



Universidad
Francisco de
Vitoria
*Centro de
Documentación
Europea*
UFV Madrid

Analizando discord 8.5.5

Héctor Ramírez López

índice

1.Introducción.....	3
2.Cronología	5
3.Resultados	17
4.Recomendaciones	18
1: Exposición de Spotify Client Secret (CRÍTICO)	18
2: Google API Keys Hardcodeadas (MEDIO)	19
3: Clave Pública de reCAPTCHA (Baja)	19
5. Conclusiones Generales (Mejores Prácticas)	20
6. Prueba de Secret spotify con (Curl).....	21

1. Introducción

Para la realización de esta práctica de análisis de código, he seleccionado la aplicación oficial de **Discord** para Android en lugar de entornos vulnerables como DIVA o insecure bank. La he elegido porque quería evaluar un entorno de producción real que tuviera alguna dificultad.

Para llevarla a cabo he utilizado un móvil físico

Marca: Redmi

Modelo: Note 9 pro

Versión de MIUI: 14.0.5

Versión de Android: 12 skq1.211019.001

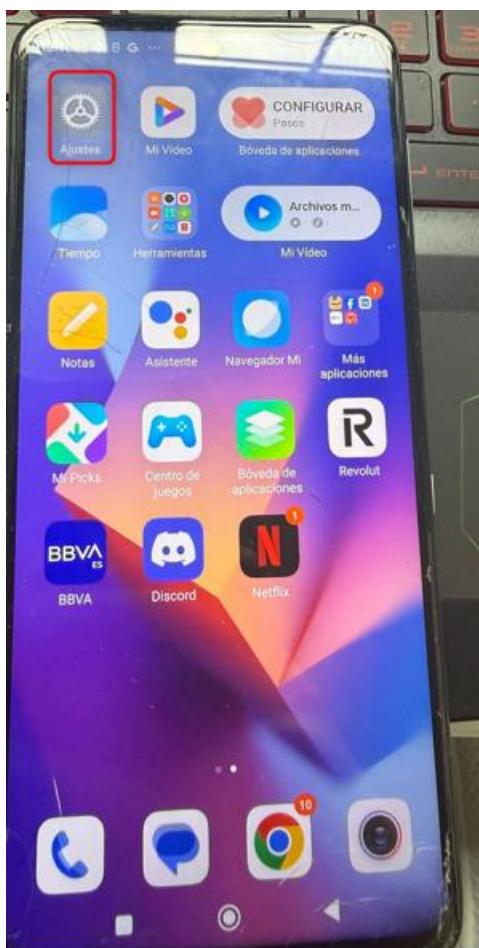


2.Cronología

El primer paso a realizar sería conectar el móvil al ordenador mediante el cable USB en este caso un cable tipo C



Tendremos que habilitar las opciones de desarrollador para que luego no nos de problemas. Para ello iremos a ajustes-->sobre el teléfono y pulsamos varias veces en versión de MIUI hasta que nos aparezca un mensaje que dice que las opciones para desarrolladores ya han sido activadas.

