QCon 全球软件开发大会
INTERNATIONAL SOFTWARE DEVELOPMENT CONFERENCE

BEIJING 2017

iOS App内存专项实践

一个封闭系统下的大自由

SPEAKER / victorhuang

主办方 Geekbang. 极客邦科技 InfoQ

自我介绍

- 《Android移动性能实践》作者之一
- 勉强跑来说iOS
  - 急事被迫
  - 弥补遗憾

移动专项地图

与自动化、流程结合

CI、自动化测试、自动化分析、自动提单
众测、灰度

指标，工具，方法论

资源类性能：内存，CPU，磁盘，网络，电量
交互类性能：流畅度，响应时延
兼容性 / 稳定性
竞品测试，分层测试，分层研发，性能bug处理流程

底层技术能力

os& os kernel，网络 / 通讯，逆向，注入，hook

封闭系统下的工具

工具

Instruments

XCode

封闭系统下的工具

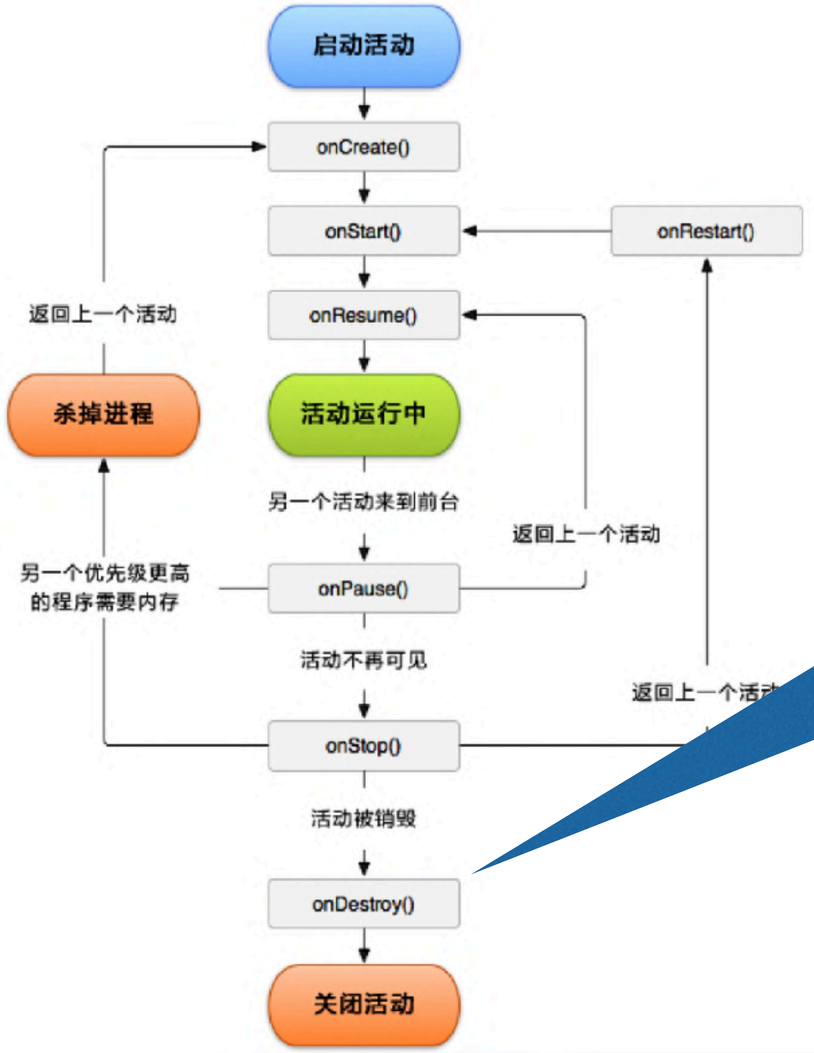
工具	优点	缺点	解决方案
Instruments XCode	• 与IDE编译打通 • 直达代码，高效 • 独立运作	• 开发角色用的工具 • 自动化，CI无法结合	• 团队转型，项目做小 • 封闭系统下找自由

