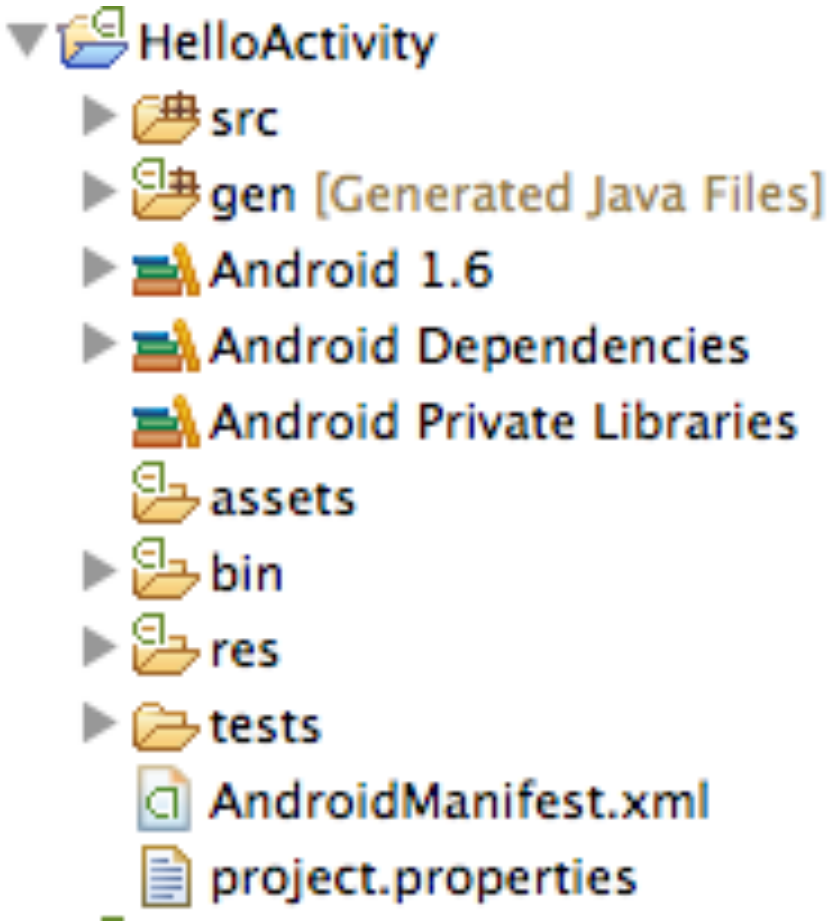
Android Basics and Reverse Engineering

GTFO Security - Joseph Paul Cohen

APK Contents



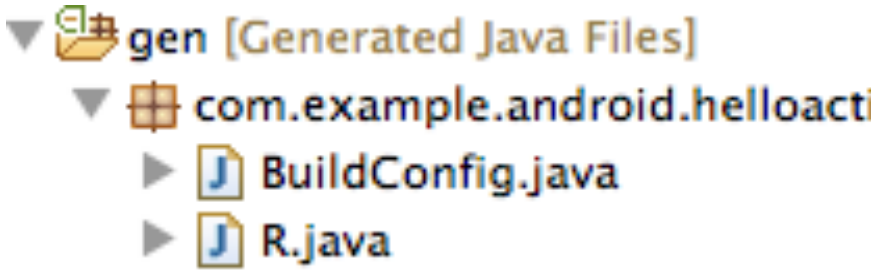
Java in src and gen

Compiled files in bin

Images in res

Manifest contains the integration with Android (Activities, Listeners)

