杭银金融App扫描报告

杭州安恒信息技术有限公司

安全得分:43,风险等级:中危

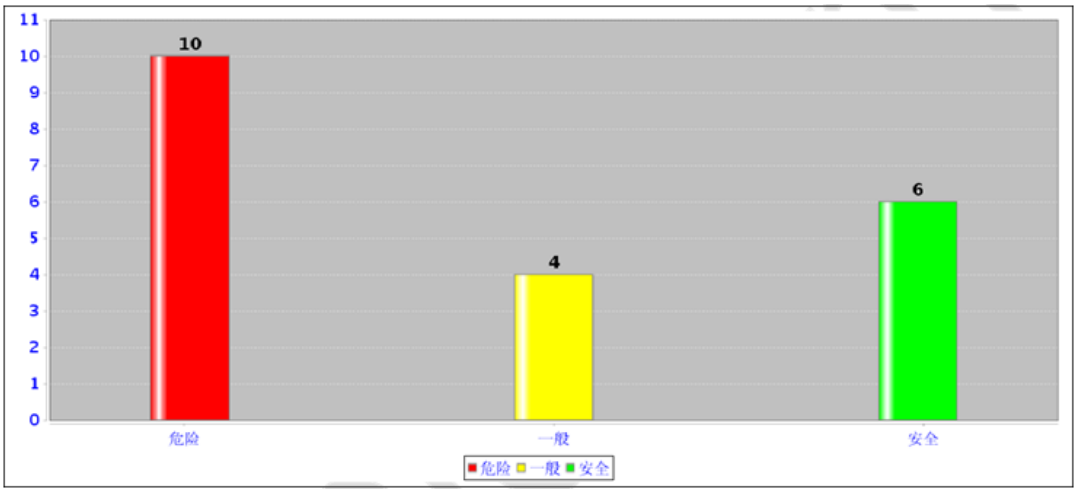
2017-06-19 16:28:35

本报告是App在采用安全检测后得出的报告。由于报告内容属于敏感资料，请报告拥有者在阅读和发布时妥善进行适当的安全控制。

目录

1.应用检测结果预览

通过对应用安全体检后，该应用健康情况见下图：



序列号	风险名称	危险系数	风险描述
1	异常处理	危险	在开发时偶尔会由于疏忽导致有些异常没有进行处理。如果不提示详细信息又会给用户报告异常带来麻烦，不利于开发人员及时发现并处理异常。但是如果将异常详细信息不经处理直接提示给用户则会带来安全隐患。
2	Activity安全	危险	导出的组件可以被第三方app任意调用，导致敏感信息泄露或者恶意攻击者精心构造攻击载荷达到攻击的目的。
3	Receiver安全	危险	可造成信息泄露，拒绝服务攻击等
4	Service安全	危险	Service存在的安全漏洞包括：权限提升，拒绝服务攻击。没有声明任何权限的应用即可在没有任何提示的情况下启动该服务，完成该服务所作操作，对系统安全性产生极大影响。
5	ContentProvider安全	安全	Content Provider的不安全使用会产生sql注入、文件遍历等漏洞，导致用户数据泄

			露
6	Intent安全	危险	intent scheme URLs(意图协议URL)，可以通过解析特定格式的URL直接向系统发送意图，导致自身的未导出的组件可被调用,隐私信息泄露。Intent隐式调用发送的意图可能被第三方劫持，如果含有隐私信息可能导致内部隐私数据泄露
7	WebView安全	危险	利用android的webView组件中的addJavascriptInterface接口函数，可以实现本地 java与js之间交互,但是在安卓4.2以下的系统中，这种方案却给我们的应用带来了很大的安全风险。攻击者如果在页面执行一些非法的JS（诱导用户打开一些钓鱼网站以进 入风险页面），通过远程代码执行，反弹拿到用户手机的shell权限。接下来攻击者就可以在后台默默安装木马，完全控制用户的手机。另外，android webview 组件包含3个隐藏的系统接口：“accessibility”、“accessibilityTraversal”以及“searchBoxJavaBridge_”，同样会造成远程代码执行
8	测试数据移除安全	一般	如果在AndroidManifest.xml配置文件中设置了application属性为debuggable=“true”，则应用可以被任意调试，这就为攻击者调试和破解程序提供了极大方便。如果设置

