手机淘宝Dexposed
技术介绍

阿里巴巴集团 无线事业部 胡文江/白衣
邮箱：baiyi.hwj@alibaba - inc.com
2015.10.16

Geekbang>.
极客邦科技

全球领先的技术人学习和交流平台

InfoQ

专注中高端技术
人员的社区媒体

EGO EXTRA GEEKS' ORGANIZATION NETWORKS

高端技术人员
学习型社交网络

StuQ

实践驱动的IT职业
学习和服务平台

扫我，码上开启新世界

Geekbang>.
InfoQ | EGO NETWORKS | StuQ

InfoQ

促进软件开发领域知识与创新的传播

ArchSummit
全 球 架 构 师 峰 会

实践第一　案例为主

时间：2015年12月18-19日 / 地点：北京 · 国际会议中心

欢迎您参加ArchSummit北京2015，技术因你而不同

ArchSummit北京二维码

QCon
全球软件开发大会

[北京站]

2016年04月21日-23日

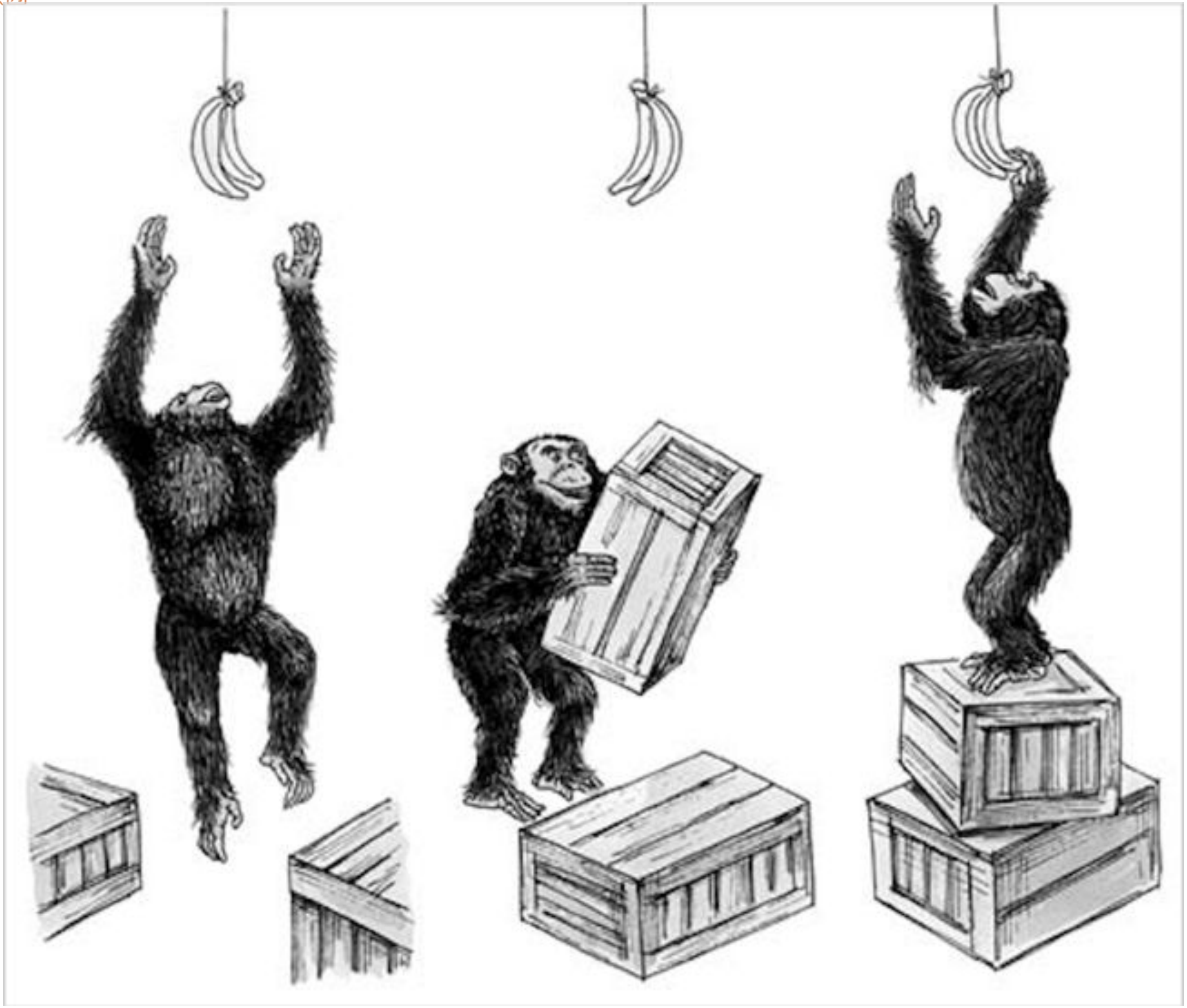
关注InfoQ官方信息
及时获取QCon演讲视频信息

关于我

- 2007：杭州虹软 移动多媒体
- 2010：阿里巴巴 手机淘宝
    - 负责手机淘宝Android客户端
    - 为阿里巴巴各条无线Android产品线提供基础技术和设施支撑

Agenda

- Dexposed 诞生前的故事
- Dexposed 的原理介绍
- Dexposed 在手淘内部的应用及一些 Case
- Dexposed 未来



->2013 2013->2014 2014->

缘起

- 端侧的动态性
  - H5
  - 服务器开关
  - jar 替换
  - 脚本

缘起

- 端侧的动态性方案比较

端侧的动态性	优缺点
H5	灵活，复杂交互性能差
服务端开关项	需要程序预埋代码
Jar 包替换	实时性稍差，有冗余，资源无解
脚本	学习成本高，性能差

缘起

- 性能监控和优化
  - 侵入性问题
  - 可行性问题
- AOP(Aspect-Oriented Programming)
  - AspectJ
  - Java 动态代理

缘起

AOP(Aspect-Oriented Programming)

