移 动 安 全 形 势
及客户端安全编码规范

目录页

Contents Page

目录页

Contents Page

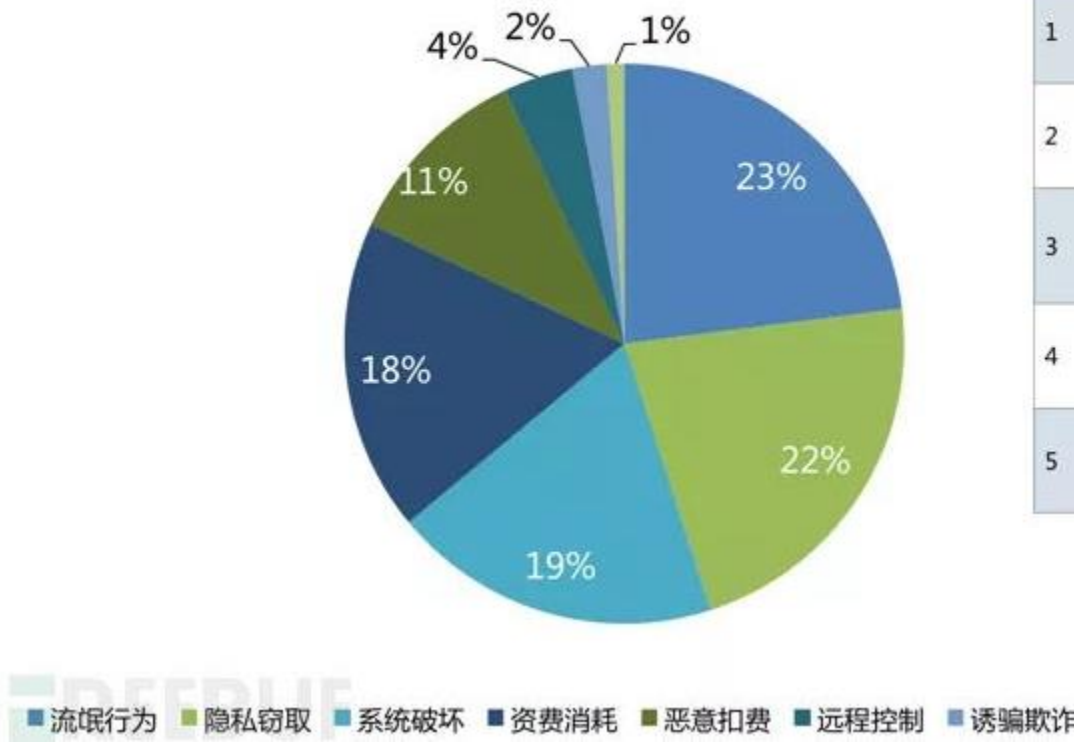
第一章

移动互联网安全形势

移动互联网情况概述

2017年截获手机病毒样本505万个，新增病毒类型以流氓行为、信息窃取、系统破坏、资费消耗四类为主，其中流氓行为类病毒占比23.3%，位居第一。其次是隐私窃取类病毒占比22.3%，第三名是系统破坏类病毒，占比19%。

2017年手机病毒类型比例



2017年手机病毒Top5

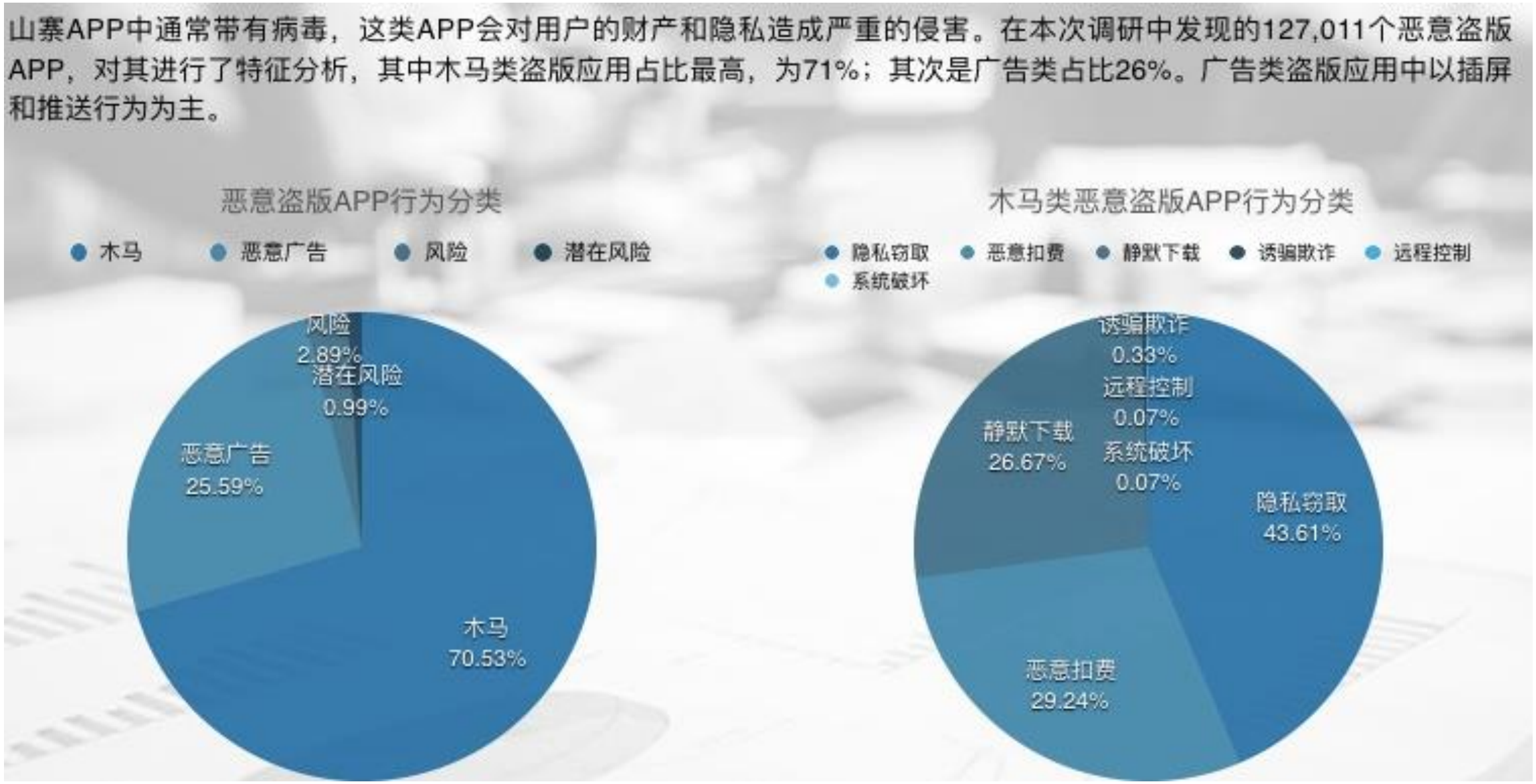
序号	病毒名	恶意行为
1		
2		
3		
4		
5		

2017年Android手机漏洞Top5

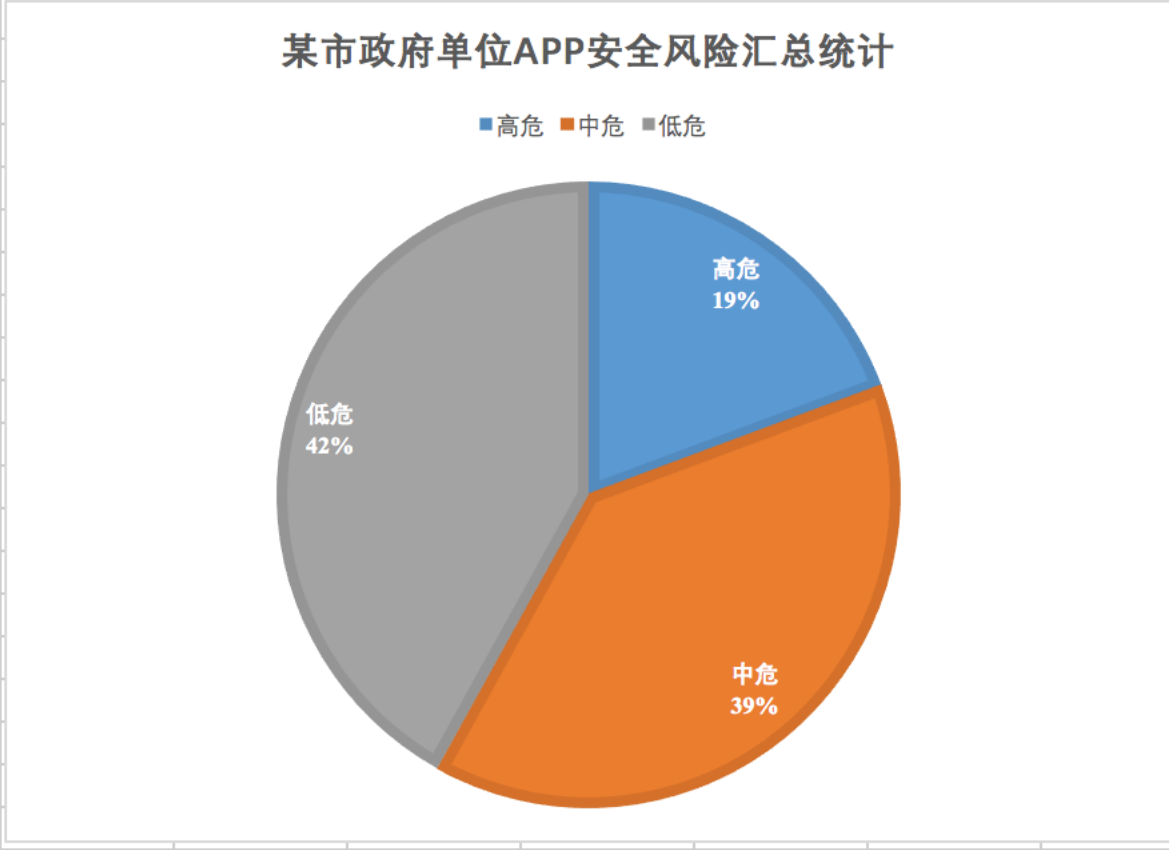
序号	漏洞名称	漏洞编号	简介
1	Janus高危漏洞	CVE-2017-13156	该漏洞允许攻击者任意修改Android应用中的代码，而且不会影响其签名。
2	BlueBorne蓝牙漏洞	CVE-2017-0785	蓝牙协议漏洞，只要手机开启了蓝牙，就可能被远程控制。
3	Android WebView 跨域访问漏洞	CNVD-2017-36682	攻击者利用该漏洞，可远程获取用户隐私数据（包括手机应用数据、照片、文档等敏感信息），还可窃取用户登录凭证，在受害者毫无察觉的情况下实现对APP用户账户的完全控制。
4	Android Media framework avc decoder远程代码执行漏洞	CNVD-2017-23420	远程攻击者可利用该漏洞执行任意代码。
5	roadpwn 漏洞	CVE-2017-9417	通过Wi-Fi使安卓手机崩溃。

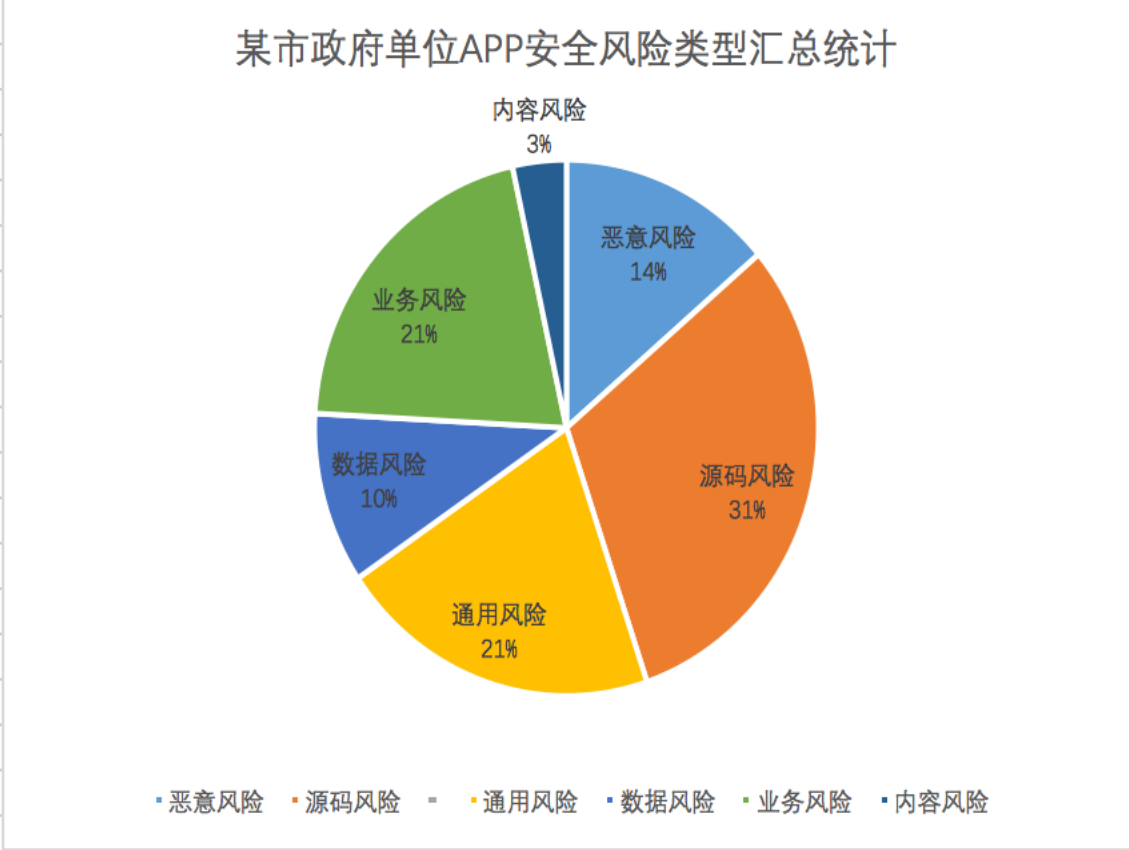
山寨APP恶意行为概述

山寨APP中通常带有病毒，这类APP会对用户的财产和隐私造成严重的侵害。在本次调研中发现的127,011个恶意盗版APP，对其进行了特征分析，其中木马类盗版应用占比最高，为71%；其次是广告类占比26%。广告类盗版应用中以插屏和推送行为为主。



政府行业移动应用安全风险



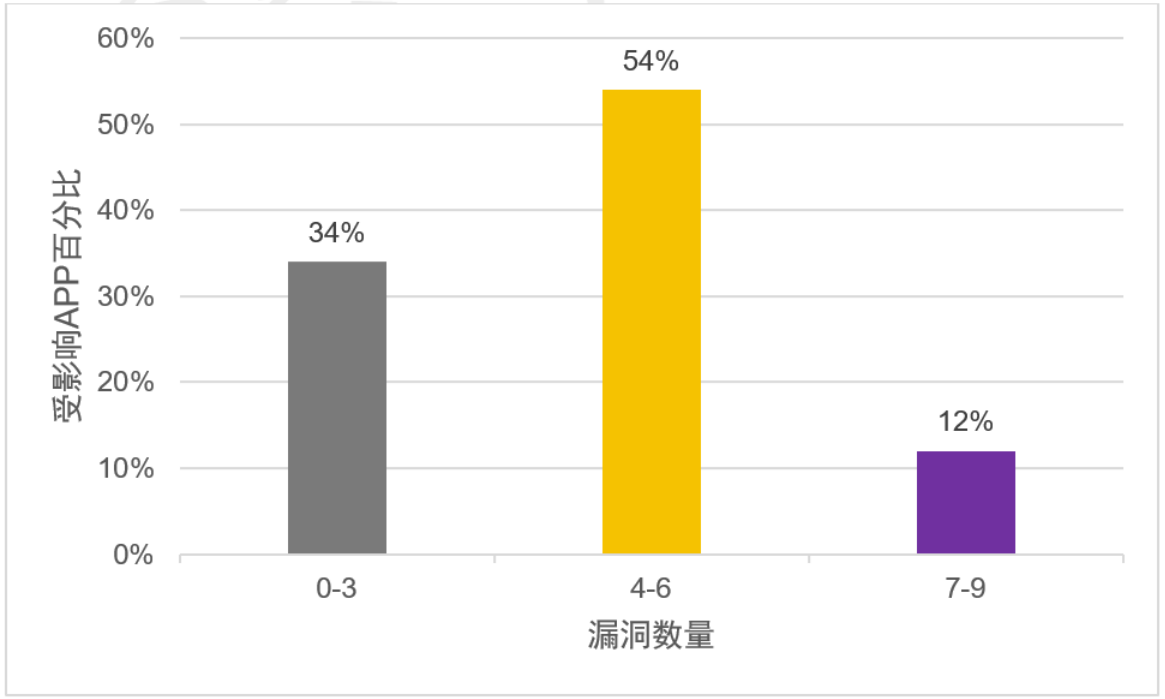


高危风险：对业务数据（如：用户账号、密码、银行卡密码等等）有窃取风险或者后门；对业务核心代码存在泄露得分析或者后门；严重漏洞

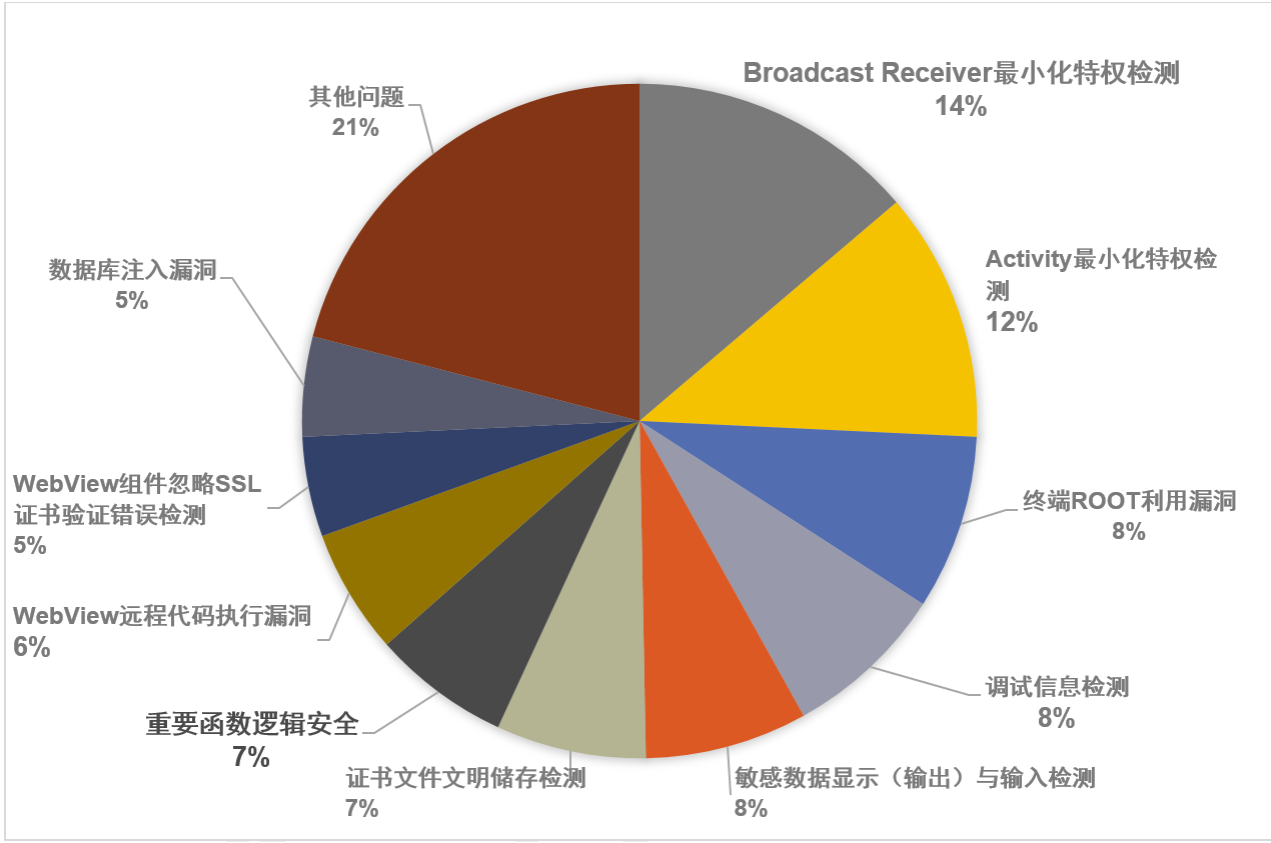
中危风险：对应用代码的不符合安全开发规范，有被第三方利用的风险；对应用内部文件存在数据被读取风险

