar sourceDir, no como antes que se suponía que si terminaba en -1 o -2 ya tenía su directorio.

			aplicaciones de segundo plano para mejorar el consumo de la batería.			* Trabajar por separado el contenido Web y el proceso de cada aplicación.
9.0	Pie	2018	Modernización del sistema operativo Android con batería inteligente y auto brillo. Herramientas para administrar el uso del dispositivo.	Arquitectura Actual	EXT4	* Se retira la compatibilidad con ASEC y mejoras en el cifrado con ARC4. * Apps no debe compartir información con apps con permisos UNIX, para que solo una app pueda acceder a los datos privados.

Tabla 3.1-2 *Cambios en las arquitecturas en las versiones de Android[5][84][85][86]*

[5] La tabla es de realización propia de los autores de esta investigación

3.2 Selección de un método apropiado de cifrado

En la tabla 3.2-1, se puede evidenciar las mediciones que se obtuvieron con el proceso de cifrado.

ALGORITMO		TIEMPO DE CIFRADO MINUTOS	CPU MÁXIMA	RAM MÁXIMA
3DES	DES_EDE3	06:41	100,7%	0,7%
AES	AES128	02:31	46,5%	0,8%
	AES192	02:36	51,3%	0,8%
	AES256	02:35	43,9%	0,8%
BLOWFISH	BLOWFISH128	02:40	65,2%	1,0%
	BLOWFISH192	02:47	61,9%	0,8%
	BLOWFISH256	02:49	61,1%	0,8%
	BLOWFISH384	02:52	61,6%	0,8%
	BLOWFISH448	02:55	63,1%	0,8%
TWOFISH	TWOFISH128	02:29	55,8%	0,8%
	TWOFISH192	02:28	57,1%	0,8%
	TWOFISH256	02:31	55,3%	0,8%
THREEFISH	THREEFISH256	02:37	40,5%	0,7%
	THREEFISH512	02:17	47,2%	0,7%
	THREEFISH1024	02:21	60,1%	0,7%
CAST	CAST128	02:57	88,4%	1,0%
	CAST256	02:56	100,8%	1,0%
ICE_ENCRYPT	ICE_ENCRYPT	03:35	30,7%	0,9%
IDEA	IDEA_CBC_SHA	07:13	24,7%	1,0%
	IDEA	04:03	95,4%	0,8%
RC5	RC5_128	02:53	84,1%	1,0%
	RC5_192	02:54	82,7%	1,0%

ALGORITMO		TIEMPO DE CIFRADO MINUTOS	CPU MÁXIMA	RAM MÁXIMA
	RC5_256	02:58	94,7%	1,0%
	RC5_384	02:00	74,4%	1,0%
	RC5_512	03:02	100,7%	1,0%
RC6	RC6_128	02:49	51,8%	0,7%
	RC6_192	02:52	79,8%	0,6%
	RC6_256	02:57	52,5%	0,7%
	RC6_384	02:56	62,3%	0,7%
	RC6_512	03:00	58,1%	0,7%
SALSA20	SALSA20_128	02:09	36,2%	1,0%
	SALSA20_256	02:11	31,2%	1,0%
SERPENT	SERPENT_128	03:45	98,7%	0,8%
	SERPENT_192	03:46	100,3%	0,8%
	SERPENT_256	03:50	99,7%	0,8%
SHACAL2	SHACAL2_128	02:40	57,8%	0,7%
	SHACAL2_192	02:43	54,8%	0,7%
	SHACAL2_256	02:43	57,6%	0,7%
	SHACAL2_384	02:46	55,3%	0,7%
	SHACAL2_512	02:49	72,4%	0,7%

Peor puntaje Mejor puntaje

Tabla 3.2-3 *Comportamiento del hardware al realizar el cifrado*[6]

[6] La tabla es de realización propia de los autores de esta investigación en Microsoft Excel.

Lo que se pudo apreciar con estos resultados en el comportamiento del *hardware* al realizar el cifrado, es que algunos de los métodos que actualmente se utilizan como los más confiables, por ejemplo AES, toman más tiempo para este proceso y la CPU tiene un mayor consumo, aunque esto último no es generalizado.

Un algoritmo como AES, que se tiene como bastante confiable para cifrar información y es utilizado en dispositivos de varias marcas para la conexión inalámbrica, tiene un consumo alto de CPU aunque tiene uno de los marcadores más bajos en cuanto a tiempo de cifrado. Sin embargo, un algoritmo que no hace cifrado por bloques sino por *stream*, como lo es Salsa20, tiene menor consumo de CPU que los algoritmos más usados y toma menos tiempo para el cifrado. El consumo de memoria no es significante en ninguno de los métodos.

En este caso, el algortimo Salsa20 es el escogido para la metodología propuesta en esta investigación, tanto por el consumo de CPU y tiempo de cifrado, como también el hecho que no haya tenido reportes de ataques cuando este algoritmo hace 20 rondas para completar el cifrado. De hecho, con la premisa que mientras más bits para la clave aumenta la seguridad, se escogió Salsa20/20-256. Es preciso advertir que para la extracción de una imagen forense en un dispositivo móvil, no sólo la seguridad es el factor más importante, también lo es la velocidad con que se hace el cifrado y el consumo de los recursos de ese dispositivo. Habrán comentarios que el algoritmo AES es el mejor para estos procedimientos; en verdad AES es en estos momentos uno de los mejores algoritmos de cifrado y de hecho está presente en la mayoría de los enrutadores WiFi, pero para el efecto de esta investigación, el que mejor resultados ofrece en los factores mencionados es Salsa20.

En la tabla 3.2-2, se muestran las métricas del *hardware* obtenidas para el proceso de descifrado.

ALGORITMO		TIEMPO DE DESCIFRADO MINUTOS	CPU MÁXIMA	RAM MÁXIMA
3DES	DES_EDE3	06:45	100,7%	0,7%
AES	AES128	03:39	56,8%	0,8%
	AES192	02:45	59,5%	0,8%
	AES256	02:47	60,5%	0,8%
BLOWFISH	BLOWFISH128	02:37	59,6%	0,8%
	BLOWFISH192	02:40	59,8%	0,8%
	BLOWFISH256	02:39	57,8%	0,8%
	BLOWFISH384	02:38	57,1%	0,8%
	BLOWFISH448	02:44	60,8%	0,8%
TWOFISH	TWOFISH128	02:22	58,3%	0,8%
	TWOFISH192	02:23	56,8%	0,8%
	TWOFISH256	02:25	74,8%	0,8%
THREEFISH	THREEFISH256	02:17	49,8%	0,7%
	THREEFISH512	02:17	45,8%	0,7%
	THREEFISH1024	02:23	56,0%	0,7%
CAST	CAST128	02:28	69,5%	1,0%
	CAST256	02:39	99,3%	1,0%
ICE_ENCRYPT	ICE_ENCRYPT	02:40	34,9%	0,9%
IDEA	IDEA_CBC_SHA	05:14	26,0%	1,0%
	IDEA	04:09	95,7%	0,8%
RC5	RC5_128	02:31	60,6%	1,0%
	RC5_192	02:33	58,8%	1,0%

ALGORITMO		TIEMPO DE DESCIFRADO MINUTOS	CPU MÁXIMA	RAM MÁXIMA
	RC5_256	02:36	88,1%	1,0%
	RC5_384	02:36	59,5%	1,0%
	RC5_512	02:37	84,1%	1,0%
RC6	RC6_128	02:45	50,8%	0,6%
	RC6_192	02:48	58,9%	0,6%
	RC6_256	02:51	74,8%	0,7%
	RC6_384	02:55	51,8%	0,7%
	RC6_512	02:59	64,8%	0,7%
SALSA20	SALSA20_128	02:02	34,1%	1,0%
	SALSA20_256	02:39	31,6%	1,0%
SERPENT	SERPENT_128	03:27	100,3%	0,8%
	SERPENT_192	03:29	97,4%	0,8%
	SERPENT_256	03:33	99,3%	0,8%
SHACAL2	SHACAL2_128	02:34	75,4%	0,7%
	SHACAL2_192	02:36	67,4%	0,7%
	SHACAL2_256	02:39	76,8%	0,7%
	SHACAL2_384	02:39	57,5%	0,7%
	SHACAL2_512	02:42	72,5%	0,7%

Tabla 3.2-2 *Comportamiento del hardware al realizar el descifrado*[7]

Así como fue evidente en el proceso de cifrado, en el proceso de descifrado tuvo unos resultados mucho mejores que en el cifrado para el algoritmo Salsa20. Este algortimo usó menos tiempo y

[7] La tabla es de realización propia de los autores de esta investigación en Microsoft Excel.

menos recurso de CPU en el descifrado que en el cifrado. Por lo tanto, el dispositivo que va a tener el repositorio de la imagen extraída, puede ejecutar sin inconvenientes el algortimo de descifrado de Salsa20 al comenzar el análisis forense. Es consistentemente más rápido que AES (uno de los algoritmo de cifrado más utilizados) y es utilizado para aplicaciones dónde es más importante la velocidad que la confianza [87], pero da seguridad a la información.