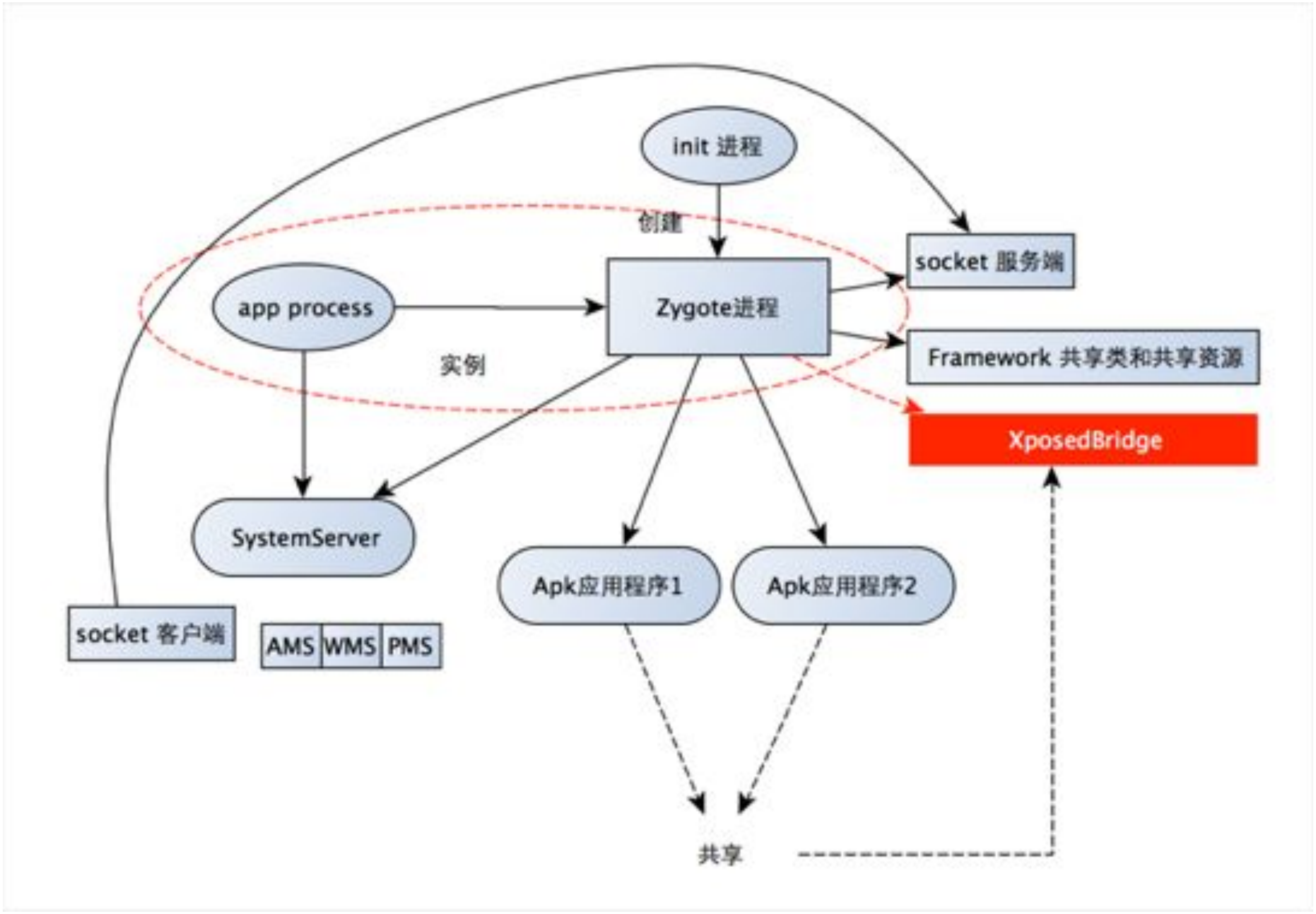
AOP 方案比较	缺点
AspectJ	特殊的语法 静态编译 编译时间长， 必须使用 AspectJ 编译器 对系统类无能为力
Java 动态代理	只能针对 interface 类

Xposed 和 Substrate

运行时AOP(Aspect-Oriented Programming)

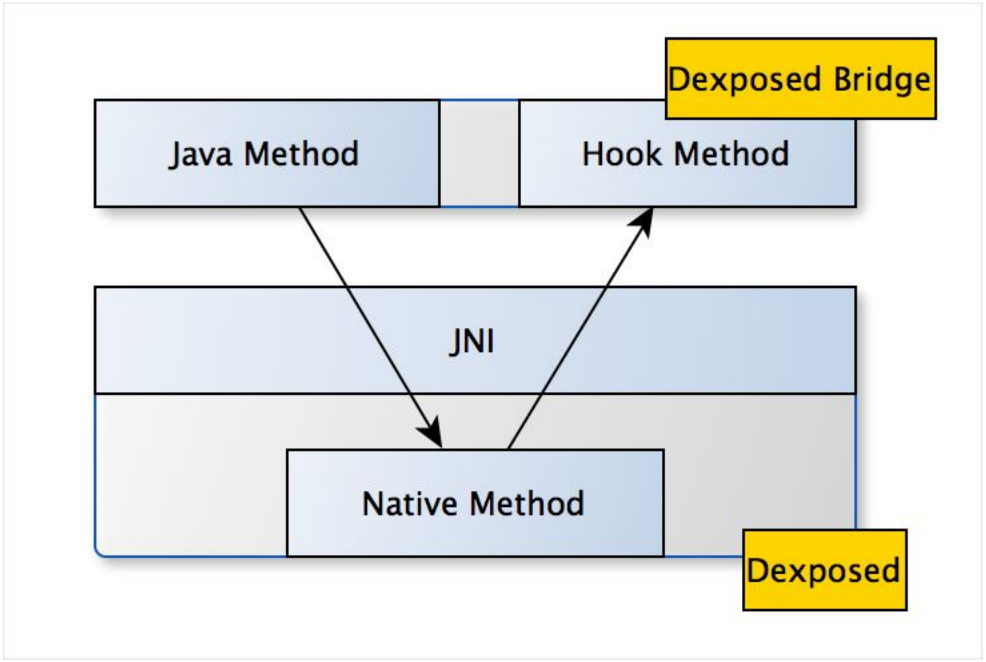
	Xposed	Substrate
AOP	✔	✔
ROOT	✔	✔
OpenSource	✔	✖
Java Method	✔	✔
Ndk Method	✖	✔

