CASO PRÁCTICO: FUNDAMENTOS DE SEGURIDAD CLOUD E INFRAESTRUCTURAS INDUSTRIALES

Alumno: Gengis Rovi

---

1. INTRODUCCIÓN

El presente informe detalla el procedimiento técnico y metodológico llevado a cabo para resolver dos escenarios prácticos de seguridad y despliegue de infraestructuras.

La primera sección aborda un análisis forense digital sobre un dispositivo móvil Android (evidencia `data.dd`), con el objetivo de esclarecer las actividades sospechosas del sujeto "Mr. X". Se ha utilizado una metodología de extracción física simulada y análisis de sistemas de ficheros y bases de datos SQLite en un entorno macOS.

La segunda sección detalla la implementación de arquitecturas basadas en contenedores Docker, aplicando buenas prácticas de seguridad como el uso de usuarios no privilegiados, persistencia de datos mediante volúmenes y orquestación de servicios mediante Docker Compose.

---

2. ANÁLISIS FORENSE DE DISPOSITIVOS MÓVILES (CASO MR. X)

2.1. Metodología y Herramientas

Para el análisis del archivo de evidencia `data.dd`, se ha procedido a trabajar sobre una estación de trabajo macOS. Dado que el sistema de ficheros de Android (típicamente EXT4) no es nativo en este entorno, se han utilizado herramientas de análisis forense para la localización de inodos y extracción de ficheros sin alterar la integridad de la evidencia.

- **Evidencia:** Archivo de imagen `data.dd`.
- **Herramientas:**
    - **The Sleuth Kit (TSK):** Herramientas `mmls` (para particionado), `fls` (para listar archivos) y `icat` (para extracción de contenido).
    - **DB Browser for SQLite:** Para el análisis de las bases de datos extraídas (`.db`).
    - **Terminal macOS:** Para la ejecución de comandos y filtrado con `grep`.

2.2. Preparación y Análisis de Particiones

En primer lugar, se analizó la estructura de particiones de la imagen para localizar la partición de datos de usuario (`/data`), donde reside la información crítica de Android como SMS y contactos.

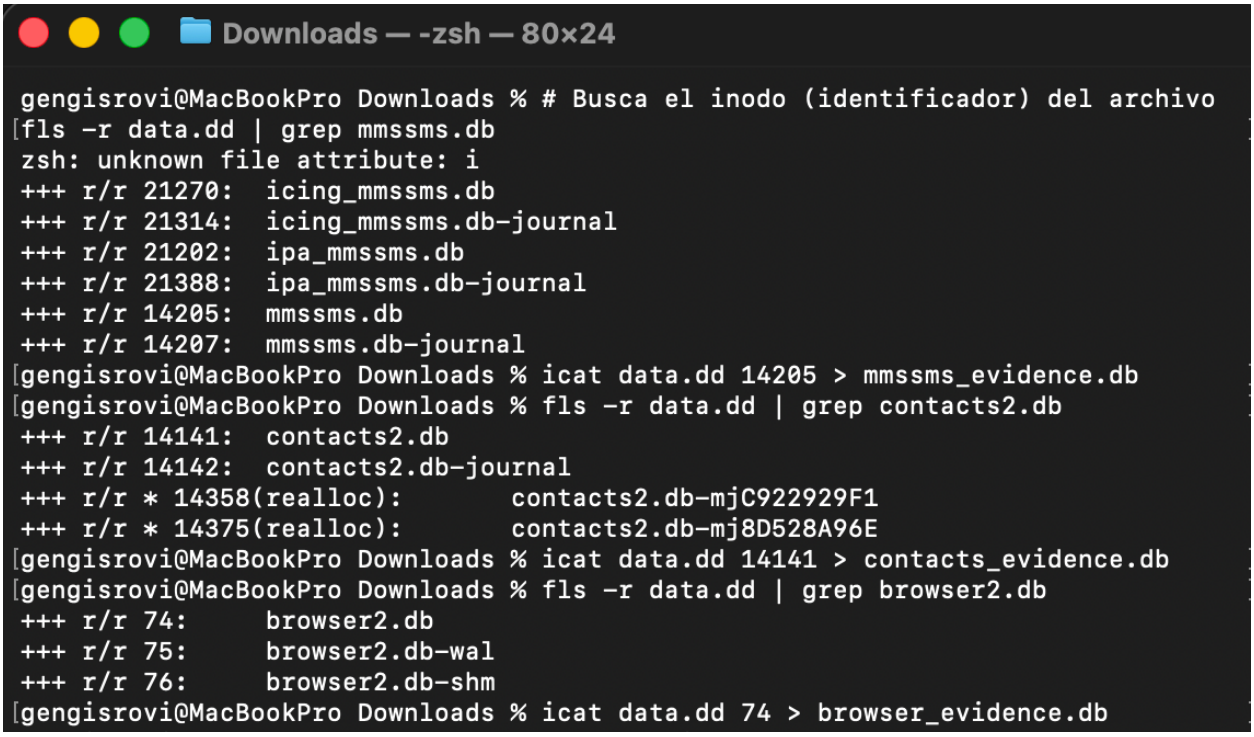


Figura 1. Análisis de la tabla de particiones de la imagen forense.

