



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

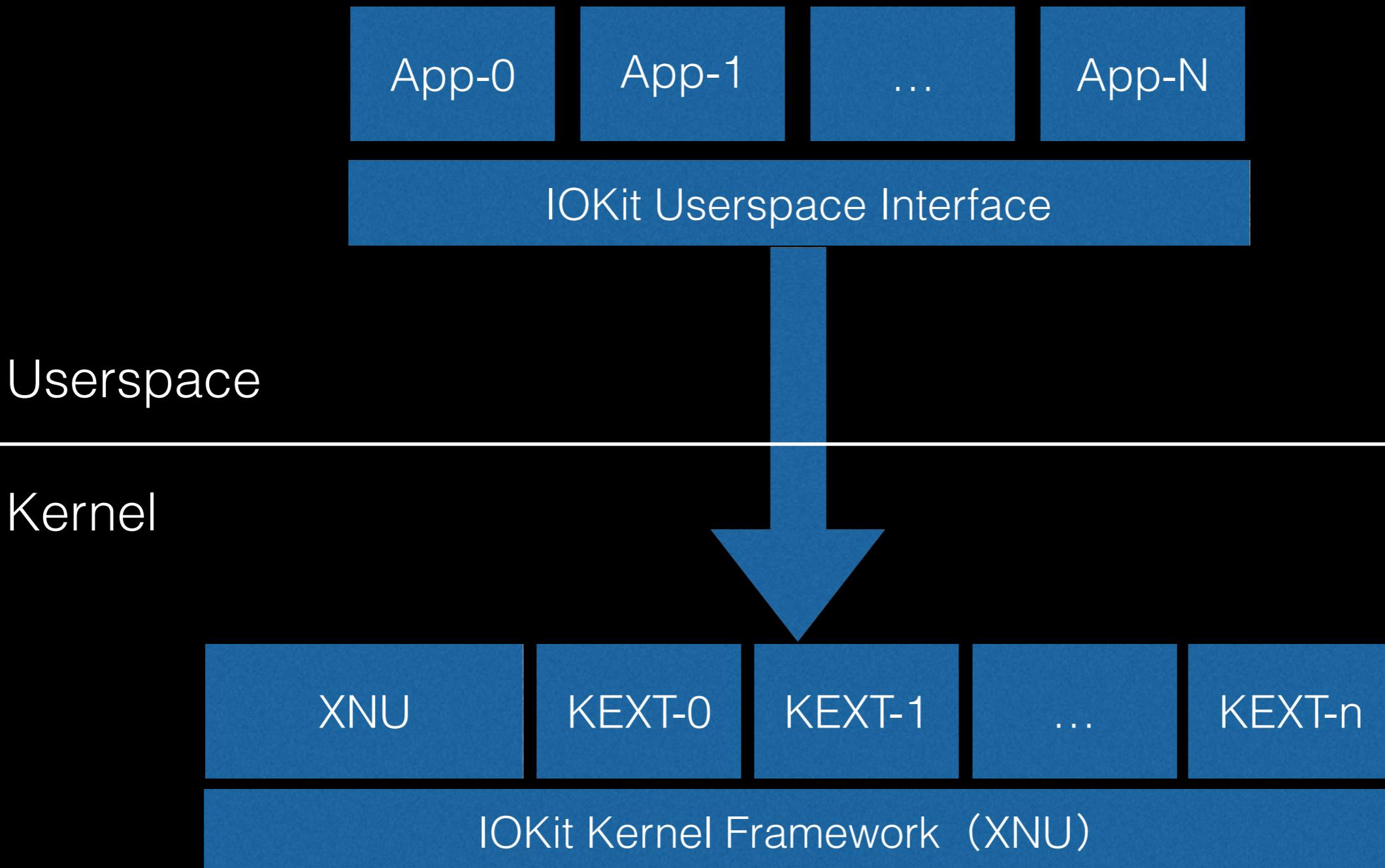
- Introduction
- Information Extraction
- Fuzzing
- Results

Part I

Introduction



IOKit





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: AppleARMPMPowerSource

BundleID: com.apple.driver.AppleARMPlatform
vtableaddr: 0x804c71a8
instance size: 0x184

SuperclassNames:

IOPMPowerSource
IOService
IORRegistryEntry
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 __ZN15IORRegistryEntry13setPropertiesEP8OSObject 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



