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

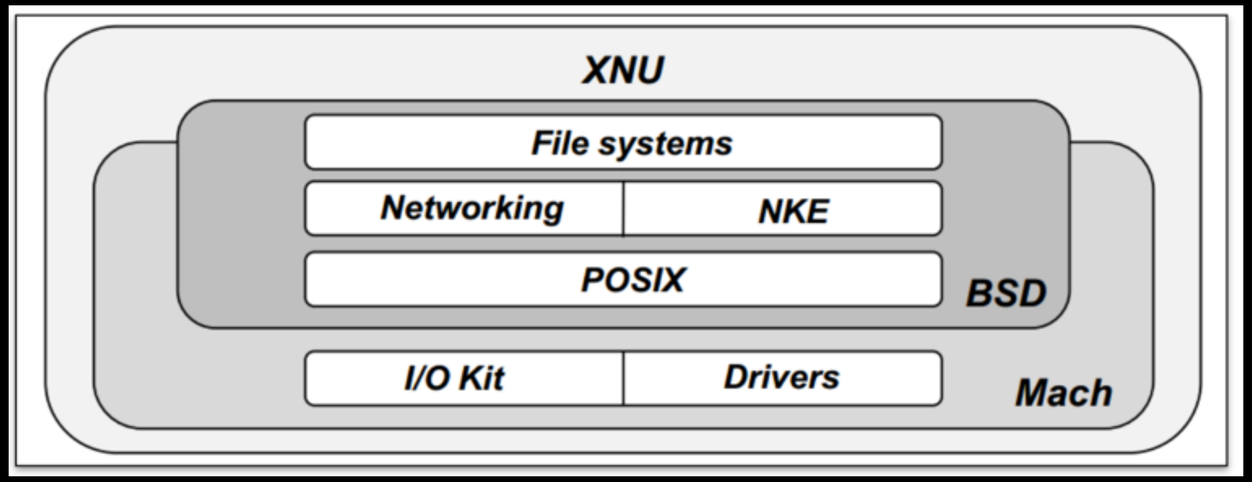
6 Conclusions

Overview

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

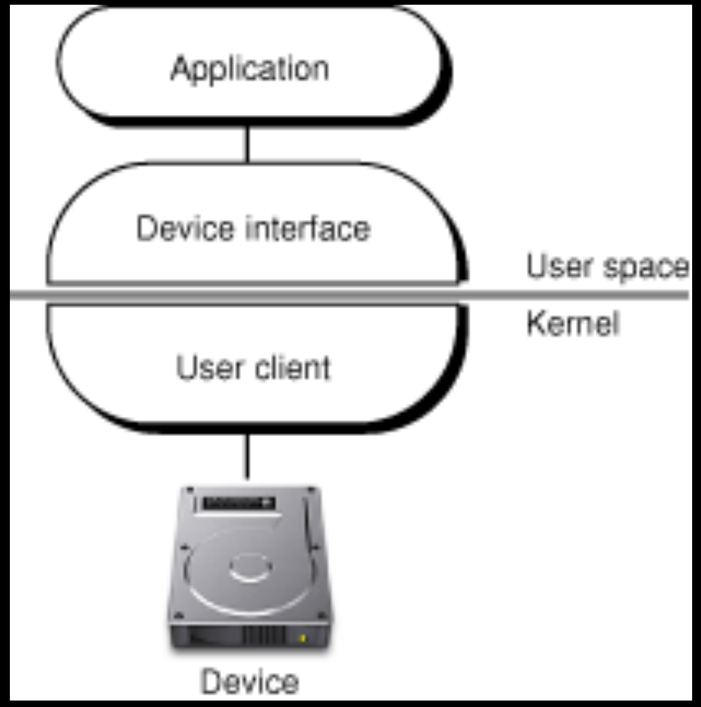
Location of driver binaries:

- 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

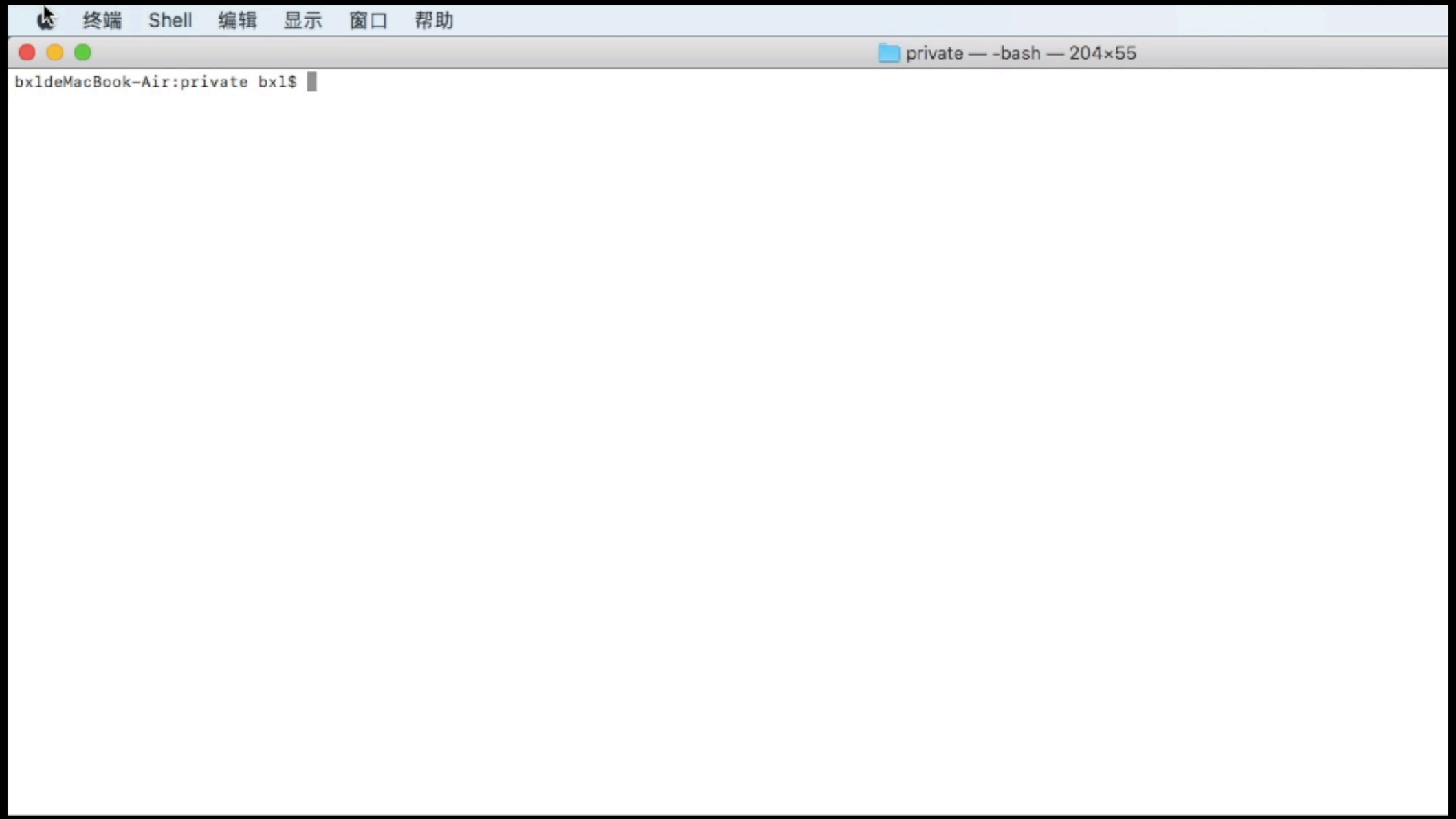
Drivers are frequently exploited in attacks against the kernel, including jailbreaks

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!

Privilege escalation on macOS 10.13.3



This privilege escalation leverages a new 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

This privilege escalation leverages a new 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);
```

This privilege escalation leverages a new 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

This privilege escalation leverages a new 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

- Creat 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 !

Besides the above vuln, we also discovered some other new vulns in Apple drivers

- CVE-2017-7119
- CVE-2018-4135

Next, we will show details of these vulns

CVE-2017-7119

- Information leakage in IOFirewireFamily
- Caused by 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

CVE-2018-4135

- Use-After-Free in IOFirewireFamily driver
- Lack of locking or serialization when using and releasing a member variable
- Can be exploited to control PC in kernel

CVE-2018-4135

- Use-After-Free in IOFirewireFamily driver
- Lack of locking or serialization when using and releasing a member variable
- In IOFireWireUserClient::externalMethod

```
case kCommand_Submit:
{
    IOFireWireUserClient * fw_uc = OSDynamicCast( IOFireWireUserClient, targetObject );
    if( fw_uc )
    {
        result = fw_uc->userAsyncCommand_Submit(arguments->asyncReference,
                                                (CommandSubmitParams*)arguments->structureInput,
                                                (CommandSubmitResult*)arguments->structureOutput,
                                                (IOByteCount) arguments->structureInputSize,
                                                (IOByteCount*) &arguments->structureOutputSize);
    }
    else
    {
        result = kIOReturnBadArgument;
    }
    break;
}
```

CVE-2018-4135

- IOFireWireUserClient::userAsyncCommand_Submit looks up for a IOFWUserReadCommand and calls its “submit” method

```
IOReturn
IOFireWireUserClient::userAsyncCommand_Submit(
    OSAsyncReference64          asyncRef,
    CommandSubmitParams *       params,
    CommandSubmitResult *       outResult,
    IOByteCount                 paramsSize,
    IOByteCount *               outResultSize) {
    IOFWUserCommand * cmd = NULL ;
    ...
    if ( params->kernCommandRef ) {
        const OSObject * object = fExporter->lookupObject( params->kernCommandRef ) ;
        ...
    }
    ...
    if ( cmd ) {
        ...
            error = cmd->submit( params, outResult ) ;
        ...
    }

}
```

CVE-2018-4135

- In IOFWUserReadCommand::submit

```
IOReturn
IOFWUserReadCommand::submit(
    CommandSubmitParams*    params,
    CommandSubmitResult*    outResult)
{
    IOReturn    error       = kIOReturnSuccess ;
    Boolean     syncFlag    = ( params->flags & kFWCommandInterfaceSyncExecute ) != 0 ;
    Boolean     copyFlag    = ( params->flags & kFireWireCommandUseCopy ) != 0;
    Boolean     absFlag     = ( params->flags & kFireWireCommandAbsolute ) != 0 ;
    bool        forceBlockFlag  = (params->flags & kFWCommandInterfaceForceBlockRequest) != 0;

    if ( params->staleFlags & kFireWireCommandStale_Buffer )
    {
        if ( fMem ) // whatever happens, we're going to need a new memory descriptor
        {
            fMem->complete() ;
            fMem->release() ;   <-- (a)
            fMem = NULL;
        }
        ...

