**UNIVERSIDAD PRIVADA DE TACNA**

**FACULTAD DE INGENIERÍA**

**ESCUELA DE INGENIERÍA DE SISTEMAS**

****

**Informe de Laboratorio**

**Laboratorio 02  
“Auditoria Móvil”**

**Que se presenta para el curso:**

“Auditoría de sistemas”

**LO**

**Integrante(s):**

* Martinez Yufra, Ericka Esther 2018000368

**Docente:**

Dr. Oscar Juan Jimenez Flores

**TACNA – PERÚ**

2025

**CONTENIDO**

[RESUMEN 3](#_Toc206165042)

[INTRODUCCIÓN 4](#_Toc206165043)

[1. Información sobre el evento práctico 5](#_Toc206165044)

[1.1. Título del evento práctico 5](#_Toc206165045)

[Laboratorio 01. Auditoria de vulnerabilidades en imágenes Docker 5](#_Toc206165046)

[1.2. Objetivos 5](#_Toc206165047)

[1.3. Tiempo de duración (horas) 5](#_Toc206165048)

[1.4. Resultados de Aprendizaje (RA) 5](#_Toc206165049)

[1.5. Recursos (Equipos, materiales, programas y otros) 5](#_Toc206165050)

[2. Procedimiento o Metodología 6](#_Toc206165051)

[3. Conclusiones 18](#_Toc206165052)

[4. Referencias Bibliográficas 19](#_Toc206165053)

# RESUMEN

En el Laboratorio 01 “Auditoría de vulnerabilidades en imágenes Docker” se llevó a cabo un análisis exhaustivo de seguridad sobre la imagen base ubuntu:latest utilizando la herramienta Trivy, con el objetivo de identificar vulnerabilidades presentes en paquetes y librerías del sistema, evaluar los riesgos asociados y aplicar controles efectivos para minimizar dichos riesgos. Durante la actividad, se descargó la imagen desde Docker Hub, se creó un contenedor a partir de ella y se realizó un escaneo inicial que permitió detectar 15 vulnerabilidades distribuidas en niveles LOW, MEDIUM y HIGH, afectando librerías esenciales, módulos de autenticación PAM, herramientas de red y archivos sensibles, incluyendo posibles secretos y certificados.

Posteriormente, se aplicaron medidas de mitigación que incluyeron la actualización completa de paquetes y la distribución del sistema (apt update y apt-get dist-upgrade), la eliminación de software innecesario y la regeneración de archivos sensibles. A partir del contenedor actualizado se generó una nueva imagen denominada ubuntu-updated, la cual fue auditada nuevamente, evidenciando una reducción drástica de vulnerabilidades, quedando únicamente 2 vulnerabilidades de severidad MEDIUM en librerías críticas (libc-bin y libc6), asociadas al CVE-2025-4802.

Este resultado confirma que la aplicación de procedimientos de actualización, limpieza y endurecimiento de la imagen permite minimizar significativamente la superficie de ataque, reduciendo riesgos de explotación y garantizando un entorno más seguro para pruebas y despliegues controlados. La actividad fortaleció competencias en análisis de problemas, uso de herramientas de auditoría, diseño de soluciones de seguridad y toma de decisiones basadas en riesgos, cumpliendo con los objetivos planteados del laboratorio y los resultados de aprendizaje asociados.

# INTRODUCCIÓN

El presente laboratorio, titulado *Auditoría de vulnerabilidades en imágenes Docker*, se desarrolló con el propósito de identificar y analizar posibles riesgos de seguridad asociados a entornos virtualizados en contenedores. Durante la actividad se aplicaron metodologías de escaneo de vulnerabilidades utilizando la herramienta Trivy sobre una imagen base de Ubuntu descargada desde Docker Hub, con el objetivo de establecer un diagnóstico inicial del estado de seguridad del sistema operativo contenido en la imagen. La práctica incluyó la creación de un entorno controlado para la ejecución de contenedores, la configuración de directorios de caché y la utilización de procedimientos de actualización y endurecimiento de la imagen, asegurando así que los resultados obtenidos reflejaran un escenario realista y seguro.

Asimismo, el laboratorio permitió a los participantes desarrollar competencias vinculadas con la gestión de vulnerabilidades, el análisis de riesgos y la aplicación de controles correctivos. Se enfatizó en la importancia de mantener entornos de contenedores actualizados y depurados, así como en la identificación de paquetes y componentes potencialmente inseguros. Este enfoque facilitó la comprensión de la relación entre la configuración inicial de la imagen, la exposición a amenazas y las medidas que pueden implementarse para mitigar los riesgos detectados, promoviendo la práctica de buenas normas de seguridad y el uso ético de herramientas de auditoría.

**GUÍA DE LABORATORIO N.º 02   
“AUDITORIA MÓVIL”**

|  |
| --- |
| Información sobre el evento práctico |

### Título del evento práctico

### Laboratorio 02. Auditoría Móvil

### Objetivos

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

### Tiempo de duración (horas)

06 horas académicas

### 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

### Recursos (Equipos, materiales, programas y otros)

* Computador con S.O. Windows
* Descargar e instalar repositorio GIT

<https://github.com/OscarJimenezFlores/CursoAuditoria/tree/main/AuditoriaMovil>

|  |
| --- |
| Procedimiento o Metodología |

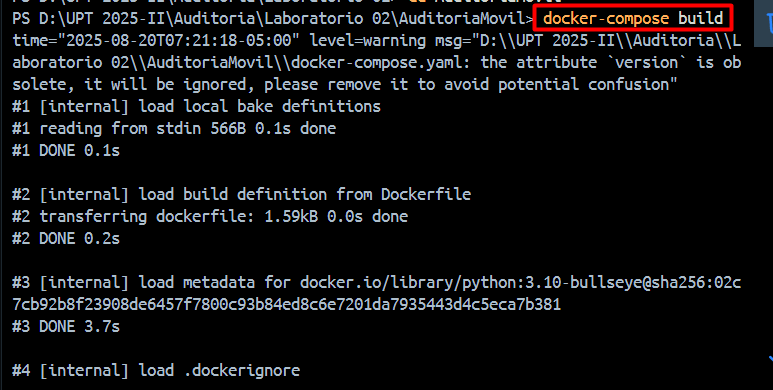
* 1. **Construcción de la imagen Docker**

