Android 反卸载

deepwaterooo

May 15, 2019

Contents

1 Android App 不被手动强制停止和卸载的实现 **1**
1.1 实现步骤 . . . 1
1.2 验证一下 . . . 3

2 无法卸载 app(DevicePolicManager) **5**
2.1 DevicePolicManager 可以做什么? . . . 5
2.2 原理解析 . . . 5
2.3 实现步骤 . . . 5

3 Android 设备管理器——DevicePolicyManager **6**

4 Android 防卸载实现代码 **6**
4.1 通过锁屏方式 . . . 6
4.2 通过阻塞函数返回方式 . . . 7
4.3 通过透明窗口劫持方式 . . . 7

5 安卓防卸载的实现 **8**

6 Android APP 中卸载其他 APP 的三种方法 **8**
6.1 直接使用 Intent 卸载 . . . 8
6.2 使用 PackageManager 静默卸载 . . . 8
6.3 通过 pm 命令方式实现静默卸载 . . . 9

7 其它思路 **10**

1 Android App 不被手动强制停止和卸载的实现

- https://blog.csdn.net/weiren1101/article/details/81668420
- 最近在做项目时，遇到一个这样的问题，如何让我们的 App 在“系统设置”里面，不被手动强制停止和卸载?
- 首先，我们分析一下:
  - 要实现不被手动停止，必须让系统设置的 App 详情页面的“强制停止”按钮被置灰才能做到。
  - 要实现不被手动卸载，必须让 App 获取系统的设备管理权限。
- 要实现以上两点，我们必须想办法让 App 在启动时自动获取系统的设备管理权限。接下来，我们通过查阅相关资料，需要按照以下步骤来做:

1.1 实现步骤

- 第一步：实现一个广播监听器，继承 DeviceAdminReceiver。如下：

```
public class MyAdminReceiver extends DeviceAdminReceiver {
    private static final String TAG = "MyAdminReceiver";
    @Override
    public void onReceive(Context context, Intent intent) {
        super.onReceive(context, intent);
    }
    @Override
    public void onEnabled(Context context, Intent intent) {
        super.onEnabled(context, intent);
    }
    @Override
    public void onDisabled(Context context, Intent intent) {
        // onDisabled
    }
}
```

- 第二步：在工程的 res 目录下，创建 xml 目录，在 xml 目录下创建 my_admin.xml 文件，具体内容如下：

```
<?xml version="1.0" encoding="utf-8"?>
<device-admin xmlns:android="http://schemas.android.com/apk/res/android">
  <uses-policies>
    <!--设置密码规则-->
    <limit-password />
    <!--监控登录次数-->
    <watch-login />
    <!--重置密码-->
    <reset-password />
    <!--强制锁屏-->
    <force-lock />
    <!--清空数据（恢复出厂设置）-->
    <wipe-data />
  </uses-policies>
</device-admin>
```

- 第三步：在 AndroidManifest.xml 清单文件，添加以下内容：

```
<uses-permission android:name="android.permission.BIND_DEVICE_ADMIN" />
<receiver
    android:name=".MyAdminReceiver"
    android:exported="false"
    android:permission="android.permission.BIND_DEVICE_ADMIN">
  <meta-data
      android:name="android.app.device_admin"
      android:resource="@xml/my_admin" />
  <intent-filter>
    <action android:name="android.app.action.DEVICE_ADMIN_ENABLED" />
  </intent-filter>
</receiver>
```

- 第四步：创建 DeviceAdminUtil 类，主要是做相关初始化，设备管理权限获取等。具体如下：

```java
public class DeviceAdminUtil {
    private static DeviceAdminUtil instance;
    private DevicePolicyManager devicePolicyManager;
    private ComponentName componentName;
    private Context mContext;

    public static synchronized DeviceAdminUtil getInstance() {
        if (instance == null) {
            instance = new DeviceAdminUtil();
        }
        return instance;
    }
    /**
     * 初始化“设备管理权限的获取”
     *
     * @param context
     */
    public void init(Context context) {
        mContext = context.getApplicationContext();
        // 获取系统管理权限
        devicePolicyManager = (DevicePolicyManager) mContext.getSystemService(Contex
        // 申请权限
        componentName = new ComponentName(mContext, MyAdminReceiver.class);
    }
    /**
     * 判断 App 是否已激活获取了设备管理权限,true: 已激活, false: 未激活
     *
     * @return
     */
    private boolean isAdminActive() {
        // 判断组件是否有系统管理员权限
        return devicePolicyManager.isAdminActive(componentName);
    }
    public void lockScreen() {
        if (isAdminActive() && devicePolicyManager != null) {
            // 立刻锁屏
            devicePolicyManager.lockNow();
        }
    }
}
```

