



Android热修复技术原理详解与升级探索

手机淘宝资深无线开发工程师 甘晓霖(万壑) 2017.6







有什么办法可以使客户端与开发节奏最快同步?

靠用户手动下载新版本更新? 太慢

Hybrid方案,如WEEX、RN? 跨语言

插件化方案,如Atlas、DroidPlugin? 代码重构

或者,更简单的方式?







阿里云移动热修复框架Sophix

- 云端下发补丁包到app,直接修复
- 用户无需手动更新app , 无任何感知。

补丁工具

服务端配置

运行时框架

生成需要更新的部分

云端控制

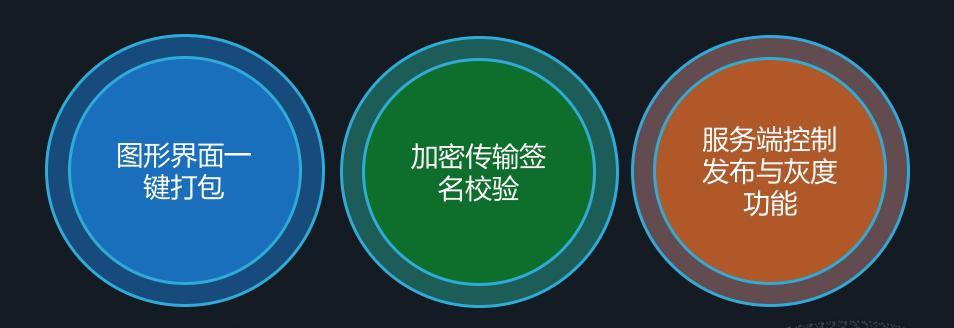
应用生效







阿里云移动热修复框架Sophix









热修复框架涵盖的范围









目录 content

- 1 代码修复
- 2 资源修复
- 3 SO库修复
- 4 竞品对比
- 5 快速上手







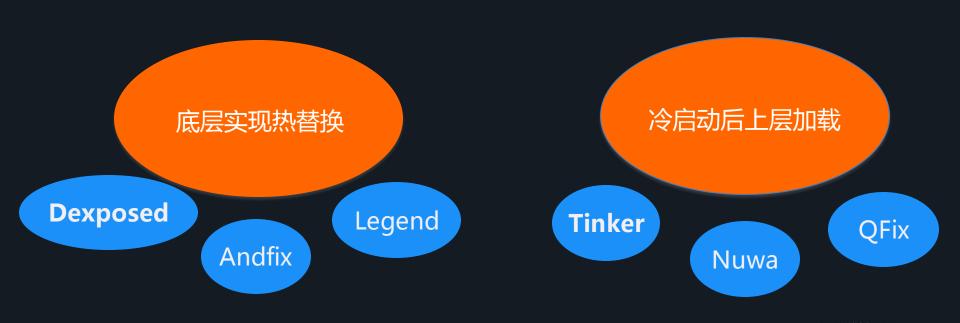
代码修复







代码修复两大主要流派









传统方式:底层结构固定字段的替换

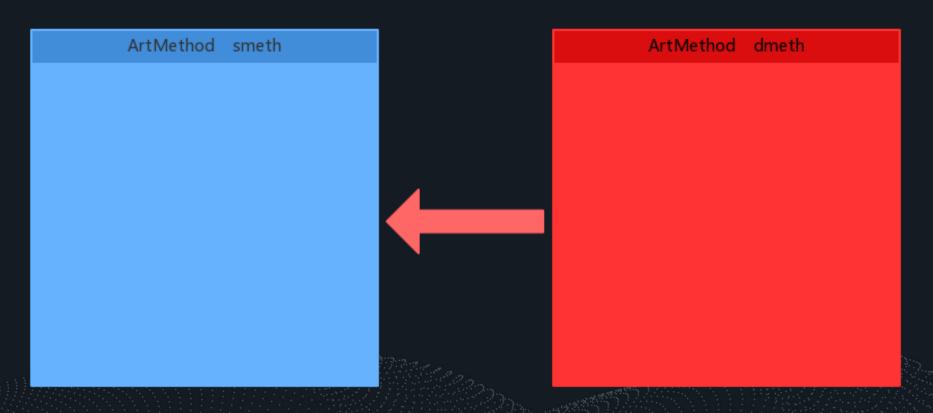
ArtMethod smeth	
declaring_class_	←
dex_cache_resolved_methods_	—
dex_cache_resolved_types_	
access_flags_	—
dex_code_item_offset_	
dex_method_index_	-
method_index_	-
entry_point_from_interpreter_	
entry_point_from_ini_	
entry_point_from_quick_compiled_code_	No.

ArtMethod dmeth declaring_class_ dex_cache_resolved_methods_ dex_cache_resolved_types_ access_flags_ dex_code_item_offset_ dex_method_index_ method index entry_point_from_interpreter_ entry_point_from_jni_ entry_point_from_quick_compiled_code_





新的思路:整体替换









ArtMethod Array

ArtMethod
ArtMethod

获取方法大小的妙计

size of a ArtMethod

取得相邻方法的起始地址,相减即可!