    }

    if ( not error )
    {
        ...
            fCommand = fUserClient->getOwner()->createReadCommand( target_address,
                fMem, syncFlag ? NULL : & IOFWUserCommand::asyncReadWriteCommandCompletion,
                this, params->newFailOnReset ) ;      <-- (b)
        ...
    }
    ...
}
```

CVE-2018-4135

- (a) release fMem, (b) uses fMem (fMem is a member)

```
IOReturn
IOFWUserReadCommand::submit(
    CommandSubmitParams*    params,
    CommandSubmitResult*    outResult)
{
    IOReturn    error       = kIOReturnSuccess ;
    Boolean     syncFlag    = ( params->flags & kFWCommandInterfaceSyncExecute ) != 0 ;
    Boolean     copyFlag    = ( params->flags & kFireWireCommandUseCopy ) != 0;
    Boolean     absFlag     = ( params->flags & kFireWireCommandAbsolute ) != 0 ;
    bool        forceBlockFlag  = (params->flags & kFWCommandInterfaceForceBlockRequest) != 0;

    if ( params->staleFlags & kFireWireCommandStale_Buffer )
    {
        if ( fMem ) // whatever happens, we're going to need a new memory descriptor
        {
            fMem->complete() ;
            fMem->release() ;   <-- (a)
            fMem = NULL;
        }
        ...

    }

    if ( not error )
    {
        ...
            fCommand = fUserClient->getOwner()->createReadCommand( target_address,
                fMem, syncFlag ? NULL : & IOFWUserCommand::asyncReadWriteCommandCompletion,
                this, params->newFailOnReset ) ;      <-- (b)
        ...
    }
    ...
}
```

CVE-2018-4135

Exploit:

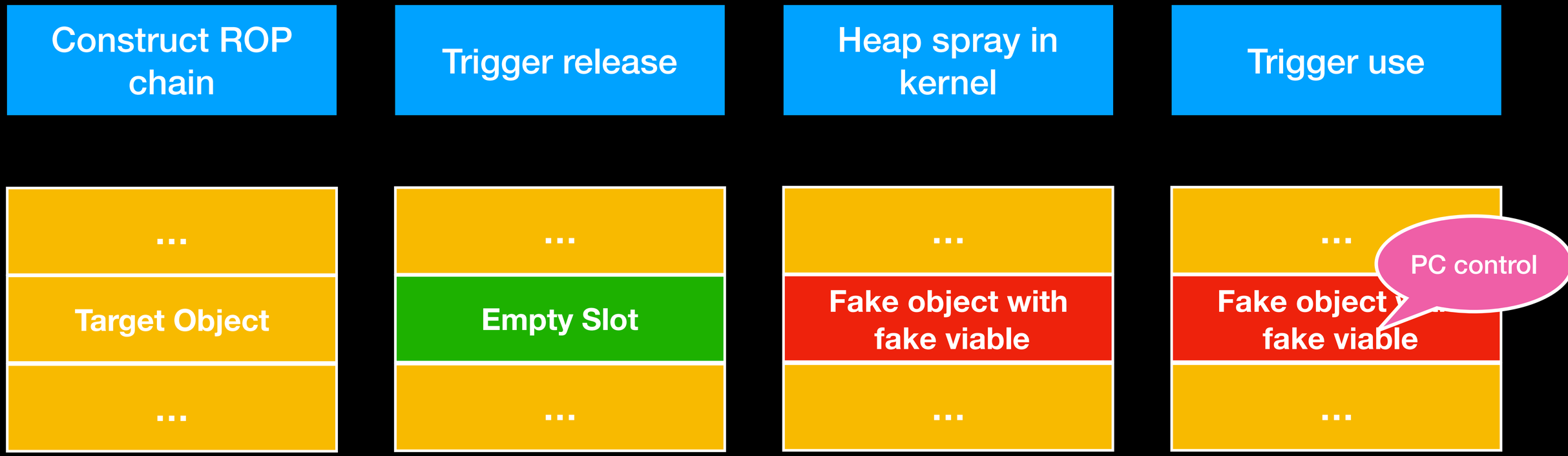
- Create two threads to invoke method on the same userclient
- One thread release fMem, the other uses it

```
    0xffffff7f94c8be50 <+160>: testq  %r13, %r13
    0xffffff7f94c8be53 <+163>: je     0xffffff7f94c8be68
    0xffffff7f94c8be55 <+165>: movq   (%r13), %rax
    0xffffff7f94c8be59 <+169>: movq   %r13, %rdi
->  0xffffff7f94c8be5c <+172>: callq  *0x1c8(%rax)
```

```
(lldb) re r
General Purpose Registers:
        rax = 0x4141414141414141
```

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

Basic flow:



A new heap spray method on macOS

- Utilize OSUnserializeXML
- Set properties of a device

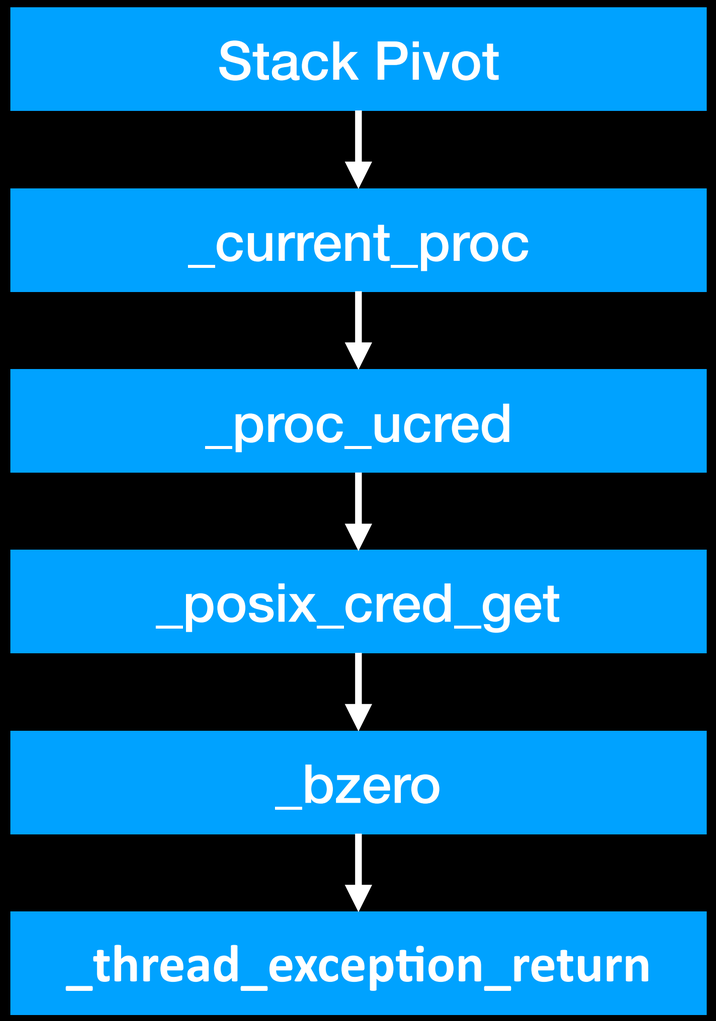
```
/* Routine io_registry_entry_set_properties */
kern_return_t is_io_registry_entry_set_properties(
    io_object_t registry_entry,
    io_buf_ptr_t properties,
    mach_msg_type_number_t propertiesCnt,
        kern_return_t * result) {
        ...
        obj = OSUnserializeXML( (const char *) data,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

A new 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

- Previous Methods (Unavailable Now)

In tpwn (on 10.10)

```
50                      push rax
0100                    add DWORD PTR [rax],eax
005b41                  add BYTE PTR [rbx+0x41],bl
5c                      pop rsp
415e                    pop r14
415f                    pop r15
5d                      pop rbp
c3                      ret
```

In rootsh (on 10.11)

```
static const uint8_t xchg_esp_eax_pop_rsp_ins[] = {
        0x94,   /* xchg esp, eax */
        0x5c,   /* pop rsp       */
        0xc3,   /* ret           */
};
```

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

- New Method

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

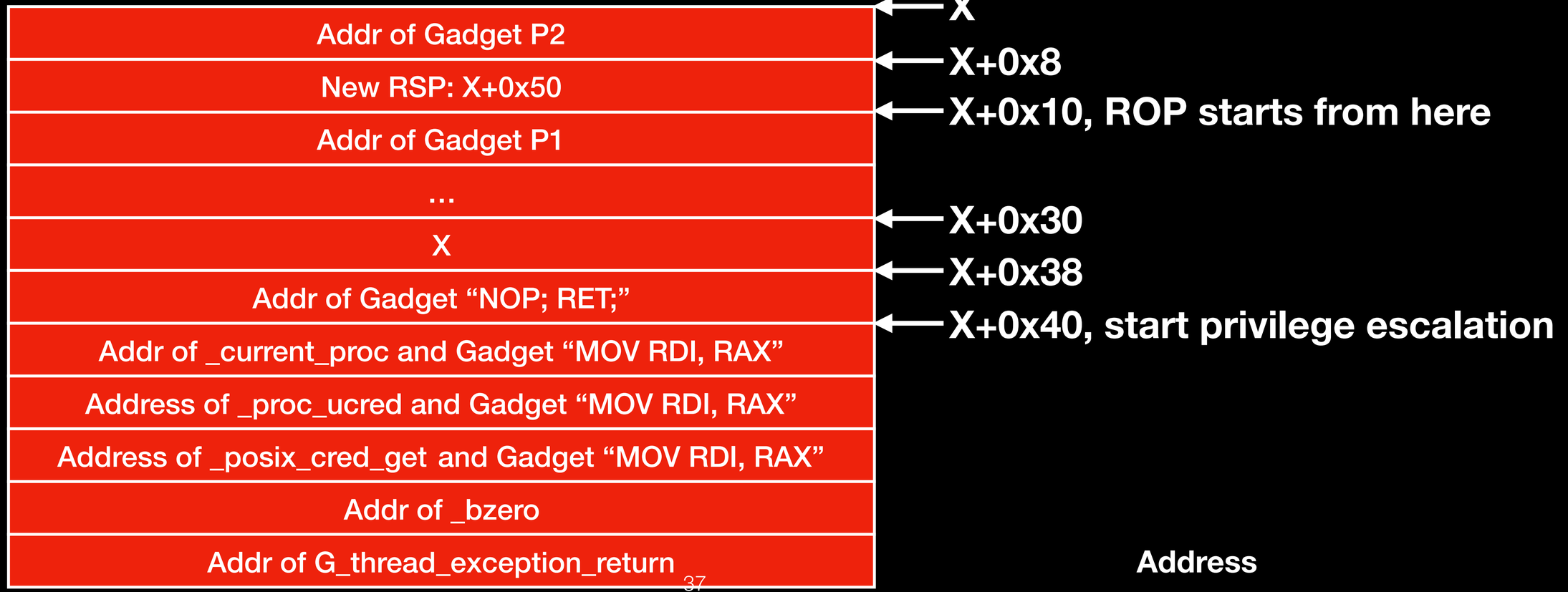
```
mov     rcx, [rax+30h]
mov     [rbp+var_50], rcx
call    qword ptr [rax]
```

