移动金融应用安全风险及解决方案

内容

- 安全评估
  - 由具备高技能和高素质的安全服务人员发起、并模拟常见黑客所使用的攻击手段对目标系统进行模拟入侵
  - 目的在于充分挖掘和暴露系统的弱点，从而让管理人员了解其系统所面临的威胁。
  - 渗透测试工作往往作为风险评估的一个重要环节，为风险评估提供重要的原始参考数据

移动金融应用的安全评估 - 方法及标准

- 基于数据生命周期的安全测试，对手机银行客户端的程序、数据、通信、业务、系统环境等进行全面安全测试，检测数据的输入、处理、输出以及数据运行时的系统环境的安全性

- 手机银行客户安全测试标准依据中国人民银行发布的中国金融移动支付系列技术标准中的
  - 《中国金融移动支付 检测规范 第3部分:客户端软件》

移动金融应用的安全评估 - 工具

- 反编译及重打包
  - dex2jar、baksmali、IDA pro、apktool
- 调试工具
  - gdb 等
- 代码注入工具
  - Xposed、Substrate 等
- 网络
  - Burp suite等

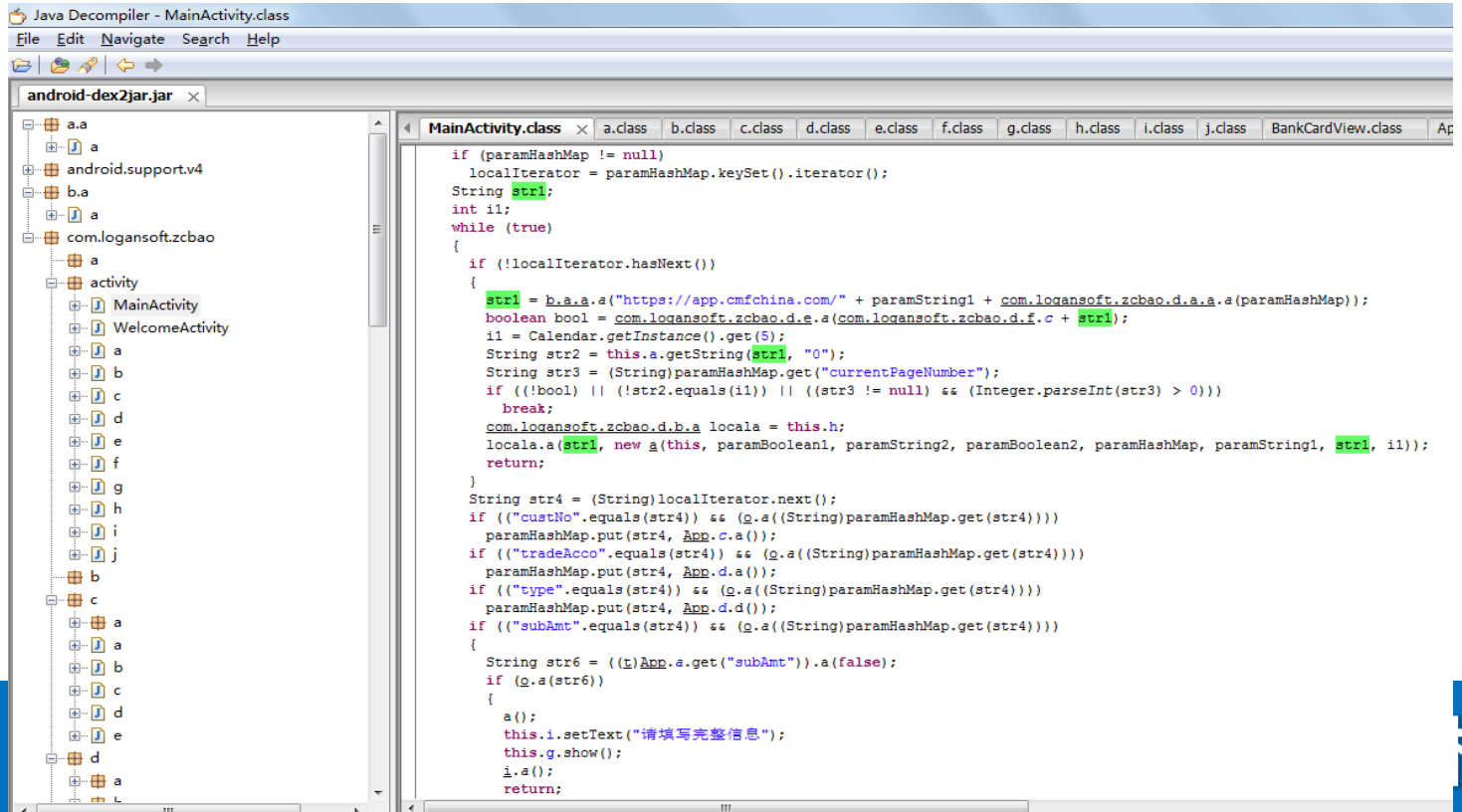
移动金融应用的安全评估 - 测试内容

采用黑盒渗透攻击和白盒代码审计的方式发现移动应用的安全缺陷及安全漏洞

评估内容	描述
程序安全	安装与卸载、人机交互、登陆检测、发布规范、第三方SDK安全等方面
代码安全	是否具有防逆向、防动态注入、防篡改等能力
数据安全	应用的数据录入、数据访问、数据存储、数据传输、数据显示是否存在安全风险
组件安全	移动应用暴露的组件是否可以被恶意攻击
通信安全	检查客户端软件和服务器间的通信协议是否安全，能否被攻击
业务安全	移动应用的核心业务是否存在安全缺陷。例如银行客户端，针对转账的过程应进行安全性检测，检测是否有可能进行转账的篡改
系统安全	移动应用的运行环境是否安全

