

第 9 章 手机杀毒

第 9 章 手机杀毒模块.....	2
9.1 模块概述.....	2
9.1.1 功能介绍.....	2
9.1.2 手机病毒.....	3
9.2 数据库操作.....	3
9.2.1 数据库展示.....	4
9.2.2 数据库操作.....	4
9.2.3 获取 MD5 码.....	5
9.3 手机杀毒之应用信息展示.....	8
9.3.1 应用信息展示主界面.....	8
9.3.2 应用信息展示之 ListView 的 Item 布局.....	9
9.3.3 应用信息的实体类.....	11
9.3.4 应用信息展示之数据初始化.....	11
9.3.5 应用信息展示之数据适配器.....	13
9.4 手机杀毒之查杀进度展示.....	15
9.4.1 进度信息界面.....	15
9.4.2 进度信息初始化.....	17
9.4.3 查杀结束界面.....	20
9.4.4 查杀结束动画的布局.....	27
9.4.5 查杀结束动画的逻辑实现.....	30
9.5 手机杀毒之模拟查杀病毒.....	37
9.5.1 复制病毒库.....	37
9.5.2 模拟扫描查杀病毒.....	39
9.6 本章小结.....	41

第 9 章 手机杀毒模块

- ◆ 了解手机杀毒模块功能
- ◆ 掌握第三方数据库的使用
- ◆ 掌握病毒查杀的原理
- ◆ 了解应用程序的 MD5 码

大家在使用电脑时，最烦恼的事情就是被病毒感染，病毒会干扰电脑系统，并破坏系统中的程序，最主要的是有些病毒会窃取用户隐私和资料。同样，在使用手机时也会遇到这种情况，只不过手机中的病毒都存在于 APK 文件中，只要将病毒所在的 APK 文件删除即可将其清理掉，本章将针对手机杀毒模块进行详细讲解。

9.1 模块概述

9.1.1 功能介绍

手机杀毒模块主要用于全盘扫描手机程序，当点击全盘扫描条目时，会显示当前的查杀进度以及正在扫描的程序，如果某个程序是病毒则应用名称会显示成红色，手动卸载程序即可，如果没有病毒则显示扫描完成。当下次再进入手机杀毒模块时，会显示上次病毒查杀的时间，该功能界面效果如图 9-1 所示。

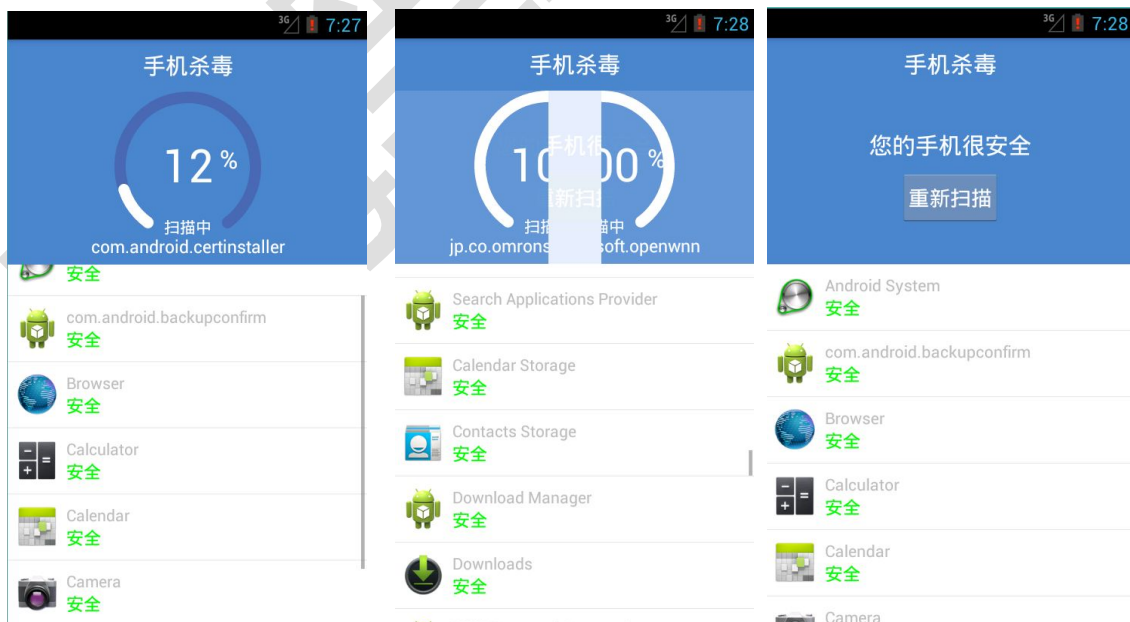


图 9-1 手机杀毒界面

9.1.2 手机病毒

手机病毒是一种具有传染性、破坏性的手机程序，可用杀毒软件进行清除与查杀，也可以手动卸载。其可利用短信、彩信、电子邮件、蓝牙等方式进行传播，会导致用户手机死机、关机、个人资料被删、泄露个人信息、自动拨打电话、发短（彩）信等进行恶意扣费，甚至会损毁 SIM 卡、芯片等硬件，导致手机无法正常使用。

历史上最早的手机病毒出现在 2000 年，当时，手机公司 Movistar 收到大量由计算机发出的名为“Timofonica”的骚扰短信，该病毒通过西班牙电信公司“Telefonica”的移动系统向系统内的用户发送脏话等垃圾短信。事实上，该病毒最多只能被算作短信炸弹。真正意义上的手机病毒直到 2004 年 6 月才出现，那就是“Cabir”蠕虫病毒，这种病毒通过诺基亚 s60 系列手机复制，然后不断寻找安装了蓝牙的手机。从此以后，手机病毒开始泛滥。

手机中的软件都安装在系统中，这个系统是一个嵌入式操作系统（固化在芯片中的操作系统，一般由 JAVA、C++ 等语言编写），相当于一个小型的智能处理器，手机病毒就是靠系统的漏洞来入侵手机的。手机病毒要传播和运行，必须要有移动服务商提供数据传输功能，而且手机需要支持 Java 等高级程序的写入功能。许多具备上网及下载等功能的手机都可能会被手机病毒入侵。手机病毒一般具有以下几个特点：

- 传染性：病毒通过自身复制来感染正常文件，达到破坏目标程序的目的，但是它的感染是有条件的，也就是病毒程序必须被执行之后它才具有传染性，才能感染其他文件。
- 破坏性：任何病毒侵入目标后，都会或大或小的对系统正常使用造成一定的影响，轻者降低系统的性能，占用系统资源，重者破坏数据导致系统崩溃，甚至损坏硬件。
- 潜伏性：一般病毒在感染文件后不是立即发作，而是隐藏在系统中，在满足条件时才能激活。病毒如果没有被激活，它就像其他没执行的程序一样，安静的呆在系统中，没有传染性也不具有杀伤力。
- 寄生性：病毒嵌入到载体中，依靠载体而生存，当载体被执行时，病毒程序也就被激活，然后进行复制和传播。

手机病毒的感染主要来源于应用程序的安装，要想杀死某个病毒只需要卸载病毒程序即可。每一个应用程序都有对应的特征码，这个特征码称之为 MD5 码，MD5 码是程序的唯一标识，手机杀毒厂商搜集了大量手机病毒应用的 MD5 码存入数据库中，当杀毒软件对手机应用进行扫描时，会先得到被扫描应用的 MD5 码，并将该 MD5 码与病毒数据库中存储的 MD5 码进行比对，如果发现应用程序的 MD5 码存在于数据库中，说明这个应用是病毒，可以直接清除或者手动卸载该应用。

要防范手机感染病毒，最好在安装程序时就要做好防范，例如不要点击手机短信中的 APK 链接下载应用；不要扫描不安全的二维码下载应用。由于有些病毒会伪装成市面上较流行的应用，通过源码解析将恶意代码植入到程序中再进行发布，这样的病毒普通用户辨别不了，因此下载应用最好在知名度较高的应用市场中下载。

9.2 数据库操作

手机杀毒模块的核心在于病毒数据库的操作，根据病毒数据库中存储的 MD5 码与应用的 MD5 码进行对比，如果匹配则该应用是病毒，本节将针对病毒数据库的操作进行详细讲解。

9.2.1 数据库展示

随着智能手机的发展，手机病毒种类越来越多，病毒数据库中的数据也越来越庞大，为了方便起见，这里直接使用较完整的第三方病毒数据库 `anvirus.db`。使用 SQLite Expert 打开病毒数据库的 `databale` 表，该表就用于存放病毒的相关信息，具体如图 9-2 所示。

