



ALIBABA SECURITY AGENCY

OPTIMIZED FUZZING IOKIT IN IOS

LEI LONG

WHO AM I?

- LEI LONG
- Security Expert in Mobile Security of Alibaba Group
- Focus on Security Research of iOS
- Twitter: @cererdlong

Outlines

- Introduction
- Information Extraction
- Fuzzing
- Results

Part I

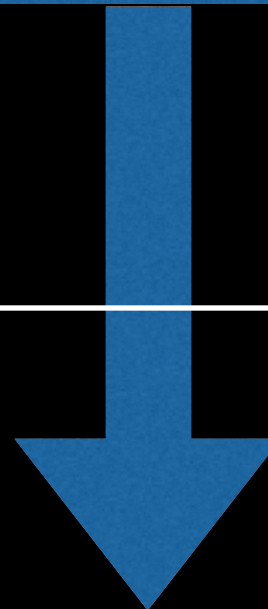
Introduction



IOKit



Userspace



Kernel





Previous Research

- Focused on IOExternalMethodDispatch
- Base on IDA static analysis
- Disadvantages
 - sMethod symbols required
 - Decrypted kernelcache required
 - Unresolved instructions of some KEXTs
 - Insufficient information



Our Work

- Not only IOExternalMethodDispatch
- Base on dynamically kernel read/write
- Advantages
 - Independent of sMethod symbols
 - A decrypted kernelcache not required
 - More sufficient information



Prerequisite

- Jailbroken device
- tfp0 kernel patch

Part II

Information Extraction



Information Extraction

- Basic Information
- IOUserClients' Access Info
- IOExternalMethodDispatch
- IOExternalMethod



Basic Information



Basic Information

- All OSObject subclasses
- Information Types
 - Class name
 - Vtable start address
 - Virtual method address and its vtable offset
 - Symbol of overwritten virtual method
 - Instance size
 - Inheritance relationships



Basic Information

Class:AppleARMPMUPowerSource

BundleID:com.apple.driver.AppleARMPlatform

vtableaddr:0x804c71a8

instance size:0x184

SuperClassNames:

IOPMPowerSource

IOService

IORegistryEntry

OSObject

Method Virtuals:

[vtable,0x0] overwrite at 0x804b4804

[vtable,0x4] overwrite __ZN15IOPMPowerSourceD0Ev at 0x804b4808

[vtable,0x1c] overwrite __ZNK15IOPMPowerSource12getMetaClassEv at 0x804b481c

[vtable,0x50] overwrite __ZN9IOService4initEP12OSDictionary at 0x804b4858

[vtable,0xac] overwrite __ZN15IORegistryEntry13setPropertiesEP8OSObject at 0x804b5c08

[vtable,0x168] overwrite __ZN9IOService5startEPS_ at 0x804b4888

[vtable,0x1d0] overwrite __ZNK9IOService11getWorkLoopEv at 0x804b5458

[vtable,0x1e8] overwrite __ZN9IOService20callPlatformFunctionEPK8OSSymbolbPvS3_S3_S3_ at 0x804b5518

[vtable,0x300] overwrite __ZN9IOService13setPowerStateEmPS_ at 0x804b54e8

[vtable,0x344] at 0x804b5454

[vtable,0x348] at 0x804b6070



Motivation Of Basic Information Extraction

```
.AppleS5L8930XUSBPhy: __text:80C8AF18
.DCD 0x440510C4, 0x4608680A, 0x47906C52, 0x28014428, 0x6820D11F
.AppleS5L8930XUSBPhy: __text:80C8AF18
.DCD 0xF06F2108, 0xF8D00210, 0x4620336C, 0x6820E016, 0xF06F2108
.AppleS5L8930XUSBPhy: __text:80C8AF18
.DCD 0xF8D00208, 0x4620336C, 0x6820E00E, 0xF06F2108, 0xF8D00204
.AppleS5L8930XUSBPhy: __text:80C8AF18
.DCD 0x4620336C, 0x6820E006, 0xF06F2108, 0xF8D00220, 0x4620336C
.AppleS5L8930XUSBPhy: __text:80C8AF18
.DCD 0x26004798, 0x22006820, 0x10ACF8D4, 0x3350F8D0, 0x47984620
.AppleS5L8930XUSBPhy: __text:80C8AF18
.DCD 0xF0006DA0, 0x4630F9EF, 0x8B04F85D, 0xBF00BDF0, 0xE00002C2
.AppleS5L8930XUSBPhy: __text:80C8AF18
.DCD 0x9C3, 0x9F9, 0x979, 0x98D, 0xAF03B5F0, 0x8D04F84D
.AppleS5L8930XUSBPhy: __text:80C8AF18
.DCD 0x48244604, 0x46154698, 0x460E4478, 0x68006800, 0xD12A42B0
.AppleS5L8930XUSBPhy: __text:80C8AF18
.DCD 0x68F84920, 0x68094479
.AppleS5L8930XUSBPhy: __text:80C8B718 dword_80C8B718
.DCD 0xF0006809, 0x4601FA0B, 0xB1E94818, 0x6E826808, 0x47904608
.AppleS5L8930XUSBPhy: __text:80C8B718
.DCD 0x68404916, 0xD1184288, 0xF0002018, 0x4D17FA4D, 0x447D4604
.AppleS5L8930XUSBPhy: __text:80C8B718
.DCD 0xF0004629, 0x4815FA1F, 0x30084478, 0x46286020, 0xFA48F000
.AppleS5L8930XUSBPhy: __text:80C8B718
.DCD 0x2C00480D, 0x6938BF1E, 0x20006004, 0x8B04F85D, 0x480EBDF0
.AppleS5L8930XUSBPhy: __text:80C8B718
.DCD 0x462A4631, 0x44784643, 0xF8D06800, 0x4620C1F0, 0x8B04F85D
.AppleS5L8930XUSBPhy: __text:80C8B718
.DCD 0x40F0E8BD, 0xBF004760, 0xE00002C2, 0x444D4F4E, 0xE00002BD
.AppleS5L8930XUSBPhy: __text:80C8B718
.DCD 0x96C, 0x950, 0xE92, 0xD24, 0x8FA, 0x4D08B5B0, 0xAF024604
.AppleS5L8930XUSBPhy: __text:80C8B718
.DCD 0x4629447D, 0xF9E8F000, 0x44784805, 0x60203008, 0xF0004628
.AppleS5L8930XUSBPhy: __text:80C8B718
.DCD 0x4620FA11, 0xBF00BDB0, 0xE24, 0xCB6, 0xAF03B5F0, 0x8D04F84E
.AppleS5L8930XUSBPhy: __text:80C8B718
.DCD 0x4606B081, 0x46152030, 0xF0004688, 0x4604F9F7, 0xF9BCF000
.AppleS5L8930XUSBPhy: __text:80C8B718
.DCD 0x6820B184, 0x4A0A4643, 0x402A6E31, 0xF4426B85, 0x90003080
.AppleS5L8930XUSBPhy: __text:80C8B718
.DCD 0x46324620, 0xB92047A8, 0x69416820, 0x47884620, 0x46202400
.AppleS5L8930XUSBPhy: __text:80C8B718
.DCD 0xF85DB001, 0xBDF08B04, 0x60070, 0x4A07B580, 0x49072318
.AppleS5L8930XUSBPhy: __text:80C8B718
.DCD 0x447A466F, 0x68124479, 0xF970F000, 0x44794904, 0x60013108
.AppleS5L8930XUSBPhy: __text:80C8B718
.DCD 0xBF00BD80, 0x836, 0x7B0, 0xC96, 0xB96AF000, 0x466FB580
.AppleS5L8930XUSBPhy: __text:80C8B718
.DCD 0xF996F000, 0x44794902, 0x60013108, 0xBF00BD80, 0xC12
.AppleS5L8930XUSBPhy: __text:80C8B718
.DCD 0x466FB580, 0xF98AF000, 0x44794902, 0x60013108, 0xBF00BD80
.AppleS5L8930XUSBPhy: __text:80C8B718
.DCD 0xBFA, 0xB988F000, 0xB986F000, 0x466FB580, 0xF982F000
.AppleS5L8930XUSBPhy: __text:80C8B718
.DCD 0xE8BD2118, 0xF0004080, 0xBF00B995, 0x44784801, 0xBF004770
.AppleS5L8930XUSBPhy: __text:80C8B718
.DCD 0xD32, 0x4A07B580, 0x49072318, 0x447A466F, 0x68124479
.AppleS5L8930XUSBPhy: __text:80C8B718
.DCD 0xF92CF000, 0x44794904, 0x60013108, 0xBF00BD80, 0x7AE
.AppleS5L8930XUSBPhy: __text:80C8B718
.DCD 0x728, 0xC0E, 0x2018B5B0, 0xF000AF02, 0x4D07F97B, 0x447D4604
.AppleS5L8930XUSBPhy: __text:80C8B718
.DCD 0xF0004629, 0x4805F94D, 0x30084478, 0x46286020, 0xF976F000
.AppleS5L8930XUSBPhy: __text:80C8B718
.DCD 0xBDB04620, 0xCEE, 0xB80, 0x4D08B5B0, 0xAF024604, 0x4629447E
.AppleS5L8930XUSBPhy: __text:80C8B718
.DCD 0xF938F000, 0x44784805, 0x60203008, 0xF0004628, 0x4620F961
.AppleS5L8930XUSBPhy: __text:80C8B718
.DCD 0xBF00BDB0, 0xCC4, 0xB56, 0x4606B5F0, 0x460D480A, 0x4478AF03
.AppleS5L8930XUSBPhy: __text:80C8B718
.DCD 0x6C046800, 0x47A04630, 0xBF1C2801, 0xBDF02000, 0x46284905
.AppleS5L8930XUSBPhy: __text:80C8B718
.DCD 0x70004479, 0x6170F000, 0xBF102000, 0xBF002001, 0x722
```