Generated Files



R.java

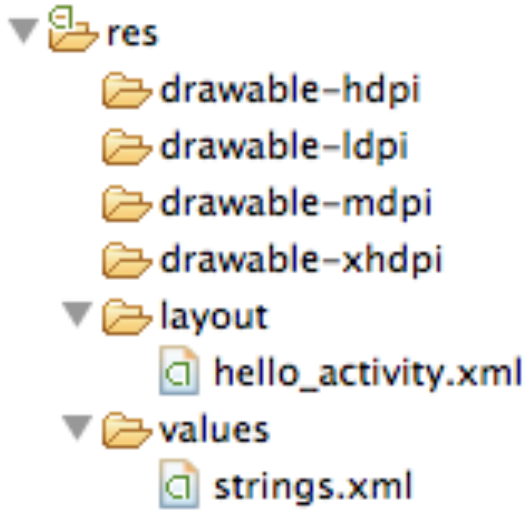
```
/* AUTO-GENERATED FILE.  DO NOT MODIFY.

package com.example.android.helloactivity;

public final class R {
    public static final class attr {
    }
    public static final class id {
        public static final int text=0x7f040000;
    }
    public static final class layout {
        public static final int hello_activity=0x7f
    }
    public static final class string {
        public static final int hello_activity_text
    }
}
```

Specifying values in res generates the R class which is used to deal with visual objects and strings. This is to allow the APK to adapt to various devices and langauges

Resource Files



strings.xml

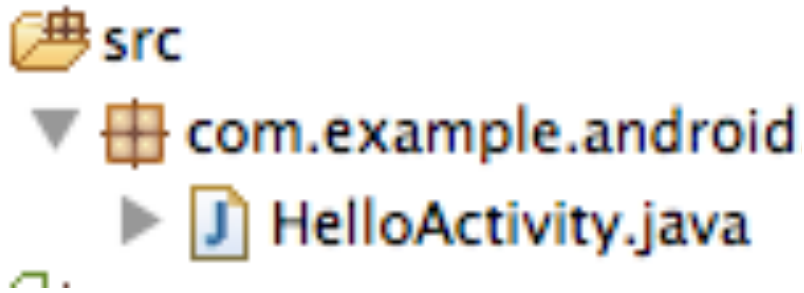
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="hello_activity_text_text">Hello,
</resources>
```

hello_activity.xml

```
<?xml version="1.0" encoding="utf-8"?>
<EditText xmlns:android="http://schemas.and
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:textSize="18sp"
    android:autoText="true"
    android:capitalize="sentences"
    android:text="@string/hello_activity_t
```

layouts and strings are stored as xml. Drawable resources are stored as images. Everything is identified by name and is referenceable via the R object.

Source Files



HelloActivity.java

```
package com.example.android.helloactivity;
import android.app.Activity;

public class HelloActivity extends Activity {
    public HelloActivity() {
    }
    /** Called with the activity is first created
    @Override
    public void onCreate(Bundle savedInstanceStat
        super.onCreate(savedInstanceState);

        // Set the layout for this activity.  You
        // in res/layout/hello_activity.xml
        setContentView(R.layout.hello_activity);
    }
}
```

An Activity is defined in the source and implements many onSomething functions. A layout is referenced using the R object.

Manifest

HelloActivity Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.android.helloactivity">
    <application android:label="Hello, Activity!">
        <activity android:name="HelloActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Specifies the activity to associate with an application. The package is the name that identifies the entire program in the Android OS

APK location on device

