APP安全加固技术的讨论——安全小课堂第二十八期

京东安全应急响应中心 2016-09-30

点击上方"蓝字"关注本宝宝公众号

安全小课堂第二十八期

概括来说，app加固是对原本容易暴露的程序运行逻辑进行保护，而不影响程序本身的运行结果。app安全加固有些弊端，但总体优点大于缺点。本期我们来聊一聊APP安全加固。

本期邀请到
盘古安全专家小傅
四叶草安全专家小峰
大家欢迎~

/ 01 /

豌豆妹

什么是app安全加固?

小新 

概括来说，app加固是对原本容易暴露的程序运行逻辑进行保护，而不影响程序本身的运行结果。包括dex和so。目前来看，保护dex文件更为重要，因为dex文件反编译之后的代码可读性更强。

柴可夫斯基 

详细点说，就是在二进制的程序中植入一段代码，在运行时优先取得程序的控制权，做一些额外的工作。大多数病毒就是基于此原理。是应用加固的一种手法对原始二进制原文进行加密/隐藏/混淆。

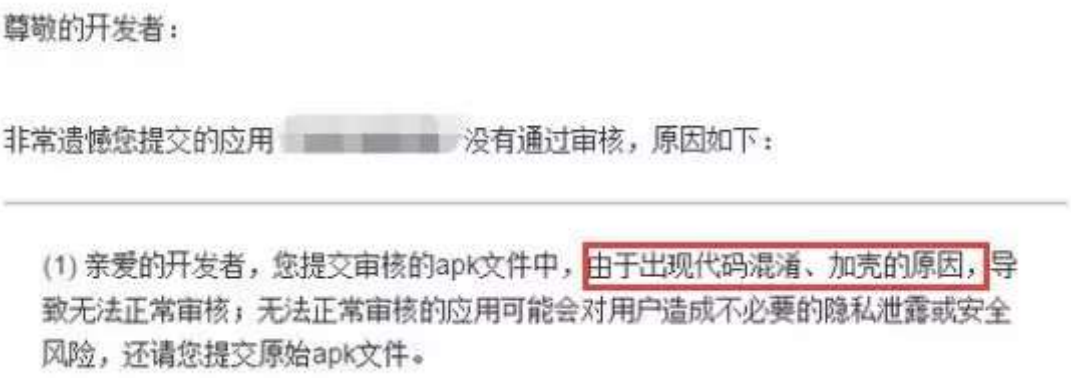
/ 02 /

豌豆妹

能说说app安全加固的影响吗?

小丸子 

我的理解，影响有两方面：正面和负面。 正面：保护自己核心代码算法，提高破解/盗版/二次打包的难度，缓解代码注入/动态调试/内存注入攻击。负面：1.影响兼容性；2.影响程序运行效率，目前大的加固厂家兼容性、运行效果都做的还是不错，需要多试一试；3.部分流氓、病毒也会使用加壳技术来保护自己；4.部分应用市场会拒绝加壳后的应用上架
http://www.zhihu.com/question/32084878。


尊敬的开发者：

非常遗憾您提交的应用 [redacted] 没有通过审核，原因如下：

(1) 亲爱的开发者，您提交审核的apk文件中，由于出现代码混淆、加壳的原因，导致无法正常审核；无法正常审核的应用可能会对用户造成不必要的隐私泄露或安全风险，还请您提交原始apk文件。

上图这个问题就比较悲催了。因为加固之后，应用商店审核起来很麻烦，我猜测是为了防止别人把加固后的恶意软件或病毒放入应用商店，所以做的限制。没加固的容易被破解、被重打包，造成仿冒应用泛滥，甚至犯罪分子会增加恶意代码然后重打包发布。这些行为会严重损害开发者和开发厂商的利益。加固服务出现的背景也基本是针对打包党的。总而言之，正面效果强于负面效果。

哆啦A梦

兼容性的问题，是app加固技术上的一个难题。很多时候为了追求兼容性，不得不放弃一些技术。并且，目前很多应用市场要求程序带有自己的加固方案。导致一些应用不能占用许多市场、或者发布不同的apk，需要维护多版本。比如360加固的应用，baidu认为不能确保程序的安全性，就不会给发布。不过，正面的效果也是很明显的。前几年猖獗的二次打包事实上受到了不少打击。而且，随着加固技术的发展，也带来了脱壳技术的提升。有加壳的病毒在之前看来比较难分析，现在基本用一些通用技术就可以搞定。

