KeepCoding Bootcamp Ciberseguridad | Edición IX

Informe Pentesting Android DragonBall

CONFIDENCIAL

Auditores:

Oscar Uriel Tobar Rios
David Groning Hernández
Andres Jesus Ricaurte Valera

Fecha del Informe: 19/06/2025

Contenido

Declaración de confidencialidad

Este documento es propiedad exclusiva de KeepCoding y el administrador de Android DragonBall. Este documento contiene información confidencial y de propiedad exclusiva. La duplicación, distribución o uso, total o parcial, en cualquier forma, requiere el consentimiento de KeepCoding y el administrador de Android DragonBall.

KeepCoding puede compartir este documento con auditores bajo acuerdos de confidencialidad para demostrar el cumplimiento de los requisitos de prueba de penetración.

Descargo de responsabilidad

Una prueba de penetración se considera una instantánea en el tiempo. Los hallazgos y recomendaciones reflejan la información recopilada durante la evaluación y no los cambios o modificaciones realizados fuera de ese período.

Los compromisos con límite de tiempo no permiten una evaluación completa de todos los controles de seguridad. KeepCoding priorizó la evaluación para identificar los controles de seguridad más débiles que un atacante podría explotar. KeepCoding recomienda realizar evaluaciones similares anualmente por parte de evaluadores internos o externos para garantizar el éxito continuo de los controles.

Información de Contacto

Nombre	Cargo	Correo
Android DragonBall		
	Administrador Android DragonBall	
KeepCoding		
Oscar Uriel Tobar Rios	Penetration Tester	oscartobarrios@gmail.com
David	Penetration Tester	davidgroninghernández@gmail.com
Andres	Penetration Tester	andresricv@outlook.es

1 Ámbito y Alcance de la Auditoría

- **Objetivo:** El objetivo de esta auditoria hacer un pestenting para el sitio collecto.es en el marco del proyecto de Pestesting del BootCamp de Ciberseguridad IX de Keepcoding, identificando las vulnerabilidades de este sitio
- **Alcance:** La auditoría se realiza sobre el sitio collecto.es que sirvió para hacer la auditoria.

Las fases de las actividades de pruebas de penetración incluyen las siguientes:

• Planificación: De acuerdo al Caso final práctico entregado se debe segun las reglas entregadas

• Descubrimiento: Se realizaron escaneos y enumeraciones para identificar posibles vulnerabilidades, áreas débiles y exploits con dispositivos android.

• Ataque: se Confirmaron vulnerabilidades potenciales mediante explotación y realice descubrimientos adicionales tras un nuevo acceso.

• Informes: Se documentaron todas las vulnerabilidades y exploits encontrados en el APK

---

2 Clasificación de los Hallazgos

La siguiente tabla define los niveles de gravedad y el rango de puntuación CVSS correspondiente que se utilizan en todo el documento para evaluar la vulnerabilidad y el impacto del riesgo.

Severidad	CVSS V3 Score Range	Definición
Crítico	9.0-10.0	La explotación es sencilla y suele provocar una vulneración a nivel del sistema. Se recomienda elaborar un plan de acción y aplicar el parche de inmediato.
Alto	7.0-8.9	La explotación es más difícil, pero podría provocar privilegios elevados y, potencialmente, una pérdida de datos o tiempo de inactividad. Se recomienda elaborar un plan de acción y aplicar el parche lo antes posible.
Moderado	4.0-6.9	Existen vulnerabilidades, pero no se pueden explotar ni requieren medidas adicionales, como ingeniería social. Se recomienda elaborar un plan de acción y aplicar parches después de que se hayan resuelto los problemas de alta prioridad.
Bajo	0.1-3.9	Las vulnerabilidades no se pueden explotar, pero reducirían la superficie de ataque de una organización. Se recomienda elaborar

		un plan de acción y aplicar parches durante la próxima ventana de mantenimiento.
Informativo	N/A	No existe vulnerabilidad. Se proporciona información adicional sobre elementos detectados durante las pruebas, controles estrictos y documentación adicional.

2.1 Risk Factors

El riesgo se mide por dos factores: probabilidad e impacto:

2.1.1 Probabilidad

La probabilidad mide la posibilidad de que se explote una vulnerabilidad. Las clasificaciones se otorgan en función de la dificultad del ataque, las herramientas disponibles, el nivel de habilidad del atacante y el entorno del cliente.

