Android最强黑科技-Xposed

1.Xposed简介

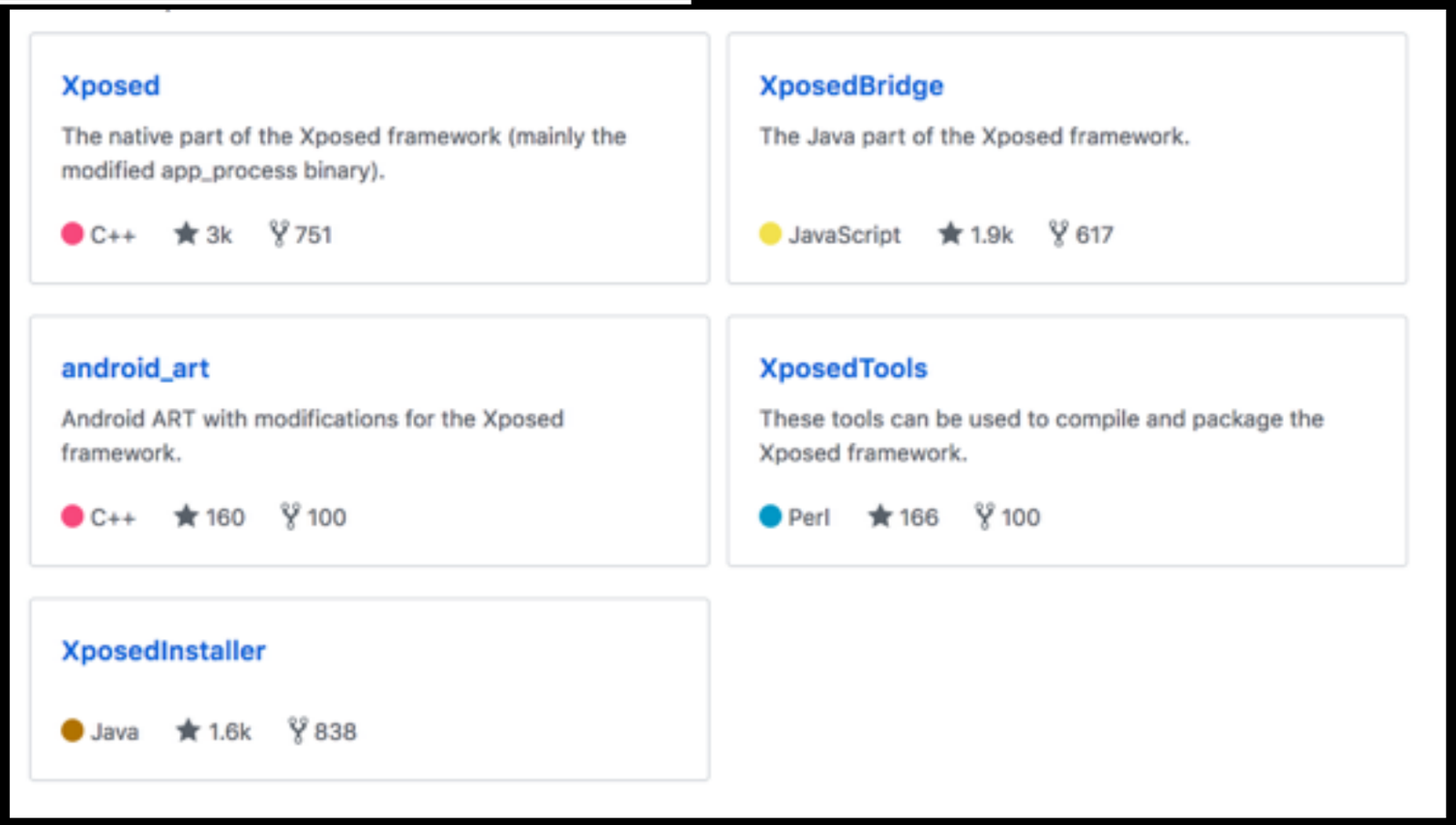
2.如何开发Xposed模块

3.Demo:去掉小米桌面的广告

4.扩展

Xposed简介

Xposed框架是一款可以在不修改APK的情况下影响程序运行(修改系统)的框架服务，基于它可以制作出许多功能强大的模块，且在功能不冲突的情况下同时运作。https://github.com/rovo89



应用场景

- qq、微信抢红包
- 消息防撤回
- 阻止运行、绿色守护、黑狱之类的
- 修改系统变量
- 去除各种广告
- …

使用方法

- root
- 安装Xposed
- 酷安市场或者Xposed installer里安装模块
- 勾选，重启生效

如何开发Xposed模块

- https://github.com/rovo89/XposedBridge/wiki/Development-tutorial
- XposedBridge，Xposed Framework API
- 找到突破点

去除小米桌面广告

能想到什么方法?

1.获取小米Home的代码，odex->smail->dex->jar,

https://github.com/JesusFreke/smali (https://bitbucket.org/JesusFreke/smali/downloads/)

https://github.com/pxb1988/dex2jar

2.找到Activity

3.找到对应的实体类

4.替换掉这个方法

Xposed原理

- https://github.com/rovo89/XposedBridge/wiki/Development-tutorial
- http://blog.csdn.net/innost/article/details/50461783, 邓凡平

How Xposed works

Before beginning with your modification, you should get a rough idea how Xposed works (you might skip this section though if you feel too bored). Here is how:

There is a process that is called "Zygote". This is the heart of the Android runtime. Every application is started as a copy ("fork") of it. This process is started by an `/init.rc` script when the phone is booted. The process start is done with `/system/bin/app_process`, which loads the needed classes and invokes the initialization methods.

This is where Xposed comes into play. When you install the framework, an extended app_process executable is copied to `/system/bin`. This extended startup process adds an additional jar to the classpath and calls methods from there at certain places. For instance, just after the VM has been created, even before the `main` method of Zygote has been called. And inside that method, we are part of Zygote and can act in its context.

The jar is located at `/data/data/de.robv.android.xposed.installer/bin/XposedBridge.jar` and its source code can be found here. Looking at the class [XposedBridge](https://github.com/rovo89/XposedBridge/blob/master/src/de/robv/android/xposed/XposedBridge.java), you can see the `main` method. This is what I wrote about above, this gets called in the very beginning of the process. Some initializations are done there and also the modules are loaded (I will come back to module loading later).

Method hooking/replacing

What really creates the power of Xposed is the possibility to "hook" method calls. When you do a modification by decompiling an APK, you can insert/change commands directly wherever you want. However, you will need to recompile/sign the APK afterwards and you can only distribute the whole package. With the hooks you can place with Xposed, you can't modify the code inside methods (it would be impossible to define clearly what kind of changes you want to do in which place). Instead, you can inject your own code before and after methods, which are the smallest unit in Java that can be addressed clearly.

XposedBridge has a private, native method `hookMethodNative`. This method is implemented in the extended `app_process` as well. It will change the method type to "native" and link the method implementation to its own native, generic method. That means that every time the hooked method is called, the generic method will be called instead without the caller knowing about it. In this method, the method `handleHookedMethod` in XposedBridge is called, passing over the arguments to the method call, the `this` reference etc. And this method then takes care of invoking callbacks that have been registered for this method call. Those can change the arguments for the call, change instance/static variables, invoke other methods, do something with the result... or skip anything of that. It is very flexible.

衍生物

- https://github.com/alibaba/dexposed 阿里早期的一个框架，可以用来做热修复，