**Sophix介绍与实践**

1. 介绍

移动热修复（Sophix）是阿里云提供的Android平台App热修复服务方案

1. Sophix优势



通过比较可以发现Sophix相比其他平台的实现更具优势

A．对于部分修改，补丁即时生效，不需要应用重启（这个还得看情况，较大的修改或者修改类结构之类的，会导致重启）

B．补丁包同样采用差异技术，生成的PATCH体积小

C．对应用无侵入，几乎没有性能损耗

D．傻瓜式接入

E．能够实现代码修复、资源修复、so修复等方面，因此更加全面的满足修复要求。

缺点：

1. 不支持四大组件的修复，这里的修复是新增，或更改。因为如果要修复四大组件，必须在AndroidManifest里面预先插入代理组件，并且尽可能声明所有的权限，这么做会给原先的App添加很多臃肿的代码，对app的侵入性很强，因此不支持四大组件新增和更改的修复，但是四大组件里面的代码是可以修复的。
2. Sophix原理简单介绍

Sophix的代码修复体系**同时涵盖了底层替换方案和类加载方案**，两种方案的结合，可以实现优势互补，完全兼容的作用，可以灵活的根据实际情况启动切换。在生成补丁时，补丁工具会根据实际代码变动情况进行自动选择，针对小修改，在底层替换方案限制范围内的，就直接采用底层替换修复，这样可以做到代码修复即时生效，而对于代码修改超出底层替换限制的，会使用类加载替换，虽然及时性不是很好，但是可以达到热修复目的。

4.阿里巴巴提供的快速接入文档：

<https://help.aliyun.com/document_detail/53240.html?spm=5176.doc53238.6.546.T3HWbI>

1. 补丁工具下载地址

Window：

<http://ams-hotfix-repo.oss-cn-shanghai.aliyuncs.com/SophixPatchTool_windows.zip?spm=5176.doc53247.2.2.2ur263&file=SophixPatchTool_windows.zip>

Mac

<http://ams-hotfix-repo.oss-cn-shanghai.aliyuncs.com/SophixPatchTool_macos.zip?spm=5176.doc53247.2.1.2ur263&file=SophixPatchTool_macos.zip>

Linux

<http://ams-hotfix-repo.oss-cn-shanghai.aliyuncs.com/SophixPatchTool_linux.zip?spm=5176.doc53247.2.3.2ur263&file=SophixPatchTool_linux.zip>

1. 补丁工具使用文档

<https://help.aliyun.com/document_detail/53247.html?spm=5176.doc53248.6.548.lHabEj>

1. 调试工具下载地址

<http://ams-hotfix-repo.oss-cn-shanghai.aliyuncs.com/hotfix_debug_tool-release.apk?spm=5176.doc53247.2.4.2ur263&file=hotfix_debug_tool-release.apk>

1. 调试工具使用文档：

<https://help.aliyun.com/document_detail/53248.html?spm=5176.doc53247.6.549.zPEvTh>

**Demo实践**

在新建项目之前，需要提前在阿里云移动热修复中申请账号，并创建相应的App。审核通过后再进行剩余操作。

1. 新建Sophix工程
2. 在工程中配置Sophix依赖
3. 在根节点下配置阿里云仓库

repositories {  
 //阿里云仓库  
 maven {  
 url "http://maven.aliyun.com/nexus/content/repositories/releases"  
 }  
}

1. 在依赖(dependencies)中添加

compile 'com.aliyun.ams:alicloud-android-hotfix:3.0.5'

1. 本Demo为了兼容Android6.0，增加了v7兼容包

compile 'com.android.support:appcompat-v7:25.0.1'

1. 配置混淆

#基线包使用，生成mapping.txt  
**-printmapping** mapping.txt  
#生成的mapping.txt在app/buidl/outputs/mapping/release路径下，移动到/app路径下  
#修复后的项目使用，保证混淆结果一致  
#-applymapping mapping.txt  
#hotfix  
**-keep** class com.taobao.sophix.\*\*{\*;}  
**-keep** class com.ta.utdid2.device.\*\*{\*;}  
#防止inline  
**-dontoptimize**

