谈谈移动应用加固

张勇 LBE安全大师

为什么需要加固

- Android应用主要以Java语言编写，混淆作用有限
- Android为开源系统，逆向和反编译工具非常成熟
- Android系统淡化进程概念，基于消息和事件触发运行的机制使得插入恶意代码甚至无需接触字节码
- Root后可利用ptrace API或修改system image实现动态注入

应用保护

保护你的App，增加开发

对App进行加固保护，彻底

不改变App的用户体验，不

渠道监测

站、论坛等），能够及

漏洞分析

1、DEX文件保护检测
2、代码混淆保护检测
3、资源文件保护检测
4、主配文件保护检测
5、防二次打包保护检测
6、SO库文件保护检测

加密服务

防逆向分析

应用加固

我的应用列表

立即加固

免费渠道检测，甄别盗版APP

精准识别APP盗版
多维度盗版信息详情展示
多渠道数据统一监控

操

影响

只需上传移动应用

能和运行环境零影响

APP免费保护，解除盗版威胁

防插广告、防篡改

加固零成本

线上加固，无开发成本
一键拥有顶级安全保护

应用零风险

防止应用被二次打包、恶意篡改、内存截取等风险

大小零增加

独创隐形压缩技术
加固后文件大小零增加

使用零影响

完美兼容各版本安卓系统
对应用功能、性能零影响

- 代码加壳
  - classes.dex整包加密方案
    - 将原始classes.dex加密后单独保存，使用壳加载器替换原始classes.dex
    - 运行时解密classes.dex
    - 梆梆，爱加密，百度，通付盾
  - classes.dex字节码变形方案
    - 对原始classes.dex进行预处理，隐藏关键方法的字节码
    - 腾讯，360
  - 虚拟机方案
    - 将关键部分代码编译为专用虚拟机代码，于专用虚拟机中执行
    - 部分厂商正在研发中

- 防篡改
  - 记录加固后APK内文件hash值，在运行时比对
  - （部分）记录加固前APK的签名信息，要求加固后使用相同签名重新签名
  - 记录加固前AndroidManifest.xml文件内容，在运行时比对，防止二次打包，添加权限等

- 运行时
  - –classes.dex整包加密方案
    - 壳Application会首先运行
      - –在壳Application的构造函数, attachBaseContext或onCreate中，执行脱壳操作
      - –解密Dex文件并将其加载至内存中
      - –反射LoadedApk.mApplication, LoadedApk.mClassLoader, ActivityThread.mInitialApplication, ActivityThread.mAllApplications等值，将其重新指向目标Application和ClassLoader。确保系统稍后构造组件时能正确的加载到目标类
      - –最后，将控制权交回目标Application

- 运行时
  –classes.dex字节码变形方案
    - 相对简单，无需反射修改
    - 在Application加载之前首先加载解密代码，重新连接加密后的class和method
    - 解密后，将控制权交还至目标Application

- 运行时
  - Ptrace保护
    - 多进程相互ptrace防止ptrace attach或者memory dump
      - GC, SignalCatcher线程怎么办?
      - /proc/[pid]/mem仍然可读
    - 会轮询/proc/[pid]/status以确保TracerPid为0
      - 影响性能，电池在哭泣
  - Android进程从zygote fork而来，直接ptrace zygote即可绕过所有ptrace保护

- 梆梆加固分析
  - 加密原始classes.dex并放置于assets/bangcle_classes.jar内
  - 修改AndroidManifest.xml，替换Application对象
  - 对AndroidManifest.xml中声明的所有Receiver, Service和ContentProvider构造对应的Stub implementation
  - 运行时首先在内存中解密bangcle_classes.jar，然后使用custom classloader将其加载至内存并跳转至其中代码运行
  - 未保护资源文件和so文件
  - 运行时会启动2个辅助进程实现anti-ptrace

- 梆梆加固脱壳思路
  - Classes.dex并未在加固时执行预处理，运行时memory dump即可获得完整原始文件
  - zjDroid或dextractor可直接脱壳
  - 也可直接分析/proc/[pid]/maps并dd /proc/[pid]/mem导出odex文件

DEMO

- 爱加密加固分析
  - 加密原始classes.dex并放置于assets/ijiami.dat内
  - 修改AndroidManifest.xml，替换Application对象
  - 在SuperApplication.attachBaseContext处开始解密（因此无需对Service, Receiver等对象生成Stub)
  - 运行时首先在内存中解密bangcle_classes.jar，然后使用custom classloader将其加载至内存并跳转至其中代码运行
  - 未保护资源文件和so文件
  - 运行时会使用3个进程相互保护ptrace

- 爱加密加固脱壳思路
  - Classes.dex并未在加固时执行预处理，运行时memory dump即可获得完整原始文件
  - zjDroid或dextractor可直接脱壳
  - 也可直接分析/proc/[pid]/maps并dd /proc/[pid]/mem导出odex文件

DEMO

- 百度加固分析
  - 加密原始classes.dex并放置于assets/baiduprotect.jar内
  - 修改AndroidManifest.xml，替换Application对象
  - 对ContentProvider生成Stub
  - 脱壳流程同梆梆和爱加密几乎相同
  - 未保护资源文件和so文件
  - 运行时会使用3个进程相互保护ptrace

- 百度加固脱壳思路
  - Classes.dex并未在加固时执行预处理，运行时memory dump即可获得完整原始文件
  - zjDroid或dextractor可直接脱壳
  - 也可直接分析/proc/[pid]/maps并dd /proc/[pid]/mem导出odex文件

DEMO

- 腾讯云加固分析
  - 不替换原始classes.dex，也没有做任何加密处理
  - 修改Dex的ClassData, 将指定方法标记为native, 从而绕过静态分析工具，原始字节码仍然未加密的保存在dex文件中
  - 应用启动时，在Application的入口点处加载libtup.so, libtup.so会在内存中修改DexFile结构体，重建Method同CodeItem的映射关系
  - 此方法和xposed的思路有异曲同工之妙
  - 原理决定了不支持ART

- 腾讯云加固脱壳思路
  - 虽然Classes.dex做了较多的预处理操作，但原始字节码明文保存在classes.dex中显得非常不安全。
  - 应用运行后，libtup.so会修改Method对象的accessFlags, registersSize, outsSize, insSize, insns等属性，将函数复原
  - 使用xposed或者注入进程后，找到目标Dex文件的DvmDex结构体，遍历pResMethods，记录每个函数的insns偏移量。利用这个偏移量，修复加固后的classes.dex文件即可脱壳

DEMO

- 360应用加固分析
  - TODO

- 360加固脱壳思路
  - TODO

DEMO

应用加固，是否名副其实？

- classes.dex加密技术可以通过memory dump或zjDroid破解
- ptrace保护可以轻松绕过
  –现有单一应用加固不足以提供足够安全性保障

- 腾讯和360使用的字节码变形能够提高破解门槛
- 类VMProtect虚拟机技术可能是未来发展方向

- 开发者需要清醒面对宣传，做好充分的准备，多管齐下解决安全问题。