2.3. Resolución de las Incógnitas del Caso

2.3.1. Trama, Cómplices y Dinero (Análisis de SMS)

Para determinar las intenciones de Mr. X, se procedió a localizar y extraer la base de datos de mensajería `mmssms.db`, ubicada teóricamente en `/data/data/com.android.providers.telephony/databases/`.

Se utilizó el comando `fls` para localizar el inodo del archivo y `icat` para su extracción:

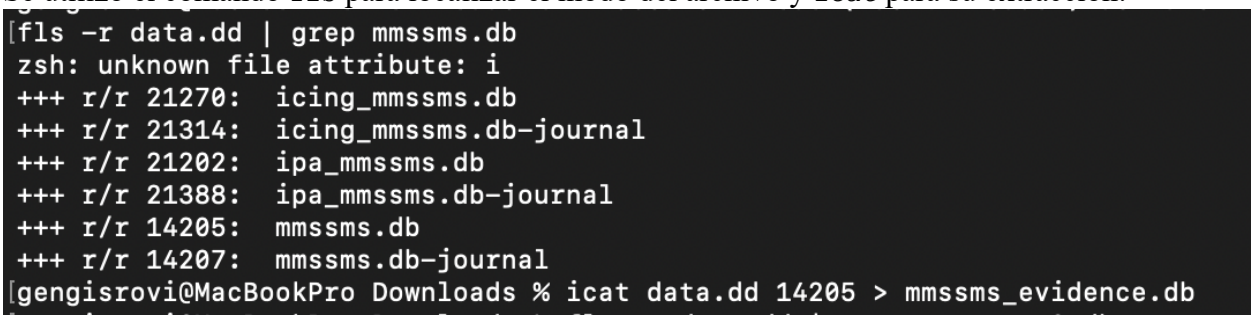
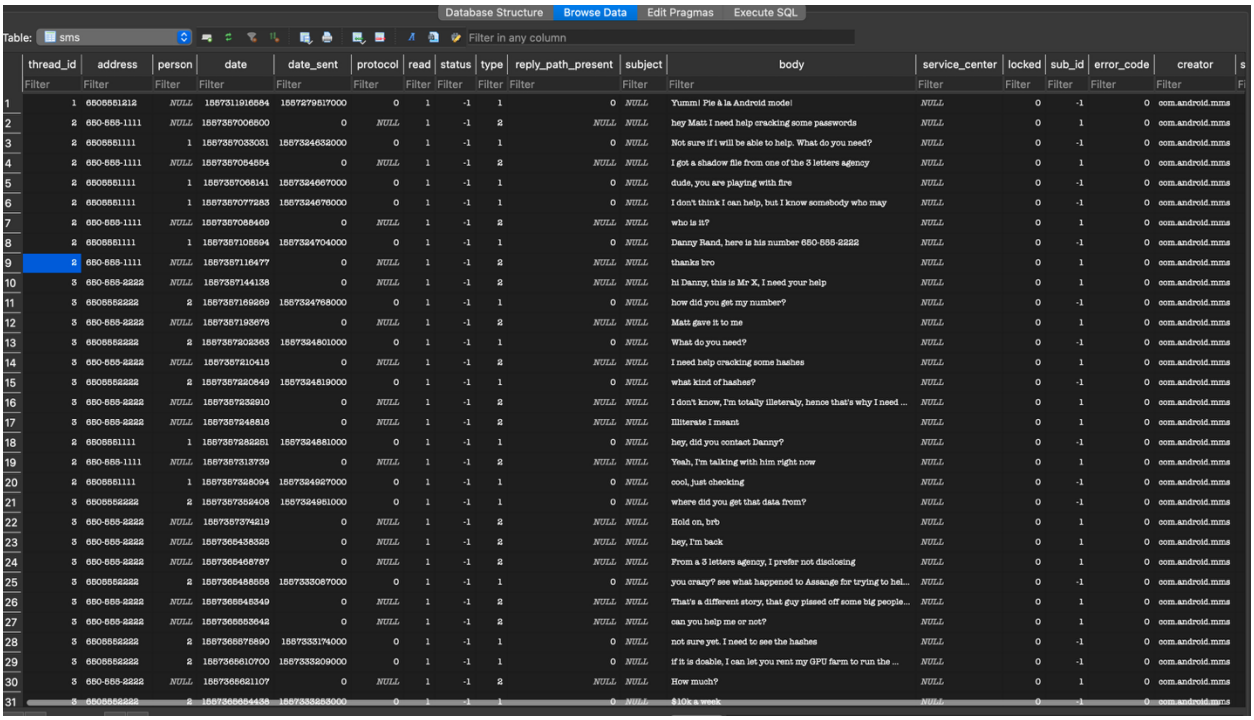


Figura 2. Localización y extracción forense de la base de datos de mensajería.

