One bite and all your dreams will come true: Analyzing and Attacking Apple Kernel Drivers

Xiaolong Bai & Min (Spark) Zheng

Xiaolong Bai

Alibaba Security Engineer

Ph.D. graduated from Tsinghua University

Published papers on the top 4: S&P,
Usenix Security, CCS, NDSS

Twitter, Weibo, Github: bxl1989

Min (Spark) Zheng

Alibaba Security Expert

Ph.D. graduated from The CUHK

Twitter@SparkZheng
Weibo@蒸米spark



1 Overview

2 New vulns in Apple drivers

3 Obstacles in analyzing Apple drivers

4 Ryuk: a new static analysis tool to analyze drivers

5 Ryuk-Fuzz: combine fuzzing with static analysis

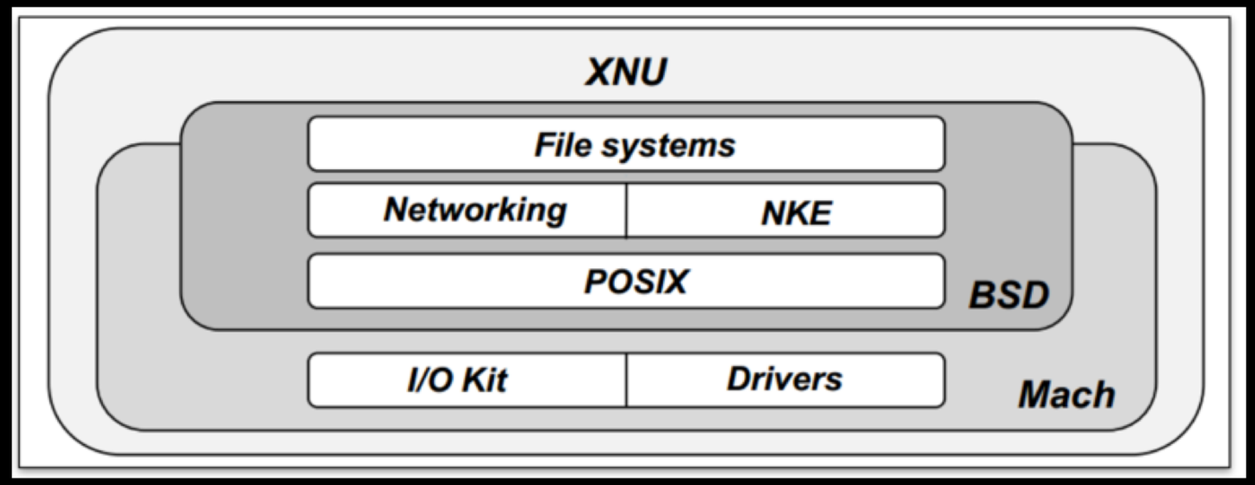
6 Conclusions

Overview

Every driver is a kernel extension (.kext) sharing the same space with the kernel

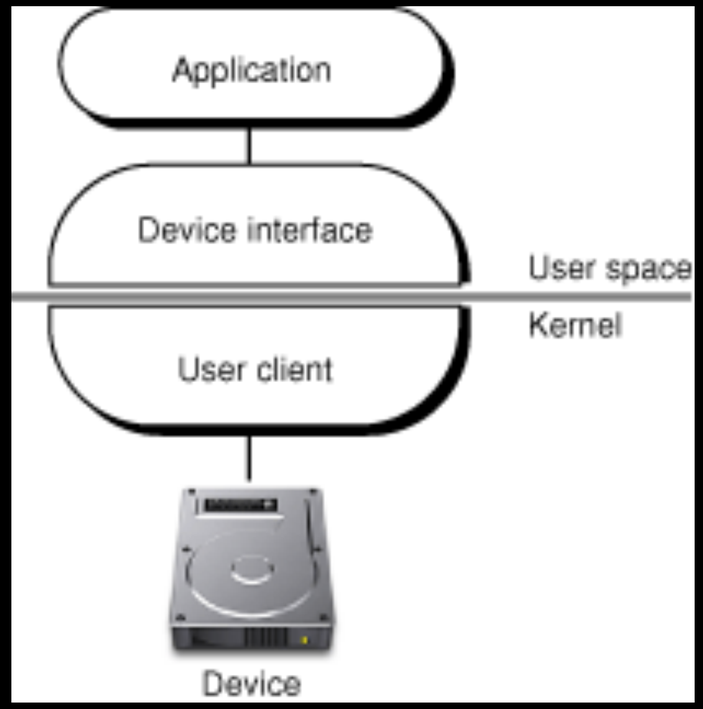
Location of driver binaries:

- On macOS: /System/Library/Extensions
- On iOS: integrated with kernel in kernelcache



Userclient: kernel objects for drivers to provide service to programs in userspace

Userclient is the interface between user-space applications and devices



In order to provide services, userclients need to implement several callback methods:

- externalMethod: Provide methods that can be called in user-space
- clientMemoryForType: Share memory with apps in user-space
- registerNotificationPort: Allow user-space app to register for notifications
- clientClose: Close connection with user-space app
- …

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

- Callback to provide methods to userspace program
- selector: to select method in userclient
- arguments: arguments passed to the selected method
- dispatch: a struct representing the method to be called
- target: the target userclient for the method to be called on
- reference: reference to send results back to userspace program

Despite of strict sandbox restriction, some userclients are still be accessible to sandboxed apps on iOS:

- IOHIDLibUserClient
- IOMobileFramebufferUserClient
- IOSurfaceAcceleratorClient
- AppleJPEGDriverUserClient
- IOAccelDevice2, IOAccelSharedUserClient2, IOAccelCommandQueue2
- AppleKeyStoreUserClient
- IOSurfaceSendRight, IOSurfaceRootUserClient

New vulns in Apple drivers

Drivers are good targets for exploiting the kernel

- Share the same space with the kernel
- Have kernel privileges
- Some programmed by third-party vendors, not kernel developer
- Code quality is not guaranteed

Drivers are frequently exploited in attacks against the kernel, including jailbreaks

Previous vulns in drivers used in jailbreaks:

- 11 (v0rtex/electra): IOSurfaceRoot (CVE-2017-13861)
- 9 (pangu): IOMobileFrameBuffer (CVE-2016-4654)
- 8 (TaiG): IOHIDFamily (CVE-2015-5774)
- 7 (pangu): AppleKeyStore (CVE-2014-4407)

Next, let’s have a look at some new vulns we recently discovered in Apple drivers!

CVE-2017-7119

- Information leakage in IOFirewireFamily
- Caused by an uninitialized stack variable
- In IOFireWireUserClient::externalMethod

```
case kIsochChannel_Allocate:
{
    IOFireWireUserClient * fw_uc = OSDynamicCast( IOFireWireUserClient, targetObject );
    if( fw_uc )
    {
        UserObjectHandle outChannelHandle;
        result = fw_uc->isochChannel_Create((bool)arguments->scalarInput[0],
                                            (UInt32)arguments->scalarInput[1],
                                            (IOFWSpeed)arguments->scalarInput[2],
                                            &outChannelHandle);
        arguments->scalarOutput[0] = (uint64_t) outChannelHandle;
    }
    else
    {
        result = kIOReturnBadArgument;
    }
    break;
}
```

CVE-2017-7119

- In IOFireWireUserClient::isochChannel_Create

```
IOReturn
IOFireWireUserClient::isochChannel_Create (
    bool                    inDoIRM,
    UInt32                  inPacketSize,
    IOFWSpeed               inPrefSpeed,
    UserObjectHandle *  outChannelHandle )
{
    // this code the same as IOFireWireController::createIsochChannel
    // must update this code when controller changes. We do this because
    // we are making IOFWUserIsochChannel objects, not IOFWIsochChannel
    // objects

    IOReturn error = kIOReturnSuccess ;
    IOFWUserIsochChannel * channel = OSTypeAlloc( IOFWUserIsochChannel );
    if ( channel )
    {
        if ( channel->init( getOwner()->getController(), inDoIRM, inPacketSize, inPrefSpeed ) )
        {
            fExporter->addObject( channel,
                    (IOFWUserObjectExporter::CleanupFunction) & IOFWUserIsochChannel::s_exporterCleanup,
                    outChannelHandle ) ;
        }
```

CVE-2017-7119

- In IOFWUserObjectExporter::addObject

```
IOReturn
IOFWUserObjectExporter::addObject ( OSObject * obj, CleanupFunction cleanupFunction, IOFireWireLib::UserObjectHandle *
    outHandle )
{
    IOReturn error = kIOReturnSuccess ;
    lock () ;
    // if at capacity, expand pool
    if ( fObjectCount == fCapacity )
    {
        unsigned newCapacity = fCapacity + ( fCapacity >> 1 ) ;
        if ( newCapacity > 0xFFFE )
            newCapacity = 0xFFFE ;
        if ( newCapacity == fCapacity ) // can't grow!
        {
            DebugLog( "Can't grow object exporter\n" ) ;
            error = kIOReturnNoMemory ;
        }
    }
```

- outHandle should be set with the index of new-added object
- But, when newCapacity reaches 0xFFFE, new object will not be added and outHandle will not be set

CVE-2017-7119

```
case kIsochChannel_Allocate:
{
    IOFireWireUserClient * fw_uc = OSDynamicCast( IOFireWireUserClient, targetObject );
    if( fw_uc )
    {
        UserObjectHandle outChannelHandle;
        result = fw_uc->isochChannel_Create((bool)arguments->scalarInput[0],
                                            (UInt32)arguments->scalarInput[1],
                                            (IOFWSpeed)arguments->scalarInput[2],
                                            &outChannelHandle);
        arguments->scalarOutput[0] = (uint64_t) outChannelHandle;
    }
    else
    {
        result = kIOReturnBadArgument;
    }
    break;
}
```

- Recall in IOFireWireUserClient::externalMethod
- outHandle is unintialized and returned to userspace
- Information leak!