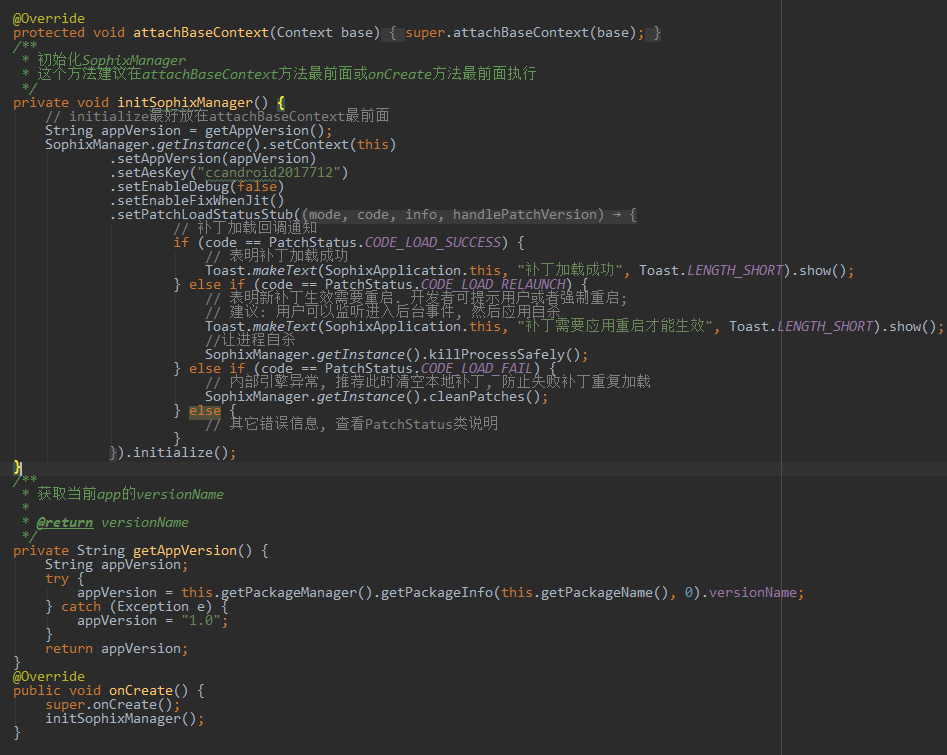
1. 配置相关权限

<!-- 网络权限 -->  
<uses-permission android:name="android.permission.INTERNET" />  
<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />  
<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" />  
<!-- 外部存储读权限，调试工具加载本地补丁需要 -->  
<uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />  
<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"/>

1. 在Application节点中添加阿里云热修复服务后台申请的App id ，App Secrets和RSA密钥

<meta-data  
 android:name="com.taobao.android.hotfix.IDSECRET"  
 android:value="app id" />  
<meta-data  
 android:name="com.taobao.android.hotfix.APPSECRET"  
 android:value="app secrets" />  
<meta-data  
android:name="com.taobao.android.hotfix.RSASECRET"  
android:value="app rsa密钥" />

1. 建立Application的子类，并在Manifest中添加。在Application子类的attachBaseContext方法最前面或者onCreate方法最前面完成SophixManager的初始化。



下面是初始化方法中的介绍（来自阿里巴巴Sohpix接入文档）

#### initialize方法

* initialize(): **<必选>**

该方法主要做些必要的初始化工作以及如果本地有补丁的话会加载补丁, 但不会自动请求补丁。因此需要自行调用queryAndLoadNewPatch方法拉取补丁。这个方法调用需要尽可能的早, 推荐在Application的onCreate方法中调用, initialize()方法调用之前你需要先调用如下几个方法, 方法调用说明如下:

* setContext(this): **<必选>** Application上下文context
* setAppVersion(appVersion): **<必选>** 应用的版本号
* setEnableDebug(true/false): **<可选>** 默认为false, 是否调试模式, 调试模式下会输出日志以及不进行补丁签名校验. 线下调试此参数可以设置为true, 查看日志过滤TAG:Sophix, 同时强制不对补丁进行签名校验, 所有就算补丁未签名或者签名失败也发现可以加载成功. 但是正式发布该参数必须为false, false会对补丁做签名校验, 否则就可能存在安全漏洞风险
* setEnableFixWhenJit(): **<可选>** 默认情况下会在Android N以后的版本发现jit后跳过，这会使得部分7.0以上设备不进行修复。而如果想要此时不跳过，需要打开这个选项进行配置。打开后，需要做对Application进行改造。要尽可能避免Application类与和它同包名的类互相访问，如果确实需要访问，接口应设为public权限，[详见常见问题文档](https://help.aliyun.com/knowledge_detail/55422.html)，也可寻求群里技术支持解决。Android 7.0后带来的jit问题很隐蔽，只有频繁使用的app会由系统进行jit，该接口可以彻底解决Android N带来的jit问题。
* setAesKey(aesKey): **<可选>** 用户自定义aes秘钥, 会对补丁包采用对称加密。这个参数值必须是16位数字或字母的组合，是和补丁工具设置里面AES Key保持完全一致, 补丁才能正确被解密进而加载。此时平台无感知这个秘钥, 所以不用担心阿里云移动平台会利用你们的补丁做一些非法的事情。
* setPatchLoadStatusStub(new PatchLoadStatusListener()): **<可选>** 设置patch加载状态监听器, 该方法参数需要实现PatchLoadStatusListener接口, 接口说明见1.3.2.2说明
* setUnsupportedModel(modelName, sdkVersionInt):**<可选>** 把不支持的设备加入黑名单，加入后不会进行热修复。modelName为该机型上Build.MODEL的值，这个值也可以通过adb shell getprop | grep ro.product.model取得。sdkVersionInt就是该机型的Android版本，也就是Build.VERSION.SDK\_INT，若设为0，则对应该机型所有安卓版本。