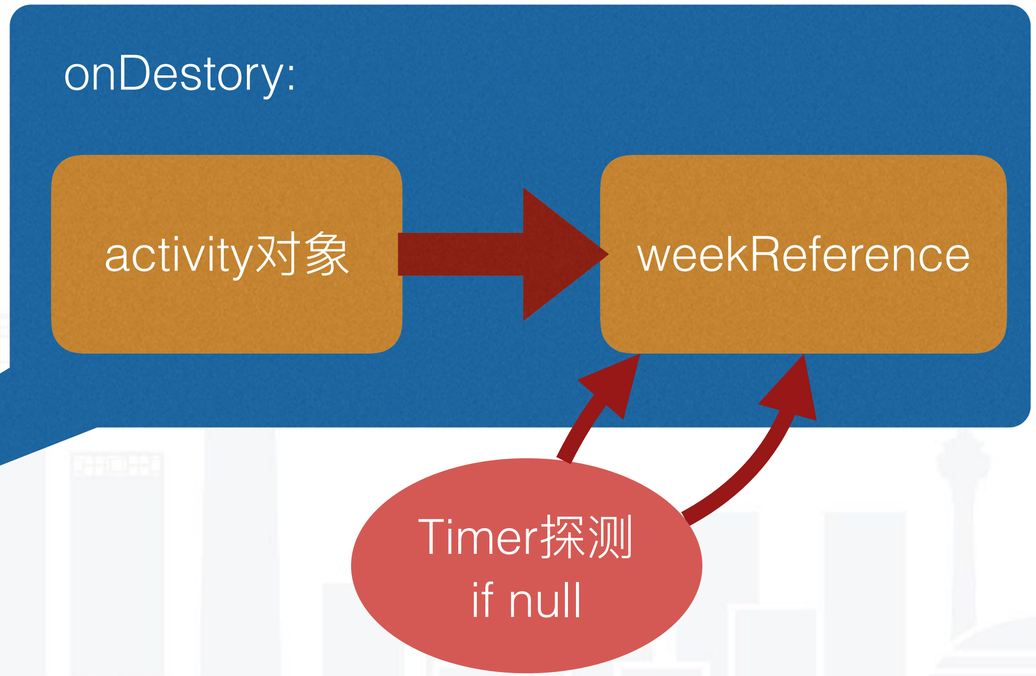
封闭系统下的自由

西部世界

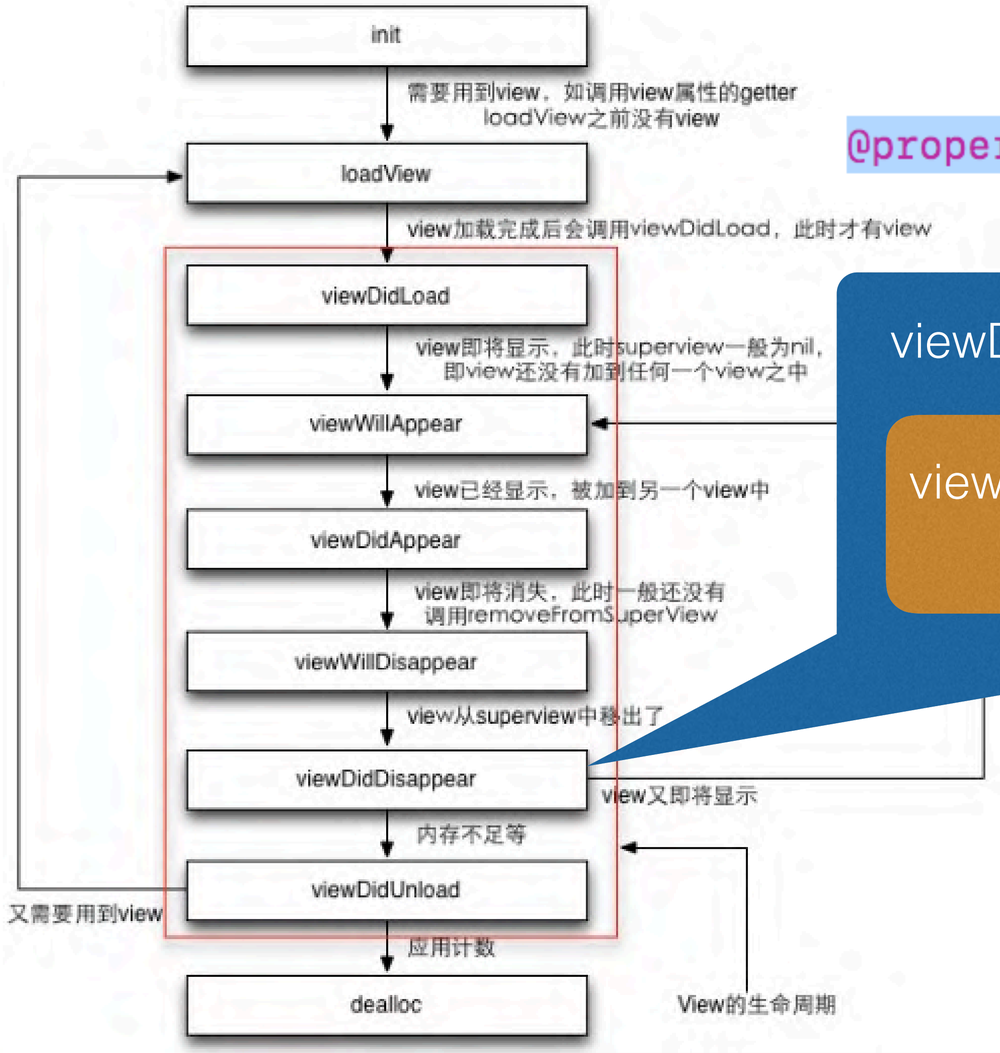
ANDROID的成功能否参照?

LEAKCANARY，内存分析云





界面生命周期：VCLEAKDETECTOR




```
@property (nonatomic,weak) UIViewController* weakVC;
```

viewDidDisappear:

viewController
对象

weekVC
弱引用指针

Timer探测
if nil

案例

[Magnifier分析云][ios_yellow_log_analysis]LEAK OF VIEW CONTROLLER <QQWebViewController>(1 time) ID: 56328633 状态:

详细信息 | SVN提交 (0) | 变更历史 (10) | 更多

这张单是否对您有帮助？点击链接来吐槽反馈 ~:
http://10.185.21.209/cloud_regression/bug_feedback/?product=%E6%89%8B%E6%9C%BAQQ%28iOS%29&bug_type=ios_yellow_log_analysis

Rainbow Log:
VC memory monitor:POSSIBLE LEAK OF VIEW CONTROLLER <<QQWebViewController: 0x109dd0800>>, version:6.7.0.26991_CheckIn_287677_01-20_11:01 , systemVersion:Version 10.2 (Build 14C92) , device_type:iPhone 6 ,feedBack uin:36428816, The Opration sequence is:(

<QQNavigationController: 0x109876400> PopVC: <QQWebViewController: 0x109dd0800>,
<QQNavigationController: 0x109876400> PushVC: <QQWebViewController: 0x109dd0800>,