Xposed 原理



- 劫持 zygote
- 劫持 java method ->Dexposed

Dexposed 原理



- 运行时 / 方法级 / 性能损耗少 / 学习成本低 / 本进程

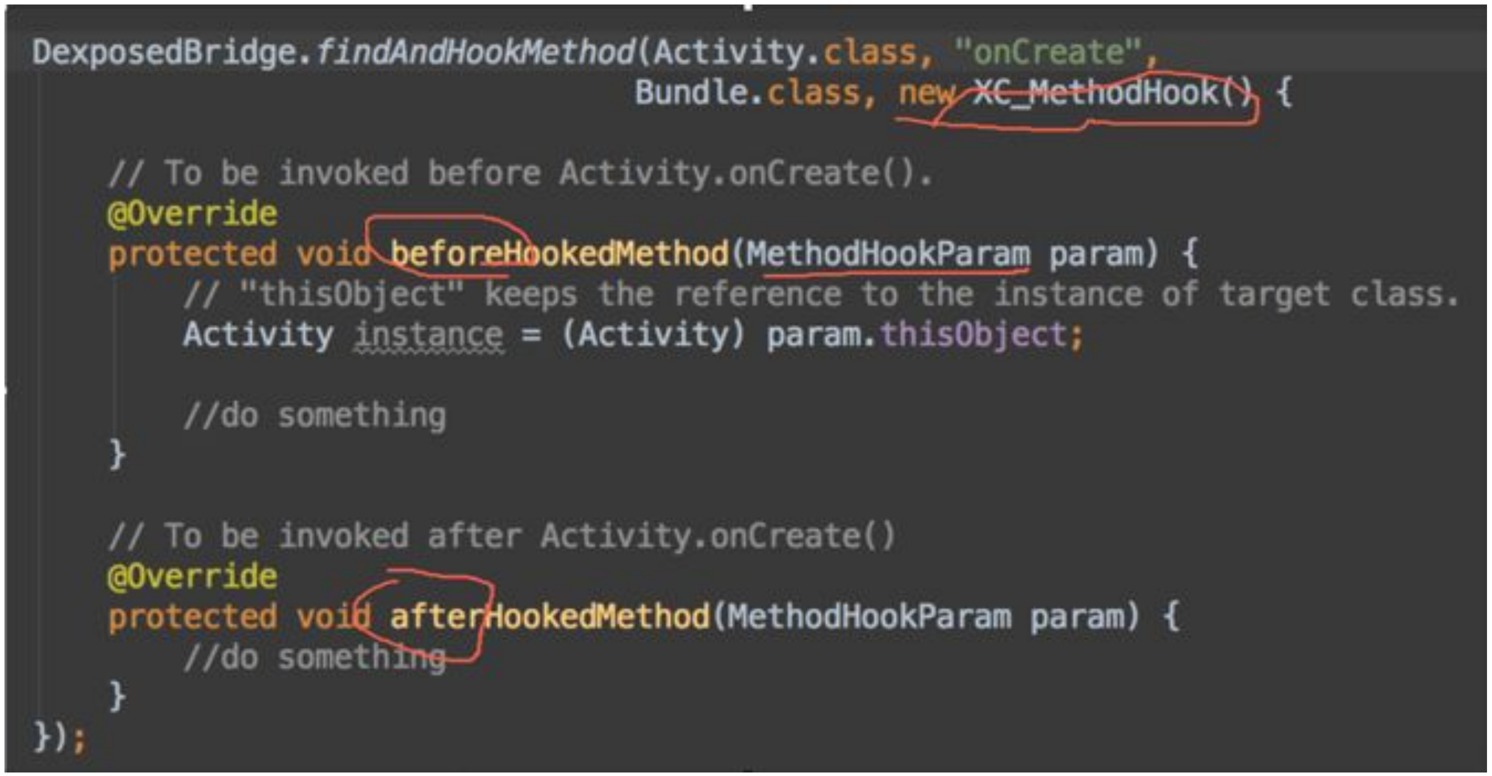
Dexposed 开源地址

- https://github.com/alibaba/dexposed

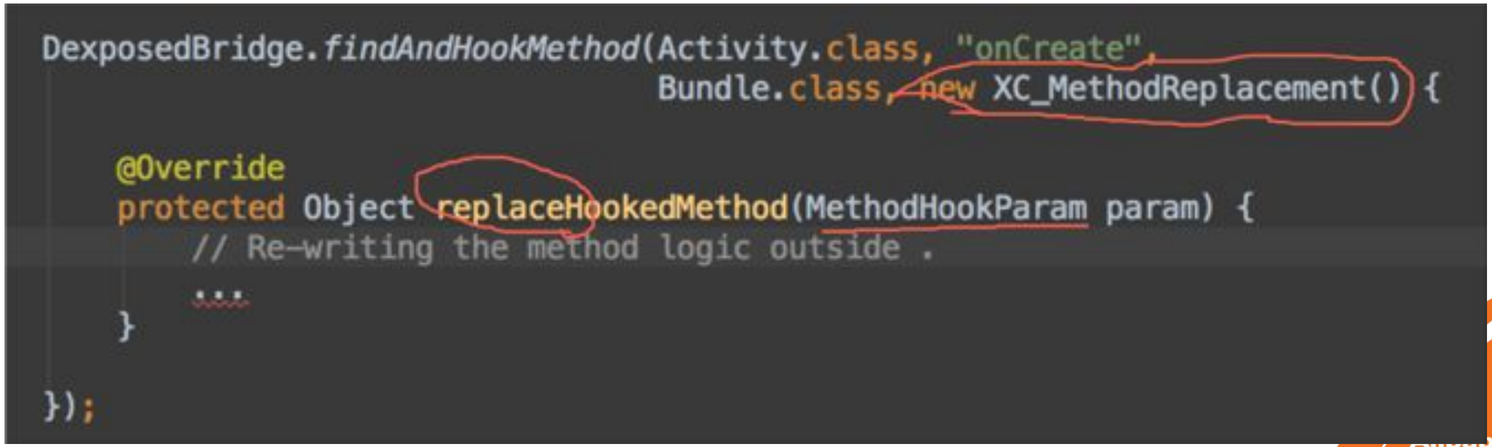
Dexposed 使用介绍

- Before
- After
- Replace

Dexposed 使用例子



```
DexposedBridge.findAndHookMethod(Activity.class, "onCreate",
                                 Bundle.class, new XC_MethodHook() {

    // To be invoked before Activity.onCreate().
    @Override
    protected void beforeHookedMethod(MethodHookParam param) {
        // "thisObject" keeps the reference to the instance of target class.