ALGORITMO		SC (SEGURIDAD DEL CIFRADO)	UC (USO DEL CIFRADO EN LA COMUNIDAD)	VC (VELOCIDAD DEL CIFRADO)	CONVERSIÓN MINUTOS DE CIFRADO A DECIMALES	MINUTOS EN DECIMALES DE CIFRADO PARA 48	NIC (NÚMERO DE IMÁGENES CIFRADAS EN 30 MINUTOS)	NIC
3DES	DES_EDE3	3	4	1	6,683333333	64,16	0,47	1
AES	AES128	2	5	4	2,516666667	24,16	1,24	3
	AES192	3	5	4	2,6	24,96	1,20	3
	AES256	3	5	4	2,583333333	24,8	1,21	3
BLOWFISH	BLOWFISH128	2	3	4	2,666666667	25,6	1,17	2
	BLOWFISH192	3	3	4	2,783333333	26,72	1,12	2
	BLOWFISH256	3	3	4	2,816666667	27,04	1,11	2
	BLOWFISH384	4	3	4	2,866666667	27,52	1,09	2
	BLOWFISH448	4	3	4	2,916666667	28	1,07	2
TWOFISH	TWOFISH128	2	4	4	2,483333333	23,84	1,26	3
	TWOFISH192	3	4	4	2,466666667	23,68	1,27	3
	TWOFISH256	3	4	4	2,516666667	24,16	1,24	3
THREEFISH	THREEFISH256	3	4	4	2,616666667	25,12	1,19	2
	THREEFISH512	4	4	4	2,283333333	21,92	1,37	3
	THREEFISH1024	5	4	4	2,35	22,56	1,33	3
CAST	CAST128	2	4	4	2,95	28,32	1,06	2
	CAST256	3	4	4	2,933333333	28,16	1,07	2
ICE_ENCRYPT	ICE_ENCRYPT	1	4	3	3,583333333	34,4	0,87	1
IDEA	IDEA_CBC_SHA	2	5	1	7,216666667	69,28	0,43	1
	IDEA	2	5	3	4,05	38,88	0,77	1
RC5	RC5_128	2	2	4	2,883333333	27,68	1,08	2
	RC5_192	3	2	4	2,9	27,84	1,08	2

ALGORITMO		SC (SEGURIDAD DEL CIFRADO)	UC (USO DEL CIFRADO EN LA COMUNIDAD)	VC (VELOCIDAD DEL CIFRADO)	CONVERSIÓN MINUTOS DE CIFRADO A DECIMALES	MINUTOS EN DECIMALES DE CIFRADO PARA 48	NIC (NÚMERO DE IMÁGENES CIFRADAS EN 30 MINUTOS)	NIC
	RC5_256	3	2	4	2,966666667	28,48	1,05	2
	RC5_384	4	2	4	2	19,2	1,56	4
	RC5_512	4	2	3	3,033333333	29,12	1,03	2
RC6	RC6_128	2	4	4	2,816666667	27,04	1,11	2
	RC6_192	3	4	4	2,866666667	27,52	1,09	2
	RC6_256	3	4	4	2,95	28,32	1,06	2
	RC6_384	4	4	4	2,933333333	28,16	1,07	2
	RC6_512	4	4	4	3	28,8	1,04	2
SALSA20	SALSA20_128	2	4	4	2,15	20,64	1,45	4
	SALSA20_256	3	4	4	2,183333333	20,96	1,43	4
SERPENT	SERPENT_128	2	4	3	3,75	36	0,83	1
	SERPENT_192	3	4	3	3,766666667	36,16	0,83	1
	SERPENT_256	3	4	3	3,833333333	36,8	0,82	1
SHACAL2	SHACAL2_128	2	5	3	2,666666667	25,6	1,17	2
	SHACAL2_192	3	5	4	2,716666667	26,08	1,15	2
	SHACAL2_256	3	5	4	2,716666667	26,08	1,15	2
	SHACAL2_384	4	5	4	2,766666667	26,56	1,13	2
	SHACAL2_512	4	5	4	2,816666667	27,04	1,11	2

Peor puntaje Mejor puntaje

Tabla 3.2-3 *Resultados del cálculo por cada variable en el cifrado*[8]

[8] La tabla es de realización propia de los autores de esta investigación en Microsoft Excel.

En la Tabla 3.2-3, se muestra la cuantificación explicada en el numeral 2.2 de este trabajo. Acá entonces es donde la premisa anterior de escoger el algortimo Salsa20 para el cifrado de la imagen, cobra sentido cuando los valores se ingresan a la fórmula propuesta y seguidamente, en la Figura 3.2-1 se grafica este resultado. Es conveniente recordar que el resultado de la fórmula mientras más cerca está a cero (0), se puede inferir que el método de cifrado es mejor para la propuesta.

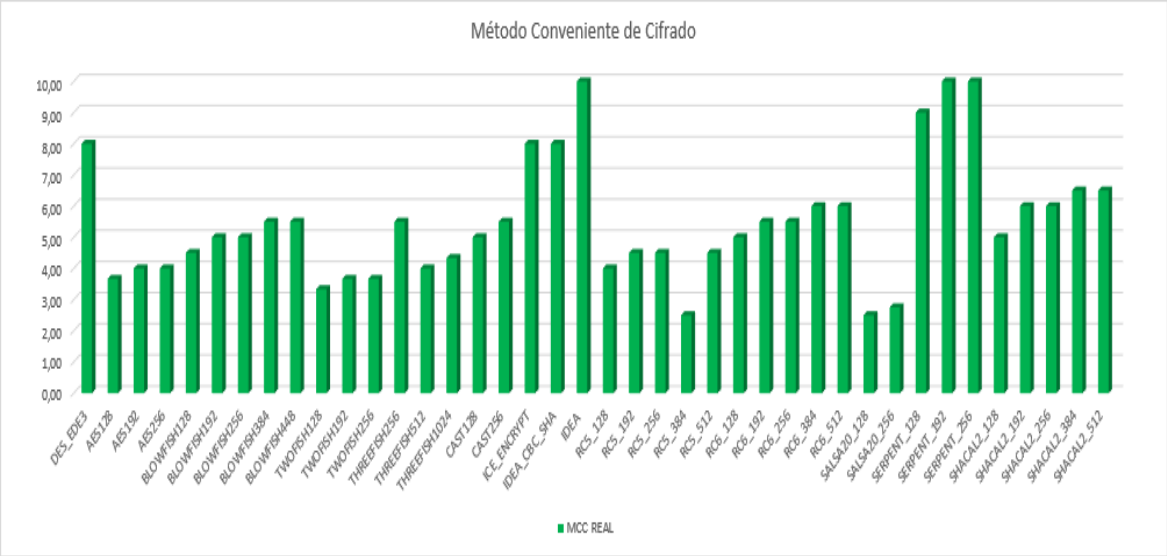

Figura 3.2-1 *Cuantificación del muestreo de los métodos de cifrado*[9]

3.3 Selección de extracción de imágenes incrementales, teniendo en cuenta la compresión y la integridad

El uso de **rsync** para la extracción de imágenes incrementales arrojó un resultado satisfactorio en cuanto a la sencillez del método. Sin embargo, hubo muchas desconexiones del protocolo SSH para hacer el envío por WiFi de la imagen incremental.

[9] La figura es de realización propia de los autores de esta investigación en Microsoft Excel.

Para una imagen de prueba de 5.36 GB, el tiempo que se tomó para el proceso de imagen incremental, fue de 6:10 minutos.

Haciendo esta misma prueba en un entorno inalámbrico, el proceso de llevar una imagen incremental de 5.36 GB se tomó un tiempo de 48:20 minutos.

Aunque los tiempos tomados del proceso con **rsync** son altos para estas pruebas, este método está en constante evolución que posiblemente pueda mejorarse con el tiempo y sobre todos, que las redes inalámbricas han mejorado en su velocidad de transmisión.

En la siguiente tabla se puede visualizar que los mejores tiempos de compresión no son directamente proporcional con la tasa de compresión. Sin embargo, la diferencia con los métodos con mejor compresión no es mucha. El uso de CPU es crítico, porque todos los métodos evaluados tienen un consumo bastante alto, llegando inclusive al 180%. El uso de memoria no es crítico en ninguno de los métodos.

ALGORITMO		TAMAÑO ORIGINAL EN BYTES	TIEMPO H:MM:SS	TASA DE COMPRESIÓN %	USO CPU %	USO MEMORIA %	TAMAÑO FINAL BYTES
ZIP	ZIP	5239471104	0:26:51	33,19	99,7	0,1	3500527522
GZIP	GZIP	5239471104	0:07:54	33,19	99,3	0	3500527268
BZIP2	BZIP2	5239471104	0:16:21	33,66	98,7	0,2	3475915836
XZ	XZ	5239471104	1:01:10	40,44	100	2,4	3120418460
RAR	RAR	5239471104	0:19:56	33,49	180,1	2,4	3484513586
ZPAQ	ZPAQ	5239471104	6:13:25	41,877	99,7	2,8	3045331056
TAR	TAR GZIP	5239471104	0:06:39	33,189	98,5	0,1	3500528766
	TAR BZIP2	5239471104	0:27:52	33,659	99,7	0,2	3475893039

Peor puntaje | Mejor puntaje

Tabla 3.3-4 *Resultados del cálculo por cada variable en la compresión*

Por lo tanto, el método que se escogió para este trabajo es el TAR GZip, se aprovecha tanto la compresión por GZip y el empaquetamiento con TAR. Además, es un comando que se encuentra tanto en Linux como en Android.

En cuanto al tema de cifrado *hash*, se hicieron las pruebas de tiempo y arrojó que para una imagen de 5.2 GB, el proceso de *hash* puede durar unos 5 segundos máximo para generar la firma con cualquiera de los métodos probados, sólo lo diferencia la longitud de la firma. Se probaron SHA-2

(224, 256, 384 y 512) y MD5. Aunque MD5 es un algoritmo que ha reportado en la literatura con problemas de seguridad, aun se sigue utilizando. En la tabla 3.3-2 se visualizan los resultados de los factores medidos.

ALGORITMO	TAMAÑO ORIGINAL EN BYTES	TIEMPO H:MM:SS	USO CPU %	USO MEMORIA %
MD5	5239471104	0:00:59	42,1	0
SHA1	5239471104	0:01:01	46,4	0
SHA256	5239471104	0:00:59	88,7	0
SHA512	5239471104	0:01:00	71,5	0

Tabla 3.3-2 *Resultados del cálculo por cada factor en la firma de integridad*

3.4 Validación de la metodología propuesta bajo un ambiente controlado

Se obtuvieron las imágenes digitales forense de los dispositivos MV_Android_4-4 y MV_Android_4-4(2) en un tiempo estimado de dos horas y media por cada procedimiento en un ambiente con comportamiento real. Es por esta razón para la actualizada el canal de WIFI no es el más idóneo para la extracción de imágenes digitales lógicas forenses de volúmenes muy elevados, aunque puede ser utilizado para la extracción de imágenes digitales forense de bajo volumen de almacenamiento o información localizada importante para una investigación forense.