Pick Out Vtable

- Locating kernel mach-o's `__DATA,__const`
 - Kernel
 - Kernel extensions
- Vtable filter
 - Vtable layout
 - Vtable characteristic



Vtable Layout

0(≥ 4 Byte)
virtual method 0
virtual method 1
virtual method 2
virtual method 3
....
virtual method N-3
virtual method N-2
virtual method N-1

- Thumb virtual method addresses
 $x \in (\text{TEXT_StartAddrss}, \text{TEXT_EndAddress})$
or $x \in (\text{PERLINKTEXT_StartAddrss}, \text{PERLINKTEXT_EndAddress})$
- $N \in [14, \infty)$
- Starting after at least four all-0 bytes



Vtable Characteristic

virtual method 0
....
...
...
virtual method 7:getMetaClass
...
...
...
virtual method N-1

→ key to get runtime information



getMetaClass Definition

- OSDefineMetaClassAndStructors

```
Define OSDefineMetaClassAndStructors \  
....  
    const OSMetaClass * className ::getMetaClass() const \  
        { return &gMetaClass; } \  
....
```

- gMetaClass is the key to get runtime information.



gMetaClass address

- **KERNEL**

```
addr_a1:MOV R0,#imm1  
addr_a2:ADD R0,PC  
addr_a3:BX LR
```



```
gMetaClass2=addr_b2 + imm1 + 4
```

- **KEXT**

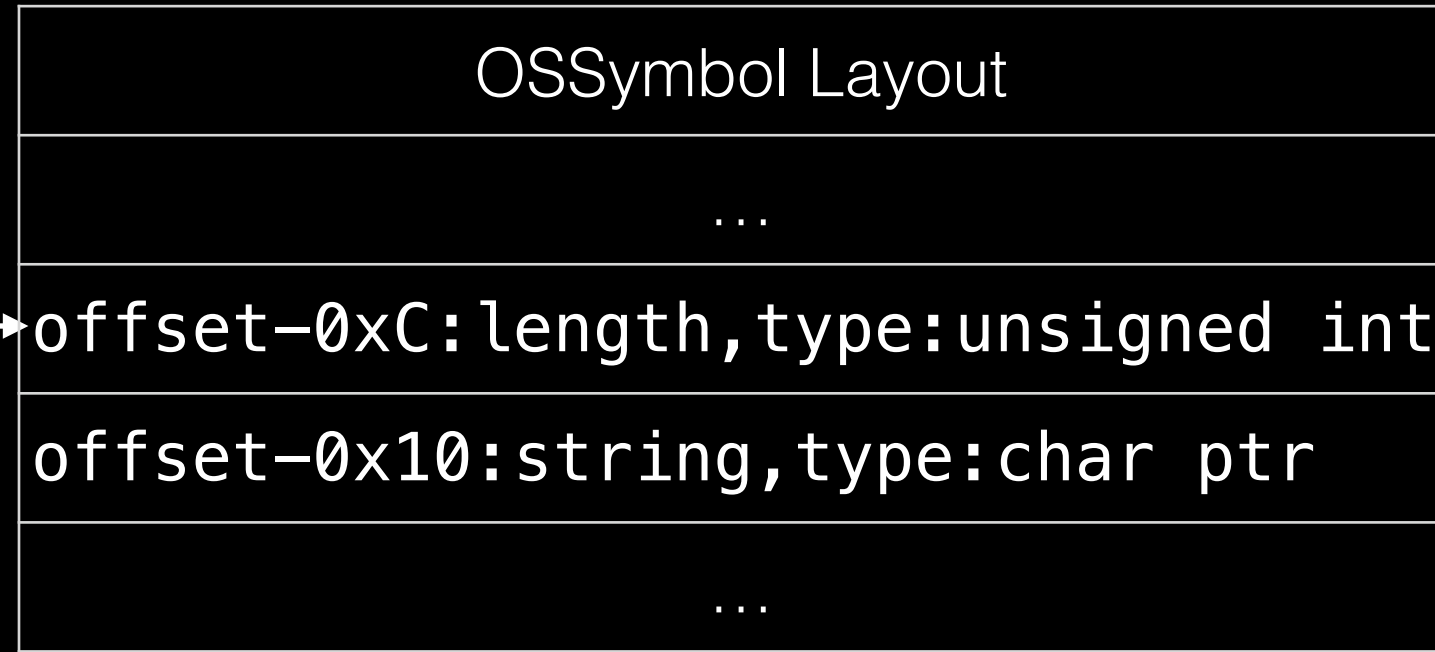
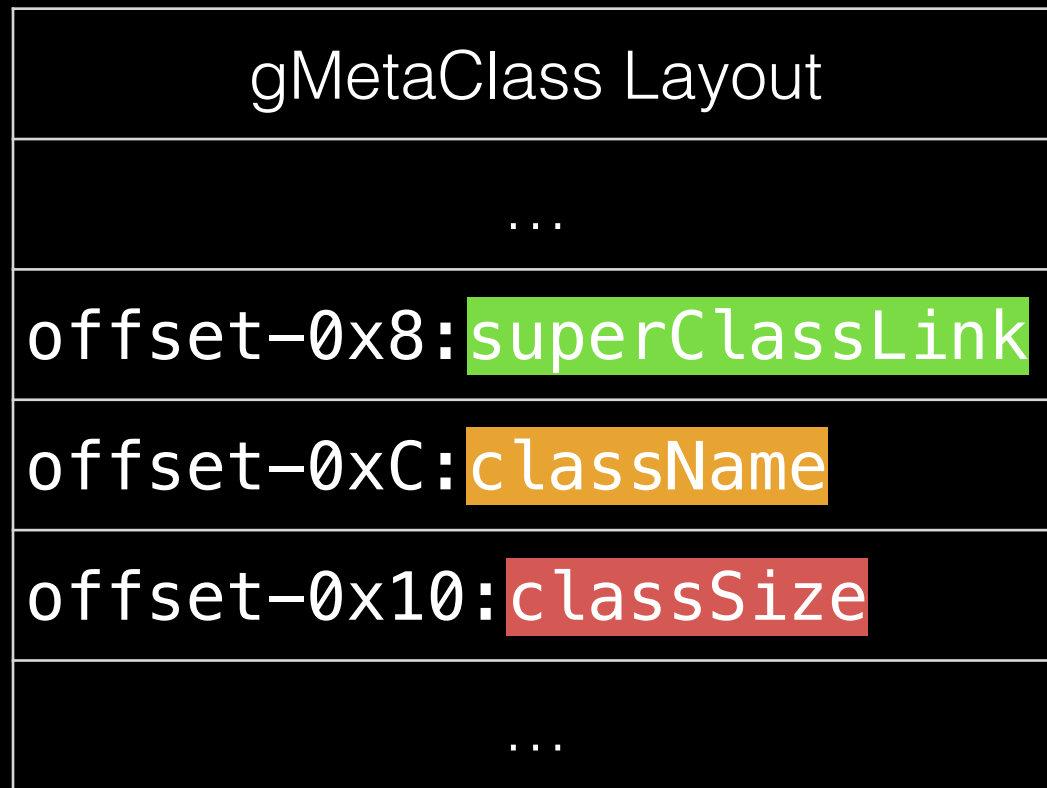
```
addr_b1:LDR R0 =#imm2  
addr_b2:ADD R0,PC  
addr_b3:BX LR
```



```
gMetaClass2=addr_b2 + KernelRead4Byte(addr_b1+(4-(addr_b1+imm2)%4)) + 4
```



