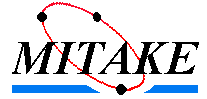
**三竹資訊股份有限公司**



**Android 安全性機制**

|  |  |
| --- | --- |
| 部門 | 金融研發二部 |
| 作者 | 黃士瑋 |
| 文件版本 | 1.0 |

目錄

[一. 目的 3](#_Toc439255627)

[二. 客户端程序安全 3](#_Toc439255628)

[2.1 客户端程序保护 3](#_Toc439255629)

[2.1.1 .解決方法: 3](#_Toc439255630)

[2.1.2 問題反饋 3](#_Toc439255631)

[2.1.3 經驗分享 3](#_Toc439255632)

[2.2 应用完整性检测 4](#_Toc439255633)

[2.2.1 .解決方法: 4](#_Toc439255634)

[2.2.2 程式碼 4](#_Toc439255635)

[2.3 组件安全 5](#_Toc439255636)

[2.3.1 .解決方法: 5](#_Toc439255637)

[2.4 webview组件安全 6](#_Toc439255638)

[2.4.1 解決方法 6](#_Toc439255639)

[2.4.2 經驗分享 7](#_Toc439255640)

[三. 敏感信息安全 7](#_Toc439255641)

[3.1 数据文件 7](#_Toc439255642)

[3.1.1 解決方法 7](#_Toc439255643)

[3.1.2 程式碼 7](#_Toc439255644)

[四. 密码软键盘安全 7](#_Toc439255645)

[4.1 键盘劫持與隨機鍵盤 7](#_Toc439255646)

[4.1.1 解決方法 8](#_Toc439255647)

[4.1.2 程式碼 8](#_Toc439255648)

[4.2 屏幕录像 8](#_Toc439255649)

[4.2.1 .解決方法: 9](#_Toc439255650)

[4.2.2 經驗分享 9](#_Toc439255651)

[五. 应用界面防止被劫持 10](#_Toc439255652)

[5.1 反劫持 10](#_Toc439255653)

[5.1.1 解決方法 10](#_Toc439255654)

[5.1.2 程式碼 10](#_Toc439255655)

[5.1.3 經驗分享 12](#_Toc439255656)

[六. 客户端未进行证书校验 12](#_Toc439255657)

[6.1 证书校验 12](#_Toc439255658)

[6.1.1 解決方法 12](#_Toc439255659)

[6.1.2 程式碼 12](#_Toc439255660)

[6.1.3 經驗分享 14](#_Toc439255661)

[七. 結論 14](#_Toc439255662)

1. 目的

此文件為Android客戶端安全性機制的描述與解決方法，其內容將描述各種程序上安全的問題、詳細解法與程式碼。

1. 客户端程序安全
   1. 客户端程序保护

防止程序在被外部人員反編譯後，程式的可讀性以及程序代碼保護。

* + 1. .解決方法:

使用代碼混淆的方法，詳細混淆的方法請參考小鄧發出來的信，也可參考以下網址中混淆代碼的說明: http://my.oschina.net/aibenben/blog/371889。

* + 1. 問題反饋

使用混淆的方法來保護程序，而在混淆後包出來的apk可能會發生一些造成程序不正常的現象，其中此次在混淆中遇到比較大的問題如下:

* 使用到反射調用的code會出現問題:此問題可以將有使用到反射的部分的code用別的方法實現，或者在混淆的時候不要混淆使用到反射的類。
  + 1. 經驗分享

1. 由於要測試混淆有沒有成功，必須在包apk檔後才能測試的出來；因此有時候再開發過程所使用的代碼程序沒有問題，但再包成apk檔測試的時候才發生問題，所以如果有發現是在發出去的版本才有的bug，在自我開發過程的程式是不會發生的，可以多一個方向去思考是否是混淆所生成的bug，而不是code本身的bug(嘗試將混淆先關掉，然後包apk測試看看)。
2. 混淆後的app有可能會發生的問題不一定是造成程式crash，有些是很小的問題點較不容易測試到(如畫面作動與原本預期不太一樣)，因此如果程序是第一次做混淆的話，則必須做app全面的測試，將每個功能都測試過，以防混淆所產生的bug。
3. 若能不用反射就不用反射，如果真的必須用到，則要記得在proguard-project.txt混淆時避開有用到反射的代碼，而詳細如何避開的方法請參考網址內容 http://my.oschina.net/aibenben/blog/371889。
   1. 应用完整性检测