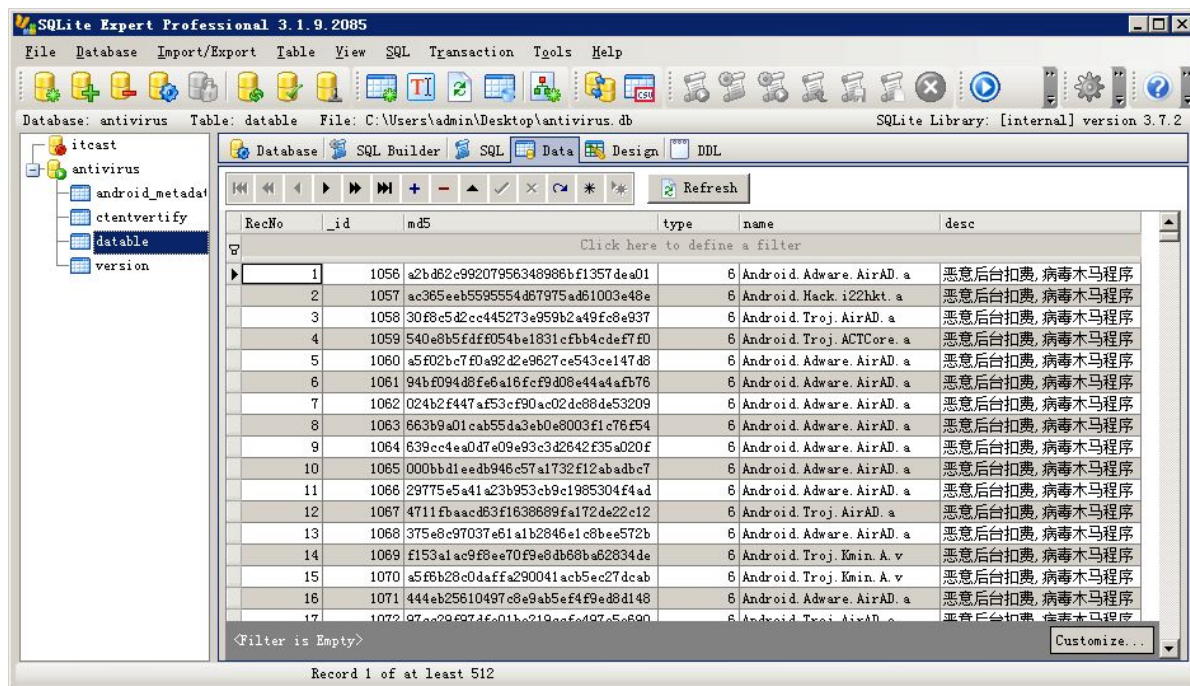


图 9-2 病毒数据库

如图 10-2 所示，`databale` 表中主要包含 `_id`、`md5`、`type`、`name`、`desc` 等字段，这些字段分别用于表示自增主键、病毒 MD5 特征码、病毒类型、病毒名称、病毒描述信息。

9.2.2 数据库操作

当数据库拷贝成功后，通过获取手机应用的 MD5 码与病毒数据库中的 MD5 码进行比对，如果当前应用的 MD5 码在病毒数据库中，则说明该应用就是病毒，这个过程实质上就是病毒查杀的过程，具体代码如【文件 9-1】所示。

【文件 9-1】AntivirusDao.java

```
1. public class AntivirusDao {
2.     public static boolean isVirus(Context context, String md5) {
3.         boolean result = false;
4.         String path = new File(context.getFilesDir(), "antivirus.db")
5.             .getAbsolutePath();
6.         // 打开数据库
7.         // 第一个参数：数据库的路径
8.         // 第二个参数：游标工厂
9.         // 第三个参数：标记(只读)
10.        SQLiteDatabase db = SQLiteDatabase.openDatabase(path, null,
11.            SQLiteDatabase.OPEN_READONLY);
```

```
12.         // select desc from databale where md5 =
13.         // 'a2bd62c99207956348986bf1357dea01'
14.         String sql = "select desc from databale where md5 = ?";
15.         Cursor cursor = db.rawQuery(sql, new String[] { md5 });
16.         if (cursor.moveToNext()) {
17.             result = true;
18.         }
19.         return result;
20.     }
21. /**
22.  * 添加病毒到数据库里面
23.  * @param context
24.  * @param md5
25.  * @param desc
26.  */
27.     public static void addVirus(Context context, String md5, String desc) {
28.         boolean result = false;
29.         String path = new File(context.getFilesDir(), "antivirus.db")
30.             .getAbsolutePath();
31.         // 打开数据库
32.         // 第一个参数：数据库的路径
33.         // 第二个参数：游标工厂
34.         // 第三个参数：标记(只读)
35.         SQLiteDatabase db = SQLiteDatabase.openDatabase(path, null,
36.             SQLiteDatabase.OPEN_READWRITE);
37.         // select desc from databale where md5 =
38.         // 'a2bd62c99207956348986bf1357dea01'
39.         ContentValues values = new ContentValues();
40.         values.put("md5", md5);
41.         values.put("desc", desc);
42.         values.put("type", "6");
43.         values.put("name", "Troj.Feiliu.a");
44.         db.insert("databale", null, values);
45.     }
46. }
```

在上述代码中，isVirus()方法接收的参数为应用的 MD5 码，打开病毒数据库，得到 MD5 列表，将参数中的 MD5 码与列表中的进行比对，如果存在则返回该应用为病毒的提示。addVirus()方法用于向病毒库中添加病毒信息。

9.2.3 获取 MD5 码

通过前面讲解可知，病毒数据库中保存的是手机病毒的 MD5 码，如果需要判断某个应用是否为病毒，

则必须要知道该应用的 MD5 码，接下来创建一个获取应用 MD5 码的工具类，具体代码如【文件 9-2】所示。

【文件 9-2】 MD5Utils.java

```
1. public class MD5Utils{
2.     /**
3.      * 加密
4.      * @param pwd
5.      */
6.     public static String encode(String pwd) {
7.         // MessageDigest 消息摘要
8.         try {
9.             MessageDigest digester = MessageDigest.getInstance("MD5");
10.            // 加密
11.            byte[] digest = digester.digest(pwd.getBytes());
12.            StringBuffer buffer = new StringBuffer();
13.            for (byte b : digest) {
14.                // 0xff 表示低 8 位
15.                int number = b & 0xff;
16.                // int 32 为 一个 int 是四个字节 一个字节 8 位
17.                // 任何一个值& 等于 0
18.                // & 1 = 1
19.                String hexString = Integer.toHexString(number);
20.                if (hexString.length() == 1) {
21.                    buffer.append("0" + hexString);
22.                } else {
23.                    buffer.append(hexString);
24.                }
25.            }
26.            return buffer.toString();
27.        } catch (Exception e) {
28.            // TODO Auto-generated catch block
29.            e.printStackTrace();
30.        }
31.        return "";
32.    }
33.    public static String encode(FileInputStream is) {
34.        try {
35.            MessageDigest digester = MessageDigest.getInstance("MD5");
36.            byte[] bytes = new byte[8192];
37.            int byteCount;
38.            while ((byteCount = is.read(bytes)) > 0) {
39.                digester.update(bytes, 0, byteCount);
40.            }
41.            byte[] digest = digester.digest();
```

```

42.         StringBuffer buffer = new StringBuffer();
43.         for (byte b : digest) {
44.             int number = b & 0xff; // 获取到低八位
45.             String hex = Integer.toHexString(number);
46.             if (hex.length() == 1) {
47.                 buffer.append("0" + hex);
48.             } else {
49.                 buffer.append(hex);
50.             }
51.         }
52.         return buffer.toString();
53.     } catch (Exception e) {
54.         e.printStackTrace();
55.     }
56.     return null;
57. }
58. }

```

在上述代码中，getFileMd5()方法接收一个应用的全路径名，最后返回该应用的 MD5 码。网络上许多获取文件 MD5 码的工具内部就是这样实现的。

至此，手机杀毒模块所用到的数据库和工具类已准备完成，包括数据库操作、获取应用程序的 MD5 码，接下来进行界面的开发。



多学一招：获取 MD5 码工具


市面上有很多获取 MD5 码的工具，例如 MD5Count.exe 就是其中的一个，他的图标是一个骷髅头 ，看起来很霸气。在使用这个工具时，直接双击它的图标运行程序，然后将要计算的文件直接拖到该界面中即可生成 MD5 码，如图 9-3 所示。



图 9-3 MD5 计算工具

需要注意的是，每个程序的 MD5 码都是唯一的。图 10-3 中的两个文件内容是相同的，只不过名字不同而已，因此获得到的 MD5 码是一样的。

9.3 手机杀毒之应用信息展示

9.3.1 应用信息展示主界面

观察完整版的界面效果图可知，如下图 9-4 所示，其中 9-4（a）是完整效果图，9-4（b）是用于简单实现展示应用程序信息的效果图。由此可知，主界面的上面是扫描的进度信息（包含一系列的动画效果），下面是 ListView 展示扫描的应用程序信息。

这里我们首先实现 ListView 的应用信息展示，创建手机杀毒 AntivirusActivity.java 界面的布局文件 activity_antivirus.xml，具体的代码如文件【9-3】所示。



图 9-4 病毒查杀 ListView 的 Item 布局效果

【文件 9-3】 res/layout/activity_antivirus.xml

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.     android:layout_width="match_parent"
4.     android:layout_height="match_parent"
5.     android:orientation="vertical" >
6.
7.     <TextView
8.         style="@style/TitleBarTextView"
9.         android:gravity="center"
10.        android:text="手机杀毒" />
11.     <FrameLayout
12.         android:layout_width="match_parent"
13.         android:layout_height="150dp"
14.         android:background="#4A86CE" >
```



