Dex简单保护

yong夜 / 2019-07-30 09:10:00 / 浏览数 4513 安全技术 移动安全 顶(0) 踩(0)

[TOC]

加固原理

利用加密算法将原始APK文件加密保护起来，然后将加密后的数据附加到壳尾部或其他地方，当运行起来的时候，壳会把动态解密加载起来。

加固流程

1. 编写解密并加载原始APK的壳App，生成对应的壳Dex
2. 加密原始APK/DEX文件，写入到壳Dex尾部或者存放到其他目录下
3. 修改壳Dex相应字段和被加固App的AndroidManifest.xml文件添加Application组件从壳App启动
4. 删除签名文件、替换dex文件、添加so文件等

壳App

1.解密出Dex/APK/JAR文件（为了简单化，这里暂时不用加密）

2.替换dexclassloader对象

3.执行原始APK的Application类

```
package com.stub;

import android.app.Application;
import android.app.Instrumentation;
import android.content.Context;
import android.content.pm.ApplicationInfo;
import android.util.Log;

import com.stub.utils.FileUtils;
import com.stub.utils.RefInvoke;
import java.io.File;
import java.lang.ref.WeakReference;
import java.util.List;
import java.util.Map;
import dalvik.system.DexClassLoader;

public class shell extends Application {

   private Application mApp = this;
   private File LIBS;
   private static final String DEFAULT_APPLICATION = "android.app.Application";


   @Override
   protected void attachBaseContext(Context base) {
       super.attachBaseContext(base);

       /*1. ■■■■Dex■■■■■■■■■■classes.dex■■app-libs■■■*/
       LIBS = mApp.getDir("libs", Context.MODE_PRIVATE);
       StringBuffer dexPathList = new StringBuffer();
       dexPathList.append(LIBS);
       dexPathList.append("/");
       dexPathList.append("classes.dex");
       System.out.printf("load classes.dex in %s", dexPathList);
       Log.i("shell", "load classes.dex in " + dexPathList);


       /* 2. ■■Dex■■*/
       String classActivityThread = "android.app.ActivityThread";
       String classLoadedApk = "android.app.LoadedApk";
```

```
        DexClassLoader loader;
        File nativeLib = new File(FileUtils.getParent(LIBS), "lib");

        Object activityThread = RefInvoke.invokeStaticMethod(classActivityThread, "currentActivityThread", new Class[]{}, new O

        String packageName = this.getPackageName();//■■apk■■■

        Map<?,?> mPackage = (Map<?,?>)RefInvoke.getField(activityThread, classActivityThread, "mPackages");

        WeakReference<?> wr = (WeakReference<?>) mPackage.get(packageName);

        loader = new DexClassLoader(dexPathList.toString(), mApp.getCacheDir().getAbsolutePath(), nativeLib.getAbsolutePath(),
        RefInvoke.setField(wr.get(), classLoadedApk, "mClassLoader", loader);

        /* 2. ■■application*/

        String main = "com.scoreloop.games.gearedub.GearApplication";
        String applicationName = main;
        //ActivityThread.currentActivityThread().mBoundApplication.info.mApplication = null;
        Object currentActivityThread = RefInvoke.invokeStaticMethod(classActivityThread, "currentActivityThread", new Class[]{}
        Object mBoundApplication = RefInvoke.getField(currentActivityThread, classActivityThread, "mBoundApplication");
        Object loadedApkInfo = RefInvoke.getField(mBoundApplication, classActivityThread+"$AppBindData", "info");
        RefInvoke.setField(loadedApkInfo, classLoadedApk, "mApplication", null);

        //currentActivityThread.mAllApplications.remove(currentActivityThread.mInitialApplication)
        Object mInitApplication = RefInvoke.getField(currentActivityThread, classActivityThread, "mInitialApplication");
        List<Application> mAllApplications = (List<Application>) RefInvoke.getField(currentActivityThread, classActivityThread,
        mAllApplications.remove(mInitApplication);

        //(LoadedApk) loadedApkInfo.mApplicationInfo.className = applicationName
        ((ApplicationInfo) RefInvoke.getField(loadedApkInfo, classLoadedApk, "mApplicationInfo")).className = applicationName;

        //(ActivityThread$AppBindData) mBoundApplication.appInfo.className = applicationName
        ((ApplicationInfo) RefInvoke.getField(mBoundApplication, classActivityThread+"$AppBindData", "appInfo")).className = ap

        //currentActivityThread.mInitApplication = loadedApkInfo.makeApplication(false, null)
        Application makeApplication = (Application) RefInvoke.invokeMethod(loadedApkInfo, classLoadedApk, "makeApplication", ne
        RefInvoke.setField(currentActivityThread, classActivityThread, "mInitialApplication", makeApplication);

        //currentActivityThread.mProviderMap
        Map<?,?> mProviderMap = (Map<?,?>) RefInvoke.getField(currentActivityThread, classActivityThread, "mProviderMap");
        for (Map.Entry<?, ?> entry : mProviderMap.entrySet()) {
            Object providerClientRecord = entry.getValue();
            Object mLocalProvider = RefInvoke.getField(providerClientRecord, classActivityThread+"$ProviderClientRecord", "mLoc
            RefInvoke.setField(mLocalProvider, "android.content.ContentProvider", "mContext", makeApplication);
        }
        makeApplication.onCreate();
    }
}
```

