

**UNIVERSIDAD PRIVADA DE TACNA
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA DE SISTEMAS**



Informe de Laboratorio

Laboratorio 02 “Auditoría Móvil”

Que se presenta para el curso:
“Auditoría de sistemas”

Integrante(s):

- Martinez Yufra, Ericka Esther 2018000368

Docente:

Dr. Oscar Juan Jimenez Flores

**TACNA – PERÚ
2025**

CONTENIDO

RESUMEN	3
INTRODUCCIÓN	4
1. Información sobre el evento práctico	5
1.1. Título del evento práctico.....	5
1.2. Objetivos.....	5
1.3. Tiempo de duración (horas)	5
1.4. Resultados de Aprendizaje (RA)	5
1.5. Recursos (Equipos, materiales, programas y otros).....	5
2. Procedimiento o Metodología	6
2.1. Construcción de la imagen Docker	6
2.2. Levantamiento de contenedores	7
2.3. Acceso a la aplicación Mobile Audit	9
2.4. Ejecución del análisis de un APK: F-Droid	10
2.5. Ejecución del análisis de un APK: DailyTube - media player	26
2.6. Integración en la API Swagger y Redoc	41
3. Conclusiones	48
4. Referencias Bibliográficas	49

RESUMEN

En este laboratorio se desplegó Mobile Audit utilizando Docker Compose, construyendo imágenes sobre distintas versiones de Python y levantando contenedores que incluyen PostgreSQL, Nginx, RabbitMQ, Celery y la aplicación en Django. El entorno quedó totalmente operativo, permitiendo el análisis de APKs mediante un proceso automatizado de detección de vulnerabilidades, análisis de permisos, revisión de actividades y componentes, así como la evaluación de certificados y cadenas de texto. La interfaz web ofreció una visualización integral de los resultados, centralizando los hallazgos de seguridad y las prácticas implementadas en cada aplicación.

Se ejecutaron análisis sobre los APKs F-Droid y DailyTube - media player, detectando un total de 19,215 hallazgos en F-Droid y 9,090 en DailyTube. Se identificaron vulnerabilidades críticas, permisos peligrosos, componentes mal configurados y exposiciones de información sensible. Asimismo, se evaluaron las buenas prácticas de seguridad implementadas, como SSL Pinning, verificación de depuración, protección frente a root y generación segura de números aleatorios, evidenciando tanto fortalezas como áreas de mejora en las aplicaciones auditadas.

INTRODUCCIÓN

El presente laboratorio se centró en la auditoría de aplicaciones móviles mediante el despliegue de la plataforma Mobile Audit en contenedores Docker. Se abordó la construcción de imágenes, el levantamiento de servicios esenciales como base de datos, servidor web, sistema de mensajería y trabajadores asíncronos, así como la integración de estos elementos para proporcionar un entorno funcional completo. Además, se realizaron pruebas de análisis estático de seguridad sobre aplicaciones Android, enfocándose en la identificación de vulnerabilidades y exposición de datos sensibles.

Durante la actividad, se exploró la funcionalidad de Mobile Audit para el escaneo de APKs, incluyendo la revisión de componentes internos, permisos, actividades, certificados digitales y prácticas de seguridad implementadas. Se complementó el laboratorio con la documentación interactiva de la API mediante Swagger y ReDoc, lo que permitió comprender la estructura de los endpoints, la gestión de hallazgos y la interacción con los servicios web, asegurando la trazabilidad de cada operación en la auditoría de seguridad móvil.

GUÍA DE LABORATORIO N.º 02

“AUDITORIA MÓVIL”

1. Información sobre el evento práctico

1.1. Título del evento práctico

Laboratorio 02. Auditoría Móvil

1.2. Objetivos

- Identificar los elementos que participan en la Auditoría de seguridad.
- Analizar los posibles riesgos asociados.
- Proponer controles efectivos para minimizar el riesgo asociado.

1.3. Tiempo de duración (horas)

06 horas académicas

1.4. Resultados de Aprendizaje (RA)

[AG-I02] Ética

[AG-I04] Comunicación

[AG-I07] Conocimientos de Ingeniería

[AG-I08] Análisis de Problemas

[AG-I09] Diseño y Desarrollo de Soluciones

[AG-I11] Uso de Herramientas

1.5. Recursos (Equipos, materiales, programas y otros)

- Computador con S.O. Windows
- Descargar e instalar repositorio GIT
<https://github.com/OscarJimenezFlores/CursoAuditoria/tree/main/AuditoriaMovil>
- Repositorio GitHub de evidencia
<https://github.com/ErickaEmy/AuditoriaMovil>

2. Procedimiento o Metodología

2.1. Construcción de la imagen Docker

El sistema cuenta con imágenes base alojadas en Docker Hub, construidas sobre python:buster, en diferentes versiones:

- mpast/mobile_audit:3.0.0 → python:3.9.16-buster
- mpast/mobile_audit:2.2.1 → python:3.9.7-buster
- mpast/mobile_audit:1.3.8 → python:3.9.4-buster
- mpast/mobile_audit:1.0.0 → python:3.9.0-buster

Mobile Audit provee un archivo docker-compose.yml que define los servicios requeridos. Para construir las imágenes se ejecuta: docker-compose build

Figura 1: Construcción de la imagen Docker de Mobile Audit

```
PS D:\UPT 2025-II\Auditoria\Laboratorio 02\AuditoriaMovil> docker-compose build
time="2025-08-20T07:21:18-05:00" level=warning msg="D:\\UPT 2025-II\\Auditoria\\L
laboratorio 02\\AuditoriaMovil\\docker-compose.yml: the attribute `version` is ob
solete, it will be ignored, please remove it to avoid potential confusion"
#1 [internal] load local bake definitions
#1 reading from stdin 566B 0.1s done
#1 DONE 0.1s

#2 [internal] load build definition from Dockerfile
#2 transferring dockerfile: 1.59kB 0.0s done
#2 DONE 0.2s

#3 [internal] load metadata for docker.io/library/python:3.10-bullseye@sha256:02c
7cb92b8f23908de6457f7800c93b84ed8c6e7201da7935443d4c5eca7b381
#3 DONE 3.7s

#4 [internal] load .dockerignore

#20 exporting attestation manifest sha256:6e0c3d742c4cc0a3db9f0bfac883cd2ef1fb1a0
648433fdcaa926fcf462c5b93 0.1s done
#20 exporting manifest list sha256:21b73c2aa54ff9ffc7de51d8a30c12d8342c0e531d358a
696e97f88ac48348e2
#20 exporting manifest list sha256:21b73c2aa54ff9ffc7de51d8a30c12d8342c0e531d358a
696e97f88ac48348e2 0.1s done
#20 naming to docker.io/library/mobile_audit:latest 0.0s done
#20 unpacking to docker.io/library/mobile_audit:latest
#20 unpacking to docker.io/library/mobile_audit:latest 37.8s done
#20 DONE 100.4s

#21 resolving provenance for metadata file
#21 DONE 0.1s
[+] Building 1/1
✓ mobile_audit Built 0.0s
```

Fuente: Elaboración propia.

Nota: La figura muestra el proceso de construcción de la imagen a partir del archivo Dockerfile definido en el proyecto. En este paso, Docker descarga las dependencias, configura el entorno Python y prepara la aplicación para ser ejecutada en contenedores.

2.2. Levantamiento de contenedores

Una vez construida la imagen, se inician los servicios definidos en el docker-compose.yml: docker-compose up.

Figura 2: Ejecución de contenedores Mobile Audit

```
PS D:\UPT 2025-II\Auditoria\Laboratorio 02\AuditoriaMovil> docker-compose up
time="2025-08-20T07:34:46-05:00" level=warning msg="D:\UPT 2025-II\Auditoria\L
laboratorio 02\AuditoriaMovil\docker-compose.yaml: the attribute `version` is ob
solete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 34/34
 ✓ db Pulled                                178.7s
 ✓ nginx Pulled                             106.3s
 ✓ rabbitmq Pulled                          145.9s

gned) fd 3
web-1      | Python version: 3.10.13 (main, Feb 13 2024, 10:45:10) [GCC 10.2.1 2
0210110]
web-1      | Python main interpreter initialized at 0x6418050d6c10
web-1      | python threads support enabled
web-1      | your server socket listen backlog is limited to 100 connections
web-1      | your mercy for graceful operations on workers is 60 seconds
web-1      | mapped 145840 bytes (142 KB) for 2 cores
web-1      | *** Operational MODE: preforking ***
web-1      | WSGI app 0 (mountpoint='') ready in 1 seconds on interpreter 0x6418
050d6c10 pid: 11 (default app)
web-1      | *** uWSGI is running in multiple interpreter mode ***
web-1      | spawned uWSGI worker 1 (pid: 11, cores: 1)
web-1      | spawned uWSGI worker 2 (pid: 13, cores: 1)
db-1       | 2025-08-20 12:43:00.725 UTC [62] LOG: checkpoint starting: time
db-1       | 2025-08-20 12:43:31.258 UTC [62] LOG: checkpoint complete: wrote 2
96 buffers (1.8%); 0 WAL file(s) added, 0 removed, 0 recycled; write=29.427 s, sy
nc=1.020 s, total=30.534 s; sync files=249, longest=0.012 s, average=0.005 s; dis
tance=1766 kB, estimate=1766 kB; lsn=0/1AC7BF0, redo lsn=0/1AC7BB8
```

Fuente: Elaboración propia.

Nota: La figura evidencia cómo Docker inicia los servicios asociados: PostgreSQL, Nginx, RabbitMQ, Celery y la aplicación web en Django. El resultado es un entorno completo para el análisis estático de aplicaciones móviles.

Opcionalmente, se pueden levantar en segundo plano: `docker-compose up -d`

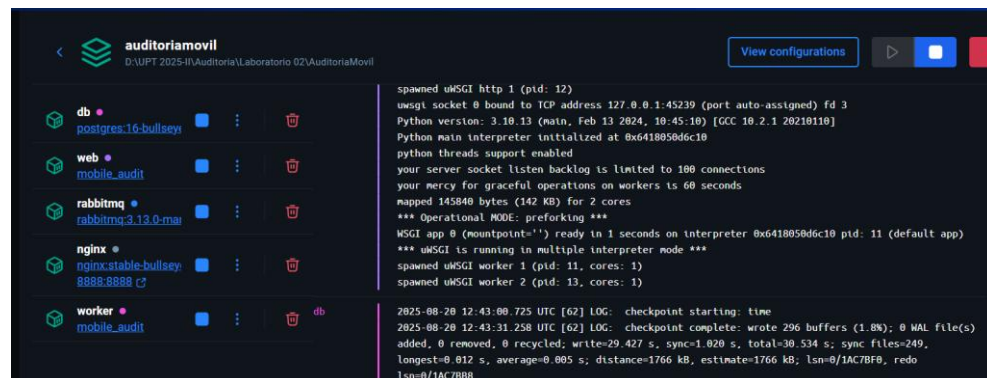
Figura 3: Ejecución de los contenedores de Auditoría Móvil mediante Docker Compose

```
PS D:\UPT 2025-II\Auditoria\Laboratorio 02\AuditoriaMovil> docker-compose up -d
time="2025-08-20T08:00:19-05:00" level=warning msg="D:\UPT 2025-II\Auditoria\Laboratorio 02\AuditoriaMovil\docker-compose
.yaml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 5/5
  ✓ Container auditoriamovil-db-1      Started      0.5s
  ✓ Container auditoriamovil-web-1     Started      0.6s
  ✓ Container auditoriamovil-rabbitmq-1 Started      0.6s
  ✓ Container auditoriamovil-nginx-1   Started      0.7s
  ✓ Container auditoriamovil-worker-1  Started      0.6s
```

Fuente: Elaboración propia.

Nota: En la figura se observa el panel de Docker Desktop mostrando la ejecución de los contenedores correspondientes al despliegue de la aplicación Auditoría Móvil. Se evidencian los servicios db (PostgreSQL 16), web (aplicación Mobile Audit en Django), rabbitmq (gestión de mensajería), nginx (servidor proxy inverso) y worker (procesamiento asíncrono con Celery). La salida de la consola indica que los contenedores se encuentran en ejecución, donde por ejemplo el servicio web ha iniciado correctamente bajo uWSGI con múltiples intérpretes y el servicio de base de datos ha completado su checkpoint. Esto valida que la instrucción `docker-compose up -d` ha desplegado exitosamente la infraestructura en segundo plano, permitiendo liberar la terminal mientras los servicios permanecen activos para posteriores pruebas y análisis del sistema.

Figura 4: Ejecución de contenedores de Mobile Audit en Docker Desktop



Fuente: Elaboración propia.

Nota: En la figura se observa la ejecución de los distintos contenedores que conforman la aplicación Mobile Audit a través de Docker Desktop, levantados mediante el comando `docker-compose up`. Los servicios desplegados corresponden a la base de datos PostgreSQL, el servidor web Nginx, el sistema de mensajería RabbitMQ, el servicio de trabajadores Celery y la aplicación web principal desarrollada en Django. En la parte derecha se visualizan los registros de logs generados en tiempo real, donde se aprecia el arranque exitoso del servidor uWSGI que gestiona las peticiones HTTP hacia la aplicación, así como la correcta inicialización del contenedor de base de datos con los procesos de checkpoint de PostgreSQL. Esta evidencia confirma que el entorno de análisis estático de seguridad y detección de malware para archivos APK se encuentra en funcionamiento y listo para ser accedido desde el navegador en el puerto 8888.

2.3. Acceso a la aplicación Mobile Audit

Una vez que todos los contenedores están en ejecución, se accede a Mobile Audit desde el navegador en la siguiente dirección:

- Entorno estándar: <http://localhost:8888/>
- Entorno TLS (opcional): <https://localhost/>

Figura 5: Acceso a la interfaz web de Mobile Audit en el navegador



Fuente: Elaboración propia.