低危风险：对应用代码有安全隐患，风险低

金融行业移动应用安全风险



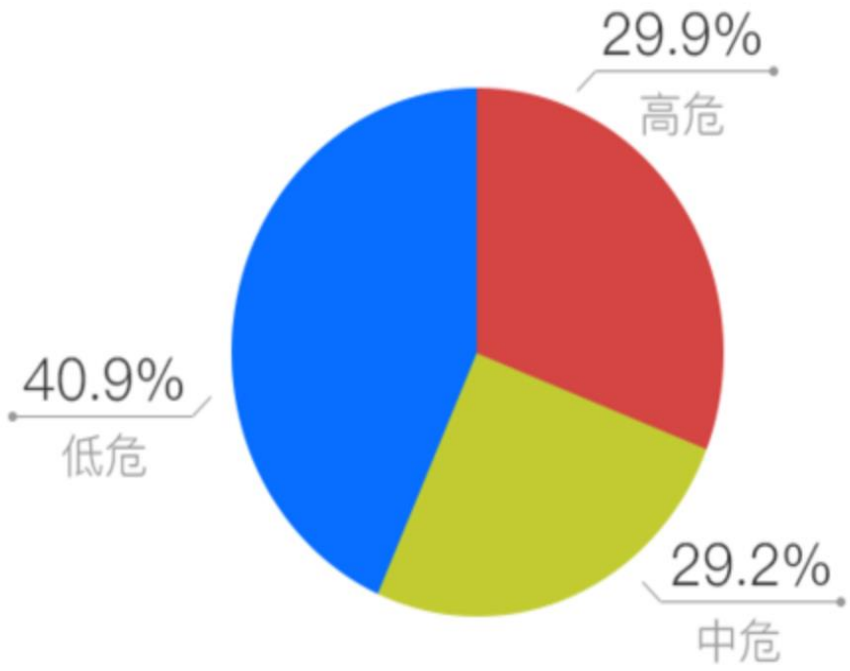
亚太地区金融行业APP（106款）分析表示，如图所示，当前85%应用没有通过基本的安全检测，50%的应用至少存在4至6个漏洞。



国内金融行业移动应用TOP10安全威胁

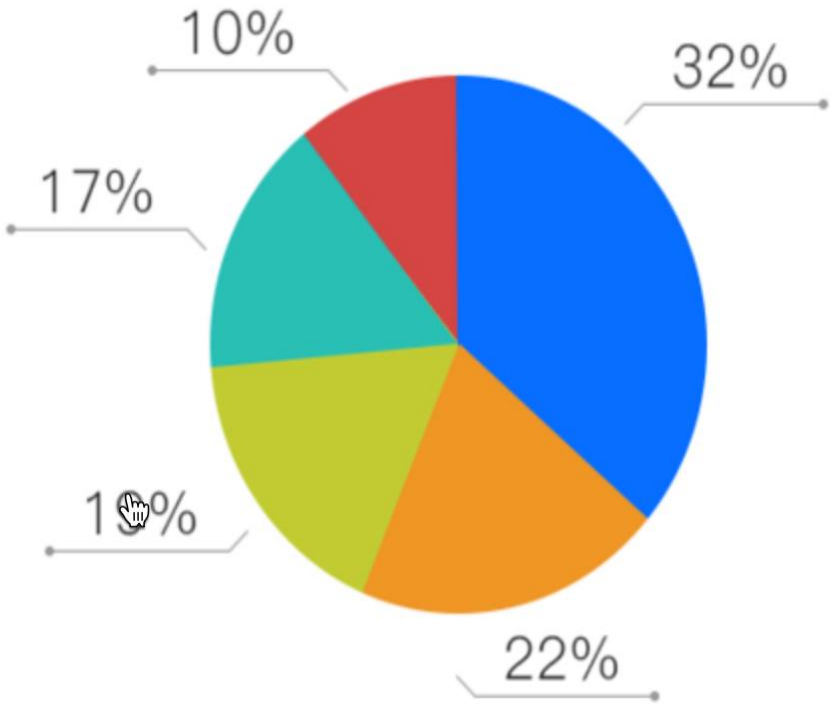
游戏行业移动应用安全风险

游戏行业移动 APP 漏洞评测



- 高危 游戏客户端与服务器数据传输不安全导致游戏数据可篡改、可泄漏等
- 中危 游戏客户端本地存档数据未加密，导致存档可篡改、隐私泄漏
- 低危 应用内存在部分逻辑不严谨，导致在某些情况下应用崩溃

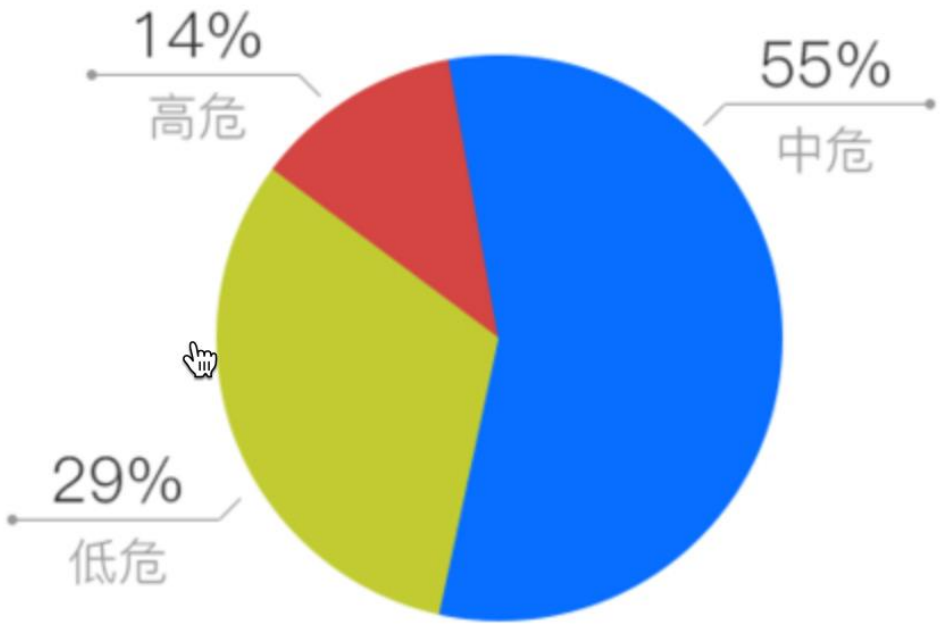
游戏行业移动 APP 安全问题类别



APP 破解　插入恶意代码　插入广告　外挂　其他

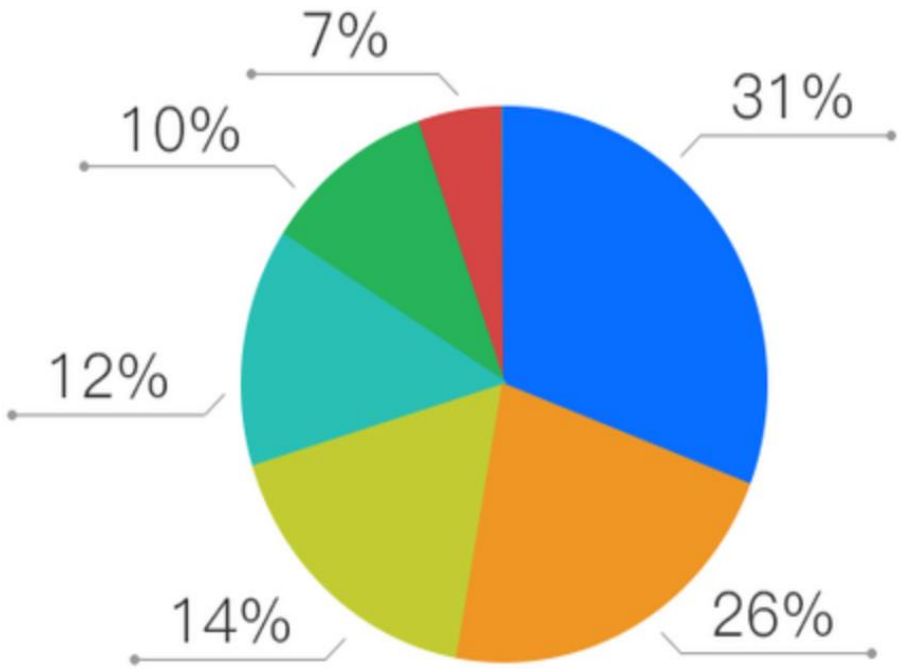
电商行业移动应用安全风险

电商行业移动 APP 漏洞评测



- 高危 数据传输不安全导致用户订单泄露、篡改
- 中危 本地数据存储不安全、用户隐私泄露
- 低危 APP 业务逻辑被破解、算法剽窃

电商 APP 安全问题类别



- 支付问题
- 被“薅羊毛
- 盗版问题
- 隐私泄露问题问题
- 应用破解
- 其他问题

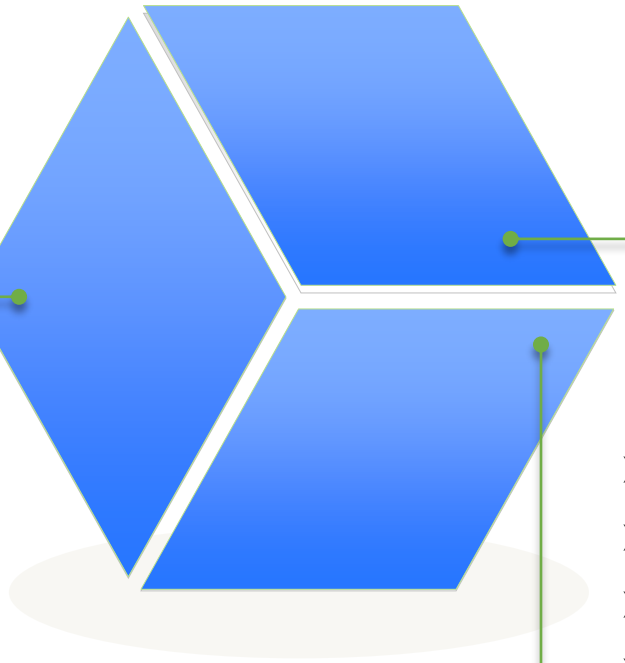
目录页

Contents Page

第二章

移动应用安全编码

2.1 Android客户端常见漏洞



组件安全

- Android组件安全
- Activity界面劫持
- Backup备份安全
- Java调试开启安全
- 本地拒绝服务安全
- WebView远程代码执行安全

数据安全

- WebView密码明文保存漏洞
- 本地数据全局读写漏洞
- 界面敏感数据显示泄露
- 调试日志泄露
- 内置账号/测试IP泄露

业务安全

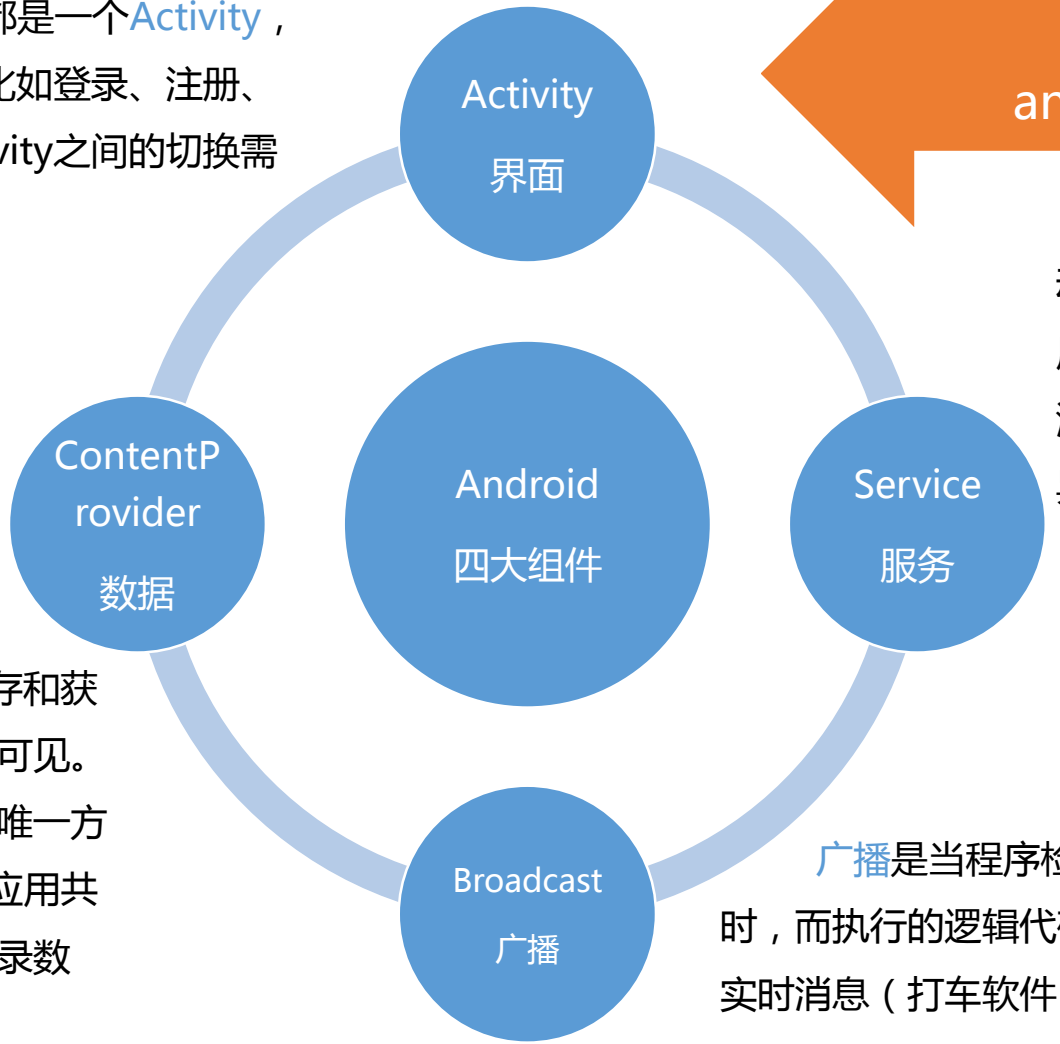
- HTTP/HTTPS中间人攻击
- 数据封包弱加密
- 手机验证码机制安全
- 服务器权限绕过
- 用户敏感信息泄露
- 手势密码绕过
- 会话保持机制安全
- 服务器请求重发攻击

第一大类：组件/控件安全

- Android组件安全
- Activity界面劫持
- Backup备份安全
- Java调试开启安全
- 本地拒绝服务安全
- WebView远程代码执行安全

1. Android组件安全

对于程序中的每一个界面都是一个Activity，每一个Activity有不同的功能，比如登录、注册、注册码验证、手势密码等，Activity之间的切换需要满足一定的条件。



最重要的风险点
android:exported属性

Service服务是伴随着程序启动，一直运行在后台，主要起检测作用的执行代码。服务一般用于时刻检测客户端的更新状态、时刻检测是否异地登录、时刻上传用户的操作信息

ContentProvider用于保存和获取数据，并使其对所有应用程序可见。这是不同应用程序间共享数据的唯一方式，因为android没有提供所有应用共同访问的公共存储区。比如通讯录数据。

广播是当程序检测到外界的某种运行环境发生变化时，而执行的逻辑代码，比如程序的自启、网络变化变化、实时消息（打车软件）。

四大组件的访问权限控制

组件暴露	android:exported 是Android中的四大组件 Activity，Service，Provider，Receiver 四大组件中都会有的一个属性，如果android:exported设置了true表示可对外进行访问或使用，如果android:exported设置了false表示不可对外进行访问或使用。
漏洞	如果对内部组件进行设置android:exported值为true，则可能出现组件被外部APP进行恶意访问、非法操作等风险。
演示	testapp.apk
编码要求	业务需求中需要对组件进行对外开放，则根据该开放性是针对于所有还是针对于个别，如果针对于所有则可设置android:exported为true即可，如果是针对于个别则通过自定义权限来进行访问安全控制。

Activity暴露风险

➢ 权限绕过

```
am start -n cn.com.gxluzj/.frame.impl.module.home.HomeActiv
 app_process has text relocations. This is wasting memory a
ase fix.
 app_process has text relocations. This is wasting memory a
```

adb启动登录界面后的界面Activity

POC:

商旅纵横手势密码绕过

```
dz> run app.activity.start --component com.umetrip.android.msky.app com.umetrip.
android.msky.activity.MainActivity
```

运营商内部APP登录界面绕过

Activity暴露风险

➢ 本地拒绝服务攻击

```
Caused by: java.lang.IllegalStateException: Can't find the mandatory extra identified by key [oid] on field class
com.meituan.android.hotel.voucher.HotelVoucherVerifyActivity.orderId
java.lang.RuntimeException: Unable to resume activity {com.sankuai.meituan/com.meituan.android.hotel.booking.Book
java.lang.NullPointerException
java.lang.RuntimeException: Unable to start activity ComponentInfo{com.sankuai.meituan/com.meituan.android.hotel.
java.lang.NullPointerException
java.lang.RuntimeException: Unable to start activity ComponentInfo{com.sankuai.meituan/com.sankuai.meituan.poi.br
java.lang.NullPointerException
java.lang.RuntimeException: Unable to start activity ComponentInfo{com.sankuai.meituan/com.sankuai.meituan.topic.
java.lang.NullPointerException
```

手机美团拒绝服务攻击

11.Android本地拒绝服务攻击

Android提供Intent机制来协助应用间的交互与通讯，Intent负责对一次操作的动作、动作涉及的数据进行描述，系统则根据此Intent描述，来调用对应的Activity、servicer和BroadCast等组件，来完成组件的调用。如果程序没有对Intent.getXXXExtra()获取的异常或者畸形数据处理时没有进行异常捕获，就会导致攻击者可通过向受害者应用发送此类空数据、异常或者畸形数据来使该应用崩溃，简单的说就是通过intent发送空数据、异常或畸形数据给应用，会导致其崩溃。

Java.lang.NullPointerException	Java.lang.ClassCastException	Java.lang.IndexOutOfBoundsException	Java.lang.ClassNotFoundException
• NullPointerException异常导致的拒绝服务，源于程序没有对getAction()等获取到的数据进行空指针判断，从而导致空指针异常而导致应用崩溃	• ClassCastException异常导致的拒绝服务, 源于程序没有对getSerializableExtra()等获取到的数据进行类型判断而进行强制类型转换	• IndexOutOfBoundsException异常的拒绝服务，源于程序没有对getIntegerArrayListExtra()等获取到的数据数组元素大小的判断	• lassNotFoundException异常导致的拒绝服务，源于程序没有无法找到从getSerializableExtra()获取到的序列化类对象的类定义.