			application 属性为allowBackup=“true”，则应用在系统没有root的情况下其私有数据也可以通过备份方式进行任意读取和修改，造成隐私泄露和信息被恶意篡改。
9	日志信息移除安全	一般	通过logcat打印的调试信息或者错误异常信息，可以定位应用运行的流程或者关键代码，从而降低黑客破解的难度
10	权限管理	一般	冗余权限可导致串谋攻击，串权限攻击的核心思想是程序A有某个特定的执行权限,程序B没有这个权限。但是B可以利用A的权限来执行需要A权限才能完成的功能。
11	模拟器检测	安全	模拟器具有经济成本低、高度可定制、易于开发、容易部署等优点，攻击者可以通过自己修改定制特定的模拟器来达到监控应用关键函数、获取应用敏感数据，破解应用的目的。为了增加破解阻力，应用中应该进行模拟器检测，添加反模拟器运行代码
12	动态调试	安全	动态调试可以获取函数运行时各个变量的取值，程序运行逻辑，加密的数据信息，从而为破解应用提供便利
13	代码混淆	安全	对代码进行混淆，能够提高黑客阅读和理解代码的门槛，在一定程度上增加了黑客破解的难度
14	Dex保护	危险	Dex为Android应用的核心，保护不当容易被反编译 ，暴露程序重要信息，面临被植入广告、恶意代码 、病毒等风险。

15	SO保护	安全	Android so通过C/C++代码来实现，相对于Java代码来说其反编译难度要大很多，但对于经验丰富的破解者来说，仍然是很容易的事。应用的关键性功能或算法,都会在so中实现，如果so被逆向，应用的关键性代码和算法都将会暴露。
16	资源文件保护	危险	反编译apk获得源码,通过资源文件或者关键字符串的ID定位到关键代码位置，为逆向破解应用程序提供方便
17	数据访问控制	危险	检测数据是否仅被授权用户或应用进程访问。如果开发者使用 openFileOutput(String name,int mode)方法创建内部文件或者使用getSharedPreferences读取配置信息时，如果使用MODE_WORLD_READABLE 或MODE_WORLD_WRITEABLE模式，就会让这个文件变为全局可读或全局可写的。
18	敏感数据加密	安全	本地加密时如果使用SecureRandom中的setSeed方法设置种子将会造成生成的随机数不随机，使加密数据容易被破解。在SecureRandom生成随机数时，如果我们不调用setSeed方法，SecureRandom会从系统中找到一个默认随机源。每次生成随机数时都会从这个随机源中取seed。在linux和Android中这个随机源位于/dev/urandom文件。 如果我们在终端可以运行cat /dev/urandom命令，会观察到

			随机值会不断的打印到屏幕上。在Android 4.2以下，SecureRandom是基于老版的Bouncy Castle实现的。如果生成SecureRandom对象后马上调用setSeed方法。SecureRandom会用用户设置的seed代替默认的随机源。使得每次生成随机数时都是会使用相同的seed作为输入。从而导致生成的随机数是相同的。
19	证书验证	危险	开发者在代码中不检查服务器证书的有效性，或选择接受所有的证书。这种做法可能导致的问题是中间人攻击。
20	远程数据通信加密	一般	如果数据通信没有经过加密，直接使用HTTP协议登录账户或交换数据。例如，攻击者在自己设置的钓鱼网络中配置DNS服务器，将软件要连接的服务器域名解析至攻击者的另一台服务器，在这台服务器就可以获得用户登录信息，或者充当客户端与原服务器的中间人，转发双方数据。另外，程序与服务器通信的敏感数据如果使用了分组加密算法，采用 ECB 模式将会使相同的明文会产生相同的密文，容易受到字典攻击，安全性不够高

2. 评估环境与配置

1)评估应用：杭银金融
2)分析工具：ApkIDE、Jd-Gui、IdaPro、Fiddler2 等。

3.应用详细检测过程及内容

3.1. 应用基本信息

1) 应用程序名称：杭银金融
2) 包名：tomcat360.com.hyxfjr
3) 安装包大小：38.88M
4) 版本：2.0.2

3.2. 应用安全评估

✓业务安全
✓组件安全
✓WebView安全
✓发布规范
✓应用管理
✓应用合规
✓安全增强

3.2.1. 业务安全

✓用户登录
✓密码管理
✓支付安全
✓身份认证
✓超时设置
✓异常处理

3.1.1.1 异常处理

漏洞说明	在开发时偶尔会由于疏忽导致有些异常没有进行处理。如果不提示详细信息又会给用户报告异常带来麻烦，不利于开发人员及时发现并处理异常。但是如果将异常详细信息不经处理直接提示给用户则会带来安全隐患。
分析方案	检测在软件操作异常时是否有异常处理机制，错误提示是否泄漏
分析结果	危险
模拟演示	应用没有在Application中注册“未捕获异常处理器(即Thread.UncaughtExceptionHandler,是用来处理未捕获异常的)”，进行全局异常的处理。
修复建议	建议在application中设置全局异常捕获，在异常发生时立即记录异常，进行加密处理后通过直接发送邮件或者其它方式将异常情况报告给开发人员，以在最短的时间内发现并处理异常。