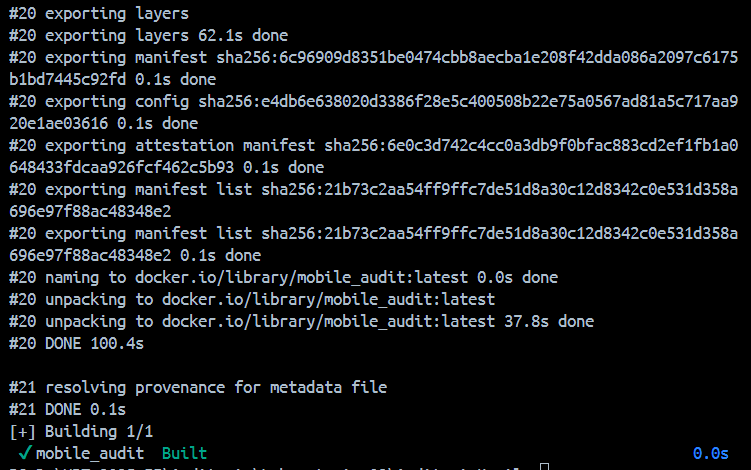
El sistema cuenta con imágenes base alojadas en Docker Hub, construidas sobre python:buster, en diferentes versiones:

* 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



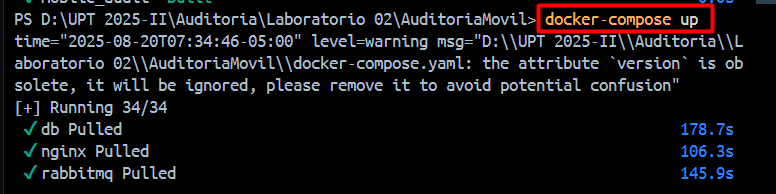
Fuente: Elaboración propia.

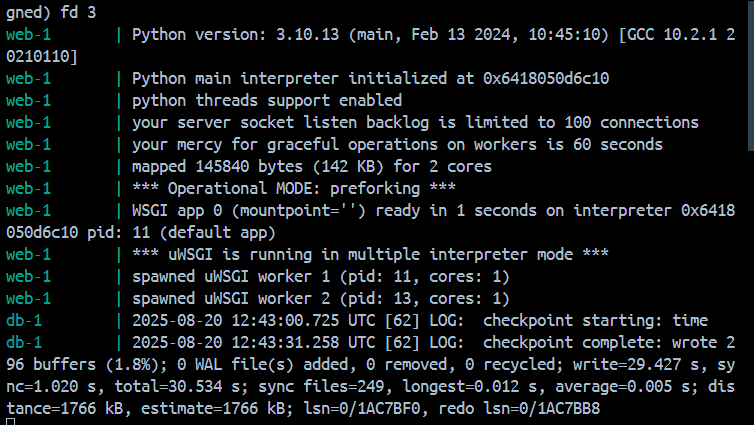
*Nota:* *La figura muestra el proceso de construcción de la imagen a partir del archivo Dockerfile definido en el proyecto. En este paso, Docker descarga las dependencias, configura el entorno Python y prepara la aplicación para ser ejecutada en contenedores.*

* 1. **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

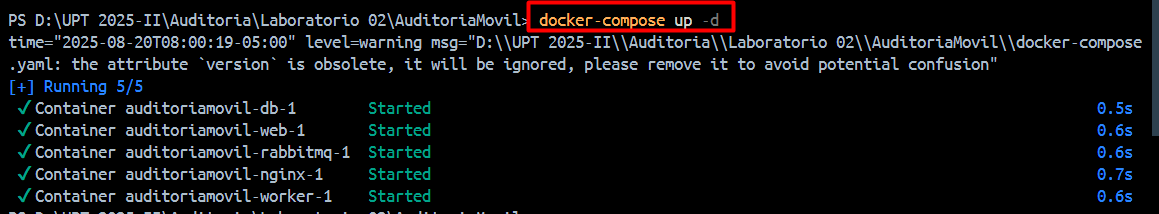


Fuente: Elaboración propia.

*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.*

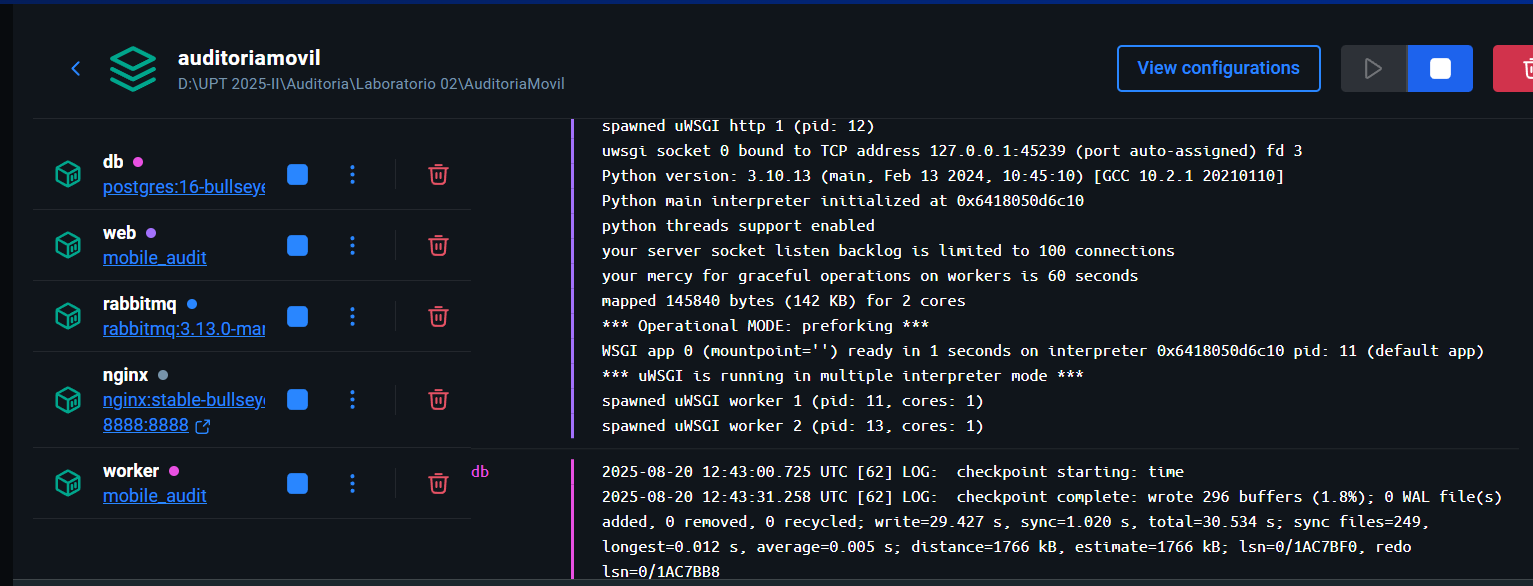
Opcionalmente, se pueden levantar en segundo plano: docker-compose up -d