2.Service服务暴露

① 优酷土豆检测版本更新使用的是service服务，反编译到的代码：

```
<service
    android:label="Youku Push Notifications StartActivityService"
    android:name="com.youku.service.push.StartActivityService"
    android:exported="true"
>
```

② 传入PushMsg的Serializable的数据：

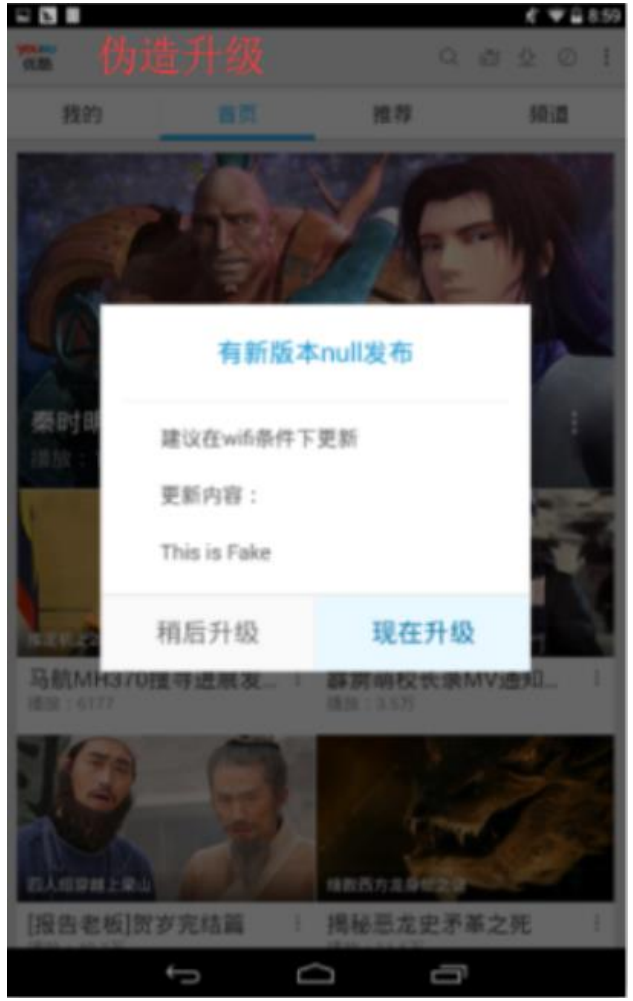
```
......
label_53:
    intent.setFlags(876609536);
    intent.setClass(this, UpdateActivity.class);
    intent.putExtra("updateurl", pushMsg.updateurl);
    intent.putExtra("updateversion", pushMsg.updateversion);
    intent.putExtra("updatecontent", pushMsg.updatecontent);
    intent.putExtra("updateType", 2);
    this.startActivity(intent);
    return;
......
```

优酷客户端内的数据序列

③ 恶意伪造并启动暴露的service：

```
PushMsg pushMsg = new PushMsg();
pushMsg.type = 1;
pushMsg.updateurl = "http://gdown.baidu.com/data/wisegame/41839d1d51
pushMsg.updatecontent = "This is Fake";
Intent intent = new Intent();
intent.setClassName("com.youku.phone", "com.youku.service.push.Start
intent.putExtra("PushMsg", pushMsg);
startService(intent);
```

Service服务暴露



伪造升级

拒绝服务

3.Broadcast 暴漏

借助广播，监听易到用车的订单信息

```
<receiver android:name=".PushMsgReceiver" >
    <intent-filter>
        <action android:name="com.yongche.component.groundhog.RECEIVED_MESSAGE" />
    </intent-filter>
</receiver>
```

恶意注册易到广播

```
public class PushMsgReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        String str = intent.getAction();
        System.out.println(str);
        String str2 = intent.getExtras().getString("message");
        System.out.println(str2);
    }
}
```

设置Massage监听

```
om.example.pushmessager...  System.out
om.example.pushmessager...  System.out
```

最终监听到订单

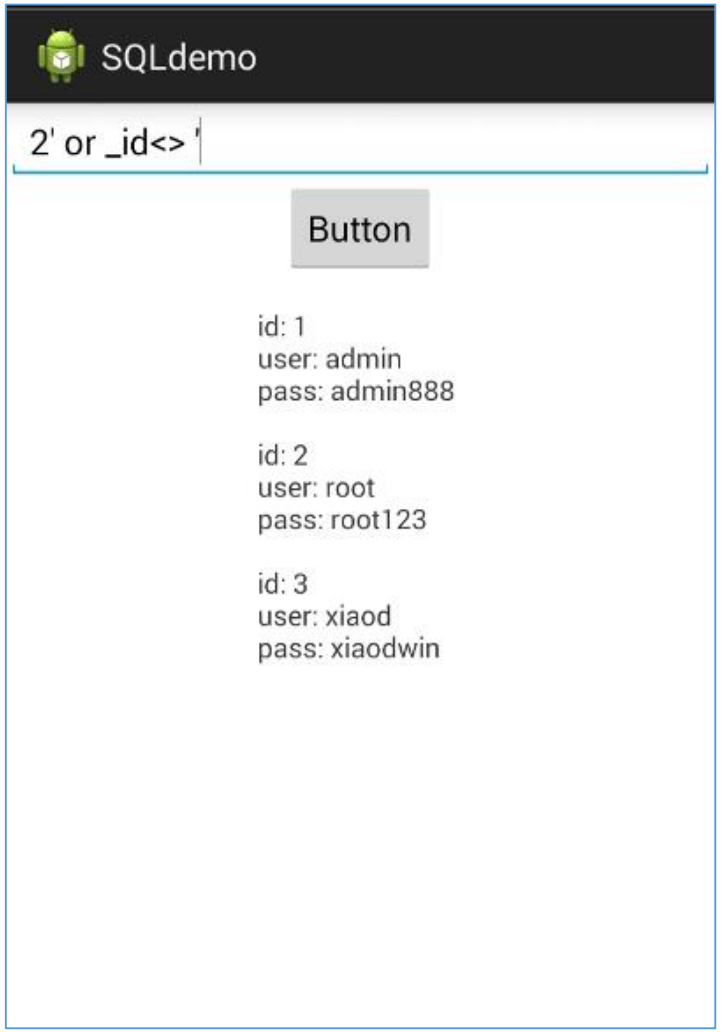
```
com.yongche.component.groundhog.RECEIVED_MESSAGE
{"jsonrpc":"2.0","id":1,"method":"order.add","params":{"order":{"order_id":48528760,"last_update
,"asap":0,"product_type":8,"amount":"0.00","extra_amount":"0.00","expect_start_time":1431504000,
t":6,"is_coporate":0,"is_face_pay":0,"is_support_face_pay":1,"balance_description":"无需代充值",
```

5.Android本地数据库SQL注入

两种不同的DB数据库查询方法

```
private String ShowData(String m_id) {
    this.result = "";
    new String[1][0] = m_id;
    Cursor v1 = this.m_db.rawQuery("SELECT * FROM usertable WHERE _id = \'" + m_id + "\'", null);
    v1.moveToFirst();
    while(!v1.isAfterLast()) {
        this.result = String.valueOf(this.result) + "id: " + v1.getInt(0) + "\n" + "user: " + v1
                .getString(1) + "\n" + "pass: " + v1.getString(2) + "\n\n";
        v1.moveToNext();
    }

    v1.close();
    return this.result;
```

```
vate String ShowData2(String m_id) {
this.result = "";
Cursor v1 = this.m_db.rawQuery("SELECT * FROM usertable WHERE _id = ?", new String[]{m_id});
v1.moveToFirst();
while(!v1.isAfterLast()) {
    this.result = String.valueOf(this.result) + "id: " + v1.getInt(0) + "\n" + "user: " + v1
            .getString(1) + "\n" + "pass: " + v1.getString(2) + "\n\n";
    v1.moveToNext();
}
```



防止SQL注入编码

编码要求

应避免使用外来字符串拼接SQL语句，或对参与SQL语句拼接的外来字符串进行过滤。

➢ 修复示例：SQL_Injection（SQL注入）

通过使用正则表达式对参数进行非法过滤来防止sql 注入。

```
Cursor m_cursor;
String m_argv[] = { m_id };
m_cursor = m_db.rawQuery("SELECT * FROM usertable WHERE _id
= '" + m_id+ "'", null);
m_cursor.moveToFirst();
while (!m_cursor.isAfterLast()) {
        result += "id: "+m_cursor.getInt(0)+"\n"+"user:"
        + m_cursor.getString(1) + "\n" + "pass: "
        + m_cursor.getString(2) + "\n\n";
        m_cursor.moveToNext();
}
m_cursor.close();
```

修复为

```
//引入包
import java.util.regex.*;
//典型的SQL 注入攻击的正则表达式
private String CHECKSQL =
“/\w*((\%27)|(\’))((\%6F)|o|(\%4F))((\%72)
|r|(\%52))/ix”;
判断是否匹配：
Pattern.matches(CHECKSQL,targerStr);
```

6.Intent隐式跳转风险

隐试Intent跳转	隐式调用一般是用于在不同应用程序之间的组件跳转，因跳转目标是由系统来进行判断，特殊情况下会出现目标错误，造成传递数据泄露的安全风险
漏洞	通过设置动作(action)、类别(category)、数据（URI和数据类型）等隐式意图的方式来进行目标跳转，Android系统会根据设置的隐式意图找到最合适的组件作为目标组件进行跳转。
演示	IntentDemo.apk IntentCall.apk

Intent跳转编码规范

编码要求

在使用Intent进行组件间跳转时应尽量使用显示调用，非特殊情况应避免使用隐式调用。

➢ 修复示例：应尽量使用显示调用

在使用Intent进行组件间跳转，组件间数据传递应尽量使用显示调用。

```
Intent intent = new Intent();
intent.setAction("com.xiazdong.action");
intent.addCategory("com.xiazdong.category");
intent.setData(Uri.parse("xiazdong://www.xiazdong.com/xia"));
startActivity(intent);
```

修复为

```
Intent intent = new Intent();
Bundle bundle = new Bundle();
bundle.putString("id", "strID");
intent.setClass(this,
Intent_Demo1_Result1.class);
intent.putExtras(bundle);
startActivity(intent);
```

7.Backup备份安全

Android API Level 8及其以上Android系统提供了为应用程序数据的备份和恢复功能。此功能的开关决定于该应用程序中AndroidManifest.xml文件中的allowBackup属性值，其属性值默认是True。当allowBackup标志为true时，用户即可通过adb backup和adb restore来进行对应用数据的备份和恢复，这可能会带来一定的安全风险。

- 备份指令：

```
adb backup -nosystem -noshared -apk -f com.xx.xxxx F:\Restore
```

- 还原指令：

```
adb backup -f F:\ Restore -apk com.xx.xxxx
```

Backup备份安全

情景案例

某IT男一直暗恋部门某女神，一天女神手机太卡了找IT男帮助清理手机空间。

IT男高兴地答应女神两分钟搞定，屁颠屁颠的跑到自己电脑旁边连上手机。

女神在一边呆呆的看着IT男敲了几行代码然后在手机上点了几下，最后果然两分钟不到就搞定了。

在女神谢着离开后，IT男露出了WS的笑容……

8.Java调试开启风险

Java调试的开启是指在客户端开发时，主配置文件AndroidMainfest.xml中设置了debuggable的Debuggable属性为true，导致产生以下风险：

➢ jdb调试

获取和篡改用户敏感信息，甚至分析并且修改代码实现的业务逻辑，例如窃取用户密码，绕过验证码防护等

➢ Release和Debug的调试

Release和Debug模式下，程序运行两种不同逻辑流程，比如：

- Debug则走程序中的内测流程，使用内网IP、Log调试日志全开；
- Relase下则走线上的发布版本。

jdb进行调试

获取和篡改用户敏感信息，甚至分析并且修改代码实现的业务逻辑，例如窃取用户密码，绕过验证码防护等

```
$ adb shell ps | grep com.android.settings
system    3107  2379  1676912 132884 ffffffff 00000000 S com.android.settings
$ adb forward tcp:12345 jdwp:3107
$ jdb -attach localhost:12345
设置未捕获的java.lang.Throwable
设置延迟的未捕获的java.lang.Throwable
正在初始化jdb...
> classes
** 类列表 **
android.R$styleable
android.accounts.Account
android.accounts.Account$1
android.accounts.AccountManager
android.accounts.AccountManager$16
android.accounts.AccountManager$8
android.accounts.AccountManager$AmsTask
android.accounts.AccountManager$AmsTask$1
android.accounts.AccountManager$AmsTask$Response
android.accounts.AccountManagerCallback
```

```
设置断点com.android.settings.Settings.onCreate(android.os.Bundle)
>
断点命中: "线程=main", com.android.settings.Settings.onCreate(), 行=715 bci=3

main[1] next
>
已完成的步骤: "线程=main", com.android.settings.Settings.onCreate(), 行=718 bci=

main[1] step
>
已完成的步骤: "线程=main", com.android.internal.widget.LockPatternUtils.<init>(
```

9.使用IMEI作为设备唯一标识风险

唯一ID认证

由于Android领域中有不少模拟器，而模拟器可模拟与篡改IMEI；IMEI在个别特殊系统与终端中是无法获取到的，基于如上分析如果把IMEI作为设备的唯一ID将出现一定的重复与无法获取的几率。

演示

imeiDemo.apk

编码要求

在需要使用设备唯一ID的业务场景中，应使用DEVICE_ID、 MAC ADDRESS、 Sim Serial Number、 IMEI等数据进行字符串组合后生成Hash值作为设备唯一ID。避免单独使用IMEI作为设备唯一ID标识。

可识别设备的标识特征

```
// 获取设备识别号
@SuppressLint("ServiceCast")
public static void getDevide(Context Ctx) {
    TelephonyManager tm = (TelephonyManager) Ctx.getSystemService(Cont
    imei = tm.getDeviceId();  //IMEI：手机序列号
    imsi = tm.getSubscriberId();  //IMEI：手机卡序列号
    sim = tm.getSimState();  //获取SIM卡状态
    iccid = tm.getSimSerialNumber();  //获取SIM卡串号
    netopera = tm.getNetworkOperatorName();  //获取手机网络运营商的名称
    carrier = tm.getSimOperatorName();   //获取手机卡的运营商名称
```

```
public static void sysinfo(Context Ctx) {
    model = Build.MODEL;  //手机型号
    manufactor = Build.MANUFACTURER;  //手机生产厂商
    osver = Build.VERSION.RELEASE;   //
    osverint = Build.VERSION.SDK_INT;  //手机系统API版本
    android_id = Secure.getString(Ctx.getContentResolver(), Secure.ANDROID_ID
    lang = Locale.getDefault().getLanguage();
    country = Locale.getDefault().getCountry();

    WindowManager window = (WindowManager) Ctx.getSystemService(Context.WINDO
    DisplayMetrics Dis = new DisplayMetrics();
    window.getDefaultDisplay().getMetrics(Dis);
    int widthPixels = Dis.widthPixels;
    int heightPixels = Dis.heightPixels;
    sccreen = widthPixels + "x" + heightPixels;
```

10.Release和Debug模式

➢ Release和Debug模式

Release和Debug模式下，程序运行两种不同逻辑流程，比如：

- Debug则走程序中的内测流程，使用内网IP、Log调试日志全开；
- Relase下则走线上的发布版本。

```
setContentView(2130968688);
this.mLogo = ((ImageView)findViewById(2131624528));
String str1 = ZbjSecureUtils.a().getAuthKey(0);
String str2 = MD5.Md5(ZbjPackageUtils.getSignature(this, "com.zhubajie.client"));
a.c("-------------", str2.equals(str1) + "");
if (!str2.equals(str1)) && (!Config.DEBUG))
{
  showTip("您的软件有盗版风险，请卸载后从正规途径重新下载！");
  finish();
  return;
}
this.mHandler.postDelayed(this.enterHome, 1000L);
```

从Native层获取一个字符串

读取该APP的签名信息

在签名信息不一致，并且不是DEBUG模式下盗版提示

所以只需将str1恒等于str2，或者将Config中的DEBUG调试模式即可。

```
96
97      .prologue
98      .line 50
```

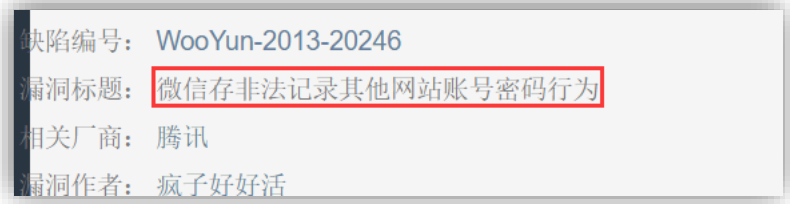
设置为1，开启DEBUG调试模式

12.WebView控件漏洞

WebView是Android客户端浏览Web端页面（ “跨端” ）的系统控件，存在着Web逻辑与本地客户端逻辑的交互，基于这个特点，WebView存在安全风险点主要有以下几点：