gMetaClass Layout



- classSize - unsigned int
- superClassLink - OSMetaClass ptr
 - Backwards to OSObject
 - All inheritance relationships



Functionality Provided by KEXT

Example

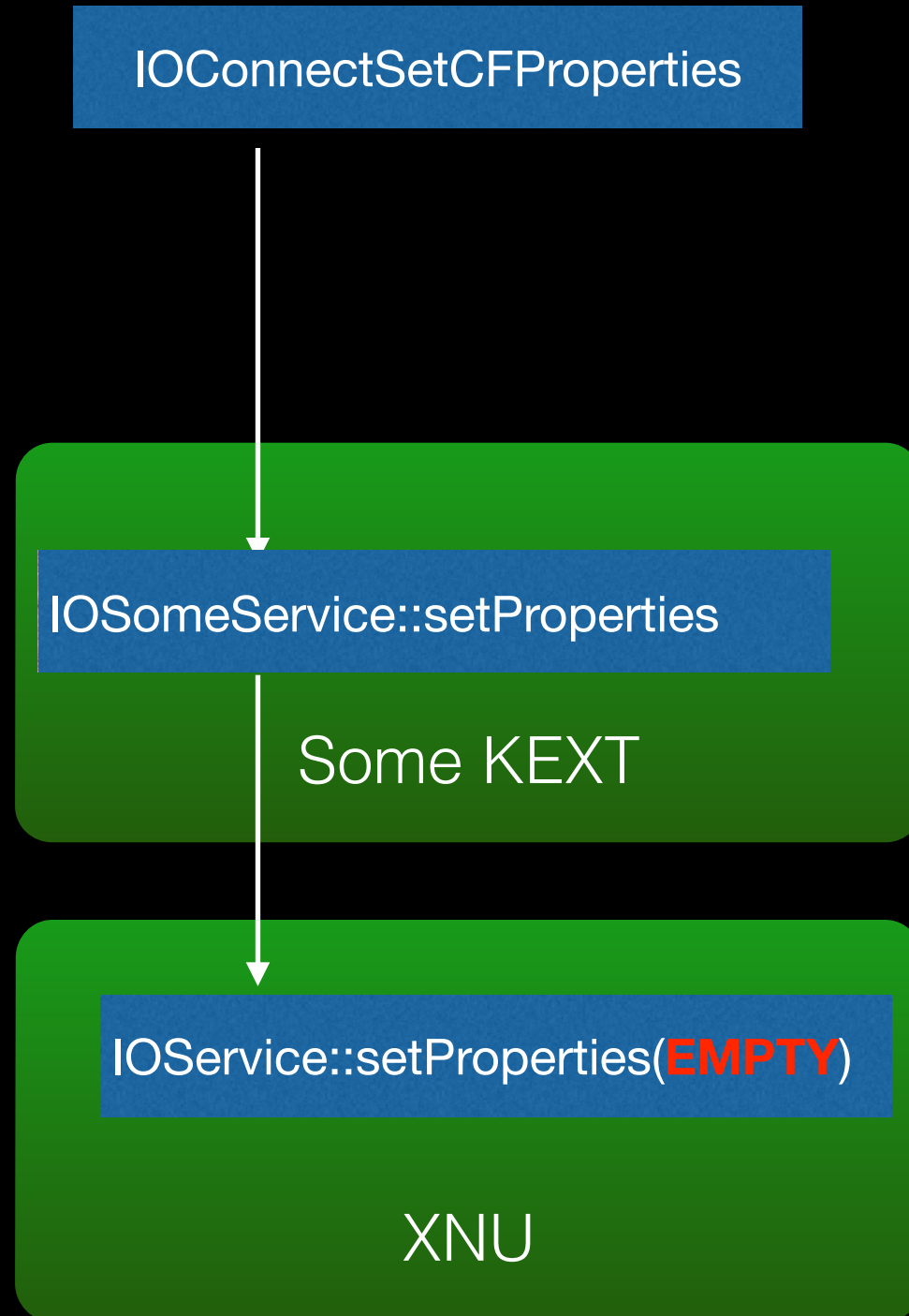
IOConnectSetCFProperties

IOService::setProperty

Some KEXT

IOService::setProperty(**EMPTY**)

XNU





virtual methods' "overwritten"

virtual method 0
virtual method 1
virtual method 2
virtual method 3
....
virtual method N-3
virtual method N-2
virtual method N-1
virtual method N

```
if( address∈(KERNEL_TEXT_StartAddress,  
            KERNEL_TEXT_EndAddress) {  
    //implementation in XNU without overwriting  
    .....  
} else {  
    //overwritten implementation  
    .....  
}
```



Overwritten virtual methods symbolization

- Assumption
 - The same names and sequences in the same iOS version in different devices
- Obtaining names and sequences from kernelcaches with leaked decrypting-keys

