当 App 有了系统权限，真的可以为所欲为？

测试 Gracker Android Performance 2023-03-09 08:00 发表于四川

收录于合集
#系统开发
4个 >

微信扫一扫
关注该公众号

看到群里发了两篇文章，出于好奇，想看看这些个 App 在利用系统漏洞获取系统权限之后，都干了什么事，于是就有了这篇文章。由于准备仓促，有些 Code 没有仔细看，感兴趣的同学可以自己去研究研究，多多讨论

1. 深蓝洞察：2022 年度最 “不可思” 漏洞
2. XXX apk 内嵌提权代码，及动态下发 dex 分析：https://github.com/davinci1010/pinduoduo_backdoor

关于这个 App 是如何获取这个系统权限的，「深蓝洞察」2022 年度最 “不可思” 漏洞，这篇文章讲的很清楚，就不再赘述了：

> Android Framework 中一个核心的对象传递机制是 Parcel，希望能通过 Parcel 传递的对象需要定义 readFromParcel 和 writeToParcel 接口函数，并实现 Parcelable 接口，理论上来讲，匹配序列化和反序列化函数应当是自反等效的，但系统 ROM 的开发者在编程过程中可能会出现不匹配的情况，例如写入的时候使用了 writeLong，读取的时候却使用了 readInt，这类问题在运行过程中一般不会引起注意，也不会导致崩溃或错误，但在双击老精心布局下，即可最终利用 Settings 和 system_server 进程，将这个微小的错误转化为 StartAnyWhere 提权。Android 近年来重计已修复上百个这类漏洞，并在 Android 13 中对 Parcel 机制做了改装，但提权绕了大部分此类攻击面。
>
> 但对于鸿蒙和绝大部分未升级到 Android 13 的设备和用户来说，他们仍处于危险之中。

我们这篇主要来看看 XXX apk 内嵌提权代码，及动态下发 dex 分析 这个库里面提供的 Dex 。看看 App 到底想知道用户的什么信息？

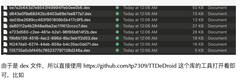
总的来说，App 获取系统权限之后，主要做了下面几件事（正常 App 无法或者很难做到的事情），各种不把用户当人了

1. **自启动、关联启动相关的修改。偷偷打开或者默认打开**
2. **开启通知权限**
3. **监听通知内容**
4. **获取用户的使用手机的信息，包括安装的 App，使用时长、用户 ID、用户名等**
5. **修改系统设置**
6. **整一些系统权限的工具方便自己使用**

另外也可以看到，App 对于各个手机厂商的研究还是比较深入的，针对华为、Oppo、Vivo、Xiaomi 等厂商都有专门的处理，这个也是值得手机厂商去反向研究的

0. Dex 文件信息

XXX apk 内嵌提权代码，及动态下发 dex 分析 这个库提供的的 Dex 文件总共有 37 个，不多，也不大，慢慢看



由于 dex 文件，所以直接使用 https://github.com/ty7309/TTDeDroid 这个库的工具打开看即可，比如

```
showjar 95cd95ab4d694ad8bf49f07e3599b3.dex
```

默认是用 jadx 打开，我们重点看内容

a$b(String, String)
a$c(Context, Bundle)
a$d(Context, Bundle)
a$e(ReactApplicationContext, String)
Resources
a$a(Context, int, int, int

3. 类名：
com.google.android.sd.biz_dynamic_dex.vivo_open_push.VivoOpenPushExecutor

核心和上面那个是一样的，只不过这个是专门针对 vivo 手机的

```
if (cursorToString != 0L
    && !isClosed()
    && !isNull(cursorToString
    && !getString(cursorToString).
    return false;
}
Method method = null;
int[] iArr2 = new int[length];
int i = 0;
while (true) {
    if (i >= length) {
        break;
    }
    Method method2 = getMethod(cls
    if (TextUtils.equals("com.vivo.
        method = method2;
        break;
    }
    iArr2[i] = 0;
    i = i + 1;
}
if (method == null) {
    return false;
}
method.invoke(obj, new Object[]{
return true;
}
```

1.4 打开 Oppo 手机的通知权限

1. 文件： 67c9e6860d45158e94002e8e78119 2.dex
2. 类名：
com.google.android.sd.biz_dynamic_dex.oppo_notification_ut.OppoNotificationUTExecutor

