神策数据 SDK 源码安全审计报告

(Android SDK)

■ 文档编号		■ 密级	商业机密
■ 版本编号	V1.0	■ 日期	2020 年 08 月 02 日

■ **版本变更记录**

时间	版本	说明	修改人
2020-8-2	V1.0	文档创建	刘红涛
2020-8-28	V2.0	更新文档	武立鑫

■ **适用性声明**

本文档为北京神州绿盟信息安全科技股份有限公司（以下简称“绿盟科技”）在神策数据安全服务中所提供的源代码安全审计报告，适用于相关技术人员在针对发现的漏洞进行安全修复时参考。

目录

一. 执行摘要

经神策数据授权，绿盟科技对其 Android SDK 进行源代码安全审计。审计发现的问题总结如下：

表 1.1 发现问题汇总

威胁程度	问题	描述
严重问题	无	--
中等问题	无	--
轻度风险	无	--
风险提示	3.7 加密协议有效性	如果客户端与服务器之间的通信加密协议实现不当，攻击者将有机会对当前网络环境中其他合法用户的通信内容进行窃听甚至篡改。
	3.8 Activity 组件防劫持	Activity 劫持，又称界面劫持，是一种恶意覆盖其它 APP 界面的本地攻击方式。成功的 Activity 劫持将会替换客户端的启动界面，以诱骗用户输入认证凭据或其它各种五花八门的诈骗。
	3.9 屏幕截屏和录像保护	在密码输入屏幕中，密码可以在屏幕上清晰显示。 在处理个人信息的屏幕中，如果屏幕截图功能处于启用状态，则可能会导致个人敏感信息泄漏

通过本次代码审计活动，我们发现 Android SDK 设计了较好的安全防护策略和技术防控功能。但是因为种种原因，开发过程存在需注意的风险点。我们建议开发商在保证业务功能正常使用的基础上有针对性的避免风险，切实保障系统业务稳定运行以及用户信息安全。

二. 服务综述

2.1 审计概述

在对神策数据 Android SDK 进行源代码安全审计活动前，我们会审阅系统相关文档，对技术架构的安全性进行审计和评估，并且熟悉系统实现的业务流程，评估其可能存在的安全风险。在审计活动起始阶段，我们会使用源代码安全审计工具进行全面的审计，发现其存在的不安全编码并进行人工分析和确认。自动化审计完成后，审计实施人员对重要业务场景进行深入分析。不论是客户端代码还是服务端代码，除常规的安全漏洞如 SQL 注入、XSS、CSRF 等漏洞外，我们还会参照相关行业标准及规范，对其合规性进行审计。

图 2.1 源代码安全审计技术模型

2.2 审计依据

为保证项目实施的标准化和规范化，本次源代码安全审计活动主要参考如下依据：

国内通用标准、指南或规范

- GB/T 22239 信息安全技术 网络安全等级保护基本要求
- ISO/IEC 27001:2013 信息技术-安全技术-信息系统规范与使用指南
- ISO/IEC 13335-1: 2004 信息技术-安全技术-信息技术安全管理指南
- ISO/IEC TR 15443-1: 2005 信息技术安全保障框架

- GB/T 20984-2007 信息安全技术 信息安全风险评估规范
- GB/T 19715.1-2005 信息技术-信息技术安全管理指南
- GB/T 19716-2005 信息技术-信息安全管理实用规则
- GB/T 18336-2015 信息技术-安全技术-信息技术安全性评估准则
- GB 17859-1999 计算机信息系统安全保护等级划分准则
- GB/T 20984-2007 信息安全技术 信息安全风险评估规范
- 绿盟科技安全编码规范

国际标准、指南或规范

- OWASP Top 10（2013）
- OWASP Top 10（2017）
- OWASP Development Guide
- OWASP Testing Guide v4
- OWASP Application Security Verification Standard
- OWASP Secure Coding Practices
- OSSTMM OSSTMM_Web_App_Alpha