进出web页面后，出现VC泄漏

<RNLBSTestViewController: 0x11f105550> Dismiss
<ARMapViewController: 0x119cf5060> PresentVC:
<QQNavigationController: 0x109876400> SetVCs:<
<QQNavigationController: 0x109876400> SetVCs:<
<QQNavigationController: 0x109876400> PushVC:
<QQNavigationController: 0x109876400> PushVC:
<ARMapWGSplashViewController: 0x109b30e00> [
<QQRecentController: 0x10985e400> PresentVC: <
)
[**LeakCheck**] Leak addr:0x109dd0800, type:VC name:QQWebViewController, leak num:1 stack:

Alloc stack:
0 QQ +[NSObject(Leak) leakAlloc] (in QQ) (UIViewController+Leak.m:32)
1 QQ -[ARMapCountView topCloudTapAction:] (in QQ) (ARMapCountView.m:523)

泄漏的内存分配点

2 UIKit 0x18eba3000 0x18f17af80
3 UIKit 0x18eba3000 0x18f17e688

案例

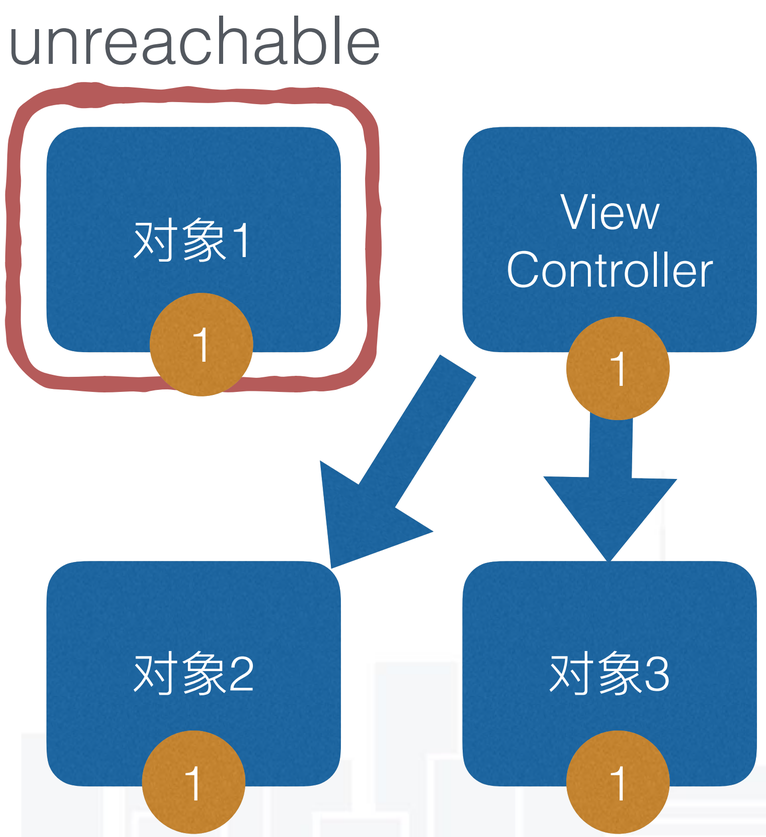
全局强引用，自动+1

```
_webView = [[                                alloc] initWith:realURL forStyle:QQWebViewStyle_TopBar];
_webView.isGamePlaying = NO;
_webView.stackStyle = QQViewControllerStackStyle_Push;
_webView.isPushViewController = YES;
_webView.title = nil;
UINavigationController *nav = (UINavigationController *)[
    appQQTabBarController] selectedViewController];
[nav pushViewControllerAnimated:_webView];
```