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#
```

Obstacles in analyzing Apple drivers

Analyzing macOS and iOS kernel drivers is not easy!

- Drivers are closed-source: binaries are lack of high-level semantics and variable types are lost
- Drivers are programmed in C++: virtual functions are widely used but object types are unknown —> the real function called is unknown
- Symbols are lost in iOS drivers: iOS kernelcache strips all symbols in drivers

What does a driver's binary look like in IDA pro?

- macOS driver: some symbols are kept

```
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;
}
```

What does a driver's binary look like in IDA pro?

- macOS driver: But! Wrong parameter inference, lack of variable types, unrecognizable virtual function calls

```
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;
}
```

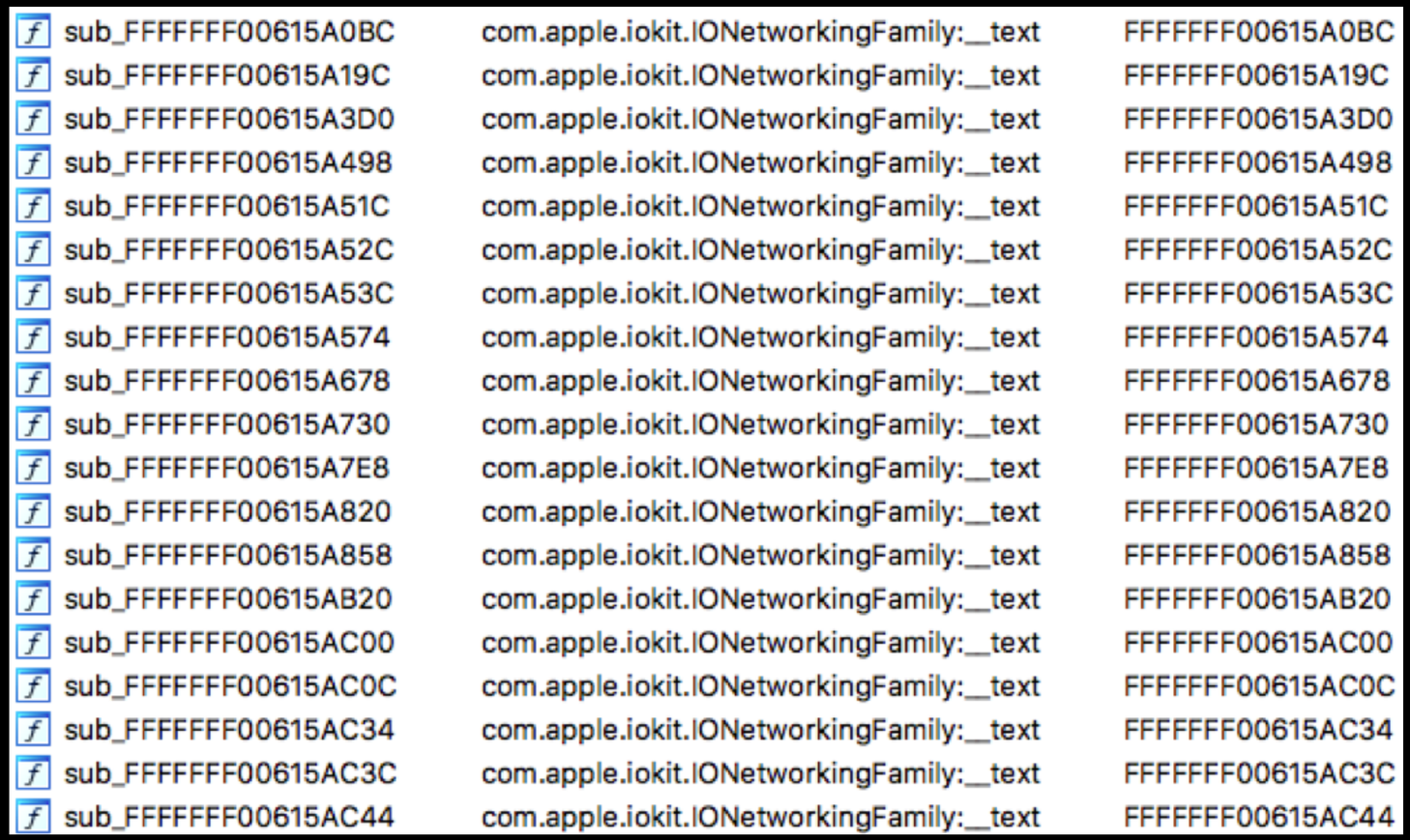
Wrong parameter inference

Variable types are lost

Virtual function calls are unrecognizable

What does a driver’s binary look like in IDA pro?

- 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

What does a driver’s binary look like in IDA pro?

- iOS driver: data structure is gone, e.g. this figure should be the location of a virtual table

```
com.apple.iokit.IONetworkingFamily:__const:FFFFFFF006E047A8                     DCQ unk_FFFFFFF0076DC0C8
com.apple.iokit.IONetworkingFamily:__const:FFFFFFF006E047B0                     DCQ unk_FFFFFFF0076DC248
com.apple.iokit.IONetworkingFamily:__const:FFFFFFF006E047B8 unk_FFFFFFF006E047B8 DCB     0
com.apple.iokit.IONetworkingFamily:__const:FFFFFFF006E047B8
com.apple.iokit.IONetworkingFamily:__const:FFFFFFF006E047B9                     DCB     0
com.apple.iokit.IONetworkingFamily:__const:FFFFFFF006E047BA                     DCB     0
com.apple.iokit.IONetworkingFamily:__const:FFFFFFF006E047BB                     DCB     0
com.apple.iokit.IONetworkingFamily:__const:FFFFFFF006E047BC                     DCB     0
com.apple.iokit.IONetworkingFamily:__const:FFFFFFF006E047BD                     DCB     0
com.apple.iokit.IONetworkingFamily:__const:FFFFFFF006E047BE                     DCB     0
com.apple.iokit.IONetworkingFamily:__const:FFFFFFF006E047BF                     DCB     0
com.apple.iokit.IONetworkingFamily:__const:FFFFFFF006E047C0                     DCB     0
com.apple.iokit.IONetworkingFamily:__const:FFFFFFF006E047C1                     DCB     0
com.apple.iokit.IONetworkingFamily:__const:FFFFFFF006E047C2                     DCB     0
com.apple.iokit.IONetworkingFamily:__const:FFFFFFF006E047C3                     DCB     0
com.apple.iokit.IONetworkingFamily:__const:FFFFFFF006E047C4                     DCB     0
com.apple.iokit.IONetworkingFamily:__const:FFFFFFF006E047C5                     DCB     0
com.apple.iokit.IONetworkingFamily:__const:FFFFFFF006E047C6                     DCB     0
com.apple.iokit.IONetworkingFamily:__const:FFFFFFF006E047C7                     DCB     0
com.apple.iokit.IONetworkingFamily:__const:FFFFFFF006E047C8                     DCB  0x64 ; d
com.apple.iokit.IONetworkingFamily:__const:FFFFFFF006E047C9                     DCB  0x40 ; @
com.apple.iokit.IONetworkingFamily:__const:FFFFFFF006E047CA                     DCB  0x15
com.apple.iokit.IONetworkingFamily:__const:FFFFFFF006E047CB                     DCB     6
com.apple.iokit.IONetworkingFamily:__const:FFFFFFF006E047CC                     DCB  0xF0
com.apple.iokit.IONetworkingFamily:__const:FFFFFFF006E047CD                     DCB  0xFF
com.apple.iokit.IONetworkingFamily:__const:FFFFFFF006E047CE                     DCB  0xFF
com.apple.iokit.IONetworkingFamily:__const:FFFFFFF006E047CF                     DCB  0xFF
com.apple.iokit.IONetworkingFamily:__const:FFFFFFF006E047D0                     DCB  0x68 ; h
com.apple.iokit.IONetworkingFamily:__const:FFFFFFF006E047D1                     DCB  0x40 ; @
```

What does a driver's binary look like in IDA pro?

- iOS driver: meaningless function names, totally lost variable types, unrecognized virtual function calls

```
__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);
```

Wrong decompile result everywhere !

What exactly do we want for a binary to be?

- Function (including virtual func) calls are recognizable, the called functions are known
- Missing symbols are recovered, decompiled code is understandable…

Why?

- Knowing the real call target is a prerequisite for inter-procedure analysis and futher analysis
- Manual review needs meaningful decompiled code

What exactly do we want for a binary to be?

- Just like we have the 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 solving uncertainties in Apple drivers, for further analyzing drivers' security

- Implemented as an IDA pro python script

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

Ryuk’s features (functionalities):

- Class recognition: identify classes’ information
- VTable recognition: identify classes’ virtual functions
- Recover function names: only for iOS drivers
- Resolve variable types: resolve and mark variable types
- Add cross references: for members and virtual funcs
- UI support: better UI support for manual review
- Call graph generation: for inter-procedure analysis

Class recognition

- Purpose: for variable type inference and further analysis
- Method: depend on RTTI information left in binaries
- Apple drivers are programed in C++ and support a limited RTTI feature
- __mod_init_func: a section containing initialization functions to support RTTI
- Analyzing functions in __mod_init_func section, we can get information of classes (name, size…)

Class recognition

- __mod_init_func

macOS

