360技术嘉年华

安全能力SDK交付质量保障与实践

手机卫士-刘春勇

360

目录

- SDK交付背景与输出能力介绍
- SDK逻辑架构及测试场景分析
- 测试重点与方案实施
- 持续集成测试方案探索

一、SDK交付背景与输出能力介绍

背景

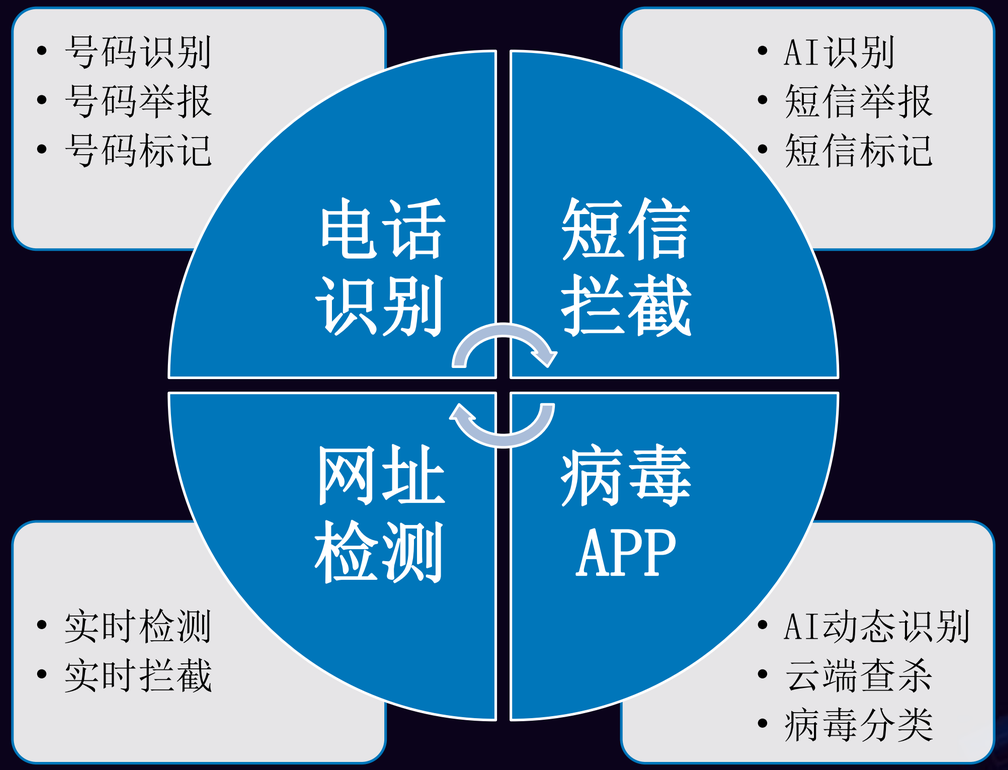
随着互联网的快速发展，**电信诈骗**、**网络钓鱼**、**隐私泄露**、**病毒APP**等各种网络安全隐患无刻不在，直接导致个人的**经济损失**，小则关己，大则关国，**安全能力SDK**，提供了四大安全防护能力，作为保护移动互联网安全的重要方式之一，做到防患于未然。

交付方式与输出能力

360技术嘉年华

To-B方式/To-C

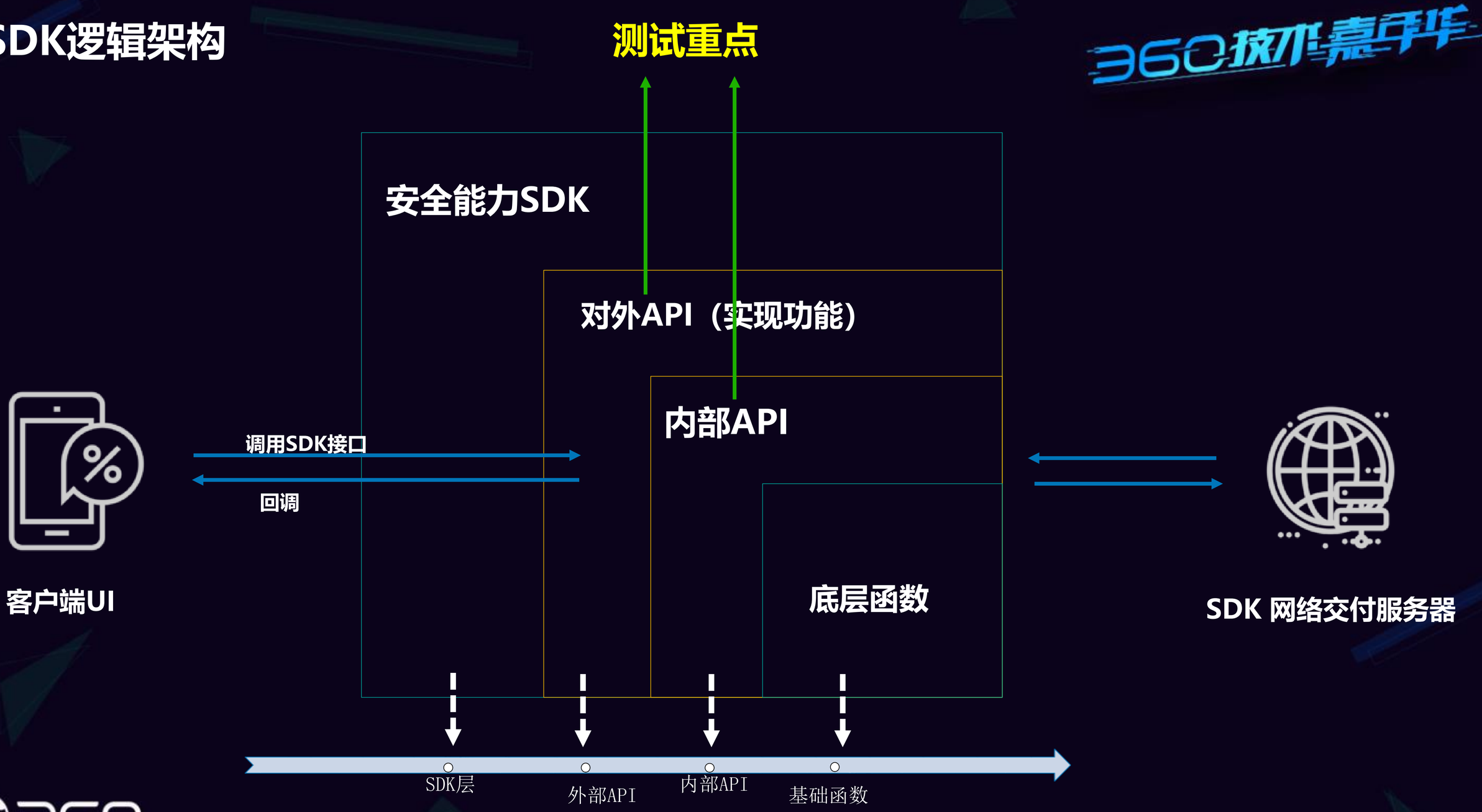
- huawei
- vivo
- oppo
- meizu
- 360手机卫士
- ...