        Activity instance = (Activity) param.thisObject;

        //do something
    }

    // To be invoked after Activity.onCreate()
    @Override
    protected void afterHookedMethod(MethodHookParam param) {
        //do something
    }
});
```

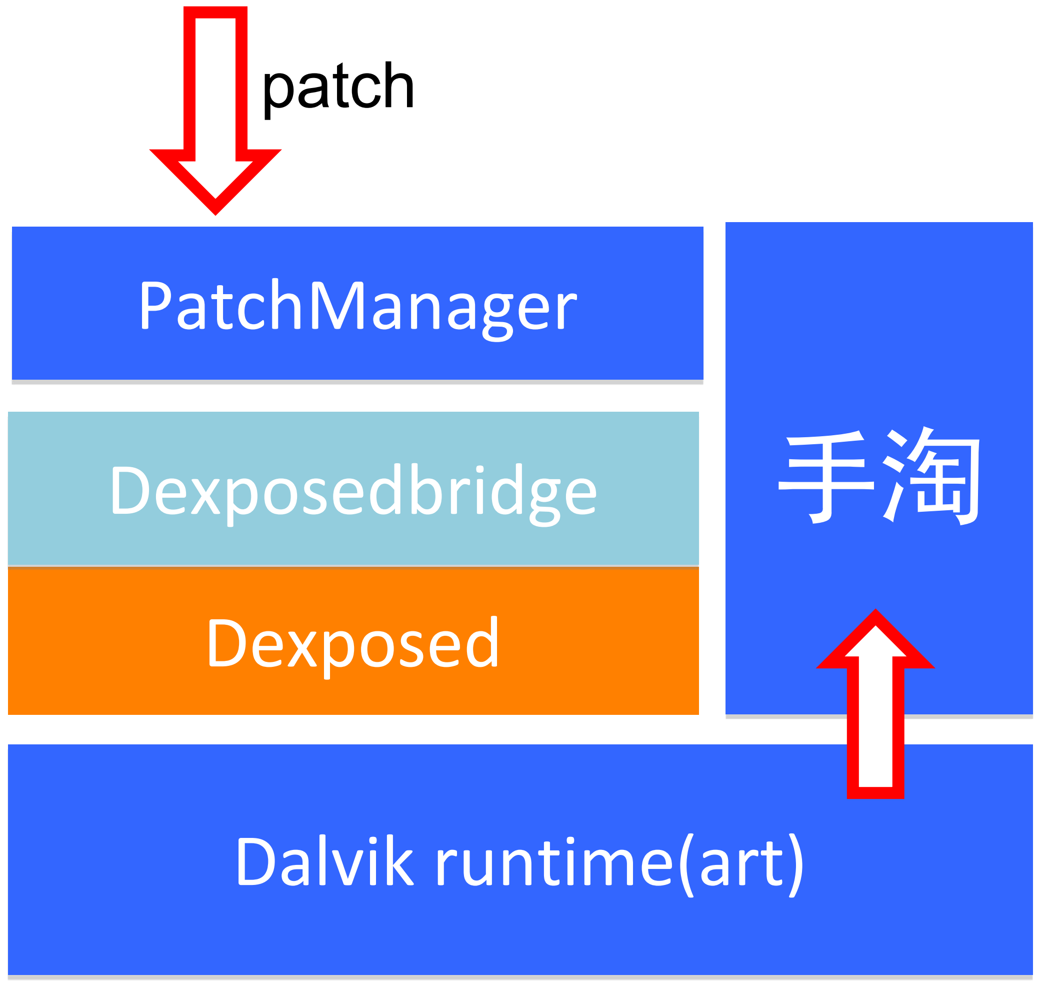


```
DexposedBridge.findAndHookMethod(Activity.class, "onCreate",
                                 Bundle.class, new XC_MethodReplacement() {

    @Override
    protected Object replaceHookedMethod(MethodHookParam param) {
        // Re-writing the method logic outside .
        ...
    }

});
```