没有反编译出来，看大概的逻辑应该是打开 App 在 oppo 手机上的通知权限

```
    java.lang.String r3 = "com.coloros.notificationmanager"
    android.content.pm.PackageInfo r8 = r9.getPackageInfo(r1, r7)     // Catch:{ all -> 0x0402 }
    if (r8 == null) goto L_0x0402
    java.lang.String r9 = "content://com.coloros.notificationmanager/shared_preference"
    r15.f3a = r9     // Catch:{ all -> 0x0402 }
L_0x0272:
    android.content.pm.PackageManager r4 = r9.getPackageManager();     // Catch:{ all -> 0x0402 }
    java.lang.String r2 = "com.oplus.notificationmanager"
    android.content.pm.PackageInfo r5 = r4.getPackageInfo(r2, r7)     // Catch:{ all -> 0x0402 }
    if (r5 == 0) goto L_0x0402
    java.lang.String r7 = "content://com.oplus.notificationmanager/shared_preference"
    r15.f4a = r7     // Catch:{ all -> 0x0402 }
L_0x0282:
    java.lang.String r7 = r5.f3a     // Catch:{ all -> 0x0804 }
    if (r7 == null || r7.length() <= 0) goto L_0x0804
    if (r7 == 0) goto L_0x0804
    java.lang.String r7 = "init ori fail"
    r15.f4a = r7     // Catch:{ all -> 0x0804 }
    return;
L_0x0804:
```

1.5 Notification 监听

1. 文件： ab8ed4c3482c42a1b8baef55fee79deb.dex
2. 类名：
com.google.android.sd.biz_dynamic_dex.ud_notification_listener.UdNotificationListenerExecutor

这个就有点厉害了，在监听 App 的 Notification 的发送，然后进行统计

```
    JsonArray jsonArray = new JsonArray();
    String optString = jSONObject.optString("work_type");
    if (TextUtils.equals(optString, "listen")) {
        new CBH000().m11lo(context, jSONObject);
    }
    if (TextUtils.equals(optString, "collect")) {
        CBH0la ahur = new CBH0la();
        aHur.f4a = jSONObject.optString("mapId");
        context.registerReceiver(aHur.f3a, new IntentFilter("send_data"));
        Intent intent = new Intent("collect_data");
        intent.setPackage(context.getPackageName());
        context.sendBroadcast(intent);
        CBH0la.m8a(new Runnable(context) {
            /* renamed from: a */
            final /* synthetic */ Context f8au;

            {
                this.f8a = r2;
            }

            public final void run() {
                SystemClock.sleep(1000);
                CBH0la.m1a(CBH0la.this, this.f8a, (String) null);
            }
        });
    }
}
```

监听 App 的 Notification

```
/* JADX WARN: Type inference failed for: r2, r3, r4, r5, r6, r7, r8, r9, r10, r11, r12, r13, r14, r15, r16, r17, r18, r19, r20, r21, r22, r23, r24, r25, r26, r27, r28, r29, r30, r31, r32, r33, r34, r35, r36, r37, r38, r39, r40, r41,
```

