Lesson 2 – Improper Platform Usage – Intents

Introduction

This lesson focuses on finding and exploiting security issues in Android applications that are caused by improper usage of Intents and QR codes.

Intents are messaging objects provided by Android API to request an action from a different components. There are two types of Intents – explicit that are targeted towards a specific component, and implicit that only specify the action, they allow the component to be chosen by operating system or a user.

Getting started

You will need:

- an Android device or emulator with working camera and Barcode Scanner by Zxing Team installed,
- computer (Windows or Linux) with Android SDK.

Finding vulnerabilities

Firstly, start FourGoats application, open the menu and choose My QR Code. Scan the code, with a QR codes scanner to see what is inside. You will find out that the barcode contains an URI:

```
fourgoats://viewprofile?username=joegoat
```

Use one of the online QR codes generation tools to generate different barcodes and scan them with the app to see what happens. Try to check for different kinds of injections including SQL Injection.

If you try the WWW URL, you will find out that the web browser was opened with the given URL. You can also try URL for Zxing app at Google Play Store:

```
market://details?id=com.google.zxing.client.android
```

It worked. This is because the application passes the URI from the QR code to an Implicit Intent without any validation. Find the code of the QR Code scanner Activity in the mobile application:

```
public void onActivityResult(int requestCode, int resultCode, Intent intent) {
      IntentResult scanResult = IntentIntegrator.parseActivityResult(requestCode,
            resultCode, intent);
      if (scanResult != null) {
            try {
                  Intent profileIntent = new Intent(Intent.ACTION_VIEW,
                  Uri.parse(scanResult.getContents()));
                  startActivity(profileIntent);
            } catch (ActivityNotFoundException ex) {
                  Toast.makeText(this, "Incorrect QR Code", Toast.LENGTH_LONG).show();
            }
      }
}
```

Now look at the Intent filter in the manifest that handles these Intents:

```
<activity
      android:name=".activities.ViewProfile"
      android:exported="true"
      android:label="@string/profile">
      <intent-filter>
      <action android:name="android.intent.action.VIEW" />
      <category android:name="android.intent.category.DEFAULT" />
      <category android:name="android.intent.category.BROWSABLE" />
      <data android:host="viewprofile" android:scheme="fourgoats" />
      </intent-filter>
</activity>
```

There are two security issues with the QR code scanning Activity. Firstly, it can be used with a malicious WWW URL that would automatically get opened in the browser. Secondly, implicit Intent could be handled by a different, malicious application with the same Intent filter. In this case the risk is not so high, because the only data that the malicious application could get is the username from the URI, but some other sensitive data could be handled through the URI. There were some vulnerabilities published that resulted from session tokens being passed through URLs in Implicit Intens.

Exploiting the vulnerability

Make a new Android application using the creator from Android Studio. Add the Intent filter from ViewProfile Activity of FourGoats to MainActivity of your exploit application. Put the code for displaying the Intent content in the onCreate() method of MainActivity:

```
if (getIntent() != null && getIntent().getData() != null)
Toast.makeText(this, getIntent().getDataString(), Toast.LENGTH_LONG).show();
```

Now if you try to scan a barcode with user profile QR code a menu will be displayed, because two applications can handle the same Implicit Intent. The malicious application could use the same logo and a name that would persuade the victim to use it.

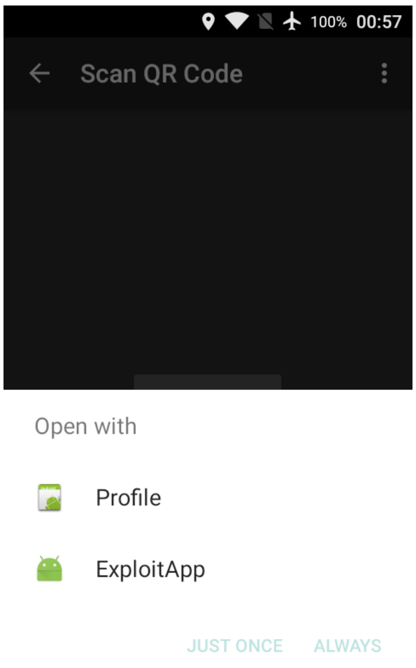
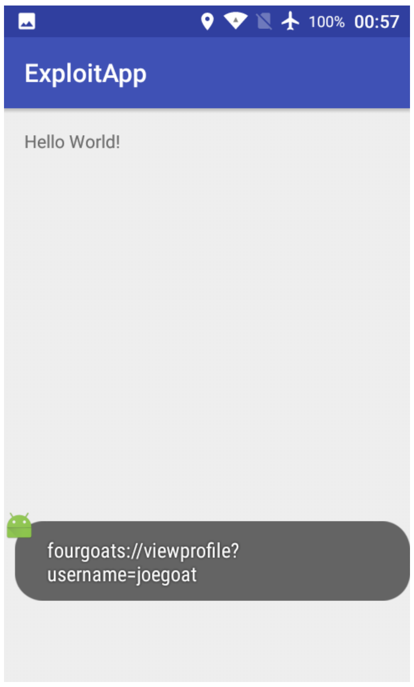


Illustration 1: A menu with applications that handle this Intent



fourgoats://viewprofile?username=joegoat

Illustration 2: Exploit Application displaying the captured URI

Fixing the vulnerability

This vulnerability can be fixed by using and Explicit Intent – there is no need for an implicit one.

```
Intent intent = new Intent(this, ViewProfile.class)
intent.setData(Uri.parse(scanResult.getContents()));
startActivity(intent);
```

Additionally, the application should be checked if the ViewProfile Activity really needs to be exported, and if so, the validation of data received with an Intent should be performed cautiously, because it provides an additional potential attack surface.