➢ WebView密码明文保存漏洞

- （ mWebView.getSettings().setSavePassword(false) ）
- 在使用WebView的过程中忽略了WebView setSavePassword，当用户选择保存在WebView中输入的用户名和密码，则会被明文保存到应用数据目录的databases/webview.db中。



这个问题的产生原因并不复杂：微信使用webview加载微博的OAuth登录和认证页面，并采用了webview的默认设置。这种情况下，用户输入账户和密码登陆微博时，系统会弹出提示询问是否保存密码。如果用户选择了是，密码就会保存在这个应用私有目录的databases/webview.db中。

➢ WebView忽略SSL证书验证错误漏洞

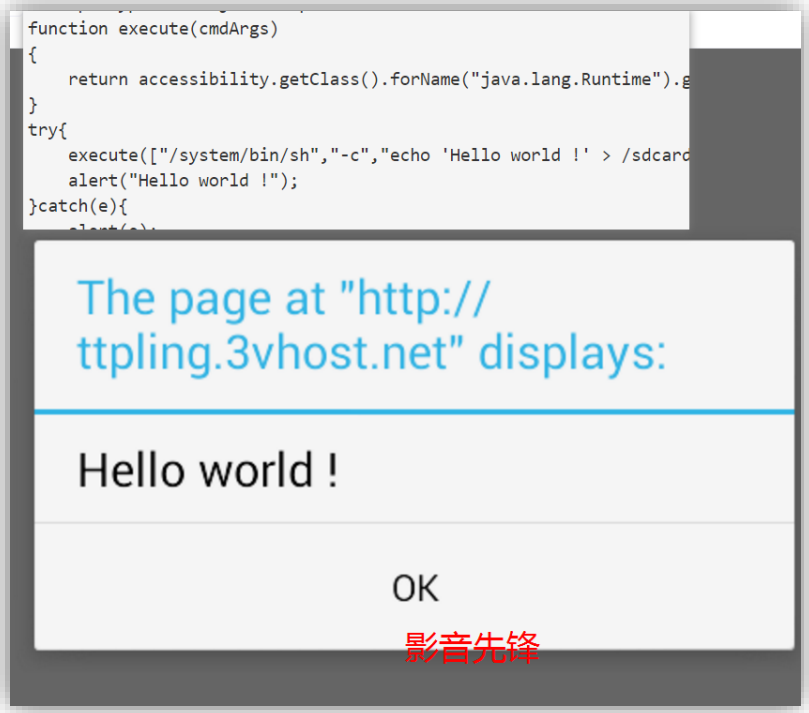
- （ handler.proceed() ）
- WebView组件加载网页发生证书认证错误时，会调用WebViewClient类的onReceivedSslError方法。 如果该方法实现调用了handler.proceed()来忽略该证书错误，则会受到中间人攻击的威胁。该漏洞常见于Android浏览器在访问Https时候的认证方式。

WebView远程代码执行漏洞

漏洞成因：Android系统为了方便应用中Java代码和网页里的Javascript代码进行交互，在WebView控件中实现了addJavascriptInterface接口，网页中的Javascript代码可以利用该接口定义的名字调用Java层的代码。而Java对象继承关系会导致很多Public的函数（其中包括getClass）都可以在JS中访问，所以JS可以通过该接口利用Java的反射机制获得系统类的函数，绕开Webview所在的代码上下文。

该漏洞曾在乌云上风靡一时！然而，该漏洞已一去不返，目前只存在Android4.4.2（API：17）及以下。

WebView远程代码执行漏洞



Activity界面劫持

Activity界面劫持类似于Web端的“页面钓鱼”，就是通过伪造与原应用非常相像的登录界面、注册界面、找回密码界面等，欺骗用户输入账户名和密码，然后后台将账户密码发送到指定地方的黑客手段，比如以下对几个金融APP敏感界面劫持的测试结果：

第二大类：数据安全

- 本地数据全局读写漏洞
- 调试信息泄露

1.本地数据全局读写漏洞

APP在创建数据库时，将数据库设置了全局的可读权限，攻击者恶意读取数据库内容，获取敏感信息。在设置数据库属性时如果设置全局可写，攻击者可能会篡改、伪造内容，可以能会进行诈骗等行为，造成用户财产损失。

以下为数据库全局读写的一种实现方式，漏洞代码样例：

```
...
        try {
            SQLiteDatabase rdb = openOrCreateDatabase("all_r_db",
                    Context.MODE_WORLD_READABLE, null);
            SQLiteDatabase wdb = openOrCreateDatabase("all_w_db",
                     Context.MODE_WORLD_WRITEABLE, null, null);
        } catch (SQLiteException e) {
            e.printStackTrace();
        }
    ...
```

正确做法

应该是使用*MODE_PRIVATE*模式创建数据库。

2.调试信息泄露

➢ 运行日志信息泄露

调试信息泄露

➢ 测试内网/账号的泄露

```
setContentView(2130903089);
this.mAccountInput = ((CPPhoneInput)findViewById(2131165399));
this.mPwdInput = ((CPXPasswordInput)findViewById(2131165400));
this.mAccountInput.setText("18601005417");
this.mPwdInput.setText("sx291584");
this.mLoginBtn = ((CPButton)findViewById(2131165401));
this.mLoginBtn.observer(this.mAccountInput);
this.mLoginBtn.observer(this.mPwdInput);
```

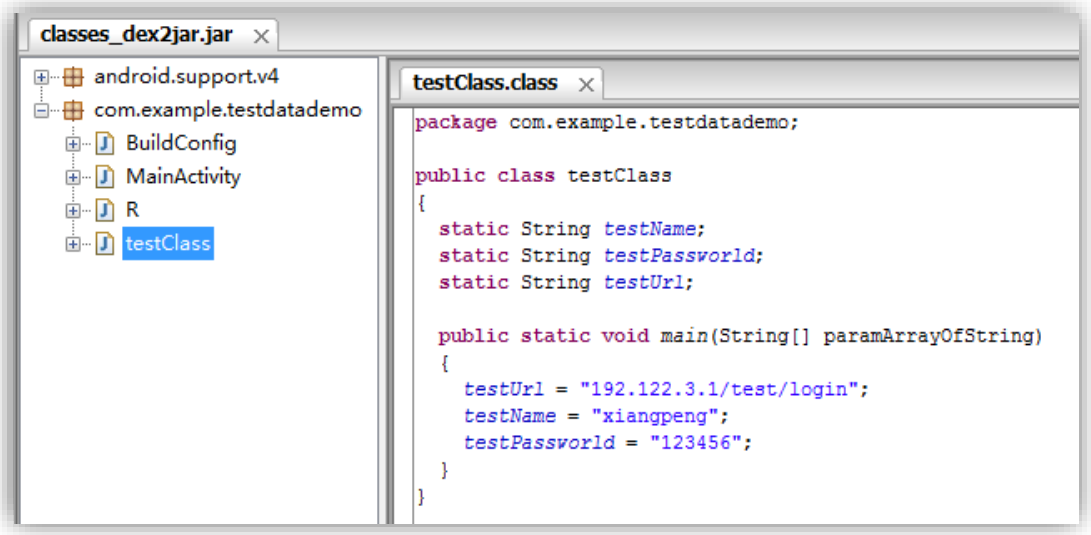
京东某钱包泄露内部测试人员账号密码

还有些APP在发布版本内部还嵌这测试环境代码，当符合某种条件时，顺利调出测试流程，如某八戒外包网：

3.避免测试数据残留

测试数据残留	发布版本应对程序中所有测试数据、测试方法进行统一删除。
内网数据残留	发布版本应对程序中所有的内网数据进行统一删除。
演示	testDataDemo.apk

残留的测试数据，例如URL地址、测试账号、密码等可能会被盗取并恶意使用在正式服务器上进行攻击，例如账号重试，攻击安全薄弱的测试服务器以获取服务器安全漏洞或逻辑漏洞



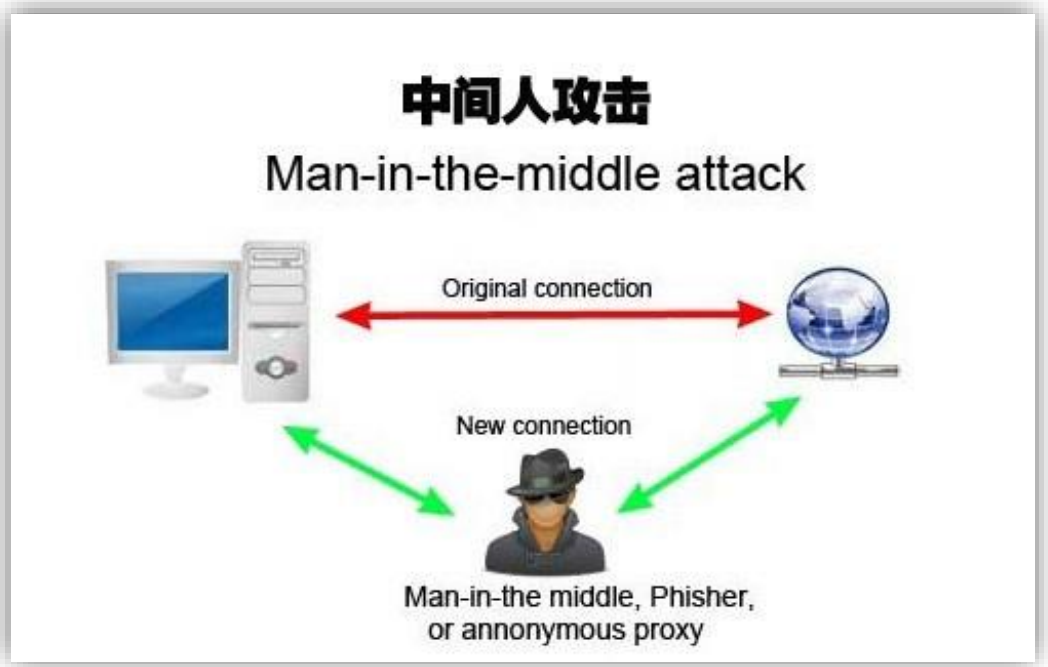
第三大类：业务安全

- HTTP/HTTPS中间人攻击
- 数据封包弱加密
- 手机验证码机制安全
- 会话保持机制安全
- 其他漏洞

1.HTTP/HTTPS中间人攻击

中间人攻击（Man-in-the-middle attack，缩写：MITM）是指攻击者与通讯的两端分别建立独立的联系，并交换其所收到的数据，使通讯的两端认为他们正在通过一个私密的连接与对方直接对话，但事实上整个会话都被攻击者完全控制。

在中间人攻击中，攻击者可以拦截通讯双方的通话并插入新的内容。



HTTP下的升级劫持

- HTTP协议不具有防止中间人攻击的安全性（易信、友盟）

缺陷编号： WooYun-2015-150915
漏洞标题： 易信安卓客户端升级过程存在缺陷，可被中间人攻击利用植入木马
相关厂商： 网易
漏洞作者： zhchbin
提交时间： 2015-11-02 16:32

http://**.**.**.**/android_update.json

检查是否有新版本更新，该接口返回内容如下：

{"versionCode":217,"md5":"c2c5fa68420ca56e91e72e38ef140ec8","downloadUrl":"http:/
k","title":"5qOA5rWL5Yiw5paw54mI5pys","description":"MS7jgJDmlrDlop7jgJHnvqTkuLvm
DmiJDlkZjnmoTmnYPpmZDphY3nva4KMi7jgJDmlrDlop7jgJHpnZLmnpznmoRH\nSUbmiqXorablm77ni
pnZLmnpznlKjmiLflj6/ku47igJzlj5HnjrDigJ3kuK3igJzpnZLmnpzmkYTlg4/mnLrigJ3n\nm7Tmjc
a/liJfo\noajkuK3kv6Hmga/mlLblj5HnmoTnvKnnlaXmlofmoYgKNS7jgJDkvJjljJbjgJHosIPmlbTk
3ml7bplb/lhZHmjaLm\ntYHnqIsKNi7jgJDkvJjljJbjgJHkvJjljJboh6rlrprkuYnooajmg4Xmt7vli
lpI3oi6XlubJidWc=","notify":true,"mini":0,"length":46306404,"patch":false,"patchM

HTTPS中间人攻击

- 客户端不验证服务器是否可信，即checkServerTrusted()方法为空

```
@Override
public void checkClientTrusted(
            X509Certificate[] chain, String authType) {
    // Do nothing -> accept any certificates
}

@Override
public void checkServerTrusted(
            java.security.cert.X509Certificate[] chain, String authType) {
    // Do nothing -> accept any certificates
}
```

- 不检查站点域名与站点证书的域名是否匹配

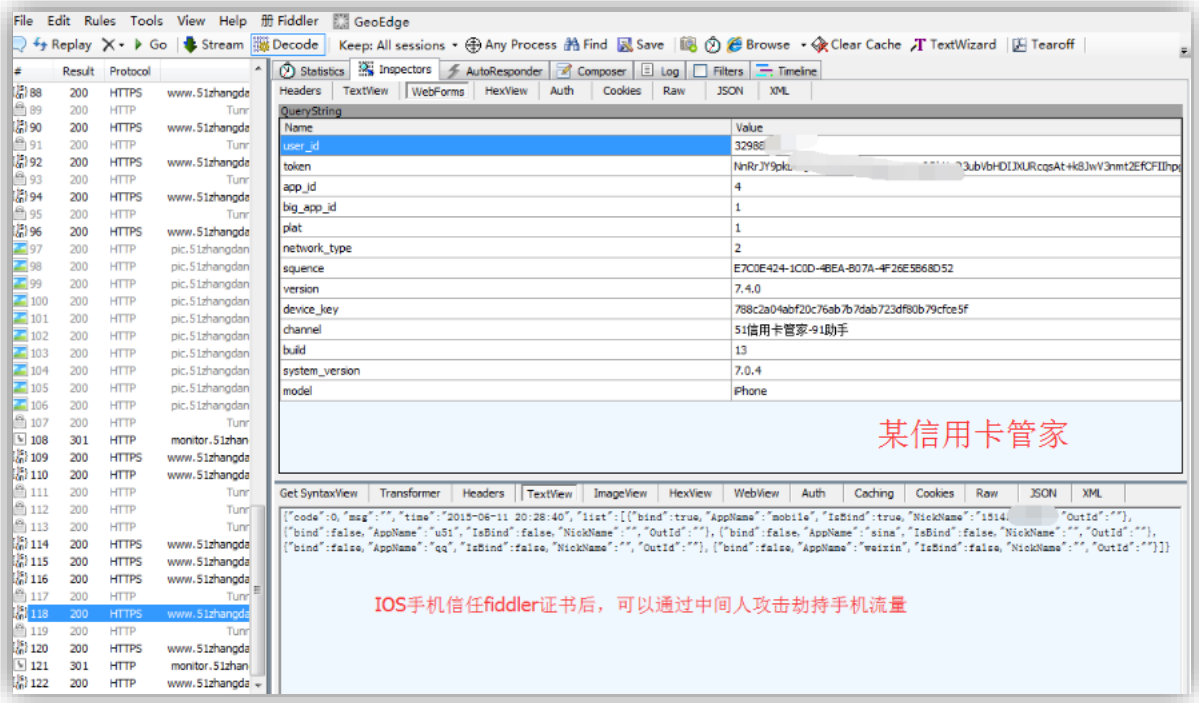
```
HostnameVerifier hv = new HostnameVerifier() {
    @Override
    public boolean verify(String hostname, SSLSession session) {
    // Always return true -> Accespt any host names
    return true;
    }
};
```

- 接受任意域名

```
SSLSocketFactory sf;
……
sf.setHostnameVerifier(SSLSocketFactory.ALLOW_ALL_HOSTNAME_VERIFIER);
```

【银行案例】


873 200 HTTP Tunnel to mobilebank.bsb.com.cn:443 0
874 200 HTTPS mobilebank.bsb.co... /MBMC/login/ClientLogin.do?method=... 65
875 200 HTTP Tunnel to mobilebank.bsb.com.cn:443 0
876 200 HTTPS mobilebank.bsb.co... /MBMC/login/ClientLogin.do?method=... 89
Headers TextView WebForms HexView Auth Cookies Raw JSON XML
Request Headers
POST /MBMC/login/ClientLogin.do?method=auth HTTP/1.1
X-Powered-By: Servlet/2.5 JSP/2.1
Transfer-Encoding: chunked
Content-Type: application/json;charset=UTF-8
Connection: close
Via: 1.1 ID-0314217204522734 uproxy-3
4e
{"MSG":"该用户未开通手机银行！","OS_TYPE":"iphone","STATUS":"0005"}
0
某银行客户端
www.wooyun.