缺点

- android 80%的内存Bug都是演变成Activity泄漏
- 可惜iOS并不是？最简单的leak就不行

Android



iOS

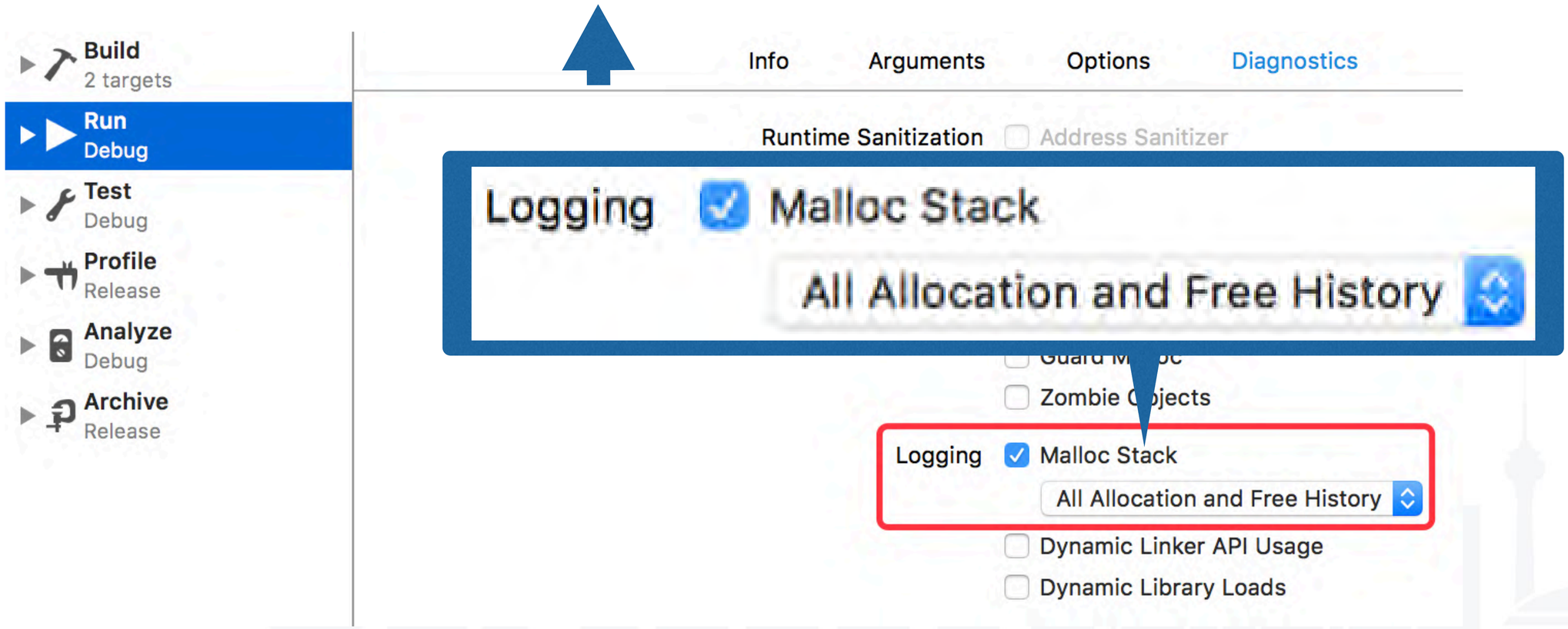
unreachable: 没有指针变量关联的内存地址

获取新建对象申请的内存地址与堆栈

- Hook
  - OC：利用method_exchangeImplementations接管alloc，dealloc
  - 那C++的怎么办?
  - 借用开发工具的权限

开发工具XCODE的能力

追索到环境变量：MallocStackLogging



开发工具XCODE的能力

https://opensource.apple.com/source/Libc/Libc-594.1.4/gen/malloc.c

```
flag = getenv("MallocStackLogging");
if (!flag) {
    flag = getenv("MallocStackLoggingNoCompact");
    stack_logging_dontcompact = 1;
}
// For debugging, the MallocStackLogging or MallocStackLoggingNoComp
// values of "memory", "disk", or "both" to control which stack logg
// in the flag variable, and the strtoul() call below will return 0,
// value of flag.  The default stack logging now is disk stack loggi
if (flag) {
    unsigned long val = strtoul(flag, NULL, 0);
    if (val == 1) val = 0;
    if (val == -1) val = 0;
    if (val) {
        malloc_logger = (void *)val;
        _malloc_printf(ASL_LEVEL_INFO, "recording stacks using recor
    } else if (strcmp(flag,"memory") == 0) {
        malloc_logger = (malloc_logger_t *)stack_logging_log_stack;
        _malloc_printf(ASL_LEVEL_INFO, "recording malloc stacks in m
    } else if (strcmp(flag,"both") == 0) {
        malloc_logger = stack_logging_log_stack_debug;
        _malloc_printf(ASL_LEVEL_INFO, "recording malloc stacks to b
    } else {            // the default is to log to disk
        malloc_logger = __disk_stack_logging_log_stack;
        _malloc_printf(ASL_LEVEL_INFO, "recording malloc stacks to d
```

```
#define stack_logging_type_free             0
#define stack_logging_type_generic          1
#define stack_logging_type_alloc            2
#define stack_logging_type_dealloc          4
#define stack_logging_flag_zone             8
#define stack_logging_flag_calloc           16
#define stack_logging_flag_object           32
#define stack_logging_flag_cleared          64
#define stack_logging_flag_handle           128
#define stack_logging_flag_set_handle_size  256

#define MALLOC_LOG_TYPE_ALLOCATE    stack_logging_type_alloc
#define MALLOC_LOG_TYPE_DEALLOCATE  stack_logging_type_dealloc
#define MALLOC_LOG_TYPE_HAS_ZONE    stack_logging_flag_zone
#define MALLOC_LOG_TYPE_CLEARED     stack_logging_flag_cleared


static void __OFMallocLogger(unsigned type, uintptr_t arg1, uintptr_t arg
    uint32_t num_hot_frames_to_skip)
{
    if(type & stack_logging_flag_zone) {
        type &= ~stack_logging_flag_zone;
        switch (type) {
            case stack_logging_type_alloc:
                malloc_printf("[malloc  zone=%p size=%d ptr=%p]\n", arg
                break;
            case stack_logging_type_alloc | stack_logging_flag_cleared:
                malloc_printf("[calloc  zone=%p size=%d ptr=%p]\n", arg
                break;
            case stack_logging_type_alloc | stack_logging_type_dealloc:
                malloc_printf("[realloc zone=%p oldPtr=%p size=%d ptr=%
                break;
            case stack_logging_type_dealloc:
                malloc_printf("[free    zone=%p ptr=%p]\n", arg1, arg2)
                break;
            default:
                malloc_printf("[???? type=%d arg1=%p arg2=%p arg3=%p re
                    result);
                break;
```

获取新建对象申请的内存地址与堆栈

- Hook
  - OC：利用method_exchangeImplementations接管alloc，dealloc
  - C++ 内网版本：借用开发工具的权限，malloc_logger简单直接
  - C++ 外网版本：fishhook hook：malloc, calloc, free