2.1.2 Impacto

El impacto mide el efecto de la vulnerabilidad potencial en las operaciones, incluida la confidencialidad, integridad y disponibilidad de los sistemas y/o datos del cliente, el daño a la reputación y la pérdida financiera.

3 Alcance

Evaluación	Detalles
Android DragonBall	APK

3.1 Exclusiones del Alcance

A solicitud de KeepCoding no realizó ninguno de los siguientes ataques durante las pruebas:

- Ataques Denial of Service (DoS)
- Phishing/Ingenieria Social

Todos los demás ataques no especificados anteriormente fueron permitidos por KeepCoding.

4 Información

4.1 Información de la Aplicación

Nombre de la aplicación: Android_DragonBall

Nombre del archivo: superapp.apk

Tamaño: 16,18 MB

Nombre del paquete: com.example.android_dragonball

MD5: e0e474b22a39b42272e067af558e483b

SHA1: 89e525d9435ff980c2bb14285e76dfd9dd20a33a

SHA256: 56209d60ecae22c63f05bffb6a92c4f76e3167e2ed787c2410d674782e9d8933

SDK mínimo: 26

SDK de destino: 33

Código de versión de Android: 1

Nombre de la versión de Android: 1.0

4.2 Componentes de la Aplicación

Actividades: 2

Servicios: 0

Receptores: 1

Proveedores: 1

Receptores exportados: 1

4.3 Información del Certificado

El binario está firmado.

Firma v1: Falso.

Firma v2: Verdadero.

Firma v3: Falso.

Firma v4: Falso.

Asunto X.509: CN=Android Debug, O=Android, C=US.

Algoritmo de firma: rsassa_pkcs1v15.

Válido desde: 2023-01-05 12:01:24+00:00

Válido hasta: 2052-12-28 12:01:24+00:00

Emisor: CN=Android Debug, O=Android, C=US

Número de serie: 0x1

Algoritmo hash: sha1

sha1: f68318690429d90856acd3e330c2b6c8c80d9781

md5: 053b4f9eeb205de9de5f87ba3da4b0ad

sha512: 535786a569185512d718c5b88b3566dbafaf0e572f935b4251a42907cc3c3d36
970e52b09d794942b13679feefe9d806704bba8f5dff9b3548e80c6b06f67ddd

sha256: 92148484959a479fdf60b55b3d8c8aa0db859b0941a4543141f7add4cf644a36

Se encontró 1 certificado único

Huella dactilar: 38eb86e621b2aa4d9b028c08ce1968d5f391c7593ff0d6f53116a6c02a167f71

---

5 Informe Ejecutivo

5.1 Breve Resumen del Proceso Realizado

Se evaluó la postura de seguridad interna de Android DragonBall través de pruebas de penetración del 29 de mayo al 18 de Junio de 2025. Las siguientes secciones brindan una descripción general de alto nivel de las vulnerabilidades descubiertas, los intentos exitosos y fallidos, y las fortalezas y debilidades.

La auditoría se realizó en un entorno controlado. Sobre el sitio Android DragonBall se hizo inicialmente un trabajo de Reconocimiento (Information Gathering) y luego se procedió a hacer un trabajo de reconocimiento de las vulnerabilidades solicitadas para este informe asi como su explotación.

5.2 Alcance y limitaciones de tiempo

Alcance y limitaciones de tiempo

El alcance durante el compromiso no permitió la denegación de servicio o la ingeniería social en todos los componentes de prueba.

Se establecieron limitaciones de tiempo para las pruebas. Se permitió la prueba de penetración de la red interna durante diez (10) días calendario.

5.3 Resumen de la prueba

Durante el análisis de seguridad de la aplicación Android_DragonBall (superapp.apk), se identificaron varias vulnerabilidades de severidad crítica, alta y moderada. Entre los principales hallazgos destacan el uso de certificados de depuración, algoritmos criptográficos inseguros, permisos mal configurados y exposición de información sensible. Se recomienda aplicar las medidas correctivas para mitigar los riesgos asociados antes de su despliegue en entornos productivos.