Es importante resaltar que las imágenes digitales lógicas forenses son de igual tamaño por ser extraídas de dispositivos clonados. Por otro parte se puede evidenciar que el sistema distribuido y remoto puede adoptarse para realizar la extracción de imágenes digitales forenses de bajo volumen de almacenamiento y pueda ser comparada con las metodologías que se utilizan en la actualidad.

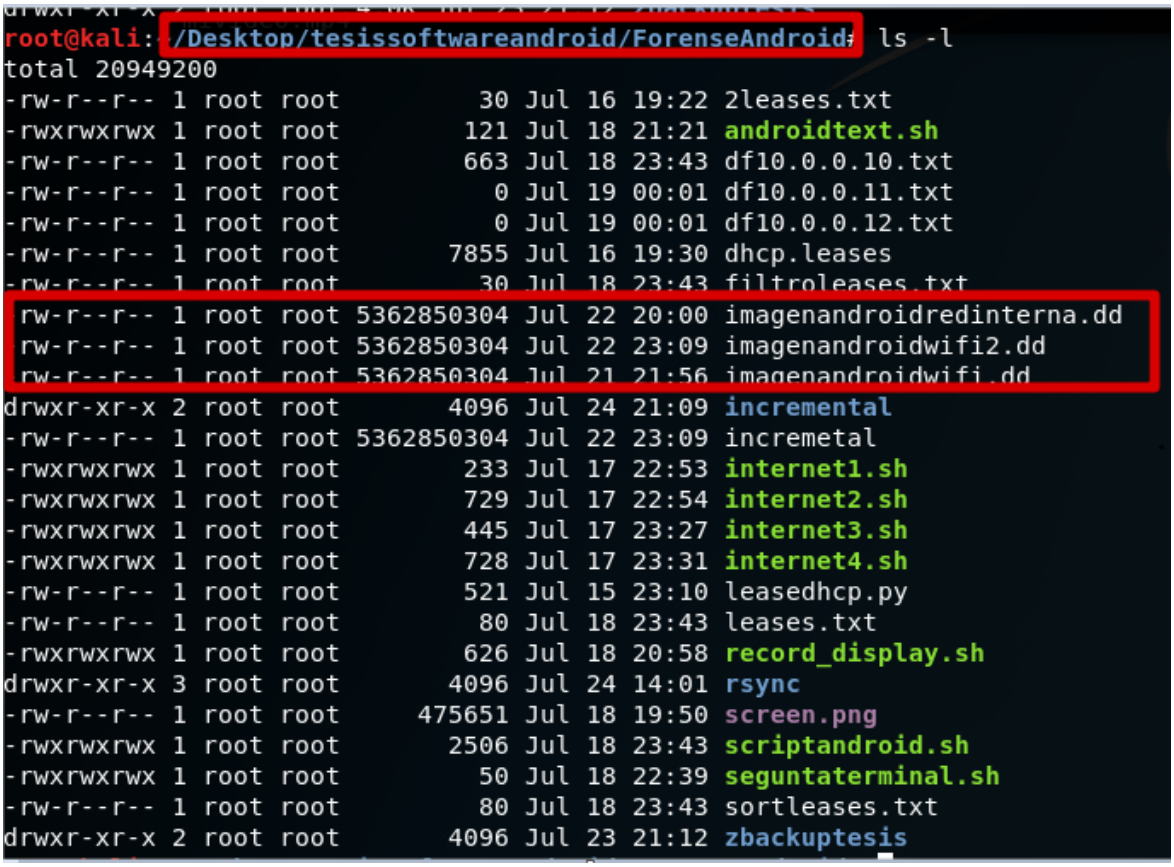

Figura 3.4-1 *Repositorio de las imágenes forenses extraídas*

Al momento de hacer la extracción de la imagen digital forense se evidenció una lentitud en la máquina HP, por lo que se verifica en el administrador de tareas donde se puede observar un alto consumo en la memoria y la CPU, identificando a la aplicación VMWare Workstation como uno de los causantes de este evento. Esto quiere decir que el ambiente controlado no contaba con computadores con grandes recursos, pero, aun así, se pudo llevar a cabo todo el desarrollo de lo que se planteaba en este trabajo. Inclusive, se demuestra la extracción distribuida de imágenes de dos dispositivos.

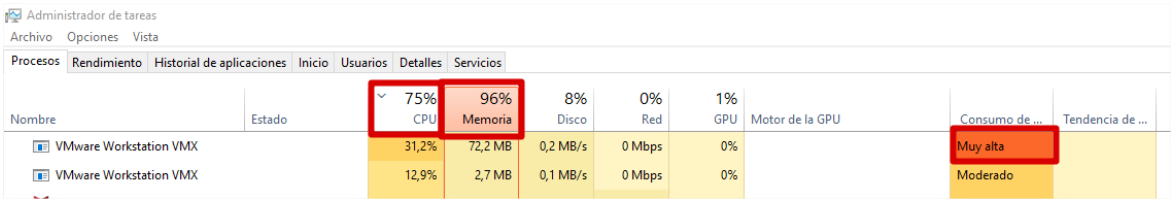

Figura 3.4-2 *Consumo de memoria*

La aplicación Autopsy es una de las herramientas con que cuenta el profesional forense para hacer esta clase de análisis. En este apartado, se puede evidenciar que la imagen extraída del dispositivo Android era funcional y que contenía toda la información que éste proporcionaba.

En las siguientes figuras se plasma la evidencia de la imagen extraída con la aplicación Autopsy.