Pick Out Vtable

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



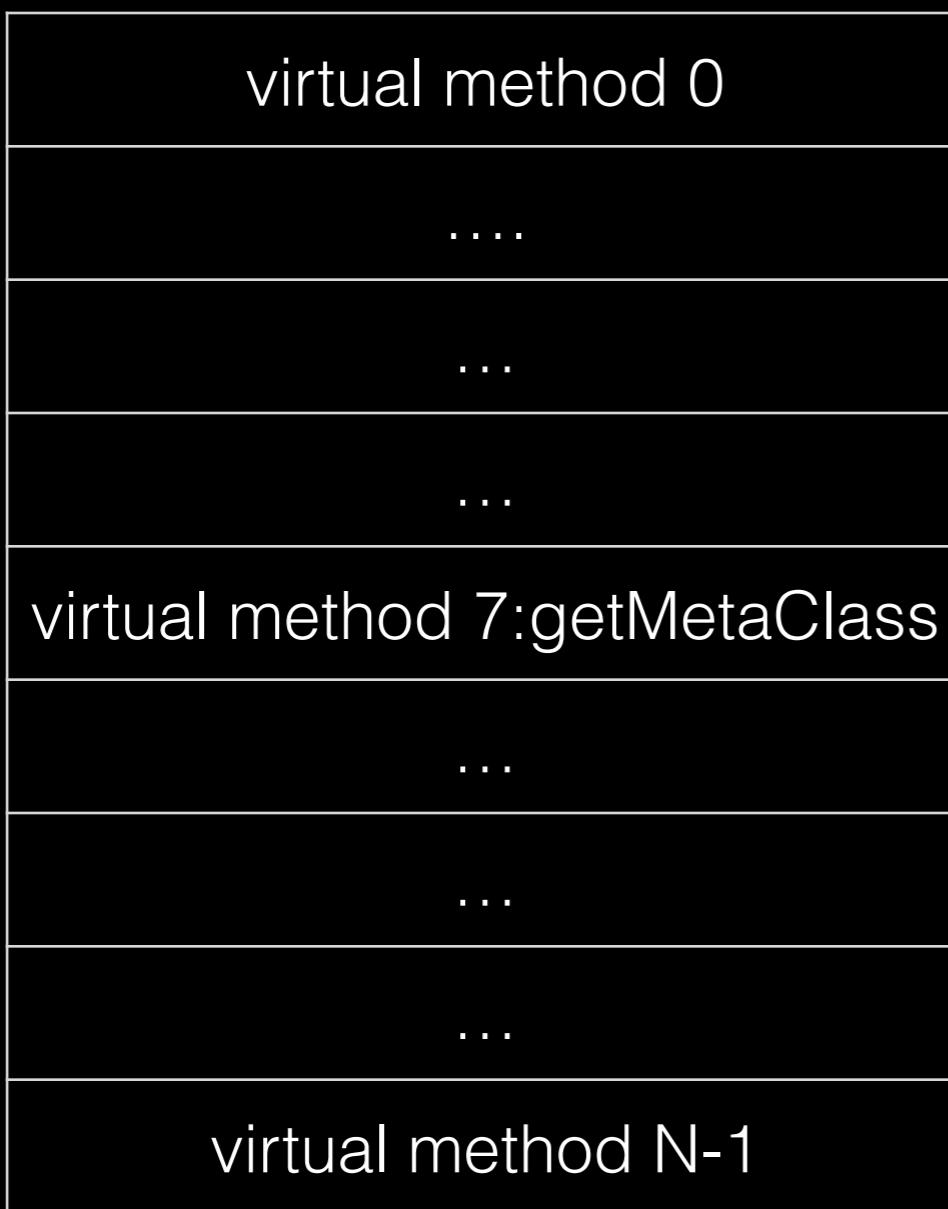
Vtable Layout

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
 $x \in (\text{TEXT_StartAddress}, \text{TEXT_EndAddress})$
or $x \in (\text{PERLINKTEXT_StartAddress}, \text{PERLINKTEXT_EndAddress})$
- $N \in [14, \infty)$
- Starting after at least four all-0 bytes



Vtable Characteristic



→ 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

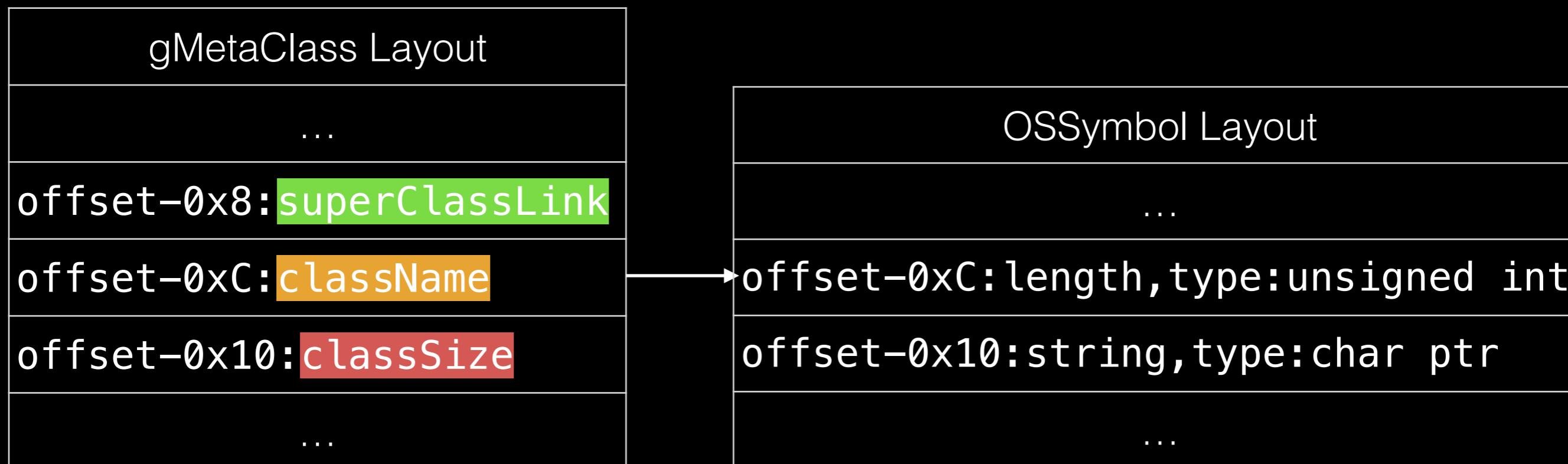
```
addr_b1:LDR R0 =#imm2  
addr_b2:ADD R0,PC  
addr_b3:BX  LR
```