```
15.         <RelativeLayout
16.             android:id="@+id/rl_content"
17.             android:layout_width="match_parent"
18.             android:layout_height="150dp" >
19.
20.             <TextView
21.                 android:id="@+id/tv_progress"
22.                 android:layout_width="wrap_content"
23.                 android:layout_height="wrap_content"
24.                 android:layout_centerInParent="true"
25.                 android:text="18%"
26.                 android:textColor="#fff"
27.                 android:textSize="50sp" />
28.             <TextView
29.                 android:id="@+id/tv_app_package_name"
30.                 android:layout_width="wrap_content"
31.                 android:layout_height="wrap_content"
32.                 android:layout_below="@id/tv_progress"
33.                 android:layout_centerInParent="true"
34.                 android:layout_marginTop="5dp"
35.                 android:text="包名"
36.                 android:textColor="#fff" />
37.         </RelativeLayout>
38.     </FrameLayout>
39.
40.     <ListView
41.         android:id="@+id/list_view"
42.         android:layout_width="match_parent"
43.         android:layout_height="match_parent" >
44.     </ListView>
45. </LinearLayout>
```

9.3.2 应用信息展示之 ListView 的 Item 布局

由于病毒查杀界面中使用了 ListView 控件，因此需要定义一个 Item 布局用于展示应用程序信息，如下图所示 9-5 所示。ListView 的条目布局的具体代码如【文件 9-4】所示。

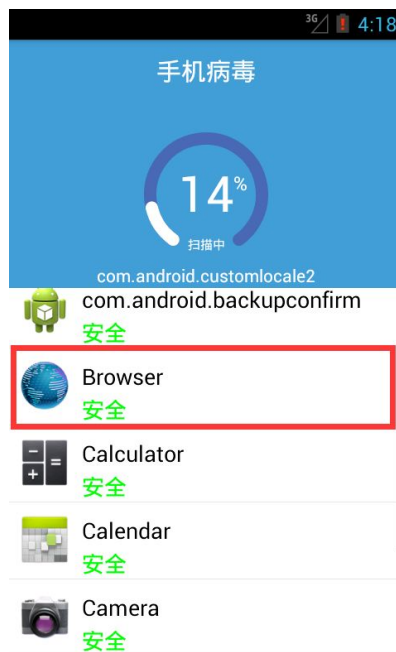


图 9-5 病毒查杀 ListView 的 Item 布局效果

【文件 9-4】 item_virus_activity.xml

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.     android:layout_width="match_parent"
4.     android:layout_height="wrap_content" >
5.     <ImageView
6.         android:id="@+id/iv_icon"
7.         android:layout_width="40dp"
8.         android:layout_height="40dp"
9.         android:layout_margin="10dp"
10.        android:src="@drawable/ic_launcher" />
11.    <TextView
12.        android:id="@+id/tv_app_name"
13.        android:layout_width="wrap_content"
14.        android:layout_height="wrap_content"
15.        android:layout_marginTop="10dp"
16.        android:layout_toRightOf="@id/iv_icon"
17.        android:text="应用的名字"
18.        android:textColor="#000"
19.        android:textSize="18sp" />
20.    <TextView
21.        android:id="@+id/tv_chache_size"
22.        android:layout_width="wrap_content"
23.        android:layout_height="wrap_content"
24.        android:layout_below="@id/tv_app_name"
25.        android:layout_marginTop="2dp"
26.        android:layout_toRightOf="@id/iv_icon"
```

```
27.         android:text="安全"
28.         android:textColor="#88000000"
29.         android:textSize="18sp" />
30.     <ImageButton
31.         android:id="@+id/ib_list_button_clean"
32.         android:layout_width="wrap_content"
33.         android:layout_height="wrap_content"
34.         android:layout_alignParentRight="true"
35.         android:layout_margin="10dp"
36.         android:src="@drawable/list_button_clean_selector"
37.         android:background="@android:color/transparent" />
38. </RelativeLayout>
```

上述布局中,使用了一个 ImageView 控件、两个 TextView 控件和一个 ImageButton 控件,其中 ImageView 控件用于展示应用程序图标,第 1 个 TextView 控件用于展示应用程序的名称,第 2 个 TextView 用于展示应用程序是否安,ImageButton 按钮用于点击清理当前应用程序。

9.3.3 应用信息的实体类

由于 ListView 列表显示的是手机中应用程序列表,其中包括应用名称、包名、应用图标以及该应用是否是病毒。因此首先需要根据这些信息定义一个应用程序的实体类,具体代码如【文件 9-5】所示。

【文件 9-5】 com.itheima.mobilesafe_sh2.bean/AntivirusInfo.java

```
1. public class AntivirusInfo {
2.     public Drawable icon;
3.     public String appName;
4.     public String appPackageName;
5.     public boolean isVirus;
6. }
```

9.3.4 应用信息展示之数据初始化

布局文件我们有了之后,下面就要开始初始化数据,这里我们使用 AsyncTask 机制进行异步获取手机应用信息,具体的代码如文件【9-6】所示。

【文件 9-6】 com.itheima.mobilesafe_sh2.act/AntivirusActivity.java

```
1.     public class AntivirusActivity extends Activity {
2.         private ListView mListView;
3.         private TextView mTvAppPackageName;
4.         private TextView mTvProgress;
5.         private ScanTask task;
6.         private PackageManager mPackageManager;
7.         @Override
8.         protected void onCreate(Bundle savedInstanceState) {
9.             super.onCreate(savedInstanceState);
```

```
10.         iniview();
11.         inidata();
12.     }
13.     private List<AntivirusInfo> mDatas;
14.     private AntivirusAdapter adapter;
15.     private class ScanTask extends AsyncTask<Void, AntivirusInfo, Void> {
16.         @Override
17.         protected Void doInBackground(Void... params) {
18.             List<PackageInfo> packages = mPackageManager
19.                 .getInstalledPackages(0);
20.             mDatas = new ArrayList<AntivirusInfo>();
21.             try {
22.                 for (PackageInfo packageInfo : packages) {
23.                     AntivirusInfo info = new AntivirusInfo();
24.                     String appName = packageInfo.applicationInfo.loadLabel(
25.                         mPackageManager).toString();
26.                     Drawable icon = packageInfo.applicationInfo
27.                         .loadIcon(mPackageManager);
28.                     String appPackageName = packageInfo.applicationInfo.packageName;
29.                     // 获取所有文件的路径
30.                     String path = packageInfo.applicationInfo.sourceDir;
31.                     FileInputStream is = new FileInputStream(path);
32.                     String md5 = MD5Utils.encode(is);
33.                     // 查询数据库是否有病毒
34.                     boolean isVirus = AntivirusDao.isVirus(
35.                         getApplicationContext(), md5);
36.                     info.icon = icon;
37.                     info.appName = appName;
38.                     info.isVirus = isVirus;
39.                     info.appPackageName = appPackageName;
40.                     // 如果有病毒。放到上面
41.                     if (info.isVirus) {
42.                         mDatas.add(0, info);
43.                     } else {
44.                         mDatas.add(info);
45.                     }
46.                     publishProgress(info);
47.                     SystemClock.sleep(100);
48.                 }
49.             } catch (FileNotFoundException e) {
50.                 // TODO Auto-generated catch block
51.                 e.printStackTrace();
52.             }
53.             return null;
```

```
54.     }
55.     @Override
56.     protected void onPostExecute(Void result) {
57.         // TODO Auto-generated method stub
58.         super.onPostExecute(result);
59.     }
60.     @Override
61.     protected void onProgressUpdate(AntivirusInfo... values) {
62.         // TODO Auto-generated method stub
63.         super.onProgressUpdate(values);
64.         AntivirusInfo info = values[0];
65.         if (adapter == null) {
66.             adapter = new AntivirusAdapter();
67.             mListView.setAdapter(adapter);
68.         } else {
69.             adapter.notifyDataSetChanged();
70.         }
71.         // 自己滚动到最后
72.         mListView.smoothScrollToPosition(adapter.getCount());
73.     }
74. }
75. private void inidata() {
76.     if (null != task) {
77.         task = null;
78.     }
79.     task = new ScanTask();
80.     task.execute();
81. }
82. private void iniview() {
83.     setContentView(R.layout.activity_antivirus);
84.     mListView = (ListView) findViewById(R.id.list_view);
85.     mTvProgress = (TextView) findViewById(R.id.tv_progress);
86.     mTvAppPackageName = (TextView) findViewById(R.id.tv_app_package_name);
87.     mPackageManager = getPackageManager();
88. }
89. }
```

9.3.5 应用信息展示之数据适配器

初始化数据之后，这里我们需要创建一个数据适配器来展示数据，具体的代码如下所示。

```
1.     private class AntivirusAdapter extends BaseAdapter {
2.         @Override
3.         public int getCount() {
```

```
4.         return mDataas.size();
5.     }
6.     @Override
7.     public Object getItem(int position) {
8.         return mDataas.get(position);
9.     }
10.    @Override
11.    public long getItemId(int position) {
12.        return position;
13.    }
14.    @Override
15.    public View getView(int position, View convertView, ViewGroup parent) {
16.        if (convertView == null) {
17.            convertView = View.inflate(getApplicationContext(),
18.                R.layout.item_virus_activity, null);
19.        }
20.        ImageView iv_icon = (ImageView) convertView
21.            .findViewById(R.id.iv_icon);
22.        // 应用的名字
23.        TextView tv_app_name = (TextView) convertView
24.            .findViewById(R.id.tv_app_name);
25.        // 是否有病毒
26.        TextView tv_chache_size = (TextView) convertView
27.            .findViewById(R.id.tv_chache_size);
28.        // 清理病毒
29.        ImageButton ib_list_button_clean = (ImageButton) convertView
30.            .findViewById(R.id.ib_list_button_clean);
31.        AntivirusInfo antivirusInfo = mDataas.get(position);
32.        iv_icon.setImageDrawable(antivirusInfo.icon);
33.        tv_app_name.setText(antivirusInfo.appName);
34.        //判断是否有病毒
35.        if(antivirusInfo.isVirus){
36.            tv_chache_size.setTextColor(Color.RED);
37.        }else{
38.            tv_chache_size.setTextColor(Color.GREEN);
39.        }
40.        tv_chache_size.setText(antivirusInfo.isVirus ? "病毒" : "安全");
41.        ib_list_button_clean.setVisibility(antivirusInfo.isVirus ? View.VISIBLE
42.            : View.INVISIBLE);
43.        return convertView;
44.    }
45. }
```

运行程序，效果图如图 9-6 所示，ListView 中已经将手机中的所有应用信息展示出来。要注意的是这里只实现了主界面下面的应用信息展示，上面的进度条等信息将在下一节中进行介绍。