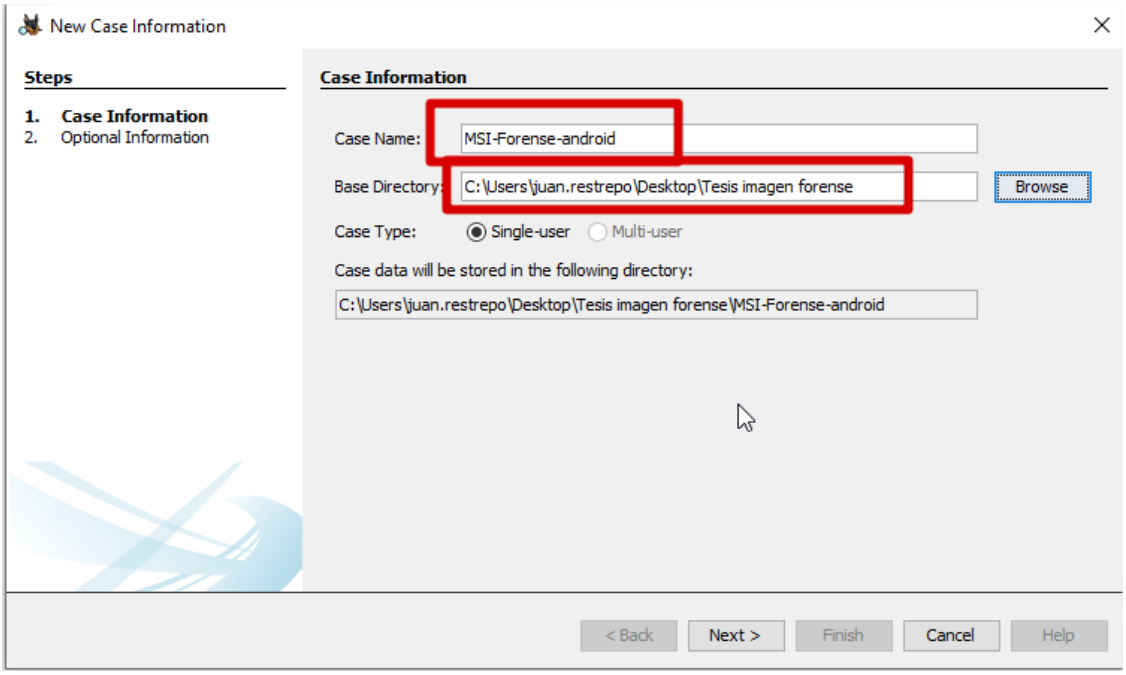

Figura 3.4-3 *Ubicación de la imagen forense*

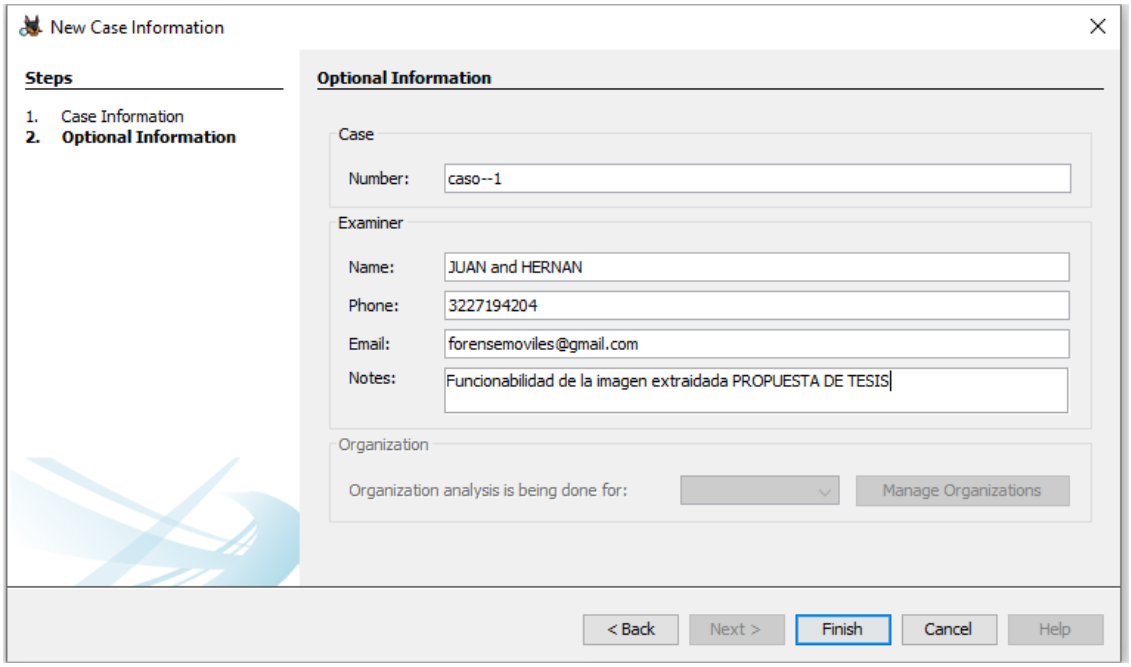

Figura 3.4-4 *Creación del caso*

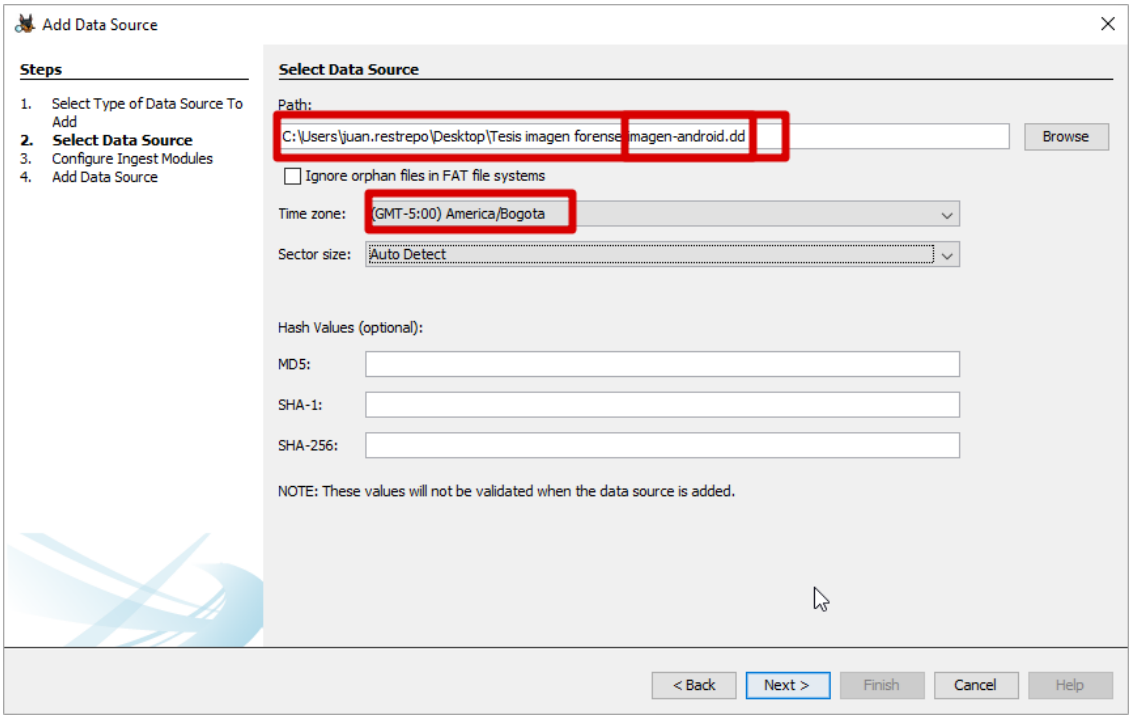

Figura 3.4-5 *Agregación de la imagen extraída*

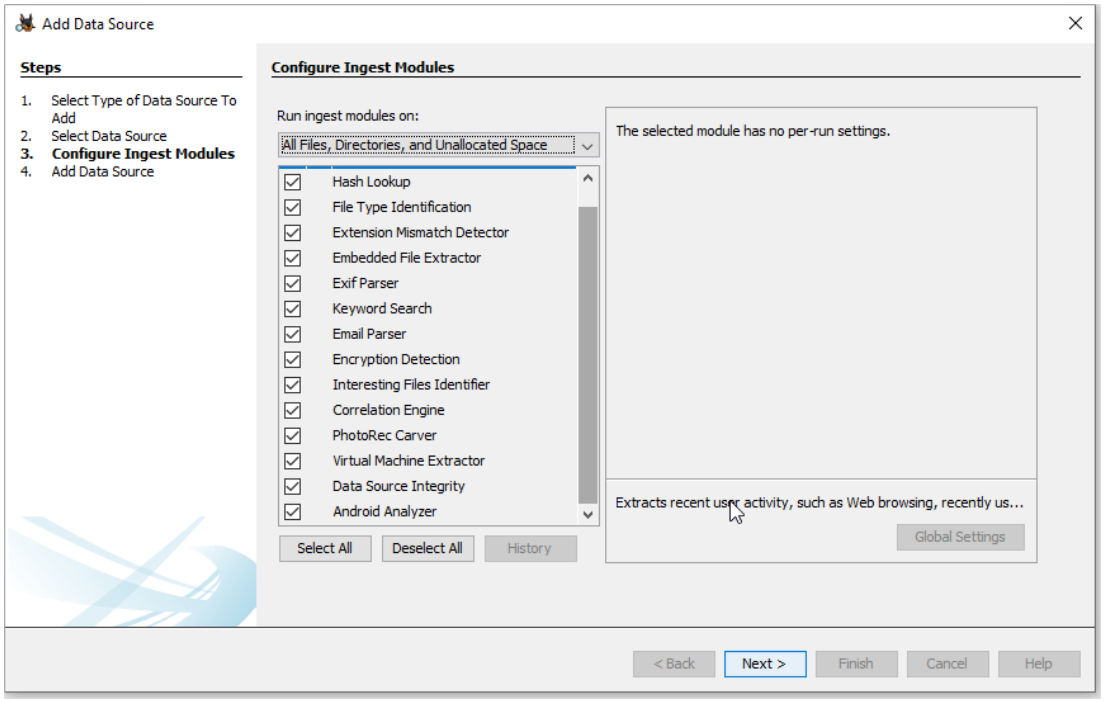

Figura 3.4-6 *Módulos para el análisis forense*

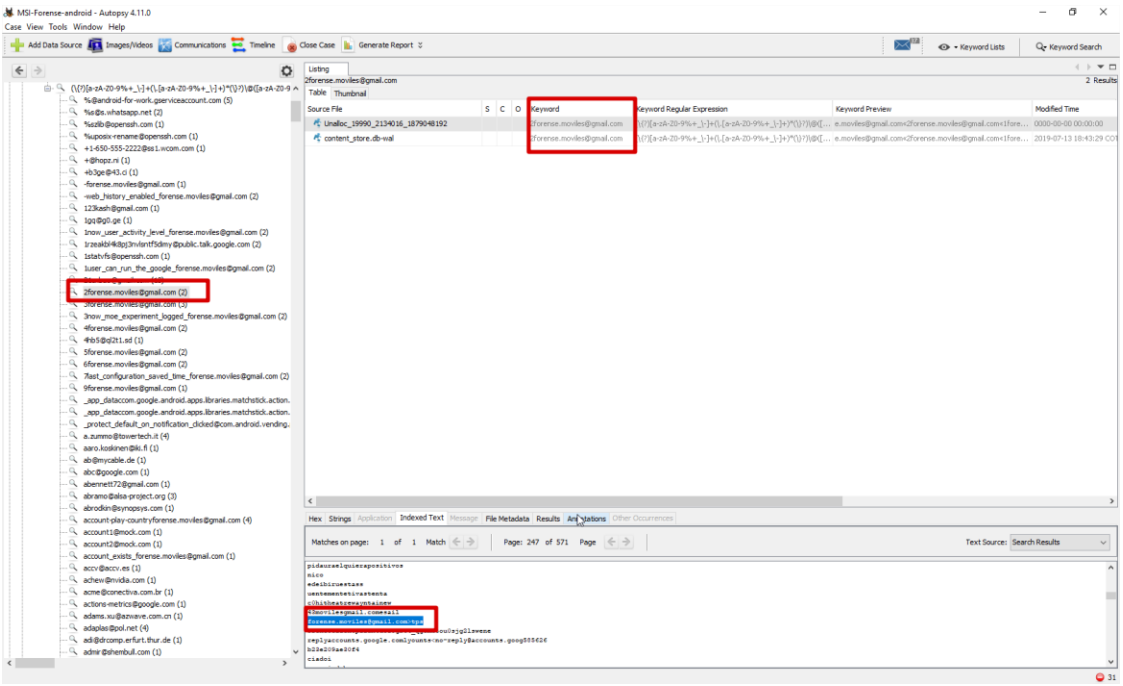

Figura 3.4-7 *Información obtenida del análisis forense (1)*

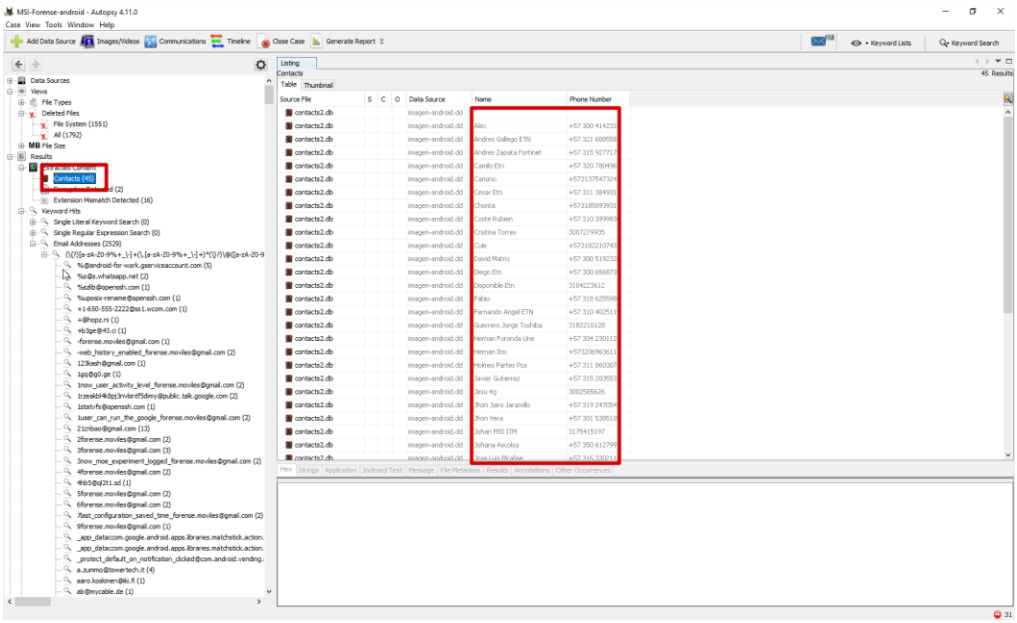

Figura 3.4-8 *Información obtenida del análisis forense (2)*

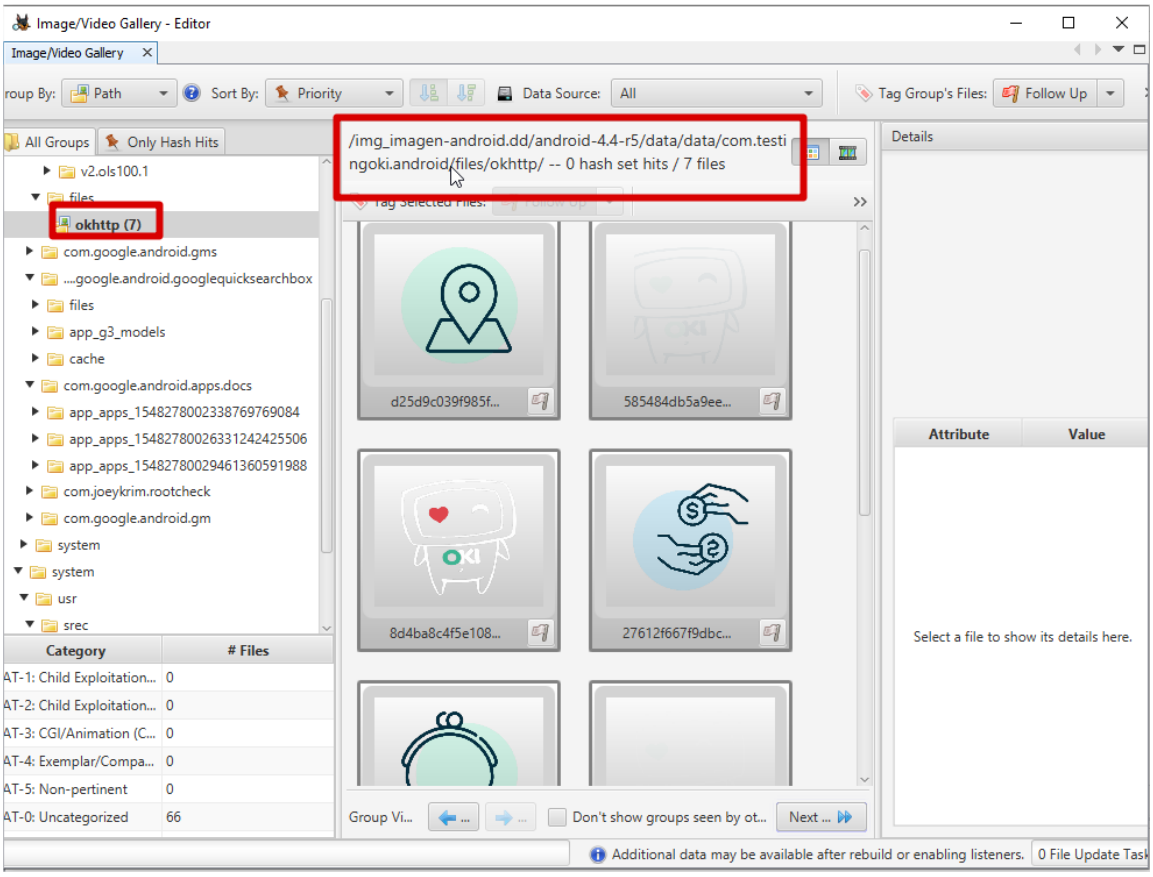

Figura 3.4-9 *Información obtenida del análisis forense (3)*

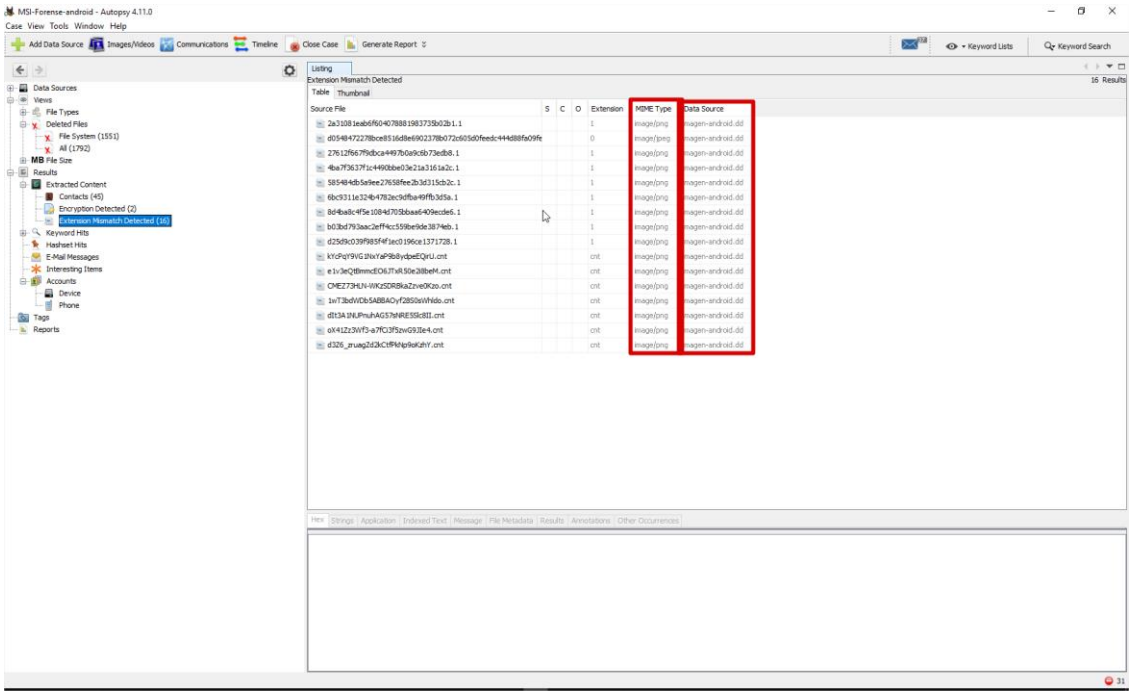

Figura 3.4-10 *Información obtenida del análisis forense (4)*

La siguiente tabla muestra los tiempos de extracción de la imagen digital lógica forense, la configuración de las interfaces y el tamaño de imagen lógica forense de los diferentes escenarios.

Escenario	Interfaz Física	Interfaz virtual	Modo conexión	Banda	Tamaño de la imagen Forense	Tempo Estimado de extracción/minutos
Uno		Red Interna	Red interna		5.36 GB	16
Dos	Compartida	Una tarjeta de red	Modo puente	5 GHz	5.36 GB	30
Tres	Dividida	Dos tarjetas de red	Modo puente	5 GHz	5.36 GB	154

Tabla 3.4-5 *Tiempo estimado en la extracción de la imagen digital forense a través de WIFI*

De la evidencia tomada de la tabla anterior, se observa en el escenario uno que en la red interna de la aplicación de virtualización VMware tiene un tiempo más corto en el proceso de extracción de la imagen forense, seguida por el escenario dos que tiene un modo puente y que comparte la misma interfaz de red inalámbrica y por último, el escenario tres con un mayor tiempo en la

extracción de la imagen digital forense y está dado por el modo puente con diferentes interfaces de red inalámbrica, tanto para la máquina de análisis como para las máquinas virtuales con sistema operativo Android; este último entorno simula un ambiente real.