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



gMetaClass Layout

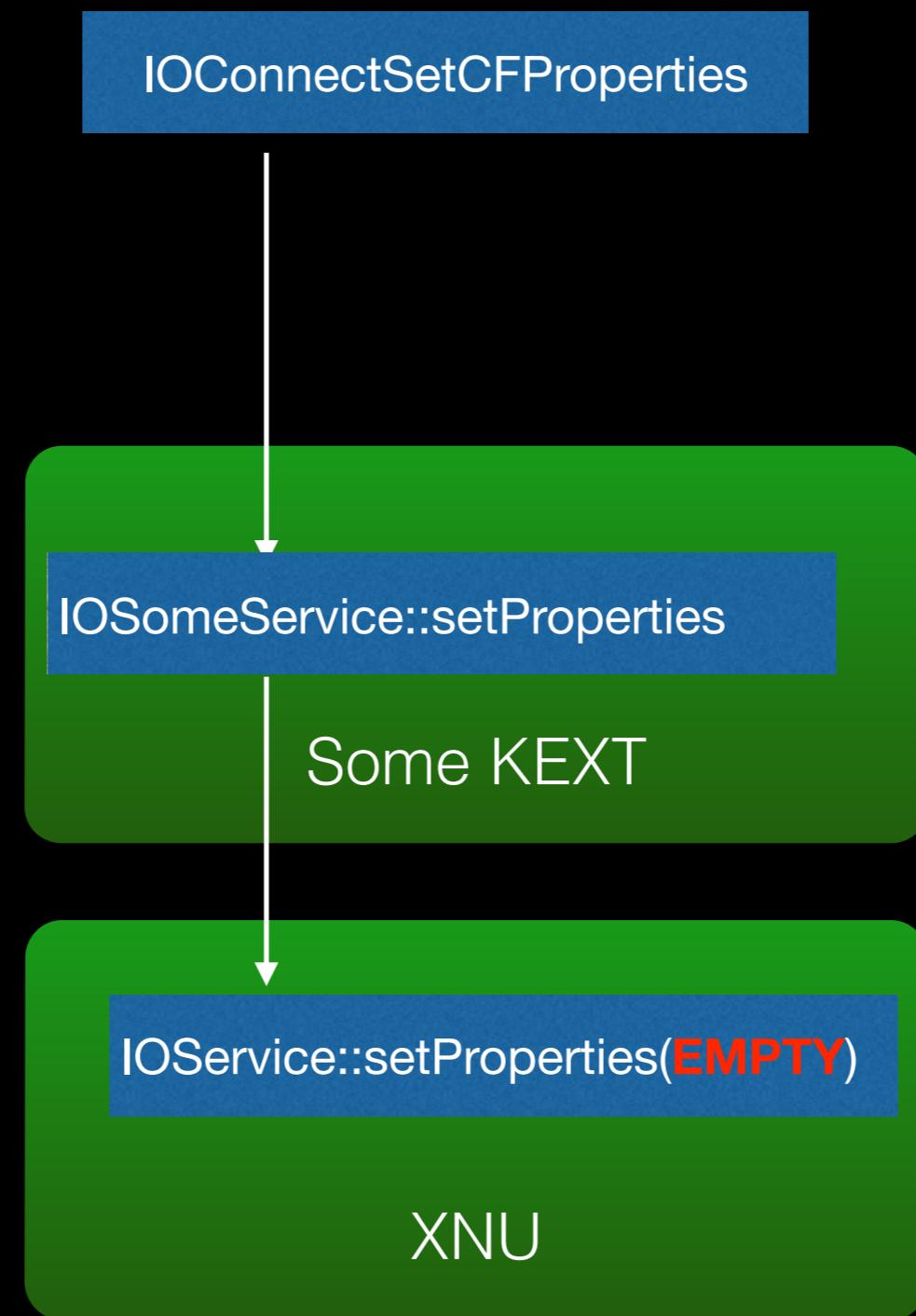


- classSize - unsigned int
- superClassLink - OSMetaClass ptr
 - Backwards to OSObject
 - All inheritance relationships



Functionality Provided by KEXT

Example





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:

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

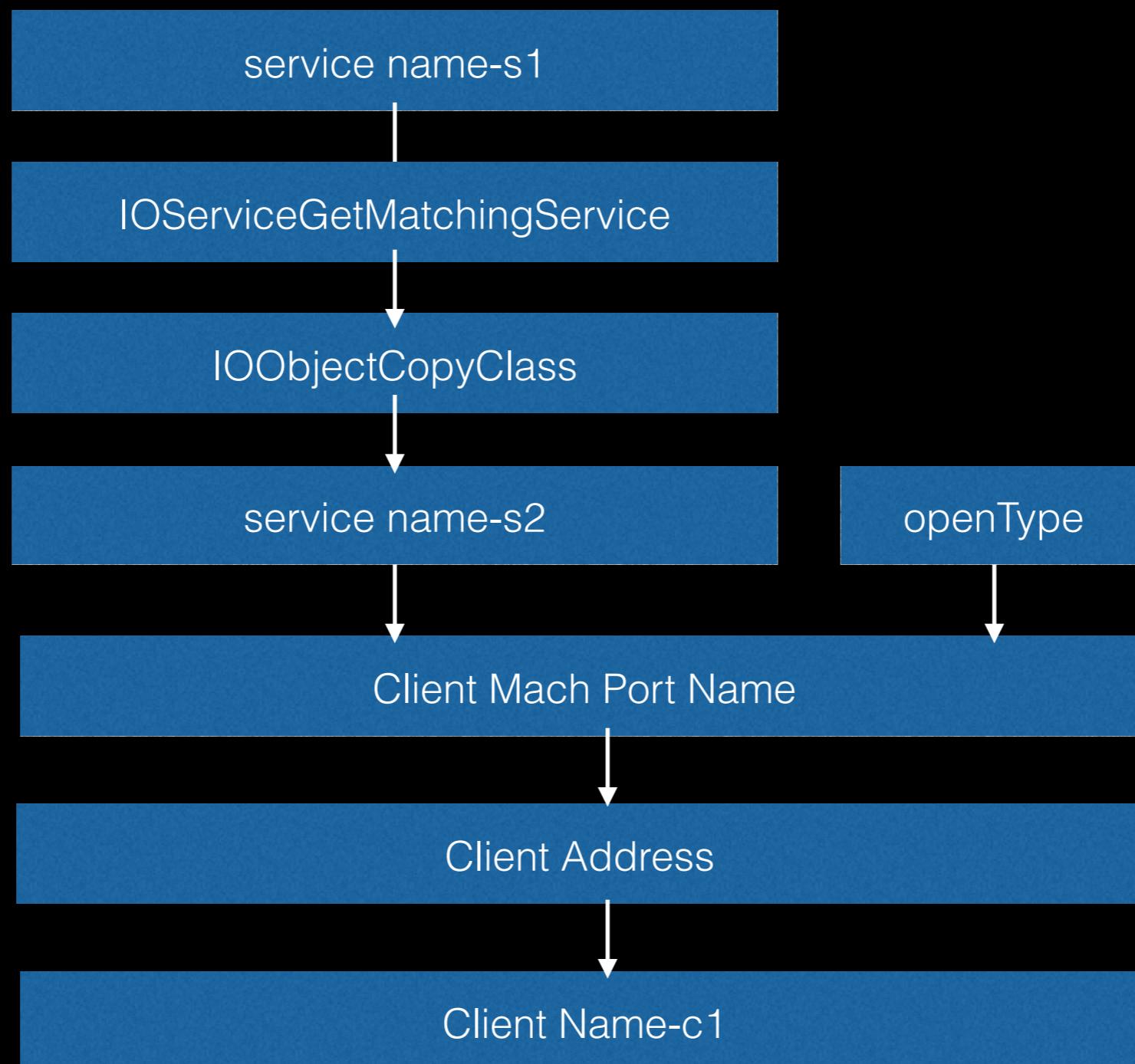
vtableaddr:0x807341f8

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

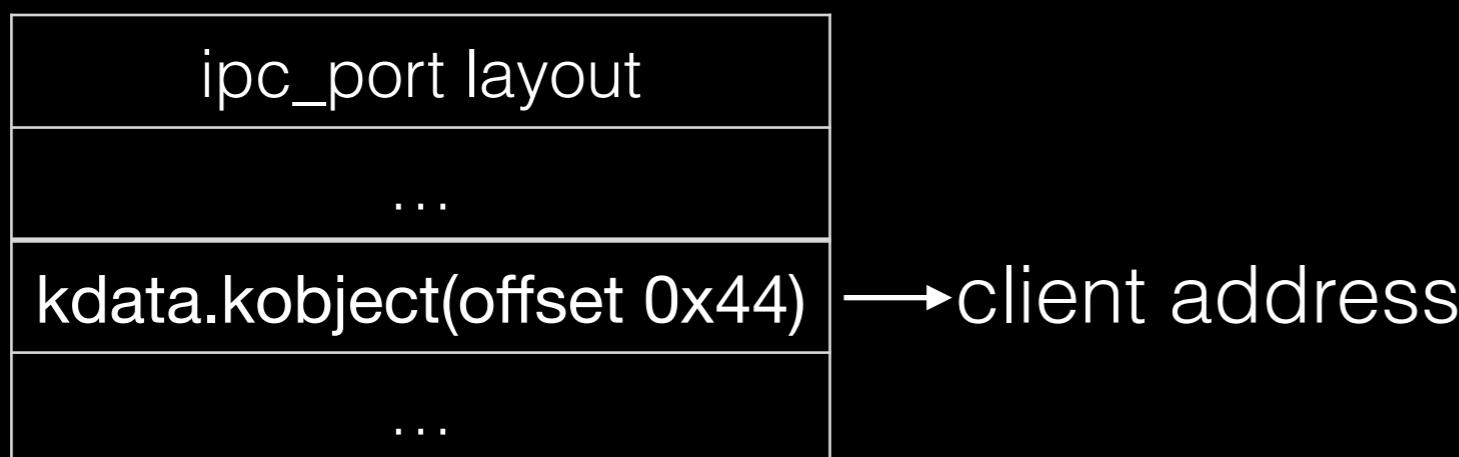
```
IOReturn IOGeneralMemoryDescriptor::wireVirtual(IODirection forDirection)
{
    .....
        OSReportWithBacktrace("IOMemoryDescriptor 0x%lx prepared read
only", VM_KERNEL_ADDRPERM(this));
    .....
}
```

