Android Security: Inter-App and Network Communication Security

SECURING INTERACTION WITH OTHER APPS

Nitin Singh
MOBILE DEVELOPER

Securing Interaction with Intents

Intent Types

Outgoing Intents

Originating from your app

Incoming Intents

Originating from other apps

Securing Outgoing Intents

Implicit vs explicit intents

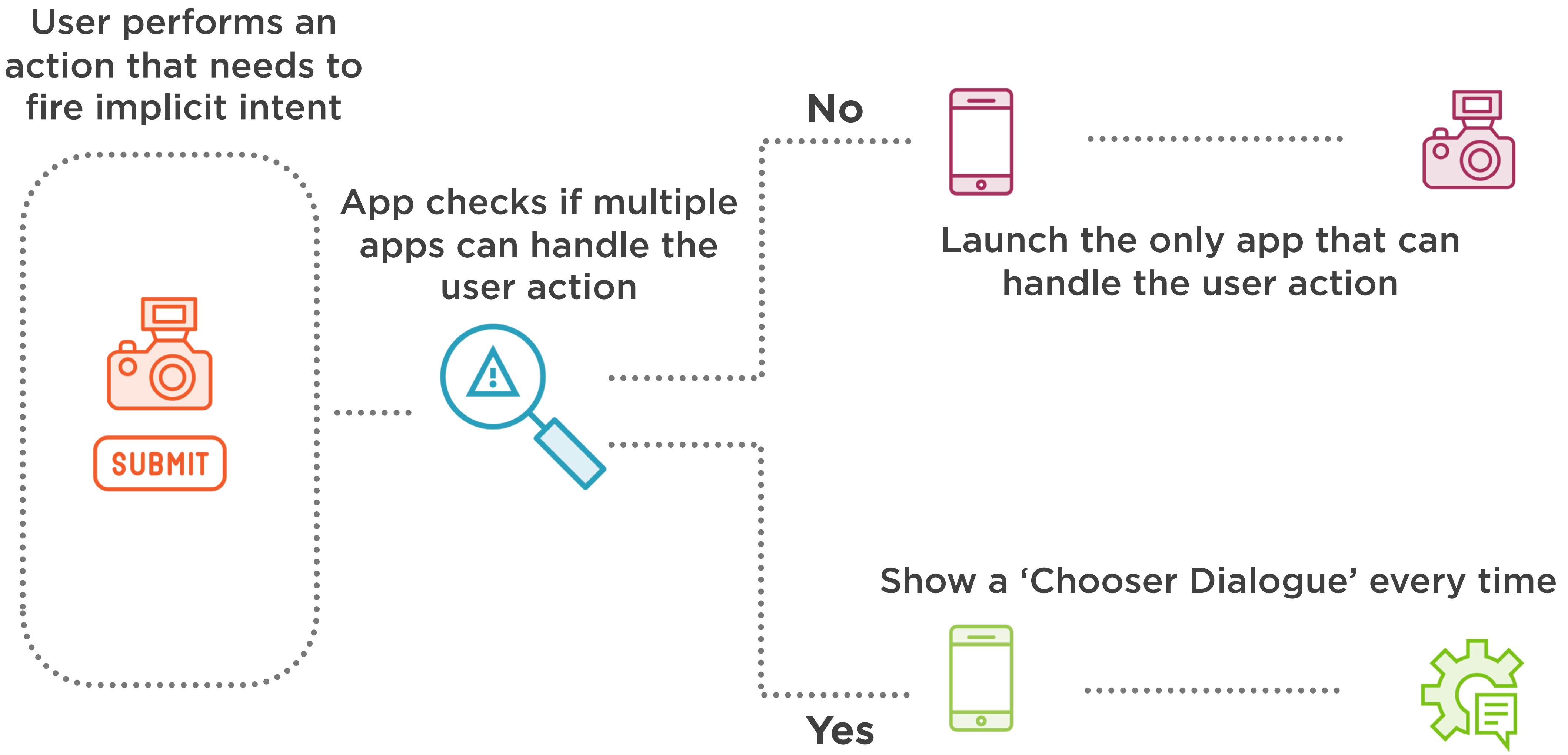
Explicit Intent

Don’t need securing as its your code

Implicit Intent

Can be secured using app chooser

Flow When Using App Chooser


User performs an action that needs to fire implicit intent

App checks if multiple apps can handle the user action

Launch the only app that can handle the user action

Show a ‘Chooser Dialogue’ every time

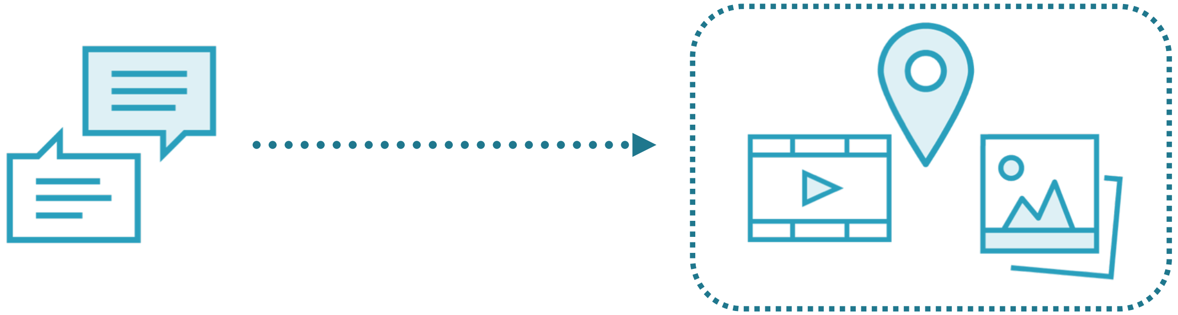
```
Intent intent = new Intent(Intent.ACTION_SEND);
List<ResolveInfo> possibleActivitiesList =
queryIntentActivities(intent, PackageManager.MATCH_ALL);

if (possibleActivitiesList.size() > 1) {
    String title = getResources().getString(R.string.chooser_title);
    Intent chooser = Intent.createChooser(intent, title);
    startActivity(chooser);
} else if (intent.resolveActivity(getPackageManager()) != null) {
    startActivity(intent);
}
```

Showing App Chooser

Demo

Contact Application



Securing Incoming Intents

Securing Incoming Intents

Intents fired by other apps that your app can handle

Secured by guarding components with android:permission attribute

Explicit vs implicit does not matter

Permission Protection levels

1 Normal

2 Dangerous

3 Signature

4 Special

Normal Permissions

1

For accessing device features that are low risk to user's privacy or other apps

2

Auto-grant at install time

3

No user interaction

Example: Internet permission

Dangerous Permissions

1