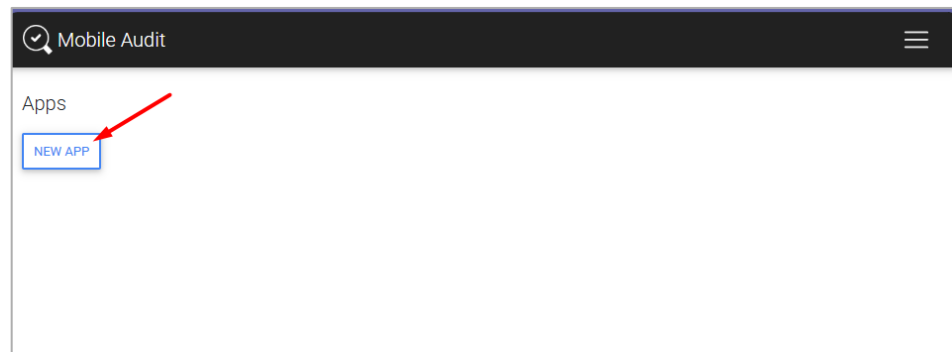
Nota: La figura muestra el acceso exitoso a la aplicación web Mobile Audit a través del navegador en la dirección <http://localhost:8888/>, una vez desplegados todos los contenedores mediante Docker Compose. En la parte superior se observa la barra de navegación con las secciones principales de la plataforma, como Home, Findings, Create y Others, junto con la opción de autenticación (Login). En la sección inferior aparece el botón New App, que permite al usuario cargar un archivo APK para su análisis. Esta interfaz confirma que los servicios web, base de datos, proxy inverso y worker se encuentran correctamente integrados y en funcionamiento, brindando un entorno listo para realizar auditorías móviles mediante análisis estático de seguridad y detección de malware en aplicaciones Android.

2.4. Ejecución del análisis de un APK: F-Droid

Se analizará el APK F-Droid, por lo que se ejecutarán los siguientes pasos:

1. Desde la interfaz, se selecciona “NEW APP” para cargar un archivo APK.

Figura 6: Pantalla inicial de Mobile Audit con opción “New App”



Fuente: Elaboración propia.

Nota: La figura muestra la interfaz principal de Mobile Audit accesible en <http://localhost:8888/>, con la barra de navegación superior y el botón New App disponible en la sección Apps; esta vista confirma que la plataforma está operativa y lista para registrar una nueva aplicación, paso imprescindible para iniciar el proceso de auditoría sobre el APK “F-Droid”.

2. Carga el APK “F-Droid” e inicio del escaneo

En el formulario New App, completa:

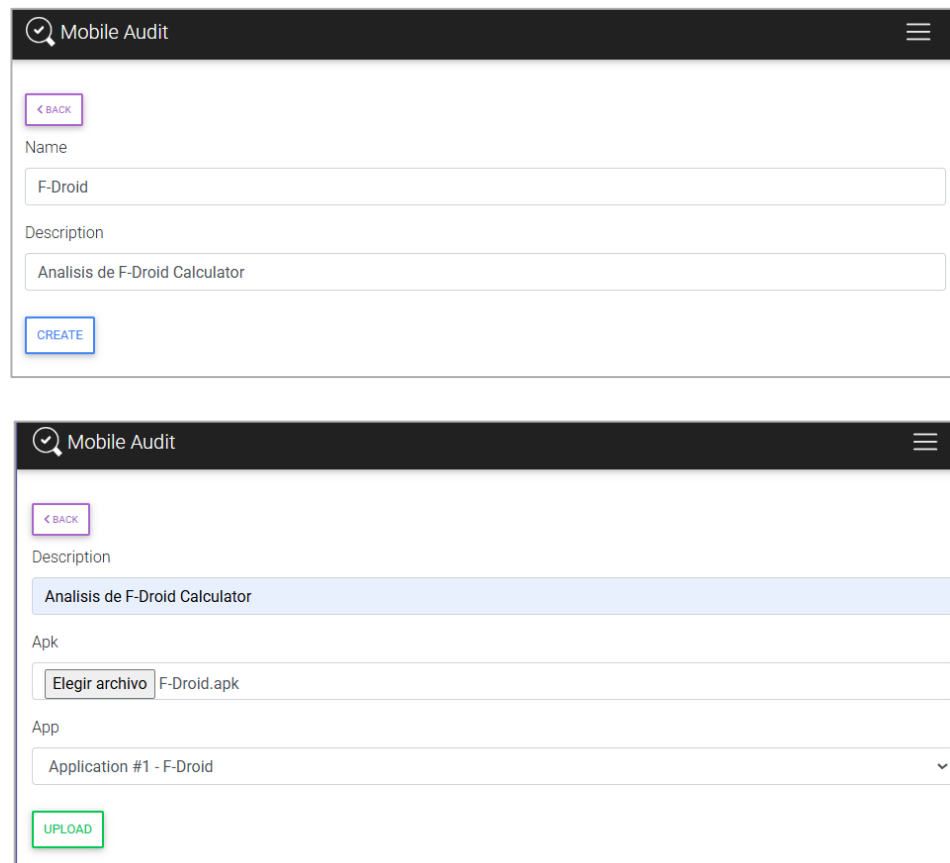
- Name: F-Droid (o el identificador que prefieras).
- APK file: Selecciona el archivo F-Droid.apk desde tu equipo.
- (Opcional) Description/Tags para clasificar el análisis.

Pulsa Create o Upload para enviar el APK.

El sistema encola la tarea al worker (Celery) y comienza:

- Análisis estático (SAST) con el motor de patrones/reglas.
- Análisis de malware (URLs maliciosas y firmas en strings; VirusTotal si está habilitado).

Figura 7: Formulario de carga de APK “F-Droid”



The figure consists of two screenshots of the 'Mobile Audit' application interface. The top screenshot shows the 'New App' form with the 'Name' field set to 'F-Droid' and the 'Description' field set to 'Análisis de F-Droid Calculator'. A 'CREATE' button is visible at the bottom. The bottom screenshot shows the same form with the 'Description' field highlighted in blue. Below the 'Description' field, there is an 'Apk' section with a file selection button labeled 'Elegir archivo' and the file 'F-Droid.apk' selected. Below the 'Apk' section, there is an 'App' section with a dropdown menu showing 'Application #1 - F-Droid'. An 'UPLOAD' button is visible at the bottom of the second screenshot.

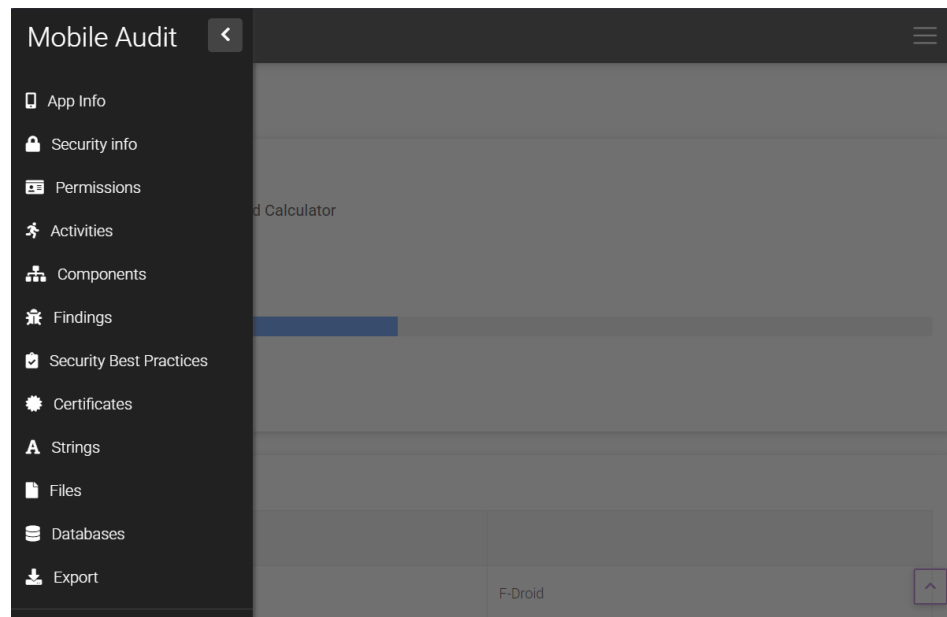
Fuente: Elaboración propia.

Nota: La figura evidencia el registro de la aplicación “F-Droid” y la selección del archivo F-Droid.apk para su análisis; al enviar el formulario, el backend Django recibe el artefacto, lo almacena y delega su procesamiento al worker Celery a través de RabbitMQ, dejando el estado del escaneo en curso y habilitando la posterior visualización de resultados.

3. Navegación por la barra lateral del escaneo

Una vez finalizado el procesamiento, se habilita una vista de resultados con barra lateral izquierda que agrupa la información del análisis. Usa esa barra para recorrer cada sección y toma las evidencias indicadas a continuación.

Figura 8: Barra lateral del escaneo y menú de secciones



Fuente: Elaboración propia.

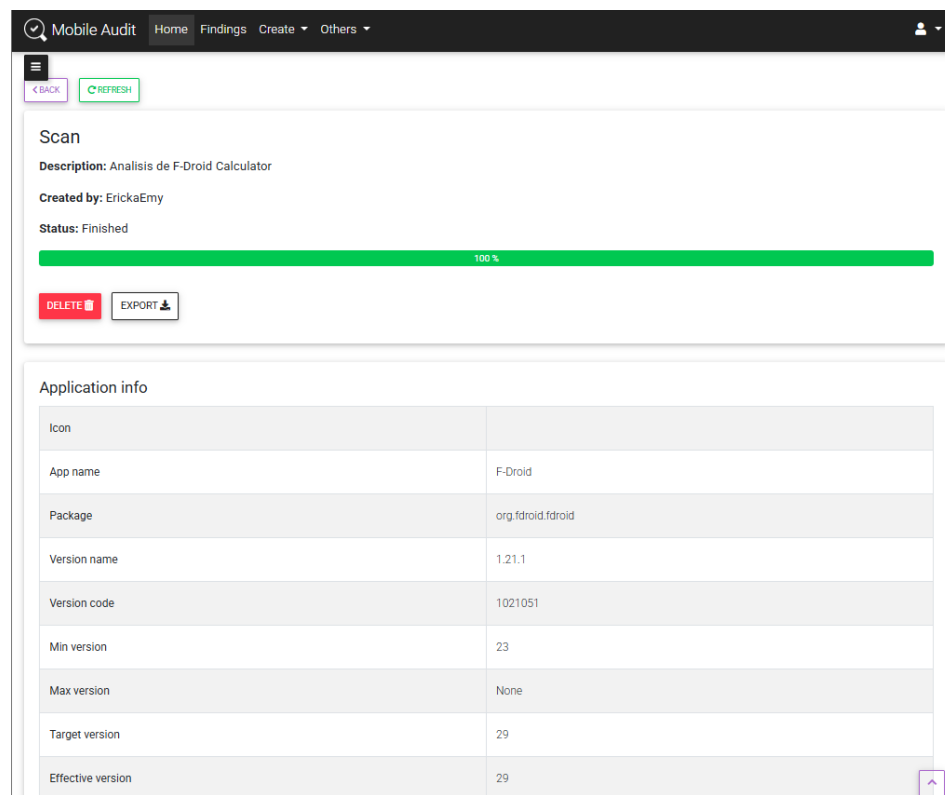
Nota: La figura muestra la organización de resultados en la barra lateral izquierda que centraliza el acceso a Información de la aplicación, Información de seguridad, Componentes, Hallazgos SAST, Mejores prácticas implementadas, VirusTotal, Certificados, Cadenas, Bases de datos y Archivos, demostrando que el análisis del APK “F-Droid” ha

concluido y que los hallazgos están disponibles para su revisión estructurada.

4. Información de la aplicación

- Ruta en la UI: dentro del escaneo → Application info / App info.
- Qué validar/explicar: nombre de paquete (package), versión (versionName/versionCode), minSdkVersion, targetSdkVersion, permisos declarados, y metadatos del AndroidManifest.xml.

Figura 9: Acceso a la interfaz web de Mobile Audit en el navegador



Fuente: Elaboración propia.

Nota: La figura presenta la ficha técnica extraída del APK “F-Droid”, incluyendo identificador de paquete, versión, niveles de SDK mínimos y objetivo, así como el conjunto de permisos declarados en el manifiesto; esta información permite contextualizar el alcance del binario y entender la superficie de permisos que puede solicitar en tiempo de ejecución.

5. Security Info

La sección Security Info dentro del análisis realizado a la aplicación F-Droid muestra un resumen general de los hallazgos de seguridad identificados en el archivo APK durante el proceso de análisis estático. Este módulo categoriza las vulnerabilidades encontradas según su severidad (Critical, High, Medium, Low y None), lo que permite priorizar de manera efectiva los riesgos más relevantes para la seguridad. En este caso, se identificaron 19.215 hallazgos en total, entre los cuales destacan 54 vulnerabilidades críticas que representan riesgos de mayor impacto, así como 3.424 catalogadas como de severidad alta, lo que refleja un nivel considerable de exposición que debe ser gestionado de manera prioritaria.

Figura 10: Resultados de seguridad del APK F- Droid en Mobile Audit

Security info

Number of findings	19215
By Severity	
Critical	54
High	3424
Medium	2824
Low	1654
None	11259

Fuente: Elaboración propia.

Nota: La figura muestra la salida generada por el módulo de Security Info tras el análisis del APK de F-Droid. En la gráfica se presentan los hallazgos clasificados por nivel de severidad, donde se observa un total de 19.215 vulnerabilidades detectadas. Entre ellas, 54 corresponden a nivel crítico, 3.424 a nivel alto, 2.824 a nivel medio, 1.654 a nivel bajo y 11.259 sin severidad asignada. Este resultado evidencia que, aunque la

mayoría de los hallazgos no representan un riesgo inmediato (clase None), existe una cantidad significativa de vulnerabilidades de impacto medio y alto que deben ser atendidas para fortalecer la seguridad de la aplicación.

6. Permissions

La sección Permissions dentro del análisis de seguridad estático de F-Droid enumera todos los permisos que la aplicación solicita durante su instalación y ejecución en Android. Estos permisos son evaluados en función de su tipo (Normal, Dangerous, Special, Other) y clasificados por niveles de severidad, ya que el abuso o la mala gestión de permisos puede comprometer la seguridad y privacidad del usuario. En este caso, el análisis arrojó un total de 28 permisos solicitados, entre los que destacan permisos de alto riesgo como el acceso a la cámara, almacenamiento externo, notificaciones y control de paquetes, los cuales requieren especial atención para evitar posibles vectores de ataque.