[http://theiphonewiki.com/wiki/Firmware_Keys:](http://theiphonewiki.com/wiki/Firmware_Keys)

kernelcache.release.n94

IV: ae291ecd536ab102e6975a730f065f2f

Key: c45aac2036dea7bf564bd99399e6ff35b241b580afd323a7aee1b6e9162b1d4f

TextBlock 10

- deducing the symbolization in those encrypted kernelcaches without keys



Obtaining names and sequences from decrypted kernelcache

- Command “nm kernelcache”
- Vtable information export
- Name-Address pair matching



IOUserClients' Access Info



Example

Client: IOPKEAcceleratorUserClient

Service: AppleSamsungPKE:0

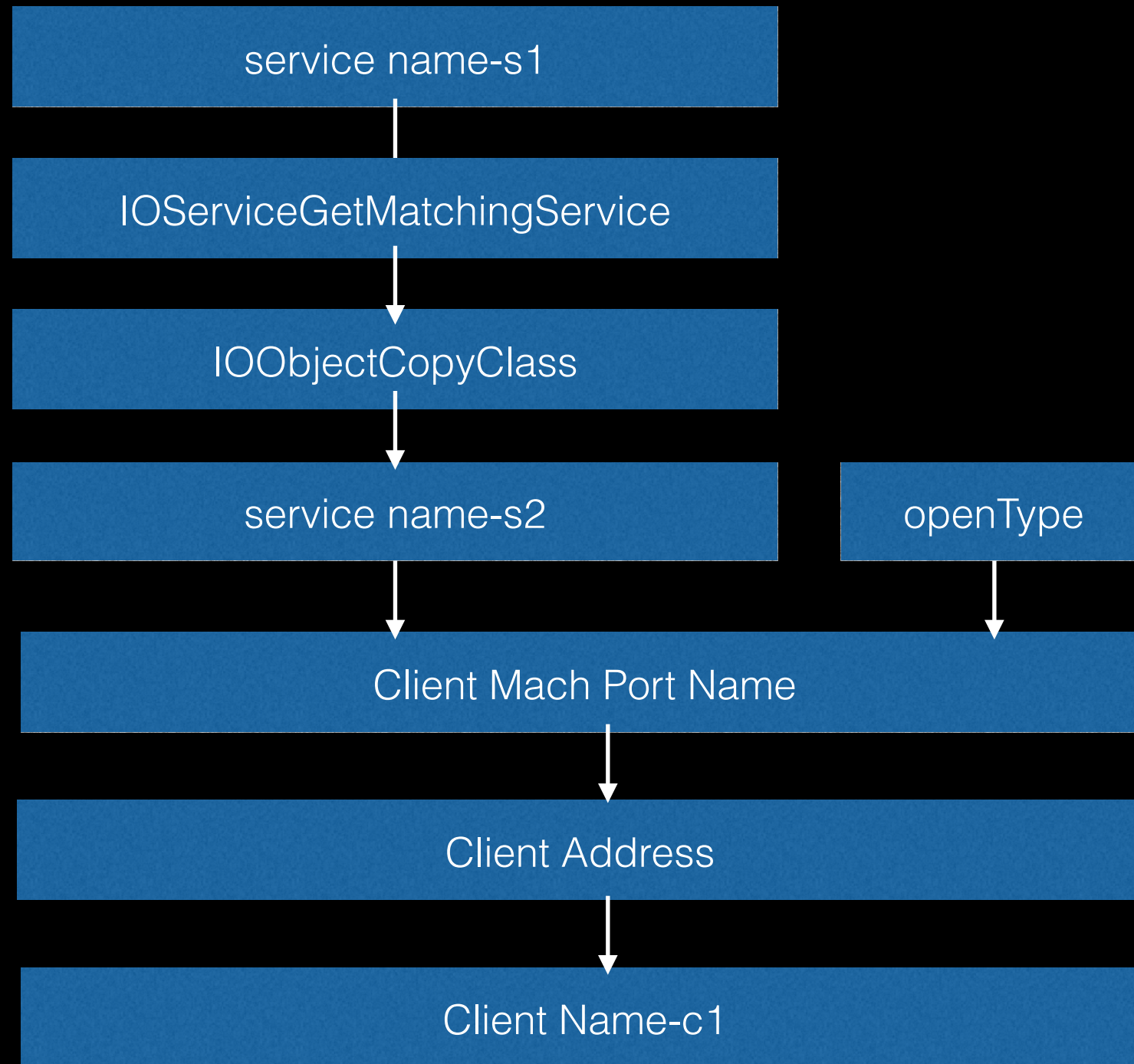
vtableaddr: 0x807341f8

canOpen: 1

instance size: 0x80



Access Info Export



(c1, s2, openType)



Detail Steps

- s1: obtain all subclasses of IOService
 - `OSKextCopyLoadedKextInfo`
 - Basic Information Extraction
- openType: try all openTypes
 - 0x00~0xff
 - magicCodes: locating newUserClient
- c1: retrieve Client Name
 - `mach_port_kobject`
 - `mach_port_space_info`



ipc_port ptr via mach_port_space_info

```
vm_address_t cr_mach_port_kobject(vm_address_t portname) {
    ipc_info_space_t info;
    ipc_info_name_array_t table = 0;
    mach_msg_type_number_t tableCount = 0;
    ipc_info_tree_name_array_t tree = 0;
    mach_msg_type_number_t treeCount = 0;
    vm_address_t obaddress = 0;
    mach_port_space_info(mach_task_self(), &info, &table,
&tableCount, &tree, &treeCount);
    for( int index = 0 ; index < tableCount ; index++ ) {
        ipc_info_name_t info = table[index];
        if(portname == info.iin_name) {
            obaddress = info.iin_object;
        }
    }
    obaddress -= vm_kernel_addrperm;
    //obaddress is the address of structure ipc_port. By adding
offset
//0x44,we can get ipc_kobject_t kobject in 32-bit devices.
    return CRReadAtAddress(obaddress+0x44);
}
```

- ipc_info_name_t->iin_name == client port name
- ipc_info_name_t->iin_object = obfuscated ipc_port ptr
- ipc_port ptr = obfuscated - vm_kernel_addrperm



vm_kernel_addrperm

- By locating a kernel function with
 - VM_KERNEL_ADDRPERM
 - A unique characteristics string
- Luckily, IOGeneralMemoryDescriptor::wireVirtual