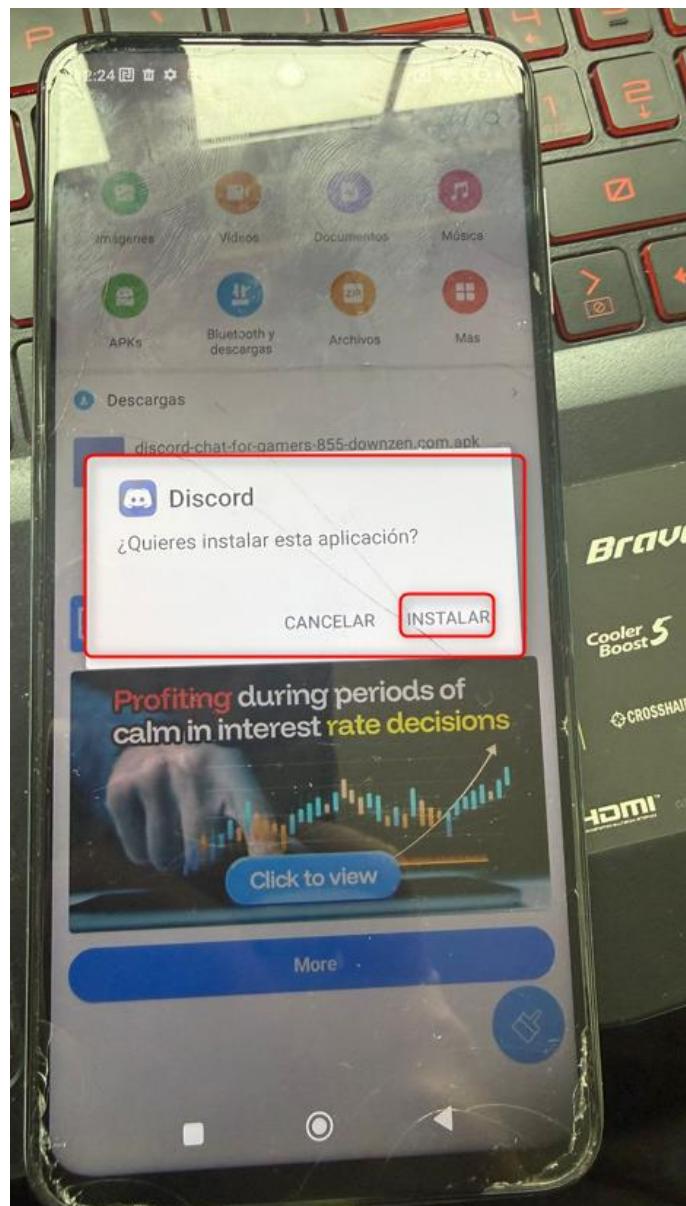


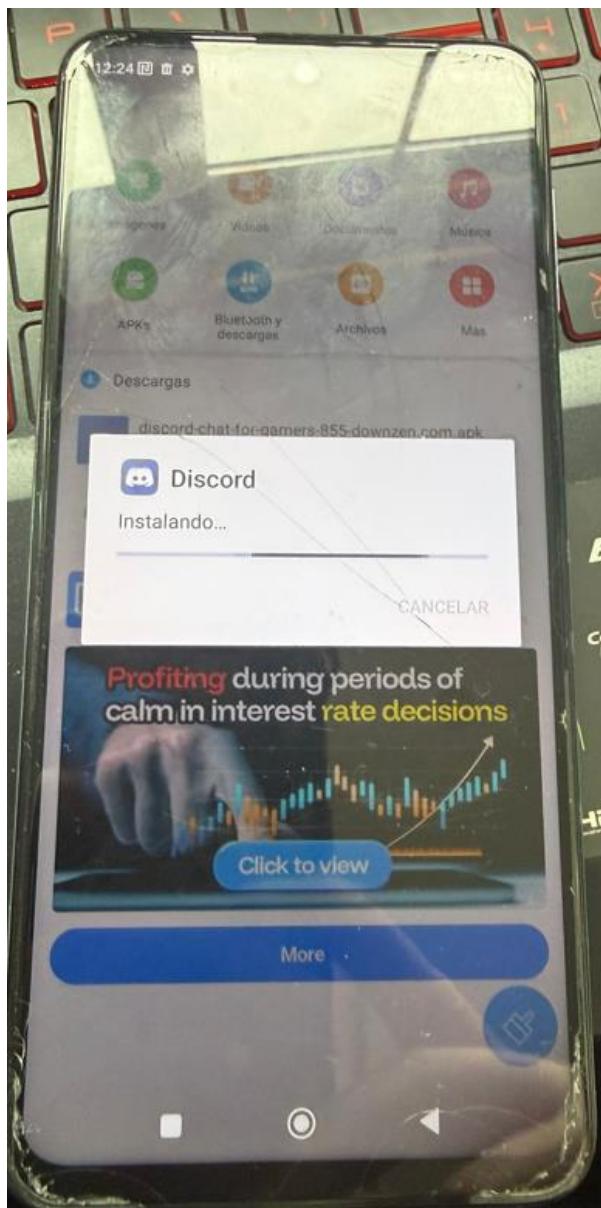






El siguiente paso sería ir a Google y descargar desde ahí la versión 8.5.5 de discord que es la que vamos a analizar en este caso la descargaremos de apkmirror y procedemos a instalarla.





El siguiente paso ya lo realizaremos desde la consola cmd del equipo, hay que tener adb instalado en el equipo y tener activada la depuración USB en las opciones de desarrollador del móvil.

