Mobile Security - Challenge 2

Justask

Marco Marchiante[1] Gabriel Rovesti[2]

Fri 20 Oct 2023

[1]marco.marchiante@studenti.unipd.it

[2]gabriel.rovesti@studenti.unipd.it

Summary

- Theoretical overview
- Assignment Rundown
- Solution Writeup
- Security Implications

Overview

Remembering some theory...

Implicit intents action invoked without exact knowledge about called component

Explicit intents launch a specific service or activity, passing data from one activity to another

The challenge

There is an app installed on the system.
The app has four activities.
Each of them has one part of the flag.

If you ask them nicely, they will all kindly reply with their part of the flag. They will reply with an Intent, the part of the flag is somehow contained there. Check the app's manifest for the specs. Good luck ;-)

The challenge

There is an app installed on the system.
The app has four activities.
Each of them has one part of the flag.

If you ask them nicely, they will all kindly reply with their part of the flag. They will reply with an Intent, the part of the flag is somehow contained there. Check the app's manifest for the specs. Good luck ;-)

The challenge

There is an app installed on the system.
The app has four activities.
Each of them has one part of the flag.

If you ask them nicely, they will all kindly reply with their part of the flag. They will reply with an Intent, the part of the flag is somehow contained there. Check the app's manifest for the specs. Good luck ;-)

The challenge

There is an app installed on the system.
The app has four activities.
Each of them has one part of the flag.

If you ask them nicely, they will all kindly reply with their part of the flag. They will reply with an Intent, the part of the flag is somehow contained there. Check the app's manifest for the specs. Good luck ;-)

The challenge

I Analyzing the Manifest file

II Launching the activities

III Getting activity result

IV Parsing the FLAG

```
There is an app installed on the
system.
The app has four activities.
Each of them has one part of the
flag.

If you ask them nicely, they will
all kindly reply with their part
of the flag. They will reply with
an Intent, the part of the flag
is somehow contained there. Check
the app's manifest for the specs.
Good luck ;-)
```

Let’s see how it’s done…

Analyzing the Manifest file

Launching the activities

Getting activity result

Obtaining the FLAG

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.victimapp">

  <application ... >

    <activity android:name=".MainActivity" ... />

    <activity android:name=".PartOne" android:exported="true"/>

    <activity android:name=".PartTwo" android:exported="true">
      <intent-filter>
        <action android:name="com.example.victimapp.intent.action.JUSTASK"/>
        <category android:name="android.intent.category.DEFAULT"/>
      </intent-filter>
    </activity>

    <activity android:name=".PartThree" android:exported="true"/>

    <activity android:name=".PartFour" android:exported="true">
      <intent-filter>
        <action android:name="com.example.victimapp.intent.action.JUSTASKBUTNOTSOSIMPLE"/>
        <category android:name="android.intent.category.DEFAULT"/>
      </intent-filter>
    </activity>

  </application>

</manifest>
```

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.victimapp">

  <application ... >

    <activity android:name=".MainActivity" ... />

    <activity android:name=".PartOne" android:exported="true"/>

    <activity android:name=".PartTwo" android:exported="true">
      <intent-filter>
        <action android:name="com.example.victimapp.intent.action.JUSTASK"/>
        <category android:name="android.intent.category.DEFAULT"/>
      </intent-filter>
    </activity>

    <activity android:name=".PartThree" android:exported="true"/>

    <activity android:name=".PartFour" android:exported="true">
      <intent-filter>
        <action android:name="com.example.victimapp.intent.action.JUSTASKBUTNOTSOSIMPLE"/>
        <category android:name="android.intent.category.DEFAULT"/>
      </intent-filter>
    </activity>

  </application>

</manifest>
```

IMPLICIT

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.victimapp">

  <application ... >

    <activity android:name=".MainActivity" ... />

    <activity android:name=".PartOne" android:exported="true"/>

    <activity android:name=".PartTwo" android:exported="true">
      <intent-filter>
        <action android:name="com.example.victimapp.intent.action.JUSTASK"/>
        <category android:name="android.intent.category.DEFAULT"/>
      </intent-filter>
    </activity>