Figura 11: Acceso a la interfaz web de Mobile Audit en el navegador

Permissions						
Search: <input type="text"/>						
ID	Name	Type	Severity	Status		
3	android.permission.ACCESS_NETWORK_STATE	Normal	Low	✓		
24	android.permission.ACCESS_WIFI_STATE	Normal	Low	✓		
16	android.permission.BLUETOOTH	Normal	Medium	⚠		
23	android.permission.BLUETOOTH_ADMIN	Normal	High	❌		
20	android.permission.BLUETOOTH_CONNECT	Other	High	❌		
11	android.permission.BLUETOOTH_SCAN	Other	High	❌		
13	android.permission.CAMERA	Dangerous	High	❌		
1	android.permission.CHANGE_NETWORK_STATE	Normal	Low	✓		
21	android.permission.CHANGE_WIFI_MULTICAST_STATE	Normal	Low	✓		
22	android.permission.CHANGE_WIFI_STATE	Normal	Low	✓		
14	android.permission.ENFORCE_UPDATE_OWNERSHIP	Other	High	❌		
15	android.permission.INTERNET	Normal	Medium	⚠		
16	android.permission.BLUETOOTH	Normal	Medium	⚠		
17	android.permission.WRITE_EXTERNAL_STORAGE	Dangerous	High	❌		
18	android.permission.WRITE_SETTINGS	Special	High	❌		
19	android.permission.REQUEST_INSTALL_PACKAGES	Normal	Low	✓		
20	android.permission.BLUETOOTH_CONNECT	Other	High	❌		

21	android.permission.CHANGE_WIFI_MULTICAST_STATE	Normal	Low	●
22	android.permission.CHANGE_WIFI_STATE	Normal	Low	●
23	android.permission.BLUETOOTH_ADMIN	Normal	High	●
24	android.permission.ACCESS_WIFI_STATE	Normal	Low	●
25	android.permission.REQUEST_DELETE_PACKAGES	Other	High	●
26	android.permission.READ_EXTERNAL_STORAGE	Dangerous	High	●
27	android.permission.QUERY_ALL_PACKAGES	Other	High	●
28	org.f-droid.f-droid.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION	Other	High	●

Showing 1 to 28 of 28 entries

Fuente: Elaboración propia.

Nota: La figura presenta los resultados obtenidos en el apartado de Permissions, donde se listan los 28 permisos requeridos por la aplicación F-Droid junto con su tipo, severidad y estado. El análisis evidencia que la aplicación solicita permisos de nivel bajo y medio como el acceso a red, Bluetooth o NFC, pero también incluye permisos clasificados como de severidad alta, entre ellos android.permission.CAMERA, WRITE_EXTERNAL_STORAGE, READ_EXTERNAL_STORAGE, BLUETOOTH_ADMIN, POST_NOTIFICATIONS y otros relacionados con la gestión de paquetes y almacenamiento. Estos hallazgos son relevantes, ya que los permisos catalogados como Dangerous, Special u Other con severidad alta representan potenciales riesgos de seguridad si son explotados por código malicioso o configuraciones inseguras. De esta manera, el análisis permite identificar de forma precisa qué permisos podrían ser sensibles y deben ser controlados para garantizar la seguridad del sistema.


7. Activities

La sección Activities dentro del análisis del APK “F-Droid” permite identificar todas las actividades declaradas en la aplicación, es decir, las pantallas o componentes interactivos que conforman su flujo de uso en Android. Cada actividad se encuentra listada con su identificador de clase completo, lo que facilita comprender la estructura funcional y la manera en que la aplicación organiza sus vistas, menús y módulos. Este análisis resulta fundamental para auditar posibles puntos de entrada, revisar el

comportamiento de cada actividad y determinar si alguna de ellas puede ser mal utilizada en escenarios de ataque.

Figura 12: Listado de Activities detectadas en el APK “F-Droid”

Activities


Search:

ID	Name	Main
1	org.fdroid.fdroid.nearby.SwapWorkflowActivity	✗
2	org.fdroid.fdroid.panic.PanicPreferencesActivity	✗
3	org.fdroid.fdroid.panic.SelectInstalledAppsActivity	✗
4	org.fdroid.fdroid.panic.PanicResponderActivity	✗
5	org.fdroid.fdroid.panic.ExitActivity	✗
6	org.fdroid.fdroid.panic.CalculatorActivity	✓
7	org.fdroid.fdroid.privileged.views.InstallConfirmActivity	✗
8	org.fdroid.fdroid.privileged.views.UninstallDialogActivity	✗
9	org.fdroid.fdroid.views.repos.ManageReposActivity	✗
10	org.fdroid.fdroid.views.repos.AddRepoActivity	✗
11	org.fdroid.fdroid.views.IpfsGatewaySettingsActivity	✗
12	org.fdroid.fdroid.views.IpfsGatewayAddActivity	✗

Fuente: Elaboración propia.

Nota: La figura presenta el resultado del análisis de las actividades incluidas en el APK “F-Droid”, mostrando un total de 25 entradas obtenidas desde el manifiesto de la aplicación. El listado evidencia que la gran mayoría de las actividades no están configuradas como Main, pues aparecen con una marca de rechazo, mientras que únicamente CalculatorActivity y MainActivity cuentan con aprobación como actividades válidas de entrada. Este resultado confirma la estructura interna de navegación de la aplicación, en la cual solo la actividad

MainActivity cumple la función de interfaz principal y CalculatorActivity se encuentra habilitada como un punto alternativo de acceso. El resto de las actividades, aunque forman parte de la lógica de la aplicación, solo son accesibles de manera secundaria o a través de invocaciones internas. De esta forma, el análisis permite comprender cómo está definida la arquitectura de pantallas y cuáles son efectivamente ejecutables como puntos de inicio dentro del sistema Android.

8. Components

El análisis de Components dentro del APK “F-Droid” permite identificar todos los elementos declarados en el manifiesto (AndroidManifest.xml) que forman parte de la estructura funcional de la aplicación. Estos componentes incluyen activities, services, receivers y content providers, cada uno con sus respectivos intent-filters, acciones y categorías, los cuales determinan cómo interactúa la aplicación tanto de manera interna como con el sistema operativo Android. En total, el sistema detectó 60 componentes, entre los que destacan 25 actividades (Activities), 16 servicios (Services), 12 receptores (Broadcast Receivers) y 4 proveedores de contenido (Providers). Esta información es esencial para comprender la arquitectura interna de la aplicación, la manera en que se gestionan las interacciones entre procesos y los posibles vectores de exposición frente a amenazas de seguridad.

Figura 13: Componentes detectados en el análisis del APK “F-Droid”

Components

Search:

ID	Type	Name	Intents												
1	activity	org.fdroid.fdroid.nearby.SwapWorkflowActivity													
2	activity	org.fdroid.fdroid.panic.PanicPreferencesActivity	<table><tr><th>ID</th><th>Intent</th><th>Action</th></tr><tr><td>1</td><td>info.guardianproject.panic.action.CONNECT</td><td>action</td></tr><tr><td>2</td><td>info.guardianproject.panic.action.DISCONNECT</td><td>action</td></tr><tr><td>3</td><td>android.intent.category.DEFAULT</td><td>category</td></tr></table>	ID	Intent	Action	1	info.guardianproject.panic.action.CONNECT	action	2	info.guardianproject.panic.action.DISCONNECT	action	3	android.intent.category.DEFAULT	category
ID	Intent	Action													
1	info.guardianproject.panic.action.CONNECT	action													
2	info.guardianproject.panic.action.DISCONNECT	action													
3	android.intent.category.DEFAULT	category													

54	receiver	androidx.work.impl.background.systemalarm.ConstraintProxyUpdateReceiver	<table><tr><th>ID</th><th>Intent</th><th>Action</th></tr><tr><td>118</td><td>androidx.work.impl.background.systemalarm.UpdateProxies</td><td>action</td></tr></table>	ID	Intent	Action	118	androidx.work.impl.background.systemalarm.UpdateProxies	action									
ID	Intent	Action																
118	androidx.work.impl.background.systemalarm.UpdateProxies	action																
55	receiver	androidx.work.impl.diagnostics.DiagnosticsReceiver	<table><tr><th>ID</th><th>Intent</th><th>Action</th></tr><tr><td>119</td><td>androidx.work.diagnostics.REQUEST_DIAGNOSTICS</td><td>action</td></tr></table>	ID	Intent	Action	119	androidx.work.diagnostics.REQUEST_DIAGNOSTICS	action									
ID	Intent	Action																
119	androidx.work.diagnostics.REQUEST_DIAGNOSTICS	action																
56	receiver	androidx.profileinstaller.ProfileInstallReceiver	<table><tr><th>ID</th><th>Intent</th><th>Action</th></tr><tr><td>120</td><td>androidx.profileinstaller.action.INSTALL_PROFILE</td><td>action</td></tr><tr><td>121</td><td>androidx.profileinstaller.action.SKIP_FILE</td><td>action</td></tr><tr><td>122</td><td>androidx.profileinstaller.action.SAVE_PROFILE</td><td>action</td></tr><tr><td>123</td><td>androidx.profileinstaller.action.BENCHMARK_OPERATION</td><td>action</td></tr></table>	ID	Intent	Action	120	androidx.profileinstaller.action.INSTALL_PROFILE	action	121	androidx.profileinstaller.action.SKIP_FILE	action	122	androidx.profileinstaller.action.SAVE_PROFILE	action	123	androidx.profileinstaller.action.BENCHMARK_OPERATION	action
ID	Intent	Action																
120	androidx.profileinstaller.action.INSTALL_PROFILE	action																
121	androidx.profileinstaller.action.SKIP_FILE	action																
122	androidx.profileinstaller.action.SAVE_PROFILE	action																
123	androidx.profileinstaller.action.BENCHMARK_OPERATION	action																
57	provider	org.f-droid.f-droid.installer.ApkFileProvider																
58	provider	androidx.core.content.FileProvider																
59	provider	org.f-droid.f-droid.nearby.PublicSourceDirProvider																
60	provider	org.acra.attachment.AkraContentProvider																

Showing 1 to 60 of 60 entries

Fuente: Elaboración propia.

Nota: La figura muestra la salida del módulo de Components tras el escaneo de la aplicación “F-Droid”, donde se enlistan los 60 elementos que conforman la estructura lógica de la aplicación. El análisis evidencia un conjunto variado de Activities (pantallas principales y secundarias), Services relacionados con instalación, actualización de repositorios, conectividad y manejo de descargas, así como múltiples Broadcast Receivers que permiten responder a eventos del sistema tales como conexión USB, estado de batería, almacenamiento y cambios de conectividad. También se identifican Content Providers encargados de exponer recursos de la aplicación a otros procesos, como el ApkFileProvider o AkraContentProvider. Además, se observa la configuración detallada de intent-filters con acciones, categorías, hosts y esquemas que habilitan interacciones específicas, como enlaces hacia repositorios F-Droid, acceso a repositorios externos, integración con Google Play y Amazon, así como soporte para búsquedas mediante la acción SEARCH.

9. Findings

El apartado Findings concentra los hallazgos derivados del análisis estático de seguridad realizado sobre el APK “F-Droid”. Esta sección agrupa todas las debilidades y patrones inseguros identificados en el código, clasificándolos por categoría y cuantificando cuántas veces aparecen dentro de la aplicación. En total, se detectaron 19.215 hallazgos, entre los que destacan la presencia de funciones inseguras, operaciones de almacenamiento, consultas SQL en crudo, direcciones IP y URLs codificadas de manera estática, así como registros de información sensible. La información recolectada en esta sección constituye un insumo fundamental para evaluar la exposición de la aplicación frente a vulnerabilidades reales que podrían ser explotadas en un entorno de ejecución.

Figura 14: Hallazgos de seguridad (Findings) del análisis del APK “F-Droid”

Findings

NEW FINDING

Number of findings: 19215

BULK EDITVIEW FINDINGSDELETE FINDINGS

Search:

ID	Finding	Number	Findings																																								
38	Exported Component	10	<div>Search:</div> <table><tr><th>ID</th><th>Severity</th><th>File</th><th>LN</th><th>Line</th><th>Status</th><th>CWE</th><th>Risk</th></tr><tr><td>1</td><td>High</td><td>/resources/AndroidManifest.xml</td><td>62</td><td>android:exported="true"</td><td>To Do</td><td>926</td><td>M</td></tr><tr><td>2</td><td>High</td><td>/resources/AndroidManifest.xml</td><td>70</td><td>android:exported="true"</td><td>To Do</td><td>926</td><td>M</td></tr><tr><td>3</td><td>High</td><td>/resources/AndroidManifest.xml</td><td>77</td><td>android:exported="true"</td><td>To Do</td><td>926</td><td>M</td></tr><tr><td>4</td><td>High</td><td>/resources/AndroidManifest.xml</td><td>112</td><td>android:exported="true"</td><td>To Do</td><td>926</td><td>M</td></tr></table>	ID	Severity	File	LN	Line	Status	CWE	Risk	1	High	/resources/AndroidManifest.xml	62	android:exported="true"	To Do	926	M	2	High	/resources/AndroidManifest.xml	70	android:exported="true"	To Do	926	M	3	High	/resources/AndroidManifest.xml	77	android:exported="true"	To Do	926	M	4	High	/resources/AndroidManifest.xml	112	android:exported="true"	To Do	926	M
ID	Severity	File	LN	Line	Status	CWE	Risk																																				
1	High	/resources/AndroidManifest.xml	62	android:exported="true"	To Do	926	M																																				
2	High	/resources/AndroidManifest.xml	70	android:exported="true"	To Do	926	M																																				
3	High	/resources/AndroidManifest.xml	77	android:exported="true"	To Do	926	M																																				
4	High	/resources/AndroidManifest.xml	112	android:exported="true"	To Do	926	M																																				

Fuente: Elaboración propia.

Nota: La figura presenta la salida detallada de la sección Findings, donde se contabilizan los 19.215 hallazgos de seguridad detectados durante el escaneo del APK “F-Droid”. Los resultados más relevantes incluyen 2.345 excepciones genéricas mal gestionadas, 1.345 solicitudes HTTP sin cifrado, 975 registros de información sensible en logs, 830 direcciones IP y 123 URLs codificadas en el código, además de 196 consultas SQL en

crudo y 54 credenciales embebidas directamente. Asimismo, se identificaron prácticas criptográficas débiles como el uso del modo ECB en algoritmos de cifrado (13 casos), funciones hash obsoletas (105 instancias) y generación insegura de números aleatorios (49 ocurrencias). También se evidenciaron categorías de riesgo en la gestión de componentes exportados, validación inadecuada de certificados y acceso a datos sensibles como GPS o portapapeles. Este análisis muestra que, aunque muchas de las vulnerabilidades se repiten en múltiples instancias, existen hallazgos críticos relacionados con el uso de información codificada y malas prácticas de seguridad que requieren atención prioritaria para mitigar posibles ataques y garantizar la robustez de la aplicación.