**Figura 4:** Ejecución de los contenedores de Auditoría Móvil mediante Docker Compose

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 3:** Ejecución de contenedores de Mobile Audit en Docker Desktop



Fuente: Elaboración propia.

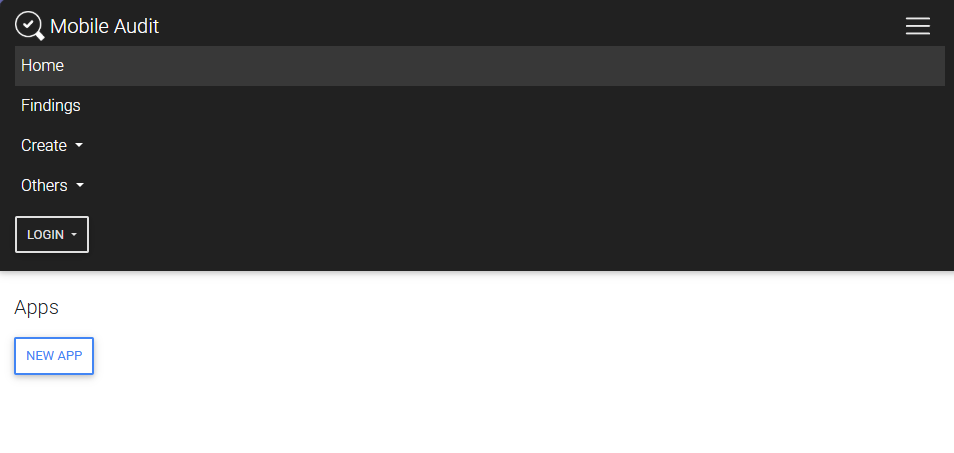
*Nota:* *En la figura se observa la ejecución de los distintos contenedores que conforman la aplicación Mobile Audit a través de Docker Desktop, levantados mediante el comando docker-compose up. Los servicios desplegados corresponden a la base de datos PostgreSQL, el servidor web Nginx, el sistema de 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.*

* 1. **Acceso a la aplicación**

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 4:** Acceso a la interfaz web de Mobile Audit en el navegador

Fuente: Elaboración propia.

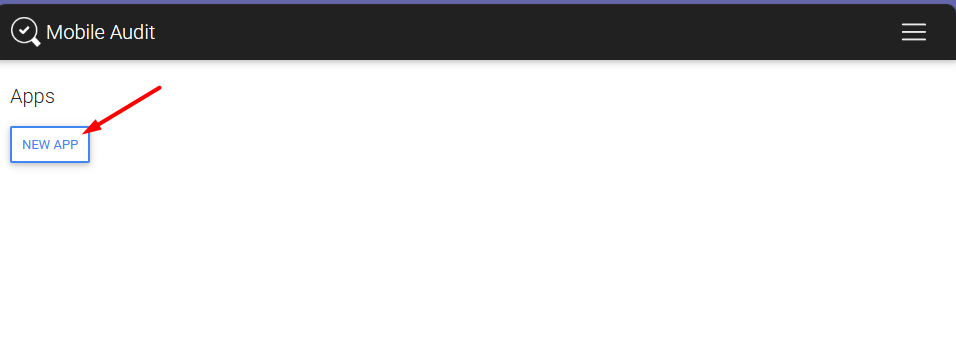
*Nota:* *La figura muestra el acceso exitoso a la aplicación web Mobile Audit a través del navegador en la dirección http://localhost:8888/, una vez desplegados todos los contenedores mediante Docker Compose. En la parte superior se observa la barra de navegación con las secciones principales de la plataforma, 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.*

* 1. **Ejecución del análisis de un APK**

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

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

**Figura 4:** 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”.*

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

En el formulario New App, completa:

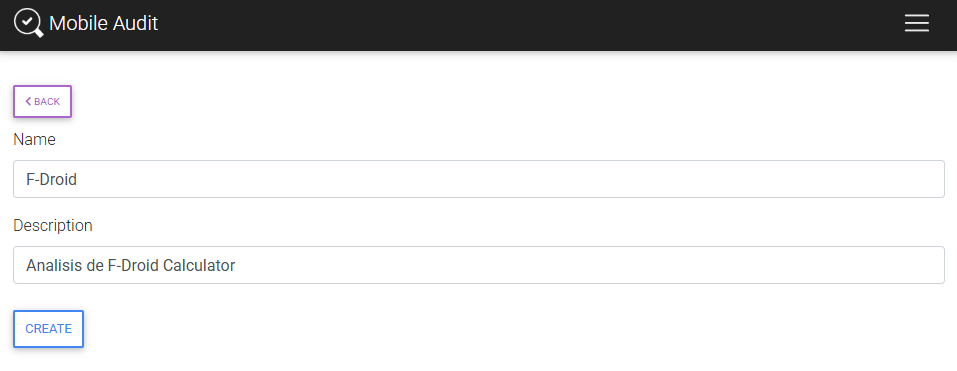
* Name: F-Droid (o el identificador que prefieras).
* APK file: Selecciona el archivo F-Droid.apk desde tu equipo.
* (Opcional) Description/Tags para clasificar el análisis.