反编译测试

将二进制程序转换成人们易读的一种描述语言的形式，是逆向工程中的常见手段。**反编译的结果是易读的代码，这样就暴露了客户端的所有逻辑，比如与服务端的通讯方式，加解密算法、密钥，转账业务流程、软键盘技术实现等等。**



测试结果：代码没有做保护，左图是主界面转账与存入的代码片段截图

重打包测试

对客户端程序添加或修改代码，修改客户端资源图片，配置信息，图标等，再生成新的客户端程序，实现应用钓鱼。对金融客户端，可能添加病毒代码、广告SDK，推广自己的产品；添加恶意代码窃取登录账号密码、支付密码、拦截验证码短信，修改转账目标账号、金额等等。

droid
你的App

动态调试测试

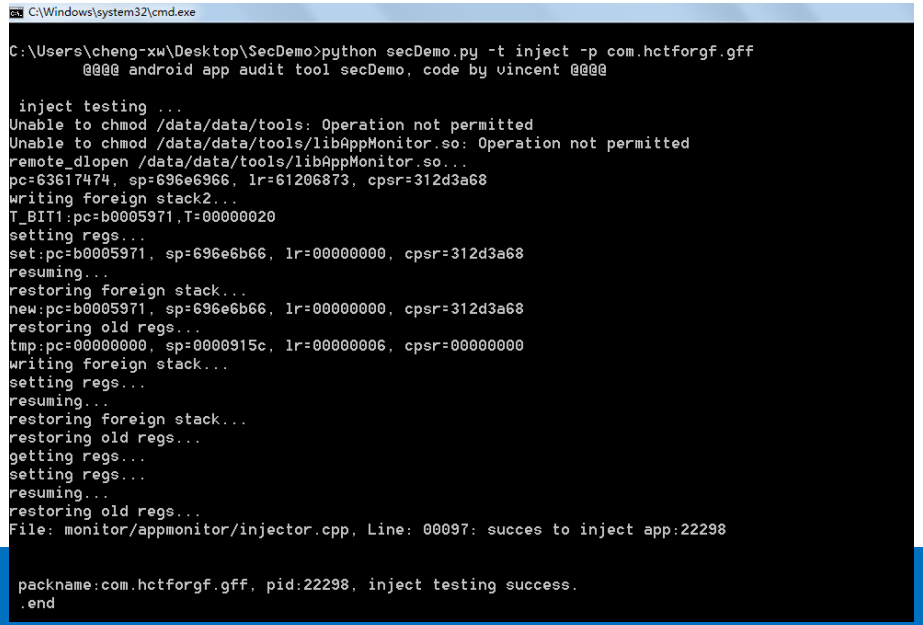
指攻击者利用调试器跟踪目标程序运行，查看、修改内存代码和数据，分析程序逻辑，进行攻击和破解等行为。对于金融行业客户端，该风险可修改客户端业务操作时的数据，比如账号、金额等。

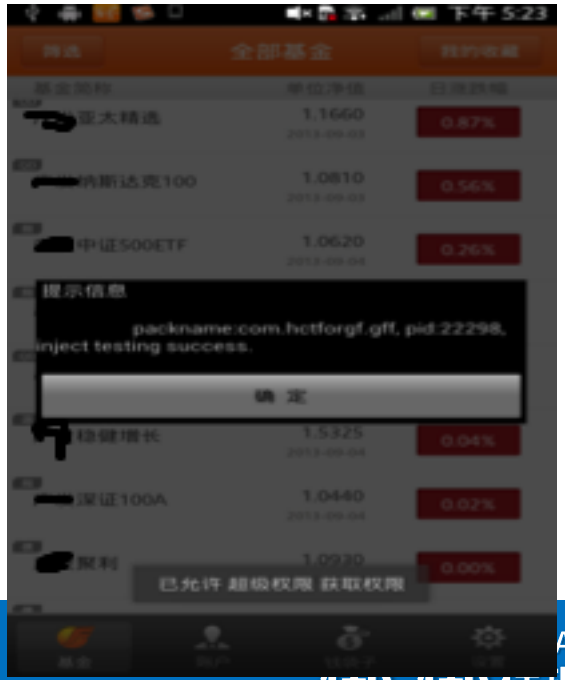
```
C:\[redacted]\Desktop\SecDemo>python secDemo.py -t debug -p com.logansoft.zcbao
        @@@@ android app audit tool secDemo, code by vincent @@@@

 debug testing ...
adb shell su -c "/data/data/tools/gdb -pid 31630" > /data/local/tmp/gdblogtmp.txt
GNU gdb 6.7
Copyright (C) 2007 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=arm-none-linux-gnueabi --target=".
Attaching to process 31630
Reading symbols from /system/bin/app_process...(no debugging symbols found)...done.
(no debugging symbols found)
0x4027db10 in _start () from /system/bin/linker
(gdb) Hangup detected on fd 0
Error detected on fd 0
error detected on stdin
The program is running.  Quit anyway (and detach it)? (y or n) EOF [assumed Y]
Detaching from program: /system/bin/app_process, process 31630

 packname:com.logansoft.zcbao, pid:31630, debug testing success.
 .end
```

测试情况说明：进行可被调试，获取、修改内存数据等，挂载进程，查看、修改内存数据。

代码注入测试

通过OS特定技术，将代码写入到目标进程并让其执行的技术。攻击者可以将一段恶意代码写到目标进程，这段代码可以加载其它可执行程序，进而实施hook，监控程序运行行为、获取敏感信息等。**对于金融客户端，可通过代码注入技术，将恶意代码注入到客户端中，窃取输入的登录账号、密码、支付密码，修改转账的目标账号、金额，窃取通讯数据等。**