File Edit Rules Tools View Help Fiddler GeoEdge
Replay Go Stream Decode Keep: All sessions Any Process Find Save Browse Clear Cache TextWizard Tearoff
Statistics Inspectors AutoResponder Composer Log Filters Timeline
Headers TextView WebForms HexView Auth Cookies Raw JSON XML
QueryString
Name Value
user_id 32985
token NnRrJY9pk...
app_id 4
big_app_id 1
plat 1
network_type 2
squence E7C0E424-1C0D-4BEA-B07A-4F26E5B68D52
version 7.4.0
device_key 788c2a04abf20c76ab7b7dab723df80b79cfce5f
channel 51信用卡管家-91助手
build 13
system_version 7.0.4
model iPhone
某信用卡管家
Get SyntaxView Transformer Headers TextView ImageView HexView WebView Auth Caching Cookies Raw JSON XML
IOS手机信任fiddler证书后，可以通过中间人攻击劫持手机流量

2.数据封包弱加密

常见的对称加密算法（AES/DES/3DES/PEB）

algo	key len	iv len	cipher mode
AES 128	16	ecb:nil ctr:16	"ecb", "cbc"
AES 192	24	ofb:16 cfb:16	"cfb", "gcm"
AES 256	32	cbc:16 gcm:16	"ctr", "ofb"
DES	8	ecb:nil others:8	"ecb" "cbc" "cfb" "ofb"
DESede	16	ecb:nil others:8	"ecb" "cbc" "cfb" "ofb"
3DES	24	ecb:nil others:8	"ecb" "cbc" "cfb" "ofb"

常见的非对称加密算法（RSA/DSA、ECB）

```
byte[] keyBytes = {...};
byte[] data = {...};
X509EncodedKeySpec keySpec = new X509EncodedKeySpec(keyBytes);
KeyFactory keyFactory = KeyFactory.getInstance("RSA");
PublicKey pubKey = keyFactory.generatePublic(keySpec);
Cipher cipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");
cipher.init(Cipher.ENCRYPT_MODE, pubKey);
return cipher.doFinal(data);
```

APP网络协议封包普遍存在弱加密易解密的主要原因是：Key/IV/CER等密钥的硬编码。

【数据封包弱加密案例】

1

某金融APP，IV向量硬编码

```
package com.yirendai.util;

import javax.crypto.Cipher;

public class ai
{
  private static byte[] a = { 1, 2, 3, 4, 5, 6, 7, 8 };

  public static String a(String paramString1, String paramString2)
  {
    new IvParameterSpec(a);
    SecretKeySpec localSecretKeySpec = new SecretKeySpec(paramString2.getBytes(), "DES");
    Cipher localCipher = Cipher.getInstance("DES/ECB/PKCS5Padding");
    localCipher.init(1, localSecretKeySpec);
```

```
.prologue
.line 36
const-string v0, "yrdAppKe"
```

加密key也是硬编码

```
sput-object v0, Lcom/yirendai/ui/lockPattern/SetLocusPassWord
```

```
this.iv = new byte[]{10, 1, 11, 5, 4, 15, 7, 9, 23, 3, 1, 6, 8, 12, 13, 91};
String v5 = "shenzhoucar123123";
byte[] v0 = new byte[16];
try {
    byte[] v4 = v5.getBytes("UTF-8");
    int v2;
    for(v2 = 0; v2 < v5.length(); ++v2) {
        if(v2 >= v0.length) {
            break;
        }

        v0[v2] = v4[v2];
    }

    IvParameterSpec v3 = new IvParameterSpec(this.iv);
    this.sKey = new SecretKeySpec(v0, "AES");
    this.eCipher = Cipher.getInstance("AES/CBC/PKCS7Padding");
    this.eCipher.init(1, this.sKey, ((AlgorithmParameterSpec)v3));
```

某租车APP AES的加密key和向量是硬编码

2

```
private static Key a()
{
  if (a == null) {
    a = new SecretKeySpec("CSIIQZBk".getBytes("UTF-8"), "DES");
  }
  return a;
}

public static String b(String paramString)
```

苏宁某金融APP des硬编码

3

【案例】铁路12306通信加解密破解

漏洞概要

缺陷编号： WooYun-2016-190761

漏洞标题： 铁路

相关厂商： 123

漏洞作者： cha

提交时间： 201

公开时间： 201

一、 https未校验证书导致数据被捕获（中间人攻击风险）

客户端虽采用了https，但并未对服务端证书进行校验能够捕获通讯数据，可导致中间人攻击。

```
C:\dex\12306_test\classes\com\lidroid\xutils\http\client\DefaultSSLSocketFactory.smali (1 hit)
  Line 113:     sget-object v1, Lorg/apache/http/conn/ssl/SSLSocketFactory;->ALLOW_ALL_HOSTNAME_VERIFIER:Lorg/apache/http/conn/ssl/X509HostnameVerifier
C:\dex\12306_test\classes\com\lidroid\xutils\util\OtherUtils.smali (1 hit)
  Line 712:     sget-object v3, Lorg/apache/http/conn/ssl/SSLSocketFactory;->ALLOW_ALL_HOSTNAME_VERIFIER:Lorg/apache/http/conn/ssl/X509
```


信任所有证书

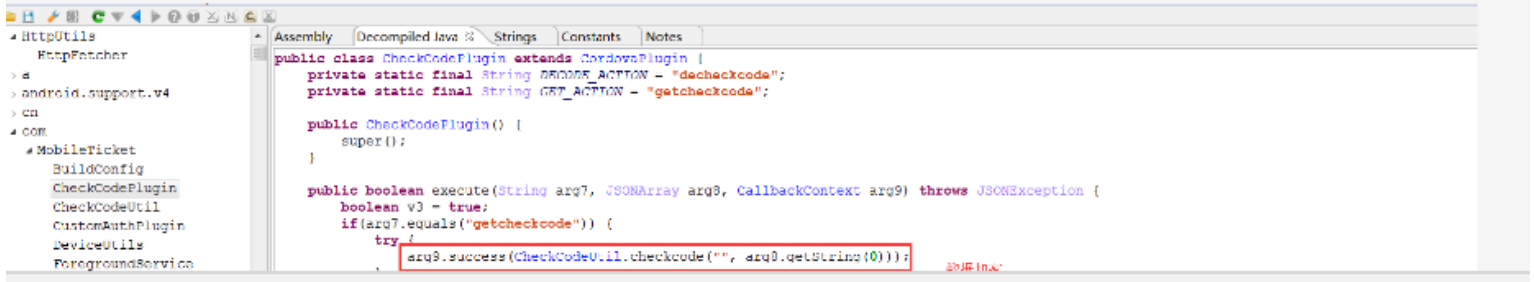
可以正常截获通讯数据

二、通信数据解密

铁路12306通信加解密主要通过checkcode、decheckcode两个函数进行通信加解密，虽然采用加固方式，但是

于暴露。

dex部分：



3.手机验证码机制安全

就目前来看，手机验证码可谓是一个账号的命脉，几乎贯穿着一个账号的注册、登录、找回密码、修改密码、绑定银行卡、支付确认、设备认证等所有的敏感操作过程，对于其安全机制，可以从以下几个方面着手：

验证码是否可无限请求	• 如果验证码可无限制的请求发送，则有可能会被利用成为短信轰炸接口，浪费资源；
验证码是否可重复使用	• 验证码大有效次数只一次；且新验证码要覆盖旧验证码；
验证码是否是真正随机	• 验证码在后台服务器与任何字符串无任何的对应关系，实时使用实时随机；
验证码是否返回到本地	• 这里不仅仅是验证码本身，而是只所有具有验证码凭证性质的数据都不能返回在本地，比如验证码的编码/加密形式、验证码对应的固定Code等；
验证码的错误次数是否限定	• 防止位数较短的验证码被恶意爆破；
验证是否可被绕过	• 验证码只在当下某一个步骤有验证、阀门的作用，但在注册、找回密码的最终请求中抛弃了验证码，这时候就可能存在绕过。

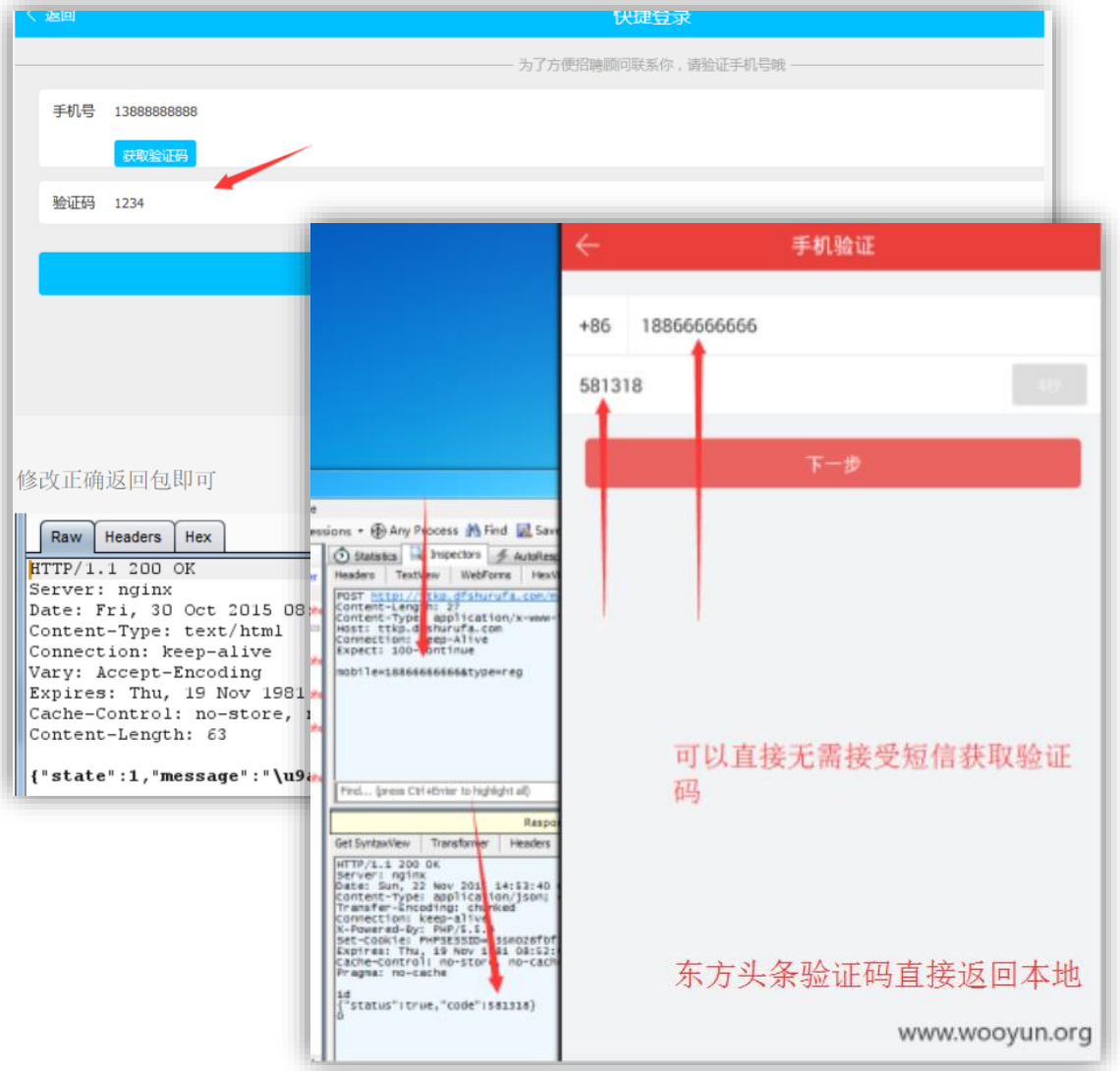
【手机验证码机制安全案例】

标题	微信公共号注册获取短信接口无任何限制
漏洞类型	WEB漏洞 逻辑漏洞
漏洞级别	中
站点URL	http://wqs.jd.com/my/register.shtml
利用参数	/user/smsmsg/SendMobileMsg
Payload	Replay
漏洞说明	通过微信公共号注册京东商城的账号时，发送短信的接口连接无限制，可无限replay同

京东某微信公共号注册验证码可爆破

浪费和短信轰炸，调整下吧。。。

苏宁某系统手机验证码形同虚设
直接绕过进行无限制注册



7.手势密码的那些坑

保存在本地

可能造成手势密

- Activity组件

- 手势密码的

就算是MD

才389112中情

- 判断逻辑混

典型案例：

- 修改手势密

电信翼支付

有

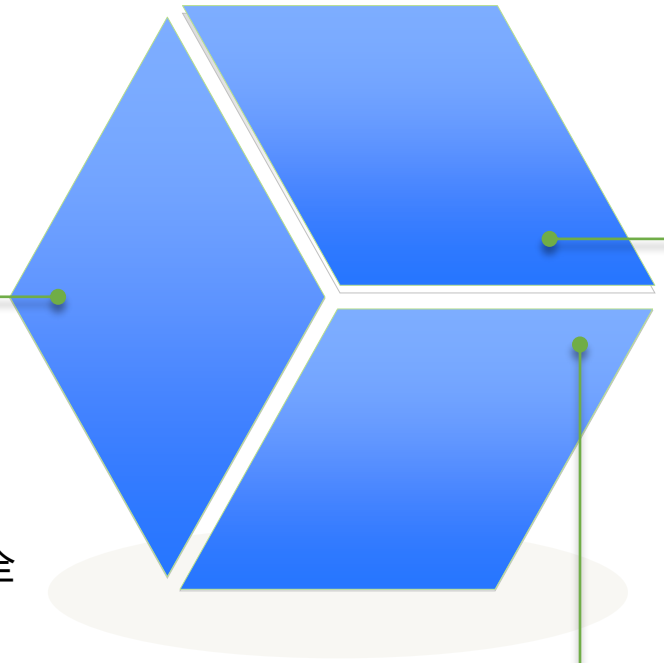
正确使用手势密码的姿势

1. 全局设置手势密码验证的FLAG，任何界面启动前都对这个FLAG进行验值，验证不通过就重新调到手势密码验证界面，这样无论是Activity exported了，还是su权限基本都绕不过去。

2. 当设置、修改手势密码验证登录密码的情况，要对服务器返回的登录结果做完全校验，设置检测下登录返回的session等标识是否改变等等。

3. 我认为手势密码利用最好的情况：手势密码作为用户登陆的第二密码、保存在服务器上、有登录请求的试错限制、有设备绑定的限制（这个主要是防范修改手势密码无任何校验的情况）。目前只发现两款：工商银行、苏宁的任性付。

2.2 iOS客户端常见漏洞



程序安全

- iPA砸壳解包分析
- 源码头Dump安全
- 系统Log日志安全
- Mach-o程序源代码安全

数据安全

- 敏感界面明文显示截屏安全
- 界面切换没有缓存模糊处理
- 本地数据储存安全

动态调试安全

- Intropy动态追踪分析
- Cycript调试程序

第一大类：程序安全

- iPA砸壳解包分析
- 源码头Dump安全
- 系统Log日志安全
- Mach-o程序源代码安全

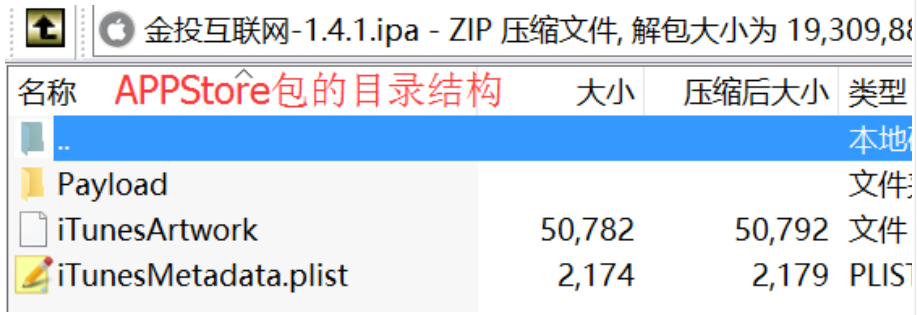
1. iPA砸壳解包分析

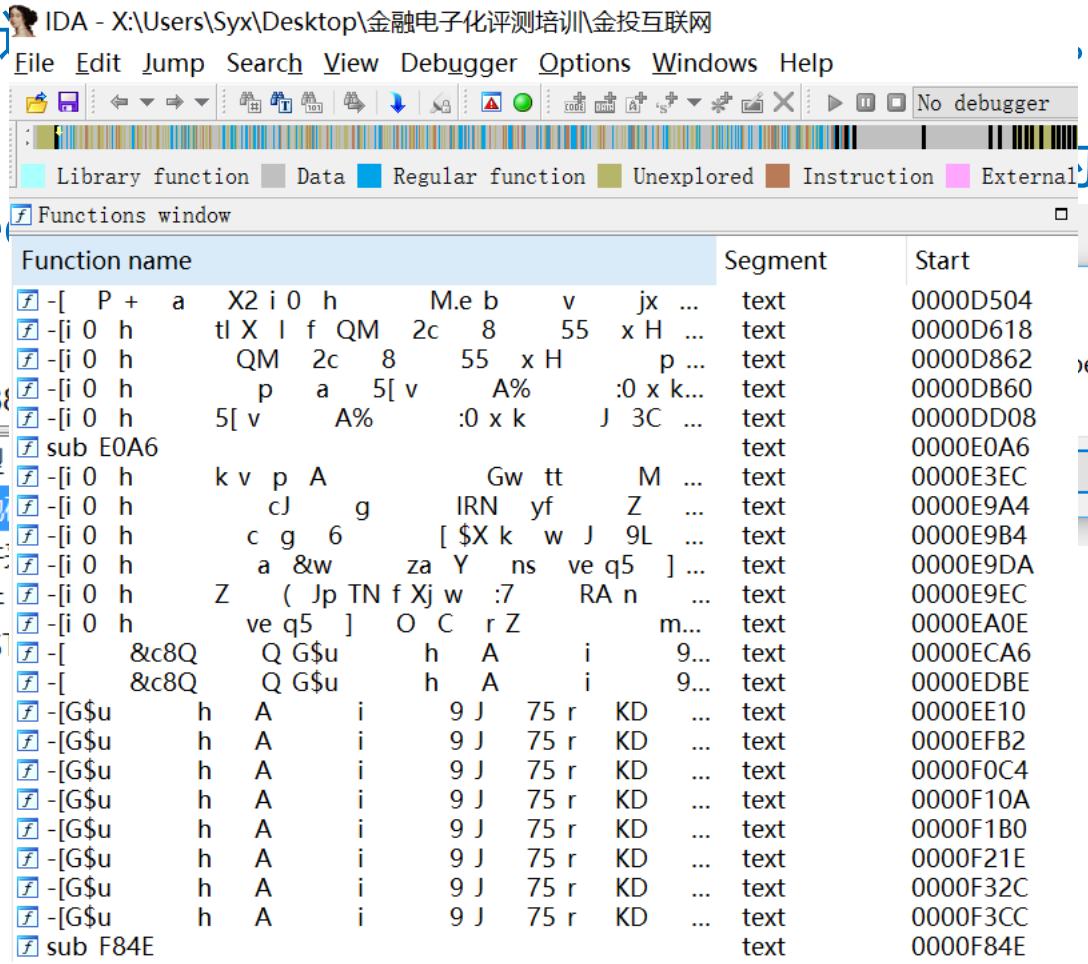
砸壳原因：

Apple公司为了保护开发者原创不被窃取，维护其运营生态，每款iOS APP上架成功后，都会在外面加一层“保护壳”，

但同时又由于系统运行解密的机制

破解工具，如Clutch、dumpdecrypte



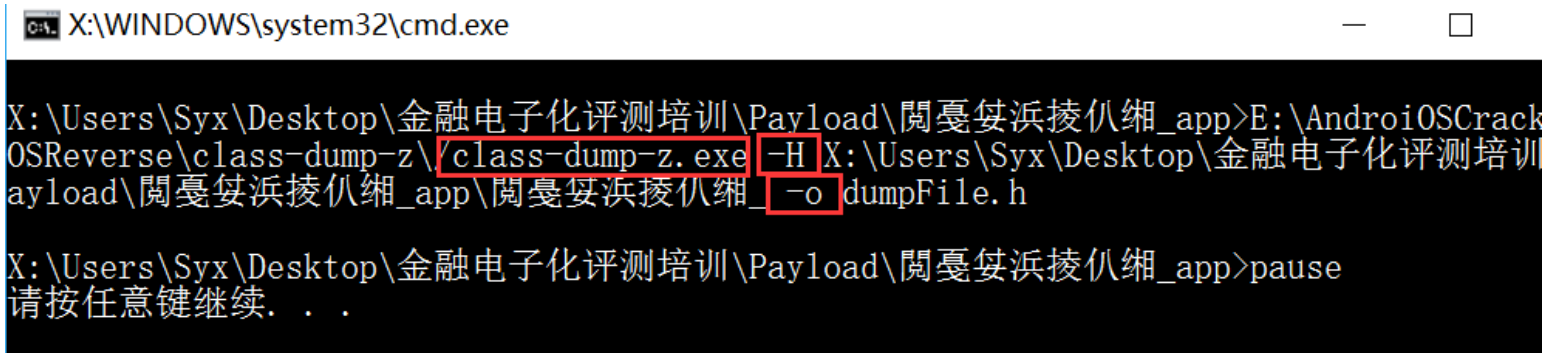


2.源码头Dump安全

iOS程序iPA使用的是O-bject语言进行编写，像apk一样包含了各种类、方法、字段、控件等，通过class-dump可以将已经砸壳的iPA程序中的类名、方法名、字段名称以及他们之间的相关继承、调用关系都罗列出来，极大的方便了逆向分析。

