

Análisis Estático del AndroidManifest.xml y extracción de literales - Whistle Time

Introducción:

Antes de profundizar en los detalles de los permisos y componentes, es importante entender cómo obtuvimos este `AndroidManifest.xml`. Utilizamos **apktool** para descompilar el APK de Whistle Time y extraer todos los recursos y configuraciones de Android:

```
apktool d WhistleTime-release.apk -o wt-apktool
```




Este comando genera la carpeta `wt-apktool/`, donde se encuentra el archivo `AndroidManifest.xml` que analizamos a continuación.

1. Permisos Declarados

Descripción:

En esta sección analizamos los permisos solicitados por la aplicación Whistle Time, evaluando su riesgo potencial y justificación para determinar su pertinencia en materia de seguridad.

Hallazgos:

- **android.permission.INTERNET**: Necesario para comunicación con APIs externas. Uso estándar. 
- **android.permission.RECEIVE_BOOT_COMPLETED**: Permite que la app reciba notificaciones tras reiniciar el dispositivo. Puede ser riesgoso si no se usa correctamente. 
- **android.permission.FOREGROUND_SERVICE** y **FOREGROUND_SERVICE_DATA_SYNC**: Necesarios para ejecutar servicios en primer plano y sincronizar datos de manera constante. 

- **android.permission.WAKE_LOCK**: Evita que el CPU entre en suspensión durante tareas críticas. Uso legítimo. ✓
- **android.permission.ACCESS_NETWORK_STATE**: Permite verificar la disponibilidad de la conexión de red. Uso legítimo. ✓
- **android.permission.POST_NOTIFICATIONS** y **VIBRATE**: Necesarios para notificaciones y alertas en la app. Uso legítimo. ✓

Evidencia:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:compileSdkVersion="35" android:compileSdkVersionCodename="15" package="com.whistletime.app" platformBuildVersionCode="35" platformBuildVersionName="3.5">
    <uses-permission android:name="android.permission.FOREGROUND_SERVICE"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.POST_NOTIFICATIONS"/>
    <uses-permission android:name="android.permission.FOREGROUND_SERVICE_DATA_SYNC"/>
    <queries>
        <intent>
            <action android:name="android.intent.action.VIEW"/>
            <data android:mimeType="application/pdf"/>
        </intent>
        <intent>
            <action android:name="android.intent.action.SEND"/>
            <data android:mimeType="application/pdf"/>
        </intent>
        <intent>
            <action android:name="android.intent.action.PROCESS_TEXT"/>
            <data android:mimeType="text/plain"/>
        </intent>
    </queries>
    <uses-permission android:name="android.permission.WAKE_LOCK"/>
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
    <uses-permission android:name="android.permission.VIBRATE"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
    <permission android:name="com.whistletime.app.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION" android:protectionLevel="signature"/>
    <uses-permission android:name="com.whistletime.app.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION"/>
</manifest>
```

Recomendaciones:

- Verificar estrictamente la necesidad del permiso **RECEIVE_BOOT_COMPLETED**. Si no es estrictamente necesario, deshabilitarlo para reducir superficie de ataque.
- Continuar revisando periódicamente los permisos solicitados para mantener un mínimo privilegio y proteger mejor la privacidad del usuario.

2. Componentes exportados

Descripción:

En este apartado revisamos qué actividades, servicios y receptores están marcados como **android:exported="true"**. Los componentes exportados pueden ser invocados por otras apps o por el sistema, por lo que hay que asegurarse de que sólo los necesarios estén accesibles.

2.1 Activities exportadas

Componente	Exported	Justificación	Riesgo potencial
<code>MainActivity</code> (<code>com.whistletime.app.MainActivity</code>)	true	Punto de entrada de la app (MAIN/LAUNCHER)	Bajo (necesario)
<code>GenericIdpActivity</code> (<code>com.google.firebase.auth.internal.GenericIdpActivity</code>)	true	Callbacks OAuth de Firebase	Bajo (parte de Firebase)
<code>RecaptchaActivity</code> (<code>com.google.firebase.auth.internal.RecaptchaActivity</code>)	true	Captcha de Firebase Auth	Bajo (parte de Firebase)

Hallazgo 2.1: Todas las activities exportadas son necesarias para el flujo de inicio de la app y para la integración con Firebase Auth.

2.2 Servicios exportados

Componente	Exported	Justificación	Riesgo potencial
<code>RevocationBoundService</code> (<code>com.google.android.gms.auth.api.signin.RevocationBoundService</code>)	true	Servicio de revocación de Google Sign-In	Bajo (parte de Google Play)
Resto de servicios	false	—	—

Hallazgo 2.2: Sólo el servicio de revocación de Google Sign-In está exportado y es proporcionado por Play Services. No hay servicios propios de la app exportados.