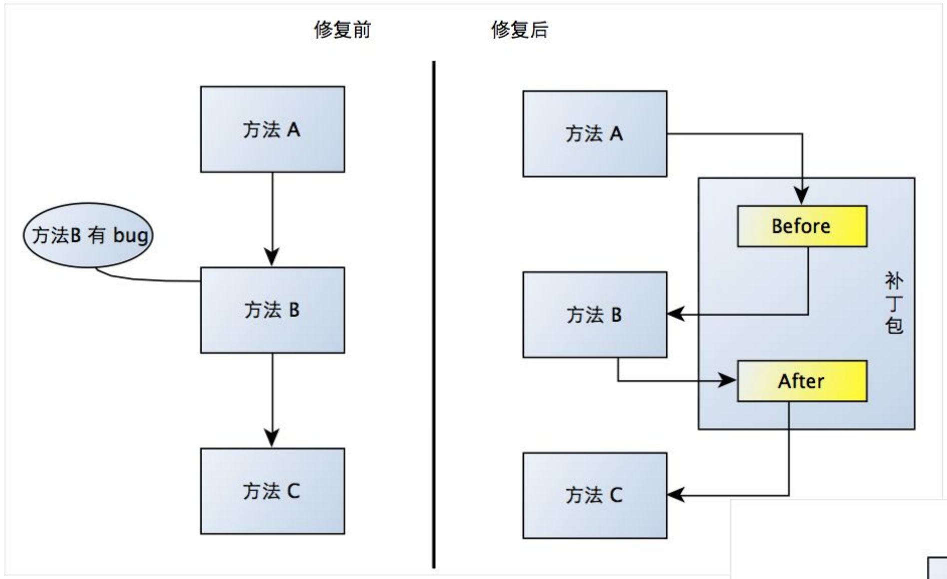
手淘内部使用 Dexposed

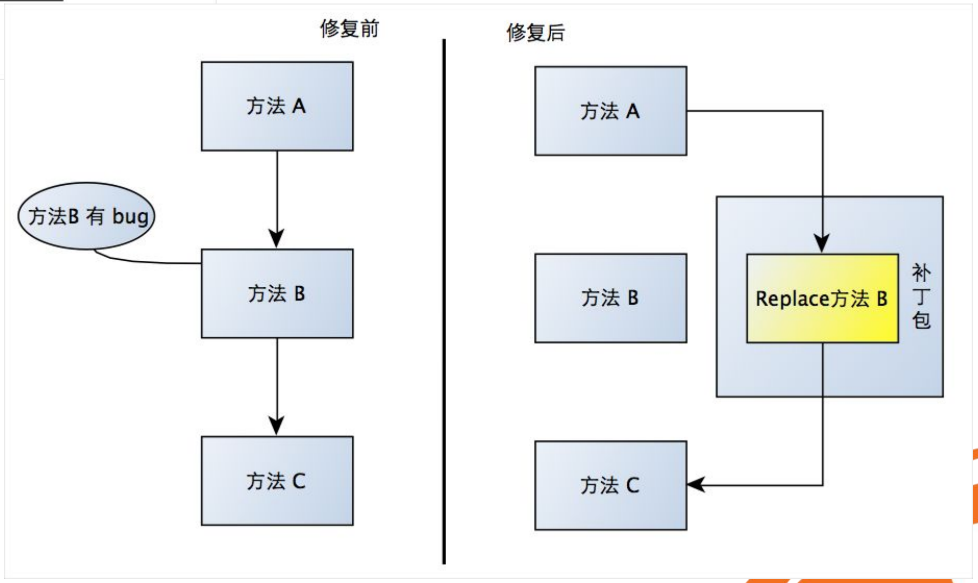
- Hotpatch(方法AOP以及资源替换)
- 线上性能监控
- AOP 编程(查 bug，开发行为监控)