Tras abrir el archivo `mmssms_evidence.db` en DB Browser y analizar la tabla `sms`, se obtuvieron los siguientes hallazgos:



	thread_id	address	person	date	date_sent	protocol	read	status	type	reply_path_present	subject	body	service_center	locked	sub_id	error_code	creator
1	1	6505551212	NULL	1557311916584	1557279517000	0	1	-1	1	0	NULL	Yumm! Pie à la Android mode!	NULL	0	-1	0	com.android.mms
2	2	650-555-1111	NULL	1557357006800	0	NULL	1	-1	2	NULL	NULL	hey Matt I need help cracking some passwords	NULL	0	1	0	com.android.mms
3	2	6505551111	1	1557357033031	1557324632000	0	1	-1	1	0	NULL	Not sure if I will be able to help. What do you need?	NULL	0	-1	0	com.android.mms
4	2	650-555-1111	NULL	1557357054554	0	NULL	1	-1	2	NULL	NULL	I got a shadow file from one of the 3 letters agency	NULL	0	1	0	com.android.mms
5	2	6505551111	1	1557357056141	1557324667000	0	1	-1	1	0	NULL	dude, you are playing with fire	NULL	0	-1	0	com.android.mms
6	2	6505551111	1	1557357077263	1557324676000	0	1	-1	1	0	NULL	I don't think I can help, but I know somebody who may	NULL	0	-1	0	com.android.mms
7	2	650-555-1111	NULL	1557357084469	0	NULL	1	-1	2	NULL	NULL	who is it?	NULL	0	1	0	com.android.mms
8	2	6505551111	1	1557357105594	1557324704000	0	1	-1	1	0	NULL	Danny Rand, here is his number 650-555-2222	NULL	0	-1	0	com.android.mms
9	2	650-555-1111	NULL	1557357116477	0	NULL	1	-1	2	NULL	NULL	thanks bro	NULL	0	1	0	com.android.mms
10	3	650-555-2222	NULL	1557357144138	0	NULL	1	-1	2	NULL	NULL	hi Danny, this is Mr X, I need your help	NULL	0	1	0	com.android.mms
11	3	6505552222	2	1557357169269	1557324768000	0	1	-1	1	0	NULL	how did you get my number?	NULL	0	-1	0	com.android.mms
12	3	650-555-2222	NULL	1557357193676	0	NULL	1	-1	2	NULL	NULL	Matt gave it to me	NULL	0	1	0	com.android.mms
13	3	6505552222	2	1557357202363	1557324801000	0	1	-1	1	0	NULL	What do you need?	NULL	0	-1	0	com.android.mms
14	3	650-555-2222	NULL	1557357210415	0	NULL	1	-1	2	NULL	NULL	I need help cracking some hashes	NULL	0	1	0	com.android.mms
15	3	6505552222	2	1557357220849	1557324819000	0	1	-1	1	0	NULL	what kind of hashes?	NULL	0	-1	0	com.android.mms
16	3	650-555-2222	NULL	1557357232910	0	NULL	1	-1	2	NULL	NULL	I don't know, I'm totally illeteraly, hence that's why I need ...	NULL	0	1	0	com.android.mms
17	3	650-555-2222	NULL	1557357248516	0	NULL	1	-1	2	NULL	NULL	Illiterate I meant	NULL	0	1	0	com.android.mms
18	2	6505551111	1	1557357282251	1557324881000	0	1	-1	1	0	NULL	hey, did you contact Danny?	NULL	0	-1	0	com.android.mms
19	2	650-555-1111	NULL	1557357313739	0	NULL	1	-1	2	NULL	NULL	Yeah, I'm talking with him right now	NULL	0	1	0	com.android.mms
20	2	6505551111	1	1557357328094	1557324927000	0	1	-1	1	0	NULL	cool, just checking	NULL	0	-1	0	com.android.mms
21	3	6505552222	2	1557357352408	1557324981000	0	1	-1	1	0	NULL	where did you get that data from?	NULL	0	-1	0	com.android.mms
22	3	650-555-2222	NULL	1557357374219	0	NULL	1	-1	2	NULL	NULL	Hold on, brb	NULL	0	1	0	com.android.mms
23	3	650-555-2222	NULL	1557356436325	0	NULL	1	-1	2	NULL	NULL	hey, I'm back	NULL	0	1	0	com.android.mms
24	3	650-555-2222	NULL	1557356466787	0	NULL	1	-1	2	NULL	NULL	From a 3 letters agency, I prefer not disclosing	NULL	0	1	0	com.android.mms
25	3	6505552222	2	1557356486558	1557333087000	0	1	-1	1	0	NULL	you crazy? see what happened to Assange for trying to hel...	NULL	0	-1	0	com.android.mms
26	3	650-555-2222	NULL	1557356545349	0	NULL	1	-1	2	NULL	NULL	That's a different story, that guy pissed off some big people...	NULL	0	1	0	com.android.mms
27	3	650-555-2222	NULL	1557356553642	0	NULL	1	-1	2	NULL	NULL	can you help me or not?	NULL	0	1	0	com.android.mms
28	3	6505552222	2	1557356575890	1557333174000	0	1	-1	1	0	NULL	not sure yet. I need to see the hashes	NULL	0	-1	0	com.android.mms
29	3	6505552222	2	1557356610700	1557333209000	0	1	-1	1	0	NULL	if it is doable, I can let you rent my GPU farm to run the ...	NULL	0	-1	0	com.android.mms
30	3	650-555-2222	NULL	1557356621107	0	NULL	1	-1	2	NULL	NULL	How much?	NULL	0	1	0	com.android.mms
31	3	6505552222	2	1557356654436	1557333263000	0	1	-1	1	0	NULL	$10k a week	NULL	0	-1	0	com.android.mms

Figura 3. Contenido de los mensajes SMS incriminatorios.