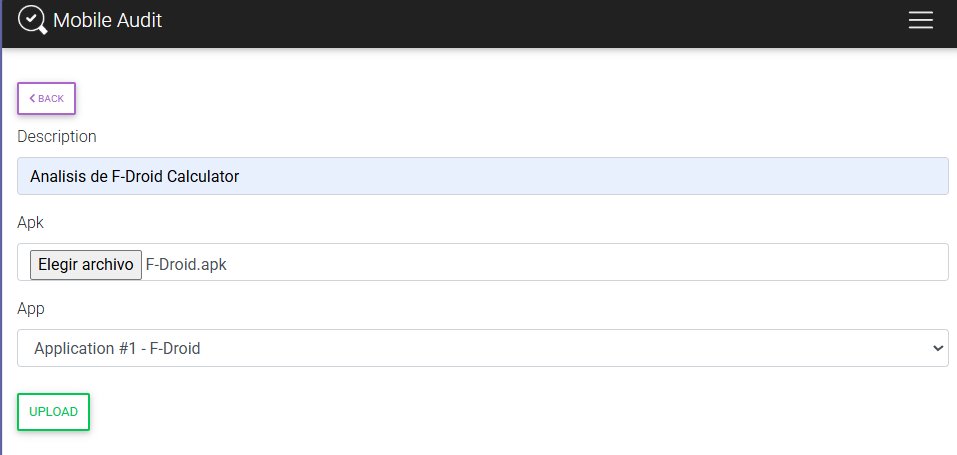
Pulsa Create o Upload para enviar el APK.

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 4:** Formulario de carga de APK “F-Droid”



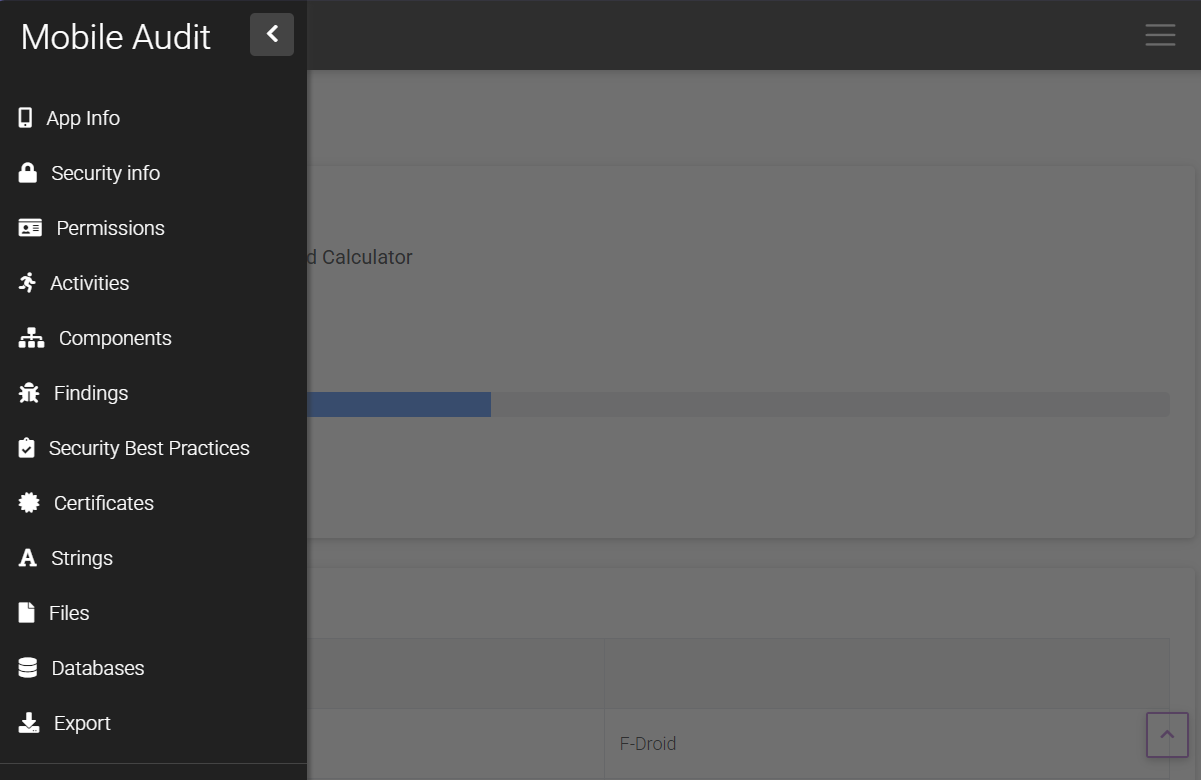
Fuente: Elaboración propia.