2.3 项目实施

2.3.1 审计对象

表 2.1 审计对象

单位	系统名称	审计对象
神策数据	神策分析 SDK	Android SDK

2.3.2 审计时间

表 2.2 项目实施时间

源代码安全审计时间	
起始时间	2020 年 7 月 29 日
结束时间	2020 年 8 月 2 日

2.3.3 审计人员

表 2.3 源代码安全审计实施人员

安全审计人员名单					
姓名	刘红涛	**所属部门**	华东安全服务交付部	**联系方式**	liuhongtao@nsfocus.com

2.3.4 审计工具

本次源代码安全审计活动所使用的工具主要有：

表 2.4 源代码安全审计工具

工具名称	工具版本	简要介绍
HP Fortify SCA	4.02	源代码安全审计工具
SpotBugs	3.1.1	源代码安全审计工具
Eclipse	Oxygen 2	开发 IDE 工具
IntelliJ IDEA	2018.1	开发 IDE 工具
Sublime Text	3.0	代码查看工具

三. 源代码安全审计结果

3.1 组件导出安全

3.1.1 漏洞概述

风险描述	导出组件没有进行严格的访问控制，那么其它 APP 就可以通过调用这个导出组件的接口来访问原本没有声明权限的功能，构成本地权限提升
威胁级别	严重
可利用性	容易
测试结果	安全

3.1.2 代码安全分析

分析查看 AndroidManifest.xml 文件中声明的组件可导出权限，判断是否可导出且存在敏感数据。

代码位置： src/main/AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.sensorsdata.analytics.android.sdk">

    <!-- 同步数据需要网络权限 -->
    <uses-permission android:name="android.permission.INTERNET" />
    <!-- 获取网络状态 -->
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <!-- 获取运营商信息 -->
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />

    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />

    <application>
        <provider
            android:name="com.sensorsdata.analytics.android.sdk.data.SensorsDataContentProvider"
            android:authorities="${applicationId}.SensorsDataContentProvider"
            android:enabled="true"
            android:exported="false" />

        <meta-data
            android:name="com.sensorsdata.analytics.android.MainProcessName"
            android:value="${applicationId}" />
    </application>
</manifest>
```

代码安全分析： 无可导出组件。

3.1.3 安全建议

1. 避免不必要的组件导出，尤其是涉及敏感数据的组件。

3.2 敏感权限申请安全

3.2.1 漏洞概述

风险描述	不必要的权限申请很有可能被恶意程序利用，造成用户敏感信息泄露和短信扣费风险，同时可能降低应用程序的可信度
威胁级别	低危
可利用性	容易
测试结果	安全

3.2.2 代码安全分析

分析查看 AndroidManifest.xml 文件中申请的系统权限，判断是否为敏感权限。

代码位置：src/main/AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.sensorsdata.analytics.android.sdk">

    <!-- 同步数据需要网络权限 -->
    <uses-permission android:name="android.permission.INTERNET" />
    <!-- 获取网络状态 -->
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <!-- 获取运营商信息 -->
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />

    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />

    <application>
        <provider
            android:name="com.sensorsdata.analytics.android.sdk.data.SensorsDataContentProvider"
            android:authorities="${applicationId}.SensorsDataContentProvider"
            android:enabled="true"
            android:exported="false" />

        <meta-data
            android:name="com.sensorsdata.analytics.android.MainProcessName"
            android:value="${applicationId}" />
    </application>
</manifest>
```

代码安全分析：无敏感权限申请。

3.2.3 安全建议

限制不必要权限的使用。

3.3 备份权限安全

3.3.1 漏洞概述

风险描述	被测应用的 AndroidManifest.xml 文件中 allowBackup 属性值没有被设置为 false，可通过 adb backup 对应用数据进行备份，在无 root 的情况下可以导出应用中存储的所有数据，造成用户数据泄露
威胁级别	中危
可利用性	容易
测试结果	安全

3.3.2 代码安全分析

查看在 AndroidManifast.xml 文件下是否存在暴露的 allowbackup。

代码位置：src/main/AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.sensorsdata.analytics.android.sdk">

    <!-- 同步数据需要网络权限 -->
    <uses-permission android:name="android.permission.INTERNET" />
    <!-- 获取网络状态 -->
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <!-- 获取运营商信息 -->
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />

    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />

    <application>
        <provider
            android:name="com.sensorsdata.analytics.android.sdk.data.SensorsDataContentProvider"
            android:authorities="${applicationId}.SensorsDataContentProvider"
            android:enabled="true"
            android:exported="false" />