- **¿Qué tramaba Mr. X?:** Descifrar algunas contrasenas de una compania de 3 letras
- **¿A quién le pidió ayuda inicialmente?:** Le pide ayuda a Matt, su numero seria 650-555-1111, para descifrar algunas contrasenas.
- **¿Quién le intentó ayudar y por cuánto dinero?:** Danny Rand y su numero seria 650-555-2222, intento ayudar por 10k a la semana

2.3.2. Enlace del Fichero y Navegación

Para localizar el enlace compartido y analizar la navegación, se extrajo la base de datos del historial del navegador, típicamente `browser2.db` o `History` (Chrome), ubicada en `/data/data/com.android.browser/databases/`.

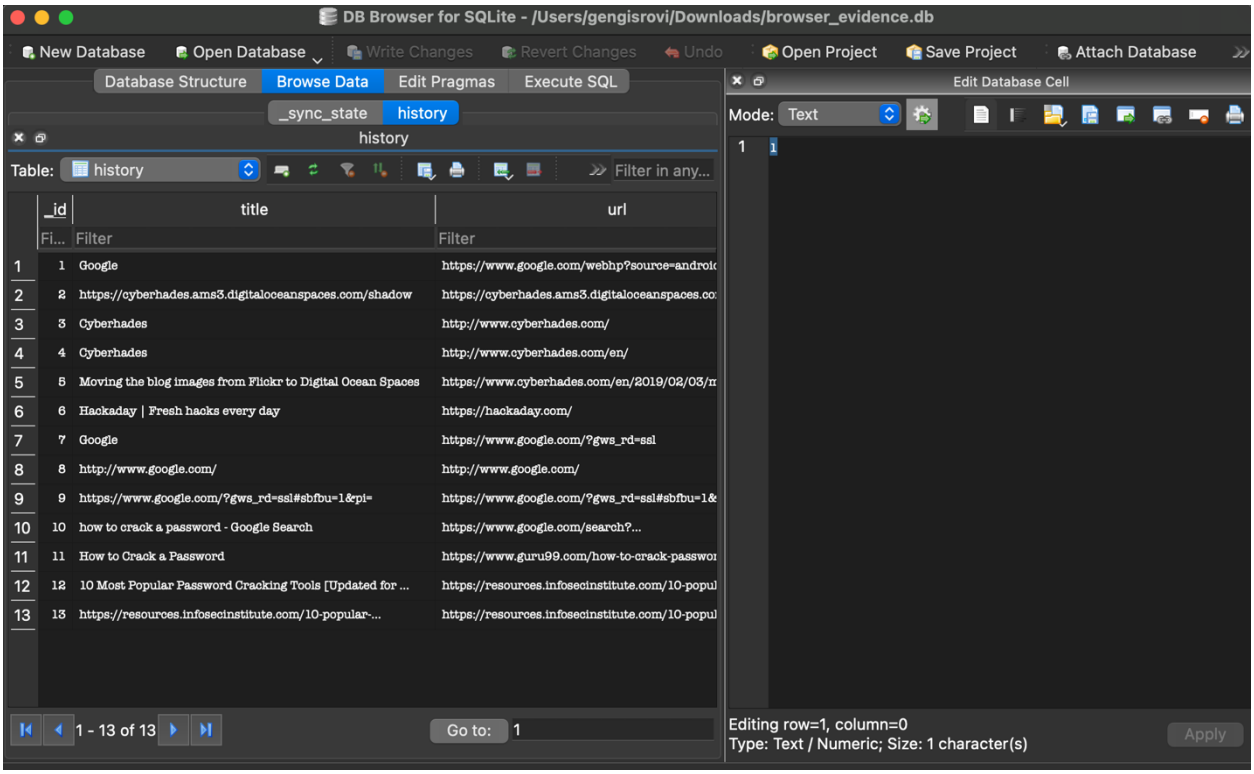
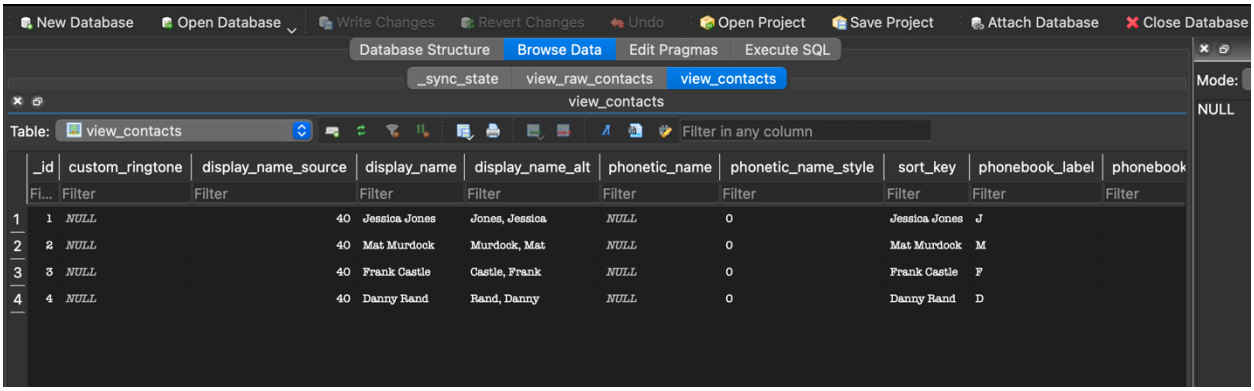