    <activity android:name=".PartThree" android:exported="true"/>

    <activity android:name=".PartFour" android:exported="true">
      <intent-filter>
        <action android:name="com.example.victimapp.intent.action.JUSTASKBUTNOTSOSIMPLE"/>
        <category android:name="android.intent.category.DEFAULT"/>
      </intent-filter>
    </activity>

  </application>

</manifest>
```

IMPLICIT

EXPLICIT

Analyzing the Manifest file

Launching the activities

Getting activity result

Obtaining the FLAG

Intent declaration

Implicit intent:

```
val implicitIntent = Intent(
  "com.example.victimapp.intent.action.JUSTASK")
```

Explicit intent:

```
val explicitIntent = Intent()
intentPartOne.component = ComponentName(
  "com.example.victimapp",
  "com.example.victimapp.PartOne")
```

Launching the actvities

```
var explicitIntent = Intent() ...
var implicitIntent = Intent() ...

var activityLauncher = registerForActivityResult( ... )
activityLauncher.launch(explicitIntent)
activityLauncher.launch(implicitIntent)
```

!! Deprecated !!

```
const val PART_XYZ = 1 .. 4

startActivityForResult(explicitIntent, PART_XYZ)
startActivityForResult(implicitIntent, PART XYZ)
```

```
var activityLauncher = registerForActivityResult( ... )

val intent1 = Intent()
intentPartOne.component = ComponentName(
  "com.example.victimapp", "com.example.victimapp.PartOne")
val intent2 = Intent("com.example.victimapp.intent.action.JUSTASK")
val intent3 = Intent()
intentPartOne.component = ComponentName(
  "com.example.victimapp", "com.example.victimapp.PartThree")
val intent4 = Intent("com.example.victimapp.intent.action.JUSTASKBUTNOTSOSIMPLE")

activityLauncher.launch(intent1)
activityLauncher.launch(intent2)
activityLauncher.launch(intent3)
activityLauncher.launch(intent4)
```

Analyzing the Manifest file

Launching the activities

Getting activity results

Obtaining the FLAG

The result callback

```
val resultLauncher = registerForActivityResult(
    StartActivityForResult()) { result ->

  ...
}
```

!! Deprecated !!

```
protected void onActivityResult (
    int requestCode, int resultCode, Intent data) {
  ...
}
```

The result callback

```
val TAG = "MOBIOTSEC"

val resultLauncher = registerForActivityResult(
    StartActivityForResult()) { result ->

  val resultIntent: Intent? = result.data
  Log.i(TAG, resultIntent.extras)
}
```

Execution:

```
$ python 3 justask_checker.py victim.apk attacker.apk

> Starting: Intent { cmp=com.example.justask/.MainActivity }
> --------- beginning of main
> I MOBIOTSEC: Sent Intent 1
> I MOBIOTSEC: Bundle[mParcelledData.dataSize=64]
```

Bundle format

```
I MOBIOTSEC: Bundle[mParcelledData.dataSize=64]
```

- Special type of dictionary (collection of *key-value* pairs)
- Every *value* is *Parcelable*
- A Parcel is some data that can be sent through the IBinder (IPC)

Decoding the bundle payload

```
val bundle = resultIntent.extras

val strBundle = "Bundle{ "
for (key in bundle.keySet()) {
  val value = bundle.get(key)
  bundle += "$key => $value ; "
}
Log.i(TAG, bundle + "}")
```

Execution:

```
$ python 3 justask_checker.py victim.apk attacker.apk

> Starting: Intent { cmp=com.example.justask/.MainActivity }
> --------- beginning of main
> I MOBIOTSEC: Sent Intent 1
> I MOBIOTSEC: Bundle{ flag => FLAG{Gutta_cavat; }
```

Decoding the bundle payload