360

二、SDK逻辑架构及测试场景分析

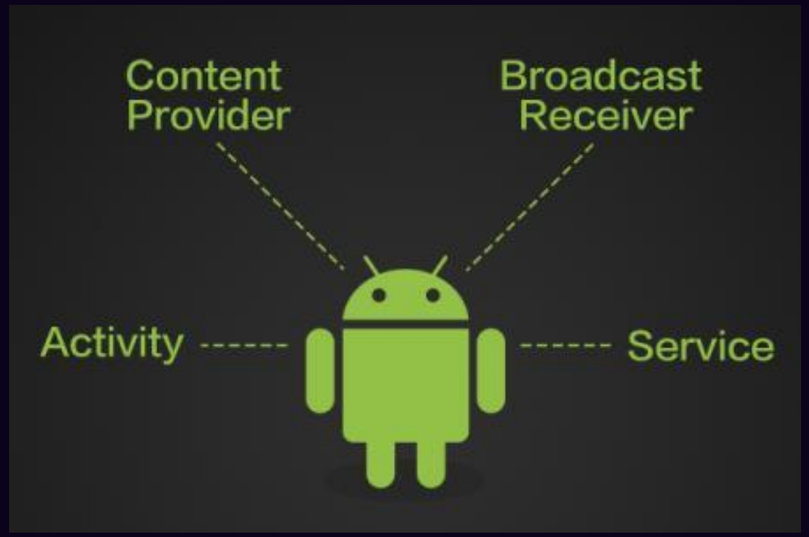
SDK逻辑架构


测试重点
安全能力SDK
对外API（实现功能）
内部API
底层函数
调用SDK接口
回调
客户端UI
SDK 网络交付服务器
SDK层
外部API
内部API
基础函数

测试场景分析

覆盖安卓四大组件调用方式：

- Activity
- Service
- Broadcast Receiver
- Content Provider



三、SDK测试重点与方案实施

SDK测试重点

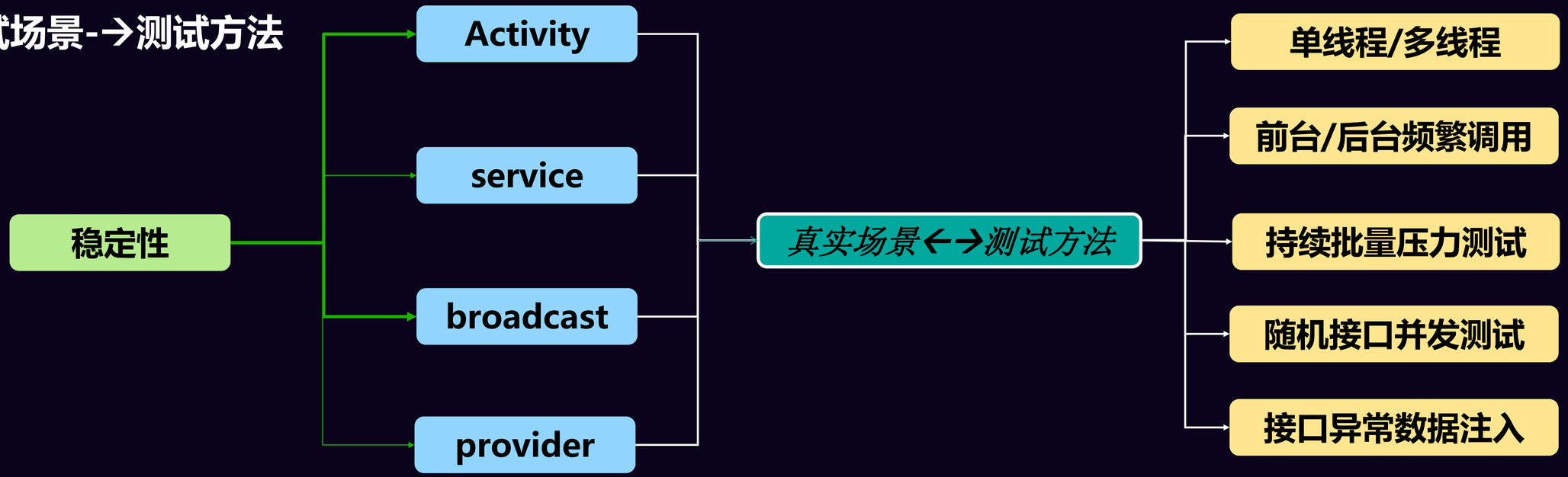
稳定性

兼容性

性能

安全合规

- 测试场景-→测试方法



- Crash监控

全局捕获crash方法:

```
public class Crasher implements UncaughtExceptionHandler {
    @Override
    public void uncaughtException(Thread thread, Throwable ex) {
        // TODO Auto-generated method stub
        Log.e( tag: "Crasher",  msg: "error thread id is:"+thread.getId());
        Log.e( tag: "Crasher",  msg: "Exception:"+"Name="+ex.getClass().getName()+",Message="+ex.getMessage());
    }
}
```