#### PatchLoadStatusListener接口

该接口需要自行实现并传入initialize方法中, 补丁加载状态会回调给该接口, 参数说明如下:

* mode: 补丁模式, 0:正常请求模式 1:扫码模式 2:本地补丁模式
* code: 补丁加载状态码, 详情查看PatchStatusCode类说明
* info: 补丁加载详细说明, 详情查看PatchStatusCode类说明
* handlePatchVersion: 当前处理的补丁版本号, 0:无 -1:本地补丁 其它:后台补丁

这里列举几个常见的code码说明,

* code: 1 补丁加载成功
* code: 6 服务端没有最新可用的补丁
* code: 11 RSASECRET错误，官网中的密钥是否正确请检查
* code: 12 当前应用已经存在一个旧补丁, 应用重启尝试加载新补丁
* code: 13 补丁加载失败, 导致的原因很多种, 比如UnsatisfiedLinkError等异常, 此时应该严格检查logcat异常日志
* code: 16 APPSECRET错误，官网中的密钥是否正确请检查
* code: 18 一键清除补丁
* code: 19 连续两次queryAndLoadNewPatch()方法调用不能短于3s

1. 在接口中可能会调用的方法：

#### killProcessSafely方法

可以在PatchLoadStatusListener监听到CODE\_LOAD\_RELAUNCH后在合适的时机，调用此方法杀死进程。注意，不可以直接Process.killProcess(Process.myPid())来杀进程，这样会扰乱Sophix的内部状态。因此如果需要杀死进程，建议使用这个方法，它在内部做一些适当处理后才杀死本进程。

#### cleanPatches()方法

清空本地补丁，并且不再拉取被清空的版本的补丁。

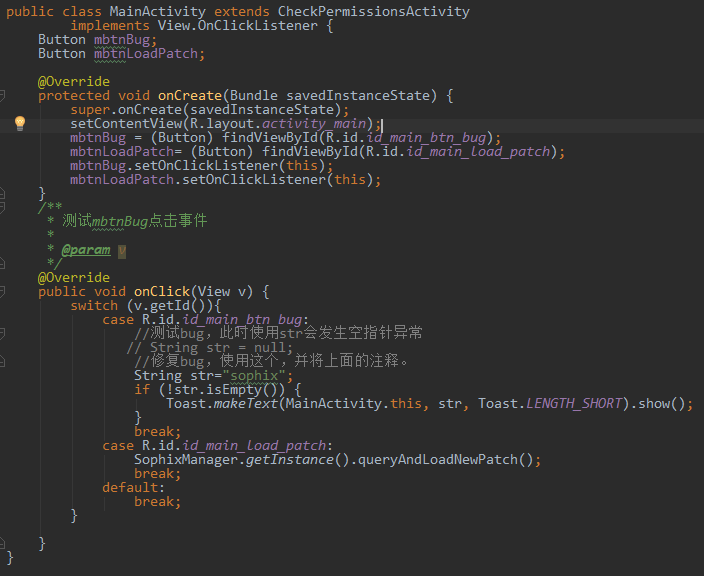
8. **补丁在后台发布之后, 并不会主动下行推送到客户端, 需要手动调用queryAndLoadNewPatch方法查询后台补丁是否可用.**

9. 在本demo中，通过按钮点击，调用**queryAndLoadNewPatch方法，实际开发视实际情况而定。**

10. 在布局中添加两个按钮，一个点击会Crash掉，一个用于**queryAndLoadNewPatch。**

11. 在MainActivity中，完成按钮初始化，并定义点击事件

内容如下：



再onClick方法中，R.id.id\_main\_btn\_bug:下

有问题的版本是：

使用String str = null;注释: String str="sophix";

修复后的版本是：

使用String str="sophix";注释: String str = null;

SophixManager.getInstance().queryAndLoadNewPatch()的详细介绍

该方法主要用于查询服务器是否有新的可用补丁. SDK内部限制连续两次queryAndLoadNewPatch()方法调用不能短于3s, 否则的话就会报code:19的错误码. 如果查询到可用的话, 首先下载补丁到本地, 然后

* 应用原本没有补丁, 那么如果当前应用的补丁是热补丁, 那么会立刻加载(不管是冷补丁还是热补丁). 如果当前应用的补丁是冷补丁, 那么需要重启生效.
* **应用已经存在一个补丁, 请求发现有新补丁后，本次不受影响。并且在下次启动时补丁文件删除, 下载并预加载新补丁。在下下次启动时应用新补丁。**

补丁在后台发布之后, 并不会主动下行推送到客户端, 需要手动调用queryAndLoadNewPatch方法查询后台补丁是否可用.