Ya conocido el comportamiento de la extracción de la imagen digital forense en los diferentes escenarios tanto para un ambiente virtualizado como un entorno real, se adoptó el escenario uno para esta investigación, ya que no interfirió en la obtención de los otros planteamientos. Sin embargo, se aclara que el escenario tres tiene un comportamiento similar a un entorno real por las conexiones a las interfaces de red físicas y presenta un tiempo de extracción de la imagen forense mayor.

3.5 Identificación de la metodología propuesta

Como la propuesta es de una metodología para la extracción de imágenes forenses incrementales de dispositivos Android y a través de redes WIFI, se pueden encontrar los siguientes resultados de los objetivos específicos:

- Como la mayoría de los dispositivos celulares que están siendo utilizados en las empresas son con sistema operativo Android y por encima (e inclusive) de la versión 5.0 (Lollipop), se consideró entonces la posición de los archivos y nombres de las carpetas según la estructura definida en este trabajo como la segunda arquitectura. Por lo tanto, no se consideró la arquitectura anterior a la versión 5 de Android.
- Aunque en la metodología tradicional de extracción de imágenes forenses lo normal es que no se cifre la imagen para ser enviada al dispositivo de almacenamiento, se pudo verificar que hay algunos fabricantes de *hardware* que sí lo toman en consideración y que por lo tanto para la metodología propuesta en este trabajo, no va en contravía de la seguridad de la información. De hecho, se propuso y se probó con toma de tiempo y uso de recursos informáticos, que el método escogido ayuda a que esos tiempos y usos de recursos no se incrementen y al mismo que no se incremente el costo de la metodología con relación a la tradicional. Sin embargo, se debe tener en cuenta que la velocidad de transmisión de esas imágenes por un medio como WiFI, a pesar del desarrollo de dispositivos con mayor

velocidad de transmisión, no asegura y no se evidenció que sea más rápido que cuando se realiza con cables conectados a puertos USB.