加密Dex文件

这里为了方便起见，不进行加密操作，直接将原始dex文件手动上传到/data/data/<packagename>/app-libs目录下等待壳App进行加载</packagename>

修改AndroidManifest.xml文件

1.使用apktool将原始apk进行反编译

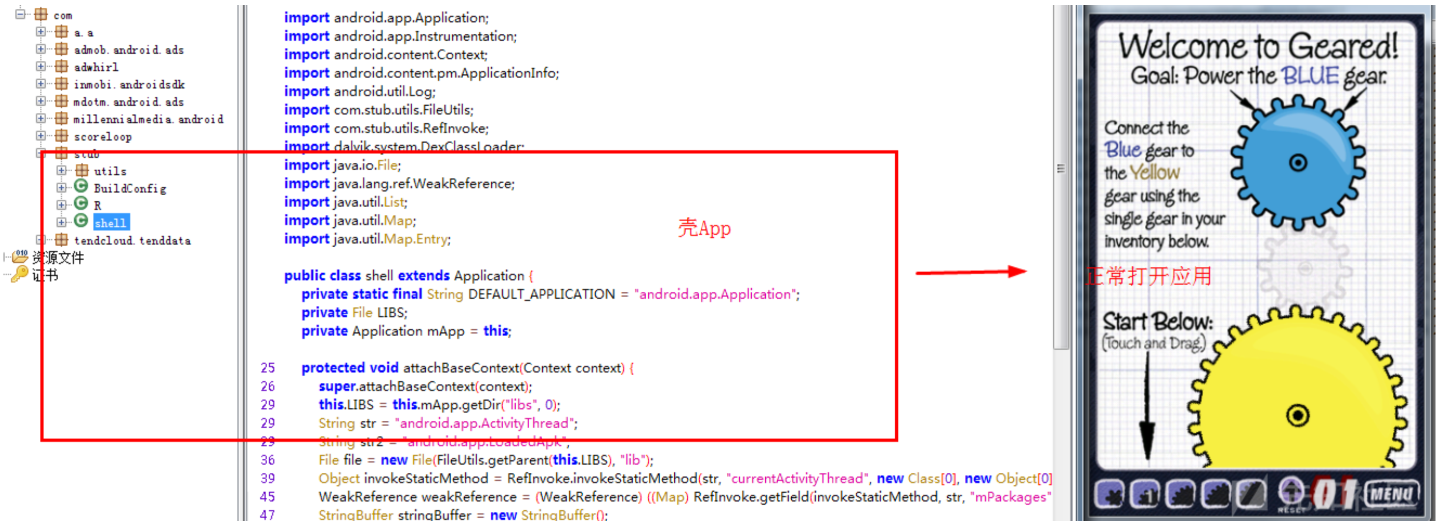
2.修改Manifest文件中的application组件的android:name属性值为壳App的application

3.重打包、签名

```
<application android:description="@string/app_description" android:icon="@drawable/icon" android:label="@string/app_label" and
```

结果

加固成功，应用正常打开执行



```java
import android.app.Application;
import android.app.Instrumentation;
import android.content.Context;
import android.content.pm.ApplicationInfo;
import android.util.Log;
import com.stub.utils.FileUtils;
import com.stub.utils.RefInvoke;
import dalvik.system.DexClassLoader;
import java.io.File;
import java.lang.ref.WeakReference;
import java.util.List;
import java.util.Map;
import java.util.Map.Entry;

public class shell extends Application {
    private static final String DEFAULT_APPLICATION = "android.app.Application";
    private File LIBS;
    private Application mApp = this;

    protected void attachBaseContext(Context context) {
        super.attachBaseContext(context);
        this.LIBS = this.mApp.getDir("libs", 0);
        String str = "android.app.ActivityThread";
        String str2 = "android.app.LoadedApk";
        File file = new File(FileUtils.getParent(this.LIBS), "lib");
        Object invokeStaticMethod = RefInvoke.invokeStaticMethod(str, "currentActivityThread", new Class[0], new Object[0]
        WeakReference weakReference = (WeakReference) ((Map) RefInvoke.getField(invokeStaticMethod, str, "mPackages"
        StringBuffer stringBuffer = new StringBuffer();
```

参考

【1】github加壳开源项目：https://github.com/kavmors/ApkSheller
【2】CSDN加壳原理：https://blog.csdn.net/jiangwei0910410003/article/details/48415225
【3】详细加壳步骤https://github.com/Herrrb/DexShell