由安全评估得出的移动金融安全现状观点：

1.根据我们的安全实验室统计测试，国内大多数移动金融App这几项安全测试都是未通过的，移动安全机制存在缺失

2. 未通过这些安全测试，意味着移动金融客户端可能面临下面业务风险

内容

移动金融应用的安全风险

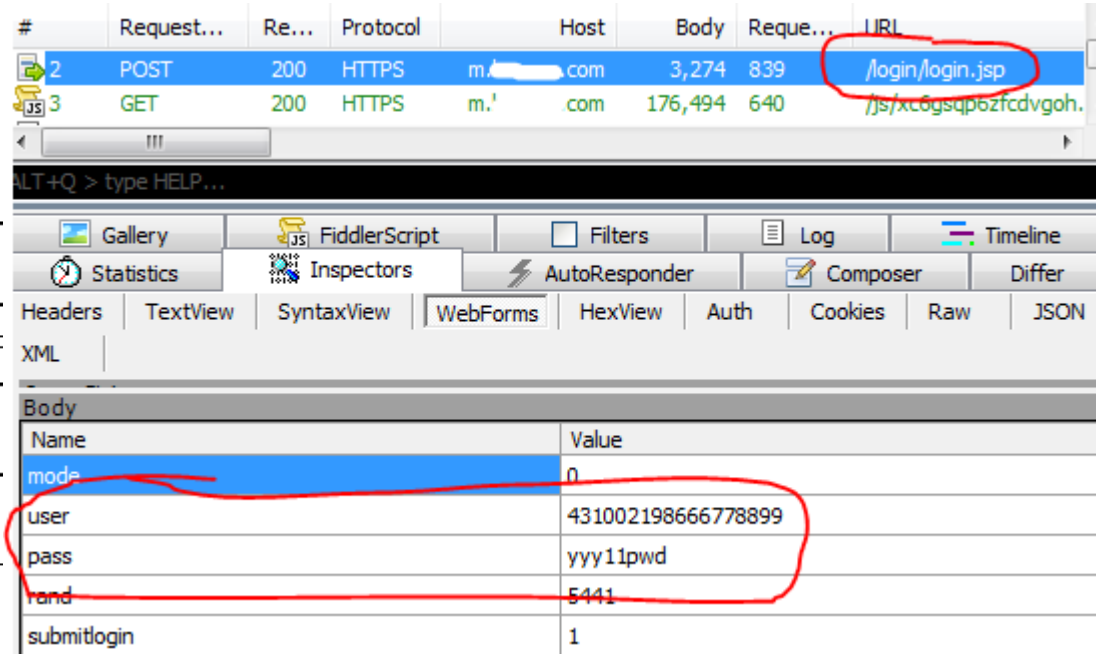
- 用户信息安全风险
- 交易安全风险
- 二次打包风险
- 暴露移动端和服务端漏洞风险

用户信息安全风险 - 例举1

- 软键盘 SDK 内存明文风险

- 漏洞概述

漏洞描述	键盘输入信息在内存为明码
利用场景	通过内存关键点注入，在内存中
可利用性	容易
风险等级	高



- 漏洞验证

可以直接取得用户输入密钥

用户信息安全风险 - 例举2

- 本地存储未加密泄漏敏感信息

Table: cookies

New Record | Delete Record

	creation_utc	host_key	name	value
1	39500544952518	ebank.[redacted]com.cn	BIGipServerWangYing_pool_8080	402854572.36895.0000
2	39500544952999	ebank.[redacted]com.cn	JSESSIONID	QLGcTmrb23fJ34mgGdLrhn7jDYl76MQBJyblV9

- 漏洞概述

漏洞描述	客户端的本地存储都没有做加密处理，例如shared_prefs目录下的配置文件，以及databases下面的数据库文件，可被偷取敏感信息，如配置信息，用户名，cookie会话信息（该cookie由WebView组件生成）
利用场景	如果客户端所在设备中了木马，或用户手机被恶意攻击者窥看，便能轻易获取明文暴露的客户端敏感信息，造成用
可利用性	中
风险等级	高

LOGIN_INFOS.xml[redacted].id.mobilebank\shared_prefs) - GVIM

文件(F) 编辑(E) 工具(T) 语法(S) 缓冲区(B) 窗口(W) 帮助(H)