- Se propuso una técnica para hacer imágenes incrementales con algunos comandos que se mencionaron anteriormente, sin embargo, se notó que, con algunas pruebas a los sistemas operativos virtualizados, se producen desconexiones fortuitas. Aun así, la técnica propuesta de imagen incremental cumple con el objetivo. Igualmente, la compresión de esa imagen incremental y la firma digital (*hash*), son adiciones necesarias y seguras para la extracción digital de la imagen forense.
- El entorno o ambiente controlado que se construyó para la validación de las técnicas antes mencionadas arrojó unos resultados satisfactorios en cuanto a que la metodología propuesta es válida, tomando en consideración que la velocidad de transmisión es similar que la tradicional, pero que, si se evalúa la prontitud de un caso que se necesite contener las pruebas de un evento, fue una excelente alternativa además que se puede tomar imágenes de varios dispositivos al mismo tiempo.

4. Conclusiones y recomendaciones

4.1 Conclusiones

En el objetivo “determinar cuáles arquitecturas de los actuales sistemas operativos Android son apropiadas de cara a la extracción distribuida, el cifrado y formato de las imágenes forenses”, se estableció que, aunque la arquitectura del sistema operativo Android tuvo algunos cambios a lo largo de su trayectoria, no es impedimento para realizar la extracción de la imagen digital forense a sus diferentes versiones.

Con el objetivo “seleccionar un método apropiado de cifrado, de cara al envío seguro de la imagen a través de redes WiFi”, se logró exitosamente la selección correcta del método más apropiado de cifrado para la investigación. Se probaron 13 métodos de cifrado con diferentes tamaños de clave, dando un total de 40 métodos probados, cuantificando su costo computacional. Por los factores a tener en cuenta, Salsa20/20 demostró que es un método de cifrado conveniente para la investigación y la validación de la metodología.

Teniendo en cuenta el objetivo “proponer las técnicas para la extracción de imágenes incrementales tomando en consideración su factor de compresión e integridad”, se logró establecer que las imágenes incrementales son posibles de hacer, que era una de las hipótesis intrínsecas en el proyecto, pero se realizan de igual forma que un *backup* incremental. No se escogen los archivos para el *backup* sino que se escoge una partición. Como Android se ve, opera y maneja las instrucciones como Linux, es evidente saber dónde están los datos del sistema y del usuario. Por eso **rsync** es una aplicación vigente para hacer imágenes incrementales y para envío a repositorios externos; WiFi no es un medio muy rápido así la imagen incremental sea pequeña. El comando “tar” y “gzip” en combinación pueden hacer una compresión rápida y empaquetado eficaz de esa imagen. El método Zpaq es considerablemente más lento que cualquiera de los probados, que, en esta investigación, se demoró más de 10 horas en completar la compresión a un archivo de 5.2 GB.

Con respecto al objetivo “validar la metodología propuesta bajo un ambiente controlado a partir de casos de estudio, estableciendo un marco comparativo para la medición de fortalezas y debilidades”, la virtualización de máquinas fue una solución apropiada para el ambiente de pruebas, ya que, al no contar con el suficiente *hardware*, la posibilidad de emularlos con estas

máquinas virtuales arroja resultados tanto o casi parecidos como los reales. Sin embargo, se necesita de características más elevadas que con las que se contaba, para poder aislar apropiadamente las posibles interferencias que se puedan presentar.

En cuanto al objetivo general "proponer una metodología para la extracción distribuida de imágenes forenses incrementales de dispositivos móviles con sistema operativo Android a través de redes WIFI", se concluye que aunque no se mejora la eficiencia en tiempo de la metodología propuesta en la investigación con respecto a la metodología tradicional o alámbrica, es una excelente alternativa cuando no se tenga del *hardware* o no se cuente con el suficiente tiempo para aislar o contener el acto delictivo que ocurrió. Para dispositivos móviles con sistema operativo Android con alta capacidad de almacenamiento, no es la metodología más recomendada; pero al tener imágenes incrementales, esos archivos serán más pequeños que la imagen original, para los cuales la metodología es una posible solución a la toma rápida de imágenes.

4.2 Recomendaciones

Se recomienda que para las pruebas en ambiente controlado y si se utiliza máquinas virtuales e imágenes de sistemas operativos, que estos últimos sean conseguidos de fuentes fidedignas u oficiales, ya que posiblemente no estén bien elaboradas esas imágenes y se presenten inconvenientes al ponerlos en funcionamiento en las máquinas virtuales.

Igualmente, se sugiere que para cualquier tipo de pruebas y que se utilizan equipos de cómputo, que no tengan otras aplicaciones fuera de los que trae el sistema operativo por defecto en su mínima configuración, con el fin que no se presenten interferencias en el procesamiento real.

Es importante resaltar que para futuros montajes de un ambiente controlado se mejore el hardware de la máquina anfitriona, ya que al momento de hacer la extracción de la imagen digital forense se evidenció una lentitud en las máquinas. Esto quiere decir que el ambiente controlado no contaba con computadores con grandes recursos, pero, aun así, se pudo llevar a cabo todo el

desarrollo de lo que se planteaba en este trabajo. Inclusive, se demuestra la extracción distribuida de imágenes de dos dispositivos.

Los requerimientos mínimos que se necesitan para un entorno virtual de la máquina anfitriona, es que se cuente con más seis *cores* de procesador, con memoria mínima de 16 GB y con mínimo dos interfaces de red inalámbrica que soporten la banda de 5 GHz que la máquina tenga la opción de modo virtualización.

También se debe considerar para esta máquina anfitriona, que el tema de software en la virtualización se trabaje con sistema operativo Kali Linux como mínimo en la versión 2017.1 y sistemas operativos Android como mínimo versión 4.4. Para la virtualización como tal, se recomienda mínimo VMWare Workstation 12 y VirtualBox 6.0.

Muy importante tener en cuenta, que se debe contar con acceso a internet para las actualizaciones que posiblemente se necesiten en el ambiente a nivel de software.

Las empresas para tener esta metodología en su ambiente corporativo deben considerar tener un equipo de comunicaciones inalámbricas con un estándar de conexión de muy buena velocidad de transmisión. Igualmente, deben contar con un dispositivo de almacenamiento acorde a la cantidad de teléfonos celulares y su posible crecimiento de unidades. También deben considerar contar con un equipo de tecnología de la información incluyendo personal de telecomunicaciones, seguridad y analistas de datos para un constante mejoramiento de la metodología.

Una recomendación adicional es que se continúe con la investigación para la mejora de los tiempos y de los procesos, puesto que la tecnología seguirá mejorando el *hardware* y los consumos de recursos que hace el *software*. La extracción de imágenes forenses inalámbrica será probablemente una alternativa mejor que la tradicional alámbrica.

Bibliografía

[1] D. Ussía, "Móviles personales y corporativos con una separación de entornos segura - Think Big", *Think Big/Empresas*, 2015. [En línea]. Disponible en: https://bit.ly/2YviRpG. [Accedido: 15- Dic- 2017].

[2] Google Developers, "Distribution dashboard-Android Developers", *Android Developers*, 2017. [En línea]. Disponible en: https://developer.android.com/about/dashboards/index.html. [Accedido: 10- Abr- 2017].

[3] Mintic, "Guía para la Gestión y Clasificación de Incidentes de Seguridad de la Información", *Mintic.gov.co*, 2016. [En línea]. Disponible en: https://bit.ly/2yaBgLu. [Accedido: 10- Abr- 2017].

[4] M. Guido, J. Buttner y J. Grover, "Rapid differential forensic imaging of mobile devices", en Proceedings of the 16th annual USA digital forensics research conference, 2016, vol. 18, supplement, pp S46-S54.

[5] N. Aziz, F. Mokhti y M. Nadhar, "Mobile Device Forensics: Extracting and Analysing Data from an Android-Based Smartphone" en 2015 Fourth International Conference on Cyber Security, Cyber Warfare, and Digital Forensic (CyberSec), 2015, DOI: 10.1109/CyberSec.2015.32.

[6] Intel, "Diferentes protocolos de Wi-Fi y velocidades de datos", *Intel*, 2017. [En línea]. Disponible en: https://intel.ly/2JmNhnL. [Accedido: 18- Abr- 2017].

[7] Fiscalía General de la Nación, "Manual de Procedimientos para Cadena de Custodia", *Fiscalia.gov.co*, 2012. [En línea]. Disponible en: https://bit.ly/2L0Rtwz. [Accedido: 16- Abr- 2017].

[8] Agrawal, V., y Tapaswi, S. Forensic analysis of Google Allo messenger on Android platform. Information and Computer Security, 27(1), 62–80. 2019, DOI:10.1108/ICS-03-2017-0011.

[8] Android, "Android Open Source Project", *Android Open Source Project*, 2018. [En línea]. Disponible en: https://source.android.com/. [Accedido: 16- Mar- 2018].

[9] J. Gironés, *El gran libro de Android*, 6th ed. México: Alfaomega Grupo Editor, S.A. de C.V, 2018.

[10] Bendinelli, M. (2013). *Desarmando al Androide*. Maestría. Universidad de Buenos Aires.

[11] Android, "Android Architecture | Android Open Source Project", *Android Open Source Project*, 2018. [En línea]. Disponible en: https://source.android.com/devices/architecture/. [Accedido: 13- Abr- 2018].

[12] Android, "Platform Architecture | Android Developers", *Android Developers*, 2018. [En línea]. Disponible en: https://developer.android.com/guide/platform/index.html. [Accedido: 14- Abr- 2018].

[13] Android, "Centro de seguridad de Android", *Android*, 2018. [En línea]. Disponible en: https://www.android.com/intl/es_es/security-center/. [Accedido: 07- May- 2018].

[14] J. Joshi y C. Parekh, "Android smartphone vulnerabilities: A survey" en 2016 International Conference on Advances in Computing, Communication, & Automation (ICACCA) (Spring), 2016, DOI: 10.1109/ICACCA.2016.7578857.