為了避免apk被竄改，或者資源檔被替換掉，所做的完整性驗證功能。

* + 1. .解決方法:
* 判斷簽名:由於apk如果被修改過，理想狀況下修改者必須使用自己的key包版本，因此可判斷簽名來測得應用完整性。
* 程序進入時判斷資源檔的完整性:後將所有資源檔以md5加密取得一段字串碼，然後將此字串碼寫到程序的判斷函式中，每次程序啟動時判斷目前資源檔所取到字串碼與原本的相不相同。
  + 1. 程式碼

/\*

\*判斷簽名

\*/

if(getApkInfo.getPublicKey(this).equals("modulus=bcdd07a3acef6f8e2e12cd adcdca235e2e4571bfaa5b08b78ac2920ad0964a153acef6f8e2e12cdc5f5472c0f8adcdca238e0afbc52e9bdcb1733b770632400bbeacda7dcd229f9de75fb0617e0c8d64e4396fa6719e8dd4553b0df0340c0923c62fe92d2ec266974315e52f0ed 41be87c337b802d22425f"))  
{

/\*

\*在最开始做app完整性校验,对资源文件做比對

\*/

JarFile jarFile = null;  
 try {

//取得資源擋路徑，並找出所有資源檔  
   jarFile = new JarFile(this.getPackageResourcePath());  
   Enumeration entries = jarFile.entries();  
   StringBuffer buffer = new StringBuffer();  
   while (entries.hasMoreElements())   
   {       
   JarEntry ze = (JarEntry) entries.nextElement();    
   if(ze.getName().contains("res")){   
   InputStream in = jarFile.getInputStream(ze);  
   buffer.append(MD5Util.md5sum(in));  
   }  
   }   
   if(!MD5Util.get(buffer.toString()).equals("edc255a6f8c598bcd842768f53601f62")){  
   isDeal = true;  
   DialogUtility.showSimpleAlertDialog(this, "安装包已被破坏，请重新下载安装包！   
}else {  
isDeal = true;  
   DialogUtility.showSimpleAlertDialog(this, "安装包已被破坏，请重新下载安装包！", new DialogInterface.OnClickListener()  
   {  
   @Override  
   public void onClick(DialogInterface arg0, int arg1)  
   {  
   finish();  
   System.exit(0);  
   }  
   }, false).show();  
   return ;  
}

* 1. 组件安全

BroadcastReceiver组件暴露风险，广播可被外部调用导致敏感信息泄露。

* + 1. .解決方法:
* 在ManiFest內设置android:exported="false"，不讓外部調用进程内动态广播注册建议 使用LocalBroadcastManage；或者使用registerReceiver(BroadcastReceiver,IntentFilter, broadcastPermission, Handler)替代registerReceiver(BroadcastReceiver, IntentFilter)
* 若需要外部调用，需定义signature或者signatureOrSystem级别的权限signature或signatureOrSystem级别的私有权限保护；需要暴露的组件请严格检查输入参数，避免应用出现拒绝服务。
  1. webview组件安全
* webview中的javascript导出java本地接口，可能导致任意命令执行。
* 系统webkit内置的危险接口searchBoxJavaBridge。

以上問題在4.2以上的版本android官方都已經有解決了，但我們的app有支援到4.2以下，因此要針對4.2以下的版本防堵這些危險性的問題

* + 1. 解決方法

4.2以下不能使用addJavaScriptInterface來實現javascript接口，而要改用其他JS与Java进行交互的方法，有以下几种，比prompt, alert等，而這些动作都会对应到android中WebChromeClient类中相应的方法，对于prompt，它对应的方法是onJsPrompt方法，其android上實現代碼如下:

**private** WebChromeClient client = **new** WebChromeClient()

{

@Override

**public** **boolean** onJsPrompt(WebView view, String url, String message,

String defaultValue, JsPromptResult result) {

//判斷message內容得知js需要我們做什麼事情，以及取得js傳過來的參數

if(message.equals("done"))

{

result.confirm(“”);

return true;

}else if(message.equals("finish"))

{

result.confirm(“”);

return true;

}else if(message.contains("play"))

{

result.confirm(“”);

return true;