- **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

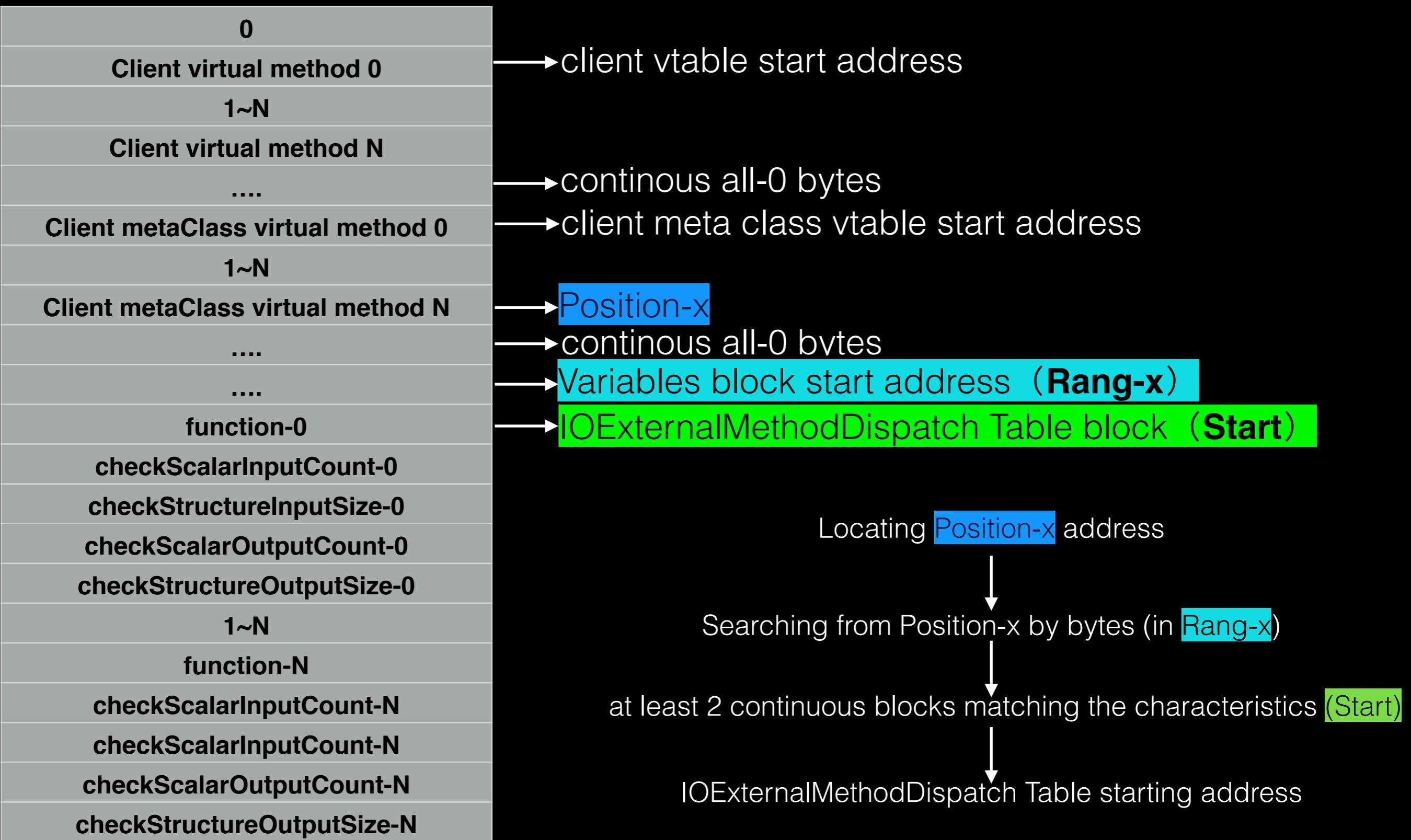
- IOExternalMethodDispatch fields

Function	
checkScalarInputCount	$\text{Function} \in (\text{TEXT_START}, \text{TEXT_END})$ $\text{Function} \in (\text{PRELINK_TEXT_START}, \text{PRELINK_TEXT_END})$
checkScalarInputCount	$\text{checkScalarInputCount} = 0xffffffff$ or $\text{checkScalarInputCount} \in [0, 0xffff]$
checkScalarOutputCount	$\text{checkScalarInputCount} = 0xffffffff$ or $\text{checkScalarInputCount} \in [0, 0xffff]$
checkStructureOutputSize	$\text{checkScalarOutputCount} = 0xffffffff$ or $\text{checkScalarOutputCount} \in [0, 0xffff]$
	$\text{checkStructureOutputSize} = 0xffffffff$ or $\text{checkStructureOutputSize} \in [0, 0xffff]$

- Table length ≥ 2



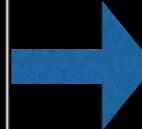
IOExternalMethodDispatch Table Dump





Complemental Mechanism

virtual method 0
....
...
...
overwrite externalMethod
...
...
...
virtual method N-1

externalMethod START
...
LDR Rn0,#imm0 —> check address-0 with the characteristic
...
ADD Rn0,PC
...
 LDR Rn1,#imm1 —> check address-1 with the characteristic
...
ADD Rn1,PC
...
LDR Rnn,#immn —> check address-n with the characteristic
...
ADD Rnn,PC
...
externalMethod END