```
__mod_init_func:000000000000E090 ; Segment type: Pure data
__mod_init_func:000000000000E090 ; Segment alignment 'qword' can not be represented in assembly
__mod_init_func:000000000000E090 __mod_init_func segment para public 'DATA' use64
__mod_init_func:000000000000E090                 assume cs:__mod_init_func
__mod_init_func:000000000000E090                 ;org 0E090h
__mod_init_func:000000000000E090                 dq offset __GLOBAL__sub_I_IOSurface_cpp
__mod_init_func:000000000000E098                 dq offset __GLOBAL__sub_I_IOSurfaceClient_cpp
__mod_init_func:000000000000E0A0                 dq offset __GLOBAL__sub_I_IOSurfaceDeviceCache_cpp
__mod_init_func:000000000000E0A8                 dq offset __GLOBAL__sub_I_IOSurfaceRoot_cpp
__mod_init_func:000000000000E0B0                 dq offset __GLOBAL__sub_I_IOSurfaceRootUserClient_cpp
__mod_init_func:000000000000E0B8                 dq offset __GLOBAL__sub_I_IOSurfaceSendRight_cpp
__mod_init_func:000000000000E0B8 __mod_init_func ends
```

iOS

```
com.apple.iokit.IOSurface:__mod_init_func:FFFFFFF006ED75D8 ; Segment type: Pure data
com.apple.iokit.IOSurface:__mod_init_func:FFFFFFF006ED75D8                 AREA com.apple.iokit.IOSurface:__mod_init_func,
com.apple.iokit.IOSurface:__mod_init_func:FFFFFFF006ED75D8                 ; ORG 0xFFFFFFF006ED75D8
com.apple.iokit.IOSurface:__mod_init_func:FFFFFFF006ED75D8                 DCQ IOSurface_InitFunc_0
com.apple.iokit.IOSurface:__mod_init_func:FFFFFFF006ED75E0                 DCQ IOSurface_InitFunc_1
com.apple.iokit.IOSurface:__mod_init_func:FFFFFFF006ED75E8                 DCQ IOSurface_InitFunc_2
com.apple.iokit.IOSurface:__mod_init_func:FFFFFFF006ED75F0                 DCQ IOSurface_InitFunc_3
com.apple.iokit.IOSurface:__mod_init_func:FFFFFFF006ED75F8                 DCQ IOSurface_InitFunc_4
com.apple.iokit.IOSurface:__mod_init_func:FFFFFFF006ED7600                 DCQ IOSurface_InitFunc_5
com.apple.iokit.IOSurface:__mod_init_func:FFFFFFF006ED7608                 DCQ IOSurface_InitFunc_6
com.apple.iokit.IOSurface:__mod_init_func:FFFFFFF006ED7610                 DCQ IOSurface_InitFunc_7
com.apple.iokit.IOSurface:__mod_init_func:FFFFFFF006ED7618                 DCQ IOSurface_InitFunc_8
com.apple.iokit.IOSurface:__mod_init_func:FFFFFFF006ED7618 ; com.apple.iokit.IOSurface__mod_init_func ends
```

Class recognition

- Functions in __mod_init_func: register class information

macOS

```
                public __GLOBAL__sub_I_IOSurfaceRootUserClient_cpp
__GLOBAL__sub_I_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
__GLOBAL__sub_I_IOSurfaceRootUserClient_cpp endp
```

iOS

```
                EXPORT IOSurface_InitFunc_6
IOSurface_InitFunc_6                    ; DATA XREF: com.apple.iokit.IOSurface:__mod_init_func

var_s0          = 0

                STP             X29, X30, [SP,#-0x10+var_s0]!
                MOV             X29, SP
                ADRP            X0, #qword_FFFFFFF0076EBC30@PAGE
                ADD             X0, X0, #qword_FFFFFFF0076EBC30@PAGEOFF
                ADRP            X1, #aIosurfacerootu@PAGE ; "IOSurfaceRootUserClient"
                ADD             X1, X1, #aIosurfacerootu@PAGEOFF ; "IOSurfaceRootUserClient"
                ADRP            X2, #qword_FFFFFFF006ED7350@PAGE
                LDR             X2, [X2,#qword_FFFFFFF006ED7350@PAGEOFF]
                MOV             W3, #0x150
                BL              sub_FFFFFFF0064CC910
                ADRP            X8, #unk_FFFFFFF006ED8F20@PAGE
                ADD             X8, X8, #unk_FFFFFFF006ED8F20@PAGEOFF
                ADD             X8, X8, #0x10
                STR             X8, [X0]
                LDP             X29, X30, [SP+var_s0],#0x10
                RET
```

Class Name

Class Size

Class recognition

- Decompiled code of functions in __mod_init_func

macOS

```
__int64 (__fastcall **_GLOBAL__sub_I_IOSurfaceRootUserClient_cpp())(IOSurfaceR
{
  __int64 (__fastcall **result)(IOSurfaceRootUserClient::MetaClass *__hidden);

  OSMetaClass::OSMetaClass(
    &IOSurfaceRootUserClient::gMetaClass,
    "IOSurfaceRootUserClient",
    IOUserClient::gMetaClass,
    336LL);
  result = off_10110;
  IOSurfaceRootUserClient::gMetaClass = off_10110;
  return result;
}
```

iOS

```
_QWORD *IOSurface_InitFunc_6()
{
  _QWORD *result; // x0

  result = (_QWORD *)sub_FFFFFFF0064CC910(&qword_FFFFFFF0076EBC30, aIosurfacerootu, qword_FFFFFFF006ED7350, 336LL);
  *result = &unk_FFFFFFF006ED8F30;
  return result;
}
```

Class recognition

- 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]
```

VTable recognition

- In C++, virtual functions of a class are organized in a structure called VTable, which is in __const section
- In every C++ class, the first member is always a pointer to this VTable.

Source code

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

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

VTable

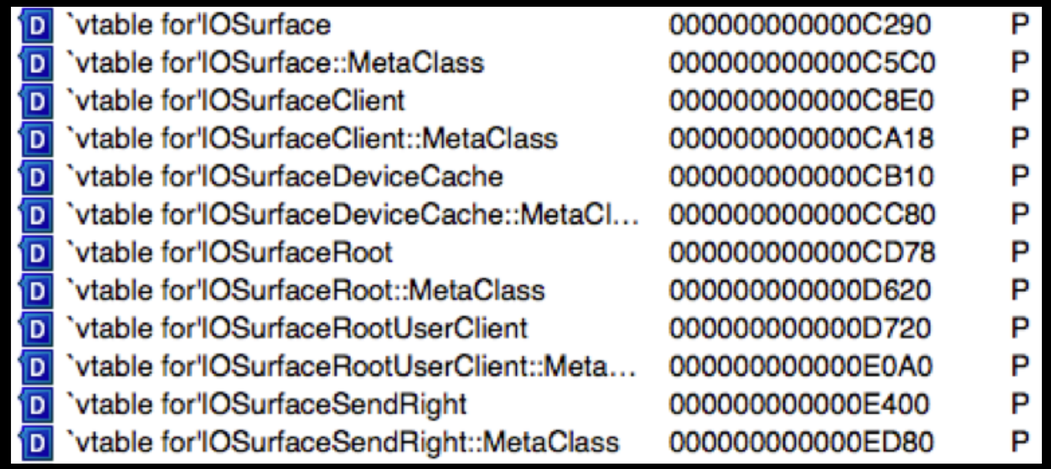
```
__const:0000000100001020                        public __ZTV1A
__const:0000000100001020 ; `vtable for'A
__const:0000000100001020 __ZTV1A                dq 0
__const:0000000100001028                        dq offset __ZTI1A
__const:0000000100001030                        dq offset __ZN1A4foo1Ev
__const:0000000100001038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

Decompiled code

```
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;
}
```

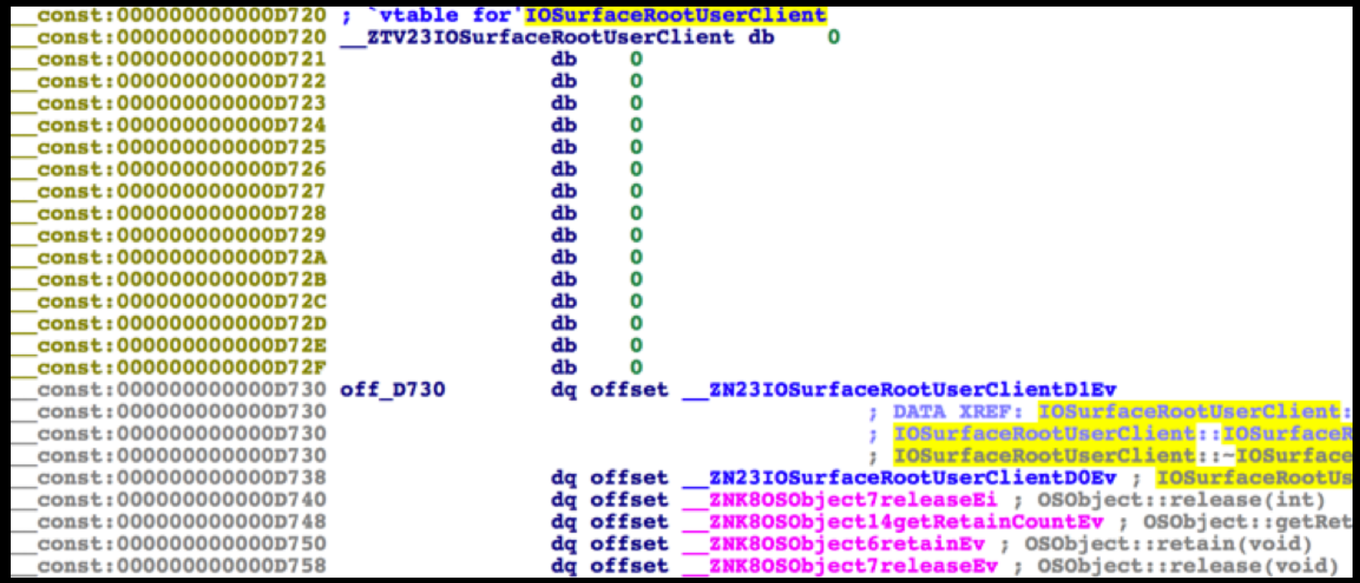
VTable recognition

- Purpose: resolve the real targets of virtual function calls
- Method: on macOS, there are symbols for VTables



D	`vtable for'IOSurface	000000000000C290 P
D	`vtable for'IOSurface::MetaClass	000000000000C5C0 P
D	`vtable for'IOSurfaceClient	000000000000C8E0 P
D	`vtable for'IOSurfaceClient::MetaClass	000000000000CA18 P
D	`vtable for'IOSurfaceDeviceCache	000000000000CB10 P
D	`vtable for'IOSurfaceDeviceCache::MetaCl…	000000000000CC80 P
D	`vtable for'IOSurfaceRoot	000000000000CD78 P
D	`vtable for'IOSurfaceRoot::MetaClass	000000000000D620 P
D	`vtable for'IOSurfaceRootUserClient	000000000000D720 P
D	`vtable for'IOSurfaceRootUserClient::Meta…	000000000000E0A0 P
D	`vtable for'IOSurfaceSendRight	000000000000E400 P
D	`vtable for'IOSurfaceSendRight::MetaClass	000000000000ED80 P

Symbols of VTables