10. Security Best Practices

El análisis de Security Best Practices permite verificar qué prácticas recomendadas de seguridad han sido implementadas en la aplicación “F-Droid”. Esta sección es clave porque muestra hasta qué punto el desarrollador ha aplicado controles preventivos frente a vulnerabilidades comunes en Android, tales como la validación de certificados, el uso de criptografía, la protección contra el acceso no autorizado a componentes internos, la generación segura de números aleatorios y la desactivación de configuraciones inseguras como el modo debuggable o la copia de seguridad (backup). Los resultados reflejan que la aplicación incorpora varias buenas prácticas en seguridad, aunque presenta vacíos en otras áreas como detección de root, prevención de tapjacking y verificación avanzada frente a herramientas de manipulación como Frida.

Figura 15: Prácticas de seguridad implementadas en el análisis del APK “F-Droid”

Security Best Practices

Search:

Name	Description	Implementation																																																
Conection Verification/SSL Pinning	The application verifies the certificate with SSL Pinning	<div>Show 10 entries<div>Search: <input type="text"/></div></div> <table><tr><th>ID</th><th>Path</th><th>Line</th><th>Line</th></tr><tr><td>2248</td><td>sources/ch/qos/logback/core/net/ssl/SSLContextFactoryBean.java</td><td>15</td><td>TrustManager</td></tr><tr><td>2249</td><td>sources/ch/qos/logback/core/net/ssl/SSLContextFactoryBean.java</td><td>16</td><td>TrustManager</td></tr><tr><td>2250</td><td>sources/ch/qos/logback/core/net/ssl/SSLContextFactoryBean.java</td><td>26</td><td>TrustManager</td></tr><tr><td>2251</td><td>sources/ch/qos/logback/core/net/ssl/SSLContextFactoryBean.java</td><td>47</td><td>TrustManager</td></tr><tr><td>2252</td><td>sources/ch/qos/logback/core/net/ssl/SSLContextFactoryBean.java</td><td>47</td><td>TrustManager</td></tr></table> <table><tr><td>6984</td><td>sources/org/bouncycastle/cert/crmf/bc/BcFixedLengthMGF1Padder.java</td><td>3</td><td>java.security.SecureRandom</td></tr><tr><td>6989</td><td>sources/org/bouncycastle/cert/crmf/bc/CRMFHelper.java</td><td>3</td><td>java.security.SecureRandom</td></tr><tr><td>7003</td><td>sources/org/bouncycastle/cert/crmf/jcajce/CRMFHelper.java</td><td>15</td><td>java.security.SecureRandom</td></tr><tr><td>7027</td><td>sources/org/bouncycastle/cert/crmf/jcajce/JceCRMFEncryptorBuilder.java</td><td>7</td><td>java.security.SecureRandom</td></tr><tr><td>7127</td><td>sources/org/bouncycastle/cms/PasswordRecipientInfoGenerator.java</td><td>3</td><td>java.security.SecureRandom</td></tr><tr><td>7133</td><td>sources/org/bouncycastle/cms/bc/BcCMSContentEncryptorBuilder.java</td><td>5</td><td>java.security.SecureRandom</td></tr></table> <div>Showing 1 to 10 of 371 entries<div>Previous12345...38</div></div>	ID	Path	Line	Line	2248	sources/ch/qos/logback/core/net/ssl/SSLContextFactoryBean.java	15	TrustManager	2249	sources/ch/qos/logback/core/net/ssl/SSLContextFactoryBean.java	16	TrustManager	2250	sources/ch/qos/logback/core/net/ssl/SSLContextFactoryBean.java	26	TrustManager	2251	sources/ch/qos/logback/core/net/ssl/SSLContextFactoryBean.java	47	TrustManager	2252	sources/ch/qos/logback/core/net/ssl/SSLContextFactoryBean.java	47	TrustManager	6984	sources/org/bouncycastle/cert/crmf/bc/BcFixedLengthMGF1Padder.java	3	java.security.SecureRandom	6989	sources/org/bouncycastle/cert/crmf/bc/CRMFHelper.java	3	java.security.SecureRandom	7003	sources/org/bouncycastle/cert/crmf/jcajce/CRMFHelper.java	15	java.security.SecureRandom	7027	sources/org/bouncycastle/cert/crmf/jcajce/JceCRMFEncryptorBuilder.java	7	java.security.SecureRandom	7127	sources/org/bouncycastle/cms/PasswordRecipientInfoGenerator.java	3	java.security.SecureRandom	7133	sources/org/bouncycastle/cms/bc/BcCMSContentEncryptorBuilder.java	5	java.security.SecureRandom
ID	Path	Line	Line																																															
2248	sources/ch/qos/logback/core/net/ssl/SSLContextFactoryBean.java	15	TrustManager																																															
2249	sources/ch/qos/logback/core/net/ssl/SSLContextFactoryBean.java	16	TrustManager																																															
2250	sources/ch/qos/logback/core/net/ssl/SSLContextFactoryBean.java	26	TrustManager																																															
2251	sources/ch/qos/logback/core/net/ssl/SSLContextFactoryBean.java	47	TrustManager																																															
2252	sources/ch/qos/logback/core/net/ssl/SSLContextFactoryBean.java	47	TrustManager																																															
6984	sources/org/bouncycastle/cert/crmf/bc/BcFixedLengthMGF1Padder.java	3	java.security.SecureRandom																																															
6989	sources/org/bouncycastle/cert/crmf/bc/CRMFHelper.java	3	java.security.SecureRandom																																															
7003	sources/org/bouncycastle/cert/crmf/jcajce/CRMFHelper.java	15	java.security.SecureRandom																																															
7027	sources/org/bouncycastle/cert/crmf/jcajce/JceCRMFEncryptorBuilder.java	7	java.security.SecureRandom																																															
7127	sources/org/bouncycastle/cms/PasswordRecipientInfoGenerator.java	3	java.security.SecureRandom																																															
7133	sources/org/bouncycastle/cms/bc/BcCMSContentEncryptorBuilder.java	5	java.security.SecureRandom																																															

Showing 1 to 11 of 11 entries

Fuente: Elaboración propia.

Nota: La figura muestra los resultados obtenidos en el apartado Security Best Practices, donde se identificaron las medidas de seguridad aplicadas por la aplicación “F-Droid”. Entre los controles implementados se observa el uso de SSL Pinning para la validación de certificados, la aplicación de mecanismos criptográficos en distintas librerías, la inclusión de verificaciones para detectar entornos de depuración, la configuración de componentes no exportados en el AndroidManifest.xml y el empleo de generadores seguros de números aleatorios mediante la clase java.security.SecureRandom. También se confirma que la aplicación no está marcada como debuggable y que se deshabilita la opción de backup,

reduciendo así el riesgo de extracción de datos por medio de depuración USB. No obstante, el análisis revela la ausencia de controles en otras áreas, como la detección de dispositivos roteados, la protección frente a ataques de tapjacking, el bloqueo de manipulación con Frida y la falta de evidencia explícita del uso de HTTPS seguro en todas las comunicaciones.

11. Certificates

El apartado Certificates presenta la información sobre los certificados digitales utilizados para la firma del APK de la aplicación “F-Droid”. Este análisis es relevante porque la firma digital garantiza la autenticidad del software, la integridad de los archivos y la identidad del desarrollador responsable. La herramienta empleada identificó un único certificado asociado al APK, lo que confirma que la aplicación fue firmada por su desarrollador original. Los detalles del certificado permiten evaluar el algoritmo de hash, el método de firma, la versión soportada y la identidad declarada en el campo Subject. Este tipo de validaciones es esencial para comprobar si la aplicación mantiene una distribución confiable y libre de alteraciones externas.

Figura 16: Certificado digital del APK “F-Droid” identificado en el análisis

Certificates

Search:

ID	Version	Subject	Issuer	Hash algorithm	Signature algorithm	Serial number	Sha1	Sha256
1	v1, v2, v3	Common Name: Claran Gultnieks, Organizational Unit: Unknown, Organization: Unknown, Locality: Wetherby, State/Province: Unknown, Country: UK	Common Name: Claran Gultnieks, Organizational Unit: Unknown, Organization: Unknown, Locality: Wetherby, State/Province: Unknown, Country: UK	sha1	rsassa_pkcs1v15	1279905024	b'\x05\xf2\xe6V(\x08\x89\x81\x03\x17\xfc\x9am\xbf\xedK\x0f\xa1;N'	b'C#\x8dQ\x1e^(\xb2\x06V\x9fJ:\xfbl\x5R4\x18\x08 \n>\x0d1U'p\xab\x09\xa9\x09\xcc\xab'

Showing 1 to 1 of 1 entries

Fuente: Elaboración propia.

Nota: La figura presenta los resultados obtenidos en la sección Certificates, donde se detectó un único certificado digital con versiones v1, v2 y v3 activas. El Subject e Issuer corresponden a la misma identidad, declarada como Ciaran Gultnieks (Wetherby, UK), lo que indica que el certificado es autofirmado. El análisis confirma que la firma se realizó mediante el algoritmo de hash SHA-1 junto con el esquema de firma RSASSA-PKCS1-v1_5, acompañado de los identificadores únicos de serial, SHA1 y SHA256. La existencia de un certificado autofirmado es coherente con el modelo de distribución de F-Droid, donde los desarrolladores mantienen el control de la firma en lugar de depender de autoridades de certificación externas. Sin embargo, también implica que la confianza recae en la verificación del origen del APK, ya que no existe una entidad certificadora que respalde su validez.

12. Strings

El apartado Strings recopila todas las cadenas de texto detectadas en el análisis del APK de la aplicación F-Droid. Estas cadenas incluyen tanto fragmentos en formato hexadecimal como direcciones URL y otros valores de interés que pueden evidenciar rutas de comunicación, referencias internas o datos sensibles incrustados en el código. El análisis reveló un volumen significativo de entradas, lo que permite identificar componentes críticos como certificados en formato hexadecimal y múltiples enlaces oficiales y espejos del repositorio de F-Droid.

Figura 17: Cadenas de texto detectadas en el APK de la aplicación “F-Droid”

Strings

Show 10 entries

Search:

ID	Type	Value	Finding
1	hex	3082035e30820246a00302010202044c49cd0 0300d6092a864886f70d0101050500307131 0b300906035504061302554b3110300e0603 5504081307556e6b6e6f776e3111300f06035 50407130857657468657262793110300e060 355040a1307556e6b6e6f776e3110300e060 55040b1307556e6b6e6f776e3119301706035 50403131043696172616e2047756c746e696 56b73301e170d31303037323331373130323 45a170d3337313230383137313032345a307 1310b300906035504061302554b3110300e0 6035504081307556e6b6e6f776e3111300f06 03550407130857657468657262793110300e 060355040a1307556e6b6e6f776e3110300e0 060355040b1307556e6b6e6f776e3110300e0	46

5	URL	http://dotsrccccbldkzg7oc7oj4ugxrfbt64qebyunxbrqghxiwj3n6vccad.onion/fdroid/repo	50
6	URL	http://ftpfaudev4trw2vxiwzf4334e3mynz7osqgtozhbc77fxncqzbyoyd.onion/fdroid/repo	51
7	URL	http://lysator7eknrf47rlyxvgeamiv7ucefgrlrik7rouv3sna25asetwid.onion/pub/fdroid/repo	52
8	URL	http://mirror.ossplanetnyou5xif6liw5vhwzcg2fmmiohza25wwgnaw65ytfasad.onion/fdroid/repo	53
9	URL	https://fdroid.tetaneutral.net/fdroid/repo	54
10	URL	https://ftp.agdsn.de/fdroid/repo	55

Showing 1 to 10 of 1,745 entries

Previous 1 2 3 4 5 ... 175 Next

Fuente: Elaboración propia.

Nota: La figura muestra una parte de las 1,745 cadenas de texto encontradas en el APK de F-Droid. Entre los resultados más relevantes se incluyen valores en formato hexadecimal correspondientes a certificados digitales y diversas URLs que apuntan tanto al sitio oficial (<https://f-droid.org>) como a múltiples espejos y direcciones en la red Tor (.onion), lo cual confirma la existencia de rutas alternativas para la descarga segura del repositorio. Estos hallazgos evidencian que la aplicación integra mecanismos para asegurar disponibilidad y redundancia de acceso, reforzando la confianza en la distribución del software al permitir verificar fuentes legítimas desde diferentes canales.

13. Files

El apartado Files muestra los archivos extraídos del paquete APK de la aplicación F-Droid. Estos archivos son fundamentales para el funcionamiento interno de la aplicación, ya que incluyen tanto elementos esenciales como el AndroidManifest.xml, las clases compiladas en DEX, configuraciones de propiedades, metadatos de Kotlin, así como recursos gráficos y plantillas HTML que conforman la interfaz y funcionalidades de la app. En total se identificaron 399 archivos, lo que refleja la complejidad y estructura modular de la aplicación.

Figura 18: Archivos extraídos del APK de la aplicación “F-Droid”

Files

Show 10 entries Search:

ID	Path	Type
1	/resources/AndroidManifest.xml	xml
2	/resources/classes.dex	other
3	/resources/classes2.dex	other
4	/resources/DebugProbesKt.bin	other
5	/resources/kotlin-tooling-metadata.json	other
6	/resources/version.properties	properties
7	/resources/assets/index.template.html	html
8	/resources/assets/swap-icon.png	image
9	/resources/assets/swap-icon.svg	image
10	/resources/assets/swap-tick-done.png	image

Showing 1 to 10 of 399 entries Previous 1 2 3 4 5 ... 40 Next

Fuente: Elaboración propia.