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



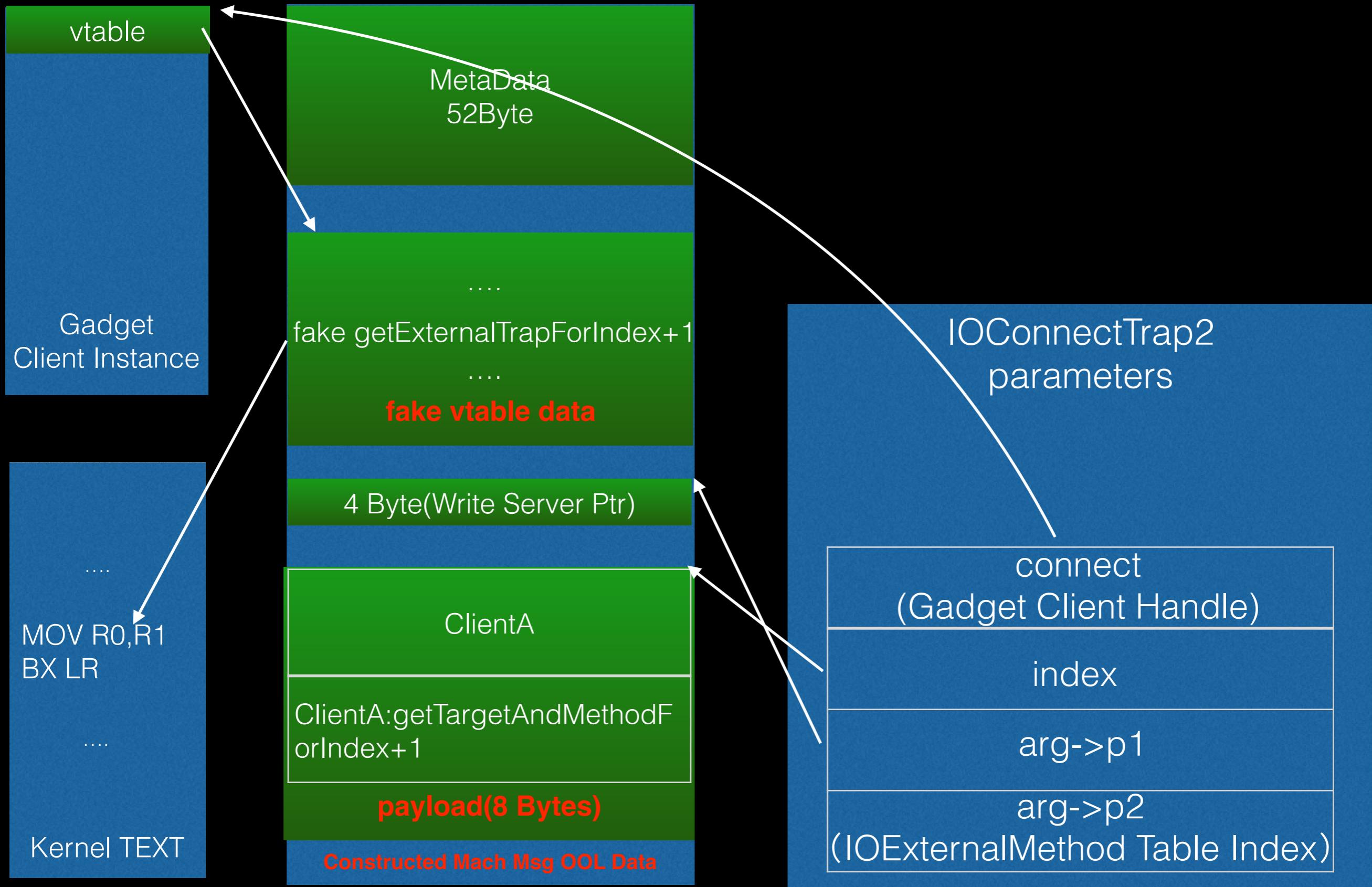
Carrier

- 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



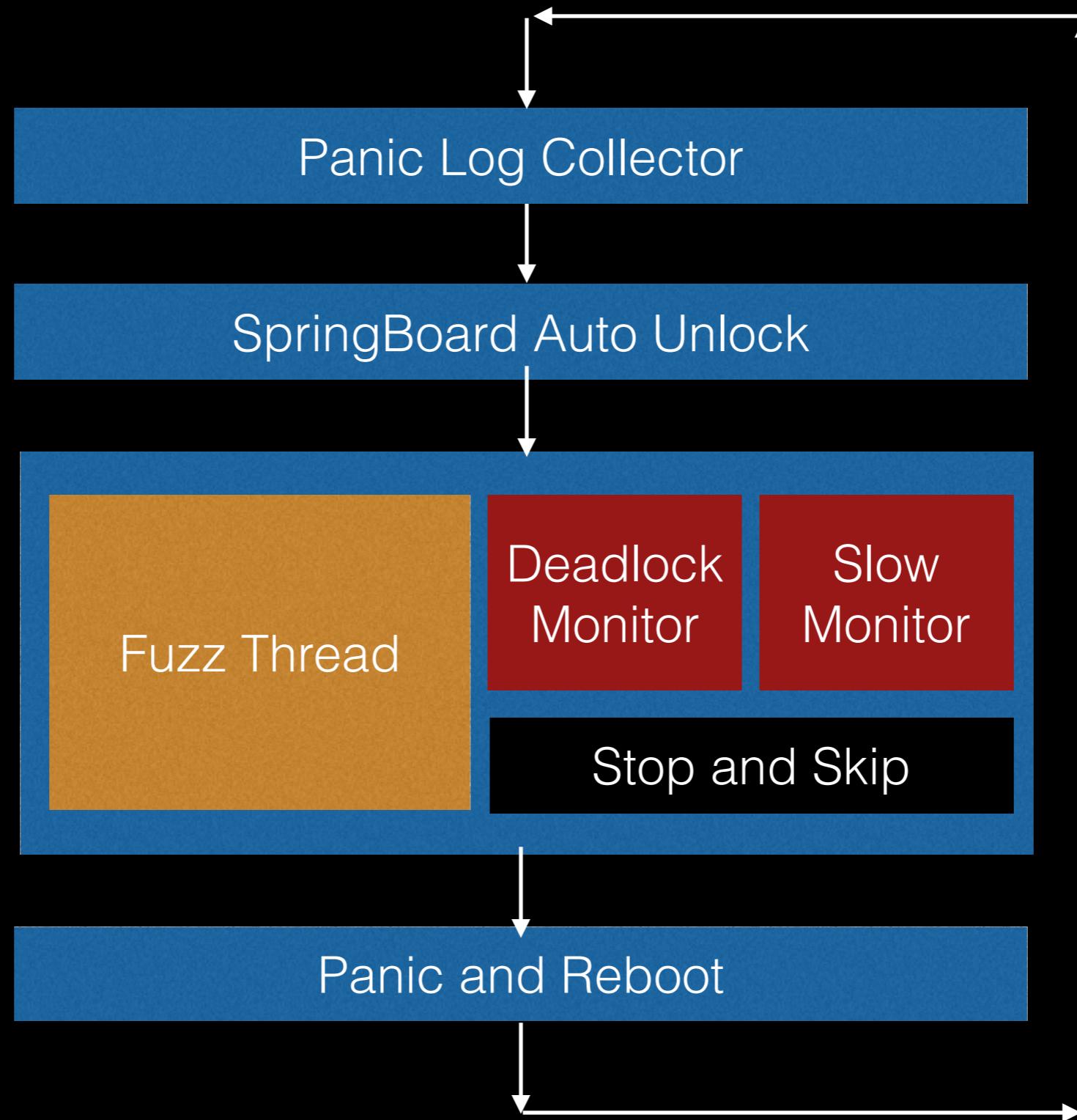


Part III

Fuzzing



Fuzzing Application's Architecture