手淘Hotpatch 端侧架构



手淘Hotpatch 端侧修复流程

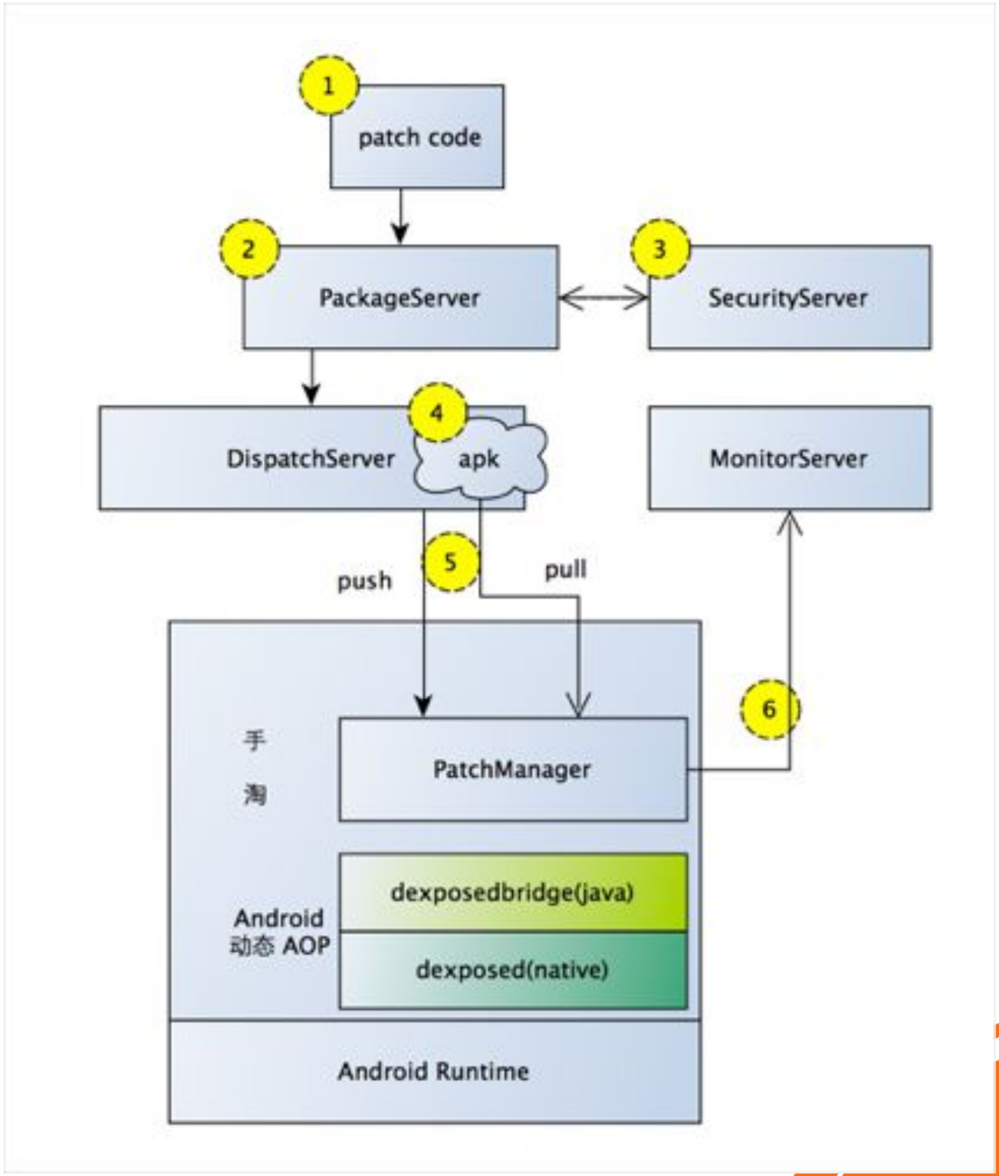




手淘 Hotpatch 后端支撑

- 多维度

  App版本/
  系统版本/
  机型/
  ROM/
  手机制造商/
  渠道/
  地区/
  分仓

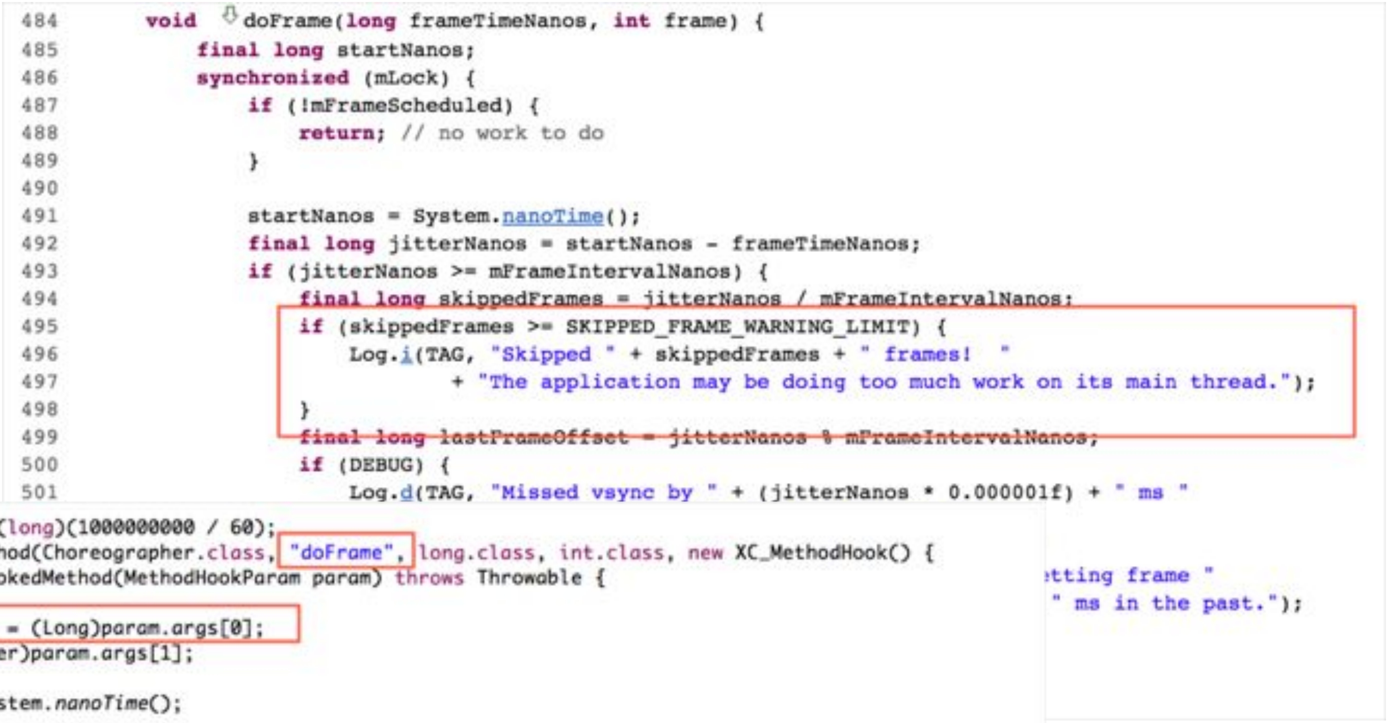


Hotpatch的安全性

- Https
- 多重强校验

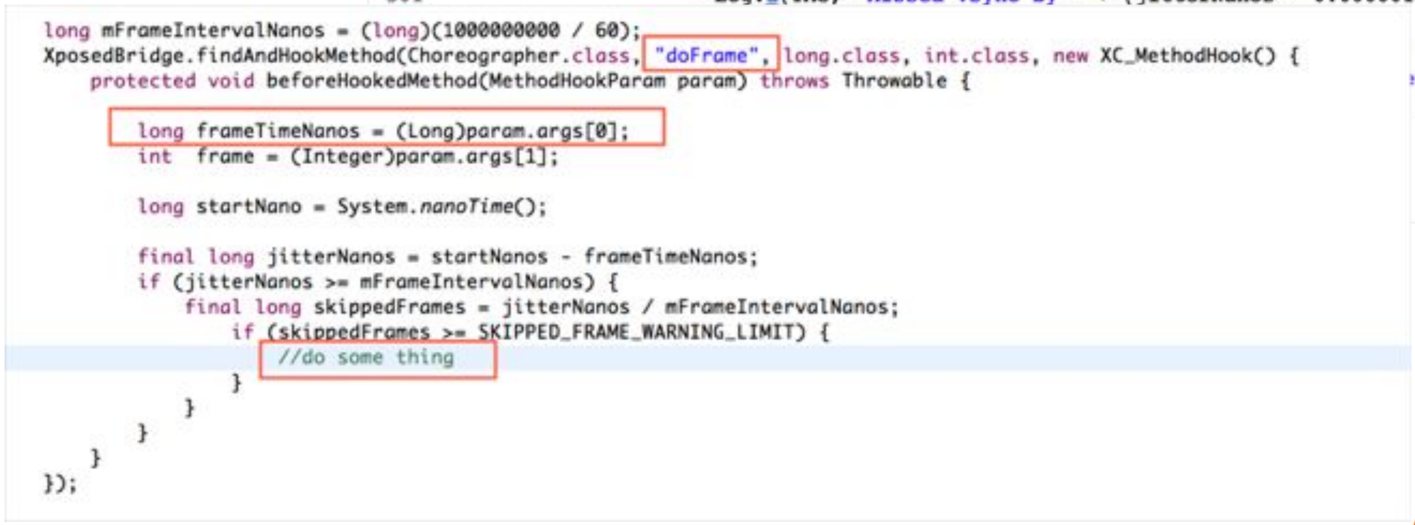
手淘性能监控

- Android 帧率　　android.view.Choreographer.java



```
    void doFrame(long frameTimeNanos, int frame) {
        final long startNanos;
        synchronized (mLock) {
            if (!mFrameScheduled) {
                return; // no work to do
            }

            startNanos = System.nanoTime();
            final long jitterNanos = startNanos - frameTimeNanos;
            if (jitterNanos >= mFrameIntervalNanos) {
                final long skippedFrames = jitterNanos / mFrameIntervalNanos;
                if (skippedFrames >= SKIPPED_FRAME_WARNING_LIMIT) {
                    Log.i(TAG, "Skipped " + skippedFrames + " frames!  "
                            + "The application may be doing too much work on its main thread.");