CVE-2017-7119

- Can be exploited to get kernel slide, defeat kaslr, e.g.

```
* thread #1, stop reason = breakpoint 2.1
    frame #0: 0xffffff7f856947ac IOFireWireFamily`IOFireWireUserClie
nt::isochChannel_Create(this=0xffffff80177a2a00, inDoIRM=false, inPa
cketSize=0, inPrefSpeed=kFWSpeed100MBit, outChannelHandle=0xffffff91
340b3b48) at IOFireWireUserClient.cpp:4504 [opt]
(lldb) x/5g $r8
0xffffff91340b3b48: 0xffffff8004ebc0b6 0xffffff8016a8d000
0xffffff91340b3b58: 0xffffff80177a2a00 0x0000000000000039
0xffffff91340b3b68: 0xffffff80218791f4
```

```
(lldb) dis -a 0xffffff8004ebc0b6
kernel`IOEventSource::closeGate:
    0xffffff8004ebc0a0 <+0>:  pushq  %rbp
    0xffffff8004ebc0a1 <+1>:  movq   %rsp, %rbp
    0xffffff8004ebc0a4 <+4>:  pushq  %rbx
    0xffffff8004ebc0a5 <+5>:  pushq  %rax
    0xffffff8004ebc0a6 <+6>:  movq   %rdi, %rbx
    0xffffff8004ebc0a9 <+9>:  movq   0x30(%rbx), %rdi
    0xffffff8004ebc0ad <+13>: movq   (%rdi), %rax
    0xffffff8004ebc0b0 <+16>: callq  *0x180(%rax)
    0xffffff8004ebc0b6 <+22>: movq   0x40(%rbx), %rax
    0xffffff8004ebc0ba <+26>: movq   (%rax), %rbx
    0xffffff8004ebc0bd <+29>: testq  %rbx, %rbx
    0xffffff8004ebc0c0 <+32>: je     0xffffff8004ebc0d5
    0xffffff8004ebc0c2 <+34>: leaq   0x14cd57(%rip), %rdi
    0xffffff8004ebc0c9 <+41>: callq  0xffffff8004897880
    0xffffff8004ebc0ce <+46>: movq   %rax, 0x18(%rbx)
    0xffffff8004ebc0d2 <+50>: incl   0x28(%rbx)
    0xffffff8004ebc0d5 <+53>: addq   $0x8, %rsp
    0xffffff8004ebc0d9 <+57>: popq   %rbx
    0xffffff8004ebc0da <+58>: popq   %rbp
    0xffffff8004ebc0db <+59>: retq
```

```
FFFFFF80008BC0A0 ; __int64 __fastcall IOEventSource::closeGate(IOEventSo
FFFFFF80008BC0A0                 public __ZN13IOEventSource9closeGateEv
FFFFFF80008BC0A0 __ZN13IOEventSource9closeGateEv proc near
FFFFFF80008BC0A0                 push    rbp
FFFFFF80008BC0A1                 mov     rbp, rsp
FFFFFF80008BC0A4                 push    rbx
FFFFFF80008BC0A5                 push    rax
FFFFFF80008BC0A6                 mov     rbx, rdi
FFFFFF80008BC0A9                 mov     rdi, [rbx+30h]
FFFFFF80008BC0AD                 mov     rax, [rdi]
FFFFFF80008BC0B0                 call    qword ptr [rax+180h]
FFFFFF80008BC0B6                 mov     rax, [rbx+40h]
FFFFFF80008BC0BA                 mov     rbx, [rax]
FFFFFF80008BC0BD                 test    rbx, rbx
FFFFFF80008BC0C0                 jz      short loc_FFFFFF80008BC0D5
FFFFFF80008BC0C2                 lea     rdi, _pal_rtc_nanotime_info
FFFFFF80008BC0C9                 call    __rtc_nanotime_read
FFFFFF80008BC0CE                 mov     [rbx+18h], rax
FFFFFF80008BC0D2                 inc     dword ptr [rbx+28h]
FFFFFF80008BC0D5
FFFFFF80008BC0D5 loc_FFFFFF80008BC0D5:                   ; CODE XREF: IO
FFFFFF80008BC0D5                 add     rsp, 8
FFFFFF80008BC0D9                 pop     rbx
FFFFFF80008BC0DA                 pop     rbp
FFFFFF80008BC0DB                 retn
FFFFFF80008BC0DB __ZN13IOEventSource9closeGateEv endp
```

Kernel slide = 0x4ebc0b6-0x8bc0b6 = 0x4600000

Though outChannelHandle is only 32bit, but enough

since the high 32bit is always 0xffffff80 here

A Use-After-Free vulnerability in the driver IOAcceleratorFamily2

- IOAccelDisplayPipeUserClient2::s_transaction_end

```
__int64 __cdecl IOAccelDisplayPipeUserClient2::s_transaction_end(
{
  IOAccelDisplayPipeUserClient2 *v3; // rax@1
  void *v4; // rdx@2
  IOAccelDisplayPipe2 *v5; // rdi@2
  __int64 result; // rax@3

  v3 = a1;
  if ( BYTE1(a1->member50) )
  {
    v4 = (void *)a3->structureInput;
    BYTE1(a1->member50) = 0;
    v5 = (IOAccelDisplayPipe2 *)a1->member29;
    if ( v5 )
      result = IOAccelDisplayPipe2::transaction_end(v5, v3, v4);
    else
      result = 3758097113LL;
  }
  else
  {
    result = 3758097122LL;
  }
  return result;
}
```

All IOAccelDisplayPipeUserClient2 share the same IOAccelDisplayPipe2

A Use-After-Free vulnerability in the driver IOAcceleratorFamily2

- IOAccelDisplayPipe2::transaction_end

IOAccelDisplayPipe2 contains a link list of IOAccelDisplayPipe2Transaction2.
IOAccelDisplayPipe2::transaction_end traverse the link list to find a transaction

```
v12 = (char *)&this->mAccelDisplayPipeTransaction2_Head;
do
{
  foundTransaction = *(IOAccelDisplayPipeTransaction2 **)v12;
  if ( !*(_QWORD *)v12 )
  {
    v7 = 0xE00002F0;
    goto LABEL_35;
  }
  v12 = (char *)&foundTransaction->nextTransaction;
}
while ( HIDWORD(foundTransaction->mTransactionIndexInPipe) != LODWORD(transactionArgs_userInput->transactionIndex) );
```

A link list of transactions in this IOAccelDisplayPipe2

And further call IOAccelDisplayPipeTransaction2::set_transaction_args using the found transaction

```
v7 = IOAccelDisplayPipeTransaction2::set_transaction_args(foundTransaction, transactionArgs_userInput);
```

A Use-After-Free vulnerability in the driver IOAcceleratorFamily2

- IOAccelDisplayPipeTransaction2::set_transaction_args

**mAccelDisplayPipeUserClient2 is a member of IOAccelDisplayPipeTransaction2,
whose type is IOAccelDisplayPipeUserClient2,
representing which userclient the transaction belongs to**

```
  userclient = this->mAccelDisplayPipeUserClient2;
  eventSourceInUserClient = userclient->mInterruptEventSource;
  if ( !eventSourceInUserClient )
  {
    _this->mInterruptEventSource = 0LL;
LABEL_6:
    transactionArgs = (IOAccelDisplayPipeTransactionArgs *)_os_log_default_0;
    _os_log_internal(
      &dword_0,
      _os_log_default_0,
      17LL,
      IOAccelDisplayPipeTransaction2::set_transaction_args(IOAccelDisplayPipeTransactionArgs *)::_os_log_fmt,
      "IOReturn IOAccelDisplayPipeTransaction2::set_transaction_args(sIOAccelDisplayPipeTransactionArgs *)");
    _this->member40 = 0LL;
    goto LABEL_7;
  }
  ((void (*)(void))eventSourceInUserClient->vtable->__ZNK8OSObject6retainEv)();
```

The userclient that this transaction belongs to

A Use-After-Free vulnerability in the driver IOAcceleratorFamily2

- But, mAccelDisplayPipeUserClient2 can be released by calling IOServiceClose from user space, causing use-after-free

```
  userclient = this->mAccelDisplayPipeUserClient2;
  eventSourceInUserClient = userclient->mInterruptEventSource;
  if ( !eventSourceInUserClient )
  {
    _this->mInterruptEventSource = 0LL;
LABEL_6:
    transactionArgs = (IOAccelDisplayPipeTransactionArgs *)_os_log_default_0;
    _os_log_internal(
      &dword_0,
      _os_log_default_0,
      17LL,
      IOAccelDisplayPipeTransaction2::set_transaction_args(IOAccelDisplayPipeTransactionArgs *)::_os_log_fmt,
      "IOReturn IOAccelDisplayPipeTransaction2::set_transaction_args(sIOAccelDisplayPipeTransactionArgs *)");
    _this->member40 = 0LL;
    goto LABEL_7;
  }
  ((void (*)(void))eventSourceInUserClient->vtable->__ZNK8OSObject6retainEv)();
```

Userclient can be released from user space

Cause use-after-freee

Exploitation of the bug

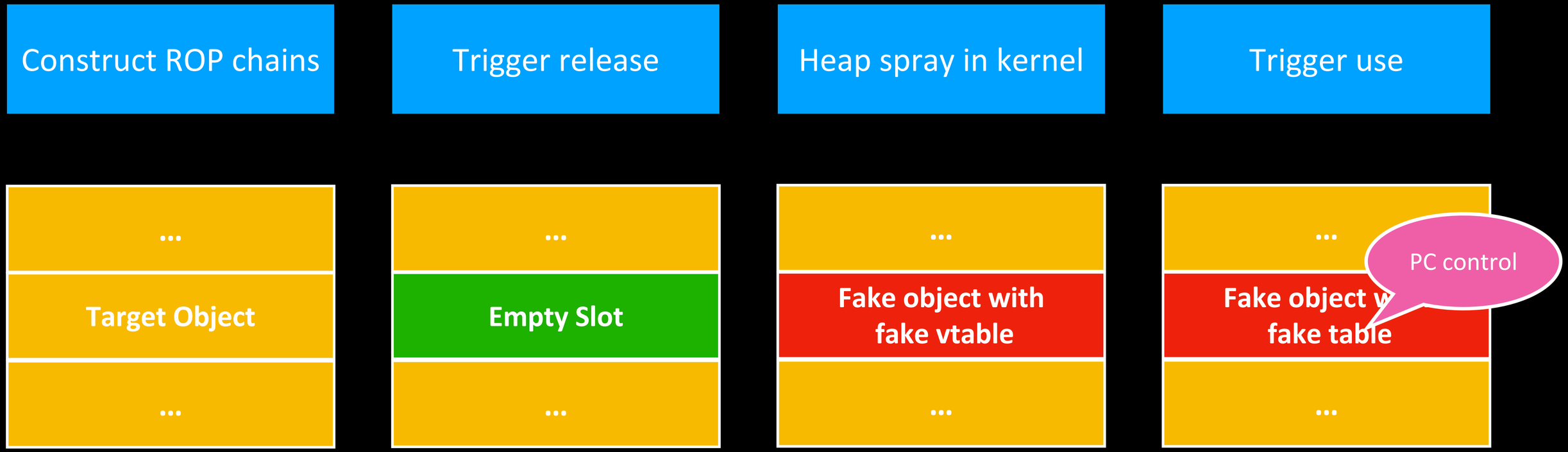
- Create 2 IOAccelDisplayPipeUserClient2
- Begin a IOAccelDisplayPipeTransaction2 by userclient1
- Release userclient1
- End the transaction from userclient2 —> use-after-free

The bug is not exploitable now!