Figura 4. Historial de navegación y localización del enlace.

- **¿Quién le pasó el enlace?:** Jessica con numero 6505554444, le compartio el enlace.
- **¿Cuál es el enlace?:** https://cyberhades.ams3.digitaloceanspaces.com/shadow
- **Historial y Cookies:** Se ha verificado la tabla de historial mostrando la actividad reciente del usuario, efectivamente se puede ver como Mr. x cuenta con el enlace en cuestion en el historial.

2.3.3. Identificación de Contactos y Llamadas

Para corroborar la identidad de los interlocutores, se analizó la base de datos `contacts2.db`, ubicada en `/data/data/com.android.providers.contacts/databases/`.



DB Browser mostrando la tabla `calls` *Figura 5. Registro de llamadas y lista de contactos.*

Se cruzaron los números de teléfono de los SMS con esta base de datos para identificar los nombres reales de los implicados.

2.3.4. Información del Sistema y Software (Análisis de `packages.xml`)

Ante la ausencia del archivo `build.prop` en la partición de datos recuperada, se procedió al análisis forense del archivo de registro de paquetes `packages.xml` (Inodo recuperado mediante TSK), ubicado en `/data/system/packages.xml`. Este archivo contiene el inventario histórico de la instalación del sistema.

A. Versión del Sistema Operativo El análisis de la etiqueta XML `<last-platform-version>` reveló los siguientes metadatos:

- **SDK Version:** 22
- **Release:** Android 5.1.1
- **Fingerprint:** `Android/sdk_google_phone_x86_64/generic_x86_64:5.1.1...`

```
[gengisrovi@MacBookPro Downloads % fls -r data.dd | grep packages.xml
 + r/r 14307:    packages.xml
[gengisrovi@MacBookPro Downloads % icat data.dd 14307 > packages.xml
```