Puntuación de seguridad de la aplicación: 40/100 (RIESGO MEDIO)

5.4 Notas y recomendaciones del auditor

Durante el análisis de seguridad de la aplicación Android_DragonBall (superapp.apk), se identificaron múltiples vulnerabilidades que comprometen principios clave de seguridad como la confidencialidad, integridad y control de acceso. La mayoría de los hallazgos corresponden a prácticas inseguras comunes en versiones de desarrollo que no deben trasladarse al entorno de producción.

Recomendaciones generales:

1. **Firmado de la aplicación**: Se debe evitar el uso de certificados de depuración en versiones productivas. Asegúrese de firmar la APK con un certificado válido y seguro.

2. **Configuración de seguridad**: Establecer debuggable=false, allowBackup=false y exported=false en el archivo **AndroidManifest.xml** para reducir la superficie de ataque.

3. **Criptografía**: Sustituir algoritmos de hash vulnerables por estándares actuales como SHA-256 con claves de longitud adecuada.

4. **Compatibilidad con versiones obsoletas**: Incrementar el **minSdkVersion** a API nivel 29 (Android 10) o superior para beneficiarse de las mejoras de seguridad del sistema operativo.

5. **Protección de datos sensibles**: Evitar registrar o almacenar información confidencial sin cifrado fuerte. Revisar logs y archivos internos para eliminar credenciales codificadas o claves estáticas.

6. **Mejores prácticas de desarrollo seguro**: Aplicar controles contra ingeniería inversa y ofuscación de código para proteger la lógica interna de la aplicación.

5.5 Resumen de vulnerabilidades

Las siguientes tablas ilustran las vulnerabilidades encontradas por impacto y las soluciones recomendadas

0	3	4	1	1
Critico	Alta	Moderada	Baja	Informativa

Hallazgo	Severidad	Recomendación
H-001: Aplicación firmada con certificado de depuración	Alta	La aplicación de producción no debe enviarse con un certificado de depuración
H-002: El Algoritmo del certificado es vulnerable a colisiones de Hash	Alta	Se recomienda usar SHA256withRSA (clave mínima de 2048 bits) o SHA256withECDSA (clave mínima de 256 bits)
H-003: Depuración habilitada para la aplicación [android:debuggable=true]	Alta	Generar una nueva la versión final que esté firmada al ser publicada sea la que tiene debuggable=false
H-004: La aplicación se puede instalar en un Android vulnerable version 8.0, minSDK=26	Moderada	La aplicación debe ser compatible con una versión de Android 10 o superior, API 29 para obtener una seguridad razonable.
H-005: Se pueden realizar copias de seguridad de los datos de la aplicación [android:allowBackup=true]	Moderada	Editar el el archivo AndroidManifest.xml y establecerlo en android:allowBackup="false"
H-006: Receptor de transmisión está protegido por un permiso, pero se debe verificar el nivel de protección del permiso.	Moderada	Editar el el archivo AndroidManifest.xml y establecerlo en android:exported="false"

Hallazgo	Severidad	Recomendación
H-007: Los archivos pueden contener información confidencial codificada, como nombres de usuario, contraseñas, claves, etc	Moderada	Usar cifrado fuerte con claves gestionadas por el sistema y solucionar evitar que se permita ingenieria inversa en la aplicación
H-008: La aplicación registra información. Nunca se debe registrar información confidencial.	Baja	Identificar y eliminar registros inseguros en el log y evitar en produccion logs sensibles
H-009: Aplicación firmada	Informativa	

6 Hallazgos Técnicos

6.1 H-001: Aplicación firmada con certificado de depuración (Alta)

Descripción:	Aplicación firmada con un certificado de depuración.
Severidad:	Alta
Riesgo:	Probabilidad Baja
Sistema:	Android_DragonBall
Herramientas:	MOBSF
Referencias:	CVE-2011-2523

.

6.1.1 Evidencia

6.1.2 Recomendación

La aplicación de producción no debe enviarse con un certificado de depuración

6.2 H-002: El Algoritmo del certificado es vulnerable a colisiones de Hash (Alta)

Descripción:	La aplicación está firmada con SHA1 con RSA. Se sabe que el algoritmo hash SHA1 presenta problemas de colisión. Ya es **obsoleto y vulnerable**
Severidad:	Alta
Riesgo:	Probabilidad Alta
Sistema:	Android_DragonBall
Herramientas:	MOBSF
Referencias:	

6.2.1 Evidencia

6.2.2 Recomendación

Se recomienda usar SHA256withRSA (clave mínima de 2048 bits) o SHA256withECDSA (clave mínima de 256 bits)

Google Play no acepta desde 2019 aplicaciones nuevas firmadas con SHA1.