- 经过上述 4 步操作，已基本实现我们当初的想法。现在我们来验证下。

1.2 验证一下

- 首先：那我们在 App 的 Application 的 onCreate 方法调用

```java
DeviceAdminUtil.getInstance().init(this);
```

- 然后，再某一个按钮点击事件里，调用

```java
DeviceAdminUtil.getInstance().lockScreen();
```

- 接下来，我们运行下 App，点击那个按钮并没有锁屏，在系统设置的 App 详情页面，我们也发现 App 还是可以手动强制停止，还是可以手动卸载。怎么回事呢?

- 我们调试发现，做了上述 4 步，devicePolicyManager.isAdminActive(componentName) 这个方法返回仍是 false，也就是说 App 并没有被激活。原因找到了，我们就要想办法激活 App。我们查阅 DevicePolicyManager 的源码，里面有很多方法，其中有一个方法如下：

```
public void setActiveAdmin(@NonNull ComponentName policyReceiver,
                           boolean refreshing) {
    setActiveAdmin(policyReceiver, refreshing, myUserId());
}
```

- 这个方法就是激活 App 获取设备管理权限的方法，但被 @hide 了，也就是说这个方法只有系统源码层才能直接调用，应用层不能直接调用。到这里，有经验的小伙伴应该会想到，用反射来调用 setActiveAdmin 方法，实现如下：

```
private void setDeviceAdminActive(boolean active) {
    try {
        if (devicePolicyManager != null && componentName != null) {
            Method setActiveAdmin = devicePolicyManager.getClass().
                getDeclaredMethod("setActiveAdmin",
                                  ComponentName.class, boolean.class);
            setActiveAdmin.setAccessible(true);
            setActiveAdmin.invoke(devicePolicyManager, componentName, active);
        }
    } catch (Exception e) {
        LogUtil.e(e);
    }
}
```

- 这里给出 DeviceAdminUtil 类的最终实现，如下：

```
public class DeviceAdminTool {
    private static DeviceAdminUtil instance;
    private DevicePolicyManager devicePolicyManager;
    private ComponentName componentName;
    private Context mContext;

    public static synchronized DeviceAdminUtil getInstance() {
        if (instance == null) {
            instance = new DeviceAdminUtil();
        }
        return instance;
    }
    /**
     * 初始化“设备管理权限的获取”
     *
     * @param context
     */
    public void init(Context context) {
        mContext = context.getApplicationContext();
        // 获取系统管理权限
        devicePolicyManager = (DevicePolicyManager) mContext.
            getSystemService(Context.DEVICE_POLICY_SERVICE);
        // 申请权限
        componentName = new ComponentName(mContext, MyAdminReceiver.class);
        setDeviceAdminActive(true);
    }
```

```
    /**
     * 判断 App 是否已激活获取了设备管理权限,true: 已激活, false: 未激活
     * app 预装，启动后要授权静默激活
     *
     * @return
     */
    private boolean isAdminActive() {
        // 判断组件是否有系统管理员权限
        return devicePolicyManager.isAdminActive(componentName);
    }
    public void lockScreen() {
        if (isAdminActive() && devicePolicyManager != null) {
            // 立刻锁屏
            devicePolicyManager.lockNow();
        }
    }
    private void setDeviceAdminActive(boolean active) {
        try {
            if (devicePolicyManager != null && componentName != null) {
                Method setActiveAdmin = devicePolicyManager.getClass().
                    getDeclaredMethod("setActiveAdmin",
                                      ComponentName.class, boolean.class);
                setActiveAdmin.setAccessible(true);
                setActiveAdmin.invoke(devicePolicyManager, componentName, active);
            }
        } catch (Exception e) {
            LogUtil.e(e);
        }
    }
}
```