豌豆妹

前面提到加固（dex加固、so加固）可能会影响兼容性、维护的难度等问题。那在重打包问题上，本地验证不靠谱，若采用服务器校验app签名，需要注意些什么问题？或者其他有效的实现方案？

柴可夫斯基

不加固的话，反编译之后代码完全暴露，校验算法还是能看到的。比如把你服务器校验地址改成127.0.0.1，修改成本地校验，或者去掉校验逻辑。总体利大于弊。

/ 03 /

豌豆妹

那app安全加固的必要性呢？

葫芦娃

必要性前面其实已经提到一些了，总结来说，一方面保护开发者，防止被重打包，保

必要性前面其实已经提到一些了。总结来说，一方面保护开发者，防止被重打包，保护重要代码逻辑、算法，防止被反编译，被破解。破解的话，一般是修改关键逻辑，破解应用内购买，加固技术对这类破解也有很大的保护作用。有些病毒就是重打包一些热门游戏，向里面增加病毒代码，来传播的；一方面，保护了开发厂商利益，有些代码里会有一些漏洞，apk加固也可以在一定程度上做到保护，给开发商更多时间；另一方面也提高了用户体验。重打包的app从外面几乎分辨不出来。大多数用户的甄别没有那么强。在有了加固技术后，这类应用会少很多。

/ 04 /

豌豆妹

请教下app安全加固的分类。

小新

核心逻辑上，一种是动态载入，另一种是vmp，相当于自己实现虚拟机，但通常只是指令的简单替换。由于兼容性问题，目前没遇见dex文件的加固用到vmp。从加固的范围上，可以分为dex加固，so和dex加固，核心逻辑加固。因为加固会影响运行效率，所以有些加固只对核心逻辑进行有效的保护，大多数维护运行的代码还是暴露的。资源也会被加密，还有一些加密技术会结合一些防止反编译的技术。比如在resource文件里做一些手脚，可以影响apktool。在smali里加一些代码，可以影响baksmali工具。不过这些都是辅助性的。

/ 05 /

豌豆妹

能分享下app安全加固的思路么？

小丸子

目前app加固几乎都是利用DexClassLoader来动态加载dex来实现的。由于这段逻辑本身会暴露dex文件。所以通常要配合反调试技术。一方面是混淆逻辑，另一方面可以对一些文件的特殊内容进行检查，还可以检查代码运行时间，来判断是否在被调试。在对抗一些脱壳工具方面，会在程序中做hook，检查是否有特殊的逻辑存在。

葫芦娃

一般使用常见的加固工具加固应用后，他会把你的dex、so加密存在apk中，然后apk运行过程会先运行壳的代码，壳的代码再把原来的dex、so解出来加载，不同厂商自己的方案略有差距，但目前多数都是这个思路，具体细节各厂家有各自的方法。

/ 06 /

豌豆妹

有不错app安全加固案例可以分享么?