Fuzzing Elements

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

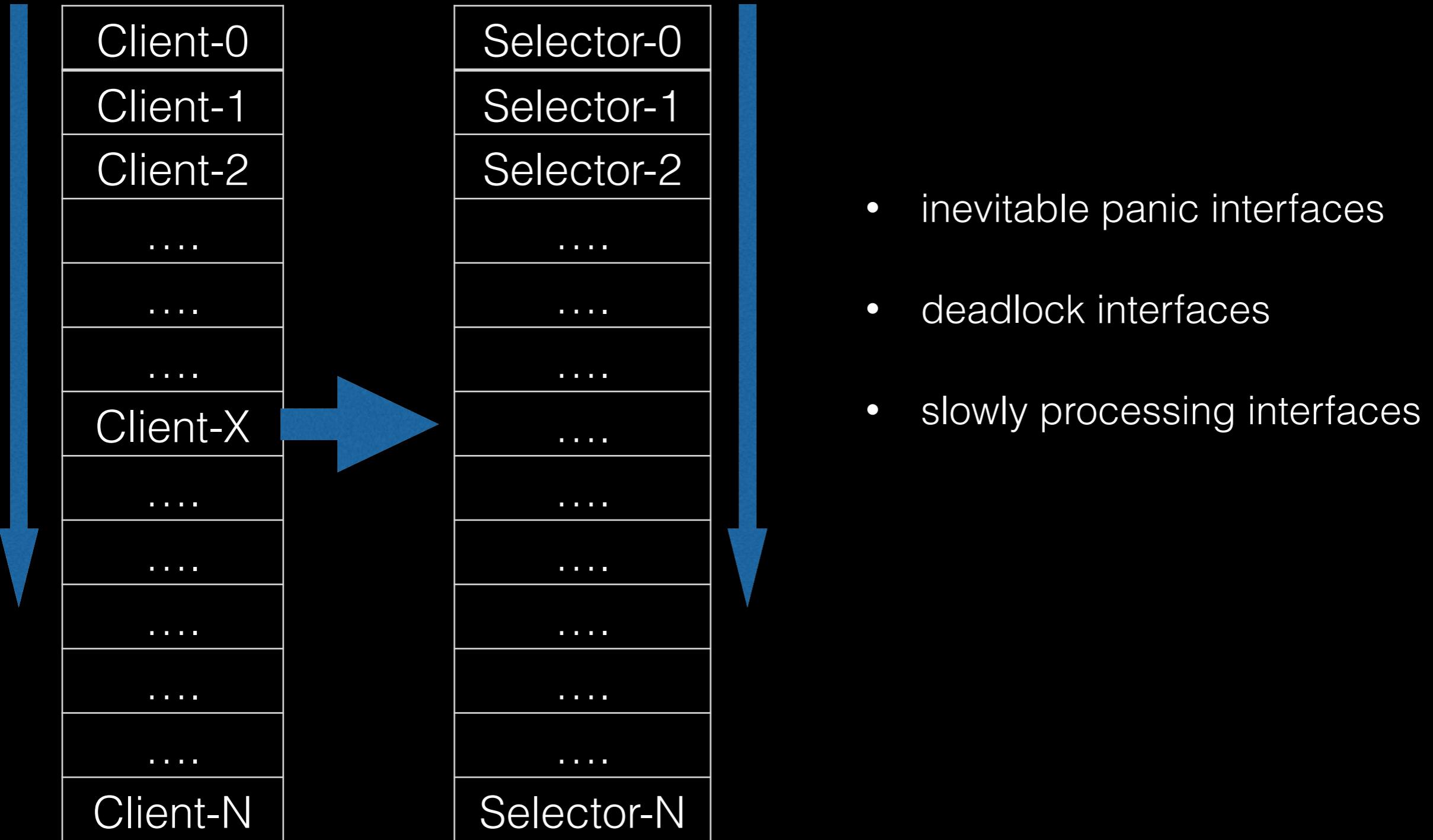


Fuzzing Elements

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



Unavailable Interfaces Identification

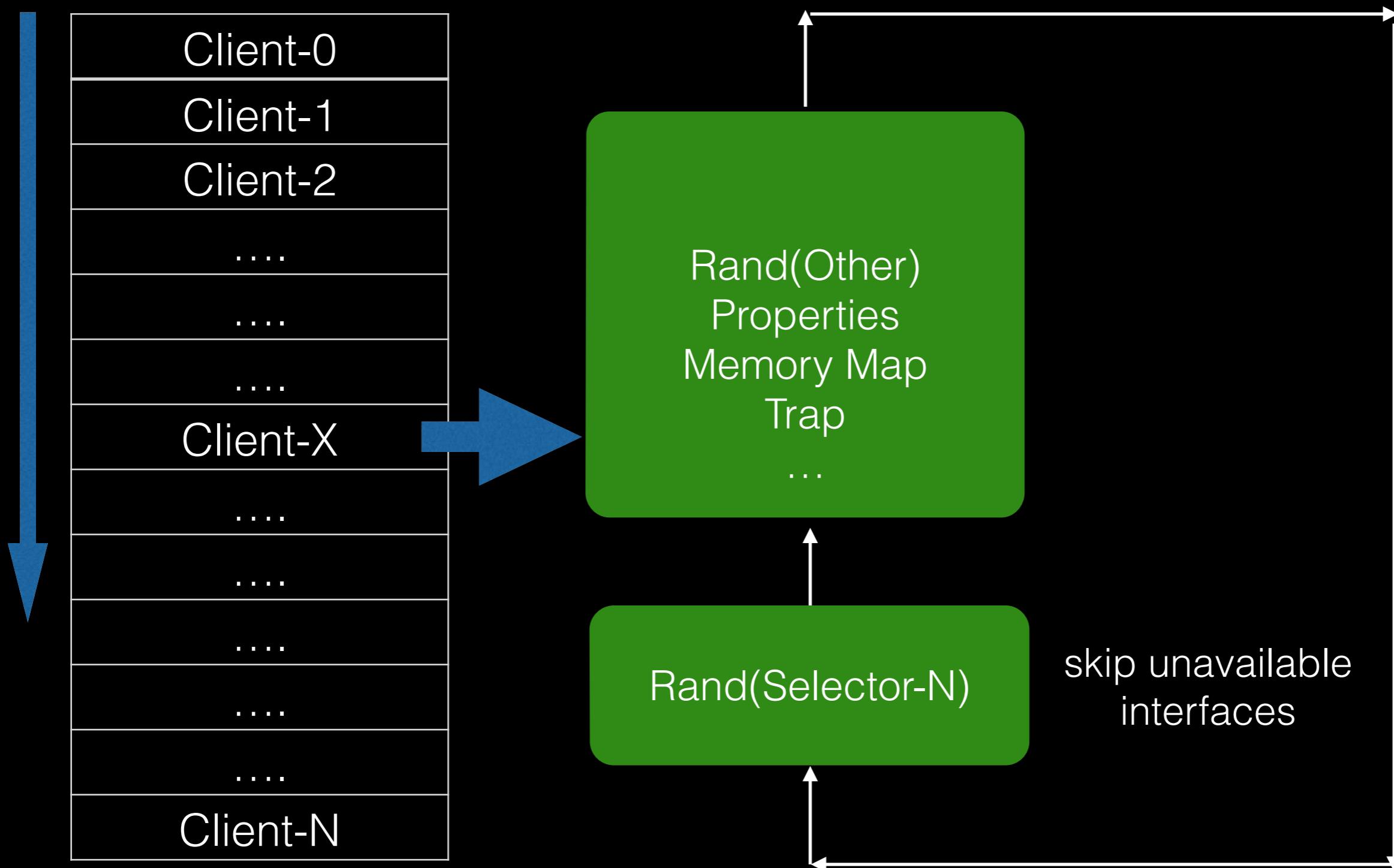




Fuzz Thread



Fuzzing(III): Fuzz Thread





Fuzzing(III): Fuzz Thread

Fuzzing Application

System Process

- 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 OSObject subclasses' information.
- An effective fuzzing framework to fuzz IOKit in iOS
- Several vulnerabilities sharing