兼容性测试也是非常重要的，但是并不同于app测试，安全能力SDK并不需要兼容分辨率、UI等。

system app

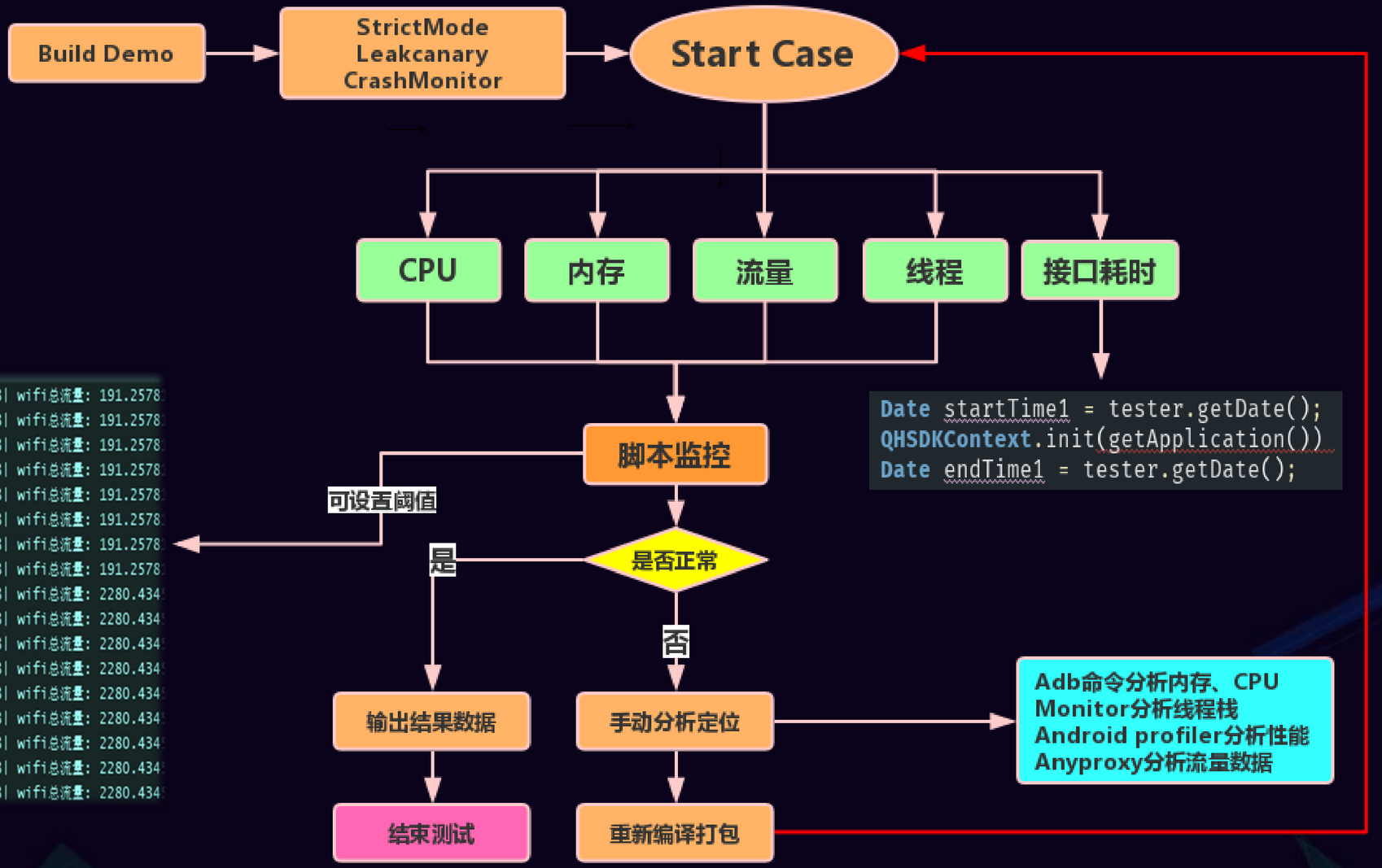
安卓版本

OS版本

targetSdkVersion

弱网测试

性能测试在sdk中是非常重要的，尤其是to-b项目，厂商都有着严格的标准，SDK性能测试重点包含**内存、cpu、流量，接口效率**。

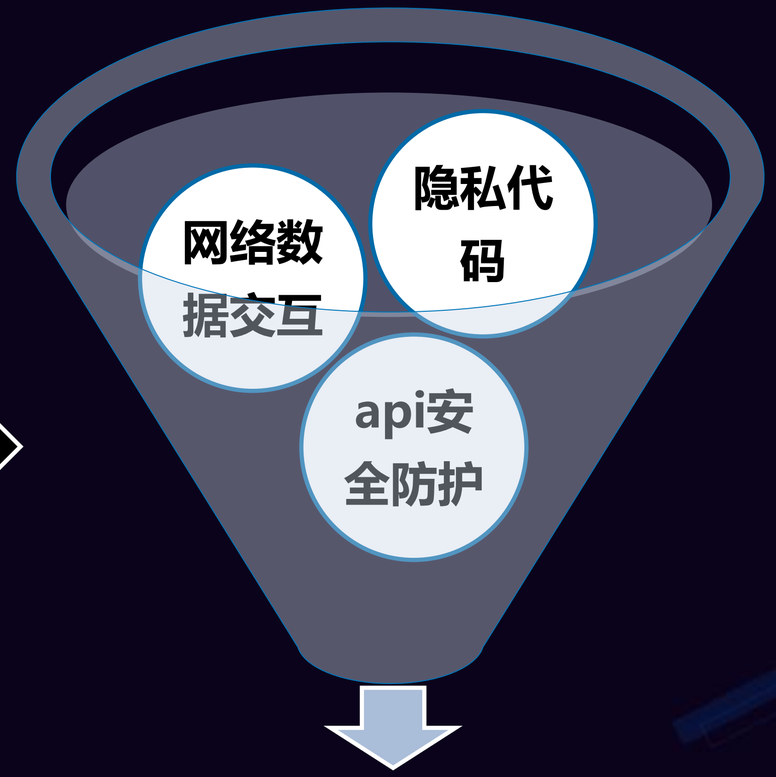



```
10132KB Java占用内存: 9500KB| CPU占用率: 0.00%| 数据总流量: 0.0KB| wifi总流量: 191.2578
10072KB Java占用内存: 9536KB| CPU占用率: 0.00%| 数据总流量: 0.0KB| wifi总流量: 191.2578
10092KB Java占用内存: 9536KB| CPU占用率: 0.00%| 数据总流量: 0.0KB| wifi总流量: 191.2578
10088KB Java占用内存: 9532KB| CPU占用率: 0.00%| 数据总流量: 0.0KB| wifi总流量: 191.2578
10088KB Java占用内存: 9536KB| CPU占用率: 0.00%| 数据总流量: 0.0KB| wifi总流量: 191.2578
10088KB Java占用内存: 9532KB| CPU占用率: 0.00%| 数据总流量: 0.0KB| wifi总流量: 191.2578
10092KB Java占用内存: 9536KB| CPU占用率: 3.33%| 数据总流量: 0.0KB| wifi总流量: 191.2578
10704KB Java占用内存: 9576KB| CPU占用率: 2.67%| 数据总流量: 0.0KB| wifi总流量: 191.2578
11380KB Java占用内存: 9784KB| CPU占用率: 0.00%| 数据总流量: 0.0KB| wifi总流量: 2280.434
11392KB Java占用内存: 9784KB| CPU占用率: 0.00%| 数据总流量: 0.0KB| wifi总流量: 2280.434
11392KB Java占用内存: 9784KB| CPU占用率: 0.00%| 数据总流量: 0.0KB| wifi总流量: 2280.434
11396KB Java占用内存: 9784KB| CPU占用率: 0.00%| 数据总流量: 0.0KB| wifi总流量: 2280.434
11248KB Java占用内存: 9784KB| CPU占用率: 0.00%| 数据总流量: 0.0KB| wifi总流量: 2280.434
11244KB Java占用内存: 9780KB| CPU占用率: 0.00%| 数据总流量: 0.0KB| wifi总流量: 2280.434
11244KB Java占用内存: 9780KB| CPU占用率: 0.00%| 数据总流量: 0.0KB| wifi总流量: 2280.434
11244KB Java占用内存: 9784KB| CPU占用率: 0.00%| 数据总流量: 0.0KB| wifi总流量: 2280.434
11244KB Java占用内存: 9780KB| CPU占用率: 0.00%| 数据总流量: 0.0KB| wifi总流量: 2280.434
```

为什么要做安全合规测试?