Comprobaremos que la conexión entre el ordenador y el móvil se ha establecido correctamente para ello ejecutaremos en cmd el comando **adb devices**, como podemos ver en la imagen si lo detecta.

```
Microsoft Windows [Versión 10.0.19045.6466]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\rami>adb devices
* daemon not running; starting now at tcp:5037
* daemon started successfully
List of devices attached
92caeef0      device
```

A continuación, listaremos con la herramienta adb todos los paquetes de las aplicaciones que hay instaladas en el dispositivo móvil, usaremos el comando **adb shell pm list packages**, nos saldrá una lista de todas las aplicaciones instaladas.

```
C:\Users\rami>adb shell pm list packages
package:com.miui.screenrecorder
package:com.google.android.networkstack.tethering
package:com.amazon.mShop.android.shopping
package:com.google.android.apps.subscriptions.red
package:com.android.cts.priv.ctsshim
package:com.google.android.youtube
package:com.qualcomm.qti.qcolor
package:com.android.internal.display.cutout.emulation.corner
package:com.google.android.ext.services
package:com.android.internal.display.cutout.emulation.double
package:com.android.providers.telephony
package:com.android.dynsystem
package:com.miui.powerkeeper
package:com.goodix.fingerprint
package:com.google.android.googlequicksearchbox
```

En esta lista tendremos que buscar el paquete de la aplicación que hemos instalado en este caso discord pero para no buscar entre todas las líneas que aparecen que son un montón podemos usar el comando **adb shell pm list packages | findstr discord** esto nos buscaría las líneas donde ponga discord. Nos muestra el paquete de discord.

```
C:\Users\rami>adb shell pm list packages | findstr discord
package:com.discord
```

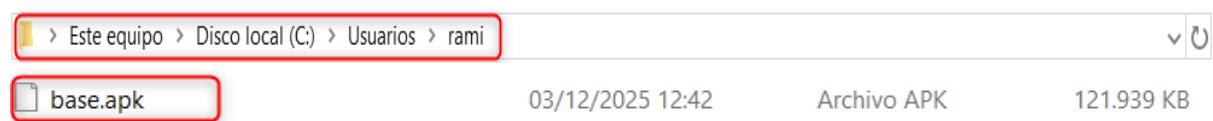
Una vez encontrado el paquete deberemos encontrar la ruta de instalación de este lo haremos con el comando **adb shell pm path (nombre del paquete)** en este caso el nombre del paquete es **com.discord**.

```
C:\Users\rami>adb shell pm path com.discord
package:/data/app/~~mj0zyncwxGiRb1QXU7_UeA==/com.discord-UDh6MG4dGDTwkGW4bHk7pw==/base.apk
package:/data/app/~~mj0zyncwxGiRb1QXU7_UeA==/com.discord-UDh6MG4dGDTwkGW4bHk7pw==/split_config.arm64_v8a.apk
package:/data/app/~~mj0zyncwxGiRb1QXU7_UeA==/com.discord-UDh6MG4dGDTwkGW4bHk7pw==/split_config.es.apk
package:/data/app/~~mj0zyncwxGiRb1QXU7_UeA==/com.discord-UDh6MG4dGDTwkGW4bHk7pw==/split_config.xxhdpi.apk
```

Ya tenemos la ruta de instalación, ahora tendríamos que extraer la apk base al ordenador para poder analizar el código con jadx esto lo hacemos con el comando adb **pull /data/app/~~mj0zyncwxGiRb1QXU7_UeA==/com.discord-UDh6MG4dGDTwkGW4bHk7pw==/base.apk**