        <meta-data
            android:name="com.sensorsdata.analytics.android.MainProcessName"
            android:value="${applicationId}" />
    </application>
</manifest>
```

代码安全分析：无备份权限申请。

3.3.3 安全建议

1. 将 AndroidMainfast.xml 文件下 Allowbackup 属性设置为 false 或不做声明；
2. 使用良好加密，例如启动应用时计算设备唯一标识，从服务端获取对应密钥后解密本地配置。

3.4 调试权限配置安全

3.4.1 漏洞概述

风险描述	被测应用的 AndroidManifest.xml 文件中 Debuggable 属性值被设置为 true，可以设置断点来控制程序的执行流程，在应用程序运行时修改其行为。
威胁级别	中风险
可利用性	容易
测试结果	安全

3.4.2 代码安全分析

查看在 AndroidManifast.xml 文件下是否存在暴露的 Debuggable 配置。

代码位置：src/main/AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.sensorsdata.analytics.android.sdk">

    <!-- 同步数据需要网络权限 -->
    <uses-permission android:name="android.permission.INTERNET" />
    <!-- 获取网络状态 -->
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <!-- 获取运营商信息 -->
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />

    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />

    <application>
        <provider
            android:name="com.sensorsdata.analytics.android.sdk.data.SensorsDataContentProvider"
            android:authorities="${applicationId}.SensorsDataContentProvider"
            android:enabled="true"
            android:exported="false" />

        <meta-data
            android:name="com.sensorsdata.analytics.android.MainProcessName"
            android:value="${applicationId}" />
    </application>
</manifest>
```

代码安全分析：无可导出组件。

3.4.3 安全建议

避免不必要的组件导出，尤其是涉及敏感数据的组件。

3.5 SharedPreferences 安全

3.5.1 漏洞概述

风险描述	如果 APP 在创建 SharedPreference 时，将数据库设置了全局的可读权限，攻击者就有机会恶意读取 SharedPreference 内容，获取敏感信息。
威胁级别	高风险
可利用性	容易
测试结果	安全

3.5.2 代码安全分析

分析查看创建应用 SharedPreference 时，调用了 getSharedPreferences，访问权限是否设置为 MODE_WORLD_READABLE 或者 MODE_WORLD_WRITEABLE。

代码位置：src/main/AndroidManifest.xml

```
public static SharedPreferences getSharedPreferences(Context context) {
        return context.getSharedPreferences(SHARED_PREF_EDITS_FILE,
Context.MODE_PRIVATE);
}
```

代码安全分析：SharedPreference 的权限不存在问题。

3.5.3 安全建议

1. 用 MODE_PRIVATE 模式创建 SharedPreference；
2. 避免在 SharedPreference 中存储明文和敏感信息。

3.6 WebView 组件安全

3.6.1 漏洞概述

风险描述	低版本的 Webview 启用 addJavascriptInterface，实现本地 java 代码和远程 js 代码交互时，会存在远程代码执行漏洞； WebView 组件忽略 ssl 证书异常时，会受到中间人攻击的威胁，可能导致隐私泄露
威胁级别	高风险
可利用性	容易
测试结果	安全

3.6.2 代码安全分析

分析查看 AndroidManifest.xml 文件中声明的组件可导出权限，判断是否可导出且存在敏感数据。

代码位置：

src/main/java/com/sensorsdata/analytics/android/sdk/AppWebViewInterface.java

```
    @JavascriptInterface
        public String sensorsdata_call_app() {
            try {
                if (properties == null) {
                    properties = new JSONObject();
                }
                properties.put("type", "Android");
                String loginId =
SensorsDataAPI.sharedInstance(mContext).getLoginId();
                if (!TextUtils.isEmpty(loginId)) {
                    properties.put("distinct_id", loginId);
                    properties.put("is_login", true);
```

```
                } else {
                    properties.put("distinct_id",
SensorsDataAPI.sharedInstance(mContext).getAnonymousId());
                    properties.put("is_login", false);
                }
                return properties.toString();
            } catch (JSONException e) {
                SALog.i(TAG, e.getMessage());
            }
            return null;
        }

        @JavascriptInterface
        public void sensorsdata_track(String event) {
            try {
                SensorsDataAPI.sharedInstance(mContext).trackEventFromH5(event,
enableVerify);
            } catch (Exception e) {
                com.sensorsdata.analytics.android.sdk.SALog.printStackTrace(e);
            }
        }

        @JavascriptInterface
        public boolean sensorsdata_verify(String event) {
            try {
                if (!enableVerify) {
                    sensorsdata_track(event);
                    return true;
                }
                return
```

```
SensorsDataAPI.sharedInstance(mContext)._trackEventFromH5(event);
                } catch (Exception e) {
                    com.sensorsdata.analytics.android.sdk.SALog.printStackTrace(e);
                    return false;
                }
    }
```