```
> I MOBIOTSEC: Got data from Part 1
> I MOBIOTSEC: Bundle{ flag => FLAG{Gutta_cavat_; }Bundle
> I MOBIOTSEC: Got data from Part 2
> I MOBIOTSEC: Bundle{ flag => lapidem_non_vi; }Bundle
> I MOBIOTSEC: Got data from Part 3
> I MOBIOTSEC: Bundle{ hiddenFlag => _sed_saepe; flag => let\'s spice this
> I MOBIOTSEC: Got data from Part 4
> I MOBIOTSEC: Bundle{ follow => Bundle[mParcelledData.dataSize=220]; }Bundle
```

Decoding the bundle payload

```
> I MOBIOTSEC: Got data from Part 1
> I MOBIOTSEC: Bundle{ flag => FLAG{Gutta_cavat_; }Bundle
> I MOBIOTSEC: Got data from Part 2
> I MOBIOTSEC: Bundle{ flag => lapidem_non_vi; }Bundle
> I MOBIOTSEC: Got data from Part 3
> I MOBIOTSEC: Bundle{ hiddenFlag => _sed_saepe; flag => let\'s spice this
> I MOBIOTSEC: Got data from Part 4
> I MOBIOTSEC: Bundle{ follow => Bundle[mParcelledData.dataSize=220]; }Bundle
> I MOBIOTSEC: Bundle{ the value => Bundle[mParcelledData.dataSize=180]; }Bundle
```

Decoding the bundle payload

```
fun unpackBundle(bundle: Bundle) {

  val strBundle = "Bundle{ "
  for (key in bundle.keySet()) {
    val value = bundle.get(key)
    bundle += "$key => $value ; "
  }
  Log.i(TAG, bundle + "}")

  for (key in bundle.keySet()) {
    val value = bundle.get(key)
    if (value is Bundle) {
        extractFlagFromBundle(value) // Recursive step
    }
  }
}

unpackBundle(resultIntent.extras)
```

Decoding the bundle payload

```
> I MOBIOTSEC: Got data from Part 1
> I MOBIOTSEC: Bundle{ flag => FLAG{Gutta_cavat_; }Bundle
> I MOBIOTSEC: Got data from Part 2
> I MOBIOTSEC: Bundle{ flag => lapidem_non_vi; }Bundle
> I MOBIOTSEC: Got data from Part 3
> I MOBIOTSEC: Bundle{ hiddenFlag => _sed_saepe; flag => let\'s spice this
> I MOBIOTSEC: Got data from Part 4
> I MOBIOTSEC: Bundle{ follow => Bundle[mParcelledData.dataSize=220]; }Bundle
> I MOBIOTSEC: Bundle{ the value => Bundle[mParcelledData.dataSize=180]; }Bundle
> I MOBIOTSEC: Bundle{ rabbit => Bundle[mParcelledData.dataSize=144]; }Bundle
> I MOBIOTSEC: Bundle{ hole => Bundle[mParcelledData.dataSize=112]; }Bundle
> I MOBIOTSEC: Bundle{ deeper => Bundle[mParcelledData.dataSize=76]; }Bundle
> I MOBIOTSEC: Bundle{ never ending story => _cadendo}; }Bundleup; }Bundle
```

```
val TAG = "MOBIOTSEC"

fun unpackBundle(bundle: Bundle) {

  val strBundle = "Bundle{ "
  for (key in bundle.keySet()) {
    val value = bundle.get(key)
    bundle += "$key => $value ; "
  }
  Log.i(TAG, bundle + "}")

  for (key in bundle.keySet()) {
    val value = bundle.get(key)
    if (value is Bundle) {
        extractFlagFromBundle(value) // Recursive step
    }
  }
}

val resultLauncher = registerForActivityResult(
    StartActivityForResult()) { result ->

  val resultIntent: Intent? = result.data
  unpackBundle(resultIntent.extras)
}
```

Execution:

```
$ python 3 justask_checker.py victim.apk attacker.apk

> Starting: Intent { cmp=com.example.justask/.MainActivity }
> --------- beginning of main
> I MOBIOTSEC: Sent Intent 1
> I MOBIOTSEC: Sent Intent 2
> I MOBIOTSEC: Sent Intent 3
> I MOBIOTSEC: Sent Intent 4
> I MOBIOTSEC: Got data from Part 4
> I MOBIOTSEC: Bundle{ follow => Bundle[mParcelledData.dataSize=220]; }Bundle
> I MOBIOTSEC: Bundle{ the value => Bundle[mParcelledData.dataSize=180]; }Bundle
> I MOBIOTSEC: Bundle{ rabbit => Bundle[mParcelledData.dataSize=144]; }Bundle
> I MOBIOTSEC: Bundle{ hole => Bundle[mParcelledData.dataSize=112]; }Bundle
> I MOBIOTSEC: Bundle{ deeper => Bundle[mParcelledData.dataSize=76]; }Bundle
> I MOBIOTSEC: Bundle{ never ending story => _cadendo}; }Bundle
> I MOBIOTSEC: Got data from Part 3
> I MOBIOTSEC: Bundle{ hiddenFlag => _sed_saepe; flag => let\'s spice this up; }Bundle
> I MOBIOTSEC: Got data from Part 2
> I MOBIOTSEC: Bundle{ flag => lapidem_non_vi; }Bundle
> I MOBIOTSEC: Got data from Part 1
> I MOBIOTSEC: Bundle{ flag => FLAG{Gutta_cavat_; }Bundle
```

Analyzing the Manifest file

Launching the activities

Getting activity results

Obtaining the FLAG

Obtain the flag

```
$ python 3 justask_checker.py victim.apk attacker.apkv

> Starting: Intent { cmp=com.example.justask/.MainActivity }
> --------- beginning of main
> I MOBIOTSEC: Sent Intent 1
> I MOBIOTSEC: Sent Intent 2
> I MOBIOTSEC: Sent Intent 3
> I MOBIOTSEC: Sent Intent 4
> I MOBIOTSEC: Got data from Part 4
> I MOBIOTSEC: Bundle{ follow => Bundle[mParcelledData.dataSize=220]; }Bundle
> I MOBIOTSEC: Bundle{ the value => Bundle[mParcelledData.dataSize=180]; }Bundle
> I MOBIOTSEC: Bundle{ rabbit => Bundle[mParcelledData.dataSize=144]; }Bundle
> I MOBIOTSEC: Bundle{ hole => Bundle[mParcelledData.dataSize=112]; }Bundle
> I MOBIOTSEC: Bundle{ deeper => Bundle[mParcelledData.dataSize=76]; }Bundle
> I MOBIOTSEC: Bundle{ never ending story => _cadendo}; }Bundle
> I MOBIOTSEC: Got data from Part 3
> I MOBIOTSEC: Bundle{ hiddenFlag => _sed_saepe; flag => let\'s spice this up; }Bundle
> I MOBIOTSEC: Got data from Part 2
> I MOBIOTSEC: Bundle{ flag => lapidem_non_vi; }Bundle
> I MOBIOTSEC: Got data from Part 1
> I MOBIOTSEC: Bundle{ flag => FLAG{Gutta_cavat_; }Bundle
```

Obtain the flag

```
...
> I MOBIOTSEC: Bundle{ never ending story => _cadendo}; }Bundle
...
> I MOBIOTSEC: Bundle{ hiddenFlag => _sed_saepe; flag => let\'s spice this up; }Bundle
...
> I MOBIOTSEC: Bundle{ flag => lapidem_non_vi; }Bundle
...
> I MOBIOTSEC: Bundle{ flag => FLAG{Gutta_cavat_; }Bundle
```

FLAG:

```
> FLAG{Gutta_cavat_lapidem_non_vi_sed_saepe_cadendo}
```

Security implications

Intent and Manifest analysis

A few points to consider:

- Understanding the clear goal of intents and how to use them
- Retrieving intent data correctly
- Proper handling of intent resolution

Intent and Manifest analysis

Possible problems:

- Intent redirection
- Intent injection
- Vulnerabilities from implicit intents
- Leveraging serialized objects maliciosuly