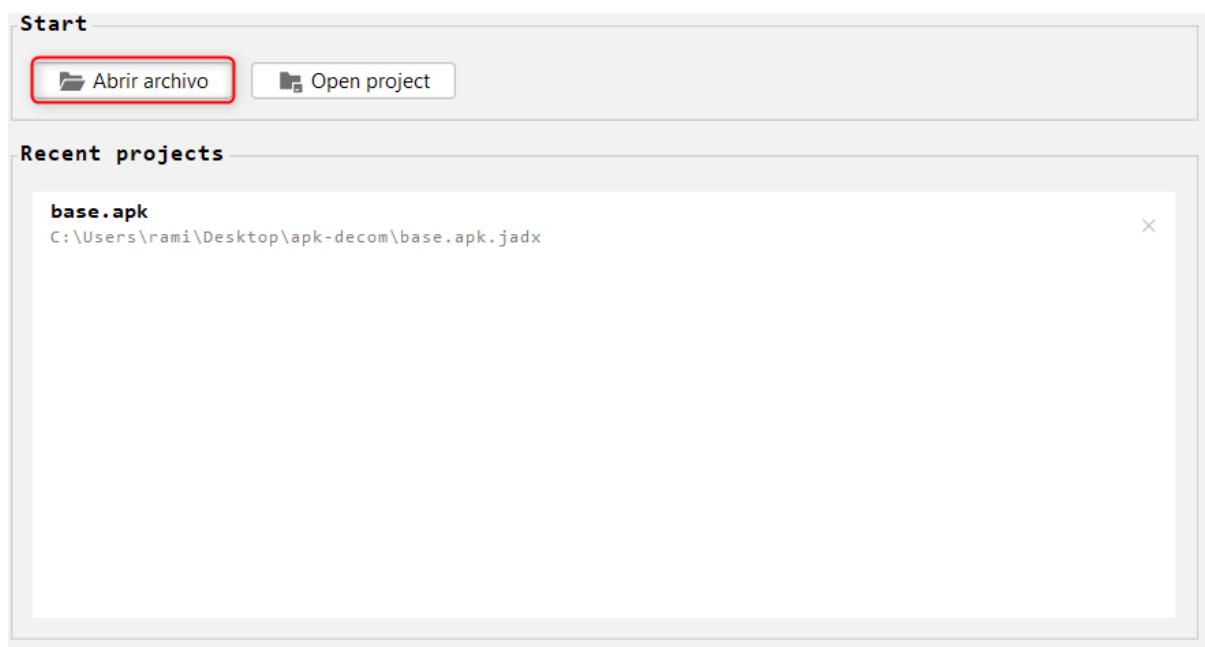
```
C:\Users\rami>adb pull /data/app/~~mj0zyncwxGiRb1QXU7_UeA==/com.discord-UDh6MG4dGDTwkGW4bHk7pw==/base.apk
/data/app/~~mj0zyncwxGiRb1QXU7_UeA==/com.discord-UDh6MG4dGDTwkGW4bHk7pw==/base.apk: 1 file pulled, 0 skipped. 35.8 MB/s (124864602 bytes in 3.328s)
```

Cuando ejecutemos el anterior comando se nos extraerá el archivo del apk en la carpeta donde nos encontramos en cmd en este caso C:\users\rami como podemos tenemos un archivo llamado base.apk sería ese el que vamos a analizar