```
root@android:/ # ls /data/app | grep hello
com.example.android.helloactivity-1.apk
```

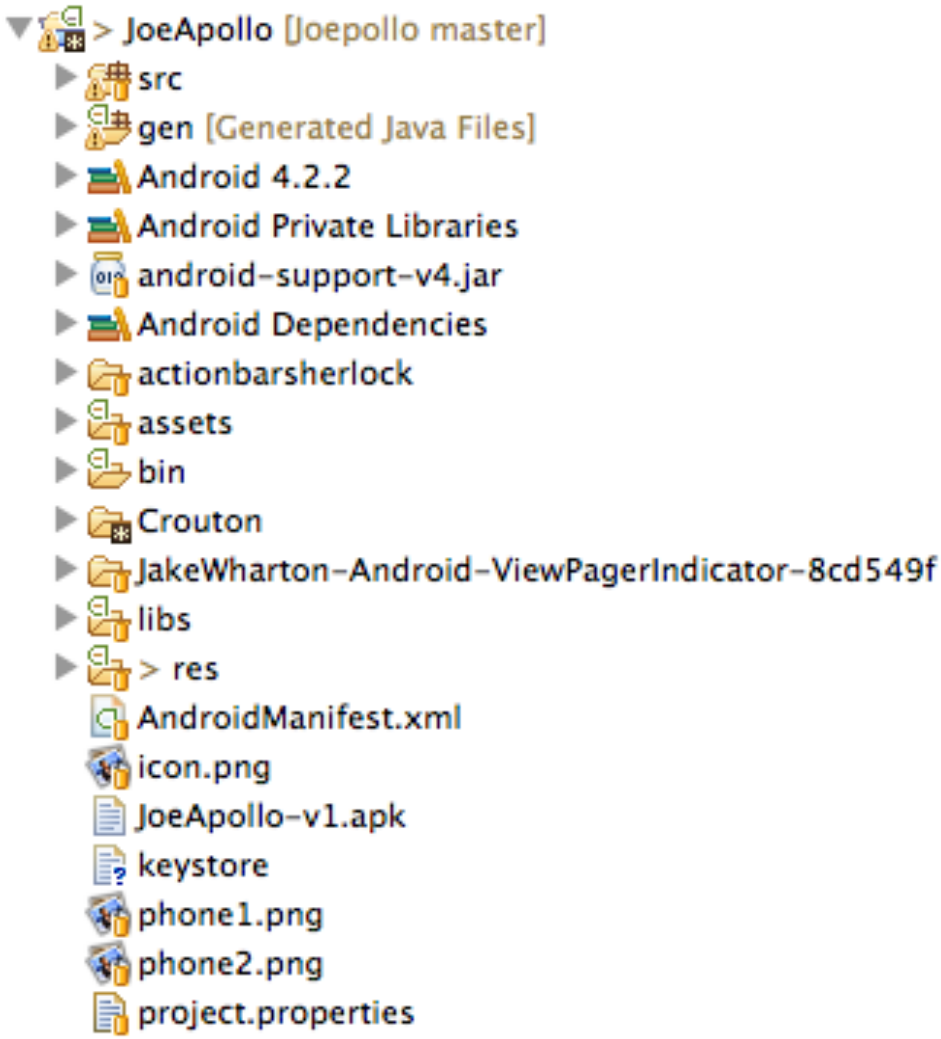
```
root@android:/ # ls -la /data/data/com.example.android.
helloactivity/
drwxrwx--x u0_a123  u0_a123      cache
lrwxrwxrwx install  install      lib -> /data/app-lib/com.
example.android.helloactivity-1
```

Program private storage

```
root@android:/ # ls /data/data/com.google.android.gm
app_sslcache
cache
databases
files
lib
shared_prefs

oot@android:/ # ls /data/data/com.google.android.
gm/databases
google_analytics_v2.db
google_analytics_v2.db-journal
internal.joseph@josephpcohen.com.db
internal.joseph@josephpcohen.com.db-journal
```

APK Contents (In the wild)



Support libraries

Many dependencies

Native code

Signing keys

Proguard

Reverse Engineering

Reconstruction of source and understanding of application logic.

dex2jar - Convert Android .dex code to Java .class files. 1-1 translation.

https://code.google.com/p/dex2jar/

JD-Core - Reconstruction of source from .class files. Good results most of time.

Obtaining APK files from Android

User install apps are installed to /data/app/

```
root@android:/ # ls /data/app/
com.andrewshu.android.reddit-1.apk
com.android.chrome-2.apk
com.android.vending-1.apk
com.bittorrent.client-2.apk
com.estrongs.android.pop-1.apk
com.estrongs.android.taskmanager-2.apk
com.example.android.helloactivity-1.apk
com.google.android.apps.currents-1.apk
com.google.android.apps.docs-1.apk
```

Obtaining APK files from Android

Using ADB pull we can copy the apk to our local machine