热替换方案的局限之处

- •引起原有了类中发生结构变化的修改。
- •静态块中的代码,也就是<clinit>方法。
- •修复了的非静态方法会被反射调用。





冷启动方案

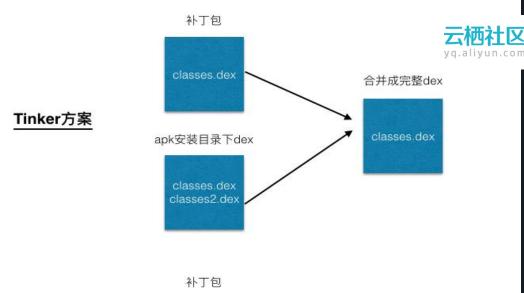
作为Plan B。补足热替换方案不适用的情况。

由此完成所有情况的代码热修复。

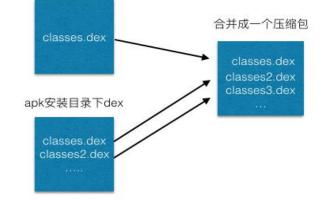




以Art为例:利用Android原生Multi-dex功能合成新包。



新方案





热替换方式

时效快,效率高

代码改动小,满足代码变动条件的,打包工具自动判定为采用热替换修复。

冷启动修复

完整更新代码

代码改动大,代码的变动情况比较复杂。







资源修复







仍然有两大主要流派

修改AAPT, 打基础包时就已经事先留空。

会侵入打包流程。要修改原生SDK中的aapt,且不利于兼容以 后的版本。

参照Instatnt Run的实现,完整替换AssetManager

要合成新资源包,替换整个AssetManager,操作太重。

补丁包尽可能地小。加载补 丁迅速,性能好,内存和时 间消耗极小。

不改变打包流程,传^{yq.aliyun.com} sdk工具链的完整性。

一个优秀的资源更 新方案应做到

开发透明,开发者无感知

。不需要事先固定资源id

方便易用,傻瓜式操作。 键完成patch工作。







我们实现的资源修复方案

type 0x01	attr		type 0x01	attr
0x7f010000	subtitle		0x7f010000	subtitle
0x7f010001	expanded		0x7f010001	expanded
0x7f010002	textAllCaps		0x7f010002	textAllCaps
0x7f010004	homeLayout		0x7f010004	homeLayout
0x7f010004	isLightTheme		0x7f010004	isLightTheme
type 0x02	drawable		type 0x02	drawable
0x7f020000	d_background		0x7f020000	d_background
0x7f020001	holo_dark		0x7f020001	holo_dark
0x7f020002	holo_light	\vdash	0x7f020002	holo_grey
type 0x03	layout	│ └ ▶	0x7f020003	holo_light
0x7f030000	activity_main	_	type 0x03	layout
0x7f030001	content_main	ـا ا	0x7f030000	activity_main
0x7f030002	nav_menu	X	0x7f030001	content_main
0x7f030003	dropdn_item	X	0x7f030002	dropdn_item2
0x7f030004	abc_dialog	-	0x7f030003	abc_dialog
type 0x04	string		type 0x04	string
0x7f040000	app_name		0x7f040000	app_name
0x7f040002	no		0x7f040002	no
0x7f040002	dialog_title		0x7f040003	dialog_title

type 0x01 drawable
0x66010000 holo_grey
type 0x02 layout
0x66020000 activity_main
0x66020001 dropdn_item2
type 0x03 string
0x66030000 no

patch

old resources

new resources







资源更新方案特点

- •直接应用补丁就可以生效。
- •不修改AAPT,不侵入打包。
- •资源包差量,不需重新合成。
- •直接根据新旧包进行解析重构生成。







SO库修复



云栖社区

反射注入方案,重 启生效。

不侵入开发流程,对 开发者透明。









竞品对比





Sophix与目前市面上已有热修复集成方案对比



方案对比	Sophix	Tinker	Amigo
DEX修复	同时支持即时生效修复和冷启动修复	冷启动修复	冷启动修复
资源更新	差量包,不用合成	差量包,需要合成	全量包,不用合成
SO库更新	插桩实现,开发透明	替换接口,开发不透明	插桩实现,开发透明
性能损耗	低,仅冷启动情况下有些损耗	高,有合成操作	低,全量替换
生成补丁	直接选择已经编好的新旧包在本地生成	编译新包时设置基线包	上传完整新包到服务端
补丁大小	小	小	大
接入成本	傻瓜式接入	复杂	一般
Android版本	全部支持	全部支持	全部支持
安全机制	加密传输及签名校验	加密传输及签名校验	加密传输及签名校验
服务端支持	支持服务端控制	支持服务端控制	支持服务端控制







快速上手





接入Sophix,2行代码搞定!

// 初始化

SophixManager.getInstance().setContext(this).setAppVersion(appVersion).initialize();

// 请求补丁

SophixManager.getInstance().queryAndLoadNewPatch();







简洁易用,一键生成补丁。

	阿里SOPHIX热更新补丁工具				
旧包		选择			
新包		选择			
	日志 设置 高级	GO!			

选择旧包和新包,点击GO!



阿里云移动热修复框架Sophix

技术领先:

- •实现了目前市面上唯一不侵入打包流程的热修复。
- •在代码修复、资源修复、SO库修复方面,都做到了业界最佳。

安全易用:

- •更加完美的客户端服务端一体的热更新方案。
- •图形界面一键打包。
- •加密传输、签名校验。
- •服务端控制发布与灰度功能。







从现在起,让你的APP随心所欲地热更新!

阿里云移动热修复入口



https://www.aliyun.com/product/hotfix







乙天・智能