3.2.2. 组件安全

✓ Activity安全

- ✓ Broadcast Receiver 安全
- ✓ Service安全
- ✓ Content Provider安全
- ✓ Intent安全

3.2.2.1.Activity 安全

漏洞说明	导出的组件可以被第三方app任意调用，导致敏感信息泄露或者恶意攻击者精心构造攻击载荷达到攻击的目的。
分析方案	检测Activity是否会被 权限攻击检测Activity 是否会被劫持
分析结果	危险
模拟演示	该应用中存在可以任意导出的Activity组件，可以导致Activity被劫持、绕过本地认证，造成拒绝服务，导致程序崩溃等。可以导出的Activity组件如下：
修复建议	建议如果组件不需要与其它app共享数据或交互，请在AndroidManifest.xml配置文件中将该组件设置为exported = “False”。如果组件需要与其它app共享数据或交互，请对组件进行权限控制和参数校验。

3.2.2.2.Broadcast Receiver 安全

漏洞说明	可造成信息泄露，拒绝服务攻击等
分析方案	检测Broadcast Receiver是否会被监听、劫持
分析结果	危险
模拟演示	该应用广播接收器默认设置exported='true'，导致应用可能接收到第三方恶意应用伪造的广播，利用这一漏洞，攻击者可以在用户手机通知栏上推送任意消息，通过配合其它漏洞盗取本地隐私文件和执行任意代码。可以导出的Broadcast Receiver如下： <code>com.xiaomi.push.service.receivers.NetworkStatusReceiver tomcat360.com.hyxfjr.recivice.DemoMessageReceiver</code>

修复建议	建议如果组件不需要与其它app共享数据或交互，请在AndroidManifest.xml 配置文件中将该组件设置为exported = “False”。如果组件需要与其它app共享数据或交互，请对组件进行权限控制和参数校验。

3.2.2.3.Service 安全

漏洞说明	Service存在的安全漏洞包括：权限提升，拒绝服务攻击。没有声明任何权限的应用即可在没有任何提示的情况下启动该服务，完成该服务所作操作，对系统安全性产生极大影响。
分析方案	检测Service是否会被权限攻击
分析结果	危险
模拟演示	设置exported属性为true，导致service组件暴露，第三方程序在没有声明任何权限的情况下，通过发送恶意intent，可以在没有任何提示情况下启动该服务，进行该服务所作的操作，对
修复建议	建议如果组件不需要与其它app共享数据或交互，请在AndroidManifest.xml 配置文件中将该组件设置为exported = “False”。如果组件需要与其它app共享数据或交互，请对组件进行权限控制和参数校验。

3.2.2.4.ContentProvider 安全

漏洞说明	Content Provider的不安全使用会产生sql注入、文件遍历等漏洞，导致用户数据泄露
分析方案	检测Content Provider是否会被权限攻击
分析结果	安全
模拟演示	该应用中不存在可以任意导出的Content Provider组件。
修复建议	

3.2.2.5.Intent 安全

漏洞说明	intent scheme URLs(意图协议URL)，可以通过解析特定格式的URL直接向系统发送意图，导致自身的未导出的组件可被调用，隐私信息泄露。Intent隐式调用发送的意图可能被第三方劫
分析方案	intent scheme URLs(意图协议URL)，可以通过解析特定格式的URL直接向系统发送意图，导致自身的未导出的组件可被调用，隐私信息泄露Intent隐式调用发送的意图可能被第三方劫持，
分析结果	危险

模拟演示	android.util.Log.e("baidumapsdk", "bind service

	failed\uff0ccall openapi")
	;
	com.baidu.mapapi.utils.a.a(p5, p4);
	} else {
	com.baidu.mapapi.utils.a.v = new Thread(new
	com.baidu.mapapi.utils.e(p4, p
	5));
	com.baidu.mapapi.utils.a.v.setDaemon
	(1); com.baidu.mapapi.utils.a.v.start();
	}
	}
	return;
	}
	</code>