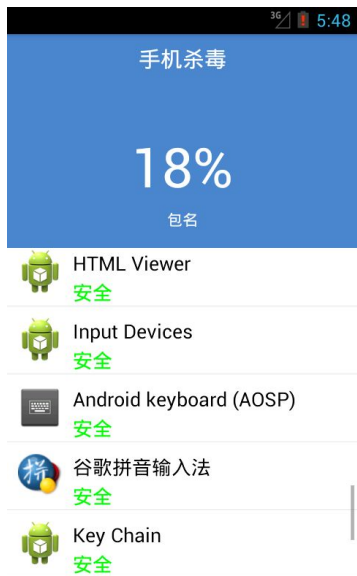


图 9-6 手机病毒之应用信息展示效果图

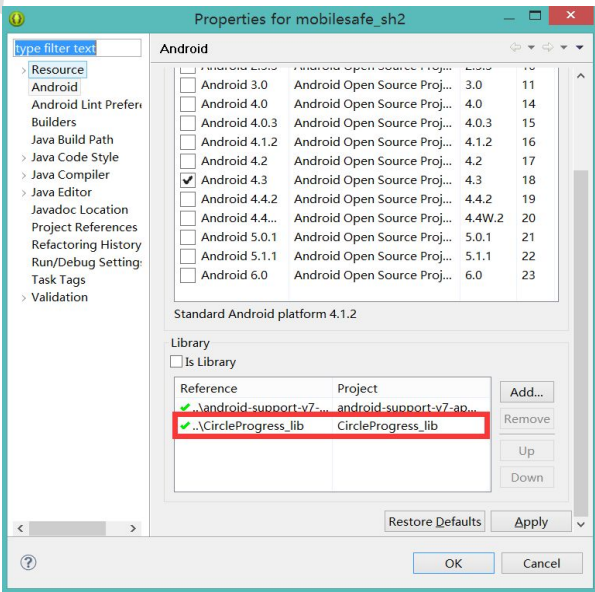
9.4 手机杀毒之查杀进度展示

9.4.1 进度信息界面

手机杀毒界面中，上面的查杀进度信息主要用于显示当前病毒查杀进度以及正在扫描的应用包名。其中，查杀进度的进度条是一个自定义的进度条，它是 GitHub 上的一个已经定义好的第三方框架 CircleProgress_lib，使用的时候直接将这个类库导入项目即可，如图 9-7（b）。另外，自定义进度的中间是一个 TextView 用于按百分比显示查杀进度，最下面有一个 TextView 用于展示当前扫描应用信息的包名。查杀进度的效果图如图 9-7（a）所示。



(a)



(b)

图 9-7 手机病毒查杀进度效果图

这里，我们需要重新改写主界面的布局代码，将第三方的进度条引入当前布局中。

图 9-7 查杀进度界面所对应的布局文件如【文件 9-7】所示。

【文件 9-7】res/layout/activity_antivirus.xml

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.     xmlns:custom="http://schemas.android.com/apk/res/com.itheima.mobilesafe_sh2"
4.     android:layout_width="match_parent"
5.     android:layout_height="match_parent"
6.     android:orientation="vertical" >
7.     <TextView
8.         style="@style/TitleBarTextView"
9.         android:gravity="center"
10.        android:text="手机杀毒" />
11.    <FrameLayout
12.        android:layout_width="match_parent"
13.        android:layout_height="150dp"
14.        android:background="#4A86CE" >
15.        <RelativeLayout
16.            android:id="@+id/rl_content"
17.            android:layout_width="match_parent"
18.            android:layout_height="150dp" >
19.            <!-- <TextView
20.                android:id="@+id/tv_progress"
21.                android:layout_width="wrap_content"
22.                android:layout_height="wrap_content"
23.                android:layout_centerInParent="true"
24.                android:text="18%"
25.                android:textColor="#fff"
26.                android:textSize="50sp" /> -->
27.            <com.github.lzyzsd.circleprogress.ArcProgress
28.                android:id="@+id/tv_progress"
29.                android:layout_width="100dp"
30.                android:layout_height="100dp"
31.                android:layout_centerInParent="true"
32.                android:layout_marginLeft="50dp"
33.                android:background="#4A86CE"
34.                custom:arc_bottom_text="扫描中"
35.                custom:arc_progress="55"
36.                custom:arc_stroke_width="10dp"
37.                custom:arc_text_color="#fff" />
38.            <TextView
39.                android:id="@+id/tv_app_package_name"
40.                android:layout_width="wrap_content"
41.                android:layout_height="wrap_content"
```



```
42.         android:layout_below="@id/tv_progress"
43.         android:layout_centerInParent="true"
44.         android:layout_marginTop="5dp"
45.         android:text="包名"
46.         android:textColor="#fff" />
47.     </RelativeLayout>
48.
49.     <LinearLayout
50.         android:id="@+id/ll_result_content"
51.         android:layout_width="match_parent"
52.         android:layout_height="150dp"
53.         android:gravity="center"
54.         android:orientation="vertical"
55.         android:visibility="invisible" >
56.         <TextView
57.             android:id="@+id/tv_is_virus"
58.             android:layout_width="wrap_content"
59.             android:layout_height="wrap_content"
60.             android:text="您的手机很安全"
61.             android:textColor="#fff"
62.             android:textSize="20sp" />
63.         <Button
64.             android:id="@+id/bt_scan"
65.             android:layout_width="wrap_content"
66.             android:layout_height="wrap_content"
67.             android:text="重新扫描"
68.             android:textColor="#fff"
69.             android:textSize="20sp" />
70.     </LinearLayout>
71. </FrameLayout>
72. <ListView
73.     android:id="@+id/list_view"
74.     android:layout_width="match_parent"
75.     android:layout_height="match_parent" >
76. </ListView>
77. </LinearLayout>
```

跟之前的布局文件相比，这里我们只是将之前的显示进度的 `TextView` 换成了第三方的 `ArcProgress`。

9.4.2 进度信息初始化

这里我们需要更新主界面上面的进度信息，具体的逻辑主要在 `AsyncTask` 中，具体代码如【文件 9-8】所示。

【文件 9-8】 com.itheima.mobilesafe_sh2.act/AntivirusActivity.java 部分代码

```
1.     private class ScanTask extends AsyncTask<Void, AntivirusInfo, Void> {
2.         private int progress = 0;
3.         private int max = 0;
4.         @Override
5.         protected Void doInBackground(Void... params) {
6.             List<PackageInfo> packages = mPackageManager
7.                 .getInstalledPackages(0);
8.             max = packages.size();
9.             mDatas = new ArrayList<AntivirusInfo>();
10.            try {
11.                for (PackageInfo packageInfo : packages) {
12.                    progress++;
13.                    AntivirusInfo info = new AntivirusInfo();
14.                    String appName = packageInfo.applicationInfo.loadLabel(
15.                        mPackageManager).toString();
16.                    Drawable icon = packageInfo.applicationInfo
17.                        .loadIcon(mPackageManager);
18.                    String appPackageName = packageInfo.applicationInfo.packageName;
19.                    // 获取所有文件的路径
20.                    String path = packageInfo.applicationInfo.sourceDir;
21.                    FileInputStream is = new FileInputStream(path);
22.                    String md5 = MD5Utils.encode(is);
23.                    // 查询数据库是否有病毒
24.                    boolean isVirus = AntivirusDao.isVirus(
25.                        getApplicationContext(), md5);
26.                    info.icon = icon;
27.                    info.appName = appName;
28.                    info.isVirus = isVirus;
29.                    info.appPackageName = appPackageName;
30.                    // 如果有病毒。放到上面
31.                    if (info.isVirus) {
32.                        mDatas.add(0, info);
33.                    } else {
34.                        mDatas.add(info);
35.                    }
36.                    publishProgress(info);
37.                    SystemClock.sleep(100);
38.                }
39.            } catch (FileNotFoundException e) {
40.                e.printStackTrace();
41.            }
42.            return null;
43.        }
```