6.3 Hallazgo H-003: Depuración habilitada para la aplicación (Alto)

Descripción:	Se habilitó la depuración en la aplicación, lo que facilita a hacer ingenieria inversa al conectar un depurador. Esto permite volcar un seguimiento de la pila y acceder a las clases auxiliares de depuración
Severidad:	Alto
Riesgo:	Probabilidad Alta
Sistema:	Android_DragonBall
Herramientas:	MOBSF
Referencias:	

6.3.1 Evidencia

```
<application
  android:debuggable="true"
  ...>
```

6.3.2 Recomendación

Generar una nueva la versión final que esté firmada al ser publicada sea la que tiene debuggable=false, no usar APKs de tipo debug.

```
android {
  buildTypes {
    debug {
      debuggable true
    }
    release {
      minifyEnabled true
      shrinkResources true
      debuggable false    // 👈 Asegúrese de que esté explícitamente en false
      proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
    }
  }
}
```

6.4 Hallazgo H-004: La aplicación se puede instalar en un Android vulnerable version 8.0, minSDK=26 (Moderada)

Descripción:	Esta aplicación se puede instalar en una versión anterior de Android que tenga múltiples Vulnerabilidades.
Severidad:	Modedada
Riesgo:	Probabilidad Baja
Sistema:	Android_DragonBall

Herramientas:	MOBSF
Referencias:	CVE-2011-2523
Verificado:	

6.4.1 Evidencia

6.4.2 Recomendación

La aplicación debe ser compatible con una versión de Android 10 o superior, API 29 para obtener una seguridad razonable.

6.5 Hallazgo H-005: Se pueden realizar copias de seguridad de los datos de la aplicación (Moderada)

Descripción:	Tener `android:allowBackup="true"` en una aplicación Android de **producción** representa una vulnerabilidad de seguridad, ya que permite que los datos privados de la app se puedan copiar (backup) y restaurar, incluso en dispositivos comprometidos
Severidad:	Moderada
Riesgo:	Impacto Alto
Sistema:	Android_DragonBall
Herramientas:	MOBSF
Referencias:	

6.5.1 Evidencia

6.5.2 Recomendación

Editar el el archivo **AndroidManifest.xml** y establecer explícitamente asi:

<application

android:allowBackup="false"

android:fullBackupContent="false"

... >

</application>

Si no se define allowBackup, por defecto es true, así que debes definirlo manualmente como false.

6.6 Hallazgo H-006: Receptor de transmisión está protegido por un permiso, pero se debe verificar el nivel de protección del permiso. (Moderado)

Descripción:	Receptor de transmisión (androidx.profileinstaller.ProfileInstallReceiver) está protegido por un permiso, pero se debe verificar el nivel de protección del permiso. Permiso: android.permission.DUMP [android:exported=true]
Severidad:	Moderado
Riesgo:	Probabilidad baja
Sistema:	Android_DragonBall
Herramientas:	MOBSF
Referencias:	

6.6.1 Evidencia

Se ha descubierto que un receptor de difusión se comparte con otras aplicaciones del dispositivo, lo que lo deja accesible a cualquier otra aplicación. Está protegido por un permiso no definido en la aplicación analizada. Por lo tanto, se debe comprobar el nivel de protección

del permiso donde esté definido. Si se configura como normal o peligroso, una aplicación maliciosa puede solicitar y obtener el permiso e interactuar con el componente. Si se configura como de firma, solo las aplicaciones firmadas con el mismo certificado pueden obtener el permiso.

En el archivo AndroidManifest.xml (directo o por herencia de dependencias), aparece:

```
<receiver
  android:name="androidx.profileinstaller.ProfileInstallReceiver"
  android:exported="true"
  android:permission="android.permission.DUMP">
  <intent-filter>
    <action android:name="androidx.profileinstaller.action.INSTALL_PROFILE" />
  </intent-filter>
</receiver>
```

6.6.2 Recomendación

Se recomienda Deshabilitar la exportación si no se necesita este receptor ya que En la mayoría de aplicaciones no es necesario exportarlo, pues el propósito de ProfileInstallReceiver es optimizar el rendimiento con perfiles de compilación durante instalación

```
<receiver
  android:name="androidx.profileinstaller.ProfileInstallReceiver"
  android:exported="false"
 />
```