化，Configuration 变化，亮灭屏，开关机等，感兴趣的可以看一下：

```
frameworks/base/core/java/android/app/usage/UsageEvents.java
    private static String eventToString(int eventType) {
        switch (eventType) {
            case Event.NONE:
                return "NONE";
            case Event.ACTIVITY_PAUSED:
                return "ACTIVITY_PAUSED";
            case Event.ACTIVITY_RESUMED:
                return "ACTIVITY_RESUMED";
            case Event.FOREGROUND_SERVICE_START:
                return "FOREGROUND_SERVICE_START";
            case Event.FOREGROUND_SERVICE_STOP:
                return "FOREGROUND_SERVICE_STOP";
            case Event.ACTIVITY_STOPPED:
                return "ACTIVITY_STOPPED";
            case Event.END_OF_DAY:
                return "END_OF_DAY";
            case Event.ROLLOVER_FOREGROUND_SERVICE:
                return "ROLLOVER_FOREGROUND_SERVICE";
            case Event.CONTINUE_PREVIOUS_DAY:
                return "CONTINUE_PREVIOUS_DAY";
            case Event.CONTINUING_FOREGROUND_SERVICE:
                return "CONTINUING_FOREGROUND_SERVICE";
            case Event.CONFIGURATION_CHANGE:
                return "CONFIGURATION_CHANGE";
            case Event.SYSTEM_INTERACTION:
                return "SYSTEM_INTERACTION";
            case Event.USER_INTERACTION:
                return "USER_INTERACTION";
            case Event.SHORTCUT_INVOCATION:
                return "SHORTCUT_INVOCATION";
            case Event.CHOOSER_ACTION:
                return "CHOOSER_ACTION";
            case Event.NOTIFICATION_SEEN:
                return "NOTIFICATION_SEEN";
            case Event.STANDBY_BUCKET_CHANGED:
                return "STANDBY_BUCKET_CHANGED";
            case Event.NOTIFICATION_INTERRUPTION:
                return "NOTIFICATION_INTERRUPTION";
            case Event.SLICE_PINNED:
                return "SLICE_PINNED";
            case Event.SLICE_PINNED_PRIV:
                return "SLICE_PINNED_PRIV";
            case Event.SCREEN_INTERACTIVE:
                return "SCREEN_INTERACTIVE";
            case Event.SCREEN_NON_INTERACTIVE:
                return "SCREEN_NON_INTERACTIVE";
            case Event.KEYGUARD_SHOWN:
                return "KEYGUARD_SHOWN";
            case Event.KEYGUARD_HIDDEN:
                return "KEYGUARD_HIDDEN";
            case Event.DEVICE_SHUTDOWN:
                return "DEVICE_SHUTDOWN";
            case Event.DEVICE_STARTUP:
                return "DEVICE_STARTUP";
            case Event.USER_UNLOCKED:
                return "USER_UNLOCKED";
            case Event.USER_STOPPED:
                return "USER_STOPPED";
            case Event.LOCUS_ID_SET:
                return "LOCUS_ID_SET";
            case Event.APP_COMPONENT_USED:
                return "APP_COMPONENT_USED";
            default:
                return "UNKNOWN_TYPE_" + eventType;
        }
    }
```

4.2 获取用户使用数据

1. 文件：b50477f70bd14479a50e6a34e18b2a0.dex
2. 类名：com.google.android.sd.biz_dynamic_dex.usage_event.UsageEventExecutor

上面部个是 UsageEventAllExecutor，这个是 UsageEventExecutor，主要拿用户使用 App 相关的数据，比如什么时候打开某个 App，什么时候关闭某个 App，6 得很，真毒瘤

4.3 获取用户使用数据

1. 文件：1a6dd982e029c22b46469a0d5238fac.dex
2. 类名：com.google.android.sd.biz_dynamic_dex.app_usage_observer.AppUsageObserver

看样子是注册了 App Usage 的权限，具体 Code 没有出来，不好分析

5. Widget 和 icon 相关

5.1. Vivo 手机添加 Widget

1. 文件：f46eb139-4516-4ac2-896d-8bc3eb1f2d03.dex
2. 类名：com.google.android.sd.biz_dynamic_dex.vivo_widget.VivoAddWidgetExecutor

这个比较好理解，在 Vivo 手机上加个 Widget

5.2 获取 icon 相关的信息

1. 文件：da60112a4b2848adba2ac11412ccccc7.dex
2. 类名：com.google.android.sd.biz_dynamic_dex.get_icon_info.GetIconInfoExecutor

这个好理解，获取 icon 相关的信息，比如在 Launcher 的哪一行，哪一列，是否在文件夹里面，问题是获取这玩意干嘛？？？这

```
}
```

5.3 Oppo 手机添加 Widget

1. 文件：75dcc8ea-d0f9-4222-b8dd-2a83444f9cd6.dex
2. 类名：com.google.android.sd.biz_dynamic_dex.oppoaddwidget.OppoAddWidgetExecutor

5.4 Xiaomi 手机更新图标?

1. 文件：5d372522-b6a4-4c1b-a0b4-8114d142e6c0.dex
2. 类名：com.google.android.sd.biz_dynamic_dex.xm_akasha.XmAkashaExecutor

小米手机上的桌面 icon 、shorcut 相关的操作，小米的同学来认领

6. 自启动、关联启动、保活相关

6.1 打开 Oppo 手机自启动

1. 文件：e723d56d-c2ee-461e-b2a1-96f85b614f2b.dex
2. 类名：com.google.android.sd.biz_dynamic_dex.oppo_boot_perm.OppoBootPermExecutor