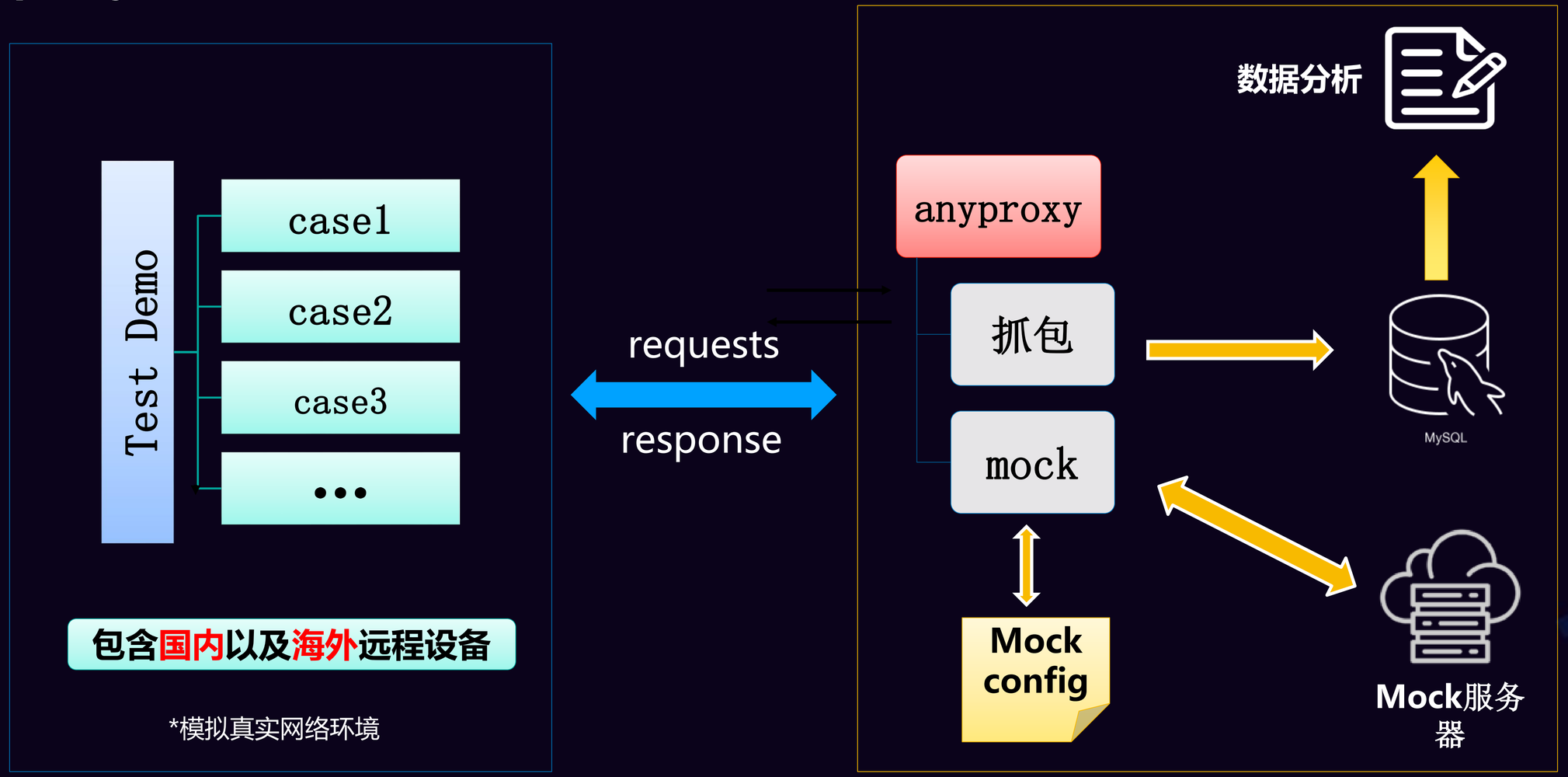
- 国内工信部app隐私保护条例
- 各应用市场隐私条款
- 欧盟GDPR规则
- 美国COPPA法规
- api滥用
- ……

隐私&安全



如何测试？

基于anyproxy的网络交互分析



1、触发SDK调用

2、HOOK网络请求

基于anyproxy的网络交互分析

网络交互数据统计：

requestMethod	resouce	RequestData	ResponseData	totalFlow	start_time
GET	https://r t. .cn/ sChecl	0.0	0.034	0.034	2019-03-29
POST	https:// f.f. date	0.286	0.624	0.90999999	2019-03-29
POST	https:// 26.3 M: lterJs Jr	0.318	32.784	33.102	2019-03-29
POST	https:// n rci Com rc	1.918	1.6	3.518	2019-03-29
GET	https:/ s.s cd m/ar v5/ 9(	0.0	43.236	43.236	2019-03-29
GET	https .s nuc /anc 5/ 9(	0.0	0.16	0.16	2019-03-29
GET	htt /m q m/ o ne	1.084	0.044	1.12800000	2019-03-29
GET	htt //m dl. a /a	0.19	0.24	0.43	2019-03-29
POST	htt // 26. a Msg r Jr	1.838	1.6	3.438	2018-03-29
POST	http .2 3 p Jr	0.54	0.0	0.54	2018-03-29