Dump源码头的工具叫class-dump-z，dump Mach-o文件的指令：

class-dump-z.exe -H <Mach-o文件> -o <文件夹>



Dump得到的文件

生成很多的.h类文件

CDVPictureOptions.h	2016/11/22 16:56	JetBrains Cl
CDVPlugin.h	2016/11/22 16:56	JetBrains Cl
CDVPluginResult.h	2016/11/22 16:56	JetBrains Cl
CDVReachability.h	2016/11/22 16:56	JetBrains Cl
CDVScreenOrientationDelegate.h	2016/11/22 16:56	JetBrains Cl
CDVSplashScreen.h	2016/11/22 16:56	JetBrains Cl
CDVTimer.h	2016/11/22 16:56	JetBrains Cl
CDVTimerItem.h	2016/11/22 16:56	JetBrains Cl
CDVURLProtocol.h	2016/11/22 16:56	JetBrains Cl
CDVUserAgentUtil.h	2016/11/22 16:56	JetBrains Cl
CDVViewController.h	2016/11/22 16:56	JetBrains Cl
CDVWebViewDelegate.h	2016/11/22 16:56	JetBrains Cl
CDVWeixin.h	2016/11/22 16:56	JetBrains Cl

每个.h类中的控件、方法类、字段

```
@interface MainViewController : CDVViewController {
}
-(void)webViewDidFinishLoad:(id)webView;
-(BOOL)shouldAutorotateToInterfaceOrientation:(int)interfaceOrientatio
-(void)viewDidUnload;
-(void)viewDidLoad;
-(void)viewWillAppear:(BOOL)view;
-(void)didReceiveMemoryWarning;
-(id)init;
-(id)initWithNibName:(id)nibName bundle:(id)bundle;
@end
```

一些相关的方法

```
@interface JPFClient : XXUnknownSuperclass {
    NSTimer* _deviceTokenStatusCheckTimer;
    int _deviceTokenStatusCheckTimes;
    double _becomeActiveTime;
    BOOL _isSimulator;
    BOOL _deviceTokenUpdated;
    BOOL _isReportCrash;
    NSURL* _emProvisionUrl;
    NSURL* _pushConfigUrl;
    NSString* _systemVersion;
    NSString* _bundleIdentifier;
    NSNumber* _bundleVersion;
    NSString* _modelName;
```

变量定义

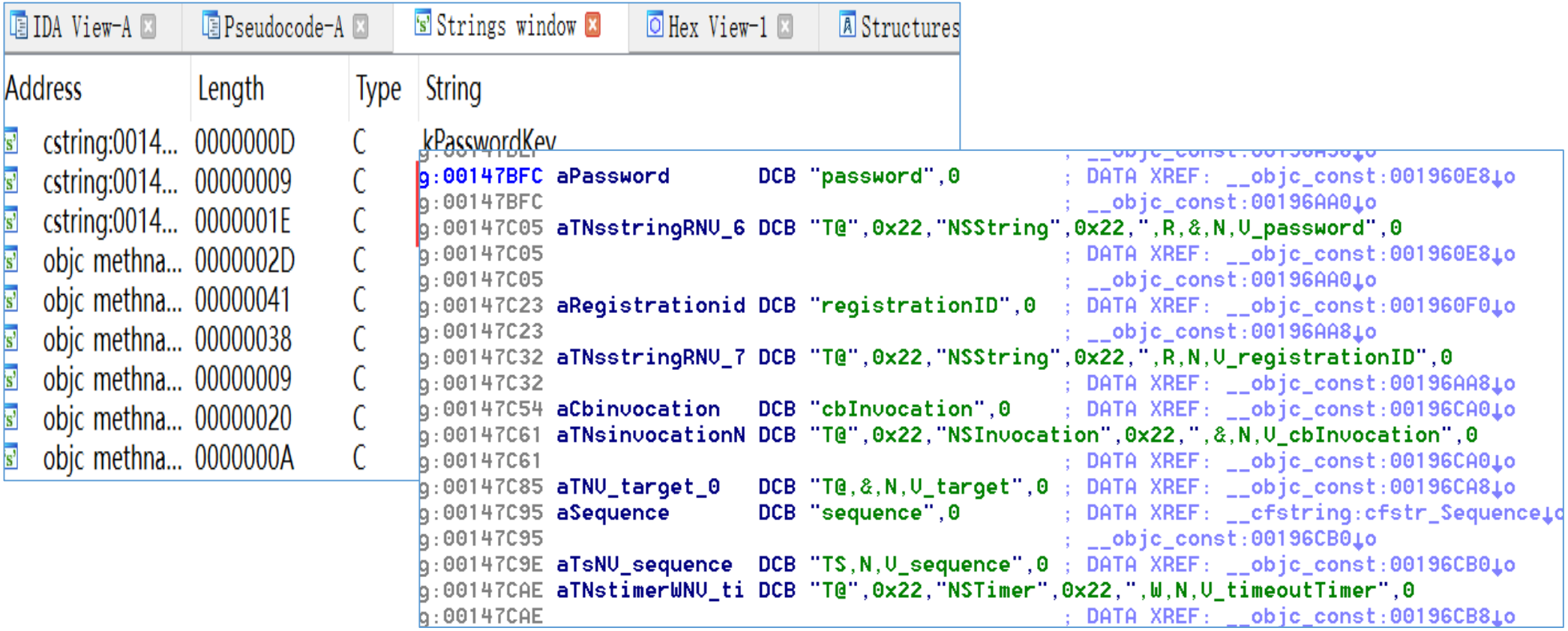
字符串

3.系统Log日志

在APP的开发过程中，为了方便调试，通常会使用log函数输出一些关键流程的信息，这些信息中通常会包含敏感内容，如执行流程、明文的用户名密码等，这会让攻击者更加容易的了解APP内部结构方便破解和攻击，甚至直接获取到有价值的敏感信息。

```
金投互联网[26242] <Warning>: load push config error!
金投互联网[26242] <Warning>: retryHandleOpenURL
金投互联网[26242] <Warning>: localURI:wap/x5/UI2/v-mqmq6b-zh_CN-/kifpappui/themes/shanxi/index.w
金投互联网[26242] <Warning>: 本地存在文件:/private/var/mobile/Containers/Bundle/Application/56785
金投互联网[26242] <Warning>: localURI:wap/x5/UI2/v-mqmq6b-zh_CN-/kifpappui/themes/shanxi/index.w
金投互联网[26242] <Warning>: 本地存在文件:/private/var/mobile/Containers/Bundle/Application/56785
金投互联网[26242] <Warning>: Resetting plugins due to page load.
金投互联网[26242] <Warning>: localURI:wap/x5/UI2/v-mqmq6b-zh_CN-/kifpappui/themes/shanxi/index.w
金投互联网[26242] <Warning>: 本地存在文件:/private/var/mobile/Containers/Bundle/Application/56785
金投互联网[26242] <Warning>: http://117.78.36.87:8001/wap/x5/UI2/v-mqmq6b-zh_CN-/kifpappui/themes/
金投互联网[26242] <Warning>: 本地存在文件:/private/var/mobile/Containers/Bundle/Application/56785
金投互联网[26242] <Warning>: extension.w
金投互联网[26242] <Warning>: stoploading!
```

源码字符串暴露


IDA View-A
Pseudocode-A
Strings window
Hex View-1
Structures
Address
Length
Type
String
cstring:0014... 0000000D C kPasswordKey
cstring:0014... 00000009 C
cstring:0014... 0000001E C
objc methna... 0000002D C
objc methna... 00000041 C
objc methna... 00000038 C
objc methna... 00000009 C
objc methna... 00000020 C
objc methna... 0000000A C
g:00147BFC aPassword DCB "password",0 ; DATA XREF: __objc_const:001960E8↓o
g:00147BFC ; __objc_const:00196AA0↓o
g:00147C05 aTNsstringRNU_6 DCB "T@",0x22,"NSString",0x22,",R,&,N,V_password",0
g:00147C05 ; DATA XREF: __objc_const:001960E8↓o
g:00147C05 ; __objc_const:00196AA0↓o
g:00147C23 aRegistrationid DCB "registrationID",0 ; DATA XREF: __objc_const:001960F0↓o
g:00147C23 ; __objc_const:00196AA8↓o
g:00147C32 aTNsstringRNU_7 DCB "T@",0x22,"NSString",0x22,",R,N,V_registrationID",0
g:00147C32 ; DATA XREF: __objc_const:00196AA8↓o
g:00147C54 aCbinvocation DCB "cbInvocation",0 ; DATA XREF: __objc_const:00196CA0↓o
g:00147C61 aTNsinvocationN DCB "T@",0x22,"NSInvocation",0x22,",&,N,V_cbInvocation",0
g:00147C61 ; DATA XREF: __objc_const:00196CA0↓o
g:00147C85 aTNV_target_0 DCB "T@,&,N,V_target",0 ; DATA XREF: __objc_const:00196CA8↓o
g:00147C95 aSequence DCB "sequence",0 ; DATA XREF: __cfstring:cfstr_Sequence↓o
g:00147C95 ; __objc_const:00196CB0↓o
g:00147C9E aTsNV_sequence DCB "TS,N,V_sequence",0 ; DATA XREF: __objc_const:00196CB0↓o
g:00147CAE aTNstimerWNV_ti DCB "T@",0x22,"NSTimer",0x22,",W,N,V_timeoutTimer",0
g:00147CAE ; DATA XREF: __objc_const:00196CB8↓o

第二大类：数据安全

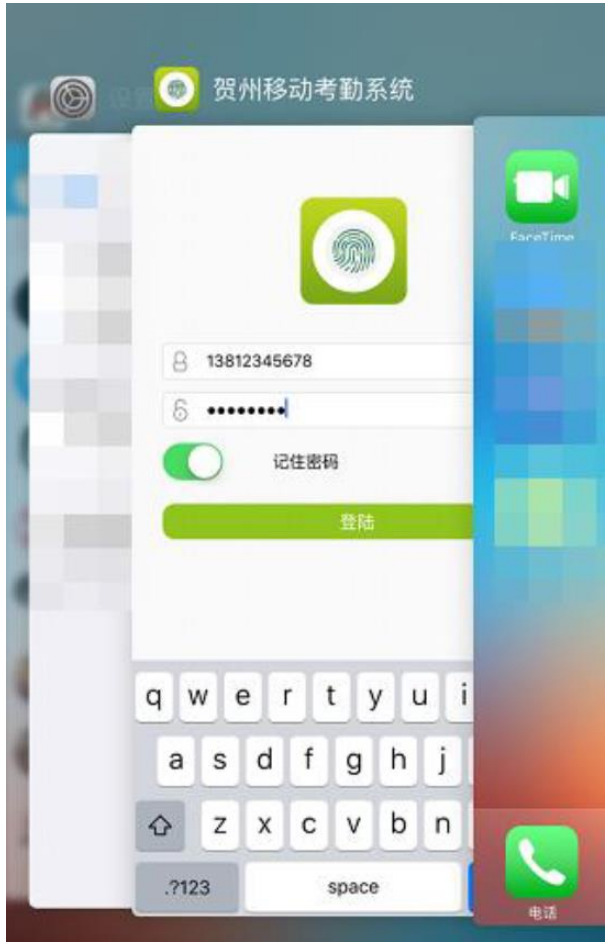
- 敏感界面明文显示截屏安全
- 界面切换缓存没有模糊处理
- 本地数据储存安全

1.敏感界面明文显示截屏安全

登录、注册、找回密码、手势密码、支付等敏感界面截图操作

2.界面切换缓存没有模糊处理

将APP 切换至后台，检测APP 的界面是否做模糊处理。



有些程序在登录界面输入信息切换入后台，

可以清晰的看到信息，如左图：

3.本地数据储存安全

三个检测项：

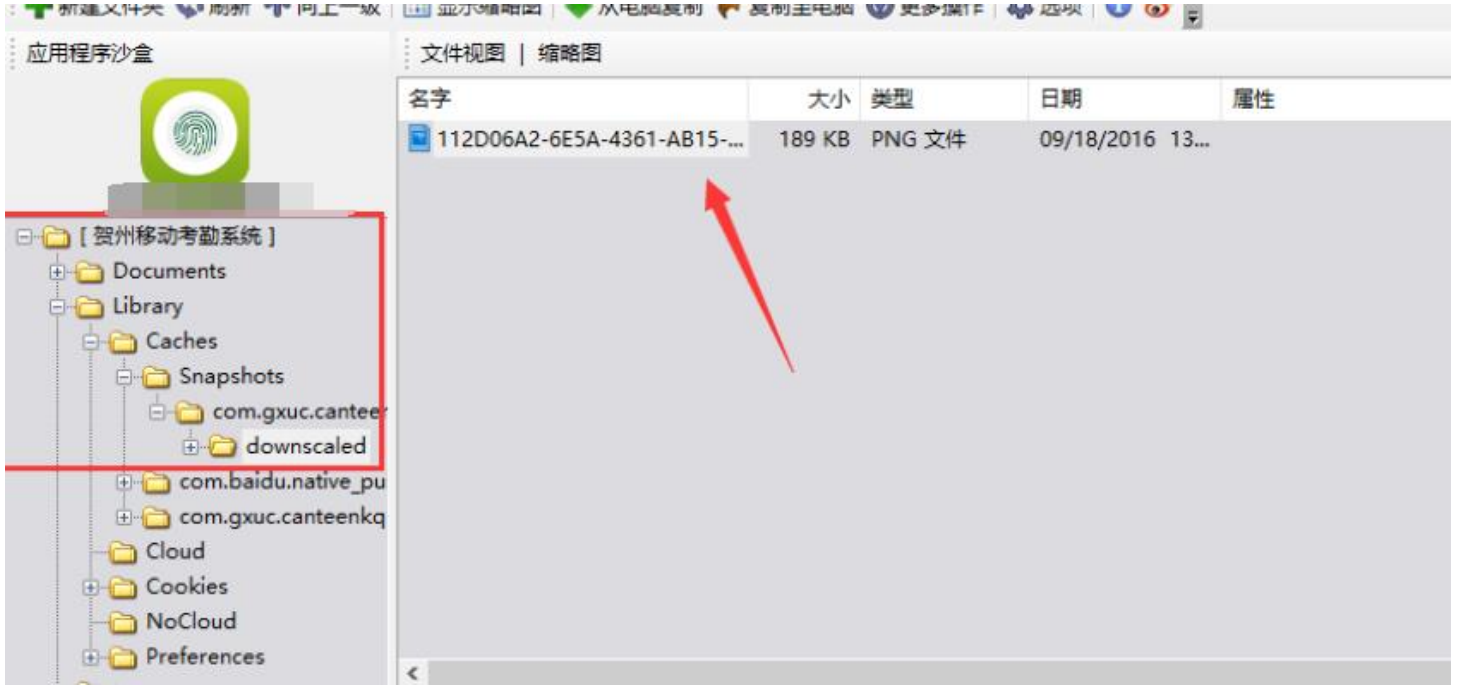
- 在plist 文件中明文存储敏感信息
- 在SQLite 数据库中明文存储敏感信息
- 缓存文件中存在其他敏感信息

```
33      <true/>
34      <key>WebKitStoreWebDataForBackup</key>
35      <true/>
36      <key>password</key>
37      <string>jc_11898</string>
38      <key>username</key>
39      <string>13712345678</string>
40  </dict>
41  </plist>
42
```

```
<key>mainPageClassName说明</key>
<string>HomePageView9Controller前端定制首页风格,HomePageView6Controller后台定制6宫格首页风格，NewsPageController新闻列表</string>
<key>member_if</key>
<string>http:/uc.nfapp.southcn.com/amuc/api/member</string>
<key>member_if说明</key>
<string>登录URL 测试：http://183.63.143.99:8080/amuc/api/member|||正式：http:/uc.nfapp.southcn.com/amuc/api/member</string>
<key>server_if</key>
<string>http://183.63.143.167/nanfang_if</string>
<key>server_if 说明</key>
<string>正式http://api.nfapp.southcn.com/nanfang_if测试http://183.63.143.97:8080/nanfang_if预发布环境: http://api.nfapp.southcn.com:8080/nanfang_if阿里http://112.74.141.80:80/nanfang_if 开发调试环境：http://120.24.40.162:8080/nanfang_if</string>
<key>sina_weibo_name</key>
<string>新闻客户端</string>
<key>tengxun_weibo_name</key>
<string>新闻客户端</string>
```

