AutoStart方案分析

一. 组成

1. 一个在`SystemServer`中启动的服务-`AppPcService`
2. 一个将白名单写入`ContentProvider`的应用-`AutoBootProvider`
3. 一个让用户从设置中修改数据库的页面-`BackgroundAppFragment`

二. 实现

简单来说就是在收听广播调起应用的时候会比对白名单和调用应用的包名,如果在白名单才会允许被调用. 其中做判断的是`AppPcService`

2.1 AppPcService

- 在`SystemServer`的`startOtherService()`中创建了`AppPcService`,并加入到`ServiceManager`中.

```
try {
    Slog.i(TAG, "AppPcService");
    appPcService = new AppPcService(context);
    ServiceManager.addService("apppc", appPcService.onBind(null));
} catch (Throwable e) {
    Slog.e(TAG, "FAIL starting AppPcService", e);
}
```

- 在`ActivityManagerService`中的`systemReady()`中调用`AppPcService`的`systemReady()`.

```
try {
    if (mappPcService != null)
        mappPcService.systemReady();
} catch (Throwable e) {
    reportWtf("Notifying appPcService excetpion:", e);
}
```

2.2 AppPcService的构造函数和systemReady函数

- 构造函数:

```
public AppPcService(Context context) {
    mCr = context.getContentResolver();
    mCo = new ContentObserver(null) {
        @Override
        public void onChange(boolean self) {
            updateList();
        }
    };
    mCr.registerContentObserver(CONTENT_URI, true, mCo);
}
```

注册了一个观察者,观察`AutoBootProvider`的变化,实时更新白名单

- `systemReady`函数:

```
public void systemReady() {
    updateList();
}
```

其中直接调用了`updateList`函数

- `updateList`函数:

```
private void updateList() {
    if(mCr == null) {
        return;
    }

    Cursor listCursor;
    try{
        listCursor = mCr.query(CONTENT_URI, new String[] { "_id",
                "packagename", "grant" }, "grant='1'", null, null);
                whiteAppPackageList.clear();
                while (listCursor.moveToNext()) {
                whiteAppPackageList.add(listCursor.getString(1));
        }
    }catch (Exception e) {
        Slog.e(TAG, "FAIL starting listCursor", e);
    }

 }
```

在这个函数里直接去查询了`AutoBootProvider`数据库,并更新了`whiteAppPackageList`这个数组.

- `Binder`调用:

```
private Binder mBinder = new IAppPcService.Stub() {
    @Override
    public boolean checkAutoStart(String packageName, String callerApp,
        String action) {
            Log.d(TAG, "checkAutoStart() run packageName = " + packageName+ " callerApp = " + callerApp + " action = " + actio
            Log.d(TAG,"whiteAppPackageList.size:==" + whiteAppPackageList.size());
            Log.d(TAG, "whiteAppPackageList.contains (packageName) is:=="+ whiteAppPackageList.contains(packageName));
         if (whiteAppPackageList.contains(packageName)
            || packageName.contains("com.android")
            || packageName.contains("com.lenovo"))
            return true;
         else
            return false;
       }
};
```

判断传入的包名是否是`系统应用`或`联想应用`或在`whiteAppPackageList`中,都不满足返回`false`

2.3 AutoBootProvider应用

1. 创建的第一步就是通过IO流将`whiteAppPackageList.txt`文件写入到数据库.
2. 提供一系列数据库操作符来对`autoboot.db`进行操作

2.4 用户接口BackgroundAppFragment

1. 通过`PackageManager`来查询所有的安装的程序并和`AutoBootProvider`进行比对.
2. 通过开关来更新`AutoBootProvider`里面的数据

BroadcastQueue实现

在`processBroadcast(boolean msg)`中

```
boolean isGranted = true;
android.util.Log.i(TAG,"packageName uid:==="+info.activityInfo.applicationInfo.uid);
if (info.activityInfo.applicationInfo.uid>10000) {

    if (r.intent.getAction() != null && r.intent.getAction().startsWith("android.appwidget.action")) {
        android.util.Log.i(TAG," appwidget broadcast continue send action :  "+r.intent.getAction());
    }else if (r.callerPackage != null && r.callerPackage.equals(info.activityInfo.packageName)) {
        android.util.Log.i(TAG," broadcast to self continue send action :  "+r.callerPackage);
    }else{
```

```
       try {
         IAppPcService  mAppPcService = IAppPcService.Stub.asInterface(ServiceManager.getService("apppc"));
          if(mAppPcService != null) {
              android.util.Log.i(TAG,"packageName is:==="+info.activityInfo.packageName);
              isGranted = mAppPcService.checkAutoStart(info.activityInfo.packageName, r.callerPackage, r.intent.getAction());
              if(!isGranted){
                  for (int i = mService.mLruProcesses.size() - 1; i >= 0; i--) {
                    ProcessRecord recapp = mService.mLruProcesses.get(i);
                    if (recapp.info!=null && recapp.info.packageName != null && recapp.info.packageName.equals(info.activityInfo.a
                        isGranted = true;
                        break;
                    }
                  }
              }
          }else{
              android.util.Log.i(TAG, "AppPcService is null!");
          }
       } catch(Exception e) {
            android.util.Log.e("AppPcService", "init AppPcService");
       }
    }
}
if (!isGranted) {
    Slog.w(TAG, "Unable to launch app "
            + info.activityInfo.applicationInfo.packageName + "/"
            + info.activityInfo.applicationInfo.uid + " for broadcast "
            + r.intent + ": process is not granted");
    logBroadcastReceiverDiscardLocked(r);
    finishReceiverLocked(r, r.resultCode, r.resultData,
            r.resultExtras, r.resultAbort, false);
    scheduleBroadcastsLocked();
    r.state = BroadcastRecord.IDLE;
    return;
}
```

其中有几种情况不用进行审核:

- 通过桌面和组件进行开启应用
- 应用间内部调用
- 跑CTS的时候

除以上情况外会审核这个应用是否可以被广播调起,审核过程如下

1. 通过`Binder`调用来判断是否是属于白名单被允许的应用
2. 如果不再白名单内,再判断是否是在`RecentApp`且包含有UI界面的话,也可以允许调用
3. 如果都不符合就重置状态.