- Reason: different IOAccelDisplayPipeUserClient2 do not share the same IOAccelDisplayPipe2 now !

How to exploit a Use-After-Free bug in the kernel for privilege escalation?

Basic flow:



A heap spray method on macOS:

- Utilize OSUnserializeXML
- Set properties of a device

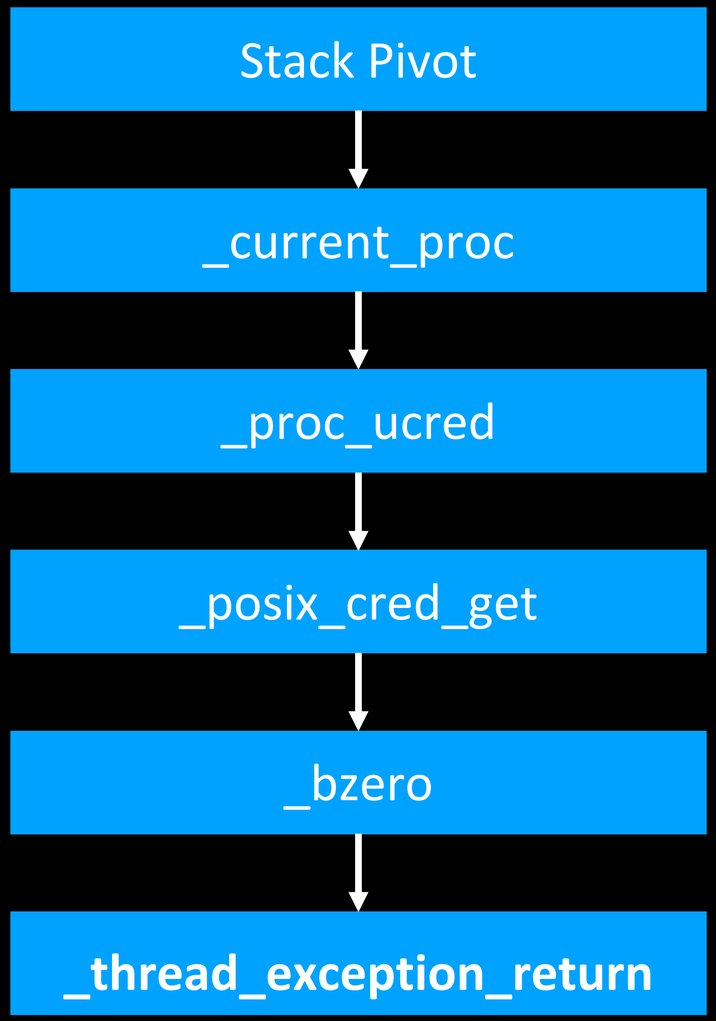
```
/* Routine io_registry_entry_set_properties */
kern_return_t is_io_registry_entry_set_properties(
    io_object_t registry_entry,
    io_buf_ptr_t properties,
    mach_msg_type_number_t propertiesCnt,
        kern_return_t * result) {
        ...
        obj = OSUnserializeXML( (const char *) data, propertiesCnt );
        ...
        res = entry->setProperties( obj );
}
```

A heap spray method on macOS:

- Some drivers keep any properties set by userspace, e.g., IOHIDEventService
- Pros: the sprayed data can be read; the head of sprayed data is controllable

```
/* Routine io_registry_entry_set_properties */
kern_return_t is_io_registry_entry_set_properties(
    io_object_t registry_entry,
    io_buf_ptr_t properties,
    mach_msg_type_number_t propertiesCnt,
        kern_return_t * result) {
        ...
        obj = OSUnserializeXML( (const char *) data, propertiesCnt );
        ...
        res = entry->setProperties( obj );
}
```

ROP chain for privilege escalation



Get ptr to struct proc of current process

Get ucred from struct proc, i.e., process owner's identity

Get ptr to struct cr_posix

Clear cr_uid, cr_ruid, cr_svuid in cr_posix struct

Exit kernel, return to userspace

Stack Pivot: Change current stack, for performing ROP in an elegant way

Step 1: Control RAX and affect RCX (Gadget P1)

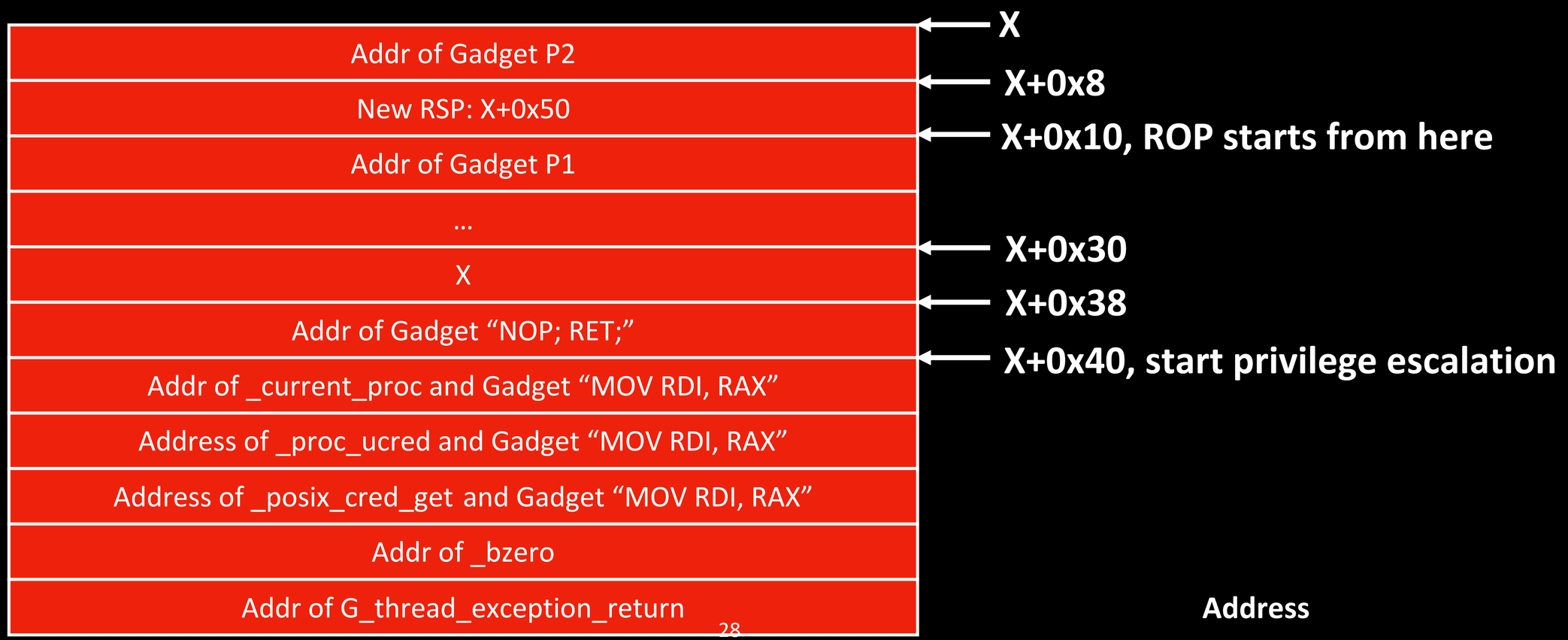
```
mov     rcx, [rax+30h]
mov     [rbp+var_50], rcx
call    qword ptr [rax]
```

Step 2: Modify RSP by controlled RCX (Gadget P2)

```
mov     rsp, [rcx+8]
mov     rbx, [rcx]
mov     rbp, [rcx+10h]
mov     r12, [rcx+18h]
mov     r13, [rcx+20h]
mov     r14, [rcx+28h]
mov     r15, [rcx+30h]
jmp     qword ptr [rcx+38h]
```

Layout of gadgets in a ROP chain

- Assume: RAX is controlled and [RAX+0x10] is to be called
- Store address X in RAX, X contains a ROP chain