获取指针变量

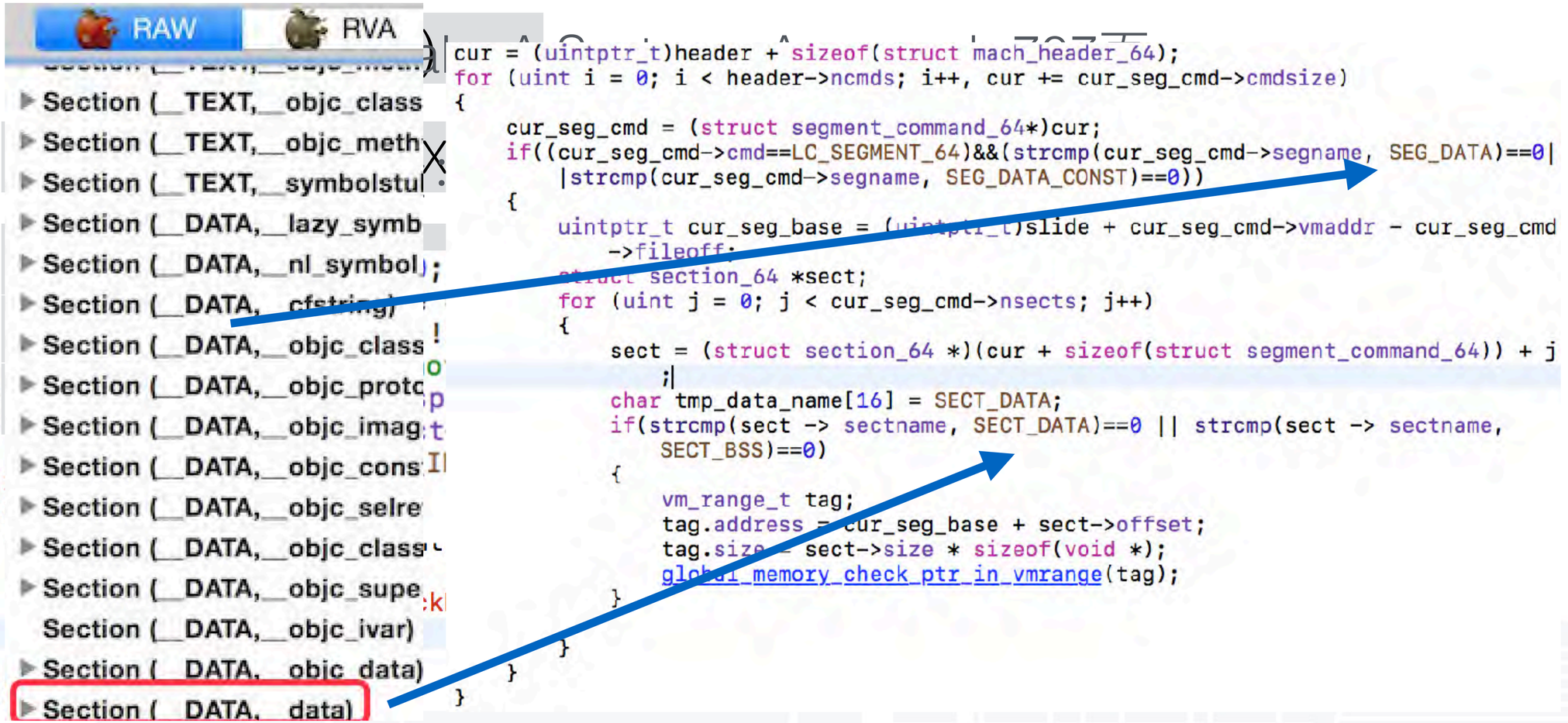
获取新建对象申请
内存地址与堆栈

获取指针变量

堆区

全局区

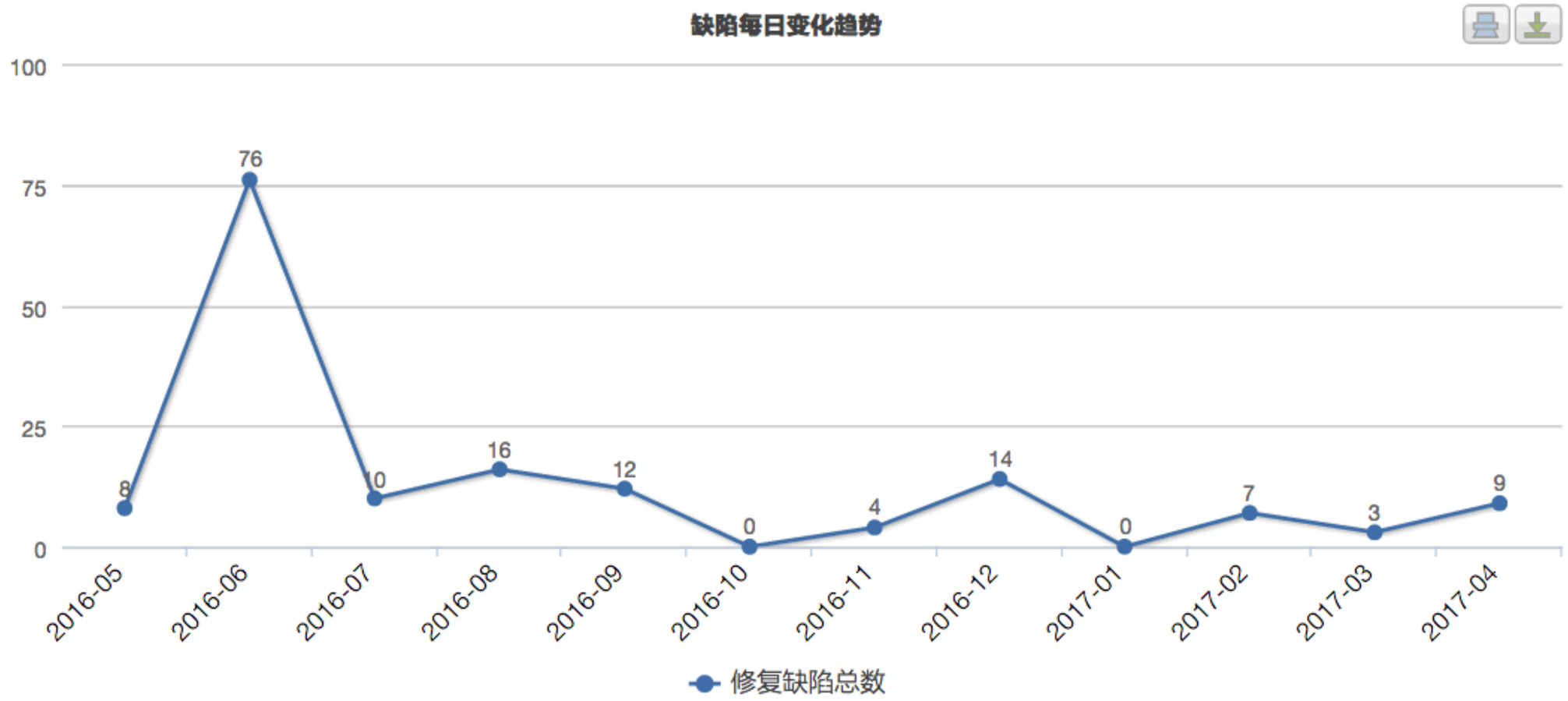
栈区




HashMap
变量1 地址1
变量2 地址2
变量3 地址3
遍历
找没有指针变量
指向的内存地址
Leaks
HashMap
地址1 堆栈1
地址2 堆栈2
地址3 堆栈3
地址4 堆栈4

效果

形成工具：**QQLeak**， 全自动发现并解决159个泄漏BUG



问题

- 内存泄漏的后果是什么？ 内存耗尽，闪退
- 同样造成这个后果的只有Leaks么？

JETSAM EVENT

大内存常驻和侵略性的内存使用也会导致闪退

When Jetsam needs to reclaim memory, it kills lower priority processes first

It also kills any process that goes over its limit, or *high water mark*

https://vimeo.com/140377195

解决思路

- 记录申请的内存大小和堆栈
- 记录其释放
- 按照堆栈聚合数据

判断：

1. 单个申请的内存块大于50MB
2. 申请内存(未释放)的总和大于阈值

- 上报数据
- 手动分析提单

iphone机型	触顶阀值
iphone4、iphone4s	200Mb
iphone5、iphone5s、iphone6、iphone6s	300Mb
iphone se、iphone6s、iphone6sp、iphone7、iphone7p	800Mb

案例

SNGAPM

添加产品　手机QQ(iC

- 指标
- 统计
- 个例
- 配置
- BUG汇总
- 文档
- 实验室

17:14:50					