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
<boolean name="IS_SAVE_ACC" value="true" />
<string name="UserNos">hsj[redacted]</string>
</map>
```

- 问题代码分析

问题产生在客户端，cookie 信息由 WebView 组件生成

用户信息安全风险

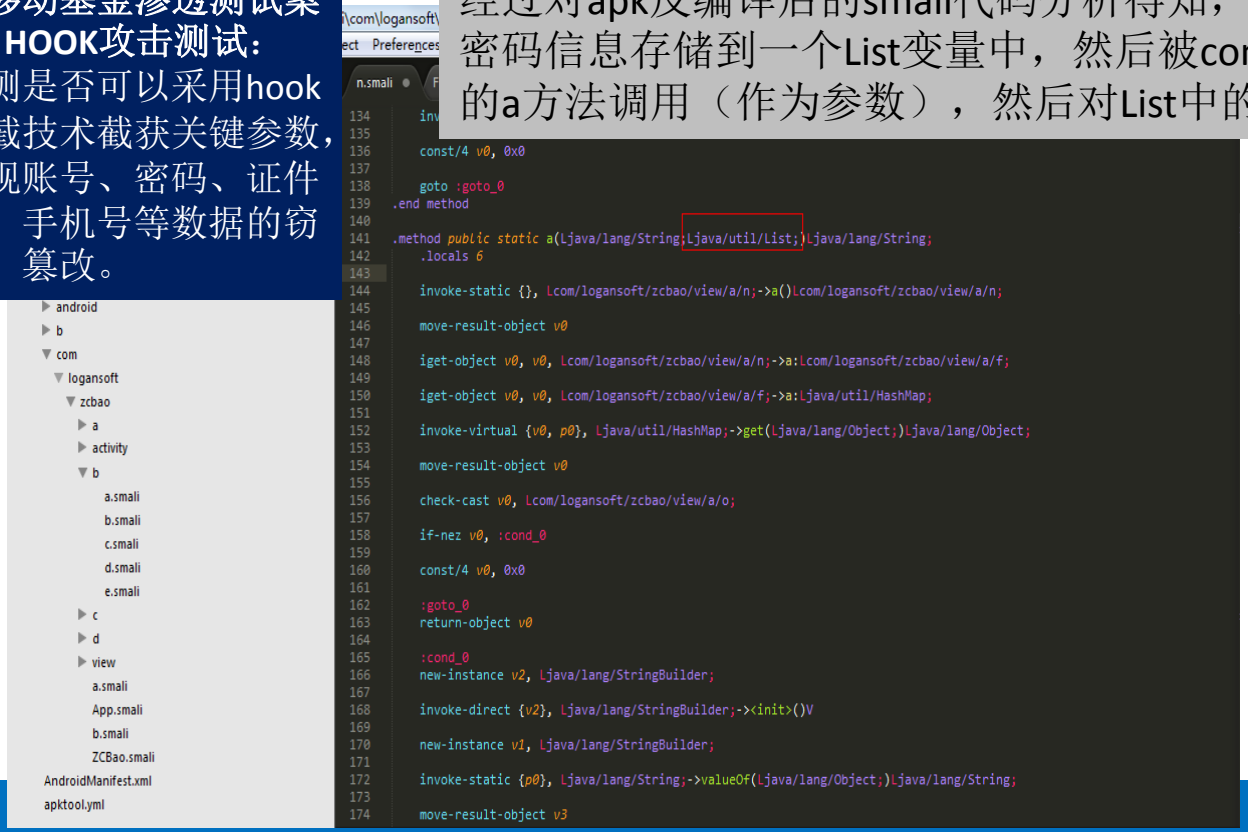
还有以下多种攻击方式可能导致用户信息泄露

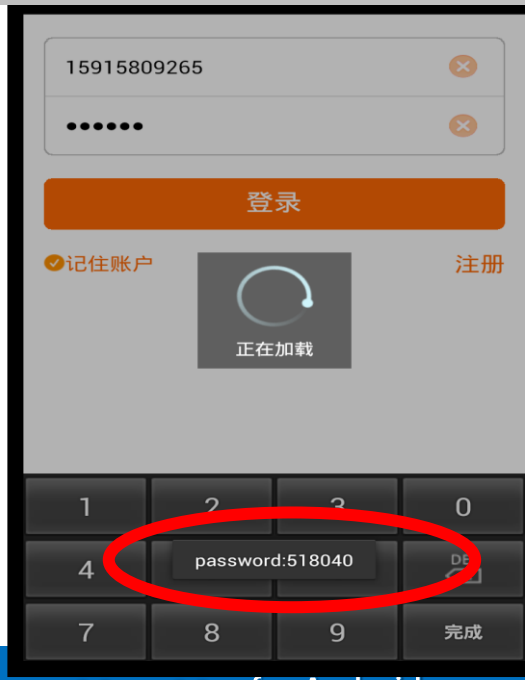
	攻击方式
攻击方式1	界面劫持攻击
攻击方式2	篡改/二次打包
攻击方式3	键盘劫持
攻击方式4	键盘录屏记录
攻击方式5	Hook攻击测试
攻击方式6	账号密码、Token本地存储
攻击方式7	中间人攻击

交易安全风险 - 例举1

某移动基金渗透测试案例- HOOK攻击测试：
检测是否可以采用hook拦截技术截获关键参数，实现账号、密码、证件号、手机号等数据的窃取、篡改。

经过对apk反编译后的smali代码分析得知，账户登录时通过软键盘键入的密码信息存储到一个List变量中，然后被com.logansoft.zcbao.view.a.n类中的a方法调用（作为参数），然后对List中的密码信息进行变换和加密。





交易安全风险 - 例举2

某手机银行动态篡改攻击-POC

通过劫持网络传输函数，在数据加密前动态修改提交到服务器的交易请求：

- 修改收款人账号
- 修改收款人姓名
- 修改转账金额

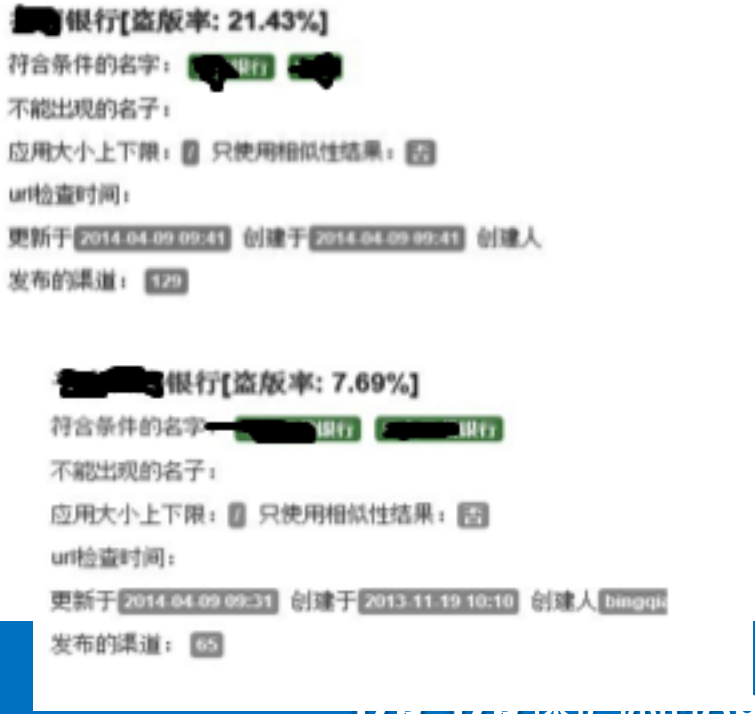
交易安全风险

以下诸多环节都可能存在交易安全攻击风险

	可能的风险环节点
1	用户开户签约、注册、登录、资料变更等流程安全控制机制
2	业务交易流程安全控制机制
3	支付流程安全控制机制
4	业务交易监控和处置机制
5	其他重要流程安全控制

二次打包风险

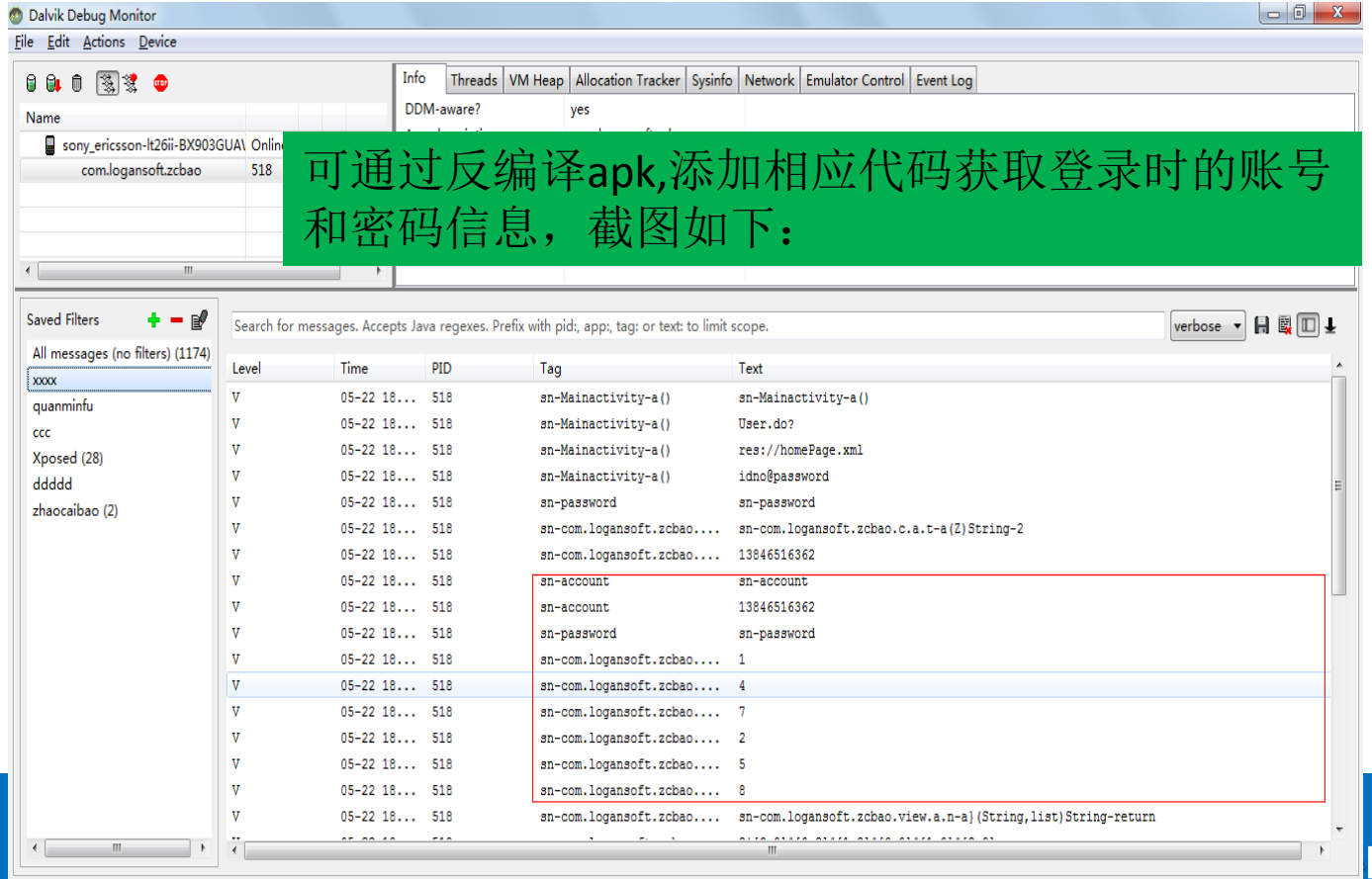
根据梆梆安全大数据监控系统统计，许多知名移动金融App都遭遇过篡改和病毒二次打包



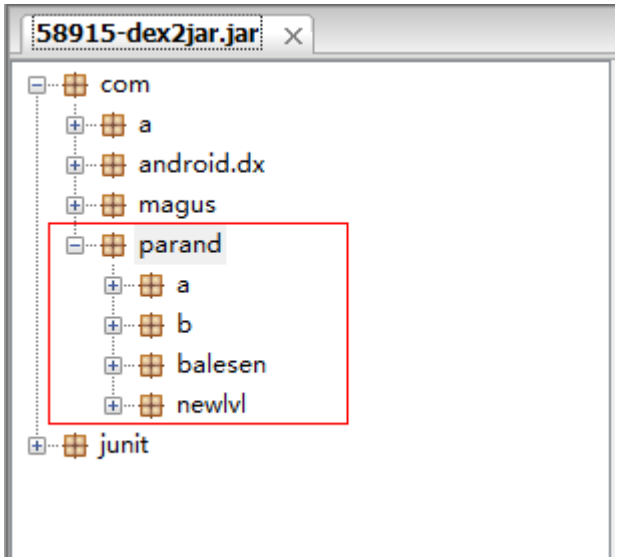
二次打包风险 - 例举1

某移动基金渗透测试案例-篡改/二次打包测试：

对客户端程序添加或修改代码，修改客户端资源图片，配置信息，图标等，再生成新的客户端程序，实现应用钓鱼。对金融客户端，可能添加病毒代码、广告SDk，推广自己的产品；添加恶意代码窃取登录账号密码、支付密码、拦截验证码短信，修改转账目标账号、金额等等。



二次打包风险 - 例举2



如上图所示，在源代码基础上加入了parand包，该包中包含了恶意操作的代码

客户端代码缺陷和逻辑漏洞暴露 - 例举1

通过反编译逆向分析，可以发现移动金融客户端的很多代码缺陷和逻辑漏洞。

漏洞概要