缓存在本地的信息

查看客户端缓存文件，可以看到存储的图片等信息：



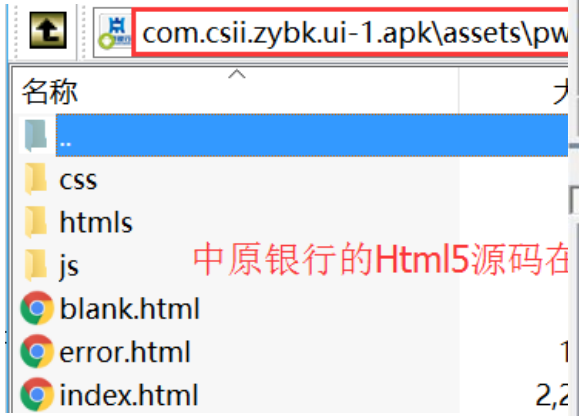
2.3 Html5 常见漏洞

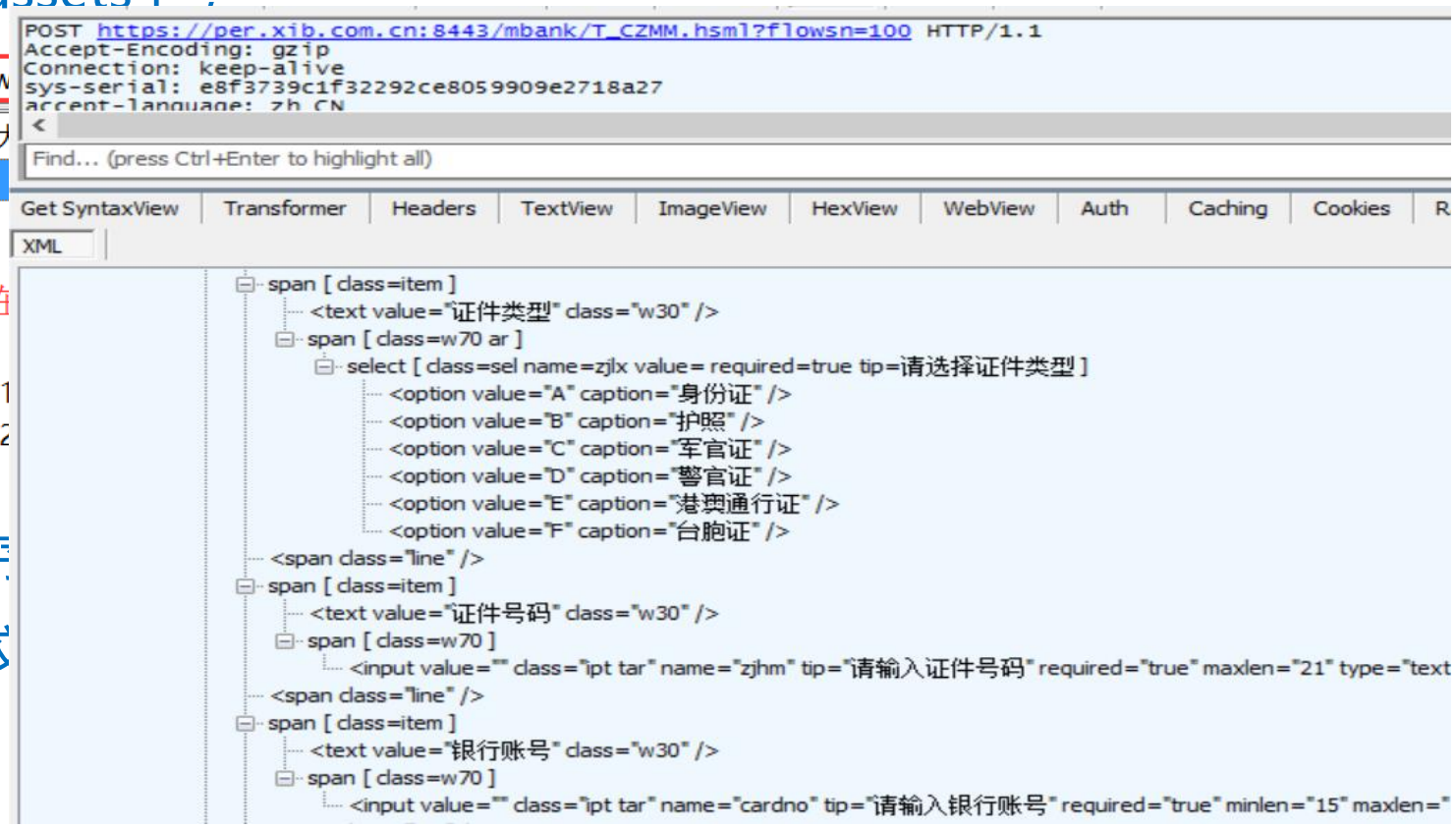
- Html5源码安全
- 会话保持机制安全
- 请求数据篡改测试
- 链接重放攻击
- 敏感信息泄露安全

2.3.1 Html5源码安全

目前流行的Html5页面源码的保存有以下两种形式：

- 客户端通过WebView控件展示Html5页面，并处理相应逻辑，该种情况Html5源码一般放在客户端本地的assets下：



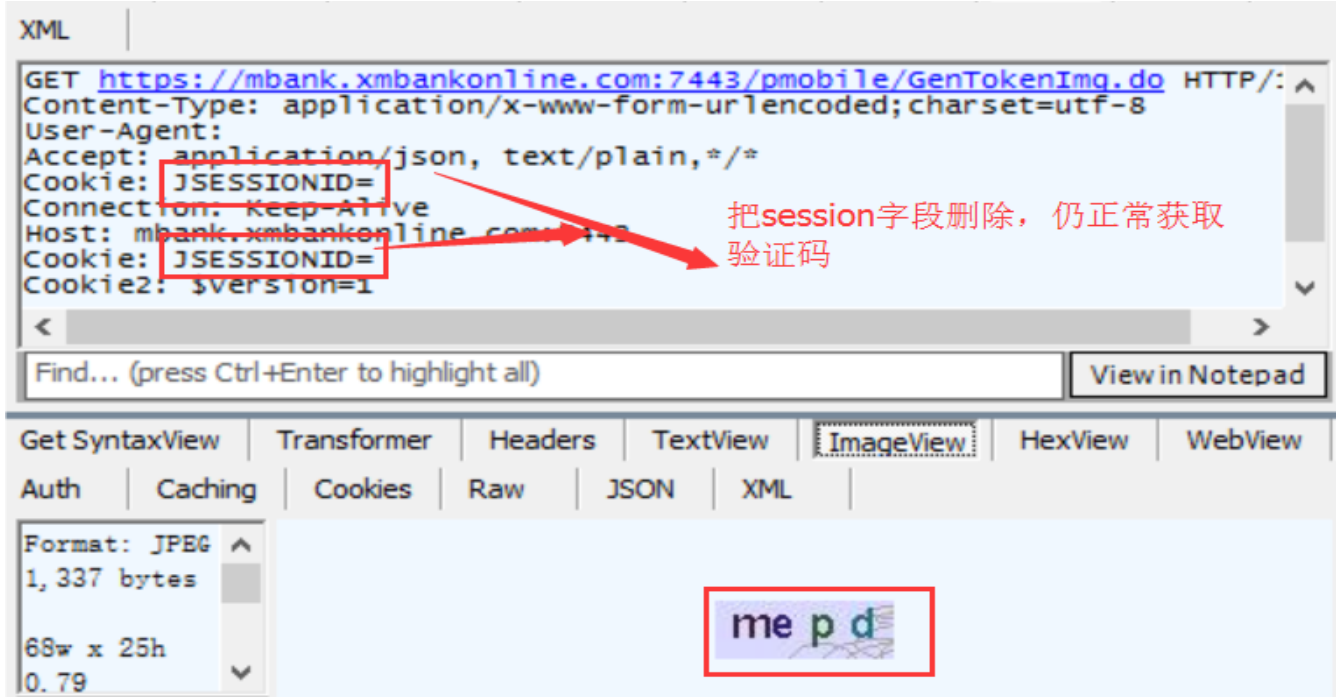


- 在一些敏感页面、小程序可以与客户端函数进行交

2.3.2 会话保存机制安全

这部分主要是检测用户输入账号密码登录成功后，是怎么进行会话保持的：

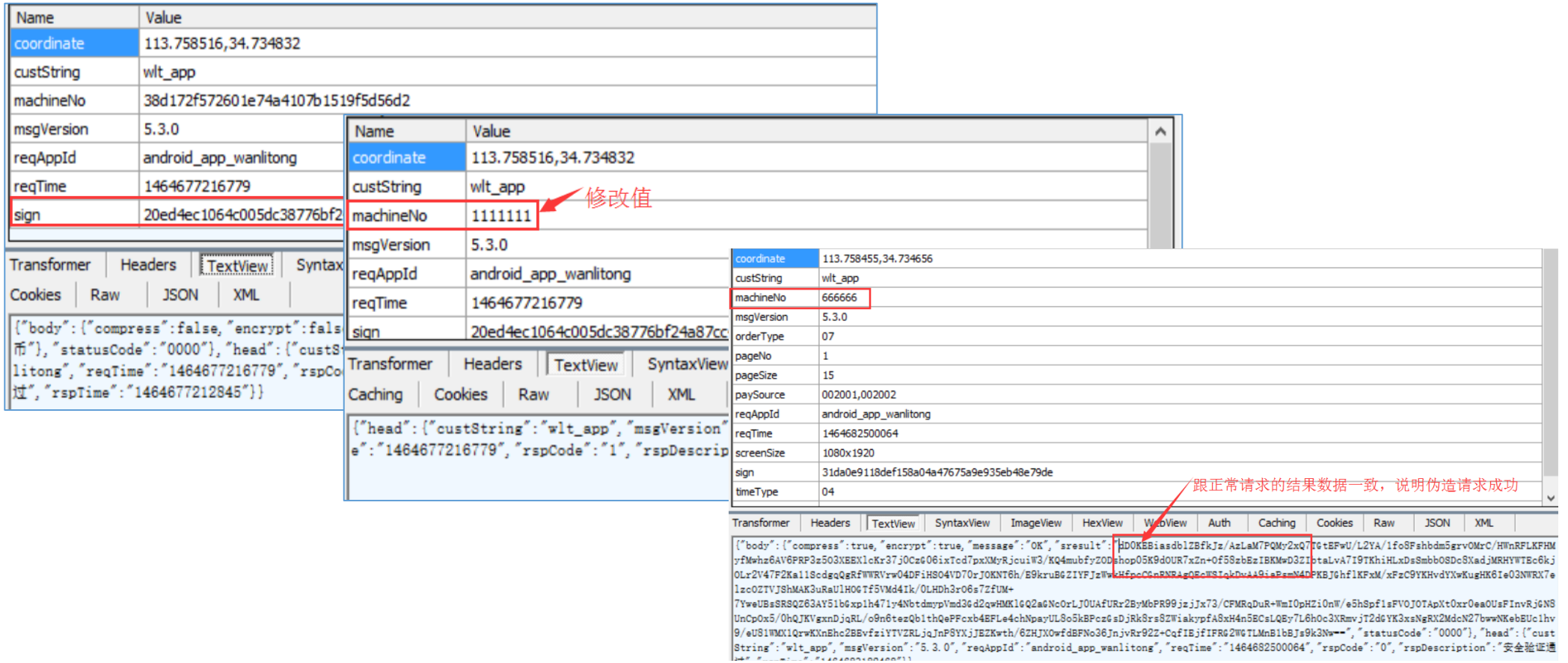
- 检测客户端登录后，是否有Cookie、Jsession、Session、Token等对登录身份的严格校验机制；



- 会话保持机制是否安全。

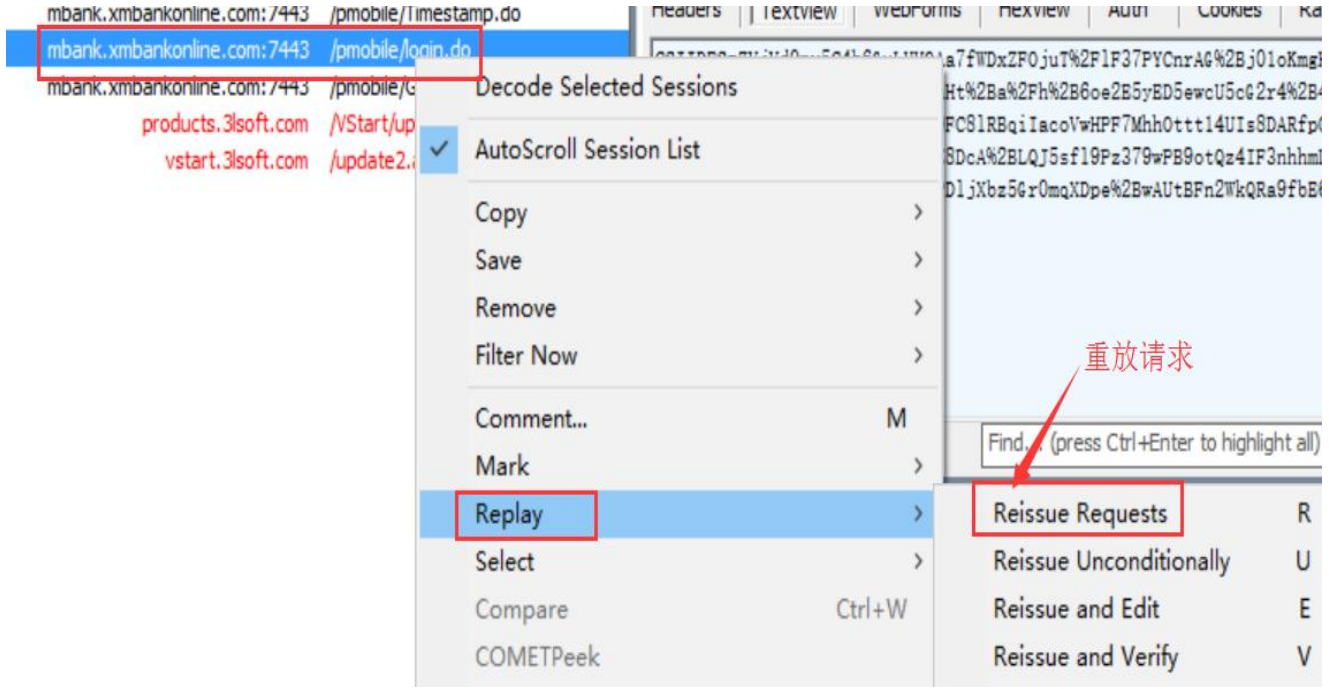
2.3.3 请求数据篡改测试

检测交易数据被非法篡改后，服务器是否能够检测到，而拒绝服务。



2.3.4 链接重放攻击

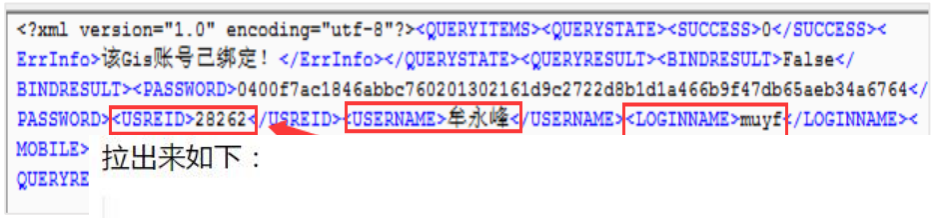
对同一个链接向服务器重复发送多遍，检测服务器的处理策略



2.3.5 敏感信息泄露安全

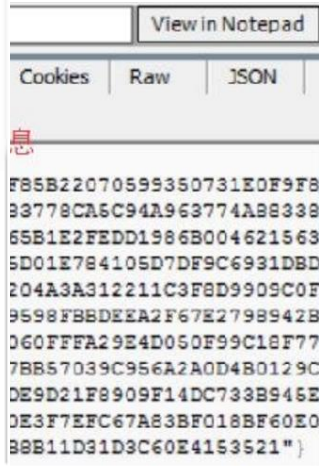
登录过程中会做很多的验证，比如：

- 登陆请求时会首先验证用户输入的账户名是否存在。如果账户存在，服务器就可能返回该用户的个人信息，造成信息泄露；
- 即使由于设备绑定不成功，造成登录失败，服务器也可能返回敏感的个人信息。



拉出来如下：

{"responseCode":"0000","responseMsg":"success","userInfoList":[{"isExistLoginPwd":"false","userAlias":"15093159098","pubSecStatus":"4","freezeStatus":"0","aliasType":"2","userIconURL":"","occupation":"销售/采购等行业","city":"郑州","userNo":"0000000000123858955","district":"管城回族区","province":"河南","registerTime":"2015-11-13
11:59:54","isCompleteInfo":true,"address":"宝景花园","authFlag":"2","custNo":"6131534353","userInfoFullFlag":"1","idType":"131000000010","userName":"宋煜禧","idNo":"412725199101127014","bindMobile":"15093159098"}]}



2.3.5 敏感信息泄露安全

“论坛交流”板块，各个账户之间进行交互时，服务器可能会返回多余的个人信息，导致其他用户标识暴露，再加以伪造，可能会造成任意账户登录风险。

第一小节：开发阶段防护。1.客户端开发时

AndroidManifest安全配置

- allowBackup备份设置为false
- debuggable调试模式设置为false
- 不必要的导出的组件export属性设置为false

```
<application
    android:allowBackup="false"
    android:debuggable="false"
```

本地数据储存

- 对于特别敏感的用户信息、银行信息、实名信息不要保存在本地
- 本地保存的数据要加密

```
String account = edtAccount.getText().toString();编辑框获取账号密码
String password = edtPassword.getText().toString();
try{
    account = AESEncryptor.encrypt(keystore, account);AES加密
}catch(Exception ex){
    Toast.makeText(this, "给密码加密时产生错误!", Toast.LENGTH_SHORT);
    password = "";
}
//获取名字为“MY_PREFERENCES”的参数文件对象。
SharedPreferences sp = this.getSharedPreferences(MY_PREFERENCES, 0
//使用Editor接口修改SharedPreferences中的值并提交。
Editor editor = sp.edit();
editor.putString(MY_ACCOUNT, account);储存加密后的账号
```

1.客户端开发时

截屏安全

- 对于用户参与的登录、注册、找回密码、支付等要输入敏感密码的界面防止被恶意截图

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

防止截屏FLAG

界面劫持安全

- 对于用户参与的登录、注册、找回密码、支付等要输入敏感密码的界面防止被恶意切换钓鱼。

```
ActivityManager activityManager = (ActivityManager) getSystemService(Context.ACTIVI
// getRunningTasks会返回一个List，List的大小等于传入的参数。
// get(0)可获得List中的第一个元素，即栈顶的task
ActivityManager.RunningTaskInfo info = activityManager.getRunningTasks(1).get(0);
// 得到当前栈顶的类名，按照需求，也可以得到完整的类名和包名
String shortClassName = info.topActivity.getShortClassName(); // 类名
Toast.makeText(getApplicationContext(), "当前运行的程序为" + shortClassName, Toast.LENG
```

检测运行的顶层界面反劫持

2.apk的签名信息校验

当一个程序包被黑客反编译、篡改、重打包后，apk的签名信息一定是发生改变，利用这一特性，开发者可以对正在运行的程序包进行正盗版的校验。

Java获取签名信息

```
// 获取签名信息
private String getSign(Context Ctx) {
    // TODO Auto-generated method stub
    String sign = "";
    try {
        PackageManager pm = Ctx.getPackageManager();
        PackageInfo info = pm.getPackageInfo(Ctx.getPackageName(), pm.GET
        Signature[] sig = info.signatures;
        sign = sig[0].toCharsString();
    } catch (Exception e) {
        // TODO: handle exception
    }

    return sign;
}
```

