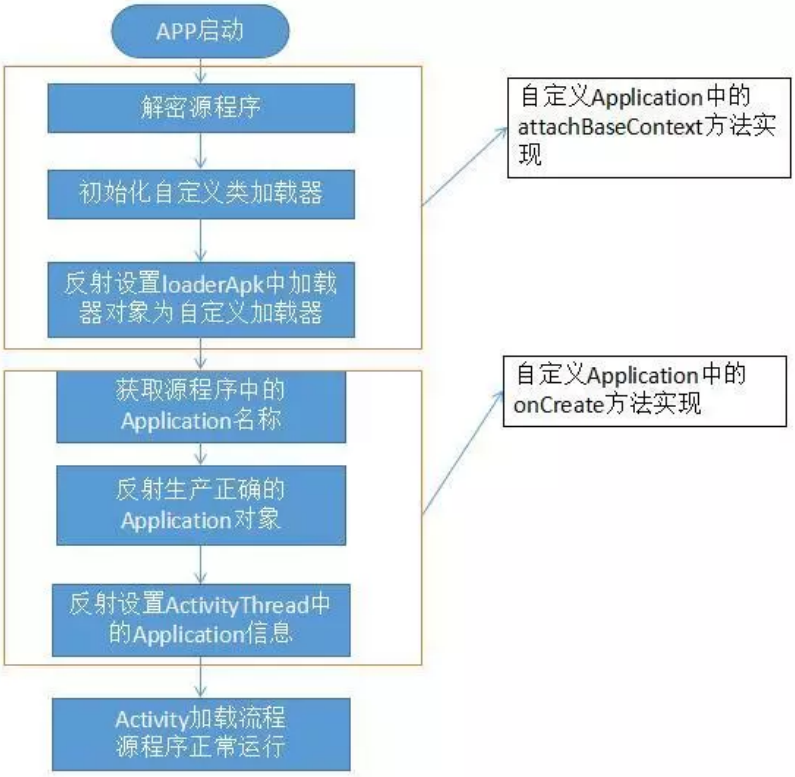
Andrid 脱壳研究

目录

1．Android App 加固及脱壳原理分析

加壳是在二进制的程序中植入一段代码，在运行的时候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。



加固方式

1.1 代码层级加密

（1）代码混淆

代码混淆是一种常用的加密方式。本质是把工程中原来的有具体含义的类名、变量名、方法名，修改成让人看不懂的名字。常见的代码混淆工具 proguard（有兴趣的可以自己看一下该工具：http://t.cn/ELjgHdi）。该加密方式只是对工程提供了最小的保护，并不是说不能逆向破解；只是说难度增加，需要耐心。

（2）dex 文件加密

dex 是 Android 工程中的代码资源文件,通过 dex 可以反编译出 java 代码。dex 的加壳是常见的加密方式。通过对 dex 文件加密拼接加壳，可以有效的对工程代码进行保护。apk 工程在安装成功后，app 启动时会有 dex 解密的过程，然后重新加载解密后的 dex 文件。

1.2 Jni 层级加密

原理是在 jni 层， 使用 DexClassLoader 动态加载技术完成对加密 classex.dex 的动态加载，dex 文件可以附属在 assert 或 raw 目录。

2 脱壳分析

Android APP 脱壳的本质就是对内存中处于解密状态的 dex 的 dump。不管是函数抽取、dex2c 还是 vmp 壳，首要做的就是对整体 dex 的 dump，然后再对脱壳下来的 dex 进行修复。要达到对 apk 的脱壳，最为关键的就是准确定位内存中解密后的 dex 文件的起始地址和大小。那么这里要达成对 apk 的成功脱壳，就有两个最为关键的要素：