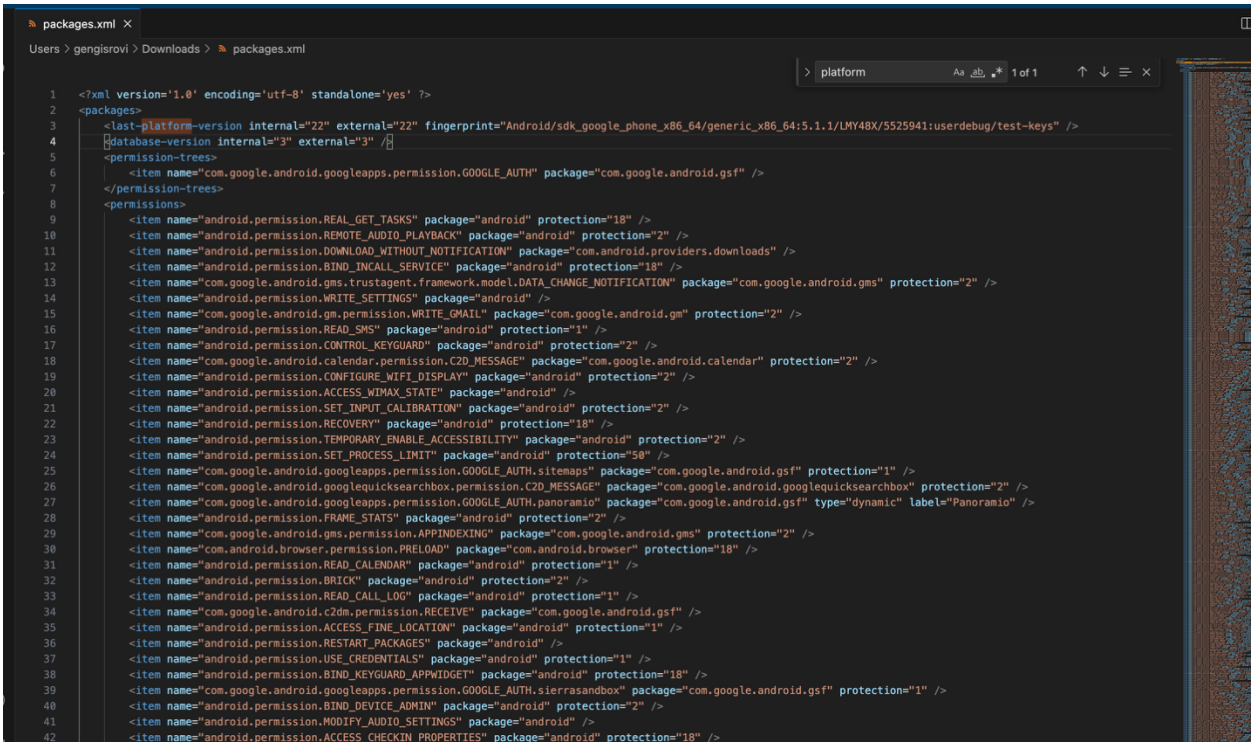


Figura 6. Evidencia de la versión del SO y tipo de dispositivo en packages.xml.

Hallazgo Forense: El dispositivo analizado no es un terminal físico convencional, sino un **Emulador Android x86_64**. El uso de un emulador es una técnica común en ciberdelincuencia para evitar dejar rastros físicos de hardware (IMEI real, ubicación GPS física) y para realizar

pruebas de penetración (hacking) en un entorno controlado. Por otro lado la cuenta registrada fue ceupe.forensics@gmail.com

B. Aplicaciones Instaladas (User Apps) Se filtraron las entradas del XML buscando la ruta `/data/app/`, la cual corresponde a aplicaciones instaladas por el usuario, descartando las aplicaciones de sistema (`/system/app`).

Se identificó una aplicación atípica instalada manualmente:

- **Nombre del Paquete:** `org.troncoso.droidpond`
- **Ruta:** `/data/app/org.troncoso.droidpond-1`
- **Permisos:** `android.permission.INTERNET`

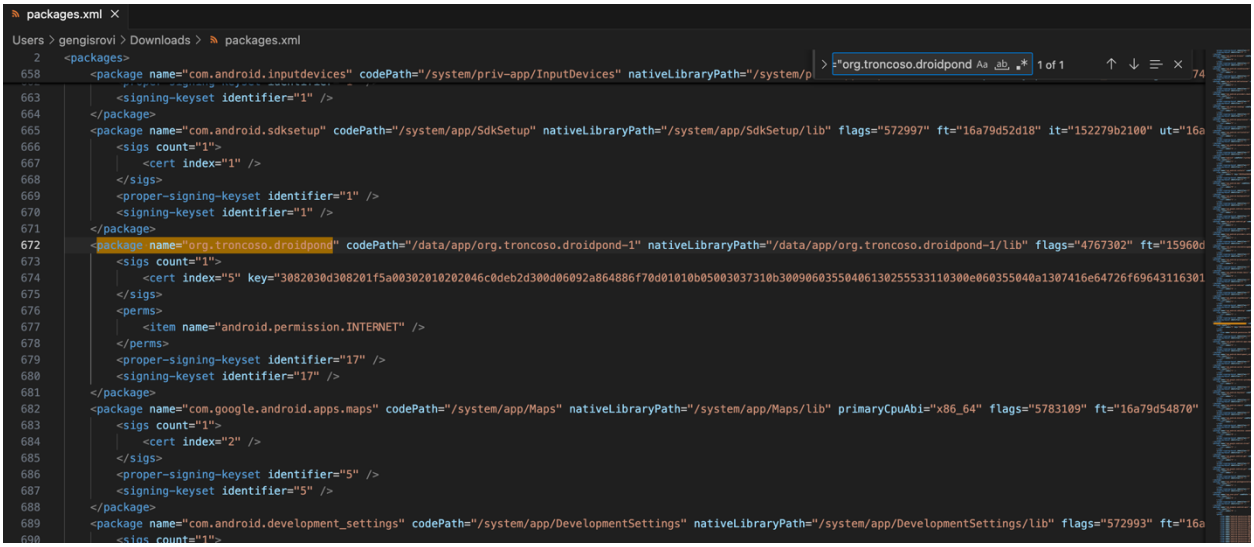


Figura 7. Identificación de la aplicación objetivo instalada en el emulador.

Esta aplicación (`DroidPond`) no corresponde a software comercial estándar, lo que sugiere que es el software objetivo sobre el cual Mr. X estaba intentando realizar la ingeniería inversa o el ataque de fuerza bruta mencionado en los SMS

Conclusiones

La investigación ha permitido reconstruir la cadena delictiva completa. Las evidencias demuestran que Mr. X utilizó un **emulador Android versión 5.1.1** para ocultar su identidad física. En este entorno virtual, instaló la aplicación objetivo **"DroidPond" (`org.troncoso.droidpond`)**, perteneciente a la organización atacada.

La trama consistía en vulnerar la seguridad de esta aplicación específica. Para ello:

1. Contactó inicialmente a **Matt (650-555-1111)** sin éxito.
2. Contrató los servicios de **Danny Rand (650-555-2222)** por una suma de 10.000 (10k) semanales para labores de descifrado.