}

return super.onJsPrompt(view, url, message, defaultValue, result);

}

};

而如果要把字串參數帶回給js中，可在result.confirm(“要帶的信息”)，將字串帶過去，即可做到ｊｓ與ｊａｖａ互通的功能需求。

* 移除有危險的接口:

//移除有風險的接口

判斷ＳＤＫ版本，使用removeJavascriptInterface移除系統有危險的接口

**if**(Build.VERSION.*SDK\_INT*>10&&Build.VERSION.*SDK\_INT*<17){

webView.removeJavascriptInterface("searchBoxJavaBridge\_");

}

* + 1. 經驗分享
* 由於web方面也要做修改才有辦法實現以上第一點修正的方法，因此新版本的web方面如果有修改，也必須要兼容舊版本客戶端的app。
* 請注意onJsPrompt的回傳值控制，不然會影響本來使用到Prompt彈掉視窗的功能

1. 敏感信息安全
   1. 数据文件
      1. 解決方法

数据文件安全性-文件限制于程序内才可存取。

* + 1. 程式碼

FileOutputStream fos = context.openFileOutput(name, Context.MODE\_PRIVATE);

//禁用MODE\_WORLD\_READABLE和MODE\_WORLD\_WRITEABLE选项打开文件

1. 密码软键盘安全
   1. 键盘劫持與隨機鍵盤

在应用内部能直接调用第三方键盘及输入法，键盘可以被劫持，並且未使用随机软键盘布局。

* + 1. 解決方法

使用自訂義的鍵盤，並支援隨機鍵盤功能。

* + 1. 程式碼

請使用開發並封裝好的小鍵盤元件(附件: MyKeyBoard)，其使用元件的方法如下:

其中MyPop初始化所帶的第四個參數為需不需要鍵盤做隨機變換的參數。

pop = new MyPop(activity, ViewGroup.LayoutParams.MATCH\_PARENT, ViewGroup.LayoutParams.WRAP\_CONTENT, AppInfo.isRan);

//監聽要彈出鍵盤的EditText Listener

inputPassword.setOnTouchListener(new OnTouchListener() {

@Override

public boolean onTouch(View arg0, MotionEvent arg1) {

if(pop != null){

pop.setEditText(inputPassword);

pop.hideSoftInputMethod();

//pop.showAtLocation(layout, Gravity.BOTTOM, 0, 0);

pop.setShow(layout, Gravity.BOTTOM, 0, 0);

}

return false;

}

});

* 1. 屏幕录像

可以使用连续截屏工具获取用户正在输入的密码，如下：



* + 1. .解決方法:

按照android SDK的要求，开启FLAG\_SECURE选项的窗口不能被截屏。

Window win = getWindow();

win.addFlags(WindowManager.LayoutParams.*FLAG\_SECURE*);

* + 1. 經驗分享

由於設置此flag要在activity onCreate開始時設置才會有效，因此如果會員頁面(fragment)跟其他頁面(fragment)都是在同個activity下，那這些activity下的全部頁面就都不能截圖，可能會造成某些的不方便；因此如果只要會員頁面不能截圖，可能就必須要開啟另一個activity來打開這些會員頁面。

1. 应用界面防止被劫持
   1. 反劫持

正常应用的界面被恶意攻击者劫持，替换上仿冒的恶意界面作恶。界面劫持类攻击极具迷惑性很难被识别出来。

* + 1. 解決方法

程式中如果界面被其他软件界面覆盖开启一个服务，并提示用户风险。其中用户主动操作的不启动服务（如按home键，后退键等），代码如下

* + 1. 程式碼

新增一个service 如下:

**public class** AlarmService **extends** Service {  
 **boolean isStart** = **false**;  
 **private** PackageManager **pm**;  
 Handler **handler** = **new** Handler() {  
 **public void** handleMessage(android.os.Message msg) {  
 String name = msg.getData().getString(**"name"**);  
 Toast.*makeText*(getApplicationContext(), name, 0).show();  
 };  
 };  
  
 Runnable **alarmRunnable** = **new** Runnable() {  
  
 @Override  
 **public void** run() {  
  
 *// 得到ActivityManager* ActivityManager activityManager = (ActivityManager) getSystemService(Context.***ACTIVITY\_SERVICE***);  
 *// 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  
 // String packageName = info.topActivity.getPackageName();* **pm** = getPackageManager();  
 String packageName = activityManager.getRunningTasks(1).get(0).**topActivity** .getPackageName();  
  