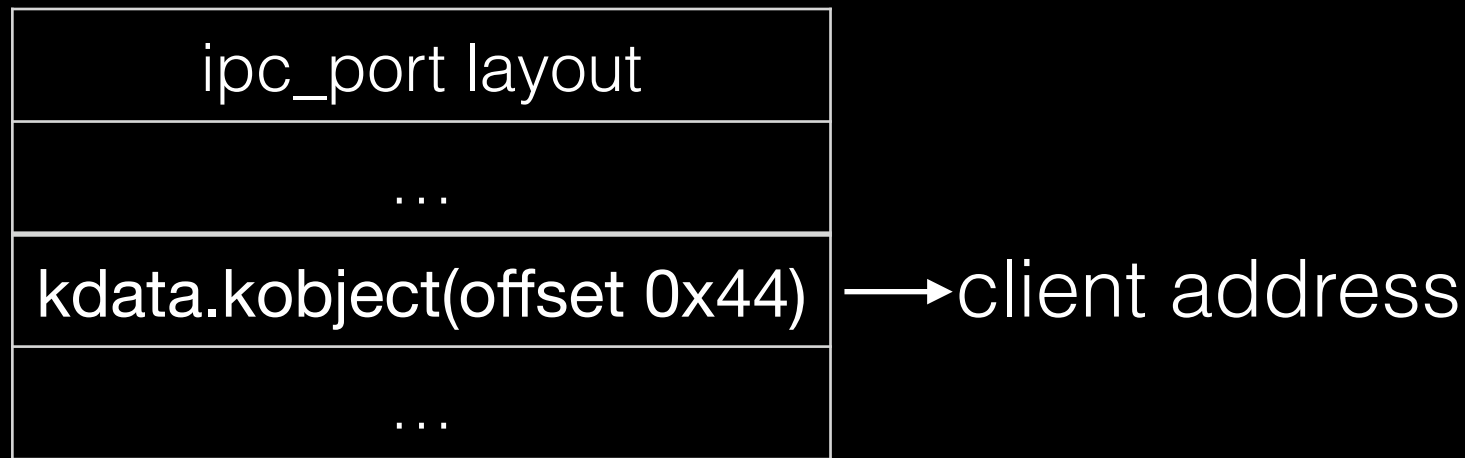
```
IOReturn IOGeneralMemoryDescriptor::wireVirtual(IODirection forDirection)
{
.....
        OSReportWithBacktrace("IOMemoryDescriptor 0x%lx prepared read
only", VM_KERNEL_ADDRPERM(this));
.....
}
```

- ipc_port ptr = obfuscated - KernelRead(vm_kernel_addrperm's address)



Retrieve Client Name

- Client address in struct ipc_port



- Retrieve Client Name Via Client Address
 - Get vtable address
 - Locate getMetaClass()
 - Get gMetaClass ptr
 - Get client name



IOExternalMethodDispatch



IOExternalMethodDispatch

IOUserClient::externalMethod

- Be overwritten to provide IO services
- Use IOExternalMethodDispatch for input/output check
 - type
 - length
- 0xe00002c2 error if check failed



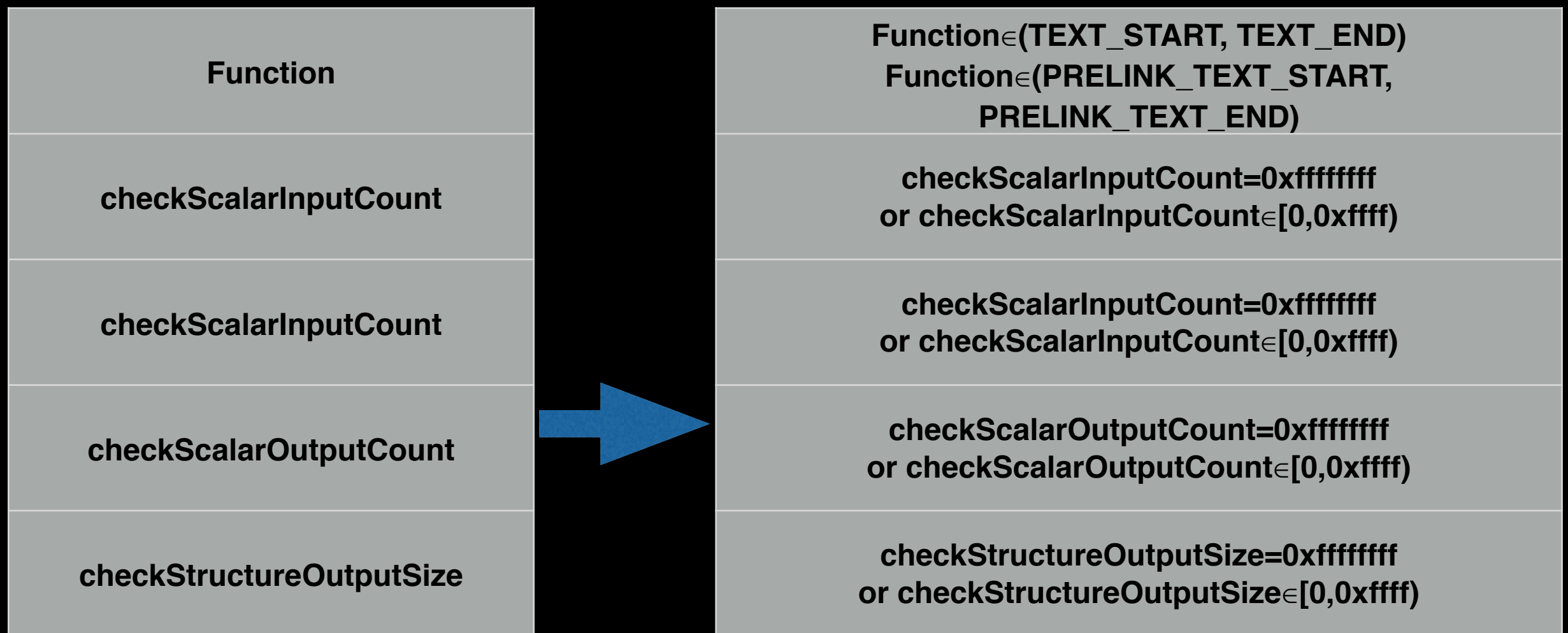
Extracting IOExternalMethodDispatch

1. Narrow and determine the searching scope.
2. Match IOExternalMethodDispatch Table characteristics.
3. Locate IOExternalMethodDispatch Table address.
4. Dump all.



IOExternalMethodDispatch Table characteristics

- IOExternalMethodDispatch fields



- Table length ≥ 2



IOExternalMethodDispatch Table Dump

0
Client virtual method 0
1~N
Client virtual method N
....
Client metaClass virtual method 0
1~N
Client metaClass virtual method N
....
....
function-0
checkScalarInputCount-0
checkStructureInputSize-0
checkScalarOutputCount-0
checkStructureOutputSize-0
1~N
function-N
checkScalarInputCount-N
checkScalarInputCount-N
checkScalarOutputCount-N
checkStructureOutputSize-N

→ client vtable start address

→ continous all-0 bytes

→ client meta class vtable start address

→ Position-x

→ continous all-0 bytes

→ Variables block start address (Rang-x)

→ IOExternalMethodDispatch Table block (Start)