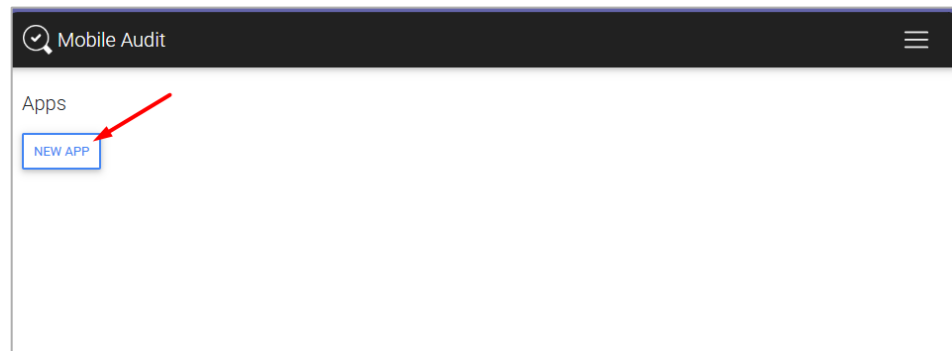
Nota: La figura presenta una muestra de los 399 archivos identificados en el análisis de F-Droid, donde se destacan componentes clave como el AndroidManifest.xml, que define permisos y configuraciones de la aplicación; los ficheros classes.dex, que contienen el código ejecutable; así como recursos adicionales en formato HTML, imágenes y propiedades. Estos resultados evidencian la organización interna del APK, permitiendo observar cómo la aplicación gestiona tanto la lógica de ejecución como los elementos gráficos y de configuración, lo que facilita un análisis forense y de seguridad más detallado de cada uno de sus módulos.

2.5. Ejecución del análisis de un APK: DailyTube - media player

Se analizará el APK DailyTube - media player, por lo que se ejecutarán los siguientes pasos:

1. Desde la interfaz, se selecciona “NEW APP” para cargar un archivo APK.

Figura 19: Pantalla inicial de Mobile Audit con opción “New App”



Fuente: Elaboración propia.

Nota: La figura muestra la interfaz principal de Mobile Audit accesible en <http://localhost:8888/>, con la barra de navegación superior y el botón New App disponible en la sección Apps; esta vista confirma que la plataforma está operativa y lista para registrar una nueva aplicación, paso imprescindible para iniciar el proceso de auditoría sobre el APK DailyTube - media player.

2. Carga el APK “DailyTube - media player” e inicio del escaneo

En el formulario New App, completa:

- Name: DailyTube (o el identificador que prefieras).
- APK file: Selecciona el archivo DailyTube.apk desde tu equipo.
(Opcional) Description/Tags para clasificar el análisis.

Pulsa Create o Upload para enviar el APK.

El sistema encola la tarea al worker (Celery) y comienza:

- Análisis estático (SAST) con el motor de patrones/reglas.
- Análisis de malware (URLs maliciosas y firmas en strings; VirusTotal si está habilitado).

Figura 20: Formulario de carga de APK “DailyTube - media player”

The screenshot shows a web browser window with the address bar displaying 'localhost:8888/app/create'. The browser's bookmark bar includes 'Aula Virtual', 'Intranet', 'Mesa', 'Pagos', 'Idiomas', 'EPIS', and 'Otros favoritos'. The page title is 'Mobile Audit'. The form contains the following elements:

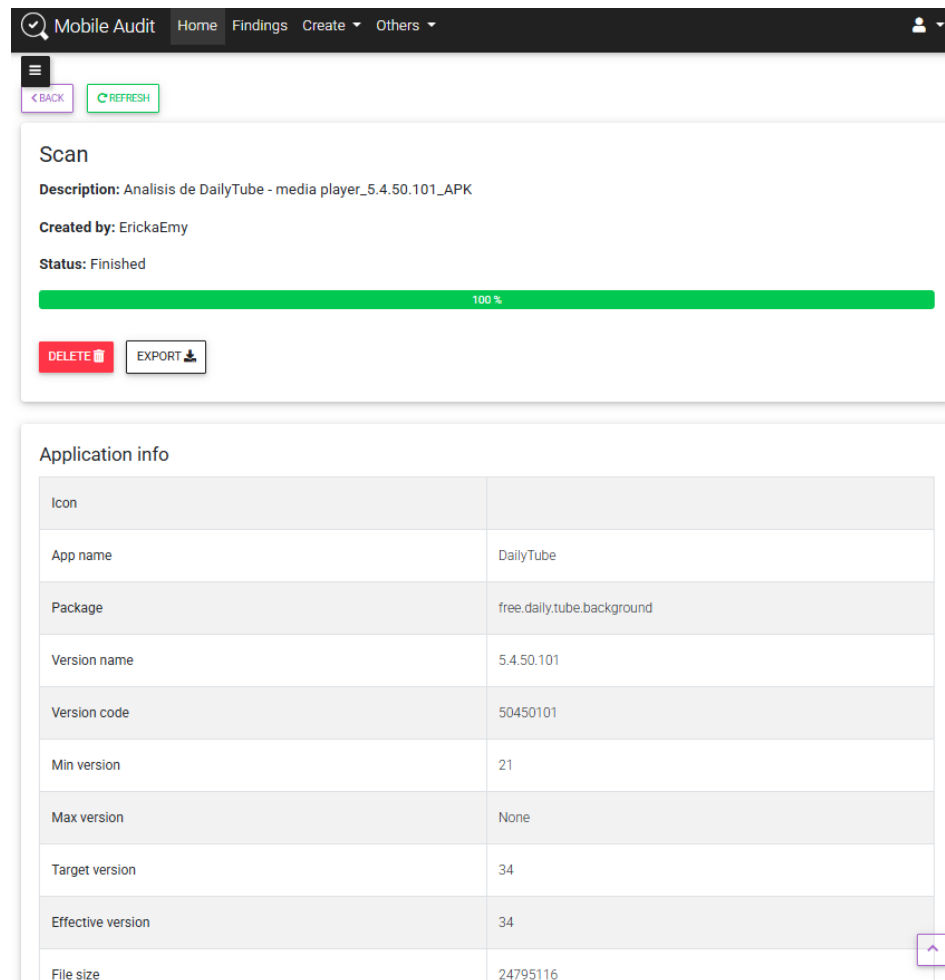
- A purple 'BACK' button.
- A 'Name' field with the value 'DailyTube - media player'.
- A 'Description' field with the value 'Análisis de DailyTube - media player_5.4.50.101_APK'.
- A blue 'CREATE' button.
- A 'Description' field (repeated) with the value 'Análisis de DailyTube - media player_5.4.50.101_APK'.
- An 'Apk' field with a file selection button labeled 'Elegir archivo' and the filename 'DailyTube - media player_5.4.50.101_APKPure.apk'.
- An 'App' dropdown menu showing 'Application #3 - DailyTube - media player'.
- A green 'UPLOAD' button.

Fuente: Elaboración propia.

Nota: La figura evidencia el registro de la aplicación DailyTube - media player y la selección del archivo DailyTube.apk para su análisis; al enviar el formulario, el backend Django recibe el artefacto, lo almacena y delega su procesamiento al worker Celery a través de RabbitMQ, dejando el estado del escaneo en curso y habilitando la posterior visualización de resultados.

3. Información de la aplicación

- Ruta en la UI: dentro del escaneo → Application info / App info.
- Qué validar/explicar: nombre de paquete (package), versión (versionName/versionCode), minSdkVersion, targetSdkVersion, permisos declarados, y metadatos del AndroidManifest.xml.

Figura 21: Acceso a la interfaz web de Mobile Audit en el navegador

Fuente: Elaboración propia.

Nota: La figura muestra la ficha técnica extraída del APK DailyTube - media player, evidenciando información clave como el nombre de la aplicación, el paquete (free.daily.tube.background), la versión (5.4.50.101 / 50450101), la versión mínima de Android soportada (21), la versión objetivo (34), el tamaño del archivo (24,795,116 bytes) y los hashes de integridad md5 y sha256; esta visualización permite verificar que el APK ha sido correctamente cargado y procesado en la plataforma Mobile Audit.

4. Security Info

La sección Security info muestra un resumen del análisis de seguridad realizado sobre el APK DailyTube - media player, identificando vulnerabilidades y riesgos potenciales en la aplicación. El escaneo automático detectó un total de 9,090 hallazgos, clasificados por su severidad, lo que permite evaluar la criticidad de cada incidencia y priorizar acciones correctivas para mejorar la seguridad del binario.

Figura 22: Resultados de seguridad del APK DailyTube - media player en Mobile Audit

Security info

Number of findings	9090
By Severity	
	Critical 6
	High 937
	Medium 2001
	Low 5707
	None 439

Fuente: Elaboración propia.

Nota: La figura muestra el panel completo de Security info del APK DailyTube - media player, evidenciando un total de 9,090 hallazgos de seguridad distribuidos por severidad: 6 críticos, 937 altos, 2,001 medios, 5,707 bajos y 439 sin severidad asignada; esta visualización permite interpretar de manera estructurada el estado de seguridad de la aplicación, identificando tanto las incidencias que requieren atención inmediata como aquellas de menor impacto, reflejando que el análisis automatizado de Mobile Audit procesó exhaustivamente el binario combinando detección de vulnerabilidades, patrones de riesgo y posibles puntos débiles en el código, dejando los resultados listos para su revisión, priorización de mitigaciones y comprensión del nivel de seguridad general de la aplicación.

5. Permissions

La sección Permissions del APK DailyTube - media player muestra los permisos que la aplicación solicita para funcionar correctamente en el dispositivo, clasificándolos por tipo y severidad. Este análisis permite identificar tanto los permisos necesarios para funcionalidades básicas como aquellos que representan riesgos de seguridad o privacidad, facilitando la evaluación de la exposición del binario a posibles vulnerabilidades.

Figura 23: Permisos del APK DailyTube - media player en Mobile Audit

Permissions

Search:

ID	Name	Type	Severity	Status
1	android.permission.SYSTEM_ALERT_WINDOW	Special	High	❌
2	android.permission.ACCESS_WIFI_STATE	Normal	Low	✅
3	android.permission.WAKE_LOCK	Normal	Medium	⚠️
4	android.permission.READ_MEDIA_AUDIO	Other	High	❌
5	android.permission.ACCESS_AD_SERVICES_AD_ID	Other	High	❌
6	android.permission.READ_MEDIA_VIDEO	Other	High	❌
15	android.permission.ACCESS_AD_SERVICES_TOPICS	Other	High	❌
16	android.permission.ACCESS_NETWORK_STATE	Normal	Low	✅
17	com.google.android.gms.permission.AD_ID	Other	High	❌
18	android.permission.FOREGROUND_SERVICE_MEDIA_PLAYBACK	Other	High	❌
19	android.permission.FOREGROUND_SERVICE	Normal	Medium	⚠️
20	free.daily.tube.background.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION	Other	High	❌
21	android.permission.WRITE_EXTERNAL_STORAGE	Dangerous	High	❌
22	android.permission.VIBRATE	Normal	Low	✅
23	android.permission.POST_NOTIFICATIONS	Other	High	❌

Showing 1 to 23 of 23 entries

Fuente: Elaboración propia.

Nota: La figura presenta la lista completa de permisos solicitados por el APK DailyTube - media player, incluyendo 23 permisos con distintos niveles de severidad, desde críticos y peligrosos como SYSTEM_ALERT_WINDOW, READ_EXTERNAL_STORAGE y WRITE_EXTERNAL_STORAGE, hasta permisos de severidad media y baja relacionados con conectividad, notificaciones y reproducción de medios, además de permisos asociados a servicios de publicidad y seguimiento de usuario; esta vista permite evaluar la superficie de ataque de la aplicación y comprender los recursos del dispositivo a los que puede acceder, evidenciando los riesgos de seguridad y privacidad potenciales asociados a cada permiso, reflejando que Mobile Audit procesó exhaustivamente el APK para extraer, clasificar y priorizar los permisos de acuerdo con su criticidad, dejando los resultados listos para su revisión y análisis de mitigación.

6. Activities

La sección Activities identifica todas las actividades presentes en el APK DailyTube - media player, evaluando cuáles están correctamente configuradas como principales y cuáles presentan problemas de inicialización. Este análisis permite comprender la estructura interna de la aplicación y su flujo de navegación, destacando las actividades críticas para el arranque y la interacción del usuario.

Figura 24: Listado de Activities detectadas en el APK “F-Droid”

Activities

Search:

ID	Name	Main
26	free.daily.tube.background.main.MainActivity	✓
27	free.daily.tube.background.crash_proxy.DTCPActivity	✗
28	free.daily.tube.background.dtoapp.re_captcha.DTRCAActivity	✗
29	free.daily.tube.background.dtoapp.DTRActivity	✗
30	free.daily.tube.background.dtoapp.player.background.guide_dialog.DTLRSActivity	✗

31	free.daily.tube.module.me_impl.me.DTMEActivity	✖
32	free.daily.tube.module.me_impl.policy.DTPWActivity	✖
33	free.daily.tube.module.fans_zone_impl.page.DTFZActivity	✖
34	free.daily.tube.module.settings_impl.DTSTActivity	✖
35	free.daily.tube.ad.adbusiness.test.DTADTActivity	✖
36	com.facebook.ads.AudienceNetworkActivity	✖
37	free.daily.tube.module.account_impl.page.account.DTACActivity	✖

Fuente: Elaboración propia.

Nota: La figura presenta la lista completa de actividades del APK DailyTube - media player, indicando que de las 65 actividades identificadas, únicamente la MainActivity (free.daily.tube.background.main.MainActivity) se encuentra correctamente configurada como principal, mientras que el resto presenta errores de inicialización señalados con un símbolo de equis; esta visualización permite comprender de manera detallada la arquitectura interna de la aplicación, incluyendo actividades relacionadas con la interfaz de usuario, reproducción de medios, servicios de publicidad y módulos auxiliares, evidenciando que Mobile Audit ha procesado exhaustivamente el APK para extraer, clasificar y verificar la configuración de cada actividad, lo que facilita la evaluación de estabilidad, seguridad y flujo funcional de la aplicación en tiempo de ejecución.

7. Components

La sección Components analiza y detalla los distintos elementos constitutivos del APK DailyTube - media player, incluyendo actividades, servicios, receptores y proveedores declarados en el manifiesto. Este análisis permite evaluar la arquitectura interna del binario, identificar los puntos de interacción del usuario y del sistema, y determinar cómo se gestionan los flujos de datos y las funcionalidades internas de la aplicación.