看下面这一堆就知道是和自启动相关的，看来自启动权限是每个 App 都垂涎的东西啊

6.2 打开 Vivo 关联启动权限

1. 文件：8b56d820-cac2-4ca0-8a3a-1083c5cca7ae.dex
2. 类名：com.google.android.sd.biz_dynamic_dex.vivo_association_start.VivoAssociationStartExecutor

看名字就是和关联启动相关的权限，vivo 的同学来领了

```
private static boolean getCallerIsNotVivoApk(
    ...
    return sb.toString().startsWith("com.vivo.sdkplugin");
```

直接写了个节点进去

6.3 关闭华为耗电精灵

1. 文件：7c6e6702-e463-4315-8631-eee246aeba95.dex
2. 类名：com.google.android.sd.biz_dynamic_dex.hw_hide_power_window.HidePowerWindowExecutor

看名字和实现，应该是和华为的耗电精灵有关系，华为的同学可以来看看

```
    public static final String TABLE_NAME = "response";
    private static final HiveLibraries isPresent = new HiveLibraries(false);
```

6.4 Vivo 机型保活相关

1. 文件：7877ec6850344e7aad56d5776ab0238.dex
2. 类名：com.google.android.sd.biz_dynamic_dex.vivo_get_loc.VivoGetLocExecutor

猜测和保活相关，Vivo 的同学可以来认领一下

```
[ Process com.google.android.sd.bizdynamic.dex.vivo.rollback, 17, 1, 1, 1
```

6.10 LogcatExecutor

1. 文件：218a37ea-710d-49cb-bd72-2a47a1115c69.dex
2. 类名：com.google.android.sd.biz_dynamic_dex.logcat.LogcatExecutor

配置 Log 的参数

```
public abstract class LogcatExecutor {
    public static boolean[] $jacocoInit() {
        boolean[] $jacocoData = null;
        if (!Boolean.getBoolean("jacoco.agent.disabled.")) {
            $jacocoData = Object.getClass("com.google.android.sd.biz_dynamic_dex.logcat.LogcatExecutor").$jacocoInit();
        }
        return $jacocoData;
    }
}
```

6.11 VivoBrowserSettingsExecutor

1. 文件：136d4651-d847-41b4-bb80-2ecfdb1bc775.dex
2. 类名：
com.google.android.sd.biz_dynamic_dex.vivo_browser_settings.VivoBrowserSettingsExecutor

Vivo 浏览器相关的设置，不太懂要干嘛

```
    if (!TextUtils.isEmpty(str)) {
        JSONObject jSONObject = new JSONObject();
        int[] $jacocoInit = $jacocoInit();
        String aqString = jSONObject.optString("values", "");
        boolean[] $jacocoInit2 = VivoBrowserSettingsExecutor.getCorrectResults();
        if (TextUtils.equals(aqString, "")) {
            if (TextUtils.equals(str, "com.vivo.phone.upgrade:remoting")) {
                $jacocoInit[2] = true;
                jSONObject.put("values", str);
                if (TextUtils.equals(aqString, "browsers_url_authority_default")) {
                    $jacocoInit[3] = true;
                    updateValue(context, "com.android.phone.upgrade:remoting");
                } else {
                    $jacocoInit[4] = true;
                    fallcontext, "host_default_suff", aqString);
                }
            } else if (TextUtils.equals(str, "browsfff")) {
                $jacocoInit[5] = true;
                if (TextUtils.equals(aqString, "browsers_url_authority_default")) {
                    $jacocoInit[6] = true;
                    updateValue(context, "com.android.phone.upgrade:remoting");
                } else {
                    $jacocoInit[7] = true;
                    updateValue(context, "browsers_url_authority_default", aqString);
                }
                $jacocoInit[8] = true;
                fallcontext, "checkid_head_suff", aqString);
            }
        }
    }
}
```


Android Performance
分享 Android 系统和 App 的性能优化知识、开发技巧、Android 开发工具、性能优化工...
响篇原创内容

公众号

收录于合集 #系统开发 4

《上一篇 · Android 画面显示流程分析 (2)

喜欢此内容的人还喜欢

ChatGPT正式开放API，企微接入平台上线
品茶生活吧

Redis和数据库之间的一致性如何保证？|金三银四系列
后端开发技术

jspxcms代码审计学习笔记
XINYU2428