3. Obtuvo las herramientas de hacking (alojadas en el archivo `shadow`) a través de un enlace proporcionado por **Jessica (650-555-4444)**, al cual accedió desde el navegador nativo del emulador.

El análisis cruzado de los SMS (`mmssms.db`), el historial de navegación (`browser2.db`) y el registro de sistema (`packages.xml`) confirma de manera irrefutable la intencionalidad, los medios técnicos y los cómplices involucrados en el ciberataque.

---

3. DOCKER: INFRAESTRUCTURA Y DESPLIEGUE

En esta sección se resuelven los ejercicios de contenedorización, aplicando principios de seguridad como el principio de mínimo privilegio (usuarios no root) y persistencia de datos.

3.1. Ejercicio 1: Generador de ASCII Art

Se solicita una imagen Docker que convierta imágenes a ASCII art y guarde el resultado en un volumen persistente.

Código del Dockerfile: Se ha utilizado una imagen base ligera (`ubuntu:20.04`) y se ha creado un usuario específico para evitar la ejecución como `root`.

Dockerfile
```
FROM ubuntu:20.04
LABEL maintainer="Estudiante"
ENV DEBIAN_FRONTEND=noninteractive

# Instalación de jp2a y limpieza de caché para optimizar capas [cite: 1249]
RUN apt-get update && apt-get install -y jp2a \
    && rm -rf /var/lib/apt/lists/*

# Creación de usuario no-root por seguridad
RUN useradd -m appuser
RUN mkdir /datos && chown appuser:appuser /datos

# Definición del volumen persistente [cite: 1332]
VOLUME /datos

USER appuser
WORKDIR /datos
ENTRYPOINT ["jp2a"]
```

Proceso de Construcción y Ejecución:

1. **Construcción de la imagen:** `docker build -t ascii-art-converter .`

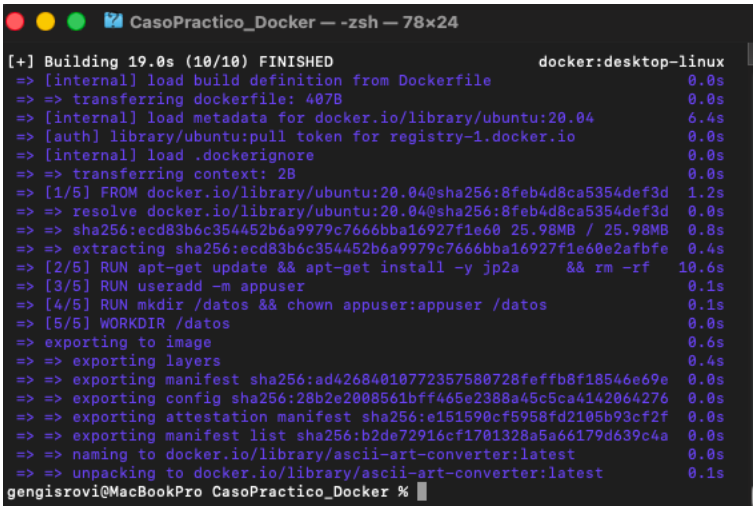


Figura 7. Construcción de la imagen Docker.

2. **Creación del volumen:** `docker volume create asciiart`

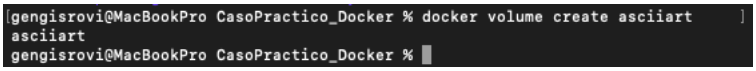


Figura 8. Creación del volumen persistente.

3. **Ejecución y Prueba:** Se ejecutó el contenedor mapeando una imagen local (`test.jpg`) y guardando la salida en el volumen. `docker run --rm -e TERM=xterm -v $(pwd):/entrada -v asciiart:/datos ascii-art-converter --output=/datos/resultado.txt /entrada/test.jpg`

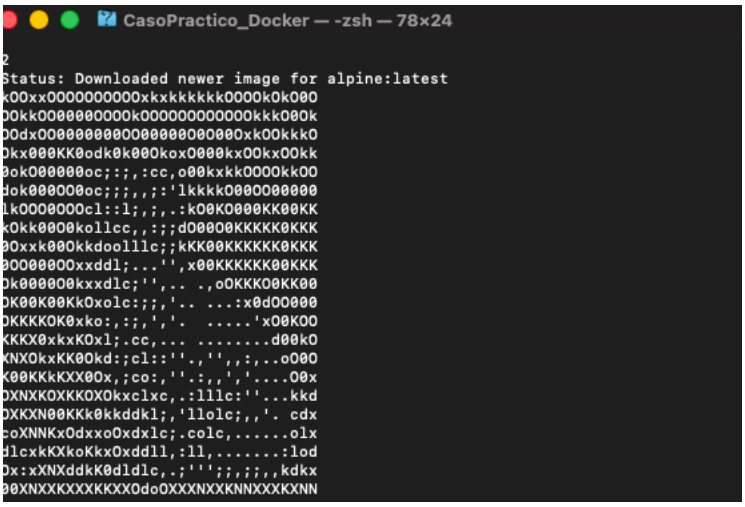


Figura 9. Ejecución exitosa y generación del fichero en el volumen.