（1）内存中 dex 的起始地址和大小，只有拿到这两个要素，才能够成功 dump 下内存中的 dex

（2）脱壳时机，只有正确的脱壳时机，才能够 dump 下明文状态的 dex。否则，时机不对，及时是正确的起始地址和大小，dump 下来的也可能只是密文。

2．Android 加壳技术历程及加壳技术识别

2.1 加壳历程

2.1.1 第一代壳 Dex 整体加密

Dex 字符串加密、资源加密、对抗反编译、反调试、自定义 DexClassLoader

2.1.2 第二代壳 Dex 抽取与 So 加固

DEX 代码抽取到外部(类抽取加密按需解密和动态方法修改替换)，DEX 动态加载(分为利用 jni 和自定义 jni 即自定义底层函数)，SO 加密

2.1.3 第三代壳 Dex 动态解密与 So 混淆

DEX 动态解密及代码动态解密，SO 混淆及代码膨胀混淆

2.1.4 第四代壳 arm vmp（未来）

vmp 壳的识别

2.2 常用加固厂商特征

360：libprotectClass.so, libjiagu.so，360 基本上是把原始的 dex 加密存在了一个 so 中，加载之前解密。

通付盾：libegis.so。

网秦：libnqshield.so

百度：libbaiduprotect.so。把一些 class_data_item 拆走了，与阿里很像，同时它还会抹去 dex 文件的头部；它也会选择个别方法重新包装，达到调用前还原，调用后抹去的效果。我们可以通过对 DoInvoke (ART) 和 dvmMterp_invokeMethod (DVM)监控来获取到相关代码。

娜迦：libchaosvmp.so ， libddog.solibfdog.so

爱加密：libexec.so, libexecmain.so

梆梆：libsecexe.so, libsecmain.so , libDexHelper.so。把一堆 read,write, mmap 等 libc 函数 hook 了，防止读取相关 dex 的区域，爱加密的字符串会变，但是只是文件名变目录不变。

2.3 常用加固厂商脱壳点总结

2.3.1 脱掉梆梆的壳

assets/secData0.jar

lib/armeabi/libSecShell.so

lib/armeabi/libSecShell-x86.so

梆梆企业版

assets/classes0.jar

lib/armeabi-v7a/libDexHelper.so

lib/armeabi-v7a/ libDexHelper-x86.so

梆梆是把原 dex 文件加密放到了 secData0.jar，所以直接拿到 dex 文件，修复配置文件的程序入口点就可以重打包完美运行。通过还原加密算法，解密 secData0.jar，直接解压解密 jar 就是原 dex。secData0.jar 文件保存

在.cache 的 classes.dex 是加密的，主要是通过 hook 实现，打开时解密，关闭时加(open、mmap)。

dvmRawDexFileOpen

dexfileopenpartial

2.3.2 脱掉爱加密的壳

lib/armeabi/libexecmain.so

assets/ijiami.ajm

assets/af.bin

assets/signed.bin

assets/ijm_lib/armeabi/libexec.so

assets/ijm_lib/X86/libexec.so

fopen、fgets

bsd_signal

dexfileopenpartial

openDexFileNative

defineClassNative

2.3.3 脱掉 360 的壳

assets/.appkey

assets/libjiagu.so

assets/libjiagu_x86.so

1.识别 360 加固的代数

2.第一代 360 脱壳：xposed 插件脱壳

3.第二代 360 脱壳：so 手动 dump 并修复、mmap 手动脱壳、open+memcmp 手动脱壳

4.第三代 360 脱壳：drizzledumper 脱壳、dex2oat 脱壳

2.3.4 脱掉百度的壳

lib/armeabi/libbaiduprotect.so

assets/baiduprotect1.jar 或者 assets/baiduprotect.jar

1.加密流程:

采用动态加载 assets 下的 baiduprotect.jar。然后采用重写 onCreate,用 onCreate001 代替,onCreate 内容为修复 onCreate001 代码、执行 onCreate001 代码、清楚 onCreate001 代码。修复代码不能连续运行两次。

3.采用 Hook DexParse 来获取 Dex 相关数据，然后遍历 ClassDef 将所有 onCreate001 类直接解码。

4.Dump 出修复好的 Dex。

5.然而 Dump 的 Dex 还要修复(可以根据 ClassDef 自动修改)

Lcom/baidu/protect/A;->d(Ljava/lang/String;)V->解密方法

Lcom/baidu/protect/A;->e(Ljava/lang/String;)V->加密方法

Lcom/qsq/qianshengqian/XXXXX;->

onCreate001(Landroid/os/Bundle;)V->加解密传入参数

xdex 脱壳机与骗出 jni_onload 的 oncreate 抽取指令

vm 一维置换表和二维置换表

2.3.5 脱掉阿里的壳

assets/armeabi/libfakejni.so
assets/armeabi/libzuma.so
assets/libzuma.so
assets/libzumadata.so
assets/libpreverify1.so