```
desktop$ adb pull /data/app/com.example.android.
helloactivity-1.apk .
681 KB/s (5096 bytes in 0.007s)

desktop$ file com.example.android.helloactivity-1.apk
com.example.android.helloactivity-1.apk: Zip archive data,
at least v2.0 to extract
```

dex2jar https://code.google.com/p/dex2jar/

```
$ ./d2j-dex2jar.sh com.example.android.helloactivity-1.apk
dex2jar com.example.android.helloactivity-1.apk -> com.example.android.helloactivity-1-dex2jar.jar
Done.
```

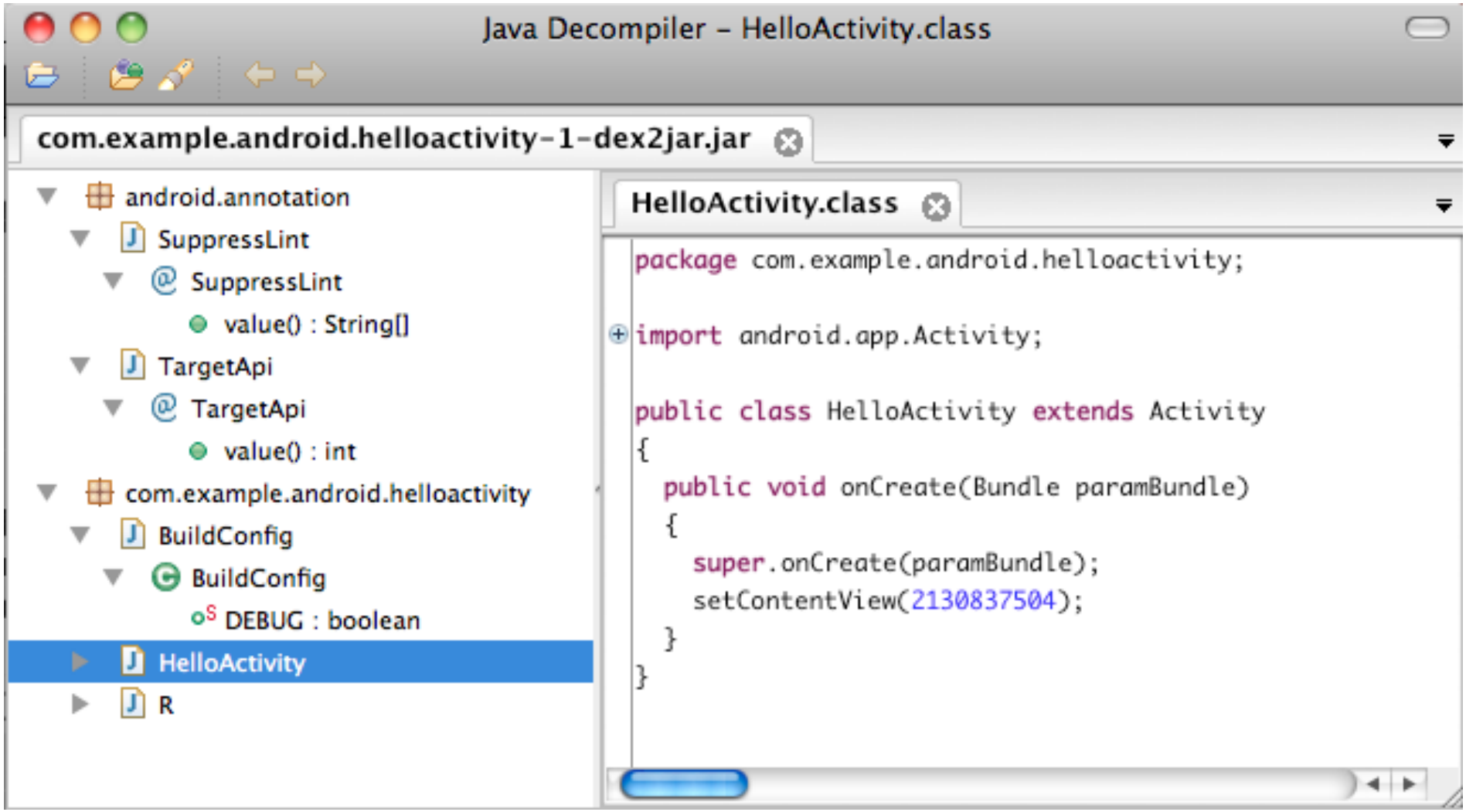
We can then extract the jar to inspect it