Locating Position-x address

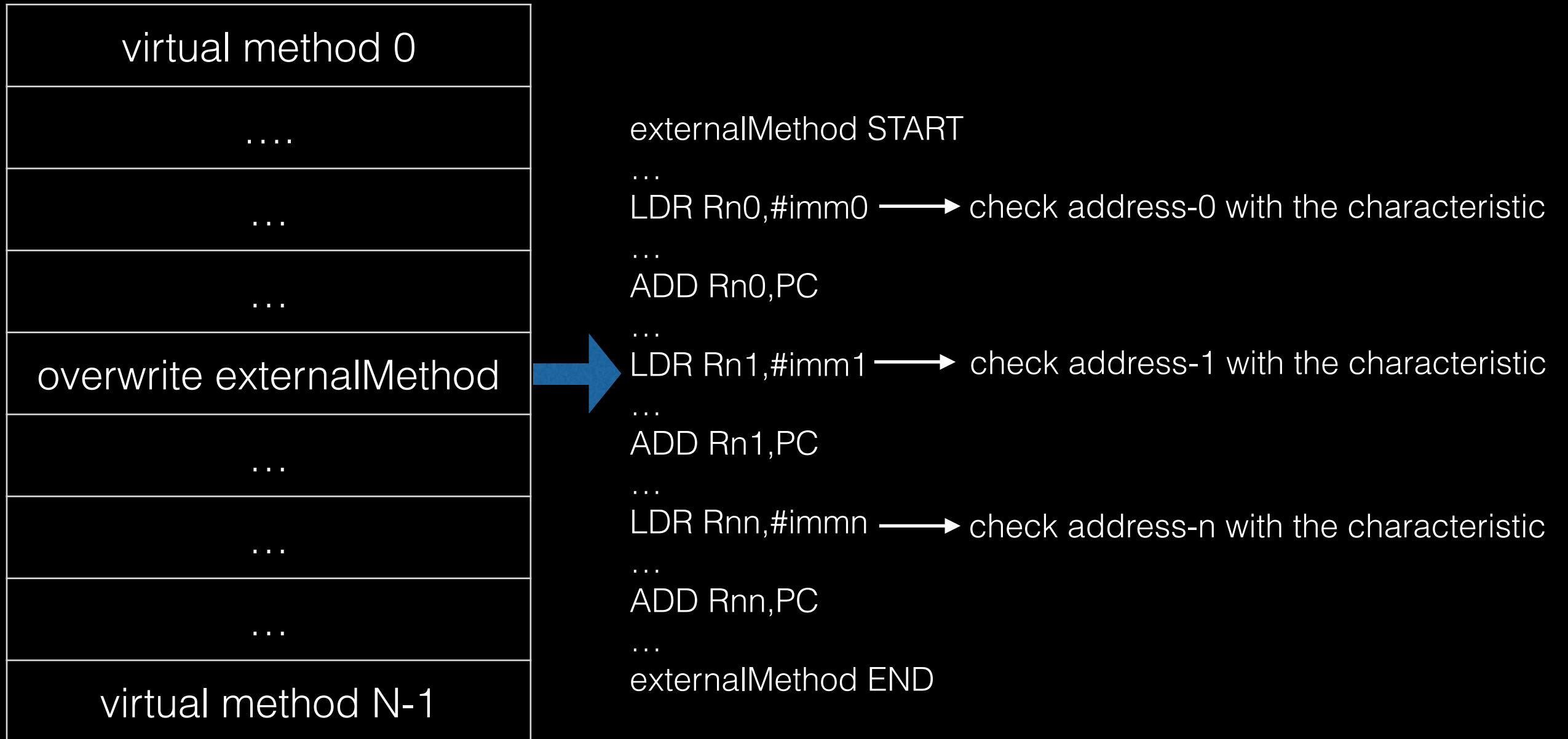
Searching from Position-x by bytes (in Rang-x)

at least 2 continuous blocks matching the characteristics (Start)

IOExternalMethodDispatch Table starting address



Complemental Mechanism





IOExternalMethod



IOExternalMethod

IOUserClient::getTargetAndMethodForIndex

- be overwritten to provide IO services
- use IOExternalMethod for input/output check
 - type
 - length
- 0xe00002c2 error if check failed