2017-04-14 17:12:49		iPhone7,2	10.3		chunk_malloc
2017-04-14 16:50:13		iPhone9,2	10.3.1		chunk_malloc
2017-04-14 16:46:44		iPhone 6Plus	8.1.1		normal
2017-04-14 16:45:08		iPhone9,2	10.3.1		chunk_malloc
2017-04-14 16:45:08		iPhone9,2	10.3.1		chunk_malloc
2017-04-14 16:42:21		iPhone9,2	10.3.1		chunk_malloc

案例

【OOMDetector】QQGifDecode爆内存　ID：56690069　状态：已关闭

详细信息 | SVN提交(0) | 变更历史(3) | 更多

此堆栈累计申请296MB的内存

Total memory occupied by the target stack:296.610000mb;

Total object that allocate by the target stack and still unfreed in memory:1240

user infos:

uin:[redacted] occur_time:2017-03-03 13:01:53

uin:[redacted] occur_time:2017-03-02 22:07:51

uin:[redacted] occur_time:2017-03-02 11:59:43

uin:[redacted] occur_time:2017-03-02 19:01:06

stack:

"0 libsystem_kernel.dylib 0x197ed8000 0x197ed9740"

"1 CoreGraphics 0x187294000 0x18729eb88"

"2 CoreGraphics 0x187294000 0x18729e9c0"