2.3.6 脱掉腾讯的壳

腾讯加固
tencent_stub
lib/armeabi/libshella-xxxx.so
lib/armeabi/libshellx-xxxx.so
lib/armeabi/mix.dex

lib/armeabi/mixz.dex
腾讯御安全
assets/libtosprotection.armeabi-v7a.so
assets/libtosprotection.armeabi.so
assets/libtosprotection.x86.so
assets/tosversion
lib/armeabi/libtest.so
lib/armeabi/libTmsdk-xxx-mfr.so
特别注意：
apktool 助手伪加固特征：监测 application 即可
假 360 加固：没.appkey、application 为 com.qihoo.util.stub2678363137
假梆梆加固：application 为 com.secoen.apkwrapper.ApplicationWrapper

3．脱壳方法及脱壳时机

3.1 一代脱壳

（1）内存 dump 法 内存中寻找 dex.035、dey.036

找到后/proc/某个 pid/maps 中查找后，手动 dump

（2）Hook 法 Hook dvmDexFileOpenPartial （拿到两个参数 dump）

（3）定制系统 修改安卓源码并刷机,如 dexHunter

3.2 二代脱壳

（1）Hook 法

针对无代码抽取且 Hook dvmDexFileOpenPartial 失败, Hook dexFileParse

针对无代码抽取且 Hook dexFileParse 失败，Hook memcmp

（2）定制系统

修改安卓源码并刷机 - 针对无抽取代码

DexHunter

断点 mmap 调试，针对 Hook dexFileParse 无效

（3）静态脱壳机

分析 SO 壳逻辑并还原加密算法

自定义 linker 脱 SO 壳

3.3 三代脱壳

（1）dex2oat ART 模式下，dex2oat 生成 oat 文件时，内存中的 dex 是完整的。

（2）定制系统

Hook Dalvik_dalvik_system_DexFile_defineClassNative

枚举所有 DexClassDef,对所有的 Class，调用 dvmDefineClass 进行强制加载

dex 脱壳分析：

对于整体加密：真实完整的 dex 出现在内存中，把 dex 整体隐藏在 so 中。

对于抽取加密，真实、完整的 dex 从步出现在内存中，先从 so 里解密出一个待修复的 dex 文件，真正的 dex 从这个文件抽取部分。

3.4 脱壳时机

为了脱壳，需要建立一个概念，就是“时机”。对于非虚拟机壳，从内存中转储是一个最为有效和统用的技巧，那么就必须要找到一个时机，保证内存中的数据是完全正确的。

5.0 以下的脱壳时机

rewriteDex(u1* addr, int len, bool doVerify, bool doOpt,DexClassLookup** ppClassLookup,DvmDex** ppDvmDex)

dexSwapAndVerify(addr, len)

dvmDexFileOpenPartial(addr, len, &pDvmDex)dexFileParse(const u1* data, size_t length, int flags)

5.0~8.0 以下的脱壳时机

DexFile::OpenMemory(const byte* base ,size_t size, const std:string& location, uint32_tlocation_checksum,MemMap* mem_map)

8.0 及 8.0 以上的脱壳时机

DexFile:OpenCommon(const uint8_t* base, size_t size, const std:string& location, uint32_tlocation_checksum, const OatDexFile

```
* oat_dex_file, bool verify, bool verify_checksum,
std:string* error_msg, VerifyResult* verify_result)
```

4．脱壳工具介绍

4.1 Frida fart 脱壳

目前只能在 andorid8 上使用，该 frida 版 fart 是使用 hook 的方式实现的函数粒度的脱壳，仅仅是对类中的所有函数进行了加载，但依然可以解决绝大多数的抽取保护。

4.1.1 脱壳的步骤

（1）内存中 DexFile 结构体完整 dex 的 dump

（2）主动调用类中的每一个方法，并实现对应 CodeItem 的 dump

（3）通过主动调用 dump 下来的方法的 CodeItem 进行 dex 中被抽取的方法的修复

4.1.2 脱壳工具下载

电脑需要装好 python 环境和 frida 环境。参考链接：https://blog.csdn.net/qq_38044574/article/details/107788122