```
44.         @Override
45.         protected void onPostExecute(Void result) {
46.             // TODO Auto-generated method stub
47.             super.onPostExecute(result);
48.         }
49.         @Override
50.         protected void onProgressUpdate(AntivirusInfo... values) {
51.             // TODO Auto-generated method stub
52.             super.onProgressUpdate(values);
53.             AntivirusInfo info = values[0];
54.             int mCurrentProgress = (int) (progress * 100f / max);
55.             // mTvProgress
56.             // 设置进度
57.             mTvProgress.setProgress(mCurrentProgress);
58.             // 设置包名
59.             mTvAppPackageName.setText(info.appPackageName);
60.             if (adapter == null) {
61.                 adapter = new AntivirusAdapter();
62.                 mListView.setAdapter(adapter);
63.             } else {
64.                 adapter.notifyDataSetChanged();
65.             }
66.             // 自己滚动到最后
67.             mListView.scrollToPosition(adapter.getCount());
68.         }
69.     }
```

上述进度信息设置之后，运行程序，效果图如图 9-8 所示。

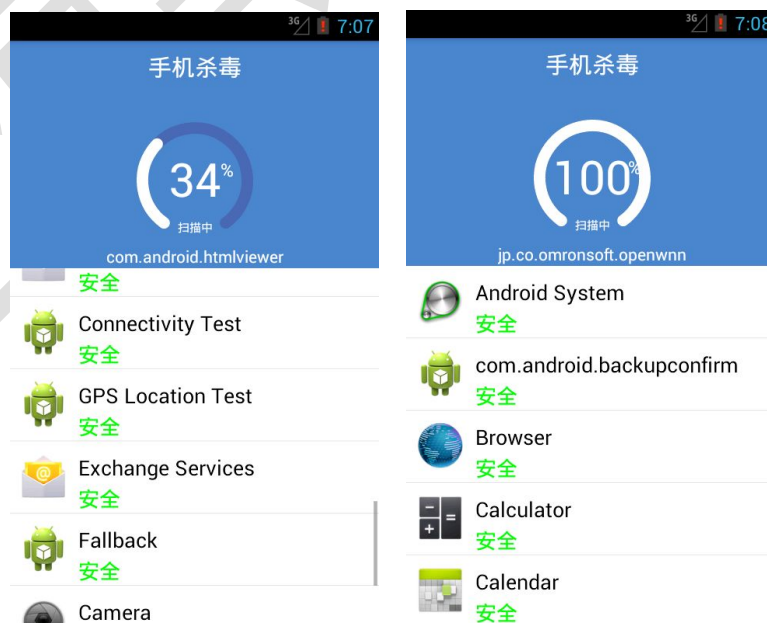


图 9-8 手机病毒之查杀进度更新效果图

9.4.3 查杀结束界面

观察完整的效果图，当扫描手机应用程序结束之后，界面的效果图如图 9-9 所示，

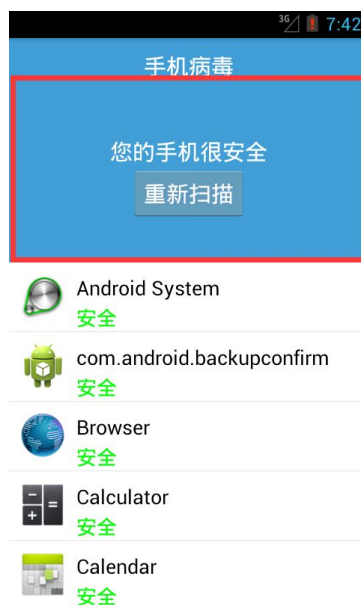


图 9-9 手机病毒之查杀结束后的画面

首先，我们需要修改主界面的布局文件 `activity_antivirus.xml`，修改后的代码文件如【9-9】所示。

【文件 9-9】 `res/layout/activity_antivirus.xml`

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.     xmlns:custom="http://schemas.android.com/apk/res/com.itheima.mobilesafe_sh2"
4.     android:layout_width="match_parent"
5.     android:layout_height="match_parent"
6.     android:orientation="vertical" >
7.     <TextView
8.         style="@style/TitleBarTextView"
9.         android:gravity="center"
10.        android:text="手机杀毒" />
11.     <FrameLayout
12.        android:layout_width="match_parent"
13.        android:layout_height="150dp"
14.        android:background="#4A86CE" >
15.        <RelativeLayout
16.            android:id="@+id/rl_content"
17.            android:layout_width="match_parent"
18.            android:layout_height="150dp" >
19.            <!-- <TextView
20.                android:id="@+id/tv_progress"
21.                android:layout_width="wrap_content"
22.                android:layout_height="wrap_content"
```

```
23.         android:layout_centerInParent="true"
24.         android:text="18%"
25.         android:textColor="#fff"
26.         android:textSize="50sp" /> -->
27.     <com.github.lzyzsd.circleprogress.ArcProgress
28.         android:id="@+id/tv_progress"
29.         android:layout_width="100dp"
30.         android:layout_height="100dp"
31.         android:layout_centerInParent="true"
32.         android:layout_marginLeft="50dp"
33.         android:background="#4A86CE"
34.         custom:arc_bottom_text="扫描中"
35.         custom:arc_progress="55"
36.         custom:arc_stroke_width="10dp"
37.         custom:arc_text_color="#fff" />
38.     <TextView
39.         android:id="@+id/tv_app_package_name"
40.         android:layout_width="wrap_content"
41.         android:layout_height="wrap_content"
42.         android:layout_below="@id/tv_progress"
43.         android:layout_centerInParent="true"
44.         android:layout_marginTop="5dp"
45.         android:text="包名"
46.         android:textColor="#fff" />
47. </RelativeLayout>
48. <LinearLayout
49.     android:id="@+id/ll_result_content"
50.     android:layout_width="match_parent"
51.     android:layout_height="150dp"
52.     android:gravity="center"
53.     android:orientation="vertical"
54.     android:visibility="invisible" >
55.     <TextView
56.         android:id="@+id/tv_is_virus"
57.         android:layout_width="wrap_content"
58.         android:layout_height="wrap_content"
59.         android:text="您的手机很安全"
60.         android:textColor="#fff"
61.         android:textSize="20sp" />
62.     <Button
63.         android:id="@+id/bt_scan"
64.         android:layout_width="wrap_content"
65.         android:layout_height="wrap_content"
```

```

66.         android:text="重新扫描"
67.         android:textColor="#fff"
68.         android:textSize="20sp" />
69.     </LinearLayout>
70. </FrameLayout>
71. <ListView
72.     android:id="@+id/list_view"
73.     android:layout_width="match_parent"
74.     android:layout_height="match_parent" >
75. </ListView>
76. </LinearLayout>

```

跟前面相比，这里新增加了一个线性布局，用于展示扫描结束的结果，如红色字体标注。接下来就需要在主界面的逻辑代码中进行修改，让手机在扫描结束之后显示扫描结果，代码如下所示，如红色字体标注。

```

1.  public class AntivirusActivity extends Activity implements OnClickListener {
2.      private ListView mListView;
3.      private TextView mTvAppPackageName;
4.      private ArcProgress mTvProgress;
5.      private ScanTask task;
6.      private PackageManager mPackageManager;
7.      private LinearLayout ll_result_content;
8.      private TextView tv_is_virus;
9.      private RelativeLayout rl_content;
10.     private Button bt_scan;
11.     private List<AntivirusInfo> mDatas;
12.     private AntivirusAdapter adapter;
13.     @Override
14.     protected void onCreate(Bundle savedInstanceState) {
15.         super.onCreate(savedInstanceState);
16.         initView();
17.         initData();
18.     }
19.     //用于当手机杀毒界面失去焦点时停止扫描
20.     @Override
21.     protected void onPause() {
22.         super.onPause();
23.         if (null != task) {
24.             task.stop();
25.             task = null;
26.         }
27.     }
28.     private class ScanTask extends AsyncTask<Void, AntivirusInfo, Void> {
29.         private int progress = 0;
30.         private int max = 0;

```

```
31.     private boolean isFinish = false;
32.     @Override
33.     protected void onPreExecute() {
34.         super.onPreExecute();
35.         // 扫描开始的时候。扫描结果不可见
36.         ll_result_content.setVisibility(View.INVISIBLE);
37.         // 扫描开始的时候。扫描内容可见
38.         rl_content.setVisibility(View.VISIBLE);
39.     }
40.     public void stop() {
41.         isFinish = true;
42.     }
43.     @Override
44.     protected void doInBackground(Void... params) {
45.         List<PackageInfo> packages = mPackageManager
46.             .getInstalledPackages(0);
47.         max = packages.size();
48.         mDatas = new ArrayList<AntivirusInfo>();
49.         try {
50.             for (PackageInfo packageInfo : packages) {
51.                 if (isFinish) {
52.                     break;
53.                 }
54.                 progress++;
55.                 AntivirusInfo info = new AntivirusInfo();
56.                 String appName = packageInfo.applicationInfo.loadLabel(
57.                     mPackageManager).toString();
58.                 Drawable icon = packageInfo.applicationInfo
59.                     .loadIcon(mPackageManager);
60.                 String appPackageName = packageInfo.applicationInfo.packageName;
61.                 // 获取所有文件的路径
62.                 String path = packageInfo.applicationInfo.sourceDir;
63.                 FileInputStream is = new FileInputStream(path);
64.                 String md5 = MD5Utils.encode(is);
65.                 // 查询数据库是否有病毒
66.                 boolean isVirus = AntivirusDao.isVirus(
67.                     getApplicationContext(), md5);
68.                 info.icon = icon;
69.                 info.appName = appName;
70.                 info.isVirus = isVirus;
71.                 info.appPackageName = appPackageName;
72.                 // 如果有病毒。放到上面
73.                 if (info.isVirus) {
```

```
74.         mDatas.add(0, info);
75.     } else {
76.         mDatas.add(info);
77.     }
78.     publishProgress(info);
79.     SystemClock.sleep(100);
80. }
81. } catch (FileNotFoundException e) {
82.     // TODO Auto-generated catch block
83.     e.printStackTrace();
84. }
85. return null;
86. }
87. @Override
88. protected void onPostExecute(Void result) {
89.     // TODO Auto-generated method stub
90.     super.onPostExecute(result);
91.     if (isFinish) {
92.         return;
93.     }
94.     // 自己滚动到最后
95.     mListView.smoothScrollToPosition(0);
96.     // 扫描完成之后。扫描结果可见
97.     ll_result_content.setVisibility(View.VISIBLE);
98.     // 扫描完成之后不可见
99.     rl_content.setVisibility(View.INVISIBLE);
100. }
101. @Override
102. protected void onProgressUpdate(AntivirusInfo... values) {
103.     // TODO Auto-generated method stub
104.     super.onProgressUpdate(values);
105.     if (isFinish) {
106.         return;
107.     }
108.     AntivirusInfo info = values[0];
109.     int mCurrentProgress = (int) (progress * 100f / max);
110.     // mTvProgress
111.     // 设置进度
112.     mTvProgress.setProgress(mCurrentProgress);
113.     // 设置包名
114.     mTvAppPackageName.setText(info.appPackageName);
115.     if (adapter == null) {
116.         adapter = new AntivirusAdapter();
117.         mListView.setAdapter(adapter);
```



```
118.         } else {
119.             adapter.notifyDataSetChanged();
120.         }
121.         // 自己滚动到最后
122.         mListview.smoothScrollToPosition(adapter.getCount());
123.     }
124. }
125. private class AntivirusAdapter extends BaseAdapter {
126.     @Override
127.     public int getCount() {
128.         return mDatas.size();
129.     }
130.     @Override
131.     public Object getItem(int position) {
132.         return mDatas.get(position);
133.     }
134.     @Override
135.     public long getItemId(int position) {
136.         return position;
137.     }
138.     @Override
139.     public View getView(int position, View convertView, ViewGroup parent) {
140.         if (convertView == null) {
141.             convertView = View.inflate(getApplicationContext(),
142.                 R.layout.item_virus_activity, null);
143.         }
144.         ImageView iv_icon = (ImageView) convertView
145.             .findViewById(R.id.iv_icon);
146.         // 应用的名字
147.         TextView tv_app_name = (TextView) convertView
148.             .findViewById(R.id.tv_app_name);
149.         // 是否有病毒
150.         TextView tv_cache_size = (TextView) convertView
151.             .findViewById(R.id.tv_cache_size);
152.         // 清理病毒
153.         ImageButton ib_list_button_clean = (ImageButton) convertView
154.             .findViewById(R.id.ib_list_button_clean);
155.         AntivirusInfo antivirusInfo = mDatas.get(position);
156.         iv_icon.setImageDrawable(antivirusInfo.icon);
157.         tv_app_name.setText(antivirusInfo.appName);
158.         // 判断是否有病毒
159.         if (antivirusInfo.isVirus) {
160.             tv_cache_size.setTextColor(Color.RED);
```

```
161.         } else {
162.             tv_cache_size.setTextColor(Color.GREEN);
163.         }
164.         tv_cache_size.setText(antivirusInfo.isVirus ? "病毒" : "安全");
165.         ib_list_button_clean.setVisibility(antivirusInfo.isVirus ? View.VISIBLE
166.             : View.INVISIBLE);
167.         return convertView;
168.     }
169. }
170. private void inidata() {
171.     if (null != task) {
172.         task = null;
173.     }
174.     task = new ScanTask();
175.     task.execute();
176. }
177. private void iniview() {
178.     setContentView(R.layout.activity_antivirus);
179.     mListView = (ListView) findViewById(R.id.list_view);
180.     mTvProgress = (ArcProgress) findViewById(R.id.tv_progress);
181.     mTvAppPackageName = (TextView) findViewById(R.id.tv_app_package_name);
182.     mPackageManager = getPackageManager();
183.     // 扫描的结果
184.     ll_result_content = (LinearLayout) findViewById(R.id.ll_result_content);
185.     tv_is_virus = (TextView) findViewById(R.id.tv_is_virus);
186.     // 重新扫描
187.     bt_scan = (Button) findViewById(R.id.bt_scan);
188.     bt_scan.setOnClickListener(this);
189.     // 扫描的内容
190.     rl_content = (RelativeLayout) findViewById(R.id.rl_content);
191. }
192. @Override
193. public void onClick(View v) {
194.     // TODO Auto-generated method stub
195.     inidata();
196. }
197. }
```