代码安全分析：通过声明@JavascriptInterface 来进行代码交互调用。

3.6.3 安全建议

如果是 Android 4.2 之前版本，对 addJavascriptInterface 的输入参数进行过滤；

如果是 Android 4.2 及之后版本，声明@JavascriptInterface 来代替 addjavascriptInterface。。

3.7 加密协议有效性

3.7.1 漏洞概述

风险描述	如果客户端与服务器之间的通信加密协议实现不当，攻击者将有机会对当前网络环境中其他合法用户的通信内容进行窃听甚至篡改。
威胁级别	中风险
可利用性	容易
测试结果	仅作风险提示

3.7.2 代码安全分析

分析查看 AndroidManifest.xml 文件中声明的组件可导出权限，判断是否可导出且存在敏感数据。

代码位置：com/sensorsdata/analytics/android/sdk/HeatMapViewCrawler.java

```
private class CustomTrustManager implements X509TrustManager {
    @Override
    public void checkClientTrusted(X509Certificate[] chain, String authType) {
    }

    @Override
    public void checkServerTrusted(X509Certificate[] chain, String authType) {
    }

    @Override
    public X509Certificate[] getAcceptedIssuers() { return null; }
}

private class CustomHostnameVerifier implements HostnameVerifier {
    @Override
    public boolean verify(String hostname, SSLSession session) { return true; }
}
```

代码安全分析：忽略了对 SSL 证书与 HOST 一致性的检查。

结合业务安全分析：针对该位置的代码属于点击图忽略证书部分，点击图是神策提供的一个用于分析用户点击行为分析的功能，点击图功能的开启只能由内部的业务员扫码开启，二维码是在神策分析后台动态生成，所以只能是内部业务人员开启，开启后才生效。点击图的实现需要上报手机屏幕的截图，会涉及到网络请求。网络请求分为 HTPP 请求和 HTTPS 请求，HTTPS 请求会进行证书的校验，为了满足部分用户不进行校验检查的需求，神策 Android SDK 增加相关代码开关用于忽略证书校验，属于动态配置的功能。默认是开关关闭状态，进行证书校验，如果开启开关，则不校验证书。

所以只要不进行手动开启则进行证书校验，并且功能只能由内部业务人员开启使用，风险可控。

3.7.3 安全建议

建议使用 SSL 协议，并在客户端对服务端的证书进行验证。如果自行实现加密协议，建议在客户端预先存储服务端公钥，在建立会话时随机生成对称加密密钥，用服务端公钥加密并发送给服务端，随后服务端用私钥解密后，正式开始进行通信。加密过程尽量避免使用 ECB 模式。

3.8 Activity 组件防劫持

3.8.1 漏洞概述

风险描述	Activity 劫持，又称界面劫持，是一种恶意覆盖其它 APP 界面的本地攻击方式。成功的 Activity 劫持将会替换客户端的启动界面，以诱骗用户输入

	认证凭据或其它各种五花八门的诈骗。
威胁级别	低风险
可利用性	容易
测试结果	仅作风险提示

3.8.2 代码安全分析

分析查看 AndroidManifest.xml 文件中声明的组件可导出权限，判断是否可导出且存在敏感数据。

代码位置：com/sensorsdata/analytics/android/sdk/HeatMapViewCrawler.java

```
private class LifecycleCallbacks
        implements Application.ActivityLifecycleCallbacks {

    public LifecycleCallbacks() {
    }

    @Override
    public void onActivityCreated(Activity activity, Bundle bundle) {
    }

    @Override
    public void onActivityStarted(Activity activity) {
    }

    @Override
    public void onActivityResumed(Activity activity) { mEditState.add(activity); }

    @Override
    public void onActivityPaused(Activity activity) { mEditState.remove(activity); }

    @Override
    public void onActivityStopped(Activity activity) {

    }

    @Override
    public void onActivitySaveInstanceState(Activity activity, Bundle bundle) {
    }

    @Override
    public void onActivityDestroyed(Activity activity) {
    }
}
```

代码安全分析：Activity 组件不具备防护 Activity 劫持的能力。

结合业务安全分析：Activity 劫持代码属性点击图上报屏幕截图部分。点击图功能需要上报每个 Activity 页面的截图，所以会通过 Android 系统提供的 ActivityLifecycleCallbacks 回调接口进行检测 Activity 的生命周期，SDK 这里通过 Android 系统提供的 ActivityLifecycleCallbacks 回调接口进行页面的回调监听，不对 Activity 进行二次处理，所以可避免发生劫持风险。同时关于 Activity 的可导出权限由 App 进行配置，SDK 不做配置，