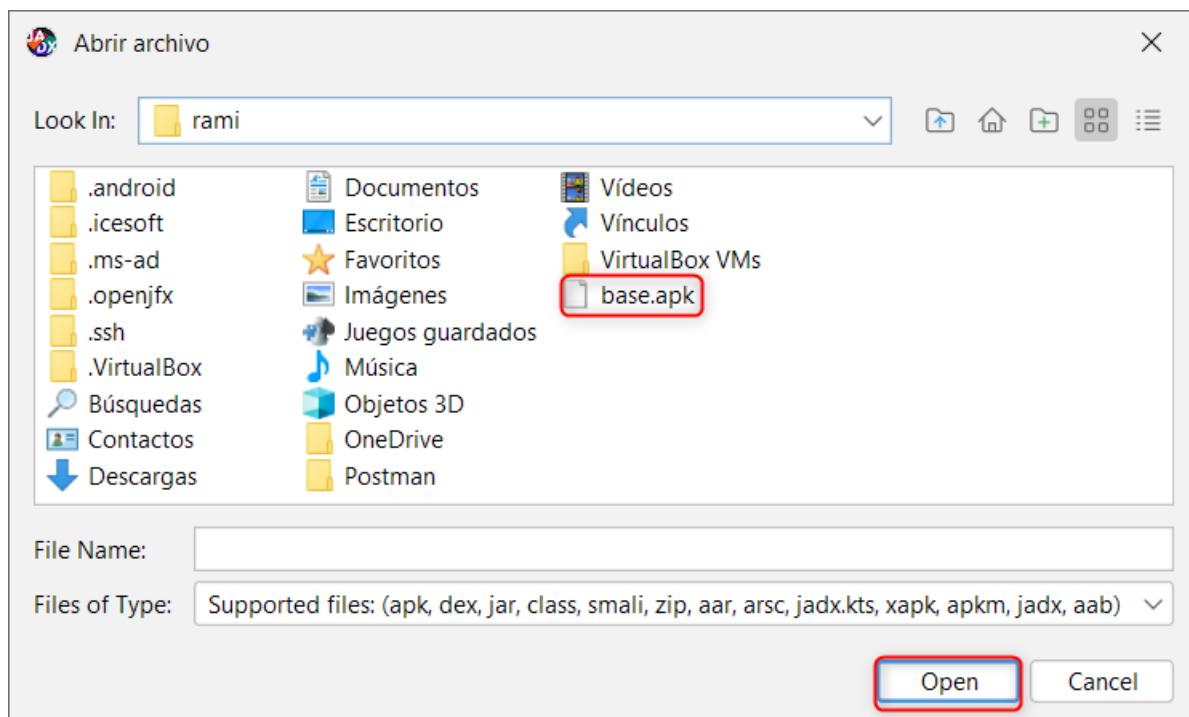


Una vez que ya se ha extraído y vemos el archivo base.apk procederemos a abrirlo en jadx para analizar el código en busca de hardcoding.

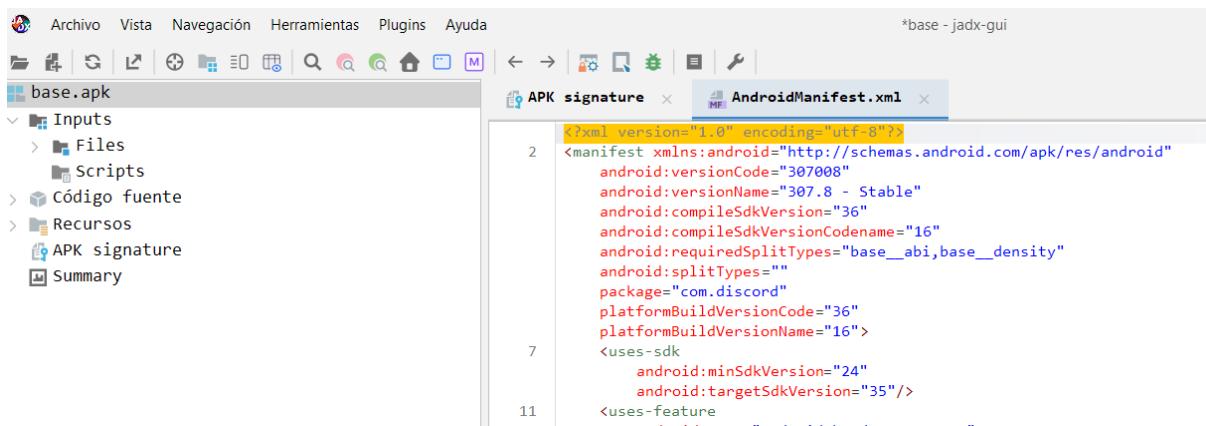
Abrimos jadx y nos iremos a la opción de abrir archivo



Seleccionaremos la ruta donde lo tenemos pinchamos en el archivo base.apk y le daremos a open.