还有一种内存问题

- 内存泄漏
- 内存常驻
- 内存越界

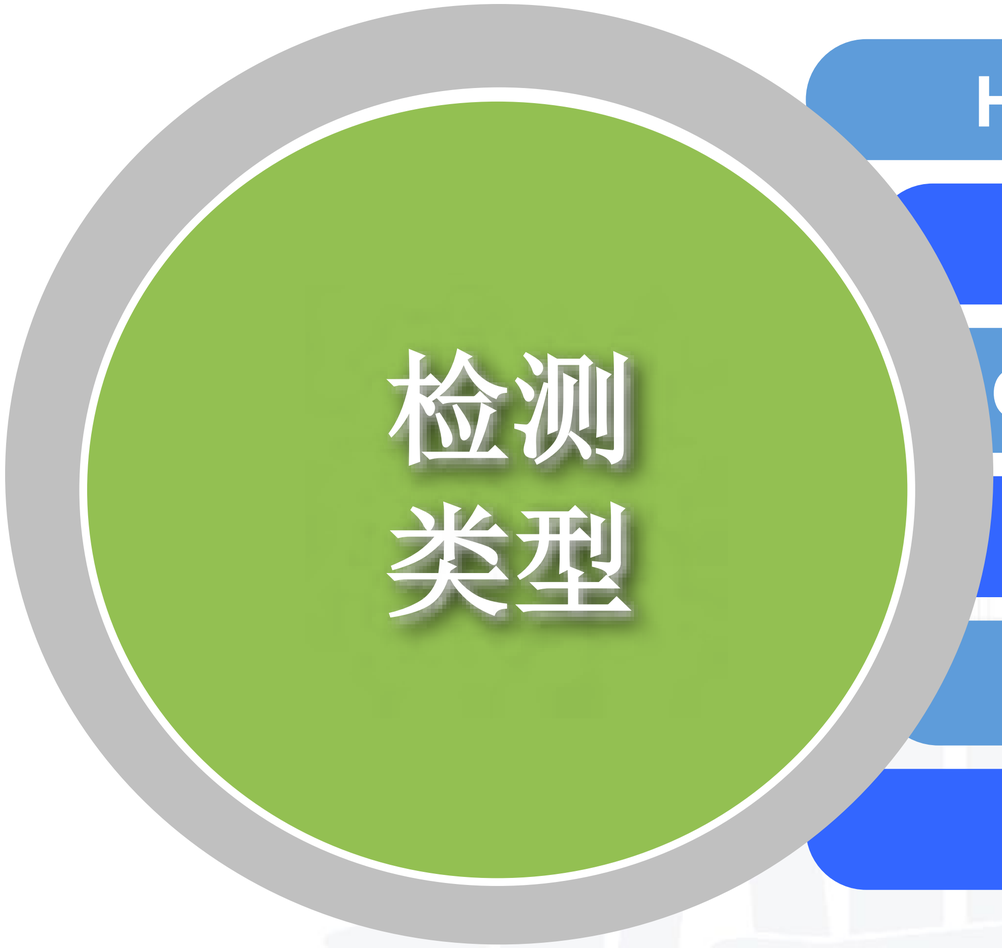
QQLeak

OOMDetector

VCLeakDetector

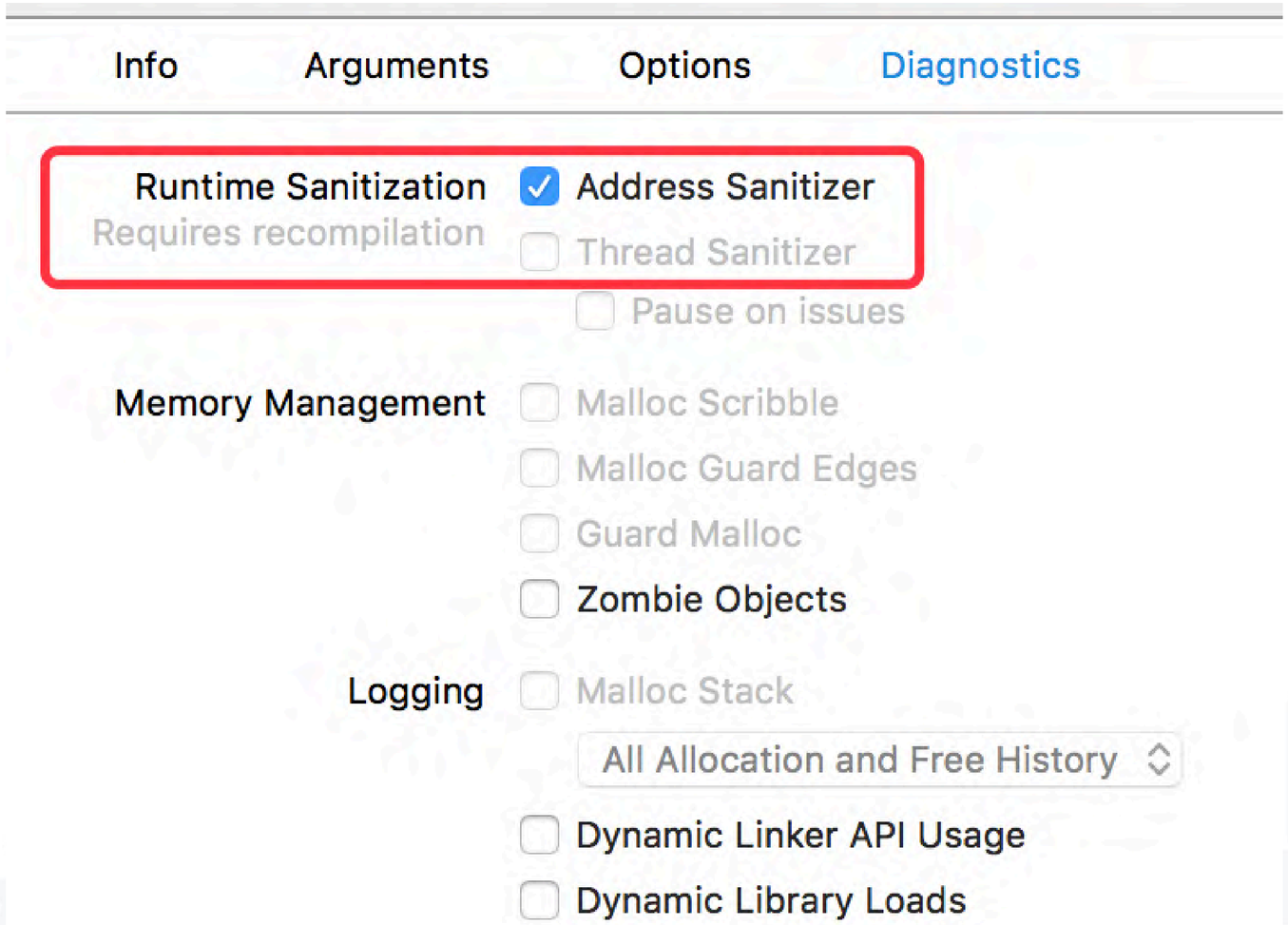
?