Extract IOExternalMethod

- IOExternalMethod Export

by directly invoking `getTargetAndMethodForIndex`

- Arbitrary kernel code execution

Stefan Esser, "Tales from iOS 6 Exploitation and iOS 7", HITB 2013



Carrier

- Mach Msg OOL Data
- locating OOL Data address

`mach_port_space_info->`

`struct ipc_mqueue->`

`struct ipc_kmsg_queue messages->`

`struct ipc_kmsg *ikmq_base->`

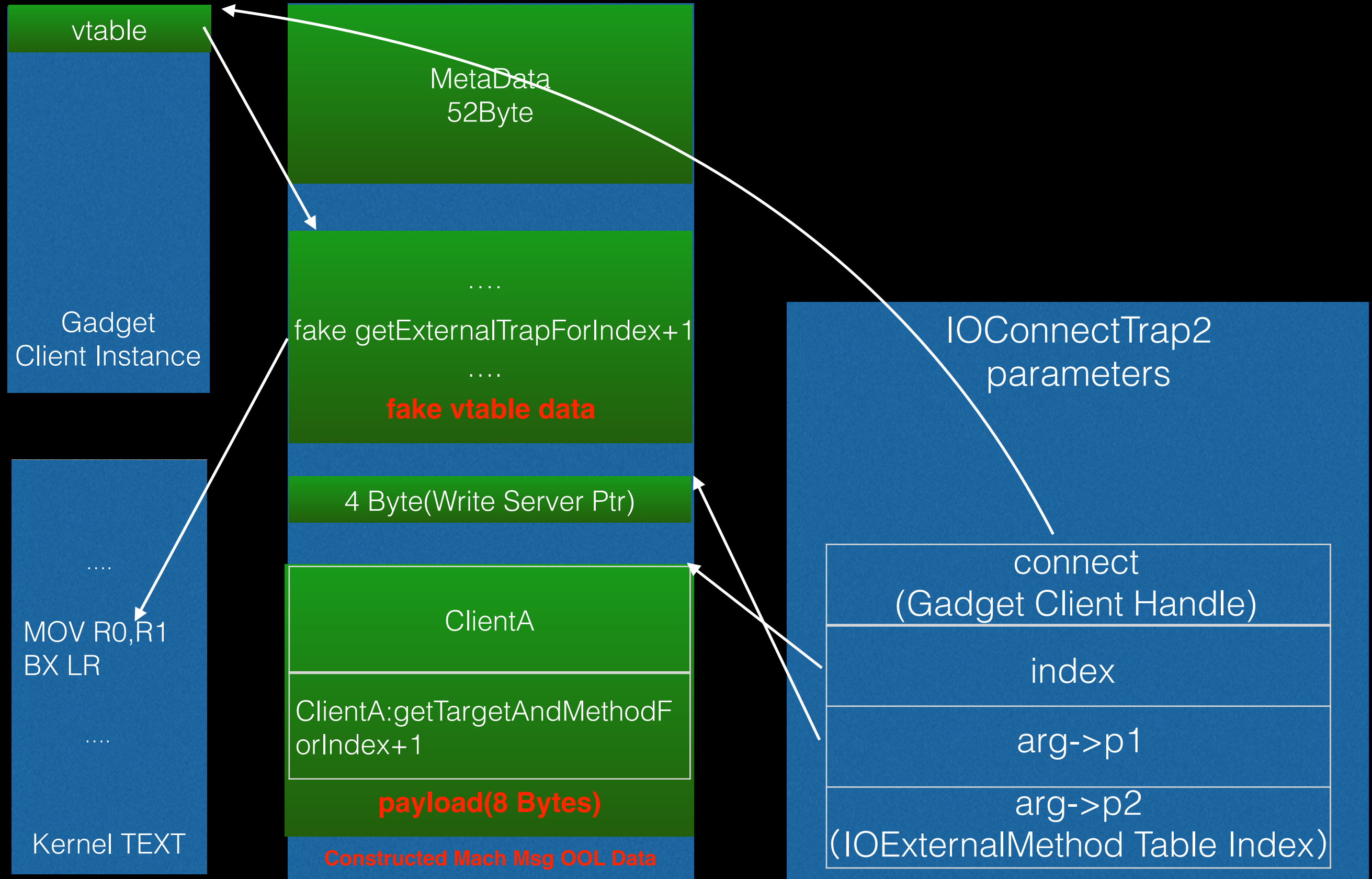
`mach_msg_header_t *ikm_header->`

`msgh_remote_port (ool address)->`

`msgh_remote_port + 52`



Information Extraction(II): IOExternalMethod



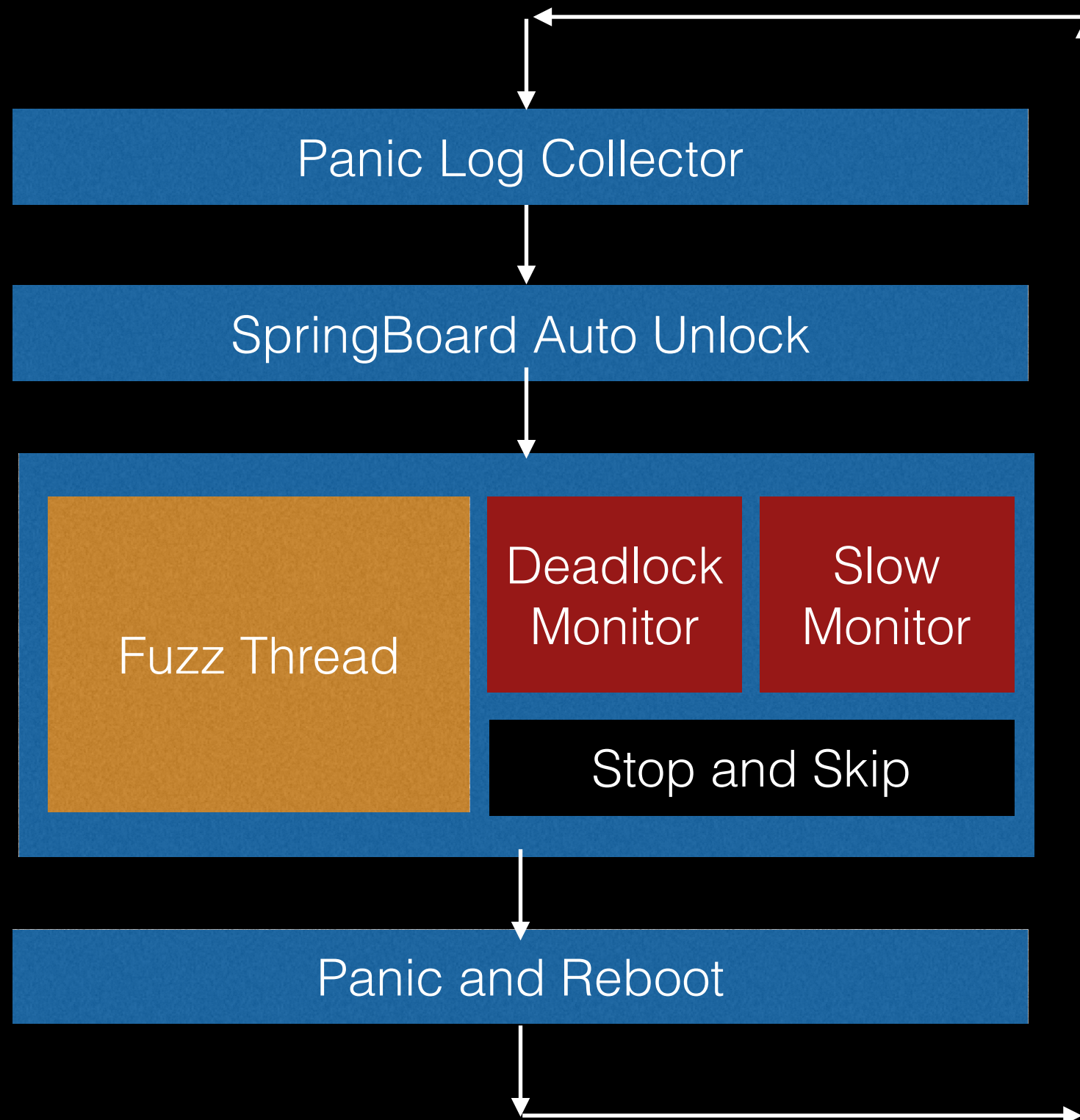


Part III

Fuzzing



Fuzzing Application's Architecture





Fuzzing Elements

- Fuzzing IOConnectMapMemory
 - If overwriting clientMemoryForType?
- Fuzzing IOConnectCallMethod
 - If overwriting externalMethod?
 - If overwriting getTargetAndMethodForIndex?
 - If overwriting getExternalMethodForIndex?

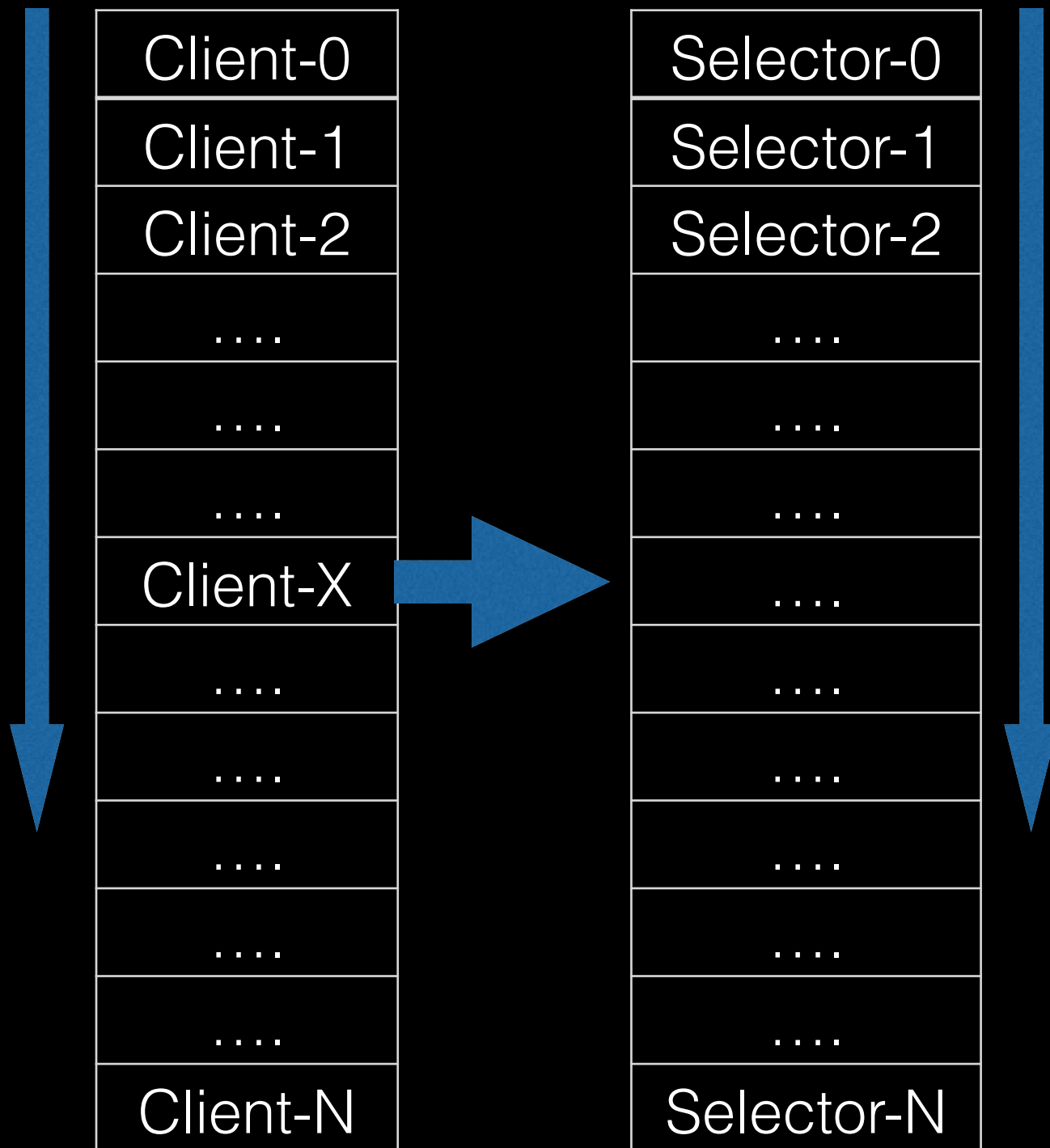


Fuzzing Elements

- Fuzzing IOConnectSetCFProperties
 - Client/Service
 - If overwriting setProperties?
- Fuzzing IOConnectTrap
 - If overwriting getTargetAndTrapForIndex?
 - If overwriting getExternalTrapForIndex?