```
__const:000000000000D720 ; `vtable for'IOSurfaceRootUserClient
__const:000000000000D720 __ZTV23IOSurfaceRootUserClient db    0
__const:000000000000D721                 db    0
__const:000000000000D722                 db    0
__const:000000000000D723                 db    0
__const:000000000000D724                 db    0
__const:000000000000D725                 db    0
__const:000000000000D726                 db    0
__const:000000000000D727                 db    0
__const:000000000000D728                 db    0
__const:000000000000D729                 db    0
__const:000000000000D72A                 db    0
__const:000000000000D72B                 db    0
__const:000000000000D72C                 db    0
__const:000000000000D72D                 db    0
__const:000000000000D72E                 db    0
__const:000000000000D72F                 db    0
__const:000000000000D730 off_D730        dq offset __ZN23IOSurfaceRootUserClientD1Ev
__const:000000000000D730                                 ; DATA XREF: IOSurfaceRootUserClient:
__const:000000000000D730                                 ; IOSurfaceRootUserClient::IOSurfaceR
__const:000000000000D730                                 ; IOSurfaceRootUserClient::~IOSurface
__const:000000000000D738                 dq offset __ZN23IOSurfaceRootUserClientD0Ev ; IOSurfaceRootUs
__const:000000000000D740                 dq offset __ZNK8OSObject7releaseEi ; OSObject::release(int)
__const:000000000000D748                 dq offset __ZNK8OSObject14getRetainCountEv ; OSObject::getRet
__const:000000000000D750                 dq offset __ZNK8OSObject6retainEv ; OSObject::retain(void)
__const:000000000000D758                 dq offset __ZNK8OSObject7releaseEv ; OSObject::release(void)
```

VTable structure

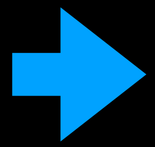
VTable recognition

- Method: on iOS, several steps depending on the vtable's specific structure
- Hint: find class's global metaclass object and further find VTable

VTable recognition

- iOS Step 1: adjust __const section in IDA pro
  - In __const, mark each 8 bytes as a pointer

```
__const:FFFFFFF006E04FCF                         DCB 0xFF
__const:FFFFFFF006E04FD0                         DCB 0xB8
__const:FFFFFFF006E04FD1                         DCB 0xC2
__const:FFFFFFF006E04FD2                         DCB 0x6D
__const:FFFFFFF006E04FD3                         DCB   7
__const:FFFFFFF006E04FD4                         DCB 0xF0
__const:FFFFFFF006E04FD5                         DCB 0xFF
__const:FFFFFFF006E04FD6                         DCB 0xFF
__const:FFFFFFF006E04FD7                         DCB 0xFF
__const:FFFFFFF006E04FD8 unk_FFFFFFF006E04FD8    DCB
__const:FFFFFFF006E04FD8
__const:FFFFFFF006E04FD9                         DCB   0
__const:FFFFFFF006E04FDA                         DCB   0
__const:FFFFFFF006E04FDB                         DCB   0
__const:FFFFFFF006E04FDC                         DCB   0
__const:FFFFFFF006E04FDD                         DCB   0
__const:FFFFFFF006E04FDE                         DCB   0
__const:FFFFFFF006E04FDF                         DCB   0
__const:FFFFFFF006E04FE0                         DCB   0
__const:FFFFFFF006E04FE1                         DCB   0
__const:FFFFFFF006E04FE2                         DCB   0
__const:FFFFFFF006E04FE3                         DCB   0
__const:FFFFFFF006E04FE4                         DCB   0
__const:FFFFFFF006E04FE5                         DCB   0
__const:FFFFFFF006E04FE6                         DCB   0
__const:FFFFFFF006E04FE7                         DCB   0
__const:FFFFFFF006E04FE8                         DCB 0x18
__const:FFFFFFF006E04FE9                         DCB 0x47
__const:FFFFFFF006E04FEA                         DCB 0x15
__const:FFFFFFF006E04FEB                         DCB   6
__const:FFFFFFF006E04FEC                         DCB 0xF0
__const:FFFFFFF006E04FED                         DCB 0xFF
```



```
__const:FFFFFFF006E04FC8                         DCQ unk_FFFFFFF0076DC0F0
__const:FFFFFFF006E04FD0                         DCQ unk_FFFFFFF0076DC2B8
__const:FFFFFFF006E04FD8 off_FFFFFFF006E04FD8    DCQ 0                  ; DATA XREF: com.app
__const:FFFFFFF006E04FD8                                                ; com.apple.iokit.IC
__const:FFFFFFF006E04FE0                         DCQ 0
__const:FFFFFFF006E04FE8                         DCQ sub_FFFFFFF006154718
__const:FFFFFFF006E04FF0                         DCQ sub_FFFFFFF00615471C
__const:FFFFFFF006E04FF8                         DCQ __ZNK8OSObject7releaseEi ; OSObject::rel
__const:FFFFFFF006E05000                         DCQ __ZNK8OSObject14getRetainCountEv ; OSObj
__const:FFFFFFF006E05008                         DCQ __ZNK8OSObject6retainEv ; OSObject::reta
__const:FFFFFFF006E05010                         DCQ __ZNK8OSObject7releaseEv ; OSObject::rel
__const:FFFFFFF006E05018                         DCQ __ZNK8OSObject9serializeEP11OSSerialize
__const:FFFFFFF006E05020                         DCQ sub_FFFFFFF006154734
__const:FFFFFFF006E05028                         DCQ __ZNK15OSMetaClassBase9isEqualToEPKS_ ;
__const:FFFFFFF006E05030                         DCQ __ZNK8OSObject12taggedRetainEPKv ; OSObj
__const:FFFFFFF006E05038                         DCQ __ZNK8OSObject13taggedReleaseEPKv ; OSOb
__const:FFFFFFF006E05040                         DCQ __ZNK8OSObject13taggedReleaseEPKvi ; OSO
__const:FFFFFFF006E05048                         DCQ __ZN8OSObject4initEv ; OSObject::init(vo
__const:FFFFFFF006E05050                         DCQ sub_FFFFFFF006154E68
```

VTable recognition

- iOS Step 2: find address of class's global metaclass object in initialization function of __mod_init_func
  - In Apple driver, most C++ classes have a global metaclass object describing the class's basic information

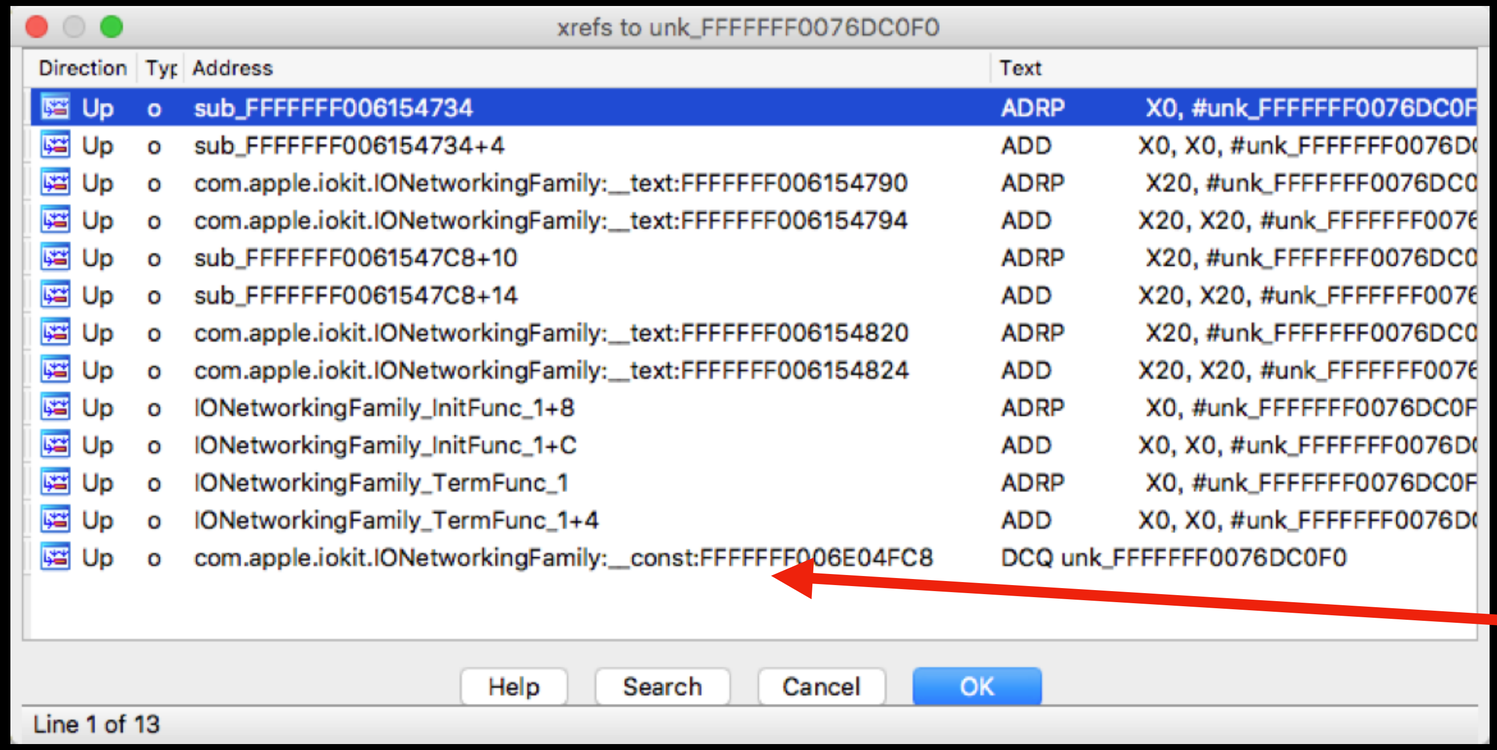
Address of class's global metaclass object