运行程序，效果图如图 9-10 所示。



图 9-10 手机病毒查杀结束后重新扫描效果图

9.4.4 查杀结束动画的布局

手机杀毒扫描应用程序结束之后是有一个动画效果的，即有向左和向右的两张平移动画，效果图如下图 9-11 所示，

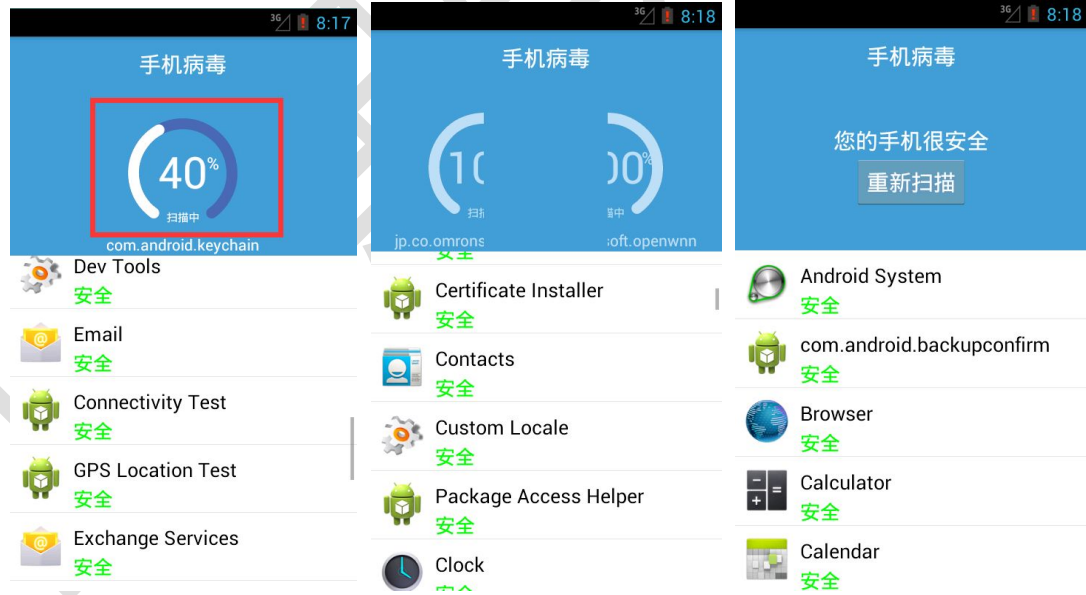


图 9-11 手机病毒查杀结束后的动画效果图

下面，我们需要分析一下这两个移动的图片，分析图如图 9-12 所示，可知这两张图片是根据扫描结果的图片一分为二画出来的。我们可以在布局中使用 `ImageView` 来实现这样的图片动画效果。



图 9-12 手机病毒查杀结束后动画效果分析图

首先，需要修改之前主界面的布局文件，具体代码如下所示。

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.     xmlns:custom="http://schemas.android.com/apk/res/com.itheima.mobilesafe_sh2"
4.     android:layout_width="match_parent"
5.     android:layout_height="match_parent"
6.     android:orientation="vertical" >
7.     <TextView
8.         style="@style/TitleBarTextView"
9.         android:gravity="center"
10.        android:text="手机杀毒" />
11.     <FrameLayout
12.         android:layout_width="match_parent"
13.         android:layout_height="150dp"
14.         android:background="#4A86CE" >
15.         <RelativeLayout
16.             android:id="@+id/rl_content"
17.             android:layout_width="match_parent"
18.             android:layout_height="150dp" >
19.             <com.github.lzyzsd.circleprogress.ArcProgress
20.                 android:id="@+id/tv_progress"
21.                 android:layout_width="100dp"
22.                 android:layout_height="100dp"
23.                 android:layout_centerInParent="true"
24.                 android:layout_marginLeft="50dp"
25.                 android:background="#4A86CE"
26.                 custom:arc_bottom_text="扫描中"
```

```
27.         custom:arc_progress="55"
28.         custom:arc_stroke_width="10dp"
29.         custom:arc_text_color="#fff" />
30.     <TextView
31.         android:id="@+id/tv_app_package_name"
32.         android:layout_width="wrap_content"
33.         android:layout_height="wrap_content"
34.         android:layout_below="@id/tv_progress"
35.         android:layout_centerInParent="true"
36.         android:layout_marginTop="5dp"
37.         android:text="包名"
38.         android:textColor="#fff" />
39. </RelativeLayout>
40. <LinearLayout
41.     android:id="@+id/ll_result_content"
42.     android:layout_width="match_parent"
43.     android:layout_height="150dp"
44.     android:gravity="center"
45.     android:orientation="vertical"
46.     android:visibility="invisible" >
47.     <TextView
48.         android:id="@+id/tv_is_virus"
49.         android:layout_width="wrap_content"
50.         android:layout_height="wrap_content"
51.         android:text="您的手机很安全"
52.         android:textColor="#fff"
53.         android:textSize="20sp" />
54.     <Button
55.         android:id="@+id/bt_scan"
56.         android:layout_width="wrap_content"
57.         android:layout_height="wrap_content"
58.         android:text="重新扫描"
59.         android:textColor="#fff"
60.         android:textSize="20sp" />
61. </LinearLayout>
62. <!--用于展示图片移动的动画-->
63.     <LinearLayout
64.         android:id="@+id/ll_anim_content"
65.         android:layout_width="match_parent"
66.         android:layout_height="150dp"
67.         android:gravity="center"
68.         android:orientation="horizontal"
69.         android:visibility="invisible" >
```

```
70.         <ImageView
71.             android:id="@+id/iv_left"
72.             android:layout_width="wrap_content"
73.             android:layout_height="match_parent"
74.             android:layout_weight="1" />
75.         <ImageView
76.             android:id="@+id/iv_right"
77.             android:layout_width="wrap_content"
78.             android:layout_height="match_parent"
79.             android:layout_weight="1" />
80.     </LinearLayout>
81. </FrameLayout>
82. <ListView
83.     android:id="@+id/list_view"
84.     android:layout_width="match_parent"
85.     android:layout_height="match_parent" >
86. </ListView>
87. </LinearLayout>
```

9.4.5 查杀结束动画的逻辑实现

可知，当手机病毒扫描结束之后，动画需要在 `onPostExecute()` 中实现，此时两张图片是向两边移动的，而当用户点击重新扫描按钮时，动画效果是从两边向中间来的。

接下来，我们就需要修改主界面的代码，具体的逻辑如下所示。