                }
                final long lastFrameOffset = jitterNanos % mFrameIntervalNanos;
                if (DEBUG) {
                    Log.d(TAG, "Missed vsync by " + (jitterNanos * 0.000001f) + " ms "
```

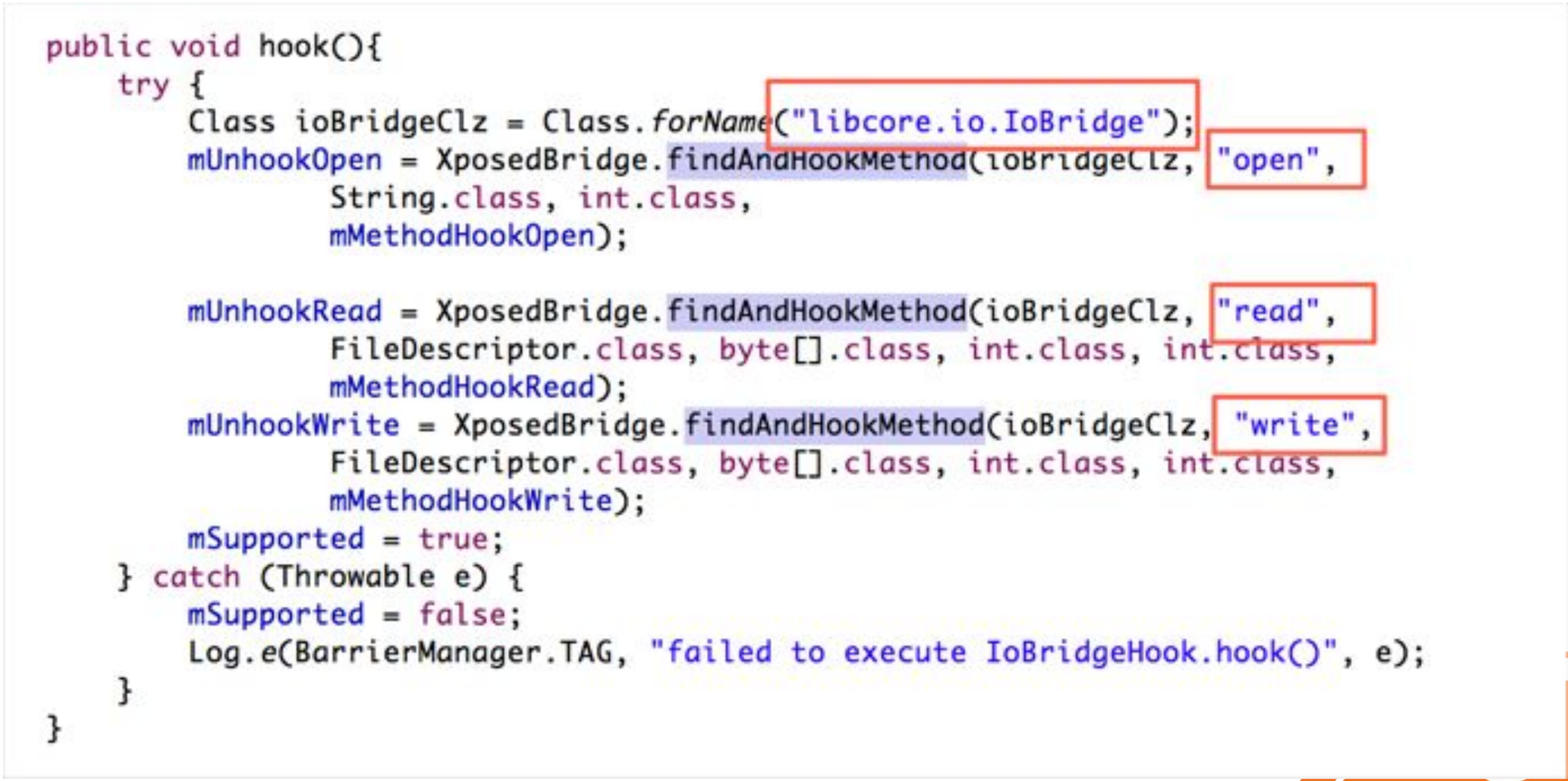
```
tting frame "
" ms in the past.");
```



```
long mFrameIntervalNanos = (long)(1000000000 / 60);
XposedBridge.findAndHookMethod(Choreographer.class, "doFrame", long.class, int.class, new XC_MethodHook() {
    protected void beforeHookedMethod(MethodHookParam param) throws Throwable {

        long frameTimeNanos = (Long)param.args[0];
        int  frame = (Integer)param.args[1];

        long startNano = System.nanoTime();

        final long jitterNanos = startNanos - frameTimeNanos;
        if (jitterNanos >= mFrameIntervalNanos) {
            final long skippedFrames = jitterNanos / mFrameIntervalNanos;
                if (skippedFrames >= SKIPPED_FRAME_WARNING_LIMIT) {
                    //do some thing
                }
            }
        }
    }
});
```