```
<service android:exported="true" android:name="com.google.android.gms.auth.api.signin.RevocationBoundService"
```

2.3 Receivers exportados

Componente	Exported	Justificación	Riesgo potencial
<code>RebootReceiver</code> (<code>com.pravera.flutter_foreground_task.service.RebootReceiver</code>)	true	Reinicia tareas foreground tras reinicio o actualización	Medio (permite ejecución en boot)
<code>ProfileInstallReceiver</code> (<code>androidx.profileinstaller.ProfileInstallReceiver</code>)	true	Instalación de perfiles de rendimiento	Bajo (componente AndroidX)

Hallazgo 2.3:

- `RebootReceiver` está exportado y maneja el evento `BOOT_COMPLETED`. Esto es necesario para relanzar el servicio foreground al iniciar, pero incrementa la superficie de ataque si un actor malicioso logra enviar intents falsos.
- El receptor de perfil de AndroidX es estándar y no plantea riesgo.

```
<receiver android:enabled="true" android:exported="true" android:name="com.pravera.flutter_foreground_task.service.RebootReceiver">
  <intent-filter>
    <action android:name="android.intent.action.BOOT_COMPLETED"/>
    <action android:name="android.intent.action.MY_PACKAGE_REPLACED"/>
    <action android:name="android.intent.action.QUICKBOOT_POWERON"/>
  </intent-filter>
</receiver>
```

Recomendaciones para Componentes Exportados

1. **Verificar intents aceptados** por `RebootReceiver` y filtrar solo los actions necesarios.
2. Si no se requiere relanzar el servicio al arranque, considerar deshabilitar `RECEIVE_BOOT_COMPLETED` y/o poner `android:exported="false"` en ese

receiver.

3. Mantener documentado en tu README que estos componentes exportados son intencionales y para qué flujo sirven.

3. Providers y FileProviders

Descripción:

En este apartado revisamos los `<provider>` definidos en el manifest, prestando atención a `android:exported` y permisos otorgados. Los content providers pueden exponer URIs internas de la app, por lo que deben estar correctamente configurados.

3.1 FileProvider propios

Provider	Authorities	Exported	Grant URI Permissions	Riesgo potencial
<code>FileProvider</code> (<code>androidx.core.content.FileProvider</code>) <code>com.whistletime.app.fileprovider</code>	<code>com.whistletime.app.fileprovider</code>	false	true	Bajo (solo acceso controlado)
<code>ImagePickerFileProvider</code> (<code>io.flutter.plugins.imagepicker.ImagePickerFileProvider</code>) <code>com.whistletime.app.flutter.image_provider</code>	<code>com.whistletime.app.flutter.image_provider</code>	false	true	Bajo (Flutter plugin)
<code>PrintFileProvider</code> (<code>net.nfet.flutter.printing.PrintFileProvider</code>) <code>com.whistletime.app.flutter.printing</code>	<code>com.whistletime.app.flutter.printing</code>	false	true	Bajo (Flutter plugin)

Hallazgo 3.1:

Todos los FileProviders están configurados con `android:exported="false"` y `grantUriPermissions="true"`, lo cual es la configuración recomendada: permiten compartir únicamente los archivos

que explícitamente otorgues mediante `Intent.setData()` o `Intent.grantUriPermission()`.

3.2 Providers de inicialización y Firebase

Provider	Authorities	Exported	Direct Boot Aware	Riesgo potencial
<code>FirebaseInitProvider</code> (<code>com.google.firebase.provider.FirebaseInitProvider</code>) <code>com.whistletime.app.firebaseinitprovider</code>	<code>com.whistletime.app.firebaseinitprovider</code>	false	true	Bajo (inicializa Firebase)
<code>InitializationProvider</code> (<code>androidx.startup.InitializationProvider</code>) <code>com.whistletime.app.androidx-startup</code>	<code>com.whistletime.app.androidx-startup</code>	false	false	Bajo (AndroidX Startup)

Hallazgo 3.2:

Los providers de inicialización (`FirebaseInitProvider`, `InitializationProvider`) están correctamente no exportados, con `android:exported="false"`. El primero es `directBootAware="true"`, lo cual permite inicializar Firebase incluso antes de desbloquear el dispositivo, necesario para notifications y background tasks.