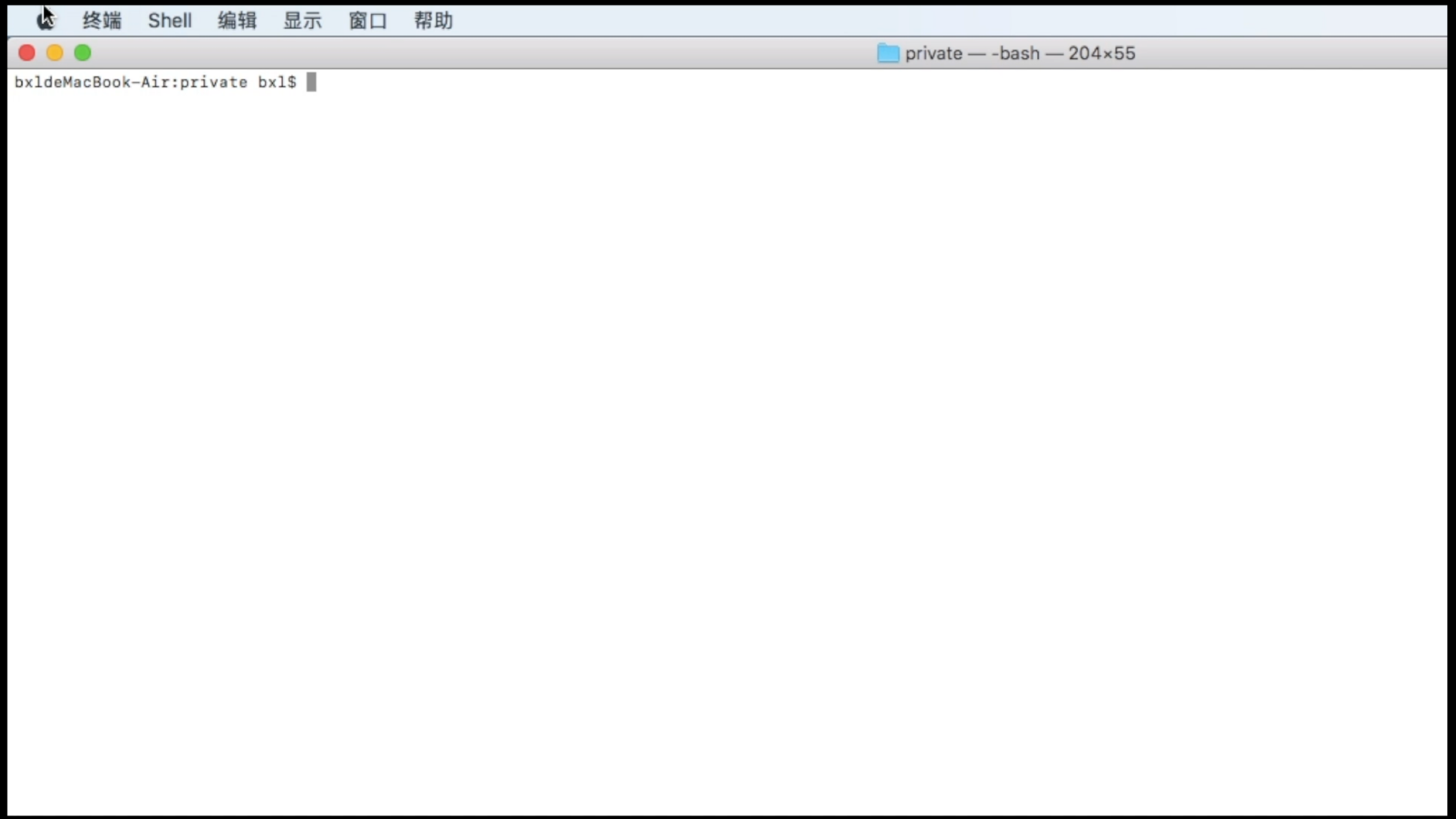


Privilege escalation on the macOS 10.13 and 10.13.2

```
sh-3.2# uname -a
Darwin bxldeMacBook-Air.local 17.0.0 Darwin Kernel Version 17.0.0: Thu Aug 24 21
:48:20 PDT 2017; root:xnu-4570.1.46~2/DEVELOPMENT_X86_64 x86_64
sh-3.2# whoami
root
sh-3.2#
```

```
sh-3.2# uname -a
Darwin bxldeMacBook-Air.local 17.3.0 Darwin Kernel Version 17.3.0: Thu Nov  9 18:09:22 PST 2017; root:xnu-4570.31.3~1/DEVELOPMENT_X86_64 x86_64
sh-3.2# whoami
root
sh-3.2#
```

Privilege escalation on macOS 10.13.3



Obstacles in analyzing Apple drivers

Apple drivers are mostly closed-source, in order to find vulnerabilities, we need to do some reverse engineering.

BUT, reverse engineering is not that easy…

Let’s first have a look at the decompiled code of driver binaries in IDA pro

The decompiled code of a macOS driver

```
char __fastcall IOSurfaceRootUserClient::taskHasEntitlement(IOSurfaceRootUserClient *this, task *a2,
{
  IOUserClient *v3; // rax@1
  const char *v4; // rdx@1
  __int64 v5; // rbx@1
  __int64 v6; // rsi@2
  __int64 v7; // rax@2
  char v8; // r14@3

  LODWORD(v3) = current_task(this, a2, a3);
  v5 = IOUserClient::copyClientEntitlement(v3, (task *)&"com.apple.private.iosurfaceinfo", v4);
  if ( v5 )
  {
    v6 = *off_C048;
    LODWORD(v7) = OSMetaClassBase::safeMetaCast(v5, *off_C048);
    if ( v7 )
      v8 = (*(int (__fastcall **)(__int64, __int64))(*(_QWORD *)v7 + 280LL))(v7, v6);
    else
      v8 = 0;
    (*(void (__fastcall **)(__int64))(*(_QWORD *)v5 + 40LL))(v5);
  }
  else
  {
    v8 = 0;
  }
  return v8;
}
```

Wrong parameter inference

Variable types are lost

Virtual function calls are not recognized

The decompiled code of a iOS driver

```
__int64 __fastcall sub_FFFFFFF00615A3D0(__int64 a1, __int64 a2, int a3)
{
  int v3; // w20
  __int64 v4; // x19
  __int64 v5; // x21
  __int64 result; // x0
  __int64 v7; // x0
  __int64 v8; // x21
  void (__fastcall *v9)(__int64, __int64); // x22
  __int64 v10; // x0
  signed __int64 v11; // x1

  v3 = a3;
  v4 = a2;
  v5 = (*(__int64 (**)(void))(*(_QWORD *)a1 + 1536LL))();
  result = sub_FFFFFFF006166F10(v4, off_FFFFFFF006E07190);
  if ( result )
  {
    if ( v5 )
    {
      v7 = (*(__int64 (__fastcall **)(__int64))(*(_QWORD *)v5 + 208LL))(v5);
      v8 = v7;
      if ( v7 )
      {
        (*(void (**)(void))(*(_QWORD *)v7 + 152LL))();
        v9 = *(void (__fastcall **)(__int64, __int64))(*(_QWORD *)v4 + 1488LL);
        v10 = (*(__int64 (__fastcall **)(__int64))(*(_QWORD *)v8 + 208LL))(v8);
```

No idea what the code represents

Actually, we want the decompiled code to be just like source code

```
void __cdecl IOAVControllerUserClient::start(IOAVControllerUserClient *this, IOAVController *provider)
{
  const void *v2; // x2
  IOAVControllerUserClient *v3; // x20
  IOAVController *v4; // x0
  unsigned __int64 v5; // x1
  IOWorkLoop *v6; // x21
  IOEventSource *v7; // x8

  v3 = this;
  v4 = (IOAVController *)OSMetaClassBase::safeMetaCast((OSMetaClassBase *)provider, off_FFFFFFF006EED5E0, v2);
  v3->member27 = (__int64)v4;
  if ( v4 )
  {
    v4->vtable->__ZNK8OSObject6retainEv((OSObject *)v4);
    if ( IOUserClient_vtableRef32->vtable.__ZN9IOService5startEPS_((IOService *)v3) )
    {
      v6 = v3->vtable->__ZNK9IOService11getWorkLoopEv((IOService *)v3);
      if ( !v6
        || (v7 = (IOEventSource *)sub_FFFFFFF00653ED58((OSObject *)v3, v5), (v3->member28 = (__int64)v7) == 0)
        || (unsigned int)v6->vtable->__ZN10IOWorkLoop14addEventSourceEP13IOEventSource(v6, v7) )
      {
        v3->vtable->__ZN24IOAVControllerUserClient4stopEPS_(v3);
      }
    }
  }
}
```

Ryuk: a new static analysis tool to analyze Apple drivers

Ryuk: a new static analysis tool aiming at making drivers' decompiled code more like source code

- Implemented as an IDA pro python script





***Ryuk: a character in the comics series Death Note, who loves eating apples.**

Ryuk’s features (functionalities):

- Identify name, size, and vtables for classes
- Recover function names: only for iOS drivers
- Resolve variable types: resolve and mark variable types
- Extend cross references: for members and virtual funcs
- UI support: for better reverse engineering experience
- Call graph generation: for inter-procedure analysis

Identify name and size for classes

- How? analyze specific global initialization functions
- A sample initialization function

```
public __GLOBAL__sub_I_IOSurfaceRootUserClient_cpp
IOSurfaceRootUserClient_cpp proc near
                        ; DATA XREF: __mod_init_func:000000000000E0B0 o
push    rbp
mov     rbp, rsp
lea     rdi, __ZN23IOSurfaceRootUserClient10gMetaClassE ; IOSurfaceRootUserClient::gMetaClass
lea     rsi, aIosurfacerootu ; "IOSurfaceRootUserClient"
mov     rdx, cs:__ZN12IOUserClient10gMetaClassE_0 ; IOUserClient::gMetaClass
mov     ecx, 150h
call    __ZN11OSMetaClassC2EPKcPKS_j ; OSMetaClass::OSMetaClass(char const*,OSMetaClass const*,uint)
lea     rax, off_10110
mov     cs:__ZN23IOSurfaceRootUserClient10gMetaClassE, rax ; IOSurfaceRootUserClient::gMetaClass
pop     rbp
retn
IOSurfaceRootUserClient_cpp endp
```

Metaclass object for the class

The class's name

The class's size

Identify name and size for classes

- With identified class names and sizes, we can create structures to represent these classes in IDA pro

```
[00000090 BYTES. COLLAPSED STRUCT IODMAEventSource. PRESS CTRL-NUMPAD+ TO EXPAND]
[00000078 BYTES. COLLAPSED STRUCT IOFilterInterruptEventSource. PRESS CTRL-NUMPAD+ TO EXPAND]
[00000060 BYTES. COLLAPSED STRUCT IOTimerEventSource. PRESS CTRL-NUMPAD+ TO EXPAND]
[000000E8 BYTES. COLLAPSED STRUCT IOBufferMemoryDescriptor. PRESS CTRL-NUMPAD+ TO EXPAND]
[00000078 BYTES. COLLAPSED STRUCT IODMACommand. PRESS CTRL-NUMPAD+ TO EXPAND]
[00000090 BYTES. COLLAPSED STRUCT IOInterleavedMemoryDescriptor. PRESS CTRL-NUMPAD+ TO EXPAND]
[000000D0 BYTES. COLLAPSED STRUCT IOMapper. PRESS CTRL-NUMPAD+ TO EXPAND]
[00000030 BYTES. COLLAPSED STRUCT IOMemoryCursor. PRESS CTRL-NUMPAD+ TO EXPAND]
[00000030 BYTES. COLLAPSED STRUCT IONaturalMemoryCursor. PRESS CTRL-NUMPAD+ TO EXPAND]
[00000030 BYTES. COLLAPSED STRUCT IOBigMemoryCursor. PRESS CTRL-NUMPAD+ TO EXPAND]
[00000030 BYTES. COLLAPSED STRUCT IOLittleMemoryCursor. PRESS CTRL-NUMPAD+ TO EXPAND]
[00000060 BYTES. COLLAPSED STRUCT IOMemoryDescriptor. PRESS CTRL-NUMPAD+ TO EXPAND]
[000000B0 BYTES. COLLAPSED STRUCT IOGeneralMemoryDescriptor. PRESS CTRL-NUMPAD+ TO EXPAND]
[00000188 BYTES. COLLAPSED STRUCT IOMemoryMap. PRESS CTRL-NUMPAD+ TO EXPAND]
[00000070 BYTES. COLLAPSED STRUCT IOMultiMemoryDescriptor. PRESS CTRL-NUMPAD+ TO EXPAND]
[00000030 BYTES. COLLAPSED STRUCT IORangeAllocator. PRESS CTRL-NUMPAD+ TO EXPAND]
[00000070 BYTES. COLLAPSED STRUCT IOSubMemoryDescriptor. PRESS CTRL-NUMPAD+ TO EXPAND]
[000000E0 BYTES. COLLAPSED STRUCT IOPlatformExpert. PRESS CTRL-NUMPAD+ TO EXPAND]
[000000F0 BYTES. COLLAPSED STRUCT IODTPlatformExpert. PRESS CTRL-NUMPAD+ TO EXPAND]
[00000098 BYTES. COLLAPSED STRUCT IOPlatformExpertDevice. PRESS CTRL-NUMPAD+ TO EXPAND]
[00000090 BYTES. COLLAPSED STRUCT IOPlatformDevice. PRESS CTRL-NUMPAD+ TO EXPAND]
[000000E0 BYTES. COLLAPSED STRUCT IOPanicPlatform. PRESS CTRL-NUMPAD+ TO EXPAND]
[000000B8 BYTES. COLLAPSED STRUCT IOCPU. PRESS CTRL-NUMPAD+ TO EXPAND]
[000000B8 BYTES. COLLAPSED STRUCT IOCPUInterruptController. PRESS CTRL-NUMPAD+ TO EXPAND]
[00000118 BYTES. COLLAPSED STRUCT IODTNVRAM. PRESS CTRL-NUMPAD+ TO EXPAND]
[00000098 BYTES. COLLAPSED STRUCT IODMAController. PRESS CTRL-NUMPAD+ TO EXPAND]
[000000A0 BYTES. COLLAPSED STRUCT IOInterruptController. PRESS CTRL-NUMPAD+ TO EXPAND]
[000000C8 BYTES. COLLAPSED STRUCT IOSharedInterruptController. PRESS CTRL-NUMPAD+ TO EXPAND]
```