故对 Activity 无劫持的影响。并且点击图功能的开启只能由内部的业务员扫码开启，二维码是在神策分析后台动态生成，所以只能是内部业务人员开启，开启后才生效。

3.8.3 安全建议

- **方案一：设置最小 SDK 版本大于等于 21（Android 5.0）**

在 Android 5.0 之后，Google 从系统机制层面修复了这个问题。用户必须手动授权“允许查看使用情况的应用”，否则攻击者无法接收到其它 APP 运行状态有关的广播事件。但此方案也将导致 APP 在 Android 4.4.4 及以前的设备上无法安装和运行。

- **方案二：检测当前应用 Activity 处于停止状态时是否被劫持**

在 Acitivity 的 onStop 方法中调用封装的 AntiHijackingUtil 类（检测系统程序白名单）检测程序是否被系统程序覆盖，并给予用户合理的提示。

```
import android.app.ActivityManager;
import android.app.KeyguardManager;
import android.content.Context;
import android.content.Intent;
import android.content.pm.ApplicationInfo;
import android.content.pm.PackageManager;
import android.content.pm.ResolveInfo;

import java.lang.reflect.Field;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

/**
 * Description: Activity 反劫持检测工具
*/
public class AntiHijackingUtil {
    public static final String TAG = "AntiHijackingUtil";

    /**
     * 检测当前 Activity 是否安全
     */
    public static boolean checkActivity(Context context) {
```

```
                PackageManager pm = context.getPackageManager();
                // 查询所有已经安装的应用程序
                List<ApplicationInfo> listAppcations =

pm.getInstalledApplications(PackageManager.GET_UNINSTALLED_PACKAGES);
                Collections.sort(listAppcations,                                    new
ApplicationInfo.DisplayNameComparator(pm));// 排序

                List<String> safePackages = new ArrayList<>();
                for (ApplicationInfo app : listAppcations) {// 这个排序必须有.
                    if ((app.flags & ApplicationInfo.FLAG_SYSTEM) != 0) {
                        safePackages.add(app.packageName);
                    }
                }
                // 得到所有的系统程序包名放进白名单里面.
                ActivityManager activityManager =
                        (ActivityManager)
context.getSystemService(Context.ACTIVITY_SERVICE);
                String runningActivityPackageName;
                int sdkVersion;
                try {
                    sdkVersion = Integer.valueOf(android.os.Build.VERSION.SDK);
                } catch (NumberFormatException e) {
                    sdkVersion = 0;
                }
                if (sdkVersion >= 21) {// 获取系统 api 版本号,如果是 5x 系统就用这个方法获
取当前运行的包名
                    runningActivityPackageName = getCurrentPkgName(context);
                } else {
                    runningActivityPackageName =

activityManager.getRunningTasks(1).get(0).topActivity.getPackageName();
                }
                // 如果是 4x 及以下,用这个方法.
                if (runningActivityPackageName != null) {
                    // 有些情况下在 5x 的手机中可能获取不到当前运行的包名, 所以要非空
判断。
                    if (runningActivityPackageName.equals(context.getPackageName())) {
```

```
                    return true;
                }
                // 白名单比对
                for (String safePack : safePackages) {
                    if (safePack.equals(runningActivityPackageName)) {
                        return true;
                    }
                }
            }
            return false;
        }

        private static String getCurrentPkgName(Context context) {
            // 5x 系统以后利用反射获取当前栈顶 activity 的包名.
            ActivityManager.RunningAppProcessInfo currentInfo = null;
            Field field = null;
            int START_TASK_TO_FRONT = 2;
            String pkgName = null;
            try {
                // 通过反射获取进程状态字段.
                field = ActivityManager.RunningAppProcessInfo.class.getDeclaredField("processState");
            } catch (Exception e) {
                e.printStackTrace();
            }
            ActivityManager am = (ActivityManager) context.getSystemService(Context.ACTIVITY_SERVICE);
            List appList = am.getRunningAppProcesses();
            ActivityManager.RunningAppProcessInfo app;
            for (int i = 0; i < appList.size(); i++) {
                //ActivityManager.RunningAppProcessInfo app : appList
                app = (ActivityManager.RunningAppProcessInfo) appList.get(i);
                //表示前台运行进程.
                if (app.importance == ActivityManager.RunningAppProcessInfo.IMPORTANCE_FOREGROUND) {
                    Integer state = null;
                    try {
```

```
                    state = field.getInt(app);// 反射调用字段值的方法,获取该进程的状态.
                } catch (Exception e) {
                    e.printStackTrace();
                }
                // 根据这个判断条件从前台中获取当前切换的进程对象
                if (state != null && state == START_TASK_TO_FRONT) {
                    currentInfo = app;
                    break;
                }
            }
        }
        if (currentInfo != null) {
            pkgName = currentInfo.processName;
        }
        return pkgName;
    }

    /**
     * 判断当前是否在桌面
     *
     * @param context 上下文
     */
    public static boolean isHome(Context context) {
        ActivityManager mActivityManager =
                (ActivityManager) context.getSystemService(Context.ACTIVITY_SERVICE);
        List<ActivityManager.RunningTaskInfo> rti = mActivityManager.getRunningTasks(1);
        return getHomes(context).contains(rti.get(0).topActivity.getPackageName());
    }

    /**
     * 获得属于桌面的应用的应用包名称
     *
     * @return 返回包含所有包名的字符串列表
```

```
     */
    private static List<String> getHomes(Context context) {
        List<String> names = new ArrayList<String>();
        PackageManager packageManager = context.getPackageManager();
        Intent intent = new Intent(Intent.ACTION_MAIN);
        intent.addCategory(Intent.CATEGORY_HOME);
        List<ResolveInfo> resolveInfo =
packageManager.queryIntentActivities(intent,
                PackageManager.MATCH_DEFAULT_ONLY);
        for (ResolveInfo ri : resolveInfo) {
            names.add(ri.activityInfo.packageName);
        }
        return names;
    }

    /**
     * 判断当前是否在锁屏再解锁状态
     *
     * @param context 上下文
     */
    public static boolean isReflectScreen(Context context) {
        KeyguardManager mKeyguardManager =
                (KeyguardManager)
context.getSystemService(Context.KEYGUARD_SERVICE);
        return mKeyguardManager.inKeyguardRestrictedInputMode();
    }
}
```