- 以上，在 Android7.1，Android8.1 上测试有效。

2 无法卸载 app(DevicePolicManager)

- https://www.jianshu.com/p/8f9b44302139

2.1 DevicePolicManager 可以做什么?

1. 恢复出厂设置 2. 修改屏幕解锁密码 3. 修改屏幕密码规则长度和字符 4. 监视屏幕解锁次数 5. 锁屏幕 6. 设置锁屏密码有效期 7. 设置应用数据加密 8. 禁止相机服务，所有 app 将无法使用相机

- 首先我想，如果你是一个 Android 重度体验用户，在 Rom 支持一键锁屏之前，你也许装过一种叫快捷锁屏、一键锁屏之类的替代实体键锁屏的应用。其中导致的问题就是当我们不需要用它的时候却发现无法被卸载。

2.2 原理解析

- 从功能上来看，本身该项服务是用来控制设备管理，它是 Android 用来提供对系统进行管理的。所以一但获取到权限，不知道 Android 出于什么考虑, 系统是不允许将其卸载掉的。我们只是在这里钻了空子。

2.3 实现步骤

- 继承 DeviceAdminReceiver 类，里面的可以不要做任何逻辑处理。

```
public class MyDeviceAdminReceiver extends DeviceAdminReceiver {
}
```

- 注册一下，description 可以写一下你给用户看的描述。

```
<receiver
    android:name=".MyDeviceAdminReceiver"
    android:description="@string/description"
    android:label=" 防卸载"
    android:permission="android.permission.BIND_DEVICE_ADMIN" >
  <meta-data
      android:name="android.app.device_admin"
      android:resource="@xml/deviceadmin" />

  <intent-filter>
    <action android:name="android.app.action.DEVICE_ADMIN_ENABLED" />
  </intent-filter>
</receiver>
```

- 调用系统激活服务

```
// 激活设备超级管理员
public void activation() {
    Intent intent = new Intent(DevicePolicyManager.ACTION_ADD_DEVICE_ADMIN);
    // 初始化要激活的组件
    ComponentName mDeviceAdminSample = new ComponentName(MainActivity.this, MyDevice
    intent.putExtra(DevicePolicyManager.EXTRA_DEVICE_ADMIN, mDeviceAdminSample);
    intent.putExtra(DevicePolicyManager.EXTRA_ADD_EXPLANATION, " 激活可以防止随意卸载
    startActivity(intent);
}
```

- github 地址：https://github.com/BolexLiu/SuPerApp

3 Android 设备管理器——DevicePolicyManager

- https://www.oschina.net/question/54100_29057

4 Android 防卸载实现代码

- https://blog.csdn.net/Out_Put_Stream/article/details/50079801
- 实现防卸载首先需要激活设备管理器。激活设备管理器后应用将无法正常卸载，取消激活后可即可正常卸载。
- 所以我们要禁止用户取消激活。
- 激活设备管理器请参考：http://www.oschina.net/question/54100_29057
- 由于 app 注册了 android.app.action.DEVICE_ADMIN_ENABLED 的广播接受，所以当用户将要点击取消激活时系统会调用 app 中注册 android.app.action.DEVICE_ADMIN_ENABLED 实现类中的 onDisableRequested 方法。
- 我们可以在应用的 onDisableRequested 方法中添加以下代码阻止用户取消激活：

4.1 通过锁屏方式

```
public  CharSequence onDisableRequested(Context context, Intent intent) {
    // TODO Auto-generated method stub
    Intent intent1 = context.getPackageManager().getLaunchIntentForPackage("com.android.s
    intent1.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    context.startActivity(intent1);
    final DevicePolicyManager dpm = (DevicePolicyManager) context.getSystemService(Contex
    dpm.lockNow();
    new Thread(new Runnable() {
            @Override
            public void run() {
                int  i =  0 ;
                while (i < 70 ){
                    dpm.lockNow();
                    try {
                        Thread.sleep( 100 );
                        i++;
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
        }).start();
    return "This is a onDisableRequested response message";
}
```