缺陷编号： **WooYun-2013-41332**

漏洞标题： 民生银行android客户端可查询修改他人账号各种信息

相关厂商： **中国民生银行**

漏洞作者： **lupin**

提交时间： 2013-10-29 10:02

漏洞类型： 非授权访问/认证绕过

危害等级： 高

漏洞状态： 已交由第三方厂商(cncert国家互联网应急中心)处理

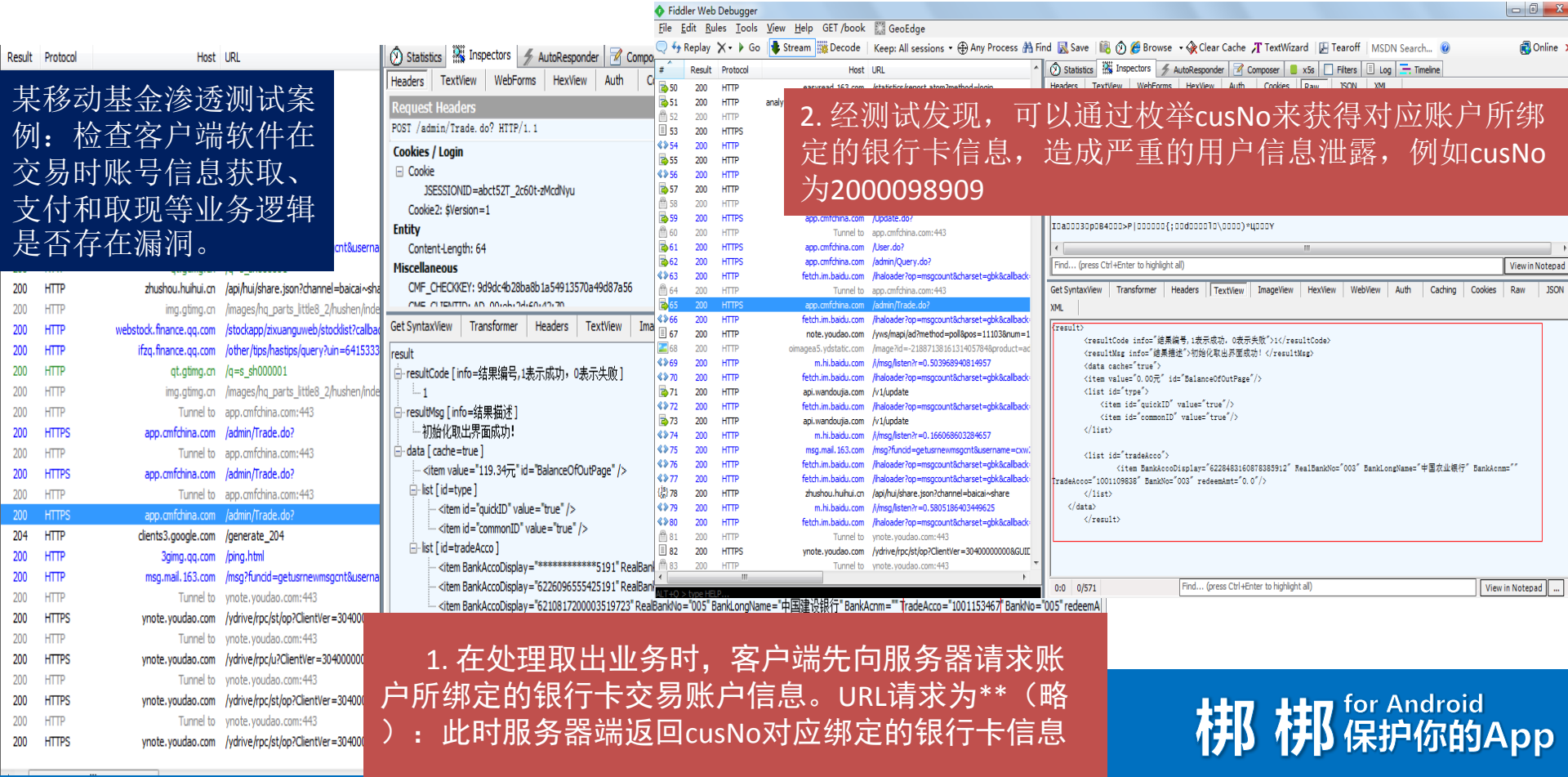
漏洞来源： **http://www.wooyun.org**

Tags标签： 无

分享漏洞： 分享到 0

客户端代码缺陷和逻辑漏洞暴露 - 例举2

某移动基金渗透测试案例：检查客户端软件在交易时账号信息获取、支付和取现等业务逻辑是否存在漏洞。



2. 经测试发现，可以通过枚举cusNo来获得对应账户所绑定的银行卡信息，造成严重的用户信息泄露，例如cusNo为2000098909

1. 在处理取出业务时，客户端先向服务器请求账户所绑定的银行卡交易账户信息。URL请求为**（略）：此时服务器端返回cusNo对应绑定的银行卡信息

客户端代码缺陷和逻辑漏洞暴露 - 例举3

5.1.2.1 漏洞概述

漏洞描述	手机银行系统采用动态图片验证码防范客户使用机器人程序不断尝试获取用户密码。它在图片中加入肉眼可以识别而计算机不能自动识别的随机数字或符号，从而达到防范黑客攻击的目的。但是开发人员在实现图像验证码过程中存在一个逻辑缺陷，产生了一个万能验证码hccb，无论系统显示给客户的是什么图像验证码，只要用户输入hccb就可以绕过验证。
利用场景	黑客通过逆向分析客户端程序，可以发现该万能密码hccb，进而可以固定图形验证码，使用自动化的程序批量破解手机银行的用户密码。虽然密码错误达到一定次数后会触发账号锁定策略，黑客可以选择固定密码批量跑账号的方式，而账号就是手机号，格式较为固定，很容易进行自动化的暴力破解攻击。
可利用性	容易。
风险等级	高风险，可造成大量账号密码被破解。

5.1.2.2 缺陷代码分析

1. 服务端相关代码

·5.1.7 转账操作的动态密码发往客户端

·5.1.7.1 漏洞概述

漏洞描述	当用户要转账操作时，会发送短信动态密码到开户时设定的手机，此动态密码用于确认是客户本人在尝试绑定操作，但是在目前设计中，动态密码不但发送到了手机短信，还发送了一份拷贝到客户端，并在本地做动态密码比对，使动态密码在一定程度上没起到应有的作用。
利用场景	转账操作时，通过截获网络流量，或hook关键API，可以接收到发送到手机短信的动态密码
可利用性	中
风险等级	高

内容

移动金融客户端的解决方案

- 安全开发规范及安全评估
  - 开发者应遵循移动应用的安全开发规范，进行移动金融应用客户端的开发
  - 使用一些成熟的安全组件，如软键盘SDK、清场等
  - 定期对客户端进行安全评估
- 安全加固
  - 发布前加固应用，保证代码安全
- 上线后的渠道监控
  - 监控第三方应用市场，及时发现各种盗版、钓鱼、山寨等恶意应用