隐私代码检测

敏感信息获取包括imei，serial，Android_id，短信、电话等。

分为**动态监测，静态检测**。

- **动态监测**：监测sdk运行时是否调用系统隐私api接口。
- **静态检测**：直接扫描源码中是否有获取隐私信息相关的代码。

静态代码检测流程

自研隐私代码扫描引擎

自动上传源码或apk文件

Jenkins

手动上传源码或apk文件

隐私代码扫描

权限检测

日志检测

病毒扫描

Sonar扫描

Firelin e扫描

内网代码扫描服务器

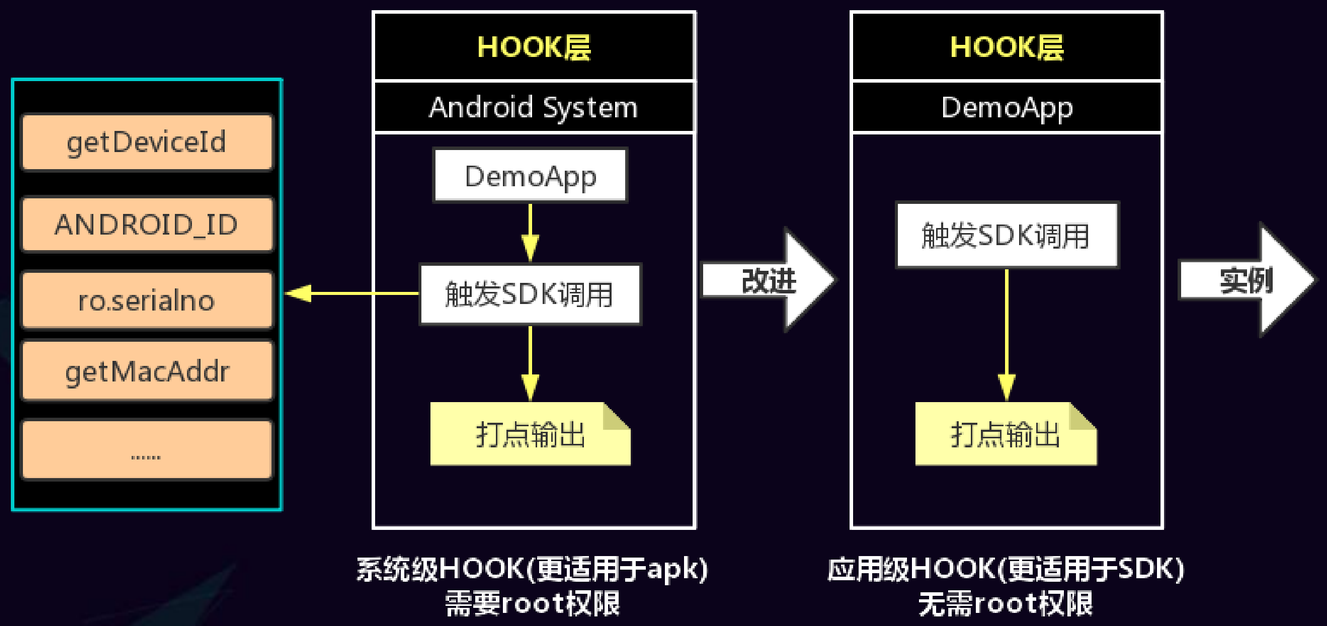
自动化报告

MySQL

Web展示

集成多家病毒扫描引擎

动态监测流程



系统级HOOK(更适用于apk)
需要root权限

应用级HOOK(更适用于SDK)
无需root权限

填写HOOK清单

```
/*
* 配置HOOK 名单
* */
public static String hookClassConfig = "java.net.NetworkInterface,NetworkInterface,getNetworkInterfac
                                       "android.telephony.TelephonyManager,TelephonyManager,getDevic
```

触发HOOK操作

```
/**
 * 获取设备的id
 * @return
 */
public String getImei() {
    String imei = null;
    try {
        TelephonyManager telephonyManager = (TelephonyManager) getSystemService(Context.TELEPHONY_SERVICE)
        imei = telephonyManager.getDeviceId();
    } catch (Exception e) {
        e.printStackTrace();
    }
    return imei;
}
```

打印输出

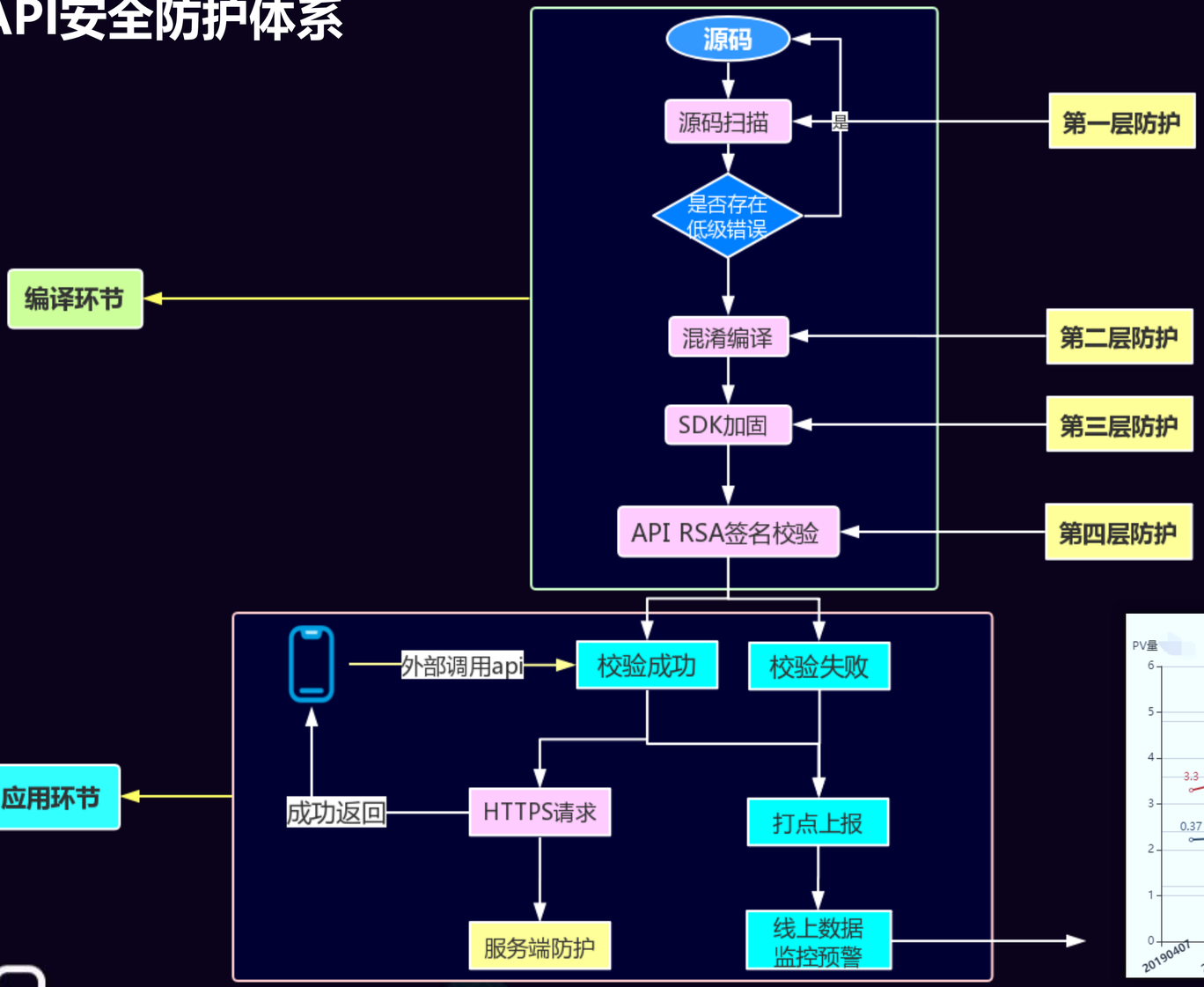
```
E/hook_test: Start monitoring
E/art_hook: NOT FOUND art_quick_resolution_trampoline in /system/lib64/libart.so@0xbd1ce00
E/art_hook: NOT FOUND art_quick_to_interpreter_bridge in /system/lib64/libart.so@0xbd1ce00
E/art_hook: NOT FOUND art_jni_dlsym_lookup_stub in /system/lib64/libart.so@0xbd1ce000
E/art_hook: NOT FOUND _ZN3artL19gJniNativeInterfaceE in /system/lib64/libart.so@0xbd1ce000
E/hook_test: beforeHook
E/hook_test: in hook intercepter 12589056 TelephonyManager.getDeviceId(NO_PARAM)
E/hook_test: afterHook
E/hook_test: [redacted]030649759
E/hook_test: beforeHook
E/hook_test: in hook intercepter 12589056 NetworkInterface.getNetworkInterfaces(NO_PARAM)
E/hook_test: afterHook
E/hook_test: 9[redacted]8:69:29
```