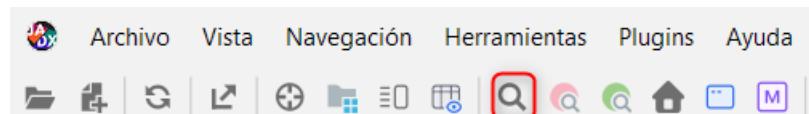


Una vez abierto el archivo e jadx se verá algo parecido a esto



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    android:versionCode="307008"
    android:versionName="307.8 - Stable"
    android:compileSdkVersion="36"
    android:compileSdkVersionCodename="16"
    android:requiredSplitTypes="base_abi,base_density"
    android:splitTypes=""
    package="com.discord"
    platformBuildVersionCode="36"
    platformBuildVersionName="16">
<uses-sdk
    android:minSdkVersion="24"
    android:targetSdkVersion="35"/>
<uses-feature
    ... >
```

Para empezar a buscar por palabras clave podemos pinchar en el icono de la lupa.



Por ejemplo, podemos buscar por palabras clave como api_key, acces_token, client_secret o passwd.

Aquí buscamos por api_key pro no encontramos nada importante

Buscar texto: api_key

Buscar definiciones de: Clase Método Campo Código Resource Comments Limit to package:

Nodo
com.discord.R.string
com.google.firebaseio.b.an(Context) b
io.fabric.sdk.android.a.b.g.au(Context) String
io.fabric.sdk.android.a.b.g.au(Context) String
com.crashlytics.android.core.k.AnonymousClass13.AnonymousClass1(AnonymousClass13) void put("api_key", k.this.pR.oC);

```
public static final int google_api_key = 0x7f120725;
public static final int google_crash_reporting_api_key = 0x7f120727;
return new b(string, rVar.getString("google_api_key"), rVar.getString("fi
throw new IllegalArgumentException("Fabric could not be initialized, API
io.fabric.sdk.android.c.wA().e("Fabric", "Fabric could not be initialized
put("api_key", k.this.pR.oC);
```

luego buscamos por acces_token y directamente no encuentra nada

Buscar texto: acces_token

Buscar definiciones de: Clase Método Campo Código Resource Comments Limit to package:

Nodo

Buscamos por client_secret y ya si encontramos algo más una credencial en el código esta credencial es la que autentica la aplicación de discord ante los servidores de spotify

Buscar texto: client_secret

Buscar definiciones de: Clase Método Campo Código Resource Comments Limit to package:

Nodo
com.discord.utilities.integrations.SpotifyHelper
com.discord.utilities.rest.RestAPI.Companion.init(Context) void

```
public static final String SPOTIFY_API_CLIENT_SECRET = "836a44ab33cb40f893cdd180f7f0ccc6";
companion.setApiSpotifyAuth(new RestAPI((RestAPIInterface) new RestAPIBuilder("https://accounts.spotify.
```

3.Resultados

Como he comentado he encontrado una credencial en el código que es la que autentica la aplicación de discord ante los servidores de spotify

```
public static final String SPOTIFY_API_CLIENT_SECRET = "836a44ab33cb40f893cdd180f7f0ccc6";
```

También se ha encontrado la credencial perteneciente a la API de Google reCAPTCHA.

```
private static final String CAPTCHA_SITE_KEY = "6Lff5jIUAAAAAImNxvYYPv2Vw2En3Dexy4oX2o4s";
```

Se han encontrado claves de API de Google (Google API Keys) expuestas en texto plano

```
<resources.arsc:/res/values/strings.xml> <string name="google_api_key">AIzaSyCY8pVLbc0IWz6NdlbaGckvh0mfIu02U</string>
<resources.arsc:/res/values/strings.xml> <string name="google_crash_reporting_api_key">AIzaSyCY8pVLbc0IWz6NdlbaGckvh0mfIu02U</string>
```

Esta cadena de texto es un Google OAuth 2.0 Client ID y sirve para identificar la aplicación ante los servidores de Google, generalmente para Iniciar sesión con Google