For accessing user’s data or sensitive device features

2

Need explicit consent from user

3

User consents via dialog shown by system on app’s request

Example: Camera permission

Signature Permissions

1

For custom-permission sharing between apps from the same developer

2

Auto-grant at install time

3

Used only for custom permissions specific to the developer's apps

Example: A second app from a developer accessing the first app's content provider

Special Permissions

1

For features which can affect user's entire experience on device

2

Accessed via system Intents

3

System shows detailed management screen to user

Example: WRITE_SETTINGS permission used to change system settings

Demo

Contact application

Create custom permission for content provider and enforce it

Preventing Other Apps from Accessing Your App’s Components

Controlling Access to App Components

Achieved using android:exported tag on the component

Setting it to “false” prevents other apps from accessing your app’s components

Default is ”true” for API level 16 or lower

```
<provider
    android:name="android.support.v4.content.FileProvider"
    ...
    android:exported="false">
</provider>
```

android:exported Attribute

Needs to be set on each component

Set to “false” for everything that doesn’t need to be exposed

Demo

Contact application

Safeguard an activity from other apps using android:exported attribute

Summary

Outgoing vs incoming Intents

Implicit vs explicit intents

Securing outgoing intents with app chooser

Securing incoming intents with permissions

Various permission protection levels and their behaviour

Protecting your app’s components using android:exported attribute

What’s next

Securing Network Communication Using Network Security Configs

Thank you