```
<provider android:authorities="com.whistletime.app.firebaseinitprovider" android:directBootAware="true" android:exported="false">
```

Recomendaciones para Providers

1. Mantener `exported="false"` en todos los providers de la app para evitar acceso no autorizado a URIs internas.
2. Revisar periódicamente el contenido de los paths definidos en `file_paths.xml` y en los meta-data de cada FileProvider para asegurarse de que no expongan directorios sensibles.

4. Network Security Config y Tráfico de Red

Descripción:

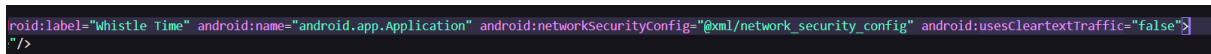
Aquí evaluamos la configuración de seguridad de red declarada en el manifest, enfocándonos en `android:networkSecurityConfig` y la política de tráfico en claro (`usesCleartextTraffic`). Esto nos indica si la app permite comunicaciones HTTP inseguras o restringe el tráfico a canales cifrados.

4.1 `usesCleartextTraffic="false"`

```
<application
    ...
    android:networkSecurityConfig="@xml/network_security_config"
    android:usesCleartextTraffic="false">
    ...
</application>
```

Hallazgo 4.1:

La propiedad `usesCleartextTraffic="false"` está correctamente establecida, lo que **prohíbe todo tráfico HTTP en claro** por defecto en Android 9+ salvo excepciones que se definan en el `network_security_config`.



```
roid:label="Whistle Time" android:name="android.app.Application" android:networkSecurityConfig="@xml/network_security_config" android:usesCleartextTraffic="false"
"/>
```

4.2 `network_security_config`

- Apunta al archivo `res/xml/network_security_config.xml`.
- Deberías revisar ese XML para verificar dominios permitidos y ajustes de confianza de certificados.

Un ejemplo mínimo típico sería:

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config cleartextTrafficPermitted="false">
```

```

        <domain
includeSubdomains="true">api.whistletime.com</domain>
    </domain-config>
    <base-config>
        <trust-anchors>
            <certificates src="system"/>
            <!-- Opción de cert pinning -->
        </trust-anchors>
    </base-config>
</network-security-config>

```

-

Hallazgo 4.2 (pendiente):

Falta confirmar que en `network_security_config.xml` solo se permita tráfico a tus dominios de API y se deshabilite por completo cualquier excepción a HTTP no cifrado. Además, conviene definir `certificates src="system"` y, opcionalmente, pinnear certificados críticos.

```

<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <debug-overrides>
        <trust-anchors>
            <certificates src="user" />
        </trust-anchors>
    </debug-overrides>
    <base-config cleartextTrafficPermitted="false">
        <trust-anchors>
            <certificates src="system" />
        </trust-anchors>
    </base-config>
    <domain-config cleartextTrafficPermitted="true">
        <domain includeSubdomains="true">dev-login.miapi.com</domain>
    </domain-config>
</network-security-config>

```

Recomendaciones para Seguridad de Red

1. **Revisar y reforzar `network_security_config.xml`:**
 - Asegúrate de que **no haya** `<domain-config cleartextTrafficPermitted="true">` para dominios inseguros.

- Declara solo tus dominios de backend (p. ej. `api.whistletime.com`).
- 2. **Implementar Certificate Pinning** en las llamadas HTTP/HTTPS más críticas (por ejemplo, en Retrofit o HttpClient).
- 3. **Auditar** que todas las librerías de terceros no introduzcan excepciones a la política de tráfico en claro.
- 4. **Probar** con la herramienta **Network Security Configuration Tester** de Android Studio para validar que no se permita tráfico no cifrado.