Identify vtables for classes

- In C++, if a class has any virtual function, the first member of the class will be a pointer to a structure called vtable
- Vtable contains pointers to the class's all virtual funcs

```
class A {
public:
    virtual void foo1();
    virtual void foo2();

};
void A::foo1(){}
void A::foo2(){}
int main(){
    A *a = new A();
    a->foo2();
}
```

Source code

```
__const:0000000100001020                        public __ZTV1A
__const:0000000100001020 ; `vtable for'A
__const:0000000100001020 __ZTV1A                dq 0
__const:0000000100001028                        dq offset __ZTI1A
__const:0000000100001030                        dq offset __ZN1A4foo1Ev
__const:0000000100001038                        dq offset __ZN1A4foo2Ev
```

vtable

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  void *v3; // ST10_8@1

  v3 = (void *)operator new(8uLL);
  memset(v3, 0, 8uLL);
  A::A(v3, 0LL);
  (*(void (__fastcall **)(void *))(*(_QWORD *)v3 + 8LL))(v3);
  return 0;
}
```

Retrieve pointer to the vtable

Get the pointer to the virtual function

Decompiled code

Identify vtables for classes

- Hint: ahead of a class's vtable, there is a pointer pointing to the class's metaclass object

```
const:000000000000D710                  public __ZN23IOSurfaceRootUserClient9metaClassE
const:000000000000D710 ; IOSurfaceRootUserClient::metaClass
const:000000000000D710 __ZN23IOSurfaceRootUserClient9metaClassE dq offset __ZN23IOSurfaceRootUserClient10gMetaClassE
const:000000000000D710                                          ; DATA XREF: IOSurfaceRoot::newUserClient(task *,void
const:000000000000D710                                          ; IOSurfaceRoot::userClientForTask(task *)+3A↑o ...
const:000000000000D710                                          ; IOSurfaceRootUserClient::gMetaClass
const:000000000000D718                  public __ZN23IOSurfaceRootUserClient10superClassE
const:000000000000D718 ; IOSurfaceRootUserClient::superClass
const:000000000000D718 __ZN23IOSurfaceRootUserClient10superClassE dq offset __ZN12IOUserClient10gMetaClassE ; IOUserC
const:000000000000D720                  public __ZTV23IOSurfaceRootUserClient
const:000000000000D720 ; `vtable for'IOSurfaceRootUserClient
const:000000000000D720 __ZTV23IOSurfaceRootUserClient dq 0
const:000000000000D728                  dq 0
const:000000000000D730                  dq offset __ZN23IOSurfaceRootUserClientD1Ev
const:000000000000D730                                          ; DATA XREF: IOSurfaceRootUserClient::IOSurfaceRootUse
const:000000000000D730                                          ; IOSurfaceRootUserClient::IOSurfaceRootUserClient(OSM
const:000000000000D730                                          ; IOSurfaceRootUserClient::~IOSurfaceRootUserClient()
const:000000000000D738                  dq offset __ZN23IOSurfaceRootUserClientD0Ev ; IOSurfaceRootUserClient::~IOSurf
const:000000000000D740                  dq offset __ZNK8OSObject7releaseEi ; OSObject::release(int)
const:000000000000D748                  dq offset __ZNK8OSObject14getRetainCountEv ; OSObject::getRetainCount(void)
```

Pointer to the class's global metaclass object

VTable start

Identify vtables for classes

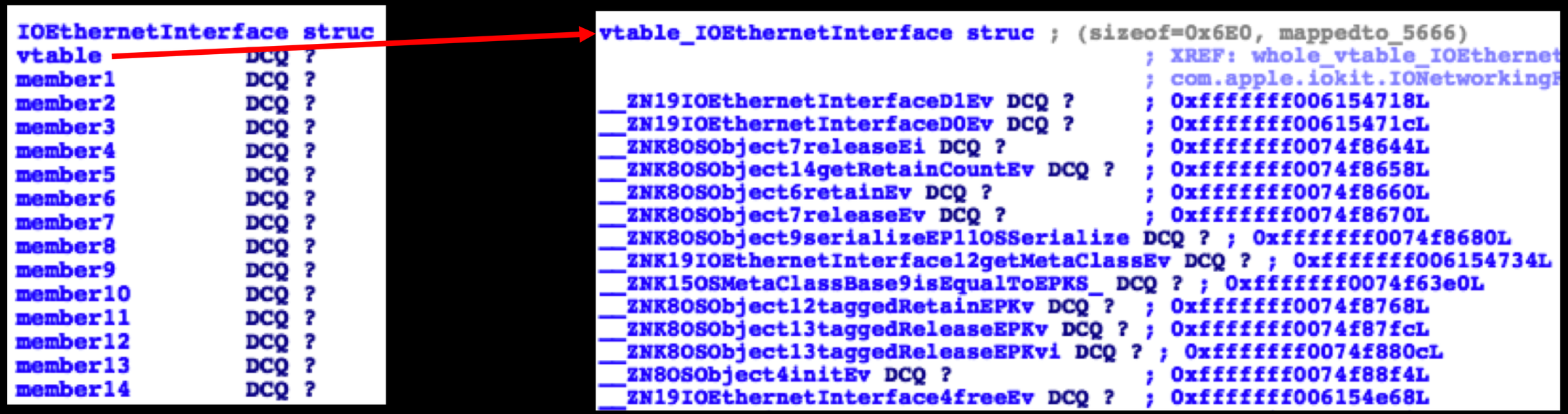
- Recall that the initialization function shows where the metaclass object is, with above hint, we can use it to find the vtable