Figura 25: Componentes detectados en el análisis del APK “F-Droid”

Components

Search:

ID	Type	Name	Intents												
61	activity	free.daily.tube.background.main.MainActivity	<table><tr><th>ID</th><th>Intent</th><th>Action</th></tr><tr><td>124</td><td>android.intent.action.MAIN</td><td>action</td></tr><tr><td>125</td><td>android.intent.category.LAUNCHER</td><td>category</td></tr><tr><td>126</td><td>android.intent.category.LEANBACK_LAUNCHER</td><td>category</td></tr></table>	ID	Intent	Action	124	android.intent.action.MAIN	action	125	android.intent.category.LAUNCHER	category	126	android.intent.category.LEANBACK_LAUNCHER	category
ID	Intent	Action													
124	android.intent.action.MAIN	action													
125	android.intent.category.LAUNCHER	category													
126	android.intent.category.LEANBACK_LAUNCHER	category													
62	activity	free.daily.tube.background.crash_proxy.DTCAActivity													

160	provider	com.facebook.FacebookContentProvider	
161	provider	free.daily.tube.ad.ad_one.sdk.ui.DTOADFileProvider	
162	provider	com.flatads.sdk.util.FlatFileProvider2	
163	provider	com.huawei.openalliance.ad.ppskit.provider.InnerApiProvider2	
164	provider	com.huawei.openalliance.ad.ppskit.provider.PPSInstallFileProvider2	
165	provider	com.vungle.warren.utility.VungleProvider2	
166	provider	com.yandex.mobile.ads.core.initializer.MobileAdsInitializeProvider2	
167	provider	com.google.android.gms.ads.MobileAdsInitProvider2	
168	provider	free.daily.tube.ad.ad_sdk.proxy.DTAHPProvider	
169	provider	leakcanary.internal.PlumberInstaller	
170	provider	com.squareup.picasso.PicassoProvider	
171	provider	androidx.startup.InitializationProvider	
172	provider	androidx.lifecycle.ProcessLifecycleOwnerInitializer	

Showing 1 to 112 of 112 entries

Showing 1 to 112 of 112 entries

Fuente: Elaboración propia.

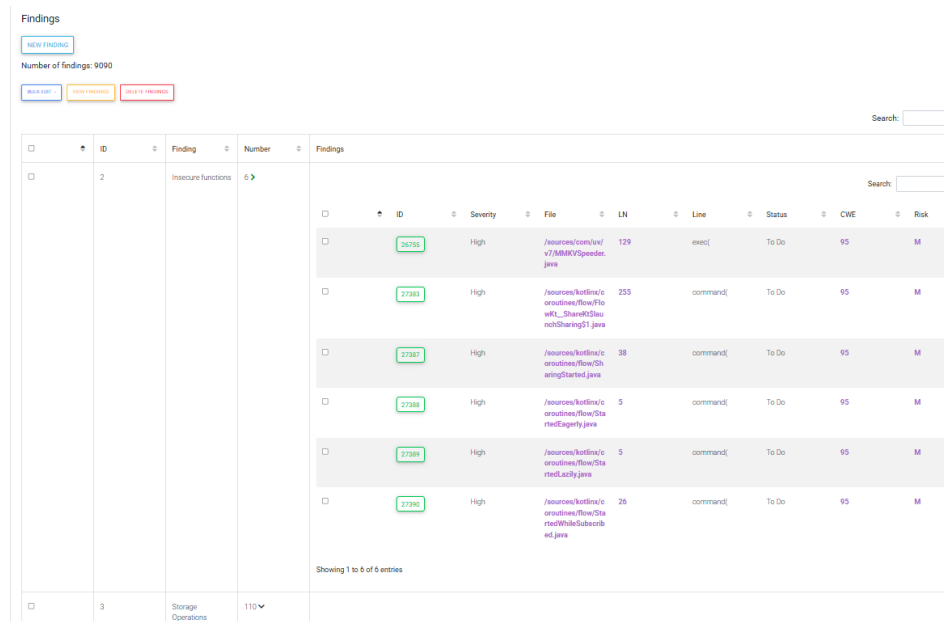
Nota: La figura muestra la lista completa de 112 componentes detectados en el APK DailyTube - media player, incluyendo actividades principales y secundarias, servicios de reproducción, mensajería y publicidad, receptores de eventos del sistema y proveedores de contenido; se observan intents asociados a acciones y categorías como MAIN, LAUNCHER, VIEW, MEDIA_BUTTON, así como esquemas de URI para navegación interna y externa, evidenciando la conectividad entre componentes y la capacidad de la aplicación para interactuar con el sistema operativo y servicios de terceros; esta información permite evaluar la estructura funcional del binario, la exposición de servicios críticos, la gestión de intents y la seguridad de los flujos de información, demostrando que Mobile Audit procesó exhaustivamente cada

componente y sus intents para ofrecer un mapa detallado de la arquitectura interna del APK, facilitando la identificación de posibles riesgos de seguridad y puntos de integración de alto impacto en tiempo de ejecución.

8. Findings

La sección Findings presenta los hallazgos de seguridad detectados en el APK DailyTube - media player durante el análisis de Mobile Audit, clasificando vulnerabilidades, prácticas inseguras y exposición de información sensible. Este análisis permite identificar riesgos potenciales, evaluar la seguridad del código y determinar áreas críticas que requieren atención prioritaria.

Figura 26: Hallazgos de seguridad del APK DailyTube - media player en Mobile Audit



The screenshot displays the 'Findings' section of the Mobile Audit tool. It shows a summary of 9,090 findings, categorized by severity: High (9,090), Medium (0), and Low (0). Below this, a detailed table lists specific findings. The table has columns for ID, Severity, File, Line, Status, CWE, and Risk. The findings listed are all 'High' severity and 'To Do' status, related to insecure functions and storage operations.

ID	Severity	File	Line	Status	CWE	Risk
26758	High	/sources/com/v7/MMKV/Speeder.java	129	To Do	95	M
27383	High	/sources/kotlin/coroutines/flow/FlowKt_ShareKtLazyStart.java	255	To Do	95	M
27382	High	/sources/kotlin/coroutines/flow/FlowKt_ShareKtLazyStart.java	38	To Do	95	M
27384	High	/sources/kotlin/coroutines/flow/FlowKt_ShareKtLazyStart.java	5	To Do	95	M
27385	High	/sources/kotlin/coroutines/flow/FlowKt_ShareKtLazyStart.java	5	To Do	95	M
27390	High	/sources/kotlin/coroutines/flow/FlowKt_ShareKtLazyStart.java	28	To Do	95	M

Fuente: Elaboración propia.

Nota: La figura muestra un total de 9,090 hallazgos detectados en el APK DailyTube - media player, categorizados según el tipo de riesgo, incluyendo funciones inseguras, operaciones de almacenamiento, obtención de información del dispositivo y de red, uso de GPS, manejo de credenciales y datos sensibles hardcodedos, web views, registro de

información sensible, consultas SQL sin protección, solicitudes HTTP inseguras, generación de números aleatorios inseguros, decodificación de datos hexadecimales, lectura del portapapeles, algoritmos de hash débiles, notificaciones, descarga de archivos, excepciones genéricas y componentes exportados; también se identificaron claves y credenciales hardcodeadas de Google API y Google Cloud Platform; esta información detalla minuciosamente cada vulnerabilidad y exposición.

9. Security Best Practices

La sección Security Best Practices analiza las implementaciones de seguridad adoptadas en el APK DailyTube - media player, enfocándose en mecanismos de protección frente a ataques, exposición de datos y explotación de vulnerabilidades. Este análisis permite evaluar si la aplicación sigue las mejores prácticas de seguridad móvil, incluyendo verificación de conexiones, manejo seguro de datos, protección contra depuración y accesos no autorizados.

Figura 27: Implementación de buenas prácticas de seguridad en DailyTube - media player

Security Best Practices

Name	Description	Implementation																																												
Connection Verification/SSL Pinning	The application verifies the certificate with SSL Pinning	<div>Search: <input type="text"/></div> <div>Show 10 entries</div> <table><thead><tr><th>ID</th><th>Path</th><th>LN</th><th>Line</th></tr></thead><tbody><tr><td>24985</td><td>/sources/com.facebook.ads/adidgen/SC3824N4.java</td><td>227</td><td>onFacebookSSL</td></tr><tr><td>24982</td><td>/sources/com.facebook.ads/adidgen/SC3824N4.java</td><td>144</td><td>onFacebookSSL</td></tr><tr><td>24983</td><td>/sources/com.facebook.ads/adidgen/SC3824N4.java</td><td>145</td><td>onFacebookSSL</td></tr><tr><td>27146</td><td>/sources/lec1/m.java</td><td>10</td><td>HostnameVerifier</td></tr><tr><td>27147</td><td>/sources/lec1/m.java</td><td>12</td><td>TrustManager</td></tr><tr><td>27148</td><td>/sources/lec1/m.java</td><td>40</td><td>TrustManager</td></tr><tr><td>27149</td><td>/sources/lec1/m.java</td><td>64</td><td>HostnameVerifier</td></tr><tr><td>27150</td><td>/sources/lec1/m.java</td><td>188</td><td>TrustManager</td></tr><tr><td>27151</td><td>/sources/lec1/m.java</td><td>196</td><td>TrustManager</td></tr><tr><td>27152</td><td>/sources/lec1/m.java</td><td>276</td><td>HostnameVerifier</td></tr></tbody></table> <div>Showing 1 to 10 of 165 entries</div> <div>Previous 1 2 3 4 5 ... 17 Next</div>	ID	Path	LN	Line	24985	/sources/com.facebook.ads/adidgen/SC3824N4.java	227	onFacebookSSL	24982	/sources/com.facebook.ads/adidgen/SC3824N4.java	144	onFacebookSSL	24983	/sources/com.facebook.ads/adidgen/SC3824N4.java	145	onFacebookSSL	27146	/sources/lec1/m.java	10	HostnameVerifier	27147	/sources/lec1/m.java	12	TrustManager	27148	/sources/lec1/m.java	40	TrustManager	27149	/sources/lec1/m.java	64	HostnameVerifier	27150	/sources/lec1/m.java	188	TrustManager	27151	/sources/lec1/m.java	196	TrustManager	27152	/sources/lec1/m.java	276	HostnameVerifier
ID	Path	LN	Line																																											
24985	/sources/com.facebook.ads/adidgen/SC3824N4.java	227	onFacebookSSL																																											
24982	/sources/com.facebook.ads/adidgen/SC3824N4.java	144	onFacebookSSL																																											
24983	/sources/com.facebook.ads/adidgen/SC3824N4.java	145	onFacebookSSL																																											
27146	/sources/lec1/m.java	10	HostnameVerifier																																											
27147	/sources/lec1/m.java	12	TrustManager																																											
27148	/sources/lec1/m.java	40	TrustManager																																											
27149	/sources/lec1/m.java	64	HostnameVerifier																																											
27150	/sources/lec1/m.java	188	TrustManager																																											
27151	/sources/lec1/m.java	196	TrustManager																																											
27152	/sources/lec1/m.java	276	HostnameVerifier																																											
Cryptography	The application is using cryptography	<div>Search: <input type="text"/></div> <div>Show 10 entries</div> <table><thead><tr><th>ID</th><th>Path</th><th>LN</th><th>Line</th></tr></thead><tbody><tr><td>24248</td><td>/sources/android/support/v4/media/session/MediaControllerCompat.java</td><td>19</td><td>Key</td></tr></tbody></table>	ID	Path	LN	Line	24248	/sources/android/support/v4/media/session/MediaControllerCompat.java	19	Key																																				
ID	Path	LN	Line																																											
24248	/sources/android/support/v4/media/session/MediaControllerCompat.java	19	Key																																											

Fuente: Elaboración propia.

Nota: La figura muestra los mecanismos de seguridad implementados en DailyTube - media player, detallando prácticas como la verificación de certificados mediante SSL Pinning, uso de criptografía para proteger información sensible, detección de depuración de dispositivos (Debugger detection) y verificación de root para prevenir ataques desde dispositivos comprometidos. Además, se observa que la aplicación desactiva el respaldo de datos (Prevent Backup components) y no se marca como depurable (Prevent debuggable components), asegurando que un atacante no pueda ejecutar código arbitrario ni acceder a datos sin privilegios elevados. Los componentes exportados están restringidos explícitamente (Prevent exported components) y se asegura la comunicación cifrada mediante HTTPS/TLS, así como la generación de números aleatorios con SecureRandom para operaciones críticas.

10. Certificates

La sección Certificates analiza los certificados digitales asociados al APK DailyTube - media player, los cuales permiten verificar la autenticidad del desarrollador y garantizar la integridad de la aplicación. Este análisis confirma que la firma del APK no ha sido manipulada y que las comunicaciones y actualizaciones se realizan desde una fuente confiable.

Figura 28: Certificados digitales y verificación de integridad en DailyTube - media player

Certificates

Search:

ID	Version	Subject	Issuer	Hash algorithm	Signature algorithm	Serial number	Sha1	Sha256
2	v1, v2, v3	Common Name: dailytube, Organizational Unit: dailytube, Organization: dailytube, Locality: dailytube, State/Province: dailytube, Country: dailytube	Common Name: dailytube, Organizational Unit: dailytube, Organization: dailytube, Locality: dailytube, State/Province: dailytube, Country: dailytube	sha256	rsassa_pkcs1v15	374061043	b'\x94\x1d\t\x87 \xa2g3m\wfb%\ xcd\x9c\xedf\x 031v98\x9a\x17 \x8c'	b'\xdc\x11\xd76 \xb4\xfc7(\xef\x b5Auz)\x19\xa5 \xe7\xaa- \xf1v86*\By\xcf fxd8\vd5\vb3\x 864+

Showing 1 to 1 of 1 entries

Fuente: Elaboración propia.

Nota: La figura muestra que DailyTube - media player utiliza un único certificado con versiones v1, v2 y v3, emitido por sí mismo, donde el sujeto y el emisor coinciden en todos los campos (Common Name, Organizational Unit, Organization, Locality, State/Province y Country), lo que indica que es un certificado auto-firmado. Se emplea el algoritmo de hash SHA-256 junto con RSASSA-PKCS1v15 para la firma digital, asegurando integridad y autenticidad del APK. Los valores de SHA-1 y SHA-256 proporcionan huellas únicas del certificado, permitiendo verificar que el archivo no ha sido alterado desde su firma.