[15] MITRE Corp., "Google Android: CVE security vulnerabilities, versions and detailed reports", *Cvedetails.com*, 2019. [En línea]. Disponible en: https://bit.ly/2ArK1lG. [Accedido: 29- May- 2019].

[16] V. Venkateswara y A. Chakravarthy, "Forensic analysis of android mobile devices" en Conference: 2016 International Conference on Recent Advances and Innovations in Engineering (ICRAIE), 2016, DOI: 10.1109/ICRAIE.2016.7939540.

[17] Z. Li, B. Xi and S. Wu, "Digital forensics and analysis for Android devices", in *2016 11th International Conference on Computer Science & Education (ICCSE)*, Nagoya, 2016 [En línea]. Disponible en: https://ieeexplore.ieee.org/document/7581630. [Accedido: 29- Ago- 2019].

[18] D. Čečavac, M. Dinić, A. Marković and P. Jovanović, "Adapting UBIFS file system to android operating system requirements", in *2017 25th Telecommunication Forum (TELFOR)*, Belgrado, 2017 [En línea]. Disponible en: https://ieeexplore.ieee.org/document/8249445. [Accedido: 29- Ago- 2019].

[19] S. Wu, X. Xiong, Y. Zhang, Y. Tang y B. Jin. “A general forensics acquisition for Android smartphones with qualcomm processor” en Conference: 2017 IEEE 17th International Conference on Communication Technology (ICCT), 2017, DOI: 10.1109/ICCT.2017.8359976.

[20] Android, "Data and file storage overview | Android Developers", *Android Developers*, 2019. [En línea]. Disponible en: https://developer.android.com/guide/topics/data/data-storage.html. [Accedido: 14- Mar- 2019].

[21] H. Kummert, "The PPP Triple-DES Encryption Protocol (3DESE)", *Network Working Group*, 1998. [En línea]. Disponible en: https://www.hjp.at/doc/rfc/rfc2420.txt. [Accedido: 05- May- 2018].

[22] NIST, "AES Development - Cryptographic Standards and Guidelines | CSRC", *Csrc.nist.gov*, 2016. [En línea]. Disponible en: https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines/archived-crypto-projects/aes-development. [Accedido: 13- Mar- 2018].

[23] Y. Medina y H. Miranda, *Comparación de Algoritmos Basados en la Criptografía Simétrica DES, AES y 3DES*, 9th ed. Cúcuta: Mundo FESC, 2015, pp. 14-21 [En línea]. Disponible en: https://dialnet.unirioja.es/descarga/articulo/5286657.pdf. [Accedido: 26- Abr- 2018].

[24] O. Almasri y H. Jani. “Introducing an Encryption Algorithm based on IDEA” en International Journal of Science and Research (IJSR), 2013, vol 2, issue 9.

[25] B. Schneier, "Schneier on Security: The Blowfish Encryption Algorithm", *Schneier.com*, 2010. [En línea]. Disponible en: https://www.schneier.com/academic/blowfish/. [Accedido: 10- May- 2018].

[26] B. Schneier, "Academic: The Twofish Encryption Algorithm - Schneier on Security", *Schneier.com*, 2019. [En línea]. Disponible en: https://bit.ly/2XUnInw. [Accedido: 16- Mar- 2019].

[27] B. Schneier, "Schneier on Security: Skein: Threefish", *Schneier.com*, 2009. [En línea]. Disponible en: https://www.schneier.com/academic/skein/threefish.html. [Accedido: 07- May- 2018].

[28] H. Heys y S. Tavares, "On the security of the CAST encryption algorithm", en 1994 Proceedings of Canadian Conference on Electrical and Computer Engineering, 1994, DOI: 10.1109/CCECE.1994.405756.

[29] M. Kwan, "The design of ICE encryption algorithm", en FSE '97 Proceedings of the 4th International Workshop on Fast Software Encryption, 2006, pp 69-82.

[30] M. Monger, "RC6: The Simple Cipher", *Users.cs.jmu.edu*, 2004. [En línea]. Disponible en: https://users.cs.jmu.edu/abzugcx/Public/Student-Produced-Term-Projects/Cryptology-2002-SPRING/RC6-by-Morgan-Monger-2004-Fall.doc. [Accedido: 27- Abr- 2018].

[31] D. Bernstein, "The Salsa20 Family of Stream Ciphers", in *New Stream Cipher Designs*, 1st ed., M. Robshaw and O. Billet, Ed. Berlin: Springer-Verlag Berlin, 2008, pp. 84-97 [En línea]. Disponible en: https://link.springer.com/chapter/10.1007%2F978-3-540-68351-3_8. [Accedido: 02- Abr- 2018].

[32] J. Sugier, " Implementing Serpent Cipher in Field Programmable Gate Arrays", en ICIT 2011 The 5th International Conference on Information Technology, 2011, pp 69-82 [En línea]. Disponible en:

http://icit.zuj.edu.jo/icit11/PaperList/Papers/Information%20Security/620_Jaroslaw.pdf. [Accedido: 16-May-2018].

[33] M. McLoone, "Hardware performance analysis of the SHACAL-2 encryption algorithm", *IEE Proceedings - Circuits Devices and Systems*, no. 152, pp. 478-484, 2005 [En línea]. Disponible en: https://ieeexplore.ieee.org/document/1522046. [Accedido: 16- May- 2018].

[34] T. Isobe y K. Shibutani, "Fast Software Encryption", *Google Books*, 2014. [En línea]. Disponible en: https://bit.ly/2S75Yj9, pp 122-124. [Accedido: 16- May- 2018].

[35]L. Gómez, "PROTOCOLO DE ACTUACIÓN EN PERITACIONES INFORMÁTICAS", Ingeniería, Universidad Autónoma de Madrid, 2016.

[36] M. Porolli, "¿Qué herramienta de adquisición de imágenes forenses elegir? | WeLiveSecurity", *WeLiveSecurity*, 2013. [En línea]. Disponible en: https://www.welivesecurity.com/la-es/2013/07/17/herramienta-adquisicion-imagenes-forenses-elegir/. [Accedido: 30- Ago- 2019].

[37] AccessData, "Forensic Toolkit (FTK)®", *AccessData*, 2019. [En línea]. Disponible en: https://accessdata.com/products-services/forensic-toolkit-ftk. [Accedido: 26- Ago- 2019].

[38] Magnet Forensics, "Forensic Examiners - Speed up Investigations | Magnet Forensics", *Magnet Forensics*, 2019. [En línea]. Disponible en: https://www.magnetforensics.com/for-forensic-examiners/. [Accedido: 26- Ago- 2019].

[39] Guidance Software, "EnCase Forensic Software - Top Digital Forensics & Investigations Solution", *Guidancesoftware.com*, 2019. [En línea]. Disponible en: https://www.guidancesoftware.com/encase-forensic?cmpid=nav_r. [Accedido: 27- Ago- 2019].

[40] BlackBag, "Mobilyze iOS and Android Forensics Software", *Blackbagtech.com*, 2019. [En línea]. Disponible en: https://www.blackbagtech.com/mobilyze.html. [Accedido: 26- Ago- 2019].

[41]NEO, "Forensic Falcon®-NEO - Logicube", *Logicube.com*, 2019. [En línea]. Disponible en: https://www.logicube.com/shop/forensic-falcon-neo/?v=42983b05e2f2. [Accedido: 29- Aug- 2019].

[42] IONOS, "Imaging software: una comparativa", *Digital Guide*, 2019. [En línea]. Disponible en: https://www.ionos.es/digitalguide/servidores/herramientas/imaging-software/. [Accedido: 16- Abr- 2019].

[43] J. Pillou, "Crear una imagen del sistema", *CCM*, 2010. [En línea]. Disponible en: https://es.ccm.net/faq/180-crear-una-imagen-del-sistema. [Accedido: 26- Feb- 2019].

[44] D. Naranjo, "Zbakcup una herramienta para crear respaldos desde la terminal", *Linux Adictos*. [En línea]. Disponible en: https://www.linuxadictos.com/zbakcup-una-herramienta-para-crear-respaldos-desde-la-terminal.html. [Accedido: 30- Jun- 2019].

[45] WikiArchLinux, "rsync (español) - ArchWiki", *Wiki.archlinux.org*, 2019. [En línea]. Disponible en: https://wiki.archlinux.org/index.php/Rsync_(Espa%C3%B1ol). [Accedido: 30- Jun- 2019].

[46] Linuxize, "How to Use Rsync for Local and Remote Data Transfer and Synchronization", *Linuxize.com*, 2019. [En línea]. Disponible en: https://linuxize.com/post/how-to-use-rsync-for-local-and-remote-data-transfer-and-synchronization/. [Accedido: 31- Jun- 2019].

[47] Isaac, "Todos los secretos de la compresión en GNU/Linux", *Linux Adictos*, 2017. [En línea]. Disponible en: https://www.linuxadictos.com/todos-los-secretos-la-compresion-gnu-linux.html. [Accedido: 15- May- 2019].

[48]Linuxito, "Comprimir y extraer archivos con ZPAQ", *Linuxito.com*, 2017. [Online]. Available: https://www.linuxito.com/nix/869-comprimir-y-extraer-archivos-con-zpaq. [Accessed: 25- Jun- 2019].

[49] Infosegur, "Integridad | Seguridad Informática", *Infosegur.wordpress.com*, 2013. [En línea]. Disponible en: https://infosegur.wordpress.com/tag/integridad/. [Accedido: 17- Nov- 2018].

[50] PowerData Solutions, "Qué se entiende por integridad de los datos", *Blog.powerdata.es*, 2013. [En línea]. Disponible en: https://bit.ly/2Ije2bG. [Accedido: 20- Dic- 2018].

[51] Endpoint, "Claves para garantizar la integridad de los datos", *Itdigitalsecurity.es*, 2017. [En línea]. Disponible en: https://www.itdigitalsecurity.es/endpoint/2017/12/claves-para-garantizar-la-integridad-de-los-datos. [Accedido: 20- Dic- 2018].