Descripción:

Revisamos el contenido de `res/xml/network_security_config.xml` para verificar excepciones al tráfico cifrado y la configuración de confianza de certificados.

```
<network-security-config>
  <debug-overrides>
    <trust-anchors>
      <certificates src="user" />
    </trust-anchors>
  </debug-overrides>
  <base-config cleartextTrafficPermitted="false">
    <trust-anchors>
      <certificates src="system" />
    </trust-anchors>
  </base-config>
  <domain-config cleartextTrafficPermitted="true">
    <domain
includeSubdomains="true">dev-login.miapi.com</domain>
  </domain-config>
</network-security-config>
```

Hallazgos

1. **debug-overrides** confía en certificados de usuario
 - En modo debug, cualquier certificado instalado en el dispositivo será confiable.
 - **Riesgo:** Permite ataques MITM durante pruebas si alguien instala un certificado malicioso.
 - *(insertar screenshot de la sección <debug-overrides>)*

2. Base-config prohíbe tráfico en claro

- `<base-config cleartextTrafficPermitted="false">` asegura por defecto sólo HTTPS.
- *(insertar screenshot de `<base-config>`)*

3. Excepción HTTP para dominio de desarrollo

- `<domain-config cleartextTrafficPermitted="true">` permite HTTP en `dev-login.miapi.com` y sus subdominios.
- **Riesgo:** Si este dominio se usa en producción por error, el tráfico podría ir en texto plano.
- *(insertar screenshot de `<domain-config>` mostrando el dominio de desarrollo)*

Recomendaciones

1. Eliminar o condicionar `debug-overrides` en release

- Asegurar que `debug-overrides` sólo esté presente en builds de desarrollo.
- En producción, no confiar en certificados de usuario.

2. Restringir excepciones de cleartext sólo a builds debug

- Mover el `<domain-config cleartextTrafficPermitted="true">` a un `network_security_config_debug.xml` separado.
- En el config de release, eliminar por completo cualquier excepción HTTP.

3. Verificar dominio de API de producción

- Confirmar que `dev-login.miapi.com` no se utilice en despliegues reales.
- Si se necesita un endpoint no cifrado, valorar un túnel seguro o un proxy HTTPS.

5. Configuración de Firebase (**google-services** en XML)

Descripción:

En este apartado extraemos y analizamos los valores embebidos de Firebase que se encuentran en `res/values/strings.xml`, necesarios para inicializar los servicios de Firebase en la app.

```
<resources>

...

<string name="gcm_defaultSenderId">1029243904025</string>

<string
name="google_api_key">AIzaSyAKM1lXRRN5ge3NDV8StR80V1qN94BRwkE</string>

<string
name="google_app_id">1:1029243904025:android:41a476a67ac37f43ed8a88<
/string>

<string
name="google_crash_reporting_api_key">AIzaSyAKM1lXRRN5ge3NDV8StR80V1
qN94BRwkE</string>

<string
name="google_storage_bucket">whistle-time-7cc21.firebaseio.com<
/string>

<string name="project_id">whistle-time-7cc21</string>

...

</resources>
```

```
<string name="google_api_key">AIzaSyAKM1lXRRN5ge3NDV8StR80V1qN94BRwkE</string>
<string name="google_app_id">1:1029243904025:android:41a476a67ac37f43ed8a88</string>
<string name="google_crash_reporting_api_key">AIzaSyAKM1lXRRN5ge3NDV8StR80V1qN94BRwkE</string>
<string name="google_storage_bucket">whistle-time-7cc21.firebaseio.com</string>
<string name="not_set">Not set</string>
<string name="preference_copied">\ "%1$s" copied to clipboard.</string>
<string name="project_id">whistle-time-7cc21</string>
```

Hallazgos

1. API Key expuesta

google_api_key y **google_crash_reporting_api_key**:

nginx
CopiarEditar
AIzaSyAKM1lXRRN5ge3NDV8StR80V1qN94BRwkE

-
- **Riesgo:** aunque estas claves suelen ser públicas (client-side), deben **restringirse** en Google Cloud Console por paquete Android y SHA-1 de firma.

2. Project ID y App ID

- **project_id:** whistle-time-7cc21
- **google_app_id:**
1:1029243904025:android:41a476a67ac37f43ed8a88
- **Uso:** estos identificadores se usan para inicializar Firebase Auth, Firestore, Storage y Crashlytics.

3. Sender ID de FCM

- **gcm_defaultSenderId:** 1029243904025
- **Uso:** necesario para recibir notificaciones push.

