ijiami.cn

移动应用安全态势感知及数据分析

深圳爱加密科技有限公司
2017年6月

目录页

Contents Page

目录页

Contents Page

第 1 章

常见的安全风险技术解析

二次打包风险

渗透分析：通过AndroidKiller反编译工具，可以在源码中植入风险提示，二次打包成功

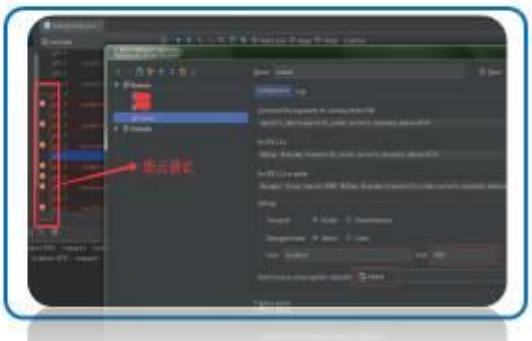
自身保护：防二次打包，做正盗版签名校验

校验分析：获取程序签名》获取签名信息》查找AES算法》绕过伪代码

Java层调试

漏洞解释：通过IDEA可以断点动态的调试APP，分析APP的加密算法、密钥、功能模块

破解工具：IDEA,Elipse,jdb



组件安全

漏洞解释：组件安全包括后台服务、Content Provider、第三方调用和广播等组件的安全，Intent权限的设置是否安全。应用不同组成部分之间的机密数据传递是否安全 敏感组件是否可以被其他程序导出等。

风险分析：本地拒绝服务器攻击、权限绕过、伪造升级、广播拦截等。

逆向工具：审计工具Drozer

```
dz> run app.package.attacksurface cn.com.bsb.mbank
Attack Surface:
  1 activities exported
  0 broadcast receivers exported
  0 content providers exported
  0 services exported
dz>
```

SO库安全

漏洞解释：so库是APP的核心代码块，很多重要的源码或者核心算法会放到so库中。

分析方案：Java源码中搜索调用so库中的关键方法，使用ida对so文件进行逆向分析。

搜索loadLibrary-》搜索native方法-》IDA打开libpnc-crypto.so-》查看Export回调方法-》

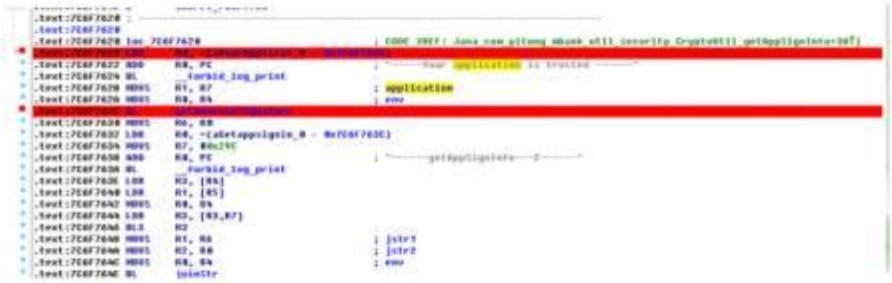
getAndroidSignature( ) 获取apk中签名信息-》查看AES加密算法

进行包验证的 getAppSignInfo() 方法逻辑如下图所示：

IDA动态调试

漏洞解释：so库是APP的核心代码块，很多重要的源码或者核心算法会放到so库中。

分析方案：通过IDA动态的Attach内存中的so库



完毕后，启动 start process，当 APK 操作到对应位置时，将会在对应位置停止！

敏感数据

漏洞解释：敏感数据包括：身份证、手机号码、银行卡账号、密码、用户名、登录密码等。

分析方案：界面Activity->关键代码处->Log注入取出敏感明文信息


```
StringBuffer v2 = new StringBuffer();
String v1 = arg9.aZ.getText().toString();  获取手机号码
FineT.printStr00(v1);
if(v1.startsWith("1") || v1.length() != v6) {
    v2.append("请输入正确的手机号码\n");
}
else if(v1.contains("*")) {
    v0 = com.yitong.mbank.util.security.a.c(arg9.C.getString("MobileNo", ""));
    if(v1.contains("******")) {
        if((v1.subSequence(0, v7).equals(v0.substring(0, v7))) && (v1.subSequence(v8, v6).eq
                v0.substring(v8, v6)))) {
            v1 = v0;
            goto label_17;
        }

        v2.append("请输入正确的手机号码\n");
    }
```

文件性验证

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(b1.m(this.a).applicationInfo.sourceDir);
Signature[] v2 = b1.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];
    b1.a(this.a, v4.toByteArray());
    v4.toCharsString();
    b1.b = ba.a("3082024b308201b4a00302010202044d56bbf300d06092a86
}
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

自验证案例

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调式下盗版提示
  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);
```

网络接口保护

- **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

资源混淆保护

- 资源防止反编译

- 修改XML结构
- 资源加花
- 源代码混淆

目录页

Contents Page

第 4 章

风险感知系统数据分析

安全数据管理分析平台架构

感知数据服务全景

基于感知数据的运营业务功能拓展

威胁感知系统

Thank you

ijiami.cn