重写 onKeyDown 方法、onPause 方法和 onStop 方法，判断是否给予告警提示：

- ✓ 程序进入后台是否为用户主动行为（触摸返回键或 HOME 键），是则无需弹出警示；
- ✓ APP 在锁屏再解锁的情况下无需给用户弹出警告提示。

```
import android.os.Bundle;
import android.os.Looper;
import android.support.v7.app.AppCompatActivity;
import android.text.TextUtils;
import android.view.KeyEvent;
import android.view.View;
```

```
import android.view.View.OnClickListener;
import android.view.inputmethod.EditorInfo;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

/**
 * A login screen that offers login via email/password.
 */
public class LoginActivity extends AppCompatActivity {



    // UI references.
    private AutoCompleteTextView mEmailView;
    private EditText mPasswordView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        // Set up the login form.
        mEmailView = findViewById(R.id.email);

        mPasswordView = findViewById(R.id.password);
        mPasswordView.setOnEditorActionListener(new
TextView.OnEditorActionListener() {
            @Override
            public boolean onEditorAction(TextView textView, int id, KeyEvent
keyEvent) {
                if (id == EditorInfo.IME_ACTION_DONE || id ==
EditorInfo.IME_NULL) {
                    attemptLogin();
                    return true;
                }
                return false;
```

```
                }
            });

            Button mEmailSignInButton = findViewById(R.id.email_sign_in_button);
            mEmailSignInButton.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View view) {
                    attemptLogin();
                }
            });
        }

        @Override
        protected void onStop() {
            super.onStop();
            new Thread(new Runnable() {
                @Override
                public void run() {
                    // 白名单
                    boolean safe =
AntiHijackingUtil.checkActivity(getApplicationContext());
                    // 系统桌面
                    boolean isHome =
AntiHijackingUtil.isHome(getApplicationContext());
                    // 锁屏操作
                    boolean isReflectScreen =
AntiHijackingUtil.isReflectScreen(getApplicationContext());
                    // 判断程序是否当前显示
                    if (!safe && !isHome && !isReflectScreen) {
                        Looper.prepare();
                        Toast.makeText(getApplicationContext(),
R.string.activity_safe_warning,
                                Toast.LENGTH_LONG).show();
                        Looper.loop();
                    }
                }
            }).start();
```

```
}

/**
 * Attempts to sign in or register the account specified by the login form.
 * If there are form errors (invalid email, missing fields, etc.), the
 * errors are presented and no actual login attempt is made.
 */
private void attemptLogin() {
    // Reset errors.
    mEmailView.setError(null);
    mPasswordView.setError(null);

    // Store values at the time of the login attempt.
    String email = mEmailView.getText().toString();
    String password = mPasswordView.getText().toString();

    boolean cancel = false;
    View focusView = null;

    // Check for a valid password, if the user entered one.
    if (!TextUtils.isEmpty(password)) {
        focusView = mPasswordView;
        cancel = true;
    }

    // Check for a valid email address.
    if (TextUtils.isEmpty(email)) {
        mEmailView.setError(getString(R.string.error_field_required));
        focusView = mEmailView;
        cancel = true;
    }
    if (cancel) {
        // There was an error; don't attempt login and focus the first
        // form field with an error.
        focusView.requestFocus();
    }
```

```
        }
}
```