安全加固技术

代码加密

阻止代码被反编译

反调试

阻止APP运行时被动态注入

完整性校验

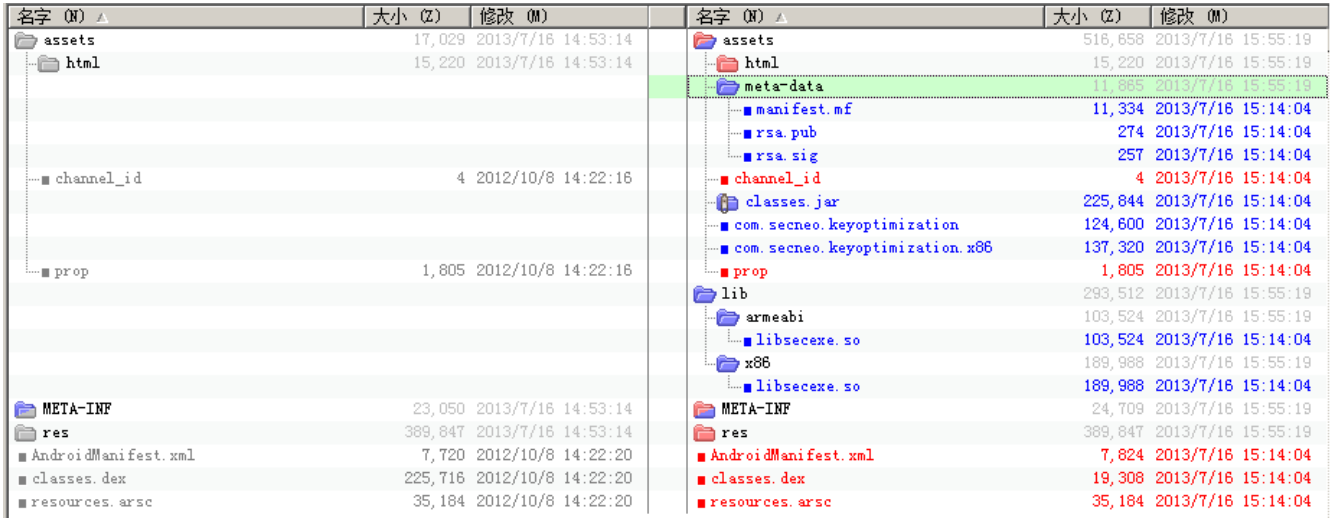
阻止APP被重新打包

安全加固技术 - 第一代加固技术

第一代加固技术基于类加载的技术

- classes.dex被完整加密，放到APK的资源中
- 运行时修改程序入口，将classes.dex在内存中解密并让Dalvik虚拟机加载执行

第一代安全加固技术 - 加固前后的变化



名字 (N)	大小 (Z)	修改 (M)
assets	17,029	2013/7/16 14:53:14
html	15,220	2013/7/16 14:53:14
channel_id	4	2012/10/8 14:22:16
prop	1,805	2012/10/8 14:22:16
META-INF	23,050	2013/7/16 14:53:14
res	389,847	2013/7/16 14:53:14
AndroidManifest.xml	7,720	2012/10/8 14:22:20
classes.dex	225,716	2012/10/8 14:22:20
resources.arsc	35,184	2012/10/8 14:22:20

名字 (N)	大小 (Z)	修改 (M)
assets	516,658	2013/7/16 15:55:19
html	15,220	2013/7/16 15:55:19
meta-data	11,665	2013/7/16 15:55:19
manifest.mf	11,334	2013/7/16 15:14:04
rsa.pub	274	2013/7/16 15:14:04
rsa.sig	257	2013/7/16 15:14:04
channel_id	4	2013/7/16 15:14:04
classes.jar	225,844	2013/7/16 15:14:04
com.secneo.keyoptimization	124,600	2013/7/16 15:14:04
com.secneo.keyoptimization.x86	137,320	2013/7/16 15:14:04
prop	1,805	2013/7/16 15:14:04
lib	293,512	2013/7/16 15:55:19
armeabi	103,524	2013/7/16 15:55:19
libsecexe.so	103,524	2013/7/16 15:14:04
x86	189,988	2013/7/16 15:55:19
libsecexe.so	189,988	2013/7/16 15:14:04
META-INF	24,709	2013/7/16 15:55:19
res	389,847	2013/7/16 15:55:19
AndroidManifest.xml	7,824	2013/7/16 15:14:04