```
$ dtrx com.example.android.helloactivity-1-dex2jar.jar
```

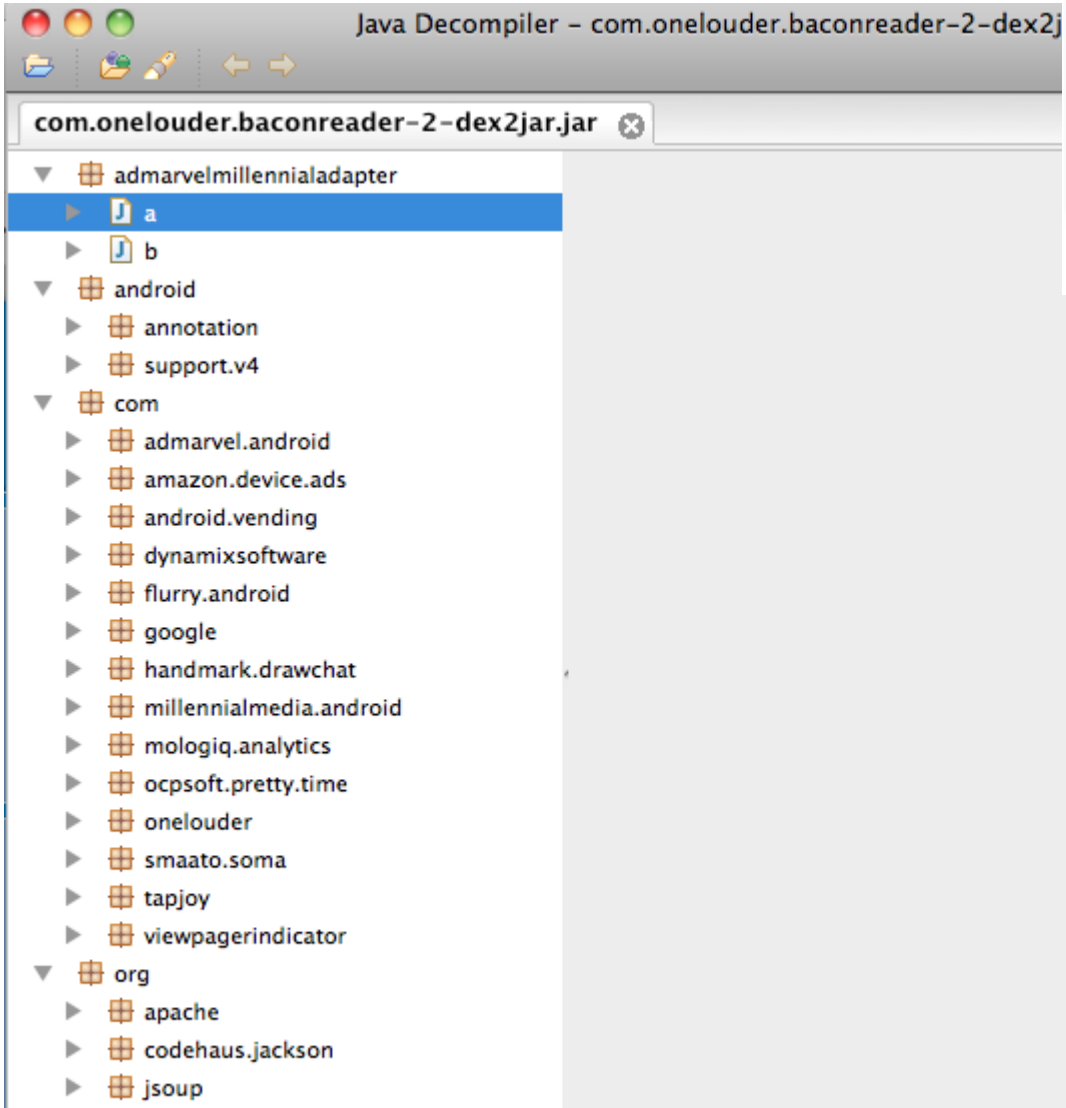
View the folder we just created