```
_QWORD *IONetworkingFamily_InitFunc_1()
{
  _QWORD *result; // x0

  result = (_QWORD *)sub_FFFFFFF006166E44(&unk_FFFFFFF0076DC0F0, aIoethernetinte, &unk_FFFFFFF0076DC2B8, 328LL);
  *result = &unk_FFFFFFF006E056E0;
  return result;
}
```

Initialization function of __mod_init_func

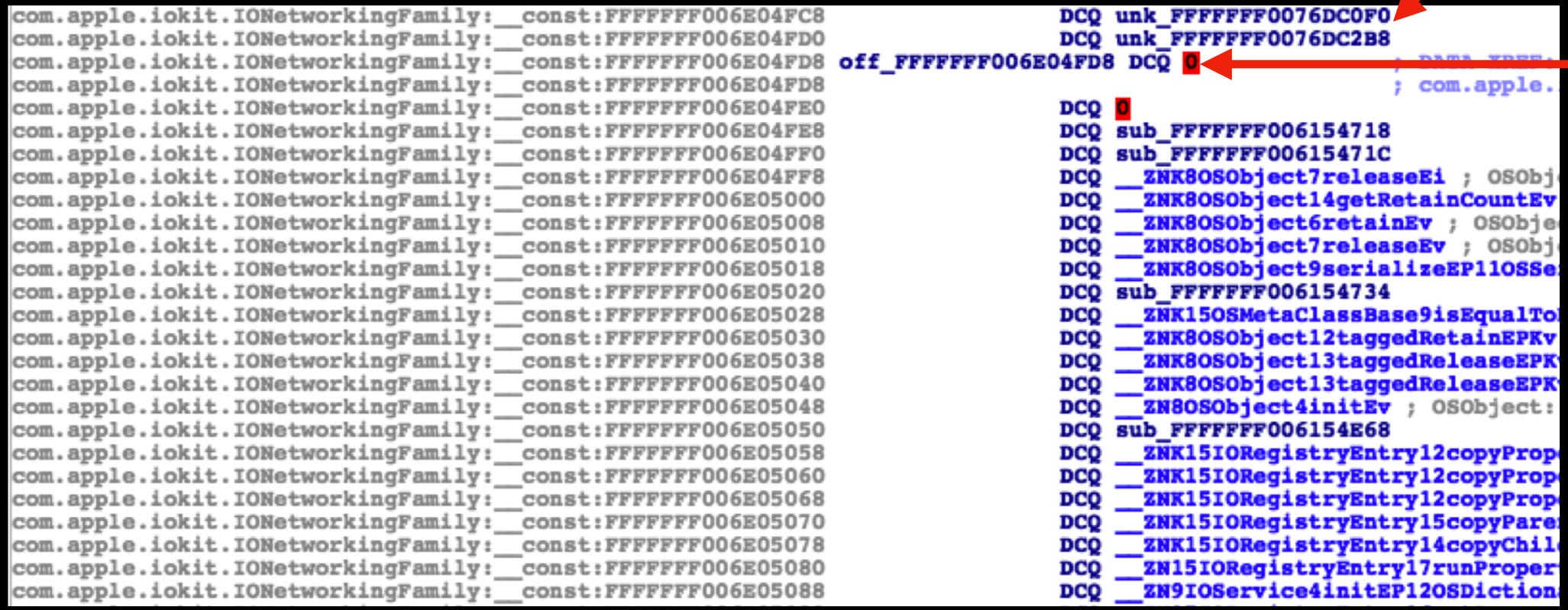
VTable recognition

- iOS Step 3: check cross references of the global metaclass object, find the one in const section and near the VTable



VTable recognition

- iOS Step 4: found VTable



VTable recognition

- After recognition, 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

```
vtable_IOEthernetInterface struc ; (sizeof=0x6E0, mappedto_5666)
                                 ; XREF: whole_vtable_IOEthernet
                                 ; com.apple.iokit.IONetworkingF
__ZN19IOEthernetInterfaceD1Ev DCQ ?     ; 0xffffff006154718L
__ZN19IOEthernetInterfaceD0Ev DCQ ?     ; 0xffffff00615471cL
__ZNK8OSObject7releaseEi DCQ ?          ; 0xffffff0074f8644L
__ZNK8OSObject14getRetainCountEv DCQ ?  ; 0xffffff0074f8658L
__ZNK8OSObject6retainEv DCQ ?           ; 0xffffff0074f8660L
__ZNK8OSObject7releaseEv DCQ ?          ; 0xffffff0074f8670L
__ZNK8OSObject9serializeEP11OSSerialize DCQ ? ; 0xffffff0074f8680L
__ZNK19IOEthernetInterface12getMetaClassEv DCQ ? ; 0xffffff006154734L
__ZNK15OSMetaClassBase9isEqualToEPKS_ DCQ ? ; 0xffffff0074f63e0L
__ZNK8OSObject12taggedRetainEPKv DCQ ?  ; 0xffffff0074f8768L
__ZNK8OSObject13taggedReleaseEPKv DCQ ? ; 0xffffff0074f87fcL
__ZNK8OSObject13taggedReleaseEPKvi DCQ ? ; 0xffffff0074f880cL
__ZN8OSObject4initEv DCQ ?              ; 0xffffff0074f88f4L
__ZN19IOEthernetInterface4freeEv DCQ ?  ; 0xffffff006154e68L
```

VTable structure

```
IOEthernetInterface struc
vtable              DCQ ?
member1             DCQ ?
member2             DCQ ?
member3             DCQ ?
member4             DCQ ?
member5             DCQ ?
member6             DCQ ?
member7             DCQ ?
member8             DCQ ?
member9             DCQ ?
member10            DCQ ?
member11            DCQ ?
member12            DCQ ?
member13            DCQ ?
member14            DCQ ?
```

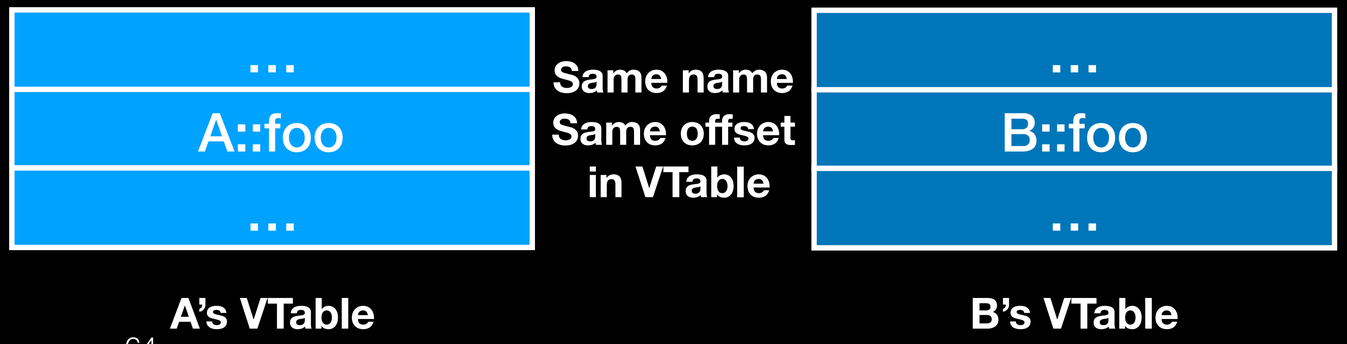
Class structure

Recover function names (only for iOS)

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

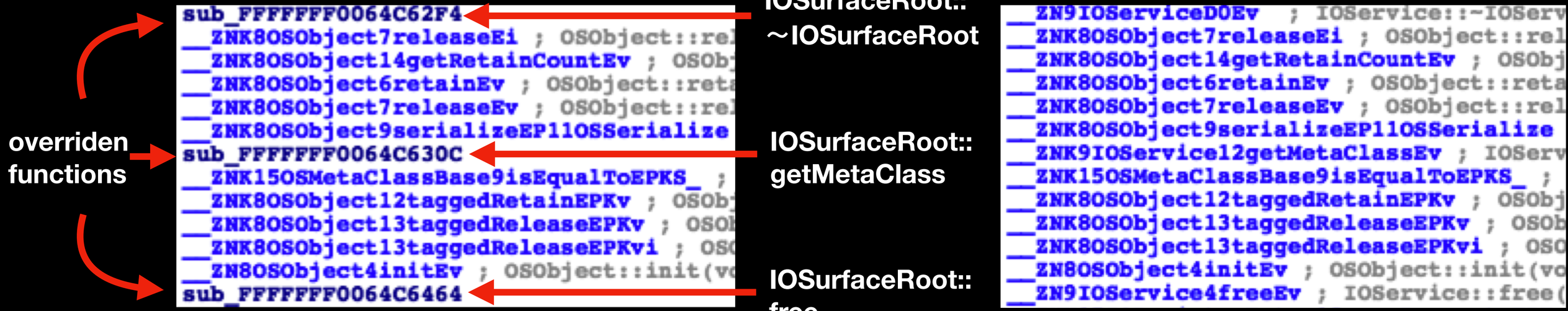
```
class A {
  public:
  virtual void foo();
}
```

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



Recover function names (only for iOS)

- Only recover the overridden virtual functions in classes inherited from classes in the kernel
- Not a complete solution, but cover plenty of critical functions

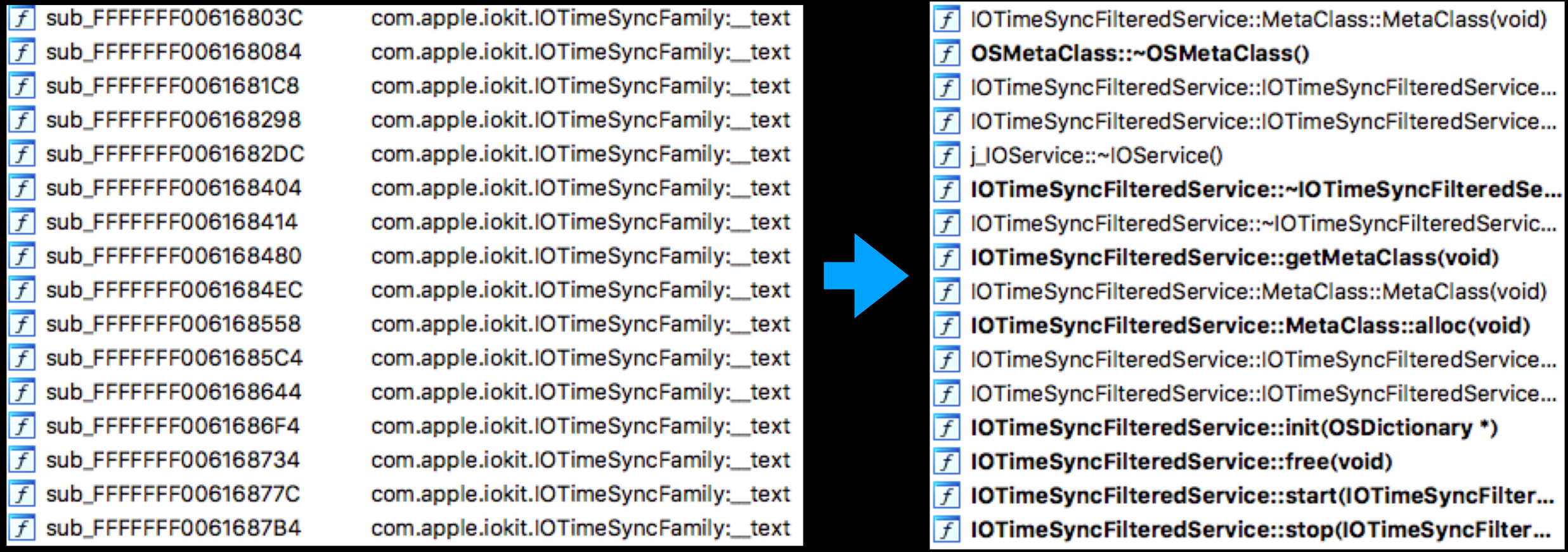


IOSurfaceRoot's VTable
(IOSurfaceRoot is inherited from IOSerivce)

IOService' VTable

Recover function names (only for iOS)

- Many function names can be recovered in this way



Original **Now**

Resolve variable types (in IDA pro’s decompiled code)

- For local, global and member variables
- Method: identify variable types and perform type propagation
- How to identify variable types: two ways
  - 1. Depend on features of function names in C++
  - 2. Figure out the creation and iniatialization of variables

Resolve variable types (in IDA pro's decompiled code)

- Identify variable types method 1:
  - After compilation, names of functions in C++ classes will be encoded with the function name and argument types
  - Decode (demangle) the C++ function names to get the type of function arguments, e.g.

__ZN23IOSurfaceRootUserClient4initEP13IOSurfaceRootP4taskP12OSDictionary

Decode (Demangle)

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

Variable Types

Resolve variable types (in IDA pro's decompiled code)

- Identify variable types method 2:
  - Figure out the creation and iniatialization of variables
  - Find invocation of variable allocation, initialization and type casting functions, 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 IDA pro's decompiled code)

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