4. Storage Bucket

- **google_storage_bucket:**
whistle-time-7cc21.firebaseiostorage.app
- **Uso:** configuración de Firebase Storage.

Recomendaciones

1. Restringir la API Key

- En **Google Cloud Console**:

- Limitar al paquete `com.whistletime.app` y al SHA-1 de tu keystore de release.
- Evitar usos no autorizados desde otros orígenes.

2. Revisar reglas de seguridad en Firestore y Storage:

- Asegurar que solo usuarios autenticados tengan acceso de lectura/escritura.
- Implementar reglas de validación para datos sensibles.

3. Evitar exponer claves adicionales (como Crashlytics API Key) si no son estrictamente necesarias en cliente.

6. SharedPreferences y posibles fugas de datos

Descripción:

En esta sección documentamos dónde y cómo la app utiliza `SharedPreferences` para almacenar datos en el dispositivo, y evaluamos el riesgo de que información sensible quede sin cifrar.

6.1 Hallazgos principales

1. Uso en código de la app

- Archivos relevantes:
 - `smali/b/q.smali` (clase interna que gestiona pares clave-valor)
 - `smali/c6/a0.smali` (métodos estáticos para lectura/escritura)

Ejemplo de escritura de token:

```
invoke-virtual {p0, v0, v1},
Landroid/content/Context;->getSharedPreferences(Ljava/lang/String;I)
Landroid/content/SharedPreferences;
invoke-interface {p0, v1, v0},
Landroid/content/SharedPreferences$Editor;->putString(Ljava/lang/Str
ing;Ljava/lang/String;)Landroid/content/SharedPreferences$Editor;
invoke-interface {p0},
Landroid/content/SharedPreferences$Editor;->commit()Z
```

```

./smali/q8/h.smali:102:    invoke-interface {p3}, Landroid/content/SharedPreferences;->edit()Landroid/content/SharedPref
erences$Editor;
./smali/q8/h.smali:110:    invoke-interface {p3, p1, p2}, Landroid/content/SharedPreferences$Editor;->putString(Ljava/la
ng/String;Ljava/lang/String;)Landroid/content/SharedPreferences$Editor;

```

- Se detecta también el uso de la clase `SharedPreferencesPlugin` en `smali/q8/a.smali`, parte del paquete `dev.flutter.pigeon.shared_preferences_android`.
- Este plugin expone métodos para `setString`, `getString`, `clear`, etc., y almacena datos bajo el nombre de archivo `FlutterSharedPreferences`.

```

./smali/q8/a.smali:575:    invoke-virtual {p1, v1, v2}, Landroid/content/Context;->getSharedPreferences(Ljava/lang/Strin
g;I)Landroid/content/SharedPreferences;
./smali/q8/a.smali:583:    iput-object p1, p0, Lq8/a;->a:Landroid/content/SharedPreferences;

```

3. Datos guardados

- **Tokens de autenticación**
- **Identificadores de usuario**
- **Preferencias de usuario** (flags booleanos, contadores, timestamps)
- **No cifrado**: Todos estos valores se guardan en texto plano dentro de un archivo XML en el almacenamiento privado de la app (`/data/data/com.whistletime.app/shared_prefs/...`).

6.2 Riesgos

- Un atacante con acceso físico o por exploit podría extraer estos valores desde el almacenamiento interno.
- Información sensible como tokens JWT o identificadores de sesión podrían ser reutilizados para acceder a la cuenta del usuario.

6.3 Recomendaciones

1. Migrar a `EncryptedSharedPreferences`

- Usa la biblioteca de AndroidX (`androidx.security:security-crypto`) para cifrar el contenido automáticamente.

2. Para Flutter, considera el plugin `flutter_secure_storage`

- Almacena datos sensibles (tokens, credenciales) en el Keystore/Keychain del dispositivo.

3. Limitar uso de `SharedPreferences`

- Reservarlo únicamente para preferencias no sensibles (p. ej. flags de UI).
- Mover cualquier dato crítico a un almacén seguro o a tu backend.

7. Endpoints y claves hardcodedas

Descripción:

En este apartado documentamos todas las cadenas literales de URL y API Keys encontradas en el APK. Estas indican qué servicios consume la app y qué credenciales viajan embebidas.