*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.*

1. 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 4:** Barra lateral del escaneo y menú de secciones

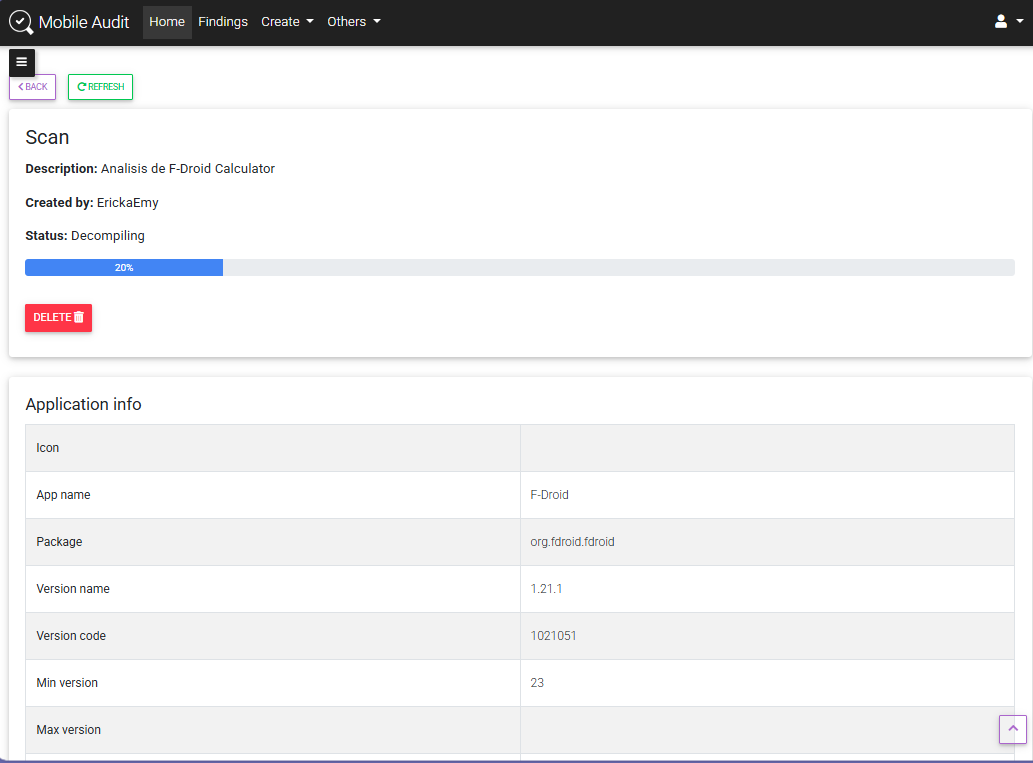
Fuente: Elaboración propia.

*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.*

1. 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 4:** Acceso a la interfaz web de Mobile Audit en el navegador

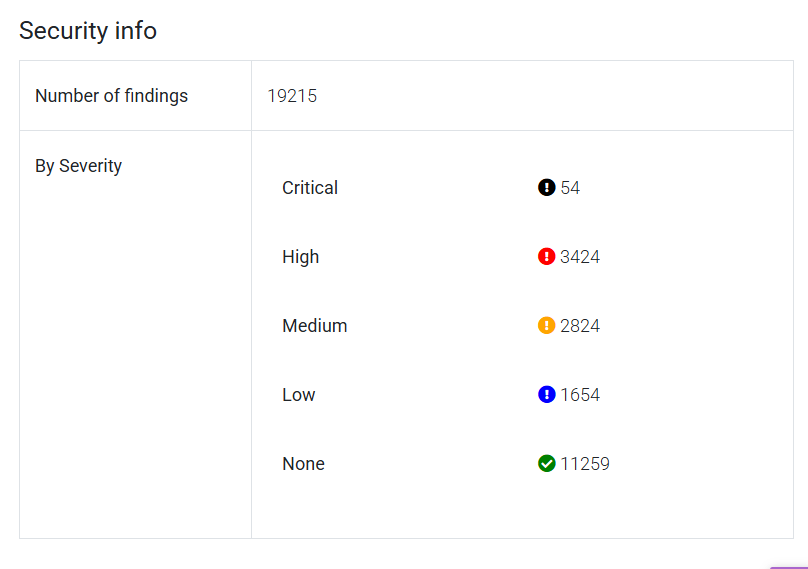
Fuente: Elaboración propia.

*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.*

1. 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 4:** Acceso a la interfaz web de Mobile Audit en el navegador



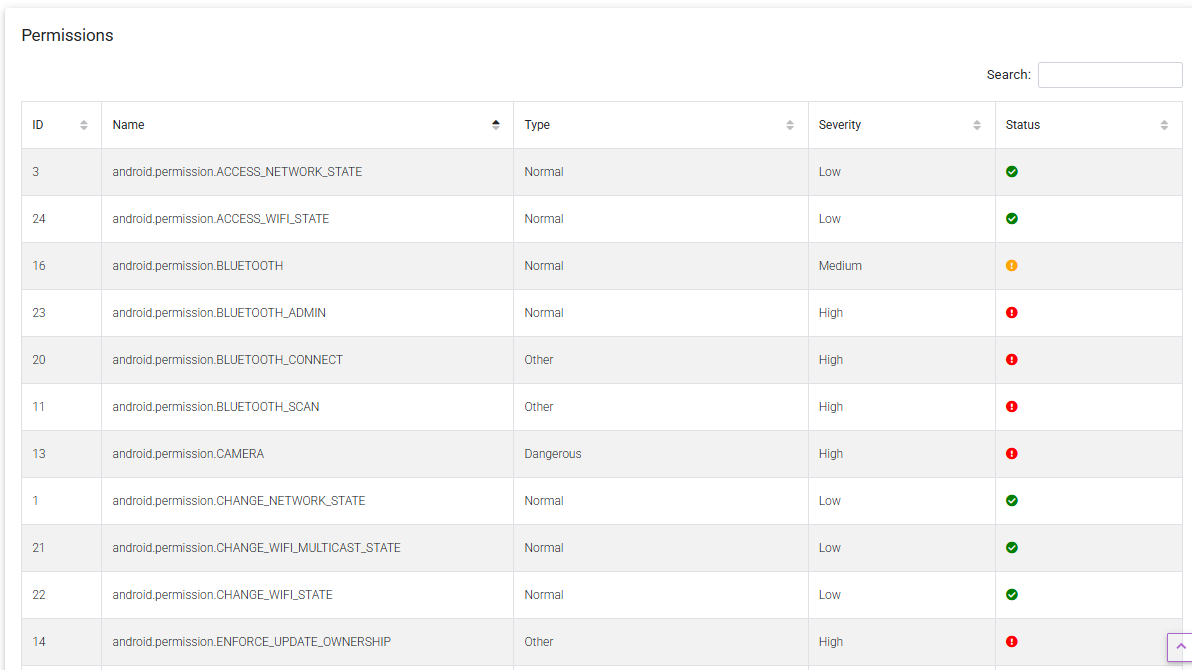
Fuente: Elaboración propia.

