Challenge description:

*You need to declare and implement a service with an intent filter with action com.mobiotsec.intent.action.GIMMELOCATION. The system will find your service and it will start it with a startForegroundService() method (and an appropriate intent as argument). The system expects to get back the current location (as a Location object).  
  
During the test, the system will change the current location at run-time and it will query your service to get the updated location. If the expected location matches with what you reply back, the flag will be printed in the logs.  
  
Your service should "return" the reply to the system with a broadcast intent, with a specific action and bundle, as in the snippet below:  
  
Location currLoc = getCurrentLocation(); // put your magic here  
Intent i = new Intent();  
i.setAction("com.mobiotsec.intent.action.LOCATION\_ANNOUNCEMENT");  
i.putExtra("location", currLoc);  
sendBroadcast(i);*

Solution

First thing first, we need as said to declare an intent filter with the required intent to get the location, which will be called from inside the victim app via the *startForegroundService()* method

<service  
 android:name=".LocationService"  
 android:exported="true">  
 <intent-filter>  
 <action android:name="com.mobiotsec.intent.action.GIMMELOCATION" />  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
</service>

Then, we need to declare 2 location permissions, in order to retrieve correctly the location via Service call (also, others can be *FOREGROUND\_SERVICE* and *INTERNET* to further get access to location and sending of data via internet data).

In our location service, we actually need to start the Service call inside the *onCreate* method and then retrieve via a Broadcast Receiver the correct location. One approach, which I followed, is based on the newer API based on:

fusedLocationClient = LocationServices.*getFusedLocationProviderClient*(this);

We have also to define the *onStartCommand* call, which handles incoming commands or start requests from clients. This way, we get data from services and get the corresponding callback data from intents when they’re called. This makes us start the location retrieval logic, also spawning a corresponding notification if needed. In any case, the service will be recreated in case it’s killed via this method return variable.

As said, one can define a particular *getCurrentLocation()* method to properly get data and retrieve correctly latitude, longitude and altitude (if you open the Python checker script, this is what gets called; the teacher also told us the script for this challenge must be called in order to see the flag).

Here, we see if the correct permission was grated, then we create a callback to receive the broadcast location data via the correct intent, and then simply retrieve everything:

package com.example.whereareyou;  
  
import android.Manifest;  
import android.app.Notification;  
import android.app.NotificationChannel;  
import android.app.NotificationManager;  
import android.app.Service;  
import android.content.Context;  
import android.content.Intent;  
import android.content.pm.PackageManager;  
import android.location.Location;  
import android.os.IBinder;  
import android.os.Looper;  
import android.util.Log;  
  
import androidx.annotation.Nullable;  
import androidx.core.app.NotificationCompat;  
import androidx.core.content.ContextCompat;  
  
import com.google.android.gms.location.FusedLocationProviderClient;  
import com.google.android.gms.location.LocationCallback;  
import com.google.android.gms.location.LocationRequest;  
import com.google.android.gms.location.LocationResult;  
import com.google.android.gms.location.LocationServices;  
  
public class LocationService extends Service {  
 private static final String *TAG* = "MOBIOTSEC";  
 private FusedLocationProviderClient fusedLocationClient;  
  
 @Override  
 public void onCreate() {  
 super.onCreate();  
 fusedLocationClient = LocationServices.*getFusedLocationProviderClient*(this);  
 startForegroundServiceNotification(); // Start foreground service with a notification  
 }  
  
 @Nullable  
 @Override  
 public IBinder onBind(Intent intent) {  
 return null;  
 }  
  
 @Override  
 public int onStartCommand(Intent intent, int flags, int startId) {  
 if (intent != null && "com.mobiotsec.intent.action.GIMMELOCATION".equals(intent.getAction())) {  
 startLocationUpdates(); // Start location updates when the appropriate intent is received  
 }  
  
 return *START\_STICKY*;  
 }  
  
 private void startForegroundServiceNotification() {  
 String channelId = "location\_service\_notification";  
 NotificationCompat.Builder notificationBuilder = new NotificationCompat.Builder(this, channelId)  
 .setSmallIcon(android.R.drawable.*ic\_dialog\_info*)  
 .setContentTitle("Location service")  
 .setPriority(NotificationCompat.*PRIORITY\_DEFAULT*);  
  
 if (android.os.Build.VERSION.*SDK\_INT* >= android.os.Build.VERSION\_CODES.*O*) {  
 NotificationChannel channel = new NotificationChannel(channelId, "Location Service Notification", NotificationManager.*IMPORTANCE\_DEFAULT*);  
 NotificationManager notificationManager = getSystemService(NotificationManager.class);  
 notificationManager.createNotificationChannel(channel);  
 }  
  
 Notification notification = notificationBuilder.build();  
  
 startForeground(1, notification);  
 }  
  
 private void startLocationUpdates() {  
 if (checkLocationPermission()) {  
 // Define the location request  
 LocationRequest locationRequest = new LocationRequest();  
 locationRequest.setPriority(LocationRequest.*PRIORITY\_HIGH\_ACCURACY*);  
 locationRequest.setInterval(1000);  
  
 // Create the location callback  
 LocationCallback locationCallback = new LocationCallback() {  
 @Override  
 public void onLocationResult(LocationResult locationResult) {  
 super.onLocationResult(locationResult);  
 for (Location location : locationResult.getLocations()) {  
 // Handle the received location  
 sendLocationBroadcast(location);  
 }  
 }  
 };  
  
 fusedLocationClient.requestLocationUpdates(locationRequest, locationCallback, Looper.*getMainLooper*());  
 } else {  
 Log.*e*(*TAG*, "Location permission not granted");  
 }  
 }  
  
 private boolean checkLocationPermission() {  
 return ContextCompat.*checkSelfPermission*(this, Manifest.permission.*ACCESS\_FINE\_LOCATION*) == PackageManager.*PERMISSION\_GRANTED*;  
 }  
  
 private void sendLocationBroadcast(Location location) {  
 Intent i = new Intent("com.mobiotsec.intent.action.LOCATION\_ANNOUNCEMENT");  
 i.putExtra("location", location);  
 sendBroadcast(i);  
 Log.*i*(*TAG*, "Sent location broadcast");  
 }  
}

The MainActivity in this case can be created with simply the *onCreate* method, but we can explicitly call the *LocationService* class just defined as follows:

package com.example.whereareyou;  
import android.content.Context;  
import android.content.Intent;  
import android.os.Bundle;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
public class MainActivity extends AppCompatActivity {  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
  
 Context context = getApplicationContext();  
 Intent explicitIntent = new Intent(context, LocationService.class);  
 context.startService(explicitIntent);  
 }  
  
  
}

If everything goes well, inside the prompt for *adb logcat* command, you can see:

*10-29 18:38:13.234 20208 20208 I MOBIOTSEC: Good job! The flag is FLAG{piger\_ipse\_sibi\_obstat}*

*10-29 18:38:14.232 620 636 W AlarmManager: Unrecognized alarm listener com.android.server.location.gnss.GnssLocationProvider$$ExternalSyntheticLambda10@90b1290*

*10-29 18:38:14.236 20208 20208 I MOBIOTSEC: Good job! The flag is FLAG{piger\_ipse\_sibi\_obstat}*