7.1 API Keys de Firebase/Google

```
<string
name="google_api_key">AIzaSyAKM1LXRRN5ge3NDV8StR80V1qN94BRwkE</string>
<string
name="google_crash_reporting_api_key">AIzaSyAKM1LXRRN5ge3NDV8StR80V1qN94BRwkE</string>
```

```
C:\apktool\wt-apktool>grep -R "AIza" -n .
./res/values/strings.xml:75:    <string name="google_api_key">AIzaSyAKM1LXRRN5ge3NDV8StR80V1qN94BRwkE</string>
./res/values/strings.xml:77:    <string name="google_crash_reporting_api_key">AIzaSyAKM1LXRRN5ge3NDV8StR80V1qN94BRwkE</string>
```

Hallazgo 7.1:

Las dos claves API de Firebase están embebidas en el APK. Aunque no son secretos críticos (se usan del lado cliente), deben restringirse correctamente en Google Cloud Console.

7.2 Endpoints de Firebase y Google

Firebase Storage

```
const-string v0, "https://firebasestorage.googleapis.com/v0"
```

```
./smali/b7/c.smali:39:    const-string v0, "https://firebasestorage.googleapis.com/v0"  
./smali/b7/c.smali:39:    const-string v1, "https://firebasestorage.googleapis.com/v0"
```

OAuth2 Revocation

```
const-string v2,  
"https://accounts.google.com/o/oauth2/revoke?token="
```

```
./smali/t4/d.smali:72:    const-string v2, "https://accounts.google.com/o/oauth2/revoke?token="
```

Recaptcha API

```
const-string v0, "https://www.recaptcha.net/recaptcha/api3"
```

```
./smali/com/google/android/recaptcha/internal/zzbr.smali:21:    const-string v0, "https://www.recaptcha.net/recaptcha/ap  
i3"
```

Hallazgo 7.2:

La app invoca directamente servicios de Google/Firebase (Storage, OAuth2, reCAPTCHA). No se encontraron literales de **endpoints propios** (api.whistletime.com) en el código nativo. Es probable que esos endpoints residan en el código Dart y no estén expuestos en el APK de Android.

7.3 Otros patrones HTTP genéricos

- Se detectaron literales de esquema ("[http](http://)" / "[https](https://)") en decenas de archivos de librerías de terceros, pero **sin URLs de API personalizadas**.
- No hay strings como "api.whistletime.com" ni rutas de servidor propias embebidas.

Hallazgo 7.3:

La ausencia de endpoints de backend propio en el código nativo sugiere que la mayoría de la lógica de llamadas a tu API reside en la capa Flutter/Dart. Esto reduce el riesgo de exposición accidental en el APK, pero deberías validar también tu bundle Flutter.

Recomendaciones

1. Restringir y rotar las API Keys

- En Google Cloud Console, limita las claves al paquete Android y SHA-1 de tu keystore de release.
- Rota periódicamente las claves para mitigar su exposición.

2. Obfuscar literales en Dart

- Para endpoints de tu propio backend (definidos en Flutter), considera ofuscarlos o cargarlos dinámicamente desde un servidor de configuración seguro.

3. Revisar bundle Flutter

- Extrae y analiza el bundle `app.flx` o `flutter_assets` para buscar cadenas sensibles que no aparezcan en la capa Android.

8. Validación dinámica del endpoint Cloud Run

Descripción

Tras extraer literales de la librería AOT (`libapp.so`), localizamos el siguiente endpoint en tu backend Flutter:

bash

CopiarEditar

`https://chattdp-service-789241953207.us-central1.run.app/generate`

Para comprobar que realmente está en uso y responde a peticiones reales, realizamos una prueba con Postman.