```
1. public class AntivirusActivity extends Activity implements OnClickListener {
2.     private ListView mListView;
3.     private TextView mTvAppPackageName;
4.     private ArcProgress mTvProgress;
5.     private ScanTask task;
6.     private PackageManager mPackageManager;
7.     private LinearLayout ll_result_content;
8.     private TextView tv_is_virus;
9.     private RelativeLayout rl_content;
10.    private Button bt_scan;
11.    private List<AntivirusInfo> mDatas;
12.    private AntivirusAdapter adapter;
13.    private LinearLayout ll_anim_content;
14.    private ImageView iv_left;
15.    private ImageView iv_right;
16.    @Override
17.    protected void onCreate(Bundle savedInstanceState) {
18.        super.onCreate(savedInstanceState);
19.        initView();
```

```
20.         inidata();
21.     }
22.     // 用于当手机杀毒界面失去焦点时停止扫描
23.     @Override
24.     protected void onPause() {
25.         super.onPause();
26.         if (null != task) {
27.             task.stop();
28.             task = null;
29.         }
30.     }
31.     private class ScanTask extends AsyncTask<Void, AntivirusInfo, Void> {
32.         private int progress = 0;
33.         private int max = 0;
34.         private boolean isFinish = false;
35.         @Override
36.         protected void onPreExecute() {
37.             super.onPreExecute();
38.             // 扫描开始的时候。扫描结果不可见
39.             ll_result_content.setVisibility(View.INVISIBLE);
40.             // 扫描开始的时候。扫描内容可见
41.             rl_content.setVisibility(View.VISIBLE);
42.             // 扫描开始关闭动画不可见
43.             ll_anim_content.setVisibility(View.INVISIBLE);
44.             // 重新扫描不可以被点击
45.             bt_scan.setEnabled(false);
46.         }
47.         public void stop() {
48.             isFinish = true;
49.         }
50.         @Override
51.         protected Void doInBackground(Void... params) {
52.             List<PackageInfo> packages = mPackageManager.getInstalledPackages(0);
53.             max = packages.size();
54.             mDatas = new ArrayList<AntivirusInfo>();
55.             try {
56.                 for (PackageInfo packageInfo : packages) {
57.                     if (isFinish) {
58.                         break;
59.                     }
60.                     progress++;
61.                     AntivirusInfo info = new AntivirusInfo();
62.                     String appName = packageInfo.applicationInfo.loadLabel(
```

```
63.             mPackageManager).toString();
64.             Drawable icon = packageInfo.applicationInfo
65.                 .loadIcon(mPackageManager);
66.             String appPackageName = packageInfo.applicationInfo.packageName;
67.             // 获取所有文件的路径
68.             String path = packageInfo.applicationInfo.sourceDir;
69.             FileInputStream is = new FileInputStream(path);
70.             String md5 = MD5Utils.encode(is);
71.             // 查询数据库是否有病毒
72.             boolean isVirus = AntivirusDao.isVirus(
73.                 getApplicationContext(), md5);
74.             info.icon = icon;
75.             info.appName = appName;
76.             info.isVirus = isVirus;
77.             info.appPackageName = appPackageName;
78.             // 如果有病毒。放到上面
79.             if (info.isVirus) {
80.                 mDatas.add(0, info);
81.             } else {
82.                 mDatas.add(info);
83.             }
84.             publishProgress(info);
85.             SystemClock.sleep(100);
86.         }
87.     } catch (FileNotFoundException e) {
88.         // TODO Auto-generated catch block
89.         e.printStackTrace();
90.     }
91.     return null;
92. }
93. @Override
94. protected void onPostExecute(Void result) {
95.     // TODO Auto-generated method stub
96.     super.onPostExecute(result);
97.     if (isFinish) {
98.         return;
99.     }
100.    // 自己滚动到最后
101.    mListView.smoothScrollToPosition(0);
102.    // 扫描完成之后。扫描结果可见
103.    ll_result_content.setVisibility(View.INVISIBLE);
104.    // 扫描完成之后不可见
105.    rl_content.setVisibility(View.INVISIBLE);
106.    // 扫描完成之后。执行动画
```



```
107.         ll_anim_content.setVisibility(View.VISIBLE);
108.         // 需要把背景图片一分为二
109.         // 设置背景图片，设置为 true 可以从当前 View 对象画出图片
110.         rl_content.setDrawingCacheEnabled(true);
111.         // 设置图片的质量
112.         // 设置一张高清的图片
113.         rl_content.setDrawingCacheQuality(View.DRAWING_CACHE_QUALITY_HIGH);
114.         // 首先获取到背景图片
115.         Bitmap drawingCache = rl_content.getDrawingCache();
116.         // 设置左边的图片
117.         iv_left.setImageBitmap(getBitmapLeft(drawingCache));
118.         // 设置右边的图片
119.         iv_right.setImageBitmap(getBitmapRight(drawingCache));
120.         // 打开背景图片的动画
121.         showOpenAnim();
122.     }
123.     @Override
124.     protected void onProgressUpdate(AntivirusInfo... values) {
125.         // TODO Auto-generated method stub
126.         super.onProgressUpdate(values);
127.         if (isFinish) {
128.             return;
129.         }
130.         AntivirusInfo info = values[0];
131.         int mCurrentProgress = (int) (progress * 100f / max);
132.         // mTvProgress
133.         // 设置进度
134.         mTvProgress.setProgress(mCurrentProgress);
135.         // 设置包名
136.         mTvAppPackageName.setText(info.appPackageName);
137.         if (adapter == null) {
138.             adapter = new AntivirusAdapter();
139.             mListView.setAdapter(adapter);
140.         } else {
141.             adapter.notifyDataSetChanged();
142.         }
143.         // 自己滚动到最后
144.         mListView.smoothScrollToPosition(adapter.getCount());
145.     }
146. }
147. private class AntivirusAdapter extends BaseAdapter {
148.     .....适配器代码省略
149. }
```

```
150.     private void inidata() {
151.         if (null != task) {
152.             task.stop();
153.             task = null;
154.         }
155.         task = new ScanTask();
156.         task.execute();
157.     }
158.     /**
159.      * 打开背景图片的动画
160.      */
161.     public void showOpenAnim() {
162.         AnimatorSet set = new AnimatorSet();
163.         ObjectAnimator animatorTranslationLeft = ObjectAnimator.ofFloat(
164.             iv_left, "translationX", 0, -iv_left.getWidth());
165.         ObjectAnimator animatorTranslationRight = ObjectAnimator.ofFloat(
166.             iv_right, "translationX", 0, iv_right.getWidth());
167.         ObjectAnimator animatorAlphaLeft = ObjectAnimator.ofFloat(iv_left,
168.             "alpha", 1.0f, 0);
169.         ObjectAnimator animatorAlphaRight = ObjectAnimator.ofFloat(iv_right,
170.             "alpha", 1.0f, 0);
171.         set.playTogether(animatorTranslationLeft, animatorTranslationRight,
172.             animatorAlphaLeft, animatorAlphaRight);
173.         set.setDuration(2000);
174.         //动画监听，当动画结束之后，将扫描结果展示
175.         set.addListener(new AnimatorListener() {
176.             @Override
177.             public void onAnimationStart(Animator arg0) {
178.                 // TODO Auto-generated method stub
179.             }
180.             @Override
181.             public void onAnimationRepeat(Animator arg0) {
182.                 // TODO Auto-generated method stub
183.             }
184.             @Override
185.             public void onAnimationEnd(Animator arg0) {
186.                 ll_result_content.setVisibility(View.VISIBLE);
187.                 bt_scan.setEnabled(true);
188.             }
189.             @Override
190.             public void onAnimationCancel(Animator arg0) {
191.                 // TODO Auto-generated method stub
192.             }
193.         });
```

```
194.         set.start();
195.     }
196.     /**
197.      * 设置右边的图片
198.      * @param drawingCache
199.      * @return
200.      */
201.     public Bitmap getBitmapRight(Bitmap drawingCache) {
202.         // 得到原图的一半
203.         int width = drawingCache.getWidth() / 2;
204.         int height = drawingCache.getHeight();
205.         // 获取到图片
206.         Bitmap createBitmap = Bitmap.createBitmap(width, height,
207.             drawingCache.getConfig());
208.         //将白纸铺在画布上
209.         Canvas canvas = new Canvas(createBitmap);
210.         Paint paint = new Paint();
211.         Matrix matrix = new Matrix();
212.         matrix.setTranslate(-width, 0);
213.         // 绘制图片
214.         canvas.drawBitmap(drawingCache, matrix, paint);
215.         return createBitmap;
216.     }
217.     /**
218.      * 获取到左边的图片
219.      * @param drawingCache 原图
220.      * @return
221.      */
222.     public Bitmap getBitmapLeft(Bitmap drawingCache) {
223.         // 得到原图的一半
224.         int width = drawingCache.getWidth() / 2;
225.         int height = drawingCache.getHeight();
226.         // 获取到图片
227.         Bitmap createBitmap = Bitmap.createBitmap(width, height,
228.             drawingCache.getConfig());
229.         Canvas canvas = new Canvas(createBitmap);
230.         Paint paint = new Paint();
231.         Matrix matrix = new Matrix();
232.         // 绘制图片
233.         canvas.drawBitmap(drawingCache, matrix, paint);
234.         return createBitmap;
235.     }
236.
```