 String appName = **""**;  
 **try** {  
 appName = **pm**.getApplicationLabel(  
 **pm**.getApplicationInfo(packageName,  
 PackageManager.***GET\_META\_DATA***)).toString();  
 } **catch** (NameNotFoundException e) {  
 *//* ***TODO Auto-generated catch block*** e.printStackTrace();  
 }  
**if** (!shortClassName.equals(**".Launcher"**)) {  
  
 Toast.*makeText*(getApplicationContext(),  
 **"疑似非"** + appName + **"app界面，请注意！"**, Toast.***LENGTH\_LONG***).show();  
   
 }  
 }  
  
 };  
  
 @Override  
 **public int** onStartCommand(Intent intent, **int** flag, **int** startId) {  
  
 **super**.onStartCommand(intent, flag, startId);  
  
 **while** (!**isStart**) {  
  
 **isStart** = **true**;  
  
 *// 启动alarmRunnable* **handler**.postDelayed(**alarmRunnable**, 1000);  
  
 stopSelf();  
  
 }  
  
 **return *START\_STICKY***;  
  
 }  
  
 @Override  
 **public** IBinder onBind(Intent intent) {  
  
 **return null**;  
  
 }

2. 在onpause内加入下方

Intent intent = new Intent(this, AlarmService.class);

startService(intent);

* + 1. 經驗分享

如果为使用者自己跳出登入画面或程序的话（home键、返回键与画面跳转等）,不要呼叫警告的service,因此必须加入flag去判断。

目前按menu键也会弹出提示风险的提示。

1. 客户端未进行证书校验
   1. 证书校验

需针对服务器的证书进行严格校验，防止中间人攻击。

* + 1. 解決方法

服务器需先将证书.bks档案产出并且把产出过程的密码和此.bks给客户端 ,客户端将此.bks档放在raw下面,以便于做校验，防止中间人攻击。

* + 1. 程式碼

1.由服务器端产生的密码(KEY\_PASS)和.bks档案,将原本 trustStore 改为读取本地

InputStream ins = mContext.getResources().openRawResource(R.raw.certsseinfo);

trustStore = KeyStore.getInstance(KeyStore.getDefaultType());

try {

trustStore.load(ins, KEY\_PASS.toCharArray());

}

finally {

ins.close();

}

2.接著设定mitakeSSLSocketFactory.setHostnameVerifier

由原本的

SSLSocketFactory.ALLOW\_ALL\_HOSTNAME\_VERIFIER

改为

SSLSocketFactory.STRICT\_HOSTNAME\_VERIFIER

下方为详细代码

private static final String KEY\_PASS = "pasw123456";

  SchemeRegistry schemeRegistry = new SchemeRegistry();

schemeRegistry.register(new Scheme("http", PlainSocketFactory.getSocketFactory(), 80));

KeyStore trustStore=null;

if(mContext!=null){

InputStream ins = mContext.getResources().openRawResource(R.raw.certsseinfo);

trustStore = KeyStore.getInstance(KeyStore.getDefaultType());

try {

trustStore.load(ins, KEY\_PASS.toCharArray());

}

finally {

ins.close();

}

}else{

trustStore = KeyStore.getInstance(KeyStore

.getDefaultType());

trustStore.load(null, null);

}

MitakeSSLSocketFactory mitakeSSLSocketFactory = new MitakeSSLSocketFactory(trustStore);

//mitakeSSLSocketFactory.setHostnameVerifier(SSLSocketFactory.ALLOW\_ALL\_HOSTNAME\_VERIFIER);

mitakeSSLSocketFactory.setHostnameVerifier(SSLSocketFactory.STRICT\_HOSTNAME\_VERIFIER);

* + 1. 經驗分享

1. 結論

以上安全性為目前上海專案中發現的安全性問題，而解決方法並不會只有一種

，且安全性的強度也依程序代碼的機密性有不同的策略做法，因此還需依專案需求做適時的修正；而目前這邊往後的專案都會導入上方所有安全性的機制，以此為標準，若各位同仁有其他關於Android安全性的機制或者對目前安全機制有更優化的或更好的解決方法也可以分享出來。