手淘性能监控

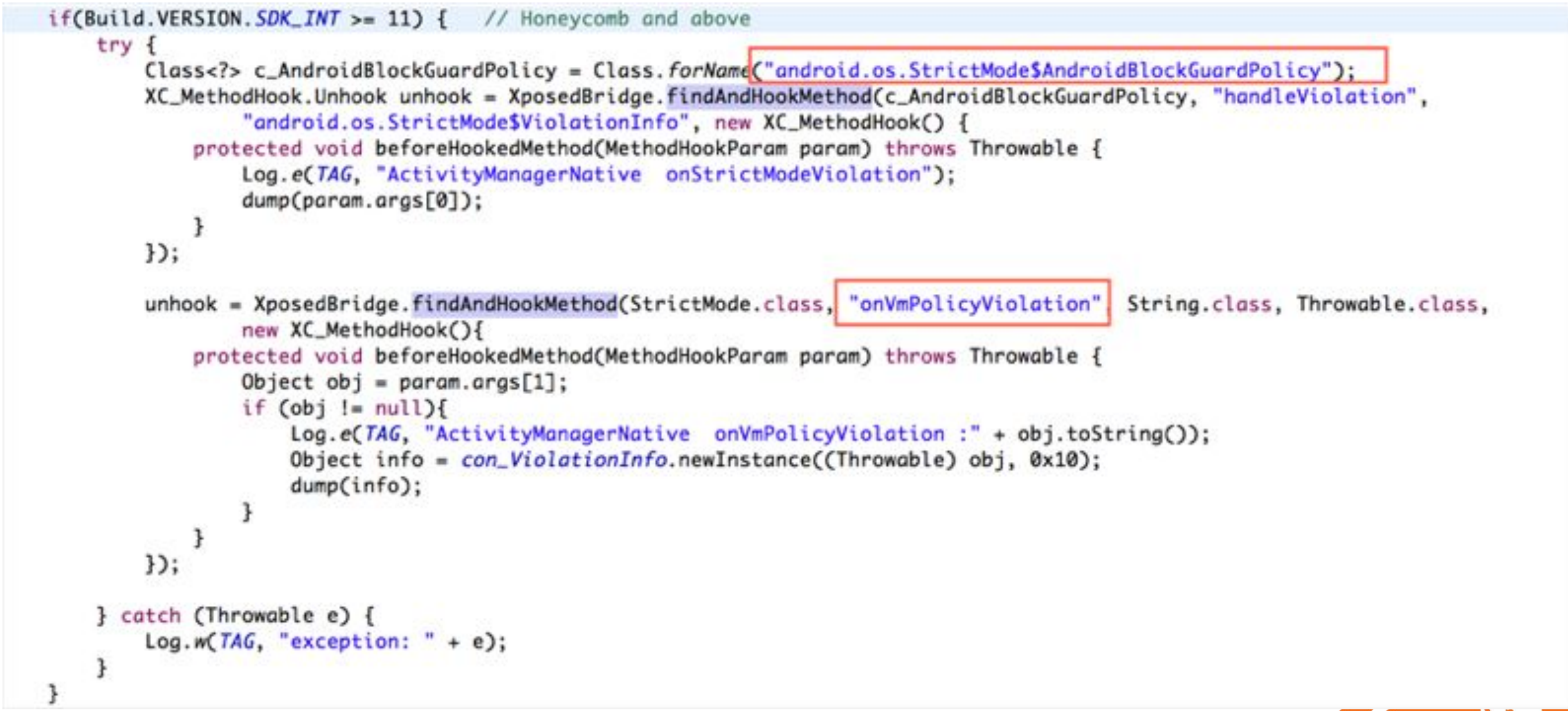
- IO 监测



```
public void hook(){
    try {
        Class ioBridgeClz = Class.forName("libcore.io.IoBridge");
        mUnhookOpen = XposedBridge.findAndHookMethod(ioBridgeClz, "open",
                String.class, int.class,
                mMethodHookOpen);

        mUnhookRead = XposedBridge.findAndHookMethod(ioBridgeClz, "read",
                FileDescriptor.class, byte[].class, int.class, int.class,
                mMethodHookRead);
        mUnhookWrite = XposedBridge.findAndHookMethod(ioBridgeClz, "write",
                FileDescriptor.class, byte[].class, int.class, int.class,
                mMethodHookWrite);
        mSupported = true;
    } catch (Throwable e) {
        mSupported = false;
        Log.e(BarrierManager.TAG, "failed to execute IoBridgeHook.hook()", e);
    }
}
```

手淘性能监控

- StrictMode



```
if(Build.VERSION.SDK_INT >= 11) {   // Honeycomb and above
    try {
        Class<?> c_AndroidBlockGuardPolicy = Class.forName("android.os.StrictMode$AndroidBlockGuardPolicy");
        XC_MethodHook.Unhook unhook = XposedBridge.findAndHookMethod(c_AndroidBlockGuardPolicy, "handleViolation",
                "android.os.StrictMode$ViolationInfo", new XC_MethodHook() {
            protected void beforeHookedMethod(MethodHookParam param) throws Throwable {
                Log.e(TAG, "ActivityManagerNative  onStrictModeViolation");
                dump(param.args[0]);
            }
        });

        unhook = XposedBridge.findAndHookMethod(StrictMode.class, "onVmPolicyViolation", String.class, Throwable.class,
                new XC_MethodHook(){
            protected void beforeHookedMethod(MethodHookParam param) throws Throwable {
                Object obj = param.args[1];
                if (obj != null){
                    Log.e(TAG, "ActivityManagerNative  onVmPolicyViolation :" + obj.toString());
                    Object info = con_ViolationInfo.newInstance((Throwable) obj, 0x10);
                    dump(info);
                }
            }
        });

    } catch (Throwable e) {
        Log.w(TAG, "exception: " + e);
    }
}
```

Dexposed 未来

- Support Art runtime

- NDK AOP

参考资料

- https://github.com/rovo89/Xposed
- https://github.com/alibaba/dexposed

Q/A

谢谢！

手淘技术团队
官方微信账号