3.9 屏幕截屏和录像保护

3.9.1 漏洞概述

风险描述	在密码输入屏幕中，密码可以在屏幕上清晰显示。 在处理个人信息的屏幕中，如果屏幕截图功能处于启用状态，则可能会导致个人敏感信息泄漏
威胁级别	低风险
可利用性	容易
测试结果	仅作风险提示

3.9.2 代码安全分析

分析查看 AndroidManifest.xml 文件中声明的组件可导出权限，判断是否可导出且存在敏感数据。

代码位置：com/sensorsdata/analytics/android/sdk/HeatMapViewCrawler.java

```
private class LifecycleCallbacks
        implements Application.ActivityLifecycleCallbacks {

    public LifecycleCallbacks() {
    }

    @Override
    public void onActivityCreated(Activity activity, Bundle bundle) {
    }

    @Override
    public void onActivityStarted(Activity activity) {
    }

    @Override
    public void onActivityResumed(Activity activity) { mEditState.add(activity); }

    @Override
    public void onActivityPaused(Activity activity) { mEditState.remove(activity); }

    @Override
    public void onActivityStopped(Activity activity) {

    }

    @Override
    public void onActivitySaveInstanceState(Activity activity, Bundle bundle) {
    }

    @Override
    public void onActivityDestroyed(Activity activity) {
    }
}
```

代码安全分析：Activity 组件不具备防护截屏的能力。

结合业务安全分析：在点击图的功能中，为了展示实际的点击效果，需要使用屏幕截屏的功能。神策 SDK 中的截屏方法都是使用系统方法进行截屏。并且功能的开启只能由内部的业务员扫码开启，二维码是在神策分析后台动态生成，所以只能是内部业务人员开启。同时可由 App 开发人员配置相关的启动主页面，此风险点需 APP 开发人员在开发相关 APP 时避免截屏密码等敏感信息页面，风险可控。

3.9.3 安全建议

MediaProject 是 Android 5.0 追加的新功能，APP 只要声明相关权限，即可进行录音和屏幕录像。此种攻击方式无需 ROOT 权限，防范起来也比较容易。只要对相关 Activity 设置 FLAG_SECURE 属性，基本上可以阻止所有用户态程序的屏幕录像，包括 MediaProject、screencap、以及使用电源键和音量减小键的内置截屏功能。示例代码如下：

```
@Override
public void onCreate(Bundle saveInstanceState) {
    [...]

    Window window = getWindow();
    window.addFlags(WindowManager.LayoutParams.FLAG_SECURE);
```

```
    setContentView(R.layout.passwordInputScreen);

    [...]
}
```

四. 审计结果及建议

4.1 源代码安全审计结果

通过本次源代码安全审计活动，我们发现 Android SDK 设计了较好的安全防护策略和技术防控功能，但是因为种种原因，开发过程存在需注意的风险点。我们建议开发商在保证业务功能正常使用的基础上有针对性的避免风险，切实保障系统业务稳定运行以及用户个人信息安全。

本次对 Android SDK 的源代码安全审计，已发现安全问题的开发建议如下：

表 4.1 源代码安全审计结果及建议

发现问题	威胁级别	安全建议
加密协议有效性	风险提示	结合业务因素分析，仅作风险提示，可选择不做代码修改
Activity 组件防劫持	风险提示	结合业务因素分析，仅作风险提示，可选择不做代码修改
屏幕截屏和录像保护	风险提示	结合业务因素分析，仅作风险提示，可选择不做代码修改