4.2 通过阻塞函数返回方式

```
@Override
public CharSequence onDisableRequested(Context context, Intent intent) {
    // TODO Auto-generated method stub
    Intent intent1 = context.getPackageManager().getLaunchIntentForPackage("com.android.s
    intent1.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    context.startActivity(intent1);
    try {
        Thread.sleep( 7000 );
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    return "This is a onDisableRequested response message" ;
}
```

4.3 通过透明窗口劫持方式

```
@Override
public CharSequence onDisableRequested(Context context, Intent intent) {
    // TODO Auto-generated method stub
    Intent intent1 = context.getPackageManager().getLaunchIntentForPackage("com.android.s
    intent1.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    context.startActivity(intent1);

    WindowManager.LayoutParams wmParams;
    final WindowManager mWindowManager;
```

```
    wmParams = new WindowManager.LayoutParams();
    mWindowManager = (WindowManager)context.getSystemService(Context.WINDOW_SERVICE);
    wmParams.type = WindowManager.LayoutParams.TYPE_SYSTEM_ALERT;
    wmParams.format = PixelFormat.RGBX_8888;
    wmParams.flags = WindowManager.LayoutParams.FLAG_NOT_FOCUSABLE;
    wmParams.gravity = Gravity.LEFT | Gravity.TOP;
    wmParams.alpha = 0;
    wmParams.x = 0;
    wmParams.y = 0;
    wmParams.width = WindowManager.LayoutParams.MATCH_PARENT;
    wmParams.height = WindowManager.LayoutParams.MATCH_PARENT;
    final View contentView =  new  Button(context);
    mWindowManager.addView(contentView, wmParams);
    new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    Thread.sleep( 7000 );
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
                mWindowManager.removeView(contentView);
            }
        }).start();
    return "This is a onDisableRequested response message" ;
}
```

5 安卓防卸载的实现

- https://blog.csdn.net/u012833250/article/details/50448136
- 最近搞到个木马，安装之后确认了设备管理器权限竟然无法取消掉设备管理器的权限!! 无法取消设备管理器的权限也就意味着软件不能以正常的方式卸载，然后我用终端模拟器在 root 环境下执行 pm 命令，竟然提示失败!!! 最终还是直接使用 rm 命令删掉 data 空间中的 apk 才删掉的。
- 那究竟是什么原因呢，我反编译看了一下源码，原来在重写 DeviceAdminReceiver 的时候有个回调方法 onDisableRequested，当取消设备管理器的时候就会执行这个方法，而返回值就是取消设备管理器的提示信息

```
public CharSequence onDisableRequested(Context paramContext, Intent paramIntent) {
    a(paramContext, paramIntent);
    Log.i("Ray: ", "onDisableRequested");
    return " 谨慎操作: 取消激活可能会影响手机正常使用。";
}
```

- 所以只要在 return 之前做一些不友好的事情，比如无限循环卡死设置之类的，就能够防止用户卸载应用，当然卸载应用的方式很多，这里只是防止正常的应用卸载，如果直接用 root 权限删除 apk 包还是可以卸载的。等下我会将测试源码上传到 csdn，在第一个评论里面回复地址，感兴趣的可以下载看看

6 Android APP 中卸载其他 APP 的三种方法

- https://blog.csdn.net/ta_ab/article/details/77949348

6.1 直接使用 Intent 卸载

```
Uri uri = Uri.fromParts("package", "com.example.demo", null);
Intent intent = new Intent(Intent.ACTION_DELETE, uri);
startActivity(intent);
```

- 这是最简单的方式，调用卸载方法系统会弹出卸载 APP 对话框，点击确定就会立即卸载，不需要额外权限

6.2 使用 PackageManager 静默卸载

- 谷歌认为该方法是不安全的行为，因此该接口是 @hide 的，不是公开的接口，调用此接口需要有系统签名和相应的系统级权限
- 具体来说就是需要

```
<uses-permission android:name="android.permission.DELETE_PACKAGES"/>
```

权限，但 <uses-permission android:name=”android.permission.DELETE_PACKAGES”/> 是系统级权限，普通 APP 根本无法获取到，如果在 AndroidManifest.xml 强行加入该权限编译也不会通过