11. Strings

La sección Strings analiza las cadenas de texto contenidas en el APK, incluyendo URLs, direcciones IP y otros valores codificados, lo que permite identificar posibles conexiones externas, recursos utilizados y referencias internas del aplicativo. Este análisis es útil para evaluar riesgos de seguridad y dependencias externas que la aplicación podría utilizar durante su ejecución.

Figura 29: Cadenas de texto y referencias externas en DailyTube - media player

Strings

Show entries

Search:

ID	Type	Value	Finding
1751	IP	5.4.50.101	19236
1752	IP	3.4.28.313	19237
1753	IP	3.4.28.313	19238
1754	URL	http://cdn.link.net/154335.jpg	19308
1755	URL	http://purl.org/rss/1.0/modules/content/	19309
1756	URL	http://wellformedweb.org/CommentAPI/	19310
1757	URL	http://www.w3.org/2005/Atom	19311
1758	URL	https://www.bola.net/	19312
1759	URL	https://www.bola.net/	19313
1760	URL	https://cdns.klimg.com/bola.net/library/logo.jpg	19314

Showing 1 to 10 of 4,967 entries

Previous [1](#) [2](#) [3](#) [4](#) [5](#) ... [497](#) Next

Fuente: Elaboración propia.

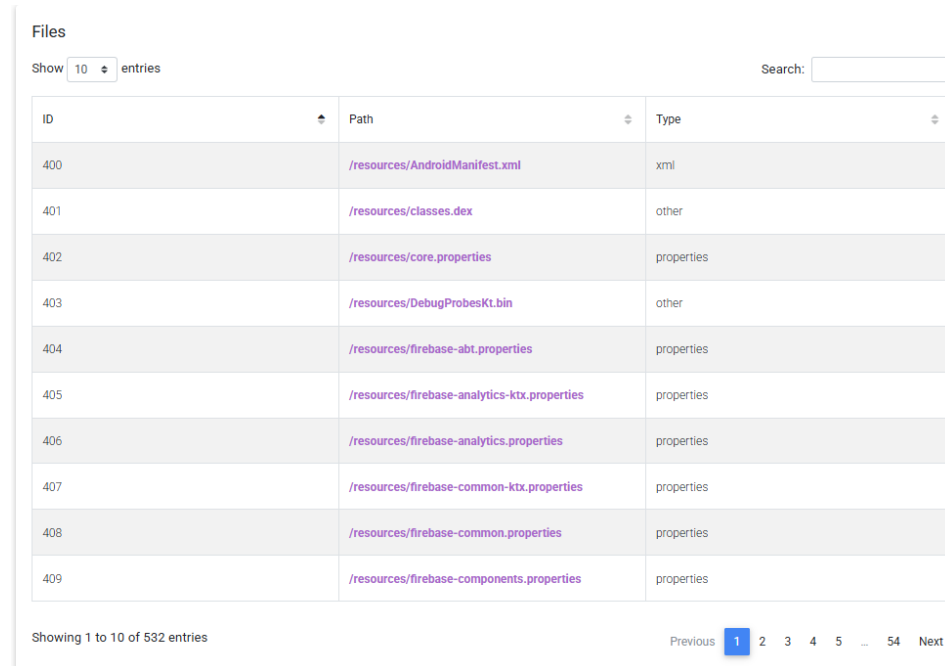
Nota: La figura muestra un extracto representativo de las 4,967 cadenas identificadas en el APK, incluyendo direcciones IP como 5.4.50.101 y 3.4.28.313, así como URLs HTTP y HTTPS de diversos servicios y recursos multimedia, por ejemplo <http://cdn.link.net/154335.jpg> y <https://www.bola.net/>. Estas cadenas permiten mapear los posibles endpoints y servicios externos que la aplicación puede consultar, además de identificar recursos estáticos integrados en el APK. El análisis de strings facilita la detección de información sensible expuesta, referencias a servicios web, APIs externas y rutas internas, contribuyendo a una evaluación más completa de la seguridad y comportamiento de la aplicación en tiempo de ejecución.

12. Files

La sección Files identifica y detalla los archivos contenidos en el APK, incluyendo manifiestos, librerías, archivos de configuración y

componentes propios del sistema de análisis. Este examen permite conocer la estructura interna de la aplicación, los recursos integrados y las configuraciones utilizadas, facilitando la detección de posibles riesgos o dependencias externas.

Figura 30: Estructura de archivos internos en DailyTube - media player



ID	Path	Type
400	/resources/AndroidManifest.xml	xml
401	/resources/classes.dex	other
402	/resources/core.properties	properties
403	/resources/DebugProbesKt.bin	other
404	/resources/firebase-abt.properties	properties
405	/resources/firebase-analytics-ktx.properties	properties
406	/resources/firebase-analytics.properties	properties
407	/resources/firebase-common-ktx.properties	properties
408	/resources/firebase-common.properties	properties
409	/resources/firebase-components.properties	properties

Fuente: Elaboración propia.

Nota: La figura presenta un extracto de los 532 archivos encontrados en el APK, mostrando ejemplos representativos como AndroidManifest.xml, que define permisos y componentes de la aplicación; classes.dex, que contiene el código ejecutable de Android; y varios archivos de propiedades relacionados con Firebase, como firebase-analytics.properties y firebase-common.properties, que configuran servicios de analítica y mensajería. Este análisis permite mapear los recursos internos de la aplicación, evaluar la presencia de configuraciones sensibles y comprender cómo se integran los distintos módulos y servicios en la estructura del APK, contribuyendo a un estudio técnico exhaustivo de su comportamiento y seguridad.

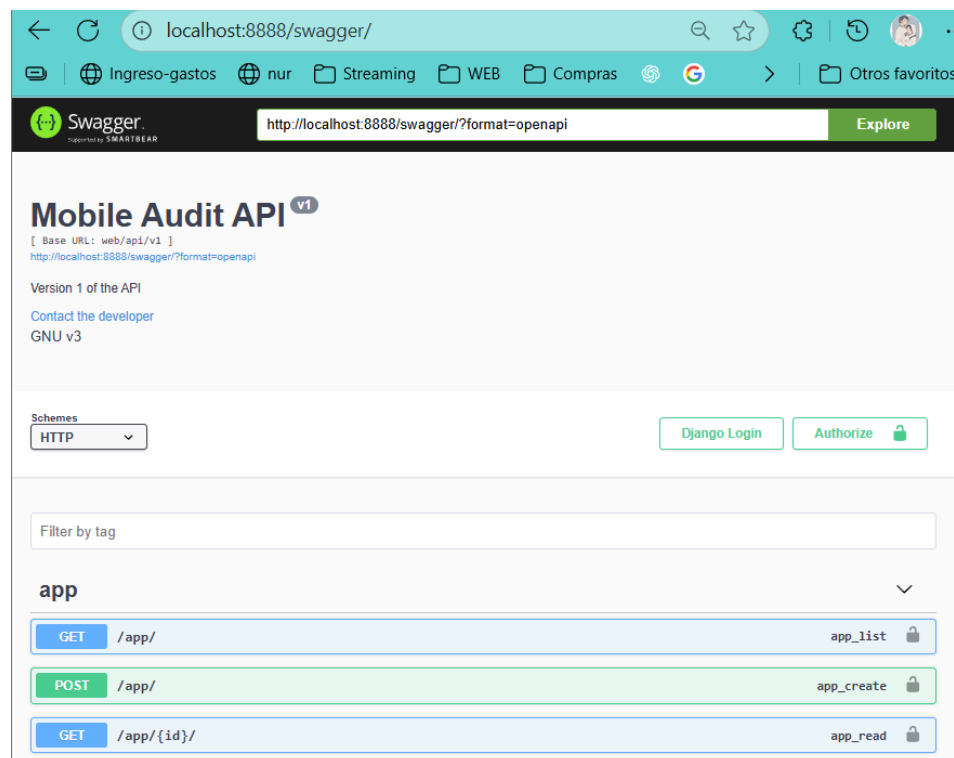
2.6. Integración en la API Swagger y Redoc

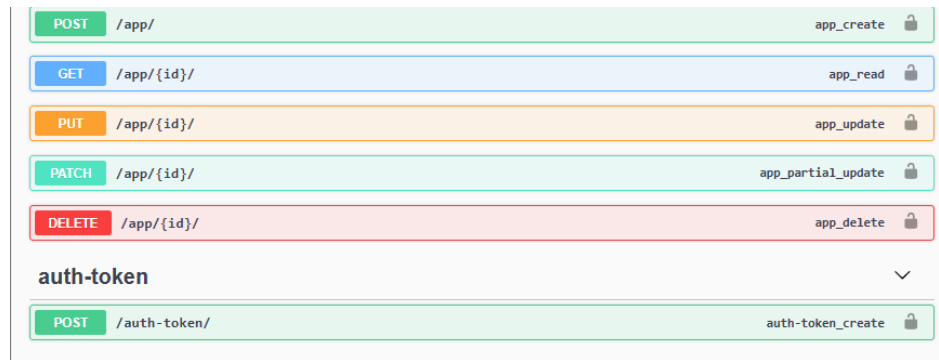
Swagger es una interfaz de documentación interactiva para APIs REST, que permite al usuario no solo visualizar la estructura de los endpoints expuestos, sino también ejecutar peticiones directamente desde el navegador.

En este laboratorio, al acceder a la URL <http://localhost:8888/swagger/>, se despliega una representación dinámica de la API v1, en donde cada endpoint se encuentra documentado con su respectiva descripción, parámetros, tipos de datos y respuestas esperadas.

Lo que caracteriza a Swagger frente a otras herramientas es su interactividad, ya que posibilita probar en tiempo real los métodos disponibles (GET, POST, PUT, DELETE, etc.) sin necesidad de recurrir a clientes externos como Postman. Esto convierte a Swagger en una herramienta idónea tanto para desarrolladores, al momento de validar la correcta implementación de los servicios, como para testers, al facilitar la ejecución y verificación de pruebas directamente desde la interfaz.

Figura 31: Visualización de la API v1 mediante Swagger UI

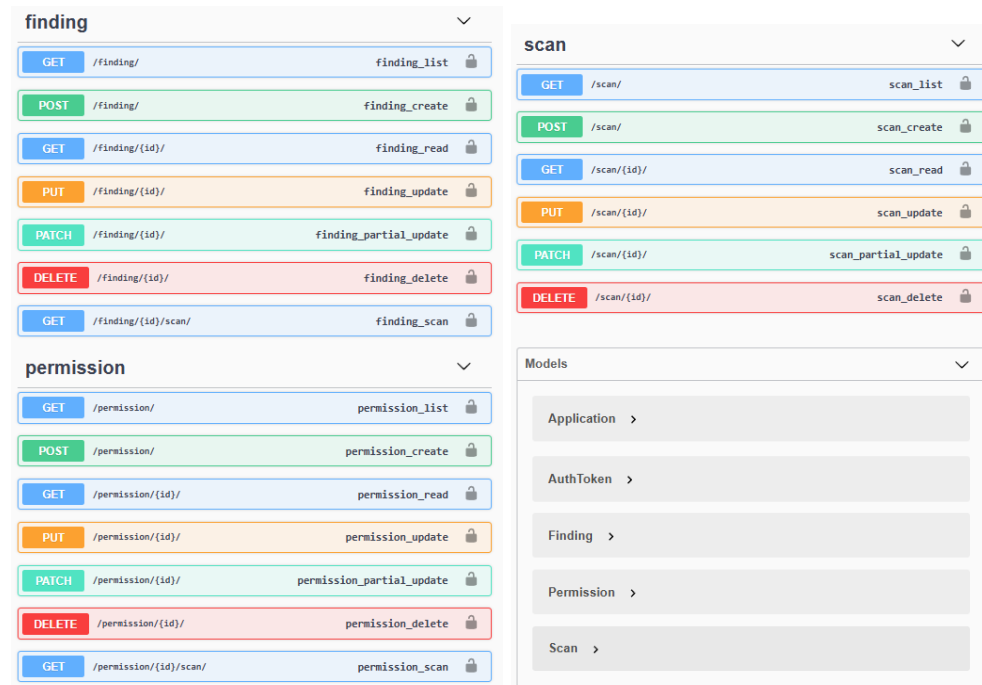




Fuente: Elaboración propia.

Nota: La figura muestra la interfaz de Swagger UI correspondiente a la documentación de la API v1 del sistema Mobile Audit, accesible desde la URL <http://localhost:8888/swagger/>. En esta vista se aprecian los distintos endpoints disponibles, organizados por recurso, como el módulo app, que incluye operaciones CRUD (crear, leer, actualizar y eliminar aplicaciones), y el endpoint de autenticación /auth-token/ para la obtención del token de acceso. Swagger permite explorar cada operación detallando el método HTTP utilizado (GET, POST, PUT, PATCH, DELETE), los parámetros requeridos y las posibles respuestas.

En esta sección se presenta la documentación generada mediante Swagger UI para la API v1 del sistema Mobile Audit, enfocada en los módulos de findings, permissions y scans. Esta interfaz permite explorar de manera estructurada los endpoints expuestos por el sistema, organizados por recurso, y revisar las operaciones soportadas en cada caso. La vista proporciona tanto la descripción de los métodos HTTP disponibles como los parámetros de entrada y las posibles respuestas, lo que la convierte en una herramienta clave para verificar el correcto funcionamiento de la API durante la auditoría móvil.

Figura 32: Endpoints de Findings, Permissions y Scans en Swagger UI

Fuente: Elaboración propia.

Nota: La figura muestra la documentación de los endpoints correspondientes a los módulos finding, permission y scan de la API v1 en Swagger UI, accesible desde el entorno desplegado en <http://localhost:8888/swagger/>. Cada módulo ofrece operaciones CRUD completas (crear, leer, actualizar y eliminar), implementadas con los métodos HTTP estándar (GET, POST, PUT, PATCH, DELETE), además de operaciones específicas como la ejecución de un scan asociado a un finding o un permission. Asimismo, en la parte inferior se listan los modelos que estructuran los datos intercambiados: Application, AuthToken, Finding, Permission y Scan.