```
public __GLOBAL__sub_I_IOSurfaceRootUserClient_cpp
IOSurfaceRootUserClient_cpp proc near
                        ; DATA XREF: __mod_init_func:000000000000E0B0 o
push    rbp
mov     rbp, rsp
lea     rdi, __ZN23IOSurfaceRootUserClient10gMetaClassE ; IOSurfaceRootUserClient::gMetaClass
lea     rsi, aIosurfacerootu ; "IOSurfaceRootUserClient"
mov     rdx, cs:__ZN12IOUserClient10gMetaClassE_0 ; IOUserClient::gMetaClass
mov     ecx, 150h
call    __ZN11OSMetaClassC2EPKcPKS_j ; OSMetaClass::OSMetaClass(char const*,OSMetaClass const*,uint)
lea     rax, off_10110
mov     cs:__ZN23IOSurfaceRootUserClient10gMetaClassE, rax ; IOSurfaceRootUserClient::gMetaClass
pop     rbp
retn
IOSurfaceRootUserClient_cpp endp
```

Metaclass object for the class

Identify vtables for classes

- After identified, create structure in IDA pro standing for the vtable, each member is pointer to a virtual func
- Set the first member of class structure as a pointer to this vtable structure

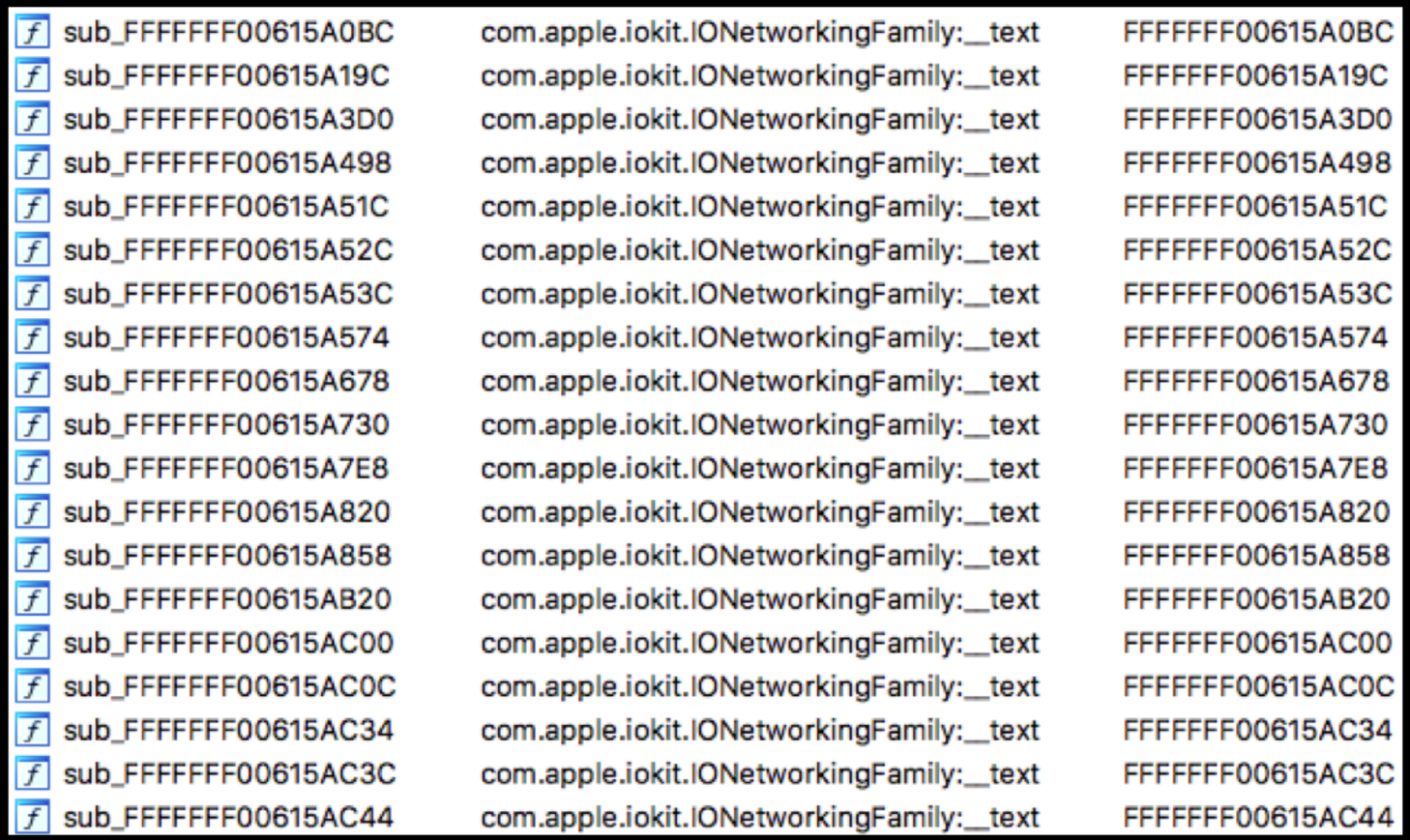


Class structure

Vtable structure

Recover function names (only for iOS)

- iOS driver: symbols are totally lost



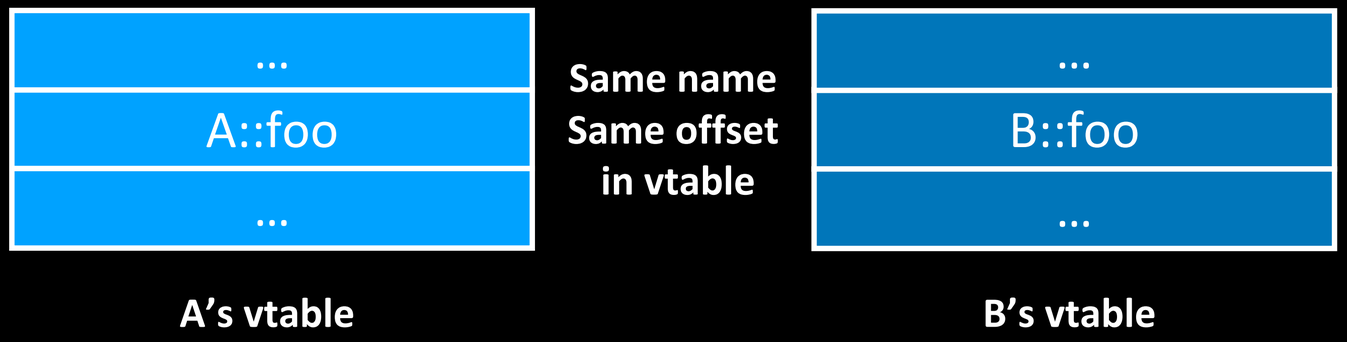
sub_FFFFFFF00615A0BC	com.apple.iokit.IONetworkingFamily:__text	FFFFFFF00615A0BC
sub_FFFFFFF00615A19C	com.apple.iokit.IONetworkingFamily:__text	FFFFFFF00615A19C
sub_FFFFFFF00615A3D0	com.apple.iokit.IONetworkingFamily:__text	FFFFFFF00615A3D0
sub_FFFFFFF00615A498	com.apple.iokit.IONetworkingFamily:__text	FFFFFFF00615A498
sub_FFFFFFF00615A51C	com.apple.iokit.IONetworkingFamily:__text	FFFFFFF00615A51C
sub_FFFFFFF00615A52C	com.apple.iokit.IONetworkingFamily:__text	FFFFFFF00615A52C
sub_FFFFFFF00615A53C	com.apple.iokit.IONetworkingFamily:__text	FFFFFFF00615A53C
sub_FFFFFFF00615A574	com.apple.iokit.IONetworkingFamily:__text	FFFFFFF00615A574
sub_FFFFFFF00615A678	com.apple.iokit.IONetworkingFamily:__text	FFFFFFF00615A678
sub_FFFFFFF00615A730	com.apple.iokit.IONetworkingFamily:__text	FFFFFFF00615A730
sub_FFFFFFF00615A7E8	com.apple.iokit.IONetworkingFamily:__text	FFFFFFF00615A7E8
sub_FFFFFFF00615A820	com.apple.iokit.IONetworkingFamily:__text	FFFFFFF00615A820
sub_FFFFFFF00615A858	com.apple.iokit.IONetworkingFamily:__text	FFFFFFF00615A858
sub_FFFFFFF00615AB20	com.apple.iokit.IONetworkingFamily:__text	FFFFFFF00615AB20
sub_FFFFFFF00615AC00	com.apple.iokit.IONetworkingFamily:__text	FFFFFFF00615AC00
sub_FFFFFFF00615AC0C	com.apple.iokit.IONetworkingFamily:__text	FFFFFFF00615AC0C
sub_FFFFFFF00615AC34	com.apple.iokit.IONetworkingFamily:__text	FFFFFFF00615AC34
sub_FFFFFFF00615AC3C	com.apple.iokit.IONetworkingFamily:__text	FFFFFFF00615AC3C
sub_FFFFFFF00615AC44	com.apple.iokit.IONetworkingFamily:__text	FFFFFFF00615AC44

Recover function names (only for iOS)

- In drivers, most classes inherit from classes in the kernel, e.g., IOService, OSObject. These kernel classes still keep symbols
- In C++, when a child class inherits from a parent class and overrides a virtual function, the parent's function and child's function have same name and same offset

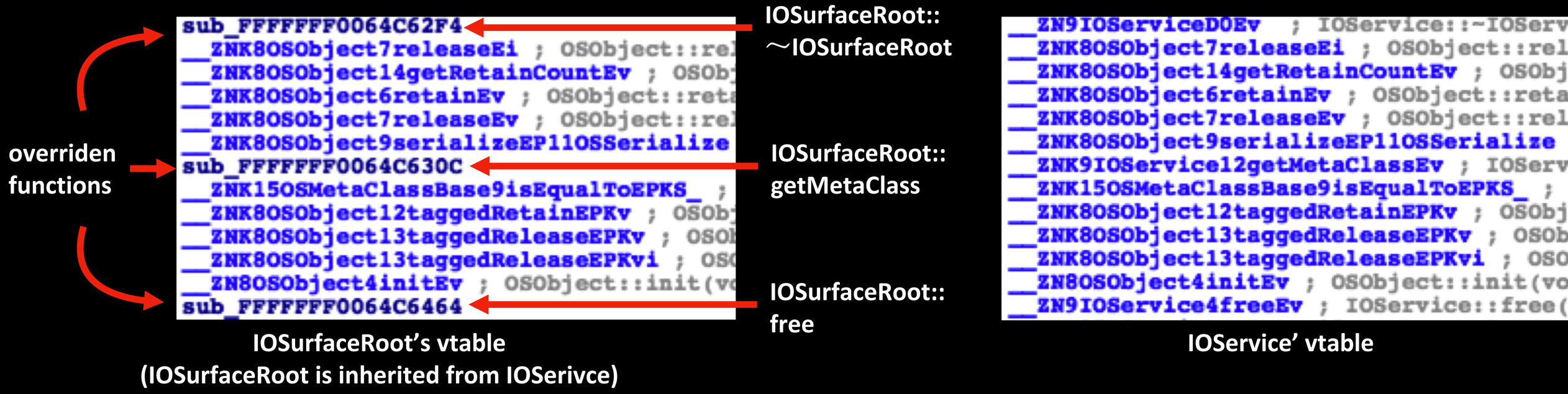
```
class A {
  public:
  virtual void foo();
}
```

```
class B: public A {
  public:
  virtual void foo();
}
```



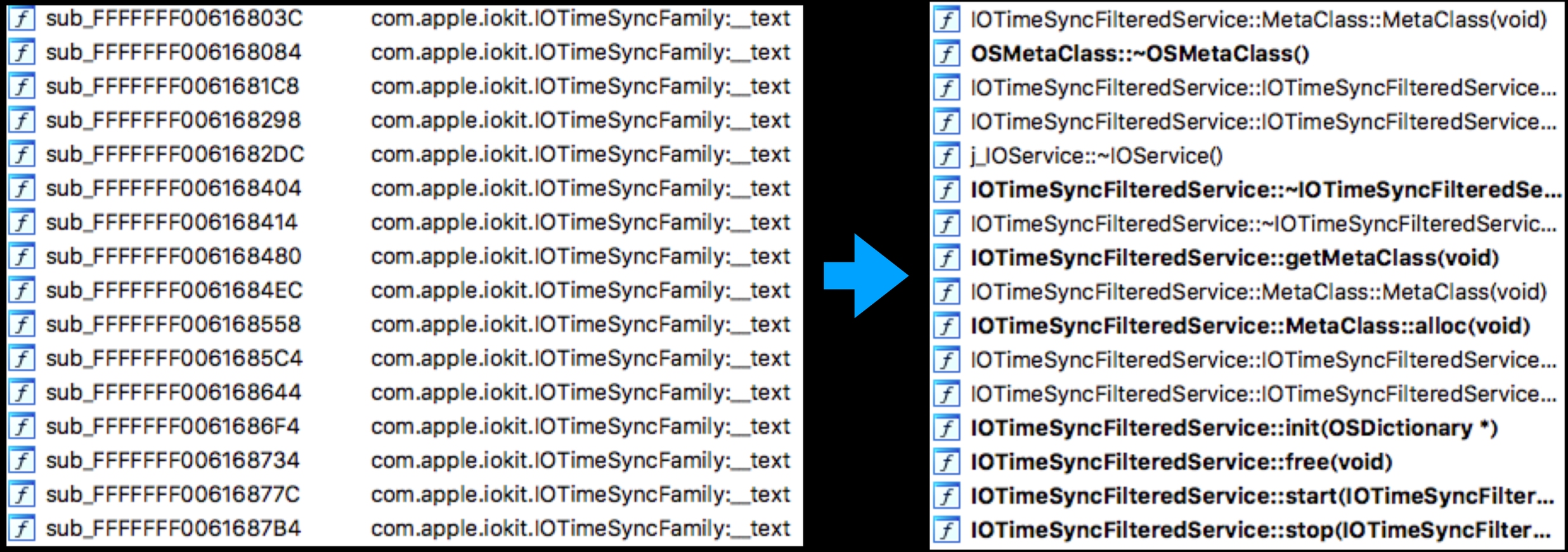
Recover function names (only for iOS)

- Recover function names in classes inherited from kernel classes
- Not a complete solution, but cover plenty of critical functions



Recover function names (only for iOS)

- Many function names can be recovered in this way



Original **Now**

Resolve variable types (in the decompiled code)

- For local, global and member variables
- Method: identify local variables' types in a function and perform type propagation along control flow
- Identify local variables' types:
  - Identify parameter variables' types
  - Identify the creation and initialization of stack variables

Resolve variable types (in the decompiled code)

- Identify parameter variables' types :
  - In the binary, C++ function names are encoded with the function's class name and parameter types

__ZN23IOSurfaceRootUserClient4initEP13IOSurfaceRootP4taskP12OSDictionary



Decode (Demangle)

IOSurfaceRootUserClient::init (IOSurfaceRoot *, task *, OSDictionary *)



Parameter Variable Types

Resolve variable types (in the decompiled code)

- Identify the creation and initialization of stack variables :
  - Many variable allocation, initialization, or type casting functions indicate the variables' types, e.g.,

Allocation Func: OSMetaClass::allocClassWithName (const char *name)

Allocatoin Func: IOSurfaceRootUserClient::MetaClass::alloc ( )

Init Func: IOCommandGate::IOCommandGate (IOCommandGate *)

Type Cast: OSMetaClassBase::safeMetaCast (OSMetaClassBase *, OSMetaClass *)

Resolve variable types (in the decompiled code)

- After variable types are identified, we can perform type propagation along function's control flow