classes.dex	19,308	2013/7/16 15:14:04
resources.arsc	35,184	2013/7/16 15:14:04

DEFCON 22

BANGCLE

Packers

- Well written, lots of anti-* tricks
- Seems to be well supported and active on development
- Does a decent job at online screening - no tool released for download
  - Though things clearly to slip through
- Not impossible to reverse and re-bundle packages
- Current weakness (for easy runtime unpacking) is having a predictable unpacked memory location
- Hacker Protect Factor 5

DEFCON 22上对梆梆1.0的分析 “Android Hacker Protection Level 0”
https://www.defcon.org/images/defcon-22/dc-22-presentations/Strazzere-Sawyer/DEFCON-22-Strazzere-and-Sawyer-Android-Hacker-Protection-Level-UPDATED.pdf

第一代安全加固技术 - 存在的问题

- 难以对抗动态分析
  - 内存中存在连续完整的解密后的代码，可通过内存dump的方式得到解密代码
- 针对内存dump，各家加固厂商都会打一些patch
  - Dex加载完毕后，抹掉或者混淆内存中dex的头部或者尾部信息(爱加密）
  - 检查Xposed框架是否存在，如存在，加固应用不运行 （360）
  - …
- 这些patch都是治标不治本的措施，无法从本质上解决内存dump的问题
  - 例如:通过修改Dalvik虚拟机，在载入dex时候进行dump，而不是在Dex已加载完成之后

第二代安全加固技术 - 原理

- 利用java虚拟机执行方法的机制
  - Java虚拟机在第一次执行某个类的某个方法前，才真正开始加载这个方法的代码
- 第二代加固技术
  - 加密粒度从Dex文件变为方法级别
  - 按需解密
  - 解密后代码在内存中不连续

第二代安全加固技术 - 实现

- 基于方法替换方式
  - 将原APK中的所有方法的代码提取出来，单独加密
  - 当Dalvik要执行某个方法是，加固引擎才解密该方法，并将解密后的代码交给虚拟机执行引擎执行
- 优点
  - 内存中无完整的解密代码
    - 每个方法单独解密，内存中无完整的解密代码
    - 如果某个方法没有执行，不会解密
    - 在内存中dump代码的成本代价很高

第二代安全加固技术 - 实现

原来的APK中的方法