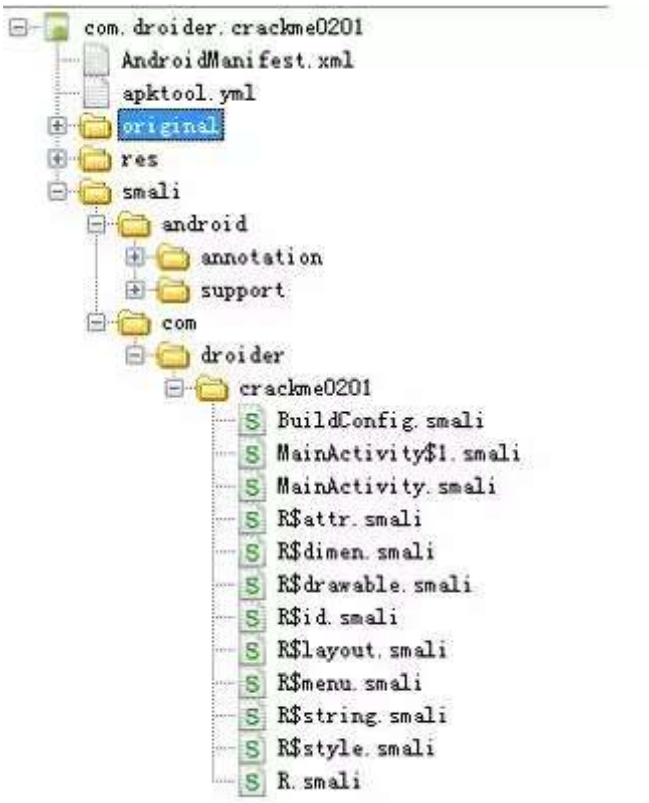
小新

分享一个早先版本的ali的dex加密技术吧。加密后的程序会有一些额外的文件。libmobisecx.so,libmobisecy1.so,libmobisecz1.so。其中有一个是加壳过的so文件。但是不是程序本身的，而是阿里加壳解密的核心逻辑。在so中，几乎所有的字符串都是被加密过的。可以通过两段数据解密得到。这样就可以很大程度上防止通过字符串来推测逻辑。因为很多字符串是一些函数名，或者参数。在so中许多逻辑都是反射调用java层实现的，这类代码逻辑不容易弄清，会加大跟踪难度。然后其中的反调试技术，有检查TracerPid和命令行中的gdb gdbserver android_server等。而且核心的逻辑基本都是动态解密的。ali的加壳，在调用DexClassLoader的时候，加载的不是完整的dex。在真正加载类的时候，会重新修改一些偏移，使文件变得完整。再具体的细节说起来就比较麻烦了。

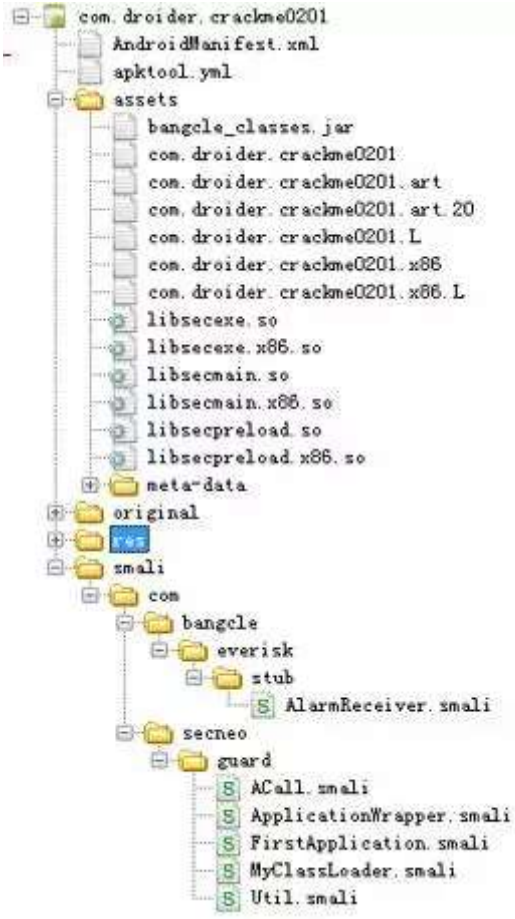
柴可夫斯基

恩，so层很多函数名都混淆了。有些加固后，全部函数名被改成没有意义的字符了。我放几个图，看看加固效果，会更好理解一点。

加固前：



加固后（assets目录下多了很多文件）：



so库函数名加密，应该算是so代码混淆：

没记错的话，libsecmain.so是可执行程序，这个应该是入口。fork之后，子进城execl替换了原程序。做起了"专职监护人"。

哆啦A梦

目前也有一些会对原本的执行逻辑进行混淆。APKProtector就有这样的功能。即使脱壳成功，代码的可读性也不是很好。

小丸子

在使用这些加固产品之前，有些工作可以自己做一下。比如java代码中的字符串加

密，可以自己实现加解密算法，全部用加密字符串，代码中解密。这些简单工作也可以增加逆向难度。这是一个上千万装机量的app中使用的方法，不能完全依赖加壳。在不影响效率的情况下，增加逻辑的复杂性是很不错的选择。可以适当减少程序的封装。对付逆向的方式就是消磨对方耐心，增加时间成本。

```
Intent localIntent = new Intent(paramActivity, AddAccountActivity.class);
localIntent.putExtra(a.c("JAoHLRgTFyobDQYmFhs3MQIRDRkCLBoa"), paramBoolean);
localIntent.putExtra(a.c("IQEOExAe"), paramString2);
localIntent.putExtra(a.c("IQEOExAeKzEXExc="), paramInt);
localIntent.putExtra(a.c("MQcXHhw="), paramString1);
```

豌豆妹

哈~谢谢小伙伴的答疑解惑呢~咱们本期到此结束哦，有想知道的话题，也可以回复公共号告诉豌豆妹，宝宝会尽力满足大家的哟！下期见！

长按识别二维码
关注“公众号”