目录

[**1.** **防止工具反编译** 2](#_Toc442119151)

[1.1 伪加密 2](#_Toc442119152)

[1.2 APK压缩包故意破坏 2](#_Toc442119153)

[1.3 素材中的图片故意破坏 2](#_Toc442119154)

[**2.** **代码高级混淆** 2](#_Toc442119155)

[2.1 花指令在源码被窃后增加黑客阅读难度 2](#_Toc442119156)

[2.2 文件夹混淆(文件.2   正式版本文件夹取名尽量不要太正规) 2](#_Toc442119157)

[**3.** **运行时验证** 2](#_Toc442119158)

[3.1 签名验证的各种方法 2](#_Toc442119159)

[**4.** **防止内存修改** 2](#_Toc442119160)

[4.1 如何防止“八门神奇”通过内存数据 2](#_Toc442119161)

[**5.** **进程注入** 2](#_Toc442119162)

[5.1 静态注入 2](#_Toc442119163)

[5.2 动态注入 2](#_Toc442119164)

1. **防止工具反编译**
   1. 伪加密
   2. APK压缩包故意破坏
   3. 素材中的图片故意破坏
2. **代码高级混淆**
   1. 花指令在源码被窃后增加黑客阅读难度
   2. 文件夹混淆(文件.2   正式版本文件夹取名尽量不要太正规)
3. **运行时验证**
   1. 签名验证的各种方法
4. **防止内存修改**
   1. 如何防止“八门神奇”通过内存数据
5. **进程注入**
   1. 静态注入

静态注入比较经典的案例就是二次打包了。比如说 登陆注册的输入框里面注入钩子，盗取账号密码。动态注入的经典例子就是 内存修改器比如说八门神奇。  防止静态注入 就得防止二次打包了。防止二次打包的方式很多的。

* 1. 动态注入

防止内存修改就得编码的时候注意了，比如说：关键变量(当前分数，当前时间，当前血量，当前等级 等等) 写两份，一份加密的 一份未加密的，  定时用加密那份来检验。

1. 附录

一致性参考代码：

@Override  
**public void** onCreate(Bundle savedInstanceState)  
{  
 **super**.onCreate(savedInstanceState);  
 getSingInfo();  
 **boolean** bCorrectSign = **false**;  
 **if** (bCorrectSign){  
 *// Set by <content src="index.html" /> in config.xml* loadUrl(**launchUrl**);  
 }  
 **else** {  
 System.*exit*(0);  
 }  
  
}  
**private** String toHex(**byte**[] paramArrayOfByte) {  
 StringBuffer localStringBuffer = **new** StringBuffer();  
 **for** (**int** i = 0; i < paramArrayOfByte.**length**; i++) {  
 Object[] arrayOfObject = **new** Object[1];  
 arrayOfObject[0] = Byte.*valueOf*(paramArrayOfByte[i]);  
 localStringBuffer.append(String.*format*(**"%02x"**, arrayOfObject));  
 }  
 **return** localStringBuffer.toString();  
}  
**private void** parseSignature(**byte**[] signature) {  
 **try**{  
 CertificateFactory certFactory = CertificateFactory.*getInstance*(**"X.509"**);  
 X509Certificate cert = (X509Certificate) certFactory.generateCertificate(**new** ByteArrayInputStream(signature));  
 **byte**[] buffer = cert.getEncoded();  
 String pubKey = cert.getPublicKey().toString();  
 String signNumber = cert.getSerialNumber().toString();  
 String na = cert.getSigAlgName();  
 String dn = cert.getSubjectDN().toString();  
 buffer = cert.getEncoded();  
 } **catch** (CertificateException e) {  
 e.printStackTrace();  
 }  
}  
**public void** getSingInfo(){  
 **try** {  
 PackageInfo packageInfo = getPackageManager().getPackageInfo(getPackageName(), PackageManager.***GET\_SIGNATURES***);  
 android.content.pm.Signature[] signs = packageInfo.**signatures**;  
 Signature sign = signs[0];  
 MessageDigest localMessageDigest = MessageDigest.*getInstance*(**"MD5"**);  
 localMessageDigest.update(sign.toByteArray());  
 *// md5 stringֵ* String str2 = toHex(localMessageDigest.digest());  
 *//int h = sign.hashCode();  
 //parseSignature(sign.toByteArray());* } **catch** (Exception e){  
 e.printStackTrace();  
 }  
}