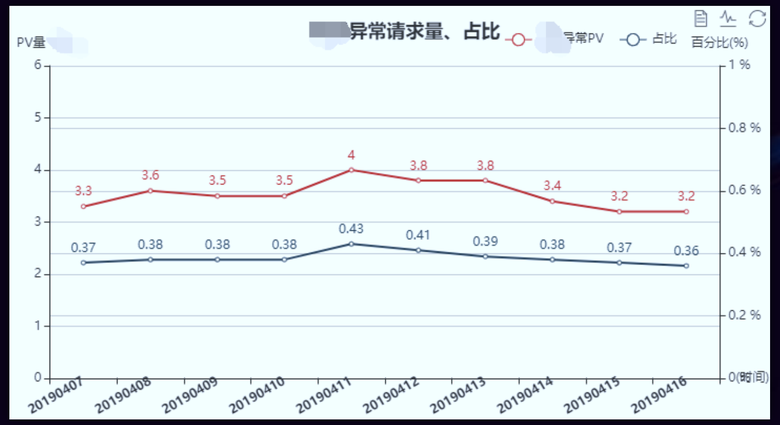
SDK中API安全防护体系

- API的占比逐渐增高，直接导致风险加大。
- 问题：SDK被**反编译**，API被**恶意调用**。
- 结果：造成**数据泄露、数据污染、恶意流量**消耗等。