修复建议	建议将隐式调用改为显式调用。

3.2.2.6.WebView 安全

漏洞说明	利用android的webView组件中的addJavascriptInterface接口函数，可以实现本地java与js之间交互，但是在安卓4.2以下的系统中，这种方案却给我们的应用带来了很大的安全风险。攻击者如果在页面执行一些非法的JS（诱导用户打开一些钓鱼网站以进 入风险页面），通过远程代码执行，反弹拿到用户手机的shell权限。接下来攻击者就可以在后台默默安装木马，完全控制用户的手机。另外，android webview 组件包含3个隐藏的系统接口：“accessibility”、 “accessibilityTraversal”以及“searchBoxJavaBridge_”，同样会造成远程代码执行
分析方案	检测是否利用webView组件中的addJavascriptInterface接口函数，是否移除 an- droid webview组件所包含的3个隐藏的系统接口：“accessibility”、 “accessibilityTraversal”以及“searchBoxJavaBridge_”。
分析结果	危险

模拟演示	android的webview组件包含3个隐藏的系统接口：“accessibility”、
	“ccessibilityaversal”以及“searchBoxJavaBridge_”，在使用webview加载网页时，如果没有使用removeJavascriptInterface方法移除掉这三个组件 ，同样会造成远程代码执行。
	<code>类名: Lcom/alipay/sdk/auth/c;
	public final void run()
	{
	try {
	</code>

	com.alipay.sdk.auth.AuthActivity.h(this.b).loadUrl(new String-Builder("javascri
	pt:").append(this.a).toString());
	} catch (Exception v0) {
	}
	return;
	}
	类 名 :
	Ltomcat360/com/hyxfjr/v/view_impl/activity/HtmlPostActivity$MyWebViewClie nt;
	public boolean shouldOverrideUrlLoading(android.webkit.WebView p3, String p4)
	{
	int v0_0 = 0;
	if (util.s.b(p4)) {
	if (!p4.contains("/center")) {
	if (!p4.contains("/close"))
	{ this.this$0.webview.loadUrl(p4); v0_0 = 1;
	} else {
	this.this$0.finish();
	}
	} else {
	this.this$0.finish();
	}
	}
	return v0_0;
	}
	</code>

修复建议	（1）建议开发者不要使用addJavascriptInterface，使用注入javascript和第三方协议的替代方案，具体实现代码可参考：https://github.com/pedant/safe-java-js-webview-bridge。（2）使用removeJavascriptInterface方法移除掉“accessibility”、“accessibilityTraversal”以及“searchBoxJavaBridge_”这3个隐藏的webview组件接口。

3.2.3. 发布规范

✔测试数据移除安全
✔日志信息移除安全

3.2.3.1.测试数据移除安全

漏洞说明	如果在AndroidManifest.xml配置文件中设置了application属性为debuggable=“true”，则应用可以被任意调试，这就为攻击者调试和破解程序提供了极大方便。如果设置application属性为allowBackup=“true”，则应用在系统没
分析方案	检测发布应用的主配文件AndroidManifest.xml文件中是否包含不应该包含的测试代码
分析结果	一般
模拟演示	设置application属性为allowBackup=“true”，则应用在系统没有root的情况下其私有数据也可以通过备份方式进行任意读取和修改，造成隐私泄露和信息被恶意篡改。 <code>AndroidManifest.xml文件中
修复建议	建议应用发布时在AndroidManifest.xml配置文件中设置application属性为 debuggable=“false”，allowBackup=“false”，同时在代码中加入判断application属性debuggable=“false”是否被修改的代码。

3.2.3.2.日志信息移除安全

漏洞说明	通过logcat打印的调试信息或者错误异常信息，可以定位应用运行的流程或者关键代码，从而降低黑客破解的难度
分析方案	检测发布应用中是否包含不应该包含的日志信息
分析结果	一般