Unavailable Interfaces Identification



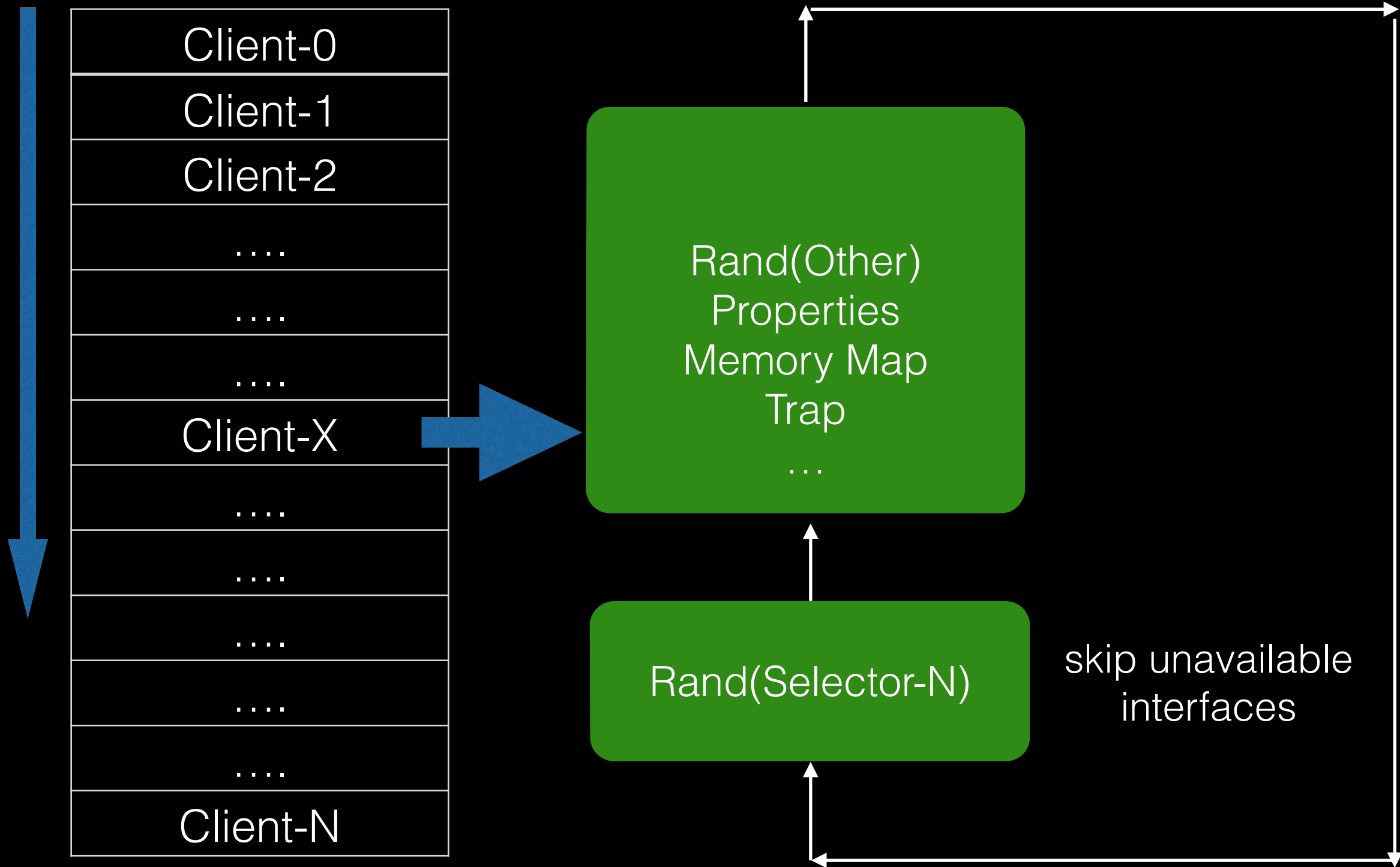
- inevitable panic interfaces
- deadlock interfaces
- slowly processing interfaces



Fuzz Thread



Fuzzing(III): Fuzz Thread





- MobileSubstrate: inject fuzzing into system process
- Implement clients' initialization in system process
- Use `mach_port_space_info` to get the client mach port



Part IV

Experimental Results



Setup

- Hardware
 - iPhone 4S
 - MacBook
- Software
 - iOS 8.1.2
 - Open-source XNU 2782.1.97



Vulnerability Case 1

IODataQueue

```
void IODataQueue::free()
{
    if (dataQueue) {
        IOFreeAligned(dataQueue, round_page(dataQueue->queueSize +
DATA_QUEUE_MEMORY_HEADER_SIZE));
        dataQueue = NULL;

        if (notifyMsg) {
            IOFree(notifyMsg, sizeof(mach_msg_header_t));
            notifyMsg = NULL;
        }
    }

    super::free();

    return;
}
```



Vulnerability Case 1

- Details
 - HighlandParkAudioDeviceUserClient-clientMemoryForType-44
 - Use IODataQueue to share memories
 - Buffers in kalloc.4096 can be released into bigger kalloc zone
- Panic Logs
 - Unavailable address to read and write
 - A freed zone element has been modified...



Vulnerability Case 2

- IOResources's setProperties

```
IOReturn IOResources::setProperties( OSObject * properties )
{
...
    while( (key = OSDynamicCast(OSSymbol, iter->getNextObject()))
        { ...
            publishResource( key, dict->getObject(key) );
        }

...
    return( kIOReturnSuccess );
}
```

- IOResources is inherited from IOService



Vulnerability Case 2

- IOService::newUserClient

```
IOReturn IOService::newUserClient( task_t owningTask, void * securityID,
                                   UInt32 type, OSDictionary * properties,
                                   IOUserClient ** handler )
{
...
    temp = getProperty(gIOUserClientClassKey);
    if (temp) {
        if (OSDynamicCast(OSSymbol, temp))
            userClientClass = (const OSSymbol *) temp;
...
    }
...
    temp = OSMetaClass::allocClassWithName(userClientClass);
    if (!temp)
        return kIOReturnNoMemory;

    if (OSDynamicCast(IOUserClient, temp))
        client = (IOUserClient *) temp;
...
}
...
}
```




Vulnerability Case 2

Exploiting

- IOResources can be bounded to any client as a service
- A new attack surface
- Fuzzing it

END

Last

Black Hat Sound Bytes

- An information export approach to dump all NSObject subclasses' information.
- An effective fuzzing framework to fuzz IOKit in iOS
- Several vulnerabilities sharing