*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.*

1. 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 4:** Acceso a la interfaz web de Mobile Audit en el navegador

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.*

1. Activities

Texto

**Figura 4:** Acceso a la interfaz web de Mobile Audit en el navegador

Fuente: Elaboración propia.

*Nota: La*

1. Components

Texto

**Figura 4:** Acceso a la interfaz web de Mobile Audit en el navegador

Fuente: Elaboración propia.

*Nota: La*

1. Findings

Texto

**Figura 4:** Acceso a la interfaz web de Mobile Audit en el navegador

Fuente: Elaboración propia.

*Nota: La*

1. Security Best Practices

Texto

**Figura 4:** Acceso a la interfaz web de Mobile Audit en el navegador

Fuente: Elaboración propia.

*Nota: La*

1. Certificates

Texto

**Figura 4:** Acceso a la interfaz web de Mobile Audit en el navegador

Fuente: Elaboración propia.

*Nota: La*

1. Strings

Texto

**Figura 4:** Acceso a la interfaz web de Mobile Audit en el navegador

Fuente: Elaboración propia.

*Nota: La*

1. Files

Texto

**Figura 4:** Acceso a la interfaz web de Mobile Audit en el navegador

Fuente: Elaboración propia.

*Nota: La*

1. Databases

Texto

**Figura 4:** Acceso a la interfaz web de Mobile Audit en el navegador

Fuente: Elaboración propia.

*Nota: La*

1. Export

Texto

**Figura 4:** Acceso a la interfaz web de Mobile Audit en el navegador

Fuente: Elaboración propia.

*Nota: La*

1. Security Info

Texto

**Figura 4:** Acceso a la interfaz web de Mobile Audit en el navegador

Fuente: Elaboración propia.

*Nota: La*

1. Security Info

Texto

**Figura 4:** Acceso a la interfaz web de Mobile Audit en el navegador

Fuente: Elaboración propia.

*Nota: La*

1. Security Info

Texto

**Figura 4:** Acceso a la interfaz web de Mobile Audit en el navegador

Fuente: Elaboración propia.

*Nota: La*

1. Security Info

Texto

**Figura 4:** Acceso a la interfaz web de Mobile Audit en el navegador

Fuente: Elaboración propia.

*Nota: La*

1. Componentes (Activities, Services, Receivers, Providers)
   * + Ruta en la UI: Components.
     + Qué validar/explicar: listado de cada componente con su atributo exported, filtros de intent, permisos requeridos, y posibles vectores de entrada.

Figura 6: Inventario de componentes Android del APK “F-Droid”

Nota: La figura muestra el inventario de activities, services, broadcast receivers y content providers extraído del APK “F-Droid”, junto con sus flags de exposición y permisos asociados, proporcionando una visión de los puntos de interacción entre la aplicación y el sistema o terceros que puede derivar en superficies de ataque si existen configuraciones laxas.

1. Hallazgos SAST (clasificados por CWE y Mobile Top 10)

Ruta en la UI: SAST findings / Findings.

Qué validar/explicar: vulnerabilidades clasificadas por severidad, referencia a CWE y mapeo a OWASP Mobile Top 10. Abre un hallazgo para ver descripción, evidencia, ubicación (archivo/línea) y recomendación.

Edición/triage: puedes editar la descripción, marcar falsos positivos y reclasificar.

📸 Figura 7: Hallazgos SAST del APK “F-Droid”

Nota: La figura consolida las vulnerabilidades detectadas por el motor estático para el APK “F-Droid”, indicando severidades, referencias a CWE y su alineamiento con OWASP Mobile Top 10; al ingresar en un hallazgo se observa la evidencia y la ubicación en el código desensamblado o recursos, funcionalidades que permiten priorizar mitigaciones y realizar triage marcando falsos positivos cuando corresponde.

8. Mejores prácticas implementadas

Ruta en la UI: Best practices.

Qué validar/explicar: presencia de mecanismos seguros (pinning, uso de HTTPS, almacenamiento seguro, protecciones de logs, ProGuard/R8, etc.).

📸 Figura 8: Mejores prácticas de seguridad identificadas

Nota: La figura expone las prácticas de seguridad correctamente aplicadas en el APK “F-Droid”, como el uso de canales cifrados o políticas de registro restrictivas, evidenciando controles defensivos que reducen la exposición a amenazas y complementan el conjunto de hallazgos del análisis estático.

9. Integración con VirusTotal (opcional para mostrar información)

Si no configuras la API, la sección puede indicar “deshabilitado / sin datos”.

Para mostrar resultados, habilita la integración:

Edita el .env en la carpeta del proyecto y ajusta:

VIRUSTOTAL\_ENABLED=True

VIRUSTOTAL\_API\_KEY=<TU\_API\_KEY>

VIRUSTOTAL\_UPLOAD=True ; opcional, sube el APK si no existe reporte

Reinicia servicios:

docker-compose down

docker-compose up -d

Reabre el escaneo de “F-Droid” y entra a VirusTotal.

📸 Figura 9: Información de VirusTotal para el APK “F-Droid”

Nota: La figura refleja la consulta a VirusTotal habilitada mediante variables de entorno y clave API, mostrando el hash del archivo, fecha de análisis y los motores que detectan o no actividad maliciosa; esta integración añade inteligencia de amenazas de terceros al resultado local, permitiendo contrastar indicadores con una base reputacional amplia.

10. Información de certificados (firma)

Ruta en la UI: Certificates.

Qué validar/explicar: detalles del certificado de firma (Subject/Issuer), algoritmo, huella SHA-256/SHA-1, validez, y si el build parece debug o release.