模拟演示	{
	com.xiaomi.a.a.c.c.c(new StringBuilder().append("[Slim] ").append(com.xiaomi.b.a.a .a(this.a).format(new java.util.Date())).append(" Reconnection failed due to an exception (").append(com.xiaomi.b.a.a.b(this.a).hashCode()).append(")").toString()) ;
	p5.printStackTrace(); return;
	}
	类名: Lcom/baidu/location/d/j;
	public static boolean b(String p3, String p4, String p5)
	{
	try {
	int v0_2 = ja- va.security.KeyFactory.getInstance("RSA").generatePublic(new java .security.spec.X509EncodedKeySpec(com.baidu.android.bbalbs.common.a.b.a(p 5.getBytes())));
	java.security.Signature v1_3 = ja- va.security.Signature.getInstance("SHA1WithRS A");
	v1_3.initVerify(v0_2);
	v1_3.update(p3.getBytes()); int v0_6 = v1_3.verify(com.baidu.android.bbalbs.common.a.b.a(p4.getBytes()));
	} catch (int v0_7)
	{ v0_7.printStackTrace(); v0_6 = 0;
	}
	return v0_6;
	}
	</code>

修复建议	开发者在开发应用过程中应该注意防范信息泄露。在应用发布时要注意删除logcat输出或者打印的诸如用户名，密码，token，cookie和imei等隐私、敏感信息。

3.2.4. 安全增强

✔权限管理
✔输入检查
✔键盘记录
✔界面劫持
✔模拟器检测
✔进程保护

动态检测

第三方SDK安全

3.2.4.1.权限管理

漏洞说明	冗余权限可导致串谋攻击，串权限攻击的核心思想是程序A有某个特定的执行权限
分析方案	是否存在权限滥用问题
分析结果	一般
模拟演示	通过对Java层代码进行检测发现该应用存在以下冗余权限，即在本应用中并不需要但是却在Androidmanifest.xml文件中申请了这些权限。请开发人员仔细确认在native层是否需要用到这些权限。冗余权限可以被一些恶意程序所利用来进行一些恶意行为，给用户造成损失。 <code>android.permission.WRITE_EXTERNAL _STORAGE an- droid.permission.SYSTEM_ALERT_WINDOW android.permission.RECEIVE_SMS an- droid.permission.READ_SMS an- droid.permission.READ_EXTERNAL_STORAGE android.permission.READ_LOGS an- droid.permission.SET_DEBUG_APP an- droid.permission.USE_CREDENTIALS an- droid.permission.MANAGE_ACCOUNTS an- droid.permission.MOUNT_FORMAT_FILESYSTE MS com.android.launcher.permission.READ_SE TTINGS android.permission.WRITE_SETTINGS
修复建议	建议去掉多余权限。

3.2.4.2.模拟器检测

漏洞说明	模拟器具有经济成本低、高度可定制、易于开发、容易部署等优点，攻击者可以通过自己修改定制特定的模拟器来达到监控应用关键函数、获取应用敏感数据，破解应用的目的。为了增
分析方案	检测应用是否可以运行在模拟器环境中
分析结果	安全
模拟演示	该应用进行了模拟器检测。
修复建议	

3.2.4.3.动态调试

漏洞说明	动态调试可以获取函数运行时各个变量的取值，程序运行逻辑，加密的数据信息
分析方案	检测程序是否具备抵抗动态调试的能力（Java层和Native层）
分析结果	安全
模拟演示	apk已反调试
修复建议	

3.3. 源码安全评估

程序完整性

程序机密性

3.3.1.1.代码混淆

漏洞说明	对代码进行混淆，能够提高黑客阅读和理解代码的门槛，在一定程度上增加了黑客破解的难度
分析方案	检测程序是否进行代码混淆
分析结果	安全
模拟演示	该应用代码已经被混淆。
修复建议	

3.3.1.2.Dex 保护

漏洞说明	Dex为Android应用的核心，保护不当容易被反编译，暴露程序重要信息，面临被植入广告、恶意代码、病毒等风险
分析方案	检测程序可执行文件Dex是否做加密处理
分析结果	危险
模拟演示	检测中发现程序完成后没有对dex进行隐藏加密，可以获取smali代码，包括应用分析入口Launcher Activity的信息；
修复建议	建议对dex文件进行加密保护，防止被dex2jar等工具反编译。

3.3.1.3.SO 保护

漏洞说明	Android so通过C/C++代码来实现，相对于Java代码来说其反编译难度要大很多，但对于经验丰富的破解者来说，仍然是很容
分析方案	检测程序的本地库文件是否做加密处理
分析结果	安全
模拟演示	so已经进行了加密处理。
修复建议	

3.3.1.4.资源文件保护

漏洞说明	反编译apk获得源码，通过资源文件或者关键字符串的ID定位到关键代码位置，为逆向破解应用程序提供方便
分析方案	检测程序的资源文件是否做加密处理 检测程序的资源文件、字符串是否做加密处理
分析结果	危险