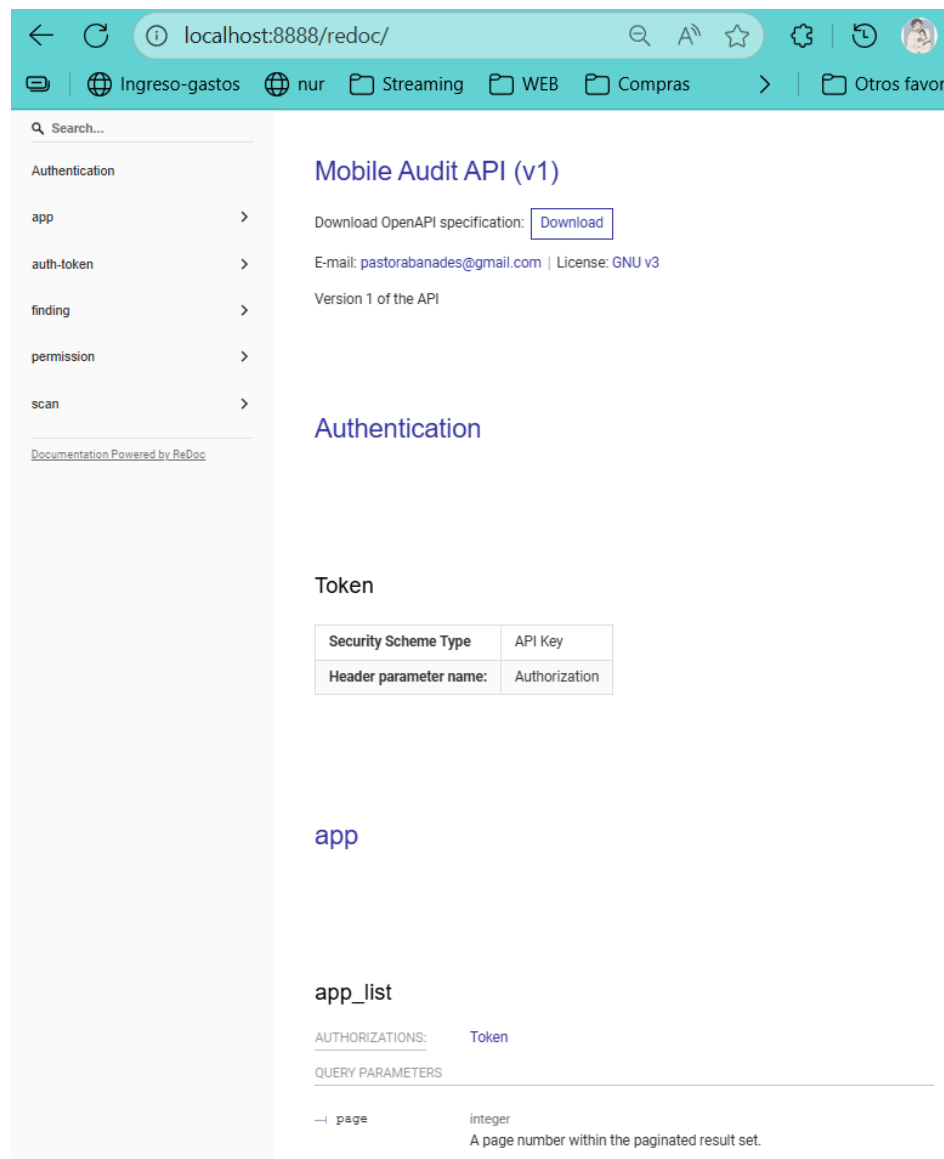
ReDoc es una herramienta de visualización de documentación de APIs que se distingue por ofrecer una interfaz más estructurada, jerárquica y orientada a la lectura.

En el laboratorio, al acceder a la URL <http://localhost:8888/redoc/>, se obtiene una documentación generada automáticamente a partir de la especificación OpenAPI de la API v1. Su principal característica es la presentación en formato de navegación lateral, lo que permite explorar los endpoints y sus

detalles de forma ordenada y legible, priorizando la claridad por encima de la ejecución directa.

A diferencia de Swagger, ReDoc no está enfocada en la interacción en tiempo real con los servicios, sino en la presentación estática, limpia y profesional de la API, lo cual resulta especialmente útil en entornos de publicación o entrega de documentación a terceros que requieren comprender la estructura y alcance de los servicios sin necesariamente consumirlos en ese mismo entorno.

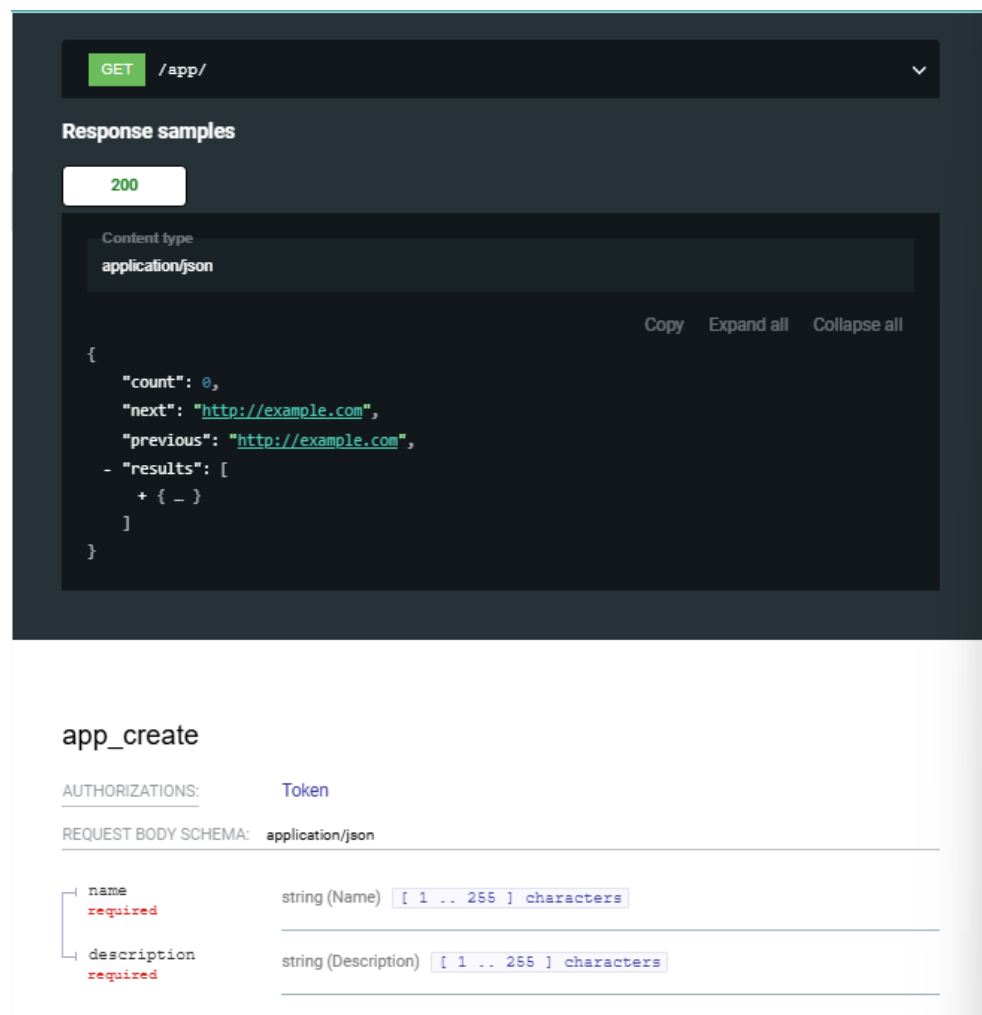
Figura 33: Documentación de la API v1 en ReDoc



Fuente: Elaboración propia.

Nota: La figura muestra la interfaz de ReDoc correspondiente a la API v1 de Mobile Audit, desplegada en la URL <http://localhost:8888/redoc/>. En esta captura se observa la documentación organizada en secciones principales, que incluyen los recursos `app`, `auth-token`, `finding`, `permission` y `scan`. ReDoc detalla los esquemas de autenticación mediante token, indicando que las solicitudes deben incorporar el encabezado `Authorization: Token <ApiKey>`, además de describir los parámetros de consulta y las opciones de paginación. La interfaz permite descargar la especificación OpenAPI en formato estándar y facilita la comprensión de la API gracias a su diseño estructurado y jerárquico.

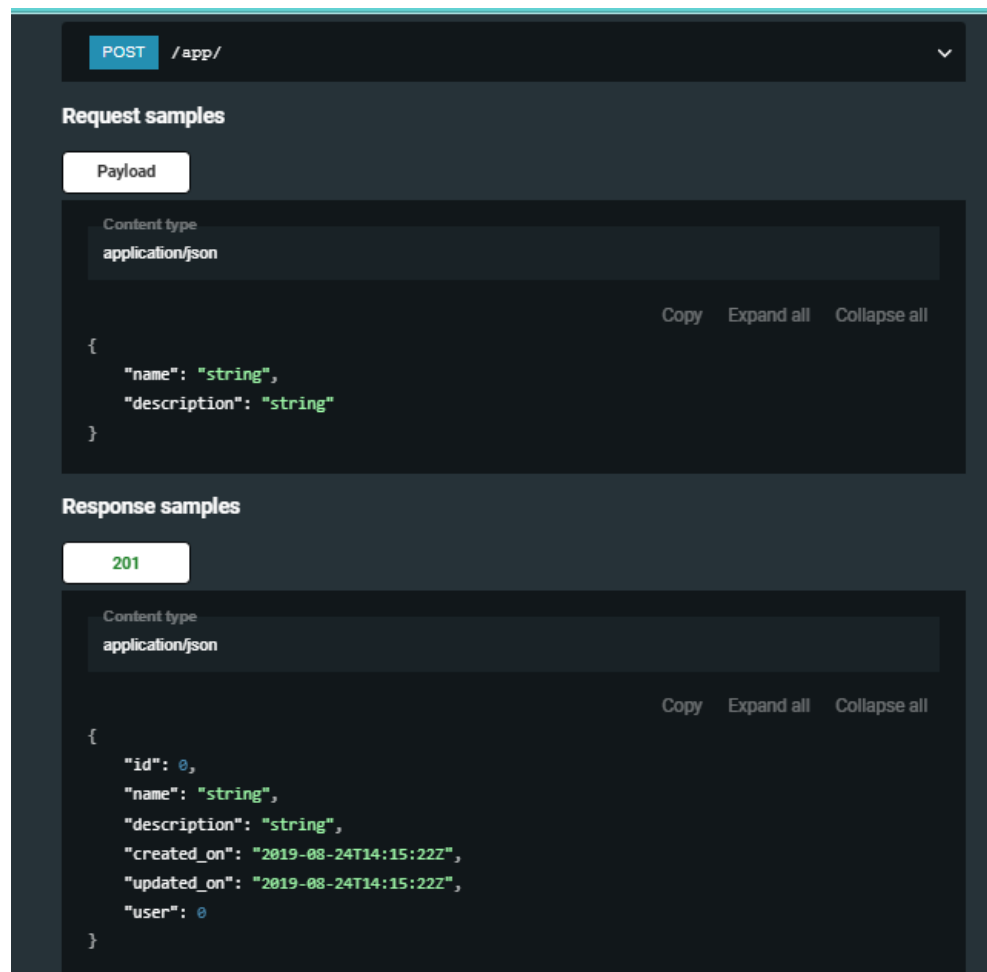
Figura 34: Acceso a la interfaz web de Mobile Audit en el navegador



Fuente: Elaboración propia.

Nota: La figura muestra la documentación en ReDoc del endpoint /app/ perteneciente a la API v1 de Mobile Audit. En la parte superior se observa la operación GET /app/, cuyo ejemplo de respuesta JSON incluye campos como count, next, previous y results, lo que evidencia que el recurso está diseñado para manejar resultados paginados. Asimismo, se presenta la operación POST /app/ (app_create), donde se especifica el esquema de la petición en formato application/json, incluyendo los campos obligatorios name y description con validaciones de longitud. La captura también refleja la necesidad de autenticación mediante Token, lo cual asegura que solo usuarios autorizados puedan crear y consultar aplicaciones.

Figura 35: Acceso a la interfaz web de Mobile Audit en el navegador



Fuente: Elaboración propia.

Nota: La figura presenta la documentación del recurso POST /app/ en ReDoc, donde se observa un ejemplo de solicitud en formato JSON que incluye los parámetros obligatorios name y description para registrar una nueva aplicación en la plataforma. Asimismo, se muestra la respuesta esperada con código de estado 201 (Created), en la cual el sistema devuelve un objeto JSON que contiene el id asignado, junto con los campos name, description, las marcas temporales created_on y updated_on, y el identificador del user que ejecutó la operación.

3. Conclusiones

El laboratorio permitió validar la efectividad de Mobile Audit como herramienta integral para la auditoría de seguridad móvil, demostrando que el despliegue mediante contenedores Docker garantiza la portabilidad y consistencia del entorno de análisis. La ejecución de los contenedores y la correcta integración de servicios fundamentales aseguraron la disponibilidad de un entorno estable y confiable para la evaluación de APKs.

Los análisis realizados evidenciaron vulnerabilidades significativas en las aplicaciones F-Droid y DailyTube, incluyendo permisos peligrosos, consultas SQL en crudo, exposición de información sensible y componentes mal configurados. Estos hallazgos permiten priorizar acciones de mitigación y refuerzan la importancia de realizar auditorías periódicas para reducir los riesgos asociados a aplicaciones móviles.

Finalmente, la documentación de la API mediante Swagger y ReDoc facilitó la comprensión de los endpoints disponibles y la interacción con los servicios, destacando la relevancia de contar con interfaces claras y accesibles para desarrolladores y auditores. Se constató que la combinación de análisis estático, revisión de permisos, componentes y certificados, junto con documentación técnica, proporciona una visión completa del estado de seguridad de las aplicaciones móviles, constituyendo una base sólida para futuras auditorías y mejoras en la seguridad de software móvil.

4. Referencias Bibliográficas

- Calder, A., & Watkins, S. (2015). IT Governance: An International Guide to Data Security and ISO27001/ISO27002. Kogan Page.
- Orebaugh, A., Ramirez, D., Beale, J., & Wright, J. (2006). Wireshark & Ethereal Network Protocol Analyzer Toolkit. Syngress.
- Stallings, W., & Brown, L. (2018). Computer Security: Principles and Practice (4th ed.). Pearson.
- Weaver, A. C. (2013). Computer Security: A Hands-on Approach. CRC Press.
- Del Peso, E., Del Peso, M., & Piattini, M. (2008). Auditoría de tecnologías y sistemas de información. Rama. ISBN 9788499646039.