```
$ tree com.example.android.helloactivity-1-dex2jar
com.example.android.helloactivity-1-dex2jar
├── android
│   └── annotation
│       ├── SuppressLint.class
│       └── TargetApi.class
└── com
    └── example
        └── android
            └── helloactivity
                ├── BuildConfig.class
                ├── HelloActivity.class
                ├── R$attr.class
                ├── R$id.class
                ├── R$layout.class
                ├── R$string.class
                └── R.class
```

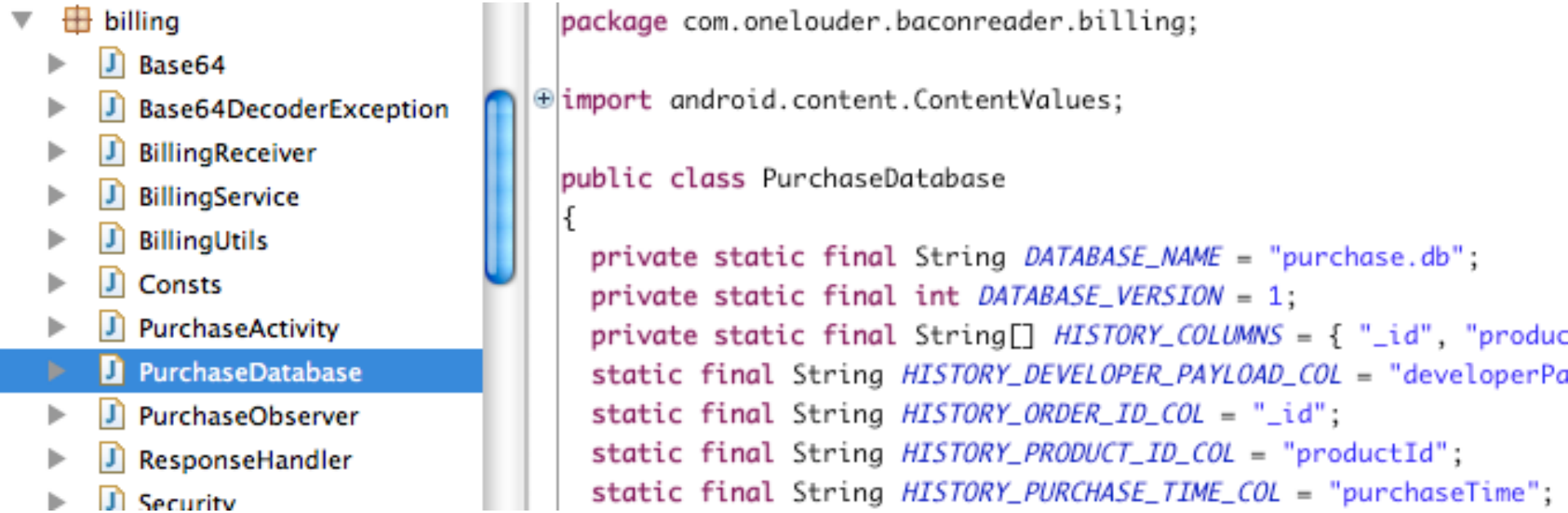
Decompile the class files from the jar



A look at BaconReader

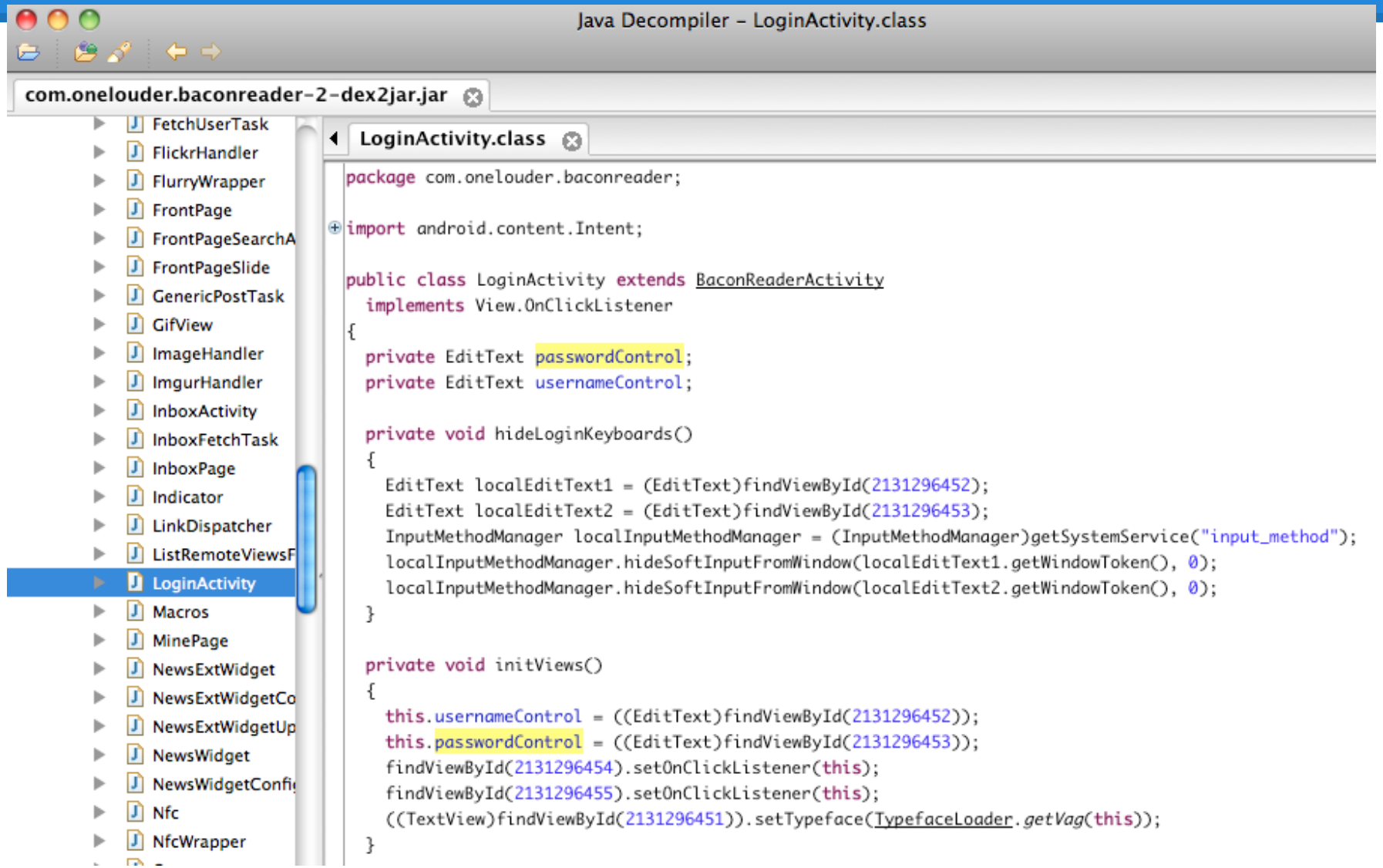


Look for PII



Found purchase history database “purchase.db”

LoginActivity



Follow login process

Login calls function execute

InitalLogin extends Authentication which extends AsyncTask

```
private void login(boolean paramBoolean)
{
  String str1 = this.usernameControl.getText().toString();
  String str2 = this.passwordControl.getText().toString();
  ((ViewFlipper)findViewById(2131296450)).showNext();
  new InitialLogin(str1, str2, paramBoolean).execute(new Void
}

private class InitialLogin extends Authentication
{
  public InitialLogin(String paramString1, String arg
  {
    super(paramString1, str);
  }

public class Authentication extends AsyncTask<Void, Void, Boolean>
{
```