- **如何解决？如何防护？**

SDK中API安全防护体系





四、SDK持续集成测试方案探索

SDK自动化方案的选择

调研了三种SDK

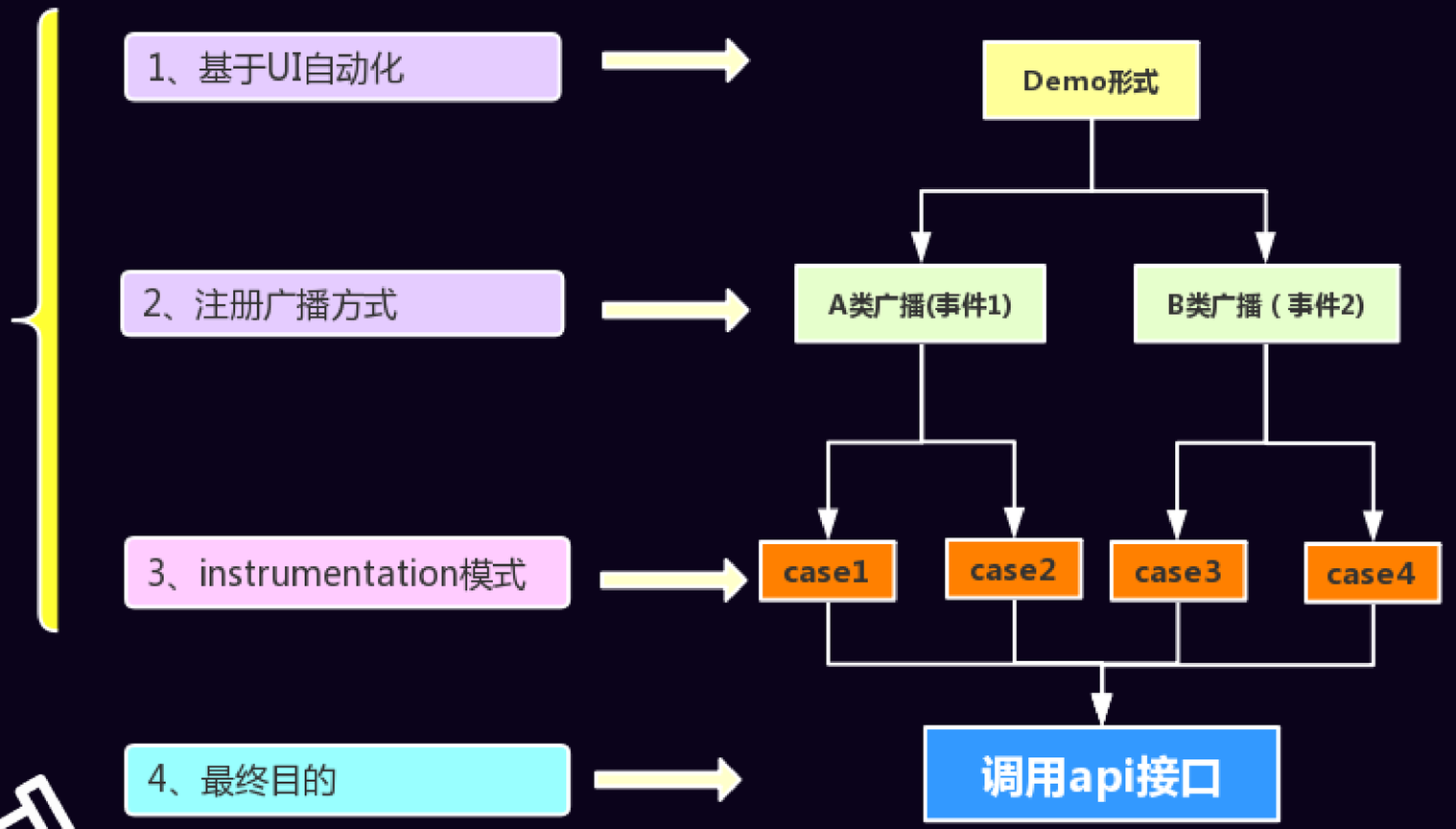
自动化测试方法：

1、基于UI自动化

2、注册广播方式

3、instrumentation模式

4、最终目的



SDK自动化方案的选择

- Instrumentation框架使SDK自动化测试更加快捷高效

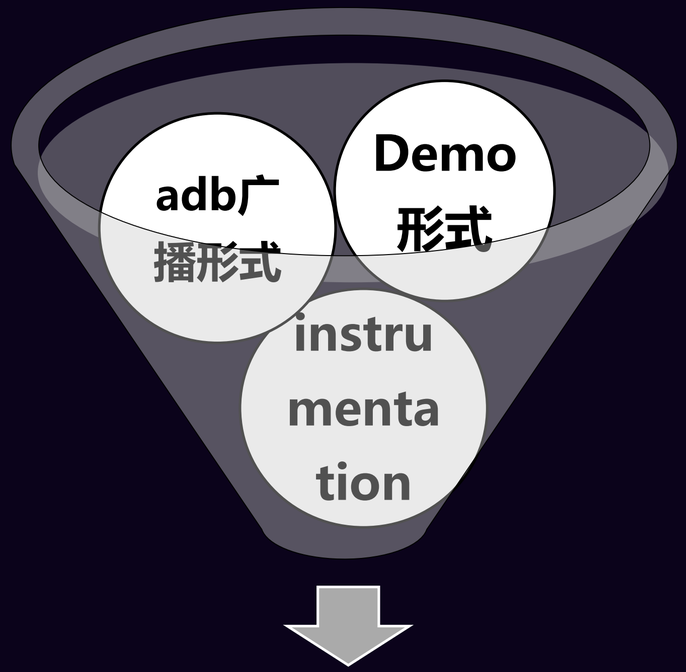
Instrumentation是Android测试的核心框架，可使用它进行Android应用和SDK的单元测试以及自动化测试。