GitHub 地址下载：https://github.com/hanbinglengyue/FART，下载 frida_fart.zip 即可

（1）解压 frida_fart.zip

（2）将目录中的 fart.so 与 fart64.so 推送到 /data/app 目录下并使用 chmod 777

（3）需要以 spawn 方式启动 app，等待 app 进入 Activity 界面后，执行 fart() 函数即可。如 app 包名为 com.example.test,则

```
frida -U -f com.example.test -l frida_fart_hook.js --no-pause
```

4.2 Fdex2 脱壳

通过 Hook ClassLoader 的 loadClass 方法，反射调用 getDex 方法取得 Dex(com.android.dex.Dex 类对象)，再将里面的 dex 写出，代码十分简单，就 hook 了一个方法而已，可对抗 Android 的 dex 自加载型壳，需要 xposed 环境。

4.2.1 脱壳环境

（1）安卓手机 root ，必须 root，记住是必须，只支持 6.0 或者更低的版本

（2） Xposed 的安装，使用

下载地址：https://089u.com/dir/3843664-40606878-902c5f

4.2.2 核心代码：


```
public class MainHook implements IXposedHookLoadPackage {

    XSharedPreferences xsp;
    Class Dex;
    Method Dex_getBytes;
    Method getDex;
    String packagename;
    public void handleLoadPackage(XC_LoadPackage.LoadPackageParam
lpparam) throws Throwable {
        xsp = new XSharedPreferences("com.ppma.appinfo", "User");
        xsp.makeWorldReadable();
        xsp.reload();
        initRefect();
        packagename = xsp.getString("packagename", null);
        XposedBridge.log("设定包名："+packagename);
        if
((!lpparam.packageName.equals(packagename))||packagename==null) {
            XposedBridge.log("当前程序包名与设定不一致或者包名为空");
            return;
        }
        XposedBridge.log("目标包名："+lpparam.packageName);
        String str = "java.lang.ClassLoader";
        String str2 = "loadClass";
```

```
        XposedHelpers.findAndHookMethod(str, lpparam.classLoader,
str2, String.class, Boolean.TYPE, new XC_MethodHook() {
            protected void afterHookedMethod(MethodHookParam param)
throws Throwable {
                super.afterHookedMethod(param);
                Class cls = (Class) param.getResult();
                if (cls == null) {
                    //XposedBridge.log("cls == null");
                    return;
                }
                String name = cls.getName();
                XposedBridge.log("当前类名: " + name);
                byte[] bArr = (byte[])
Dex_getBytes.invoke(getDex.invoke(cls, new Object[0]), new
Object[0]);
                if (bArr == null) {
                    XposedBridge.log("数据为空: 返回");
                    return;
                }
                XposedBridge.log("开始写数据");
                String dex_path = "/data/data/" + packagename + "/" +
packagename + "_" + bArr.length + ".dex";
                XposedBridge.log(dex_path);
                File file = new File(dex_path);
                if (file.exists()) return;
                writeByte(bArr, file.getAbsolutePath());
            }
            } );
    }

    public void initRefect() {
        try {
            Dex = Class.forName("com.android.dex.Dex");
            Dex_getBytes = Dex.getDeclaredMethod("getBytes", new
Class[0]);
            getDex =
Class.forName("java.lang.Class").getDeclaredMethod("getDex", new
Class[0]);
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (NoSuchMethodException e) {
            e.printStackTrace();
        }
    }
```

```
    public  void writeByte(byte[] bArr, String str) {
        try {
            OutputStream outputStream = new FileOutputStream(str);
            outputStream.write(bArr);
            outputStream.close();
        } catch (IOException e) {
            e.printStackTrace();
            XposedBridge.log("文件写出失败");
        }
    }
}
```

参考链接：

https://bbs.pediy.com/thread-252630.htm

https://bbs.pediy.com/thread-263290.htm

https://juejin.cn/post/6844903449004343303#heading-5

https://bbs.pediy.com/thread-226216.htm

知识星球

优惠券

Andy

送你一张星球优惠券

移动安全

49 元立减金

限前 20 名加入星球使用

2022/04/15 12:00 至 2022/04/30 12:00

长按扫码领取优惠

加入星球立减