📸 Figura 10: Certificados y huellas criptográficas del APK

Nota: La figura presenta la información del certificado con el que fue firmado el APK “F-Droid”, incluyendo sujeto, emisor, algoritmo y huellas digitales, datos que sirven para validar la procedencia del binario, identificar versiones firmadas con llaves de depuración o potenciales anomalías de integridad.

11. Cadenas (strings) extraídas

Ruta en la UI: Strings / Cadenas.

Qué validar/explicar: búsqueda de URLs, tokens, claves o mensajes sensibles; utiliza el buscador de la sección si está disponible.

📸 Figura 11: Cadenas relevantes extraídas del APK

Nota: La figura muestra el conjunto de cadenas identificadas en los recursos y binarios del APK “F-Droid”, útiles para descubrir endpoints de red, parámetros ocultos o secretos inadvertidamente embebidos, lo que puede conducir a hallazgos de exposición de información o abuso de funcionalidades internas.

12. Bases de datos internas

Ruta en la UI: Databases / Bases de datos.

Qué validar/explicar: archivos .db o .sqlite incluidos, tablas detectadas si la herramienta las indexa, y su ubicación dentro del paquete.

📸 Figura 12: Bases de datos detectadas dentro del APK

Nota: La figura evidencia los archivos de base de datos distribuidos con el APK “F-Droid”, detallando rutas y, cuando es posible, estructuras; este resultado sirve para identificar almacenamiento local de información sensible y validar políticas de cifrado o controles de acceso.

13. Archivos y estructura del paquete

Ruta en la UI: Files / Archivos (árbol de recursos).

Qué validar/explicar: estructura del APK (carpetas res/, assets/, lib/, META-INF/, classes\*.dex, etc.) y presencia de configuraciones específicas como network\_security\_config.xml.

📸 Figura 13: Estructura de archivos interna del APK

Nota: La figura muestra el árbol de archivos del APK “F-Droid” con los recursos, bibliotecas nativas y metadatos empaquetados, permitiendo localizar configuraciones de seguridad, librerías de terceros y activos que influyen en el comportamiento de la aplicación y su superficie de ataque.

14. Exportación de resultados (PDF / CSV / Markdown)

Ruta en la UI: dentro del escaneo, menú Export (puede estar en la parte superior o en la barra lateral).

Acciones:

Export to PDF: reporte consolidado.

Export to CSV: hallazgos tabulares para análisis externo.

Export to Markdown: para documentación técnica.

📸 Figura 14: Exportación de resultados del análisis

Nota: La figura ilustra las opciones de exportación disponibles en Mobile Audit para el escaneo del APK “F-Droid”, con la generación de reportes en PDF, CSV y Markdown que facilitan la trazabilidad, el intercambio y la incorporación de hallazgos en procesos de gestión de vulnerabilidades o entregables formales de auditoría.

15. Motor de reglas y patrones

Objetivo: mostrar que los patrones SAST/malware son configurables.

Ruta en la UI: http://localhost:8888/patterns (o menú Others → Patterns si existe).

📸 Figura 15: Reglas y patrones de detección configurables

Nota: La figura expone el motor de reglas utilizado durante el análisis, donde es posible activar o desactivar patrones de detección de vulnerabilidades y código malicioso, permitiendo ajustar la sensibilidad del escaneo y calibrar la herramienta a diferentes contextos de evaluación.

16. Modelos de datos (esquema)

Objetivo: evidenciar la estructura de datos empleada para relacionar resultados.

Ruta en la UI: http://localhost:8888/models (o Others → Models).

📸 Figura 16: Esquema de modelos de Mobile Audit

Nota: La figura muestra el diagrama de modelos que sustentan la persistencia de los escaneos, hallazgos, componentes, certificados y demás entidades, destacando las relaciones que permiten consultas correlacionadas y la generación de conclusiones sólidas para cada APK auditado.

17. Verificación de componentes en ejecución (Docker)

Objetivo: asociar el resultado del escaneo con la infraestructura activa.

Comando (opcional):

docker ps

Qué validar/explicar: contenedores db, nginx, rabbitmq, worker, web activos; puertos expuestos (8888:8888); imágenes base (por ejemplo, mpast/mobile\_audit:3.0.0 basada en python:3.9.16-buster).

📸 Figura 17: Componentes de la solución en contenedores Docker

Nota: La figura confirma la ejecución simultánea de los contenedores db (PostgreSQL), nginx (proxy inverso), rabbitmq (broker), worker (Celery) y web (Django/uWSGI), evidenciando la arquitectura modular de Mobile Audit y la utilización de imágenes base derivadas de python:3.9-buster para garantizar un despliegue consistente y reproducible.

18. (Opcional) Uso de la API para evidenciar autenticación

Objetivo: complementar el informe con interacción programática.

Obtener token:

curl -X POST http://localhost:8888/api/v1/auth-token/ \

-H "Content-Type: application/json" \

-d '{"username":"<usuario>","password":"<clave>"}'

Consumir API autenticada:

curl -H "Authorization: Token <TOKEN\_OBTENIDO>" \

http://localhost:8888/api/v1/

📸 Figura 18: Autenticación y consumo de la API v1

Nota: La figura documenta la obtención del token de acceso mediante el endpoint /api/v1/auth-token/ y su uso en el encabezado Authorization: Token <ApiKey> para consumir la API, demostrando la disponibilidad de interfaces REST autenticadas para automatización e integración con flujos de auditoría.

**Figura 4:** Acceso a la interfaz web de Mobile Audit en el navegador

Fuente: Elaboración propia.

*Nota:*

1. Mobile Audit ejecuta automáticamente:

* Análisis SAST → revisa código y configuraciones inseguras.
* Análisis de malware → busca cadenas maliciosas, URLs y consultas externas.