```
__int64 __cdecl IOAccelDevice2::start(IOAccelDevice2 *this, IOService *a2)
{
  IOGraphicsAccelerator2 *v2; // rax@2
  IOGraphicsAccelerator2 *v3; // rbx@2
  IOGraphicsAccelerator2 *v4; // rbx@3
  __int64 result; // rax@3

  if ( (unsigned __int8)((int (*)(void))`vtable for'IOUserClient->__ZN9IOService4openEPS_jPv)()
    && (v2 = (IOGraphicsAccelerator2 *)OSMetaClassBase::safeMetaCast(
                                         (OSMetaClassBase *)a2,
                                         (const OSMetaClassBase *)IOGraphicsAccelerator2::metaClass),
        v3 = v2,
        (this->member33_IOGraphicsAccelerator2 = v2) != 0LL) )
```

v2's type is identified

Propagate v2's type and modify member variable's type

Add cross references for member variable and vfunc:

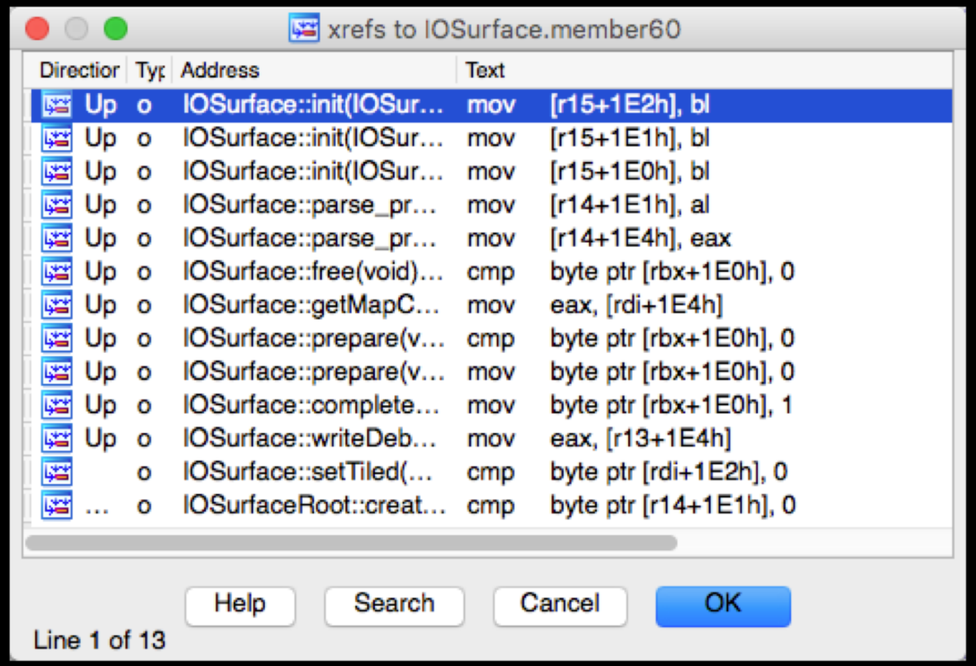
- Examine every sentence in the decompiled code of all functions, check whether member variable or virtual function is used, e.g.

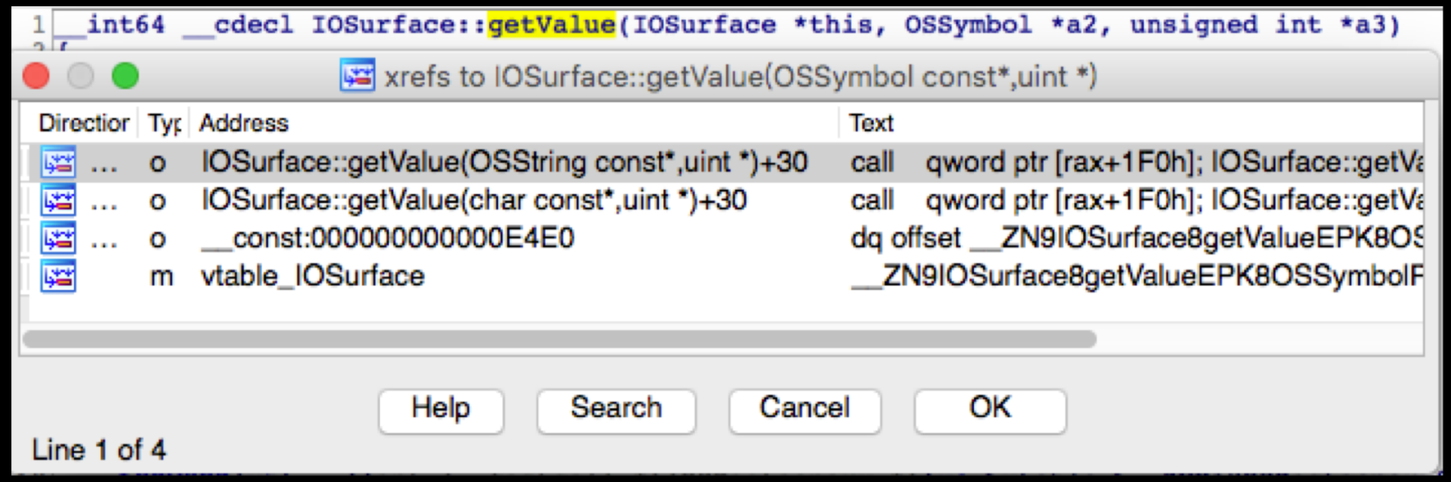
```
__int64 __cdecl IOSurface::setTiled(IOSurface *this, bool a2)
{
  __int64 result; // rax@1

  result = 3758097095LL;
  if ( a2 )
  {
    if ( BYTE2(this->member60) )
    {
      LOBYTE(this->member59) = a2;
      result = 0LL;
    }
  }
  return result;
}
```

```
__int64 __cdecl IOSurface::getValue(IOSurface *this, char *a2, unsigned int *a3)
{
  unsigned int *v3; // r14@1
  OSSymbol *v4; // rbx@2
  __int64 v5; // r14@4

  v3 = a3;
  if ( a2 )
    v4 = OSSymbol::withCString(a2);
  else
    v4 = 0LL;
  v5 = this->vtable->__ZN9IOSurface8getValueEPK8OSSymbolPj(this, v4, v3);
  if ( v4 )
    v4->vtable->__ZNK8OSObject7releaseEv((OSObject *)v4);
  return v5;
}
```

Add cross references for member variable and vfunc:

- Add cross references from the usage in code to their location in class or vtable structures





Now, driver's decompiled code in IDA pro looks like

- Looks more like source code, right?

```
__int64 __cdecl IOSurface::getValue(IOSurface *this, char *a2, unsigned int *a3)
{
  unsigned int *v3; // r14@1
  OSSymbol *v4; // rbx@2
  __int64 v5; // r14@4

  v3 = a3;
  if ( a2 )
    v4 = OSSymbol::withCString(a2);
  else
    v4 = 0LL;
  v5 = this->vtable->__ZN9IOSurface8getValueEPK8OSSymbolPj(this, v4, v3);
  if ( v4 )
    v4->vtable->__ZNK8OSObject7releaseEv((OSObject *)v4);
  return v5;
}
```

UI support for better RE experience

- Extend UI operations that can be performed in IDA pro’s decompiled code, e.g. jump to virtual function’s implementation by just a double click
- Method:
  - Register action to double-click events
  - Register action to key events
  - Register action to name change events
  - Register action to type change events

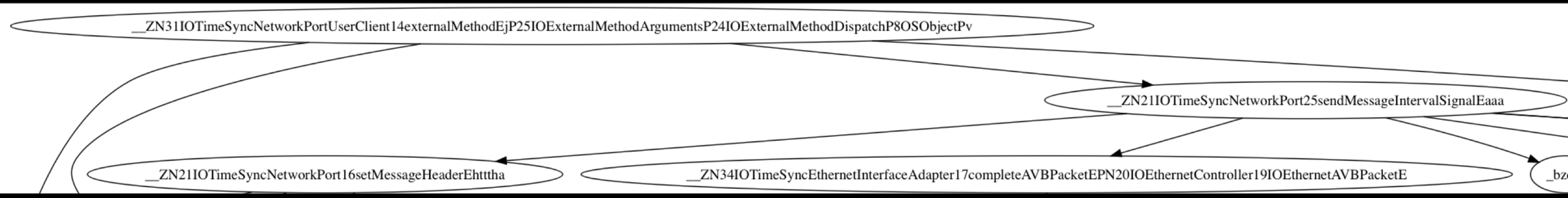
UI support for better RE experience

IDA Vie... | Pseudocod... | Structu... | Names win... | Strings win... | Pro