```
IOAccelShared2 *__cdecl IOGraphicsAccelerator2::newShared(IOGraphicsAccelerator2 *this)
{
  unsigned __int64 v1; // rsi@0
  IOAccelShared2 *v2; // rbx@1

  v2 = (IOAccelShared2 *)OSObject::operator new(&stru_108, v1);
  IOAccelShared2::IOAccelShared2(v2);
```

Type of v2 is IOAccelShared2 *

```
__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),
```

Type of v2 is IOGraphicsAccelerator2 *

Resolve variable types (in IDA pro's decompiled code)

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

```
IOAccelShared2 *__cdecl IOGraphicsAccelerator2::newShared(IOGraphicsAccelerator2 *this)
{
  unsigned __int64 v1; // rsi@0
  IOAccelShared2 *v2; // rbx@1

  v2 = (IOAccelShared2 *)OSObject::operator new(&stru_108, v1);
  IOAccelShared2::IOAccelShared2(v2);
  return v2;
}
```

Propagate v2's type and affect the return type of function

```
__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) )
```

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

Add cross references:

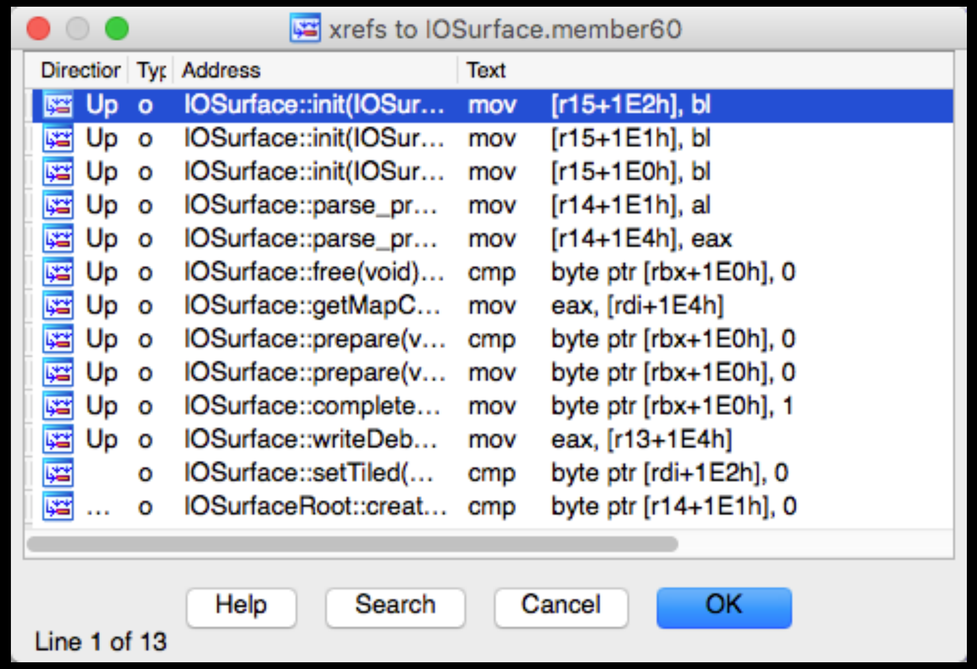
- Purpose: add cross references for member varaibles and virtual functions
- Method: examine every sentence in the decompiled code of all functions, check whether member variable or virtual function is used, e.g.

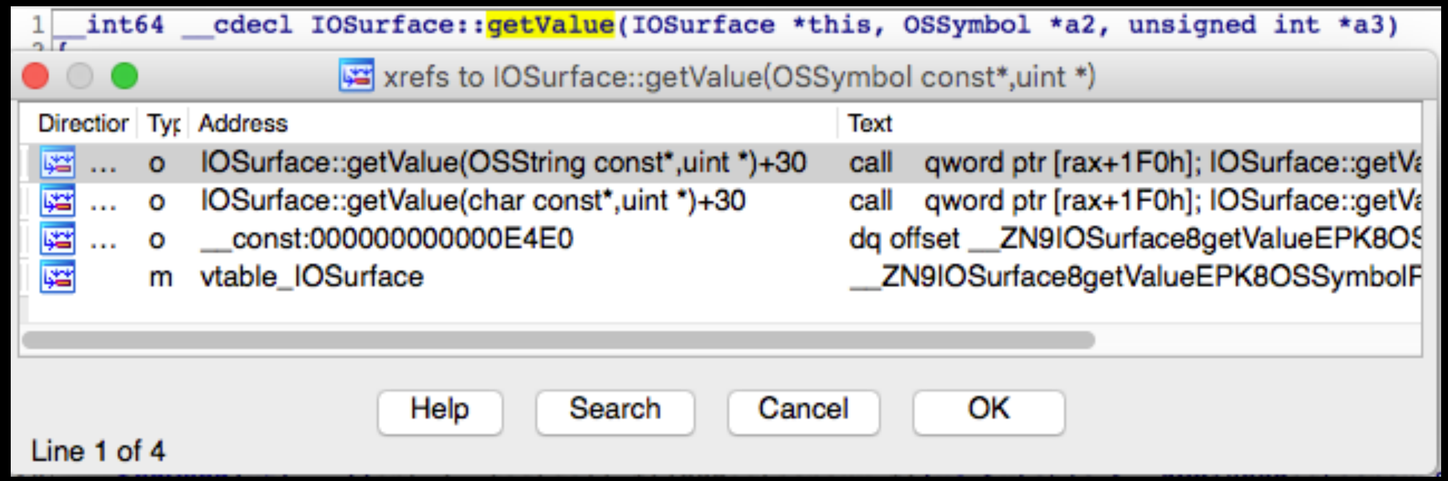
```
__int64 __cdecl IOSurface::setTiled(IOSurface *this, bool a2)
{
  __int64 result; // rax@1

  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:

- References are added from the usage to members in the class structures and to virtual function's implementation





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

- Looks more like source code, right?

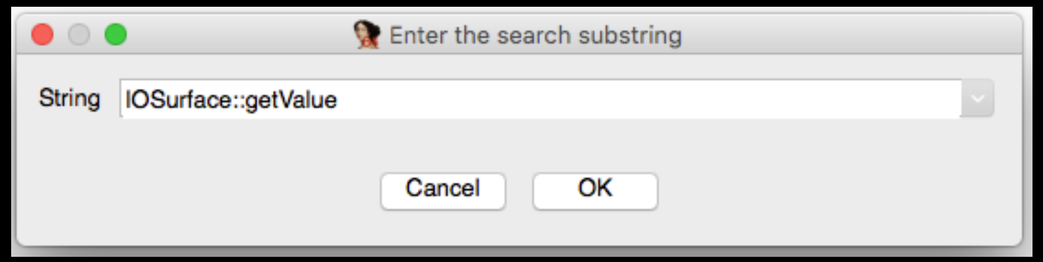
```
__int64 __cdecl IOSurface::getValue(IOSurface *this, char *a2, unsigned int *a3)
{
  unsigned int *v3; // r14@1
  OSSymbol *v4; // rbx@2
  __int64 v5; // r14@4

  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;
}
```

But, for manual review in IDA pro's decompiled code, though virtual function calls are recognized and identified with the help of above features, researchers still need more manual operation to examine the function's implementation. e.g.