1. Los resultados se presentan en una barra lateral de navegación que organiza la información en:

* Información de la aplicación
* Información de seguridad
* Componentes
* Hallazgos SAST
* Mejores prácticas
* VirusTotal
* Certificados
* Cadenas
* Bases de datos
* Archivos

**Figura 5:** Construcción de la imagen Docker de Mobile Audit

Fuente: Elaboración propia.

*Nota:* *La figura*

* 1. **Construcción de la imagen Docker**

El

**Figura 5:** Construcción de la imagen Docker de Mobile Audit

Fuente: Elaboración propia.

*Nota:* *La figura*

* 1. **Construcción de la imagen Docker**

El

**Figura 5:** Construcción de la imagen Docker de Mobile Audit

Fuente: Elaboración propia.

*Nota:* *La figura*

* 1. **Construcción de la imagen Docker**

El

**Figura 5:** Construcción de la imagen Docker de Mobile Audit

Fuente: Elaboración propia.

*Nota:* *La figura*

Para crear la imagen localmente:

**Figura 6:** Construcción de la imagen Docker de Mobile Audit

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.*

Teniendo un S.O. con Docker instalado, procedemos a abrir una terminal y descargar la imagen de Ubuntu con el siguiente comando: docker pull ubuntu:latest

**Figura 7:** Descarga de la imagen ubuntu:latest desde Docker Hub.

Fuente: Elaboración propia.

*Nota:* *Texto*

La imagen quedaría descargada y debe recordar el “Image ID” con el siguiente comando: docker images

* 1. **Análisis Comparativo**

En la primera evaluación de seguridad realizada con Trivy, la imagen auditada presentaba 15 vulnerabilidades distribuidas principalmente en severidades LOW, MEDIUM y HIGH. Estas vulnerabilidades estaban asociadas a múltiples paquetes del sistema, incluidas librerías de propósito general, herramientas de red y componentes criptográficos, además de llaves privadas y certificados incrustados que fueron identificados como riesgos de tipo secret. La presencia de estos elementos reflejaba un estado inicial en el que la imagen contenía software innecesario, versiones obsoletas y datos sensibles no gestionados adecuadamente.

**Tabla 1:** Comparativa de vulnerabilidades antes y después de la optimización de la imagen

|  |  |  |  |
| --- | --- | --- | --- |
| **Categoría / Métrica** | **Análisis Inicial** | **Análisis Final** | **Variación** |
| Severidad CRITICAL | 0 | 0 | 0 |
| Severidad HIGH | 4 | 0 | -4 |
| Severidad MEDIUM | 6 | 2 | -4 |
| Severidad LOW | 5 | 0 | -5 |
| Vulnerabilidades en librerías base | 5 | 2 | -3 |
| Vulnerabilidades en herramientas de red | 4 | 0 | -4 |
| Vulnerabilidades en secretos/credenciales | 3 | 0 | -3 |
| Otros componentes | 3 | 0 | -3 |
| **Total de vulnerabilidades** | 15 | 2 | -13 |

*Nota: La reducción del número total de vulnerabilidades (de 15 a 2) es consecuencia directa de la aplicación de medidas de endurecimiento de la imagen, que incluyeron la eliminación de paquetes no requeridos, actualización de librerías, y regeneración/eliminación de archivos sensibles. En el análisis final, las únicas vulnerabilidades detectadas corresponden a librerías del sistema (libc-bin y libc6), ambas con severidad MEDIUM, lo que sugiere que la imagen se encuentra en un estado seguro para entornos de prueba o controlados, aunque aún requiere aplicar parches específicos para alcanzar un estado completamente libre de vulnerabilidades conocidas.*

|  |
| --- |
| Conclusiones |

En el desarrollo de este laboratorio se evidenció que la auditoría de vulnerabilidades en imágenes Docker es un proceso crítico para la identificación y mitigación de riesgos en entornos de contenedores. El análisis inicial realizado sobre la imagen ubuntu:latest permitió detectar un total de 15 vulnerabilidades, distribuidas en niveles de severidad LOW (6), MEDIUM (9) y HIGH (4), afectando componentes esenciales como librerías del sistema, herramientas de red, módulos de autenticación PAM y archivos sensibles que representan riesgos de exposición de secretos. Este hallazgo inicial evidenció que las imágenes base contienen paquetes desactualizados y componentes que podrían ser explotados si se desplegaran en entornos de producción sin medidas correctivas.

La ejecución de las acciones de actualización y endurecimiento de la imagen —incluyendo la actualización de paquetes mediante apt update y apt-get dist-upgrade, la eliminación de software innecesario y la regeneración de archivos sensibles— permitió optimizar significativamente el estado de seguridad de la imagen. Posteriormente, el análisis comparativo con Trivy evidenció que las vulnerabilidades se redujeron a solo 2 de severidad MEDIUM, asociadas a las librerías esenciales libc-bin y libc6, correspondientes al CVE-2025-4802. Esto refleja que la mayoría de los riesgos originales fueron mitigados, incluyendo la eliminación de posibles exploits asociados a herramientas de red, módulos PAM y secretos, reduciendo la superficie de ataque de manera considerable.

Los resultados obtenidos permiten concluir que el procedimiento seguido en el laboratorio no solo identifica los riesgos presentes, sino que también valida la efectividad de las medidas correctivas aplicadas. La comparación cuantitativa y cualitativa de los análisis inicial y final confirma que la actualización y depuración de imágenes Docker es una práctica esencial para mantener contenedores seguros, minimizando la probabilidad de explotación de vulnerabilidades conocidas y asegurando un entorno confiable para pruebas o despliegues controlados. Asimismo, la experiencia práctica fortalece competencias en análisis de problemas, uso de herramientas de auditoría, diseño de soluciones de seguridad y toma de decisiones basadas en riesgos, alineándose con los objetivos del laboratorio y los resultados de aprendizaje definidos.

|  |
| --- |
| Referencias Bibliográficas |

* Boersma, N. (2015). Docker: Up & Running: Shipping Reliable Containers in Production. O'Reilly Media.
* Turnbull, J. (2014). The Docker Book: Containerization is the New Virtualization. James Turnbull.