```
237.     private void initView() {
238.         setContentView(R.layout.activity_antivirus);
239.         mListView = (ListView) findViewById(R.id.list_view);
240.         mTvProgress = (ArcProgress) findViewById(R.id.tv_progress);
241.         mTvAppPackageName = (TextView) findViewById(R.id.tv_app_package_name);
242.         mPackageManager = getPackageManager();
243.         // 扫描的结果
244.         ll_result_content = (LinearLayout) findViewById(R.id.ll_result_content);
245.         tv_is_virus = (TextView) findViewById(R.id.tv_is_virus);
246.         // 重新扫描
247.         bt_scan = (Button) findViewById(R.id.bt_scan);
248.         bt_scan.setOnClickListener(this);
249.         // 扫描的内容
250.         rl_content = (RelativeLayout) findViewById(R.id.rl_content);
251.         // 动画的布局
252.         ll_anim_content = (LinearLayout) findViewById(R.id.ll_anim_content);
253.         // 左边的图片
254.         iv_left = (ImageView) findViewById(R.id.iv_left);
255.         // 右边的图片
256.         iv_right = (ImageView) findViewById(R.id.iv_right);
257.     }
258.
259.     //用户点击重新扫描的逻辑
260.     @Override
261.     public void onClick(View v) {
262.         // 重新扫描不可以被点击
263.         bt_scan.setEnabled(false);
264.         // 关闭动画
265.         AnimatorSet set = new AnimatorSet();
266.         ObjectAnimator animatorTranslationLeft = ObjectAnimator.ofFloat(
267.             iv_left, "translationX", -iv_left.getWidth(), 0);
268.         ObjectAnimator animatorTranslationRight = ObjectAnimator.ofFloat(
269.             iv_right, "translationX", iv_right.getWidth(), 0);
270.         ObjectAnimator animatorAlphaLeft = ObjectAnimator.ofFloat(iv_left,
271.             "alpha", 0, 1.0f);
272.         ObjectAnimator animatorAlphaRight = ObjectAnimator.ofFloat(iv_right,
273.             "alpha", 0, 1.0f);
274.         set.playTogether(animatorTranslationLeft, animatorTranslationRight,
275.             animatorAlphaLeft, animatorAlphaRight);
276.         set.setDuration(2000);
277.         set.addListener(new AnimatorListener() {
278.             @Override
279.             public void onAnimationStart(Animator arg0) {
280.                 // TODO Auto-generated method stub
```

```
281.         }
282.         @Override
283.         public void onAnimationRepeat(Animator arg0) {
284.             // TODO Auto-generated method stub
285.         }
286.         @Override
287.         public void onAnimationEnd(Animator arg0) {
288.             inidata();
289.         }
290.         @Override
291.         public void onAnimationCancel(Animator arg0) {
292.             // TODO Auto-generated method stub
293.         }
294.     });
295.     set.start();
296. }
297. }
```

运行程序，效果图如图 9-13 所示。



图 9-13 手机病毒查杀结束的动画效果图

9.5 手机杀毒之模拟查杀病毒

9.5.1 复制病毒库

这里，我们开启 Tomcat 服务器，模拟从服务器端获取最新的一个病毒信息，并将该病毒信息添加到当前的病毒库中。而此时的这个最新的病毒信息，这里假设的是模拟器上当前已经安装的 API demos 这个应用程序的 Md5 值，这样就可以模拟手机扫描病毒的过程。

首先，将病毒库复制到当前应用中，注意，病毒库开始是放在 assets 目录下的，复制该病毒库的逻辑放在 SplashActivity.java 界面中的 initView()方法实现，具体代码如下所示。

```
1. private void initView() {
2.     .....
3.     // 拷贝数据库
4.     copyDB("address.db");
5.     // 拷贝病毒数据库
6.     copyDB("antivirus.db");
7.     // 复制常用号码
8.     copyDB("commonnum.db");
9.     // 更新病毒数据库
10.    updateAntivirus();
11. }
12. /**
13.  * 更新病毒
14.  */
15. private void updateAntivirus() {
16.     // TODO Auto-generated method stub
17.     HttpUtils httpUtils = new HttpUtils();
18.     httpUtils.send(HttpMethod.GET, "http://192.168.0.105:8080/virus.json",
19.         new RequestCallBack<String>() {
20.             @Override
21.             public void onFailure(HttpException arg0, String arg1) {
22.                 // TODO Auto-generated method stub
23.             }
24.             @Override
25.             public void onSuccess(ResponseInfo<String> arg0) {
26.                 System.out.println(arg0.result);
27.                 try {
28.                     // md5---5ccb10017b53f30008c3cbe9b79301e2 为 API demos 的 MD5 值
29.                     // {"md5": "5ccb10017b53f30008c3cbe9b79301e2", "desc": "蝗虫病毒非常赋害"}
30.                     JSONObject json = new JSONObject(arg0.result);
31.                     String md5 = json.getString("md5");
32.                     System.out.println("-----服务器获取的md5-----" + md5);
33.                     String desc = json.getString("desc");
34.                     System.out.println("-----服务器获取的desc-----" + desc);
35.                     AntivirusDao.addVirus(getApplicationContext(), md5, desc);
36.                 } catch (JSONException e) {
37.                     // TODO Auto-generated catch block
38.                     e.printStackTrace();
39.                 }
40.             }
41.         });
42. }
```

9.5.2 模拟扫描查杀病毒

当我们将病毒添加到病毒库中，这时我们需要在查杀逻辑中将病毒信息标注出来，并在扫描结果中展示，具体的代码如下所示。

我们需要在扫描结束之时，计算病毒的数量，并显亮显示病毒信息，其中 `AsyncTask` 中的具体代码如下所示。

```
1. private class ScanTask extends AsyncTask<Void, AntivirusInfo, Void> {
2.     private int progress = 0;
3.     private int max = 0;
4.     private boolean isFinish = false;
5.     //病毒的总数
6.     private int virusCount = 0;
7.     @Override
8.     protected void onPreExecute() {
9.         .....
10.    }
11.    public void stop() {
12.        isFinish = true;
13.    }
14.
15.    @Override
16.    protected Void doInBackground(Void... params) {
17.        List<PackageInfo> packages = mPackageManager
18.            .getInstalledPackages(0);
19.        max = packages.size();
20.        mDatas = new ArrayList<AntivirusInfo>();
21.        try {
22.            for (PackageInfo packageInfo : packages) {
23.                if (isFinish) {
24.                    break;
25.                }
26.                progress++;
27.                AntivirusInfo info = new AntivirusInfo();
28.                String appName = packageInfo.applicationInfo.loadLabel(
29.                    mPackageManager).toString();
30.                Drawable icon = packageInfo.applicationInfo
31.                    .loadIcon(mPackageManager);
32.                String appPackageName = packageInfo.applicationInfo.packageName;
33.                // 获取所有文件的路径
34.                String path = packageInfo.applicationInfo.sourceDir;
35.                FileInputStream is = new FileInputStream(path);
36.                String md5 = MD5Utils.encode(is);
```

```
37.         // 查询数据库是否有病毒
38.         boolean isVirus = AntivirusDao.isVirus(
39.             getApplicationContext(), md5);
40.         info.icon = icon;
41.         info.appName = appName;
42.         info.isVirus = isVirus;
43.         info.appPackageName = appPackageName;
44.         // 如果有病毒。放到上面
45.         if (info.isVirus) {
46.             mDatas.add(0, info);
47.             virusCount++;
48.         } else {
49.             mDatas.add(info);
50.         }
51.         publishProgress(info);
52.         SystemClock.sleep(100);
53.     }
54. } catch (FileNotFoundException e) {
55.     // TODO Auto-generated catch block
56.     e.printStackTrace();
57. }
58. return null;
59. }
60.
61. @Override
62. protected void onPostExecute(Void result) {
63.     .....
64.     //判断是否有病毒
65.     if(virusCount > 0){
66.         tv_is_virus.setText("手机很不安全");
67.     }else{
68.         tv_is_virus.setText("您的手机很安全");
69.     }
70.     // 首先获取到背景图片
71.     Bitmap drawingCache = rl_content.getDrawingCache();
72.     // 设置左边的图片
73.     iv_left.setImageBitmap(getBitmapLeft(drawingCache));
74.     // 设置右边的图片
75.     iv_right.setImageBitmap(getBitmapRight(drawingCache));
76.     // 打开背景图片的动画
77.     showOpenAnim();
78. }
79. }
```

适配器中 getView()方法中的部分代码如下所示。


```
1      @Override
2      public View getView(int position, View convertView, ViewGroup parent) {
3          .....
4          // 判断是否有病毒
5          if (antivirusInfo.isVirus) {
6              tv_cache_size.setTextColor(Color.RED);
7          } else {
8              tv_cache_size.setTextColor(Color.GREEN);
9          }
10         tv_cache_size.setText(antivirusInfo.isVirus ? "病毒" : "安全");
11         ib_list_button_clean.setVisibility(antivirusInfo.isVirus ? View.VISIBLE
12                                           : View.INVISIBLE);
13         return convertView;
14     }
```

运行程序，效果图如图 9-14 所示，可以看到有病毒的应用。

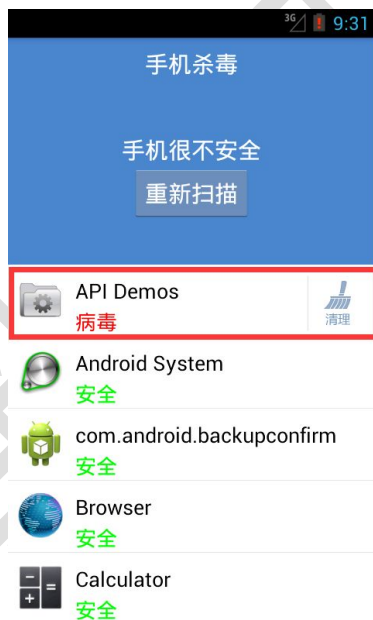


图 9-14 模拟查杀病毒查杀结束的动画效果图

9.6 本章小结

本章主要是针对手机杀毒模块进行讲解，首先讲解了病毒的特点与查杀方法，然后讲解病毒数据库的操作以及如何获取 MD5 码。最后讲解了手机杀毒模块的布局以及逻辑代码的开发。通过该模块的学习，可以让编程者掌握如何使用第三方数据库以及如何进行病毒查杀。