- Instrumentation特点

1. 无需编写activity代码
2. 无需注册广播等
3. 自动生成报告
4. 适合做单元测试

ps：我们把SDK中每个api接口作为一个单元用例

5. 将测试代码和应用代码分离
6. 自动给与权限

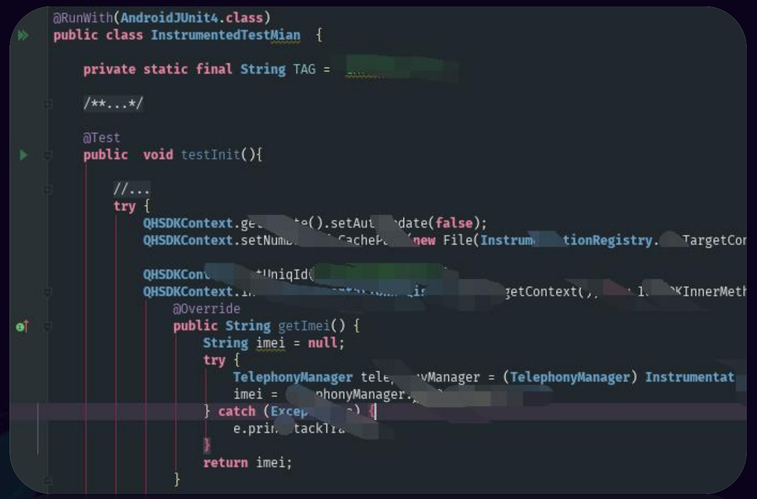


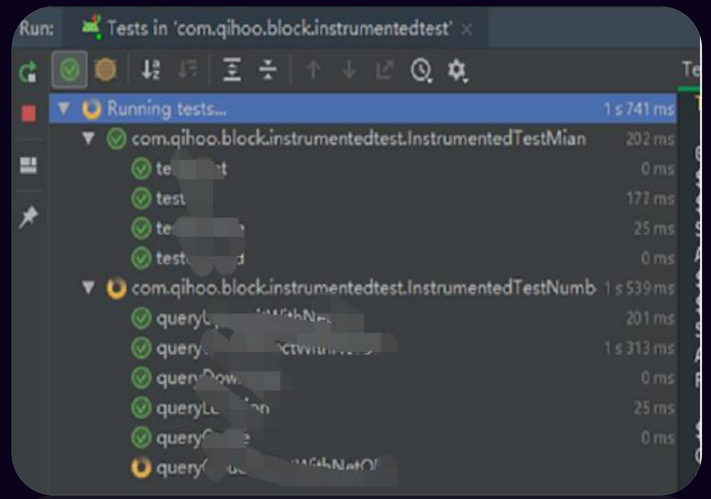
Instrumentation Test

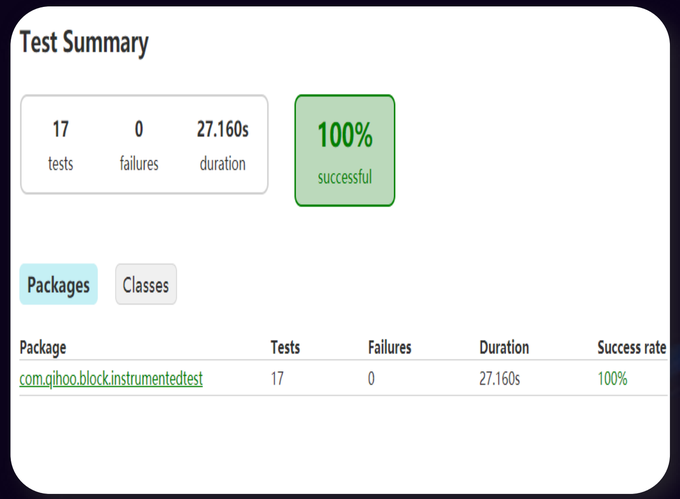
SDK自动化方案的选择

- Instrumentation自动化流程简析

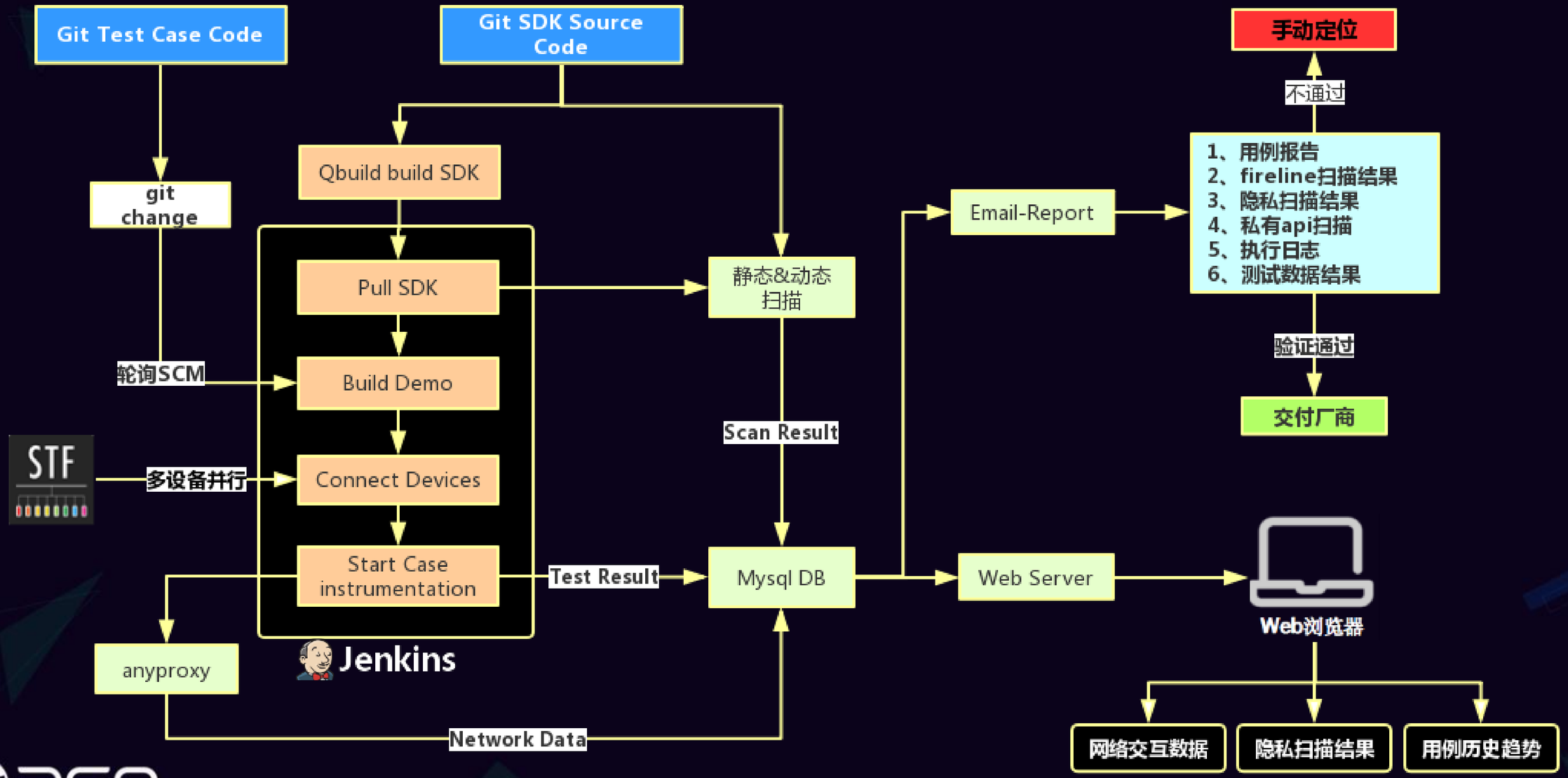
编写case　　运行case　　生成报告





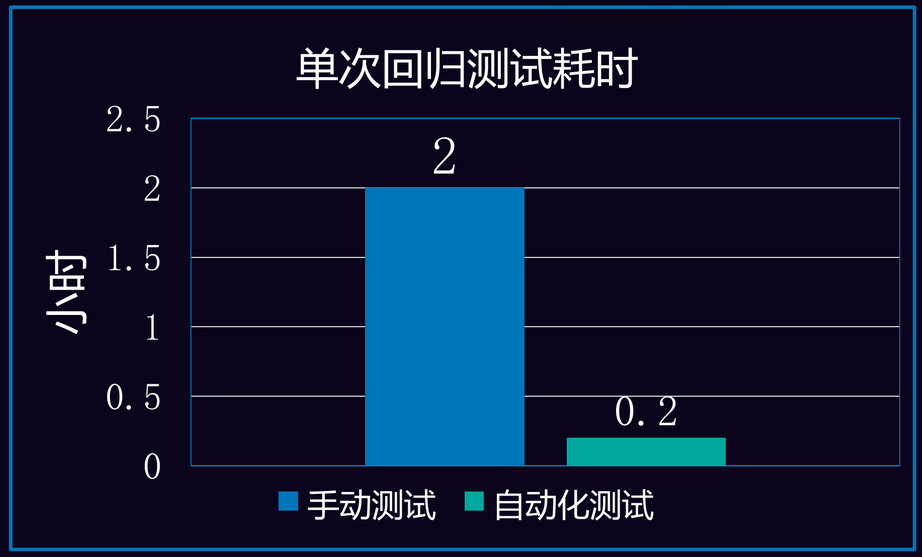


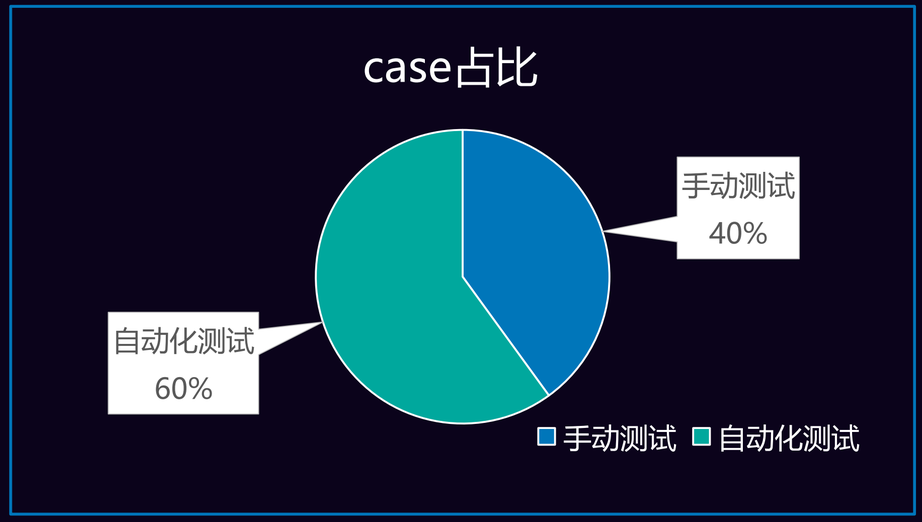
CI搭建流程


Git Test Case Code
Git SDK Source Code
git change
Qbuild build SDK
Pull SDK
Build Demo
Connect Devices
Start Case instrumentation
轮询SCM
STF
多设备并行
anyproxy
Jenkins
静态&动态扫描
Scan Result
Test Result
Mysql DB
Network Data
Email-Report
Web Server
手动定位
不通过
1、用例报告
2、fireline扫描结果
3、隐私扫描结果
4、私有api扫描
5、执行日志
6、测试数据结果
验证通过
交付厂商
Web浏览器
网络交互数据
隐私扫描结果
用例历史趋势

CI持续集成测试的优势及成果

- 节省人力，提升效率
- 快速回归，减少疏漏（目前已经支持case 60条+）
- 加快了安全合规、隐私扫描速度
- 减少风险，具有一致性（SDK迭代速度快）
- 提高产品质量，增加SDK信任度





Thank you