- 唯一的办法就是使用 APK 反编译工具在 Android Studio 之外修改权限，比如用 apktool 反编译工具先把 apk 文件解压出来，用编辑器在 AndroidManifest.xml 中加入上面的两个权限，然后在用工具 apktool 重新打包
- 获得 <uses-permission android:name=”android.permission.DELETE_PACKAGES”/> 权限后，定义 PackageDeleteObserver 实现类，实现 packageDeleted 方法

```
private class PackageDeleteObserver extends IPackageDeleteObserver.Stub {
    private int position;
    private int mFlag;
    public PackageDeleteObserver(int index, int flag) {
        position = index;
        mFlag = flag;// 0 卸载 1 个包，1 卸载 N 个包 N>1
    }
    @Override
    public void packageDeleted(String arg0, int arg1)
    throws RemoteException {
        // TODO Auto-generated method stub
        Message msg;
        msg = mHandle.obtainMessage();
        msg.what = FLAG_DELETE_VIRUS;
        msg.arg1 = position;
        msg.arg2 = mFlag;
        msg.sendToTarget();
    }
}
```

- 获取 PackageManager 对象，调用 deletePackage 方法

```
PackageManager pkgManager = mContext.getPackageManager();
PackageDeleteObserver observer = new PackageDeleteObserver(currVirus, 1);
pkgManager.deletePackage(pakName, observer, 0);
```

- 最后，还需要进行系统签名才能使用
- 对 apk 进行系统签名：

```
java -jar signapk.jar platform.x509.pem platform.pk8 test.apk test_signed.apk
```

- 将签名之后的文件 push 到手机中，需要 root 权限

6.3 通过 pm 命令方式实现静默卸载

- 该方法直接对 Android 系统执行卸载命令，需要 root 权限

```
//pm 命令可以通过 adb 在 shell 中执行，同样，我们可以通过代码来执行
public static String execCommand(String... command) {
    Process process = null;
    InputStream errIs = null;
    InputStream inIs = null;
    String result = "";
    try {
        process = new ProcessBuilder().command(command).start();
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        int read = -1;
        errIs = process.getErrorStream();
        while ((read = errIs.read()) != -1) {
            baos.write(read);
        }
        inIs = process.getInputStream();
        while ((read = inIs.read()) != -1) {
            baos.write(read);
        }
        result = new String(baos.toByteArray());
        if (inIs != null)
        inIs.close();
        if (errIs != null)
        errIs.close();
        process.destroy();
    } catch (IOException e) {
        result = e.getMessage();
    }
    return result;
}
```

- 执行卸载命令

```
execCommand("pm","uninstall", "packageName");
```

- 编译生成 apk 时，要在 manifest 文件下添加 Android:sharedUserId=”android.uid.system”

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="com.xieyuan.mhfilemanager"
          android:versionCode="1"
          android:versionName="1.0"
          android:installLocation="internalOnly"
          android:sharedUserId="android.uid.system" >
```

7 其它思路

- Android 系统应用隐藏和应用禁止卸载
  - https://blog.csdn.net/yao891203/article/details/84809257
- Android 系统预制 APK(不可卸载)
  - https://blog.csdn.net/u010218230/article/details/79153986
- 如何防止 android app 被 kill
  - https://blog.csdn.net/csh86277516/article/details/79929595
- Android 如何监控本应用被卸载
  - https://blog.csdn.net/huiguixian/article/details/43228091
- android 如何监听自身应用被卸载
  - https://blog.csdn.net/xinzhou201/article/details/78632154
- Android 如何实现应用卸载反馈，卸载监控
  - https://blog.csdn.net/wangbaochu/article/details/50697125
- JNI 基础（九）android 如何监控到应用被卸载?
  - https://www.csdndoc.com/article/9352234
- Android 监听自身卸载，弹出用户反馈调查
  - https://cloud.tencent.com/developer/article/1033962
- 防止 APK 被反编译
  - https://blog.csdn.net/qq_35414752/article/details/79823069
- 浅谈安卓 apk 加固原理和实现
  - https://my.oschina.net/u/3920392/blog/2967330