ADDRESS SANITIZER



- Heap buffer overflow
- Stack buffer overflow
- Global buffer overflow
- dangling pointer
- Use-after-free
- Double-free

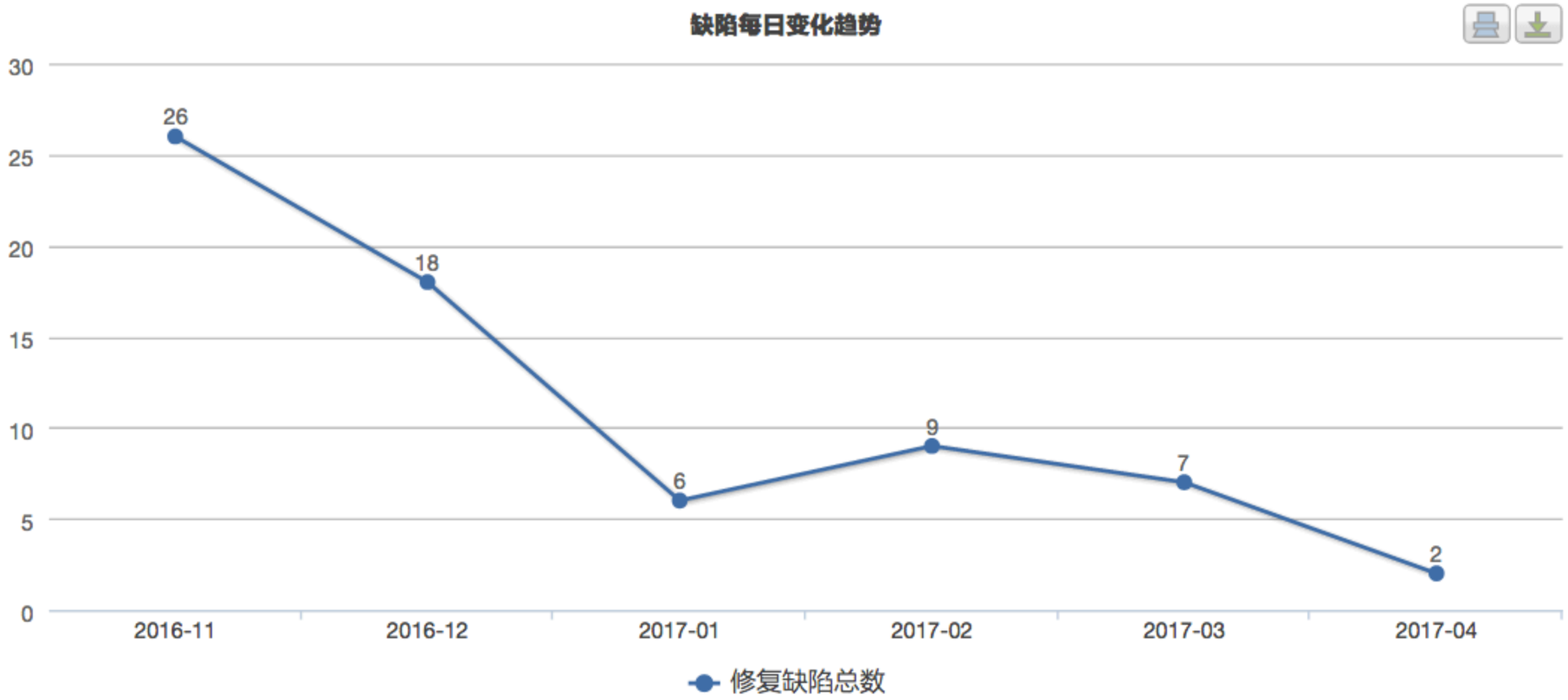
ADDRESS SANITIZER



无法利用NEWMONKEY做自动化

- 3.4倍内存占用导致Jetsam：
  - 用7s，内存大呀， 20分钟后CRASH
  - 用模拟器跑，改造monkey适配模拟器
- 越界马上crash
  - 增加参数-continue-after-error
  - monkey记录跑过的路径

效果

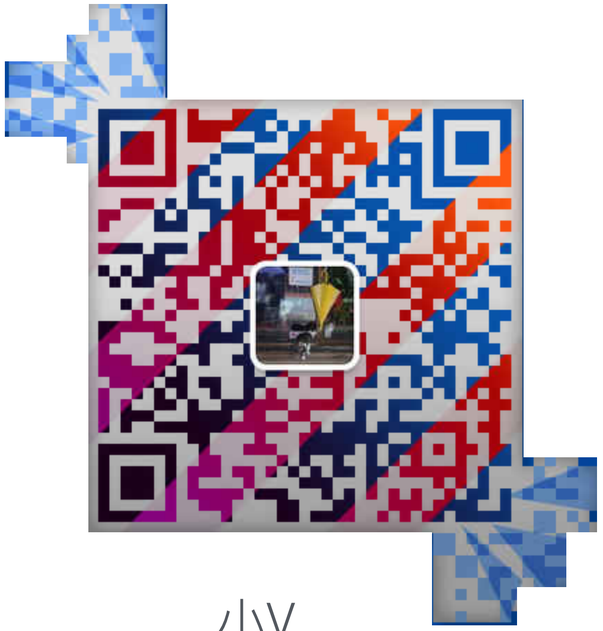


- 减少40%的无堆栈CRASH

封闭系统下的自由

西部世界

- **使用好工具：**ASAN，XCode，Instruments
- **利用系统常理：**界面生命周期
- **借用开发工具权限：**
  xcode(malloc_logger)，LLDB
- **利用hook接管系统函数：**
  fishhook,method_exchangeImplementations
- **利用如hook功能的源码能力：**
  fishhook如何遍历SECT_DATA，获取全局区



小V
天天爱晴天

《Android移动性能实战》
4月20日前预售

Qcon2017北京站博文视点
摊位首发

Thanks!

主办方 Geekbang. 极客邦科技 InfoQ

拓展阅读

- Flipboard/FLEX　https://github.com/Flipboard/FLEX
- FaceBook ios内存profile工具介绍：https://code.facebook.com/posts/583946315094347/automatic-memory-leak-detection-on-ios/
- jetsam介绍：https://vimeo.com/140377195，http://newosxbook.com/articles/MemoryPressure.html
- malloc源码：https://opensource.apple.com/source/Libc/Libc-166/gen.subproj/malloc.c