6.7 H-007: Los archivos pueden contener información confidencial codificada, como nombres de usuario, contraseñas, claves, etc (Moderada)

Descripción:	Los archivos pueden contener información confidencial codificada, como nombres de usuario, contraseñas, claves, etc

Severidad:	Moderado
Riesgo:	Impacto Alto
Sistema:	Android_DragonBall
Herramientas:	MOBSF
Referencias:	CWE: CWE-312: Almacenamiento de texto sin cifrar información sensible OWASP Top 10: M9: Ingeniería Reversa OWASP MASVS: MSTG-ALMACENAMIENTO-14

6.7.1 Evidencia

Información como tokens, credenciales, datos personales o claves se almacenan sin cifrado en:

- SharedPreferences
- Archivos locales (/data/data/...)
- Bases de datos SQLite

ademas la app puede ser fácilmente decompilada con herramientas como Jadx, apktool, o Frida.

com/bumptech/glide/load/Option.java com/bumptech/glide/load/engine/DataCacheKey.java
com/bumptech/glide/load/engine/EngineResource.java
com/bumptech/glide/load/engine/ResourceCacheKey.java
com/bumptech/glide/manager/RequestManagerRetriever.java

6.7.2 Recomendación

Usa cifrado fuerte con claves gestionadas por el sistema como EncryptedSharedPreferences (Android Jetpack Security) y Cifrado de archivos (EncryptedFile)

Y para solucionar ingeniería inversa se recomienda usar las siguientes acciones o herramientas para solucionarlo

- Ofuscación de y clases, métodos: usar ProGuard / R8
- Eliminación de nombres simbólicos: usar -dontusemixedcaseclassnames, -dontobfuscate
- Protección contra análisis: User SafetyNet o Play Integrity API
- Detección de herramientas: Usar Detección de Frida, Xposed, Magisk
- Integridad de firma: Usar Signature check en tiempo de ejecución

6.8 Hallazgo H-008: La aplicación registra información. Nunca se debe registrar información confidencial. (Baja)

Descripción:	La aplicación está registrando (haciendo logs) de **información sensible**, como contraseñas, tokens, números de tarjetas, datos personales, etc. En esta aplicacion los logs pueden ser accesibles a otras apps si el dispositivo está rooteado o comprometido.
Severidad:	Baja
Riesgo:	Impacto Medio
Sistema:	Android_DragonBall
Herramientas:	MOBSF
Referencias:	Información en el archivo de registro OWASP MASVS: MSTG-ALMACENAMIENTO-3 CWE: CWE-532: Inserción de información sensible

6.8.1 Evidencia

Los siguientes archivos usan logs de la aplicación

com/bumptech/glide/Glide.java
com/ bumptech/glide/disklrucache/DiskLruCache.java com/bumptech/glide/
gifdecoder/GifHeaderParser.java com/bumptech/glide/gifdecoder/
StandardGifDecoder.java
com/bumptech/glide/load/data/AssetPathFetcher.java
com/bumptech/ glide/load/data/HttpUrlFetcher.java com/bumptech/glide/load/data/
LocalUriFetcher.java com/bumptech/glide/load/data/mediastore/ ThumbFetcher.java
com/bumptech/glide/load/data/mediastore/ThumbnailStreamOpener.java
com/bumptech/glide/load/engine/DecodeJob.java
com/bumptech/glide/load/engine/DecodeJob.java
com/bumptech/glideload/engine/DecodePath.java com/bumptech/glide/load/engine/
Engine.java com/bumptech/glide/load/engine/GlideException.java
com/bumptech/glide/load/engine/SourceGenerator.java
com/bumptech/glide/load/engine/bitmap_recycle/LruA rrayPool.java com/bumptech/
glide/load/engine/ bitmap_recycle/LruB itmapPool.java
com/bumptech/glide/load/engine/cache/ DiskLruCache Wrapper.java
com/bumptech/glide/load/engine/cache/MemorySizeC alculator.java
com/bumptech/glide/load/engine/executor/GlideExecutor.java
com/bumptech/glide/load/engine/ prefill/BitmapPreFill Runner.java
com/bumptech/glide/load/model/ByteBufferEncoder.java
com/bumptech/glide/load/model/ByteBufferFileLoader.java
com/bumptech/glide/ load/ model/FileLoader.java com/bumptech/glide/load/model/ResourceLoader.java
com/bumptech/glide/load/model/StreamEncoder.java
com/bumptech/glide/load/resource/DefaultOnHeader/DecodedListener.java