[52] E. Gelbstein, "La integridad de los datos: el aspecto más relegado de la seguridad de la información", *ISACA Journal*, vol. 6, no. 2011, pp. 1-11, 2011 [En línea]. Disponible en: https://www.isaca.org/Journal/archives/2011/Volume-6/Pages/Data-Integrity-Information-Securitys-Poor-Relation-spanish.aspx. [Accedido: 19- May- 2019].

[53] B. Donohue, "¿Qué Es Un Hash Y Cómo Funciona?", *Latam.kaspersky.com*, 2014. [En línea]. Disponible en: https://latam.kaspersky.com/blog/que-es-un-hash-y-como-funciona/2806/. [Accedido: 12- Feb- 2019].

[54] M. Ramírez, C. Pino, V. Trujillo Olaya y J. Velasco Medina, "Implementación hardware del algoritmo Keccak para Hash-3 y comparación con Blake, Grøstl, JH y Skein", *Informador Técnico*, vol. 77, no. 2, pp. 167-171, 2013 [En línea]. Disponible en: https://www.researchgate.net/publication/317121299_Implementacion_hardware_del_algoritmo_Keccak_para_Hash-3_y_comparacion_con_Blake_Grostl_JH_y_Skein. [Accedido: 18- Jun- 2019].

[55] R. Grimes, "Why aren't we using SHA3?", *CSO Online*, 2018. [En línea]. Disponible en: https://www.csoonline.com/article/3256088/why-arent-we-using-sha3.html. [Accedido: 21- May- 2019].

[56] Jota, "Checksum MD5 y SHA: funciones hash e integridad de datos en Linux y UNIX", *El array de Jota*, 2014. [En línea]. Disponible en: https://www.elarraydejota.com/checksum-md5-y-sha-funciones-hash-e-integridad-de-datos-en-linux-y-unix/. [Accedido: 15- Ene- 2019].

[57] Senado de la República de Colombia, "Leyes desde 1992 - Vigencia expresa y control de constitucionalidad [LEY_1581_2012]", *Secretariasenado.gov.co*, 2012. [En línea]. Disponible en: http://www.secretariasenado.gov.co/senado/basedoc/ley_1581_2012.html. [Accedido: 10- Mar- 2018].

[58] Ministerio de Industria, Comercio y Turismo, "DECRETO NÚMERO 1317 DE 2013", *Mintic.gov.co*, 2013. [En línea]. Disponible en: https://bit.ly/2p5jCal. [Accedido: 08- May- 2018].

[59] M. Rouse, "¿Qué es Caja de arena o sandbox? - Definición en WhatIs.com", *SearchDataCenter en español*, 2019. En línea]. Disponible en: https://bit.ly/2GdiBGo. [Accedido: 20- Ene- 2019].

[60] J. Carles, "¿Qué es y para qué sirve un sandbox en la informática?", *Geekland. Blog de Tecnología*, 2017. [En línea]. Disponible en: https://geekland.eu/que-es-y-para-que-sirve-un-sandbox/. [Accedido: 12- Feb- 2019].

[61] M. Rouse, "¿Qué es Sandbox de aplicaciones? - Definición en WhatIs.com", *SearchDataCenter en español*, 2019. [En línea]. Disponible en: https://bit.ly/2Gcdiad. [Accedido: 06- Mar- 2019].

[62] R. Andrés, "Qué es una máquina virtual, cómo funciona y para qué sirve", *ComputerHoy*, 2017. [En línea]. Disponible en: https://computerhoy.com/noticias/software/que-es-maquina-virtual-como-funciona-que-sirve-46606. [Accedido: 18- Mar- 2019].

[63] J. Ranchal, "10 trucos para mejorar el uso de máquinas virtuales", *MuyComputer*, 2019. [En línea]. Disponible en: https://www.muycomputer.com/2019/01/01/uso-de-maquinas-virtuales/. [Accedido: 10- Feb- 2019].

[64] Cavsi, "Wifi - ¿Qué es WiFi? - Definición - Ventajas y Desventajas", *¿Qué es WiFi?* [En línea]. Disponible en: http://www.cavsi.com/preguntasrespuestas/que-es-wifi/. [Accedido: 14- May- 2019].

[65] K. Shaw, "802.11: estándares de Wi-Fi y velocidades", *Networkworld.es*, 2018. [En línea]. Diisponible en: https://www.networkworld.es/wifi/80211-estandares-de-wifi-y-velocidades. [Accedido: 15- Abr- 2019].

[66] S. Arteaga, "Todo lo que necesitas saber del WiFi AD, WiFi AH y HaLow", *ComputerHoy*, 2016. [En línea]. Disponible en: https://computerhoy.com/noticias/internet/todo-que-necesitas-saber-del-wifi-ad-wifi-ah-halow-39101. [Accedido: 15- Abr- 2019].

[67] O. Martínez, "Kali Linux, una gran suite de seguridad informática", *Desde Linux*, 2019. [En línea]. Disponible en: https://blog.desdelinux.net/kali-linux-suite-seguridad-informatica/. [Accedido: 07- Mar- 2019].

[68] MLX, "¿Que es Kali GNU/Linux?", *Maslinux.es*, 2018. [En línea]. Disponible en: https://maslinux.es/que-es-kali-gnu-linux/. [Accedido: 18- May- 2019].

[69] Offensive Security, "Download Kali Linux Images", *Kali.org*, 2019. [En línea]. Disponible en: https://www.kali.org/downloads/. [Accedido: 06- Abr- 2019].

[70] MLX, "Kali Linux: Lo que debes saber antes de usarlo", *Maslinux.es*, 2018. [En línea]. Disponible: https://maslinux.es/kali-linux-lo-que-debes-saber-antes-de-usarlo/. [Accedido: 10- Ene- 2019].

[71] Android, "Android Debug Bridge | Android Developers", *Android Developers*, 2019. [En línea]. Disponible en: https://developer.android.com/studio/command-line/adb?hl=es-419. [Accedido: 09- Abr- 2019].

[72] P. Galeas, "CURSO-PRODIST", *GitHub*, 2017. [En línea]. Disponible en: https://github.com/pgaleas/CURSO-PRODIST/wiki/2.-%C2%BFQu%C3%A9-es-un-Sistema-Distribuido%3F. [Accedido: 17- Jul- 2019].

[73] Y. Tanaka, N. Hayashibara, T. Enokido y M. Takizawa. "Transactional agento n distributed objects", 11th International Conference on Parallel and Distributed Systems (ICPADS'05), 2005, DOI: 10.1109/ICPADS.2005.284.

[74] J. Royo, "Agentes móviles inteligentes aplicados al diseño y desarrollo de servicios de datos en entornos inalámbricos y distribuidos", Doctorado, Universidad de Zaragoza, 2009.

[75] C. Llorca, "Estas son las versiones más utilizadas de Android en mayo de 2019", *Techdroy*, 2019. [En línea]. Disponible en: https://www.techdroy.com/versiones-utilizadas-android-mayo-2019/. [Accedido: 17- May- 2019].

[76] D. Bilić, "Seguridad en dispositivos móviles: resumen de lo que fue el 2018 | WeLiveSecurity", *WeLiveSecurity*, 2018. [En línea]. Disponible en: https://bit.ly/2XOiVzE. [Accedido: 25- Ene- 2019].

[77] Open Geek, "¿Qué son y para qué sirven los metadatos?", *Opengeekservice.cl*. [En línea]. Disponible en: https://bit.ly/2JCEHRY. [Accedido: 29- Feb- 2019].

[78] Edu4rdSHL, "Criptografía: simétrica, asimétrica e hibrida. | Security Hack Labs", *Security Hack Labs*, 2018. [En línea]. Disponible en: https://www.securityhacklabs.net/articulo/criptografia-simetrica-asimetrica-e-hibrida. [Accedido: 19- Jul- 2019].

[79] H. Chang, "International Data Encryption Algorithm", James Madison University, 2004 [En línea]. Disponible en: /abzugcx/Public/Student-Produced-Term-Projects/Cryptology-2002-SPRING/IDEA-by-How-Shen-Chang-2004-FALL.doc. [Accedido: 14- Nov- 2017].

[80] Dai, W. (2014). *Crypto++ Library 8.2 | Free C++ Class Library of Cryptographic Schemes*. [En línea] Cryptopp.com. Disponible en: https://www.cryptopp.com/ [Accedido: 14-Oct-2018].

[81] VMWareDOcs, "Tipos de adaptador de red", *Docs.vmware.com*, 2015. [En línea]. Disponible en: https://bit.ly/2GeMioR. [Accedido: 09- Jun- 2019].

[82] Magnet, "Advanced Mobile Acquisition for Android - Magnet Forensics", *Magnet Forensics*, 2019. [En línea]. Disponible en: https://www.magnetforensics.com/resources/advancedmobile/. [Accedido: 29- Jun- 2019].

[83] C. Dobraunig, M. Eichlseder y F. Mendel, "Analysis of SHA-512/224 and SHA-512/256", *Eprint.iacr.org*, 2015. [En línea]. Disponible en: https://eprint.iacr.org/2016/374.pdf. [Accedido: 29- Abr- 2019]

[84] Android, "Android 10 | Android Developers", *Android Developers*, 2019. [En línea]. Disponible en: https://developer.android.com/about/versions/10. [Accedido: 04- Ago- 2019].

[85] Android, "Centro de seguridad de Android", *Android*, 2019. [En línea]. Disponible en: https://www.android.com/intl/es_es/security-center/. [Accedido: 04- Ago- 2019].

[86] Android, "Android 4.4 APIs | Android Developers", *Android Developers*, 2019. [En línea]. Disponible en: https://developer.android.com/about/versions/android-4.4. [Accedido: 04- Ago- 2019].

[87] D. Bernstein, "The Salsa20 family of stream ciphers", *Cr.yp.to*, 2007. [En línea]. Disponible: https://cr.yp.to/snuffle/salsafamily-20071225.pdf. [Accedido: 20- Ago- 2018].