```
<string name="default_web_client_id">162066849712-o63evu2i36kqp7qa13r8np9c5m6ce3ff.apps.googleusercontent.com</string>
```

4. Recomendaciones

Durante el análisis de la aplicación, se han identificado credenciales hardcodeadas en el código fuente descompilado. El hallazgo más crítico es la exposición de un **Client Secret de Spotify**, lo cual compromete la integridad de la integración OAuth2. También se han encontrado claves de API de Google y URLs de infraestructura.

1: Exposición de Spotify Client Secret (CRÍTICO)

- **Descripción:** Se encontró la variable SPOTIFY_API_CLIENT_SECRET con la clave privada en texto plano dentro de la clase SpotifyHelper.
- **Impacto:** El *Client Secret* es la llave para autenticar a la aplicación ante Spotify. Su exposición permite a un atacante suplantar la identidad de la aplicación Discord, realizar ataques de *Man-in-the-Middle* durante la vinculación de cuentas o abusar de la cuota de la API, provocando una denegación de servicio.
- **Recomendación:**
 - **Eliminación:** Eliminar el *Client Secret* del código de la aplicación móvil.
 - **Patrón Backend-for-Frontend (BFF):** Implementar un servicio intermediario en el backend de Discord. La aplicación móvil debe solicitar la autenticación al servidor de Discord, y es este servidor el que se comunica con Spotify para intercambiar el código de autorización por el token de acceso.
 - **Rotación de Credenciales:** Revocar la clave actual en el panel de desarrolladores de Spotify y generar una nueva.

2: Google API Keys Hardcodeadas (MEDIO)

- **Descripción:** Se identificaron claves API de Google (probablemente para Firebase, Maps o GCM) en el archivo **strings.xml**.
- **Impacto:** Si estas claves no tienen restricciones, un atacante podría clonarlas y usarlas en otras aplicaciones, consumiendo la cuota financiera de Discord (robo de recursos) o accediendo a datos analíticos.
- **Recomendación:**
 - **Restricción de Aplicación (SHA-1):** Configurar en la consola de Google Cloud que estas claves solo puedan ser usadas por la aplicación con el nombre de paquete com.discord y la firma digital (SHA-1) del certificado de producción de Discord.
 - **Restricción de API:** Limitar el alcance de la clave para que solo sirva para los servicios estrictamente necesarios, impidiendo su uso para servicios más críticos.

3: Clave Pública de reCAPTCHA (Baja)

- **Descripción:** Se halló la CAPTCHA_SITE_KEY.
- **Impacto:** Es necesario que esta clave sea pública para que el cliente renderice el captcha. El riesgo es el "robo de cuota" si se usa en webs de terceros.
- **Recomendación:**
 - **Lista Blanca de Dominios/Paquetes:** Verificar en la consola de administración de reCAPTCHA que la clave esté restringida para aceptar peticiones únicamente provenientes del applicationId de Discord. Cualquier intento de uso desde otro origen debe ser bloqueado por Google.

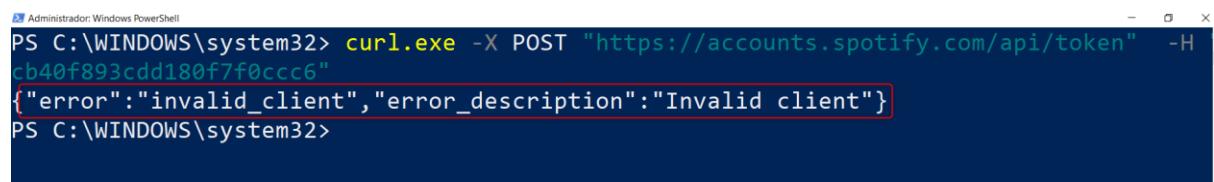
5. Conclusiones Generales (Mejores Prácticas)

Para evitar estos fallos en futuras versiones, se recomienda implementar las siguientes prácticas:

1. **Gestión de Secretos:** Utilizar herramientas como **Vault** para las claves públicas, asegurando que los secretos privados (como el de Spotify) nunca lleguen al repositorio de código fuente final.
2. **Escaneo de Seguridad Automatizado:** Integrar herramientas SAST (como SonarQube o TruffleHog) para detectar credenciales hardcodeadas antes de compilar la APK .
3. **Ofuscación Avanzada:** Se recomienda usar herramientas comerciales (como DexGuard) que cifran las cadenas de texto y las descifran en tiempo de ejecución, dificultando el análisis con herramientas como JADX.

6. Prueba de Secret spotify con (Curl)

Abrimos una powershell y ejecutamos el siguiente comando :
(curl.exe -X POST "https://accounts.spotify.com/api/token" -H "Content-Type: application/x-www-form-urlencoded" -d "grant_type=client_credentials&client_id=PON_AQUI_TU_CLIENT_ID&client_secret=PON_AQUI_EL_SECRETO")



```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> curl.exe -X POST "https://accounts.spotify.com/api/token" -H "Content-Type: application/x-www-form-urlencoded" -d "grant_type=client_credentials&client_id=PON_AQUI_TU_CLIENT_ID&client_secret=PON_AQUI_EL_SECRETO"
{"error": "invalid_client", "error_description": "Invalid client"}
```

Como podemos ver nos da el error invalid client esto significa que Discord ya se dio cuenta del fallo y cambió la contraseña.



Universidad
Francisco de
Vitoria
Centro de
Documentación
Europea
UFV Madrid