com/bumptech/glide/load/ resource/bitmap/BitmapEncoder.java
com/bumptech/glide/load/resource/bitmap/BitmapImageDecoderResourceDecoder.java
com/bumptech/glide/load/resource/bitmap/ DefaultImageHeaderParser.java
com/bumptech/glide/load/resource/bitmap/ DownSampler.java
com/bumptech/glide/load/ resource/bitmap/DrawableToBitmapConverter.java com/bumptech/glide/load/ resource/bitmap/Hardware ConfigState.java com/bumptech/glide/load/resource/bitmap/Transform ationUtils.java
com/bumptech/glide/load/resource/bitmap/VideoDecoder.java
com/bumptech/ glide/load/ resource/gif/ByteBufferGifD ecoder.java
com/bumptech/glide/load/resource/gif/ GifDrawableEn coder.java
com/bumptech/glide/load/resource/gif/StreamGifDeco der.java
com/ bumptech/ glide/manager/DefaultConnectivityMonitorFactory.java com/bumptech/glide/ manager/ RequestManagerFragment.java com/bumptech/glide/manager/ RequestManagerRetriever.java
com/bumptech/glide/manager/RequestTracker.java
com/ bumptech/glide/manager/SingletonConnectivityReceiver.java
com/bumptech/ glide/ manager/SupportRequestManage rFragment.java com/bumptech/glide/ module/ManifestParser.java com/bumptech/glide/request/SingleRequest.java
com/bumptech/ glide/request/target/CustomViewTarget.java com/bumptech/glide/request/target/ ViewTarget.java
com/ bumptech/glide/signature/ApplicationVersionSign ature.java
com/ bumptech/glide/util/ContentLengthInputStream.java
com/bumptech/glide/util/pool/FactoryPools.java

6.8.2 Recomendación

Revisa el código en busca de líneas como:

```
Log.d("Login", "Usuario: " + usuario + ", Contraseña: " + password);
Log.e("Token", "Token recibido: " + token);
e.printStackTrace();  //  también puede mostrar información sensible
```

Se recomienda

- Nunca loguear contraseñas, tokens, o datos personales
- Usar logs solo para mensajes de diagnóstico controlado.
- En producción, desactivar los logs o haz que usen un nivel mínimo (WARN o ERROR).
- Usar una utilidad centralizada para logs (opcional y recomendado)Puedes crear una clase SafeLog para controlar qué se permite loguear
- Revisar de que en el *build.gradle* el modo release tenga habilitada la minificación y remoción de logs

6.9 Hallazgo H-009: Aplicación firmada (Informativa)

Descripción:	Aplicación está firmada con un certificado de firma de código. Todas las aplicaciones Android deben estar firmadas **digitalmente** con un certificado para poder instalarse. Este certificado:

	• **No necesita** ser emitido por una CA pública. • Usualmente es **autogenerado por el desarrollador**. • Se usa para **verificar la integridad y autoría** de la app. Por tanto, “estar firmada con un certificado de firma de código” no es malo por sí solo, pero puede tener implicaciones dependiendo del contexto.
Severidad:	Informativa
Riesgo:	Impacto Bajo
Sistema:	Android_DragonBall
Herramientas:	MOBSF
Referencias:	

6.9.1 Evidencia

Algoritmo hash: sha1

sha1: f68318690429d90856acd3e330c2b6c8c80d9781

6.9.2 Recomendación

- Usa SHA256withRSA o SHA256withECDSA y Evita SHA1withRSA, ya no es seguro. La clave debe tener al menos 2048 bits.

Usa Google Play App Signing

- Permite que Google gestione la clave de producción.
- Tú firmas con una clave de carga, y Google re-firma con la clave de producción real.

Beneficios de Google Play App Signing

- Puedes rotar claves si la pierdes.
- Más seguridad en la distribución.
- Activar desde: Google Play Console → Release → Setup → App integrity

7 Herramientas Utilizadas

- MORBSF: Se utilizo para revisar el APK
- Burp proxy: Capturas de trafico y modificación de los request al servidor