3.4. 数据安全评估

数据输入
数据存储
数据传输
数据输出

3.4.1.1.数据访问控制

漏洞说明	检测数据是否仅被授权用户或应用进程访问。如果开发者使用openFileOutput(String name,int mode)方法创建内部文件或者使用getSharedPreferences读取配置信息时，

分析方案	检测数据是否仅被授权用户或应用进程访问。如果开发者使用openFileOutput(String name,int mode)方法创建内部文件时，将第二个参数设置为Context.MODE_WORLD_READABLE或 Context.MODE_WORLD_WRITEABLE,就会让这个文件变为全局可读或全局可写的。
分析结果	危险

模拟演示	使用openFileOutput(String name,int mode)方法创建内部文件或者使用getSharedPreferences读取配置信息时，如果使用MODE_WORLD_READABLE或MODE_WORLD_WRITEABLE模式，就会让这个文件变为全局可读或全局可写的，可以被其它程序任意读取和修改，从而造成信息泄露和恶意篡改。
	<code>类名: Lcom/xiaomi/a/a/d/f;
	private f(android.content.Context p3)
	{
	this.b = new java.util.concurrent.ScheduledThreadPoolExecutor(1); this.c = new android.util.SparseArray();

修复建议	建议 禁止全局文件可读写。 如果开发者要跨应用共享数据，一种较好的方法是实现一个Content Provider组件，提供数据

3.4.1.2.敏感数据加密

漏洞说明	本地加密时如果使用SecureRandom中的setSeed方法设置种子将会造成生成的随机数不随机，使加密数据容易被破解。在SecureRandom生成随机数时，如果我们不调用setSeed方法，SecureRandom会从系统中找到一个默认随机源。每次生成随机数时都会从这个随机源中取seed。在linux和Android中这个随机源位于/dev/urandom文件 。 如果我们在终端可以运行cat /dev/urandom命令，会观察到
分析方案	检测对敏感数据加密，在使用SecureRandom时如果使用setSeed方法设置种子将会造成生成的随机数不随机
分析结果	安全

模拟演示	该应用不存在该漏洞。
修复建议	

3.4.1.3.证书验证

漏洞说明	开发者在代码中不检查服务器证书的有效性，或选择接受所有的证书。这种做法可能导致的问题是中间人攻击
分析方案	检测程序与服务器的通信验证是否有证书，是否有证书合法性和
分析结果	危险
模拟演示	应用没有严格对服务端和客户端证书进行校验，或者信任了所有主机，覆盖了Google的证书认证机制，对于异常事件的处理直接return null，攻击者可以通过设置DNS服务器使客户端与指定的服务器进行通信。攻击者在服务器上部署另一个证书，在会话建立阶段，客户端会收到这张证书，如果客户端忽略这个证书上的异常，或者接受这个证书，就会成功建立会话、开始加密通信。但攻击者拥有私钥，因此可以解密得到客户端发来数据的明文。攻击者还可以模拟客户端，与真正的服务器联系，充当中间人做监听。 <code>类名：Lcom/lzy/a/g/a$1; public void checkServerTrust-
修复建议	建议严格对服务端和客户端证书进行校验，禁止信任所有证书，对于异常事件的处理不应该直接return null。

3.4.1.4.远程数据通信加密

漏洞说明	如果数据通信没有经过加密，直接使用HTTP协议登录账户或交换数据。例如，攻击者在自己设置的钓鱼网络中配置DNS服务器，将软件要连接的服务器域名解析至攻击者的另一台服务器，在这台服务器就可以获得用户登录信息，或者充当客户端与原服务器的中间人，转发双方数据。另外，程序与服务器通信的敏
分析方案	检测程序与服务器通信的敏感数据是否使用分组加密算法，如果使用了分组加密算法，则检测是否使用了ECB模式
分析结果	一般

模拟演示	应用虽然使用了强加密算法加密，但是这个算法使用了ECB模式，这种模式下加密会使相同的明文产生相同的密文，容易遭到字典攻击，安全性不够高。漏洞代码如下 : <code>类名: Lcom/baidu/android/bbalbs/common/util/b; private static byte[] a(byte[] p2, ja- va.security.PublicKey p3) { byte[] v0_1 = ja- vax.crypto.Cipher.getInstance("RSA/ECB/PKC
修复建议	建议使用CBC模式进行初始化，CBC模式明文被加密前要与前面的密文进行异或运算后再加密，因此只要选择不同的初始向量