Manual operation:

```
__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;
}
```

Step 1: Copy name of the called function



Step 2: Search the function name

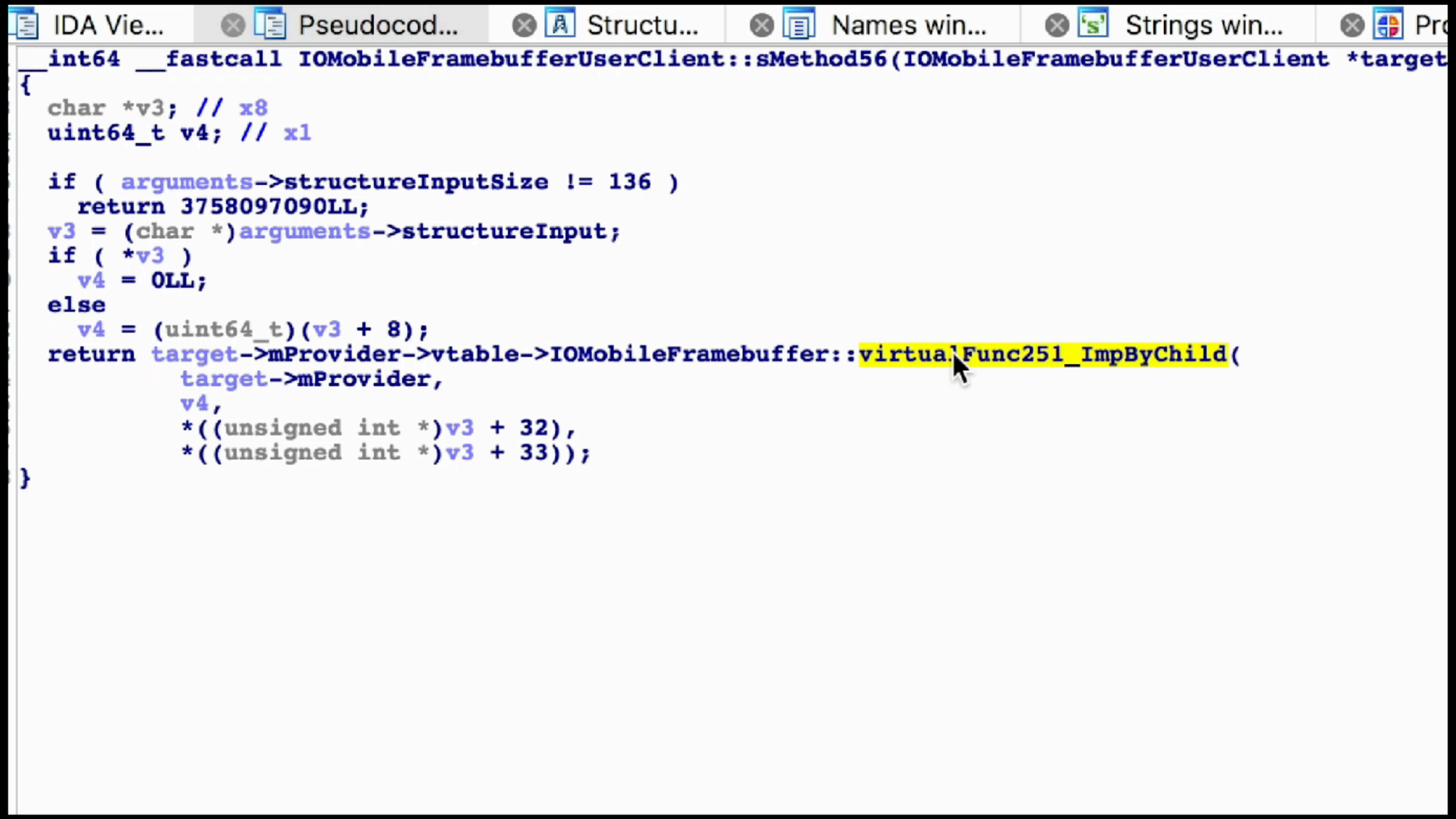
UI support

- Extend UI operation that can be performed in IDA pro’s decompiled code
- 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

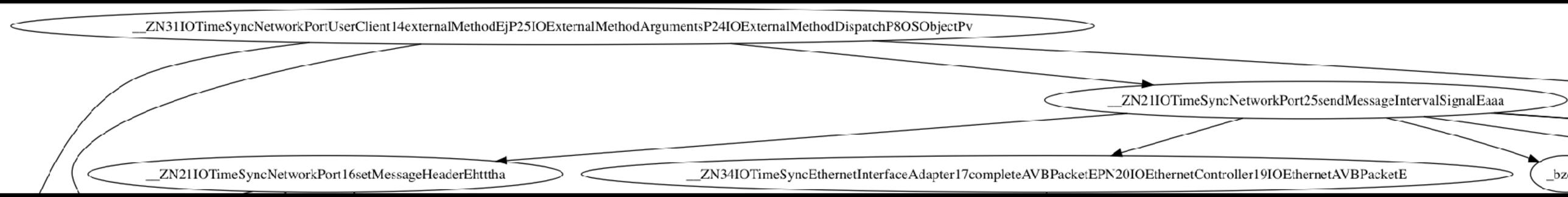
- Extend UI operation that can be performed in IDA pro’s decompiled code, e.g. jump to virtual function’s implementation by just a double click
- Method:
  - Register action to double-click events
  - Register action to key events
  - Register action to name change events
  - Register action to type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

UI support


IDA Vie...
Pseudocod...
Structu...
Names win...
Strings win...
__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-procedure 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

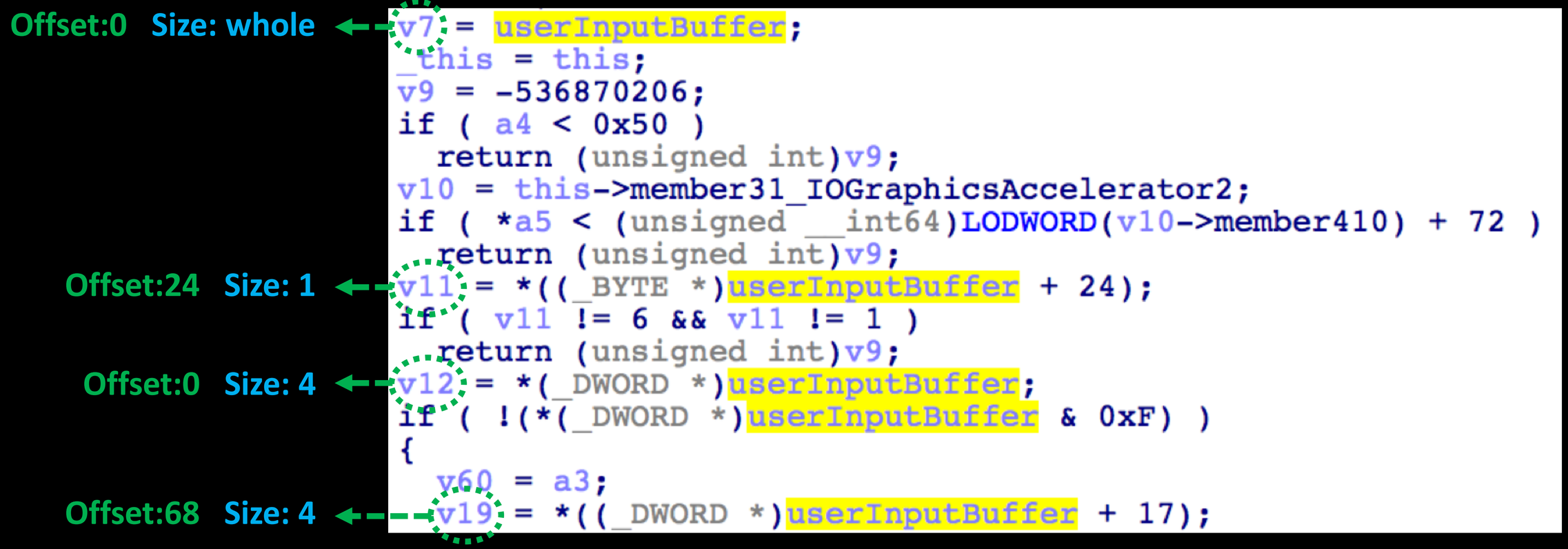
```
__int64 __cdecl IOSurface::getValue(IOSurface *this, OSSymbol *a2, unsigned int *a3)
{
  unsigned int *v3; // rbx@1
  OSDictionary *v4; // rbx@1
  OSDictionary *v5; // rax@4

  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 fuzzing of drivers with Ryuk's static analysis
- 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 to infer user input formats



Ryuk-Fuzz Step 1: Static data flow analysis to infer user input formats

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

```
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);
```

Ryuk-Fuzz Step 2: Guide fuzzing

- BUT ! PassiveFuzzFrameworkOSX has implementation 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	5F5F4C494E4B45544954000…	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	5F5F4C494E4B45544954000…	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

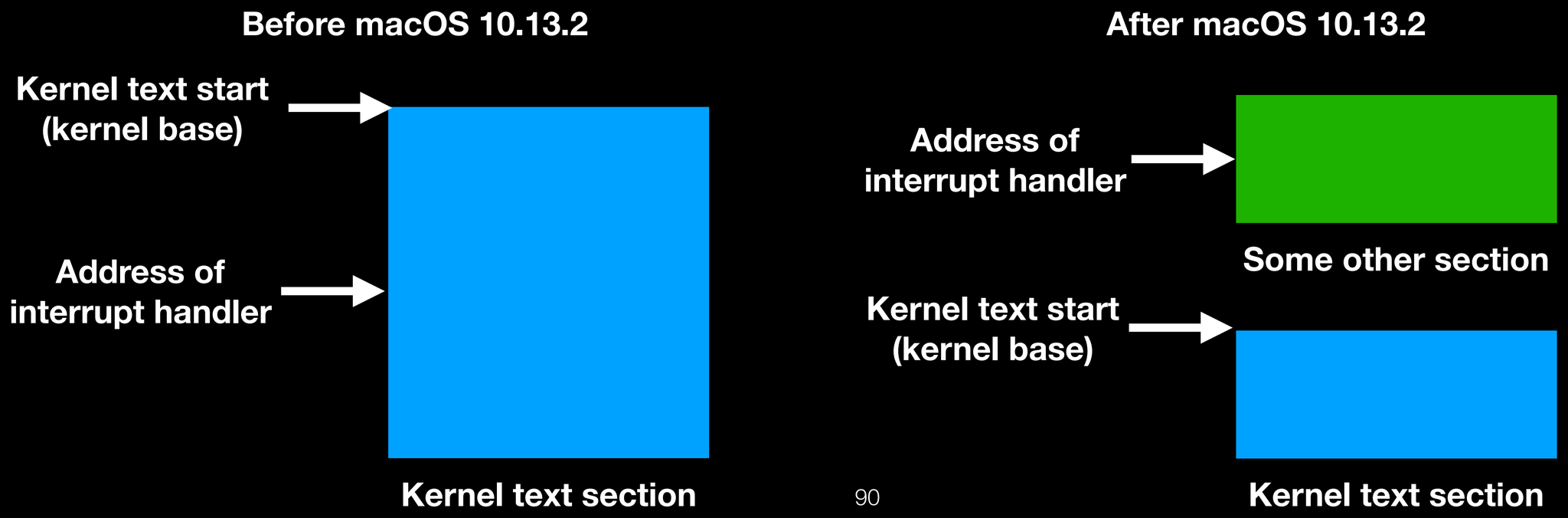
- Error still exists in PassiveFuzzFrameworkOSX
- Error 2: Wrong way to infer kernel base at runtime (affect 10.13.2 and after)

```
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, interrupt 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

Q&A