```
method public onCreate(Landroid/os/Bundle;)V
    .locals 2
    .parameter "savedInstanceState"

    .prologue

    invoke-super {p0, p1}, Landroid/app/Activity;-
>onCreate(Landroid/os/Bundle;)V

    new-instance v0, Landroid/widget/TextView;

    invoke-direct {v0, p0}, Landroid/widget/TextView;-
><init>(Landroid/content/Context;)V

    return-void
.end method
```

原APK中方法的代码被空指令替换

实际代码被单独加密处理

加固后的APK中的方法

```
method public onCreate(Landroid/os/Bundle;)V
    .locals 2
    .parameter "savedInstanceState"

    .prologue

    nop

    nop

    nop

    nop

    nop

    return-void
.end method
```

APK的广度防御 - 资源、编程框架、So库

- 资源文件
  - 对APK中资源(图片，XML等文件)进行加密保护
- 采用编程框架开发的程序进行加密保护
  - DLL （C# Unity3D）
  - Lua
  - HTML/Javascript(IBM worklight、PhoneGap等)
  - Flash
  - …
- So库
  - 二进制C/C++的加壳保护

梆梆安全的加固技术

- APK由以下几个部分组成
  - **classes.dex**
    - 所有的Java代码
  - so库
    - 利用C/C++开发的库文件
  - 资源文件
    - 图片、xml配置文件等
    - 使用框架开发的脚本文件
      - Html5 (phoneGap、IBM worklight等)
      - lua
      - C# (Unity3D游戏引擎)
      - …

- 梆梆Android保护技术针对APK中的所有类型的文件都可以进行保护

上线后的渠道监控

- 目的
  - 监控第三方应用市场，及时发现盗版、山寨、钓鱼应用等

- 技术原理
  - 爬虫技术抓取第三方的应用程序
  - 应用相似度分析引擎判断是否有盗版和山寨
    - 基本信息相似度比较引擎
    - 资源相似度比较引擎
    - 代码相似度比较引擎
    - …

总结和建议

作为一种新的入口和用户交互平台，移动金融应用面临：

- 用户信息安全风险
- 交易安全风险
- 暴露移动端和服务端漏洞风险
- 钓鱼和二次打包风险

加强移动金融应用安全的途径

- 遵循安全规范的客户端开发
- 定期进行移动安全评估
  - 合规性评估/源代码审计/渗透性测试
- 移动客户端上线前进行安全加固
  - 防止反编译，逆向分析，动态注入攻击等等
- 上线后的渠道监测