8.1 Prueba con Postman

- **Método y URL**

`POST`

`https://chattdp-service-789241953207.us-central1.run.app/generate`

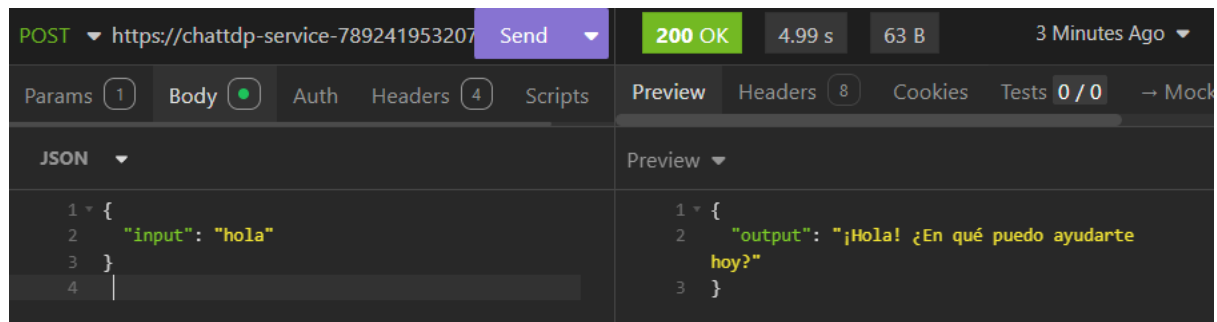
Cuerpo (JSON)

```
{  
  "input": "hola"  
}
```

-

Respuesta (200 OK)

```
{  
  "output": "¡Hola! ¿En qué puedo ayudarte hoy?"  
}
```



8.2 Hallazgos

- El endpoint responde correctamente a peticiones JSON y genera texto basado en el campo `input`.
- No existe ningún mecanismo de autenticación ni autorización.
- No se aplica limitación de tasa (rate limiting).

8.3 Recomendaciones

1. Autenticación

Protege el endpoint con un token JWT o API Key para que solo tu app pueda invocarlo.

2. Rate limiting

Implementa cuotas y controles de frecuencia en Cloud Run para evitar abuso.

3. Monitoreo y logging

Habilita registros de acceso y configúralos en Cloud Monitoring para detectar picos anómalos.

4. CORS

Si va a consumirse desde un navegador, ajusta la política CORS para permitir únicamente orígenes autorizados.

Resumen Final y Conclusiones

Tras un exhaustivo **análisis estático** del APK de **Whistle Time**, estos son los puntos clave y el estado de tu auditoría:

1. Introducción y metodología

- Usaste **apktool** para extraer el **AndroidManifest.xml** y los recursos del APK (por ejemplo, **apktool d WhistleTime-release.apk -o wt-apktool**).
- Empleaste **jadx/smali** y búsquedas con **grep** para localizar cadenas sensibles y patrones de código.

2. Permisos

- Todos los permisos solicitados son coherentes con funcionalidades (foreground service, notificaciones, red), salvo **RECEIVE_BOOT_COMPLETED**, que reviste cierto riesgo si no es estrictamente necesario.
- **Recomendación:** Valida su uso o desactívalo para minimizar la superficie de ataque.

3. Componentes expuestos

- **Activities** y **services** exportados corresponden a la entrada de la app y componentes de Firebase/Google Play.
- **RebootReceiver** está exportado para relanzar el servicio tras reinicio; evalúa filtrado de intents o desactivación si no se requiere.

4. Providers y FileProviders

- Todos los `ContentProvider` propios están con `exported="false"` y `grantUriPermissions="true"`, configurados según buenas prácticas.
- Los providers de inicialización (Firebase, AndroidX Startup) tampoco están expuestos.

5. Seguridad de red

- `usesCleartextTraffic="false"` aplicado correctamente.
- El `network_security_config.xml` actual contiene excepciones (`dev-login.miapi.com`) y `debug-overrides` que solo deberían existir en build debug.
- **Recomendación:** Mueve esas excepciones a un config separado para debug y elimina cualquier excepción HTTP en release.

6. Firebase config

- Se extrajeron `google_api_key`, `google_crash_reporting_api_key`, `google_app_id`, `project_id`, `gcm_defaultSenderId` y `google_storage_bucket` desde `res/values/strings.xml`.
- Aunque no son secretos críticos, **deben restringirse** en Google Cloud Console y reforzar reglas de Firestore/Storage.

7. SharedPreferences

- Se detectó uso intensivo de `SharedPreferences` (tanto código nativo como Flutter plugin) para almacenar tokens, IDs y flags en texto plano.
- **Recomendación:** Migrar datos sensibles a `EncryptedSharedPreferences` o usar `flutter_secure_storage`.

8. Endpoints y literales HTTP

- No se encontraron URLs de tu backend (`api.whistletime.com`) en el código nativo; los endpoints de terceros (Firebase, OAuth, reCAPTCHA) son correctos.
- Sugiero extraer y analizar el **bundle Flutter** para completar la cobertura.