4.2 整体安全改进建议

针对本次源代码安全审计发现的安全问题，以及持续保证应用系统的安全防护能力，我们给出如下安全编码建议：

1. 梳理应用系统接收参数及数据的格式，通过过滤器/拦截器的方式，对用户输入数据中可能包含的攻击利用字符进行整体处理，如<、>、’、”、*、|、,、.、\等；
2. 数据库在设计过程中对于主键设计为 UUID 格式，应用程序配合数据库设计实现，降

低因存在水平越权所导致的危害；

3. 应用程序实现文件上传功能时，应通过白名单的方式限定上传文件的类型，保存在服务器中时，应对保存的文件重新命名，并保存于中间件可解析的目录中，如独立的文件服务器。
4. 对于文件下载功能，应判断请求下载的文件是否处于指定目录范围之内。
5. 设计实现系统容错机制，自定义友好的异常错误显示页面，当发生 404、500 错误时，跳转到指定页面，防范服务器端敏感信息泄露。
6. 涉及短信验证码以及发送邮件相关功能时，应在服务器端代码中限定发送的频率，防范接口被滥用对其他用户的正常生活造成影响。
7. 应用系统发布时，对前端 JavaScript 代码进行 YUI Compressor 压缩处理，删除相关注释，并且通过压缩降低代码的可阅读性。
8. 保证业务系统的运行环境安全，安装中间件当前版本的最新版，应用程序所引用的组件为最新版本。
9. 定制安全编码规范，提高开发人员的安全编码意识；
10. 在系统设计和编码过程中，关注行业标准相关文件的要求，提高系统的合规性。

五. 感谢

感谢项目实施过程中神策数据相关部门协调工作，感谢相关开发人员的积极配合，也感谢绿盟科技项目组成员付出的努力，通过大家有效的沟通和积极协作，使得本次源代码安全审计工作顺利完成。

附录 A漏洞风险定级

风险值计算方式说明如下：

评定维度	说明
影响范围系数 F	大（3）：能够获取大量数据、敏感数据或影响大量用户。 中（2）：能够获取部分数据、一般敏感数据或影响部分用户 小（1）：能够获取少量数据、非敏感数据或影响少量用户
漏洞系数 Y	权限获取类（20）:可获取服务器权限的漏洞 数据获取类（8）：可导致结构化数据批量获取的漏洞 非法访问类（6）：以非正常方式访问资源的漏洞 业务缺陷类（5）：业务逻辑不完善导致的漏洞 暴力破解类（4）：与登录或认证凭据相关的暴力破解 客户端攻击类（3）：攻击其他客户端的漏洞 信息泄露类（3）：泄漏敏感的技术类信息及非结构化数据的漏洞 辅助攻击类（1）：单独存在不能形成攻击，但可以增加其他攻击手段效果的漏洞 其中权限获取类、暴力破解类、辅助攻击类影响范围系数 F 固定为 1。
系统重要性系数 I	核心应用系统（4）：主站、网银、商城等盈利站点 普通应用系统（2）：客服、OA 等辅助类站点 边缘应用系统（1）：单一功能、单一接口类站点
漏洞风险值	漏洞风险值= F×Y×I

漏洞风险评定说明

威胁级别	评定说明
严重问题	风险值>18，直接导致系统被入侵或数据被破坏，一旦发生，就是严重的安全事件。建议紧急修复。
中等问题	6 ≤风险值≤18，可能导致重要信息的泄漏、系统拒绝服务或有较高可能导致系统被入侵控制。
轻度问题	2 ≤风险值≤6，可导致敏感信息泄漏或存在轻微安全问题，一般不会产生严重的安全事件。
风险提示	风险值小于 2，不合规编码且利用难度较大的安全问题，一般不会产生严重的安全事件

漏洞可利用性评价说明

可利用性定级	定级标准（如具备如下任一条件即可）
容易	有公开的利用工具或攻击代码
	无需登录系统即可利用
	无需和被攻击用户交互即可利用
	本地修改数据包或客户端程序即可利用
一般	需要和目标用户交互才能够利用
	通过网络远程抓取目标用户数据包才可利用

困难	需要控制被攻击用户的终端设备才可利用
	需要知道被攻击用户的账号口令才可利用