```
__int64 __fastcall IOMobileFramebufferUserClient::sMethod56(IOMobileFramebufferUserClient *target
{
  char *v3; // x8
  uint64_t v4; // x1

  if ( arguments->structureInputSize != 136 )
    return 3758097090LL;
  v3 = (char *)arguments->structureInput;
  if ( *v3 )
    v4 = 0LL;
  else
    v4 = (uint64_t)(v3 + 8);
  return target->mProvider->vtable->IOMobileFramebuffer::virtualFunc251_ImpByChild(
           target->mProvider,
           v4,
           *((unsigned int *)v3 + 32),
           *((unsigned int *)v3 + 33));
}
```

Generate call graph

- For further inter-procedural analysis, call graph is the prerequisite
- After variable types are resolved, targets of virtual function calls can be recognized
- Then we can build call graph for all functions



Now, everything is ready, you can do your own manual analysis and static analysis

```
__int64 __cdecl IOSurface::getValue(IOSurface *this, OSSymbol *a2, unsigned int *a3)
{
  unsigned int *v3; // rbx@1
  OSDictionary *v4; // rbx@1
  OSDictionary *v5; // rax@4

  v3 = a3;
  IORecursiveLockLock(this->member16_IORecursiveLock);
  *v3 = *(_DWORD *)(this->member24 + 24);
  v4 = this->member26_OSDictionary;
  if ( !v4 )
  {
    IOSurface::init_values(this);
    v4 = this->member26_OSDictionary;
  }
  if ( a2 )
  {
    LODWORD(v5) = ((int (__fastcall *)(OSDictionary *))v4->vtable->__ZNK12OSDictionary9getObjectEPK8OSSymbol)(v4);
    v4 = v5;
  }
  IORecursiveLockUnlock(this->member16_IORecursiveLock);
  return (__int64)v4;
}
```

Ryuk-Fuzz: combine fuzzing with static analysis

One use case of Ryuk: Ryuk-Fuzz

- Idea: Guide driver fuzzing with static analysis results
- Implementation: Integrate Ryuk with the state-of-art Apple kernel driver fuzzer, PassiveFuzzFrameworkOSX
- Method:
  - Step 1: Perform data flow analysis in Ryuk to infer drivers' required user input formats
  - Step 2: During fuzzing, use the inferred input formats to guide input generation and improve fuzzing efficiency

Ryuk-Fuzz Step 1: Static data flow analysis in the decompiled code to infer user input formats

- identify where the user input is checked, what parts are checked, and what is the condition

```
v7 = userInputBuffer;
_this = this;
v9 = -536870206;
if ( a4 < 0x50 )
  return (unsigned int)v9;
v10 = this->member31_IOGraphicsAccelerator2;
if ( *a5 < (unsigned __int64)LODWORD(v10->member410) + 72 )
  return (unsigned int)v9;
v11 = *((_BYTE *)userInputBuffer + 24);
if ( v11 != 6 && v11 != 1 )
  return (unsigned int)v9;
v12 = *(_DWORD *)userInputBuffer;
if ( !(*(_DWORD *)userInputBuffer & 0xF) )
{
  v60 = a3;
  v19 = *((_DWORD *)userInputBuffer + 17);
```

Condition: offset 24, size: 1
should not be equal to 6 or 1

Condition: offset 0, size: 4
last 4 bits should not
be equal to 0

Ryuk-Fuzz Step 2: Guide fuzzing

- PassiveFuzzFrameworkOSX only works on macOS 10.11 now. For later versions, it has several errors
- Error 1: Wrong buffer size for reading the kernel header (affect macOS 10.12 and later)

5.1 MacOS 10.12 MachO format parse error

process_kernel_mach_header(void *kernel_header, struct kernel_info *kinfo)

could not analyze the machO file format in 10.12 which may be different with 10.11 or before so as to cause kext load fail.

Error in PassiveFuzzFrameworkOSX

Ryuk-Fuzz Step 2: Guide fuzzing

- Error 1: Wrong buffer size for reading the kernel header (affect macOS 10.12 and later)

```
error = vnode_lookup(MACH_KERNEL, 0, &kernel_vnode, vfs);
if (error)
{
    return KERN_FAILURE;
}
void *kernel_header = _MALLOC(PAGE_SIZE_64, M_TEMP, M_ZERO);
if (kernel_header == NULL) return KERN_FAILURE;

// read and process kernel header from filesystem
error = get_kernel_mach_header(kernel_header, kernel_vnode, vfs);
if (error) goto failure;
error = process_kernel_mach_header(kernel_header, kinfo);
if (error) goto failure;
```

PassiveFuzzFrameworkOSX assumes kernel header is smaller than a page

Ryuk-Fuzz Step 2: Guide fuzzing

- Cause of Error 1: Wrong buffer size for reading the kernel header (affect macOS 10.12 and later)

Before 10.12
Header size is smaller than 1 page

Offset	Data	Description	Value
00000E30	00000019	Command	LC_SEGMENT_64
00000E34	00000048	Command Size	72
00000E38	5F5F4C494E4B45444954000…	Segment Name	__LINKEDIT

After 10.12
Header size is larger than 1 page

Offset	Data	Description	Value
00001128	00000019	Command	LC_SEGMENT_64
0000112C	00000048	Command Size	72
00001130	5F5F4C494E4B45444954000…	Segment Name	__LINKEDIT

Ryuk-Fuzz Step 2: Guide fuzzing

- Solution to Error 1: enlarge the allocation size

```
error = vnode_lookup(MACH_KERNEL, 0, &kernel_vnode, vfs);
if (error)
{
    return KERN_FAILURE;
}
void *kernel_header = _MALLOC(PAGE_SIZE_64, M_TEMP, M_ZERO);
if (kernel_header == NULL) return KERN_FAILURE;

// read and process kernel header from filesystem
error = get_kernel_mach_header(kernel_header, kernel_vnode, vfs);
if (error) goto failure;
error = process_kernel_mach_header(kernel_header, kinfo);
if (error) goto failure;
```

2 * PAGE_SIZE_64

Ryuk-Fuzz Step 2: Guide fuzzing

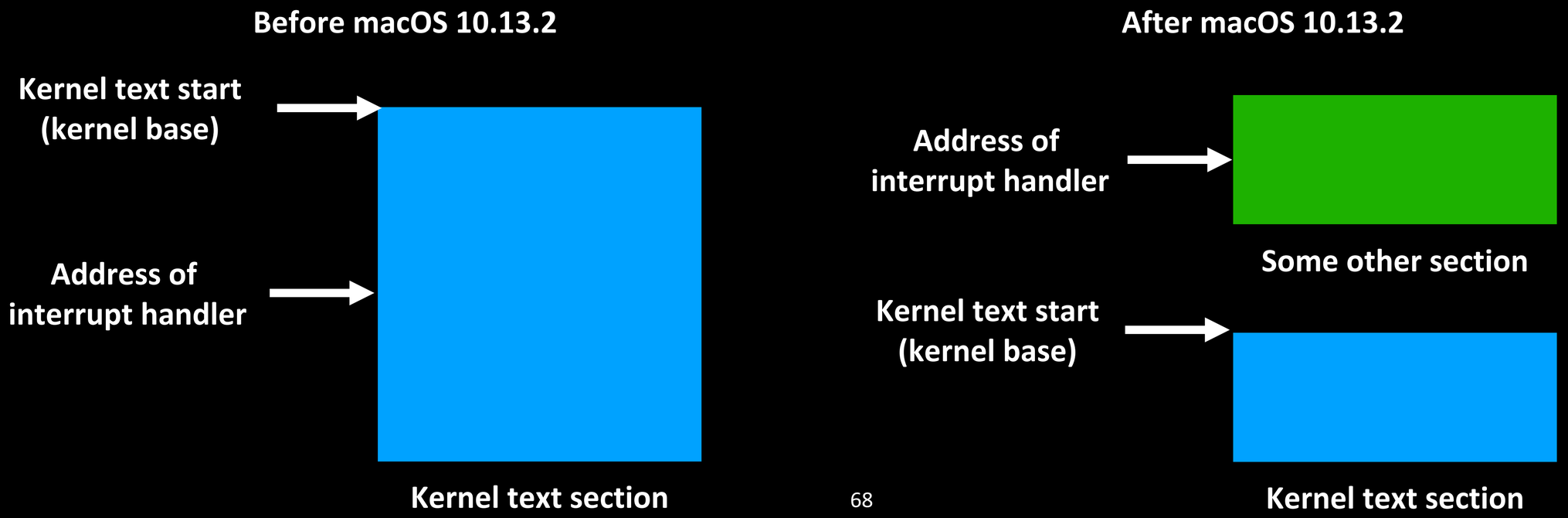
- Error still exists in PassiveFuzzFrameworkOSX
- Error 2: Wrong way to infer kernel base at runtime (affect 10.13.2 and after)

```
static void
get_running_text_address(struct kernel_info *kinfo)
{
    // retrieves the address of the IDT
    mach_vm_address_t idt_address = 0;
    get_addr_idt(&idt_address);
    // calculate the address of the int80 handler
    mach_vm_address_t int80_address = calculate_int80address(idt_address);
    // search backwards for the kernel base address (mach-o header)
    mach_vm_address_t kernel_base = find_kernel_base(int80_address);
```

PassiveFuzzFrameworkOSX calcuate the address of interrupt handler and search backwards for mach-o header (i.e. the kernel base address)

Ryuk-Fuzz Step 2: Guide fuzzing

- Cause of Error 2: In current macOS, int80 handler is not in kernel text section now. As a result, we can not find the kernel base by searching backwards from the interrupt handler



Ryuk-Fuzz Step 2: Guide fuzzing

- Why the interrupt handler is not in kernel text now?
  - The side effect of Apple's patch to mitigate Meltdown vulnerability
  - Apple splits user and kernel space, in which interrupt handler entries are not in kernel text space now
  - As a result, kernel text base can not be inferred by the interrupt handler



Ryuk-Fuzz Step 2: Guide fuzzing

- Solution to Error 2: Search backwards from the address of some other address, e.g. address of _lck_mtx_lock

```
static void
get_running_text_address(struct kernel_info *kinfo)
{
    // retrieves the address of the IDT
    mach_vm_address_t idt_address = 0;
    get_addr_idt(&idt_address);
    // calculate the address of the int80 handler
    mach_vm_address_t int80_address = [redacted];
    // search backwards for the kernel base address (mach-o header)
    mach_vm_address_t kernel_base = find_kernel_base(int80_address);
```

Get the address of _lck_mtx_lock

Conclusions

Vulnerabilities that can be exploited for privilege escalation on macOS

Technique of exploiting use-after-free vulnerability in the kernel

Ryuk: a new static analysis tool assisting manually reverse engineering and static analysis

Ryuk-Fuzz: guide fuzzing of drivers with static analysis results

Part of Ryuk has been open source now

https://github.com/bxl1989/Ryuk

Welcome to follow us on twitter and weibo

bxl1989@twitter, bxl1989@weibo

SparkZheng@twitter, 蒸米spark@weibo

Q&A