* 只会下载补丁版本号比当前应用存在的补丁版本号高的补丁, 比如当前应用已经下载了补丁版本号为5的补丁, 那么只有后台发布的补丁版本号>5才会重新下载.

同时1.4.0以上版本服务后台上线了“一键清除”补丁的功能, 所以如果后台点击了“一键清除”那么这个方法将会返回code:18的状态码. 此时本地补丁将会被强制清除, 同时不清除本地补丁版本号

12. 生成两款Apk，一款由bug，一款修复了bug。使用补丁工具生成补丁包

补丁工具使用见：

<https://help.aliyun.com/document_detail/53247.html?spm=5176.doc53240.6.548.p42vuJ>

13. 上传补丁包，阿里云后台创建的版本要和当前App版本一致。

14. 推荐操作：

发布前请严格按照：扫码内测 => 灰度发布 => 全量发布的流程进行，以保证补丁包能够正常在所有Android版本的机型上生效。为了保险起见，理论上应该对每个版本的android手机都测一遍是否生效会比较好。不过，其实只需测试通过以下具有代表性的Android版本就基本没什么大问题了：2.3、4.4、5.1、7.0

15. 调试工具（Apk，需要安装到手机上）

<http://ams-hotfix-repo.oss-cn-shanghai.aliyuncs.com/hotfix_debug_tool-release.apk?spm=5176.doc53248.2.1.apER0c&file=hotfix_debug_tool-release.apk>

16. 调试工具扫码内测操作流程见

<https://help.aliyun.com/document_detail/53248.html?spm=5176.doc53247.6.549.Xbsn42>

17. 版本说明情况：

说明一：patch是针对客户端具体某个版本的，patch和具体版本绑定

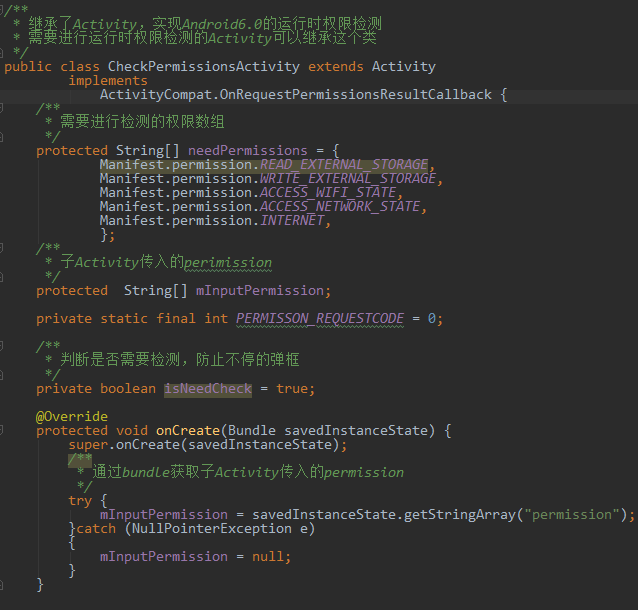
* eg. 应用当前版本号是1.1.0, 那么只能在后台查询到1.1.0版本对应发布的补丁, 而查询不到之前1.0.0旧版本发布的补丁.

说明二：针对某个具体版本发布的5新补丁, 必须包含所有的bugfix, 而不能依赖补丁递增修复的方式, 因为应用仅可能加载一个补丁

* eg. 针对1.0.0版本在后台发布了一个补丁版本号为1的补丁修复了bug1, 然后发现此时针对这个版本补丁1修复的不完全, 代码还有bug2, 在后台重新发布一个补丁版本号为2的补丁, 那么此时补丁2就必须同时包含bug1和bug2的修复才行, 而不是只包含bug2的修复(bug1就没被修复了)

18. 其中MainActivity继承的父类CheckPermissionActivity主要是进行动态权限的检查。

内容如下：





19. 运行程序



1.点击第一个按钮bug，应用会由于NPE而Crash。

2.点击load patch查询后台是否有可用补丁，应用会因为补丁需要冷启动才能生效，而进行应用自杀，强制让用户重启。

3.重启应用后，再次点击bug，NPE问题已经修复。