3.2. Ejercicio 2: Arquitectura WordPress + MySQL

Se despliega una arquitectura web con base de datos, garantizando la persistencia mediante volúmenes y la comunicación mediante una red interna.

3.2.1. Despliegue mediante Docker Compose

Se ha optado por Docker Compose para la orquestación, ya que permite definir la infraestructura como código de manera clara y reproducible.

Fichero `docker-compose.yml`:

YAML

```
version: '3.8'

services:
  db:
    image: mysql:5.7
    container_name: mysql_db
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: password_root
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wp_user
      MYSQL_PASSWORD: wp_password
    volumes:
      - db_data:/var/lib/mysql
    networks:
      - wp-net

  wordpress:
    image: wordpress:latest
    container_name: wordpress_app
    restart: always
    depends_on:
      - db
    ports:
      - "8080:80"
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_USER: wp_user
      WORDPRESS_DB_PASSWORD: wp_password
      WORDPRESS_DB_NAME: wordpress
    volumes:
      - wp_data:/var/www/html
    networks:
      - wp-net

volumes:
  db_data:
  wp_data:

networks:
  wp-net:
```

```
driver: bridge
```

Despliegue y Verificación:

1. **Ejecución del entorno:** `docker-compose up -d`

Figura 10. Despliegue de la pila de servicios con Docker Compose.

2. **Estado de los contenedores:** `docker-compose ps`

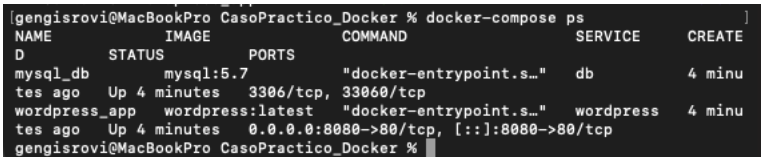


Figura 11. Verificación de contenedores activos.

3. **Comprobación funcional:** Acceso a través del navegador a `http://localhost:8080`.

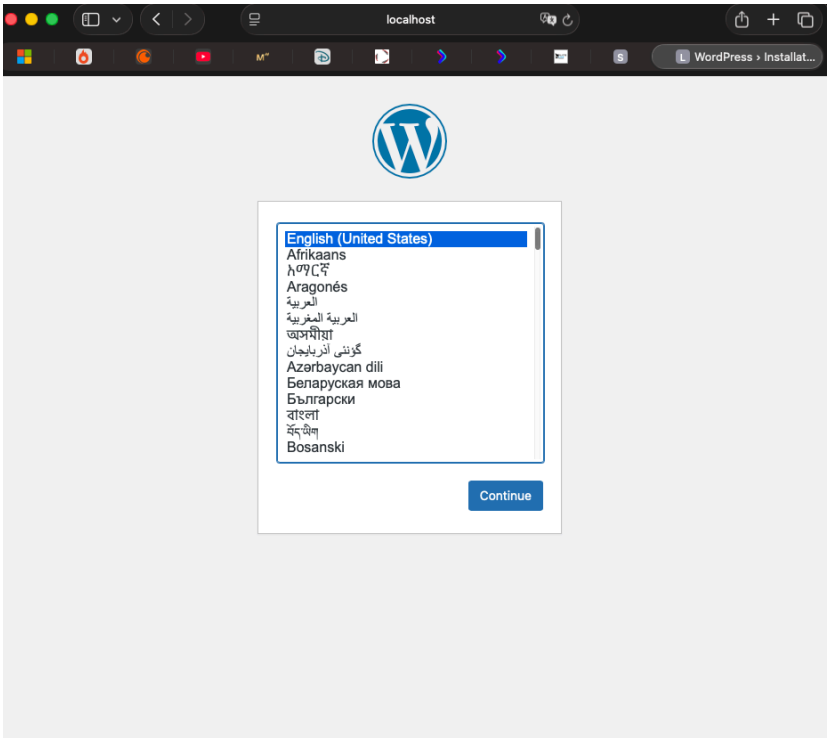


Figura 12. Acceso exitoso a la aplicación WordPress.

4. CONCLUSIONES

A través del desarrollo de este caso práctico se han alcanzado las siguientes conclusiones:

1. **Análisis Forense:** La extracción y análisis de bases de datos SQLite específicas (`mmssms.db`, `contacts2.db`, `browser2.db`) alojadas en las particiones de datos del sistema Android es fundamental para reconstruir la cronología de los hechos delictivos. El uso de herramientas como *The Sleuth Kit* permitió acceder a la información sin comprometer la integridad de la imagen original.
2. **Seguridad en Docker:** La implementación de contenedores debe seguir principios de seguridad desde el diseño. En el ejercicio 1, se demostró cómo evitar la ejecución como `root` mediante la directiva `USER` en el Dockerfile.
3. **Orquestación y Persistencia:** El uso de Docker Compose simplifica drásticamente el despliegue de arquitecturas multicontenedor (WordPress + MySQL), gestionando automáticamente la red interna para la comunicación segura entre servicios y los volúmenes para garantizar que la información no se pierda al reiniciar los contenedores.