JNI反射获取签名信息

```
//获取context->activity
jobject context = getApplication(env);
jclass activity = env->GetObjectClass(context);
// 得到 getPackageManager 方法的 ID
jmethodID methodID_func = env->GetMethodID(activity, "getPackage
        "()Landroid/content/pm/PackageManager;");
// 获得PackageManager对象
jobject packageManager = env->CallObjectMethod(context, methodID
jclass packageManagerclass = env->GetObjectClass(packageManager)
//得到 getPackageName 方法的 ID
jmethodID methodID_pack = env->GetMethodID(activity, "getPackage
        "()Ljava/lang/String;");
//获取包名
jstring name_str = static_cast<jstring>(env->CallObjectMethod(co
        methodID_pack));
// 得到 getPackageInfo 方法的 ID
jmethodID methodID_pm = env->GetMethodID(packageManagerclass,
        "getPackageInfo",
        "(Ljava/lang/String;I)Landroid/content/pm/PackageInfo;")
// 获得应用包的信息
jobject package_info = env->CallObjectMethod(packageManager, met
        name_str, 64);
// 获得 PackageInfo 类
jclass package_infoclass = env->GetObjectClass(package_info);
```

3.apk的文件校验

当一个程序包被黑客反编译、篡改、重打包后，不仅仅只有签名信息发生了改变，还有重新编译的classes.dex、xml、修改过的so/dll、apk的整体包等，这些都能成为判断包体是否经过修改的依据。

关键词

- sourceDir
- getPackageCodePath
- getPackageResourcePath

```
File v1 = new File(bl.m(this.a).applicationInfo.sourceDir);
Signature[] v2 = bl.m(this.a).signatures;
int v3 = v2.length;
int v0_1;
for(v0_1 = 0; v0_1 < v3; ++v0_1) {
    Signature v4 = v2[v0_1];
    bl.a(this.a, v4.toByteArray());
    v4.toCharsString();
    bl.b = ba.a("3082024b308201b4a003020102020454d56bbf300d06092a864
}

ba.a(v1);
```

获取apk文件路径

签名信息

取apk整体包的MD5值

```
public static long b(Context arg2) {
    return new File(arg2.getApplicationContext().getPackageResourcePath()).length();
}
```

apk的大小

【案例展示】

① sign签名信息校验

② 整体包MD5校验

③ classes.dex/so/dll/META-INF等文件校验。


```
if (!str2.equals(str1)) && (!Config.DEBUG)
{  在签名信息不一致，并且不是DEBUG模式下盗版提示
  showTip("您的软件有盗版风险，请卸载后从正规途径重新下载！")
  finish();
  return;
}
this.mHandler.postDelayed(this.enterHome, 1000L);
```

第二小节：服务端做防护。1.网络接口开发时

■ 防止重放攻击

网络接口的重放攻击是指客户端的某个封包，可以连续一直发送，服务器并不做异常反应。特别是对于比较敏感的网络请求，比如登录、注册、修改密码等，如登录接口可重放，登录请求一旦泄露一次，就可能被别人一直可登录。

登录之前向服务器索要一个随机生成的登录序号→登录封包中添加该序号，服务器校验通过

■ 延时保护策略

对于大多数APP客户端使用的是HTTP/HTTPS的协议进行传输，所以在传输过程中难免会被中间人截包工具，对修改密码、支付过程等截包、分析、篡改、重发，操作之间必有发包到服务器接包的时间戳较大差别，记录此特征，服务器可拒绝返回正常服务。

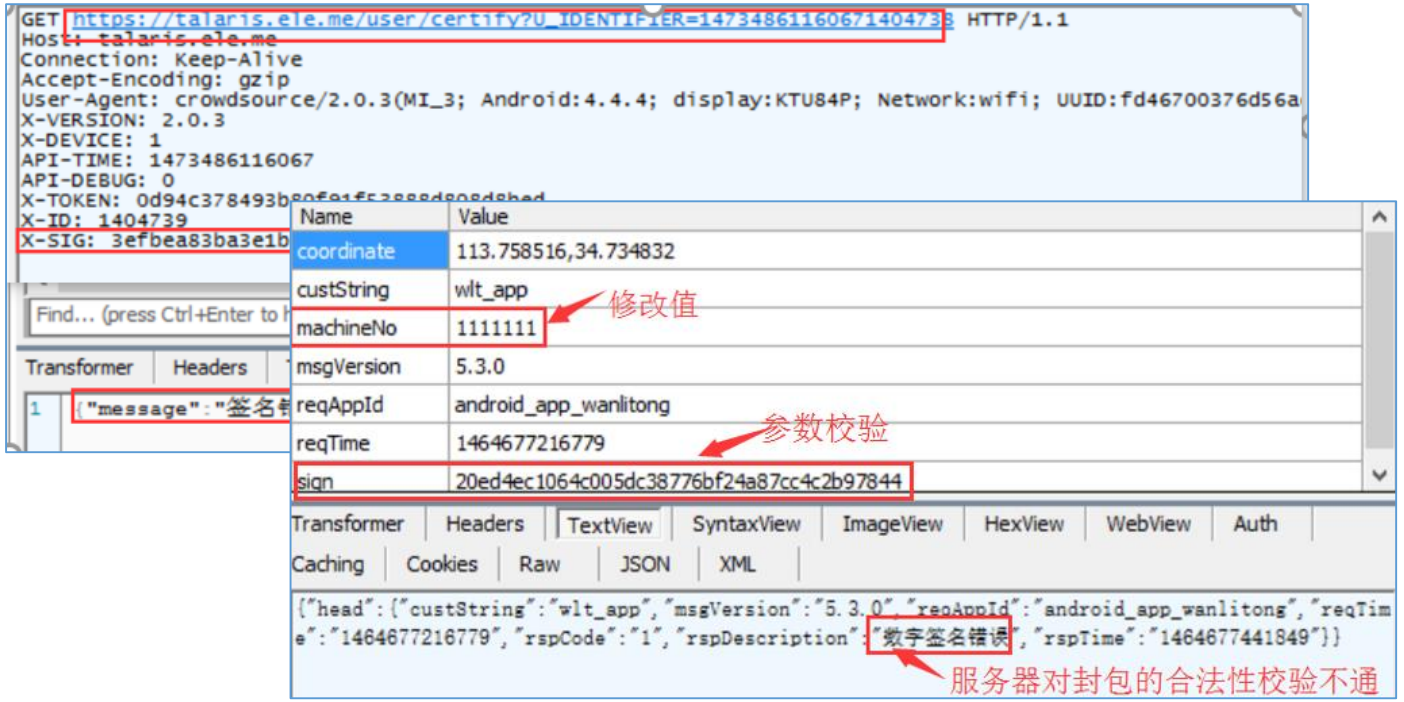
封包内加时间戳参数并加密→服务器对发包时间进行校验

2.封包参数加Sign防篡改

客户端的BS结构决定了网络接口请求安全的重要性，一旦接口被破译，所有的请求就能实现脱离程序本身来进行各种实质性的业务操作。

➢ 应用程序

- 恶意注册
- 频繁登录
- 刷积分
- 模拟请求
- ……



3.包体的反编译处理

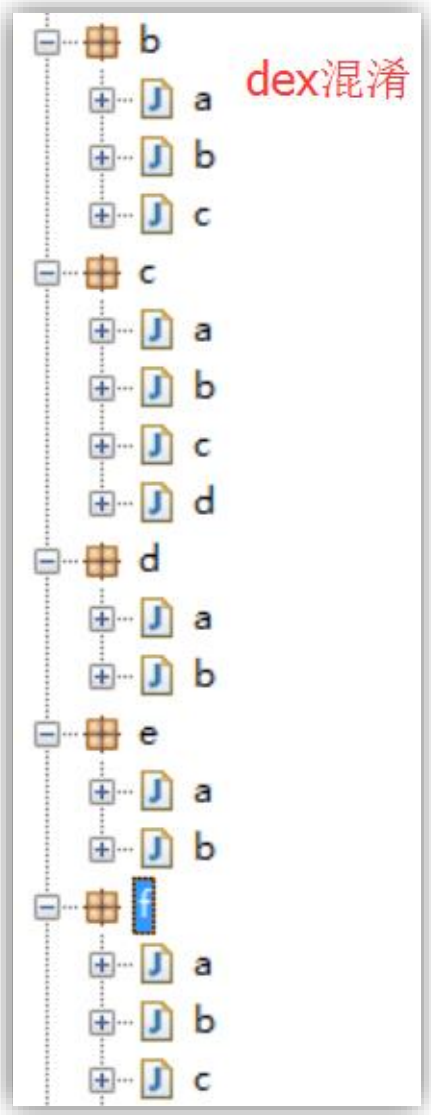
- DEX文件加花

```
    return v0
.end method

# write writeVirtualMethods error : 注册弹出菜单(Landroid/view/View;)V

    # write writeVirtualMethods error : 窗口置后台()V

    # write writeVirtualMethods error : 绑定活动栏(Lcom/e4a/runtime/component

        # write writeVirtualMethods error : 结束程序()V

            # write writeVirtualMethods error : 获取上下文()Landroid/content/

                # write writeVirtualMethods error : 获取全局应用()Landroid/ap

                    # write writeVirtualMethods error : 获取全局应用上下文()L
```

dex加花之后

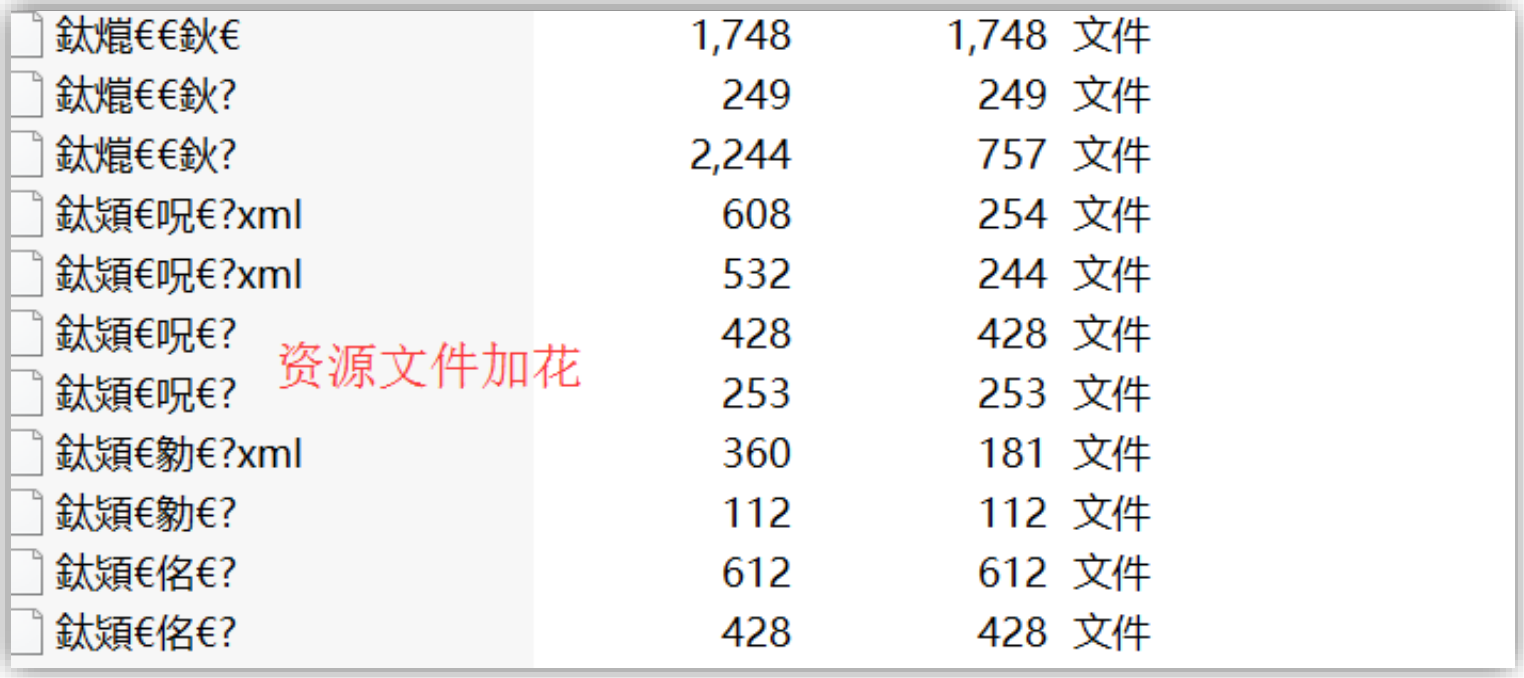
4.资源防止反编译

- DEX文件混淆



资源防止反编译

➢ 资源文件加花



2.第1代、2代、3代的加固特点

第一代壳：
Dex加密

1.Dex字符串加密

- 2.资源加密
- 3.对抗反编译
- 4.反调试
- 5.自定义 DexClassLoader

第二代壳：
Dex抽取与So加固

1.对抗第一代壳常见的脱壳法

- 1.Dex Method代码抽取到外部（通常企业版）
- 2.Dex动态加载
- 3.So加密

第三代壳 ：
Dex动态解密与So混淆

1.Dex Method代码动态解密

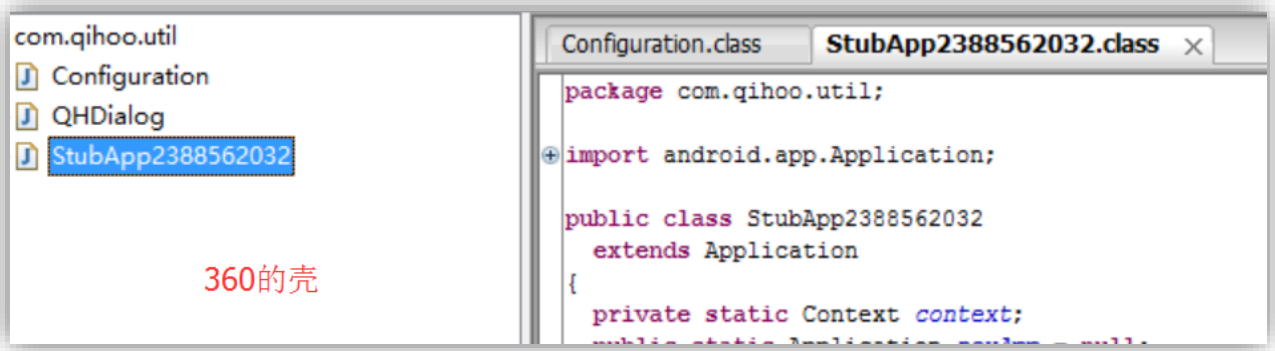
- 2.So代码膨胀混淆
- 3.对抗之前出现的所有脱壳法

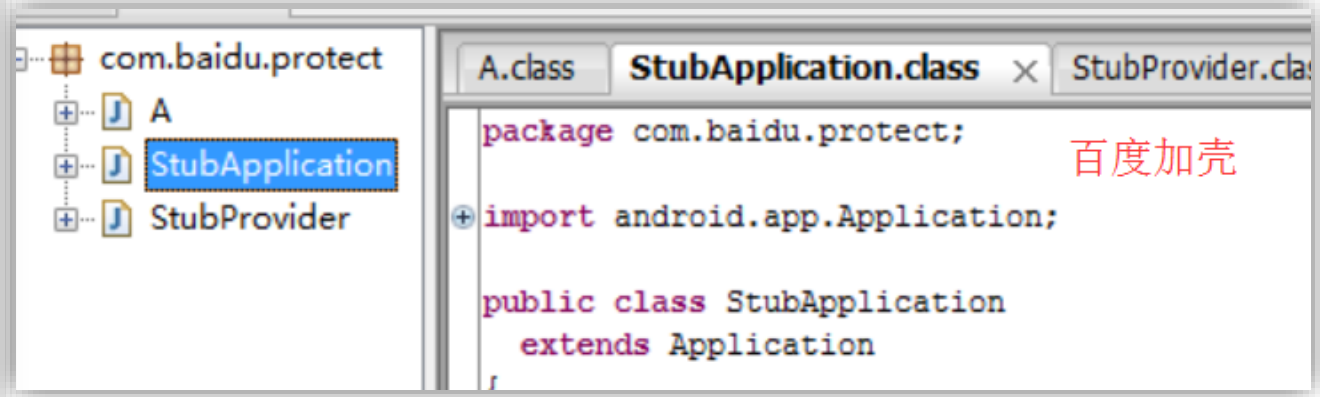
第四代壳：
arm vmp

1.vmp

3.加固方式

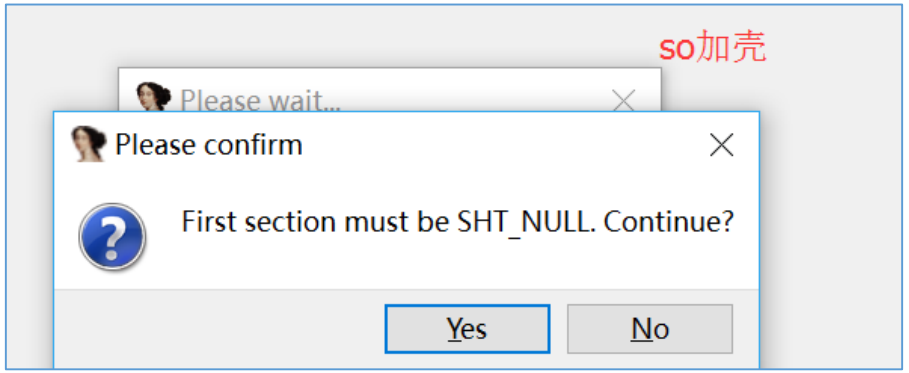
➢ dex源码加壳





3.加固方式

So加固



Name	Address	Ordinal
□	00038A04	
□	000390F4	
`p□p倏□C□s倡4s□@咀8p□p偊□C□s倡4s□@咀8p□p便□C...	00069210	
5p倏□C□s倡4s□@咀8p□p偊□C□s倡4s□@咀8p□p便□C□...	00072500	
C□s倡4s□@咀8p□p偊□C□s倡4s□@咀8p□p便□C□s倡4s□...	000563B0	
C□s倡4s□@咀8p□p便□C□s倡4s□@咀8p□p偲□C□s倡4s□...	00072504	
9s倡4s□@咀8p□p便□C□s倡4s□@咀8p□p偲□C□s倡4s□...	000691D4	
□@咀8p□p便□C□s倡4s□@咀8p□p偲□C□s倡4s□@咀8p...	00072508	
p□p便□C□s倡4s□@咀8p□p偲□C□s倡4s□@咀8p□p偩□C...	0007250C	
□p便□C□s倡4s□@咀8p□p偲□C□s倡4s□@咀8p□p偩□C□...	00055698	
s倡4s□@咀8p□p偩□C□s倡4s□@咀8p□p傛□C□s倡4s□@...	0006BA80	
p傛□C□s倡4s□@咀8p□p傚□C□s倡4s□@咀8p□p□C□s偺...	00065A28	
@咀8p□p傚□C□s倡4s□@咀8p□p□C□s偺4s□.	000654A4	
C□s倡4s□@咀8p□p□C□s偺4s□.	0006B880	
s□@咀8p□p□C□s偺4s□	0006CBE4	

so加壳

Thank you